Modeling Expressive Musical Performance with Transformers: An Empirical Error Study

by

Richard W Timpson

A Senior Thesis Submitted to the Faculty of
The University of Utah
In Partial Fulfillment of the Requirements for the Degree

Bachelor of Computer Science

School of Computing
The University of Utah
April 2021

Approved:

_____/DATE	_____/DATE
Vivek Srikumar	H. James de St. Germain
Supervisor	Director of Undergraduate Studies
	School of Computing

_____/DATE
Mary Hall
Director
School of Computing

Modeling Expressive Musical Performance with Transformers: An Empirical Error Study

ABSTRACT

Current state-of-the-art modeling of expressive musical performance (EMP) is based on hierarchical Recurrent Neural Networks (RNN). The Transformer is a recent Neural Network (NN) sequence modeling architecture that has led to significant research improvement in Natural Language Processing (NLP) and other related fields. To date, there has been no application of the Transformer in music performance modeling – we present the first study that attempts to do so. The results indicate that our encoder-only Transformer model outperforms a similar encoder-only RNN but does not outperform the existing hierarchical RNN state-of-the-art. However, an analysis of current evaluation methods for EMP generation models reveals that the current metrics for "objective" quantitative evaluation do not correctly capture the essence of musical performance and are significant bottlenecks in developing robust EMP models. Therefore, we cannot draw any meaningful conclusions about our models' performance when evaluating them quantitatively. We present an error analysis of the current evaluation methods in EMP and provide suggestions for future efforts in building better models and finding better evaluation metrics.

Acknowledgments

I would first like to acknowledge Professor Srikumar for agreeing to take me as his student and advise me in my research. When I first approached him with the general idea of doing a research project involving Machine Learning and music, neither he nor I had the slightest bit of background in the research area. Although this project's scope has been relatively small, we have both learned an immense amount about music information retrieval and broadened our scope of understanding in applied Machine Learning, which had previously centered around NLP. I want to thank Vivek for his efforts in supporting me in my research interests even though he had limited domain knowledge.

I would also like to thank my wife Vivian and my daughter Sonya for understanding and supporting me through some of the struggles that I encountered while working on this project. The last six months have been a chaotic time for both my family and me, and without their continued support, I would not have been able to complete the project on time. I look forward to giving them more of my attention and time now that it is complete.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In 1952 L.A. Hiller and L.M Issacson ushered forth a new era of the study of both music and computer science when they introduced the Illiac Suite – the first musical piece composed solely by a computer [36]. What we'll refer to broadly as Music Information Retrieval (MIR) [1] research has continued to see impressive advancements since the introduction of the Illiac Suite in several different domains, including musical composition [3], instrument and sound synthesis [12], and musical analysis [40]. Musical recommendation systems are the most commonly known application of MIR. They extract common musical patterns from different genres and songs to suggest new music to listeners, which they may appreciate in the future given their past listening habits[2]. Much of the challenge in MIR research is to close the gap between music's phenomenological nature as experienced by human listeners and the hierarchical and mathematical quantitative musical patterns present in nature.

Widmer [40] suggests that there are a set of deep problems that the MIR community should focus on, which by their nature, are more fundamental to understanding the nature of music itself. One such problem is the exploration of what constitutes *expression* in musical performance. Current commercial automatic performance generation systems render deterministic and uninteresting performances that don't contain the "human" element. These "deadpan" performances do not use the different musical performance components such as variations in

[1] Widmer [40] points out that MIR itself does not encompass the entire scope of computer music research but that it is a good proxy to use when referring to the field as a whole. We will operate under the same assumption

[2] Spotify is the most well-known commercial platform that implements musical recommendation systems.

timing, dynamics, and articulation to "express" creative musical ideas or emotions[3]. Every musical performance results from an interpretation of a composition, and this interpretation is communicated through expressive performance. Our work is a further continuation of the computational modeling of expressive musical performance (EMP) in the context of automatic performance generation.

Typical EMP generation systems use Machine Learning (ML) to build computational models trained on existing data sets comprised of actual human performance. Recent ML models are either probabilistic (usually using Hidden Markov Models) in nature or based on artificial neural networks (ANN). Deep Learning (a common term used to describe ML involving ANNs)[4] has led to the widespread success of Artificial Intelligence (AI) systems in a variety of applications, including computer vision, natural language processing (NLP), speech processing, and audio processing [16]. State-of-the-art EMP models are mostly comprised of Recurrent Neural Networks (RNN) and their common adaptation as a Long Short Term Memory Network (LSTM). RNNs are designed to model sequential data (see chapter 2), such as music. A relatively new model in sequential Deep Learning, the Transformer, has led to impressive advances over RNN based models in NLP [4, 8] and other sequential data modeling problems [10]. We apply the Transformer to EMP generation, which to our knowledge is the first such application, using an existing end-to-end state-of-the-art EMP generation system.

We evaluated our model quantitatively using standard evaluation metrics, and it performed worse than the existing RNN based state-of-the-art. However, a ~~personal~~ listening test revealed a disconnect between quantitative and qualitative evaluation. We ran further exper-

[3] MuseScore, a popular open-source musical notation software, is one such commercial system. MuseScore, and other notation software programs like it, generate deadpan performances.

[4] The use of neural networks in ML is commonly referred to as "Deep Learning" because of the many connected layers that comprise the networks. The term "deep" describes the long path which information must follow to propagate through the extensive network. Typical ML models usually do not have that depth

iments deviating from standard evaluation methods to identify why this might be the case. This experimentation led to insights about potential problems with current evaluation metrics and intuition for creating better evaluation methods. Our intuition comes from an error analysis of our model, which revealed that the performance decrease might not be attributed to the underlying Transformer mechanism but our high-level network architecture.

We also bring to light some of the philosophical conundrums present when considering EMP generation from a computational level. Because the "quality" of a musical experience is highly subjective, creating the right incentives for a computer model to generate novel performances is not trivial. In agreement with other authors [40], we advocate for further research to draw just as much from music at the human psychological level as the mathematical and statistical.

Chapter 2

Background: Expressive Performance

This project is based upon two major research components. The first is the MIR problem domain of expressive musical performance, and the second is the ML modeling domain of Transformers. In this chapter, we will introduce and discuss EMP. In chapter 3 we'll do the same for the Transformer.

Expressive musical performance is a small subset of Music Information Retrieval research. We broadly categorized MIR research into two separate tasks: developing computational methods for musical analysis and developing computational methods for music generation. We are interested in generative models, although it is worthwhile to note that there is considerable overlap between the two areas[1]. Proper knowledge of the entire musical generation process as a whole is necessary to understand how EMP generation models work. Ji et al. [26] break the generation process into three stages with four different roles that interact with that process at each stage. Figure 2.1 shows each step in the process as well as the agents that participate[2].

An EMP generation model is analogous to the performer shown in 2.1, who takes as input a composition and produces as output a performance. We define musical expression as the performers' interpretation of a composition codified into different performance parameters

[1]Creating a performance generation system is useful for performance analysis as long as the generation system is interpretable. The reverse is also true. Analysis provides insight to generation, and generation provides insight to analysis

[2]This figure is our reproduction of a similar image presented by Ji et al. [26]

move to caption

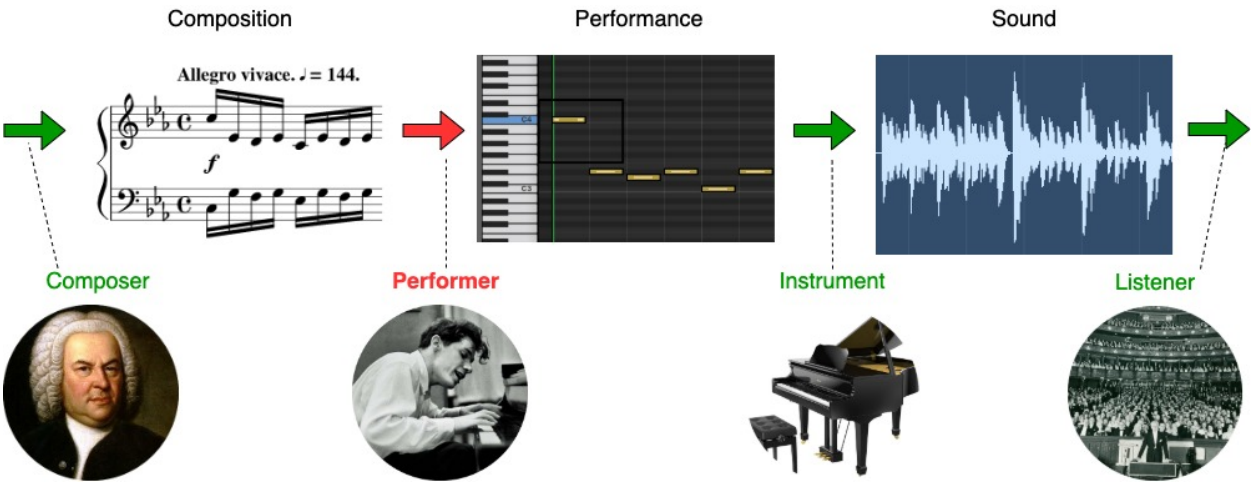

Figure 2.1: The first step of musical generation is composition, shown as a score in the figure. The second is performance, our area of interest, and is indicated by a MIDI file's piano roll representation. The third is the production of sound, shown as a raw audio wave. Each different agent: composer, performer, instrument, and listener, can be conceptualized as a separate computational model in the generation process

intended to contribute to the quality of a musical experience[3].

Figure 2.1 shows that a performer interacts with a composed score to create a performance. To more clearly define EMP generation, we start by describing the essential elements of both scores and performances. This chapter's descriptions are not intended to be at a detailed mathematical level but the general musicological level. We provide feature definitions, which contain the score and performance information embedded at a mathematical level, of the EMP models upon which our work is based in section 4.1. Due to the constraint of our data, we focus only on western classical piano music.

[3]As has already been mentioned and as will show in detail later in this work, defining the "quality" of musical experience is no minor task. Such a definition is necessary to train performers (either computer or human) to create desirable performances.

2.1 Scores

Scores are symbolic representations of a musical composition. They are comprised of markings that belong to a grammar of symbolic notation used to express musical ideas and information. Scores organize this information in a hierarchical structure with different musical details at each level. Figure 2.2 shows a sample score and the different hierarchical levels of information that it contains

The lowest level contains information about the pitch and timing of every single note, as well as optional information about how the note should be played. The low level can include information specific to instruments such as the bow direction of a violin, but for our purposes (dealing only with piano), we will consider this to be the articulation of each note. Note articulation is usually indicated by ***legato***, which suggests the notes should be played smooth and connected without breaks between the notes, or ***staccato***, which means that each note should be played sharply and detached from other notes.

The middle level contains information related to certain substructures of the musical composition, expressed within a grouping of notes or measures. The most common score annotations at this level are dynamic markings which indicate whether to play a collection of notes

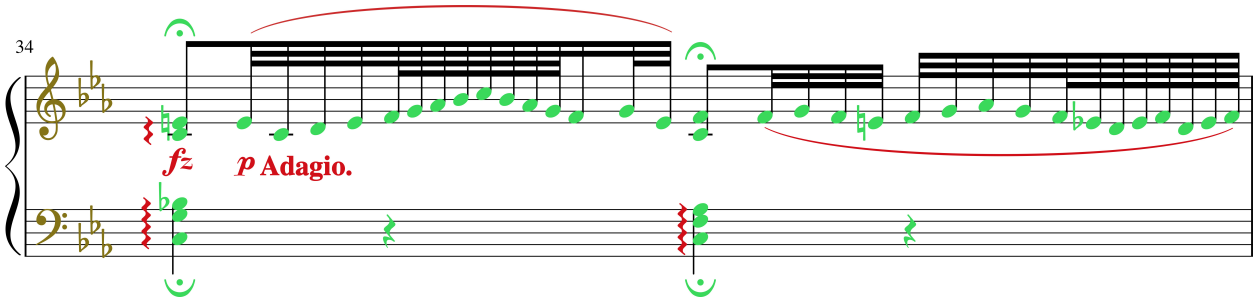

Figure 2.2: A sample hierarchy of a score delineated by different colors. The green shows the low level, which includes the pitch and timing of every note. The red represents the mid-level which shows dynamics and tempo information about local musical substructures. The tan color illustrates the high level and includes the key and time signatures.

as ***f*** (loud), ***p*** (soft), or as a ***crescendo*** or ***decrescendo*** (gradually increase or decrease the volume). Although dynamic markings are the most common at this level, it is also possible to see score markings for all other musical features, such as local tempo or articulation of a particular substructure. Perhaps the most essential score marking at this level is that of a slur, which implies that a group of notes should be interpreted as belonging to a singular musical phrase and that each note should fit within the context of the phrase as a whole. A slur can be expressed through all of the different aforementioned musical features, including the tempo, timing, dynamics, and articulation of the notes.

The highest level contains meta-information that relates to the entire composition as a whole. This information typically includes the key signature and time signature, as well as the global tempo for the entire piece, most commonly represented as BPM.

2.2 Peformance

An expressive musical performance contains most of the same musical information as a score, including pitch, tempo, timing, and articulation. The critical difference between scores and performances is that an expressive performance is a unique and measurable instantiation of a single score – no two performances will ever be the same. Expressive performances will deviate (or interpret) from the exact information given in the score.

