Exposing Attention Glitches with Flip-Flop Language Modeling

Anonymous Authors[1]

Abstract

Why do large language models hallucinate? This work identifies and analyzes the phenomenon of *attention glitches*, in which the Transformer architecture's inductive biases intermittently fail to capture robust reasoning. To isolate the issue, we introduce *flip-flop language modeling* (FFLM), a parametric family of synthetic benchmarks designed to probe the extrapolation of language models. This simple generative task requires a model to copy binary symbols over long-range dependencies, ignoring the tokens in between. We find that Transformer FFLMs suffer from a long tail of sporadic reasoning errors, some of which we can eliminate using various regularization techniques. Our preliminary mechanistic analyses show why the remaining errors may be very difficult to diagnose and resolve. We hypothesize that attention glitches account for (some of) the closed-domain hallucinations in natural LLMs.

1. Introduction

Large language models (LLMs) are known to produce incorrect outputs, often referred to colloquially as "hallucinations", creating challenges of their safe deployment (Ji et al., 2023). Generally, hallucinations refer to the phenomenon that a model's outputs are syntactically and grammatically accurate but factually incorrect. There are various types of hallucinations, and the focus of this work is the "closed-domain" variety (Saparov & He, 2022; OpenAI, 2023), where the model predictions contain factually incorrect or made-up information *according to a given context*, regardless of their correctness in the real world. Perhaps surprisingly, such hallucinations can be observed even on simple algorithmic reasoning tasks. As a warmup, consider the queries shown in Figure 1 (and Appendix B.1), where we prompt LLMs to solve addition problems of various lengths. The responses simultaneously illustrate the following:

1. *Nontrivial algorithmic generalization:* In cases where the models succeed, it is unlikely that these exact numerical sequences appeared in the training data. To correctly output the first digit of the answer, the LLM must resolve a long dependency chain which generally depends on every digit in the input. Somewhere within these networks' internal representations, implementations of addition algorithms have emerged.

2. *Sporadic errors ("hallucinations"):* These internal algorithms can be brittle and unreliable, especially when processing long inferential chains. Their failures can be subtle and unpredictable.

In this work, we consider *flip-flop language* processing, a minimal unit of sequential computation which consists of memory operations on a single bit (see Definition 1) and underlies virtually all[1] syntactic parsing and algorithmic reasoning capabilities. A *flip-flop language modeling* (FFLM) task is defined on sequences of `write`, `read`, and `ignore` instructions: `write` sets the memory state to a certain value which is later retrieved by `read`, while `ignor`ing any contents in between. We find that when trained to complete flip-flop sequences, the Transformer architecture exhibits a long tail of reasoning errors, unlike previous-generation recurrent models such as the LSTM (Hochreiter & Schmidhuber, 1997). We coin the term *attention glitch* for this phenomenon, and hypothesize that this captures a systematic failure mode of Transformer-based LLMs when internally representing long chains of algorithmic reasoning.

Our contributions are as follows:

- **FFLM: a minimalistic long-range dependency benchmark.** We propose *flip-flop language modeling*, a parametric family of synthetic benchmarks for autoregressive sequence modeling, designed to isolate and probe reasoning errors like those demonstrated in Figure 1.

- **An empirical failure of attention to attend.** We find that while Transformer models can appear to learn flip-flop languages perfectly on held-out samples in distribution, they make a long tail of unpredictable reasoning errors

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

[1]More precisely, whenever the desired algorithm needs to "store memory" (i.e. contains a non-invertible state transformation); see Section 2.

User: What is 8493 + 2357?
GPT-3.5: `10850` ✓
GPT-4: `10850` ✓

User: What is 84935834 + 23572898?
GPT-3.5: `108008732` ×
GPT-4: `108508732` ✓

User: What is 9991999919909993 + 6109199190990097?
GPT-3.5: `16111199190810090` ×
GPT-4: `16101199100890090` ×
Answer: `16101199110900090`

Figure 1: Cherry-picked integer addition prompts: state-of-the-art LLMs can generalize non-trivially on algorithmic sequences, but sporadic reasoning errors persist. This (and many other algorithmic reasoning capabilities) can be implemented by a Transformer model using internal *flip-flops*.

(*attention glitches*) OOD on both long-range and short-range dependencies. We evaluate various direct and indirect mitigations, including commonly-used regularization techniques and *attention-sharpening regularizers*—a plug-and-play way to sparsify self-attention architectures. We find that attention-sharpening reduces reasoning errors by an order of magnitude, but none of our attempts were successful in driving the number of errors to exactly 0. Meanwhile, even tiny recurrent models work perfectly.

- **Preliminary mechanistic analyses.** We provide some theoretical and empirical explorations which account for some of the internal mechanisms for attention glitches, and why they are so difficult to eliminate completely.

Related work. It has become an important empirical challenge to eliminate the sporadically erroneous outputs of LLMs, popularly called "hallucinations" (Saparov & He, 2022; Ji et al., 2023). Our investigation opens the architectural black-box towards these ends (see the discussion in Appendix A.5); other approaches include explicitly writing out the intermediate reasoning steps (Nye et al., 2021; Wei et al., 2022), and self-consistency (Wang et al., 2022). There have also been many datasets and tasks designed to isolate considerations similar to ours (Tay et al., 2020; Wu et al., 2021; Zhang et al., 2021; 2022; Saparov & He, 2022; Shi et al., 2023). Aside from being focused on the "smallest" and "purest" sequential reasoning capability, FFLM is distinguished by a few factors:

- **"L_∞" objective:** Unlike usual benchmarks, we consider any model with less than 100% accuracy as exhibiting a *reasoning error*. Aside from the motivation of completely eliminating hallucinations, we argue that this stringent notion of correctness is needed to avoid error amplification when flip-flops are embedded in more complex networks (see Appendix A.1).
- **Parametric, procedurally generated, and generalizable:** Our empirical study precisely quantifies long-tail errors via a large number of replicates over the randomness of both model initialization and data generation. Our methodology can be adapted and resized to probe language models of any size.

Detailed discussions are deferred to Appendix A.2.

2. Flip-flops and FFLM

For any even number $T \geq 4$, we define a flip-flop string as a sequence of symbols $\{\mathtt{w}, \mathtt{r}, \mathtt{i}, 0, 1\}^T$, which have the semantics of *instructions* (`write`, `read`, `ignore`) and *data* (one bit). A valid flip-flop string consists of alternating pairs of instructions and data (e.g. "`w 0 i 1 i 0 r 0`"), for which every symbol following a `r` instruction must be equal to the symbol following the most recent `w`; thus, "`w 0 i 1 w 1 r 0`" is not a legal flip-flop string. These sequences can be viewed as correct execution transcripts of a program which can (perhaps occasionally) `write` to a single bit of memory, and always correctly `reads` its contents. We also require that all sequences begin with `w`.

We define a canonical family of the distributions of *flip-flop languages*: let $\mathsf{FFL}(T, \mathbf{p})$ be the distribution over length-T flip-flop strings, parameterized by $\mathbf{p} = (p_{\mathtt{w}}, p_{\mathtt{r}}, p_{\mathtt{i}}) \in \Delta(\{\mathtt{w}, \mathtt{r}, \mathtt{i}\})$, such that:

(i) The first instruction x_1 is always `w`, and the last instruction x_{T-1} is always `r`.

(ii) The other instructions are drawn i.i.d. according to $(p_{\mathtt{w}}, p_{\mathtt{r}}, p_{\mathtt{i}})$ with $p_{\mathtt{i}} = 1 - p_{\mathtt{w}} - p_{\mathtt{r}}$.

(iii) The nondeterministic data symbols (paired with `w` or `i`) are drawn i.i.d. and uniformly.

We are interested in whether language models can learn a flip-flop language from samples, which we define as processing the `read` operations *perfectly*. In this paper, we focus on the **deterministic ("clean") mode**, [2] where the predictions are on deterministic positions only; that is, the model is only required to correctly output x_{t+1} for the prefixes $x_{1:t}$ such that $x_t = \mathtt{r}$. At the cost of a slight departure from vanilla language modeling, this setting isolates the long-range memory task. It is similar to the non-autoregressive flip-flop monoid simulation problem (Liu et al., 2023), and it is easy to see that recurrent networks and 2-layer Transformers (see Proposition 2) *can* both represent FFLM parsers. The question of whether they *do*, especially from less-than-ideal data, turns out to be extremely subtle, and is the subject of the remainder of this paper.

Why focus on the flip-flop? There are several perspectives on why the flip-flop is fundamental: (1) Flip-flop simulation (maintaining memory in a sequence) is a direct necessity in many reasoning settings (Figure 4c). It is a special (depth-1) case of Dyck language processing (Chomsky &

[2] We also look at another *generative* ("noisy") mode which is closer to language modeling; see Appendix B.2 for details.

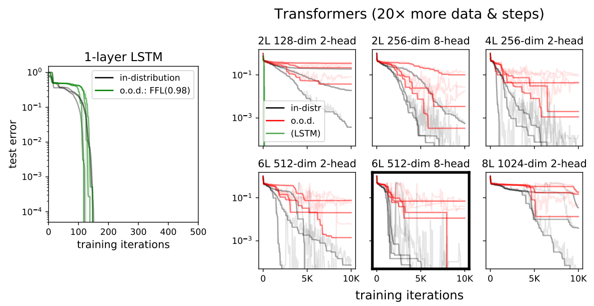


Figure 2: *Top:* Training curves of recurrent (left) vs. Transformer (right) architectures on FFLM, with *best-so-far* evaluation errors highlighted for clarity. **Transformers fail to extrapolate robustly** on this extremely simple task (bolded box denotes our choice of canonical baseline).

Schützenberger, 1959; Yao et al., 2021; Zhao et al., 2023), which is necessary for parsing recursive grammars. (2) Flip-flops are the computational building blocks of memory: the *flip-flop monoid* $\mathcal{F}$ (Definition 1), an algebraic encoding of a flip-flop's dynamics, plays an essential role in the Krohn-Rhodes theory of automata and semigroups (Rhodes et al., 2010). (3) Attention was originally (Bahdanau et al., 2014; Luong et al., 2015; Vaswani et al., 2017) designed to *attend* to (i.e. selectively retrieve and copy) data over long-range dependencies. Indeed, it is easy to verify a single attention head can perform this lookup (Proposition 2).

3. Attention glitches: a long tail of errors for Transformer FFLMs

In our main battery of synthetic experiments, we train neural language models to generate strings from the flip-flop language $\mathsf{FFL}(T = 512, \mathbf{p} = (0.1, 0.1, 0.8))$ (for short, $\mathsf{FFL}(p_{\mathtt{i}} = 0.8)$), and probe whether the networks robustly learn the language. Although every valid flip-flop string is supported in this distribution, some sequences are far rarer than others; we measure tail behavior via probes of extrapolation, defined here as out-of-distribution evaluations which amplify the probabilities of the rare sequences. To create these "challenging" sequences, we sample $> 3 \times 10^5$ sequences from $\mathsf{FFL}(0.98)$ (containing unusually many "sparse" sequences with mostly `ignore` instructions), as well as $\mathsf{FFL}(0.1)$ (many "dense" sequences). Training and evaluating the `read` accuracies of Transformer models of various sizes, as well as a recurrent LSTM model, we find the following results (see Figure 2):

(R1) **Transformers exhibit a long, irregular tail of errors.** Such errors occur on both sparse and dense sequences. Further, a model's out-of-distribution test error varies significantly between random seeds, and even between iterates within the same training run.

(R2) **1-layer LSTM extrapolates perfectly.** In stark contrast, with 20 times fewer training samples and iterations, a small recurrent model achieves 100% accuracy, on 100 out of 100 runs.

As a counterpart to these findings, we observe similar anomalies in real LLMs, when prompted to complete natural textual embeddings (Figure 4, top right) of flip-flop tasks:

(R3) **10B-scale natural LMs can correctly process flip-flop languages, but not robustly.** Beyond a certain scale, natural language models can learn to process (natural embeddings of) flip-flop languages from in-context demonstrations. However, this emergent capability is not robust: there exist rare `read` errors, whose probabilities amplify as the sequence length T grows. We provide details for the few-shot evaluation protocol in Appendix B.2.1.

We discuss potential mechanisms that account for attention glitches in Appendix A.4.

4. Mitigations for attention glitches

We investigate various approaches towards eliminating the long tail of reasoning errors exhibited by Transformer FFLMs. We select the 19M-parameter model (with $L = 6$ layers, $d = 512$ embedding dimensions, and $H = 8$ heads) from Section 3 as a canonical baseline, and conduct precise evaluations of various direct and indirect interventions.

4.1. Direct solutions

We begin by examining what is perhaps the most obvious solution: removing the need for out-of-distribution extrapolation, by training directly with **improved data coverage**. Indeed, we verify that this works near-perfectly:

(R4) **Training on rare sequences works best, by a wide margin.** By training on a uniform mixture of FFL distributions with $p_{\mathtt{i}} = \{0.9, 0.98, 0.1\}$, the baseline architecture reliably converges to solutions with significantly fewer errors on each of these 3 distributions (teal violins in Figure 3).

This should not be surprising, in light of the realizability of flip-flops by self-attention (and, more generally, the existence of shortcuts functionally identical to RNNs (Liu et al., 2023)), and corroborates similar conclusions from (Zhang et al., 2021). We also find that weaker improvements emerge by straightforwardly increasing scale parameters in the model and training pipelines:

(R5) **Resource scaling (in-distribution data, training steps, network size) helps**, but the improvements are orders of magnitude smaller than those in (R4), and we observe tradeoffs between sparse- and dense-sequence extrapolation (blue violins in Figure 3).

