Hit Song Prediction system based on audio and lyrics embeddings

Tesi di Laurea Magistrale in
Music and Acoustic Engineering

Author: **Elisa Castelli**

Student ID: 969454
Advisor: Prof. Massimiliano Zanoni
Academic Year: 2022-23

Abstract

Nowadays, thanks to web platforms, a great amount of new songs are released every day. Hit Song Prediction (HSP) is a field of Music Information Retrieval that aims to investigate whether a song has the potential to become popular or not, in order to help talent scouts, labels and producers to make a preliminary automatic selection of songs that can be appealing in an artistic or in a market perspective.
After examining the current models and techniques employed in HSP, we explore the areas where there is room for improvement. Based on these insights, we outline the decisions that guided the development of our model. The system proposed has the novelty of employing, for the first time in this field, a multi-modal approach based on audio and lyrics embeddings. In detail, when provided with audio, lyrics, and the song's release year, our model generates a popularity score or classifies the song accordingly. In order to do this it involves a Multi-Layer Perceptron that takes as input features the concatenation of three data: audio embedding extracted from the audio melspectrogram using a ResNet-50, lyrics embedding computed by a Sentence-BERT transformer and release year.
In order to evaluate the effective applicability of our method we test it in three tasks. Before doing this, we create two new version of the SpotGenTrack Popularity dataset, after having performed some cleaning operations: an English and a multi-lingual dataset. The first test conducted investigates the impact of using also text embeddings instead of using only audio embeddings for a classification problem. The other tests conducted aim to compare the performance of our solution with the state-of-the-art systems. Results evidence how the lyrics contribution has a key-role in HSP. Moreover, the overall results obtained are comparable with the ones achieved by the reference papers. This demonstrates that our proposed system is a valid solution for tackling HSP. In particular multi-lingual setup outperforms the English-only experiments, underlying the importance of having a significant amount of songs at disposal to model the complexity of HSP problem.

Keywords: Hit Song Prediction, Deep Learning, Audio Embedding, Lyrics, Multilingual Embedding, Sentence-BERT

Abstract in lingua italiana

Grazie alle piattaforme web una grande quantità di nuove canzoni vengono rilasciate ogni giorno. Hit Song Prediction (HSP) è un campo di Music Information Retrieval che ha lo scopo di indagare se una canzone ha il potenziale per diventare popolare o meno, al fine di aiutare talent scout e produttori a fare una prima selezione automatica di canzoni che possono risultare accattivanti, in una prospettiva artistica o di mercato.
Dopo aver studiato modelli e tecniche attualmente utilizzati in HSP, partendo dagli aspetti su cui esiste un margine di miglioramento, si descrivono le scelte che ci hanno portato a progettare il nostro modello. Con l'architettura proposta vogliamo impiegare, per la prima volta in HSP, un approccio multi-modale basato su embeddings audio e di testo.
Il sistema proposto prende in input audio, testo e anno di uscita di una canzone per produrre come risultato il punteggio o la classe di popolarità a cui la canzone appartiene. Per fare questo, si utilizza un Multi-Layer Perceptron che riceve come feature di ingresso la concatenazione di tre dati: l'audio embedding estratto dal melspectrogram utilizzando una rete Resnet-50, l'embedding calcolato da un transformer Sentence-BERT a partire dai lyrics e l'anno di uscita. Prima di condurre gli esperimenti, due nuove versioni del dataset SpotGenTrack Popularity sono state create: una inglese e una multilingua.
Per valutare l'effettiva applicabilità del nostro metodo nell'HSP lo sottoponiamo a tre test. Il primo esamina l'impatto dell'utilizzo degli embeddings testuali confrontandolo con l'utilizzo di soli embeddings audio, per svolgere un problema di classificazione. Gli altri invece mirano a confrontare le prestazioni della nostra soluzione con i sistemi dello stato dell'arte. I risultati ottenuti dimostrano che il contributo del testo ha un ruolo chiave in HSP. Inoltre, le performance complessive del nostro sistema risultano comparabili con quelle ottenute dai modelli di riferimento, portandoci ad affermare l'effettiva applicabilità del metodo proposto. In particolare, il dataset multilingua porta a risultati migliori rispetto al dataset inglese, sottolineando l'importanza di avere una quantità significativa di dati per modellare la complessità del problema di HSP.

Parole chiave: Predizione di Canzoni Hit, Deep Learning, Audio Embedding, Testi, Embedding Multilingua, Sentence-BERT

Contents

1 | Introduction

Music industry market is drastically changed with the diffusion of digital audio formats and the growth of streaming platforms. Technologies used in these last 15 years, to manage and analyze songs data have evolved. In particular, the development of machine learning has given rise to a new field: Music Information Retrieval (MIR). MIR is an interdisciplinary field that employs computational techniques to retrieve musical information, analyzing them with the aim of solving tasks like music classification, recommendation, music source separation, automatic music transcription and many others.
Among all these applications, there is also a field of studies, called Hit Song Science that aims to investigate whether a song has the potential to become popular or not.
Producing an artist or deciding to invest in marketing campaign for a song requires to consider several factors to understand the risks and the possible results in terms of popularity and profits.
Revenues in music industry are mainly obtained from live music and recorded music. Both of them are influenced by different factors. Especially during the pandemic, because of the restrictions, many live performances were canceled and live music revenues decreased drastically while, on the other hand, streaming incomes boosted.
Social Networks and music streaming platforms, for example Spotify and TikTok, have increased the popularity. They have contributed to music revenues so much that digital media and music streaming become the majority source of profit for music industry.
In particular, popularity of a song on digital platforms and streaming applications has become a measure of success and consequently of the incomes a song may generate.
Everyone can understand the genres of songs they appreciate, but it's not clear precisely why a certain song is more appealing or popular. Considering this scenario, labels and producers are looking for ways to answer questions like "Which are the characteristics of songs that make them popular? Perhaps the lyrics speak to an experience? Perhaps the energy makes it appealing?". In this scenario technology innovation might play an important role.
Hit Song Science (HSS) is a term invented by Mike McCready [1], an American entrepreneur in the music industry, that became famous for having been the pioneer of the

topic of hit song prediction using acoustic analysis software to gather and study the underlying patterns in music.
Nowadays, HSS is an hot and active research topic in Music Information Retrieval (MIR). It's important to precise that HSS's main goal is not to substitute talent scouts but, given the great amount of new songs released every day, thanks to web platforms and streaming applications, it can be a useful tool to make a preliminary automatic selection of songs that can be appealing from an artistic or a market perspective.
In this context, machine learning algorithms and deep learning techniques gain a key role thanks to their capabilities in automatic capturing information from audio signals and lyrics texts that are fundamental to search and identify underlying patterns in songs.
Researches conducted in these last years propose Hit Song Prediction systems that employ mainly machine learning methods to address this challenge, only in few cases deep learning techniques. Even in the case of deep learning approaches, the systems proposed are based on Multi-Layer Perceptron structures applied on songs metadata and sets of manually hand-crafted features computed from audio and lyrics. Starting from these observations, encouraged by the advancements in deep learning technologies able to automatically extract features from images and texts, in this work we will study the embedding-based methods applied to Music Information Retrieval. After stating that embeddings lead to state-of-the-art systems not only in computer vision field, but also in audio-related tasks, we will conduct an investigation of the applicability of these embedding-based solutions in the field of Hit Song Prediction.
This thesis aims to introduce a novel approach to address HSP. Going into detail, we will explore the use of CNN-based methods to automatically extract features from raw audio files and the usage of transformer-based systems to extract text embeddings from lyrics. According to this, we will not take advantage of other information, such as the artist's popularity, the markets in which the songs are available that can bias the resulting popularity predicted.
We will develop a neural network architecture that takes as input for each song the relative audio, lyrics and release year to temporally contextualize songs. Starting from these three rough data, the system architecture involves three main processing parts to achieve the final prediction result. An audio processing chain, built with a ResNet-50 [2] model, produces the audio embeddings taking advantage of its convolutional layers applied to the audio mel-spectrogram. A text processing chain, based on a Sentence-BERT [3] transformer-model, produces the lyrics embeddings. These two contributions, together with the song release year, are then processed as feature input vector by a Multi-Layer Perceptron that produces the popularity prediction.
The popularity value used as target by this model is the popularity score assigned by

Spotify. This score is available through the Spotify API and it is a parameter computed considering the number of plays a track has on Spotify and how recent those plays are. In order to evaluate the model performance, under a classification perspective, we define four classes matching four Spotify popularity ranges.

The model proposed is then tested comparing performances when using only audio embeddings and considering both audio and text embeddings, to investigate the impact of text in HSP. Moreover, other tests are conducted to analyze how the system performance changes, both in classification and regression tasks, with two kinds of datasets: English-only and multi-lingual.

The thesis is organized as follows. In *Chapter 2*, theoretical concepts necessary to the understanding of the thesis. In *Chapter 3*, the state-of-the-art in Hit Song Science, and in particular in music popularity prediction, is presented, we will give an overview of the main techniques used in the literature leading the state-of-the-art at the actual point. Moreover an overview of audio and text embedding networks is provided with the aim of investigating cases of applicability in the MIR field.

Chapter 4 introduces the method we design to address the problem of Music Popularity Prediction.

Chapter 5 illustrates the experimental setup used to evaluate the performance of the solution proposed. First of all the process of training setup is explained, with particular attention to the dataset building phase. Successively an introduction of the evaluation strategies used is presented and finally the results obtained, with the relative discussion, are reported. Lastly in *Chapter 6* conclusions, possible improvements and future works are summarized.

2 | Theoretical Background

In this chapter, we will introduce theoretical concepts related to the thesis work developed. All the concepts explained here are the bricks for Hit Song Prediction applications, the design of the solution proposed, and its implementation.
The chapter is organized in two subsections with the aim of dividing the topics in two main fields: audio features extraction and deep learning methods.

2.1. Sound Representation

Audio music signals have to be pre-processed to extract meaningful descriptors to be fed into Machine Learning or Deep Learning models. Features are fundamental to represent music signals so that machines can interpret them.
Generally, music audio signals can be represented in two domains: time and frequency. Consequently, also features can be computed in these two domains depending on if they are calculated starting from the sound wave or from the audio signal spectrogram. Both of these type of features have to be considered because of their contribute in identifying different relevant aspects of the signal.
In order to compute features in the frequency domain, the frequency signal content has to be computed through the application of the Fourier Transform (FT) [4]. Fourier Transform is a mathematical function that converts a signal from the time domain into a form that describes the frequencies that compose it. A variation on FT in the discrete domain consists in the Discrete Fourier Transform (DFT) [4]. The DFT operation is similar to Fourier Transform but it takes as input a sampled signal, instead of a continuous signal. It produces, as output, a collection of coefficients of a complex sinusoidal linear combination, ordered by frequency.
The fact of having as input a discrete signal makes DFT optimal for processing digital audio signals, that consist in a sequence of quantized samples obtained from sampling an analogue audio signal.

2.1.1. Short Time Fourier Transform

In music frequency content changes over time, therefore signals have to be considered simultaneously in frequency and time domain, to be analyzed in a more complete way. This is possible through the use of Short-Time Fourier Transform (STFT) [5].
The Short-Time Fourier Transform representation of a signal is obtained by dividing it into shorter overlapping segments, of equal length, multiplying them by a sliding window and then stacking the Discrete Fourier Transform (DFT) computed on each of them.
In this way, applying the Discrete Fourier Transform on consecutive windows of the signal, it's possible to observe how the frequency components evolve over time providing insights into the dynamics of the signal. An example of the procedure to compute Short Time Fourier Transform can be seen in Figure 2.1.

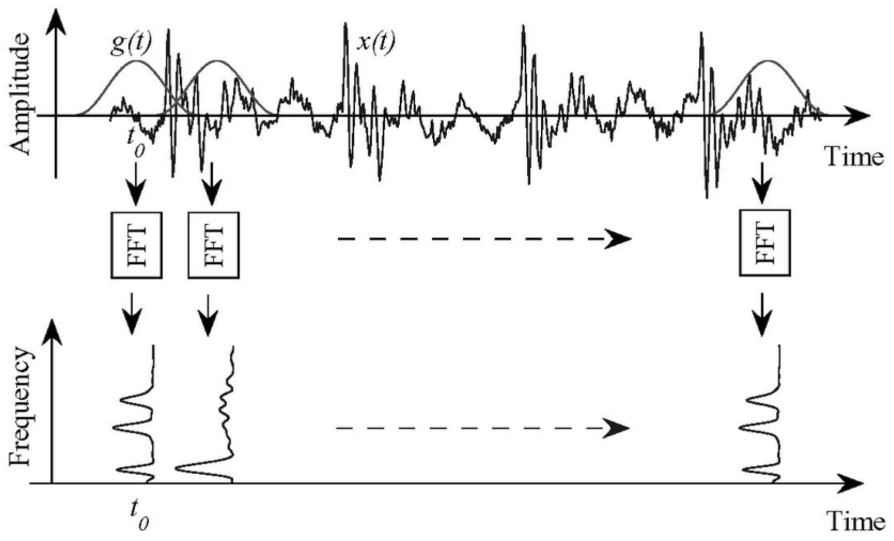


Figure 2.1: Illustration of Short Time Fourier Transform computation

Going into details, multiplying a chunk of the original signal by a bell-shaped window $w[n]$, with a certain hop-size H between windows, the windowed frame $x_w[n, m]$ obtained can be described by the equation:

$$x_w[m] = \sum_{n=0}^{N-1} w[n] \cdot w[n + mH] \tag{2.1}$$

with N the size of the frame and m the number of frame. DFT of each signal frame can be obtained as:

$$X[m, k] = \sum_{n=0}^{N-1} x_w[m] \cdot e^{-j2\pi nk/N}, \tag{2.2}$$

in which N is the size of the frame. The variable m denotes the number of the frame considered along the time axis, while k is the frequency index, and is sometimes referred to as a frequency bin.
Finally, by stacking the DFTs corresponding to each frame in the temporal dimension, a 2D representation is obtained. Usually, the spectrogram is returned by the modulus (magnitude spectrogram) or squared magnitude (power spectrogram) of the STFT. The size of window and hop influences the spectral and temporal resolution of the STFT, as shown in Figure 2.2.

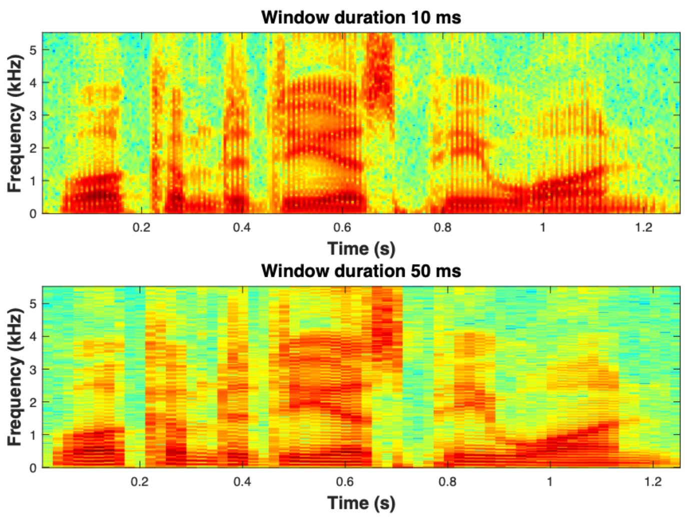


Figure 2.2: Impact of window duration on the STFT. A spectrogram is shown, using Hamming windows of duration 10 ms (upper figure) and 50 ms (lower figure).

2.1.2. Mel Spectrogram

Short-Time Fourier Transform works over a linear-frequency scale, but human perception of sounds shows a logarithmic behavior. Moreover it doesn't take into consideration pitch sensitivity and varying frequency resolution, making it less suited to represent how we perceive sound.
In fact, frequency resolution of the human ear varies along the frequency axis. In audiology and psycho-acoustics [6], the concept of critical bands was introduced by Harvey Fletcher in 1933 and refined in 1940. Critical bands are designed to approximate the bandwidth of auditory filters in the cochlea. Two tones within the same band will interfere with each other leading to the masking phenomenon.
Following this assumptions, other representations of the frequency content of signals have been considered, paying attention to the perceptual descriptor. In this way, more high-level information about the signal analyzed could be obtained.

Mel scale is a perceptual scale of frequencies that approximate how human perceive sounds. There exist several analytical expressions, but a common relation between the mel scale $mel(f)$ and the Hertz scale f was given by Fant in 1968 [7]:

$$mel(f) = \frac{1000}{\log 2} \cdot \log(1 + \frac{f}{1000}). \tag{2.3}$$

Mel scale represents the fact that ears perceive in a more sensitive way changes lower frequencies while they are less sensitive to small changes at higher frequencies.
The mel-spectrogram is a representation of the STFT where frequencies scale changes, according to the mel-scale, to better align with human perception of sound.
The mel-spectrogram is obtained by computing the STFT and then applying the mel-filterbank. Mel-filterbank is a set of triangular filters spaced evenly on the mel scale so that each filter covers a certain frequency range.
After multiplying the mel-filterbank to the STFT, the logarithm is applied to obtain a more perceptually relevant representation. An example of mel-spectrogram obtained from the spectrogram of a given waveform is depicted in Figure 2.3.

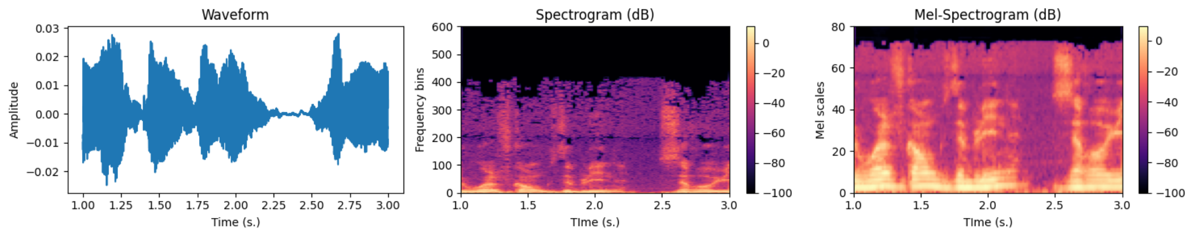


Figure 2.3: Example of waveform and relative spectrogram and mel-spectrogram

The resulting mel-spectrogram is a 2D representation where the x-axis represents time, the y-axis represents mel-frequency, and the intensity at each point represents the magnitude or energy of the corresponding frequency component at that time.
The mel-spectrogram has been widely used and found to be effective in various audio processing tasks, such as speech recognition, music classification, and sound event detection. By focusing on perceptually relevant frequency information, the mel-spectrogram helps to capture the characteristics that are important for these tasks and can lead to better performance than using a spectrogram obtained using the STFT.

2.2. Deep Learning

Deep learning focuses on algorithms that take inspiration from human brain and that are structured according with its functioning. Deep learning algorithms are known as Artificial

Neural Networks (ANN) and consist of a collection of interconnected basic units, called neurons, that can "fire" based on the inputs received, from previous neurons, and can send output signals to subsequent neurons. Each neuron can be connected with one or more other neurons and each connection has a different importance defined by a weight. These connections are organized in layers and they allow the algorithm to exchange information with the purpose of classifying information, clustering data or predicting outcomes.
During the years, neural networks have become an evolution of machine learning algorithms due to their ability to learn the mapping between an input and a desired output by capturing a-priori unknown information hidden in the data.
In fact, Deep Learning models can recognize complex patterns in raw input such as images, texts, sounds and other data to produce accurate information and predictions.
This knowledge is acquired trough a training step. The training process consists of finding the combination of weights of each connection in order to achieve the desired result of having as output the right prediction.
There are two kinds of methods for learning techniques: supervised and unsupervised learning. The latter technique is characterized by the unavailability of the target desired in output, given a certain input sample. In this case, the training process of the model consists of taking some inputs and trying to reach a stability, building clusters between similar data.
In supervised learning instead, every sample is provided with a label that identifies the expected output or target. The training process of the model in this case consists of tuning the weights of the connections between neurons in a proper way to estimate correctly the well-known target, given a certain data in input.

2.2.1. Multi-Layer Perceptron

The basic organization structure of an ANN is called **perceptron** and it consists of a single layer neural network composed by a single neuron, as it is shown in Figure 2.4. In fact, it is a simplified model of a neuron that takes in input a signal $X = x_1, x_2, \dots, x_n$ with its assigned weight $W = w_1, w_2, \dots, w_n$, with $n \in N$ corresponding to the dimension of the input. The perceptron calculates the weighted sum of the inputs, adds a bias scalar value b, and applies an *activation function* ϕ to obtain the output y. The complete equation is

$$y = \phi(\sum_{j=1}^{n} x_j w_j + b_j). \tag{2.4}$$

In general, the activation function is a mathematical function applied to the output of a neuron or a layer of neurons. In particular, the perceptron algorithm, introduced by Frank

Rosenblatt in the late 1950s [8], is specifically designed for binary classification tasks. For this reason, it uses a linear activation function, such as the sign function, which allows the perceptron to make binary decisions based on whether the weighted sum of inputs exceeds a certain threshold.

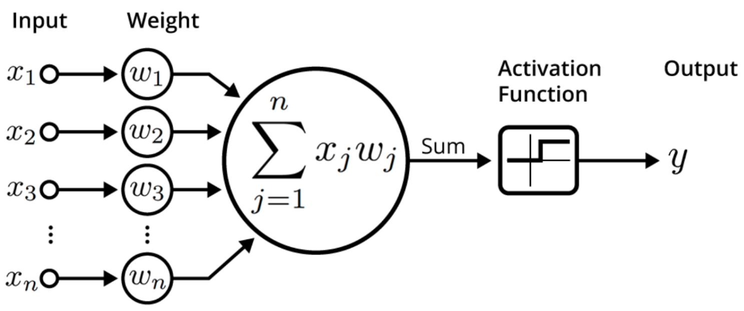


Figure 2.4: Structure of a Perceptron

The goal of a perceptron is to learn automatically weights and bias parameters during the training process, to achieve the desired values and the correct output. Going into details, the training process consists in a first forward pass step, in which labelled training data are pass through the perceptron and a prediction is obtained according to the Equation 2.4. Successively, after having compared prediction and target label, a phase of weights updating is required, following this criteria:

- Increase the weights if the predicted output is too low for a positive instance
- Decrease the weights if the predicted output is too high for a negative instance

This process will be repeated in an iterative way until a convergence criteria is met, such as reaching a maximum number of iterations or when the perceptron achieves satisfactory accuracy in prediction.
Multiple perceptrons can be interconnected in a structured network, according to a topology, creating a Neural Network, also called **Multi-Layer Perceptron** (MLP), that overcomes the limitations of perceptron in handling non-linear data. The structure of MLP is illustrated in Figure 2.5. It consists in layers - set of neurons equally distant from the input neuron - of three main types: input, hidden and output layers.

