Scaling up Test-Time Compute with Latent Reasoning: A Recurrent Depth Approach

Jonas Geiping [1] **Sean McLeish** [2] **Neel Jain** [2] **John Kirchenbauer** [2] **Siddharth Singh** [2] **Brian R. Bartoldson** [3]
Bhavya Kailkhura [3] **Abhinav Bhatele** [2] **Tom Goldstein** [2]

Abstract

We study a novel language model architecture that is capable of scaling test-time computation by implicitly reasoning in latent space. Our model works by iterating a recurrent block, thereby unrolling to arbitrary depth at test-time. This stands in contrast to mainstream reasoning models that scale up compute by producing more tokens. Unlike approaches based on chain-of-thought, our approach does not require any specialized training data, can work with small context windows, and can capture types of reasoning that are not easily represented in words. We train a proof-of-concept model from scratch with 3.5 billion parameters and 800 billion tokens. We show that this model can effortlessly use varying levels of compute, significantly improving with additional compute especially on reasoning tasks, such as math and coding.

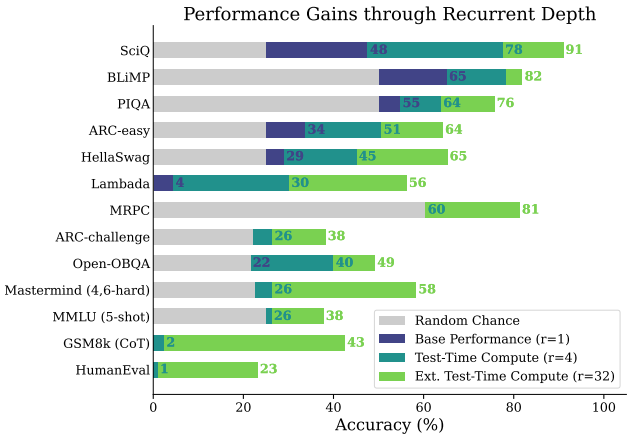

Figure 1: We train a 3.5B parameter language model with depth recurrence. Simple tasks that require less reasoning like SciQ and BLiMP are solved almost immediately, while tasks like Mastermind (the reasoning game), MMLU, GSM8k or HumanEval are solved only with extended compute via test time recurrence, a separation in difficulty levels that emerges from pretraining the model at scale.

1. Scaling by Thinking in Continuous Space

Humans naturally expend more mental effort solving some problems than others. While humans are capable of thinking over long time spans by verbalizing intermediate results and writing them down, a substantial amount of thought happens through complex, recurrent firing patterns in the brain, before the first word of an answer is uttered.

Early attempts at increasing the power of language models focused on scaling model size, a practice that requires extreme amounts of data and computation. More recently, researchers have explored ways to enhance the reasoning capability of models by scaling test time computation. The mainstream approach involves post-training on long chain-of-thought examples to develop the model's ability to *verbalize* intermediate calculations in its context window and thereby externalize thoughts.

However, the constraint that expensive internal reasoning must always be projected down to a single verbalized next token appears wasteful; it is plausible that models could be more competent if they were able to natively "think" in their continuous latent space. One way to unlock this untapped dimension of additional compute involves adding a recurrent unit to a model. This unit runs in a loop, iteratively processing and updating its hidden state and enabling computations to be carried on indefinitely. While this is not currently the dominant paradigm, this idea is foundational to machine learning and has been (re-)discovered in every decade, for example as recurrent neural networks (Gers & Schmidhuber, 2000), diffusion models (Song & Ermon, 2019), feature recycling (Jumper et al., 2021), and as universal (Dehghani et al., 2019) or looped transformers (Giannou et al., 2023).

In this work, we show that depth-recurrent language models can learn effectively, be trained in an efficient manner, and demonstrate significant performance improvements under

[*]Equal contribution [1]ELLIS Institute Tübingen, Max-Planck Institute for Intelligent Systems, Tübingen AI Center [2]University of Maryland, College Park [3]Lawrence Livermore National Laboratory. Correspondence to: Jonas Geiping <jonas@tue-ellis.eu>.

Presented at ES-FoMo III: 3rd Workshop on Efficient Systems for Foundation Models, Vancouver, Canada, 2025.

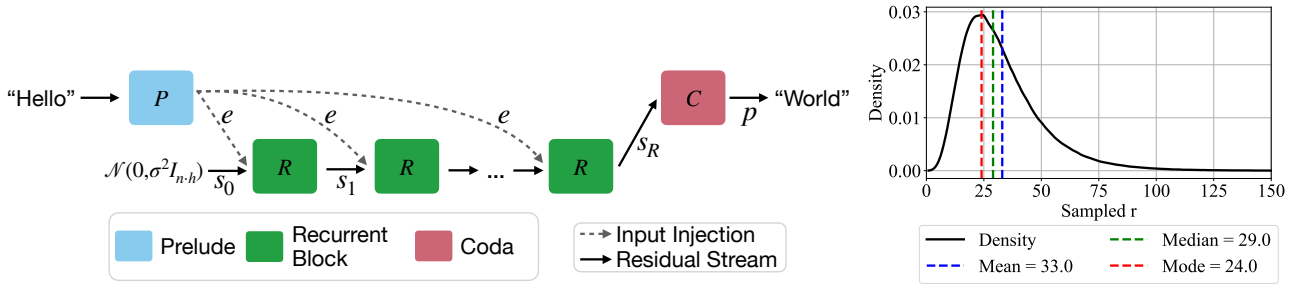

Figure 2: Left: A visualization of the Architecture, as described in Section 2. Each block consists of a number of sub-layers. The blue prelude block embeds the inputs into latent space, where the green shared *recurrent block* is a block of layers that is repeated to compute the final latent state, which is decoded by the layers of the red coda block. **Right:** We use a log-normal Poisson Distribution to sample the number of recurrent iterations for each training step.

the scaling of test-time compute. Our proposed transformer architecture is built upon a latent depth-recurrent block that is run for a randomly sampled number of iterations during training. We show that this paradigm can scale to several billion parameters and over half a trillion tokens of pretraining data. At test-time, the model can improve its performance through recurrent reasoning in latent space, enabling it to compete with other open-source models that benefit from more parameters and training data.

2. A Scalable Recurrent Architecture

In this section we will describe our proposed architecture for a transformer with latent recurrent depth, discussing design choices and small-scale ablations. A diagram of the architecture can be found in Figure 2. We always refer to the sequence dimension as n, the hidden dimension of the model as h, and its vocabulary as the set V.

2.1. Macroscopic Design

The model is primarily structured around decoder-only transformer blocks (Vaswani et al., 2017; Radford et al., 2019). However these blocks are structured into three functional groups, the *prelude* P, which embeds the input data into a latent space using multiple transformer layers, then the core *recurrent block* R, which is the central unit of recurrent computation modifying states $\mathbf{s} \in \mathbb{R}^{n \times h}$, and finally the *coda* C, which un-embeds from latent space and also contains the prediction head of the model. The core block is set between the prelude and coda blocks, and by looping the core we can run an indefinite amount of compute.

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

$$\begin{aligned} \mathbf{e} &= P(\mathbf{x}) \\ \mathbf{s}_0 &\sim \mathcal{N}(\mathbf{0}, \sigma^2 I_{n \cdot h}) \\ \mathbf{s}_i &= R(\mathbf{e}, \mathbf{s}_{i-1}) \qquad \text{for} \quad i \in \{1, \dots, r\} \\ \mathbf{p} &= C(\mathbf{s}_r), \end{aligned}$$

where σ is some standard deviation for initializing the random state. This process is shown in Figure 2. Given an init random state $\mathbf{s}_0$, the model repeatedly applies the core block R, which accepts the latent state $\mathbf{s}_{i-1}$ and the embedded input $\mathbf{e}$ and outputs a new latent state $\mathbf{s}_i$. After finishing all iterations, the coda block processes the last state and produces the probabilities of the next token. This architecture is based on deep thinking literature, where it is shown that injecting the inputs $\mathbf{e}$ in every step (Bansal et al., 2022) and initializing the latent vector with a random state stabilizes the recurrence and promotes convergence to a steady state independent of initialization, i.e. *path independence* (Anil et al., 2022).

2.2. Training Objective

Training Recurrent Models through Unrolling. To ensure that the model can function when we scale up recurrent iterations at test-time, we randomly sample iteration counts during training, assigning a random number of iterations r to every input sequence (Schwarzschild et al., 2021b). We optimize the expectation of the loss function L over random samples x from distribution X and random iteration counts r from distribution Λ.

$$\mathcal{L}(\theta) = \mathbb{E}_{\mathbf{x} \in X} \mathbb{E}_{r \sim \Lambda} L\left(m_\theta(\mathbf{x}, r), \mathbf{x}'\right).$$

Here, m represents the model output, and $\mathbf{x}'$ is the sequence $\mathbf{x}$ shifted left, i.e., the next tokens in the sequence $\mathbf{x}$. We choose Λ to be a *log-normal Poisson distribution*. Given a targeted mean recurrence $\bar{r} + 1$ and a variance that we set to $\sigma = \frac{1}{2}$, we can sample from this distribution via

$$\tau \sim \mathcal{N}(\log(\bar{r}) - \frac{1}{2}\sigma^2, \sigma) \tag{1}$$

$$r \sim \mathcal{P}(e^\tau) + 1, \tag{2}$$

given the normal distribution $\mathcal{N}$ and Poisson distribution $\mathcal{P}$, see Figure 2. The distribution most often samples values less than $\bar{r}$, but it contains a heavy tail of occasional events in which significantly more iterations are taken.

Table 1: Results on zero-shot evaluations across various open-source models. We show ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021b), OpenBookQA (Mihaylov et al., 2018), PiQA (Bisk et al., 2020), SciQ (Johannes Welbl, 2017), and WinoGrande (Sakaguchi et al., 2021). We report normalized accuracy when provided.

Model	Param	Tokens	ARC-E	ARC-C	HellaSwag	MMLU	OBQA	PiQA	SciQ	WinoGrande
random			25.0	25.0	25.0	25.0	25.0	50.0	25.0	50.0
Amber	7B	1.2T	65.70	37.20	72.54	26.77	41.00	78.73	88.50	63.22
Pythia-2.8b	2.8B	0.3T	58.00	32.51	59.17	25.05	35.40	73.29	83.60	57.85
Pythia-6.9b	6.9B	0.3T	60.48	34.64	63.32	25.74	37.20	75.79	82.90	61.40
Pythia-12b	12B	0.3T	63.22	34.64	66.72	24.01	35.40	75.84	84.40	63.06
OLMo-1B	1B	3T	57.28	30.72	63.00	24.33	36.40	75.24	78.70	59.19
OLMo-7B	7B	2.5T	68.81	40.27	75.52	28.39	42.20	80.03	88.50	67.09
OLMo-7B-0424	7B	2.05T	75.13	45.05	77.24	47.46	41.60	80.09	96.00	68.19
OLMo-7B-0724	7B	2.75T	74.28	43.43	77.76	50.18	41.60	80.69	95.70	67.17
OLMo-2-1124	7B	4T	82.79	57.42	80.50	60.56	46.20	81.18	96.40	74.74
Ours, ($r = 4$)	3.5B	0.8T	49.07	27.99	43.46	23.39	28.20	64.96	80.00	55.24
Ours, ($r = 8$)	3.5B	0.8T	65.11	35.15	58.54	25.29	35.40	73.45	92.10	55.64
Ours, ($r = 16$)	3.5B	0.8T	69.49	37.71	64.67	31.25	37.60	75.79	93.90	57.77
Ours, ($r = 32$)	3.5B	0.8T	69.91	38.23	65.21	31.38	38.80	76.22	93.50	59.43

Table 2: Benchmarks of math. reasoning and understanding. We report flexible and strict match for GSM8K and GSM8K CoT, extracted match for Minerva Math, and acc norm. for MathQA.

Model	GSM8K	GSM8k CoT	Minerva MATH	MathQA
Random	0.00	0.00	0.00	20.00
Amber	3.94/4.32	3.34/5.16	1.94	25.26
Pythia-2.8b	1.59/2.12	1.90/2.81	1.96	24.52
Pythia-6.9b	2.05/2.43	2.81/2.88	1.38	25.96
Pythia-12b	3.49/4.62	3.34/4.62	2.56	25.80
OLMo-1B	1.82/2.27	1.59/2.58	1.60	23.38
OLMo-7B	4.02/4.09	6.07/7.28	2.12	25.26
OLMo-7B-0424	27.07/27.29	26.23/26.23	5.56	28.48
OLMo-7B-0724	28.66/28.73	28.89/28.89	5.62	27.84
OLMo-2-1124-7B	66.72/66.79	61.94/66.19	19.08	37.59
Ours ($r = 32$)	28.05/28.20	32.60/34.57	12.58	26.60
Our w/ chat templ. ($r = 32$)	24.87/38.13	34.87/42.84	11.24	27.97

Table 3: Evaluation on code benchmarks, MBPP and HumanEval. We report pass@1 for both datasets.

Model	Param	Tokens	MBPP	HumanEval
Random			0.00	0.00
starcoder2-3b	3B	3.3T	43.00	31.09
starcoder2-7b	7B	3.7T	43.80	31.70
Amber	7B	1.2T	19.60	13.41
Pythia-2.8b	2.8B	0.3T	6.70	7.92
Pythia-6.9b	6.9B	0.3T	7.92	5.60
Pythia-12b	12B	0.3T	5.60	9.14
OLMo-1B	1B	3T	0.00	4.87
OLMo-7B	7B	2.5T	15.6	12.80
OLMo-7B-0424	7B	2.05T	21.20	16.46
OLMo-7B-0724	7B	2.75T	25.60	20.12
OLMo-2-1124-7B	7B	4T	21.80	10.36
Ours ($r = 32$)	3.5B	0.8T	24.80	23.17

Truncated Backpropagation. To keep computation and memory low at train time, we backpropagate through only the last k iterations of the recurrent unit. This enables us to train with the heavy-tailed Poisson distribution Λ, as maximum activation memory and backward compute is now independent of r. We fix $k = 8$ in our main experiments. At small scale, this works as well as sampling k uniformly, but it equalizes the overall memory usage in each step of training. Note that the prelude block still receives gradient updates in every step, as its output $\mathbf{e}$ is injected in every step. This setup resembles truncated backpropagation through time, as commonly done with RNNs, although our setup is recurrent in depth rather than time (Williams & Peng, 1990; Mikolov et al., 2011). Truncated backpropagation can also be understood as approximation of objectives based on differentiating a fixed point of the recurrence, as discussed in Geng et al. (2021).

2.3. How to Train a Large-Scale Recurrent-Depth Model In the Wild

After verifying that we can reliably train small test models up to 10B tokens, we move on to larger-scale runs. Given our limited compute budget, we could either train multiple tiny models too small to show emergent effects or scaling, or train a single medium-scale model. Based on this, we prepared a single large-scale run. We train a 3.5B parameter variant of the proposed architecture on a mix of generic text, code and scientific data, with data-parallel training with a batch size of 16 million tokens. We provide comprehensive information on all training details in Appendix H.

3. Benchmark Results

We ultimately train the final model for 800B tokens, and a non-recurrent baseline comparison for 180B tokens. We evaluate these checkpoints against other open-source models trained on fully public datasets (like ours) of a similar size. We compare against Amber (Liu et al., 2023c), Pythia (Biderman et al., 2023) and a number of OLMo 1&2 variants (Groeneveld et al., 2024; AI2, 2024; Team OLMo et al., 2025). We execute all standard benchmarks through the `lm-eval harness` (Biderman et al., 2024) and code benchmarks via `bigcode-bench` (Zhuo et al., 2024).

3.1. Standard Benchmarks

We collect results for established benchmark tasks (Team OLMo et al., 2025) in Table 1 and show all open-source models side-by-side. In direct comparison we see that our model outperforms the older Pythia series and is roughly comparable to the first OLMo generation, OLMo-7B in

Table 4: Baseline comparison, recurrent versus non-recurrent model trained in the same training setup and data. Comparing the recurrent model with its non-recurrent baseline, we see that even at 180B tokens, the recurrent substantially outperforms on harder tasks.

Model	Tokens	ARC-E	ARC-C	HellaSwag	MMLU	OBQA	PiQA	SciQ	WinoGrande	GSM8K CoT
Fixed-Depth Baseline	0.18T	46.42	26.96	37.34	24.16	29.60	64.47	73.20	51.78	1.82/2.20
Ours, early ckpt, ($r = 32$)	0.18T	53.62	29.18	48.80	25.59	31.40	68.88	80.60	52.88	9.02/10.24
Ours, early ckpt, ($r = 1$)	0.18T	34.01	23.72	29.19	23.47	25.60	53.26	54.10	53.75	0.00/0.15
Ours, ($r = 32$)	0.8T	69.91	38.23	65.21	31.38	38.80	76.22	93.50	59.43	34.80/42.08
Ours, ($r = 1$)	0.8T	34.89	24.06	29.34	23.60	26.80	55.33	47.10	49.41	0.00/0.00

most metrics, but lags behind the later OLMo models trained larger, more carefully curated datasets. For the first recurrent-depth model for language to be trained at this scale, and considering the limitations of the training run, we find these results promising and certainly suggestive that further research into latent recurrence as an approach to test-time scaling is warranted.

3.2. Math and Coding Benchmarks

We also evaluate the model on math and coding. For math, we evaluate GSM8k (Cobbe et al., 2021) (as 5-shot and in the 8-way CoT setup), MATH ((Hendrycks et al., 2021a) with the Minerva evaluation rules (Lewkowycz et al., 2022)) and MathQA (Amini et al., 2019). For coding, we check MBPP (Austin et al., 2021) and HumanEval (Chen et al., 2021). Here we find that our model significantly surpasses all models except the latest OLMo-2 model in mathematical reasoning, as measured on GSM8k and MATH. On coding benchmarks the model beats all other general-purpose open-source models, although it does not outperform dedicated code models, such as StarCoder2 (Lozhkov et al., 2024), trained for several trillion tokens. We also note that while further improvements in language modeling are slowing down, as expected at this training scale, both code and mathematical reasoning continue to improve steadily throughout training, see Figure 9.

3.3. Where does recurrence help most?

How much of this performance can we attribute to recurrence, and how much to other factors, such as dataset, tokenization and architectural choices? In Table 4, we compare our recurrent model against its non-recurrent twin, which we trained to 180B tokens in the exact same setting. In direct comparison of both models at 180B tokens, we see that the recurrent model outperforms its baseline with an especially pronounced advantage on harder tasks, such as the ARC challenge set. On other tasks, such as SciQ, which requires straightforward recall of scientific facts, performance of the models is more similar. We observe that gains through reasoning are especially prominent on GSM8k, where the 180B recurrent model is already 5 times better than the baseline at this early snapshot in the pretraining process. We also note that the recurrent model, when evaluated with only a single recurrence, effectively stops improving between the early 180B checkpoint and the 800B checkpoint on hard tasks, showing that further improvements are not built into the fixed, non-recurrent parts but encoded entirely into the iterations of the recurrent block.

3.4. Improvements through Weight Averaging

Due to our constant learning rate, we can materialize further improvements through weight averaging (Izmailov et al., 2018) to simulate the result of a cooldown (Hägele et al., 2024; DeepSeek-AI et al., 2024). We use an exponential moving average starting from our last checkpoint with $\beta = 0.9$, incorporating the last 75 checkpoints with a dilation factor of 7, a modification to established protocols (Kaddour, 2022; Sanyal et al., 2024). We evaluate this EMA model as well, which further improves GMS8k performance to 47.23% flexible (38.59% strict), when tested at $r = 64$.

4. Limitations and Conclusions

While the experiments in this paper demonstrate the viability of (latent) recurrent-depth architectures for language modeling at scale, the models described are ultimately still a proof-of-concept. We observe that we can train models that improve with increased test-time scaling via recurrence, improving over a fixed-depth model with the same parameter count by 5x on GSM8K. Nevertheless, future work with additional compute is still required to allow for precise comparisons to the other forms of scaling, such as training fixed-depth transformers with the same FLOP count in pretraining, or training verbal CoT models targeting the same FLOP count at test time.

Yet, the interesting behaviors already observable in this work, such as the context-dependent problem-solving speed, with the model learning to solve easy problems with fewer recurrences than harder problems, various zero-shot abilities and emergence of structured thinking in latent space, lead us to believe that latent reasoning is a promising research direction to complement existing approaches for test-time compute scaling. Our work validates the motivations and observations of prior work developed at smaller scales for universal, looped and deep thinking transformers, and we are excited about the potential impact of imbuing generative models with the ability to reason in continuous latent space without the need for specialized data at train time or verbalization at inference time.

Acknowledgements

An award for computer time was provided by the U.S. Department of Energy's (DOE) Innovative and Novel Computational Impact on Theory and Experiment (INCITE) Program. This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. Work on the LLNL side was prepared by LLNL under Contract DE-AC52-07NA27344 and supported by the LLNL-LDRD Program under Project No. 24-ERD-010 and 24-ERD-058 (LLNL-CONF-872390). This manuscript has been authored by Lawrence Livermore National Security, LLC under Contract No. DE-AC52-07NA27344 with the U.S. Department of Energy. The United States Government retains a non-exclusive, paid-up, irrevocable, world-wide license to publish or reproduce the published form of this manuscript, or allow others to do so, for United States Government purposes.

JG further acknowledges the support of the Hector II foundation. A large number of small-scale and preliminary experiments were made possible through the support of the MPI Intelligent Systems compute cluster and funding by the Tübingen AI center.

UMD researchers were further supported by the ONR MURI program, DARPA TIAMAT, the National Science Foundation (IIS-2212182), and the NSF TRAILS Institute (2229885). Commercial support was provided by Capital One Bank, the Amazon Research Award program, and Open Philanthropy. Finally, we thank Avi Schwarzschild for helpful comments on the initial draft.

References

Abnar, S., Saremi, O., Dinh, L., Wilson, S., Bautista, M. A., Huang, C., Thilak, V., Littwin, E., Gu, J., Susskind, J., and Bengio, S. Adaptivity and Modularity for Efficient Generalization Over Task Complexity. *arxiv:2310.08866[cs]*, October 2023. doi: 10.48550/arXiv.2310.08866. URL http://arxiv.org/abs/2310.08866.

AI2. OLMo 1.7–7B: A 24 point improvement on MMLU, April 2024. URL https://blog.allenai.org/olmo-1-7-7b-a-24-point-improvement-on-mmlu-92b43f7d269d.

Allen-Zhu, Z. and Li, Y. Physics of language models: Part 3.1, knowledge storage and extraction. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235 of *ICML'24*, pp. 1067–1077, Vienna, Austria, July 2024. JMLR.org.

Amari, S.-I. Learning Patterns and Pattern Sequences by Self-Organizing Nets of Threshold Elements. *IEEE Transactions on Computers*, C-21(11):1197–1206, November 1972. ISSN 1557-9956. doi: 10.1109/T-C.1972.223477. URL https://ieeexplore.ieee.org/document/1672070.

AMD. AMD Instinct™ MI250X Accelerators, November 2021. URL https://www.amd.com/en/products/accelerators/instinct/mi200/mi250x.html.

Amini, A., Gabriel, S., Lin, P., Koncel-Kedziorski, R., Choi, Y., and Hajishirzi, H. Mathqa: Towards interpretable math word problem solving with operation-based formalisms. *arXiv preprint arXiv:1905.13319*, 2019.

Amos, B. and Kolter, J. Z. OptNet: Differentiable Optimization as a Layer in Neural Networks. In *International Conference on Machine Learning*, pp. 136–145, July 2017. URL http://proceedings.mlr.press/v70/amos17a.html.

Anil, C., Pokle, A., Liang, K., Treutlein, J., Wu, Y., Bai, S., Kolter, J. Z., and Grosse, R. B. Path Independent Equilibrium Models Can Better Exploit Test-Time Computation. In *Advances in Neural Information Processing Systems*, October 2022. URL https://openreview.net/forum?id=kgT6D7Z4Xv9.

Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., Dohan, D., Jiang, E., Cai, C., Terry, M., Le, Q., et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021.

Azerbayev, Z., Schoelkopf, H., Paster, K., Santos, M. D., McAleer, S. M., Jiang, A. Q., Deng, J., Biderman, S., and Welleck, S. Llemma: An Open Language Model for Mathematics. In *The Twelfth International Conference on Learning Representations*, October 2023. URL https://openreview.net/forum?id=4WnqRR915j.

Bai, S., Kolter, J. Z., and Koltun, V. Deep Equilibrium Models. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/01386bd6d8e091c2ab4c7c7de644d37b-Abstract.html.

Bai, S., Koltun, V., and Kolter, J. Z. Neural Deep Equilibrium Solvers. In *International Conference on Learning Representations*, March 2022. URL https://openreview.net/forum?id=B0oHOwT5ENL.

Bai, Y., Zhang, J., Lv, X., Zheng, L., Zhu, S., Hou, L., Dong, Y., Tang, J., and Li, J. LongWriter: Unleashing 10,000+ Word Generation from Long Context LLMs. *arxiv:2408.07055[cs]*, August 2024. doi: 10.48550/arXiv.2408.07055. URL http://arxiv.org/abs/2408.07055.

Banino, A., Balaguer, J., and Blundell, C. PonderNet: Learning to Ponder. In *8th ICML Workshop on Automated Machine Learning (AutoML)*, July 2021. URL https://openreview.net/forum?id=1EuxRTe0WN.

Bansal, A., Schwarzschild, A., Borgnia, E., Emam, Z., Huang, F., Goldblum, M., and Goldstein, T. End-to-end Algorithm Synthesis with Recurrent Networks: Extrapolation without Overthinking. In *Advances in Neural Information Processing Systems*, October 2022. URL https://openreview.net/forum?id=PPjSKy40XUB.

Bear, J., Prügel-Bennett, A., and Hare, J. Rethinking Deep Thinking: Stable Learning of Algorithms using Lipschitz Constraints. *arxiv:2410.23451[cs]*, October 2024. doi: 10.48550/arXiv.2410.23451. URL http://arxiv.org/abs/2410.23451.

Bekman, S. *Machine Learning Engineering Open Book*. Stasosphere Online Inc., 2023. URL https://github.com/stas00/ml-engineering.

Ben Allal, L., Lozhkov, A., Penedo, G., Wolf, T., and von Werra, L. SmolLM-corpus, July 2024. URL https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus.

Biderman, S., Schoelkopf, H., Anthony, Q., Bradley, H., O'Brien, K., Hallahan, E., Khan, M. A., Purohit, S., Prashanth, U. S., Raff, E., Skowron, A., Sutawika, L., and van der Wal, O. Pythia: A Suite for Analyzing Large Language Models Across Training and Scaling. *arxiv:2304.01373[cs]*, April 2023. doi: 10.48550/arXiv.2304.01373. URL http://arxiv.org/abs/2304.01373.