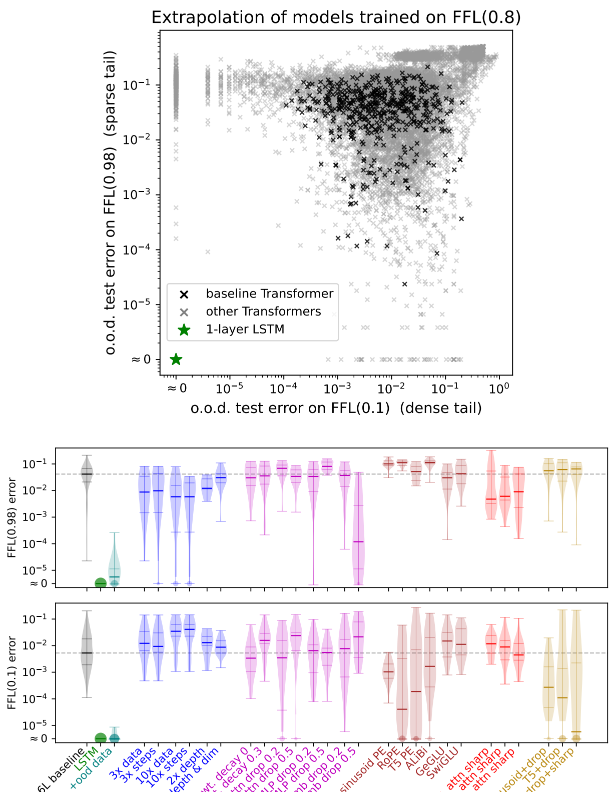


Figure 3: A long tail of flip-flop errors for 10,625 Transformer models; some configurations reduce attention glitch rates by orders of magnitude. *Left:* Out-of-distribution evaluations for all models. *Right:* Effects of individual architectural and algorithmic choices; dots at the bottom indicate runs with 0 error.

Another class of direct solutions is to *externalize the chain of thought* (CoT), for which we provide additional references and discussions in Appendix A.2.

4.2. Indirect algorithmic controls: a bag of regularization tricks

The interventions listed in Section 4.1 are all potentially practical, and may shed light on how closed-domain LLM hallucinations will diminish with data quality, scale, and improved inference strategies. However, it is not always *feasible* to implement these fixes under resource constraints (especially data). We next investigate an orthogonal design space, of how to robustify the *internal* memory mechanisms of neural sequence models. Note that the exceptionally strong extrapolative performance of the LSTM provides a "skyline", showing the possibility of far more robust architectures than the Transformer (in the flip-flop setting, with this restricted set of considerations).

We investigate a large array of not-fully-understood algorithmic tricks for "smoothing" the behavior of LLMs: weight decay, dropout, batch sizes, learning rates, optimizer hyperparameters, position embeddings, and activation functions. We also train Transformer models with *attention-sharpening regularizers*:[3] during training, for attention weights $\alpha \in \Delta([T])$, adding differentiable loss terms which encourage sparsity (e.g. the mixture's entropy $H(\alpha)$, or negative p-norms $-\|\alpha\|_2$, $-\|\alpha\|_\infty$).

(R6) **Many algorithmic choices influence extrapolative behaviors;** see the purple, brown, red, and gold violins in Figure 3 (right). Our strongest improvements on sparse sequences are obtained by large (0.5) embedding dropout and attention-sharpening losses; on dense sequences, non-trainable position embeddings are the most helpful.

(R7) **Despite many partial mitigations, nothing eliminates attention glitches entirely.** We found it extremely difficult to find a setting that reliably produces Transformers with simultaneous improvements over the baseline on sparse and dense sequences (Figure 3 left), which is trivial to do so with an LSTM.

Additionally, our preliminary mechanistic study shows that (details deferred to Appendix B.5):

(R8) **Attention-sharpening regularizers successfully promote hard attention, but errors persist.**

5. Conclusion and future challenges

We have introduced *flip-flop language modeling* (FFLM), a synthetic benchmark for probing the fine-grained extrapolative behavior of neural sequence models, based on a one-bit memory operation which forms a fundamental building block of algorithmic reasoning. Transformer models, trained on insufficiently diverse flip-flop sequences, make a long tail of sporadic reasoning errors, which we call *attention glitches*.[4] Through extensive controlled experiments, we find that many algorithmic mitigations can reduce the frequency of attention glitches, but none can eliminate them entirely. The strikingly outsized benefit of replacing the Transformer with an LSTM network suggests that *architectural* innovations towards the same ends are well worth examining.

[3] While less popular, such losses have been used to sparsify dependencies in similar contexts (Zhang et al., 2018; Sukhbaatar et al., 2021).

[4] In Appendix A.5, we discuss connections to the phenomenon of "closed-domain hallucinations" in non-synthetic LLMs (e.g. the errors demonstrated in Figure 1), and ambiguities inherent in generalizing the definition of attention glitches.

References

Anil, C., Wu, Y., Andreassen, A., Lewkowycz, A., Misra, V., Ramasesh, V., Slone, A., Gur-Ari, G., Dyer, E., and Neyshabur, B. Exploring length generalization in large language models. *arXiv preprint arXiv:2207.04901*, 2022.

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

Barak, B., Edelman, B., Goel, S., Kakade, S., Malach, E., and Zhang, C. Hidden progress in deep learning: Sgd learns parities near the computational limit. *Advances in Neural Information Processing Systems*, 35:21750–21764, 2022.

Barrington, D. A. M. and Thérien, D. Finite monoids and the fine structure of NC^1. *Journal of the ACM*, 1988.

Bertsch, A., Alon, U., Neubig, G., and Gormley, M. R. Unlimiformer: Long-range transformers with unlimited length input. *arXiv preprint arXiv:2305.01625*, 2023.

Bhattamishra, S., Ahuja, K., and Goyal, N. On the ability and limitations of transformers to recognize formal languages. In *Conference on Empirical Methods in Natural Language Processing*, 2020.

Bolukbasi, T., Pearce, A., Yuan, A., Coenen, A., Reif, E., Vi'egas, F., and Wattenberg, M. An interpretability illusion for bert. *ARXIV.ORG*, 2021.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chen, S., Zhang, F., Sone, K., and Roth, D. Improving faithfulness in abstractive summarization with contrast candidate generation and selection. *North American Chapter Of The Association For Computational Linguistics*, 2021. doi: 10.18653/V1/2021.NAACL-MAIN.475.

Chiang, D. and Cholak, P. Overcoming a theoretical limitation of self-attention. *arXiv preprint arXiv:2202.12172*, 2022.

Chomsky, N. and Schützenberger, M. P. The algebraic theory of context-free languages. In *Studies in Logic and the Foundations of Mathematics*. 1959.

Creswell, A., Shanahan, M., and Higgins, I. Selection-inference: Exploiting large language models for interpretable logical reasoning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3Pf3Wg6o-A4.

Dai, Z., Yang, Z., Yang, Y., Carbonell, J., Le, Q. V., and Salakhutdinov, R. Transformer-xl: Attentive language models beyond a fixed-length context. *ACL*, 2019.

Dhingra, B., Faruqui, M., Parikh, A., Chang, M.-W., Das, D., and Cohen, W. Handling divergent reference texts when evaluating table-to-text generation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4884–4895, Florence, Italy, jul 2019. Association for Computational Linguistics. doi: 10.18653/v1/P19-1483. URL https://aclanthology.org/P19-1483.

Dziri, N., Madotto, A., Zaiane, O., and Bose, A. Neural path hunter: Reducing hallucination in dialogue systems via path grounding. *Conference On Empirical Methods In Natural Language Processing*, 2021. doi: 10.18653/v1/2021.emnlp-main.168.

Dziri, N., Milton, S., Yu, M., Zaiane, O. R., and Reddy, S. On the origin of hallucinations in conversational models: Is it the datasets or the models? *North American Chapter Of The Association For Computational Linguistics*, 2022. doi: 10.48550/arXiv.2204.07931.

Eccles, W. and Jordan, F. A trigger relay utilizing three-electrode thermionic vacuum tubes. *The Electrician*, 83: 298, 1919.

Eccles, W. H. and Jordan, F. W. Improvements in ionic relays. *British patent number: GB*, 148582:704, 1918.

Eilenberg, S. *Automata, languages, and machines*. Academic Press, 1974.

Garg, S., Tsipras, D., Liang, P. S., and Valiant, G. What can transformers learn in-context? a case study of simple function classes. *Advances in Neural Information Processing Systems*, 35:30583–30598, 2022.

Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020.

He, T., Zhang, J., Zhou, Z., and Glass, J. R. Exposure bias versus self-recovery: Are distortions really incremental for autoregressive text generation? *Conference On Empirical Methods In Natural Language Processing*, 2019. doi: 10.18653/v1/2021.emnlp-main.415.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

Jain, S. and Wallace, B. C. Attention is not explanation. *North American Chapter Of The Association For Computational Linguistics*, 2019. doi: 10.18653/v1/N19-1357.

Ji, Z., Lee, N., Frieske, R., Yu, T., Su, D., Xu, Y., Ishii, E., Bang, Y. J., Madotto, A., and Fung, P. Survey of hallucination in natural language generation. *ACM Computing Surveys*, 55(12):1–38, 2023.

Khandelwal, U., He, H., Qi, P., and Jurafsky, D. Sharp nearby, fuzzy far away: How neural language models use context. *arXiv preprint arXiv:1805.04623*, 2018.

Khandelwal, U., Levy, O., Jurafsky, D., Zettlemoyer, L., and Lewis, M. Generalization through memorization: Nearest neighbor language models. *International Conference On Learning Representations*, 2019.

Krohn, K. and Rhodes, J. Algebraic theory of machines, I: Prime decomposition theorem for finite semigroups and machines. *Transactions of the American Mathematical Society*, 1965.

Lanchantin, J., Toshniwal, S., Weston, J., Szlam, A., and Sukhbaatar, S. Learning to reason and memorize with self-notes. *arXiv preprint arXiv: Arxiv-2305.00833*, 2023.

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. 2023. doi: 10.48550/arXiv.2210.10749. URL https://openreview.net/forum?id=De4FYqjFueZ.

Longpre, S., Perisetla, K., Chen, A., Ramesh, N., DuBois, C., and Singh, S. Entity-based knowledge conflicts in question answering. *Conference On Empirical Methods In Natural Language Processing*, 2021. doi: 10.18653/v1/2021.emnlp-main.565.

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

Luong, M.-T., Pham, H., and Manning, C. D. Effective approaches to attention-based neural machine translation. *arXiv preprint arXiv:1508.04025*, 2015.

Malkin, N., Wang, Z., and Jojic, N. Coherence boosting: When your pretrained language model is not paying enough attention. In Muresan, S., Nakov, P., and Villavicencio, A. (eds.), *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, pp. 8214–8236. Association for Computational Linguistics, 2022. doi: 10.18653/v1/2022.acl-long.565. URL https://doi.org/10.18653/v1/2022.acl-long.565.

Meister, C., Lazov, S., Augenstein, I., and Cotterell, R. Is sparse attention more interpretable? *Annual Meeting Of The Association For Computational Linguistics*, 2021. doi: 10.18653/v1/2021.acl-short.17.

Nanda, N. and Lieberum, T. A mechanistic interpretability analysis of grokking. *Alignment Forum*, 2022. URL https://www.alignmentforum.org/posts/N6WM6hs7RQMKDhYjB/a-mechanistic-interpretability-analysis-of-grokki

Nogueira, R., Jiang, Z., and Lin, J. Investigating the limitations of transformers with simple arithmetic tasks. *arXiv:2102.13019*, 2021.

Nye, M., Andreassen, A. J., Gur-Ari, G., Michalewski, H., Austin, J., Bieber, D., Dohan, D., Lewkowycz, A., Bosma, M., Luan, D., et al. Show your work: Scratchpads for intermediate computation with language models. *arXiv preprint arXiv:2112.00114*, 2021.

OpenAI. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Orvieto, A., Smith, S. L., Gu, A., Fernando, A., Gulcehre, C., Pascanu, R., and De, S. Resurrecting recurrent neural networks for long sequences. *ARXIV.ORG*, 2023. doi: 10.48550/arXiv.2303.06349.

Parikh, A. P., Wang, X., Gehrmann, S., Faruqui, M., Dhingra, B., Yang, D., and Das, D. Totto: A controlled table-to-text generation dataset. *Conference On Empirical Methods In Natural Language Processing*, 2020. doi: 10.18653/v1/2020.emnlp-main.89.

Paszke, A., Gross, S., Chintala, S., Chanan, G., Yang, E., DeVito, Z., Lin, Z., Desmaison, A., Antiga, L., and Lerer, A. Automatic differentiation in pytorch. 2017.

Petroni, F., Rocktäschel, T., Lewis, P., Bakhtin, A., Wu, Y., Miller, A. H., and Riedel, S. Language models as knowledge bases? *Conference On Empirical Methods In Natural Language Processing*, 2019. doi: 10.18653/v1/D19-1250.

Press, O., Smith, N. A., and Lewis, M. Train short, test long: Attention with linear biases enables input length extrapolation. *arXiv preprint arXiv:2108.12409*, 2021.

Rhodes, J., Nehaniv, C. L., and Hirsch, M. W. *Applications of automata theory and algebra: via the mathematical theory of complexity to biology, physics, psychology, philosophy, and games*. World Scientific, 2010.

Saparov, A. and He, H. Language models are greedy reasoners: A systematic formal analysis of chain-of-thought. *arXiv preprint arXiv:2210.01240*, 2022.

Shao, Z., Gong, Y., Shen, Y., Huang, M., Duan, N., and Chen, W. Synthetic prompting: Generating chain-of-thought demonstrations for large language models. *ARXIV.ORG*, 2023. doi: 10.48550/arXiv.2302.00618.

Shi, F., Chen, X., Misra, K., Scales, N., Dohan, D., Chi, E., Schärli, N., and Zhou, D. Large language models can be easily distracted by irrelevant context. *arXiv preprint arXiv:2302.00093*, 2023.

Srivastava, A., Rastogi, A., Rao, A., Shoeb, A. A. M., Abid, A., Fisch, A., Brown, A. R., Santoro, A., Gupta, A., Garriga-Alonso, A., et al. Beyond the imitation game: Quantifying and extrapolating the capabilities of language models. *arXiv preprint arXiv:2206.04615*, 2022.

Su, J., Lu, Y., Pan, S., Murtadha, A., Wen, B., and Liu, Y. Roformer: Enhanced transformer with rotary position embedding. *arXiv preprint arXiv:2104.09864*, 2021.

