CAN MAMBA LEARN HOW TO LEARN? A COMPARATIVE STUDY ON IN-CONTEXT LEARNING TASKS

Jongho Park[1], **Jaeseung Park**[2]*, **Zheyang Xiong**[3], **Nayoung Lee**[3], **Jaewoong Cho**[1], **Samet Oymak**[4], **Kangwook Lee**[1,3], **Dimitris Papailiopoulos**[1,3]

[1] KRAFTON, [2] Seoul National University,
[3] University of Wisconsin-Madison, [4] University of Michigan, Ann Arbor

ABSTRACT

State-space models (SSMs), such as Mamba (Gu & Dao, 2023), have been proposed as alternatives to Transformer networks in language modeling, by incorporating gating, convolutions, and input-dependent token selection to mitigate the quadratic cost of multi-head attention. Although SSMs exhibit competitive performance, their in-context learning (ICL) capabilities, a remarkable emergent property of modern language models that enables task execution without parameter optimization, remain underexplored compared to Transformers. In this study, we evaluate the ICL performance of SSMs, focusing on Mamba, against Transformer models across various tasks. Our results show that SSMs perform comparably to Transformers in standard regression ICL tasks, while outperforming them in tasks like sparse parity learning. However, SSMs fall short in tasks involving non-standard retrieval functionality. To address these limitations, we introduce a hybrid model, MambaFormer, that combines Mamba with attention blocks, surpassing individual models in tasks where they struggle independently. Our findings suggest that hybrid architectures offer promising avenues for enhancing ICL in language models.

1 INTRODUCTION

Meta-learning, or "learning to learn," has been extensively studied Schmidhuber et al. (1997); Ravi & Larochelle (2016) and recently regained interest in the context of ICL, particularly concerning Transformer models Vaswani et al. (2017). Garg et al. (2022), for example, proposed various ICL tasks, such as learning linear regression, and evaluated the ability of transformers to perform them when specifically trained to do so. On the other hand, Min et al. (2022) studied fine-tuning language models to explicitly learn and perform ICL. Following these footsteps, numerous research studies have been dedicated to understanding the mechanics of Attention that enable such meta-learning capabilities, either through constructive arguments or extensive experimental investigation Akyürek et al. (2022); Li et al. (2023b); von Oswald et al. (2023b); Bai et al. (2023); Yang et al. (2023a); Li et al. (2023a); von Oswald et al. (2023a).

As Transformer language models are currently the only large models that have been reported to be capable of ICL in practice, this raises the question:

Can attention-free models perform ICL?

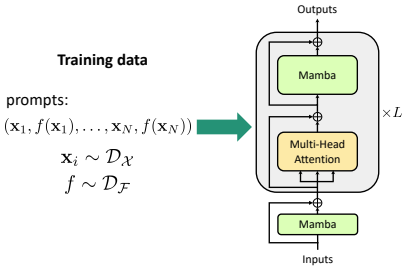
This question holds merit, especially considering that several recent studies have attempted to move beyond attention-based networks due to their quadratic cost Gu et al. (2022c); Dao et al. (2022); Gu & Dao (2023); Poli et al. (2023); Peng et al. (2023); Sun et al. (2023); Yang et al. (2023b). In this work, we focus specifically on state-space models (SSMs), and particularly Mamba (Gu & Dao, 2023). Mamba was recently demonstrated to be highly efficient while achieving near state-of-the-art performance in standard pretraining language data sets, such as the Pile (Gao et al., 2020), but at

*This work was done during an internship at KRAFTON.
Email: <jongho.park@krafton.com>. Correspondence: <dimitris@papail.io>

smaller model scales (*e.g.*, up to 3 billion parameters), surpassing transformers and other attention-free architectures across various language and non-language tasks. However, ICL capabilities usually emerge at scales beyond 3 billion parameters. As a result, the potential of these attention-free models to perform ICL remains underexplored, as testing such hypotheses usually requires scaling beyond the 7 billion parameter level. Nonetheless, we can still investigate small-scale ICL capabilities by specifically training a model to perform in-context learning, following the approach of Garg et al. (2022).

	Transformer	Mamba	MambaFormer
Linear Regression	✓	✓	✓
Sparse Linear Regression	✓	✓	✓
2NN Regression	✓	✓	✓
Decision Tree	✓	▲	✓
Orthogonal-outlier Regression	✓	▲	✓
Many-outlier Regression	▲	✓	✓
Sparse Parity	✗	✓	✓
Chain-of-Thought I/O	✓	✓	✓
Vector-valued MQAR	✓	✗	✓

(a) Model performances on various ICL tasks. We label the model's performance with ✓ if the model performs on par with other baseline models, ✗ if the model struggles to learn the task, and ▲ if the performance improves but with a performance gap compared to other baseline models. Transformer fails in learning sparse parity, showing performance no better than random guessing, while Mamba suffers to accurately retrieve the value vector in vector-valued MQAR. Our proposed MambaFormer performs on par with other baseline models in all tasks.



(b) MambaFormer is a hybrid architecture that replaces MLP blocks within the transformer with Mamba blocks. Importantly, the architecture starts with a Mamba block and does not use positional encoding. In our ICL evaluations, we find that MambaFormer consistently achieves a best-of-both-worlds performance when compared to Transformer and Mamba.

Contributions. In this study, we introduce a diverse set of ICL tasks to evaluate the performance of Transformer and various SSMs, including state-of-the-art models like Mamba and S4 Gu et al. (2022b). Our findings reveal that most of these SSMs can effectively perform ICL, matching the performance of Transformers across multiple tasks. However, Mamba demonstrates some limitations in learning decision trees and retrieval tasks (as also noted in Arora et al. (2023)), but can outperform Transformers in other complex ICL tasks, such as sparse parity, where Transformer models struggle.

Since there seem to be tasks where either family of models is better, we explore the impact of interleaving SSM blocks with multi-head attention blocks, similar to Gu & Dao (2023). We introduce MambaFormer, a novel hybrid architecture that integrates Mamba and Attention layers, while eliminating the need for positional encodings, as shown in Fig. 1b. MambaFormer seems to leverage the strengths of both Mamba and Transformers, exhibiting good performance across all evaluated ICL tasks and simultaneously learning sparse parity and retrieval.

We believe that our findings underscore the importance of broadening the understanding of ICL beyond Transformers, as significant progress has been made in the context of attention-free models.

2 In-context Learning Tasks

We provide a brief outline of the ICL and related tasks investigated in this study. Some tasks are adapted from Garg et al. (2022), and we follow the settings outlined in their work. We only include a table summarizing the ICL tasks here. For a detailed description of each task, refer to A.3

3 In-context Learning Capabilities of Mamba

In this section, we demonstrate that Mamba can be trained from scratch to perform various ICL tasks. Furthermore, we identify specific tasks in which one model performs better than others and vice versa, given the same amount of computation resources measured in terms of its total floating point operations (FLOPs) used in training.

