Team Deep House: Project Final Report

Patrick Bjornstad bjornsta@usc.edu
Cole Howard colehowa@usc.edu
Mike DeFranco madefran@usc.edu
Pasha Biglarzadeh biglarza@usc.edu

Abstract

The best deep generative models that explicitly capture percussive elements conditioned on other musical elements (drum accompaniments) use MIDI in an ordered, sequential manner. We propose two modifications to this dominant approach. First, we leverage the Transformer architecture's attention mechanism to remove the need for implicit order in the input. Instead, we explicitly encode time as a token. Because of this, the transformer can better isolate individual instruments. Second, we add genre as extra information for the model to be conditioned on via feature-wise linear modulation (FiLM) to improve adaptability to different music and performance quality. Our model achieves a cross-entropy loss of .38 on the validation set extracted from the Lakh MIDI Dataset (LMD-Matched) conditioned on 30 second song splices.

1 Introduction

Our project aims to extend current research to enhance the creative abilities of generative music models. Music is an interesting data structure in the diversity of ways it can be represented; the two predominant methodologies in deep learning contexts involve 1) converting raw audio into spectrograms using Fourier transformations, which can then be processed using familiar image-based architectures like CNNs, or 2) working with the musical sequence directly using the digital MIDI format, lending itself to sequence-based approaches most commonly leveraged in NLP scenarios.

Inspired by the current progress in language modeling, the group is interested in working with MIDI data in a generative sequence-to-sequence context, especially considering its added interpretability and value as a "tool" to musicians that comes with representing music much the same way humans represent sheet music. Specifically, we aim to generate a fully-featured MIDI percussion arrangement conditioned on MIDI melodic input, that recognizes the implied musical dynamics of the input and deviates meaningfully and obviously from "baseline" drumming behavior in the context of these dynamics.

2 Background & Related Work

2.1 Sequence-to-Sequence Music Generation

Research in music generation with sequence-to-sequence modeling has been conducted with many popular architectures including Hidden Markov Models, RNN, LSTM, CNN, Diffusion, and Transformers. Promising results have been shown with LSTM, CNN, Diffusion, and Transformers. GrooVAE (Gillick et al., 2019) uses a LSTM to generate drum sequences from MIDI. It uses a bidirectional LSTM encoder-decoder architecture to train on melodic sequences and produce percussion output. MuseGan uses a GAN with CNN based implementations of the generator and discriminator. In this case, the piano roll is represented as an image analog and fed into the network. The application of CNN is particularly useful for music prediction because it is able to focus on relative relationships between timing and pitch (Dong et al., 2018).

2.2 Growing Methods in Music Generation

More recently, diffusion models have been explored with results mostly on-par with benchmark models. Their core process relies on gradually adding Gaussian noise to the input and then training a model to reverse the process and learn a latent representation that can generalize on the input's core features. Though most progress in these models has been seen in image generation, Huang et al. demonstrate their capabilities for generating high fidelity music conditioned on text-prompts (Huang et al., 2023). Their approach consists of two diffusion processes: one to encode the audio input into an intermediate representation and a second cascaded version to transform this representation into music. While the model's output produces music of similar quality to its input in terms of genre, tempo, and mood, it's computational complexity allows for only 30 seconds of output. Overall, there is relatively little research on diffusion for music generation, but the preliminary attempts are promising.

2.3 MIDI Tokenization

At its core, MIDI can be thought of as computerized sheet music. It is simply a sequence of distinct messages informing the computer about different events in a song. Most commonly these events are notes, but they can also encode other information like the changing of tempo or note velocity. Tokenization of the complex MIDI data is a key issue in the development of models that generate musical sequences. Unlike language sequences, music sequences contain essential high resolution timing information that the model must keep track of in addition to just the relative position of tokens in the sequence itself. How this added aspect is handled by the tokenization scheme can have large implications for the underlying biases of the model and significantly effect results.

The most intuitive method for tokenizing MIDI treats each step in the sequence as a fixed time step forward in the song itself. This approach can be seen in the context of drum generation in the work by Dahale and others, "Generating Coherent Drum Accompaniment With Fills And Improvisations" (Dahale et al., 2022). This paper was of particular interest to the team due to the similarity in its goals. However, this representation comes with its drawbacks. For time steps in which multiple events occur, (e.g. a chord is played, which involves 3 simultaneous note on events), events either need to be combined or distinct events for combinations need to be created, potentially causing the dimensionality of the data to explode. Furthermore, this method forces the output into a particular rhythmic structure as defined by the time step used in the sequence.

