UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

LUCRARE DE LICENȚĂ

Enhancing our lives through music: Genre Classification and Generation

propusă de:

Pricop Tudor-Constantin

Sesiunea: **iulie, 2023**

Coordonator științific

Prof. Dr. Iftene Adrian

UNIVERSITATEA “ALEXANDRU IOAN CUZA” DIN IAȘI

FACULTATEA DE INFORMATICĂ

Enhancing our lives through music: Genre Classification and Generation

Pricop Tudor-Constantin

Sesiunea: **iulie, 2023**

Coordonator științific:

Prof. Dr. Iftene Adrian

Avizat,

Îndrumător Lucrare de Licență

Titlul, Numele și prenumele Prof. Dr. Iftene Adrian

Data _____ Semnătura_____

DECLARAȚIE privind originalitatea conținutului lucrării de licență

Subsemnatul Pricop Tudor-Constantin cu domiciliul în județul Bacău, orașul Onești, strada Buciumului nr. 11 sc. A ap. 6, născută la data de 15.05.2001, identificată prin CNP 5010515045353, absolventă a Universității ”Alexandru Ioan Cuza” din Iași, Facultatea de Informatică, specializarea Română, promoția 2020, declar pe propria răspundere, cunoscând consecințele falsului în declarații, în sensul art. 326 din Noul Cod Penal și dispozițiile Legii Educației Naționale nr. 1/2011 art.143 al. 4 și 5 referitoare la plagiat, că lucrarea de licență cu titlul: “Enhancing our lives through music: Genre Classification and Generation” elaborată sub îndrumarea dl. Prof. Dr. Iftene Adrian, pe care urmează să o susțin în fața comisiei este originală, îmi aparține și îmi asum conținutul sau în întregime.

De asemenea, declar că sunt de acord ca lucrarea mea de licență să fie verificată prin orice modalitate legală pentru confirmarea originalității, consimțind inclusiv la introducerea conținutului său într-o bază de date în acest scop.

Am luat la cunoștință despre faptul că este interzisă comercializarea de lucrări științifice în vederea facilitării falsificării de către cumpărător a calității de autor al unei lucrări de licență, de diploma sau de disertație și în acest sens, declar pe proprie răspundere că lucrarea de față nu a fost copiată ci reprezintă rodul cercetării pe care am întreprins-o.

Data azi, _____ Semnătură student, ...

DECLARAȚIE PRIVIND ORIGINALITATEA ȘI RESPECTAREA

DREPTURILOR DE AUTOR

Prin prezenta declar că Lucrarea de licență cu titlul "Enhancing our lives through music: Genre Classification and Generation" este scrisă de mine și nu a mai fost prezentată niciodată la o altă facultate sau instituție de învățământ superior din țară sau din străinătate. De asemenea, declar că toate sursele utilizate, inclusiv cele preluate de pe Internet, sunt indicate în lucrare, cu respectarea regulilor de evitare a plagiatului:

- toate fragmentele de text reproduse exact, chiar și în traducere proprie din altă limbă, sunt scrise între ghilimele și dețin referința precisă a sursei;
- reformularea în cuvinte proprii a textelor scrise de către alți autori deține referința precisă;
- codul sursa, imaginile etc. preluate din proiecte open-source sau alte surse sunt utilizate cu respectarea drepturilor de autor și dețin referințe precise;
- rezumarea ideilor altor autori precizeaza referința precisă la textul original.

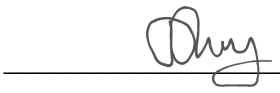
Iași, _____ Absolvent Pricop Tudor-Constantin

DECLARAȚIE DE CONSIMȚĂMÂNT

Prin prezenta declar că sunt de acord ca Lucrarea de licență cu titlul "Enhancing our lives through music: Genre Classification and Generation", codul sursa al programelor și celelalte conținuturi (grafice, multimedia, date de test etc.) care însoțesc această lucrare să fie utilizate în cadrul Facultății de Informatică.

De asemenea, sunt de acord ca Facultatea de Informatică de la Universitatea "Alexandru Ioan Cuza" din Iași, să utilizeze, modifice, reproducă și să distribuie în scopuri necomerciale programele-calculator, format executabil și sursa, realizate de mine în cadrul prezentei lucrări de licență.

Iași, _____ Absolvent Pricop Tudor-Constantin



Abstract:

In this paper, I investigate a comprehensive approach to music genre classification and generation, utilizing a combination of deep neural networks, machine learning algorithms, variational autoencoders (VAEs), long short-term memory (LSTM) networks, and Transformers. My research aims to advance the understanding of music's impact on our lives and develop methodologies to create diverse and engaging musical experiences tailored to individual preferences.

I begin by extracting relevant features from a large and diverse collection of music samples from different genres. These features, encompassing spectral properties, rhythmic patterns, and tonal characteristics, serve as the foundation for my genre classification and generation models. I employ deep neural networks and machine learning algorithms to effectively classify music genres by capturing the distinct characteristics of each genre.

In order to generate music, I explore the potential of VAEs, LSTMs, and Transformers, each offering unique capabilities for handling different aspects of the task. VAEs are employed to learn a continuous latent space representation of the music samples, enabling the generation of novel compositions within a specified genre. LSTMs and Transformers, on the other hand, are used to model the temporal dependencies and intricate patterns inherent in music.

My methodology is evaluated through a series of experiments, with the results compared against conventional machine learning algorithms and other deep learning architectures. While not claiming state-of-the-art performance, my approach demonstrates promising outcomes in both classification and generation tasks, showcasing its potential to enhance music-related applications such as recommendation systems and creative tools for composers.

In conclusion, the suggested framework provides a flexible and thorough answer, opening the door for more research and advancement in music technology.

Contents

1. Introduction....9
1.1. Motivation....9
1.2. Outline of the thesis....10
1.3. Application fields....10
2. Related work....13
2.1. Similar research....13
2.1.1. Music Genre Classification....13
2.1.2. Looking for the Perfect Network....13
2.1.3. JukeBox....14
2.1.4. Groove2Groove....14
2.2. Applications in the field....15
2.2.1. Spotify....15
2.2.2. SoundHound....16
2.2.3. Soundraw....17
3. Technical background....19
3.1. EDA....20
3.1.1. Correlation Heatmap....21
3.1.2. Principal Component Analysis....22
3.1.3. t-Distributed Stochastic Neighbor Embedding (t-SNE)....25
3.1.4. Uniform Manifold Approximation and Projection (UMAP)....25
3.1.3. Feature Engineering....26
3.1.4. Feature selection....26
3.2. Music Genre Recognition....27
3.2.1. Audio signals....30
3.2.1.1. Chroma STFT....32
3.2.1.2. Root-Mean-Square (RMS)....33
3.2.1.3. Spectral centroid....33
3.2.1.4. Spectral bandwidth....34
3.2.1.5. Spectral Rolloff....35
3.2.1.6. Flatness....36
3.2.1.7. Contrast....37
3.2.1.8. Flux....38
3.2.1.9. Zero crossing rate (ZCR)....38
3.2.1.10. Harmonics and Perceptrual/Percussive....39
3.2.1.11. Tempo....40
3.2.1.12. Mel Frequency Cepstral Coefficients (MFCC)....40
3.2.1.13. Mel Spectrogram....42

3.2.2. Machine Learning..... 43
3.2.2.1. KNN..... 43
3.2.2.2. SVM..... 43
3.2.2.3. Random Forest..... 44
3.2.2.4. Logistic Regression..... 44
3.2.2.5. Naïve Bayes..... 44
3.2.2.6. XGBoost..... 45
3.2.3. Deep Learning..... 45
3.2.3.1. CNN..... 45
3.2.3.2. RNN..... 45
3.2.3.3. Transfer Learning..... 46
3.2.3.5. Reinforcement Learning..... 46
3.3. Music Generation..... 47
3.3.1. Music notation..... 47
3.3.2. LSTM..... 48
3.3.3. Transformer..... 48
3.3.4. VAE..... 49
3.3.5. GAN..... 50
4. Proposed framework..... 52
4.1. Music Genre Recognition..... 52
4.1.1. Datasets..... 52
4.1.1.1. GTZAN..... 53
4.1.1.2. FMA..... 56
4.1.1.3. MSD..... 60
4.1.1.4. Spotify..... 61
4.1.2. Machine Learning..... 65
4.1.3. Deep Learning..... 69
4.2. Music Generation..... 73
4.2.1. Dataset..... 73
4.2.2. Model..... 75
5. Evaluation..... 78
5.1. Music Genre Recognition..... 78
5.2. Music Generation..... 83
6. Conclusion and future work..... 86
7. Bibliography..... 87

1. Introduction

1.1. Motivation

The transformative power of music has been a cornerstone of human culture and expression throughout history. As diverse as our societies, the vast array of music genres reflects our rich cultural heritage and individual creativity. With the rapid advancements in technology, new paradigms in music analysis and synthesis are emerging, further enhancing our understanding of this universal language.

The motivation for this research paper stems from a desire to harness the potential of cutting-edge learning techniques, such as Machine Learning, Deep Neural Networks (DNNs), Long Short-Term Memory (LSTM) networks, Variational Autoencoders (VAEs), and Transformers, to address two primary challenges in the field of music informatics: genre classification and music generation.

The accurate classification of music genres is a crucial task that aids in music discovery, recommendation, and tagging. However, the subjective nature of genre definitions and the complexity of musical elements make this a challenging problem. By employing deep neural networks and machine learning techniques, this research aims to develop robust and sophisticated models that can effectively capture the intricate patterns and features of various music genres, leading to improved classification performance.

The creative process of composing music has long been considered the exclusive domain of human intellect. However, recent advancements in artificial intelligence and machine learning have demonstrated the potential for machines to generate music with remarkable coherence and originality. This research explores the application of LSTM, VAE, and Transformer models to the task of music generation, with the goal of developing systems capable of producing high-quality compositions across different genres, while respecting the unique characteristics and structures inherent to each.

By combining expertise in music theory, machine learning and artificial intelligence, this research paper aims to contribute to the ongoing efforts to revolutionize the field of music informatics. It is hoped that the findings will not only serve as a foundation for future

research endeavors but also pave the way for innovative applications that enrich our understanding and appreciation of the world of music.

"*Music gives a soul to the universe, wings to the mind, flight to the imagination, and life to everything*." – Plato

1.2. Outline of the thesis

This thesis is organized as follows:

After providing a brief introduction to the goals and motivation of this research paper, Chapter 2 will present various related experiments that demonstrate diverse approaches to music classification and generation. These studies include unique datasets, models, metrics, and results, giving a well-rounded view of the different methods used in this field.

Moving forward, Chapter 3 offers a comprehensive exploration of music theory, delving into the various theoretical elements, structures and algorithms employed throughout this study. Chapter 4 thoroughly outlines the data preparation process for information extraction, the selected models, and the overall framework. Chapter 5 then explores the interpretation of outcomes and showcases a selection of pertinent metrics that effectively represent the issues under discussion.

Music tagging, like identifying genres or emotions, is a common practice in the music industry. However, there hasn't been a ton of work done in generating music in a specific style. Despite this, the work that has been done is pretty varied, because it can involve different ways of representing music and address different concepts. Pursuing the directions offered in this study, numerous intriguing concepts merit further exploration, including transferring styles between songs while preserving the original content or making remixes of melodies while controlling specific attributes like chromaticism, groove, or instrumentation, aspects discussed in Chapter 6.

1.3. Application fields

The swift advancement of machine learning and artificial intelligence methodologies has accelerated the growth of music information retrieval and music creation. These technologies

offer numerous applications, which possess the potential to radically transform our interaction, consumption and production of music.

Content-based music retrieval and recommendation systems constitute a principal application domain of music genre identification. By classifying music tracks into well-defined genres, these systems can deliver customized music suggestions based on users' predilections and listening behavior. Additionally, refined genre identification can aid in enhancing playlist creation, empowering users to uncover new music within their favored genres or based on their listening mood.

Music genre identification is integral to Music Information Retrieval (MIR), an interdisciplinary area concerned with extracting and organizing information from music. Advanced genre identification methods streamline the examination and cataloging of extensive music databases, facilitating the detection of patterns, trends, and connections between various musical pieces and enriching our comprehension of music as a cultural phenomenon.

On the other hand, music creation techniques have given rise to interactive music composition tools, allowing users to produce original music with minimal expertise. Employing AI-generated music, these tools can support users in composing melodies, harmonies, and rhythms, enabling them to concentrate on their artistic vision. Furthermore, such tools can serve educational purposes, assisting budding musicians in honing their skills and grasping music theory.

Automatic music transcription systems can leverage improvements in music genre identification and music creation. By incorporating genre-specific knowledge, these systems can enhance their precision in transposing audio recordings into symbolic notation, enabling musicians to analyze, modify, and perform the transcribed music more effectively.

Music creation techniques have been applied to video game and film scoring. AI-generated music can adapt to the dynamic nature of video games, offering immersive and responsive auditory experiences for players. Similarly, in the film industry, AI-generated music can function as an economical alternative to conventional scoring methods, addressing the unique demands of individual scenes and narratives.

Music therapy has long been acknowledged for its potential advantages in fostering mental and emotional well-being. AI-generated music can be customized to address the specific

requirements of individuals, delivering personalized therapeutic experiences. Additionally, music creation techniques can be employed in the development of assistive technologies for people with disabilities, enabling them to interact with music in innovative and accessible ways.

2. Related work

2.1. Similar research

2.1.1. Music Genre Classification

Classification of music genres has been an extensively studied topic with numerous techniques applied. One of the many papers that approaches this theme is [1]. The authors worked with the GTZAN dataset, using both Machine Learning and Deep Learning techniques. The features and melspectrograms used were computed by the team, and as a dimensional reduction technique they used Principal Component Analysis. The models used for experimentation: K-Nearest Neighbors, Support Vector Machine, Feed Forward Neural Network and Convolutional Neural Network. Following data processing methods and training the models for a large number of epochs, the results showed a maximum of 60% accuracy for ML algorithms and a dramatic jump of performance with melspectrograms, at approximately 82%. Regarding the visible overfitting of each model, the authors introduced batch normalization, dropout layers and L2 regularization. Looking at the confusion matrices, the genres that were hardest to classify were Rock (due to lack of visible distinctive traits in spectrograms) and Jazz (piano-heavy tracks may have been too close to classical music).

2.1.2. Looking for the Perfect Network

A more recent work, [27], the authors use another dataset for the task of genre classification, the small subset of the Free Music Archive, which contains 8,000 tracks categorized into 8 top-level genres. They experiment with both features and melspectrograms, making usage of Convolutional Neural Networks, Convolutional Recurrent Neural Networks with 1D and 2D convolutions and Recurrent Neural Networks. To combine the advantages of each model, they also use ensembles. As they delve into their work, they also emphasize that the task is very challenging, even for music experts, especially when some genres are vaguely defined and interpolated with other genres (Pop, Experimental). It is also mentioned how ambiguous the conclusions can be between different research papers that study the same task, as the databases, the genres and even the measurements vary. However, they compare their results with other algorithms used in several papers, results that resemble the ones that will be presented in this paper's work.

No.	Model	Accuracy	No.	Model	Accuracy	Recall	F1 score
1	K-Nearest Neighbors [42]	36.4	12	MoER [41]	55.9	–	–
2	Logistic regression [42]	42.3	13	FCN [39]	63.9	43.0	40.3
3	Multilayer perceptron [42]	44.9	14	TimbreCNN [39]	61.7	36.4	35.0
4	Support vector machine [42]	46.4	15	End-to-end [39]	61.4	38.4	34.5
5	Original spectrogram [41]	49.4	16	CRNN [39]	63.4	40.7	40.2
6	Harmonic spectrogram [41]	43.4	17	CRNN-TF [39]	64.7	43.5	42.3
7	Percussive spectrogram [41]	50.9	18	CRNN [32]	53.5	–	–
8	Modulation spectrogram [41]	55.6	19	CNN-RNN [32]	56.4	–	–
9	MFCC [41]	47.1	20	CNN TL [16]	51.5	–	–
10	MoEB [41]	54.1	21	CNN TL [28]	56.8	–	–
11	MoEC [41]	55.6	22	C-RNN [42]	65.2	–	–
23	**Ensemble 1 - vote**	**56.4**	**23**	**Ensemble 1 - vote**	**56.4**	**54.8**	**54.9**

Figure 2.1.2. Comparison between algorithms that reach state-of-the-art results - FMA

2.1.3. JukeBox

The paper [2] describes a generative model called JukeBox, that can create music and singing in raw audio domains. It uses a multi-scale VQ-VAE (Vector Quantized Variational AutoEncoder) to compress raw audio to discrete codes and models those using autoregressive Transformers. This model can be conditioned on artist and genre to steer the musical style. It was trained on a large dataset of 1 million songs, paired with the corresponding metadata. After a slow training process due to the complexity of the model, the results shown are quite remarkable. It generates coherent music pieces with harmony, rhythm and even singing in multiple genres. Even if the model has its limitations in controlling the high-level attributes of the generated song, its capabilities in mimicking artists style and generating lyrics are still impressive.

2.1.4. Groove2Groove

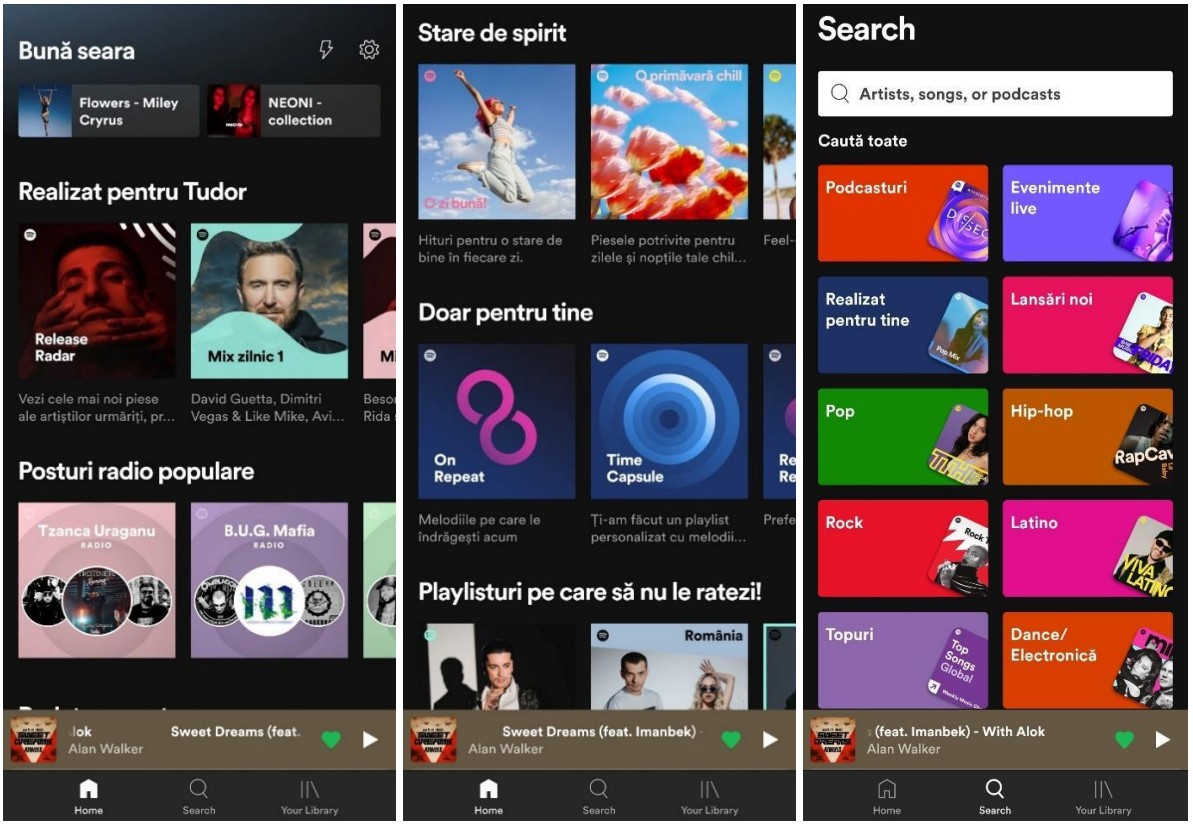
Another interesting approach regarding music generation in a specific style is presented in [3]. The paper references one-shot transfer, which involves taking a piece of music in one style (jazz) and transforming it into another style (rock), using only a single example of the target style. The model consists of a style encoder, which takes as input a single example of the target and encodes it, and a decoder which takes as input a MIDI file and the target style, and outputs a new MIDI file containing the original content in the target style. The model is evaluated on a variety of musical styles (not only broad genres), and shows that it is able to

perform the transfer with high fidelity. There is also a user study conducted by the team which shows that participants prefer the output of this model over other state-of-the-art methods. This approach has the potential to be very useful in a variety of musical applications like remixing music.

2.2. Applications in the field

2.2.1. Spotify

Spotify[1] is a popular digital music, podcast and video streaming service that gives access to millions of songs and other content from creators all over the world. Artificial Intelligence (AI) plays an integral role in Spotify's functionality, analyzing the acoustic, cultural and personal inputs for every user, in order to create personalized recommendations. There are even some AI-driven systems that predict upcoming hits based on users' behavior patterns. Moreover, Spotify is developing several models to create music and provide songwriting assistance, showcasing AI's potential in the creative aspects of the music industry.



[1] https://open.spotify.com/

Figure 2.2.1.1. Spotify

2.2.2. SoundHound

SoundHound[2] is a versatile and popular music application that serves as a platform that identifies songs, displays real-time lyrics and even offers a voice-controlled AI. The functioning of SoundHound's song identification system is based on a methodology known as audio fingerprinting. A user plays a song near the device running SoundHound, the app listens to a segment of a song, transforms it into a unique numerical identifier (fingerprint) and matches it against a vast database of music. The process is very fast and can identify songs within a matter of seconds, even in noisy environments. Furthermore, it has a unique ability to identify a fingerprint even on a user's humming, singing or whistling, without the need for the original recording or lyrics.

