*Proceedings of Machine Learning Research* **318** (2026)
The 39th Canadian Conference on Artificial Intelligence



# Efficient Additive Relative Information Attention for Transformer-based Symbolic Music Composition

Felix Schön* and Hans Tompits**
Institute of Logic and Computation E192-03,
Technische Universität Wien,
Favoritenstraße 9-11, 1040 Vienna, Austria

**Abstract**

*Symbolic music generation* deals with automatically composing music in which the latter is treated as a language whose words represent musical events. In recent years, approaches based on the Transformer architecture using relative positional attention showed particular promise. However, a drawback common to the existing approaches is their limitation to relative distances between the positions of tokens only, rather than properties of the elements represented by them. To overcome this limitation, we introduce an efficient novel method for additive relative information injection based on block-sparse matrix operations. We evaluate the effectiveness of our approach by comparing it to different network architectures and conducting an array of experiments which show improvements over previous approaches.

**Keywords:** Attention, Block-Sparsity, Triton, Symbolic Music Generation

## 1. Introduction

*Symbolic music generation* refers to the process of composing music based on formalisable processes. In this discipline, music is represented using discrete elements called *tokens*. These tokens can, e.g., mark the beginning and end of notes. In this way, music can be seen as a type of "language", which lends itself well to the application of *large language models* (*LLM*s). Although various approaches have been applied to symbolic music generation [1–7], Transformer models [8] have shown particular promise [9–15]. Indeed, models that incorporate information on the relative distances between the positions of pairs of tokens, as was done with the Music Transformer [16], proved to be especially successful.

Several approaches for incorporating relative token distances have been discussed in the literature [17–20]. Notable examples include, e.g., the Transformer-XL [21], which uses a recurrence mechanism in combination with an additive relative information term to learn long-term relationships, and the RoFormer [22], which encodes positions of tokens using rotations of their corresponding vectors, allowing for a multiplicative approach for relative attention. A drawback common to all the additive approaches is the fact that they are only able to take the relative distances between the positions of a token in a sequence into consideration rather than properties of the elements represented by the tokens, like, e.g., differences in pitch between two notes or their temporal distances.

In this paper, we introduce a novel method for additive relative information attention with support for arbitrary integer information, which we refer to as *sparse pre-calculated relative information injection* (SPRII). Our approach is highly time and memory efficient, allowing for a practical implementation that was infeasible using previous approaches [16]. Our approach uses Triton [23], a language for creating efficient, highly parallel GPU operations. We utilise the `blksprs` library, introduced in a companion paper [24], a highly efficient library for operations on *block-sparse matrices*. Such matrices are used, e.g., by the Sparse Transformer [25] and the Longformer [26], and are characterised by structured patterns of zero-valued submatrices.

*schoen@kr.tuwien.ac.at **tompits@kr.tuwien.ac.at

We realise Triton kernels for common neural network operations on block-sparse matrices, including matrix multiplication, softmax, gather, and scatter. Using these kernels, we are able to (i) efficiently compute the relative distances between two given information vectors, (ii) efficiently compute the matrix product between queries and the information embeddings, and (iii) correctly position them so as to obtain the relative attention scores. Adapting the attention mechanism then to use these attention scores results in an attention process that depends on the relative distances between the provided information values.

We illustrate our SPRII approach on the task of symbolic music generation, applying our methods to information such as pitch values, temporal positions, or timings of notes. To evaluate the efficacy of our approach, we trained an ensemble of Transformer models using different selections of musical information and compared them to a GPT-like Transformer, the Music Transformer [16], and a RoFormer baseline [22]. Here, our models show good promise, outperforming the baselines on multiple metrics with statistical significance. Furthermore, we conducted a small-scale user study which suggests that these objective improvements in performance need not come at the cost of perceived musical quality. For reproducibility purposes, the source code of our implementation is publicly available at

https://github.com/FelixSchoen/SPRII.

The paper is structured as follows: In Section 2, we provide some background on music theory, the attention mechanism, and conventional relative information attention. In Section 3, we introduce our novel SPRII approach, discussing both its theoretical specification as well as the approach leveraged for practical deployment. The results of our experiments are outlined in Section 4, and Section 5 concludes the paper with a brief summary and outlook.

## 2. Background

### 2.1. Music Theory

As our work is closely tied to musical compositions, we briefly overview the central elements of music notation. For more information on this topic, we refer to, e.g., the works of Benward and Saker [27], Aldwell, Schachter, and Cadwallader [28], and Laitz [29].

In music notation, *notes* refer to pitch. The higher a note on the musical score, the higher its corresponding pitch is. Notes have a *value*, which determines how long they last. This in combination with *rests*, which also have a value but do not denote a pitch, allows for the construction of rhythm. These values are defined in relation to the *bar* an element is contained in. Bars are groupings of notes with a specific overall duration. Typically, a bar can fit up to four consecutive quarter notes or any combination of (simultaneous) notes of equivalent value. Using the MIDI file format,[1] we can store musical performance data. Here, time is represented using discrete *ticks*, with the duration of a quarter note commonly set to 24 ticks.

### 2.2. Scaled Dot-Product Attention

At the heart of the Transformer architecture [8] lies the *scaled dot-product attention* mechanism, which computes similarity scores between pairs of token vector representations. Scaled dot-product attention is calculated as follows:

$$sdpa(Q, K, V) \coloneqq softmax\Big(\frac{1}{\sqrt{d_{att}}} QK^\intercal\Big)V, \tag{2.1}$$

where $Q, K \in \mathbb{R}^{l_{seq} \times d_{att}}$ and $V \in \mathbb{R}^{l_{seq} \times d_{mod}}$ are matrices containing sets of *query*, *key*, and *value* vectors, respectively. Here, $l_{seq}$, $d_{att}$, and $d_{mod}$ give the *sequence length*, *dimensionality*

[1] https://midi.org.

