Separating Spaces in Relative Attention for Music Generation

Michael Conner, Jonathan Keane, Josiah Yoder
EECS
Milwaukee School of Engineering
Milwaukee, WI 53203
{connerm, keanej, yoder}@msoe.edu

March 19, 2023

Abstract

Songwriters rely heavily on timing when creating structure to their music. Not just absolute timing but the relative distances between notes, motifs, phrases, and more are all important when creating a song. In current state-of-the-art transformers for music generation, a variant of self attention which efficiently captures relationships between tokens based on relative distance is used. The efficient method helps overcome challenges to memory requirements which restricted possible sequence lengths when relative relationships were first proposed. However, this method leaves too much room to get lost in the latent space of the attention matrix as the relative positional information is added straight into the attention matrix. We propose alternative methods for integrating the relative positional information instead to help keep a clear separation between semantic and positional information. While the complexity does increase with the proposed solution, it is still requires much less computation than the quadratic requirements of the original relative self attention system. We trained and quantitatively evaluated our transformer on the Piano-e-Competition dataset as well as qualitatively with human evaluations.

1 Introduction

When writing music, songwriters rely heavily on the timing between different musical elements, and often need to look far back at previous sections of a song in order to iterate and alter those past sections. In the past, recurrent neural networks have been used in artificial music generation, however they are restricted to storing all past elements in a fixed size state which does not lend itself well to long sequences like music or generalizing to sequences of any length. More recently, transformers have been used for this task and have show impressive performance compared to their RNN predecessors. Huang et al. [2] for instance, have achieved state-of-the-art results using a transformer with relative positional information added to the attention matrix in a similar fashion to [5].

While there are great results achieved by [2], they rely on the same latent space to capture both the positional and semantic information for the pairwise relationships between elements. We believe that this space will get too complex and confusing with the two different types of information pushed into that space. Therefore, we propose an alternative implementation for the relative information in order to keep the different types of information in completely different spaces to be mixed later in more complex ways than addition.

When the original transformers were introduced with global positional information added to the input embeddings, many attempted, and acheived better results, by concatenating the positional information to the inputs instead. This was done for the same reasoning as we stated above and serves as the inspiration for this work.

For data, the Piano-e-Competition dataset consists of human played piano performances represented as MIDI. As the performances are human, there are expressive dynamics which have very fine timings. Therefore, following [2], we use a similar representation to the one proposed by [4] where MIDI events are expressed as note-on, note-off, velocity change, and time change events and are then put into one-hot vectors.

2 Previous Work

2.1 Multi-Head Attention

An attention head takes in an input sequence $X = (\mathbf{x}_1, ..., \mathbf{x}_n)$, $X \in \mathbb{R}^{n \times d_x}$ of n elements where $\mathbf{x}_i \in \mathbb{R}^{d_x}$ is a row vector, and creates a new sequence $Z = (\mathbf{z}_1, ..., \mathbf{z}_n)$, $Z \in \mathbb{R}^{n \times d_z}$ where $\mathbf{z}_i \in \mathbb{R}^{d_z}$ and each row$\mathbf{z}_i$ is a weighted sum of a linear transformation of an input row:

$$\mathbf{z}_i = \sum_{j=1}^{n} \alpha_{ij}(\mathbf{x}_j W^V) \tag{1}$$

The weight coefficients α_{ij}, are computed via a softmax:

$$\alpha_{ij} = \frac{\exp e_{ij}}{\sum_{k=1}^{n} \exp e_{ik}} \tag{2}$$

The element $e_{ij}$is computed using a matrix multiplication of two linear transformations of input elements and normalized with $\sqrt{d_z}$, also known as a scaled dot product:

$$e_{ij} = \frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K)^T}{\sqrt{d_z}} \tag{3}$$

Because $\mathbf{x}_i$ and $\mathbf{x}_j$ are row vectors, this is an inner product.

The parameter matrices $W^Q, W^K, W^V \in \mathbb{R}^{d_x \times d_z}$ are learned and unique for every attention head. Each head can be seen as getting a section of the embeddings, and heads learn to attend tokens to each other differently based on the section of data that they receive. Later, the data is aggregated back together to get single positions to attend.

