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

Modern large language models (LLMs) exhibit remarkable in-context learning (ICL) capabilities, enabling them to learn new tasks with a few demonstrations and without further weight fine-tuning. Although the exact emergence mechanism of these capabilities warrants further theoretical and empirical investigation (Chan et al., 2022; Wei et al., 2022; Min et al., 2022b; Schaeffer et al., 2023), experiments on larger Transformer-based models consistently demonstrate that their ICL capabilities improve as training loss reduces (Brown et al., 2020; Kaplan et al., 2020; Muennighoff et al., 2023).

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. (2022a) 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; Liu et al., 2022; Bai et al., 2023; Giannou et al., 2023; Li et al., 2023a; von Oswald et al., 2023a,b; Yang et al., 2023a; Zhou et al., 2023).

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

Preprint. Under review.

	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	✓	✗	✓

Table 1: 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.

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 (Katharopoulos et al., 2020; Zhai et al., 2021; Dao et al., 2022; 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 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).

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, but can outperform Transformers in other complex ICL tasks, such as sparse parity, where Transformer models struggle. Performance of different models on each task is summarized in Table 1.

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 Figure 1. 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. [2]

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 architectures.

We acknowledge that a limitation of our study lies in the focus on non-language ICL tasks and smaller models. It is possible that an architectural comparison between SSMs and transformers for more

[2]Code is available at https://github.com/krafton-ai/mambaformer-icl.

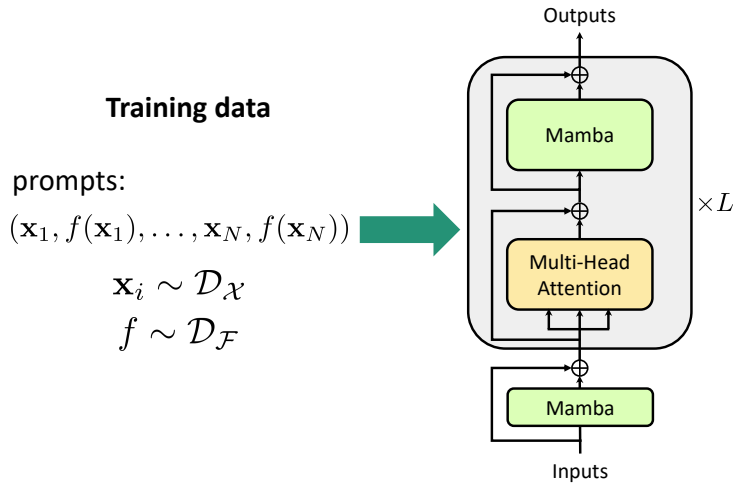


Figure 1: MambaFormer is a hybrid architecture that replaces MLP blocks within the transformer with Mamba blocks. Importantly, the architecture starts with a Mamba block and does not use positional encoding. In our ICL evaluations, we find that MambaFormer consistently achieves a best-of-both-worlds performance when compared to Transformer and Mamba.

general ICL tasks in actual language settings at higher parameter counts might not be yield the same observations as we offer here. Nevertheless, we show potential ICL language capabilities of these architectures by conducting experiments on synthetic formal language ICL datasets (Xie et al., 2021; Akyürek et al., 2024). Moreover, our non-language empirical findings indicate that, apart from its difficulty in some retrieval tasks, similar to those noted by Arora et al. (2023), there seems to be no fundamental obstacle for Mamba to perform in-context learning.

2 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., 2022a), 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., 2022b).

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.

3 Experimental Setup

We evaluate the ICL capabilities of SSMs and Transformers by training each model from scratch on each specific task, detailed in Section 3.1. Section 3.2 outlines the ICL and related tasks investigated in our study. We provide a brief summary of our tasks in the following Table 2.

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	$\mathbf{W}^{(1)}_{ij}, \mathbf{W}^{(2)}_{ij} \sim \mathcal{N}(0, 1)$	–
Decision Tree	20	101	$\mathbf{x}, \text{Leaf} \sim \mathcal{N}(0, 1), \text{non\_leaf} \sim \{1, ..., d\}$	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 2: Summary of Tasks. All models are trained for 500,000 iterations (except for the vector MQAR; see Appendix A.5).

3.1 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.[3]

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 Appendix A.5).

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 $P_{\text{test}} = (\mathbf{x}_1, f(\mathbf{x}_1), \ldots, \mathbf{x}_N, f(\mathbf{x}_N))$. We create 1,280 prompts and measure the empirical mean of Eq. (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 Appendix 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. The calculation for Transformer and Mamba can be found in Appendix B, which are based on (Kaplan et al., 2020; Gu & Dao, 2023).

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

3.2 In-context learning tasks

We provide an overview 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. The tasks are summarized in Table 2.

3.2.1 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 $\mathcal{F} = \{f | f(\mathbf{x}) = \mathbf{w}^\top \mathbf{x}, \mathbf{w} \in \mathbb{R}^d\}$ where $\mathbf{w}$ 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 after sampling $\mathbf{w}$ from $\mathcal{N}(0, \mathbf{I}_d)$, k coordinates are randomly retained and the rest are set to zero. We set $k = 3$.

Two-layer neural network. We consider the class of two-layer ReLU neural networks $\mathcal{F} = \{f | f(\mathbf{x}) = \mathbf{W}^{(2)} \sigma\left(\mathbf{W}^{(1)} \mathbf{x}\right)\}$, where $\mathbf{W}^{(2)} \in \mathbb{R}^{1 \times h}$, $\mathbf{W}^{(1)} \in \mathbb{R}^{h \times d}$, 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 $h = 100$.

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 at random from $\{1, ..., d\}$, functioning as indices of $\mathbf{x}$. At a given non-leaf node, we move to the right if $x[i] > 0$, where i is the sampled index, and otherwise move to the left. y is the leaf node value when the traversal terminates.

3.2.2 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 $p \in \{0.5, 0.9\}$ as the replacement probabilities for the tasks described below. During training, we do not compute the loss for the replaced outliers.

For evaluation, however, the locations of the dummy vectors were fixed to ensure that the models are evaluated on the same number of in-context examples across batches. So we evaluate the loss at the 50th clean in-context example, where there is a clean example after every nine outliers for many-outlier ICL and after every one outlier for orthogonal-outlier ICL.

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

Orthogonal-outlier regression. Each pair of $(\mathbf{x}_i, f(\mathbf{x}_i))$ is 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%. Here, we test longer sequences of $N = 512$, where only 10% of the sequence yields useful information about the true target vector.

3.2.3 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 subset $\mathcal{S}$ 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.

3.2.4 Learning Chain-of-Thought

Chain-of-Thought-I/O. Following the setting from Li et al. (2023b), we consider the class of two-layer ReLU neural networks $\mathcal{F} = \{f | f(\mathbf{x}) = \mathbf{W}^{(2)}\sigma\left(\mathbf{W}^{(1)}\mathbf{x}\right)\}$, where $\mathbf{W}^{(2)} \in \mathbb{R}^{1\times h}, \mathbf{W}^{(1)} \in \mathbb{R}^{h\times d}$, and $\sigma(\cdot)$ is the ReLU function. We set $d = 10$ and $h = 8$. 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 (CoT) 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 $\hat{\mathbf{s}}_{\text{test}}$.

3.2.5 Learning retrieval

Vector multi-query associative recall. We test the model's ability to do multi-query associative recall (MQAR) (Arora et al., 2023). While MQAR is not an ICL task, model's ability to do associative recall (AR) is highly related to model's ability to learn in-context (Olsson et al., 2022). To better measure the model's ability to retrieve information from context, we consider a variant of MQAR. The keys and the values are vectors, which can be interpreted as unique token embeddings. Specifically, 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 $\mathbf{k}_i, \mathbf{v}_i \in \mathcal{S}^{d-1}$ are sampled uniformly from the unit d-sphere. 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 models with mean squared error loss.

