Linear RNNs for autoregressive generation of long music samples

Konrad Szewczyk*
University of Amsterdam
konrad.szewczyk@student.uva.nl

Daniel Gallo Fernández*
University of Amsterdam
daniel.gallo.fernandez@student.uva.nl

James Townsend
University of Amsterdam
j.h.n.townsend@uva.nl

Abstract

Directly learning to generate audio waveforms in an autoregressive manner is a challenging task, due to the length of the raw sequences and the existence of important structure on many different timescales. Traditional approaches based on recurrent neural networks, as well as causal convolutions and self-attention, have only had limited success on this task. However, recent work has shown that deep state space models, also referred to as linear RNNs, can be highly efficient in this context. In this work, we push the boundaries of linear RNNs applied to raw audio modeling, investigating the effects of different architectural choices and using context-parallelism to enable training on sequences up to one minute (1M tokens) in length. We present a model, HarmonicRNN, which attains state of the art log-likelihoods and perceptual metrics on small-scale datasets.

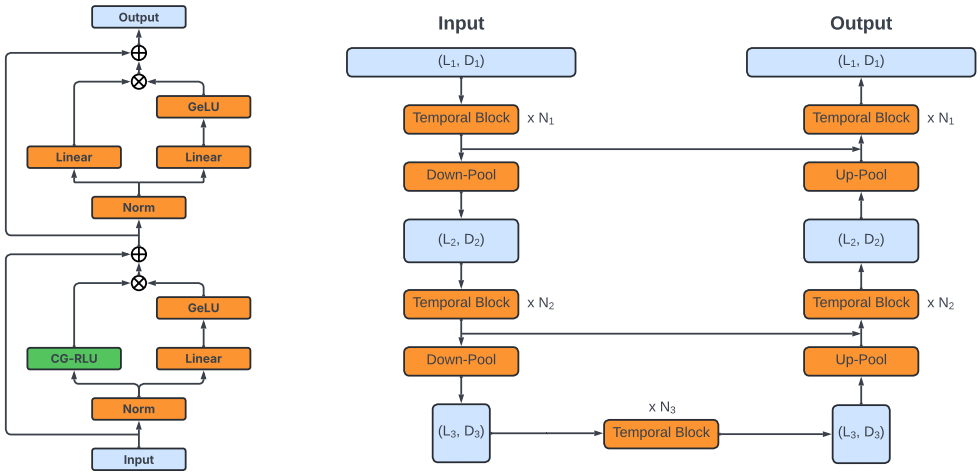
1 Introduction

Sequence-to-sequence models have become central in artificial intelligence, particularly following the introduction of the transformer architecture. While initially developed for natural language processing, these models have demonstrated utility across various domains. A notable example is computer vision, where vision transformers are increasingly displacing traditional convolutional neural networks. Sequence-to-sequence models require mechanisms to exchange information along the time dimension, typically using recurrent or self-attention layers. Recurrent layers need to compress the past into a fixed-size state, while self-attention uses a state with size growing linearly with the sequence length. Using T to denote sequence length, the time complexity of recurrent layers is $O(T)$. Self-attention, on the other hand, is $O(T^2)$ but is easily parallelizable at training time. However, the quadratic complexity of self-attention becomes a serious problem at inference (generation) time, particularly for long sequences, which commonly occur in audio modeling.

Recent work by Gu et al. (2021), Goel et al. (2022), and Orvieto et al. (2023) has shown that recurrent neural network (RNN) layers with a *linear* recurrence can be effective for long sequence modeling, as well as enabling training time parallelism (Smith et al., 2022). In this work we propose HarmonicRNN, a sequence-to-sequence model that uses linear recurrent layers with pooling to reduce the effective sequence length, inspired by the SaShiMi model introduced by Goel et al. (2022). We demonstrate HarmonicRNN on autoregressive modeling of raw audio, and use multi-host context parallelism to enable training directly on sequences up to 1M tokens (1 minute at 16 kHz) in length. We show that pooling is necessary in order to attain coherent sounding samples over a long timescale, and report state of the art log-likelihood and perceptual metrics on small audio benchmark datasets.

*Equal contribution.

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: AI for Music.



(a) Temporal block. (b) Overall architecture of the HarmonicRNN, here with two pooling layers.

Figure 1: Data flow graphs for the HarmonicRNN.