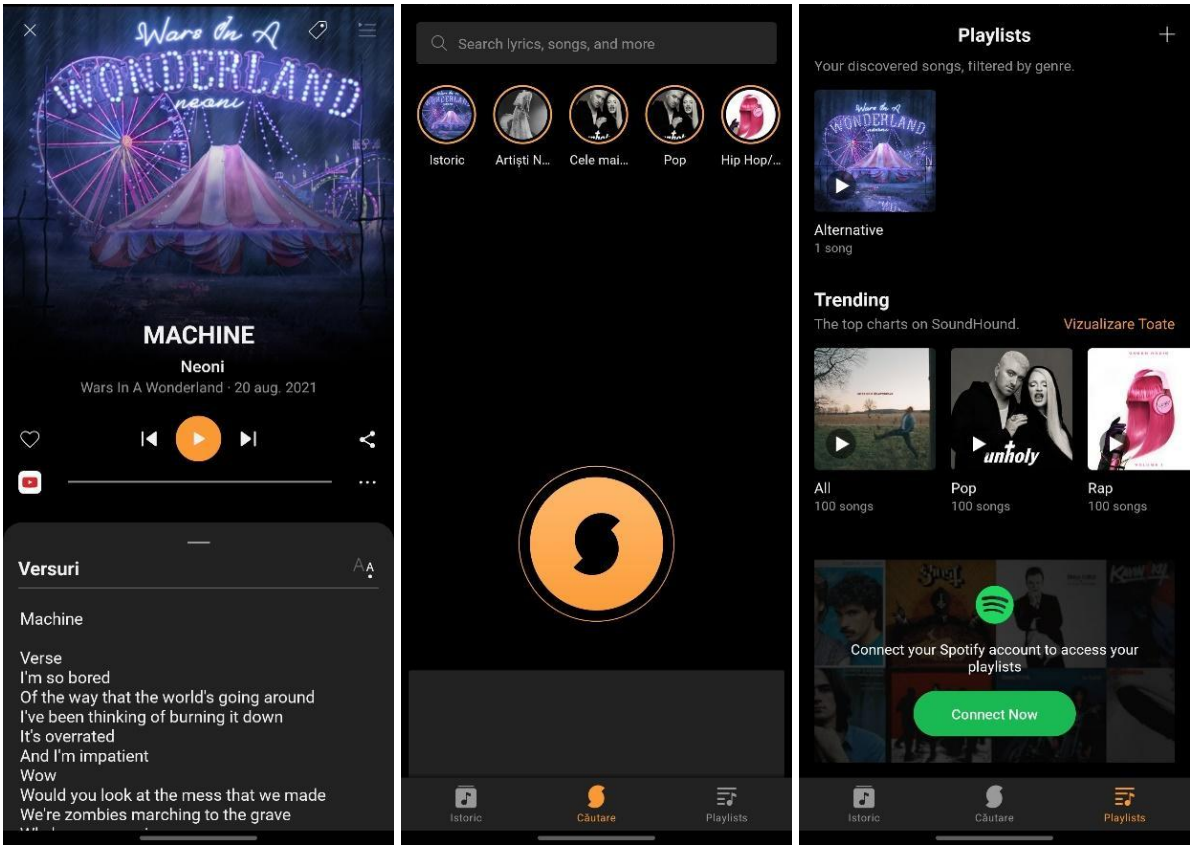
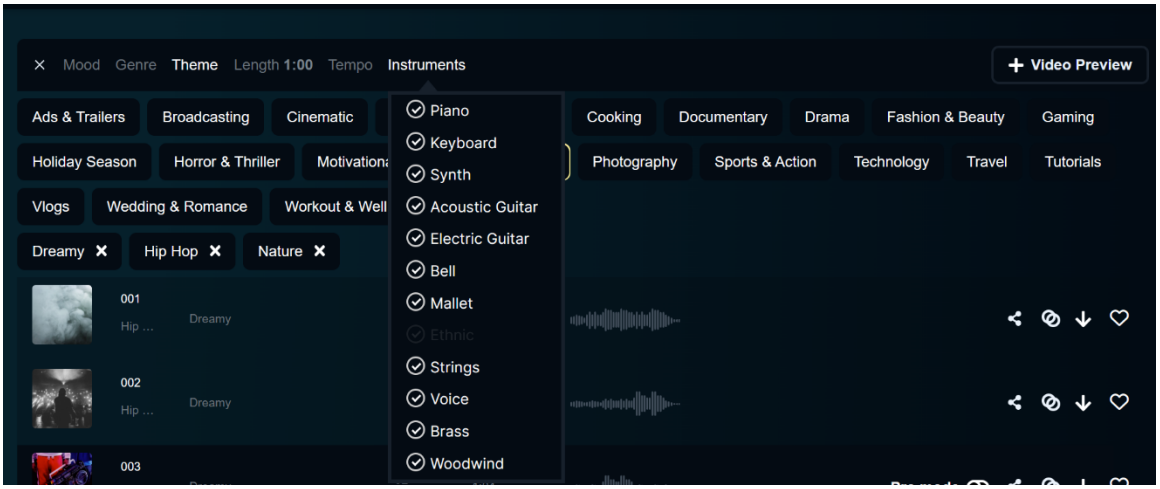


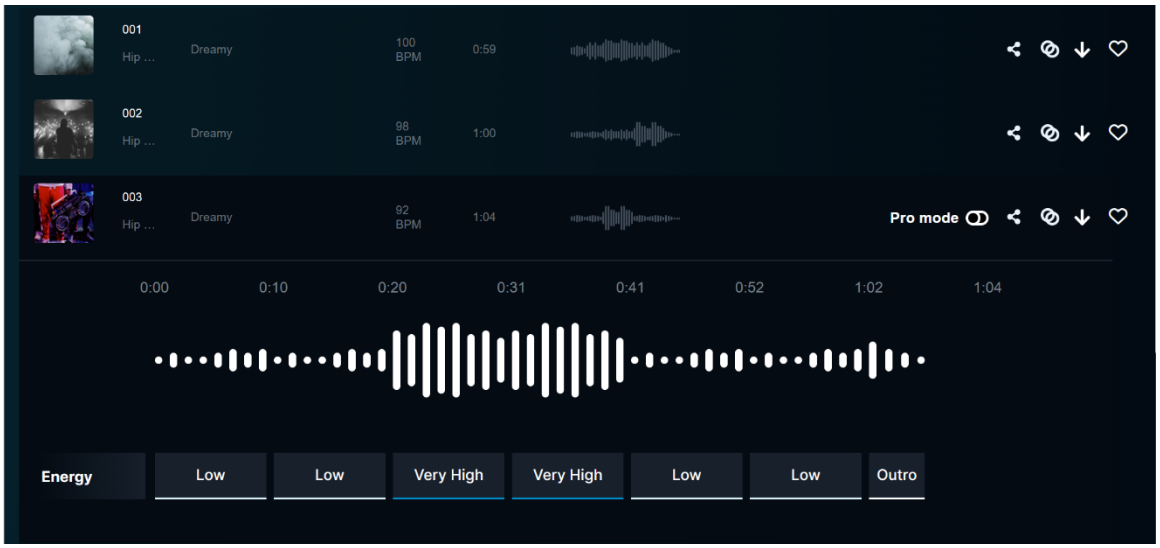
Figure 2.2.2.1. SoundHound

[2] https://www.soundhound.com/soundhound

2.2.3. Soundraw

Soundraw[3] is an AI-driven platform designed to democratize the music production process. The app enables users to generate original music compositions based on various attributes, accessible to both novices and musicians alike. One of the most impressive features is the ability to create music according to a genre. The result can also be fine-tuned to reflect a desired mood or a theme, leading to highly personalized pieces of music tailored to users' preferences. Other powerful features include tempo and length altering and also audio editing tools like specifying notes, changing instruments or adjusting the mix of the track.





[3] https://soundraw.io/

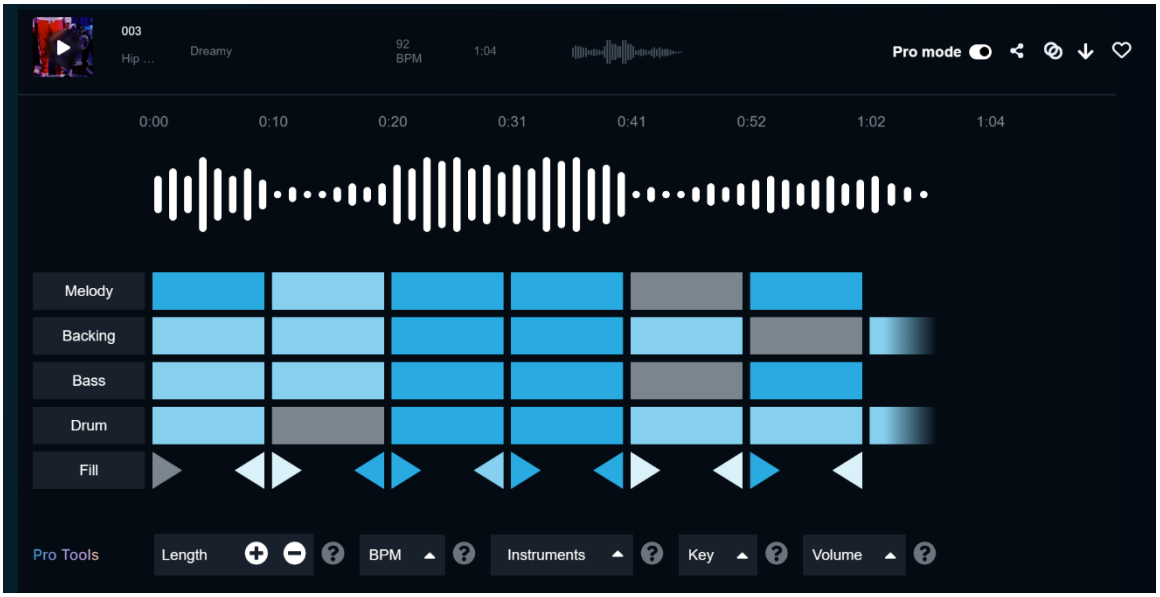


Figure 2.2.3.1. Soundraw

3. Technical background

Music Information Retrieval (MIR), as described in [28], is a field that involves the study of processing audio signals in order to access relevant information from music. It combines elements of music theory, physics, psychology, signal processing, linguistics, mathematics and computer science. A few applications to this field include:

1. **Understanding Music Data**: Music can be represented in various formats, such as audio signals, symbolic notations, or lyrics. MIR aims to extract meaningful information from these different data representations.

2. **Feature Extraction**: This process involves identifying key aspects of music data, such as pitch, tempo, rhythm, melody, harmony, and timbre. These features can help in the analysis and categorization of music.

3. **Music Analysis**: MIR can be used to analyze music on various levels. This includes low-level analysis (determining the beat or pitch), mid-level analysis (identifying beat tracking or chord recognition) and high-level analysis (classifying the genre or mood of a piece).

4. **Search and Recommendation**: techniques in the field are often used in music recommendation systems, such as those found in Spotify or Pandora. These systems analyze the features of music to provide personalized recommendations to users.

5. **Music Transcription**: it also encompasses automatic music transcription, which is the process of converting a music audio signal into a symbolic representation such as sheet music or MIDI.

6. **Music Classification and Clustering**: these techniques can also be used to classify music into different genres, moods, or even identifying similar songs which is extremely useful in music libraries and streaming platforms to organize music and create playlists.

7. **Music Generation**: MIR is also used in music generation, where algorithms learn the patterns in music data to create new compositions.

Research in Music Information Retrieval can lead to advancements in various applications such as music search engines, digital libraries, music composition software, interactive music

systems, and even music education and therapy. However, it also poses great challenges due to the complexity and subjectivity of music perception.

3.1. EDA

According to [4], exploratory Data Analysis (EDA) is a strategic approach employed by data scientists to analyze and investigate datasets, thereby summarizing their predominant characteristics, frequently via visualization techniques. It helps in identifying the most efficient way to manipulate data sources to derive the required information, discovery of patterns, detection of anomalies, hypothesis testing and verification of assumptions.

Primarily, EDA is utilized to glean insights that data can offer beyond the conventional modeling or hypothesis testing activities. It bestows a more profound understanding of the dataset's variables and their interrelationships. Moreover, it can assist in verifying whether the statistical techniques considered for data analysis are appropriate.

The core objective is to enable an in-depth examination of data prior to making any assumptions. It facilitates the identification of apparent errors, deeper comprehension of data patterns, detection of outliers or anomalous occurrences and discovery of interesting relationships among variables.

In the realm of data science, exploratory analysis serves to validate the results, ensuring their applicability to the intended outcomes and objectives. It further confirms the relevance of the questions being asked. It also provides answers regarding standard deviations, categorical variables, and confidence intervals. Upon completion of EDA and drawing of insights, its attributes can be employed for more advanced data analysis or modeling, including machine learning.

Statistical functions and techniques that can be performed using EDA include:

- Clustering and dimension reduction techniques, which facilitate the generation of visual representations for high-dimensional data containing many variables.
- Univariate visualization of the fields in the original dataset, with descriptive statistics.
- Bivariate visualizations and descriptive statistics that enable the evaluation of the relationship between each dataset variable and the target variable under examination.

- Multivariate visualizations, for mapping and understanding the interactions among different fields in the data.
- K-means Clustering, an unsupervised learning method, where data points are assigned into K groups (clusters), based on their proximity to each group's centroid. The data points nearest to a particular centroid will fall under the same category. This method is commonly used in market segmentation, pattern recognition and image compression.
- Predictive models, like linear regression, employ statistics and data to predict possible outcomes.

3.1.1. Correlation Heatmap

A correlation heatmap is a type of data visualization tool that represents the correlations between multiple variables in a dataset and it is thoroughly described in [29]. It's a grid of cells, where each cell represents a correlation coefficient between two variables. It's especially useful for feature selection in machine learning, identifying multicollinearity in regression analysis and for exploratory data analysis.

Correlation is a statistical measure that describes the degree of relationship between two variables. The correlation coefficient shows the strength of the correlation and its magnitude can range from -1 to 1. A correlation of -1 indicates a perfect negative correlation (as one variable increases, the other decreases), a correlation of 1 indicates a perfect positive correlation (as one variable increases, the other increases as well) and a correlation of 0 indicates no linear relationship between the variables. However, a correlation between two variables does not imply causation, it only indicates a statistical relationship between the variables. Further investigation is needed to determine if one variable is causing the other.

A **heatmap** is a graphical representation of data where values are depicted by color. It gives a visual representation of data using colors, where higher intensity ("hotter" / brighter) colors represent higher values, and lower intensity ("cooler" / darker) colors represent lower values. Heatmaps are used when there are too many data points to plot individually. The color of the cell is indicative of the correlation coefficient: positive correlations are displayed in one color gradient (increasingly intense shades of red) and negative correlations in another color gradient (increasingly intense shades of blue).

As stated in the article [29], the correlation heatmap is useful for:

- **Visual representation**: easier to understand and interpret the relationships between variables, especially when dealing with a large number of them.
- **Identifying relationships**: highlights the variables that are highly correlated with each other and those that have little or no correlation.
- **Feature selection**: by identifying the variables that are highly correlated with the target variable, one can choose the most important variables for the model, reducing the risk of overfitting and improving model performance.
- **Data cleaning**: identify and remove redundant or unnecessary variables, which can improve the accuracy of the analysis and the interpretability of the results.
- **Hypothesis testing**: generate hypotheses about the relationships between variables, hypotheses that can then be tested further through statistical analysis.

3.1.2. Principal Component Analysis

Principal Component Analysis (PCA), explained in [30], is a dimensionality reduction method that is often used **to reduce the dimensionality** of large data sets, by transforming a large set of variables into a smaller one that **still contains most of the information** in the large set.

Reducing the number of variables of a data set naturally comes at the expense of accuracy, but the trick in dimensionality reduction is to trade a little accuracy for simplicity. This is because smaller data sets are easier to explore and visualize, and facilitate quicker and more efficient analysis of data points for machine learning algorithms, devoid of superfluous variables to process.

PCA can be broken down into five steps:

1. **Standardization**: The purpose of this step is to standardize the range of the continuous initial variables, ensuring each contributes equally to the analysis. It is essential to carry out standardization before performing PCA, as PCA is particularly sensitive to the variances of the initial variables. That is, if there are substantial discrepancies in the ranges of the initial variables, those with broader ranges will dominate over those with narrower ones, leading to biased results. Mathematically, for each value of each variable:

$$z = \frac{value - mean}{standard\ deviation}$$

2. **Covariance Matrix Computation**: Sometimes variables are highly correlated in such a way that they contain redundant information. It is the sign of the covariance that really matters: when positive, the two variables increase/decrease together (correlated); when negative, as one increases, the other decreases (inversely correlated).

$$Cov(x,x)\ Cov(x,y)\ Cov(x,z)\ Cov(y,x)\ Cov(y,y)\ Cov(y,z)\ Cov(z,x)\ Cov(z,y)\ Cov(z,z)$$

3. **Eigenvectors and eigenvalues**: Principal components are new variables constructed as linear combinations, or mixtures, of the initial variables. These mixtures are arranged so that the new variables, known as principal components, are uncorrelated and most of the information found in the initial variables is concentrated or compressed into the first few components. From a geometric perspective, principal components represent the data directions that account for the maximum variance, which are essentially the lines that encapsulate most of the information of the data. The larger the variance along a line, the greater the data point dispersion around it, thus the more information it possesses. To put all this simply, consider principal components as new axes that offer the optimal perspective to observe and evaluate the data, thereby making the differences between observations more discernible. Underpinning PCA are eigenvectors and eigenvalues. The eigenvectors of the Covariance matrix essentially outline the directions of the axes where the most variance, or the most information, is located, and those are called Principal Components. Eigenvalues are coefficients attached to the eigenvectors, indicating the amount of variance each component carries. By ranking the eigenvectors in descending order of their eigenvalues, it results in the principal components in order of their significance.

$$Matrix * v = \lambda * v; \qquad \lambda - eigenvalue,\ v - eigenvector$$

4. **Feature vector**: The feature vector is essentially a matrix whose columns are the eigenvectors of the components we choose to retain. This serves as the initial step

towards reducing dimensionality, as if we opt to keep only ‘p’ eigenvectors, or components, out of ‘n’, the resulting dataset will be confined to ‘p’ dimensions.

5. **Recast data along PCA axes**: Up to this point, the input dataset remains unmodified with respect to the original axes (in terms of the initial variables). The goal of this final step is to make use of the feature vector, constructed from the covariance matrix’s eigenvectors, to reposition the data from the original axes to those represented by the principal components. This can be accomplished by multiplying the transpose of the original dataset by the transpose of the feature vector.

$$FinalDataset = FeatureVector^{T} * StandardizedOriginalDatset^{T}$$

PCA is useful for:

- **Dimensionality reduction**: reduce the dimensionality of a data set, which can be especially useful when dealing with high-dimensional data. This can improve the performance of machine learning models, increase the interpretability of the results, and reduce the risk of overfitting.
- **Data visualization**: visualize complex data sets by projecting the data onto the principal components. This can help to identify patterns and relationships in the data that may not have been apparent in the original variables.
- **Data compression**: compress the data by retaining only the most important components and discarding the others. This can reduce the storage and computation requirements for large data sets.
- **Noise reduction**: remove noise from the data by retaining only the components that explain the most variance. This can improve the accuracy and stability of the results.
- **Feature extraction**: extract new features from the data, which can be used as input variables in machine learning models. This can improve the performance of the models by reducing the dimensionality of the data and eliminating redundant or irrelevant variables.

3.1.3. t-Distributed Stochastic Neighbor Embedding (t-SNE)

t-SNE is a ML algorithm primarily used for the visualization of high-dimensional data, carefully explained in [31]. This dimensionality reduction technique is an invaluable asset for the exploratory phase of data analysis and for presenting complex data in a more intuitive way, through scatter plots. It operates by preserving the similarity between data points as it maps from the high to the low dimensional space. This similarity is represented as a probability distribution with more similar objects having a higher probability of being picked. The algorithm then finds a low-dimensional embedding that best matches this distribution. The process consists of 2 main steps:

1. It computes a probability distribution over pairs of high-dimensional objects in such a way that similar objects have a higher probability of being picked.
2. It computes a similar probability distribution over the points in the low-dimensional map and it minimizes the Kullback-Leibler (KL) divergence between the two distributions with respect to the locations of the points in the map.

t-SNE has one peak ability to preserve local structure in the data, making it particularly useful to visualize datasets where local relationships are more important than global relationships. Due to this strength, it is often used in image recognition tasks, where each image can be thought of as a point in a high dimension space and similar images are expected to be located nearby in this space. Despite its strengths, the algorithm is sensitive to hyperparameters, which can have a large impact on the resulting visualizations.

3.1.4. Uniform Manifold Approximation and Projection (UMAP)

Uniform Manifold Approximation and Projection is a novel algorithm, described in [32], that is used for dimensionality reduction and visualization of data. It is regarded for its flexibility, scalability and ability to preserve both local and global data structures. UMAP begins by viewing the data as a high dimensional geometric shape and attempts to understand the shape of this data. It also translates high dimensional distances into a probabilistic space but unlike t-SNE, it uses Riemann geometry, which is concerned with distances on curved surfaces, plus it employs a cross-entropy mechanism to build the low dimensional representation. A key strength is its computational efficiency and the fact that it is a general-purpose technique, used in many ML algorithms.

3.1.3. Feature Engineering

Feature engineering is a critical step in building accurate and effective models. One key aspect of feature engineering is scaling, normalization and standardization, which involves transforming the data to make it more suitable for modeling. These techniques can help to improve performance, prevent the outliers from having a disproportionate effect on the results, improve the convergence of optimization algorithms and ensure that the data is on the same scale.

Feature scaling is a technique that involves transforming the values of features or variables in a dataset to a similar scale. This is done to ensure that all features contribute equally to the model and to prevent features with larger values from dominating the model. Feature scaling is essential when working with datasets where the features have different ranges, units of measurement or orders of magnitudes. It is critical to scale data for some machine learning algorithms, especially those that use distance measures (KNN) or those that use gradient descent to optimize their cost functions (logistic regression or deep learning algorithms).

Normalization is a scaling technique in which values are shifted and rescaled so that they end up ranging between 0 and 1; it is also known as Min-Max scaling. It is sensitive to outliers, as an extreme value can shift the entire range. The mathematical formula used for scaling uses the minimum and maximum value of each feature column:

$$X' = \frac{X - X_{min}}{X_{max} - X_{min}}$$

Standardization is also a scaling technique where the values are centered around the mean with a unit standard deviation. In this case, the values are not restricted to a particular range, which can be useful for algorithms that do not require a specific scale. Furthermore, it is not as sensitive to outliers as normalization.

$$X' = \frac{X - \mu}{\sigma}; \qquad \mu - mean\ of\ features,\ \sigma - standard\ deviation\ of\ features$$

3.1.4. Feature selection

From the machine learning perspective, not all features carry equal significance or quality, and the presence of unnecessary or repetitive features could result in inaccurate outcomes. As

evidenced by experiments from [5], utilizing many features can enhance performance to some extent, but ultimately leads to performance deterioration. It is incorrect to assume that a highly discriminative set of features, when combined, will yield superior discriminatory power. For instance, if there are 'n' features in a set, the number of possible combinations is:

$$n_{combinations} = \sum_{k=1}^{n} (n\ k)$$

This can easily become impossible to compute, that is why some feature selection algorithm must be applied. There are 3 primary categories of feature selection methods: filter, wrapper and embedded. Filter methods use statistical measures to rank features based on their relevance, they are fast but do not account for potential features interactions. Wrapper methods are computationally expensive methods that treat the selection of a set of features as a search problem. Finally, embedded methods are a blend of filter and wrapper methods, selecting features in the process of model training and are specific to certain machine learning algorithms.

3.2. Music Genre Recognition

Music holds a significant place in human existence, a prominence that has only increased in the era of digital technology. There's never been such an extensive array of music created and engaged with daily as there is today. Initially, compact audio formats like MP3 provided near CD quality music, but now numerous streaming platforms have further spurred the significant expansion of digital music collections.

Traditionally, organizing music collections has relied on cataloging metadata like the artist's name, album title, and song name. However, with the exponential surge of content, this traditional method might not be adequate any longer. The way we categorize and access musical information needs to evolve in response to the ever-growing demand for efficient and effortless information access.

Music, with its intricate acoustic and temporal structure, is abundant in content and expressiveness. When someone interacts with music, whether as a composer, performer, or listener, a broad array of cognitive processes come into play. These include representational processes like understanding rhythm, meter, melody, harmony, style, and form, and

evaluative processes involving preferences, aesthetic experience, mood, and emotions. The term "evaluative" is used because these processes often involve subjective and differing responses. Both the representational and evaluative aspects of music listening could be exploited to improve music retrieval.

According to a study by Last.fm [6], tags related to emotions are the third most commonly applied to music pieces by online users, with genre and geographical area being the first and second most frequently assigned, respectively.

A survey conducted in 2004, [7], showed that a significant number of the participants, about 62.7%, identified the style/genre of a musical piece as an important criterion in music seeking and organization.

SEARCH / BROWSE BY	**POSITIVE RATE**
Singer / performer	96.2%
Title of work(s)	91.6%
Some words of the lyrics	74.0%
Music style/genre	**62.7%**
Recommendations	62.2%
Similar artist(s)	59.3%
Similar music	54.2%
Associated usage	41.9%
Singing	34.8%
Theme (main subject)	33.4%
Popularity	31.0%
Mood / emotional state	28.2%
Time period	23.8%
Occasions to use	23.6%
Instrument(s)	20.8%
Place / event where heard	20.7%
storyline of music	17.9%
Tempo	14.2%
Record label	11.7%
Publisher	6.0%

Table 3.2.1: Responses of 427 subjects to the question “When you search for music or music information, how likely are you to use the following search/browse options?”

In recent years, there has been a lot of activity in the field of audio research, especially with the use of advanced computer techniques like Machine Learning and Deep Learning. With the rise of "Big Data," there is now a lot of digital music available online, which has led to the creation of many online music databases. To make it easier for people to find the music they want, it is important to group music by genre. This is where genre classification comes in, and it has many practical applications, like helping to organize music collections and making it easier to find songs in search engines and music databases. Automated genre classification can be especially useful for companies like Spotify and iTunes, who add thousands of new songs every month.

Music is something that everyone around the world can understand and enjoy. It is a way for us to share our feelings and experiences, and it is also different everywhere you go. This is because there are so many types of music, called 'genres’ and they all sound unique. Think about how different blues music sounds compared to hip-hop, or how classical music is so different from electronic music. These genres have been created over many years and each one tells its own story about people's lives and history. This section will show you a map of all these different music genres. It is like a big adventure through the world of music, showing you just how many types there are and how beautiful each one is in its own way.

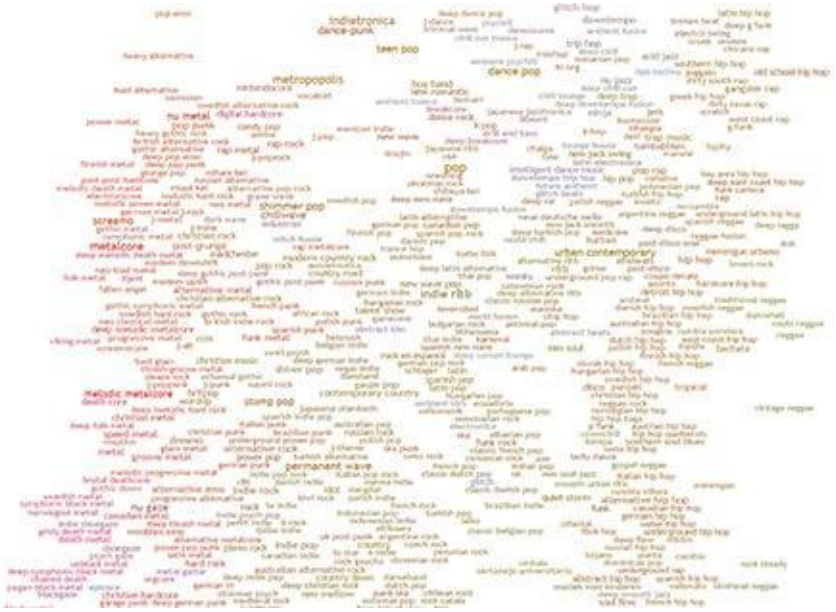


Figure 3.2.1. Every music genre from [8]

3.2.1. Audio signals

Sound is a type of energy vibrating through a medium (such as air), if there is no medium (in space) there is no sound; this energy, within a specific range of frequencies (20Hz – 20kHz), is interpreted by the human ear as sound. It is made up of three basic elements: **frequency**, how fast the vibrations are occurring, **intensity**, how loud the sound is, and **timbre**, the sound's quality. Thus, in a physical sense, an audio signal is a representation of sound as a function of time, typically as a variation of air pressure. It can be converted to an electrical signal by a microphone and then digitized using an analog-to-digital converter.