For example, although a score may indicate a tempo of 120 BPM, it is unlikely that a given performer will perfectly adhere to this tempo throughout the piece's entirety. Some performers may choose to play faster than the indicated global tempo, while others will play slower. It's also likely that there will be natural deviations in the tempo, dependent on other musical structures, such as dynamics. Certain phrases of a composition may "feel" slower or faster than the indicated global tempo, and the deviation in tempo is an essential

component of a performer's interpretation. Deadpan performances, which are hardcoded and deterministic, will strictly comply with the notated global tempo unless the score explicitly indicates otherwise[4]. Each expressive parameter will be measurable and absolute, whereas the score markings of these features are more of a suggestion than a rule. Figure 2.3 shows an example of two performances of the same score and the difference in their measurable expressive attributes.

Performance has a few additional components that are not necessarily indicated in scores but are relevant in the context of performance alone. Piano pedaling is one crucial component. There are several different types of piano pedals. The most common are the sustain pedal, which prolongs every notes' duration when activated, and the soft pedal, which softens the entire piano's sound. Although these pedals' effects are directly related to the articulation and dynamics of the performance, their presence (or lack of) is a crucial component of piano performance. The sustain pedal is actively used in almost all modern piano performance, even in the absence of pedal score markings.

2.3 Data

The data required for EMP generation includes a digital representation of a score and a corresponding performance. MusicXML, a text-based representation of a composition, is the most common data format for a score. Performances could directly be represented through raw audio, which is widely available on the internet. Instead of audio, however, an

[4]Compare two performances of Bach's Prelude in C Minor BWV 847, one by Glenn Gould and the other by Vikingur Olafsson. Gould's interpretation is much slower than Olaffson's (Gould was known for his unique musical interpretations, which often went against the grain of the musical establishment of the time). This significant difference in tempo leads to a drastically different musical experience. We can also compare these human performances with MuseScore's deadpan performance (make sure to choose the "MuseScore audio" audio source, which can be found by selecting the icon of audio switches next to the "Off" text in the top menu bar), which illustrates the necessity of expression in creating good performance

Do we have BWV 847 generated by LSTM, Transformer & the HAN?

intermediate data form, MIDI, is commonly used to constitute the performance. MIDI is an event-based data format and software interface that contains information about different aspects of a recorded performance. MIDI defines NOTE-ON and NOTE-OFF events, and each event specifies the pitch and velocity of the note. Velocity represents the 'speed' at which a note was hit and is a good proxy for dynamics. Both pitch and velocity are represented as discrete numbers that exist in the 0-127 range. MIDI also defines control channels which are streams of data separate from the NOTE-ON and NOTE-OFF events. The control channels contain the information for the values of different pedals throughout a performance. Figure 2.3 shows a visualization of a MIDI performance.

There are several reasons for the preference of MIDI over raw audio. The first is that audio data is inherently noisy, sparse, and vast in its dimensionality and size. Using audio to represent performance requires a significant compression of natural frequencies into more abstract musical representations such as pitch – MIDI natively contains these abstractions. MIDI also better aligns with the generation process outlined in 2.1 in which performance is seen separate from sound synthesis and allows computational models to define such boundaries easily. In the full computational generation process, a different model would be used to take the performance data in MIDI and synthesize it into raw audio presented to the listener.

Computational models that deal with the relationship between scores and performances often need additional metadata about the alignment between every note in both the score and performance. This metadata is generated by running a performance and its corresponding score through a data alignment process in which every performance note is matched to a marking in the score. Given the highly dynamic nature of musical performance, it is a non-trivial task to run this alignment process for a set of scores and performances, especially if the task is performed by manual human annotation. There exist methods for both manual

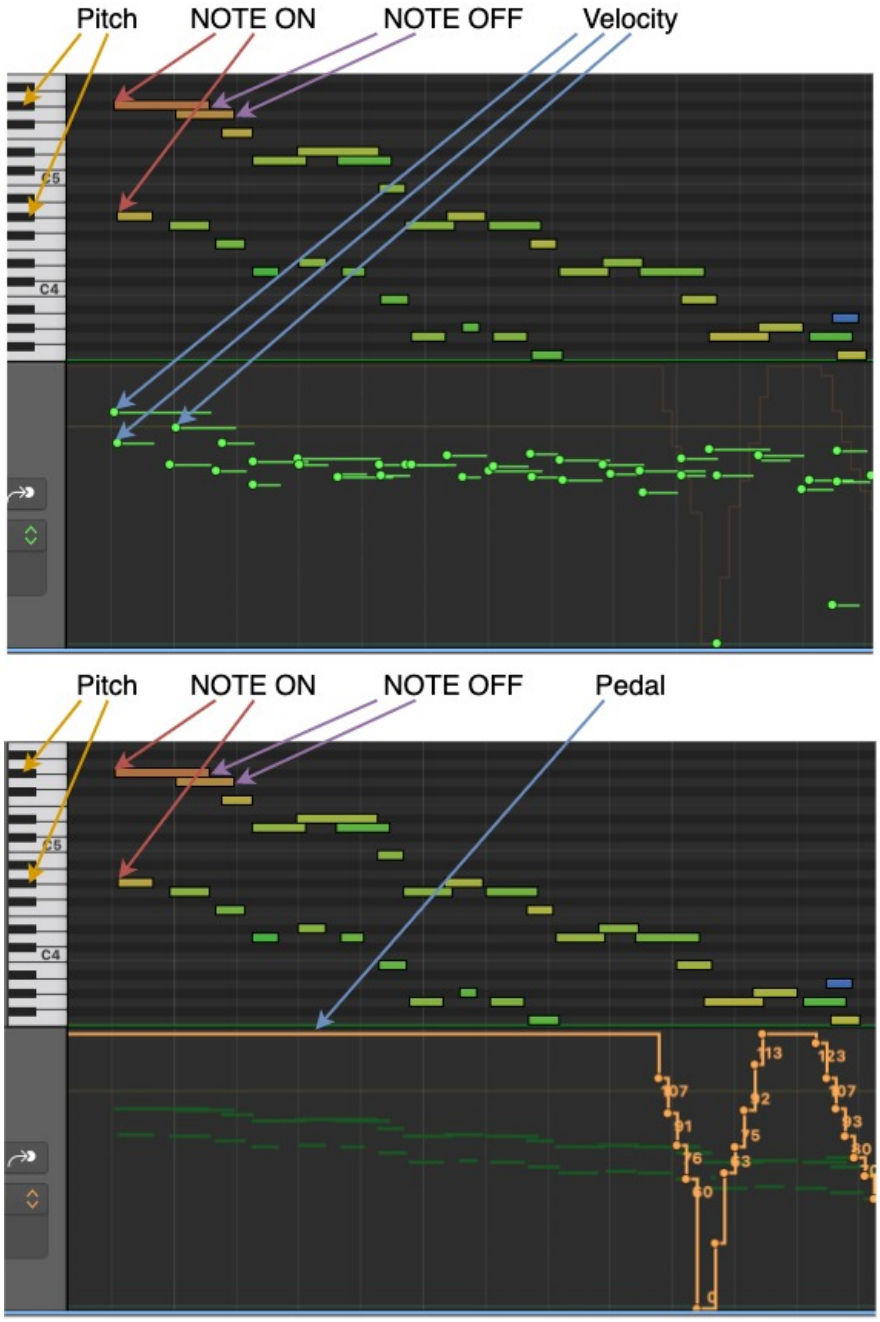

Figure 2.3: A visualization of a MIDI performance taken as a screenshot from Logic Pro X. Each NOTE-ON and NOTE-OFF event holds the information about the pitch and velocity. A separate control channel encodes the value of the sustain pedal

and automatic alignment. Due to the time-consuming nature of manual alignment and the need for large data sets to build higher quality models, automatic alignment algorithms are an active research area. An example alignment is shown in figure 2.4

2.3.1 Existing Data Sets

One of the problems facing EMP and MIR is the lack of high-quality and large-scale datasets [5]. The biggest hindrance in data gathering is that most of the musical data available on the internet is in the form of raw audio. As we have pointed out, processing audio data for musically plausible purposes is difficult compared to other better-suited formats. While a fair amount of MusicXML and MIDI does exist, it is relatively small compared to the amount of audio data.

We draw a comparison to the data used in NLP research, where textual data is analogous to MusicXML and MIDI, and speech audio is analogous to musical audio. One of the reasons NLP has been able to advance its progress in the last few decades is the massive amount of readily available textual data on the internet. If most available text data had to be extracted from audio, it would be significantly more challenging to develop NLP systems and applications. Such is the case with MIR research data. Nevertheless, there has been a recent effort to gather large-scale quality data in MIR. We present some of these datasets as they are relevant to EMP.

There are normally three required components for a EMP dataset.

1. Scores (usually in the form of MusicXML)
2. Performances (usually in the form of MIDI)
3. Metadata about the matching alignment between the score and performance.

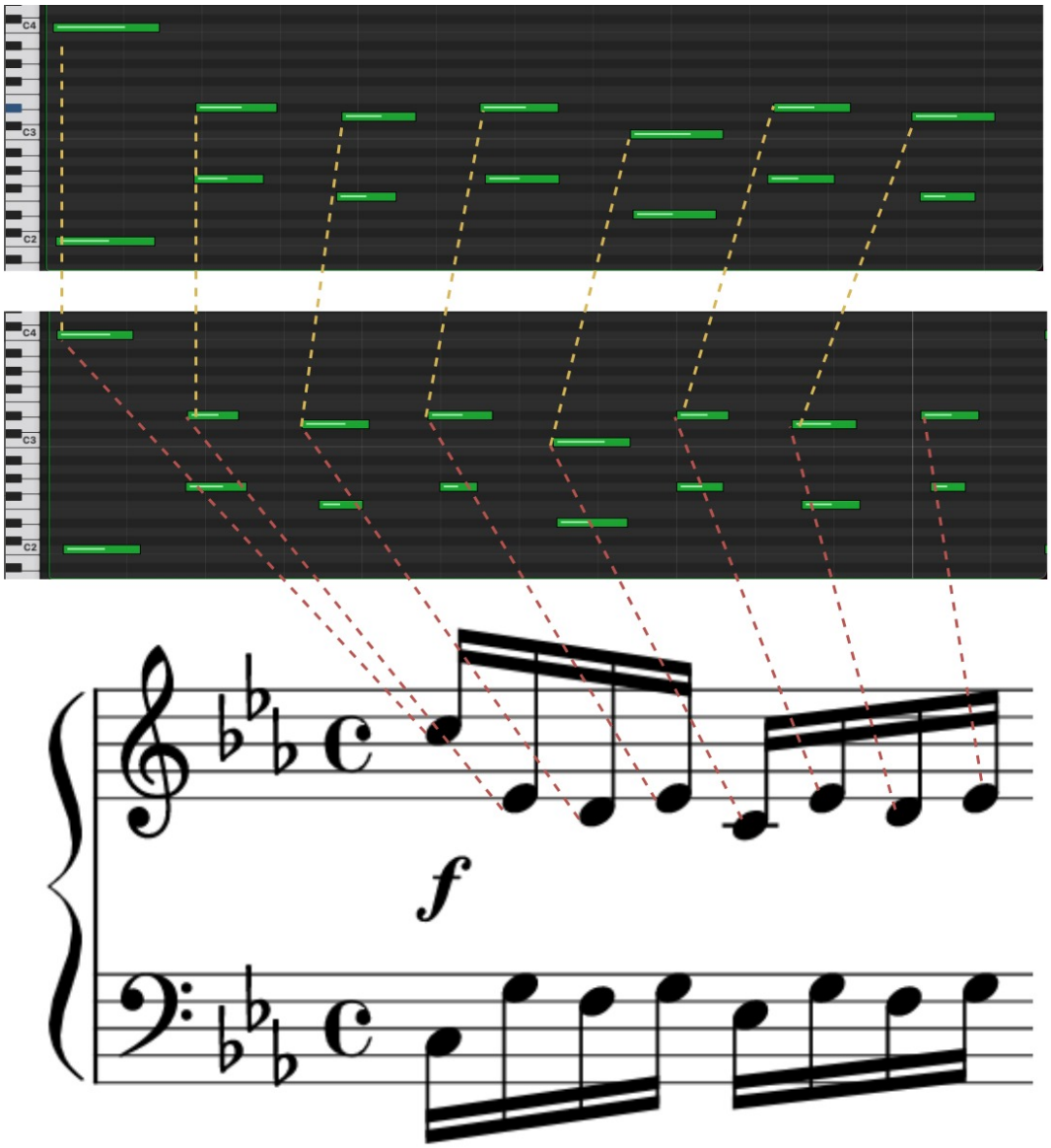

Figure 2.4: The top two images are piano roll representations of the first measure of Bach's Prelude in C Minor BWV 847, recorded in MIDI ~~by the author.~~ Also shown is a score of the same composition. The first performance is aligned to the second performance. The second performance is aligned to the score. Notice the marked difference in tempo between the two performances. These example performances illustrate why score to performance alignment is necessary. Different performances of the same score can significantly vary in their note onset, note duration, and note velocity values. As such, it is necessary to create an alignment between every performance and its corresponding score.

unless otherwise stated, this is assumed

Score data is gathered by finding readily available MusicXML files from open source software projects which contain music that is in the public domain (which all western classical music is) [5], or by using Optical Music Recognition (OMR) to automatically scan paper sheet music into a digital form followed by manual corrections where needed. MIDI performances can only be recorded when a performer uses an instrument compatible with the MIDI protocol. Although there are many digital piano keyboards with this combability, semi-professional and professional pianists primarily play on acoustic pianos. Some computer-controlled acoustic pianos can record MIDI performance, such as the Yamaha Disklavier and the older Bosendorfer CEUS system. Therefore, MIDI data of professional piano performance has historically been limited to performances on these piano systems.

