UNIVERSITY OF CALIFORNIA SAN DIEGO

Application of Attention Mechanism on Multi-Instrumental Music Generation

A Thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Yiting Li

Committee in charge:

2021

University of California San Diego

2021

DEDICATION

I dedicate this thesis to my parents who never stopped supporting me with their love and affection during my lowest times. Without their support I could never have overcome such hardship in life.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to acknowledge Professor Garrison W. Cottrell for his immense help and support during my study. Without his mentorship, I could not have finished my thesis.

Chapter 1, in full, is accepted to the 20th Conference of the International Society for Music Information Retrieval 2019. Donahue, Chris; Mao, Huanru Henry; Li, Yiting Ethan; Cottrell, Garrison W.; McAuley, Julian, International Society for Music Information Retrieval Conference, 2019. The thesis author was one of the primary authors of this paper.

ABSTRACT OF THE THESIS

Application of Attention Mechanism on Multi-Instrumental Music Generation

by

Yiting Li
Master of Science in Computer Science

University of California San Diego, 2021

Professor Garrison W. Cottrell, Chair

This thesis focuses on the domain of multi-instrumental music generation using deep learning. In particular, this thesis focuses on the application of the attention-based Transformer neural network architecture in the aforementioned domain, along with exploring different music representations, and the application of transfer learning in the context of music generation.

Chapter 1

1.1 Introduction

This chapter discusses extending recent results for natural language processing [DYY$^+$19] to the domain of symbolic multi-instrumental music generation. Based on previous work in the attention mechanism [VSP$^+$17], the Transformer-XL model is an advancement in addressing the model's ability to learn long-term contexts from training sequences, thus resulting in improved ability to predict results in the domain of human languages. We explore the possibility of applying this model to the context of multi-instrumental music generation.

Music generation models have struggled to capture two basic elements of musical form: long-term structure and repetition. In their work on Music Transformer, Huang et al. [HVU$^+$19] demonstrated that powerful neural network language models, i.e., models which assign likelihoods to sequences of discrete tokens, could be used to model—and subsequently generate—classical piano music with long-term structure. To adapt this method to our multi-instrumental setting, we encode intrumental specific information directly into our language-like symbolic music representation. However, given the scarcity of large multi-instrumental music dataset, this method alone may not be sufficient for generating high-quality multi-instrumental music, since the results of Music Transformer also depended on large quantities of symbolic piano music.

In order to address the concern on data availability, we focus on the Nintendo Entertainment System Music Database (NES-MDB) [DMM18], an unusually large multi-instrumental music dataset, which contains 46 hours of chiptune music, music written for the Nintendo Entertainment System sound chip. In addition to its large size, this dataset is also appealing to music generation tasks for its homogeneity, since all of its music are composed for the exact same four-instrument ensemble. It is, however, still significantly smaller than the MAESTRO dataset [HSR+19] the Music Transformer is trained on.

The largest available multi-instrumental music dataset is the Lakh MIDI dataset [Raf16], which contains over 9000 hours of multi-instrumental music. This dataset is structrually heterogeneous, however, as instrumentation varies depending on pieces. However, intuition suggests that we might be able to benefit from the musical knowledge ingrained in this dataset to improve our performance on chiptune generation. Accordingly, we propose a procedure to map the Lakh dataset ensemble to the NES ensemble used in the NES-MDB. We then pre-train our model on the modified Lakh dataset before fine-tuning on the NES-MDB. Such transfer learning methods are common practice in other domains of deep learning tasks, and more importantly in natural language processing [DCLT18, RWC+19].

We refer to the generative model pre-trained on the Lakh dataset and fine-tuned on the NES-MDB as LakhNES. In addition to strong quantitative performance, we also conduct user studies that shows its strong qualitative performance compared to other models for music generation. LakhNES is capable of generating chiptunes from scratch, continuing human-composed material, and producing melodic material corresponding to human-specified rhythms.