2.2 Relative Self-Attention

The self-attention mechanism proposed by [5] alters the preexisting attention mechanisms in order to allow information on relational distances between items to be encoded directly into the attention matrix. They model the input as a labeled, directed, fully-connected graph where the edge between two input elements x_i and x_j is represented by the vectors $\mathbf{a}_{ij}^V, \mathbf{a}_{ij}^K \in \mathbb{R}^{d_a}$ where $d_a = d_z$. They use two different edge representations so that one can be used to modify eq. (1) and the other to modify eq. (3). Unlike the parameter matrices, the edge representations are shared across all attention heads. As they are meant to capture information about the relative distances between elements, it makes sense that they should be consistent in all scenarios for any context. In order to inject these representations into the attention matrices, they propose adding them to the results of their corresponding linear transformations $\mathbf{x}_j W^V$ and $\mathbf{x}_j W^K$ in eq. (1) and eq. (3) respectively. This results in the first new formula:

$$\mathbf{z}_i = \sum_{j=1}^{n} \alpha_{ij}(\mathbf{x}_j W^V + \mathbf{a}_{ij}^V) \tag{4}$$

The authors hypothesize that the relative information is important in the above equation for tasks where edge information is useful to future layers for information already being attended to by a given head, however it is the less important of the two places where edge information is injected.

This also results in the second, more important, new formula:

$$e_{ij} = \frac{(\mathbf{x}_i W^Q)(\mathbf{x}_j W^K)^T + (\mathbf{x}_i W^Q)(\mathbf{a}_{ij}^K)^T}{\sqrt{d_z}} \tag{5}$$

The idea behind adding the relative information here is to use that information when determining compatibility of queries and keys. They state the motivation

for using simple addition in these cases is for an efficient implementation. Note as well that while this equation could factor out $(\mathbf{x}_i W^Q)$, multiplying before the addition eliminates the need to broadcast the relative position information. While this is more efficient, it is still quite costly with a space complexity of $O(n^2 d_x)$ as pointed out and improved by [2].

2.3 Memory Efficient Relative Self-Attention

To improve upon this, [2] first dispose of the relative position representation used for values (Eq. 4) entirely as [5] was unable to prove a significant improvement while including it with the query relative representations. Huang et al. [2] thus propose the new relative attention function:

$$RelativeAttention = Softmax(\frac{QK^T + S_{rel}}{\sqrt{d_h}})V \tag{6}$$

where $S_{rel} = QR^T$, $R_{ij} = \mathbf{a}_{ij}^K$, $Q = XW^Q$, $d_h = \frac{d_x}{h}$, and $K = XW^K$,$V = XW^V$ are their respective linear transformations. To reiterate, $Q \in \mathbb{R}^{h \times n \times d_h}$, $R \in \mathbb{R}^{h \times n \times d_h}$, and $S_{rel} \in \mathbb{R}^{h \times n \times n}$, where h is the number of heads and R is the matrix of relative positional representations. The benefit of using S_{rel} is that the memory requirements go from $O(n^2 d_x)$ to $O(n d_x)$, additionally, while the time requirements remain the same on paper, the authors found that there was about a 6x speedup as well. Huang et al. [2] achieve this by creating what they call the 'skewing' procedure.

The skewing procedure performs the operation $QR^T$where R is a matrix representing the embeddings of the relative positional information. While on paper the details of this and the original relative self attention look the same, [5] uses an implementation which adds an extra dimension to the data as an intermediate step in the computation of S_{rel}. As shown below, Q and R are matrices where each row is representing a query in the case of q, and a relative positional embedding in the case of r, r_{-2} denotes an embedding for the relationship between a query and the value that is two behind it relatively. After computing QR^T, the values that relate queries to non-existent values due to looking too far behind are removed with an upper left mask. These positions will eventually end up in the location where the look ahead mask in regular attention is. Now the task is to efficiently shift each row so that it's masked values are on the right instead of the left. In order to do this, one can pad a column onto the left of the matrix, reshape the matrix from $(n+1) \times n$ to $n \times (n+1)$, and slice out the desired lower portion of the result in order to get the final value of S_{rel}.

$$Q = \begin{bmatrix} q_1 \\ q_2 \\ q_3 \end{bmatrix}$$

$$R = \begin{bmatrix} r_{-2} \\ r_{-1} \\ r_0 \end{bmatrix}$$

$$QR^T = \begin{bmatrix} q_1 r_{-2} & q_1 r_{-1} & q_1 r_0 \\ q_2 r_{-2} & q_2 r_{-1} & q_2 r_0 \\ q_3 r_{-2} & q_3 r_{-1} & q_3 r_0 \end{bmatrix}$$

$$QR^T = \begin{bmatrix} 0 & 0 & q_1 r_0 \\ 0 & q_2 r_{-1} & q_2 r_0 \\ q_3 r_{-2} & q_3 r_{-1} & q_3 r_0 \end{bmatrix}$$

$$S_{rel} = \begin{bmatrix} q_1 r_0 & 0 & 0 \\ q_2 r_{-1} & q_2 r_0 & 0 \\ q_3 r_{-2} & q_3 r_{-1} & q_3 r_0 \end{bmatrix}$$

3 Experiments

In the relative positional embeddings used by [2] the embedding matrix is incorporated into the model by multiplying it by the query matrix, and then adding the result to the attention matrix. However, this leaves room for the model to intermingle and confuse semantic and positional information which was shown in a different context to hurt the perormance of a Transformer model [3]. In order to help alleviate this we propose different integrations of the relative positional information that aim to better fuse the information into the model instead of simple addition.