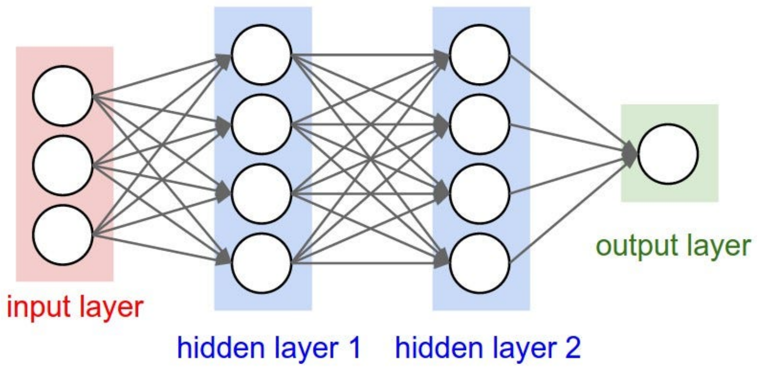


Figure 2.5: Structure of a Multi-Layer Perceptron

Each neuron, belonging to each layer in an MLP, applies an activation function to the weighted sum of its inputs. Activation function introduces non-linearity to the network, allowing it to learn complex patterns and make more sophisticated predictions. There are several activation functions commonly used in neural networks, here some commonly used ones:

- Linear
$$f(x) = x \tag{2.5}$$

- Sigmoid Function
$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.6}$$

- Rectified Linear Unit (ReLU)

$$f(x) = max(0, x) \tag{2.7}$$

- Hyperbolic Tangent (Tanh)
$$f(x) = \frac{e^x - e^{-x}}{e^x + e{-}x} \tag{2.8}$$

Also in MLP the training process can be divided in two main steps: forward and backward-propagation. The first one consists in taking labelled-training data, making them propagate across the neural network until the last layer in which the final prediction is computed. Once the prediction is obtained, for each data given as input, the error committed is calculated using a *loss function* that measures the discrepancy of the prediction result in relation to the correct label. This loss value is then backward propagated, computing gradient of the loss function with respect to the network's parameters (weights and bi-

ases), enabling iterative updates that progressively improve the model's performance. This is typically done using an optimization algorithm like stochastic gradient descent (SGD) or one of its variants, such as Adam, according to a certain learning rate, which determines the step size of the updates. The goal of the training process is to minimize as much as possible the loss function, so that the prediction produced is as much precise as possible. For this reason, the training process terminates when a convergence criteria or a certain accuracy level is reached.
During the training process, it can happen that the neural network suffers of overfitting, this means that the model starts to perform too well on the training data learning also noise, but fails to generalize to new unseen data. To avoid overfitting there are several techniques:

- **Increase training data**: collecting more training data can help the model learn a better representation of the underlying patterns in the data. Exposing the model to a wider range of examples makes it less weak on noise or outliers.
- **Regularization**: technique that add a penalty term to the loss function, discouraging the model from assigning excessive importance to individual features or making overly complex decisions.
- **Dropout**: a regularization technique that consists in dropping out and ignoring a group of nodes of the neural network chosen randomly with a certain probability. The set of ignored neurons changes at each forward-backward interaction and doing so the main advantage is that the network becomes more robust avoiding co-adaptation between neurons that will lead to generalization of the unseen data.
- **Data Augmentation**: is a method to increase training data from the ones already had. It consists of generating additional training data by applying transformations, distortions, or perturbations to the existing data. An example applied to audio spectrograms can be seen in Figure 2.6.
- **Simplify the model**: to decrease the complexity of a model and consequently improve its capacity of generalizing on unseen data, one of the simplest solution is decrease its own complexity removing layers or simply reducing the number of neurons.
- **Early stopping**: a technique that keeps track of the validation loss and if it observes a stop in the loss decreasing for several epochs in a row it forces the training stopping. The number of how many epochs we want to wait after the last time the validation loss improved before breaking the training loop is set according to the patience

parameter.

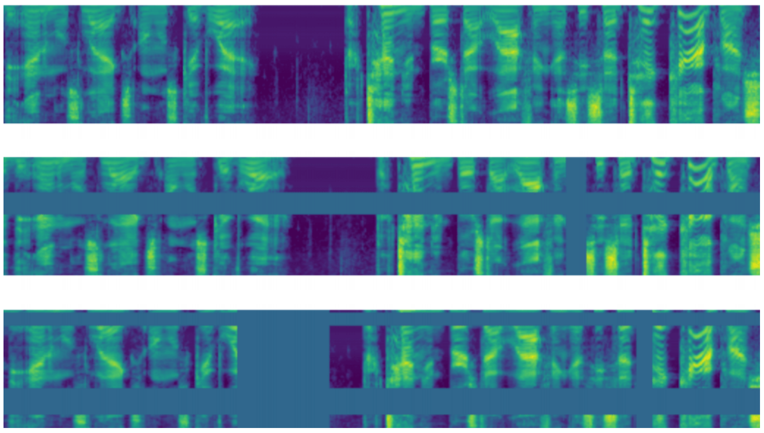


Figure 2.6: Example of data augmentation performed on an audio spectrogram following the SpecAugment[9] technique

2.2.2. Convolutional Neural Network

By utilizing deep architectures, researchers have achieved remarkable success and state-of-the-art performances in many applications in domains like computer vision, natural language processing and speech recognition. In particular, these results have been achieved taking advantage of some categories of neural networks such as Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs).

A Convolutional Neural Network (CNN) is a type of sparse connected neural network widely used for various computer vision tasks, such as image classification and object detection, due to its capability of capturing local features and patterns in an image.

In particular, CNNs are able to learn features hierarchically, starting with low-level features (e.g. edges) in early layers, going on to more complex and abstract features in deeper layers, until the last layers in which classification is performed. Moreover, CNNs are also robust to small spatial translations, rotations and distortions. For this reason, this kind of Neural Network is well-suited for tasks where the position of objects or features may vary.

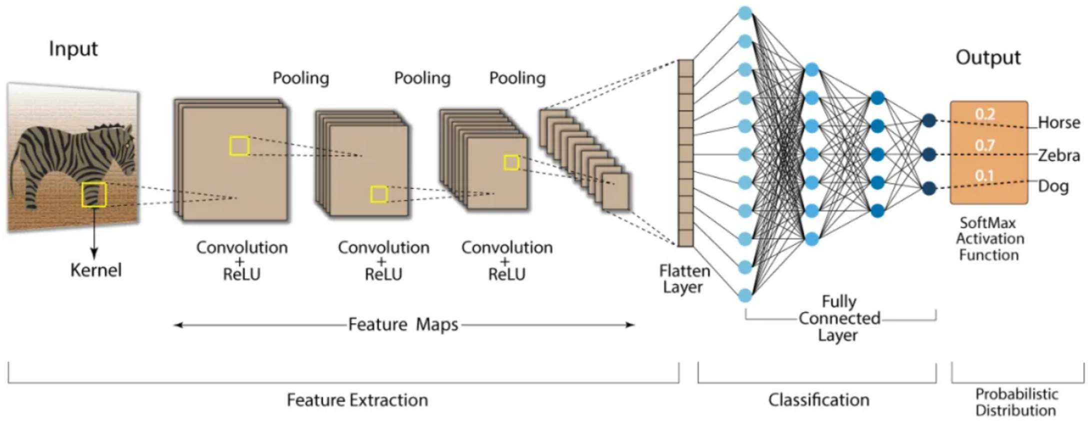


Figure 2.7: Example of CNN

As can be seen in Figure 2.7, the set of layers that constitute the structure of a CNN is the following:

- **Input layer**: the first processing unit that receives the input data. Its dimensions correspond to the dimensions of the input, hence height, width and the number of channels.
- **Convolutional layers**: a set of several layers that are designed to capture and process data efficiently, like human visual cortex cells that are sensitive to small receptive field. They apply a set of filters, also known as *kernels*, to the input data. These filters slide across the input data using a specified stride value and they perform the convolution operation at each location.
  The convolution is computed following this formula:

$$y(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m)(j-n) \tag{2.9}$$

  where $y(i,j)$ is the feature map, I and K are respectively the input and the filter matrices. As the filter slides across the input, it generates a matrix known as **feature map**. Each element in the feature map represents the activation value of the filter at a specific spatial location of the input. The number of filters used in a convolutional layer determines the number of feature maps produced. Each filter focuses on capturing different patterns or features, such as edges, corners or textures.
  Performing a convolution, the main parameters that have to be set are the filter size, the depth, that indicates the number of filters used in the layer, and the horizontal and vertical stride, that represent how far the filter moves from one position to the

following one. Lastly, after each convolution layer, an activation function is applied.

- **Pooling layer**: it is a layer used to downsample feature maps, reducing their dimensions while keeping the most important information. The most common pooling operations are *max pooling*, in which the maximum value is retained inside each sliding window, and *average pooling* that takes the average value inside a sliding window. Pooling helps the model to be more efficient, also reducing overfitting.
- **Fully Connected Layers**: they are typically placed at the end of the CNN architecture to extract the output. These layers connect every neuron from the previous layer to the ones in the subsequent layer. By doing this, they capture global patterns and relationships among the learned features allowing the network to make predictions. The number of neurons in the last fully connected layer corresponds to the number of classes in a classification task or the number of regression outputs in a regression task.

CNNs are particularly suitable for transfer learning. Pre-trained CNN models, trained on large-scale datasets, like ImageNet [10], have learned general features that can be transferred to new, smaller datasets or related tasks. By leveraging pre-trained models as a starting point, CNNs can accelerate training and achieve good performance even with limited training data.
In conclusion, CNNs have become widely used due to their ability to extract high-level information from images, in order to acquire the ability to both classify and generate them. The CNNs application in the audio domain arises from the intuition to apply them to 2D representation of sounds in time-frequency domains, such as spectrograms.

2.2.3. Transformers

Transformers are a type of neural network architecture developed to solve the problem of sequence-to-sequence modelling, that finds a lot of applications such as speech recognition, chat bots, language translation, text summarization, music generation, image captioning and many others. All these problems have in common the need of some sort of memory. For example, if we consider translating sentences, a neural network model needs to figure out connections between words, hence it must remember them.
Sequence-to-sequence modelling in general is composed of two steps: encoding and decoding processes. The *encoder* builds a representation of the source and gives it to the decoder. The *decoder* takes the source representation to create the target sequence. Different neural network configurations have been used to implement Sequence-to-Sequence models.

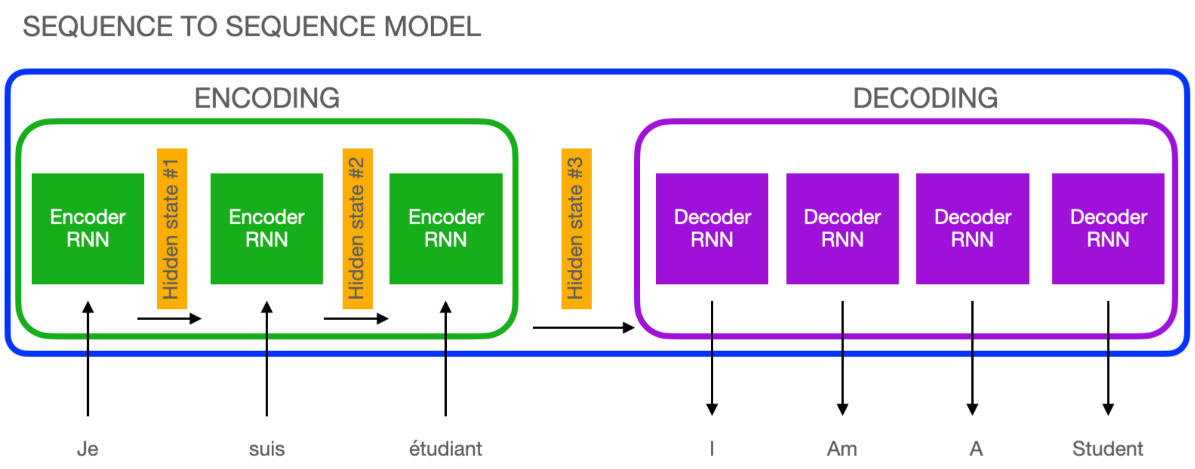


Figure 2.8: Sequence to sequence model functioning - RNN

Initially, Recurrent Neural Networks (RNNs) have been used to deal with the memory problem, because of their naturally constitution by loops, that allow information to persist. However, the problem with RNNs is that information is passed at each step and the longer the chain becomes, the more probable the information gets lost. As can be seen in Figure 2.8, with RNN only one hidden state is passed to the decoder, while the single words hidden states are propagated only inside the following hidden states. For this reason, RNNs have become very ineffective when the gap becomes very large between the more relevant information and the point where this information is needed to give a global sense to the context.
Long Short-Term Memory (LSTM), a special type of RNN, has been introduced to solve this kind of problem, but even with this innovation the probability of extract context, given by a word that is far away from the current word being processed, decreases exponentially with the distance.
Finally, to solve this issue, researchers have created a technique for paying attention to each specific word, driven by the idea that there might be relevant information in every word in a sentence. For this reason, in order for the decoding to be precise, it needs to take into account every word of the input, using a technique called **Attention**.

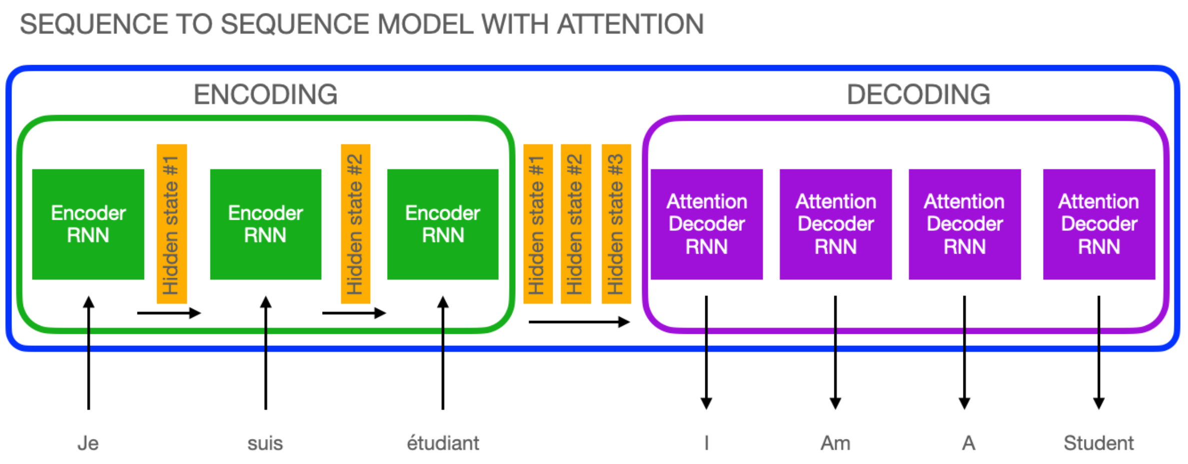


Figure 2.9: Sequence to sequence model functioning with Attention

As reported in Figure 2.9, in sequence-to-sequence modelling with Attention mechanism, instead of passing only one hidden state to the decoder as done previously, all the hidden states generated by every word of the sentence are passed to the decoding stage. In this way, each hidden state is used to figure out where the network should pay attention to during the decoding phase.

2.3. Natural Language Processing

Natural Language Processing (NLP) [11, 12] is a branch of AI that allows computers to read, generate, understand and derive meaning from human language.
NLP combines linguistics and grammar-based rules with computer science, in particular machine learning and deep learning models, to comprehend, break down and separate significant details from text and speech, even in real time. In fact, some NLP applications are speech recognition, word sense disambiguation, sentiment analysis and natural language generation. Several studies [13, 14] in computer vision field have demonstrated the utility of jointly considering media content with natural language captions and this is valid for music and lyrics as well. In fact, lyrics have been shown to be effective in predicting emotions and they can be used to perform genre and mood classification of songs [15–19].
Going into details, to reach this goal, NLP has to understand the sintax and, above all, the semantic of a text to overcome the ambiguity of the human natural language. Regarding sintax, NLP follows a series of passages to analyze texts:

- **Segmentation** : operation that breaks down documents into sentences.

- **Tokenization** : operation that breaks down sentences into tokens, each one corresponding to words or even characters and punctuation, depending on the level of granularity required for the task at hand.
- **Identification of stop words**: step that identifies and eventually removes stop words such as articles, to reduce noise in the data because these words do not carry much meaning.
- **Speech tagging** : operation that assigns grammatical label (noun, verb, adjective, etc.) to each word in a sentence.
- **Name entity recognition** : step that identifies and classifies named entities (people, organizations, locations, dates etc).
- **Stemming** : operation that reduces words to their base root form by removing prefixes and suffixes, often leading to incorrect meanings and spelling.
- **Lemmatization** : reduces words to their meaningful base or root form (lemmas), considering the context and part of speech.

Regarding the semantic of a text, it can be learned by NLP applying algorithms to understand the meaning and structure of sentences, for example with *Word Sense Disambiguation* technique that derives the meaning of a word based on context.
The most fundamental step for Natural Language Processing tasks is to convert words in a way that machines can understand them and also decode patterns within a language. This step of converting words in numbers or vectors of numbers, so that a computer can handle them, is called *text representation* and can be performed in two ways: in a discrete or in a distributed way.
For what concern the discrete text representation, two traditional way for representing texts are *one-hot encoding* and **Bag-of-Words** techniques. They consist in creating a dictionary of words from the ones used inside a text and counting the instances of each word, discarding any information about the order of words in the document. This approach is then used considering each word count as a feature and concluding that documents are similar if they contain similar content.
These methods suffer from many problems: size of vocabulary, sparsity with all its consequences of expensive computation and, above all, the assumption that each word is unique and independent of each others, neglecting context and meaning. Doing this, it's not possible to understand which words are similar, because all vectors are orthogonal to each other: the inner product of any pair of vectors is zero and their similarities cannot be measure by distance nor cosine-similarity.

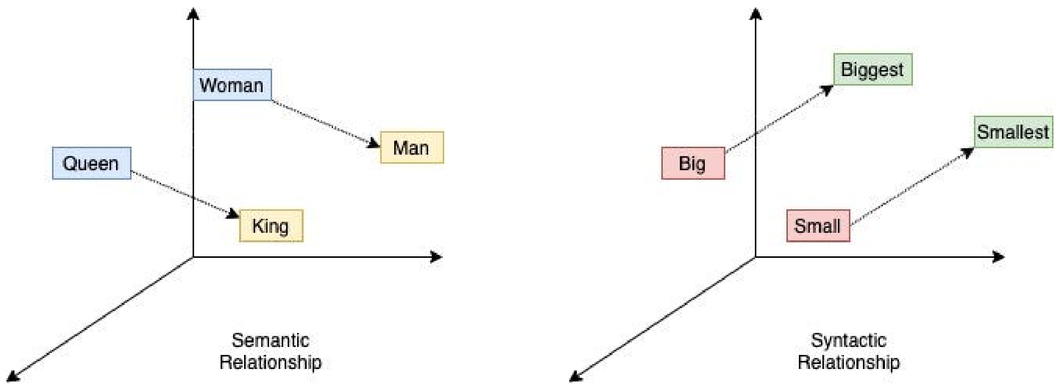


Figure 2.10: Embedding space of Word2Vec

Distributed ways to perform text representation allow the representation of each word as a vector, while capturing its semantic and syntactic relation with other words. This way of representing words as vectors, taking advantage of their position to understand their meaning, is one of the key turning points for the recent great performance improvement of deep learning methods in NLP problems. The main concept introduced with this knowledge is **word embedding**: a learned representation of text according to which words that have the same meaning have a similar and closer representation. In fact, the Figure 2.10 shows how words with a similar meaning are closer than others and also how the distance between words is proportional to their semantic. For example, "Queen" and "King" shares the same distance of "Woman" and "Man" expressing the meaning of genre.
Word2Vec [20] is one of the most used ways to train embeddings, a two-layer neural network that is trained to reconstruct linguistic contexts of words. Word2Vec models generate embeddings that are context-independent: there is just one vector representation for each word, meaning that, if there are multiple instances and senses of the same word, they are combined into one single vector. However, new deep learning models, based on transformers and encoders, have introduced a new revolution. The new embeddings generation process allows to have multiple vector representations for the same word, based on the context in which the word is used. These models can produce embeddings that are context-dependent, ensuring that not only single words, but also sentences or entire documents could be embedded.

2.4. Conclusive Remarks

In this chapter, we have explained the main theoretical knowledge used in this thesis. Particular attention has been payed to some well-known techniques used to represent sounds in frequency domain and also main deep learning strategies that will be used in the proposed method to predict song popularity. Lastly, we have also considered concepts, methods and strategies related to NLP, from which this work will take advantage.

3 | State of the Art

In this chapter, we will report an overview of the state-of-the-art in music popularity prediction and related topics. First of all, we have studied the most relevant approaches in the field, developed and investigated in the last past years. These works can be divided in machine learning and deep learning-based approaches.
The analysis of state-of-the-art in Hit Song Prediction evidences how almost every method, used nowadays in the field, is strictly related to the use of metadata or features manually extracted from music audio and lyrics texts. By the way, the tendency of considering embedding and features automatically extracted from neural networks emerges investigating other fields in Music Information Retrieval, such as music recommendation systems, music genre and mood classification, music generation and many others.
After taking awareness of this, our research has been moved towards state-of-the-art methods to extract meaningful compact representations of texts and spectrograms. The purpose of this choice is obtaining a more powerful way of representing music audio and lyrics. For this reason, we will report the state-of-the-art models to extract audio and text embeddings applied in Music Information Retrieval tasks.

3.1. Music Popularity Prediction Overview

Music Popularity Prediction is the field of studies that aims to investigate how much a song has the potential of becoming popular, after having analyzed its characteristics. This science becomes an interesting field of studies with the growth of the music industry. Understand if there is some repeated pattern or shared characteristic among all the songs that have obtained a great success, in terms of popularity, can help music labels and record companies to produce and focusing only on the most promising songs and artists. To reach this goal, several analysis are conducted to find out which characteristics are more relevant and influence the song popularity. Of course, all these studies are more focused on characteristics that can be quantified, even if the popularity of a song is conditioned on some psychological and cultural factors that make the popularity not completely predictable, but we can assume those aspects somehow related to intrinsic

music qualities.
To extract these aspects and features, a central role is played by the datasets available. In fact, even if many music datasets are published, over the years there has been a lack of data and there is still a lack of information to properly compute an accurate estimation of the impact and the popularity of a song within a platform.

