Ca' Foscari
University
of Venice

Università Ca' Foscari di Venezia
Master's degree programme – Second Cycle
(D.M. 270/2004)

in Informatica – Computer Science

Final Thesis

Combining Deep Learning and Game Theory for Music Genre Classification

Supervisor
Ch. Prof. Marcello Pelillo

Assistant supervisor
A. Torcinovich

Graduand
Paola Urbani
Matriculation Number 825330

Academic Year
2016 / 2017

Contents

Abstract

In this thesis we have studied how a technique based on how game theory can improve classification results obtained with a deep learning module. In order to get this improvement we have applied this method to the music genre classification problem, comparing the obtained results. The proposed model is composed by a convolutional recurrent neural network (CRNN), that deals with classifying every single element, and the Graph Transduction Game (GTG) method, that allows to compare these elements on the basis of a similarity measure and thus exploit the contextual information in order to get a better classification. The idea behind this work is that the neural network architecture does not directly exploit the information coming from the comparison of the observations passed as input. Therefore we think that the introduction of a module in charge for this purpose can improve final accuracy, especially when we work with limited datasets. In order to assess the effect of the proposed approach we have performed experiments on benchmark datasets and we report the obtained results.

1 Introduction

Artificial intelligence is a branch of data science that attempts to reproduce the mechanisms that human brain uses to grasp and interpret surrounding reality. This is for example the case of the recognition of objects in an image, which can be done using algorithms designed to reproduce the way in which central nervous system captures information, processes it and classifies the recognized objects.

Among the elements of reality we can take into account sound and music, which have given rise to a wide range of applications named as Sound and Music Computing. One of the topics addressed in this branch of computer science is Music Information Retrieval: this is a multidisciplinary field that includes musicology, signal processing and artificial intelligence and deals with gathering and studying sound information. One of the studied topics in Music Information Retrieval is that of music genre classification, which consists in inferring the musical genre to which a piece belongs according to its features. In order to retrieve this information artificial intelligence provides us with methods, such as those belonging to the so-called data-driven approach, a method that infers the structural criteria of a piece thanks to which it is possible to produce a classification. The most interesting among the techniques of this branch are those that are able to "learn" from a set of observations and adapt their behavior to others on the basis of what they have learned. However, it is also interesting to see how the use of a combination of different approaches can contribute in producing better results compared to the ones produced by each method on its own.

In this thesis we combined a neural network with a game theory based algorithm and we tested the results of this combination with those obtained from the neural network alone. Specifically, we studied how the Graph Transduction Game (GTG) proposed by Pelillo et al. [1] can improve the results obtained by the Convolutional Recurrent Neural Network (CRNN) proposed by Choi et al. [2] in the field of music genre classification for single labeling. The purpose of these experiments is to exploit the comparison information between observations belonging to the same category to improve final accuracy even when we are using datasets having few known observations.

This thesis is expounded on the treatment of sound, music genres and some methods

of machine learning applied for music genre classification and it analyzes the proposed model and the performed experiments with related results.

In chapter 1 we give a brief description of the theory behind sound, its synthesis and a general discussion about music genres.

In chapter 2 we define what the music information retrieval is, giving an overview of the addressed issues. Furthermore, a brief summary of some of the techniques used for music genre classification is provided.

In chapter 3 we give an in-depth explanation of the model we have used, explaining the theory of each of its components. Particular attention is given to the Graph Transduction Game algorithm, on which the performed experiments are based.

In chapter 4 we explain our experiments and discuss the obtained results.

Finally, in chapter 5 we present the concluding remarks and the future works that can be inspired by this thesis.

2 From sound to music genres

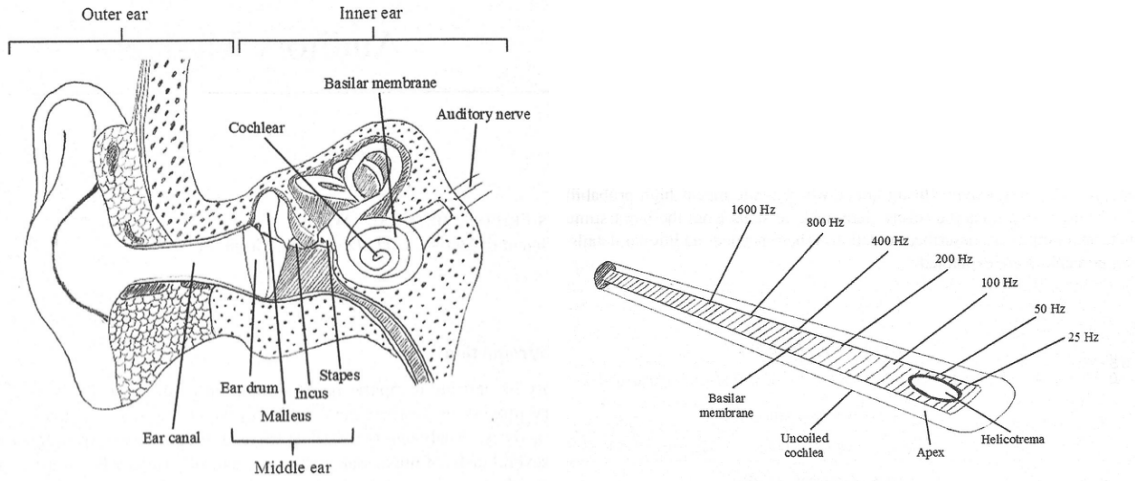
In this chapter we give a brief explanation of the theoretical knowledge behind the sound processing. Starting from the nature of sound, we show how a sound is brought from analog to digital. Finally we give a brief description of what is meant with music and what are music genres. The content of this chapter is basically taken from the sources[8, 15].

2.1 Nature of sound

From physics' point of view, sound is an undulatory phenomenon. At the origin of a sound there is an object that generates it, called sound source; this object is put into vibration by an internal or external cause. Typically, in musical instruments the vibrating body is a string, an air column, a membrane or a reed. These elements are put into vibration by applying some energy by the performer.

The vibrating object transmits its energy causing a succession of rarefactions and compressions to the medium in which it is immersed (e.g. air or water). This vibration propagates in all directions starting from the source by vibrating every objects it finds along its path. As a result of the existing frictions and resistances, the phenomenon decreases as it moves away from the source and the objects put in vibration gradually dampen their movement.[8]

The organ responsible for receiving sounds is the ear. The ear is composed of the external pinna, which serves to convey and amplify the sound energy transmitted by the air into a channel, called the ear canal, which ends with a membrane called the ear drum. The oscillations of this membrane through a mechanical transmission formed by three ossicles (malleus, incus and stapes) are transferred to an organ called cochlea, a scroll-shaped organ containing the basilar membrane, composed of hair cells. These cells are arranged in the base membrane in order to respond to different frequency bands according to their position. Their oscillation is transmitted via the acoustic nerve to the area of the brain responsible for the recognition and processing of sounds.[8]



(a) Model of human ear

(b) Basilar membrane

Figure 2.1.1: Model of human ear and basilar membrane with frequency [8]

Since it is a periodic signal, sound is basically characterized by three characteristics: amplitude, frequency and waveform.

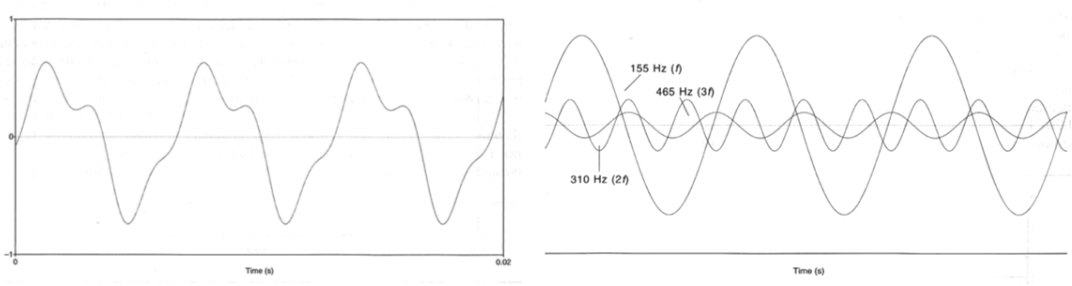
Amplitude. This parameter represents the energy carried by the signal.

Frequency. The frequency of a sound or pitch is the number of complete vibrations that the source performs in a second and is responsible for the feeling of lowness / highness that is perceived in a sound.

Waveform. The waveform identifies the source from which the sound originates.

> "The waveform is what characterizes the way in which a specific source vibrates, thus allowing to identify the origin of a sound. [...] The parameter that best corresponds to the waveform is the timbre, i.e. that element that makes it possible to distinguish" [15] whether a note is emitted from an instrument or from a different instrument.

We distinguish pure sounds represented by a simple sinusoid and complex sounds, whose form is given by the sum of several pure sounds (see Fourier Analysis).



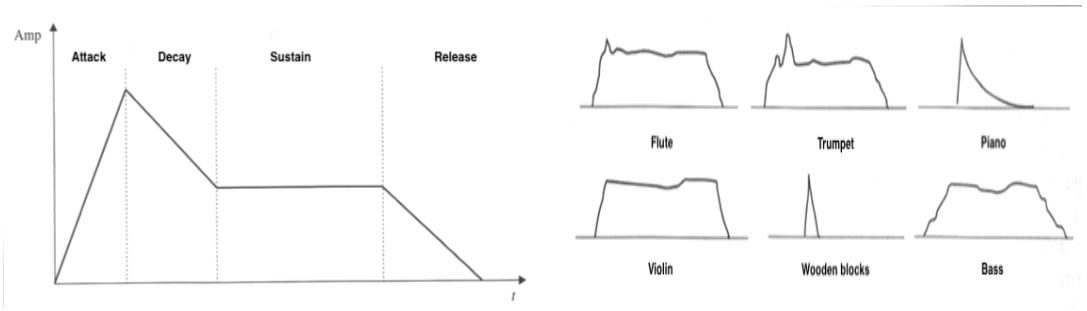
(a) (b)

Figure 2.1.2: Complex sound a) and its harmonics b) [15]

Taking into account how the sound is generated, the complex sound that characterizes the instrument is superimposed with a whole series of undesired sounds, called noises, due to spurious vibrations, refractions, reverberations, echoes and other caused by the performer's interaction with the instrument (see breath in wind instruments, frictions on the strings, etc). Real sounds, which are always complex sounds, are periodic waves limited in time. "Each source must pass from a rest condition to a periodic vibration one" to return "to a condition in which the sound is extinguished. Only for an intermediate phase we can speak about [...] periodicity. The way in which sound evolves over time with respect to its amplitude is called envelope ". The envelope corresponds to the curve obtained by joining the peaks of the positive part of the signal. In this way the waveform is modeled on the envelope. Four phases can be identified and they are distinguished in:

- Attack: is the time interval in which the waveform reaches its maximum amplitude, i.e. when the source passes from the rest condition to the periodic vibration one.
- Decay: in this phase the amplitude decreases to a certain level
- Sustain: phase in which the amplitude remains almost constant
- Release: phase in which the amplitude is reduced to zero, which corresponds to the return to the rest condition of the source

Therefore the envelope together with the timbre characterize the kind of the source.



(a) Envelop phases (b) Envelop examples

Figure 2.1.3: Envelop phases and examples [15]

2.2 From analog to digital

2.2.1 Signal analysis

In 1822 J. B. J. Fourier published a famous theorem which is a very powerful instrument for frequency analysis of vibrations and sounds.

Theorem 1. *(Fourier) Any periodic signal is given by the superposition of simple sinusoidal waves, each with its frequency and phase and whose frequencies are multiple (harmonics) of the fundamental frequency of the signal.*

Given a periodic signal f, such that $f(x+T) = f(x)$, this theorem says that f can be expressed according to the following series of functions, called Fourier series:

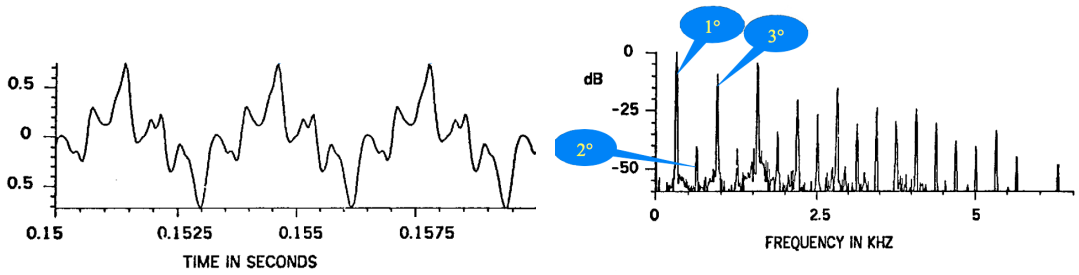
$$f(x) = \frac{a_0}{2} + \sum_{n=1}^{\infty} \left(a_n \cos(nx) + b_n \sin(nx)\right)$$

where:

- $\frac{a_0}{2} = \frac{1}{2\pi}\int_0^{2\pi} f(x)\,dx$
- $a_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\cos(nx)\,dx$
- $b_n = \frac{1}{\pi}\int_0^{2\pi} f(x)\sin(nx)\,dx$

The term of the series for which $n = 1$ is called fundamental harmonic of the signal and determines the perceived pitch. The Fourier series shifts the analysis of the signal from time domain to that of the frequency, so the set of signal harmonics constitutes what is called the amplitude spectrum of the signal. Above a certain level of harmonics

$n > k, k \in \mathbb{N}$ the amplitude, i.e. the coefficients a_n and b_n, is so small that it is possible to ignore the contribution that these harmonics bring to the signal. The sum of the Fourier series can therefore be considered as a summation limited above the *k-th* term and to consider this frequency as the maximum one required to define the signal.



(a) Clarinet E flat 311Hz signal

(b) Clarinet E flat 311Hz spectrum indicating the first three harmonics

Figure 2.2.1: Signal and spectrum of a clarinet playing E flat at 311Hz

2.2.2 Analog to Digital Conversion

The idea of conversion from an analog signal to its digital representation passes through two fundamental concepts: sampling and sample quantization. Sampling is the operation in which a certain device picks up a signal value at predetermined time intervals. These samples are then *quantized* by an operation to divide the maximum amplitude into levels. Quantization consists of a word formed by m bits, which represents the (voltage) level of the signal at that moment by means of a translation function. A digital file representing an audio signal is therefore represented by a collection of this kind of words. The Nyquist-Shannon theorem guarantees that the original signal can be reconstructed, provided that the sampling rate is at least twice the frequency of the maximum harmonic of the signal to be converted. The standards currently used range from minimum sample rates of 44100 Hz up to 96 kHz, while the word size is of 16 bits, which allows the distinction of 65,536 different levels, or of 24 bits, which allows the definition of 16,777,216 levels. Currently, various compression methods have made it possible to reduce file sizes with minimal information loss. Once the digital file is obtained, we can use it for purposes ranging from entertainment to information retrieval.

