POLITECNICO
MILANO 1863

SCUOLA DI INGEGNERIA INDUSTRIALE
E DELL'INFORMAZIONE

Executive Summary of the Thesis

A Music Similarity Metric Space to study the Evolution of Music Trends

Laurea Magistrale in Music and acoustic Engineering - Ingegneria Acustica e Musicale

Author: Nicoletta Brundo

Advisor: Prof. Massimiliano Zanoni

Co-advisor: Giorgio Arcella

Academic year: 2020-2021

---

1. Abstract

The aim of this research is to create a tool to support musicologists in tracing and studying the evolution of music trends through years. For this purpose, we create an Euclidean music similarity metric space employing Transformers architectures, which currently hold the state of the art in Natural Language Processing tasks, and the triplet loss function. In addition, we create a graphic interface to support the visualization of music pieces in a 2D space. Experiments reveal that Transformer architecture over−performed with respect to the CNN. We test our models both with algorithm metrics and human evaluations provided by music experts. At the end of the study, our model show an evolutionary trends of popular music in Italy in a range of year between 2016 to 2021.

2. Introduction

Digital revolution has brought significant changes in the music production world, as it allows to reduce the cost and the space required for the electronic hardware. The main consequence of this phenomena is that, today, an increasing number of people produce music. This revolution is leading to a faster evolution of the music much more that it happens in the past. Furthermore, people's music tastes have been evolving through years in parallel to music styles. Similarly, music styles are influenced by popular customs and social changes. As the number of music sub-genres is growing more and more, we believe that music genre classification can not be as exhaustive as it would be in the past years. Instead, we believe that the categorization of music should mainly relies on similarity. For this purpose, we create a similarity metric space in which similar songs are located in near-by regions, while dissimilar one are far away from each other. Music similarity literature is vast, in MIREX context 'Audio Music Similarity' task has been run from 2006. Most of that works rely on Machine Learning classifier, hand crafted features and human evaluations. Today, new Deep Leaning techniques leads to much performative algorithms. Many works exploit triplet loss function with Artificial Neural Network, as the Multi-Layer Perceptron or CNN, using perceptive features mixed with hand-crafted features. Similarly, the problem of the evolution of music trends has been treated by many other works. Some of them try to capture the genre

trends predictions through the influence of music in time and space, with the beliefs that an Artist's musical influence is not limited to just a genre, but also to the others. Others treat the problem of music prediction employing LSTM architectures, using the song playback volumes, collection volumes and download of music by several artists. In this thesis, we presents a novel approach based on Transformer [4] architecture, which exploits triplet loss function and employ perceptual features as music descriptors.

3. Features and Datasets

In this section we give an overview of the perceptive features and the datasets we employ for our research.

3.1. Perceptual features

Music similarity can be described under several dimensions including sound, harmony, melody, rhythm and growth. What is more, it can be related to psychological, cultural, perceptual aspects of human life and differently perceived by everyone due to the non-linear humans perception of sound. For this reasons, we leave the comprehension of all the high level categories to deep learning model, while employing low level features to describe the logarithmic human perception of sound. For this thesis we extract five perceptual features, which are able to capture different aspects of music. They are MFCC, MFCC delta, CQT, MEL and Chroma coefficients, for a complete features vector of 324 music descriptors per 130 time frames.

3.2. Dataset

A dataset must satisfy a certain number of requirements. It must provides a number of balanced examples which must cover all the relevant classes of interest, it must be large enough to avoid over−training and to effectively learn models incorporating inconsistencies in the data. It also should be free licensing and provides raw audio data. Consequently, we choose 'Free Music Archive' library, as it satisfies those requirements and also provides metadata related to title, artist, genre and year. It contains eight balanced genres for a total amount of 8000 songs. To test our model, we also select six playlists from Spotify, which collect all the most popular and listened Italian songs from 2016 to 2021. Spotify's API allow us to download 30 seconds of audio per tracks, for a total amount of 500 songs.

4. Networks and Triplet loss function

In this section we present the model architectures we employed for the experiments and the implementation of triplet loss function.

4.1. Convolutional Neural Network

We build a Convolutional Neural Network composed of three convolutional layers with 64, 128 and 256 number of kernels each. After every convolutional layer, we insert a Max Pooling 2D layer, to downsample the input by a factor of 3, and a Batch Normalization layer, to prevent overfitting in the training stage. Then, we insert a flatten layer to reduce the dimensionality of data and add a dropout layer for more regularization. The input is then feed to a dense layer with 'Relu' activation and a final L2 normalization layer, to restrict the output embedding vectors inside a unit hypersphere.

4.2. Transformers

Transformers are architectures which over−performed with respect to the past recurrent models. The reasons for that are due to the ability of processing long sequence of input data at the same time. While past recurrent models processed their input sequentially, thus requiring expensive memory, Transformers process data in parallel at the same time. The transformer acts as it does not know the temporal dimension of data, so positional encoding is added to the input before being processed. The greatest power of Transformer is the self-attention mechanism. This allows to create weighted connections among the input data and to focus the attention on the most relevant ones. Thanks to the multi-head attention layer, Transformers are also able to 'see' different representations of the same input at the same time and elaborate them all together, requiring the same memory as it would use to process a single representation of the input.
Our Transformer includes the encoder structure only. This is the only necessary structure for the purpose of our thesis, as we don't want to generate data, but instead learn a similarity metric

space. The encoder has the same structure of the one presented in the original paper [4]. In such structure, we remove the last Dense layer with Softmax activation function. Instead, we add a Dropout layer to prevent overfitting, we add a 1D Global Average Pooling to downsample the input, we insert a Dense Layer and finally, as for the CNN structure, we add a Lambda layer to perform the L2 normalization. In our best model, we set the number of encoder layers to 12, the number of heads to 27, the number of units neurons in the dense layer to 1024 and the embedding vectors output to 1024. We introduce a linear activation function in the first Dense layer before adding the positional encoding to the input, increasing the performance of the model of the 17 percent, as we learned from [2].

4.3. Triplet loss

Triplet loss is the technique we employ for the generation of the similarity metrics space. We select from the datasets triplets of samples in order to have two very similar samples, the anchor and the positive sample, and one very dissimilar from the first one, the negative sample. Triplet loss minimize the Euclidean distance between the anchor and the positive sample, while increasing the one of the negative with respect to the anchor. The following Formula shows the loss function for a single triplet:

$$\{L = \max(0, D(f_a - f_p) - D(f_a - f_n) + \alpha), \tag{1a}$$

where:

- f_a is the embedding vector for the anchor sample;
- f_p is the embedding vector for the positive sample;
- f_n is the embedding vector for the negative sample;
- '*alpha*' is the margin, a value which makes sure that the network is not allowed to output the insignificant solution where all embedding's vectors are zero or contain the same values;
- D is the metrics to compute the distance, which in our case is the Euclidean distance.

In our research, we implement a triplets mining based on three different labels, which are the genre, the artist and the track id label. The reasons which lead us to make this choice are simple. A song is more similar to itself than to any other song. Two songs by the same artists or of the same genre are more similar than songs by other artists or others genres. The We implement the triplet loss layer exploiting Siamese Networks. These networks work in parallel while sharing the same weights. Given a base model, which could be for example one of the architectures mention in 4.1 or 4.2, we compile a new model with three new input layers, each corresponding to one of the samples in a triplets.

5. Graphic Interface

To support the study of the evolution of music trends, we create a graphic interface, depicted in Figure 1. It provides very basic operations, which revealed to be extremely useful in tracing trends. The interface shows all the music pieces by 'Free Music Archive' datasets, which are represented as points in space, where each color corresponds to a particular genre. Black points, instead, represents Spotify's tracks. A slider on the top left of the window allows to visualize songs by year. Clicking on a point song, information relative to title, artist, genre and year are illustrated in a info box on the bottom right of the window. A heat map allows the visualization of the space in which the color and the color's intensity are parameters related to the genre and to the concentration of the embedding vectors presented in a certain region. There is the possibility to shows Spotify songs over the heat map. Clicking a point and and pressing 'i' on the keyboard, neighbours of that point are shown over the heat map.

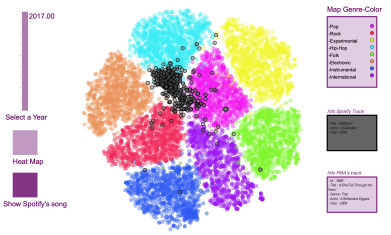

Figure 1: The figure shows the graphic interface. Colored points are music tracks from FMA datasets, while black points represent Spotify's music tracks.

6. Experiments and results

In this section we describe the four experiments we conduct with both architectures. Then, we introduce the metrics used for the final evaluations and finally, we show the results.

6.1. Experiments

We decide to conduct four different experiments to test the performances of our model with different settings. While we maintain the CNN structure the same for all the tests, in the Transformer case, we experiments different model complexities and add new layers in the final test, to show the improvements with respect to the first scenario.
In a baseline approach, we feed the networks a training set with examples including only 20MFCC features by 130 time frames. In this first test, we mine the triplets from the dataset according to the only genre label. In this way, each triplet has the anchor and the positive sample sharing the same genre, while the negative sample having a different genre.
In a complex approach, the experiments are conduct employing all the complete features vectors with 324 descriptors by 130 time frames. In this second scenario, the triplets selection is performed according to three different labels, the genre, the artist and track id label. In the training process, each label is given a probability of being selected at each time step. A batch of triplets at each time step is generated according to the select labels. We notice that track id label notably reduce the computational time in the training process, act as a similarity regularization and make the loss decreasing faster, as we learned from [1].

6.2. Metric

To evaluate the performance of the models we employ a machine learning classifier, the 'K-nearest neighbours', with 'ball tree' algorithm. First, we predict with our trained model (CNN or Transformer), the embedding vectors of both the training and test set. Than we fit the 'K-nearest neighbours' classifier with the predicted training set, with the corresponding true labels. Then we use this classifier to predict the genre labels of the test set and count how many of the predicted labels are equal to the true labels. The accuracy is calculated as the number of correct predictions divided the test set size.

6.3. Results of the four Experiments

Table 1 shows the results of the four experiments. The CNN is very performative in learning a simple metric space based only on MFCC coefficients, but its accuracy decreases when it deals with the complete features vectors. Transformer, instead, shows very poor performances in this first experiments. This is due to the fact that Transformers are architectures which works best with longer inputs data. Furthermore, in the special case of audio signal, acoustic events in adjacent frames are very similar to each other. So, in this case, Transformer shows its weakness in focusing the attention on such short input sequences of similar events. To the contrary, in the second experiment, it leads to very good results, after we double the dimensionality of the features from 324 to 648, following the guidelines of [3]. We reshape the matrix according to this Formula:

$$\left\{ X \in R^{(lxd)} -> reshape X \in R^{(\frac{l}{a} xad)} \right. \quad (2a)$$

We think that music cannot be described just by a single feature, so 20 MFCC are little informative. Multiple features, instead, are able to capture different and relevant aspects of the same song. For these reasons, we think that the results shown by the Transformer are much performative than the CNN.

Final results of the four experiments

	Baseline app.	Complex app.
CNN	90	75
Transformer	56	91

Table 1: Accuracy of of the models in both tests and with both networks.

7. Humans evaluations

We ask three experts to evaluate the performance of our model. We submit to them a simple test composed of six questions to which they have to answer giving a numerical score in a range between 1 and 10, where 1 is the worst evaluation and 10 is the best evaluation.

For each participants, twenty randomly songs are selected from Spotify's playlists, for a total amount of 60 songs. The first three questions are concern the absolute position in space of a query song, the similarity of its neighbours and the proximity of its location with respect to the right cluster. The final three questions concern about to the overall performances of the systems. We ask the tester how useful is the visualization by year, how much is the ability of the system in showing trends and how performative is the overall system.

7.1. Results of Human evaluations

In table 2, we can see the final results of the first three questions. We compute the mean of the three evaluations given by the participants for each song and for each question. Then, we compute the mean of the mean twenty values for each question. The results are shown in the first rows of the table. In second row, instead, we show the variance of the mean of the three evaluations for each song and for each question. In table 3 are shown all the evaluations given by the each participant for the last three questions concerning the overall performances of the system, plus the mean and the variance.

Final results of the first three questions

	Q1	Q2	Q3
Mean	7,2	6,3	7,1
Var	0,61	0,82	0,49

Table 2: Mean and variance of the evaluations given for the first three questions by the experts.

Final results of the last three questions

	Q4	Q5	Q6
V1	4	7	7
V2	7	8	8
V3	8	7	8
Mean	6,3	7,3	7,6
Var	2.9	0.2	0.2

Table 3: All the scores, the mean and variance of the evaluations given for the last three questions by the experts (V1, V2, V3).

The first question gain a 7,2 mean value and 0.61 score for the variance. According to this result, our model classifies almost correctly the absolute positions of the points songs. The second question obtain mean value of 6.3 score and very high variance. This result reveal one of the main weakness of our model which can a consequence of FMA issues, in which the compared datasets show very different characteristics. The third question reaches the mean value of 7,1 and lower variance, which means that all the scores are concentrated around that value. In this case, our model seems to be more performative, as the songs are located in proximity of the correct clusters. The result of the fourth question is the lowest with respect to all the others, which is 6,3. The visualization per year of the FMA tracks lucks of a sufficient number of examples per year, so comparing a query song with neighbours of different year results to be hard. The fifth question obtain a final mean vote of 7.3 with a very low variance. This means that the evaluations were consistent in agreeing that this tool can well visualizes trends. The results of the last question of the test test reveal a 7,6 score on the overall performances of the system. This is a good results, even though it appears to be lower compared with what we expect from the first metric evaluation. The reasons for that could mainly be address to the characteristics and limits of FMA datasets. A good point to notice is that, all the Spotify's playlists include the most popular tracks in Italy from 2016 to 2021 and the most popular genres in Italy in these years, including 'trap', 'indie', 'Pop', 'drill'. The Embedding vector points generated by our model are

predicted in the exact position of space where we expect they would be. In fact, they are organized inside a region of space between Hip-Hop, Pop and Rock.

7.2. Tracking music trends

We plot the songs by Spotify's playlists, year by year, over all the embedding points of the FMA dataset. From this plot, we trace an evolutionary trend of music. Playlists contain the most popular songs in Italy from 2016 till 2021, so classifiable as Italian Pop music. In our interface, black points, which represent them, gradually move from the violet Pop cluster to the light-blue Hip-Hop cluster. This fact could be addresses to the advent of the trap music in Italy in 2016. Actually, 2016 was the year in which trap music reached the maximum popularity in Italy, enjoying great success among the audience. If we look back in years before 2010, Pop music was mainly based on songwriting folk music. Later, the influence of American trap music affected the Italian artist, thus having consequence on Pop music.

8. Conclusions

In this work we employed Deep Neural Architecture to study the evolution of music trends through years. We employed both a 'Convolutional Neural Network' and a 'Transformer' architecture, with in addition the triplet loss function, to generate a music similarity metric space based on Euclidean Distance. In such a space songs in near-by regions share similar characteristics, while dissimilar ones are far away from each others. We conducted four experiments with both architecture with a baseline and a complex approach, showing very different results. In the complex approach Transformers outperformed over the CNN reaching 91 percent of accuracy. A graphic interface, created for the purpose of this research, was used to show the output embedding vectors of our best model and to conduct human evaluations of the performances. At the end of the study, we was able to traced a music trends of the popular music in Italy which moves from the Pop genre to the Hip-Hop genre. In futures works we will implement unsupervised techniques for different triplet mining strategies in order to be able to make use of larger and unlabeled datasets.

9. Acknowledgements

Vorrei ringraziare il professore Massimiliano Zanoni, Riccardo Diviggiano, Giorgio Arcella e la sua etichetta, per i consigli e la disponibilità concessa per rendere possibile questo lavoro di tesi.

References

[1] Jongpil Lee, Nicholas J Bryan, Justin Salamon, Zeyu Jin, and Juhan Nam. Disentangled multidimensional metric learning for music similarity. In *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6–10. IEEE, 2020.

[2] Ngoc-Quan Pham, Thai-Son Nguyen, Jan Niehues, Markus Müller, Sebastian Stüker, and Alexander Waibel. Very deep self-attention networks for end-to-end speech recognition. *arXiv preprint arXiv:1904.13377*, 2019.

[3] Matthias Sperber, Jan Niehues, Graham Neubig, Sebastian Stüker, and Alex Waibel. Self-attentional acoustic models. *arXiv preprint arXiv:1803.09519*, 2018.

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in neural information processing systems*, pages 5998–6008, 2017.

Dipartimento di Elettronica, Informazione e Bioingegneria

Master Degree in Music and Acoustic Engineering

A Music Similarity Metric Space to study the Evolution of Music Trends

by:
Nicoletta Brundo

matr.:
913118

Supervisor: Prof. Massimiliano Zanoni

Co-supervisor: Dr. Giorgio Arcella

Academic Year
2020-2021

Abstract

New digital technologies completely revolutionized the world of music production and made it accessible to almost everyone, expanding the possibilities of the creativity process and giving the opportunity of a low cost self-promotion on the social networks. The main consequence of this revolution is the increasing numbers of new musical 'trends', which are hardly classifiable into just a single musical genre and make the identification of new musical 'sub-genres' almost impossible.

From this perspective, it arises the need of changing the point of view and look for a new way of identifying music trends, to let the music being free from genres boundaries though still giving a clear idea of the songs style. The concept of similarity naturally encounters this need, as it might be disjoint by the concept of 'genre'.

In this study we address the problem of creating a similarity metric space to trace how music evolves through years and what genres are more influenced by other genres or by new emergent artists, as they gain popularity, giving rise to new 'musical trends'.

For this purpose, we propose a novel approach based on Transformer architecture with triplet loss function to generate a music similarity metric space and a graphic interface to visualize the results and to identifying the music trends. Other precedent works focused their attention on music similarity, but, at the best of our knowledge, no one of them exploit music similarity to study the evolution of music trends through years, employing perceptual features as input to the current state of the art Transformer architecture.

Sommario

L'avvento del digitale nel mondo della musica sta spostando lentamene il baricentro della produzione musicale dai grandi studi di registrazione alle case di nuovi piccoli artisti emergenti. La principale conseguenza di questo fenomeno è la nascita di sempre più numerose nuove 'sonorità' musicali, difficilmente incorniciabili in un certo genere musicale, che rendono faticosa, ed in un qualche modo inutile, la definizione di altrettanti nuovi 'sottogeneri'.

Sotto questa prospettiva nasce l'esigenza di rompere i confini di genere ed esplorare spazi più liberi nel campo della classificazione musicale: il concetto di similarità risponde naturalmente a questa esigenza, evitando di 'etichettare' un certo brano musicale, ma piuttosto di collocarlo in uno spazio metrico in cui brani vicini presentano una qualche somiglianza. Il concetto di similarità risponde naturalmente a questa necessità essendo idealmente distaccata dal concetto di genere.

In questo studio diamo uno panoramica dell'evoluzione della musica negli anni, evidenziando come alcuni generi musicali siano fortemente influenzabili da altri generi, nonché da nuovi artisti emergenti, dando origine a nuove 'tendenze musicali.

