Università degli studi Roma Tre
Dipartimento di Ingegneria,
Sezione Elettronica Applicata

Semantic processing of multimedia data and applications

Federico Colangelo

Abstract

Nowadays systems able to process multimedia data automatically are of primary interest. As a matter of fact, volumes of data that cannot be handled by human operators are now generated everyday. Managing this kind of data becomes increasingly difficult when it must be processed according to human-level attributes. In fact, understanding human perception is a complex task and it is difficult to implement algorithms able to extract human-level attributes from multimedia data. Since many of such attributes lack an analytical definition, it is desirable to leverage how they emerge from the raw data in a data-driven fashion, mining association between raw values and high-level attributes. However, multimedia data is characterized by high dimensionality. This feature causes numerical problems that hinder the performances of many state-of-the-art techniques. Novel models are thus being developed for the estimation of human-level attributes from multimedia data and, consequently, new attributes can be leveraged to perform complex tasks.

In this context, the two most important research questions that we attempt to answer are how to estimate the semantic attributes of multimedia content and how to leverage them for building systems that can assist humans in harnessing the stream of data. In this thesis, the first question is addressed in the domain of audio surveillance. This task deals with the detection and classification of selected audio events in contexts characterized by non-relevant, background events as well as unstructured noise, with critical constraints on false rejection rate for the detection phase. A novel model for the classification and detection of critical audio events is proposed, based on the aforementioned requirements. The second question is addressed in the visual domain, more

specifically in the context of unsupervised video orchestration. Here, a method to combine different types of high-level attributes in order to enhance the quality of viewers' experience is shown. More specifically, the proposed method leverages frame-based aesthetic values estimation, as well as automatic estimation of the quality of camera changes through a Markov model, combined through a multi-objective optimization algorithm. In both cases, the proposed methods show satisfying results, contributing to the growing field of semantic data processing.

Acknowledgements

Questi tre anni si sono rivelati ricchi di sfide, sul piano professionale e personale. Molte di queste sfide sono state al limite di quello che ero in grado di fare, se non oltre. Se non mi sono schiantato completamente in questo percorso, è in gran parte grazie alle persone che ho avuto intorno, come mentori, familiari ed amici. Sono solo poche parole, che rappresentano la punta dell'iceberg del contributo che avete dato alla mia vita. Nonostante questo, voglio ringraziarvi qui.

Per primo ringrazio il mio supervisore, il prof. Alessandro Neri. Grazie soprattutto per avermi voluto dare un'occasione in un momento del mio percorso di studi in cui in pochi lo avrebbero fatto. Grazie per tutti gli insegnamenti, troppi per citarli, di questi anni. In special modo, grazie per la capacità di vedere sempre lo scenario pi grande (cosa che sto ancora apprendendo). Grazie prof. Marco Carli. Anche se a più riprese non siamo andati d'accordo, da quei confronti ho imparato (spesso malvolentieri) più di quanto non vorrei ammettere. Devo molto alla tua schiettezza. Alla prof. Federica Battisti va un enorme ringraziamento, per tutto quello che mi ha insegnato in questi anni. È stata ed una guida nel mondo dell'università ed ha dato uno dei contributi pi grandi alla mia sopravvivenza in questi tre anni, specialmente nei momenti più disorientanti.

Una delle cose migliori che ho compreso davvero in questi anni il valore della mia famiglia. Ci sono voluti anni, ma ho finalmente capito quanto sono fortunato. Grazie a tutti voi, dal più profondo. Grazie a papà per le chiacchierate ed i consigli che mi hanno fatto crescere, diventare pi simile alla persona che un giorno spero di essere. Grazie a mamma, per esserti avvicinata a me nei

momenti in cui più ne avevo bisogno. Grazie a mia sorella, per aver riso sempre assieme a me. Grazie a zia Teresa, zio Salvatore ed Aldo, Claudia e Valentina, Liberato ed al nuovo arrivato, Lorenzo. Ogni volta che torno, non importa dopo quanto, trovo sempre un sorriso che più di ogni altra cosa significa casa. Grazie a tutti voi a Roma: ci vediamo poco, meno di quanto vorrei, eppure siete sempre capaci di farmi sorridere, in ogni occasione. Ed infine grazie a tutti gli spoletini: ci siamo ritrovati da poco e nonostante questo vi sento vicini.

Grazie Noemi. Grazie infinite, per tutto quello che hai portato e porti nella mia vita. Per quello che mi hai insegnato e per i momenti in cui mi sei stata vicina e per tutte le volte che mi hai sopportato. Chi mi conosce sa che non sono una persona facilissima a cui stare accanto, specialmente nei momenti pi complicati grazie per aver trovato una parte di me che nessuno (incluso me) conosceva, grazie per tutti i nostri momenti. Grazie anche alla tua famiglia, che mi ha accolto calorosamente.

Ai miei amici devo molto. È molto difficile raccogliere tutti voi in un singolo paragrafo. La mia amicizia con ciascuno di voi piè diversa ed ognuna porta qualcosa di differente. Mi preme particolarmente citare due grandi categorie però, perdonatemi se non vi nomino e non rendo giustizia a tutti.

Grazie a voi che mi avete sostenuto, ascoltato e consigliato nei momenti bui. A più riprese questo percorso si è rivelato incredibilmente difficile, in un modo che non riesco a descrivere qui. Aver trovato qualcuno nei momenti più bui è stata una delle mie fortune più grandi.

Grazie a voi, con cui ho condiviso momenti allegri, stupidi e senza pensieri. Non è facile spiegare il valore di quei momenti, oltre l'allegria e le risa mi hanno ricordato quanto ci fosse al di fuori del dottorato, che in certi momenti sembrava essere l'unica, titanica componente della mia vita.

A molti di quelli che ho incontrato durante questi anni devo dei ringraziamenti e forse anche delle scuse. Sono sicuro che avremmo condiviso molte risate in più se fossi riuscito ad gestire meglio lo stress da lavoro in questi anni. Ci rifaremo.

Molte altre persone meritano un ringraziamento. Grazie ad esempio a Raffaella, Enrico, Annalisa, Gemma e Laura (e, più recentemente, Marzia) per avermi aiutato ad arginare i disastri di documentazione che sono così bravo a combinare. Grazie a tutte le persone di Roma Tre con cui ogni tanto chiacchiero: rendete questa università un posto pi bello (e decisamente più umano).

Più rileggo questi ringraziamenti più mi rendo conto di quanto siano incompleti, incoerenti ed a tratti melensi. Mi perdoneranno tutti quelli a cui non ho reso giustizia ma che hanno contribuito al raggiungimento di questo giorno. In ogni caso, grazie a tutti voi per avermi aiutato ad arrivare qui. Spero di condividere ancora strada con tutti voi nei miei percorsi futuri.

Contents

List of Tables

List of Figures

List of Acronyms

ASR Automatic Speech Recognition

BN Batch Normalization

CNN Convolutional Neural Network

DCASE Detection and Classification of Acoustic Scenes and Events

DFT Discrete Fourier Transform

DNN Deep Neural Network

DSS Decision Support System

FRR False Rejection Rate

GA Genetic Algorithm

GAN Generative Adversarial Network

GMM Gaussian Mixture Model

HMM Hidden Markov Model

HSV Hue Saturation Value

LSTM Long Short Term Memory

ME Motion Estimation

MFCC Mel Frequency Cepstral Component

OF Optical Flow

PCM Pulse Code Modulation

RGB Red Green Blue

RNN Recurrent Neural Network

SGD Stochastic Gradient Descent

SNR Signal to Noise Ratio

STFT Short-Time Fourier Transform

SVM Support Vector Machine

TTS Text-To-Speech

VTLP Vocal Tract Length Perturbation

ZCR Zero Crossing Rate

Chapter 1

Introduction

In recent years, data generation has increased exponentially. According to Eric Schmidt, Google CEO, the amount of information created between the dawn of civilization and 2003 (5 exabytes) is being created every two days, as of 2011. The progressive availability of cheap acquisition hardware (e.g. smartphones, cameras) as well as storage space has certainly played a key role in this phenomenon.

While multimedia data constitutes an important portion of this data, IoT (Internet of Things) devices are also producing a large amount of data. According to Gartner, within the next three years there will be a massive increase of Internet of Things devices connected to the internet (i.e. estimated growth from 8.4 billions in 2017 to 20.4 billions in 2020).

Nevertheless, it is progressively unfeasible for human beings to process the

data stream without the help of automated techniques.

The term Decision Support System (DSS) has been in use since the 70's to describe techniques for the assistance of human operators, especially in fields in which the dimensionality of the problem cannot be handled through human means. In particular, data-driven DSS defines systems based on the extraction of knowledge from large scale analysis of data. The major limits of current DSSs lie in the type of data that can be handled and in the type of attributes that can be estimated. These techniques have been successfully applied in scenarios characterized by simple input data (e.g. input variables that are either qualitative or of limited dimensionality) such as demographics and market basket analysis. However, with more complex data, such as in recommendation systems for multimedia content, the DSS relies on input simplification (e.g. genre tags in music) defined and selected by human operators. It is therefore evident that, in order to effectively use this large amount of data, better DSSs are needed and that they must be able to handle high-dimensional data and to estimate abstract properties from it. To better characterize the concept of abstract attributes, as opposed to low-level properties, the terms semantic and syntactic are often used. They are both inherited from the linguistics field, where semantic refers to the meaning while syntactic refers to the structure of the language.

Thus, in multimedia, we refer to semantic attributes to describe the content of data as opposed to syntactic attributes to describe the structure of data. In the following Section the issue of estimating semantic attributes, especially in

multimedia content, is described.

1.0.1 Semantic processing of data

Semantic features are intrinsically difficult to estimate, since they lack an analytical form. In general, complex attributes emerge from the combination of simpler ones. This is true also within the set of semantic attributes. For example, if we consider the picture of a street from the point of view of a self-driving car, not only must significant objects be recognized (e.g. pedestrians, cars, road signs), but also their position in space must be used to infer a higher-level attribute, e.g. if it safe to accelerate. Semantic meaning can thus emerge at different scales, usually depending on the abstraction level of the attribute itself.

Since it is not possible to formulate an analytical rule that defines how the specific pattern of an attribute expresses a semantic concept, the estimation of the decision rule must be inferred automatically from the data. This task is nowadays performed by means of machine learning algorithms. Machine learning concerns algorithms that are not programmed to perform a specific task, but rather *learn* to do so in a data-driven fashion. Learning a task consists in estimating a function that associates a specific output to a particular configuration of inputs. While these techniques are a promising solution to the problem of estimating semantic attributes, practical limitations, described in Chapter 2, prevent inference in semantic tasks. More specifically, the most

significant limitation lies in the complexity of the function that can be learned. More complex functions enable a wider abstraction gap between the input data and the estimated attribute. Therefore, in order to effectively estimate semantic attributes from syntactic properties or from the data itself, new machine learning models are needed.

Considerable advancements have been recently introduced in the field of machine learning. One of the key factors in this evolution is Deep Learning, a term that describes the recent techniques in the field of neural networks. Deep learning techniques infer complex features hierarchically, starting from a low-level representation of the data. Deep learning has been applied in several different domains such as in image classification [1] and generative modeling of multimedia content [2] [3], outperforming state-of-the-art algorithms.

To summarize, the availability of data and of modern machine learning models, such as deep learning, motivates us to develop and adopt new approaches for processing multimedia content on a semantic level. While in the more established domains of machine learning, such as image classification and speech recognition, deep learning represents the new standard, other areas of multimedia currently offer good opportunities to use these techniques. In many fields, however, additional constraints are present, together with achieving the best possible accuracy, such as:

- Timeliness of the recognition (i.e. delay due to processing)
- Computational complexity

- Need of minimizing false negatives or false positives
- Robustness of the system to noise
- Robustness to attackers

The applications of modern machine learning algorithms in these framework still need to be deeply investigated. In this context, the contribution of this thesis is two-fold. First, new models for the classification of multimedia data according to semantic attributes are defined. These models can be used in many different contexts, such as tagging specific portions of a multimedia element with labels (i.e. content enrichment). Such data can then be leveraged for semantic-based data processing (e.g. multimedia data retrieval, machine intelligence). This problem is studied in the audio domain and, more specifically, in the context of smart audio surveillance systems. This application is especially challenging, given the constraints on classification and detection accuracy, as well as the system's efficiency.

In this sense, the most important contribution is presented in Section 4.4, where it is shown how techniques of machine learning, and more specifically deep learning, can be leveraged to build a low-delay detector and classifier of critical audio events.

Second, novel techniques for combining multimedia data based on semantic attributes are studied. As more and more information becomes available thanks to the algorithms studied in this thesis, the problem of leveraging and combining the information in a coherent and efficient manner arises. In the context

of smart surveillance, semantic information could be used to combine video and audio information coming from different sensors in order to obtain a single video summarizing the events that have taken place. In this sense, many criteria should influence the final product: it should give priority to sources containing anomalous events while maintaining an overall vision of the whole monitored area in order to give security operators a good situational awareness. These types of constraints are common in multimedia orchestration tasks and can be more severe depending on the context. A particularly challenging scenario is orchestrating amateur videos of an event (e.g. concert) since the output video must also provide a good quality of experience, which, in turn, is influenced by multiple semantic attributes, interacting in a non-trivial manner. In this thesis, algorithms for leveraging and combining data are developed in the context of video orchestration for amateur videos. In this framework, Section 5.4 introduces an automatic video orchestration system based on the automatic estimation of high-level semantic concepts, such as aesthetics, combined through multi-objective optimization. It is also shown how the evaluation of these systems can be used as a knowledge discovery tool for mining higher-level influences between semantic attributes. The thesis is structured as follows: an introduction to machine learning and its most important techniques is given in Chapter 2, while Chapter 3 introduces deep learning, presenting the state-of-the-art in terms of models and techniques for training deep learning models. Chapter 4 introduces the audio domain and reviews the state-of-the-art concerning audio machine learning. Experimental results within the domain of

audio surveillance are contextually presented.

Chapter 5 introduces the images and video domain. Applications of machine learning techniques to produce an automatic video orchestration system are presented.

Finally, the conclusions are drawn in Chapter 6.

Chapter 2

Machine learning fundamentals

2.1 Fundamentals of machine learning

The purpose of a machine learning system is to learn to make predictions about given attributes of a data instance, such as the content of a picture or the evolution of the stock market. The value that is being predicted is called target and, in order to predict it, a dependence between such value and some available data is assumed (e.g. the dependency between the future values of a stock and the past ones). This is why the quantities used to predict the target are called explanatory variables (i.e. they *explain* a certain property of the data), albeit in practice the term features is often used, as will be shown in Section 2.2.

The ensemble of data instances used to learn the relations between the ex-

planatory variables and the target are called training set. The selection of the explanatory variables plays a critical role in the success of the learning phase. On the one hand, it is desirable to have a larger number of features to train the model, as this gives more information and thus better explanatory power to the model itself. On the other hand, the model complexity needed to handle more features introduces significant obstacles in the learning process. In detail, the number of possible configurations of the model's parameters grows exponentially with the number of parameters influencing not only the computational complexity of the learning, but also the ability to find an effective solution to the problem. This phenomenon is known as *the curse of dimensionality*.

Having selected the training data, it is necessary to select the model, as well as the number of its parameters, which has another key effect on the success of the learning process. As a matter of fact, a model that lacks a sufficient number of parameters could prove unable to learn the relations between the input and the target. This phenomenon is called under-fitting.

Conversely, when too many parameters are used, the model tends to learn a mapping for each observation in the training set, failing to learn the general relations between the input and the target. This phenomenon is known as over-fitting. Moreover, reducing model parameters, over-fitting can be mitigated by increasing the number of observations in training data and by imposing constraints during the learning phase. These techniques are called regularization.

In general, since the model can over-fit the training data, a tool to measure the generality of the function learned by the model is needed. For this reason, a

second set of observations of the training variables are collected, but not used for the training. Instead, this set of observations, known as testing data, is used to benchmark the generalization capabilities of the model.

Machine learning algorithms are categorized according to the training data used and the type of target variable. The first categorization regards the exact form of the training data. Supervised learning algorithms learn by being exposed to observation from the training data and the corresponding ground truth target value, called label.

Unsupervised learning algorithms are exposed only to the training data, with the general aim of finding similarities in it. Finally, reinforcement learning algorithms are trained by exposing the model (often called agent) to the state of the environment and a set of actions with an associated reward.

The second attribute that describes a learning algorithm is the type of target variable.

Classification algorithms aim to predict an integer value, representing the presence of one of the classes considered in the observation (e.g. determine if the input image has a cat, a dog or neither of two in it). However, since encoding the classes as successive integers would introduce a spurious ordinal relation (e.g. class 2 would result closer to class 3 than to class 5), it is common practice to use the one-hot encoding technique for target vectors. A one-hot encoded target vector, representing the i_{th} class out of m is an m dimensional vector composed entirely of zeros except for the i_{th} element. As a result, no unintentional ordinal relation between classes is introduced.

Classification algorithms can be further labeled as multi-class (if the target value is non-binary) and multi-label (if more than one class can be present at a time, e.g. a cat, a dog or both).

Regression algorithms predict continuous values for each observation (e.g. predict the future value of a stock). Generally, the output of regression tasks are real numbers.

Generative algorithms are different from regression and classification, as their aim is not to predict values in the stricter sense. Generative models are trained to create new instances of a certain quantity (e.g. generate images of cats). Trained models typically take noise as input and use it as a sort of seed for the trained generator.

Other types of algorithmic attributes are sometimes used. As an example, clustering refers to the problem of predicting a cluster value for input based on unsupervised learning methods. However, these problems can usually be casted into a more general supervised/unsupervised classification/regression framework.

In order to train models, a distance function between the current model output y_c and the target ground truth y_t must be calculated. In regression problems this is straightforward, as generally both y_c and y_t are real number, and their distance can be measured by simple approaches such as the squared difference. In classification problems, this is often achieved by normalizing the output of the model to resemble a probability distribution (i.e outputs into the [0 1] interval that sums to 1) with a function such as the softmax, described in Section

3.2.1. In this way, the current output y_c is interpreted as a probability distribution over the possible classes, P_c. Probability distribution distance measure can thus be used to measure the difference between model output and ground truth, such as the Kullback-Leibler divergence, D_{KL} between two probability distributions P_c and P_t, defined as:

$$D_{KL}(P_t||P_c) = \sum_i P_t(i) \log \frac{P_t(i)}{P_c(i)} \tag{2.1}$$

The divergence can be interpreted as the loss of information due to the approximation of P_t with P_c. In practice, the most used distance function is the cross-entropy between two probability distributions P_t and P_c is defined as:

$$H(P_t, P_c) = H(P_t) + D_{KL}(P_t||P_c) \tag{2.2}$$

where $H(P)$ is the Shannon entropy. However, given that P_t is a fixed quantity, the two measures are equivalent in practice. Since the outputs of the models are normalized, the numerical result obtained in the output y_c are used as a measure of the belief of the model about $\mathbf{x}$ belonging to a certain class.

The distance function is then used to optimize the model's parameters. As a matter of fact, in machine learning, to learn means to optimize the parameters of a model with respect to data in the training set. Optimization is performed by minimizing the distance function, also called error or loss function.

The properties of the loss function $\mathscr{L}$ strongly influence the convergence of the optimization algorithm. While it is generally difficult to determine if the

optimization procedure will converge, some machine learning algorithms are structured so that the loss function is easier to optimize. The most important distinction to be made is between convex and non-linear loss functions.

Convex functions are defined based on convex sets. A set X is said to be convex if, for any $x, y \in X$, given a scalar $\alpha \in [01]$, then $(1-\alpha)x + \alpha y \in X$. Given a convex set X a function $f : X \to R$ is said to be convex if, given an scalar $\alpha \in [0\ 1]$, $\forall x_1, x_2 \in X$

$$f(\alpha x_1 + (1-\alpha)x_2) \leq \alpha f(x_1) + (1+\alpha) f(x_2) \tag{2.3}$$

An important property of convex functions is that they admit only a single local minimum (that is therefore also the global one). Thus, simply updating the model's parameters in the direction in which the loss function gradient decreases guarantees finding the optimal solution of the problem. An example of a convex function is given in Figure 2.1, showing how the structure of the function does not admit local minima. The more general class of non-linear functions admit multiple local minima. Therefore, especially in high-dimensional spaces, it is generally not possible to recognize a local minimum from a the global minimum.

Another, indirect, optimization procedure is needed when developing a machine learning model. As previously cited, the number of parameters in the model influences the results of the optimization, regulating under-fitting and over-fitting. However, more complex models have other parameters that can also influence the learning process efficiency and the quality of the final result.

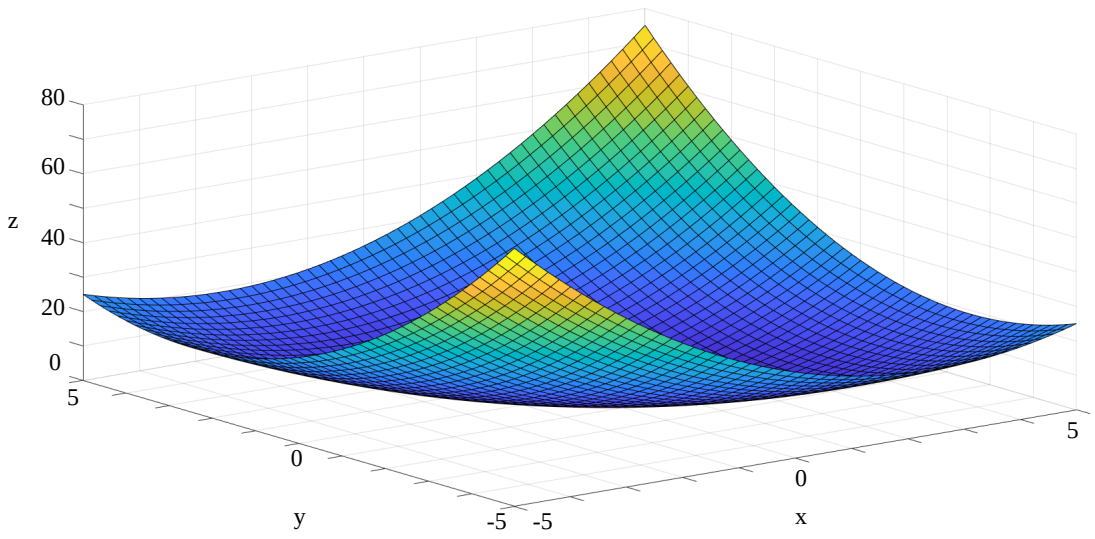


Figure 2.1: Example of convex function, given by $z = x^2 + xy + y^2$

These variables are called hyper-parameters. Hyper-parameters *tuning* is referred to as model validation.

