UNIVERSITAT POLITÈCNICA DE VALÈNCIA

Escola Tècnica Superior d'Enginyeria Informàtica

Escola Tècnica Superior d'Enginyeria Informàtica

Universitat Politècnica de València

Finding music's latent dimensions with Variational Autoencoders

Trabajo Fin de Grado

Grado en Ingeniería Informática

Autor: Joan Coll Alonso

Tutor: Jon Ander Gómez Adrian

Tutor externo: Georg Groh

2022

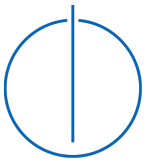
SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Finding Music's Latent Dimensions with Variational Autoencoders

Joan Coll Alonso



SCHOOL OF COMPUTATION, INFORMATION AND TECHNOLOGY - INFORMATICS

TECHNISCHE UNIVERSITÄT MÜNCHEN

Bachelor's Thesis in Informatics

Finding Music's Latent Dimensions with Variational Autoencoders

Auffinden der latenten Dimensionen von Musik mit Variational Autoencoders

Author:	Joan Coll Alonso
Supervisor:	Apl. Prof. Dr. Georg Groh
Advisor:	Jon Ander Gómez Adrián, PhD
Submission Date:	August 16, 2023

I confirm that this bachelor's thesis in informatics is my own work and I have documented all sources and material used.

Munich, August 16, 2023 Joan Coll Alonso

Acknowledgments

First of all, I would like to thank both of my supervisors, Professor Groh and Associate Professor Jon Ander Gómez. To Professor Groh I owe the idea and basis of this project and the guidance necessary to carry it out. To Associate Professor Jon Ander Gómez I owe his mentorship, advice and assistance for the past four years.

Secondly, I would like to thank my parents, Salvador and Marina, for all their support through difficult times. For having the necessary patience to deal with me and always wanting the best for me. I would like to thank my brother, Salva, for always putting a smile on my face when I need it the most and for providing motivation and energy when nothing seems to be working. I would like to thank my grandfather Juan for those much needed conversations in between hours of working in the project. I am happy to have such a wonderful family.

I would like to thank in a special way my girlfriend, Tianyu, for putting up with me every day. For providing priceless advice about things that I completely missed. For her relentless affection and her special way of cheering me up, that no one else can do. But most importantly, for being my place where I take shelter during a storm and for the (almost) blind trust she puts in me. I am forever thankful for having you by my side, always.

Finally, I would like to thank myself. For pushing through the hardest times, with the support of those around me. For not giving up once during the past four years, with their up and downs. For being a person I can (most times) coexist with.

I am thankful for the present and I look forward to the future.

Abstract

With the recent advancements in the field of artificial intelligence, multiple applications of it have been developed and studied for music. From genre classification to performance interaction, there are many areas of interest that result from the combination of these two fields.

In this work, we are particularly interested in discovering and analyzing music's latent dimensions making use of the latest technology available. We will introduce the topic, present a thorough literature review, introduce the considered models, present the underlying technology we base our work off in a detailed way, explain our planned methodology and experiments, discuss the obtained results and draw conclusions and possible lines of future work.

Although both VAE and BERT were considered, BERT was deemed as the better option given that music is sequential data. t-SNE was chosen as the dimensionality reduction technique and k-means as the clustering method. BERT provided numerical data for every musical piece analyzed, whose dimensions we reduced with t-SNE and on which we performed clustering with k-means to analyze the resulting representation. We found an optimal value of k for which the different clusters could be explained, thus proving BERT understands music according to some latent parameters.

Kurzfassung

Mit den jüngsten Fortschritten auf dem Gebiet der künstlichen Intelligenz, wurden zahlreiche Anwendungen für die Musik entwickelt und untersucht. Von der Genreklassifizierung bis hin zur Interaktion bei der Aufführung gibt es viele Bereiche von Interesse, die sich aus der Kombination dieser beiden Gebiete ergeben.

In dieser Arbeit sind wir besonders an der Entdeckung und Analyse von die latenten Dimensionen der Musik zu entdecken und zu analysieren, indem wir die neueste verfügbare Technologie nutzen. Wir werden in das Thema einführen, einen gründlichen Literaturüberblick geben, die in Frage kommenden Modelle vorstellen, die zugrundeliegende Technologie, auf die wir unsere Arbeit stützen Wir stellen die unserer Arbeit zugrundeliegende Technologie detailliert vor, erläutern unsere geplante Methodik und unsere Experimente, diskutieren die erzielten Ergebnisse und ziehen Schlussfolgerungen sowie mögliche Ansätze für zukünftige Arbeiten.

Obwohl sowohl VAE als auch BERT in Betracht gezogen wurden, wurde BERT als die bessere Option angesehen da es sich bei Musik um sequenzielle Daten handelt. t-SNE wurde als Dimensionalitätsreduktion und k-means als Clustermethode gewählt. BERT lieferte numerische Daten für jedes analysierte Musikstück, dessen Dimensionen wir mit t-SNE reduzierten und auf Clustering mit k-means durchgeführt haben, um die resultierende Darstellung zu analysieren. Wir fanden einen optimalen Wert von k, für den die verschiedenen Cluster erklärt werden konnten, und bewiesen damit, dass BERT Musik anhand einiger latenter Parameter versteht.

Contents

1 Introduction

Music has been a central aspect of human culture for millennia. All around the world, all kinds of civilizations have developed their own music with different pitches, instruments, rhythms and styles. Whether it was used in a divine and ritualistic setting or for different celebrations, everybody was surrounded by music. In the later centuries, and especially in Western culture, the act of music creation has been surrounded by a mystic and mysterious aura, that has cast some of these qualities into music itself. Only the highly trained are able to really analyze and bring out the characteristics of a music piece. However, in this work, we do not focus on human understanding of music but on computer understanding.

While there are many kinds of music and many different genres, we will generally say that music is a story told with sounds. These sounds can be stored in multiple ways: in a measurement of frequency captured by a microphone or in a symbolic representation. Most of the work done in computer science towards music makes use of the first storage method, while there is not as much research done on symbolic representations. While it is true that an audio representation of a music piece captures more nuances and details about its performance, it is also true that symbolic representations are handled in a less complex way. Symbolic representations allow us to easily obtain features about the piece, eliminate performance variability and simplify comparison between different pieces.

With the surge of Natural Language Processing models in recent years, like GPT from OpenAI or BERT from Google, it is clear that the Artificial Intelligence paradigm is expanding. These models are able to learn not only the meaning of words but also their meaning with context. The context in music is vital. The same note with the same pitch played by the same instrument with the same volume can feel very different in different parts of the piece or just depending on which notes are played around it or at the same time. As the purpose of this work is to analyze and understand music's latent dimensions, models that are able to learn with context seem to be the perfect choice.

1.1 Machine Learning and Music

Machine learning (ML) is a branch of artificial intelligence (AI) that allows computers to learn without being explicitly programmed. Methods like regression, dimensionality reduction, or clustering, or areas like Neural Networks and Deep Learning all fall under ML. ML algorithms can be used to identify patterns in data and make predictions. In the field of music, ML has been used for a variety of tasks; some of these tasks are music generation, music recommendation and music analysis.

These areas of applications can be put together to aid composers and musicians,

generate music similar to already existing ones, improve recommendation algorithms or even interact with a performance. Research in this area can impact different groups, like musicians, producers and even fans. That is why the application of ML in music has a lot of potential in different areas.

1.2 Latent Spaces

Latent spaces are sets of dimensions that are not directly observed but rather are inferred from the data itself. It can be interpreted as the "underlying structure of the data" and has many different uses. Latent spaces are a result of performing dimensionality reduction.

Latent spaces can be used to perform a variety of tasks in machine learning, like classification or clustering. Note that the techniques can be applied without the use of latent spaces, but using them will generally improve the results. This is because latent spaces capture the underlying structure of the data and are able to extract non-trivial features that can be relevant to these tasks.

Given the nature of latent spaces, capturing features that escape the human eye, its application to music is very interesting. Latent space enables the extraction of essential features and patterns from audio signals or symbolic data, such as rhythm, harmony, and melody, without requiring explicit labels or annotations. Moreover, latent space facilitates the creation of continuous representations for musical attributes, enabling smooth interpolations between genres or styles We aim to open new perspectives, find different relations between pieces and enhance comprehension.

2 State of the art

During the development of this thesis, a significant amount of reading and research was undertaken in the field of machine learning applied to the domain of music, with a specific focus on studying latent spaces and the use of Variational Autoencoders (VAE) and transformers in music.

The process of searching and selecting articles was meticulous and rigorous, ensuring the inclusion of relevant and high-quality works in this literature review. Additionally, a thorough review and analysis of each selected article was conducted to gain a deep understanding of the approaches, methods, and outcomes achieved in each of them. This allowed the identification of trends, significant advancements, and areas of opportunity in the field, significantly enriching this work and providing a solid theoretical foundation for the research.

The final result is a comprehensive and rigorous compilation of existing literature, showcasing the knowledge gained. We are convinced that this research will be a valuable contribution to the field, and its findings may prove beneficial for future studies and investigations in this interdisciplinary domain.

This literature review has been subdivided in the following way: machine learning -in its general meaning- applied to music, findings regarding latent spaces in music and finally the usage of VAE -a specific model- in music. The last section is dedicated to the introduction of VAE and transformer models.

2.1 Machine Learning

Machine learning has revolutionized the music industry, playing a pivotal role in music classification by distinguishing and categorizing various genres. Its importance lies in its ability to automate and optimize the process of genre identification, saving time and effort for musicians, producers, and music platforms alike.

Traditionally, genre classification relied on subjective human judgment, leading to inconsistencies and biases. However, machine learning algorithms, through the analysis of massive music datasets, can learn intricate patterns and features that define each genre, leading to more accurate and objective classifications.

Moreover, the applications of machine learning in music extend beyond just genre classification. With the advent of deep learning techniques, music generation and recommendation systems have emerged, allowing for the creation of original compositions based on learned patterns from existing songs. These generative models hold tremendous potential for expanding the boundaries of musical creativity and introducing innovative sounds to the industry.

Considering the significance of this topic, we will now provide an overview of the

prior research conducted, focusing on the most compelling articles that delve into this subject.

Ndou et al. [1] research provides a comparative study of genre classification performance between deep-learning and traditional machine-learning models. Furthermore, it investigates the performance of machine-learning models implemented on three-second duration features, compared to those implemented on thirty-second duration features.

The categories of features utilized for automatic genre classification are introduced, and the Information Gain Ranking algorithm is employed to identify the most influential features for accurate music piece classification. Subsequently, machine-learning models and Convolutional Neural Network (CNN) are trained and tested on ten GTZAN dataset genres.

The GTZAN dataset contains ten music genres, each having 100 audio clips, with each clip lasting 30 seconds in .wav format sampled at 22050Hz, 16-bit.

The k-Nearest Neighbours (kNN) model exhibits the highest classification accuracy of 92.69% with three-second duration input features. The research aims to achieve automatic music genre classification using both deep-learning and traditional machine-learning models. A comprehensive literature review substantiates the capabilities of these classifiers and serves as a benchmark for comparing the research findings.

The findings highlight the superiority of three-second duration input features, which yield higher accuracy and more extensive training data, resulting in improved performance compared to thirty-second duration input features. Notably, the kNN achieves outstanding accuracy with a relatively short training time of 78 milliseconds, outperforming related literature due to the robust feature set.

Moreover, Linear Logistic Regression and Support Vector Machines (SVM) demonstrate noteworthy performances, achieving accuracies of 81.00% and 80.80%, respectively. In contrast, the Convolutional Neural Network (CNN) implementations in this research display relatively lower accuracy, with the most accurate CNN implementation reaching 72.40%.

This study demonstrates the feasibility of automatic music genre classification, with traditional machine-learning models exhibiting superior performance compared to deep-learning approaches. The results pave the way for further advancements in the field of music genre classification, benefiting various music-related applications and industries.

Bahuleyan [2] conducts a comparison of the performance of two classes of models in the challenging task of categorizing music files based on their genre within the domain of music information retrieval (MIR). The first approach involves deep learning, where a Convolutional Neural Network (CNN) model is trained end-to-end to predict the genre label of an audio signal using its spectrogram. The second approach utilizes handcrafted features from both the time domain and frequency domain. Four traditional machine-learning classifiers are trained with these features, and their performance is compared. The features that contribute the most to this classification task are identified using the Information Gain Ranking algorithm. The experiments are conducted on the Audioset dataset, and an AUC value of 0.894 is reported for an ensemble classifier that combines the two proposed approaches.

Four machine-learning classifiers are adopted in this study. Logistic Regression

(LR) is implemented as a one-vs-rest method, where separate binary classifiers are trained, and the class with the highest probability among the classifiers is chosen as the predicted class during test time. Random Forest (RF) is an ensemble learner that combines predictions from a pre-specified number of decision trees using bootstrap aggregation (bagging) and random subsets of features. Gradient Boosting (XGB) is another ensemble classifier that combines weak learners in a sequential manner using forward stagewise additive modeling. Support Vector Machines (SVM) transform the input data into a high-dimensional space using a radial basis function (RBF) kernel for non-linear problems and are also implemented as a one-vs-rest classification task.

The CNN model based on VGG-16, which uses only the spectrogram to predict the music genre, performs the best among the models studied. The transfer learning and fine-tuning settings show no significant difference in performance. The baseline feed-forward neural network that uses the unrolled pixel values from the spectrogram performs poorly, highlighting the significant improvement that CNNs offer for image classification tasks. Among the models using manually crafted features, the Logistic Regression model has the least performance, as expected due to its linear nature. SVMs outperform Random Forests in terms of accuracy, but the XGB version of the Gradient Boosting algorithm performs the best among the feature-engineered methods.

Silla et al. [3] present a non-conventional approach to address the automatic music genre classification problem. The proposed method employs multiple feature vectors and a pattern recognition ensemble approach based on space and time decomposition schemes. Despite being a multi-class problem, the task is accomplished using a set of binary classifiers whose results are merged to obtain the final music genre label, employing space decomposition. The music segments are also decomposed into time segments from the beginning, middle, and end parts of the original music signal, utilizing time decomposition. Classical machine learning algorithms, such as Naïve-Bayes, Decision Trees, k Nearest-Neighbors, Support Vector Machines, and Multi-Layer Perceptron Neural Nets, are employed for training.

Experiments are conducted on the Latin Music Database, which contains 3,160 music pieces categorized into ten musical genres. The results show that the proposed ensemble approach outperforms global and individual segment classifiers in most cases. Experiments related to feature selection using the genetic algorithm paradigm reveal that the most important features for the classification task vary according to their origin in the music signal.

The paper evaluates the effect of using the ensemble approach in the music genre classification problem, where individual classifiers are applied to a special decomposition of the music signal encompassing both space and time dimensions. Space decomposition strategies based on the One-Against-All and Round-Robin approaches are used, along with features extracted from different time segments. Multiple feature vectors and component classifiers are employed in each music part, and a combination procedure produces the final class label for the music.

The research conducted in the article [4] compares different classification models and introduces a novel Convolutional Neural Network (CNN) model, outperforming previous approaches. The models are trained and compared on the GTZAN dataset, which

consists of audio files from various genres, with some models trained on spectrogram representations.

Audio features relevant to solving the classification problem are extracted, comprising Time Domain Features and Frequency Domain Features. These features are used as inputs to the CNN model, where the training images (spectrogram slices) pass through a four-layer convolution neural network to extract relevant features. The extracted features then go through a second sub-network for classification, a fully connected network containing two fully connected layers. A dense layer is utilized to predict the genre of the audio.

The proposed model uses various inputs, including audio Mel spectrogram and sound file characteristics stored in ANN, SVM, MLP, and Decision Tree csv archives, achieving an accuracy of 91%, comparable to human understanding of genre classification with the highest accuracy. While some music styles are easily distinguishable, others, such as country and rock genres, might be confused with other styles. Traditional and blues genres are more readily identified.

The proposed research work enhances music genre classification using the GTZAN dataset and demonstrates the effectiveness of the CNN model and other classifiers in achieving accurate genre classification results.

Qi et al. [5] explore and identify a more effective machine learning algorithm for predicting the genre of songs compared to existing models. Multiple classification models were constructed and trained using the Free Music Archive (FMA) dataset. The performances of these models were compared, and their prediction accuracies were recorded. Some of these models were trained on the mel-spectrograms of the songs along with their audio features, while others were solely trained on the spectrograms.

The subset of the Free Music Archive dataset used for evaluation, known as fma_small, comprises a balanced collection of 8000 songs categorized into eight genres. The dataset offers 30-second audio segments with pre-computed features, along with track-level and user-level metadata, tags, and free-form text. Variants of the FMA dataset, such as fma_medium, fma_large, and fma_full, are also available, each containing varying numbers of tracks and genres.

Among the models tested, a convolutional neural network (CNN) using only the spectrograms as the dataset exhibited the highest accuracy. The CNN model using only spectrogram input achieved a test accuracy of 88.54%, outperforming the CRNN and CNN-RNN models, which had more complex designs but lower accuracies, possibly due to the limited dataset size. Models that used handcrafted features, such as Logistic Regression (LR) and simple Artificial Neural Network (ANN), exhibited comparatively lower accuracy than the CNN model.

In this study, image-based classification demonstrated superior performance compared to feature-based classification.

The study by Elbir et al. [6] focuses on the extraction and utilization of various acoustic features from music for the purpose of music genre classification and recommendation. The selected features include zero-crossing rate, spectral centroid, spectral contrast, spectral bandwidth, spectral roll-off, and Mel-frequency Cepstral Coefficients-MFCC. Among the deep learning methods, Convolutional Neural Network-CNN has

been employed due to its effectiveness. The CNN is applied by transforming the uploaded song's spectrogram through Short Time Fourier Transform (STFT), utilizing the output of a dense layer as a feature vector for music genre classification and song similarity calculation.