*of the attention mechanism*, and *dimensionality of the model*, respectively, where $d_{att}$ and $d_{mod}$ are powers of 2. Moreover, $K^\intercal$ denotes as usual the transpose of $K$ and *softmax* is a function which produces a probability distribution matrix based on an input matrix $X \in \mathbb{R}^{m \times n}$ as follows: for $X = (x_{ij})$ and $softmax(X) = (y_{ij})$,

$$y_{ij} = \frac{e^{x_{ij}}}{\sum_{k=1}^{j} e^{x_{ik}}}, \qquad \text{for } 1 \leq i \leq m, 1 \leq j \leq n.$$

In practice, the arguments of $sdpa(\cdot, \cdot, \cdot)$ are the result of linear transformations applied to the internal representations of the tokens passed through the network. Given the transformation weights $W_Q, W_K \in \mathbb{R}^{d_{mod} \times d_{att}}$, $W_V \in \mathbb{R}^{d_{mod} \times d_{mod}}$, and the pre-transformation sequences $S \in \mathbb{R}^{l_S \times d_{mod}}$ and $T \in \mathbb{R}^{l_T \times d_{mod}}$ of length $l_S$ and $l_T$, respectively, we can define the following:

$$sdpa_{plt}^{W_Q, W_K, W_V}(S, T) \coloneqq sdpa(SW_Q, TW_K, TW_V).$$

Using the attention mechanism, the model is able to assign degrees of importance to relations between pairs of tokens. While initially the Transformer was conceptualised as a *sequence-to-sequence* [30] model, where *cross-attention* was conducted between two different sequences, after the advent of BERT [31], architectures solely relying on *self-attention* surged in popularity. Here, both $Q, K$, and $V$ stem from the same (input) sequence.

Note that, in practice, we apply *batching*, *masking*, and *multi-head attention* [8], but omit their discussion for the sake of brevity.

### 2.3. Relative Position Attention

As discussed in the introduction, different approaches for incorporating information about the relative distances between the positions of the tokens in a sequence exist [17–20]. In what follows, we focus on *additive relative attention* as introduced by Shaw et al. [17] and improved in the Music Transformer [16], but our approach introduced in the next section can be extended to conform to the different (additive) relative attention formulations as well.

Both Shaw et al. [17] and Huang et al. [16] start by encoding all possible distance values ranging from $-l_{seq}$ to $l_{seq}$ in a relative position embedding matrix $E$. In a more space-efficient approach compared to the former work, Huang et al. [16] then calculate the matrix product $QE^\intercal = (e'_{ij})$. They then apply a *skewing* procedure, which is a strategic application of a padding, reshaping, and slicing operation to ensure that for the relative scores $S^{rel}_{Q,E} = (s_{ij})$ with $s_{ij} = e'_{ik}$, where $k = l_{seq} + (j-1) - (i-1)$. In other words, $s_{ij}$ corresponds to the product of the $i$-th row of $Q_i$ and the $k$-th row of $E$, where the latter is the embedding for the distance $j - i$.

In order to include relative scores, Equation (2.1) is modified as follows, where $S \in \mathbb{R}^{l_{seq} \times l_{seq}}$:

$$sdpa_{rel}(Q, K, V, S) \coloneqq softmax\Big(\frac{QK^\intercal + S}{\sqrt{d_{att}}}\Big)V. \tag{2.2}$$

Huang et al. [16] then use $sdpa_{rel}(Q, K, V, S^{rel}_{Q,E})$ in place of $sdpa(Q, K, V)$.

We note that this approach only works with "information" (i.e., the absolute positions of tokens in a sequence in this case) that is starting with 0 and is strictly monotonically increasing by 1.

## 3. Relative Information Injection

We now introduce SPRII, our *sparse pre-calculated relative information injection* method. Section 3.1 first presents our general approach. Section 3.2 then details SPRII, realising an efficient formulation for practical applicability by exploiting properties of block-sparse matrices and the `blksprs` library introduced in a companion paper [24].

### 3.1. General Approach

#### 3.1.1. Formal Description

Let us consider (i) a sequence $T = (t_1, \ldots, t_{l_{seq}})$ of tokens, where $t_j \in \mathbb{Z}$, (ii) their embeddings $T^{emb} = (t_1^{emb}, \ldots, t_{l_{seq}}^{emb})$, for $t_j^{emb} \in \mathbb{R}^{d_{mod}}$, and (iii) an *information* vector $I = (i_1, \ldots, i_{l_{seq}})$, where $i_j \in \{v_{min}^i, \ldots, v_{max}^i, \epsilon\}$, for $1 \leq j \leq l_{seq}$. Here, we use an information value $\epsilon$ to indicate invalid information, e.g., only tokens corresponding to notes can have information about their pitch. Then, for every pair $(t_j, t_k)$, where $1 \leq k \leq l_{seq}$, we want to "inject" information about (some) relative distance of their information values $i_j, i_k$ into the attention process. Here, $v_{min}^i, v_{max}^i \in \mathbb{Z}$ give the limits for the possible information values. Although for the distance function most commonly $f(x, y) = y - x$ is chosen, in contrast to, e.g., RoPE [22], our SPRII approach supports a greater range of formulations for $f$, allowing for, e.g., the calculation of distances on the circle of fifths.

