Using Deep Neural Models for Symbolic Music Completion and Generation

Hari Sethuraman
University of Washington, Seattle
hsethu@cs.washington.edu

Derek Zhou
University of Washington, Seattle
derekz0@uw.edu

Mark Pock
University of Washington, Seattle
markpock@cs.washington.edu

Abstract

The symbolic music domain has seen a wide variety of architectures for its sequence-to-sequence transformations. In this paper, we explore two architectures for two structurally different tasks in this domain.

*We first apply neural machine translation techniques to jazz soloing **(from-context)**, where we use a transformer based model 'JazzGPT' that takes in a sequence of chords, and returns a melody that fits over that sequence of chords. We train this model on quantized data from jazz solos by several jazz musicians.*

*We also explore ex nihilo **(from-scratch)** music generation. To this end, propose 'MAR-Y' (Multiple Attention RNN), as an architecture for symbolic music generation. We train the model on the MIDI scores of several J-pop songs to generate complete melodic-harmonic excerpts in this style. We then use rhythmic metrics to evaluate the model against actual J-pop.*

1. Introduction

Symbolic music generation is a general class of tasks encompassing the generation of musical elements from structured symbolic representations. There has been controversy in recent years over the provenance of AI-generated music (and art more generally), but we argue that a musician's creative output can almost always be traced back to creative influences. In jazz improvisation, it's common for one musician to cite entire passages. In composition, chord changes are often directly borrowed from other songs. The use of deep neural models to generate music thus is a natural step, as they train on a large quantity of data, analogous to listening to a large quantity of music, and generate something in the style of the training data.

Both language modelling and symbolic music generation are subsets of the sequence generation task class, which naturally leads to a next-token prediction strategy. Like language, music can be characterized by a certain arc or 'form', usually genre-dependent. For instance, an arc may have a mellow start and end with a denser middle, analogous to development in linguistic rhetoric. Unlike language, however, form persists at all levels of structure for music, from small cells at the level of measures to overarching structure at the level of songs. In particular, most music is characterized by motifs, which are short melodic fragments that are repeated multiple times throughout a piece. For symbolic music generation, tokens are embeddings of notes storing information such as pitch and duration. These are crucial as they lend cohesion to a piece. Hence, when predicting the next note, the model will need access to the entirety or a significant length of the preceding sequence. Concretely, this means either an LSTM or attention-based model with a large context window will need to be used.

Music generation can be categorized into three general categories: music completion (given a sequence, extend or complete sequence), generation from scratch (no input sequence), and melodic generation over a harmonic structure (i.e. jazz soloing). This paper will implement both generation from scratch and melodic generation over a harmonic structure. Though there is some inevitable commonality between semantic representation between the two (the initial motivation for pursuing both subdomains), each paradigm demands a different methodology for approaching data. For from-scratch generation, the model will not be given any input, and must output a complete cell of music. The paper will also implement music generation given a context i.e. by generating jazz solos over a chord sequence.

MIDI is frequently used to represent music digitally. Each note in MIDI consists of a pitch and the duration the pitch is held, and the offset of the pitch from the start of the file. An issue with MIDI, however, is that notes are not quantified with regards to their duration i.e. notes may not exactly be a beat or a quarter of a beat. This project will use

MIDI to encode the training data, and will generate music in the form of MIDI, which can then be translated into an audio format such as MP3. However, much modern research in MIDI generation has focused on generating something affectively complete, which most important entails generating dynamics (represented in MIDI as a velocity). We are interested the practical applications of AI-assisted technology for composition and so as such see such models as a means of ideation. We thus exclude dynamics (amongst other factors such as tempo and time signature) from our representation in the hopes that the focus on a few attributes will better allow the model to hone in cogently on certain aspects of the style. Hence, when processing a MIDI file, the notes will need to be quantized according to beat and stripped of dynamic information.

In the case of the jazz (from context) model, since only a melody line is being generated, and because the data set consists of solos played by many instruments, the instrument will not be as important, and can be changed based on preference.