An alternative method for MIDI tokenization that addresses these concerns was first proposed by Oore et al. in "This Time with Feeling: Learning Expressive Musical Performance" and later adopted in other leading transformer-based MIDI tasks (Oore et al., 2020; Huang et al., 2019), which treats the event sequence itself as time-independent while using dedicated tokens to encode the passage of time. This encoding largely preserves the message structure of the raw MIDI data, and the decoupling of the sequence from time allows for arbitrary amounts of information to be encoded into a single point in the song. In the context of drums, this would allow a model to learn dynamic behavior such as distinct velocities for every different drum note, even if they all occur simultaneously in the song. It also effectively avoids unwanted implicit biases on the rhythmic structure of the song that might be imposed by alternative schemes, which is especially important to the task of generating drums.

2.4 Transformers

Transformers have shown the most promise for music generation. Google Brain's Music Transformer (Huang et al., 2019) is often cited, and uses a relative version of self attention to help represent relative position of notes and timing; while, traditional transformers use absolute position. In addition, they are able to use a memory efficient version of relative self-attention. Magenta, the company that performed the research, also included a visualization tool to help understand where the transformer network is applying focus. Compared to Magenta's PerformanceRNN, an LSTM based architecture, the transformer architecture achieved better validation results as well as better results when presented to a human audience (Huang et al., 2019).

Music Transformer uses the tokenization scheme developed by (Oore et al., 2020) and demonstrates its effectiveness as input for a transformer specifically. They were able to use longer input sequences due to the efficiency of the transformer architecture, accommodating one of the main drawback's of the Oore scheme in "This Time with Feeling".

The aforementioned work by Dahale used a transformer architecture alongside the time-locked token sequence approach to generate drum accompaniments (Dahale et al., 2022), where significant emphasis was placed on generating dynamic and improvisational performances. The work addresses a challenge with generating drum sequence that is present in many transformer based architectures; transformer generated drum sequences lack creative motifs known as fills or improvisations, which are closely associated with human creativity. A drum track with the absence of these fills would be recognized as a valid drum track to the human ear, but would lack interest and subtlety. The paper uses a multi-tiered architecture: a transformer network first generates a valid drum pattern that likely represents a prototypical arrangement for a given melody, then, a BERT-like model figures out where a good improvisation location would be conditioned on the melodic accompaniment, and finally a MLP model for generating a creative improvisation.

2.5 Remaining Work

The goals of the group revolve around three key existing works in the space. Inspired by the potential of Oore's tokenization scheme and the freedom it allows (both in the performance itself, allowing for dynamic use of things like velocity to create varied patterns, but also in the rhythmic structure, allowing the model to work with whichever time scheme it deems most effective), as well as the scheme's demonstrated effectiveness in the Music Transformer paper, the group aims to build a similar encoder-decoder architecture focused specifically on drum generation as a "translation" task from an entire melodic input sequence and then augment it with extra information like genre. The hope is that the more dynamic tokenization scheme allows for expressive and interesting performances in lieu of a dedicated module to generate improvisations as implemented by Dahale et al.

3 Data Processing and Representation

MIDI data naturally lends itself to sequence based approaches most commonly found in natural language processing applications, but its increased complexity as compared to basic text data requires additional steps to be taken to prepare the data for the task. At its core, MIDI consists of a sequence of timestamped message objects which each encode a specific action for the computer playing music to execute (e.g. turn a specific note on or off). These message sequences are organized into independent tracks so that a single MIDI file can encode multiple instrumental elements.

The specific dataset selected for use is the "LMD-Matched" variant of the Lakh MIDI Dataset v0.1 (Raffel, 2016), which includes raw MIDI files for over 40,000 songs spanning a diversity of genres, most notably rock, pop, and electronic. The model presented here was only trained on a small subset of this larger LMD-Matched dataset. Unlike similar works in the space, which oftentimes train on a specific genre of music and predefined instruments, we seek to condition the model on global features of the song like genre or time signature. This allows for training across a variety of different song types and in theory enables customization in the output by preconditioning on different genres or time signatures. In doing so this necessitates accommodating a wide variety of possible instruments in the melodic input, and an adaptation of the Oore tokenization scheme.