The results indicate that SVM outperformed other methods in classification, and variations in window size and type had minimal impact on performance. MFCC had a more significant effect on classification performance compared to other methods. Despite the use of deep learning methods, no significant performance difference was observed in music genre classification, with SVM achieving higher success than the CNN algorithm. Regarding music recommendations, the study proposes recommending songs by genre as a solution to the challenge of the lack of objective metrics for music recommendation.

In this section of the literature review, aimed at reviewing previous work that applied Machine Learning to music, all works treat the domain of genre classification, particularly in audio-based representations. Different approaches are presented, with more traditional and more recent techniques and different ways of treating data.

2.2 Latent Space Exploration

During the comprehensive search and exploration of the literature, a notable collection of scholarly papers and research works has been identified, all of which delve into the intriguing and fruitful connection between latent spaces and the domain of music. These academic publications provide valuable insights, methodologies, and findings, shedding light on the utilization of latent spaces as a fundamental framework for representing, modeling, and analyzing musical data.

Roberts et al. [7] introduce interfaces that allow users to explore low-dimensional control spaces for high-dimensional note sequences, providing a simple 2-D grid for composing melodies or drum beats. The interface for musical creation in this system consists of a 2-D control surface called the palette, which is combined with four user-defined basis points (the corners). The MusicVAE sequencer allows users to record MIDI or audio externally, or to download a MIDI file containing a particular sequence to use as part of an arrangement. Users can also use the "draw" tool to define a 1-D curve within the palette, which the puck will then move through at a rate the user controls.

To make this system accessible to a wide range of users, the researchers have curated a set of original sounds ahead of time and synthesized all of their interpolated latent representations, rather than generating sounds on demand. This is because the NSynth model used in this system is extremely computationally expensive (~30 minutes of GPU synthesis to generate four seconds of audio). Additionally, machine learning techniques can learn the shape of low-dimensional manifolds from data, allowing us to avoid heuristics and hand-tuned features, along with the biases and preconceptions about the data that would normally accompany those.

This work exemplifies how machine learning can serve as the foundation for creative musical tools, challenging the traditional view of machine learning solely as a means for outsourcing discriminative tasks. Instead, it points towards the promising direction of

using machine learning, particularly latent space representations, to create UI mappings that offer music creators novel and compelling creativity-supporting tools.

Shen et al. [8] explore the use of deep learning models for generating compact and effective numerical representations of traditional Chinese music data. Traditionally, handcrafted features have been employed for such tasks, but the potential of deep learning models, particularly autoencoders, in extracting latent representations is gaining popularity. However, in the context of music information retrieval (MIR), especially with regard to incorporating visualization, limited attention has been given to this area.

To address this gap, the researchers propose a visual analysis system that utilizes autoencoders to facilitate the analysis and exploration of traditional Chinese music. Due to the scarcity of appropriate traditional Chinese music data, a labeled dataset is constructed from a collection of pre-recorded audio files, which are then converted into spectrograms. The system takes music features learned from two deep learning models: a fully-connected autoencoder for note latent vectors and a Long Short-Term Memory (LSTM) autoencoder for segment latent vectors. Interactive selection, similarity calculation, clustering, and listening are performed using the encoded data's latent representations, allowing the system to identify essential music elements and lay the groundwork for further analysis and retrieval of Chinese music.

The spectrogram is chosen as the initial representation of music, containing both frequency and amplitude information of the music signals. To extract more compact features, two neural network models, the fully-connected autoencoder, and the LSTM Autoencoder are used. The autoencoders' latent spaces are employed to generate music features, which are then subjected to visual analysis in the music latent space.

Roberts et al. in [9], proposes a solution to the use of VAE in sequential data. The VAE has shown promise in generating semantically meaningful latent representations for natural data. However, its application to sequential data has been limited, and existing recurrent VAE models face challenges in modeling sequences with long-term structure. To address this issue, this work studies a hierarchical decoder, which first outputs embeddings for subsequences of the input and then uses these embeddings to independently generate each subsequence. This architecture encourages the model to effectively utilize its latent code, thus mitigating the "posterior collapse" problem that is commonly encountered in recurrent VAEs.

The experiments conducted to evaluate the performance of MusicVAE in modeling long-term structure in music are divided into two parts: quantitative evaluation and qualitative evaluation.

In the quantitative evaluation, the authors compare the performance of MusicVAE to a "flat" baseline model on three tasks: reconstruction, sampling, and interpolation. They use two datasets of symbolic music: the JSB Chorales dataset and the Piano-MIDI.de dataset. The results show that MusicVAE outperforms the baseline model on all three tasks, with statistically significant differences in most cases.

In the qualitative evaluation, the authors conduct a listening test to evaluate the musical quality of the generated samples. They use three hierarchical models and one flat model to generate samples, and they also include samples from the real dataset. The results show that participants rated samples from the hierarchical models as more

musical than samples from the corresponding flat models or the real dataset. There was no significant difference between samples from the hierarchical models and the real dataset.

The experiments demonstrate that MusicVAE is a powerful model for learning long-term structure in music, and it outperforms existing models on a variety of tasks.

The study of Pati and Lerch [10] introduces a novel latent space regularization technique to enhance the interpretability of deep generative models for music. The proposed approach allows users to encode musically meaningful attributes along specific dimensions of the latent space, enabling them to exercise explicit control over these attributes during inference. By structuring the latent space with musically relevant attributes, this technique offers an intuitive way to design musical interfaces that facilitate creative workflows.

To assess the effectiveness of the proposed method, experiments were conducted using hierarchical VAE models trained on a dataset of monophonic folk melodies. The regularization technique was applied to two attributes: rhythmic complexity and pitch range. The results demonstrate that the regularized models show a clear ordering of attributes along their respective dimensions in the latent space, while the baseline model lacks such structure.

Furthermore, attribute surface maps obtained by decoding latent vectors on a two-dimensional plane of the latent space reveal a similar structure, with attribute values monotonically ordered along the corresponding regularized dimensions. The model evaluation using an interpretability metric indicates that the regularized models achieve higher scores compared to the baseline, demonstrating the effectiveness of the proposed method.

This study provides evidence that the proposed latent space regularization technique enables the encoding of selected musical attributes and offers users more intuitive control over the generated music. The simplicity of the regularization loss computation and the absence of hyperparameter tuning make this approach promising for practical applications.

Yang et al. work in [11] introduces an explicitly constrained conditional variational autoencoder (EC2-VAE) as an effective solution to address the challenges of analogy-making in computer-generated music. The process of analogy involves partially transferring high-level music abstractions, such as pitch and rhythm representations and their relationships, from one music piece to another. However, this procedure requires the disentanglement of music representations, which is straightforward for musicians but non-trivial for computers. Three sub-problems arise: extracting latent representations from the observation, disentangling the representations to ensure unique semantic interpretation, and mapping of the disentangled representations back to actual music.

The proposed EC2-VAE offers a unified solution to all three sub-problems. It disentangles the pitch and rhythm representations of 8-beat music clips conditioned on chords, enabling the generation of music analogies with altered pitch contours, rhythm patterns, and chord progressions. The disentanglement is explicitly coded, allowing specific latent dimensions to represent distinct semantic factors in the model structure. Moreover, it preserves the intrinsic relationship between representations and does not

sacrifice much of the reconstruction process. The EC2-VAE learns to make analogies without requiring analogous examples during the training phase.

To evaluate the proposed method, objective measurements and a subjective survey were conducted. The results showed significant improvement over baselines in terms of creativity and musicality. The proposed model demonstrated the ability to generate interesting and analogous musical versions of existing music through analogical reasoning.

In this section of the state of art, dedicated to the application of latent spaces to the musical domain, the different works are much more diverse. This portrays the versatility of the latent space concept and the many different ways it can be made use of.

2.3 Variational Autoencoders

In the field of artificial intelligence, VAE, or Variational Autoencoder, is an architecture of artificial neural networks used for unsupervised learning and data generation. It is a variant of the traditional autoencoder, which is a type of neural network that learns to compress data into a lower-dimensional space and then decompress it to obtain a reconstruction close to the original data.

The main difference between VAE compared to the conventional autoencoder is in how it approaches the encoding and decoding process. Instead of using a deterministic encoder, VAE employs a probabilistic encoder that maps the input data to a probability distribution in the latent space. This distribution can be interpreted as a continuous and distributed representation of the data, enabling greater flexibility and generalization in the generation process.

VAE is trained using the concept of inference and generation. During the training phase, the model attempts to learn the distributions that best describe the input data and then uses optimization techniques to minimize the discrepancy between the distribution of the real data and the distribution generated by the model. Once trained, VAE can generate new data samples by randomly sampling from the latent space and decoding these sample points to produce synthetic data resembling the original input data.

This ability to generate new data and perform smooth interpolations in the latent space makes VAE useful for tasks of generation and creative exploration in artificial intelligence. It has been successfully applied in various areas, such as image generation, music, text, and other types of data, and has proven to be a powerful tool for unsupervised learning and high-dimensional data generation.

Research works on this matter have predominantly focused on monophonic music, while the polyphonic counterpart, characterized by richer modality and more complex musical structures, remains relatively unexplored in the context of music representation learning.

Wang et al. [12] propose a novel extension of VAE called "PianoTreeVAE," which introduces a tree-structured architecture to specifically cater to polyphonic music learning.

The experiments conducted in this study demonstrate the efficacy of PianoTreeVAE in multiple aspects: (i) producing semantically meaningful latent codes for polyphonic music segments; (ii) achieving more satisfactory reconstruction results along with decent

geometry learning in the latent space; and (iii) showcasing the model's benefits for a variety of downstream music generation tasks.

The objective evaluation involves comparing different models based on their reconstruction accuracy of pitch onsets and note duration, which are commonly used measurements in music information retrieval tasks.

The results show that PianoTreeVAE, with its design incorporating both the music data structure and model architecture, along with sparsity and hierarchical priors, outperforms other approaches in terms of reconstruction, interpolation, downstream generation, and overall model interpretability.

Brunner at al. [13] introduce MIDI-VAE, a neural network model based on Variational Autoencoders, designed to handle polyphonic music with multiple instrument tracks while capturing the dynamics of music through note durations and velocities.

The model uses parallel Variational Autoencoders with a shared latent space and an additional style classifier, which encodes style information in the shared latent space, facilitating style manipulation.

The model is evaluated using separate style validation classifiers and can interpolate between short music pieces, create medleys, and generate mixtures of entire songs with smooth transitions in pitch, dynamics, and instrumentation. MIDI-VAE operates on symbolic music representations extracted from MIDI files, extending the standard piano roll representation of note pitches with velocity and instrument rolls to effectively model MIDI data.

MIDI-VAE demonstrates its effectiveness in style transfer through smooth interpolations between bars, enabling the generation of medleys and mixtures with harmonic bridges between pieces. The proposed model outperforms existing approaches by incorporating both dynamics and instrumentation of music.

Jiang et al. in [14] aimed to address the desired properties of structure awareness and interpretability in music generation algorithms. To achieve these goals simultaneously, they introduced the Transformer Variational AutoEncoder (VAE), a hierarchical model that combines two recent breakthroughs in deep music generation: the Music Transformer and Deep Music Analogy.

The Music Transformer learns long-term dependencies in music using an attention mechanism, while Deep Music Analogy focuses on interpretability by employing a disentangled conditional-VAE. The Transformer VAE demonstrated the ability to learn a context-sensitive hierarchical representation, with local representations serving as context and dependencies among them forming the global structure.

Experimental results showed that the Transformer VAE achieved satisfactory reconstructions for phrase-level music and successfully transferred melodic and rhythmic contexts from one phrase to another. This allowed for context transfer, enabling the model to generate music in the style and flow of another piece.

Yang et al [15] propose a novel method termed "disentanglement by augmentation" that enables the inspection of pitch and rhythm interpretations within the latent representations. By leveraging interpretable representations, they design an intuitive graphical user interface to empower users in directing the music creation process through the manipulation of pitch contours and rhythmic complexity.

The investigation focuses on a compressed MusicVAE model, revealing that the latent space can be disentangled, with certain dimensions associated with pitch changes and others linked to rhythm changes. By adjusting values in these selected dimensions, the average pitch and rhythm complexity can be modulated, or music clips can be interpolated based on pitch or rhythm separately.

However, the proposed method has certain limitations. First, it is incapable of determining the consequences of more complex operations on the latent representation. Additionally, the inspection process relies on manual disentanglement, suggesting the potential for future research to design an automatic disentangling model for music.

The current landscape of multi-track symbolic music generation predominantly relies on Convolutional Neural Networks (CNNs) due to the additional dimension of tracks in the data representation, rendering Recurrent Neural Network (RNN)-based models unsuitable for this task.

Liang et al [16] explore the problem of multi-track symbolic music generation being unattainable using RNN-based generation models.

A multi-modal fusion generation model is proposed employing a layered VAE architecture. This model is divided into two major modules, namely BVAE and MFG-VAE. The BVAE, a RNN-based VRAE, enhances the performance of BMuseGAN by transforming the binarization problem into a multi-label classification problem. This addresses the challenge faced in differentiating the refinement step in the original scheme and the difficulty in gradient descent within BMuseGAN. The underlying BVAE model is independent for processing distinct track data. The upper model MFG-VAE adopts the multi-modal fusion-generation VAE architecture.

The encoder serves as the multi-modal fusion network, while the decoder functions as the multi-modal generation network. Through the fusion/generation of the latent vector generated by the underlying BVAE, the entire model can generate multi-track symbolic music.

Cifka et al [17] present a novel method for one-shot instrument timbre transfer, based on an extension of the vector-quantized variational autoencoder (VQ-VAE) and a simple self-supervised learning strategy to obtain disentangled representations of timbre and pitch. The method is evaluated using objective metrics and demonstrates superior performance compared to selected baselines. The contributions of this work can be summarized in three aspects: (i) introducing the first neural model for one-shot instrument timbre transfer, utilizing mutually disentangled pitch and timbre representations learned in a self-supervised manner without annotations; (ii) training and testing the model on a dataset containing single, possibly polyphonic instrument recordings; (iii) the data-driven approach to disentanglement can be extended to other music transformation tasks, such as arrangement or composition style transfer.

2.4 Machine Learning Models

Variational Autoencoders (VAEs) and transformers are two revolutionary models in the field of machine learning that have transformed the way we approach various data processing tasks. Both models have shown great potential and have been widely used in

applications such as natural language processing and content generation.

Variational Autoencoders (VAEs) are a class of generative models that have been highly successful in generating data from latent distributions. The fundamental idea behind VAEs is to use variational inference theory to learn the latent distribution of data and, in turn, generate samples from that distribution. VAEs have been successfully applied in various tasks such as image generation and speech synthesis and have demonstrated their ability to generate realistic and high-quality data.

On the other hand, transformers, are a type of deep learning model architecture based on the attention mechanism. Transformers have particularly excelled in sequence processing and have overcome the limitations of traditional recurrent models in capturing long-term dependencies. The architecture of Transformers allows for parallel processing of sequences, making them highly efficient and suitable for tasks involving large volumes of data.

In this section, we will focus on explaining in detail the fundamental concepts of Variational Autoencoders and transformers, analysing their respective architectures and key mechanisms. Additionally, we will discuss the advantages and applications of each model in various areas, highlighting how they have significantly impacted natural language processing and other machine learning tasks.

As we explore and understand these models, we hope to provide a clear and comprehensive insight into their capabilities and how they have opened new possibilities for the development of intelligent systems and practical applications in artificial intelligence.

Variational Autoencoders (VAEs). In this section, we present a detailed review of VAEs, describe their structure, functioning, and applications in different domains, and analyse their advantages and limitations.

1. Introduction. VAEs are a type of generative model that belongs to the family of Deep Neural Networks (DNNs). They were initially proposed by Kingma and Welling in 2013 as an extension of traditional Autoencoders, with the aim of learning a latent representation of high-dimensional data and performing the generation of novel samples.

2. Structure and operation. VAEs consist of two main components: the encoder and the decoder. The encoder takes a sample of high-dimensional data and maps it to a probability distribution in the latent space. This distribution is typically characterized by a mean and a variance. Next, a technique called stochastic sampling is used to generate a random latent sample from this distribution. Finally, the decoder takes the generated latent sample and reconstructs it in the original data space.

3. Advantages. Firstly, VAEs excel in learning meaningful latent representations of the data, allowing the model to capture relevant information in a compact and structured manner. This enhances tasks like classification, generation, and style transfer. Secondly, VAEs are capable of generating novel samples within the original data space. The continuous distribution of the latent space enables stochastic sampling, facilitating the exploration of different regions and the generation of samples not present in the training set. This attribute makes VAEs particularly

valuable for creative generation applications, including music, image, and text generation.

4. Applications. VAEs have demonstrated success in a wide range of applications, including generating realistic images, modeling chemical molecules, music generation, and coherent text creation. They have also been used in style transfer problems, where data generation is performed while maintaining certain specific attributes while manipulating others.

Transformers. In this section, we describe the transformer architecture, its key components, and the principles that underline its success. Additionally, we discuss the advantages and applications of the transformer in various tasks, including language translation, text generation, and image processing.

1. Introduction. The transformer model was developed as a response to the limitations of traditional RNNs, which struggle with parallelization and capturing long-range dependencies due to their sequential nature. The transformer model's introduction marked a significant milestone in the domain of machine translation and quickly found applications in various other tasks as well.

2. Transformer architecture. The transformer model comprises two key components: the encoder and the decoder. Each component consists of multiple layers, and these layers are built around the self-attention mechanism. Self-attention allows the model to weigh the importance of different positions in the input sequence when generating each output, enabling it to effectively capture long-range dependencies and maintain contextual coherence. Unlike RNNs, which process sequential data one step at a time, self-attention allows the Transformer to process all words in the sequence simultaneously, making it highly parallelizable and efficient.

3. Positional encoding. Since the transformer model does not inherently capture the sequential nature of the input, positional encoding is introduced to inject positional information into the input embeddings. Positional encoding enables the model to understand the order and position of words in the sequence, which is crucial for maintaining the contextual meaning of the input.