Sukhbaatar, S., Ju, D., Poff, S., Roller, S., Szlam, A. D., Weston, J., and Fan, A. Not all memories are created equal: Learning to forget by expiring. *International Conference On Machine Learning*, 2021.

Sun, S., Krishna, K., Mattarella-Micke, A., and Iyyer, M. Do long-range language models actually use long-range context? In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 807–822, Online and Punta Cana, Dominican Republic, nov 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.62. URL https://aclanthology.org/2021.emnlp-main.62.

Tay, Y., Dehghani, M., Abnar, S., Shen, Y., Bahri, D., Pham, P., Rao, J., Yang, L., Ruder, S., and Metzler, D. Long range arena: A benchmark for efficient transformers. *arXiv preprint arXiv:2011.04006*, 2020.

Tian, R., Narayan, S., Sellam, T., and Parikh, A. P. Sticking to the facts: Confident decoding for faithful data-to-text generation. *arXiv preprint arXiv: 1910.08684*, 2019.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., and Polosukhin, I. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

Wang, X., Wei, J., Schuurmans, D., Le, Q., Chi, E., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv: Arxiv-2203.11171*, 2022.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E., Le, Q., and Zhou, D. Chain of thought prompting elicits reasoning in large language models. *arXiv preprint arXiv:2201.11903*, 2022.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019.

Wu, Y., Rabe, M. N., Li, W., Ba, J., Grosse, R. B., and Szegedy, C. Lime: Learning inductive bias for primitives of mathematical reasoning. In *International Conference on Machine Learning*, pp. 11251–11262. PMLR, 2021.

Wu, Y., Rabe, M. N., Hutchins, D. S., and Szegedy, C. Memorizing transformers. *International Conference On Learning Representations*, 2022. doi: 10.48550/arXiv.2203.08913.

Yao, S., Peng, B., Papadimitriou, C., and Narasimhan, K. Self-attention networks can process bounded hierarchical languages. *arXiv preprint arXiv:2105.11115*, 2021.

Zeiger, H. P. Cascade synthesis of finite-state machines. *Information and Control*, 1967.

Zhang, C., Raghu, M., Kleinberg, J., and Bengio, S. Pointer value retrieval: A new benchmark for understanding the limits of neural network generalization. *arXiv:2107.12580*, 2021.

Zhang, J., Zhao, Y., Li, H., and Zong, C. Attention with sparsity regularization for neural machine translation and summarization. *IEEE/ACM Transactions on Audio, Speech, and Language Processing*, 27(3):507–518, 2018.

Zhang, Y., Backurs, A., Bubeck, S., Eldan, R., Gunasekar, S., and Wagner, T. Unveiling transformers with lego: a synthetic reasoning task. *arXiv preprint arXiv:2206.04301*, 2022.

Zhao, H., Panigrahi, A., Ge, R., and Arora, S. Do transformers parse while predicting the masked word? *arXiv preprint arXiv:2303.08117*, 2023.

Zheng, C., Liu, Z., Xie, E., Li, Z., and Li, Y. Progressive-hint prompting improves reasoning in large language models. *arXiv preprint arXiv.2304.09797*, 2023.

Zhou, H., Nova, A., Larochelle, H., Courville, A., Neyshabur, B., and Sedghi, H. Teaching algorithmic reasoning via in-context learning. *arXiv preprint arXiv:2211.09066*, 2022.

Appendix

A. Deferred background and discussion

A.1. Flip-flop terminology and history

The *flip-flop automaton*[5] is a two-state machine which remembers a single bit of memory, and enables retrieval of this bit. More precisely, the flip-flop automaton (illustrated in Figure 4(a)) is defined as:

Definition 1 (Flip-flop automaton)**.** *A flip-flop automaton* $\mathcal{A} = \{Q, \Sigma, \delta\}$ *is defined with state space* $Q = \{0, 1\}$*, input alphabet* $\Sigma = \{\sigma_0, \sigma_1, \perp\}$*, and transition function* $\delta : Q \times \Sigma \to Q$ *where*

$$\begin{cases} \delta(q, \sigma_0) & = 0, \\ \delta(q, \sigma_1) & = 1, \\ \delta(q, \perp) & = q; \end{cases} \qquad \forall q \in \{0, 1\}.$$

The semantics of the input symbols can be intuitively be identified with "write 0", "write 1", and "do nothing". This mathematical object is named after a type of electronic circuit which can store a single bit of state information (Eccles & Jordan, 1918; 1919); such physical constructions appear ubiquitously in electrical engineering as the building blocks of memory. The task of interest in Appendix B.5 is *simulating the flip-flop automaton*: the model takes as input a sequence of $x_1, x_2, \cdots, x_T \in \Sigma$, and learns to output the corresponding states $q_t \in Q$ for each $t \in [T]$ after processing inputs $x_{1:t}$.

Naturally associated with the flip-flop automaton is its *transformation monoid*, the closure[6] of its *state transformations* $\delta(\,\cdot\,, \sigma) : Q \to Q$ under function composition. Identifying each symbol with its state transformation map, we can compute the multiplication table of this monoid ($f \circ g$ for every pair of transformations f, g):

	$g = \sigma_0$	$g = \sigma_1$	$g = \perp$
$f = \sigma_0$	σ_0	σ_0	σ_0
$f = \sigma_1$	σ_1	σ_1	σ_1
$f = \perp$	σ_0	σ_1	$\perp$

This algebraic object is called the *flip-flop monoid* $\mathcal{F}$. Its binary operation $\circ$ is clearly non-invertible (intuitively: the history of the bit cannot be recovered after a "memory write") and non-commutative (the order of "write" operations matters); it also has an identity element $\perp$ (which does nothing to the memory bit). By enumeration of smaller objects, it can be seen that $\mathcal{F}$ is the smallest monoid (in terms of order $|\mathcal{F}|$, or fewest number of automaton states $|Q|$) which has these properties.

The flip-flop monoid plays a special role in the algebraic theory of automata (Rhodes et al., 2010): flip-flops can be cascaded to represent more complex functions. In particular, the celebrated Krohn-Rhodes theorem (Krohn & Rhodes, 1965) gives a "prime decomposition" theorem for *all* finite semigroups (associative binary operations), representing them as alternating wreath products of flip-flop monoids and finite simple groups. Further developments (Zeiger, 1967; Eilenberg, 1974) have interpreted this theorem as a structural reparameterization of any finite-state automaton into a feedforward hierarchy of simple "atomic" machines (namely, flip-flops and permutation semiautomata). Basic quantitative questions (e.g. "which functions of n variables can L layers of $\mathrm{poly}(n)$ flip-flops represent?") have proven to be extremely opaque for current mathematical tools; these are studied by the theories of Krohn-Rhodes complexity and circuit complexity.

It was noted by (Barrington & Thérien, 1988) that these reparameterizations of finite-state automata entail the existence of parallel algorithms (i.e. shallow, polynomial-sized circuits) for sequentially executing finite-state recurrences (thus, processing formal languages) on sequences of length T. More recently, (Liu et al., 2023) establish implications for shallow Transformer neural networks: they show that they can size-efficiently (with depth $O(\log T)$ and parameter count

[5] Sometimes, a distinction is made between a semiautomaton (Q, Σ, δ) and an automaton, which is a semiautomaton equipped with a (not necessarily invertible) mapping from states to output symbols. We do not make such a distinction; we equip a semiautomaton with the output function which simply emits the state q, and use "automaton" to refer to this dynamical system.

[6] In this case, the closure is the same as the generator set: no functions distinct from $\sigma_0, \sigma_1, \perp$ can be obtained by composing these three functions. This is not true for a general automaton.

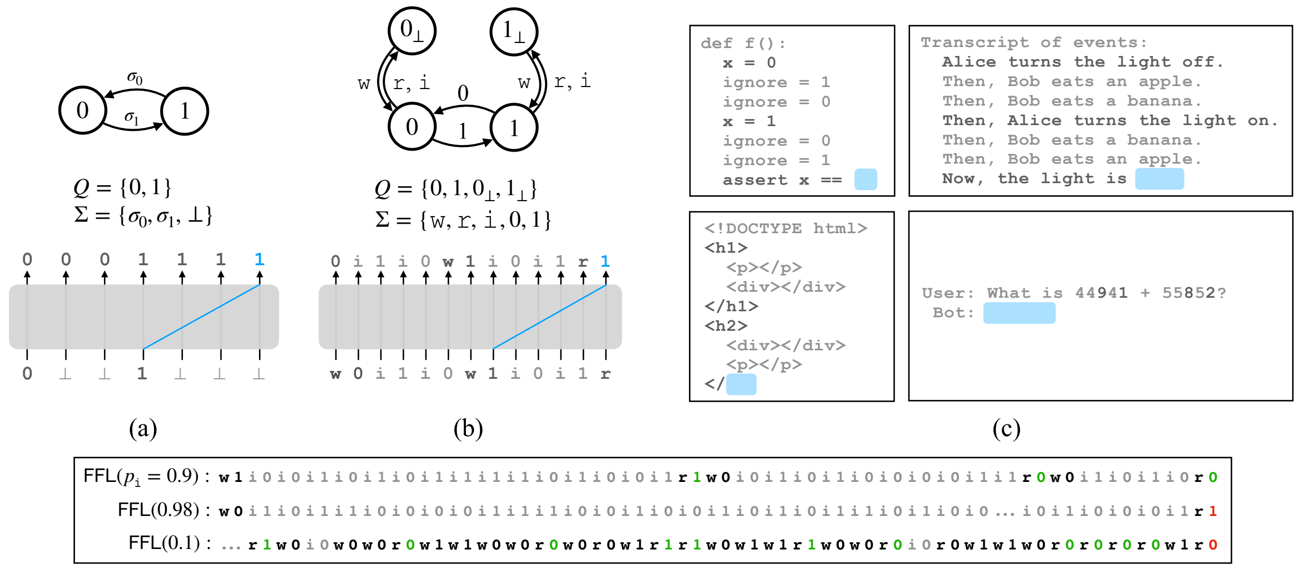


Figure 4: Elementary objects and examples associated with flip-flop languages. (a) the 2-state flip-flop machine (elided transitions are self-loops). (2) A 4-state automaton which processes flip-flop languages (implying the existence of a small RNN). (c) Simple examples of sequential prediction tasks which require processing a flip-flop language. *Bottom:* Examples from the sparser $\mathsf{FFL}(0.98)$ and denser $\mathsf{FFL}(0.1)$ distributions, causing distinct (*long-range* and *short-range*) failure modes for the baseline Transformer model.

$\Theta(T)$; sometimes both improvable to $O(1)$) realize these parallel algorithms, and that standard gradient-based training can empirically learn $\ll T$-layer solutions on a variety of "hard" automata (e.g. composing a sequence of T 5-element permutations; multiplying T unit quaternions). Here, the role of the flip-flop monoid is essential: it provides a natural way to think about the role of a single self-attention head in a hierarchy of indirections, in order to learn a depth-constrained parallel implementation of a sequential algorithm.

A.2. Additional related work

Relevant challenges in NLP: hallucinations and long-range dependencies. The empirical literature is rife with corroborations that neural language models have trouble with robustly fitting long-range memory and multi-step reasoning (Khandelwal et al., 2018; Sun et al., 2021; Sukhbaatar et al., 2021; Malkin et al., 2022; Saparov & He, 2022; Orvieto et al., 2023; Creswell et al., 2023). Such failures can result in "hallucinations": incorrect outputs which either directly contradict factual input in the context, or contain information absent in the context (Ji et al., 2023).

Hallucination can be attributed to various factors, such as the noisiness in data sources (Dhingra et al., 2019; Dziri et al., 2022), imperfect encoding/decoding (Parikh et al., 2020; Tian et al., 2019), or the discrepancy in training and evaluation setups (He et al., 2019). In particular, the most related to our paper are the characteristics inherent to the model itself. For example, prior work has found that Transformers tend to be biased towards information covered during training (Petroni et al., 2019; Longpre et al., 2021), a potential cause to their poor out-of-distribution performance.

In terms of mitigation, various "external" methods (i.e. ones which do not modify the internal representations of the neural network) have been proposed to address some of the above factors, or post-processing model generations (Dziri et al., 2021; Chen et al., 2021), possibly based on several forward passes (Wang et al., 2022; Zheng et al., 2023). Another line of work that have gained much popularity and success is to incorporate explicit memory mechanisms, which we discuss next.

Explicit memory mechanisms in Transformers. Prior work has shown that augmenting the neural network with memory modules or knowledge base helps improve the performance on long-range texts (Khandelwal et al., 2019; Wu et al., 2022; Bertsch et al., 2023). An approach particularly effective for large-scale Transformers is to *externalize the chain of thought* (CoT): train (or finetune, or prompt) the model to explicitly output the intermediate reasoning steps to a "scratchpad" which the model subsequently processes (Nye et al., 2021; Wei et al., 2022; Zhou et al., 2022; Anil et al., 2022; Shao et al., 2023), similar to writing to and reading from a memory tape. A particular way to interact with the scratchpad is to interlace every other token with an annotation of "as a reminder, this is the state" (Liu et al., 2023; Lanchantin et al., 2023), so that there are no more explicit long-range dependencies.

We do not investigate this strategy in this paper, and note that prior work has provided sufficient evidence to affirm its success in inducing the robust learning of recurrences on long synthetic sequences (Anil et al., 2022; Zhou et al., 2022; Liu et al., 2023). Moreover, this strategy cannot fully resolve attention glitches. To begin with, it cannot be guaranteed that a single indivisible reasoning step in a CoT is free of attention glitches; the focus of this work is to isolate and mitigate this intrinsic architectural issue. Additionally, this strategy is the same as the recurrent solution implementable by RNNs, and it does not always exist, especially when attention glitches occur in an internal component of the model.