To achieve this, we first calculate the feasible relative information bounds $v_{fmin}^i = v_{min}^i - v_{max}^i$ and $v_{fmax}^i = v_{max}^i - v_{min}^i$, as well as $v_{range}^i = v_{fmax}^i - v_{fmin}^i + 1$, i.e., the possible range of numbers produced by $f$.[2] We then embed all values $p_j' = p_j - v_{fmin}^i$, for $p_j \in \{v_{fmin}^i, \ldots, v_{fmax}^i\}$, using a trainable embedding lookup table to obtain $I^{emb} = (p_0^{emb}, \ldots, p_{v_{range}^i}^{emb})$, where $p_j^{emb} \in \mathbb{R}^{d_{mod}}$. Afterwards, we apply the linear transformations $Q = T^{emb} W_T^{emb}$ and $E = I^{emb} W_I^{emb}$. Finally, we build the information matrix $R = (r_{jk})$, where $r_{jk} = f(i_j, i_k)$, a matrix containing the relative distances between $i_j$ and $i_k$ according to the function $f$.

Constructing a mathematical solution to the problem in question is relatively straightforward as one has to simply ensure that, for each pair $(t_j, t_k)$, the correct rows of $Q$ and $E$, identified by $f(j, k)$, are multiplied. To this end, we introduce the notion of an *indexed matrix multiplication* defined as follows:

**Definition 1.** *Let $A = (a_{ij}) \in \mathbb{R}^{m \times n}$, $B = (b_{ij}) \in \mathbb{R}^{n \times p}$, and $C = (c_{ij}) \in \{1, \ldots, p\}^{m \times p}$. Then, the indexed matrix multiplication, $matmul_{idx}(A, B, C)$, is the matrix $D = (d_{ij}) \in \mathbb{R}^{m \times p}$, where*

$$d_{ij} = \sum_{k=1}^{n} a_{ik} b_{kc_{ij}} = \sum_{l=1}^{p} \sum_{k=1}^{n} a_{ik} b_{kl} \delta(c_{ij}, l). \tag{3.1}$$

*Here, $\delta(x, y)$ is the indicator function that equals 1 if $x = y$, and 0 otherwise.*

Intuitively, in $matmul_{idx}(A, B, C)$, the values of $C$ define which columns of $B$ to multiply with a given row of $A$.

Let $\mathcal{L}$ be an arbitrary loss function dependent on the input of the model. We can then compute the derivatives of $\mathcal{L}$ with respect to the elements of $A = (a_{ij})$ as follows:

$$\frac{\partial \mathcal{L}}{\partial a_{ij}} = \sum_{g=1}^{p} \frac{\partial \mathcal{L}}{\partial d_{ig}} \frac{\partial d_{ig}}{\partial a_{ij}} = \sum_{g=1}^{p} \frac{\partial \mathcal{L}}{\partial d_{ig}} b_{jc_{ig}}.$$

The derivatives of $\mathcal{L}$ with respect to the elements of $B = (b_{ij})$ can be computed in a similar way:

$$\frac{\partial \mathcal{L}}{\partial b_{ij}} = \sum_{h=1}^{m} \sum_{g=1}^{p} \frac{\partial \mathcal{L}}{\partial d_{hg}} \frac{\partial d_{hg}}{\partial b_{ij}} = \sum_{h=1}^{m} \sum_{g=1}^{p} \frac{\partial \mathcal{L}}{\partial d_{hg}} a_{hi} \delta(c_{hg}, j).$$

Intuitively, the derivative of the element $b_{ij}$ is the sum of the derivatives of all the computations it is part of.

[2] Note that the exact range depends on the formulation of $f$.

#### 3.1.2. **A Direct Implementation**

We implemented the $matmul_{idx}$ function in PyTorch. To achieve performance close to traditional matrix multiplication, we opted to implement a block-sparse version of the $matmul_{idx}$ function in Triton [23]. The latter is a GPU programming language enabling the creation of highly-efficient GPU kernels without the need to take deep dives into the CUDA[3] platform. This allows for rapid prototyping of production-ready code.

Using this kernel, we can calculate $S^{info}{=}matmul_{idx}(Q, E^\intercal, R')$ (where $R' = R - (v^i_{min} - v^i_{max}) + 1$ is a shifted, positive-only-valued version of the information matrix $R$), which can then be applied analogously to Equation (2.2).

Although great attention was paid to the efficient design of the GPU kernel, the implementation of this straightforward approach incurs a significant performance penalty. We note three possible reasons for this fact: (i) in contrast to traditional matrix multiplication, the kernel has to fetch data from $E$ for every row of $Q$, (ii) the memory addresses of the blocks to load are not consecutive, and (iii) the GPU cores are optimised for matrix multiplications rather than vector multiplications which the kernel has to use.

In summary, our general approach aims to inject relative information (such as pitch or timing differences) into the attention mechanism by computing products between query vectors and information embeddings. The key challenge is ensuring that for each token pair, the correct embedding (determined by the relative distance between their information values) is used. While the indexed matrix multiplication provides a mathematically sound solution, its direct implementation is computationally expensive.

These shortcomings are remedied by our SPRII approach, which we introduce now in the next section. It relies heavily on the usage of *block-sparse matrices* in *compressed form* [24] in order to reduce its memory- and computational resource footprint.

### 3.2. **Sparse Pre-Calculated Relative Information Injection**

#### 3.2.1. **Block-Sparse Matrices**

The following definition adopts the notion of block-sparse matrices as used, e.g., by the Sparse Transformer [25] and the Longformer [26].