4. Advantages and applications. The transformer's ability to efficiently handle long-range dependencies and capture contextual information has led to its success in various natural language processing tasks. It has achieved state-of-the-art results in machine translation, language modeling, text generation, sentiment analysis, and question-answering tasks. Additionally, the Transformer has also found applications beyond NLP, such as image captioning and speech recognition.

To sum up, Variational Autoencoders are a powerful tool in deep learning, especially when it comes to learning latent representations and data generation. However, they also face challenges like computational complexity or the difficulty in evaluating the quality of generated samples. On the other hand, the transformer model has revolutionized the field of natural language processing. Its innovative self-attention mechanism, parallel

processing capabilities, and efficient handling of long-range dependencies have made it a preferred choice for a wide range of applications.

The research paper "All You Need is Attention", [18], holds significant importance in the field of natural language processing and machine learning due to its groundbreaking introduction of the Transformer model architecture. Published by Vaswani et al. in 2017, this paper presented a novel self-attention mechanism that revolutionized language modeling and sequence-to-sequence tasks. The transformer's attention mechanism allowed for more efficient parallelization and reduced the reliance on recurrent neural networks (RNNs), thereby overcoming the limitations of sequential processing. It also facilitated better learning and understanding of patterns in language, enabling substantial improvements in machine translation, language generation, and other natural language processing tasks.

Moreover, the Transformer's parallelization capabilities significantly accelerated model training, leading to more efficient and scalable language models. Its wide adoption and contributions to transfer learning has also spurred advancements in other domains beyond natural language processing, such as computer vision and audio processing.

3 Model Introduction

3.1 Comparative: VAEs and Transformers

This chapter presents an in-depth and comprehensive comparison between two prominent machine learning models, Variational Autoencoders (VAEs) and Transformers, summarized in Section 2.4. We aim to highlight the advantages of transformers and the limitations of VAEs. We discuss the fundamental concepts, architectures, and key mechanisms of both models, exploring their applications in different domains. Through this analysis, we explain why transformers have gained popularity over VAEs in various tasks and provide insights into the potential areas of transformer adoption.

As stated in Chapter 2 Machine learning has witnessed significant advancements with the emergence of novel models, including VAEs and transformers. VAEs are generative models that have proven successful in generating data from latent representations, while transformers have revolutionized natural language processing and sequence modeling tasks.

Variational Autoencoders (VAEs) are probabilistic generative models designed to learn a latent representation of data through an encoder-decoder architecture. VAEs optimize the lower bound of the log-likelihood to infer the latent variables and generate new data samples. While VAEs have shown promise in image generation and data compression, they come with certain drawbacks:

- Inherent latent space constraints: VAEs often suffer from limited representation power due to the smoothness assumption of the latent space, which may lead to blurry or unrealistic sample generations.
- Mode collapse: VAEs are prone to mode collapse, wherein the model generates samples concentrated around a few dominant modes, limiting the diversity of generated data.
- Difficulty in capturing long-term dependencies: VAEs struggle to capture long-range dependencies in sequential data, hindering their performance in tasks involving complex sequential patterns.

Transformers, on the other hand, are based on the self-attention mechanism and have proven to be highly effective in various natural language processing tasks and sequence modeling. Transformers employ an attention mechanism to process sequences in parallel, enabling efficient processing of long-range dependencies. The key advantages of transformers are as follows:

- Parallelism: transformers can process sequences in parallel, leading to faster training and inference times, making them highly scalable for large datasets.

- Global context awareness: the attention mechanism allows transformers to capture long-range dependencies in sequential data, facilitating better context understanding and more coherent generation.

- Multi-modal processing: transformers can handle multiple modalities of data, such as text and images, in a unified manner, enabling more versatile applications.

To sum up, the superior capabilities of transformers in handling long-range dependencies, generating coherent sequences, and processing multiple modalities make them the preferred choice for various machine learning tasks.

3.2 Variational Autoencoders and Transformers in MIDI Music Language Processing

This section presents a comparative study of Variational Autoencoders (VAEs) and Transformers in the context of processing musical language for MIDI music representation. We present the fundamental concepts, architectures, and key mechanisms of both models, evaluating their applications in the domain of MIDI music generation and understanding. MIDI music, represented as sequential data, poses unique challenges that require models capable of capturing long-range dependencies and generating coherent musical sequences. Through this analysis, we highlight the advantages of transformers over VAEs in handling long-range dependencies, generating coherent musical sequences, and supporting multi-modal processing, making them a superior choice for MIDI music language processing.

Variational Autoencoders (VAEs) as MIDI Music Language Processing have been applied to music generation tasks, including MIDI music. They offer the ability to learn a latent representation of MIDI data and generate novel sequences based on learned priors. However, VAEs encounter certain challenges in the context of MIDI music processing:

- Latent Space Constraints: VAEs may suffer from a limited representation of complex musical patterns due to the smoothness assumption in the latent space, leading to less expressive and coherent musical sequences.

- Model Collapse: Like other applications, VAEs in MIDI music generation may experience model collapse, limiting the diversity of generated musical phrases.

- Long-Term Dependency Handling: Capturing long-term dependencies in MIDI music is essential for generating musically meaningful compositions, but VAEs may struggle in this regard.

Transformers as MIDI Music Language Processing. Transformers have shown exceptional capabilities in natural language processing tasks and have been successfully extended to handle sequential data such as MIDI music representation. The advantages of using Transformers in MIDI music language processing are as follows:

- Global Context Awareness: Transformers' self-attention mechanism enables the model to capture long-range dependencies in musical sequences, allowing for better context understanding and coherent composition.
- Parallel Processing: Transformers process MIDI music sequences in parallel, significantly reducing training and inference times, making them suitable for large MIDI datasets.
- Multi-Modal Processing: Transformers can handle multiple modalities of MIDI data, such as pitch, duration, and velocity, in a unified manner, supporting multi-instrument compositions and versatile MIDI music applications.

In MIDI music language processing, transformers offer significant advantages over VAEs. Their ability to capture long-term dependencies, generate coherent and expressive musical sequences, and support multi-modal processing makes them well-suited for complex MIDI music generation tasks. The parallel processing capability of transformers enables efficient handling of large MIDI datasets, making them scalable for real-world applications. In contrast, the limitations of VAEs in handling long-range dependencies and mode collapse make them less ideal for MIDI music language processing.

Transformers have achieved state-of-the-art performance in various natural language processing tasks, such as language modeling and machine translation. Their success in these tasks indicates their potential for handling sequential data effectively. Recent studies have demonstrated the application of transformers in music generation tasks, with impressive results in generating expressive and coherent music [19, 7].

To sum up, the choice of transformers over VAEs for MIDI music language processing is justified by their ability to handle long-range dependencies, generate coherent musical sequences, and support multi-modal processing. Transformers have shown promising results in music generation tasks and have become a prominent choice for processing sequential data. Leveraging their global context awareness and self-attention mechanisms, transformers offer a robust and powerful approach for generating musically meaningful and expressive MIDI compositions.

3.3 Application of Transformers in Music Processing

This section explores the use of transformers in music processing, highlighting their applications and advantages in the musical domain. A research review is conducted, and relevant examples are presented to showcase how transformers have been employed in music generation, AI-assisted composition, and audio quality enhancement.

As stated in Section 3.2, music processing has witnessed significant advancements through the adoption of artificial intelligence techniques. Among these techniques, transformers have proven to be a powerful tool in tasks related to music. They have been effectively adapted to work with music sequences, enabling analysis and generation of musical content.

Processing music sequences with transformers. Relevant uses of transformers in music processing, as reference

The introduction of the transformer architecture [18] and its application in natural language processing tasks highlighted its ability to handle sequences. This characteristic is essential for its utilization in the musical domain, where sequences represent aspects such as melodies, chords, and rhythms. As a consequence, transformed have been applied to various applications in music:

- Music Generation. Dhariwal et al. [20] present a relevant example of how Transformers have been used for generating original music. They mention the use of the GPT-2 architecture in polyphonic music generation and realistic improvisation. This approach has enabled the production of creative music through generative models.
- AI-Assisted Composition. The article also highlights the use of transformers in AI-assisted music composition [20]. These models can propose musical arrangements and specific styles, facilitating the exploration of new ideas and approaches in composition.
- Audio Quality Enhancement.Transformers have been also used to improve the quality of music recordings through noise reduction techniques and signal reconstruction [21, 22]. This has contributed to the preservation and restoration of musical heritage.

In summary, the use of transformers in music processing has led to significant advancements in various areas, such as music generation, AI-assisted composition, and audio quality enhancement. As research in artificial intelligence continues to progress, transformers are expected to remain a valuable tool for the evolution and appreciation of music.

3.4 Application of BERT in Music Processing

This section delves into the utilization of Bidirectional Encoder Representations from Transformers (BERT) in the field of music processing [23]. By conducting a comprehensive literature review, the section explores the various ways BERT has been employed to address music-related tasks, including music recommendation, mood analysis, and music generation. Additionally, the section sheds light on the potential benefits and challenges associated with incorporating BERT into music processing pipelines.

With the advent of natural language processing (NLP) models like BERT, the domain of music processing has witnessed a paradigm shift. BERT, a transformer-based neural network architecture, has demonstrated exceptional performance in capturing context and semantics from sequential data, which has been found applicable in various areas beyond NLP, including music processing.

BERT, as introduced in [23], is a pre-trained model that utilizes a bidirectional approach to comprehend the context of words in a sentence. This unique ability to capture context makes BERT suitable for music processing tasks, where sequential patterns play a significant role. In the context of music, BERT can analyse and interpret sequential

musical elements, such as notes, chords, and rhythms, enhancing the understanding and processing of musical data.

- Applications of BERT in Music Processing.
  - Music Recommendation. Various research jobs ([24, 25, 26]) present a compelling application of BERT in music recommendation. By leveraging its contextual comprehension capabilities, BERT can effectively capture user preferences and interpret music features, leading to more accurate and personalized music recommendations.
  - Mood Analysis in Music. Choi et al ([27]) illustrate how BERT has been used for mood analysis in music. By analyzing lyrics and music features, BERT can effectively identify the emotional content of songs, facilitating the categorization of music based on moods and sentiments.
  - Music Generation. Research done by Hawthorne et al ([28]) explores the use of BERT in music generation. By training the model on large-scale music datasets, BERT can generate novel musical sequences with coherent structures, showcasing its potential as a creative tool for composers and music enthusiasts.
- Benefits and Challenges. BERT's strength lies in its ability to capture rich contextual information, leading to enhanced performance in music-related tasks. However, the large model size and computational requirements of BERT present challenges in real-time applications and resource-constrained environments [29].

In conclusion, the application of BERT in music processing holds tremendous potential to revolutionize various music-related tasks. Its success in music recommendation, mood analysis, and music generation showcases the versatility and efficacy of this NLP model in a music context. However, addressing computational challenges will be vital to fully harness BERT's capabilities and ensure its seamless integration into music processing pipelines. For this reason, we have considered the application of BERT-based models as a promising area of research.

3.5 Application of Midi-BERT in Music Processing

This section explores the adaptation of Midi-BERT, a specialized variant of the Bidirectional Encoder Representations from Transformers (BERT), for music-related tasks. Through an extensive literature review, the current thesis part investigates the architecture and capabilities of Midi-BERT, focusing on its applications in music understanding, composition, and generation. Additionally, this thesis part discusses the potential impact of Midi-BERT on the field of music processing and its implications for future research. With the growing interest in applying natural language processing (NLP) models to music, Midi-BERT emerges as a promising approach tailored specifically for music data.

Derived from the original BERT architecture, Midi-BERT is designed to understand and process music sequences represented in the Musical Instrument Digital Interface (MIDI) format.

In this context, the application of Midi-BERT allows for the identification and analysis of musical elements relevant to genre characterization, such as rhythmic patterns, harmonic structures, and the use of specific instruments. Prior work has demonstrated the potential of Midi-BERT in classifying and labeling musical pieces based on their specific genres [29, 30, 31].

This adaptation of BERT to the task of genre analysis in music represents a significant advancement in the field of computational music, providing new opportunities for studying and understanding the distinctive characteristics that define different musical genres. The current section delves into the applications and potential benefits of Midi-BERT in various music-related tasks.

Relatively little research has been done on music understanding technology for music in symbolic formats such as MusicXML and MIDI. The job by Chou et al [30] provides detailed insights into the implementation of Midi-BERT, where the model's bidirectional attention mechanism is adapted to capture the sequential patterns and hierarchical structures inherent in MIDI data. This transformation enables Midi-BERT to encode musical sequences more effectively and enhance its understanding of music elements, such as notes, chords, and tempo. Several promising applications of the model include the application of Midi-BERT in music understanding tasks, music composition and music generation:

- By training Midi-BERT on large-scale music datasets, the model can capture complex relationships between musical events, facilitating tasks like music segmentation, instrument recognition, and phrase analysis. This use of Midi-BERT in music understanding tasks lays the groundwork for more sophisticated music analysis and interpretation.

- By fine-tuning the model on music composition datasets, Midi-BERT can generate harmonious and coherent musical sequences based on input criteria. This application empowers composers and musicians with a creative tool to explore novel musical ideas and facilitate the composition process.

- By leveraging its hierarchical attention mechanism, Midi-BERT can generate realistic and expressive music sequences, offering potential applications in automatic music composition and creative content generation.

The potential impact of Midi-BERT on the field of music processing is huge. As stated by Zhu et al [31], the adaptation of BERT for music opens new research avenues in improving music understanding, analysis, and creative expression. Moreover, this adaptation paves the way for developing more sophisticated music AI systems that can interact with musicians and audiences in novel ways.

To conclude, the adaptation of Midi-BERT for music presents a significant advancement in the field of music processing. With its specialized architecture tailored for MIDI data, Midi-BERT demonstrates promising capabilities in music understanding, composition, and generation. The successful application of Midi-BERT opens exciting possibilities for enhancing music-related tasks and inspires further research to unlock the full potential of this NLP-based model in the realm of music.

4 Methodology

In this chapter, we present the procedure followed for the research undertaken in this study. Firstly, in Section 4.1 we provide a comprehensive exposition of the primary works that have served as sources of inspiration for our investigation. By presenting a coherent narrative of the foundational papers and our original research, we aim to offer a comprehensive understanding of the scholarly endeavors that have shaped the course of this study. Secondly, in Section 4.2 we delve into a detailed account of the process followed in conducting the research conducted in this thesis.

4.1 Background

This section is devoted to providing a detailed explanation of the primary article upon which we have grounded our research: the paper *MIDIBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding* [30].

This paper presents an empirical study that employs PTMs for symbolic-domain music understanding tasks. Inspired by the trend of treating MIDI music as a "language" in deep generative models, a transformer-based network is pre-trained using a self-supervised training strategy called "mask language modeling" (MLM), widely used in BERT-like PTMs in NLP. Despite BERT's fame, only two publications are known to employ BERT-like PTMs for symbolic music classification. The PTM is evaluated on four music understanding tasks, including note-level classification tasks like melody extraction and velocity prediction, as well as sequence-level classification tasks like composer classification and emotion classification. The study employs a total of five datasets to conduct the evaluation and analysis.

The Datasets section describes five publicly available datasets used in the study, consisting of classical and pop music pieces in MIDI format. These datasets were used to pre-train the MIDIBERT-Piano model and evaluate its performance on four downstream symbolic music understanding tasks. The section provides important statistics of the datasets and details on their composition. Additionally, links to the datasets and pre-trained and fine-tuned models are provided for reproducibility purposes. These datasets were used to pre-train the MIDIBERT-Piano model and evaluate its performance on four downstream symbolic music understanding tasks. The section provides important statistics of the datasets and details on their composition. Additionally, links to the datasets and pre-trained and fine-tuned models are provided for reproducibility purposes.

The results show that MIDIBERT-Piano consistently outperforms the Bi-LSTM or Bi-LSTM-Attn baselines in all tasks, regardless of whether the representation is REMI or CP. When combining MIDIBERT-Piano with CP, denoted as MIDIBERT-Piano+CP, the best results are obtained for all tasks. Additionally, MIDIBERT-Piano+CP outperforms

Bi-LSTM+CP with only 1 or 2 epochs of fine-tuning, highlighting the strength of pre-trained models in symbolic-domain music understanding tasks. Moreover, the CP token representation generally outperforms the REMI representation across different tasks for both baseline models and PTM-based models. This observation underscores the importance of token representation for music applications.

The evaluation of the models focuses on melody extraction, velocity prediction, composer classification, and emotion classification tasks. In the melody extraction task, the proposed model exhibits significant improvement compared to the baseline, achieving an accuracy boost of almost 8%. In a simplified binary classification problem of "melody vs. non-melody," our model's accuracy increases further to 97.09%. When compared to the skyline algorithm, known for its limitations in distinguishing between melody and bridge, the model performs favorably.

For the velocity classification task, the accuracy is relatively low due to the subjective nature of velocity and class imbalance. Bi-LSTM tends to classify most notes into the most popular class, "f," and MIDIBERT-Piano shows improved performance but faces challenges in predicting the lowest dynamics, "p" and "pp."

In composer classification, MIDIBERT-Piano significantly outperforms Bi-LSTM-Attn, achieving the highest accuracy. Additionally, a considerable performance gap is observed between REMI and CP representations for this task.

For the emotion classification task, MIDIBERT-Piano also outperforms Bi-LSTM-Attn by a substantial margin in both REMI and CP representations. Although both representations perform well in distinguishing between high arousal and low arousal pieces, they struggle to differentiate along the valence axis.

Ablation studies are conducted to gain insights into the effect of different design choices. These studies include evaluating the model without pre-trained parameters, exploring partial freezing during fine-tuning, and assessing the impact of different pre-training data. The results suggest that pre-training plays a vital role in achieving high accuracy, and partial freezing during fine-tuning leads to a drop in performance. Additionally, the choice of pre-training data affects the composer classification task significantly while having a less pronounced effect on melody extraction and emotion classification tasks.