There is no standardized method for score-to-performance alignment methods and data representations. Each dataset presents its own alignment method as well as the metadata that represents the alignment.

To provide context for the progression of data used in EMP generation, we will introduce an older dataset used in EMP research, the Magaloff Corpus. We then describe a much larger scale dataset, the Piano-e-competition, which has recently been adapted for EMP generation use. Cancino-Chacón et al. [5] give a full overview of datasets used in EMP generation.

Magaloff Corpus

Nikita Magaloff was a Russian pianist known for his performance cycles of Chopin's entire works for the solo piano. In one of his final cycles of performances recorded in 1989, he played on a Bosendorfer SE computer-controlled piano. Flossmann et al. [13] obtained special permission to access the MIDI data from this performance cycle. They gathered score data using OMR with manual corrections where needed. The alignment method of Grachten et al.

[5] MuseScore is the most common. The International Music Score Library Project is also commonly used ← missing period.

[18] was used to produce the note-matching annotations, along with manual corrections. The dataset contains over 10 hours of playing, 150 compositions, and over 320,000 performed notes. The corpus, however, is not publicly available and has only been used in research by Flossmann et al. [13] and colleagues [11].

Piano-e-competition

As has been discussed, there is a large push in modern MIR to produce high-quality large datasets. At the heart of this research in MIR is the Piano-e-competition. Started in 2002, it is an international piano competition that attracts some of the promising up and coming musicians at both the senior and junior level [1]. Every performance from the competition is played on a Yamaha Disklavier and is recorded in both MIDI and audio. Due to the data set's size and availability, it is commonly used in MIR research. There exist several different adaptations of the original data that are specific to particular research purposes.

The first of these is the MAESTRO (Midi and Audio Edited for Synchronous TRacks and Organization) dataset. Hawthorne et al. [20] introduce the MAESTRO dataset, which presents both MIDI and audio data from the Piano-e-competition in a canonical and easily accessible form. The dataset was first used to build a complete musical analysis and generation process framework named wav2midi2wave. This framework includes a musical transcription process [19] from raw audio to midi (wav2midi), a direct musical composition and performance generation model (can be seen as the midi or midi2midi part of the wav2midi2wav framework) [21][6], and a synthesis model that takes MIDI and generates raw audio [30] (midi2wav). The MAESTRO dataset is the most commonly used form of the Piano-e-competition data.

[6]This model directly generates MIDI files without using scores. It simultaneously generates a composition and performance. This direct generation is a merging of the two separate tasks into one as shown in figure 2.1 ← missing period

The Piano-e-competition also forms the basis for a data collection, which we will refer to as the KAIST dataset[7]. The Piano-e-competition dataset itself does not provide any score data about any of the compositions used in performance. The KAIST dataset was created specifically for an EMP generation system, and therefore needs score data for every performance recorded in MIDI. This score data was collected by Jeong et al. [23] by downloading MusicXML files online, mostly from MuseScore. On top of gathering the score data for all performances in the Piano-e-competition, Jeong et al. [23] run the automatic score-to-performance alignment algorithm of Nakamura et al. [28] to provide metadata about the alignment between each score and performance. Automatic score-to-performance alignment is error-prone, especially in the case of performance mistakes[8]. As a result, some performance notes are not aligned to those in a score. Due to the possibility for error in automatic alignment, Jeong et al. [23] also add additional manual and heuristic corrections to the alignment where needed.

Other than the discrepancy in scale, the most significant difference between the KAIST dataset and the Magaloff corpus is that the KAIST dataset contains multiple performances for the same score. In contrast, the Magaloff Corpus has a 1-1 mapping between a score and performance. The KAIST dataset has 226 scores across 16 different composers, roughly 660,000 score notes, and around 3,500,000 performance notes. The number of matched performance notes is ten times larger than the Magaloff Corpus, and all data is publicly available[9].

The Aligned Scores and Performances (ASAP) dataset [14] is a recent adaptation of both the KAIST dataset and the MAESTRO dataset. It uses the MusicXML files from the KAIST dataset, audio from the MAESTRO dataset, and MIDI files from both sources extracted from

[7]We take the name from the KAIST Graduate School of Technology, which the researchers who created the dataset work for.

[8]A mistake in the performance results in a performance note having no correct match with a score note.

[9]The dataset is open sourced at https://github.com/mac-marg-pianist/chopin_cleaned

the common origin of the Piano-e-competition. It provides additional alignment metadata for both MIDI and audio and more manual correction in the MusicXML score files. Although the purpose of the ASAP dataset is for Automatic Music Transcription (AMT)[10] it is just as equally beneficial for EMP generation. To our knowledge, it hasn't seen an application in any EMP generation task. Although it is mainly similar to the KAIST dataset, the implications of its extensions are undetermined in EMP research.

2.4 Performance Evaluation

Evaluation is an essential component of developing ML models. A models' evaluation is a measure of its quality and serves as a benchmark that can be compared to other models in the same domain. Due to the inherently subjective nature of music and musical performance discussed in chapter 1, evaluation is notoriously difficult to understand and define correctly for EMP generation models [5].

The evaluation of computational models, specifically for EMP models, is typically categorized in two ways: quantitative and qualitative evaluation. Quantitative evaluation methods produce numerical metrics, which are computationally generated and deterministic, to define and measure a model's quality. Such evaluations are considered to be an "objective" measure of quality, given that they don't have any reliance on human judgment. Qualitative evaluation methods involve human feedback and assessment, often presented in some standardized statistical measures. Qualitative methods are not as consistent and more challenging to reproduce, given the reliance on human listeners' subjective feedback. Therefore, quantitative evaluation methods are traditionally preferred over qualitative because of their consistency and reliability. However, in music generation models, qualitative evaluation methods may

[10] AMT is the task of transcribing a score from a performance (either in audio or MIDI form). AMT is the "opposite" of EMP. It maps a performance to a score instead of a score to performance

be more relevant because of music's highly subjective nature. Finding suitable evaluation methods is an active area of research in EMP [5], and as we will see in later chapters, need significant improvement.

Quantitative Evaluation

Several different metrics are commonly used in the evaluation process, all of which are specific to the data type and model problem domain. Our model, which we cover in chapter 5, is a regression model. We present some standard metrics used to evaluate regressive EMP models.

The two common metrics used for evaluation and regression are Mean-Squared-Error (MSE) and the Coefficient of Determination (commonly abbreviated as R^2). MSE is used to measure the difference between a prediction and an actual observed target value and is denoted as

$$MSE = \frac{1}{n}\sum_{i=1}^{n}(Y_i - \hat{Y}_i)^2,$$

where Y_i is the observed value at time step i, and $\hat{Y}_i$ is the predicted value. R^2 is a probabilistic measure of the linear correlation between variables X and Y, and is denoted as

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$$

where cov indicates covariance and σ indicates standard deviation.

Qualitative

Qualitative evaluation in EMP is most often conducted by presenting computationally generated performance to human listeners and gathering feedback and judgment. Although there

have been efforts to standardize this evaluation process, the qualitative evaluation methods of different EMP models significantly vary. The lack of consistency in the evaluative methods makes reproducing experiment results difficult in EMP research. However, it is still commonly used to provide context to a model's performance that is not present from quantitative metrics.

Although the competition no longer runs, the Performance Rendering Contents (Rencon) [27] has been the most significant research effort in developing EMP generation qualitative evaluation. Starting in 2002 and ending sometime around 2013 [5], Rencon was an annually held competition that provided a way to compare EMP systems using standardized evaluation methods. Similarly to human performers' performance competition, it subjected each contestant models' performances to the same listening judges and judgment rubric. Subjecting all models to the same human judges and rubrics adds a higher level of confidence in the resulting evaluation by reducing the potential bias present in separate evaluations (often by the same institution that built the model) conducted during each model's development.

Although the Rencon evaluation methods changed from year to year, they all followed the same general structure. To enter the competition, contestants would submit MIDI files of generated performances by their respective models. The choice for compositions was usually restricted to a small set of composers, such as Chopin or Mozart. A Yamaha Disklavier was then used to create an acoustic rendering of the generated MIDI file played in front of a set of judges. The judges' makeup usually included musical experts with varying experience. The judges submitted a feedback survey that would rate aspects of performance such as "rhythmic accuracy," "musical skill," and "overall quality," along some standard statistical scale. A winner was chosen according to the judge's survey entries.

Although Rencon does not currently hold competitions, it forms the basis for many of the methods used in the qualitative evaluation of EMP models. The general method of presenting

generated performances to a set of human judges to assess according to a standard rubric is the same as that of the Rencon. However, the exact survey methodology and the judges' makeup varies across each experiment [23, 37], which introduces problems of reproducibility and bias.

There are some methods of qualitative evaluation which don't follow the listening test method of Rencon. They usually involve a mixture of both computational and human analysis. Such methods may use computation to generate visualizations or find patterns in performance data, which a researcher then interprets in a qualitative way [17, 25, 41]. There are also more informal listening-based evaluation methods, in which the feedback given by the listening judges exists not as a statistical measure but as general feedback and reaction [31]

Chapter 3

Background: Sequence Modeling

In the following chapter, we provide the background and context behind the choice of our model. Recurrent Neural Networks currently provide the foundation for state-of-the-art EMP models. Recent developments in neural sequence modeling move entirely away from RNNs and toward a new family of ANN architectures, the Transformer. ~~Because~~ while Transformers have not seen application in EMP generation models, they are the focus of our work.

3.1 Sequential Data

The modeling of sequence data is a fundamental aspect of modern machine learning. Sequence data consists of individual data points with a relationship to each other according to some specific order and position in time. A simple example of sequential data is the weather, which follows predictable patterns according to the time of year. Although it is difficult to predict precise weather patterns using the year alone, we can easily forecast general weather patterns (temperatures will always be warmer in the summer than in the winter).

Musical data is also fundamentally sequential[40] given that we experience music as events that happen in time, and the relationship of such events according to their position in time is paramount to the phenomena of musical experience. Language and speech exhibit this same property, and the research in natural language processing drives much of the research advance in neural sequential data modeling.

Sequence data modeling is typically categorized into several different tasks. Perhaps the most common task is sequence classification, which seeks to assign some sequence of data points to a particular class of data. When using real-valued features, input data is defined as a sequence of data points $X = \{x_1, x_2, x_3, ..., x_n\}$, where each data point is an m-dimensional vector $x_i \in \mathbb{R}^m$. The output data is a single value $y \in L$ where L is a set of class labels. A sequence classification model $C : \mathbb{R}^{n \times m} \rightarrow L$ will then map from an input sequence X to a class label y. Email spam detection and genre classification are common use cases of sequence classification models in NLP and MIR, respectively.

EMP generation is a more complicated process. It involves mapping an input sequence (score) to another output sequence (performance). We call such a task a sequence-to-sequence (*seq2seq*) model. In this case our input data X is the same, but our output data is also defined as a sequence of vectors $Y = \{y_1, y_2, y_3, ..., y_{\hat{n}}\}$, where $y_i \in \mathbb{R}^{\hat{m}}$. We can then define a *seq2seq* model as $S : \mathbb{R}^{n \times m} \rightarrow \mathbb{R}^{\hat{n} \times \hat{m}}$ which will produce an output sequence Y given the input sequence X. In EMP generation, m is the number of score features, n is the number of input score notes, $\hat{m}$ is the number of performance features, and $\hat{n}$ the number of output performance notes.

3.2 Case Study: Neural Machine Translation

RNNs and their common adaptations as an LSTM have historically been the default modeling choices for sequential data Deep Learning[1]. However, in recent years the Transformer [38] architecture model has outperformed RNNs in many tasks and is becoming the de-facto standard for sequential data modeling in modern Machine Learning [4, 8]. To provide context for the Transformer's origin, we will outline the historical progress of an NLP task known as

[1]For brevities sake, we do not provide the detailed mathematical definition for RNNs here and refer to the reader to Goodfellow et al. [16]

neural machine translation (NMT).

Machine translation (MT) is the task of computationally translating one natural language to another[2]. Traditional MT systems relied on complicated rule sets and decoding algorithms stitched together to create a statistical model known as a statistical machine translation (SMT) system. Rather than an amalgamation of several different systems developed independently, NMT systems are trained end-to-end with a single ANN architecture and significantly reduce the complexity of building MT models. NMTs are *seq2seq* models and typically operate by translating a single sentence at a time.

One of the challenges in building an MT systems is that the source and target sentences are often not the same lengths - $n \neq \hat{n}$ in our definition of *seq2seq* models given in section 3.1. NMT translation systems use what is known as an *encoder-decoder* architecture to address the variable-length input and output sequences. From a probabilistic perspective, the job of the *encoder-decoder* architecture is to model the conditional probability distribution of a variable-length output sentence Y given a variable-length input sequence X, as

$$P(y_1, y_2, ...y_n | x_1, x_2, ..., x_l).$$

In the *encoder-decoder* architecture, calculation of the distribution is decomposed into two separate models. The encoder's job is to read in the source sequence and find a good representation, or encoding, of that sequence. In the original formulation of the *encoder-decoder* NMT architecture, Cho et al. [7] present this encoding in the form of a fixed size vector c. The decoder is an autoregressive language model[3], and uses c as a condition to

[2] Google Translate is one successful commercial application.

