Jamba:
A Hybrid Transformer-Mamba Language Model

Opher Lieber[*] **Barak Lenz**[*] **Hofit Bata** **Gal Cohen** **Jhonathan Osin**
Itay Dalmedigos **Erez Safahi** **Shaked Meirom** **Yonatan Belinkov**
Shai Shalev-Shwartz **Omri Abend** **Raz Alon** **Tomer Asida**
Amir Bergman **Roman Glozman** **Michael Gokhman** **Avashalom Manevich**
Nir Ratner **Noam Rozen** **Erez Shwartz** **Mor Zusman** **Yoav Shoham**

AI21labs

Abstract

We present Jamba, a new base large language model based on a novel hybrid Transformer-Mamba mixture-of-experts (MoE) architecture. Specifically, Jamba interleaves blocks of Transformer and Mamba layers, enjoying the benefits of both model families. MoE is added in some of these layers to increase model capacity while keeping active parameter usage manageable. This flexible architecture allows resource- and objective-specific configurations. In the particular configuration we have implemented, we end up with a powerful model that fits in a single 80GB GPU. Built at large scale, Jamba provides high throughput and small memory footprint compared to vanilla Transformers, and at the same time state-of-the-art performance on standard language model benchmarks and long-context evaluations. Remarkably, the model presents strong results for up to 256K tokens context length. We study various architectural decisions, such as how to combine Transformer and Mamba layers, and how to mix experts, and show that some of them are crucial in large scale modeling. We also describe several interesting properties of these architectures which the training and evaluation of Jamba have revealed, and plan to release checkpoints from various ablation runs, to encourage further exploration of this novel architecture. We make the weights of our implementation of Jamba publicly available under a permissive license.

Model: `https://huggingface.co/ai21labs/Jamba-v0.1`

1 Introduction

We introduce Jamba, a new publicly available large language model. Jamba is based on a novel hybrid architecture, which combines Transformer layers [46] with Mamba layers [16], a recent state-space model [17, 18], as well as a mixture-of-experts (MoE) component [13, 41]. Jamba thus combines two orthogonal architectural designs that together give it improved performance and higher throughput, while maintaining a manageable memory footprint. The 7B-based Jamba model (12B active parameters, 52B total available parameters) we are releasing was designed to fit in a single 80GB GPU, but the Jamba architecture supports other design choices, depending on one's hardware and performance requirements.

[*]Equal contribution.

The fundamental novelty of Jamba is its hybrid Transformer-Mamba architecture (though see mention below of recent related efforts). Despite the immense popularity of the Transformer as the predominant architecture for language models, it suffers from two main drawbacks. First, its high memory and compute requirements hinders the processing of long contexts, where the key-value (KV) cache size becomes a limiting factor. Second, its lack of a single summary state entails slow inference and low throughput, since each generated token performs a computation on the entire context. In contrast, older recurrent neural network (RNN) models, which summarize an arbitrarily long context in a single hidden state, do not suffer from these limitations. RNN models have their own shortcomings, however. They are costly to train since training cannot be parallelized across time steps. And they struggle with long distance relationships, which the hidden state captures to only a limited extent.

Recent state space models (SSMs) like Mamba are more efficient to train than RNNs and are more capable at handling long distance relationships, but still lag behind the performance of comparably sized Transformer language models. Taking advantage of both model families, Jamba combines Transformer and Mamba layers, at a certain ratio. Varying the ratio of Transformer/Mamba layers allows balancing memory usage, efficient training, and long context capabilities.

A few other recent attempts to combine Attention and SSM modules are worth noting. [50] mixes an S4 layer [17] with a local attention layer, followed by a sequence of local attention layers; it shows experiments with small models and simple tasks. [16] reports that interleaving Mamba and attention layers is only slightly better than pure Mamba in terms of perplexity, with models up to 1.3B parameters. [33] starts with an SSM layer followed by chunk-based Transformers, with models up to 1.3B showing improved perplexity. [12] adds an SSM layer before the self-attention in a Transformer layer, while [38] adds the SSM after the self-attention, both showing improvements on speech recognition. [32] replaces the MLP layers in the Transformer by Mamba layers, and shows benefits in simple tasks. These efforts are different from Jamba both in the particular way in which the SSM component is mixed with the attention one, and in the scale of implementation. Closest are perhaps H3 [14], a specially designed SSM that enables induction capabilities, and a generalization called Hyena [35]. The former proposed a hybrid architecture that replaces the second and middle layers with self-attention, and was implemented with up to 2.7B parameters and 400B training tokens. However, as shown in [16], its perfomance lags that of pure Mamba. Based on Hyena, StripedHyena [36] interleaves attention and SSM layers in a 7B parameter model. However, it lags behind the Attention-only Mistral-7B [22]. All of this renders Jamba the first production-grade Attention-SSM hybrid model. Scaling the hybrid Jamba architecture required overcoming several obstacles, which we dicsuss in Section 6.

Jamba also includes MoE layers [13, 41], which allow increasing the model capacity (total number of available parameters) without increasing compute requirements (number of active parameters). MoE is a flexible approach that enables training extremely large models with strong performance [23]. In Jamba, MoE is applied to some of the MLP layers. The more MoE layers, and the more experts in each MoE layer, the larger the total number of model parameters. In contrast, the more experts we use at each forward pass, the larger the number of active parameters as well as the compute requirement. In our implementation of Jamba, we apply MoE at every other layer, with 16 experts and the top-2 experts used at each token (a more detailed discussion of the model architecture is provided below).

We evaluated our implementation of Jamba on a wide range of benchmarks and found it performs comparably to Mixtral-8x7B [23], which has a similar number of parameters, and also to the larger Llama-2 70B [45]. In addition, our model supports a context length of 256K tokens – the longest supported context length for production-grade publicly available models. On long-context evaluations, Jamba outperformes Mixtral on most of the evaluated datasets. At the same time, Jamba is extremely efficient; for example, its throughput is 3x that of Mixtral-8x7B for long contexts. Moreover, our model fits in a single GPU (with 8bit weights) even with contexts of over 128K tokens, which is impossible with similar-size attention-only models such as Mixtral-8x7B.

Somewhat unusually for a new architecture, we release Jamba (12B active parameters, 52B total available parameters) under Apache 2.0 license: `https://huggingface.co/ai21labs/Jamba-v0.1`. We do so since we feel that the novel architecture of Jamba calls for further study, experimentation, and optimization by the community. Our design was based on various ablation experiments we conducted to explore the effect of different tradeoffs and design choices, and insights gleaned from those. These ablations were performed at scales of up to 7B parameters, and training runs of up to 250B tokens. We plan to release model checkpoints from these runs.