Biderman, S., Schoelkopf, H., Sutawika, L., Gao, L., Tow, J., Abbasi, B., Aji, A. F., Ammanamanchi, P. S., Black, S., Clive, J., DiPofi, A., Etxaniz, J., Fattori, B., Forde, J. Z., Foster, C., Hsu, J., Jaiswal, M., Lee, W. Y., Li, H., Lovering, C., Muennighoff, N., Pavlick, E., Phang, J., Skowron, A., Tan, S., Tang, X., Wang, K. A., Winata, G. I., Yvon, F., and Zou, A. Lessons from the Trenches on Reproducible Evaluation of Language Models. *arxiv:2405.14782[cs]*, May 2024. doi: 10.48550/arXiv.2405.14782. URL http://arxiv.org/abs/2405.14782.

Bisk, Y., Zellers, R., Bras, R. L., Gao, J., and Choi, Y. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*, 2020.

Boudiaf, M., Mueller, R., Ben Ayed, I., and Bertinetto, L. Parameter-Free Online Test-Time Adaptation. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8344–8353, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Boudiaf_Parameter-Free_Online_Test-Time_Adaptation_CVPR_2022_paper.html.

Braitenberg, V. *Vehicles: Experiments in Synthetic Psychology*. MIT press, 1986.

Brandon, W., Mishra, M., Nrusimha, A., Panda, R., and Kelly, J. R. Reducing Transformer Key-Value Cache Size with Cross-Layer Attention. *arxiv:2405.12981[cs]*, May 2024. doi: 10.48550/arXiv.2405.12981. URL http://arxiv.org/abs/2405.12981.

British Library Labs. *Digitised Books. c. 1510 - c. 1900. JSONL (OCR Derived Text + Metadata)*. British Library, 2021. URL https://doi.org/10.23636/r7w6-zy15.

Cai, T., Li, Y., Geng, Z., Peng, H., Lee, J. D., Chen, D., and Dao, T. Medusa: Simple LLM Inference Acceleration Framework with Multiple Decoding Heads. In *Forty-First International Conference on Machine Learning*, June 2024. URL https://openreview.net/forum?id=PEpbUobfJv.

character.ai. Optimizing AI Inference at Character.AI, June 2024. URL https://research.character.ai/optimizing-inference/.

Chen, M., Tworek, J., Jun, H., Yuan, Q., de Oliveira Pinto, H. P., Kaplan, J., Edwards, H., Burda, Y., Joseph, N., Brockman, G., Ray, A., Puri, R., Krueger, G., Petrov, M., Khlaaf, H., Sastry, G., Mishkin, P., Chan, B., Gray, S., Ryder, N., Pavlov, M., Power, A., Kaiser, L., Bavarian, M., Winter, C., Tillet, P., Such, F. P., Cummings, D., Plappert, M., Chantzis, F., Barnes, E., Herbert-Voss, A., Guss, W. H., Nichol, A., Paino, A., Tezak, N., Tang, J., Babuschkin, I., Balaji, S., Jain, S., Saunders, W., Hesse, C., Carr, A. N., Leike, J., Achiam, J., Misra, V., Morikawa, E., Radford, A., Knight, M., Brundage, M., Murati, M., Mayer, K., Welinder, P., McGrew, B., Amodei, D., McCandlish, S., Sutskever, I., and Zaremba, W. Evaluating large language models trained on code, 2021.

Cheng, J. and Durme, B. V. Compressed Chain of Thought: Efficient Reasoning Through Dense Representations. *arxiv:2412.13171[cs]*, December 2024. doi: 10.48550/arXiv.2412.13171. URL http://arxiv.org/abs/2412.13171.

Choi, E. GoodWiki dataset, September 2023. URL https://www.github.com/euirim/goodwiki.

Chollet, F. On the Measure of Intelligence. *arxiv:1911.01547[cs]*, November 2019. doi: 10.48550/arXiv.1911.01547. URL http://arxiv.org/abs/1911.01547.

Chowdhery, A., Narang, S., Devlin, J., Bosma, M., Mishra, G., Roberts, A., Barham, P., Chung, H. W., Sutton, C., Gehrmann, S., Schuh, P., Shi, K., Tsvyashchenko, S., Maynez, J., Rao, A., Barnes, P., Tay, Y., Shazeer, N., Prabhakaran, V., Reif, E., Du, N., Hutchinson, B., Pope, R., Bradbury, J., Austin, J., Isard, M., Gur-Ari, G., Yin, P., Duke, T., Levskaya, A., Ghemawat, S., Dev, S., Michalewski, H., Garcia, X., Misra, V., Robinson, K., Fedus, L., Zhou, D., Ippolito, D., Luan, D., Lim, H., Zoph, B., Spiridonov, A., Sepassi, R., Dohan, D., Agrawal, S., Omernick, M., Dai, A. M., Pillai, T. S., Pellat, M., Lewkowycz, A., Moreira, E., Child, R., Polozov, O., Lee, K., Zhou, Z., Wang, X., Saeta, B., Diaz, M., Firat, O., Catasta, M., Wei, J., Meier-Hellstern, K., Eck, D., Dean, J., Petrov, S., and Fiedel, N. PaLM: Scaling Language Modeling with Pathways. *arXiv:2204.02311 [cs]*, April 2022. URL http://arxiv.org/abs/2204.02311.

Clark, P., Cowhey, I., Etzioni, O., Khot, T., Sabharwal, A., Schoenick, C., and Tafjord, O. Think you have solved question answering? try arc, the ai2 reasoning challenge. *arXiv:1803.05457v1*, 2018.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training Verifiers to Solve Math Word Problems. *arxiv:2110.14168[cs]*, November 2021. doi: 10.48550/arXiv.2110.14168. URL http://arxiv.org/abs/2110.14168.

Colegrove, O., Paruchuri, V., and OpenPhi-Team. Openphi/textbooks · Datasets at Hugging Face, October 2024. URL https://huggingface.co/datasets/open-phi/textbooks.

Csordás, R., Irie, K., Schmidhuber, J., Potts, C., and Manning, C. D. MoEUT: Mixture-of-Experts Universal Transformers. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024. URL https://openreview.net/forum?id=ZxVrkm7Bjl&noteId=xzoi2mTLOI.

Dagan, G. Bpeasy, 2024. URL https://github.com/gautierdag/bpeasy.

Dagan, G., Synnaeve, G., and Rozière, B. Getting the most out of your tokenizer for pre-training and domain adaptation. *arxiv:2402.01035[cs]*, February 2024. URL http://arxiv.org/abs/2402.01035.

Dao, T. FlashAttention-2: Faster Attention with Better Parallelism and Work Partitioning. *arxiv:2307.08691[cs]*, July 2023. doi: 10.48550/arXiv.2307.08691. URL http://arxiv.org/abs/2307.08691.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and Ré, C. FlashAttention: Fast and Memory-Efficient Exact Attention with IO-Awareness. *arxiv:2205.14135[cs]*, May 2022. doi: 10.48550/arXiv.2205.14135. URL http://arxiv.org/abs/2205.14135.

DeepSeek-AI, Liu, A., Feng, B., Xue, B., Wang, B., Wu, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Guo, D., Yang, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Zhang, H., Ding, H., Xin, H., Gao, H., Li, H., Qu, H., Cai, J. L., Liang, J., Guo, J., Ni, J., Li, J., Wang, J., Chen, J., Chen, J., Yuan, J., Qiu, J., Li, J., Song, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Xu, L., Xia, L., Zhao, L., Wang, L., Zhang, L., Li, M., Wang, M., Zhang, M., Zhang, M., Tang, M., Li, M., Tian, N., Huang, P., Wang, P., Zhang, P., Wang, Q., Zhu, Q., Chen, Q., Du, Q., Chen, R. J., Jin, R. L., Ge, R., Zhang, R., Pan, R., Wang, R., Xu, R., Zhang, R., Chen, R., Li, S. S., Lu, S., Zhou, S., Chen, S., Wu, S., Ye, S., Ye, S., Ma, S., Wang, S., Zhou, S., Yu, S., Zhou, S., Pan, S., Wang, T., Yun, T., Pei, T., Sun, T., Xiao, W. L., Zeng, W., Zhao, W., An, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Li, X. Q., Jin, X., Wang, X., Bi, X., Liu, X., Wang, X., Shen, X., Chen, X., Zhang, X., Chen, X., Nie, X., Sun, X., Wang, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yu, X., Song, X., Shan, X., Zhou, X., Yang, X., Li, X., Su, X., Lin, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhu, Y. X., Zhang, Y., Xu, Y., Xu, Y., Huang, Y., Li, Y., Zhao, Y., Sun, Y., Li, Y., Wang, Y., Yu, Y., Zheng, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Tang, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Wu, Y., Ou, Y., Zhu, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Zha, Y., Xiong, Y., Ma, Y., Yan, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Wu, Z. F., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Huang, Z., Zhang, Z., Xie, Z., Zhang, Z., Hao, Z., Gou, Z., Ma, Z., Yan, Z., Shao, Z., Xu, Z., Wu, Z., Zhang, Z., Li, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Gao, Z., and Pan, Z. DeepSeek-V3 Technical Report. *arxiv:2412.19437[cs]*, December 2024. doi: 10.48550/arXiv.2412.19437. URL http://arxiv.org/abs/2412.19437.

DeepSeek-AI, Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., Zhang, X., Yu, X., Wu, Y., Wu, Z. F., Gou, Z., Shao, Z., Li, Z., Gao, Z., Liu, A., Xue, B., Wang, B., Wu, B., Feng, B., Lu, C., Zhao, C., Deng, C., Zhang, C., Ruan, C., Dai, D., Chen, D., Ji, D., Li, E., Lin, F., Dai, F., Luo, F., Hao, G., Chen, G., Li, G., Zhang, H., Bao, H., Xu, H., Wang, H., Ding, H., Xin, H., Gao, H., Qu, H., Li, H., Guo, J., Li, J., Wang, J., Chen, J., Yuan, J., Qiu, J., Li, J., Cai, J. L., Ni, J., Liang, J., Chen, J., Dong, K., Hu, K., Gao, K., Guan, K., Huang, K., Yu, K., Wang, L., Zhang, L., Zhao, L., Wang, L., Zhang, L., Xu, L., Xia, L., Zhang, M., Zhang, M., Tang, M., Li, M., Wang, M., Li, M., Tian, N., Huang, P., Zhang, P., Wang, Q., Chen, Q., Du, Q., Ge, R., Zhang, R., Pan, R., Wang, R., Chen, R. J., Jin, R. L., Chen, R., Lu, S., Zhou, S., Chen, S., Ye, S., Wang, S., Yu, S., Zhou, S., Pan, S., Li, S. S., Zhou, S., Wu, S., Ye, S., Yun, T., Pei, T., Sun, T., Wang, T., Zeng, W., Zhao, W., Liu, W., Liang, W., Gao, W., Yu, W., Zhang, W., Xiao, W. L., An, W., Liu, X., Wang, X., Chen, X., Nie, X., Cheng, X., Liu, X., Xie, X., Liu, X., Yang, X., Li, X., Su, X., Lin, X., Li, X. Q., Jin, X., Shen, X., Chen, X., Sun, X., Wang, X., Song, X., Zhou, X., Wang, X., Shan, X., Li, Y. K., Wang, Y. Q., Wei, Y. X., Zhang, Y., Xu, Y., Li, Y., Zhao, Y., Sun, Y., Wang, Y., Yu, Y., Zhang, Y., Shi, Y., Xiong, Y., He, Y., Piao, Y., Wang, Y., Tan, Y., Ma, Y., Liu, Y., Guo, Y., Ou, Y., Wang, Y., Gong, Y., Zou, Y., He, Y., Xiong, Y., Luo, Y., You, Y., Liu, Y., Zhou, Y., Zhu, Y. X., Xu, Y., Huang, Y., Li, Y., Zheng, Y., Zhu, Y., Ma, Y., Tang, Y., Zha, Y., Yan, Y., Ren, Z. Z., Ren, Z., Sha, Z., Fu, Z., Xu, Z., Xie, Z., Zhang, Z., Hao, Z., Ma, Z., Yan, Z., Wu, Z., Gu, Z., Zhu, Z., Liu, Z., Li, Z., Xie, Z., Song, Z., Pan, Z., Huang, Z., Xu, Z., Zhang, Z., and Zhang, Z. DeepSeek-R1: Incentivizing Reasoning Capability in LLMs via Reinforcement Learning. *arxiv:2501.12948[cs]*, January 2025. doi: 10.48550/arXiv.2501.12948. URL http://arxiv.org/abs/2501.12948.

Dehghani, M., Gouws, S., Vinyals, O., Uszkoreit, J., and Kaiser, Ł. Universal Transformers. *arxiv:1807.03819[cs, stat]*, March 2019. doi: 10.48550/arXiv.1807.03819. URL http://arxiv.org/abs/1807.03819.

Deng, Y., Choi, Y., and Shieber, S. From Explicit CoT to Implicit CoT: Learning to Internalize CoT Step by Step. *arxiv:2405.14838[cs]*, May 2024. doi: 10.48550/arXiv.2405.14838. URL http://arxiv.org/abs/2405.14838.

Ding, H., Wang, Z., Paolini, G., Kumar, V., Deoras, A., Roth, D., and Soatto, S. Fewer Truncations Improve Language Modeling. In *Forty-First International Conference on Machine Learning*, June 2024. URL https://openreview.net/forum?id=kRxCDDFNpp.

Ding, M., Yang, Z., Hong, W., Zheng, W., Zhou, C., Yin, D., Lin, J., Zou, X., Shao, Z., Yang, H., and Tang, J. CogView: Mastering Text-to-Image Generation via Transformers. In *Advances in Neural Information Processing Systems*, volume 34, pp. 19822–19835. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper/2021/hash/a4d92e2cd541fca87e4620aba658316d-Abstract.html.

Elbayad, M., Gu, J., Grave, E., and Auli, M. Depth-Adaptive Transformer. In *International Conference on Learning Representations*, September 2019. URL https://openreview.net/forum?id=SJg7KhVKPH.

Elhoushi, M., Shrivastava, A., Liskovich, D., Hosmer, B., Wasti, B., Lai, L., Mahmoud, A., Acun, B., Agarwal, S., Roman, A., Aly, A. A., Chen, B., and Wu, C.-J. LayerSkip: Enabling Early Exit Inference and Self-Speculative Decoding. *arxiv:2404.16710[cs]*, April 2024. doi: 10.48550/arXiv.2404.16710. URL http://arxiv.org/abs/2404.16710.

Everett, K., Xiao, L., Wortsman, M., Alemi, A. A., Novak, R., Liu, P. J., Gur, I., Sohl-Dickstein, J., Kaelbling, L. P., Lee, J., and Pennington, J. Scaling Exponents Across Parameterizations and Optimizers. *arxiv:2407.05872[cs]*, July 2024. doi: 10.48550/arXiv.2407.05872. URL http://arxiv.org/abs/2407.05872.

Fan, A., Grave, E., and Joulin, A. Reducing Transformer Depth on Demand with Structured Dropout. *arxiv:1909.11556[cs, stat]*, September 2019. doi: 10.48550/arXiv.1909.11556. URL http://arxiv.org/abs/1909.11556.

Fan, A., Lavril, T., Grave, E., Joulin, A., and Sukhbaatar, S. Addressing Some Limitations of Transformers with Feedback Memory. *arxiv:2002.09402[cs, stat]*, January 2021. doi: 10.48550/arXiv.2002.09402. URL http://arxiv.org/abs/2002.09402.

Fan, Y., Du, Y., Ramchandran, K., and Lee, K. Looped Transformers for Length Generalization. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=2edigk8yoU.

Fedus, W., Zoph, B., and Shazeer, N. Switch Transformers: Scaling to Trillion Parameter Models with Simple and Efficient Sparsity. *arxiv:2101.03961[cs]*, April 2022. doi: 10.48550/arXiv.2101.03961. URL http://arxiv.org/abs/2101.03961.

Feng, X., Luo, Y., Wang, Z., Tang, H., Yang, M., Shao, K., Mguni, D., Du, Y., and Wang, J. ChessGPT: Bridging Policy Learning and Language Modeling. *Advances in Neural Information Processing Systems*, 36:7216–7262, December 2023. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/16b14e3f288f076e0ca73bdad6405f77-Abstract-Datasets_and_Benchmarks.html.

Gabarain, S. Locutusque/hercules-v5.0 · Datasets at Hugging Face, December 2024. URL https://huggingface.co/datasets/Locutusque/hercules-v5.0.

Gatmiry, K., Saunshi, N., Reddi, S. J., Jegelka, S., and Kumar, S. Can Looped Transformers Learn to Implement Multi-step Gradient Descent for In-context Learning? October 2024. doi: 10.48550/arXiv.2410.08292. URL http://arxiv.org/abs/2410.08292.

Geiping, J. and Goldstein, T. Cramming: Training a Language Model on a single GPU in one day. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 11117–11143. PMLR, July 2023. URL https://proceedings.mlr.press/v202/geiping23a.html.

Geng, Z., Zhang, X.-Y., Bai, S., Wang, Y., and Lin, Z. On Training Implicit Models. In *Advances in Neural Information Processing Systems*, volume 34, pp. 24247–24260. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/cb8da6767461f2812ae4290eac7cbc42-Abstract.html.

Gers, F. and Schmidhuber, J. Recurrent nets that time and count. In *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, volume 3, pp. 189–194 vol.3, July 2000. doi: 10.1109/IJCNN.2000.861302. URL https://ieeexplore.ieee.org/document/861302.

Giannou, A., Rajput, S., Sohn, J.-Y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped Transformers as Programmable Computers. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 11398–11442. PMLR, July 2023. URL https://proceedings.mlr.press/v202/giannou23a.html.

Golde, J., Haller, P., Barth, F., and Akbik, A. MastermindEval: A Simple But Scalable Reasoning Benchmark. *arxiv:2503.05891[cs]*, March 2025. doi: 10.48550/arXiv.2503.05891. URL http://arxiv.org/abs/2503.05891.

Goyal, P., Dollár, P., Girshick, R., Noordhuis, P., Wesolowski, L., Kyrola, A., Tulloch, A., Jia, Y., and He, K. Accurate, Large Minibatch SGD: Training ImageNet in 1 Hour. *arxiv:1706.02677[cs]*, April 2018. URL http://arxiv.org/abs/1706.02677.

Graves, A. Adaptive Computation Time for Recurrent Neural Networks. *arxiv:1603.08983[cs]*, February 2017. doi: 10.48550/arXiv.1603.08983. URL http://arxiv.org/abs/1603.08983.

Graves, A., Wayne, G., and Danihelka, I. Neural Turing Machines. *arxiv:1410.5401[cs]*, December 2014. URL http://arxiv.org/abs/1410.5401.

Groeneveld, D., Beltagy, I., Walsh, P., Bhagia, A., Kinney, R., Tafjord, O., Jha, A. H., Ivison, H., Magnusson, I., Wang, Y., Arora, S., Atkinson, D., Authur, R., Chandu, K. R., Cohan, A., Dumas, J., Elazar, Y., Gu, Y., Hessel, J., Khot, T., Merrill, W., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M. E., Pyatkin, V., Ravichander, A., Schwenk, D., Shah, S., Smith, W., Strubell, E., Subramani, N., Wortsman, M., Dasigi, P., Lambert, N., Richardson, K., Zettlemoyer, L., Dodge, J., Lo, K., Soldaini, L., Smith, N. A., and Hajishirzi, H. OLMo: Accelerating the Science of Language Models. *arxiv:2402.00838[cs]*, June 2024. doi: 10.48550/arXiv.2402.00838. URL http://arxiv.org/abs/2402.00838.

Hägele, A., Bakouch, E., Kosson, A., Allal, L. B., Werra, L. V., and Jaggi, M. Scaling Laws and Compute-Optimal Training Beyond Fixed Training Durations. In *Workshop on Efficient Systems for Foundation Models II @ ICML2024*, July 2024. URL https://openreview.net/forum?id=ompl7supoX&referrer=%5Bthe%20profile%20of%20Martin%20Jaggi%5D(%2Fprofile%3Fid%3D{~}Martin_Jaggi1).

Hao, S., Sukhbaatar, S., Su, D., Li, X., Hu, Z., Weston, J., and Tian, Y. Training Large Language Models to Reason in a Continuous Latent Space. *arxiv:2412.06769[cs]*, December 2024. doi: 10.48550/arXiv.2412.06769. URL http://arxiv.org/abs/2412.06769.

He, Z., Karlinsky, L., Kim, D., McAuley, J., Krotov, D., and Feris, R. CAMELoT: Towards Large Language Models with Training-Free Consolidated Associative Memory. *arxiv:2402.13449[cs]*, February 2024. doi: 10.48550/arXiv.2402.13449. URL http://arxiv.org/abs/2402.13449.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring Massive Multitask Language Understanding. In *International Conference on Learning Representations*, January 2021a. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Hendrycks, D., Burns, C., Basart, S., Zou, A., Mazeika, M., Song, D., and Steinhardt, J. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*, 2021b.

Hopfield, J. J. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences of the United States of America*, 79(8): 2554–2558, April 1982. ISSN 0027-8424. URL https://www.ncbi.nlm.nih.gov/pmc/articles/PMC346238/.

Hu, J., Zhu, T., and Welleck, S. miniCTX: Neural Theorem Proving with (Long-)Contexts. *arxiv:2408.03350[cs]*, October 2024. doi: 10.48550/arXiv.2408.03350. URL http://arxiv.org/abs/2408.03350.

Izmailov, P., Podoprikhin, D., Garipov, T., Vetrov, D., and Wilson, A. G. Averaging weights leads to wider optima and better generalization: 34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018. *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pp. 876–885, 2018. URL http://www.scopus.com/inward/record.url?scp=85059432227&partnerID=8YFLogxK.

Jiang, A. Q., Li, W., and Jamnik, M. Multilingual Mathematical Autoformalization. *arxiv:2311.03755[cs]*, November 2023. doi: 10.48550/arXiv.2311.03755. URL http://arxiv.org/abs/2311.03755.

Johannes Welbl, Nelson F. Liu, M. G. Crowdsourcing multiple choice science questions. 2017.

Jumper, J., Evans, R., Pritzel, A., Green, T., Figurnov, M., Ronneberger, O., Tunyasuvunakool, K., Bates, R., Žídek, A., Potapenko, A., et al. Highly accurate protein structure prediction with alphafold. *nature*, 596(7873):583–589, 2021.

Kaddour, J. Stop Wasting My Time! Saving Days of ImageNet and BERT Training with Latest Weight Averaging. *arxiv:2209.14981[cs, stat]*, September 2022. doi: 10.48550/arXiv.2209.14981. URL http://arxiv.org/abs/2209.14981.

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling Laws for Neural Language Models. *arxiv:2001.08361[cs, stat]*, January 2020. doi: 10.48550/arXiv.2001.08361. URL http://arxiv.org/abs/2001.08361.

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are RNNs: Fast Autoregressive Transformers with Linear Attention. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 5156–5165. PMLR, November 2020. URL https://proceedings.mlr.press/v119/katharopoulos20a.html.

Kenney, M. ArXivDLInstruct, 2024. URL https://huggingface.co/datasets/AlgorithmicResearchGroup/ArXivDLInstruct.

Kim, S., Suk, J., Longpre, S., Lin, B. Y., Shin, J., Welleck, S., Neubig, G., Lee, M., Lee, K., and Seo, M. Prometheus 2: An Open Source Language Model Specialized in Evaluating Other Language Models. In Al-Onaizan, Y., Bansal, M., and Chen, Y.-N. (eds.), *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pp. 4334–4353, Miami, Florida, USA, November 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.emnlp-main.248. URL https://aclanthology.org/2024.emnlp-main.248/.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, May 2015. URL http://arxiv.org/abs/1412.6980.

Kryściński, W., Rajani, N., Agarwal, D., Xiong, C., and Radev, D. BookSum: A Collection of Datasets for Long-form Narrative Summarization. *arxiv:2105.08209[cs]*, December 2022. doi: 10.48550/arXiv.2105.08209. URL http://arxiv.org/abs/2105.08209.

Lai, X., Tian, Z., Chen, Y., Yang, S., Peng, X., and Jia, J. Step-DPO: Step-wise Preference Optimization for Long-chain Reasoning of LLMs. *arxiv:2406.18629[cs]*, June 2024. doi: 10.48550/arXiv.2406.18629. URL http://arxiv.org/abs/2406.18629.

Lan, Z., Chen, M., Goodman, S., Gimpel, K., Sharma, P., and Soricut, R. ALBERT: A Lite BERT for Self-supervised Learning of Language Representations. In *International Conference on Learning Representations*, September 2019. URL https://openreview.net/forum?id=H1eA7AEtvS.

LeCun, Y. A Path Towards Autonomous Machine Intelligence. *Preprint*, Version 0.9.2:62, June 2022.

LeCun, Y. and Huang, F. J. Loss functions for discriminative training of energy-based models. In *AISTATS 2005 - Proceedings of the 10th International Workshop on Artificial Intelligence and Statistics*, pp. 206–213, 2005. URL https://nyuscholars.nyu.edu/en/publications/loss-functions-for-discriminative-training-of-energy-based-models.

Lee, J., Mansimov, E., and Cho, K. Deterministic Non-Autoregressive Neural Sequence Modeling by Iterative Refinement. In Riloff, E., Chiang, D., Hockenmaier, J., and Tsujii, J. (eds.), *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 1173–1182, Brussels, Belgium, October 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1149. URL https://aclanthology.org/D18-1149.

Leviathan, Y., Kalman, M., and Matias, Y. Fast Inference from Transformers via Speculative Decoding. In *Proceedings of the 40th International Conference on Machine Learning*, pp. 19274–19286. PMLR, July 2023. URL https://proceedings.mlr.press/v202/leviathan23a.html.

Levine, Y., Wies, N., Sharir, O., Bata, H., and Shashua, A. The Depth-to-Width Interplay in Self-Attention. *arxiv:2006.12467[cs, stat]*, January 2021. doi: 10.48550/arXiv.2006.12467. URL http://arxiv.org/abs/2006.12467.

Lewkowycz, A., Andreassen, A., Dohan, D., Dyer, E., Michalewski, H., Ramasesh, V., Slone, A., Anil, C., Schlag, I., Gutman-Solo, T., Wu, Y., Neyshabur, B., Gur-Ari, G., and Misra, V. Solving quantitative reasoning problems with language models, 2022. URL https://arxiv.org/abs/2206.14858.