L'architettura proposta in questo studio è un Transformer, che tramite la funzione di costo triplet loss e l'utilizzo di descrittori percettivi del suono, è impiegato allo scopo di creare uno spazio metrico Euclideo di similarità musicale. Uno spazio grafico 2D ci permette di visualizzare l'evoluzione della musica negli anni e di capire come un certo genere sia influenzato da altri generi o da artisti emergenti, dando dunque vita alla scia di nuovi trend musicali. Altri lavori precedenti hanno focaliz-

zato la loro attenzione sulla similarità musicale, ma nessuno di questi sull'evoluzione della musica nel tempo con i Transformer.

Ringraziamenti

La mia lista di ringraziamenti sarà lunga e sentita, dato che davvero sono tante le persone hanno dato un contributo importante per far si che io oggi possa esser giunta fin qua. Vorrei ringraziare Ouahid, che è propabilmente una delle persone a cui più voglio bene al mondo, che in particolare in questi ultimi due anni si è rivelato essere un amico speciale, che ha saputo darmi la forza e la calma in sistuazioni in cui chiunque sarebbe crollato. Vorrei ringraziare Cristina Salemi, una donna spettacolare come veramente poche se ne trovano al mondo, che mi ha praticamente raccolto da terra e rimessa in piedi la scorsa estate, in un momento in cui nessun altro ne sarebbe stato assolutamente capace. Vorreri ringraziare Gaetano, che è stato forse uno tra i pochi che è riuscito a regalarmi momenti di sorrisi e leggerezza, allontanandomi da tutti i miei problemi senza che io me ne rendessi nemmeno conto. Vorrei ringraziare la mia amica Carmela, che seppur incontrata da un tempo relativamente breve, si è rivelata una persona con un cuore enorme e tenace come nessun altro nel sopportarmi, rimasta sempre con me nei momenti felici, ma che ha avuto il coraggio di restare anche in ogni giorno grigio. Vorrei ringraziare la mia cagnolina Nina, arrivata esattamente un mese dopo la scomparsa di mia madre, che mi ha regalato così all'improvviso un amore infinito e coccole, ricordandomi tante volte quanto la vita sia più semplice di quella che appare, oltre che farmi rendere conto di quanto io fossi ormai incapace di fermarmi e riposare. Vorrei ringraziare Clara Assenza, che non si è mai dimenticata di pensarmi e preoccuparsi e di sapere di me e

Marta che mi ha sempre ricordata che i problemi non li avrei affrontata da sola. Il professore Zanoni che mi ha avuto la pazienza di ascoltarmi, di capirmi e soprattutto di riuscire a convincermi nel proseguire il percorso per completare una tesi, invece che una tesina. Jacopo Gino per i mille consigli e le mille chicche di conoscenza reglate a tutte le ore del giorno senza esitare mai. Giorgio Arcella e Riccardo Diviggiano che hanno arricchito questa tesi con la loro conoscenza in campo musicale e la loro disponibilità nel concedermi il loro tempo. Vorrei ringraziare la mia famiglia. Mia sorella Vincenza, mi ha praticamente nutrita per due anni, dato che altrimenti non avrei nemmeno mangiato. Antonio che mi ha invogliato sempre a fare un sacco di cose e dato l'esempio di quanto bisogna lavorare nella vita per ottenere risultati. Mary che mi ha incoraggiata innumerevoli volta a seguire le mie passioni, ricordandomi quanto sia importante farlo e la piccola Cate, che mi ha riportato tante volte indietro nel tempo alla mia infanzia, con soli semplici piccoli gesti. Ed infine i più importanti di tutti, la mia Mamma e e il mio Papà, che mi hanno fatta bellissima e mi hanno guidata nei miei studi, insegnandomi che la libertà di un individuo deriva dalla sua conoscenza e insegnato le cose veramente importanti della vita per cui vale la pena combattere.

Nicoletta

Contents

List of Figures

List of Tables

Introduction

Many producers, which I had the pleasure to met during my musical experiences in different recording studios and music laboratories, shared to me how laborious it was to record and produce music with the analog technologies in past years. It was a long and demanding work, in which each mistake cost a lot of money and time. However today, the advent of digital technologies is revolutionizing the music production rules and practises. Very cheap hardware, which can almost simulate the relative analog ones, have the advantage of being portable with a considerable size reduction.

A large number of these devices are midi controllers, which are designed to work with their own software, though they can be also configured to allow the control of the functionality of other ones. This is a grate advantage as it grant musicians the access to an unlimited number of digital synthesizers and effects, just by holding a single controller and a personal computer. There is the possibility to choose between an infinite list of VSTs but also to design personal sounds with tools like 'Max Msp', thus enormously expanding the possibility of the creativity process. Digital Audio Workstations grant the integration of different plugins in a single platform, facilitate the recording session as well as simplify each step of the signal processing chain. They also offer all the tools needed for the mixing and mastering process such as equalizers, compressors, limiters and so many others electronic hardware which are completely replaced by software. The trend allows music production studies to reduce required space, so that small rooms are potentially enough. For those reasons, the on going digital revolution encourages young artists to start a career in the music industry much more than it happens in the past.

This phenomenon has been slowly growing through years and today a larger number of artists can self-produce their own music. Self-promotion is also catching on in the society more and more. Social media, like Instagram or Tik-Tok and the new music platforms, like Spotify or Tindal, make possible to share music freely or at a very low price, among the public, focusing the attention not only to the artist's music style, but also to the artist's image itself. Today artists become as influencers and their music acquires more relevance if people tend to follow them. For this reason, artists might also become trends setters.

But what is the impact of this revolution in the evolution of music?

Today, the categorization of music based on genre is no more explicative as it would have been years ago. It is unlikely to happen that asking to some musician the genre he/she produces, answers like pop, electronic or classical, would give an exhaustive definition of the style of the music produced. Today, a real distinction between genres is not feasible anymore. People travel, people share ideas, people are strictly connected to each other, distances are reduced, so contamination is a commonplace. Moreover, there is the propensity of any aspiring producers to investigate and explore new exclusive 'sounds', on one hand, for a personal research, and on the other hand, to have a chance to emerge among all of the other artists and being more 'identifiable'. In this context, the investigation of new directions that music will pursue and maybe what trends will be popular in the future is a under-the-spot topic.

For everything said so far, music cannot be clustered anymore in rigid borders. Music is not more just belonging to a genre'. Instead, what we can better do is to think to music as located somewhere in an imaginary space. In such a space, all of the music's pieces which are similar, which means that they have something in common or share characteristics which make them resemble one with each other, are closed. Greatest aggregations of similar songs are trends. Given information on the released years for the songs allows, in the space, to also track the evolution of music trends through time. These could be used also with the aim of searching for a correlation between the rise of a new emergent artist and a corresponding trend in past years. In this work we present a

novel approach to music similarity space creation based on deep learning. A representation of music based on released year allows to observe how music spontaneously tend to aggregate in regions and form new musical currents.

The literature for music similarity is vast: Audio Music Similarity (ASM) is a task that has been proposed for the first time in 2006, in the 'Music Information Retrieval Evaluation eXchange' MIREX event [1]. In its first editions, music similarity's researches where based on handcrafted features and Machine Learning ML techniques which involves the use of classifiers. The evaluation of the similarity measurements, instead, relied primarily on subjective ratings of some music experts, which evaluated the performance of the algorithms according to their personal judgements. In MIREX context, the work proposed by Bogdanov et. al. [2] is one of the first study which lead to very good results in the audio music similarity task. They employ different hybrid systems based on tempo-related music features and 'Support Vector Machine' SVM classifiers which involve the use of high-level semantic descriptors derived from low level features.

Today, Deep Learning (DL) techniques demonstrates to be much more performative than the classical Machine Learning Classifier. Furthermore, the works mentioned above result to be unsuitable to efficiently address the similarity task, as it is something very subjective cannot simply rely to just the ratings of human judgments. Provide a qualitative objective measurement and a representation of similarity is something challenging. New approaches to this problem employ 'Artificial Neural Networks' and perceptual features, build on psychoacustical scales created on the basis of human's sound perception, which substitute the less descriptive handcraft features and ML techniques.

The work by [3] addresses the similarity task by introducing a novel approach with the use of Conditional Similarity Networks which allows to jointly model several dimensions in which the similarity can be described. In addition, triplet loss technique, trained on a combination of users tags and algorithmic estimates, is employed to create the similarity metric space. Other similar works exploit the triplet loss technique, for example the work by Cleveland et al. [4] which employ simple fully connected

networks with a triplets mining based on the artist label, or the work by Pretet et al. [5] which use more advance Deep Learning architectures, like CNN, with perceptual features as music descriptors.

In this thesis, a novel approach based on the current state of the art architecture in Natural Language Processing tasks, the 'Transformers' [6], is used to the generation and optimization of the embedding music space. In our model, we employ the encoder structure only, removing the last Softmax layer, commonly used for multi-class classification tasks, and replacing it with a triplet loss layer for the generation of the similarity metric space. Triplet Loss technique was first introduced by Google in [7]. The aim of the triplet loss is to create an L2 Euclidean space such that similar samples are located in near-by regions, while dissimilar one are far from each other. We use a multi labels triplets mining strategy based on genre, artist and id track label. The triplets are randomly sampled according to one of this labels from the dataset with different probability. In our final test we choose 20 percent of probability for the artist, 30 percent probability for id track label and 50 percent probability for genre label. The reasons which move this choice is that a song is more similar to itself than to the other and songs by the same artist are similar. In this study we implement an offline strategy for triplet mining, generating batches of N triplets at each time steps by N*3 randomly samples from the dataset. For the training stage we use 'FMA' small library [8], which includes eight different balanced genres, while for the test stage, we select Spotify's Italian playlists with songs released from 2016 till 2021.

To evaluate the performance and the effectiveness of our model, we create a 2D graphic interface, in which show all the FMA music pieces colored according to the genre and located in space according to similarity in a timeline starting from 2007 till 2017. We also plot the test music tracks, represented by black point in space. Through simple basic functions, this interface helps the evaluation of our model performances through human evaluation conducted by music experts.

This thesis is structured as follow: in Chapter 1 we present the state of the art on the music similarity task, triplet loss strategy and transformer architecture. We give a summary of the needed theoretical background knowledge for the comprehension of our research in Chapter 2, discussing

about the perceptual features, the Deep Neural Networks and the triplet loss technique. Next, we present our approach to the music similarity task, explaining the reasons to employ perceptual features, the architectures used and the presentation of the graphic interface in Chapter 3. In Chapter 4 we discuss the setup for all the experiments, showing the one which lead us to better results. Finally, we summarise the key findings of our research, discuss the limitations or weaknesses of the study and the possible futures improvement of our work in Chapter 5.

1 State of the Art

In this chapter we will present the reasons why, today, music similarity has gained importance with the advent of digital revolution and why it needs to be further explored. We will go through the most relevant approaches in the field. Finally we will give an overview of what has been proposed in the field of musicology for the study of the evolution of music trends.

1.1 Music streaming service provider

In recent years, new music streaming service providers have seen a considerable number of daily listeners growing. Digital revolution has brought a drastically changed in the technologies employed for audio reproduction, bringing the old vinyls, cassettes and CDs to be replaced by the new online music streaming services. The reasons can be found in the accessibility provided by portable devices and by the offering of an incredibly large number of music tracks. The most immediate consequence of this phenomena is the needs of having an efficient system to suggest and help people to find new music. In this context, music similarity is

becoming a central topic which need to be taking into account when it comes to design recommendation systems.

Nowadays, *'Spotify'* [9] use three main types of recommendation models which are:

- Collaborative filtering [10]: it is the process through which it is possible to make automatic predictions on the users' tastes. This type of recommendation model filters for information or patterns from the collaboration of multiple agents, even if these agents might not explicitly collaborate with each other, and try to identify similarity between them. In real world, people have always been making use of different kind of collaboration which rely on spoken words, advertising, media and so on, to make decision. Collaborative Filtering follow the rules of this natural process and it based on the assumption of the 'nearest neighbors'. This means that is if the behaviour of user A is similar to B in a issue, probably the two users would have a similar behaviour in other issue;

- Natural Language Processing (NLP) [11]: it is a subfield of Artificial Intelligence AI which employ computational techniques to analyse music tracks, playlists, blog posts, social media comments. These are turned into text documents and they are used to search for similar patterns among them. These NLP Spotify's algorithms constantly search the web to find any kind of text related to a particular track to come up with a description for each song. Artists and songs are assigned to classifying keywords based on the data and than they are used to classify songs and match them with other similar profile's song.

- Audio models [12]: these models rely directly on particular music descriptors which are used to search for similarity between tracks. These features are extracted directly from raw audio and they can be both low or high level descriptors. They give information about the key, tempo, mood, danceability, valence, energy, loudness, speechiness, instrumentalness, segments, tatums, bars, beats, pitches, Timbre and so on. With this approach, new tracks on Spotify have the possibility to be suggested to users and added

to playlists even if they have very little initial listeners, so when collaborative filtering could be less efficient.

All of these models try to answer the same question, which is 'what makes two songs similar'?

In this study we will explore the third of these approaches, with the aim of creating a similarity metric space through the use of perceptual features and deep learning DL Artificial Neural Networks ANN, which can be used as a tool for the study of music trends evolution.

1.2 Similarity

In this section we will see what is MIREX and its oldest music similarity's algorithm in 2006. Then we will move to most recent methods which instead involve deep learning DL techniques, focusing mainly the attention on triplet loss, which is the focus mechanism employed in this research. We will briefly talk about music similarity and the music features needed to describe it.

1.2.1 Similarity in MIREX

MIREX

'Music Information Retrieval Evaluation eXchange' MIREX [13] is a community-based framework which, every year, submits new challenges related to a multitude of different 'Music Information Retrival' MIR and 'Music Digital Libraries' MDL [14] tasks, which are chosen by the community itself, through different inputs, such as a mailing list or MIREX-WIKI. The aim of this context is to define standard metrics of evaluation in the 'MIR' field. Since the datasets used in MIREX context cannot be freely shared, the 'International Music Information Retrieval Systems Evaluation Laboratory' IMIRSEL [15] provides the infrastructures to upload and to evaluate the different algorithms submitted by the participants, which are subjected to both statistical test and human evaluations.

ASM

'Audio Music Similarity' AMS is a task that was run for the first time in MIREX context in 2006 and whose last edition was in 2016. In its first edition, the similarity evaluation was based on two different metrics. The first metric requires the experts or the volunteers in MIREX lists to compare, using the spiffy Evalutron 6000 web interface [16], a set of thirty songs to a given query track, giving both a numerical score from zero(not similar) to 10(very similar) and a personal judgment to state if the song is similar to the query track. Tracks were previously filtered in order to avoid to select song's cover or tracks by the same artists. The second metric relied on objective statistics, as the average percentage of genre, artist and album matches in a certain number of top results. The final evaluation of the participant algorithm was an attempt to find any kind of correlations between these two different metrics.

A very performative work in audio music similarity task submitted in MIREX 2011, was the one proposed by Bogdanov et. al. [2]. They authors proposed three different baseline approaches based on low-level features. The first one was an unweighted Euclidean distance performed on 59 of manually selected descriptors, which were extracted frame by frame, then summarized through mean and variance statistic and finally reduced in dimension by the principal component analysis (L1-PCA). The second one was an Euclidean distance based on relevant component analysis [17] RCA. The features were further reduced through RCA algorithm previously trained on a set of similar songs, thus minimizing the irrelevant variability in the data and amplifying relevant one. Finally, the third baseline approach was a Kullback-Leibler divergence based on 'Gaussian Mixture Models' GMM [18] of MFCCs, reducing the computational complexity using just a single Gaussian with full covariance matrix.

Against those three baseline approaches they proposed three new hybrid approaches. The first one exploits a simple sum of measure distances performed on two tempo-related musical parameters, the beat per minute BPM and the onset rate OR. They move from the assumption that songs with the same BPM, or multiples of the same BPM, are more similar than songs with non-multiple BPM. Given two tracks, similarity scores were the result on the sum of weighted distance measure based on

the two descriptor. The second approach exploits machine learning techniques based on semantic measures. They exploit a 14 multi-class support vector machine SVM classifier, to infer different groups of musical dimensions which include genre, culture, moods, instruments, rhythm, and tempo annotations. To do this, high-level semantic features are first inferred from the low-level ones. The third measure, instead, involves linearly combining previously introduced metrics into a hybrid one: an Euclidean distance based on principal component analysis of timbral, temporal and tonal descriptors, and a timbral distance based on single Gaussian MFCC modeling.

An important observation that must be made, coming from the aforementioned works, is that ASM tasks in MIREX, in most of the cases, were evaluate relying primarily on human and subjective judgments. This is the principal cause of the limitations on these algorithms, as the luck of a clear definition of similarity and the variability deriving from subjective judgements, make it hard to find both a reasonable approach and a metric to treat the similarity task. As the author of the paper [19] suggested, these rely most on the fact that people perceive music differently one from each other, so the accuracy of the final result would not be as expected and there is a low inter-rater agreement between the given ratings.

1.2.2 Deep Learning Approaches

Artificial Intelligence (AI) has brought substantial improvements to the music similarity field thanks to the new knowledge and techniques developed in deep learning DP [20]. DP techniques allow an automatic learning of the training dataset which result in an internal representation of the data distribution with the consequent possibility to recognize and predict the class for unseen data. DP approaches in music tasks generally include the extraction of perceptual music features from raw audio, which are then feed to the artificial neural networks ANN, with the aim of learning some patterns and peculiarity characteristic of the data. Works like [21] try to deal with the problem of audio-based near-duplicate video retrieval through Convolutional Neural Network (CNN). Others like [22] employ the neural networks to build robust music recommendation systems and others more like [23], use deep learning techniques

for music classification system and autotagging for most of the popular music streaming services. Triplet loss is a well known DL deep learning's loss function build on the top of ANN which is used to deal with the problem of similarity. It was introduced for the first time by Google, in 2015, in the paper [7] where the authors shown an high-performing algorithm employed in the field of computer vision for the face recognition task. Triplet Loss is based on the idea that similar data must be near while dissimilar ones must be pushed away from each other in the metric similarity space. The implementation includes the splitting of the dataset into sets of triplets composed by an anchor, a positive and a negative sample, in which the first two samples share similar characteristics, while the third one show very dissimilar characteristics from the other. The objective is to create a metric space by minimizing the Euclidean distance between anchor and positive samples, while maximizing the one between the anchor and the negative samples. The main important step in this process concerns the criterion according to which the triplets are selected, as it have a strong impact on the final result. Following, we shows some of the latest implementation of this technique with employ different strategies for the triplet mining.

Triplet Loss

[3] the authors introduce the concept of multidimensional similarity. They employ conditional similarity networks [24], which are networks able to learn embeddings differentiated into semantically distinct subspace. They combined them with ad-hoc masking functions, to train a single network with four disentangled semantic dimensions of similarity, which are the instrument, mood, tempo and genre dimension. The triples mining was performed by choosing the three samples according similarity in each dimension. In addition, they include a set of triplets mined according to the only tracks information as a similarity regularization to impose consistency across the embedding space, such that the anchor and positive samples must belong to the same track. To conduct the experiments they use Million song dataset [25] in addition to the metadata provides by last.FM dataset [26]. Two songs are considered similar if they share at least one tag in the same dimension. They extract 128 Mel-spectrogram's coefficients from three seconds of song excerpts as mu-

sic descriptors and performed a multi-label classification (genre, mood, instrumentation, tempo). Triplets are than feed to a CNN composed by a first layer with 64 convolutional filters, followed by six Inception Blocks plus a final inception module with an output embedding size of 256, ending with a final L2 normalization before computing the euclidean distance for the triplet loss. The main problem of this approach is that it still rely on human annotation as the four dimensions tags are subjective evaluation of the users.