Transformers and algorithmic tasks. Compared to real-world language datasets, synthetic tasks provide a cleaner and more controlled setup for probing the abilities and limitations of Transformers. Specific to algorithmic reasoning, (Liu et al., 2023) puts a unifying perspective on the ability of small Transformers to succeed at tasks corresponding to algorithmic primitives. Specific tasks of interest include hierarchical languages (Yao et al., 2021; Zhao et al., 2023), modular prefix sums (Anil et al., 2022), adders (Nogueira et al., 2021; Nanda & Lieberum, 2022), regular languages (Bhattamishra et al., 2020), and following a chain of entailment (Zhang et al., 2022).

Comparison with *Transformers Learn Shortcuts to Automata*. Liu et al. (2023) study the parallel circuits efficiently realizable by low-depth Transformers. The authors identify *shortcut solutions*, which exactly replicate length-T recurrent computations ("chains of algorithmic reasoning") in the absence of recurrence, with very few ($O(\log T)$; sometimes $O(1)$) layers. Their results contain a general structure theorem of *representability*, and preliminary positive empirics for *generalization* and *optimization*, demonstrating that Transformers can learn these shallow solutions via gradient-based training on samples. In contrast, the present work is a fine-grained study of the issue of generalization. Our main empirical contributions are a minimally sufficient setup (FFLM) and a set of large-scale[7] controlled experiments, towards providing reasonable scientific foundations for addressing the unpredictable reasoning errors of LLMs.

A.3. Why *this* flip-flop language?

(Liu et al., 2023) (as well as our mechanistic interpretability experiments) use a purer instantiation of flip-flop sequence processing, in which the sequence-to-sequence network is tasked with *non-autoregressive transduction*: given the sequence of input symbols $\sigma_1, \ldots, \sigma_T$, output the sequence of states $q_1, \ldots, q_T$. This is most natural when studying the Transformer architecture's algebraic representations in their most isolated form.

Our autoregressive sequence modeling setting is a slight departure from this setting; we discuss its properties and rationale below.

- The autoregressive setting "type-checks" exactly with standard state-of-the-art autoregressive (a.k.a. causal, forward, or next-token-prediction) language modeling. This makes it more convenient and intuitive as a plug-and-play benchmark.
- The cost is a layer of indirection: the model needs to associate "instruction" tokens with their adjacent "data" tokens. This is a natural challenge for representation learning, and is certainly a necessary cursor for robust extrapolation on natural sequences that embed similar tasks (like those considered in Figure 4c). It is straightforward to remove this challenge: simply tokenize at a coarser granularity (i.e. treat (instruction, data) pair as a distinct vocabulary item).
- The multi-symbol (and variable-length-symbol, etc.) generalizations of the binary flip-flop language are more parsimonious. If there are n instead of 2 tokens, this language can be encoded with = $n + 3$ commands. Without the decoupling of "instruction" tokens from "data", the vocabulary size would scale suboptimally with n.
- The conclusions do not change: in smaller-scale experiments, we observe the same extrapolation failures between the autoregressive and non-autoregressive task formulations.

A.4. Multiplicity of mechanisms for attention glitches

In this section, we discuss how Transformer self-attention modules, when tasked with representing flip-flops, can exhibit multiple (perhaps mutually entangled) failure mechanisms. The accompanying propositions are proven in Appendices C.2

[7] $\sim 10^4$ 19M-parameter Transformers were trained in the making of this paper; see Appendix B.6.

and C.3.

An insufficient explanation: n-gram models. As a warmup, consider a language model $\widehat{\Pr}[x_{t+1}|x_{\leq t}]$ which only depends on the n most recent tokens in the context. Then, if $n \ll \frac{1}{1-p}$, the bulk of $\widehat{\Pr}$'s predictions on $\mathsf{FFL}(p_{\mathtt{i}} = p)$ can be no more accurate than random guessing. This recovers one qualitative trend (degradation of accuracy with dependency length) observed in the experiments. However, this cannot fully explain our findings: it fails to account for the incorrect predictions on dense sequences. Furthermore, the Transformers' outputs on $\mathsf{FFL}(0.98)$ are *mostly* correct; their accuracies on very long-range dependencies are nontrivial, despite not being perfect. There must therefore be subtler explanations for these errors.

Lipschitz limitations of soft attention. Moving to finer-grained failure mechanisms, a known (Hahn, 2020; Chiang & Cholak, 2022) drawback of soft attention is that its softmax operation is "too soft"– for any weight matrices with fixed norms, the attention gets "diluted" across positions as the sequence length T increases, and can fail to perform an intended "selection" operation. We provide a formal statement and proof (Proposition 3) in Appendix C.2.

Difficulty of non-commutative tiebreaking. Can we simply robustify soft attention by replacing it with hard attention? We present a brief analysis which suggests that even hard attention can be brittle. In a stylized setting (one-layer models with linear position encodings), we show that self-attention can *confidently attend to the wrong index*, unless the weight matrices precisely satisfy an orthogonality condition (Proposition 4). This suggests the existence of *spurious local optima*, which we do not attempt to prove end-to-end; however, we provide supporting empirical evidence in the experiments in Appendix C.3.

A.5. Attention glitches in natural LLMs

In this section, we expand on the brief discussion from Section 5. At a high level, *we hypothesize that attention glitches cause (some) closed-domain hallucinations in Transformer models of more complex languages*. However, due to the fact that neural networks' internal representations evade simplistic mechanistic characterization, it is a significant challenge to formulate a rigorous, testable version of this hypothesis. We discuss the subtleties below.

First, we discuss a more general notion of attention glitches, of which the flip-flop errors considered in this papers are a special case. We define attention glitches as *failures of trained attention-based networks to implement a hard retrieval functionality perfectly*. To formalize this notion, there are several inherent ambiguities– namely, the notions of "hard retrieval" and "perfectly", as well as the granularity of "subnetwork" at which an attention glitch can be defined non-vacuously. The FFLM reasoning errors considered in this work provide a minimal and concrete resolution of these ambiguities. We discuss each of these points below:

- **Hard retrieval:** To succeed at FFLM, a network's internal representations must correctly implement the functionality of retrieving a single bit (from a sequence of bits, encoded unambiguously by the network), selected via the criterion of "most recent `write` position". This can be expanded into a richer functional formulation of hard attention, by generalizing the set of possible *retrieved contents* (a discrete set of larger cardinality, or, even more generally, a continuous set), as well as more complex *selection criteria* (e.g. "least recent position").

- **Ground truth:** Of course, to define "errors" or "hallucinations" in reasoning, there must be a well-defined *ideal* functionality. For FFLM, the notion of "closed-domain" reasoning and hallucinations is evident: the ideal behavior is for a model's outputs to coincide with that of the flip-flop machine on all input sequences. This straightforwardly generalizes to all formal languages, where the model is expected to correctly produce the deterministic outputs of automata which parse these languages. By considering expanded notions of "ground truth", it is possible to capture other notions of model hallucinations (such as incorrectly memorized facts). Our work does not address open-domain hallucinations, which may be unrelated to attention glitches.

- **Submodules:** Towards attributing implementations and errors to localized components of a network, it is impossible to provide a single all-encompassing notion of "localized component". This is a perennial challenge faced in the mechanistic interpretability literature. Our work considers two extremes: the entire network (in the main experiments, where we probe end-to-end behavior), and a single self-attention head (in Appendix A.4 and Appendix B.5, in which we probe whether a single attention head can learn multiplication in the flip-flop monoid). Even when considering the

Input	GPT-3.5	GPT-4	Answer
8493 + 2357	10850 ✓	10850 ✓	10850
84935834 + 23572898	108008732 ✗	108508732 ✓	108508732
9991999919909993 + 6109199190990097	16111199100810090 ✗	16101199100890090 ✗	16101199110900090

Table 1: Examples (in Figure 1) of GPT variants on addition: While models tend to succeed at additions with a small number of digits, they (nondeterministically) fail at longer additions.

Input	GPT-3.5	GPT-4	Answer
4491 + 8759	13250 ✓	13250 ✓	13250
80087394 + 63457948	143045342 ✗	143545342 ✓	143545342
5101611078665398 + 8969499832688802	1.4071110911354202e+16 ✗	14071110911354196 ✗	14071110911354200

Table 2: More examples of GPT variants on addition: While models tend to succeed at additions with a small number of digits, they (nondeterministically) fail at longer additions.

same functionality, attention glitches can be considered for different choices of "submodule".[8] Our results reveal a key subtlety: in the presence of overparameterization (more layers and parallel heads than necessary according to the theoretical constructions), Transformers learn to process flip-flop languages via soft attention.

We expect that to effectively debug the full scope of LLM hallucinations, all of the above choices will need to be revisited, perhaps in tandem.

We hypothesize that the algorithmic reasoning capabilities of real LLMs (i.e. their ability to recognize, parse, and transduce formal symbolic languages) are implemented by *internal* subnetworks whose functionalities can be identified with generalizations of the flip-flop machine. To the extent that such modules exist, attention glitches (the failure of these modules to represent the flip-flop operations perfectly, due to insufficient training data coverage) cause sporadic end-to-end errors ("closed-domain hallucinations"). In this work, we have treated the *external* case (where the task is to learn the flip-flop directly).

B. Full experimental results

B.1. Details for LLM addition prompts (Figure 1)

These addition problem queries serve as a quick demonstration of (1) non-trivial algorithmic generalization capabilities of Transformer-based LLMs; (2) the brittleness of such capabilities: we directly address this type of reasoning error in this work. Table 1,2 show these queries and results in detail.

We emphasize that these examples were selected in an adversarial, ad-hoc manner; we do not attempt to formalize or investigate any claim that the errors made by larger models are at longer sequence lengths. We also cannot rule out the possibility that some choice of prompt elicits robust algorithmic reasoning (e.g. the prompting strategies explored in (Zhou et al., 2022)). The only rigorous conclusion to draw from Figure 1 is that of non-robustness: even LLMs exhibiting state-of-the-art reasoning continue to make these elementary errors for some unambiguous queries with deterministic answers. It was last verified on May 8, 2023 that GPT-4 (in its ChatGPT Plus manifestation) demonstrates the claimed failure mode.

[8] Beyond the two extremes considered in this work, some examples include "a subset of attention heads", "a subset of layers", and "a subspace of the entire network's embedding space".

B.2. Extrapolation failures of standard Transformers (Section 3)

This section provides full details for our empirical findings (R1) through (R3).

Architecture size sweep. We consider a sweep over Transformer architecture dimensionalities, varying the three main size parameters. We emphasize that these are somewhat larger than "toy" models: the parameters go up to ranges encountered in natural sequence modeling (though, of course, far short of state-of-the-art LLMs).

- The *number of layers* (depth) $L \in \{2, 4, 6, 8\}$.
- The *embedding dimension* $d \in \{128, 256, 512, 1024\}$.
- The *number of parallel attention heads* per layer $H \in \{2, 4, 8, 16\}$. In accordance with standard scaling rules-of-thumb, each head's dimension is selected to be d/H.

Other hyperparameter choices. We use a sequence length of $T = 512$, again to reflect a typical length of dependencies considered by nontrivial Transformer models. We use a canonical set of training hyperparameters for this sweep: the AdamW (Loshchilov & Hutter, 2017) optimizer, with $(\beta_1, \beta_2) = (0.9, 0.999)$, learning rate 3×10^{-4}, weight decay 0.1, 50 steps of linear learning rate warmup, and linear learning rate decay (setting the would-be 10001th step to 0). We train for 10000 steps on freshly sampled data, and choose a minibatch size of 16; consequently, the models in this setup train on 81,920,000 tokens.

Training and evaluation data. We probe the extrapolative behavior of Transformers on the flip-flop language, training on online samples containing mostly moderate-length dependencies ($p_{\mathtt{i}} = 0.8, p_{\mathtt{w}} = p_{\mathtt{r}} = 0.1$), and evaluating on a distribution containing longer-range dependencies ($p_{\mathtt{i}} = 0.98, p_{\mathtt{w}} = p_{\mathtt{r}} = 0.01$). Every 100 training steps, we evaluate out-of-distribution test errors achieved by these models, on an online evaluation set of 10^3 sequences (which is identical between and within runs; training curves show these errors), containing 3567 occurrences of the `r` instruction. For offline evaluation, we expand this test set to 10^5 sequences, containing 353875 `r` commands, to obtain more precise measurements of o.o.d. error. Training curves are shown with the smaller test set; all other results are reported using the larger one.

(R1) **Transformers exhibit a long, irregular tail of errors.** Figure 5 shows training curves for 3 replicates (random seeds) in each setting, while the scatter plot in the main paper shows variability of out-of-distribution accuracy across random seeds for the baseline setup. We find that Transformers sometimes succeed at extrapolation, but erratically.

(R2) **1-layer LSTM extrapolates perfectly.** We train a 1-layer LSTM (Hochreiter & Schmidhuber, 1997) network, with hidden state dimension 128 (for a total of 133K parameters), for 500 steps with the same optimizer hyperparameters as above. The LSTM model achieves exactly 0 final-iterate o.o.d. error, over 100 out of 100 replicates.

Canonical baseline. We select the **6-layer, 512-dimensional, 8-head** architecture (with 19M trainable parameters) as our *canonical baseline* model: it is large in relevant dimensions[9] to real Transformers, while being small enough to allow for thousands of training runs at a reasonable cost. To fully understand the variability of this single architectural and algorithmic setup, we train and evaluate 500 replicates in this setting.

Random data vs. random initialization. Recent synthetic probes on the surprising behavior of deep neural nets on hard synthetic tasks (Barak et al., 2022; Garg et al., 2022) obtain additional insights by disentangling the effects of *data randomness* (i.e. the precise sequence of minibatches) vs. *model randomness* (e.g. random initialization and dropout). We provide a quick demonstration in Figure 6 (left) that *both sources of stochasticity matter*. We do not perform a more detailed investigation of their precise influence and roles.

Fully generative setting: similar negative results. As mentioned in Section 2, in addition to the deterministic setup where the model is only required to predict for positions where the next token is deterministic, we also consider a **generative ("noisy") mode** where the model estimates the conditional next-token distribution $\Pr[x_{t+1}|x_{1:t}]$, for each $t = 1, \ldots, T-1$. [10] In this mode, the sequences can be treated as drop-in replacements for natural text in GPT-style training. Generative

[9] Except the vocabulary size. In preliminary experiments, we obtained similar findings in the case of token spaces larger than $\{0, 1\}$.

