NATIONAL AND KAPODISTRIAN UNIVERSITY OF ATHENS

SCHOOL OF SCIENCES
DEPARTMENT OF INFORMATICS AND TELECOMMUNICATIONS

BSc THESIS

Exploring Automatic Music Generation using Transformer encoder-based Language Models

Alexios A. Spiliotopoulos

Supervisors: **Yannis Panagakis,** Associate Professor
Spyridon Polychronopoulos, Research Fellow

ATHENS

SEPTEMBER 2022

ΕΘΝΙΚΟ ΚΑΙ ΚΑΠΟΔΙΣΤΡΙΑΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εξερευνώντας την Αυτόματη Σύνθεση Μουσικής με τη χρήση Γλωσσικών Μοντέλων βασισμένων σε κωδικοποιητή Transformer

Αλέξιος Α. Σπηλιωτόπουλος

Επιβλέποντες: **Γιάννης Παναγάκης,** Αναπληρωτής Καθηγητής
Σπυρίδων Πολυχρονόπουλος, Επιστημονικός Συνεργάτης

ΑΘΗΝΑ

ΣΕΠΤΕΜΒΡΙΟΣ 2022

BSc THESIS

Exploring Automatic Music Generation using Transformer encoder-based Language Models

Alexios A. Spiliotopoulos

S.N.: 1115201700147

SUPERVISORS: **Yannis Panagakis,** Associate Professor
Spyridon Polychronopoulos, Research Fellow

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Εξερευνώντας την Αυτόματη Σύνθεση Μουσικής με τη χρήση Γλωσσικών Μοντέλων βασισμένων σε κωδικοποιητή Transformer

Αλέξιος Α. Σπηλιωτόπουλος

Α.Μ.: 1115201700147

ΕΠΙΒΛΕΠΟΝΤΕΣ: **Γιάννης Παναγάκης,** Αναπληρωτής Καθηγητής
Σπυρίδων Πολυχρονόπουλος, Επιστημονικός Συνεργάτης

ABSTRACT

Automatic music generation is a long-standing goal in artificial intelligence, with deep learning-based methods emerging as the most prominent approach in recent years. Language Models (LMs) such as GPT and BERT, which have achieved great success in various natural language processing tasks, have also been applied to music generation. Despite the fact that most research focuses on decoder-based Transformer models in order to compose music, to our knowledge there have been no previous attempts to use Transformer encoder-only models. In this work, we attempt to explore the domain of automatic music generation by employing MusicBERT, a pre-trained large-scale Language Model based on the Transformer encoder, and further training it using Baroque piano music from MIDI data. Considering that encoder-based networks are not autoregressive, we adapt several sampling techniques in order to exploit the captured knowledge. The results of the experiments have shown that directly using an encoder-only Language Model as a generative model can be trickier than originally thought and poses a number of challenges. Potential future improvements are discussed in the conclusion of this work.

SUBJECT AREA: Automatic Music Composition

KEYWORDS: Transformer, BERT, MusicBERT, MIDI, Symbolic Music, Music Generation, Language Models

ΠΕΡΙΛΗΨΗ

Η αυτόματη παραγωγή μουσικής είναι ένας μακροχρόνιος στόχος στην τεχνητή νοημοσύνη, με μεθόδους που βασίζονται στη βαθιά μάθηση να αναδεικνύονται ως η πιο εξέχουσα προσέγγιση τα τελευταία χρόνια. Γλωσσικά μοντέλα όπως το GPT και το BERT, τα οποία έχουν κατορθώσει μεγάλη επιτυχία σε διάφορα προβλήματα επεξεργασίας φυσικής γλώσσας, έχουν επίσης εφαρμοστεί στη δημιουργία μουσικής. Παρά το γεγονός ότι οι περισσότερες έρευνες επικεντρώνονται σε μοντέλα Transformer βασισμένα σε αποκωδικοποιητή για τη σύνθεση μουσικής, από όσο γνωρίζουμε δεν έχουν γίνει προηγούμενες προσπάθειες χρήσης μοντέλων μόνο με τον κωδικοποιητή ενός Transformer. Σε αυτή την εργασία, επιχειρούμε να εξερευνήσουμε τον τομέα της αυτόματης παραγωγής μουσικής χρησιμοποιώντας το MusicBERT, ένα προ-εκπαιδευμένο γλωσσικό μοντέλο μεγάλης κλίμακας που βασίζεται στον κωδικοποιητή του Transformer, και εκπαιδεύοντάς το περαιτέρω χρησιμοποιώντας μουσική πιάνου Μπαρόκ μουσικής από δεδομένα MIDI. Λαμβάνοντας υπόψη ότι τα δίκτυα που βασίζονται σε κωδικοποιητές δεν είναι αυτοπαλίνδρομα, προσαρμόζουμε διάφορες τεχνικές δειγματοληψίας προκειμένου να εκμεταλλευτούμε τη συλλεγόμενη γνώση. Τα αποτελέσματα των πειραμάτων έδειξαν ότι η απευθείας χρήση ενός γλωσσικού μοντέλου μόνο με κωδικοποιητή ως παραγωγικού μοντέλου μπορεί να είναι πιο περίπλοκη από ό,τι είχε αρχικά θεωρηθεί και θέτει πολλές προκλήσεις. Πιθανές μελλοντικές βελτιώσεις συζητούνται στο τέλος αυτής της εργασίας.