Another work which employs the triplet loss technique is [4], which focus its attention on the artist-similarity. The triplets mining, here, is implemented so that the anchor and the positive samples result to be songs by the same artists, while the negative one is by another artist. They used a two-layer fully- connected network, in which the first layer has a *'sigmoid'* activation function, while the other a 'tanh' activation function. FMA dataset [8] provides tracks raw audio and additional metadata from which the authors extracted 518 audio music descriptors, like Zero- Crossing Rate [27], CQT [28], Tonnetz, MFCC [29], and STFT [30], to which they apply a z-score transformations. The distance metrics of the triplet loss is calculated as an euclidean distance between these features.

In a most recent work that exploit triplet loss for the music similarity task [5], authors propose a new recommendation system based only on the audio content which does not require music tagging. Instead of using classical handcrafted features feed to a classifier, they employ perceptual features feed to a CNN, with an additional pooling layer which allows to interpolate between different [31] statistics to down sample and summarized the features. They analyzed 246 music tracks from Pandora multi-labels dataset, with 488 tags, organized in five categories, extracting 96 bins of CQT, (12 bins/octave and fmin= 32.70 Hz), and a hop size of 1024 at 44.1 kHz, per 512 time frames. The basic idea of the triplets mining relies on an oracle similarity function which returns a list of songs, selected from the datasets, which are the most similar to a given query track. Songs are ordered by descending similarity, in which the similarity score is based on tags likelihood. The query track acts as the anchor sample and the positive sample is chosen such that his index

in the ranked list is lower than the negative one. They also limit the number of positive sample per anchor and the number of negative sample per anchor-positive pairs as only the first few songs in list turn out to be very similar to the query track. They results demonstrate that the triplet loss gives better result with ranked list of songs instead of tags.

1.3 Evolution of music trends

Music styles have been evolving through years as people's music tastes have, since they influence both each other and they are a direct consequence of popular custom and social changes. As researches in modern music history have revealed, different generations of people tend to like different music genres and if we take a look to music genres across all ages, we can easily notice how the music has dramatically changed decade after decade. There is no a single factor which induce this evolution, but instead there can be identified an infinite number of causes, which, all together, equally contribute to this phenomena.

One of the possible causes of this phenomena can be found in the field of Neuropsychology, which plays an important role in this sense. The perception of a musical sounds has a deeper influence on people than any other kind of sounds or noises. When human brain process musical sounds, they are not perceived just as a physical sensation, but instead, they are able to affect humans also from an emotionally point of view and have the power of bringing back in mind memories belonging to the past. In [32] the authors conducted an experiment, called 'Memory Lane', which shows the effect of *'reminiscence bump'*. This effect occurs in adults which show an increased recollection of the past events happened during their adolescence. This tendency of developing such a stronger remembrance of the past could be the consequence of hormonal, neurobiological or social relationship and moreover, it affects every aspect of life. For example, this happens also for music, as the experiment in mentioned article demonstrated. In such test, some undergraduate university students were asked to rate the liking of each song, according to the personal perceived emotion, the music's feeling conveid, its sense of nostalgia and to relate that song to a person. Reminiscence bump

was found from the results, as they shows high ratings for music from 1980 and 1984, years which could be reconnected to the period of adolescence of participants' parents and their relative reminiscence bump. The reported results of the experiments leave researchers with the open question of whether perhaps musical preferences are transmitted over time from generation to generation

Other reasons, that could be related to the changes of genre and musical tastes through years, could be addressed to the specific social context in which a certain population live, because music is always a direct consequence of historical events, people's habits, traditions, ideas beliefs, religion, culture, commonly accepted way of behaving or doing something, geographical regions or currents of thoughts. There are a very large number of factors which play an important role in this scenario. There is a close relationship between society and music, as the society influence the music and the music, in the same way, influence the society. As any other form of art, music is a pure expression of human being and thus perfectly reflects whatever is around it.

Record labels, in order to strictly have the power to rule music, need to always be a step forward in knowing the new direction in which the music is evolving so they can anticipate new music styles. In this context, tracing the evolution of music styles through years and try to identify the most popular trends in a certain time or in a certain region, or both, could be an interesting starting point to lay the basis for the study of the future trends' prediction. This is the main reason why a solution to this problem need to be addressed.

This topic has been tackled by different researchers and following, we will go through the most recent works to show some baselines approaches.

Some works like [33] treats the problem of predicting the trends' evolution of music genre according to two different approaches, which are the spatial and temporal music influence and the music similarity. In the paper, the authors move from the idea that an Artist's musical influence is not limited to just a genre. Instead it can also be able to influence other genres and such influence can be effective for a certain period of time. With 'Spatio-Temporal Music Influence Model' STMI calculated

by a cosine similarity between the features embeddings of the artists, they can obtain specific spatial and temporal values. They evaluate the influence that musicians have not only on music, but also on the other musicians, taking into account that if they are located in different geographical areas, this influence would be poorer, and that ordinary artists would be more conditioned by more famous artists. The 'Euclidean Similarity Model' (ESM) is instead used to measure music similarity through an euclidean distance based on music features. Using PCA [34] they select the most relevant music descriptors from an initial set of 13 features, reducing the dimensionality of the data. They also propose an 'Euclidean Music Similarity' instead of the classical Euclidean distance to calculate a similarity score between the features.

Another work which focuses its attention on the evolution of music trends is [35], in which the authors proposed a music popular trend prediction model based on LSTM [36] architectures. To conduct the study, they rely on data which comes from the 'Alibaba Cloud Music Platform'. It is the top platform in China's digital music industry, which has been developed since 2016 and which held a 21.9 percent market shared of music both from Chinese digital music platforms and Xiami Music, with millions of active users. From this dataset, they extracted information about users activity, including collections, downloads and playbacks of songs, and metadata of each song and artist's song. The aim of the research is to predict the future popularity of an artist's song looking at the average playing volume of its past songs. They used an LSTM architecture composed of three input neurons, which represents the playing amount, the playing average and the playing variance of the singers, two hidden layers with 35 and 10 units and three output neurons, representing the play volume, average and variance. At the end of the experimental results, they found that song playback volumes, collection volumes and download volumes are the best data to predict the popularity of an artist's song. They also demonstrated that LSTM architecture outperform the state of the art of the previous ML's techniques.

1.4 Our contribution in musicology

How can this work could really help musicologist in the study of the evolution of music trends? The aim of this research is to create a basic tool able to assign a specific position in a 2d space to each added songs from several different genres. In such a space, songs which show similar characteristics of sound, harmony, melody, rhythm, growth or which belongs to similar genre or sub-genres, are located in near-by regions. In such a metric space, distance becomes the measure of similarity itself. To allow the visualization of songs and so to allow researchers to conduct their study on a concrete tool, we create a simple interactive interface, in which songs are represented by points in space characterized by a color according to their genre. Clicking on them, information related to artist, title and year are available, in order to allow comparisons with neighboring songs. Furthermore, there is the possibility to show only songs by the same year or songs published in a certain range of years. In addition, the heat map helps to better show the distribution of genre in the 2d space, so that are no more boundaries between them or kind of clusters, but instead, a continuously changing shades of color. All of these basic and elementary functions, results to be extremely useful to trace the evolution of a particular style of music. Studying a particular sub-genre results to be extremely simple, as it can be added, once per time, songs by consecutive years, so that its evolution through years could be easier to track. With this tool, a large dataset of a specific genre, providing songs from multiple years, could clearly show new directions undertaken and towards what other genres is going to resembling more. This means, that we will able to know what genre is able to influence more and where the center of mass of that particular style is gravitating on. Contamination are also shown, as a song point with of a particular genre could be located in other color regions, where it would not be might expected. Probably, that particular artist's song could be affected by the influence of other different music styles, even if does not belong to that current of music. these are just some little suggestions on how this interface can be used, but in reality the possible real world applications could be much more.

1.5 Some of the best Transformers practices

For the purpose of our research, we have decided to rely on relatively new architectures, called transformers, introduced for the first time in 2017 by Vaswani et al. [6]. As the best of our knowledge, we employ, for the first time, these architectures for music similarity tasks. The reason behind this choice can be found in the high performances shown by these networks in several deep learning tasks in which they outperform the state of the art, reaching better results with respect to classical deep learning networks. In this section we will give a summary of some of most recent works which employs in which the transformers represents the state of the art.

Works like [37], show how transformers architectures outperforms the previous end-to-end sequence-to-sequence model, like time-delay neural networks TDNN [38] and long short-term memory recurrent neural networks LSTM on the speech recognition task. They employ the same encoder decoder structure of the original paper 'Attention is all you need', with a very deep model. In fact, the use 48 encoder layers, which generate a high-level representation of the input and a 48 decoder layers, which models the data as a conditional language model. The experiments were conducted on Switchboard-1 Release 2 (LDC97S62) [39] datasets, which contains several hours of audio speech, from which they extracted 40 log Mel filter-bank as music descriptor. At the end of the experiments, they demonstrate that the introduction of stochastic residual layers in the architecture further increases the results and enhances the performance with respect to even more complex hybrid model.

[40] is a work that introduces a new state-of- the-art classifier that uses label-attention, called Roberta, for the extreme multi-label classification (XMC) problem. They also present a framework for data augmentation, which rely on ruled-based strategy and language-model-based strategy, with the aim of mitigate the problems arise due to the scarce data for tail labels, which in real-world scenario are extremely unbalanced. To validate the impact of the augmentation framework, they use

'LA-RoBERTa'. This architecture is obtained from a pretrained model call 'RoBERTa' [41], further fine-tune it on the AmazonCat-13K [42] dataset, with label attention, so that each token embedding can have different impact on each label. They show how 'LA-RoBERTa' outperforms previous state of the art model like the 'XML-CNN' [43] or the AttentionXML [44].

In the field of music information retrieval, Zhao et al. [45] present 'MusicCoder', which is a transformer inspired by BERT [46] architecture, which includes the only encoder structure. It is self-supervised pretraining model which is able to learn a powerful representation of music thanks two masking strategies, the Contiguous Frames Masking (CFM) and Contiguous Channels Masking (CCM). The first one randomly masked different length of consecutive frames, whose size is sample by a normal distribution. The second one, instead, randomly masked block of consecutive channels. With this technique, the encoder is force to reconstruct missing data, so acquiring a high-level understanding of them. them They join songs from three datasets Music4all [47], FMA-Large [8] and MTG-Jamendo [48] and extracted music features, with 'Librosa' library [49], which includes Mel-scaled Spectrogram, Constant-Q Transform (CQT), Mel- frequency cepstral coefficients (MFCCs), MFCCs delta and Chromagram. This research shows how these architecture can outperform the state of the art both in classification task, where it achieve accuracy of 94,2 percent, and in music autotagging task.

2

Theoretical Background

In this second chapter we will first gives an overview of the features we have chosen for our experiments. We will also introduce the Deep Learning (DP) architectures that we employ for the generation of the similarity metric space. To achieve the objectives already discussed in the chapter 1 of this thesis, two different kinds of Artificial Neural Networks has been used: the Transformers, which are the current state of the art in the field of Natural Language Processing and the Convolutional Neural networks, which were originally employed in computer vision applications and images processing. In Section 2.2 we will illustrate the triplet loss function and the different strategies that can be implemented to mined the triplets. Finally, at the end of this chapter, we will discuss the dimensionality reduction technique we use to generate the final output embedding vectors, in a way such that they can be represented in a 2D space.

2.0.1 Features as descriptors for music similarity

Music similarity is not something that can be easily defined since it involves several aspects of human perception of sounds and also different dimensions on which it can be described and evaluated. Consequently, one good point to start our research is to look for the most basic and elementary parts in which a music piece can be break up and than try to define a similarity between them. Probably, the easiest way to find these elementary components is to learn how musicologists analyze and decompose a music piece. *'Guidelines for style Analysis'* [50] is a book written in 1992 by one of the greatest musicologist of the XX century, called *Jan LaRue*. This book lay the foundation for modern music analysis with the identification of six fundamental parameters, which in some way subdivide the phenomena of the music into manageable parts. They are: 'sound', 'harmony', 'melody','rhythm', 'growth' and 'text influence'. According to LaRue, conducting an analysis based on these six categories is sufficient to give a whole description of the song itself and of its style. However, these parameters would be insufficient for our purpose as we know the behaviour of human auditory system is much more complex than this. In fact, when we state that a music piece is similar to another piece we are certainly referring to the similarity of the sound, the harmony, the melody, the rhythm, or the mood, the meaning of the lyrics, but there could be other hidden reasons we cannot either understand and which we don't really know. Psychological, cultural, perceptual aspects, in fact are involved in the process of assessing the similarity among songs by humans. Therefore, what we can do is to entrust the comprehension of human perception of similarity to AI and in particular to Deep Learning architectures. On one hand, we employ low level perceptual features, which are just simple vectors of numbers that a machine can understand,to the description music sounds as the humans logarithmically perceive it. On the other hand, we leave the comprehension and the description of all these high level features, mentioned above, including Psychological, cultural, perceptual aspects, to the deep neural architectures. For this thesis the music low level descriptors we have chosen are: MFCC, MFCC delta, Chroma, CQT and Mel coefficients.

Mel Spectrum

Humans perception of sound is not linear. As a matter of fact we are able to distinguish two pure sine waves at 90 Hz and 91 Hz, but we cannot do that between to sine waves at 10000 Hz and 10001 Hz. It means that we perceived the distance between two close frequencies differently according on where they are 'located' in the spectrum. The Mel scale born with the aim of linearize the frequencies in order to make the distance's perception of them equal in all the spectrum. In other terms, Mel scale is a non-linear transformation which allows to convert the frequencies from Hz in a new scale, Mel, such that they well approximate the psychological sensation of heights of a pure sinusoid. Here is the formula to convert from a frequency from Hz to Mel:

$$m(f) = \frac{1000}{\log 2} \log(1 + \frac{f}{1000}), \tag{2.1}$$

where f is the frequency to convert, log is the natural logarithm and m indicates the equivalent value in Mel.

The process to convert a signal in a power spectrum based on a Mel scale involves four steps. First the input signal is sampled and windowed, generating a certain number of segments on which Discrete Fourier Transform DFT [51] is performed. Then, the obtained spectrum is splitted into a certain number of equally spaced central frequencies, according to the Mel scale. Finally, the Mel spectrum is computed by multiplying the magnitude spectrum of each segments by a filter bank, composed by triangular filters centered on Mel frequencies.

$$s(m) = \sum_{k=0}^{N-1} [|X(k)|^2 H_m(k)]; 0 \leq m \leq M-1, \tag{2.2}$$

where $X(k)$ is the magnitude spectrum of a segment, M is the total number of triangular Mel weighting filters and $H_m(k)$ is the weight given to the k-th energy spectrum bin contributing to the m-th output band.

MFCC

Mel Frequency Cepstral Coefficients MFCC [29] are perceptual features, which similarly to the Mel scale, are based on the logarithmic human

perception of sound and which are particularly suited to describe audio and speech signal. They can be calculated according to the formula:

$$C(x(t)) = DCT[\log(DFT[x(t)])], \tag{2.3}$$

where C is the cepstrum, $x(t)$ is the time domain signal, DFT is the Discrete Fourier Transform [51] and DCT is the Discrete Cosine Transform. The MFCC feature extraction technique basically involves a first step of sampling and windowing of the signal, to perform the analysis on short segments of audio. Then, DFT is applied to each of the generated segments, which are now in frequency domain. Mel spectrum is computed by multiplying each power spectrum by the Mel-filter bank, after applying a logarithmic function, which ends up in segments of logarithmic power Spectrum. Finally, the Discrete Cosine Transform DCT [52] is performed on the Mel spectrum, converting the signal in the cepstral domain. Cepstrum is a spectrum of a spectrum in which a 'quefrequency' peak corresponds to the pitch of the signal and the low 'quefrequency' peaks represents the number of formants. Normally, in real world application are used a number between 8 and 20 MFCC coefficients as increasing the number would just increase the complexity of the model with no relevant advantages. Low order coefficients contain most of the information about the overall spectral shape of the signal's spectrum, which can be thought as the timbre. Even though higher order coefficients represent increasing levels of spectral details, there are no big advantages to use them because most of the more relevant information of the signal is concentrated in low order coefficients.

In particular, in speech signals, zero order coefficients represent the average log-energy of the signal, while the first order coefficients indicate the distribution of the spectral energy between low and high frequencies (in Mel scale). Sonorant sounds have positive coefficients because the energy is concentrated in low frequencies, while fricative sounds have negative coefficients as the energy is most concentrated in high frequency.

MFCC delta

MFCC delta coefficients [53] are static features, as they only contain information related to a given frame. Dynamic MFCC delta coefficients,

which show information about the temporal dynamics of the signal, can be obtained by calculating the first and the second derivatives according to this formula:

$$\Delta c_m(n) = \frac{\sum_{i=-T}^{T} k_i c_m(n+1)}{\sum_{i=-T}^{T} \mid i \mid}, \tag{2.4}$$

where $c_m(n)$ denotes the m^th feature for the n^th time frame, k_i is the i^th weight, and T is the number of successive frames used for computation, generally taken as 2.

Chroma Vector

Chroma [54] is the representation of the spectral energy of a signals in a musical chromatic scale format. In fact, the whole tonal content of an audio signal is condensed in just twelve coefficients, each of which represents the semitones we found between a frequency and the double of that frequency. Therefore, *chromagram* must be take into account when dealing with high-level semantic analysis, like in the case of this study, which results fundamental for the harmonic similarity estimation as it allow to see all the pitches played in the analyzed music piece.

$$v_k = \sum_{n \epsilon S_k} \frac{X_i(n)}{N_k}, \tag{2.5}$$

where v_k is the mean of log-magnitudes of the respective DFT coefficients, S_k is a subset of the frequencies that correspond to the DFT coefficients and N_k is the cardinality of S_k.

CQT

Constant Q transform [28] is a spectral descriptor which resemble the DFT. The difference is that it maintain for each frequency a constant ratio between the central frequency of a bands filter f_k and the spectral width of that filter (f_k-f_{k-1}), helping to increase an equally energy distribution of a music signal. One important aspect to notice is that the CQT prevents from some issues when frequencies are converted into a logarithmic scale. For example, it enhance the resolution of peaks on the lower end as the window length of each bin is function of the bin

number, so the buffer size increase for lower frequency, while reducing the buffer size for high frequencies and thus require less computational power. Given the above, CQT emulates human logarithmic perception better than how Short Time Fourier Transform STFT [30] does. .

$$f_k = f_0 * 2^{\frac{k}{b}} \tag{2.6}$$

where f_0 is the central frequency of the first band and b the number of frequencies per octave.

$$X[k] = \frac{1}{N[k]} \sum_{n=0}^{N[k]-1} W[k,n]x[n]e^{\frac{-j2\pi Qn}{N[k]}}, \tag{2.7}$$

where W is the window function and $\frac{2\pi Q}{N[k]}$ is the spatial frequency,

$$Q = \frac{f_k}{\delta f_k}, \tag{2.8}$$

where Q is the quality factor, the constant ratio between the central frequency of a band f_k and the frequency resolution δf_k,

$$N[k] = Q\frac{f_s}{f_k}, \tag{2.9}$$

where f_s is the sampling frequency and $N[k]$ is the window length for the $k-th$ bin.

