Recurrent Transformers

B.N. Kausik[1]

June 23 2025

Abstract

We propose recurrent transformers, a neural network architecture that combines the simplicity and efficiency of recurrent networks with the efficacy of transformers. Specifically, a recurrent transformer is a transformer that is progressively applied to a fixed-width sliding window across the input sequence, thereby operating in linear time in the length of the sequence during both training and inference. In our experiments on a smaller natural language data set, the recurrent transformer outperforms the corresponding multi-layer transformer. Experiments at larger scales are required to validate the general applicability of the approach.

[1]Author: https://sites.google.com/view/bnkausik/ Contact: bnkausik@gmail.com
Thanks to J. Jawahar, R. Kannan and P. Tadepalli for comments and suggestions

Introduction

Transformers, Vaswani et al (2017), have found wide use in learning and modeling human perception. Since causal attention, the core computation in a transformer, is quadratic in the length of the input, a broad range of approaches have been proposed to reduce the time and space complexity of transformers in practical applications. The approaches range across model pruning e.g., LeCun et al., (1989), Hassibi et al, (1993) Han et al, (2015), Liu et al, (2019), Blalock et al, (2020), Heoffler et al (2022), Sun et al (2023), and Frantar & Alistarh (2023); model distillation, e.g., Hinton et al, (2015), Chen et al (2017) and Asami et al (2017); modified gradient descent, Kausik (2024); as well as algorithms that directly reduce the computational complexity of calculating attention, e.g., Parmar et al., (2018) Child et al., (2019), Beltagy et al. (2020), Kitaev et al. (2020), Tay et al., (2020), Wang et al., (2020), Bello et al. (2021) Choromanski et al., (2021), Peng et al., (2021), Xiong et al., (2021), Ma et al., (2021), Zheng et al., (2022), Alman and Song, (2023), Ma et al, (2023), Han et al., (2024), Ma et al., (2024), and Kannan et al (2024).

Preceding transformers, Recurrent Neural Networks (RNNs), e.g., Elman (1990), their variants Long Short-Term Memory (LSTM), e.g., Hochreiter & Schmidhuber (1997) and Gated Recurrent Units (GRUs), e.g., Cho et al., (2014), were popular for linear time processing of sequential data streams such as natural language processing. More recent architectural alternatives include State Space Models, e.g., Gu et al (2021), and minimal RNNs, Feng et al (2024).

Building on prior work, we propose recurrent transformers, combining the simplicity and efficiency of recurrent networks with the efficacy of transformers. Specifically, a recurrent transformer is a transformer that is progressively applied to a fixed-width sliding window across the input sequence, thereby operating in linear time in the length of the sequence during both training and inference. In our experiments with the nanoGPT model of Karpathy (2024) on the Shakespeare data set, we find that the recurrent transformer outperforms the corresponding multi-layer transformer.

Architecture

Let $(x_1, \ldots x_i, \ldots x_N)$ be an input sequence. Let T denote the function computed by a transformer in that on input X the transformer outputs $T(X)$. The recurrent transformer based on T is as follows:

$$h_1 = T(x_1) \qquad (1)$$

$$(y_i, h_{i+1}) = T(h_i, x_{i+1}) \qquad (2)$$

$$y_N = T(h_N) \qquad (3)$$

In the above, h_i and y_i are the hidden state and output at position i respectively. Equation (1) initializes the recurrence at position 1, Equation (2) iteratively applies the transformer function with causal attention between position $i + 1$ and position i, and Equation (3) closes the recurrence at the last position.

Fig. 1 is a visual representation of the recurrent transformer, where the bottom row of the table shows an input sequence x_1, x_2, x_3, x_4 of five tokens. Per Equation (1), the transformer function T is first applied to input x_1 to produce the hidden state h_1. Then per Equation (2), the transformer function is applied to the sequence (h_1, x_2) to obtain (y_1, h_2). And so on, capping the recurrence per Equation (3) at the last position.

