UNIVERSIDADE FEDERAL DE MINAS GERAIS
Instituto de Ciências Exatas
Programa de Pós-Graduação em Ciência da Computação

Gabriel Souza Gomes

Explicit Representation of Note Duration Improves Structural Similarity in Transformer Models

Belo Horizonte
2023

Gabriel Souza Gomes

Explicit Representation of Note Duration Improves Structural Similarity in Transformer Models

Final Version

Dissertation presented to the Graduate Program in Computer Science of the Federal University of Minas Gerais in partial fulfillment of the requirements for the degree of Doctor in Computer Science.

Advisor: Flávio Vinícius Diniz de Figueiredo
Co-Advisor: Alexei Manso Corrêa Machado

Belo Horizonte
2023

Gomes, Gabriel Souza.

G633e Explicit representation of note duration improves structural similarity in transformer models [recurso eletrônico] / Gabriel Souza Gomes - 2023.

1 recurso online (52 f. il., color.) : pdf.

Orientador: Flávio Vinícius Diniz de Figueiredo.
Coorientador: Alexei Manso Corrêa Machado.

Dissertação (Mestrado) - Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Ciência da Computação.
Referências: f. 48-52

1. Computação – Teses. 2. Aprendizado do computador – Teses. 3. Aprendizado profundo – Teses. 4. Inteligência computacional – Música – Teses. II. Machado, Alexei Manso Corrêa. III. Universidade Federal de Minas Gerais, Instituto de Ciências Exatas, Departamento de Computação. IV. Título.

CDU 519.6*82(043)

Ficha catalográfica elaborada pela bibliotecária Irénquer Vismeg Lucas Cruz
CRB 6/819 - Universidade Federal de Minas Gerais - ICEx

UNIVERSIDADE FEDERAL DE MINAS GERAIS
INSTITUTO DE CIÊNCIAS EXATAS
PROGRAMA DE PÓS-GRADUAÇÃO EM CIÊNCIA DA COMPUTAÇÃO

FOLHA DE APROVAÇÃO

Explicit Representation of Note Duration Improves Structural Similarity in Transformer Models

GABRIEL SOUZA GOMES

Dissertação defendida e aprovada pela banca examinadora constituída pelos Senhores:

PROF. FLAVIO VINICIUS DINIZ DE FIGUEIREDO - Orientador
Departamento de Ciência da Computação - UFMG

PROF. ALEXEI MANSO CORRÊA MACHADO - Coorientador
Departamento de Anatomia e Imagem - UFMG

PROF. ALEXEI MANSO CORRÊA MACHADO
Departamento de Anatomia e Imagem - Faculdade de Medicina - UFMG

PROF. PEDRO OLMO STANCIOLI VAZ DE MELO
Departamento de Ciência da Computação - UFMG

PROF DIEGO FURTADO SILVA
Instituto de Ciências Matemáticas e de Computação - USP

Documento assinado digitalmente
gov.br
DIEGO FURTADO SILVA
Data: 13/08/2024 15:47:55-0300
Verifique em https://validar.iti.gov.br

Belo Horizonte, 15 de setembro de 2023.

Resumo

Aprendizado profundo demonstrou, recentemente, resultados formidáveis em computação criativa, mesmo para dados complexos. Alguns trabalhos são notórios por criarem imagens de alta resolução impressionantes a partir de entradas de texto, enquanto outros são renomados por escreverem textos longos coerentes e concisos. Mas o mesmo não pode ser dito para criatividade computacional aplicada a composição musical, uma vez que mesmo os melhores trabalhos conseguem gerar resultados com qualidade aceitável apenas para obras curtas. Apesar de parecer mais simples se comparada a imagens de alta resolução ou textos longos, música apresenta desafios únicos devido à natureza de sua estrutura, que contém padrões de repetição (motifs) em escalas de tempo variadas. No cenário atual, modelos com arquitetura transformer são a melhor abordagem para gerar música e ao treinar estes modelos, é necessário escolher entre várias opções de arquitetura e estilos de representação de entrada. Alguns modelos são treinados e testados apenas em datasets com anotações adicionais de estrutura, como tempo, compassos ou frases, sendo que estas anotações são normalmente usadas para melhorar a performance do modelo para gerar tais estruturas. Neste trabalho, questionamos se a arquitetura padrão do MusicTransformer apresenta perda de performance mesmo usando apenas informações MIDI (isto é, sem anotações adicionais de estrutura). Mostramos que uma pequena mudança na representação mais comumente usada resulta em melhorias pequenas, mas significativas. Nossa análise experimental focada em quatro datasets com estilos musicais diferentes (Jazz, Maestro, SNES e Pop) conclui que gerar músicas usando o MusicTransformer e uma representação MIDI que codifica duração de nota explicitamente apresenta melhoria em métricas de estrutura (um fator comumente atribuído a anotações de estrutura ou melhorias em arquitetura) e é corroborada por avaliação humana de qualidade musical. Uma vez que nossa abordagem é aplicável a qualquer dataset MIDI, argumentamos que há mais ganho de performance potencial em geração de músicas usando grande quantidade de dados sem anotação, ao invés de menos dados com maior quantidade de informação.

Palavras-chave: música; aprendizado de máquina; aprendizado profundo; computação criativa.

Abstract

Deep learning has recently demonstrated formidable results in creative computing even when dealing with complex data types. Some works are notorious for being able to create impressive high-resolution images from text prompts, while others are renowned for being able to write many paragraphs of coherent and concise text. But the same can't be said for creative computing applied to music composition, since even the best works can only convincingly create short musical pieces with adequate quality. While music may seem much simpler than high-resolution images or long stretches of text, it presents unique challenges due to the nature of its structure containing coherent repeated structures or motifs at varied timescales. With that said, transformer models have become the go-to approach for generating music. However, when training such models, one is faced with choosing from many options of architecture and input representation to use. More importantly, some models are only trained and tested on datasets with annotated structural information such as tempo, beats, bars, or phrases. This annotated information is usually used to improve the model's performance regarding structural similarity in generated musical pieces. In this work, we inquire if the off-the-shelf MusicTransformer models perform just as well using only MIDI information (that is, with no additional annotations). We show that a slight tweak to the representation most commonly used can yield small but significant improvements. Our experimental analysis focused on four datasets with different musical genres (Jazz, Maestro, SNES, and Pop) finds that generating musical pieces using the MusicTransformer architecture and a MIDI representation that encodes note duration explicitly presents improvements in structural similarity measures (a factor that is usually attributed to the exploitation of different annotations and architectures) and is corroborated by human evaluation of musical quality. Given that our approach is applicable to any plain MIDI dataset (with no external annotation), we argue that there is performance yet to be harnessed on music generation using transformers by using larger quantities of data without any extra annotations.

Keywords: music; machine learning; deep learning; creative computing

List of Figures

List of Tables

Contents

Chapter 1

Introduction

From simple Markovian models [35, 3], to the most recent Deep Learning Transformer architectures [21, 22, 48, 42], automatic music generation systems [20, 2] have come a long way and, most recently, gained significant performance. In particular, and depending on the context or setting, automatically generated music *can* fool humans into believing that such scores were created by humans [8].

Nevertheless, it is well known that music is self-referential on many levels. A musical phrase lasting only a few seconds may contain variations of a few melodic elements, whereas a musical section lasting a minute may contain variations of a few phrases. The same can be said about a musical piece lasting several minutes, which may contain variations of a few sections. Overall, the exploration of repeated musical structures is a major part of musical composition [5, 30, 25].

As an example, Figure 1.1 shows labeled musical sections on a piece by Brahms. Even without knowledge on how to read sheet music, it can be seen how some sections are identical, like B1 and B2, while others like A1 and A2 are slightly different. The difficulty

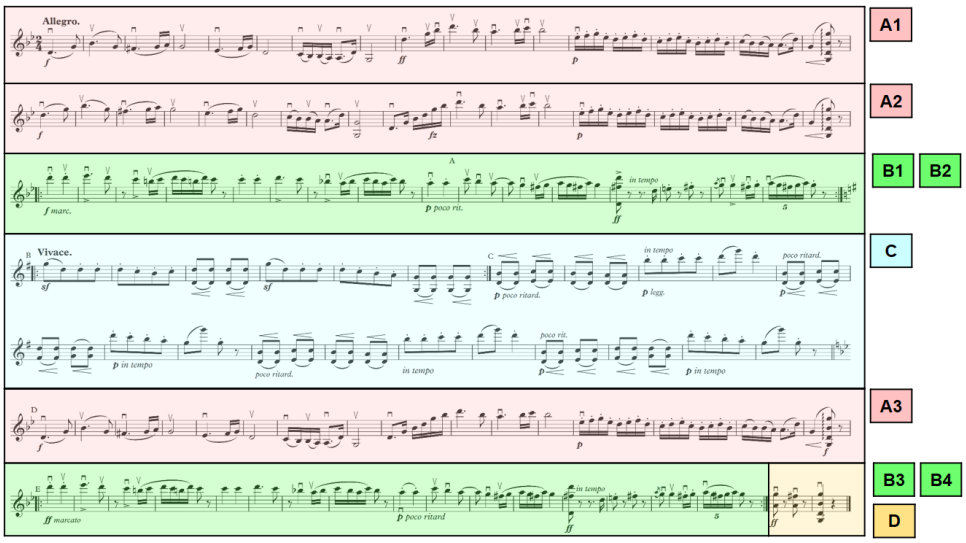

Figure 1.1: A stretch of Hungarian Dance No. 5 by Johannes Brahms, with annotated structure. Source: https://www.audiolabs-erlangen.de/resources/MIR/FMP/C4/C4S1_MusicStructureGeneral.html

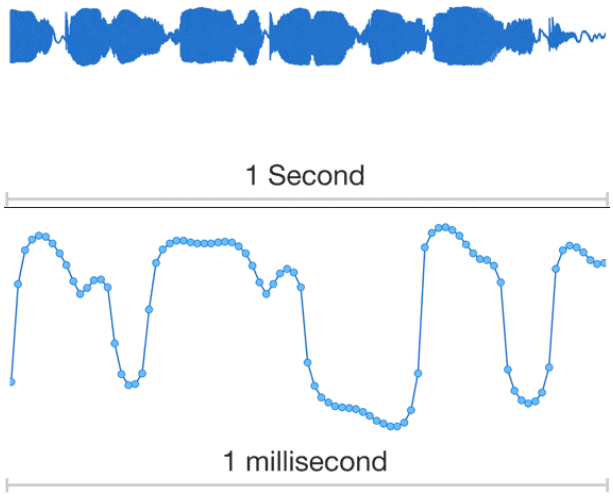

Figure 1.2: An audio file's waveform is a very long timeseries. In standard quality, these waveforms contain 44100 data points in a second of audio. Source: Modified from https://www.deepmind.com/blog/wavenet-a-generative-model-for-raw-audio

in replicating this structure is why deep learning is still not very successful on music composition. Of course, some deep learning architectures are known for their exceptional capacity for learning complex structures in large sequences of data, as exemplified by the immediate popularity achieved by large language models [37]. However, these large transformers are prohibitively expensive for most, as they require top of the line hardware to be trained and used as well as vast amounts of data, which is why over the last few years authors exploring smaller transformer architectures with the goal of improving music generation have tried using different styles of music representation.

There are many ways to represent music on the context of deep learning. Some works like [7] and [31] use the whole sound wave as is, which means they can pick up on every nuance and imperfection inherent to human musical performance. This approach is currently not suited for generating music with long-term structure because raw audio files contain a lot of information with few seconds of audio being made up of hundreds of thousands of data points as shown by Figure 1.2.

Most works, on the other hand, use symbolic notations that are not the music piece itself, but instructions on how to recreate it: which notes must be played by which instrument - much like sheet music such as the one seen in Figure 1.1 and others shown in Chapter 2. This kind of symbolic notation requires less memory since it results in shorter sequences, making it more suitable for the goal of creating music with long-term structure. However there are still no deep learning works that are able to consistently create convincing 3 min pop songs or 20 min concertos.