2.1 Deep Learning Background

Following, we discuss about the *'Artificial Neural Networks'* ANN, which are deep learning architectures inspired by 'Biological Neural Networks' of living being. These last ones are composed by billion of densely processing units, called neurons, which are connected with each other by a similar number of synapses. Synapses are structures responsible for the transmission of the electrical signals through the neural system. We will explain step by step how ANN are composed, how they work and their main characteristics. Then, we will focus our attention on the two particular type of architectures we employ in this study for the generation of the similarity metric space, which are the 'Convolution Neural Networks'

and the 'Transformers'. Finally, we provide a brief summary of triplet loss technique and its different strategies.

2.1.1 Artificial Neural Network

As the name itself suggests, Artificial Neural Networks ANN [55] are parallel computing systems inspired by what is the 'learning center' of human beings, the neural system. Therefore, these architectures are designed to simulate how it can analyze and process all the external stimuli to which human sensory system react. This kind of architectures are also known as 'deep models' and are mainly characterized by compositions of non-linear function, which can be denoted as:

$$(f_i(x)) := (f_m \circ f_{m-1} \circ \dots f_1)(x) \tag{2.10}$$

where f_i is a stage of processing, called layer, in which the input is transformed, with a non-linear function, to an output and feed again as input to the following layer and m is the models depth or the number of these layers, which does not include the input one.

Multi-Layer Perceptron

The simplest model, but also the oldest one we can trace back, is the Multi-Layer Perceptron MLP [56]: it is basically a fully connected sequence of layers composed of a variable number of neurons.

A neuron is a processing unit of the layer responsive for a particular non-linear transformation of the input. We can formally define it with this expression:

$$f_i(z|\theta) := \rho_i(w_i^T * z + b_i) \tag{2.11}$$

$$\theta = (w_i, b_i, \rho_i) i = 1^m \tag{2.12}$$

where f_i is the $i - th$ layer of the network, z is the input array, 'ρ' represents a non-linear transformation and θ is shown in 2.12. Other variables are the learnable parameters of the network:

- w_i, the weights of each connection;

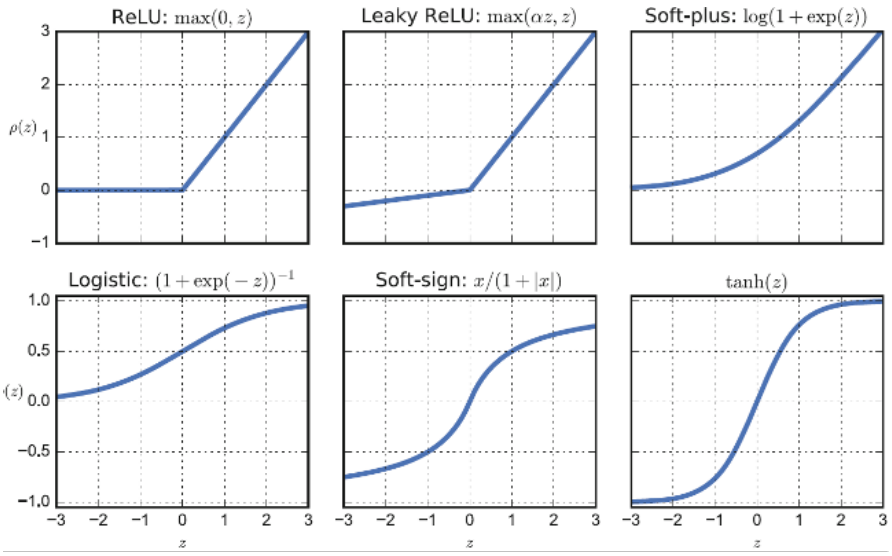

Figure 2.1: *Different types of Activation Function*

- b_i, the bias;
- ρ, the transfer function, also called activation function;

Activation function

The activation function helps the model to learn non-linear structure in data and to control the learning process of the training set. Some of these functions depicted in Figure 2.1, as the input diverge from zero, approach a saturating value, like *tanh* function. In these region these of constant constant values, the first derivative to zero and so the error cannot be back propagated. The choice of the activation function varies depending on the structure of the output space or according to the network's task.

As we said before, each layer is composed of a variable number of units (neurons) and we call this number 'shape'. $d0$ is the shape of the input layer and d_i is the shape of the $i-th$ internal layer. We call 'd_{out}' the shape of the output layer, which usually represents the number of classes of the input dataset when deal with classification problems. We can imagine that each stage is a kind of feature extractor, where each neuron acts on a different portion of the data to learn patterns. The aim is to make, in the last layer, the input data matches the correct output. The

loss function in the output layer states how well the prediction is, then computes an error and backpropagates it. The weights of the network are accordingly updated, so that at the end of the training process, makes the network performing as expected.

Multi-label classification

Multi-label classification is a type of prediction task in which the network is required to predict more than one label class per input sample. In this case, the labels are usually encoded as a binary vectors and are not mutually exclusive. The output layer has a logistic transfer function, like Sigmoid and the standard loss function is the Binary Cross-Entropy. This last one is also called Sigmoid Cross-Entropy Loss, as it is a Sigmoid activation function plus a Cross-Entropy loss. This function acts independently for each vector component (class), so that the loss computed for each of them cannot affected by the others labels. This is the formula for the Binary Cross-Entropy loss, where y' stands for the predicted output.

$$f_{err}(y', y) := \sum_{c=1}^{C} y_c - \log(y)'_c - (1 - y_c) \log(1 - y_c) \tag{2.13}$$

Multi-class classification

In multi-class classification task, each input sample fed to the network, at the prediction stage, can be assigned to just one of the possible output class. Here labels are one hot encoded, meanings that labels are matrix where rows are the numbers of sample in dataset and columns are vectors with the length equal to the number of total classes. All of the vectors values are zeros, except for the index of the correct class, for the corresponding sample, which has the value of one. In the output layer is commonly employed a Softmax activation function and the loss is computed through the Categorical Cross Entropy Loss, which is shown in the formula 2.14:

$$f_{err}(y', y) := -\sum_{c=1}^{c} y_c \log(y) \tag{2.14}$$

Gradient

The gradient is a numerical calculation, in vector form, which gives us information, during the training process, about the direction where the loss function is increasing faster. So it tells how the weights should be updated in order to follow the exact opposite direction. It is calculated for a single input-output sample through a process called back propagation [57], which is an iterative method which calculates partial derivatives of the loss function with respect to any weights and bias in the output layer. This is actually done for every layer, starting from the last to the first, in a way that every layer does not affect the other. Each of the values computed by these operations tell us how sensitive is the loss function to each of the network's parameters. Once the gradient vector is obtained, the corresponding gradient value is subtracted from each of the weights and then multiplied by the learning rate, which is a small number between zero and one, which allow us to regularize the speed of the learning process.

$$\theta \rightarrow \theta - \eta \nabla_(x, y, |\theta) \tag{2.15}$$

where ∇ is the gradient function with respect to the parameter θ, and η is the learning rate, which tell us how big are the steps the network walks to reach ∇ the global minimum at each iteration.

2.1.2 Convolutional Neural Network

The Convolutional Neural Network [58] is a class of Artificial Neural Network which is very performative in processing data with a grid-like topology, such for example a spectrogram or an image. This kind of Network draws inspiration by researches on biological visual system conduct by David Hubel and *Torsten Wiesel* [59] in the half ot the twentieth century, who brought great advances in the understanding of the visual system's functioning. In particular, they focus their attention on cats, which reveal to have two major types of cells, which were simple or complex. The simple one was able to respond to bars of light or dark in specific location and with specific orientation. The complex was still able

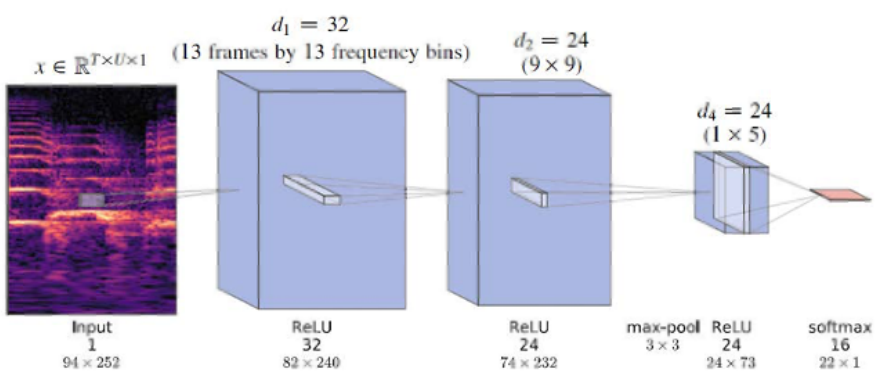

Figure 2.2: *Spectrogram feed to a Convolutional Neural Network with multiple convolutional blocks*

to respond to orientation, but in different nearby locations. These Hubel and Wiesels researches brought significant changes on the world of machine learning. Actually, later in the 80', Kunihiko Fukushima, inspired by the Hubel and Wiesels researches, proposed an hierarchical, multilayered artificial neural network called Neocognitron [60]. This model was relies on almost the same visual system's principal of operation and it is said to be the precursor of the modern CNN. CNN show similarity to the visual system, as they have a set of filters which react differently to specific portion and characteristics of the input and process these results sequentially, increasing the complexity of the patterns matched.

2D CNN Architecture

If we look at the architecture of a CNN, we can see that it is composed of several convolutional layers, which are the building blocks of this type of networks. In addition, we generally find also pooling layers, which do not have any parameters to train, as their only aim is just to apply some statics to the output of a convolutional layer. This is done to downsample the input, reduce the temporal dimension and pass the output result to the next layer.

The input array of the 2D CNN z is three dimensional:

- T is the temporal dimension;
- U is the spatial dimension;

- d is the number of channels or dimensionality of the input;

In each convolutional layer there are a certain number of d_i filters, called kernels, which represent the learnable parameters and which slide over the input to perform a dot product, a convolution, with a portion of the input restricted to their receptive field, or kernel size. In Figure 2.2 we can see how CNN with two convolutional blocks, a Max Pooling layer and a final softmax function, process an input spectrogram. The following Formula shows the convolutional operations applied over the input z:

$$(w * z)[t, u] := \sum_{j=1}^{n} \sum_{k=1}^{p} (w[j, k], z[t + j - [n/2], u + k - [p/2]]) \quad (2.16)$$

where n is the length of the receptive field of the filter w, which for convention is an odd number. p is the filter's depth, j and K represents the filter's position on the the signal and the relative of filters coefficient.

The filters have the ability to learn pattern like edges, circles, squares, for images, or envelope, transient or sustained tones in the case of sound signals. While in the first layer filters can learn just simple and local features, as the number of layers grow, more complex and general pattern are detected. During the training process, the kernels modify their coefficients to make the input sample match the right output in the final layer. Th kernel has the same number of dimensions of the input, but it is always smaller in size. This is the reason for which the CNN have sparse interaction across the input, which reduce the required memory space and in addition improve its performances.

Output size

What is the output size of a layer given an input dimension of T x U x d? In the case of an spectrum, T would represent the number of frames,

U the number of frequency bins and d the number of stereo channels. Then we can define other relevant parameters which are:

- Kernels height (K_T), kernel's width (K_U), channels (K_d);
- Strides height (S_T), stride's width (S_U);
- Padding over height and width: P_T1, P_T2, P_U1, P_U2;

After every Convolutional or Pooling layer these parameters are modified according to the following different cases: changes due to the Padding:

- $T_1 = \text{H} + P_T1 + P_T2$
- $U_1 = \text{W} + P_U1 + P_U2$

changes due to the kernel (K_T x K_U):

- $T_2 = T_1$ - $K_T + 1$
- $U_2 = U_1$ - $K_U + 1$

changes due to the sliding of the kernel over the input (strides):

- $T_3 = (T_1$ - $K_T)$ / $S_T + 1$
- $U_3 = (U_1$ - $K_U)$ / $S_U + 1$

Taking into account all of the above, the final output size would be T_3 x U_3 x K_d.

2.1.3 Transformer

Transformers are Artificial Neural Networks ANN which currently hold the state of the art in many different tasks, especially in the field of Natural Language Processing, and outperform with respect to the previous sequence to sequence models [61]. Transformers were first presented in 2017 in [6], where the authors Vaswani et al. demonstrated how transformers can overcome the limitations of the previous recurrent models such as RNN [62], LSTM[63] and GRU [64]. In fact, they show higher accuracy, especially with large datasets. As the papers title 'Attention is all you need' [6] could suggest, the biggest novelty in these networks is the so called self-attention or 'intra-attention'. It is the ability of the

network to give some context to its input data and establish relationships among them in order to create different weighted connections and give more attention to the most relevant words. While the previous recurrent models processed their input sequentially, thus requiring expensive memory and time-consuming computations, the most important innovation in transformers is the ability to process very long input data in parallel and acting as they would have an infinite memory. There are different types of transformer-based machine learning architectures, like BERT [46], which stands for 'Bidirectional Encoder Representations from Transformers'.

A simple intuition of the Self-attention

In the following lines, before we dive into mathematical formula, we presents a simple and intuitive explanation of what is the self-attention mechanism and how it ideally works.

In the case of transformer employed in 'Natural Language Processing' task, we call input embedding vectors the input sentences first splitted in sequences of words and punctuation, converted into token ids and then mapped into an n-dimensional space. Briefly, input embedding vectors are encoded numerical vector representation of the input sentences. Given some input embedding vectors with length n, $\mathrm{V} = [v_1, v_2, \ldots, \mathrm{v}_n]$, we want first of all give some context to them. Words can have multiple meanings, as for the English word 'bank', so when we encounter such word in our input sentence, we need to know the context in which it is used to know its real meaning. To do that, what a transformer does is to create some relationships between this word and any other words in the phrase, in order to establish weighted connections between them. RNN or GRU or LSTM architectures, as they process their input sequentially, would have take into account just the n-closest words, supposing that all the words related to 'bank' would have been located near it. But, as it could be observed in subordinate sentences, this is absolutely wrong because some words in a sentence could be referred to words in the other sentences and be located far away from each other. The first step in this process is the re-weighting of the input through weights that the are derived directly from the input itself as is it shown in this formula 2.17:

$$W_{ij} = v_i * v_j; \tag{2.17}$$

The second step, consist in multiplying every of the weights calculated in 2.17 with the original input values, finally obtaining the contextualized data Y_i.

$$Y_i = \sum (W_{ij} * v_j). \tag{2.18}$$

Up to here, we can make some important observations. First, the self-attention mechanism is shape independent, as it does not depend on the input length; second, proximity has no effect between vectors, this means that a transformer can ideally learn and contextualize infinite input length, and order has no influence.

Positional Encoding

The input embedding vectors which enter the encoder structure represents tokens, which means text splitted in words and punctuation, in a 'd' dimensional space, where words with similar meaning are located close in the space. So in this case, words have a special position according to their similarity. By the way, as we said in Section 2.1.3, the transformer has no any kind of information about the words position in a sentence, as it process their input in parallel and not sequentially. For this reason, we need something that makes the transformer aware of the index positions of the elements in the input tensors for different input sequences lengths. A first solution to this problem, would be to simply return the count, in the natural number set, of each element (word). This approach would lead to exploding gradient scenario, as the longer is the length of the input sentence, the bigger would be the numbers introduced in the network. So, we can try to normalized those natural index numbers in a range between zero and one, to avoid instability in the networks. Unfortunately, even this solution results to be inefficient, because in this second approach, the value of the position would depend on the sequence length. So, in this case, different positions would be assigned the same

index number for different sequences lengths. Another approach would be to exploit binary numbers. In this case every integer is first converted into a binary vector, thus creating a positional matrix with time temps row and depth-model columns. Unfortunately, also this method is results to be inefficient, as the binary vector comes from a discrete function and not from a continuous one. Therefore, we can think how to make this binary vectors continuous and make them cycles between zero and one. A solution can be found in a sine function, resulting in a vector positions such as in the following equation:

$$PE = (v^{(0)}, \ldots, v^{(depth-1)}) \tag{2.19}$$

where depth is the model dimension,

$$v^{(i)} = [sin(\omega_o x_i), \ldots, sin(\omega_n x_i)] \tag{2.20}$$

where v_i is the positional encoding vector of a single discrete position, x_i is the index of the position in space and i is the index position in time. Then, we want to find a linear transformation which makes this equation holds:

$$PE(x + \Delta x) = PE(x) * T(\Delta x) \tag{2.21}$$

To do this, we build a rotation function, making use of this trigonometric identity:

$$\begin{pmatrix} cos(\theta + \phi \\ sin(\theta + \phi \end{pmatrix} = \begin{pmatrix} cos\phi - sin\phi \\ sin\phi cos\phi \end{pmatrix} \begin{pmatrix} cos\theta \\ sin\theta \end{pmatrix} \tag{2.22}$$

So we create an encoding matrix alternating cosine and sine function resulting in these new vectors:

$$v^{(i)} = [cos(\omega_o x_i)sin(\omega_o x_i), \ldots, cos(\omega_n x_i)sin(\omega_n x_i)] \tag{2.23}$$

Finally, we end up in the formula presented in the paper [6]:

$$PE_{pos,2_i} = sin(pos/10000^{2i/d_{model}}) \tag{2.24}$$

$$PE_{pos,2_{i+1}} = cos(pos/10000^{2_i/d_{model}}) \tag{2.25}$$

where *pos* indicate the position in time, d_model is the dimensionality of the input embeddings and i is the dimension position index in the input tensor. Positional encoding is added to the input embedding vectors. This is the way of how transformers can trace the positions of elements in the input sequence, but also in multiple sentences. Positional encoding results to be a matrix with 'time-step' rows and 'd-dimensional' columns, where each row is a sequence of cosine and sine functions with frequencies that decrease according to a geometric progression from 2pi to 1000*2pi. This mechanism also allows, in the prediction stage, to feed the transformer different lengths of inputs even if they are longer or smaller from the ones seen during the training stage.

Scaled Dot-Product attention

Given the above, now we can dive into the real self-attention's mechanism, shown in Figure 2.3. We introduced the 'Scaled Dot-product Attention', which is the layer in which is performed the dot-product attention, which, in contrast to the additive attention, is much faster and more space-efficient, as it can be implemented using highly optimized matrix multiplication code. For each input word, or better each input embedding vector, that we here call query, we ask some more 'context', so we compare it with all the possible keys, composed by the whole set of the input embedding vectors. Combining a query with a key, we want back a value. These three objects, keys, queries and values, are the key aspect of the self-attention block and are implemented as dense layer in the network, with trainable weights, in order to make the queries, coupled with the keys, returning back the correct values.