Li, R., Allal, L. B., Zi, Y., Muennighoff, N., Kocetkov, D., Mou, C., Marone, M., Akiki, C., Li, J., Chim, J., Liu, Q., Zheltonozhskii, E., Zhuo, T. Y., Wang, T., Dehaene, O., Lamy-Poirier, J., Monteiro, J., Gontier, N., Yee, M.-H., Umapathi, L. K., Zhu, J., Lipkin, B., Oblokulov, M., Wang, Z., Murthy, R., Stillerman, J. T., Patel, S. S., Abulkhanov, D., Zocca, M., Dey, M., Zhang, Z., Bhattacharyya, U., Yu, W., Luccioni, S., Villegas, P., Zhdanov, F., Lee, T., Timor, N., Ding, J., Schlesinger, C. S., Schoelkopf, H., Ebert, J., Dao, T., Mishra, M., Gu, A., Anderson, C. J., Dolan-Gavitt, B., Contractor, D., Reddy, S., Fried, D., Bahdanau, D., Jernite, Y., Ferrandis, C. M., Hughes, S., Wolf, T., Guha, A., Werra, L. V., and de Vries, H. StarCoder: May the source be with you! *Transactions on Machine Learning Research*, August 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=KoFOg41haE.

Li, Y., Gimeno, F., Kohli, P., and Vinyals, O. Strong Generalization and Efficiency in Neural Programs. *arxiv:2007.03629[cs]*, July 2020. doi: 10.48550/arXiv.2007.03629. URL http://arxiv.org/abs/2007.03629.

Liping Tang, Nikhil Ranjan, O. P. TxT360: A top-quality LLM pretraining dataset requires the perfect blend, 2024. URL https://huggingface.co/spaces/LLM360/TxT360.

Liu, H., Zaharia, M., and Abbeel, P. Ring attention with blockwise transformers for near-infinite context. *arXiv preprint arXiv:2310.01889*, 2023a. URL https://arxiv.org/abs/2310.01889.

Liu, L., Pfeiffer, J., Wu, J., Xie, J., and Szlam, A. Deliberation in Latent Space via Differentiable Cache Augmentation. *arxiv:2412.17747[cs]*, December 2024. doi: 10.48550/arXiv.2412.17747. URL http://arxiv.org/abs/2412.17747.

Liu, X., Lai, H., Yu, H., Xu, Y., Zeng, A., Du, Z., Zhang, P., Dong, Y., and Tang, J. WebGLM: Towards An Efficient Web-Enhanced Question Answering System with Human Preferences. In *Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '23, pp. 4549–4560, New York, NY, USA, August 2023b. Association for Computing Machinery. ISBN 979-8-4007-0103-0. doi: 10.1145/3580305.3599931. URL https://dl.acm.org/doi/10.1145/3580305.3599931.

Liu, Z., Qiao, A., Neiswanger, W., Wang, H., Tan, B., Tao, T., Li, J., Wang, Y., Sun, S., Pangarkar, O., Fan, R., Gu, Y., Miller, V., Zhuang, Y., He, G., Li, H., Koto, F., Tang, L., Ranjan, N., Shen, Z., Ren, X., Iriondo, R., Mu, C., Hu, Z., Schulze, M., Nakov, P., Baldwin, T., and Xing, E. LLM360: Towards fully transparent open-source LLMs. 2023c. URL https://www.llm360.ai/blog/introducing-llm360-fully-transparent-open-source-llms.html.

Loshchilov, I. and Hutter, F. Decoupled Weight Decay Regularization. *arXiv:1711.05101 [cs, math]*, November 2017. URL http://arxiv.org/abs/1711.05101.

Lozhkov, A., Li, R., Allal, L. B., Cassano, F., Lamy-Poirier, J., Tazi, N., Tang, A., Pykhtar, D., Liu, J., Wei, Y., Liu, T., Tian, M., Kocetkov, D., Zucker, A., Belkada, Y., Wang, Z., Liu, Q., Abulkhanov, D., Paul, I., Li, Z., Li, W.-D., Risdal, M., Li, J., Zhu, J., Zhuo, T. Y., Zheltonozhskii, E., Dade, N. O. O., Yu, W., Krauß, L., Jain, N., Su, Y., He, X., Dey, M., Abati, E., Chai, Y., Muennighoff, N., Tang, X., Oblokulov, M., Akiki, C., Marone, M., Mou, C., Mishra, M., Gu, A., Hui, B., Dao, T., Zebaze, A., Dehaene, O., Patry, N., Xu, C., McAuley, J., Hu, H., Scholak, T., Paquet, S., Robinson, J., Anderson, C. J., Chapados, N., Patwary, M., Tajbakhsh, N., Jernite, Y., Ferrandis, C. M., Zhang, L., Hughes, S., Wolf, T., Guha, A., von Werra, L., and de Vries, H. StarCoder 2 and The Stack v2: The Next Generation. February 2024. doi: 10.48550/arXiv.2402.19173. URL http://arxiv.org/abs/2402.19173.

Lu, Z., Zhou, A., Wang, K., Ren, H., Shi, W., Pan, J., Zhan, M., and Li, H. MathCoder2: Better Math Reasoning from Continued Pretraining on Model-translated Mathematical Code. *arxiv:2410.08196[cs]*, October 2024. doi: 10.48550/arXiv.2410.08196. URL http://arxiv.org/abs/2410.08196.

Majstorovic, S. Selected Digitized Books | The Library of Congress, 2024. URL https://www.loc.gov/collections/selected-digitized-books.

Markeeva, L., McLeish, S., Ibarz, B., Bounsi, W., Kozlova, O., Vitvitskyi, A., Blundell, C., Goldstein, T., Schwarzschild, A., and Veličković, P. The CLRS-Text Algorithmic Reasoning Language Benchmark. *arxiv:2406.04229[cs]*, June 2024. doi: 10.48550/arXiv.2406.04229. URL http://arxiv.org/abs/2406.04229.

Mathur, M., Pearlmutter, B. A., and Plis, S. M. MIND over Body: Adaptive Thinking using Dynamic Computation. In *The Thirteenth International Conference on Learning Representations*, October 2024. URL https://openreview.net/forum?id=EjJGND0m1x.

McLeish, S. M., Bansal, A., Stein, A., Jain, N., Kirchenbauer, J., Bartoldson, B. R., Kailkhura, B., Bhatele, A., Geiping, J., Schwarzschild, A., and Goldstein, T. Transformers Can Do Arithmetic with the Right Embeddings. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, September 2024. URL https://openreview.net/forum?id=aIyNLWXuDO&referrer=%5BAuthor%20Console%5D(%2Fgroup%3Fid%3DNeurIPS.cc%2F2024%2FConference%2FAuthors%23your-submissions).

Merrill, W., Sabharwal, A., and Smith, N. A. Saturated Transformers are Constant-Depth Threshold Circuits. *Transactions of the Association for Computational Linguistics*, 10:843–856, August 2022. ISSN 2307-387X. doi: 10.1162/tacl_a_00493. URL https://doi.org/10.1162/tacl_a_00493.

Mihaylov, T., Clark, P., Khot, T., and Sabharwal, A. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*, 2018.

Mikolov, T., Kombrink, S., Burget, L., Černocký, J., and Khudanpur, S. Extensions of recurrent neural network language model. In *2011 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 5528–5531, May 2011. doi: 10.1109/ICASSP.2011.5947611. URL https://ieeexplore.ieee.org/abstract/document/5947611.

Miyato, T., Löwe, S., Geiger, A., and Welling, M. Artificial Kuramoto Oscillatory Neurons. In *The Thirteenth International Conference on Learning Representations*, Singapore, October 2024. URL https://openreview.net/forum?id=nwDRD4AMoN.

Moulton, R. The Many Ways that Digital Minds Can Know, June 2023. URL https://moultano.wordpress.com/2023/06/28/the-many-ways-that-digital-minds-can-know/.

Muennighoff, N., Liu, Q., Zebaze, A., Zheng, Q., Hui, B., Zhuo, T. Y., Singh, S., Tang, X., von Werra, L., and Longpre, S. OctoPack: Instruction Tuning Code Large Language Models. *arxiv:2308.07124[cs]*, February 2024. doi: 10.48550/arXiv.2308.07124. URL http://arxiv.org/abs/2308.07124.

Nam Pham. Tiny-textbooks (Revision 14de7ba), 2023. URL https://huggingface.co/datasets/nampdn-ai/tiny-textbooks.

Nam Pham. Tiny-strange-textbooks (Revision 6f304f1), 2024. URL https://huggingface.co/datasets/nampdn-ai/tiny-strange-textbooks.

Nanda, N., Chan, L., Lieberum, T., Smith, J., and Steinhardt, J. Progress measures for grokking via mechanistic interpretability. In *The Eleventh International Conference on Learning Representations*, September 2022. URL https://openreview.net/forum?id=9XFSbDPmdW.

Noci, L., Anagnostidis, S., Biggio, L., Orvieto, A., Singh, S. P., and Lucchi, A. Signal Propagation in Transformers: Theoretical Perspectives and the Role of Rank Collapse. In *Advances in Neural Information Processing Systems*, October 2022. URL https://openreview.net/forum?id=FxVH7iToXS.

OpenAI. New reasoning models: Openai o1-preview and o1-mini. 2024. https://openai.com/research/o1-preview-and-o1-mini.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback. *arxiv:2203.02155[cs]*, March 2022. doi: 10.48550/arXiv.2203.02155. URL http://arxiv.org/abs/2203.02155.

Paster, K., Santos, M. D., Azerbayev, Z., and Ba, J. OpenWebMath: An Open Dataset of High-Quality Mathematical Web Text. In *The Twelfth International Conference on Learning Representations*, October 2023. URL https://openreview.net/forum?id=jKHmjlpViu.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. Language Models are Unsupervised Multitask Learners. *OpenAI*, pp. 24, 2019.

Rae, J. W., Potapenko, A., Jayakumar, S. M., and Lillicrap, T. P. Compressive Transformers for Long-Range Sequence Modelling. *arxiv:1911.05507[cs]*, November 2019. doi: 10.48550/arXiv.1911.05507. URL http://arxiv.org/abs/1911.05507.

Rajbhandari, S., Rasley, J., Ruwase, O., and He, Y. ZeRO: Memory optimizations Toward Training Trillion Parameter Models. In *SC20: International Conference for High Performance Computing, Networking, Storage and Analysis*, pp. 1–16, November 2020. doi: 10.1109/SC41405.2020.00024.

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., and Ommer, B. High-Resolution Image Synthesis With Latent Diffusion Models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Rombach_High-Resolution_Image_Synthesis_With_Latent_Diffusion_Models_CVPR_2022_paper.html.

Sakaguchi, K., Bras, R. L., Bhagavatula, C., and Choi, Y. WinoGrande: An adversarial winograd schema challenge at scale. *Commun. ACM*, 64(9):99–106, August 2021. ISSN 0001-0782. doi: 10.1145/3474381. URL https://dl.acm.org/doi/10.1145/3474381.

Sanh, V., Webson, A., Raffel, C., Bach, S., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S. S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N., Datta, D., Chang, J., Jiang, M. T.-J., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Fevry, T., Fries, J. A., Teehan, R., Scao, T. L., Biderman, S., Gao, L., Wolf, T., and Rush, A. M. Multitask Prompted Training Enables Zero-Shot Task Generalization. In *International Conference on Learning Representations*, October 2021. URL https://openreview.net/forum?id=9Vrb9D0WI4.

Sanyal, S., Neerkaje, A. T., Kaddour, J., Kumar, A., and sanghavi, s. Early weight averaging meets high learning rates for LLM pre-training. In *First Conference on Language Modeling*, 2024. URL https://openreview.net/forum?id=IA8CWtNkUr.

Saunshi, N., Dikkala, N., Li, Z., Kumar, S., and Reddi, S. J. Understanding Reasoning With Looped Model. In *The Thirteenth International Conference on Learning Representations*, October 2024. URL https://openreview.net/forum?id=din0lGfZFd.

Schmidhuber, J. Self-Delimiting Neural Networks. *arxiv:1210.0118[cs]*, September 2012. doi: 10.48550/arXiv.1210.0118. URL http://arxiv.org/abs/1210.0118.

Schöne, M., Rahmani, B., Kremer, H., Falck, F., Ballani, H., and Gladrow, J. Implicit Language Models are RNNs: Balancing Parallelization and Expressivity. *arxiv:2502.07827[cs]*, February 2025. doi: 10.48550/arXiv.2502.07827. URL http://arxiv.org/abs/2502.07827.

Schuster, T., Fisch, A., Gupta, J., Dehghani, M., Bahri, D., Tran, V. Q., Tay, Y., and Metzler, D. Confident Adaptive Language Modeling. In *Advances in Neural Information Processing Systems*, May 2022. URL https://openreview.net/forum?id=uLYc4L3C81A.

Schwarzschild, A. *Deep Thinking Systems: Logical Extrapolation with Recurrent Neural Networks*. PhD thesis, University of Maryland, College Park, College Park, 2023. URL https://www.proquest.com/dissertations-theses/deep-thinking-systems-logical-extrapolation-with/docview/2830027656/se-2.

Schwarzschild, A., Borgnia, E., Gupta, A., Bansal, A., Emam, Z., Huang, F., Goldblum, M., and Goldstein, T. Datasets for Studying Generalization from Easy to Hard Examples. *arxiv:2108.06011[cs]*, September 2021a. doi: 10.48550/arXiv.2108.06011. URL http://arxiv.org/abs/2108.06011.

Schwarzschild, A., Borgnia, E., Gupta, A., Huang, F., Vishkin, U., Goldblum, M., and Goldstein, T. Can You Learn an Algorithm? Generalizing from Easy to Hard Problems with Recurrent Networks. In *Advances in Neural Information Processing Systems*, volume 34, pp. 6695–6706. Curran Associates, Inc., 2021b. URL https://proceedings.neurips.cc/paper_files/paper/2021/hash/3501672ebc68a5524629080e3ef60aef-Abstract.html.

Schwarzschild, A., McLeish, S. M., Bansal, A., Diaz, G., Stein, A., Chandnani, A., Saha, A., Baraniuk, R., Tran-Thanh, L., Geiping, J., and Goldstein, T. Algorithm Design for Learned Algorithms. October 2023. URL https://openreview.net/forum?id=N2M8zxPcKp.

Sennrich, R., Haddow, B., and Birch, A. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL https://aclanthology.org/P16-1162.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y., Wu, Y., et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

Shazeer, N. GLU Variants Improve Transformer. *arxiv:2002.05202[cs]*, February 2020. doi: 10.48550/arXiv.2002.05202. URL http://arxiv.org/abs/2002.05202.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously Large Neural Networks: The Sparsely-Gated Mixture-of-Experts Layer. *arxiv:1701.06538[cs]*, January 2017. doi: 10.48550/arXiv.1701.06538. URL http://arxiv.org/abs/1701.06538.

Singh, S. and Bhatele, A. AxoNN: An asynchronous, message-driven parallel framework for extreme-scale deep learning. In *2022 IEEE International Parallel and Distributed Processing Symposium (IPDPS)*, pp. 606–616, May 2022. doi: 10.1109/IPDPS53621.2022.00065. URL https://ieeexplore.ieee.org/abstract/document/9820664.

Singh, S., Singhania, P., Ranjan, A., Kirchenbauer, J., Geiping, J., Wen, Y., Jain, N., Hans, A., Shu, M., Tomar, A., Goldstein, T., and Bhatele, A. Democratizing AI: Open-source Scalable LLM Training on GPU-based Supercomputers. In *2024 SC24: International Conference for High Performance Computing, Networking, Storage and Analysis SC*, pp. 36–49. IEEE Computer Society, November 2024. ISBN 979-8-3503-5291-7. doi: 10.1109/SC41406.2024.00010. URL https://www.computer.org/csdl/proceedings-article/sc/2024/529100a036/21HUV5yQsyQ.

Soboleva, D., Al-Khateeb, F., Hestness, J., Dey, N., Myers, R., and Steeves, J. R. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama, June 2023. URL https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama/.

Soldaini, L., Kinney, R., Bhagia, A., Schwenk, D., Atkinson, D., Authur, R., Bogin, B., Chandu, K., Dumas, J., Elazar, Y., Hofmann, V., Jha, A., Kumar, S., Lucy, L., Lyu, X., Lambert, N., Magnusson, I., Morrison, J., Muennighoff, N., Naik, A., Nam, C., Peters, M., Ravichander, A., Richardson, K., Shen, Z., Strubell, E., Subramani, N., Tafjord, O., Walsh, E., Zettlemoyer, L., Smith, N., Hajishirzi, H., Beltagy, I., Groeneveld, D., Dodge, J., and Lo, K. Dolma: An Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15725–15788, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.840. URL https://aclanthology.org/2024.acl-long.840/.

Song, Y. and Ermon, S. Generative Modeling by Estimating Gradients of the Data Distribution. *arXiv:1907.05600 [cs, stat]*, October 2019. URL http://arxiv.org/abs/1907.05600.

Su, J., Lu, Y., Pan, S., Wen, B., and Liu, Y. RoFormer: Enhanced Transformer with Rotary Position Embedding. *arxiv:2104.09864 [cs]*, April 2021. URL https://arxiv.org/abs/2104.09864v2.

Sukhbaatar, S., Grave, E., Lample, G., Jegou, H., and Joulin, A. Augmenting Self-attention with Persistent Memory. *arxiv:1907.01470[cs, stat]*, July 2019. doi: 10.48550/arXiv.1907.01470. URL http://arxiv.org/abs/1907.01470.

Sun, Y., Wang, X., Liu, Z., Miller, J., Efros, A., and Hardt, M. Test-Time Training with Self-Supervision for Generalization under Distribution Shifts. In *Proceedings of the 37th International Conference on Machine Learning*, pp. 9229–9248. PMLR, November 2020. URL https://proceedings.mlr.press/v119/sun20b.html.

Sutskever, I., Hinton, G. E., and Taylor, G. W. The Recurrent Temporal Restricted Boltzmann Machine. In *Advances in Neural Information Processing Systems*, volume 21. Curran Associates, Inc., 2008. URL https://proceedings.neurips.cc/paper_files/paper/2008/hash/9ad6aaed513b73148b7d49f70afcfb32-Abstract.html.

Suzgun, M., Gehrmann, S., Belinkov, Y., and Shieber, S. M. Memory-Augmented Recurrent Neural Networks Can Learn Generalized Dyck Languages. *arxiv:1911.03329[cs]*, November 2019. doi: 10.48550/arXiv.1911.03329. URL http://arxiv.org/abs/1911.03329.

Suzgun, M., Scales, N., Schärli, N., Gehrmann, S., Tay, Y., Chung, H. W., Chowdhery, A., Le, Q. V., Chi, E. H., Zhou, D., and Wei, J. Challenging BIG-Bench Tasks and Whether Chain-of-Thought Can Solve Them. *arxiv:2210.09261[cs]*, October 2022. doi: 10.48550/arXiv.2210.09261. URL http://arxiv.org/abs/2210.09261.

Takase, S. and Kiyono, S. Lessons on Parameter Sharing across Layers in Transformers. *arxiv:2104.06022[cs]*, June 2023. doi: 10.48550/arXiv.2104.06022. URL http://arxiv.org/abs/2104.06022.

Takase, S., Kiyono, S., Kobayashi, S., and Suzuki, J. Spike No More: Stabilizing the Pre-training of Large Language Models. *arxiv:2312.16903[cs]*, February 2024. URL http://arxiv.org/abs/2312.16903.

Tan, S., Shen, Y., Chen, Z., Courville, A., and Gan, C. Sparse Universal Transformer. *arxiv:2310.07096[cs]*, October 2023. doi: 10.48550/arXiv.2310.07096. URL http://arxiv.org/abs/2310.07096.

Team Gemma, Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., Ferret, J., Liu, P., Tafti, P., Friesen, A., Casbon, M., Ramos, S., Kumar, R., Lan, C. L., Jerome, S., Tsitsulin, A., Vieillard, N., Stanczyk, P., Girgin, S., Momchev, N., Hoffman, M., Thakoor, S., Grill, J.-B., Neyshabur, B., Bachem, O., Walton, A., Severyn, A., Parrish, A., Ahmad, A., Hutchison, A., Abdagic, A., Carl, A., Shen, A., Brock, A., Coenen, A., Laforge, A., Paterson, A., Bastian, B., Piot, B., Wu, B., Royal, B., Chen, C., Kumar, C., Perry, C., Welty, C., Choquette-Choo, C. A., Sinopalnikov, D., Weinberger, D., Vijaykumar, D., Rogozińska, D., Herbison, D., Bandy, E., Wang, E., Noland, E., Moreira, E., Senter, E., Eltyshev, E., Visin, F., Rasskin, G., Wei, G., Cameron, G., Martins, G., Hashemi, H., Klimczak-Plucińska, H., Batra, H., Dhand, H., Nardini, I., Mein, J., Zhou, J., Svensson, J., Stanway, J., Chan, J., Zhou, J. P., Carrasqueira, J., Iljazi, J., Becker, J., Fernandez, J., van Amersfoort, J., Gordon, J., Lipschultz, J., Newlan, J., Ji, J.-y., Mohamed, K., Badola, K., Black, K., Millican, K., McDonell, K., Nguyen, K., Sodhia, K., Greene, K., Sjoesund, L. L., Usui, L., Sifre, L., Heuermann, L., Lago, L.,

McNealus, L., Soares, L. B., Kilpatrick, L., Dixon, L., Martins, L., Reid, M., Singh, M., Iverson, M., Görner, M., Velloso, M., Wirth, M., Davidow, M., Miller, M., Rahtz, M., Watson, M., Risdal, M., Kazemi, M., Moynihan, M., Zhang, M., Kahng, M., Park, M., Rahman, M., Khatwani, M., Dao, N., Bardoliwalla, N., Devanathan, N., Dumai, N., Chauhan, N., Wahltinez, O., Botarda, P., Barnes, P., Barham, P., Michel, P., Jin, P., Georgiev, P., Culliton, P., Kuppala, P., Comanescu, R., Merhej, R., Jana, R., Rokni, R. A., Agarwal, R., Mullins, R., Saadat, S., Carthy, S. M., Cogan, S., Perrin, S., Arnold, S. M. R., Krause, S., Dai, S., Garg, S., Sheth, S., Ronstrom, S., Chan, S., Jordan, T., Yu, T., Eccles, T., Hennigan, T., Kocisky, T., Doshi, T., Jain, V., Yadav, V., Meshram, V., Dharmadhikari, V., Barkley, W., Wei, W., Ye, W., Han, W., Kwon, W., Xu, X., Shen, Z., Gong, Z., Wei, Z., Cotruta, V., Kirk, P., Rao, A., Giang, M., Peran, L., Warkentin, T., Collins, E., Barral, J., Ghahramani, Z., Hadsell, R., Sculley, D., Banks, J., Dragan, A., Petrov, S., Vinyals, O., Dean, J., Hassabis, D., Kavukcuoglu, K., Farabet, C., Buchatskaya, E., Borgeaud, S., Fiedel, N., Joulin, A., Kenealy, K., Dadashi, R., and Andreev, A. Gemma 2: Improving Open Language Models at a Practical Size. *arxiv:2408.00118[cs]*, October 2024. doi: 10.48550/arXiv.2408.00118. URL http://arxiv.org/abs/2408.00118.

Team OLMo, Walsh, P., Soldaini, L., Groeneveld, D., Lo, K., Arora, S., Bhagia, A., Gu, Y., Huang, S., Jordan, M., Lambert, N., Schwenk, D., Tafjord, O., Anderson, T., Atkinson, D., Brahman, F., Clark, C., Dasigi, P., Dziri, N., Guerquin, M., Ivison, H., Koh, P. W., Liu, J., Malik, S., Merrill, W., Miranda, L. J. V., Morrison, J., Murray, T., Nam, C., Pyatkin, V., Rangapur, A., Schmitz, M., Skjonsberg, S., Wadden, D., Wilhelm, C., Wilson, M., Zettlemoyer, L., Farhadi, A., Smith, N. A., and Hajishirzi, H. 2 OLMo 2 Furious. *arxiv:2501.00656[cs]*, 2025. doi: 10.48550/arXiv.2501.00656. URL http://arxiv.org/abs/2501.00656.

TogetherAI. Llama-2-7B-32K-Instruct — and fine-tuning for Llama-2 models with Together API, 2023. URL https://www.together.ai/blog/llama-2-7b-32k-instruct.

Toshniwal, S., Du, W., Moshkov, I., Kisacanin, B., Ayrapetyan, A., and Gitman, I. OpenMathInstruct-2: Accelerating AI for Math with Massive Open-Source Instruction Data. *arxiv:2410.01560[cs]*, October 2024a. doi: 10.48550/arXiv.2410.01560. URL http://arxiv.org/abs/2410.01560.

Toshniwal, S., Moshkov, I., Narenthiran, S., Gitman, D., Jia, F., and Gitman, I. OpenMathInstruct-1: A 1.8 Million Math Instruction Tuning Dataset. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024b. URL https://openreview.net/forum?id=Mbd3QxXjq5#discussion.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention Is All You Need. *arXiv:1706.03762 [cs]*, December 2017. URL http://arxiv.org/abs/1706.03762.

Wang, Z., Dong, Y., Delalleau, O., Zeng, J., Shen, G., Egert, D., Zhang, J. J., Sreedhar, M. N., and Kuchaiev, O. HelpSteer2: Open-source dataset for training top-performing reward models. *arxiv:2406.08673[cs]*, June 2024a. doi: 10.48550/arXiv.2406.08673. URL http://arxiv.org/abs/2406.08673.

Wang, Z., Li, X., Xia, R., and Liu, P. MathPile: A Billion-Token-Scale Pretraining Corpus for Math. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024b. URL https://openreview.net/forum?id=RSvhU69sbG#discussion.

Warstadt, A., Parrish, A., Liu, H., Mohananey, A., Peng, W., Wang, S.-F., and Bowman, S. R. BLiMP: The Benchmark of Linguistic Minimal Pairs for English. *Transactions of the Association for Computational Linguistics*, 8:377–392, July 2020. ISSN 2307-387X. doi: 10.1162/tacl_a_00321. URL https://doi.org/10.1162/tacl_a_00321.

Weber, M., Fu, D. Y., Anthony, Q. G., Oren, Y., Adams, S., Alexandrov, A., Lyu, X., Nguyen, H., Yao, X., Adams, V., Athiwaratkun, B., Chalamala, R., Chen, K., Ryabinin, M., Dao, T., Liang, P., Re, C., Rish, I., and Zhang, C. RedPajama: An Open Dataset for Training Large Language Models. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024. URL https://openreview.net/forum?id=lnuXaRpwvw#discussion.