3.2.6 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.

4 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.

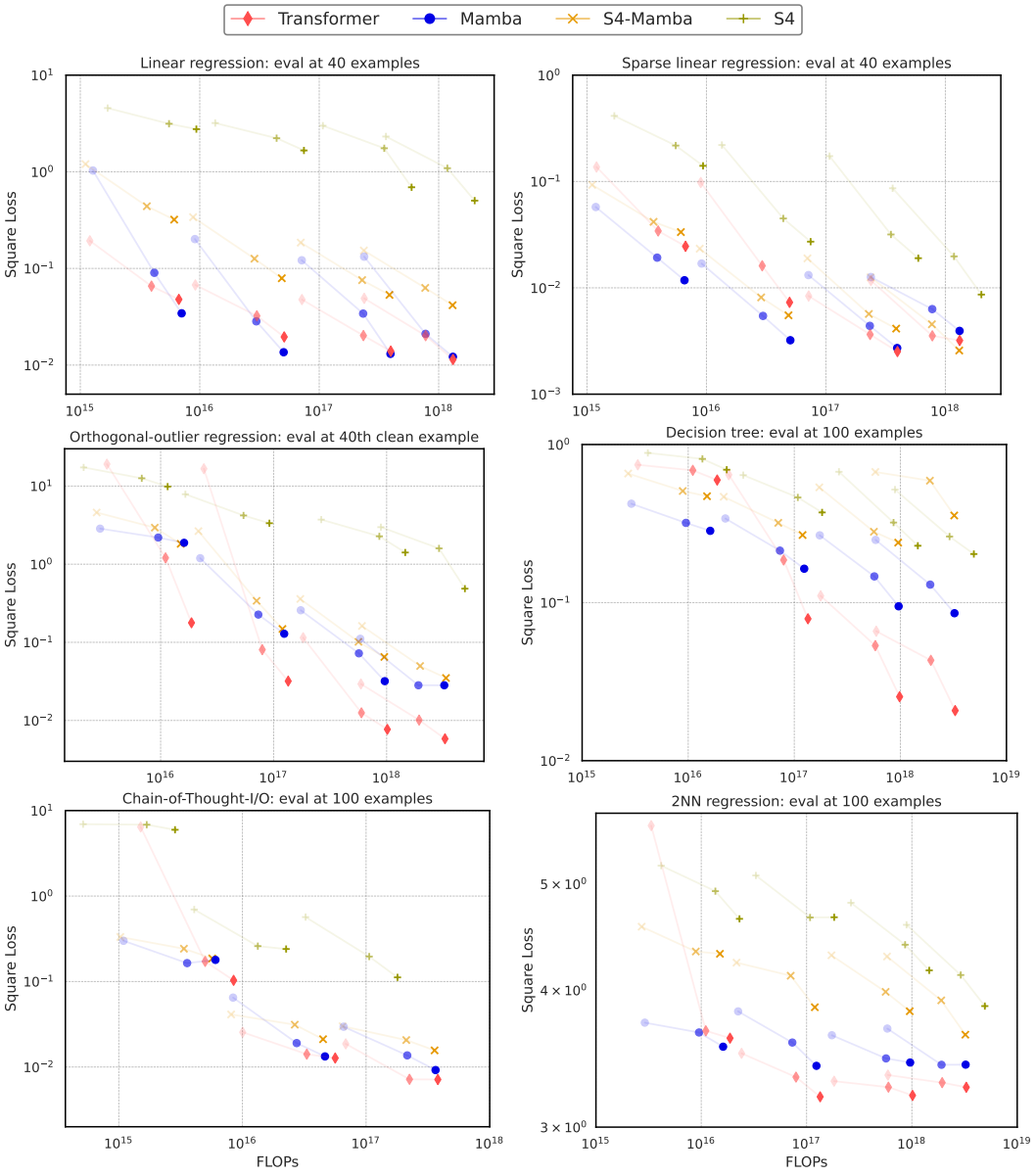


Figure 2: Model performance on our suite of ICL tasks for Transformer, Mamba, S4-Mamba, and S4 where each color represents a different architecture. For each architecture, the best performing model given the same amount of FLOPs is plottet (see Appendix A for details on model configurations). Transparent points indicate earlier stages of training; plotted models are trained in between {100k, 300k, 500k} iterations. The descriptions of tasks can be found in Section 3.2.

4.1 Mamba *can* in-context learn!

> **Finding 1:** *Mamba outperforms its simpler counterparts, while performing as well as Transformer on a range of ICL tasks.*

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.

4.2 Performance gaps in more complex ICL tasks

We also consider a family of more complex ICL tasks, namely learning decision tree, sparse parity, outlier-robust regression (Figure 3) and Chain-of-Thought (Figure 4). We elaborate on the performances of each model on each task in our findings below.

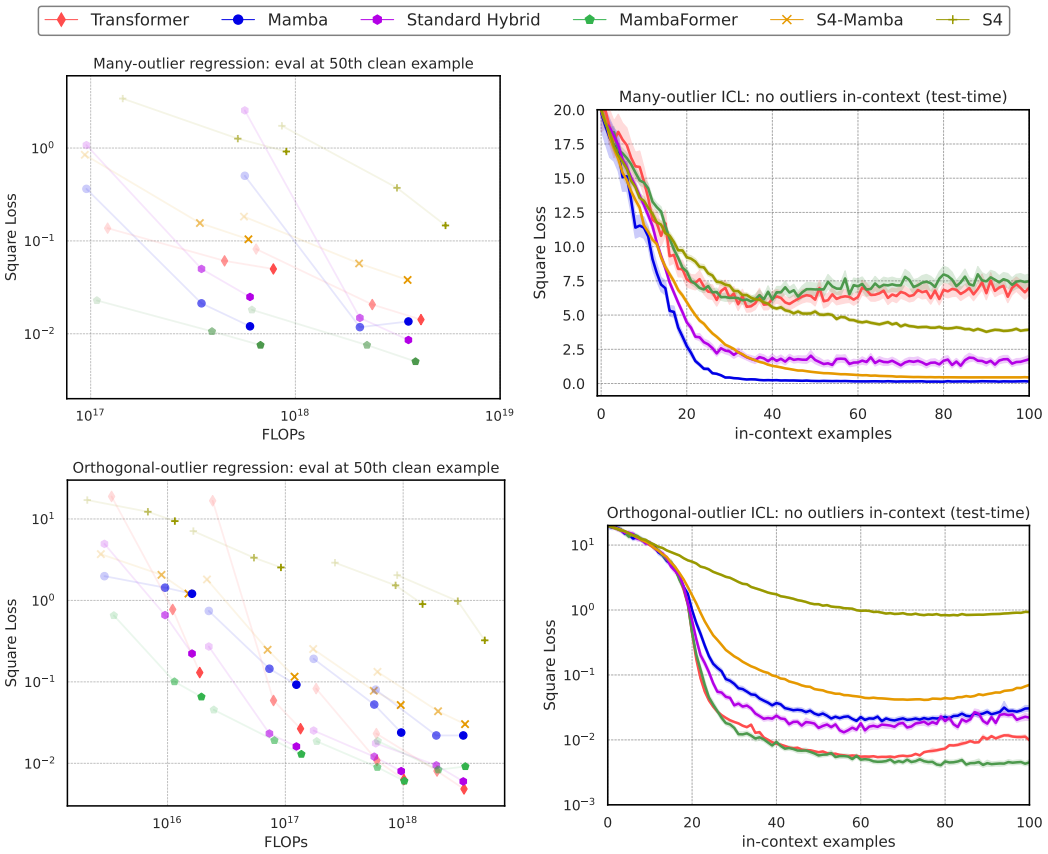


Figure 3: **(Left)** Performance of various architectures on two robust linear regression tasks. More transparent points indicate earlier stages of training; plotted models are trained in between {100k, 300k, 500k} iterations. **(Right)** Out-of-distribution performances when models do not see outliers during test-time, *i.e.*, standard linear regression. Task descriptions can be found in Table 2. Standard Hybrid and MambaFormer are hybrid models of Transformer and Mamba defined in Section 5.