3.1 Preprocessing

Not all aspects of the raw files are useful for the task, and not all raw files/songs will contain the specific information required for the task. Especially considering the large volume and diversity of the LMD-Matched dataset, a number of preprocessing steps are required prior to tokenization:

1. Discard invalid/corrupt files
2. Quantize note durations to desired resolution (high enough to still enable expressiveness)
3. Identify MIDI tracks by instrument and categorize into buckets
4. Discard files without drums or sufficient accompaniment
5. Filter accompaniment tracks to only include the 4 most important/active instruments (reduces sequence size, with a priority on bass and piano)

Any MIDI files not matching these specifications are discarded from the final cleaned dataset. At this point, the songs are spliced into 30 second clips before being tokenized. This was done to accommodate our finite resources, most notably memory, since transformer attention on long sequence lengths can quickly become intractable. Exploring longer song clips in an effort to capture longer term progressive performances would be an interesting future direction for the project. Using 30 second clips, input sequence lengths could be kept below 2048, which was tractable using small batches given our HPC resources. After being spliced, the files are ready to be tokenized.

3.2 Tokenization Scheme

As previously mentioned, the group adapts the base tokenization scheme proposed by Oore et al. The Python library MidiTok supports tokenization of MIDI files into this format. In line with the original paper, the main tokenization vocabulary consists of 128 NOTE_ON events and 128 corresponding NOTE_OFF events (1 event for each of the 128 pitches considered in typical western music), TIME_SHIFT tokens for each increment of beat resolution, and VELOCITY_SHIFT tokens for each increment of velocity resolution. The TIME_SHIFT tokens manually move the actual time of the song forward, so any other tokens that appear between two time shift tokens can be interpreted as occurring simultaneously when the track is played back. VELOCITY_SHIFT tokens change the note velocity (how "hard" a note is pressed) of all subsequent NOTE_ON tokens, until the next velocity change.

Additionally, given the task's focus on drums, the group also elected to include TEMPO_SHIFT tokens, which encode a change of tempo (BPM) of the song. This token was also natively supported by MidiTok and thus was easy to implement.

The main adaption of the original scheme revolves around accommodating multiple instruments in the same melodic input sequence. Unfortunately, given the format of MIDI data, and the software libraries being used, it was not feasible to merge instrument sequences together and introduce tokens that change the current instrument much in the way the VELOCITY_SHIFT tokens function. Instead, each instrument must be tokenized separately. However, we argue that simply concatenating the individual instrument sequences together is still a practical approach. Here, the individual instrument sequences are separated by INSTRUMENT_CHANGE tokens that set the new instrument to a specific bucket or category and implicitly "reset" the timing of the track and allow for a new instrument to play its contribution to the song clip. The benefit of this approach is that there is no need to introduce distinct versions of the main event tokens for each instrument. The cost is a long sequence length and important relationships between tokens that are far apart in the sequence (e.g., a bass and piano playing simultaneously).

The transformer architecture is very well suited for this task, and mitigates many of the disadvantages of this approach. The self-attention mechanism is ideal for learning relationships between tokens that are far apart, in our case learning relationships between the different instruments' contributions to the overall sequence.

Considering these adaptations, and including the basic START, END, and PAD tokens, the full event vocabulary is as follows (320 tokens total):

- NOTE_ON tokens for each possible pitch
- NOTE_OFF tokens for each possible pitch
- TIME_SHIFT tokens for each increment of beat resolution
- VELOCITY_SHIFT tokens for each increment of velocity resolution
- TEMPO_SHIFT tokens for each increment of tempo resolution
- START, END, PAD tokens
- INSTRUMENT_CHANGE tokens, for each instrument type (accompaniment sequence only)

The INSTRUMENT_CHANGE token encodes a change to a specific new track/instrument in the sequence, and thus is only relevant for our accompaniment input sequence. There are as many INSTRUMENT_CHANGE events as there are distinct instrument types to be considered (in our results, instruments were categorized into 12 generic buckets of different types like piano, brass, guitar, etc). It is worth noting that the concatenation of tracks in the sequence to represent multiple instruments has precedence in Dahale's works despite the different tokenization scheme used (Dahale et al., 2022).