Having defined the basic terminology, the problem of developing a machine learning system can be considered as two-fold.

Firstly, the type of data to be used must be selected and, for the chosen type of data, the representation must be selected. This process is described in Section 2.2.

Secondly, having selected the data, the algorithms must be chosen. This choice is dependent on the type of data and especially on their dimensionality. The main properties of machine learning algorithms are discussed in Section 2.3.

Furthermore, an overview of the major machine learning models is presented.

2.2 Selecting and preparing data

The selection and preparation of the data encompass feature's selection and data pre-processing. The pre-processing step is performed both before and after the feature extraction step, since it entails both cleaning the data and operations to improve the training as will be discussed in the following section.

2.2.1 Feature selection

Feature selection refers to the choosing of alternative representations for the input data, usually with the purpose of simplifying the learning process by reducing dimensionality and removing less informative data. Features are lossy representations, imposing a trade-off on information content against simplicity. The selection of features should thus be considered as part of the model selection problem, in the sense that the feasibility of the learning process depends directly on it.

Features can be characterized as syntactic or semantic. Syntactic features relate to the structure of the signal (as syntax describes the structure of sentences in linguistics) while semantic features relate to the content (as semantics is concerned with the meaning of sentences in linguistics). A similar categorization is done in multimedia, where features are characterized as low or high-level.

Low-level features are usually very close to physical properties of signal, while high-level ones are designed to capture concepts closer to human perception. High-level/semantic features are more difficult to estimate, as generally perceptual concepts cannot be expressed easily in analytical form. Low-level descriptors, on the other hand, can be defined and calculated reliably and can characterize the signal completely from a mathematical point of view. However, they are lackluster when tasked with replicating a human operator.

A hierarchical approach is usually adopted, that is building high-level representations of concepts as compositions of low-level features. Since the relations between these quantities cannot be estimated manually, machine learning algorithms are used. In general, it is desirable to feed the algorithms data as close as possible to the original representation. In this way, the model can discover other representations that were not discovered manually.

In this process, it is commonplace to attempt to use of domain-specific knowledge. More specifically, every domain (especially in multimedia) has some kind of ontology [1] that can be used in order to select features in relation to the high-level target variable. This procedure is widely used in the process of the selection and design of features. In the image domain, for example, it is common to design features to capture specific aspects of an image, such as borders [4] and textures [5]. Similarly, in the audio domain, features can be selected according to their ontological relation with low level spectral and temporal characteristics, as will be described in Chapter 4. However, this procedure is

[1] Here, the term ontology is intended as a set of definition and relationship between quantities in a specific domain of knowledge

not well defined and often relies on heuristics. This is due to the ontologies being poorly defined, aside from a restricted number of concepts. Despite the attempts described in literature [6] [7], the definition of a common ontology for features in multimedia is still an open problem.

Recently, Deep Neural Networks (DNN) have shown the ability to learn from minimally preprocessed input data, simultaneously outperforming state-of-the-art feature based approaches. DNN learn by composing complex features hierarchically, discovering non-linear relations between data at different levels of abstraction. This procedure can be considered opposite to the ontological-based selection described above. As a matter of fact, most deep models do not leverage domain-specific knowledge, but rather learn the associations from the input to the target variable. As it has been proven in literature, intermediate representations learned by DNN can be used as features with state-of-the-art performances [8] [9]. The first important results were obtained in image classification tasks where the architecture known as AlexNet [1] outperformed the state of the art on the ImageNet dataset [10], an image classification dataset composed of one thousand image classes with a hierarchical ontology. Since then, the use of deep learning has extended from multimedia classification to a variety of tasks including classification of very different data types (e.g. from malware detection [11], [12]) to generative models ([2], [3]).

Pre-processing

The term pre-processing encompasses *cleaning* the data as well as performing transformation that simplify the learning process.

Cleaning data refers to the removal of known artifacts not due to the phenomenon that is being analyzed. As an example, considering sensor readings, the hardware gathering data can impact the data in a systematic way for which a mathematical model is known. In this case, it is desirable to remove artifacts coming from the hardware by applying an inverse transformation. As with this example, pre-processing is in general dependent on the type of data considered as well as on the acquisition process noise.

Some practice applies to a more general case. However, it is worth to note that most of the following methods are actually heuristics: intuition can be obtained in toy problems and experimental results confirm the validity of the technique in higher dimension. Nevertheless, there are no formal proofs or rules that apply in these cases.

A common practice is to normalize the data into a unique interval by applying:

$$x' = \frac{x_0 - x_{min}}{x_{max} - x_{min}} \tag{2.4}$$

Where x_{min} and x_{max} are the boundary values of the new interval. Normalization to the [0 1] interval is a common practice. In general, given two quantities that possess different numerical ranges (e.g. height and income of a person), the differences in magnitude introduce spurious information into the system.

Reducing the magnitudes of the values improves the numerical stability of many training algorithms. Finally, gradient descent-based algorithms show improved convergence speed with normalization [13].

Another common pre-processing step is per-feature zero-centering of the data. For example, given a dataset of grey-scale images, $\mathbf{X}$, with j indexing pixels and i indexing samples:

$$\mathbf{X}'_{i,j} = \mathbf{X}_{i,j} - \mu_j \tag{2.5}$$

where μ_j is the dataset mean along the j_{th} pixel of all the samples in the dataset. Another commonly performed operation is dataset scaling in order to have unit variance along each feature. Maintaining the previous example of an image dataset, the normalization is obtained by performing the following transformation:

$$\mathbf{X}'_{i,j} = \mathbf{X}_{i,j} - \sigma_j \tag{2.6}$$

where σ_j is the variance calculated along the j_{th} pixel of all the samples in the dataset.

Another transformation that simplifies the learning is the decorrelation of the input features. The most common method is Principal Component Analysis (PCA), a technique to remove linear correlations from a set of input data. Basically, PCA linearly un-correlates the data by calculating the covariance matrix and calculating its eigenvalue decomposition. The dataset is then multiplied by the calculated eigenvectors to re-orient the samples in the direction of the principal components. A secondary benefit of performing PCA is that it

can be used to perform dimensionality reduction as well. Extracting the eigen-representation of the covariance matrix, the eigenvectors whose corresponding eigenvalues are small can be discarded in order to simplify the problem.

2.3 Selecting the model

Model selection is a step that is often performed based on heuristics and available literature results. However, machine learning algorithms can be characterized based on different attributes useful to, at least, exclude some choices. Representation power (i.e. the maximum complexity of results that can be achieved) is clearly the first attribute, as it determines if a given problem can be solved by a model class. In this sense, the most important attribute is the ability to perform a non-linear classification (i.e. to learn a non-linear function from the data $\mathbf{x}$ to target y).

In general, a variety of models are able to solve a problem, with varying levels of effectiveness. However, better performances are generally achieved by more complex models. Thus, a trade-off between performances and computational complexity exists.

Some models have additional constraints that limit their applicability. DNNs are a prime example of that, since they require considerable amount of data to yield state-of-the-art results (although this problem can be bypassed under certain circumstances, as will be discussed in Section 3.3).

2.3.1 Models

In this section, the most important models are presented, focusing on the pros and cons of each. While many of these models are native of a particular domain of machine learning (e.g. classification, regression), they can generally be easily re-purposed for other applications.

Linear models

Linear models represent the simplest alternative in the field of machine learning. A linear dependence is assumed between the features $\mathbf{x}$ and the target y is assumed:

$$y(\mathbf{x}) = \sum_{i=1}^{N} w_i x_i + b \tag{2.7}$$

where N is the dimensionality of $\mathbf{x}$ while the w_i and b are the model parameters. Linear models can be used in regression problems or extended to classification by using a function that maps the output to integers, such as the sigmoid function discussed in Section 3.2.1. The most common choice for the error function is least square.

While the linear separability assumption is rarely valid in complex problem, the simplicity of the model makes it useful in many practical applications.

Trees and forests

Decision trees, first proposed in [14], are algorithms that handle data through a graph-like structure. In a decision tree, the leaf node represents the target variable, such as the different classes in a classification model. Inference is performed by traversing the graph, deciding the direction at each node depending on the value of one of the features. The feature x_f with the most explanatory power is selected greedily at each step by maximizing a criterion such as the information gain I_g:

$$I_g(P(y), x_f = f) = H(P(y)) - H(P(y/x_f = f)) \tag{2.8}$$

Decision trees are often used as they provide a model that is simple to understand. However, a single tree does not perform very well in advanced tasks. Random forests [15] are an advanced version of decision trees. Basically, a random forest is built from an ensemble of decision trees trained on random overlapping subsets of the dataset (random sampling with replacement or boosting) and produces an estimation based on the average of the single trees.
The resulting *forest* can then be *pruned* in order to keep only the most valuable trees and leaves nodes. This procedure is generally performed to simplify the model and reduce the computational burden, especially in tasks with constrained resources [16]. Nevertheless, more elaborated pruning procedures can also yield better performances, when the pre-trained model contains redundant leaf nodes and trees suffering from over-fitting [17].

In past years, random forests have been applied widely, especially in computer vision, including image classification [18], object detection [19], segmentation [20] and hand tracking [21].

Recently, given the success of DNNs in computer vision, extension of random forests that leverage principles of deep learning have been developed. Examples of this paradigm are deep neural decision forests, presented in [22], using a random forest to replace the final layers of a DNN and [23], where the authors use Dropout, a technique for regularizing DNNs, discussed in 3, to improve the generalization capabilities of a random forest.

Support Vector Machines

Support Vector Machines (SVM), first proposed in [24] rely on the use of a linear classifier coupled with a non-linear transformation. The basic idea is to transform data to a space with higher dimensions by applying a kernel function. The hypothesis is that in the new space the data will be linearly separable. Thus, the hyper-plane that best separates the data is learned by maximizing the margin (i.e. the distance from the data samples that are closer to the hyper-plane). Some of the most used kernel functions are the polynomial kernel and the Gaussian Radial Basis Function kernel.

One of the main advantages of SVM classifiers is that the error function is convex. Therefore, optimization is greatly simplified by the absence of local minima.

SVMs represent some of the most widely used classifiers in practice since they

offer good performances without requiring complex optimization procedures. For this reason, while DNNs usually show better performances, SVMs are still widely used. Recently, SVMs are also being used as a replacement for the last layer of DNNs [25] [26].

Neural networks

Neural networks are machine learning models based on hierarchical concatenation of linear classifiers with non-linear functions. In the following section, the two fundamental neural network architectures are introduced

Feed-forward neural networks Feed-forward neural networks are organized in layers of *neurons*, that is units receiving inputs from the previous layer and whose output is used as input in the next layer. The connections in feed-forward neural networks do not form cycles.
Each unit calculates a weighted sum of its inputs with weights $\mathbf{w}$, applies a bias term b and finally applies an activation function to the result. Thus, each unit can be seen as a function f defined as follows:

$$f(\mathbf{x}) = h(\sum_{i=1}^{N} w_i x_i + b) \tag{2.9}$$

Where $\mathbf{x}$ represents the vector of the N inputs, and $h()$ is the activation, a non-linear function. An example of activation function is the sigmoid function:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{2.10}$$

The non-linear function enables the stacking of layers of neurons to create more complex representations, as linear operators are closed under composition. As a matter of fact, using linear activation, a neural network would not gain representation power from additional layers. Neural networks are thus built stacking an input layer, an output layer, and a number of hidden layers. The parameters learned from the data are the weights $\mathbf{w}$ and the bias term b. Neural networks are trained by error back-propagation. Back-propagation is a gradient descent based algorithm. A series of training iterations are performed. At each iteration, the output of the model, $\mathbf{y}_c$ is calculated. This is referred to as the forward pass.

Then the loss $\mathscr{L}$ between the output and the target ground truth, $\mathbf{y}_c$ is calculated, for example by means of equation 2.2.

The derivative of $\mathscr{L}$ with respect to the model parameters θ (i.e. weights and biases) is calculated by mean of the chain rule of calculus. This is referred to as the backward pass

Finally, the parameters are updated according to the loss:

$$\Delta\Theta_t = \Theta_{t-1} - \eta\nabla_\Theta\mathscr{L}(\mathbf{x}, y) \tag{2.11}$$

where η is the learning rate, a scalar in the [0 1] interval while $\mathscr{L}(\mathbf{x}, y)$ is the loss associated with the model output for $\mathbf{x}$ and y. This step is performed over the entire dataset multiple times.

Neural networks have the property of being universal function approximators. In more details, a neural network with at least one hidden layer can approximate any function, given enough neurons in the hidden layer. Thus, from a theoretical point of view, any machine learning problem can be solved effectively with a neural network.

However, while the existence of the solution is guaranteed, there is no information regarding the optimization procedure or the data that are necessary to reach it. Furthermore, the loss function for optimizing a neural network is non-linear. Thus, local minima can hamper the optimization procedure.

Practical results have shown that stacking layers yields better performances with respect to *widening* the network (i.e. adding more neurons per-layer). Furthermore, in the past, neural networks with more than two hidden layers have gotten worse results than architectures with fewer layers. This problem has been attributed to the joint effect of the non-linear loss function and the curse of dimensionality, since adding layers also adds poor local minimums to the optimization problem.

Recurrent neural networks While feed-forward neural networks can be used to process any kind of data, more specialized models can be used when

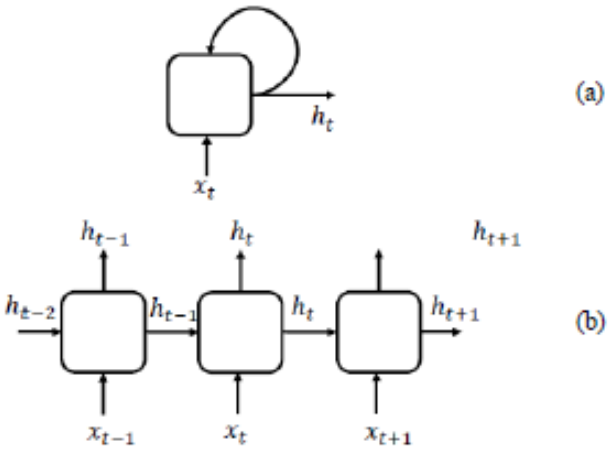


Figure 2.2: Alternative representations of RNN

the data possess certain properties.

Sequential data are quantities composed of multiple, subsequent observations of a certain quantity. Example of these types of data are sentences (i.e. sequences of words) or non-stationary audio data (sequences of sums of frequency components of varying intensities).

Since every element of the sequence has the same properties, it is desirable to re-use the learned weights.

Recurrent neural networks (RNNs) are models in which the connections between neurons can form cycles. In more details, given a recurrent neural network, the t_{th} unit outputs a value $\mathbf{h}_t$ at each time-step t (i.e. element of the sequence). $\mathbf{h}_t$ depends on the current input, $\mathbf{x}_t$ and the output of the previous unit, which, recursively, depends on the previous outputs. A recurrent network containing N units can be *unrolled* into a corresponding feed-forward

architecture, as shown in Figure 2.2.

Since RNNs do not contain non-differentiable functions, they are still trainable by error back-propagation. However, in this case, the derivatives for the recursive connections are calculated backward to the first unit. For this reason, back-propagation in RNNs is called back-propagation through time.

Non-causal extensions to RNNs have been proposed as well. Bi-directional RNNs, proposed in [27], calculate the current state based on future information as well (i.e. output from successive units). This is accomplished by training simultaneously two RNNs: one cell processes the sequence in the natural order, while the second cell is trained with the reversed sequence. The output is computed by considering the output of both cells. Bi-directional RNNs have shown better performances than simple RNNs, but they are more computationally expensive and cannot be used where causality is needed.

While a RNN can be used recursively on sequences containing any number of time-steps, the number of units, and thus the longer temporal dependency the network can learn, is limited by numerical problems. An explanation for this phenomenon has been given in [28]. In detail, calculating derivatives over RNNs with a large number of units yields a progressively larger gradient (exploding gradient), if the recurrent weights are greater than 1 or a progressively smaller gradient (vanishing gradient) if the recurrent weights are lesser than 1. Thus, RNNs could not be used over long input sequences in the past.

2.4 Conclusions

In this section, fundamental concepts of machine learning have been introduced, as well as an overview of the most important algorithms that are still relevant.

In the following Section, DNNs will be introduced in detail, giving an overview of the most relevant architectures as well as discussing the methodologies for training DNNs.

Chapter 3

Deep learning

In the past, neural networks models were limited to a shallow number of hidden layers (i.e. 1-2).

Recently, technical advancements, as well as the availability of better hardware and training data, enabled deeper neural network models less limited in the number of layers. The term deep learning has been used to describe the set of all those techniques, as opposed to shallower neural networks. DNNs represent the state-of-the-art in many machine learning tasks. More specifically, while previous machine learning systems rely on the use of a particular set of features, DNNs generally use coarser representation of the input data.

A DNN can be considered composed of two systems: the upper layers, learning a suitable representation of data for the problem at hand, and the bottom layer, a linear classifier using the learned data representation to predict on input. In

the following, the most important deep architecture are described, as well as the most important techniques used to train DNN.

3.1 Deep architectures

3.1.1 Convolutional networks

Convolutional neural networks (CNNs) are among the most important architectures in deep learning. A CNN is basically a DNN that includes one or more convolutional layers.
The convolutional layer generally contains three building blocks: the convolution, the non-linear activation and the pooling operation. The convolution and the pooling operation are described in the following, while the more general topic of the activation functions is discussed in Section 3.2.1.

Convolution operation

The most crucial element in the convolutional layer is the convolution operation. Basically, the name convolution refers to the arrangement of the connections to the next layer: in a fully-connected layer every neuron in layer l_i is connected to every neuron in layer l_{i+1} whereas, in a convolutional layer, patches of nearby neurons are connected to a single output neuron. The parameters learned are shared across all the patches. The resulting operation is

analogous to the convolution between the input to the layer x_i and the filter defined by the parameters of the connections θ_i, called the convolution kernel. For each convolutional layer, c kernels are learned, generating an output feature map with c channels. The convolution operation is shown in Figure 3.1.

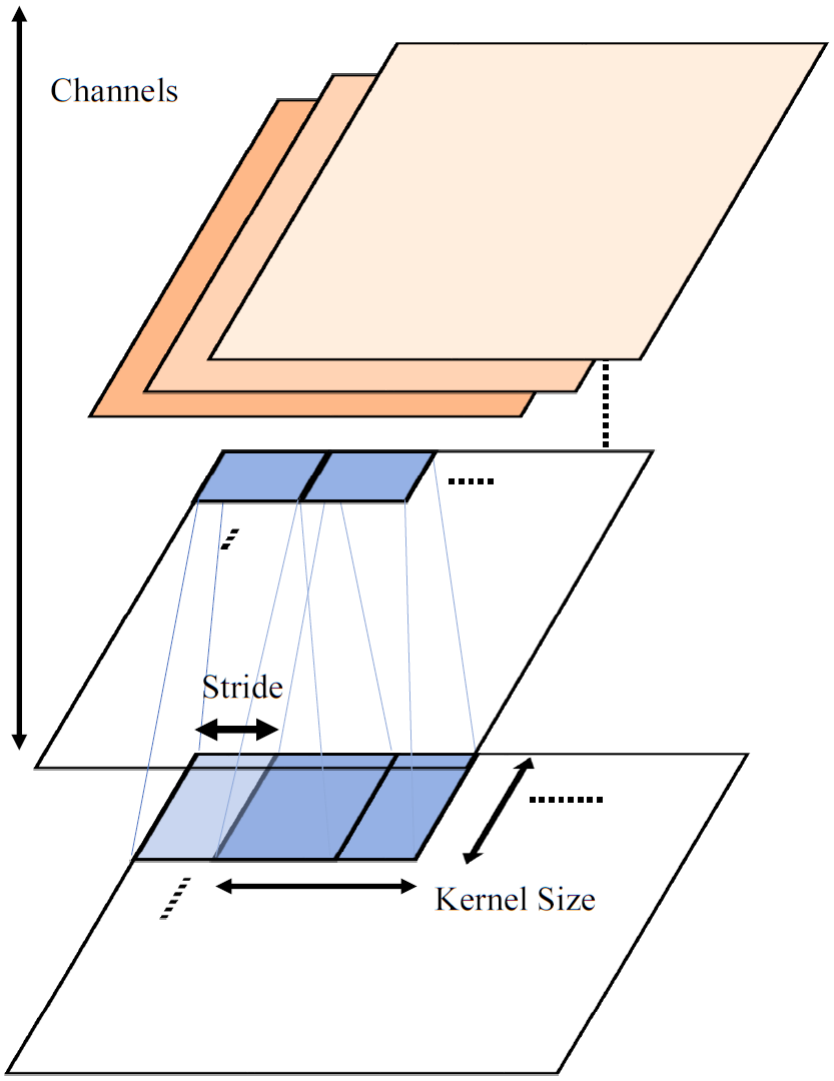


Figure 3.1: The convolution operation

This connection topology offers multiple advantages. First of all, the number of inter-layer connections is greatly reduced, simplifying the optimization.
The use of shared parameters in based on the idea that filters should act as feature detectors. The ability to detect a feature independently from its position in the input, known as invariance to translation, is necessary in many fields (e.g. image classification). The shared parameters achieve this property without having to learn the feature detector in each portion of the input, differently from fully-connected layers, simplifying the learning.
Another key advantage of the convolution operation is the local influence of the input. While it is important to detect a feature independently from where it appears, the information about the position is essential to build a meaningful, hierarchical representation of the input. As an example, an image classification problem can be considered: if a filter is trained to detect edges with a certain orientation, the pattern must be detected anywhere in the input data. However, its position with respect to the other features gives meaningful information. Figure 3.2 shows the influence field of a region of the input across multiple convolutional layers. As can be seen, in the upper layer only a limited portion of the output is influenced. In the deeper layers, a progressively larger part of the input is known to each neuron. This property is also known as sparse connectivity [29].