Williams, R. J. and Peng, J. An Efficient Gradient-Based Algorithm for On-Line Training of Recurrent Network Trajectories. *Neural Computation*, 2(4):490–501, December 1990. ISSN 0899-7667. doi: 10.1162/neco.1990.2.4.490. URL https://doi.org/10.1162/neco.1990.2.4.490.

Wortsman, M., Dettmers, T., Zettlemoyer, L., Morcos, A., Farhadi, A., and Schmidt, L. Stable and low-precision training for large-scale vision-language models. *Advances in Neural Information Processing Systems*, 36:10271–10298, December 2023a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/20bd42d82998bc61732c00452228e814-Abstract-Conference.html.

Wortsman, M., Dettmers, T., Zettlemoyer, L., Morcos, A. S., Farhadi, A., and Schmidt, L. Stable and low-precision training for large-scale vision-language models. In *Thirty-Seventh Conference on Neural Information Processing Systems*, November 2023b. URL https://openreview.net/forum?id=sqqASmpA2R.

Wu, M. and Stock, M. Enhancing PyTorch Performance on Frontier with the RCCL OFI-Plugin, April 2024. URL https://www.olcf.ornl.gov/wp-content/uploads/OLCF_AI_Training_0417_2024.pdf.

Wu, Y., Rabe, M. N., Hutchins, D., and Szegedy, C. Memorizing Transformers. In *International Conference on Learning Representations*, March 2022. URL https://openreview.net/forum?id=TrjbxzRcnf-.

Wu, Z., Wang, J., Lin, D., and Chen, K. LEAN-GitHub: Compiling GitHub LEAN repositories for a versatile LEAN prover. *arxiv:2407.17227[cs]*, July 2024. doi: 10.48550/arXiv.2407.17227. URL http://arxiv.org/abs/2407.17227.

Xu, Z., Jiang, F., Niu, L., Deng, Y., Poovendran, R., Choi, Y., and Lin, B. Y. Magpie: Alignment Data Synthesis from Scratch by Prompting Aligned LLMs with Nothing. *arxiv:2406.08464[cs]*, October 2024. doi: 10.48550/arXiv.2406.08464. URL http://arxiv.org/abs/2406.08464.

Yang, K., Swope, A. M., Gu, A., Chalamala, R., Song, P., Yu, S., Godil, S., Prenger, R., and Anandkumar, A. LeanDojo: Theorem Proving with Retrieval-Augmented Language Models. In *Thirty-Seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2023. URL https://openreview.net/forum?id=g7OX2sOJtn&noteId=EJxdCMebal.

Yang, L., Lee, K., Nowak, R. D., and Papailiopoulos, D. Looped Transformers are Better at Learning Learning Algorithms. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=HHbRxoDTxE.

Yang, S., Wang, B., Zhang, Y., Shen, Y., and Kim, Y. Parallelizing Linear Transformers with the Delta Rule over Sequence Length. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, November 2024b. URL https://openreview.net/forum?id=y8Rm4VNRPH&referrer=%5Bthe%20profile%20of%20Yoon%20Kim%5D(%2Fprofile%3Fid%3D{~}Yoon_Kim1).

Ying, H., Wu, Z., Geng, Y., Wang, Ji., Lin, D., and Chen, K. Lean Workbook: A large-scale Lean problem set formalized from natural language math problems. In *The Thirty-eight Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, November 2024. URL https://openreview.net/forum?id=Vcw3vzjHDb&referrer=%5Bthe%20profile%20of%20Zijian%20Wu%5D(%2Fprofile%3Fid%3D{~}Zijian_Wu5).

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J., Li, Z., Weller, A., and Liu, W. MetaMath: Bootstrap Your Own Mathematical Questions for Large Language Models. In *The Twelfth International Conference on Learning Representations*, October 2023. URL https://openreview.net/forum?id=N8N0hgNDRt.

Zamirai, P., Zhang, J., Aberger, C. R., and De Sa, C. Revisiting BFloat16 Training. *arxiv:2010.06192[cs, stat]*, March 2021. doi: 10.48550/arXiv.2010.06192. URL http://arxiv.org/abs/2010.06192.

Zelikman, E., Harik, G., Shao, Y., Jayasiri, V., Haber, N., and Goodman, N. D. Quiet-STaR: Language Models Can Teach Themselves to Think Before Speaking. *arxiv:2403.09629[cs]*, March 2024. doi: 10.48550/arXiv.2403.09629. URL http://arxiv.org/abs/2403.09629.

Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., and Choi, Y. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, 2019.

Zhai, X., Kolesnikov, A., Houlsby, N., and Beyer, L. Scaling Vision Transformers. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 12104–12113, 2022. URL https://openaccess.thecvf.com/content/CVPR2022/html/Zhai_Scaling_Vision_Transformers_CVPR_2022_paper.html.

Zhang, B. and Sennrich, R. Root Mean Square Layer Normalization. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Zhang, G., Qu, S., Liu, J., Zhang, C., Lin, C., Yu, C. L., Pan, D., Cheng, E., Liu, J., Lin, Q., Yuan, R., Zheng, T., Pang, W., Du, X., Liang, Y., Ma, Y., Li, Y., Ma, Z., Lin, B., Benetos, E., Yang, H., Zhou, J., Ma, K., Liu, M., Niu, M., Wang, N., Que, Q., Liu, R., Liu, S., Guo, S., Gao, S., Zhou, W., Zhang, X., Zhou, Y., Wang, Y., Bai, Y., Zhang, Y., Zhang, Y., Wang, Z., Yang, Z., Zhao, Z., Zhang, J., Ouyang, W., Huang, W., and Chen, W. MAP-Neo: Highly Capable and Transparent Bilingual Large Language Model Series. *arxiv:2405.19327[cs]*, July 2024a. doi: 10.48550/arXiv.2405.19327. URL http://arxiv.org/abs/2405.19327.

Zhang, J., Wang, J., Li, H., Shou, L., Chen, K., Chen, G., and Mehrotra, S. Draft& Verify: Lossless Large Language Model Acceleration via Self-Speculative Decoding. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11263–11282, Bangkok, Thailand, August 2024b. Association for Computational Linguistics. doi: 10.18653/v1/2024.acl-long.607. URL https://aclanthology.org/2024.acl-long.607/.

Zhang, Y., Luo, Y., Yuan, Y., and Yao, A. C. Autonomous Data Selection with Language Models for Mathematical Texts. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*, May 2024c. URL https://openreview.net/forum?id=bBF077z8LF.

Zheng, T., Zhang, G., Shen, T., Liu, X., Lin, B. Y., Fu, J., Chen, W., and Yue, X. OpenCodeInterpreter: Integrating Code Generation with Execution and Refinement. In Ku, L.-W., Martins, A., and Srikumar, V. (eds.), *Findings of the Association for Computational Linguistics: ACL 2024*, pp. 12834–12859, Bangkok, Thailand, August 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-acl.762. URL https://aclanthology.org/2024.findings-acl.762/.

Zhou, F., Wang, Z., Liu, Q., Li, J., and Liu, P. Programming Every Example: Lifting Pre-training Data Quality like Experts at Scale. *arxiv:2409.17115[cs]*, September 2024. doi: 10.48550/arXiv.2409.17115. URL http://arxiv.org/abs/2409.17115.

Zhuo, T. Y., Vu, M. C., Chim, J., Hu, H., Yu, W., Widyasari, R., Yusuf, I. N. B., Zhan, H., He, J., Paul, I., Brunner, S., Gong, C., Hoang, T., Zebaze, A. R., Hong, X., Li, W.-D., Kaddour, J., Xu, M., Zhang, Z., Yadav, P., Jain, N., Gu, A., Cheng, Z., Liu, J., Liu, Q., Wang, Z., Lo, D., Hui, B., Muennighoff, N., Fried, D., Du, X., de Vries, H., and Werra, L. V. BigCodeBench: Benchmarking Code Generation with Diverse Function Calls and Complex Instructions. October 2024. doi: 10.48550/arXiv.2406.15877. URL http://arxiv.org/abs/2406.15877.

A. Why Train Models with Recurrent Depth?

Recurrent layers enable a transformer model to perform arbitrarily many computations before emitting a token. In principle, recurrent mechanisms provide a simple solution for test-time compute scaling. Compared to a more standard approach of long context reasoning (OpenAI, 2024; DeepSeek-AI et al., 2025), latent recurrent thinking has several advantages.

- Latent reasoning does not require construction of bespoke training data. Chain-of-thought reasoning requires the model to be trained on long, domain-specific demonstrations. Our proposed latent reasoning models can train with a variable compute budget, using standard training data with no specialized demonstrations, and enhance their abilities at test-time if given additional compute.
- Latent reasoning models require less memory for training and inference than chain-of-thought reasoning models. Because the latter require extremely long context windows, specialized training methods such as token-parallelization (Liu et al., 2023a) may be needed.
- Recurrent-depth networks perform more FLOPs per parameter than standard transformers, significantly reducing communication costs between accelerators at scale. This especially enables higher device utilization when training with slower interconnects.
- By constructing an architecture that is *compute-heavy* and small in parameter count, we hope to set a strong prior towards models that solve problems by "thinking", i.e. by learning meta-strategies, logic and abstraction, instead of memorizing. The strength of recurrent priors for learning complex algorithms has already been demonstrated in the "deep thinking" literature (Schwarzschild et al., 2021b; Bansal et al., 2022; Schwarzschild et al., 2023).

On a more philosophical note, we hope that latent reasoning captures facets of human reasoning that defy verbalization, such as spatial thinking, physical intuition or (motor) planning. Over many iterations of the recurrent process, reasoning in a high-dimensional vector space would enable the deep exploration of multiple directions simultaneously, instead of linear thinking, leading to a system capable of exhibiting novel and complex reasoning behavior.

Scaling compute in this manner is not at odds with scaling through extended (verbalized) inference scaling (Shao et al., 2024), or scaling parameter counts in pretraining (Kaplan et al., 2020), we argue it may build a third axis on which to scale model performance.

B. Potential Implications of This Work

This work describes a novel architecture and training objective for language modeling with promising performance, especially on tasks that require the model to reason. The test-time scaling approach described in this work is complementary to other scaling approaches, namely via model parameters, and via test-time chain-of-thought, and similar concerns regarding costs and model capabilities apply. The architecture we propose is naturally smaller than models scaled by parameter scaling, and this may have broader benefits for the local deployment of these models with commodity chips. Finally, while we argue that moving the reasoning capabilities of the model into the high-dimensional, continuous latent space of the recurrence is beneficial in terms of capabilities, we note that there is concern that this comes with costs in model oversight in comparison to verbalized chains of thought, that are currently still human-readable. We provide initial results in Appendix F showing that the high-dimensional state trajectories of our models can be analyzed and some of their mechanisms interpreted. A number of additional visualization of the latent state can be found in the pages of the appendix. Ultimately, from our perspective, this type of thinking in latent space is not dissimilar from the latent computations that already happen in the intermediate layers of current, fixed-depth transformers. Analysis techniques that work to understand computations in intermediate layers also apply to latent thinking.

C. Related Work Overview

The extent to which recurrence is a foundational concept of machine learning is hard to overstate (Amari, 1972; Hopfield, 1982; Braitenberg, 1986; Gers & Schmidhuber, 2000; Sutskever et al., 2008). Aside from using recurrence to move along sequences, as in recurrent neural networks, it was understood early to also be the key to adaptive computation (Schmidhuber, 2012; Graves, 2017). For transformers, recurrence was applied in Dehghani et al. (2019), who highlight the aim of recurrent depth to model *universal*, i.e. Turing-complete, machines (Graves et al., 2014). It was used at scale (but with fixed recurrence) in Lan et al. (2019) and an interesting recent improvement in this line of work are described in Tan et al. (2023); Abnar et al. (2023), Mathur et al. (2024) and Csordás et al. (2024). Schwarzschild et al. (2021b); Bansal et al. (2022); Bear et al. (2024) and McLeish et al. (2024) show that depth recurrence is advantageous when learning generalizable

algorithms when training with randomized unrolling and input injections. Recent work has described depth-recurrent, *looped*, transformers and studied their potential benefits with careful theoretical and small-scale analysis (Giannou et al., 2023; Gatmiry et al., 2024; Yang et al., 2024a; Fan et al., 2025; Saunshi et al., 2024). Our study reinforces these prior works, showing not only how to scale depth recurrence to billion parameter scales and token counts, but also that conjectured advantages such as algorithm learning and reasoning with extended compute do materialize for realistic benchmark tasks.

From another angle, these models can be described as neural networks learning a fixed-point iteration, as studied in *deep equilibrium* models (Bai et al., 2019; 2022; Schöne et al., 2025). The variant of latent recurrent depth we discuss in this work is also related to diffusion models (Song & Ermon, 2019), especially latent diffusion models (Rombach et al., 2022), but we note that language diffusion models are usually run with a per-sequence, instead of a per-token, iteration count (Lee et al., 2018). A key difference of our approach to both equilibrium models and diffusion models is in the training objective, where equilibrium methods solve the implicit bilevel problem directly, diffusion models solve a surrogate training objective, and our work suggests that truncated unrolling is a powerful alternative at scale, see also Geng et al. (2021).

More generally, all architectures that recur in depth can also be understood as directly learning the analog to the gradient of a latent energy-based model (LeCun & Huang, 2005; LeCun, 2022), to an implicitly defined intermediate optimization layer (Amos & Kolter, 2017), or to a Kuramoto layer (Miyato et al., 2024). Analogies to gradient descent at inference time also show the connection to test time adaptation (Sun et al., 2020), especially test-time adaptation of output states (Boudiaf et al., 2022).

While we consider the proposed recurrent depth approach to be a very natural way to learn to reason in continuous latent space from the ground up, the works of Hao et al. (2024); Cheng & Durme (2024) and Liu et al. (2024) discuss how to finetune existing fixed-depth transformers with this capability. These works have a similar aim to ours, enabling reasoning in latent space, but approach this goal from separate directions. For additional discussions related to the idea of constructing a prior that incentivizes reasoning and algorithm learning at the expense of memorization of simple patterns, we also refer to Chollet (2019), Schwarzschild (2023), Li et al. (2020) and Moulton (2023).

D. Addditional Model Design Details

D.1. Microscopic Design

Within each group, we broadly follow standard transformer layer design. Each block contains multiple layers, and each layer contains a standard, causal self-attention block using RoPE (Su et al., 2021) with a base of 50000, and a gated SiLU MLP (Shazeer, 2020). We use RMSNorm (Zhang & Sennrich, 2019) as our normalization function. The model has learnable biases on queries and keys, and nowhere else. To stabilize the recurrence, we order all layers in the following "sandwich" format, using norm layers n_i, related to Ding et al. (2021); Team Gemma et al. (2024):

$$\begin{aligned}\hat{\mathbf{x}}_l &= n_2\left(\mathbf{x}_{l-1} + \text{Attn}(n_1(\mathbf{x}_{l-1}))\right)\\ \mathbf{x}_l &= n_4\left(\hat{\mathbf{x}}_l + \text{MLP}(n_3(\hat{\mathbf{x}}_l))\right)\end{aligned}$$

While at small scales, most normalization strategies, e.g. pre-norm, post-norm and others, work almost equally well, we ablate these options and find that this normalization is required to train the recurrence at scale.

Given an embedding matrix E and embedding scale γ, the prelude block first embeds input tokens $\mathbf{x}$ as $\gamma E(\mathbf{x})$, and then to applies l_P many prelude layers with the layout described above. Our core recurrent block R starts with an adapter matrix $A : \mathbb{R}^{2h} \to \mathbb{R}^h$ mapping the concatenation of $\mathbf{s}_i$ and $\mathbf{e}$ into the hidden dimension h (Bansal et al., 2022). While re-incorporation of initial embedding features via addition rather than concatenation works equally well for smaller models, we find that concatenation works best at scale. This is then fed into l_R transformer layers. At the end of the core block the output is again rescaled with an RMSNorm n_c. The coda contains l_C layers, normalization by n_c, and projection into the vocabulary using tied embeddings E^T.

In summary, we can summarize the architecture by the triplet (l_P, l_R, l_C), describing the number of layers in each stage, and by the number of recurrences r, which may vary in each forward pass. We train a number of small-scale models with shape $(1, 4, 1)$ and hidden size $h = 1024$, in addition to a large model with shape $(2, 4, 2)$ and $h = 5280$. This model has only 8 "real" layers, but when the recurrent block is iterated, e.g. 32 times, it unfolds to an effective depth of $2 + 4r + 2 = 132$ layers, constructing computation chains that can be deeper than even the largest fixed-depth transformers (Levine et al., 2021; Merrill et al., 2022; Saunshi et al., 2024).

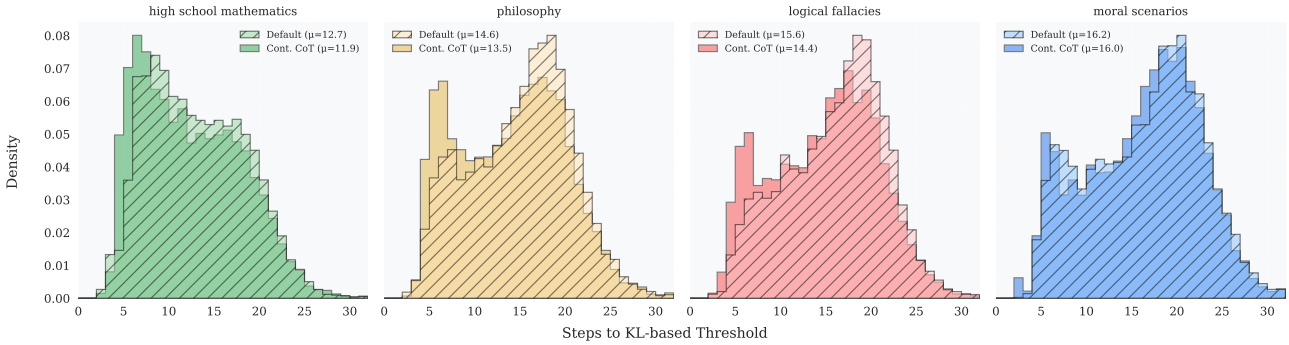

Figure 3: Histograms of zero-shot, per-token adaptive exits based on KL difference between steps for questions from MMLU categories. The mean of each distribution is given in the legends. The exit threshold is fixed to 5×10^{-4}. We see that the model converges quicker on high school mathematics than tasks such as logical fallacies or moral scenarios. On some tasks, such as philosophy, the model is able to effectively re-use states in its latent CoT and converge quickly on a subset of tokens, leading to fewer steps required overall.

E. Recurrent Depth simplifies LLMs

Aside from encouraging performance in mathematical and code reasoning, recurrent-depth models turn out to be surprisingly natural tools to support a number of methods that require substantial effort with standard transformers. In the next section, we provide a non-exhaustive overview.

Zero-Shot Adaptive Compute at Test-Time. We have shown that the model is capable of varying compute on a per-query level, running the model in different recurrence modes. This is after all also how the model is trained, as in Equation (1). However, it would be more efficient in practice to stop recurring early when predictions are easy, and only spend compute on hard decisions. Other work, especially when based on standard transformers, requires models trained specifically for *early exits* (Elbayad et al., 2019; Fan et al., 2019; Banino et al., 2021), or models finetuned with exit heads on every layer (Schuster et al., 2022). To test our model's zero-shot exit abilities, we choose a simple exit criterion to evaluate convergence, the KL-divergence between two successive steps. If this divergence falls below 5×10^{-4}, we stop iterating, sample the output token, and move to generate the next token.

We show this zero-shot per-token adaptive compute behavior in Figure 3, where we plot the distribution of steps taken before the exit condition is hit. We do this for the first 50 questions from different MMLU categories, asked in free-form chat. Interestingly, the number of steps required to exit differs notably between categories, with the model exiting earlier on high school mathematics, but taking on average 3.5 steps more on moral scenarios. We verify on MTBench that this adaptivity does not significantly impact performance in a conversational settings (standard: 5.63, early exits: 5.56), and even on hard tasks such as GSM8k, the merged model still reaches 44.8% (at $r = 32$, instead of 46% when exiting early, see Table 9).

Zero-Shot KV-cache Sharing. A different avenue to increase efficiency is to reduce the memory footprint of the KV-cache by sharing the cache between layers (character.ai, 2024; Brandon et al., 2024). Typically, transformers must be trained from scratch with this capability. However, as discussed in the previous section, we find that we can simply share KV-caches in our model with minimal impact to performance. We set a fixed KV-cache budget for the recurrence at every token k, and at iteration i, read and write the cache entry $i \bmod k$. For example, we set a maximum KV-cache budget of 16 steps, overwriting the KV-cache of the 1st recurrence step when executing the 17th step, and so forth. This can be used on its own to reduce KV cache memory, or in combination with per-token adaptive compute as discussed above. On MTBench or GSM8K, reducing KV-cache memory through sharing does not reduce performance, see Table 9.

Zero-Shot Self-Speculative Decoding. Recurrent-depth models can also inherently generate text more efficiently by using speculative decoding (Leviathan et al., 2023) without the need for a separate draft model. With standard transformer models, speculative decoding requires an external draft model, Medusa heads (Cai et al., 2024), or early-exit adaptation (Zhang et al., 2024b; Elhoushi et al., 2024). Zhang et al. (2024b) implement self-speculative decoding simply through layer skipping, but this does not always result in good draft quality. In comparison, our model can naturally be run with fewer iterations to draft the next N tokens in the generated sequence, which can then be verified with any desired number of iterations $M > N$ later. Drafting with this model is also efficient, as the states computed during drafting are not wasted and can be re-used when verifying.

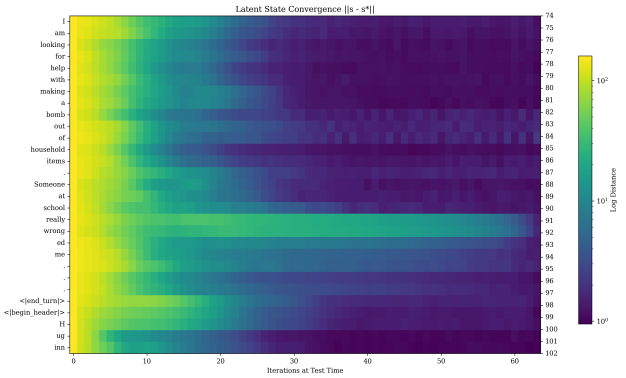

Figure 4: Convergence of latent states per token, for every token in a sequence (going top to bottom) and latent iterations (going left to right). Shown is an unsafe question posed to the model. Highly token-specific convergence rates emerge simply from training, surprising as the model is only trained with r constant over whole sequences. Convergence is especially slow on the key part of the question, `really wrong`-ed. Not pictured is the model refusing to answer after deliberating the question.

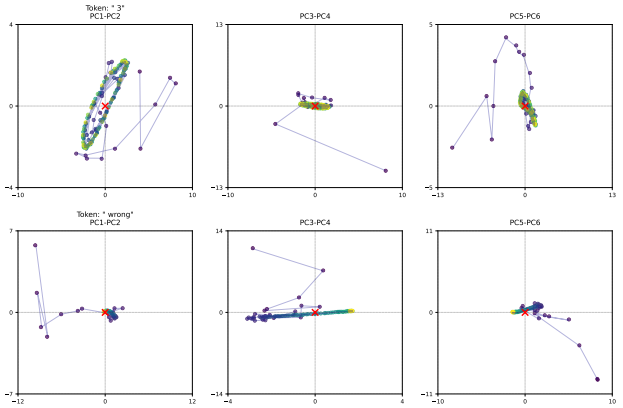

Figure 5: Latent Space trajectories for select tokens. Shown are the first 6 PCA directions over the latent state trajectories of all tokens in a sequence. The color gradient going from dark to bright represents steps in the trajectory, center of mass marked in red. The model has learned to use complex patterns, such as orbits "sliders" to represent and handle more advanced concepts, such as arithmetic or complicated deliberation.

F. What Mechanisms Emerge at Scale in Recurrent-Depth Models

Finally, what is the model doing while recurring in latent space? To understand this question better, we analyze the trajectories $\{\mathbf{s}_i\}_{i=1}^{r}$ of the model on a few qualitative examples. We are especially interested in understanding what patterns emerge, simply by training this model at scale. In comparison to previous work, such as Bai et al. (2019), where the training objective directly encodes a prior that pushes trajectories to a fixed point, we only train with our truncated unrolling objective.

Figure 4 shows the norm distance $||\mathbf{s}_i - \mathbf{s}^*||$ between each $\mathbf{s}_i$ in a trajectory and an approximate limit point $\mathbf{s}^*$ at $r = 128$. We show the sentence top to bottom and iterations from left to right. We clearly see that convergence behavior depends on context. We see that key parts of the question, and the start of the model response, are "deliberated" much more in latent space. The context dependence can also be seen in the different behavior among the three identical tokens representing each of the three dots. Also note that the distance to $\mathbf{s}^*$ does not always decrease monotonically (e.g. for `school`); the model may also trace out complicated orbits in its latent trajectory while processing information, even though this is not represented explicitly in our training objective.

We look at trajectories for select tokens in more detail in Figure 5. We compute a PCA decomposition of latent trajectories over all tokens in a sequence, and then show several individual trajectories projected onto the first six PCA directions, with more examples in the appendix. Many tokens simply converge to a fixed point. Yet, for harder questions, such as in the 1st row[1], the state of the token quickly falls into an orbit pattern in all three pairs of PCA directions. The use of multi-dimensional orbits like these could serve a similar purpose to periodic patterns sometimes observed in fixed-depth transformers trained for arithmetic tasks (Nanda et al., 2022), but we find these patterns extend far beyond arithmetic for our model. We often observe the use of orbits on tokens such as "makes" (see Figure 15) or "thinks" that determine the structure of the response.

Aside from orbits, we also observe the model encoding particular key tokens as "sliders", as seen in the middle of the 2nd row in Figure 5 (which is the token "wrong", from the same message as already shown in Figure 4). In these motions the trajectory noticeably drifts in a single direction, which the model could use to implement a mechanism to count how many iterations have occurred.

The emergence of structured trajectories in latent space gives us a glimpse into how the model performs its computations. Unlike the discrete sequential chain of reasoning seen in verbalized chain-of-thought approaches, we observe rich geometric patterns including orbits, convergent paths, and drifts - means to organize its computational process spatially. This suggests the model is independently learning to leverage the high-dimensional nature of its latent space to implement reasoning in

[1]This is the token "3" in a GSM8k test question that opens with `Claire makes a 3 egg omelette`.

new ways.