Each event is simply tokenized as an integer index and fed into the familiar transformer architecture with learned embeddings.

3.3 Global Song Conditioning Features

In order to condition the model on song aspects like genre, additional global features for each song were extracted at tokenization time. These included the top 3 genre tags for the song (extracted from the metadata associated with the LMD-Matched dataset), the time signature of the song (extracted from the MIDI itself), and the ticks per beat of the song (also extracted from MIDI). Ticks per beat encodes how the MIDI file measures time, so while it is not a feature we introduce with the intent to experiment at inference time, it is still important information to tell the model so it can interpret the sequence.

All of the global features were considered categorical and represented to the model using learnt embeddings. The group felt it was important to preserve the sequential aspect of the main melodic input, and not simply pollute it with global features attached to the end of the input. Instead, the conditioning variables are fed to the network at different stages of the main transformer computation flow via individual "FiLM" (Feature-Wise Linear Modulation) branches of the network (Perez et al., 2017). These FiLM layers take in the conditioning input and learn affine transformations of the weights of the transformer (a scale and a shift parameter for each activation in a certain component of the transformer). This allows the model to ingest the global features of the song and use them to further inform any aspect of the main encoder-decoder process without corrupting the main input sequence itself. More details on the use of FiLM will be discussed in further sections.

3.4 Pipeline

The group has developed scripts to automate the tokenization process for input data as well as the detokenization process for our generated drum sequences. The inference functions developed by the group produce drum sequences in the scheme described above, allowing the detokenizer to compile the input sequence to MIDI with either the original drums or the generated drums, or compile any of the sequences individually.

4 Model Architecture

The initial model architecture consisted of two major increments. The first uses a Standard Transformer Encoder-Decoder architecture used in machine translation. We used this architecture to evaluate how the Oore tokenizations in conjunction with a Standard Transformer performed on Drum Translation. The second model incorporated feature-wise linear modulation in order improve generalization and to generate interesting sequences by applying transformations to layer activations based on genre and tempo.

4.1 Transformer Encoder-Decoder

The Encoder-Decoder model accepts as input the tokenized sequence of spliced accompaniments and drum accompaniment labels. First, parameters are initialized using a Xavier uniform initialization. The spliced accompaniments are fed through an embedding layer with hidden dimension of 512, followed by sinusoidal positional encodings. The output of these are fed into the Transformer Encoder, where attention is applied and two linear layers with Dropout and Layer Normalization produce the latent representation. In the transformer decoder layer, the latent representation and masked input are used to generate a target sequence, followed by a generator, which is a fully connected layer to raise the sequence length to the proper output. The output is compared to the label using cross-entropy loss. (Vaswani et al., 2017)

4.2 Feature-wise Linear Modulation (FiLM)

The motivation to apply FiLM was twofold. First, the initial model performed well, but drum outputs were quite different from the original, so including genre and tempo into the model provided the necessary context to improve generalization. Additionally, we would like to improve the creative abilities of the model and be able to generate sequences that sound interesting and real. In regards to music, blending styles or playing one style with inspiration from another can lead to unique sounding music, so incorporating FiLM generators and layer transformations should allow for some creative input from the user upon accompaniment generation. (Perez et al., 2017)

To incorporate feature-wise layer transformations from genre and tempo, the Transformer architecture was changed to include FiLM generators and FiLM layer transformations. The FiLM generator is essentially a FC layer that takes as input three tokenized genres of music, the tokenized time signature, and ticks per beat of the song and produces the scaling and shifting factor for every weight targeted in the transformer network itself. Separate parameterizations are learned for the scale and shift of unique layer outputs like the self attention module or the feed-forward networks inside a typical transformer setup. The FiLM incorporated Transformer applied here uses the scale and shift at all major points in the network's encoder and decoder layers: the output of the attention layers themselves (including both self attention and cross attention) and the output of the feed-forward layers pre-dropout and normalization. In effect, this implementation of the model has FiLM applied globally to all of the transformer's parameters apart from the positional and token embeddings. Experimenting with more customized configurations applied to select aspects of the transformer would be an interesting hyperparameter to explore further.(Perez et al., 2017)

5 Results and Evaluation

5.1 Training and Testing Evaluation

Both subjective and qualitative tests were performed to evaluate the model. Training and testing loss improved substantially as updates were made to both the data and the model. Our original input data included only the beginning slices of song accompaniments, which drums and melody potentially not being in the same bucket of time. Training and testing loss saw significant improvement by adding multi instrument splicing such that the drums were learned from multiple other elements of the song.

