A MULTIMODAL TEXT-TO-MIDI TRANSFORMER MODEL WITH SPECIAL CONSIDERATION TO ARTIST USAGE

A THESIS IN

Electrical and Computer Engineering

Presented to the Faculty of the University
of Missouri-Kansas City in partial fulfillment of
the requirements for the degree
MASTER OF SCIENCE

by

SAMUEL ALAN MILES

Kansas City, Missouri

2023

A MULTIMODAL TEXT-TO-MIDI TRANSFORMER MODEL WITH SPECIAL CONSIDERATION TO ARTIST USAGE

Samuel Alan Miles, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2023

ABSTRACT

Generative AI systems are of great interest in the field of automated music production. Current well-known state-of-the-art music generation systems such as MusicLM, while incredibly versatile and tractable, do not allow for direct control of tone or texture of the generated instruments other than by altering the text prompt, which also alters the structure of the generated composition. Therefore, symbolic music generation would still be of great use to musicians.

Using current state-of-the-art advancements in Deep Learning and an off-the-shelf MIDI dataset and pretrained English encoder, this work proposes a novel sequence-to-sequence music generation model that converts written descriptions of the style and artistic themes of a song into coherent MIDI musical representations.

The architecture and synthetic dataset used in training this model were constructed with special consideration to the needs of musicians, namely, a native musical format, lack of association between any particular artist and musical style, and the possibility of live usage with a human accompaniment.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Science and Engineering, have examined a thesis titled "A Multimodal Text-To-MIDI Transformer Model With Special Consideration To Artist Usage" presented by Samuel A. Miles, candidate for the Master of Science degree, and certify that in their opinion it is worthy of acceptance.

Supervisory Committee

Reza Derakhshani, Ph.D. (Committee Chair)

Department of Computer Science Electrical Engineering, UMKC

Cory Beard, Ph.D.

Department of Computer Science Electrical Engineering, UMKC

Dr. Yugyung Lee, Ph.D.

Department of Computer Science Electrical Engineering, UMKC

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

ACKNOWLEDGMENTS

Special thanks to my thesis advisor Dr. Reza Derakhshani, whose support, encouragement, and instruction brought this project to fruition and success beyond what I could have imagined at the outset. Additional thanks to my advisory committee for their advice and guidance, as well as their willingness to take on the additional workload for my sake.

Special thanks to those who offered to evaluate the model, your assistance was invaluable in putting a yardstick up to the model. Thanks to the local musicians whose input when planning the model's workflow and style was invaluable, namely, Harrison Breshears, Ben Baker, David Muolo, and Seth Harper.

Special thanks to UMKC, for providing compute resources to train the model and the educational and scientific resources to train me.

Sincere thanks to my immediate family, Sam, Cheryl, Seth, Lavonne, Sophie, and Braden, whose love, friendship, and support made this project possible.

Thanks to my dear friends, my church family, and my coworkers, whose little favors and acts of kindness and encouragement were refreshing.

Thanks to the recording musicians whose art made me fall in love with music. I made this for you, I hope you think it's fun.

GPT-4 was utilized for programming assistance, but none of the body of this thesis was written by a language model except where explicitly stated.

CHAPTER 1

INTRODUCTION

Generative AI models have recently been the subject of a massive amount of social focus and scientific research. Somewhat surprisingly, one of the primary fields that Generative AI-powered automation threatens to disrupt is the arts. Stewardship of the arts moving forward is going to be a societal challenge that AI researchers should navigate trepidatiously, out of respect for artist's craft and out of empathy for the potential harm Generative models could cause to their livelihood. Therefore, in designing generative models, it is desirable that the needs of artists be considered.

Current well-known state-of-the-art music generation systems such as MusicLM[1] have made great strides into start-to-finish automated music production. MusicLM allows artists to create arbitrary audio files in a crisp 24khz product with a simple text prompt as an input. Additionally, MusicLM can be conditioned on pure audio of a melody, such as whistling or humming. These models, while incredibly versatile and tractable, have a few current limitations. Firstly, for MusicLM in particular, there is a limiting hard cap on the maximum context length, as the model cannot process more than 30 seconds of audio. Additionally MusicLM, Musicgen, and models like them do not allow for direct control of tone or texture of the generated instruments other than by altering the text prompt. However, this also alters the structure of the generated composition, and as such, audio artifacts cannot be easily removed. In contrast, symbolic music representations are easily edited or filtered, and can be reproduced however an artist chooses, with nearly infinite control over tone, instrument, and expression. Therefore, symbolic music generation would still be of great use to musicians.

The initial concept of this thesis was to investigate and develop a model that could potentially accompany a live artist, either just to jam out for fun or to come up with ideas, or potentially in experimental live performance environments. Discussion with musicians regarding what such an automated music generation system would need to do in order to be useful to them included several common themes. For one, the model would need to be able to understand the musical structure that the artist was going for in order to accompany it appropriately (tempo, chord structure, etc). Secondly, the model should be able to understand the "vibe" that the artist desires to create with their music, with information such as genre and thematic intent.

To this end, using current state-of-the-art advancements in Deep Learning and an off-the-shelf dataset and pretrained english encoder, this work proposes a novel sequence-to-sequence music generation model that converts written descriptions of the style and artistic themes of a song into coherent MIDI musical representations. The architecture and synthetic dataset used in training this model were constructed with special consideration to the needs of musicians, namely, a native musical format, lack of association between any particular artist and musical style, and the possibility of live usage with a human accompaniment.

CHAPTER 2

BACKGROUND AND RELATED WORK

Artificially constructing novel, coherent pieces of music is no simple task. While algorithms that generate sequences and melodies in a specific key have existed for a long time in the field of generative synthesis popularized by Brian Eno [2], there is no traditional algorithmic structure that can codify such specifics as musical genre or emotional tone in a way that computers can intuitively understand and interpolate between with natural language input, nor is there a convenient way to incorporate the long-term structure of the composition into the decision-making process. Therefore, an alternative computing schema such as Deep Learning is of interest for the task.

2.1 Artificial Neural Networks and Methods

Firstly, it will be helpful to frame the task specifically. Music is commonly understood to be a sequence of notes and rhythms in a coherent order. Therefore, it would be appropriate to tackle the task of generating music as a sequence of events. "Tokenization" is the process of converting an input sequence of symbols into a vector of corresponding numbers for later processing, and will be mentioned throughout this paper. This process can be executed for both text and MIDI data. Additionally, these sequences of symbols can be compressed with a process known as Byte-Pair Encoding, which allows for stand-in symbols to be utilized for the most common subsequences of bytes in a given dataset. This allows for longer sequences of events to be converted into a smaller sequence of symbols.

Next, we will take a moment to discuss the history and design of Artificial Neural Networks, the simplest useful example being the Multi-Level Perceptron. This device consists of an interconnected network of artificial neurons, with at least one hidden layer between the input and the output. The artificial neurons each consist of a set of weighted inputs that are summed, which are then used to evaluate a nonlinear function, henceforth known as the activation function. Importantly, multilayer neural networks with at least one hidden layer and a nonlinear activation function have been shown to be universal function approximators [3].

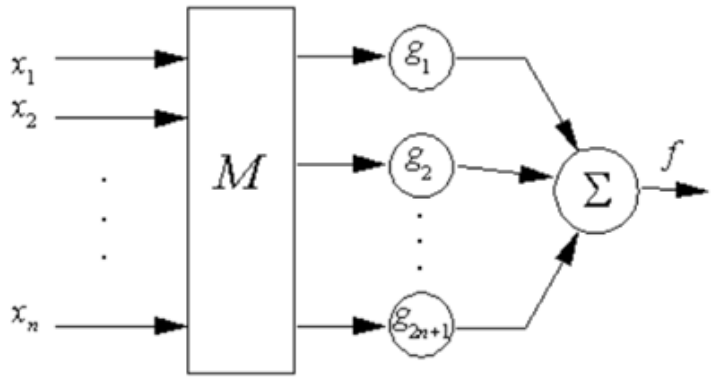

Figure 1: Diagram of an artificial neuron [3]

The most common training paradigm used with artificial neural networks is backpropagation. This method functionally implements gradient descent by evaluating an input training example, calculating the error at the output, evaluating what changes need to be made across the network by use of the chain rule, and calculating the average value of each parameter change over a small batch of examples. Modern optimization algorithms, such as AdamW, combine standard backpropagation with both momentum

and weight decay, which has been shown to increase generalization when compared to standard backpropagation [4].

