Benchmarking Recurrent Depth Architectures

Gavin Tseng

Received June 16, 2025
Accepted October 12, 2025
Electronic access November 30, 2025

Recurrent depth models show remarkable promise in improving model performance without scaling up model parameters. Multiple architectures and models have been proposed and trained. However, benchmarking the efficacy of the architecture when model hyperparameters differ makes it difficult to accurately attribute the source of performance differences. Three different recurrent depth architectures are trained on the TinyStories dataset along with a control, and compared to determine the optimal architecture. Results showed that the difference between recurrent architecture performance can be attributed to initialization. However performance compared to the control model was substantial.

Literature Review

Over the past decade, transformer-based models have revolutionized deep learning, achieving state-of-the-art performance across numerous tasks[1,2]. However, their increasing size has led to a substantial increase in the demands for memory and computational resources required, posing significant challenges during training and inference[3–5]. As transformer models continue to advance the state-of-the-art, their growing size has become a significant bottleneck[4]. Larger models require more memory, bandwidth, and compute, making them expensive to deploy, especially in resource-constrained environments such as mobile devices or consumer hardware. Several strategies have been introduced to mitigate these challenges, including programmatic improvements[6–8] mixed precision training[9,10] and architectural improvements[11–14]. These efforts primarily aim to reduce VRAM usage with the memory-bandwidth-intensive demands of large transformer models[4,15]. One promising direction is recurrent depth transformer architectures. By reusing model parameters through recursion, allowing for lower bandwidth requirements and more efficient training on GPUs due to their FLoP-heavy nature[5]. Various architectures have been proposed: Tabak[16] recursed through layers, Geiping et al.[17] used two layers for encoding and four recursive layers before decoding, and Bae et al.[18] incorporated LoRAs alongside recurrence for better performance. Empirical studies demonstrated the efficacy of the recurrent structure. It achieves performance closing in on models twice their parameter count through repeated computation while enabling adaptive compute and early exiting[16,17]. Additionally, recurrent models can scale compute depth dynamically, allowing them to allocate computational resources on harder problems, akin to human reasoning[5,6,19]. While results are promising, these models lack standardization—being trained on varying hyperparameters and datasets—leading to incomparable benchmarks. For instance, the parameter counts of these models were drastically different, Bae et al.[18] trained models from one to two billion parameters, Geiping et al.[17] 3.5 billion, and[16] on the scale of millions. We trained all three architectures on the same dataset and control for as many hyperparameters as possible.

Methods

We evaluated 3 different recurrent architectures and one control model on a 3070 laptop GPU, while keeping most hyperparameters the same.

Hyperparameters	Value
Layers	4
Hidden Dimension	256
Sequence Length	128
Intermediate Dimension	512
Vocab	4096
Attention Heads	8
Key Value Heads	8
Batch Size	32
Recurrent depth	4

The implementation extended on top of the llama2.c github project with architectural changes for each variant.

Training Data and Evaluation

We will be training on Tiny Stories[20]. Tiny Stories is a dataset that contains elementary stories generated by GPT-4 and is designed to test the efficacy of reasoning in small language models[20]. Common LLM evaluations require a model with general knowledge of the world. With a model so small trained

© The National High School Journal of Science 2025

on stories, results would be unusable[21–23]. Instead gemini-2.5-flash-preview is used to grade on 4 criteria with a similar prompt to[16]:

In the following exercise, the student is given a task of writing a short story. The student needs to come up with a coherent story that is graded on criteria mentioned below. The exercise tests the student's language abilities, creativity, and ability to follow directions.

{story}

Please provide your general assessment about the story written by the student. Consider the following aspects:

1 Grammar: Is the completion grammatically correct?

2 Creativity: Does the completion show creativity and original thought?

3 Consistency: Is the completion consistent with the beginning of the story?

4 Plot: Does the plot of the completion make sense and is it coherent throughout?

Now, grade the student's completion in terms of the following categories, each on a scale from 1 to 10.