Over the last few years, authors have tried to incorporate structural information

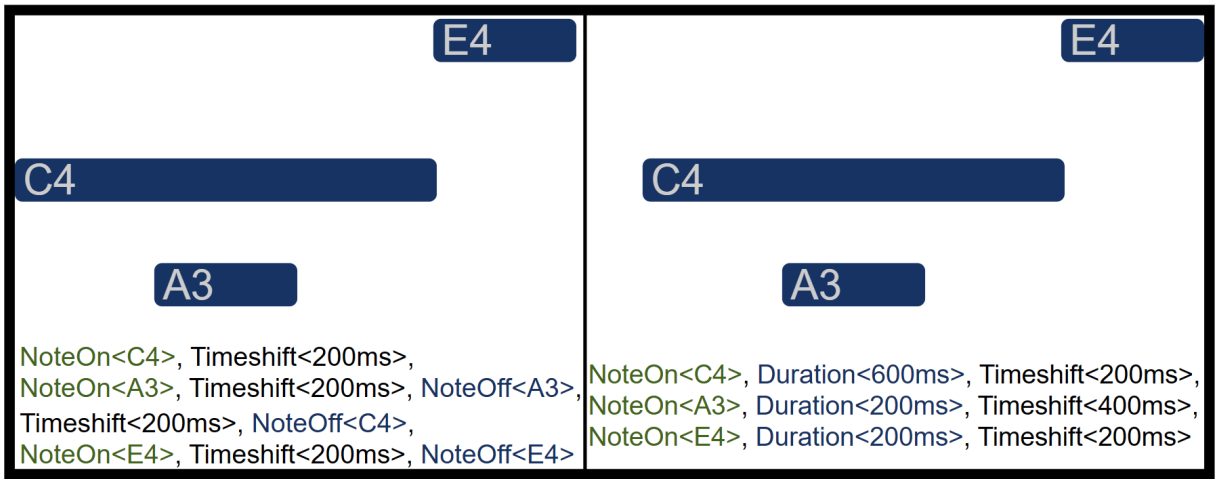

Figure 1.3: An example of the same stretch of music notated on the traditional way (left) and our proposed way (right), showing how our proposed notation does not require note duration to be inferred from the sequence of tokens

into their models by either using external annotations [48] (i.e., annotations of music segments or bars), or changes in the models architecture [42] (i.e., in order to incorporate similarity), and even sometimes both [22]. Despite the effectiveness of this approach, it comes with its own drawbacks given that these annotations still cannot be automatically generated, making large quantities of annotated data expensive as well.

Research Hypothesis: *The driving hypothesis of this dissertation is that simplifying music representation actually provides similar or better results in music generation when compared to adding musical information (via new tokens).*

This hypothesis stems from the fact that, as previously stated, transformer architectures have achieved great performance with other data types (like text) without any additional information. As such, in this work we propose a simpler musical representation that explores explicit note duration as input. To the best of our knowledge, every prior endeavour on transformers for generating music has explored implicit duration, with the exception of the ones using hand-annotated datasets. In order to understand the difference, we refer the reader to Figure 1.3. Instead of using `NoteOn` and `NoteOff` events, we feed the model simply with the `NoteOn` and `Duration` event. When processing the Standard MIDI Format [27], the `Duration` token is pre-computed when processing the musical piece for input. Before playing, we convert back to `NoteOff` events.

What is gained in this notation? It is easy to see that, semantically speaking, nothing is lost. We are still able to capture when each note is generated and for how long it is played. However, this approach has some advantages. From a computational perspective, the model's input (in each line) is mostly composed of triplets, and thus the effect of padding is much smaller[1]. That is, the inference will be less affected by `<pad>`

[1] https://huggingface.co/docs/transformers/pad_truncation. *URLs last accessed on April 2023.*

tokens. Secondly, when capturing the sequences, the attention mechanism [21, 40, 45] does not need to learn that `NoteOff` events are necessary some time after a `NoteOn` event, facilitating learning by shortening the distance between tokens with strong ties to each other and avoiding problems like orphan events without their pair. Thirdly, the proposed notation results in generally shorter token sequences as shown in Table 4.2; and even if this fact results in more unique tokens being needed per piece, the problem of long-term structure is more directly related to sequence length.

If this hypothesis is correct, this representation is promising because it does not need external annotations and is applicable to any MIDI dataset. Finally, we show that structural similarity measures (such as the structureness indicator [48] and the pitch class consistency [42]) are overall improved with this approach, with little to no impact on other metrics such as accuracy. The results are backed by a listening study where listeners are generally better able to identify repeating structure using the new notation.

Overall, we argue that external annotations may not be necessary to incorporate structure into transformer models. This result is beneficial when working on pure MIDI datasets and attempting to generate more extensive sequences with a coherent structure. Also, it is essential to point out that most of the transformer models proposed in recent years, including our major motivators [21, 42, 48, 22], have only explored a single dataset in their analysis. Differently, the method proposed in the present work is evaluated on four datasets: Weimar Jazz Database [34], MAESTRO dataset [19], VGMusic Super Nintendo Entertainment System (SNES) dataset (available online and crawled by the authors), and the POP909 [46] dataset.

The rest of this work is organized as follows. Chapter 2 gives the necessary background in music theory and transformer architectures to discuss the methodology and results, as well as related works. This section is followed by Chapter 3 data which presents the metrics used to compare our results with the current state of the art. Chapter 4 describes the four datasets used in this work. Afterwards, Chapter 5 presents our methodology and results both on quantitative metrics (i.e., structural indicators, entropy, and pitch consistency), as well as our listening study (where over 100 participants evaluated 16 pieces of music, two for each dataset). Chapter 6 concludes our work.

Chapter 2

Background and Related Work

This chapter contains the theoretical background on the two main subjects that comprise this work: music notation and the transformer neural network architecture. Section 2.1 addresses the music notation side of this work, showing what needs to be represented (i.e. how music is structured and organized) and how these components can be represented in different ways.

The next section gives an overview of how the transformer architecture works via a timeline that starts on the early recurrent neural networks and shows how each one of the transformer's constituent parts improves on their performance. Finally, this chapter discusses other works related to music composition using neural networks, how they combine different network architectures and music notation styles and their results.

2.1 Music Theory and Music Notation

This section presents the necessary concepts on music theory and music notation in order to adequately discuss the rest of this work. This chapter starts with the understanding of what defines a musical note followed by the concepts of octaves and pitch classes, the concept of scales and tonality, and after that we give three examples of how music can be represented.

2.1.1 The Musical Note

In order to understand and evaluate musical notation, one must first understand what must be notated; and in the scope of this work, a musical piece can be simplified to be defined as a sequence of notes. A musical note is a sound wave defined by its pitch,

duration, volume and timbre. A note's pitch is the sound wave's frequency, and since this work exclusively addresses western tonal music, pitch does not need to be represented in Hertz as a continuous value but can instead be represented by its note name. Western

	Octaves				
Notes (Hertz)	**1**	**2**	**3**	**4**	**5**
C	32	65	130	261	523
C#	34	69	138	277	554
D	36	73	146	293	587
D#	38	77	155	311	622
E	41	82	164	329	659
F	43	87	174	349	698
F#	46	92	185	369	739
G	49	98	196	392	784
G#	52	104	208	415	830
A	55	110	220	440	880
A#	58	116	233	466	932
B	61	123	246	493	987

Figure 2.1: A reference table with some note names and their frequencies in Hertz rounded to the nearest integers. Source: [33]

tonal musical pieces are built on standardized and named note frequencies, with deviations being considered "out of tune". Figure 2.1 exemplifies some of these names and frequencies and shows that each note's pitch in Hertz is $\sqrt[12]{2}$ times the value of the previous one, with A4 being standardized as having a frequency of 440Hz.

A note's volume is the sound wave's amplitude; the duration is how long the wave should be sustained. The timbre refers to other qualities of the wave related to the source that produced it, ambient reverberations, and others. For example, a flute and a violin playing the same pitch with the same amplitude can still be distinguished by their timbres. Alternatively, the same instrument can produce different timbres via different techniques: violins can be bowed or plucked, guitars sound different if when plucked with short vs long nails, and synthesizers are capable of producing any kind of sound wave, and thus allow for complete control of timbre. This concept can be more easily visualized with the aid of spectrograms, visual representations of the spectrum of frequencies in a signal as it varies with time. Figure 2.2 shows how notes of the same frequency, duration and volume produce different spectrograms when played by different instruments.

Given that musical notes are sound waves, it is natural to anticipate a greater level of intricacy in defining a musical note, as a sound wave cannot be perfectly encapsulated

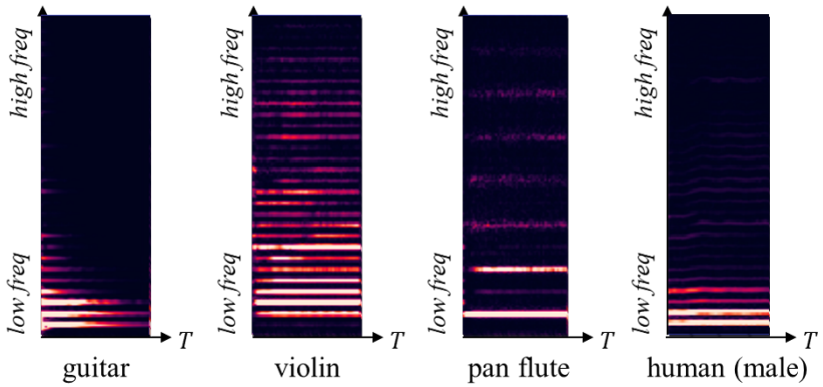

Figure 2.2: Magnitude Spectrograms of four different instruments playing the same note, C4. Source: https://www.arxiv-vanity.com/papers/1912.02591/

by these four characteristics alone. However, for the context of this project, we can streamline the concept of a musical piece as a sequence of notes, and narrow down the attributes of a note to solely these four aspects.

2.1.2 Octave and Pitch Classes

As shown in the previous section by Figure 2.1, in western tonal music there are only 12 note names. These 12 are known as pitch classes. In order to keep increasing or decreasing in frequency, the note names are simply repeated cyclically, and we say that there is one octave between two notes of the same class, such as C3 and C4 or A2 and A3.

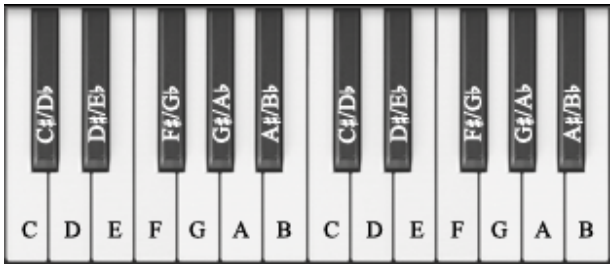

Figure 2.3: Two octaves on piano with the pitch classes shown on their corresponding keys. Source: https://www.musiccrashcourses.com/lessons/chromatic.html

As an example, Figure 2.3 shows how notes are laid out on a piano and their cyclic nature when it comes to pitch classes and octaves. The concept of pitch classes in conjunction with the concepts of scales and tonality presented in the following section are necessary to understand the relevance of some metrics in Chapter 3.

2.1.3 Scales and Tonality

In western tonal music, not all pitch classes are used in every musical piece in the same proportion. Instead, tonal musical pieces are constructed using a subset of all notes that appear more predominantly, while others appear more sporadically. This subset of pitch classes is a musical scale.

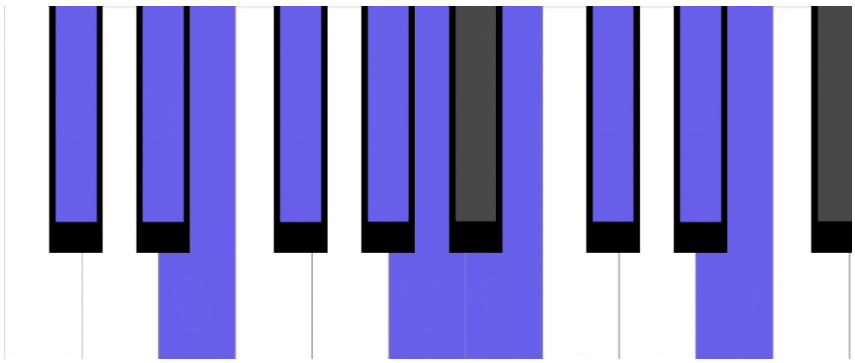

Figure 2.4: A piano with the keys from the E major scale highlighted in blue. Source: modified from https://petersonpianoacademy.com/e-major-scale-piano/

Indiscriminate use of pitch classes outside the scale departs from the concept of tonality and results in musical pieces that are considered exotic for most listeners and thus are outside the scope of this work. Figure 2.4 shows the E major scale as an example: if a musical piece is in the tonality of E major, we can expect that most of its notes will be those found on the E major scale, and we say it's in the key of E major. While the concept of tonality is much more complex and also encompasses expectations and conventions regarding chord progressions, the definition given in this section is sufficient for the purposes of this work.