In the case of the J-pop (from scratch) model, keyboard arrangements of pieces would be the most concise way to capture harmonic, melodic and rhythmic information, due to the polyphonic nature of keyboard instruments (ability to play multiple notes simultaneously).

1.1. Related works

A variety of other models have attempted symbolic music generation. Three distinct architectures follow below.

A model by Skuli [4] used a series of two LSTMs and dense layers, training on a MIDI dataset of 350 pieces from the Japanese franchise Final Fantasy. Notes and chords are distinct objects, and are represented using one-hot embedding. An embedding consists of pitch and duration. Training over 20 hours, the model produced music that, on first listen sounds passable. The music has rhythmic consistency and captures the feel of Final Fantasy. It also prominently uses motifs and repetition, which are key components of any song. However, the model does not incorporate rests, which can result in the music being too dense, without any pause.

Another model by Huang et al. [2] used an attention-based architecture to generate Pop Piano compositions. Unlike other models that used MIDI, this paper proposes the use of ‘REMI,’ which is allows temporal information such as the bar and the beat (represented as the offset within a bar) to be encoded. The offset can be any integer value between 1 and 16. This discrete nature allows predictions to be more rhythmically coherent. The paper also proposes metrics for evaluating a prediction’s rhythm. These include the downbeat salience and the downbeat standard deviation. Another heuristic proposed is for evaluating conformity to tonal harmony, via adding points for chord tones (notes within a chord) and deducting points for dissonant notes.

BebopNet [1], an architecture proposed for generating jazz solos, suggests a metric for harmonic coherence, i.e. by measuring the number of melody notes that align with the chord or scale. Another metric suggested is regarding plagiarism, which can be measured by comparing the percent of n-grams that appear in a song to other songs in the training data set. The use of this is not limited to jazz either as plagiarism is a concern for music generation.

These papers show that multiple metrics exist to quantify how ‘good’ a piece of music is. It is worth noting that, while these papers did use multiple numerical heuristics, they also relied heavily on the subjective judgement of other people. Thus, we intend to use both methods when evaluating the results of our models.

1.2. Problem Statement

This paper attempts to solve two problems:

- Given a sequence of chord changes (represented by strings i.e. ’Bb7’, ’C-6’,, ...), the model should generate a sequence of notes that can be played over the changes in the style of jazz. The sequence of notes can then be translated to MIDI.
- From no input, our first model must generate a piece in the style of J-Pop with n notes, where the i^{th} note is a prediction given at most the previous c notes, where c is the context length (i.e. how many preceding notes the model has access to). The piece will be generated in MIDI format.

They are united by their consciousness for generation as a horizontal next-token prediction strategy, demanding a broad understanding of musical information as embedded in discrete higher-dimensional spaces.

2. Method

In this section we discuss the implementation details and methods used to develop our models.

2.1. Models

2.1.1 JazzGPT: From-context

The information required for jazz soloing can be categorized into the harmonic structure of the song to solo over i.e. the chord changes, which is often learned by looking at a ‘lead sheet’, and the general vocabulary of jazz soloing, often learned by learning several jazz solos of other musicians and incorporating the solo’s vocabulary into one’s own soloing. We decided to approach developing a jazz generating model in a similar manner: the model is fed in a sequence of chords, and it produces a sequence of notes

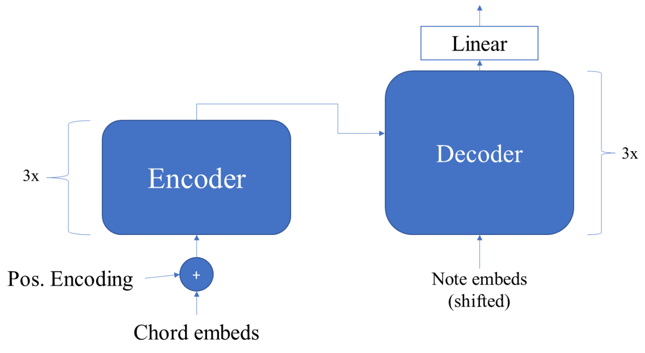

Figure 1. Sequence-to-sequence transformer

over the chords. This task, hence, reduces to a sequence-to-sequence one. Specifically, it can be treated as a language translation problem, where the input 'language' would be the chord changes, and the output would be the melody.