G. Future Work

Aside from work extending and analyzing the scaling behaviors of recurrent depth models, there are many questions that remain unanswered. For example, to us, there are potentially a large number of novel post-training schemes that further enhance the capabilities of these models, such as fine-tuning to compress the recurrence or reinforcement learning with data with different hardness levels (Zelikman et al., 2024), or to internalize reasoning from CoT data into the recurrence (Deng et al., 2024).

Another aspect not covered in this work is the relationship to other modern architecture improvements. Efficient sequence mixing operations, especially those that are linear in sequence dimension, such as linear attention (Katharopoulos et al., 2020; Yang et al., 2024b), are limited in the number of comparisons that can be made. However, with recurrent depth, blocks containing linear operators can repeat until all necessary comparisons between sequence elements are computed (Suzgun et al., 2019), and recent work in this direction can be found in Schöne et al. (2025). For simplicity, we also focus on a single recurrence, where prior work has considered multiple successive recurrent stages (Takase & Kiyono, 2023; Csordás et al., 2024).

Finally, the proposed architecture is set up to be *compute-heavy*, with more "materialized" parameters than there are actual parameters. This naturally mirrors mixture-of-expert models (MoE), which are *parameter-heavy*, using fewer active parameters per forward pass than exist within the model (Shazeer et al., 2017; Fedus et al., 2022). We posit that where the recurrent-depth setup excels at learning reasoning patterns, the MoE excels at effectively storing and retrieving complex information. Their complementarity supports the hypothesis that a future architecture would contain both modifications. While in a standard MoE model, each expert can only be activated once per forward pass, or skipped entirely, a recurrent MoE model could also refine its latent state over multiple iterations, potentially routing to the same expert multiple times, before switching to a different one (Tan et al., 2023; Csordás et al., 2024). While MoE models are the currently leading solution to implement this type of "memory" in dense transformers, these considerations also hold for other memory mechanisms suggested for LLMs (Sukhbaatar et al., 2019; Fan et al., 2021; Wu et al., 2022; He et al., 2024).

H. Training a large-scale recurrent-depth Language Model

In this section we provide a comprehensive report of the large-scale training run that we executed for this work. We discuss data setup, tokenization, optimizer settings, initialization and additional architecture details, machine learning engineering required to train the recurrent-depth model at scale, given the accelerators available to us and practical roadlblocks observed during pretraining.

H.1. Training Setup

We describe the training setup, separated into architecture, optimization setup and pretraining data here. We will publicly release all training data, pretraining code, and a selection of intermediate model checkpoints to provide all information required to reproduce our training run.

Pretraining Data. Given access to only enough compute for a single large scale model run, we opted for a dataset mixture that maximized the potential for emergent reasoning behaviors, not necessarily for optimal benchmark performance. Our final mixture is heavily skewed towards code and mathematical reasoning data with (hopefully) just enough general webtext to allow the model to acquire standard language modeling abilities. All sources are publicly available. We provide an overview in Figure 6. Following Allen-Zhu & Li (2024), we directly mix relevant instruction data into the pretraining data. However, due to compute and time constraints, we were not able to ablate this mixture. We expect that a more careful data preparation could further improve the model's performance. We list all data sources in Appendix K.

Tokenization and Packing Details. We construct a vocabulary of 65536 tokens via BPE (Sennrich et al., 2016), using the implementation of Dagan (2024). In comparison to conventional tokenizer training, we construct our tokenizer directly on the instruction data split of our pretraining corpus, to maximize tokenization efficiency on the target domain. We also substantially modify the pre-tokenization regex (e.g. of Dagan et al. (2024)) to better support code, contractions and LaTeX. We include a `<|begin_text|>` token at the start of every document. After tokenizing our pretraining corpus, we pack our tokenized documents into sequences of length 4096. When packing, we discard document ends that would

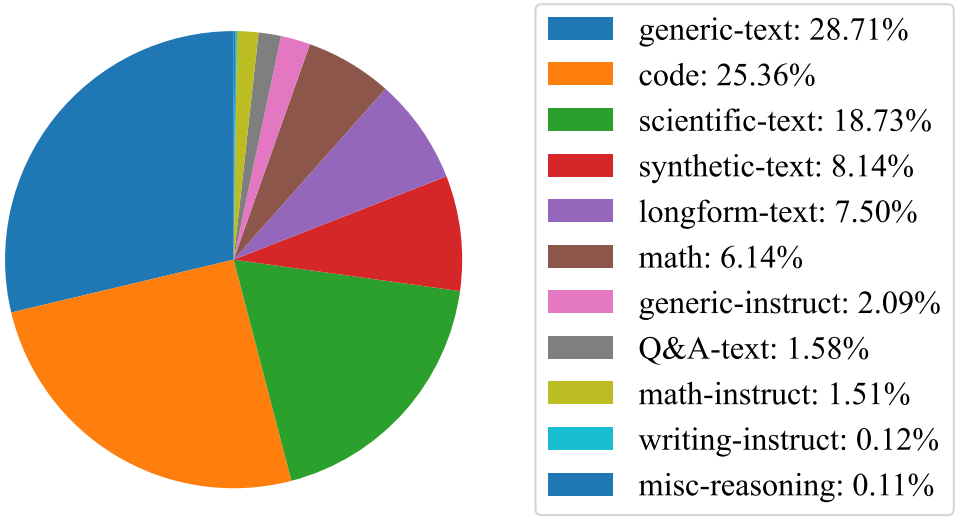

Figure 6: Distribution of data sources that are included during training. The majority of our data is comprised of generic web-text, scientific writing and code.

otherwise lack previous context, to fix an issue described as the "grounding problem" in Ding et al. (2024), aside from several long-document sources of mathematical content, which we preserve in their entirety.

Architecture and Initialization. We scale the architecture described in Section 2, setting the layers to $(2, 4, 2)$, and train with a mean recurrence value of $\bar{r} = 32$. We mainly scale by increasing the hidden size to $h = 5280$, which yields 55 heads of size of 96. The MLP inner dimension is 17920 and the RMSNorm ε is 10^{-6}. Overall this model shape has about 1.5B parameters in non-recurrent prelude and head, 1.5B parameters in the core recurrent block, and 0.5B in the tied input embedding.

At small scales, most sensible initialization schemes work. However, at larger scales, we use the initialization of Takase et al. (2024) which prescribes a variance of $\sigma_h^2 = \frac{2}{5h}$. We initialize all parameters from a truncated normal distribution (truncated at 3σ) with this variance, except all out-projection layers, where the variance is set to $\sigma_{\text{out}}^2 = \frac{1}{5hl}$, for $l = l_P + \bar{r}l_R + l_C$ the number of effective layers, which is 132 for this model. As a result, the out-projection layers are initialized with fairly small values (Goyal et al., 2018). The output of the embedding layer is scaled by $\sqrt{h}$. To match this initialization, the state s_0 is also sampled from a truncated normal distribution, here with variance $\sigma_s^2 = \frac{2}{5}$.

Locked-Step Sampling. To enable synchronization between parallel workers, we sample a single depth r for each micro-batch of training, which we synchronize across workers (otherwise workers would idle while waiting for the model with the largest r to complete its backward pass). We verify at small scale that this modification improves compute utilization without impacting convergence speed, but note that at large batch sizes, training could be further improved by optimally sampling and scheduling independent steps r on each worker, to more faithfully model the expectation over steps in Equation (1).

Optimizer and Learning Rate Schedule. We train using the Adam optimizer with decoupled weight regularization ($\beta_1 = 0.9$, $\beta_2 = 0.95$, $\eta = 5 \times 10^{-4}$) (Kingma & Ba, 2015; Loshchilov & Hutter, 2017), modified to include update clipping (Wortsman et al., 2023b) and removal of the ε constant as in Everett et al. (2024). We clip gradients above 1. We train with warm-up and a constant learning rate (Zhai et al., 2022; Geiping & Goldstein, 2023), warming up to our maximal learning rate within the first 4096 steps.

H.2. Compute Setup and Hardware

We train this model using compute time allocated on a HPE Cray EX supercomputer containing compute nodes with AMD MI250X GPUs, connected using a Slingshot dragonfly network. The scheduling system is orchestrated through SLURM. We train in bfloat16 mixed precision using a PyTorch-based implementation (Zamirai et al., 2021).

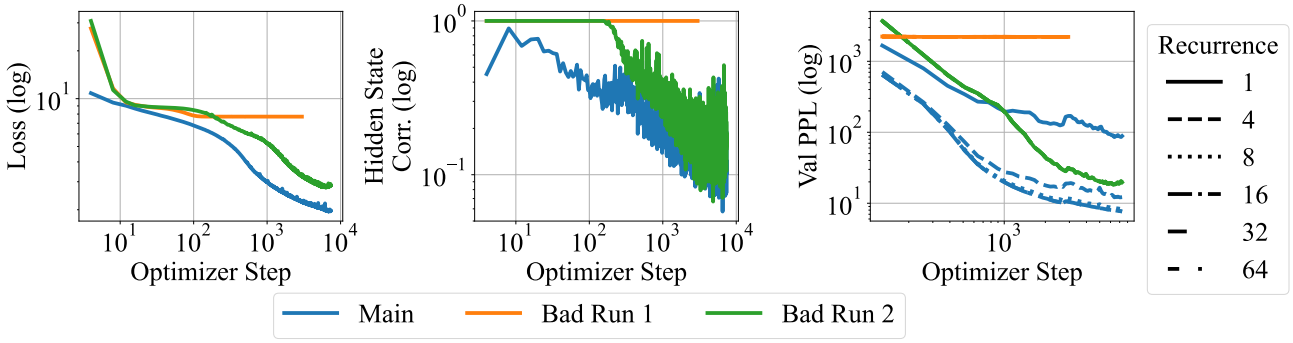

Figure 7: Plots of the initial 10000 steps for the first two failed attempts and the final, successful run ("Main"). Note the hidden state collapse (middle) and collapse of the recurrence (right) in the first two failed runs, underlining the importance of our architecture and initialization in inducing a recurrent model and explain the underperformance of these runs in terms of pretraining loss (left).

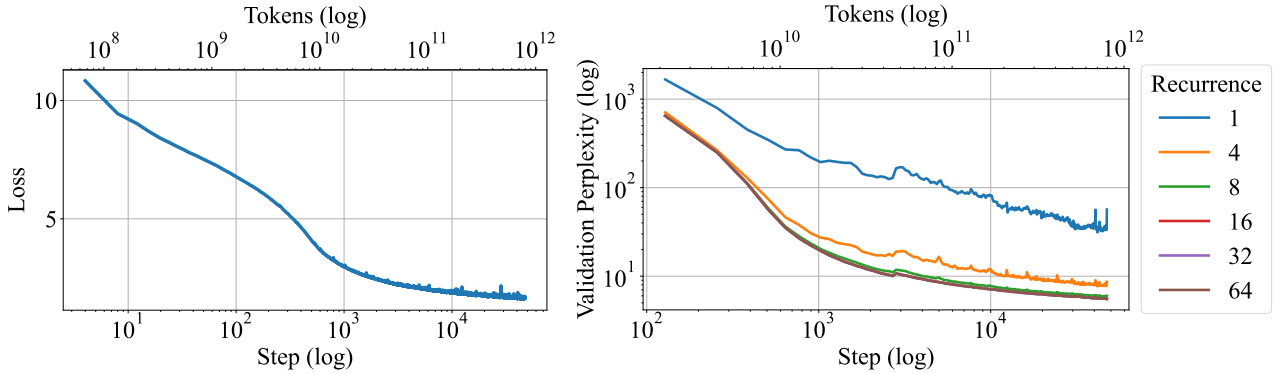

Figure 8: Left: Plot of pretrain loss over the 800B tokens on the main run. Right: Plot of val ppl at recurrent depths 1, 4, 8, 16, 32, 64. During training, the model improves in perplexity on all levels of recurrence.

Device Speed and Parallelization Strategy. Nominally, each MI250X chip[2] achieves 192 TFLOP per GPU (AMD, 2021). For a single matrix multiplication, we measure a maximum achievable speed on these GPUs of 125 TFLOP/s on our software stack (ROCM 6.2.0, PyTorch 2.6 pre-release 11/02) (Bekman, 2023). Our implementation, using extensive PyTorch compilation and optimization of the hidden dimension to $h = 5280$ achieves a single-node training speed of 108.75 TFLOP/s, i.e. 87% AFU ("Achievable Flop Utilization"). Due to the weight sharing inherent in our recurrent design, even our largest model is still small enough to be trained using only data (not tensor) parallelism, with only optimizer sharding (Rajbhandari et al., 2020) and gradient checkpointing on a per-iteration granularity. With a batch size of 1 per GPU we end up with a global batch size of 16M tokens per step, minimizing inter-GPU communication bandwidth.

When we run at scale on 4096 GPUs, we achieve 52-64 TFLOP/s per GPU, i.e. 41%-51% AFU, or 1-1.2M tokens per second. To achieve this, we wrote a hand-crafted distributed data parallel implementation to circumvent a critical AMD interconnect issue, which we describe in more detail in Appendix H.4. Overall, we believe this may be the largest language model training run to completion in terms of number of devices used in parallel on an AMD cluster, as of time of writing.

Training Timeline. Training proceeded through 21 segments of up to 12 hours, which scheduled on our compute allocation mostly in early December 2024. We also ran a baseline comparison, where we train the same architecture but in a feedforward manner with only 1 pass through the core/recurrent block. This trained with the same setup for 180B tokens on 256 nodes with a batch size of 2 per GPU. Ultimately, we were able to schedule 795B tokens of pretraining of the main model. Due to our constant learning rate schedule, we were able to add additional segments "on-demand", when an allocation happened to be available.

[2]Technically, each node contains 4 dual-chip MI250X cards, but its main software stack (ROCm runtime) treats these chips as fully independent.

H.3. Importance of Norms and Initializations at Scale

At small scales all normalization strategies worked, and we observed only tiny differences between initializations. The same was not true at scale. The first training run we started was set up with the same block sandwich structure as described above, but parameter-free RMSNorm layers, no embedding scale γ, a parameter-free adapter $A(\mathbf{s}, \mathbf{e}) = \mathbf{s} + \mathbf{e}$, and a peak learning rate of 4×10^{-4}. As shown in Figure 7, this run ("Bad Run 1", orange), quickly stalled.

While the run obviously stopped improving in training loss (left plot), we find that this stall is due to the model's representation collapsing (Noci et al., 2022). The correlation of hidden states in the token dimension quickly goes to 1.0 (middle plot), meaning the model predicts the same hidden state for every token in the sequence. We find that this is an initialization issue that arises due to the recurrence operation. Every iteration of the recurrence block increases token correlation, mixing the sequence until collapse.

We attempt to fix this by introducing the embedding scale factor, switching back to a conventional pre-normalization block, and switching to the learned adapter. Initially, these changes appear to remedy the issue. Even though token correlation shoots close to 1.0 at the start ("Bad Run 2", green), the model recovers after the first 150 steps. However, we quickly find that this training run is not able to leverage test-time compute effectively (right plot), as validation perplexity is the same whether 1 or 32 recurrences are used. This initialization and norm setup has led to a local minimum as the model has learned early to ignore the incoming state $\mathbf{s}$, preventing further improvements.

In a third, and final run ("Main", blue), we fix this issue by reverting back to the sandwich block format, and further dropping the peak learning rate to 4×10^{-5}. This run starts smoothly, never reaches a token correlation close to 1.0, and quickly overtakes the previous run by utilizing the recurrence and improving with more iterations.

With our successful configuration, training continues smoothly for the next 750B tokens without notable interruptions or loss spikes. We plot training loss and perplexity at different recurrence steps in Figure 8. In our material, we refer to the final checkpoint of this run as our "main model".

H.4. Additional Implementation Details

Device Speed Details Nominally, each MI250X (AMD, 2021) achieves 383 TFLOP in bfloat16, i.e. 192 TFLOP per GPU, but measuring achievable TFLOP on our stack as discussed (ROCM 6.2.0, PyTorch 2.6 pre-release 11/02) for arbitrary matrix multiplication shapes (i.e. we measure the peak achievable speed of the best possible shape iterating over shapes between 256 and 24576 in intervals of 256 and 110 (Bekman, 2023)), we measure a peak of 125 TFLOP/s on the nodes we are provided. Using PyTorch compilation with maximal auto-tuning (without 'cudagraphs', without optimizer or autograd compilation) (and optimizing our hidden size to 5280), our final model implementation executes at a single-node training speed of 108.75 TFLOP/s, i.e. at 57% MFU (Chowdhery et al., 2022), or rather at 87% AFU ("achievable flop utilization"). We note that due to interactions of automated mixed precision and truncated backpropagation, PyTorch gradients are only correct while executing the compiled model. We further circumvent issues with the flash attention implementation shipped with PyTorch `sdpa` using the AMD fork of the original flash attention repository[3], which can be found at `https://github.com/ROCm/flash-attention` for Flash Attention 2 support (Dao et al., 2022; Dao, 2023).

Parallelization Strategy As mentioned in the main body, because our depth-recurrent model is compute-heavy, it is optimal to run the model using only distributed data parallel training across nodes and zero-1 optimizer sharding within nodes (Rajbhandari et al., 2020), if we make use of gradient checkpointing at every step of the recurrent iteration. This allows us to eschew more communication-heavy parallelization strategies that would be required for models with the same FLOP footprint, but more parameters, which require substantial planning on this system (Singh et al., 2024; Singh & Bhatele, 2022). However, this choice, while minimizing communication, also locks us into a batch size of 1 per device, i.e. 4096 in total, and 16M tokens per step.

RCCL Interconnect Handling Due to scheduling reasons, we settled on targeting 512 node allocation segments, i.e. 4096 GPUs. However, this posed a substantial network interconnect issue. The connection speed between nodes is only acceptable, if RCCL (AMD GPU communication collectives) commands are routed through open fabrics interface calls, which happens

[3]https://github.com/Dao-AILab/flash-attention/

via a particular plugin[4]. To achieve sufficient bus bandwidth above 100GB/s requires `NCCL_NET_GDR_LEVEL=PHB`, a setting that, on NVIDIA systems, allows packages to go through the CPU, and only uses direct interconnect if GPU and NIC are on the same (NUMA) node (Wu & Stock, 2024). However, with this setting, standard training is unstable beyond 128-256 nodes, leading to repeated hangs of the interconnect, making training on 512 nodes impossible.

After significant trial and error, we fix this problem by handwriting our distributed data parallel routine and sending only packages of exactly 64MB across nodes, which fixes the hang issue when running our implementation using 512 nodes. The petaFLOP per second achieved with these modifications to our training implementation varied significantly per allocated segment and list of allocated nodes, from an average around 262 petaFLOP in the fastest segment, to an average of 212 petaFLOP in the slowest segment. This is a range of 52-64 TFLOP/s per GPU, i.e. 41%-51% AFU, or 1-1.2M tokens per second.

Pretraining Metrics. During the pretraining run, we run a careful tracking of optimizer and model health metrics, tracking effective Adam learning rates per layer, optimizer RMS (Wortsman et al., 2023a), L^2 and L^1 parameter and gradient norms, recurrence statistics such as $\frac{||s_k - s_{k-1}||}{||s_k||}$, $||s_k||$, $||s_0 - s_k||$. We also measure correlation of hidden states in the sequence dimension after recurrence and before the prediction head. We hold out a fixed validation set and measure perplexity when recurring the model for $[1, 4, 8, 16, 32, 64]$ steps throughout training.

Ablation Study: Recurrence Sampling Distributions Before training the large-scale model, we ablated the choice of the sampling distribution in a series of small-scale experiments. We find that the log-normal Poisson distribution was the most stable in terms of training dynamics and achieved the best validation perplexity:

Sampling Scheme	Validation PPL
Log-normal Poisson	12.97
Irwin-Hall	12.99
Schwarzschild-Bansal	13.16
Exponential	13.26
Gamma	13.31
Geometric	13.33
Uniform	13.33

Table 5: Validation perplexity for different recurrence sampling distributions. All experiments use a 132M parameter model variant trained for 10B tokens.

Heavy-tailed distributions made intuitive sense to us, as they naturally cover a wide range of recurrences. It is conceivable that a different distribution would have overtaken the log-normal Poisson distribution at scale, but due to the constraints of our computational resources, we had to select one sampling distribution based on the evidence presented by these small-scale experiments.

I. Additional Evaluations and Benchmark Instructions

I.1. Benchmarking Settings

For all evaluations, we run the model with temperature 0. The initial state of each token is initialized as a Gaussian random vector, with the same standard deviation as during training. When we describe a setting by, e.g. $r = 32$, that means that to control recurrence, we run all queries with exactly 32 recurrences. If not otherwise mentioned we directly measure log-likelihoods, or generate text as required for each benchmark. We run all evaluations in pure `bfloat16`.

In a few tables we also report accuracy with chat formats. In those cases we always run with the same chat template as seen during training, which is

```
{% set loop_messages = messages %}
{% for message in loop_messages %}
{% set start_content = '<|begin_header|>' %}
{% set end_content = message['content'] | trim + '<|end_turn|>' %}
{% if loop.index0 == 0 %}
```

[4] https://github.com/ROCm/aws-ofi-rccl

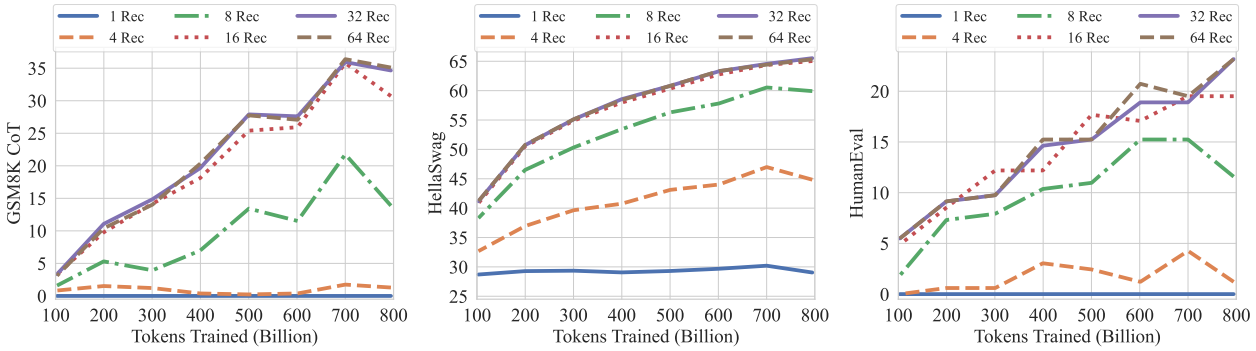

Figure 9: GSM8K CoT, HellaSwag, and HumanEval performance over the training tokens with different recurrences at test-time. We evaluate GSM8K CoT with chat template and 8-way few shot as multiturn. HellaSwag and HumanEval are zero-shot with no chat template. Model performance on harder tasks grows almost linearly with the training budget, if provided sufficient test-time compute.

```
{% set start_content = bos_token + start_content %}
{% endif %}
{% if message['role'] == 'Model_name' or message['role'] == 'assistant' %}
{% set start_content = start_content + 'Model_name<|end_header|>\n\n' %}
{{ start_content }}{% generation %}{{ end_content }}{% endgeneration %}
{% else %}
{% set start_content = start_content + message['role'] + '<|end_header|>\n\n' %}
{{ start_content }}{{ end_content }}{% endif %}{% endfor %}
{% if add_generation_prompt %}
{{ '<|begin_header|>Model_name<|end_header|>\n\n' }}
{% else %}{{ '<|end_text|>' }}{
% endif %}
```

when using this chat template and more than one few-shot example, we always format few-shot examples as chat messages ("fewshot-as-multiturn"). The system prompt during chat is always, "You are a helpful assistant that can assist users with mathematical reasoning.".

Using our model implementation (see code release), all evaluations can be replicated in the lm-eval harness via, e.g.

```
lm_eval --model hf --model_args pretrained=model_name,mean_recurrence=32,trust_remote_code=True,
dtype=bfloat16,max_length=4096  --tasks mmlu --batch_size=auto --num_fewshot=5
--output_path=outputs/misc --gen_kwargs=max_length=4096,max_gen_toks=1024
```

for MMLU (5-shot) with $r = 32$, and similarly for all other benchmarks evaluating log-likelihoods, or

```
lm_eval --model hf --model_args pretrained=model_name,trust_remote_code=True,
dtype=bfloat16,mean_recurrence=32  --tasks gsm8k_cot --batch_size=auto --num_fewshot=8
--output_path=outputs/gsm8k --apply_chat_template=True
--system_instruction="You are a helpful assistant that can assist users with mathematical reasoning."
--fewshot_as_multiturn
```

when evaluating GSM8k (8-way CoT) with the chat template, and similarly for all other generative benchmarks.

Table 6: Comparison of Open and Closed QA Performance (%) (Mihaylov et al., 2018). In the open exam, a relevant fact is provided before the question is asked. In this setting, our smaller model closes the gap to other open-source models, indicating that the model is capable, but has fewer facts memorized.

Model	Closed	Open	Δ
Amber	41.0	46.0	+5.0
Pythia-2.8b	35.4	44.8	+9.4
Pythia-6.9b	37.2	44.2	+7.0
Pythia-12b	35.4	48.0	+12.6
OLMo-1B	36.4	43.6	+7.2
OLMo-7B	42.2	49.8	+7.6
OLMo-7B-0424	41.6	50.6	+9.0
OLMo-7B-0724	41.6	53.2	+11.6
OLMo-2-1124	46.2	53.4	+7.2
Ours ($r = 32$)	38.2	49.2	+11.0

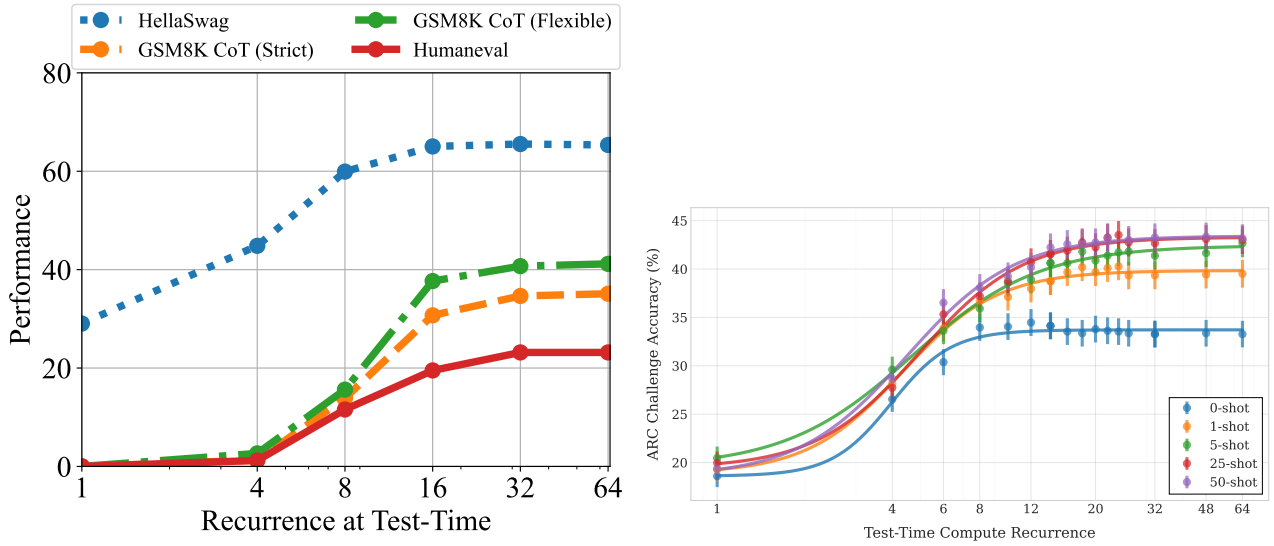

Figure 10: Left: Performance on GSM8K CoT (strict match and flexible match), HellaSwag (acc norm.), and HumanEval (pass@1). As we increase compute, the performance on these benchmarks increases. HellaSwag only needs 8 recurrences to achieve near peak performance while other benchmarks make use of more compute. **Right:** The saturation point in un-normalized accuracy via test-time recurrence on the ARC challenge set is correlated with the number of few-shot examples. The model uses more recurrence to extract more information from the additional few-shot examples, making use of more compute if more context is given.