This concept is important because some of the metrics later discussed in Chapter 3 are essentially ways of measuring if a musical piece follows the concept of tonality or not, and a model incapable of learning the concept of tonality is undesirable in the scope of this work. The following sections show how each type of musical representation notates pitch, duration, volume, timbre, and how each one addresses polyphony - when more than one note is played at a time.

2.1.4 Sheet Music

Sheet Music is the traditional way of representing music, and the most comprehensive among those cited in this section. Sheet music represents musical notes on the structure with five lines called the staff. The staff is read left to right and on it, pitch is represented on the vertical axis with noteheads placed on the upper lines and spaces having higher pitch.

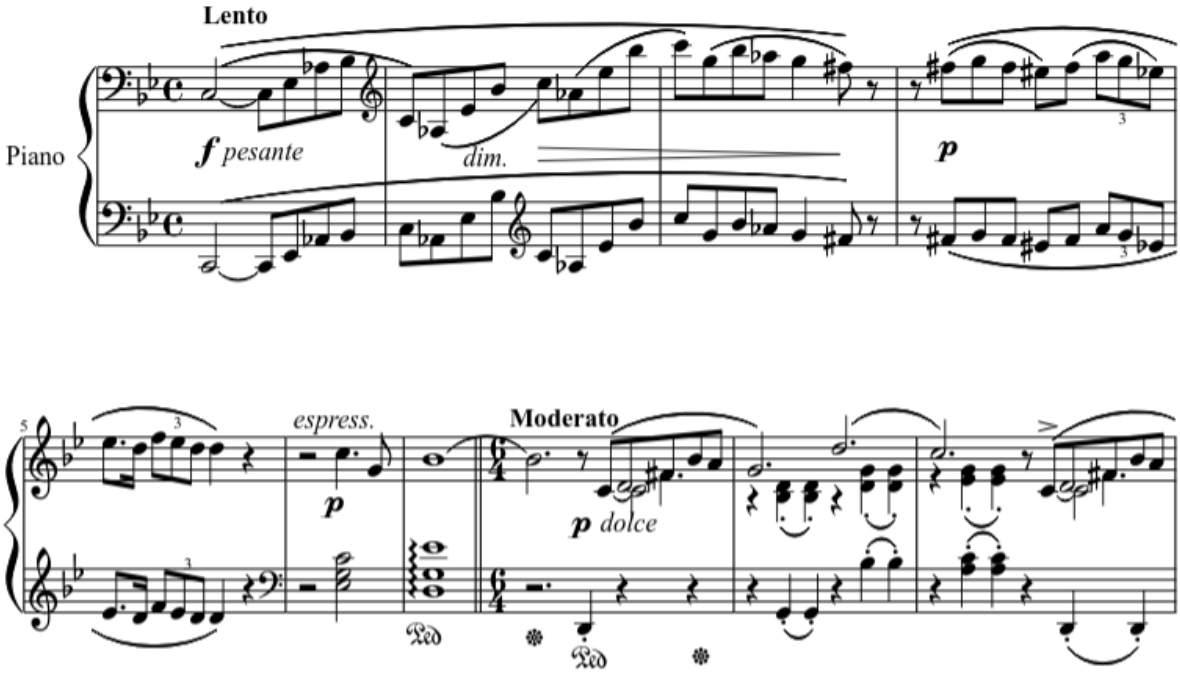

Figure 2.5: The start of Chopin's Ballade No. 1 Opus 23 in G minor. This piece is meant to be played by a solo pianist, but it contains two staves: the lower one indicates what should be played by the left hand, and the upper one what should be played by the right hand. Source: https://musescore.com/classicman/scores/174214

A note's duration is represented by its rhythmic figure, which is determined by factors such as:

- The color of the notehead. Figure 2.5 shows some notes with black noteheads and some with white noteheads.
- The presence or absence of a stem, the vertical line touching the notehead.
- The presence and shape of the flags (the adornments on the stems) or beams (similar to flags, but simplified and joining multiple notes).
- The presence of a dot next to the right side of the notehead.

- Tempo markings dictating how fast the piece should be played, like the word "Lento" on the upper left side of Figure 2.5.

Volume and timbre are represented by various markings on the sheet music. The letters f and p seen between the two staves on Figure 2.5 for example instruct the player to produce louder or quieter notes, and the upper left corner shows that the two staves should be played by a single pianist. Polyphony can be notated with more than one staff or by placing multiple notes on the same staff, with both options being used in the piece on Figure 2.5.

The reason sheet music is presented in this work is to serve as a benchmark, as it is the standard way of representing music. As such, any other music representation style ideally should also be able to address pitch, duration, volume, timbre and polyphony at the same time.

2.1.5 Piano roll

Piano roll is a simpler way of representing music commonly seen on music production software. As the name suggests, it was directly inspired by old player pianos and music boxes and the paper or metallic rolls they read to play music.

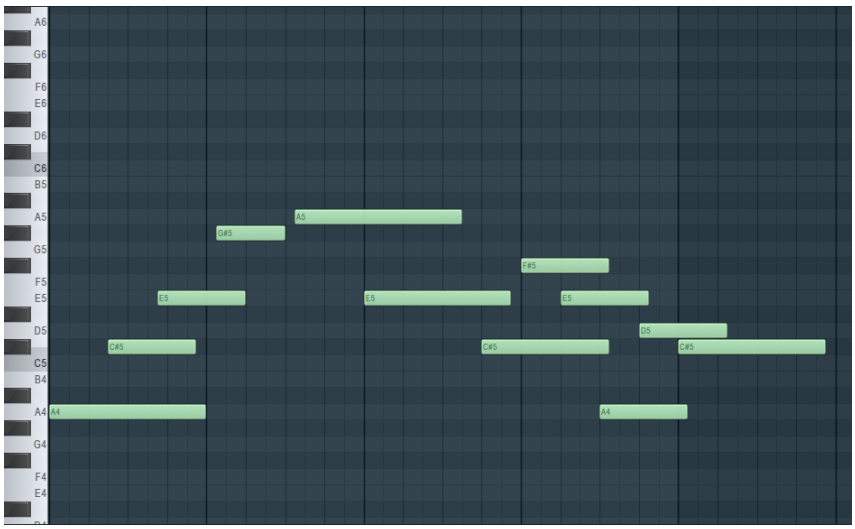

Figure 2.6: A short melody notated using piano roll as it is commonly seen in music production software. Source: https://www.tooboat.com/?p=1150

It is visually intuitive and also has pitch on its vertical axis, with higher meaning higher pitch, but duration is visually represented by the horizontal length of each note. Some works like [11] and [39] use variations of the piano roll as inputs for their neural

networks, building a matrix M where M_{ij} is 1 if the note i is active on time j and 0 otherwise; or alternatively, using a sparse matrix with zeros in every instant when a note is not being activated or silenced. Volume can be notated by making the notes different colors and using a third dimension on matrix M, and as shown in Figure 2.6, polyphony is intuitively represented by placing more than one distinct note at the same time instant.

2.1.6 Event sequence

Event sequences are a style of musical representation used mostly in MIDI files and old music production software called "trackers", and rarely seen outside the context of computing. As the name implies, this type of musical notation consists of a sequence of events of various types, such as `NoteOn` events to activate notes, `NoteOff` events to deactivate notes, Program Change events to change timbre, and others. These events usually have one or more parameters and are associated with timestamps, e.g.: the `NoteOn` event by needs a parameter telling the program which note to activate and the note should be activated at a certain timestamp. The representation used in recent works involving Transformer networks like [21], [48] and [22] are variations of this style of notation, since this architectures excel at processing sequences of tokens.

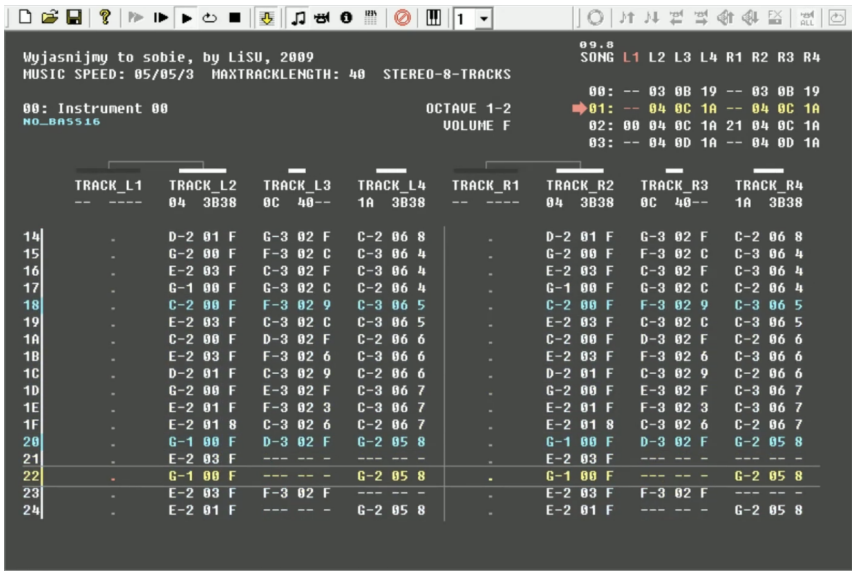

Figure 2.7: RASTER Music Tracker, a software used to write musical pieces for the POKEY chip found on the old Atari. This software represents music with events and timestamps. Source: https://www.youtube.com/watch?v=cm-BvMSgEDA

2.2 Transformers

Up so far we have discussed different forms of symbolic music notation. Moreover, as we mentioned, event sequences (such as MIDI) are the standard commonly explored by neural networks. In this section we present an overview on the transformer architecture, the current state-of-the-art in music generation and the architecture used in this work. Originally presented by [44], transformers quickly gained notoriety for their improved results in large text datasets when compared to even the most advanced variants of RNNs, along with their improved computing performance thanks to their capacity to parallelize calculations instead of being bottlenecked by the recurrence in RNNs. Though not to the same magnitude, transformers presented similar improvements in results for music generation, as event sequences are analogous to tokenized text.

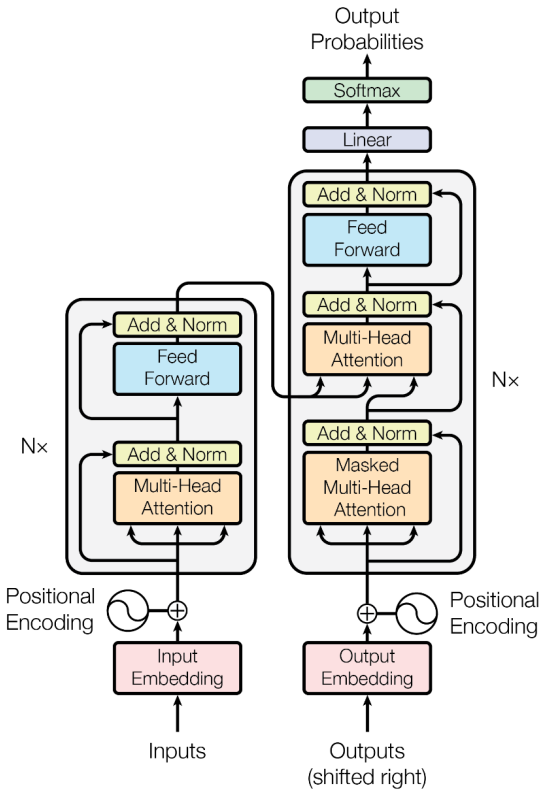

Figure 2.8: The famous illustration of the transformer proposed by [44]. These components and the reason they are improvements on the previous recurrent networks will be explained on the following subsections. Source: [44]

2.2.1 Early recurrent neural networks

As mentioned in the introduction, Transformers like the one initially proposed by [44], shown in Figure 2.8, are the current state of the art in natural language processing tasks. They are the result of many small improvements on architectures which started with the first sequence-to-sequence recurrent neural networks made to handle translation tasks. They work by first letting an encoder process each token of the input sentence sequentially, then its internal state is given as input to a decoder, which in turn produces the tokens of the output sentence sequentially.