The standard Transformer model performed satisfactorily on the training dataset of 1000 songs. We have included loss curves and a couple of audio snippets below to listen to the output of this model in comparison with the original drums. The loss curves indicate that the model stops learning anything useful around epoch 40 in this example with validation loss at a consistent 0.6. Subjective listening tests of the standard model indicate that the transformer is able to successfully learn from the Oore tokenizations a number of important musical qualities including, the tempo of the song, the relative complexity of the drum sequence, and the appropriate position of kick drum, snare drum, and cymbals in order to play along to the melody. However, The model is challenged to create drum fills, improvisations or transitions in agreement with Dahale et al. (Dahale et al., 2022)

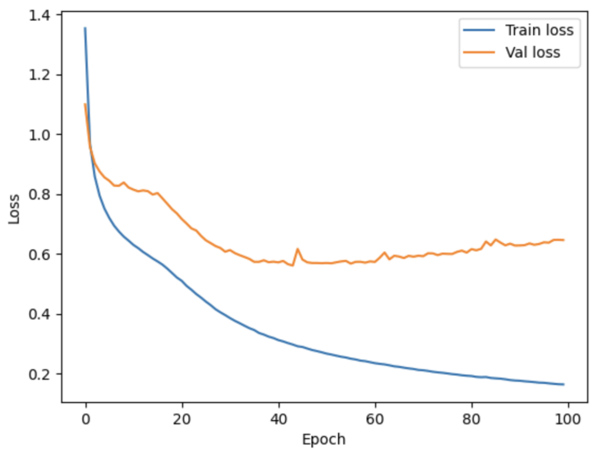

Figure 1: Training and Validation loss for the Standard Transformer model. The following links are to a sample with the original drums followed by a sample with the generated drums. [Original Audio Sample] [Generated Audio Sample]

The FiLMed Transformer saw significant improvement in training and testing loss over the standard transformer model (see below). The output of the MIDI data upon subjective analysis indicates that the model is able to more closely match the original drum accompaniment in style, timing, and complexity. The addition of genre, time signature, and ticks per beat information to add context to the model proved successful for generating more high fidelity music with the same amount of training data. The output of the model upon listening tests is able to produce more human-like drums, often opting to depart from the standard four-on-the-floor model that the Standard Transformer liked to default to.

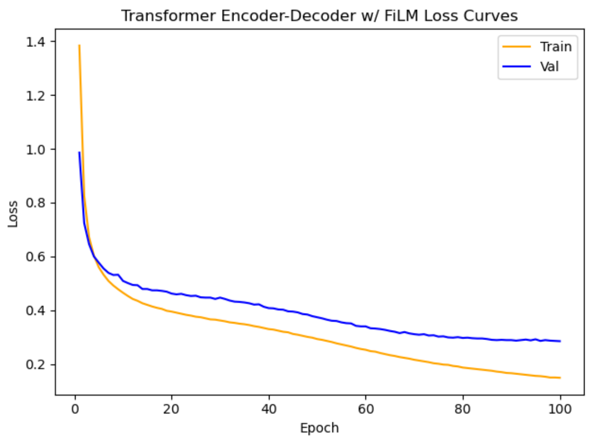

Figure 2: Training and Validation loss for the FiLMed Transformer model. The following links are to a sample with the original drums followed by a sample with the generated drums. [Original Audio Sample] [Generated Audio Sample]

5.2 Experimentation with Conditioning Variables

A large benefit of the FiLMed Transformer model is the ability to provide input to the model during generation. The list of available genres is substantial and descriptive, allowing the decoder to apply feature-wise transformations during generation to a different genre than the original, or a combination of multiple sub-genres. Our approach differed from Dahale et al. in that instead of focusing on where to apply creative input, the model accepts input from the user to potentially generate many different versions of drums for an accompaniment, and pick the output that suits the musician style.