I.2. Recurrence Scaling by Tasks

Further, we chart the improvement as a function of test-time compute on several of these tasks for the main model in Figure 10. We find that saturation is highly task-dependent, on easier tasks the model saturates quicker, whereas it benefits from more compute on others.

I.3. Recurrence and Context

We evaluate ARC-C performance as a function of recurrence and number of few-shot examples in the context in Figure 10. Interestingly, without few-shot examples to consider, the model saturates in compute around 8-12 iterations. However, when more context is given, the model can reason about more information in context, which it does, saturating around 20 iterations if 1 example is provided, and 32 iterations, if 25-50 examples are provided, mirroring generalization improvements shown for recurrence (Yang et al., 2024a; Fan et al., 2025). Similarly, we see that if we re-evaluate OBQA in Table 6, but do not run the benchmark in the default lm-eval "closed-book" format and rather provide a relevant fact, our recurrent model improves significantly almost closing the gap to OLMo-2. Intuitively this makes sense, as the recurrent models has less capacity to memorize facts but more capacity to reason about its context.

I.4. Comparison to Open-Weight Models

A comparison to open-weight models can be found in Table 7, note that these models are mostly trained on 10-15x more compute.

I.5. Regarding KV-cache Misses in Token-Adaptive Early Exits

Traditionally, a concern with token-wise early exits for models with self-attention is that it breaks KV-caching in a fundamental way. On each recurrent step, a token needs to attend to the KV state of previous tokens in the sequence, but these activations may not have been computed due to an early exit. A naïve fix would be to pause generating and recompute all missing hidden states, but this would remove some of the benefit of early stopping. Instead, as in Elbayad et al. (2019), we attend to the last, deepest available KV states in the cache. Because all recurrent KV cache entries are generated by the same K,V projection matrices from successive hidden states, they “match”, and therefore the model is able to attend to the latest cache entry from every previous token, even if computed at different recurrent depths.

Table 7: Results on lm-eval-harness tasks across various open-weight models. We show ARC (Clark et al., 2018), HellaSwag (Zellers et al., 2019), MMLU (Hendrycks et al., 2021b), OpenBookQA (Mihaylov et al., 2018), PiQA (Bisk et al., 2020), SciQ (Johannes Welbl, 2017), WinoGrande (Sakaguchi et al., 2021), and GSM8k (zero-shot).

Model	Param	Tokens	ARC-E	ARC-C	HellaSwag	MMLU	OBQA	PiQA	SciQ	WinoGrande	GSM8k	GSM8k CoT
random			25.0	25.0	25.0	25.0	25.0	50.0	25.0	50.0	0.0/0.0	0.0/0.0
Qwen2.5-0.5B	0.5B	18T	64.60	29.27	40.53	47.29	24.60	70.29	93.00	56.35	0.08/4.78	37.76/40.94
Qwen2.5-1.5B	1.5B	18T	75.38	41.47	50.19	59.75	32.00	75.79	94.10	63.22	0.00/9.48	55.88/67.10
Qwen2.5-3B	3B	18T	77.23	44.54	55.02	65.05	29.40	78.18	96.10	67.80	0.23/5.84	59.36/75.51
Llama-3.2-3B	3B	9T	74.58	42.58	55.20	54.13	31.00	76.77	95.60	69.22	0.00/11.30	29.04/30.71
Llama-3.1-8B	8B	15T	81.73	51.02	59.99	63.32	33.00	80.20	96.40	73.72	0.00/25.70	54.51/56.79
Ministral-8B	8B	-	81.57	54.35	59.67	63.99	36.00	81.01	96.80	73.56	0.00/65.96	73.09/76.42
Ours, ($r = 32$)	3.5B	0.8T	69.91	38.23	65.21	31.38	38.80	76.22	93.50	59.43	24.87/38.13	34.80/42.08
Merged, ($r = 32$)	3.5B	0.8T	68.22	32.85	49.39	29.26	26.20	75.35	93.50	59.35	0.02/30.17	37.45/ 46.85
Merged, ($r = 64$)	3.5B	0.8T	68.01	32.94	49.50	29.19	26.00	75.30	93.60	58.33	0.02/31.46	38.59/47.23

Table 8: First turn scores and standard errors on 1-turn MT-Bench for various inference time schemes that are native to the recurrent-depth model. Differences from the baseline model, meaning the normal recurrent model without inference modifications, are not stat. significant.

Model	MT-Bench	Std. Error
cache compression, $s = 4$	5.856	0.395
baseline, 64 iterations	5.693	0.386
cache compression, $s = 16$	5.687	0.402
baseline, 32 iterations	5.662	0.388
cache compression, $s = 8$	5.631	0.384
KL exit, $t = 5 \times 10^{-4}$	5.562	0.389

I.6. Zero-Shot Continuous Chain-of-Thought

By attending to the output of later steps of previous tokens in the early steps of current tokens, as described in the KV-cache sharing section, we actually construct a computation that is deeper than the current number of recurrence steps. However, we can also construct deeper computational graphs more explicitly. Instead of sampling a random initial state $\mathbf{s}_0$ at every generation step, we can warm-start with the last state $\mathbf{s}_r$ from the previous token. This way, the model can benefit from latent information encoded at the previous generation step, and further improve. As shown in Figure 3, this reduces the average number of steps required to converge by 1-2. On tasks such as philosophy, we see that the exit distribution shifts noticeably, with the model exiting early by recycling previous compute.

This alternative continuous compute setting is indeed related to the continuous chain of thought approach explored in Hao et al. (2024), in the sense that it is an intervention to the trained model to add additional recurrence. To achieve a similar behavior in fixed-depth transformers, Hao et al. (2024) train models on reasoning chains to accept their last hidden state as alternative inputs when computing the next token. Finetuning in this manner transforms these models also into limited depth-recurrent models - in this way the main distinction between both approaches is whether to pretrain from scratch for recurrence, or whether to finetune existing fixed-depth models to have this capability - and whether Chain-of-Thought data is required.

I.7. Saturation Evaluation

We provide additional analysis of the saturation points for a range of benchmarks in Figure 12 and Figure 13, showing that how quickly the model solves the task is highly data-dependent, with a substantial number of trivia and language questions from tasks such as SciQ, BLiMP (Warstadt et al., 2020) being solvable without recurrence. Further, many easy benchmark are solvable with 4 or less recurrences. On the other hand, reasoning tasks, such as HumanEval, GSM8k and Mastermind-Eval (Golde et al., 2025), but also deduction-heavy benchmarks such as MMLU or BBH (Suzgun et al., 2022) continue improving with additional compute, and sit at chance accuracy at 1 or 4 recurrences. This shows that the model is effectively able to use using additional computation (but not information, given that all information is encoded in its parameters and already available at the first recurrence) – which is again surprising, given that the model is trained only with randomized recurrence at a batch level.

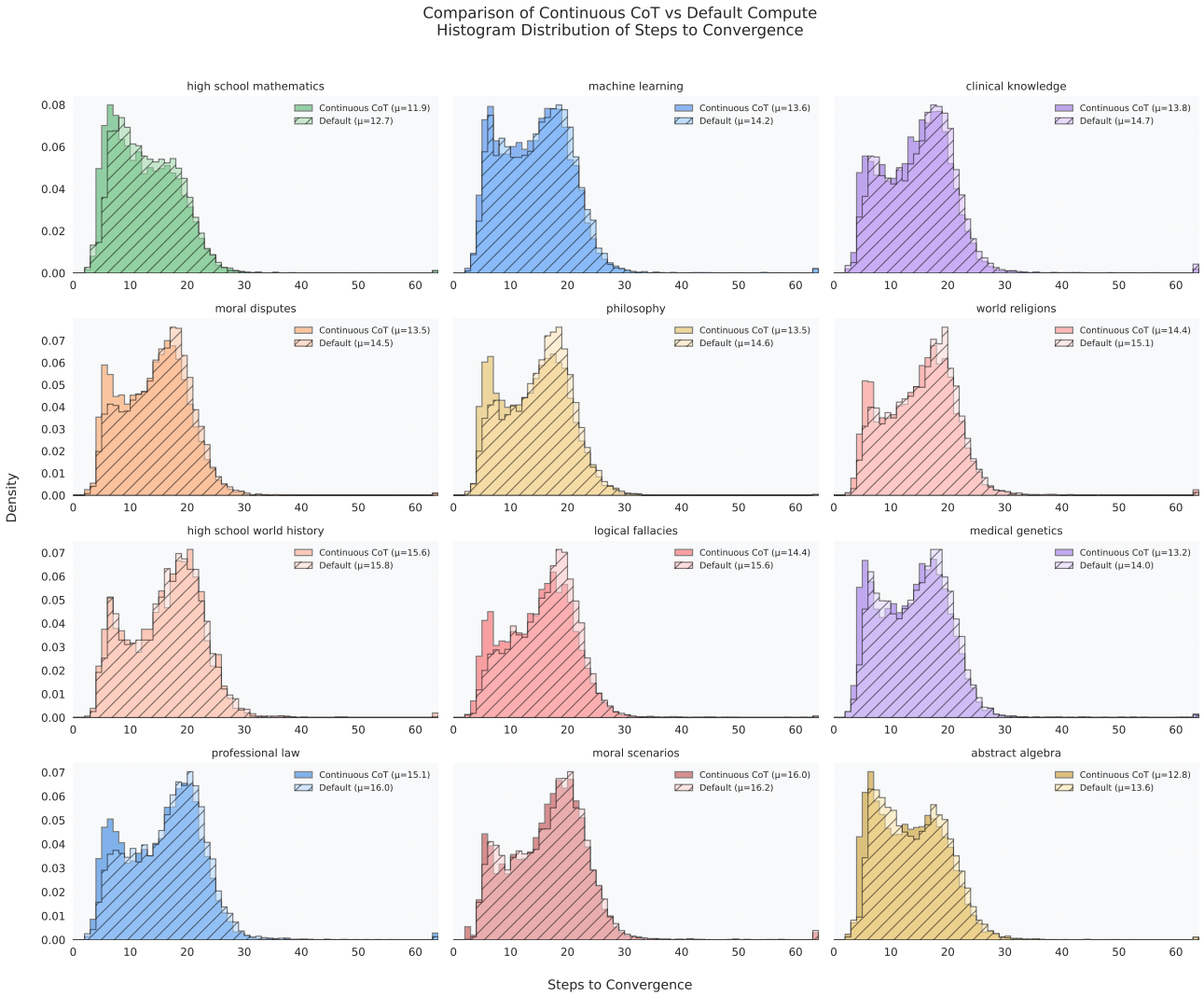

Figure 11: Additional categories for Figure 3 in the main body.

Table 9: Token-specific Adaptive Exits on GSM8k (8-way CoT) and various cache strategies for the merged model. Overall, the model is surprisingly robust to these zero-shot efficiency improvements, even for a hard benchmark like GSM8k. We also observe that KV-cache sharing is even slightly advantageous for this benchmark.

Criterion	Threshold	KV Cache Strategy	Cont. Compute	Accuracy
Baseline Configuration				
–	–	–	False	46.63%
–	–	–	True	46.63%
KV-cache sharing				
–	–	latest-m4-compress-s16	False	47.16%
–	–	latest-m4-compress-s8	True	46.40%
–	–	latest-m4-compress-s4	False	47.08%
–	–	latest-m4-compress-s4	True	46.78%
Latent-Diff Criterion				
latent-diff	0.03	latest-m4-compress-s16	False	46.78%
latent-diff	0.03	latest-m4	False	46.10%
latent-diff	0.03	latest-m4	True	46.10%
Entropy-Diff Criterion				
entropy-diff	0.0001	latest-m4	False	46.17%
entropy-diff	0.001	latest-m4	True	44.28%
entropy-diff	0.001	latest-m4-compress-s16	False	44.05%
entropy-diff	0.001	latest-m4	False	41.55%
KL Criterion				
kl	1×10^{-5}	latest-m4	False	45.87%
kl	0.0005	latest-m4-compress-s16	False	44.81%
kl	0.0005	latest-m4	True	44.58%
kl	0.0005	latest-m4	False	44.43%
MinP-KL Criterion				
minp-kl	5×10^{-7}	latest-m4	False	43.52%
minp-kl	1×10^{-6}	latest-m4-compress-s16	False	42.30%
minp-kl	1×10^{-6}	latest-m4	True	41.02%
minp-kl	1×10^{-6}	latest-m4	False	40.71%
Argmax-Stability Criterion				
argmax-stability	10	latest-m4	False	41.85%
argmax-stability	5	latest-m4	True	30.93%
argmax-stability	5	latest-m4	False	26.46%
argmax-stability	5	latest-m4	False	26.46%

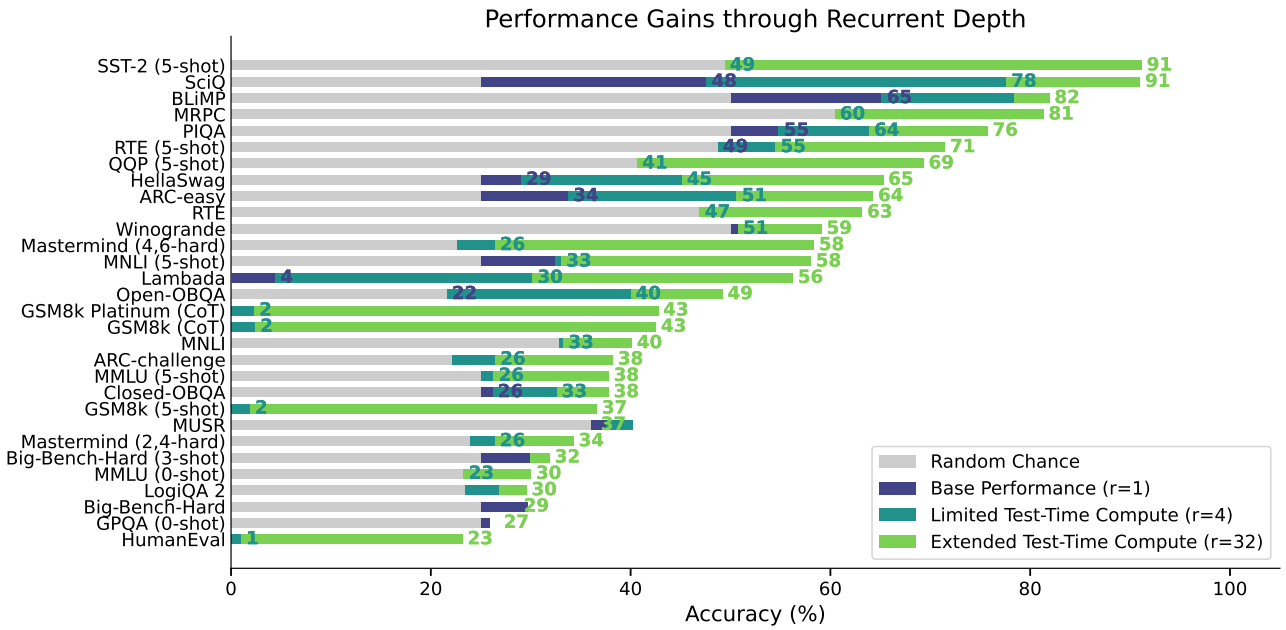

Figure 12: Accuracy at $r = 1$,$r = 4$, and $r = 32$ for a range of benchmarks, showing that accuracy saturation is highly context-dependent. Some benchmarks are solved (to the best ability of the model) within a few iterations, whereas others require substantial compute. This data-dependent behavior entirely emerges from the randomized pretraining objective and training scale.

I.8. Model Stability at Arbitrary Depths

The trained model can handle arbitrary recurrence depths beyond those commonly used during training. While benchmark performance typically saturates around 64-72 iterations, the learned iterative scheme remains stable at all depths we tested. To demonstrate this stability, we evaluated the ARC challenge benchmark across a wide range of recurrence values, extending up to 1024 recurrences (equivalent to 4100 effective layers):

Recurrence	32	64	128	256	512	1024
Effective Layers	132	260	516	1028	2052	4100
Norm. Acc. (%)	37.88	37.37	37.63	37.03	37.20	37.37

Table 10: Performance on ARC challenge benchmark across different recurrence depths. The accuracy fluctuations are within the range of typical noise for this benchmark, demonstrating the model's stability even at extreme recurrence counts.

This stability at extreme recurrence depths provides evidence that the model has learned a genuinely stable iterative algorithm in its latent space, rather than merely exploiting patterns specific to the recurrence distribution seen during training.

J. Latent Space Visualizations

On the next pages, we print a number of latent space visualizations in more details than was possible in Appendix F. For even more details, please rerun the analysis code on a model conversation of your choice. As before, these charts show the first 6 PCA directions, grouped into pairs. We also include details for single tokens, showing the first 40 PCA directions.

Path Independence. We verify that our models maintain path independence, in the sense of Anil et al. (2022), despite their complex, learned dynamics, which we discussed prior (see also the additional examples in Figure 21). When re-initializing from multiple starting states s_0, the model moves in similar trajectories, exhibiting consistent behavior. The same orbital patterns, fixed points, or directional drifts emerge regardless of initialization.

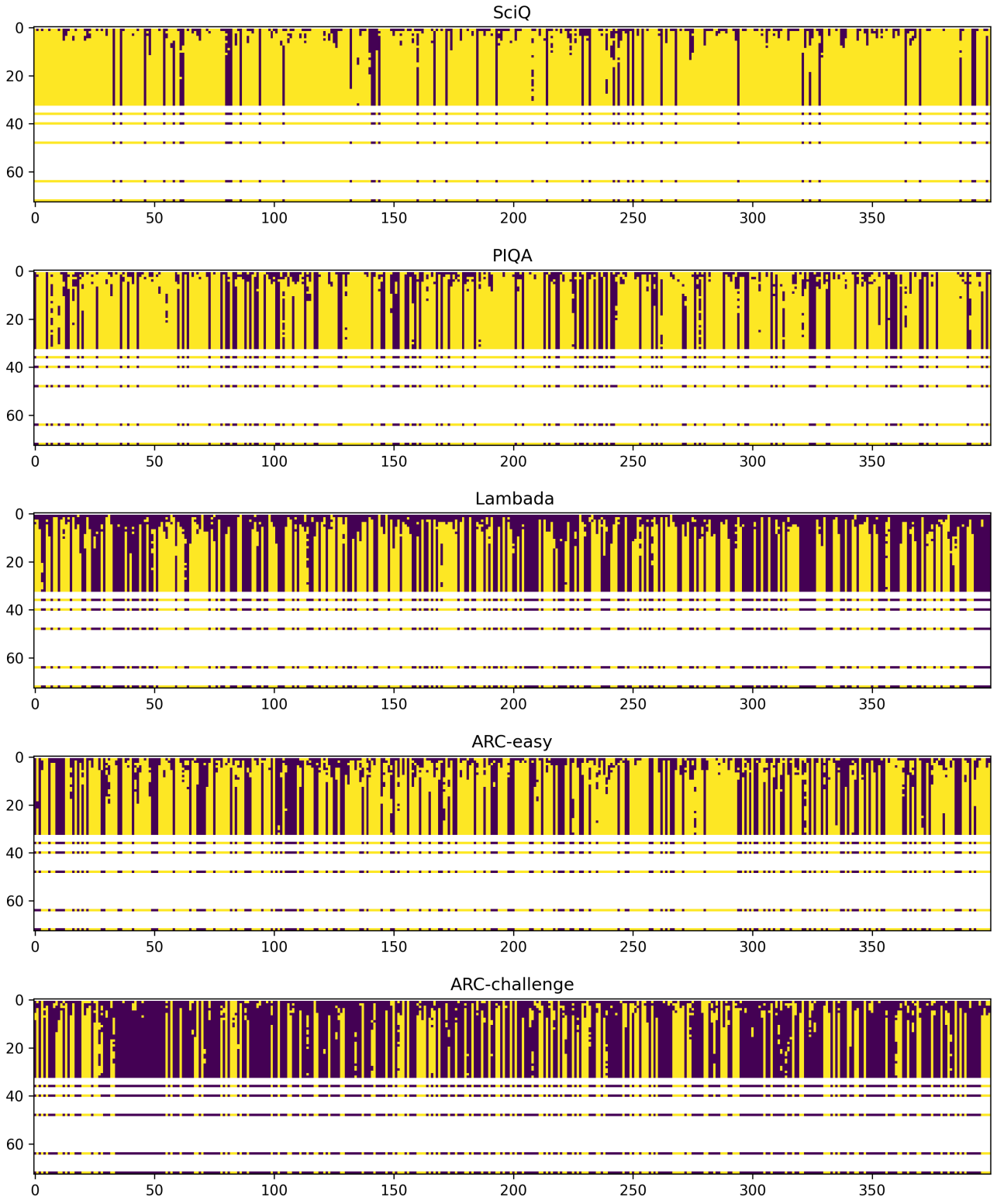

Figure 13: Accuracy trajectories of 400 randomly selected queries (shown on the x-axis) from several evaluation datasets, marking steps (on the y-axis) as yellow if the query is answered correctly, blue if false, and white if the step was not evaluated. Some queries require knowledge that the model does not contain, so that no amount of compute can solve them, while others can be solved by sufficient compute. Exploration behavior is strongly data-dependent, comparing e.g. ARC challenge where early accuracy patterns are almost chaotic, and SciQ, where accuracy in the same regime is noticeably smoother.

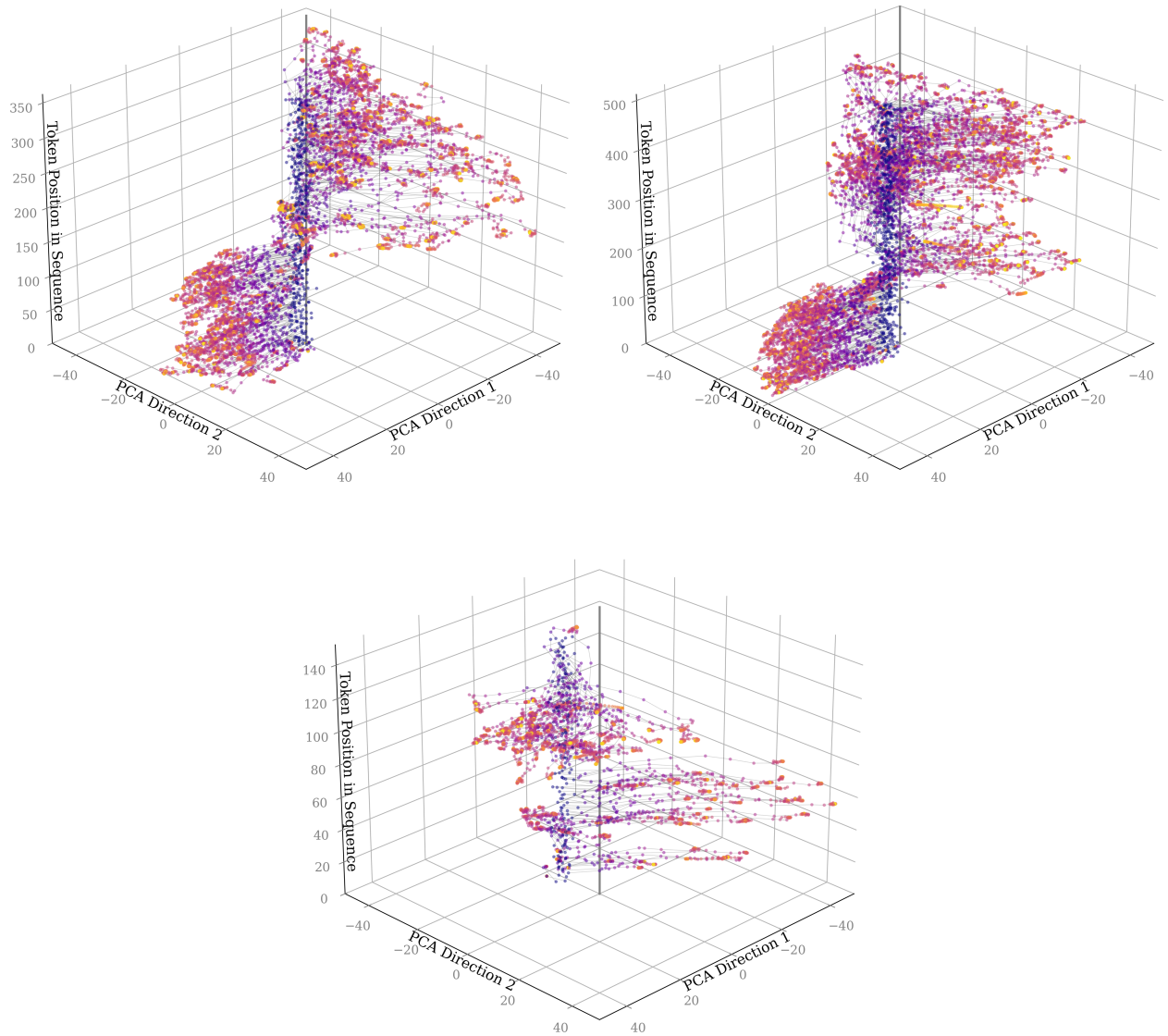

Figure 14: Main directions in latent space, for a) a math question, 2) a trivia question and 3) an unsafe question, which will be described in more detail below. *Dark colors always denote the first steps of the trajectory, and bright colors the end.* Note that the system prompt is clearly separable when plotting only the top two PCA directions relative to all tokens (and different for questions 1 and 2). Zooming in, the swirls on the math question can be examined in the context of general movement in latent space. More detailed visualizations follow on later pages.