Overall, the study demonstrates the effectiveness and advantages of MIDIBERT-Piano+CP in symbolic-domain music understanding tasks, establishing it as a strong candidate for further research and application in this field.

The decision to base our implementation and experiments in this thesis on the work presented in the referenced paper is justified by several compelling factors. Firstly, the paper introduces MIDIBERT-Piano, which is among the pioneering large-scale pre-trained models specifically designed for musical data in the MIDI format. By employing five publicly available polyphonic piano MIDI datasets for BERT-like masking-based pre-training, the authors have laid a strong foundation for symbolic music understanding tasks.

The paper's evaluation encompasses four challenging downstream tasks in symbolic music analysis, most of which involve fewer than 1,000 labeled MIDI pieces. Despite the limited labeled data, the experiments demonstrate the efficacy of pre-training for

both note-level and sequence-level classification tasks. The promising results validate the potential of pre-trained models in tackling symbolic-domain music understanding, making it an ideal starting point for our research.

Additionally, the availability of public-domain polyphonic piano MIDI datasets and the open-source nature of the paper's implementation facilitates reproducibility and enables us to build upon the established benchmarks. Leveraging the pre-trained model and openly shared resources allow us to verify the reported results, extend the experiments, and explore various avenues for improvement and novel applications.

In addition, the authors acknowledge that their work can be extended in multiple ways, indicating potential research directions and avenues for future exploration. By building upon this foundational work, we have the opportunity to contribute to the advancement of the field and explore new possibilities in symbolic music understanding.

In conclusion, the comprehensive evaluation, the effectiveness of pre-training demonstrated in challenging tasks with limited labeled data, and the availability of resources and potential for extension make the referenced paper an ideal choice as the basis for our implementation and experimentation in this thesis. By leveraging MIDIBERT-Piano and its publicly available resources, we aim to make meaningful contributions to the field of computational music analysis and further the understanding of symbolic music data.

4.2 Research Strategy

The objective of this work, as stated previously, is to analyze music's latent dimensions. For this purpose, we have employed the Piano MIDI dataset and the MIDIBERT model explained above. In this section, the dataset will be presented and the intended steps to follow will be explained.

The Piano MIDI dataset is a dataset composed of 295 `.mid` files. It is comprised solely of classical works for piano. It features compositions from the Baroque era, like some of Bach's works, the Classical era, like Haydn and Mozart, the Romanticism era, like Beethoven or Mendelssohn, and also works classified as "nationalist" like those by Albéniz or Mussorgsky. All files can be listened to on the dataset's website. As indicated on the website, the pieces are developed at a digital piano by means of a sequencer on MIDI base and then converted to audio formats. Some scores and audio files, showing the scores during playing are available. The full list of pieces is shown in Table 7.1 in the General Adenda chapter.

The dataset employs MIDI format 0. It is one of the three standard formats used to store MIDI data. In this format, all the MIDI data, including note events, control messages, and tempo information, are combined into a single track, making it simpler and more compact compared to other formats. All the musical events are time-stamped based on the global timing resolution, specified in the header. It is suitable for compositions that do not require multitrack structures.

Other datasets were considered, comprised of more modern works like pop music, jazz, or even videogame music. However, it was determined that the dataset used had the perfect size for the hardware constraints faced, and classical music within itself

provided enough variability for a non-trivial result.

The general approach is as follows. Obtain a numeric representation of each music piece employing MIDIBERT. Afterward, a dimensionality-reduction method must be employed on the dataset as a whole to proceed to clustering.

t-SNE (t-distributed Stochastic Neighbor Embedding) was chosen for reducing dimensions. t-SNE is a non-linear dimensionality reduction technique that is often used for visualizing high-dimensional data [32]. It is a popular choice for visualization because it can preserve the local structure of the data, meaning that points that are close together in the high-dimensional space will also be close together in the low-dimensional space. This makes it easier to see patterns and relationships in the data. PCA was also considered as a way to reduce dimensions. However, PCA can only handle linear data and provides a slightly worse performance than t-SNE regarding preserving local structure and managing outliers. It is important to take into account also that t-SNE is computationally more expensive than PCA, and it can be slow to run on large datasets. t-SNE requires as well a bit of hyperparameter tuning, while PCA does not.

We are interested in finding out commonalities among our data observations. One way to determine that commonality or similarity is through a measure of distance among the data points. The shorter the distance, the more similar the observations are. For that purpose, we selected k-means clustering, which tries to minimize the distances between the observations that belong to a cluster and maximize the distance between the different clusters.

Right before clustering, a significant sample of the dataset is randomly taken for manual annotation of different features. This control group is used for two reasons. The first reason is to make sure, besides visually, that the clustering is indeed taking place correctly. The second reason is to delve into the model's categorization of the pieces provided and understand how they are grouped by t-SNE. Finally, conclusions are to be extracted from the findings obtained.

5 Results

In this chapter, we will follow the steps mentioned at the end of Section 4, explaining how they took place and what modifications, if any, were necessary for the correct carrying out of the work.

5.1 MIDIBERT models Hidden States Analysis on MIDI Files

The first step was to adapt the MidiBERT-Piano code to fit our needs. The authors provided code to train the model on compressed data. Since no training was planned for this work, this code was not used. However, the authors did provide a file to perform melody segmentation in a single MIDI file. Since our dataset is not massive and evaluation on single files was needed, the code used for this project was based on the code mentioned. The code was modified so that it would only perform the forward, reshape the hidden states so one was associated with each token, and t-SNE was performed on them.

The size of the MIDI-adapted BERT used is 12 layers and 768 hidden states. Each token forward passed through the model generates a vector of 768 values. Since a maximum sequence of 512 tokens was used, as indicated in Section 4.1, for each batch in the input, we obtained a vector of 512 times 768 values. In addition, since MidiBERT-Piano showed CP tokenization was superior in all tasks, the same one was used in this work. Our smaller MIDI files were made up of 2 or 3 batches, while the biggest ones took as many as 17 batches. This meant that we had as much as $17 \times 512 \times 768 = 6.6 \times 10^6$ floating point values. Working one vector or one `.csv` file of this size is more than manageable. However, working with almost 300 of them would go over the hardware constraints. That is why a decision was made to apply a dimensionality reduction step to each piece.

Before processing all files, a model checkpoint was needed. The authors of the original paper provided checkpoints to the pre-trained model and four finetuned models as follows. These four finetuned models were trained with some sort of classifier employing BERT's output. The four tasks, as mentioned previously, are emotion and composer, which perform a sequence-level based classification of pieces, and velocity and melody, which operate at note-level segmentation.

Figures 5.1 and 5.2 show hidden state analysis of two reference MIDI pieces from the data set with different models integrated in MIDIBERT: Beethoven Sonata No. 5 C minor, Opus 10/1 (1798), first movement and Mozart Sonata No. 8 D major, KV 311 (1777), first movement. The experiments are performed using the default perplexity value of 30 (Figures 5.1(e) and 5.2(e), which gives a good sense of the global geometry in the experiments [32].

The diagrams show a similar distribution of points that forms small clusters of up to

several tens of members. Comparable results were obtained for all the files in the data set. Although the plots for all the models are comparable, the diagrams for the Emotion model are the ones that show the clearest distribution of points in groups, which points to considering Emotion the most appropriate model for analyzing the hidden states obtained when applying the model to an unknown MIDI file.

Of course, finetuning of t-SNE had to also be done. t-SNE takes as input, apart from the data whose dimensions are wished to reduce, perplexity and random state value parameters. t-SNE relies on several parameter settings, *perplexity* being the most significant on t-SNE performance. Perplexity provides an indication of how to balance attention between local and global aspects of data. This parameter represents an estimation of the number of close neighbors each point has. For result replication, value 32 was passed in as random state.

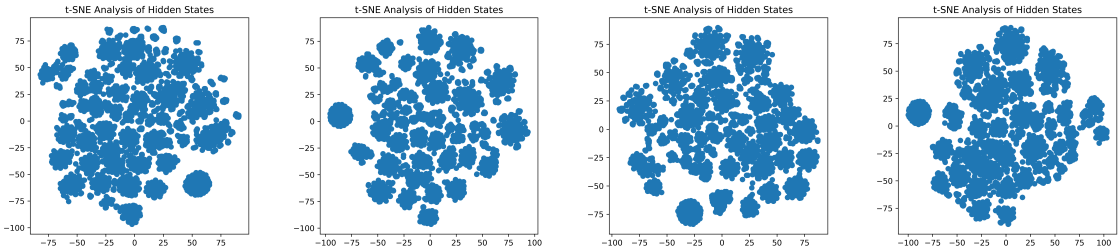
To validate the initial perplexity value of 30, additional experiments were conducted to test different settings, in particular 3, 5, 30 and 50. Figures 5.3 and 5.4 show our results for Beethoven's and Mozart's pieces presented above. The diagrams for the lower values of perplexity (Figures 5.3(a,b) and 5.4(a,b) for perplexity 3 and 5, respectively) show a single cloud of points that are evenly distributed, not providing meaningful information. In contrast, higher values of perplexity (Figures 5.3(c,d) and 5.4(c,d) for perplexity 30 and 50, respectively) show a cloud of well-defined clusters of various sizes. Similar results were obtained for the rest of the files in the data set.

The plots generated with perplexity 30, which is the default t-SNE setting and the basic recommendation by the authors of the t-SNE technique [32], provide the most meaningful results, and very similar to those with perplexity 50. This fact suggested that choosing perplexity 30 for analyzing MIDI files would be the most promising research path. As a consequence, we fixed perplexity 30 for analyzing hidden states of MIDI files. All results shown in the rest of the document are obtained with that setting when applied to hidden states of MIDI files processed by the MIDIBERT model.

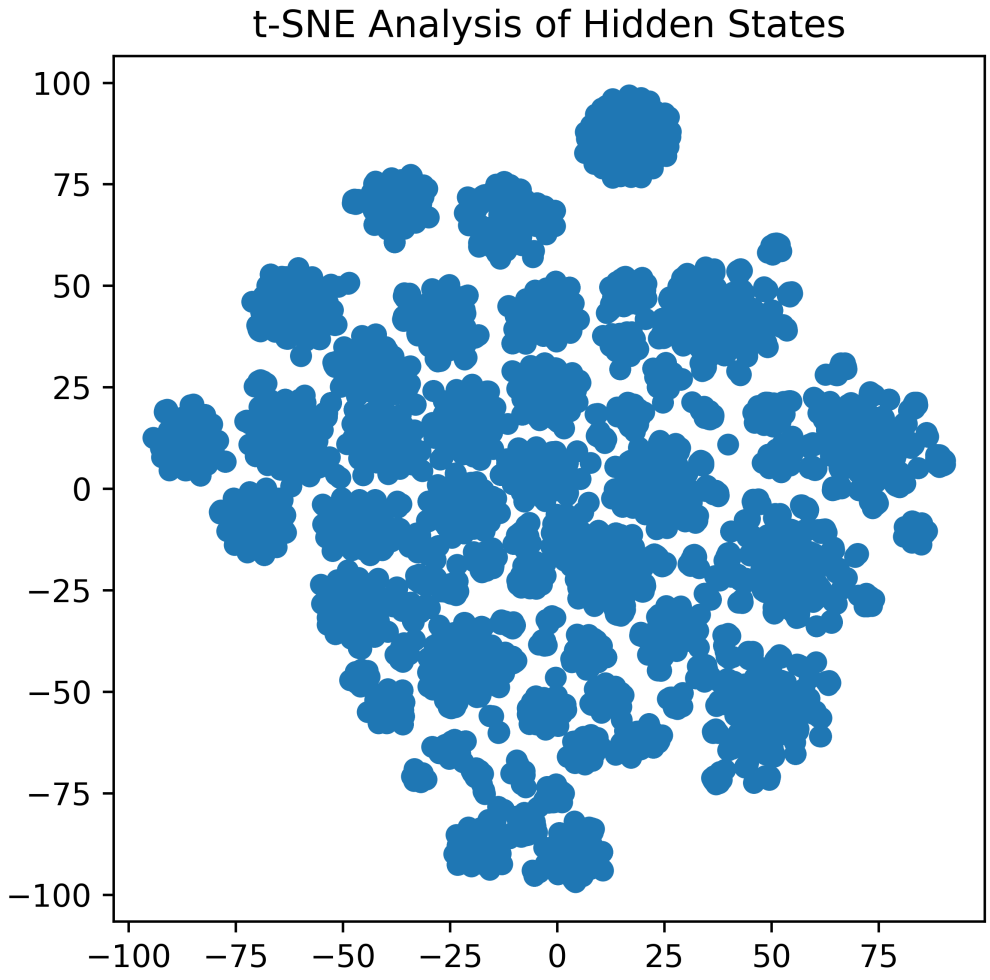
Let us briefly summarize the process up to this point. Each file is forward passed individually through BERT, obtaining a 3-dimensional representation as follows: (n batches, 512, 768). This representation is reshaped so that each token is associated with one set of hidden states: $(number\_of\_batches \times 512, 768)$. Then, t-SNE is performed, obtaining embedded data of size $(number\_of\_batches \times 512, 2)$. Finally, this data is saved in a csv file for future use.

5.2 t-SNE Analysis of Hidden States of a MIDI File Data Set

For the next step, a Python script was composed. This script finds the largest number of points for any of the files. In our case, it was 8704 points. Please note that we are working with multiples of 512, so this corresponds to a musical piece that had to be divided into 17 batches. Then, all smaller sizes were padded with zeros so that dimensions were matched to perform t-SNE again. Since the 3D vector now was of size $(number\_of\_MIDIs, 8704, 2)$, in order for it to be accepted by t-SNE (requiring two-dimensional data), the vector was reshaped as $(number\_of\_MIDIs, 8704 \times 2)$ so that each point in t-SNE accounted for one and only one piece of music.

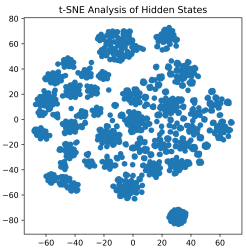


(a) Pretrain model (b) Melody model (c) Velocity model (d) Composer model

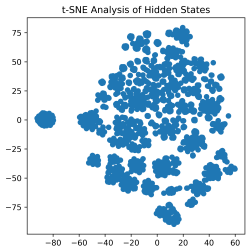


(e) Emotion model

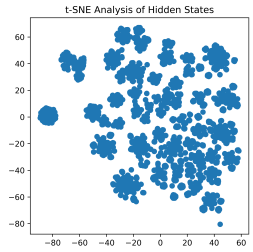
Figure 5.1: t-SNE analysis of hidden states for Beethoven Sonata No. 5 C minor, Opus 10/1 (1798), first movement