Candidate architectures to use are LSTMs and Transformers, both of which are state-of-the-art models for sequential tasks. While both can have similar performances for language modelling, we opt to use a transformer for a few reasons. First, the parallelizable nature of transformers makes it easier to train. Additionally, long range dependencies are more important in music than in regular translation, as it is common for motifs to repeat sparsely through a piece to make it more cohesive. Finally, we can use masked attention to ensure that the transformer does not look ahead when training, making it equivalent to the LSTM in this regard. The model will hence be an encoder-decoder transformer. We will be using three encoder and three decoder layers.

Thus, the model, in its entirety, consists first of an embedding layer for the harmony with positional encoding followed by the transformer encoder. The output of the encoder is passed into the decoder, along with embeddings with positional encoding for the notes, with an embedding size of 128. The output of the decoder is passed into an affine layer, of size $(128, V)$, where V refers to the total number of distinct notes that exist in our dataset, analogous to the vocabulary length in text generation tasks. These scores are then passed into a softmax layer over which cross-entropy loss is applied.

For inference, we initially used a greedy decoder that always picked the note with the highest score. This resulted in the same note being chosen repeatedly. Therefore, we decided to use random sampling on the probabilities generated by the softmax along with a mild sharpening function of the probability distribution, such as $x\ln(x)$ or $x\sqrt{x}$.

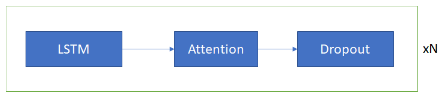

Figure 2. A single cell of the model architecture.

2.1.2 MAR-Y: From-scratch

Our second model is based upon the use of a LSTM, as proposed by Skuli [4] - an LSTM is one of the most common and intuitive methods for sequence generation. However, our the model is vastly expanded 1.

We name our model architecture 'Multiple Attention RNN' (MAR-Y). A single cell involves a bi-directional LSTM with a hidden dimension of 512, the output of which is passed through a sequential-self-attention layer, followed by a dropout layer. This cell is repeated some N times.

Musically speaking, notes rely on both the preceding and succeeding notes for structure. We thus find the bidirectional property of our LSTM a natural move. Since motifs are often repeated or repeated with slight modifications, a bidirectional LSTM would be effective in learning their patterns. The model uses singular bidirectional LSTMs rather than ones stacked on each other, as following the LSTM is an attention layer. The attention layer is more computationally efficient due to its parallelizability. Moreover, the layer allows the model to be more coherent with its predictions as the attention layer will have access to the entire sequence. Finally, following the attention layer is a dropout layer for regularization. These three comprise a cell.

Following the N cells are two affine layers, followed by a softmax to predict the probabilities for the next note, for which cross-entropy loss is used.

2.2. Baseline

2.2.1 From-context

To measure the performance of our model, we will compare the music it generates to the dataset i.e. solos by jazz musicians. This will ensure we can determine if model is, at least in the metrics we choose, able to capture certain aspects of jazz soloing.

2.2.2 From-scratch

As with the from-context model, given that this model attempts to generate models in the J-pop genre, we use actual J-pop MIDI data as a baseline for evaluation.

If our models and the solos/compositions have similar scores for our evaluation metrics, then the models are effectively able to emulate the solos/compositions.

2.3. Evaluation

2.3.1 From-context

To evaluate the performance of the jazz model, we will compute the inside ratio, to measure how well the melody matches the harmony. Inside playing refers to playing notes in the melody that exist in the underlying chord for instance. The notes ‘C’, ‘E’ and ‘G’ and present in a C chord. Therefore, if the model generates a note of any of these three pitches during the beats in which a C chord is the underlying harmony, the note would count as an ’inside’ note. Any other note would be an ’outside’ note. Inside notes are more consonant, while outside notes are dissonant and add tension. Thus there should be a presence of both over any chord. Thus, this metric will measure the percent of notes that are inside.