The hyperparameters were kept consistent between our experiments and the baseline model. The optimizer, and learning rate warmup and decay were the same as described in [6], however, we reset the learning rate every 400,000 steps in order to speed up training. The inputs were learned embeddings of size 512 and were then concatenated with sinusoidal position embeddings again as described in [6] resulting in inputs of size 1024.

Table 1: Hyperparameter configuration.

Hyperparameter	Value
Sequence Length	2048
Batch Size	8
Transformer Layers	6
Model Dimension	1024
Feed Forward Dimension	2048
Heads	8
Dropout	0.1
Learning Rate	0.1

3.1 Additional Attention

In order to better integrate the relative positional information into the attention mechanism, we propose adding an additional query and value transform of the input which require their own parameters $W_2^Q, W_2^V \in \mathbb{R}^{d_x \times d_z}$ and results in $Q_2, V_2 \in \mathbb{R}^{h \times n \times d_h}$. The relative self attention mechanism originally used the

same values to query into the keys given by inputs and the keys that translate to relative positions. We plan to learn those queries and corresponding values again as functions of the input but separately:

$$RelativeAttention = Softmax(\frac{QK^T}{\sqrt{d_h}})V + Softmax(\frac{Q_2R^T}{\sqrt{d_h}})V_2 \quad (7)$$

As in previous works $R \in \mathbb{R}^{n \times d_h}$. R, unlike the other parameters in the attention mechanism, is consistent across attention heads, although still unique per transformer layer as this is at the level of the attention mechanism. In order to efficiently compute this across heads, an einsum is used. This increased our parameter count by 1.3x, and there was no noticeable difference in run time.

3.2 Data Augmentation

As previously stated, we used the same data augmentation methods as proposed by [4] except for cases where the vocabulary of notes would be extended. We also dropped time shift events. For implementation, we generated all of the avaliable pitch shifts in the range of [-3,3] ahead of time, and each epoch iterated over all of the new data.

4 Results

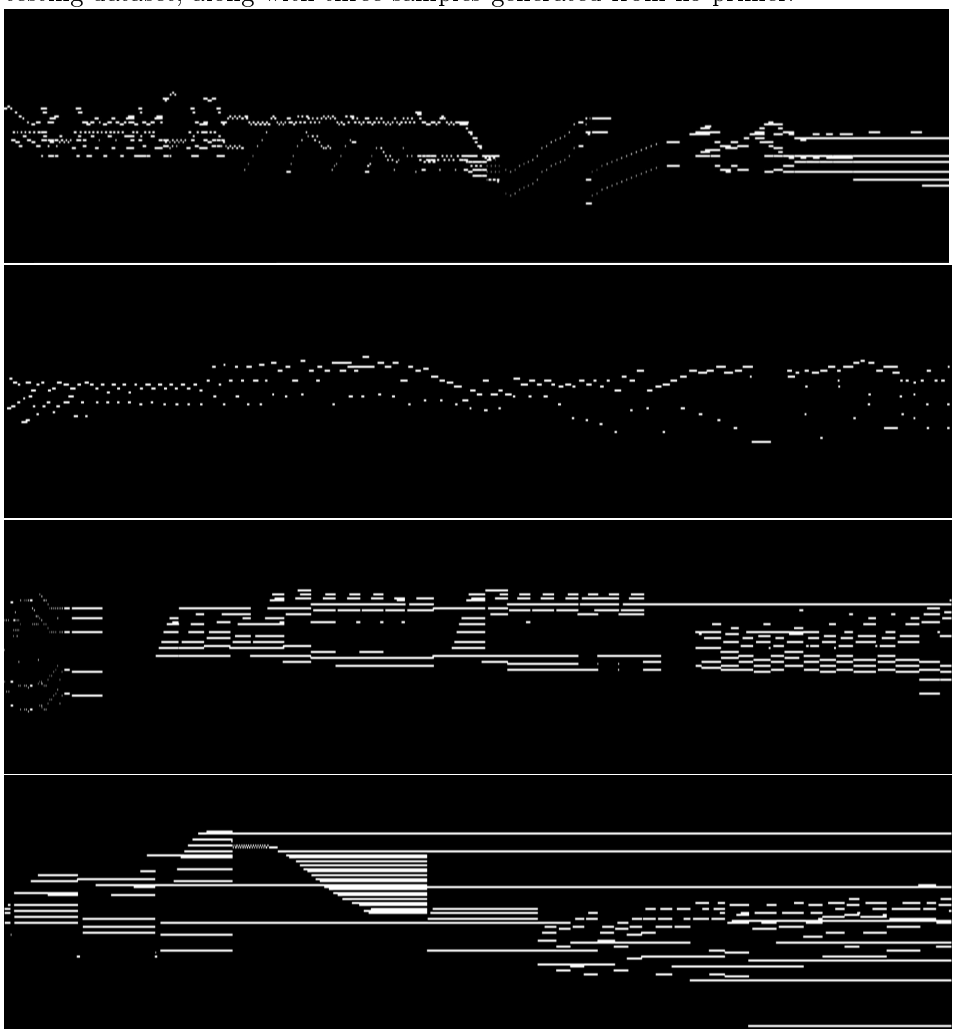
For training, 4 Tesla V100 GPU's were used. The compute was supplied by the the Milwaukee School of Engineering's super computer ROSIE. After hyperparameter tuning on each different experiment, two types of evaluation were used. First is the quantitative evaluation which was used for hyperparameter selection. The quantitative evaluation was the categorical cross entropy of a portion of the training data reserved for testing. The train/test split was 90/10 after shuffling the training data. After model selection for the individual experiments based on quantitative evaluation we will move on to qualitative evaluation. The qualitative evaluation will consist of blind participants listening to samples generated by a pair of models off of the same primer sequence or off of no primer and choosing which output they favored more.