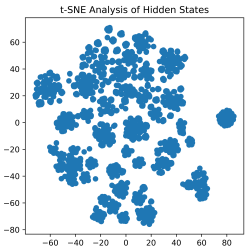
(a) Pretrain model



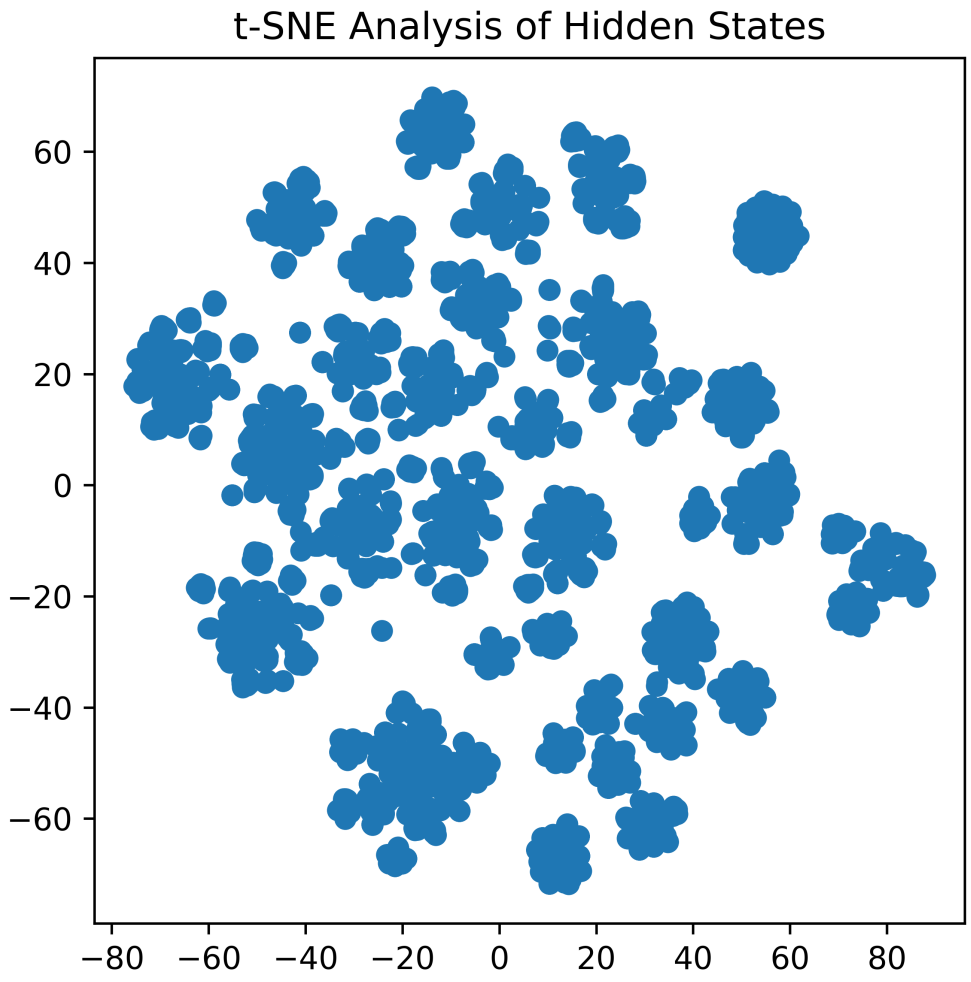
(b) Melody model



(c) Velocity model

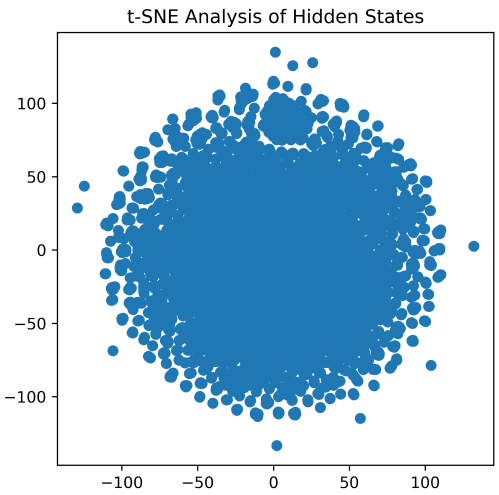


(d) Composer model

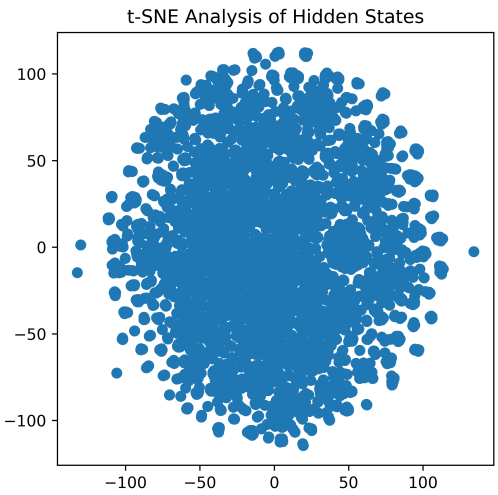


(e) Emotion model

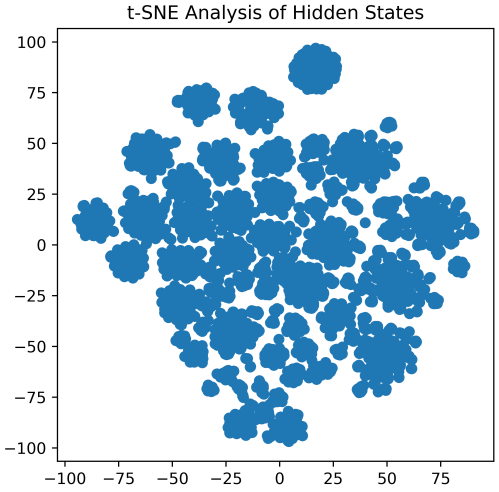
Figure 5.2: t-SNE analysis of hidden states for Mozart Sonata No. 8 D major, KV 311 (1777), first movement



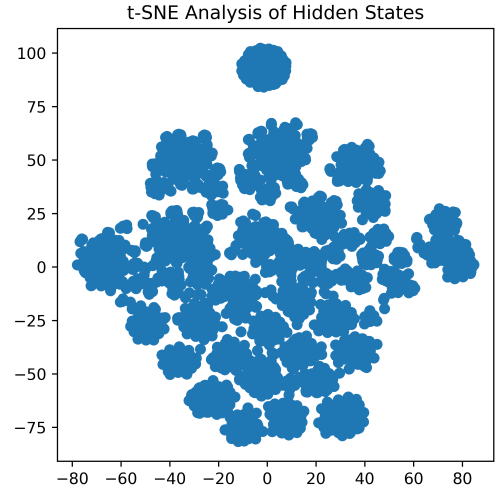
(a) Perplexity = 3



(b) Perplexity = 5

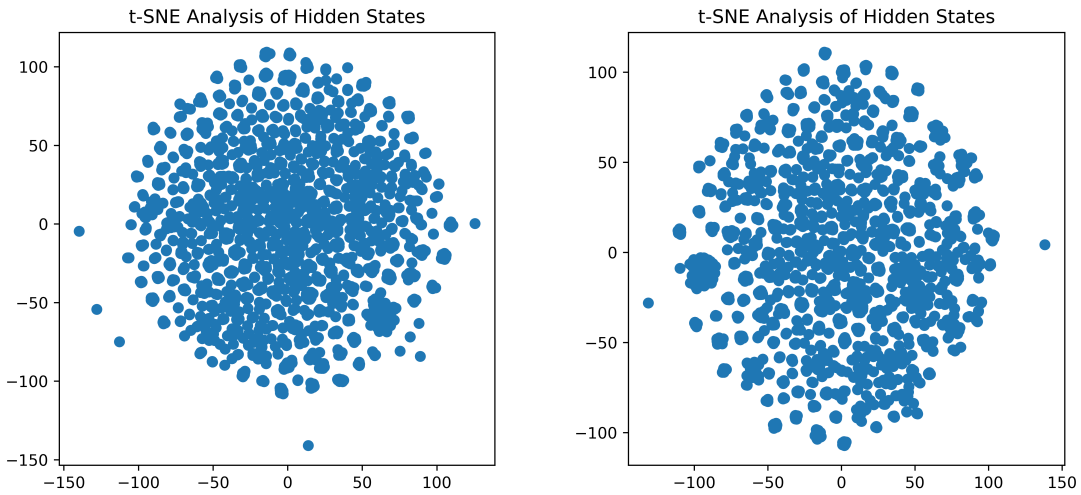


(c) Perplexity = 30



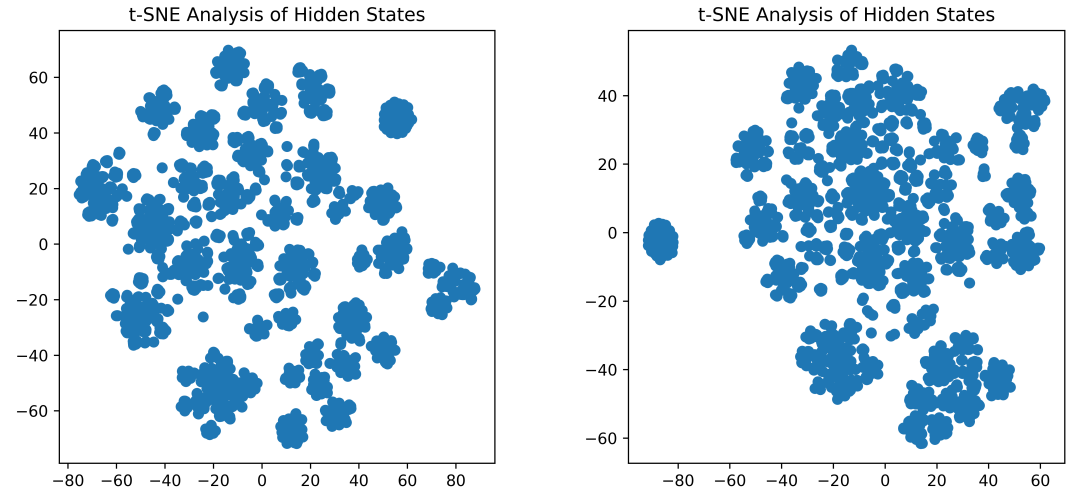
(d) Perplexity = 50

Figure 5.3: t-SNE analysis of hidden states for various settings of perplexity parameter. Beethoven Sonata No. 5 C minor, Opus 10/1 (1798), first movement



(a) Perplexity = 3

(b) Perplexity = 5



(c) Perplexity = 30

(d) Perplexity = 50

Figure 5.4: t-SNE analysis of hidden states for various settings of perplexity parameter. Mozart Sonata No. 8 D major, KV 311 (1777), first movement

Following the same steps as before, a good perplexity value had to be found. Values of 3, 5, 30 and 50 were tested, as shown in 5.5. However, some outliers appeared. In the plots for perplexity 3 and 5, they can be seen clearly, evenly spread out surrounding the rest of the points. In the plots for perplexity 30 and 50, they can be seen as the few points below the concentrated group of points. It is important to notice that different values of perplexity generate different amounts of outliers. As a first measure, the largest size allowed was set to 3000 points, with no difference in the observed plots. Finally, the outliers shown in the perplexity 3 plot were removed in order to ensure a meaningful result, and t-SNE was conducted again. The removed files are indicated in 7.1, for reference.

Again t-SNE's perplexity had to be tuned. The same values as before were tried, and the results are shown in 5.6. Surprisingly, this time around, the best value was deemed to be 3, under the lower limit of the recommended values to try. Values of 5 and 30 still produced outliers, while the central group of points seemed to have no patterns whatsoever. There were no outliers for perplexity 50, but the points were spread out in some even manner. For this reason, the value of 3 was validated as the one to use. One reason for this behavior could be the application of t-SNE twice on the same set of data. Firstly, on each file and, secondly, on all the files together.

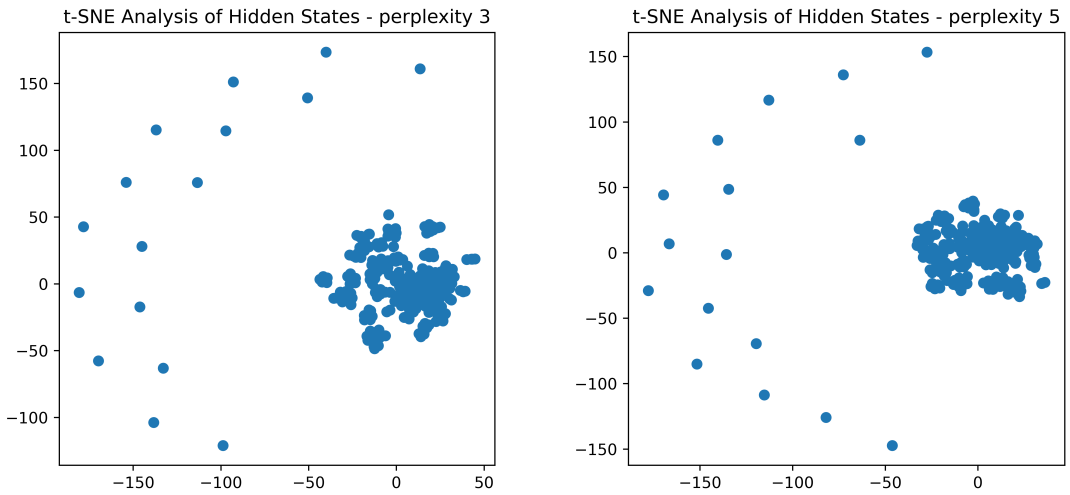
5.3 K-means Analysis of Hidden States of a MIDI File Data Set

As we explained before, the next step comprised selecting random samples that represented the data for later analysis. Since 271 files remained after removing the outliers, 25 random files, accounting for a bit under 10%, were sampled before performing the clustering. Those files were researched and manually annotated according to their full name, composer, year of composition (where there was a doubt or a period of time, the later year was selected), key and dissonance ratio, as shown in 5.1 and 5.2.

As a representation of the general character or a piece of music, the dissonance ratio was selected. The dissonance ratio is computed as the time dissonant intervals are played divided by the time consonant intervals are played. Dissonance ratio has been calculated on MIDI files by jSymbolic [33]. This tool was taken into account in earlier stages of the thesis's development but was finally not selected for the final iteration. However, it provides a fast and simple way of calculating meaningful features of MIDI files. Tests were made, and this ratio provided distinct values for different pieces. As shown in 5.2, most pieces have a value between 0.1 and 0.3, however, some of them go as high as 0.6.

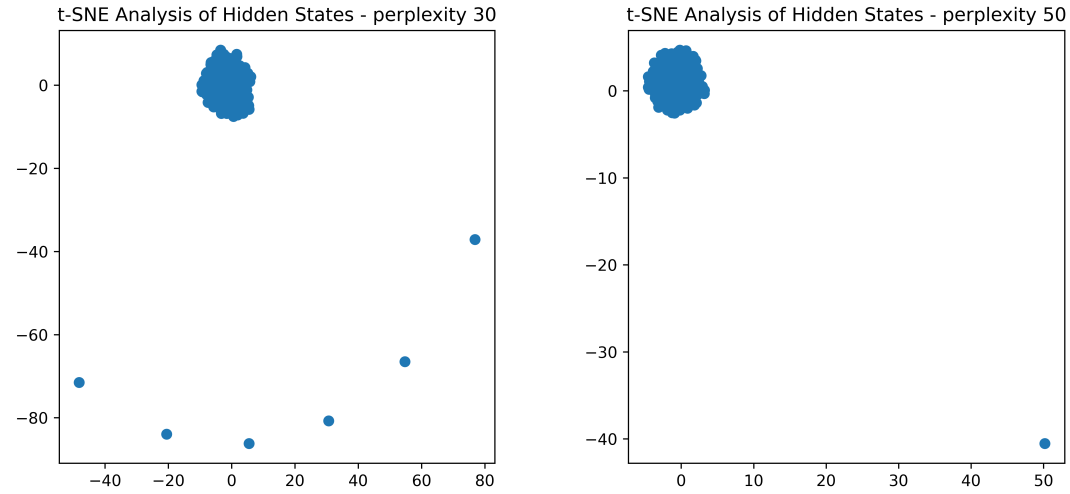
Finally, clustering was performed on the resulting data. Consider that this data is the result of performing t-SNE on each file with perplexity of 30, padding with zeros to fit the larger vectors and performing t-SNE again on all files with perplexity 3. k-means requires the number of clusters to be found a priori. Figure 5.7 and Figure 5.8 show the results of applying k-means clustering for cluster sizes ranging from 2 to 9.

Unlike in supervised learning, where we possess a definitive reference for assessing the model's effectiveness, clustering analysis lacks a definitive evaluation metric that can be employed to assess the results yielded by various clustering algorithms. Additionally,



(a) Perplexity = 3

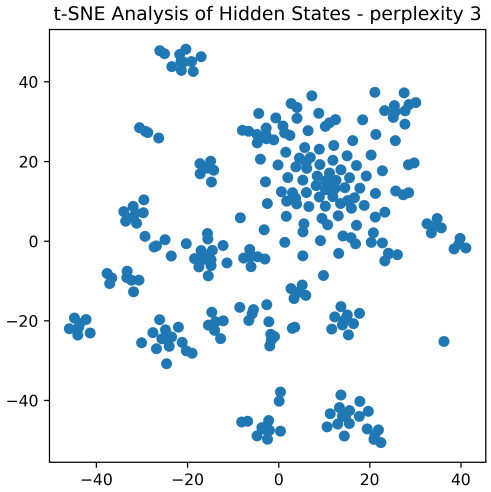
(b) Perplexity = 5



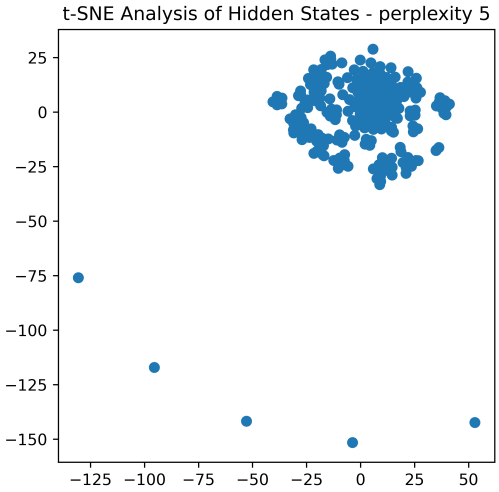
(c) Perplexity = 30

(d) Perplexity = 50

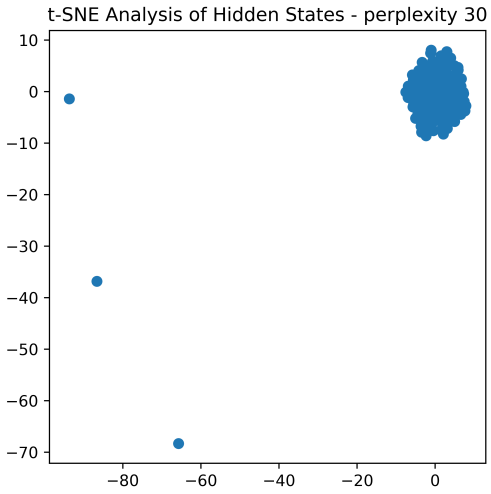
Figure 5.5: t-SNE analysis of hidden states for various settings of perplexity parameter. Full data set