We will be using a common chord sequence in jazz, ‘Rhythm Changes,’ for evaluation. This is for a few reasons. Firstly, several solos exist in our dataset over these chord changes. The greater number of samples can be used to measure a more meaningful baseline score. Secondly, Rhythm Changes is a widely applicable chord sequence, with several songs such as ‘I Got Rhythm’ (Gershwin), ‘Meet the Flintstones’ (Curtin), and ‘Anthropology’ (Parker), conforming to this sequence. Thus, the model being able to perform well on this chord sequence suggests it will be able to perform well on a large array of songs.

2.3.2 From-scratch

Given the genre of our dataset, J-Pop, we recognize that rhythm is key - dance prefigures heavily as a consideration for J-pop composers. More specifically, it is the downbeat of the rhythm that most heavily influences the resemblance of the music to its genre. The downbeat refers to the first and third beat of a bar of four.

To quantify the ability of our model to emphasize the downbeat, we use a metric proposed by Huang et al. [2]: downbeat standard deviation (downbeat STD), which measures the consistency of the rhythm with respect to the downbeat. Thus, a lower downbeat STD would mean the model is more rhythmically consistent.

It should be noted, however, that while we have formulated quantitative evaluation metrics, due to the qualitative nature of music, our evaluation will heavily rely on our qualitative assessment of the results.

3. Dataset

In this section we discuss the datasets our models use and some of the preliminary results.

3.1. Data collection pipeline

3.1.1 From-context

The dataset for the from-context model, the Weimar Jazz Database [3], consists of 435 jazz solos from several subgenres such as modal jazz, bebop and cool jazz, performed by a wide array of artists such as Miles Davis and Art Pepper. It also contains the harmonic content for each song as lists of chords. Additionally, the dataset consists of several solos over the same or similar chord sequences, which will allow our model to learn a greater number of melodic possibilities over the same harmony.

The dataset stores the pitch information for each note as integers that can easily be translated to MIDI. It stores rhythmic information in two forms. First, storing the time into the song at which a note begins and how long the note is held. This information is represented in seconds and is hence continuous. The second form involves storing the beat, the bar, and how many division of a beat into a beat at which the note is played. This form is discrete.

In addition to storing the harmony and melody for several solos, the dataset also stores other pieces of information regarding the solos, such as time signature and chorus count (how many times the chord sequence repeats). This has been be useful when filtering data, but is ultimately unused for the model itself.

3.1.2 From-scratch

The dataset for the from-scratch model consists of 512 scores in the J-Pop genre obtained from Musescore. In larger datasets like the Lakh dataset, the midi files may have multiple instruments and contain multiple genres of music, so to simplify processing and possible noise during training, we decided to use the Musescore data.

The reason we use music of a certain genre to train is because introducing multiple genres of music will likely introduce a lot of noise in the network as different genres will have widely different rhythms and notation. This approach has seen considerable success in the past across multiple genres of music, highlighting the strong differences by genre for generation.

3.2. Data pre-processing

For the from-context model, to simplify the dataset, we decided not to include any songs with a time signature other than $4/4$. Additionally, due to memory constraints, we limited the ’chorus count’ i.e. how many times a chord sequence is repeated, to 7 or less.

As previously mentioned, the rhythmic information is represented in two forms: one continuous, and one quantized. We chose to use the continuous data and quantize it ourselves. This was done to ensure we could quantize it to a precision we believed was appropriate, which in this case, was $1/24^{th}$ of a beat. This would also allow us to represent rests explicitly, and capture the 'swing' feel of jazz better.

Thus, a note is represented by a vector consisting of two elements. The first is an integral value, storing the pitch, where, for instance, $6o$ represents a C4. Incrementing or decrementing the pitch by 1 corresponds to a step up or step down the chromatic scale respectively. A pitch of 0 refers to a rest. The second element is a floating-point value, storing the duration of the note in relation to a beat. For instance, 0.5 refers to half a beat and 4 refers to four beats.