3.1.1. Machine Learning Methods

The very first attempts to predict music popularity are made in 2005 by Dhanaraj and Logan [21], considering the Hit Song Prediction problem as a classification task and addressing it with a Support Vector Machine model that discriminates hit from not-hit songs. The input features used are MFCCs, for what concern the timbral aspect of audio, and the main topic of lyrics extracted among a list of pre-learnt topics, using a Probabilistic Latent Semantic Analysis (PLSA) [22].
Interesting aspects emerged from this study are that lyrics play a key role in classification, but the authors suggest that different kinds of acoustic features should be considered and also time-varying classifiers should be studied to improve results. In fact, this first attempt obtains only a ROC of 0.69 as best result, but the most important aspect to underline is that they demonstrate, through this research, that this kind of classifier is better than random, hence they state that Hit Song Science (HSS) is possible.
Despite the results obtained, this statement is far from obvious and easily accepted. In fact, in 2008, François Pachet and Pierre Roy publish a research [23] in which they show that the popularity of a song cannot be learnt by using state-of-the-art machine learning techniques, contradicting the claims of Hit Song Science. By the way, further investigations [24], lead once again to the affirmation that HSS is possible. Once this becomes a certainty, several authors focus their researches on mining musical track popularity information using ML techniques with the aim of improving the first raw attempts of music popularity prediction.
In particular, improvements start thanks to the availability of Billboard charts [25] and the Spotify's API [26], that finally make gaining data easier. Billboard Hot 100 charts is a record list, published by Billboard magazine, that measures song popularity based on radio airplay, audience impressions, digital song sales and streaming activity. Taking advantage of this, several works [27–31] start from the Billboard Hot 100 charts and then, for each song represented inside the ranking, they retrieve high level acoustic features exploiting the Spotify's API, in particular using the ones related to tracks information.
The last work in terms of time that lead to meaningful results is developed in 2023 [32]. The innovation of this research consists of using a new dataset called "Billboard Hot-100

Songs 2000-2018 w/Spotify Data+Lyric" [33] that contains features related to songs in the Billboard Hot 100 weekly charts from 2000 to 2018, in addition to the corresponding lyrics, for a total of 3581 unique songs. Each song present in the dataset is labelled as "hit" if it reaches the Top 10 positions in the Billboard Hot 100 list at least once, otherwise as "not-hit".
The goal of the model is to classify a song as hit or not based on the features provided by the dataset and using other features computed starting from the metadata.
In particular, the features used for each song are:

- 12 Spotify audio features, such as energy, liveness, acousticness, danceability, etc.
- Song lyrics topic: feature extracted using bag-of-words representation with Latent Dirichlet Allocation (LDA) [34].
- Popularity continuity: an engineered feature that assigns a certain amount of points to each track based on the number of weeks in which it was in chart, i.e. for more than 50 weeks 3 points, between 20 and 50 weeks in chart 2 points, between 20 and 10 1 point and otherwise 0 points are assigned. The utility of this feature indicates that the longer a song can maintain a position in the charts, the more likely it is to become a hit.
- Song title topic: feature extracted using bag-of-words representation with Latent Dirichlet Allocation (LDA) [34].
- Genre class: data that replaces the existing string variable broad genre with a numerical value.

In this work, five machine learning approaches are applied and evaluated: K-Nearest Neighbours, Naïve Bayes, Random Forest, Logistic Regression and also a first simple version of Multilayer Perceptron. A comparison between their results is performed showing that Random Forest (RF) and Logistic Regression (LR) outperforms other models, achieving 89.1% and 87.2% accuracy, and 0.91 and 0.93 AUC, respectively, providing a significant improvement with respect to previous works.
Interesting aspects emerged from this research are that lyrics and title topics once again play a key role in identifying hit songs. On the other hand, the negative side showed up is that the dataset is small both in number of songs and in terms of years considered. Moreover, it takes into account only songs that reach the top 100 position in the Billboard list, so considering also tracks outside this chart could be useful to have a better representation of a real and not biased scenario.
Lastly, a very important point, not already taken into account, is the consideration that

the characteristics of a hit song may change over time. Certainly, to reach further improvements, more complex models have to be employed to include temporal aspects and model changes in popularity over time.

3.1.2. Deep Learning Methods

Due to the fact that Deep Learning techniques improve the state-of-the-art in audio and image processing, in addition to many other fields of study, also in the area of Music Popularity Prediction some research works try to take advantage of this kind of technology. The works developed in this way are very few but, starting from the ones available, it's possible to see how these models outperform, in terms of accuracy, the results obtained with raw machine learning models.
L.- C. Yang et Al. [35] introduce Convolutional Neural Network approach to improve the prediction performances. This work has to be considered important because, for the first time, deep learning is considered a superior approach and HSP a regression problem.
In particular, CNN model takes mel-spectrogram as input and the hit score to be predicted is computed as a product of play count and number of users (both in logarithmic scale) who listened to a song. The dataset used is different with respect to all the other works taken into consideration: in collaboration with KKBOX Inc. [36], they obtain a subset of songs reproduced by Taiwanese listeners over one year.
Also Zangerle et Al. in 2019 [37] treat Music Popularity Prediction as a regression problem but they change the type of features used, considering also low-level features. Going into detail, this work takes the Billboard Hot 100 chart as starting point, but combined it with Million Song Dataset (MSD) [38]. MSD contains one million songs that are representative for western commercial music released between 1922 and 2011. Each song in the Billboard chart is selected from MSD and, for each one, high- and low-level audio features are computed using the Essentia toolkit [39]. The resulting dataset is published and available at [40]. Pushed by the short-life nature of trends in music industry, they do an hypothesis that plays a key role in their work: to allow modeling dynamics more efficiently the authors embed songs with a temporal context by adding to features the release year.

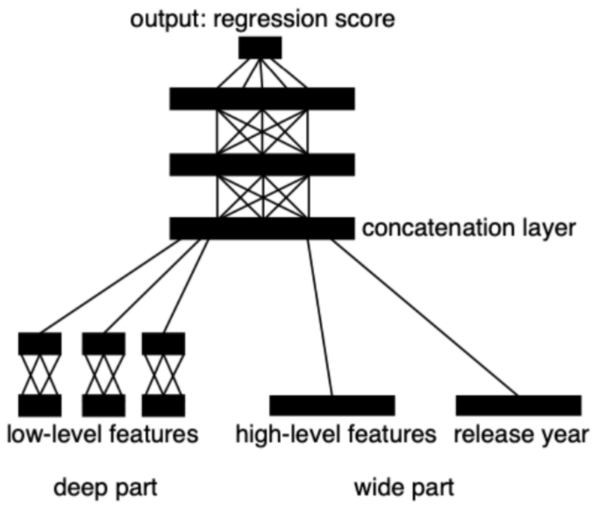


Figure 3.1: Wide and Deep network architecture of Zangerle et. Al [37]

Taking three groups of features of different nature (low and high-level features extracted from audio in addition to the release year), they use a network architecture, shown in Fig. 3.1 , inspired by the structural concept of the Wide and Deep network architecture by Cheng et al. [41]. They use this system to perform a regression task for predicting the peak position a song could reach in the charts.
The relevant aspect is that, even if lyrics are not considered at all, with this work the key role of the release year emerges to allow the network to temporally contextualize songs, reflecting musical trends. In fact, the introduction of the year is a novelty and it's one of the most important contribution of this research, because they find out that the inclusion of this feature, contributes to improve the prediction performance in every experiment by 12–13%.

HitMusicNet

Martín-Gutiérrez et Al. in [42] propose a paper that contributes to improve two aspects in Hit Song Science: they create a new dataset called *SpotGenTrack Popularity Dataset (SPD)* [43] and they develop an innovative multimodal end-to-end deep learning architecture named *HitMusicNet* for predicting popularity of music tracks.
The dataset is created to overcome the restrictions of the already existing datasets. In particular, SpotGenTrack Popular Dataset (SPD) unifies musical knowledges in different fields employing Spotify and Genius API to collect music and lyrics content. SPD starts collecting music tracks data of the top 50 playlists of 26 countries where Spotify is available. For each song, SPD computes and stores many low and high-level audio features but

it provides also the URL's of audio previews as well as the complete set of lyrics to avoid limitations in new features extraction. Some text features are included in the dataset as an initial approach to Natural Language Processing (NLP). In addition, some metadata such as the number of followers of the artist, the popularity of the artist and the number of available markets where the song will be or is released are considered.

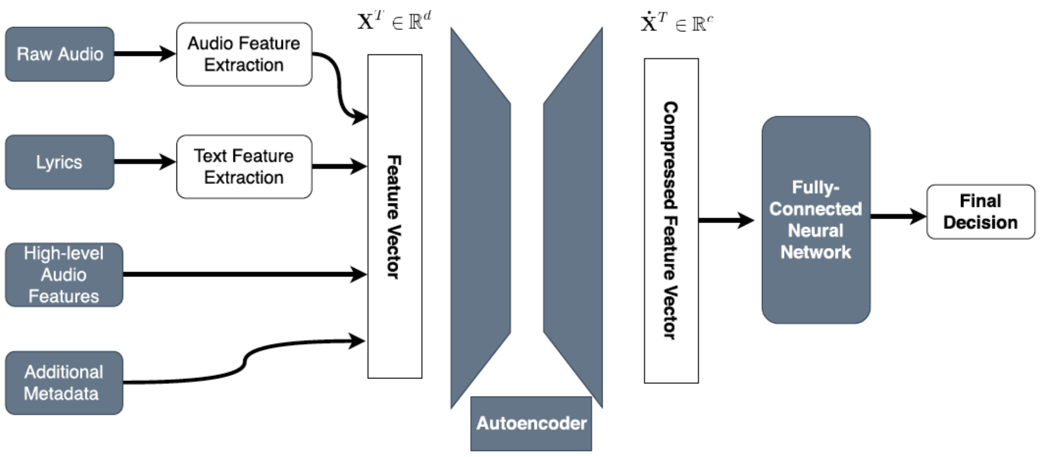

$\mathbf{X}^T \in \mathbb{R}^d$
$\dot{\mathbf{X}}^T \in \mathbb{R}^c$
Raw Audio
Audio Feature Extraction
Lyrics
Text Feature Extraction
High-level Audio Features
Additional Metadata
Feature Vector
Autoencoder
Compressed Feature Vector
Fully-Connected Neural Network
Final Decision

Figure 3.2: HitMusicNet Model

From the point of view of the deep learning architecture used instead, as it can be seen in Figure 3.2, the model is divided in two parts: after having computed all the features and having composed a feature vector, firstly an Autoencoder (AE) is applied to perform features compression. After this step, a multi-layer perceptron, named as *MusicPopNet* and represented in Figure 3.3, receives the compressed representation and, through three hidden fully connected layers, it predicts the popularity value. Using this architecture, many experiments from both classification and regression perspectives are conducted, until the best configuration of the two components in order to address the problem is found. The number of neurons of each layer is based on the input dimension c and three parameters: α_1, α_2 and α_3. In particular, after performing different experiments the set of α_1, α_2, α_3 is determined as 1, 1/2, 1/3 respectively. Hence, the dimension of the hidden layers decreases linearly with respect to the input dimension. Moreover, all the hidden layers have a Dropout layer to avoid suffering from overfitting. Lastly, the dimension of the output layer, denoted with γ, is composed by a unique neuron as the problem is addressed from a regression perspective, with the aim of getting the final prediction in the range [0; 1].

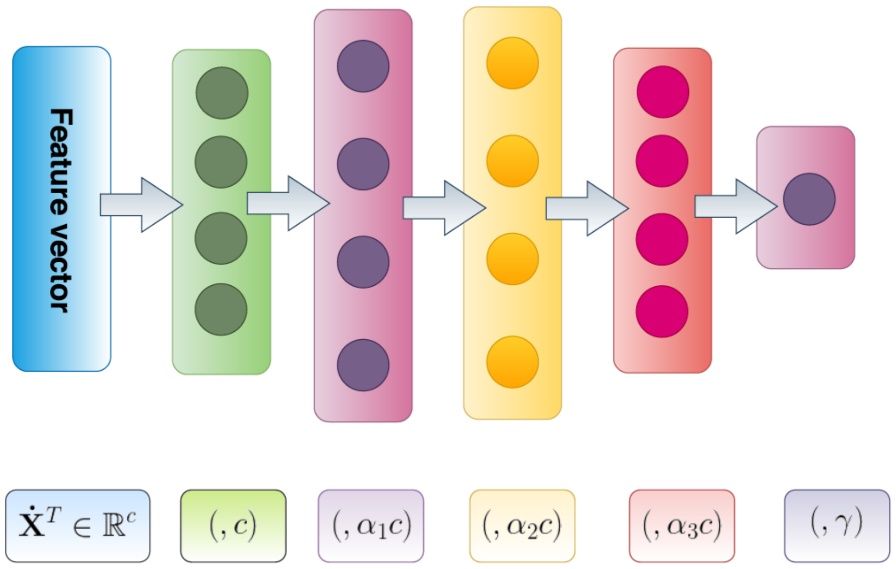


Figure 3.3: Fully-connected Neural Network Architecture of MusicPopNet

Taking advantage of this configuration it is demonstrated that the obtained results are considerably better than in previous studies, conducting to the state of the art in Music Popularity Prediction using DL techniques. Moreover, this research lead to the creation of a new dataset rich of information both regarding audio and lyrics.
In conclusion, despite this successful result, some considerations might be done. For example, automatic extraction of audio features using CNN nor Word Embedding representations for lyrics are not considered, as well as the release year of the analyzed tracks, that might provide temporal contextualization.

3.2. Embeddings

Advancements in deep neural networks lead to learn useful domain-conditioned representations in audio domain from raw audio input with no human intervention, known as deep audio embeddings [44]. Furthermore, deep audio embeddings turn out to frequently outperform hand-crafted feature representations. In fact, pre-trained audio embeddings also achieve state-of-the-art transfer learning performance on many other MIR fields, such as cross-modal retrieval, music genre classification and music tagging benchmarks. At the same way, also for what concern the text representation, deep learning and transformers methods overcome previously used models, as anticipated in Chapter 2.3. For this reason, our study moves on these topics to find which are the state-of-the-art methods to obtain text and audio embeddings.

3.2.1. Audio Embeddings

In Machine Learning, feature maps for data are firstly created and then a classifier is applied on them. As explained in [45], Deep Learning algorithms extract high-level, complex abstractions as data representations through a hierarchical learning process. Using CNN's ability to exploit spatial or temporal correlation in data, as in [46], features that are generated automatically by convolutional and pooling layers can be extracted and then combined with other types of deep neural network to perform classification or regression tasks. Many researches such as [47–51], carried out in the past years, show how features learned by deep Convolutional Neural Networks (CNNs) are recognized to be robust and expressive, so much that they are often preferred to handcrafted ones. This is the reason why features extracted from CNNs have become diffusely used in different computer vision tasks, such as object detection, image captioning and many others.
Encouraged by the positive results obtained in computer vision field, researchers investigate the possibility of apply CNNs also to audio. In 2014, [52] shows that spectrograms can be successfully treated as input images in CNN models paving the way toward exploiting the power of this deep networks also for MIR tasks, following a procedure that can be summarized as in Figure 3.4. For example, as an application of these concepts, [53] uses a CNN model such a MobileNetV2 to extract features and then applies a KNN (K-Nearest Neighbours) algorithm to embedding vectors to perform anomalous sound detection in a self-supervised way.

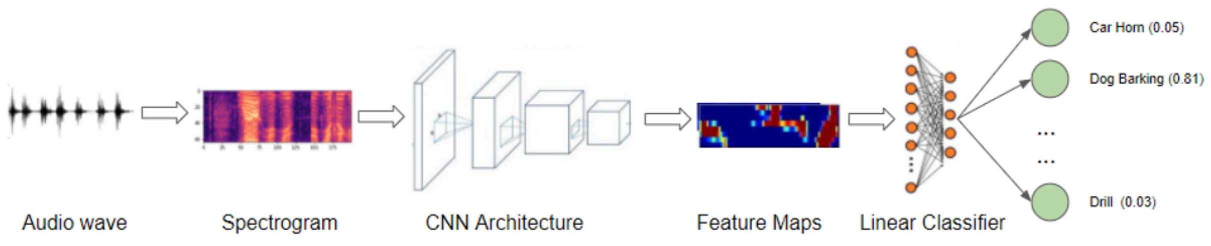


Figure 3.4: Classification chain example using features extracted from a CNN

A turning point in the usage of CNNs coincides with the introduction of the concept of **transfer learning**, pushed by the discovery that using pre-trained weights, instead of randomly initialized weights, leads to better results. In fact, transfer learning is the method used to exploit the knowledge of a model, trained in a particular domain with a large amount of data, to extract useful features for a new task in a related domain, different from the original one but for which less data are available.
In computer vision domain, this happens using ImageNet pre-trained models for particular tasks of object detection and image understanding, such as Medical Image Analysis.

Initially, in audio domain, transfer learning is applied from models exclusively pre-trained on huge audio datasets. Indeed, the very first examples of audio classification tasks, employing transfer learning, use CNN models pre-trained on Million Song Dataset [54] or models like VGG pre-trained on AudioSet [55]. In [47], but in particular Grzywczak D. and Gwardys G. in [52] for the first time show that transfer learning for audio tasks is possible, also from models trained on image datasets, such as ImageNet. However, only [56] understand the potential of this technology using a single model and a single set of input features proving the ability of CNN models to learn boundaries of energy distribution in mel-spectrograms to classify them.

3.2.2. Text Embeddings

As already mentioned, from the beginning of researches in Hit Song Prediction, many studies such as [21, 32] point out the key-role of lyrics in predicting the popularity of a song. In fact, in [57], Singhi and Brown propose a Bayesian network model to perform hit detection based exclusively on lyrics' features that outperforms the state-of-the-art results of that time, demonstrating the power of lyrics in classifying hits. However, that classification is based on rhyme and syllable features, ignoring the semantic and the meaning of the lyrics.
Another example of research, in which features of that kind are used, is HitMusicNet [42]. In fact, Martín-Gutiérrez et Al. in their proposed method use text features such as the total number of sentences, average number of words per sentence, total number of words and average number of syllables per word. Moreover, some engineered features are employed, such as a sentence similarity coefficient, to investigate the influence of repetitive patterns, and also a vocabulary wealth coefficient, that is an indicator of the diversity of the vocabulary, in terms of words. Zhao M. et al. [32] instead use a bag-of-word representation to encode lyrics but, as explained in Chapter 2.3, this approach has many drawbacks, in particular it is computationally expensive and the lyrics significance is once again ignored.
Nowadays, many of the state-of-the-art text embedding models are based on transformer architectures, we have previously explained in Chapter 2.2.3. Transformers revolution Natural Language Processing tasks by providing powerful language modeling capabilities, enabling models to capture long-range dependencies and context in texts.
BERT (Bidirectional Encoder Representations from Transformers) [58] is considered a groundbreaking model in NLP and has a significant impact on the field for several reasons. BERT's idea is born starting from the limitation of all the previous language models to learn general language representations in unidirectional way. BERT

solves the unidirectionality problem by using a “masked language model” (MLM) pre-training objective. As Devlin et Al. in [58] explain, the MLM randomly masks some of the words received in input and the objective is to predict the original masked word based only on its context. Unlike left-to-right language model pre-training, MLM enables the representation to fuse the left and the right context, obtaining a pre-trained deep bidirectional Transformer. This novelty, together with a “next sentence prediction” task that jointly pre-trains text-pair representations, makes BERT embeddings more powerful for various NLP tasks as they capture word meaning based on their context. In fact, due to this ability of extracting the semantic of a text, some researches [59–62] take advantage of BERT/RoBERTa to extract embeddings from lyrics.

3.2.3. Audio and Text Embeddings combined

An overall example in the audio domain of usage of BERT and also CNN to extract embeddings is BECMER [63]. This research work tries to predict the type of music emotion based on audio signal and lyrics.

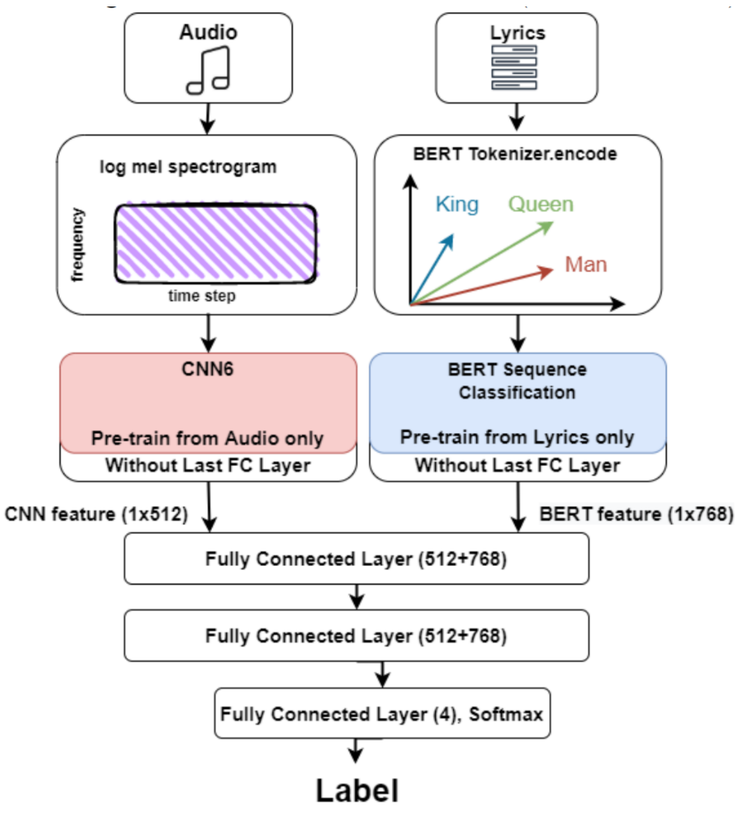

Audio
Lyrics
log mel spectrogram
frequency
time step
BERT Tokenizer.encode
King
Queen
Man
CNN6
Pre-train from Audio only
Without Last FC Layer
BERT Sequence Classification
Pre-train from Lyrics only
Without Last FC Layer
CNN feature (1x512)
BERT feature (1x768)
Fully Connected Layer (512+768)
Fully Connected Layer (512+768)
Fully Connected Layer (4), Softmax
Label

Figure 3.5: BECMER fusion model architecture

In particular, for the NLP chain BERT and ALBERT [64] models are used to compute text

embeddings. Instead, for what concern the audio, CNN is employed to extract features from the mel-spectrograms and at the end the two types of embeddings are fused together to create the input of a multi-layer perceptron, as in Figure 3.5. The experiment confirms that using audio and lyrics information to classify the emotions of songs has a better performance than using the audio-only learning methods as in previous studies, stating the effectiveness of BERT and of audio embeddings.
Other examples, that use audio and text embeddings are [65, 66]. In these researches, the authors explore cross-modal learning in an attempt to bridge audio and language in the music domain. To align text and audio both of them uses features extracted from ResNet-50 [2] model, given in input the melspectrogram of the songs, and a transformer-based model to extract text embeddings. They reach valuable results in multi-modal contrastive learning but also in music classification tasks, confirming the effectiveness of methods that employ BERT's embeddings and features automatically extracted from CNNs.

3.3. Conclusive Remarks

In this chapter, we have introduced the state-of-the-art in the main fields involved in our research. Firstly, we have made an overview of the actual situation in Hit Song Prediction. Taking awareness of the lack of employment of deep learning techniques, that have earned lot of results in other audio tasks, we have moved our attention to state-of-the-art embedding methods to find meaningful representation of audio and texts that can lead our research toward a different interesting perspective with respect to the state-of-the-art in Hit Song Prediction.

4 | Proposed Method

In this chapter, we will formalize and explain in detail our solution to the problem of Hit Song Prediction we are going to address. First of all, having in mind the considerations made exploring the state-of-the-art in the field, after stating the problem, we will describe the choices that lead us to design the global architecture to face HSP. Successively, we will explain more precisely the functioning of each sub-model that contributes to create the overall system. In particular, we will clarify how audio embeddings and text embeddings are extracted with two independent processing chains. Lastly, we will illustrate how this two contributions are used together to build a unique feature vector that is processed by a final model to predict songs popularity.

4.1. Problem Statement

The problem we want to tackle in this thesis is Music Popularity Prediction, also known as Hit Song Prediction. Hit Song Prediction aims to predict whether a song can become a hit or not based on its objective characteristics. In this thesis work, we discriminate hit and not-hit songs assigning to each song a popularity class and also predicting a popularity score, reducing HSP to a multi-class or a regression problem.
The aim of this thesis work is to propose a novel approach with respect to the state-of-the-art works in this field. In fact, we want to investigate the power of embeddings applied to this challenge and their applicability in this area of study. Moved by these ambitions, we use a prediction model that takes as input feature vector an array created joining three contributions: embeddings extracted directly from the audio, text embeddings extracted from the songs lyrics and the songs release year.
These design choices emerge with the study of the state-of-the-art in Chapter 3, because of these main motivations:

- A deep learning approach is not yet applied at its full potential in this field. Studying the state-of-the-art in Hit Song Prediction with particular attention to solutions based on deep learning methods, we have found uniquely some examples of MLP applied to array of high and low-level features in addition to songs' metadata. By the way, advancements in deep neural networks show how CNN-based models obtain great results in classification and retrieval tasks in the audio field.