[3] Autoregressive models are sequence-based models that take as input the model's output at previous time steps. Language models are instances of autoregressive models that are capable of generating novel texts of varying lengths. Language models can generate novel text from scratch, although they are often prompted with existing text to write in a particular style or on a certain subject.

generate the new sentence in the target language. Using this decomposition, we can view the decoder as calculating the probability of the next word in the sentence given all of the previous words and the hidden encoding vector [2]. This probability distribution is given as

$$P(Y) = \prod_{\hat{n}=1}^{n} P(y_i|y_1, y_2, ..., y_{i-1}, c)$$

The RNN-based *encoder-decoder* model of Cho et al. [7] achieved state-of-the-art in MT, improving upon the results of SMT systems. However, there is an inherent limit imposed on the system for long sentences. Cho et al. [7] show that the performance of a basic *encoder-decoder* model deteriorates rapidly as the length of an input sentence increases. To account for this, Bahdanau et al. [2] present the foundations for what has become known as the *attention* mechanism. Instead of using a fixed-sized vector encoding, *attention* allows the decoder model to search for a set of positions in the source sentence where the most relevant information is concentrated and uses this information as it generates text in the target language. In simpler terms, the decoder "pays attention" to the source sentence's most relevant words to find the right translation of every word at each time step.

Rather than use a fixed-length vector at every time step, the decoder is modeled as follows

$$P(y_t|y_1, y_2, ..., y_{t-1}, X) = g(y_{i-1}, s_i, c_i)$$

where g is some non-linear potentially multi-layered function that outputs the probability of y_t and $s_{\hat{t}}$ is the hidden state of the RNN at time step t. c_i is a context vector using information about the relationship between certain words in the source sentence and the next word y_t to be generated. The context vector encodes the attention weights applied at every time step i (see Bahdanau et al. [2] for the full description behind this context vector). Using a context vector at every time step, the model is not restricted to a single fix-sized

vector that encodes the source sentence. This attention-based model achieved state of the art results in NMT over the purely RRN based *encoder-decoder* model.

Since the introduction of the attention mechanism, it has been used in tandem with RNNs and other DL models to push state of the art in various sequence-based tasks, such as Question Answering, Sentiment Analysis, and Part-of-Speech tagging [6]. There are several reasons for the increase in performance. One reason is that attention can better model a sequence's dependencies without respect to the distance between any two elements (this is a known limitation of RNNs) [38]. Until the introduction of the Transformer architecture, almost all applications of *attention* were used in conjunction with RNN-based modeling architectures.

3.3 Transformers

The Transformer model is an attention-only neural network architecture designed for sequence data modeling. The original Transformer by Vaswani et al. [38] was built for NMT as an *encoder-decoder* model. This Transformer model significantly advanced the state-of-the-art in NMT and has since been applied to many other sequence modeling tasks, both inside and outside NLP.

3.3.1 Attention is All You Need

Both the encoder and decoder of the Transformer architecture consist of a stack of N layers. Each layer combines the attention mechanism with a standard pointwise fully connected feed-forward neural network (FFNN). The encoder layer uses *self-attention* – attention applied in a single sequence rather than attention between an input and output sequence. In *self-attention*,

each element in the sequence "pays attention" to other elements in the same sequence. The decoder uses *self-attention* as the basis for the language model and applies the regular attention mechanism to the outputs from the encoder layers. The Transformer architecture is shown in figure 3.1. This model is conceptually similar to the RNN-based attention model of Bahdanau et al. [2] which uses the attention mechanism to model the relationship between the encoder and decoder but uses the self-attention mechanism instead of an RNN hidden state to form the modeling basis for both the encoder and decoder.

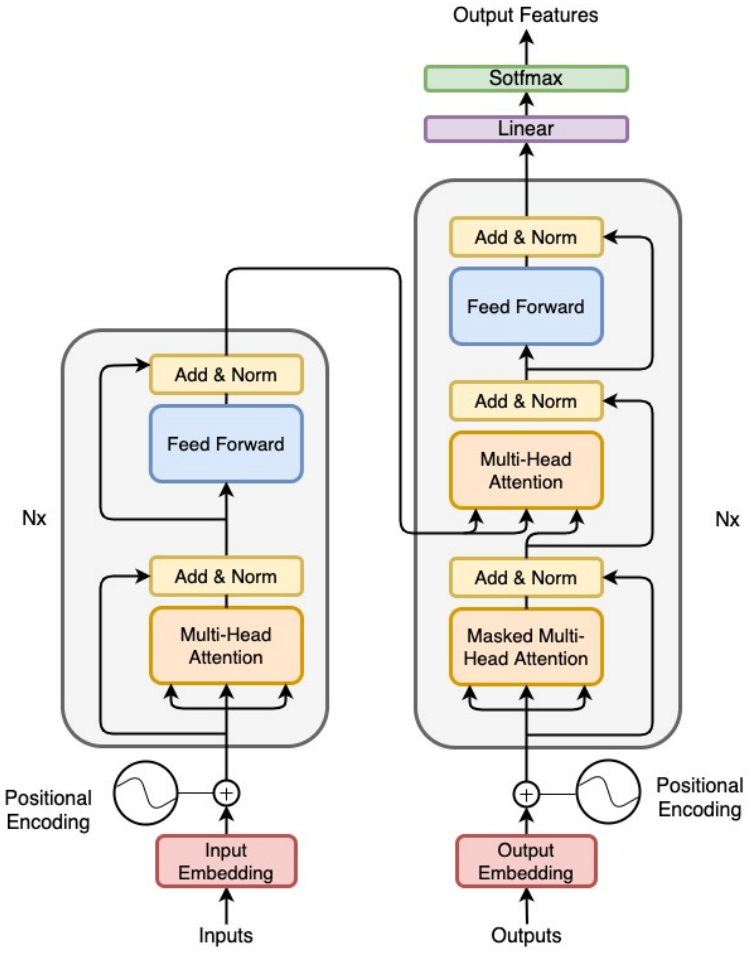

Figure 3.1: This is a reproduction of the graphic presented by [38]. The encoder uses self-attention to produce encoding outputs which are fed to the decoder. The decoder uses these encoding outputs and the outputs from its autoregressive self-attention layer to generate the final output text. The graphic shows a single Transformer layer. An actual Transformer model will stack N layers before calculating the output probabilities.

As discussed, the attention mechanism can better model longer-term memory than an RNN hidden state. Attention also allows for better parallelization in training than does the RNN hidden state. Better parallelization implies faster training and more efficient use of compute resources, which creates the capability for bigger models and larger datasets. Due to these reasons, among others, the Transformer has achieved impressive state-of-the-art results over RNN-based models.

3.3.2 Transformer Adaptations: BERT and GPT

The original Transformer was built with machine translation in mind, but several other NLP tasks could benefit from using an attention-only architecture. Some of these tasks include standard text classification, textual entailment, sentiment analysis, and question answering. A recent trend in NLP research is the pre-training of large models on a single task, which creates generic language data representations fed into models trained for a specific task [34]. BERT is such a model but uses a Transformer architecture for pre-training as opposed to an RNN. BERT is an encoder-only Transformer that significantly increases the original model's parameters and size and is trained on a massive dataset [8]. The pre-trained representations from BERT are then used with much simpler models trained on specific tasks. This approach led to a significant improvement in the state-of-the-art for general language understanding benchmarks.

The Generative Pre-Trained Transformer (GPT) is similar to BERT, utilizing only the decoder from the original Transformer instead of the encoder [35][4]. Like BERT, it is trained on massive amounts of data, is significantly large than the original Transformer, and is used

[4]The model we cite here is the second version of the GPT architecture and is canonically known as GPT-2. A third iteration of GPT, GPT-3, has since been introduced [4]. All GPT models use the same model architecture but increase the size of the model and scale of training data at each successive version. We use GPT to refer to all models which use this architecture.

to pre-train a generic language representation that is fed into other models trained for specific tasks. Because GPT is a large Transformer decoder-only language model trained on an immense data corpus, it is capable of writing novel text of surprising quality[5].

The original Transformer, BERT, and GPT have all significantly improved state-of-the-art for many different tasks NLP. How effective attention-only Transformer models are in other types of sequence data modeling is still an open research question. As we will see in chapter 4, there are already promising results inside the music domain, and even in other fields such as image processing [10].

[5]Samples of text written by GPT can be found online

Chapter 4

Music Generation Models

As in other fields, state-of-the-art models in music generation are mostly comprised of ANN models. Some music generation tasks have already seen a Transformer application, while others (such as expressive performance generation) have not. We will provide an overview of the existing state-of-the-art in EMP generation and transformer-based music generation models. We also give some background into the data representation and feature engineering of each model.

4.1 Existing Expressive Musical Performance Generation Models

EMP generation models fit into one of two categories, rule-based and data-based. Rule-based systems use hardcoded rules derived using pre-existing musical knowledge and empirical studies involving human cognition. Data-driven models rely on ML methods and use an existing dataset of human performances as a guide to learn the mapping between score features and performance features. Data-based models most commonly use sequential probabilistic or non-linear ANN methods [5], although there has been previous work with linear and non-sequential modeling. Cancino-Chacón et al. [5] give a complete overview of all relevant EMP generation models. We describe a few models, both rule and data-based, which are pertinent

to our work.

4.1.1 Rule-Based EMP Models

The KTH system [15] sits at the center of rule-based EMP models and created the foundation for all EMP generation models. Development of the KTH system started in the 1980s and has continued well into the 21st century. The KTH system's initial methodology defined rules relating to musical composition structure and how it affects a resulting performance. The first set of rules applied specifically to singing synthesis and were later adapted to general musical performance.

Since then, there have been two general methods in the continued development of the KTH rule system. The first is that of *analysis-by-synthesis*, which involved using the rules to synthesize musical performances presented to human listeners (both professional and non-professional), gathering listening feedback, and then using this feedback to modify the rules where needed. The second was an *analysis-by-measurement* method. This method uses direct computation to evaluate a computational generated performance by comparing it with an existing real performance[1]. The *analysis-by-synthesis* and *analysis-by-measurement* correspond to qualitative and quantitative evaluation methods, respectively. Example rules from the KTH system are found in figure 4.1.

To our knowledge, the KTH rule-based system is the first sophisticated computational model for generating expressive performance. The explicitly defined rules in the KTH system may be those we expect a data-based model to learn. Widmer [39] shows that data-driven methods learn some rules consistent with the KTH rules and some that aren't. Model

[1]This evaluation method is consistent with the data-driven approaches but can generally apply to any model. Data-driven models use the performance data to directly build the model, whereas real performance data in the KTH system is for evaluation purposes only. Any further updates to the model still rely on a hardcoded set of rules

Phrasing	
Phrase Arch	Create arch-like tempo and sound level changes over phrases
Final ritardondo	Apply a ritardando in the end of the piece
High Loud	Increase sound level in proportion to height
Micro-level timing	
Duration contrast	Shorten relatively short notes and lengthen relatively long notes
Faster uphill	Increase tempo in rising pitch sequences
Articulation	
Punctuation	Find short melodic fragments and mark them with a final micropause
Score legato/staccato	Articulate legato/staccato when marked in the score
Repitition articulation	Add articulation for repeated notes
Overall articulation	Add articluation for all notes except very short ones

Table 4.1: Some of the KTH rules given by Friberg et al. [15].

evaluation's problematic nature may describe this phenomenon, as it is not straightforward how to determine which rules are more "correct" than others. Nevertheless, the KTH rule system has been an essential milestone in the evolution of EMP generation.

4.1.2 Data-Based EMP Models: Basis Function Models

The Basis Function Modeling (BM) framework for EMP describes the complete end-to-end process involving musical performance generation and analysis. This end-to-end process includes mathematical definitions of score and performance features, computational models for EMP, and the data processing necessary to convert MusicxML to input score features and output performance features to MIDI. Eduardo [11] outlines the full mathematical definition of the BM framework, the evolution of the framework, and its application with specific feature and model definitions.

Features

The name "Basis Model" is derived from the BM framework's definition of "Basis Functions". Basis Functions capture different aspects of a score into numerical encodings (i.e. score features). There are some relatively straightforward Basis Functions, such as the duration of notes and MIDI pitch (we described these features as the "low level" of the score hierarchy in section 2.1). However, most of the information in a score cannot be represented so easily. It becomes a theory-laden activity to use musical background knowledge to find good score information encodings, especially at the mid and high levels.

The BM framework defines Ω as a space containing aspects of a score. A score basis function $\varphi : \Omega \to \mathbb{R}$ will map elements of a score into some numerical expression based on an aspect of the score. The value of a score basis function for one element $x_i \in \Omega$ is denoted as $\varphi(x_i)$. For example, the score basis function for pitch,

$$\varphi_{pitch}(n_i) = \text{MIDI}_{\text{pitch}}(n_i)$$

will represent the pitch of a given note as the numerical MIDI pitch. The basis function for dynamic markings, such as forte, would be defined as

$$\varphi_{\boldsymbol{f}}(n_i) = forte(n_i)$$

where $forte(n_i)$ returns 1 if the score element is marked with $\boldsymbol{f}$, and 0 if it doesn't.

