Spring 2021

Impromptune: Symbolic Music Generation with Relative Attention Mechanisms

Connor J. Lennox
University of New Hampshire, Durham

Impromptune: Symbolic Music Generation with Relative Attention Mechanisms

Connor Lennox

May 2021

Abstract

By combining attention-based mechanisms that have proved beneficial in the field of natural language processing with domain-specific knowledge about the structure of music, better predictions about piece continuations can be made. The goal of this work is to adapt current natural language processing techniques to a musical domain, and to generate new music by predicting continuations on a sequence of notes. An adaptation of traditional attention mechanisms to create a single prediction from sequential input is used to extend musical pieces by appending new elements repeatedly.

1 Introduction

Music is, in most contexts, structured with motifs and repetitions driving the overall form of a piece. Through most musical pieces, both long-term and short-term patterns can be used to predict how the piece will continue as time goes on. Recent work in the field of natural language processing focuses primarily on text analysis and generation, however some concepts within text structure mirrors that in music. Music structure is almost entirely driven by self-referential relations to prior sections of the piece. In order to best take advantage of these relations when attempting to predict continuations on a stub piece, a model must potentially consider all prior elements of the sequence.

The problem is music generation is to create an extension for a given "prompt". A good extension is one that sounds pleasant to a listener while also following traditional ideas of music theory: for example, notes within the prompt will define the key signature of a piece, and it is unusual (but not unheard of) for pieces to contain many notes that do not belong to their key signature. Musical aspects such as tempo should also be maintained, as while pieces do frequently change tempo it is something that usually happens gradually over time as opposed to rapid, sudden changes. As there are many complex facets to music generation, it is important that all possible information is gleaned from the prompt before predictions are made. In this context, every note in the prompt must be considered to make a well-informed prediction.

While there are several types of models that are capable of considering all elements of an input sequence, care must be taken as to not lose information from the start of a piece as the piece continues. The reason for this distinction stems from the usual structure of classical

pieces, which is what this work primarily studies. It is very common for pieces to define motifs and overall moods at the beginning of the piece, and so it is likely that predictions very far in the future will still need to reference these far back elements in order to have well-informed results.

Recurrent models are capable of parsing inputs that are of arbitrary sequence length, however as sequences get longer it becomes more likely that information from the very start of the sequence is lost [4]. As such, they are not well suited to handling the very long sequences used to describe music. Attention-based models are capable of assigning attention weights to any element of the input sequence, and as such have a significantly lesser risk of losing early sequence information as sequence length increases. Due to this property, attention-based models are better suited for the prediction task in this work.

2 Prior Work

The field of symbolic music generation has seen great improvement in the last few years with the advent of the Transformer network [6]. This model was proposed in order to allow better work on Natural Language Processing problems, however it was also found to be helpful with music generation tasks [2]. This may be in part because natural language and music share many structural qualities, including a highly important temporal component and differentiation of short/long-term relationships between elements. Prior to the usage of Transformers, Oore et. al. [4] used an LSTM for the same purpose, but had issues with continuity in their pieces, self described as a pianist who "knows very well how to play, but has not yet figured out what they want to play, nor is quite able to remember what they just played".

Attention-based methods of generating music have also been performed, and mirror the attention mechanisms of the Transformer [2]. This method achieved impressive results, especially compared to the earlier recurrent approaches of the same problem.

Beyond neural networks, other statistical methods have also been used for music generation. Lavrenko and Pickens model music using Random Fields [3], and are able to generate polyphonic music by modeling the probability distribution of notes at a future timestep.

All of these works (with the exception of the Random Fields approach) use a data representation that is event-driven, as opposed to note-driven [4]. The benefit of this representation is that it considerably simplifies the prediction state space from a two-dimensional piano-roll to a one-dimensional sequence of events. This work also takes advantage of this more efficient data representation, which will be described in more detail in a later section.