FraiLT Approach

Used in[16], this architecture uses a transformer layer, with an iteration encoder similar to how learnable positional encodings work. It will have 4 transformer layers and recur 4 times.

Three Part Approach

In[17], a three-part approach was used. A prelude, recurrent, and coda blocks. In this experiment, the prelude will consist of the embedding matrix as well as one transformer layer. The coda will contain the unembedding and one transformer layer as well. The recurrent block will have 2 layers and recur 7 times.

Given a number of recurrent iterations r, and a sequence of input tokens $x \in V^n$, these groups are used in the following way to produce output probabilities $p \in \mathbb{R}n \times |V|$:

$$e = P(x)$$
$$s_0 \sim \mathcal{N}(0, \sigma^2 I_{n \cdot h})$$
$$s_i = R(e, s_{i-1}) \quad \text{for } i \in \{1, \ldots, r\}$$
$$p = C(s_r)$$

where σ is some standard deviation for initializing the random state. This process is shown in Figure 2. Given an initial random state s_0, the model repeatedly applies the core block R, which accepts the latent state s_{i-1} and the embedded input e and outputs a new latent state s_i. After finishing all iterations, the coda block processes the last state and produces the probabilities of the next token.

Adding LoRAs

[18] employs LoRAs to mimic bigger large language models[24]. This model also recurs through 4 layers. A LoRA of rank 16 will be unique for each layer for each depth. The output of the transformer layer will be combined with the LoRAs for each layer.

Differences in Hyperparameters for each model

Table 1 The differences between each of the models hyperparameters

Hyperparameter	FraiLT	Three Part	Relaxed	Control
Recurrent Depth	4	7	4	0
Recurrent Blocks	4	2	4	0
Lora Rank	0	0	16	0

Table 2 Differences in parameter count in each of the models.

Component	FraiLT	Three Part	Relaxed	Control
adapt	0	131,072	0	0
depth_encoding	1,024	0	0	0
lora	0	0	1,246,976	0
Total Parameters	4,214,016	4,344,320	5,462,272	4,212,992

To try and minimize the recurrence difference between the models, a few modifications had to be made. With the Three Part architecture needing Prelude and Coda layers, of the four, only two could be utilized when recurring, as detailed in Table 1. To compensate for having two layers, recurrent depth was changed to seven, resulting in 16 layers of computation. Lora Rank and adapt are integral to their respective architectures, but added additional parameters, as shown by the parameter counts in Table 2. Compensating for the additional parameters through changes in other hyperparameters would change other hyperparameters. We let it be.