Similarly to basis functions, the BM framework defines *expressive parameters* which are mathematical encodings of information taken from a performance matched to a score. An *expressive encoding function* $y : \beta_\Omega \to \mathbb{R}$ will encode an event in a matched performance into an *expressive parameter*, or performance feature. The value of each parameter corresponding to a score

element $x_i \in \Omega$ given a matched performance β_Ω is denoted as $y(x_i|\beta_\Omega) = t_i$ where $t_i \in \mathbb{R}$. Example encoding functions are

$$y_{\text{dyn}}(n_i|\beta_\Omega) = \text{MIDI}_{\text{vel}}(n_i) \text{ and } y_{\text{tempo}}(n_i|\beta_\Omega) = \text{bpm}(n_i)$$

which represent expressive dynamics as MIDI velocity and expressive tempo as a local measure of bpm, respectively.

There are many possible definitions for both basis and expressive encoding functions. Here we give some example functions used in the first iteration of the BM framework – refer to Eduardo [11] for a full list.

Basis Functions

- **Polynomial Pitch Model**: This model is presented by Grachten and Widmer [17] and describes the dependency of expressive dynamics (MIDI velocity) on pitch. This model defines three different pitch functions.

$$\varphi_{\text{pitch}}(n_i) = \frac{\text{pitch}(n_i)}{127} \quad \varphi^2_{\text{pitch}}(n_i) = \left(\frac{\text{pitch}(n_i)}{127}\right)^2 \quad \varphi^3_{\text{pitch}}(n_i) = \left(\frac{\text{pitch}(n_i)}{127}\right)^3$$

- **Dynamics Markings**: These are functions which encode dynamics markings such as ***p***, ***f***, and ***crescendo***. There are three different types of dynamics encoding functions. The first is an impulse function which is a binary indicator of a dynamic marking for a single note (this applies to single note marking such as an accent). The second is a ramp function which gradually increases or decreases a value from note to note (useful for ***crescendo*** or ***decresenco*** markings which apply to a specified number of notes). The last is a constant function which is either on or off for a set of notes at a time (useful for markings such as ***p*** or ***f*** which apply to a group of notes at a time). Each

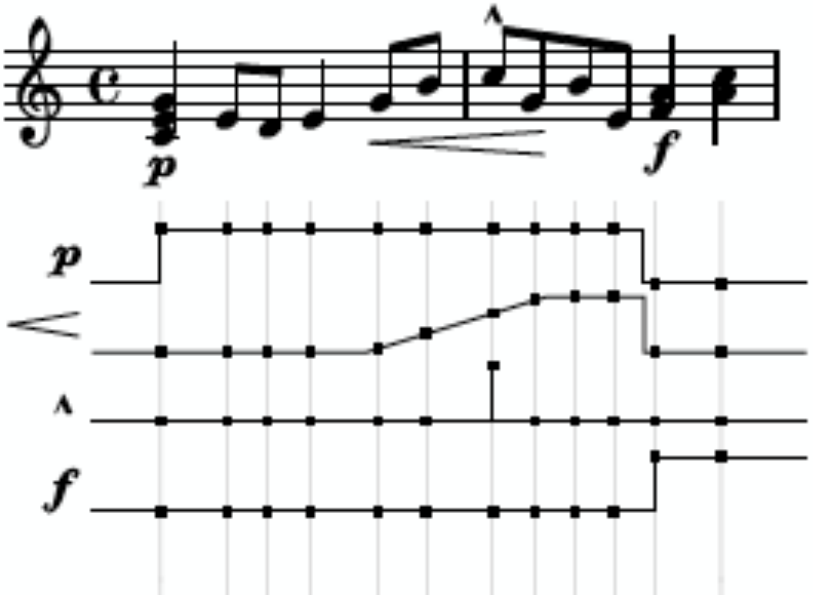

Figure 4.1: An illustration of the different dynamics basis functions, taken from Eduardo [11]. The function describing $\boldsymbol{p}$ is a constant function, which is activated for every note until the presence of a different dynamic marking, $\boldsymbol{f}$. The function describing the crescendo (the second from the top) is a ramp function and gradually increases until the crescendo finishes. The function describing the accent (third from the top) is an impulse function and holds a value of 1 for the note which contains the accent mark.

dynamic function is shown graphically in figure 4.1.

- **Inter-Onset-Interval (IOI)**: This includes a total of six basis functions which include the difference in onset times of a group of notes belonging to a particular note onset o_i an its surrounding note onsets. The IOI is calculated for the onsets between $(i-2, i-3), (i-1, i-2), (i, i-1), (i, i+1), (i+1, i+2),$ and $(i+2, i+3)$. These functions provide context for the local tempo structure of the composition.

- **Duration**: A basis function that encodes the notated duration of a single note.

Expressive Encoding Functions

- **MIDI Velocity**: This encoding function uses the recorded hammer-velocities of the

performed notes as a proxy for expressive dynamics. The loudness of a note is estimated as a normalized MIDI velocity value.

$$y_{vel}(n_i) = \frac{\text{vel}(n_i)}{127}$$

- **log BPR**: The Beat Period Ratio (BPR) is a way to define the local tempo calculated for every note. It uses the Beat Period (BP), defined as

$$BP(n_i) = \frac{\text{IOI}^{perf}_{o(n_i)}}{\text{IOI}^{score}_{o(n_i)}}$$

where $\text{IOI}^{perf}_{o(n_i)}$ and $\text{IOI}^{score}_{o(n_i)}$ are the IOI for the performance and score onsets at note n_i. The final log BPR is defined as

$$y_{\text{log bpr}}(n_i) = log_2 \frac{\text{BP}(n_i)}{\text{BP}_{ave}},$$

where BP_{ave} is the average BP over the entire piece. This normalizes the local BPR definition according the global tempo of the piece.

- **Timing (deviation)**: Whereas the log BPR defined above encodes the tempo of a performance (how "fast" it is being played), the function for timing encodes expressive timing deviations from the indicated timing of a score. It is defined as

$$y_{\text{tim}} = \hat{o}^{perf}(n_i) - \text{onset}_{perf}(n_i)$$

where $\hat{o}^{perf}(n_i)$ is an estimation of the "correct" onset time, and $\text{onset}_{perf}(n_i)$ is the actual observed onset of a note (see section 3.3.1 of Eduardo [11] for a full working example).

- **log Articulation**: Articulation is a measure of the lengthening or shortening of a performed note duration relative to a noted duration and the current local tempo. The articulation is computed by dividing the measured note duration by the duration given by the score. It is defined as

$$y_{\log \text{ art}}(n_i) = \log_2 \frac{\text{duration}_{perf}(n_i)}{\text{duration}(n_i) BP(n_i)}$$

Models

BM framework computational models first started as simple linear non-sequential models that learned the relationship between a set of defined basis functions and a single expressive parameter, such as MIDI velocity. This version of the BM models each expressive parameter independently from all others and implies that one expressive parameter's interpretation will not affect the other. Although this is not necessarily the case in actual performance [2], it is a simplifying mathematical assumption that makes the models' development and interpretation simpler. Both standard least squares regression and a probabilistic Bayesian approach are used to model the linear relationship.

The BM framework later introduced non-linear and sequential models, both as ANNs. A standard Feed-Forward Neural Network (FFNN) added the capability for non-linear modeling. This model showed an increase in both the goodness-of-fit and predictive accuracy over the linear model. A standard RNN was used to implement the sequential model with features where the time-dependent and sequential nature of music was relevant. Used in conjunction with a FFNN, this model performed the best relative to all other models.

[2] For example, the effect of a ***crescendo*** marking may affect both the dynamics and the tempo at the same time

4.1.3 Data-Based EMP Models: VirtuosoNet

VirtuosoNet, similarly to the BM framework, is a complete end-to-end system for training EMP models and generating novel performance. Although VirtuosoNet does not explicitly define a clear mathematical framework and abstract representations of EMP, as does the BM framework, it contains all of the moving parts necessary to train an EMP model from scratch. We make a distinction between VirtuosoNet as a set of NN-based EMP models and the VirtuosoNet framework, which comes with all of the necessary data processing and featuring engineering as well as model training.

Features

The VirtuosoNet framework defines both input and output features. Input features contain standard score data such as pitch, timing, key, and metric information. The VirtuosoNet framework uses these features as well as more detailed note level information such as the duration of rests, articulation markings, and others. The framework's output features are similar to those of the BM defined above and include tempo, note onset deviation, MIDI velocity, and articulation. Additionally, the VirtuosoNet framework, unlike any other EMP generation model, uses features to model the pedal directly. They include the pedal value at note onset, pedal value at note offset, minimum pedal value between note onset and note offset, and minimum pedal value between note offset and the next note's onset. A full list of data features is given by Jeong et al. [25].

Models

The development of VirtuosoNet models is gradual, although the general model architecture is the same. The model comprises three parts; a score encoder, a performance encoder, and a

performance decoder. The score encoder learns a score representation C, which is a sequence with the same length as the input notes. The performance encoder's job is to learn the style of a particular performance, and it encodes this into a style vector z. This style vector guides performance generation so that the model can render different performances given the same input, depending on the learned style. The performance decoder is a generative model which uses the score encoding C and the style vector z to generate the resulting performance.

The first version of the VirtuosoNet model presented by Jeong et al. [22] is a primitive model and forms the basis for later versions. It introduces a recurrent hierarchical attention network (HAN) made up of hierarchical LSTM layers used in conjunction with the attention mechanism. The HAN forms the building block for all of the different components of VirtuosoNet. This model's dataset is an early version of the KAIST dataset and consists of Chopin performances taken from the Piano-e-competition, with 25 compositions and 217 full performances. No quantitative or qualitative experiment results are given in the first formulation.

The next iteration of VirtuosoNet builds from the initial HAN layers but also uses a novel graph-based data representation and model. This model is defined as an iterative sequential graph-based neural network (ISGN). The score is represented as a graph where the nodes are the individual notes. The graph's edges define relationships between the notes; one example is an edge indicating two notes with the same onset (which implies they belong to the same chord), another indicating notes belonging to a slur. A gated graph neural network (GGNN) is used to model the note-level graph representation and is combined iteratively with a HAN to model measure-level features.

The latest and best-performing version of VirtuosoNet [23] returns to the HAN only architecture. It uses an LSTM based HAN network for every different component of the architecture, explicitly defining the hierarchical models according to various musical boundaries, includ-

ing note, beat, and measure levels. This model adds additional more abstract hierarchical models that exist to create better structure at the metrical level and preserve patterns across the composition's mid-level structures.

Both the ISGN [24] and HAN [23] version of VirtuosoNet are trained using the full KAIST dataset and use the same evaluation methods; MSE for the quantitative evaluation and listening tests for the qualitative. The final version of HAN reports better MSE metrics than the ISGN. The qualitative evaluation shows that both the ISGN and HAN perform better than baseline models and better than a "deadpan" performance. The HAN version's qualitative evaluation also includes a comparison between the HAN and the publicly available version of the BM framework model [3].

The results in [23] show that the HAN performs better than the BM model. Many plausible reasons may explain the difference in results other than the HAN being a superior model to the BM. These include differences in the training data for both models, a bias of the qualitative method towards the HAN, and the fact that the listening test members' opinion doesn't necessarily imply one model being "superior" to another. However, given the results presented by Jeong et al. [23], we will assume that this version of the HAN represents the current "state-of-the-art" in the field.

4.2 Music Generation with Transformers

The Transformer has seen some application in music generation. Such models use a discrete representation of music as a sequence of events with highly sparse encoding vectors. This representation is consistent with the word embeddings used to train most NLP models and

[3]The website for the BM model can be found https://basismixer.cp.jku.at/static/app.html. At the time of this writing, the website is currently unavailable

so has a natural extension in Transformers, designed with language data in mind.

The first model to use the Transformer for music generation is known as the "Music Transformer" [21]. It directly builds from the PerformanceRNN of Oore et al. [31]. PerformanceRNN uses the Piano-e-competition data to generate MIDI directly. This model *simultaenously generates a composition and a performance*, combining the role of the composer and performer as presented in figure 2.1. PerformanceRNN uses an event-based representation of MIDI, and all input tokens are a one-hot encoding over the different possible MIDI events. There are 128 possible NOTE-ON events, 128 possible NOTE-OFF events, 125 possible TIME-SHIFT events, and 32 possible VELOCITY events, which leads to a 413-dimensional one-hot encoded vector. PerformanceRNN is an autoregressive LSTM model that models the probability of generating the next note given all previous notes, similar to a language model.

Music Transformer uses the original Transformer architecture and implementation of Vaswani et al. [38] with some adaptation and applies it to the same data set and representation of the PerformanceRNN. One crucial component of the original Transformer architecture we have not mentioned is its positional embedding. This embedding encodes the sequential nature of the data into the model, which is otherwise not accounted for using only the attention mechanism. The original positional embedding is "absolute", meaning that the embedding only encodes information about each element's position in relation to the beginning of the sequence. Music Transformer uses a "relative" position, which instead adds information about how far apart two positions in the sequence are. The relative positional embedding accounts for every possible pairwise distance of each element. The attention mechanism processes this information throughout the rest of the model. It allows attention to account for the distance of any two notes to each other, rather than the global position of a single note in the sequence.

The Transformer architecture with the relative positional embedding outperforms other baseline models, including the LSTM based PerformanceRNN. Qualitative evaluation shows that the Music Transformer generates performance with much better long term structure and overall musical cohesiveness than the performances from the PerformanceRNN[4]. This result is consistent with the observation that attention has a better memory than the hidden state of an RNN cell.