For the from-scratch model, each score follows a certain structure where key and time signature act as guides, limiting the number of possible combinations of notes/accidentals to a smaller set. Our approach is to encode each note or chord along with its duration in the music in a string and then append it to a list of all unique notes/chords. For example, we represent a note with pitch C held over two beats as the string "C 2", where C is the note and 2 represents the duration. Across all midi files we end up with a large list of musical notes sequenced like a text corpus.

For both models, we augment the data by transposing every song into different keys. In the case of the from-context model, we transpose every jazz solo, melody and harmony, into all 12 keys. In the case of the from-scratch model, we transpose each MIDI score into five other keys, due to the larger size of each score compared to a jazz solo.

3.3. Preliminary results

The preliminary results of our from-context model involved using a greedy decoder for inference i.e. at any time step, selecting the token with the highest score. This, however almost always resulted in the same note i.e. same pitch and duration, being predicted repeatedly. Sometimes, a sequence of three notes would repeat, but this was not significantly better.

With regards to the from-scratch model, qualitatively, we noticed that with $N = 1$ (i.e. one cell), the model's output is both melodically and rhythmically arbitrary. With $N = 2$, we observed a greater rhythmic coherence, and with $N = 3$, a greater melodic coherence.

3.4. Baseline method evaluation

We begin by evaluating the baselines for both our models:

- From-context: we decided to measure the cohesion between the harmony and the rhythm as the primary quantitative evaluation metric. Our baseline involved running the inside ratio over 14 different songs that used the 'Rhythm Changes' chord sequence, and averaged them. The result was that $\sim 5.35\%$ of notes were inside their respective chords. This makes sense as 'Rhythm Changes' is often used in a Bebop context, where rapid linear and chromatic successions of notes are common, meaning several notes are bound to be outside their chords.

- From-scratch: for this model, we decided to measure the rhythmic cohesion of the music in relation to the downbeats. More specifically, this involves calculating the Downbeat Standard Deviation: the average distance of a note from the nearest downbeat. Our baseline involved running the Downbeat STD over our dataset and averaging their STDs. We calculated the average Downbeat STD for J-Pop songs to be ~ 0.271, or a quarter of a beat.

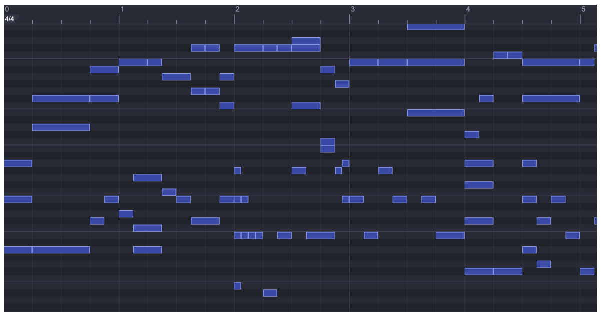

Figure 3. A midi roll representation for a generated piece by the from-scratch model

4. Experiments

4.1. From-context

In order to solve the issue of the model repeatedly choosing the same note, we decided to use random sampling. Once the scores from the final affine layer are calculated, the scores are passed through a softmax for a probability distribution. This distribution is then sampled from at random.

Attempting this, however, resulted in a highly arbitrary sound, and it often sounded entirely unrelated notes were played. Therefore, we applied various non-linear transformations to the probability distribution. Given the limited memory available and the quadratic space complexity of attention-based models, we were limited in our ability to expand the model's size. Even without the limitation, a larger model may learn the distribution even more rigorously, possibly leading to greater repetition. In other words, music cannot be locally optimized as in the case of using a greedy decoder, meaning a probabilistic element needed to be introduced. Therefore, the non-linear transformation is a significant variable that we attempted to optimize. The result for several non-linear transformations are shown below:

- $\text{argmax}(x)$: As a baseline, using argmax (greedy decoding) effectively creates a probability distribution that can only take on one value. Musically speaking, a local view of next-note prediction will almost always yield a repetition of the original (or a stepwise increment) since notes of the same duration tend to follow each other. This results in an output which incessantly hammers the same note, usually decided based on the first chord of the piece.