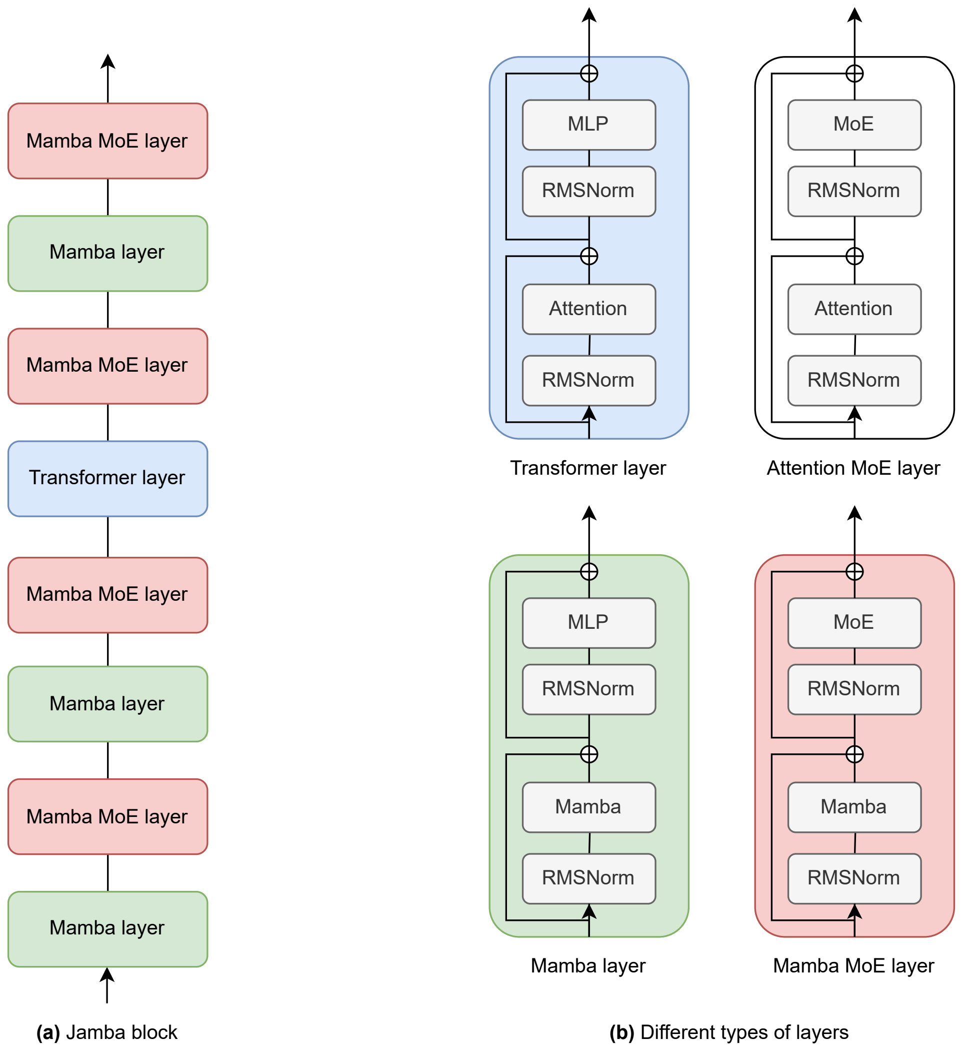


Figure 1: **(a)** A single Jamba block. **(b)** Different types of layers. The implementation shown here is with $l = 8$, $a : m = 1 : 7$ ratio of attention-to-Mamba layers, and MoE applied every $e = 2$ layers.

Important notice*: The Jamba model released is a pretrained* ***base*** *model, which did not go through alignment or instruction tuning, and does not have moderation mechanisms. It should not be used in production environments or with end users without additional adaptation.*

2 Model Architecture

Jamba is a hybrid decoder architecture that mixes Transformer layers [46] with Mamba layers [16], a recent state-space model (SSM) [17, 18], in addition to a mixture-of-experts (MoE) module [13, 41]. We call the combination of these three elements a Jamba block. See Figure 1 for an illustration.

Combining Transformer, Mamba, and MoE elements allows flexibility in balancing among the sometimes conflicting objectives of low memory usage, high throughput, and high quality. In terms of memory usage, note that comparing the total size of the model parameters can be misleading. In an MoE model, the number of active parameters that participate in any given forward step may be much smaller than the total number of parameters. Another important consideration is the KV cache – the memory required to store the attention keys and values in the context. When scaling Transformer models to long contexts, the KV cache becomes a limiting factor. Trading off attention layers for Mamba layers reduces the total size of the KV cache. Our architecture aims to provide not only a small number of active parameters but also an 8x smaller KV cache compared to a vanilla Transformer. Table 1 compares Jamba with recent publicly available models, showing its advantage in maintaining a small KV cache even with 256K token contexts.

	Available params	Active params	KV cache (256K context, 16bit)
LLAMA-2	6.7B	6.7B	128GB
Mistral	7.2B	7.2B	32GB
Mixtral	46.7B	12.9B	32GB
Jamba	52B	12B	4GB

Table 1: Comparison of Jamba and recent open models in terms of total available parameters, active parameters, and KV cache memory on long contexts. Jamba provides a substantial reduction in the KV cache memory requirements.

In terms of throughput, with short sequences, attention operations take up a small fraction of the inference and training FLOPS [6]. However, with long sequences, attention hogs most of the compute. In contrast, Mamba layers are more compute-efficient. Thus, increasing the ratio of Mamba layers improves throughput especially for long sequences.

Here is a description of the main configuration, which provides improved performance and efficiency. Section 6 contains results from ablation experiments supporting the design choices.

The basic component is a Jamba block, which may be repeated in sequence. Each Jamba block is a combination of Mamba or Attention layers. Each such layer contains either an attention or a Mamba module, followed by a multi-layer perceptron (MLP). The different possible types of layers are shown in Figure 1(b).[2] A Jamba block contains l layers, which are mixed at a ratio of $a : m$, meaning a attention layers for every m Mamba layers.

In Jamba, some of the MLPs may be replaced by MoE layers, which helps increase the model capacity while keeping the active number of parameters, and thus the compute, small. The MoE module may be applied to MLPs every e layers. When using MoE, there are n possible experts per layer, with a router choosing the top K experts at each token. In summary, the different degrees of freedom in the Jamba architecture are:

- l: The number of layers.
- $a : m$: ratio of attention-to-Mamba layers.
- e: how often to use MoE instead of a single MLP.
- n: total number of experts per layer.
- K: number of top experts used at each token.