Building from the Music Transformer, Open AI introduced MuseNet [33]. MuseNet is both philosophically and architecturally similar to GPT. It uses an enormous decoder-only Transformer model and trains it on a massive amount of data to build a model capable of generating novel music across many different domains. MuseNet is trained using only MIDI data gathered from various internet sources (including the MAESTRO dataset) and uses an event-based token representation similar to the Music Transformer. In contrast to the Music Transformer, which only operates with western classical solo piano music, MuseNet can generate music across a variety of genres in multiple instruments. Open AI has not directly released any research results comparing the benchmark results of MuseNet to other models. Still, it is evident from listening to MuseNet samples[5] that it generates high-quality music.

Open AI also released another music generation model which uses Transformers, JukeBox [9], that deals directly in the audio domain. The choice to directly model audio is motivated by the fact that symbolic data forms don't contain enough information to capture musical performance's subtleties across a variety of different instruments[6]. JukeBox is comprised of a Vector Quantized Variational AutoEncoder (VQ-VAE), which learns to encode the audio data into a low-dimensional representation, and a Transformer model similar to MuseNet,

[4]Sample performances for both the PerformanceRNN and Music Transformer are available online from the Magenta Research group of Google AI.

[5]https://openai.com/blog/musenet/

[6]It is also motivated by the abundance of audio data compared to symbolic, which we discussed in chapter 2

which uses the encoding to generate new music. Although the samples generated by JukeBox are impressive in their sound quality and overall musical cohesiveness, much work is still needed in the direct audio generation domain to produce audio of high enough quality that it competes with human productions.

Chapter 5

Methods and Experiments Results

Given the powerful advances in NLP and music generation tasks, our general goal was to investigate the Transformer model in application to EMP generation. Both language and music are by nature sequential and hierarchical. As such, one would expect the strengths of a Transformer model, created specifically for an NLP task, to have natural extensions into musical tasks (as was shown by the "Music Transformer" and OpenAI's generative models). This project's initial purpose was to determine if a Transformer-based model improves upon VirtuosoNet, given the same data, features, and evaluation metrics. However, as discussed in section 6, answering that question proved to be more complicated than initially anticipated and led our research in a different direction. We present our initial method and experiment results in this chapter.

We use the complete end-to-end generation system of the VirtuosoNet framework. There are three major reasons for adopting the VirtuosoNet system.

1. The KAIST dataset is the largest publicly available dataset used in EMP generation.

2. VirtuosoNet models are the "best performing" in EMP.

3. The data processing and model architecture code of the VirtuosoNet framework are open-sourced[1].

[1]https://github.com/jdasam/virtuosoNet

If we consider VirtuosoNet to be state-of-the-art in EMP generation, it provides a natural starting place to compare against any further model development.

5.1 Data and Features

The VirtuosoNet system uses handcrafted features for both scores and performances, which are outlined in section 4. We use the same feature set.

5.2 Our Model

In the VirtuosoNet system, there is a 1:1 mapping between notes in scores and performances. As mentioned in chapter 3, encoder-decoder *seq2seq* models with variable-length input/output sequences account for the difference in length by using a generative decoder model which relies on the input encoding. Because our input and output sequences are the same lengths, we saw no need to include a generative decoder. Our model is an encoder-only Transformer model, ~~conceptually similar to BERT~~. Sitting on top of the Transformer encoder is a simple FFNN that takes as input the learned representation from the Transformer and maps it to the final output feature set. We use the original absolute positional embedding of Vaswani et al. [38]. The model architecture is shown in figure 5.1

5.3 Experiment Methods

Our initial experiments were designed to answer two questions:

1. Does a Transformer based EMP generation model outperform a similar LSTM based

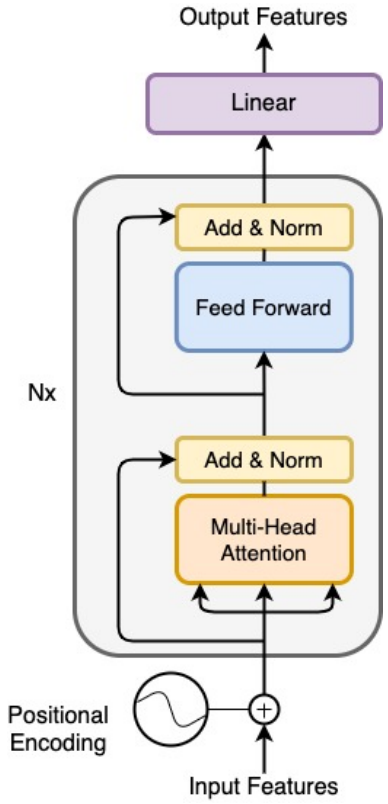

Figure 5.1: Our model uses the same encoder Transformer architecture as shown by Vaswani et al. [38]. There are N Transformer layers which are composed of multi-head attention and feed-forward mechanisms. There is a simple linear layer at the top of the architecture which creates the final mapping to the output features.

EMP generation model?

2. Does a Transformer based EMP generation model outperform existing state-of-the-art (VirtuosoNet) models?

To answer both of these questions, we ran all of our experiments using the existing VirtuosoNet framework. The VirtuosoNet framework defines real-valued features, and the resulting models are regression models. All VirtuosoNet models use MSE as both the loss function for training and reconstruction error for quantitative evaluation. VirtuosoNet models are trained using an 8-1-1 train/valid/test data split, and Jeong et al. [23] present MSE results for several different models. These MSE results are given for each different expressive

parameter, including tempo, velocity, deviation (timing), articulation, and pedal. We follow the same method for our quantitative evaluation, using the same data split, loss function, and evaluation metrics.

Our models were trained at 50 epochs, and the best model parameters were selected according to the lowest validation evaluation score. Our models were implemented in PyTorch using their native Transformer library. We used Neptune AI [29] to manage our experiments and report the metric feedback. Data for all of our experiments, including the model hyperparameters and metrics, can be found online[2].

Our main experimental goal was to find the best model configuration for a Transformer based model. To do so, we ran a reasonably exhaustive hyperparameter search for our Transformer model. To guide our search, we trained our baseline LSTM model and a few VirtuosoNet models in tandem with our Transformer model. We used the evaluation results from the validation set to gain an intuition about which family of models performed the best. The final test set evaluation happened after all model development was finished.

5.4 Model Evaluation: Quantitative

We use several different model configurations for the Transformer. Our Transformer baseline has six layers, six attention heads, and a hidden size of 256. We chose this as a base configuration because it closely matches the size of the original Transformer [38], except for the hidden size of the feed-forward layer. We use a smaller hidden size of 256 for our base layer hidden size due to the relatively small dimensionality, 78, of the model's input data[3].

[2] http://ui.neptune.ai/richt3211/thesis

[3] The original Transformer uses a baseline input dimensionality of 528. In this case, the model inputs are word vectors whose size can be configured dynamically to find the best word representation for a given model. In our case, our input size dimensionality is fixed according to the input features of the VirtuosoNet framework

Model Configuration								Results in MSE					
N_{id}	M	L	d_{hid}	D	LR	C	H	Tot	t	v	d	a	p
123	**LSTM**	3	256	0.1	0.1	0.5		**1.08**	0.84	1.35	1.02	1.11	1.08
147	T-BL	6	256	0.1	3e-5	0.5	6	0.86	0.54	0.80	0.88	0.80	0.92
169			128					0.87	0.55	0.79	0.96	0.88	0.92
128			528					0.86	0.50	0.76	0.88	0.82	0.93
133	**T-Best**		1024					**0.83**	0.47	0.76	0.88	0.82	0.88
118		12						0.89	0.65	0.82	0.88	0.82	0.95
181		24						0.93	0.62	0.97	0.89	1.09	0.95
132							13	0.84	0.51	0.77	0.95	0.81	0.88
171				0.2				0.91	0.74	0.82	0.95	0.86	0.94
173					0.01			1.01	0.86	1.05	0.90	1.18	1.01
188							26	0.84	0.62	0.78	0.89	0.79	0.87
134		12	528					0.85	0.54	0.75	0.90	0.86	0.89
190		12					13	0.87	0.49	0.78	0.89	0.87	0.94
135		12	528				13	0.84	0.47	0.81	0.89	0.86	0.89
125	T-Worst	24	528					0.93	0.69	0.99	0.91	1.12	0.94
	HAN-BL	-	-	-	-	-	-	0.77	0.40	0.67	0.77	0.72	0.84
	HAN-S	-	-	-	-	-	-	0.73	0.27	0.61	0.75	0.69	0.82
	HAN-M	-	-	-	-	-	-	**0.72**	0.22	0.53	0.75	0.75	0.81

Table 5.1: A comparison of 3 different families of EMP generation models: virtuosoNet models, Transformer models, and our LSTM baseline models. N_{id} is the ID of the Neptune experiment, L is the number of layers, d_{hid} is the dimension of the hidden layers, D is the dropout, LR is the learning rate, C is the gradient clip, and H is the number of attention heads. Empty spaces for the Transformer configuration values imply the baseline value (for example, the number of layers for model 169 is 6). The right side of the table presents the MSE results for all models along the five different expressive parameters used in the VirtuosoNet framework, as well as the total MSE which is an aggregation of all the individual expressive features. The entries for the HAN models come from virtuosoNet and are given in [23]

Table 5.1 contains the full results of the final evaluation and includes MSE results for all of the different Transformer models, our baseline LSTM model, and the best performing VirtuosoNet.

As shown in figure 5.2, the first major implication of the results is that the Transformer model easily outperforms the baseline LSTM but does not outperform the VirtuosoNet models. The best performing Transformer model uses all of the same baseline Transformer parameters, except for a major increase in the hidden layer's dimensionality (this is found in model $T_{N_{133}}$, we use the notation T_{N_i} to refer to the Neptune ID of a Transformer model.). Other models that perform well include those which only increase the number of attention heads ($T_{N_{132}}$ and $T_{N_{26}}$) and those with an increase across several different parameters ($T_{N_{135}}$). It appears that increasing the number of layers degrades performance ($T_{N_{118}}$, $T_{N_{181}}$, $T_{N_{125}}$), and so does increasing the dropout and learning rates ($T_{N_{171}}$, $T_{N_{173}}$).

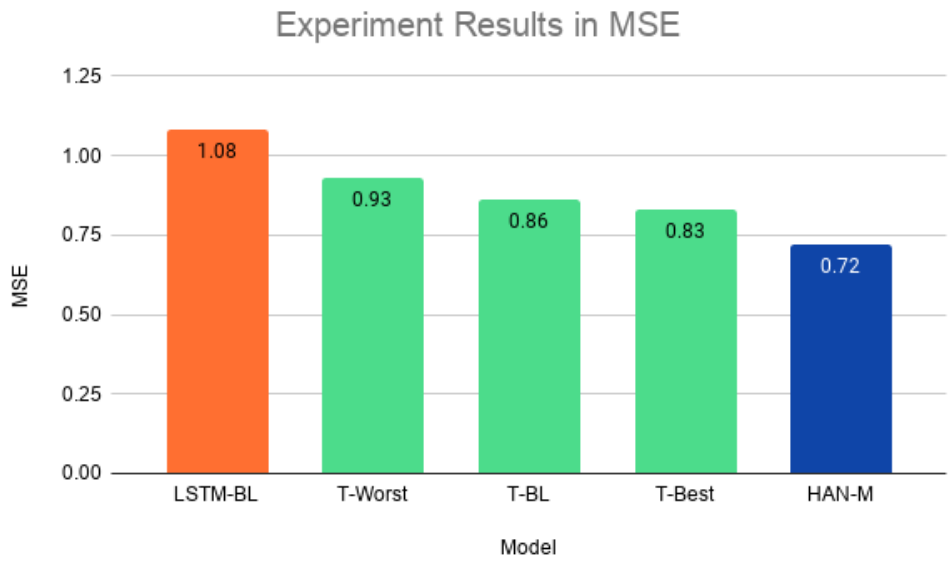

Figure 5.2: Results for the three different model families, LSTM baseline, Transformer, and VirtuosoNet, are shown in different colors. There is a significant difference in results for each model family, with the LSTM baseline performing the worst, VirtuosoNet performing the best, and the Transformers sitting in between.

Although we see improvements by increasing the hidden dimension and number of attention heads, the performance improvements aren't significant. There is only a difference of 0.03 in MSE between the best performing Transformer model and the Transformer baseline. In contrast, there is a difference of 0.11 between the best Transformer model and the best HAN model. An initial interpretation of these results would indicate that a Transformer model can't even match the existing state of the art in EMP generation (let alone beat it). As we discuss in later sections, this interpretation of results is not necessarily valid.

Chapter 6

Analysis

Our initial experiments using a quantitative evaluation indicated that our Transformer model does not improve the state-of-the-art in EMP generation. As a method of "debugging" our Transformer model to look for potential errors and improvements, we wanted to listen to the performances generated by our models to identify any problems, if they existed at all.

These listening tests acted both as an error analysis as well as our qualitative evaluation of the models. It became apparent to us right away that there was a disconnect between the quality of the model according to the quantitative metric and our evaluation. In this chapter, we present this analysis. The analysis includes our quantitative evaluation method, new experimental methods to improve our models, and conclusions about each model's quality according to our listening tests.