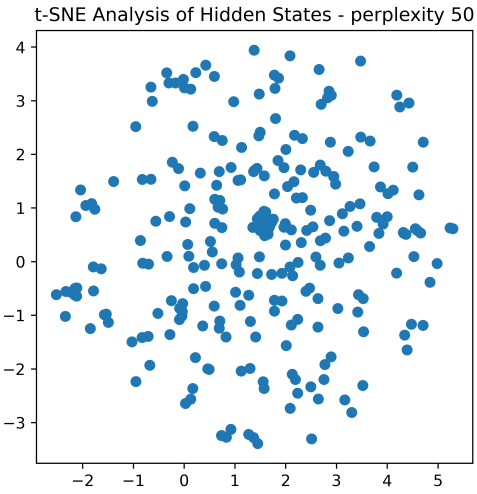
(a) Perplexity = 3



(b) Perplexity = 5

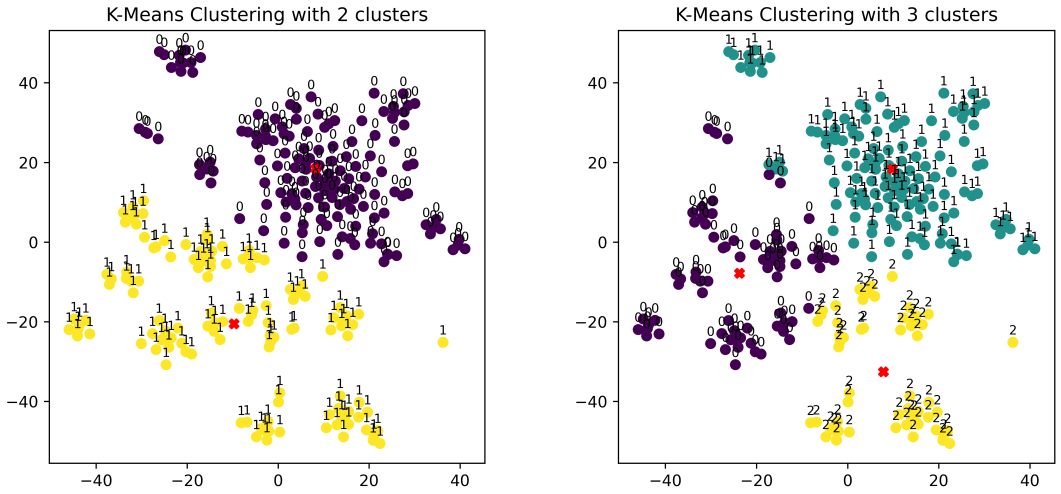


(c) Perplexity = 30



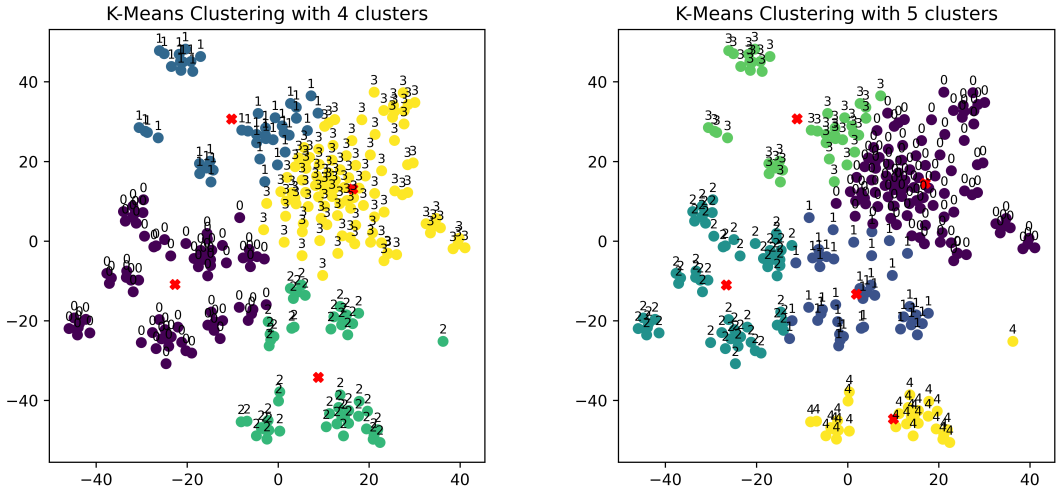
(d) Perplexity = 50

Figure 5.6: t-SNE analysis of hidden states for various settings of perplexity parameter. Full data set where outlier points are deprecated



(a) 2 clusters

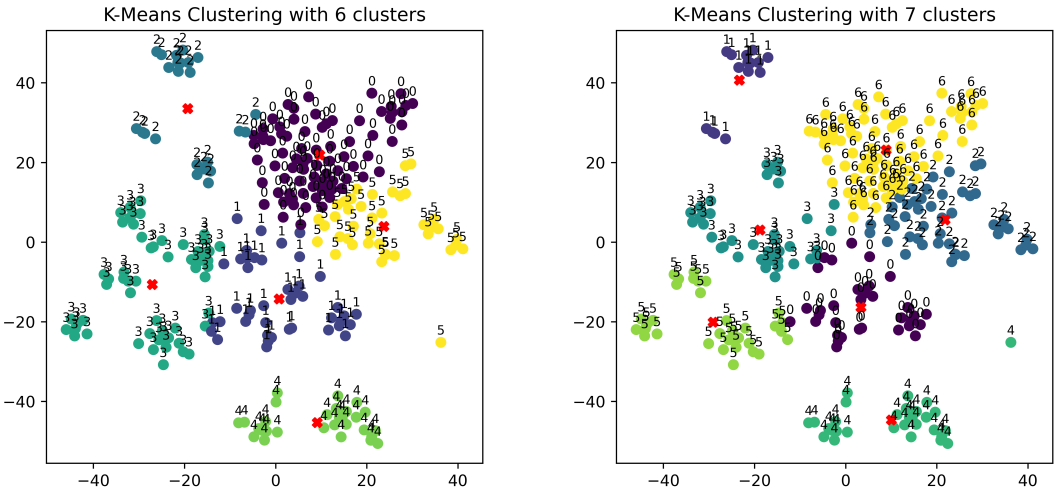
(b) 3 clusters



(c) 4 clusters

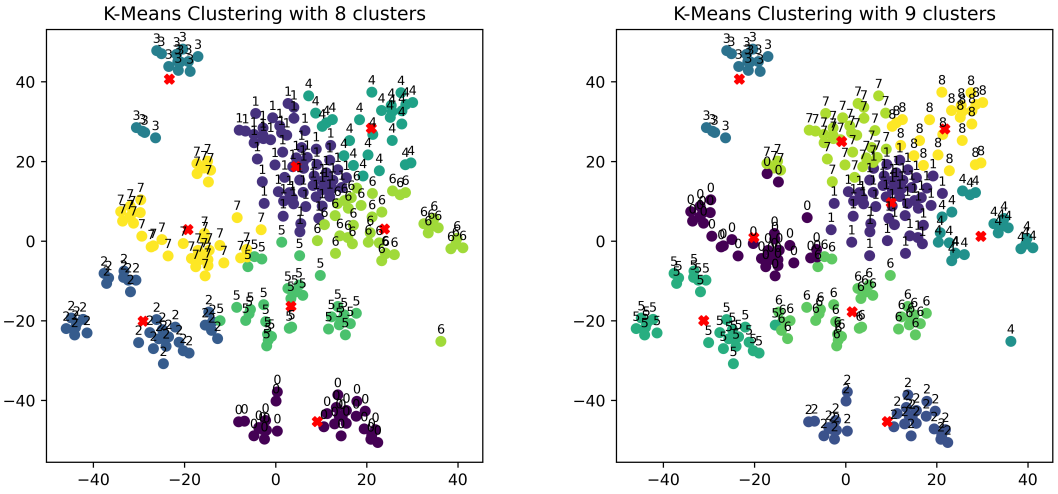
(d) 5 clusters

Figure 5.7: k-Means analysis for 2 to 5 clusters



(a) 6 clusters

(b) 7 clusters



(c) 8 clusters

(d) 9 clusters

Figure 5.8: k-Means analysis for 6 to 9 clusters

because k-means necessitates the manual specification of the number of clusters (k) and doesn't learn this value from the data, determining the optimal cluster count is inherently ambiguous for any given problem.

The elbow method is a technique to determine the optimal number of clusters (k) in k-means clustering. It plots the sum of squared distance for different values of k and looks for the point where the curve bends sharply, like an elbow. This point indicates that adding more clusters does not improve the model significantly.

Figure 5.9 shows a graph of the sum of squared distance versus the number of clusters for our data. The figure does not show a clear elbow. The curve experiences a steep decrease for 2 clusters and a transition of decreasing gradient until 4 clusters. For a higher number of clusters, the curve is monotonically decreasing and starts flattening out. From this data, values of k in the range from 2 to 5 seem the most appropriate.

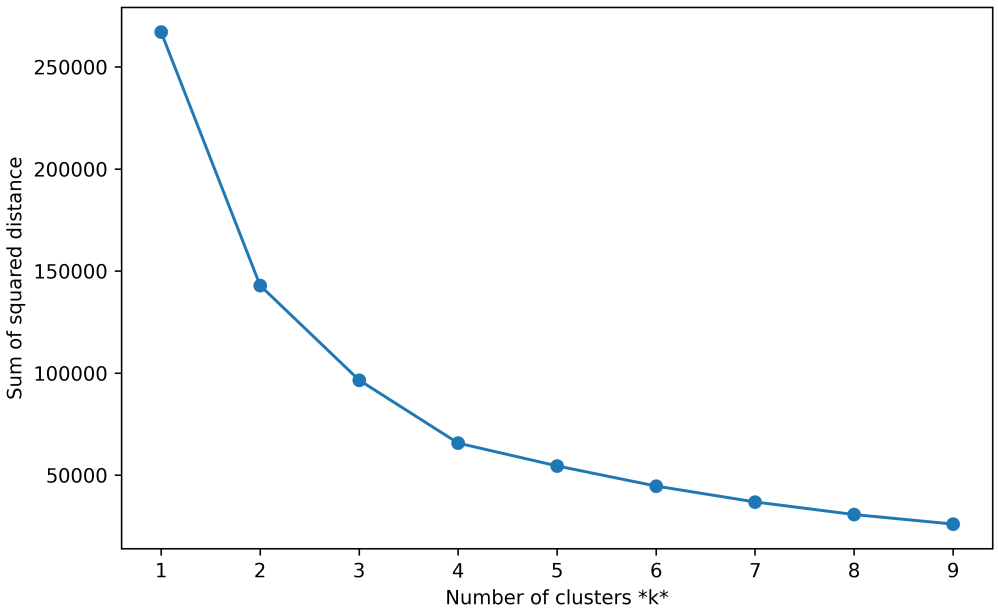


Figure 5.9: Elbow method visualization.

5.4 Analysis of results

The last step in this project is to locate the sampled data (listed in Table 5.1 and Table 5.2) on the clustered plots and review the results. In the appendix, we provide Table 7.2 which contains the assigned group for every piece for every number of clusters. In this table, we have highlighted in red color the members of the control group mentioned above. Now, an analysis of each number of clusters from 2 to 5 will be conducted using the annotated sample.

For two clusters, our control group is balanced. We find 13 samples in group 0 and 12

in group 1. The average dissonance ratio for group 0 is 0.24, while the ratio for group 1 is 0.27. However, the piece with the most dissonance ratio *Pictures at an exhibition - Samuel Goldenberg and Schmuyle* is classified in group 0. The second and third with the highest dissonance ratio, *Schumann Op. 15 - Blindman's Buff* and *Pictures at an exhibition - Bydlo*. The average year of composition for group 1 is 22 years earlier than that of group 0 (1828 against 1850). However, the most recent piece, *Brahms Op. 116 No. 2*, composed in 1892, is classified in group 1. For the most repeated composer, 3 out of 5 of Chopin's works are classified in group 0 and 2 in group 1. However, both of Beethoven's works are classified in the same group, and the same is true for Schubert. The rest of the repeating composers have their works classified as different groups. In our control group, there are 11 major keys and 14 minor keys. The 11 major keys are classified 7 in group 0 and 4 in group 1. This means that 6 pieces in minor keys are classified in group 0 and 8 are classified in group 1. As evidenced by the analysis, there seem to be no clear criteria for the classification of two clusters. As such, this value for k is deemed as too small.

For three clusters, we find 9 pieces classified as Group 0, 4 as Group 1 and 12 as Group 2. It is not very balanced. Group 0 possesses an average dissonance ratio of 0.27, Group 1 of 0.18 and Group 2 of 0.27. Group 1 seems to group those pieces with higher consonance. Group 0 has an average year of composition of 1847, Group 1 of 1855 and Group 2 of 1828. Group 2 seems to generally retain works of earlier composition, while still taking the latest work from 1892. Chopin's works are distributed between Group 0 and Group 2. Beethoven's are both in the same group, same as Schubert's. The rest of repeating composers are again split between groups. Out of the 11 major keys we find now 5 in Group 0, 2 in Group 1 and 4 in Group 2. Meaning that out of the 14 minor keys we find 4 in Group 0, 2 in Group 1 and 8 in group 2. Upon a more general inspection of the clusters, for our control group, former Group 1 has remained stable and become Group 2. Former Group 0 has been divided into Group 0 and new Group 1. This split seems to shed more light on the distribution of pieces, since dissonance ratio seems to have split into Group 0 and Group 1. However, it is still not enough to understand the categorization of pieces.

For four clusters, we find 8 pieces classified as Group 0, 3 as Group 1, 4 as Group 2 and 10 as Group 3. The average dissonance ratio for Group 0 is 0.27, for Group 1 is 0.2, for Group 2 is 0.26 and for Group 3 is 0.25. This measurement seems to have been stabilized between most groups, while Group 1 possesses the lowest value. The average composition year is 1843 for Group 0, 1860 for Group 1, 1769 for Group 2 and 1841 for Group 3. Cluster 2 groups the earliest works with a big margin. Chopin's works are distributed between Group 0 and Group 3. Surprisingly, Beethoven's works have split, while Schubert's have remained in the same group. The rest of repeating composers remain split. Out of the 11 major keys, we find now only 2 in Group 0, 1 in Group 1, 2 in Group 2 and 6 in Group 3. Out of the 14 minor keys, we find 6 in Group 0, 2 in Group 1, 2 in Group 2 and 4 in Group 3. In this split we clearly have Group 1 with the least dissonance ratio, Group 2 with the earliest works, Group 0 with most minor keys and Group 3 with most major keys. Upon a more general evaluation, all former members of Group 0 have become Group 3, with the addition of one coming from former Group

1, remerging in some way. All other members from former Group 1 have remained as Group 1. 8 of the former members of Group 2 have become Group 0, while the other 4 have remained Group 2.

For five clusters, we find 2 pieces in Group 0, 8 in Group 1, 8 in Group 2, 4 in Group 3 and 3 in Group 4. The average dissonance ratio for Group 0 is 0.24, for Group 1 is 0.27, for Group 2 is 0.28, for Group 3 is 0.26 and for Group 4 is 0.14. Group 4 reunites those pieces with more consonance, while it is spread out evenly in other groups. The average composition year is 1864 for Group 0, 1843 for Group 1, 1850 for Group 2, 1798 for Group 3 and 1840 for Group 4. Group 3 retains the earlier works, but not with as big of a margin as before. Chopin's works are distributed between Group 1 and Group 2. Beethoven's remain split into Group 1 and Group 3, and surprisingly Schubert's have been split as well into Group 2 and Group 4. All other repeating composers remain split into groups. Out of the 11 major keys, we find 1 in Group 0, 2 in Group 1, 4 in Group 2, 2 in Group 3 and 2 in Group 4. Out of the 14 minor keys, we find 1 in Group 0, 6 in Group 1, 4 in Group 2, 2 in Group 3 and 1 in Group 4. In this split, we also have one cluster with the least dissonance with some margin, one cluster with the earlier works and one cluster with the most minor keys. However, there is no major key dominance in any cluster. This is a distinction that has probably been lost due to too many clusters. Upon a more general evaluation, former Group 0 has become Group 1, former Group 1 has split into Group 0 and Group 4, former Group 2 has become Group 3 and former Group 3 has become Group 2 with two exceptions.

It is interesting to note that, generally, pieces that were classified together remained together through all the iterations analyzed. The following works belonged in the same group respectively 0, 0, 3 and 2: Chopin Op. 10 No. 5, Tchaikovsky's The seasons - March, Mussorgsky's Pictures at an exhibition - Samuel Goldenberg and Schmuyle, Chopin Op. 23, Mendelssohn Op. 30 No. 5, Chopin Op. 35 1st mov, Debussy's Suite bergamasque - Prélude and Schubert's Piano Sonata in Bb major 4th mov. The following works also remained together for groups 1, 2, 0 and 1: Brahms Op. 116 No. 2, Mussorgsky's Pictures at an exhibition - Bydlo, Chopin Op. 10 No. 12, Beethoven Op. 22 3rd mov, Schumann Op. 15 - Blindman's Buff, Burgmüller Op. 109 - Agitato, Chopin Op. 27 No. 1 and Mendelssohn Op. 19 No. 1. Other works that remained together are: Albeniz Op. 165 No. 1 and Schumann Op. 15 - Curious Story for 0, 1, 1 and 0; Bach's Prelude and Fugue in C minor BWV 847, Beethoven Op. 22 1st mov, Sonata No. 12 F major 3rd mov. Besides the metrics explained in the previous paragraphs, there is no obvious relationship between these pieces. However, this group stability proves that k-means was stable and generally grouped the same pieces together, and that indeed BERT places them in the same category.

It is also interesting to note that, during the analysis, it was observed that BERT differentiated each cluster according to a different metric. For example, one cluster would have the earlier works, while the rest would have a much closer average of years of composition. Or one cluster would have a significantly lower dissonance ratio while the others would have a much closer average. This might suggest that the metrics according to which BERT decides some pieces are similar and some are different are still under the surface, which we just started scraping.

To sum up, according to our control group and the metrics chosen for evaluation, clusterings with 3, 4 and 5 clusters have provided good results. Clustering with only 2 groups did not show any difference between them. With 3 clusters, a difference could be told. With 4 clusters, they could be labeled. With 5 clusters, some meaning was lost. Thus, we find the optimal number of clusters for our case to be 4.

File name	Piece name	Year	Composer
chpn_op10_e05	Chopin Op. 10 No. 5	1830	Chopin
br_im2	Brahms Op. 116 No. 2	1892	Brahms
muss_4	Pictures at an exhibition - Bydlo	1874	Mussorgsky
liz_et3	Etude No. 3 La campanella	1838-1851	Liszt
chpn_op10_e12	Chopin Op. 10 No. 12	1831	Chopin
ty_maerz	The seasons - March	1876	Tchaikovsky
muss_6	Pictures at an exhibition - Samuel Goldenberg and Schmuyle	1874	Mussorgsky
alb_esp1	Albeniz Op. 165 No. 1	1890	Albéniz
beethoven_opus22_3	Beethoven Op. 22 3rd mov	1800	Beethoven
scn15_3	Schumann Op. 15 - Blindman's Buff	1838	Schumann
chpn_op23	Chopin Op. 23	1831	Chopin
burg_spinnerlied	Burgmüller Op. 109 - Spinning Song	1844	Burgmüller
bach_847	Prelude and Fugue in C minor BWV 847	1722-1723	Bach
burg_agitato	Burgmüller Op. 109 - Agitato	1844	Burgmüller
mendel_op30_5	Mendelssohn Op. 30 No. 5	1834	Mendelssohn
chpn_op27_1	Chopin Op. 27 No. 1	1836	Chopin
mendel_op19_1	Mendelssohn Op. 19 No. 1	1830	Mendelssohn
beethoven_opus22_1	Beethoven Op. 22 1st mov	1800	Beethoven
scn15_2	Schumann Op. 15 - Curious Story	1838	Schumann
schubert_D850_4	Piano Sonata in D major 4th mov	1825	Schubert
chpn_op35_1	Chopin Op. 35 1st mov	1839	Chopin
deb_prel	Suite bergamasque - Prélude	1890	Debussy
mz_332_3	Sonata No. 12 F major 3rd mov	1783	Mozart
schub_d960_4	Piano Sonata in Bb major 4th mov	1828	Schubert
grieg_album	Grieg Op. 47 No. 2	1884	Grieg

Table 5.1: Annotated data set test group. Part 1.