3 Relative Attention

In traditional query-key-value (QKV) attention layers, three transformation matrices are used to derive a vector query, key, and value from the input. For any given pair of input values, a compatibility score is calculated as in equation (1). In this equation, d_v is a normalization factor that is equal to the square root of the dimension of the final output values (a hyperparameter).

$$c_{ij} = \frac{(x_i Q)(x_j K)^T}{\sqrt{d_v}} \tag{1}$$

Once compatibility scores are calculated for every key and a single query, a value vector is calculated for each input element as shown in equation (2). Here, s_{ij} is equal to the softmax of the compatibility in the context of all compatibilities corresponding the the element the value vector is being calculated for (equation (3). This process is done for all queries simultaneously using tensor multiplications, and may be repeated several times with a given input and differing QKV transformation matrices (this technique is called *multi-headed attention*, and it allows more than one unique relation to be found in the data).

$$v_i = \sum_j s_{ij}(x_j V) \tag{2}$$

$$s_{ij} = \frac{e^{c_{ij}}}{\sum_j e^{c_{ij}}} \tag{3}$$

However, one flaw of this form of attention is its lack of concern for spatial relations between inputs. Since the compatibility between any two sequence elements is driven solely by the corresponding key and query with no respect to the actual position of the elements in the sequence, the final result is not informed at all by the relative distance between the elements. An attempt to alleviate this was proposed by Vaswani et. al. in the original paper outlining the Transformer model [6], and works by either overlaying or appending several sine and cosine functions of varying period over the input data prior to attention being applied. Since sine and cosine functions are continuous, this provides a position-based structure for the model to gain positional information from. In optimal cases, the model learns to differentiate between the actual content of the input values and the additional temporal components, leading to better informed results. There is, of course, no guarantee that this will be the case. It is quite possible that having additional sine waves stacked on top of actual inputs leads to muddled results, as the model may struggle to properly interpret each portion of the input. In addition to introducing this potential problem, overlaying additional synthetic features over input data is a technique that does not mirror any other in machine learning.

As an alternative to this method of overlaying temporal features, Shaw et. al. propose that a model can instead learn to directly use the differences in distance between two input elements to adjust its attention weights [5]. The impact of this is that compatibility between sequence elements is informed by both the elements themselves and their positions in the sequence. In addition to learning the standard QKV transformation matrices, the attention mechanism also learns secondary key and value matrices that are based around the relative difference in position between two elements as opposed to the content of the elements themselves.

4 Data Representation

MIDI files provide a convenient way to store music data in a discretized form. The MIDI specification allows for "note on" and "note off" events, among others. Notes are represented

by a pair of events: one to start the note and one to end it. By parsing a MIDI file a format called piano-roll can be made, which is a 2d array marking the velocity (volume) of a given note at a given moment in time. This piano roll can then be queried for a given time in the piece to know what notes are active.

In order to convert the data from MIDI into a format more suitable for machine learning, a relatively simple process is used. First, the MIDI file is read into a discretized piano roll format with a fixed time delta of 8 ms. This ensures that there is a suitable level of detail to capture fast elements present in the performances. Then, the piano roll is read sequentially to identify onsets and offsets of notes, as well as pauses between note events and changes in note velocity. The final format for learning purposes is sequential data with four event types, encoded as one-hot vectors. The different event types are described in Table 1. Each event type has many values that correspond to it, each representing a different actual event. The note onset and offset events each correspond to a specific pitch value, time steps correspond to different durations of delay (in 8 ms increments, up to approximately a quarter second), and velocity changes correspond to the new velocity value. Pitches considered are only those that fall within the bounds of a standard 88 key piano, as opposed to the 128 pitches defined in the MIDI specification. With all event types defined, there are a total of 240 unique events that can occur in sequences.

This data representation is a condensed form of that proposed in Oore et. al. [4]. In that work, time steps are allowed up to an entire second and all 128 pitches in the MIDI specification are available. In that case, there are 413 possible events. However, the upper three-quarters of the time step events are hardly ever used, and since this model is geared specifically to solo piano music there are no notes that land outside of the standard 88 keys of a piano. As such, reducing the event space does not lose any information from the data, and leads to better prediction as a significant portion of the events are no longer considered.

Event Type	**Numerical Range**
Note Onset	0-87
Note Offset	88-175
Time Step	176-207
Velocity Change	208-239

Table 1: Event types and the numerical range that corresponds to that type. Events within a range refer to different forms of the event type.

