Modelling long- and short-term structure in symbolic music with attention and recurrence

Jacopo de Berardinis[1,3⋆], Samuel Barrett[2⋆], Angelo Cangelosi[1], and Eduardo Coutinho[3]

[1] Machine Learning and Robotics Group, University of Manchester
[2] Department of Computer Science, University of Oxford
[3] Applied Music Research Lab, University of Liverpool

Abstract. The automatic composition of music with long-term structure is a central problem in music generation. Neural network-based models have been shown to perform relatively well in melody generation, but generating music with long-term structure is still a major challenge. This paper introduces a new approach for music modelling that combines recent advancements of transformer models with recurrent networks – the long-short term universal transformer (LSTUT), and compare its ability to predict music against current state-of-the-art music models. Our experiments are designed to push the boundaries of music models on considerably long music sequences – a crucial requirement for learning long-term structure effectively. Results show that the LSTUT outperforms all the other models and can potentially learn features related to music structure at different time scales. Overall, we show the importance of integrating both recurrence and attention in the architecture of music models, and their potential use in future automatic composition systems.

Keywords: Music modelling; Predictive models for music; Long-term music structure; Long-short term memory; Transformers.

1 Introduction

Music is hierarchical in nature, with *notes* played over milliseconds, *motifs* organised over hundreds of milliseconds, *phrases* spanning over seconds, and so on (Koelsch et al., 2013). For automatic composition systems, successful modelling of music at all time scales in this hierarchy is critical for the music to match the conventions of Western tonal harmony. As human composers work with the interrelations among local (e.g. *a perfect cadence*) and global structures (e.g. *sonata form*), we would like generative systems to imitate similar capabilities.

Neural network-based methods for automatic music composition are generally trained as sequence models, which learn to predict the next event in a sequence given the historical context of the previous events. In music modelling, events are notes and silences, temporal slices of notes, or groupings of notes,

⋆ Equal contribution.

depending on the granularity of the chosen music representations. Recurrent neural networks (RNNs), and in particular long-short term memory networks (LSTM) (Hochreiter & Schmidhuber, 1997), have been the dominant paradigm in music modelling and generation. Previous work has demonstrated their ability to compose motifs and melodies in diverse styles and genres, from Bach chorales (Liang et al., 2017) to blues and folk music (Eck & Schmidhuber, 2002; Sturm et al., 2015). More recently, self-attention networks – the *transformers* (Vaswani et al., 2017), revolutionised the field of natural language processing, and more generally, all tasks dealing with sequential data exhibiting long-term dependencies. Transformers have also been applied to music generation and proven to be an effective paradigm (Huang et al., 2018; Choi et al., 2019; Wu et al., 2020). Conversely to RNNs, transformers do not keep a memory representation while processing an input sequence, but a soft (differentiable) mechanism is learned to contextually access all past sequence elements (c.f. Section 3.1).

However, music modelling is a challenging task for both *transformers* and *RNNs*. The former suffer from complexity and resolution issues (Dehghani et al., 2018) and the latter struggle to learn long-term dependencies from sequences (Bengio et al., 1994). The inability of these models to deal with music structure at different temporal resolutions directly affects the quality of the generated music. This is a crucial limitation of current music generation system as generated music tends to be either too repetitive or too contrasting (Briot et al., 2020).

In this paper, we introduce a novel architecture for music modelling and generation – the long-short term universal transformer (LSTUT). Our method leverages the ability of LSTMs to learn short musical ideas and the long-term modelling capabilities of transformers by combining them in a single architecture. To test our contribution, we evaluate the LSTUT on the MAESTRO dataset (Hawthorne et al., 2019) – a collection of classical symbolic music for piano, and compare it to state-of-the-art (SoA) neural network models for music modelling. We demonstrate that our method outperforms other approaches and can learn features related to musical structure. Our contributions to the field of music modelling and generation are twofold: (i) we demonstrate the importance of *recurrence* and *attention* for music modelling; (ii) we introduce a class of music models focused on learning music structure that outperform SoA approaches.

2 Related Work

For LSTM networks, improving the ability to relate musical content at longer time scales is typically done through *conditioning* or increasing *memory capacity*. The first method provides the network with additional input at each time step, such as the current position within the measure (Johnson, 2017). The second consists in either increasing the number of memory cells or introducing an attention layer (Bahdanau et al., 2014) to let the network contextually access the input sequence up to a certain number of elements (Waite, 2016).