Besides the connection parameters learned during training, the convolution is characterized by the following hyper-parameters: the number of learned filters, the filters' dimensions and the convolution stride. The stride refers to

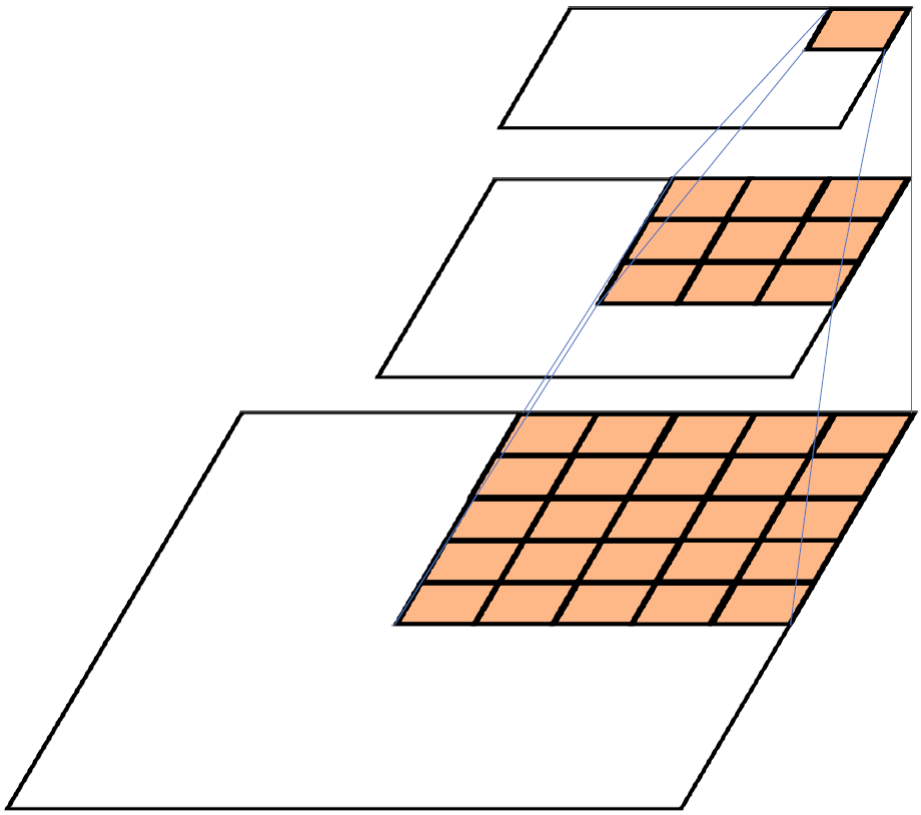


Figure 3.2: Influence of input region on single neuron activation

the density of the filters in the layer. The name comes from the analogy with the convolution operation, since the filter density can be seen as the number of discrete *steps* (i.e. input elements such as pixels) that are jumped when the filter shifts.

Finally, an important variation of the convolution operation is the dilation. Dilation refers to the filtering of the inputs considering non-adjacent elements. Figure 3.3 shows how a dilated convolution is performed. The default convolution is considered to have a dilation rate of 1. However, a dilation factor

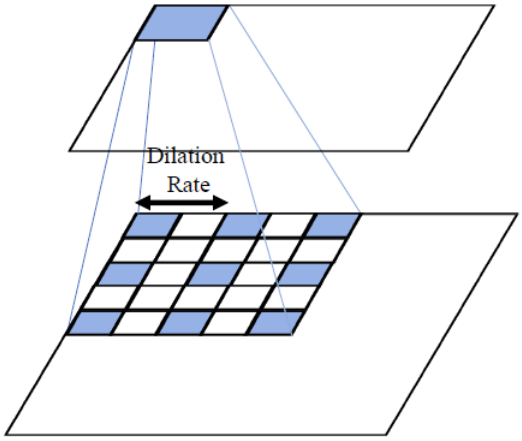


Figure 3.3: Dilated convolution operation

greater than 1 basically inserts *holes* in the filter, skipping input elements while widening the receptive field. Dilated convolutions have been proposed in [30] and have shown good results in practice, especially in object segmentation tasks [31].

Pooling

The pooling operation replaces values in the input with a statistic calculated on them. Common examples are the average pooling (i.e. replace a patch of values with their average) and the max pooling (i.e. replace a patch of values with the maximum), albeit many forms of pooling have been proposed [29]. While pooling can be lossless, it is usually used as a form of downsampling. Pooling is performed to reduce the input of the next layer and to introduce invariance to small translation in the input. While in principle it seems redun-

dant to the invariance introduced by the convolution operation, in this case the aim is to impose the prior that the position at which a particular occurs is invariant to small translation. Thus, while convolution introduces invariance to translation at a higher representation level, keeping the position information at feature level, the pooling operation loses the position information to introduce lower-level invariance. Theoretical attempts at analyzing the effect of max-pooling have shown that the dimension of the pooling window has important influences on the features that are propagated to the next step [32]. In practical applications, max pooling is the most used variation, since it generally shows enhanced effectiveness with respect to other operations [33]. More elaborated pooling operations have been proposed as well in literature. In [34], the authors propose max-pooling dropout a pooling scheme that samples the activations based on a multinomial distribution during training while leveraging a weighted sampling at test time. In [35], different pooling schemes are proposed, based on learned combination of max and average pooling.

Evolution of CNNs

Among deep learning architectures, CNN is the most studied one and was first proposed in [36]. However, it was popularized in [1], where the proposed model, named Alexnet, surpassed the competitors by more than 10% in accuracy [10]. The Alexnet model leveraged several results in neural network training, such as dropout and rectified linear units (described in the following). These tech-

niques have since become fundamental building blocks of deep neural networks and are described in the following sections.

Since the introduction of Alexnet, advancements have been summarized inside named architectures that represent milestones in CNN research. In the following, the most important architectures are presented, focusing on the structural improvement to the basic CNN architecture.

An architecture worth mentioning, despite it not introducing modifications to the convolutional layers' structure, is the Vggnet, proposed in [37]. The main architectural difference with respect to previous architectures is the number of layers: while Alexnet has only 5 convolutional layers, VGGnet has 19, demonstrating that depth of the model is directly related to performances.

The next breakthrough in convolutional architectures was achieved by Googlenet [38]. Beside increasing the number of convolutional layer to 21, the authors introduced the inception module. The idea at the basis of the inception module is to use a convolutional layer that leverages several filter dimensions per-layer. However, since this would introduce too many parameters in the network, each module first performs a 1x1 convolution on its input. The 1x1 convolution acts as a pooling on single features across the channels. The number of channels of the 1x1 convolution is a hyper-parameter that controls the intensity of the channel pooling.

In 2015, a further record was achieved by the Resnet architecture [39]. The two main features of ResNet are the level of depth, that reached 150 layers, as well as the introduction of the residual block. As a matter of fact, the authors

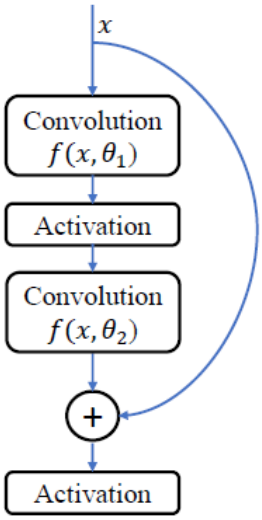


Figure 3.4: Architecture of the residual block

claim that the increase in depth without the residual block would yield worse performances in both training and testing. The residual block structure is shown in Figure 3.4. Basically, a residual block works like a normal convolutional layer, except the original input gets added after two layers of weighted summation:

$$y_i = \sigma(\sum_{i=1}^{N} w_{j+1,i}\sigma(\sum_{i=1}^{N} w_{j,i}x_i + b_j) + b_j + 1 + x_i) \tag{3.1}$$

Where σ is the activation function, while w_j, i and b_j represents the parameters of the j_{th} layer.

3.1.2 Long short-term memory and gated architectures

Long short-term memory (LSTM) architectures, first proposed in [40], are variants of the previously described RNNs, developed with the purpose of addressing the vanishing/exploding gradient problem. An LSTM is a recurrent neural network with additional parameters to control the flow of information (and thus the gradient) through the recurrent connections. In more details, an LSTM cell is composed of recurrently connected units. Each unit has three inputs: the cell state from the previous unit, $\mathbf{C}_{t-1}$, the previous output, $\mathbf{h}_{t-1}$, and the current input, $\mathbf{x}_t$. The inputs are used to update the cell state, yielding $\mathbf{C}_t$, and to calculate the current output $\mathbf{h}_t$.

The update of the cell state is performed in two steps: first, the input from the previous cell state $\mathbf{C}_{t-1}$ is modulated by the Forget Gate as detailed in Equation 3.2; then, the update of the cell state due to the current input is computed by the combination of the content of the Contribution Selection block and the Input Gate.

Equation 3.2 is used for updating the cell state:

$$\mathbf{C}_t = f_t \mathbf{C}_{t-1} + i_t \tanh(\mathbf{W}_c \mathbf{x}_t + \mathbf{U}_c \mathbf{h}_{t-1} + \mathbf{b}_c) \tag{3.2}$$

where f_t and i_t are scalars in the range [0 1], and represent the outputs of the Forget and Input gate respectively, $\mathbf{W}_c, \mathbf{U}_c, \mathbf{b}_c$ are learned parameters and $\mathbf{x}_t$ is the input. f_t and i_t are calculated in the Forget and Input gates by means

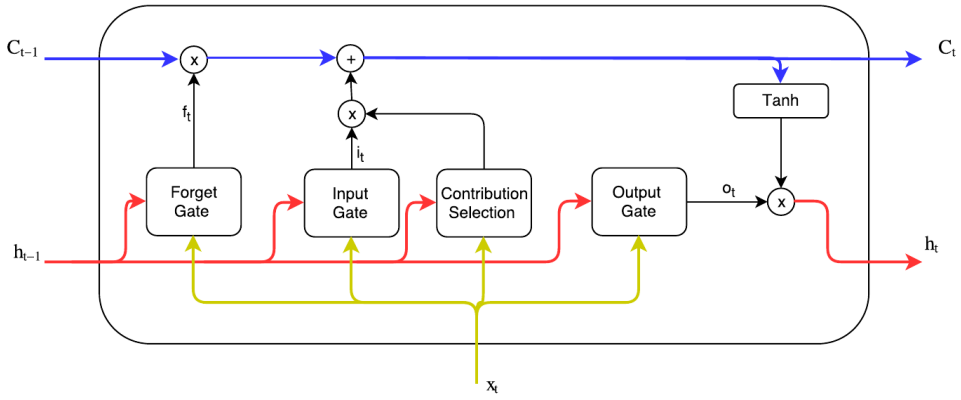


Figure 3.5: Architecture of an LSTM unit

of Equations 3.3 and 3.4 respectively:

$$f_t = \sigma(\mathbf{W}_f \mathbf{x}_t + \mathbf{U}_f \mathbf{h}_{t-1} + \mathbf{b}_f) \tag{3.3}$$

$$i_t = \sigma(\mathbf{W}_i \mathbf{x}_t + \mathbf{U}_i \mathbf{h}_{t-1} + \mathbf{b}_i) \tag{3.4}$$

where σ is the sigmoid function.

Finally, the output of the unit, $\mathbf{h}_t$, is calculated with the following equation:

$$\mathbf{h}_t = o_t \circ \tanh(\mathbf{C}_t) \tag{3.5}$$

where o_t is a scalar in the range [0 1] given by the Output Gate:

$$o_t = \sigma(\mathbf{W}_o \mathbf{x}_t + \mathbf{U}_o \mathbf{h}_{t-1} + \mathbf{b}_o). \tag{3.6}$$

Variations of the basic LSTM architectures have also been proposed. Basi-

cally, the units in the cell can be modified by either adding connections, such as in [41], or by simplifying the unit architecture, such as in [42]. Nevertheless, a comparison among many unit structural variations of LSTM cells, performed in [43], demonstrates that none of the variations offer a significant advantage over the basic one.

Gated architectures have been successfully applied to multiple tasks, including language modeling [44], machine translation [45] and natural language processing [46]. However, good performances have been demonstrated also in image processing tasks thanks to their memory. Examples of this are [47] and [48], where recurrent architectures are used for image generation and inpainting, respectively. While LSTMs (and RNNs in general) are limited to bi-dimensional data, extensions have been proposed to handle multi-dimentional sequences, such as videos [49], where the authors leverage multiple hidden connections to handle the data dimensionality. In [50], hierarchical RNNs are proposed. The model uses hierarchical connections between hidden units to learn multi-scale information from sequences.

3.1.3 Autoencoders

An autoencoder is a deep neural network for unsupervised learning, often with the purpose of training a generative model (i.e. a model that can generate new instances of a particular type of data). The basic structure of an autoencoder

can be divided into two parts, the encoder and the decoder. More specifically, the upper part of the network (i.e. encoder) learns a lossy representation for the input data. The lower part (i.e. decoder) learns to reconstruct the input data from a given coded representation (i.e. the code generated by the decoder). In this case, the loss function $\mathscr{L}$ is generated by a distance function between the original input $\mathbf{x}$ and the reconstructed output $\hat{x}$, such as mean squared error:

$$\mathscr{L}_{MSE} = \sum_{i=1}^{N} (x_i - \hat{x_i})^2 \tag{3.7}$$

While the most general autoencoder is built from fully-connected layers, the encoder and the decoder can be composed of arbitrary layers. As an example, the popularity of CNNs in classification tasks has inspired convolutional autoencoders [51]. In general, variations of deep architectures are first proposed in the classification domain and then re-used in other fields.

Variations of the autoencoder architecture have also been proposed to deal with particular properties of the data.

Denoising autoencoders [52] learn a representation from data corrupted by stochastic noise. A common form of noise used is the elimination of random features in the input, that is stochastically setting some of the $x_i = 0$. The network is then trained by calculating the loss between the reconstructed output (obtained by the noisy version of $\mathbf{x}_n$) and the uncorrupted input $\mathbf{x}$, so that robustness to noise is forced into the learned representation.

Finally, a notable evolution of the basic autoencoder are Variational Autoencoders (VAEs) [53]. The idea behind VAEs is to model the observations as

produced by a set of latent variables, which distribution $P(z)$ must be learned. An encoding function P_e is learned, maximizing $P(z/\mathbf{x})$ The generation of new data is achieved by sampling $P(z)$. The sampled values are fed to a function $P_d$that produces a sample $\hat{\mathbf{x}}$ that maximize $P(\hat{\mathbf{x}}/z)$. A gaussian prior is imposed over $P(z)$ to make the problem tractable.

3.1.4 Generative adversarial networks

Generative Adversarial Networks (GANs) have been proposed in [54] and have rapidly gained popularity in the field of DNN-based generative modeling. The basic idea behind GANs is to train a generative model by making it play an adversarial game (in the game theoretic sense) against a discriminative model (i.e. a classifier). More specifically, the generative model produces samples for the target distribution. The discriminator is fed with data from both the dataset and the generator and has to yield a label for the current sample: genuine (i.e. from the dataset) or fake (i.e. from the generator). It is possible to prove that a GAN is able to recover the true probability distribution generating the data, given infinite capacity for the models [54]. Nevertheless, the training process is not stable. A balance between the training iteration of the discriminator and the generator must be maintained, since otherwise the networks are unable to train. As an example, if the discriminator is too efficient, the generator will not be able to get meaningful gradient and train. GANs are currently very popular in generative modeling and have been ap-

plied to a variety of fields, from traditional tasks such as image in-painting and denoising to the generation of images starting from text description [55].
Furthermore, many variations of the traditional architecture have been proposed incorporating elements from other deep architectures, such as Deep Convolutional GANs [56], which employ de-convolutional layers [57], a variant of the convolutional layer.

3.2 Deep networks components and techniques

In this section, the components of DNNs are presented, discussing their importance and performances. The most relevant techniques to effectively train a DNN are also presented in this section.

3.2.1 Activation functions

Activation functions have a critical role in determining the success in training a DNN. As already stated in Chapter 2, non-linear activation functions are needed since linear operators are closed under composition.
Generally, the main concern with non-linearities is the saturation region, that is the intervals of the input domain for which the non-linear function h does not have a response proportional to the input. If the input falls in these regions during training, the corresponding gradient will be zero, thus preventing any learning. As a matter of fact, the use of better non-linear functions is

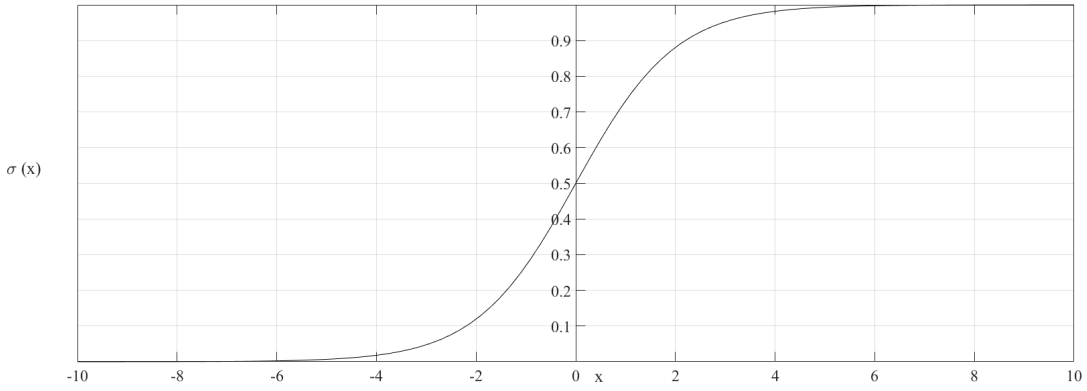


Figure 3.6: The sigmoid function

considered one of the key factors that contributed to the success of DNNs. In the following sections, the most important activation functions are discussed.

Sigmoid

The sigmoid function was defined in Equation 2.10 and it is here repeated for the sake of clarity:

$$\sigma(x) = \frac{1}{1 + e^{-x}} \tag{3.8}$$

The sigmoid function was mostly used in older neural networks. As can be seen in Figure 3.6, the sigmoid function presents two drawbacks: the saturation regions and the non-zero-centered outputs.

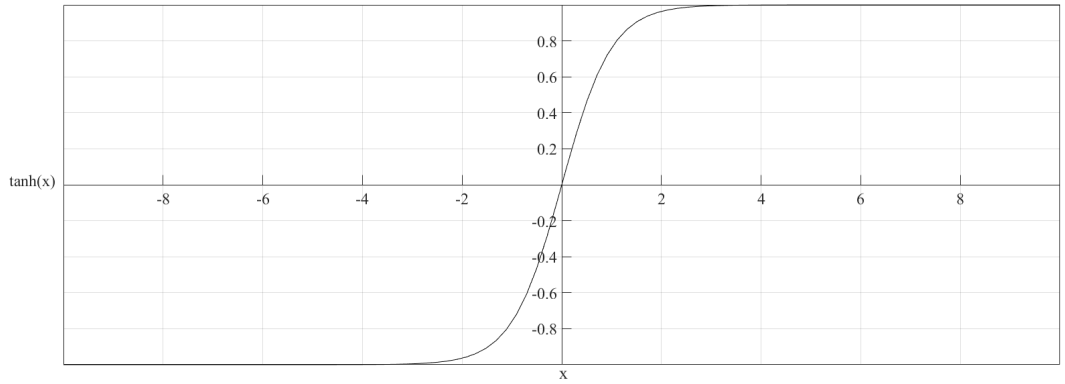


Figure 3.7: The hyperbolic tangent function

Hyperbolic tangent

The hyperbolic tangent is defined by:

$$tanh(x) = \frac{1 - e^{-2x}}{1 + e^{-2x}} \tag{3.9}$$

The *tanh* function was first recommended in [13] as a strictly better alternative to the sigmoid function. As can be seen in Figure 3.7, while the saturation problem is still present, the output of the *tanh* is zero-centered, avoiding potential oscillation in the parameters updates.

Rectified linear units

The Rectified Linear Unit (ReLU) was first proposed in [58] as a mean to improve the performances of Restricted Boltzmann Machines. The ReLU is

defined as:

$$ReLU(x) = \begin{cases} x & x \geq 0 \\ 0 & x \leq 0 \end{cases} \tag{3.10}$$

The ReLU output is not zero-centered. However, a ReLU does not saturate for large values of the activation. In [1], it has been shown that the training process is much more efficient when ReLUs are used in place of $tanh$.

Nevertheless, the ReLU still saturates for low (i.e. < 0) activation values. This property can still hinder training, especially if the random parameters' initialization is sampled so that a part of the network starts in a saturated state. For this reason, variation of the ReLU have been proposed.

Leaky variants of the ReLU (lReLU) with a small proportionality relation between the input and the output when the activation is less than 0 have also been proposed:

$$lReLU(x) = \begin{cases} x & x \geq 0 \\ \alpha x & x \leq 0 \end{cases} \tag{3.11}$$

where α can be a constant or a trainable parameter [59]. Other proposed improvements to the ReLU include units able to output zero-mean values, such as Exponential Linear Units (ELU) [60], defined as:

$$ELU(x) = \begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x \leq 0 \end{cases} \tag{3.12}$$

where α is a constant hyper-parameter. However, since no activation function has shown strictly better performances, ReLUs are still considered the safe default in terms of activation functions.

Maxout activation

The Maxout activation was proposed in [61]. Basically, the activation outputs the maximum value between k inputs:

$$\sigma_{max} = \underset{j \in 1,2...k}{max}(\sum_{i=1}^{N} w_{j,i}x_i + b_j) \tag{3.13}$$

Maxout can be considered as an extension of dropout, described in Section 3.2.3, that is a combination of different sub-models, or as a generalization of a ReLU, that is effectively a particular case of the Maxout activation ($k = 2$, with one of the inputs being 0). However, a maxout activation has a considerable computational cost, since each of the k inputs has its own set of parameters.

Activation of the output layer

The final layer of a neural network is a linear classifier that does not need a non-linear activation function to be effective. As a matter of fact, in regression problems, the output of the neuron is taken as it is.
However, in classification problems, it is useful to get outputs that are more comparable to one-hot encoded labels or, better yet, values that can be inter-

preted as probabilities. In binary classification problems, the sigmoid function is used, as it can be seen as the posterior probability of the classes [62].
For multi-class classification, the softmax function is used, defined as:

$$\sigma_m = \frac{e^{x_i}}{\sum_{i=1}^{N} e^{x_i}} \tag{3.14}$$

The softmax can be seen as an extension of the sigmoid. Being a normalized exponential function, the probabilistic interpretation is also straightforward.

3.2.2 Optimization for deep neural networks

Optimization algorithms are selected based on their convergence properties, such as, theorems on converge speed and optimality of the solution found. Furthermore, secondary benefits, such as low sensitivity to the algorithm's hyper-parameters can also be considered.
As previously stated, the training of a neural network is a non-linear optimization problem. Given the number of parameters involved, it is usually feasible to only compute the first order derivatives of the loss function with respect to the parameters. Besides, given the nature of the problem, the loss function minima found by the gradient descent are not guaranteed to be global.
Extensions of the previously described gradient descent algorithm have been developed to improve its convergence, as well as to decrease its sensitivity to hyper-parameters. The most well-known extension is the Stochastic Gradient Descent (SGD). The SGD represents an extension of gradient descent suited

for very large datasets. In these cases, computation of the loss gradient over the whole dataset can be impossible because of hardware constraints (e.g. the dataset does not fit into the working memory) or very slow. In the SGD the gradient of the loss function is estimated from a limited number of samples at a time. Thus, the parameters' update equation is given by:

$$\Delta\Theta_t = \Theta_{t-1} - \eta \sum_{i=1}^{n} \nabla_\Theta \mathscr{L}(\mathbf{x}_i, y_i) \tag{3.15}$$

where η is the learning rate, n, the batch size, represents the number of samples used in a single parameters update and $\mathscr{L}(\mathbf{x}, y)$ is the loss calculated on the n-element batch. The reduced number of samples hinder the variance of the gradient estimation, thus smaller η are used.