Given this design space, Jamba provides flexibility in preferring certain properties over others. For example, increasing m and decreasing a, that is, increasing the ratio of Mamba layers at the expense of attention layers, reduces the required memory for storing the key-value cache. This reduces the overall memory footprint, which is especially important for processing long sequences. Increasing the ratio of Mamba layers also improves throughput, especially at long sequences. However, decreasing a might lower the model's capabilities.

Additionally, balancing n, K, and e affects the relationship between active parameters and total available parameters. A larger n increases the model capacity at the expense of memory footprint, while a larger K increases the active parameter usage and the compute requirement. In contrast, a larger e decreases the model capacity, while decreasing both compute (when K>1) and memory requirements, and allowing for less communication dependencies (decreasing memory transfers as well as inter-GPU communication during expert-parallel training and inference).

Jamba's implementation of Mamba layers incorporate several normalizations that help stabilize training in large model scales. In particular, we apply RMSNorm [48] in the Mamba layers.

[2]The figure shows a potential Attention MoE layer, which our architecture does not use, but future variants could.

We found that with the Mamba layer, positional embeddings or mechanisms like RoPE [42] are not necessary, and so we do not use any explicit positional information.

Other architecture details are standard, including grouped-query attention (GQA), SwiGLU activation function [6, 40, 45], and load balancing for the MoE [13]. The vocabulary size is 64K. The tokenizer is trained with BPE [15, 29, 39] and each digit is a separate token [6]. We also remove the dummy space used in Llama and Mistral tokenizers for more consistent and reversible tokenization.

3 Reaping the Benefits

3.1 Jamba Implementation for a Single 80GB GPU

The specific configuration in our implementation was chosen to fit in a single 80GB GPU, while achieving best performance in the sense of quality and throughput. In our implementation we have a sequence of 4 Jamba blocks. Each Jamba block has the following configuration:

- $l = 8$: The number of layers.
- $a : m = 1 : 7$: ratio attention-to-Mamba layers.
- $e = 2$: how often to use MoE instead of a single MLP.
- $n = 16$: total number of experts.
- $K = 2$: number of top experts used at each token.

The $a : m = 1 : 7$ ratio was chosen according to preliminary ablations, as shown in Section 6, since this ratio was the most compute-efficient variant amongst the best performing variants in terms of quality.

The configuration of the experts was chosen to enable the model to fit in a single 80GB GPU (with int8 weights), while including sufficient memory for the inputs. In particular, n and e were balanced to have an average of $\sim$8 experts per layer. In addition, we balanced n, K, and e to allow for high quality, while keeping both compute requirements and communication dependencies (memory transfers) checked. Accordingly, we chose to replace the MLP module with MoE on every other layer, as well as have a total of 16 experts, two of which are used at each token. These choices were inspired by prior work on MoE [7, 49] and verified in preliminary experiments.

Figure 2 shows the maximal context length that fits a single 80GB GPU with our Jamba implementation compared to Mixtral 8x7B and Llama-2-70B. Jamba provides 2x the context length of Mixtral and 7x that of Llama-2-70B.

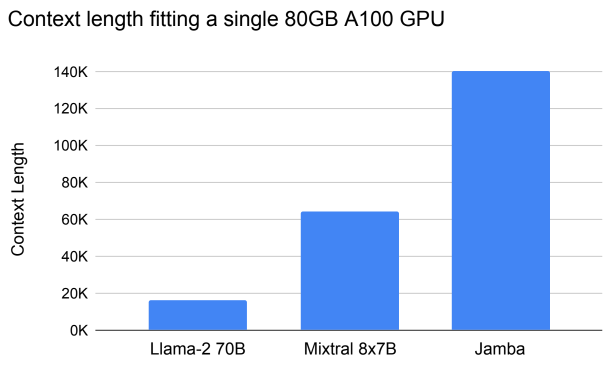


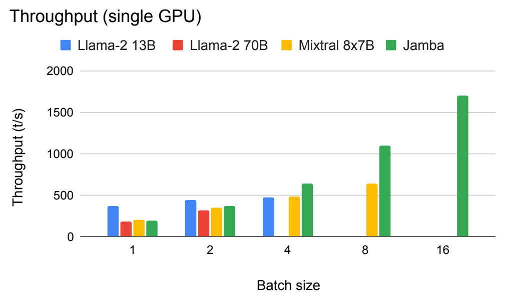
Figure 2: Comparison of maximum context length fitting in a single A100 80GB GPU. Jamba enables 2x the context length of Mixtral and 7x that of Llama-2-70B.

Overall, our Jamba implementation was successfully trained on context lengths of up to 1M tokens. The released model supports lengths of up to 256K tokens.

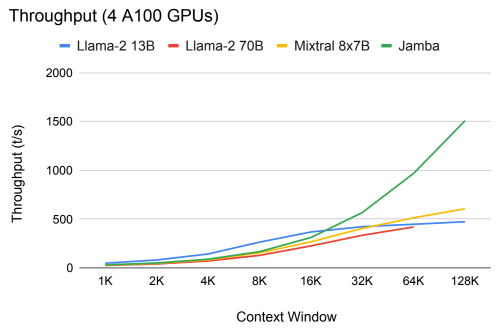
3.2 Throughput Analysis

For concreteness, we present results of the throughput in two specific settings.[3] In the first setting, we have varying batch size, a single A100 80 GB GPU, int8 quantization, 8K context length, generating output of 512 tokens. As Figure 3a shows, Jamba allows processing of large batches, leading to a 3x increase in throughput (tokens/second) over Mixtral, which does not fit with a batch of 16 despite having a similar number of active parameters.

In the second setting, we have a single batch, 4 A100 GPUs, no quantization, varying context lengths, generating output of 512 tokens. As demonstrated in Figure 3b, at small context lengths all models have a similar throughput. Jamba excels at long contexts; with 128K tokens its throughput is 3x that of Mixtral. Note that this is despite the fact that Jamba has not yet enjoyed optimizations of the kind the community has developed for pure Transformer models over the past six years. We can expect the throughput gap to increase as such optimizations are developed also for Jamba.



(a) Throughput at different batch sizes (single A100 GPU, 8K context length). Jamba allows processing large batches, with a throughput 3x greater than Mixtral.