2.2 Recurrent Neural Networks

Recurrent Neural Networks are an alternative to feed-forward networks such as the Multi-Level Perceptron, that possess the innate ability to process time-series data. They accomplish this by implementing feedback loops that allow the model to recall previously processed data.

The Long-Short Term Memory architecture, or LSTM[5], is a widely studied and implemented RNN architecture. This architecture consists of an arbitrary number of cells connected in series. Each cell features several gates that allow the model to selectively forget irrelevant information. Recurrent Neural Networks often suffer from what is commonly known as the vanishing gradient problem, in which gradients corresponding to weight updates regarding potentially relevant information in the past decrease to nothing during training. The various learned gates make back-propagation across time simpler, keeping relevant weight updates relevant.

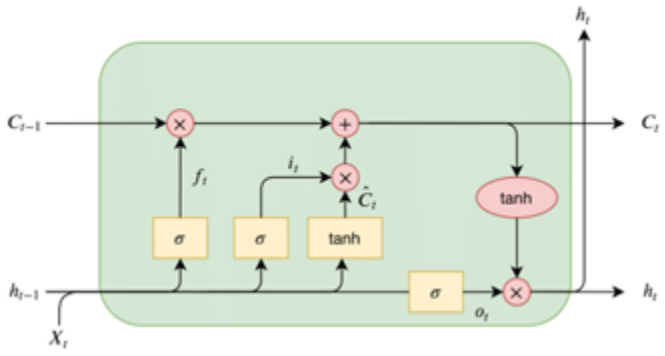

Figure 2: Diagram of an LSTM cell [6]

There is a long history of LSTMs being used to generate musical sequences, likely the first of which being used to compose blues riffs [7]. This model represented its input to the model as a series of vectors, with each vector representing whether or not a specific note was being played for each quarter note in a measure. The model was then trained to predict the next value in a given input sequence, and was able to generalize the structure of blues music and compose novel examples.

Google's Magenta Labs outlined the use of an LSTM that was able to compose music with expressive, sub-note timing, known as the Performance RNN. It used a more advanced tokenization schema than previous papers, allowing for 32 different note volumes and an arbitrary delay length at time steps as small as 1 millisecond and as long as one second [8]. The model was then trained on a custom dataset of MIDI generated by professional pianists and produced locally cohesive and expressive music.

2.3 Transformer Neural Networks

Transformer neural networks were first outlined in the paper "Attention is All You Need" and was originally formulated as a method of translating between languages [9]. It proposes a more concrete solution to the vanishing gradients problem, by eliminating recurrence and avoiding backpropagation across time while keeping all relevant time-series information available to the model at once. The traditional Transformer architecture consists of two halves: an encoder and a decoder, each consisting of nearly identical units known as Transformer blocks. The encoder converts the input sequence into a high-level neural embedding, and the decoder is responsible for generating the output sequence based on information from the encoder. To avoid recurrence while

retaining the ability to understand time-series data, both the encoder and decoder learn to process sequential data by its positional encoding, mapping each token to a position vector. The vector is determined as a function of three different sine waves at different frequencies, which keeps their values distinct from each other but predictable. This trick allows the network to rely completely on its attention mechanism to perform sequence modeling.

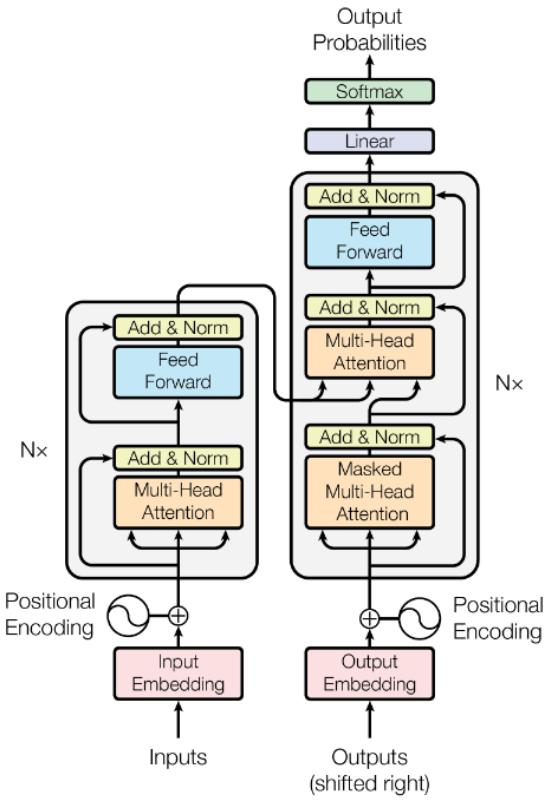

Figure 3: Diagram of the Transformer Neural Network [9]

Next, we will address the Scaled Dot-product attention mechanism of the Transformer block. A learned matrix of weights is developed, which affects the signal sent between a “query” and “key” vector, these being two positional tokens in a

sequence. This is then processed, and then compared with a "value" vector. For self-attention, attention is typically performed with the key, query, and value each being evaluated over the same sequence of tokens, allowing the model to process the relationships between them. Additionally, for complicated sequences, multiple attention heads are useful, allowing tokens to form more nuanced relationships with each other. The decoder side of the Transformer network has an optional layer in its blocks known as the cross-attention layer. The final output of the encoder layer for each input token is routed into this layer with the decoder inputs as the key and query, enabling the decoder to determine which aspects of the encoder signal are important when continuing the sequence found in the decoder. Additionally, in the decoder's self-attention layers, a causal mask is applied, which exclusively allows tokens to attend to tokens that came before them in the sequence.

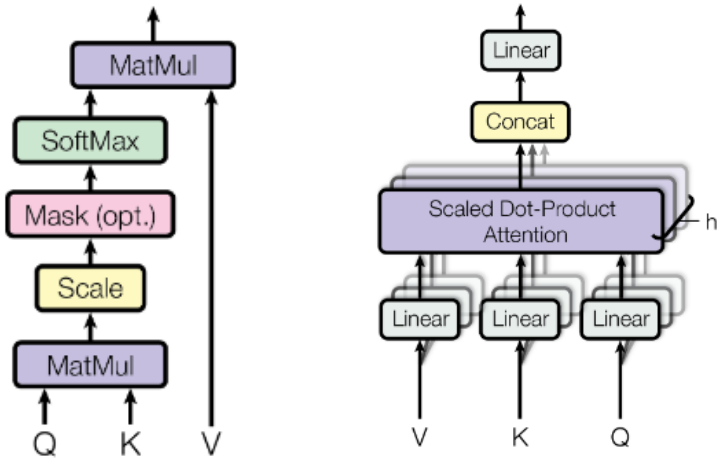

Figure 4: Diagrams of the Scaled Dot-Product Attention Mechanism (left) and Multi-Headed Attention(right) [9]

Lastly, after passing each token through an identical feed-forward neural network for processing, the output of the layer across each token is normalized, and a residual connection to the input of the previous layer is made. By directly attending to each token of the encoder and past decoder tokens, the model is able to better model long-term relationships between symbols than by simply relying on recurrence.

To generate the output sequence, first the input sequence is fed into the encoder. The encoder attends to the entire input sequence at once and passes its findings to the decoder. The decoder's output begins with a beginning-of-sequence token, such that the model is prompted to continue the sequence. For each iteration of its output, the decoder predicts the likeliest next token based on its self-attention and the cross-attention through a language modeling classification head, consisting of a feedforward network reduced to probabilities by the softmax function. This output is then fed back into the input of the decoder, and the next token is predicted again.