5 Model Design

Several factors influenced the design of the Impromptune model. It was deemed critical that some form of attention be applied to the sequence, as it is logical that there is a benefit to be gained when viewing each sequence element in the context of the sequence itself. By applying attention to the sequence, every event in the input sequence has additional information of how it pertains to the structure of the sequence as a whole. In addition, the relative positioning of sequence elements must have some impact on their compatibility when calculating attention. This is important, as two events that seem to have high compatibility

may actually be so far separated in the sequence positionally that they do not actually have any relation to each other. Finally, the model will produce a single prediction from a given input sequence. This will lead to an autoregressive behavior, where singular output events are continuously appended to the input sequence, which is then fed back into the model, until a desired target length is reached.

5.1 Impromptune Model

The Impromptune model uses attention mechanisms to create a single-element prediction to continue a given input sequence. The model begins by embedding a sequence of events into a one-hot representation. This step is critical as events that are adjacent in the event space are not necessarily related to one another (that is, the events are 240 distinct labels as opposed to a single continuous value). The one-hot embedding removes the implied relation between events that are nearby each other in the event space.

Then, two relative attention layers are applied to the embedding. After each of these layers, a ReLU activation function is used to apply non-linearity to the result. Both of the relative attention layers are global attention layers, which calculate compatibilities for all sequence elements. This stack is followed by the Predictive Attention layer, which collapses the sequential input into a single vector. The mechanism behind this collapsing procedure is explained in the next section. Finally, this vector is projected into the 240-length event space using a learned projection function. These final values, when normalized via softmax function, represent the probabilities that any given event follows next in the sequence.

A slight alternative on the model changes the second global attention layer for a local attention layer. The main difference between global attention and local attention is in the calculation of compatibility. In the global attention mechanism, all sequence elements have compatibility calculated with all other sequence elements. On the other hand, local attention forces the compatibility between distant elements to be 0. This effectively masks out compatibility from sequence elements that are too far away, allowing information to be gained only from the local neighborhood around a sequence element. While locality is already introduced via predictive attention, having more model emphasis on elements that are nearby one another within the sequence has a minor beneficial impact on long-term prediction (however tends to cause slight declines in short-term prediction quality).

Generation with the Impromptune model works autoregressively: a single output element is generated at a time, and is appended to the input. This new input sequence is then fed back into the model, which continues to provide outputs until a fixed final sequence length is reached. The actual process of selecting the next event from the model output (240-length vector of probabilities) can be achieved either deterministically or stochastically. In the deterministic case, the most probable event is taken. In the stochastic case, the next event is sampled from the top-k most probable events. The weights for the sample are equal to the predicted probabilities, so in the event where the model believes a single event to be undeniably correct it is overwhelmingly likely to be the one that is sampled.

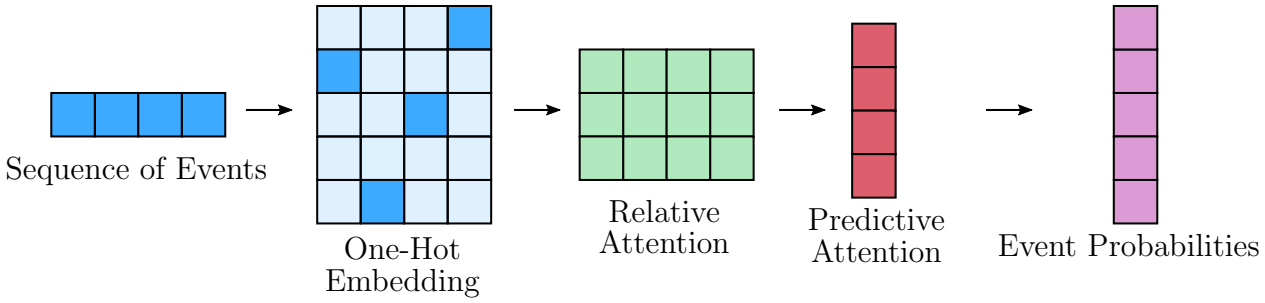

Figure 1: An overview of the Impromptune model design. Input comes in a sequence of integer-based events, and is first converted into a one-hot matrix representation. Both relative attention layers are applied (only one is pictured here), before predictive attention collapses the input sequence into a single vector. This vector has a linear transformation applied to bring it to the dimensionality of the event space, such that values represent the probability of a given event.