- Considering the complexity of the problem we want to solve, we do not have at disposal datasets that store so many musical tracks annotated with Spotify's information and for which audio files are available. Due to these motivations and moved by the positive results obtained in researches reported in Chapter 3.2, we want to employ audio and text embeddings taking advantage of transfer learning, to explore their effective usability.
- The majority of the solutions proposed in papers, for example [42], uses as features some biasing information, such as the artists popularity, the markets in which the song is available and the number of followers of the artist. This choice means that songs of already famous artists are facilitated to obtain high popularity score. Instead, we want to conduct our study starting from raw data such as the audio file content and the lyrics transcription of each song, investigating results obtainable in a scenario unconditioned from those metadata.
- Even if some state-of-the-art researches, such as [32], classify musical tracks as "hit" or "not-hit", based on the position a song reaches in Billboard charts, we decide to use as ground truth value of popularity the popularity score assigned by Spotify and available through the Spotify's API. In this way, the problem can be faced as a regression task.
- Many papers developed in the HSP field do not consider the release year of songs. Researches like [40] demonstrate how the presence of this feature can have benefits on the final model results. Moreover, the release year can add a temporal context to songs allowing modeling dynamics more efficiently. For this reason, we consider it as input feature.

4.2. Overall Model Architecture

Our proposed method, bearing in mind the motivations listed before and following the promising results obtained from [42, 63], consists in a neural network model that can be broken down in three main parts: an audio processing chain and a lyrics processing chain, that are totally independent, and a final multi-layer perceptron, as shown in Fig. 4.1.
The system takes in input the .mp3 audio file, the song's lyrics and the song's release year. Then the workflow is divided in two autonomous branches:

- An **audio pipeline** that processes the track audio content and extracts a meaningful representation of it. The feature-extraction system is based on a pre-trained CNN, a *ResNet-50* network, that takes the audio mel-spectrogram as input and from which the embeddings are retrieved extrapolating the features from the second-last inner layer of the ResNet model.

- A **text pipeline** that processes the songs lyrics to extract the text embeddings. The designed system relies on a transformer-based model, moved by their successful application, already explained in Chapter 3.2.2. The input of this processing chain is constituted by the lyrics and they are processed by a *Sentence-BERT* [3].

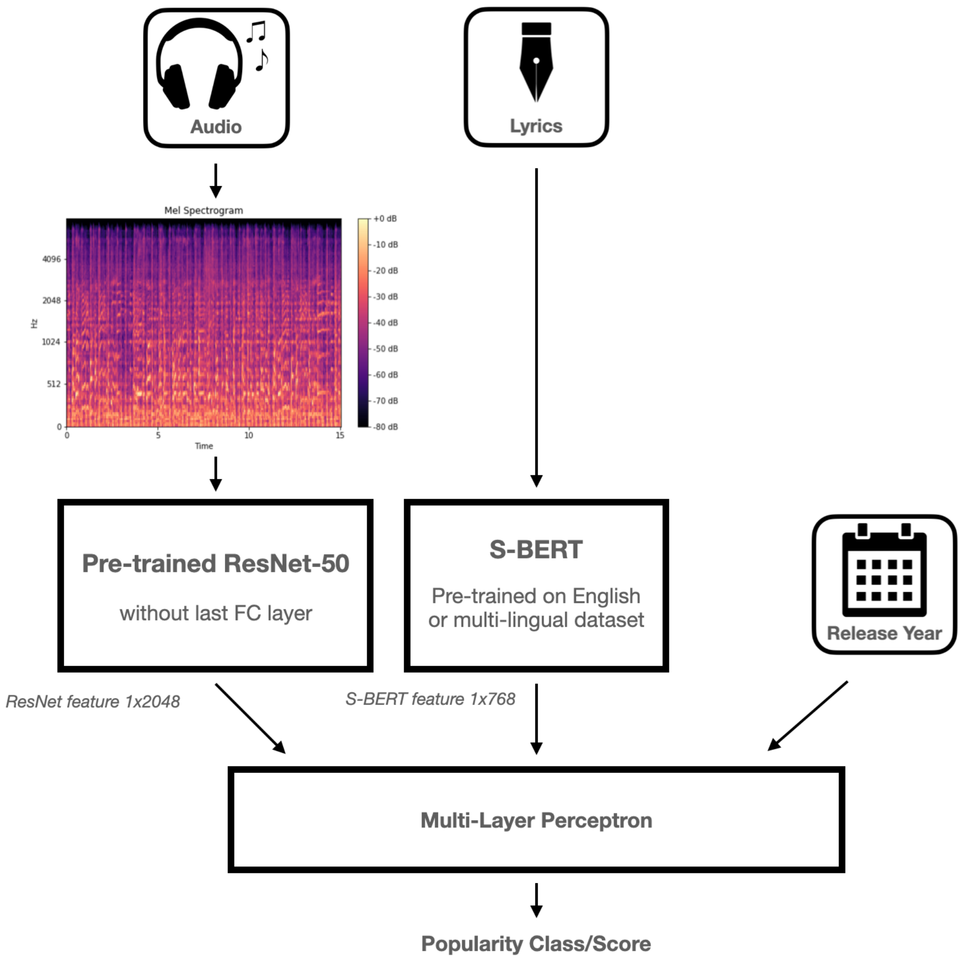


Figure 4.1: Overall scheme of the proposed method to address popularity prediction

After the application of these two independent chains, the two contributions, automatically extracted by audio and lyrics, are concatenated by adding the release year. The feature array obtained is given in input to a final multi-layer perceptron that returns an output of two types: the expected popularity score, between 0 and 100, or a popularity class.

The popularity score is predicted having as target the popularity score achievable through the Spotify's API [26], that is computed considering the number of plays a track has and how recent those plays are. The popularity class of a song is computed as follows:

- **Low Popularity** (class 0): popularity value between 0 and 24
- **Mid-Low Popularity** (class 1): popularity value between 25 and 49
- **Mid-High Popularity** (class 2): popularity value between 50 and 74
- **High Popularity** (class 3): popularity value between 75 and 100

In the next sections each component of this pipeline will be described in detail.

4.3. Audio Processing Chain

In this section we will describe the extraction of audio embeddings starting from the raw audio signal. Firstly, we illustrate the computation of the melspectrograms and then we explain how they are processed to achieve a meaningful and compact representation, suitable for our final prediction purpose.

4.3.1. Audio pre-processing

All the audios can be downloaded using the Spotify's API preview URL, linked to each song. The files obtained have length l equal to 30 seconds. The first operation of the overall system is computing the melspectrogram of each audio file received in input.
Before being ready to be processed, once the spectrograms are retrieved, they are converted to decibel (dB). Lastly, the input log melspectrogram X is submitted to a *Min-Max Scaler* normalization process according to the equation:

$$X_s = \frac{X - Min(X)}{Max(X) - Min(X)} \tag{4.1}$$

4.3.2. Embedding extraction

Starting from this 2-D representation of the signal in time and frequency domains, the aim of the system is to not use hand-crafted features to perform the prediction task, but to obtain a meaningful and compact representation of the raw audio and use it to infer the regression or classification problem. So we want to obtain, from each song audio examined x_a, an audio representation that we call e_a:

$$x_a \to e_a. \tag{4.2}$$

In this perspective, the task of this first audio-processing chain is to return an automatically extracted representation of each song, given as input the relative melspectrogram. As explained in the previous Section 3.2.1, meaningful representations of an image, hence of an audio melspectrogram, can be obtained from convolutional and pooling layers of a CNN.
In particular, in this thesis work, the convolutional-based model chosen to extract the embeddings is a ResNet model [2]. It has been chosen for the audio chain, because of its power and efficiency that have revolutionized the field of computer vision. The key introduction of ResNet is the concept of residual learning, which allows the network to be significantly deeper than previous architectures without suffering from the vanishing gradient problem, that leads to a degradation of the performances. The main idea to prevent degradation is the use of residual connections, also known as skip connections, which skip one or more layers and directly connect the input of a layer to its output.
The theoretical concept behind residual learning is the following. In a standard feed forward neural network, each layer taking a certain input x aims to learn the desired underlying mapping $H(x)$. When training deep networks with many layers, a significant issue arises due to vanishing gradient problem. During back propagation, gradients tend to become extremely small as they are propagated backward through many layers. This can result in very slow convergence or even in the complete failure of the network to learn. In residual connections, the stacked nonlinear layers aim to fit another mapping

$$F(x) = H(x) - x, \tag{4.3}$$

that represents the difference (or residual) between the desired mapping $H(x)$ and x. The hypothesis under this strategy is that it is easier to optimize the residual mapping than to optimize the original, unreferenced mapping. Following this reasoning, the original mapping is recast into

$$H(x) = F(x) + x. \tag{4.4}$$

Figure 4.2 shows that, having identity connection coming from x, residual block aims to learn the residual function. As advantage, in an extreme situation, it would be easier to push the residual to zero than to fit an identity mapping by a stack of nonlinear layers. If $F(x)$ is close to zero, it means the layer has learned to preserve the input's information while modifying it as needed. In this way, exploiting skip connections, the training process can propagate larger gradients to initial layers, and these layers also could learn as fast as the final layers, gaining the ability to train deeper networks.

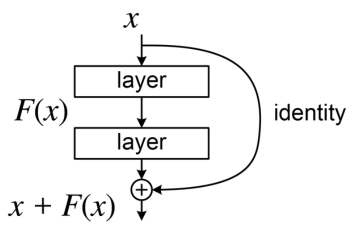


Figure 4.2: Residual learning schematic representation [67]

Another important aspect of ResNet is that the deep representations learned by this type of networks are highly transferable. Pre-trained ResNet models on large datasets, like ImageNet [10], can be fine-tuned for various specific computer vision tasks, even with limited labeled data. This transfer learning capability makes it easier for researchers and developers to create accurate models for new tasks with less data and training time.
The promising results obtained from [65, 66], that demonstrated ResNet is also well-suited to work with audio melspectrograms, joined with the reasons explained before, make ResNet the model we rely on in the audio embeddings extraction process.

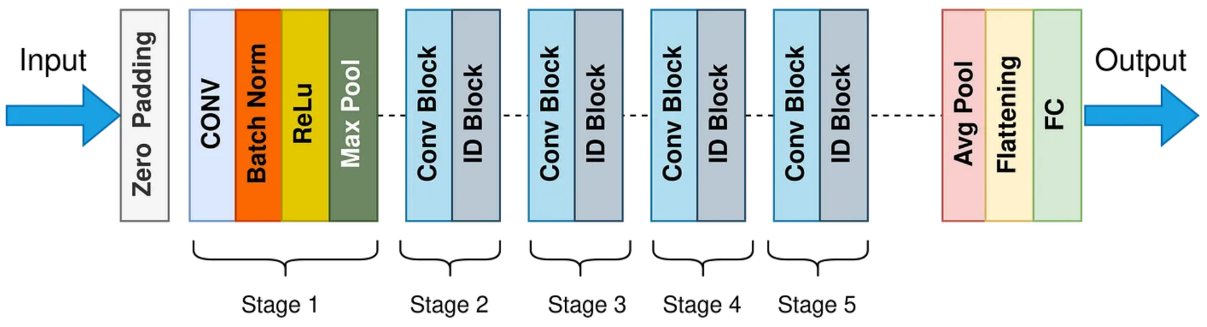


Figure 4.3: ResNet50 schematic model architecture [68]

Going into detail, the model that processes the audio melspectrogram is a ResNet-50, a version of ResNet that consists of 50 layers, including convolutional and fully connected layers. These layers can be divided into five stages, as shown in Figure 4.3. Each stage consists of a sequence of convolutional residual blocks. According to the residual learning theoretical concept explained before, in ResNet-50 bottleneck residual blocks are implemented as shown in Figure 4.4 in a specialized version designed to be computationally efficient while maintaining the benefits of residual learning. The block consists of three main components: 1x1, 3x3, and 1x1 convolution layers each one followed by Batch Normalization and a ReLU activation function.

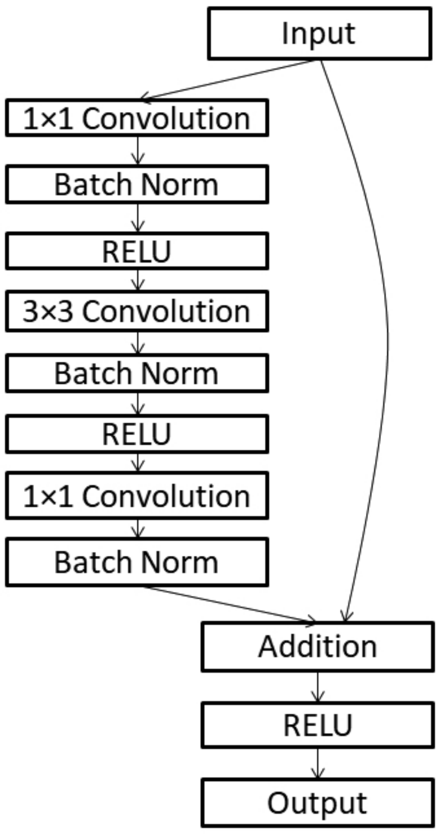


Figure 4.4: ResNet50 schematic architecture of a residual block

More precisely, the architecture of a ResNet-50 can be described by the following building blocks that process the melspectrogram given as input:

- **Stage 1 - Initial Convolutional Layer**: the first layer is a standard convolutional layer followed by batch normalization and ReLU activation function to prepare the input image for further processing. The images typically have three color channels (e.g. RGB). The output tensor is then passed through a max-pooling layer to reduce spatial dimensions.
- **Stage 2**: it consists of three residual blocks. The number of kernels in these blocks increases from 64 to 256 progressing through the stage.
- **Stage 3**: it consists of four residual blocks. The number of filters in these blocks increases from 256 to 512 progressing through the stage.
- **Stage 4**: it consists of six residual blocks. The number of kernels in these blocks increases from 512 to 1024 progressing through the stage.

- **Stage 5**: it consists of three residual blocks. The number of filters in these blocks increases from 1024 to 2048 progressing through the stage.
- **Global Average Pooling**: it reduces the spatial dimensions of the feature maps to 1x1, resulting in a feature vector of dimensions 1x2048.
- **Fully Connected Layer**: this is the final layer that maps the feature vector to the predicted class, using a Softmax activation function. The number of neurons in this layer is typically equal to the number of classes to which it can belong.

Exploiting this architecture, ResNet-50 is able to extract relevant aspects from images and to classify them. In fact, our interest in ResNet is due to its ability of automatically filtering and extracting relevant features of images, hence in our particular case from melspectrograms. In order to make the ResNet-50 learn to handle this type of images, starting from the ImageNet weights, we pre-trained it on the GTZAN Genre Dataset, introduced in [69].
After having pre-trained the model, it is used to perform transfer learning and to return embeddings from the music tracks of which we want to predict the popularity. In order to extract the embedding, the model is used with all its layer except the last one. In this way, the last fully connected layer is discarded and the result we obtain given a mel spectrogram in input is a feature vector of dimension 1x2048.

4.4. Text Processing Chain

For what concern the text embedding extraction, after retrieving each song lyrics x_l, it is necessary to perform text vectorization, to achieve a compact numerical representation of x_l that preserve the meaning of the text and that can be easily processed by the system. For this reason, we want to obtain, from each song lyrics examined, the lyrics representation that we call e_l

$$x_l \rightarrow e_l \tag{4.5}$$

As explained before in Chapter 2.3, there are several techniques for representing text as vector. The most suited one to be used is determined by the task the system has to carry out and the aspects related to text most important according to it.
In our case, the choice is conditioned by two factors:

- The system is working with lyrics and the focus is considering the meaning of the texts rather than their linguistic constructs of words or their grammatical structure.
- The system has the constraint of needing embeddings that have an homogeneous

representation in size, independently from the length of the text, in such a way that every song has assigned the same number of features.

Taking into consideration these requirements, as anticipated before, we develop a text processing chain that takes advantage of a transformer-based model.
Considering the work examined in the Chapter 3.2.2, regarding the text embeddings model, both BECMER [63] and MuLan [66] architectures consider the commonly used Bidirectional Encoder Transformer (BERT) in its base architecture. However, BERT architecture has not been developed to compute independent sentence embeddings but, in its original form, it focuses on token-level embeddings for individual words within sentences. For these reason, many studies in the past years have been conducted leading researchers to developed a new model called Sentence-BERT [3]. It is able to overcome the BERT limitation in tasks like semantic search, which require a strong sentence-level understanding and that can not be approached using just word-level transformers. We use Sentence-BERT since in this work we use song lyrics that are typically organized in sentences or verses, each carrying its own meaning.

Sentence BERT (S-BERT) [3] is a pre-trained transformer network that is developed by modifying pre-trained BERT/RoBERTa [70] network, adding a mean pooling operation to the output to obtain fixed-length sentence embeddings. During the fine-tuning training process, S-BERT employs a Siamese network architecture. Siamese neural network is a particular type of network that contains two (or more) identical models that usually have the same weights. Also parameters are shared and the updating is performed identically across both sub-models. Due to these characteristics, siamese networks are normally used to compute similarity scores of the inputs, in order to compare them.
In Figure 4.5, it is shown how the model is built with two identical BERT models. During the training phase, it can be seen that one sub-network is used to encode the first sentence and the other is used to encode the second sentence independently. After encoding both sentences, S-BERT produces fixed-length embeddings that capture the semantic meaning of each sentence. To compare sentence embeddings and determine their similarity, S-BERT typically uses cosine similarity and then apply a Softmax loss function. As training progresses, the model learns to map sentences into an embedding space where similar sentences have embeddings that are close in distance, and dissimilar sentences have embeddings that are far apart.

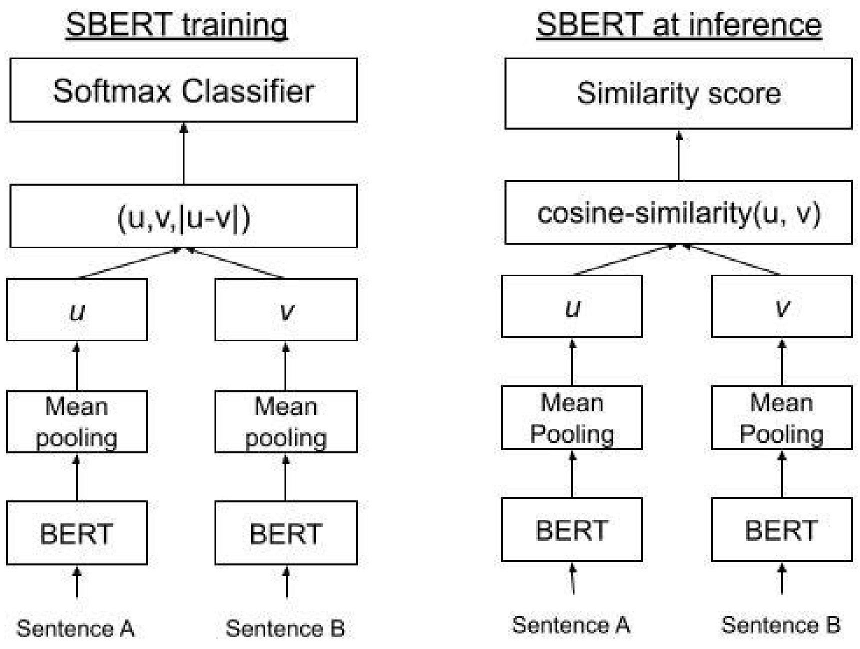


Figure 4.5: S-BERT training process [71]

Exploiting this, S-BERT directly produces embeddings that capture semantic meaning and similarity for entire sentences or paragraphs. This makes it well-suited for tasks like semantic search and document retrieval, based on similarity of meaning.
In our model case, we don't want directly to compare sentences, to know if they are similar in meaning, but we are interested in finding embeddings for each lyrics text and then using its representation as feature vector on which the system performs the popularity prediction. In order to do this, in this chain of text processing the system uses a pre-trained S-BERT only to encode lyrics. First of all, the input text is pre-processed, which includes tokenization, lowercasing, and any other necessary text cleaning steps. Tokenization splits the text into words or subword tokens, following the behaviour of the tokenizer used during pre-training. Successively, these input sentences pass through a pre-trained sentence embedding model, which is capable of producing contextual embeddings for each token in the sentences. An average pooling strategy is applied to obtain a fixed-size sentence embedding from the token embeddings. Finally, the resulting sentence or paragraph embeddings are normalized with an L2 normalization to have unit length and to ensure that they are comparable in terms of cosine similarity.

S-BERT models are developed for various languages, making them suitable for multilingual applications, such as cross-lingual similarity measurement, where sentences in different languages are compared. As explained in [72], multilingual models are developed

through knowledge distillation technique. This consists in transferring knowledge from a large, complex model (teacher) to a smaller, more efficient model (student). The distillation technique works with a fixed (monolingual) teacher model, able to produce sentence embeddings in one language. The student model is supposed to mimic the teacher but, in order to make the student "learn" further languages, it is trained on parallel (translated) sentences. The translation of each sentence is also mapped to the same vector as the original sentence.
There are different pre-trained versions of S-BERT available: the most used ones are reported in Table 4.1, each one with its correspondent performance evaluated on different datasets and tasks. We evaluate our model on two different datasets: one that contains solely English lyrics and a more extended version of it that also contains songs with French, Spanish, Italian, German and Portuguese lyrics. For this reason, we employ three types of pre-trained Sentence-BERT:

- *all-mpnet-base-v2*: it is obtained using the pre-trained microsoft/mpnet-base model and then fine-tuning it on a one billion sentence pairs dataset. Despite of the low speed and the high size, it is the model that provides the best quality, but it is trained only in English. By default, input text longer than 384 word pieces is truncated. The embeddings produced are already L2 normalized.

- *multi-qa-mpnet-base-dot-v1*: it is trained on 215 million (question, answer) pairs from diverse multi-lingual sources. It takes by default input text no longer than 512 word pieces, otherwise texts longer than that threshold are truncated. The embeddings produced are not normalized.

- *paraphrase-multilingual-mpnet-base-v2*: it is trained using "paraphrase-mpnet-base-v2" as teacher model and "xlm-roberta-base" as student. It is able to manage 50+ languages and by default, it takes as input texts no longer than 128 word pieces. The embeddings produced are not normalized and so a L2 normalization operation has to be performed.

All these three models map sentences and paragraphs to a 768 dimensional dense vector space that is suggested to be used for tasks like clustering or semantic search.