This is done in the self-attention layer. Here a scale dot product is performed between the queries and keys, and the result is than normalized with $\sqrt{d_k}$ factor: this is extremely important for numerical stability and to avoid extremely small gradients to the Softmax function which will be applied later. A padding masking is added at this point, which

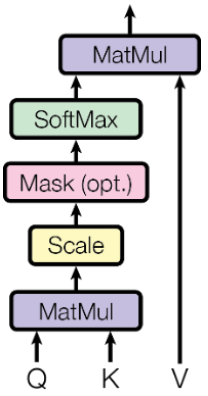

Figure 2.3: *Scaled Dot Product Attention*

substitute all the padded values in the input with a 1 and all the other values with a 0. This is done because all the input vectors have not the same length, so they must be padded before they go into network. We add a padding mask in order to make softmax ignore those values and output a zero for them. At this point, the normalization is performed with a Softmax function which ensure that all the weights sum up to one:

$$\sum_{j=0}^{n} W_{ij} = 1. \tag{2.26}$$

After the softmax normalization, the normalized weights are multiply with the original values and we get the complete 'Scaled Dot-Product attention', which is presented in the figure 2.3. This is the original formula presented in the paper [6] for the A attention function 2.3 :

$$A(Q, V, K) = Softmax(Q * K^T / \sqrt{d_k})V \tag{2.27}$$

where Q,V,K stands for *queries*, *keys*, *values* and 'd_k' is the dimension of each of these layer.

To ensure the learning process, a gradient signal is than back-propagated through the Scaled Dot-Product attention block to update all the weights.

Multi-Head attention block

The multi-head attention block, shown in Figure 2.4, is an improvement of the Scaled Dot-Product attention block, which allows the model to jointly process information from multiple representation subspaces at different positions. It is composed of 'h' different layers of such block, which works in parallel and do not share any weights. Each of these blocks is called head and has dimension equal to the input dimension divided the number of heads. This is an important advantage in term of the total computational complexity, as its overall cost is like the one of a single-head attention with full dimensionality. Each of the heads, or parallel self-attention layers, pay attention on only a particular portion of the input, thus learning different representations of the same input embeddings simultaneously. The output obtained at the end of each layer from each head is finally concatenated with all of the other and feed to a dense layer.

With a single attention head, averaging inhibits this:

$$MH(Q, V, K) = Concat(head_1, \ldots, head_n)W^o \tag{2.28}$$

$$head_i = A(QW_i^Q, KW_i^K, VW_i^V) \tag{2.29}$$

where MH is the function represented by the Multi-Head attention block and where $W_i^Q \epsilon R^{d_{model} x d_k}$,$W_i^K \epsilon R^{d_{model} x d_k}$, $W_i^V \epsilon R^{d_{model} x d_v}$, $W_i^O \epsilon R^{h d_v * d_{model}}$ are the different learned projections matrices of the of query, keys and values. d_k is equal to d_v, which is calculated as the embedding size divided by number of heads, and W_i^o is a square weight matrix which linearly projects all the learned representations to the original embedding dimensionality.

Linear layer and Residual Connection

The multi-head attention output is added to the original positional input embedding, which is called residual connection, and then feed to a layer of normalization. The normalized residual output are feed to a second sublayer added at the end of the encoder. This last one is a pointwise feed forward network, which consists of two fully-connected layers with relu activation. It is applied to each position separately and identically. The output is again added to the residual connection, which this time

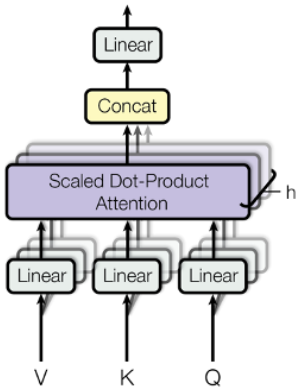

Figure 2.4: *Multi Head Attention*

is the result obtained before this second sublayer and then, normalized again. Residual connections help the network training process as they allow the gradient flow through the network directly. Instead, layer of normalization are used to stabilize the network for faster convergence, while the pointwise feed forward network is used to projects the attention output for a richer representation.

Encoder and Decoder Structures

The transformer architecture proposed in [6], depicted in Figure 2.5, includes both an encoder and a decoder structure. The encoder is composed of six identical layers, each of which is divided in two sub-layer. The first is the multi-head attention layer, followed by the first layer normalization of its output added to the residual connection of the positional encoding, and the second is the position-wise fully connected feed-forward network, which is followed by the second normalization layer of its output plus another residual connection.

The decoder has the the same structure of the encoder, but in addition it shows a second stacked multi-head attention layer. This last one performs multi-head attention over the output of the encoder stack. Residual connections are also here employed around each of the sublayers, followed by the layers of normalization. A modified self-attention sub-layer in the decoder stack is used to prevent positions from attending

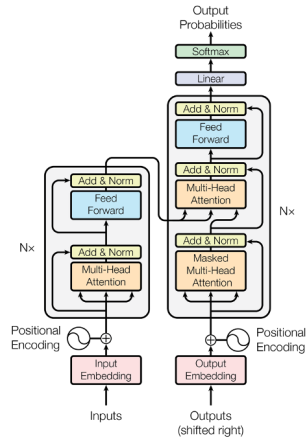

Figure 2.5: *Transformer architecture, with its encoder and decoder structure*

to subsequent positions.

A key aspect of this architecture is the 'Mask Language Modeling' MLM approach, which allows the training of bidirectional architectures. In the Figure 2.3 we can see an optional masking layer, which is used to mask the 15 percents of the input tokens in the pretraining stage before feeding the input to network. By masking some tokens randomly, which are replaced by other tokens, the model attempts to predict the original value of the masked words, based on the context provided by the other, non-masked, words in the sequence.

2.2 Triplet Loss Background

In this section we will explain what is Triplet loss function and how it can be implemented for similarity tasks with Siamese Networks, what kind of triplets can be generated from the dataset and their different mining's strategies.

2.2.1 Triplet Loss

Triplet loss is a cost function presented for the first time in 2015, in the paper [7], which employed it for images recognition tasks and similarity. The basic idea of this technique, which is visually summarized in Figure

2.6, is to train the network to learn a distributed embedding vector space in which similar near-by-region points are data which share some similar characteristics. The implementation of the triplet loss requires a first step where the training set should be reorganize in batches of triplets composed of an anchor, a positive and a negative sample, such that the first two are very similar and belong to the same class, while the third one is very dissimilar from the others two. The objective is to produce embedding vectors in the output models layer, which are a compressed representation of the anchor, positive and negative input samples, and to minimize the distance between the similar vectors and maximize the distance between dissimilar vectors, according to a specific metric. The size of the embeddings is a new hyper-parameters, because its must contain a good enough representation of the input data. The results, at the end of the learning process, is a metrics space where similar data are projected in near region which are visually condensed in clusters.

Metrics

The distance between the embedding vectors can be calculated with different metrics. The most popular one is the euclidean distance we see in the following formula:

$$L = \sum_{i=0}^{N} [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha] \tag{2.30}$$

But there is also the possibility to use a square euclidean distance, which is an euclidean distance which does not perform the square root on the results. The final loss over a batch of triplets can be calculated with different margin strategy:

- 'max margin': in which the margin is set to one and the final loss is the maximum between zero and the loss plus margin;
- 'soft margin': in which the margin depends on the loss itself according to the function 2.31 and it is calculated as the maximum between zero and the loss plus margin;

Normally, the margin is set to a standard and constant value of 0.2.

$$loss = \log(1 + e^{loss}) \tag{2.31}$$

Siamese Network

The implementation of the triplet loss require *Siamese Neural Networks*, as they are able to process multiple inputs in parallel and return a single value loss calculated over the multiple outputs. They are identical networks, which means they all share the same weights and so every input is evaluated with all the same parameters. Given three different inputs, Siamese Neural Networks produce in their output layer a single output embedding vector f(x) Rd composed by all the concatenated outputs. This last one is usually L2-normalized, which means it is forced to rely on a ideal unit hyper-sphere. These embedding vectors are passed to the triplet loss layer and used to calculate and minimize the error:

$$L = \max(0, D(f(x_i^a) - f(x_i^p)) - D(f(x_i^a) - f(x_i^n)) + \alpha) \tag{2.32}$$

where:

- f(x_i^a) is the embedding vector for the anchor;
- f(x_i^p) is the embedding vector for the positive;
- f(x_i^n) is the embedding vector for the negative;
- 'α' is the margin, used to enforce the distance between positive and negative samples. The alpha/margin value makes sure that the network is not allowed to output the insignificant solution where all embeddings vectors are zero or contain the same values;
- i is the i-th triplet;
- D is the metrics to compute the distance;

Figure 2.6: *The triplet loss is a loss function which try to minimize the Euclidean distance between the positive and the anchor sample, while maximizing the one between anchor and negative samples*

Type of Triplets

There can be three different categories of triplets:

- Easy triplets are triplets with zero loss as the distance between the anchor and the positive sample plus the margin is less than the one between the anchor and the negative sample D(f(x_i^a)-f(x_i^p)) + 'α' < D(f(x_i^a)-f(x_i^n));
- Hard triplets have the negative sample is closer to the anchor than the positive sample, D(f(x_i^a)-f(x_i^n)) < D(f(x_i^a)-f(x_i^p));
- Semi-hard triplets are the triplets where the distance between the anchor and negative sample is between the anchor-positive distance and the anchor-positive distance plus the margin D(f(x_i^a)-f(x_i^p)) < D(f(x_i^a)-f(x_i^n)) < D(f(x_i^a)-f(x_i^p)) + 'α';

These definition rely on the position of the negative sample relatively to the anchor and positive, so they can be renamed as 'easy negative', 'hard negative' or semi-hard negative' triplets. Distinguish different kinds of triplets helps to evaluated the different performances over the training process and see where and when they show better results. Hard triplets make the learning process faster as they force the network to learn more as it must push away the negative from the anchor sample. The easy triplets, instead, performs poorly. In contrast to hard triplets, they make the convergence of the gradient very slow. However, the use of the only hard triplet would requires a very high computational cost in case of big datasets, as it would be time consuming search for the hardest triplets

in a large number of examples. A good trade-off can be found in semi-hard triplets, which are easier to mine and show relatively good results comparing the accuracy and computational time.

Mining Strategies

We have talked about what type of triplets which can be selected, but now we focus the attention on how to sample them from the dataset. There are two main techniques:

- Offline mining computes the triplets before the training time on the entire dataset. This is the less efficient in term of computational cost. In this scenario, creating a batch of N triplets requires the network to compute N*3 embeddings.
- Online mining, instead, needs to compute just N embedding vectors to generate N^3 triplets. So in this case, the network generates the triplets as all the possible combination of the N embedding vectors. Of course, a large parts of these triplets are not valid, so there is the need to select the valid ones, for example using the batch hard triplet strategy.

3
Proposed Approach

In this chapter we will go through the process to build and prepare the setup for our experiments. We will talk about the dataset, the features extraction process and the architectures we employed for the generation of an euclidean metric space for music similarity. We will briefly dive into the triplet loss function to train the network and the mining triplet strategy. In addition, we will talk about the interface we create for the data visualization in a 2D space.

3.1 General Formulation

The objective of this study is the generation of an Euclidean metrics space for music genre similarity, which will be used to study the evolution of different music trends through years. As discussed in Section 1.2, different approaches have been followed to treat the music similarity problem. However, to the best of our knowledge, the use of transformers with an offline triplet loss function, is not still explored in the field of music similarity. In our study we will focus our attention on triplet loss as loss function on the top of two different networks, a CNN and a encoder

structure of the transformer. We will compare the two methods, as they are differently able to learn data representation because of their different structures. We firstly treat the problem with a baseline approach, training a CNN with an offline triplet loss and mining the triplets according to the genre label only. Then we proceed to explore the results showed by the transformer with a triplet mining based on genre, id track and artist label.

3.2 Dataset and Features

Following we provide a brief overview on the dataset we employed and the parameters we set to extract the features mentioned in Section 2.0.1.

3.2.1 Dataset: FMA

The choice of the dataset plays an important role for the success of a scientific research, because the results will directly depend on data to work with and the metadata it provides. So a dataset must satisfy a number of requirements which may vary from case to case. In general, a good dataset should have a number of balanced examples which must be representative, that is to say covering all the relevant classes of interest for the study. It must be large enough to avoid over-training and to effectively learn models incorporating inconsistencies in the data. Another important aspect in the case of a music dataset, is the availability of high quality audio raw data, which allows to extract any kind of features, with the needed level of detail and precision. This raw data must also have a permissive licensing to allow the redistribution. Taking care of the above requirements, one the most reasonable dataset for this study is FMA [8] 'Free Music Archive', which is a free and open library managed by WFMU, the longest-running radio station in the United States. FMA is an open and easily accessible dataset employed in several tasks in MIR, providing 106,574 tracks from 16,341 artists and 14,854 albums, arranged in a hierarchical taxonomy of 161 genres. It also provides a smaller and balance dataset, FMA small, which is more suitable to test the performances of the architecture, so we decide to work on this subset, as it also contain eight representative music macro-genres with 8.000 tracks of

30s each. This datasets, as mention before, provides a considerable number of metadata for each track as for each artist and album present in the dataset. This metadata include also the released date of the tracks, which in our case, result to be fundamental for the sake of this study.

3.2.2 Features

The feature extraction process starts with the removal, from the dataset, of those tracks which last zero seconds or very few seconds, which where maybe wrongly included in the dataset. To compute the features we use *Librosa* [49], a python library for music and audio analysis largely employed in MIR tasks. For baseline approach, we extract only 20 MFCC coefficients by 130 frames, with a sample rate of 22050 Hz. Instead, for the final test, we choose to extract five different features including MFCC, MFCC delta, Mel Coefficients, CQT and Chromagram for each thirty seconds tracks. We load each audio track with a sample rate of 44100 Hz and split them in three seconds segments, getting ten samples per track. For each segment we compute the STFT using a hamming window, with 2048 window length, 1024 hop length and 2048 NFFT length. Finally, we compute 20 MFCC , 20 MFCC delta, 128 Mel, 144 CQT and 12 Chroma coefficients, for a total vector length of 324 features per 130 frames.

Given the heterogeneous nature of the features, we employ a scaling procedure in order to scale them in the range of value between -1 and 1. We choose the scaling instead of the normalization because, in this study, we treat models which use the Euclidean Distance as loss function and thus they are sensitive to the magnitude of the distance. With this approach, we are instead able to compare different variables on equal grounds.

3.3 Model Architectures

In this section we take an in-depth look at the two architectures mention in the subsection 2.1.2 and in 2.1.3, paying attention to the structure, to all the regularization inserted to avoid overfitting and the setting of the hyperparameters.

3.3.1 Convolutional Neural Network

The first architecture we propose is a 'Convolutional Neural Network'. In particular we use a three convolutional layers architecture, with 'ReLU' activation. We set the padding option to 'same', so that each layer's output have the same spatial dimensions as its input. After each convolutional layer, a Max Pooling 2D layer is inserted to downsample the input along its spatial dimensions by taking the maximum value over an input window, whose size is defined by the parameters 'pool size' for each channel of the input. We set those size to (3,3) for the first layer and (2,2) for the following two. In the first layer of Max Pooling, the window is shifted by strides (3,3) along each dimension and in the other two by (2,2). In order to avoid overfitting of the network, after each Max Pooling layer, Batch Normalization layers are added as they accelerate the deep network training by reducing the internal covariate shift, as we learn from [65]. As the network easily goes in overfitting, two additional layers of dropout with rate 0.3 are added after the last two Batch Normalization layers. Next, we insert flatten layer to reduce the input dimension from two to one and feed the output to a dense layer with 1024 neurons, which will be the embeddings vectors size of the network. A last L2 Normalization layer, with epsilon parameter set to e-16, in order to normalized number smaller than e-12, is added to control the output and normalize the hidden units constraining the representation to be on a hypersphere. This architecture is summarized in Table 3.1 and we call it 'base model'. The input of the triplet loss architecture is composed of three input layers, corresponding to the anchor, the positive and the negative input sample, as we can see from the Table 3.3. We then compile a new model, with the triplet loss input layers as input new inputs of the base model. In this way we create Siamese Networks [66], which works in parallel for each input sample and share weights. The three outputs of this new model are are finally concatenated in a single output vector. The architecture shown in Table 3.1 and in Table 3.2, refers to the architectures employed for our baseline approach, in which we use just 20 MFCC coefficients as features and the final test, in which we use the complete features vector.

Layer	Output Shape	Activation Function
Conv2D	(None, 20,130,64)	ReLU
Max Pooling2D	(None, 7,44,64)	None
BatchNormalization	(None, 44,64)	None
Conv2D	(None, 7,44,128)	ReLU
Max Pooling 2D	(None, 4,22,128)	None
Batch Normalization	(None, 4,22,128)	None
Conv2D	(None, 4,22,256)	ReLU
Max Pooling 2D	(None, 2,11,256)	None
Batch Normalization	(None, 2,11,256)	None
Flatten	(None, 5632)	None
Dropout	(None, 5632)	None
Dense	(None, 1024)	ReLU
Lambda	(None, 1024)	None

Table 3.1: *CNN base model architecture composed of three Conv2D layers, one flatten layer, one dense layer and a final lambda layer for the L2 normalization. The input is refered to the features vector with only the MFCC coefficients.*

3.3.2 Encoder Structure of Transformer

The architecture of the transformer that we use includes the only encoder's structure, as we are dealing with a regression problem, which does not require to generate an output, but instead, it need an encoder to learn a latent space of the data. This last one is finally connected to an output dense layer with 1024 neurons and a 'sigmoid' activation function, followed by a lambda layer to perform the L2 normalization on the output embedding vectors. The encoder follows the same structure proposed in the original paper [6], which is composed of two sub-layers, the multi-head attention layer and the point-wise fully connected network, with residual connection and normalization and add layers. The first sub-layer is the multi-attention layer, which is build through a model subclassing implementation using the 'tensorflow.keras' functional API, as it let to mix different style of APIs and it is followed by a normalization layer in which the output of the multi-head attention layer plus the positional encoding is normalized to accelerate the convergence of the

Layer	Output Shape	Activ Func	Params
Conv2D	(None, 324,130,64)	ReLU	640
Max Pooling2D	(None, 108,44,64)	None	0
BatchNormalization	(None,108, 44,64)	None	256
Conv2D	(None, 108,44,128)	ReLU	49280
Max Pooling 2D	(None, 54,22,128)	None	0
Batch Normalization	(None, 54,22,128)	None	512
Conv2D	(None, 54,22,256)	ReLU	131328
Max Pooling 2D	(None, 27,11,256)	None	0
Batch Normalization	(None, 27,11,256)	None	1024
Flatten	(None, 76032)	None	0
Dropout	(None, 76032)	None	0
Dense	(None, 1024)	ReLU	77857792
Lambda	(None, 1024)	None	0

Table 3.2: *CNN architecture with takes as input the complete set of features. It is composed of three Conv2D layers, one flatten layer, one dense layer and a final lambda layer for the L2 normalization.*

network. The second sub-layer is the point-wise fully connected network composed by two dense layers, which have respectively 1024 and 'model's depth' neurons, followed by the second normalization layer, where the output of the point-wise fully connected network is added to the output of the first normalization layer and normalized again. Moreover, we add dropout layers at the output of each sub-layer before the normalization layers to prevent the network from overfitting, as described in [67].