> "A musical signal carries a substantial amount of information that corresponds to its timbre, melody or rhythm properties and that may be used to classify music, e.g. in terms of instrumentation, chord progression or musical

genre"[8]

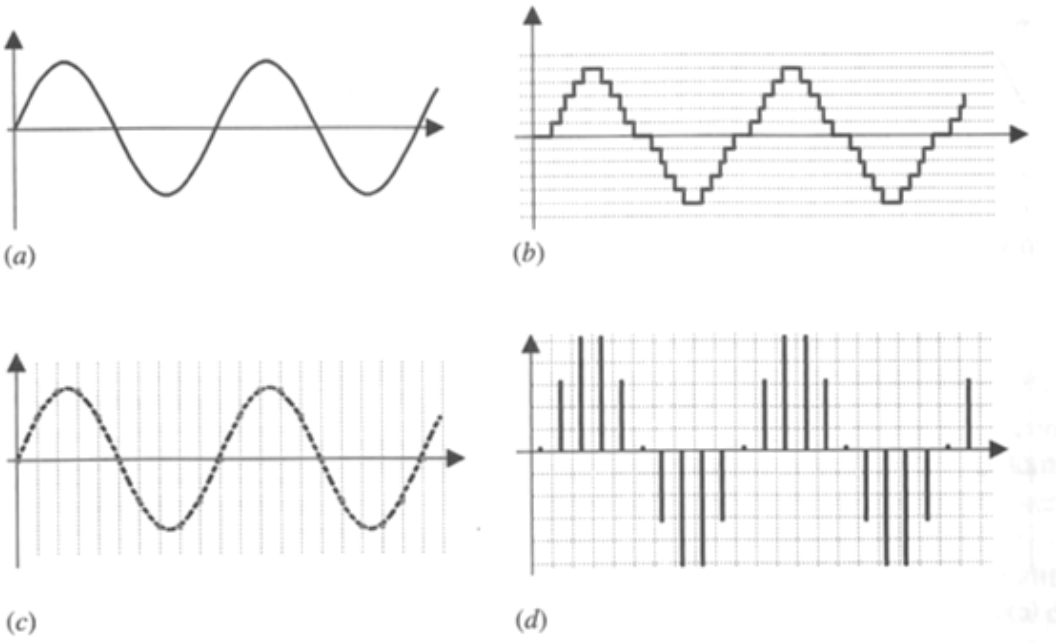


Figure 2.2.2: Analogic signal (a), quantized (b), sampled (c), quantized and sampled (d) [15]

2.3 Sound, Music and Musical Genres

A succession of sounds ruled by harmonic laws is defined as music. Music has always been subdivided into genres based on the purpose it was made for.

> "A music genre is a conventional category that identifies some pieces of music as belonging to a shared tradition or set of conventions" [16].

Since ancient times there has always been popular music and music for liturgical and secular purposes. Music can be divided into music genres based on features that determine it, — such as instrumental ones — the era in which it was composed, composition patterns, etc. Subdivisions based only on the use of particular musical instruments is a weak classification, because their use is frequent in other genres, even though some instruments are essentially associated with a particular genre for which they were created. It is more likely that a piece in which a violin is playing may belong to classical music genre than to others, in the same way of an electric guitar to the pop or rock genres. For example, we

can find a sitar playing in the song "Strawberry Fields Forever" by the Beatles or part of the symphonic orchestra in the song "Who Wants to Live Forever" by Queen, that are basically rock and pop groups.

A method to classify music genres in a more consistent way is to analyze pieces based on the so-called *musical form*. By the term musical form we mean the pattern of sentences that makes up the structure of a piece. A bit like in Italian poetry with the interweaving of rhymes, the succession of musical phrases, that characterize the parts in which a piece can be subdivided, identify a piece as belonging to a musical genre often with good precision. This kind of classification may require a complete analysis of the piece, which from the point of view of an automatic classification may not be possible due to space and time matters. The classification method that is carried out at the state-of-the-art in data science consists of analyzing 30 seconds clips of songs extracting their features.

3 Music Information Retrieval and Machine Learning

In this chapter we talk about Music Information Retrieval and how machine learning has been applied in this field, with particular attention to music genre classification. References to works done for each method addressed are also provided.

3.1 Music information retrieval: a brief overview

Music Information Retrieval is a multidisciplinary field that includes musicology, signal processing and artificial intelligence that deals with gathering and studying sound information. The information collected can be used in a wide range of areas:

- Delicate restoration operations of audio media (tapes, disks, etc.) related to sound archives of modern composers, such as Luigi Nono and Luciano Berio, with digitization of the content and its cataloging with annexed documentation about the interventions carried out. The plays thus digitized are made available for consultation by researchers and musicologists, without having to use the original supports, whose conditions would not allow to make use of such artistic works.
- Use of synthesized sound feedback in biomedical field to facilitate the rehabilitation of human motility: the addition of these sound feedback has allowed improvements in recovery time of the patients.
- Simulation of musical instruments: an application of simulation techniques was employed to digitally reconstruct the sound of a 2000-years-old pan flute through the study of its mechanical and physical components.
- Use of information from the analysis of plays by various authors for automatic generation of songs that imitate their style.
- Use of models for the automatic execution of songs, simulating the expressive performance of a human interpreter. This is done using methods such as analysis by

measures, which is based on the a priori definition of rules based on expert knowledge, or data driven approach, in which the definition of rules is inferred from provided data.

- Realization of applications for recognition of musical pieces comparing them with those contained in a database (e.g. Shazam).

Figure 3.1.1: Photo of the 2000 years old pan flute at "Museo di Scienze Archeologiche e d'Arte" of University of Padua

3.2 Machine learning used in music genre classification

Many methods of machine learning have been used to address music genre classification problem, using different approaches and representations of data. We present below some of the most important methods of supervised learning, accompanying all with a quote from some of the research projects on music genre classification that have made use of them.

3.2.1 Bayesian Methods

In Statistical Learning Theory (SLT), given a dataset to classify X and a set of classes Y, we define classifier a function $f : X \times Y$ which assigns a label $y_j \in Y$ to each object, which is represented by a feature vector X_i.

Assume that there is a joint probability distribution P over $X \times Y$ that is unknown, fixed and on which no assumptions is made. Given the function called *loss function* that evaluates how much a classifier is reliable

$$\ell(X, Y, f(X)) = \begin{cases} 1 & if\ f(X) \neq Y \\ 0 & otherwise \end{cases}$$

we define the *risk* of a function as the weighted average of the loss function for the distribution of probability on the objects, i.e. the expected value of the loss function:

$$R(f) = E(\ell(X, Y, f(X))) = \int \ell(X, Y, f(X))\, dp(x, y)$$

The best classifier is the one with the lowest risk and is identified with the *Bayes classifier* defined as follows:

$$f_{Bayes}(X) = \begin{cases} 1 & if\ P(Y = 1|X = x) \geqslant 0.5 \\ -1 & otherwise \end{cases}$$

based on the *Bayes rule*:

$$P(h|e) = \frac{P(e|h)\, P(h)}{P(e)} = \frac{P(e|h)\, P(h)}{P(e|h)\, P(h) + P(e|\neg h)\, P(\neg h)}$$

The bayesian methods offer a classifier whose risk is as close as possible to that of the Bayes classifier, since it is impossible to calculate such a classifier without knowing the probability distribution P.

Some of the research works done with this method for music genre classification can be found in the following papers: [18]

3.2.2 Nearest Neighbor

Nearest Neighbor is a learning method based on measuring the distance calculated on the features of the elements to be classified. Given a set of objects $X = \{X_1, \ldots, X_n\}$, with $X_i = \{x_1, \ldots, x_m\}$ feature vector and a measure of the distance d defined between the features of the objects (e.g. Euclidean distance), the method assigns to a point X_i the class belonging to the point X_j, whose index j solves the following equation:

$$j = \arg\min_i \{d(X_i, X_j)\}$$

Therefore X_j belongs to the class of the nearest point, i.e. with the minimum distance, excluding the point itself.

Among the variants of this method the most famous include $1 - NN$, which considers the nearest neighbor, $k-NN$, which considers a number k of nearest neighbors, and finally k_nNN, which considers a number of neighbors proportional to the number of classes in which the objects are to be classified. In the case of closer neighbors to consider, the class to which the point belongs is given by the class having the largest number of members among the k neighboring.

Some of the research works done with this method for music genre classification can be found in these papers: [18, 20, 26, 29, 32]

3.2.3 Decision Tree

A decision tree is a tree graph in which leaves represent the classes and internal nodes represent the features, whose values characterize the belonging to those classes. Given a data point to be classified, the traversal of the tree corresponds to the sequence of decisions based on the values assigned to the features of the object. Complex decisions can be expressed in terms of a series of questions. These decisions can be arranged as a tree of horn clauses in first order logic (list of "if then"/"else"); querying the tree we achieve an answer concerning the asked questions. The more attributes we choose to build the tree, the more it fits the model. When we use too much attributes, we get into overfitting problem, i.e. the tree matches too perfectly the given data. This makes the tree weak with respect to cases different from the given examples. Hence it is necessary to stop the tree once we've reached a sufficient number of attributes to get a more acceptable response.

Some of the research works done with this method for music genre classification can be found in these papers: [18, 29]

3.2.4 Support Vector Machine (SVM)

Support Vector Machines are a statistical method devised by Vladimir Vapnik for research fields such as classification and regression. This classifier was developed based on Statistical Learning Theory principle of linearly separating objects in clusters. In order to do this, in SVM a data point is seen as a p-dimensional vector, which elements represent the features of the data point. A set of such data points can be separated by an infinite number of $(p-1)$-dimensional hyperplanes. In SVM it is chosen the hyperplane that maximizes the distances between the nearest data points of the classes to be separated or clustered. In this way the method optimizes the parameters indicated by the SLT. [17]

Some of the research works done with this method for music genre classification can be found in these papers: [18, 20, 26, 29, 32]

3.2.5 Neural Networks

A machine learning method that is having an ever-increasing following by scientific community is neural networks. A neural network is a supervised learning structure that, by imitating the functioning of a human neural network, is able to recognize and classify objects. Its layered structure of artificial neurons makes it possible to learn the recognition of object classes by modifying the parameters that regulate the communication between one layer to another; moreover a trained net is reusable to recognize objects not included among the examples with which it has been trained.

In the research work carried out by Irvin et al. [12] three recurrent neural network structures are used, the simplest Vanilla RNN, the Vanilla LSTM and its slightly modified version. This type of structure analyzes the data passed in by capturing the temporal relations existing between a sequence of data at a more or less depth, starting from the RNN that looks at relations from near distance up to the LSTM, which considers longer time dependencies.

In the work of Keunwoo Choi et al. in [2] and previously in [14] Convolutional Neural Network are proposed for the study of music genre classification from another point of view. These structures process data, maintaining spatial relationships that exist between the extracted features.

Other reasearch works done with this method for music genre classification can be found in these papers: [18, 20, 29]

4 The model

In this chapter we discuss about the theory behind the elements that make up of the model we have used and its implementation. The model consists of a neural network for the learning part and an algorithm based on game and graph theory to refine the network results. Let's start with the description of how the data to classify are obtained starting from a audio file. Then we analyze the structure and functioning of the neural network, a Convolutional Recurrent Neural Network, starting from the basic concepts. Afterwards we explain the Graph Transduction Game algorithm. Finally we conclude the chapter with an overview of the model used and its operation for data classification.

4.1 Model Input

We now describe the kind of data to give as input to our model starting from an audio file. The musical signal is composed by a series of pseudo-periodic signals that come in succession over time. A periodic signal is totally defined by the set of its harmonics (see Fourier Transform). In order to define each of these signals, it is necessary to isolate a quite small portion of time, called sampling window, in which the signal can be considered to be "constant"; from this window it is possible to obtain the set of frequencies using the Fourier transform. The temporal succession of these spectra constitutes a spectrogram, i.e. the representation in time / frequency of the harmonics of the original signal as it evolves over time. In order to perform the spectrogram of an audio signal we can use the most up-to-date method, called Gaber transform.

> "The Gabor transform or windowed Fourier transform uses symmetric window functions $g(t) = g(-t)$ with norm $\int_{-\infty}^{\infty} |g(t)|^2 dt = 1$ to pick only limited portions from the musical signal and determine their frequency contents" [8].

Note that all commonly used g functions, such as the Hamming and Gaussian windows, have a bell shape for $t \in \left[-\frac{1}{2}, \frac{1}{2}\right]$, while outside this range the function is forced to zero.

The spectrogram is computed through the Gabor transform given by the following function:

$$WX(u,\xi) = \int_{-\infty}^{\infty} x(t)\, g^*_{u,\xi}(t)\, dt \tag{4.1.1}$$

where $g(t)$ is the window function.

> "The Gabor transform characterizes the strength and phase with which the frequency ξ is contained in the signal at about time u. The energy density, i.e. the square of its absolute value, is called *spectrogram" [8]*

$$P_{WX}(u,\xi) = |WX(u,\xi)|^2 \tag{4.1.2}$$

In the analysis of musical signals, in which human perception has a fundamental part, scales related to physical frequency are not the only ones used. We also use scales that take into account the perceptual effects, which apply a deformation to the physical scale. One of the most used scales is the so-called Mel-scale, whose curve, that depends on the frequency, is given in the figure (4.1.1).

Definition 2. (Ratio-Pitch and Mel Scale). The psychoacoustic quantity ratio-pitch f_{mel} with its unity mel is defined for a pure tone with a frequency f by

$$f_{mel}(f) = \log_{10}\left(1 + \frac{f}{700Hz}\right) \cdot 2595mel \tag{4.1.3}$$

The Mel-scale finds justification in correspondence with a physiological fact:

> "There is a linear relationship between the Mel-scale and the number of abutting haircells of the basilar membrane: 10 mel correspond to 15 abutting haircells, and the total of 2400 mel corresponds to the total of 3600 haircells" [8].

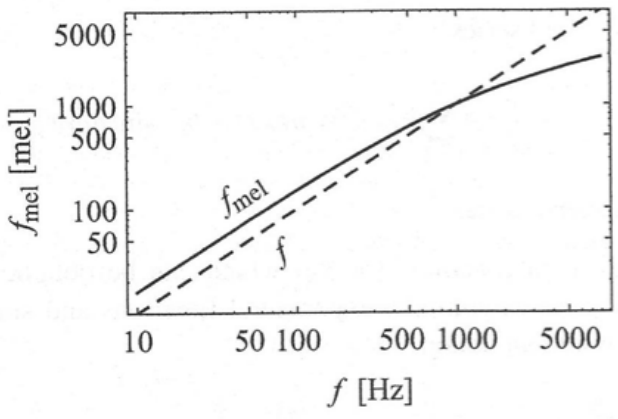


Figure 4.1.1: Relation between Mel-frequency and frequency [8]

Mel-spectrograms are produced by transforming the spectrogram frequencies into Mel-scale ones. For each audio file to be processed we obtain a mel spectrogram in the form of an image that constitutes the input element for the neural network. In this way our classification problem of musical signals is reduced to the classification of images defined on the time / frequency magnitudes and intensity represented by the color of the pixels, as it can be seen in the figure (4.1.2.).