The first test of the FiLM model with conditioning variables involves testing that the FiLM transformations are being applied correctly, and that choosing a genre will result in the output being more similar in style to that genre. For this, we performed Drum Translation on an accompaniment. We held the accompaniment constant and just changed the genre conditioning during generation. The first example held the original melodic accompaniment the same and generated drums using alternate genre tags: rave, Chicago house, and techno. The intention being to force the generated model to conform to the typical thumping kick drum of most techno music. Here is a sample of all original drums [Original], the translated drums [Translated], and the drums generated with genres rave, Chicago house, and techno [Conditioned] Here, the FiLMed transformer demonstrated the ability to learn some features of specific genres and apply them to different melodies.

The next test of the FiLM model is testing if adding conditioning genre tags can augment the creative abilities for generation and produce original musical ideas. For this, the model was conditioned with genre tags that are seldom seen together. For this example, the genre tags were Metal, Underground Rap, and Jazz. Here is the audio for the original drums [Original] the translated drums [Translated] and the conditioned drums [Conditioned]. In this example, the model has been able to produce an interesting beat that is well positioned within the song. In this example, the previous, and many others, the FiLM conditioned drums we more interesting the originally translated drum accompaniment. This is likely due to the model interpolating between the parameter space from one genre to another.

5.3 Quantitative Metrics

Like many other models generating some sort of art, whether images or sounds, human evaluation tends to be a preference as there is no truly effective objective metric to evaluate the quality of the art. Nevertheless, we have researched several evaluation methods to advance our quantitative analysis of the model's output. One of the initial methods employed was using the Jaccard similarity metric to measure the similarity between two MIDI files, where the number of notes played differed within the same period of time.

In general, the Jaccard similarity measures the similarity between two sets of objects by dividing the intersection of those sets with the union of those sets. In the context of MIDI files, we represented each file as a set of unique note events, and calculated the Jaccard similarity between the two sets of notes. However, our evaluation matrix comparing the similarity of a small sample of our midi outputs to their respective inputs did not produce a distinctive correlation based on this evaluation metric.

To get a better idea of how our model performed over a set of songs, we analyzed the how the probability distributions between the generated samples and the target samples. To do so, we employed the Wasserstein distance. After aggregating all feature counts from both the original and the generated midi files, the global dissimilarity between the distributions was .07. By comparison, the baseline model achieved a value of .08.

Separate from objective measures, we made subjective judgements on the results of the model. Consistently, we found that the generated music had more variation in the drum patterns themselves. While they still sounded plausible, they tended to have more fluctuations in timing and especially velocity. Figure 3 shows a representative example of this. To further evaluate the output of our model,

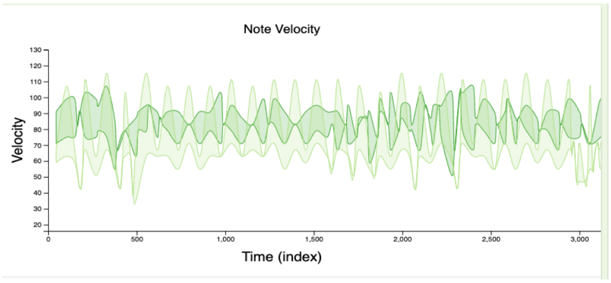

Figure 3: Graph of note velocities over time where dark green is the original song and light green is our model's output.

we utilized an open-source visualization tool that allowed us to compare the inputted MIDI files to their corresponding output files. This visualization tool enabled us to view the individual notes played at each timestamp, the frequency of the unique notes played, and the note velocity with respect to time.

To further evaluate the output of our model, we utilized an open-source visualization tool that allowed us to compare the inputted MIDI files to their corresponding output files. This visualization tool enabled us to view the individual notes played at each timestamp, the frequency of the unique notes played, and the note velocity with respect to time

(Grifski, 2019).

6 Conclusion & Future Work

This architecture shows promise in the ability to generate dynamic drum performances without relying on entire separate modules to manually inject variation. Most of the model's shortcomings can be attributed to the size, quality, and diversity of the dataset. Given more time and computational resources, the group would have liked to explore larger samples of the LMD-Matched dataset, and samples of durations longer than 30 seconds, in an effort to capture more interesting behaviors and improve generalization ability. Additionally, the group would have liked to experiment with different numbers of instruments to consider, and different instrument buckets entirely. Overall, the group is confident that given a large enough model and long enough training time, this proposed architecture would be effective at capturing, encoding, and generating dynamic performances across genres. In an ideal world, manually vetting the input data or even crafting one's own melodic inputs in MIDI form at a large enough scale would improve the average data quality significantly and improve performance more meaningfully than using LMD-Matched, which suffers from noise in both the quality of the MIDI songs and the confidence of the genre tags.