ΘΕΜΑΤΙΚΗ ΠΕΡΙΟΧΗ: Αυτόματη Σύνθεση Μουσικής

ΛΕΞΕΙΣ ΚΛΕΙΔΙΑ: BERT, MusicBERT, MIDI, Μουσική Συμβόλων, Παραγωγή Μουσικής, Γλωσσικά Μοντέλα

ACKNOWLEDGEMENTS

I am extremely grateful to both my supervisors Professor Yiannis Panagakis and Spyros Polychronopoulos for their constant support and precious guidance towards the completion of my thesis. Also, I would like to thank my family and friends for their encouragement and support throughout my studies.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

The idea of devising an algorithmic approach to music creation has been around for decades. Along with advances in technology, the need for original music has also increased. Video games, television shows, YouTube videos, and other digital content are emerging at very fast rates each year. It is not surprising that many people, such as content creators, use music-making software backed by an AI system, as it is cost-effective both in terms of budget and time. Surprisingly, musicians have now started to use such tools as well to enrich their compositions.

In the past decade, the rapid development of deep learning has led to growing interest in the field of automatic music generation, both in research and industry. Various architectures have been created to understand music and generate new original pieces. Among these, the most significant one is the adoption of the Transformer neural network [1], which became popular in the field of Natural Language Processing (NLP) and was then applied to other Machine Learning tasks. At its core lies an attention mechanism that allows the model to dynamically incorporate information from different parts of the input at once. Such a feature has great potential for modeling complex structures in music. Huang et al. (2018) [4] first used the Transformer in an attempt to utilize language modeling techniques in symbolic music, in order to achieve compelling results in music generation. There are works that focus solely on the generation of melody and/or accompaniment, while others focus on more specific tasks such as performance generation [5].

In addition to music generation, a relevant area of research is the modeling of symbolic music, also known as Symbolic Music Understanding. It is generally easier to find meaningful attributes within natural language text, which is not the case in musical pieces. There are studies [3] [6] that use Large Language Models (LLMs) such as BERT [2] or variants, and pre-train them on large corpora in order to obtain general-purpose models for this task. Such models are effective in capturing complex musical structures and knowledge that can be applied to more specific tasks and/or data sets. Previous work relied on the Transformer, or just its decoder part, due to its autoregressive nature that fits the generation objectives, and as far as we know, there have been no studies working on music generation using LLM based on Transformer encoders.

In this thesis, we explore the domain of automatic music generation by using a pre-trained large-scale Language Model based on the Transformer encoder and testing a number of methods in order to generate new music. The best model candidate for that task was MusicBERT, as it is trained on a massive symbolic music corpus of more than 1M MIDI songs. Our attempts showed that the task proved to be more challenging, as the results were not very promising. While different initial settings and parameters were put into test, it was discovered that the model struggled particularly in predicting the tokens regarding the position of a note in a composition. The key points of this thesis include the examination of MusicBERT's generative ability by conducting a series of sampling experiments and providing an initial framework for further research, and finally some insight into possible solutions. Despite the results, with some more work, the usage of LLMs such as MusicBERT could lead to better quality generations.

2. RELATED WORK

2.1 Transformers and Large Language Models

Until recently, the Recurrent Neural Networks (RNNs) were one of the best ways to capture dependencies in symbolic music sequences [7, 8]. Due to the fact that they were designed to process sequential data word by word and also remember past information, RNNs were a good fit for generation tasks. As useful as they were, they also came with some fundamental flaws [9], such as lack of parallelism, difficulty in processing long sequences, slow and complex training, etc. Ever since the Transformer architecture was invented, the usage of RNNs has declined, and Transformer-based models have become a popular choice among researchers for a vast number of tasks, one of them being music generation.

The Transformer has solved the problems RNNs faced, by processing whole input sequences, thus allowing for parallelization, which resulted in faster training. By doing so, the only limitation was the length of each sequence that could fit into memory. One key innovation of this architecture is the self-attention mechanism, a newly introduced ”unit” used to compute similarity scores between tokens in a sentence and modeling tokens' significance over the input. This architecture has captivated researchers because of its capabilities, and over the years different variants of the Transformer have emerged. In the Computer Music domain, some studies [4] made the Transformer more memory efficient while also devising new ways to model symbolic music [3, 4, 10], others [11] employ Transformer-XL [12] networks in order to increase the model's attention span on the input, creating longer pieces. Currently, there are several Transformer-based works [4, 10, 13–15], which produce great results, but due to the subjective nature of music, it is fairly difficult to qualify a generative model as state-of-the-art.

Meanwhile, pre-trained language models (e.g., BERT, GPT [16]) have proven to be powerful for generative tasks and representation learning from large corpora. These language models are also called large language models because they differ from the usual transformer-based models in the number of layers. GPT-2, which is based on the Transformer decoder, has been used to compose 4 minute-long pieces and also combine different composer and genre styles [17] or by pairing the model with a novel representation for musical material, producing interesting multi-track music while also providing increased control over the generated music [15]. However, most Transformer-based research is focusing on the Transformer encoder-decoder scheme or the Decoder part of the Transformer only, due to its ability to function in an autoregressive manner; ideal for generating musical sequences. To the best of our knowledge, no prior work has used encoder-only language models such as BERT to generate music.

3. TECHNICAL BACKGROUND

3.1 Attention Mechanism

The attention mechanism [18] was originally introduced as an improvement over *encoder-decoder* architectures for neural machine translation in the NLP domain. Later, this mechanism or its variants were used in other applications including computer vision, speech processing, etc. The idea behind this technique was to allow the decoder to use the most relevant parts of the input sequence through a weighted combination of all the encoded input vectors, with the most relevant vectors being attributed the highest weights. This can be especially helpful when the input data is very large and/or complex. Additionally, the attention mechanism can help improve the interpretability of a model by providing a way to visualize which parts of the input the model is focusing on.