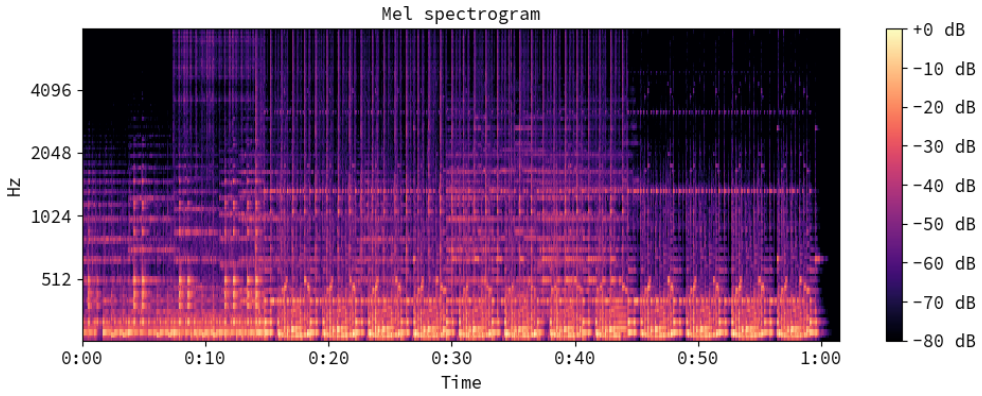


Figure 4.1.2: Example of a Mel-spectrogram

4.2 Deep Learning module

The model we have implemented consists of a neural network, one of the structures on which deep learning is based. The main feature of this structure is to classify a set of objects by implementing a learning process based on the features that characterize them. Such a learning allows the network to perform a classification even for data that has never been seen before; the only condition is that these data are consistent with the ones on which the learning phase has been made with.

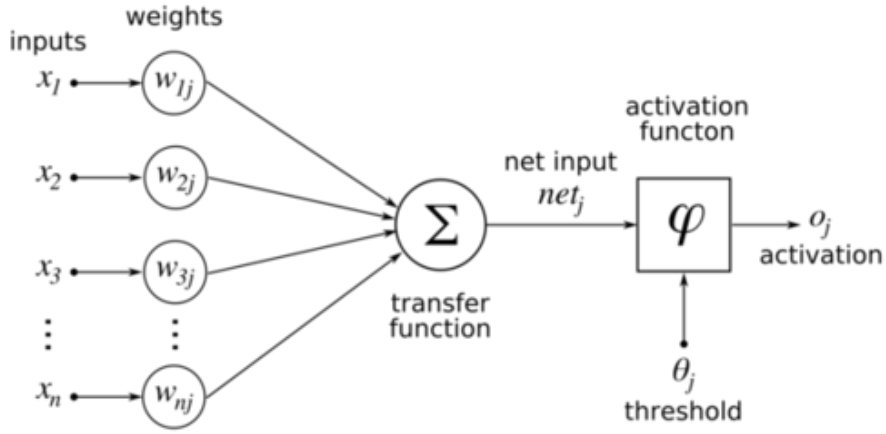


Figure 4.2.1: Artificial neuron, also known as perceptron, by Rosenblatt from the proposed model by Mc Cullough and Pitts

Neural networks principally originate when McCullough and Pitts proposed the model of an artificial neuron in 1943. Subsequently taken up by Rosenblatt with the name of perceptron in 1962, the model proposes that a series of input neurons propagate their information along the weighted arcs towards an output neuron. Each input data weighted or filtered by the weights associated to their edges enter the output neuron and they're summed up by the transfer function. The output value of this neuron is determined by the activation function, which returns a number between 0 and 1 depending on whether the result is greater than a given threshold, also known as bias, or not. In the discrete case this can be read as the turning on or off of the neuron, respectively returning 1 or 0. A single neuron is able to produce a classification between two classes 0 and 1. The presence of multiple neurons forms a layer that allows us to manage a larger amount of input information and to produce a classification to a higher number of classes.

A single-layer neural network can solve linear complexity problems, such as functions AND or OR; however it is not able to solve more complex functions, such as XOR. In order to solve the XOR problem it is necessary to add a hidden layer, which increases the complexity of the computations. The more the number of hidden layers increases the higher the complexity of the problems that can be solved by neural networks, but at the expense of greater space and time computational complexity.

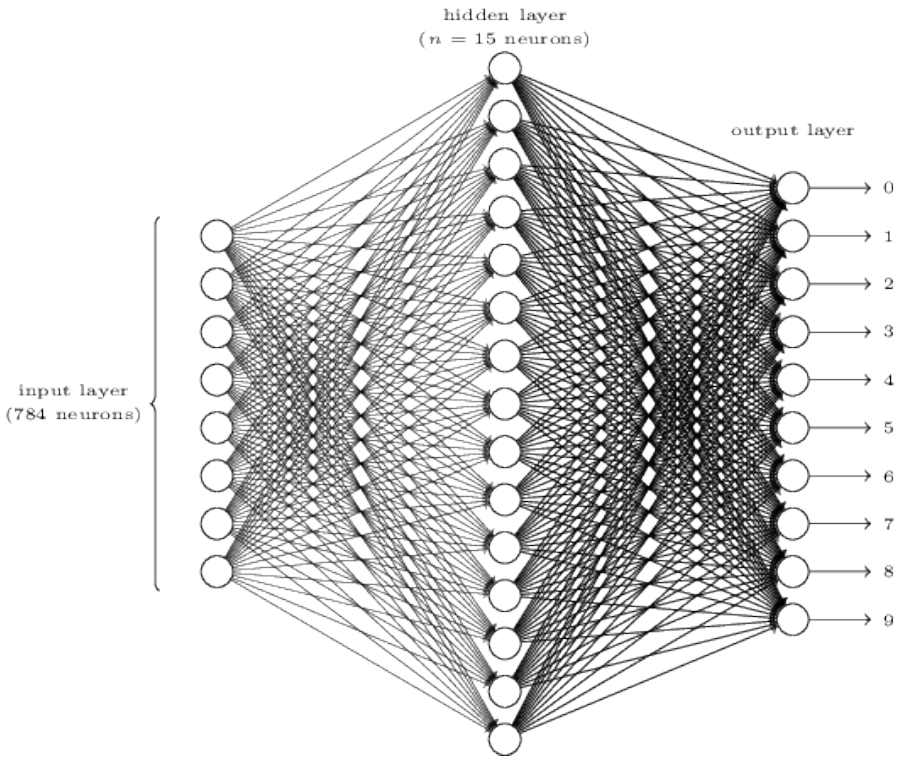


Figure 4.2.2: Example of a Neural Network with a single hidden layer [4]

The classification maden by a neural network is said to be supervised: given a set of input objects, a set of classes and the labels associated to each object indicating their class membership, the network processes the informations of the objects and it classifies them. The learning phase of a neural network starts from the estimation of the error raised by the network during its classification phase with respect to the expected labels. The error is evaluated using the so called Loss Function. Through the Back-Propagation method the error is propagated backwards using the Gradient Descent technique, thus updating the network weights from the output layer back to the input layer. Repeating this procedure of classification, error estimation and weights update through back-propagation several times, the error tends to decrease, thus increasing the accuracy of the results.

Since percettrone was conceived, after a period of decline due to the difficulty of using such structures, the model has been taken up, developed and diversified into different types of structures. This renewal of interest in neural networks has been possible thanks to the technological development of computers, which have allowed us to be able to process increasingly large amounts of data in a reasonable time.

In order to perform the classification task, we used a network architecture that was

suitable for the complexity of the problem to be faced up. Spectrograms are diagrams that show the frequencies of the notes per sampling instants and the succession of such frequencies, similarly to music notes, allows us to define the scheme according to which the music has been composed. In order to do this, we had to use a network that not only extracted the required features, but also their spatial or temporal relationships, musically speaking. The network used in our model is the Convolutional Recurrent Neural Network (CRNN) proposed by Keunwoo Choi et al. [2] for the recognition of music genres. The network consists of a convolutional neural network (CNN), which extracts features from spectrograms, and a recurrent neural network (RNN), which is responsible for producing the classification pattern.

4.2.1 Convolutional Neural Network

A convolutional neural network (CNN) is a deep learning method that allows us to learn objects according to their features. Referring to image processing its particularity consists in the fact that the features are not only extracted at multiple depth levels, but also maintain the spatial relationships existing in the original image.

ARCHITECTURE Convolutional networks in general consist of different levels of feature extraction. In particular the one proposed by Choi et al. [2] consists of four convolutional blocks. As it can be seen in figure (4.2.3) each block is composed of the following layers:

- Convolutional layer
- Activation function: ELU
- Pooling: maxpooling
- Regularization: Dropout

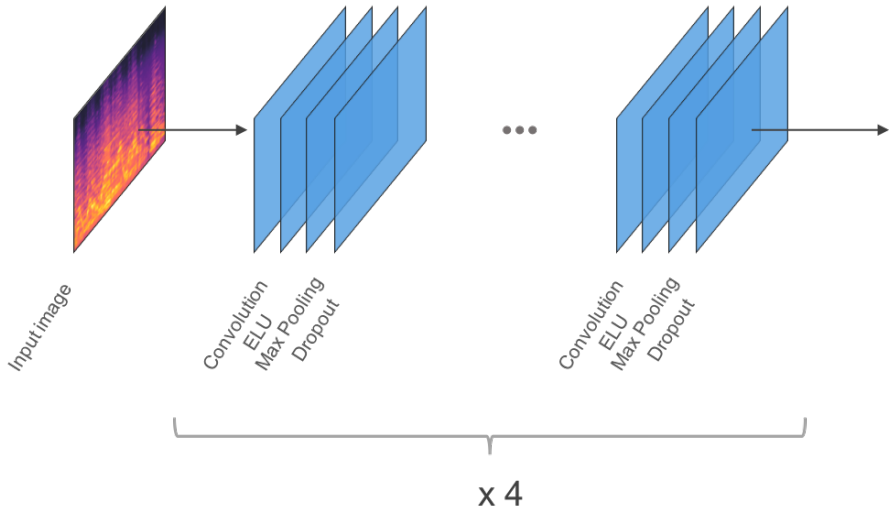
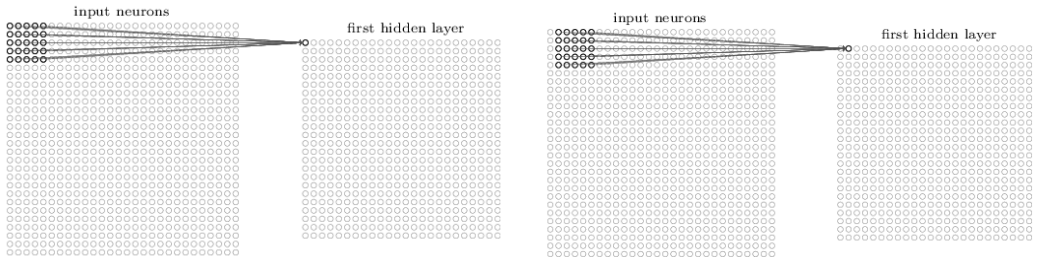


Figure 4.2.3: Convolutional Neural Network part architecture from the CRNN model proposed by Choi et al.

CONVOLUTIONAL LAYER The convolutional layer is used for the extraction of features: given an image, it is processed using a prefixed number of small matrices $m \times m$ (generally 3x3 or 4x4) called filters or kernels. Given a kernel the network computes the elementwise product between a portion of the image, called window, of the same kernel size and the kernel itself. The result is the first element of the new matrix called convolved features or *feature map*. Formally the value of the first element of the feature map of index (j, k) is equal to:

$$\sigma\left(b + \sum_{l=1}^{m}\sum_{r=1}^{m} w_{l,r} a_{j+l,k+r}\right) \tag{4.2.1}$$

where σ is the activation function, b is the shared threshold value, $w_{l,r}$ is the value of the kernel matrix element in position (l, r) and $a_{x,y}$ denotes the point of the input image at position (x, y) . After this step, the kernel window is moved by a fixed number of pixels, called *stride*, and repeats the operation until it processes the entire image. The same operation described for a kernel is also performed for all the others ones, producing as many feature maps as the number of kernels. The so obtained feature maps can be very different from each other based on the values set in the kernel matrices. A process defined in this way allows us to extract different features related to points at a predetermined distance from each other; for this reason it is said that the convolution preserves spatial relations between the features extracted from the input image.



(a) First step of convolution

(b) Second step of convolution

Figure 4.2.4: Two-step Convolution operation for a 5x5 kernel and stride = 1 [4]

A convolutional neural network that uses only one convolution layer extracts features at a first level of abstraction. If we add another layer, the features taken by this layer are at a greater level of abstraction, since they are extracted from the feature maps produced by the previous convolutional layer instead from the input image. This action allows us to generate features to a higher abstraction, which allows us to refine the recognition of objects in an image. In addition, refining features allows us to extract informations about their spatial location or internal composition, e.g. determine the presence of a person in a car or a brand on a garment. The more the number of convolutional layer increases, the more the abstraction of the feature is done; in practice the more the network goes in depth, the more the abstraction detail of the features and their spatial relationships is kept. This means that the more filters and layers the network has, the more it is able to learn better a greater number of different kind of input data. The fundamental elements that influence the functioning of a convolutional layer are the following:

- Depth: Depth corresponds to the number of filters we use for the convolution operation.
- Stride: Stride is the number of pixels by which we slide our filter matrix over the input matrix: e.g. if the stride is 1 then we move the filters one pixel at a time. Having a larger stride will produce smaller feature maps and more information get lost.
- Zero-padding: Sometimes, it is convenient to pad the input matrix with zeros around the border, so that we can apply the filter to bordering elements of our input image matrix. A nice feature of zero padding is that it allows us to control the size of the feature maps. Adding zero-padding is also called wide convolution, while not using zero-padding is called a narrow convolution.

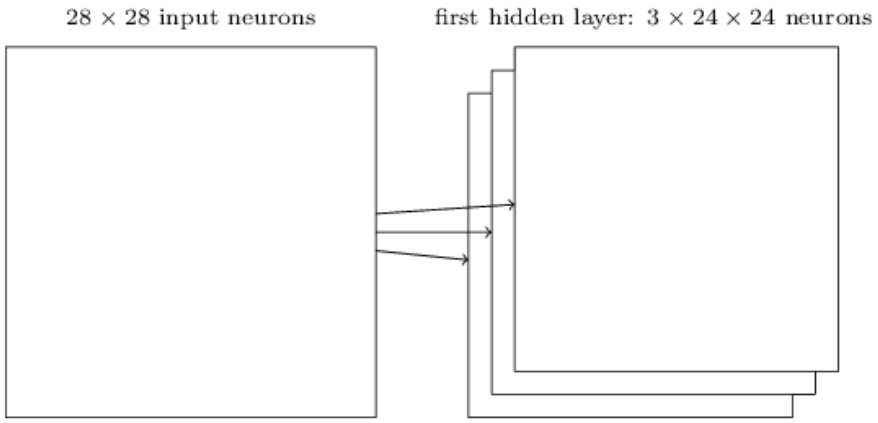


Figure 4.2.5: Result of the convolution step for 3 kernel [4]

ACTIVATION FUNCTION: ELU After extracting the features from the image using the convolutional layer, an activation function is applied to the feature maps, which we had previously referred to as σ. In neural networks some of the most known activation functions are the sigmoid, the softmax and ReLU. ReLU stands for Rectified Linear Unit and it is a non-linear function. Its output is given by:

$$f(x) = \max(0, x)$$

ReLU is an elementwise operation and replaces all pixels having a negative value in the feature map with a zero; the feature map thus obtained is called *Rectified feature map*. The purpose of the ReLU is to introduce a non-linearity in the convolutional neural network, since most of the real-world data we want our network to learn about are just non-linear.