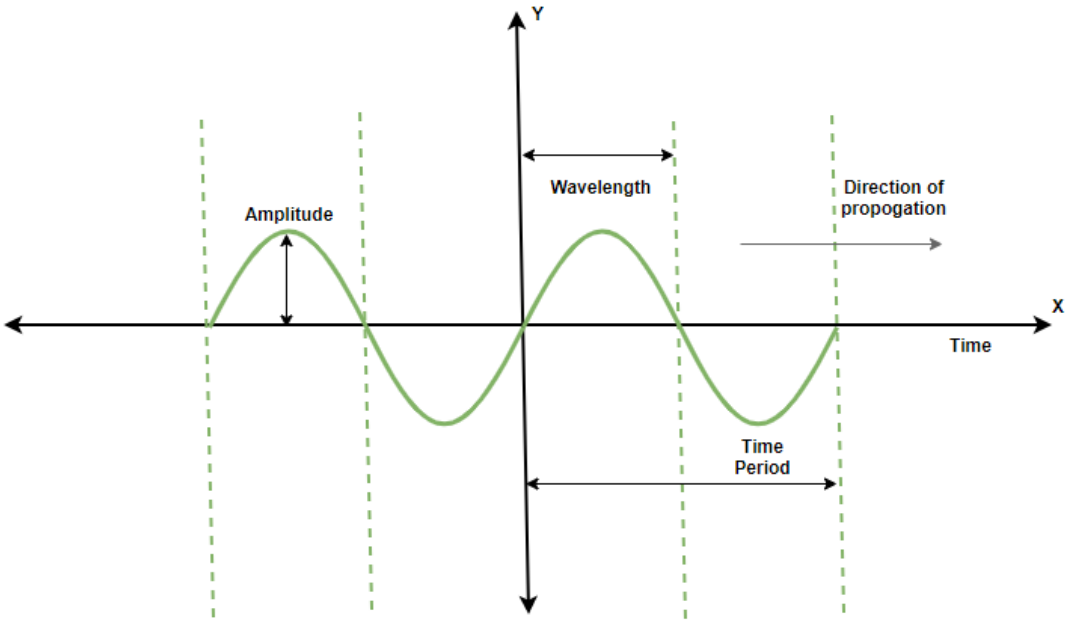


Figure 3.2.1.1. Sound waves in 2D axes

In a digital sense, the audio signal is a sequence of numbers that represents the amplitude of the signal at each point in time. For stereo audio, there are typically two such sequences (left / right channel), while mono audio consists of only a single sequence. Digital audio signals can be characterized by their sample rate (the number of samples per second) and bit depth (the number of bits used to represent each sample). For example, an audio of CD-quality has a sample rate of 44.1kHz and a bit depth of 16 bits.

There are several audio formats in which the sound can be stored and manipulated: .wav (lossless, large files), .mp3(lossy, compressed files), .flac(lossless, compressed files). In order

to manipulate an audio file in Python one can use numerous libraries: Librosa, PyAudio or built-in modules.

In order to play audio files inside notebooks, IPython module offers a pretty and simple configuration:

```python
import IPython.display as ipd

audio_test_path = './Data/genres_original/rock/rock.00000.wav'

# Play audio input
ipd.Audio(audio_test_path)
```

Python

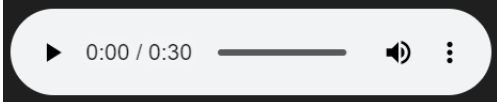


Figure 3.2.1.2. Playable in-notebook audio file

```
Sample rate: 22050Hz
Total samples: 661794
Audio length: 30.013333333333332

<librosa.display.AdaptiveWaveplot at 0x1961c3afe20>
```

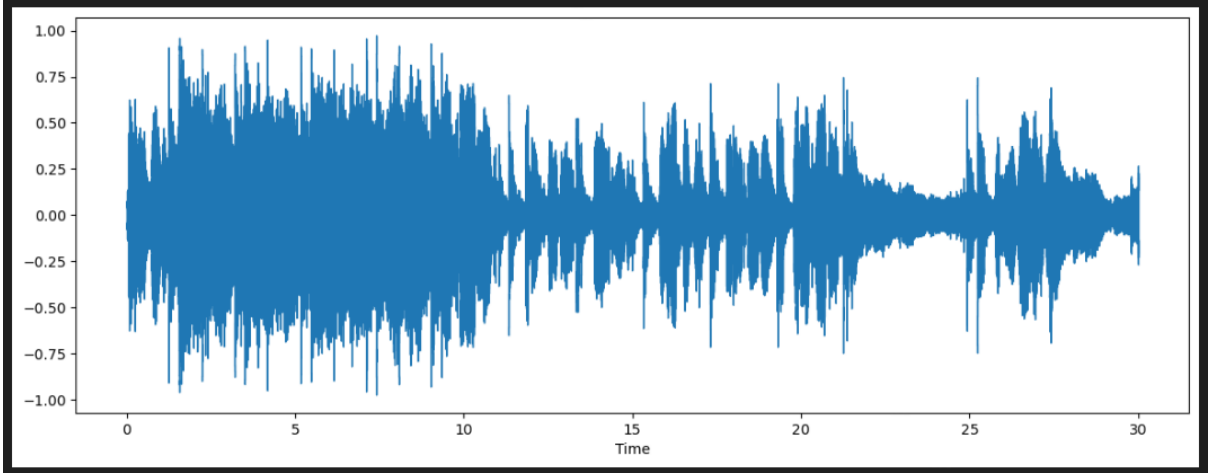


Figure 3.2.1.3. Waveform and audio specifications

In order to use audio signals / music for the task of genre classification it is necessary to extract relevant features from these signals. The features were extracted both taking into

account the whole excerpt and by dividing the musical file into smaller windows. Audio features extracted can be grouped into: temporal features, chroma features, spectral features and cepstrum features.

3.2.1.1. Chroma STFT

According to [9], the human perception of pitch is periodic in the sense that two pitches are perceived as similar in "color" (playing a similar harmonic role) if they differ by one or several octaves (an octave = the distance of 12 pitches). A pitch can be separated into two components, which are referred to as *tone height* and *chroma*. The tone height refers to the octave number and the chroma to the respective pitch spelling attribute. In Western music notation, the twelve chroma values are given by the set:

$$C, C\#, D, D\#, E, F, F\#, G, G\#, A, A\#, B$$

A pitch class is defined as the set of all pitches that share the same chroma; for example, the pitch class corresponding to the chroma $c = 0$ (C) consists of the set {0, 12, 24, 36, …} == {C0, C1, C2, C3, …}. The main idea of chroma features is to aggregate all spectral information that relates to a given pitch class into a single coefficient. Chroma features can be significantly changed by introducing pre/post-processing steps that modify spectral, temporal and dynamical aspects. The following chromagram is obtained by performing Short Time Fourier Transform (SFTF):

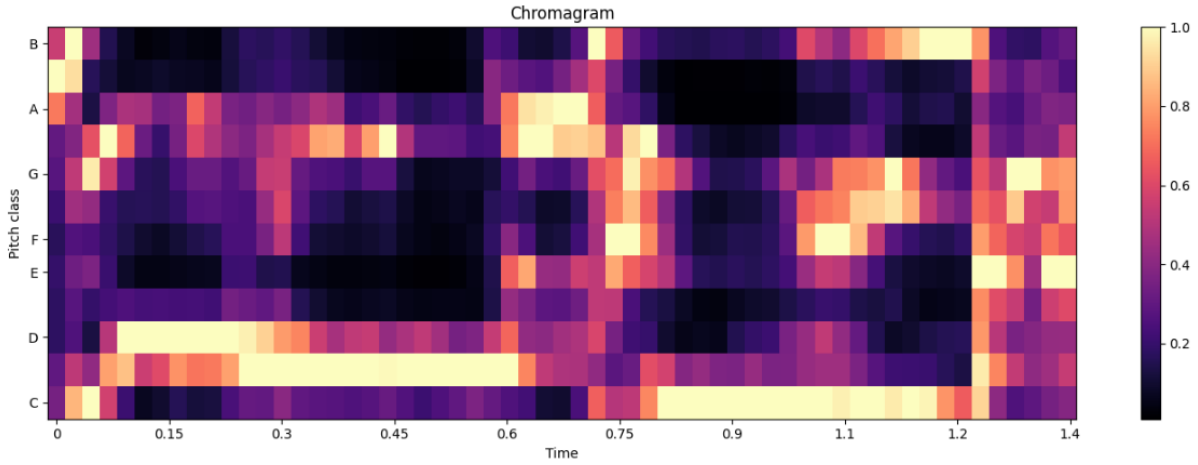


Figure 3.2.1.1.1. Chromagram on time window

3.2.1.2. Root-Mean-Square (RMS)

The RMS value of an audio signal, according to [33], is a measure of its overall energy or volume level. It is a measure of the amplitude of the audio signal over time. The RMS is a broad measure of an audio signal's volume and it does not take into account human perception of loudness. Human ears perceive some frequencies as being louder than others, even if they have the same RMS level. Mathematically, since the audio signals are often represented as continuous-time signals $x(t)$ over a period T, the RMS value is computed as follows:

$$RMS = \sqrt{\frac{1}{T} * \int_0^T x(t)^2 dt}$$

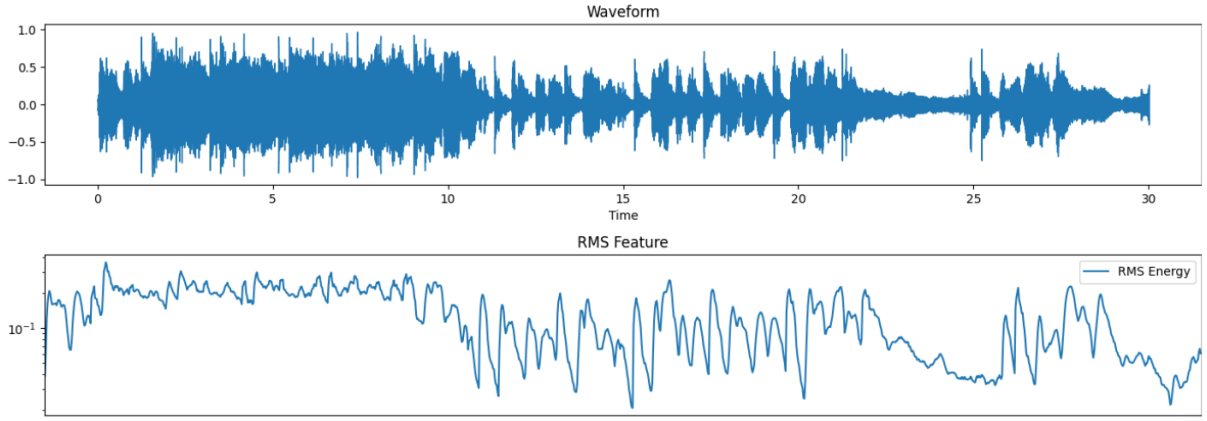


Figure 3.2.1.2.1. RMS and waveform

3.2.1.3. Spectral centroid

Many spectral features are described in [34], including spectral centroid, which is a measure to characterize a spectrum and indicates where the 'center of mass' for a sound is located. In other words, it gives the frequency band where the most of the energy is concentrated. It maps into a very prominent timbral feature called "brightness of sound" (energetic, open, dull). Mathematically, it is computed as the weighted mean of the frequency bins present in the audio signal, determined using a Fourier transform, with their magnitudes as the weights:

$$spectral\ centroid = \frac{\sum_{n=0}^{N-1} f(n)x(n)}{\sum_{n=0}^{N-1} x(n)}$$

$x(n)$: the weighted frequency value (magnitude) of bin number n

$f(n)$: the center frequency of that bin

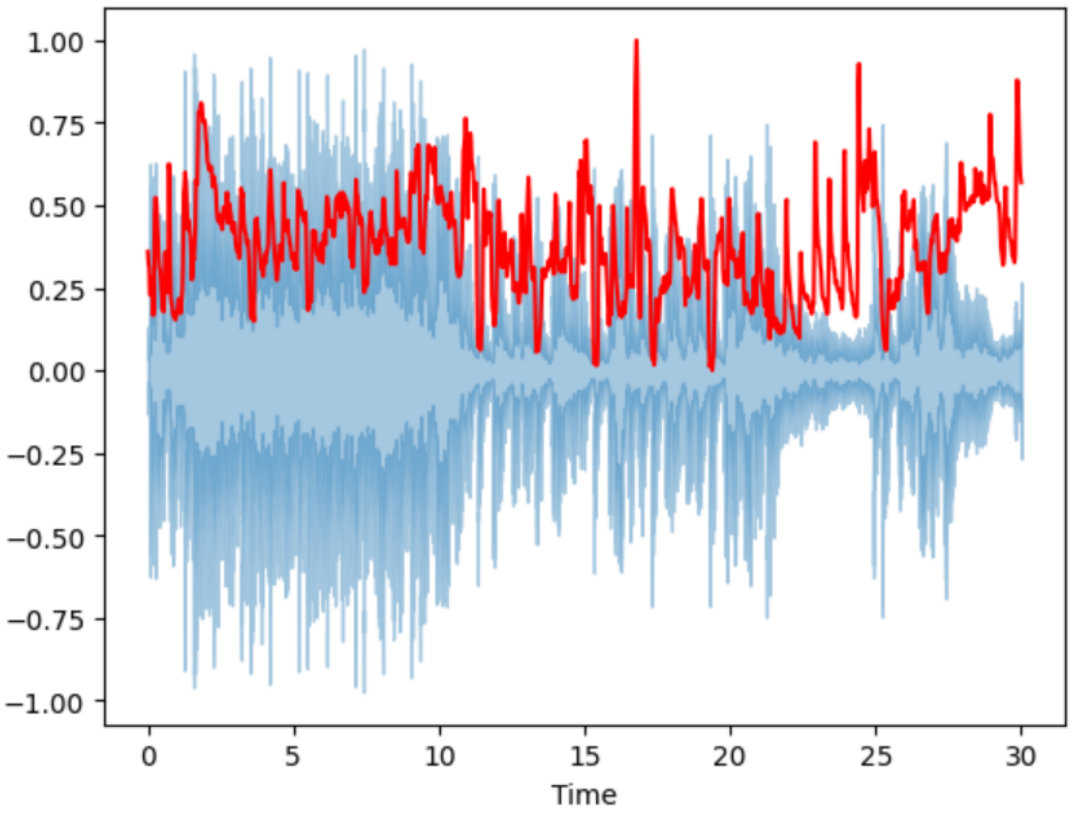


Figure 3.2.1.3.1. Spectral centroid and waveform

3.2.1.4. Spectral bandwidth

The spectral bandwidth, also described in [34], or spectral spread is derived from the spectral centroid. It is the spectral range of interest around the centroid, that is, the variance from the spectral centroid. It has a direct correlation with the perceived timbre. The bandwidth is directly proportional to the energy spread across frequency bands. It provides a measure of the spectral complexity of the sound: narrow-band signals (pure tones) have a low spectral bandwidth, whereas wide-band signals (like noise) have high spectral bandwidth.

Mathematically, it is the weighted mean of the distances of frequency bands from the Spectral Centroid:

$$spectral\ bandwidth\ = \sqrt{\frac{\sum_{n=0}^{N-1}(f(n)-centroid)^2 * x(n)}{\sum_{n=0}^{N-1} x(n)}}$$

$x(n)$: the weighted frequency value (magnitude) of bin number n

$f(n)$: the center frequency of that bin

$centroid$: the spectral centroid

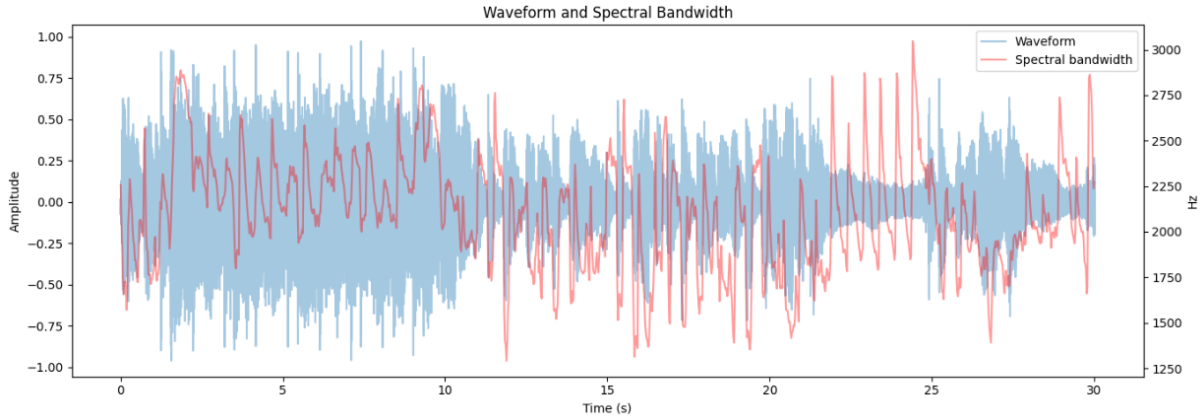


Figure 3.2.1.4.1. Spectral bandwidth and waveform

3.2.1.5. Spectral Rolloff

Spectral rolloff is defined in [35] as the N^{th} percentile of the power spectral distribution, where N is usually 85% or 95%. The rolloff point is the frequency below which the N of the magnitude distribution is concentrated. For example, this can be used to approximate the maximum / minimum frequency by setting roll percent to a value close to 1/0. It is a measure of the shape of the signal and represents the frequency below which the specified percentage of the total spectral energy lies.

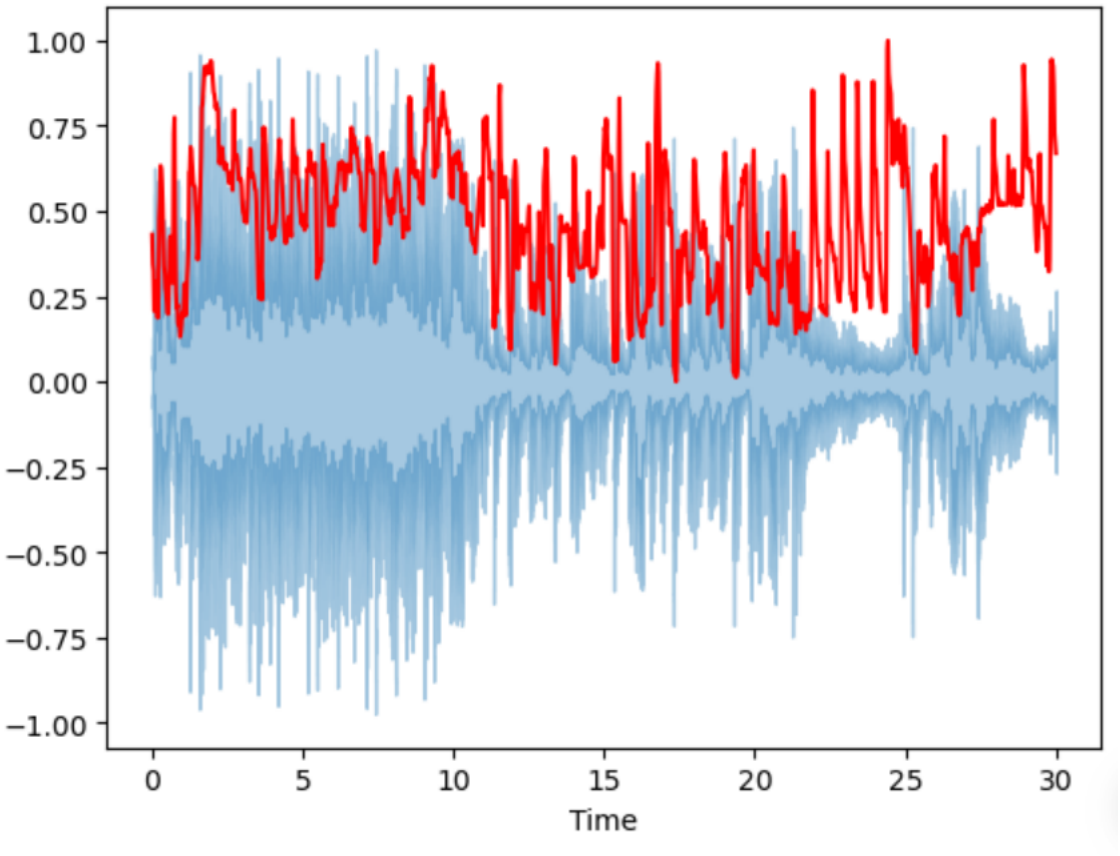


Figure 3.2.1.5.1. Spectral rolloff and waveform

3.2.1.6. Flatness

Spectral flatness [36] (or tonality coefficient, or Wiener entropy) is a measure used to characterize the audio spectrum. It provides a way to quantify how noise-like versus how tone-like a sound is. It is defined as the ratio of the geometric mean to the arithmetic mean of a power spectrum. If the spectrum is equally distributed (the same amount of power in all frequency bands), it is considered 'flat' and the spectral flatness is close to 1. Otherwise, if the spectrum is peaky (more power in some frequency bands than in others), it is considered 'sharp' and the value of the feature is close to 0. The value of the spectral flatness is computed as follows, where P_i represents the power of the i-th frequency bin:

$$geometric\ mean = \sqrt[N]{\prod_{i=0}^{N-1} P_i}$$

$$arithmetic\ mean = \frac{\sum_{i=0}^{N-1} P_i}{N}$$

$$spectral\ flatness = \frac{geometric\ mean}{arithmetic\ mean}$$

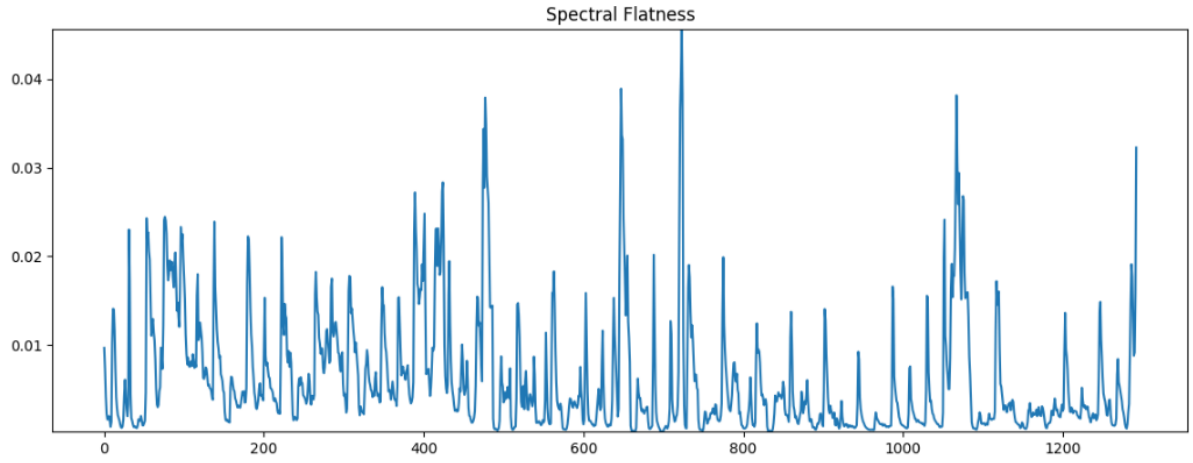


Figure 3.2.1.6.1. Spectral flatness

3.2.1.7. Contrast

Spectral contrast is a feature used to classify music genres. Spectral contrast is expressed as the difference in decibels between peaks and valleys in the frequency spectrum, which can represent the relative spectral characteristics of music. It can be seen from the experimental results [10] that the spectral contrast has a good ability to discriminate between music genres.

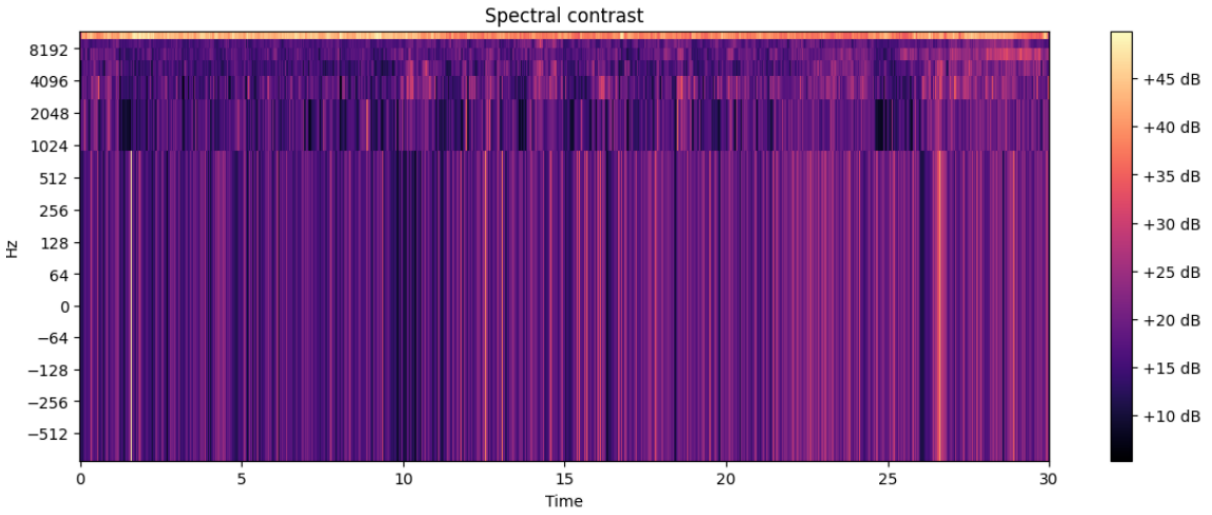


Figure 3.2.1.7.1. Spectral contrast log-scaled

3.2.1.8. Flux

Spectral flux, [37], measures the spectral change between two successive frames and is computed as the squared difference between normalized magnitudes of the spectra of the two successive short-term windows. It is a measure of how quickly the power spectrum of an audio signal is changing. It can be used to detect the onset of musical notes or beats. A high spectral flux might indicate a percussive sound, while a low flux might indicate a sustained note. In order to compute the value, the time-domain signal must be transformed into the frequency domain, by using either a Discrete Fourier Transform (DFT) or a Fast Fourier Transform (FFT). Mathematically, the formula is as follows:

$$F(t) = \sum (P(t,f) - P(t-1,f))^2$$

where $F(t)$ is the spectral flux at time 't' and is given by the sum over all frequencies 'f' in the analysis window. The power spectrum $P(t,f)$ is given by $P(t,f) = |X(t,f)|^2$, with $X(t,f)$ being the Fourier transformed signal at time 't' and frequency 'f'.