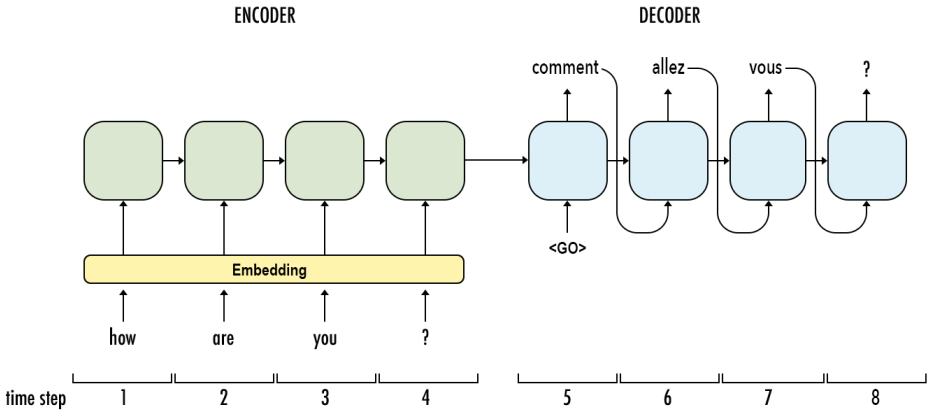

Figure 2.9: An example of the early encoder-decoder recurrent neural networks used for machine translation. When everything works, the encoder is able to encode enough information on its internal state for the decoder to produce the correct output. Source: https://towardsdatascience.com/language-translation-with-rnns-d84d43b40571/

2.2.2 The Attention Mechanism

These early RNN architectures suffer from being slow since they cannot be effectively parallelized because the timesteps must be processed sequentially, and the vanishing gradient problem prevents the encoder from properly encoding all the relevant information in very long sequences. The attention mechanism proposed by [1] helped solve this problem by no longer requiring that the decoder work with a fixed-length vector created by the encoder, but instead "storing" all the encoder's hidden states after processing each token of the input and assigning them trainable weights. This allows the decoder to learn which of those hidden states are the most relevant for generating each step of the output sequence. Figure 2.10 exemplifies how each hidden state h contributes to the value of the context vector, modified by its corresponding alignment weight a.

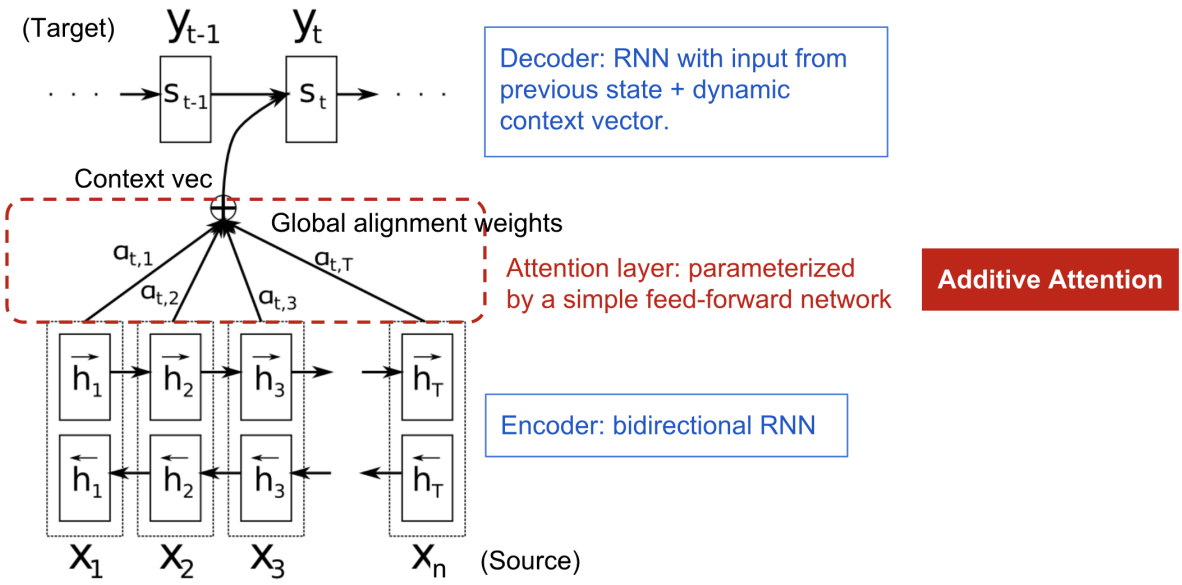

Figure 2.10: An illustration of an attention mechanism. Source: https://lilianweng.github.io/posts/2018-06-24-attention/

2.2.3 Self-Attention and Positional Encoding

Transformers do away with the recurrent aspect of the architecture entirely and rely only on stacked self-attention and feed-forward layers as shown in Figure 2.8. Self-attention, as the name suggests, is a type of attention layer that allows the network to process each input token with the context given by every other input token, learning which ones provide the most useful context. Figure 2.11 shows an example of how context can change the meaning of a token and how self-attention would help with these situations.

At that point, if there is no longer recurrence and the neural network has access to all the inputs simultaneously, the temporal locality of the timesteps in relation to each other is lost. Transformers solve this problem by adding positional information to the embedding of every item on the sequence - this process is referred to as positional encoding.

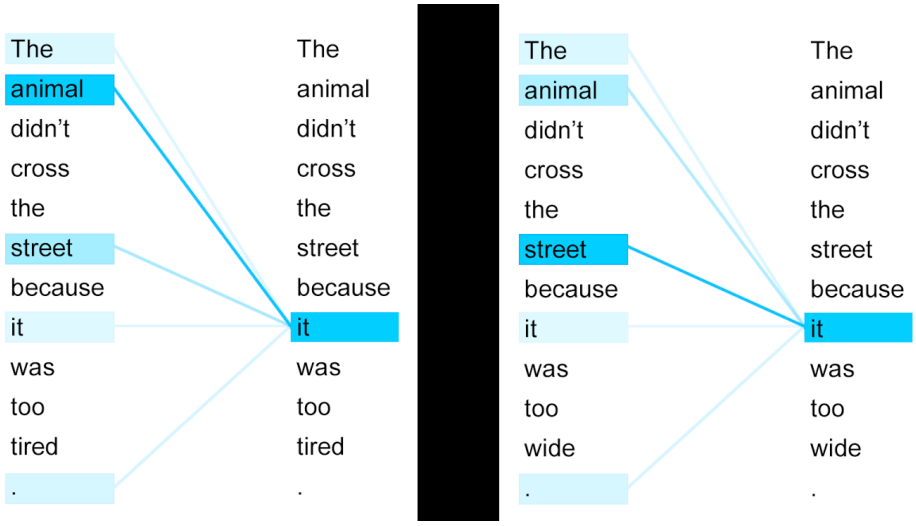

Figure 2.11: An example of how the ability to learn which other input tokens provide the most relevant context might be useful. Arrows with lighter colors represent lower weights and, thus lower relevance. Source: Modified from https://ai.googleblog.com/2017/08/transformer-novel-neural-network.html

2.2.4 Relative Position Representation

Relative position representation (RPR) is a modification to the self-attention mechanism proposed by [41]. Relative position representation assigns weights to tokens according to their position in relation to the token currently being processed instead of their

absolute position within the sequence, as shown in Figure 2.12. Huang et al. argue that in the context of music composition, relative differences are more important than absolute values, and provide experiments to confirm that RPR indeed provides better performance.

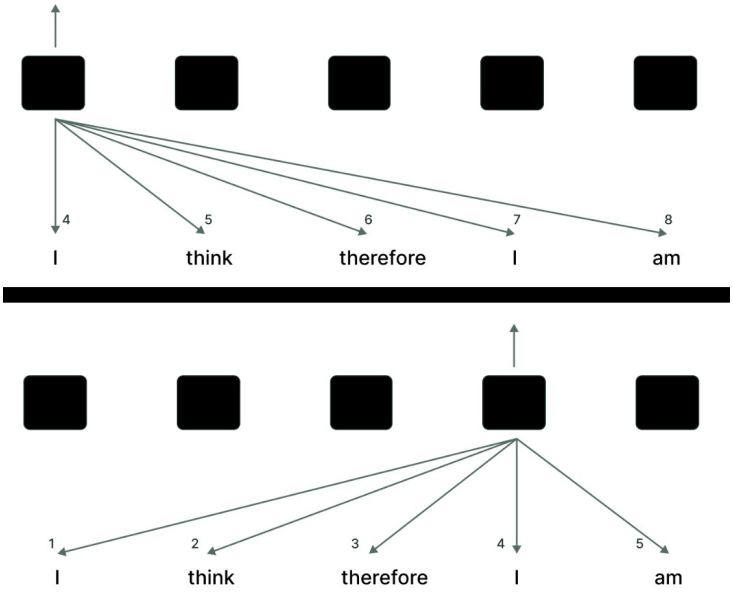

Figure 2.12: A diagram illustrating the concept of relative position representation. This diagram shows, for example, how the parameter used to relate the current token to the next one is always the same independent of their absolute position in the sequence. Source: Modified from https://medium.com/@_init_/how-self-attention-with-relative-position-representations-works-28173b8c245a/

2.3 Related Work

At this point we have enough theoretical background in music notation and transformer neural networks to look at other related works and how changes to notation and architecture affect results. For example, researchers have early realized that the representation of music was not only important to serve as the input to a model, but it was rather critical for improving the ability of these models to learn and consequently output better sequences [27].

Additionally, in a pioneer work, Todd [43] explored recurrent neural networks (RNNs) and generated monophonic melodies. The author also presented a discussion on the advantages of relative pitch representation. Chen and Miikkulainen [4] also produced small music pieces using deep learning with limited results, as the networks could

consider only a limited part of the musical structure.

While encoding music in symbolic file format, the Musical Instrument Digital Interface (MIDI) has been a standard in both industry and academia [32] because of their small file sizes and rich representation capacity. Nevertheless, one-dimensional note threads have been the canonical choice for music representation in Hidden Markov Models and Long Short Term Memory networks (LSTM) [13, 18, 32], with polyphonic music requiring the serialization of the multiple tracks into a single sequence, as adopted by Dong et al. [12]. Alternatively, explicit polyphonic representations have been successfully used by Hadjeres et al. [18] on the DeepBach model, an RNN able to handle choral-like scores described by 4 simultaneous sequences, later expanded by Mao et al. [26] for generating music of specific styles. Additional works with polyphonic notation include the bi-axial LSTM proposed by Johnson [24] that was trained with piano-roll note representation and the RNN framework proposed by Zhu et al. [51] for pop music generation based on chords. More recently, works like WaveNet [31] and Jukebox [7] have tried working on raw audio samples again with increased computational power, and while they show good improvements it's still not possible to create a cohesive structured piece with even a few minutes with these approaches.

Relative positional encoding schemes have shown to be particularly beneficial when applied to music transformers [21] for generating piano music with long-term structure. Originally designed to work with absolute position representations, transformers had their capabilities improved as relative position representation was introduced by Shaw et al. [40], and developed in subsequent works [8, 22, 48] for both pitch and time shifts. Rhyu et al. proposed the STHarm, a transformer in which both the input and output used a piano roll representation with serialized chord labels instead of musical event tokens [38]. Qin et al. applied the self-attention properties of the transformers to favour the generation of longer sequences with stronger structural constraints. In the Bar Transformer model [36], the notes within each bar were pre-processed in order to preserve the long-term dependencies of the music and to produce pieces with extended structural unit.