Finding 2: *For outlier-robust regression, Mamba outperforms Transformer in ignoring prevalent fixed outliers, while Transformer is better when the outliers are not fixed.*

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 3). 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.

Finding 3: *For Chain-of-Thought-I/O, Mamba shows comparable performance to Transformer.*

Figure 2 and Figure 4 shows that Mamba models are capable of in-context learning in a chain-of-thought manner, showing comparable performance to Transformer models across the tested configurations. In smaller model configurations, Mamba models exhibit superior performance compared to Transformer models. However, as the 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.

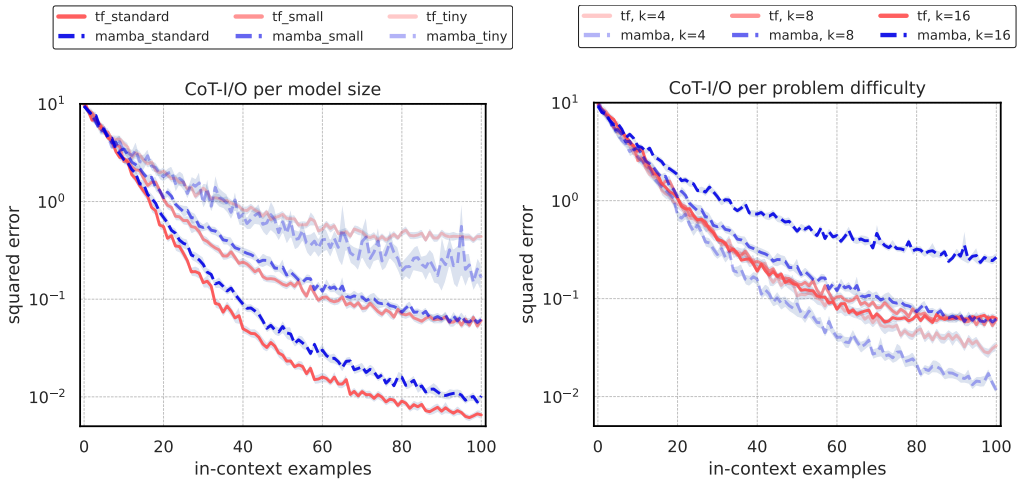


Figure 4: Performance of Transformer and Mamba models on the Chain-of-Thought-I/O task. Experiments on varying the model size (left) and varying the hidden dimension (right). Model configurations can be found in Appendix A Table 7.

4.3 Challenges in parity and retrieval

We run vector MQAR on two settings: (1) 32 key-value pairs with 16 queries and (2) 32 key-value pairs with 4 queries.

Number of queries	4		16	
Embedding dimension	64	128	64	128
Mamba	8.64e-1	1.64e-1	7.23e-1	1.50e-1
Transformer without PE	1.14e-3	8.66e-5	7.61e-5	5.55e-5
Transformer with PE	**5.17e-6**	**8.76e-7**	3.99e-5	**2.46e-7**
MambaFormer	7.30e-6	3.37e-6	**1.03e-5**	3.79e-7
6 Mamba Blocks + 1 Standard Hybrid	1,99e-2	1.37e-2	1.54e-3	5.86e-5

Table 3: Test loss (mean squared error) on vector MQAR and respective model configurations. We test Transformers with and without Positional Encoding (PE). All models have 4 layers with roughly the same number of parameters. We consider the "6 Mamba Blocks + 1 Standard Hybrid" model as 4 layers since one Mamba layer consists of two Mamba blocks as described in Figure 6.

Finding 4: *Mamba struggles to retrieve vectors within its context in vector MQAR, a task Transformer can easily perform.*

From Table 3, 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.

Note that the models trained with 16 queries have lower test loss than models trained with 4 queries. We conjecture that for a single sequence of data that represents an MQAR task, each $(\mathbf{q}, \mathbf{v})$ pair can be thought of as a training sample. Hence a sequence with 16 queries contains more training samples than that of a sequence with 4 queries. This also shows that having more queries does not necessarily make the task harder. Notably, our setting is more challenging than token-based MQAR, as we sample new random vectors each batch. Similar findings on retrieval were observed in Arora et al. (2023).

Finding 5: *Mamba can in-context learn sparse parity, a task Transformer cannot perform.*

While Mamba fails on simple retrieval tasks such as MQAR, the tables turn for the task of learning sparse parity (Figure 5). Transformer fails to do better than random guessing, in line with evidence from prior work (Bhattamishra et al., 2020, 2023; Hahn, 2020). We confirm this is the case for Transformer sizes of embedding dimensions up to 768 and up to 24 layers when trained for at most 1 million iterations. 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 is able to solve parity as well, showing comparable performance to that of Mamba; this indicates that proper convolution or gating may be more important than input-dependent selection for learning parity. Given that only Transformer cannot perform better than random, sequential computations of recurrent models seem more advantageous for learning parity. Finally, our result hints at that the initial (causal) convolution that Mamba provides before the attention layer may be crucial to solving parities, a similar phenomenon observed for Vision Transformers in computer vision tasks (Yu et al., 2022).

Any algorithm for learning parities requires either a super-linear memory of $\omega(d)$ or a super-polynomial number of samples in d (Raz, 2016; Kol et al., 2017). While Transformer is known to have better memory than Mamba due to its quadratic attention mechanism, our result on learning

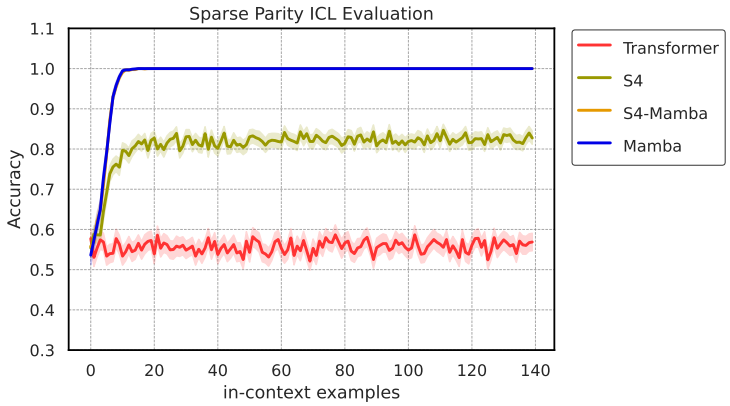
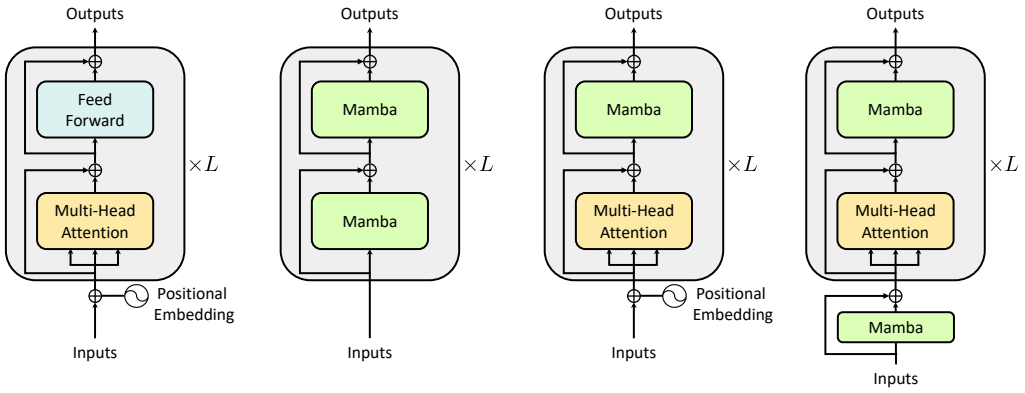


Figure 5: Although Transformer struggles to learn the task, Mamba and S4-Mamba can learn sparse parity of $d = 10$ and $k = 2$ (S4-Mamba accuracy plot is hidden behind that of Mamba). Each model is trained with an embedding dimension of 256 and depth of 12 layers (approximately 10M parameters) up to 500,000 iterations. Transformer struggles to learn even up to an embedding dimension of 768 and 24 layers and 1M iterations.

sparse parities brings forth the question on how different architectures may utilize its memory differently in terms of function approximation. We leave the theoretical and empirical question of which architectural component allows for learning parities as an avenue for further study.