Furthermore, as the parer [37] suggests, we insert a dense layer with a linear activation function before adding the positional encoding to the input. This last paper states that linear projection improves the performance of the transformer of the 17 percent with respect to just adding the positional encoding, which, alone, would be even harmful to the learning process. We also add an encoding masking layer which prevent the softmax function to paid attention to the padded tokens. By setting the mask vector to a value close to negative infinity where we have padding tokens and one otherwise, we don't let softmax output some values for them. At the end of the encoder layer, the output embedding vectors

Layer	Output Shape	Activ Func	Params
Input Layer	[(None, 324, 130,1)]	None	0
Input Layer	[(None, 324, 130,1)]	None	0
Input Layer	[(None, 324, 130,1)]	None	0
Functional	(None, 1024)	None	78040832
Concatenate	(None, 3072)	None	0

Table 3.3: *CNN input model for triplet loss. The input is refered to the complete features vector.*

Layer	Output Shape	Activ Func	Param
InputLayer	[(None, None, 20)]	ReLU	0
TFOpLambda	(None, None)	None	0
TFOpLambda	(None, None)	None	0
Enc Padding Mask	(None, 1,1,None)	None	0
Encoder	(None, None, 20)	ReLU	14476
TFOpLambda	(None, 2600)	None	0
Dense	(None,128)	None	332928
Lambda	(None, 128)	None	0

Table 3.4: *Transformer model composed of 2 layer, 5 heads, one final Dense layer of embedding vector shape 128 and a final Lambda layer for L2 Normalization. The input is refered to the baseline approach.*

are reshaped to be feed to a dense layer with 1024 neurons and finally to a lambda layer which performs the L2 normalization. In out better implementation of the transformer, the encoder's output is fed to a dropout layer and than to a 'Max Global Average Pooling 1D', which reduce the dimensionality of the vectors. The Tables 3.4 and 3.5 show the two transformer architectures for the two different tests based on two different input features. The first refers to the 20 MFCC coefficients input and the second to the complete features vector. We implement the triplet loss architecture according to the same implementation we used for the CNN, as we can see from the Table 3.6.

Layer	Output Shape	Activ Func	Params
InputLayer	[(None, None, 324)]	ReLU	0
TFOpLambda	(None, None)	None	0
TFOpLambda	(None, None)	None	0
Enc Padding Mask	(None, 1,1,None)	None	0
Encoder	(None, None, 324)	ReLU	13154052
Dropout	(None, None, 324)	None	0
Global Average Pooling 1D	(None, 324)	None	0
Dense	(None,128)	None	332800
Lambda	(None, 128)	None	0

Table 3.5: *Transformer base model composed of 12 layer, 27 heads, one final Dense layer of embedding vector shape 1024 and a final Lambda layer for L2 Normalization. The input is refered to complete set of features.*

Layer	Output Shape	Activ Func	Params
Input Layer	[(None, None, 324)]	None	0
Input Layer	[(None, None, 324)]	None	0
Input Layer	[(None, None, 324)]	None	0
Functional	(None, 1024)	None	13486852
Concatenate	(None, 3072)	None	0

Table 3.6: *Transformer input model for triplet loss. The input is refered to the complete features vector.*

3.4 Triplet Loss

In this section we describe our architecture of the triplet loss function, the strategy we use to optimize the learning process of the network with new triplets mining according to multiple labels and how we compute the final loss over a batch of samples.

3.4.1 Offline Triplet Loss

In this study we use an offline version of the triplet loss. As a first step, we set the batch size of the triplets we feed the network at each time step and then create a batch generator to mined that number of triplets for

the training set. Inside the generator, two randomly and distinct labels are selected from the dataset. Then, two random and distinct samples are selected from all those examples sharing the first label. We call them the 'anchor' and the 'positive'. Finally, a third sample, to complete the triplet, is randomly selected from all those examples in the dataset having the second label and call it 'negative'. We repeat this process N times, where N is the 'batch size', and we finally get our batch of triplets.

In our first version, we generate the triplets according to the genre only. However, we experimented that other information should be taken into account in the selection. For this reason, anchor, positive and negative samples are chosen according to three different metadata, which are the genre labels, the id track labels and the artist labels. At each step, the triplets are mined according to on of these three labels with a certain probability. The probability of being a triplet based on genre is 20 percent, while the probability of being a triplet based on id track or based on artist is 40 percent. These last two labels are intended to force the learning process to acquire knowledge beyond the only genre information. A song is more similar to itself with respect to all the others. Similarly for the artist labels. Two song are similar if they belong to the same genre, but they are probably even more similar if the are produced by the same artist.

The triplet loss function we use to calculate the loss is an euclidean distance which can be summarized in the formula 3.1:

$$L = \sum_{i=0}^{N} [||f(x_i^a) - f(x_i^p)||_2^2 - ||f(x_i^a) - f(x_i^n)||_2^2 + \alpha], \tag{3.1}$$

where f is the function that maps each input sample to an output embedding vector, x_i^a, x_i^p, x_i^n, are respectively the anchor, positive and negative sample of i-th triplets in the batch.

The final loss is calculated as the arithmetic average of all the losses in a batch.

3.5 Visualization

In this section we will show how the output embedding vectors predicted by the transformer's architecture are process to be better visualized. We will talk about the graphic interface we designed, its main characteristics and the different kind of visualization it offers to let a better visualization of music trends through years.

3.5.1 Samples' reduction

As we described in Section 3.2.2, we split each 30 second track in 3 second rows segments, resulting in ten segments per track. After the training process and the TSNE [68] dimensionality reduction of the the output embedding vectors, we want to have just a just single sample to describe each track, instead of ten. For this purpose, for each track, we select all the segments belonging to its track id and perform the local outlier factor's [69] function by sklearn, which measures the local deviation of density of a given sample with respect to its neighbors. With this method, all of the samples which have a substantially lower density than their neighbors are considered outliers and are removed from the datasets. In a final step, we take all of the remaining samples per id and perform a mean average over their spatial coordinates, thus resulting in a single representative point per track.

3.5.2 Graphic Interface

The graphic interface, in Figure 3.1, is designed in order to support the visualization of music trends through years. We code it in 'Processing' [70], which is an environment and a programming language, which allows a simple approach to the visual arts. Figure 3.1 shows the embedding vector (music pieces) represented as points in space, characterized by a color. The legend on the top right of the window, show the corresponding color-genre mapping for the tracks belonging to 'Free Music Archive', which is the dataset on which we train the network. These genres are Pop, Rock, Experimental, Hip-Hop, Folk, Electronic, Instrumental and International. Black points, which can be shown clicking on the corresponding toggle on the bottom left of the window, instead, represents

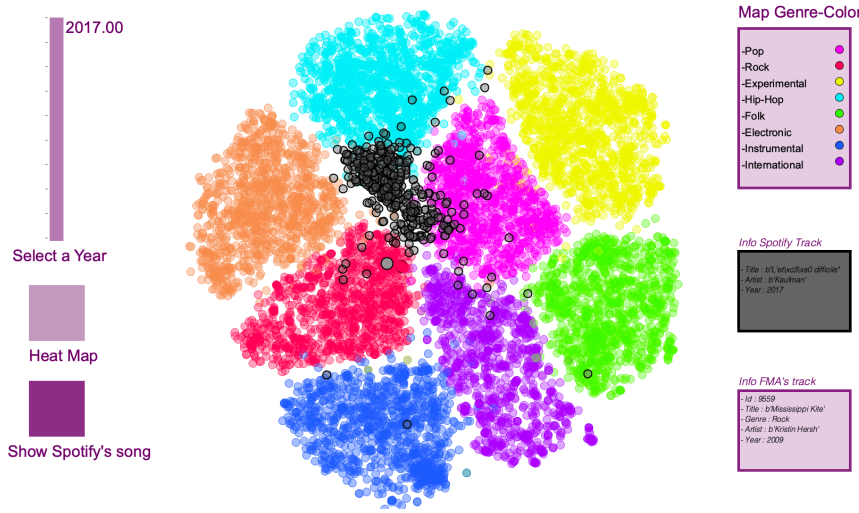

Figure 3.1: *This figure shows our graphic interface, in its first type of visualization. We can see the embedding vectors located according to their similarity in the 2d space. Each color correspond to a genre, as the map Genre-Color represents.*

Spotify's tracks belonging to several playlists we choose ad hoc for this study as test set. On the top left of the interface, a slider allows a custom visualization of all the tracks published in a certain year between 2007 and 2017. If we set the slider to a particular year and then press 'b' on the keyboard, all the songs released from 2007 till that year are shown together at the same time. This functionality is extremely useful, as it allows to add in the 2D space, once per time, all the songs released a certain year, in ascending order. This allow to see where a particular trend is evolving in time, what direction is undertaken and what other genre is going to resemble more. Furthermore, to allow a better understanding of the genre space and also better support the study of the music trends, we introduce two kind of visualizations, witch can be switched by clicking on the corresponding toggle on the bottom left of the interface. The first one plots the embedding vectors as points on a 2D space in a specific year or in multiple years. Near songs are more similar, dissimilar songs are far away from each other. Clicking left on a colored point or clicking left on a black point, the relative songs information appear in a info box

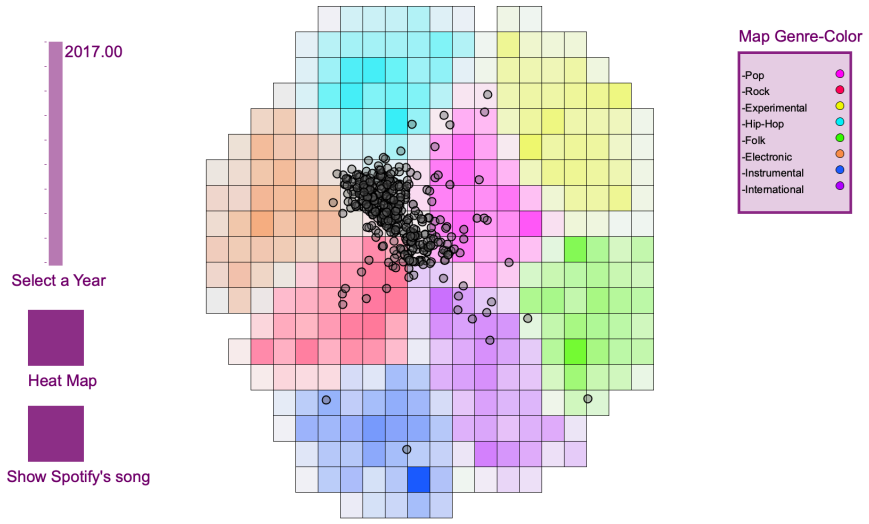

Figure 3.2: *This figure shows our graphic interface, in its first type of visualization. Black points represents Spotify's tracks located in a area between Hip-Hop, Pop, Rock and Electronic genre.*

on the bottom right of the interface, showing metadata about the title, the artist, the genre and the year.

The second type of visualization we propose is a heat map depicted in Figure 3.2, which allows a continuous visualization of the music genre distribution in the 2D space, removing boundaries between clusters of genre. In fact, the heat map shows the concentration of the embeddings vectors belonging to a certain genre in a restricted area of the 2D space. The higher is the embeddings density, the intense is the tone color of the regions of space in which they are located. White areas correspond to region of space where no embeddings are located in that region. This representation allows to see where a particular genre can influenced more a black point, as the darker region works like as center of gravity points, as the darker they are, the more power of attraction have on the near points. Spotify's songs can be visualized over the heat map as points and a similarity to a genre can be given according to the color of their corresponding area. In addition, clicking on a point and then pressing 'i' on the keyboard, we can see the local nearest neighbours points corresponding to that music piece, as we can see in Figure 3.3. Right clicking

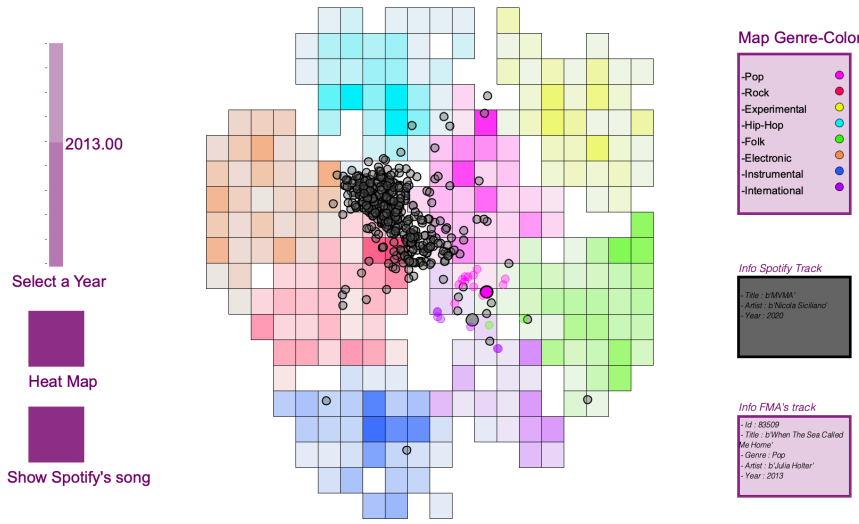

Figure 3.3: *The figure shows a selected point and its relative neighbours, which can be added pressing 'i' on the keyboard.*

on the neighbour points, we are also able to see their information related to the title, artist, genre and year. This is an important functionality, as it allows to know, for each black point, what other songs, from the same year, but also from other years, share its characteristics and some sort of similarity. In this way we are able to identify and trace particular music trends which evolve in years.

4
Experimental Setup and Evaluation

In this chapter we present all the network's hyperparameters settings we have chosen for our experiments and the metrics we have used to evaluate the different results on the different architectures. We will show our predicted embedding output vectors on a 2D space, to provide a visual intuition of the obtained results. Then, we well compare those results in different scenarios to see what architecture performs better. We will also test our best model on Spotify's tracks by some selected most popular playlists in Italy. Finally, we test the accuracy and the performances of our model through a simple test based on human evaluations by three music experts. The value of accuracy provided for the four experiments results in Section 4.1 and in Section 4.2 are referred to the first evaluation metric described in Section 4.3.2.

4.1 CNN Experiments

In this section we describe the experiments conduct on the CNN architecture with two different approaches and hyper-parameter settings. We also compare the obtained results.

4.1.1 Baseline Approach

By an experimental approach, we end up with a set up for our first baseline experiment to compare with the performances of the Transformers. We feed the network, shown in Section 3.3.1, a matrix of 75277 examples of 20 MFCC by 130 time frames. We split the data using the train test split function by sklearn library in order to have the 80 percent of the sample for the training set and the other 20 percent for the test set. We set the generator to build batches of 64 samples and our output embedding dimension to 1024. In this first experiment we select triplets according to the only genre label. We set the margin to a standard value of 0.2 and use a triplet loss function which performs an Euclidean distance between the output embedding vectors pairs, in which the loss is finally calculated as the average value over all the triplets losses in the batch. We choose 'Adam' optimizer with a decaying learning rate, starting from 1e-3, with five sustain epochs, to 1e-4 with an exponential decay of 0.95, as shown in Figure 4.1. The number of steps per epoch is calculated as the number of examples of our training set dived the batch size, which in our case result to be 941 steps per epoch. We set the 'Early Stopped' class of the callbacks to have 30 epochs of patience and the 'Model Checkpoint' class to save the best model's weights according to the minimum loss in the training process. After 140 epochs, the learning process automatically stopped.

After a dimensionality reduction performed with 'TSNE', we plot the embeddings in a 2D space. Figures 4.2 and 4.3 show the results of the training and the test set. We can clearly see how the our model performs well not only on the training set, but also in the test set. Embedding vectors of similar genre are closed in space in cluster of the same color, which are well distinguishable one from the other. It means that the CNN learns appropriately the music similarity space. Instead, very few points are misclassified.

4.1.2 Complex approach

Our last experiment on the CNN is performed with almost the same network's hyparameters described in Section 4.1.1. We feed the network with an input matrix of 75277 examples of 324 features, including all the music descriptors in 2.0.1, by 130 time frames, previously normalized

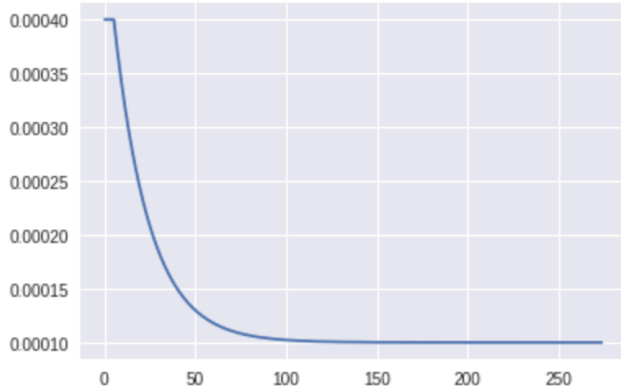

Figure 4.1: *CNN's learning rate for our baseline approach*

Figure 4.2: *Visualisation of the embeddings output vectors generated from the CNN after 140 epochs training. These are the results from the training set in our baseline approach.*

Figure 4.3: *Visualisation of the embeddings output vectors generated from the CNN after 140 epochs training. These are the results from the test set in our baseline approach.*

with the 'MinMAx Scaler' by sklearn, in a range of value between -1 and 1. We mine the triplet according to three label: the artist, the track id and genre labels, with a probability of 30 percent, 35 percent and 35 percent respectively. We train the network obtaining a final accuracy of 75 percent. The visualization of the embedding output vectors of the training and test set is depicted in the Figures 4.4 and 4.5. In this case, genre cluster's boundaries overlapped and seems to be more sparse in the space. On one hand, this could be a positive result as music, as we say in our introduction, can not be constrained in closed genre boundaries. On the other hand, we see that the prediction on the test set results to be less accurate, as there are much more misclassified points than before.

4.2 Transformer Experiments

In this section we describe the experiments conducted on Transformer in different scenarios, focusing on the hyperparameters selection and the model complexity. We will see in which of the following experiments Transformer performs better and in which is less performative.