The SGD represents the de facto standard to train a DNN given the size of the datasets employed. However, extensions to the basic SGD algorithms have also been proposed. Many algorithms have been proposed based on combinations and variations of two fundamental concepts: Momentum and variable learning rate.

Momentum

Momentum was first proposed in [63]. The idea is to use the gradients from the previous iterations linearly combined with the gradient of the current time-step. Depending on the curvature of the loss function, in [64], it was demonstrated that Momentum yields up to a quadratic reduction in iterations needed

to converge to a local minimum. Thus, the model parameters are updated according to the following equations:

$$\Delta\Theta_t = -\eta \sum_{i=1}^{n} \nabla_\Theta \mathscr{L}(\mathbf{x}_i, y_i) + \alpha \Delta\Theta_{t-1} \tag{3.16}$$

Where α is a scalar $\in [0\ 1]$

A technique closely related to Momentum that is also used to accelerate the SGD is the Nesterov Accelerated Gradient (NAG) also known as Nesterov momentum, first proposed in [65]. The NAG is similar to a standard momentum in that it uses the previous gradient to accelerate the convergence. However, in this case, the gradient for the current time-step is calculated after updating the parameters with the Momentum term. The update rule is thus:

$$\Delta\Theta_t = -\eta \sum_{i=1}^{n} \nabla_{\Theta+\alpha\Delta\Theta_{t-1}} \mathscr{L}(\mathbf{x}_i, y_i) + \alpha \Delta\Theta_{t-1} \tag{3.17}$$

In general, while momentum offers significantly improved performances in terms of convergence time, it should be noted how it also requires more computational power. As a matter of fact, during training, previous calculations of the gradient will have to be kept in the memory.

Variable learning rate

Variable learning rates refer to the use of an η that is a function of *time* (i.e. function of the number of iterations) t. In practice, it is observed that dur-

ing the earlier gradient calculations, the loss function decreases more steeply toward the minimum. In the more advanced phases, however, progressively smaller updates are needed to properly reach the minimum.

Thus, the learning rate is usually decreased as the number of gradient iteration rises. A common procedure is to decay the learning rate according to some function of the number of iterations t. Some examples are time-based decay:

$$\eta(t) = \frac{\eta_0}{1 + kt} \tag{3.18}$$

and exponential decay:

$$\eta(t) = \eta_0 \exp^{-kt} \tag{3.19}$$

where k is a scalar hyper-parameter, the decay factor.

However, adaptive learning rates have also been proposed. An adaptive scheme of per-parameter learning rates has been proposed in [66]. The proposed algorithm, Adagrad, calculates an upgrade for each parameter based on all the past updates of the parameter itself. More precisely, given a parameter θ_i, its update is calculated as follows:

$$\theta_t = \theta_{t-1} - \eta_{t,i} \nabla_{\theta_t} \mathscr{L}(\mathbf{x}_i, y_i) \tag{3.20}$$

where $\eta_{t,i}$ is given by:

$$\eta_{t,i} = \frac{\eta}{\sqrt{\sum_{\tau=1}^{t-1} (\nabla_{\theta_\tau} \mathscr{L}(\mathbf{x}_i, y_i))^2}} \tag{3.21}$$

The use of all the past gradients can generate too small updates during advanced iterations. For this reason, it is often observed in practice that Adagrad converges to slightly worse solutions than the SGD. However, Adagrad has a much weaker sensitivity to the initial learning rate η.

In order to slow the steep decay of the learning rate observed in Adagrad, the RMSProp algorithm has been proposed. The RMSProp uses a similar per-parameter decay scheme, albeit it dampens the influence of past gradients by averaging and decaying exponentially with respect to iterations. The normalized learning rate is calculated as follows:

$$\eta_{t,i} = \frac{\eta}{\sqrt{\gamma(\frac{\sum_{\tau=1}^{t-2} \nabla_{\theta_\tau} \mathscr{L}(\mathbf{x}_i, y_i)}{t}) + (1-\gamma)\nabla_{\theta_t} \mathscr{L}(\mathbf{x}_i, y_i))}} \tag{3.22}$$

where γ is the decay factor. Albeit the RMSprop algorithm is unpublished, a very similar version exists, named Adadelta [67].

Figures 3.8 shows the different performances of SGDs and its variants. As can be seen in Figure 3.8 (a), with an appropriate η, both the SGD and the momentum version converge slower than Adadelta, while Adagrad performances worsen due to the learning rate decay scheme. As the number of iterations grows, the SGD converges to the same $\mathscr{L}$ values of Adadelta. However, when η is not properly set, the SGD converges to much worse values than Adagrad and Adadelta (Figure 3.8 (b)). Finally, the Adam algorithm [68] combines the ideas of the RMSprop and the decaying momentum by applying the following

update rule:

$$\theta_t = \theta_{t-1} - \eta_{t,i}\mathscr{G}(\Theta) \tag{3.23}$$

where $\mathscr{G}$ is given by:

$$\mathscr{G} = \alpha\Delta\Theta_{t-1} - (1-\alpha)\nabla_{\theta_\tau}\mathscr{L}(\mathbf{x}_i, y_i) \tag{3.24}$$

and $\eta_{n,i}$ is given by Equation 3.22.

3.2.3 Regularization and normalization

Regularization refers to the set of techniques employed to avoid training instabilities as well as to improve the generalization capabilities of a model. In any machine learning model, imposing constraints over the parameters has a regularizing effect. In more details, the norm of the parameters is added to the function $\mathscr{L}$, optimizing over the modified loss:

$$\mathscr{L}_r = \mathscr{L} + \lambda||\Theta||_n \tag{3.25}$$

where λ is a scalar in the interval [0 1] that controls the intensity of the regularization and n is the order of the norm. Depending on the order of the norm, n, different effects are obtained.

The L_1 norm, defined as:

$$||\Theta||_1 = \sum_i |\theta_i| \tag{3.26}$$

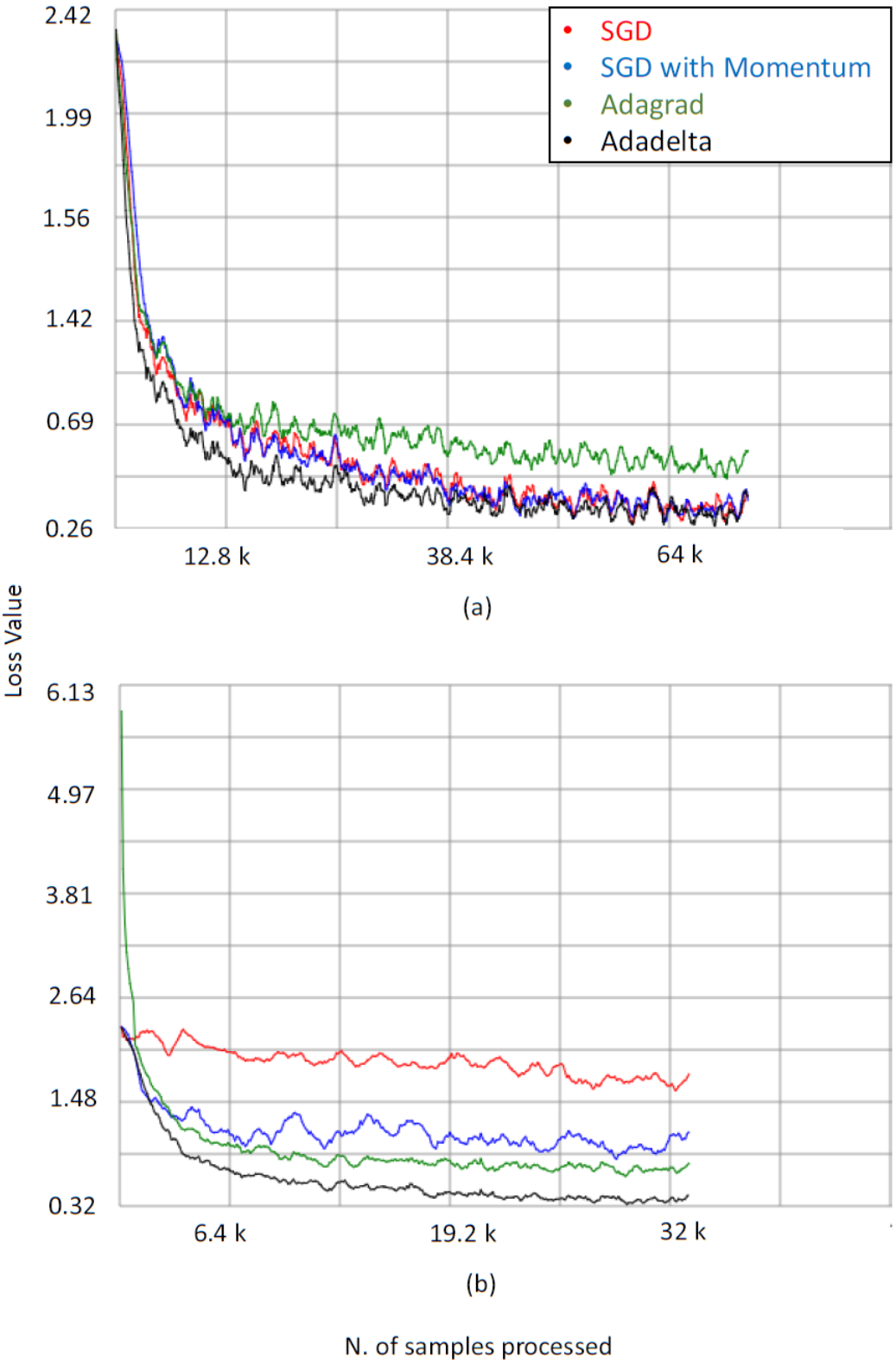


Figure 3.8: Loss for a fully connected network trained on the MNIST dataset, with $\eta = 0.01$ (a) $\eta = 0.1$ (b)

is heuristically used in order to encourage sparsity in the parameters. It has been shown analytically in linear models that the L_1 norm can converge to solutions with multiple zero-valued parameters [29].

The L_2 norm, defined as:

$$||\Theta||_2 = \sum_i \theta_i^2 \tag{3.27}$$

penalizes large weight values. Penalizing larger weights severely reduces the model over-fitting.

Among the techniques specific to DNNs, dropout [69] has been used extensively in recent years, yielding good practical results. Dropout refers to the introduction during the model's training of stochastic neuron activation. More specifically, during a forward pass, each neuron is active with static probability P_d. Dropout has been proposed initially to limit the co-adaptation of the features learned by the DNN: since neurons are not active in each training iteration, the network strives toward learning mutually independent features.

Batch normalization (BN) is a commonly used technique in modern feed-forward neural networks. In [70], the authors argue that layers in a neural network experience covariance shift [71] during training. In more details, as the network trains, the parameters are updated; deeper layers see a shift in the input data distribution due to the parameters' updates. This phenomenon can drive the non-linear functions in the network in saturation, causing the gradient to vanish. To address this concern, inter-layer whitening could be used. However, the authors point out that this operation would be inefficient and could potentially interfere with the training. BN aims at solving inter-

nal covariance shifts by introducing a mini-batch, differentiable transformation between layers. Basically, the inputs of every layer defined by:

$$\hat{x_i} = \frac{x_i - \mu(x_i)}{\sigma(x_i)} \tag{3.28}$$

where x_i represent the i_{th} feature in the input (e.g. a pixel in an image), while μ and σ are the mean and standard deviation of the i_{th} feature across the mini-batch. Since this transformation could hinder the representation capabilities of the network, a second step is introduced:

$$y_i = \gamma_i x_i + \beta_i \tag{3.29}$$

Where γ_i and β_i are learned parameters. The authors note that this scaling and shifting restores the representation power, since the whole transformation is not lossy. In practice, BN has shown impressive results in improving the iterations to convergence in training deep feed-forward networks such as CNNs, also reducing sensitivity to other hyper-parameters such as the learning rates [70].

An indirect form of regularization is done by dataset augmentation. Augmentation of the dataset refers to training the model by exposing it to modified versions of the input $\mathbf{x}$. The procedure is performed to obtain an dataset effectively bigger than the original, as well as to introduce invariance to selected input transformation in the model. Example of augmentation are cropping, rescaling and flipping in images or time stretching and pitch shifting in audio.

3.2.4 Visualization techniques

While the performance of DNNs have widely surpassed other feature-based methods, one of the most critical issues of deep learning is understanding the learned representation (i.e. explaining the performances) from both a theoretical and intuitive point of view.

One of the most used approach is to understand the learned representation through visualization techniques. An important advantage of such techniques is that they give insight about the success of the learning, as well as about the tuning of the hyper-parameters, as will be explained shortly.

In feed-forward networks, one approach is to plot the activation of filters with respect to an instance of a particular class. While this technique gives parameters plots that are harder to interpret, it is useful to assess the validity of the hyper-parameters by checking for filters that never activate [72].

Visualization of the learned weights (e.g. the filters in a CNN) is sometimes used as a debugging technique. Noisy weights, as opposed to smooth ones, generally indicate improper training parameters.

More complex approaches have also been proposed. Activation maximization [73] consists in optimizing an image so that it maximally activates a certain portion of the network. Basically, a random image is optimized so that one of the neurons in the network, generally taken from the deeper layers, yields the maximum activation possible, by optimizing the gradient of the activation

with respect to the input pixels:

$$x_i = \arg\max \frac{\delta F_i(\mathbf{x})}{x_i} \tag{3.30}$$

where $F_i(\mathbf{x})$ is the activation of the neuron corresponding to the i_{th} class. The optimization is performed similarly to the learning, albeit in this case the parameters are considered constants and the input is optimized.

This optimization approach can also be used to understand which part of the input are more significant for the DNN. In [73], the calculation of a saliency map based on the derivative of F_i with respect to the input is proposed. A saliency map is an image with the same dimensionality of the input $\mathbf{x}$, which pixels are given by:

$$s_i = \arg\max \frac{\delta F_i(\mathbf{x})}{x_i} \tag{3.31}$$

assuming that the image m by n with is re-shaped as a 1-dimensional vector with $N = mxn$ elements.

As shown above, many visualization techniques have been proposed for feed-forward architectures. RNNs have been studied less, partly because of their structure, which does not allow to establish a simple correspondence between the parameters and the visual features of an image. However, attempts to visualize RNNs have been done in [74] by studying the saturation of the gates in LSTMs as well as the activation of the units. While some insights about how the cell learned to process text have been derived(e.g. keeping track of line length, keeping track of comments), understanding the learned parameters

of a RNN remains an open problem.

3.3 Properties of DNN

In this section, two fundamental properties discovered in DNNs are presented and discussed. Transfer learning enables the use of DNNs in context where large datasets are not available, while adversarial examples constitute an important caveat for the application of DNNs in security contexts.

Transfer learning

Transfer learning refers to the use of parameters learned to solve a problem, such as the classification of a particular dataset (i.e. of particular classes) in order to solve a similar problem. Most often, the similarity lies in the type of data that is being processed.

While transfer learning has been practiced and studied in the past [75], the hierarchical structure of a DNN allows for simpler approaches to transfer learning. In more details, it has been observed that the first layer of deep networks learns similar features, independently from the problem or the formulation of the loss $\mathscr{L}$ [76]. However, learned patterns become progressively more specific with respect to the problem, its formulation and the dataset in the deeper layer of the network.

The most used transfer learning scheme leverages a feed-forward network that

has been trained on a very large dataset, such as Alexnet or Resnet, described in Section 3.1.1. Depending on the data available for the target task, two approaches are possible. If the target dataset contains the same kind of data, then the layers that work as feature extractors (e.g. the convolutional layers) are kept as they are (the parameters are said to be *frozen*), while the lower layers (e.g. the last layer, acting as a linear classifier) are trained with a low η (fine-tuning). This technique, also known as pre-training, is particularly important as it enables the use of DNNs even in the absence of dataset of proper size. Furthermore, it enables the use of deep models without the need of a high computational power. As a matter of fact, training the whole network has a much bigger computational cost than fine-tuning, since fewer iterations are required.

If the first task is very different from the second task, initializing the parameters with the values obtained by another network improves the training. In this case however no parameter is frozen and a much larger dataset is needed.

Adversarial examples

A peculiar phenomenon that has been observed in DNNs is the vulnerability to adversarial examples. Basically, DNNs consistently misclassify samples that have been modified with adversarial noise [77]. The adversarial noise was obtained by optimizing a noise mask with respect to the network loss function $\mathscr{L}$. The noteworthy discovery is that even for a very small perturbation (i.e. not

visible by the human eye), the samples are classified as completely unrelated objects with high confidence. Furthermore, adversarial samples generated for a particular DNN transfer to other networks trained on similar task as well.
Since the first contributions, other strategies have been developed to generate adversarial examples leveraging evolutionary algorithms [78], as well as attacks that can be successful in modifying only a very limited number of features in the input (e.g. a limited number of pixel in an image) [79]. Furthermore, it has been shown in [80] that knowledge of the network's parameters or structure is not necessary for an attacker, as a substitute model that enables the attack is obtainable with a limited number of queries to the model that is to be attacked.
While this property could be seen as a flaw of deep models, in [80] it is shown that other machine learning algorithms are vulnerable as well.
Some countermeasures have been proposed for this problem [81], but deeper investigation has shown that such countermeasures are not completely effective [82]. This phenomenon has important implications for the use of DNNs in contexts where security might be an issue, especially since it has been shown [83] that it is possible to achieve misclassification in real world scenarios by leveraging simple physical alterations such as stickers. In contexts such as self-driving cars, this represents a paramount problem.

3.4 Conclusions

In this section DNNs have been discussed. To date, DNNs have surpassed hand-crafted features in almost every traditional machine learning task. The phenomenon of transfer learning enables the use of DNNs even in contexts with a reduced amount of available data.

However, training DNNs is a complex process that largely relies on heuristics. Furthermore, phenomena such as adversarial examples demonstrate that the learned representations are not fully understood, despite the attempts made with techniques such as visualization.

Chapter 4

Applications to audio

4.1 Introduction

In this Chapter, the applications of the concepts of the estimation of semantic attributes based on the techniques discussed in Chapter 2 and 3 are considered in the context of audio. The effectiveness of the aforementioned methods is evaluated in the context of audio surveillance. This domain has different requirements with respect to other applications of machine learning in the auditory domain, such as music processing and speech recognition. Thus, it is worth evaluating the performances of advanced techniques such as DNNs in this context. Section 4.2 introduces the fundamentals of audio signal processing that are relevant to machine learning applications, while Section 4.3 discusses the most important audio features and the most recent contributions to audio

machine learning.

Finally, Section 4.4 presents the aforementioned applications of DNNs in the domain of audio surveillance, originally proposed in [85].

4.2 Digital audio fundamentals

In general, a sound can be modeled as a continuous signal in time, characterized by its amplitude $x(t)$. As it is well known, it is possible to represent $x(t)$ through the coefficients of its Fourier transform:

$$X(f) = \int_{-\infty}^{+\infty} x(t) e^{-i2\pi f t} dt \tag{4.1}$$

that is, $x(t)$ can be represented as a sum of harmonic components.

In order to obtain a digital representation of $x(t)$, the signal must be sampled and quantized. From the Shannon-Nyquist sampling theorem, a time-discrete representation of the signal can be obtained without losing information if the samples are taken with a sampling frequency f_s that is at least the double of the largest frequency component of x. However, due to several non-ideal features of analog to digital converters, the sampling in time loses some information, albeit small, and a higher frequency is used to compensate. In most audio-centered applications, the human hearing range of frequency is considered. In the past, most audio coding formats used 32kHz sampling frequencies, assuming that humans hear sounds in the [0 16] kHz range. Nowadays it is

common to use an f_s of 44 kHz for music applications; other systems exploit up to 192kHz sampling frequencies.

Successively, the signal amplitude is quantized. Quantization inevitably loses information. The maximum per-sample loss of information depends on how many bits are used to represent each sample. Older systems used 8 bits/sample (i.e. 256 possible values), with low audio fidelity. Currently, 16 bits/sample are used in most systems, while high fidelity standards impose quantization with 24 bits/sample. Digitalization through these steps is referred to as Linear Pulse Code Modulation (LPCM), whereas the use of non-uniform quantization schemes based on psychoacoustics considerations (i.e. A-law, μ-law) are simply called PCMs. After these steps, a digital version of $x(t)$, $x(n)$, where $n \in \mathbb{N}$ is obtained.

4.2.1 The frequency domain

As previously stated, it is possible to analyze an audio signal in the frequency domain. A PCM-digitalized signal can be analyzed through the Discrete Fourier Transform (DFT). However, unless the frequency content of the signal does not vary with time (i.e., unless $x(t)$ can be considered generated by a stationary stochastic process), analyzing the frequency components will not give any information about the time of occurrence of a particular harmonic. Since, in general, audio signals have frequency content varying with time, it

is important to identify with more precision when a particular harmonic has occurred.

To this aim, a windowed version of the DFT is used, that is the transform of a particular time window. The Short-Time Fourier Transform (STFT) has been developed for this purpose. The STFT of a discrete signal $x(n)$ is defined as:

$$\mathbf{X}(k, f) = \sum_{n=-\infty}^{+\infty} x(n)w(n-k)e^{-i2\pi fn} \tag{4.2}$$

where w is a window function, that is non-zero only in the interval under testing. The complete observation of the evolution of the signal frequency content with time is obtained by sliding the window in time by k samples.

The set of $\mathbf{X}(k, f)$ is represented graphically by using the spectrogram $S(\mathbf{X})$. The spectrogram is a matrix with the same dimensions of $X(k, f)$, and is obtained by taking the squared absolute value of the intensity for each frequency and time index:

$$S(\mathbf{X}) = |\mathbf{X}(k, f)|^2 \tag{4.3}$$

The calculation of the DFT in general introduces artifacts. This can be modeled as the product of a periodic summation (i.e. a periodic version of the signal generated by summing shifted copies of the signal) of a signal with a rectangular window w_{Rect}, defined as:

$$w_{Rect}(n) = \begin{cases} 1 & \frac{-\Delta N}{2} \leq t \geq \frac{+\Delta N}{2} \\ 0 & otherwise \end{cases} \tag{4.4}$$

where ΔN represents the length of the window in samples.
This non-ideality causes cross-talk between frequencies as well as artifacts. For this reason, other windowing functions have been developed. The following parameters are considered in designing a window function:

- Width of the main lobe.
- Intensity of the side lobes.
- Roll-off of the side lobes.