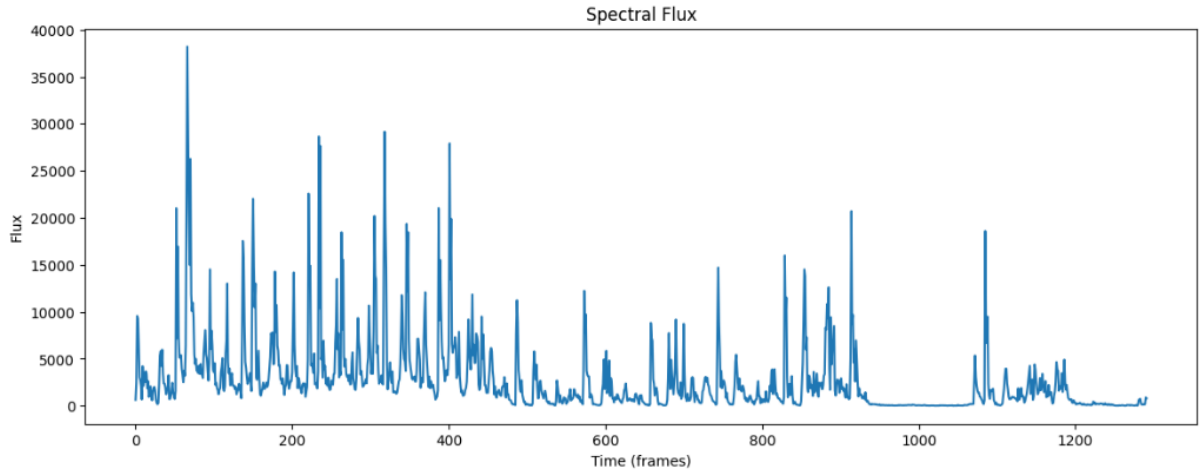


Figure 3.2.1.8.1. Spectral flux for each time frame

3.2.1.9. Zero crossing rate (ZCR)

The ZCR, described in [34], is the rate of sign-changes along a signal (positive-zero-negative or negative-zero-positive). It has higher values for highly percussive sounds like those in metal and rock. The ZCR is computed as follows, where T is the length of the time window, s_t is the magnitude of the t^{th} time domain sample:

$$ZCR = \frac{1}{T}\sum_{t=1}^{T-1}\frac{\left|sign\left(s_t\right)\right|-\left|sign\left(s_{t-1}\right)\right|}{2}$$

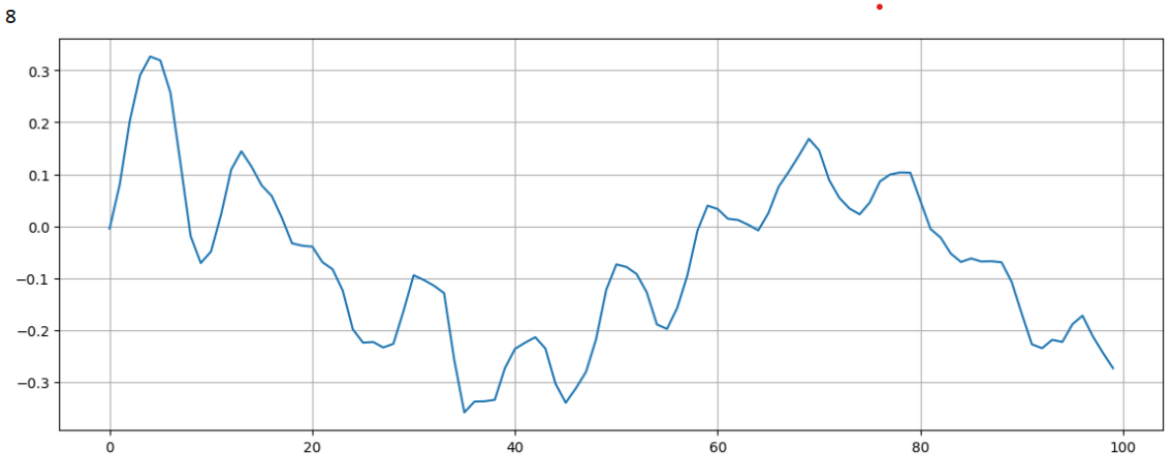


Figure 3.2.1.9.1. ZCR extracted from a time window

3.2.1.10. Harmonics and Perceptrual/Percussive

Harmonics [38] represent the multiple frequencies that are integer multiples of a fundamental frequency. When a musical instrument or a vocal chord produces a tone, it is not just a single frequency being produced, but a complex combination of tones known as the harmonic series. These overtones shape the timbre of the sound, contributing to the unique character that allows distinguishing between different types of sounds, even if they share the same pitch. Perceptual audio features are attributes of audio that the human ear can perceive directly. Key perceptual features include loudness, pitch, timbre and spatial location. Timbre, often referred to as the 'color' or 'texture' of a sound, is largely determined by the balance of harmonics in the sound and it is what allows one to distinguish between different instruments or voices even when they are playing or singing the same note.

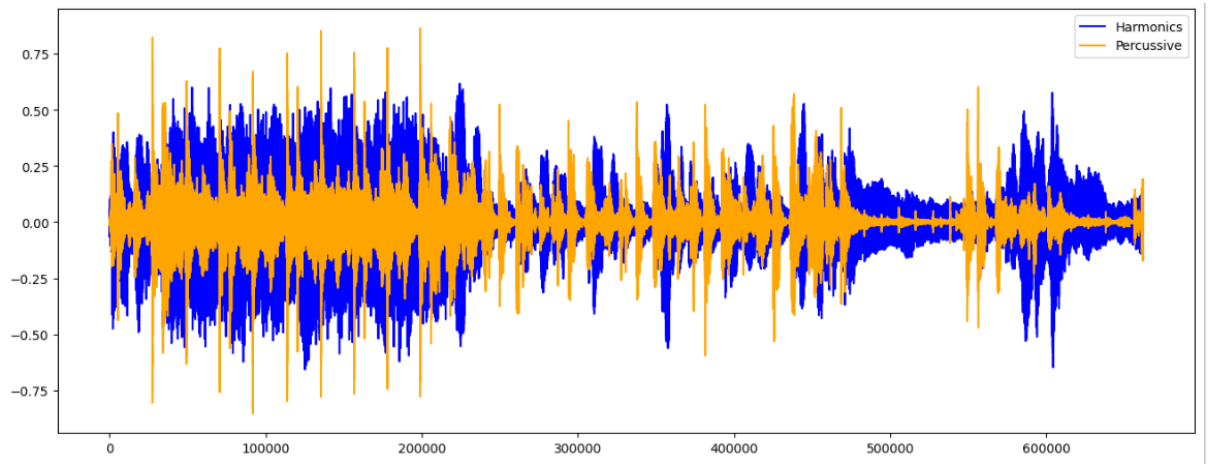


Figure 3.2.1.10.1. Harmonics and Perceptrual features

3.2.1.11. Tempo

In musical lingo, tempo is a that is used to describe the speed at which a song is played. In classical music, it is typically indicated at the beginning of a piece and it is usually measured in Beats Per Minute (BPM). The tempogram, [39], is a feature that represents the rhythmic structure of a musical piece in time, showing how the tempo of the track changes over time.

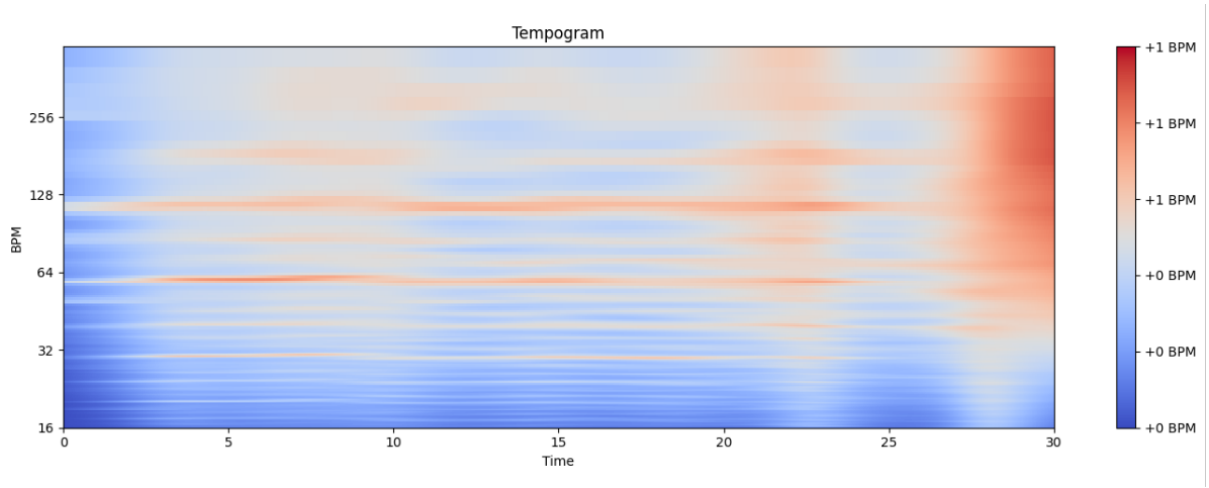


Figure 3.2.1.12.1. Tempogram of an audio file

3.2.1.12. Mel Frequency Cepstral Coefficients (MFCC)

Mel Frequency Cepstral Coefficients, described in [40], are a set of features based on human auditory perception, particularly the frequency intervals at which human ears discern individual tones. They are computed in the Mel Scale, which is a perceptual scale of pitches that approximates the human ear's response to different frequencies, equal distances in this

unit of pitch sound equally distant to the listener. What makes MFCC very useful is that they provide a robust representation of the spectral shape, which not only reduces computational requirements, but also helps to suppress noise.

MFCC is a representation of the short-term power spectrum of a sound, based on a linear cosine transform of a log power spectrum on a nonlinear mel scale of frequency. To explain this statement, several terms must be defined:

- a sound spectrum is a representation of a sound in terms of the amount of vibration at each individual frequency;
- sine and cosine transforms can be easily computed using fourier transform;
- the power spectrum is the result after the fourier transform, also known as periodgram;
- the frequency is scaled in mel scale in order to match what the human ear can hear: $Mel(f) = 1127 * ln(1 + \frac{f}{700})$;

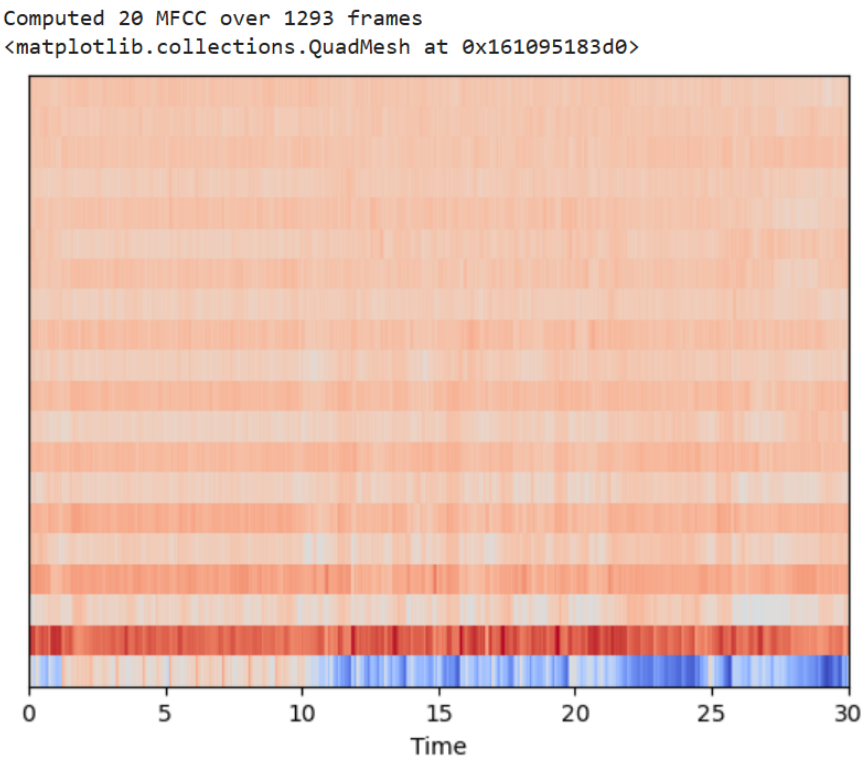


Figure 3.2.1.12.1. Mel Frequency Cepstral Coefficients

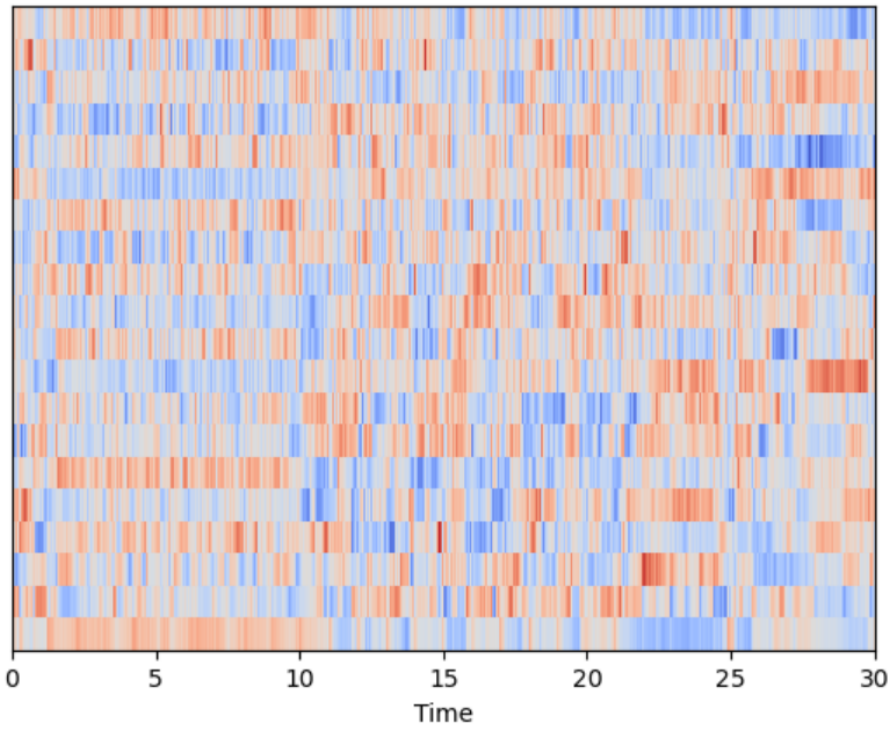


Figure 3.2.1.12.2. MFCC scaled to zero mean and unit variance

3.2.1.13. Mel Spectrogram

According to [11], a spectrogram is a visual representation of the spectrum of frequencies of a signal as it changes over time. The plot is created by showing frequency on one axis, time on another and intensity (magnitude) of the frequencies at each point in time on a third axis, which is often represented as color. Standard spectrograms have a significant limitation for human-centered tasks as they display the frequency on a linear scale. Human auditory perception is more accurately modeled with logarithmic frequency scale, especially at lower frequencies. The Mel scale is a perceptual scale of pitches that approximates the human ear's response to different frequencies. Thus, a Mel spectrogram is a spectrogram where the frequency is Mel-scaled in order to match human perception. To create a Mel spectrogram, the first step is to take the Fourier transform of the signal to obtain the spectrum of frequencies, then apply the Mel filterbank to this spectrum to obtain the result. The Mel filterbank is a set of triangular filters that are spaced according to the Mel scale and applying them amounts to calculating the sum of the energy in each filter.

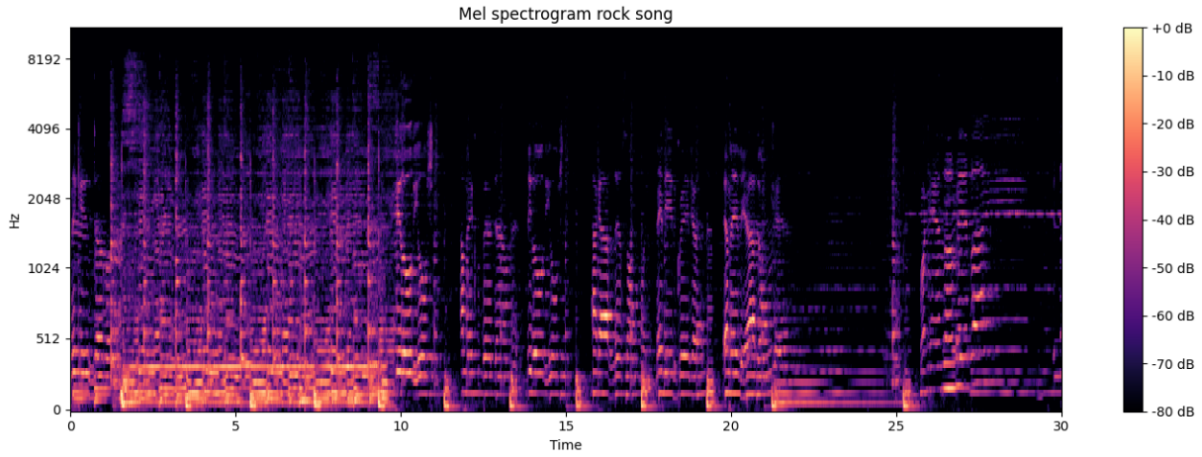


Figure 3.2.1.13.1. Mel Spectrogram

3.2.2. Machine Learning

3.2.2.1. KNN

K-Nearest Neighbors (KNN), as described by IBM in [41], is an instance-based learning method used in Machine Learning. It uses a distance function to identify the 'neighbors' that are most similar to the instance being examined. The fundamental idea is that similar things exist in close proximity. Hence, given an unknown data point, the algorithm looks for the 'k' number of points that are closest to it, with 'k' being a user-defined hyperparameter. The most commonly used distance metric used is the Euclidean distance, or the L2 norm. Once the nearest neighbors are identified, a majority vote is used to assign the instance to a class. For 2 points in a n-dimensional space (p and q), the Euclidean distance is given by:

$$d(p,q) = \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2}$$

3.2.2.2. SVM

Support Vector Machines (SVM), according to [42], is a supervised ML algorithm in which the central idea is to find the optimal hyperplane that maximally separates classes in a given dataset. The algorithm chooses a hyperplane in such a way that the distance from it to the nearest data point on each side, called the margin, is maximized. The data points that are closest to the hyperplane and essentially 'support' the maximization of the margin are called the support vectors. One advantage of SVM is that it is not limited to linear classification, it can also handle nonlinear classification. For this task it uses a kernel trick that involves

transforming the input data into a higher-dimensional space where a hyperplane can be used to separate the classes that previously were inseparable.

3.2.2.3. Random Forest

Random Forest, as described by IBM in [43], is an algorithm that consists of a large number of individual decision trees that operate as an ensemble. The idea is to combine the outputs from these smaller trees to generate the final output through majority voting. This combination of results often mitigates the overfitting problem that occurs in single decision trees. The construction of the model begins with the creation of multiple decision trees, each trained on a different subset of the total dataset (method known as "bagging"). The diversity among subsets helps to make a more robust and less prone to overfitting model. The algorithm relies on the "wisdom of the crowd" and is usually more accurate than the prediction of any individual tree, and also is relatively unaffected by outliers.

3.2.2.4. Logistic Regression

Logistic Regression, as described by IBM in [44], is a statistical analysis model used for prediction of the probability of occurrence of an event. It provides a method for modeling a binary response variable, but can be further tuned to handle multilabel classification problems through the "one-vs-rest" method. The basis is the logistic function (the sigmoid), which maps any real-valued number into a value in [0,1]. The sigmoid function transforms the linear output into a value representing probability, that can be thresholded (around the midpoint - 0.5) to produce binary classification. The logistic function is computed as follows:

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

3.2.2.5. Naïve Bayes

Naive Bayes, as described by IBM in [45], is a classification algorithm based on Bayes' theorem, which describes the relationship of conditional probabilities of statistical quantities. It provides a way to compute the probability of an instance belonging to a particular class, given prior knowledge. The intuition behind the theorem is the assumption of independence between every pair of features. Even if it is rarely true in real-world data, it simplifies the calculations and gives surprisingly good results. Bayes' theorem is the following:

$$P(y|x_1,\ldots,x_n) = \frac{P(y)*P(x_1|y)*\ldots*P(x_n|y)}{P(x_1,\ldots,x_n)}$$

3.2.2.6. XGBoost

Extreme Gradient Boosting (XGBoost), [46], is a ML algorithm based on the principles of gradient boosting. This is an ensemble learning method that sequentially combines simple models (weak learners, usually decision trees), in a way that each new model compensates for the errors of the previous ones. XGBoost uses regularization to control overfitting on both tree structure (maximum depth) and leaf weights (L1/L2 regularization). One of the core ideas is the implementation of gradient boosting in a manner that makes it computationally efficient and thus scalable.

3.2.3. Deep Learning

3.2.3.1. CNN

Convolutional Neural Networks (CNNs), as specified in [47], are a type of DL algorithm, specifically designed to process grid-structured data, like images. The main idea behind this model is the convolutional layer, which consists of many filters or kernels. These filters slide across an image in order to perform a convolution operation. This operation involves element multiplication of the filter and a portion of the image, followed by a sum, thus reducing the spatial dimensions of the input, while encoding more complex information about the data. As the model is trained, the filters learn to extract from low-level features such as edges, shapes, textures, to complex objects in later layers of the network. Pooling is typically applied after a convolutional layer in order to reduce the spatial size of the representation, resulting in a lower chance of overfitting and lower computational complexity. At the end of the network, there are typically fully connected layers that perform the final classification task.

3.2.3.2. RNN

Recurrent Neural Networks (RNNs), described in [48], are a class of neural networks designed to recognize patterns in sequences of data, such as text, spoken words. At the core, there is the concept of “recurrence”: RNNs maintain information in states and pass it along from one step in the sequence to the next, creating a form of internal memory. These networks should be able to connect past information to the present task, however, in practice,

long-term dependencies fail to be retained due to the "vanishing-gradient" problem. To solve this problem, more sophisticated RNNs were introduced, like LSTM, which includes "memory cells".

3.2.3.3. Transfer Learning

Transfer Learning is a technique where a pre-trained model is reused as a starting point for a different, yet related task. The article [49] explains that many features, like edges or shapes in images or certain word associations in text data, are universal and can be useful in multiple tasks. Just like humans leverage previous knowledge when learning a new task, in the same way models reduce the need to relearn all the features from the scratch when moving to a new task. Neural networks with millions of parameters, which were trained on massive datasets like ImageNet, can be fine-tuned to perform similar tasks, but much more efficiently than training the model from the beginning. This results in a reduction of the computational expense and in the volume of data required for training models.

3.2.3.5. Reinforcement Learning

Reinforcement Learning (RL) is a type of AI that revolves around an agent learning to make the best decisions by interacting with an environment, as said in [50]. The whole process is driven by rewards and penalties, the agent attempting to maximize its cumulative reward over time. This process involves experimenting with different actions and learning from the feedback received from the environment: either as rewards, to encourage its behavior, or as penalties, to discourage specific actions. The strategy (policy) used by the agent is continuously improved and it describes how the agent should take actions based on its current state.

When it comes to music genres recognition, RL can be applied in a unique way: an agent is trained to classify a song's genre based on different features like rhythm, melody and is rewarded when it correctly identifies the track label. The accuracy is improved over time, when the agent fine-tunes its policy to maximize the gathered rewards. Regarding music creation, an agent can be tasked with composing a piece of music that is pleasing to the listener and stylistically consistent. The action consists in choosing a note or a sequence of

notes, which receive positive or negative feedback based on its coherence and appeal. The aim is to generate high-quality compositions by maximizing the reward function.

Despite its potential, the RL technique may not be the best choice, as it requires a large amount of interaction with the environment to effectively learn, which is computationally intensive and time-consuming. In music genre classification, supervised learning methods like SVM or Neural Networks outperform RL, being able to generalize better with less data. Similarly, for music generations, there are other methods like GANs or VAEs that could be more efficient, as they can learn patterns in existing music and generate new compositions more easily. Furthermore, the feedback function is difficult to define clearly in creative tasks such as music compositions, leading to a less stable and slower learning.

3.3. Music Generation

3.3.1. Music notation

Music notation serves as a visual coding system to depict music, using a set of symbols. Each music notation is different and is used in its own way, for example the tablature notation is instrument-specific and provides information about the physical placement of the performer's fingers. Tab consists of horizontal lines that represent the strings of the instrument and numbers that represent the frets where the fingers should be placed. It can also indicate techniques such as slides, harmonics or vibrato. Despite the effectiveness of many traditional forms of musical notation for live performances, they have visible limitations when it comes to playing with music in the digital realm. Here, the Musical Instrument Digital Interface (MIDI) excels. MIDI [51] is a protocol that enables computers and musical instruments to communicate with each other. It does not actually contain any sound, as it encodes information about the audio track, like the pitch, velocity, vibrato or volume. The nature of this notation allows for easy transposition, changing time signatures or instruments (the piano part can be easily switched with any other instrument in the library). MIDI is also very efficient and economic: a symphony that might require hundreds of traditional sheet pages can be stored in a lightweight file, making it perfect to work with.