**Definition 2.** *We call a matrix* $M = (m_{ij}) \in \mathbb{R}^{m \times n}$ *a block-sparse matrix with* sparsity block size $\sigma_M$ *and* sparsity layout $L_M$ *if*

(i) $\sigma_M \in \{2^p \mid p \geq 4, m \bmod 2^p = 0, n \bmod 2^p = 0\}$,
(ii) $L_M = (l_{ij}) \in \{0,1\}^{\frac{m}{\sigma_M} \times \frac{n}{\sigma_M}}$, *where* $\frac{m}{\sigma_M}, \frac{n}{\sigma_M} \in \mathbb{N}$, *and*
(iii) $L_{M[i,j]} = 0$ *if and only if* $M_{[e,f]} = 0$, *for each* $e$ *and* $f$ *such that* $(i{-}1){\cdot}\sigma_M < e \leq i{\cdot}\sigma_M$ *and* $(j-1) \cdot \sigma_M < f \leq j \cdot \sigma_M$.

Intuitively, a value of 0 in the sparsity layout indicates a sparse (containing zeroes only) block in $M$.

Furthermore, given a block-sparse matrix $M$ as in Definition 2, we call a tensor $\overline{M} \in \mathbb{R}^{n_{sb} \times \sigma_M \times \sigma_M}$ a *compressed representation* of $M$, where $n_{sb} = \sum_{i,j} L_{M[i,j]}$, if the matrices it is made up of correspond to the non-sparse blocks in $M$, i.e., to those blocks in $M$ where the corresponding value $l_{ij} = 1$ [24]. The usage of compressed block-sparse matrices can greatly reduce the memory requirements for training neural networks, as large chunks of matrices do not have to be stored in-memory.

For the sake of simplifying notation, we treat operations on compressed representations of block-sparse matrices as if they were operating on regular matrices instead. Accordingly, in what follows, all operations making use of the simplified notation correspond to tensor

[3] https://developer.nvidia.com/cuda.

operations and are analogously marked with an overline. For example, $\overline{matmul}(\cdot,\cdot)$ refers to a block-sparse version of matrix multiplication.

#### 3.2.2. **Implementation**

Instead of employing the indexed matrix multiplication as introduced in Definition 1, we want to make use of the highly optimised architecture of a GPU for regular matrix multiplication. To this end, we "pre-calculate" the required products between $Q \in \mathbb{R}^{l_{seq} \times d_{mod}}$ and $E \in \mathbb{R}^{l_{info} \times d_{mod}}$, where $l_{info} = v^i_{max} - v^i_{min} + 1$, and only later ensure their correct positioning in the resulting $S^{info}$ matrix. Accordingly, we define

$$\overline{QE} = \overline{matmul}(\overline{Q}, \overline{E}^\intercal).$$

Note that, in practice, for this operation we specify the output sparsity layout $L_{QE}$, where $L_{QE} \in \{0,1\}^{\lfloor \frac{l_{seq}}{\sigma_Q} \rfloor \times \lfloor \frac{l_{info}}{\sigma_Q} \rfloor}$, computing only those blocks of $\overline{QE}$ covered by the layout. We obtain $L_{QE}$ by applying our custom $\overline{buildDistributionLayout}(\cdot)$ function, which we implemented as an efficient Triton kernel. For a target block-sparse matrix $\overline{T}$ and an input index matrix $\overline{M}$, let

$$L_T \coloneqq \overline{buildDistributionLayout}(\overline{M}) = (l_{ij}),$$

where

$$(l_{ij}) = \begin{cases} 1, & \text{if } \exists p, (i-1)\cdot\sigma_M < p \leq i \cdot \sigma_M, \\ & \quad \exists q, (j-1)\cdot\sigma_M < q \leq j \cdot \sigma_M, \\ & \quad \left\lfloor \frac{m_{pq}}{\sigma_M} \right\rfloor = j, \\ 0, & \text{otherwise,} \end{cases}, \tag{3.2}$$

i.e., those entries of $L_T$ (corresponding to sparsity blocks regarding $\overline{T}$) who are pointed to by $\overline{M}$ are set to 1. We can then calculate $L_{QE} = \overline{buildDistributionLayout}(\overline{R'})$. Note that the $\overline{buildDistributionLayout}$ is ordinarily used to obtain sparsity layouts for use with the *gather* and *scatter* operations, but can in this case be used to produce the desired sparsity layout of $\overline{QE}$.

By applying this procedure, we can significantly reduce the overhead that would be induced by having to compute all values of $QE$. In addition to that, when adapting, e.g., the Longformer [26] architecture, we can apply the sliding window size to the sparsity layout of $\overline{R'}$, further reducing the amount of entries to compute.

In the next step, we apply a custom block-sparse $\overline{gather}(\cdot,\cdot)$ operation to correctly position the previously calculated values. For a source matrix $S = (s_{ij}) \in \mathbb{R}^{m \times n}$ and an index matrix $X = (x_{ij}) \in \mathbb{N}^{+p \times q}$, we define

$$\overline{gather}(\overline{S}, \overline{X}) \coloneqq (s_{ix_{ij}}). \tag{3.3}$$

As Triton is not able to automatically compute derivatives for its kernels, we have to define a custom backwards function for the block-sparse $\overline{gather}(\cdot,\cdot)$ operation:

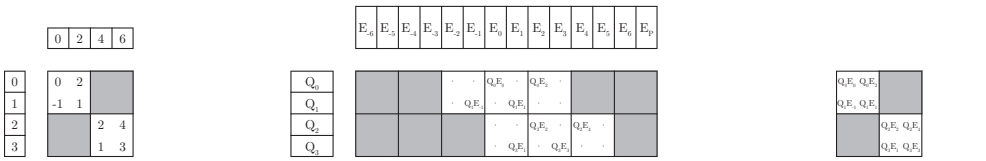
$$\frac{\partial \mathcal{L}}{\partial s_{ij}} = \sum_{k=1}^{q} \frac{\partial \mathcal{L}}{\partial t_{ik}} \frac{\partial t_{ik}}{\partial s_{ij}} = \sum_{k=1}^{q} \frac{\partial \mathcal{L}}{\partial t_{ik}} \delta(x_{ik}, j). \tag{3.4}$$

The function $\overline{scatter_{rs}}(\cdot,\cdot)$, defined below, does exactly that, reversing the gathering process and summing the derivatives up at previous positions of the entries:

$$\overline{scatter_{rs}}(\overline{S}, \overline{X}) \coloneqq (t_{ij}),$$

where

$$(t_{ij}) = \sum_{k=1}^{q} s_{ik} \delta(x_{ik}, j). \tag{3.5}$$



*(a)* First, the information matrix $\overline{R'}$ is built.

*(b)* Based on $\overline{R'}$, only the necessary blocks of $\overline{QE}$ are computed.

*(c)* Values from $\overline{QE}$ are stored in $\overline{S^{info}}$.

*Figure 1.* A visual representation of the SPRII process. The grey cells mark sparse blocks.

Finally, let $\overline{S^{info}} = \overline{gather}(\overline{QE}, \overline{R'})$ be the result of the gather operation applied to the pre-calculated scores $\overline{QE}$ and the indices $\overline{R'}$. The elements of $\overline{S^{info}}$ now correspond to the result of the vector multiplications between the rows of $\overline{Q}$ and the corresponding columns of $\overline{E}^\intercal$ as indicated by $\overline{R'}$.

We can now adapt Equation (2.2) to include the information values as follows:

**Definition 3.** *The* sparse pre-calculated relative information injection (SPRII) attention function *is given as follows:*

$$\overline{sprii}(\overline{Q}, \overline{K}, \overline{V}, \overline{S^{info}}) := \overline{softmax}\Big(\frac{\overline{Q}\overline{K}^\intercal + \overline{S^{info}}}{\sqrt{d_{att}}}\Big)\overline{V}. \tag{3.6}$$

Figure 1 shows a visual representation of the SPRII process on a small-scale example. The example illustrates how, in a first step, the information matrix is built from two input information vectors. Based on this matrix, in a next step, the $\overline{QE}$ matrix is built, which contains the final elements in the wrong positions, which are then, in the last step, correctly placed into the output $\overline{S^{info}}$ matrix.

Note that, in practice, we apply masking to this operation. We treat any relative distance $f(i_j, i_k) = \epsilon$ as undefined in case that either $i_j = \epsilon$ or $i_k = \epsilon$. In that case, we allow for either masking out the respective values, resulting in them not influencing the attention process, or replacing them by the maximum relative distance $v^i_{fmax}$. Recall that although Equations (3.2) through (3.6) all treat inputs like regular matrices, in practice we deal with block-sparse matrices in compressed form.

As a corollary of using this approach, we are less restricted in choosing $f$, the function assigning the relative distances for $\overline{R}$. As long as the values of $\overline{R}$ point to valid embedded information values in $\overline{E}$, the SPRII approach works. As a proof of concept, we implemented a Triton kernel calculating the distance between two notes on the circle of fifths, a musical categorisation of pitch classes. As its name suggests, these distances are circular and thus cannot be computed using subtraction only as used by the other relative positional approaches.

Our SPRII approach can also be used to incorporate conventional additive relative positional attention into models making use of block-sparse attention, such as, e.g., the Longformer [26] or the Museformer [32]. This was previously not feasible, as the efficient computation of $S^{rel}_{Q,E}$ required access to the full, non-sparse matrix $QE$ [16, 21].

# 4. Experiments and Evaluation

## 4.1. Experimental Settings

### 4.1.1. Models and Hyperparameters

To evaluate SPRII, we apply our approach to a practical neural network architecture. We experimented with the inclusion of different relative information and the respective bounds on the maximum distances allowed. We tested a total of four different configurations, combining relative attention over the position, temporal position, temporal position within a bar, pitch, and position on the circle of fifths of the respective tokens.

We compare our SPRII architecture to the following three approaches that all use different attention formulations: (i) a GPT-like architecture as introduced by Vaswani et al. [8] that does not make use of relative attention, (ii) the Music Transformer [16], which introduced the memory-efficient relative positional attention formulation, and (iii) the RoFormer [22], which uses multiplicative relative positional attention.

We use similar hyperparameters across all configurations for fair comparison. For all models, we use a model dimensionality of 512, 2,048 neurons per feed-forward layer, 4 attention heads, a dropout rate of 0.1, and a total of 4 layers. These hyperparameters were chosen to align with comparable work in symbolic music generation [16] while keeping model sizes tractable for fair comparison across all configurations. We trained the models for 16 epochs each using eight NVIDIA Ampere GPUs.

### 4.1.2. Datasets

We conducted our experiments based on the widely used *Lakh MIDI* dataset [33], comprising 178,561 distinct MIDI files. Although the dataset is deduplicated by the MIDI files' MD5 checksum, it still contains a large number of perfect or near-perfect duplicates as outlined by Zeng et al. [34]. We first apply the same deduplication procedure, using which we were able to remove about 17% of duplicate pieces.

Using our custom music preprocessing library, we preprocess and filter the dataset based on various criteria, such as, e.g., inconsistent channels, empty tracks, or unsupported time signatures. We then assign and merge tracks to one of the six supported instruments, these being piano, guitar, bass, strings, synth, and drums. Lastly, we quantise the notes and augment the tracks by adding a version transposed by a randomly selected value of -5 to +6 half tone steps.