[10] The generative mode is of less interest to this work since predicting the non-deterministic tokens is irrelevant to the memory task at hand.

Figure 5: Examples of training curves over various Transformer architectures, ranging from 46K to 101M trainable parameters. We exhibit 3 (randomly selected) random seeds for each architecture. Lighter curves show raw error percentages, while solid curves denote the lowest error so far in each run. Notice the following: (1) **non-convergence of shallow models** (despite representability) (2) **failure of most runs to extrapolate** (i.e. reach 0% out-of-distribution error); (3) **high variability** between runs; (4) erratic, **non-monotonic progress** on out-of-distribution data, even when the in-distribution training curves appear flat; (5) **a small LSTM outperforms all of these Transformers** (see Figure 2). The $\boxed{\text{bolded box}}$ represents our 19M-parameter baseline model.

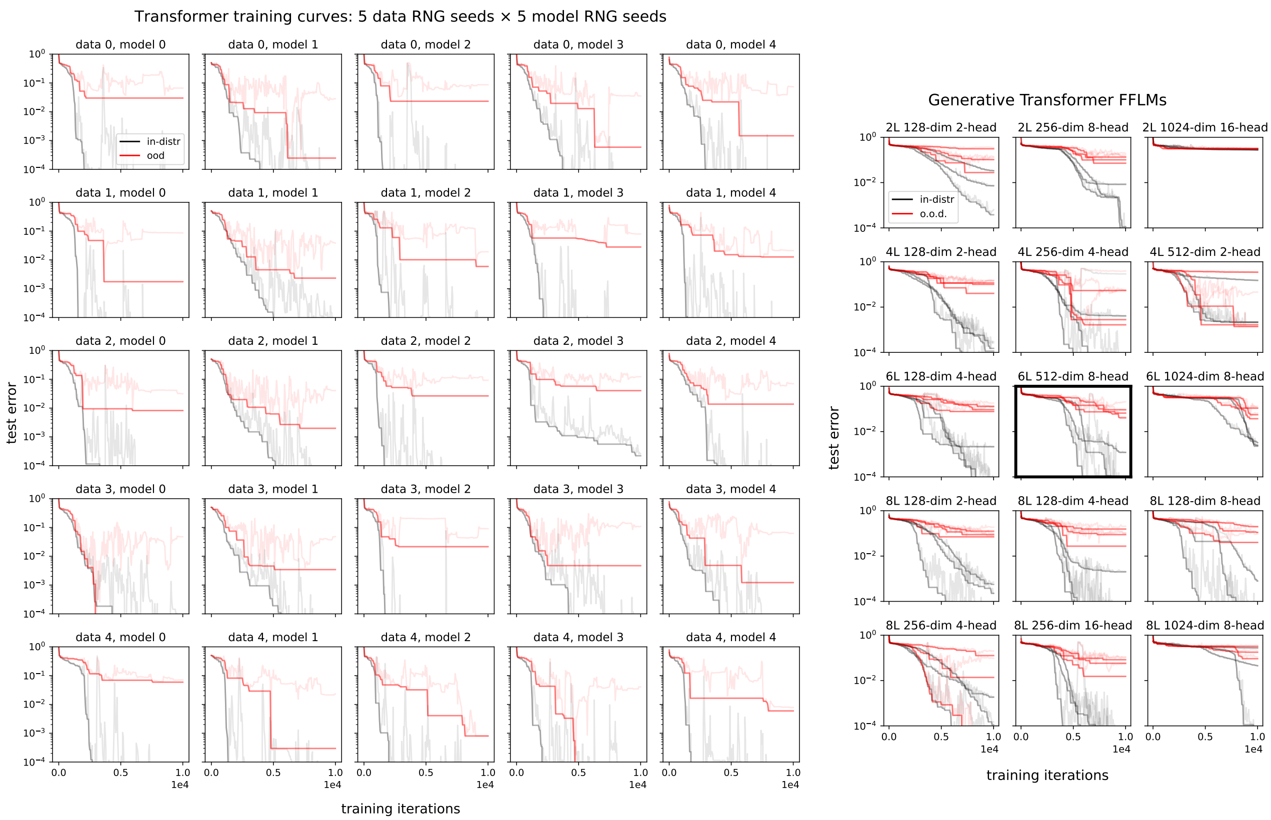


Figure 6: Additional training curves. *Left:* Identical baseline architecture, varying the 5 data seeds and 5 model seeds: models in the same row encounter the same sequence of data, while models in the same column start from identical initializations. **Both sources of randomness affect training dynamics and extrapolation**, and it is not clear which is more important. *Right:* Similar findings for models trained in "fully generative" mode (scoring on all tokens); baseline architecture is in the bolded box.

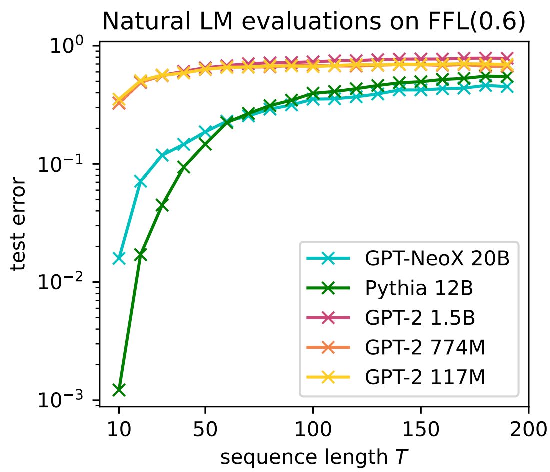


Figure 7: Natural language models fail to extrapolate robustly on FFLM.

FFLMs can be evaluated by checking their completions on prefix "prompts" (e.g. "... `w 0 i 1 i 1 r [?]`").

We observe similar extrapolation failures in this setting. Figure 6 (right) exhibits some training curves for this setting, showing non-extrapolation, variability, and instability. We observe that training (to in-distribution convergence) takes slightly longer in this setting, and usually succeeds with the baseline architecture. We do not perform further controlled experiments in this setting.

B.2.1. Evaluating real LLMs on flip-flops

We provide a quick corroboration that while LLMs in practice can perform in-context reasoning when the sequences are unambiguously isomorphic to a flip-flop language. We use the natural language example from Figure 4 (top right), and evaluate the capability of popular pretrained LLMs to correctly remember the state of a light switch. Specifically, `write` instructions in the FFLM task are either "Alice turns the light off" or "Alice turns the light on". The `ignore` instructions are either "Bob eats a banana" or "Bob eats an apple". All models are prompted with a translated, length-16 FFLM task that's been translated to English in this way before evaluation.

We measure this accuracy as a function of the sequence length for several well-known LLMs, including GPT-2, GPT-2-large, GPT-2-xl, Pythia-12C, and GPT-NeoX-20B. Figure 7 shows how well these models perform on this task (i.e. the correctness of the model when prompted with "The light is turned ") as the sequence length is varied. Consistent with the findings of this paper, larger models tend to perform best at this task, and the quality of all models deteriorates with increased sequence length. Each point on the plot considers 500 sequences of the indicated length. All models were prompted with a randomly generated, length 16 flip flop sequence to allow the model to learn the task in context. Accuracy is measured according to the frequency with which the model correctly predicts the current state of the light switch, as described in Section B.2.1.

(R3) **10B-scale natural LMs can correctly process flip-flop languages, but not robustly.**

Note that it is impossible to quantify the degree to which these sequences are "in-distribution" (it is unlikely that any sequences of this form occur in the training distributions for these LLMs). Much like linguistic reasoning evaluations in the style of BIG-bench (Srivastava et al., 2022), we rely on the emergent capability of in-context inference (Brown et al., 2020) of the task's syntax and semantics. As discussed in Appendix A.5, this layer of indirection, which is impossible to avoid in the finetuning-free regime, can cause additional (and unrelated) failure modes to those studied in our synthetic experiments. Fully reconciling our findings between the synthetic and non-synthetic settings (e.g. by training or finetuning on sequences of this form, or via mechanistic interpretation of non-synthetic language models) is outside the scope of this paper, and yields an interesting direction for future work.

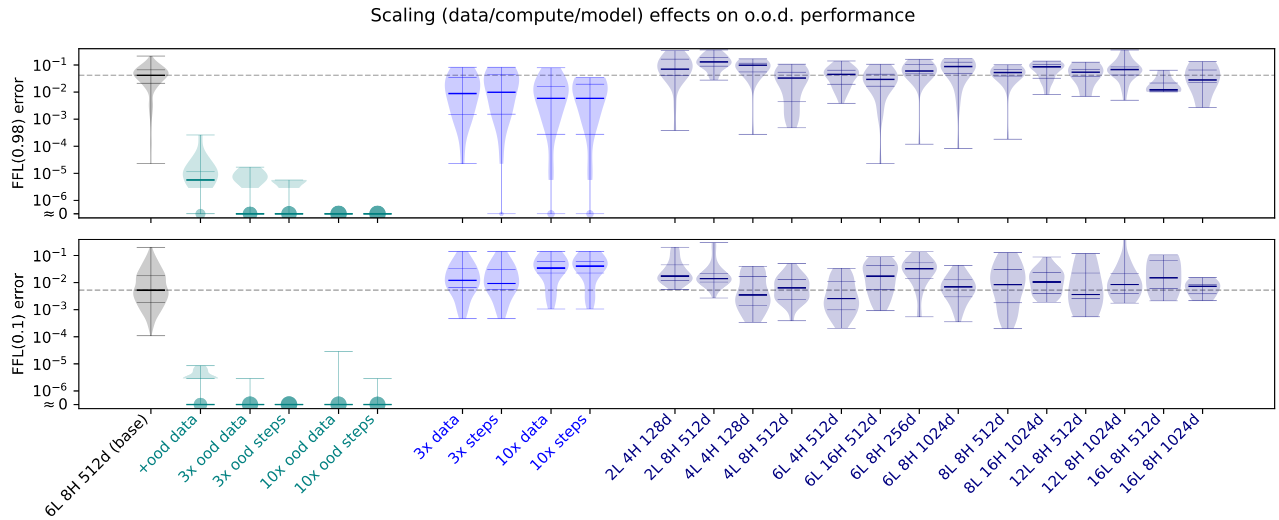


Figure 8: Full comparisons of various scaling axes.

B.3. Benefits of scale (Section 4.1)

In Section 4.1, we discussed mitigations that directly modify the training distributions and resources:

(R4) **Training on rare sequences works best, by a wide margin.**

(R5) **Resource scaling (in-distribution data, training steps, network size) helps.**

We provide more results specifically related to scaling along various axes. As shown in Figure 8, scaling helps improve the OOD performance, especially when more OOD data are introduced. However, the benefit is not clear, especially on dense sequences.

B.4. Indirect algorithmic controls for extrapolation (Section 4.2)

As shown in Figure 3, various architectural, algorithmic and regularization choices can help improve over the baseline Transformer. We recall the main findings:

(R6) **Many algorithmic choices influence extrapolative behavior.**

(R7) **Despite many partial mitigations, nothing eliminates attention glitches entirely.**

There is no clear consensus on the advantages and drawbacks of various positional encodings, but it has been known (Dai et al., 2019) that the choice of positional symmetry-breaking scheme modulates long-sequence performance on natural tasks. We evaluate various choices which appear in high-profile LLMs: sinusoidal, learned, ALiBi (Press et al., 2021), and RoPE (Su et al., 2021). We find that non-trainable position encodings help on dense sequences ($\mathsf{FFL}(0.1)$), but have no clear benefit on sparse ones ($\mathsf{FFL}(0.98)$) which require more handling of long-term dependency.

B.5. Preliminary mechanistic study and challenges

In this section, we move to a simpler setting to gain finer-grained understanding of how sparsity regularization affects the learned solutions. Specifically, we look at the task of *simulating the flip-flip automaton* (Definition 1), whose inputs consist of $\{\sigma_0, \sigma_1, \perp\}$ as two types of `write` and 1 no-op. This task (elaborated in Appendix A.1) can be solved by a 1-layer Transformer with a single attention head which attends sparsely on the most recent `write` position. It also serves as a building block for more complex tasks (Liu et al., 2023), hence observations from this simple setup can potentially be useful in broader contexts.

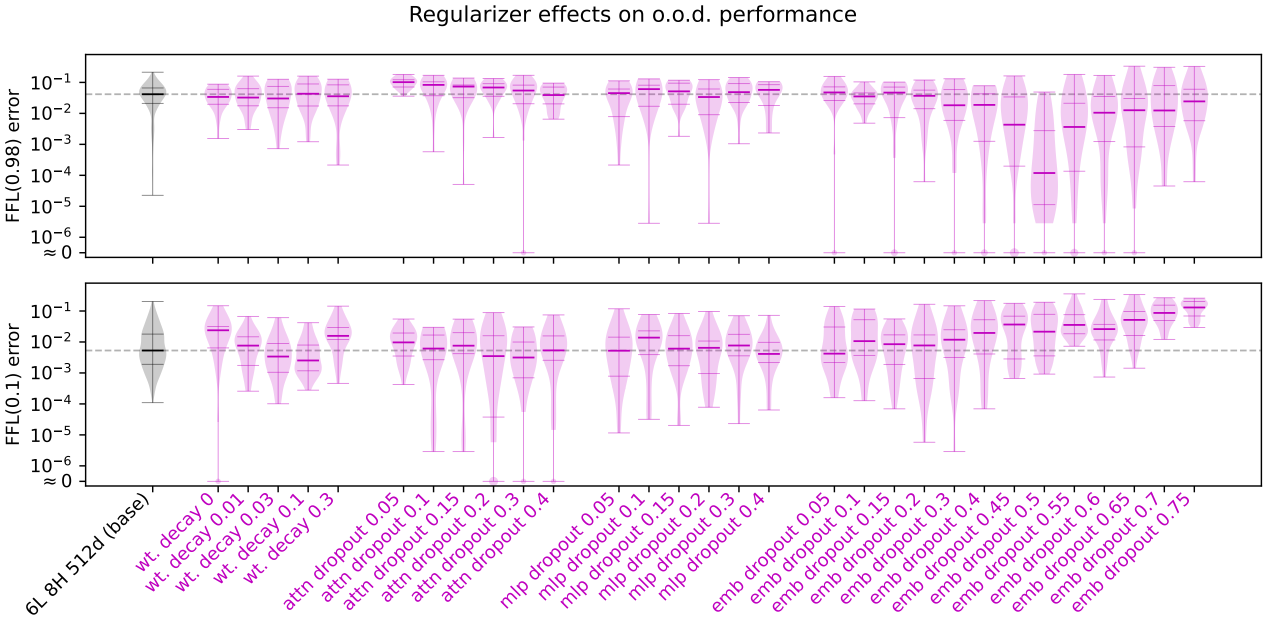


Figure 9: Full comparisons of regularizers.