1.2 Related Work

Music generation has been an active area of research for decades. Most early work involved manually encoding musical rules into generative systems or rearranging fragments of human-composed music; see [Nie09] for an extensive overview. Recent research has favored machine learning systems which automatically extract patterns from corpora of human-composed music.

Many early machine learning-based systems focused on modeling simple monophonic melodies, i.e., music where only one note can be sounding at any given point in time [Tod89, Moz94, ES02]. More recently, research has focused on polyphonic music as a piano roll—a sparse binary matrix of time and pitch—and seeks to generate sequences of individual piano roll timesteps [BLBV12, Joh17] or chunks of timesteps [YCY17]. Other work favors an event-based representation of music, where the music is flattened into a list of musically-salient events [SO17, MSC18, HVU+19]. None of these methods allow for the generation of multi-instrumental music.

Other research focuses on the multi-instrumental setting and seeks to provide systems which can harmonize with human-composed material [AW05, HCR+17, HP17, YLVD18]. Unlike the system we develop here, these approaches all require complex inference procedures to generate music without human input. Recent work [DY18] attempts multi-instrumental music generation from scratch, but these methods are limited to generating fixed lengths, unlike our method which can generate arbitrarily-long sequences. There is also music generation research that operates on the audio domain [AM18, DMP18, DvdOS18], though this work is largely unrelated to symbolic domain methods.

1.3 Dataset

The NES-MDB dataset is a multi-instrumental chiptune music dataset consisting of synthesized music taken from the Nintendo Entertainment System (NES) in MIDI format [DMM18]. Each MIDI file contains four instrumental parts with dynamics information. There are $5,278$ songs by 296 different composers, with a total note count of $2,325,636$ and total length of 46.1 hours. The advantage of this dataset for the purpose of training multi-instrumental music composition lies within its large corpus size, a uniform instrumental ensemble that they are composed for, and its symbolic representation in MIDI format.

1.3.1 MIDI format

All chiptunes in NES-MDB are stored as a MIDI file. A MIDI file describes a piece of music by storing events such as "note on" and "note off" for a specific

Table 1.1: Schematic for our event-based representation of NES-MDB, reminiscent of the one used in *Performance RNN* [SO17]. The 631 events in our representation are distributed among time-shift (ΔT) events (which allow for nuanced timing), and note off/on events for individual instruments (as in typical MIDI).

Event description	Event ID(s)
Start or end of sequence	0
ΔT for 1–100 ticks (short)	1–100
ΔT for 100–1000 ticks (medium)	101–190
ΔT for > 10000 ticks (long)	191–370
P1 Note Off/On	371–447
P2 Note Off/On	448–524
TR Note Off/On	525–613
NO Note Off/On	614–630

note, change timbre in a specific instrument, change volume (or "velocity" in MIDI terminology), or skip in time by a certain amount.

1.3.2 NES-MDB Ensemble

The NES music ensemble consists of four monophonic instruments: two pulse waveform generators (P1 and P2), one triangle waveform generator (TR), and one noise generator (NO). The pulse and triangle waveform generators are melodic voices, and the noise generator is typically used as percussion. The noise channel is capable of producing 16 distinct sounds, and each voice channel has a range of different dynamics and timbre. We choose to leave these performance-related attributes out to better focus on modeling composition only, since expressiveness can be estimated separately from the score as proven in [DMM18].