Results

Time to train

© The National High School Journal of Science 2025

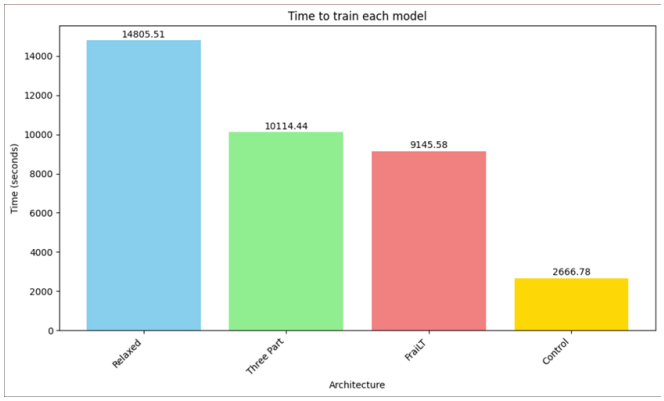

Fig. 1 Training duration across models

Memory Usage

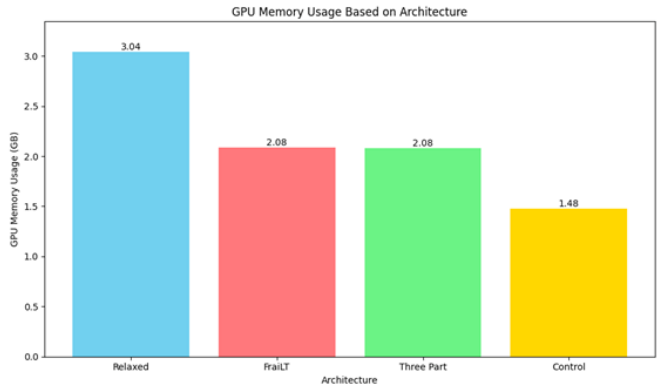

Fig. 2 VRAM usage for each model when training on a 3070 laptop GPU

Training duration was increased for all recurrent models, as evident in Figure 1. FraiLT and Three Part's duration were almost quadruple the control', while memory usage increased by one third, as shown in Figure 2. This demonstrates that recurrent models require four times the compute, but not four times the memory.

The Relaxed architecture exhibited a large increase in memory usage and training duration from the other recurrent depth models (see Figure 1 and Figure 2). This could potentially be attributed to the increase in parameters and implementation overhead(as a linear extrapolation from Three Part's memory requirements revealed ∼2.6 GB).

Validation Loss and Evaluation

The models were trained for 204 million tokens with a batch size of 32 at 8192 tokens per batch, in total 25,000 batches. The final validation loss for each model is presented in Table 3.

Table 3 Final validation loss for each of the models

	Validation Loss
Control	1.429
Relaxed	1.393
FraiLT	1.39
Three Part	1.384

Table 4 Gemini 2.5 Flash Preview evaluation of each model on the four metrics described before, grammar, creativity, consistency and plot of 1024 samples for each model.

	Grammer	Creativity	Consistency	Plot	Average
Control	3.3	3.41	2.65	2.21	2.89
Relaxed	3.53	3.48	2.75	2.32	3.02
FraiLT	3.47	3.51	2.84	2.4	3.05
Three Part	3.52	3.54	2.85	2.4	3.07

We used Gemini 2.5 Flash Preview to evaluate the model by inputting the generated story and the prompt. As shown in Table 4, the model gives feedback on four criteria: grammar, creativity, consistency, and plot coherence on a scale of one to ten. This method utilizes the Large Language Models writing capabilities to provide a more nuanced opinion compared to other approaches.

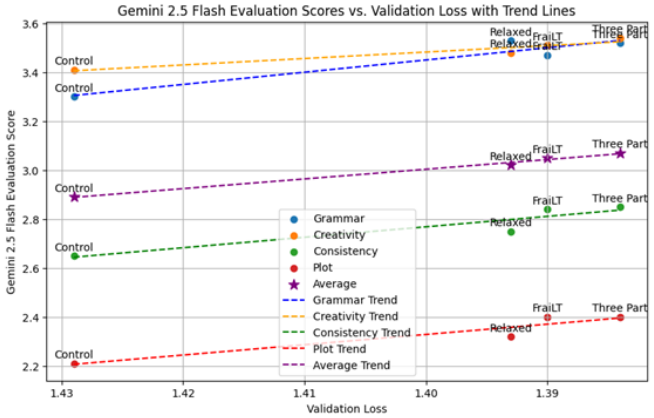

Fig. 3 The plot between Validation Loss and Evaluation Score.

From the data observed, the recurrent depth models outperform the control model, consistent with the findings in [16]. As shown in Table 5, given the minimal performance difference between all three of the recurrent depth models, the model initiation seems to have been the determining factor for which model would perform the best. However, there is an increase in the gap between the control model and the recurrent models

Table 5 The model validation loss at initiation compared to final validation loss for each architecture.