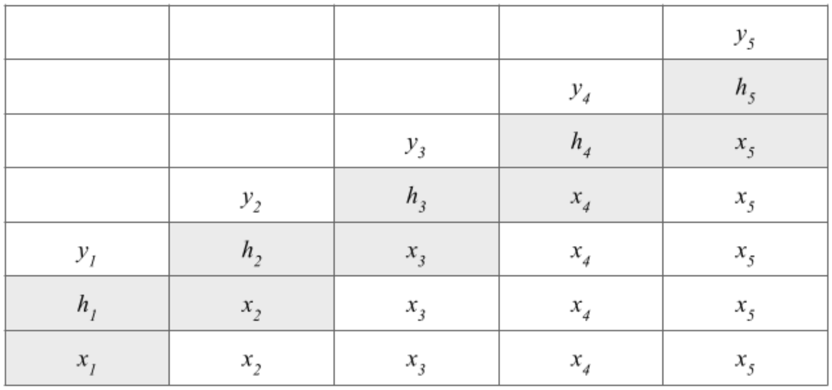

Fig. 1: Visual representation of the recurrent transformer architecture. Each shaded box represents an application of the underlying transformer layer function to produce the token(s) in the box(es) above.

Optionally, a recurrent transformer may have multiple layers, and larger sliding window widths.

Experimental Results

We compare the performance of the recurrent transformer against regular transformers on the nanoGPT Shakespeare data set of Karpathy (2023). Table 1 specifies the base transformer layer in our comparison.

Table 1: Base Transformer Layer		
Context length	**Embedding Dimension**	**Number of heads**
256	384	6

Table 2 shows the results of our experiments comparing the performance of a 1-layer and 6-layer regular transformer operating on the full sequence length, against a 1-layer recurrent transformer with and without positional embeddings added to the token embeddings.

Table 2: Experimental Results			
Model	**Parameters**	**Validation Loss**	**Standard Error**
Regular Transformer - 1 layer	1.89M	1.5697	2.23E-04
Regular Transformer - 6 layers	10.74M	1.4815	3.67E-04
Recurrent Transformer with positional embedding	1.89M	1.4738	2.16E-04
Recurrent Transformer without positional embedding	1.79M	1.4699	2.23E-04

Figs (2a) and (2b) show the validation loss and training loss for the four models in Table 2. The horizontal axis is the number of input tokens in millions, while the vertical axis is the logarithmic loss.

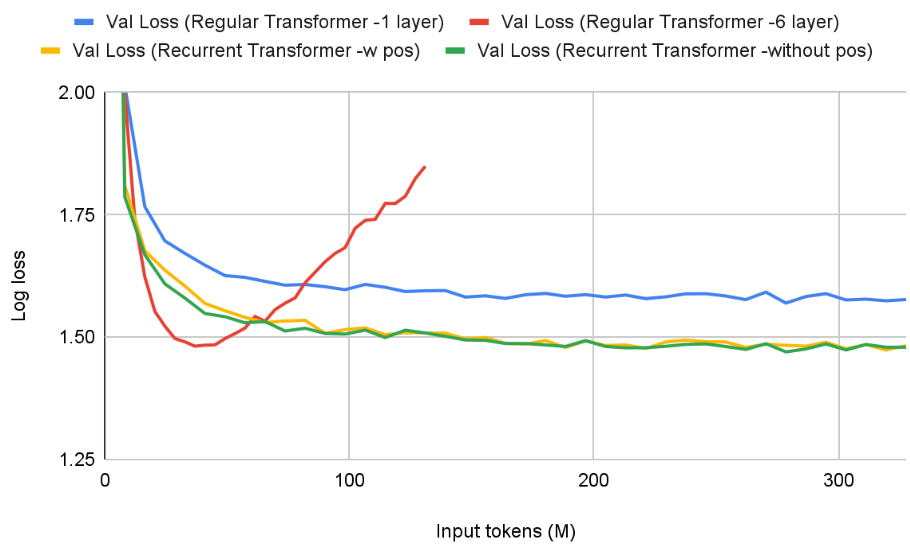

Fig. 2(a): Validation loss for the four models of Table 2.