The quest for longer sequences has been followed by the urgency for a more quantitative evaluation of musical quality. In their work on Jazz music generation [48], Wu and Yang propose a set of objective metrics to assess structuring, pitch usage, rhythm and harmony. Each note was represented by events that encode the many musical aspects, with a reference to the MIDI format, but also considering the bar structure. The attention on bars was also the guideline for the Pop Music Transformer model [22] in which rhythm was represented relatively to the start of each composition cell. By modelling the music as a grid, harmonic structures could be explicitly stated as chord events. Some works like [50] propose many similar metrics, while some others like [15, 16] have proposed comparison-based metrics that determine how similar a musical piece is to a collection of pieces or how similar a collection of pieces is to another collection of pieces. More re-

Only annotation independent structural metrics shown	**Music T.**	**Pop Music T.**	**Theme T.**	**Jazz T.**	**Ours**
Entropy				✔	✔
Pitch Consistency			✔		✔
Structural Indicator				✔	✔
Human Evaluation	✔	✔	✔	✔	✔
Maestro [19]	✔				✔
Jazz [34]				✔	✔
Pop [46]		✔	✔		✔
SNES (ours, similar to [9, 8])					✔
Any MIDI					✔

Table 2.1: Comparison of our work with the Music Transformer [21], the Pop Music Transformer [22], the Jazz Transformer [48], and the Theme Transformer [10]. The evaluation metrics will be further detailed in Section 5.

cently, Jiang and coworkers [23] have proposed a hierarchical model composed of multiple encoders that attempt to learn a latent representation of the bars, while attention blocks capture the global structure of the music. Multi-track representation was further developed by Shih et al. in the Theme Transformer [42], where melody and accompaniment were separately coded, and similarly by Dong et al. [10].

Out of the previous works, our approach is mainly motivated by the Music Transformer [21], the Pop Music Transformer [22], the Jazz Transformer [48], and the Theme Transformer [10]. As shown in Table 2.1, each of these approaches was only evaluated on a single dataset, due to the necessity of exploring external annotations that are not standardized on every dataset. Also, every one of these explores different metrics for evaluations. It is even more problematic that some metrics are only measurable exploring the external annotations, as this does not allow for a complete comparison between several datasets.

Before continuing, we point out that our goal in this thesis is to explore how good transformers are at generating music with a more coherent structure. Unlike previous efforts, we explore both an annotation-independent approach as input and annotation-independent metrics for evaluation. This choice does not allow a comparison with other approaches on every metric proposed by the authors. We shall, however, discuss in our study how our scores differed from the authors when a similar metric was used.

Chapter 3

Evaluation Metrics

Regarding metrics for evaluating deep learning applied to music generation, objective quantitative evaluation of musical quality is still currently an open problem with no standardized benchmarks. Even so, there are certain conventions and patterns expected of western tonal musical pieces that can be more easily measured and used as a reasonable proxy for musical quality. Especially considering that the current state-of-the-art works in music generation are just barely beginning to learn these patterns and expectations, thus making complex artistic evaluations less important than attesting if the results are capable of following even these traditional conventions. Works like [48, 50, 15, 16] propose a few ways to reduce the complexity of the concepts presented in the previous section to measurable formats, some are complexity measurements focused on a specific musical aspect like tonality, rhythm or musical structure while others are comparison based, measuring similarity between pieces. This chapter gives theoretical background on the metrics used to evaluate the results of this work, both from the mathematics perspective (how these metrics are calculated) as well as from the musical perspective (why are these metrics sensible for measuring musical quality).

3.1 Structureness Indicators

As mentioned in the introduction, generating music presents unique challenges due to the nature of its self-similar structure. The catch is that this structure must contain enough variety to not be boring, while also avoiding overwhelming the listener with too many musical ideas.

With the intention of measuring the presence or absence of such self-similar structures, this work uses the structureness indicators proposed by [48] to evaluate results. Structureness Indicators (SI) are extracted from the fitness scape plot [27], which is in turn derived from the self-similarity matrix.

The fitness scape plot shows in essence how much every possible sub-segment

of a musical piece explains every other sub-segment with a fitness measure $\varphi \in [0, 1]$, and it does this for segments of every possible length and starting point. To explain the calculation of this fitness measure as proposed by [28], the concept of path families must first be introduced. In a self-similarity matrix $\mathcal{S}$, a path is a sequence of cells separated by a distance within a set of admissible step sizes (in [28] this set is defined as $\Sigma = (1, 2), (2, 1), (1, 1))$, and path's score is simply the sum of all values on cells that make up that path.

For a segment α, the path family $\mathcal{P}$ consists of paths π with projections coinciding with α as illustrated in Figure 3.1, and the score of a path family is simply the sum of the scores of its paths. And so, the score of a *segment* is the score of the path family of maximal score. This is not the fitness measure yet.

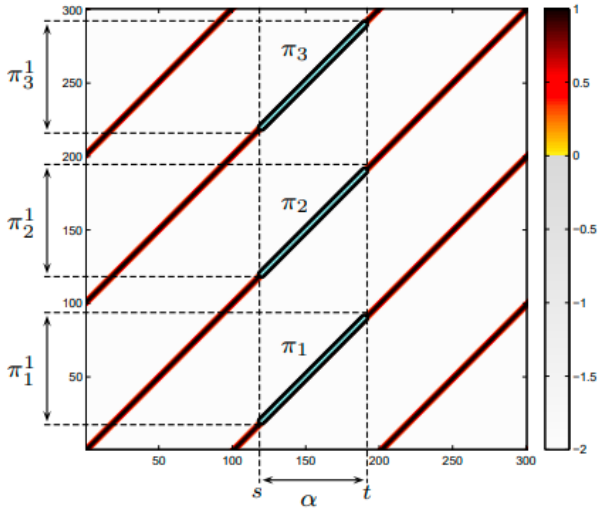

Figure 3.1: An example of a path family consisting of $\pi 1, \pi 2, \pi 3$ for the segment $\alpha = [120 : 190]$. Source: [28]

Next, the authors of [28] define a *normalized score* $\bar{\mu}(\mathcal{P})$ of a path family $\mathcal{P}$ over segment α as

$$\bar{\mu}(\mathcal{P}) := \frac{\mu(\mathcal{P}) - |\alpha|}{\sum_{k=1}^{K} L_K} \tag{3.1}$$

for every for every path of index k in $\mathcal{P}$. Then, a measure of *coverage* or recall is defined as

$$\bar{\gamma}(\mathcal{P}) := \frac{|\gamma(\mathcal{P})| - |\alpha|}{N} \tag{3.2}$$

where N is the size of the self-similarity matrix and $\gamma(\mathcal{P})$ is the union of all segments defined by the projection of the paths in $\mathcal{P}$. Then, inspired by the f-measure, the fitness φ of a *path family* is defined as

$$\varphi(\mathcal{P}) := 2.\frac{\bar{\mu}(\mathcal{P}).\bar{\gamma}(\mathcal{P})}{\bar{\gamma}(\mathcal{P}) + \bar{\mu}(\mathcal{P})} \tag{3.3}$$

and so, the fitness φ of a *segment* is defined as the fitness value of the score-maximizing path family $\mathcal{P}$.

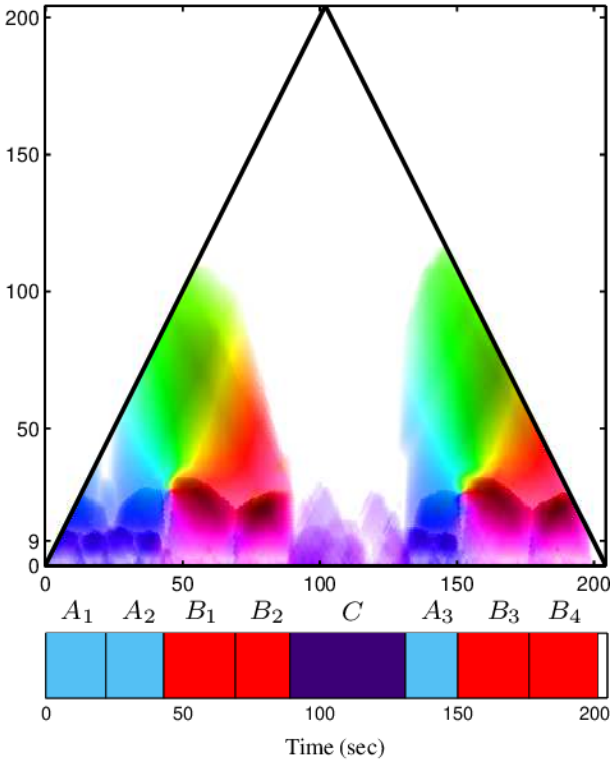

Figure 3.2: The fitness scape plot of the piece shown in Figure 1.1, with the sections shown below the plot on the corresponding timestamps. Source: [29]

With the fitness measure defined, the fitness scape plot uniquely identifies every possible sub-segment of a sequence by its length and its centre (its median point). For example, in a musical piece represented by 20 tokens, the sub-segment from token 2 to token 6 could be uniquely identified as the segment with length 5 centred on token 4. Figure 3.2 shows the fitness scape plot, with segment centre on the horizontal axis, length in the vertical axis, and colour intensity representing the fitness. In Figure 3.2 specifically, hue is used as an aid in visualizing different sections of the piece. The plot's peculiar triangular shape comes from the fact that using this way of identifying segments, the longest possible segment (the piece itself) is the segment of length L centred on $L/2$.

A more intuitive understanding of the fitness scape plot can be obtained by noting that the gap on the centre of the plot corresponds to section C of the Brahms piece on Figure 1.1, the two taller peaks on either side correspond to the A+B sections that repeat themselves, and the smaller peaks correspond to the individual A and B sections. Just as in [48], this work measures SI for short-, medium-, and long-term duration using the

same 3 to 8-second intervals (short-), 8 to 15 intervals (medium-), and 15-second intervals (long-). We ran this metric using the originally proposed code for each of the generated pieces.

3.2 Pitch Class Entropy

Next, we turn our attention to the Pitch Class Entropy. This metric is simply Shannon's entropy, defined as Equation 3.4 when measured on the 12-dimensional pitch class histogram (we use logarithm of base 2 in our computation). These pitch class histograms can be easily obtained by going through every `NoteOn` event and counting the pitch classes. As an example, Figure 3.3 shows the pitch class histogram for a famous work of Johann Sebastian Bach.

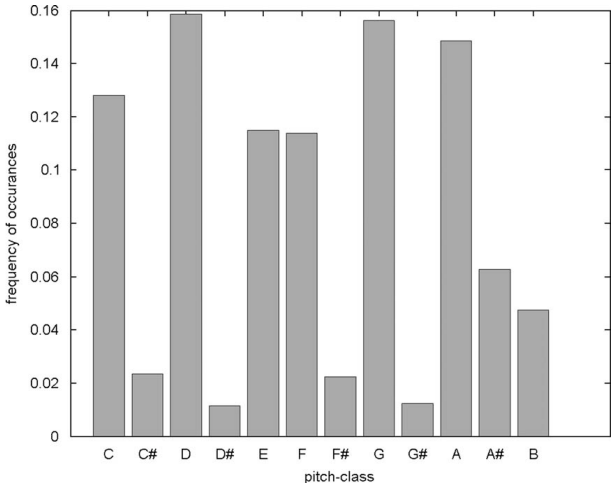

Figure 3.3: Pitch class histogram of J.S. Bach's C-major Prelude. Source: [17]

$$H(p) = -\sum_{i=1}^{12} p(i)\log_2(p(i)) \tag{3.4}$$

In this equation, $p(i)$ represents the probability of the random variable I (the 12 pitches) taking on a particular value i. The sum is taken over all possible values of I. This metric helps to gauge the overall conformity to tonality of the piece. A lower entropy indicates that some of the pitch classes dominate the histogram, and suggests that the piece has a well-established key; though very small values could also indicate excessive repetitiveness. On the other hand, a higher value of entropy indicates that all pitch classes appear with roughly the same frequency and there is no clear dominance of any key; though it could also just be that the piece modulates (changes key) or just

happens to have melodies constructed using techniques like chromaticism (use of pitch classes outside the key). We compute this entropy for every primer and generated piece and report CIs on Table 5.4.