Model	Initial model loss	Validation Loss
Control	8.39	1.492
Relaxed	8.375	1.393
FraiLT	8.373	1.39
Three Part	8.369	1.384

after training. Interestingly, even though the Three Part model only recurs through two layers, it matches the performance of the models that recur through four layers, as further illustrated in Figure 3. It could suggest that the limit of recurrence is lower than expected or the Three Part architecture would be more efficient scaled up. It might also indicate that the models are undertrained (have not been trained for long enough) as there is seemingly no performance difference between the models. Another factor could be the scale of these models, we trained on the scale of millions, while the architectures were proposed on the scale of billions. We expected the Relaxed model to outperform the others due to it's higher parameter count, but it wasn't observed here. Another factor could be the small scale, as results in [16] found that recurrent depth performance scales with embedding size.

Limitations and Conclusions

Due to time and compute restraints, these models were trained at a relatively small scale, a significant limitation given their potential complexity, and a control model double the size was not trained. This was despite the embedding size being scaled up to 1024, where in [16], FraiLT models reached similar performance to models double their size. With this information in mind, the recurrent depth models do seem to outperform models with no recurrence with roughly the same parameter count. The difference in performance between the three recurrent depth architectures seems to be due to the difference in model initialization and the inherent limitations imposed by the small scale of the training. This restricted scale likely masked the true potential for divergent performance. This could potentially be attributed to undertraining, as all three architectures had similar validation losses and scores, suggesting they may not have fully converged to their optimal performance. While recurrent depth models are effective at these small scales, this may not be indicative of model performance at larger, more representative scales. The three recurrent architectures would likely show significant differences in performance then, as the larger scale would allow their architectural nuances to manifest more clearly. While no recurrent depth architecture achieved better performance than the others under these constrained conditions, it proved the effectiveness of recurrent depth architectures in general at a relatively small scale.

Future Work

While the effectiveness of recurrent depth models seem promising, the efficacy of the different architectures were not shown and we suggest further work to further understand the difference between the architectures

- Training at Larger Scales: Due to computational and time restraints, our models were relatively small, potential emergent properties of the architectures were not shown such as how architectural nuances might manifest in performance gains or limitations at higher parameter counts. Training these architectures at larger scales will provide insights into the scalability of the models, as well as impacts each change had on the performance of the models.
- Further Understanding Recurrence: The gains from recurring additional times may taper off due to the limitations of parameters. However no research has been done testing the limits and plotting the trade off between recurrence and model performance
- Combining Proposed Architectures: While no discernible difference was observed, we believe that at larger scales differences will be more substantial. At that scale some more architectures could be introduced, through a combination of the proposed architectures or novel approaches.

References

1 A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser and I. Polosukhin, *Attention Is All You Need*, 2023, https://doi.org/10.48550/arXiv.1706.03762.

2 J. Devlin, M.-W. Chang, K. Lee and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*, 2019, https://doi.org/10.48550/arXiv.1810.04805.

3 T. Nouamane, F. Ferdinand, Z. Haojun, N. Phuc, M. Mohamed, W. Leandro and T. Thomas, *The Ultra-Scale Playbook*, 2025, https://huggingface.co/spaces/nanotron/ultrascale-playbook.

4 T. Nouamane, F. Ferdinand, Z. Haojun, N. Phuc, M. Mohamed, W. Leandro and T. Thomas, *The Ultra-Scale Playbook*, 2025, https://huggingface.co/spaces/nanotron/ultrascale-playbook.

5 J. Kaplan, S. McCandlish, T. Henighan, T. B. Brown, B. Chess, R. Child, S. Gray, A. Radford, J. Wu and D. Amodei, *Scaling Laws for Neural Language Models*, 2020, https://doi.org/10.48550/arXiv.2001.08361.

6 P. Hijma, S. Heldens, A. Sclocco, B. van Werkhoven and H. E. Bal, *Optimization Techniques for GPU Programming*, 2023.

7 T. Dao, D. Y. Fu, S. Ermon, A. Rudra and C. Ré, *FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness*, 2022, https://doi.org/10.48550/arXiv.2205.14135.