5.2 Predictive Attention

In most implementations of relative attention mechanisms, every input element has a corresponding output element. This works well in sequence-to-sequence contexts such as text translation, as when encoding an input sequence every element should have a new representation. However, this is less desirable in a predictive context, where an entire sequence of input is consumed to produce a singular output. Instead of each input element having a corresponding output element, a single output element should be generated that combines all provided inputs. This is easily done by recurrent structures such as an LSTM, but recurrent networks in general have a two-fold problem in this context. First, the time it takes both to train and evaluate inputs increases linearly with sequence length due to the need for hidden states to be fully calculated before the next input element is consumed. In an attention-based network, GPUs can parallelize the calculations for all input elements and as such do not have this limitation. The second issue is their tendency to have difficulties carrying early information through long sequences. Since early motifs will likely influence decisions much later in a piece, forgetting early information is a poor design feature.

In order to create an attention-based layer that still produces a single output element, the Predictive Attention layer is proposed. This system works similarly to standard QKV attention, however the source of the query differs from regular implementations, and there is an additional final step that flattens the output into a single element (desirable for prediction-based problem spaces). The query is generated by slicing final elements from the input sequence and multiplying by a fixed size matrix parameter. The number of elements to slice is determined by a hyperparameter, said to be the *lookback length* of the prediction In equation (4).

$$q = x_{i:end}Q \tag{4}$$

Once the query have been created, a key-value pair is generated for each input sequence element (as per usual QKV procedure), and compatibility is calculated between this single

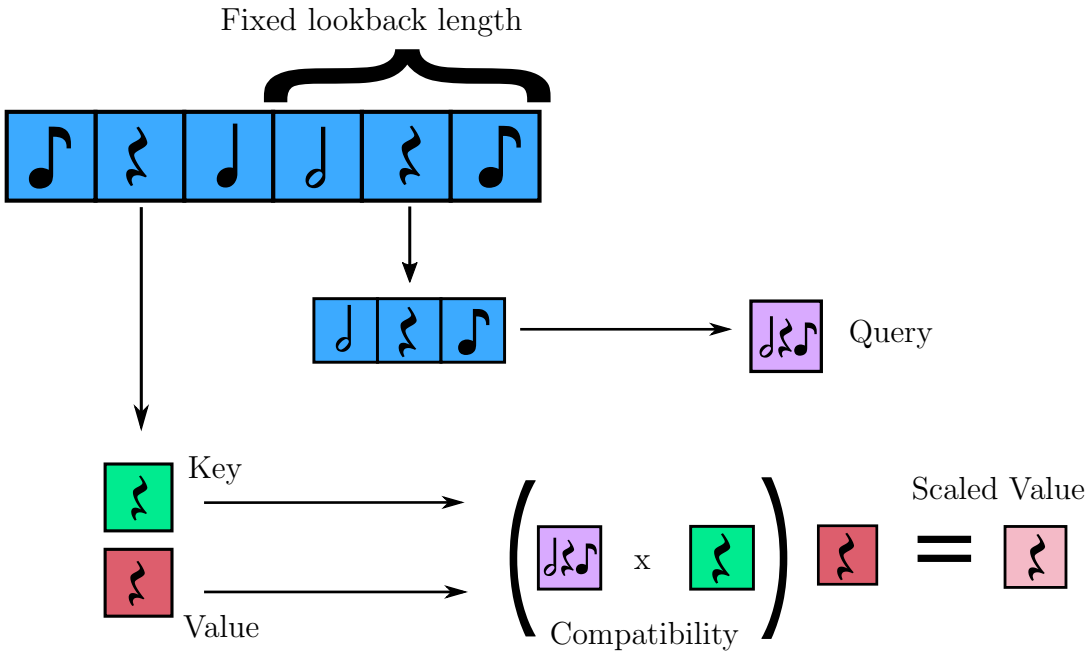

Figure 2: The predictive attention process visualized. A fixed number of elements are extracted from the end of the sequence before being transformed into a query that is applied to the key/value pairs of every element in the input sequence. The values are scaled by the product of the query and key, and these scaled values are then summed to produce a final value.

query and the key corresponding to each sequence element, much like standard attention. Once compatibility for each sequence element is found, they are used to weight the values, which are then summed. This procedure is a weighted average of all values, where the weighting of the element is determined by how compatible it is with the current tail of the input sequence. A softmax function is also applied over the compatabilities, similar to how it is applied in the regular attention calculation.