We chunk sequences into blocks of 16 bars, and split it into a training, validation, and test portion with 80%, 10%, and 10% of the data, respectively. We then used an approach similar to REMI [35] to convert the MIDI files to a machine-readable representation.

In order to further validate our results, we conducted an objective evaluation on the POP909 dataset [36] as well. Here, we apply the same deduplication and preprocessing steps as outlined above.

## 4.2. Perplexity Evaluation

*Perplexity* measures a model's ability to predict future tokens. It is calculated on the test set using

$$exp\Big(-\frac{1}{N}\sum_{i=1}^{N} log(P(t_i|t_1,\ldots,t_{i-1}))\Big), \tag{4.1}$$

where $P(t_i|t_1,\ldots,t_{i-1})$ is the conditional probability of $t_i$ given $t_1,\ldots,t_{i-1}$, i.e., by exponentiating the average negative log-likelihood. Here, lower values indicate a model that makes more correct predictions and is more certain about them.

| Model | 16 Bars (Lakh MIDI) | 32 Bars (Lakh MIDI) | 16 Bars (POP909) | 32 Bars (POP909) |
|---|---|---|---|---|
| GPT-like | 2.53 | > 999 | 5.86 | > 999 |
| Music Transformer | 2.46 | 2.77 | 5.64 | 6.28 |
| RoFormer | 2.51 | 2.94 | 5.83 | 7.01 |
| SPRII (position + time + time bar) | 2.42 | **2.64** | 5.47 | **5.70** |
| SPRII (position + time) | **2.41** | 2.80 | **5.46** | 6.42 |
| SPRII (position + pitch) | 2.47 | 2.87 | 5.70 | 6.41 |
| SPRII (position + time + pitch + C.o.F.) | 2.46 | 2.73 | 5.64 | 6.17 |
| SPRII (position only, ablation) | 2.46 | 2.73 | 5.64 | 6.16 |

*Table 1.* Perplexity results for the different approaches compared for different sequence lengths (lower is better).

| Model / Metric | Harmony | Rhythm | Variedness | Genuineness | Quality |
|---|---|---|---|---|---|
| Ground Truth | 4.14 ± 0.83 | 4.00 ± 1.13 | 3.43 ± 1.12 | 4.00 ± 1.07 | 3.79 ± 0.94 |
| GPT-like | 3.21 ± 1.01 | 3.14 ± 1.30 | 2.71 ± 1.10 | 2.71 ± 1.03 | 2.71 ± 0.88 |
| Music Transformer | 2.71 ± 0.96 | 2.79 ± 1.01 | 3.00 ± 0.93 | 2.43 ± 1.05 | 2.71 ± 1.28 |
| RoFormer | 2.64 ± 1.23 | 3.07 ± 1.39 | 2.43 ± 0.82 | 2.64 ± 1.29 | 2.43 ± 1.05 |
| SPRII (position + time + time bar) | **3.57 ± 1.05** | **3.93 ± 0.80** | **3.57 ± 1.24** | **3.36 ± 1.29** | **3.79 ± 1.01** |

*Table 2.* Results (mean ± standard deviation) of the user study per model and metric (higher is better).

Table 1 shows the results of this evaluation. Here, we computed perplexity values on both the Lakh MIDI and the POP909 dataset. We opted to compare performance on both segments of up to 16 bars (which corresponds to the length the networks were trained on) and segments of up to 32 bars in length, in order to assess the models' ability to extrapolate to longer sequences than seen during training.

We tested four different SPRII configurations as outlined in Section 4.1.1, and one model using conventional positional information only for the purposes of ablation.

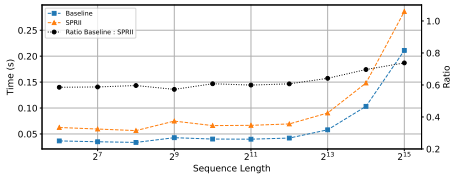
Both the first and second SPRII approaches outperform all of the baseline models in the 16-bar case, although only the model incorporating relative attention over the tokens' position, time, and intra-bar time statistically significantly outperforms the Music Transformer (and all the other baselines, as well as the ablation model) in all cases, with a level of $p < 0.01$. This advantage is especially clear in the two 32-bar cases, where our first approach shows particular promise over the baseline models, while the vanilla GPT-like Transformer completely deteriorates. The other SPRII approaches show good promise as well, but fail to outperform the Music Transformer in the two 16-bar cases for both datasets.

All in all, the inclusion of SPRII can measurably and positively impact the objective performance of the models. Furthermore, SPRII can be used to include additive relative positional attention for models making use of block-sparse attention, such as, e.g., the Longformer [26].

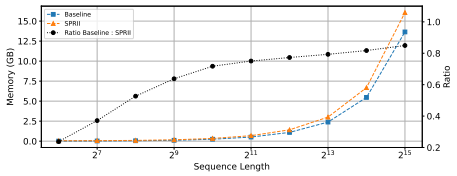
### 4.3. **User Study**

To evaluate qualitative differences between model outputs, we conducted a small-scale user study. Participants were shown five 16-bar musical pieces: one generated by each evaluated model configuration and one piece from the test dataset. They rated each piece on a 1–5 scale for (i) harmony, (ii) rhythm, (iii) variedness, (iv) genuineness, and (v) overall quality. We received 14 submissions, each rating five samples randomly drawn from the outputs of the different approaches.