Figure 10 shows examples of attention patterns on the flip-flop simulation task, subselected from 6-layer 8-head models trained with and without attention-sharpening regularization. It is evident that the attention patterns of the sparse model are less complex and easier to interpret compared to those of the un-regularized model. For example, we can identify one head in the sparse model that exactly coincide with the attention pattern [11] that an "ideal" 1-layer 1-head model implements (Figure 10c).

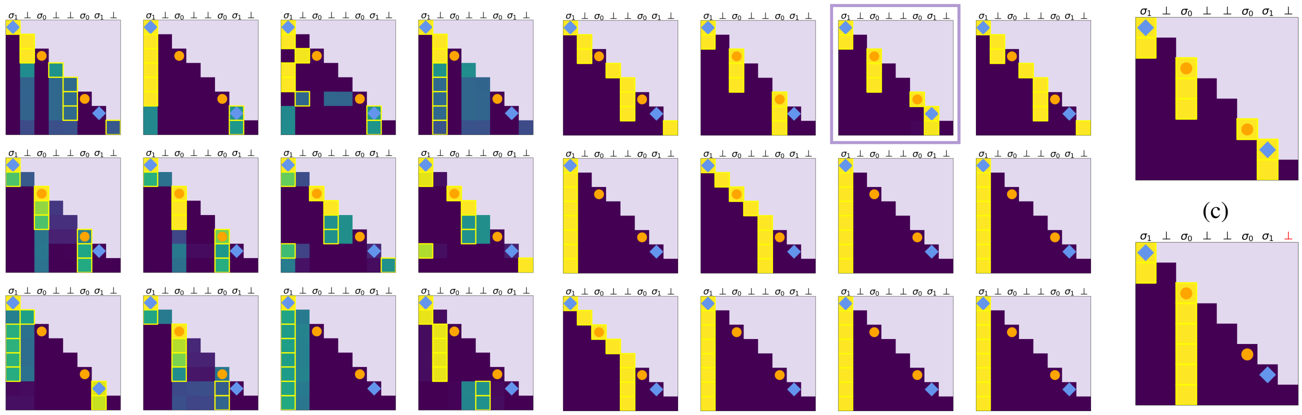
(R8) **Attention-sharpening regularizers successfully promote hard attention, but errors persist.** As mentioned in (R7), attention-sharpening regularization cannot fully eliminate the sporadic errors, which are partially induced by the complexity and redundancy of attention patterns. Moreover, sharpened attention can induce additional failure modes, such as confidently attending to incorrect `write` positions. An example is demonstrated in Figure 10d, where the attention focuses on an initial `write`, likely caused by the fact that earlier positions are overemphasized due to the use of causal attention masks. Another example occurs in length generalization, where the attention is correct at positions earlier in the sequence, but starts to confidently focus on wrong positions as it moves towards later positions (Proposition 4). Details and more discussions are provided in Appendix B.5.

Sparsity regularization helps sharpen the attention Figure 13a,13b compare the attention patterns of 1-layer 1-head models with or without attention-sharpening regularization. While both types of models give correct results, the attention-sharpened model puts all attention weights to the most recently `write` position, which is the solution given according to the definition of the task, whereas the attention patterns of the non-regularized model (Figure 13a) are much less clean.

Are there solutions other than the "ideal" solution? There is a solution naturally associated with the definition of the flip-flop automaton (i.e. the sparse pattern shown in Figure 13b), but it is not necessarily the *only* solution. For example, an equally valid (dense) solution is for the model to attend to every write token of the correct type. This is what the non-regularized (dense) models seems to be implementing, as seen in Figure 13a, except for the final row where the model puts non-negligible amount of weight on a `write` of a different type.

Are attention patterns reliable for interpretability? Prior work has pointed out the limitations of interpretations based solely

[11]While it is well-known that attention patterns can be misleading (Jain & Wallace, 2019; Bolukbasi et al., 2021; Meister et al., 2021) at times, they do provide upper bounds on the magnitude of the dependency among tokens. These upper bounds are particularly useful in the case of (1-)sparse attentions: a (near) zero attention weight signifies the absence of dependency, which greatly reduces the set of possible solutions implemented.



(a) (b) (c) (d)

Figure 10: Causal attention patterns for flip-flop simulation (Definition 1); orange dots / blue diamonds mark the positions of write tokens σ_0 / σ_1. (a),(b) are subselected respectively from a regular (non-sparse) and a sparse multi-layer model (details in Appendix B.5). (c), (d) are from two 1-layer 1-head models. The attention pattern highlighted by the purple box in (b) coincides with the "ideal" attention pattern in (c). However, sparse models can be wrong, as shown in (d) (error marked in red).

on attention patterns (Jain & Wallace, 2019; Bolukbasi et al., 2021). The intuition is that attention patterns can interact with other components of the network in various ways; for example, $\boldsymbol{W}_V$ can project out certain dimensions even though they may have contributed to a large attention score. Hence, for multi-layer multi-head non-sparse models, the magnitude of attention weights may not have an intuitive interpretation of "importance" (Meister et al., 2021). For example, Figure 14 shows examples where the attention on an incorrect token may be higher than that of the correct token. [12] However, in a 1-layer 1-head model, 1-sparse attention as shown in Figure 13b indeed offers interpretability, since if zero attention weight [13] necessarily means the absence of dependency, which greatly reduces the set of possible solutions implemented. As shown in Figure 13c, a `write` may not attend to itself due to the presence of residual link, but the attentions for `read` always focus on the closest `write` as intended.

Sporadic errors persist Section Section 4.1 (R5) showed that none of the mitigations was successful at making Transformers reach 100% accuracy. One common failure mode is long-range dependency, where the input sequences contain very few `writes`. The failure could be attributed to multiple factors; we will explore one aspect related to attention patterns, demonstrated with a 1-layer 1-head Transformers with linear position encoding, on a length-834 sequence with 2 `writes`. As shown in Figure 11, the attentions for positions early in the sequence correctly attend to the most recent `write`. However, attention starts to "drift" as we move to later positions, and the positions at the end of the sequence attend entirely [14] to the recent `read` tokens, which contains no information for solving the task. This may be because the attention weights are affected too much by the position encodings, as discussed in Proposition 4.

Optimization hurdles While sparse solutions may preferred for various reasons, sparsity itself is not sufficient to guarantee good performance: As shown in Figure 13d, sparsity regularization can lead to bad local minima, where the model tends to (incorrectly) rely on earlier positions. This is observed across different types of sparsity regularization. While we do not yet have a full explanation of the phenomenon, a possible explanation for this bias is that earlier positions show up more often during training, due to the use of the causal attention: a valid flip-flop solution is for the model to attend to every `write` token of the correct type; positions earlier in the sequence get included in more subsequences because of the causal mask, and are hence more likely to be attended to. We also observe that the phenomenon seems to be closely related to the training distribution. For example, the model is much more likely to get stuck at a bad local minima when $p(\perp) = 0.5$ (denser sequences) compared to $p(\perp) = 0.9$ (sparse sequences).

[12] However, if we consider the "importance / influence" as measured by the norms of the attetnion-weighted value vectors, then the max norm still corresponds to the correct token, which helps explain why the final output is correct.

[13] By "zero" we mean an attention score on the magnitude of 1e-8 in the experiments.

[14] The attention weights that are not on the most recent `write` sum up to around 1e-7.

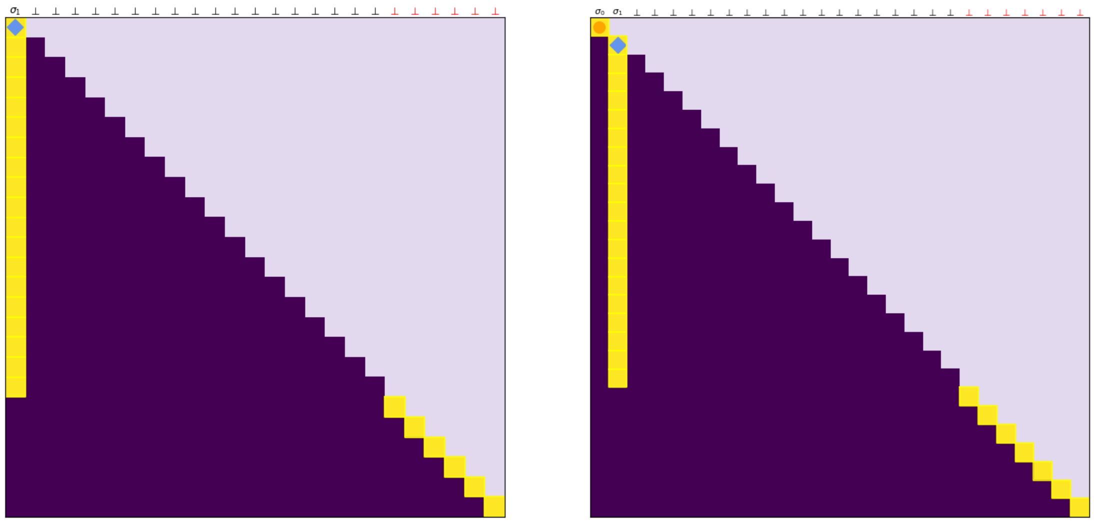


Figure 11: Attention drifts as the length increases. The model is trained on length-500 sequences with $p(\sigma \neq \perp) = 0.5$. The testing sequences are (a) $[2, \underbrace{0 \cdots, 0}_{800}]$, and (b) $[1, \underbrace{0 \cdots, 0}_{32}, 2, \underbrace{0 \cdots, 0}_{800}]$. We sample every 32 positions for visualization.

Effect of sparsity regularization on training dynamics An interesting future direction is to understand the learning dynamics of flip-flop tasks with attention-sharpening regularization, as suggested by the (quantitively and qualitatively) different results and optimization challenges. As some initial empirical evidence that the regularization indeed have a large impact on the dynamics, we found that sharpened attention seems to have a regularization effect on the weight norms (Figure 12), and also lead to different behaviors of the attention heads (Figure 15).

More examples of attention patterns Figure 16 shows the full set of attention patterns of two 6-layer 8-head models trained with and without attention-sharpening regularization, corresponding to Figure 10 (a,b). Attention-sharpening regularization can be applied in different ways; for example, Figure 17 shows results of a model for which only the first layer is regularized. The attention patterns of subsequent layers remain sharpen, even though there is no explicit regularization.

B.6. Software, compute infrastructure, and resource costs

GPU-accelerated training and evaluation pipelines were implemented in PyTorch (Paszke et al., 2017). For the FFLM experiments, we used the `x-transformers`[15] implementations of the Transformer architecture and variants. For the fine-grained mechanistic interpretability experiments on the pure flip-flops, we used the "vanilla, GPT-2"-like Transformer implementation published by HuggingFace (Wolf et al., 2019). We plan to make our benchmarks and training code publicly available.

Each training run was performed on one GPU in an internal cluster, with NVIDIA P40, P100, V100, and RTX A6000 GPUs, with at least 16GB of VRAM. Each (6-layer, 512-dimensional, 8-head) baseline model took ∼10 minutes to train (and evaluate online) for 10^4 steps. A nontrivial fraction of the compute time ($\sim 20\%$) was spent on fine-grained evaluation through the coarse of training. The vast majority of training runs are close to these specifications; consequently, one set of replicates under identical conditions (i.e. each violin plot in each figure) is the product of ∼4 GPU-hours of training time.

We hope that this computational investment will aid in understanding how to build robust Transformer models and training pipelines at much larger scales.