In our network we use a newer activation function instead of ReLU, the Exponential Linear Unit (ELU), expressed by the following functions:

$$f(x) = \begin{cases} x & if\ x > 0 \\ \alpha(\exp(x) - 1) & if\ x \leqslant 0 \end{cases} \tag{4.2.2}$$

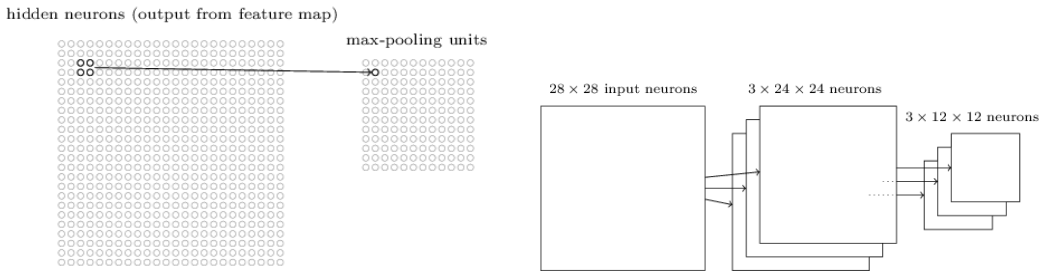
$$f'(x) = \begin{cases} 1 & if\ x > 0 \\ \alpha \exp(x) = \alpha \exp(x) - \alpha + \alpha = f(x) + \alpha & if\ x \leqslant 0 \end{cases} \tag{4.2.3}$$

with $\alpha > 0$ [3].

An issue that often looms in neural networks is that of the so called vanishing gradient,

i.e. the contribution that the furthest layers from the output provide to the classification is reduced more and more after many steps of back-propagation. This effect is mitigated by ELU, since the α parameter controls the saturation value of ELU for negative input of the network, i.e. when the derivative during back-propagation reaches a value so low that it does not bring any variation in the information that is processed successively from the network. The experiments show how the use of ELU leads to better learning results by neural networks in terms of time processing and accuracy. [3]

POOLING LAYER: MAX POOLING Spatial Pooling (also called subsamplig or downsampling) reduces the dimensionality of each feature map, keeping the most important information. Spatial pooling reduces feature map dimensionality by applying a spatial neighborohood, a modest size window, which passes on the rectified feature map and takes its most relevant value according to a defined rule; if we use Max Pooling as in our case, the element with the highest value is takens within that window. This element composes a new reduced feature array. The spatial neighborohood slides on the rectified feature map in order to cover each pixel only once; the maximum element of the covered portion is taken and added as a new element of the reduced feature map. This process is repeated for every rectified feature maps, giving rise to the respective reduced feature maps.



(a) Max Pooling step operation (b) Results matrices after Max Pooling operation

Figure 4.2.6: Max Pooling operation and results [4]

The pooling operation prevents the problem of overfitting. This problem can easily occur during the learning phase of neural networks: the processed data by the network contain useful information plus a quantity of noise; if the network learns too much from the data, it also learns the noise, making it more difficult to classify other data different from those learned. When this happens, it is said that the network is no longer able to make generalization.

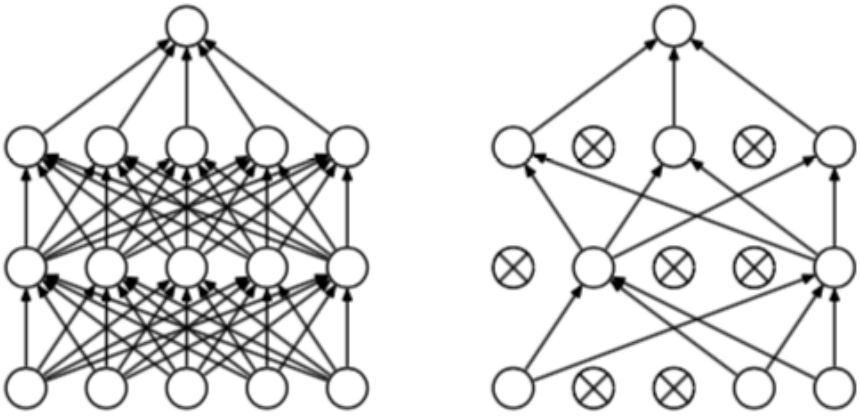
Definition 3. (Generalization). Generalization is the ability of a network to provide correct answers to examples not encountered during the training phase (test examples).

The pooling operation is applied to each rectified feature map. The purpose of pooling is to progressively reduce the spatial dimension of the representation of the input. In particular, pooling:

- makes the size of features related to the input smaller and more manageable
- reduces the number of parameters and computations in the network, controlling overfitting
- makes the network invariant to small transformations, distortions and translations of the input image (since we take the maximum)
- helps us to achieve at an almost invariant or equivariant representation scales of the input image. This is very powerful because we can locate objects in an image no matter where they are and how they are arranged.

REGULARIZATION LAYER: DROPOUT Once the feature maps have been extracted, rectified by the ELU activation function and resized using Max Pooling, another layer, called Dropout, is used to further regularize the data. This step requires that the neurons belonging to this layer are set to be active according to a certain probability p, or inactive by setting them to zero otherwise; this makes it possible to take the only information that can improve learning, reducing residual noise and consequently the overfitting.

> "During training, Dropout can be interpreted as sampling a Neural Network within the full Neural Network, and only updating the parameters of the sampled network based on the input data. (However, the exponential number of possible sampled networks are not independent because they share the parameters.)" [13]



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

Figure 4.2.7: Dropout Neural Net Model. Left: A standard neural net with 2 hidden layers. Right: An example of a thinned net produced by applying dropout to the network on the left. Crossed units have been dropped [7].

4.2.2 Recurrent Neural Network

Recurrent Neural Network (RNN) is a class of network that processes input data, exploiting the collected information from previous input data. In this way it maintains the temporal relation between data passed as input, e.g. the one existring between frames of a movie.

RNN models vary among them essentially for the level of depth of the temporal relations that we wants to take into account between the data to be evaluated. Starting from Vanilla RNN, which only takes into account the information taken from data in the previous step, we get to more complex structures. This is the case of LSTM, a structure that during processing decides which parts of the collected information is to store or to forget.

ARCHITECTURE The model of Recurrent Neural Network (RNN) of our CRNN is composed by the following layers:

- A two-layered GRU
- One Dense layers, or fully-connected layer

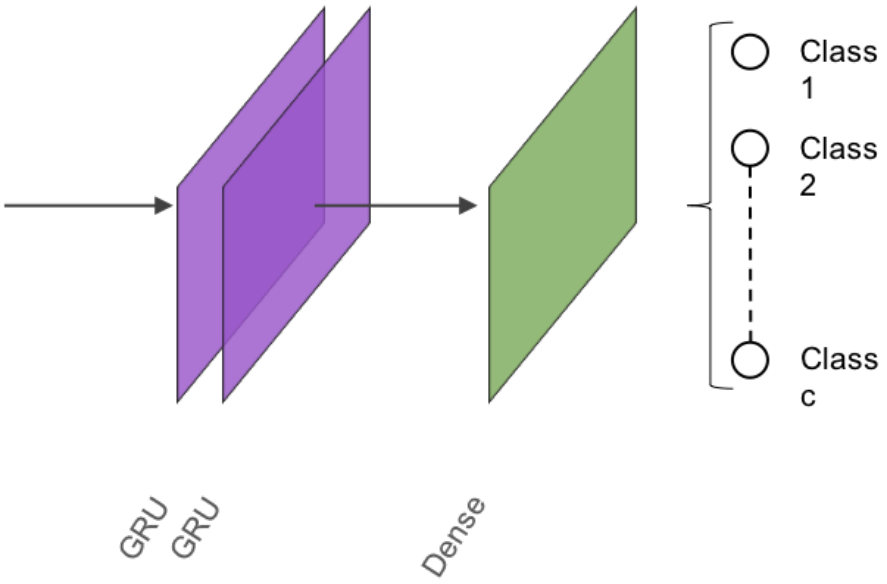


Figure 4.2.8: RNN and classification layer architecture part of the CRNN proposed by Choi et al.

GATED RECURRENT UNIT (GRU) The simplest model of RNN, also known as Vanilla RNN, has as input the input data at time t and the features produced by the network from the input of the previous time *t-1*. Such a model picks the temporal relations between two adjacent input data, but it is not suitable if such relationships propagate for a long time period. In this case, it is used a model that allows in some way to "remember" the processed information and to incorporate new information by selecting the part of information that can be forgotten and the one to be kept. This is the case of the Long Short Term Memory (LSTM), whose gated structure, composed by a memory state, the processed data passed as input at time *t-1* and the current input, allows to select the relevant information using activation functions. Recently, another RNN model has been developed based on the same concepts of LSTM, the Gated Recurrent Unit (GRU). The main characteristic of the GRU is that, unlike the LSTM, it does not contain a cell state of memory. This allows to occupy less memory for storing the computed information.

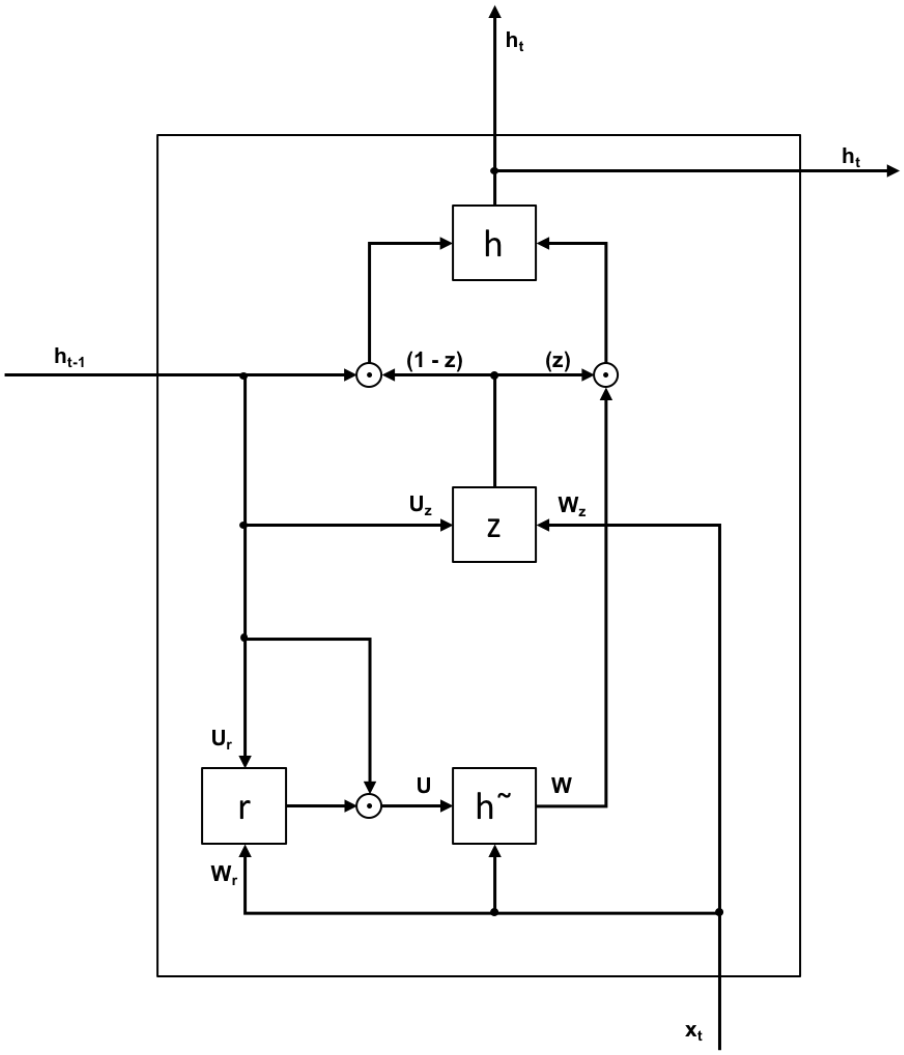


Figure 4.2.9: Gated Recurrent Unit (GRU) architecture

Figure (4.2.9) shows a GRU structure. GRU is composed by four gates: activation gate h, update gate z, candidate activation $\widetilde{h}$ and the reset r.

Let's define the following elements:

- x_t input array at time t.
- h_{t-1} the computed information at time *t-1*.

The reset gate r takes care of deciding how much to delete of h_{t-1} based on the new input data x_t.

$$r_t^j = \sigma\left(W_r x_t + U_r h_{t-1}\right)^j \tag{4.2.4}$$

The candidate activation $\widetilde{h}$ computes the information of input data at time t with h_{t-1} filtered by the reset gate r. This element represents the data that determines the output value of the network at time t.

$$\widetilde{h}_t^j = \tanh\left(W x_t + U\left(r_t \odot h_{t-1}\right)\right)^j \tag{4.2.5}$$

The update gate z decides how much of the input data at time t and the passed information h_{t-1} influences the output of the network at time t.

$$z_t^j = \sigma\left(W_z x_t + U_z h_{t-1}\right)^j \tag{4.2.6}$$

Finally, the activation gate is responsible for producing the network output at time t. The output is given by the linear interpolation between h_{t-1} and the candidate activation $\widetilde{h}$ at time t.

$$h_t^j = \left(1 - z_t^j\right) h_{t-1}^j + z_t^j \widetilde{h}_t^j \tag{4.2.7}$$

It has been prooved through experiments that LSTM and GRU have on average the same accuracy results for input data with average long time dependencies; if data to process have very long time dependencies, LSTM is more suitable. [5]

GRU takes in input the output produced by the last layer of the CNN, processing it further. The addition of a second GRU layer allows to capture higher level of time dependencies exactly in the same way as described for Convolutional layer in CNN.

DENSE LAYER Once the data has passed through CNN first and RNN then, the data characterized by the extracted features must be classified. In order to do this we introduce a layer of fully connected neurons that sums up the computed features. Finally, an activation function provides a classification pattern belonging to the standard simplex Δ_c, where c is the number of classes, whose maximum value determines to which class the input having such features belongs to.

4.2.3 Learning the network

Once the network has processed the input data, the prediction error is computed using the so-called Loss function, which evaluates the distance of the results obtained from the network with respect to what labels were expected. This distance or error is then

propagate backwards in the network updating the weights between layers down to the input layer. This process is called back-propagation and is applied using the gradient descent method, which updates the weights of the network. The update of the network weights based on the produced error allows to improve the learning ability of the network.