6.1 Evaluation Error Analysis

Listening tests require generated performances, which we produced by using the VirtuosoNet framework to create MIDI files given our models' feature output. We generated performances for several of our trained models using six compositions from three different composers, listed in table 6.1. For easy comparison of each model's performance, we loaded their relevant MIDI files into the Digital Audio Workstation (DAW) Logic Pro X. We used the "Steinway Grand" instrument synthesizer to create the audio. Logic Pro X also provides visualizations of differ-

Composer	Composition
Bach	Prelude in E Minor, BWV 855
Bach	Prelude in F-sharp Major, BWV 858
Chopin	Etude Op. 10, No. 12
Chopin	Fantaisie-Imropmptu
Beethoven	Piano Sonata No. 17 First Movement
Mozart	Piano Sonata No. 11 First Movement

Table 6.1: The compositions used for the qualitative evaluation of our models. All scores come in the form of MusicXML from MuseScore. None of the scores were present in the training data

ent performance aspects, including the MIDI velocity of each note and the sustain and soft pedals' value throughout the performance (see figure 2.3 for example pedal visualizations)[1].

As already mentioned, it was apparent that there was a mismatch between a model's performance quality according to the MSE value and our evaluation. For example, the Transformer model $T_{N_{125}}$ had much worse validation MSE metrics than almost every other model Transformer model. However, our listening test revealed that the absolute tempo for smaller models, such as the Transformer baseline $T_{N_{147}}$, was much faster and sounded worse (to the point where the performances are almost "unlistenable") than $T_{N_{125}}$. The potential disconnect between the model's quality as determined by quantitative and qualitative evaluation led us to investigate possible problems with the training methods used by Jeong et al. [23].

One of the first potential problems we identified was the MSE loss and evaluation function. The output features of virtuosoNet are a sequence of vectors with a length of 11. The first four features correspond to a single component of expression. They are tempo, velocity,

[1]Performances generated by models are available through our Neptune Project. Each experiment has an ID, and we've run performance generation code for many of the models. To listen to performances, visit an experiments' artifacts tab and download available performance MIDI files. These MIDI files can be played in any DAW software, such as Logic Pro X. If performances don't exist for an experiment, please contact the author

deviation, and articulation, respectively. The last seven features are all different values that correspond to information about the sustain pedal [25]. In table 5.1 we show MSE metrics for five different expressive parameters, including all the four features previously mentioned and the pedal. Our reference to pedal MSE is an aggregation of the seven different features that contain pedal information. In contrast, the other 4 MSE metrics only apply to a single feature.

The original MSE used to train VirtuosoNet (which we call VirtuosoNet MSE) assumes that every individual feature of the output vector contributes equally to the final output and corresponding loss optimization. The pedal parameter has seven times as much information as every other parameter, and so the VirtuosoNet framework MSE places more importance on pedal quality over every other feature. Given that the VirtuosoNet framework MSE loss is used both for model training and evaluation, this means that it inherently rewards those models which express pedal better than all other performance features. If the VirtuosoNet models outperformed our Transformer models according to this metric, does that mean that they are better at modeling EMP generation as a whole, or that they are simply more fit to model the pedal? In an attempt to answer this question, we came up with a new weighted MSE loss function that allows for a more fair representation of the evaluation method.

We define the output vector as an 11 dimensional vector

$$\mathbf{v} = \{t, v, d, a, p_0, p_1, p_2, p_3, p_4, p_5, p_6\}$$

where t, v, d, and a represent tempo, velocity, deviation, and articulation respectively, and p_i represents a single component of the pedal. For a predicted output vector $\mathbf{v}$ and the target

output vector $\hat{\mathbf{v}}$, VirtuosoNet framework MSE loss is

$$MSE(\mathbf{v}, \hat{\mathbf{v}}) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{v}_i - \hat{\mathbf{v}}_i)^2$$

This can also be re-written as

$$MSE(\mathbf{v}, \hat{\mathbf{v}}) = \frac{1}{11}[(\mathbf{v}_t - \hat{\mathbf{v}}_t)^2 + (\mathbf{v}_v - \hat{\mathbf{v}}_v)^2 + (\mathbf{v}_d - \hat{\mathbf{v}}_d)^2 + (\mathbf{v}_a - \hat{\mathbf{v}}_a)^2 + \sum_{i=1}^{7}(\mathbf{v}_{p_i} - \hat{\mathbf{v}}_{p_i})^2]$$

We introduce 5 different weight values: $\alpha_t, \alpha_v, \alpha_d, \alpha_a$ and α_p. Our weighted MSE loss is defined as

$$W_{MSE}(\mathbf{v}, \hat{\mathbf{v}}) = \frac{1}{\alpha_t + \alpha_v + \alpha_d + \alpha_a + \alpha_p}[\alpha_t(\mathbf{v}_t - \hat{\mathbf{v}}_t)^2 + \alpha_v(\mathbf{v}_v - \hat{\mathbf{v}}_v)^2 + \alpha_d(\mathbf{v}_d - \hat{\mathbf{v}}_d)^2 + \alpha_a(\mathbf{v}_a - \hat{\mathbf{v}}_a)^2 + \alpha_p\frac{1}{7}\sum_{i=1}^{7}(\mathbf{v}_{p_i} - \hat{\mathbf{v}}_{p_i})^2]$$

VirtuosoNet framework MSE can be seen as the weighted MSE with $\alpha_t, \alpha_v, \alpha_d, \alpha_a = 1$, and $\alpha_p = 7$.

This loss formulation shows the heavy bias towards the pedal feature. Is this is the right way to conceptualize what a 'good' performance is? Would a different configuration of the expressive feature weights lead to a better outcome? Answering such questions in a straightforward and 'objective' according to a quantitative metric is not plausible with our weighted MSE. Every different configuration of weights changes the evaluation metric itself, and therefore any direct metric comparison across experiments is meaningless. Knowing this, we continued to experiment with different weight configurations relying on our qualitative evaluation to determine performance quality. We present our observations from listening to model performances, with and without the change in MSE.

6.2 Qualitative Evaluation

Our first general observation given our new loss evaluation is that the two most important factors for overall performance are tempo and pedal. Performances with either of these two features outside of certain bounds make performances unlistenable. Suppose a performance's global tempo is too fast, and every other expressive parameter renders correctly. In that case, the resulting performance will still sound bad enough that it's not worth listening to at all [2]. We noticed a similar phenomenon with the pedal. Some models generated performances with the sustain pedal applied at all times with very few breaks, and resulting performances are ‘‘muddied" and unrefined. Although these performances are more bearable than those with extreme tempo, they are still hard to listen to in any meaningful way [3].

We also notice that the tempo and timing of the Transformer models are more dynamic than the LSTM models, both for our LSTM baseline and the VirtuosoNet models. For some models, the variability in timing seemed to be a good thing, while for others, it was bad enough that it almost sounded like the model was still "learning" how to play. The tempo for all LSTM based models (except for some slight variations in the performances from HAN-M [4]) was consistent and non-changing to the point of sounding robotic, almost like a deadpan performance. On one extreme with Transformer models, the highly dynamic tempo at times sounds like a real performer making mistakes [5]. The other extreme is performances of some LSTM models that are so mundane they don't sound "human" at all [6].

The pedaling in general of all models was mediocre at best. This observation is consistent with those of Jeong et al. [23]. There are models with decent performance pedaling quality

[2] See $T_{N_{86}}$ Fantaisie Impromptu and $T_{N_{126}}$ Etude Op. 10 No. 12.
[3] See $T_{N_{125}}$ Piano Sonata 11
[4] Performances for the HAN-M are available at N_{126}
[5] See $T_{N_{86}}$ Piano Sonata 11
[6] See $LSTM_{N_{123}}$ Piano Sonata 11

Model Configuration				Expressive Weights				
N_{id}	L	d_{hid}	H	α_t	α_v	α_d	α_a	α_p
150	256	6	6	1	1	1	1	7
154				0.2	0.2	0.2	0.2	0.2
156				0.33	0.11	0.11	0.11	0.33
157				0.4	0.067	0.067	0.067	0.4
159	528	12	13	0.4	0.067	0.067	0.067	0.4

Table 6.2: The model configurations of additional experiments we ran after our initial quantitative evaluation effort. We show similar hyperparemters as in table 5.1. There are additional paramter values that are not present but are used in table 5.1: LR is 0.0003, C is 0.5, and D is 0.1

that somewhat follows the composition's natural cadence but does not match pedal in actual human performance.

Our first experiments with the weighted MSE used an even weight distribution. We ran a few additional experiments with the tempo and pedal weights higher than others, along with some model size changes. Table 6.2 shows a full description of the models and their parameters.

$T_{N_{154}}$, which weights all expressive parameters evenly, generates performances with the global tempo slightly too fast and an extremely muddy pedal. $T_{N_{150}}$, which uses the VirtuosoNet MSE, produces better performances with reasonable pedaling. However, the tempo is inconsistent enough that the performance loses its cohesiveness as a whole. To further test our hypothesis that the tempo and pedal are the most important expressive parameters, we increase the weights for both tempo and pedaling higher than all other parameters in model $T_{N_{156}}$. We found that the tempo and pedal weights $T_{N_{156}}$ were a bit too low - specifically, the pedal is almost just as muddy as it is in $T_{N_{150}}$ and the tempo is still a bit too fast, albeit more consistent and cohesive than $T_{N_{150}}$. We further increased the tempo and pedal weights

in $T_{N_{157}}$ and $T_{N_{159}}$ and found that $T_{N_{157}}$ is the best sounding Transformer model[7]. It is likely that were we to continue to experiment with a different configuration of the weights that we could come up with increasingly better-sounding models.

Our last general observation is that the VirtuosoNet model HAN-M produces the best overall performances. In general, we feel that the tempo for HAN-M is a little too slow, but it still creates the most natural expression. This expression is most apparent in its performance of Beethoven's Piano Sonata 17[8], whose introduction leaves a large space for interpretation to achieve the desired listening result. We found that the only model that made this performance "interesting" was the HAN-M. Although we have previously emphasized the problem using the existing quantitative metric to evaluate our models, our quantitative and qualitative evaluation of the HAN-M model indicates that it is the "best" model. The proposed Transformer architecture does not improve upon existing models. We will provide some intuition about why this is and possible model improvements for future work in chapter 7

[7]Performances of Fantaisie Impromptu best demonstrate these differences. Compare $T_{N_{154}}$, $T_{N_{150}}$, $T_{N_{156}}$, and $T_{N_{157}}$

[8]See $T_{N_{162}}$

Chapter 7

Discussion

The main research question we set out to answer was whether or not a Transformer model could outperform the existing state-of-the-art in EMP generation – our experiment results do not indicate that is the case. We provide some hypotheses that explain the difference in model quality, which are centered around problems with the experiment method (evaluation) and our model architecture. The plausible explanations for a poorer performing model also form the foundation for future experiments in the area.

7.1 Better Evaluation

The most interesting observation from our experiment results is the mismatch between quantitative and qualitative evaluation. Although both evaluation methods identified the same model as the best performing, several mismatches were found in other experiments relating to our Transformer models. These mismatches may be explained by the particular VirtuosoNet framework feature definitions which, as we showed in chapter 6, directly affect the output of the MSE metric. Because the VirtuosoNet framework MSE weighs pedal as more important than all other expressive parameters, a model which completely fails to output a decent tempo but excels at pedaling may be evaluated as a "high-quality" model. Such an evaluation is not desirable, as tempo is an essential component of the overall listening experience. Any attempt to modify the metric calculation (as we did) or change the feature

definitions makes the comparison interpretability of different models impossible.

For these reasons, we call to question the validity of using MSE or other similar metrics for quantitatively evaluating EMP models. Although the current evaluation metrics may facilitate the proper evaluation for some aspects of performance, they are too dependent on the underlying feature definitions. The definition of features is an ongoing and evolving process that is an essential component of model development. The lack of consistent and standard evaluation methods in EMP performance (and musical performance in general) creates a problematic space for conducting useful experiments. Although this limitation can be overcome using a sophisticated qualitative evaluation involving listening tests by professional musicians, such an evaluation method is time-consuming, expensive, and potentially inaccessible.

We again draw an analogy to current research methods in NLP and machine translation to provide some concrete suggestions for developing better quantitative evaluation metrics. MT used to face a similar set of evaluation problems that currently plague EMP. MT evaluation was mostly conducted by humans, which (although extensive) was too expensive, took too long, and wasn't consistent or reliable. The Bilingual Evaluation Understudy (BLEU) metric [32] addresses these complications. The BLEU metric's two fundamental principles are that a computed translation's quality is determined by how closely it can match an existing human translation and that a single output translation should compare against multiple reference translations *at the same time*. The BLEU metric is also computed by comparing the output text from MT models instead of directly using the output features that are often specific to a model, as does MSE.

It would make sense for an EMP generation evaluation metric to loosely match the principles of BLEU. The first improvement to be made is to compare the final output representation (MIDI) rather than directly use the defined feature set. Using MIDI would allow easy

comparison of models that use different feature representations and significantly improve the robustness (or lack) of experiment results. The other potential improvement would include comparing a single generated performance to several reference performances at the same time and reporting a single metric number that represents the aggregate reference comparison. Although the underlying mathematics of evaluation widely differs between EMP and MT, the general principles that guide the evaluation can be the same.

7.2 Possible Model Improvements

With the understanding that our quantitative evaluation metrics may not properly indicate the quality of our models, we provide some hypotheses about why the Transformer model was outperformed by VirtuosoNet. They are centered around high-level components of the model architecture as opposed to the underlying mechanisms (attention in the Transformer and the hidden state in an RNN) for the different modeling domains.

The first plausible explanation for the difference in performance between VirtuosoNet and our Transformer model is that VirtuosoNet models explicitly define the hierarchical layers of music into the model. The Transformer models, which are a straight implementation of the original architecture, have no such "musical knowledge" baked in. VirtuosoNet models are primarily composed of hierarchically based attention layers defined at note, beat, and measure levels[1].