1.4 Methodology

We encode music in the NES-MDB in an event-based representation. In order to model these event sequences, we adopt a language modeling factorization. We factorize the joint probability of a musical sequence consisting of N events

$(E_1, \ldots, E_N)$ into a product of conditionals:

$$P(E_1) \cdot P(E_2 \mid E_1) \cdot \ldots \cdot P(E_N \mid E_1, \ldots, E_{N-1}). \tag{1.1}$$

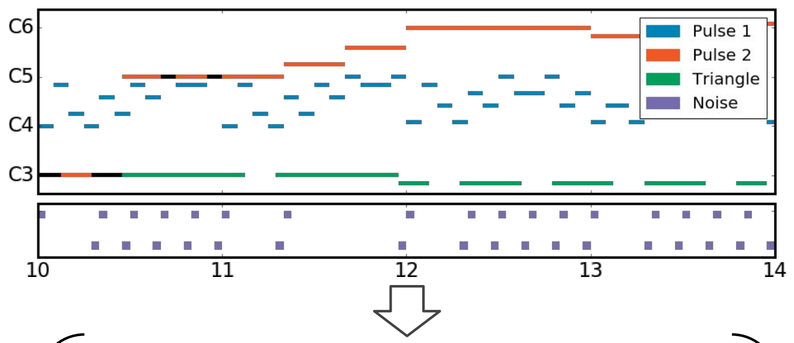
This factorization is convenient because it allows for a simple left-to-right algorithm for generating music: sampling from the distribution estimated by the model at each timestep (conditioned on previous outputs). The goal of our optimization procedure is to find a model configuration which maximizes the likelihood of the real event sequences.

1.4.1 Event-based Data Representation

In the original work on NES-MDB [DMM18], a piano-roll representation of the data was used, where musical notes are represented along a discretized time axis (top of Figure 1.1). The authors used a discretization sample rate of 24 timesteps per second in order for the musical representation to be sufficiently accurate. However, this method leads to significant information redundancy depending on sample rate, since each musical note is represented across multiple time steps. This method thus presents a challenge for learning long-term dependencies within the musical context. To address this issue, we use an event-based representation (bottom of Figure 1.1).

By encoding the files from NES-MDB dataset into text strings, we turned the said problem into a language modeling problem. The events in a MIDI file are turned into unique "characters" or "words" in the newly-defined language. We refer to these as "tokens". Time is represented implicitly, where all events are sorted in chronological order, and a "wait" event is used to forward time to the next time stamp. In reality, this wait command may be represented by numerous wait tokens, and is only approximated. This is because the amount of time that can be specified in a MIDI file is continuous, whereas we have to use a discrete set of wait tokens to represent them.

There is a trade off between sequence length and vocabulary size when encoding MIDI files in this way. We can choose to encode notes of different pitches and volumes by encoding all possible combinations as unique tokens, or to encode these



NO_NOTEON_13, DT_732, NO_NOTEOFF, DT_2197,
P1_NOTEON_76, DT_2, P1_NOTEON_70, DT_1463,
P2_NOTEON_87, DT_1, P2_NOTEOFF, DT_148,
...
P1_NOTEON_61, DT_41, P2_NOTEON_85, DT_26,
TR_NOTEON_39, DT_2, TR_NOTEON_43, DT_23,
NO_NOTEON_9, DT_646, NO_NOTEON_13, DT_733

Figure 1.1: Comparison between the piano roll representation (top) and our event-based representation (bottom). In the piano roll representation, most musical information is repeated across multiple timesteps. In the event-based representation, however, each symbol represents a musically meaningful event.

as tokens independent specific notes. The former method drastically increases vocabulary size, but the latter has a longer sequence length. Since it is generally better to have a larger vocabulary size than to have longer sequence lengths, it may be a reasonable trade off in problems like this. However, in this specific case, we choose to encode timbre and volume as separate tokens, mainly because volume does not only change at note on events, resulting in the large-vocabulary encoding not very effective in terms of decreasing sequence length.

1.4.2 Transformer Architecture

The Transformer [VSP$^+$17] is an attention-based neural network model. It utilizes a weighted attention mechanism to learn weighing on subsets of a given context, on which its prediction is conditioned, thus generating more accurate predictions. The original Transformer is an encoder-decoder architecture, and is fed with fixed-sized segment of the sequential data it is trained on. Its ability to learn long-term dependencies is thus greatly limited by the segment length it is fed with, even though this model does come with the advantage of easier parallelization.