5 The Advantage of Hybrid Architectures for In-context Learning



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

Figure 6: Model architectures. (a) and (b) are 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) depicts the proposed architecture, namely MambaFormer, which replaces the Positional Encoding with a Mamba block. We denote 2 blocks of either Mamba, multi-head Attention, or a feed forward network as 1 layer.

In the previous section, we have observed that Transformers perform better than SSMs in certain tasks such as learning decision trees or retrieval, while SSMs excel in others, such as learning sparse parities or heavy-outlier linear regression, possibly due to its recurrent nature. However, can we achieve the best of both worlds without sacrificing performance in our suite of ICL tasks?

We answer this in the affirmative; in this section, we investigate two hybrid architectures that combine Transformer and Mamba, namely Standard Hybrid and MambaFormer as illustrated in Figure 6. Standard Hybrid is the architecture of interleaving MHA and Mamba by replacing the MLP block with Mamba. MambaFormer is nearly identical to Standard Hybrid but with an additional Mamba

block as its initial layer. This removes the need of initial positional encoding as a Mamba block's recurrent nature encodes positional information.

Although many works have found that interleaving multi-head attention and LTI SSMs beneficial (Zuo et al., 2022; Mehta et al., 2022; Pilault et al., 2023), interestingly Gu & Dao (2023) have not found significant benefits of interleaving.

In the following results, however, we show that we can indeed reach competitive performance in our suite of ICL tasks by interleaving Attention and Mamba blocks. MambaFormer achieves comparable performance to that of Transformer or Mamba, while excelling in both sparse parity and retrieval, tasks unsolvable by Transformer and Mamba, respectively. We discover that the key ingredient is having Mamba as the first layer.

5.1 Simultaneously learning parities and retrieval

Finding 6: *MambaFormer can in-context learn sparse parity; moreover, having the initial layer as a Mamba block is significantly effective.*

As highlighted in Bhattamishra et al. (2023); Barak et al. (2022), learning sparse parity in-context seems to be difficult for Transformer and some SSMs like Hyena. Yet interestingly, as seen in Figure 7, MambaFormer successfully learns parity as quickly as Mamba in terms of sample complexity. While the Standard Hybrid model is also capable, it exhibits much worse sample efficiency.

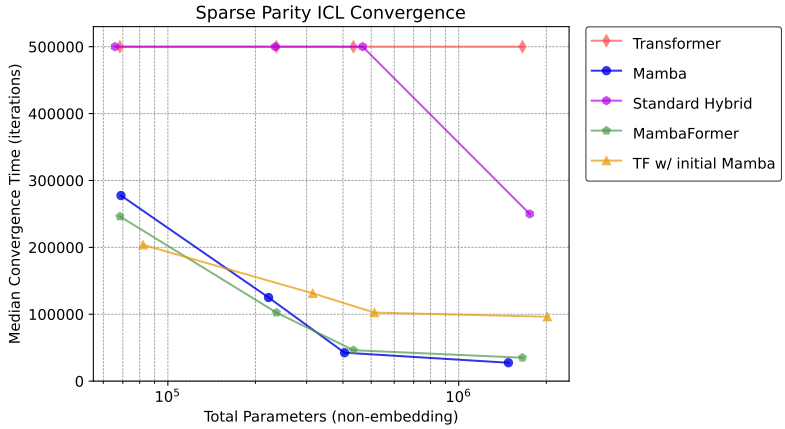


Figure 7: Median convergence time of learning parity over 5 random seeds for max. 500k iterations, where 500k convergence time signifies failed learning. Having the initial layer as Mamba is essential for efficiently learning parities. Tested model configurations are specified in Appendix A.

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 the order of layers in interleaving matters as shown in Press et al. (2022), but also that Mamba can complement Transformer without hurting performance in ICL. This result brings up intriguing differences between the function learning capabilities of Attention and Mamba; we leave this question up for further study.

Finding 7: *MambaFormer can perform retrieval as well as Transformer, closing the performance gap between Mamba and Transformer.*

The gap between Mamba and Transformer in vector MQAR is likely 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 context SSMs store at each state depends on the dimension of hidden state as the hidden states capture the important information in the context. In contrast, attention leverages all tokens in its input context, allowing Transformers and hybrid models to conveniently retrieve corresponding key-value pairs through pairwise computations.

On the other hand, SSMs would eventually be overwhelmed if the number of key-value pairs increases substantially.

To close the gap in the vector MQAR task between Mamba and Transformer without sacrificing efficiency too much, we add one attention layer within the Mamba layers. 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 Table 3, Mamba model gains a significant improvement in vector MQAR by having one Standard Hybrid. We further test MambaFormer on the same task and find that MambaFormer almost entirely closes the gap to transformer in vector MQAR task.

5.2 All-in-one ICL performance

Finding 8: *Both hybrid models perform as well as Transformer and Mamba in our suite of ICL tasks (or even better sometimes).*

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 3). 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 3 left).

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 3 top right) while Transformer learns better when there is a subspace of outlier vectors (Figure 3 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. Given our results, it will be interesting to see how hybrid architectures perform in other kinds of ICL tasks, especially in-context language benchmarks such as Xie et al. (2021); Hahn & Goyal (2023); Akyürek et al. (2024). In turn, we explore formal language ICL capabilities in the following subsection.

5.3 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.

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 4: 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).