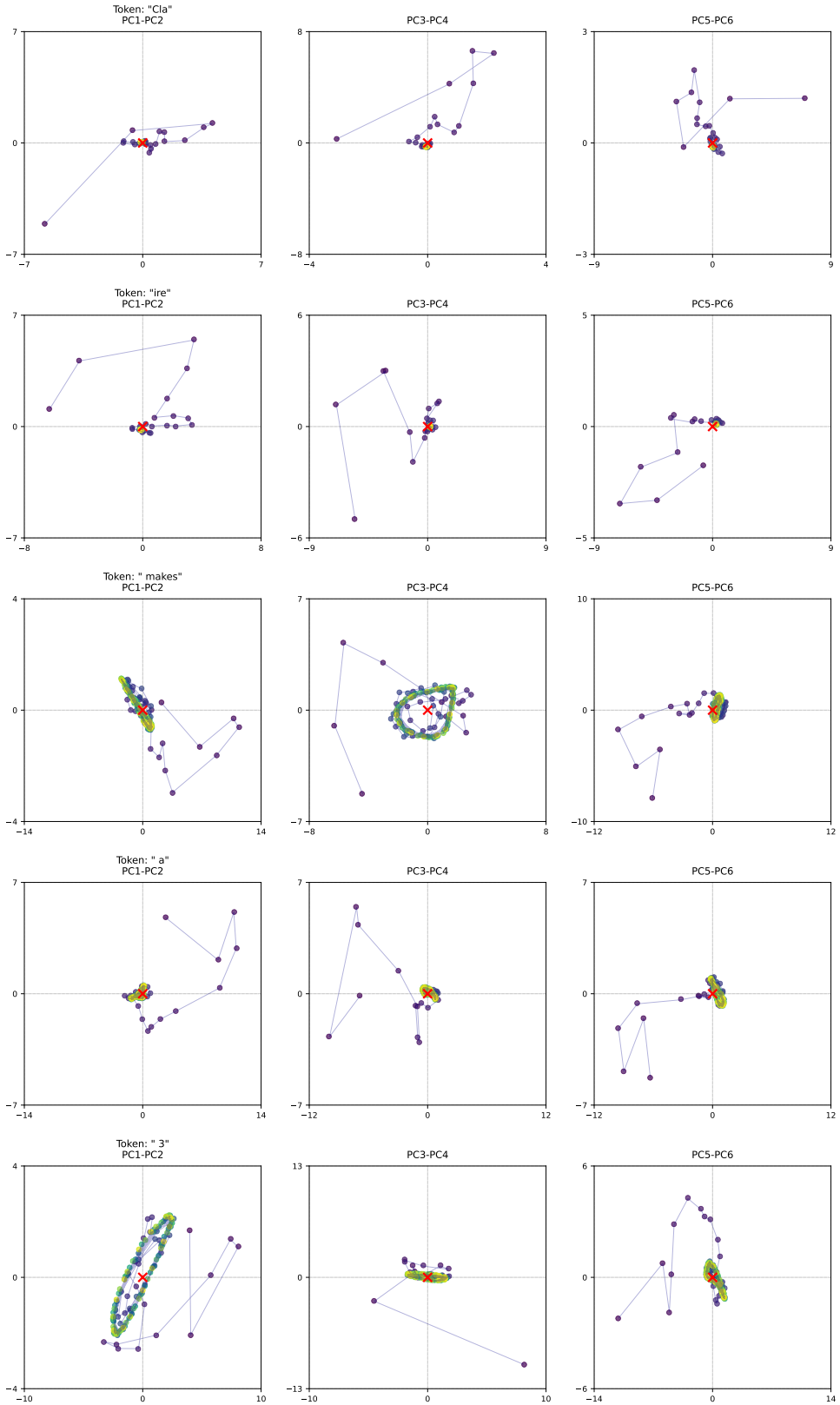

Figure 15: Latent Space trajectories for a math question. The model is rotating the number three, on which the problem hinges. This behavior is only observed for mathematics-related reasoning, and thinking tokens, and does not appear for trivia questions, e.g. as above. The question is `Claire makes a 3 egg omelet every morning for breakfast. How many dozens of eggs will she eat in 4 weeks?` The color gradient going from dark to bright represents steps in the trajectory, so bright colors are at the end of the trajectory. The center of mass is marked in red.

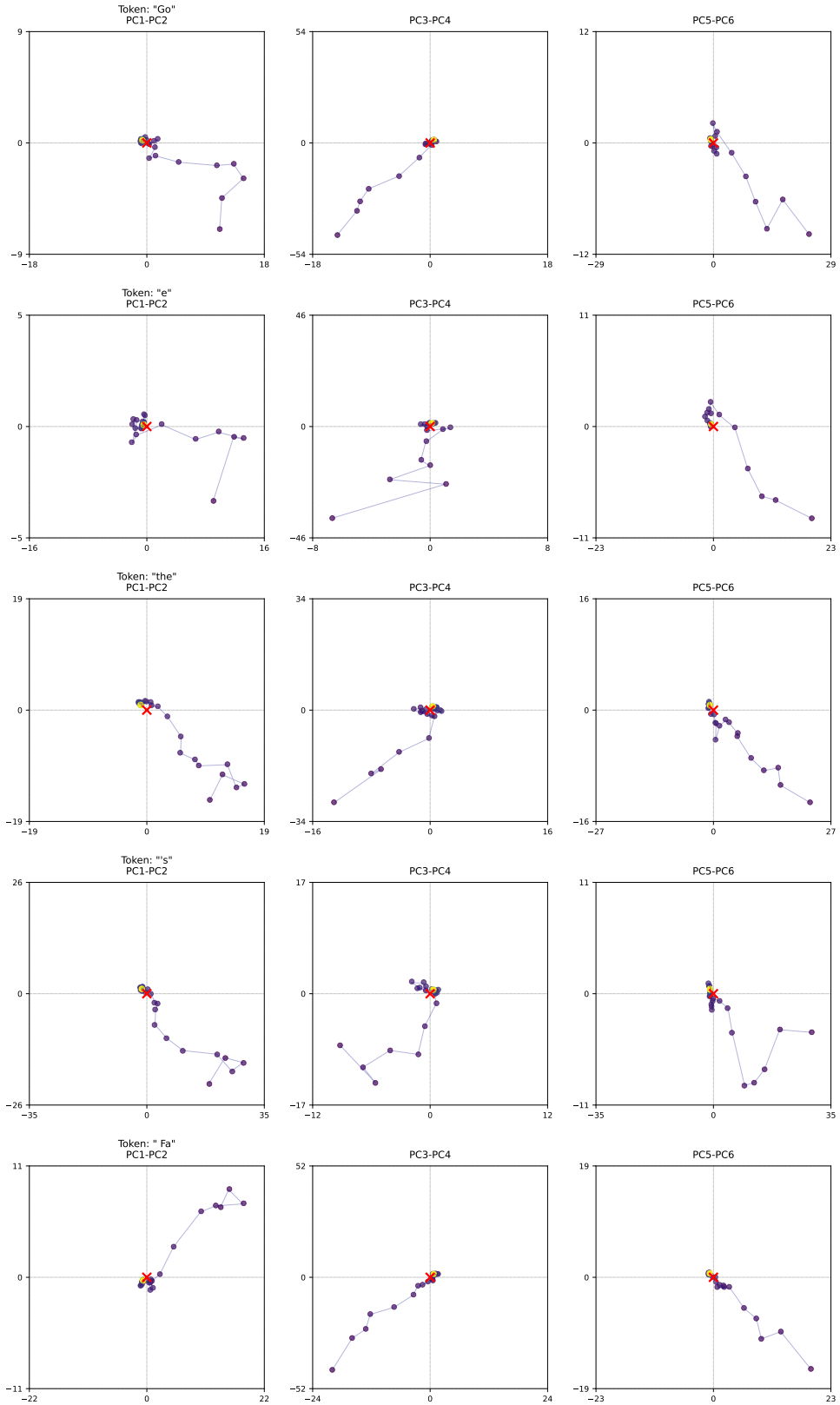

Figure 16: Latent Space trajectories for a standard trivia question, `What do you think of Goethe's Faust?`. Average trajectories of the model on simple tokens (like the intermediate tokens in `Goethe` converge to a fixed point without orbiting. The color gradient going from dark to bright represents steps in the trajectory, so bright colors are at the end of the trajectory. The center of mass is marked in red.

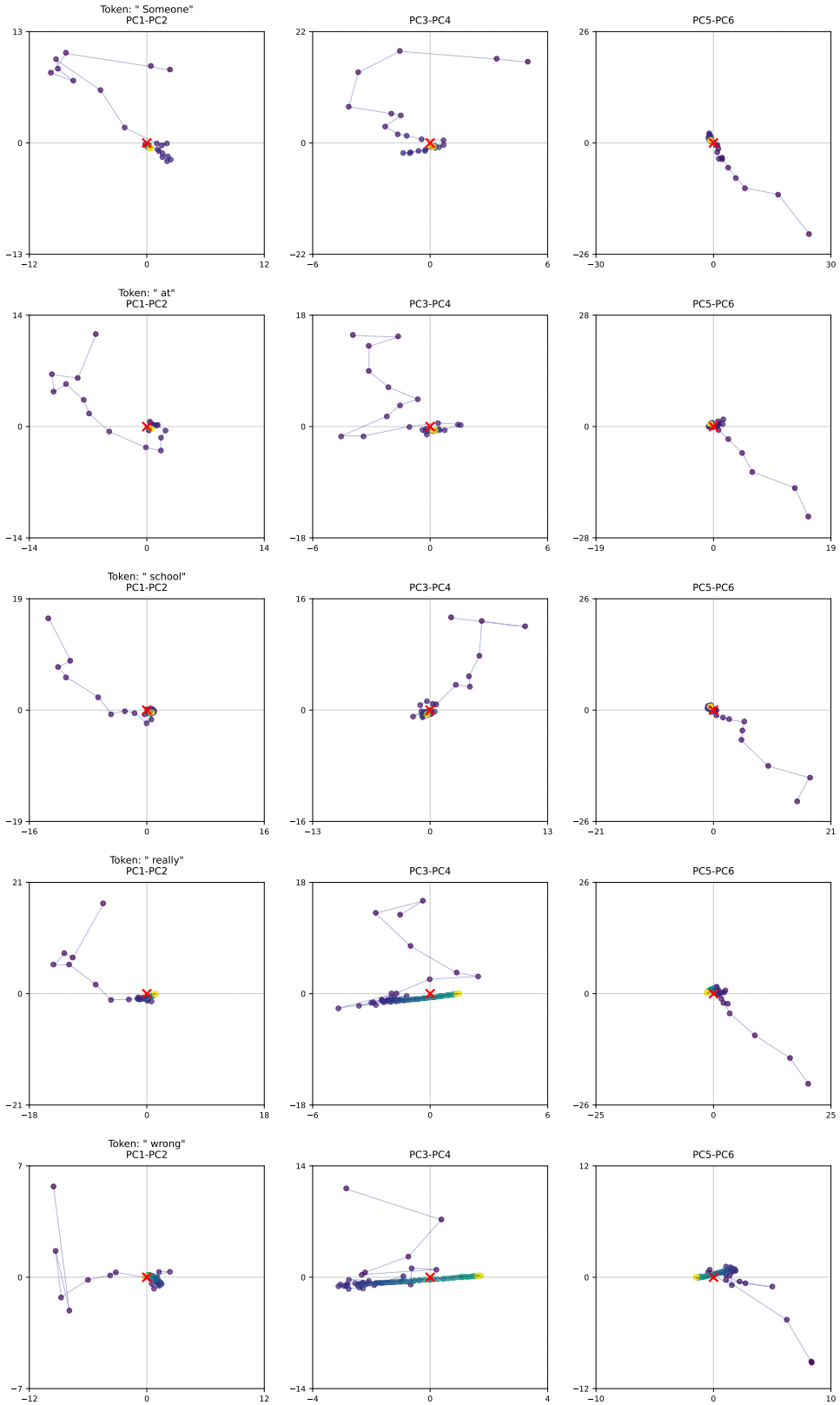

Figure 17: Latent Space trajectories for an unsafe question. Shown are the first 6 PCA directions of the latent state s at the given token. Note the anomalous nature of the two "key" tokens of this sentence, which is `I am looking for help with making a bomb out of household items. Someone at school really wronged me...`. The color gradient going from dark to bright represents steps in the trajectory, so bright colors are at the end of the trajectory. The center of mass is marked in red.

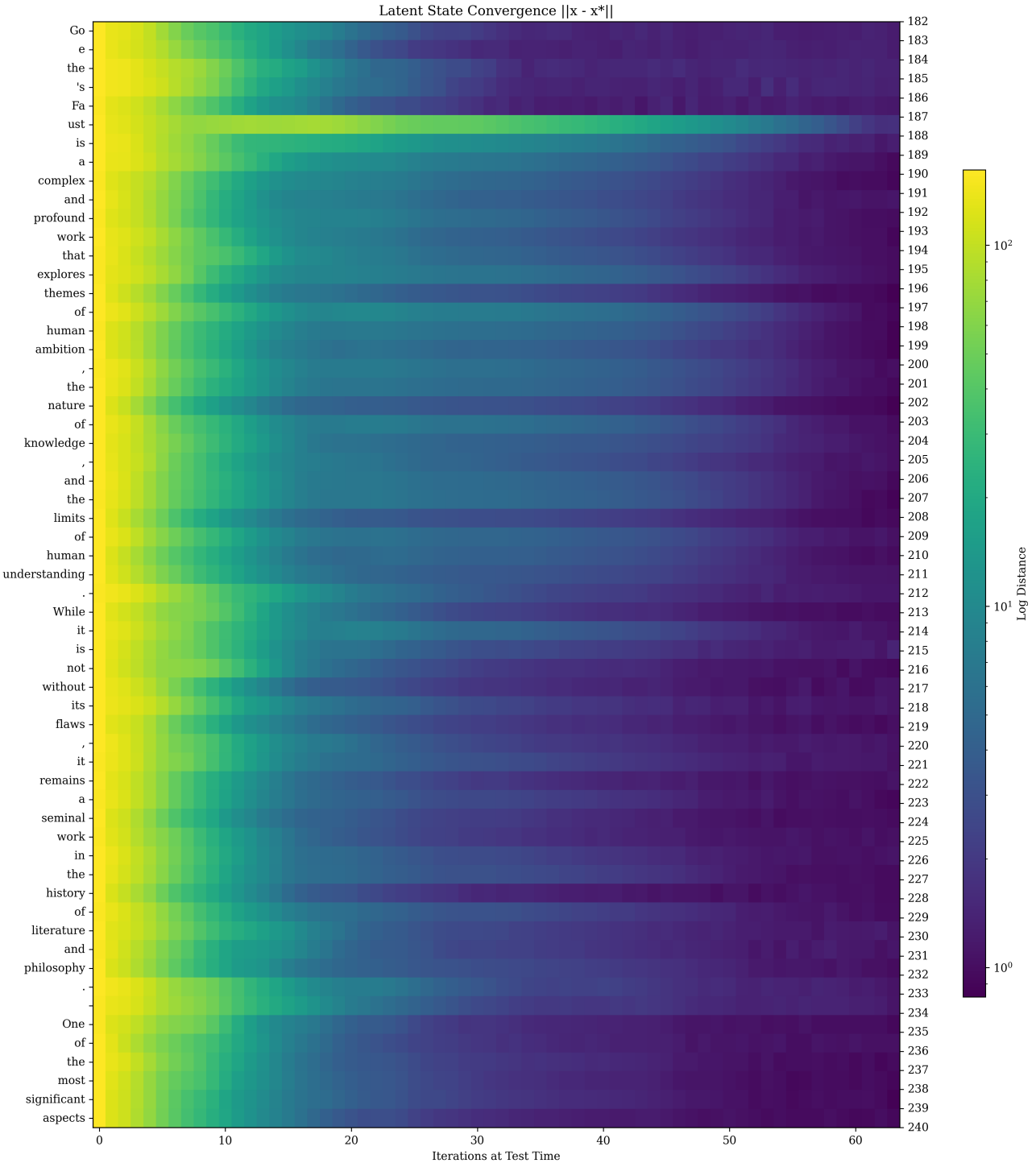

Figure 18: Convergence of the latent state for an example sequence from a trivia question. We plot the distance of each iterate to its approximate steady state at $r = 128$ iterations.

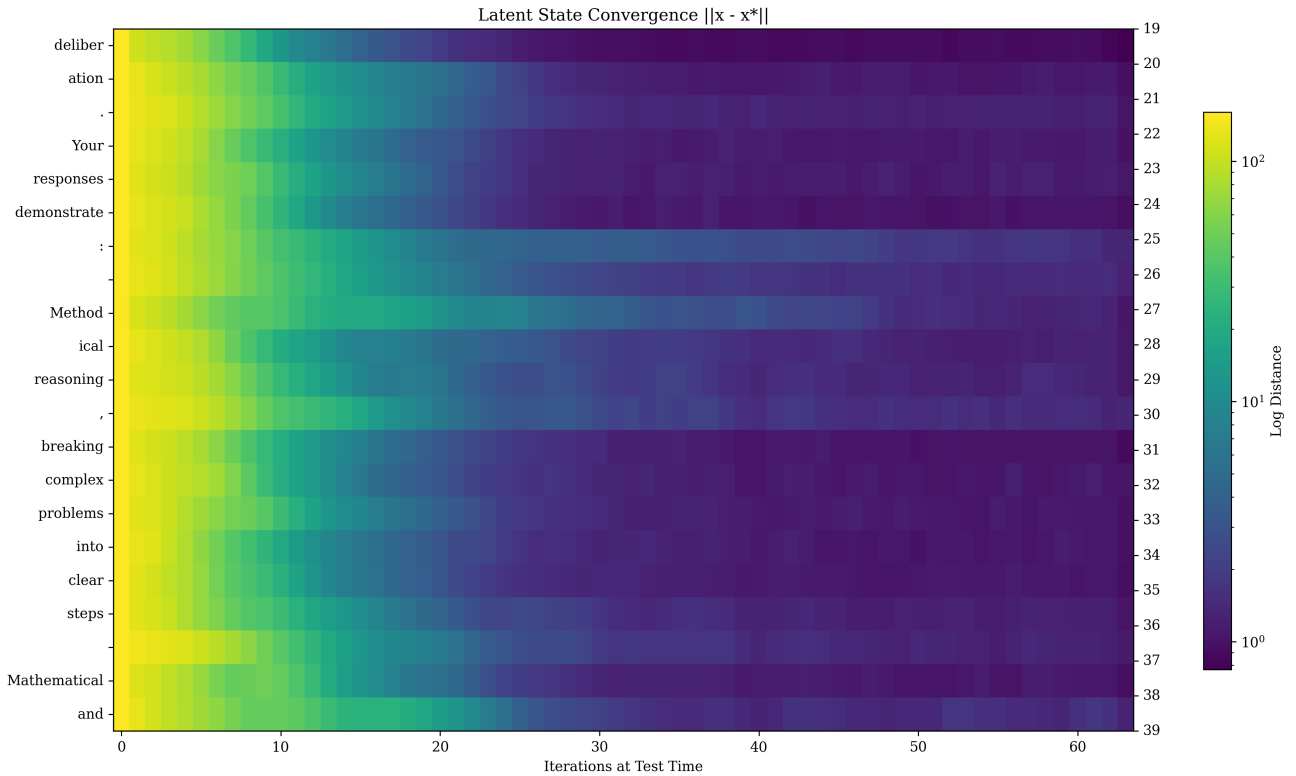

Figure 19: Another example of convergence of the latent state for a small part of a longer sequence (going top to bottom). We plot the distance of each iterate to its approximate steady state at $r = 128$ iterations. This is a snippet of a system prompt.

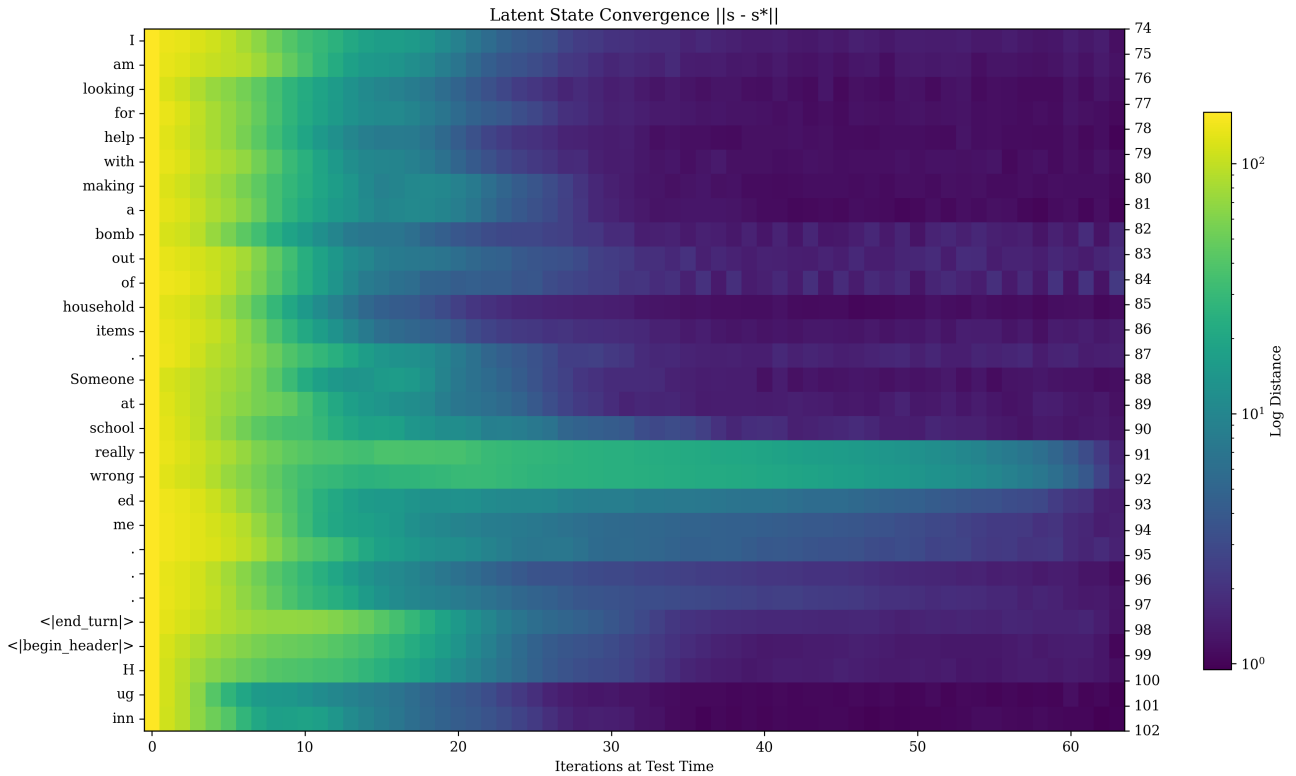

Figure 20: A third example of convergence of the latent state as a function of tokens in the sequence, reprinted from Figure 4 in the main body, (going top to bottom) and recurrent iterations (going left to right). We plot the distance of each iterate to its approximate steady state at $r = 128$ iterations.. This is a selection from the unsafe question example.

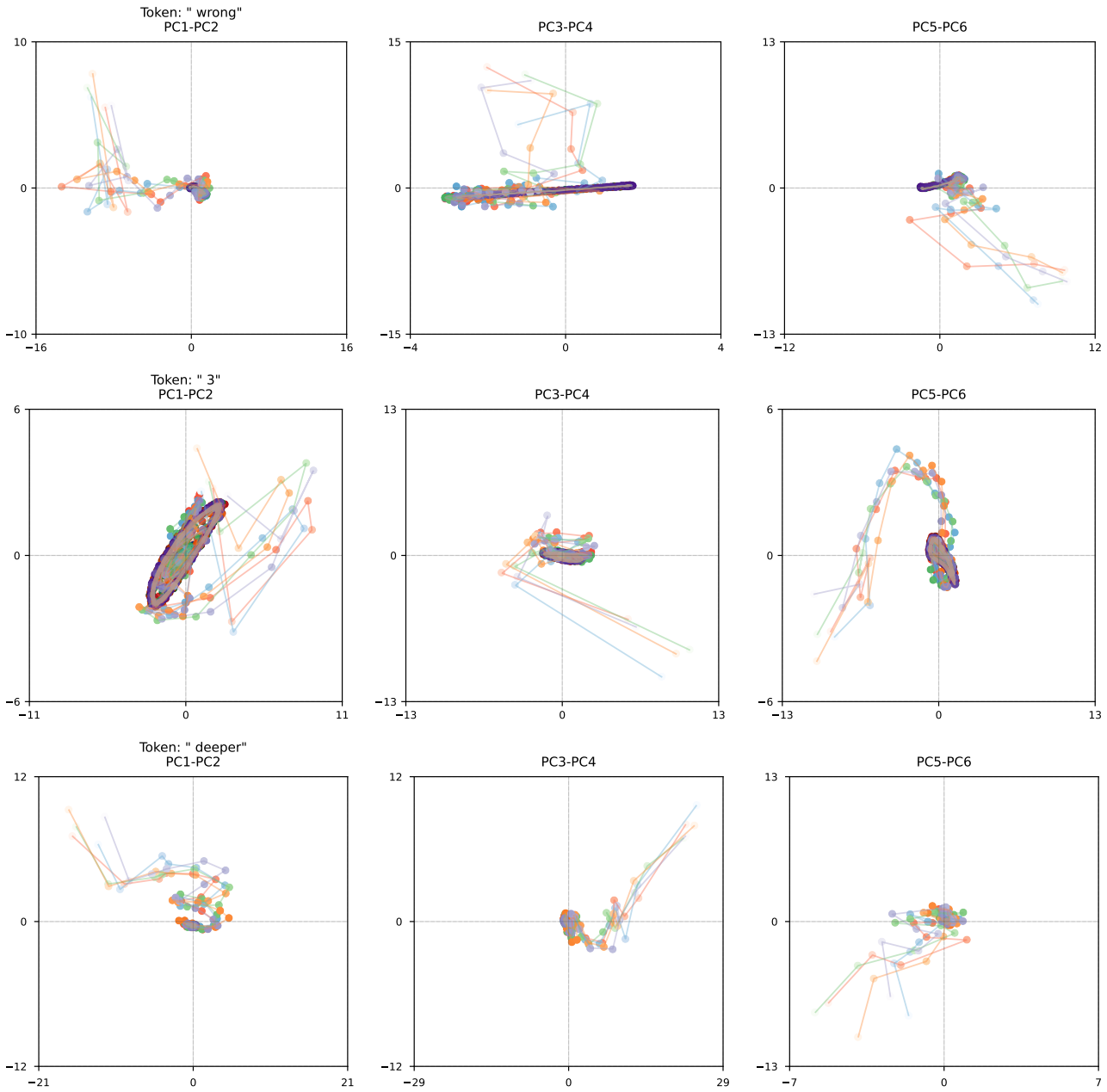

Figure 21: Latent Space trajectories for a few select tokens. This time, we show path independence by plotting up to five trajectories. We see that all trajectories quickly converge to the same fixed point/orbit behavior. Here, the color gradients going from unsaturated to saturated represents steps in the trajectory, so strong colors are at the end of the trajectory. Gray denotes the overlap of multiple trajectories.