Our work uses the Transformer-XL [DYY$^+$19], a recurrent model based on the original Transformer. Transformer-XL is capable of capturing information that came before the current training segment by caching the hidden states of the past training segment, and concatenating them to the current training segment, thus being able to learn as if it is contextualized with a longer training segment.

1.4.3 Pre-training

One of the greatest obstacles in the music domain of deep learning is the lack of large and homogeneous dataset. For example, the Bach chorale dataset [AW05] have four voices in each chrale throughout the entire dataset, but is very smallwith only 306 chrales in total. In contrast, the Lakh MIDI dataset [Raf16] consists of 175k songs, but is structurally heterogeneous, since each song has a different instrumentation and even different number of voices. The NES-MDB is a middle ground, which is structurally homogeneous, but is also fairly large. However, it is still dwarfed by larger datasets like the Lakh dataset (46 hours vs. over 9000 hours).

In order to potentially improve the performance of our model, we propose a two-step process to pre-train it on the larger Lakh dataset. First, we map each Lakh MIDI file into a structurally homogeneous one that can be performed by the NES-MDB synthesizer. Then, we pre-train a Transformer-XL model on this modified Lakh dataset before fine tuning it on the NES-MDB. Such transfer learning method is common in other research domains such as computer vision and natural language processing. However, it remains mostly unexplored in the domain of music generation, possibly due to the difficulty in mapping instrumentations between different datasets.

To map a Lakh MIDI file to one that is playable by the NES ensemble, we choose all monophonic melodic instruments, filter out those that fall outside of the NES ensemble range, and randomly assign them with one of the three melodic instruments from the NES ensemble (P1/P2/TR). Each percussion instrument is

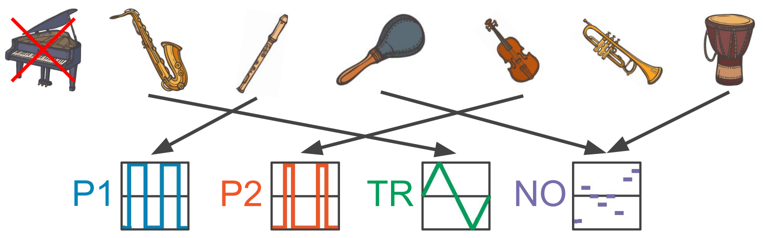


Figure 1.2: Schematic of the mapping from the NES-MDB to the Lakh MIDI dataset. We first identify monophonic melodic instruments in a Lakh MIDI file, then randomly assign to P1, P2 or TR from the NES ensemble. Percussion instruments are mapped to NO. Instruments out side of the NES ensemble range are excluded.

then randomly assigned to one of the 16 noise types availble in the NES ensemble. Since there are often multiple instruments in each Lakh MIDI file, we can generate more than one combination of instrumentation mapping from a single Lakh MIDI file. From the 175k MIDI files in Lakh, we are able to produce 775k examples that can be performed by the NES ensemble.

1.4.4 Data augmentation

Data augmentation methods standard in the music domain are also applied in order to improve the performance of our model.

- Transpose melodic voices by a random number of semitones between -6 and 5 (inclusive).
- Adjust the speed of piece by a random factor between ±5%.

In addition, we devise data augmentation methods specifically for the multi-instrumental settings.

- Half of the time, remove a random number of instruments from the ensemble (with at least one left).
- Half of the time, shuffle instrumentation of a piece among the three melodic instruments (e.g. use P2 to perform P1's part).

1.4.5 Generation Procedure

Since at each time step, Transformer-XL takes in an entire sub-sequence as input and outputs a sequence of the same length, it does not intrinsically support generation a token at a time. Furthermore, the model in the original Transformer-XL paper is not trained with fully-zero-padded sequence, which means that the first sub-sequence fed into the model always contain some information about the training sample. This creates difficulties for sampling unconditionally from the learned distribution, in which case we want to model to generate outputs without any "prompt".