© The National High School Journal of Science 2025

8 T. Dao, *FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning*, 2023, https://doi.org/10.48550/arXiv.2307.08691.

9 P. Micikevicius, S. Narang, J. Alben, G. Diamos, E. Elsen, D. Garcia, B. Ginsburg, M. Houston, O. Kuchaiev, G. Venkatesh and H. Wu, *Mixed Precision Training*, 2018, https://doi.org/10.48550/arXiv.1710.03740.

10 J. Yuan, H. Gao, D. Dai, J. Luo, L. Zhao, Z. Zhang, Z. Xie, Y. X. Wei, L. Wang, Z. Xiao, Y. Wang, C. Ruan, M. Zhang, W. Liang and W. Zeng, *Native Sparse Attention: Hardware-Aligned and Natively Trainable Sparse Attention*, 2025, https://doi.org/10.48550/arXiv.2502.11089.

11 DeepSeek-AI et al., *DeepSeek-MoE: The All-Around MoE*, 2025, https://doi.org/10.48550/arXiv.2501.07725.

12 R. Child, S. Gray, A. Radford and I. Sutskever, *Generating Long Sequences with Sparse Transformers*, 2019, https://doi.org/10.48550/arXiv.1904.10509.

13 N. Kitaev, Kaiser and A. Levskaya, *Reformer: The Efficient Transformer*, 2020, https://doi.org/10.48550/arXiv.2001.04451.

14 W. Cai, J. Jiang, F. Wang, J. Tang, S. Kim and J. Huang, *A Survey on Mixture of Experts in Large Language Models*, 2025.

15 R. Pope, S. Douglas, A. Chowdhery, J. Devlin, J. Bradbury, A. Levskaya, J. Heek, K. Xiao, S. Agrawal and J. Dean, *Efficiently Scaling Transformer Inference*, 2022, https://doi.org/10.48550/arXiv.2211.05102.

16 A. Tabak, *Freely Long-Thinking Transformer (FraiLT)*, 2024, https://doi.org/10.48550/arXiv.2401.11626.

17 J. Geiping, S. McLeish, N. Jain, J. Kirchenbauer, S. Singh, B. R. Bartoldson, B. Kailkhura, A. Bhatele and T. Goldstein, *Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach*, 2025, https://doi.org/10.48550/arXiv.2502.05171.

18 S. Bae, A. Fisch, H. Harutyunyan, Z. Ji, S. Kim and T. Schuster, *Relaxed Recursive Transformers: Effective Parameter Sharing with Layer-wise LoRA*, 2025, https://doi.org/10.48550/arXiv.2410.20672.

19 J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le and D. Zhou, *Chain-of-Thought Prompting Elicits Reasoning in Large Language Models*, 2023, https://doi.org/10.48550/arXiv.2201.11903.

20 R. Eldan and Y. Li, *TinyStories: How Small Can Language Models Be and Still Speak Coherent English?*, 2023, https://doi.org/10.48550/arXiv.2305.07759.

21 K. Cobbe, V. Kosaraju, M. Bavarian, M. Chen, H. Jun, L. Kaiser, M. Plappert, J. Tworek, J. Hilton, R. Nakano, C. Hesse and J. Schulman, *Training Verifiers to Solve Math Word Problems*, 2021, https://doi.org/10.48550/arXiv.2110.14168.

22 D. Hendrycks, C. Burns, S. Basart, A. Zou, M. Mazeika, D. Song and J. Steinhardt, *Measuring Massive Multitask Language Understanding*, 2021, https://doi.org/10.48550/arXiv.2009.03300.

23 S. Lin, J. Hilton and O. Evans, *TruthfulQA: Measuring How Models Mimic Human Falsehoods*, 2022, https://doi.org/10.48550/arXiv.2109.07958.

24 E. J. Hu, Y. Shen, P. Wallis, Z. Allen-Zhu, Y. Li, S. Wang, L. Wang and W. Chen, *LoRA: Low-Rank Adaptation of Large Language Models*, 2021, https://doi.org/10.48550/arXiv.2106.09685.

© The National High School Journal of Science 2025