> "Gradient descent is a way to minimize an objective function $J(\theta)$ parameterized by a model's parameters $\theta \in \mathbb{R}^d$ by updating the parameters in the opposite direction of the gradient of the objective function $\nabla\theta J(\theta)$ w.r.t. to the parameters. The learning rate η determines the size of the steps we take to reach a (local) minimum."[6]

The learning of a neural network is a supervised classification method using a dataset of labeled objects as input. The dataset is divided into three parts, which are called training set, validation set and test set. The training set is used to train the network in periods called epochs, during which the loss function computes two important values: loss and accuracy. These two values indicate respectively the error and accuracy produced by the network. Over the epochs, the network refines learning, observing a decreasing value of the loss and a consequent increasing value of accuracy. At the end of each epoch, the validation set is classified, taking the loss and accuracy values as a check for the achieved learning rate. A network designed to generalize well produces correct input/output mappings even if the input is slightly different from the examples used during the training phase. If the network has being trained too much, we go to risk of adapting too much to the training set data, since the network learns the noise contained in the data. This phenomenon is called overfitting: an over-trained network is too rigid and therefore loses its generalization capacity. In order to prevent this problem, the so-called "Early Stopping" technique is used: learning on the training set is carried on until the epoch at which the loss value taken for the validation set begins to increase, moment at which the effects of overfitting begin.

Once the learning phase is complete, the classification ability of a network is evaluated on the loss function values computed for the test set.

4.2.4 Complete deep learning model architecture

In figure (4.2.10) we can see the model of the complete network. The network consists of four groups of "convolutive blocks", each consisting of convolutional layer, ELU, Max Pooling and Dropout. After the last convolutive group the extracted features are passed as input to the recurrent neural network. At the end of the computation by the RNN

the output passes through a fully connected layer, which produces a probability vector belonging to the standard simplex indicating the classification rates.

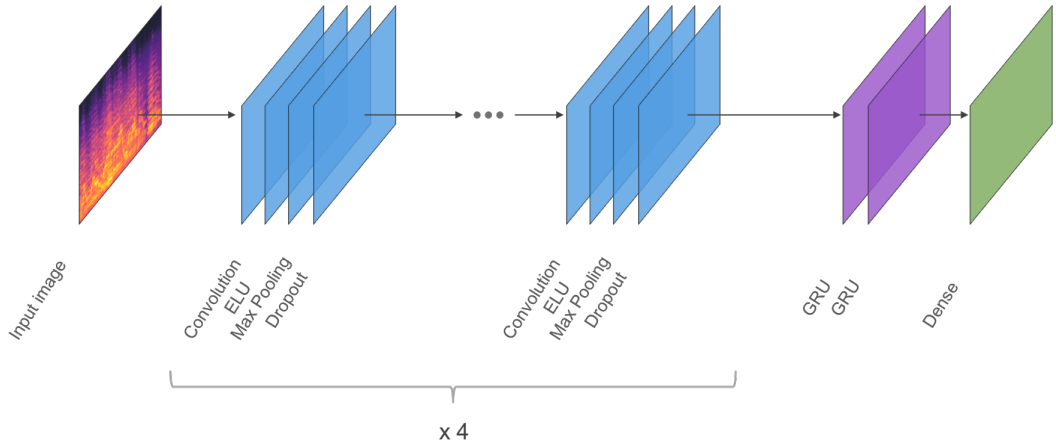


Figure 4.2.10: Convolutional Recurrent Neural Network architecture

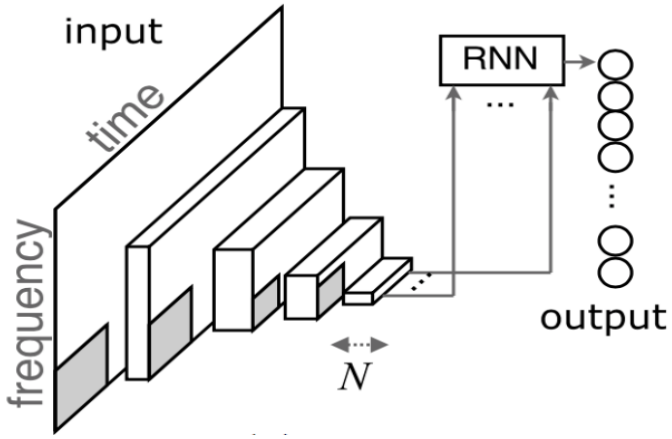


Figure 4.2.11: Block diagrams of CRNN. The grey areas illustrate the convolution kernels. N refers to the number of feature maps of convolutional layers [2]

4.3 Graph Transduction Game

In this section we explain the second module of our model: the graph transduction game. We describe this technique starting from the theoretical explanation in the article by Pelillo et al. 2011 [1].

4.3.1 Some Theory

The graph transduction game is a semi-supervised learning method for object classification. The method is mainly based on the techniques of transduction learning and game theory.

The problem of classification is to assign a class to a set of objects. Formally, assuming for simplicity to have only one class C, given a set of features X and a space of labels $Y = \{1, -1\}$ we define a classifier the function $f : X \times Y$ that assigns a label to each object, which is represented by a feature vector X_i. The set $L = \{(X_1, y_1), \dots (X_n, y_n)\}$ is called training data, where

$$y_i = \begin{cases} 1 & if\, X_i \in C \\ -1 & otherwise \end{cases}$$

The same concept can be expressed by the use of an undirected graph with binary weights $\{0, 1\}$: let $G = (D, E, w)$, where D is the set of data points, E the set of edges of the graph and $w : (D \times D) \to \{0, 1\}$ the function that assigns a weight to the edges. Two nodes i and j are said to be similar, i.e. belong to the same class, if $w_{ij} = 1$; otherwise they are said to be dissimilar. The labeling described above leads to dividing objects into two sets or partitions, that are namely defined as clusters. Clusters are sets of objects having maximum internal homogeneity and maximum external non-homogeneity, i.e. the objects contained in a cluster are very similar to each other and at the same time very dissimilar from the objects that are outside.

The transductive learning technique is said to be semi-supervised, i.e. it requires that the set of data to be classified is only partially labeled; the dataset is then defined as $D = \{D_l, D_u\}$, where $D_l = \{d_1, \dots, d_l\}$ is the subset of labeled data points and $D_u = \{d_{l+1}, \dots, d_n\}$ the one of unlabeled data points. The purpose of this technique is to classify the unlabeled data points based on their similarities with those whose class membership is already known. Formally, given a set of labels $\Phi = \{1, \dots, c\}$ and denoting with $\{\phi_1, \dots, \phi_l\}$ the set of labels associated to the labeled points D_l, we have to estimate the set of labels $\{\phi_{l+1}, \dots \phi_n\}$ for the unlabeled points D_u. In order to do this, we can use

the concept of clusters in semi-supervised learning:

> "The assumption simply states that (1) points which are close to each other are expected to have the same label, and (2) points in the same cluster (or on the same manifold) are expected to have the same label." [1].

In order to define the similarity between the data points we need to specify rules that allow us to associate a value that can measure this quantity. The constraints satisfaction problem (CSP) technique allows us to redefine the model with the following elements:

- The graph of the data points $G = (D, E)$, where D is the set of data points and E the set of edges that connect them
- A set of variables $V = \{v_1, \ldots, v_n\}$ associated with data points $1, \ldots, n$
- Domains: let's define $D_V = \{D_{v_1}, \ldots, D_{v_n}\}$, with $D_{v_i} = \begin{cases} \{\phi_i\} & \forall i : 1 \leqslant i \leqslant l \\ \Phi & \forall i : l+1 \leqslant i \leqslant n \end{cases}$ the set of domains of labels associated to the variables v_i
- Binary Constraints: we define A the matrix of binary constraints, where the element $a_{ij} = 1$ if data points v_i and v_j respects the following constraint:
  $\forall i, j$: if $a_{ij} = 1$, then $v_i = v_j$.

> "Each assignment of values to the variables satisfying all the constraints is a solution of the CSP, and provides a consistent labeling for the unlabeled points" [1].

The notion of consistency of constraints is related to the concept of Nash equilibrium in non-cooperative games [1]. In order to explain it better we need to reformulate the problem using game theory notations. Game Theory is a mathematical discipline, whose purpose is to analyze the strategic behaviors of decision makers (players), or to study the situations in which different players interact, pursuing common, different or conflicting goals.

> "In normal form, a game with many players can be expressed as a triple $G = (\mathcal{I}, S, \pi)$, where $\mathcal{I} = 1, ..., n$, with $n \geq 2$, is the set of players, $S = \times_{i \in I} S_i$ is the joint strategy space defined as the Cartesian product of the individual pure strategy sets $S_i = 1, ..., m_i$, and $\pi : S \to \mathbb{R}^n$ is the combined payoff function which assigns a real valued payoff $\pi_i(s) \in \mathbb{R}$ to each pure strategy profile $s \in S$ and player $i \in \mathcal{I}$" [1].

In a two-player game, in which i, j are the two players, payoff functions can be represented as two matrices $m_i \times m_j$, called payoff matrices.

> "A mixed strategy of player $i \in \mathcal{I}$ is a probability distribution over its pure strategy set S_i, which can be described as the vector $x_i = (x_{i1}, \ldots, x_{im_i})^T$ such that each component x_{ih} denotes the probability that the player chooses to play its h^{th} pure strategy among all the available strategies" [1].

In a two-player symmetric game, where the related payoff matrices A and B are defined to be $A = B^T$, a mixed strategy x is said to be a symmetric Nash equilibrium if

$$x^T Ax \geqslant y^T Ax, \forall y \tag{4.3.1}$$

that is, the highest payoff against strategy x can only be achieved by playing the same strategy, for each possible strategy y for the other player. If we extend the game to n players, we talk about evolutionary games, in which the players decide which class they belong to based on the payoff and how many players join that class. The distribution of players between classes is updated from one epoch to another and is comparable to the growth of a population according to the so-called Darwinian fitness or reproductive success. In this kind of games, a Nash equilibrium x is said to be an Evolutionary Stable Strategy (ESS) if, for all strategies y, the following implication is verified:

$$y^T Ax = x^T Ax \Rightarrow x^T Ay > y^T Ay \tag{4.3.2}$$

Such a problem would normally belong to the NP-complete complexity class, hence with a very high computational cost. The solution to overcome this problem comes from replicator dynamics. The replicator dynamics are a system of ordinary differential equations (ODE) that allows to find the mixed strategy x that solves the following equation:

$$\max\left(\bar{x}^T A\bar{x}\right) \tag{4.3.3}$$

with $\bar{x} \in \Delta_c$.
Let's illustrate how replicator dynamics works in the discrete case. Let's define:

1. *pre-programmed strategy*: the population is partitioned into n subset each having a particular game strategy. Let $S = \{1, 2, \ldots, n\}$ be the set of strategies

2. *state* $\bar{x}$: the state $\bar{x}$ of a system at time t will be indicated with $\bar{x}\,(t)$ and its i-th element $x_i\,(t)$ represents the portion (percentage) of individuals in the population

who play with the preprogrammed strategy i at the instant t. This value can also be void; this means that there is nobody in that instant (epoch) t that is programmed to play that strategy.

3. *fitness matrix*: let W be a matrix in which the element $w_{i,j}$ represents the average payoff that the player who plays the strategy i gets on a player who plays the strategy j

Given a certain state $\bar{x}$ a player who plays the strategy i gains a total payoff π_i equal to:

$$\pi_i(t) = \sum_{j=1}^{n} w_{ij} x_j(t) = (W \cdot \bar{x}(t))_i \tag{4.3.4}$$

that is equal to the sum of the payoff towards any other player, weighed on the percentage of people who play that strategy.

All the polpolation will have a total payoff given by:

$$\pi(t) = \sum_{i=1}^{n} x_i(t) \cdot \pi_i(t) = \left(\bar{x}^T \cdot W \cdot \bar{x}\right) \tag{4.3.5}$$

In every epoch the rate of players, who play a certain strategy i, will increase or decrease compared to the total population, in proportion to the success that its strategy had in the previous epoch. The success of the strategy i is measured by comparing the payoff of the player who plays the strategy i with the total population payoff:

$$success_i(t) = \frac{\pi_i(t)}{\pi(t)};$$

in practice, if a strategy is effective, then the portion of the population that will use it will be greater. The so-called discrete replicator equations is thus obtained:

$$x_i(t+1) = x_i(t) \cdot \frac{\pi_i(t)}{\pi(t)} \tag{4.3.6}$$

Replicator dynamics get started with a certain vector $x(0) = (x_1(0), ..., x_n(0))$. The *fundamental theorem of natural selection* states that if the weight matrix W is symmetric, i.e. $W = W^T$, then the following equation can be associated to the system:

$$F(t) = \bar{x}^T \cdot W \cdot \bar{x} = \sum_i \sum_j w_{ij} x_i(t) x_j(t) \tag{4.3.7}$$

which is a Lyapunov function for the system. In this case the function is monotonically non-decreasing along a trajectory determined by replicator dynamics up to converge to a

local maximum. Function (4.3.6) guarantees us that, starting from $x(0) \in \Delta_n$, at each step the vector $x(t)$ belongs to the standard simplex Δ_n . A stationary point is reached, when replicator dynamics find a solution to the equation (4.3.3) in polynomial time complexity. The vector of mixed strategies x that maximizes the payoff is an evolutionary stable strategy. It is prooven that "The discrete-time replicator dynamics has essentially the same dynamical properties as the continuous version" [1].

We now define the Graph Transduction Game according to game theory. Specifically, we can think of our classification problem as a non-cooperative multi-player game in which players compete to get the class that gives them the highest score, namely:

> "Non-cooperative game theory deals with models of strategic interactions (games) among anonymous agents (players), where the goal of each player is to maximize its own utility or payoff. Each player has a set of possible actions (pure strategies) to play, called the pure strategy set, and receives a payoff based on its own choice and those of the other players" [1].

Let's define the Graph Transduction Game as the normal game $G = (\mathcal{I}, S, \pi)$, where $\mathcal{I} = \mathcal{I}_l \cup \mathcal{I}_u$ is the union of the set of labeled data points $\mathcal{I}_l$ and the one of the unlabeled data points $\mathcal{I}_u$, S the set of the joint strategy space of the pure strategies and π the payoff function. Given the mixed strategy x_i associated to player i, the goal of the Graph Transduction Game is to find

$$\phi_i = \arg \max_{h=1,\dots,c} x_{ih} \tag{4.3.8}$$

for all $i \in \mathcal{I}_u$.

In a game where there are labeled and unlabeled players, the first ones are considered to have fixed strategies, since they are already defined. The expected payoff for a player against players with fixed strategies can be indicated as:

$$u_i(x) = \sum_{s \in S} x(s)\, \pi_i(s) = \sum_{k=1}^{m_j} u_i\left(e_j^k, x_{-j}\right) x_{jk} \tag{4.3.9}$$

for each strategy profile $s \in S$, where with e_i^h, called *extreme mixted strategy*, we mean the vector of the mixed strategy i having all the elements set to zero except for the h-th set to 1 [1].

Hence the payoff matrices between each pair of players are built using respectively the following expected payoff functions for a player $i \in \mathcal{I}_u$:

$$u_i\left(e_i^h\right) = \sum_{j\in\mathcal{I}_u}\left(A_{ij}x_j\right)_h + \sum_{k=1}^{c}\sum_{j\in\mathcal{I}_{D|k}} A_{ij}\left(h,k\right) \tag{4.3.10}$$

$$u_i\left(x\right) = \sum_{j\in\mathcal{I}_u} x_i^T A_{ij}x_j + \sum_{k=1}^{c}\sum_{j\in\mathcal{I}_{D|k}} x_i^T\left(A_{ij}\right)_k \tag{4.3.11}$$

Function (4.3.10) indicates the payoff obtained by the player i in the hypothesis that he has fixed label h compared to the other unlabeled players who play the same strategy h and respect to all the players for each possible strategy. Function (4.3.11) represents the same formulation considering instead all the possible strategies playable by the player i.