File name	Key	Dissonance ratio	Annotations
chpn_op10_e05	D flat major	0.2305	
br_im2	A minor	0.1584	
muss_4	G sharp minor	0.4932	
liz_et3	G sharp minor	0.1277	
chpn_op10_e12	C minor	0.221	
ty_maerz	G minor	0.2513	
muss_6	B flat minor	0.6633	Phrygian scale and augmented seconds used
alb_esp1	A minor	0.1694	Gipsy scale
beethoven_opus22_3	B flat major	0.2231	
scn15_3	B minor	0.5024	
chpn_op23	G minor	0.2637	
burg_spinnerlied	D major	0.1256	
bach_847	C minor	0.2543	
burg_agitato	E minor	0.2115	
mendel_op30_5	D major	0.2367	
chpn_op27_1	C sharp minor	0.2165	
mendel_op19_1	E major	0.1559	
beethoven_opus22_1	B flat major	0.1561	
scn15_2	D major	0.3075	
schubert_D850_4	D major	0.166	
chpn_op35_1	B flat minor	0.245	
deb_prel	F major	0.2079	
mz_332_3	F major	0.2359	
schub_d960_4	B flat major	0.123	
grieg_album	E minor	0.3802	

Table 5.2: Annotated data set test group. Part 2.

6 Conclusions

Carrying out this work has been particularly fruitful. Due to the multi-part nature of a work like this, with a literature review, an implementation and a final analysis, different skills were required, different mistakes were made, and different lessons were learned.

During the literature review, I learned that much more work than I thought was being carried out in the interdisciplinary domain of computer science and music. However, I also learned that there are even more still unexplored areas, more paths awaiting to be walked. Perhaps it is because I am a musician myself, and I consider this combination extremely interesting, but it was a great experience to learn about the work that had been made before mine, so I could stand on giants' shoulders. I know this work would have been impossible, or at least, of a much different nature, had it not been for the existence of Transformers and BERT and people that had thought about mixing them with music before me. But, as mentioned previously, there is still more work to do.

Making the implementation work was a particularly laborious part of the project. Dealing with code that is not yours is always a bit complex, and all the dependency problems that had to be solved could probably account for a thesis themselves. But we managed to make everything work smoothly and develop the "pipeline" that could take the MIDI files and finally turn them into those plots that we obtained. As a future work, it could be interesting to write a script that would run the whole process in one task. The consequences of using t-SNE twice on the same set of data should also be studied. Ideally, we would like to have worked directly with the hidden states. This could be done in future work as well if the hardware constraints are resolved.

Finally, we were able to show that music does indeed possess latent dimensions and BERT is able to capture them. Although we were not able to fully understand them, we were able to label the clusters computed. These labels were based on the dissonance ratio, the year the work was composed, the number of major keys and the number of minor keys. In future work, a fully annotated dataset should be used for a more thorough and complete examination of results, as well as datasets containing more and different genres of music. We are also aware of the constraints of BERT regarding music. Music is indeed sequential data, but the start and end of sequences are very hard to determine, sometimes even for trained professionals. If an improvement is provided in future work regarding the way MIDI or music sequences are processed by BERT or other models, we are sure that BERT's "understanding" of music would also improve.

To sum up, in this work a thorough literature review was presented, regarding the mix of both music and computer science, entire chapters were dedicated to explaining the model and the decision behind it, as well as our reference paper, its methodology and our methodology, a comprehensive explanation of experiments and their results was provided step by step and finally a meticulous and careful analysis was done on them.

7 General Addenda

Table 7.1 shows the complete data set pieces file names. The file names corresponding to pieces that were discarded during the clustering analysis, since they appeared as outliers in the graphs, are highlighted in red color.

Table 7.1: Data set file names

alb_esp1	brahms_opus1_4	chpn_op27_2	haydn_7_1	muss_5	schumm-2
alb_esp2	br_im2	chpn_op33_2	haydn_7_2	muss_6	schumm-3
alb_esp3	br_im5	chpn_op33_4	haydn_7_3	muss_7	schumm-4
alb_esp4	br_rhap	chpn_op35_1	haydn_8_1	muss_8	schumm-5
alb_esp5	burg_agitato	chpn_op35_2	haydn_8_2	mz_311_1	schumm-6
alb_esp6	burg_erwachen	chpn_op35_3	haydn_8_3	mz_311_2	schum_abegg
alb_se1	burg_geschwindigkeit	chpn_op35_4	haydn_8_4	mz_311_3	schu_143_1
alb_se2	burg_gewitter	chpn_op53	haydn_9_1	mz_330_1	schu_143_2
alb_se3	burg_perlen	chpn_op66	haydn_9_2	mz_330_2	schu_143_3
alb_se4	burg_quelle	chpn_op7_1	haydn_9_3	mz_330_3	scn15_1
alb_se5)	burg_spinnerlied	chpn_op7_2	hay_40_1	mz_331_1	scn15_10
alb_se6	burg_sylphen	chp_op18	hay_40_2	mz_331_2	scn15_11
alb_se7	burg_trennung	chp_op31	islamei	mz_331_3	scn15_12
alb_se8	chpn-p1	debussy_cc_1	liz_et1	mz_332_1	scn15_13
appass_1	chpn-p10	debussy_cc_2	liz_et2	mz_332_2	scn15_2
appass_2	chpn-p11	debussy_cc_3	liz_et3	mz_332_3	scn15_3
appass_3	chpn-p12	debussy_cc_4	liz_et4	mz_333_1	scn15_4
bach_846	chpn-p13	debussy_cc_6	liz_et5	mz_333_2	scn15_5
bach_847	chpn-p14	deb_menu	liz_et6	mz_333_3	scn15_6
bach_850	chpn-p15	deb_prel	liz_et_trans4	mz_545_1	scn15_7
beethoven_hammerklavier_1	chpn-p16	elise	liz_et_trans5	mz_545_2	scn15_8
beethoven_hammerklavier_2	chpn-p17	gra_esp_2	liz_et_trans8	mz_545_3	scn15_9
beethoven_hammerklavier_3	chpn-p18	gra_esp_3	liz_liebestraum	mz_570_1	scn16_1
beethoven_hammerklavier_4	chpn-p19	gra_esp_4	liz_rhap02	mz_570_2	scn16_2
beethoven_les_adieux_1	chpn-p2	grieg_album	liz_rhap09	mz_570_3	scn16_3
beethoven_les_adieux_2	chpn-p20	grieg_berceuse	liz_rhap10	pathetique_1	scn16_4
beethoven_les_adieux_3	chpn-p21	grieg_brooklet	liz_rhap12	pathetique_2	scn16_5
beethoven_opus10_1	chpn-p22	grieg_elfentanz	mendel_op19_1	pathetique_3	scn16_6
beethoven_opus10_2	chpn-p23	grieg_halling	mendel_op19_2	schubert_D850_1	scn16_7
beethoven_opus10_3	chpn-p24	grieg_kobold	mendel_op19_3	schubert_D850_2	scn16_8
beethoven_opus22_1	chpn-p3	grieg_march	mendel_op19_4	schubert_D850_3	scn68_10
beethoven_opus22_2	chpn-p4	grieg_once_upon_a_time	mendel_op19_5	schubert_D850_4	scn68_12
beethoven_opus22_3	chpn-p5	grieg_spring	mendel_op19_6	schubert_D935_1	ty_april
beethoven_opus22_4	chpn-p6	grieg_voeglein	mendel_op30_1	schubert_D935_2	ty_august
beethoven_opus90_1	chpn-p7	grieg_waechter	mendel_op30_2	schubert_D935_3	ty_dezember
beethoven_opus90_2	chpn-p8	grieg_walzer	mendel_op30_3	schubert_D935_4	ty_februar
bor_ps1	chpn-p9	grieg_wanderer	mendel_op30_4	schub_d760_1	ty_januar
bor_ps2	chpn_op10_e01	grieg_wedding	mendel_op30_5	schub_d760_2	ty_juli
bor_ps3	chpn_op10_e05	grieg_zwerge	mendel_op53_5	schub_d760_3	ty_juni
bor_ps4	chpn_op10_e12	haydn_33_1	mendel_op62_3	schub_d760_4	ty_maerz
bor_ps5	chpn_op23	haydn_33_2	mendel_op62_4	schub_d960_2	ty_mai
bor_ps6	chpn_op25_e1	haydn_33_3	mendel_op62_5	schub_d960_3	ty_november
bor_ps7	chpn_op25_e11	haydn_35_1	mond_1	schub_d960_4	ty_oktober
brahms_opus117_1	chpn_op25_e12	haydn_35_2	mond_3	schuim-1	ty_september
brahms_opus117_2	chpn_op25_e2	haydn_35_3	muss_1	schuim-2	waldstein_1
brahms_opus1_1	chpn_op25_e3	haydn_43_1	muss_2	schuim-3	waldstein_2
brahms_opus1_2	chpn_op25_e4	haydn_43_2	muss_3	schuim-4	waldstein_3
brahms_opus1_3	chpn_op27_1	haydn_43_3	muss_4	schumm-1	

Table 7.2 shows the complete data set pieces with their corresponding cluster according to the various cluster sizes tested in Section 5.3. The members of the control group used to draw conclusions are highlighted in red color.

Table 7.2: Data set clustering

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
alb_esp1	0	1	1	0	3	1	4	2
alb_esp2	0	1	1	0	3	1	4	2
alb_esp3	0	0	3	2	0	5	7	6
alb_esp4	1	2	0	1	1	2	5	8
alb_esp5	0	0	3	2	0	0	2	1
alb_esp6	1	2	0	1	4	4	1	8
alb_se1	0	0	3	2	0	0	2	1
alb_se2	0	1	1	0	3	1	4	2
alb_se3	0	1	1	0	3	1	4	2
alb_se4	1	2	0	1	4	4	6	3
alb_se6	1	2	0	1	4	4	6	3
alb_se7	1	2	0	1	4	4	1	8
alb_se8	1	2	0	1	1	2	5	7
appass_1	0	0	3	2	0	0	2	1
appass_2	0	0	3	4	2	3	0	5
appass_3	0	0	3	4	2	3	0	5
bach_846	0	0	3	2	2	3	7	6
bach_847	1	2	2	3	5	6	3	4
bach_850	1	2	0	1	4	4	6	3
beethoven_hammerklavier_1	1	2	0	1	1	2	5	7
beethoven_hammerklavier_2	0	0	3	2	0	0	2	1
beethoven_hammerklavier_3	0	0	3	2	0	5	7	6
beethoven_hammerklavier_4	1	2	0	1	1	2	5	8
beethoven_les_adieux_1	1	2	0	1	4	4	1	8
beethoven_les_adieux_2	0	0	3	2	0	0	2	1
beethoven_les_adieux_3	1	2	2	3	4	4	1	0
beethoven_opus10_2	1	0	2	3	5	6	3	4
beethoven_opus10_3	1	2	0	1	4	2	1	8
beethoven_opus22_1	1	2	2	3	4	4	1	0
beethoven_opus22_2	1	2	0	1	1	2	5	7
beethoven_opus22_3	1	2	0	1	4	4	6	3
beethoven_opus22_4	0	0	3	2	0	5	7	6
beethoven_opus90_1	0	0	3	2	0	5	7	6
beethoven_opus90_2	1	2	0	4	2	2	1	8

Continued on next page

Table 7.2 – continued from previous page

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
bor_ps1	0	0	3	2	0	0	2	1
bor_ps2	0	1	1	4	2	3	0	5
bor_ps3	0	0	3	2	0	5	7	6
bor_ps4	1	2	0	1	4	4	6	3
bor_ps5	1	2	0	1	1	2	5	7
bor_ps6	0	1	1	0	3	1	4	2
bor_ps7	1	2	0	1	4	2	6	8
brahms_opus117_1	0	0	3	2	0	5	7	6
brahms_opus117_2	0	0	3	2	0	5	7	6
brahms_opus1_1	0	0	3	2	0	0	2	1
brahms_opus1_2	0	0	3	2	0	5	7	6
brahms_opus1_3	1	2	0	1	1	2	5	7
brahms_opus1_4	1	2	2	3	5	6	3	4
br_im2	1	2	0	1	1	2	5	7
br_im5	0	1	1	0	3	1	4	2
br_rhap	0	1	1	0	3	1	4	2
burg_agitato	1	2	0	1	4	4	1	8
burg_erwachen	0	1	1	4	2	3	0	5
burg_geschwindigkeit	1	2	0	1	4	4	1	8
burg_gewitter	1	2	0	1	4	4	1	8
burg_perlen	1	2	0	1	4	2	1	8
burg_quelle	0	1	1	4	2	3	0	5
burg_spinnerlied	0	1	3	4	2	3	0	5
burg_sylphen	0	1	1	4	2	3	0	5
burg_trennung	0	0	3	2	0	5	7	6
chpn-p1	1	2	0	1	1	2	5	7
chpn-p10	1	2	2	3	5	4	1	0
chpn-p11	1	2	2	3	4	4	1	0
chpn-p12	0	1	1	4	2	3	0	5
chpn-p13	1	2	0	1	4	4	1	0
chpn-p14	1	2	0	1	4	4	1	0
chpn-p15	0	0	3	2	0	0	2	1
chpn-p16	1	2	2	3	5	6	3	4
chpn-p17	0	0	3	2	0	0	2	1
chpn-p18	0	1	1	0	3	1	4	2
chpn-p19	0	1	1	4	2	3	0	5
chpn-p2	1	2	2	3	4	4	1	0
chpn-p20	1	2	2	3	4	4	1	0
chpn-p21	1	2	0	1	1	2	5	8

Continued on next page

Table 7.2 – continued from previous page

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
chpn-p22	0	0	3	2	0	0	2	1
chpn-p23	0	0	3	2	0	0	2	1
chpn-p24	1	2	0	1	4	4	6	3
chpn-p3	0	1	1	0	3	1	4	2
chpn-p4	0	0	3	2	0	0	2	1
chpn-p5	1	2	0	1	1	2	5	8
chpn-p6	1	2	0	1	4	4	1	0
chpn-p7	1	2	2	3	4	4	1	0
chpn-p8	0	0	3	4	2	3	2	1
chpn-p9	1	2	0	1	4	4	1	0
chpn_op10_e01	0	1	1	4	2	3	0	5
chpn_op10_e05	0	0	3	2	0	0	2	1
chpn_op10_e12	1	2	0	1	1	2	5	7
chpn_op23	0	0	3	2	2	0	2	1
chpn_op25_e1	1	2	0	1	1	2	5	7
chpn_op25_e11	1	2	2	3	5	0	2	0
chpn_op25_e12	1	2	0	1	4	2	1	8
chpn_op25_e2	1	2	0	1	4	4	6	3
chpn_op25_e3	1	0	2	3	5	6	3	4
chpn_op25_e4	1	2	0	1	4	4	6	3
chpn_op27_1	1	2	0	1	4	4	1	8
chpn_op27_2	0	0	3	4	2	3	0	5
chpn_op33_2	1	2	0	1	4	4	1	8
chpn_op33_4	1	2	0	1	4	2	6	3
chpn_op35_1	0	0	3	2	0	5	7	6
chpn_op35_3	0	0	3	2	0	5	7	6
chpn_op35_4	1	2	0	1	4	4	1	8
chpn_op53	0	1	1	0	3	1	4	2
chpn_op7_1	1	2	0	1	4	4	6	3
chpn_op7_2	1	2	0	1	4	4	6	3
chp_op18	0	1	0	4	2	3	0	5
chp_op31	0	1	3	4	2	3	0	5
debussy_cc_1	1	2	0	1	4	4	6	3
debussy_cc_2	0	0	3	2	0	0	2	1
debussy_cc_3	0	0	3	2	2	3	7	6
debussy_cc_4	0	0	3	2	0	5	7	6
debussy_cc_6	0	1	1	4	2	3	0	5
deb_menu	0	0	3	2	0	5	7	6
deb_prel	0	0	3	2	0	5	7	6