We address these two problems by slightly differing the generation procedure from training and evaluation. Take the sub-sequence fed into the model as a sliding window on the entire sequence. Instead of moving the window with a stride of the window size, we move the window by only one token at a time. In fact, the model allows sub-sequences of arbitrary size to be fed in. This leads to the solution where we only provide a single token to the model, for it to output only one token at each iteration, while keeping the cache from previous iterations to preserve context during generation.

We show in Section 1.5 that the difference between training method and generation method does not affect sampling correctness, as the evaluation results produced by both methods are the same.

1.5 Experiments

We first conduct an experiment to train Transformer-XL [DYY$^+$19] on our event representation of NES-MDB. We train the model on excerpts from the training data of 512 events; each excerpt represents around 9 seconds of music on average. Because of the recurrent attention mechanism in Transformer-XL, the model effectively has access to twice this length in its history.

We use the smaller configuration of Transformer-XL which has 12 attention layers each with 8 heads (full details of our model can be found in our code which will be released upon publication). The learning rate 2e$-$4 used to train this model on text was found to be too high for our musical application, so we lowered it to 2e−5. Training was stopped when the performance of the model on the validation data stopped improving. We trained the model using four NVIDIA Titan X GPUs with minibatches of size 30, and it reached its early stopping criteria in less than a day.

Finally, we experimented with pre-training our model on the Lakh MIDI dataset mapped to the NES ensemble. To conduct this experiment, we first split the Lakh dataset into training and validation sets. We then trained the model for a week on the training set with data augmentation and monitored performance on the validation set. Because of the extreme size of the dataset, the model only completed four epochs of training. Even after a week, the model was underfitting the training data, as validation performance was still improving. We then fine-tuned this pre-trained model on the NES-MDB training data, again performing early stopping based on the validation performance.

1.5.1 Baseline

Some baseline models are included for performance comparison on NESMDB. The two simplest models are N-gram models, namely statistical calculations of how often a sequence of length N appears in training data. In particular, we use a unigram (1-gram) model and a 5-gram model. We also include an LSTM model as baseline, which is configured to have approximately the same number of parameters as our Transformer-XL model.

1.5.2 Quantitative Analysis

We use perplexity (PPL) as a quantitative measurement for how well our model predicts unseen data distribution. Perplexity of a discrete probabilistic model is defined as $e^{-\frac{1}{N}\sum_{i=1}^{N}\log q(x_i)}$, where $q(x_i)$ is the likelihood assigned by the model q to the test event x_i. The better the model is trained, the higher $q(x_i)$ tends to be assigned, thus the lower the perplexity becomes (i.e. the model is less surprised by the outcome).

Table 1.2: Quantitative performance of various models trained on the event-based representation (631 event types) of NES-MDB. *Params* indicates the number of parameters of each model. *Epochs* is the number of data epochs the model observed before early stopping based on the validation data. *Test PPL* represents the perplexity of the model on the test data, i.e., the exponentiation of its average negative log-likelihood on the test data. A *lower* perplexity indicates that the model *better* fits this unseen data.

Model	Params	Epochs	Test PPL
Random	0	0	631.00
Unigram	631	1	198.14
5-gram	9M	1	37.25
LSTM [HS97]	40M	18	14.11
+Data augmentation		35	12.64
Transformer-XL [DYY+19]	41M	76	3.50
+Data augmentation		350	2.74
+Pre-train (LakhNES)		250	**2.46**

The Transformer-XL model drastically outperforms the baseline models on the unmodified NES-MDB (most noticeably LSTM's PPL of 14.11 compared to Transformer-XL's PPL of 3.50). With data augmentation, Transformer-XL improves its performance from PPL of 3.50 to 2.74. The LakhNES Transformer model (pre-trained with the Lakh dataset) further improves its PPL to 2.46. The detailed results are reported in Table 1.2.