Let $\beta_i\left(y\right) = \{x_i \in \Delta_i : u_i\left(x_i, y_{-i}\right) \geqslant u_i\left(z_i, y_{-i}\right) \forall z_i \in \Delta_i\}$ the mixed best replies, which is the set of mixed strategies for player i against a mixed strategy y, where $u_i\left(x_i, y_{-i}\right)$ is the payoff obtained by the player i playing the mixed strategy x_i against the mixed strategy y_{-i}, i.e. y_j with $j \in \mathcal{I}\backslash\{i\}$.

Referring to function (4.3.1), let's give a more general definition for the Nash equilibrium:

Definition 4. A mixed strategy $x^* = (x_1^*, \ldots, x_n^*)$ is said to be a Nash Equilibium x^* if it is the best reply to itself, $x^* \in \beta\left(x^*\right)$, that is

$$u_i\left(x_i^*, x_{-i}^*\right) \geqslant u_i\left(x_i, x_{-i}^*\right) \tag{4.3.12}$$

for all $i \in \mathcal{I}, x_i \in \Delta_i$ and $x_i \neq x_i^*$. Furthermore, a Nash equilibrium x^* is called strict if each x_i^* is the unique best reply to x^*, $\beta\left(x^*\right) = \{x^*\}$.

The computations of Nash equilibria are carried out using an evolutionary approach [1], whose dynamic interpretation is expressed with the system of ordinary differential equations

$$\dot{x}_{ih} = g_{ih}\left(x\right)x_{ih} \tag{4.3.13}$$

> "where a dot signifies derivative with respect to time, and $g\left(x\right) = \left(g_1\left(x\right), \ldots, g_n\left(x\right)\right)$ is the growth rate function with open domain containing $\Theta = \times_{i\in\mathcal{I}}\Delta_i$, mixed strategy space, each component $g_i(x)$ being a vector-valued growth rate function for player i" [1].

Among the regular selection dynamics there are the so-called payoff monotonic dynamics, which have the following property:

$$u_i\left(e_i^h, x_{-i}\right) > u_i\left(e_i^k, x_{-i}\right) \Leftrightarrow g_{ih}(x) > g_{ik}(x) \tag{4.3.14}$$

for all $x \in \Theta, i \in \mathcal{I}$ and pure strategies $h, k \in S_i$. The evolution of the behavior applied to the multi population version is given by the following development of the formula (4.3.13):

$$\dot{x}_{ih} = x_{ih}\left(u_i\left(e_i^h, x_{-i}\right) - u_i(x)\right) \tag{4.3.15}$$

This system of differential equations can be rewritten in discrete case in the same way of function (4.3.6) as follows

$$x_{ih}(t+1) = x_{ih}(t)\frac{u_i\left(e_i^h\right)}{u_i(x(t))} \tag{4.3.16}$$

and it is prooven that it converges to a stationary point thanks to the following theorem:

Theorem 5. *A point $x \in \Theta$ is the limit of a trajectory of (4.3.15) starting from the interior of Θ if and only if x is a Nash equilibrium. Further, if point $x \in \Theta$ is a strict Nash equilibrium then it is asymptotically stable, additionally implying that the trajectories starting from all nearby states converge to x. [1]*

For further details, refer to the reading of the paper by Pelillo et al. 2011 [1].

4.4 The full model architecture

The model we have used consists of two modules: one that implements a deep learning structure and one that performs the GTG algorithm. The first module consists of a compound neural network by a convolutional neural network and a recurrent neural network. Its task is, given an input, to extract its features and to provide its own classification pattern based on the network ability to learn. The second module deals with refining the classification results proposed by the network, producing more consistent ones using similarity measures built on the extracted input features. Once the dataset has been divided into labeled and unlabeled input data, the GTG algorithm takes care of completing the labeling on the second ones. The algorithm produces its own classification pattern which is then compared to the one computed by the network. The GTG classification process is then repeated until we get the maximum accuracy.

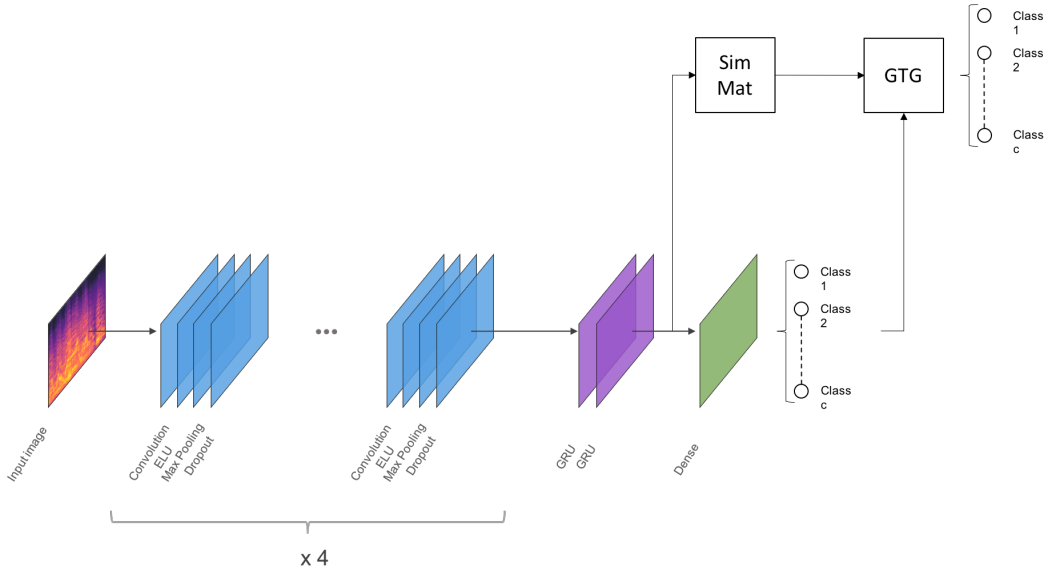


Figure 4.4.1: Model architecture used in this thesis

4.5 Classification process

The classification process is composed by the following steps:

1. Dataset early prediction: processing of the training and the test set exporting the produced featured before the classification layer of the network
2. Dataset prediction: prediction of the training and the test set
3. Similarity matrix construction: build the similarity matrix with the data extracted before the last layer of the network
4. Graph transduction game computation: apply the GTG to the data described in the previous steps
5. Predictions comparison: compare the results obtained by GTG with those from the network

Let's give a brief explanation of each steps.

Dataset early prediction

The dataset is processed by the network until the processing arrives at the classification layer. Then the features to be categorized are extracted in order to build the similarity matrix for GTG algorithm.

Dataset prediction

Once the features have been extracted before classification, let's perform the classification by the network. Such classification patterns represent the mixed strategies of the data that "play" in the Graph Transduction Game.

Similarity matrix construction

The construction of the similarity matrix is made starting from the features extracted in step 2).

Graph transduction game computation

The Graph Transduction Game algorithm takes as input the similarity matrix between data points, the lists of the labeled and unlabeled data points, the predictions produced by the network and the original labels. Then the algorithm assigns classes to unlabeled date points iteratively.

Predictions comparison

Once the GTG has produced the classification for the unlabeled data, the algorithm compares it with the original labeling, calculating the accuracy at each iteration.

5 Experiments and results

In this chapter we talk about the experiments we have performed. The aim of these experiments is to study the behavior of Graph Transduction Game comparing it with the classification results obtained by the proposed neural network in different situations. In this chapter we describe the implementation details of the used structures and technologies, the performed experiments, the obtained results and their discussion.

5.1 Settings: Implementation details

We now show the general settings that have been used for our experiments.

Dataset. The dataset that was used for these experiments is the GTZAN[35]

- GTZAN consists of 1000 songs equally divided into 10 music genres. Each file represents a 30-second excerpt and is associated with a one-hot vector, where the element equal to 1 indicates the class to which the song belongs. Since the number of files is homogeneous among classes, the dataset has been divided into training set, validation set and test set of 700, 150 and 150 files respectively, having an equal number of elements for each class. The classes in which the elements are distributed are the following:
  - jazz
  - blues
  - reggae
  - pop
  - disco
  - country
  - metal
  - hiphop
  - rock
  - classical

Structures. The structures we have used are a Convolutional Recurrent Neural Network with the Graph Transduction Game algorithm

- Convolutional Recurrent Neural Network: neural network proposed by Choi et al. [2] composed of a convolutional neural network and a recurrent neural network interconnected with each other. The convolutional neural network consists of four "convolutional modules", each composed of a convolutional layer, an ELU, a Max Pooling and a Dropout. The recurrent neural network consists of a two-layered Gated Recurrent Unit, at the end of which the dense layer is located for input classification. This neural network has been trained on the Million Song Dataset.
- Graph Transduction Game: algorithm proposed by Pelillo et al. [1] implementing a non-cooperative multiplayer game computed by dynamic replicator.

Input data. The audio files of the dataset have been processed to obtain mel spectrograms according to the following parameters:

- SR = 12000: sampling rate of the input signal
- N_FFT = 512: length of the FFT window
- N_MELS = 96: number of mel bands
- HOP_LEN = 256: number of samples between successive frames.
- DURA = 29.12: length of each sample in seconds, that corresponds to 1366 frames according to the previous data.

A mel spectrogram is represented by a 96×1366 matrix, where 96 is the number of the Mel bands and 1366 is the total number of frames.

Hardware	Operating Systems	Mac OS X 10.3.2, Ubuntu 16.04 LTS		
	Machine specs	Intel i7 4700HQ 2.60-3.6Ghz, 8GB DDR3L SDRAM 1600 MHz		
	Video Card	NVIDIA GTX 860M 2GB GDDR5 (640 Cuda Core)		
Software	Languages	Python 3.5-3.6		
	Libraries	Keras 1.2		
		Tensorflow 0.12.1		
		Theano 0.9.0		
		CUDA 0.8		
		Librosa 0.5.1		
		Experiment 1, Experiment 2		
Experiment Data	Structures	CRNN + GTG		
	Dataset	GTZAN		
	Sample number	1000		
	Partitions	Training set: 70%	Validation set: 15%	Test set: 15%

Table 5.1: General settings and tools

The table summarizes the hardware, software and other data used for the experiments.

5.2 Experiments

We have performed the following two experiments:

1. The first experiment tests the ability of the Graph Transduction Game to produce a better classification for single-label dataset;

2. The second experiment tests the ability of the graph transduction game to propose good labeling for a dataset with few observations compared to what the neural network would do.

In order to perform these experiments, we had to use a technique that allowed the network to produce classifications also for different datasets from the one used for training the network. This technique, called *fine-tuning*, was adopted for all the experiments.

FINE-TUNING A neural network is trained to classify given objects. In order to do this the dataset is divided into training set, validation set and test set; the dataset thus divided allows to train a neural network by processing the training set for the back-propagation and the validation set as a comparison to refine the accuracy of the classification during the learning phase. The network is thus trained for the required classification task. It is noteworthy, however, that the network was trained only for that particular dataset. If

we process another dataset, the network may fail to classify the input data how we want, because the network has learned how to classify objects by classes, i.e. kind of objects, belonging to the previous dataset. In order to ensure that the network can correctly classify data belonging to the new dataset, it is necessary to "extend" the classification also for these new input data. In practice, the network needs to add to the learned features also those that characterize the input data of the new dataset. The technique used to refine the learning of an already trained network is called *fine-tuning*: given an already trained network, the network is trained also for the new dataset, preventing the most remote layers to be updated. The error produced by the classification using the loss function is propagated backwards only for some layers, thus modifying only the weights of the layers closest to the output of the network. In this way the learning already carried out with the previous dataset is maintained, modifying the weights of the last layers to allow to classify the data of the new dataset.

However the elements of the new dataset may belong to classes different from those of the dataset with which the network was originally trained. For this reason fine-tuning is applied by removing the last layer of the network, the one assigned to classification, and replacing it with another specific one for the dataset on which the fine-tuning is performed. In this way the classification patterns are oriented only to the classes provided for the new dataset, thus producing a suitable result.

In the same way as for learning phase, fine-tuning also requires a gradient descent optimizer such as ADAM to adjust the learning rate adaptively, allowing convergence to be achieved within a reasonable time frame.

The method we have shown is part of the so-called transfer learning, a technique for reusing an already trained network with a different dataset. This technique is particularly suitable in those cases in which the dataset to be used consists of a few elements, and therefore not sufficient for a robust training of the network. It has been shown in the experiments that fine-tuning not only allows to obtain consistent results, but also a classification on the same level as the one performed by the network on the original dataset[33].

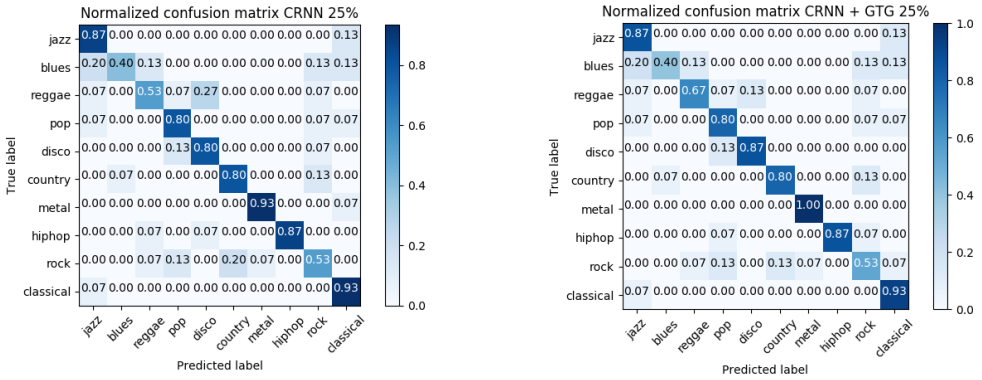
5.2.1 Single label classification

In this experiment we wanted to test how much GTG can improve the classification results of the neural network. In order to do this we tested the CRNN and the GTG on increasing portions of the training set, built adding elements to the previous portions. Finally, we compared the classification results of the CRNN and the GTG performed on the test set.