3.2 Transformer

In 2017, a major breakthrough paper in the field of NLP was published under the name *Attention Is All You Need* [1]. It introduced the Transformer, an architecture that has become a standard choice for many researchers not only in NLP but also in many other branches of machine learning, as it allowed more context to be used. It is designed to process sequential data and, unlike RNN, it can be processed in parallel. At the core of Transformer lies a new version of the attention mechanism, *self-attention*.

The Transformer model follows the standard *encoder-decoder* architecture as shown in Figure 1. A variable-length input is first passed through an encoder. The encoder, in turn, will transform the input into a fixed-length hidden state. This hidden state will go through the second component, the decoder, which converts that state into a variable-length output. At the beginning of this process, the input tokens are converted into vectors with the help of an embedding layer, and then positional information, also called *positional encoding*, is added to the vectors to take into account the order of the tokens during the concurrent processing of the model.

Generally speaking, the encoder receives an input sequence of symbol representations $(x_1, ..., x_n)$ and creates a sequence of continuous representations $z = (z_1, ..., z_n)$. This new representation of the input is given to the decoder that generates an output sequence $(y_1, ..., y_m)$, one element at a time. After the generation of an output element, the model takes that element as an additional input for the next generation.

The Transformer architecture solved two fundamental problems of RNNs. The first was the inability to capture long-range dependencies within the input sequences. In this architecture, long-range dependencies have the same likelihood of being taken into account as any other short-range dependencies. The second fundamental issue was the exploding/vanishing gradients problem. In reality, RNNs are a very deep network, where the size of the network depends on the length of the sequence. In contrast, the Transformer has fewer layers into which information is passed, meaning that the information is prone to fewer changes. Additionally, the Transformer utilizes a number of methods that help solve that issue, such as the residual connections around each layer in both the encoder and the decoder blocks. One additional characteristic is that it requires fewer training steps to achieve the same performance with the RNN.

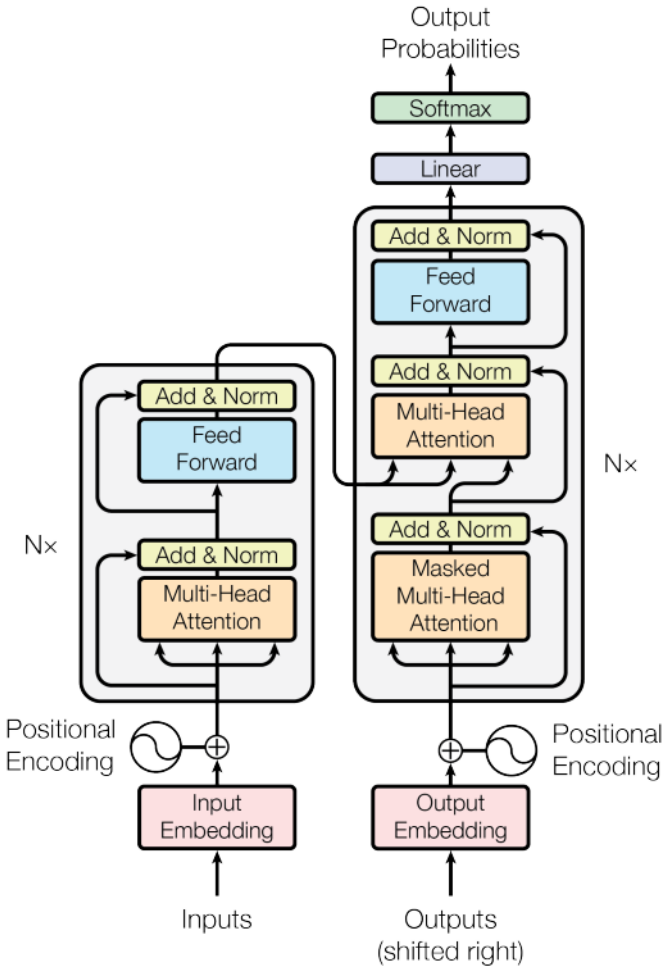

Figure 1: The Transformer model architecture; Encoder (left), Decoder (right) [1, Figure 1]

3.2.1 Encoder

The Encoder consists of 6 identical layers, each layer containing two sub-layers. The first is a multi-head self-attention mechanism and the second is a position-wise fully connected feed-forward network. Residual connections [19] are used around each of these sub-layers, followed by layer normalization. [20].

3.2.2 Decoder

The Decoder also consists of 6 identical layers. Additionally, to the two sub-layers in the encoder, the decoder inserts in between a third sub-layer, which performs attention over the output of the encoder stack. The self-attention sub-layers are also modified to prevent positions from attending to subsequent positions, by masking them. This means that the predictions for a position *i* depend only on the positions less than *i*.

3.2.3 Scaled Dot-Product Attention

The Scaled Dot-Product variant is a more efficient equation for the attention scoring function. The dot product over the input Queries (Q) and Keys (K) is calculated, and then

scaled by $\frac{1}{\sqrt{d_k}}$. The softmax function is applied to the result to obtain the weights of the values (V).

$$Attention(Q, K, V) = softmax\Big(\frac{QK^T}{\sqrt{d_k}}\Big)V$$

To interpret this equation, we can state that the attention mechanism is a neural architecture that mimics this retrieval process. The attention mechanism measures the similarity between some Queries and the Keys. This similarity returns a weight for each key. Finally, it produces an output that is the weighted combination of all the Values.