In order to gain some relative positional information in this mechanism, a second query is generated. This second query is created by a different learned matrix, and serves as the query for position-based compatibility between elements. True compatibility between two elements is informed both by the query/key relation mentioned earlier, as well as the relation described by this relative attention matrix.

$$v_{pred} = \sum_i s_i(x_i V) \tag{5}$$

$$s_i = \frac{e^{c_i}}{\sum_i e^{c_i}} \tag{6}$$

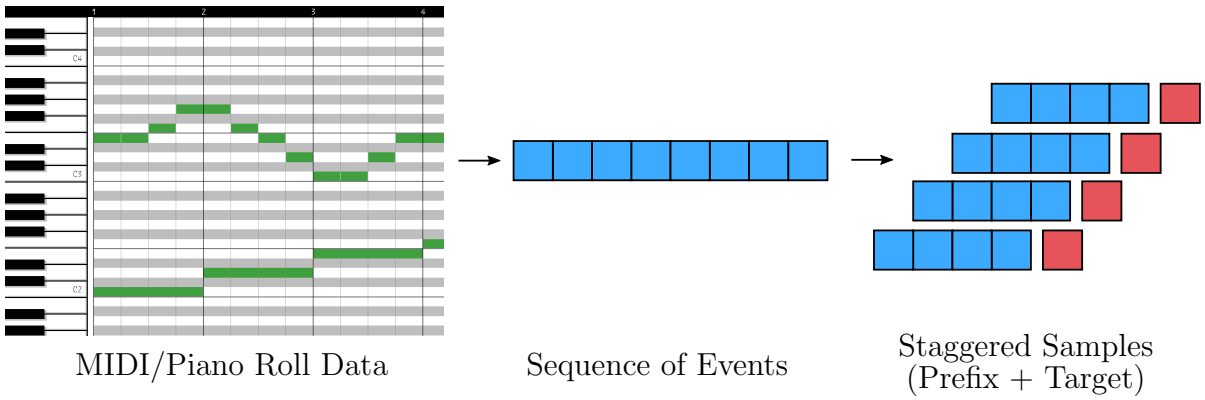

Figure 3: A visualization of the sample generation process. MAESTRO data is read from MIDI into piano roll format, and then split into 30 second sequences of events. Samples are extracted from this event sequence by sliding a fixed width window over the sequence and taking the next event after the window as the target event for prediction.

6 Evaluation Paradigm

The dataset used for this project is the MAESTRO dataset [1]. This dataset consists of approximately 200 hours of MIDI data, comprised of performances that were recorded as part of the International Piano-e-Competition. This dataset was selected due to its large volume of content, its well-controlled recording process, and single-instrument nature. The MIDI files themselves have a very high resolution, true to the original performances with ˜3 ms accuracy.

From the MAESTRO data, only a subset is used. Due to the large amount of data, it was found not feasible to use all 200 hours. Instead, a subset of the data was selected to encompass only the data from 2017. There are 140 performances included in this subset, representing about 11% of all performances included in the full MAESTRO dataset.

The selected MIDI files were converted into piano roll format with 8 ms resolution. These piano rolls were then split into segments of 30 seconds, and then each 30 second interval was converted into a sequence of events as stated in the Data Representation. From these 30 second intervals, prefixes of any target length could be extracted for training. In order to maximize the usage of the training data, a sliding window would be applied over the sequences while generating samples. This process is visualized in Figure 3.

Evaluation of model performance is done with mean reciprocal ranking to a depth of 10 predictions into the future. For a given prompt and target, points were given based on the position of the correct events in the list of events sorted by predicted probabilities. For example, if the model places the correct continuation event as the fourth most likely, it receives $\frac{1}{4}$ points. Scores were calculated for each prediction depth separately, to show how performance reacts when making predictions farther into the future. The score of a model at a given prediction depth is equal to the average score it receives for that position over all evaluation sequences. Due to the nature of this metric, a higher score is preferred.