(b) Throughput at different context lengths (single batch, 4 A100 GPUs). With a context of 128K tokens, Jamba obtains 3x the throughput of Mixtral, while Llama-2-70B does not fit with this long context.

Figure 3: Comparison of throughput (tokens/second) with Jamba and recent open models.

4 Training Infrastructure and Dataset

The model was trained on NVIDIA H100 GPUs. We used an in-house proprietary framework allowing efficient large-scale training including FSDP, tensor parallelism, sequence parallelism, and expert parallelism.

Jamba is trained on an in-house dataset that contains text data from the Web, books, and code, with the last update in March 2024. Our data processing pipeline includes quality filters and deduplication.

5 Evaluation

In general we approach benchmarks cautiously, as they correlate only partially with what matters in real applications, and furthermore invite gaming the system in order to boast vanity numbers. Nevertheless, we present several indicative results.

5.1 Academic Benchmarks

We report results with a wide range of standard academic benchmarks:

Common sense reasoning: HellaSwag (10-shot) [47], WinoGrande (5-shot) [37], ARC-E (0-shot) and ARC-Challenge (25-shot) [9], and PIQA (zero-shot) [3].

Reading Comprehension: BoolQ (10-shots) [8] and QuAC (zero-shot) [5].

Others: GSM8K (3-shot CoT) [10], HumanEval (pass@1) [4], Natural Questions closed-book (NQ; 5-shot) [26], and TruthfulQA (zero-shot) [27].

Aggregate benchmarks: MMLU (5-shot) [20] and BBH (3-shot) [43].

[3]Referring to end-to-end throughput (encoding+decoding). The results should be taken relatively rather than absolutely, as they are without possible optimizations.

	Reasoning						
	HellaSwag	**WinoGrande**	**ARC-E**	**ARC-C**	**PIQA**	**NQ**	**TruthfulQA**
Llama-2 13B	80.7	72.8	77.3	59.4	80.5	37.7	37.4
Llama-2 70B	85.3	80.2	80.2	**67.3**	82.8	**46.9**	44.9
Gemma	81.2	72.3	**81.5**	53.2	81.2	32.6	44.8
Mixtral	86.7	81.2	77.6	66	83	44.8	**46.8**
Jamba	**87.1**	**82.5**	73.5	64.4	**83.2**	45.9	46.4

	Comprehension				Aggregate	
	BoolQ	**QuAC**	**GSM8K**	**HumanEval**	**MMLU**	**BBH**
Llama-2 13B	81.7	**42.7**	34.7	18.3	54.8	39.4
Llama-2 70B	85	42.4	55.3	29.9	69.8	51.2
Gemma	87.2	39.2	54.5	32.3	64.3	**55.1**
Mixtral	**88.4**	40.9	**60.4**	**34.8**	**70.6**	50.3
Jamba	88.2	40.9	59.9	29.3	67.4	45.4

Table 2: Comparison of Jamba with other publicly available models. Jamba obtains similar or better performance with much better throughput.

Table 2 compares Jamba to several publicly available models on common academic benchmarks for evaluating language models. We compare with Llama-2 13B [45], which has about the same number of active paramters as our model, Llama-2 70B, which is larger than our model, Gemma [44], which has 7B parameters, and Mixtral [23], which has about the same number of active and total parameters as our model.

Noticebly, Jamba performs comparably to the leading publicly available models of similar or larger size, including Llama-2 70B and Mixtral. At the same time, our model has a smaller number of total available parameters than Llama-2 (52B compared to 70B). Moreover, as a sparse model, Jamba has only 12B active parameters, similar to Mixtral's 12.9B active parameters. However, as a fully-attentional model, Mixtral has a large memory footprint with long sequences, requiring 32GB for the KV cache with 256K tokens. In contrast, thanks to its hybrid Attention-Mamba architecture, Jamba's KV cache takes only 4GB even at such a long context (Section 2). Importantly, our Jamba achieves such a strong performance while having much better throughput than Llama-2 70B and Mixtral, up to 3x improvement (Section 3.2).

In summary, Jamba demostrates the ability of hybrid architectures to reach the performance of state-of-the-art Transformer based models of the same size class, while having the benefits of an SSM.

5.2 Long-Context Evaluations

We have successfully trained Jamba models with context lengths of up to 1M tokens. The released model handles context lengths of up to 256K tokens. In this section, we evaluate it on synthetic and naturalistic benchmarks that test is long-context capabilities.

5.2.1 Needle-in-a-haystack

As Figure 4 shows, Jamba has excellent performance in the needle-in-a-haystack evaluation, which requires retrieving a simple statement planted in a long context window [24]. This result is noteworthy especially given that our implementation of Jamba uses only 4 attention layers.

5.2.2 Naturalistic long-context evaluation

We evaluate Jamba's ability to handle long contexts using question-answering benchmarks, consisting of long inputs. To this end, we repurpose five of the longest-context datasets from L-Eval [2], by structuring them in a few-shot format (we use 3-shots in all experiments here). Specifically, we evaluated the models on the following datasets: NarrativeQA (QA on narratives; [25]), LongFQA (finance; [2]), Natural Questions (NQ; Wikipedia; [26]), CUAD (law; [21]), and SFiction (science

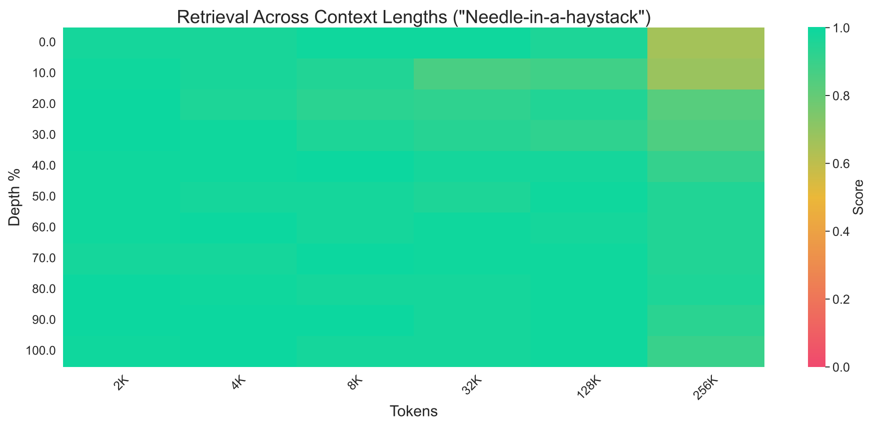


Figure 4: A needle-in-a-haystack evaluation showing Jamba's ability to recall statements placed in the middle of contexts of up to 256K tokens length.

fiction). The average input length in these datasets ranges from 6K to 62K tokens. These context lengths are further highly expanded by the few-shot format.