Google's Magenta Labs described the use of a transformer model for use in the MIDI music data generation task for the first time in 2018 [10]. They noted that Transformer models, initially designed for sequence-to-sequence tasks, could use their innate sequence comprehension abilities to model symbolic music with greater long-term coherence than that of PerformanceRNN. To model these longer sequences, the researchers elected to forego using absolute position embeddings used in previous work, in favor of relative positional embeddings. They found that this allowed the model to better grasp the relationship between notes in a chord.

OpenAI researchers in 2019 created a decoder-only music generation model based on their GPT-2 architecture called Musenet [11]. Their model featured input tokens for artist names and requested instruments. They noted that the model performed nearly identically to a model with relative embeddings when using specific chord embedding.

FIGARO is a multimodal text-to-midi model that converts written, natural language descriptions of the notes and rhythms in each measure into a MIDI file [12]. The model generates multi-instrument MIDI arrangements that closely adhere to the original intent of the written sequence. Notably, it formats this problem as a sequence-to-sequence task, similar to language translation. This model is closest to the implementation outlined in this paper, which is strikingly similar to a vanilla Transformer as outlined in Huang et al. in Attention Is All You Need [9].

It has been shown that separate pretrained Transformer encoders and decoders, possibly from different languages or modalities, can be utilized for sequence-to sequence generation tasks [13]. This method was originally conceived to create translation, summarization, and question answering models on a limited computational budget. Many of these tasks are now generally handled by billion-plus parameter bespoke language models. However, a similar process to translation using pretrained encoder and decoder models, if applied to the modalities of MIDI information and natural language prompts, held promise for the use case outlined in this thesis. With the limited number of existing text-midi pairs, a knowledge transfer mechanism seemed like a favorable avenue to pursue.

2.4 Sampling and Classifier-Free Guidance

In the next-token-prediction task, the output result is a vector of logits, which are then passed through a softmax layer to represent the probability for each token in the vocabulary that it would occur next in the sequence. The simplest decision for decoding would be to simply utilize the highest-probability token in the list each time, which is commonly known as “greedy decoding”. However, this can cause smaller transformer models to get stuck generating loops or bland and uninteresting results. To create more interesting generations and exploit the general knowledge learned by the model in its pretraining task, this vector of logits can be manipulated in a variety of ways, the most common of which being temperature sampling. This method allows each token to be considered for random selection based on its probability of occurring, and this probability distribution becomes more uniform across the vocabulary as the temperature parameter increases (ergo less common tokens are more likely to be selected than previously). However, recent research published in the paper *Stay On Topic with Classifier Free Guidance*[14] has shown that it is possible to amplify the difference between the probability distribution of the logits when prompted, and the distribution when unprompted. This is known as classifier-free guidance (CFG), and has been implemented to great success in text-to-image models like DALL-E 2. For sequence-to-sequence models, it requires that the model be trained on a small subset of unlabeled training data, which is then used to signal to the model that it should generate unconditioned logits (although this could be substituted with the outputs of another model), however the paper’s primary results showed that decoder-only models can simply be stripped of their prompt to achieve the same purpose. During inference, the model returns logits for both

prompted and unprompted inputs, and the difference is amplified per a guidance scalar γ. In this paper, this algorithm is implemented as follows, per the author's implementation in the paper:

$$P_{cfg} = 0.7[\gamma(P_{cond} - P_{ucond}) + P_{ucond}] + 0.3P_{cond} \quad (2.1)$$

where P_{cfg}, P_{ucond}, and P_{cond} are the probability distributions in logit space of the guided output, unconditioned model, and conditioned model respectively. By increasing γ above 1, the difference between the conditioned and unconditioned probabilities is amplified. The CFG probability distribution is also mixed with the standard conditioned model distribution at an arbitrary value of .7, to maintain cohesion at high CFG values.

Classifier-free guidance with this method has been shown to be effective in sequence-to-sequence music generation tasks, such as with Musicgen [15]. Musicgen's output representation is a complex sequence of interleaved tokens that represent the complete vocoded audio signal, so surely this technique would prove useful in the simpler MIDI representation.

CHAPTER 3

METHODOLOGY

3.1 Dataset Collection

Following in the footsteps of FIGARO, the REMI+ tokenizer was selected [12]. This tokenizer allows for all instruments in an arrangement to be generated in the same sequence, which avoids the complexity of training with multiple language modeling heads. REMI+ includes separate tokens to denote many of the important features of symbolic music, namely, the separation of each measure, position within each measure, the tempo at the current bar, pitch and velocity of notes, and the MIDI program of each note, among others. The tokenizer was implemented using the Miditok library, which includes many features and utilities that greatly sped the iterative process [16]. Tokenizer settings were selected such that 32 volume levels and 128 positions within a measure could be distinguished. Byte-Pair encoding was learned over a 22,234 example subset of the dataset, giving the model a final vocabulary of 1948 tokens. This was done to increase the possible sequence of MIDI events with the previously described compression methods, without increasing the memory usage of the model. This was also processed before time and instrument quantization, meaning that it could be reused down the line for more advanced models. Additionally, a bug in the MIDITOK library limited the vocabulary to 1948 tokens instead of the intended 2048.

Figure 5: Diagram of a sample REMI+ token sequence [16]

All MIDI data used to create the model tested in this thesis was sourced from the fantastic Lakh Midi dataset [17], and was quantized to one instrument ID per instrument class before being seen by the model (i.e., no distinction is made between Acoustic Grand Piano and the Clavinet, as the first 8 MIDI instruments all share the same instrument class). Any sample not found to be in 4/4 time was rejected for training purposes. As a starter point to begin pairing these MIDI files with semantically meaningful prompts, a portion of the Lakh Midi dataset is already matched to the Million Song Dataset [18], including title and artist information, as well as artist tags. Additional information about the song can be acquired from the Last.FM and MusiXMatch datasets, as well as by scraping the metadata of the MIDI itself and noting things like key signatures and tempo.

The open-source dataset used to train MusicLM, known as MusicCaps [1], while intended for a similar end goal of artist-focused natural language music generation, is not fully compatible with the symbolic music generation task. This dataset consists of high-quality text descriptions written by musical artists to describe audio recordings of music. Notably for our purposes, many of the prompts contain information about the acoustic character of certain instruments, which could be tenuously useful for symbolic music generation, and audio quality, which is irrelevant. Additionally, the duration of the labeled clips is capped at 10 seconds, so information in the dataset is highly focused to a specific musical moment. In lieu of adapting the dataset, a new one was constructed based on information available in the Lakh Midi dataset and other information extrapolated from sources that will be discussed hereafter. This gave an added advantage in terms of sample quantity, as these methods were able to yield almost 48,000

semantically meaningful and bespoke prompts, as opposed to MusicCaps' 5521 examples.

The labeling tasks required for the creation of a labeled dataset on such a massive scale were greatly accelerated by the use of Large Language Models (LLMs). Previous approaches for similar tasks in the past have used services such as MTurk, which would be cost-prohibitive and could lack expert knowledge of musical ideas. However, the advent of instruct-tuned Large Language Models such as GPT-3.5-turbo has reduced the labor required to perform such tasks exponentially, and the barrier to entry for big data tasks has been lowered substantially.

A significant amount of the artistic intent of a song is communicated in its lyrics. Lyrics often convey the emotional and narrative overtone that instrumental accompaniment is then constructed underneath, and could offer a simple method to communicate this overtone to others when discussing a song. So to perform the same task, a music generation model could greatly benefit from this lyrical information when making decisions. To provide it, a summary of the lyrics for each song that has them was provided in the training prompt. This was accomplished by first checking the MusXMatch dataset, which is a bucket-of-words lyrics dataset matched with the Million Song Dataset, to see if lyrics would likely be available. If so, these lyrics were scraped from Genius.com and then summarized with GPT-3.5-turbo.