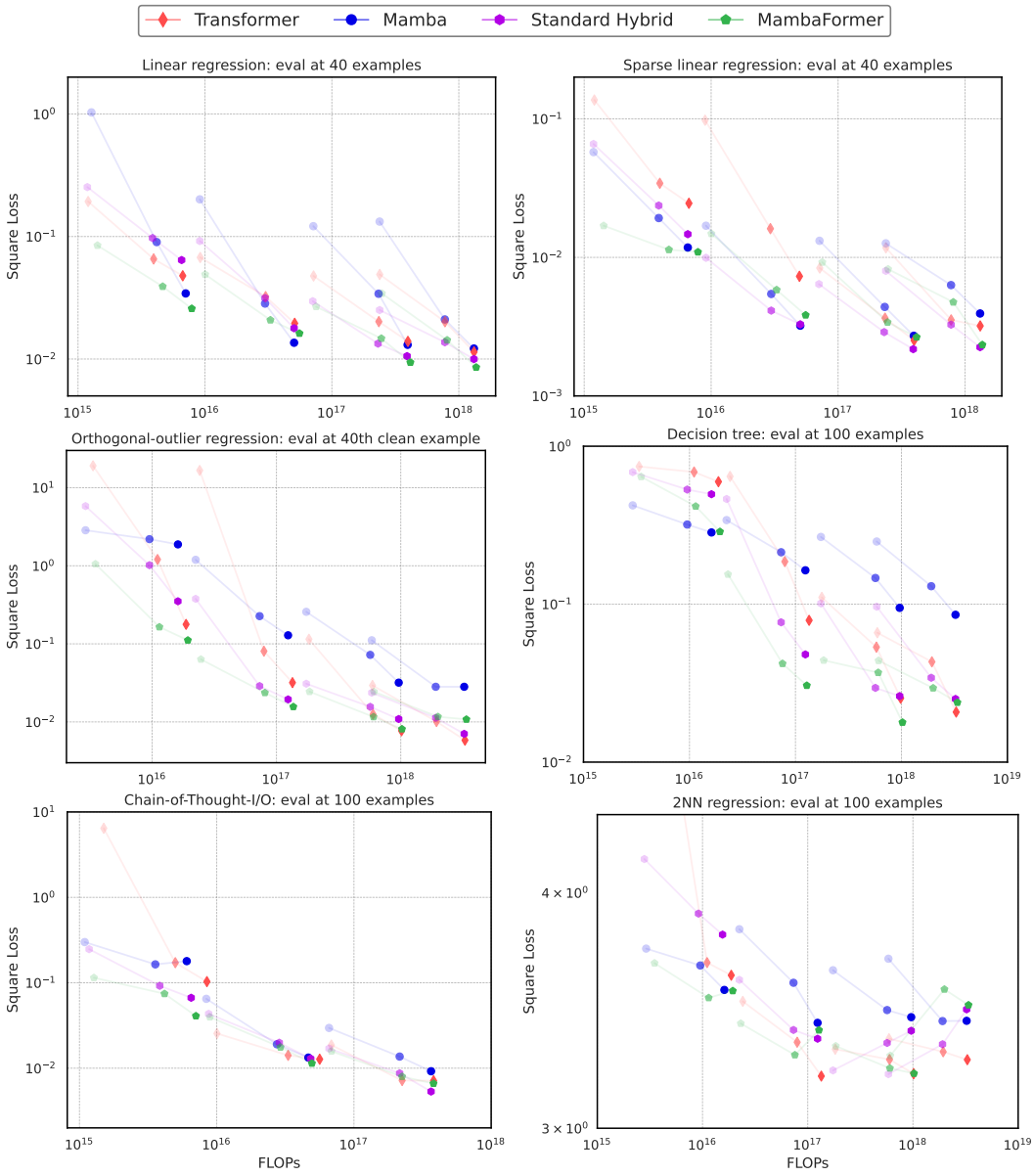


Figure 8: A suite of ICL tasks ran for Transformer, Mamba, and hybrid architectures where each color represents a different architecture. More transparent points indicate earlier stages of training; plotted models are trained {100k, 300k, 500k} iterations. Standard Hybrid and MambaFormer are hybrid models of Transformer and Mamba defined in Section 5.

Finding 9: *Hybrid models perform as well as, or outperform, Transformer and Mamba in formal language ICL, as exemplified in Tables 4 and 5.*

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

RegBench (trained 15 epochs)	**Train PPL** (↓)	**Valid PPL** (↓)	**Acc.** (↑)
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** (↓)	**Valid PPL** (↓)	**Acc.** (↑)
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 5: 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 Section 3.2 for how accuracy is measured. * denotes reported accuracy in Akyürek et al. (2024).

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.

6 Discussion

In this work, we have provided a comprehensive investigation of in-context learning with state-space models (SSMs) and contrasted them with the Transformer architecture. Our study has revealed that SSMs, especially Mamba, are capable in-context learners. On the other hand, our evaluations revealed that neither SSMs nor Transformers are great at all tasks: SSMs struggle with decision tree and retrieval tasks whereas Transformers struggle with sparse parity. This has led us to the hybrid architecture MambaFormer which achieves a best-of-both-worlds performance on our ICL suite.

Future research directions include exploring (1) how the performance on our ICL suite correlates with general language modeling capabilities, such as perplexity on standard NLP benchmarks, (2) 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.

Impact Statement

This paper provides a comprehensive study of language modeling architectures which help identify their weaknesses, strengths, and provide recipes for new architectures. The outcomes of this work will potentially facilitate efficiency and architectural improvements for large language models.

Acknowledgement

The work of Dimitris Papailiopoulos is supported in part by ONR Grant No. N00014-21-1-2806 and No. N00014-23-1-2848. The work of Samet Oymak is supported in part by NSF CAREER Award CCF-2046816. The work of Jaeseung Park was supported by KRAFTON AI Fellowship. The authors would like to thank Byeongju Kim and Seongjun Yang for helpful discussion and Gibbeum Lee for valuable feedback on an early draft of this paper.

References

Ahn, K., Cheng, X., Daneshmand, H., and Sra, S. Transformers learn to implement preconditioned gradient descent for in-context learning. *arXiv preprint arXiv:2306.00297*, 2023. 3

Akyürek, E., Schuurmans, D., Andreas, J., Ma, T., and Zhou, D. What learning algorithm is in-context learning? investigations with linear models. In *The Eleventh International Conference on Learning Representations*, 2022. 1, 3

Akyürek, E., Wang, B., Kim, Y., and Andreas, J. In-context language learning: Architectures and algorithms. *arXiv preprint arXiv:2401.12973*, 2024. URL https://arxiv.org/abs/2401.12973. 3, 4, 6, 13, 15

Arora, S., Eyuboglu, S., Timalsina, A., Johnson, I., Poli, M., Zou, J., Rudra, A., and Ré, C. Zoology: Measuring and improving recall in efficient language models. *arXiv preprint arXiv:2312.04927*, 2023. URL https://arxiv.org/abs/2312.04927. 3, 4, 6, 10

Bai, Y., Chen, F., Wang, H., Xiong, C., and Mei, S. Transformers as statisticians: Provable in-context learning with in-context algorithm selection. *arXiv preprint arXiv:2306.04637*, 2023. 1

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. 12

Beltagy, I., Peters, M. E., and Cohan, A. Longformer: The long-document transformer. *arXiv preprint arXiv:2004.05150*, 2020. 3

Bhattamishra, S., Ahuja, K., and Goyal, N. 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. 10, 14

Bhattamishra, S., Patel, A., Blunsom, P., and Kanade, V. 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. 3, 6, 10, 12, 24

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. 1

Chan, S., Santoro, A., Lampinen, A., Wang, J., Singh, A., Richemond, P., McClelland, J., and Hill, F. Data distributional properties drive emergent in-context learning in transformers. *Advances in Neural Information Processing Systems*, 35:18878–18891, 2022. 1

Dai, D., Sun, Y., Dong, L., Hao, Y., Ma, S., Sui, Z., and Wei, F. 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. 3

Dao, T., Fu, D. Y., Saab, K. K., Thomas, A. W., Rudra, A., and Ré, C. Hungry hungry hippos: Towards language modeling with state space models. *arXiv preprint arXiv:2212.14052*, 2022. URL https://arxiv.org/abs/2212.14052. 2, 4

Deletang, G., Ruoss, A., Grau-Moya, J., Genewein, T., Wenliang, L. K., Catt, E., Cundy, C., Hutter, M., Legg, S., Veness, J., et al. Neural networks and the chomsky hierarchy. In *The Eleventh International Conference on Learning Representations*, 2022. 14

Gao, L., Biderman, S., Black, S., Golding, L., Hoppe, T., Foster, C., Phang, J., He, H., Thite, A., Nabeshima, N., Presser, S., and Leahy, C. The pile: An 800gb dataset of diverse text for language modeling, 2020. 2

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. 1, 2, 3, 5, 20, 24

Giannou, A., Rajput, S., Sohn, J.-y., Lee, K., Lee, J. D., and Papailiopoulos, D. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pp. 11398–11442. PMLR, 2023. 1

Gu, A. and Dao, T. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023. URL https://arxiv.org/abs/2312.00752. 1, 2, 4, 5, 9, 11, 12, 15, 20

Gu, A., Goel, K., Gupta, A., and Ré, C. On the parameterization and initialization of diagonal state space models. In *Advances in Neural Information Processing Systems*, volume 35, pp. 35971–35983, 2022a. 3, 4