- Beam search: To counteract this, we attempted to use beam search, but still found that it was too greedy and had the same result. Unlike argmax, beam search would occasionally change which note it was staying on, yet again based generally on chord changes.

- x: We then considered the secondary baseline of random sampling directly from the logits of the model's linear head taken as a discrete distribution over the indices. As earlier mentioned, this sounded tonally arbitrary, but exhibited much greater rhythmic variety.

- $\max(\alpha x, x)$: In light of the output of x, we attempted to use various rectifier transforms to emphasize high-probability indices more. This leaky ReLU was the most successful, producing rhythmic variety with relative tonal consistency.

- e^x: This first nonlinear function illustrates the naive bound on distribution sharpening - it essentially has the same output as argmax, with complete regularity.

- $x\sqrt{\max(x, 0)}$: After trying functions between e^x and x, we eventually came down to sub-quadratic order combined with ReLU, which produced tonally and rhythmically coherent results compared to other transformations such as e^x.

- $x \ln(\max(x, 0) + 1)$: In fact, this slightly smaller function was subjectively assessed as our most successful nonlinear transformation. Output generated with this had a surprising grasp not only of rhythm and tonality but also of motivic development, occasionally integrating previously seen patterns.

- $\sigma(x)$: We also tried various functions of sigmoid nature to smooth out the distribution towards the top and select evenly from the best values while rejecting the worst. In general, such outputs had very high rhythmic variety and were able to use both long and short note values. This varying of length occasionally produced surprisingly realistic and insightful renditions, but also led to long stretches of silence interspersed by single notes.

- $\{x \geq 0 \longmapsto x\sqrt{x}, x < 0 \longmapsto \tanh(x)\}$: We also experimented with piece-wise combinations of the above functions (generally sharpening the top, smoothening the bottom), with this as one of the most successful results generally, having broadly consistent rhythm but a good grasp of tonality.

- $\text{top}_k(x)$: We noted that the general sharpening paradigm was to provide a spread around the top-probability regions, so we reasoned that taking a random distribution over the top k tokens would effectively broaden the greedy paradigm. This can also be thought of as clipping the tails of the discrete distribution. We also applied several of the above functions to the top k tokens, and experimented variously with different values for k.

- $\text{Unif}(\text{top}_k(x))$: As one interesting final note, applying a uniformly random selection over the top k tokens works surprisingly well for small k. We reason that applying smoothening functions over the top k tokens generally emulates the nondeterministic and subjective nature of composition as a semi-arbitrary choice (locally speaking) between many good alternatives. It is easy, heuristically speaking, to reason about *bad* choices to continue a solo, but there are almost always many *good* choices.

Many of the different transforms used are pictured below - we can visually see the sharpening and smoothening effects.

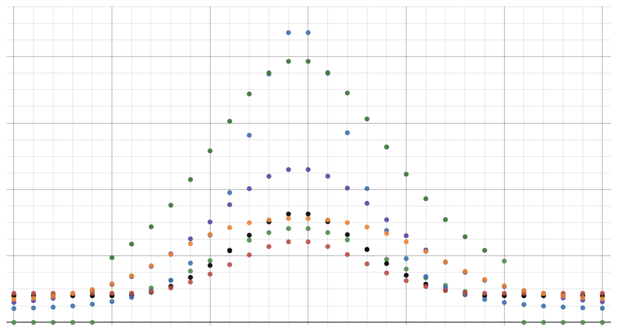

Figure 4. The effects of different nonlinear transforms on data sampled from a Poisson distribution. Our logits, of course, have little such relation between indices and probabilities, but we are interested in seeing how the transformations will act on high-probability areas versus low-probability areas. From highest peak to lowest peak, the blue data is e^x, green is top_{20}(x), purple is x, black is $x\sqrt{\max(0, x)}$, orange is $\tanh x$, the (second) green is $\max(0, x)$, and the red is $x \ln \max(0, x) + 1$.