7 Results

Three different models were compared to judge the efficacy of the attention mechanisms described above. The first two models are the Impromptune model itself and the variation that makes use of a local attention layer in lieu of the second global attention layer. The third model to be compared is an LSTM implementation that performs the same classification task by sending its final hidden state through a linear transformation.

The mean reciprocal ranking procedure was done with 300 input sequences of 500 elements and generated 10 events into the future. After this point all models tended to converge in terms of performance, with some slight variations that mirror the differences in quality in earlier predictions.

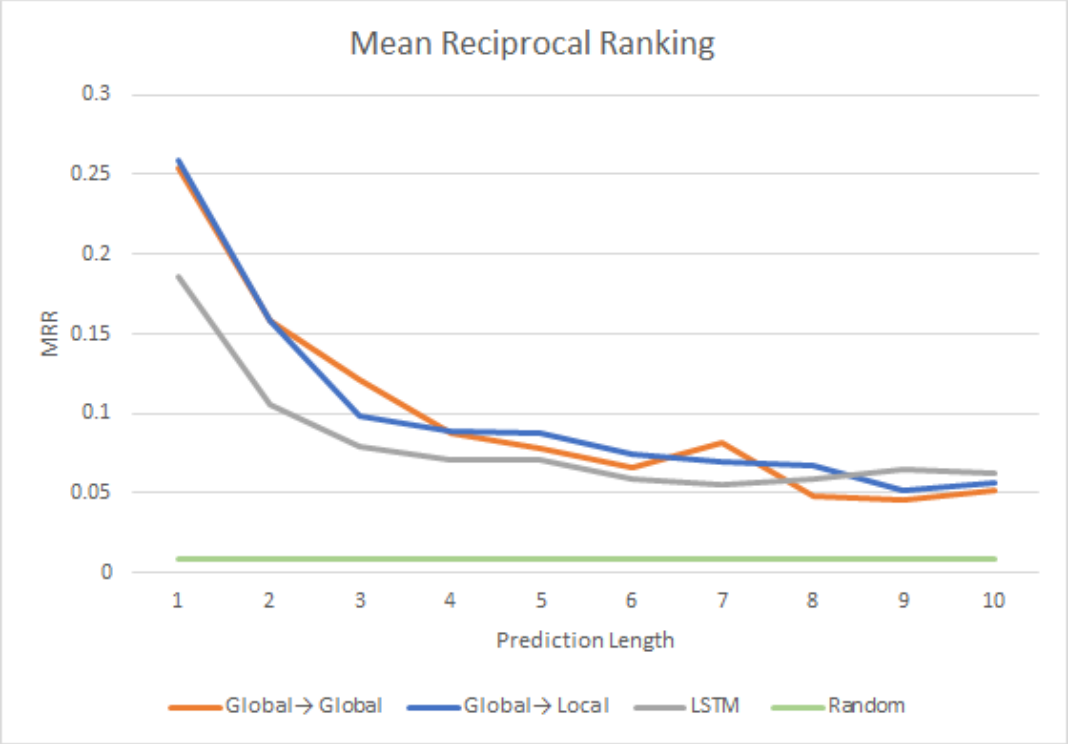

Figure 4: Results of the mean reciprocal ranking trials. The Impromptune model, in orange, uses two global attention layers to glean information from the input sequence. The Impromptune variant, in blue, drops the second global attention layer in favor of a local attention layer. The gray line is an LSTM implementation of the same prediction task, which generates predictions by transforming the final hidden state of the model.

From the results in Figure 4, it is clear that the attention-based models surpassed the performance obtained by the recurrent LSTM network. With a single prediction out from the input sequence (the problem with which the models are trained), the Impromptune models achieve a roughly .26 MRR score, while the LSTM model only scores a .19. Harsh downward trends are present for all models, as predicting elements further away from the input sequence becomes more difficult as the prediction length increases.

The results of this trial highlight the compounding error problem, which was prevalent

through much of the work. Since all intermediary results are added to the input, a single incorrect classification has already introduced information to the input sequence that might not necessarily be beneficial to the prediction process. The only way to minimize this is to ensure that the model creates correct predictions as often as possible, such that these errors are not introduced to the input.

Even though MRR scores have significant drop-off over long prediction spans, it is important to note that there is not a single "correct" extension to a stub of music, contrary to the implication made by using similarity to the original piece for performance evaluation. Music has some inherent stochasticity to it, and so it may be that the models are predicting sequences that are logical continuations but do not match what is expected as per the dataset.