Table 3 summarizes the evaluation results, in terms of F1. Jamba outperforms Mixtral on most of the datasets as well as on average. In addition, as these long-context tasks require substantial computation, here Jamba's efficiency shines, with much better throughput with long contexts (Section 3.2).

	LongFQA	CUAD	NarrativeQA	NQ	SFiction	Avg
Mixtral	0.42	0.46	0.29	0.58	0.42	0.43
Jamba	0.44	0.44	0.30	0.60	0.40	0.44

Table 3: Results (F1) on long-context QA benchmarks, with a 3-shot format.

6 Ablations and Insights

This section discusses ablation experiments we ran for different design choices in our implementation of the Jamba architecture. First we show the benefit of combining attention and Mamba layers, at which ratio they should be combined, and how to interleave them. We investigate cases where pure Mamba fails, suggesting that it struggles to develop in-context learning capabilities, while the Attention–Mamba hybrid exhibits in-context learning similar to vanilla Transformers. Then we show the benefit of adding MoE on top of a hybrid Attention–Mamba model. Finally, we share two additional learnings that we found useful: explicit positional information is not needed in Jamba, and Mamba layers necessitate special normalization to stabilize training at large scale.[4]

For these ablations, we report the following measures, which exhibit informative performance even at small data or model scale.

- Academic benchmarks: HellaSwag (10-shot) [47], WinoGrande (5-shot) [37], Natural Questions closed-book (NQ; 5-shot) [26].
- HuggingFace OpenLLM leaderboard (OLLM) [11]: a summary statistic of several datasets. We report results with our reproduction.
- Perplexity evaluations: we report log-prob (per byte) on texts from three domains: C4, Books, and code.

[4]In all the ablation experiments, "pure Mamba" refers to models with Mamba layers interleaved with MLP layers.

6.1 Benefits of combining Attention and Mamba

We first investigate the ratio of Attention to Mamba layers ($a : m$), with 1.3B parameters models trained for 250B tokens. As Table 4 shows, the hybrid Jamba model outperforms the pure attention or Mamba models. The ratio of attention-to-Mamba layers may be 1:3 or 1:7 with virtually no performance difference. Figure 5 shows the training loss of these models, where Jamba exhibits improved loss during training. Given that a 1:7 ratio is more compute-efficient and shows similar performance, we opt for it in our larger-scale experiments.

	OLLM	Hella Swag	Wino Grande	NQ	log-prob C4	log-prob Books	log-prob Code
Attention	36.4	62.4	59.6	14.5	-0.543	-0.659	-0.331
Mamba	36.1	62.6	59.4	14.5	-0.543	-0.661	-0.334
Jamba ($a : m = 1 : 3$, no MoE)	37.2	65.1	61.7	16.5	-0.533	-0.649	-0.321
Jamba ($a : m = 1 : 7$, no MoE)	37.2	65.1	61.7	16.0	-0.533	-0.650	-0.321

Table 4: Results on academic benchmarks and log-probability evaluations showing an improved performance of Attention-Mamba (no MoE) compared to vanilla Attention and Mamba models. There is no substantial difference between 1:3 and 1:7 ratios of Attention-to-Mamba layers. Models are 1.3B parameters, trained for 250B tokens.

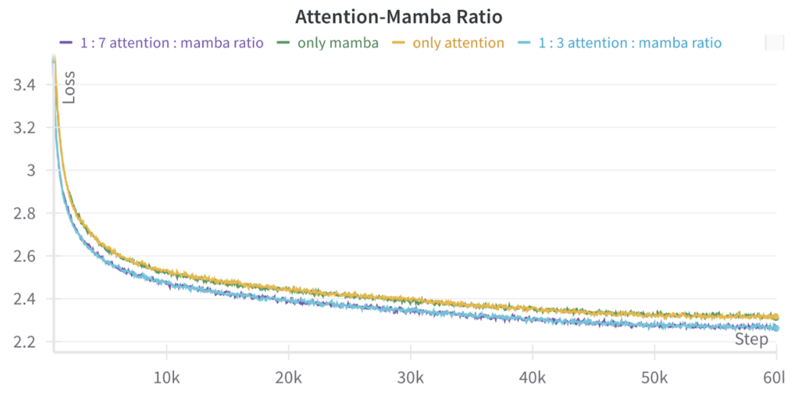


Figure 5: Training loss curves for pure Attention, pure Mamba, and Attention-Mamba hybrids (no MoE), with ratios $a : m$ of 1:3 and 1:4. All models are 1.3B parameters. The two hybrids achieve better loss throughout this training run, without any noticeable difference between the different Attention/Mamba ratios.

Next, we compare performance of vanilla Transformer, vanilla Mamba, and Attention-Mamba hybrid models, at 7B model size, after training on 50B tokens. As Table 5 shows, the pure Mamba layer is quite competitive, but lags slightly behind pure Attention. The hybrid Attention-Mamba (without MoE) outperforms the pure models while obtaining better throughput than the vanilla Transformer (Section 3.2).

	OLLM	Hella Swag	Wino Grande	NQ	log-prob C4	log-prob Books	log-prob Code
Attention	36.1	60.4	59.7	13.7	-0.555	-0.666	-0.347
Mamba	35.3	60.2	55.8	14.0	-0.554	-0.667	-0.355
Jamba ($a : m = 1 : 7$, no MoE)	36.6	62.5	58.8	15.4	-0.547	-0.658	-0.340

Table 5: Results on academic benchmarks and log-prob evaluations, comparing pure Attention, pure Mamba, and Attention-Mamba hybrid (no MoE). Models are 7B parameters, trained for 50B tokens.

Figure 6 shows the training loss of the three architectures. While the pure Transformer and Mamba models have a similar convergence, the hybrid Jamba (no MoE) has a lower loss throughout this run.