On the other hand, transformer networks have demonstrated to be promising at modelling and generating music with increased structural complexity (Huang et al., 2018; Choi et al., 2019; Wu et al., 2020). Nevertheless, one of the most problematic issues of transformers, which is further discussed in the paper, is the quadratic complexity of their attention mechanism. A common approach to circumvent this issue consists in splitting each training sequence into a number of sub-sequences of manageable length (e.g. 30 seconds). However, this prevents learning across time scales larger than the predefined window.

Similarly to the LSTUT, Wang et al. (2019) also combined transformers and RNNs. However, their architecture does not posses an inductive bias in the transformer part, and the use of the original attention mechanism, requiring quadratic complexity, hinders their applicability to long music sequences.

3 The Long Short-Term Universal Transformer

Our architecture is specifically designed to learn both short- and long-term dependencies from music sequences. Even though short musical excerpts are adequate to learn simple musical ideas, such as *figures* and *motifs*, learning long-term dependencies is only possible if entire music sequences are considered. This calls for a model which can handle long sequences efficiently and learn structural patterns at different temporal resolutions. To that aim, we adapted the transformer architecture to be computationally more efficient (both in time and space), as well as making better use of its parameters. This was achieved through linear attention and weight sharing. To learn structural patterns at different time scales, we added recurrence to the model in two ways: (i) leveraging the weight sharing mechanism to perform multiple encoding steps at a linear cost (*recurrence in depth*); and (ii) integrating recurrent neural networks upstream and downstream the transformer layer (*recurrence in time*). The latter is important to learn local structures, whereas the former improves the long-term modelling capabilities. The architecture of the resulting model – the long-short term universal transformer (LSTUT) – is illustrated in Figure 1 and detailed as follows.

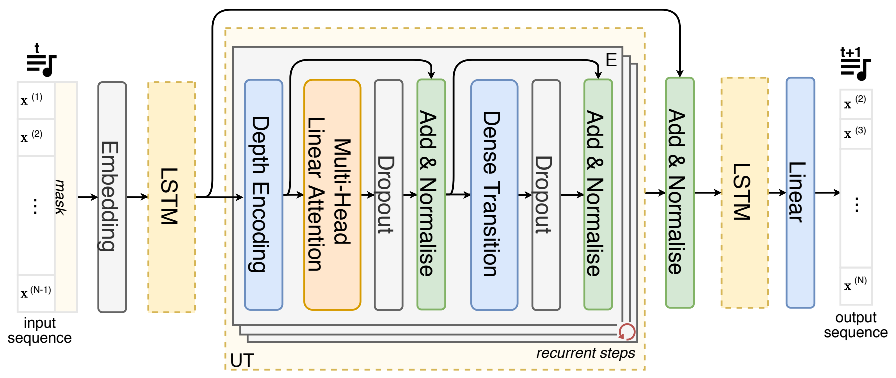

Fig. 1. Illustration of the LSTUT architecture for music modelling.

3.1 The Transformer

Transformers are a class of neural networks for sequence processing based on the concept of *self-attention*. Instead of keeping an explicit memory representation, this mechanism involves comparing the elements (or tokens) in an input sequence with all of the others, and updating the information about each token as a result. To this end, three vectors are computed from each position in the input sequence: a *query*, a *key*, and a *value*. The amount of attention that token $\mathbf{x}^{(j)}$ pays to token $\mathbf{x}^{(k)}$ is derived from the former being used as query and the latter being used as a key. The resulting attention distributions are then used as weights, so that each token is represented as a weighted sum of its value vector. To learn different representations, the attention mechanism is multiplexed into several "heads", each yielding its own attention distribution. The attention heads differentiate themselves by the way they compute the queries, keys and values.

Formally, let i be the index of an attention head, and let $Q_i, K_i, V_i \in \mathcal{M}_{n\times n}$ be query, key and value matrices for that head, respectively. For a sequence represented by the matrix $s \in \mathcal{M}_{n\times N}(\mathbb{R})$ where N is the sequence length and n is the dimension of each token, the values produced by attention head i are:

$$\texttt{softmax}\left(\frac{(Q_i s)^T (K_i s)}{\sqrt{n}}\right)(V_i s)\,. \tag{1}$$

The values produced by the attention heads are then concatenated, and matrix multiplication is applied (at each position in the sequence independently) to reduce their dimensionality back to n. After the multi-headed attention sub-layer, a dense neural network is applied as a transition function. Layer normalisation and residual connections are then used in between these two sub-layers.