From an architectural perspective, the group would have liked to experiment with different configurations of the FiLM modules. This would include testing FiLM only applied to encoder activations or decoder activations, as opposed to the entire network which was implemented in this paper. Additionally, a custom loss function that adapts cross entropy to be more aware of our tokenization scheme is an interesting hypothetical avenue to explore. In theory, there are many possible "correct" configurations of tokens that fall between two `TIME_SHIFT` tokens, since these events all occur simultaneously in the song. As long as the relative relationships between `VELOCITY_SHIFT` tokens and `NOTE_ON` tokens is preserved, the model should not be penalized for configuring the same tokens in a way that differs from the target within two time shift events. A custom loss that takes this into account could help the model better learn the underlying relationship of the sequence to the song.

Overall, this project would benefit from increased research investment into the future. Initial results were surprisingly effective given the consumer-grade data limitations posed by MIDI information from the internet. There are a number of possibilities for improvement upon this base architecture and will be an interesting problem to explore in future research.

Group Contributions

Patrick Bjornstad Primarily responsible for development of MIDI data processing, representation, tokenization, and detokenization throughout the entire process including project survey and mid-report. During the later stages of the project was very involved in adapting the original transformer encoder-decoder drum generator to ingest global conditioning features using FiLM. As with all group members, contributed to recurring deliverables like presentations and intermediate papers. Authored introduction, related work, and data sections of the final report.

Cole Howard Used initial transformer framework from Pasha and converted it to structure as delineated in (Dahale et al., 2022). Also trained/tuned it to verify functionality and efficacy. Evaluated successive models based on variety of metrics and then developed visualizations for the results. Experimented with diffusion models which ultimately led to a poor fit for the task at hand.

Mike DeFranco In Mid-Report: Ran baselines and authored section on baselines. Built off dataset and model code to write training loop with model params from (Huang et al., 2019). For Final Report: Got Standard Transformer to train. Ran training and generated midi data and audio samples from FiLM and Standard Transformer models. Incorporated improvements to traning and generation. In paper authored section on Model Architecture and Model Evaluation.

Pasha Biglarzadeh Developed initial framework of code that sets up a transformer model architecture for sequence-to-sequence learning, loads training data tailored to expected data representation, and trains the model. Researched and implemented evaluation metrics and visualization tools for model results.

References

Rishabh Dahale, Vaibhav Talwadker, Preeti Rao, and Prateek Verma. 2022. Generating coherent drum accompaniment with fills and improvisations. International Society for Music Information Retrieval Conference.

Hao-Wen Dong, Wen-Yi Hsiao, Li-Chia Yang, and Yi-Hsuan Yang. 2018. Musegan: Multi-track sequential generative adversarial networks for symbolic music generation and accompaniment. Proceedings of the AAAI Conference on Artificial Intelligence.

Jon Gillick, Adam Roberts, Jesse Engel, Douglas Eck, and David Bamman. 2019. Learning to groove with inverse sequence transformations. Proceedings of Machine Learning Research.

Jeremy Grifski. 2019. Juxtamidi.

Cheng-Zhi Anna Huang, Ashish Vaswani, Jakob Uszkoreit, Noam Shazeer, Ian Simon, Curtis Hawthorne, Andrew Dai, Matt Hoffman, Monica Dinculescu, and Douglas Eck. 2019. Music transformer: Generating music with long-term structure. International Conference on Learning Representations.

Qingqing Huang, Daniel Park, Tao Wang, Denk Timo, Andy Ly, Nanxin Chen, Zhengdong Zhang, Zhishuai Zhang, Jiahui Yu, Jesse Engel, Quoc V. Le, William Chan, Zhifeng Chen, and Wei Han. 2023. Noise2music: Text-conditioned music generation with diffusion models.

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

Ethan Perez, Florian Strub, Harm de Vries, Vincent Dumoulin, and Aaron Courville. 2017. Film: Visual reasoning with a general conditioning layer.

Colin Raffel. 2016. *Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching*. Ph.D. thesis, Columbia University.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.