Model Name	**Performance Sentence Embeddings**	**Performance Semantic Search**	**Avg. Performance**	**Encoding Speed** (sentence /sec)	**Model Size**
all-mpnet-base-v2	69.57	57.02	63.30	2800	420 MB
multi-qa-mpnet-base-dot-v1	66.76	57.60	62.18	2800	420 MB
all-distilroberta-v1	68.73	50.94	59.84	4000	120 MB
all-MiniLM-L12-v2	68.70	50.82	59.76	7500	120 MB
paraphrase-multilingual-mpnet-base-v2	65.83	41.68	53.75	2500	970 MB

Table 4.1: Five S-BERT pre-trained models, with the best performances, available at [73]

4.5. Combination of the two systems

Once the two processing chains are applied, the system ends up with:

- The feature vector that contains the **audio embedding** e_a
- The feature vector that contains the **lyrics embedding** e_t

These two arrays are then concatenated with the release year x_{year} to create the final feature vector that represent each song x. The aim of this final part of the system is to take all these contributions and compute the popularity prediction as:

$$P_x = f(e_a, e_t, x_{year}), \tag{4.6}$$

having as f the neural network function that computes the popularity.
The Multi-Layer Perceptron in charge to process these information has different technical settings based on the problem addressed. In fact, different version of MLP are presented in chapter 5.2.3 based on the dataset used and the experiment conducted. In order to

compute the popularity prediction in the case of multi-lingual dataset, the MLP has the architecture shown in Figure 4.6. In general, every model used during the experiments have a structure that can be resumed as: the input layer, some hidden layers and the output layer.

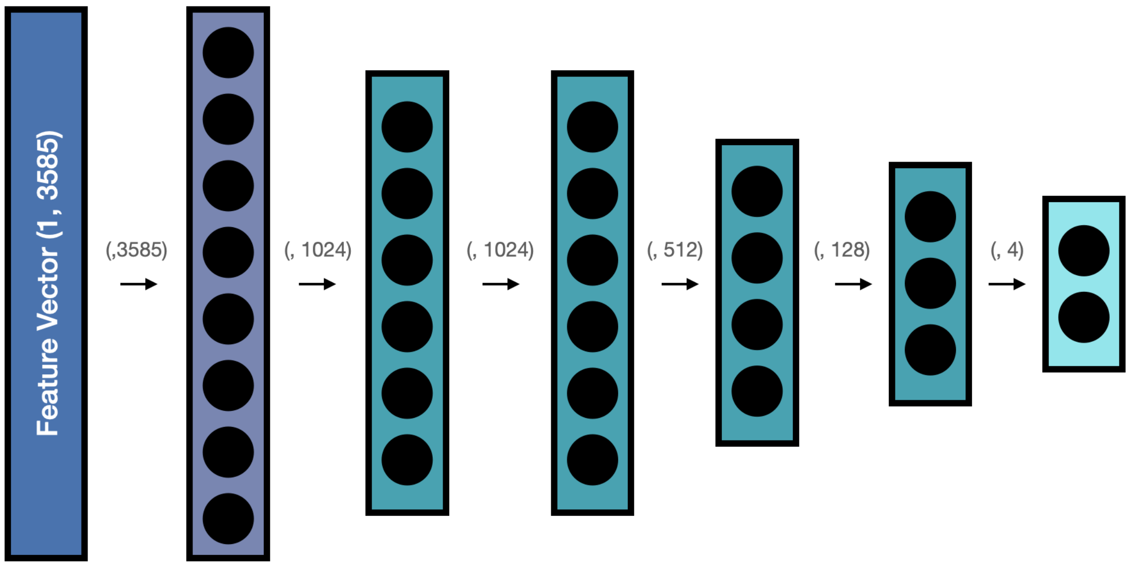


Figure 4.6: Multi-layer perceptron schematic architecture

Each sequential block that composes these architectures is built with the following layers:

- **BatchNorm1D**: this layer computes the mean and variance of the input and uses them to regularize data. This effectively scales and shifts the input data, making them more consistent across batches and reducing the impact of feature scale variations. It's very important to apply it as first step because we have embeddings and release year that are not comparable as measure.
- **Dense layer**: a linear layer with ReLU activation function to introduce non linearity into the model.
- **Dropout**: a regularization technique used to prevent overfitting.

As result of the output layer, based on the task the model has to deal with, the MLP returns the popularity class assigned to the song given as input or the popularity score.

4.6. Conclusive Remarks

In this section, we have described the structure of our model in detail, starting from the statement of the problem we want to investigate. Successively, we have discussed the design choices that lead us to the proposed model architecture. Then, we have analyzed each step of the popularity prediction pipeline, focusing on the pre-processing operations, the extractions of the audio and the text embeddings. Lastly, we have explained how the audio and text contributions are used to predict the popularity.

5 | Experiments and Results

In this chapter, we will describe the results obtained with the experiments, carried out in order to evaluate our model and prove its applicability in the Hit Song Prediction challenge. In particular, we will first describe the datasets used in training and validation phases. Successively, we will report an overview of how the experimental setup is configured and which are the metrics we will use to evaluate the performance of our model. Finally, tests and relative results are listed with observations and discussions.

5.1. Dataset

In this section, creation process of the dataset used during the tests is exposed. The dataset from which our research has began is the *SpotGenTrack Popularity Dataset (SPD)* [43], a database for Music Popularity Prediction, Genre Classification and many other tasks related to Music Information Retrieval, already mentioned in Chapter 3.1.2 as one of the contributions of Martín-Gutiérrez et Al. [42].
It consists in different .csv files, each one containing several information regarding 101939 tracks. The data collected are about not only the track but also the linked albums, artists, low-level audio features, high-level audio features obtained via Spotify's API and text features obtained using Genius' API. The tracks, from which the data are extrapolated, are gathered considering 26 countries where Spotify is available and for each country, taking in consideration the top 50 playlists per category.
Analyzing them into detail, the features collected from both audio and lyrics analysis, processing the 30 second audio previews, that can be retrieved from the Spotify preview URL, are the following:

- **Low-level Audio Features**: Mel Frequency Cepstral Coefficients (MFCCs), Mel-Spectrogram, Spectral Centroid, Energy Entropy, Spectral Roll-Off, Constant Q-Chromagram, Octave-based spectral contrast, Zero Crossing Rate and Tonnetz (a novel approach for detecting changes in the harmonic content of musical audio signals).

- **High-Level Audio Features**: acousticness, danceability, duration [ms], energy, instrumentalness, key, liveness, loudness, mode, popularity, speechiness, tempo, time signature, valence.
- **Text Features**: Total number of sentences, average number of words per sentence, total number of words, average number of syllables per word, a *sentence similarity coefficient* to investigate the influence and the correlation of repetitive patterns in lyrics with the popularity of the musical track and a *vocabulary wealth coefficient* that is an indicator of the diversity of the vocabulary, in terms of words, employed when writing a certain lyric.

The popularity value [74] of a track is computed by a Spotify's algorithm and it is mainly based on the number of plays of a track and how recent those plays are. The popularity can assume vales between 0 (not popular at all) and 100 (most popular).

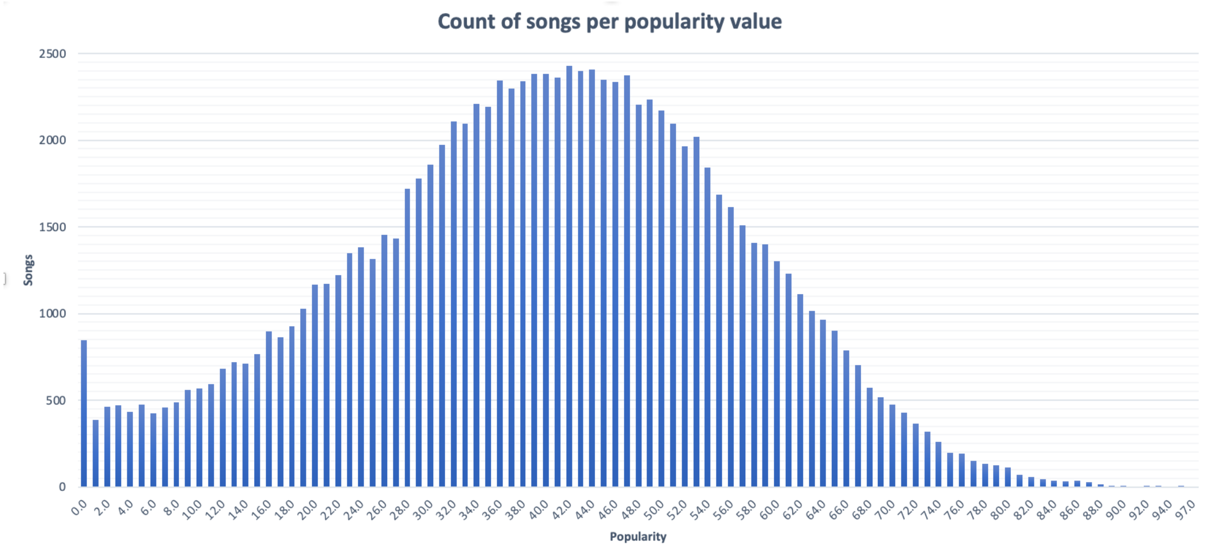


Figure 5.1: Popularity distribution of songs in SPD [43]

For example, having two songs with the same number of listening, if the first track has with the majority of plays now and the latter has a lot of plays in the past, the first one will surely have a higher popularity. Moreover, tracks that have a duplicate, for example as a single and as an album, are rated independently. Instead, artist and album popularity is derived mathematically from track popularity.
For this reason, artist and album popularity haven't been taken in consideration during the prediction process because they can lead to a misleading result: tracks related to already popular artists have for sure an higher score while our purpose is mainly analysing the solely characteristics that make a song popular or not.

The popularity value stored in this dataset is updated to 2019 and it has the distribution reported in Figure 5.1.
Even if the majority of researches in Hit Song Prediction use the Billboard 100 Hot Songs as dataset, for this thesis work the SPD has been chosen mainly for the presence of the preview URL of each song, allowing the download of the 30 second preview of the tracks and, as a result, the possibility of computing other features and handle the spectrogram of each song directly as well.
Starting from SPD dataset, we perform a set of operations schematically listed in Figure 5.2 to remove all the songs stored that have some missing information or that are not useful for our thesis, ending up with two new versions of the SPD dataset.

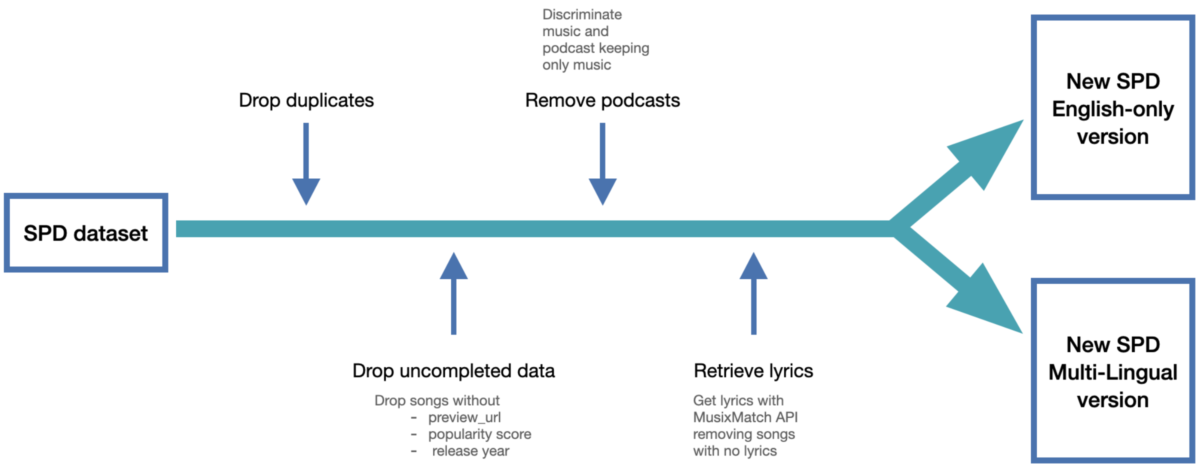

SPD dataset
Drop duplicates
Discriminate music and podcast keeping only music
Remove podcasts
Drop uncompleted data
Drop songs without
- preview_url
- popularity score
- release year
Retrieve lyrics
Get lyrics with MusixMatch API removing songs with no lyrics
New SPD English-only version
New SPD Multi-Lingual version

Figure 5.2: SpotGenTrack Popularity Dataset cleaning process schema

5.1.1. Speech-Music Classification

Analyzing the audio files of the tracks contained in the dataset, evidences have proved that not only musical tracks has been stored inside the dataset but podcasts tracks as well.
To avoid misleading results due to the presence of podcasts recordings inside the dataset, we have developed a Neural Network model able to discriminate speech and music tracks. The model used to perform this classification is a CNN trained on the GTZAN Music Speech dataset [75], a set of audio labelled with speech or music tags. The dataset consists of 120 tracks, each containing 30 seconds of audio. The tracks are all 22050Hz Mono 16-bit audio files in .wav format and for each class (music/speech) the dataset contains 60 samples.
After having performed this first distinction between podcast and music tracks, we have

obtained the results represented in Figure 5.3. We have found out that, among all the audio files contained in the SPD dataset, 6744 audio correspond to podcast content and the other 94857 audio correspond to music tracks.

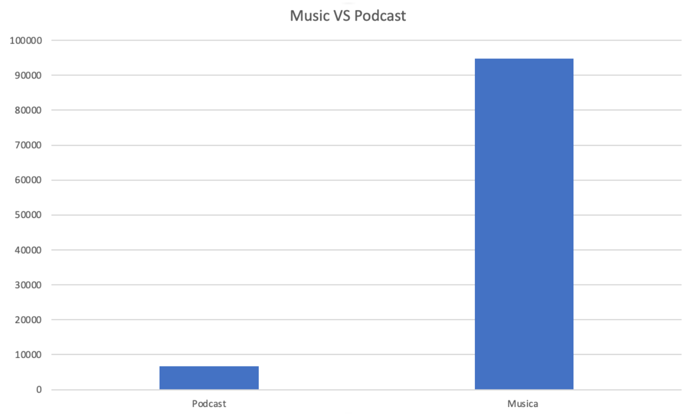


Figure 5.3: Music VS Podcast distribution

5.1.2. Language Filtering and Lyrics Retrieval

After having performed this first cleaning operation of the dataset, we have focused on lyrics. First of all, we have conducted an analysis of the language distribution among the songs. To perform language detection we have decided to use *langdetect* library, a port of Google's language-detection library [76] that supports 55 languages.

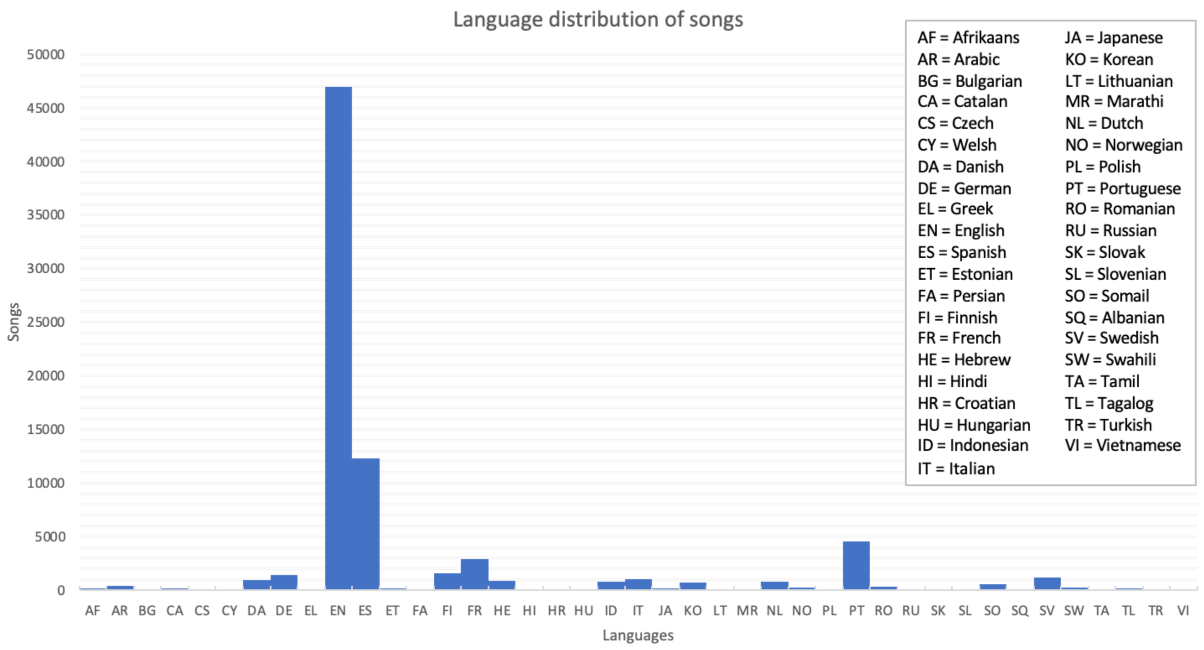


Figure 5.4: Music tracks' language distribution

The final distribution we have obtained is shown in Figure 5.4 and it evidences how even if the majority of the songs are in English there is also a great amount of songs in many other different language. In conclusion, we have ended up with a set of 46955 English musical tracks.
By doing this analysis on lyrics, some evidences proved that some songs do not correspond with the lyrics they have associated in the dataset. For this reason, we decided to retrieve the original lyrics taking advantage of the Musixmatch's API [77]. In particular, using the Musixmatch Python library, starting from the title and the name of the artist associated to each song, we used an API to get lyrics: the *matcher.lyrics.get* API method. Giving as parameters the title and the artist, this method allowed us to obtain the lyrics and the correspondent Musixmatch Id of each song.
Finally, removing all the songs for which the lyrics are not available, the duplicates and the ones for which the release year can not be found through the Spotify's API, the final version of the SpotGenTrack Popularity Dataset we obtained consists of 26711 well-annotated in English, with popularity distribution shown in Figure 5.5, that we call *SPD-English*.

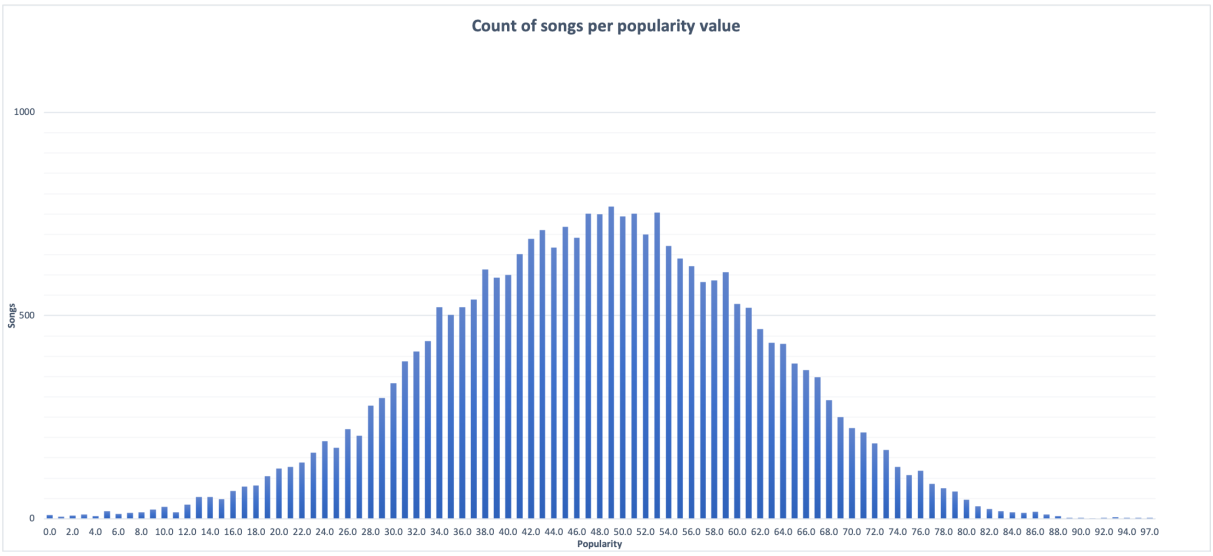


Figure 5.5: Popularity distribution of our English solely dataset (SPD-English)

The average popularity is 47.26 and the standard deviation is 13.98.
The choice of using only songs with English lyrics comes from the Sentence-BERT pre-trained models available and their correspondent results. For this reason, as explained in section 4.4 based on its average performance we choose the *all-mpnet-base-v2* that is the model that provides the best quality but it is trained only in English.

5.1.3. Increasing the dataset size

Because of the small amount of well-annotated songs remaining after SPD dataset cleaning, we looked for a way to increase the number of songs within the dataset, so as to improve the performance of the model given the complexity of the problem to be solved. In order to obtain additional song, we used two datasets:

- "Billboard Hot-100 Songs 2000-2018 w/Spotify Data+Lyric" dataset [33] already used by Mengyisong Z. et Al. in [32]. The dataset includes all songs in the Billboard hot 100 weekly charts from 2007 to 2017, as well as audio features, metadata and lyrics of each song provided by Spotify. The raw dataset includes 33 attributes for a total of 7574 songs.
- A set of Billboard's Year-End Hot 100 songs between 1964 and 2015 [78] collected using the Spotify's API for the audio features and Musixmatch's API for the lyrics, for a total of 5101 songs.

Before combining these two datasets, we remove all those songs that have some missing information such as the lyrics, the popularity, the release year or the preview URL available from Spotify that is fundamental to be able to gain the audio file of each song and so to compute the relative melspectrogram. Finally, after all the filtering process to maintain only the well-annotated data, we ended up with a final dataset of 7604 songs well-annotated of which 6625 in English. Before joining our clean SpotGenTrack Popularity Dataset version and our Billboard dataset version, we call the Spotify's API devoted to get the track information to retrieve the updated popularity value, to be sure that the annotation for all the songs considered are updated to the same date.
Going into detail, we use the Spotipy Python library [79] to interact with the Spotify's API, following these steps:

- Taking advantage of Spotipy system to manage the Spotify's OAuth2 authentication process, we create an object of class SpotifyClientCredentials that takes as parameters the Spotify CLIENT_ID and CLIENT_SECRET obtainable after having created an app from the Spotify's for Developers dashboard.
- An object of class Spotify is created using the SpotifyClientCredentials object already defined. The Spotify object is able to call every Spotify's API with the properly authorization being set.
- Exploiting the Spotify object, the Spotify's API endpoint to get the popularity information, called Get Track API, is called. It takes as parameters the unique Spotify ID that identifies a single track and returns as response the correspondent

Spotify catalog information, such as the track number, the available markets, the preview URL, the popularity value and others.

After this chain of steps, we obtain a dataset that takes advantage of both SPD and Billboard dataset's for a total of 29078 English songs.