Examples of window functions that offer a trade-off between the aforementioned parameters are the Hamming windows, w_H:

$$w_H(n) = w_{Rect}(n)(\alpha_1 + 2\alpha_2 \cos{(\frac{2\pi}{M}n)}) \tag{4.5}$$

and the Blackman-Harris windows:

$$w_{BH}(n) = w_{Rect}(n) \sum_{l=0}^{L-1} \alpha_l \cos{(l\frac{2\pi}{M}n)} \tag{4.6}$$

Besides the type of the window function w, the other parameter is the length of the window (i.e., the number of samples). By considering w_{Rect}, it is possible to understand how using a larger time resolution (i.e. transforming a lower number of sample of $x(n)$ per frame)causes, in the frequency domain, the width of the main lobe to widen. Conversely, as the time resolution is reduced (i.e. transforming a higher number of sample of $x(n)$ per frame) the width of the

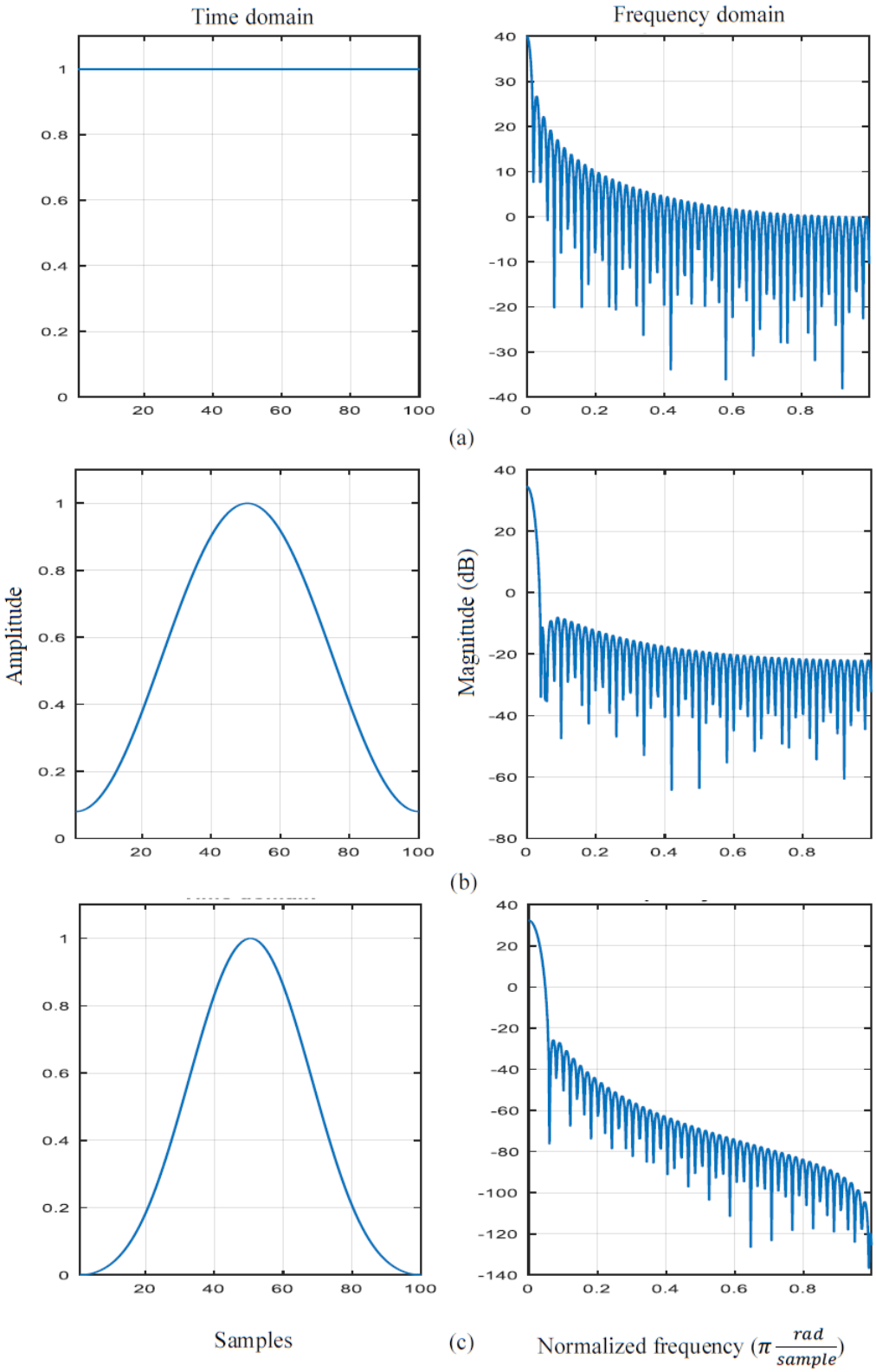


Figure 4.1: Time and frequency plots for the Rect (a), Hamming (b) and Blackman-Harris (c) windows

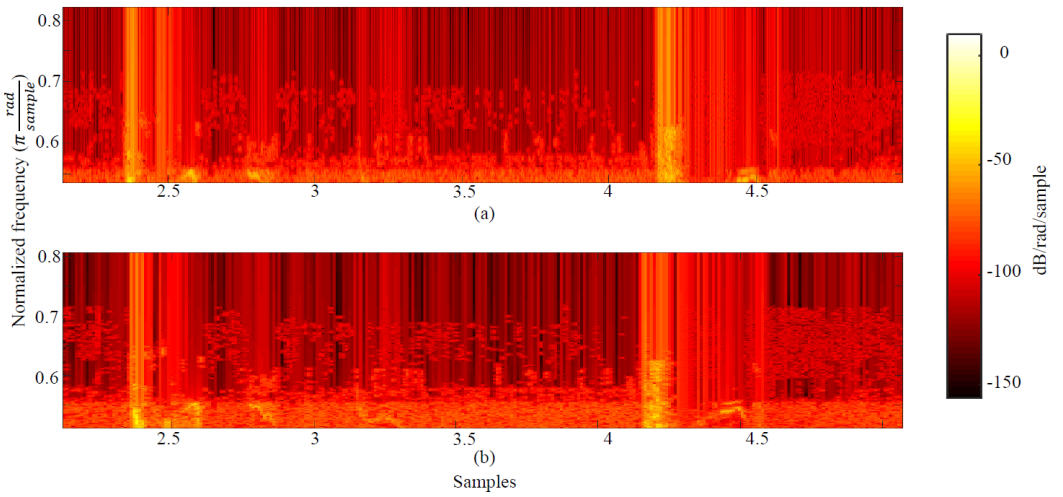


Figure 4.2: Comparison of time resolutions: spectrogram of a dog bark for 512 (a) and 1024 (b) samples windows

main lobe in frequency is reduced. Thus, another fundamental trade-off exists between time and frequency resolution, i.e. there is a fundamental limit on the resolution that can be obtained contemporary in time and frequency. It is possible to see how in Figure 4.2 (a), the spectrogram is less resolved in frequency, due to the smaller time window (i.e. 512 samples). The frequency components appear as small rectangles with the longer side parallel to the frequency axis. The opposite behaviour is shown in Figure 4.2 (b), where a longer window (1024 samples) causes the samples to be less resolved in time.

4.2.2 Perceptual considerations

Since the aim of many audio applications is to replicate the human hearing as closely as possible, auditory models have been developed with the purpose of

replicating the human frequency response. It is well known that the human ear is more sensitive to lower frequencies, and more specifically with a logarithmic perception of frequency separation (i.e. sounds are perceived equally spaced when the separation doubles).

Gammatone filters have been derived in [86] as a model of the frequency response of the cochlea (i.e. the organ that performs the frequency analysis in the human ear). Basically, a gammatone filter is obtained by modulating a sinusoidal wave (tone) with a gamma distribution [87]. This model is widely used, mainly for its simplicity.

The Constant-Q transform is an alternative construction of the STFT filterbank for music applications, based on the idea that equal separation should be granted to harmonic components in the log-frequency space [88]. The filters are spaced to obtain a constant quality factor Q, defined as:

$$Q = \frac{f_k}{\Delta f_k} \tag{4.7}$$

where f_k is the central frequency of the k_{th} bin and Δf_k is the bin width.

A representation that is commonly used in ASR and audio processing in general is the Mel-Frequency Cepstral Coefficient (MFCC). The MFCC are extracted by mapping the STFT magnitude coefficients to Mel. The mapping from Hz to Mel is performed according to perceptually derived formulas, such as:

$$m = 2595 \log_{10}(1 + \frac{f}{700}) \tag{4.8}$$

where m is the value in *mel* of the f frequency. The conversion is performed through a filterbank of M triangular filters, whose central frequencies are linearly spaced in the Mel domain (thus with logarithmic spacing in the frequency domain).

After this step, the Discrete Cosine Transform of the coefficients is taken. Finally, a subset N of the coefficients is taken (trading-off the information content and the simplicity of the representation).

In older ASR applications, MFCC greatly reduced the dimensionality of the problem, since typically only 20-30 coefficients were kept. However, by using more triangular filters and by keeping more coefficients after the DCT it is possible to control the loss of information, thus returning to the usual trade-off of feature selection.

4.3 Applications of machine learning to audio signals

Machine learning has been applied for a long time in the audio domain, primarily to process voice: this encompasses both Speech-To-Text systems, also known as Automatic Speech Recognition (ASR), and Text-To-Speech (TTS). As stated in Chapter 2, older approaches rely on the extraction of features to simplify the learning problem while keeping as much information as possible. In the following sections, the most relevant features for audio classification are

discussed. Furthermore, Hidden Markov Models (HMM)-based audio models will be presented.

4.3.1 Audio features

Designing effective features is always a challenging task. The core problem is to design descriptors that can be expressed analytically and that capture perceptual attributes of the audio signal, especially in music applications. In everyday life, general sounds (e.g. voices, music, etc.) are described according to semantic/perceptual attributes. Among the key perceptual attributes of sound, there are:

- Duration, the temporal support of the sound.
- Loudness, the perceived intensity of the sound.
- Pitch, the perceived main harmonic of the sound.
- Timbre, the set of secondary harmonics that give the *color* of the sound.
- Rhythm, the periodic pattern of organization of harmonics and in general, the temporal organization of the sound.

These types of descriptors lacks an analytical form and can exhibit a strong degree of subjectivity.
On the other hand, features with a precise analytical form are needed in order

to automatically process sound. Thus, in the designing feature, high-level, perceptual attributes of the sound must be reconstructed by mean of low and mid-level signal features. In literature, many characterizations of features rely on clustering descriptors based on the high-level attribute they attempt to estimate [89], [90].

In the following, the most important audio features will be reviewed, based on the domain upon which they are calculated (e.g. temporal, spectral).

The Zero-Crossing Rate (ZCR) is calculated by counting the number of times the signal's values change sign in the time domain. Despite being particularly sensitive to additive noise, ZCR can be used effectively in the detection of percussive sounds (i.e. non-periodic sounds) [91].

The beat histogram, proposed in [92], estimates the main periodic components inside a sound by performing a wavelet transform and extracting the envelope of the signal at different octaves, summing then and calculating the auto-correlation. The peaks of the auto-correlation function give the periodic temporal components that can be used to estimate the overall rhythm.

In general, statistics of the signal (e.g. mean, variance) extracted at various time-scales are used as temporal features.

On the other hand, spectral features consider statistical properties of the STFT short time-span between 10 and 25 ms (i.e. where the signal can be considered stationary).

The spectral centroid S_c is the mean of the frequency bins weighted with the

amplitude of the bins:

$$S_c = \frac{\sum_{n=1}^{N} f_n |X[n]|}{\sum_{n=1}^{N} |X[n]|} \tag{4.9}$$

where N is the number of bins, f_n is the frequency value associated to the n_th bin and $X[n]$ are the coefficient of the STFT for a given frame. The spectral centroid has been found correlated with the perception of the sound brightness in perceptual experiments [93].

The spectral flux S_f is calculated by evaluating the sum of squared difference between the current STFT frame and the previous one:

$$S_f = \sum_{1}^{N} (\mathbf{X}_{n,k} - \mathbf{X}_{n,k-1})^2 \tag{4.10}$$

it is used to evaluate the evolution of the spectral content.

The spectral roll-off is a percentile of the power spectrum, usually the 85_{th}, and characterizes the skew of the power spectrum distribution. Other statistical properties of the spectrum are used as features as well, such as spectrum flatness [94], envelope [95] and kurtosis [96]. Energy features can also be used: the energy of the signal is calculated over the whole vector or over sub-bands. The use of feature vectors built from the concatenation of the aforementioned features has been a very used approach in audio classification, especially for topics such as music understanding and retrieval, while for ASR, MFCC features, described in Section 4.2 have been predominantly used.

Recently, due also to the introduction of the Detection and Classification of Acoustic Scenes and Events (DCASE) challenge [97], there has been a new

wave of interest in the design of audio features. In more details, the analysis of spectrograms by means of image processing techniques has gained popularity. The first example of this approach is [98]: a pseudo-colorization technique is used to partition the spectrogram into separate images. Statistical properties are then calculated over the three resulting images and used as features.

Another example of image techniques applied to spectrograms is found in [99], where the authors use HOG descriptors with the purpose of extracting information about the spectrogram's shape, in order to perform audio event classification.

In [100], a modified version of the LBP, an image feature for texture estimation, is proposed for audio classification. The authors point out that, given the different statistical properties between spectrograms and images (e.g. translational invariance), textures are better suited to capture the information in spectrograms.

An interesting aspect to note about this trend is that the employed feature vectors grow in dimensionality with the availability of better classifiers and more processing power.

Recently, while deep learning has not been applied as much as in images, DNN architectures have been applied successfully to many tasks of the audio domain, establishing new state-of-the-art performance levels.

In the context of ASR, DNNs have been applied as an end-to-end solution [101] and have been combined with more traditional techniques such as HMM [102]. ASR architectures often exploit some kind of RNN, sometimes combining them

with convolutional layers (or evolution thereof) for feature extraction.

Most of the results obtained in ASR are still valid for speech synthesis, i.e. a prevalence of recurrent architecture is used in today's systems. A different approach to TTS was proposed in [3]. The model, Wavenet, is composed of a CNN leveraging residual blocks and dilated filters, discussed in Section 3.1.1 and learns a probability distribution over raw audio values (i.e. 16 bit samples). Despite reaching high levels of naturalness, the authors stated that the complete model is computationally too expensive to be run in real-time. Furthermore, the exact architecture of the model was not made public.

As previously said, audio event recognition and classification is becoming an increasingly popular topic. Many deep learning-based approaches have been proposed in literature to solve this problem. In [103], biologically-inspired auditory features are combined with image features of the spectrogram to train a SVM and a DNN classifier, showing that the DNN outperforms the SVM.

In [104], Deep Belief Networks (DBNs) are used to extract features from the STFTs of the audio events, calculated at various time resolutions. The learned features are compared to traditional audio domains descriptors such as MFCC and Linear Predictive Coding coefficients, showing that the DBN yields better performances.

While the earlier approaches tried to combine audio features with DNNs, architectures that leverage less processed data have recently been proposed. In [105], a Convolutional Neural Network (CNN) is trained using MFCC coefficients to detect environmental sounds.

A multi-resolution approach to acoustic event detection is presented in [106]. Multiple spectrograms are extracted and used to train a CNN in order to characterize events with different spectral features.

An architecture for audio event detection based on RNN is presented in [107], where the authors perform recognition of sound events overlapped to other background events using Mel coefficients and a LSTM.

In [108], the authors propose an unsupervised method for acoustic novelty detection based on denoising autoencoders and a bi-directional LSTM.

As can be seen, deep architectures are being used in the audio domain with increasing success. However, while in most DNN applications the data is merely pre-processed, without extracting any kind of features, in the audio domain analysis of spectrograms and MFCC coefficients are still prevalent and outperform coarser approaches.

4.3.2 Audio classifiers

HMMs have long been applied to audio processing and to ASR in particular. A HMM is a Markov model, that is a probabilistic graphical model that satisfies the Markov assumption. More precisely, in a Markov model, a system state is modeled through a chain of states. Each state allows transition to a limited number of other states, defined by transition probabilities P_t that are non-zero if the transition is allowed. Since the models satisfies the Markov assumption, the model can be specified completely through the transition probabilities

from one state to the other. Figure 4.3 shows the graphical representation of

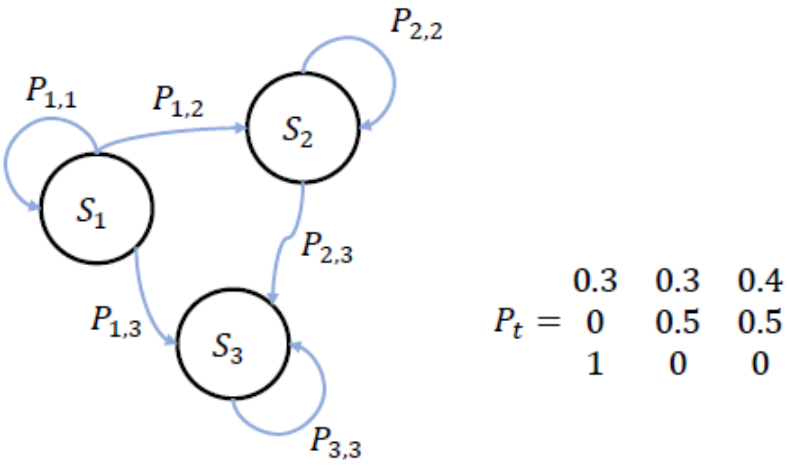


Figure 4.3: Architecture of a simple Markov model

a Markov model. The architecture can be alternatively described through a matrix M in which $m_{i,j}$ encodes the transition probability from the i_{th} to the j_{th} state.

An HMM is a Markov model in which the state is not directly observable (i.e. hidden) and thus the state of the system given the observable variables is stochastic. Therefore, in order to fully specify an HMM, the transition probabilities must be specified as well as the conditional probabilities $P(s/\mathbf{O})$ of states s given observed variables $\mathbf{O}$. The structure of an exemplary HMM is shown in Figure 4.4

In the context of ASR, the speech is modeled by a sequence of phonemes that constitute words. Each phoneme is modeled by a single HMM. The emission probabilities (i.e. the conditional probability of a state given an observed vari-

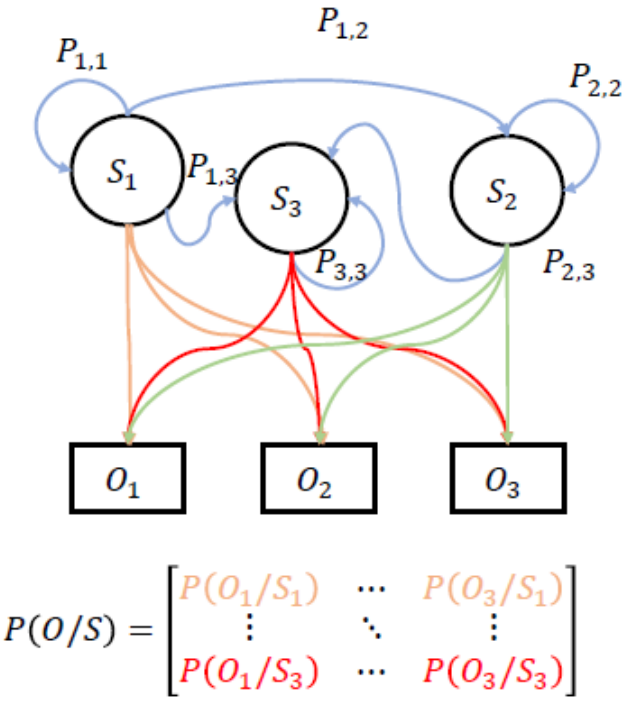


Figure 4.4: Architecture of a simple hidden Markov model

able) is modeled through a Mixture of Gaussian probability density function. In this context, the learning phase yields the parameters of the model, the state transition probabilities and the emission probabilities.

4.3.3 Data augmentation for audio signals

As said in Chapter 3, data augmentation can yield a larger effective dataset without the costly process of acquiring more data. While dataset augmentation in the audio domain is somehow less common than in images, some recent contributions show that this is a feasible option also with sound.

In [109], the authors propose augmentation techniques in the general framework of audio event classification. The proposed techniques are:

- Stretching the sample time support (time stretching)
- Shifting the height of the pitch (pitch shifting)
- Compressing the dynamic range of the sample
- Additive structured noise (i.e. background audio events)

The authors apply the proposed techniques on the UrbanSound8K [110] dataset, demonstrating that the use of augmentation allows a CNN to perform better than a non-DNN model. However, the authors note how augmentation can worsen the performance on very similar classes (e.g. air conditioner noise vs drill noise).
In [111], Vocal Tract Length Perturbation (VTLP) is introduced in the context of ASR. VTLP non-uniformly warps the frequency axis by a factor α in the interval [0.9 1], causing a larger distortion in frequencies that do not interest human voice. In [112], the authors propose another augmentation technique specific to ASR. More specifically Stochastic Feature Mapping is performed by mapping speech utterances from one speaker to another by estimating an acoustic model for each speaker and then applying it to the utterance.

4.4 Audio classification for safety applications

Surveillance systems play a critical role in ensuring safety. Sensitive targets are increasing in number, making it impossible to ensure safety through human operators alone. Nevertheless, to issue a proper response, reliable and fast detection of critical events must be ensured. In this way, command centers can effectively tackle emergencies. Since this requirement cannot be satisfied by manpower alone, modern surveillance systems rely on multimedia systems that allow to monitor wide areas from few stations. This approach enables better situational awareness for on-field teams, improving effectiveness and reducing the costs of the system as a whole, since fewer human resources are needed.

To date, surveillance applications rely mostly on visual media gathered by distributed camera systems, monitored by operators in control centers. In wide areas, cheap hardware is often used since many sensors are necessary to cover them, thus resulting in noisy, low resolution data. Furthermore, operators notoriously lose focus over long work shifts, introducing the risk of missing significant events.

Audio sensors, on the other hand, represent an opportunity, since good quality audio signals can be obtained with low cost hardware. Audio-based distributed surveillance has been seldom employed however, since operators cannot effectively monitor more than a few input streams at once. For this reason, automatic recognition of audio events in the domain of surveillance has become an interesting field of research, as it enables the monitoring of large areas while

addressing at the same time the operator's attention problem (since the system can issue an alert if it detects an anomaly).

However, in order to be used in practical applications, an audio event detection system must satisfy different key performance indicators. Timeliness of the recognition is critical: if the system is not able to work in real time, the audio stream will have to be decimated to keep processing newly available data. Decimating the audio input for reducing the computational complexity causes loss of information that can impact on the accuracy of the classification.