2 Method

We apply maximum likelihood training directly to audio data (although some quantization preprocessing was performed, see section 4.1). That is, we maximize the log-probability

$$L(\theta) := \log p(x; \theta) = \sum_t \log p(x_t \,|\, x_1, \ldots, x_{t-1}; \theta) \tag{1}$$

using stochastic gradient ascent on mini-batches. The conditional probability distributions $p(x_t \,|\, x_1, \ldots, x_{t-1}; \theta)$ are implemented using a deep linear recurrent neural network (RNN), which, for each t, outputs a categorical distribution over x_t depending only on $x_1, \ldots, x_{t-1}$ and θ. The 'linear' in linear RNN refers to the hidden state recurrence, which in fact has the form

$$h_t = a(u_t; \theta) \odot h_{t-1} + b(u_t; \theta), \tag{2}$$

where h_t and u_t are the hidden state and input, respectively, at time t; a and b are (possibly nonlinear) functions and $\odot$ denotes element-wise multiplication of vectors. This form allows efficient parallelized implementation at training time using an associative scan, and has been shown to have excellent performance (Goel et al., 2022; Gu and Dao, 2024; De et al., 2024). We use a particular setting of a and b based on the 'complex gated linear recurrent unit' (CG-LRU), for details see De et al. (2024) and Botev et al. (2024). For details on the parallelized associative scan algorithm see Blelloch (1991). The CG-LRU is wrapped in a 'temporal block', the overall structure of which is based on the transformer architecture, with self-attention replaced by the CG-LRU, as shown in fig. 1a.

On a higher level, we also enable the HarmonicRNN to model different timescales by using temporal down and up-pooling operations, with an overall architecture visualized in fig. 1b. This is inspired by the SaShiMi architecture introduced in Goel et al. (2022). Our model differs from SaShiMi in the following ways:

1. Use of the CG-LRU as the core recurrent layer, where SaShiMi used a layer based on the earlier S4 architecture of Gu et al. (2021).
2. We use strided convolutions for down-pooling and dilated (sometimes called transposed strided) convolutions for up-pooling. Crucially, we found that setting the number of feature groups greater than 1 greatly improved stability (see table 1). This differs from SaShiMi, which used reshape with a dense layer, allowing dense interactions between input/output features.
3. We used a non-learned embedding layer, with sinusoids of different periods (inspired by Kingma et al., 2021; Appendix C), instead of a learned embedding layer. We found that this led to faster training (see fig. 2).

3 Related work

State space models and linear RNNs This work builds on recent progress using state space models (from classical control theory) as layers in a deep neural network. This line of work was initiated by Gu et al. (2021), and further developed by Smith et al. (2022), Gu and Dao (2024), Orvieto et al. (2023), and De et al. (2024), among others.

Autoregressive generation of raw audio The major models proposed for this task are SampleRNN (Mehri et al., 2017); WaveNet and its derivatives (van den Oord et al., 2016; van den Oord et al., 2018; Kalchbrenner et al., 2018); and recently SaShiMi (Goel et al., 2022; Gu and Dao, 2024). SampleRNN and WaveNet use (nonlinear) RNNs and causal convolutions, respectively. SaShiMi was the first model where state space models—effectively equivalent to linear RNNs (Orvieto et al., 2023)—were applied to audio modeling, with state of the art results.

4 Experiments

We performed ablation experiments to evaluate various aspects of our model in terms of log-likelihood, perceptual metrics and inference speed.

4.1 Datasets

We use the same three datasets and train / test splits from Goel et al. (2022). All of them are audio recordings sampled at 16 kHz with 16-bit linear PCM encoding.

1. **SC09** (Donahue et al., 2018; Warden, 2018), which consists of 1-second (16k token) recordings of the spoken digits zero to nine (similar to MNIST, but with audio instead of images). For this dataset, we also compute the FID and IS scores using the SaShiMi repo (Goel et al., 2022).
2. **Beethoven** (Mehri et al., 2017), which contains 8-second (128k token) recordings of Beethoven's piano sonatas.
3. **YouTubeMix** (Kozakowski, 2017), which contains 1-minute (960,512 token) recordings of piano playing.

Following Goel et al. (2022), we apply μ-law encoding to SC09 and YouTubeMix, and linear encoding to Beethoven, and quantize to an unsigned 8-bit representation. All of the datasets are available on HuggingFace datasets: Beethoven, YouTubeMix, and SC09.

4.2 Baseline model and training setup

We used the JAX, Flax and Optax libraries to implement our experiments and ran them on TPU v4-8 and v3-128 devices. For training on the minute-long samples in YouTubeMix, we needed to use multi-host training, because of the need for more TPU device memory. We used the custom CG-LRU TPU kernels from the RecurrentGemma repository, which support context parallelism (parallelism over the sequence axis) to split the model across devices.