For generating figures that were not given a primer, simply choosing the most likely event would result in the same generated sequence every time. In order to remedy this, we used an epsilon of $e = 0.6$ as a likelihood to not always pick the most likely event, and instead sample from the top-k likelihoods with probabilities equal to their model outputs with $k = 32$.

4.1 Data

The dataset used was the piano-e-competition. All of the years prior to 2018 were used as training and testing data, while the 2018 competition was withheld throughout the entirety of training as a validation set. That set was used for the

Figure 1: An example visualization of a sample generated from a primer in our testing dataset, along with three samples generated from no primer.



final quantitative evaluation below, as well as for generating primers to generate off of for the qualitative evaluation.

4.2 Quantitative Evaluation

Table 2: Qualitative evaluation of each model based on it's cross entropy of the validation dataset. The validation dataset was completely withheld during training. We also found that our model converged much faster than the baseline.

Model Variation	Validation Cross Entropy
[2] Music Transformer (Baseline)	2.0608
Our Additional Attention Model	1.9239

4.3 Qualitative Evaluation

For a qualitative evaluation we had participants listen to pairs of samples that were generated in batches and had them pick which of the samples they preferred per pair. Pairs were either generated off of the same primer or were generated with no primer as described in section 4.

Table 3: Quantitative evaluation of models compared pairwise in human listening tests. Wins and losses are from the perspective of Model A. The p-value was calculated using a binomial test. A statistically significant difference is denoted by an asterisk.

Model A	Model B	Win	Loss	p-value
Our Model	[2] (Baseline)	43	19	3.2e-3*

5 Conclusion

We have seen both a quantitative and a qualitative improvement over the Music Transformer [2] with the trade off of increasing memory requirements due to the extra parameters. While our quantitative improvement is small the results of our qualitative evaluation shows a significant improvement.

Bibliography

References

[1] Michael Conner, Lucas Gral, Kevin Adams, David Hunger, Reagan Strelow, and Alexander Neuwirth. Music generation using an lstm, 2022.

[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] Guolin Ke, Di He, and Tie-Yan Liu. Rethinking positional encoding in language pre-training. 2020.

[4] Sageev Oore, Ian Simon, Sander Dieleman, Douglas Eck, and Karen Simonyan. This time with feeling: Learning expressive musical performance, 2018.

[5] Peter Shaw, Jakob Uszkoreit, and Ashish Vaswani. Self-attention with relative position representations, 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, 2017.