Octave	Note numbers											
	Do	Do#	Re	Re#	Mi	Fa	Fa#	Sol	Sol#	La	La#	Si
	C	C#	D	D#	E	F	F#	G	G#	A	A#	B
0	0	1	2	3	4	5	6	7	8	9	10	11
1	12	13	14	15	16	17	18	19	20	21	22	23
2	24	25	26	27	28	29	30	31	32	33	34	35
3	36	37	38	39	40	41	42	43	44	45	46	47
4	48	49	50	51	52	53	54	55	56	57	58	59
5	60	61	62	63	64	65	66	67	68	69	70	71
6	72	73	74	75	76	77	78	79	80	81	82	83
7	84	85	86	87	88	89	90	91	92	93	94	95
8	96	97	98	99	100	101	102	103	104	105	106	107
9	108	109	110	111	112	113	114	115	116	117	118	119
10	120	121	122	123	124	125	126	127				

Figure 3.3.1.1. Notes representation in MIDI format

3.3.2. LSTM

Long Short-Term Memory network (LSTM) is a type of Recurrent Neural Network (RNN) capable of learning long-term dependencies. According to [52], it addresses the issue of vanishing and exploding gradients, which can be found in the conventional RNN, enabling the model to learn from data where relevant events may have significant time delays between them. The core idea is the cell state, which carries information throughout processing steps. The information added and removed from the cell state is regulated by structures called gates. This process is done using sigmoid functions, describing how much of each component should be let through. The forget fate decides the amount of past information to be discarded, the input gate decides how much new information should be stored in the cell and the output gate returns the final output based on the input and the current cell state.

Just like a sentence is a sequence of words, the music is a sequence of notes or chords and therefore can be well represented by these cells that model long-term dependencies. Again, similar to a language model that predicts the next word in a sentence, the LSTM can learn to generate new music by providing an initial seed sequence and then repeatedly sampling from its output distribution and feeding the output back as input for the next step.

3.3.3. Transformer

The Transformer model is introduced in [12] and has become a cornerstone in the world of AI, particularly for tasks involving sequence-to-sequence predictions. It marked a departure from the traditional RNN and LSTM by discarding recurrence in favor of self-attention

mechanisms. This attention mechanism weighs the relevance of input elements, allowing the model to focus on the most important elements in the context. This results in the ability to handle long-range dependencies in the input data, which is crucial in tasks like language translation and music generation. Transformers consist of an encoder that reads a sequence of tokens (words, musical notes) at once and encodes it into a set of continuous representations, and a decoder that generates an output one part at a time, consuming the set of representations from the encoder.

A Transformer model can be trained in a similar fashion as the LSTM. After training, a seed sequence of notes is provided and the model generates the next notes based on the input. The process is repeated until a complete musical piece is created. The resulting music is often remarkably coherent and pleasing, and with its flexible architecture, one could also train the model to create music in a specific style.

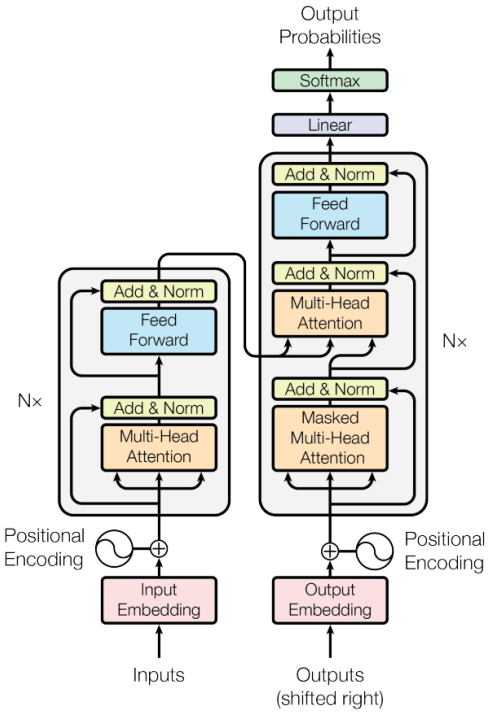


Figure 3.3.3.1. Transformer's architecture

3.3.4. VAE

Firstly introduced in [13], Variational Autoencoders (VAEs) are a type of generative model that consist of an encoder, which takes high-dimensional input data and compresses it into a lower-dimensional latent representation, and a decoder, which takes the latent space and

attempts to reconstruct the original many dimensional data. The objective function of a VAE is called the evidence lower bound (ELBO) and it is composed of 2 terms: the reconstruction loss, which encourages the decoded outputs to match the original inputs, and a Kullback-Leibler (KL) divergence term, which forces the latent variable distributions to be close to a standard normal distribution. It is the trade-off between these 2 variables that ensures the model learns useful, generalizable representations in the latent space.

VAE can be used to generate several types of data, including images and music. After training, new music can be generated by sampling points in the latent space and decoding them into notes. Because of the continuity of the latent space, small changes in the variables lead to small changes in the generated output, allowing for interesting creations like combining 2 pieces of music or varying specific musical attributes. Furthermore, conditioning the VAE on additional variables (such as the genre of a piece), lead to a control over certain aspects of the generated music.

The formula for KL distance (divergence) is as follows (for discrete/continuous distributions):

$$D_{KL}(P||Q) = \sum_{x \in X} P(x)\, log\frac{P(x)}{Q(x)} = \int_{-\infty}^{\infty} P(x)\, log\frac{P(x)}{Q(x)} dx$$

3.3.5. GAN

Generative Adversarial Networks (GANs) are a class of generative models firstly introduced in [14]. They consist of 2 neural networks that are trained simultaneously in a competitive setup: a generator and a discriminator. The generator creates fake data to pass to the discriminator, which tries to distinguish the fake samples from real data. Training at the same time enables the generator to gradually improve its ability to create realistic data, while the discriminator improves at identifying fakes (adversarial relationship).

Regarding music generation, this model can be trained either on a dataset of music in a MIDI format, or on spectrogram representation, as it is often used in the generation of images with changed styles. Both VAE and GAN offer promising approaches to music generation. VAE are particularly suitable for tasks requiring fine-grained control over the generated music, such as varying specific musical attributes or interpolating different styles, but can sometimes produce blurry outputs. On the other hand, GAN can generate more realistic samples, as it

optimizes the generator's ability to fool the discriminator, rather than reconstructing the input data. However, its downside is that it can be difficult to train due to the fact that the generator might learn to produce only a limited variety of samples.

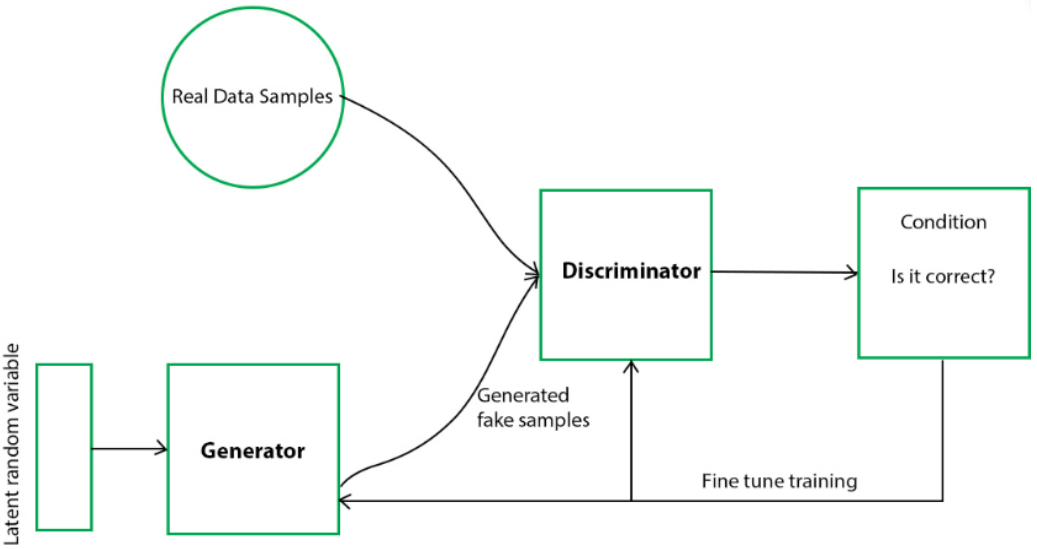


Figure 3.3.5.1. GANs components

4. Proposed framework

4.1. Music Genre Recognition

4.1.1. Datasets

When seeking the perfect dataset for the music genre classification task, it is essential to consider the size of the dataset, its diversity, balance and the quality of the genre labels. A synthesis of many musical datasets is presented in [15]:

dataset[1]	#clips	#artists	year	audio
RWC [12]	465	-	2001	yes
CAL500 [45]	500	500	2007	yes
Ballroom [13]	698	-	2004	yes
GTZAN [46]	1,000	∼ 300	2002	yes
MusiClef [36]	1,355	218	2012	yes
Artist20 [7]	1,413	20	2007	yes
ISMIR2004	1,458	-	2004	yes
Homburg [15]	1,886	1,463	2005	yes
103-Artists [30]	2,445	103	2005	yes
Unique [41]	3,115	3,115	2010	yes
1517-Artists [40]	3,180	1,517	2008	yes
LMD [42]	3,227	-	2007	no
EBallroom [23]	4,180	-	2016	no[2]
USPOP [1]	8,752	400	2003	no
CAL10k [44]	10,271	4,597	2010	no
MagnaTagATune [20]	25,863[3]	230	2009	yes[4]
Codaich [28]	26,420	1,941	2006	no
FMA	**106,574**	**16,341**	**2017**	**yes**
OMRAS2 [24]	152,410	6,938	2009	no
MSD [3]	1,000,000	44,745	2011	no[2]
AudioSet [10]	2,084,320	-	2017	no[2]
AcousticBrainz [32]	2,524,739[5]	-	2017	no

[1] Names are clickable links to datasets' homepage.
[2] Audio not directly available, can be downloaded from ballroomdancers.com, 7digital.com, youtube.com.
[3] The 25,863 clips are cut from 5,405 songs.
[4] Low quality 16 kHz, 32 kbit/s, mono mp3.
[5] As of 2017-07-14, of which a subset has been linked to genre labels for the MediaEval 2017 genre task.

Table 1: Comparison between FMA and alternative datasets.

Figure 4.1.1.1. Comparison between audio datasets

but the experiments took into account only three of the most commonly utilized datasets for this purpose: GTZAN [16], Free Music Archive (FMA) [17] and the Million Song Dataset (MSD) [18]. Each dataset has its unique strengths and weaknesses, that were carefully analyzed before choosing the perfect candidate for analysis and experimentation.

4.1.1.1. GTZAN

Usually, GTZAN is the first choice for music genre recognition tasks, primarily due to its simplicity, reduced size and balanced structure. It provides a neat, evenly distributed collection of a thousand audio tracks across ten popular genres. Because of the balanced genres, the need for additional preprocessing is eliminated; otherwise, the class imbalance could have caused bias in the models. The collection contains a folder with the 30 seconds audio files, each subfolder is labeled with one of the 10 genres and contains the respective 100 files. The other folder contains the visual representation of each audio file (Mel Spectrograms), also distributed in 10 balanced subfolders. Furthermore, there are two CSV files containing features of the audio files. One has the mean and variance computed over multiple features extracted from the entire 30 seconds songs, while the other has the same structure, but the songs were split into 3 seconds, increasing 10 times the amount of data fed to the classification models. Even if it seems that 3 second windows cannot properly model an entire genre, the results tend to agree with the motto "the more data the better". There are 29 features that can be used for the classification task (the columns in the tables). The filename, length and the label of the song will not be used in the models, so they are removed from the tables before any data processing.

```python
dataset_30sec = pd.read_csv('./Data/features_30_sec.csv')
dataset_30sec.columns
```

Python

```
Index(['filename', 'length', 'chroma_stft_mean', 'chroma_stft_var', 'rms_mean',
       'rms_var', 'spectral_centroid_mean', 'spectral_centroid_var',
       'spectral_bandwidth_mean', 'spectral_bandwidth_var', 'rolloff_mean',
       'rolloff_var', 'zero_crossing_rate_mean', 'zero_crossing_rate_var',
       'harmony_mean', 'harmony_var', 'perceptr_mean', 'perceptr_var', 'tempo',
       'mfcc1_mean', 'mfcc1_var', 'mfcc2_mean', 'mfcc2_var', 'mfcc3_mean',
       'mfcc3_var', 'mfcc4_mean', 'mfcc4_var', 'mfcc5_mean', 'mfcc5_var',
       'mfcc6_mean', 'mfcc6_var', 'mfcc7_mean', 'mfcc7_var', 'mfcc8_mean',
       'mfcc8_var', 'mfcc9_mean', 'mfcc9_var', 'mfcc10_mean', 'mfcc10_var',
       'mfcc11_mean', 'mfcc11_var', 'mfcc12_mean', 'mfcc12_var', 'mfcc13_mean',
       'mfcc13_var', 'mfcc14_mean', 'mfcc14_var', 'mfcc15_mean', 'mfcc15_var',
       'mfcc16_mean', 'mfcc16_var', 'mfcc17_mean', 'mfcc17_var', 'mfcc18_mean',
       'mfcc18_var', 'mfcc19_mean', 'mfcc19_var', 'mfcc20_mean', 'mfcc20_var',
       'label'],
      dtype='object')
```

Figure 4.1.1.1.1. The columns of the CSV files

The relationship between the variables, only their mean being taken into consideration, can be seen through the correlation heatmap. From here, the process of data cleaning and feature selection can begin, looking at the "heat" patterns. It can be observed that there are several variables exhibiting a positive correlation and a few that display an inverse correlation.

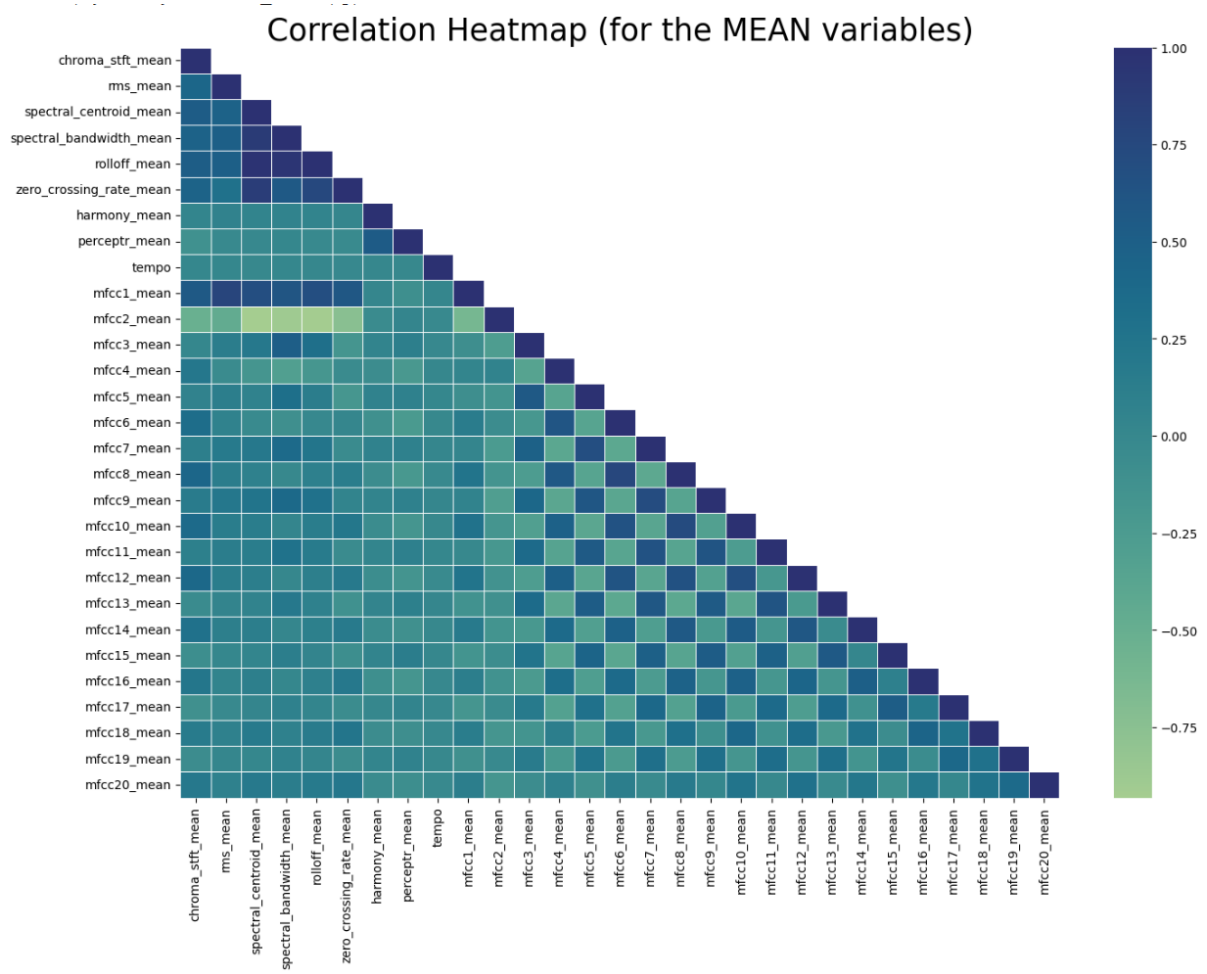


Figure 4.1.1.1.2. Correlation Heatmap for audio features

In order to visualize possible clusters of genres, Principal Component Analysis has been made. PCA is sensitive to scaling, so it is important that the variables are filtered (filename removed) and properly normalized before applying the analysis. The first two principal components often explain a significant proportion of the data's variation and can indicate whether the data can be represented in this reduced two-dimensional space without losing much information. The colored PCA points may naturally form clusters, which could correspond to different categories and could indicate underlying patterns or structures.

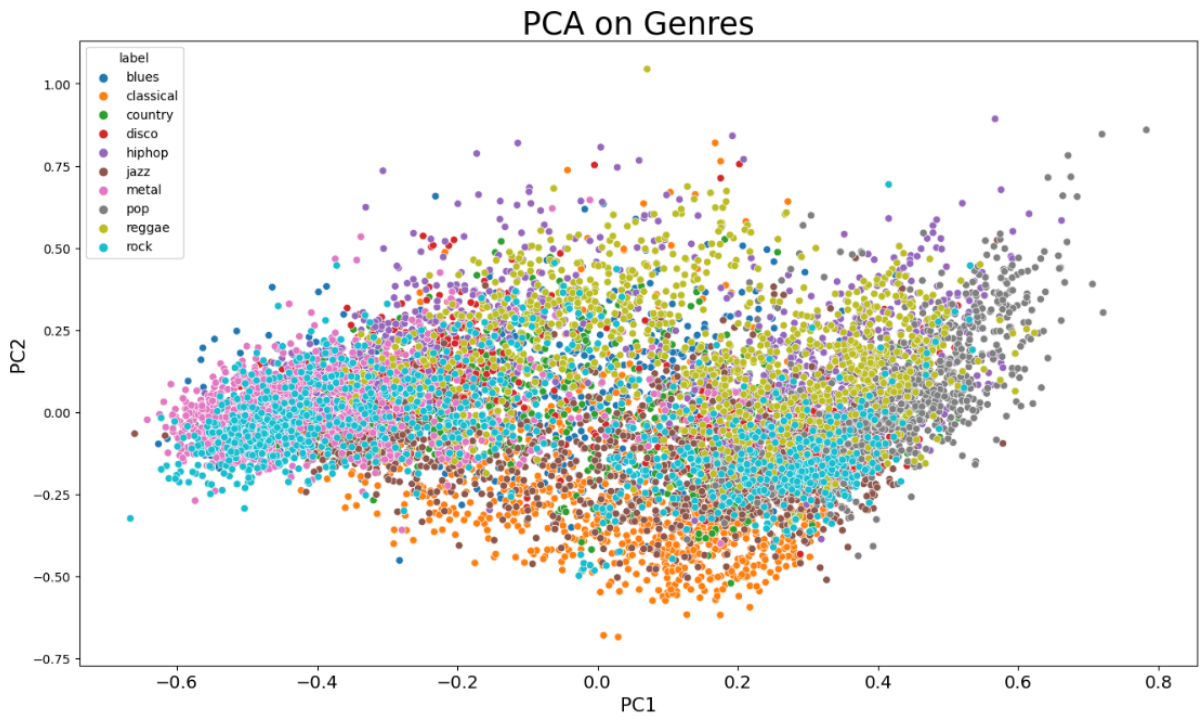


Figure 4.1.1.1.3. PCA clusters for each genre

In what concerns the folder that contains images, there are clear differences in the corresponding Mel Spectrograms, which can lead to an efficient use of Deep Learning techniques for classifying these visual representations into musical genres. The original files are 288x432 which may influence the final results when using Transfer Learning methods, so the images were re-generated in an even format of 256x256.

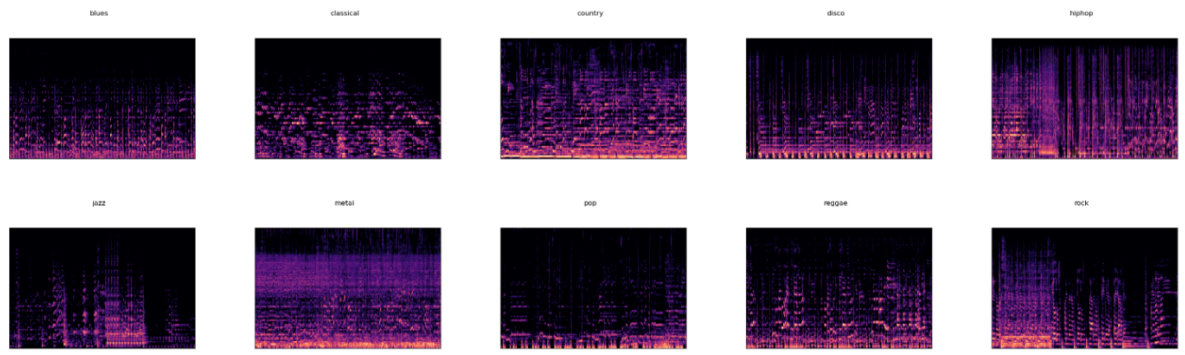


Figure 4.1.1.1.4. Mel Spectrograms for each of the 10 genres

While working with the spectrograms, it has been observed that the ‘jazz’ folder contains only 99 images, instead of 100 for each audio file. Digging a bit into the problem, it seems that the file ‘jazz00054.png’ is missing from the dataset and the audio file ‘jazz.00054.wav’ is corrupted. In order to maintain the balance between the 10 genres, a spectrogram from the same genre has been duplicated, in hopes that it does not affect the performance of the model too much. Despite its widespread use in the field of MIR, [19] it is indicated that the GTZAN is faulty and has an unexpected bias on the performance of the models and potentially skews the results. The most significant issue with the dataset is the presence of duplicate and mislabeled audio tracks. Around 6% of the songs are duplicates and several others are mislabeled with incorrect genres, which can lead to confusion in the training and even inflate the accuracy of the models. Furthermore, the distribution of the dataset across the 10 defined genres is balanced, which helps the process of data preprocessing but does not reflect the actual distribution of genres in the real world. Music is highly diverse and does not neatly fall into only 10 categories. Real-world systems would need to handle a far greater variety of music of varying quality and in a range of different environments. However, for the sake of computational efficiency it is enough to limit the musical spectrum to the most popular genres.

4.1.1.2. FMA

The Free Music Archive (FMA) dataset presented in [15] is a richly annotated, high-quality and diverse collection of music that can be utilized for various research tasks related to music analysis, including genre tagging. It provides several levels of annotation, for each track there is a unique ID, title, album and release year. Additionally, there are track genres and sub-genres, providing a robust groundwork for a broad spectrum of music analysis tasks. Tracks are categorized according to a hierarchical taxonomy that spans 16 top-level genres (like Pop, Rock, Electronic), which are further broken down into 161 sub-genres (such as Psych-Rock, Lo-Fi, Drone).