We also conduct experiments on the effects of different levels of pre-training on Lakh dataset before fine-tuning on the NES-MDB. We do so by monitoring performance on our model after 1, 2 and 4 epochs of pre-training on Lakh dataset. We show the results as a plot of test PPL against the number of pre-trainig epochs in Figure 1.3. The result agrees with our assumption that pre-training on the Lakh MIDI dataset helps improve PPL performance of our Transformer model.

1.6 User Studies

Although perplexity is a good metric that gives statistical insights to model performance, it does not directly imply human perception. In order to evaluate

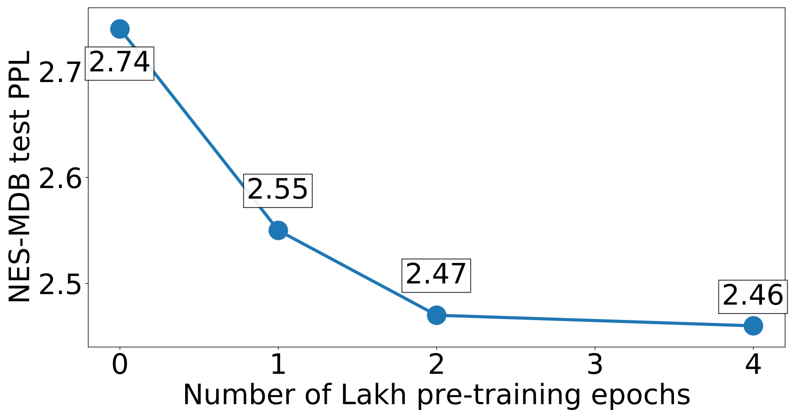


Figure 1.3: Performance, shown in PPL, of the same model pre-trained on the Lakh dataset with 0, 1, 2 and 4 epochs, respectively. This result shows that pre-training on Lakh dataset before fine-tuning on NES-MDB improves PPL performance, though with diminishing returns.

how convincing the generated music is to human, we conducted two sets of user studies on Amazon Mechanical Turk. We compare four models in both of our user studies: 5-gram model, LSTM with data augmentation, TransformerXL with data augmentation, and TransformerXL with data augmentation and pre-training on Lakh Dataset.

1.6.1 Turing Test

This user study aims to determine how well can human distinguish a human-composed piece of chiptune music from a computer-generated one, thus indicating how well the model learns to mimic human composition. The testee is presented with 2 pieces of chiptune music and is informed that one of them is human-composed while the other is computer-generated. The testee is then asked to identify the human-composed piece. Given the noisy nature of Amazon Mechanical Turk studies, we included a control group made of random noises, which are generated by selecting events uniformly at random. These samples are then presented as computer-generated tunes.

Each human judge is asked to work on 800 batches of 10 randomly ordered pairs of tunes, 2 of which have the “fake” data coming from random noises, while the other 8 have the “fake” data coming from the other 4 models we picked. The

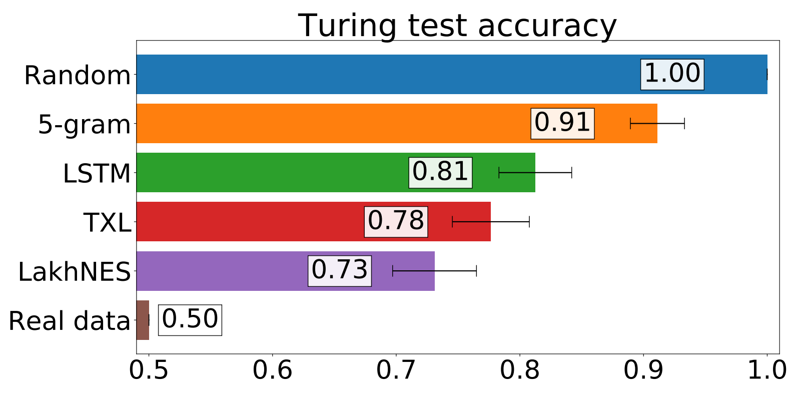


Figure 1.4: This figure shows human accuracy at identifying computer-generated music from human-composed music. Error bar shows the standard error. Each testee is presented with a pair of pieces, one human-composed, one computer-generated (as listed in the figure), and is asked to identify which one is human-composed.

testee is required to answer at least 3 batches and achieve 100% accurasy on all 6 control sets from these batches in order for their answer to be included in our result. Random answers thus have only 1.6% probability to be included.