In our architecture, the transformer is used to relate musical content throughout the composition, e.g. identifying the repetition of phrases or sections.

3.2 Adding Recurrence in Depth and Reducing Complexity

The transformer architecture originally proposed in (Vaswani et al., 2017) has been criticised for being computationally inefficient and for not containing an *inductive bias* (Dehghani et al., 2018). The computational complexity of the vanilla transformer is proportional to the square of the sequence length, which becomes prohibitive for considerably long sequences. An *inductive bias* is a tendancy to learn iterative or recursive transformations, a property that recurrent networks posses by definition. For instance, the LSTM learns a function of both the input and the state, which is then applied iteratively across the input sequence. This property is appealing for music modelling, due to the hierarchical nature of tonal music which makes it suited to such transformations.

To address the first issue, our architecture implements the attention mechanism from (Katharopoulos, Vyas, Pappas, & Fleuret, 2020), which makes the complexity linear in the sequence length. This is achieved by defining the attention among two tokens as the dot product between the queries and keys, which permits a factorisation (cf. Appendix A). To include an inductive bias in the transformer, we consider its generalisation from (Dehghani et al., 2018) – the universal transformer (UT). As can be seen in Figure 1, the transformer layer is repeated as a loop according to the number of *recurrent steps.* This originates a stack of encoder (E) layers sharing the same parameters.

3.3 Adding Recurrence in Time

In the original transformer, encoding the temporal relationships of sequence elements, prior to the self-attention sub-layer, is done by manually injecting this information for each token – a procedure known as *positional encoding.* The reliance on positional encodings has two consequences: the model struggles with both learning a precise representation of the input, and learning to reconstruct a precise representation of the output. In addition, transformers rely on having diversity of attention heads, which is needed for learning across different time scales, although not guaranteed as shown in (Li et al., 2018).

To address these issues, our architectures includes two LSTM networks between the linear UT layer. The LSTM on the lower level has two main benefits: (i) it temporally contextualises the input data, thus replacing the positional encoding; (ii) it facilitates the process of modelling local structures by learning a higher level representation of the input sequence, e.g. labelling chords. The UT then distributes this information across the entire input in a global fashion. The LSTM on the top level is used to "re-localise" the output of the UT, thereby reconstructing it to a high resolution, e.g. outputting chords.

4 Experiments

In line with previous works (Elliot, 2016; Huang et al., 2018), we hypothesise that *recurrence* and *attention* are both key components for learning short- and long-term dependencies from music. Our hypothesis can be decomposed as follows: (i) *attention* is key for learning long-term structure; (ii) *recurrence* facilitates the discovery of short-term structure; (iii) *inductive biases* improve the ability to learn long-term structure. All these elements are present in the LSTUT.

To test our hypothesis, we devised an experimental framework that includes (i) a *basic LSTM* (recurrence and inductive bias); (ii) an *attention LSTM*, adding an attention mechanism to the former; (iii) a *transformer* (self-attention); (iv) a *UT* (self-attention and inductive bias); and (v) a *LSTUT* (self-attention, inductive bias, and recurrence). The first 4 models were also used as baselines for evaluating the performance of our proposed model (LSTUT).

We evaluated all models on the music modelling task, quantifying their ability to predict symbolic music. This was done by computing the cross entropy loss and perplexity of each model on the test set. Cross-entropy is the dominant measure in machine learning for training and evaluating classification models, and it is also extensively used to evaluate music models (Briot et al., 2020). Intuitively, it measures the difference between two probability distributions: the target distribution (the actual music sequences), and the predicted one (the model's predictions of these sequences). The exponentiation of the entropy, known as perplexity, brings the metric to a linear scale which makes it easier to interpret.

Evaluating the predictive abilities of a music model concerns several aspects and properties of music, including music structure. This last property becomes more evident with longer sequences, thus pushing a model to make better use of its parameters rather than increasing its complexity. More precisely, a model that learned to predict music sequences of considerable length (e.g. with 5K+ tokens) has potentially learned to relate musical patterns at different time scales rather than memorising all sequence elements. A possible way to do this is by detecting *repetitions* and *variations*, which are key elements of music structure.