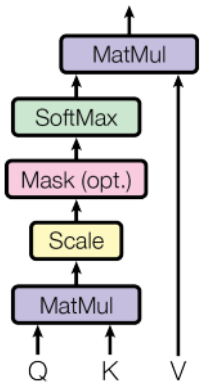

Figure 2: Scaled Dot-Product Attention [1, Figure 2a]

3.2.4 Multi-head Attention

The Queries, Keys, and Values are linearly projected h times with different learned linear projections to dimensions d_k, d_k, and d_v, respectively. Then the attention function is applied to each of these projected versions of Q, K, and V in parallel, creating d_v-dimensional output values, called heads. The heads are concatenated and projected, resulting in the final values as seen in Figure 3.

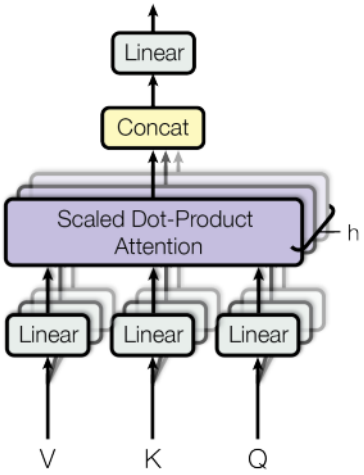

Figure 3: Multi-Head Attention [1, Figure 2b]

$$MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O$$
$$\text{where } head_1 = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

The projections $W_i^Q \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^K \in \mathbb{R}^{d_{model} \times d_k}$, $W_i^V \in \mathbb{R}^{d_{model} \times d_v}$ and $W^O \in \mathbb{R}^{hd_v \times d_{model}}$ are parameter matrices. The reason learned parameter matrices are used is that the network can learn patterns that give a much more useful context.

3.2.5 Usage of Attention in the Transformer

Figure 1 shows that the Attention mechanism is used in both the Encoder and the Decoder:

- The encoder contains a self-attention layer. All the keys, values, and queries originate from the same place, which in this case is the output of the previous encoder layer
- In the decoder layers, the self-attention layer allows each position to attend to all positions in the decoder up to (and including) that position. This feature allows for prevention of leftward information flowing in the decoder, thus preserving the auto-regressive property.
- The *encoder-decoder attention* layer, which is part of the decoder, receives the queries from the previous decoder layer, but the keys and values come from the encoder output. This allows every position in the decoder to be accounted for by all positions in the input sequence.

3.2.6 Positional Encodings

In RNNs, the input sequences maintain order because they are passed to the model one token at a time. Since the Transformer processes a whole sequence at once, it needs a way to get a sense of the order in that sequence. For this reason, the required information is injected into the sequence through the ”positional encodings”, which are, in essence, embeddings that are added to the input. These encodings are derived from the following equations:

$$PE_{(pos,2i)} = sin\Big(\frac{pos}{10000^{2i/d_{model}}}\Big)$$

$$PE_{(pos,2i+1)} = cos\Big(\frac{pos}{10000^{2i/d_{model}}}\Big)$$

3.3 BERT

BERT [2] which stands for Bidirectional Encoder Representations from Transformers is a multilayer bidirectional Transformer encoder used to improve contextual understanding of unlabeled text across a broad range of tasks by learning to predict text that might come

before and after other text, achieving state-of-the-art results. Such tasks include text summarization, question answering, sentiment analysis, and named entity recognition. At its core, it consists of a variable number of Transformer encoder layers and self-attention heads.

3.3.1 Pre-training

The training process involves two objectives. The first is Masked Language Modeling (MLM). Unlike traditional language modeling, which tries to predict the next word in a sequence based on the previous words, MLM tries to predict a word in a sequence based on its left and right contexts. Basically, it is a technique in which some of the input tokens are randomly masked (replaced with a [MASK] token), and the model is trained to predict the masked tokens. Specifically, the data generator of BERT chooses 15% of the input tokens, and each chosen token has 80% chance of being masked, 10% chance of being replaced with a random token, and 10% chance of remaining unchanged. BERT varies from denoising auto-encoders, in the sense that it simply predicts these tokens rather than reconstructing the whole input.

The second objective is Next Sentence Prediction (NSP). This training objective is crucial for many downstream tasks, such as question answering, that rely on understanding the relationship between two sentences. To tackle this, BERT is trained using different pairs of sentences, where half are actual consecutive sentences and the other half are not. The pairs are fed into the model separated by a special token, and the goal is to decide if the second sentence of the pair is next to the first one or not. Both of these objectives work together during pre-training, as shown in the left part of Figure 4.

3.3.2 Fine-tuning

What is great about pre-training is that it allows the model to gain a deeper understanding of the language structure. The pre-training tasks are not very useful in themselves, and instead the purpose of BERT is to be fine-tuned on specific language tasks. Fine-tuning is a process of further training a pre-trained model on some other downstream task. During fine-tuning, additional layers can be added to BERT with randomized parameters: these parameters and those pre-trained BERT parameters will be updated to fit training data of downstream tasks(e.g., classification, entity recognition, question answering, etc.) .

3.3.3 RoBERTa variant

RoBERTa [21], which stands for Robustly Optimized BERT Pre-training Approach, was introduced as an improved version of BERT that is faster and more accurate. The goal was to optimize the training of BERT architecture in order to take less time during pre-training. To do so, the authors basically made some improvements to the original BERT model. They experimented with the NSP objective using different input settings and found that by removing that objective, the model matched or slightly improved downstream task performance.