How then should we use this information to create prompts similar to what a user might want to use to describe the song? Large Language models in recent years have proven very functional for similar synthetic data generation tasks, such as to create synthetic instructions that were used to fine-tune an Instruct2Pix Stable Diffusion

model[19]. Utilizing GPT-3.5-turbo, information about the key, tempo, instrument categories, lyrics, as well as any tags from the Million Song and Last.FM datasets was converted into prompts for use as training data with the following example prompt, which took great care to remove the specific identifying information about the song for reasons outlined in the Results and Discussion section of this thesis:

> Instruct a small language model to generate a song by writing a concise description of the musical characteristics of the song the user mentions, without mentioning it by name, as if you were inventing it from scratch. do NOT mention it by name. If the lyrics summary includes the name of the song, do not repeat it verbatim, and rephrase it instead. Example Input:
>
> Muskrat Love" <---DO NOT INCLUDE
>
> artist:"Willis Allen Ramsey<---DO NOT INCLUDE"
>
> song tags: country, singer-songwriter
>
> Lyrics summary:The song describes the joyful and carefree love between two muskrats who dance and enjoy each other's company in their own land.
>
> tempo: 89BPM
>
> key: Emaj
>
> instrument categories: guitar, bass, piano
>
> Country song by an americana singer-songwriter. Instruments are guitar, bass, and piano. Key is Emaj. Tempo is 89BPM. Folksy, optimistic song about muskrats in love.
>
> ---or---
>
> Folksy, optimistic song about muskrats in love. Country music, americana, singer-songwriter. Tempo is 89BPM. Key is E Major. Instruments are guitar, bass, and piano.

I should note that this specific example was selected due to my personal familiarity with the song, but the prompt contains many of the properties that the artists consulted found to be important, as will be discussed later in the paper. The lyrics summary was

written by GPT-3.5-turbo, to keep the example consistent with what the model might actually see.

It should be noted that despite the stated intention to avoid including the names of songs and artists in the final dataset, this information is included in both the example prompt to the LLM and the actual information presented after the example, and thus poses a risk of reappearing in the training data. This risk is worth taking, however, when one considers GPT-3.5-turbo's general knowledge of musical artists and songs. Knowing the song and the artist allows the LLM to better make intelligent decisions when describing the musical identity of a song. Additionally, due to extensive work on OpenAI's part, the model was found to be good at following instructions like this to avoid including the specific artist mentioned in the prompt when generating its output.

What about the rest of the unlabelled dataset? A slightly more complicated labeling approach becomes necessary. Many of the songs had filenames and metadata that could be utilized to link them with the title and artist that composed the song originally, which would allow for greater understanding of the context surrounding the composition. After extracting this data and prompting GPT 3.5-turbo to predict the name and artist of every midi, up to 6,000 additional examples were located that required no further human review. A rapid manual review of an additional 4000 partially labeled (artist or title only) examples was conducted to determine if they would likely be matched with a song description down the line, or if GPT-3.5 missed any obvious clues. In a small number of cases, if any information was obvious and warranted adding to the dataset, this information was injected into the lyrics summary step of the pipeline. For instance, "slowsoul.mid" was manually tagged as slow soul, as this would not likely get labeled

accurately down the line but could still serve as a decent training example with minimal intervention.

To gather general information about the artist in a way that could inform interpretations of specific songs, Wikipedia was searched for each artist's name, and a short list of tags was produced by GPT-3.5-turbo based on the introduction paragraph of the artist's Wikipedia article. This process worked reasonably well, and data about the artists mentioned in the metadata of each MIDI file was able to be utilized in further processing, similarly to how similar artist tags were utilized with the Million Song Dataset matched files.

To pair the newly gathered artist and title data with semantically meaningful connections to the music, YouTube presents itself as a salient, public source of sentiment toward musical compositions as well as an excellent search engine. This was partially inspired by MusicThe approach taken with regards to pairing the data with midi files on a massive scale proceeded with the following assumptions at hand:

1. The name of the song and artist will bring up the most relevant results to the labeling task
2. If it is deemed relevant enough by a human labeller (or if an artist's name is attached but the previous search failed a similarity check), the name of the song will yield the most relevant results to the labeling task
3. If the song is not an exact match, similarly named songs will likely carry similar overarching themes with respect to the human experience
4. YouTube video descriptions and comments contain relevant data to describe the contents of the song in question

The fundamental assumptions under which the labeling task was carried out are not reliably correct and will need to be improved upon in future works. Steps were taken to improve the accuracy of the method, namely, a check was performed using GPT 3.5-turbo to verify that the search results matched the query and that the resultant video was in fact a song. However, this approach was not bulletproof. The following discussion regards this failure case, as it is representational of many possible errors these assumptions provided:

> "Title: Graduation Day
>
> Artist: The Four Freshmen and The Beach Boys
>
> Youtube Link: tbmNuB7spmA
>
> Song Tags from Youtube: soulful, unique, autotune, interlude, hip-hop, John Legend, Kanye West, Miri Ben Ari, explicit lyrics, iconic album, graduation, high school, Class of 2020
>
> Prompt: A soulful and unique song with autotune and an interlude. It has elements of hip-hop and features brass, strings, bass, guitar, piano, and drums. The tempo is 101 and the key is C. The lyrics are explicit and speak to the experience of graduating high school, making it an iconic album for the Class of 2020. The song features notable artists such as John Legend, Kanye West, and Miri Ben Ari."

The first YouTube search for "Graduation Day The Four Freshmen and The Beach Boys" yields results by the Beach Boys and the Four Freshmen individually, but not both groups, so the title authentication step failed when the artist was included. The automated system, working off of assumption 2, searched only for the title of the song. The first YouTube search for "Graduation Day" brings up a song of the same name by Kanye West, which passed the GPT-3.5 matching step but is not relevant to the contents of the midi. Additionally, GPT-3.5 follows its prompt to the letter by not explicitly mentioning The Four Freshmen and the Beach Boys, but found it within reason to mention all the

other artists found in its tags. While these problems would likely hamper the model's potential hip-hop and experiential knowledge from this example, it was still able to learn from the deterministic tempo, key signature, and instrument data. Additionally, the experience of graduating from a secondary institution was also explicitly mentioned.

3.2 Model Architecture

The final model architecture is a sequence-to-sequence transformer, which consists of an English-language encoder that is cross-attended to by a pretrained MIDI decoder. To most efficiently capture a wide range of written experience, beginning with an encoder that is pretrained on a large English dataset seems sensible. DistilBERT [20], a 67M parameter model trained by Huggingface using knowledge distillation on Google's BERT model, was selected due to its small size with respect to its knowledge, as well as proven utility in transfer learning.

The model's decoder was a custom 12-layer 94 M parameter GPT-2 model.The model was pretrained using the Huggingface trainer [21] over the entire available midi dataset for 0.89 epochs, stopping early due to time constraints. Sequence length during training was set to 8192, which represented MIDI sequences anywhere from about 20 seconds in length up to multiple minutes, depending on the note density, tempo, and number of instruments. A log of this training is shown below. Note the spike around iteration 500: this included 17 optimization steps. This could be due to poor examples in the dataset, but its effect was quickly smoothed out and its presence has shown no effect on the quality of the model.