4.1 Music dataset and representation

We trained and tested our models on the MAESTRO Piano-e-competition dataset (Hawthorne et al., 2019), a collection of classical piano performances containing 1281 MIDI tracks with average duration of 565 ± 447 seconds. This dataset provides an ideal test-bed for our experiments due to (i) the structural complexity of the compositions; (ii) the inclusion of long music pieces; (iii) the fact that part of Western classical understanding of musical form lies in performative interpretation of a score. Each track is quantised to the 16th note, transposed to all keys, and encoded as done in the `BachBot` (Liang et al., 2017), with the notes of chords unrolled in time and separated by specific events. Although this approach elongates sequences, it makes it easier to model polyphony, which should otherwise be learned as the joint probability distribution of all pitches being played (or released) at the same time step – the approach of piano-roll representations. In contrast to other works, we do not trim nor discard the resulting sequences. Indeed, we want to study our models on entire music pieces, thereby testing their ability to learn structural properties of music at all the possible time scales.

The dataset is split into training, validation and test sets, with the last two partitions receiving 10% of the tracks respectively. Overall, the dataset contains 2781 sequences with average length of 5074 ± 1471 music events.

4.2 Implementation Details

To ensure comparability, all the transformer-based architectures make use of the linear self-attention mechanism (cf. Section 3.2). For the LSTM models, a stack of 3 LSTMs each with hidden dimension 256 was used. The transformer layers all have 8 attention heads, and all universal transformer layers perform 4 depth-recurrence steps. All models begin with an embedding layer of dimension 32 following the input layer, and used dropout rate of 10%.

Models were trained to minimise the cross-entropy loss with label smoothing set to 0.1 and gradient norm clipping at 0.1. We used the Adam optimiser with the learning rate schedule from the original transformer paper (Vaswani et al., 2017), together with an early stopping policy (with 10 epochs of patience, and a delta of 10^{-2}) to prevent over-fitting. Our models were implemented in Python

3.7, using Tensorflow 2.2 (Abadi et al., 2015). For data processing, we used Librosa 0.8 (McFee et al., 2020) and `pretty_midi` (Raffel & Ellis, 2014).

5 Results

5.1 Learning to predict music

The results pertaining to the evaluation of the models are shown in Table 1.

From the results of a Kruskal-Wallis H-test, we found that the distributions of the cross-entropy evaluations associated to the five music models under analysis differ significantly ($\chi^2 = 682.24$), with p-value less than 0.0001. Post-hoc multiple comparisons (Kolmogorov-Smirnov tests) were then performed to detect significant differences between each pair of models (Bonferroni corrections were applied to account for multiple comparisons). From these tests, we found that only the cross-entropy evaluations of the Transformer and the UT were not statistically significant ($p = 0.61$), whereas those of all the others were ($p < 0.05$).

Results show that the LSTUT achieved the best music modelling performance, and statistically outperformed all other models. The use of an attention mechanism in the LSTM improved its modelling performance, thereby confirming previous results (Waite, 2016). All the transformer-based networks outperformed the LSTMs, with the UT achieving comparable results to the vanilla transformer with only half of the parameters.

5.2 Learning music structure features

To analyse the extent to which the LSTUT can learn features related to music structure, we manually inspected the attention distributions of its attention heads on a given piece of music. The attention distributions are then compared to the self-similarity matrix (SSM) computed on the same piece. SSMs are a powerful tool extensively used in music structure analysis. To compute an SSM, the music signal is divided into a sequence of feature vectors and all elements of the sequence are compared with each other for similarity. When visualising an SSM, repetitions typically yield *path-like structures*, whereas homogeneous regions yield *block-like structures* (Müller, 2015).

For this analysis, we chose "Fly" by Ludovico Einaudi – a minimalist composition for piano. This contemporary music piece is well suited for our analysis due

Table 1. Music modelling evaluation on the test set.