[15] `https://github.com/lucidrains/x-transformers`

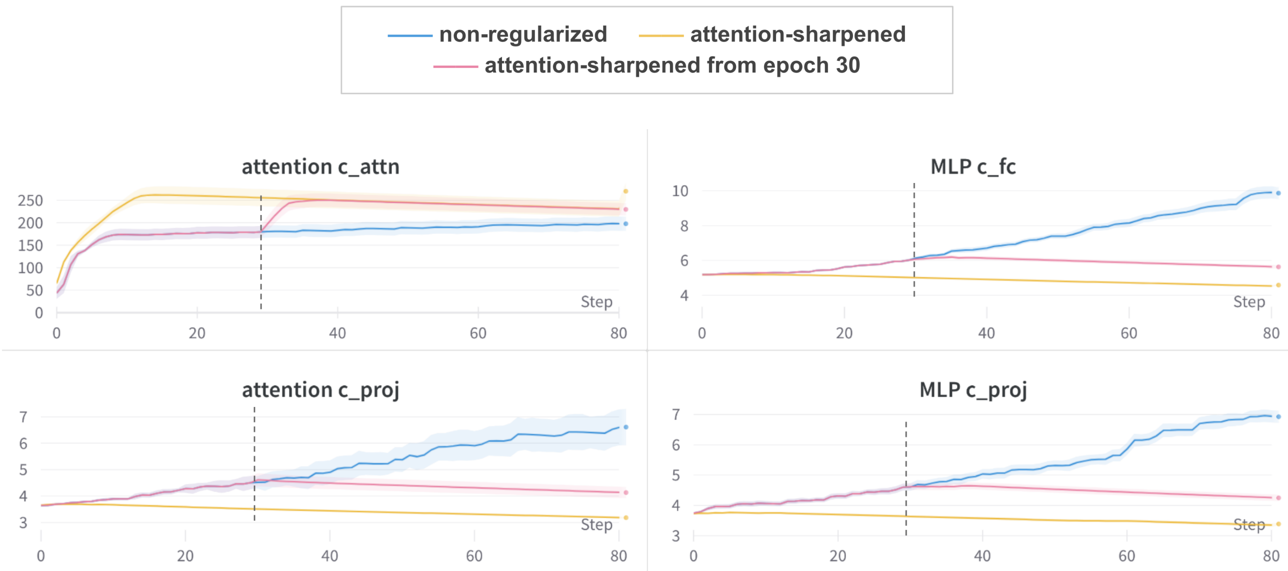
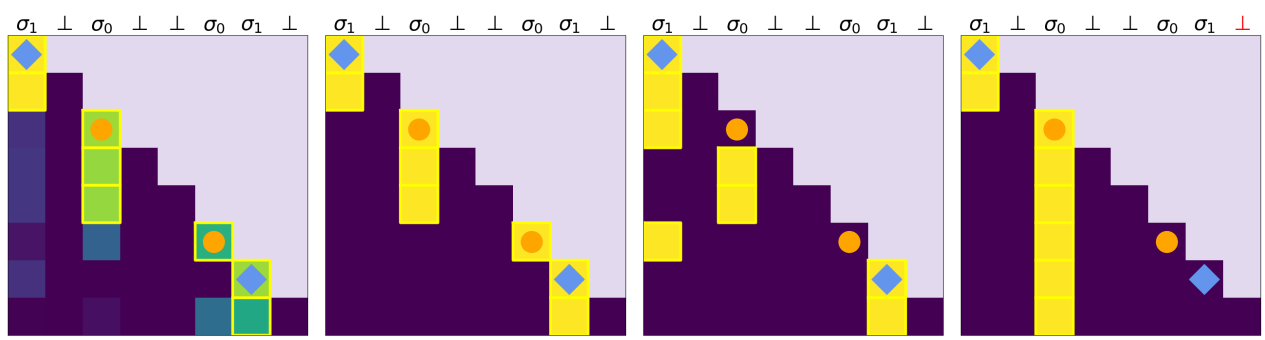


Figure 12: Frobenius norms of weight matrices in 1-layer 1-head models, trained without regularization (blue), with attention-sharpening regularization (yellow), or first without regularization and then adding regularization from epoch 30 (red; epoch 30 marked by the dashed lines). The solid curve and the shadow shows the median and the standard deviation calculated on 8 models.



(a) (b) (c) (d)

Figure 13: Attention-sharpening regularization on 1-layer 1-head models. Compared to a non-regularized model (13a), the sparsity-regularized model (13b) shows clear attention at the last write position. However, sparse attention does not have to align with the "ideal" pattern (13c), and can even be wrong (13d). Positions with yellow borders are where the max attention in each row occur; errors are marked in red.

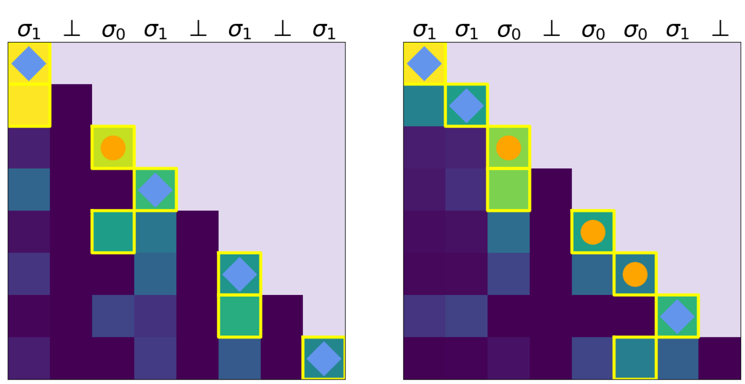
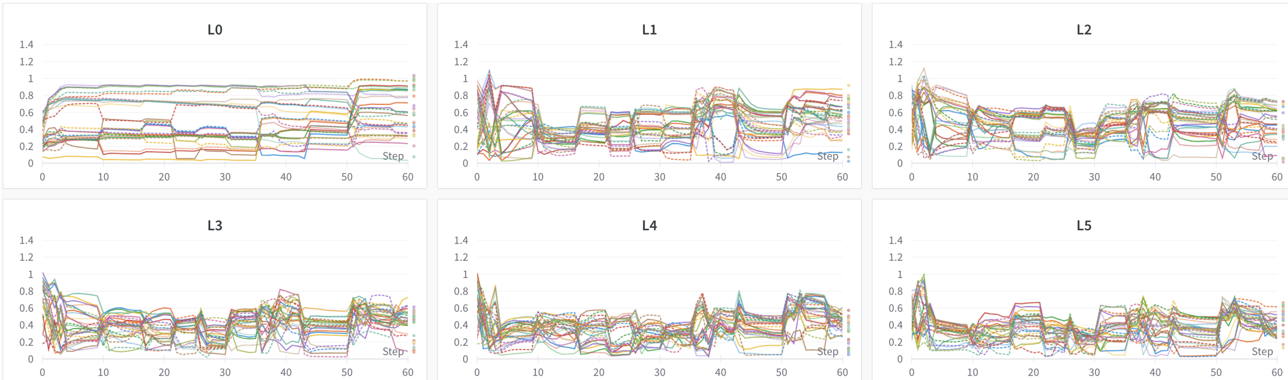
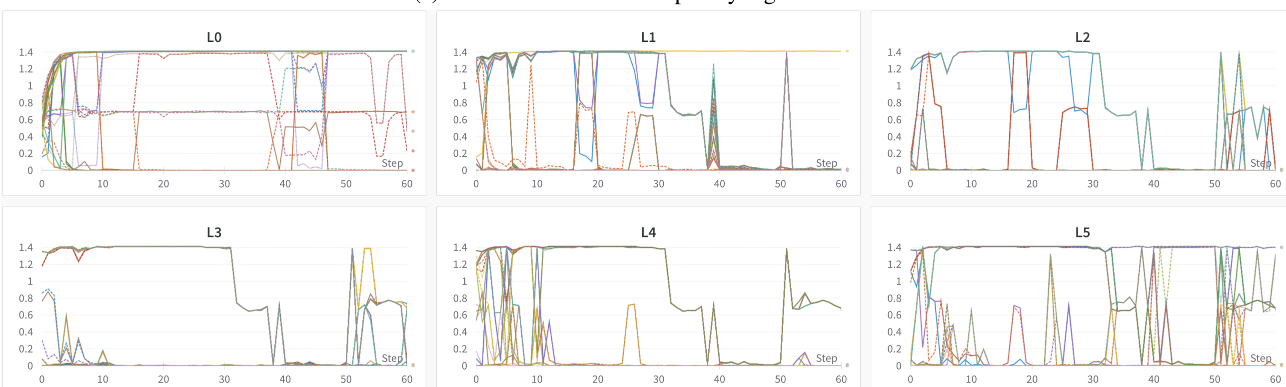


Figure 14: Non-sparse attention pattern can be misleading: a non-sparse model may put more attention on an incorrect token (i.e. a token that is not the `write` with the right type), while making the correct predictions. Yellow boxes mark the position of the max attention of each row.

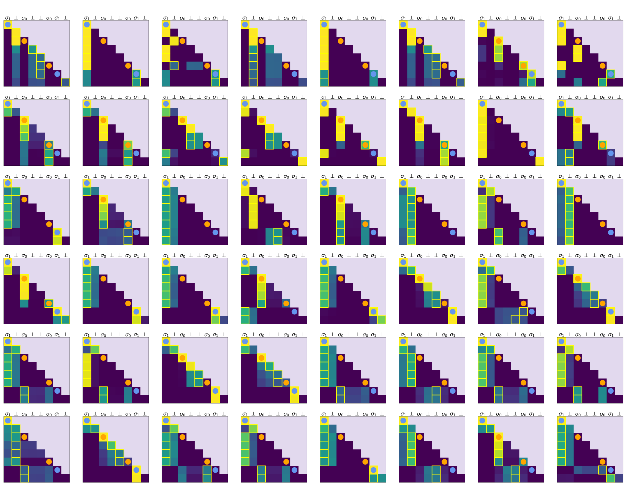


(a) Model trained without sparsity regularization.

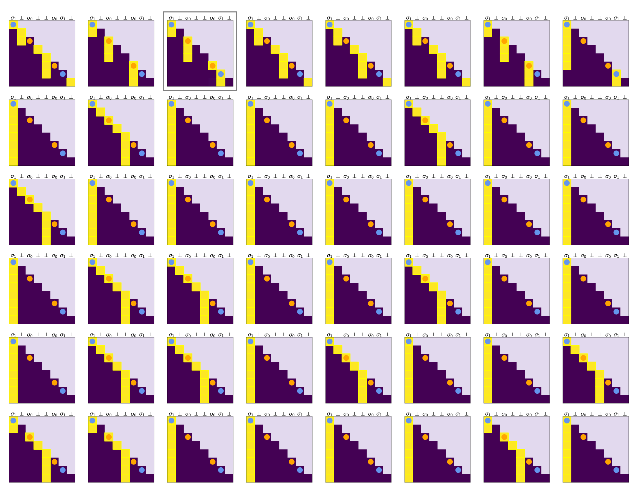


(b) Model trained with entropy sparsity regularization with $\lambda = 0.01$.

Figure 15: Examples of the ℓ_2 difference in attention patterns from two 6-layer 8-head 512-dimension models. Differences are calculated between all pairs of heads in the same layer.

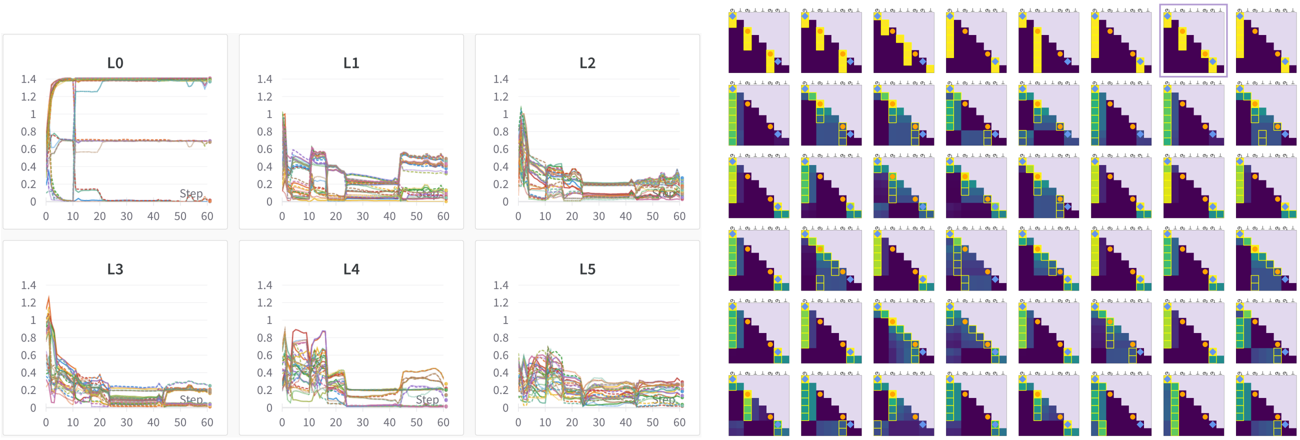


(a) Without regularization.



(b) With attention-sharpening regularization.

Figure 16: Attention patterns for 6-layer 8-head 512-dimension models on the input sequence $[\sigma_1, \perp, \sigma_0, \perp, \perp, \sigma_0, \sigma_1, \perp]$: attention-sharpening regularization lead to cleaner attention patterns. 1 attention head in the first layer of the regularized model (marked by the purple box) matches the "ideal" attention pattern Figure 10c.



(a) ℓ_2 differences between pairs of attention heads in the same layer, throughout training (x-axis).

(b) Attention patterns on the input sequence $[\sigma_1, \perp, \sigma_0, \perp, \perp, \sigma_0, \sigma_1, \perp]$.

Figure 17: Attention heads and attention patterns for a 6-layer 8-head 512-dimension model, trained with attention-sharpening regularization (entropy regularization with strength 0.01) on the first layer only. 1 attention head in the first layer (marked by the purple box) matches the "ideal" attention pattern Figure 10c.

C. Proofs for Appendix A.4

Transformer recap. A Transformer (Vaswani et al., 2017) consists of multiple self-attention layers. Given d-dimensional embeddings of a length-T sequence, denoted as $\boldsymbol{X} \in \mathbb{R}^{T\times d}$, a self-attention layer f computes

$$f(\boldsymbol{X}) = \phi(\boldsymbol{W}_V \mathrm{softmax}(\boldsymbol{X}\boldsymbol{W}_Q\boldsymbol{W}_K^\top \boldsymbol{X}^\top)\boldsymbol{X}\boldsymbol{W}_V\boldsymbol{W}_C). \tag{C.1}$$

where $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{d\times k}$ for $k \le d$ are the query and key matrix; $\boldsymbol{W}_V, \boldsymbol{W}_C^\top \in \mathbb{R}^{d\times k}$ project the representations from and back to $\mathbb{R}^d$. softmax calculates row-wise softmax. $\phi: \mathbb{R}^d \to \mathbb{R}^d$ is a 2-layer fully-connected network. Residual links and layer norm can be optionally included at different places of a self-attention layer.

C.1. Realizability of FFL by small Transformers

Proposition 2. *A 2-layer 1-head Transformer with residual connections can represent "deterministic" FFL.*

Proof. Let us consider predicting in the deterministic mode (Section 2). Then we need to predict x_{t+1} given $x_{1:t}$ with $x_t = \mathtt{r}$. In order to do this, we need to find the largest $\tau < t$ such that $x_\tau = \mathtt{w}$ and output $x_{\tau+1}$. There are multiple ways to implement this, we will consider the following: (1) layer 1 converts FFL to the flip-flop automaton (Definition 1), (2) layer 2 implements the flip-flop construction. For layer 2, we can use the construction described in (Liu et al., 2023). Here we present the full construction for completeness.