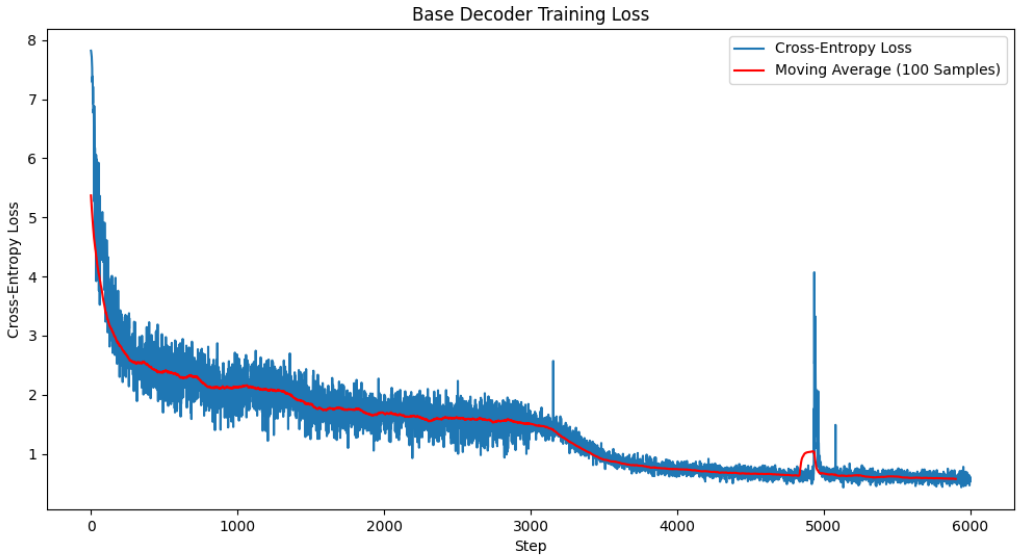

Figure 6: Cross-Entropy Loss Plot for Decoder Pretraining

Combined with cross-attention layers, the complete sequence-to-sequence model totaled ~187 million parameters and took ~4 days to train with the AdamW optimizer on a cluster of 2 A6000 GPUs over 5.69 epochs of a ~20,000 example subset of the multimodal dataset gathered by the end of this thesis. Sequence lengths were limited to 4096, as training on sequence lengths of 8192 was attempted but showed poor results on a model of the same size. The decoder model was trained at a minibatch size of 16, and the encoder-decoder model was trained at a minibatch size of 32.

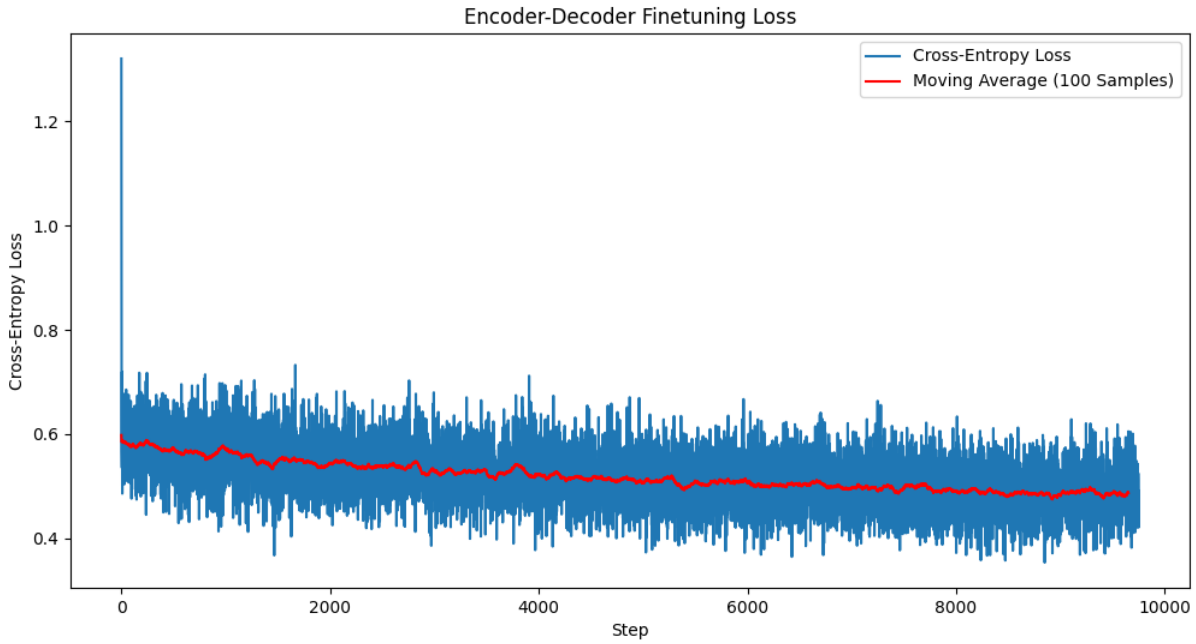

Figure 7: Cross-Entropy Loss Plot for Sequence to Sequence Finetuning

3.3 Model Inference

Due to a lack of training the model to perform unconditioned inference when prompted with a specific token, inference undergoing classifier-free guidance is performed by comparing the output of the guided model with its original decoder trained in the pretraining step. While memory-inefficient, this is functionally identical to other sequence-to-sequence classifier-free guidance methods.

3.4 History and Previous Versions

The first attempt to accomplish these tasks showed limited but promising success. It involved training a smaller model (24M param GPT2, 99M parameters in total) trained exclusively on artist names, artist tags from the Million Song Dataset, and tags from Last.fm. The model was trained at a context length of 512 tokens for a single instrument. Generated MIDIS accurately reproduced music in the style of the artist requested, albeit with a high degree of memorization. Additionally, limited evidence was observed that the model had the potential to interpolate between genres, so I continued research in that direction.

The second attempt to train the model was rather similar to the current model, but included artist names in the dataset and a smaller feed-forward layer per block (2048 instead of 3072). However, I overlooked a mistake in selecting the initial hyperparameters, and the model was trained at a batch size of 2, which was far too small. The model failed to learn anything significant with regards to relating text and music. Additionally, it was around this time that I elected to avoid using artist names in the prompts for the final model, so all of the synthetic prompts had to be rewritten.

The third and current attempt used the procedure previously described. This version is feature-complete, and does everything I set out to do. However, I wanted to see if I could improve the performance of the model while retaining its size and capabilities. The fourth attempt to train the model used a very similar process and identical model size to the third, but with more prompts, a larger vocabulary (4096 tokens with BPE), and a larger context length at 8192 tokens. This attempt ended in failure, however, when it was found that the model did not learn as expected. This was likely due to a lack of additional attention heads and feed-forward nodes, which assist in processing longer contexts.

CHAPTER 4

RESULTS AND DISCUSSION

4.1 General Results

It is difficult to precisely judge the outputs of such a model in written form, so readers are encouraged to listen to the generated examples themselves, which can be found in the supplemental materials for this paper or at https://sammiles.cool/projects/jamformer . A few key behaviors were observed:

- The model appears to have successfully learned the meaning of required/omitted instruments. While this is not perfect at temperatures that also deliver musically interesting and diverse compositions, the model often delivers samples including the requested instruments, and omitting instruments that are not requested.
- The model exhibits the same behavior for tempo descriptions and keys, be they absolute (100 BPM, C Major) or descriptive (downtempo, minor key), although this has not been tested as extensively.
- The model can generate music following specified genre conventions accurately and interestingly (i.e. synthwave, rap, hip-hop, speed metal).
- The model appears to be able to interpolate between musical genres (“Baroque classical hip-hop song”)
- The model appears to generate different results based on the emotional context of the music requested. For example, prompts that contain“the despair of a recent breakup” yield distinct results when compared to results from “losing a lover but are hopeful for the future”.

As the final result of generation is a native MIDI file, samples could easily be imported into REAPER, a digital audio workstation, which allowed for custom instruments to be selected. Samples included with this paper that are not rendered with the default Microsoft MIDI renderer include no alterations to notes or dynamics, but initial levels and synth patches were selected by me for aesthetic purposes. This workflow was very successful, and shows that this model could potentially be useful to working artists within environments they are already familiar with.