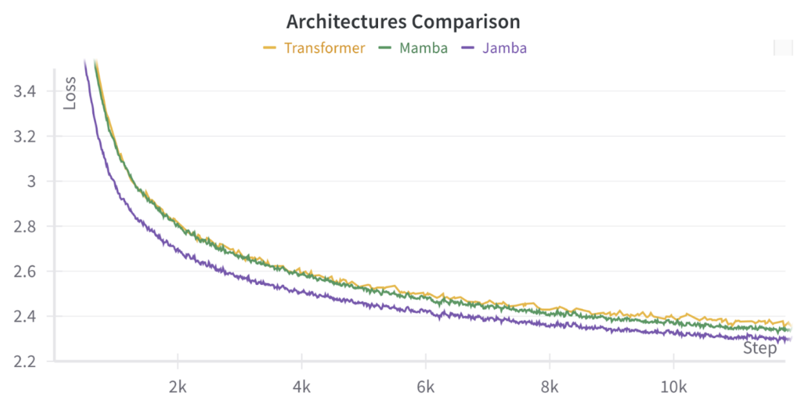


Figure 6: Training loss curves for pure Attention, pure Mamba, and an Attention-Mamba hybrid (no MoE). All models are 7B parameters. the hybrid achives better loss throughout this training run.

6.2 Why does the Combination Work?

The pure Mamba model showed fairly good results in most tasks early on, including in general perplexity evaluations. However, it performed substantially worse than the pure Attention model in three common benchmark tasks: IMDB [28], QuAC [5], and NarrativeQA [25]. In contrast, the hybrid Attention-Mamba performed similarly to the Attention model on these datasets. Table 6 shows the results for 1.3B models after 250B tokens.

	IMDB	QuAC	NarrativeQA
Attention	84.1	27.9	45.8
Mamba	48.8	20.2	27.7
Attention-Mamba	90.9	26.6	43.7

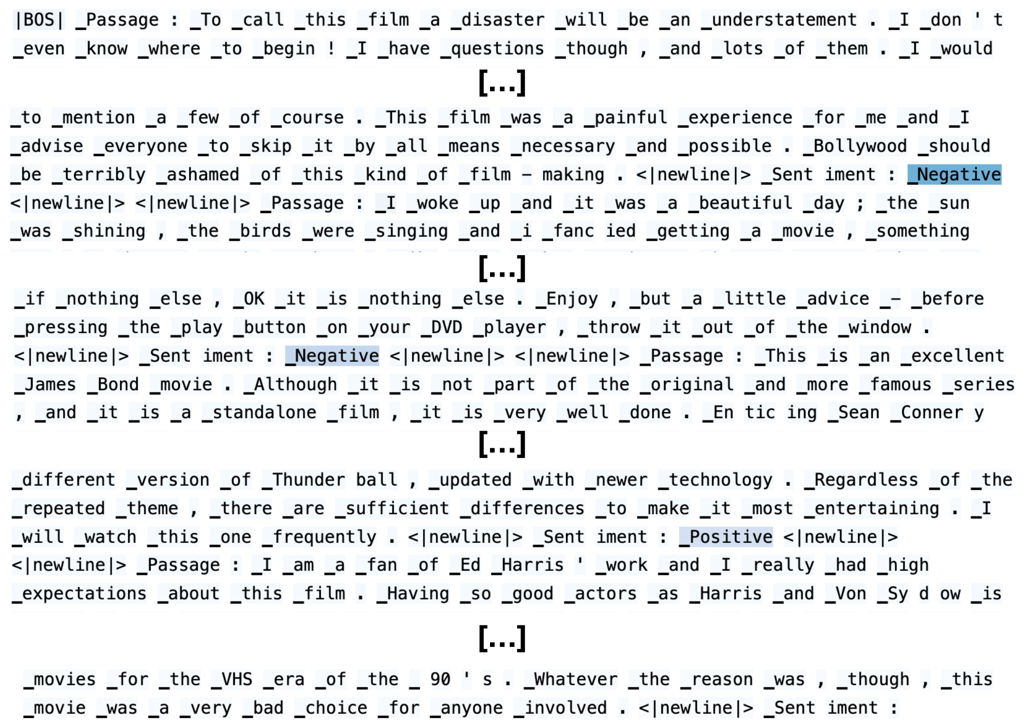
Table 6: Mamba performs poorly on certain datasets, while the Attention-Mamba hybrid performs on par with the Attention model.

Looking into these results further, we found out that the pure Mamba model often does not follow the correct format. For instance, in the IMDB dataset, answer choices are “Positive” or “Negative”. While the Attention model adheres to this format, the pure Mamba model often produces other answers, such as “Very Good”, “Very Positive”, “Funny”, “Bad”, “Poor”, and “3/10”. While these may be considered correct answers, the difficulty of Mamba to adhere to the format suggests a potential problem. Indeed, to perform successful in-context learning, it is important for models to capture the input-output format [30]. The hybrid Attention-Mamba model follows the format successfully, just like the pure Attention model.

We hypothesize that this phenomenon points to a limitation of SSMs – a potential difficulty in in-context learning (ICL). Indeed, the ability to perform ICL has been linked to the emergence of so-called induction heads in Transformer language models during training, which perform approximate copying operations that are supportive of ICL [31]. We conjecture that the lack of an attention mechanism in the pure Mamba model makes it difficult for it to learn in-context. While Mamba may learn to copy and perform simple ICL when explicitly trained to do so ([16, 32], it is not clear if ICL is an emergent capability in SSM as is typical of Transformer models. In contrast, the hybrid Attention–Mamba model does perform successful ICL, even when only 1 out of 8 layers is an Attention one.

As anecdotal evidence of an emergent induction mechanism, we visualize in Figure 7 the attention of an example head from a 1.3B Attention-Mamba hybrid model (no MoE), on an IMDB example where the pure Mamba failed and the hybrid succeeded. Clearly, the attention from the last token