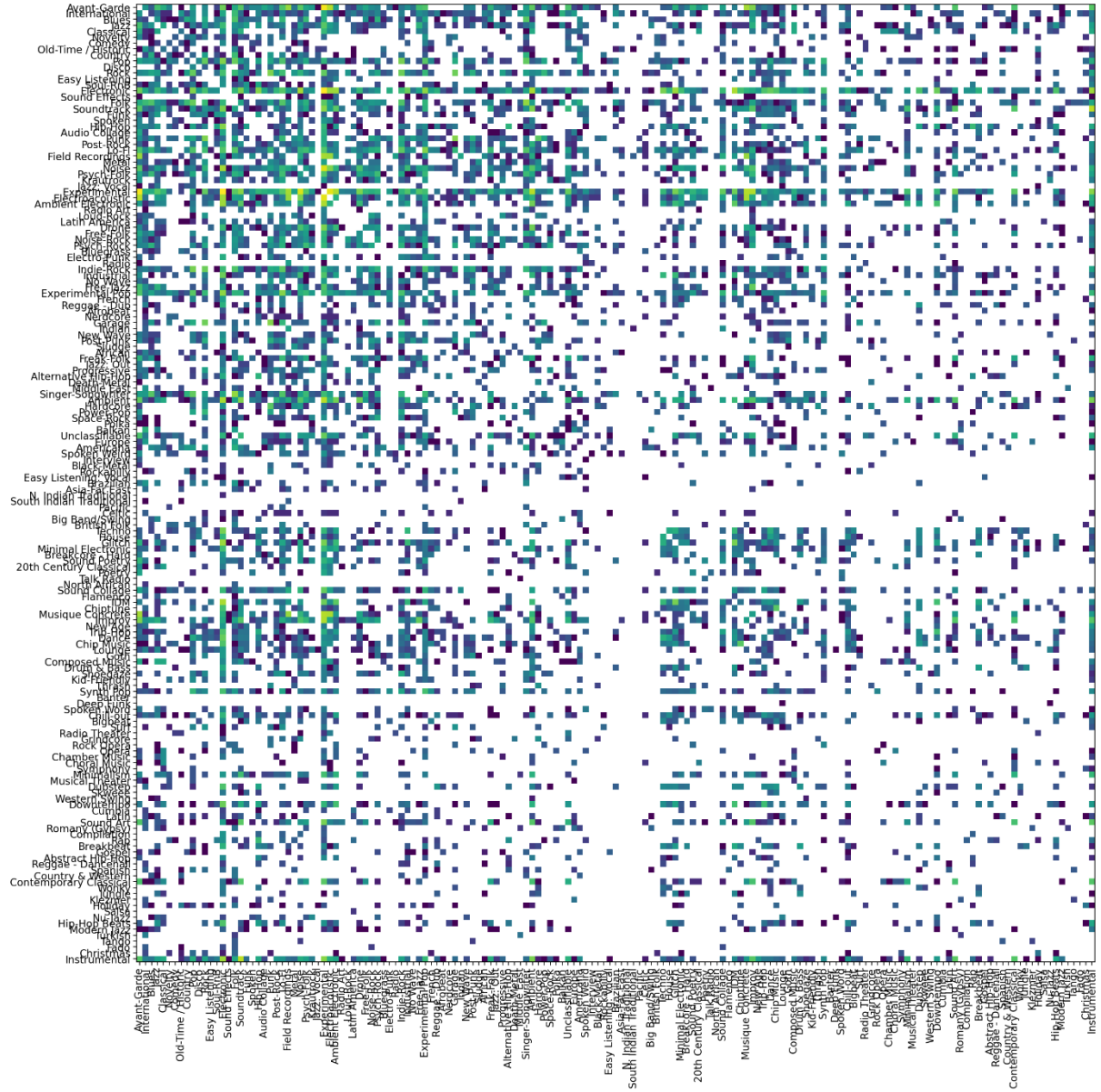


Figure 4.1.1.2.1. Related genres by cross-appearance

One important feature of this dataset is its availability in subsets of different sizes: small, medium, large and full versions. Starting from the small subset which is featuring 8,000 clips it stretches to the full-length version with more than 106,000 tracks, making this dataset suitable for a variety of projects, from personal small scale studies to large scale experiments.

dataset	clips	genres	length	size	
			[s]	[GiB]	#days
small	8,000	8	30	7.4	2.8
medium	25,000	16	30	23	8.7
large	106,574	161	30	98	37
full	106,574	161	278	917	343

Figure 4.1.1.2.1. Subsets of the FMA

title	index	genre_id	#tracks	parent	top_level	#training	#validation	#test	val_ratio	test_ratio
Rock	11	12	32923	0	12	5681	711	711	7.990155	7.990155
Electronic	14	15	34413	0	15	5050	632	632	7.990506	7.990506
Experimental	31	38	38154	0	38	1801	225	225	8.004444	8.004444
Hip-Hop	20	21	8389	0	21	1761	220	220	8.004545	8.004545
Folk	16	17	12706	0	17	1215	152	152	7.993421	7.993421
Instrumental	162	1235	14938	0	1235	1045	131	174	7.977099	6.005747
Pop	9	10	13845	0	10	945	122	119	7.745902	7.941176
International	1	2	5271	0	2	814	102	102	7.980392	7.980392
Classical	4	5	4106	0	5	495	62	62	7.983871	7.983871
Old-Time / Historic	7	8	868	0	8	408	51	51	8.000000	8.000000
Jazz	3	4	4126	0	4	306	39	39	7.846154	7.846154
Country	8	9	1987	0	9	142	18	18	7.888889	7.888889
Soul-RnB	13	14	1499	0	14	94	18	42	5.222222	2.238095
Spoken	19	20	1876	0	20	94	12	12	7.833333	7.833333
Blues	2	3	1752	0	3	58	8	8	7.250000	7.250000
Easy Listening	12	13	730	0	13	13	2	6	6.500000	2.166667

Figure 4.1.1.2.2. FMA medium partition details

title	index	genre_id	#tracks	parent	top_level	#training	#validation	#test	val_ratio	test_ratio
Hip-Hop	20	21	8389	0	21	800	100	100	8.0	8.0
Pop	9	10	13845	0	10	800	100	100	8.0	8.0
Folk	16	17	12706	0	17	800	100	100	8.0	8.0
Experimental	31	38	38154	0	38	800	100	100	8.0	8.0
Rock	11	12	32923	0	12	800	100	100	8.0	8.0
International	1	2	5271	0	2	800	100	100	8.0	8.0
Electronic	14	15	34413	0	15	800	100	100	8.0	8.0
Instrumental	162	1235	14938	0	1235	800	100	100	8.0	8.0

Figure 4.1.1.2.3. FMA small partition details

In terms of audio features, the dataset provides a rich selection of precomputed features using librosa library, including chroma, spectral features, tonnetz, mfcc and others. Each feature set (except zero-crossing rate) is computed on windows of 2048 samples spaced by hops of 512 samples, resulting in seven statistics over all windows: mean, standard deviation, skew, kurtosis, median, minimum and maximum.

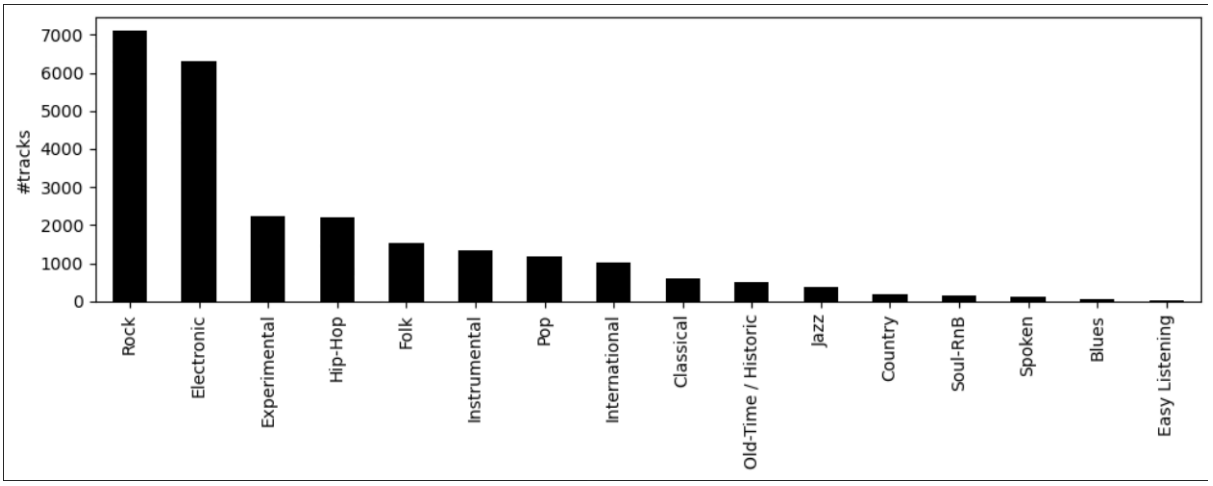


Figure 4.1.1.2.4. FMA medium dataset genres distribution

While experimenting with spectrograms for the medium dataset, several files were found to be erroneous, mostly because of the audio file's length, which did not match with the 30 seconds that each of the other files had. In addition to this, there is also a ticket on Github regarding this issue: [20].

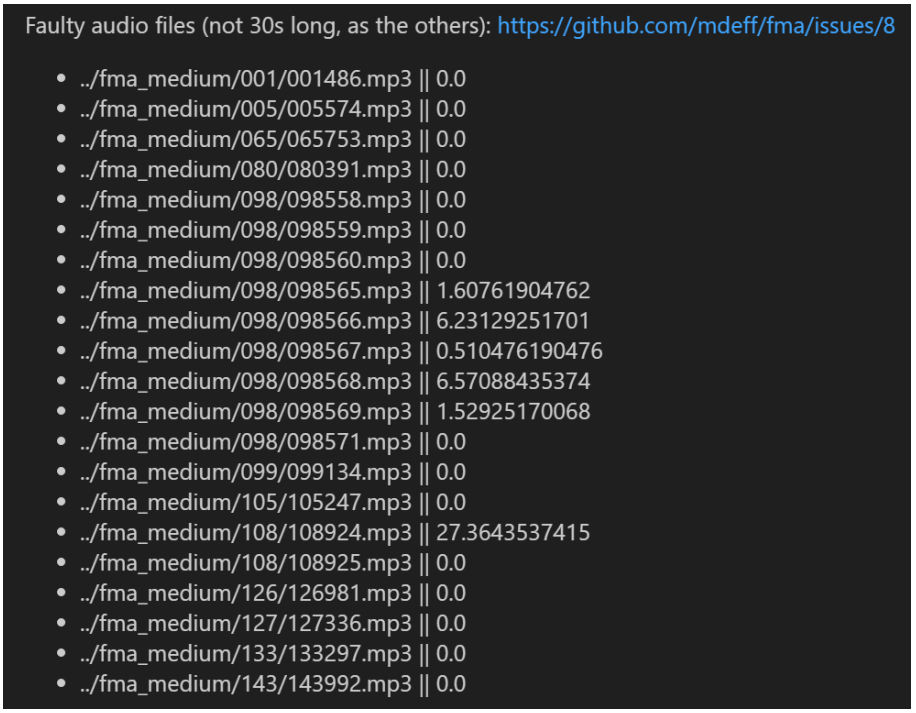


Figure 4.1.1.2.5. FMA dataset faulty audio files

After an in-depth exploratory data analysis, the conclusion is that the small partition of the FMA dataset is too small, containing only 8 genres that are not inclusive enough (classical, blues and other popular genres missing), while the medium subset is way too unbalanced and too large to be computational efficient.

4.1.1.3. MSD

The Million Song Dataset (MSD) is a publicly accessible collection of audio features and metadata for a million contemporary music tracks. With a million songs included, its sheer size offers an extensive range of data for all kind of tasks in music research, including music genre classification. Each song includes data points like the release year, the artist, the popularity, the key, tempo or duration. In addition to this, there are segments, bars and beats indicating the rhythm and the timbre of audio segments, which give an idea of the sound color and melody. The data for each track has been computed using an API called The Echo Nest's, which gives objective information about each song, unaffected by subjective factors like personal opinion or cultural context.

However, there are also a few drawbacks to this dataset. Firstly, it includes only metadata and precomputed features, it does not contain the actual audio files for the songs, due to copyright reasons. For researching with deep learning techniques using melspectrograms, this dataset is limited. Secondly, the MSD is heavily weighted towards popular Western music. Even if it offers an extensive range of data for this segment, it does not provide a diverse or representative sample of all genres within world music.

Genre Name	Number of Tracks
Pop/Rock	238786
Electronic	41075
Rap	20939
Jazz	17836
Latin	17590
R&B	14335
International	14242
Country	11772
Religious	8814
Reggae	6946
Blues	6836
Vocal	6195
Folk	5865
New Age	4010
Comedy/Spoken	2067
Stage	1614
Easy Listening	1545
Avant-Garde	1014
Classical	556
Childrens	477
Holiday	200

Figure 4.1.1.3.1. MSD all genre labels

4.1.1.4. Spotify

While GTZAN, FMA and MSD are well-known datasets used in MIR, they do not always meet specific requirements for certain projects. For instance, GTZAN is relatively small with only 1000 audio tracks and is criticized for its inaccuracies and duplications. The FMA dataset is larger and provides a broad range of metadata, but its imbalance regarding genres

distribution does not align with this project's needs. MSD is even more expansive, containing a wealth of audio features and metadata for a million songs, but accessing the audio for these tracks is a challenging task. Thus, there may be a need to build a more up-to-date, easily accessible dataset from scratch.

The Spotify API is a rich source of music data, providing access to a wide variety of information such as track details, artist information, album details, playlists and even audio analysis and features, but the most important thing is that it contains genre labels for each track, making it a suitable source for building up a dataset for music genre recognition. Downloading from the Spotify API requires client credentials obtained after making an account on the Spotify Developer platform [21]. The query used to extract audio tracks is the following:

```
results = sp.search(q=f"genre:{genre}", type="track", limit=50, offset=offset)
```

where the search is done according to a musical genre and the result shows the Spotify page starting with the **offset** song and following with other 50 (**limit**) songs. As shown:

```
for track in results["tracks"]["items"]:
    track_id = track["id"]
    track_name = track["name"]
    artist_name = track["artists"][0]["name"]
    preview_url = track["preview_url"]
```

only the track name, its artist and an audio preview of 30 seconds are saved from the query's result. The genres used to quiz the API are the following: blues, classical, country, disco, reggae, metal, hip-hop, jazz, pop, rock. It can be seen that the list is very similar to the one used in GTZAN, in order to make pertinent comparisons between the 2 datasets. The final version of the dataset contains two folders, **Audio_Spotify** with 500 audio tracks per genre and **Audio_Spotify_Test** with 300 tracks per genre and can be found here [22].

Regarding Deep Learning methodologies, for each audio track, 4 melspectrograms were created: the original, unmodified version, one with a random frequency mask, one with a time mask and the last one which has noise added to it (noise with mean 0 and standard deviation 0.3). This process is called Data Augmentation and it is used in order to increase model robustness and to avoid overfitting in neural networks. Thus, the dataset of visual representations reached a total of 32000 files, distributed evenly in 10 folders and can be found here [23]. For the Machine Learning techniques, 2 csv files were created, found in the

original dataset, that contain the mean and variance of each song for several features, including zero-crossing rate, spectral features, harmonics, percussive, mfcc and others. These audio properties were computed, similarly to the GTZAN dataset, for the entire 30 seconds tracks and for smaller 3 seconds windows (in order to increase the amount of data fed to the models).

The complexity of high-dimensionality data (such as the features proposed for analysis) can often pose a significant challenge in classification tasks. To address this, three powerful techniques were employed in this work: Principal Component Analysis (PCA), t-Distributed Stochastic Neighbor Embedding (t-SNE) and Uniform Manifold Approximation and Projection (UMAP). PCA, as a linear method focused on identifying the principal components in which the data varies the most, being helpful in eliminating noise and redundancy from the data. Moving forward with non-linear methods, in order to capture more complex relationships within the data, t-SNE created a probability distribution that denotes the similarity of instances in high-dimensional space and reproduced it in a lower-dimensional space. It is particularly adept at preserving the local structure of data, making it excellent at identifying clusters. UMAP is another non-linear method that tries to maintain both local and global structure of the data, outperforming t-SNE in terms of efficiency and scalability but it sometimes makes the data hard to interpret due to the non-linear transformations.

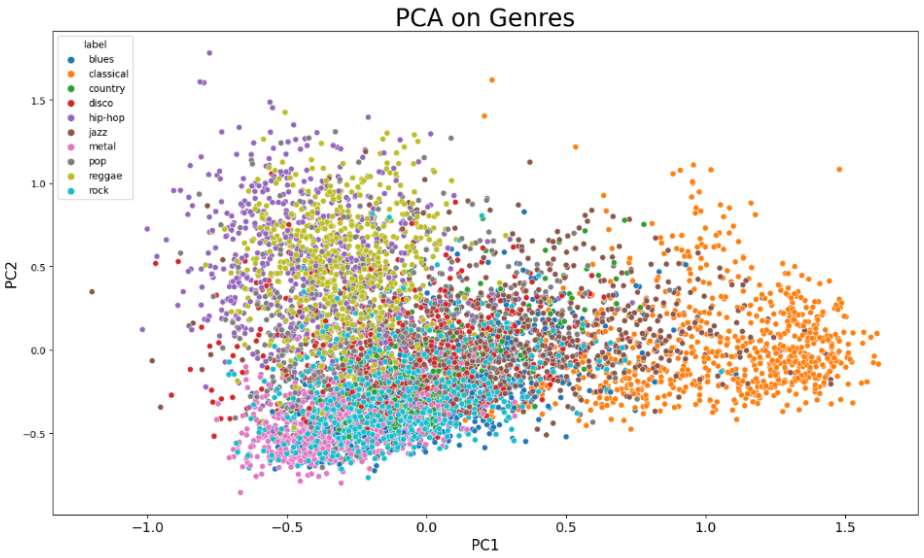


Figure 4.1.1.4.1. Principal Component Analysis

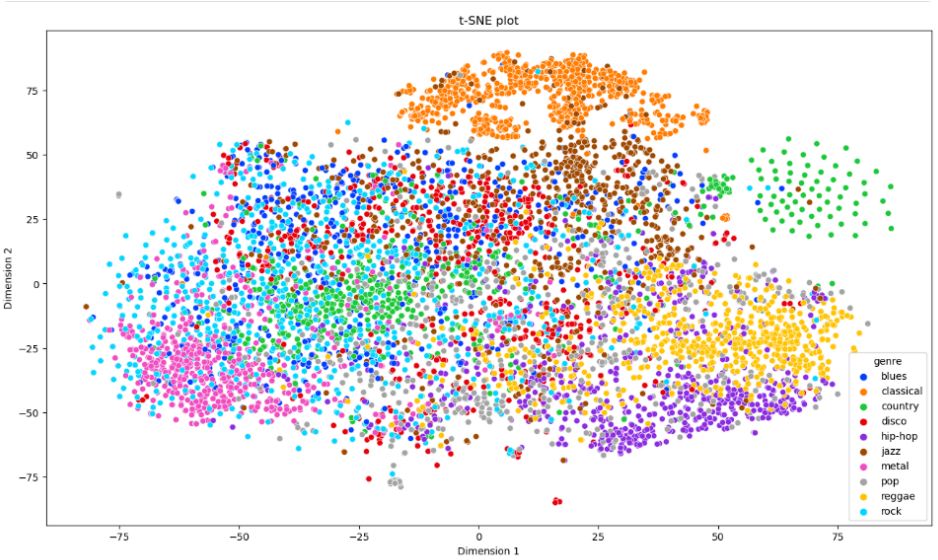


Figure 4.1.1.4.2. t-Distributed Stochastic Neighbor Embedding

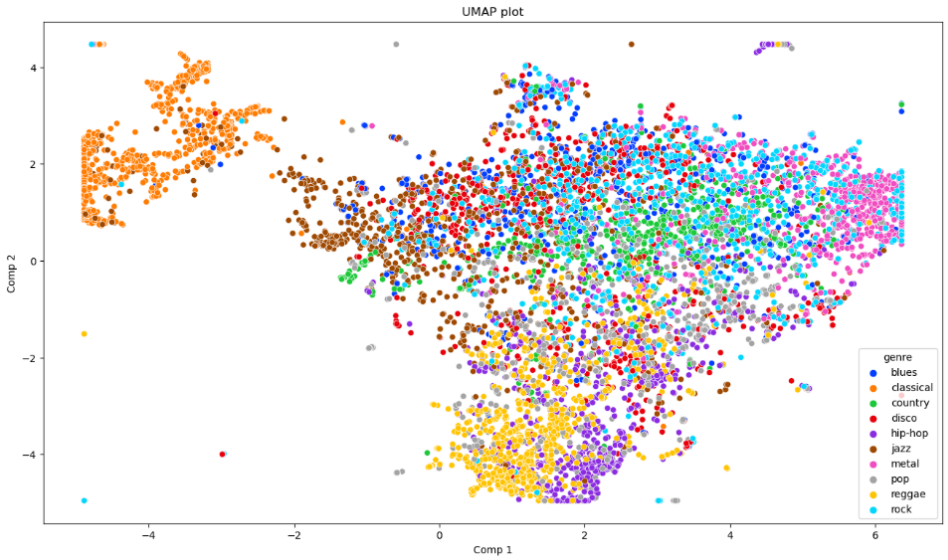


Figure 4.1.1.4.3. Uniform Manifold Approximation and Projection

The following series of plots encapsulates data gleaned from all the audio features extracted from the dataset, enabling a comparative exploration that highlights the uniqueness and the similarity across the chosen music genres.

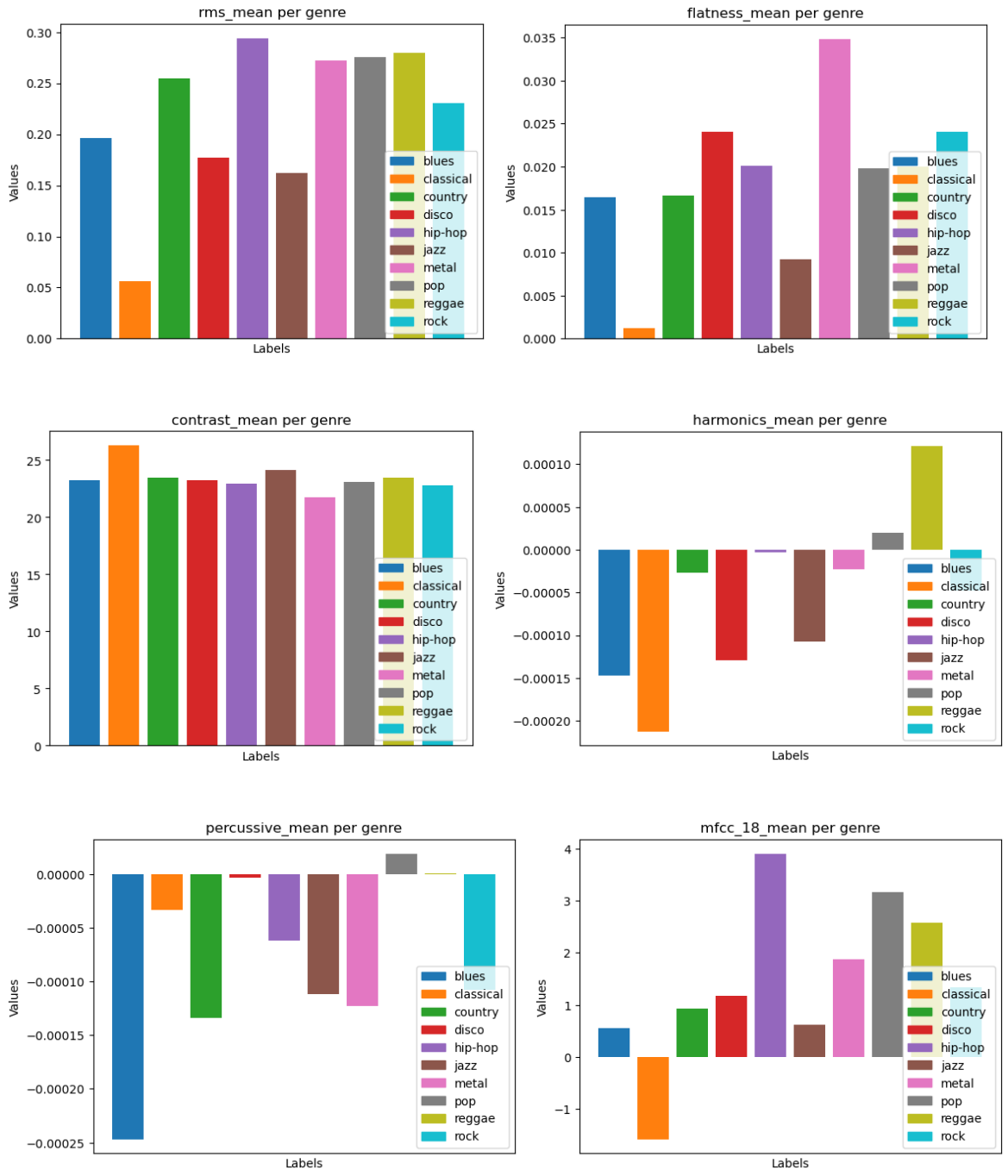


Figure 4.1.1.4.4. Audio features across genres

4.1.2. Machine Learning

This section explores the process of preparing data and makes a comparison between several machine learning algorithms employed in the genre recognition task. One important step to do before the feature selection algorithm and also before splitting the dataset into train and test data is normalization. The final choice was to use MinMaxScaler, which scales and

transforms the features into the range [0,1], in order to keep the values in a positive note and because the distribution of data is unknown (StandardScaler works best when the data follows a Gaussian distribution). While working with the 2 datasets (the one with features extracted from the full-length audio and the one augmented on 3 seconds windows), it has been observed that the models work better with more data, even if the small 3 seconds window is not big enough to encapsulate the full spectrum of changes that can appear in a song (for example Alternative Metal is an audio experience that encapsulates the musicality from heavy metal and other genres not normally associated with metal).

When looking at the feature selection step, the first thing to do is to apply filter methods in order to understand the structure of the data and the impact of individual variables. Each feature is ranked according to a score that reflects the feature's relevance in predicting the target variable.