The masked language modeling objective in BERT pre-training means masking a number of tokens from each sequence at random and then predicting them. However, in the

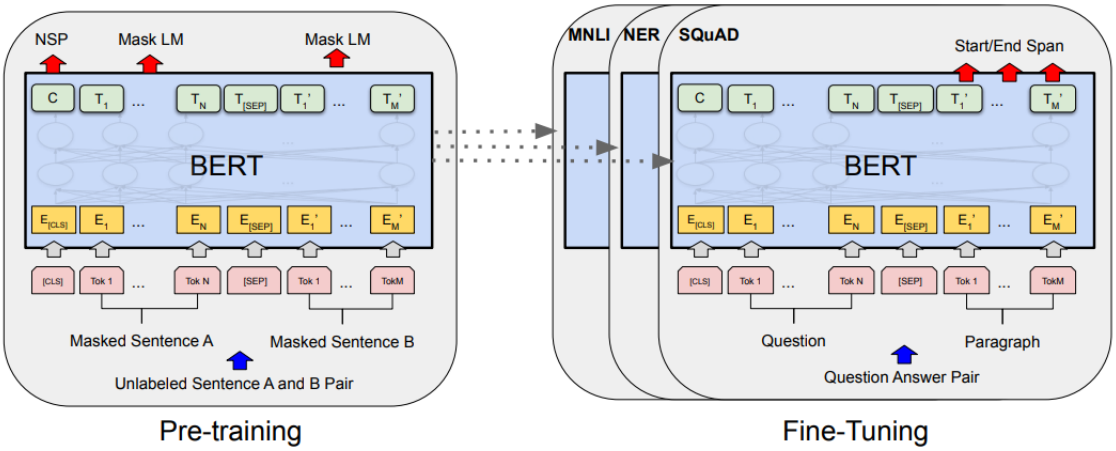

Figure 4: The pre-training and fine-tuning procedures for BERT. Except for the output layer, the same architecture is used for both pre-training and fine-tuning [2, Figure 1]

original implementation of BERT, the sequences are masked just once in the preprocessing step. Essentially, this means that the same masking pattern is used for the same sequence in every epoch. The authors of RoBERTa experimented by duplicating the training data 10 times so that each sequence was masked in 10 different patterns. This method was tested over 40 epochs, which means that each sequence was trained 4 times with the same masking patterns. Also, they experimented by masking each input dynamically on every pass from the model. They found that the dynamic pattern yielded comparable or slightly better results than the static approaches. Hence, they adopted the dynamic masking approach for pre-training.

Additional improvements included training over a longer time, with more data, using larger batch sizes and longer lengths for the input sequences. All of these changes led to an overall improved and better trained version of BERT.

3.4 MusicBERT

Although pre-trained natural language models (such as BERT) provide robust learning to represent unlabeled text corpora, it is quite difficult to translate this idea directly to learning to represent symbolic music (e.g. MIDI format). For example, musical songs have a different structure, require pitch/instrument information, etc. Other practical problems include the lack of large-scale musical corpora, and naively encoding of musical notes results in sequences too long to be processed at once. The authors address these issues and provide practical solutions by building a large-scale symbolic music corpus, proposing an efficient encoding of musical notes, and pre-training the model with an appropriate masking scheme that better suits the structure of music.

3.4.1 Overall Architecture

At its core, MusicBERT pre-trains a Transformer encoder, specifically an instance of the RoBERTa variant, with the masked language modeling objective using a masking strategy described in 3.4.3. To encode the music sequence more efficiently, an encoding method called OctupleMIDI is used, which encodes a symbolic music piece into a sequence of

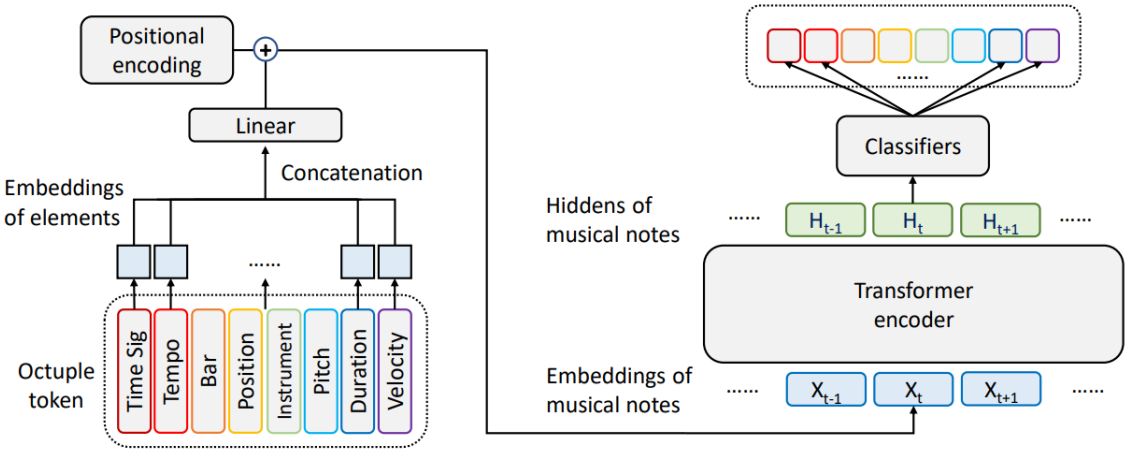

Figure 5: Architecture of MusicBERT [3, Figure 1]

octuple tokens (an 8-tuple) that contains eight basic elements related to a music note (described in Sec. 3.4.2). The embeddings of each sub-token are concatenated and fed into a linear layer. The resulting vector is then added with the corresponding positional embeddings to form the encoder's input. In order for the model to predict each of the eight sub-tokens, 8 distinct softmax layers are added, the hidden states of the encoder are passed through 8 distinct softmax layers, so that they are mapped to the vocabulary sizes of the eight different element types, respectively.

After pre-training, MusicBERT is fine-tuned on four downstream music understanding tasks, including melody completion, accompaniment suggestion, genre classification, and style classification, achieving state-of-the-art results in all tasks [3].