4.2 Subjective Evaluation of Hyperparameters and Output

To better determine ideal generation parameters, generated midis were evaluated across a small range of hyperparameters and prompts by 3 human evaluators who had musical performance or audio mixing experience. The following hyperparameters and prompts were used to generate short MIDI files in all permutations on the model:

> γ = 1, 1.2, 1.5, 2.0, 3.0,
>
> Temperature = 0.2, 0.4, 0.6, 0.8
>
> "Death metal pop hit with elements of speed metal and motown R&B. Instruments are electric guitar, bass, and drums. Lyrics are about teen rebellion.",
>
> "An ambient electronica composition with elements of folk and country. Instruments are ethnic, synth bass, synth lead, guitar, synth drums. The song is about the new life of the springtime.",
>
> "A country folk song with elements of baroque classical music.",
>
> "A drum and bass club banger with a classical piano solo."

Evaluators were presented with a pair of midi files for a specific prompt. Inspired by the human preference technique used by Christiano et.al [22], pairs of MIDI files

generated with different unknown hyperparameters were presented to each individual. They were requested to evaluate which of the two MIDI files best adhered to the prompt. Additionally, evaluators were asked to mark which MIDI they preferred between the two. In total, 60 midi files were generated, and therefore 30 comparisons were considered. The win rates for each hyperparameter value are listed in Figure 7. Each entry in the heatmap shows the number of excess wins each hyperparameter value on the Y-axis showed over the value on the x-axis, and the ideal parameters were assumed to be a sum of these excess wins.

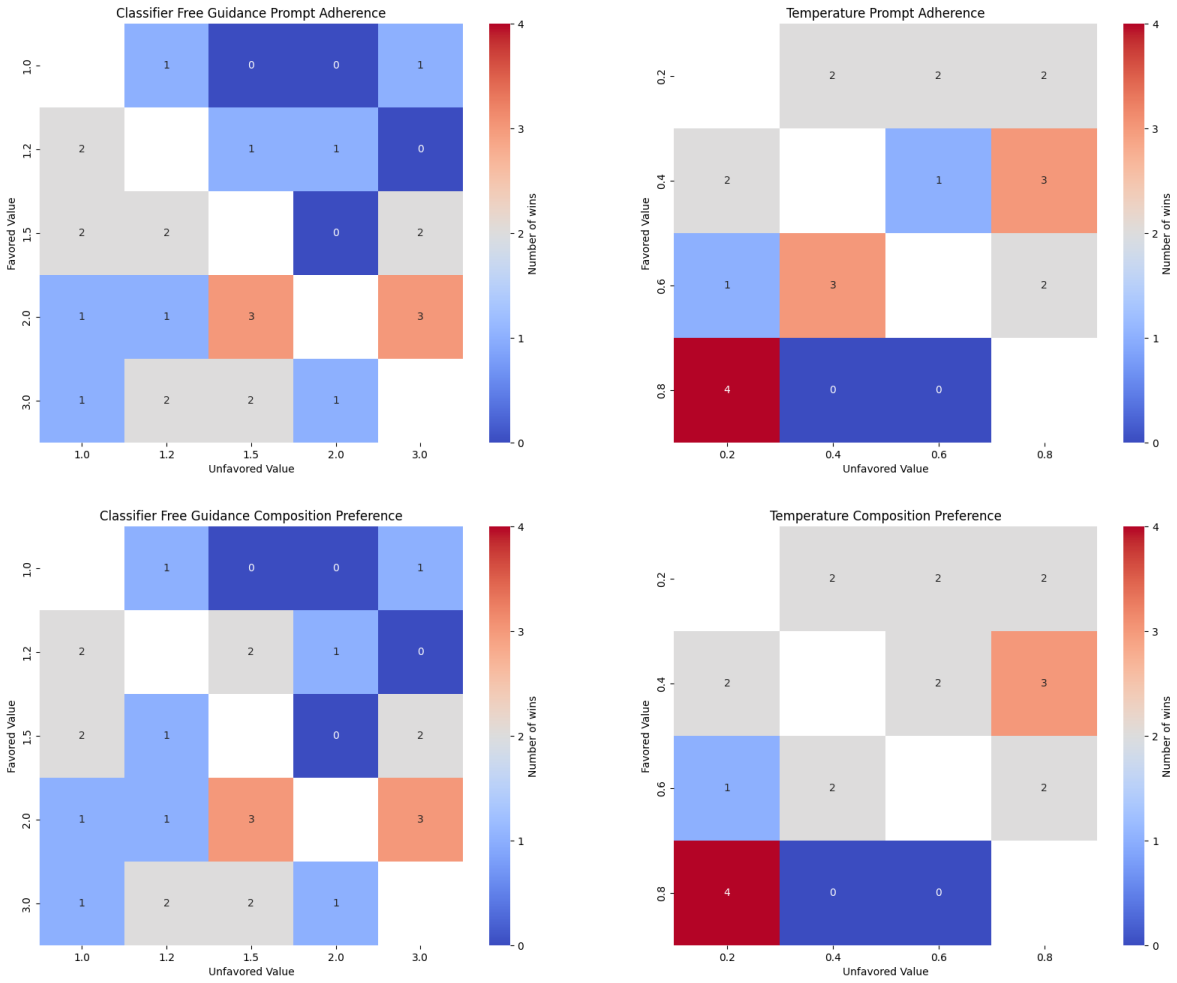

Figure 7: Generation Parameter Effects On Prompt Adherence and User Preference

Based on these results, it was found that a CFG value of 2.0 and a temperature between 0.2 and 0.8 produced results that followed the prompt most closely. Evaluators were also asked to list which sample they personally preferred. It was found that these results were highly correlated with prompt adherence, although this correlation could be exacerbated due to the design of the survey.

As a second test, to determine the quality of the generations as opposed to compositions found in the dataset, evaluators were given a set of unidentified pairs of MIDI files, one generated by the model with an unseen prompt at a temperature of 0.8 and a CFG guidance value of 1.2, and one MIDI randomly selected from the dataset. Evaluators were requested to pick which file was generated by the model. If the evaluator recognized the melody, the sample was discarded. Files from the dataset were put through the same quantization and tokenization procedures as MIDI produced by the model, and truncated to the same token length before being converted back into MIDI. Out of an initial set of 12 comparisons, 3 were discarded due to the evaluator recognizing the melody of the sample. Of the remaining 9 samples, 22% of samples from the dataset were incorrectly identified as being generated by the model. While this sample is not large enough to rigorously establish this figure, and noting these generation parameters were selected before the survey results were properly analyzed, it serves to establish a coarse estimate of the model's generative capabilities when compared with human MIDI composers.

4.3 Limitations

From these results, it is shown that the model outputs are often able to be identified when compared to random human compositions across the dataset. The model exhibits many of the same issues that plague causal generative language models, including repetitive outputs and misalignment with human intent. For instance, without additional generation parameters, the model tends to get stuck in loops as small as a single beat, or as large as a measure, repeated over and over. While many musical features are expressed, it's not uncommon for the model to generate quarter note chords in simple progressions, but this is likely due to the composition of the dataset. This simplicity may not be inherently a flaw, however. Arrangements generated with symbolic music models should hopefully serve as a jumping-off point for the creativity of human artists, whether by editing the generated MIDI and adding their own flair, or by performing alongside it in a hypothetical live generation feedback loop. There is also a limit to the cultural expressivity of the model due to its limited dataset. The Lakh Midi Dataset [18] consists primarily of western music, and therefore models trained on this data may perform inadequately when modeling the music of marginalized cultures.

4.4 Potential Harms

Before discussion of the wider societal implications model like this one impose, it is important to note the possibility of toxicity in the dataset. For instance, inherently racist examples such as minstrel songs and confederate war anthems were found in the dataset and subsequently removed. Keyword filtering was used to avoid including explicitly

bigoted prompt information in the training data, but no such process is guaranteed to be completely adequate without extensive human review, and could be greatly improved with the implementation of human feedback in the training process (as discussed in the Results and Discussion section of this thesis) .