Task	dim (d)	points (N)	Example/Function Sampling	Task-specific
Linear regression	20	41	$\mathbf{x}, \mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_d)$	–
Sparse Linear regression	20	101	$\mathbf{x}, \mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_d)$, $\texttt{sparsity}(\mathbf{w}) \leftarrow k$	$k = 3$
2NN regression	20	101	$\mathbf{W}_{ij}^{(1)}, \mathbf{W}_{ij}^{(2)} \sim \mathcal{N}(0, 1)$	–
Decision Tree	20	101	$\mathbf{x}, \text{Leaf} \sim \mathcal{N}(0, 1), \text{non\_leaf} \sim \{1, ..., d\}$	$\text{depth} = 4$
Orthogonal-outlier regression	20	101	$\mathbf{x}, \mathbf{w} \sim \mathcal{N}(0, \mathbf{I}_d)$, $\mathbf{u}, \mathbf{v} \sim \mathbf{w}^{\perp}$	$p = 0.5$
Many-outlier regression	20	512	$\mathbf{x} \sim \mathcal{N}(0, I)$ w.p. $1 - p$, else $(\mathbf{x}, y) = (\mathbf{1}, 1)$	$p = 0.9$
Sparse Parity	10	140	$\mathbf{x} \sim \{-1, 1\}^d, y = \prod_{j \in I} \mathbf{x}[j]$	$k = 2$
Chain-of-Thought I/O	10	101	$\mathbf{x} \sim \mathcal{N}(0, \mathbf{I}_d)$, $\mathbf{W}_{ij} \sim \mathcal{N}(0, 2/k)$, $\mathbf{v} \sim \mathcal{N}(0, \mathbf{I}_k)$	$h = 8$
Vector MQAR	20	128	$\mathbf{k}, \mathbf{v} \sim \text{Unif}(\mathcal{S}^{d-1})$	32 k-v pairs

Table 1: Summary of Tasks. All models are trained for 500,000 iterations (except for the vector MQAR; see A.3.7).

3.1 MAMBA *can* IN-CONTEXT LEARN!

In figure 2, Mamba consistently outperforms its simpler counterparts S4-Mamba and S4. For linear regression, the gap between Mamba and S4-Mamba is much smaller than that between S4-Mamba and S4. As the only difference between Mamba and S4-Mamba is the input-dependent selection mechanism, appropriate gating and stacking of MLPs (*i.e.*, difference between S4-Mamba and S4) seem to be more significant for such tasks. In comparison, the input-dependence of Mamba makes meaningful progress for more complex tasks such as 2NN regression and learning decision trees.

Mamba can also perform on par with Transformer even as the total FLOPs scale up. This is surprising given that Transformer and attention have been the focus of many previous works for its unique ICL capability. Moreover, Mamba tends to perform better in smaller parameter settings when controlling for equal depth, *i.e.*, keeping the number of attention, MLP, and Mamba blocks equivalent.

3.2 LEARNING PARITY AND RETRIEVAL

Challenges in Learning Parity and Retrieval From table 3b, we can see that Mamba struggles to accurately retrieve the vectors as the mean squared error for retrieving normalized vectors are greater than 0.1 in all cases. Since SSMs are limited by their hidden state dimension in carrying information to predict the next token, they would eventually be overwhelmed if the number of key-value pairs within the context (not queries) increases substantially.

While Mamba fails on simple retrieval tasks such as MQAR (table 3b), the tables turn for the task of learning sparse parity. Transformer fails to do better than random guessing, in line with the empirical evidence of Bhattamishra et al. (2023). We confirm this for Transformer sizes of embedding dimensions up to 768 and 24 layers, trained for 1M iterations (refer to A.3.5 for median convergence rate of each model). However, Mamba succeeds in this task with ease, solving sparse parity for $(d, k) = (10, 2)$ with a network as small as 2 layers. Even more surprisingly, S4-Mamba solves sparse parity as well; this may mean that proper convolution or gating may be more important than input-dependent selection.

We perform an ablation study by equipping Transformer with an initial Mamba block without any positional encoding. Furthermore, we test MQAR on MambaFormer, which will be explained in the subsequent section. For more details, refer to A.3.8

3.3 ALL-IN-ONE ICL PERFORMANCE

While MambaFormer succeeds in two tasks that were deemed difficult for either Mamba or Transformer, it also performs equally well as Transformer and Mamba do in the rest the ICL tasks. In figure 8, we see that MambaFormer and Standard Hybrid both learn decision trees as well as Transformer does and better than Mamba, even at larger parameter sizes.

More surprisingly, MambaFormer efficiently learns linear regression more robustly even in the presence of noisy data in Many-outlier regression and Orthogonal-outlier regression (see figure 4). In particular, a small MambaFormer trained on 100k iterations ($< 10^{17}$ FLOPs) performs as well as models trained with nearly 5 times the number of FLOPs (figure 4 left).

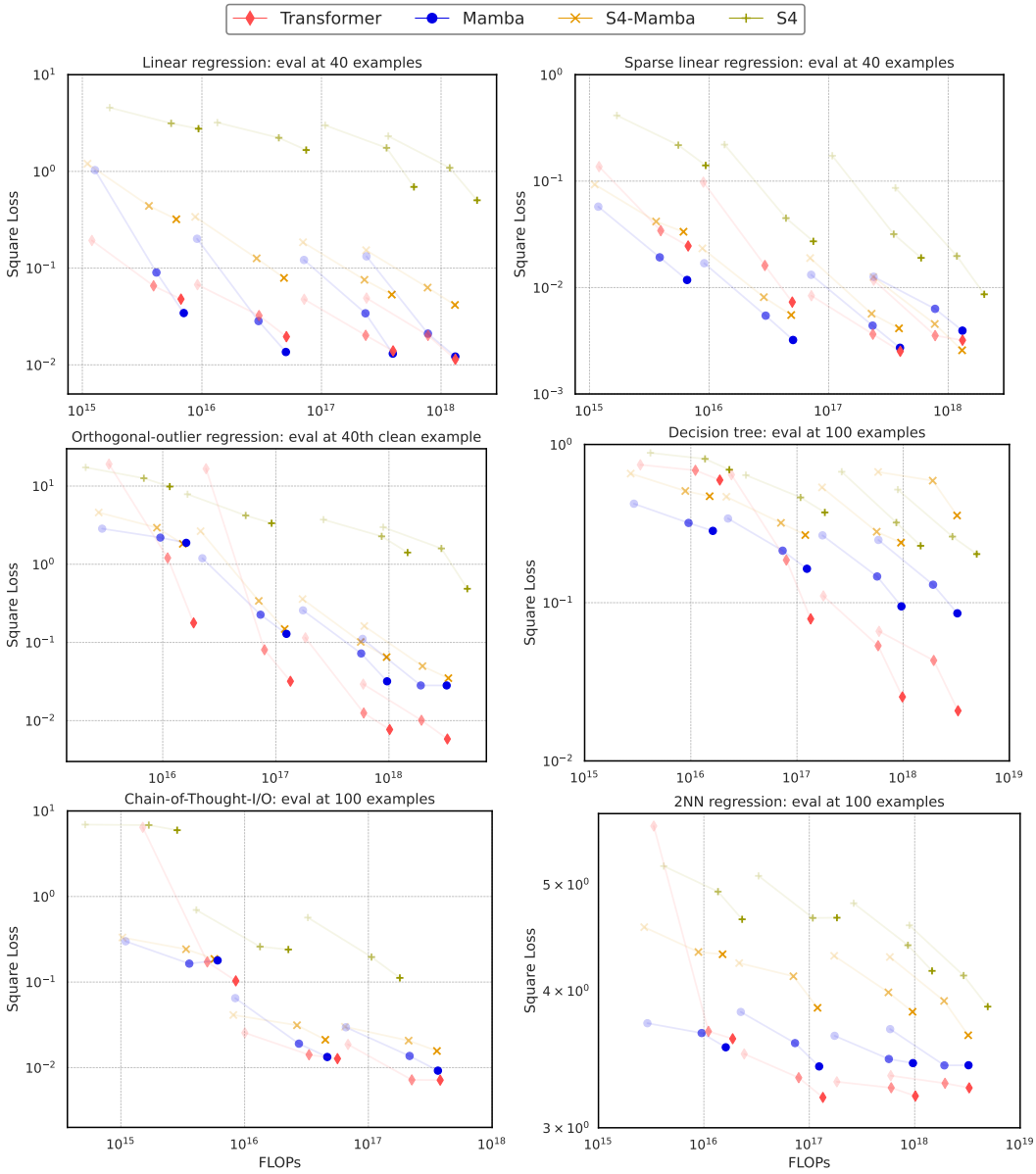
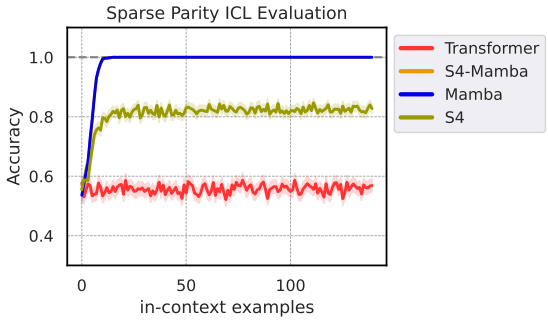