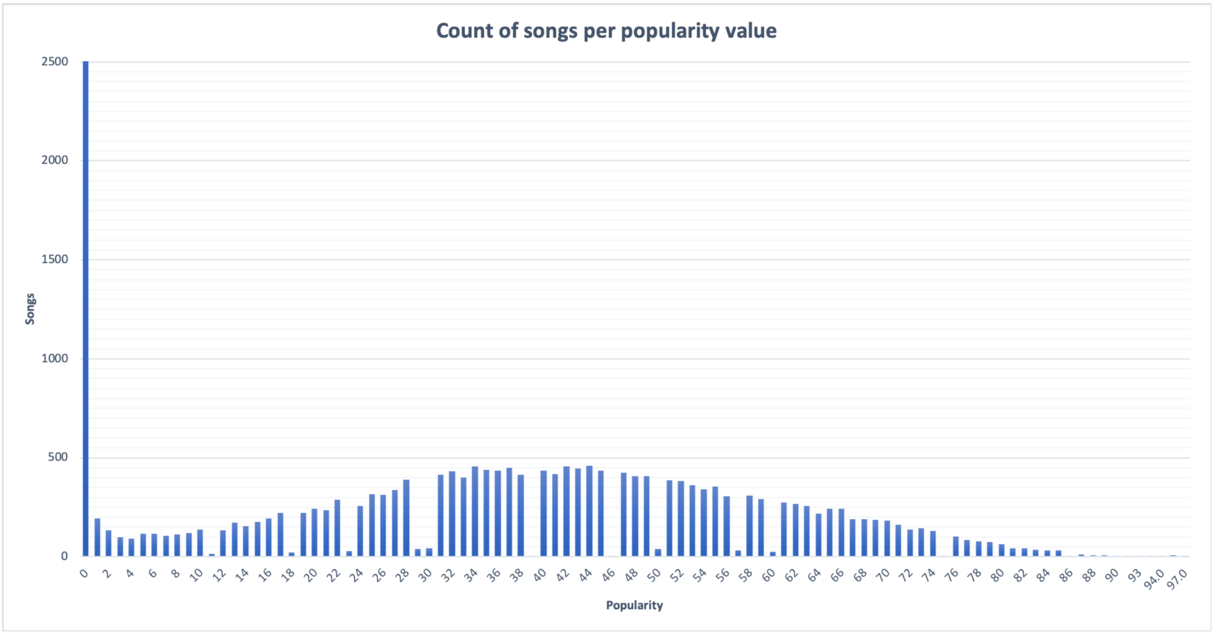


Figure 5.6: Distribution of the updated popularity obtained joining SPD and Billboard datasets

At this point, another observation is done: having passed at least 3 years from the construction of the SPD dataset the songs that at that time were popular, now we discover that have popularity score 0. As result, the final popularity distribution is unbalanced towards, zero as can be seen in Figure 5.6.
Instead of having a small increase in size but a dataset not balanced, we disregard this option of integrating the Billboard dataset. In conclusion, to increase the number of songs with which training our system, we consider also songs with lyrics in Spanish, French, German, Italian and Portuguese from the cleaned version of SDP ended up with a collection of data relative to 41333 songs, that we call *SPD-Multilingual*. In order to obtain lyrics, the steps followed are the same used in the case of English dataset, taking advantage of the Musixmatch's API. As mentioned before, the negative side of this choice is that the average performance of the Sentence Transformer used with multi-languages is worst than the results obtained by Sentence-BERT trained only in English. On the other hand, the positive side is the great amount of songs we gain considering also other languages.
Moreover, in Figure 5.7 we can see how the popularity has a Gaussian-shaped distribu-

tion, having average popularity 46.45 and standard deviation 13.97.
After all this pre-processing phase, we decide to take experiments employing the dataset of solely English song and this last multi-lingual dataset, obtained from the SPD dataset after having re-collect lyrics and after having clean all the data not properly annotated or with some missing information.

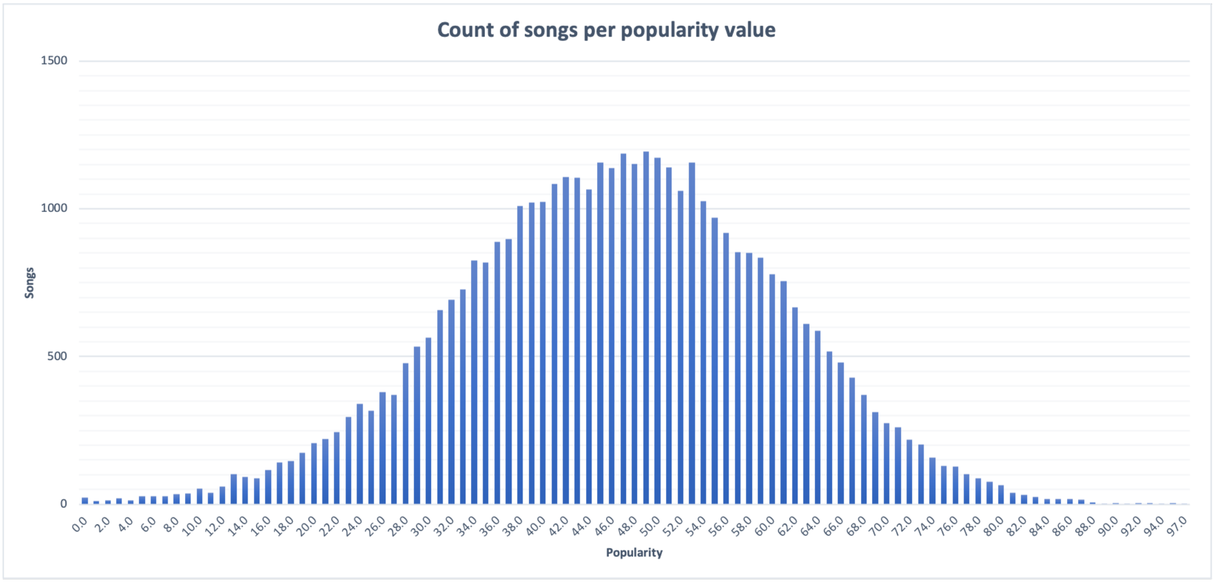


Figure 5.7: Popularity distribution of the multi-lingual dataset (SPD-Multilingual)

The attributes stored in these final datasets are the ones we need for our thesis work: preview URL, release year, lyrics and popularity value computed in 2019. In addition, we stored also the Spotify ID and the Musixmatch ID in case we need in future to retrieve other information from the relative APIs. The structure of the dataset and some rows taken as example are shown in Table 5.1.

Spotify ID	Musixmatch ID	Preview URL	Release Year	Lyrics	Popularity
6lsumzVFKP9SnfKcuGwqVp	87425313	https://p.scdn.co/ mp3-preview/...	2000	You and me we're meant to be Walking free in harmony One fine day we' ll fly away Don' t you know that Rome wasn' t built in a day ...	63
...	...	...	...	...	...
3MjUtNVVq3C8Fn0MP3zhXa	113838494	https://p.scdn.co/ mp3-preview/...	1999	Oh, baby, baby Oh, baby, baby Oh, baby, baby, how was I supposed to know That something wasn't right here? Oh, baby, baby, I shouldn't have let you go And now you're out of sight, yeah ...	82
0ofHAoxe9vBkTCp2UQIavz	227606494	https://p.scdn.co/ mp3-preview/...	1977	Now here you go again, you say you want your freedom Well, who am I to keep you down? It's only right that you should play the way you feel it ...	87

Table 5.1: Sample rows extrapolated from the final dataset

5.2. Experimental Setup

In this section, more technical details about our experimental setup are described. First of all, the method to pre-process audio file, used in training and evaluation phase, is explained. Successively, the pre-training strategy used to fine tune the ResNet-50 model is reported. Lastly, regarding the multi-layer perceptron used as last prediction phase, technical specifics about layers, neurons and hyper parameters used during the training procedure are depicted.

5.2.1. Audio Pre-Processing

Both in training and validation phase, audio files have to undergo pre-processing operations to ensure each audio is in the same suitable format for the neural network to learn effectively and efficiently.
In particular, using Librosa [80] Python library, first of all the signals are re-sampled with a sample rate $F_s = 22050$ Hz, to maintain a common sampling frequency between all the tracks. After this step, the melspectrogram is computed using these parameters:

- $window\_size$=1024 samples

- hop_size=512 samples
- n_fft=1024 samples
- n_mels=256 samples

The final melspectrogram obtained, with the listed parameters above, leads to a F×T frequency-time representation, having as number of frequency bands $F = n\_mels$ and as number of time frames

$$T = \frac{l * F_s}{hop\_length} = 1291\ samples. \tag{5.1}$$

Before passing the computed mel-spectrogram to the ReNet-50 model, it has to be converted in decibel scale, applying a logarithmic transformation to the squared magnitude of the spectrogram values. Then a normalization scaling process is applied, using a *Min-Max Scaler*. As last operation, due to the fact that ResNet-50 model needs three input channels images, we concatenate three times the spectrogram computed, to create a properly formatted image ready to be processed that has a shape of (3, 256, 1291).

5.2.2. ResNet Pre-Training

Before using the ResNet-50 model to extract audio embeddings, a pre-training operation is performed because this model is originally designed for image classification tasks. By fine-tuning it on an audio dataset, we adapt the model to the specific characteristics and patterns present in audio data. This allows the network to learn relevant features and representations for audio signals.
The task on which we pre-train the model is genre classification, using the dataset GTZAN Genre Dataset [69]. It is one of the most-used public dataset for evaluation in Music Information Retrieval tasks, such as music genre recognition, because it is well labeled and balanced. It contains a total of 1000 audio tracks of 30-seconds duration. The dataset songs can be divided into a total of 10 genres, each one represented by 100 tracks. All the tracks are 22050Hz Mono 16-bit audio files in .wav format.
Due to the fact that the number of songs is low, also to reduce overfitting and to improve the robustness of machine learning models to variations, we use three different techniques to perform data augmentation:

- **Time stretching**: audio processing method that alters the temporal duration of the audio signal, preserving its spectral characteristics intact. It effectively manipulates the playback speed without affecting its underlying spectral content. This strategy is applied using Librosa time stretching function, giving as stretch ratio parameter a value chosen randomly in a range from 0.5 to 1.5.

- **Pitch shifting**: in contrast to time stretching, this technique enables the alteration of spectral characteristics, while preserving the original duration of the track. This manipulation can change the perceived musical key or tone, without affecting the track's length. This strategy is applied using Librosa pitch shifting method, giving as number of shift steps a value chosen randomly in a range from 1 to 5.
- **SpecAugment** [9]: this is a spectrogram data enhancement technique. Concretely, with this method one or more covering bands are set along time and frequency dimension, respectively. The bandwidth is an integer randomly selected and the value of the frequency covered by the covering band is set to 0. However, if the spectrogram is covered by too many bands or the width of bands is too wide, information can be lost. Inspired by [81], we cover only one frequency band, with a maximum bandwidth of 16 samples, and two time bands with a maximum bandwidth of 75 samples.

In each training iteration, two variations are generated for identical samples exhibiting slight distinctions attributed to the stochastic nature of the data augmentation process. Consequently, this approach encourages the model to prioritize the comprehension of time-frequency patterns within the spectrogram environment throughout the training phase.
In conclusion, the ResNet-50 model is pre-trained on the GTZAN genre classification task so, for this reason, we change the number of output neurons of last layer to 10, that is the number of classes represented in the dataset. Before this last layer we add a Dropout layer with probability $p = 0.5$ and a Batch Normalization layer. Other hyper parameters used in the pre-training process are: the *learning rate* to 1e-5, *batch size* to 16, *optimizer* of type Adam with weight decay 1e-1.

5.2.3. System Training Setup

The overall system has the theoretical structure explained in section 4.5. The audio embeddings are computed taking advantage of the ResNet-50 pre-trained network and they are obtained as an array of 1x2048 elements. The text embeddings are calculated through the use of the pre-trained Sentence-BERT. The English-only models use Sentence-BERT *"all-mpnet-base-v2"* while the multi-lingual models use *"multi-qa-mpnet-base-dot-v1"*, according to the languages used to pre-training the sentence transformers. In both cases of English and multi-lingual setup, the lyrics embeddings are returned as an array of 1x768 elements.
Even if this first step of creation of the feature vector is shared, regarding more technical

details, we can distinguish different versions of the final Multi-Layer Perceptron, depending on the dataset we use and on the tasks we are approaching. In fact, we evaluate these different setups:

- **Audio-only setup**, with SPD-English dataset, neglecting lyrics to perform classification task
- **Audio and lyrics basic setup**, with SPD-English dataset considering one single lyrics embedding to perform classification task
- **Audio and lyrics English weighted setup**, with SPD-English dataset and double text embeddings to perform classification and regression tasks
- **Audio and lyrics multi-lingual weighted setup**, with SPD-Multilingual dataset and double text embeddings to perform classification and regression tasks

This distinction is done because for example dealing with a multi-lingual dataset, with a greater number of songs, the complexity of the problem increases, with respect to the case in which the dataset contains just English songs. For this reason, we can not conduct every experiment with the same asset.
These four configurations can be summarized in two principal setups, successively explained, based on the fact that they use the English on the multi-lingual dataset.
All the implementation is done with Pytorch library. For every training procedure, data augmentation is applied following the same approach already introduced in the ResNet-50 pre-training explanation in chapter 5.2.2. In every experimental setup, performing classification task over the four classes of popularity, the loss function used is a CrossEntropyLoss, with mean reduction. *CrossEntropyLoss* is a common loss function used for training classification models, especially in multi-class classification tasks. It combines the Softmax activation function and the negative log-likelihood loss, to calculate the loss between predicted class probabilities and true class labels. Going into detail, this loss function takes as input the predicted class probabilities produced by the model and the true class labels, for each data point in the batch. Internally it applies the Softmax activation function to the predicted class probabilities to normalize the raw scores for each class, converting them into probabilities that sum to 1. After Softmax, the loss function computes the negative log-likelihood loss. For each data point in the batch, it calculates the negative natural logarithm of the predicted probability assigned to the true class label. This penalizes the model more heavily for confidently incorrect predictions.
In regression case instead the loss function used is the Mean Absolute Error that is a metric usually employed to evaluate the system performances and that is explained in chapter 5.3.

As optimizer we have chosen Adam with *learning rate* set to 1e-5 and *weight decay* to 1e-2.

English Models Setup

In case of using the dataset with just English songs, after comparing different MLP architectural structures, we conclude that too complex architectures lead the model to overfit. For this reason, for the experiments on English dataset, the MLP structure chosen is composed by the input layer, one hidden layer and the output layer.

- Input layer: this layer takes n input features, apply them a batch normalization to have all the input features on the same scale, and map them to 512 output neurons. The value of n changes according to the particular setup used:
  - In audio-only setup, lyrics embeddings are not considered, so the input feature vector is composed only by the audio embeddings and the release year. For this reason the input layer takes an array of $n = length(audio\ embeddings) + length(year) = 2048 + 1 = 2049$ elements.
  - In audio and lyrics basic setup, the system takes as input the audio embedding from ResNet-50, the text embeddings from S-BERT and the release year of each songs. For this reason the input layer takes an array of $n = length(audio\ embeddings) + length(text\ embeddings) + length(year) = 2048 + 768 + 1 = 2817$ elements.
  - In audio and lyrics weighted setup, text embeddings gain more relevance, accordingly to the results obtained during experiment 5.4.1. This setup requires the usage of the audio embedding, the text embeddings used twice and the release year. The final input array in this configuration contains $n = length(audio\ embeddings) + 2 * length(text\ embeddings) + length(year) = 2048 + 1536 + 1 = 3585$ elements.
- Hidden layers: these blocks are constituted by linear layers with ReLU activation function. There is a single hidden layer that takes the 512 inputs and maps them into other 128 neurons. Before applying the transformation, regularization and dropout are applied to reduce overfitting.
- Final layer: this layer takes the 128 input features from the hidden layer and maps them to four neurons in case of classification, otherwise in a single neuron if performing regression. In case of classification, each of the four neurons represents one of the popularity class in which the MLP can map a song. From a regression

perspective, the unique neuron uses a Sigmoid activation function with the aim of getting the final prediction in the range [0; 1]. Hence, a normalization procedure is employed to adapt the range of the original popularity value that lies in the interval [0; 100], to the one required by the neural network, in order to compute the loss function.

Multi-lingual Models Setup

In case of multi-lingual dataset the task of predicting popularity is more difficult and for this reason the structure of the architecture gets more complex. In fact, in this MLP structure we have the input layer, three hidden layers and finally the output layer.

- Input layer: this layer takes the 3585 input features, as in audio and lyrics English weighted setup, apply them a batch normalization to have all the input features on the same scale, and map them to 1024 output features. The layer used is a Linear and the activation function is ReLU.
- Hidden layers: this block of layers is constituted by three linear layers each one with ReLU activation function. The first layer takes the 1024 inputs and maps them into other 1024 neurons, the second one takes the 1024 inputs and maps them to 512 neurons, the final one takes the 512 input and maps them to 128 neurons. Between each level regularization and dropout are applied to reduce overfitting.
- Final layer: this layer takes the 128 input features from the last hidden layer and maps them to four neurons in case of classification, otherwise in a single neuron if performing regression. In case of classification, each of the four neurons represents one of the popularity class in which the MLP can map a song. From a regression perspective, the unique neuron uses a Sigmoid activation function with the aim of getting the final prediction in the range [0; 1]. Hence, a normalization procedure is employed to adapt the range of the original popularity value that lies in the interval [0; 100], to the one required by the neural network.

All the configurations summaries are reported in a detailed version in Appendix A.

5.3. Evaluation metrics

We will define the evaluation methods that have been chosen to examine the outcomes of the system. In order to evaluate our model performances during training and validation phase we use different metrics based on the task performed by the neural network. The metrics used are the traditional ones used in Hit Song Prediction, so that we can compare

the performances obtained by our model with other systems.
In case of **regression** the metric we decide to use is the Mean Absolute Error. This metric as been chosen also because it is used in [42] and this allow us to compare our results with the one reported in the paper.
Mean Average Error (MAE), also known as Mean Absolute Error, is a commonly used metric in regression problems due to its interpretability that makes it used as a fundamental tool for assessing the performance of regression models. The calculation of MAE is given by

$$MAE = \frac{1}{n}\sum_{i=1}^{n} |\hat{y}_i - y_i| , \tag{5.2}$$

computing, for each data point i-th in the dataset, the absolute difference between the predicted value $\hat{y}_i$ and the actual target value y_i.
In summary, MAE provides a straightforward way to measure the average magnitude of errors between predicted values and actual target values, offering a clear and intuitive understanding of how well a regression model is performing. Moreover, one of the key advantages of MAE is its robustness to outliers. Unlike other metrics such as Mean Squared Error (MSE), which penalizes larger errors more heavily due to the squaring operation, MAE treats all errors equally.
In fact, Mean Squared Error is used to measure the average of the squared differences between predicted values and actual target values. The computation of MSE is done following this equation:

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(\hat{y}_i - y_i)^2, \tag{5.3}$$

considering $\hat{y}_i$ the predicted value for the i-th data point, y_i the target value for the i-th data sample. In the context of **classification**, various metrics are employed to assess the system's performance, with each measure offering a distinct viewpoint on the system's capabilities. The counts of correct and incorrect predictions are referred to as intermediate test results. Given a certain class c these results can be defined as:

- True positive (TP): it indicates a correct prediction, meaning that the system output and the reference both indicate class c as present.
- True negative (TN): it indicates a correct prediction, meaning that the system output and the reference both indicate class c as not present.
- False positive (FP): it indicates a wrong prediction. The class c is predicted as present while the reference indicates class c as not present.
- False negative (FN): it indicates a wrong prediction. Class c is predicted as not

present, but target indicates class c present.

Our prediction problem is organized as a multi-class problem with 4 classes which a song can belong to. The intermediate test results in a multi-class scenario can be represented as reported in Figure 5.8.

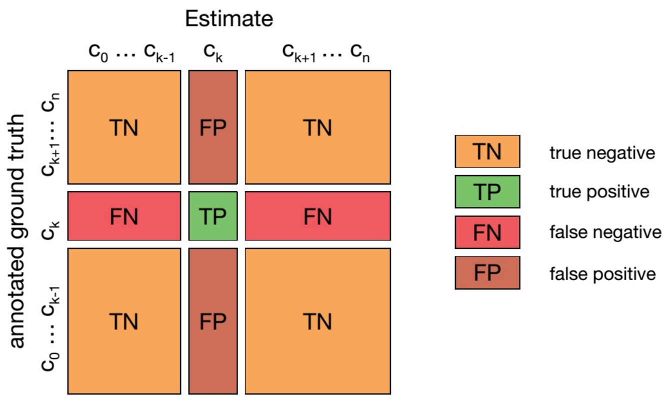


Figure 5.8: Multi-class confusion matrix

Based on the total counts of these intermediate test results, different measures can be derived:

- Precision is the proportion of true positives among all the elements. It is computed as:

$$P = \frac{TP}{TP + FP} \tag{5.4}$$

- Recall is the proportion of positives that are correctly identified among all the real positives. It is computed as:

$$R = \frac{TP}{TP + FN} \tag{5.5}$$

- Accuracy measures how often the classifier makes the correct decision, as the ratio of correct system outputs to total number of outputs. It is computed as:

$$A = \frac{TP + TN}{TP + TN + FP + FN} \tag{5.6}$$

- F1Score is the harmonic mean of precision and recall. It is computed as:

$$F = \frac{2PR}{P + R} \tag{5.7}$$

5.4. Results

In this section we will present the different tests conducted to evaluate the performance of the system and investigate the effectively applicability of our solution in Hit Song Prediction. First of all, to investigate the influence of lyrics embeddings in Hit Song Prediction, we conduct a comparison between the performances of an audio-only based model, with respect to the performances of the same model that also takes in input the lyrics.
After this first step, we move our attention to the system overall results, in two main case scenarios: popularity score and popularity class prediction. Doing this evaluation of classification and regression capabilities of our system, we use as reference system the architecture proposed by Gutierrez in [42] that is, in our knowledge, the state-of-the-art in HSP, using deep learning methods. In their paper, Gutierrez et Al. compare the metrics achieved by their solution with the results of [82]. In [82] the authors propose a set of classifiers to determine the popularity of a song, as a binary classification problem, and some regression models to compute the popularity score. They employ the Million Song Dataset (MSD) [38] which provides a set of meta-data, neglecting raw audio files content or lyrics meaning. They just incorporate additional features such as Bag of Words to take into consideration also lyrics. For this reason we use also [82] as reference.

5.4.1. Audio vs Audio and Lyrics Classification

A first experiment is conducted to investigate the effectiveness of employing lyrics embeddings. Previous researches, conducted in different MIR field, demonstrate the key-role of lyrics in many different tasks, hence we want to prove the positive contribute of text embeddings even in performing music popularity prediction.
In order to conduct this test, we compare the results obtained classifying songs in four popularity classes, using two different sets of features: the first with only audio embeddings and release year; the second including also lyrics embeddings. The two cases of study involve the identical system architecture, a part from the MLP input layer that changes dimension according to the size of the input feature array, as explained in Chapter section 5.2.3. We evaluate the performance in terms of accuracy both during training and validation phase.
As it is shown in Figure 5.9, the validation accuracy envelope of model that takes advantage of both audio and text embeddings reaches an higher accuracy level.