Gu, A., Goel, K., and Ré, C. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022*, 2022b. URL https://openreview.net/forum?id=uYLFoz1vlAC. 2, 3

Hahn, M. Theoretical limitations of self-attention in neural sequence models. *Transactions of the Association for Computational Linguistics*, 8:156–171, 2020. 10

Hahn, M. and Goyal, N. A theory of emergent in-context learning as implicit structure induction. *arXiv preprint arXiv:2303.07971*, 2023. URL https://arxiv.org/abs/2303.07971. 13

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. 4

Kaplan, J., McCandlish, S., Henighan, T., Brown, T. B., Chess, B., Child, R., Gray, S., Radford, A., Wu, J., and Amodei, D. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020. 1, 5, 21

Katharopoulos, A., Vyas, A., Pappas, N., and Fleuret, F. Transformers are rnns: Fast autoregressive transformers with linear attention. In *Proceedings of the 37th International Conference on Machine Learning, ICML 2020*, volume 119 of *Proceedings of Machine Learning Research*, pp. 5156–5165, 2020. URL http://proceedings.mlr.press/v119/katharopoulos20a.html. 2, 4

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5, 20

Kol, G., Raz, R., and Tal, A. Time-space hardness of learning sparse parities. In *Proceedings of the 49th Annual ACM SIGACT Symposium on Theory of Computing*, pp. 1067–1080, 2017. 10

Li, Y., Ildiz, M. E., Papailiopoulos, D., and Oymak, S. Transformers as algorithms: Generalization and stability in in-context learning, 2023a. 1

Li, Y., Sreenivasan, K., Giannou, A., Papailiopoulos, D., and Oymak, S. Dissecting chain-of-thought: Compositionality through in-context filtering and learning. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. 6, 21

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Transformers learn shortcuts to automata. In *The Eleventh International Conference on Learning Representations*, 2022. 1

Liu, B., Ash, J. T., Goel, S., Krishnamurthy, A., and Zhang, C. Exposing attention glitches with flip-flop language modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 4

Loshchilov, I. and Hutter, F. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2018. 22

Mahankali, A., Hashimoto, T. B., and Ma, T. 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. 3

Mehta, H., Gupta, A., Cutkosky, A., and Neyshabur, B. Long range language modeling via gated state spaces. *arXiv preprint arXiv:2206.13947*, 2022. 12

Min, S., Lewis, M., Zettlemoyer, L., and Hajishirzi, H. 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, 2022a. 1, 3

Min, S., Lyu, X., Holtzman, A., Artetxe, M., Lewis, M., Hajishirzi, H., and Zettlemoyer, L. Rethinking the role of demonstrations: What makes in-context learning work? *arXiv preprint arXiv:2202.12837*, 2022b. URL https://arxiv.org/abs/2202.12837. 1

Muennighoff, N., Rush, A. M., Barak, B., Scao, T. L., Tazi, N., Piktus, A., Pyysalo, S., Wolf, T., and Raffel, C. Scaling data-constrained language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=j5BuTrEj35. 1, 21

Olsson, C., Elhage, N., Nanda, N., Joseph, N., DasSarma, N., Henighan, T., Mann, B., Askell, A., Bai, Y., and et al., A. C. In-context learning and induction heads. *arXiv preprint arXiv:2209.11895*, 2022. URL https://arxiv.org/abs/2209.11895. 3, 4, 6

Peng, B., Alcaide, E., Anthony, Q., Albalak, A., Arcadinho, S., Cao, H., Cheng, X., Chung, M., Grella, M., Kiran, K. G., et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023. URL https://arxiv.org/abs/2305.13048. 2, 4

Pilault, J., Fathi, M., Firat, O., Pal, C., Bacon, P.-L., and Goroshin, R. Block-state transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. 12

Poli, M., Massaroli, S., Nguyen, E., Fu, D. Y., Dao, T., Baccus, S., Bengio, Y., Ermon, S., and Ré, C. Hyena hierarchy: Towards larger convolutional language models. *arXiv preprint arXiv:2302.10866*, 2023. URL https://arxiv.org/abs/2302.10866. 2, 4

Press, O., Zhang, M., Min, S., Schmidt, L., Smith, N. A., and Lewis, M. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*, 2022. 12

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., Sutskever, I., et al. Language models are unsupervised multitask learners. *OpenAI blog*, 1(8):9, 2019. 20

Ravi, S. and Larochelle, H. Optimization as a model for few-shot learning. In *International conference on learning representations*, 2016. 1, 3

Raz, R. Fast learning requires good memory: A time-space lower bound for parity learning. In *2016 IEEE 57th Annual Symposium on Foundations of Computer Science (FOCS)*, pp. 266–275. IEEE, 2016. 10

Schaeffer, R., Miranda, B., and Koyejo, S. Are emergent abilities of large language models a mirage? In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ITw9edRDlD. 1

Schmidhuber, J., Zhao, J., and Wiering, M. Shifting inductive bias with success-story algorithm, adaptive levin search, and incremental self-improvement. *Machine Learning*, 28:105–130, 1997. 1

Sun, Y., Dong, L., Huang, S., Ma, S., Xia, Y., Xue, J., Wang, J., and Wei, F. Retentive network: A successor to transformer for large language models. *arXiv preprint arXiv:2307.08621*, 2023. URL https://arxiv.org/abs/2307.08621. 2, 4

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023. 22

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. 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. 1

von Oswald, J., Niklasson, E., Randazzo, E., Sacramento, J., Mordvintsev, A., Zhmoginov, A., and Vladymyrov, M. Transformers learn in-context by gradient descent. In *International Conference on Machine Learning*, pp. 35151–35174. PMLR, 2023a. 1

von Oswald, J., Niklasson, E., Schlegel, M., Kobayashi, S., Zucchet, N., Scherrer, N., Miller, N., Sandler, M., Vladymyrov, M., Pascanu, R., et al. Uncovering mesa-optimization algorithms in transformers. *arXiv preprint arXiv:2309.05858*, 2023b. URL https://arxiv.org/abs/2309.05858. 1, 3

Wang, S., Li, B. Z., Khabsa, M., Fang, H., and Ma, H. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020. 3

Wei, J., Tay, Y., Bommasani, R., Raffel, C., Zoph, B., Borgeaud, S., Yogatama, D., Bosma, M., Zhou, D., Metzler, D., et al. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*, 2022. 1

Xie, S. M., Raghunathan, A., Liang, P., and Ma, T. An explanation of in-context learning as implicit bayesian inference. In *International Conference on Learning Representations*, 2021. 3, 4, 6, 13, 14

Yang, L., Lee, K., Nowak, R., and Papailiopoulos, D. Looped transformers are better at learning learning algorithms, 2023a. 1

Yang, S., Wang, B., Shen, Y., Panda, R., and Kim, Y. Gated linear attention transformers with hardware-efficient training. *arXiv preprint arXiv:2312.06635*, 2023b. 2, 4

Yu, W., Luo, M., Zhou, P., Si, C., Zhou, Y., Wang, X., Feng, J., and Yan, S. Metaformer is actually what you need for vision. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 10819–10829, 2022. 10

Zhai, S., Talbott, W., Srivastava, N., Huang, C., Goh, H., Zhang, R., and Susskind, J. An attention free transformer. *arXiv preprint arXiv:2105.14103*, 2021. 2

Zhou, H., Bradley, A., Littwin, E., Razin, N., Saremi, O., Susskind, J., Bengio, S., and Nakkiran, P. What algorithms can transformers learn? a study in length generalization. *arXiv preprint arXiv:2310.16028*, 2023. 1

Zuo, S., Liu, X., Jiao, J., Charles, D., Manavoglu, E., Zhao, T., and Gao, J. Efficient long sequence modeling via state space augmented transformer. *arXiv preprint arXiv:2212.08136*, 2022. 12

A Experimental Setup

In this section, we describe our experimental design and configured setup. Our code and detailed implementations can be found in https://github.com/krafton-ai/mambaformer-icl.