(":") is focused on the labels from the few-shot examples. We have found 12 such heads in our hybrid model, in all three attention layers (which correspond to layers 4, 12, 20 in the model).



```
|BOS| _Passage : _To _call _this _film _a _disaster _will _be _an _understatement . _I _don ' t
_even _know _where _to _begin ! _I _have _questions _though , _and _lots _of _them . _I _would
                                    [...]
_to _mention _a _few _of _course . _This _film _was _a _painful _experience _for _me _and _I
_advise _everyone _to _skip _it _by _all _means _necessary _and _possible . _Bollywood _should
_be _terribly _ashamed _of _this _kind _of _film - making . <|newline|> _Sent iment : _Negative
<|newline|> <|newline|> _Passage : _I _woke _up _and _it _was _a _beautiful _day ; _the _sun
_was _shining , _the _birds _were _singing _and _i _fanc ied _getting _a _movie , _something
                                    [...]
_if _nothing _else , _OK _it _is _nothing _else . _Enjoy , _but _a _little _advice _- _before
_pressing _the _play _button _on _your _DVD _player , _throw _it _out _of _the _window .
<|newline|> _Sent iment : _Negative <|newline|> <|newline|> _Passage : _This _is _an _excellent
_James _Bond _movie . _Although _it _is _not _part _of _the _original _and _more _famous _series
, _and _it _is _a _standalone _film , _it _is _very _well _done . _En tic ing _Sean _Conner y
                                    [...]
_different _version _of _Thunder ball , _updated _with _newer _technology . _Regardless _of _the
_repeated _theme , _there _are _sufficient _differences _to _make _it _most _entertaining . _I
_will _watch _this _one _frequently . <|newline|> _Sent iment : _Positive <|newline|>
<|newline|> _Passage : _I _am _a _fan _of _Ed _Harris ' _work _and _I _really _had _high
_expectations _about _this _film . _Having _so _good _actors _as _Harris _and _Von _Sy d ow _is
                                    [...]
_movies _for _the _VHS _era _of _the _ 90 ' s . _Whatever _the _reason _was , _though , _this
_movie _was _a _very _bad _choice _for _anyone _involved . <|newline|> _Sent iment :
```

Figure 7: Example induction head (H3, first attention layer) from a hybrid Attention-Mamba model. Highlighted words reflect strong attention from the last token, ":", just before the model is about to predict the label. We see that the attention is focused on label tokens from the few-shot examples.

Future work can further investigate the emergence of ICL in hybrid models at large scale. Our released checkpoints would hopefully facilitate such investigations. Finally, very recent work has attempted to extract attention-like scores from state-space models like Mamba [1], which opens another direction to search for induction capabilities in state-space models.

6.3 The Effect of Mixture-of-Experts (MoE)

Recent work has shown that MoE improves Transformer language models while keeping compute manageable [23].[5] However, it is not clear if MoE integrates well with state-space models at a large scale, and specifically with our hybrid Attention–Mamba architecture. Indeed, Table 7 shows that MoE improves the performance of the hybrid Attention-Mamba architecture at large scale (7B parameters trained on 50B tokens). The MoE variant has $n = 16$ total experts, $K = 2$ experts used at each token, and MoE is applied every $e = 2$ layers, as described in Section 3.1.

		Hella	Wino		log-prob		
	OLLM	Swag	Grande	NQ	C4	Books	Code
Jamba (no MoE)	36.6	62.5	58.8	15.4	-0.547	-0.658	-0.340
Jamba+MoE	38.1	66.0	61.2	18.9	-0.534	-0.645	-0.326

Table 7: Mixture-of-experts improves the Attention-Mamba hybrid.

[5]There is also initial evidence that MoE helps Mamba layers, albeit at small model and data scale [34].

6.4 Stabilizing Mamba at large scale

When training Jamba models of up to 1.3B parameters, we observed stable training without special problems. However, when scaling to the largest model released here (7B-based, which has 12B/52B active/total parameters), we encountered large loss spikes. Investigating this revealed that inner parts of the Mamba layers suffer from large activation values, leading to the spikes. We therefore added RMSNorm [48] to internal activations. As Figure 8 shows, this stabilized training and prevented additional loss spikes.

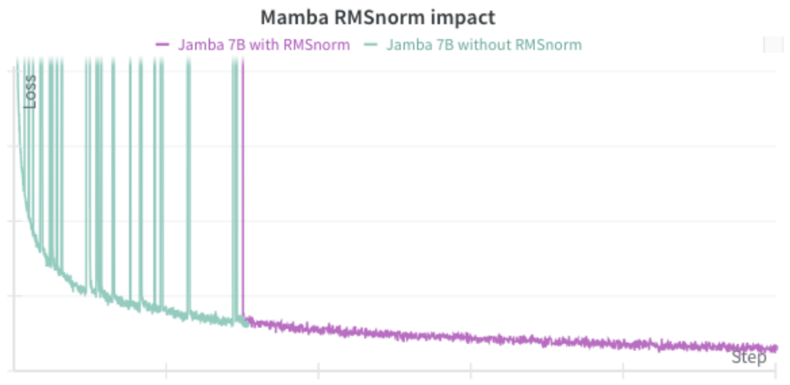


Figure 8: Adding RMSNorm to Mamba layers prevents loss spikes.

6.5 Jamba does not Require Explicit Positional Information

Table 8 shows results of the Jamba architecture (with MoE) with no positional information and when applying RoPE [42] in the attention layers (1.3B parameter models, 250B tokens). The results are similar, suggesting that explicit positional information may not be required for the hybrid architecture. Presumably, the Mamba layers, which are placed before attention layers, provide implicit position information.[6]

								log-prob		
	OLLM	Hella Swag	Wino Grande	ARC-C	Narrative QA	NQ	BoolQ	C4	Books	Code
Jamba	39.6	71.5	64.2	40.7	50.5	22.2	68.9	-0.516	-0.623	-0.299
Jamba+RoPE	40.1	71.8	65.5	40.4	46.2	22.2	67.9	-0.516	-0.623	-0.299

Table 8: Comparison of Jamba with and without explicit positional information.

7 Conclusion

We presented Jamba, a novel architecture which combines Attention and Mamba layers, with MoE modules, and an open implementation of it, reaching state-of-the-art performance and supporting long contexts. We showed how Jamba provides flexibility for balancing performance and memory requirements, while maintaining a high throughput. We experimented with several design choices such as the ratio of Attention-to-Mamba layers and discussed some discoveries made during the development process, which will inform future work on hybrid attention–state-space models. To facilitate such research, we plan to release model checkpoints from smaller-scale training runs. The largest model we provide with this release has 12B active and 52B total available parameters, supporting context lengths of up to 256K tokens and fitting in a single 80GB GPU even when processing 140K-token texts.

[6]Some prior evidence suggested that Transformer decoder models do not need positional encodings [19]. However, all existing large scale models do use some sort of explicit position information.

References

[1] Ameen Ali, Itamar Zimerman, and Lior Wolf. The hidden attention of mamba models. *arXiv preprint arXiv:2403.01590*, 2024.

[2] Chenxin An, Shansan Gong, Ming Zhong, Mukai Li, Jun Zhang, Lingpeng Kong, and Xipeng Qiu. L-Eval: Instituting standardized evaluation for long context language models. *arXiv preprint arXiv:2307.11088*, 2023.

[3] Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. PIQA: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 7432–7439, 2020.

[4] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[5] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer. QuAC: Question answering in context. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2174–2184, 2018.

[6] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.