Figure 2: Model performance on our suite of ICL tasks for Transformer, Mamba, S4-Mamba, and S4 where each color represents a different architecture. For each architecture, the best performing model given the same amount of FLOPs is plotted (see A.2 for details on model configurations). Transparent points indicate earlier stages of training; plotted models are trained in between {100k, 300k, 500k} iterations. The descriptions of tasks can be found in A.3.

When evaluated with no outliers during test-time, MambaFormer resembles Transformer and Standard Hybrid resembles Mamba in terms of its out-of-distribution performance, where Mamba easily learns linear regression when there is only one outlier vector (figure 4 top right) while Transformer learns better when there is a subspace of outlier vectors (figure 4 bottom right).

In conclusion, we find the best of both worlds within our diverse array of ICL tasks; a hybrid architecture that can solve as difficult problems as retrieval and parity, while performing on par with Transformer and Mamba in other ICL tasks.

(a) Although Transformer fails to converge, Mamba and S4-Mamba can learn sparse parity of $d = 10, k = 2$. Each model is trained with width 256 and depth 12.



(b) Test loss (mean squared error) on vector-valued MQAR and respective model configurations.

d_model	64	128
Mamba	8.64e-1	1.64e-1
Transformer w/ PE	**5.17e-6**	**8.76e-7**
Transformer w/o PE	1.14e-3	8.66e-5
MambaFormer	0.73e-5	3.37e-6

(i) 32 key-value pairs with 4 queries.

d_model	64	128
Mamba	7.23e-1	1.50e-1
Transformer w/ PE	3.99e-5	**2.46e-7**
Transformer w/o PE	7.61e-5	5.55e-5
MambaFormer	**1.03e-5**	3.79e-7

(ii) 32 key-value pairs with 16 queries.

Given our results, it will be interesting to see how hybrid architectures perform in other kinds of ICL tasks, as those found in Xie et al. (2021); Akyürek et al. (2024). In turn, we explore formal language ICL capabilities in the following subsection.

3.4 In-context learning formal languages

Given the empirical strength in hybrid models, this subsection analyzes their performance on synthetic formal language benchmarks, namely GINC and ICLL RegBench. We use these benchmarks as a proxy to measure language ICL capabilities. The experiment results can be found in A.3.9.

On GINC, Mamba achieves the best ICL accuracy among non-LSTM models, though Transformer achieves lower perplexity. Interestingly, Standard Hybrid performs on par with Transformer and Mamba, while MambaFormer performs slightly worse than other models here. However, findings from Xie et al. (2021) indicate that LSTMs excel over Transformers on GINC, even when accounting for different settings such as vocabulary size or the number of in-context examples. This aligns with previous findings in which Transformers perform worse or comparably to LSTMs in many formal languages considered (Bhattamishra et al., 2020; Deletang et al., 2022). Yet, Transformers are the *de facto* superior model for language modeling, so it remains unclear how performance on this benchmark translates to real-world language ICL, where Transformers typically outperform LSTMs.

On RegBench, which favors Transformers over attention-free models, Mamba indeed performs worse than Transformer, consistent with previous findings. Notably, hybrid architectures excel on this benchmark, converging much faster both Mamba and Transformer while achieving higher accuracy.

Given prior evidence that Standard Hybrid achieves lower perplexity in language modeling (Gu & Dao, 2023), our new results suggest that hybrid models offer a promising direction for both language modeling and in-context learning on language tasks. We hope these results and analysis demonstrate the potential of hybrid models for language-based applications of ICL.

4 Discussion

In this work, we have provided a comprehensive investigation of in-context learning with state-space models (SSMs) and contrasted them with transformer architecture. Future research directions include exploring (1) how performance on our ICL suite correlates with general language modeling capabilities, such as perplexity on standard NLP benchmarks, (2) the potential for developing more effective architectures by integrating elements from transformers, SSMs, and gating mechanisms, (3) identifying architectural features that contribute to effective in-context learning, and (4) assessing the impact of MambaFormer and other innovative architectures on language modeling performance.

REFERENCES

Kwangjun Ahn, Xiang Cheng, Hadi Daneshmand, and Suvrit Sra. Transformers learn to implement preconditioned gradient descent for in-context learning. *arXiv preprint arXiv:2306.00297*, 2023.

Ekin Akyürek, Dale Schuurmans, Jacob Andreas, Tengyu Ma, and Denny Zhou. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2022.

Ekin Akyürek, Bailin Wang, Yoon Kim, and Jacob Andreas. In-context language learning: Architectures and algorithms. *arXiv preprint arXiv:2401.12973*, 2024. URL https://arxiv.org/abs/2401.12973.

Simran Arora, Sabri Eyuboglu, Aman Timalsina, Isys Johnson, Michael Poli, James Zou, Atri Rudra, and Christopher Ré. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023. URL https://arxiv.org/abs/2312.04927.

Yu Bai, Fan Chen, Huan Wang, Caiming Xiong, and Song Mei. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023.

Iz Beltagy, Matthew E Peters, and Arman Cohan. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020.

Satwik Bhattamishra, Kabir Ahuja, and Navin Goyal. On the ability and limitations of transformers to recognize formal languages. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7096–7116. Association for Computational Linguistics, 2020. doi: 10.18653/v1/2020.emnlp-main.576. URL https://aclanthology.org/2020.emnlp-main.576.

Satwik Bhattamishra, Arkil Patel, Phil Blunsom, and Varun Kanade. Understanding in-context learning in transformers and llms by learning to learn discrete functions. *arXiv preprint arXiv:2310.03016*, 2023. URL https://arxiv.org/abs/2310.03016.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. Why can gpt learn in-context? language models secretly perform gradient descent as meta-optimizers. In *Findings of the Association for Computational Linguistics: ACL 2023*, pp. 4005–4019, 2023.