3.3 Pitch Class Consistency

The Pitch Class Consistency metric was designed to determine the consistency of tonality along the piece, helping with some of the ambiguity inherent to the pitch class entropy. It is computed via the Kullback-Leibler divergence, defined in Equation 3.5, between consecutive Pitch Class histograms when a musical piece is split in equal-sized windows. If a piece is divided in N intervals, that are $N-1$ divergences computed in that piece. We here report the CI of the mean divergences.

$$D_{kl}(p \mid\mid q) = -\sum_{i=1}^{12} p(i) \log_2(q(i)/p(i)) \quad \text{where } p \text{ is the current interval and } q \text{ the past one} \tag{3.5}$$

The Kullback-Leibler divergence in essence measures how different two distributions are. If two sections of a musical piece have very different pitch class histograms, this may indicate that tonality is not well established or alternatively that the piece has different keys in these two sections. On the other hand, similar histograms may indicate a consistency in the established tonality or if the histograms happen to be a uniform distribution of pitch classes, it could actually indicate the absence of the concept of tonality. These ambiguities mean that the pitch class entropy and consistency are better evaluated in conjunction. In [49], the consistency was measured from bar to bar, but since our notation does not include bar tokens, in this work we decided to split each piece in 8 equal-sized intervals (similar results were obtained when splitting in 16 intervals). A small value of divergence means that there is little variation in terms of tonality throughout the piece, which, at first thought, implies cohesion. However, a piece that is too consistent can also be overly repetitive, which is not what we are striving for. Also, musical pieces with modulations may have large values for divergence even if they have clear and consistent tonalities. Thus we again compare results across notations. The results are shown in Table 5.5.

3.4 Subjective Evaluation

When evaluating generated music, human evaluation is still a necessity, as there is not (and it is likely that there never will be) an objective metric to measure musical quality. In this section, we describe how the listening study was set up. Initially, 16 random pieces of music were chosen at random from our total of 240 generated pieces (30 pieces per 8 models). This choice was randomly selected, with no human interference, and pieces contain both primer and generated tokens. With these 16 pieces, we set up an online website where evaluators were presented with one of such pieces at random. Evaluators could evaluate how many pieces they wanted (up to 16), being one piece presented per page (randomly selected). Accompanying each piece was the following form, mostly identical to the one from [49], where evaluators were required to answer:

"In this form, you will answer questions regarding a small piece of music (song). The music that you will hear has a small initial part composed by human beings. The exact size of the human-composed snippet is not disclosed to participants, so please do not assume any value. Just know it is a few seconds. After this initial excerpt, the rest of the song (a few minutes) was generated by AI.

Musical pieces are presented in a simple format called MIDI. So for your answers, please try not to make a comparison with professionally recorded music. Knowing this information, please let us know if you identify themes that are repeated in the song, in addition to your general opinion of it, according to the following questions (answers to all questions are mandatory):

- **Overall Quality (O):** Does it sound good overall?
- **Impression (I):** Can you remember a certain part or the melody? That is, is the major theme of the initial seconds present in the overall piece.
- **Structureness (S):** Does it involve recurring music ideas, clear phrases, and coherent sections? This recurrence may occur in any part of the piece.
- **Richness (R):** Is the music diverse and interesting?"

  Results are presented in Section 5.3.

Chapter 4

Datasets

As stated, the proposed method was evaluated on four distinct datasets covering different musical genres. This choice allows for verifying whether the proposed notation presents gains for more than one style of music. Before detailing the datasets, it is essential to point out that not only do they cover different genres, but also, these four datasets represent different styles of musical compositions and transcription styles. That is, among these four datasets, there are pieces collected from performances on a physical piano, pieces written by non-professionals using computer software, transcriptions of real performances, as well as professional arrangements of existing musical pieces. There are compositions with many unique elements, complex musical phrases and structures and others with more straightforward and repetitive themes, as well as some that are improvisations.

These datasets are described below and summarized in Table 4.1 for the traditional notation and in Table 4.2 for the proposed notation. For each set of data, we show the total number of music pieces (files), as well as how many pieces were used for training, validation and testing. Moreover, we also show the number of MIDI tokens for train, validation, and test files combined. This is referred to as the Average Length of pieces. Finally, we also show the average number of unique MIDI tokens in each file.

Given that the evaluation is focused on structure, we condense each musical piece into a single track (as was done in most previous papers [21, 22, 48]). Thus, we remove any tracks with percussion instruments from the MIDIs and combine instruments in a single piano track. Moreover, both the original MIDI and our notation are compared when trained with the off-the-shelf Music Transformer [21]. To train the models, we employ the author-recommended hyper-parameters, with the validation sets (here presented for completeness) being only used to report the validation loss.

	Maestro	SNES	Jazz	Pop
Total Files	1282	6591	455	909
Train Files	967	5937	413	849
Val. Files	137	501	29	36
Test Files	178	154	13	24
Avg. Length	22820	7316	2147	6831
Avg. Uniques	231	95	125	149

Table 4.1: Some characteristics of the datasets on the traditional notation, including average length and number of distinct unique tokens in each piece. The last two metrics are calculated on the sequence of tokens that describe the piece.

	Maestro	SNES	Jazz	Pop
Total Files	1282	6591	455	909
Train Files	967	5937	413	849
Val. Files	137	501	29	36
Test Files	178	154	13	24
Avg. Length	22608	5641	1692	6119
Avg. Uniques	267	70	140	186

Table 4.2: Some characteristics of the datasets on the proposed notation, including average length and number of distinct unique tokens in each piece. The last two metrics are calculated on the sequence of tokens that describe the piece.

4.1 Maestro

The Maestro Dataset [19], here only referred to as Maestro, contains a collection of classical piano pieces performed by contestants of the International Piano-e-Competition[1]. Their performances are captured by an acoustic piano with sensors on the keys, which means an event-based notation can be built directly from the live performance, with all its nuances and variances. This dataset represents the classical musical style, in the colloquial sense of the name, with artists like Bach, Mozart, Debussy and others from similar time periods all present. As shown in Table 4.1 and in accordance with common knowledge about classical music, these pieces are long and varied, often being made up of long sections with many motifs. These characteristics are also visible in Figure 4.1, which also illustrates how despite containing structured repetition, they have more novelty than pieces from the SNES or Pop datasets.

[1] https://magenta.tensorflow.org/datasets/Maestro

	Maestro	SNES	Jazz	Pop
Pitch Class Consistency	0.26	0.51	0.68	0.21
Pitch Class Entropy	3.34	2.88	3.34	2.82
Structureness Indicator Short	0.37	0.38	0.29	0.45
Structureness Indicator Medium	0.36	0.35	0.18	0.48
Structureness Indicator Long	0.35	0.35	0.13	0.51

Table 4.3: Some musical characteristics of the datasets.

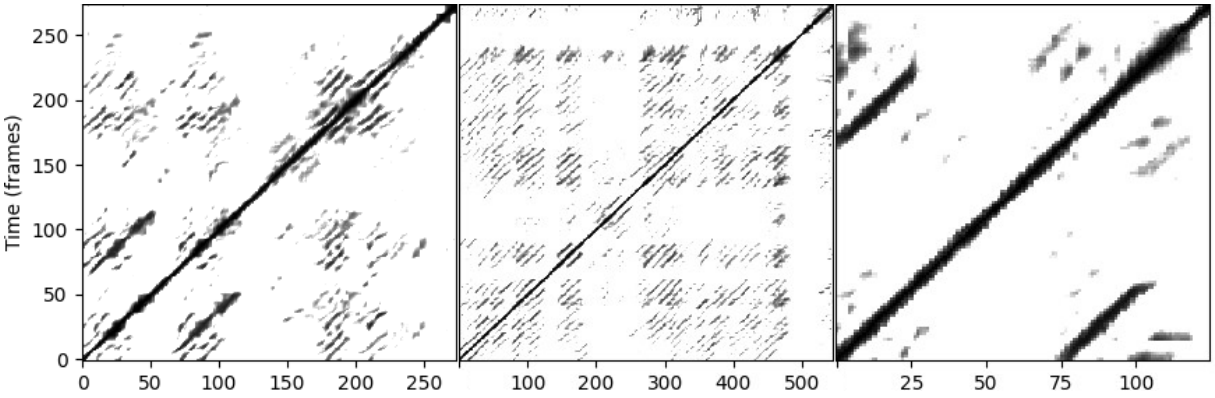

Figure 4.1: Self-similarity matrix of three random pieces from the maestro dataset.

4.2 SNES

The VGMusic Super Nintendo Entertaiment System (SNES) Dataset, or simply SNES, contains MIDI files of soundtracks from various SNES games[2]. This dataset was crawled by the authors of this study. The SNES dataset is similar in nature to the Nintendo Entertaiment System (NES) Music Database (NES-MDB) [9, 8]. Nevertheless, we work with SNES MIDIs not only to provide a novel dataset, but also because the VGMusic website provides video game soundtracks sequenced by humans. NES-MDB was automatically extracted and we found that the MIDI representation of several pieces did not perform well with the Music Transformer. These pieces all come from a very narrow time period and have a similar artistic style informed by the limitations of the hardware for which they were built. In essence, this dataset is characterized by its repetitive pieces with few motifs, and by the fact that it contains pieces densely packed with many notes in rapid succession that cannot be played by humans. Figure 4.2 makes it easy to see how these pieces are comprised of very few novel sections when compared to every other dataset.

[2] http://vgmusic.com/music/console/nintendo/snes/

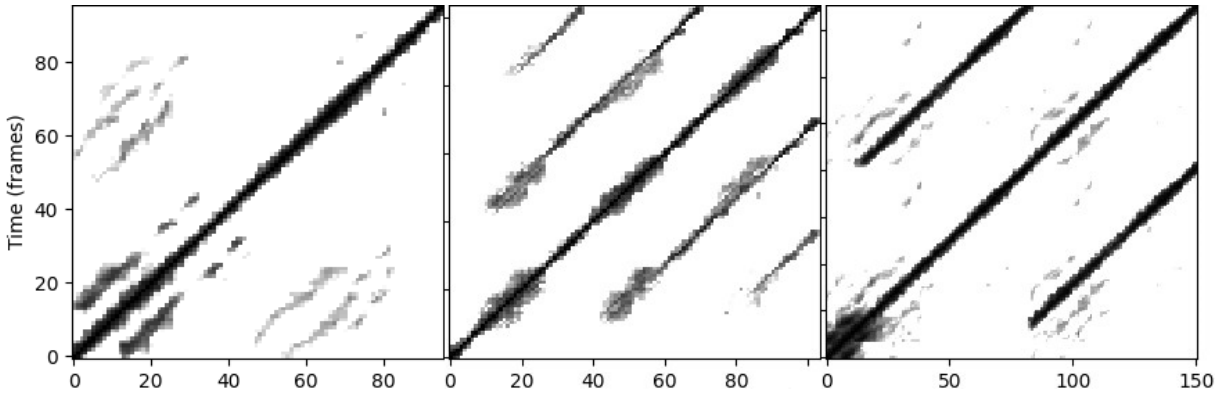

Figure 4.2: Self-similarity matrix of three random pieces from the snes dataset.

4.3 Jazz

The Weimar Jazz Database [34], or simply Jazz, contains transcriptions of jazz solos[3]. As usual in jazz, these are mostly improvisations, transcribed to MIDI files. Since only the solos are transcribed and they're mostly played on brass instruments that can only produce one note at a time, this dataset on average contains the shortest pieces with few tokens, but their improvisational nature makes their structure less repetitive, as evidenced by Figure 4.3 showing how the self-similarity matrix of these pieces consist mostly of only the diagonals.

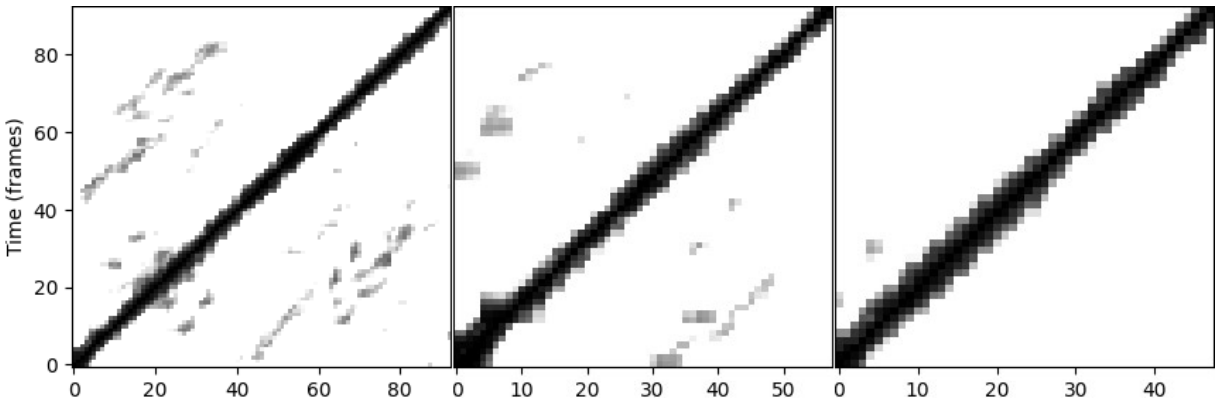

Figure 4.3: Self-similarity matrix of three random pieces from the jazz dataset.