	Features	Score
15	flux_var	561.284439
21	percussive_var	337.811766
5	spectral_centroid_var	337.614549
2	rms_mean	333.995748
3	rms_var	320.105203
9	rolloff_var	307.314851
10	flatness_mean	293.793871
32	mfcc_5_var	264.789455
8	rolloff_mean	260.595909
25	mfcc_2_mean	255.649183
36	mfcc_7_var	255.413452
14	flux_mean	243.979344
19	harmonics_var	239.811881
38	mfcc_8_var	225.028543
40	mfcc_9_var	219.819015
30	mfcc_4_var	210.930879
0	chroma_stft_mean	207.843380
12	contrast_mean	205.466692
4	spectral_centroid_mean	204.321117
6	spectral_bandwidth_mean	199.745111
34	mfcc_6_var	197.415082
17	zcr_var	177.261642
42	mfcc_10_var	171.643548
44	mfcc_11_var	160.352774
16	zcr_mean	157.450452
23	mfcc_1_mean	156.029405
28	mfcc_3_var	148.958935
48	mfcc_13_var	134.653725
62	mfcc_20_var	115.072779
7	spectral_bandwidth_var	115.005453

Figure 4.1.2.1. Scores of the first 30 selected features

Then, wrapper and embedded methods are applied to a series of ML algorithms to find the best combination of features that best contribute to the accuracy of the model. These methods are integrated into the pipeline that is used for model training, evaluation and hyperparameter tuning.

```
pipeline = Pipeline([('svm', SVC())])

param_grid = {
    'svm__kernel': ['linear', 'rbf', 'poly'],
    'svm__C': [0.1, 1, 10, 100],
    'svm__gamma': [0.1, 1, 10, 100]
}

grid = GridSearchCV(pipeline, param_grid=param_grid, cv=5, n_jobs=-1, scoring='accuracy', verbose=10)
grid.fit(X_train, Y_train)

print(f"Best parameters: {grid.best_params_}")
print(f"Best score: {grid.best_score_}")

model = grid.best_estimator_
```

Figure 4.1.2.2. Interface for analyzing a ML algorithm (SVM)

For all experiments, five-fold cross validation was used on the training data. This was combined with grid search in order to help find the best performing hyperparameters. Cross-validation allows the classifier to train on more data than it would need if both validation and test set had to be used. Grid search ensures that each combination of parameters is chosen and evaluated and eventually reaches the optimum set, the one with the maximum accuracy score.

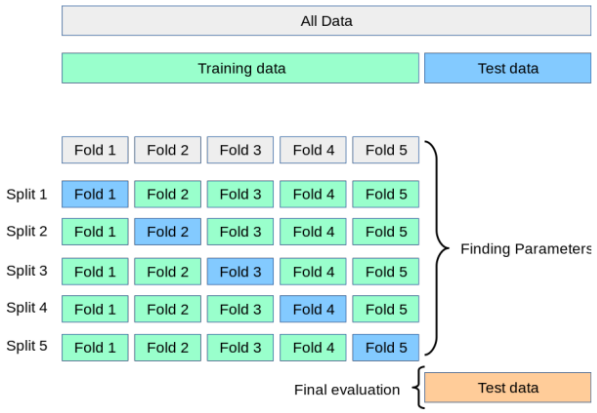


Figure 4.1.2.3. Cross validation

Following this ‘interface’, several algorithms were trained: KNN, SVM, Random Forest, Logistic Regression, Naïve Bayes, XGBoost. Each was evaluated on the test set and a confusion matrix was made in order to visualize the model’s predictions compared to the actual labels of the dataset and to assess which genres are easily recognized and which are similar.

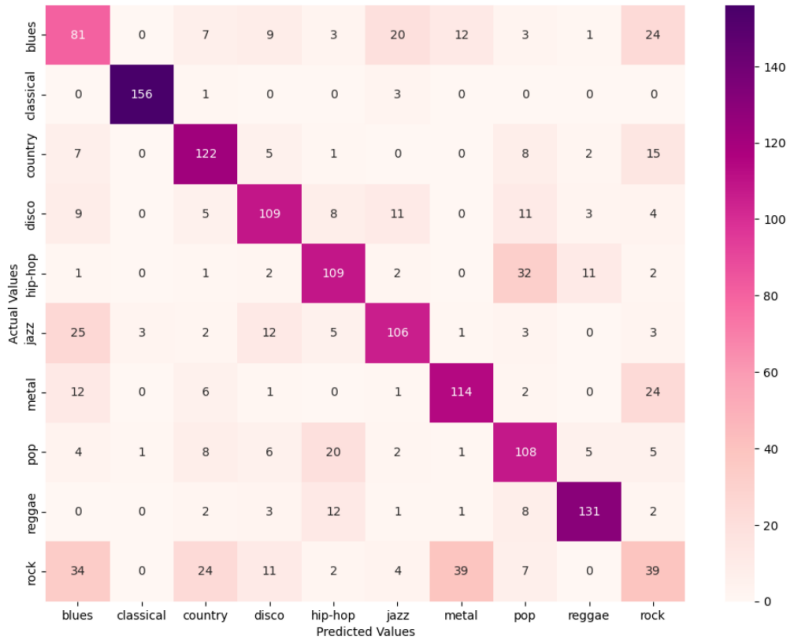


Figure 4.1.2.4. Confusion matrix SVM

The analysis concludes with a report that includes the score for several metrics such as: precision, recall and F1-score. This report provides these scores for each label class individually, allowing to see how well the model performs on each class.

blues	0.47	0.51	0.49	160
classical	0.97	0.97	0.97	160
country	0.69	0.76	0.72	160
disco	0.69	0.68	0.69	160
hip-hop	0.68	0.68	0.68	160
jazz	0.71	0.66	0.68	160
metal	0.68	0.71	0.70	160
pop	0.59	0.68	0.63	160
reggae	0.86	0.82	0.84	160
rock	0.33	0.24	0.28	160
accuracy			0.67	1600
macro avg	0.67	0.67	0.67	1600
weighted avg	0.67	0.67	0.67	1600

Figure 4.1.2.5. Classification report SVM

In the end, multiple models were combined into an ensemble model, in order to improve accuracy, generalize better and to offset the weaknesses of some models with the strengths of others. The models are put together using the voting technique in a soft manner, meaning that the average predicted probabilities for each class gives the output. This final model leads to an improvement in accuracy and in robustness over the individual models.

4.1.3. Deep Learning

After harnessing the capabilities of ML algorithms in the genre recognition task, a quick analysis on the results of the classifiers indicated that more advanced techniques should be implemented for better precision and performance. In the following section, various DL algorithms are explored and their potential to enhance the model's predictive power and adaptability is examined. Given the complexity and richness of music signals, one popular choice of data representation across many research studies is to use Mel Spectrograms. These are visual representations of a sound signal that conveys time, frequency expressed on a Mel scale and amplitude information. While Mel Spectrograms provide a robust and practical representation, they also have some limitations. The process of transforming raw audio into images inevitably results in some loss of information, especially during the conversion from the time domain to the frequency domain and also during Mel scaling. Furthermore, spectrograms typically use a static window size when computing the Fourier Transform, while humans naturally perceive low frequencies over longer windows and high frequencies over shorter windows.

When working with images, there are typically two types of color representations used in neural networks: grayscale and RGB. In grayscale (monochrome), darker shades represent lower amplitudes, while brighter shades indicate higher amplitudes. It has only one channel, meaning they are often preferred for tasks where computational efficiency is a priority, with the trade-off that it does not hold as much information as a colored image. On the other hand, RGB uses 3 channels and has the potential to convey more information, each color channel can represent different aspects or layers of the audio signals. This comes with an increased computational cost, but potentially can enhance the performance. Both interpretations were used for the genre classification task and there were no visible differences in the models' performances, thus the grayscale was used to speed up further experiments. After reading the Mel Spectrograms from the dataset, the final shape of these images is (256, 256, 1).

Scaling data is a fundamental step in preparing the input for AI algorithms, including when using Mel Spectrograms for DL techniques. Doing so, the convergence speed is improved, the problem of exploding or vanishing gradients can be mitigated and the features are now contributing equally to the model's prediction. Before feeding the input to the networks, a new DataFrame is built using scaled images and then it is randomly split into train and test subsets (30% test partition). After this split, each train and test array must be converted to tensorflow tensors in order to be passed as inputs to the framework's neural networks. However, holding the scaled dataset, the train/test split and the tensors into the memory is an intensive task for the computational resources and might lead to the kernel's crashing. A solution to this problem is using an ImageDataGenerator. It is memory efficient as it loads only batches of images when they are needed for training or evaluation. It also provides on-the-fly data augmentation, applying random rotations, shifts, flips and zooms to the images as they are loaded into memory. The process of preparing data for networks is also simplified and automated, saving a significant amount of time and avoiding potential mistakes that can appear in manual data preparation.

Convolutional Neural Networks have been demonstrated to be particularly effective for tasks involving image analysis, thus their use in processing Mel Spectrograms for genre recognition is a natural fit. CNNs are capable of learning hierarchical representations of the input data, from low-level features such as edges and textures to more complex, abstract patterns. In the context of spectrograms, these could correspond to different musical elements such as rhythm, melody or timbre. These networks also have a property called translation invariance, which is crucial with Mel spectrograms, as the same musical pattern can be learnt and identified from it, regardless of its position in time. The network used for the classification task is made up of 3 convolutional layers with 32 or 64 filters and a kernel size of 3 x 3. As kernel initializer, Kaiming ('he_uniform') is used, because in some cases the default Glorot initializer can lead to inconclusive results [24]. For each Convolution there is a Batch Normalization Layer, which scales the inputs for each layer and stabilizes the learning process, reducing the number of training epochs required to train the network. In order to prevent overfitting, which appeared before the first 10 epochs in the training process, Dropout technique was used, which disables a chosen percentage (25%) of the neurons during training, making a more robust model. One of the most popular activation functions used in the hidden layers is the Rectified Linear Unit (ReLU), however the choice was to use the Exponential Linear Unit (ELU) function, in order to avoid the dead neurons and

vanishing gradients which can appear when working with ReLU. Adaptive Momentum Estimation is used as the optimization algorithm for several reasons: it is computed efficiently, it can adapt the learning rate for each weight individually and requires less tuning of the learning rate hyperparameter. In the end, several callbacks were used to monitor the evolution of the training loop. To prevent wasting computational resources and to stop excessive model training, which may lead to overfitting, EarlyStopping is used. When the monitored property (validation loss) has stopped improving, the training is stopped. Also, ReduceLROnPlateau is added to improve the model's learning efficiency: when the learning curve reaches a plateau, failing to converge, the learning rate is further decreased. Because many models perform better at a step before reaching the final number of epochs, ModelCheckpoint saves the weights of each epoch for later use of the model. It has been observed that 15 epochs are enough to reach either a convergence or the point in which the model starts to overfit (training loss tends to become 0 while testing loss increases).

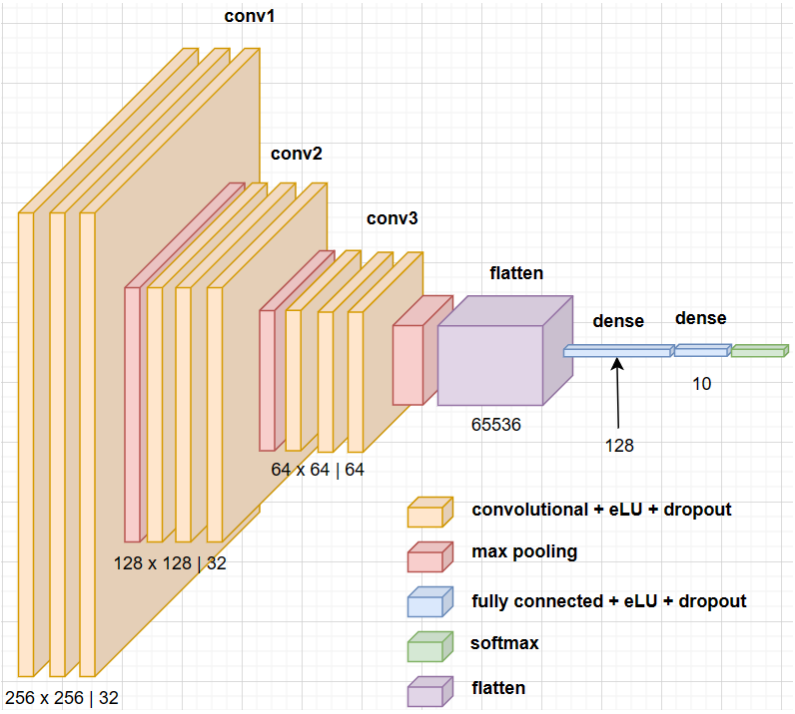


Figure 4.1.3.1. CNN architecture

Another approach for genre recognition is to combine the strengths of Convolutional Neural Networks and Recurrent Neural Networks. RNNs are designed to remember previous inputs in their hidden layers, providing a form of memory. It helps to understand the context over time, which is crucial in understanding the temporal structure in a song. While CNNs extract

features and patterns from audio signals, RNNs take these high-level concepts and understand their evolution over time. This model is very similar to the CNN, with a LSTM cell of 128 units added after the flattened input, this cell being capable of learning long-term dependencies.

Transfer learning is a technique where a pre-trained model is repurposed for a similar but more specific task. The idea is to leverage the feature extraction capabilities learnt by models from extensive training on large datasets, like ImageNet. Training these models from scratch would require high computational and temporal cost. The melspectrograms become inputs to these models, which will specialize in isolating the features that make each musical genre unique, after the training phase concludes. MobileNetV2 is a lightweight and efficient model, making it ideal for real-time tasks like on the spot genre prediction in music streaming services. On the other hand, VGG16 is larger and more complex but it is more robust and should be used when the maximum accuracy is sought. On top of the base pre-trained model there are added 2 Dense layers which are trained and specialized in detecting musical genres from spectrograms.

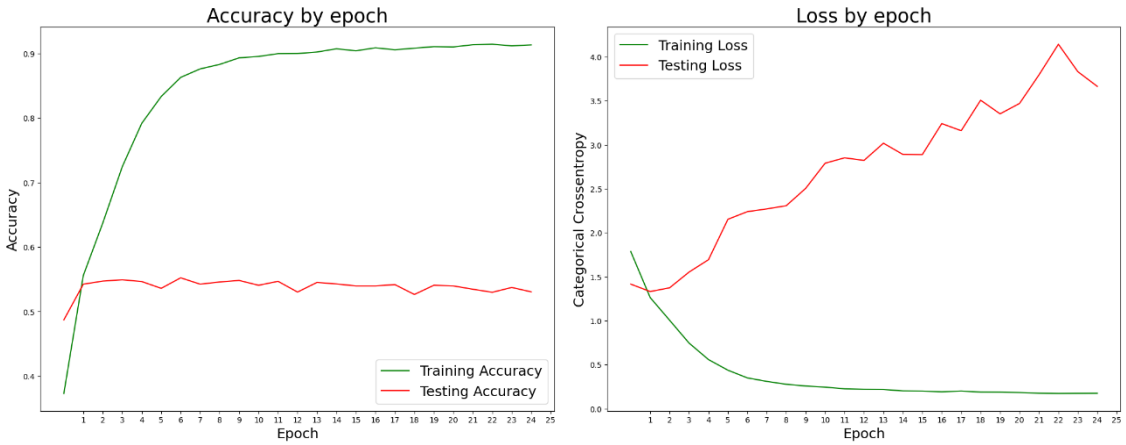


Figure 4.1.3.2. Accuracy and Loss functions per epoch

For each model, the loss and accuracy of each epoch was plotted in order to understand better how the learning process evolves. Furthermore, the confusion matrix shows the precision of the model's predictions and the final scores are computed in the classification report. The experiments were conducted on the spectrograms of both low quality (256x256) and on high quality (512x512), however the results did not improve, only the computational and temporal resources increased, leading to the conclusion that a better resolution does not help in musical

patterns recognition. Preprocessing steps like the creation of melspectrograms might induce a level of abstraction and simplification which can be beneficial in many cases, but sometimes it might neglect potentially meaningful information present in the raw audio data. Another experiment shows the capabilities of deep learning techniques while working directly with the features extracted from the tracks. For this, a network with a simpler architecture was used: 5 Dense blocks with a BatchNormalization layer and a Dropout rate of 25% before the final layer. Despite its simplicity, this network shows promising results as it does not indicate signs of overfitting like the ones that worked with spectrograms.

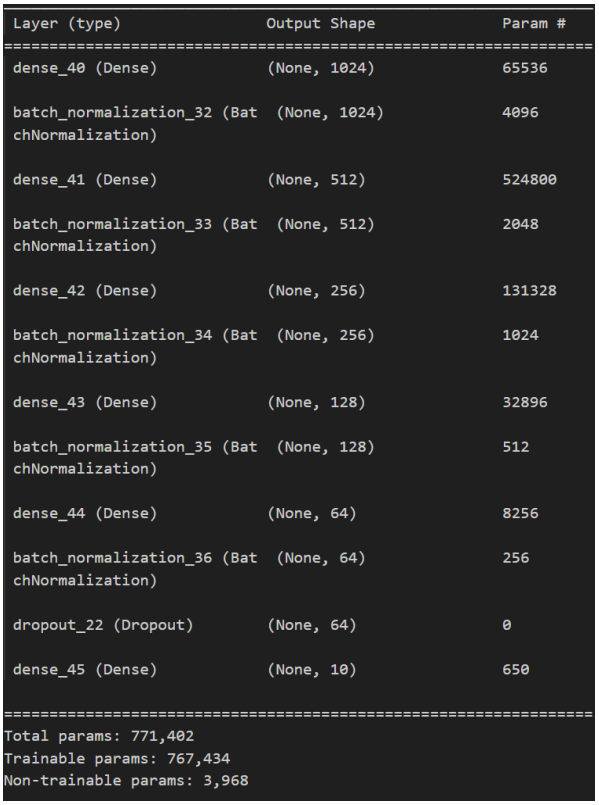


Figure 4.1.3.3. Simple model summary

4.2. Music Generation

4.2.1. Dataset

The first step towards generating music in a particular style requires an adequate dataset that represents the genre well. While studying the previous 10 popular music genres, the one that clearly stood out from the rest was Jazz. Firstly, it is a highly improvisational style, being the perfect playground for an AI model to “improvise” within the given musical parameters. Jazz

is often seen as a deeply emotional genre, musicians using various techniques to express their feelings through instruments, and while AI cannot experience human emotions, it can be trained to understand and replicate some musical patterns that emanate specific grooves and feelings. Such a jazz-inclined dataset can be found here [25].

With raw audio files, the generation of new, coherent melodies from scratch is a particularly challenging task due to the high dimensional, time-series nature of this data. In order to simplify the process of understanding the underlying structure of an audio track, one should consider using other musical notations which in exchange for a simpler, programmatically-friendly interface, gives up the subtle variations in timing and dynamics. Such a practical notation is called MIDI and contains structured, symbolic information about a piece of music. The chosen dataset contains 935 audio files in MIDI format, including several pre-computed properties in csv format. Firstly, the dataset is preprocessed, transforming the musical information into a format similar to text processing. For each audio track the notes are extracted and outputted in their musical notation (F2), also the chords are extracted as sequences of integer pitch classes, all notes of the chord being transposed to a single octave. Since this process requires a considerable amount of time to complete, the preprocessed notes are saved as text in a file for further usage. The sequences of notes and chords are then prepared for the model as follows: each sequence of 100 notes represents the input for the next note in the sequence of all notes. The model will learn to predict a new pitch for a musical pattern given as input and will shift the sequence and repeat this process until the desired length of the audio track is reached.

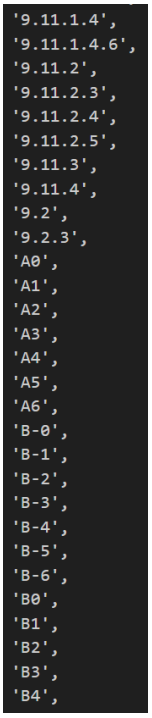
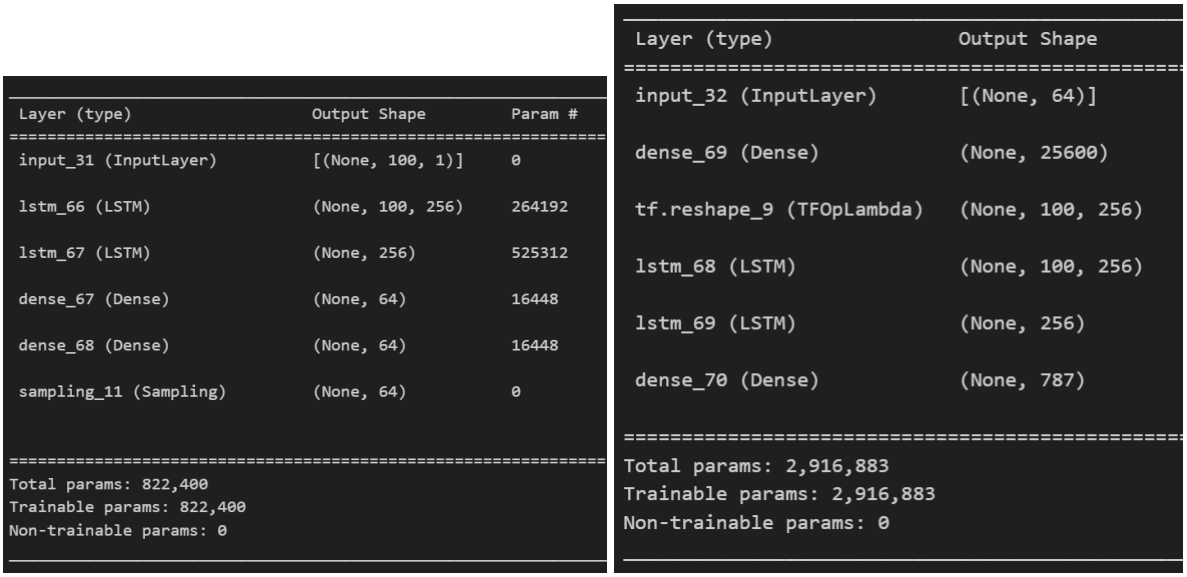


Figure 4.2.1.1. Unique chords and notes from the dataset

4.2.2. Model

The model used for generating new, pleasant Jazz music is a generative model called Variational Autoencoder (VAE), which is composed of 2 primary components: an encoder and a decoder. The goal is to ensure that the reconstructed input is as close as possible to the original input, while also ensuring that the latent space has certain properties (for example that it follows a standard normal distribution). The encoder and the decoder are given as parameters to the network, as they are defined separately. The encoder consists of 2 LSTM layers, each with 256 units. This is followed by 2 Dense layers that output a mean and a log variance, and have the dimension of the latent space, which is 64. These variables represent the parameters of the Gaussian distribution that the encoder learns to map the data to. The sampling layer that is applied afterwards uses the previously mentioned parameters to generate a latent vector, which is the encoded representation of the input. This layer uses the mean and the variance to sample from the distribution defined by these parameters. The purpose of it is to introduce randomness in the encoding process, which ensures that points that are close in the input space are also close in the latent space. This makes the latent space continuous, which is useful for the generation of new data points. The decoder also consists of 2 LSTM layers of 256 units, but the input is expected to have the dimension of the latent

space. Before entering the LSTM cells, the input is fed to a Dense layer that upscales the latent vector to a larger representation. Afterwards, the vector is passed to a Dense layer that outputs a probability (softmax activation) for each unique chord or note in the vocabulary.



Layer (type)	Output Shape	Param #
input_31 (InputLayer)	[(None, 100, 1)]	0
lstm_66 (LSTM)	(None, 100, 256)	264192
lstm_67 (LSTM)	(None, 256)	525312
dense_67 (Dense)	(None, 64)	16448
dense_68 (Dense)	(None, 64)	16448
sampling_11 (Sampling)	(None, 64)	0

Total params: 822,400
Trainable params: 822,400
Non-trainable params: 0

Layer (type)	Output Shape
input_32 (InputLayer)	[(None, 64)]
dense_69 (Dense)	(None, 25600)
tf.reshape_9 (TFOpLambda)	(None, 100, 256)
lstm_68 (LSTM)	(None, 100, 256)
lstm_69 (LSTM)	(None, 256)
dense_70 (Dense)	(None, 787)

Total params: 2,916,883
Trainable params: 2,916,883
Non-trainable params: 0

Figure 4.2.2.1. Encoder and decoder for VAE

The metrics tracked during the training process are the reconstruction loss, Kullback-Leibler (KL) loss and the total loss. These losses are computed manually, as Keras does not yet support the VAE pre-computed model. Training a VAE can take considerable time, for a bunch of reasons: musical data is often high-dimensional and complex, the model is also complex, including LSTM and Dense layers, the process involves multiple stages as it must learn both to encode and to decode the representations accurately. After the training process concludes, the model can be used to make predictions in the following way. From the input sequence given as training input to the model, select a random shorter sequence. This selected sequence is fed as input and receives a note as output. For 100 iterations, the input sequence is shifted, adding the predicted note to its end and removing the start note, receiving another predicted note. At the end of the loop, there is a sequence of 100 generated notes which can be transformed to a midi file and played as a new, AI-composed song.

```
Epoch 1/5
4699/4699 [==============================] - 53397s 11s/step - loss: 4.8321 - reconstruction_loss: 4.8198 - kl_loss: 2.1082e-05
Epoch 2/5
4699/4699 [==============================] - 17760s 4s/step - loss: 4.8084 - reconstruction_loss: 4.8081 - kl_loss: 6.8637e-06
Epoch 3/5
4699/4699 [==============================] - 20140s 4s/step - loss: 4.8044 - reconstruction_loss: 4.8053 - kl_loss: 2.7446e-06
Epoch 4/5
4699/4699 [==============================] - 50350s 11s/step - loss: 4.8073 - reconstruction_loss: 4.8039 - kl_loss: 7.2207e-07
Epoch 5/5
4699/4699 [==============================] - 18632s 4s/step - loss: 4.8027 - reconstruction_loss: 4.8030 - kl_loss: 2.7516e-06
```

Figure 4.2.2.2. VAE training phase

Another approach, inspired from [26], is to combine the VAE with the strengths of Transformers, which are generally thought of as the potent successors of RNNs. The primary advantage of Transformers over LSTMs lies in the attention mechanism, which allows them to better capture long-term dependencies in the data. This is very beneficial in music generation, where rhythm, harmony and other musical attributes that define a genre are often spanned across longer sequences. Research in various areas of sequence data processing, including natural language processing and music generation, has also shown that Transformers models often outperform LSTM-based models in terms of performance and quality. Concerning pre and post processing steps required by the model in order to work with appropriate data, they are similar in both cases (for the simple and augmented VAEs). Consistent differences appear inside the model, as the LSTM cells inside the encoder and the decoder are replaced with Transformer blocks. The encoder follows the standard transformer architecture, with multi-head self-attention and position-wise feed-forward networks, as for regularization there are LayerNormalization and Dropout Layers. The decoder takes sampled points in the latent space as input, applies the transformer encoding and outputs the reconstructed notes through a Time Distributed layer.