Our initial expectation was that we would see similar results because the Transformer outperforms other comparative models in NLP. We did see a significant improvement over our baseline LSTM model, which is architecturally similar to our Transformer model. This in-

[1]This type of hierarchical structure is something that we would expect the Transformer model to learn given its composition of stacked attention-based layers. Although we never ran experiments to verify this, each layer should be learning a more abstract representation of the score.

dicates that the attention mechanism in a Transformer does improve upon the hidden state recurrence mechanism of an RNN. However, it appears that the "hardcoded" musical information in VirtuosoNet models is more important than merely throwing a different type of sequential computation at the problem.

One interesting observation from the qualitative evaluation is that the Transformer models were more dynamic, both good and bad for the resulting performances. A model that is too dynamic (especially in tempo) lacks the cohesiveness that makes performance identifiable. Performance that isn't dynamic enough is musically uninteresting. The desire to create more interesting computational performances is the driving factor for using ML to build computational models, given the highly undesirable deadpan performances. Transformer models' dynamics indicate that they are potentially more expressive and creative, which are desired traits in generating novel performance. It could be the case that a Transformer architecture that explicitly defines hierarchical musical layers in the model results in performances that have both consistency and creativity.

The other primary plausible explanation for the difference in model performance is that we lost a fundamental component of expressiveness by removing the Transformer decoder. We chose an encoder-only Transformer model based on the VirtuosoNet framework definition of EMP as a one-to-one mapping between a note in the score and its corresponding expression in performance. We also chose this model because of the success of similar Transformer adaptations such as BERT [8]. As discussed in chapter 3, the original *encoder-decoder* Transformer architecture maps between *variable* length sequences. The decoder uses an internal representation of the input from the encoder as well as an autoregressive model to *generate* from scratch the target sequence. Although our model learns a one-to-one mapping, we believe that the full *encoder-decoder* architecture may better model EMP in general. From a philosophical perspective, the *encoder-decoder* assumes that *generation by the performer* is as fundamental a

component of the performance process as is understanding a score. An encoder-only model assumes that performance is encapsulated only by apprehending the score. Removing the decoder cuts out creative freedom on the part of the performer.

VirtuosoNet is an encoder-decoder architecture and does not lose this fundamental component of performance, as does our model. We suspect that a full encoder-decoder Transformer architecture would have better capability to create both consistent and creative performance. Although it is possible to develop a model which explicitly defines the hierarchical layers as does VirtuosoNet, merely adding the decoder may be enough to build a better performing model[2].

[2]We attempted to implement such a model after our initial experiments but ran into practical problems with the model dimensionality using PyTorch's native Transformer implementation

Chapter 8

Conclusion

As we have outlined, this project underwent an unexpected evolution of purpose. Our initial goal was to determine if we could outperform the existing state-of-the-art EMP generation system with the same research methods and only a computational model change. The initial quantitative experiment results indicated that a Transformer does not outperform hierarchical-based recurrent models. However, when we ran our subjective evaluation comparing performance among the two families of models, we determined that declaring one model as better than another is not straightforward. This ambiguity led us to question the validity of the original research method, particularly with the evaluation metric, and shift our own toward more nuanced and discovery-based experiments. We want to use our experience in running these experiments to provide some high-level thoughts on some philosophical related to computational EMP.

8.1 Looking Forward: Finding the *Essence* of Performance

As we discussed and developed this project, one thought that plagued us is that we don't have a proper understanding of fundamental components that comprise musical performance. Without a strong background in music, it was hard to understand how to build and ana-

lyze our computational models. Widmer, in his *Con Espressione*[1] Manifesto [40], emphasizes the importance of focusing on discovering the *essence* of music itself and using a more in-depth understanding to develop more impactful technology related to MIR. When we first started the project, we thought it would be relatively simple to empirically show that we can build better EMP models by throwing the powerful attention mechanism at a fundamentally sequence-based modeling problem, which had never been done. It is only after running our experiments and taking a step back to look at the results that we understand what Widmer means when he refers to the *essence* of music.

Our experience in developing this project convinces us that music's computational study is an inherently difficult problem, not because of the limits of computation but because of our current limited understanding of musical phenomena. It is relatively simple to experience music personally and share that experience with others. However, our (and Widmer's) conjecture is that there is a fundamental disconnect between our understanding of music's phenomenological aspect and the actual statistical patterns of nature that make music so appealing. MIR research attempts to encode these statistical patterns in computation, and as such, deriving results that are meaningful in practical application with real human interaction is non-trivial. Widmer suggests that we focus on understanding the relationship between music in nature and music as it is perceived, and we echo that sentiment here.

A deeper understanding of music may be an essential component needed to derive better evaluation systems for performance, which is a driving factor in the future development of performance generation models. To create a proper evaluation system or metric, we first need to understand *what exactly it is that constitutes a "good" or "bad" musical performance to a human listener*[2]. The answer to this question needs to come from a study in musicology, as well as

[1] *Con Espression* is the Italian phrase for "with feeling" and is used as a direction in musical notation

[2] This is a separate question from determining the quality of a musical composition, or even of the synthesis of performance (whether it is the form of an acoustic instrument or a digital synthesizer). It is not clear where to draw the exact line between composition, performance, and synthesis from the human

computer science or artificial intelligence.

We want to adopt Widmer's philosophy to guide our future explorations by drawing from the fields of musicology, music psychology, music cognition, and further advancement in computer science and mathematics to come closer to discovering this essence. From this work, we have learned that using the attention mechanism and Transformer models in EMP generation creates much more dynamic performances, both "good" and "bad," than recurrence models. The *essence* of this finding could be that using the long-term memory that attention provides allows more creative freedom in the performance process. The shorter-term memory of an LSTM may constrain the models to rely on global score features such as the overall tempo. It also may be the case that having too much creative freedom without respect for global conditioning breaks the inherent musical boundaries defined by cognitive perception. We found this to be the case with our Transformer models, which were so fast that they were "unlistenable."

Further experiments with modeling, data gathering, feature extraction, and evaluation, as well as an exploration of what determines a quality musical experience from a musicological and human cognitive perspective, will help answer these questions. We hope that we can

listener's perspective. Although we conceptualize them as independent from each other (which simplifies our mathematical assumptions), as shown in Figure 2.1, they may be entirely dependent or even the same phenomena expressed differently in nature. For example, a musician performing a Jazz improvisation on a guitar may use the physical process of synthesizing sound, such as how he strikes or bends a guitar string to reach a particular note, as driving factors in the musical piece. In this case, the guitar instrument's physical construction enables him to create musical subtleties in both the spontaneous composition and improvisation performance. Is there a clear cut line between what constitutes composition, performance, and synthesis in such a case?

On the other hand, we can analyze over 1000 symbolic musical compositions by Johann Sebastian Bach and their many musical adaptations over the last several hundred years. One example is the well-known adaptation of his Prelude No. 1 in C Major (BWV 846) as the accompaniment to a melody composed by Charles Gounod and set to the famous Latin prayer's lyrics, *Ave Maria*. The original arrangement, published in 1853, was for violin (or cello) with the piano. It has since been arranged and performed countless times for different instruments, including guitar, string quartet, piano solo, solo vocal, and full choir. We can view the original piano Prelude as its own composition and separate from the many different adaptations in design, performance, and synthesis that complete the full musical experience. This ambiguity is one example of the considerations that need to be taken into account when creating the musical definitions necessary to build computational models

continue to explore these problems using additional views drawing from musical research to come closer to finding the *essence* of EMP performance and music as a whole.

Bibliography

[1] 2002 international piano-e-competition. URL https://www.yamahaden.com/midi-files/item/2002-international-piano-e-competition.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

[3] Jean-Pierre Briot, Gaetan Hadjeres, and Francois-David Pachet. Deep learning techniques for music generation–a survey. *arXiv preprint arXiv:1709.01620*, 2017.

[4] Tom B Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.

[5] Carlos E Cancino-Chacón, Maarten Grachten, Werner Goebl, and Gerhard Widmer. Computational models of expressive music performance: A comprehensive and critical review. *Frontiers in Digital Humanities*, 5:25, 2018.

[6] Sneha Chaudhari, Gungor Polatkan, Rohan Ramanath, and Varun Mithal. An attentive survey of attention models. *arXiv preprint arXiv:1904.02874*, 2019.

[7] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*, 2014.

[8] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-

training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[9] Prafulla Dhariwal, Heewoo Jun, Christine Payne, Jong Wook Kim, Alec Radford, and Ilya Sutskever. Jukebox: A generative model for music. *arXiv preprint arXiv:2005.00341*, 2020.

[10] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

[11] Carlos Eduardo. *Computational modeling of expressive music performance with linear and non-linear basis function models*. PhD thesis, JOHANNES KEPLER UNIVERSITY LINZ, 2018.

[12] Jesse Engel, Cinjon Resnick, Adam Roberts, Sander Dieleman, Mohammad Norouzi, Douglas Eck, and Karen Simonyan. Neural audio synthesis of musical notes with wavenet autoencoders. In *International Conference on Machine Learning*, pages 1068–1077. PMLR, 2017.

[13] Sebastian Flossmann, Werner Goebl, and Gerhard Widmer. The magaloff corpus: An empirical error study. *Proceedings of the 11th ICMPC. Seattle, Washington, USA*, 2010.

[14] Francesco Foscarin, Andrew Mcleod, Philippe Rigaux, Florent Jacquemard, and Masahiko Sakai. Asap: a dataset of aligned scores and performances for piano transcription. In *ISMIR 2020-21st International Society for Music Information Retrieval*, 2020.

[15] Anders Friberg, Roberto Bresin, and Johan Sundberg. Overview of the kth rule system for musical performance. *Advances in Cognitive Psychology*, 2(2-3):145–161, 2006.

[16] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016.

[17] Maarten Grachten and Gerhard Widmer. Linear basis models for prediction and analysis of musical expression. *Journal of New Music Research*, 41(4):311–322, 2012.

[18] Maarten Grachten et al. *Expressivity-aware tempo transformations of music performances using case based reasoning*. Universitat Pompeu Fabra, 2006.

[19] Curtis Hawthorne, Erich Elsen, Jialin Song, Adam Roberts, Ian Simon, Colin Raffel, Jesse Engel, Sageev Oore, and Douglas Eck. Onsets and frames: Dual-objective piano transcription. *arXiv preprint arXiv:1710.11153*, 2017.

[20] 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. *arXiv preprint arXiv:1810.12247*, 2018.

[21] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew M Dai, Matthew D Hoffman, Monica Dinculescu, and Douglas Eck. Music transformer. *arXiv preprint arXiv:1809.04281*, 2018.

[22] Dasaem Jeong, Taegyun Kwon, and Juhan Nam. Virtuosonet: A hierarchical attention rnn for generating expressive piano performance from music score. In *NeurIPS 2018 Workshop on Machine Learning for Creativity and Design*, 2018.

[23] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, Kyogu Lee, and Juhan Nam. Virtuosonet: A hierarchical rnn-based system for modeling expressive piano performance. In *ISMIR*, pages 908–915, 2019.

[24] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Graph neural network for music score data and modeling expressive piano performance. In *International Conference on Machine Learning*, pages 3060–3070, 2019.

[25] Dasaem Jeong, Taegyun Kwon, Yoojin Kim, and Juhan Nam. Score and performance features for rendering expressive music performances. In *Proc. of Music Encoding Conf*, 2019.

[26] Shulei Ji, Jing Luo, and Xinyu Yang. A comprehensive survey on deep music generation: Multi-level representations, algorithms, evaluations, and future directions. *arXiv preprint arXiv:2011.06801*, 2020.

[27] Haruhiro Katayose, Mitsuyo Hashida, Giovanni De Poli, and Keiji Hirata. On evaluating systems for generating expressive music performance: the rencon experience. *Journal of New Music Research*, 41(4):299–310, 2012.

[28] Eita Nakamura, Kazuyoshi Yoshii, and Haruhiro Katayose. Performance error detection and post-processing for fast and accurate symbolic music alignment. In *ISMIR*, pages 347–353, 2017.

[29] neptune.ai. Neptune: experiment management and collaboration tool, 2020. URL https://neptune.ai.

[30] 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. *arXiv preprint arXiv:1609.03499*, 2016.

[31] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance. *Neural Computing and Applications*, 32(4):955–967, 2020.

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[33] Christine McLeavey Payne. Musenet, Nov 2020. URL https://openai.com/blog/musenet/.

[34] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[35] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019.

[36] Orjan Sandred, Mikael Laurson, and Mika Kuuskankare. Revisiting the illiac suite–a rule-based approach to stochastic processes. *Sonic Ideas/Ideas Sonicas*, 2:42–46, 2009.

[37] Emery Schubert, Sergio Canazza, Giovanni De Poli, and Antonio Rodà. Algorithms can mimic human piano performance: the deep blues of music. *Journal of New Music Research*, 46(2):175–186, 2017.

[38] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30:5998–6008, 2017.

[39] Gerhard Widmer. Machine discoveries: A few simple, robust local expression principles. *Journal of New Music Research*, 31(1):37–50, 2002.

[40] Gerhard Widmer. Getting closer to the essence of music: The con espressione manifesto. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 8(2):1–13, 2016.

[41] Gerhard Widmer, Sebastian Flossmann, and Maarten Grachten. Yqx plays chopin. *AI magazine*, 30(3):35–35, 2009.