Appendix A: Related Work

For a long time, sequence models have been the tool of choice for single track music generation, from Recurrent Neural Networks, to Gated Recurrent Units, and Long Short Term Memory [4, 1]. The representation of music has taken many forms, such as a discretized multi-hot grid based on time [4] [2], or using one-hot representations with specific events for changing time [4] [2], or using multi-hot representations to capture the note turned on, velocity, duration, and time-shift as a single event [1].

In recent years the transformer has been used in many applications ranging from image generation, speech summation, and chat bots. Many of these applications all share the improvements from their predecessors by increasing the length of sequences that are able to be input or generated. While the improvements are substantial there is an issue, as memory requirements grow quickly with longer sequences. This is especially the case for some implementations which increase memory requirements already in order to capture more information such as the relative self attention mechanism proposed by [5].

Appendix B: Annotated Bibliography

Attention is All You Need

[6] describe a new alternative to RNNs, LSTMs, GRUs and other models that handle sequential information, which they named the Transformer. The key element to their work is the lack of hidden states encoded into other hidden states. This caused information to be lost as time went on making large sequence lengths impossible. As the name suggests, the authors suggest a model which relies almost exclusively on attention mechanisms and remove the use of hidden states as RNNs use them. As they attend to entire sequences and no longer have the issue of hidden states being drowned out as sequences increase in length, they are able to handle tasks with much larger sequences. Like many other models made for sequence to sequence tasks, they use an encoder and a decoder. Each

uses attention in a new mechanism they propose called multi-head attention. Then using regular dense layers and normalization, they are able to achieve state of the art results in translation tasks. Since then, the Transformer has been iterated upon and used in many other tasks.

Self-Attention with Relative Position Representations

[5] builds upon the attention mechanism by creating a new positional representation. Unlike the sinusoidal positional embeddings used by [6], the new positional representation is first learned, and also captures positional information about the relative distances between tokens rather than absolute. This is stored in a matrix of the same size of the attention matrix, and is then added pairwise to get the new attention matrix. Also unlike the original positional embeddings, the relative information will be the same between the first and third elements, as it is for the second and fourth elements.

This Time with Feeling: Learning Expressive Musical Performance

[4] introduces many new ways to capturing many different types of representations for music. The relevant sections of the paper are those which cover expressive performances of polyphonic piano captured as midi data, as well as their methods of data augmentation. They introduce a way to use one-hot encoding to capture midi events and their expressive timings for live performances. In addition, this proposal also captures more events per time period when music is more dense in content, this gives a better representation that past representations which use sequences that correspond to fixed time periods.

The data starts as MIDI files which are preprocessed to remove all events that are irrelevant to capturing the needed data. Then all of the relevant events are condensed to account for events like a change in sustain pedal usage. The sustain pedal events give a value in [0-127], the pedal is regarded as in the on state when the event's value is $>= 64$ and off when the value is < 64. The note off events during time frames where the sustain pedal is in the on state are shifted to be at the same time the sustain pedal transitions to it's off state, or when the same note is repeated again, whichever happens first.

The events are then one-hot encoded where [0-127] represents note on events, [128-255] represents note off events, [256-355] represents time shifts, and [356-387] represents velocities. Unlike [4] which proposes 125 time shifts of 8ms increments, we chose 100 time shifts of 10ms time shifts. MIDI velocity events are also represented as a value between [0-127], however binning these into 32 bins greatly reduces complexity and still captures the dynamics needed to represent the performances.

They propose two types of data augmentation for less and more augmenting. The less augmentation consists of transposing all examples up or down all intervals up to a major third which creates 8 new examples. In addition, all

examples can be stretched by +-2.5% or +- 5.0% in time which creates 4 new examples.

Music Transformer

[2] makes the first introduction of using transformers for music generation. They show that transformers provided better results than their LSTM predecessors in both quality and output sequence length. In addition to their contributions to the field of music generation, they also provide algorithmic contributions to the relative self attention mechanism proposed in [5]. Their algorithmic contributions take the previous relative self attention mechanism which has memory requirements of $O(n^2 d_x)$ and reduces them to $O(nd_x)$. While the time complexity still remains at $O(n^2 d_x)$ they show improvements in model performance as well. This reduction is what allows them to handle sequences of higher lengths than previous works.

Additionally, the authors propose two different model architectures based on the type of music to be generated, continuations or accompaniments. In the case of accompaniments, which the authors generate for the JS-Bach dataset and introduce first, they use the original encoder-decoder type architecture as proposed in [6]. However, when the authors shift to generating continuations of given music pieces they instead opt to completely remove the encoder section of the network and put everything into a decoder. The decoder, and thus the core of the network, is then just an encoder with masking and a different attention mechanism followed by the final linear and softmax layers.