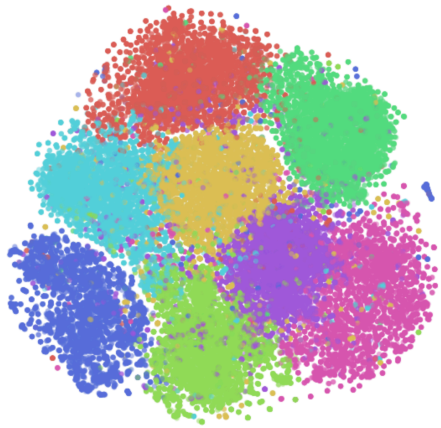

Figure 4.4: *Visualisation of the embeddings output vectors generated from the CNN . These are the results generated by the training set in our complex approach.*

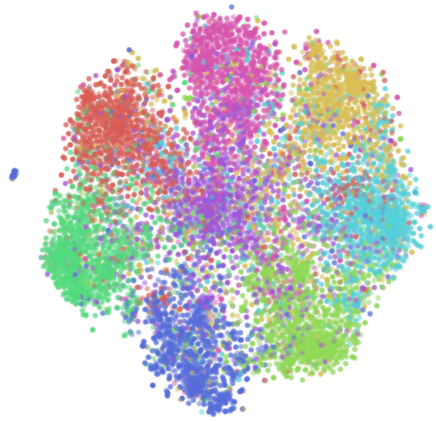

Figure 4.5: *Visualisation of the embeddings output vectors generated from the CNN . These are the results generated by the test set in our complex approach.*

4.2.1 Transformer Baseline Approach

First experiment

In this first experiment, as we done for the CNN, we feed the transformer a matrix of 75277 examples of 20 MFCC by 130 time frames. We split the data using the sklearn function 'train test split' with the test size parameters set to 0.2. We set the generator with a batch size of 64 and perform the triplets mining according to the genre label only. We conduct this test keeping low the complexity of the model, as Transformers require very high computational resources and time, so we increased its complexity progressively. We use two layers of encoder and five heads in the scaled dot product attention. We insert some dropout in order to avoid overfitting with a rate of 0.1 and set the units of neurons in the dense layers and the output embedding size to 128. We set a 'relu' activation on the output dense layer. After that, we insert an L2 normalization layer in order to keep the embeddings restricted inside a unit hypersphere. The training steps are calculated as the number of training examples divided the batch size, so in this case 941. We set the 'Early Stopped' class of the callbacks to have 30 epochs of patience and the 'Model Checkpoint' class to save the weights of best model according to the minimum loss in the training process. We use a decaying learning rate with starting value of 1e-5 with 941 warm up steps. In two epochs this value reaches 5e-5 and remain fixed for 50 epochs and finally it exponentially decay to 1e-6. Figure 4.6 shows the learning rate function. We train the transformer mining the triplets according to the genre labels and get for each step the final loss as the average within all the losses calculated in a batch. After 200 epochs the Transformer ends up with a final accuracy of 56 percent. This first experiment with the transformer doesn't show any improvements compared to the one of the CNN. Instead, it performed worst. Figure 4.7 shows the Transformer

loss function in the training stage. As we learned from the paper [37], Transformer architecture's performances reach best results with higher complexity of the model. Pham et al. recommended to use an input model dimension of 512, with 8 heads, 48 layer and 2048 neurons in the encoder's dense layers. In our case, the input model depth is barely 20 MFCC's coefficients, which results in very a poor performance. Figure

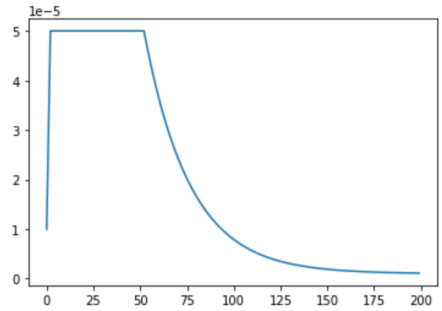

Figure 4.6: *Visualisation of the learning rate's function we set up for the transformer baseline's approach.*

4.8 and 4.9 show the visualization of the embedding output vectors in a 2D space, after a dimensionality reduction performed with TSNE. In this case, the clusters are completely overlapping and it is very hard to distinguish were a genre is more present than other. The same results is obtained both for the training and test set, in which the Transformer performs worst than CNN.

Second experiment

In the second experiment, we use the same settings in 4.2.1, but with a considerable increasing of the model complexity, to make the number of parameters be comparable with the one of the CNN in 4.1.1. We set the number of encoder layers to in 8, we use 512 neurons per dense layer and add a 1D Global Average Pooling layer before the last dense layer. We set the dropout rate in the encoder to 0.3 and 0.2 in the output one. We use a 'gelu' activation and set the margin's value to 0.3, but also this approach leads to very poor results.

4.2.2 Transformer complex approach

The following experiments are conduct on Transformer with different hyperparameters. We feed the network 75277 examples, consisting of 324 features per 130 time frames. The matrix is previously normalized with the 'MinMAx Scaler' by sklearn, in a range of value between -1 and 1.

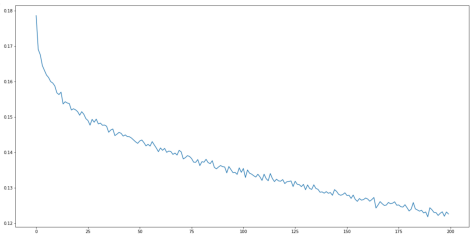

Figure 4.7: *Visualisation of the transformer loss function after 200 epochs of training. The figure suggests that more epochs would probably reduce the loss's value, but that anyway the loss's decreasing is too slow.*

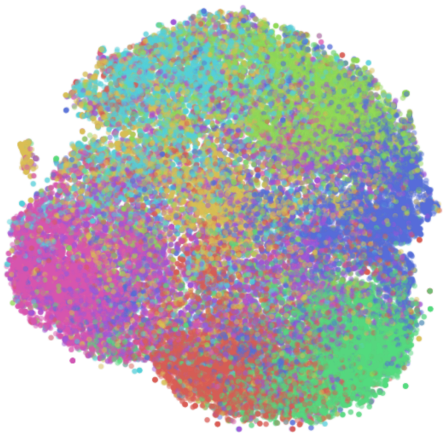

Figure 4.8: *Visualisation of the output embeddings vectors generated from the transformer after 200 epochs training. These are the results from the training set in our baseline approach.*

Figure 4.9: *Visualisation of the output embeddings vectors generated from the transformer after 200 epochs training. These are the results from the test set in our baseline approach.*

First experiment

We start the first experiment training the network with a low complexity of the model, for the same reasons we discussed in 4.2.1. We use the following setup: 8 layer, 27 heads, 512 units in the dense layers and 128 output embedding size. We use batches of 32 examples and change the final activation function in the last dense layer before the l2 normalization with *'gelu'* activation. We train the transformer for 200 epochs, with 941 steps per epoch, using a starting learning rate of 1e-5, sustained for 100 epochs and than an exponential decaying learning rate ending to 1e-6. This first result is quiet improved with respect to the previous one, with a final accuracy of 86 percent.

Second experiment

To considerably improve the performance of the transformer as compared to the previously first experiments, we set the number of encoder layers to 12, the number of heads to 27, the number of units neurons in the dense layers to 1024 and the embedding vectors output to 1024. We increase the warm up steps of the the learning rate function to 941*2 . We mine the triplets according to three different labels, related to the

Figure 4.10: *Visualisation of the output embeddings vectors generated from the transformer in our best experiments of the complex approach. These are the results from the train set.*

genre, the id track and the artist, with a probability of 35, 35 and 30 percents respectively. We use a rate of 0.11 for the dropout and sigmoid activation function in the last dense layer before the l2 normalization. We train the network for 200 epochs, reaching a final accuracy of 88 percent. We deduce that the reasons for the poor previously results is the low dimensionality of the data we feed in the previous experiment and the little number of layers, heads and units, which over reduce the model complexity.

Third experiment

In this last experiment, we used the same settings in 4.2.2, but we increase the dropout rate to 0.3 in the encoder structure and to 0.2 in the output layer. We set the probability of mining a triplets with a certain label to 20 percent for artist label, 30 percent for id label and 50 percent for genre label. We also increase the triplet loss margin to 0.4 and finally, we reduce the minimum learning rate to 1e-7. Following the guidelines of [71], we also reshape the matrix according to this formula:

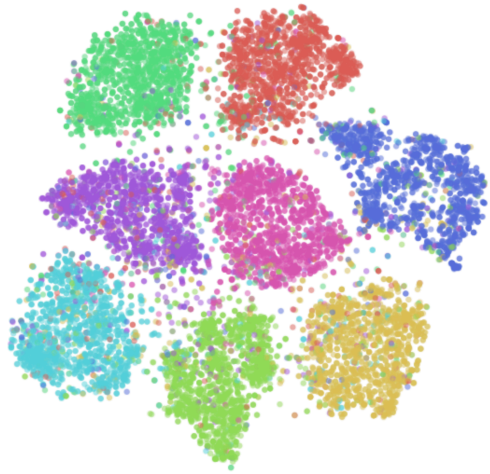

Figure 4.11: *Visualisation of the embeddings output vectors generated from the transformer after 200 epochs training. These are the results from the test set in our baseline approach.*

$$X \in R^{(lxd)} -> reshapeX \in R^{(\frac{l}{a}xad)} \tag{4.1}$$

where l is the temporal dimension, d is the model depth and a is a reshaping factor. After 200 epochs of training, the result is still more improve to 91 percent of accuracy. The resulting the output embedding vectors can be seen in Figures 4.10 and 4.11. In the case of the training set, points are almost perfectly located inside the correct cluster. Also the embedding vector predicted from the test set show very good results, as the cluster are well distinguishable and separated by a large margin one from the other in space.

4.3 Evaluation Metrics

In this section we will present about the dataset employed as test set to evaluate the performance of the system on new and unseen data. The we will describe the two metrics used to evaluate the models.

Network	Baseline Accuracy	Complex Approach Accuracy
CNN	90	75
Transformer	56	91

Table 4.1: *Comparisons of the four experiments' results according to the first metric proposed with K-neighbour classifier. CNN seems performing better on a low dimensional input, while the transformer outperforms the CNN when it receives high dimensional input value.*

4.3.1 Spotify's Playlists

To identify and to study the evolution of trends, we decide to use tracks from only Italian artists. We choose the most quoted playlists on Spotify in the last five years including different sub-genres of hip hop as trap, drill and rap, and in addition, a new popular Italian style in the last five years, called 'indie' music. We download and process tracks taken from six selected playlists specially chosen for the purpose of this thesis, for a total amount of 500 tracks. We download them through the Spotify's API, which let us download thirty seconds for each track with additional metadata including title, artist, year and genre.

4.3.2 K-nearest neighbours Classifier

To evaluate the performances of both networks, we use the following system metrics. We use a machine learning classifier, the 'K-nearest neighbours' [72], to learn the train output embeddings, setting its neighbors parameters, which is the number of neighbours searched for each query point, to 100. We employ a 'ball tree' algorithm [73], which works best for fast generalized N-point problems. First, we use a trained model (CNN or transformer) to predict the embedding output vectors of both the training and test set. Second, we fit the 'K-nearest neighbours' classifier with the predicted output embedding vectors of the training set, with their corresponding true labels. Third, we use the trained classifier to predict the genre labels of the test set. Finally, we count how many of the predicted labels are equal to the true labels of the test set. The resulting accuracy is calculated as the number of correct predictions divided the test set size.

All the results provided by this metric, for all the experiments mentioned before, are shown in Table 4.1. Results show that the CNN is very performative in learning a simple metric space based only on one single features, but decrease its accuracy when it deal with the complete features vector. Transformer, instead, show very poor performances in first experiments. This is due to the fact that transformers are architectures which works best with longer inputs sequence. In the special case of acoustic events, in which adjacent frames share very similar sound characteristics, transformer show its weakness in focus the attention on such lower input sequence of similar events. To the contrary, in the second experiment, it leads to very good results, after we double the dimensionality of the features. We believe that address the problem of the similarity and creating a metric space with just a single features, in this case the 20 MFCC, is to little informative. For this reasons, results provide by the transformer are much more relevant than the one of a CNN, as the only MFCC could not be able to capture different and relevant aspects of the music.

4.3.3 Human Evaluations

The second metric completely relies, instead, on human evaluations. We ask three music experts to evaluate twenty songs, which were randomly selected from the Spotify's playlists. The mentioned experts are a music producer (credits for Sugar, Universal and Sony record label), an artist manager and AR of Music e Orangle labels. The test is conducted individually and secretly, so that each participant could not be influenced by the others.

4.3.4 First part of the test

We submit to the testers the following first three questions:

- Q1 How much do you think the music piece is correctly located in space?
- Q2 How much its relative neighbours are effectively similar to it?
- Q3 How much the music piece is in proximity of the correct cluster?

	Q1	Q2	Q3
Mean	7,2	6,3	7,1
Variance	0.61	0.82	0.49

Table 4.2: *Q1,Q2 and Q3 corresponds to the questions of the test we submitted to the testers. Rows represents the mean and the variance of the overall result.*

For each question, they are asked to answer with an evaluation score in a range between 1 and 10, where 1 represents the poor evaluation, while 10 the best one.

The first question is about the correctness of the absolute location in the space of each query song. In this case, the experts are required to look at the color region where the music piece was located and say how much the color genre match the style of the song. The second question is related to each point neighbours. Given a query song, we ask the experts how much its neighbours effectively sound similar to it and how much the style of the query point matches the style of its neighbours. In the evaluation score also parameters like sound, harmony, melody and rhythm are take into account. The third question ask the expert to evaluate how much the query song is correctly locate in the right cluster proximity or inside of it. For music piece from multiple genre, they have to quantify how much the song location match the proximity of the correct cluster.

In table 4.2, we can see the final results of this test. First we calculate the mean of the three evaluations given by the participants for each song and for each question. Then, we compute the mean of the mean twenty values for each question. The results are shown in the first rows of the table. In second row, instead, we show the variance of the mean of the three evaluations for each song and for each question. In table 4.3 the overall evaluations given by the participants for the first question are shown. Columns represents the twenty query songs. The first three rows show the scores by each expert and the last one the mean value over the three evaluation. Table 4.4 and table 4.5 similarly show the results for the second and third question.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20
V1	5	5	7	8	5	9	7	7	7	8	4	6	8	7	7	8	8	7	6	9
V2	7	8	9	8	7	8	8	7	6	8	6	7	8	5	7	8	8	6	8	8
V3	6	9	8	7	8	7	6	8	9	7	5	9	6	6	8	9	7	7	8	7
Mean	6	7.3	8	7.7	6.7	8	7	7.3	7.3	7.7	5	7.3	7.3	6	7.3	8.3	7.7	6.7	7.3	8

Table 4.3: *This table is show all the evaluations given by the three experts and the mean for the first question.*

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20
V1	7	4	6	6	3	7	3	7	6	7	6	7	7	6	6	7	3	4	8	2
V2	5	7	8	7	8	7	8	6	6	7	6	9	5	8	5	9	8	8	7	6
V3	6	5	8	8	8	8	4	5	4	8	5	9	5	6	8	6	5	6	5	7
Mean	6	5.3	7.3	7	6.3	7.3	5	6	5.3	7.3	5.7	8.3	5.6	6.7	6.3	7.3	5.3	6	6.7	5

Table 4.4: *This table is show all the evaluations given by the three experts and the mean for the second question.*

4.3.5 Second part of the test

Finally, three other questions are submitted to the participants regarding the overall performances of the model. For each question the tester are asked to answer giving an evaluation score in a range between 1 and 10, where 1 is the worst evaluation and 10 the best one. The three questions are here reported:

- Q4 How much the visualization per year is useful ?
- Q5 How much the system is able to visualize music trends?
- Q6 How much do you evaluate the overall system?

The first questions concern the usefulness of the visualization by year. Participants have to evaluate how much help to see only the songs released in the same year to a better comparison of the test songs. The second question ask how much the system is able to show trends and its effectiveness in doing it. The third question requires a final evaluation of the overall system, including all of the aspect took into account in the precedent questions. The table 4.6 shows the results of the final question of the test, where the participants where required to answer question relative to the overall performances of the model.

	S1	S2	S3	S4	S5	S6	S7	S8	S9	S10	S11	S12	S13	S14	S15	S16	S17	S18	S19	S20
V1	8	4	7	7	5	9	6	7	7	8	6	7	7	6	7	8	6	7	6	8
V2	8	8	9	7	5	7	8	8	8	9	6	7	8	6	7	6	7	9	8	7
V3	9	7	8	6	7	7	7	6	7	7	5	7	6	8	6	9	8	8	7	6
Mean	8.3	6.3	8	6.7	5.7	7.7	7.	7.	7.3	8	5.7	7	7	6.7	6.7	7.7	7	8	7	7

Table 4.5: *This table is show all the evaluations given by the three experts and the mean for the third question.*

	Q4	Q5	Q6
V1	4	7	7
V2	7	8	8
V3	8	7	8
Mean	6,3	7,3	7,6
Variance	2.9	0.2	0.2

Table 4.6: *The three columns corresponds to the last three questions on the overall performances of the model. The first three rows show the evaluation given by the participants and final one the mean value and the variance of them.*

4.3.6 Limits shown by FMA and Music consideration

To clearly understand and makes considerations on the final test results, we should before highlighting the weakness of the dataset in the context of our research. First, FMA does not contains a balanced number of songs per year and this makes difficult the comparisons of the query songs with their neighbours in multiple years, as sometimes they do not exist at all. In addition, the fast evolution of the Pop and Hip-hop music in years makes very hard comparing songs by these genres in multiple years, and in our case, we compared relatively recent songs from 2016 to 2021, with music from 2007 to 2017. On one hand, genres like Rock are not affected by the influence of other genres and, in years, they shows very few changing in their sounds. We can think to Rock as a 'stable' genre, which means that if we compare the sounds of different Rock music pieces from different years they are similar. Consequently, Rock music evaluation scores by the experts were higher than other genres in our test. On the other hand, genres like Pop or Hip-Hop suffer from

contamination from the other genres and are continuously changing, year after year, as they do not have their own dimension where to live. Hip-Hop engross beats and sounds by other genres, as in its first years was produced mainly by people who do not have a big knowledge of musics. Pop, instead, tends to be more influenced by the popularity of the main music trend in a year, so it is constantly changing. It follows that the evaluation scores for this genre results to be lower than the others.

4.3.7 Consideration on the results

The first question, related to the first part of the test, gained a 7,2 mean value and 0.61 score for the variance. According to this result, our model can almost correctly classifies the absolute positions of the points songs, but with some uncertainty in doing that. The second question obtained the worst results of the first part of the test, with a mean of 6.3 and very high variance. Second question revealed the main weakness of our model, as FMA song's are much different from the music tracks from Spotify's. Even though similarity genre could be caught by the testers, the evaluations were low due to their poor similarity in sound. The third question, related to the first part of the test, reaches the higher mean value with the less variance, meaning that all the scores are concentrated around that value. In this case, our model seems to be much performative, as the songs are to be located in proximity of the correct clusters. The result of the fourth question is the lowest gained in all the test, which is 6,3. The visualization per year of the FMA tracks lucks of a sufficient number of examples per year, so comparing a query song with neighbours of different year results to be hard. The fifth question gained a final mean vote of 7.3 with a very low variance. It means that the evaluations were consistent in agreeing that this tool can well visualizes trends. The results of the last question of the test revealed a 7,6 mean score on the overall performances of the system. This is a good results, even though it appear to be lower compare with what we expect from the first metric evaluation. The reasons for that could mainly be address to the characteristics and limits of FMA datasets, discussed in Section 4.3.6. A good point to notice is that, all the Spotify's playlists include the most popular tracks in Italy from 2016 to 2021. They included the most popular genres in Italy in these years, including 'trap', 'indie', 'Pop', 'drill'.

The Embedding vector points generated by our model are located in the exact position of space where we expect they would be. In fact, they are organized inside a region of space between Hip-Hop, Pop and Rock. The reasons, for the non excellent results gained by human evaluations, can be addressed to the different kind of music from the American Playlists of the FMA dataset and the Italian songs by Spotify, which hardly conditioned the evaluations. As a matter of fact, the similarity between the adjacent Italian songs in space are higher. This means that, probably, in the metric similarity space we trained, there was no a region where the similarity between the two datasets would be higher, as the training set is very different from the test. To the contrary, the visualization per year helps in tracking an evolution in the popular music in Italy.

4.3.8 Evolution of Pop music

For an experimental validation of the method we study the evolution of Italian songs among the years. We plot the songs by Spotify's playlist year by year over all the embedding points of the FMA dataset. This plot seems to reveal an evolution of music contained in out playlist from Pop to the Hip-Hop. The playlists contain the most popular songs in Italy from 2016 till 2021, so classifiable as Italian Pop music. From the Figures 4.12, 4.13 and 4.14, we can trace an evolution of the black points, which gradually move from the violet Pop cluster to the light-blue Hip-Hop cluster from 2016 till 2021. This fact could be due to the advent of the trap music in Italy in 2016. That year was the pinnacle of success of trap music in Italy where it reach the apex of popularity and gained much of the favour of the audience. While in year before 2010, pop music mainly based on folk music by songwriting, today the influence of American trap music is affecting the Italian artist. With our representation, music trends are now visible.