[3] https://jazzomat.hfm-weimar.de/dbformat/dboverview.html

4.4 Pop

Finally, the POP909 Dataset [46], referred to as Pop, contains professional piano arrangements of popular songs from the 1950s to the 2010s[4]. The authors hired professional musicians not only to write the arrangements but also to review other musicians' arrangements to ensure high data quality. However, this dataset's main selling point, the extensive annotations of musical information like beats and chords, are not used in this work. Pop songs are a useful middle ground in length, complexity, repetition and note density, with well-defined sections, as evidenced by Figure 4.4 and how it shows many sections very clearly defined when compared to the other three datasets.

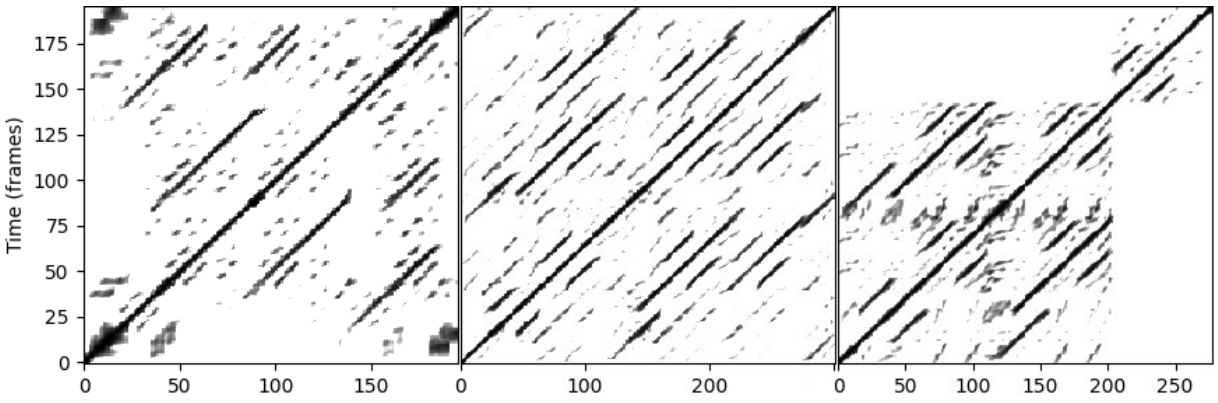

Figure 4.4: Self-similarity matrix of three random pieces from the pop dataset.

[4] https://github.com/music-x-lab/Pop-Dataset

Chapter 5

Methods and Experiments

After presenting the music notation, architecture and datasets used in this work, the experiments performed on this work can now be accurately understood and evaluated. Hence, this chapter contains all experiments performed and the results obtained, divided into sections containing standard training and validation metrics applicable to most machine learning models, structural metrics more directly related to music, and finally a subjective listening evaluation.

As a reminder, our experiment consists in using the Music Transformer [21] as a baseline, and substituting their music representation with ours, exemplified in Figure 1.3. We compare the original notation with our notation using the Music Transformer [21]. In our experiments, we used a PyTorch implementation of the model[1] that was trained with the recommended hyper-parameters (see [21]). The original and new notation models were trained for 100 epochs for the Maestro dataset. For every other dataset, the models were trained for 20 epochs. We have eight models, one for each dataset and input representation.

After training, we generated 30 pieces per dataset[2]. Those 30 pieces were generated by randomly choosing 10 files from the test dataset, giving the model the first 256 tokens of those files as primers, and letting the model generate 3 continuations for each one by completing the sequence until it has 2048 tokens. These primers range from 3 (SNES) to 42 (Maestro) seconds. Generated pieces range from 13 (SNES) to 254 seconds (Maestro).

Throughout the rest of this section, we shall present our comparisons for each notation on each metric for each dataset. Section 5.1 presents a small discussion on our training/validation losses and accuracy. These metrics are usually reported and monitored when training deep models. However, such measures are not the focus of our study as they do not capture musical semantics. Thus, in Section 5.2, we discuss metrics that are related to song structure. Section 5.3 presents our listening study.

[1] https://github.com/gwinndr/MusicTransformer-Pytorch

[2] https://drive.google.com/file/d/1qZp4wAxqcgE2w-u4h9xZnKw1TY3cyIY-/view?usp=sharing

	Original		**New Not.**	
	Train.	Val.	Train.	Val
Maestro	1.78	2.00	2.12	2.29
SNES	0.93	0.95	0.82	0.86
Jazz	2.00	2.10	2.58	2.64
Pop	2.04	2.23	2.58	2.69

Table 5.1: The training and validation losses of the model considering the original and new notations.

	Original		**New Not.**	
	Train.	Val.	Train.	Val
Maestro	0.46	0.41	0.40	0.37
SNES	0.75	0.75	0.79	0.78
Jazz	0.39	0.37	0.22	0.23
Pop	0.42	0.38	0.33	0.32

Table 5.2: The training and validation accuracy of our model considering the original and new notations.

5.1 Training and Validation metrics

Although loss and accuracy (defined as how often the model predicts the next token of a musical piece correctly) do not directly translate do musical quality, these metrics are included for the sake of completeness. Table 5.1 shows models' training and validation losses, and Table 5.2 shows accuracy. From both tables, we can see that both loss and accuracy are comparable in both notations. Regarding accuracy, when we compare the validation columns, there is a slight downgrade in performance for the new notation in the Maestro dataset (0.04). SNES accuracy is slightly improved, from 0.75 to 0.78. Overall, this dataset has the highest accuracy (most likely due to its fewer unique tokens per song, as shown in Table 4.1). The most considerable downgrade in performance we encounter is on the Jazz dataset, where this loss in performance is over 0.14 in both train and validation sets. Pop music has a similar effect as the Jazz data.

Even though we encounter some downgrades in accuracy, as we shall discuss in the following sections, the structureness indicator metric is improved in our notation. Also, in our listening study, both notations achieve statistically equal rating. More importantly, metrics such as loss and accuracy do not capture musical semantics (structure).

	Original			New Not.			Improvement		
	Short	Medium	Long	Short	Medium	Long	Short	Medium	Long
Maestro	(0.32, 0.37)	(0.18, 0.23)	(0.07, 0.13)	(0.31, 0.37)	(0.19, 0.27)	(0.10, 0.19)	-0.56%	9.08%	29.69%
SNES	(0.30 0.39)	(0.14, 0.25)	(0.08, 0.17)	(0.37, 0.45)	(0.25, 0.36)	(0.16, 0.26)	17.44%	61.93%	71.56%
Jazz	(0.27, 0.29)	(0.23, 0.24)	(0.16, 0.19)	(0.28, 0.31)	(0.25, 0.29)	(0.20, 0.24)	3.54%	10.49%	22.92%
Pop	(0.29, 0.32)	(0.21, 0.25)	(0.11, 0.16)	(0.34, 0.42)	(0.28, 0.37)	(0.23, 0.31)	23.85%	40.83%	97.86%

Table 5.3: Structureness Indicators for the original and new notation models. The values are expressed in mean confidence intervals with a confidence level of 95%. Overall, the new notation model shows the most improvement in the long-term.

	Original		New Not.	
	Real	Generated	Real	Generated
M	(3.00, 3.35)	(3.04, 3.17)	(2.99, 3.34)	(3.09, 3.28)
S	(2.36, 2.91)	(1.55, 2.03)	(2.38, 2.92)	(1.96, 2.34)
J	(3.18, 3.38)	(3.35, 3.41)	(3.18, 3.38)	(3.41, 3.47)
P	(2.76, 2.91)	(2.65, 2.73)	(2.76, 2.91)	(2.87, 2.97)

Table 5.4: Pitch Class Entropy of the music generated by the original and new notation models. The values are expressed in mean confidence intervals with a confidence level of 95%. Only the first letter per dataset shown.

5.2 Structural Metrics

Following up on loss and accuracy, three structural metrics were used to evaluate models. Namely, Structureness Indicators and Pitch Class Entropy, also explored by the Jazz Transformer [48], and the Pitch Class Consistency, explored by the Theme Transformer [42].

The Structureness Indicators (SI) are extracted from the fitness scape plot [27]. This plot extracts the degree of repeated structures on a musical piece and is extracted from the self-similarity matrix (SSM). The measure ranges from 0 to 1, with higher values indicating more repetition. We refer the reader to 5.2 for details on how SI is measured. Moreover, as proposed by the original authors, SI is measured for short-, medium-, and long-term duration. In this sense, we used 3 to 8-second intervals (short-), 8 to 15 intervals (medium-), and 15-second intervals (long-). We ran this metric using the originally proposed code for each of the generated pieces. In order to obtain more representative results, given the randomness involved in creative computing using deep learning, we bootstrap [3][47] from the generated pieces by re-sampling 9999 times and using the bias-corrected and accelerated method by [14]. A bootstrap mean confidence interval (CI) with a 95% confidence level is reported.

The results for SI are shown in Table 5.3. On this table, we not only show the SI

[3]Bootstrap CIs were chosen as these metrics do not have statistical tests related to them.

	Original		New Not.	
	Real	Generated	Real	Generated
M	(0.33, 1.07)	(2.10, 2.93)	(0.31, 1.19)	(1.79, 3.28)
S	(0.48, 1.35)	(1.13, 2.31)	(0.47, 1.24)	(0.91, 1.64)
J	(1.22, 3.00)	(0.77, 0.99)	(1.23, 3.05)	(0.47, 0.67)
P	(0.16, 0.37)	(0.91, 1.32)	(0.16, 0.38)	(0.77, 1.02)

Table 5.5: Pitch Class Consistency of the music generated by the original and new notation models. The values are expressed in mean confidence intervals with a confidence level of 95%. Only the first letter per dataset is shown.

confidence intervals (low, high), but also the relative improvement (in percentage) for the mean SI when comparing the new notation to the original one. Underscores represent cells where statistical improvement (intervals with no overlap) are observed when comparing notations for the same dataset and duration. Initially, we point out that the new notation shows improvements overall. A higher impact is present when measuring SI on the long-term. Nevertheless, statistical significance is only present on the Jazz and Pop datasets (where improvements range from 10% to 97%).

These results are interesting for two reasons. Firstly, the overall improvements are higher than those reported by [48] (we used the same code) for Jazz (0.27, 0.18, and 0.14, for short-, mid-, and long-term). This indicates that our approach is actually better than using complex notations for SI. More curious, is that our results increase as the song becomes longer. This is likely because our models do not explore diversity annotations as the Jazz Transformer does.

Next, we turn our attention the Pitch Class Entropy results, keeping in mind what these values may represent, as discussed in Section 3.2. From the table, there is only one *small* statistical difference (Pop) comparing the original and new notations. This indicates that both approaches, overall, perform just as well (be it in a good or bad manner), thus showing that the new notation will not reduce this score.

Finally, we present the Pitch Class Consistency results, shown in Table 5.5. Similar to the results for Entropy, overall notations perform equally, with one exception occurring for Jazz pieces. Overall, our approach presents an increase in Pitch Class consistency similar to the one observed by the Theme Transformer [42] (though the authors are not clear on which divergence metric they use).

So far, our notation improves SI without major drawbacks. However, the metrics we discussed only tell one part of the story. Optimizing for any metric may lead to overly repetitive (or on the other extreme, random) pieces of music.

	O	I	S	R
New Notation	2.61	2.93	3.58	2.52
Original Not.	2.56	2.84	3.30	2.50

Table 5.6: Average Results for the listening study per notation. Questions were rated on Likert-5 Scale. Statistical significance (under a MannWhitney-U test), with a significance level of $p < 0.05$, is shown in underscores.