Model We use the model structure visualized in fig. 1b, with four levels of down-pooling. The down-pooling factors (from outer to inner) are [2, 4, 4, 5], meaning that at the innermost layer, the sequence length is $2 \times 4 \times 4 \times 5 = 160$ times smaller than the data sequence. The number of temporal mixing blocks between the pooling layers is 4 at every level, in both the down-pooling and up-pooling parts of the network, leading to a total of 36 temporal blocks. We use a feature width of 128, and an RNN hidden dimension of 256. This setup gives a model with 7.3M parameters.

Training We use the AdamW optimizer (Loshchilov and Hutter, 2018) with a learning rate of 0.002 and a weight decay of 0.0001. All other hyperparameters are set to the default values in Optax: $\beta_1 = 0.9$, $\beta_2 = 0.999$. We used a constant learning rate schedule after 1,000 steps warm-up, batch size 32 and 500 training epochs. We evaluated on the test sets. We used exponential moving average (EMA) for the model weights, with rate equal to 0.999.

4.3 Effect of input embedding

As described in section 4.1, the model's input is an array of 8-bit unsigned integers. We tried four approaches to embed data before feeding it to the neural network; fig. 2 shows a comparison of training curves. 'Linear scaling' was simply scaling the data into the real interval [-1, 1] (this technique is used by e.g. WaveNet). The sinusoidal embeddings were sinusoids of different frequencies applied to the input integers, based on Kingma et al. (2021, Appendix C). Finally, we tried a standard embedding layer, as is commonly used in transformer architectures, with and without dropout. We found that the sinusoidal embeddings consistently performed better, with significantly faster convergence and better final loss value.

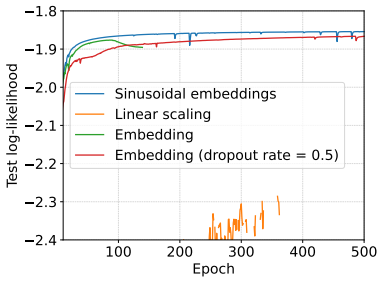


Figure 2: Training curves with different embedding methods (SC09 dataset).

4.4 Effect of pooling

# Groups	# Params	Test NLL ↓
1 (dense)	7.8M	1.956
4	7.4M	**1.848**
128 (diagonal)	7.3M	1.854

Table 1: Effect of number of feature groups in pooling convolutions on total parameter count and test NLL (SC09 dataset).

Pooling	Train speed	Infer throughput	NLL ↓	FID ↓	IS ↑
Baseline	**11.5 epoch/h**	**103 ktok/s**	1.854	**0.46**	**6.46**
1 pool lyr.	6.7 epoch/h	55 ktok/s	1.853	6.85	2.03
No pooling	6.1 epoch/h	46 ktok/s	**1.852**	2.95	3.33

Table 2: Effect of pooling configuration on training and inference speed; NLL, FID and IS performance metrics. Total layer count is held constant. The model with 1 pooling layer has pooling factor 2 with 24 temporal blocks outside the pooling and 12 within.

We measured the effect on performance of the group count in the pooling convolutions (Table 1), and the effect on training and inference time, as well as other metrics, of removing pooling altogether (Table 2). For the group count, we found that the best performing in negative log-likelihood (NLL) was 4 groups, though this had slightly more parameters and had a slower training runtime than the fully diagonal convolution. The effect of reducing or removing pooling is to significantly reduce training and inference speed, very slightly decrease NLL, and to worsen significantly the perceptual metrics Frechét Inception Distance (FID) and Inception Score (IS). We used the recently developed Scanagram library to implement efficient inference (Townsend, 2025). Inference batch size and sequence length were the same as during training.

4.5 Comparison to prior state of the art

In table 3, we compare our best configurations with SaShiMi and Mamba (the prior state of the art) on the SC09 dataset. Versions of HarmonicRNN achieved state of the art results on all metrics. Table 4 compares with SaShiMi on the other two datasets. Unlike SaShiMi, we trained on full minute-long YouTubeMix sequences. In the supplementary material we provide eight minute-long, non-cherry-picked samples, which compare favourably with the SaShiMi samples at `https://hazyresearch.stanford.edu/sashimi-examples/#music`.

Model	# Params	Test NLL ↓	FID ↓	IS ↑
HarmonicRNN baseline	7.3M	1.854	**0.46**	**6.46**
HarmonicRNN 4 conv grps.	7.4M	**1.848**	–	–
SaShiMi	5.8M	1.873	1.99	5.13
Mamba	6.1M	1.852	0.94	6.26

Table 3: Comparison of our models with the prior state-of-the-art (SC09 dataset). Note that the results for SaShiMi are those reported in Gu and Dao (2024), which differ from those in the original SaShiMi paper (Goel et al., 2022).