The issue of copyright is still an open question when it comes to generative models. At the time of writing, a high-profile lawsuit against StabilityAI on behalf of the artists whose work was used to train the model [23], and a similar lawsuit has also been served to OpenAI [24]. A key consideration with these models is that they could potentially damage an artist's ability to intake commissions, due to their style being replicated by a machine that requires very little in terms of time and monetary resources to get running. Additionally, there is a chance that the model could reproduce memorized samples, which could result in direct, simple copyright infringement.

Greg Rutowski, renowned artist behind much of the art found on Magic the Gathering cards, has found his name in popular use in text-to-image prompts, and as a result has requested models exclude living artists from their training datasets [25]. To this end, albeit imperfectly, out of respect for the artists and composers, the dataset used to train this model was arranged with the express purpose of occluding artist's names. These names should be re-implemented into the training data of future models if specific artists would find it to be a relevant and desirable form of accreditation. However, these names are included in the training dataset of many pretrained text encoders, so future models may understand the connection between these artists and the genres and styles they work within. In our case, likely due to the small size of the DistilBERT model, it appears to

have forgotten or failed to make this association, but this could prove problematic for larger models with greater sparsity.

A lot of stylistic information about one's art is encapsulated in their name, and it is certain that models trained using the methods outlined in this paper will be implemented involving the names of the artists and songs included in the MIDI datasets available. These models will likely align better with user interest and could be of higher utility to artists, as is clearly demonstrable by the popularity of the practice among those who utilize popular image generation frameworks.

Removing the name of the artists also makes purposeful recollection of memorized data more challenging. One of the first toy models trained in the process of developing this thesis was only prompted with artist names and artist tags on a subset of the final training data, and produced short piano sequences. Without additional information, such as instrumentation, key, and lyrical information, this model likely had the best shot at reproducing memorized samples based on artist names. While this was not extensively evaluated across a gamut of artists, simply prompting the model with "artist name:Rick Astley" would produce a highly recognizable snippet of his song "Never Gonna Give You Up". A similar result was observed by prompting the model to produce a song by The Jackson 5, as many of the melodies generated sounded distinctly similar to "ABC". Therefore, a more general and obfuscated approach to prompting the model with genre, instrumentation, and lyrical intent information would prove valuable in making it difficult for users to purposefully extrapolate memorized samples.

While this technology is still in its infancy and is not able to directly compete with human composers in terms of quality, online music generation services are beginning to

chip away at the market cap for the stock music industry. Additionally, there is significant risk that models such as this one will be used to generate massive amounts of music that will be used for online content farms. Large content creation firms such as Buzzfeed have committed to creating LLM generated listicles and articles [26], surely content that is even less substantive is sure to follow from other groups who have more to gain from producing a massive amount of automated content.

However, there is a key safety feature inherent to the structure of the model that gives it a major safety advantage when compared to other music generation models: it outputs MIDI data. A level of artistic consideration is required to convert MIDI bytes into a listenable or enjoyable piece of music. Spammers with access to such models will find it difficult to escape sounding samey and monotonous, and would likely turn to more hands-free approaches to music generation. Additionally, it is unlikely that someone using these MIDI models to copy the style of a specific artist would get very close without a significant amount of effort on their end, and at that point, it could be considered a cover.

Overall, musical generation systems like this one will have a substantial impact on the future of artists, musicians and content creators. However, I believe that the potential benefits of such a system to working artists outweigh the potential harms, and it is therefore worth publishing research regarding this subject. It is my earnest hope that the artists whose songwriting, performance, recording, and MIDI arrangement work was used to create this model will see it as an attempt to make a useful productivity tool for them.

4.5 Future Work

This model, while interesting, is clearly not the pinnacle of success in this field, nor with its general methods. Many potential improvements to the architecture, training methods, and generation methods are within easy reach of future research.

More high-quality data and a larger parameter count are obvious areas of improvement, a bigger model would of course achieve a lower loss across the training dataset, and a larger quantity of quality training data would of course improve the quality of its output. This could quite simply be accomplished by more thorough labeling methods, and by utilizing more human labellers.

Additionally, for future models, talented human artists could be involved in writing high-quality compositions with the specific intent that they be used to train the model. The paper *Textbooks Are All You Need*[27] shows that training models on only the highest quality data had a significant effect on the quality of a model's output. Setting up incentive structures to solicit high-quality training data from enthusiastic human composers would not only produce better results from the model, it could also serve to alleviate any ethical concerns artists may have regarding the training data.

Fine-tuning the training data for intuitive musical prompting would be immensely useful. One of the artists consulted during the writing of this paper made the following statement: "If I was a robot trying to write a certain type of music I would *need* the mode." How should we deliver this information? It would be straightforward enough to use an off-the-shelf musical analysis library to make a prediction about the mode of the training midis, such that this information could be fed into the prompt creation pipeline.

However, it seems much more intuitive to simply condition the model based on the melodies that the artist has in mind and would like to create a live backing track for, as we can gain insight into how progressions and rhythms will play out. Previous methods of executing this goal, such as for the Music Transformer [10], involves training a separate encoder and cross-attending to the sequence. However, GPT models have shown excellent results by simply prompting the decoder. For example, GPT-2 can famously be prompted to write a summary of previous information, simply by appending "TL;DR"(Too long; didn't read) to the end of the prompt, as this is a common prefix to summaries found on long posts online [28]. One of the experiments attempted to this end was to prompt the model by playing a sequence of notes, then implementing a four note countoff with the hi-hat instrument (as many songs in the training data do). The hope was that the model would attend to the previous melody information and then take the hi-hat countoff as a cue to continue the arrangement. However, the model took this as a cue to continue hitting the hi-hat, and it can be concluded that including a section of the melody before the countoff was too far outside of the training distribution to make sense. Construction of such examples, upon which the model could be fine-tuned, would be somewhat trivial for future work to implement and could be of great use to working musicians.

Instruct-tuning the model to perform notational tasks quickly and easily would also be of great use. It's easy to imagine workflows in which an artist could take an existing midi file, and remix it to fit a different style or meter rather than transcribing it themselves. This training method has seen great success in Stable Diffusion [19]. GPT models can also be prompted with a special "INSERT" token between two sequences, which it is then

trained to interpolate between [29]. This could be especially useful for artists who could use this method for inspiration if they are unsure how to create a cohesive connection between two musical ideas.

With the attention mechanism described in Vaswani et. al [9], memory usage increases exponentially at a rate of O(n^2) as each token performs an attention calculation for all of those that come before it. Recent advancements in Transformer model architectures have decreased the rate at which this memory usage increases by a substantial amount, which makes them easier to run on consumer hardware. From simple changes like the sparse attention found in Longformer[30], to the utilization of state-space machines to replace it almost entirely such as in the H3 language model [31], more efficient methods of attention would greatly reduce the memory usage, and therefore increase the maximum sequence length computable on consumer hardware. The initial conceit of this paper was to create a model that could perform live alongside a human artist on a computer that they could conceivably afford, and a more computationally efficient model would lower the bar for success significantly.

The safety and utility of OpenAI's ChatGPT and GPT-4 can largely be attributed to Reinforcement Learning with Human Feedback, or RLHF [28]. This process involves the creation of a smaller reward model trained to detect which examples of a model's output a human would prefer. This reward model then provides a signal to the primary model during training, indicating the likeliness that a human labeler would approve of the model's output. When applied to music, this could go both ways: it could greatly benefit the quality of instruction adherence and better weed out poor-quality arrangements. Additionally, this could greatly decrease the model's ability to generate toxic content,

such as by rejecting certain requests from the outset. However, there is something somewhat unsettling about setting a goal for the model to make “likable” music instead of just “music likely to occur in the dataset of all music”. Music is interesting and messy, and it could be considered to be inappropriate to create a model that is tuned in such a way that it produces MIDI for arrangements of pop radio hits that are sure to please, instead of allowing flexibility. On the moderation side of things, if RLHF is not implemented for the purposes of rejecting toxic prompts, it would make sense for anyone hosting such a model in a public-facing manner to first check the prompt with another LLM that is fine-tuned for moderation.