METHOD The method we have applied for this experiment is as follows:

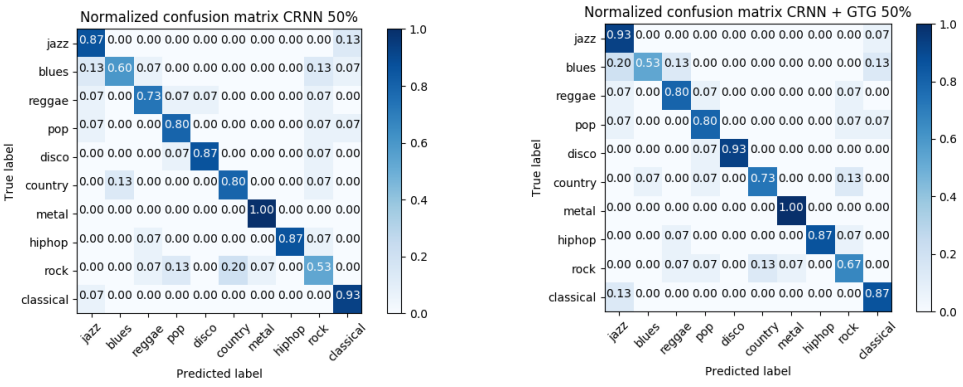
- Reduce the training set in 25%, 50%, 75% and 100% size portions
- For each portion of the training set
  - Fine-tune the network with the portion of the training set and the validation set
  - Process the training and the test set exporting the featureds produced before the classification layer of the network
  - Perform prediction of the training and the test set using the network
  - Build the similarity matrix with the data extracted before the last layer of the network
  - Apply the GTG taking the training set as the labeled data and the test set as the unlabeled data
  - Compare the results obtained by GTG with those by the network

5.2.1.1 Discussing the results



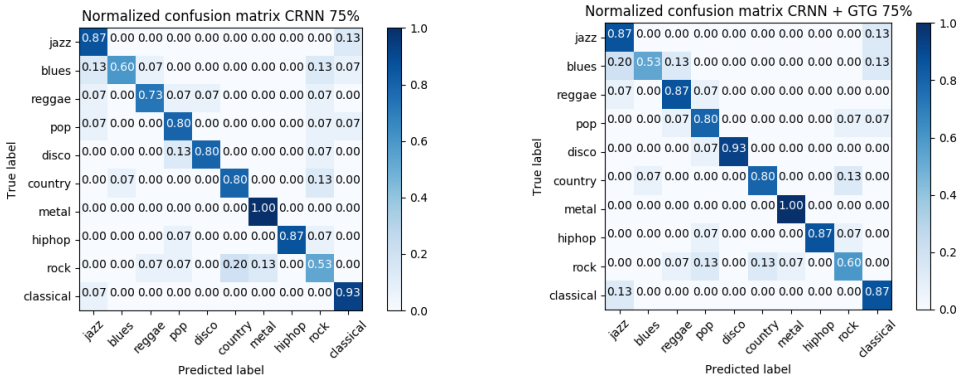
(a) (b)

Figure 5.2.1: CRNN VS CRNN + GTG with 25% labeled training set



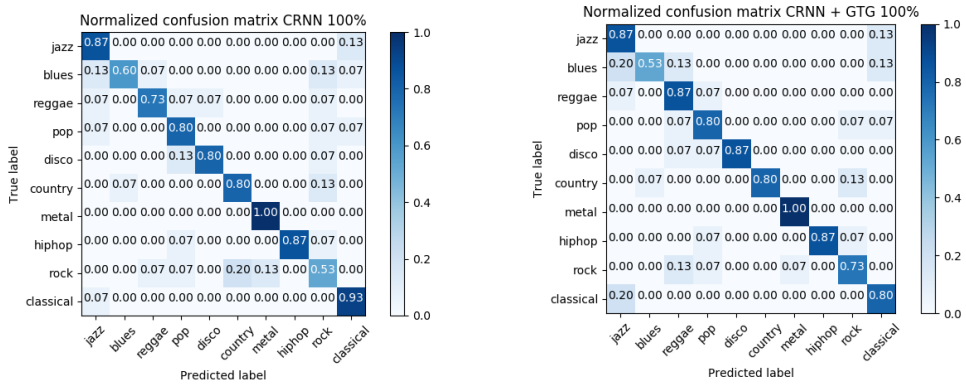
(a) (b)

Figure 5.2.2: CRNN VS CRNN + GTG with 50% labeled training set



(a) (b)

Figure 5.2.3: CRNN VS CRNN + GTG with 75% labeled training set



(a) (b)

Figure 5.2.4: CRNN VS CRNN + GTG with 100% labeled training set

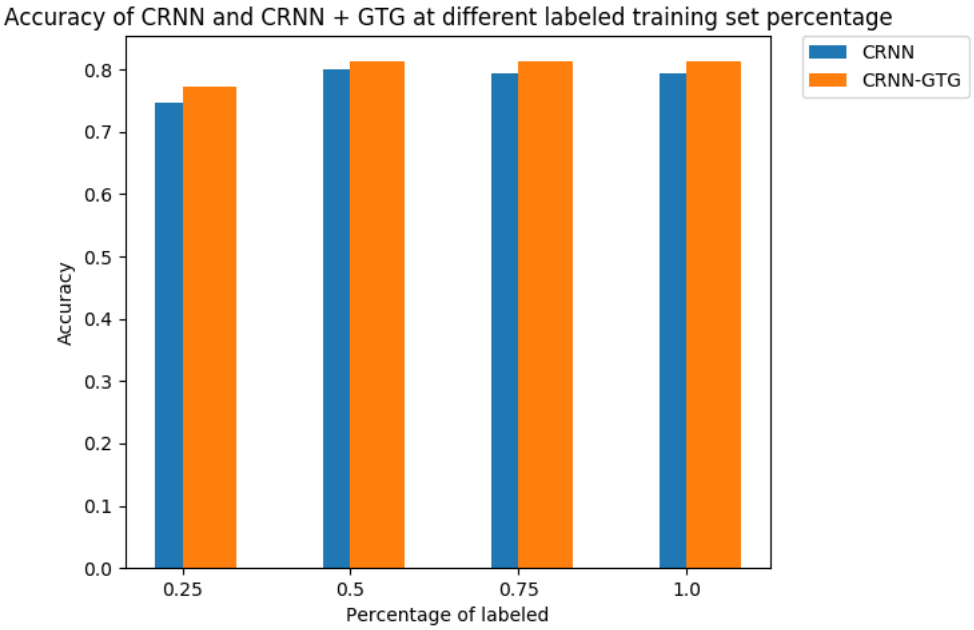


Figure 5.2.5: Accuracy comparison between CRNN and CRNN + GTG at different percentages of labeled training set in single label classification

	Accuracy			
	25%	50%	75%	100%
CRNN	0.746	0.800	0.793	0.793
CRNN + GTG	0.774	0.813	0.814	0.814

Table 5.2: Accuracy value comparison between CRNN and CRNN + GTG at different percentages of labeled training set

The tables and the histogram show a progressive improvement of the classifications obtained by the GTG compared to the one obtained with the CRNN. The slight decline in the accuracy of CRNN is probably due to the fact that, having few observations per class, increasing the portion more noise is introduced due to observations that carry some ambiguity in classification. An example of this is given by the "reggae" class with the 100% partition in which the CRNN distributes the elements in five different classes, while GTG groups them in only three with prevalence of the expected class and with fewer classification errors.

5.2.2 Label Augmentation

Suppose that we are in a real case, in which we have few known observations and many others to be classified. In a case like this the learning of a network is unsuccessful, since the network has few observations on which to generalize. Learning on such a dataset is possible thanks to the so-called *Label Augmentation* technique. This technique consists of artificially increasing the available training set by inferring the class for the unlabeled observations using the information of the known part.

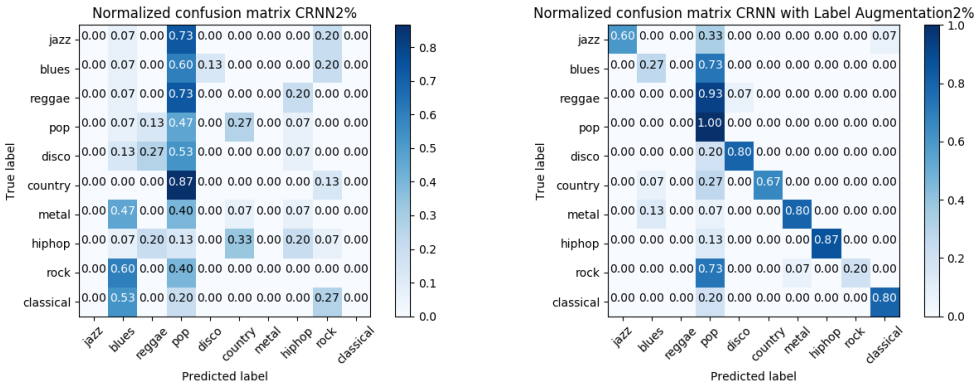
In this experiment we wanted to test how much label augmentation can improve the classification results of the neural network performed on a training set consisting of few observations. In order to do this we used small labeled portions of the training set, progressively built in the same way of the previous experiment, to infer the labels of the remaining part using GTG. The training set obtained in this way was used to perform another fine-tuning of the network with the validation set. Finally we have compared the classification results of the CRNN using the first fine-tuning and the ones of the CRNN applying the label augmentation method.

METHOD The method we have applied for this experiment is as follows:

- Reduce the training set in 2%, 5% and 10% size portions
- For each portion of the training set

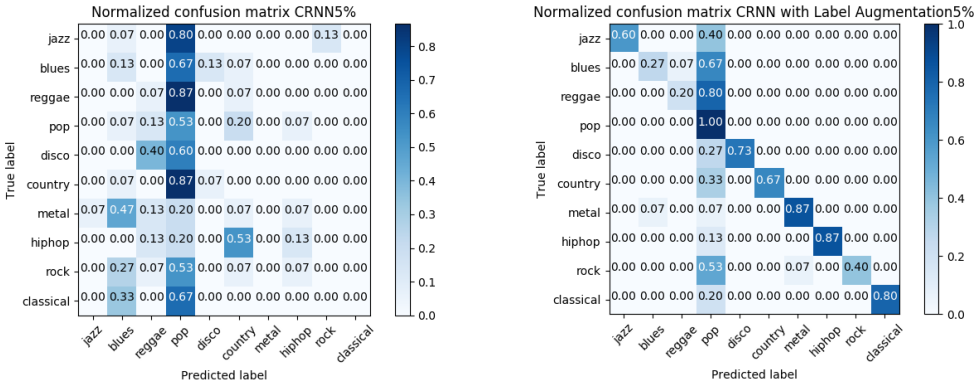
- Fine-tune the network to get the features to infer the labels of the rest of the training set with the GTG
- Apply the GTG, taking the part of training set on which the fine-tuning is done as labeled data and the rest of the training set as the unlabeled data
- Infer the labels of the rest of the training set from the GTG results and use them instead of the original ones as the new labels together with the ones associated to the percentage used for fine-tuning
- Make a new fine-tuning with all the so-labeled training set
- Prediction of the test set on both the fine-tuned version of the network
- Compare the results obtained by the two versions of the network

5.2.2.1 Discussing the results



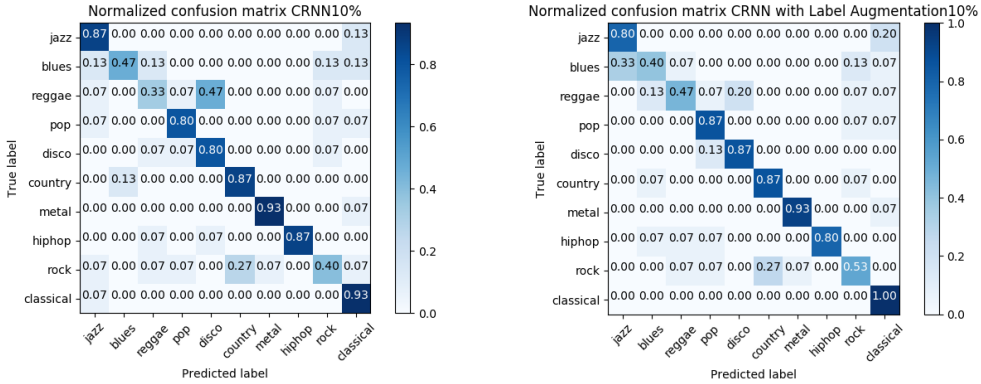
(a) (b)

Figure 5.2.6: CRNN a) and CRNN + label augmentation b) comparison in label augmenting with 2% labeled training set



(a) (b)

Figure 5.2.7: CRNN a) and CRNN + label augmentation b) comparison in label augmenting with 5% labeled training set



(a) (b)

Figure 5.2.8: CRNN a) and CRNN + label augmentation b) comparison in label augmenting with 10% labeled training set

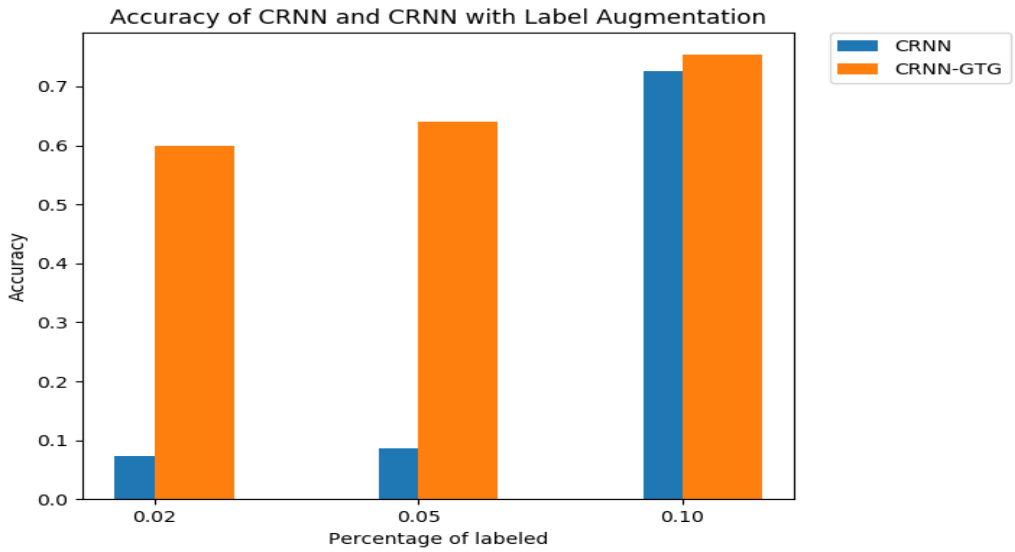


Figure 5.2.9: Accuracy comparison between CRNN and CRNN + label augmentation at different percentages of labeled training set in label augmenting

	Accuracy		
	2%	5%	10%
CRNN	0.074	0.086	0.727
CRNN + Label Augmentation	0.61	0.641	0.754

Table 5.3: Accuracy value comparison between CRNN and CRNN + label augmentation at different percentages of labeled training set in label augmentation

The tables and the histogram show how the augmentation label allows us to increase the size of the training set. The so-inferred observations allow to obtain discreet results even using very small training sets. The problem in recognizing each music genre lies in the inclusion in the initial percentages of the training set of significant data for each class. This explains the breakthrough in quality passing from 5% to 10%.