5. Evaluation

This chapter displays the results in the form of highest scores achieved for each experiment, as well as confusion matrices to help visualize which genres are most confusing. However, the results are only close estimates and not entirely honed to perfection, mostly because of the lack of enough computational power. These experiments were conducted both on a laptop with Intel i7 (no GPU) and on the cloud Kaggle platform.

5.1. Music Genre Recognition

The subtleties between genres seem to still pose a great challenge when it comes to correctly separating them, showing that even if a song mostly encapsulates one genre according to the musical theory, it can still show signs of playing sequences that resemble other genres. The performance varies with each classification algorithm and shows a clear limitation when it comes to the visual interpretation of the sound.

Regarding Machine Learning techniques, the following table shows the best parameters chosen after running a Grid Search algorithm with 5-cross-validation. For each model, the average accuracy across genres is printed. It seems that the lowest score is acquired by the simple Naive Bayes, while the maximum score is given to a distance-based algorithm like KNN. For this model, the choice of the neighbors hyperparameter for the best accuracy is 1, however, it is a sensitive choice as the decision boundaries become unstable, leading to poor generalization on new data. Even if the hyperparameters are alike, there is a clear gap between the accuracy interval resulted on the GTZAN dataset (50 - 92%) and on the Spotify generated dataset (45 - 77%), with the models keeping the same trend (the rankings of each model tend to be the same). One possible motivation behind this difference may be the fact that the GTZAN dataset is biased towards genre classification [19]. Perhaps another reason could be the weakness of the Spotify API in what concerns the resulting list of songs after a genre-specific query (further musical analysis is required to assess this affirmation). Further experiments showed another interesting result: a Stacking Classifier with all the low-level models and a Logistic Regression as meta-classifier has a great potential in separating genres, reaching a maximum accuracy score on both datasets: 92% on GTZAN and 79% on Spotify.

	KNN	SVM	Random Forest	Logistic Regression	Naive Bayes	XGBoost	Voting ensemble	Stacking ensemble
GTZAN - params	n_neighbours: 1	C: 100 gamma: 1 kernel: rbf	max_depth: None min_samples_split: 2 n_estimators: 200	C: 0.0001 penalty: none solver: sag	var_smoothing: 1e-06	learning_rate: 0.1 max_depth: 7 n_estimators: 200	default	default
GTZAN accuracy	92%	91%	87%	72%	50%	87%	88%	92%
Spotify - params	n_neighbours: 1	C: 10 gamma: 1 kernel: rbf	max_depth: None min_samples_split: 2 n_estimators: 250	C: 0.0001 penalty: none solver: newton-cg	var_smoothing: 1e-09	learning_rate: 0.1 max_depth: 7 n_estimators: 200	default	default
Spotify - accuracy	77%	71%	65%	59%	45%	67%	67%	79%

Figure 5.1.1. ML algorithms scores

Across all experiments with the Spotify generated dataset, it can easily be seen that rock is the most misclassified genre among all the classes, being predominantly confused with metal. One main cause might be the fact that there is a considerable amount of overlap and fluidity between the 2 genres. Additionally, there are a lot of sub-genres (hard rock, heavy metal, progressive rock) where the boundaries become even blurrier. The second most misclassified genre is blues, as it is known to share certain features with other genres like jazz, rock and country, just as is shown in the confusion matrices.

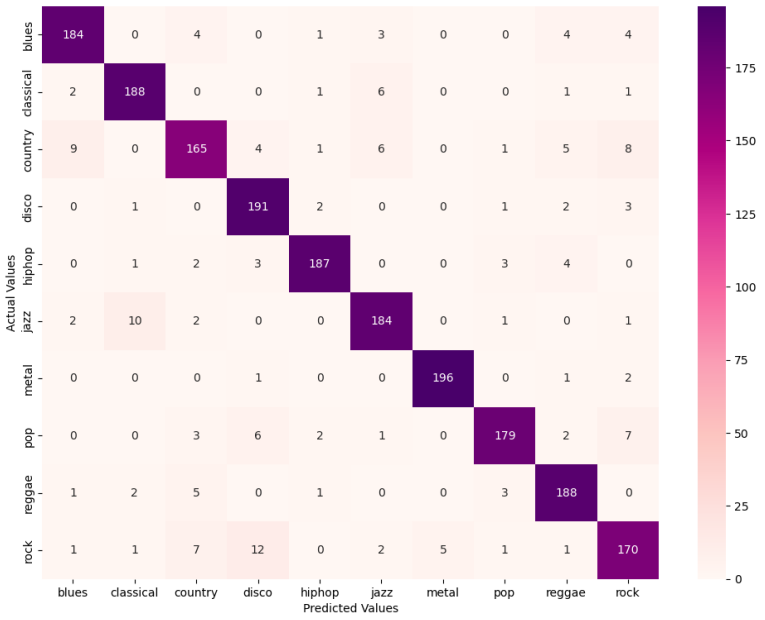


Figure 5.1.2. KNN confusion matrix with maximum accuracy on GTZAN

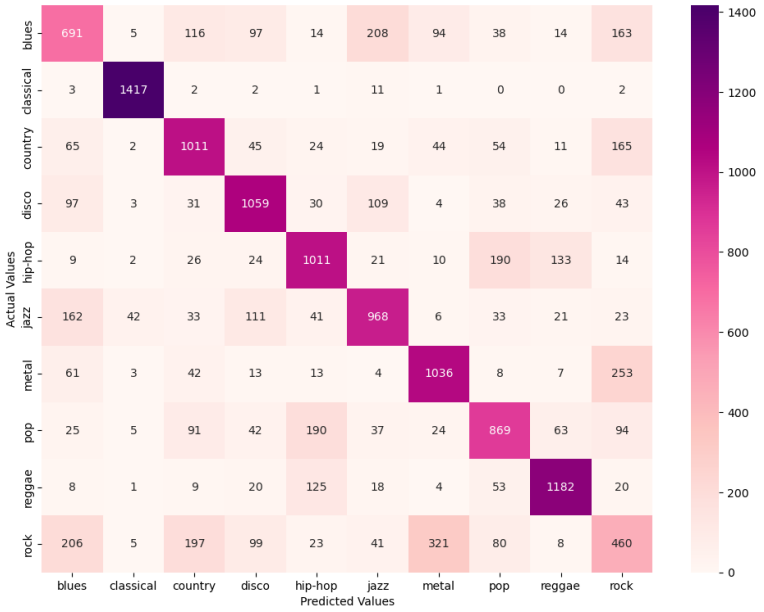


Figure 5.1.3. XGBoost confusion matrix on Spotify

Deep Learning experiments start with a comparison between the ML algorithms and a simple neural network. According to the model's results after 50 training epochs, it reaches 89% accuracy on the GTZAN dataset, which is better than 5 of the ML techniques tried before, and 70% accuracy on the Spotify dataset, with the same conclusion. Both models tend to slightly overfit and reach an accuracy plateau within the first 20 epochs.

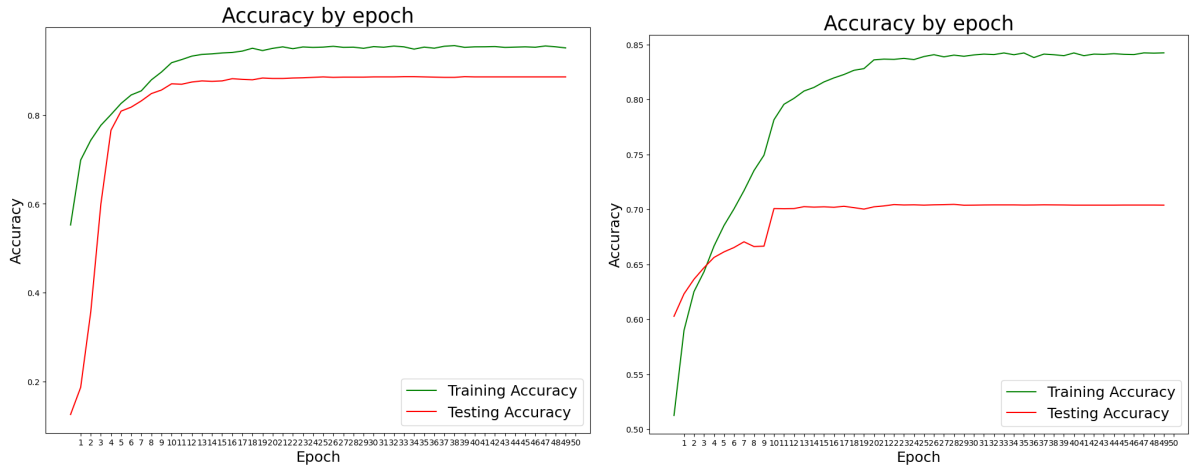


Figure 5.1.4. Accuracy spanned across 50 epochs (GTZAN left - Spotify right)

The experiments continue showcasing the results of classifying audio tracks into genres based on their respective melspectrograms. As previously mentioned, the scores tend to look better for the GTZAN dataset, even in the case of the images representations. The best performing DL algorithm is the one used to classify the features from csv files, which results in a lack of critical genre properties in the melspectrogram. The overtrained CNN network has an accuracy equivalent to the ResNet, and close to the MobileNet, which leads to the conclusion that it can be further improved in order to achieve better results. Due to the small amount of data, all these networks seem to be trained quite fast (no more than 130 seconds per step, even in the case of transfer learning).

GTZAN	5-block Dense	3-block CNN	ResNetV2	MobileNetV2
Accuracy train	92%	99%	94%	95%
Accuracy test	89%	70%	70%	76%
Loss train	0.26	0.01	0.32	0.24
Loss test	0.37	1.15	0.93	0.82
Training epochs	50	100	30	100
Seconds per epoch	1	40	130	50

Figure 5.1.5. Scores for DL algorithms - GTZAN

Even if the scores decrease in case of the Spotify dataset, the results are clearly much better than the random choice (10%). The best model remains the one that classifies based on features, followed by transfer learning (MobileNet). The combined network of CNN and RNN still remains around the same score as the Convolutional Network, even if it has some room from improvement (has not reached overfitting yet). Due to the high amount of data in this augmented dataset, the computational and temporal power needed for training is also greatly increased, that is why the networks are trained for only a limited amount of epochs. Unfortunately, better models like Vision Transformers and VGG16 can not be taken into account as the memory and the time (9 hours / epoch) needed for training are inaccesible.

Spotify	5-block Dense	3-block CNN	MobileNetV2	CNN + RNN
Accuracy train	84%	91%	94%	51%
Accuracy test	70%	56%	59%	50%
Loss train	0.46	0.27	0.12	1.37
Loss test	0.83	1.35	1.48	1.4
Training epochs	50	15	25	10
Seconds per epoch	7	340	400	1960

Figure 5.1.6. Scores for DL algorithms - Spotify

The features importance is computed using SHAP (SHapley Additive exPlanations). This method is based on cooperative game theory and it shows the features that contribute the most to the classification into genres, and also how much each feature contributes to specific genres identification.

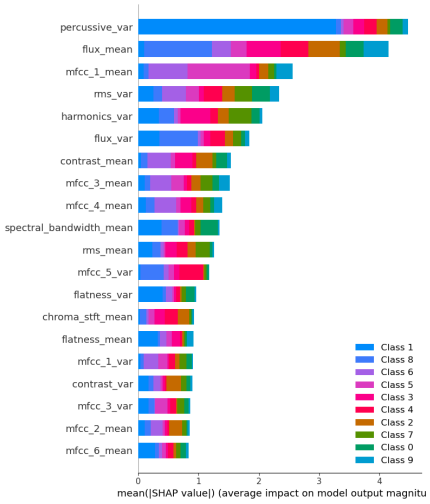


Figure 5.1.7. Features importance and contribution to the final classes

Rather than being uniquely integrated into one genre, a single piece of music often transcends boundaries, encapsulating elements from several genres. This phenomenon can be attributed to the artist's diverse inspirations or the evolution of music over time. Consequently, assigning a singular genre to a song is not always accurate, nor does it encapsulate the full spectrum of its musicality. That is the reason why the model will output several percentages showing how many elements from each genre one song contains. The process involves separating the audio track into windows of 3 seconds each, from which several features are extracted and genre membership percentages are computed and summed.

```
Probability for blues: 2.32%
Probability for classical: 13.02%
Probability for country: 3.81%
Probability for disco: 2.77%
Probability for hip-hop: 30.00%
Probability for jazz: 4.42%
Probability for metal: 2.98%
Probability for pop: 27.46%
Probability for reggae: 10.87%
Probability for rock: 2.33%
```

Figure 5.1.8. Genre identification in *Dance Monkey - Tones and I*

In reference to the findings detailed in the reports of the music genre classification models, it appears that there are mixed results in the models' performances. There are instances in which the classification models demonstrate remarkable precision, successfully categorizing musical genres with high accuracy, as shown when passing *Alla Turca - Mozart* through the model and receiving great results on the correct label. However, there are also highlighted instances where the models face confusion, struggling to distinguish and correctly classify genres. This disparity underscores the complexity and nuances inherent in the task of music genre classification, shown in the incorrect classification of a rock song, *Natural - Imagine Dragons*.

```
Probability for blues: 1.77%
Probability for classical: 73.68%
Probability for country: 1.52%
Probability for disco: 1.59%
Probability for hip-hop: 3.52%
Probability for jazz: 3.07%
Probability for metal: 1.47%
Probability for pop: 1.78%
Probability for reggae: 10.04%
Probability for rock: 1.56%
```

```
Probability for blues: 2.58%
Probability for classical: 3.52%
Probability for country: 2.48%
Probability for disco: 2.87%
Probability for hip-hop: 26.68%
Probability for jazz: 45.19%
Probability for metal: 2.42%
Probability for pop: 5.38%
Probability for reggae: 6.37%
Probability for rock: 2.50%
```

Figure 5.1.9. Opposed identification: *Alla Turca - Mozart* vs *Natural - Imagine Dragons*

5.2. Music Generation

Evaluating AI-generated music is a challenging task, coming from the fact that musical appreciation is highly subjective and nuanced. In what concerns computational evaluation,

music is a complex, multi-dimensional medium that encompasses many aspects like rhythm, timbre, dynamics, which cannot be captured in a single quantitative metric. Some researchers have built specific metrics that measure the re-creation fidelity of the generated music like chroma similarity, grooving similarity or instrumentation similarity, all of these found in [26]. However, the work in this paper is limited to the classical metrics for Variational Autoencoders: the reconstruction, KL-divergence and total loss. The reconstruction loss models how well the model can recreate the input data after encoding it into the latent space and decoding it back into the original space. A high reconstruction accuracy means that the generated music closely resembles the training data in terms of many musical attributes like rhythm, note sequences. Improving only this loss could easily lead to overfitting and a generated music that lacks creativity. This is where the KL-divergence intervenes. This metric measures the difference between the learnt latent distribution and the normal distribution. Now, the model is encouraged to maintain a structured latent space that allows for more diversity and creativity. The total loss is the weighted sum of these 2 losses and tries to keep a balance between the 2: focusing on the reconstruction leads to overfitting and lack of creativity, while focusing on the KL leads to diverse but low-quality output.

As it can be easily seen from the training steps, this process is arduous, working with the notes directly from the memory and with the notes being memorized in a generator. There is also a similarity regarding the values for the loss function in the VAE model, where the total loss drops from 10 to around 4.8 during the first epoch, and it oscillates around this value for the next epochs, and in the TransformerVAE model, where the total loss jumps from 7 to 4.8 in the first epoch, then it starts to slowly decrease again. The KL-loss in the latter model is however a few times higher than the former, perhaps showing that this model is learning a better representation of the data. The time needed for one epoch to complete in the training phase is also relatively large, reaching approximately 10000 seconds.

```
Epoch 1/2
4698/4698 [==============================] - 10047s 2s/step - loss: 4.8186 - reconstruction_loss: 4.7541 - kl_loss: 0.0645
Epoch 2/2
4698/4698 [==============================] - 10008s 2s/step - loss: 4.7785 - reconstruction_loss: 4.7263 - kl_loss: 0.0522
```

Figure 5.1.9. TransformerVAE training phase

Even if the songs generated by these models fail to create a convincing illusion of being human-produced to people, they maintain the characteristics inherent to the training dataset

genre (in this case Jazz). Despite the potentially identifiable artificiality of the produced audio, the compositional structure and the unique elements of the intended genre are effectively captured and reproduced, and they can be recognized by other machines, specifically the models involved in genre classification.

```
Probability for blues: 8.52%
Probability for classical: 9.94%
Probability for country: 8.52%
Probability for disco: 8.52%
Probability for hip-hop: 8.66%
Probability for jazz: 17.69%
Probability for metal: 8.52%
Probability for pop: 8.53%
Probability for reggae: 12.56%
Probability for rock: 8.52%
```

Figure 5.1.10. Generated song passed through DL model - still Jazz

An interesting phenomenon appears when training both VAEs, due to the nature of their underlying probabilistic models. They can sometimes predict a sequence of repetitive notes when performing the maximum likelihood estimate, a sequence that changes along with the initial seed. The causes for this "laziness" of the models might be the lack of diversity in the dataset, the overfitting of the model into the same high occurring note or the poor choice of architecture and hyperparameters. However, when looking at the probability distribution of the predictions, there were several other notes in the same exponential range, indicating that all the other notes could be a fit good enough for the prediction. To mitigate this issue, a "temperature" parameter was introduced, replacing the "argmax" function that selected the final prediction. This parameter effectively controls the sharpness of the probability distribution, at high values outputting a more uniform distribution that encourages diversity, and at low values outputting a more peaked distribution that makes the model more confident in its reconstructed predictions. By carefully tuning this hyperparameter, more pleasing and diverse results can be achieved. A few samples generated by both of these models can be downloaded from the notebooks' repository[4].

[4] https://github.com/pricoptudor/Licenta_workspace

6. Conclusion and future work

This thesis presented an extended work on the complex tasks of music genre recognition and music generation. Several machine learning and deep learning tools were used to sort the music into different genres, leading to interesting results. Starting from understanding audio signals by extracting features from it, and visualizing it in a compact form, the study shows promising models, and the importance of the data used for the task to be carefully analyzed and selected. Regarding the generation of new music, the models were picked in order to mimic the way humans produce music, and to add a little bit of creativity to it. The complete set of analyses, methodologies and results described in the paper have been documented and are available in a structured collection of Jupyter notebooks. This collection can be accessed at the associated Github repository[5], providing an in-depth overview at the research and facilitating replication or further exploration of the work.

In the future, the music generation task could be further refined, either by removing the dependency of the dataset, as for now the generated music is related to the input data, by creating unique architectures for models that target specific musical attributes, or by leveraging the so called “one-shot transfer”, in which the musical style and content is extract from a single audio track.

[5] https://github.com/pricoptudor/Licenta_workspace

7. Bibliography

[1] Derek A. Huang, Arianna A. Serafini, Eli J. Pugh, “Music Genre Classification” - https://github.com/derekahuang/Music-Classification

[2] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alex Radford, Ilya Sutskever, “Jukebox: A Generative Model for Music” - https://arxiv.org/pdf/2005.00341.pdf

[3] Ondrej Cifka, Umut Simsekli, Gael Richard, “Groove2Groove: One-Shot Music Style Transfer with Supervision from Synthetic Data” - https://hal.science/hal-02923548v2/document

[4] IBM, “What is exploratory data analysis?” - https://www.ibm.com/topics/exploratory-data-analysis

[5] J. L. Zhang, X. L. Huang, L. F. Yang, Y. Xu, S. T. Sun, “Feature selection and feature learning in arousal dimension of music emotion by using shrinkage methods”

[6] Last.fm - https://www.last.fm/home

[7] J. H. Lee, J. S. Downie, “Survey of music information needs, uses, and seeking behaviours: preliminary findings”

[8] Every noise at Once - https://everynoise.com/engenremap.html

[9] Chroma Short-Time Fourier Transform - https://www.audiolabs-erlangen.de/content/05-fau/professor/00-mueller/02-teaching/2016s_apl/LabCourse_STFT.pdf

[10] D. N. Jiang, L. Lu, H. J. Zhang, J. H. Tao, L. H. Cai, “Music type classification by spectral contrast feature”

[11] Understanding the Mel Spectrogram - https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53

[12] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N, Gomez, Lukasz Kaiser, Illia Polosukhin, “Attention Is All You Need” - https://arxiv.org/pdf/1706.03762.pdf

[13] Diederik P. Kingma, Max Welling, “Auto-Encoding Variational Bayes” - https://arxiv.org/pdf/1312.6114.pdf

[14] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio, “Generative Adversarial Nets” - https://arxiv.org/pdf/1406.2661v1.pdf

[15] Michael Defferrard, Kirell Benzi, Pierre Vandergheynst, Xavier Bresson, “FMA: A Dataset for Music Analysis” - https://arxiv.org/pdf/1612.01840.pdf

[16] GTZAN dataset on Kaggle - https://www.kaggle.com/datasets/andradaolteanu/gtzan-dataset-music-genre-classification

[17] FMA dataset - https://github.com/mdeff/fma

[18] MSD dataset - http://millionsongdataset.com/pages/getting-dataset/

[19] Bob L. Sturm, “The GTZAN dataset: Its contents, its faults, their effects on evaluation, and its future use” - https://arxiv.org/pdf/1306.1461.pdf

[20] FMA faults ticket - https://github.com/mdeff/fma/issues/8

[21] Spotify Developer Platform - https://developer.spotify.com/

[22] Spotify features dataset - https://www.kaggle.com/datasets/pricoptudor/spotify-dataset

[23] Spotify melspectrograms dataset - https://www.kaggle.com/datasets/pricoptudor/spotify-image-dataset

[24] Why Keras CNNs are broken - https://towardsdatascience.com/why-default-cnn-are-broken-in-keras-and-how-to-fix-them-ce295e5e5f2

[25] Jazz MIDI dataset on Kaggle - https://www.kaggle.com/datasets/saikayala/jazz-ml-ready-midi?select=Jazz-midi.csv

[26] Shih-Lun Wu, Yi-Hsuan Yang, “MuseMorphose: Full-Song and Fine-Grained Piano Music Style Transfer with One Transformer VAE” - https://arxiv.org/pdf/2105.04090.pdf

[27] Daniel Kostrzewa, Piotr Kaminski, Robert Brzeski, “Music Genre Classification: Looking for the Perfect Network”

[28] Instructional materials for Music Information Retrieval - https://musicinformationretrieval.com/

[29] Introduction to Correlation Concepts, Matrix and Heatmap - https://vitalflux.com/correlation-heatmap-with-seaborn-pandas/

[30] Explained Principal Component Analysis - https://builtin.com/data-science/step-step-explanation-principal-component-analysis

[31] Explained t-SNE - https://towardsdatascience.com/t-sne-clearly-explained-d84c537f53a

[32] Explained UMAP - https://towardsdatascience.com/how-exactly-umap-works-13e3040e1668

[33] Audio features for analysis - https://athina-b.medium.com/audio-signal-feature-extraction-for-analysis-507861717dc1

[34] Spectral feature extraction for audio analysis - https://analyticsindiamag.com/a-tutorial-on-spectral-feature-extraction-for-audio-analytics/

[35] Audio features extraction - https://rramnauth2220.github.io/blog/posts/code/200525-feature-extraction.html#rolloff

[36] Spectral flatness - https://www.johndcook.com/blog/2016/05/03/spectral-flatness/

[37] Spectral flux - https://www.sciencedirect.com/topics/engineering/spectral-flux

[38] Harmonics - https://electrical-engineering-portal.com/harmonics-what-are-they-what-do-they-do

[39] Tempo - https://maelfabien.github.io/machinelearning/Speech10/#tempogram

[40] MFCC technique - https://www.analyticsvidhya.com/blog/2021/06/mfcc-technique-for-speech-recognition/

[41] What is KNN? - https://www.ibm.com/topics/knn

[42] All you need to know about SVM - https://towardsdatascience.com/support-vector-machines-svm-c9ef22815589

[43] What is Random Forest? - https://www.ibm.com/topics/random-forest

[44] What is Logistic Regression? - https://www.ibm.com/topics/logistic-regression

[45] What is Naive Bayes? - https://www.ibm.com/topics/naive-bayes

[46] Guide to XGBoost - https://towardsdatascience.com/a-beginners-guide-to-xgboost-87f5d4c30ed7

[47] Explained Convolutional Neural Networks - https://towardsdatascience.com/convolutional-neural-networks-explained-9cc5188c4939

[48] Explained Recurrent Neural Networks - https://towardsdatascience.com/recurrent-neural-networks-explained-with-a-real-life-example-and-python-code-e8403a45f5de

[49] Understanding Transfer Learning for Deep Learning - https://www.analyticsvidhya.com/blog/2021/10/understanding-transfer-learning-for-deep-learning/

[50] Essentials of Reinforcement Learning - https://towardsdatascience.com/reinforcement-learning-101-e24b50e1d292

[51] Beginner's guide to MIDI - https://musicianshq.com/a-beginners-guide-to-midi/

[52] Introduction on LSTM networks - https://towardsdatascience.com/lstm-networks-a-detailed-explanation-8fae6aefc7f9