A.1 Model architectures

We focus on decoder-only Transformer models, particularly those from the GPT-2 family (Radford et al., 2019), Mamba (Gu & Dao, 2023), and their hybrid variants, including Standard Hybrid and MambaFormer configurations. These models are evaluated across a range of sizes, as detailed in Table 6. Transformer layers consist of a Multi-Head Attention (MHA) block followed by a Multilayer Perceptron (MLP) block. Mamba models consist of two Mamba blocks per layer. The hybrid variants merge these approaches, combining a single MHA block with a Mamba block. For MHA blocks, we use 8 number of heads. Refer to Figure 6 for a visualization of the architectures considered.

A.2 Model training and configurations

Size group	(# layers, embed dim)	Transformer	Mamba	S4*
Large	{(12, 768)}	86M	90M	88M
Medium	{(8, 512), (32, 256)}	25M	27M	26M
Small	{(4, 256), (16, 128)}	3M	3M	3M
X-Small	{(2, 128), (8, 64), (32, 32)}	420K	460K	430K
		S4-Mamba	Standard Hybrid	MambaFormer
Large	{(12, 768)}	86M	74M	77M
Medium	{(8, 512), (32, 256)}	26M	22M	24M
Small	{(4, 256), (16, 128)}	3M	3M	3.2M
X-Small	{(2, 128), (8, 64), (32, 32)}	430K	400K	480K

Table 6: The four size groups of model architectures we have used for our experiments. For each size group, we run various learning rates in addition to training 'narrower and deeper' models of the same size and FLOPs. We keep the number of heads fixed at 8. We do not train models deeper than 32 layers, as we have observed that accuracy is best in between 4 to 32 layers according to Appendix C. *For S4 models, the embedding dimensions were multiplied by a factor of 1.75 to match parameters.

We train all of our models on A100-SXM4-40GB GPUs for 500,000 training steps on all tasks, except for vector-valued MQAR, in which the models were trained for 300,000 training steps. We use Adam optimizer (Kingma & Ba, 2014) with a fixed learning rate. The default learning rate is set to $1e-4$. We also search various learning rates in $\{5e-5, 2e-4, 4e-4\}$. We observe that the training procedure is the most sensitive to choosing the right learning rate. In particular, as the number of parameters of the models increases, the training procedure is prone to gradient explosions, especially in Mamba and hybrid architecutres. Hence, we clip the gradient norm, with values in $\{5.0, 10.0, 50.0\}$.

As for the train and test data, we fix the dimension of $\mathbf{x}$ to be 20, and fix the batch size to be 64. As suggested in Garg et al. (2022), we also observe that curriculum is helpful in certain ICL tasks. We adopt a curriculum; every 2000 steps, we increase both the dimension of $\mathbf{x}$ and the number of points within the input context.

For model configurations, we mainly follow the four size groups of Transformers listed below (Table 6). As explained in Figure 6, we denote 2 blocks of Mamba, multi-head attention, or a feed forward network as 1 layer. This roughly aligns the number of parameters in Transformer, Mamba and S4-Mamba. For Standard Hybrid and MambaFormer, we follow the same design. This yields in models with less parameters from the lack of feed forward networks. However, both models showed strong performance to other models and hence the model configurations were kept as is. For S4, however, we further increase the embedding dimension by a factor of 1.75 to match the number of parameters of transformers.

As demonstrated, trading embedding dimension (narrow width) for additional layers (greater depth) allows for diverse model configurations while maintaining the same total parameter count. We

conducted an ablation study on linear regression to explore the impact of width versus depth on ICL tasks and to identify the optimal configurations for peak performance, given the same FLOPs. For the rationale behind our experimental choices in these configurations, refer to Appendix C as detailed in Table 6 for ICL tasks. Furthermore, in Table 6, we also explore whether the choice of optimizers affect our results and alter our conclusions.

A.3 Chain-of-Thought-I/O settings

Table 7 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.

Model	# layers	embed dim	# heads (MHA)
standard	12	256	8
small	6	128	4
tiny	3	64	2

Table 7: Model configurations for Chain-of-Thought-I/O experiments in Figure 4.

A.4 Many-outlier regression settings

We run many-outlier regression on two size groups listed below in Table 8. The configurations below required multi-GPU training due to its long context length of 1024 ($N = 512$).

Model size	# of layers	Embed dim	# of heads (MHA)
Regular	6	512	8
Mini	4	256	8

Table 8: Model configurations for many-outlier regression ICL in Figure 3.

A.5 Vector-valued MQAR

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.

B FLOPs Computation

We count the number of multiplications in a Mamba block and a Transformer block in Table 9 and Table 10. We assume batch size $B = 1$. To calculate FLOPs, we follow the similar methodology used in Kaplan et al. (2020); Muennighoff et al. (2023) and multiply the number of multiplications by 6 to account for the multiply-accumulate cost in both forward and backward pass. Note that a Standard Hybrid block is an attention block stacked with a Mamba block, so the number of multiplications in a Standard Hybrid block is $10LD^2 + 2L^2D$, ignoring the linear terms.

C Exploring effects of width vs. depth in ICL

In this section, we empirically validate our experimental design and setup, detailed in Appendix A, through a comprehensive comparison of wide and shallow networks versus deep and narrow networks, to investigate the effect of model design while fixing the total number of FLOPs. Additionally, we investigate whether comparing different architectures with Adam only is a fair comparison. We evaluate the choice of optimizers, specifically comparing Adam–used in our main experiments–with

	Number of multiplications
QKV projection	$3LD^2$
Outer product and multiply V	$2L^2D$
Outer projection	LD^2
FFN with `ffw_width=4`	$8LD^2$

Table 9: Number of multiplications in a Transformer block. L denotes the input sequence length and D denotes the hidden dimension of the model.

	Number of multiplications
Input projection	$2LED^2$
SSM	$7LEDN + 4LED$
Output projection	LED^2

Table 10: Number of multiplications in a Mamba block. L, D are the same as Table 9. N represents the state size of the SSM and E denotes the expansion factor of the hidden dimension within each Mamba block. We assume $E = 2$ and $N = 16$.

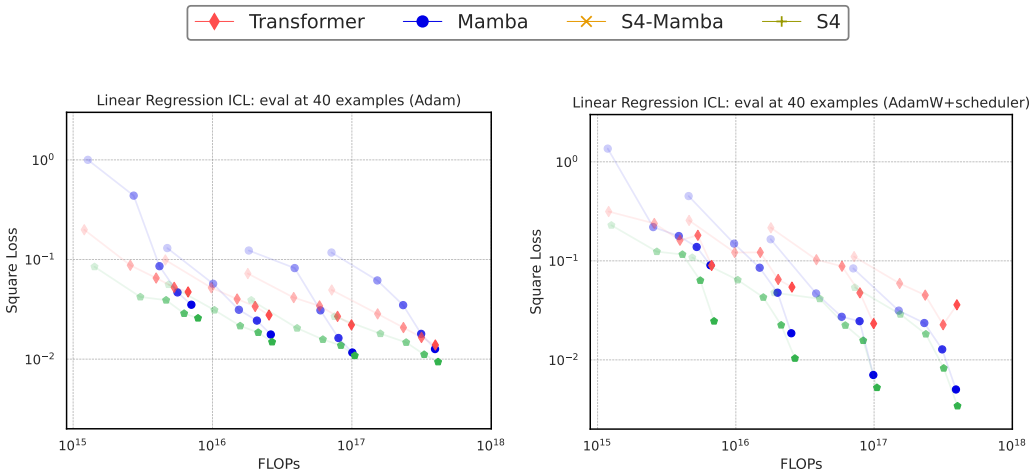


Figure 9: Performance of Transformer, Mamba, and MambaFormer on linear regression ICL, using two different optimizers. The left side shows results obtained using the Adam optimizer, while the right side shows results obtained using AdamW coupled with linear warmup and cosine decay scheduler. We observe that Mamba and MambaFormer gain considerably more than Transformer when following the new optimizer setup. See Appendix C for implementation details.