We will consider a two-layer Transformer with one head in each layer followed by a 2-layer MLP and a residual connection. In particular, for $x \in \{\mathtt{w}, \mathtt{r}, \mathtt{i}, 0, 1\}^T$:

$$f(x) = \phi_2(\boldsymbol{W}_V^{(2)} \mathrm{softmax}(f_1(x)\boldsymbol{W}_Q^{(2)}{\boldsymbol{W}_K^{(2)}}^\top f_1(x)^\top) f_1(x)\boldsymbol{W}_V^{(2)}\boldsymbol{W}_C^{(2)})$$
$$\text{where } f_1(x) = E(x) + \phi_1(\boldsymbol{W}_V^{(1)} \mathrm{softmax}(E(x)\boldsymbol{W}_Q^{(1)}{\boldsymbol{W}_K^{(1)}}^\top E(x)^\top) E(x)\boldsymbol{W}_V^{(1)}\boldsymbol{W}_C^{(1)})$$

where $E(x) \in \mathbb{R}^{T\times d}$ is the encoding for the input sequence x given some encoding function E.

Our construction is as follows:

- Select $d = 7, k = 2, H = 1$ (recall from Equation C.1that d, k are the dimensions of $\boldsymbol{W}_Q, \boldsymbol{W}_K$). Among the $d = 7$ embedding dimension, two dimensions are for the operations ($\mathtt{w}$ versus $\mathtt{r}$, $\mathtt{i}$), two for the two `write` values, one for the positional embedding, one for padding, and the final dimension is for storing whether the previous position is the most recent `write`, as calculated by the first layer.

- Select input symbol encodings such that for the token at position t, denoted as x_t,

$$E(x_t) := \mathbb{1}[x_t = \mathtt{w}]e_1 + \mathbb{1}[x_t = \mathtt{r} \vee x_t = \mathtt{i}]e_2 + \mathbb{1}[x_t = 0]e_3 + \mathbb{1}[x_t = 1]e_4 + e_5 + P_t \in \mathbb{R}^7,$$

  where P_t is the positional encoding. We use the linear positional encoding $P_t := (t/C) \cdot e_6$, for some (large) constant C. For a fixed sequence length T, we can set $C = T$.

- $\boldsymbol{W}_Q^{(1)} := \begin{bmatrix} e_5 & e_5 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, $\boldsymbol{W}_K^{(1)} := \begin{bmatrix} 3c\frac{e_1}{2T} & ce_6 \end{bmatrix} \in \mathbb{R}^{7\times 2}$ for $c = O(T\log(T))$, $\boldsymbol{W}_V^{(1)} := \begin{bmatrix} e_1 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, and $\boldsymbol{W}_C^{(1)\top} := \begin{bmatrix} e_7 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$.

- $\boldsymbol{W}_Q^{(2)} := \begin{bmatrix} e_5 & e_5 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, $\boldsymbol{W}_K^{(2)} := \begin{bmatrix} ce_7 & ce_6 \end{bmatrix} \in \mathbb{R}^{7\times 2}$ for $c = O(T\log(T))$, $\boldsymbol{W}_V^{(2)} := \begin{bmatrix} e_4 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$, and $\boldsymbol{W}_C^{(2^\top} := \begin{bmatrix} e_1 & 0 \end{bmatrix} \in \mathbb{R}^{7\times 2}$.

In layer 1, the unnormalized attention score for query position i to key position j is

$$\left\langle \boldsymbol{W}_Q^{(1)\top} x_i, \boldsymbol{W}_K^{(1)\top} x_j \right\rangle = \left\langle \frac{c}{T} \cdot \left[\frac{3}{2} \cdot \mathbb{1}[x_j = \mathtt{w}], j\right], [1, 1] \right\rangle = \frac{c}{T} \cdot \left(\frac{3}{2}\mathbb{1}[x_j = \mathtt{w}] + j\right).$$

Note that the max attention value for position i is achieved at i if $x_{i-1} \neq \mathtt{w}$, else the max is achieved at position $i - 1$.

In the setting of hard attention, the output for the i_{th} token after the attention module is $\mathbb{1}[x_{i-1} = \mathtt{w} \vee x_i = \mathtt{w}]e_7$. Now similar to the constructions in (Liu et al., 2023) (Lemma 6), with a appropriate choice of $c = O(T \log T)$, we can approximate hard attention by soft attention, and subsequently use the MLP to round the coordinate corresponding to e_7. The MLP otherwise serves as the identity function. Together with the residual link, the first layer output (i.e. the second layer input) at position i takes the form

$$f_1(x_i) = E(x_i) + \mathbb{1}[x_{i-1} = \mathtt{w} \vee x_i = \mathtt{w}]e_7.$$

In layer 2, the unnormalized attention score computed for position i attending to j is

$$\begin{aligned}\left\langle \boldsymbol{W}_Q^{(2)\top} f_1(x_i), \boldsymbol{W}_K^{(2)\top} f_1(x_j) \right\rangle =& \frac{c}{T} \left\langle [1, 1], \left[\mathbb{1}[x_{j-1} = \mathtt{w} \vee x_j = \mathtt{w}], \frac{j}{T}\right] \right\rangle \\ =& c \cdot \left(\mathbb{1}[x_{j-1} = \mathtt{w} \vee x_j = \mathtt{w}] + \frac{j}{T}\right).\end{aligned}$$

Note that the max attention value is achieved at the position right after the closest $\mathtt{w}$ to x_i. Let us denote this position by $\tau \leq i$, then with hard attention, the output at the i_{th} position is $x_\tau e_1$, as desired. Now similar to before, we can approximate this with soft attention and use the MLP to do the appropriate rounding to get our final construction. □

Remark: The construction in Proposition 2 is *a* construction, but it is not the *only* construction. For example, for the second layer implementation for the flip-flop automaton, there could be an equally valid *dense* solution, where the model uniformly attends to all `write` tokens of the correct type.

C.2. Failure of soft attention: attention dilution with bounded Lipschitzness

Consider any attention layer with weight matrices $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{k\times d}$. If $\|\boldsymbol{W}_K^\top \boldsymbol{W}_Q\|_2$ is bounded, then the attention cannot be sparse as the sequence length increases:

Proposition 3 (Leaky soft attention)**.** *Assume the latent variables have bounded norm, i.e.* $\|\boldsymbol{v}\|_2 \leq 1$ *for any latent vector* $\boldsymbol{v} \in \mathbb{R}^d$*, and let* $\sigma_{\max}$ *denote the max singular value of* $\boldsymbol{W}_K^\top \boldsymbol{W}_Q$*. Then for* $T = \Omega(\exp(2\sigma_{\max}))$*, any sequences of latent vectors* $\{\boldsymbol{v}_\tau\}_{\tau\in[T]}$, $\|\mathrm{softmax}(\{\boldsymbol{v}_\tau\}_{\tau\in[T]})\|_\infty = 1 - \Omega(1)$.

Proof. The proof follows directly from a simple rewriting.

For any $\boldsymbol{u}, \boldsymbol{v}$ with $\|\boldsymbol{u}\|_2, \|\boldsymbol{v}\|_2 \leq 1$, the pre-softmax attention score is bounded by $\boldsymbol{u}^\top \boldsymbol{W}_K^\top \boldsymbol{W}_Q \boldsymbol{v} \in [-\sigma_{\max}, \sigma_{\max}]$.

$$\frac{\exp(\boldsymbol{v}_t^\top \boldsymbol{W}_K^\top \boldsymbol{W}_Q \boldsymbol{v}_T)}{\sum_{\tau \in [T]} \exp(\boldsymbol{v}_\tau^\top \boldsymbol{W}_K^\top \boldsymbol{W}_Q \boldsymbol{v}_T)} \leq \frac{\exp(\sigma_{\max})}{\exp(\sigma_{\max}) + (T-1)\exp(-\sigma_{\max})} = 1 - \frac{T-1}{T-1+\exp(2\sigma_{\max})},$$

where the last term is $\Omega(1)$ when $T = \Omega(\exp(2\sigma))$. □

Attention dilution and failure on dense sequences Strictly speaking, attention dilution caused by an increased sequence length does not necessarily affect the output of the layer. For example, if `ignore` gets mapped to a subspace orthogonal to that of `write`, then $\boldsymbol{W}_V$ can project out the `ignore` subspace, making the weighted averaged depending only on the number of `write`s. Hence with the presence of layer norm, attention dilution won't be a problem for the final prediction if the number of `write` is upper bounded regardless of the sequence length.

Moreover, for the experiments in Section 4.1, denser sequences (i.e. larger $p(\texttt{write})$) does increase the number of `write` compared to the training distribution, hence attention dilution can be a potential cause for the decrease in performance.

C.3. Failure of hard attention: bad margin for positional embeddings

In this section, we look at a failure mode that a 1-layer 1-head Transformer has on the flip-flop automaton simulation task. Why do we care about this setup? Simulating the automaton is in fact a sub-task of FFLM. For example, the second layer of the construction in Proposition 2 reduces to the simulation task.

Consider a 1-layer 1-head Transformer with parameters $\boldsymbol{W}_Q, \boldsymbol{W}_K \in \mathbb{R}^{k\times d}$. Write the attention query matrix $\boldsymbol{W}_Q$ as $\boldsymbol{W}_Q = [\boldsymbol{W}_{Qe}, \boldsymbol{W}_{Qp}]$, where $\boldsymbol{W}_{Qe} \in \mathbb{R}^{k\times(d-1)}$ corresponds to the embedding dimensions, and $\boldsymbol{W}_{Qp}\mathbb{R}^k$ corresponds to the dimension for the linear positional encoding. Write $\boldsymbol{W}_K = [\boldsymbol{W}_{Ke}, \boldsymbol{W}_{Kp}]$ similarly.

Then, we claim that the following must be true, regardless of the choice of the token embedding:

Proposition 4. *Consider linear positional encoding, i.e. $p_i = i/C$ for some (large) constant C. Then, perfect length generalization to arbitrary length requires $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} = 0$.*

Proof. Let $\boldsymbol{e}^{(i)} \in \mathbb{R}^{d-1}$ denote the embedding vector (without the position encoding) for token $i \in \{0,1,2\}$. Let $\boldsymbol{v}_t = [\boldsymbol{e}_t, p_t]^\top \in \mathbb{R}^d$ denote the embedding for the t_{th} token, where $\boldsymbol{e}_t \in \{\boldsymbol{e}^{(0)}, \boldsymbol{e}^{(1)}, \boldsymbol{e}^{(2)}\}\mathbb{R}^d$ is the embedding of the token itself, and $p_t := i/C$ is the linear positional encoding.

Let $s_{i\to j}$ denote the pre-softmax attention score that the i_{th} token puts on the j_{th} token, which is given by

$$\begin{aligned} s_{i\to j} &= \langle \boldsymbol{W}_Q \boldsymbol{v}_i, \boldsymbol{W}_K \boldsymbol{v}_j \rangle && \text{(C.2)} \\ &= \boldsymbol{e}_i^\top \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}_j + \boldsymbol{e}_i^\top \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp} \cdot p_j + (\boldsymbol{e}_j)^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp} \cdot p_i + \boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \cdot p_i p_j && \text{(C.3)} \\ &= \boldsymbol{e}_i^\top \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}_j + \frac{\boldsymbol{e}_i^\top \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp}}{C} \cdot j + \frac{(\boldsymbol{e}_j)^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} \cdot i + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot ij. && \text{(C.4)} \end{aligned}$$

We will prove the proposition in two cases, which respectively require $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \leq 0$ and $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \geq 0$.

Case 1: $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \leq 0$ required Consider the case of long-term dependency, where the input sequence consists of an initial write and a series of reads, i.e. $\sigma_1 = 1$ and $\sigma_t = 0$ for $t > 1$. Then for the T_{th} position, the score for the first write token is

$$\begin{aligned} s_{T\to 1} &= \langle \boldsymbol{W}_Q \boldsymbol{v}_T, \boldsymbol{W}_K \boldsymbol{v}_1 \rangle && \text{(C.5)} \\ &= \boldsymbol{e}^{(0)\top} \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}^{(1)} + \frac{\boldsymbol{e}^{(0)\top} \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp}}{C} + \frac{(\boldsymbol{e}^{(1)})^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} \cdot T + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T && \text{(C.6)} \\ &= \left(\frac{(\boldsymbol{e}^{(1)})^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \right) \cdot T + O(1) = O(T), && \text{(C.7)} \end{aligned}$$

and the score for the last write token is

$$\begin{aligned} s_{T\to T} &= \langle \boldsymbol{W}_Q \boldsymbol{v}_T, \boldsymbol{W}_K \boldsymbol{v}_T \rangle && \text{(C.8)} \\ &= {\boldsymbol{e}^{(0)}}^\top \boldsymbol{W}_{Qe} \boldsymbol{W}_{Ke} \boldsymbol{e}^{(0)} + \frac{{\boldsymbol{e}^{(0)}}^\top \boldsymbol{W}_{Qe}^\top \boldsymbol{W}_{Kp}}{C} T + \frac{{\boldsymbol{e}^{(0)}}^\top \boldsymbol{W}_{Ke} \boldsymbol{W}_{Qp}}{C} \cdot T + \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T^2 && \text{(C.9)} \\ &= \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T^2 + O(T). && \text{(C.10)} \end{aligned}$$

Think of C as going to infinity. If $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} > 0$, then there exists a sufficiently large T such that $s_{T\to T} > s_{T\to 1}$. Hence we need $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \leq 0$.

Case 2: $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \geq 0$ required Consider the input sequence where $\sigma_1 = 1$, $\sigma_{T-1} = 2$, and $\sigma_t = 0$ for $t \in [T] \setminus \{1, T-1\}$. Similar to the above, calculate the pre-softmax attention scores for σ_1, σ_{T-1} as

$$s_{T\to 1} = O(T) \quad \text{(C.11)}$$

$$s_{T\to T-1} = \frac{\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp}}{C^2} \cdot T^2 + O(T). \quad \text{(C.12)}$$

Since we need $s_{T\to T-1} > s_{T\to 1}$, it must be that $\boldsymbol{W}_{Qp}^\top \boldsymbol{W}_{Kp} \geq 0$.

$\square$