CHAPTER 5

CONCLUSION

To summarize, this thesis has outlined the basic formulation, training, evaluation, and experimentation with a large language model designed to convert natural language descriptions of songs into fully orchestrated MIDI compositions. Existing methods and a pretrained encoder were utilized, but by training a new MIDI-specific decoder and by developing a novel synthetic training dataset, a natural language understanding and user-friendly music generation model was developed. To better align the model with the needs and interests of artists, specific pretraining steps were taken to improve fairness and control over the final product. Additionally, it is noted that the architecture and output format of this model will likely be of greater utility to artists than other music generation schema. As technology marches ever forward, researchers and developers should take great care to attend to the needs and assuage the concerns artists face, and hopefully models like this one are a step in the right direction.

BIBLIOGRAPHY

[1] A. Agostinelli, T. I. Denk, Z. Borsos, J. Engel, M. Verzetti, A. Caillon, et al., "MusicLM: Generating Music From Text," 2023. [Online]. Available: arXiv:2301.11325

[2] B. Eno, Generative Music - Brian Eno - In Motion Magazine, http://www.inmotionmagazine.com/eno1.html (accessed Jul. 3, 2023).

[3] M. H. Hassoun, "2.3 Approximation Capabilities of Feedforward Neural Networks for Continuous Functions ," Fundamentals of Artificial Neural Networks, https://neuron.eng.wayne.edu/tarek/MITbook/chap2/2_3.html (accessed Jul. 3, 2023).

[4] I. Loshchilov, F. Hutter, "Decoupled Weight Decay Regularization," 2019. [Online]. Available: arXiv:1711.05101.

[5] S. Hochreiter and J. Schmidhuber, "Long Short-Term Memory," *Neural Computation*, vol. 9, 1997.

[6] T. M. Ingolfsson, "Insights into LSTM architecture," Thorir Mar Ingolfsson, https://thorirmar.com/post/insight_into_lstm/ (accessed Jul. 3, 2023).

[7] D. Eck and J. Schmidhuber, *Learning the Long-Term Structure of the Blues*, Aug. 2002.

[8] I. Simon and S. Oore, “Performance RNN: Generating music with expressive timing and dynamics,” Magenta, https://magenta.tensorflow.org/performance-rnn (accessed Jul. 3, 2023).

[9] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, et al., "Attention is all you need", *CoRR*, vol. abs/1706.03762, 2017.

[10] A. Huang, A. Vaswani, J. Uszkoreit, N. Shazeer, I. Simon, C. Hawthorne, et al., “Music Transformer,” 2018. [Online]. Available: arXiv:1809.04281

[11] C. Payne, “Musenet,” MuseNet, https://openai.com/research/musenet (accessed Jul. 3, 2023).

[12] D. Rütte, L. Biggio, Y. Kilcher, T. Hoffman, “FIGARO: Generating Symbolic Music with Fine-Grained Artistic Control,” 2022. [Online]. Available: arXiv:2201.10936

[13] S. Rothe, S. Narayan, A. Severyn, “Leveraging Pre-trained Checkpoints for Sequence Generation Tasks,” 2020. [Online]. Available: arXiv:1907.12461

[14] G. Sanchez, H. Fan, A. Spangher, E. Levi, P.s. Ammanamanchi, S. Biderman, “Stay on topic with Classifier-Free Guidance,” 2023. [Online]. Access: https://arxiv.org/abs/2306.17806

[15]J. Copet, F. Kreuk, I. Gat, T. Remez, D. Kant, G. Synnaeve, et al,, Simple and Controllable Music Generation. 2023.

[16] N. Fradet, J. Briot, F. Chhel, E. F. Seghrouchni, N. Gutowski. (2021). “Miditok: a Python package for MIDI file tokenization,” Presented at 22nd Int. Society for Music Information Retrieval Conf. [Online]. Available: https://archives.ismir.net/ismir2021/latebreaking/000005.pdf

[17] C. Raffel. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching," 2016. [Online]. Available: http://colinraffel.com/publications/thesis.pdf

[18] T. Bertin-Mahieux, D. P.W. Ellis, B. Whitman, P. Lamere. (2021) “The Million Song Dataset,” Presented at 12th Int. Society for Music Information Retrieval Conf. 2011. [Online]. Available: https://academiccommons.columbia.edu/doi/10.7916/D8377K1H/download

[19] T. Brooks, A. Holynski, A. A. Efros. “InstructPix2Pix: Learning to Follow Image Editing Instructions,” 2023. [Online] Available: arXiv:2211.09800

[20]V. Sanh, L. Debut, J. Chaumond, and T. Wolf, DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. 2019.

[21] T. Wolf, L. Debut, V. Sanh, J. Chaumond, C. Delangue, A. Moi, et al.,“HuggingFace's Transformers: State-of-the-art Natural Language Processing,” 2020. [Online] Available: https://arxiv.org/abs/1910.03771

[22]P. Christiano, J. Leike, T. B. Brown, M. Martic, S. Legg, and D. Amodei, Deep reinforcement learning from human preferences. 2017.

[23] S. E.-D. Mattei, “Artists are suing Artificial Intelligence Companies and the lawsuit could upend legal precedents around art,” ARTnews.com, https://www.artnews.com/art-in-america/features/midjourney-ai-art-image-generators-lawsuit-1234665579/ (accessed Jul. 3, 2023).

[24]C. Thorbecke, “OpenAI, maker of chatgpt, hit with proposed class action lawsuit alleging it stole people’s data | CNN business,” CNN.com, https://www.cnn.com/2023/06/28/tech/openai-chatgpt-microsoft-data-sued/index.html (accessed Jul. 3, 2023).

[25] V. Benzine, “‘A.I. should exclude living artists from its database,’ says one painter whose works were used to fuel image generators,” Artnet News, https://news.artnet.com/art-world/a-i-should-exclude-living-artists-from-its-database-says-one-painter-whose-works-were-used-to-fuel-image-generators-2178352 (accessed Jul. 3, 2023).

[26] N. A.-S. and J. Christian, “BuzzFeed is quietly publishing whole AI-generated articles, not just quizzes,” Futurism, https://futurism.com/buzzfeed-publishing-articles-by-ai (accessed Jul. 3, 2023).

[27]S. Gunasekar, Y. Zhang, J. Aneja, C. C. T. Mendes, A. D. Giorno, S. Gopi,et al, Textbooks Are All You Need. 2023.

[28]A. Radford, J. Wu, R. CHild, D. Luan, D. Amodei, I. Sutskever, “Language Models Are Unsupervised Multitask Learners,” 2019. [Online.] Available:

https://d4mucfpksywv.cloudfront.net/better-language-models/language_models_are_unsupervised_multitask_learners.pdf

[29] M. Bavarian, A. Jiang, H. Jun, and H. Pondé, New GPT-3 capabilities: Edit & insert, https://openai.com/blog/gpt-3-edit-insert (accessed Jul. 3, 2023).

[30] I. Beltagy, M.E.Peters, A.Cohan. "Longformer: The Long-Document Transformer," 2020. [Online]. Access: https://arxiv.org/abs/2004.05150

[31] D.Y. Fu, T. Dao, K.K. Saab, A.W. Thomas, A. Rudra, C. Ré,"Hungry Hungry Hippos: Towards Language Modeling with State Space Models," 2023. [Online]. Access: https://arxiv.org/abs/2212.14052

[32]OpenAI, "GPT-4 Technical Report," 2023. [Online]. Access: https://arxiv.org/abs/2303.08774

VITA

Sam Miles is an Electrical and Computer Engineering Bachelor's/ Master's student at the University of Missouri-Kansas City. He has a deep passion for everything found in the intersection between computer science, electronics, and art. He has experience as an I.T. Technician, Telecommunications intern, and hobbyist audio engineer.