The results of the user study are shown in Table 2. Based on the results of Section 4.2, we selected a configuration including relative information over the position, temporal position, and temporal position within a bar. Although the configuration eschewing the temporal




*(a)* Runtime benchmark results for the baseline Longformer model and the Longformer with SPRII.




*(b)* Memory benchmark results for the baseline Longformer model and the Longformer with SPRII.

*Figure 2.* Runtime and memory benchmarks for a baseline model with (orange) and without (blue) SPRII.

position within a bar showed slightly (but not statistically significant) improvements over the selected counterpart, the former configuration performs significantly better when it comes to interpolating to longer sequences.

Our SPRII approach outperforms the baselines across all metrics. We note that music evaluation is inherently subjective; all models also produce failure samples lacking variedness or structure, which we excluded from this survey.

Our user study suggests that, for the sampled outputs considered in the survey, SPRII does not negatively impact perceived output quality, which combined with the significant perplexity improvements (Section 4.2) supports SPRII's use for symbolic music generation.

### 4.4. **Complexity Analysis**

In contrast to our approach, the memory-efficient approach of the Music Transformer [16] cannot be extended to arbitrary information. It has the same theoretical space complexity of $O({l_{seq}}^2 d_{att})$, as given by Shaw et al. [17], which would make a practical implementation infeasible. With our SPRII attention, we are able to greatly reduce these requirements.

Recall that $v^i_{range}$ gives the feasible range of the possible relative distances between any two information values $i_j$ and $i_k$, respectively. For the full attention case, our approach then requires $O(v^i_{range} d_{mod})$ space for the information embeddings, $O({l_{seq}}^2)$ space for the distance matrix, and a theoretical maximum of $O(l_{seq} v^i_{range})$ space for the word-information matrix $\overline{QE}$. In practice, we often require significantly less space for $\overline{QE}$ due to the fact that large blocks of the matrix are not covered by values of $R$ and therefore need not be stored when represented in a compressed block-sparse format. As the exact sparsity layout of $\overline{QE}$ depends on the information provided, we can only give an overestimating upper bound of $O(l_{seq} v^i_{range} d_{mod})$ on the time complexity of the SPRII calculation for the full attention case.

As our approach makes heavy use of block-sparse operations, it can easily be applied to models such as, e.g., the Museformer [32] or the Longformer [26] that make use of (specialised) sliding window attention. Here, each token attends to only a select number of other tokens, usually within a window of size $s_{win}$.

We implemented a custom Triton kernel to efficiently produce $\overline{R}$ in $O(l_{seq} s_{win})$ time and space. Although the exact time and space complexity of the calculation of $\overline{QE}$ still depends on the values of $\overline{R}$, we obtain the same complexities for $\overline{S^{info}}$ as for the calculation of $\overline{R}$, as they share the same sparsity layout.

Figure 2 compares a regular Longformer to one with SPRII (hyperparameters as in Section 4.1.1, with 512 information positions). SPRII adds only minimal overhead, and both runtime and memory scale linearly with the baseline, showing promise for longer sequences.

## 5. Conclusion

In this paper, we introduced SPRII, a novel sparse pre-calculated relative information injection approach. Using it, we are able to efficiently incorporate arbitrary integer information, such as, e.g., a note's pitch, for additive relative attention. We achieved this by leveraging block-sparse matrix kernels for operations such as matrix multiplication, transposition, gather, and scatter. Using these kernels, we were able to significantly reduce previous complexity bounds for this approach, allowing for a practical implementation that is both memory and time efficient.

In an experimental examination, we showed that including SPRII over previous relative attention formulations can improve the perplexity of a model. Furthermore, the results of our user study suggest that models using SPRII are able to produce output of high quality, featuring musically valid harmonies and good rhythmic quality.

Our SPRII approach is very flexible, allowing for, e.g., custom relative distance functions, or the exclusion of invalid information from the process. This can be a key advantage and is not possible with current additive or multiplicative formulations. Furthermore, it can be used to incorporate traditional additive relative attention for models making use of block-sparse attention.

### References


[1] N. Boulanger-Lewandowski, Y. Bengio, and P. Vincent. "Modeling Temporal Dependencies in High-Dimensional Sequences: Application to Polyphonic Music Generation and Transcription". In: *Proc. ICML 2012*. 2012.

[2] G. Hadjeres, F. Pachet, and F. Nielsen. "DeepBach: A Steerable Model for Bach Chorales Generation". In: *Proc. ICML 2017*. 2017.

[3] H. Dong, W. Hsiao, L. Yang, and Y. Yang. "MuseGAN: Multi-track Sequential Generative Adversarial Networks for Symbolic Music Generation and Accompaniment". In: *Proc. AAAI 2018*. 2018.

[4] G. Mittal, J. H. Engel, C. Hawthorne, and I. Simon. "Symbolic Music Generation with Diffusion Models". In: *Proc. ISMIR 2021*. 2021.

[5] D. Prvulovic, R. Vogl, and P. Knees. "ReStyle-MusicVAE: Enhancing User Control of Deep Generative Music Models with Expert Labeled Anchors". In: *Proc. UMAP 2022*. 2022.

[6] Y. Huang, A. Ghatare, Y. Liu, Z. Hu, Q. Zhang, C. S. Sastry, S. Gururani, S. Oore, and Y. Yue. "Symbolic Music Generation with Non-Differentiable Rule Guided Diffusion". In: *Proc. ICML 2024*. 2024.

[7] Z. Wang, L. Min, and G. Xia. "Whole-Song Hierarchical Generation of Symbolic Music Using Cascaded Diffusion Models". In: *Proc. ICLR 2024*. 2024.

[8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin. "Attention is All you Need". In: *Proc. NIPS 2017*. 2017.