[7] Aidan Clark, Diego de Las Casas, Aurelia Guy, Arthur Mensch, Michela Paganini, Jordan Hoffmann, Bogdan Damoc, Blake Hechtman, Trevor Cai, Sebastian Borgeaud, et al. Unified scaling laws for routed language models. In *International conference on machine learning*, pages 4057–4086. PMLR, 2022.

[8] Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. BoolQ: Exploring the surprising difficulty of natural yes/no questions. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 2924–2936, 2019.

[9] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

[10] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[11] Hugging Face. Open LLM leaderboard. `https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard`, 2024.

[12] Yassir Fathullah, Chunyang Wu, Yuan Shangguan, Junteng Jia, Wenhan Xiong, Jay Mahadeokar, Chunxi Liu, Yangyang Shi, Ozlem Kalinli, Mike Seltzer, and Mark J. F. Gales. Multi-head state space model for speech recognition. In *Proceedings of INTERSPEECH 2023*, pages 241–245, 2023.

[13] William Fedus, Barret Zoph, and Noam Shazeer. Switch transformers: Scaling to trillion parameter models with simple and efficient sparsity. *Journal of Machine Learning Research*, 23(120):1–39, 2022.

[14] Daniel Y Fu, Tri Dao, Khaled Kamal Saab, Armin W Thomas, Atri Rudra, and Christopher Re. Hungry hungry hippos: Towards language modeling with state space models. In *The Eleventh International Conference on Learning Representations*, 2022.

[15] Philip Gage. A new algorithm for data compression. *The C Users Journal*, 12(2):23–38, 1994.

[16] Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *arXiv preprint arXiv:2312.00752*, 2023.

[17] Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces. In *International Conference on Learning Representations*, 2021.

[18] Albert Gu, Isys Johnson, Karan Goel, Khaled Saab, Tri Dao, Atri Rudra, and Christopher Ré. Combining recurrent, convolutional, and continuous-time models with linear state space layers. *Advances in neural information processing systems*, 34:572–585, 2021.

[19] Adi Haviv, Ori Ram, Ofir Press, Peter Izsak, and Omer Levy. Transformer language models without positional encodings still learn positional information. In *Findings of the Association for Computational Linguistics: EMNLP 2022*, pages 1382–1390, 2022.

[20] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2020.

[21] Dan Hendrycks, Collin Burns, Anya Chen, and Spencer Ball. CUAD: An expert-annotated NLP dataset for legal contract review. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 1)*, 2021.

[22] Albert Q Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, et al. Mistral 7b. *arXiv preprint arXiv:2310.06825*, 2023.

[23] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.

[24] Greg Kamradt. Needle in a haystack - pressure testing llms. `https://github.com/gkamradt/LLMTest_NeedleInAHaystack/`, 2023.

[25] Tomas Kocisky, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gabor Melis, and Edward Grefenstette. The NarrativeQA reading comprehension challenge. *Transactions of the Association for Computational Linguistics*, 6:317–328, 2018.

[26] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: a benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019.

[27] Stephanie Lin, Jacob Hilton, and Owain Evans. TruthfulQA: Measuring how models mimic human falsehoods. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3214–3252, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[28] Andrew Maas, Raymond E Daly, Peter T Pham, Dan Huang, Andrew Y Ng, and Christopher Potts. Learning word vectors for sentiment analysis. In *Proceedings of the 49th annual meeting of the association for computational linguistics: Human language technologies*, pages 142–150, 2011.

[29] Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, et al. Between words and characters: A brief history of open-vocabulary modeling and tokenization in NLP. *arXiv preprint arXiv:2112.10508*, 2021.

[30] Sewon Min, Xinxi Lyu, Ari Holtzman, Mikel Artetxe, Mike Lewis, Hannaneh Hajishirzi, and Luke Zettlemoyer. Rethinking the role of demonstrations: What makes in-context learning work? In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pages 11048–11064, 2022.

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

[32] Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. Can mamba learn how to learn? a comparative study on in-context learning tasks. *arXiv preprint arXiv:2402.04248*, 2024.

[33] Jonathan Pilault, Mahan Fathi, Orhan Firat, Christopher Pal, Pierre-Luc Bacon, and Ross Goroshin. Block-state transformers. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023.

[34] Maciej Pióro, Kamil Ciebiera, Krystian Król, Jan Ludziejewski, and Sebastian Jaszczur. MoE-Mamba: Efficient selective state space models with mixture of experts. *arXiv preprint arXiv:2401.04081*, 2024.

[35] 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. In *International Conference on Machine Learning*, pages 28043–28078. PMLR, 2023.

[36] Michael Poli, Jue Wang, Stefano Massaroli, Jeffrey Quesnelle, Ryan Carlow, Eric Nguyen, and Armin Thomas. StripedHyena: Moving Beyond Transformers with Hybrid Signal Processing Models. https://github.com/togethercomputer/stripedhyena, 2023.

[37] Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. WinoGrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pages 8732–8740, 2020.

[38] George Saon, Ankit Gupta, and Xiaodong Cui. Diagonal state space augmented transformers for speech recognition. In *ICASSP 2023-2023 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 1–5. IEEE, 2023.

[39] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, 2016.

[40] Noam Shazeer. Glu variants improve transformer. *arXiv preprint arXiv:2002.05202*, 2020.

[41] Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. In *International Conference on Learning Representations*, 2017.

[42] Jianlin Su, Murtadha Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024.

[43] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging BIG-Bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023.

[44] Gemma Team, Thomas Mesnard, Cassidy Hardin, Robert Dadashi, Surya Bhupatiraju, Shreya Pathak, Laurent Sifre, Morgane Rivière, Mihir Sanjay Kale, Juliette Love, et al. Gemma: Open models based on gemini research and technology. *arXiv preprint arXiv:2403.08295*, 2024.

[45] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023.

[46] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[47] Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4791–4800, 2019.

[48] Biao Zhang and Rico Sennrich. Root mean square layer normalization. *Advances in Neural Information Processing Systems*, 32, 2019.

[49] Barret Zoph, Irwan Bello, Sameer Kumar, Nan Du, Yanping Huang, Jeff Dean, Noam Shazeer, and William Fedus. ST-MoE: Designing stable and transferable sparse expert models. *arXiv preprint arXiv:2202.08906*, 2022.

[50] Simiao Zuo, Xiaodong Liu, Jian Jiao, Denis Charles, Eren Manavoglu, Tuo Zhao, and Jianfeng Gao. Efficient long sequence modeling via state space augmented transformer. *arXiv preprint arXiv:2212.08136*, 2022.