Tri Dao, Daniel Y Fu, Khaled K Saab, Armin W Thomas, Atri Rudra, and Christopher Ré. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022. URL https://arxiv.org/abs/2212.14052.

Gregoire Deletang, Anian Ruoss, Jordi Grau-Moya, Tim Genewein, Li Kevin Wenliang, Elliot Catt, Chris Cundy, Marcus Hutter, Shane Legg, Joel Veness, et al. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*, 2022.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. The pile: An 800gb dataset of diverse text for language modeling, 2020.

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

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL https://arxiv.org/abs/2312.00752.

Albert Gu, Karan Goel, Ankit Gupta, and Christopher Ré. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35971–35983, 2022a.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces, 2022b.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022c. URL https://openreview.net/forum?id=uYLFoz1vlAC.

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

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165, 2020. URL http://proceedings.mlr.press/v119/katharopoulos20a.html.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Yingcong Li, M. Emrullah Ildiz, Dimitris Papailiopoulos, and Samet Oymak. Transformers as algorithms: Generalization and stability in in-context learning, 2023a.

Yingcong Li, Kartik Sreenivasan, Angeliki Giannou, Dimitris Papailiopoulos, and Samet Oymak. Dissecting chain-of-thought: Compositionality through in-context filtering and learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b.

Bingbin Liu, Jordan T Ash, Surbhi Goel, Akshay Krishnamurthy, and Cyril Zhang. Exposing attention glitches with flip-flop language modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

Arvind Mahankali, Tatsunori B Hashimoto, and Tengyu Ma. One step of gradient descent is provably the optimal in-context learner with one layer of linear self-attention. *arXiv preprint arXiv:2307.03576*, 2023.

Sewon Min, Mike Lewis, Luke Zettlemoyer, and Hannaneh Hajishirzi. Metaicl: Learning to learn in context. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2791–2809, 2022.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, and Anna Chen et al. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi GV Kiran, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023. URL https://arxiv.org/abs/2305.13048.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023. URL https://arxiv.org/abs/2302.10866.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022.

Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016.

Jürgen Schmidhuber, Jieyu Zhao, and Marco Wiering. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28: 105–130, 1997.

Yutao Sun, Li Dong, Shaohan Huang, Shuming Ma, Yuqing Xia, Jilong Xue, Jianyong Wang, and Furu Wei. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023. URL https://arxiv.org/abs/2307.08621.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017*, pp. 5998–6008, 2017. URL https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html.

Johannes von Oswald, Eyvind Niklasson, Ettore Randazzo, João Sacramento, Alexander Mordvintsev, Andrey Zhmoginov, and Max Vladymyrov. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023a.

Johannes von Oswald, Eyvind Niklasson, Maximilian Schlegel, Seijin Kobayashi, Nicolas Zucchet, Nino Scherrer, Nolan Miller, Mark Sandler, Max Vladymyrov, Razvan Pascanu, et al. Uncovering mesa-optimization algorithms in transformers. *arXiv preprint arXiv:2309.05858*, 2023b. URL https://arxiv.org/abs/2309.05858.

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Sang Michael Xie, Aditi Raghunathan, Percy Liang, and Tengyu Ma. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2021.

Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms, 2023a.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023b.

A Appendix

A.1 Related Work

Transformer-based in-context learning. The role of attention in ICL has been the focus of both theoretical and empirical research. Studies have primarily focused on meta-learning (Ravi & Larochelle, 2016; Min et al., 2022), where one explicitly trains for ICL. Notably, Garg et al. (2022) have examined transformers in in-context regression tasks, from learning linear regression to learning decision trees. Subsequent works have suggested that attention may mimic various optimization algorithms (Akyürek et al., 2022; von Oswald et al., 2023b; Dai et al., 2023). In fact, Ahn et al. (2023); Mahankali et al. (2023) have provably shown that the global minimum of the linear regression ICL objective implements one step of preconditioned gradient descent for one layer of linear attention.

While these settings might appear simplistic and detached from language models, Bhattamishra et al. (2023) showed that a frozen GPT-2 can implement the nearest neighbor algorithm, drawing connections between the ICL in existing language models and the stylized setting of training for ICL from random initialization. Furthermore, Olsson et al. (2022) also empirically demonstrate that "induction heads", which are attention heads that solve a simple retrieval problem, correlate with ICL behavior, providing a strong connection between retrieval and ICL.

Sub-quadratic architectures. The number of effective floating point operations in an attention layer scales quadratically with respect to the input sequence length. Numerous approximations or alternative model architectures have been proposed to overcome the quadratic dependence. These range from approximating attention mechanisms (Beltagy et al., 2020; Wang et al., 2020) to the development of novel recurrent convolutional models such as structured state-space models (Gu et al., 2022c).

S4 (Gu et al., 2022a) is a family of sequence models characterized by a discretized state-space model

$$\mathbf{h}_t = \overline{\mathbf{A}}\mathbf{h}_{t-1} + \overline{\mathbf{B}}\mathbf{x}_t,\ y_t = \mathbf{C}\mathbf{h}_t, \tag{1}$$

where $\mathbf{h}_t$ represents the hidden state and $(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C})$ are input-independent (transformed) parameters. The recurrence is expressible as a convolution, enabling near-linear complexity using Fast Fourier Transform. Viewed in this framework, Linear Transformers (Katharopoulos et al., 2020), which employ linear attention without softmax, can be seen as a variant of linear SSM.

Building upon this concept, H3 (Dao et al., 2022) integrates an S4 with dual gated connections. The recent Mamba (Gu & Dao, 2023) departs from the standard SSM by introducing a selection mechanism that makes $(\overline{\mathbf{A}}, \overline{\mathbf{B}}, \mathbf{C})$ in equation 1 dependent on the input $\mathbf{x}_t$ allowing input-dependent sequence mixing.

There are other notable attention-free models such as Hyena (Poli et al., 2023), RWKV (Peng et al., 2023), RetNet (Sun et al., 2023), and GLA (Yang et al., 2023b). Despite of state-of-the-art performance for models like Mamba, Arora et al. (2023) have demonstrated that subquadratic models still lag behind attention on multi-query recall tasks, which is a generalization of the induction head task (Olsson et al., 2022).

In their study, Xie et al. (2021) introduced a synthetic language-based dataset for in-context learning, named GINC, and demonstrated that both transformers and LSTMs (Hochreiter & Schmidhuber, 1997) can perform ICL. Notably, LSTMs outperformed transformers in ICL accuracy on GINC, a finding similar to that found in Liu et al. (2023) for their flip-flop language modeling task. More recently, Akyürek et al. (2024) proposed a language-based ICL benchmark for training models on formal languages generated by random finite automata. Their results showed that Transformers notably better than subquadratic models, establishing a benchmark that effectively measures ICL in language modeling.