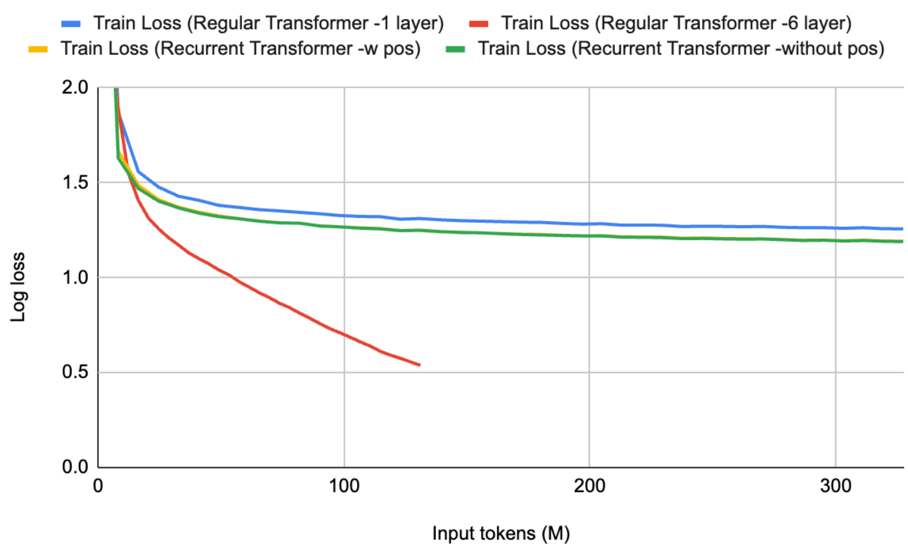

Fig. 2(b): Training loss for the four models of Table 2.

All of our experiments were run on an M4 Mac Mini with 16GB of memory. Code available at https://github.com/bnkausik/recurrent_transformer

Summary

We propose recurrent transformers, a neural network architecture that combines the simplicity and efficiency of recurrent networks with the efficacy of transformers. Specifically, a recurrent transformer is a transformer that is progressively applied to a fixed-width sliding window across the input sequence, thereby operating in linear time in the length of the sequence during both training and inference. In our experiments on a smaller natural language data set, the recurrent transformer with or without positional embeddings outperforms the corresponding multi-layer transformer. Experiments at larger scales are required to validate the general applicability of the approach.

References

1. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30
2. LeCun, Yann, John Denker, and Sara Solla. "Optimal brain damage." *Advances in neural information processing systems* 2 (1989).
3. Hassibi, B., Stork, D. G., & Wolff, G. J. (1993, March). Optimal brain surgeon and general network pruning. In *IEEE international conference on neural networks* (pp. 293-299). IEEE.
4. Han, S., Pool, J., Tran, J., & Dally, W. (2015). Learning both weights and connections for efficient neural network. *Advances in neural information processing systems*, *28*.
5. Liu, Z., Sun, M., Zhou, T., Huang, G., & Darrell, T. (2018). Rethinking the value of network pruning. *arXiv preprint arXiv:1810.05270*.
6. Blalock, D., Gonzalez Ortiz, J. J., Frankle, J., & Guttag, J. (2020). What is the state of neural network pruning?. *Proceedings of machine learning and systems*, *2*, 129-146
7. Sun, M., Liu, Z., Bair, A., & Kolter, J. Z. (2023). A simple and effective pruning approach for large language models. *arXiv preprint arXiv:2306.11695*.
8. Hoefler, T., Alistarh, D., Ben-Nun, T., Dryden, N., & Peste, A. (2021). Sparsity in deep learning: Pruning and growth for efficient inference and training in neural networks. Journal of Machine Learning Research, 22(241), 1-124.
9. Frantar, E., & Alistarh, D. (2023, July). Sparsegpt: Massive language models can be accurately pruned in one-shot. In *International Conference on Machine Learning* (pp. 10323-10337). PMLR.
10. Hinton, G., Vinyals, O., & Dean, J. (2015). Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*.
11. Chen, G., Choi, W., Yu, X., Han, T., & Chandraker, M. (2017). Learning efficient object detection models with knowledge distillation. *Advances in neural information processing systems*, *30*.
12. Asami, T., Masumura, R., Yamaguchi, Y., Masataki, H., & Aono, Y. (2017, March). Domain adaptation of DNN acoustic models using knowledge distillation. In *2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)* (pp. 5185-5189). IEEE.
13. Kausik, B. N. (2024). Occam Gradient Descent. arXiv preprint arXiv:2405.20194.
14. Niki Parmar, Ashish Vaswani, Jakob Uszkoreit, Lukasz Kaiser, Noam Shazeer, Alexander Ku, and Dustin Tran. Image transformer. In International Conference on Machine Learning, pages 4055–4064. PMLR, 2018.
15. Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. Generating long sequences with sparse transformers. arXiv preprint arXiv:1904.10509, 2019.
16. Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. arXiv preprint arXiv:2004.05150, 2020.
17. Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. Reformer: The efficient transformer. arXiv preprint arXiv:2001.04451, 2020.
18. Yi Tay, Dara Bahri, Donald Metzler, Da-Cheng Juan, Zhe Zhao, and Che Zheng. Sparse sinkhorn attention. In International Conference on Machine Learning, pages 9438–9447. PMLR, 2020.
19. Irwan Bello, William Fedus, Xianzhi Du, Yann Dauphin, Ashish Srinivas, Tengyu Zhou, Zihang Barret, and Bryan Zoph. Lambdanetworks: Modeling long-range interactions without attention. In International Conference on Learning Representations, 2021.

20. Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Davis, Afroz Belanger, Lucy Sainath, et al. Rethinking attention with performers. arXiv preprint arXiv:2009.14794, 2021.
21. Hao Peng, Nikolaos Pappas, Dani Yogatama, Roy Schwartz, Noah Smith, and Lingpeng Kong. Random feature attention. arXiv preprint arXiv:2103.02114, 2021.
22. Zhe Zheng, Da-Cheng Juan, Yi Tay, Dara Bahri, and Donald Metzler. Randomized attention with linear complexity. In Advances in Neural Information Processing Systems, 2022.
23. Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. arXiv preprint arXiv:2006.04768, 2020.
24. Yunyang Xiong, Zhanpeng Zeng, Rudrasis Chakraborty, Mingxing Tan, Glenn Fung, Yin Li, and Vikas Singh. Nystr¨omformer: A nystr¨om-based algorithm for approximating self-attention. arXiv preprint arXiv:2102.03902, 2021.
25. Xiaoyu Ma, Yiding Wen, Yuanmeng He, Hai Zhao Liu, Dianhai Liu, Jianhua Li, and Xuan Dong. Nested attention for long document language modeling. arXiv preprint arXiv:2105.11875, 2021.
26. Ma, X., Zhou, C., Kong, X., He, J., Gui, L., Neubig, G., ... & Zettlemoyer, L. (2022). Mega: Moving average equipped gated attention. arXiv preprint arXiv:2209.10655.
27. Josh Alman and Zhao Song. Fast attention requires bounded entries. In Alice Oh, Tristan Naumann, Amir Globerson, Kate Saenko, Moritz Hardt, and Sergey Levine, editors, Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023, 2023
28. Ma, X., Yang, X., Xiong, W., Chen, B., Yu, L., Zhang, H., ... & Zhou, C. (2024). Megalodon: Efficient llm pretraining and inference with unlimited context length. Advances in Neural Information Processing Systems, 37, 71831-71854.
29. Insu Han, Rajesh Jayaram, Amin Karbasi, Vahab Mirrokni, David P. Woodruff, and Amir Zandieh. Hyperattention: Long-context attention in near-linear time. In The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024. OpenReview.net, 2024
30. Kannan, R., Bhattacharyya, C., Kacham, P., & Woodruff, D. P. (2024). LevAttention: Time, Space, and Streaming Efficient Algorithm for Heavy Attentions. arXiv preprint arXiv:2410.05462.
31. Elman, J.L, (1990). Finding structure in time. Cognitive Science, 14(2):179–211, 1990. ISSN 0364-0213.
32. Hochreiter, S and Schmidhuber, J, (1997), Long short-term memory. Neural Computation, 9(8):1735–1780, 1997.
33. Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., & Bengio, Y. (2014). Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv preprint arXiv:1406.1078.
34. Gu, A., Goel, K., & Ré, C. Efficiently modeling long sequences with structured state spaces. arXiv 2021. arXiv preprint arXiv:2111.00396.
35. Feng, L., Tung, F., Ahmed, M. O., Bengio, Y., & Hajimirsadeghi, H. (2024). Were RNNs all we needed?. arXiv preprint arXiv:2410.01201.
36. Karpathy, A., (2023), https://github.com/karpathy/nanoGPT/blob/master/README.md