The final table of results tabulating the from-context inside-outside ratio over the 'Rhythm Changes' chord sequence is shown below. To be specific, we generated 10

Algorithm	Inside ratio
Beam search	0.2296
argmax(x)	0.2296
x	0.2163
e^x	0.0074
top$_{50}$ (x)	0.2170
Unif(top$_{15}$ (x))	0.2881
top$_{50}$ ($x \ln(x)$))	0.2289
tanh(top$_{50}$ (x))	0.2793
$\sigma(x)$	0.0963
$x \ln x$	0.2407
max($0.1x, x$)	0.2296

Figure 5. Inside ratios for various transformation

sequences using each of the decoding strategies using the 'Rhythm Changes' chord sequence as a source. We do want to emphasize that though many models actually fail tremendously in this respect (in relation to the original data), the inside ratio does not adequately capture one of the most essential features for affect, rhythmic cohesion and consistency.

4.2. From-scratch

Since LSTMs are used for this model, memory is not as much of a constraint, as it was in the case of the transformer-based from-context model. Therefore, optimization for this model heavily involved testing various training lengths and model sizes, by taking advantage of the modular nature of a MAR-Y block. The results described below are qualitative, however, we also ran the Downbeat STD evaluation metric on one of the iterations of our model, trained on two epochs on one MAR-Y block, which resulted in a Standard Deviation of ~ 0.263. The baseline score was ~ 0.271, meaning that the rhythmic content is similar. Therefore, in this metric, the model is successful. Qualitatively,

- 1 block for 100 epochs: The resulting music was highly arbitrary.
- 2 blocks for 100 epochs: The results showed that the model had learned, an albeit vague, representation of the key and rhythm for each generated sequence.
- 2 blocks for 150 epochs: The model appears to be learning motifs, by repeating certain melodic and rhythmic patterns throughout the piece.

However, all models appear to turn more arbitrary over the course of a song. This effect is minimized with the 2-block LSTM. This suggests that even larger models may be able to largely mitigate this issue.

5. Conclusion

In case of the from-context model, it is clear that performing better in the quantitative evaluation metric does not mean the music sounds better. The models that were moderately sharpened, such as the ones that used $x\sqrt{x}$ and $x\ln(x)$ were the most pleasant to hear. In the case of the from-scratch model, a larger model, i.e. with a greater number of blocks can result in a lowering of the arbitrary nature of the music generated.

Regarding improving the from-context model, most evidently, we could make the model larger, by using a larger number of decoder/encoder layers. Additionally, rather than tokenizing individual notes, we could tokenize common melodic fragments, more commonly known as 'licks'. We could also encode heuristics into the model, such as restricting the notes to choose from when a certain chord is underneath. Finally, we could change the representation of the harmony by perhaps using bass notes rather than chords, providing a degree of counterpoint (multiple melodies being played simultaneously) to the music in addition to harmony. Improving the from-scratch model, we could incorporate more intricate MIDI arrangements, and start incorporating a greater palette of textures by using other instruments in addition to the piano, which is something the model could learn as well.

References

[1] Shunit Haviv Hakimi, Nadav Bhonker, and Ran El-Yaniv. Bebopnet: Deep neural models for personalized jazz improvisations. In *Proceedings of the 21st International Society for Music Information Retrieval Conference (ISMIR)*, 2020. 2

[2] Yu-Siang Huang and Yi-Hsuan Yang. Pop music transformer: Generating music with rhythm and harmony. *CoRR*, abs/2002.00212, 2020. 2, 4

[3] Martin Pfleiderer, Klaus Frieler, Jakob Abeßer, Wolf-Georg Zaddach, and Benjamin Burkhart, editors. *Inside the Jazzomat - New Perspectives for Jazz Research*. Schott Campus, 2017. 4

[4] Sigurur Skúli. How to generate music using a lstm neural network in keras. 2017. 2, 3