3.4.2 OctupleMIDI Encoding Scheme

OctupleMIDI is a novel symbolic music encoding method that encodes each MIDI note into a tuple with eight elements, representing the different musical characteristics of a note, such as bar, position, instrument, pitch, duration, velocity, time signature, and tempo. This method results in significantly shorter input sequences in contrast to other musical encoding schemes such as REMI [4] or CP [10].

3.4.3 Bar-level Masking Strategy

The masking strategy in original BERT for NLP tasks randomly masks some tokens, which, in the case of music, may cause information leakage during pre-training. Therefore, the authors use a novel bar-level masking strategy in which all the tokens of the same type (for example, time signature, instruments, pitch, etc.) are masked in a bar to avoid information leakage and for better learning of hidden representations. Inspired by RoBERTa the authors use a dynamic masking strategy, as already mentioned. Experiments have shown that bar-level masking is essential for properly learning the symbolic music representation with the Transformer encoder.

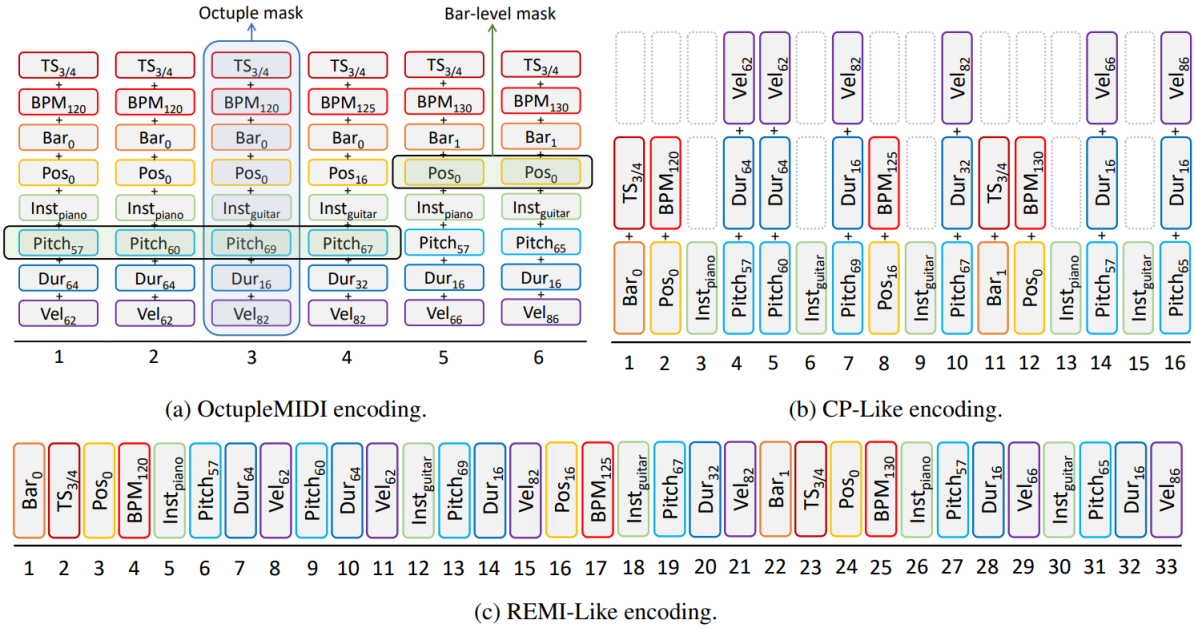

Figure 6: Different Encoding Methods for Symbolic Music. Figure (a) includes the two different masking strategies [3, Figure 2]

3.4.4 Large-scale Pre-training Corpus

In order to increase the performance of MusicBERT, a diverse large-scale symbolic music dataset was constructed, called Million Midi Dataset (MMD), which contains more than 1 million music songs, with different genres, including Rock, Electronic, Rap, Jazz, Latin, Classical, etc.

4. EXPERIMENTS

4.1 Experiment Settings

4.1.1 Dataset

Decoder-based models generally can handle a mix of different music styles during training and perform well in inference. Since encoder-based neural networks have not been tested in the music generation task, it was uncertain whether mixing music styles would be beneficial at this time. Therefore, a conscious decision was made to condition the model to a specific style of music to facilitate the generation process.

The model was fine-tuned with single-instrument classical pieces from the Baroque era. This decision is based on the fact that Baroque pieces are generally more structured and follow specific forms. Most dynamics are terraced, meaning that volume levels shift abruptly from soft to loud, and back, and the rhythm is generally continuous with little or no rests. Since the model processes symbolic music, the data are taken from the GiantMIDI-Piano [22] dataset. The composers shown in Table 1 have been chosen for the dataset.

In contrast to usually fine-tuning downstream tasks, MusicBERT was further trained on masked language modeling, but with a more specific musical corpus, consisting of several Baroque composers. As encoder-based architectures are not autoregressive by nature, they cannot generate new notes in such a manner.

Composers	Pieces	Octuple Tokens
A. Scard	1	1023
J.M. Pacheco	3	2057
J. Pachelbel	5	2895
H. Pachulski	7	4442
H. Purcell	10	6777
J. Pachman	21	17726
G.F. Handel	78	55731
J.S. Bach	147	130446
D. Scarlatti	279	277394
Total	551	498491

Table 1: The number of pieces and total tokens used in fine-tuning. Each token is composed of 8 sub-tokens, so the real total token number is 3987928.

4.1.2 Model configuration

An instance of *MusicBERT-large* was fine-tuned using fairseq-cli [23] on a single NVIDIA Quadro RTX 5000 GPU with a batch size of 256. Adam optimizer ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1e-6$ and $L2$ weight decay of 0.01), and the learning rate was warmed up over the first 25000 steps to a maximum value of $5e-4$ and then linearly decayed. The dropout value for all layers and attention weights is set to 0.1.