A.2 Model Training for In-context Learning

We train models to learn a specific function class $\mathcal{F}$ in-context. Training begins by generating random prompts: selecting a function $f \in \mathcal{F}$ from distribution $\mathcal{D}_{\mathcal{F}}$ and sampling a sequence of random inputs $\mathbf{x}_1, \ldots, \mathbf{x}_N \in \mathbb{R}^d$ i.i.d. from $\mathcal{D}_{\mathcal{X}}$. Here, N and d represent the number of in-context examples and

the dimension of $\mathbf{x}_i$, respectively. These inputs create the prompt $P = (\mathbf{x}_1, f(\mathbf{x}_1), \ldots, \mathbf{x}_N, f(\mathbf{x}_N))$. We train the model f_θ, parameterized by θ, by minimizing the expected loss over all prompts:

$$\min_\theta \mathbb{E}_P \left[\frac{1}{N} \sum_{i=1}^{N-1} \ell(f_\theta(P^i), f(\mathbf{x}_i)) \right], \tag{2}$$

where $P^i := (\mathbf{x}_1, f(\mathbf{x}_1), \ldots, \mathbf{x}_i, f(\mathbf{x}_i), \mathbf{x}_{i+1})$ and $\ell(\cdot, \cdot)$ is a loss function. Since $f : \mathbb{R}^d \to \mathbb{R}$, we append $d - 1$ zeros to $f(\mathbf{x})$ to match the dimensions. We use appropriate loss functions for each task.

Model architecture. We primarily focus on SSMs, including (1) Mamba (Gu & Dao, 2023), a state-of-the-art SSM model with selection mechanism; (2) S4 (Gu et al., 2022a), a linear time-invariant predecessor of Mamba; and (3) S4-Mamba, a variant where Mamba's input-dependent S6 is replaced with input-independent S4, while maintaining the same structure as Mamba. The primary differences between the two S4 models lie in the application of multiplicative gating and the module order.[1]

Training. We train each model by sampling a batch of random prompts at each training step and updating the model parameters using Adam optimizer (Kingma & Ba, 2014). We use a batch size of 64 and trained for 500,000 iterations (except for the vector MQAR task; see A.3.7).

Evaluation. We evaluate the model performance on in-context learning using task and data distributions $\mathcal{D}_\mathcal{F}$ and $\mathcal{D}_\mathcal{X}$ consistent to those during training. A function and a sequence of N inputs are sampled from $\mathcal{D}_\mathcal{F}$ and $\mathcal{D}_\mathcal{X}$, respectively, to generate a test prompt $P_{\mathsf{test}} = (\mathbf{x}_1, f(\mathbf{x}_1), \ldots, \mathbf{x}_N, f(\mathbf{x}_N))$. We create 1,280 prompts and measure the empirical mean of equation 2 across the prompts for in-context learning performance.

Throughout our experiments, we keep the total number of parameters of models roughly the same for each configuration as explained in A.2. To plot the model performance as the model capacity grows, we calculate the total floating point operations (FLOPs) used for training the model.

Model configurations and training implementation details are provided in A.2.

A.3 In-context Learning Tasks

In this section, we describe in detail the in-context learning tasks we have tested Transformers, SSM models, and variant models with.

A.3.1 Learning Regression

For all regression tasks, in-context examples $\mathbf{x}_i$ are sampled from the Gaussian distribution $\mathcal{N}(0, \mathbf{I}_d)$, where $\mathbf{I}_d$ is the $d \times d$ identity matrix. We use the squared error loss for model training.

Linear Regression We examine the class of linear functions $\mathcal{F} = \{f | f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \mathbf{w} \in \mathbb{R}^d\}$. $\mathbf{w}$ is sampled from the Gaussian distribution $\mathcal{N}(0, \mathbf{I}_d)$. We set $d = 20$.

Sparse Linear Regression The setting is identical to linear regression, except that $\mathbf{w}$ is sampled from $\mathcal{N}(0, \mathbf{I}_d)$, after which k coordinates are randomly retained in $\mathbf{w}$, and the rest are set to zero. We set $k = 3$.

Two-Layer Neural Network We consider the class of two-layer linear neural networks $\mathcal{F} = \{f | f(\mathbf{x}) = \mathbf{W}^{(2)} \sigma\left(\mathbf{W}^{(1)} \mathbf{x}\right)\}$, where $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times n_h}$, $\mathbf{W}^{(1)} \in \mathbb{R}^{n_h \times d}$, and $\sigma(\cdot) = \max(0, \cdot)$ is the ReLU function. Each element of the weight matrices is independently drawn from $\mathcal{N}(0, 1)$. We use $d = 20$ and $n_h = 100$.

Decision Tree We consider a full binary tree with a fixed depth and input $\mathbf{x} \in \mathbb{R}^d$. Leaf node values are sampled from $\mathcal{N}(0, 1)$, and the rest are sampled uniformly from $\{1, ..., d\}$, functioning as indices of $\mathbf{x}$. For a given index, $\mathbf{x}[i] > 0$ indicates moving to the right, and vice versa. $\mathbf{y}$ is the leaf node value when the traversal terminates.

[1] https://github.com/state-spaces/s4/blob/main/models/s4

A.3.2 Learning with Outliers

The problems that belong to this family adopt the basic setting of the standard linear regression task. With a fixed probability p, each pair of $(\mathbf{x}_i, f(\mathbf{x}_i))$ in the prompt is replaced with dummy vectors which are either out of the training distribution, or confounders designed to increase the complexity of the task. We test $p \in \{0.2, 0.5\}$ as replacement probabilities for all tasks described below.

Orthogonal-outlier Regression Each pair of $(\mathbf{x}_i, f(\mathbf{x}_i))$ are randomly replaced with $((a_x\mathbf{u} + b_x\mathbf{v})/(a_x^2 + b_x^2), (a_y\mathbf{u} + b_y\mathbf{v})/(a_x^2 + b_x^2))$ where $\mathbf{u}, \mathbf{v} \in \mathbf{w}^{\perp}$. $(\mathbf{u}, \mathbf{v}) := (\mathbf{w}_1 - \text{proj}_{\mathbf{w}}(\mathbf{w}_1), \mathbf{w}_2 - \text{proj}_{\mathbf{w}}(\mathbf{w}_2))$ and $\mathbf{w}_1$ and $\mathbf{w}_2$ are sampled from $\mathcal{N}(0, \mathbf{I}_d)$ and the coefficients a_x, b_x, a_y, b_y are independently sampled from $\mathcal{N}(0, 1)$.

Many-outlier Regression In this setting, $\mathbf{x}_i$ and $f(\mathbf{x}_i)$ are randomly replaced with a d-dimensional vector of ones $\{1\}^d$ and an one-hot vector $[1, 0, \ldots, 0]$, respectively, with probability 90%.