High detection and classification accuracy (i.e. comparable to human ones) are necessary especially in the sense of low False Rejection Rate (FRR), that is, being able to detect as many significant events as possible.

Finally, these performances must be obtained also in the presence of noise, since surveillance hardware is often of poor quality, as previously stated.

4.4.1 Previous work

In literature, the standard approach to audio event recognition is to reduce the dimensionality of the input data by extracting a set of features and then training a classifier to perform the recognition. For example, in [113], the authors use Mel Frequency Cepstral Coefficients (MFCC) as features and a set of binary Support Vector Machine (SVM) classifiers, obtaining multi-class classification by exploiting a decision tree. In [114], the authors propose a hierarchical sys-

tem where the events are modeled with Gaussian Mixture Models (GMM) optimized through a combination of temporal and spectral domain features. In [115], the authors propose a system employing two GMM-based classifiers trained with different sets of features for detecting screams and gunshots, using the delay differences extracted from a microphone array to estimate the position of the event.

In [116], the authors use a GMM-based classification, optimized on features extracted from the cepstral and spectral domain, considering additional classes of background events to improve the system's performances.

The topic is further explored in [117], where the authors propose a hierarchical approach to build features, useful to discriminate background events from relevant ones exploiting their short and long-time variations. A similar approach is used in [118] to detect anomalous audio events for road surveillance purposes.

DNNs represent a good tool for the purpose of audio surveillance. While they require a heavy computational cost during the training phase, the classification procedure is computationally fast. This factor, coupled with a reduced need for data pre-processing, allows the system to have good performances in terms of timeliness. An attempt at using DNNs specifically for audio surveillance is shown in [119], where the authors combine MFCC with a deep belief neural network to detect screams.

In the following Section, a method for audio surveillance based on hierarchically organized LSTM cells is presented.

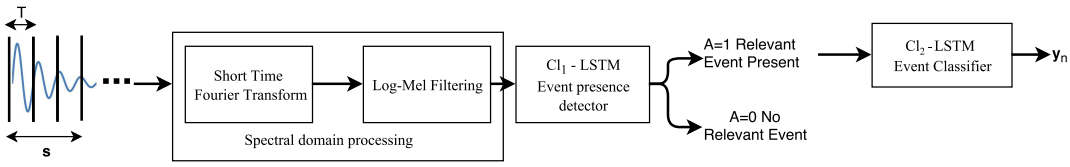


Figure 4.5: Block diagram of the proposed method

4.4.2 Proposed method

The goal is to detect, from an audio signal $\mathbf{s}$, the presence of events belonging to a selected number n of classes. To this aim, $\mathbf{s}$ is processed as shown in Figure 4.5. In more details: $\mathbf{s}$ is partitioned in non-overlapping time windows of length T s and, for each window, the STFT is calculated over N samples with a step of N_s samples. The output of the STFT is a set of T_f frames and, for each frame, the magnitude of the spectrum is filtered using a Mel filter-bank composed by M filters, resulting in a M by T_f signal, $\mathbf{x}$.

Two LSTM cells, Cl_1 and Cl_2 are fed by $\mathbf{x}$.

The classifier Cl_1 detects the presence of a relevant event, and outputs a boolean value, A. If $A = 1$, an alarm is raised and $\mathbf{x}$ is fed to the second classifier Cl_2 that gives a class label $\mathbf{y}_n$ for the ongoing event. Both LSTM cells are composed by N_h units. The outputs of the two cells, $\mathbf{h}_{Cl_1}$ and $\mathbf{h}_{Cl_2}$, are used as features for two densely connected layers, which give A and $\mathbf{y}_n$ as output, respectively. If $A = 0$, no alarm is given, otherwise an alarm is raised (i.e. $A = 1$) and a classification label $\mathbf{y}_n$ for the event is given.

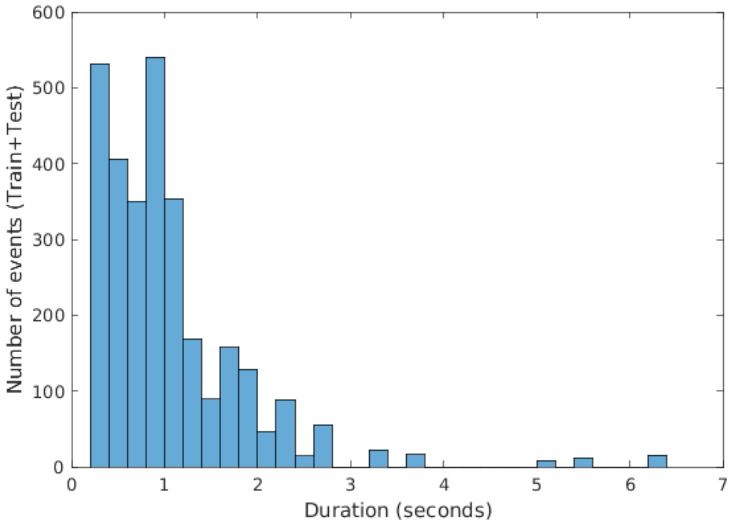


Figure 4.6: Distribution of event duration in the dataset

4.4.3 Validation

In order to assess the performances of the proposed method, experimental tests and comparisons with the state-of-the-art have been performed.

Dataset

In the performed tests, the dataset presented in [117] was used. The dataset contains 3000 audio clips, digitalized with a sampling rate of 32 kHz, and 16 bits per sample. The dataset is generated by combining 271 sounds belonging to three classes of events of interest (glass breaking, gunshots and screams) with 379 instances of background noise (i.e. cars passing by, crowd noise, ...). Furthermore, six versions of the dataset with different Signal to Noise Ration (SNR) values (5, 10, 15, 20, 25 and 30 dB) were created. The dataset is split

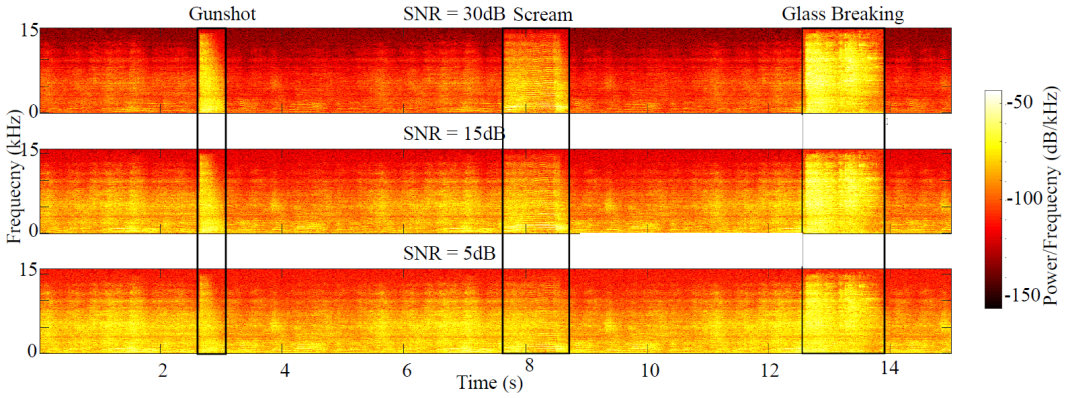


Figure 4.7: Spectrograms of events extracted from the dataset for different SNR values

into a training set of 2100 events and a testing set of 900 events. The duration of the events in the dataset is highly heterogeneous, as shown in Figure 4.6. The number of instances used to generate the dataset represents an obstacle to the use of a DNN. The generation process can be considered a procedure of data augmentation, aimed at teaching the classifier invariance to background events, but does not give additional data for the training. Thus, there is a strong risk of over-fitting the training set.

Experimental parameters

In the performed experiments, the length of the time window is set to $T = 3.23$ s, while the STFT is performed with windows of $N = 1024$ samples (i.e. 0.032 s) with a step size of $N_s = 256$ samples (i.e. 0.008 s). A Mel filterbank of $M = 40$ filters is used, thus resulting in an input vector $\mathbf{x}$ of 40 x 400 sam-

ples. This time resolution allows to completely include the largest part of the considered audio events, as shown in Figure 4.6. Longer events are handled by dividing them in portions of equal length.

The first LSTM classifier, Cl_1, is used for automatically detecting the need for raising an alarm. In order to do this, Cl_1 is trained using 4100 sounds: 2000 random portions of background noise extracted from the training set, and the complete set of training events. The classifier is tested using background noise and significant events from the test set.

The second classifier, Cl_2, is trained with the three classes of significant events (glass breaking, gunshots and screams) in the training set. Furthermore, as validation set, 10% of the test set is used. Both Cl_1 and Cl_2 contain 90 units and the Cl_2 dense layer used a softmax activation function. Both classifiers were trained using the Adam algorithm, described in Section 3.2.2, using a learning rate η of 10^{-4} and parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\epsilon = 10^{-8}$, with a batch size of 100 for Cl_1 and 50 for Cl_2. The training is run for 15 epochs for Cl_1 and 40 epochs for Cl_2 with early stopping based on validation set accuracy.

Results

The architecture is tested in three sets of experiments. In the first set of experiments, the system is trained and tested in matched conditions (i.e. with data having the same SNR values), in order to be able to compare the performances with the ones in [117]. In the second set of experiments performed, the

Dataset SNR value	Foggia et al. [117]	Proposed method
5 dB	81.1%	90.7%
10 dB	85%	92.4%
15 dB	87%	98.5%
20 dB	88.4%	98,7%
25 dB	88.7%	99.1%
30 dB	90%	99.9%

Table 4.1: Accuracy percentage for proposed method compared with the state-of-the-art

classifier is trained for each available SNR values, and each classifier is tested with data characterized by low, medium and high SNR values (i.e. 5, 15 and 30 dB) to evaluate the resilience to different levels of noise.

The third set of experiments is performed to evaluate the accuracy of the classifiers when only a portion of the audio events is available. Three SNR values (i.e. 5, 15 and 30 dB) are considered, and three test sets are generated for each value: in more details, the 25%, 50% and 75% of the original signal is discarded at random. Thus, 9 test sets are generated. Based on the previous results, in this experiment the classifier trained with SNR value equal to 15 dB is used.

Furthermore, to verify the real-time capabilities of the system, the time needed to process a time segment T is measured on a PC equipped with an Intel Core i7 5960X CPU, 64 GB of RAM and a Nvidia Titan X GPU. Finally, to assess the advantage deriving from the hierarchical architecture, a single-step classifier Cl_s is trained with four classes (i.e. the three classes of interest and the background events). The single classifier is trained with 15 dB data, with the

Training SNR value	**Test SNR value**	**Accuracy**
5 dB	5 dB	90,7%
	15 dB	95,7%
	30 dB	97,6%
10 dB	5 dB	88,2%
	15 dB	92,5%
	30 dB	92,8%
15 dB	5 dB	90.1%
	15 dB	98.5%
	30 dB	99.7%
20 dB	5 dB	85,8%
	15 dB	97,6%
	30 dB	98,7%
25 dB	5 dB	80,1%
	15 dB	97,4%
	30 dB	99,3%
30 dB	5 dB	83,9%
	15 dB	98,9%
	30 dB	99,9%

Table 4.2: Accuracy across different levels of noise

Testing SNR level	Percentage of discarded input		
	25%	**50%**	**75%**
5 dB	73.3%	60.8%	55.2%
15 dB	80.4%	71.7%	61.5%
30 dB	80.6%	72.9%	61.7%

Table 4.3: Classification accuracy with partial events

same hyper-parameters used for Cl_2. For the first set of tests, Cl_1 reaches 100% accuracy on each available SNR value, thus meaning that, in these conditions, the system never fails to raise an alarm and never raises it when it should not. The accuracy achieved by Cl_2 is reported in Table 4.1 and it is compared with the performances of the system in [117]. As it can be seen, the proposed method shows a significant increase in the accuracy with respect to the state-of-the-art. A notable difference in the performances is present for SNR values <15 dB. This can be explained by analyzing the spectrograms in Figure 4.7. It can be seen that for these range of SNR values, the background events overlap the signal at all frequencies, thus explaining the drop in accuracy. A similar pattern can also be found in the results from [117].

Table 4.2 shows the accuracy results of the second set of experiments. The classifiers exhibit different resilience performances depending on the noise with which they have been trained. Some fluctuations in the results are present due to early stopping, that can sometimes interrupt training prematurely. Moreover, exposing the classifiers to noise levels that largely differ from the ones available during the learning phase has an impact on the overall performances. Based on the results obtained, it can be concluded that the classifiers trained

with data with moderate noise values yield the best overall performance.
In the third set of experiment, Cl_1 reached again 100% accuracy. On the other hand, the performances of Cl_2 were strongly influenced by the removal of portions of the events. In fact, as can be seen in Table 4.3, in average the accuracy drops of 20% for all SNR values.
The average processing time for segments not containing events of interest (i.e. preprocessing and Cl_1 classifier) is 16.2 ms, of which 14.8 ms where due to preprocessing and 1.4 ms where due to Cl_1. The Cl_2 classifier added 1.6 ms to the process, achieving a total mean processing time of 17.8 ms. Thus, it is feasible to classify multiple audio streams in real-time using a single computer.
The single-step classifier Cl_s scored 95.0% accuracy, with a mean classification time of 2.1 ms. The proposed hierarchical architecture offers a small advantage in terms of accuracy and processing time when classifying background time segments.

4.5 Conclusions

In this chapter, the applications of machine learning to the audio domain were presented, introducing the basic elements of audio analysis and the most successfully employed features and algorithms.
Furthermore, a novel application of DNNs to the field of audio surveillance was presented, that sensibly improves the performance of the state-of-the-art. The proposed method represents a considerable improvement over the state-

of-the-art.

Firstly, the proposed system improves performances in terms of detection and classification accuracy. The detection step demonstrates optimal performances. Furthermore, it exhibits robustness to structured noise (i.e. the presence of other audio events) as well as to cropping of the input signals. The classification step is able to distinguish between different critical events, with almost perfect accuracy under the absence of noise. While structured noise degrades the performances of the system, high accuracy (i.e. greater than 90%) is maintained even in the worst case.

The second advantage of the system is timeliness. It has been demonstrated that an audio signal with good resolution (i.e. 32 kHz sampling frequency, 16 bits/sample) can be processed in real time without decimation or compression. Furthermore, the timing performances are an indication that multiple input audio streams can be processed on a single machine.

Considering the domain of application however, the system could introduce vulnerability to adversarial inputs. Albeit the literature presents mostly visual examples (easier to implement in the physical world), the existence of an adversarial audio noise, able to mask the presence of relevant events cannot be excluded. However, as shown in Chapter 3, this problem is present in any machine learning-based system.

Another issue in the system is the handling of the audio input: since the time segments T are not overlapping, there is the risk of capturing only portions of events. As shown in the results, this can hinder classification accuracy, espe-

cially in the worst case scenario in which a time window contains roughly half of the event. Taking overlapping windows can be a solution to this problem, albeit it worsens the timeliness performances.

Finally, testing the system on a larger dataset is necessary to evaluate its usability in the real world, as the currently available datasets could not contain a satisfactory representation of the classes. The proposed system also offers opportunities for future improvements. Two main directions can be considered. First, in order to improve the classification performances, convolutional feature extractors, described in Chapter 4, could be used in place of the Mel filterbank. As shown in Section 4.3, there are hints in literature that convolutional layers can extract meaningful features directly from spectrograms, especially when coupled with a RNN. However, in spite of the potential gain of classification accuracy, computational complexity would rise as well (in terms of memory usage). Thus, the increment in accuracy would be obtained by reducing efficiency.

Secondly, the system could be employed as part of a multi-modal smart surveillance system. In more details, in order to get more accurate and specific information about the event, information from a broader scope audio event detector could be coupled with an activity recognition video system. While this would result in a more complex and costly system, since good quality cameras would be needed as well as more processing power, important information could be obtained by the analysis of video cues.

Chapter 5

Applications to images and videos

5.1 Introduction

In this Chapter, the visual domain and its intersections with machine learning will be discussed.

This Chapter investigates the use of semantic attributes as an enabling factor for algorithms that replicate tasks currently performable only by human beings. In more details, the feasibility of automating the combination of multiple videos into a final *edited* version based on their estimated aesthetic content is investigated. This is a complex task, since it combines multiple perceptual factors that lack a clear definition, as will be shown in the following sections.

The fundamentals of the fields will be introduced in a functional way for the subsequent Sections. Section 5.2 introduces the fundamental elements of the image/video domain, while Section 5.3 discusses applications of machine learning in the visual domain. Since many applications have already been cited in Chapter 3, Section 5.3 focuses on the aesthetic semantic attributes.
Finally, Section 5.4 introduces the aforementioned application of aesthetic estimation for unsupervised video orchestration, originally proposed in [121].

5.2 Digital images fundamentals

A digital image is a visual representation of a scene. The first step in processing an image is acquisition. The most important parameters of a digital image are the spatial resolution and the per-pixel resolution. The first one controls how many points are acquired when the picture is taken, and is measured in pixels (vertical resolution times horizontal resolution).
The per-pixel resolution represents the number of bits per pixel used in quantizing the values of the grid, analogously to what described for audio samples in Chapter 4.
A digital image can be considered as a multi-channel array $I(m, n, c)$, where m and n represent respectively the number of rows and columns (i.e. the spatial resolution used to acquire the image). The number of channels c is dependent on the type of image that is being analyzed: a greyscale image will have $c = 1$ while color images usually have $c = 3$. The information encoded into the chan-

nels is different depending on the color space used to represent the image.

Historically, one of the most used color models is the RGB, with the three channels corresponding to the red, green and blue components. This model is based on the Young-Helmholtz color model which assumed that human vision is based on three receptors in the eye (cone cells), sensitive to the red, blue and green wavelengths. Even though cone cells sensitivity does not actually peak in those spectrum regions [1] [122], this model proved effective, also enabling color displays with a wide color gamut, and is widely used.

Other color spaces exist as well. The YUV refers to encoding of the color information into a luminance component (Y) and two chrominance components (U, V). YUV color spaces are motivated by perceptual models of vision and are used in applications such as lossy compression, where perceptually less significant chrominance components can be sub-sampled in order to lose less quality from a perceptual point of view.

Hue Saturation Values (HSVs) are cylindrical coordinates-based color spaces that have been developed to have a more perceptually intuitive representation of color. In more details, the color is specified through the H angle value, that specifies the tone while S gives the saturation. Finally, the brightness value V gives the light intensity. This separation can be useful in image processing as it enables invariance to illumination condition.

Describing videos is in many senses analogous to describing images. A video can be thought as a sequence of images and thus shares the same attributes for

[1] 564580 for long cells, 534545 for medium cells and 420440 nm for short cells, corresponding to yellow, green and violet

spatial resolution and per-pixel resolution. However, the temporal resolution f_s must also be considered (i.e. the frame rate). In general, since the human eye acts as a low pass filter with cut-off frequency around 25 Hz, having 25-30 frames/s is considered a good frame rate.

While a video is effectively a sequence of images or frames, the frames have a strong degree of correlation, that is often exploited for compression purposes. Most compression methods are based on the reduction of the temporal redundancy between adjacent frames exploiting the motion estimation (ME). ME refers to the evaluation of object displacement between successive frames through the evaluation of the object's apparent motion or Optical Flow (OF). ME achieves compression by predicting the motion and encoding the prediction error, thus eliminating the need of storing/transmitting each frame.

5.3 Images, videos and machine learning

As mentioned in Chapter 2, multimedia features can be classified in syntactic and semantic. In the field of image/video in particular, the approach of considering high-level or semantic attributes of images as particular combinations of lower level features has been used extensively. A common approach is to partition the image in blocks and then locally compute specific features, finally inferring high-level properties by statistic on the features distribution (e.g. histograms). The features are developed to capture specific properties

of the image region, whose distribution is assumed to hold information about the image's semantic content. As an example, one can assume that a region of interest in an image is characterized by a different distribution of edges and patterns.

Before the introduction of DNNs, most of the research on images and video machine learning was dedicated to developing more accurate features. One of DNNs main advantages is the ability to build hierarchical representations starting from simple filters in the upper layers, obtaining semantically significant deeper neurons activations.

Most of the advancements in machine learning applied to vision have been summarized in Chapter 3. As a matter of fact, image classification represents the main task on which new machine learning techniques are benchmarked nowadays. Other computer visions tasks, such as object segmentation, represent the immediately successive testing ground.

Given that DNNs can effectively solve many high-level vision tasks, more advanced contexts are becoming the focus. A domain that is only recently been studied is aesthetic computing.

Aesthetic computing deals with the automatic estimation of the aesthetic value of images and videos. It is useful to underline that, while human beings are generally sensible to aesthetics (i.e., they have an intuitive distinction of what is aesthetically pleasing and what is not), the precise evaluation of the aesthetic is a challenging task even for a human subject. Determining which elements contribute to aesthetic is even more difficult. In general, beauty is subjective

and, even among experts, there is a degree of subjectivity.

5.3.1 Previous work

While a correlation with the quality of the media exists, as shown in [123], this is not the only component.

In literature, many approaches rely on machine learning: in [124], the authors attempt to estimate the aesthetic value through low-level features, while a visual saliency map is introduced in [125].

In [126], the authors exploit the differences between background and foreground to estimate an appeal metric.

In [127], psycho-visual statistics extracted at different semantic levels are used.

A comparison of aesthetic models is performed in [128], taking into account the different experimental environments and a rating scale is presented. In general, the application of machine learning to aesthetic computing is a challenging task since it is difficult to obtain a good dataset. The most used datasets rely on crowdsourced labels for photos, such as the AVA dataset [129] or the CUHK [130] dataset. Furthermore, many of the techniques developed for image classification assume properties that may not hold (or at least, not to the same degree) in aesthetic estimation. As an example, invariance to translation could not be a desideratum: many rule-based approaches to aesthetic

exploit the position of objects in the scene, such as the rule of third [2].
Basically, the estimation of the aesthetic value of a visual content is an open problem. However, the current techniques can be exploited for practical application.

5.4 Video orchestration based on semantic features

As mentioned in the Introduction, a large amount of multimedia content is being created every day. This is particularly true for video contents: an estimated 300 hours of videos were being uploaded on Youtube alone, as of September 2016 [131]. In case of events of public interest, such as concerts and sport events, multiple footages are usually available, since many people own a video recording device, usually incorporated in their smartphones.
Automatic orchestration of videos is thus an interesting application, since it produces a single content, presumably more pleasant with respect to the initial videos.
This problem is also faced by professional video-makers whose goal is to avoid missing important actions or scenes by exploiting events captured with multiple cameras. In literature, methods for combining shots from different cameras have been proposed. In [132] an orchestration scheme for remote video-

[2]The rule of third is an empirical photography rule, stating that the content of the image should be divided into 9 blocks, thus dividing the image into thirds

conferences is proposed. The system aims at automatically detecting the best camera from each of the locations in the conference. In [133] the authors exploit editing rules concerned with continuity to perform video editing in the context of interacting videos.