In order to compare the models without considering a ground truth, a qualitative evaluation was done on pieces generated by both models. 10 pieces were sampled from each model, with random prefixes from the test data. The models were then empirically evaluated from 1-5 on four criteria:

1. **Boundary Distinction (BD):** If there is a clear divide between the prompt and the generated music, a low score is given. If the boundary is difficult to discern or is not immediately obvious, a high score is granted.

2. **Harmony (HA):** A measure of how harmonious the generated music was in comparison to both the prompt and other sections of the generated sequence. A more harmonious piece will have a more coherent feel overall.

3. **Repetition (RE):** A high repetition score is given to pieces that both have long-term structure, especially when this structure is present relative to the prompt. A low score is given to pieces where a single note or pair of notes is rapidly repeated.

4. **General Quality (GQ):** A relatively subjective measure, showing the quality of the generated piece. The pieces were evaluated in relation to each other, so a 5 on this scale represents the best generated pieces while a 1 represents the worst.

The two models used for the qualitative analysis are the primary Impromptune model (with two global attention layers) and the LSTM implementation that was used for the quantitative analysis above.

Model	**BD**	**HA**	**RE**	**GQ**	**Average**
Impromptune	3	3.3	3.7	3.3	3.325
LSTM	2.1	2.4	2.6	2.1	2.3

Table 2: Qualitative Analysis results for the Impromptune and LSTM models. The Impromptune model outperforms the LSTM in all categories, with an average score an entire point (on a 1-5) scale higher.

8 Conclusion

It has been shown that attention-based predictions can be used to extend sequences with strong temporal relations. The usage of the predictive attention mechanism has allowed superior performance in both quantitative and qualitative measures relative to recurrent methods of solving the problem of music prefix continuation. The proposed Predictive Attention mechanism is novel and serves as a method of condensing an input sequence while leveraging domain knowledge. Although this method of attention was demonstrated in the domain of music generation, it can be applied to other domains should the same assumptions regarding relevance of sequence endings hold.

References

[1] Curtis Hawthorne, Andriy Stasyuk, Adam Roberts, Ian Simon, Cheng-Zhi Anna Huang, Sander Dieleman, Erich Elsen, Jesse Engel, and Douglas Eck. Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International Conference on Learning Representations*, 2019.

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

[3] Victor Lavrenko and Jeremy Pickens. Polyphonic music modeling with random fields. In *Proceedings of the eleventh ACM international conference on Multimedia*, pages 120–129, 2003.

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

[5] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations. *arXiv preprint arXiv:1803.02155*, 2018.

[6] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.

A Complete Qualitative Results

10 pieces were evaluated for each of the models in the qualitative analysis, but for brevity's sake were not included in the main paper. The full results are available here. During the course of the trial, the pieces were scrambled and the true origin of each piece was hidden. This was done to reduce bias in the results, as the scores were determined by listening to the pieces. Had the origins of pieces not been hidden, it is likely underlying biases about model performance would skew results.

Piece	**Model**	**BD**	**HA**	**RE**	**GQ**	**Average**
1	Impromptune	1	2	3	2	2
2	Impromptune	4	3	3	4	3.5
3	Impromptune	2	3	4	3	3
4	Impromptune	4	5	4	4	4.25
5	Impromptune	1	2	3	2	2
6	Impromptune	3	2	3	2	2.5
7	Impromptune	4	3	3	3	3.25
8	Impromptune	4	4	4	3	3.75
9	Impromptune	4	5	5	5	4.75
10	Impromptune	3	4	5	5	4.25
11	LSTM	4	4	2	2	3
12	LSTM	1	3	2	2	2
13	LSTM	3	4	1	3	2.75
14	LSTM	1	1	3	1	1.5
15	LSTM	1	1	3	2	1.75
16	LSTM	3	3	4	3	3.25
17	LSTM	1	2	4	2	2.25
18	LSTM	3	2	2	2	2.25
19	LSTM	2	3	2	2	2.25
20	LSTM	2	1	3	2	2

Table 3: Full qualitative analysis of pieces, used to generate averages for main reporting.