5.2.2.2 A variant of Label augmentation

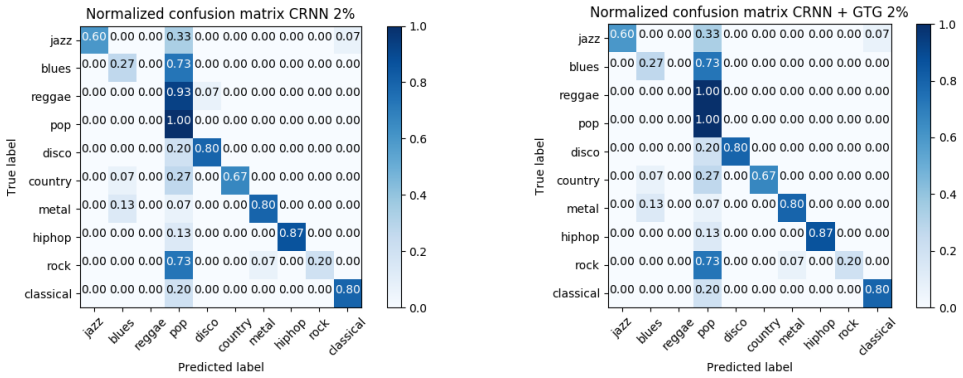
We have then performed another experiment combining the two methods of the single label classification and the one of label augmentation. In this experiment we wanted to test how much GTG can improve the classification results of the neural network performed on a training set consisting of few observations. In order to do this we used small labeled

portions of the training set, progressively built in the same way of the previous experiment, to infer the labels of the remaining part using GTG. The training set obtained in this way was used to fine-tune the network with the validation set. Finally we have compared the classification results of the CRNN and the GTG performed on the test set.

METHOD The method we have applied for this experiment is as follows:

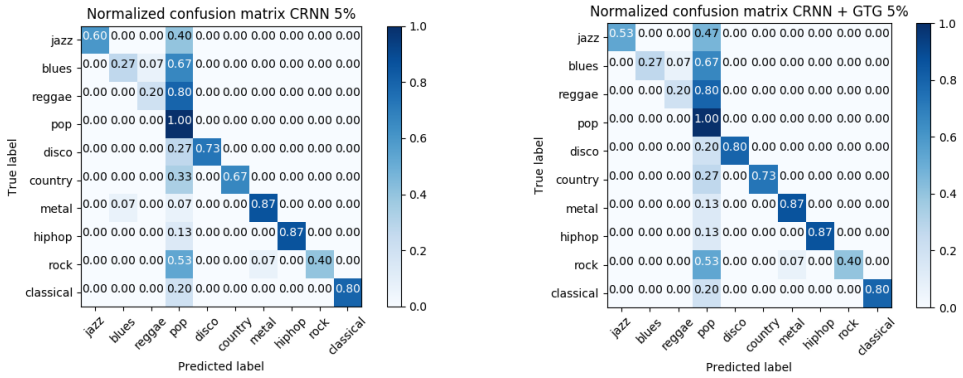
- Reduce the training set in 2%, 5% and 10% size portions
- For each portion of the training set
  - Fine-tune the network to get the features to infer the labels of the rest of the training set with the GTG
  - Apply the GTG, taking the part of training set on which the fine-tuning is done as labeled data and the rest of the training set as the unlabeled data
  - Infer the labels of the rest of the training set from the GTG results and use them instead of the original ones as the new labels together with the ones associated to the percentage used for fine-tuning
  - Make a new fine-tuning with all the so-labeled training set
  - Processing of the training and the test set exporting the produced features before the classification layer of the network
  - Prediction of the training and the test set
  - Build the similarity matrix with the data extracted before the last layer of the network
  - Apply GTG taking the training set with the new labels as labeled data and the test set as unlabeled data
  - Compare the results obtained by GTG with those from the network

5.2.2.3 Discussing the results



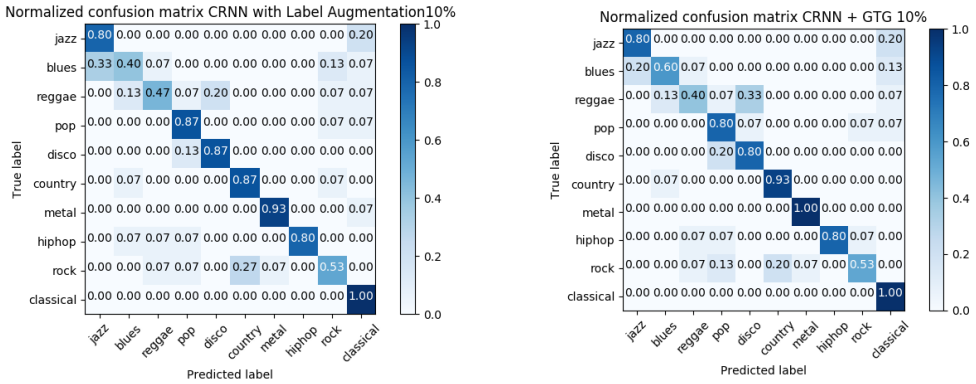
(a) (b)

Figure 5.2.10: CRNN and CRNN + GTG comparison in label augmenting with 2% labeled training set



(a) (b)

Figure 5.2.11: CRNN and CRNN + GTG comparison in label augmenting with 5% labeled training set



(a) (b)

Figure 5.2.12: CRNN and CRNN + GTG comparison in label augmenting with 10% labeled training set

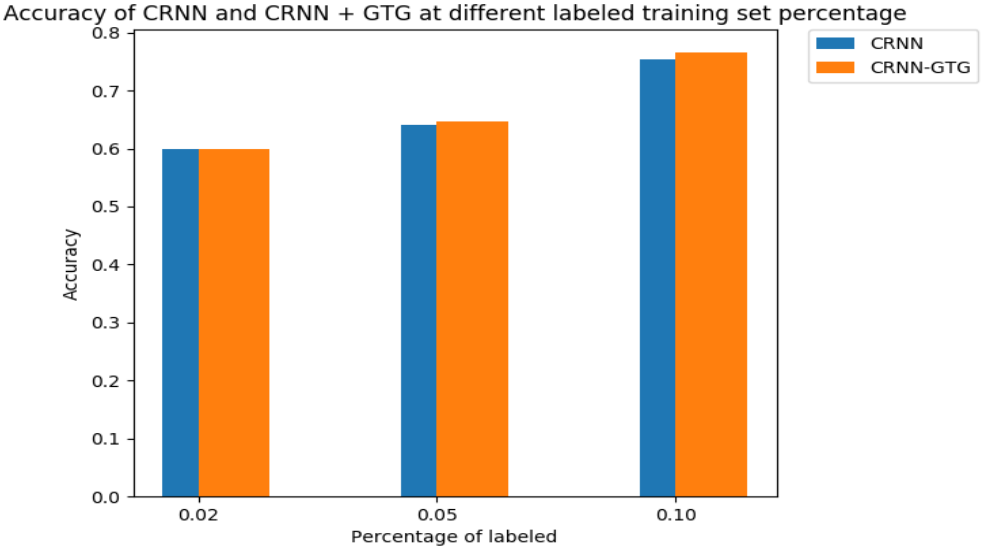


Figure 5.2.13: Accuracy comparison between CRNN and CRNN + GTG at different percentages of labeled training set in label augmenting

	Accuracy		
	2%	5%	10%
CRNN	0.601	0.641	0.754
CRNN + GTG	0.601	0.647	0.766

Table 5.4: Accuracy value comparison between CRNN and CRNN + GTG at different percentages of labeled training set in label augmentation

In the same way of the single label classification experiment we can see how GTG brings an improvement in classification, even if the lack of correctly labeled data on which the system was trained has influenced the final results in terms of accuracy, which is much lower than before. We can observe that the use of few observations leads to critical issues in the recognition of some classes, such as the "reggae" class which in the case of 2% is almost completely classified as "pop". Also in this experiment we can see how on average GTG manages to increase the internal homogeneity of clusters that are formed, concentrating more data on the cluster to which they are expected to belong. For example, the "country" class in the case of 10% is distributed by the CRNN in three classes, while GTG concentrates them in only two with prevalence of the expected class and with fewer classification errors.

6 Conclusions and Future Works

In this thesis we have studied the classification results produced by the composition of two different approach methods: the CRNN proposed by Choi et al. [2], which analyzes elements individually, and the Graph Transduction Game proposed by Pelillo et al. [1], which compares elements on the basis of a similarity measure.

The experiments carried out show that GTG can improve the classification results of the CRNN either in the case of a complete dataset and in the case of a dataset composed of few observations. This leads us to consider the combination of different approach methods, one that classify every single object based on its own features and one that exploits the comparison of features between every objects belonging to each class.

This thesis can be the basis for some future works. A first example is the development of a version of the Graph Transduction Game suitable for multilabel classification. A second example is the use of our proposed model for music genre classification problem considering different approaches from mel spectrograms.

6.1 Multi label classification

The CRNN used in this thesis was originally designed to deal with the multilabel classification problem: given a dataset, each element is associated with a group of labels, each of which refers to a particular characteristic, such as genre, instruments, mood, etc [2]. Referring to the definition of Graph Transduction Game this problem can be seen as a non-cooperative multiplayer game in which each player competes to belong simultaneously to the number of classes that maximize his payoff. A metaphor for multilabel problem would be that of a chess game in which each player can simultaneously make an unfixed number of moves at each turn. GTG has not been designed to deal with this kind of problem because it is based on a game that at each turn it chooses one and only one move, that corresponds to the most convenient one. The future work we propose is to develop a variant of GTG designed to deal with this problem.

6.2 Other music genre classification approaches

Another set of data on which we could apply our model is that which presents information related to the musical structure, also called *musical form*. This kind of structure was extremely important in the western world society especially before the nineteenth century. However the recognition of musical form requires to process the entire piece of music, which can be very onerous in terms of memory and time due to audio file format. A solution could be to use lighter digital representations, such as MusicXML or MIDI, which maintain basic information content in restrained memory.

Bibliography

[1] Aykut Erdem, Marcello Pelillo: Graph Transduction as a Noncooperative Game. Neural Computation 24(3): 700-723 (2012)

[2] Keunwoo Choi, György Fazekas, Mark B. Sandler, Kyunghyun Cho: Convolutional recurrent neural networks for music classification. ICASSP 2017: 2392-2396

[3] Djork-Arné Clevert, Thomas Unterthiner, Sepp Hochreiter: Fast and Accurate Deep Network Learning by Exponential Linear Units (ELUs). CoRR abs/1511.07289 (2015)

[4] Michael A. Nielsen, "Neural Networks and Deep Learning", Determination Press, 2015

[5] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, Yoshua Bengio: Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling. CoRR abs/1412.3555 (2014)

[6] Sebastian Ruder: An overview of gradient descent optimization algorithms. CoRR abs/1609.04747 (2016)

[7] Nitish Srivastava, Geoffrey E. Hinton, Alex Krizhevsky, Ilya Sutskever, Ruslan Salakhutdinov: Dropout: a simple way to prevent neural networks from overfitting. Journal of Machine Learning Research 15(1): 1929-1958 (2014)

[8] C. Weihs, D. Jannach, I. Vatolkin, G. Rudolph, "Music Data Analysis", CRC Press, 2017

[9] Rahul Dey, Fathi M. Salem: Gate-Variants of Gated Recurrent Unit (GRU) Neural Networks. CoRR abs/1701.05923 (2017)

[10] Klaus Greff, Rupesh Kumar Srivastava, Jan Koutník, Bas R. Steunebrink, Jürgen Schmidhuber: LSTM: A Search Space Odyssey. IEEE Trans. Neural Netw. Learning Syst. 28(10): 2222-2232 (2017)

[11] T. Feng, "Deep learning for music genre classification", 2014

[12] J. Irvin, E. Chartock, N. Hollander, “Recurrent Neural Network with Attention for Genre Classification”, 2016

[13] Stanford CS class “CS231n: Convolutional Neural Networks for Visual Recognition”, http://cs231n.github.io

[14] Keunwoo Choi, George Fazekas, Mark B. Sandler: Automatic tagging using deep convolutional neural networks. CoRR abs/1606.00298 (2016)

[15] V. Lombardo, A. Valli, “Audio e Multimedia”, 3rd edition, Apogeo, 2008

[16] Samson, Jim. "Genre". In Grove Music Online. Oxford Music Online. Accessed March 4, 2012.

[17] Gunn, Steve R. "Support vector machines for classification and regression." ISIS technical report 14.1 (1998): 5-16.

[18] Carlos Nascimento Silla Jr., Alessandro L. Koerich, Celso A. A. Kaestner: A Machine Learning Approach to Automatic Music Genre Classification. J. Braz. Comp. Soc. 14(3): 7-18 (2008).

[19] Liang, Dawen, Haijie Gu, and Brendan O’Connor. "Music genre classification with the million song dataset." Machine Learning Department, CMU (2011)

[20] Haggblade, Michael, Yang Hong, and Kenny Kao. "Music genre classification." Department of Computer Science, Stanford University (2011)

[21] Goulart, Antonio Jose Homsi, Rodrigo Capobianco Guido, and Carlos Dias Maciel. "Exploring different approaches for music genre classification." Egyptian Informatics Journal 13.2 (2012): 59-63

[22] Roberto Basili, Alfredo Serafini, Armando Stellato: Classification of musical genre: a machine learning approach. ISMIR 2004

[23] Clark, Sam, Danny Park, and Adrien Guerard. "Music genre classification using machine learning techniques." (2012).

[24] Chris Sanden, John Z. Zhang: Enhancing multi-label music genre classification through ensemble techniques. SIGIR 2011: 705-714

[25] Pérez-García, Tomás, Carlos Pérez-Sancho, and José M. Iñesta. "Harmonic and instrumental information fusion for musical genre classification." Proceedings of 3rd international workshop on Machine learning and music. ACM, 2010.

[26] Shubhanshu Gupta: Music Data Analysis: A State-of-the-art Survey. CoRR abs/1411.5014 (2014)

[27] Nasridinov, Aziz, and Young-Ho Park. "A study on music genre recognition and classification techniques." International Journal of Multimedia and Ubiquitous Engineering 9.4 (2014): 31-42

[28] Viswanathan, A., and Sriram Sundaraj. "Music Genre Classification." International Journal Of Engineering And Computer Science 4.10 (2015): 14848-14849.

[29] Creme, Matthew, Charles Burlin, and Raphael Lenain. "Music Genre Classification." (2016).

[30] Aldona Rosner, Felix Weninger, Björn W. Schuller, Marcin Michalak, Bozena Kostek: Influence of Low-Level Features Extracted from Rhythmic and Harmonic Sections on Music Genre Classification. ICMMI 2013: 467-473

[31] L. P. Coelho, W. Richert, "Building Machine Learning Systems with Python", Packt Publishing, 2015

[32] Soujanya Poria, Alexander F. Gelbukh, Amir Hussain, Sivaji Bandyopadhyay, Newton Howard: Music Genre Classification: A Semi-supervised Approach. MCPR 2013: 254-263

[33] Nima Tajbakhsh, Jae Y. Shin, Suryakanth R. Gurudu, R. Todd Hurst, Christopher B. Kendall, Michael B. Gotway, Jianming Liang: Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning? IEEE Trans. Med. Imaging 35(5): 1299-1312 (2016)

[34] Edith Law, Kris West, Michael I. Mandel, Mert Bay, J. Stephen Downie: Evaluation of Algorithms Using Games: The Case of Music Tagging. ISMIR 2009: 387-392

[35] George Tzanetakis, Perry R. Cook: Musical genre classification of audio signals. IEEE Trans. Speech and Audio Processing 10(5): 293-302 (2002)