5 Conclusion

We have presented initial results for HarmonicRNN, a deep linear RNN designed to generate music in an autoregressive manner. The model shows promising performance and we look forward to future work scaling the approach up to more challenging audio generation tasks.

Model	Beethoven ↓	YouTubeMix ↓
HarmonicRNN	**0.915**	1.324
SaShiMi	0.946	**1.294**

Table 4: Comparison of our HarmonicRNN test NLL with the prior state of the art on Beethoven and YouTubeMix.

6 Acknowledgments

James Townsend acknowledges funding from the Dutch Research Council (NWO) under Veni project VI.Veni.212.106, and the European Commission under MSCA project NNESCI. This work was done as part of an MSc thesis project by the two first authors. We would like to thank Jan-Willem van de Meent for acting as examiner for the project and for helpful discussions.

References

Blelloch, Guy (1991). Prefix Sums and Their Applications. In *Synthesis of Parallel Algorithms*. Morgan Kaufmann, pp. 35–60.

Botev, Aleksandar et al. (2024). *RecurrentGemma: Moving Past Transformers for Efficient Open Language Models*. arXiv: 2404.07839.

De, Soham et al. (2024). *Griffin: Mixing Gated Linear Recurrences with Local Attention for Efficient Language Models*. arXiv: 2402.19427.

Donahue, Chris, McAuley, Julian, and Puckette, Miller (2018). Adversarial Audio Synthesis. In International Conference on Learning Representations.

Goel, Karan, Gu, Albert, Donahue, Chris, and Re, Christopher (2022). It's Raw! Audio Generation with State-Space Models. In *Proceedings of the 39th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 7616–7633.

Gu, Albert and Dao, Tri (2024). Mamba: Linear-Time Sequence Modeling with Selective State Spaces. In First Conference on Language Modeling.

Gu, Albert, Goel, Karan, and Re, Christopher (2021). Efficiently Modeling Long Sequences with Structured State Spaces. In International Conference on Learning Representations.

Kalchbrenner, Nal, Elsen, Erich, Simonyan, Karen, Noury, Seb, Casagrande, Norman, Lockhart, Edward, Stimberg, Florian, van den Oord, Aäron, Dieleman, Sander, and Kavukcuoglu, Koray (2018). Efficient Neural Audio Synthesis. In *Proceedings of the 35th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 2410–2419.

Kingma, Diederik, Salimans, Tim, Poole, Ben, and Ho, Jonathan (2021). Variational Diffusion Models. In *Advances in Neural Information Processing Systems*. Vol. 34, pp. 21696–21707.

Kozakowski, Piotr (2017). *samplernn-pytorch*. URL: https://github.com/deepsound-project/samplernn-pytorch.

Loshchilov, Ilya and Hutter, Frank (2018). Decoupled Weight Decay Regularization. In International Conference on Learning Representations.

Mehri, Soroush, Kumar, Kundan, Gulrajani, Ishaan, Kumar, Rithesh, Jain, Shubham, Sotelo, Jose, Courville, Aaron, and Bengio, Yoshua (2017). SampleRNN: An Unconditional End-to-End Neural Audio Generation Model. In International Conference on Learning Representations.

Orvieto, Antonio, Smith, Samuel L., Gu, Albert, Fernando, Anushan, Gulcehre, Caglar, Pascanu, Razvan, and De, Soham (2023). Resurrecting Recurrent Neural Networks for Long Sequences. In *Proceedings of the 40th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 26670–26698.

Smith, Jimmy T. H., Warrington, Andrew, and Linderman, Scott (2022). Simplified State Space Layers for Sequence Modeling. In International Conference on Learning Representations.

Townsend, James (2025). *Scanagram*. URL: https://github.com/j-towns/scanagram.

Van den Oord, Aäron, Dieleman, Sander, Zen, Heiga, Simonyan, Karen, Vinyals, Oriol, Graves, Alex, Kalchbrenner, Nal, Senior, Andrew, and Kavukcuoglu, Koray (2016). *WaveNet: A Generative Model for Raw Audio*. arXiv: 1609.03499.

Van den Oord, Aäron et al. (2018). Parallel WaveNet: Fast High-Fidelity Speech Synthesis. In *Proceedings of the 35th International Conference on Machine Learning*. International Conference on Machine Learning. PMLR, pp. 3918–3926.

Warden, Pete (2018). *Speech Commands: A Dataset for Limited-Vocabulary Speech Recognition*. arXiv: 1804.03209.