A.3.3 Performance Gaps in Outlier Tasks

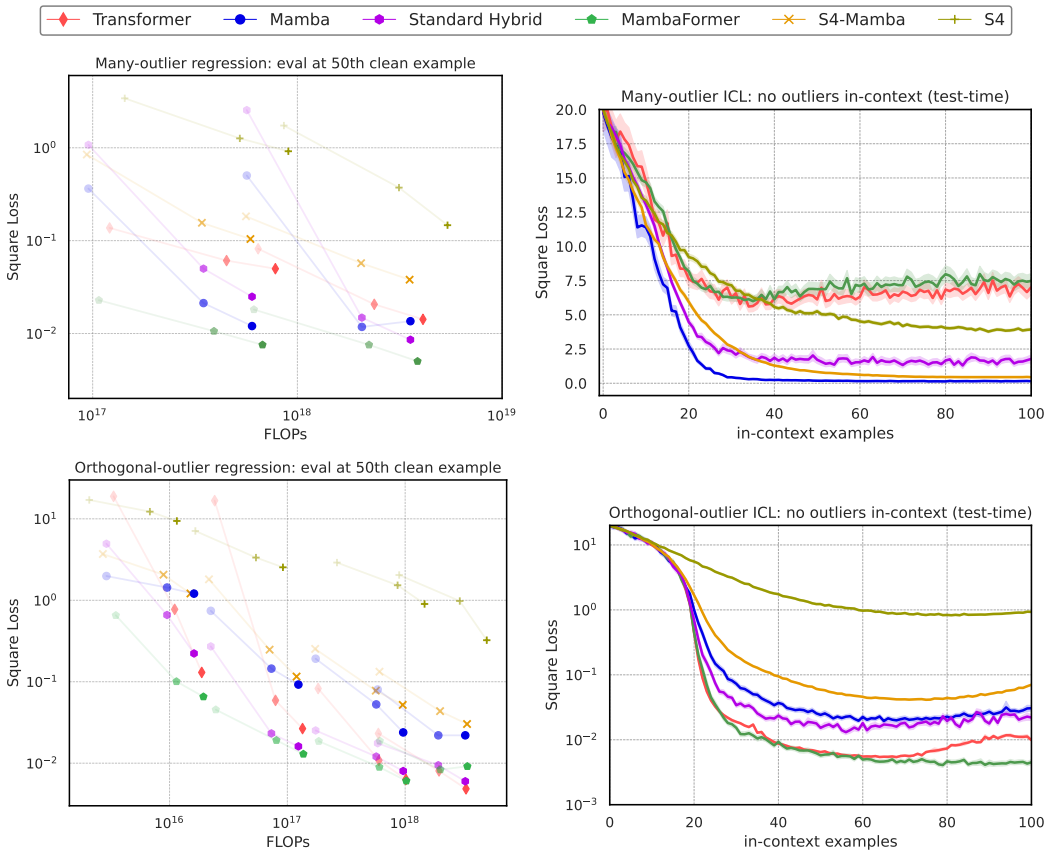


Figure 4: **(Left)** Performance of various architectures on two robust linear regression tasks. More transparent points indicate earlier stages of training; plotted models are trained in between {100k, 300k, 500k} iterations. **(Right)** Out-of-distribution performances when models do not see outliers during test-time, *i.e.*, standard linear regression. Task descriptions can be found in table 1. Standard Hybrid and MambaFormer are hybrid models of Transformer and Mamba defined in A.4.

Orthogonal-outlier regression and many-outlier regression, like other outlier tasks, focus on the model's ability to learn to ignore dummy vectors, either by the fact that the $\mathbf{x}_i \in \mathbf{w}^{\perp}$, or by the fact that $\mathbf{y}_i$ is a vector instead of a zero-padded scalar value. This explicitly requires the models to look at previous input sequences and discover the properties that distinguish the dummy vectors from training examples while learning the class of functions the training prompt represents.

For orthogonal-outlier regression task with a relatively short sequence length of 101, Mamba does not perform as well as Transformer given the same total FLOPs, though its learns significantly better than S4 (figure 4). However, for many-outlier regression where we train on a sequence length of 512 and 90% all-ones replacement, Mamba outperforms Transformers, especially in terms of its out-of-distribution (OOD) accuracy where we evaluate each model on clean sequences with no outliers at all. Recurrent models, such as S4 and Mamba, seem to generalize well in such OOD regime when the data is contaminated with many identical outlier vectors. This is also in line with what Gu & Dao (2023) reports: Mamba fares better in retrieval tasks of long sequence lengths with a single retrieval key. These results indicate that Mamba has no significant issue with filtering out unnecessary information, while retaining the ability to learn linear regression in-context.

A.3.4 Learning Discrete Functions

Sparse Parity Following the setting from Bhattamishra et al. (2023), we consider the class of functions $\mathcal{F} = \{f | f(\mathbf{x}) = \prod_{j \in \mathcal{S}} \mathbf{x}_i[j]\}$ where $\mathbf{x}_i[j]$ denotes the j-th element of the vector $\mathbf{x}_i$ and $\mathcal{S}$ is a subset of $\{1, \ldots, d\}$ with the size k. Each $\mathbf{x}_i$ is sampled uniformly at random from $\{-1, 1\}^d$, and $\mathcal{S}$ of size k is randomly sampled from the set $\{1, \ldots, d\}$. For this task, we train a model using the cross-entropy loss and evaluate the model using a binary indicator for accuracy, which assigns 1 to correct predictions and 0 to incorrect ones.

A.3.5 Median Convergence time for Sparse Parity Task

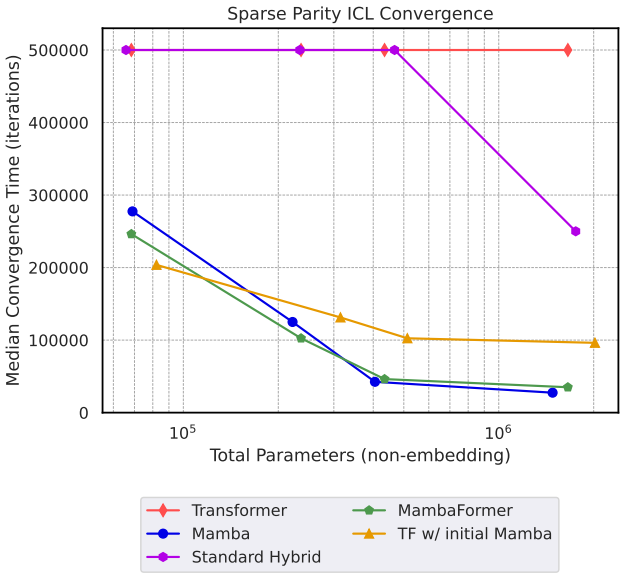


Figure 5: Median convergence time of learning parity over 5 random seeds for max. 500k iterations. Having the initial layer as Mamba is essential for efficiently learning parities.

A.3.6 Learning Chain-of-Thought

Chain-of-Thought-I/O Following the setting from Li et al. (2023b), we consider the class of two-layer linear neural networks $\mathcal{F} = \{f | f(\mathbf{x}) = \mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\mathbf{x}\right)\}$ where $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times n_h}, \mathbf{W}^{(1)} \in \mathbb{R}^{n_h \times d}$, and $\sigma(\cdot)$ is the ReLU function. We set $d = 10$, and $n_h = 8$. We additionally interleave the intermediate hidden feature $\mathbf{s}_i = \sigma\left(\mathbf{W}^{(1)}\mathbf{x}_i\right)$ in our input training sequence in a Chain-of-Thought style. Given the input sequence $(\mathbf{x}_1, \mathbf{s}_1, f(\mathbf{x}_1), \cdots, \mathbf{x}_N, \mathbf{s}_N, f(\mathbf{x}_N), \mathbf{x}_{\text{test}})$, the model is evaluated on the final output prediction $\hat{\mathbf{y}}$ based on the input sequence and the intermediate layer prediction $\hat{\mathbf{s}}_{\text{test}}$.