4.1.3 Fine Tuning

Naturally, the training process would need to start from scratch with an untrained model. However, due to limited computational resources, it was decided to further train a pre-trained checkpoint of MusicBERT. This was also beneficial as it allowed us to leverage the knowledge gained during pre-training to our advantage. Consequently, a pre-trained checkpoint of MusicBERT was fine-tuned, more specifically the ”MusicBERT-base” checkpoint which consists of 12 Transformer encoder layers.

4.2 Method Analysis

For the experiments, the idea of sampling from BERT models [24] was adapted to our framework. During the generation process, the sampling methods that were used are the natural left-to-right style, the Gibbs sampling technique, and also examined BERT's original mask filling technique (simultaneously predicting all masked tokens). In all three strategies, the distribution of logits was filtered using top-k and/or nucleus (top-p) filtering.

4.2.1 Gibbs Sampling

The idea behind sampling of BERT models with this method was originally proposed by Wang and Cho [24], and it essentially means starting with an all-mask sequence, or a sequence consisting of randomly sampled tokens, or by using any other sequence (e.g. from the corpus). By iteratively selecting random positions from the sequence and masking them if they are not already, MusicBERT was used to predict the probability distribution of the masked Octuple tokens. Then the randomly chosen tokens can be replaced with new tokens drawn from the predicted probability distribution. This process can be repeated infinitely, although it is suggested to select a sample only once a while during sampling.

The experiments follow five different initial sequence settings:

- an all-mask sequence
- an all-mask sequence but with a short prompt in the beginning
- a randomly sampled sequence
- a part from some random piece from the Baroque corpus
- a part from an out-of-corpus piece

4.2.2 Sequential Mask Filling

Given that the dominant approach to text generation is left-to-right, the first attempts at generating with MusicBERT were made in this manner. First, a seed sentence of either all masks or a prompt followed by one or more masks is initialized. At each time step t, a subset of the initial sequence is extracted, in which the rightmost token is masked. An octuple token is generated for the rightmost position and is substituted into the initial sequence. This process is repeated until all the masked positions have been filled.

4.2.3 Simultaneous Mask Filling

Lastly, several experiments were conducted in the same manner that MusicBERT is pre-trained. It would be interesting to test the ability of MusicBERT to generate a whole sequence all at once, either with no prior information or with a prompt provided.

4.3 Results

The main problem encountered in all the experiments was the fact that the model had difficulty placing the notes correctly. Specifically, the bar and position sub-tokens of each Octuple were not predicted correctly most of the time. Some notes would be placed in the same bar, sometimes resulting in a single bar having more than 20 notes, and other times they would be spaced out across a number of bars, resulting in a very sparse generation. In the worst of these cases, the notes in some generations would not start from the first bar but instead from a subsequent one.

For this reason, a slight post-processing was applied on the bar and position sub-tokens, in a way that notes far apart from each other would come closer to the start, and notes cluttered inside a bar would get scattered. However, it was not possible to find an efficient way to do so without violating the generated time signature sub-tokens. Also, in some rare cases where some tokens would be generated as *drums*, the sub-token corresponding to the MIDI instrument was overwritten to *Acoustic Grand Piano* (first MIDI instrument). Lastly, the invalid predicted Octuple tokens were removed during post-filtering, i.e. having incorrect sub-token order (bar-position-instrument-pitch-velocity-time signature-tempo). This problem can be overcome by sorting the sub-tokens, though it could be a good metric for comparing different experiments. As a general observation, most predictions had a $4/4$ time signature and a tempo around 120 BPM, which is pretty common in Baroque pieces.

4.3.1 Method comparison

It was found that the simultaneous mask filling was undoubtedly the weakest method among the three. Two main common characteristics were observed from the generated tokens. The first was the short duration of the notes. Although the model was trained on various note durations, it seemed to favor shorter ones, such as $1/16$ and $1/32$. Different prompts were also used with various note durations, and all trials ultimately yielded the same results. The second observation was that almost all notes were clustered inside one or two bars following a prompt's last bar. Additionally, the invalid predicted tokens were nearly half of the length of the generated sequence. However, when a prompt was used, that number decreased to almost zero. This can be explained by the fact that the prompt provided the model with the required context on the subtokens' order.

Regarding the Gibbs sampling method, the model was first tested by feeding it with sequences from the training corpus. Several random tokens were masked in the input -one at each time step- and the model was expected to replace them with tokens relevant to the ones around them. Some of them had fairly relevant predictions, and some others had faulty bar and position sub-tokens. The same assessment was performed, this time with pieces outside the corpus, and the results were the same. It seemed that the model did not quite overfit the sequences given to it in the sense that the predictions were not the same as the masked counterparts.

This method was also tested using all-mask inputs and it was found that the duration of the notes varied more, compared to the method discussed above. It was also observed that the issue of note positioning was still present. When given a prompt, most predicted notes were gathered inside the next few bars, and in the absence of it they were in scattered positions inside the piece.

The left-to-right generation method was evaluated using an all-mask input sequence with and without a prompt. Unfortunately, almost no differences were observed with the Gibbs method.

4.3.2 Influence of prompts and hyper-parameters

Generally, the use of prompts showed better results in terms of the position of the notes and the validity of the predictions. The context that was provided seemed to assist MusicBERT in making sense of the starting position, thus generating somewhat better bar/-position sub-tokens. As mentioned, it was noticed that the generated notes tend to be placed inside a single or two bars after a given prompt. This comes in contrast to generations without any prompt, in which notes were very far apart from each other, or sometimes clustered in far apart bars.