		O	I	S	R
Maestro	**New**	3.79	3.38	3.38	3.71
	Ori.	3.26	3.19	3.55	3.16
SNES	**New**	1.62	2.26	3.85	1.44
	Ori.	1.86	2.71	3.39	1.79
Jazz	**New**	2.42	3.31	3.23	2.54
	Ori.	2.50	2.71	3.21	2.58
Pop	**New**	3.35	3.20	3.75	3.20
	Ori.	2.55	2.64	2.95	2.41

Table 5.7: Average results for the listening study per dataset and notation. Questions were rated on Likert-5 Scale. Statistical significance, when comparing New and Original notations using a MannWhitney-U test, are shown in underscores ($p < 0.05$).

5.3 Listening Evaluation

Our evaluation was answered by a total of 123 participants (tracked by e-mail) recruited from social circles. The authors of this thesis did not evaluate any piece nor interfered with evaluators. We stopped our evaluation when at least 100 evaluations were in for each notation (100 for the original and 100 for the new). When we consider a tuple of notation and dataset, each pair was evaluated by at least 20 participants. The whole study lasted two days. In Table 5.6 we present the average result for each question when aggregated by dataset only. Statistical significance of results, that is if the distribution of answers differs per question, was tested with a MannWhitney-U test that is recommended for this style of data [6]. Statistical difference is shown in underlines. Here, we can initially see that overall scores are low. This is somewhat expected as we did not cherry-pick any musical piece for the study. Nevertheless, listeners are able to identify more structure on the new notation (providing evidence towards our hypothesis).

In Table 5.7 we break down scores per dataset. Here, we can initially see that the new notation has five statistical wins against two for the original notation. When considering structure (S), results are mostly achieved on SNES and Pop (that are more repetitive by nature). Nevertheless, it is interesting to see that under the new notation,

Maestro and Pop pieces are overall of more quality (O). Finally, Jazz notations cause an overall larger impression (I) under the new notation. All of these results support our hypothesis.

Chapter 6

Conclusion and Future Work

In this work, we discussed different ways of representing music, how they work and their pros and cons when it comes to music generation using deep learning; we presented the concepts of musical structure and tonality and how some metrics can be used to measure these characteristics of western tonal music; and we demonstrated how a simple change to input notation impacts the structural quality of AI-generated musical pieces. Instead of the common `NoteOn` and `NoteOff` tokens, our proposed method consisted of simply replacing these two tokens with a `NoteOn` and `Duration` tokens directly adjacent to each other on the sequence. Not only does this simplify the input to transformer models, but also, as we hypothesized, it generates musical pieces with more structural quality as evidenced by the SI metric and listening study.

Even though our results suggest that some musical genres show more significant improvement than others, all of them show *some* improvements (see Table 5.7), which suggests some untapped potential on Transformer models when exploring different musical notations. Considering the time and computational costs associated with transformer architectures, any improvements that come from simple input formatting deserve study. Even when these costs are not a concern, the recent success of LLMs that train on massive amounts of tokenized data without any semantic annotations is aligned with our findings, suggesting that these annotations may not be necessary in the domain of music generation either, making massive musical datasets more attainable. As such, future works on music generation using transformers should go in the same direction as text generation: massive datasets that don't include any human annotation, and thus are cheap to obtain. In addition, other tokenization techniques could be explored. Polyphony and note duration quantization are potential targets for improvements as these are usually responsible for most of the complexity in the challenge of creating tokenized musical representations. For example, the total number of possible tokens could be reduced by substituting `NoteOn` tokens for separate pitch and octave tokens. As for the construction of massive datasets, MIDI files are plentiful on the internet, and without any need for human annotation or genre-specific musical representation, of various musical genres are plentiful on the internet

We also argue that the AI-generated musical community should decide on a stan-

dardized benchmark. Ours is one of the few, if not the only, transformer-based approach that deals with four datasets with different musical genres, helping indicate that the results can generalize to all genres of music. This issue is compounded by the fact that several recent papers measure improvements on datasets that require specific annotations, with some of these annotations being genre-specific, leaving less space for these improvements to be observed in other datasets. While studies and methods focusing on specific musical genres are valuable, it's hard to assess whether a work's improvements generalize when only one data type is presented. As for how to measure musical quality, works like [48, 50, 15, 16] offer many useful metrics that work on any MIDI dataset, and while art evaluation is always subjective, standardized benchmarks are irreplaceable tools in the scientific process when searching for better tools to tackle any problem.

References

[1] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate, 2014.

[2] Jean-Pierre Briot, Gaëtan Hadjeres, and François-David Pachet. *Deep learning techniques for music generation*, volume 1. Springer, 2020.

[3] Frederick P Brooks, AL Hopkins, Peter G Neumann, and William V Wright. An experiment in musical composition. *IRE Transactions on Electronic Computers*, (3), 1957.

[4] C.-C.J. Chen and R. Miikkulainen. Creating melodies with evolving recurrent neural networks. In *Proc. IJCNN*, 2001.

[5] Grosvenor W Cooper, Grosvenor Cooper, and Leonard B Meyer. *The rhythmic structure of music*. University of Chicago press, 1963.

[6] Joost CF De Winter and Dimitra Dodou. Five-point likert items: t test versus mann-whitney-wilcoxon. *Practical Assessment, Research & Evaluation*, 15(11):1–12, 2010.

[7] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music, 2020.

[8] Chris Donahue, Huanru Henry Mao, Yiting Li, G. Cottrell, and Julian McAuley. Lakhnes: Improving multi-instrumental music generation with cross-domain pretraining. In *Proc. ISMIR.*, 2019.

[9] Chris Donahue, Huanru Henry Mao, and Julian McAuley. The nes music database: A multi-instrumental dataset with expressive performance attributes. In *Proc. ISMIR.*, 2018.

[10] Hao-Wen Dong, Ke Chen, Shlomo Dubnov, Julian McAuley, and Taylor Berg-Kirkpatrick. Multitrack music transformer. In *Proc. ICASSP*, 2023.

[11] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment, 2017.

[12] Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. *Proc. AAAI*, 2018.

[13] Douglas Eck and Juergen Schmidhuber. A first look at music composition using lstm recurrent neural networks. Technical report, Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale, 2002.

[14] Bradley Efron. Better bootstrap confidence intervals. *Journal of the American Statistical Association*, 82(397):171–185, 1987.

[15] J. Ens and P. Pasquier. A cross-domain analytic evaluation methodology for style imitation. In *Proc. Int. Conf. Computational Creativity*, Salamanca, Spain, June 2018.

[16] J. Ens and P. Pasquier. Quantifying musical style: Ranking symbolic music based on similarity to a style. In *Proc. ISMIR*, 2019.

[17] Ali Cenk Gedik and Baris Bozkurt. Pitch-frequency histogram-based music information retrieval for turkish music. *Signal Process.*, 90:1049–1063, 2010.

[18] Gaëtan Hadjeres, François Pachet, and Frank Nielsen. DeepBach: a steerable model for Bach chorales generation. In *Proc. ICML*, 2017.

[19] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *Proc. ICLR.*, 2019.

[20] Dorien Herremans, Ching-Hua Chuan, and Elaine Chew. A functional taxonomy of music generation systems. *ACM Computing Surveys (CSUR)*, 50(5), 2017.

[21] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Curtis Hawthorne, AM Dai, MD Hoffman, and D Eck. Music transformer: Generating music with long-term structure (2018). In *Proc. ICLR.*, 2018.

[22] Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Beat-based modeling and generation of expressive pop piano compositions. In *Proc. ACM MM.*, 2020.

[23] Junyan Jiang, Gus G. Xia, Dave B. Carlton, Chris N. Anderson, and Ryan H. Miyakawa. Transformer vae: A hierarchical model for structure-aware and interpretable music representation learning. In *Proc. ICASSP*, 2020.

[24] Daniel D. Johnson. Generating polyphonic music using tied parallel networks. In *EvoMUSART*, 2017.

[25] Fred Lerdahl and Ray Jackendoff. An overview of hierarchical structure in music. *Music Perception*, 1983.

[26] Huanru Henry Mao, Taylor Shin, and G. Cottrell. Deepj: Style-specific music generation. *Proc. ICSC*, 2018.

[27] Meinard Müller. *Fundamentals of music processing: Audio, analysis, algorithms, applications*, volume 5. Springer, 2015.

[28] Meinard Müller, Peter Grosche, and Nanzhu Jiang. A segment-based fitness measure for capturing repetitive structures of music recordings. In *12th International Conference on Music Information Retrieval*, pages 615–620, Miami, FL, USA, 2011.

[29] Meinard Müller and Nanzhu Jiang. A scape plot representation for visualizing repetitive structures of music recordings. *Proceedings of the 13th International Society for Music Information Retrieval Conference, ISMIR 2012*, 01 2012.

[30] Oriol Nieto, Gautham J Mysore, Cheng-i Wang, Jordan BL Smith, Jan Schlüter, Thomas Grill, and Brian McFee. Audio-based music structure analysis: Current trends, open challenges, and applications. *Transactions of the International Society for Music Information Retrieval*, 3(1), 2020.

[31] Aaron van den Oord, Sander Dieleman, Heiga Zen, Karen Simonyan, Oriol Vinyals, Alex Graves, Nal Kalchbrenner, Andrew Senior, and Koray Kavukcuoglu. Wavenet: A generative model for raw audio, 2016.

[32] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: learning expressive musical performance. *Neural Computing and Applications*, 32, 2018.

[33] Cesar Ortiz Echeverri, Juvenal Rodriguez, and Mariano Garduño-Aparicio. An approach to stft and cwt learning through music hands-on labs. *Computer Applications in Engineering Education*, 26, 04 2018.

[34] Martin Pfleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart. Inside the jazzomat. *New Perspectives for Jazz Research*, 2017.

[35] Richard C Pinkerton. Information theory and melody. *Scientific American*, 194(2), 1956.

[36] Yang Qin, Huiming Xie, Shuxue Ding, Benying Tan, Yujie Li, Bin Zhao, and Mao Ye. Bar transformer: a hierarchical model for learning long-term structure and generating impressive pop music. *Applied Intelligence*, 2022.

[37] Alec Radford, Jeff Wu, Rewon Child, D. Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners, 2019.

[38] Seungyeon Rhyu, Hyeonseok Choi, Sarah Kim, and Kyogu Lee. Translating melody to chord: Structured and flexible harmonization of melody with transformer. *IEEE Access*, 10, 2022.

[39] Adam Roberts, Jesse Engel, Colin Raffel, Curtis Hawthorne, and Douglas Eck. A hierarchical latent vector model for learning long-term structure in music. 2018.

[40] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. In *Proc. AACL*, 2018.

[41] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations, 2018.

[42] Yi-Jen Shih, Shih-Lun Wu, Frank Zalkow, Meinard Muller, and Yi-Hsuan Yang. Theme transformer: Symbolic music generation with theme-conditioned transformer. *IEEE Transactions on Multimedia*, 2022.

[43] Peter M. Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 13(4), 1989.

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017.

[45] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Proc. NeuRIPs*, 2017.

[46] Ziyu Wang, Ke Chen, Junyan Jiang, Yiyi Zhang, Maoran Xu, Shuqi Dai, Xianbin Gu, and Gus Xia. Pop909: A pop-song dataset for music arrangement generation. In *Proc. ISMIR*, 2020.

[47] Larry Wasserman. *All of statistics: a concise course in statistical inference*, volume 26. Springer, 2004.

[48] Shih-Lun Wu and Yi-Hsuan Yang. The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures. In *Proc. ISMIR.*, 2020.

[49] Shih-Lun Wu and Yi-Hsuan Yang. The jazz transformer on the front line: Exploring the shortcomings of ai-composed music through quantitative measures, 2020.

[50] L.-C. Yang and A. Lerch. On the evaluation of generative models in music. *Neural Computing and Applications*, 2020.

[51] Hongyuan Zhu, Qi Liu, Nicholas Jing Yuan, Chuan Qin, Jiawei Li, Kun Zhang, Guang Zhou, Furu Wei, Yuanchun Xu, and Enhong Chen. Xiaoice band: A melody and arrangement generation framework for pop music. In *Proc. KDD*, 2018.