Chain-of-Thought-I/O Results Table 2 presents the configurations for the Chain-of-Thought-I/O task using a 2-layer ReLU neural network, following the setup described by Li et al. (2023b). In

the model scale experiment, the input dimension $d = 10$ and hidden layer dimension $k = 8$ are held constant while varying the model scale. Additionally, the hidden dimension k is varied among $4, 8, 16$ while fixing the model scale to small to identify the effect of problem scale.

Table 2: Model configurations for Chain-of-Thought-I/O experiments.

Model	# layers	embed dim	# heads (MHA)
Standard	12	256	8
Small	6	128	4
Tiny	3	64	2

Figure 6 shows thatMamba models are capable of in-context learning in a chain-of-thought manner, performing comparably to Transformer models across the tested configurations. In smaller model configurations, Mamba models exhibit superior performance compared to Transformer models. However, as model size increases, Transformer models begin to surpass Mamba models. The performance of Transformer models remains relatively stable across different problem sizes, while Mamba models' performance is significantly influenced by the size of the hidden layer. Specifically, Mamba models excel over Transformer models at smaller problem sizes (i.e., smaller hidden dimensions), but their advantage diminishes as the problem size expands.

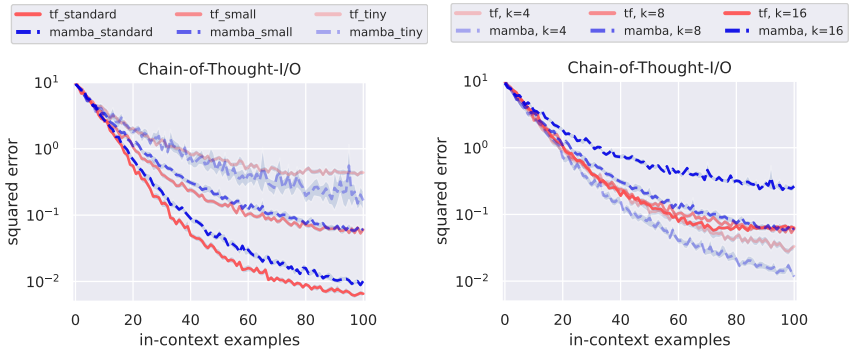


Figure 6: Performance of Transformer and Mamba Models on the Chain-of-Thought-IO Task. Experiments on varying the model size (left) and varying the hidden dimension (right).

A.3.7 LEARNING RETRIEVAL

Vector-valued Multi-Query Associative Recall We test the model's ability to do multi-query associative recall (MQAR) Arora et al. (2023). To better measure the model's ability to retrieve information from context, we consider a variant of MQAR such that keys and values are vector-valued. Specifically, in this task, the model is given a sequence of key-value pairs of vectors $\{\mathbf{k}_1, \mathbf{v}_1, ..., \mathbf{k}_n, \mathbf{v}_n\}$ where $\mathbf{k}_i, \mathbf{v}_i \in \mathcal{S}^{d-1}$. The query consists of sequence of vectors $\{\mathbf{q}_1, ..., \mathbf{q}_m\}$. For each query $\mathbf{q}_j$, there exists some $1 \leq l \leq n$ such that $\mathbf{q}_j = \mathbf{k}_l$. The model must learn to output $\mathbf{v}_l$ associated with the query $\mathbf{q}_j$ for each of the queries, producing m outputs total. We train a model using the norm-squared error.

We run vectorized MQAR on two settings: (1) 32 key-value pairs with 16 queries and (2) 32 key-value pairs with 4 queries. The training set consists of $300,000$ training samples. We train for 64 epochs with batch size of 64 and evaluate on a test set of $3,000$ samples. For each setting, we sweep with learning rates in np.logspace(-4, -2, 4) and report the best result among all learning rates. All models have 4 layers.

A.3.8 CLOSING THE GAP IN RETRIEVAL

We perform an ablation study by equipping Transformer with an initial Mamba block without any positional encoding. Although this variant Transformer only has fewer Mamba blocks than Standard Hybrid, it solves parity almost as efficiently as Mamba. Not only does this show us that order of layers in interleaving matter, as shown in Press et al. (2022), but also that Mamba can complement Transformer without hurting performance in ICL. This result brings up intriguing difference between the function learning capabilities of Attention and Mamba; we leave this question up for further study.

The gap between Mamba and Transformer in vector-valued MQAR task is largely due to the fact that Mamba (as an SSM) compresses context into smaller states when generating output, while the Attention mechanism in Transformer does not compress the context. The amount of information about the context Mamba has at each state depends on the dimension of hidden state (as the hidden states capture the important information in the context) and it is challenging if the task is to accurately retrieve a specific part of the context by a query that is placed after the context.

To close the gap in the vector-valued MQAR task between Mamba and Transformer and without sacrificing too much of the efficiency, we add one attention layer within layers of Mamba blocks. In particular, in a Mamba model of 4 layers (8 Mamba blocks stacked homogeneously), we replace the middle two blocks with Standard Hybrid (w/o positional embedding). As shown in 3b, Mamba model gains a significant improvement in vector-valued MQAR by having one Standard Hybrid. We further test MambaFormer on the same task and find that MambaFormer closes the gap in vector-valued MQAR task.

A.3.9 LEARNING SYNTHETIC FORMAL LANGUAGES

Although not the main focus of our work, we conduct initial experiments using synthetic language benchmarks designed to assess in-context learning (ICL) capabilities within the language setting. Given that real language ICL typically demands extensive datasets and computational resources, these synthetic datasets act as useful proxy for exploring language ICL. For detailed descriptions of their construction and evaluation, we direct readers to the respective publications.

GINC dataset (Xie et al., 2021). Generative In-Context learning (GINC) dataset is a small-scale language dataset synthetically generated using a mixture of hidden markov models. Its pretraining dataset contains approximately 10 million tokens and each trained model is evaluated on 2500 test-time prompts containing 0 to 64 examples. We train and test our models using a vocabulary size of 100. We additionally train LSTMs for this dataset, as done in prior work.

RegBench (Akyürek et al., 2024). In-context Language Learning (ICLL) RegBench is a synthetic regular language benchmark created by randomly generating probabilistic finite automata (PFA) with uniform transition probabilities; multiple problem instances are produced that include samples from each PFA. The models are evaluated using a greedy-decoding accuracy metric, which assesses whether each next token predicted by the model is valid under the current regular language.