Also, when a prompt was given the top_k, top_p and temperature parameters seemed to influence mostly the pitch sub-token of the predicted octuple tokens, specifically noticing changes in variability.

5. CONCLUSION AND FUTURE WORK

In this thesis, the goal was to use MusicBERT, a large-scale musical language model, to compose new music. Specifically, a pre-trained checkpoint was utilized, and further trained on specific data, assuming that it would simplify the output distribution. As initially thought, these experiments have shown that directly using an encoder-only Language Model as a generative model can be a very challenging task, and it confirmed the fact that autoregressive language models, such as the GPT series, perform better at generation tasks. However, we believe that MusicBERT has the potential to give better results at this task, and in order to do so more work has to be done, accompanied by extensive experiments. For future work, it would be interesting to experiment by training MusicBERT as a Denoising Autoencoder [25,26]. Currently, the non-masked tokens are changed when passed through the model and ignored; therefore, the training procedure can be tweaked in order to reconstruct the original input. Additionally, the experiments can be extended by adapting the pre-trained encoder to an encoder-decoder architecture, taking advantage of the hidden representations of the model, and also performing in an autoregressive manner instead of directly sampling from the encoder's output.

ABBREVIATIONS - ACRONYMS

BERT	Bidirectional Encoder Representations from Transformers
NLP	Natural Language Processing
RNN	Recurrent Neural Network
LM	Language Model
LLM	Large Language Model
MLM	Masked Language Modeling
MIDI	Musical Instrument Digital Interface

REFERENCES

[1] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017, p. 6000–6010.

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

[3] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T.-Y. Liu, "Musicbert: Symbolic music understanding with large-scale pre-training," *arxiv preprint arXiv:2106.05630*, 2021.

[4] C.-Z. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. M. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck, "Music transformer: Generating music with long-term structure," in *ICLR*, 2019.

[5] S. Oore, I. Simon, S. Dieleman, D. Eck, and K. Simonyan, "This time with feeling: Learning expressive musical performance," *arxiv preprint arXiv:1808.03715*, 2018.

[6] 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," *arxiv preprint arXiv:2107.05223*, 2021.

[7] H. H. Mao, T. Shin, and G. Cottrell, "DeepJ: Style-specific music generation," in *2018 IEEE 12th International Conference on Semantic Computing (ICSC)*. IEEE, 2018.

[8] N. Meade, N. Barreyre, S. C. Lowe, and S. Oore, "Exploring conditioning for generative music systems with human-interpretable controls," *arXiv preprint arXiv:1907.04352*, 2019.

[9] R. Pascanu, T. Mikolov, and Y. Bengio, "Understanding the exploding gradient problem," *arXiv preprint arXiv:1211.5063*, 2012.

[10] W.-Y. Hsiao, J.-Y. Liu, Y.-C. Yeh, and Y.-H. Yang, "Compound word transformer: Learning to compose full-song music over dynamic directed hypergraphs," *arXiv preprint arXiv:2101.02402*, 2021.

[11] X. Wu, C. Wang, and Q. Lei, "Transformer-xl based music generation with multiple sequences of time-valued notes," *arxiv preprint arXiv:2007.07244*, 2020.

[12] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. Le, and R. Salakhutdinov, "Transformer-XL: Attentive language models beyond a fixed-length context," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019, pp. 2978–2988.

[13] B. Yu, P. Lu, R. Wang, W. Hu, X. Tan, W. Ye, S. Zhang, T. Qin, and T.-Y. Liu, "Museformer: Transformer with fine- and coarse-grained attention for music generation," *arXiv preprint arXiv:2210.10349*, 2022.

[14] Y. Qin, H. Xie, S. Ding, B. Tan, Y. Li, B. Zhao, and M. Ye, "Bar transformer: a hierarchical model for learning long-term structure and generating impressive pop music," *Applied Intelligence*, 2022.

[15] J. Ens and P. Pasquier, "Mmm: Exploring conditional multi-track music generation with the transformer," *arxiv preprint arXiv:2008.06048*, 2020.

[16] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[17] C. Payne, "Musenet," *OpenAI*, 2019. [Online]. Available: https://openai.com/blog/musenet

[18] D. Bahdanau, K. Cho, and Y. Bengio, "Neural machine translation by jointly learning to align and translate," *arXiv preprint arXiv:1409.0473*, 2014.

[19] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," *arxiv preprint arXiv:1512.03385*, 2015.

[20] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arxiv preprint arXiv:1607.06450*, 2016.

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

[22] Q. Kong, B. Li, J. Chen, and Y. Wang, "Giantmidi-piano: A large-scale midi dataset for classical piano music," *arxiv preprint arXiv:2010.07061*, 2020.

[23] M. Ott, S. Edunov, A. Baevski, A. Fan, S. Gross, N. Ng, D. Grangier, and M. Auli, “fairseq: A fast, extensible toolkit for sequence modeling,” in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*. Minneapolis, Minnesota: Association for Computational Linguistics, 2019, pp. 48–53. [Online]. Available: https://aclanthology.org/N19-4009

[24] A. Wang and K. Cho, “BERT has a mouth, and it must speak: BERT as a Markov random field language model,” in *Proceedings of the Workshop on Methods for Optimizing and Evaluating Neural Language Generation*, 2019, pp. 30–36.

[25] M. Lewis, Y. Liu, N. Goyal, M. Ghazvininejad, A. Mohamed, O. Levy, V. Stoyanov, and L. Zettlemoyer, “BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension,” in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, 2020, pp. 7871–7880.

[26] Y. Bengio, L. Yao, G. Alain, and P. Vincent, “Generalized denoising auto-encoders as generative models,” *Advances in Neural Information Processing Systems*, 2013.