Model	Cross Entropy	Perplexity	No. of parameters
Basic LSTM	3.42	148.4	1420K
Attention LSTM	3.31	107.8	1521K
Transformer	3.08	109.5	1001K
UT	3.07	151.3	540K
LSTUT	**2.44**	**61.8**	1308K

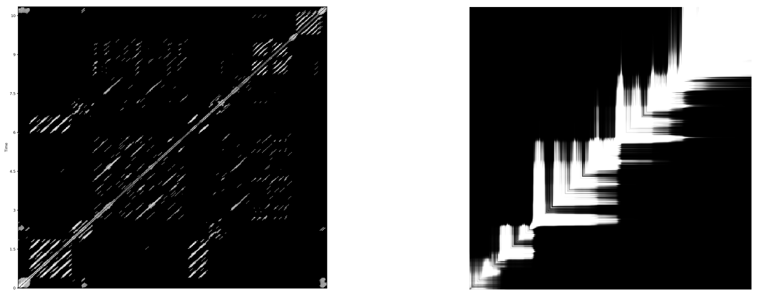

Fig. 2. A SSM of "Fly" (*Left*) compared with the attention distribution of a selected head in the LSTUT (*Right*), demonstrating that structural features are learned.

to its rich and diverse structure. The SSM of this track is shown in Figure 5.2, highlighting the presence of both local and global structural components. This is complemented by the visualisations of a selection of the attention heads of our model, obtained when the track is autoregressively fed to the network.

6 Discussion and Conclusions

In this paper, we put forth the hypothesis that modelling music – the general task on which music models are trained for generation, necessitates both recurrence and attention. These are key architectural components to learn the short- and long- term dependencies arising from the structural complexity of real music. We introduced the *long-short term universal transformer* (LSTUT), a novel architecture combining recurrence and attention paradigms, and integrating computational improvements to handle long sequences. To test our hypothesis, we pushed the boundaries of state of the art music models – variations of LSTMs and transformer architectures, to predict entire classical music pieces spanning over several minutes. We also analysed the attention distributions of the LSTUT on a structurally complex piece of music and compared them to its self-similarity matrix (SSM) – a tool extensively used in music structure analysis.

Our results showed that the LSTUT significantly outperformed our baselines on the music modelling task, and can learn features related to music structure. In this work, we focused primarily on learning music structure, and our generations still require more research effort. This implies that learning multiple music properties is still an open problem, and a gap between modelling and generating music exists. Nonetheless, a model failing to learn long- and short-term structure cannot be expected to generate music with these properties, which are instead peculiar to real compositions. In our future work, we will expand the range of musical styles and interpretations used to test our models, and attempt to bridge the gap between what a music model can learn and what it can actually generate.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2015). *TensorFlow: Large-scale machine learning on heterogeneous systems.* Retrieved from `http://tensorflow.org/` (Software available from tensorflow.org)

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Bengio, Y., Simard, P., & Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, *5*(2), 157–166.

Briot, J.-P., Hadjeres, G., & Pachet, F.-D. (2020). *Deep learning techniques for music generation.* Springer.

Choi, K., Hawthorne, C., Simon, I., Dinculescu, M., & Engel, J. (2019). *Encoding musical style with transformer autoencoders.*

Clevert, D.-A., Unterthiner, T., & Hochreiter, S. (2015). *Fast and accurate deep network learning by exponential linear units (elus).*

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., & Łukasz Kaiser. (2018). *Universal transformers.*

Eck, D., & Schmidhuber, J. (2002). A first look at music composition using lstm recurrent neural networks. *Istituto Dalle Molle Di Studi Sull Intelligenza Artificiale*, *103*, 48.

Elliot, W. (2016). *Generating Long-Term Structure in Songs and Stories.* `https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/`. (Accessed: 2020-07-02)

Hawthorne, C., Stasyuk, A., Roberts, A., Simon, I., Huang, C.-Z. A., Dieleman, S., … Eck, D. (2019). Enabling factorized piano music modeling and generation with the MAESTRO dataset. In *International conference on learning representations.* Retrieved from `https://openreview.net/forum?id=r1lYRjC9F7`

Hochreiter, S., & Schmidhuber, J. (1997, 12). Long short-term memory. *Neural computation*, *9*, 1735-80. doi: 10.1162/neco.1997.9.8.1735

Huang, C.-Z. A., Vaswani, A., Uszkoreit, J., Shazeer, N., Simon, I., Hawthorne, C., … Eck, D. (2018). *Music transformer.*

Johnson, D. D. (2017). Generating polyphonic music using tied parallel networks. In *International conference on evolutionary and biologically inspired music and art* (pp. 128–143).

Katharopoulos, A., Vyas, A., Pappas, N., & Fleuret, F. (2020). *Transformers are rnns: Fast autoregressive transformers with linear attention.*

Koelsch, S., Rohrmeier, M., Torrecuso, R., & Jentschke, S. (2013). Processing of hierarchical syntactic structure in music. *Proceedings of the National Academy of Sciences*, *110*(38), 15443–15448.