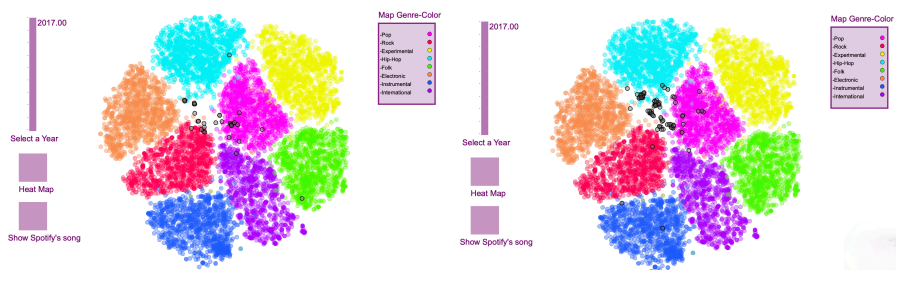

Figure 4.12: The figure shows how the music pieces from Spotify's playlists evolve in years. From the left to the right, the images show the tracks from 2016 to 2017.

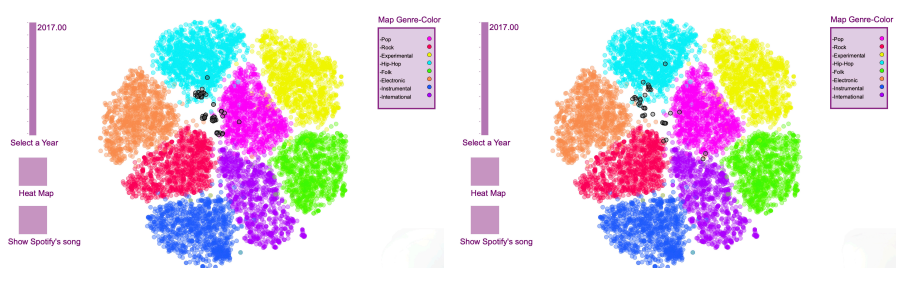

Figure 4.13: The figure shows how the music pieces from Spotify's playlists evolve in years. From the left to the right, the images show the tracks from 2018 to 2019.

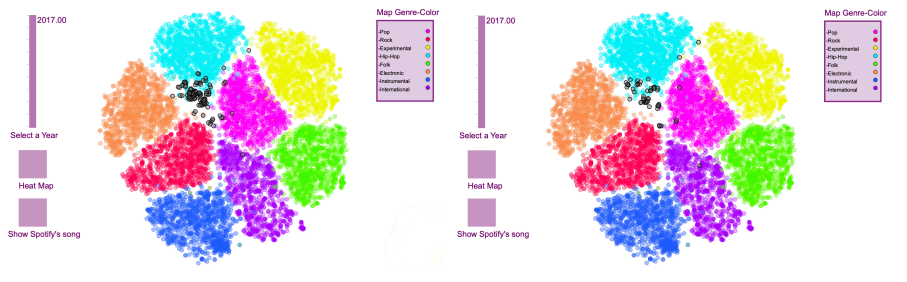

Figure 4.14: The figure shows how the music pieces from Spotify's playlists evolve in years. From the left to the right, the images show the tracks from 2020 to 2021.

5 Conclusions and Future Works

In this ending chapter we will summarise the key findings of our research, giving the final results of the experiments and our conclusion. Finally, we will talk about which could be the possible future works for this thesis, showing how our work could be boost in performances according to several proposed improvements.

5.1 Conclusions

In this thesis we shown a novel approach to study music trends through similarity. We build a similarity metric space employing Deep Neural networks with a the triplet loss function. We tested two kind of architectures for our experiments, which are a 'Convolutional Neural Network', presented in Section 3.3.1 and a 'Transformer' encoder, presented in Section 3.3.2. FMA provided the music we use for our research, for a total amount of 8000 songs. From each 30 seconds of audio, we extracted five perceptive features including MFCC, MFCC delta, MEL, CQT and Chromagram coefficients, as we discussed in Section 2.0.1. We experimented two different kinds of triplet mining, one according to the only genre la-

bel and the other according to the three different labels, including genre, artist and track id, as we discussed in Section 3.4. We experimented as well two different training sets, one containing only 20 MFCC coefficients and the other with a features vector including all the five music descriptors. We created a graphic interface, presented in Section 3.5, to show the embedding output vector generated by the model and plot over them the music from six Spotify playlists in order to test the performance of our best model on unseen data. At the end of the experiments, described in Chapter 4, Transformer outperformed over the Convolutional Neural Network in our complex approach employing the second training set. These results shown the ability of the transformer in learning a more general representation of the music based on multiple features, instead of a single descriptors. Issues related to FMA dataset, discussed in Section 4.3.6, lead to positive human evaluations scores, even though they resulted to be inaccurate in measuring the similarity between the FMA and Spotify's playlists. To the contrary, our model resulted to be performative in locating the songs in the right cluster proximity. The songs by Spotify playlists were located by our model in a region of space between Hip-Hop, Pop and Rock, which is the exact region of the space were we expect they would be.

5.1.1 Triplet Loss improvements

In our research, we used an offline triplet loss. The offline implementation requires to produce 3*N embeddings vectors to obtain N triplets. Furthermore, it requires full pass over the training set. The online version instead, with N embeddings output vectors, generate N^3 triplets as it calculates all the possible combination of the embeddings. In future works we will implement an online strategy for the triplets mining, which will considerably improve our model, in terms of efficiency and faster training, as much less examples are needed. Another improvements, which will further reduce the convergence of the network is the chosen of the valid triplets, which are the hard positive and the hard negative triplets, as we described in 2.2.1. In future approaches, triplets will be selected taking into account another parameter, which is the year, in order to produce a metric space in which similar songs are produced in the same years.

5.1.2 Supervised to Unsupervised learning improvements

In this thesis we followed a supervised approach for the learning process, as we need a labeled dataset to conduct our experiments. This is a big disadvantage as the available and free licensing datasets are few and are not so specific to a particular kind of music. In future works, we will also implement an unsupervised learning technique, which would expands the boundary of music learning, as we will use an incredibly larger number of music tracks and datasets. This approach will be done by feeding the examples to an unsupervised cluster algorithm, such as the 'K-mean cluster', which would naturally aggregate features in a space according to their similarity, and than assigned to each of them a label.

5.1.3 Dataset

Another possible improvements to our work will be the choice of a dataset with more recent songs. It will be choose for a particular case of study. For example, our model is just able to show where Hip-Hop is located, bu not all of its sub-genres. A more informative dataset on this genre would boost the performance of the algorithm in tracing trends, because it will better shows the direction of a particular trends inside a metric space which better resembles it characteristics.

Bibliography

[1] M. I. R. E. eXchange, "Audio music similarity and retrieval."

[2] D. Bogdanov, J. Serrà, N. Wack, P. Herrera, and X. Serra, "Unifying low-level and high-level music similarity measures," *IEEE Transactions on Multimedia*, vol. 13, no. 4, pp. 687–701, 2011.

[3] J. Lee, N. J. Bryan, J. Salamon, Z. Jin, and J. Nam, "Disentangled multidimensional metric learning for music similarity," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6–10, IEEE, 2020.

[4] J. Cleveland, D. Cheng, M. Zhou, T. Joachims, and D. Turnbull, "Content-based music similarity with triplet networks," *arXiv preprint arXiv:2008.04938*, 2020.

[5] L. Prétet, G. Richard, and G. Peeters, "Learning to rank music tracks using triplet loss," in *ICASSP 2020-2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 511–515, IEEE, 2020.

[6] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, pp. 5998–6008, 2017.

[7] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 815–823, 2015.

[8] M. Defferrard, K. Benzi, P. Vandergheynst, and X. Bresson, "FMA: A dataset for music analysis," in *18th International Society for Music Information Retrieval Conference (ISMIR)*, 2017.

[9] G. Kreitz and F. Niemela, "Spotify–large scale, low latency, p2p music-on-demand streaming," in *2010 IEEE Tenth International Conference on Peer-to-Peer Computing (P2P)*, pp. 1–10, IEEE, 2010.

[10] X. Su and T. M. Khoshgoftaar, "A survey of collaborative filtering techniques," *Advances in artificial intelligence*, vol. 2009, 2009.

[11] E. D. Liddy, "Natural language processing," 2001.

[12] A. Bozzon, G. Prandi, G. Valenzise, and M. Tagliasacchi, "A music recommendation system based on semantic audio segments similarity," *Proceeding of Internet and Multimedia Systems and Applications-2008*, 2008.

[13] J. Downie, K. West, A. Ehmann, and E. Vincent, "The 2005 music information retrieval evaluation exchange (mirex 2005): Preliminary overview," in *6th Int. Conf. on Music Information Retrieval (ISMIR)*, pp. 320–323, 2005.

[14] J. S. Downie, J. H. Lee, A. A. Gruzd, and M. C. Jones, "Toward an understanding of similarity judgments for music digital library evaluation," in *Proceedings of the 7th ACM/IEEE-CS joint conference on digital libraries*, pp. 307–308, 2007.

[15] J. Downie, "International music information retrieval systems evaluation laboratory (imirsel): Introducing d2k and m2k," in *Demo Handout at the 2004 International Conference on Music Information Retrieval*, 2004.

[16] T. Lidy and A. Rauber, "Evaluation of feature extractors and psycho-acoustic transformations for music genre classification," in *ISMIR*, pp. 34–41, 2005.

[17] N. Shental, T. Hertz, D. Weinshall, and M. Pavel, "Adjustment learning and relevant component analysis," in *European conference on computer vision*, pp. 776–790, Springer, 2002.

[18] P. Hedelin and J. Skoglund, "Vector quantization based on gaussian mixture models," *IEEE transactions on speech and audio processing*, vol. 8, no. 4, pp. 385–401, 2000.

[19] A. Flexer, "On inter-rater agreement in audio music similarity.," in *ISMIR*, pp. 245–250, Citeseer, 2014.

[20] H. Purwins, B. Li, T. Virtanen, J. Schlüter, S.-Y. Chang, and T. Sainath, "Deep learning for audio signal processing," *IEEE Journal of Selected Topics in Signal Processing*, vol. 13, no. 2, pp. 206–219, 2019.

[21] P. Avgoustinakis, G. Kordopatis-Zilos, S. Papadopoulos, A. L. Symeonidis, and I. Kompatsiaris, "Audio-based near-duplicate video retrieval with audio similarity learning," in *2020 25th International Conference on Pattern Recognition (ICPR)*, pp. 5828–5835, IEEE, 2021.

[22] M. Schedl, "Deep learning in music recommendation systems," *Frontiers in Applied Mathematics and Statistics*, vol. 5, p. 44, 2019.

[23] J. Nam, K. Choi, J. Lee, S.-Y. Chou, and Y.-H. Yang, "Deep learning for audio-based music classification and tagging: Teaching computers to distinguish rock from bach," *IEEE signal processing magazine*, vol. 36, no. 1, pp. 41–51, 2018.

[24] A. Veit, S. Belongie, and T. Karaletsos, "Conditional similarity networks," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 830–838, 2017.

[25] T. Bertin-Mahieux, D. P. Ellis, B. Whitman, and P. Lamere, "The million song dataset," 2011.

[26] H. Schreiber, "Improving genre annotations for the million song dataset.," in *ISMIR*, pp. 241–247, 2015.

[27] F. Gouyon, F. Pachet, O. Delerue, *et al.*, "On the use of zero-crossing rate for an application of classification of percussive sounds," in *Proceedings of the COST G-6 conference on Digital Audio Effects (DAFX-00), Verona, Italy*, vol. 5, Citeseer, 2000.

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

[29] B. Logan, "Mel frequency cepstral coefficients for music modeling," in *In International Symposium on Music Information Retrieval*, Citeseer, 2000.

[30] J. B. Allen and L. R. Rabiner, "A unified approach to short-time fourier analysis and synthesis," *Proceedings of the IEEE*, vol. 65, no. 11, pp. 1558–1564, 1977.

[31] M. O. A. T. U. PANDORA and T. DATA, "Project by, rahul raghavendiran (rahulrar@ usc. edu),"

[32] M. Clynes, *Music, mind, and brain: The neuropsychology of music*. Springer Science & Business Media, 2013.

[33] C. Qin, H. Yang, W. Liu, S. Ding, and Y. Geng, "Music genre trend prediction based on spatial-temporal music influence and euclidean similarity," in *2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC)*, pp. 406–411, IEEE, 2021.

[34] S. Roweis, "Em algorithms for pca and spca," *Advances in neural information processing systems*, pp. 626–632, 1998.

[35] Z. Wang, C. Ye, and W. Wang, "Music trend forecast based on lstm," in *2019 4th International Conference on Computational Intelligence and Applications (ICCIA)*, pp. 30–35, IEEE, 2019.

[36] A. Graves, S. Fernández, and J. Schmidhuber, "Bidirectional lstm networks for improved phoneme classification and recognition," in *International conference on artificial neural networks*, pp. 799–804, Springer, 2005.

[37] N.-Q. Pham, T.-S. Nguyen, J. Niehues, M. Müller, S. Stüker, and A. Waibel, "Very deep self-attention networks for end-to-end speech recognition," *arXiv preprint arXiv:1904.13377*, 2019.

[38] A. Waibel, "Modular construction of time-delay neural networks for speech recognition," *Neural computation*, vol. 1, no. 1, pp. 39–46, 1989.

[39] B. Spillane, E. Gilmartin, C. Saam, L. Clark, B. R. Cowan, and V. Wade, "Identifying topic shift and topic shading in switchboard," *UK Speech 2018 Abstract Book. UK Speech*, vol. 44, 2018.

[40] D. Zhang, T. Li, H. Zhang, and B. Yin, "On data augmentation for extreme multi-label classification," *arXiv preprint arXiv:2009.10778*, 2020.

[41] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[42] K. Bhatia, K. Dahiya, H. Jain, P. Kar, A. Mittal, Y. Prabhu, and M. Varma, "The extreme classification repository: Multi-label datasets and code," 2016.

[43] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang, "Deep learning for extreme multi-label text classification," in *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pp. 115–124, 2017.

[44] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu, "Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification," *Advances in Neural Information Processing Systems*, vol. 32, pp. 5820–5830, 2019.

[45] Y. Zhao and J. Guo, "Musicoder: A universal music-acoustic encoder based on transformer," in *International Conference on Multimedia Modeling*, pp. 417–429, Springer, 2021.

[46] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[47] I. A. P. Santana, F. Pinhelli, J. Donini, L. Catharin, R. B. Mangolin, Y. M. e Gomes da Costa, V. D. Feltrim, and M. A. Domingues.

[48] D. Bogdanov, M. Won, P. Tovstogan, A. Porter, and X. Serra, "The mtg-jamendo dataset for automatic music tagging," 2019.

[49] B. McFee, C. Raffel, D. Liang, D. P. Ellis, M. McVicar, E. Battenberg, and O. Nieto, "librosa: Audio and music signal analysis in python," in *Proceedings of the 14th python in science conference*, vol. 8, pp. 18–25, Citeseer, 2015.

[50] J. LaRue, *Guidelines for style analysis*. No. 12, Harmonie Park Press, 1992.

[51] L. Bluestein, "A linear filtering approach to the computation of discrete fourier transform," *IEEE Transactions on Audio and Electroacoustics*, vol. 18, no. 4, pp. 451–455, 1970.

[52] N. Ahmed, T. Natarajan, and K. R. Rao, "Discrete cosine transform," *IEEE transactions on Computers*, vol. 100, no. 1, pp. 90–93, 1974.

[53] K. Kumar, C. Kim, and R. M. Stern, "Delta-spectral cepstral coefficients for robust speech recognition," in *2011 IEEE International conference on acoustics, speech and signal processing (ICASSP)*, pp. 4784–4787, IEEE, 2011.

[54] M. A. Bartsch and G. H. Wakefield, "To catch a chorus: Using chroma-based representations for audio thumbnailing," in *Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575)*, pp. 15–18, IEEE, 2001.

[55] B. Yegnanarayana, *Artificial neural networks*. PHI Learning Pvt. Ltd., 2009.

[56] H. Ramchoun, M. A. J. Idrissi, Y. Ghanou, and M. Ettaouil, "Multilayer perceptron: Architecture optimization and training.," *Int. J. Interact. Multim. Artif. Intell.*, vol. 4, no. 1, pp. 26–30, 2016.

[57] R. Hecht-Nielsen, "Theory of the backpropagation neural network," in *Neural networks for perception*, pp. 65–93, Elsevier, 1992.

[58] S. Albawi, T. A. Mohammed, and S. Al-Zawi, "Understanding of a convolutional neural network," in *2017 International Conference on Engineering and Technology (ICET)*, pp. 1–6, Ieee, 2017.

[59] D. Hubel and T. Wiesel, "David hubel and torsten wiesel," *Neuron*, vol. 75, no. 2, pp. 182–184, 2012.

[60] K. Fukushima, "Neocognitron: A hierarchical neural network capable of visual pattern recognition," *Neural networks*, vol. 1, no. 2, pp. 119–130, 1988.

[61] L. Dong, S. Xu, and B. Xu, "Speech-transformer: a no-recurrence sequence-to-sequence model for speech recognition," in *2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5884–5888, IEEE, 2018.

[62] W. Zaremba, I. Sutskever, and O. Vinyals, "Recurrent neural network regularization," *arXiv preprint arXiv:1409.2329*, 2014.

[63] M. Sundermeyer, R. Schlüter, and H. Ney, "Lstm neural networks for language modeling," in *Thirteenth annual conference of the international speech communication association*, 2012.

[64] R. Dey and F. M. Salem, "Gate-variants of gated recurrent unit (gru) neural networks," in *2017 IEEE 60th international midwest symposium on circuits and systems (MWSCAS)*, pp. 1597–1600, IEEE, 2017.

[65] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*, pp. 448–456, PMLR, 2015.

[66] X. Dong and J. Shen, "Triplet loss in siamese network for object tracking," in *Proceedings of the European conference on computer vision (ECCV)*, pp. 459–474, 2018.

[67] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: a simple way to prevent neural networks from overfitting," *The journal of machine learning research*, vol. 15, no. 1, pp. 1929–1958, 2014.

[68] A. Gisbrecht, A. Schulz, and B. Hammer, "Parametric nonlinear dimensionality reduction using kernel t-sne," *Neurocomputing*, vol. 147, pp. 71–82, 2015.

[69] Z. Cheng, C. Zou, and J. Dong, "Outlier detection using isolation forest and local outlier factor," in *Proceedings of the conference on research in adaptive and convergent systems*, pp. 161–168, 2019.

[70] C. Reas and B. Fry, "Processing. org: programming for artists and designers," in *ACM SIGGRAPH 2004 Web graphics*, p. 3, 2004.

[71] M. Sperber, J. Niehues, G. Neubig, S. Stüker, and A. Waibel, "Self-attentional acoustic models," *arXiv preprint arXiv:1803.09519*, 2018.

[72] P. Viswanath and T. H. Sarma, "An improvement to k-nearest neighbor classifier," in *2011 IEEE Recent Advances in Intelligent Computational Systems*, pp. 227–231, IEEE, 2011.

[73] T. Liu, A. W. Moore, A. Gray, and C. Cardie, "New algorithms for efficient high-dimensional nonparametric classification.," *Journal of Machine Learning Research*, vol. 7, no. 6, 2006.