In the following sections, an unsupervised video orchestration system, first proposed in [121], is discussed. It relies on aesthetic features coupled with rules extracted from traditional video editing theory. The goal is to produce a video by exploiting the shots that in each time interval look more appealing from an aesthetic point of view, while respecting the orchestration guidelines described in [134]. This system can be useful for automatic video orchestration of amateur videos ans a pre-processing step for media production.

5.4.1 Proposed Method

In the following, the term *scene* refers to the event being filmed. The term *shot* refers to the output of a single camera while *sub-shot* is used to indicate a part of the shot.

For all available video inputs, the position of each camera with respect to the recorded scene is described through the attributes listed in Table 5.1. Since different cameras may have a different acquisition setup (i.e., sampling rate or frame size), in the proposed system the K contributions are normalized with respect to the camera having the lowest frame rate f_s and the smallest frame size. The output video $\mathbf{V}$ is composed by N time-slots whose duration

is determined based on the video content, as described in Subsection 5.4.1; for each time-slot, the system selects one shot among the available K cameras. Since there is usually a strong temporal stability in amateur videos (i.e. no abrupt scene change is present), N_f representative frames $\mathbf{R}_f$ are used for each second of the input videos, in order to reduce the computational complexity. While video editing is in general strongly dependent on the content and on the experience of the editor, there are guidelines in video editing theory [134], [135] for orchestrating multiple video sources. Creating an entertaining and dynamic experience for the viewer is the focus of video editing. General rules derived from this principle are defined in [134] and can be summarized in the following:

- Alternation of shot types and camera angles should be used to create dynamism.
- Scenes should remain on screen proportionally to their information content.
- The angle of the camera should vary in at least 30 degrees between successive sub-shots.
- Too harsh variations of framing should be avoided as it creates confusion in the viewers.

Given these guidelines, the orchestration problem is articulated into two distinct phases: a first one to determine how often the video source should be

Shot Type	Very wide/Panoramic
	Wide
	Medium
	Close-up
	Point of view (POV)
Horizontal Angle (°)	0
	70
	140
	210
	280
Vertical Angle	Bird's eye
	High
	Eye-level
	Low
	Worm's eye

Table 5.1: Camera framing features.

changed (i.e. determining the duration of the time-slots); a second one to, choose the content of each time-slot, optimizing aesthetic values and compliance with the aforementioned rules.

Time-slot calculation

Basic rules of video editing state that changes of view should be used to create a more dynamic orchestration. Intuitively, more changes are needed when the scene has a slow pace. The pace of the scene is estimated by extracting the Optical Flow (OF) from each of the available K videos.

The average OF for the K cameras is computed to obtain an estimation of the average dynamics of the content, $\overline{OF}$. This estimate is used to partition the video in three categories according to the motion rate: low, medium, and

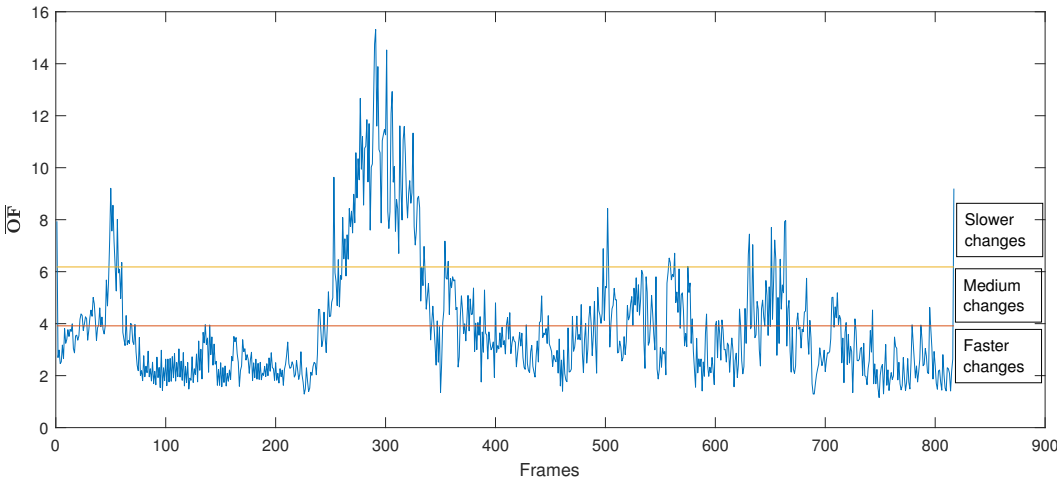


Figure 5.1: Motion rate partitioning: the frames with higher $\overline{OF}$ are considered faster and thus use less camera changes. Frames with lower $\overline{OF}$ are considered slower and thus have more camera changes.

high dynamics as shown in Figure 5.1. The length of the time slots is selected according to the $\overline{OF}$ value, as suggested in [134].

Time-slot content selection

To select the content of each time-slot, a multi-objective optimization is performed. The algorithm operates over a set of possible candidates $\mathbf{V}_c$, evaluating the fitness through three functions:

- Mean aesthetic score of the sub-shots $A(\mathbf{V}_c)$
- Rule-based evaluation $R(\mathbf{V}_c)$
- Diversity score $D(\mathbf{V}_c)$

Aesthetic score computation

Selecting the method to assess the aesthetic value is a difficult task. On the one hand, aesthetic is a high level attribute, with a strong dependence on the content: it is foreseeable that elements that give aesthetic value to a portrait will not be valuable in a landscape.

In the previous work [136], orchestration through a content-dependent features was experimented. While the results were promising, the computational complexity proved very large. Since the multi-objective optimization requires multiple calculations of the aesthetic value, as is detailed in the following, this approach proved incompatible with the system as a whole. Instead, to assess the aesthetic value of a sub-shot, the approach proposed in [137] is used due to its low computational complexity. A subset of images from the CUHK dataset [130] is selected and, based on their labels, organized in two categories, high and low-quality, to create a training set.

For each image, a 24-d feature vector is calculated and, based on the comparison with the high and low-quality images in the reference dataset, the following information is calculated:

- Color palette (f_1): the color palette evaluates the color scheme in the HSV color space. A clustering is performed over the values of the color histogram and the calculated centroids are considered as the dominant colors. f_1 is obtained by comparing dominant colors in the image under analysis with the ones extracted from the training set.

- Layout Composition ($f_2 - f_5$): a layout template is extracted from the available high and low-quality images by averaging the value of their pixels over four channels (H,S,V and H+S+V). The features are obtained by calculating the L_1 distance of the image under analysis, d_H and d_L respectively, from the templates.

- Edge Composition ($f_6 - f_9$): the edge features are obtained by averaging the edge information of the high and low-quality images to obtain reference templates. Image features are extracted by subtracting the L_1 distances from the templates.

- Global texture ($f_{10} - f_{17}$): these features are generated by dividing the image into 6 stripes and computing the sum of the differences of adjacent stripes.

- General features ($f_{18} - f_{24}$): features evaluating the amount of blur, contrast, and the number of non-zero elements in the HSV quantized histogram and the dark channels [138].

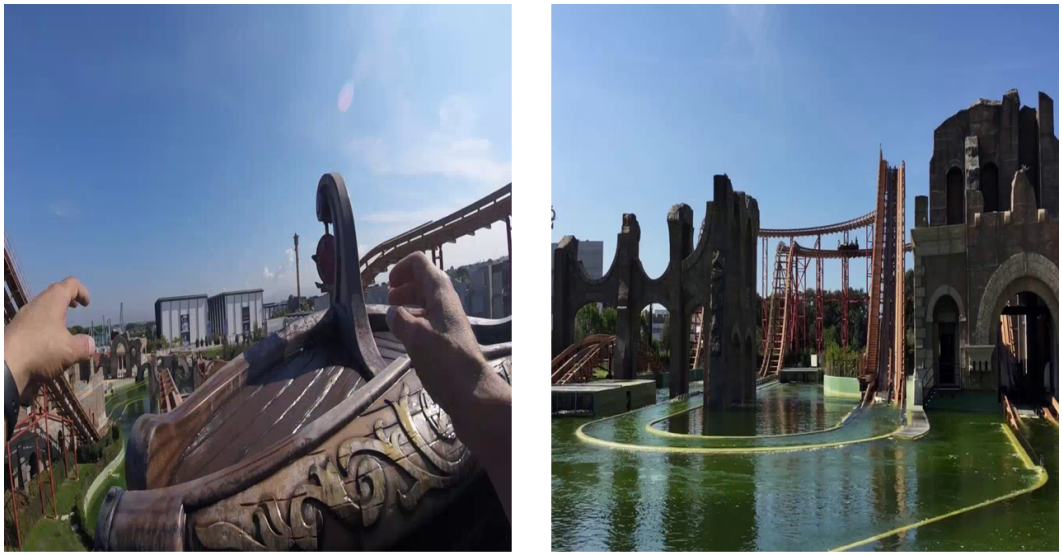
The resulting feature vector is classified with a SVM. In this work, the aesthetic score of a sub-shot is evaluated by classifying the features extracted from the $\mathbf{R}_f$ frames in probabilistic mode, assigning a default label of high quality. The classifier returns the predicted class together with the confidence level, expressed as a probability. The confidence level is used as aesthetic score.

Rule-based evaluation

The purpose of $\mathbf{R}(V_c)$ is to evaluate the smoothness of the transitions in V_c. In order to do this, V_c is modeled as a Markov chain over the possible framings of the scene. Each state (i.e. framing) is characterized by the attributes used to tag the cameras in Table 5.1. The distance between two states is given by the sum of the differences in the values of the considered attributes. Nearby states will be cameras with similar features. Transitions to nearby states have a high probability, while lower values are given to *far* states (i.e. very different framing). Based on the cinematographic rules, the probability of remaining in the same state is chosen to be low ($< 10^{-4}$). The distance between states is determined adaptively depending on which type of camera attributes are given as input: a panoramic shot and a POV shot could be considered near only if no other intermediate shot type (e.g. medium shot) is given. In this way, the system prefers smooth transitions in the view, avoiding harsh scene changes that would be annoying for the viewer. The value of $R(\mathbf{V}_c)$ is given by the probability of the correspondent path on the Markov chain:

$$R(\mathbf{V}_c) = \sum_{i=1}^{N-1} \log(P(\mathbf{V}_c(i), \mathbf{V}_c(i+1)))$$

where P is the probability of transition from $\mathbf{V}_c(i)$ to $\mathbf{V}_c(i+1)$.



(a) (b)

Figure 5.2: Example of two possible shot types: a POV (a) and a panoramic shot (b)

Diversity evaluation

In a real scenario, it can happen that one of the videos has an overall higher aesthetic score than the others. This may be due to several reasons, such as one of the operators being more skilled or be in a better position with respect to the others. In this case, performing an optimization based exclusively on $A(V_c)$ and $R(V_c)$ would result in the exclusion of a consistent number of video sources.

Preliminary tests carried out with 10 experts, have shown that a video obtained as orchestration of diverse inputs is preferable to one composed by combining a smaller number of cameras, even if they are more valuable in aesthetic. For this reason, a third function is introduced in the optimization problem,

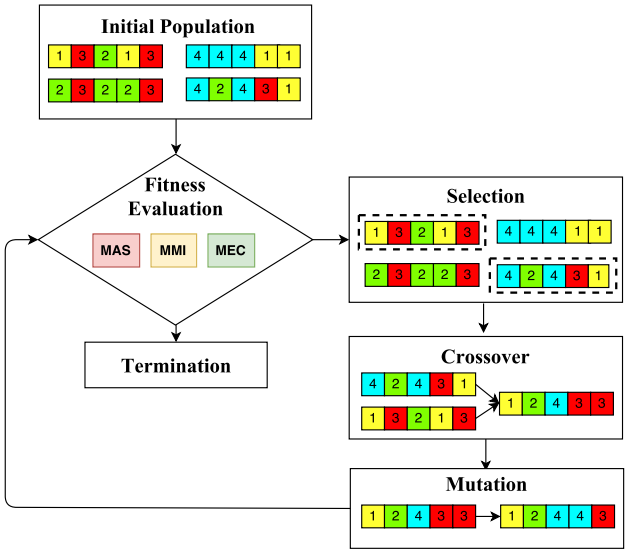


Figure 5.3: Multi-objective genetic optimization of the editing

the diversity score, to take into account and penalize the unbalanced use of cameras. The diversity score $D(\mathbf{V}_c)$ is calculated through the following steps: first, the cameras' empirical probability distribution $P_{\mathbf{V}_c}$ is calculated. $P_{\mathbf{V}_c}$ is then compared with a uniform probability distribution, P_u, over the K cameras through the Kullback-Leibler (KL) divergence:

$$D_{KL}(P_u, P_{\mathbf{V}_c}) = \sum\nolimits_{i \in K} \ln \left(\frac{P_u(i)}{P_{\mathbf{V}_c}(i)} \right) P_u(i).$$

Multi-objective optimization

The goal of finding an optimal $\mathbf{V}$ can be formulated as a multi-objective optimization problem over the three previously defined functions, $A(\mathbf{V}_c)$, $R(\mathbf{V}_c)$,

$D(\mathbf{V}_c)$. This class of problem does not have a unique optimal solution. In fact, the solutions produced by a multi-objective optimization are Pareto-optimal, that is a solution where none of the involved functions can be optimized without degrading the others. Evolutionary algorithms are often used in this setup since they can find multiple non-dominated solutions with each iteration [139]. To find a set of candidate vectors $\mathbf{V}_c$, multi-objective optimization is performed using the Genetic Algorithm (GA).

The GA is inspired by natural selection processes, taking a populations of individuals with different sets of genes. In our case, the population is composed by candidate editing vectors $\mathbf{V}_c$, where a gene is the sub-shot contained in a time-slot. A fitness value is then calculated for each individual in the population, in our case using the $A(\mathbf{V}_c)$, $R(\mathbf{V}_c)$, $D(\mathbf{V}_c)$ functions.

Individuals with higher fitness scores are more likely to be used as *parents* in the crossover step. The next generation is calculated during crossover by combining the genes of the parents. The final step of an iteration is the mutation: genes can be changed to random values with a small alteration. The optimization problem described above can be formalized as:

$$\begin{aligned} &\text{minimize} \quad (-A(\mathbf{V}_c), -R(\mathbf{V}_c), D(\mathbf{V}_c)) \\ &\text{subject to} \quad 1 \leq \mathbf{V}_c(i) \leq K,\ i = 1, \ldots, N \end{aligned}$$

this approach can generate multiple locally optimal solutions. This feature is considered desirable, as multiple possible editing solutions can be examined by professionals, fine tuning the orchestration to a desired output.

Video set id	Cameras	$\mathbf{f_s}(fps)$	Length (s)	Res. (pxl)
1	4	29	6	1080x1080
2	3	29	10	1280x720
3	4	25	10	1080x1080
4	4	25	10	1080x1080

Table 5.2: Source videos used for the experiment

Time-slot duration fine tuning

The duration of each time-slot is determined without knowing its content. Nevertheless, after the optimization procedure, such duration can be fine-tuned based on the content of $\mathbf{V}$. More specifically, longer time-slots are assigned to cameras that are richer in details, since editing guidelines state that denser scenes take longer to become boring for the viewer.
In order to estimate the content density of a scene, the mean edge values E_d is extracted in the first frame of each sub-shot:

$$E_d(\mathbf{V}(i)) = \frac{1}{m * n} \sum_{i=1}^{n} \sum_{j=1}^{m} \nabla^2 f_{\mathbf{V}(i)}(1, i, j).$$

Where $f_{\mathbf{V}(i)}(1)$ is the first frame of $\mathbf{V}(i)$ and m, n are the width and height of the frame. For each pair of consecutive sub-shots $V_c(i)$, $V_c(i+1)$ the sub-shot denser in content has its time-slot length, t_d extended by Δt. Nevertheless, this adjustment is performed only if the shortened sub-shot length is still longer or equal to 1 second to avoid introducing excessively short sub-shots.

5.4.2 System validation

In order to verify the effectiveness of the proposed method, a preliminary test has been performed. Four different contents were considered and for each of them different views were taken. Table 5.2 gives specifications for the video sets used to test the algorithm. In this trial, subjective tests were performed with the cooperation of 16 experts in media production and movie generation. As described in [140], even if the results of expert viewing cannot be considered as a replacement of the results provided by a formal subjective assessment, they can be considered a valuable preliminary indication of the performances of the tested system.

Video set 1 portrays a roller coaster ride recorded from four different points of view, selected according to cinematographic rules: subjective point of view, action of the subject, details of the subject and panoramic view. Video set 2 contains the final moments of a concert, with three views: left side and close to the stage, left side and far from the stage and central and close to the stage. Video set 3 records from four points of view a scene in a cafeteria in which a guy is drinking a coffee. Finally, video set 4 plays a scene in which a bartender opens a bottle of wine. Video sets 1, 3, and 4 were recorded by using commercial mobile phones and a Go-Pro camera, while video set 2 is gathered from the internet. From each video content, by considering all the available views, two different orchestrations were performed: one based on the proposed algorithm and another one obtained by randomly merging the input views. This means that overall, eight contents were used in the

Video set id	Proposed (%)	Random (%)
1	68.8	31.2
2	56.3	43.7
3	18.8	81.2
4	25	75

Table 5.3: Percentage of preferences expressed during the subjective tests.

subjective experiment. For each video content, the two orchestrated videos were shown to each user, who was asked to select the one that, according to his/her experience was preferable. Furthermore, an interview was performed for collecting feedbacks. The display used for the experiment is a Full HD display. The parameters used in the experiment are: $N_f = 4$, faster, medium and slower change rates are set to 1s, 2s, and 3s respectively. For the GA, an initial population of 15 individuals has been used.

Results and discussion

From the results collected, reported in Table 5.3, an important feedback can be extracted. It is possible to highlight two different trends. For the video sets 3 and 4 the randomly orchestrated videos are preferred. This is mainly due to the fact that the values used for time-slot selection were fitter to deal with dynamic content characterized by fast scene changes, while video sets 3 and 4 are characterized by more calm scenarios, incompatible with fast scene changes. Hence, a preliminary analysis of the video content and of the motion rate should be performed in order to optimize the orchestrator performances, and to adapt the scene changing rate to the video dynamics. Information

about content also allows a semantic, template-based approach, similar to the one used in [137], for video-based aesthetic features. As an example, having a reference template of motion for action videos would allow to use an aesthetic fitness function leveraging also video information.

5.5 Conclusions

In this chapter, the applications of machine learning to the visual domain were presented.

An application of aesthetic value estimation is reported, performing unsupervised orchestration of videos based on multi-objective optimization. The proposed system can be used as a preliminary step in professional video editing as well as an end-to-end system for the general public.

It is worth to note how the system supports modularity for the optimization phase. Additional rules and constraints can be added to the optimization procedure to perform a more complex orchestration (e.g. favor scenes with significant facial expressions), however, the results of the subjective experiments demonstrate how the current model may be too simplistic for such task. In particular, the time-slot selection cannot be performed without accounting for the genre of the processed content. Given the complexity of the task, and the fact that the influence of many factors on the video quality of experience is not completely understood, it is foreseeable that similar properties will be discovered in future versions of the system.

A disadvantage of the system is the processing time. In its current implementation, the system does not support real time processing. As a matter of fact, generating a video can take up to half an hour for short sequences (i.e. up to ten minutes), depending on how many shot changes are performed and thus on the cardinality of V_c. Real time processing would be a very important feature, as it would enable the use of the system in contexts where human operators do not have the physical time to perform their job.

Many improvements for the system can be considered. First of all, the selection of time-slots duration must become a function of the genre. Sampling the time-slot durations from a probability distribution parametrized based on the genre could be a solution, but it would leave unsolved the problem of estimating the genre.

Furthermore, in the future, the applications of DNN-based models for aesthetic evaluation seems to be the best direction to take, as DNNs enable semantic level representation with a low computational complexity when performing inference. Thus, this would also be a step toward real time orchestration.

Chapter 6

Conclusion

In this thesis, the use of machine learning techniques for estimating and exploiting multimedia data has been addressed. As demonstrated in Chapter 4, DNNs enable the processing of data with greater level of accuracy and efficiency with respect to previous techniques. This is an enabling factor for many practical applications, such as smart audio surveillance.

However, as discussed in Chapter 3, those techniques are not fully understood. Phenomena such as adversarial examples represent consequences of the inability to throughly understand the learning process, and represent a threat to applications in security and safety. Investigation of these phenomena is fundamental both for a safe use of machine learning in security applications and for a better understanding of the reasons for deep learning effectiveness.

The estimation of complex semantic attributes, such as the aesthetic value,

enables the use of DSSs in tasks that could previously not be handled by automatic systems, such as video editing. The unsupervised video orchestration system presented in Chapter 5 represents an attempt to perform a semantically-dense task, normally reserved to experts whose work largely relies on experience. The effectiveness in performing these tasks is still limited at present times, as shown by the results presented in Chapter 5. However, these applications help investigating the relevance of the many semantic attributes involved in complex topics such as video quality of experience, an element that adds to their scientific value.

Overall, in this thesis it was shown that semantic data processing has made considerable advancements in recent years, advancements to which this thesis contributes. As discussed in the Introduction, there are many levels of semantic attributes. While effective techniques are now available for estimating simpler ones, such as the events in a sound, understanding and estimating complex ones and effectively leveraging them for practical applications remains an open problem and an interesting direction for future work.

Bibliography

[1] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems 25* (F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, eds.), pp. 1097–1105, Curran Associates, Inc., 2012.

[2] P. Isola, J. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," *CoRR*, vol. abs/1611.07004, 2016.

[3] A. van den Oord et al., "Wavenet: A generative model for raw audio," *CoRR*, vol. abs/1609.03499, 2016.

[4] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 1, pp. 886–893 vol. 1, June 2005.

[5] L. Wang and D. He, "Texture classification using texture spectrum," *Pattern Recognition*, vol. 23, no. 8, pp. 905–910, 1990.

[6] A. Allik, G. Fazekas, and M. B. Sandler, "An ontology for audio features.," in *ISMIR*, pp. 73–79, 2016.

[7] R. I. Minu and K. K. Thyagharajan, "Semantic rule based image visual feature ontology creation," *International Journal of Automation and Computing*, vol. 11, pp. 489–499, Oct 2014.

[8] M. Cimpoi, S. Maji, and A. Vedaldi, "Deep convolutional filter banks for texture recognition and segmentation," *CoRR*, vol. abs/1411.6836, 2014.

[9] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson, "CNN features off-the-shelf: an astounding baseline for recognition," *CoRR*, vol. abs/1403.6382, 2014.