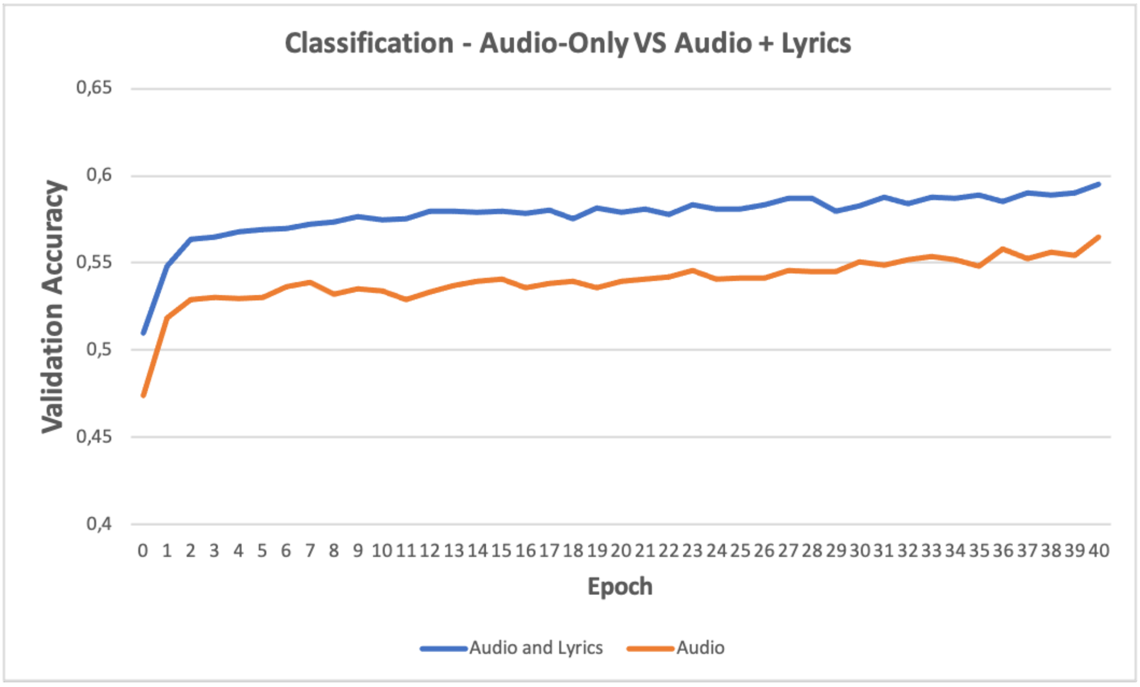


Figure 5.9: Evolution of the accuracy level reached by audio-only and audio and lyrics models during the validation phase performing classification

In particular, after 40 epochs of training, the two observed models obtain the results reported in Table 5.2.

Model	Problem	N° of classes	Dataset	Training Accuracy	Validation Accuracy
Audio-Only	Classification	4	SPD-English	0.532	0.565
Audio and Lyrics	**Classification**	**4**	**SPD-English**	**0.577**	**0.595**

Table 5.2: Comparison among the performance obtained addressing the classification popularity prediction problem considering the employment of only audio embeddings and using both audio and text embeddings

These results align with the literature, because they confirm that a multi-modal approach, that captures both the musical and lyrical aspects of a song, leads to better performance in music related tasks, in this particular case in music popularity prediction.
Furthermore, we demonstrate the effectiveness of Sentence-BERT in capturing semantic characteristic of texts. State-of-the-art models in Hit Song Prediction use only hand-crafted features computed starting from the lyrics. Analyzing this test result, we state the usability of Sentence Transformer approaches also in popularity prediction, a field in which it has never been used. We prove that NLP techniques can be used to analyze lyrics,

to extract sentiment, and to identify patterns in the text that resonate with audiences and contribute to a song popularity.
The demonstrated utility of lyrics in Hit Song Prediction reveals that the subject of lyrics can influence a song popularity, reflecting cultural trends or the spirit of a particular time. Moreover, memorable or catchy phrases in lyrics can become viral and may be shared on social media, further boosting the song's visibility.
Having proved that lyrics embeddings bring to an improvement in the system performance, we investigate the possibility of weighting embeddings.

5.4.2. English vs Multi-lingual Lyrics Classification

In section 5.4.1 we prove the utility and positive impact of text embeddings in HSP tasks. Starting from this result, we evaluate the possibility of weighting embeddings to get the most from their impact on popularity prediction. After performing different attempts, changing the MLP input vector constitution, we decide to use in the next tests the same text embedding twice, with the aim of propagate the textual information through multiple layers, providing more opportunities for the model to learn complex patterns and relationships within the text data. In this way, we are effectively doubling the impact of the textual features on the model performance, hence information contained in lyrics acquire emphasis and extra weight in the model's decision-making process.
In fact, in this test we use a final setup of the MLP, with doubled text embeddings, to evaluate our proposal in a classification task. In particular, we want to compare the results obtained from our model with respect to the ones obtained by reference papers mentioned before, such as [42, 82].
We want to evaluate the performance of our model applied in two different scenarios: processing English-only and multi-lingual songs.
In order to perform this experiment, unlike the state-of-the-art proposals, with our approach we increase the number of popularity classes to four with the aim of adding another intermediate level of popularity, in addition to the third added by [42].

Model	Study	Dataset	N° of classes	Validation Accuracy (%)
MLP	[82]	MSD	2	79.3
HitMusicNet	**[42]**	**SPD**	**3**	**83.03**
Our model	Proposed	SPD-English	4	67.95
Our model	Proposed	SPD-Multilingual	4	70.14

Table 5.3: Comparison among the performance obtained addressing the classification popularity prediction problem considering existing models and the proposed approaches

Analyzing the final results gathered after 100 epochs and reported in Table 5.3, we can see that our proposed solution leads to some positive considerations. First of all, we can notice that the multi-lingual method proposed obtains better results in terms of accuracy with respect to the English-only version. This demonstrates that greater amount of songs influences the results, bridging the performance gap of multi-lingual Sentence-BERT with respect to the one trained only in English.

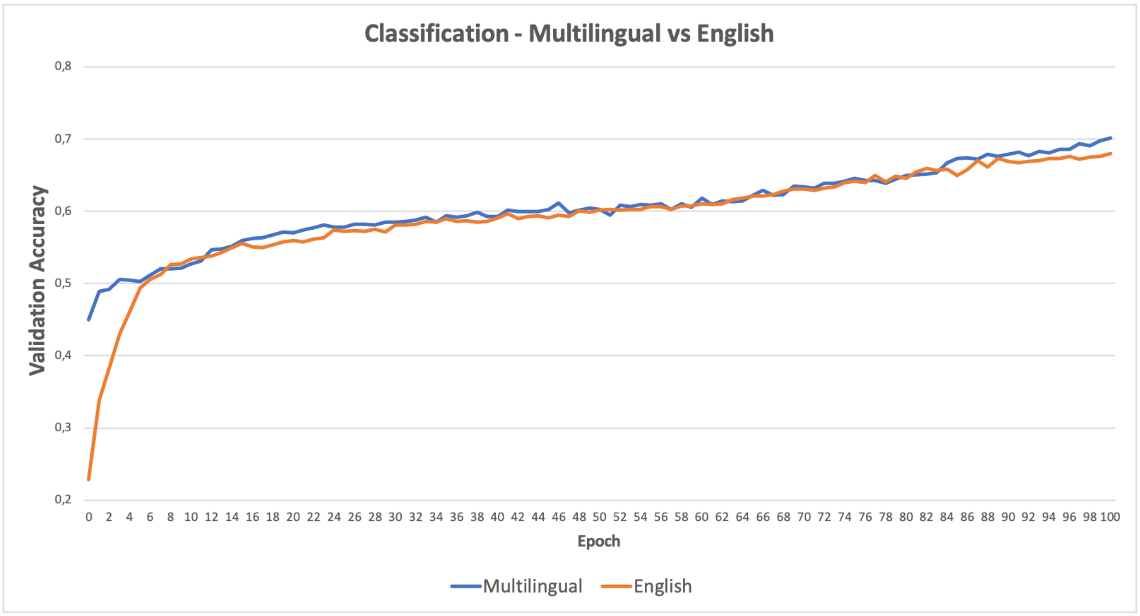


Figure 5.10: Evolution of the accuracy level reached by multi-lingual and English-only models during the validation phase performing classification

In fact, observing the Figure 5.10 we can see how our proposed models follow a similar behaviour until the 80th epoch, after which the multi-lingual one starts to perform better. The trend of the graph suggests that the English-only model keeps improving the accuracy in a constant way, while the multi-lingual one after the 80th epoch has a sharp increase

that leads the model to achieve faster 70% of accuracy.
On the other hand, assessing our results in relation with the ones obatined by [42, 82], we can state that the results are comparable if we consider the situation in which we set ourselves, with the constraints of:

- Augmenting the number of output classes, with respect to the other models evaluated, that increases the complexity of the problem.
- Discarding any metadata, such as the artist's popularity, that [82] identify as one of the most influential features in their dataset. Taking in input only audio preview file, lyrics and release year of each song, in order to keep our solution as unconditioned as possible from data related to popularity, that can affect the prediction.

This result leads us to claim that effectively the combination of audio embeddings, text embeddings and release year in music popularity prediction is a valid approach to be further investigated, since it has different grades of improvement and the resulting accuracy can surely enhance with more data at disposal and epochs considered.

5.4.3. English vs Multi-lingual Lyrics Regression

Using the same input vector configuration used in the previous experiment, we want to compare the results obtained from our models with respect to the results obtained by reference papers, [42, 82], to predict the popularity score of songs. Also performing regression, we compare the results obtained by our model both using English and multi-lingual datasets.

Model	**Study**	**Dataset**	**ML Problem**	**MAE**	**MSE**
MLR	[82]	MSD	Regression	0.1357	0.0184
MLR + Lasso	[82]	MSD	Regression	0.1342	0.0180
HitMusicNet	**[42]**	**SPD**	**Regression**	**0.0855**	**0.0118**
Our model	Proposed	SPD-English	Regression	0.0986	0.0157
Our model	Proposed	SPD-Multilingual	Regression	0.0937	0.0145

Table 5.4: Comparison among the performance obtained addressing the regression popularity prediction problem considering existing models and the proposed approaches

The results of the experimental models in terms of Mean Squared Error and Mean Absolute Error, presented in Table 5.4, show that the proposed model trained using the multi-lingual dataset outperforms the one trained using only English songs. Even in this case, we can see how the number of songs affects the overall performance. Indeed, even

if the multi-lingual Sentence-BERT has an average performance worse than the English one, the results achieved by the multi-lingual system are better with respect to the ones obtained by the English model, that contains half the songs of the other.
Mean Average Error and Mean Squared Error progressions are respectively reported in detail in Figures 5.11 and 5.12, to illustrate the evolution of the results obtained during the validation phase of the test that consists of 60 epochs. The two depicted trends represent the developments of the English-only and the multi-lingual model.

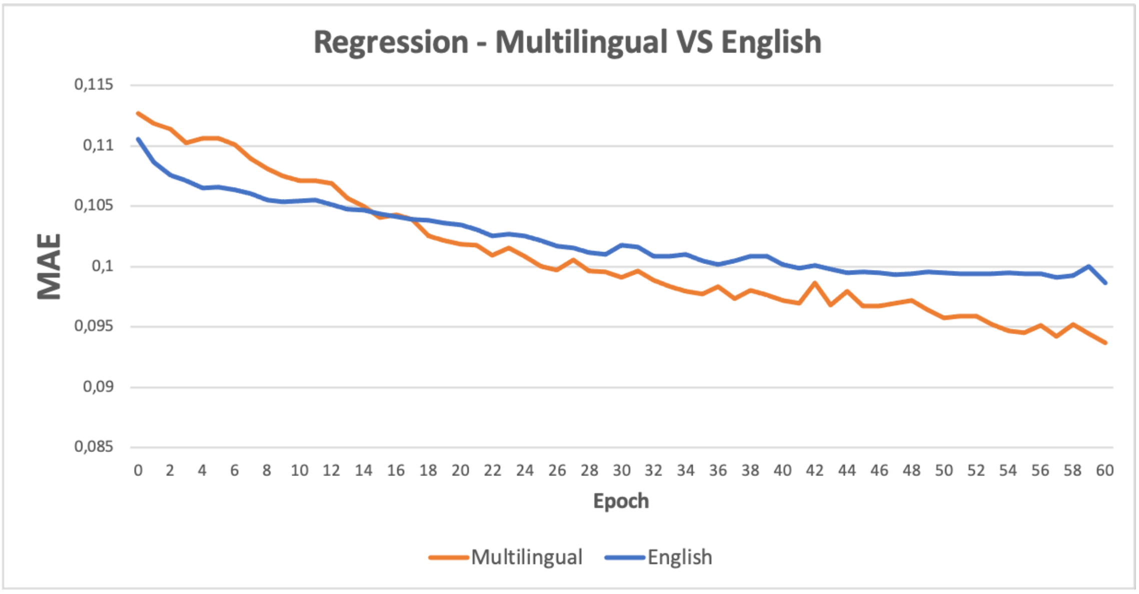


Figure 5.11: Evolution of the Mean Average Error reached by multi-lingual and English-only models during the validation phase, performing regression

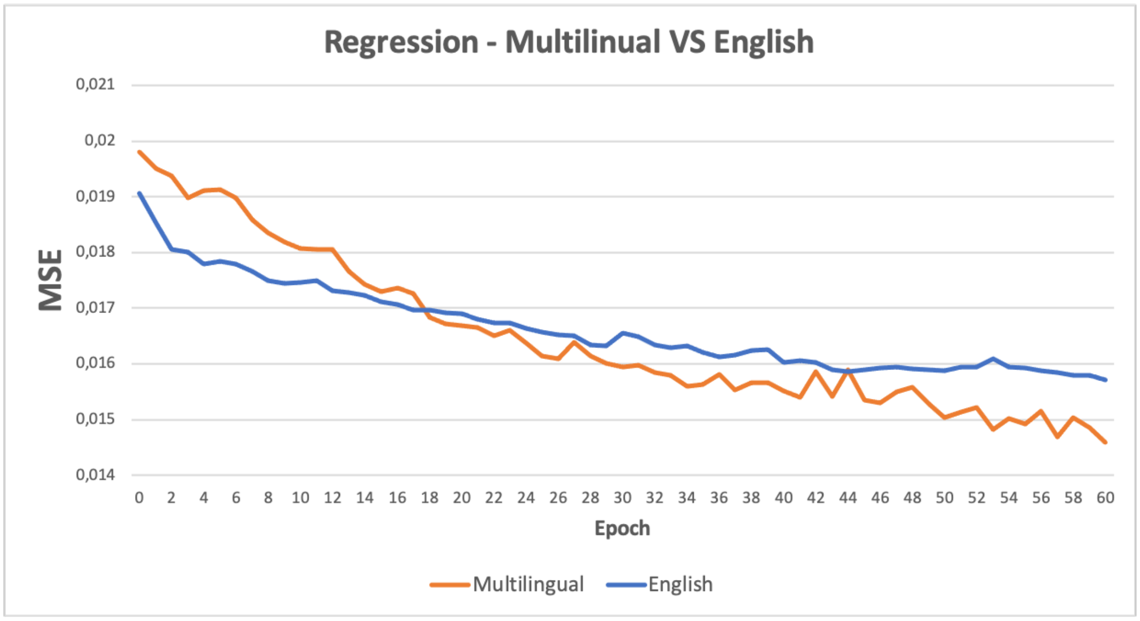


Figure 5.12: Evolution of the Mean Squared Error reached by multi-lingual and English-only models during the validation phase, performing regression

Moreover, the results obtained by both our proposed methods outperform the MAE and MSE values achieved by [82] and are close to the ones obtained by [42]. In this case, there is no mismatch in the number of output prediction classes and we are always neglecting artist popularity. For this reason, considering the MAE and MSE values reached, it is more clear that our results are comparable with the state-of-the-art. In particular, the MAE values achieved by our models are similar to the one achieved by [42], while the MSE values are higher. This observation suggests the possibility of outliers, as the MAE metric tends to handle them more effectively than MSE that is less robust and tends to amplify the effect of large errors.
In conclusion, also evaluating the regression task, we demonstrate the potential of our system in tackling HSP using an architecture based on audio and text embeddings.

5.5. Conclusive Remarks

In this chapter, we have described how we have created two new version of an existing dataset: one with English solely songs and the second one with multi-lingual songs. Successively, the training setup has been exposed in detail. The explanation begins with the initial audio pre-processing phase, extends to the ResNet-50 pre-training procedure, and culminates in a detailed overview of the overall system training process. Subsequently, we assessed the proposed system's performance across classification and regression tasks, presenting the results alongside a comparative analysis with state-of-the-art systems.

6 | Conclusions and Future Developments

In this thesis we have developed an Hit Song Prediction (HSP) system able to predict the popularity of a song both in a classification and a regression perspective, given the relative audio, lyrics and release year. To implement our model we exploited and combined Deep Learning based architectures, taking inspiration both from state-of-the-art in Hit Song Prediction and in other Music Information Retrieval systems.
Traditional HSP systems mainly involve machine learning methods and only in few case they use deep learning architectures. Even when Multi-Layer Perceptrons are used, the input data used are features manually computed from audio and lyrics. Starting from these observations, encouraged by the advancements in deep learning technologies, able to automatically extract features from audio and lyrics, we have introduced a novel approach to address HSP that exploits the usage of embeddings automatically extracted from audio and lyrics. Going into detail, we have employed a pre-trained ResNet-50 model to automatically extract features from audio melspectrograms and a Sentence-BERT transformer to extract text embeddings from lyrics. According to this, we have used only raw audio and text data without taking advantage of other information, such as the artist's popularity, that can bias the resulting popularity predicted. The unique additional data we have considered is the release year of each song to temporally contextualize them and model temporal trends.
Audio embeddings, lyrics embeddings and release year have been fed into a MLP that is responsible for producing the popularity prediction. The MLP returns the resulting popularity in two ways: as a popularity score or as a popularity class based on four ranges of score. In this way we have been able to use our model addressing two different tasks: regression and classification.

Before doing experiments, we have explained into detail how the training configuration has been set with a particular attention to the creation of the dataset used. Starting from the SpotGenTrack dataset, after performing data cleaning, we have built two versions of the dataset according to the language of the songs: an English-only and a multilingual dataset. In order to evaluate the effective applicability of our model in Hit Song Prediction and testing its performances, we have conducted three experiments. The aim of the first experiment is to investigate the influence of text in Hit Song Prediction. For this reason we have compared the accuracy obtained by our model performing classification using only audio embeddings and considering also text embeddings. Results have confirmed the key-role of text in predicting music popularity, according with the literature.

The other two tests have the purpose of proving the popularity prediction ability of our system performing regression and classification. The performances obtained have been compared with the results of the state-of-the-art architectures. In addition, a comparison has been done also between the accuracy value obtained by our proposed method trained on the English-only dataset and the one trained on the multilingual dataset.

Results have shown better results, both in classification and in regression, using the multi-lingual dataset instead of using the English-only one. This have demonstrated that greater amount of songs influences the results, bridging the performance gap of multi-lingual Sentence-BERT with respect to the one trained only in English.

Moreover, results achieved by our models have reached similar values of accuracy with respect to models taken as reference, both in classification and in regression. Results have not outperformed the state-of-the-art performance, but they are definitely conditioned from the data used as input features, because we have not used any information regarding the artist popularity that is one of the most influential feature used by the papers taken as reference. By the way, considering the results obtained, we have demonstrated the potential of a system based on audio and text embeddings to tackle HSP, giving more relevance to the content of songs than to additional metadata.

However, many improvements could be introduced. First of all, in order to get better results, a different ResNet-50 pre-training procedure could be followed, to increase the ability of our model to extract meaningful embeddings from the audio mel-spectrograms. For example we could use a dataset that contains more data with more target classes. In addition, in order to build a 3-channel image we could try to use harmonic-percussive sound separation or other techniques to create three different informative representations of the audio. Moreover, different kind of CNN-based methods can be evaluated, for example the Audio Spectrogram Transformer [83]. Doing this changes the audio representation obtained from our solution could improve, capturing other relevant aspect of audio that are representative of characteristics that have a key-role in songs popularity.

Another improvement could be using a greater dataset for evaluating songs popularity. Starting from Spotify's and MusixMatch's API a dataset specifically designed for this task could be created collecting a more relevant amount of songs with updated popularity scores.

Lastly thinking about the applications of our model, could be interesting to use it in a reverse way with the aim of answering the question "Which are the characteristics of a song that makes it popular?". In order to investigate which are the most important features of audio and lyrics that mostly influence and determine how much a song is appealing for audience.

Bibliography

[1] Mike mccready. `https://en.wikipedia.org/wiki/Mike_McCready_(businessman)`. 2023-08.

[2] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition, 2015.

[3] Nils Reimers and Iryna Gurevych. Sentence-bert: Sentence embeddings using siamese bert-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2019.

[4] Bernard Mulgrew, P. Grant, and John Thompson. Digital signal processing: Concepts and applications. 01 2002.

[5] Nasser Kehtarnavaz. Chapter 7 - frequency domain processing. In Nasser Kehtarnavaz, editor, *Digital Signal Processing System Design (Second Edition)*, pages 175–196. Academic Press, Burlington, second edition edition, 2008.

[6] William A. Yost. Psychoacoustics : A brief historical overview from pythagoras to helmholtz to fletcher to green and swets , a centuries-long historical overview of psychoacoustics. 2015.

[7] C.G.M. Fant. *Analysis and Synthesis of Speech Processes*. North-Holland Publishing Comp., 1967.

[8] Frank Rosenblatt. The Perceptron: A Probabilistic Model for Information Storage and Organization (1958). In *Ideas That Created the Future: Classic Papers of Computer Science*. The MIT Press, 02 2021.

[9] Daniel S. Park, William Chan, Yu Zhang, Chung-Cheng Chiu, Barret Zoph, Ekin D. Cubuk, and Quoc V. Le. SpecAugment: A simple data augmentation method for automatic speech recognition. In *Interspeech 2019*. ISCA, 2019.

[10] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255. Ieee, 2009.

[11] P.M. Nadkarni, L. Ohno-Machado, and WW. Chapman. Natural language processing. *Journal of the American Medical Informatics Association*, page 544–51, 2011.

[12] Prof Chowdhary. *Natural Language Processing*, pages 603–649. 04 2020.

[13] Xinlei Chen and C. Lawrence Zitnick. Learning a recurrent visual representation for image caption generation, 2014.

[14] Xinlei Chen and C. Lawrence Zitnick. Mind's eye: A recurrent visual representation for image caption generation. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2015.

[15] Michael Fell and Caroline Sporleder. Lyrics-based analysis and classification of music. In *Proceedings of COLING 2014, the 25th International Conference on Computational Linguistics: Technical Papers*, pages 620–631, Dublin, Ireland, August 2014. Dublin City University and Association for Computational Linguistics.

[16] V. R Revathy, Anitha S. Pillai, and Fatemah Daneshfar. Lyemobert: Classification of lyrics' emotion and recommendation using a pre-trained model. *Procedia Comput. Sci.*, 218(C):1196–1208, jan 2023.

[17] Rada Mihalcea and Carlo Strapparava. Lyrics, music, and emotions. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 590–599, Jeju Island, Korea, July 2012. Association for Computational Linguistics.

[18] Revanth Akella and Teng-Sheng Moh. Mood classification with lyrics and convnets. In *2019 18th IEEE International Conference On Machine Learning And Applications (ICMLA)*, pages 511–514, 2019.

[19] Yagya Raj Pandeya, Jie You, Bhuwan Bhattarai, and Joonwhoan Lee. Multi-modal, multi-task and multi-label for music genre classification and emotion regression. In *2021 International Conference on Information and Communication Technology Convergence (ICTC)*, pages 1042–1045, 2021.

[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space, 2013.

[21] Ruth Dhanaraj and Beth Logan. Automatic prediction of hit songs. In *Proceedings of the 6th International Society for Music Information Retrieval Conference 2005 (ISMIR 2005)*, page 488–491, 2005.

[22] Thomas Hofmann. Probabilistic latent semantic analysis, 2013.

[23] François Pachet and Pierre Roy. Hit song science is not yet a science. In *Proceedings of the 9th International Society for Music Information Retrieval Conference 2008 (ISMIR 2008)*, pages 355–360, 2008.

[24] Y. Ni, R. Santos-Rodriguez, R. Mcvicar, and T. De Bie. Hit song science once again a science? *4th International Workshop on Machine Learning and Music*, 2011.