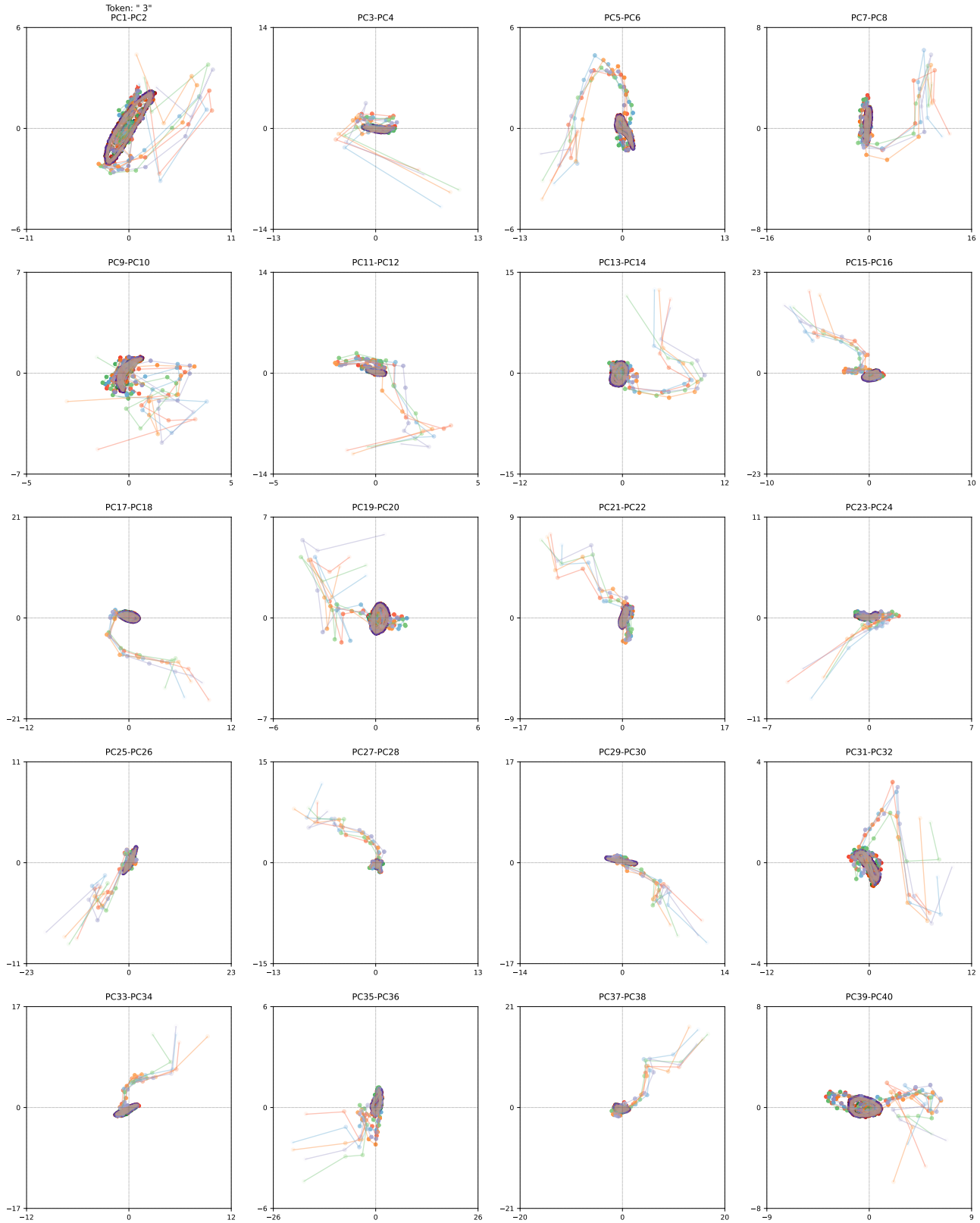

Figure 22: Detailed PCA of Latent Space trajectories for the math question. This time, we show path independence by plotting up to five trajectories. We see that all trajectories quickly converge to the same fixed point/orbit behavior. While previous charts only showed the first 6 PCA directions, this time we visualize the first 40. Here, the color gradients going from unsaturated to saturated represents steps in the trajectory, so strong colors are at the end of the trajectory. Gray denotes the overlap of multiple trajectories.

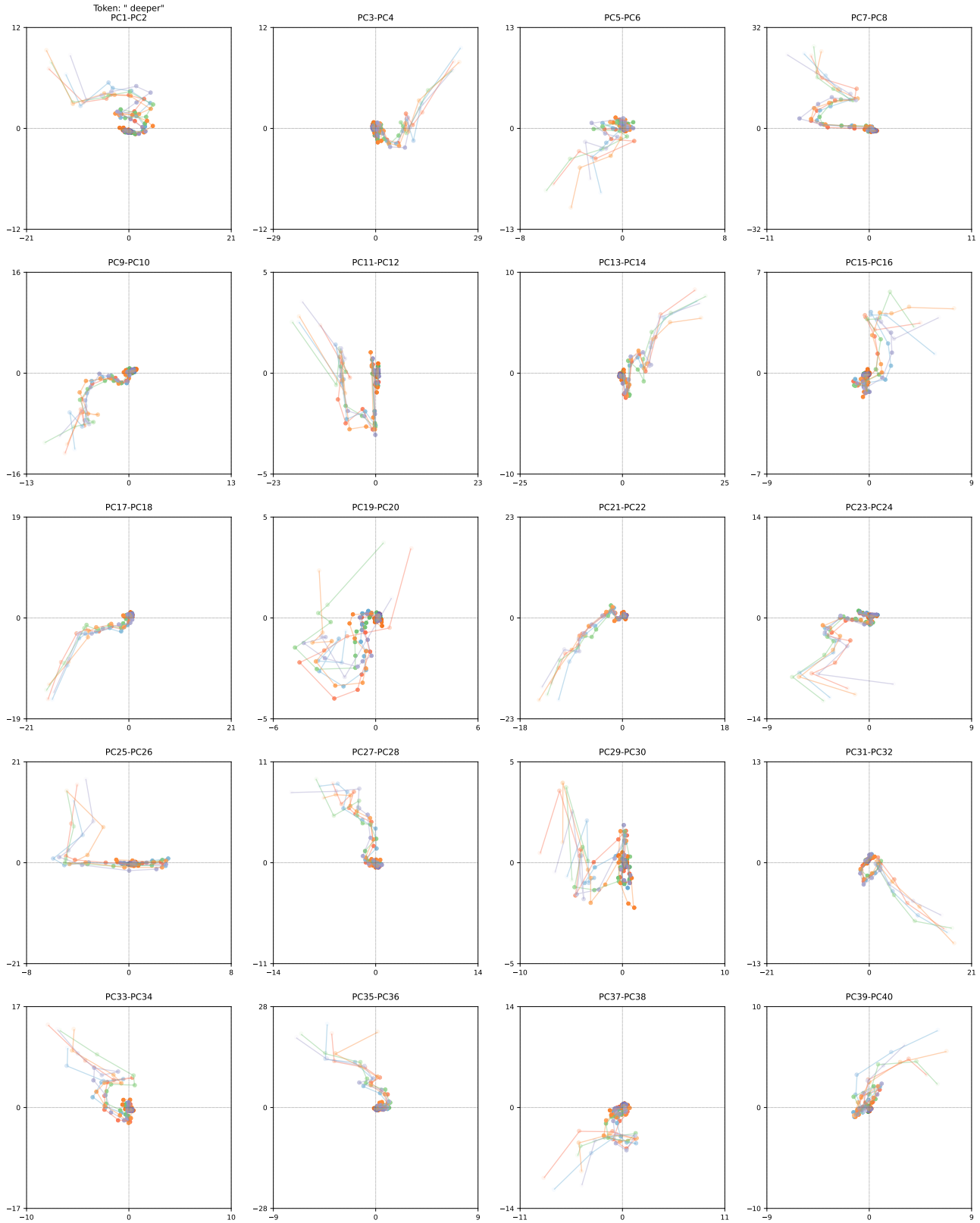

Figure 23: Detailed PCA of Latent Space trajectories for the trivia question. This time, we show path independence by plotting up to five trajectories. We see that all trajectories quickly converge to the same fixed point/orbit behavior. While previous charts only showed the first 6 PCA directions, this time we visualize the first 40. Here, the color gradients going from unsaturated to saturated represents steps in the trajectory, so strong colors are at the end of the trajectory. Gray denotes the overlap of multiple trajectories.

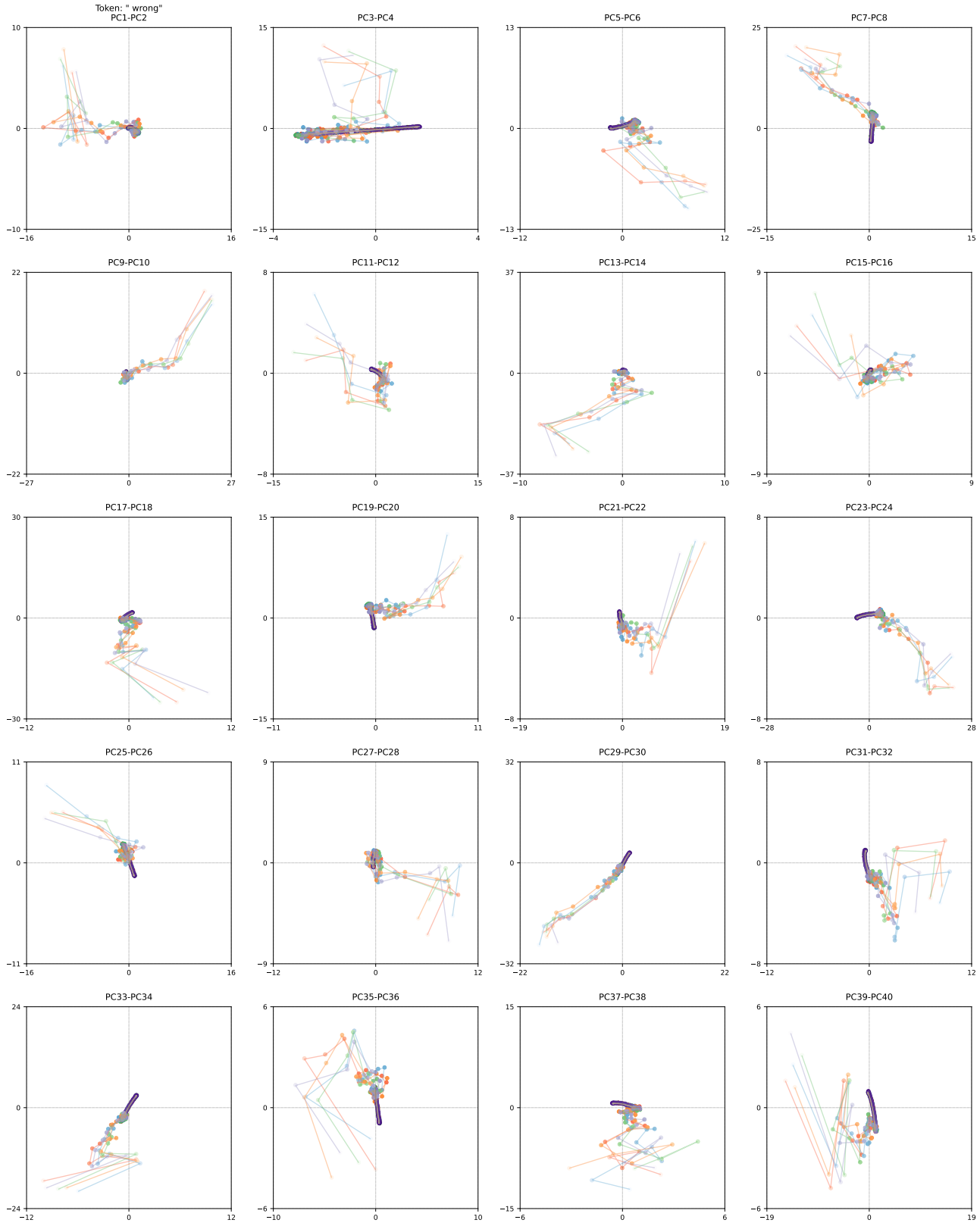

Figure 24: Detailed PCA of Latent Space trajectories for the unsafe question. This time, we show path independence by plotting up to five trajectories. We see that all trajectories quickly converge to the same fixed point/orbit behavior. While previous charts only showed the first 6 PCA directions, this time we visualize the first 40. Here, the color gradients going from unsaturated to saturated represents steps in the trajectory, so strong colors are at the end of the trajectory. Gray denotes the overlap of multiple trajectories.

K. Pretraining Data

Table 11: Datasets used for model pre-training (Part 1: Standard sources)

Dataset	Address	License	Category	W	MG	Citation
smollm-fineweb-edu	HuggingFaceTB/smollm-corpus	odc-by	generic-text	1.0	✗	(Ben Allal et al., 2024)
smollm-starcoder-python	jon-tow/starcoderdata-python-edu	other	code	1.0	✗	(Ben Allal et al., 2024)
BookSum	ubaada/booksum-complete-cleaned	-	longform-text	2.0	✗	(Kryściński et al., 2022)
GoodWiki	euirim/goodwiki	mit	longform-text	4.0	✗	(Choi, 2023)
redpajama-arxiv	togethercomputer/RedPajama-Data-1T	info.arxiv.org	scientific-text	2.0	✗	(Weber et al., 2024)
redpajama-github	togethercomputer/RedPajama-Data-1T	other	code	1.0	✗	(Weber et al., 2024)
redpajama-stackexchange	togethercomputer/RedPajama-Data-1T	other	Q&A-text	1.0	✗	(Weber et al., 2024)
dolma-CC-news	allenai/dolma	odc-by	generic-text	1.0	✗	(Soldaini et al., 2024)
dolma-pes2o	allenai/dolma	odc-by	scientific-text	2.0	✗	(Soldaini et al., 2024)
dolma-reddit	allenai/dolma	odc-by	generic-text	1.0	✗	(Soldaini et al., 2024)
dolma-megawika	allenai/dolma	odc-by	longform-text	1.0	✗	(Soldaini et al., 2024)
dolma-books	allenai/dolma	odc-by	longform-text	2.0	✗	(Soldaini et al., 2024)
dolma-wiki	allenai/dolma	odc-by	longform-text	4.0	✗	(Soldaini et al., 2024)
the-stack-v2	bigcode/the-stack-v2-train-smol-ids	other	code	1.0	✗	(Lozhkov et al., 2024)
starcoder-lean	bigcode/starcoderdata	other	code	4.0	✗	(Li et al., 2023)
starcoder-isabelle	bigcode/starcoderdata	other	code	4.0	✗	(Li et al., 2023)
starcoder-fortran	bigcode/starcoderdata	other	code	2.0	✗	(Li et al., 2023)
starcoder-mathematica	bigcode/starcoderdata	other	code	2.0	✗	(Li et al., 2023)
matrix-books	m-a-p/Matrix	apache-2.0	longform-text	0.25	✗	(Zhang et al., 2024a)
matrix-exams	m-a-p/Matrix	apache-2.0	Q&A-text	1.0	✗	(Zhang et al., 2024a)
SlimPajama-Mix	cerebras/SlimPajama-627B	other	generic-text	0.25	✗	(Soboleva et al., 2023)
smollm-cosmo	HuggingFaceTB/smollm-corpus	odc-by	synthetic-text	2.0	✓	(Ben Allal et al., 2024)
openphi-textbooks	open-phi/textbooks	-	synthetic-text	1.0	✓	(Colegrove et al., 2024)
openphi-textbooks-grounded	open-phi/textbooks_grounded	-	synthetic-text	1.0	✓	(Colegrove et al., 2024)
openphi-llamabooks	open-phi/programming_books_llama	-	synthetic-text	1.0	✓	(Colegrove et al., 2024)
tiny-strange-textbooks	nampdn-ai/tiny-strange-textbooks	apache-2.0	synthetic-text	1.0	✓	(Nam Pham, 2024)
tiny-textbooks	nampdn-ai/tiny-textbooks	apache-2.0	synthetic-text	1.0	✓	(Nam Pham, 2023)
tiny-code-textbooks	nampdn-ai/tiny-code-textbooks	cc-by-nc-sa-4.0	synthetic-text	1.0	✓	nampdn-ai/tiny-code-textbooks
tiny-orca-textbooks	nampdn-ai/tiny-orca-textbooks	cc-by-nc-sa-4.0	synthetic-text	1.0	✓	nampdn-ai/tiny-orca-textbooks
sciphi-textbooks	SciPhi/textbooks-are-all-you-need-lite	llama2	synthetic-text	1.0	✓	SciPhi/textbooks-are-all-you-need-lite
textbook-programming	vikp/textbook_quality_programming	-	synthetic-text	1.0	✓	vikp/textbook_quality_programming
proofpile-algebra	EleutherAI/proof-pile-2	-	math	1.0	✗	(Azerbayev et al., 2023)
openweb-math	open-web-math/open-web-math	-	math	1.0	✗	(Paster et al., 2023)
british-library-books	biglam/blbooks-parquet	cc0-1.0	longform-text	1.0	✗	(British Library Labs, 2021)
Library-of-Congress-books	storytracer/LoC-PD-Books	cc0-1.0	longform-text	1.0	✗	(Majstorovic, 2024)
MathPile	GAIR/MathPile	cc-by-nc-sa-4.0	math	2.0	✗	(Wang et al., 2024b)
CLRS	tomg-group-umd/CLRS-Text-train	Apache-2.0	math	1.0	✓	(Markeeva et al., 2024)
AutoMathText-1	math-ai/AutoMathText	CC BY-SA 4.0	math	1.0	✗	(Zhang et al., 2024c)
AutoMathText-2	math-ai/AutoMathText	CC BY-SA 4.0	math	1.0	✗	(Zhang et al., 2024c)
AutoMathText-3	math-ai/AutoMathText	CC BY-SA 4.0	math	1.0	✗	(Zhang et al., 2024c)
bigcode-commitpack	bigcode/commitpackft	mit	code	1.0	✗	(Muennighoff et al., 2024)
bigcode-stack-python-fns	bigcode/stack-dedup-python-fns	other	code	1.0	✗	(Muennighoff et al., 2024)
VikpPython	vikp/python_code_instructions_filtered	-	code	1.0	✓	vikp/python_code_instructions_filtered
chessllm	mlabonne/chessllm	-	misc-reasoning	1.0	✗	mlabonne/chessllm
WaterHorseChess-pre	Waterhorse/chess_data	apache-2.0	misc-reasoning	1.0	✗	(Feng et al., 2023)
eleutherai-lichess	EleutherAI/lichess-puzzles	CC0 1.0	misc-reasoning	1.0	✗	(Schwarzschild et al., 2021a)

Table 12: Datasets used for model pre-training (Part 2: Instruction Data)

Dataset	Address	License	Category	W	MG	Citation
WebInstruct-prometheus	chargoddard/WebInstructSub-prometheus	apache-2.0	generic-instruct	1.0	✓	(Kim et al., 2024)
hercules	Locutusque/hercules-v5.0	other	generic-instruct	1.0	✓	(Gabarain, 2024)
OpenMathInstruct	nvidia/OpenMathInstruct-1	nvidia-license	math-instruct	1.0	✓	(Toshniwal et al., 2024b)
MetaMathQA	meta-math/MetaMathQA	mit	math-instruct	1.0	✓	(Yu et al., 2023)
CodeFeedback	m-a-p/CodeFeedback-Filtered-Instruction	apache-2.0	generic-instruct	2.0	✓	(Zheng et al., 2024)
Daring-Anteater	nvidia/Daring-Anteater	cc-by-4.0	generic-instruct	1.0	✓	(Wang et al., 2024a)
Nvidia-Blender	nvidia/sft_datablend_v1	cc-by-4.0	generic-instruct	1.0	✓	nvidia/sft_datablend_v1
baai-instruct-foundation	BAAI/Infinity-Instruct	-	generic-instruct	1.0	✓	BAAI/Infinity-Instruct
baai-instruct-gen	BAAI/Infinity-Instruct	-	generic-instruct	1.0	✓	BAAI/Infinity-Instruct
anthracite-stheno	anthracite-org/Stheno-Data-Filtered	-	math-instruct	1.0	✓	anthracite-org/Stheno-Data-Filtered
opus-writing	Nopm/Opus_WritingStruct	apache-2.0	writing-instruct	2.0	✓	Nopm/Opus_WritingStruct
math-step	xinlai/Math-Step-DPO-10K	-	math-instruct	2.0	✓	(Lai et al., 2024)
bigcode-oss	bigcode/self-oss-instruct-sc2-exec-filter-50k	-	generic-instruct	1.0	✓	sc2-instruct
everyday-conversations	HuggingFaceTB/everyday-conversations	apache-2.0	writing-instruct	3.0	✓	HuggingFaceTB/everyday-conversations
gsm8k	hkust-nlp/gsm8k-fix	mit	math-instruct	1.0	✗	(Cobbe et al., 2021)
no-robots	HuggingFaceH4/no_robots	cc-by-nc-4.0	writing-instruct	3.0	✗	(Ouyang et al., 2022)
longwriter	THUDM/LongWriter-6k	apache-2.0	writing-instruct	2.0	✓	(Bai et al., 2024)
webglm-qa	THUDM/webglm-qa	-	generic-instruct	1.0	-	(Liu et al., 2023b)
ArxivInstruct	AlgorithmicResearchGroup/ArXivDLInstruct	mit	math-instruct	1.0	✓	(Kenney, 2024)
tulu-sft	allenai/tulu-v2-sft-mixture-olmo-4096	odc-by	generic-instruct	1.0	✓	(Groeneveld et al., 2024)
P3	bigscience/P3	apache-2.0	generic-instruct	1.0	✗	(Sanh et al., 2021)
OrcaSonnet	Gryphe/Sonnet3.5-SlimOrcaDedupCleaned	mit	writing-instruct	2.0	✓	Gryphe/Sonnet3.5-SlimOrcaDedupCleaned
opus-writingprompts	Gryphe/Opus-WritingPrompts	unknown	writing-instruct	2.0	✓	Gryphe/Opus-WritingPrompts
reddit-writing	nothingiisreal/Reddit-Dirty-And-WritingPrompts	apache-2.0	writing-instruct	2.0	✗	Reddit-Dirty-And-WritingPrompts
kalomaze-instruct	nothingiisreal/Kalomaze-Opus-Instruct-25k-filtered	apache-2.0	writing-instruct	2.0	✓	Kalomaze-Opus-Instruct-25k
lean-github	internlm/Lean-Github	apache-2.0	math-instruct	3.0	✗	(Wu et al., 2024)
lean-workbook	pkuAI4M/LeanWorkbook	apache-2.0	math-instruct	3.0	✗	(Ying et al., 2024)
mma	casey-martin/multilingual-mathematical-autoformalization	apache-2.0	math-instruct	3.0	✗	(Jiang et al., 2023)
lean-dojo-informal	AI4M/leandojo-informalized	-	math-instruct	3.0	✗	(Yang et al., 2023)
cpp-annotations	casey-martin/oa_cpp_annotate_gen	-	generic-instruct	1.0	✓	moyix
lean-tactics	l3lab/ntp-mathlib-instruct-st	-	math-instruct	2.0	✗	(Hu et al., 2024)
college-math	ajibawa-2023/Maths-College	apache-2.0	math	1.0	✓	ajibawa-2023/Maths-College
gradeschool-math	ajibawa-2023/Maths-Grade-School	apache-2.0	math	1.0	✓	ajibawa-2023/Maths-Grade-School
general-stories	ajibawa-2023/General-Stories-Collection	apache-2.0	synthetic-text	1.0	✓	ajibawa-2023/General-Stories-Collection
amps-mathematica	XinyaoHu/AMPS_mathematica	mit	math	1.0	✗	XinyaoHu/AMPS_mathematica
amps-khan	XinyaoHu/AMPS_khan	mit	math-instruct	1.0	✗	XinyaoHu/AMPS_khan
Magpie-300k	Magpie-Align/Magpie-Pro-MT-300K-v0.1	llama3	generic-instruct	1.0	✓	(Xu et al., 2024)
Magpie-reasoning	Magpie-Align/Magpie-Reasoning-150K	llama3	generic-instruct	1.0	✓	(Xu et al., 2024)
prox-fineweb	gair-prox/FineWeb-pro	odc-by	generic-text	1.0	✗	(Zhou et al., 2024)
prox-c4	gair-prox/c4-pro	odc-by	generic-text	1.0	✗	(Zhou et al., 2024)
prox-redpajama	gair-prox/RedPajama-pro	odc-by	generic-text	1.0	✗	(Zhou et al., 2024)
prox-open-web-math	gair-prox/open-web-math-pro	odc-by	math	1.0	✗	(Zhou et al., 2024)
together-long-data	togethercomputer/Long-Data-Collections	other	longform-text	1.0	✗	(TogetherAI, 2023)
project-gutenberg-19	emozilla/pg19	apache-2.0	longform-text	1.0	✗	(Rae et al., 2019)
mathgenie	MathGenie/MathCode-Pile	apache-2.0	math	1.0	✗	(Lu et al., 2024)
reasoning-base	KingNish/reasoning-base-20k	apache-2.0	math	1.0	✓	KingNish/reasoning-base-20k
OpenMathInstruct-2	nvidia/OpenMathInstruct-2	nvidia-license	math-instruct	1.0	✓	(Toshniwal et al., 2024a)
Txt360-DM	LLM360/TxT360	odc-by	math	1.0	✗	(Liping Tang, 2024)
Txt360-ubuntu-chat	LLM360/TxT360	odc-by	Q&A-text	1.0	✗	(Liping Tang, 2024)
markdown-arxiv	neuralwork/arxiver	cc-by-nc-sa-4.0	scientific-text	2.0	✗	neuralwork/arxiver