[10] O. Russakovsky and et al., "ImageNet Large Scale Visual Recognition Challenge," *International Journal of Computer Vision (IJCV)*, vol. 115, no. 3, pp. 211–252, 2015.

[11] B. Kolosnjaji, A. Zarras, G. Webster, and C. Eckert, "Deep learning for classification of malware system call sequences," 12 2016.

[12] O. E. David and N. S. Netanyahu, "Deepsign: Deep learning for automatic malware signature generation and classification," in *2015 Inter-*

national Joint Conference on Neural Networks (IJCNN), pp. 1–8, July 2015.

[13] Y. LeCun, L. Bottou, G. B. Orr, and K.-R. Müller, "Effiicient backprop," in *Neural Networks: Tricks of the Trade, This Book is an Outgrowth of a 1996 NIPS Workshop*, (London, UK, UK), pp. 9–50, Springer-Verlag, 1998.

[14] J. R. Quinlan, "Induction of decision trees," *Machine learning*, vol. 1, no. 1, pp. 81–106, 1986.

[15] T. K. Ho, "Random decision forests," in *Proceedings of 3rd International Conference on Document Analysis and Recognition*, vol. 1, pp. 278–282 vol.1, Aug 1995.

[16] F. Nan, J. Wang, and V. Saligrama, "Feature-budgeted random forest," *arXiv preprint arXiv:1502.05925*, 2015.

[17] S. Ren, X. Cao, Y. Wei, and J. Sun, "Global refinement of random forest," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 723–730, June 2015.

[18] M. Ristin, J. Gall, M. Guillaumin, and L. V. Gool, "From categories to subcategories: Large-scale image classification with partial class label refinement," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 231–239, June 2015.

[19] S. Schulter, C. Leistner, P. Wohlhart, P. M. Roth, and H. Bischof, "Alternating regression forests for object detection and pose estimation," in *2013 IEEE International Conference on Computer Vision*, pp. 417–424, Dec 2013.

[20] A. Montillo, J. Shotton, J. Winn, J. E. Iglesias, D. Metaxas, and A. Criminisi, "Entangled decision forests and their application for semantic segmentation of ct images," in *Information Processing in Medical Imaging*, pp. 184–196, 2011.

[21] T. S. Toby and et al., "Accurate, robust, and flexible real-time hand tracking," pp. 3633–3642, ACM, April 2015.

[22] P. Kontschieder, M. Fiterau, A. Criminisi, and S. R. Bulo, "Deep neural decision forests," in *Computer Vision (ICCV), 2015 IEEE International Conference on*, pp. 1467–1475, IEEE, 2015.

[23] R. K. Vinayak and R. Gilad-Bachrach, "Dart: Dropouts meet multiple additive regression trees," in *Artificial Intelligence and Statistics*, pp. 489–497, 2015.

[24] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[25] Y. Bazi and F. Melgani, "Convolutional svm networks for object detection in uav imagery," *IEEE Transactions on Geoscience and Remote Sensing*, vol. PP, no. 99, pp. 1–12, 2018.

[26] T. Guofeng, C. Huairong, L. Yong, and Z. Kai, "Traffic sign recognition based on svm and convolutional neural network," in *2017 12th IEEE Conference on Industrial Electronics and Applications (ICIEA)*, pp. 2066–2071, June 2017.

[27] M. Schuster and K. K. Paliwal, "Bidirectional recurrent neural networks," *IEEE Transactions on Signal Processing*, vol. 45, pp. 2673–2681, Nov 1997.

[28] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Transactions on Neural Networks*, vol. 5, no. 2, pp. 157–166, 1994.

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

[30] F. Yu and V. Koltun, "Multi-scale context aggregation by dilated convolutions," *CoRR*, vol. abs/1511.07122, 2015.

[31] R. Hamaguchi, A. Fujita, K. Nemoto, T. Imaizumi, and S. Hikosaka, "Effective use of dilated convolutions for segmenting small object instances in remote sensing imagery," *CoRR*, vol. abs/1709.00179, 2017.

[32] Y.-L. Boureau, J. Ponce, and Y. LeCun, "A theoretical analysis of feature pooling in visual recognition," in *Proceedings of the 27th international conference on machine learning (ICML-10)*, pp. 111–118, 2010.

[33] D. Scherer, A. Müller, and S. Behnke, "Evaluation of pooling operations in convolutional architectures for object recognition," in *International conference on artificial neural networks*, pp. 92–101, Springer, 2010.

[34] H. Wu and X. Gu, "Max-pooling dropout for regularization of convolutional neural networks," in *International Conference on Neural Information Processing*, pp. 46–54, Springer, 2015.

[35] C.-Y.Lee, P. W. Gallagher, and Z. Tu, "Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree," in *Artificial Intelligence and Statistics*, pp. 464–472, 2016.

[36] Y. Lecun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, pp. 2278–2324, Nov 1998.

[37] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.

[38] C. Szegedy and et al., "Going deeper with convolutions," *CoRR*, vol. abs/1409.4842, 2014.

[39] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *CoRR*, vol. abs/1512.03385, 2015.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, pp. 1735–1780, Nov. 1997.

[41] F. A. Gers and J. Schmidhuber, "Recurrent nets that time and count," in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, vol. 3, pp. 189–194 vol.3, 2000.

[42] K. H. Cho, B. van Merrienboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *CoRR*, vol. abs/1409.1259, 2014.

[43] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A search space odyssey," *CoRR*, vol. abs/1503.04069, 2015.

[44] T. Mikolov, M. Karafiát, L. Burget, J. Černockỳ, and S. Khudanpur, "Recurrent neural network based language model," in *Eleventh Annual Conference of the International Speech Communication Association*, 2010.

[45] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to sequence learning with neural networks," in *Advances in neural information processing systems*, pp. 3104–3112, 2014.

[46] K. M. Hermann, T. Kocisky, E. Grefenstette, L. Espeholt, W. Kay, M. Suleyman, and P. Blunsom, "Teaching machines to read and comprehend," in *Advances in Neural Information Processing Systems*, pp. 1693–1701, 2015.

[47] K. Gregor, I. Danihelka, A. Graves, D. Rezende, and D. Wierstra, "Draw: A recurrent neural network for image generation," *arXiv preprint arXiv:1502.04623*, 2015.

[48] A. Oord, N. Kalchbrenne, and K. Kavukcuoglu, "Pixel recurrent neural networks," *arXiv preprint arXiv:1601.06759*, 2016.

[49] A. Graves, S. Fernández, and e. J. Schmidhuber

[50] J. Chung, S. Ahn, and Y. Bengio, "Hierarchical multiscale recurrent neural networks," *arXiv preprint arXiv:1609.01704*, 2016.

[51] e. a. J. Masci, *Stacked Convolutional Auto-Encoders for Hierarchical Feature Extraction*, pp. 52–59. 2011.

[52] P. Vincent, H. Larochelle, Y. Bengio, and P. Manzagol, "Extracting and composing robust features with denoising autoencoders," in *Proceedings of the 25th International Conference on Machine Learning*, ICML '08, pp. 1096–1103, ACM, 2008.

[53] D. P. Kingma and M. Welling, "Auto-encoding variational bayes," *arXiv preprint arXiv:1312.6114*, 2013.

[54] I. Goodfellow and et al, "Generative adversarial nets," in *Advances in neural information processing systems*, pp. 2672–2680, 2014.

[55] e. a. S. E. Reed, "Generative adversarial text to image synthesis," *CoRR*, vol. abs/1605.05396, 2016.

[56] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[57] M. D. Zeiler, D. Krishnan, G. W. Taylor, and R. Fergus, "Deconvolutional networks," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pp. 2528–2535, IEEE, 2010.

[58] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," in *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, (USA), pp. 807–814, Omnipress, 2010.

[59] K. He, X. Zhang, S. Ren, and J. Sun, "Delving deep into rectifiers: Surpassing human-level performance on imagenet classification," *CoRR*, vol. abs/1502.01852, 2015.

[60] D.-A.

[61] I. Goodfellow, D. Warde-Farley, M. Mirza, A. Courville, and Y. Bengio, "Maxout networks," *arXiv preprint arXiv:1302.4389*, 2013.

[62] C. M. Bishop, *Pattern Recognition and Machine Learning (Information Science and Statistics)*. Secaucus, NJ, USA: Springer-Verlag New York, Inc., 2006.

[63] B. Polyak, "Some methods of speeding up the convergence of iteration methods," vol. 4, pp. 1–17, 12 1964.

[64] I. Sutskever, J. Martens, G. Dahl, and G. E. Hinton, "On the importance of initialization and momentum in deep learning," in *International conference on machine learning*, pp. 1139–1147, 2013.

[65] Y. Nesterov, "A method of solving a convex programming problem with convergence rate o (1/k2)," in *Soviet Mathematics Doklady*, vol. 27, pp. 372–376, 1983.

[66] J. Duchi, E. Hazan, and Y. Singer, "Adaptive subgradient methods for online learning and stochastic optimization," *Journal of Machine Learning Research*, vol. 12, no. Jul, pp. 2121–2159, 2011.

[67] M. D. Zeiler, "ADADELTA: an adaptive learning rate method," *CoRR*, vol. abs/1212.5701, 2012.

[68] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *CoRR*, vol. abs/1412.6980, 2014.

[69] N. Srivastava, G. E. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, pp. 1929–1958, 2014.

[70] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015.

[71] H. Shimodaira, "Improving predictive inference under covariate shift by weighting the log-likelihood function," *Journal of statistical planning and inference*, vol. 90, no. 2, pp. 227–244, 2000.

[72] S. U. CS231n Convolutional Neural Networks for Visual Recognition, "Visualizing what convnets learn."

[73] K. Simonyan, A. Vedaldi, and A. Zisserman, "Deep inside convolutional networks: Visualising image classification models and saliency maps," *CoRR*, vol. abs/1312.6034, 2013.

[74] A. Karpathy, J. Johnson, and L. Fei-Fei, "Visualizing and understanding recurrent networks," *arXiv preprint arXiv:1506.02078*, 2015.

[75] S. J. Pan and Q. Yang, "A survey on transfer learning," *IEEE Transactions on Knowledge and Data Engineering*, vol. 22, pp. 1345–1359, Oct 2010.

[76] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," *CoRR*, vol. abs/1411.1792, 2014.

[77] C. Szegedy, W. Zaremba, I. Sutskever, J. Bruna, D. Erhan, I. Goodfellow, and R. Fergus, "Intriguing properties of neural networks," *arXiv preprint arXiv:1312.6199*, 2013.

[78] A. Nguyen, J. Yosinski, and J. Clune, "Deep neural networks are easily fooled: High confidence predictions for unrecognizable images," in

Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 427–436, 2015.

[79] N. Papernot, P. McDaniel, S. Jha, M. Fredrikson, Z. B. Celik, and A. Swami, "The limitations of deep learning in adversarial settings," in *Security and Privacy (EuroS&P), 2016 IEEE European Symposium on*, pp. 372–387, IEEE, 2016.

[80] N. Papernot, P. D. McDaniel, and I. J. Goodfellow, "Transferability in machine learning: from phenomena to black-box attacks using adversarial samples," *CoRR*, vol. abs/1605.07277, 2016.

[81] N. Papernot, P. D. McDaniel, X. Wu, S. Jha, and A. Swami, "Distillation as a defense to adversarial perturbations against deep neural networks," *CoRR*, vol. abs/1511.04508, 2015.

[82] N. Carlini and D. Wagner, "Towards evaluating the robustness of neural networks," in *Security and Privacy (SP), 2017 IEEE Symposium on*, pp. 39–57, IEEE, 2017.

[83] I. E. et al., "Robust physical-world attacks on machine learning models," *CoRR*, vol. abs/1707.08945, 2017.

[84] J. Smith, S. U. C. for Computer Research in Music, Acoustics, and D. o. M. Stanford University, *Spectral Audio Signal Processing*. W3K, 2011.

[85] F. Colangelo, F. Battisti, M. Carli, A. Neri, and F. Calabró, "Enhancing audio surveillance with hierarchical recurrent neural networks," in *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, Aug 2017.

[86] R. D. Patterson, "Auditory filters and excitation patterns as representations of frequency resolution," *Frequency Selectivity in Hear-ing*, pp. 123–177, 1986.

[87] M. S. et al., "An efficient implementation of the patterson-holdsworth auditory filter bank," *Apple Computer, Perception Group, Tech. Rep*, vol. 35, no. 8, 1993.

[88] J. C. Brown, "Calculation of a constant q spectral transform," *The Journal of the Acoustical Society of America*, vol. 89, no. 1, pp. 425–434, 1991.

[89] N. Scaringella, G. Zoia, and D. Mlynek, "Automatic genre classification of music content: a survey," *IEEE Signal Processing Magazine*, vol. 23, pp. 133–141, March 2006.

[90] Z. Fu, G. Lu, K. M. Ting, and D. Zhang, "A survey of audio-based music classification and annotation," *IEEE Transactions on Multimedia*, vol. 13, pp. 303–319, April 2011.

[91] F. Gouyon, F. Pachet, and O. Delerue, "On the use of zero-crossing rate for an application of classification of percussive sounds," in *Proceedings of the COST G-6 Conference on Digital Audio Effects (DAFX-00*, 2000.

[92] G. Tzanetakis and P. Cook, "Musical genre classification of audio signals," *IEEE Transactions on Speech and Audio Processing*, vol. 10, pp. 293–302, Jul 2002.

[93] E. Schubert, J. Wolfe, and A. Tarnopolsky, "Spectral centroid and timbre in complex, multiple instrumental textures," in *Proceedings of the international conference on music perception and cognition, North Western University, Illinois*, pp. 112–116, sn, 2004.

[94] H. t. Cheng, Y. h. Yang, Y. c. Lin, I. Liao, and H. H. Chen, "Automatic chord recognition for music classification and retrieval," *ICME*, p. 2008.

[95] H.-G. Kim, N. Moreau, and T. Sikora, "Audio classification based on mpeg-7 spectral basis representations," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, pp. 716–725, May 2004.

[96] S. Zhang, Y. Guo, and Q. Zhang, "Robust voice activity detection feature design based on spectral kurtosis," in *2009 First International Workshop on Education Technology and Computer Science*, vol. 3, pp. 269–272, March 2009.

[97] D. Stowell, D. Giannoulis, E. Benetos, M. Lagrange, and M. D. Plumbley, "Detection and classification of acoustic scenes and events," *IEEE Transactions on Multimedia*, vol. 17, pp. 1733–1746, Oct 2015.

[98] J. Dennis, H. D. Tran, and H. Li, "Spectrogram image feature for sound event classification in mismatched conditions," *IEEE Signal Processing Letters*, vol. 18, pp. 130–133, Feb 2011.

[99] A. Rakotomamonjy and G. Gasso, "Histogram of gradients of time-frequency representations for audio scene classification," *IEEE-ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 142–153, Jan 2015.

[100] J. Ren, X. Jiang, J. Yuan, and N. Magnenat-Thalmann, "Sound-event classification using robust texture features for robot hearing," *IEEE Transactions on Multimedia*, vol. 19, pp. 447–458, March 2017.

[101] D. A. et al, "Deep speech 2: End-to-end speech recognition in english and mandarin," *CoRR*, vol. abs/1512.02595, 2015.

[102] D. Povey and et al., "Purely sequence-trained neural networks for asr based on lattice-free mmi.," in *INTERSPEECH*, pp. 2751–2755, 2016.

[103] I. McLoughlin, H. Zhang, Z. Xie, Y. Song, and W. Xiao, "Robust sound event classification using deep neural networks," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, no. 3, pp. 540–552, 2015.

[104] F. Guo, D. Yang, and X. Chen, "Using deep belief network to capture temporal information for audio event classification," in *Int. Conf. on In-*

telligent Information Hiding and Multimedia Signal Processing, pp. 421–424, Sept 2015.

[105] K. J. Piczak, "Environmental sound classification with convolutional neural networks," in *25th Int. Workshop on Machine Learning for Signal Processing*, pp. 1–6, Sept 2015.

[106] M. Espi, M. Fujimoto, K. Kinoshita, and T. Nakatani, "Exploiting spectro-temporal locality in deep learning based acoustic event detection," *EURASIP Journal on Audio, Speech, and Music Processing*, vol. 2015, no. 1, p. 26, 2015.

[107] G. Parascandolo, H. Huttunen, and T. Virtanen, "Recurrent neural networks for polyphonic sound event detection in real life recordings," in *Int. Conf. on Acoustics, Speech and Signal Processing*, pp. 6440–6444, IEEE, 2016.

[108] E. Marchi, F. Vesperini, F. Eyben, S. Squartini, and B. Schuller, "A novel approach for automatic acoustic novelty detection using a denoising autoencoder with bidirectional lstm neural networks," in *Int. Conf. on Acoustics Speech and Signal Processing*, pp. 1996–2000, April 2015.

[109] J. Salamon and J. P. Bello, "Deep convolutional neural networks and data augmentation for environmental sound classification," *IEEE Signal Processing Letters*, vol. 24, pp. 279–283, March 2017.

[110] J. Salamon and C. J. J. Bello, "A dataset and taxonomy for urban sound research," in *Proceedings of the 22nd ACM international conference on Multimedia*, pp. 1041–1044, ACM, 2014.

[111] N. Jaitly and G. E. Hinton, "Vocal tract length perturbation (vtlp) improves speech recognition,"

[112] X. Cui, V. Goel, and B. Kingsbury, "Data augmentation for deep neural network acoustic modeling," *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, vol. 23, pp. 1469–1477, Sept 2015.

[113] W. Huang, S. Lau, T. Tan, L. Li, and L. Wyse, "Audio events classification using hierarchical structure," in *Procs.of the 2003 Joint Fourth Int. Conf. on Information, Communications and Signal Processing, and the Fourth Pacific Rim Conference on Multimedia*, vol. 3, pp. 1299–1303 vol.3, Dec 2003.

[114] P. K. Atrey, N. C. Maddage, and M. S. Kankanhalli, "Audio based event detection for multimedia surveillance," in *Int. Conf. on Acoustics Speech and Signal Processing*, vol. 5, pp. V–V, May 2006.

[115] G. Valenzise, L. Gerosa, M. Tagliasacchi, F. Antonacci, and A. Sarti, "Scream and gunshot detection and localization for audio-surveillance systems," in *Int. Conf. on Advanced Video and Signal-Based Surveillance*, pp. 21–26, Sept 2007.

[116] W. Choi, J. Rho, D. Han, and H. Ko, "Selective background adaptation based abnormal acoustic event recognition for audio surveillance," in *Int. Conf. on Advanced Video and Signal-Based Surveillance*, pp. 118–123, IEEE, 2012.

[117] P. Foggia, N. Petkov, A. Saggese, and N. S. Vento, "Reliable detection of audio events in highly noisy environments," *Pattern Recognition Letters*, vol. 65, pp. 22–28, 2015.

[118] P. Foggia, N. Petkov, A. Saggese, N. Strisciuglio, and M. Vento, "Audio surveillance of roads: A system for detecting anomalous sounds," *IEEE Transactions on Intelligent Transportation Systems*, vol. 17, pp. 279–288, Jan 2016.

[119] M. Z. Zaheer, J. Y. Kim, H.-G. Kim, and S. Y. Na, "A preliminary study on deep-learning based screaming sound detection," in *5th Int. Conf. on IT Convergence and Security*, pp. 1–4, IEEE, 2015.

[120] R. Gonzalez and R. Woods, *Digital Image Processing,2/e.* Pearson Education, 2008.

[121] F. Colangelo, F. Battisti, M. Carli, and A. Neri, "A multi-objective optimization for video orchestration," in *2017 25th European Signal Processing Conference (EUSIPCO)*, Aug 2017.

[122] G. Wyszecki and W. Stiles, *Color Science: Concepts and Methods, Quantitative Data and Formulae*. Wiley Series in Pure and Applied Optics, Wiley, 2000.

[123] A. K. M. P., Obrador, and N. Oliver, "Towards computational models of the visual aesthetic appeal of consumer videos," in *Proc. of Computer Vision, ECCV 2010*.

[124] R. Datta, D. Joshi, J. Li, and J. Z. Wang, "Studying aesthetics in photographic images using a computational approach," in *Computer Vision ECCV 2006: 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006*.

[125] Z. Dai and Y. Wu, "Where are focused places of a photo?," in *Advances in Visual Information Systems: 9th International Conference, VISUAL 2007 Shanghai, China, June 28-29*.

[126] P. Obrador, "Region based image appeal metric for consumer photos," in *Proc. of Multimedia Signal Processing, 2008 IEEE 10th Workshop on*.

[127] Bhattacharya and et al., "Towards a comprehensive computational model foraesthetic assessment of videos," in *Proc. of the* 21^{st} *ACM International Conference on Multimedia*, 2013.

[128] E. Siahaan, A. Hanjalic, and J. Redi, "A reliable methodology to collect ground truth data of image aesthetic appeal," *IEEE Transactions on Multimedia*, vol. 18, pp. 1338–1350, July 2016.

[129] N. Murray, L. Marchesotti, and F. Perronnin, "Ava: A large-scale database for aesthetic visual analysis," in *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pp. 2408–2415, IEEE, 2012.

[130] W. Luo, X. Wang, and X. Tang, "Content-based photo quality assessment," in *Computer Vision (ICCV), 2011 IEEE Int. Conf. on*.

[131] S. brain, "Youtube company statistics."

[132] R. Kaiser, P. Torres, and M. Höffernig, "The interaction ontology: Low-level cue processing in real-time group conversations," in *2nd ACM International Workshop on Events in Multimedia*, EiMM '10, ACM.

[133] E. S. d. Lima, B. Feij, A. L. Furtado, A. Ciarlini, and C. Pozzer, "Automatic video editing for video-based interactive storytelling," in *2012 IEEE International Conference on Multimedia and Expo*, pp. 806–811, July 2012.

[134] K. Dancyger, *The Technique of Film and Video Editing History, Theory, and Practice*.

[135] W. Murch, *In the Blink of an Eye: A Perspective on Film Editing*.

[136] A. Neri, F. Battisti, F. Colangelo, and M. Carli, "Unsupervised video orchestration based on aesthetic features," in *2017 IEEE International Symposium on Circuits and Systems (ISCAS)*, pp. 1–4, May 2017.

[137] K.-Y. Lo, K.-H. Liu, and C.-S. Chen, "Assessment of photo aesthetics with efficiency," in *Pattern Recognition (ICPR), 2012 21st Int. Conf. on.*

[138] X. Tang, W. Luo, and X. Wang, "Content-based photo quality assessment," *IEEE Transactions on Multimedia*, vol. 15, no. 8, pp. 1930–1943, 2013.

[139] D. Kalyanmoy, *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, Inc., 2001.

[140] ITU, "ITU-R Recommendation, Subjective assessment of video quality using expert viewing protocol." BT.2095, Apr. 2016.