In this study, lower accuracy indicates that the model learns human composition better, and can therefore "fool" human judges more often. We find that LakhNes is more often misidentified as human than all of the other models (LSTM, Transformer without pre-training, and 5-gram), although the difference pre-training brings is not statistically significant ($p = .32$).

The results suggest that there is still a significant gap between computer-generated and human-composed music.

1.6.2 Preference Test

Since human-ness does not directly reflect human preferences, we also conduct a preference-based user study, where human judges are presented with pairs of chiptune sample selected randomly from the four aforementioned groups, in addition to a control group consisting of random noise. Similar to the Turing Test method, these pairs of chiptunes come in batches, and in order for an answer to be included in the results, the testee must not choose to prefer any random noise

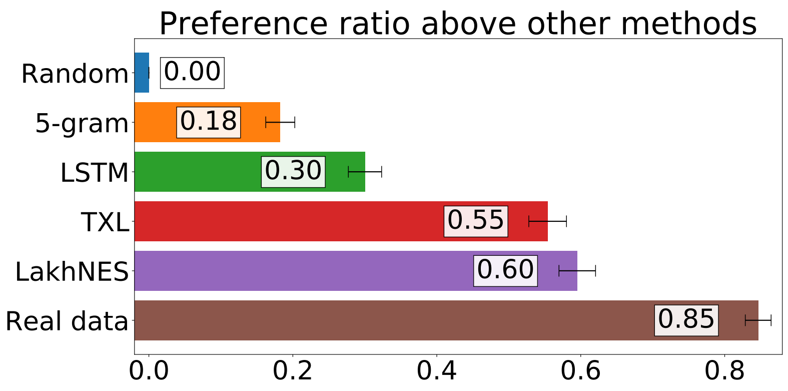


Figure 1.5: This figure shows the percentage of human that preferred the model as listed in figure compared to another randomly chosen model. Error bar shows the standard error. Each testee is presented with a pair of pieces generated by randomly selected models, and is asked which they preferred. Higher score indicates that human prefer music generated by this model more often than others.

in any of the pairs.

1.7 Conclusion

In this study, we show that Transformer models can be used to learn multi-instrumental chiptune music, and can arguably be generalized to any domain of multi-instrumental music with a similar structural complexity and number of instruments, given the voice channels used here can be replaced by any arbitrary instruments. With our proposed method to pre-train on cross-domain datasets, it is also shown that such a task can be further improved by pre-training on a larger dataset. We also developed an event-based representation suitable for multi-instrumental music.

1.8 Contribution

The author of this thesis is one of the primary authors of this work. The author contributed to methodological exploration in the early stages of this work, implementing and training models including the experiments that involves training the Transformer-XL model on the unmodified NES-MDB dataset.

1.9 Acknowledgements

Chapter 1, in full, is accepted to the 20th Conference of the International Society for Music Information Retrieval 2019. Donahue, Chris; Mao, Huanru Henry; Li, Yiting Ethan; Cottrell, Garrison W.; McAuley, Julian, International Society for Music Information Retrieval Conference, 2019. The thesis author was one of the primary authors of this paper.

Bibliography

[AM18] Mathieu Andreux and Stephane Mallat. Music generation and transformation with moment matching-scattering inverse networks. In Proc. *ISMIR*, 2018.