[25] Billboard. Billboard charts. `https://www.billboard.com/charts/`.

[26] Spotify. Spotify for developers: Api documentation. `http://developer.spotify.com/documentation/web-api`.

[27] Ioannis Dimolitsas, Spyridon Kantarelis, and Afroditi Fouka. Spothitpy: A study for ml-based song hit prediction using spotify, 2023.

[28] Kai Middlebrook and Kian Sheik. Song hit prediction: Predicting billboard hits using spotify data, 2019.

[29] E. Georgieva, Marcel Şuta, and Nichola S Burton. Hitpredict : Predicting hit songs using spotify data stanford computer science 229 : Machine learning. 2018.

[30] Adewale Adeagbo. Predicting afrobeats hit songs using spotify data, 2020.

[31] R. Rajyashree, Anmol Anand, Yash Soni, and Harshitaa Mahajan. Predicting hit music using midi features and machine learning. In *2018 3rd International Conference on Communication and Electronics Systems (ICCES)*, pages 94–98, 2018.

[32] Mengyisong Zhao, Morgan Harvey, David Cameron, Frank Hopfgartner, and Valerie J. Gillet. An analysis of classification approaches for hit song prediction using engineered metadata features with lyrics and audio features. *iConference*, 2023.

[33] Data.World. Billboard hot-100 songs 2000-2018 with spotify data + lyrics. `http://data.world/typhon/billboard-hot-100-songs-2000-2018-w-spotify-data-lyrics`.

[34] David M. Blei, Andrew Y. Ng, and Michael I. Jordan. Latent dirichlet allocation. *J. Mach. Learn. Res.*, 3(null):993–1022, mar 2003.

[35] L.-C. Yang, S.-Y. Chou, J.-Y. Liu, Y.-H. Yang, and Y.-A. Chen. Revisiting the problem of audio-based hit song prediction using convolutional neural networks. *EEE International Conference on Acoustics,Speech and Signal Processing (ICASSP)*, page 621–625, 2017.

[36] KKBOX Inc. Kkbox inc. `https://www.kkbox.com/jp/ja/`.

[37] Eva Zangerle, Ramona Huber, Michael Vötter, and Yi-Hsuan Yang. Hit Song Prediction: Leveraging Low- and High-Level Audio Features. In *Proceedings of the 20th International Society for Music Information Retrieval Conference 2019 (ISMIR 2019)*, 2019.

[38] Thierry Bertin-Mahieux, Daniel P.W. Ellis, Brian Whitman, and Paul Lamere. The million song dataset. In *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR 2011)*, 2011.

[39] Music Technology Group Universitat Pompeu Fabra. Essentia. `https://essentia.upf.edu`.

[40] Eva Zangerle. Hit song prediction (million song dataset and audio features). `https://zenodo.org/record/3258042#.ZBgaQy9aZpQ`.

[41] Heng-Tze Cheng, Levent Koc, Jeremiah Harmsen, Tal Shaked, Tushar Chandra, Hrishi Aradhye, Glen Anderson, Greg Corrado, Wei Chai, and Mustafa Ispir. Wide deep learning for recommender systems. In *Proc. ACM Workshop on Deep Learning for Recommender Systems*, page 7–10, 2016.

[42] D. Martín-Gutiérrez, G. Hernández Peñaloza, A. Belmonte-Hernández, and F. Álvarez García. A multimodal end-to-end deep learning architecture for music popularity prediction. *IEEE Access*, pages 39361–39374, 2020.

[43] David Martín-Gutiérrez, Gustavo Hernández-Peñaloza, Alberto Belmonte-Hernández, Federico Álvarez, and Mendeley Data. Spotgentrack popularity dataset. `https://data.mendeley.com/datasets/4m2x4zngny`.

[44] Yiwei Ding and Alexander Lerch. Audio embeddings as teachers for music classification. *arXiv preprint arXiv:2306.17424*, 2023.

[45] Maryam Najafabadi, Flavio Villanustre, Taghi Khoshgoftaar, Naeem Seliya, Randall Wald, and Edin Muharemagic. Deep learning applications and challenges in big data analytics. *Journal of Big Data*, 2, 12 2015.

[46] Manjunath Jogin, Mohana Mohana, M Madhulika, G Divya, R Meghana, and S Apoorva. Feature extraction using convolution neural networks (cnn) and deep learning. pages 2319–2323, 05 2018.

[47] Sungho Shin, Jongwon Kim, Yeonguk Yu, Seongju Lee, and Kyoobin Lee. Self-supervised transfer learning from natural images for sound classification. *Applied Sciences*, 11:3043, 03 2021.

[48] Gianluigi Ciocca, Paolo Napoletano, and Raimondo Schettini. *Learning CNN-based Features for Retrieval of Food Images*, pages 426–434. 12 2017.

[49] Dimpy Varshni, Kartik Thakral, Lucky Agarwal, Rahul Nijhawan, and Ankush Mittal. Pneumonia detection using cnn based feature extraction. In *2019 IEEE International Conference on Electrical, Computer and Communication Technologies (ICECCT)*, pages 1–7, 2019.

[50] Abul Abbas Barbhuiya, Ram Kumar Karsh, and Rahul Jain. Cnn based feature extraction and classification for sign language. *Multimedia Tools and Applications*, 80:3051–3069, 2020.

[51] Nima Mahmoudi, Seyed Mohammad Ahadi, and Rahmati Mohammad. Multi-target tracking using cnn-based features: Cnnmtt. *Multimedia Tools and Applications*, 78, 03 2019.

[52] Daniel Grzywczak and Grzegorz Gwardys. Deep image features in music information retrieval. volume 60, pages 187–199, 08 2014.

[53] K. Morita, T. Yano, and K. Tran. Anomalous sound detection using cnn-based features by self supervised learning. Technical report, DCASE 2021 Challenge, 2021.

[54] Keunwoo Choi, György Fazekas, Mark Sandler, and Kyunghyun Cho. Transfer learning for music classification and regression tasks, 2017.

[55] Shawn Hershey, Sourish Chaudhuri, Daniel P. W. Ellis, Jort F. Gemmeke, Aren Jansen, R. Channing Moore, Manoj Plakal, Devin Platt, Rif A. Saurous, Bryan Seybold, Malcolm Slaney, Ron J. Weiss, and Kevin Wilson. Cnn architectures for large-scale audio classification. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 131–135, 2017.

[56] Kamalesh Palanisamy, Dipika Singhania, and Angela Yao. Rethinking cnn models for audio classification, 2020.

[57] Daniel G. Brown Abhishek Singhi. Hit song detection using lyric features alone. *15th International Society for Music Information Retrieval Conference*, 2014.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019.

[59] Revathy Rajendran, Anitha Pillai, and Fatemeh Daneshfar. Lybert: Multi-class classification of lyrics using bidirectional encoder representations from transformers (bert). 03 2022.

[60] Darren Edmonds and João Sedoc. Multi-emotion classification for song lyrics. In *Proceedings of the Eleventh Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis*, pages 221–235, Online, April 2021. Association for Computational Linguistics.

[61] Gaojun Liu and Zhiyuan Tan. Research on multi-modal music emotion classification based on audio and lyirc. In *2020 IEEE 4th Information Technology, Networking, Electronic and Automation Control Conference (ITNEC)*, volume 1, pages 2331–2335, 2020.

[62] Benno Weck, Miguel Pérez Fernández, Holger Kirchhoff, and Xavier Serra. Matching text and audio embeddings: Exploring transfer-learning strategies for language-based audio retrieval, 2022.

[63] Bo-Hsun Sung and Shih-Chieh Wei. Becmer: A fusion model using bert and cnn for music emotion recognition. In *2021 IEEE 22nd International Conference on Information Reuse and Integration for Data Science (IRI)*, pages 437–444, 2021.

[64] Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. Albert: A lite bert for self-supervised learning of language representations, 2020.

[65] Ilaria Manco, Emmanouil Benetos, Elio Quinton, and György Fazekas. Contrastive audio-language learning for music, 2022.

[66] Qingqing Huang, Aren Jansen, Joonseok Lee, Ravi Ganti, Judith Yue Li, and Daniel P. W. Ellis. Mulan: A joint embedding of music audio and natural language, 2022.

[67] Sabyasachi Sahoo. Residual blocks — building blocks of resnet. `https://towardsdatascience.com/residual-blocks-building-blocks-of-resnet-fd90ca15d6ec`. 2023-08.

[68] S. Mukherjee. The annotated resnet-50. `https://towardsdatascience.com/the-annotated-resnet-50-a6c536034758`. 2023-08.

[69] G. Tzanetakis and P. Cook. Musical genre classification of audio signals. *IEEE Transactions on Speech and Audio Processing*, 10(5):293–302, 2002.

[70] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized bert pretraining approach, 2019.

[71] S. Kotamraju. An intuitive explanation of sentence-bert. `https://`

towardsdatascience.com/an-intuitive-explanation-of-sentence-bert-1984d144a868. 2023-08.

[72] Nils Reimers and Iryna Gurevych. Making monolingual sentence embeddings multilingual using knowledge distillation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 11 2020.

[73] S-BERT.net. Sentence-transformer documentation. https://www.sbert.net/index.html. 2023-08.

[74] Spotify. Spotify api: Get track. https://developer.spotify.com/documentation/web-api/reference/get-track. 2023-08.

[75] George Tzanetakis. Gtzan music/speech collection. http://marsyas.info/index.html, 1999.

[76] Nakatani Shuyo. Language detection library for java, 2010.

[77] Musixmatch. Musixmatch api: documentation. https://developer.musixmatch.com/documentation. 2023-08.

[78] Billboard's year-end hot 100. https://github.com/siddgood/billboard-hit-prediction. 2023-08.

[79] Spotipy. https://spotipy.readthedocs.io/en/2.22.1/#. 2023-08.

[80] Librosa. https://librosa.org/. 2023-08.

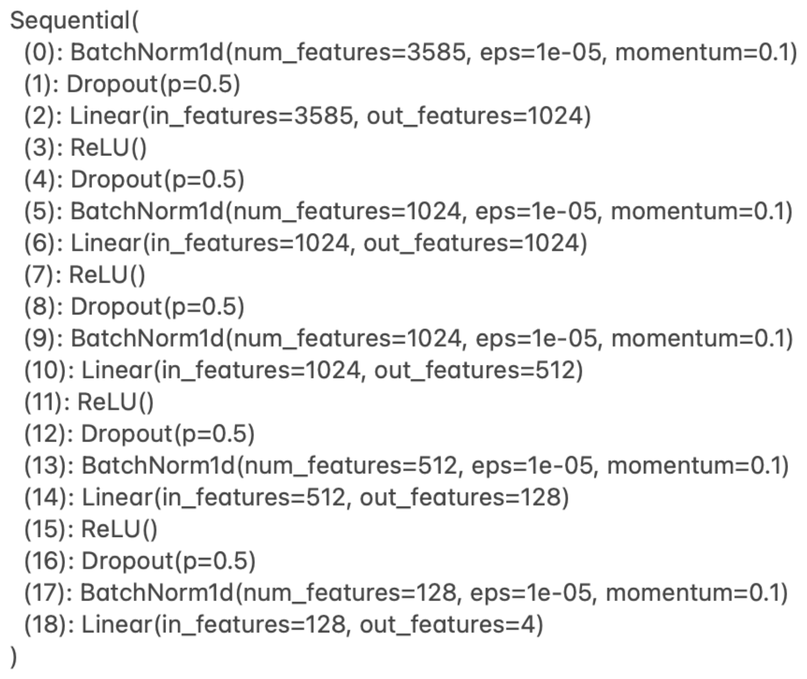
[81] Mengxiang Huang, Mei Wang, Xin Liu, Ruixiang Kan, and Hongbing Qiu. Environmental sound classification framework based on l-mhp features and se-resnet50 network model. *Symmetry*, 15:1045, 05 2023.

[82] James Q. Pham. Predicting song popularity. 2015.

[83] Yuan Gong, Yu-An Chung, and James Glass. Ast: Audio spectrogram transformer, 2021.

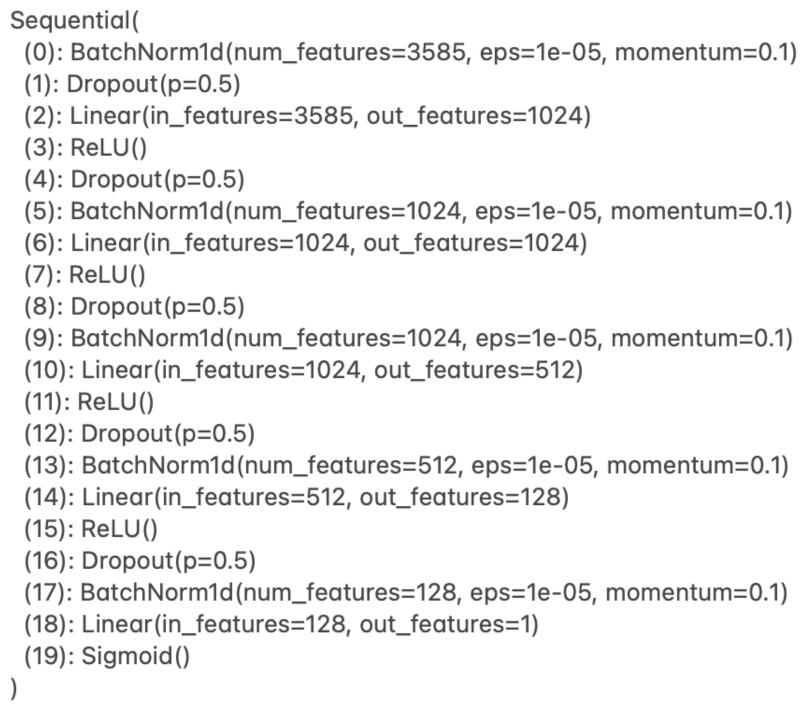
A | Appendix A

Here are reported the summaries of all the Muli-Layer Perceptron configurations implemented in Pytorch to conduct the experimetns proposed



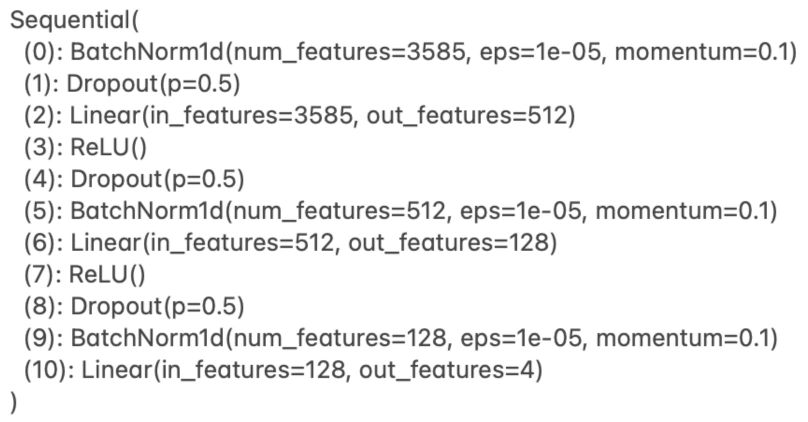
```
Sequential(
  (0): BatchNorm1d(num_features=3585, eps=1e-05, momentum=0.1)
  (1): Dropout(p=0.5)
  (2): Linear(in_features=3585, out_features=1024)
  (3): ReLU()
  (4): Dropout(p=0.5)
  (5): BatchNorm1d(num_features=1024, eps=1e-05, momentum=0.1)
  (6): Linear(in_features=1024, out_features=1024)
  (7): ReLU()
  (8): Dropout(p=0.5)
  (9): BatchNorm1d(num_features=1024, eps=1e-05, momentum=0.1)
  (10): Linear(in_features=1024, out_features=512)
  (11): ReLU()
  (12): Dropout(p=0.5)
  (13): BatchNorm1d(num_features=512, eps=1e-05, momentum=0.1)
  (14): Linear(in_features=512, out_features=128)
  (15): ReLU()
  (16): Dropout(p=0.5)
  (17): BatchNorm1d(num_features=128, eps=1e-05, momentum=0.1)
  (18): Linear(in_features=128, out_features=4)
)
```

Figure A.1: Multi-lingual Classification - Weighted Embeddings



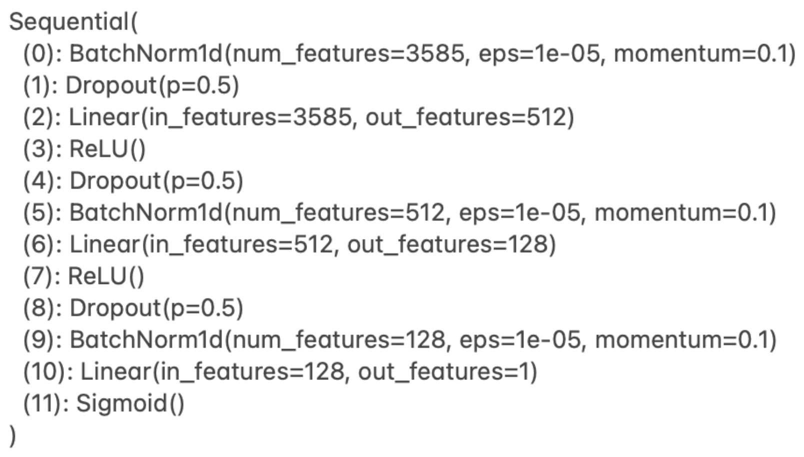
```
Sequential(
  (0): BatchNorm1d(num_features=3585, eps=1e-05, momentum=0.1)
  (1): Dropout(p=0.5)
  (2): Linear(in_features=3585, out_features=1024)
  (3): ReLU()
  (4): Dropout(p=0.5)
  (5): BatchNorm1d(num_features=1024, eps=1e-05, momentum=0.1)
  (6): Linear(in_features=1024, out_features=1024)
  (7): ReLU()
  (8): Dropout(p=0.5)
  (9): BatchNorm1d(num_features=1024, eps=1e-05, momentum=0.1)
  (10): Linear(in_features=1024, out_features=512)
  (11): ReLU()
  (12): Dropout(p=0.5)
  (13): BatchNorm1d(num_features=512, eps=1e-05, momentum=0.1)
  (14): Linear(in_features=512, out_features=128)
  (15): ReLU()
  (16): Dropout(p=0.5)
  (17): BatchNorm1d(num_features=128, eps=1e-05, momentum=0.1)
  (18): Linear(in_features=128, out_features=1)
  (19): Sigmoid()
)
```

Figure A.2: Multi-lingual Regression - Weighted Embeddings



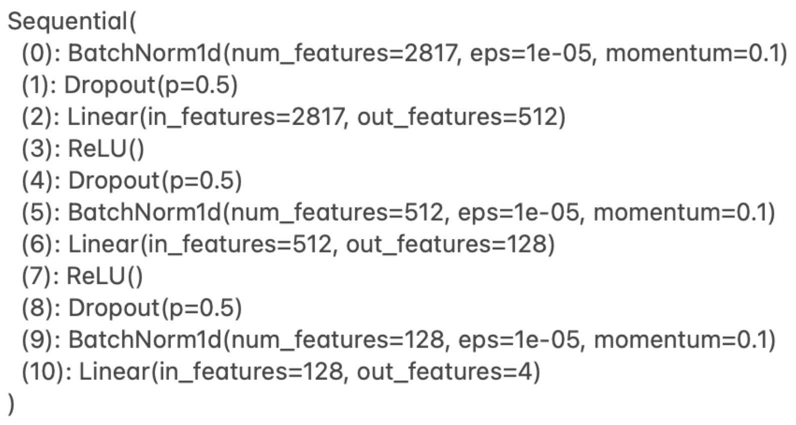
```
Sequential(
  (0): BatchNorm1d(num_features=3585, eps=1e-05, momentum=0.1)
  (1): Dropout(p=0.5)
  (2): Linear(in_features=3585, out_features=512)
  (3): ReLU()
  (4): Dropout(p=0.5)
  (5): BatchNorm1d(num_features=512, eps=1e-05, momentum=0.1)
  (6): Linear(in_features=512, out_features=128)
  (7): ReLU()
  (8): Dropout(p=0.5)
  (9): BatchNorm1d(num_features=128, eps=1e-05, momentum=0.1)
  (10): Linear(in_features=128, out_features=4)
)
```

Figure A.3: English Classification - Weighted Embeddings



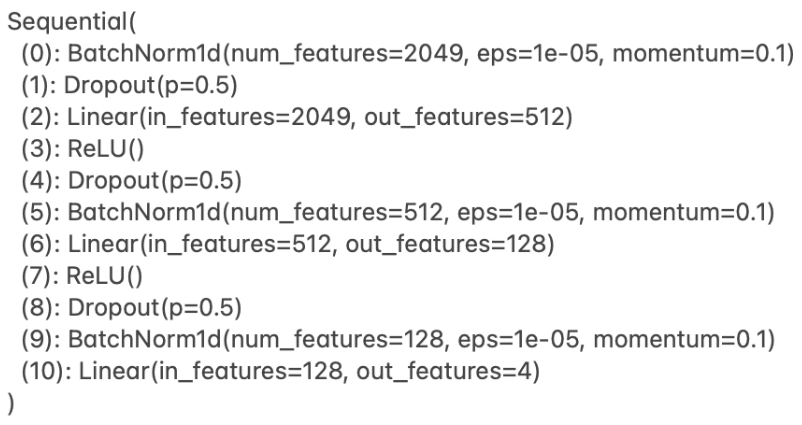
```
Sequential(
  (0): BatchNorm1d(num_features=3585, eps=1e-05, momentum=0.1)
  (1): Dropout(p=0.5)
  (2): Linear(in_features=3585, out_features=512)
  (3): ReLU()
  (4): Dropout(p=0.5)
  (5): BatchNorm1d(num_features=512, eps=1e-05, momentum=0.1)
  (6): Linear(in_features=512, out_features=128)
  (7): ReLU()
  (8): Dropout(p=0.5)
  (9): BatchNorm1d(num_features=128, eps=1e-05, momentum=0.1)
  (10): Linear(in_features=128, out_features=1)
  (11): Sigmoid()
)
```

Figure A.4: English Regression - Weighted Embeddings



```
Sequential(
  (0): BatchNorm1d(num_features=2817, eps=1e-05, momentum=0.1)
  (1): Dropout(p=0.5)
  (2): Linear(in_features=2817, out_features=512)
  (3): ReLU()
  (4): Dropout(p=0.5)
  (5): BatchNorm1d(num_features=512, eps=1e-05, momentum=0.1)
  (6): Linear(in_features=512, out_features=128)
  (7): ReLU()
  (8): Dropout(p=0.5)
  (9): BatchNorm1d(num_features=128, eps=1e-05, momentum=0.1)
  (10): Linear(in_features=128, out_features=4)
)
```

Figure A.5: English Classification - Single text Embedding



```
Sequential(
  (0): BatchNorm1d(num_features=2049, eps=1e-05, momentum=0.1)
  (1): Dropout(p=0.5)
  (2): Linear(in_features=2049, out_features=512)
  (3): ReLU()
  (4): Dropout(p=0.5)
  (5): BatchNorm1d(num_features=512, eps=1e-05, momentum=0.1)
  (6): Linear(in_features=512, out_features=128)
  (7): ReLU()
  (8): Dropout(p=0.5)
  (9): BatchNorm1d(num_features=128, eps=1e-05, momentum=0.1)
  (10): Linear(in_features=128, out_features=4)
)
```

Figure A.6: English Audio Only Classification

List of Figures

List of Tables