Continued on next page

Table 7.2 – continued from previous page

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
elise	0	0	3	4	2	3	0	5
gra_esp_2	0	1	1	0	3	1	4	2
gra_esp_3	0	1	1	0	3	1	4	2
gra_esp_4	1	2	0	1	4	4	1	8
grieg_album	1	2	2	3	4	4	6	0
grieg_berceuse	1	2	0	1	1	2	5	7
grieg_brooklet	0	0	3	2	0	5	7	6
grieg_elfentanz	1	2	0	1	4	4	1	8
grieg_halling	1	2	0	1	1	2	5	7
grieg_kobold	1	2	0	1	4	4	6	3
grieg_march	1	2	0	1	4	4	6	3
grieg_once_upon_a_time	0	0	3	2	0	0	2	1
grieg_spring	0	0	3	2	0	5	7	6
grieg_voeglein	1	2	0	1	1	2	5	7
grieg_waechter	0	1	1	0	3	1	4	2
grieg_walzer	0	1	1	0	3	1	4	2
grieg_wanderer	1	2	0	1	1	2	1	8
grieg_wedding	0	1	1	0	3	1	4	2
grieg_zwerge	1	2	0	1	4	4	1	3
haydn_33_2	1	2	0	1	4	4	6	3
haydn_33_3	1	2	0	4	2	0	2	8
haydn_35_1	1	0	2	3	5	0	2	1
haydn_35_2	1	2	0	1	1	2	5	7
haydn_35_3	0	0	3	2	0	5	7	6
haydn_43_1	1	2	0	1	1	2	5	8
haydn_43_2	1	2	2	3	4	4	1	0
haydn_43_3	1	2	0	1	1	2	5	7
haydn_7_1	1	2	2	3	4	4	1	0
haydn_7_2	0	0	3	2	0	0	2	1
haydn_7_3	1	2	0	1	4	4	1	0
haydn_8_1	1	2	0	1	1	2	5	8
haydn_8_2	1	2	0	1	4	2	1	8
haydn_8_3	1	2	2	3	4	4	1	0
haydn_8_4	1	2	0	1	4	4	6	3
haydn_9_1	1	2	0	1	4	4	1	8
haydn_9_2	1	2	2	3	5	6	3	4
haydn_9_3	1	2	0	1	1	2	5	8
hay_40_1	1	2	2	3	5	0	2	0
islamei	0	1	1	0	3	1	4	2

Continued on next page

Table 7.2 – continued from previous page

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
liz_et1	1	2	0	1	1	2	5	7
liz_et2	0	1	1	4	2	3	0	5
liz_et3	0	1	1	4	2	3	0	5
liz_et4	0	0	3	2	0	5	7	6
liz_et5	1	0	2	3	5	6	3	4
liz_et6	0	1	1	0	3	1	4	2
liz_et_trans4	0	1	1	4	2	3	0	5
liz_et_trans5	1	2	2	3	5	6	3	4
liz_et_trans8	1	2	0	1	1	2	5	7
liz_liebestraum	1	2	2	3	5	4	1	0
liz_rhap02	1	0	2	3	5	6	3	4
liz_rhap09	0	0	3	2	0	5	7	6
liz_rhap10	0	1	1	0	3	1	4	2
liz_rhap12	1	2	2	3	5	6	3	4
mendel_op19_1	1	2	0	1	4	4	6	3
mendel_op19_2	1	2	0	1	1	2	5	7
mendel_op19_4	0	1	1	0	3	1	4	2
mendel_op19_5	1	2	2	3	4	4	1	0
mendel_op19_6	0	0	3	2	0	0	2	1
mendel_op30_1	1	2	0	1	1	2	6	3
mendel_op30_2	0	1	1	4	2	3	0	5
mendel_op30_3	1	2	0	1	4	4	1	8
mendel_op30_5	0	0	3	2	0	0	2	1
mendel_op53_5	0	1	1	4	2	3	0	5
mendel_op62_3	1	2	0	1	1	2	5	8
mendel_op62_4	0	1	1	0	3	1	4	2
mendel_op62_5	0	1	1	0	3	1	4	2
mond_1	1	2	2	3	5	6	3	4
mond_3	0	1	1	4	2	3	0	5
muss_1	0	0	3	2	0	5	7	6
muss_2	0	0	3	2	0	5	7	6
muss_3	1	2	0	1	1	2	5	7
muss_4	1	2	0	1	4	4	1	8
muss_5	0	0	3	4	2	3	0	5
muss_6	0	0	3	2	0	0	2	1
muss_7	0	0	3	2	0	5	7	6
muss_8	1	2	0	4	2	3	0	8
mz_311_1	1	2	0	1	1	2	5	8
mz_311_2	0	0	3	2	0	5	7	6

Continued on next page

Table 7.2 – continued from previous page

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
mz_311_3	0	1	1	0	3	1	4	2
mz_330_1	1	2	2	3	4	4	1	0
mz_330_2	1	2	0	1	4	4	1	8
mz_331_1	0	1	1	4	2	3	0	5
mz_331_2	0	0	3	2	0	5	7	6
mz_331_3	0	0	3	2	0	0	2	1
mz_332_1	1	2	0	4	1	2	5	8
mz_332_2	1	2	0	1	4	4	1	0
mz_332_3	1	2	2	3	5	4	1	0
mz_333_1	1	2	0	4	2	3	0	5
mz_333_3	1	2	2	3	5	0	2	0
mz_545_1	1	2	2	3	4	4	1	0
mz_545_2	1	2	0	1	4	4	1	8
mz_545_3	1	2	0	1	1	2	5	7
mz_570_1	1	2	2	3	4	4	6	0
mz_570_2	0	0	3	2	0	0	2	1
mz_570_3	1	2	2	3	4	4	1	0
pathetique_1	0	0	3	2	2	3	0	5
pathetique_2	0	1	1	4	2	3	0	5
pathetique_3	1	2	0	1	1	2	5	8
schubert_D850_2	0	0	3	2	0	5	7	6
schubert_D850_3	1	0	2	3	5	0	2	1
schubert_D850_4	0	0	3	4	2	3	0	5
schubert_D935_2	0	0	3	2	0	5	7	6
schubert_D935_3	0	0	3	2	0	0	2	1
schubert_D935_4	0	0	3	4	2	0	2	1
schub_d760_2	1	2	0	1	4	4	6	3
schub_d760_3	0	0	3	2	0	3	7	6
schub_d760_4	0	0	3	2	0	0	2	1
schub_d960_2	0	0	3	2	0	5	7	6
schub_d960_3	1	2	2	3	4	4	1	0
schub_d960_4	0	0	3	2	0	5	7	6
schuim-1	0	0	3	2	2	3	7	6
schuim-2	1	2	2	3	5	0	2	0
schuim-4	0	1	1	4	2	3	0	5
schumm-1	1	2	2	3	5	6	3	4
schumm-2	0	0	3	4	2	0	2	1
schumm-3	0	1	1	4	2	3	0	5
schumm-4	0	1	1	4	2	3	0	5

Continued on next page

Table 7.2 – continued from previous page

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
schumm-6	1	0	3	4	2	0	2	1
schum_abegg	0	0	3	2	0	5	7	6
schu_143_1	1	2	0	1	4	4	1	8
schu_143_2	1	2	0	4	1	2	5	8
schu_143_3	0	1	1	0	3	1	4	2
scn15_1	1	2	0	1	4	4	1	8
scn15_10	1	2	2	3	4	4	1	0
scn15_11	1	2	0	1	1	2	5	7
scn15_12	1	2	0	1	4	4	6	0
scn15_13	1	2	2	3	4	4	1	0
scn15_2	0	1	1	0	3	1	4	2
scn15_3	1	2	0	1	1	2	5	7
scn15_4	1	2	0	1	1	2	5	8
scn15_5	1	2	0	1	4	4	1	8
scn15_6	0	0	3	2	0	0	2	1
scn15_7	1	2	0	1	4	4	1	0
scn15_8	1	2	0	1	4	4	1	0
scn15_9	0	1	1	0	3	1	4	2
scn16_1	1	2	0	1	4	4	1	8
scn16_2	0	1	1	0	1	1	5	7
scn16_3	1	2	0	1	4	4	1	8
scn16_4	0	0	3	2	0	0	2	1
scn16_5	0	1	1	0	3	1	4	2
scn16_6	0	1	1	0	3	1	4	2
scn16_7	1	2	2	3	5	6	3	4
scn16_8	1	2	0	1	1	2	6	3
scn68_10	1	2	2	3	5	4	1	0
scn68_12	0	1	1	4	2	3	0	5
ty_april	0	1	1	4	2	3	0	5
ty_august	1	2	0	1	4	4	6	3
ty_dezember	1	2	0	1	1	2	5	8
ty_februar	1	2	0	1	4	4	1	8
ty_januar	0	0	3	2	0	5	7	6
ty_juli	1	2	2	3	5	6	3	4
ty_juni	0	1	1	0	3	1	4	2
ty_maerz	0	0	3	2	0	0	2	1
ty_mai	1	2	2	3	4	4	1	0
ty_november	1	2	2	3	5	6	3	4
ty_oktober	1	2	0	1	4	4	1	8

Continued on next page

Table 7.2 – continued from previous page

File Name	Number of clusters							
	2	3	4	5	6	7	8	9
ty_september	1	2	0	1	4	4	6	3
waldstein_2	1	2	0	1	1	2	5	8
waldstein_3	0	1	1	0	3	1	4	2

List of Figures

List of Tables

Bibliography

[1] N. Ndou, R. Ajoodha, and A. Jadhav. "Music Genre Classification: A Review of Deep-Learning and Traditional Machine-Learning Approaches". In: *2021 IEEE International IOT, Electronics and Mechatronics Conference (IEMTRONICS)*. 2021, pp. 1–6. DOI: 10.1109/IEMTRONICS52119.2021.9422487.

[2] H. Bahuleyan. "Music Genre Classification using Machine Learning Techniques". In: *CoRR* abs/1804.01149 (2018). arXiv: 1804.01149. URL: http://arxiv.org/abs/1804.01149.

[3] C. N. Silla, A. L. Koerich, and C. A. A. Kaestner. "A Machine Learning Approach to Automatic Music Genre Classification". In: *Journal of the Brazilian Computer Society* 14.3 (Sept. 1, 2008), pp. 7–18. DOI: 10.1007/BF03192561. URL: https://doi.org/10.1007/BF03192561.

[4] A. Ghildiyal, K. Singh, and S. Sharma. "Music Genre Classification using Machine Learning". In: *2020 4th International Conference on Electronics, Communication and Aerospace Technology (ICECA)*. 2020, pp. 1368–1372. DOI: 10.1109/ICECA49313.2020.9297444.

[5] Z. Qi, M. Rahouti, M. A. Jasim, and N. Siasi. "Music Genre Classification and Feature Comparison Using ML". In: *Proceedings of the 2022 7th International Conference on Machine Learning Technologies*. ICMLT '22. Rome, Italy: Association for Computing Machinery, 2022, pp. 42–50. ISBN: 9781450395748. DOI: 10.1145/3529399.3529407. URL: https://doi.org/10.1145/3529399.3529407.

[6] A. Elbir, H. Bilal Çam, M. Emre Iyican, B. Öztürk, and N. Aydin. "Music Genre Classification and Recommendation by Using Machine Learning Techniques". In: *2018 Innovations in Intelligent Systems and Applications Conference (ASYU)*. 2018, pp. 1–5. DOI: 10.1109/ASYU.2018.8554016.

[7] A. Roberts, J. Engel, S. Oore, and D. Eck, eds. *Learning Latent Representations of Music to Generate Interactive Musical Palettes*. 2018. URL: http://ceur-ws.org/Vol-2068/milc7.pdf.

[8] J. Shen, R. Wang, and H.-W. Shen. "Visual exploration of latent space for traditional Chinese music". In: *Visual Informatics* 4.2 (June 2020). ISSN: 2468-502X. DOI: 10.1016/j.visinf.2020.04.003. URL: https://www.osti.gov/biblio/1772109.

[9] A. Roberts, J. Engel, C. Raffel, C. Hawthorne, and D. Eck. *A Hierarchical Latent Vector Model for Learning Long-Term Structure in Music*. 2019. arXiv: 1803.05428 [cs.LG].

[10] A. Pati and A. Lerch. "Latent Space Regularization for Explicit Control of Musical Attributes". In: 2019. URL: https://api.semanticscholar.org/CorpusID:208230803.

[11] R. Yang, D. Wang, Z. Wang, T. Chen, J. Jiang, and G. Xia. *Deep Music Analogy Via Latent Representation Disentanglement*. 2019. arXiv: 1906.03626 [cs.SD].

[12] Z. Wang, Y. Zhang, Y. Zhang, J. Jiang, R. Yang, J. Zhao, and G. Xia. *PIANOTREE VAE: Structured Representation Learning for Polyphonic Music*. 2020. arXiv: 2008.07118 [eess.AS].

[13] G. Brunner, A. Konrad, Y. Wang, and R. Wattenhofer. *MIDI-VAE: Modeling Dynamics and Instrumentation of Music with Applications to Style Transfer*. 2018. arXiv: 1809.07600 [cs.SD].

[14] J. Jiang, G. G. Xia, D. B. Carlton, C. N. Anderson, and R. H. Miyakawa. "Transformer VAE: A Hierarchical Model for Structure-Aware and Interpretable Music Representation Learning". In: *ICASSP 2020 - 2020 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. 2020, pp. 516–520. DOI: 10.1109/ICASSP40776.2020.9054554.

[15] R. Yang, T. Chen, Y. Zhang, and G. Xia. *Inspecting and Interacting with Meaningful Music Representations using VAE*. 2019. arXiv: 1904.08842 [cs.SD].

[16] X. Liang, J. Wu, and J. Cao. *MIDI-Sandwich2: RNN-based Hierarchical Multi-modal Fusion Generation VAE networks for multi-track symbolic music generation*. 2019. arXiv: 1909.03522 [cs.LG].

[17] O. Cifka, A. Ozerov, U. Simsekli, and G. Richard. "Self-Supervised VQ-VAE for One-Shot Music Style Transfer". In: *ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*. IEEE, June 2021. DOI: 10.1109/icassp39728.2021.9414235. URL: https://doi.org/10.1109%2Ficassp39728.2021.9414235.

[18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964.

[19] C. Jin, T. Wang, S. Liu, Y. Tie, J. Li, X. Li, and S. Lui. "A Transformer-Based Model for Multi-Track Music Generation". In: vol. 11. 3. July 2020, pp. 36–54. URL: https://ideas.repec.org/a/igg/jmdem0/v11y2020i3p36-54.html.

[20] P. Dhariwal, H. Jun, C. Payne, J. W. Kim, A. Radford, and I. Sutskever. *Jukebox: A Generative Model for Music*. 2020. arXiv: 2005.00341 [eess.AS].

[21] A. C. Serra, A. J. G. Busson, Á. L. V. Guedes, and S. Colcher. "Quality Enhancement of Highly Degraded Music Using Deep Learning-Based Prediction Models for Lost Frequencies". In: *Proceedings of the Brazilian Symposium on Multimedia and the Web*. WebMedia '21. Belo Horizonte, Minas Gerais, Brazil: Association for Computing Machinery, 2021, pp. 205–211. ISBN: 9781450386098. DOI: 10.1145/3470482.3479635. URL: https://doi.org/10.1145/3470482.3479635.

[22] V. Välimäki, S. Gonzalez, O. Kimmelma, and J. Parviainen. "Digital audio antiquing - Signal processing methods for imitating the sound quality of historical recordings". In: *AES: Journal of the Audio Engineering Society* 56 (Mar. 2008), pp. 115–139.

[23] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *ArXiv* abs/1810.04805 (2019). URL: https://api.semanticscholar.org/CorpusID:52967399.

[24] G. Penha and C. Hauff. "What Does BERT Know about Books, Movies and Music? Probing BERT for Conversational Recommendation". In: *Proceedings of the 14th ACM Conference on Recommender Systems*. RecSys '20. Virtual Event, Brazil: Association for Computing Machinery, 2020, pp. 388–397. ISBN: 9781450375832. DOI: 10.1145/3383313.3412249. URL: https://doi.org/10.1145/3383313.3412249.

[25] A. van den Oord, S. Dieleman, and B. Schrauwen. "Deep content-based music recommendation". In: *Advances in Neural Information Processing Systems*. Ed. by C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Weinberger. Vol. 26. Curran Associates, Inc., 2013. URL: https://proceedings.neurips.cc/paper_files/paper/2013/file/b3ba8f1bee1238a2f37603d90b58898d-Paper.pdf.

[26] M. Kelly. "An Exploration of BERT for Song Classification and Recommendation". In: 2021. URL: https://kaimalloy.com/172B_Project.pdf.

[27] K. Choi, G. Fazekas, M. Sandler, and K. Cho. *Transfer learning for music classification and regression tasks*. 2017. arXiv: 1703.09179 [cs.CV].

[28] C. Hawthorne, A. Stasyuk, A. Roberts, I. Simon, C.-Z. A. Huang, S. Dieleman, E. Elsen, J. Engel, and D. Eck. *Enabling Factorized Piano Music Modeling and Generation with the MAESTRO Dataset*. 2019. arXiv: 1810.12247 [cs.SD].

[29] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu. *MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training*. 2021. arXiv: 2106.05630 [cs.SD].

[30] Y.-H. Chou, I.-C. Chen, C.-J. Chang, J. Ching, and Y.-H. Yang. "MidiBERT-Piano: Large-scale Pre-training for Symbolic Music Understanding". In: *ArXiv* abs/2107.05223 (2021). URL: https://api.semanticscholar.org/CorpusID:235795740.

[31] H. Zhu, Y. Niu, D. Fu, and H. Wang. "MusicBERT: A Self-Supervised Learning of Music Representation". In: *Proceedings of the 29th ACM International Conference on Multimedia*. MM '21. Virtual Event, China: Association for Computing Machinery, 2021, pp. 3955–3963. ISBN: 9781450386517. DOI: 10.1145/3474085.3475576. URL: https://doi.org/10.1145/3474085.3475576.

[32] L. van der Maaten and G. Hinton. "Visualizing Data using t-SNE". In: *Journal of Machine Learning Research* 9.86 (2008), pp. 2579–2605. URL: http://jmlr.org/papers/v9/vandermaaten08a.html.

[33] C. McKay, J. Cumming, and I. Fujinaga. "JSYMBOLIC 2.2: Extracting Features from Symbolic Music for use in Musicological and MIR Research". In: *Proceedings of the 19th International Society for Music Information Retrieval Conference, ISMIR 2018, Paris, France, September 23-27, 2018*. Ed. by E. Gómez, X. Hu, E. Humphrey, and E. Benetos. 2018, pp. 348–354. URL: http://ismir2018.ircam.fr/doc/pdfs/26%5C_Paper.pdf.