AdamW (Loshchilov & Hutter, 2018) coupled with a linear warmup and cosine decay scheduler, which is the standard in language model pretraining for Transformer-based models (Touvron et al., 2023).

Experiment settings. We conduct our ablation study on the task of linear regression. We explore four size groups of models, each size group with four different configurations for width and depth as seen in Figure 10. We study the three models of interest: Transformer, Mamba, and MambaFormer. For Transformer and MambaFormer, we keep the dimension of each head constant at 16; for instance, a model with embedding dimension 256 has a total of 16 heads.

For the new optimizer, we use AdamW with the following hyperparameters: $\beta_1 = 0.9, \beta_2 = 0.95$. The final learning rate of the cosine decay scheduler is equal to 10% of the predefined learning rate. We use linear warmup for the first 50,000 steps of training, starting from 10% of the learning rate. For

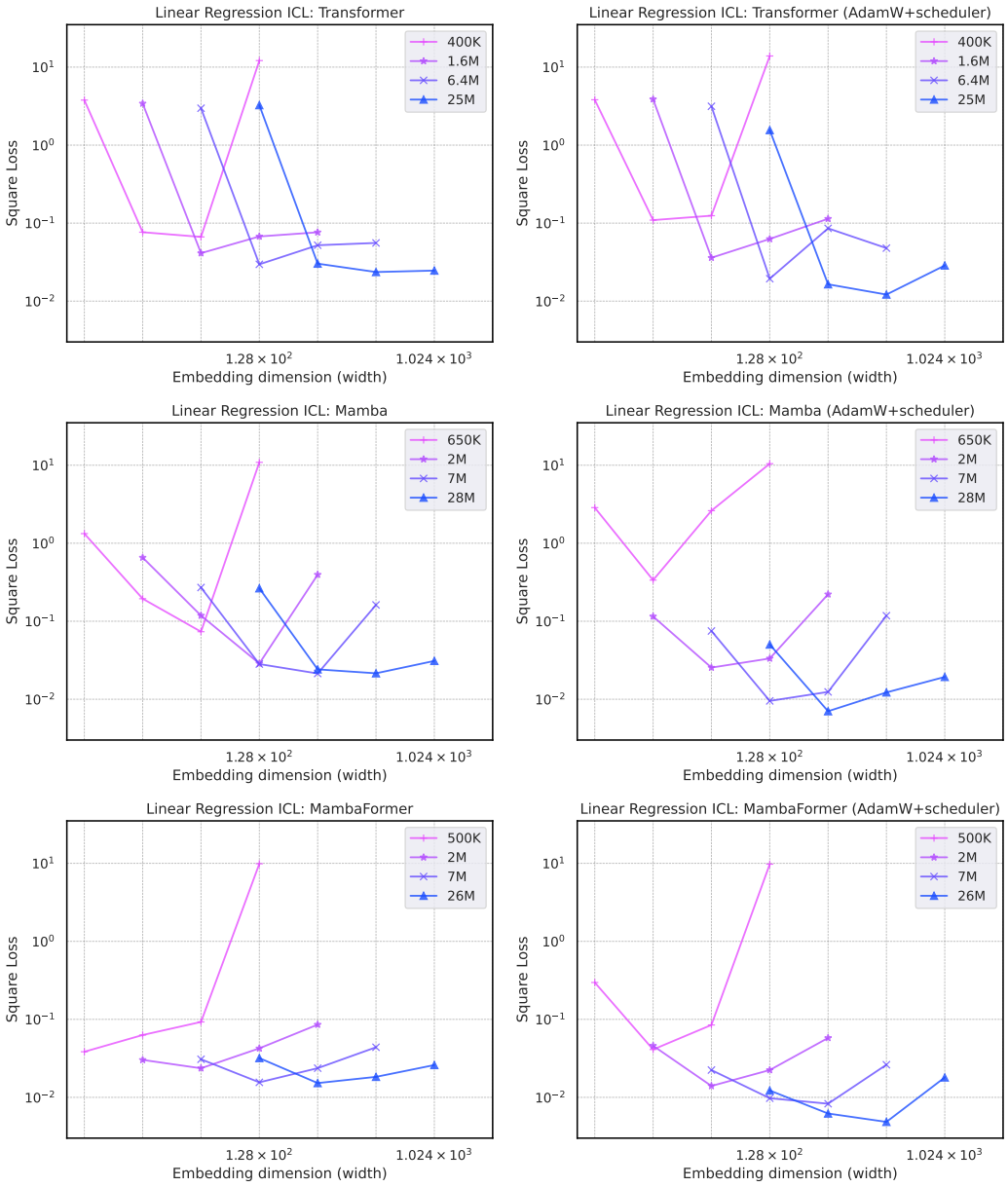


Figure 10: Performance of models with varying configurations on linear regression ICL eval at 40 examples. Each color of the legend represents a model group of fixed total number of (non-embedding) parameters. The models differ in the number of layers, specifically 2, 8, 32, and 128 layers for each connected line, *i.e.*, the model with the widest width has 2 layers, and the model with the narrowest width has 128 layers. **(Left)** Adam optimizer as used in Section 3.1. **(Right)** AdamW with linear warmup and cosine decay.

each model configuration, we choose the best performance out of learning rates $\{5e{-}5, 1e{-}4, 2e{-}4\}$. The remaining experiment details are identical to those described in Section 3.

Finding 10: *Shallow models struggle to learn regression, especially for Transformers. Very deep models also struggle, but less so if sufficient width is provided.*

For Transformers, we observe that 2-layer models struggle with learning linear regression, even after seeing 40 in-context examples, regardless of depth (see Figure 10). It appears that Transformers have

a minimum layer threshold to effectively learn regression tasks. While less pronounced, a similar pattern is observed with Mamba; however, MambaFormer demonstrates proficiency in learning regression with as few as 2 layers. Conversely, models that are excessively deep, with 128 layers, face difficulties in learning when their width is insufficient, and they also require significantly more training time due to prolonged forward and backward passes. Consequently, exploring layer counts between 4 and 32 on other tasks seems to be sufficient in identifying the optimal model configuration given fixed total FLOPs, as detailed in Table 6.

> **Finding 11:** *AdamW coupled with a scheduler improves performance, especially for larger models and for Mamba and MambaFormer. Yet, our conclusion remains the same and is agnostic to the choice of optimizer.*

As seen in Figure 9 and Figure 10, AdamW combined with a learning rate scheduler helps learning stronger models, as one may expect from its wide adoption in large language model training. The benefits of the new optimizer increase with model size across different architectures. However, we note that Mamba and MambaFormer gain more from AdamW plus a scheduler compared to Transformer. Anecdotally, the learning rate schedule is particularly beneficial for training Mamba, as deeper and larger Mamba models tend to experience gradient explosion issues.

The two primary conclusions from our work were: (1) Mamba is capable of performing ICL effectively, and (2) MambaFormer successfully combines high performance with the best attributes of both Transformer and Mamba, sometimes even surpassing them. Our empirical tests suggest that the choice of optimizer and scheduler does not fundamentally alter these conclusions. In fact, our ablation study indicates that Mamba and MambaFormer might perform even better under these conditions. Therefore, we decided to use the Adam optimizer in our main experiments, consistent with prior work (Garg et al., 2022; Bhattamishra et al., 2023) on Transformer ICL.