[AW05] Moray Allan and Christopher Williams. Harmonising chorales by probabilistic inference. In Proc. *NIPS*, 2005.

[BLBV12] Nicolas Boulanger-Lewandowski, Yoshua Bengio, and Pascal Vincent. Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. In Proc. *ICML*, 2012.

[DCLT18] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. *arXiv:1810.04805*, 2018.

[DMM18] Chris Donahue, Huanru Henry Mao, and Julian McAuley. The NES Music Database: A multi-instrumental dataset with expressive performance attributes. In Proc. *ISMIR*, 2018.

[DMP18] Chris Donahue, Julian McAuley, and Miller Puckette. Adversarial audio synthesis. In Proc. *ICLR*, 2018.

[DvdOS18] Sander Dieleman, Aäron van den Oord, and Karen Simonyan. The challenge of realistic music generation: modelling raw audio at scale. In Proc. *NeurIPS*, 2018.

[DY18] Hao-Wen Dong and Yi-Hsuan Yang. Convolutional generative adversarial networks with binary neurons for polyphonic music generation. In Proc. *ISMIR*, 2018.

[DYY$^+$19] Zihang Dai, Zhilin Yang, Yiming Yang, William W Cohen, Jaime Carbonell, Quoc V Le, and Ruslan Salakhutdinov. Transformer-XL: Attentive language models beyond a fixed-length context. *arXiv:1901.02860*, 2019.

[ES02] Douglas Eck and Jürgen Schmidhuber. Finding temporal structure in music: Blues improvisation with LSTM recurrent networks. In Proc. *Neural Networks for Signal Processing*, 2002.

[HCR$^+$17] Cheng-Zhi Anna Huang, Tim Cooijmans, Adam Roberts, Aaron Courville, and Douglas Eck. Counterpoint by convolution. In Proc. *ISMIR*, 2017.

[HP17] Gaëtan Hadjeres and François Pachet. DeepBach: A steerable model for Bach chorales generation. In Proc. *ICML*, 2017.

[HS97] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

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

[HVU$^+$19] Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Ian Simon, Curtis Hawthorne, Noam Shazeer, Andrew M. Dai, Matthew D. Hoffman, Monica Dinculescu, and Douglas Eck. Music Transformer: Generating music with long-term structure. In Proc. *ICLR*, 2019.

[Joh17] Daniel D Johnson. Generating polyphonic music using tied parallel networks. In Proc. *International Conference on Evolutionary and Biologically Inspired Music and Art*, 2017.

[Moz94] Michael C Mozer. Neural network music composition by prediction: Exploring the benefits of psychoacoustic constraints and multi-scale processing. *Connection Science*, 1994.

[MSC18] Huanru Henry Mao, Taylor Shin, and Garrison Cottrell. DeepJ: Style-specific music generation. In Proc. *International Conference on Semantic Computing*, 2018.

[Nie09] Gerhard Nierhaus. *Algorithmic composition: paradigms of automated music generation.* Springer Science & Business Media, 2009.

[Raf16] Colin Raffel. *Learning-based methods for comparing sequences, with applications to audio-to-midi alignment and matching.* PhD thesis, Columbia University, 2016.

[RWC$^+$19] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. Technical report, 2019.

[SO17] Ian Simon and Sageev Oore. Performance RNN: Generating music with expressive timing and dynamics. https://magenta.tensorflow.org/performance-rnn, 2017.

[Tod89] Peter M Todd. A connectionist approach to algorithmic composition. *Computer Music Journal*, 1989.

[VSP$^+$17] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Proc. *NIPS*, 2017.

[YCY17] Li-Chia Yang, Szu-Yu Chou, and Yi-Hsuan Yang. MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. In Proc. *ISMIR*, 2017.

[YLVD18] Yujia Yan, Ethan Lustig, Joseph VanderStel, and Zhiyao Duan. Part-invariant model for music generation and harmonization. In Proc. *ISMIR*, 2018.