GINC	**Parameters**	**Train PPL** ($\downarrow$)	**Valid PPL** ($\downarrow$)	**ICL acc.** ($\uparrow$)
LSTM	29M	**3.53**	**3.71**	**96.4** $\pm$ 0.6
Transformer	86M	4.06	4.14	84.2 $\pm$ 5.1
Mamba	90M	4.30	4.57	87.1 $\pm$ 7.8
MambaFormer	77M	4.22	4.77	79.6 $\pm$ 3.8
Standard Hybrid	74M	4.18	4.65	85.0 $\pm$ 3.1

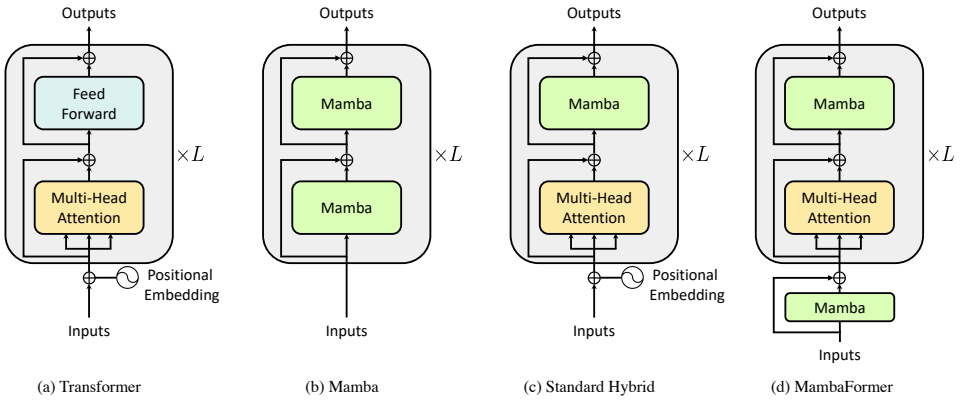
Table 3: GINC data has a vocab size of 100 and the ICL accuracy is evaluated at 64 examples, where each example has length 10. Each model is trained with embedding size 768 and 12 layers, other than LSTM, which used embedding size 768, hidden layer size 768, and 6 layers. We include 90% confidence intervals for ICL accuracy. We follow the same training recipes as Xie et al. (2021).

RegBench (trained 15 epochs)	**Train PPL** ($\downarrow$)	**Valid PPL** ($\downarrow$)	**Acc.** ($\uparrow$)
LSTM	6.20	6.39	51.0
Transformer	4.20	4.17	92.6*
Mamba	5.59	5.69	69.4
MambaFormer	**1.01**	**1.01**	99.8
Standard Hybrid	**1.01**	**1.01**	**99.9**

RegBench (trained 120 epochs)	**Train PPL** ($\downarrow$)	**Valid PPL** ($\downarrow$)	**Acc.** ($\uparrow$)
LSTM	3.33	4.37	73.5
Transformer	1.03	1.10	98.9
Mamba	3.12	3.32	87.8
MambaFormer	**1.01**	**1.01**	99.8
Standard Hybrid	**1.01**	**1.01**	**99.9**

Table 4: Perplexity (PPL) and greedy-decoding accuracy for RegBench after training each model 15 and 120 epochs. We use the same models configurations as done in Akyürek et al. (2024) and perform similar hyperparameter sweeps. See A.3 for how accuracy is measured. * denotes reported accuracy in Akyürek et al. (2024).

A.4 Hybrid Architectures



(a) Transformer (b) Mamba (c) Standard Hybrid (d) MambaFormer

Figure 7: Model Architectures. (a) and (b) denote the standard Transformer and Mamba architectures. (c) denotes the hybrid architecture of Mamba and Attention blocks, following the design proposed in Gu & Dao (2023). (d) demonstrates the proposed architecture, namely MambaFormer, which replaces the Positional Encoding with a Mamba block. For convenience, we denote 2 blocks of either Mamba, Multi-head Attention, or a Feed Forward Network as 1 layer.

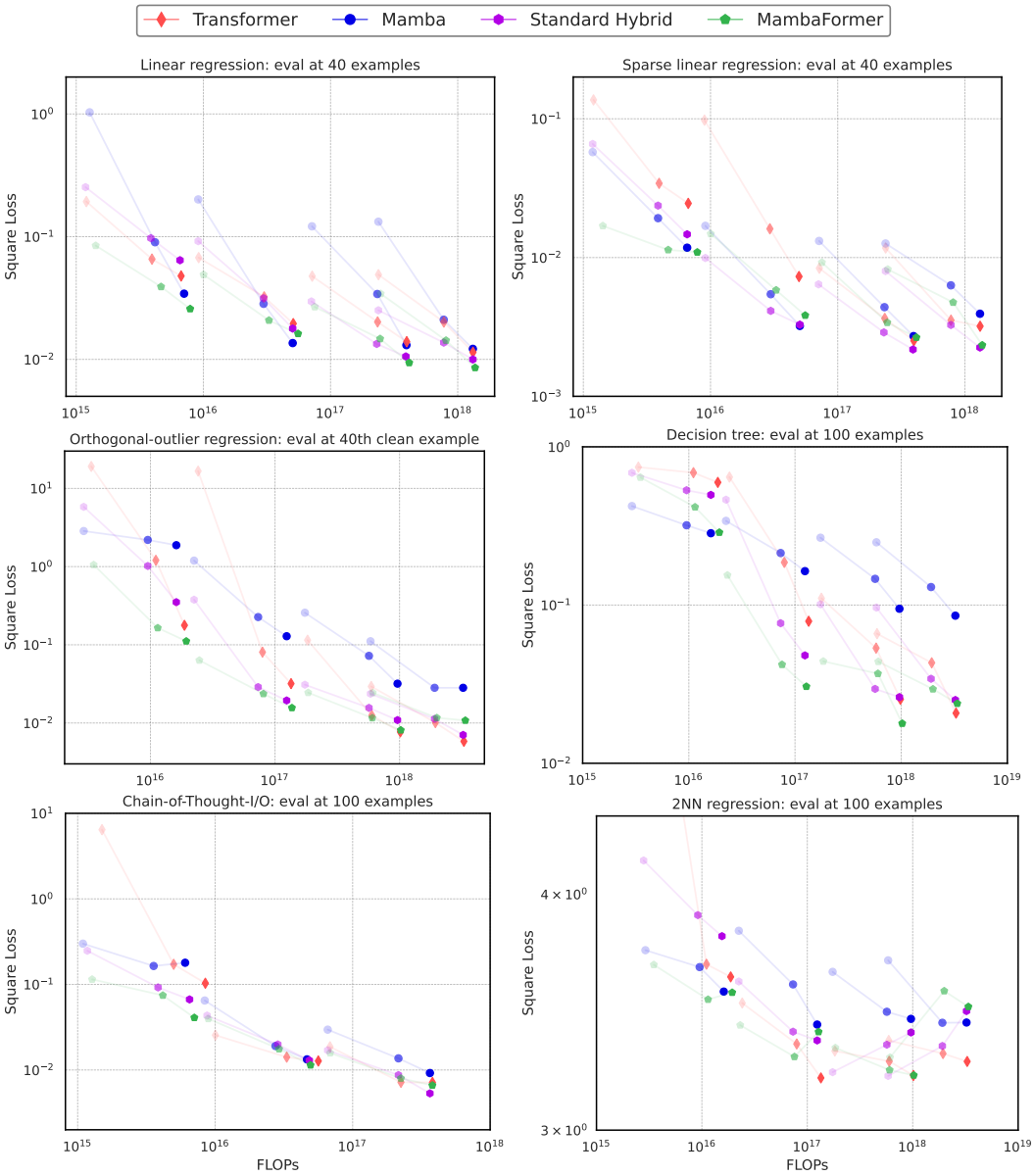


Figure 8: A suite of ICL tasks ran for Transformer, Mamba, and hybrid architectures where each color represents a different architecture. More transparent points indicate earlier stages of training; plotted models are trained {100k, 300k, 500k} iterations. Standard Hybrid and MambaFormer are hybrid models of Transformer and Mamba defined in the figure above: A.4.