[9] C. Payne. *MuseNet*. OpenAI Blog. 2019.

[10] A. Muhamed, L. Li, X. Shi, S. Yaddanapudi, W. Chi, D. Jackson, R. Suresh, Z. C. Lipton, and A. J. Smola. "Symbolic Music Generation with Transformer-GANs". In: *Proc. AAAI 2021*. 2021.

[11] W. Hsiao, J. Liu, Y. Yeh, and Y. Yang. "Compound Word Transformer: Learning to Compose Full-Song Music over Dynamic Directed Hypergraphs". In: *Proc. AAAI 2021*. 2021.

[12] J. Liu, Y. Dong, Z. Cheng, X. Zhang, X. Li, F. Yu, and M. Sun. "Symphony Generation with Permutation Invariant Language Model". In: *Proc. ISMIR 2022*. 2022.

[13] P. Lu, X. Xu, C. Kang, B. Yu, C. Xing, X. Tan, and J. Bian. "MuseCoco: Generating Symbolic Music from Text". In: *CoRR* abs/2306.00110 (2023).

[14] S. Wu and Y. Yang. "MuseMorphose: Full-Song and Fine-Grained Piano Music Style Transfer With One Transformer VAE". In: *IEEE ACM Transactions on Audio Speech and Language Processing* 31 (2023).

[15] J. Thickstun, D. L. W. Hall, C. Donahue, and P. Liang. "Anticipatory Music Transformer". In: *Transactions on Machine Learning Research* 2024.04 (2024).

[16] C. A. Huang, A. Vaswani, J. Uszkoreit, I. Simon, C. Hawthorne, N. Shazeer, A. M. Dai, M. D. Hoffman, M. Dinculescu, and D. Eck. "Music Transformer: Generating Music with Long-Term Structure". In: *Proc. ICLR 2019*. 2019.

[17] P. Shaw, J. Uszkoreit, and A. Vaswani. "Self-Attention with Relative Position Representations". In: *Proc. NAACL-HLT 2018*. 2018.

[18] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer". In: *Journal of Machine Learning Research* 21 (2020).

[19] G. Ke, D. He, and T. Liu. "Rethinking Positional Encoding in Language Pre-training". In: *Proc. ICLR 2021*. 2021.

[20] P. He, X. Liu, J. Gao, and W. Chen. "Deberta: Decoding-Enhanced Bert with Disentangled Attention". In: *Proc. ICLR 2021*. 2021.

[21] Z. Dai, Z. Yang, Y. Yang, J. G. Carbonell, Q. V. Le, and R. Salakhutdinov. "Transformer-XL: Attentive Language Models beyond a Fixed-Length Context". In: *Proc. ACL 2019*. 2019.

[22] J. Su, M. H. M. Ahmed, Y. Lu, S. Pan, W. Bo, and Y. Liu. "RoFormer: Enhanced Transformer with Rotary Position Embedding". In: *Neurocomputing* 568 (2024).

[23] P. Tillet, H. Kung, and D. D. Cox. "Triton: An Intermediate Language and Compiler for Tiled Neural Network Computations". In: *Proc. MAPL@PLDI 2019*. 2019.

[24] F. Schön and H. Tompits. "`blksprs`: A Triton Library for Block-Sparse Matrix Operations". In: *Proc. Canadian AI 2026*. Vol. 318. Proceedings of Machine Learning Research. 2026.

[25] R. Child, S. Gray, A. Radford, and I. Sutskever. "Generating Long Sequences with Sparse Transformers". In: *CoRR* abs/1904.10509 (2019).

[26] I. Beltagy, M. E. Peters, and A. Cohan. "Longformer: The Long-Document Transformer". In: *CoRR* abs/2004.05150 (2020).

[27] B. Benward and M. Saker. *Music in Theory and Practice: Volume 1*. Eighth Edition. McGraw-Hill, 2009.

[28] E. Aldwell, C. Schachter, and A. Cadwallader. *Harmony & Voice Leading*. Fifth Edition. Cengage, 2019.

[29] S. G. Laitz. *The Complete Musician: An Integrated Approach To Tonal Theory, Analysis, and Listening*. Third Edition. Oxford University Press, 2012.

[30] I. Sutskever, O. Vinyals, and Q. V. Le. "Sequence to Sequence Learning with Neural Networks". In: *Proc. NIPS 2014*. 2014.

[31] J. Devlin, M. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proc. NAACL-HLT 2019*. 2019.

[32] B. Yu, P. Lu, R. Wang, W. Hu, X. Tan, W. Ye, S. Zhang, T. Qin, and T. Liu. "Museformer: Transformer with Fine- and Coarse-Grained Attention for Music Generation". In: *Proc. NeurIPS 2022*. 2022.

[33] C. Raffel. "Learning-Based Methods for Comparing Sequences, with Applications to Audio-to-MIDI Alignment and Matching". PhD thesis. Columbia University, 2016.

[34] M. Zeng, X. Tan, R. Wang, Z. Ju, T. Qin, and T. Liu. "MusicBERT: Symbolic Music Understanding with Large-Scale Pre-Training". In: *Proc. ACL 2021*. 2021.

[35] Y. Huang and Y. Yang. "Pop Music Transformer: Beat-based Modeling and Generation of Expressive Pop Piano Compositions". In: *Proc. ACM 2020*. 2020.

[36] Z. Wang, K. Chen, J. Jiang, Y. Zhang, M. Xu, S. Dai, G. Bin, and G. Xia. "POP909: A Pop-Song Dataset for Music Arrangement Generation". In: *Proc. ISMIR 2020*. 2020.