Li, J., Tu, Z., Yang, B., Lyu, M. R., & Zhang, T. (2018, October-November). Multi-head attention with disagreement regularization. In *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 2897–2903). Brussels, Belgium: Association for Computational Linguistics. Retrieved from https://www.aclweb.org/anthology/D18-1317 doi: 10.18653/v1/D18-1317

Liang, F., Gotham, M., Johnson, M., & Shotton, J. (2017, October). Automatic stylistic composition of bach chorales with deep lstm. In *18th international society for music information retrieval conference* (18th International Society for Music Information Retrieval Conference ed.). Retrieved from https://www.microsoft.com/en-us/research/publication/automatic-stylistic-composition-of-bach-chorales-with-deep-lstm/

McFee, B., Lostanlen, V., Metsai, A., McVicar, M., Balke, S., Thomé, C., ... Kim, T. (2020, July). *librosa/librosa: 0.8.0.* Zenodo. Retrieved from https://doi.org/10.5281/zenodo.3955228 doi: 10.5281/zenodo.3955228

Müller, M. (2015). *Fundamentals of music processing: Audio, analysis, algorithms, applications.* Springer.

Raffel, C., & Ellis, D. P. (2014). Intuitive analysis, creation and manipulation of midi data with pretty midi. In *15th international society for music information retrieval conference late breaking and demo papers* (pp. 84–93).

Sturm, B., Santos, J. F., & Korshunova, I. (2015). Folk music style modelling by recurrent neural networks with long short term memory units. In *16th international society for music information retrieval conference.*

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... Polosukhin, I. (2017). *Attention is all you need.*

Waite, E. (2016). *Generating long-term structure in songs and stories.* Retrieved 2020-08-04, from https://magenta.tensorflow.org/2016/07/15/lookback-rnn-attention-rnn/

Wang, Z., Ma, Y., Liu, Z., & Tang, J. (2019). *R-transformer: Recurrent neural network enhanced transformer.*

Wu, X., Wang, C., & Lei, Q. (2020). *Transformer-xl based music generation with multiple sequences of time-valued notes.*

A Linear Attention Details

Here we give a derivation of the linear attention mechanism from (Katharopoulos et al., 2020). Suppose that for a particular attention head we have queries and keys $q_i, k_i \in \mathbb{R}^m$, and values $v_i \in \mathbb{R}^n$. If we define the attention that query q_i pays to key k_j to be

$$\frac{\phi(q_i)^T \phi(k_j)}{\sum_l \phi(q_i)^T \phi(k_l)} \tag{2}$$

for some function ϕ (applied element-wise to the vectors, and which is strictly positive) [4] then the resulting value for this attention head at sequence position i will be

$$\frac{\sum_j \phi(q_i)^T \phi(k_j) v_j}{\sum_j \phi(q_i)^T \phi(k_j)} \tag{3}$$

and factorising this gives:

$$\left(\frac{\phi(q_i)^T \sum_j \phi(k_j) v_j^T}{\phi(q_i)^T \sum_j \phi(k_j)} \right)^T \tag{4}$$

and finally observe that both of the summands above do not depend on i and thus can be computed *once* and shared between the calculations for *all positions in the sequence*. In order to prevent the model from attending to the future, instead of summing over *all* sequence positions in the above, we compute the cumulative sum (which can still be done in linear time). See listing 1.1 for an implementation of this attention mechanism in TensorFlow.

Listing 1.1. Implementation of linear attention in Python 3.7 and TensorFlow 2.2.0

```
# Suppose 'keys', 'queries' are of shape
# num_attention_heads * batch_sz * seq_len * M
# and 'values' are of shape
# num_attention_heads * batch_sz * seq_len * N
numerator_sum_elements = tf.einsum('ijkm,ijkn->ijknm',
                                   keys, values)
numerator = tf.cumsum(numerator_sum_elements, axis=2)
numerator = tf.einsum('ijknm,ijkm->ijkn', numerator,
                      queries)
denominator = tf.einsum('ijkm,ijkm->ijk',
                        tf.cumsum(keys, axis=2),
                        queries)
denominator = tf.expand_dims(denominator, axis=-1)
output = numerator / (denominator + 1.0e-6)
```

[4] We follow Katharopoulos et al. and take $\phi(x) := \texttt{ELU}(x) + 1$ (Clevert, Unterthiner, & Hochreiter, 2015).