HYMBA: A HYBRID-HEAD ARCHITECTURE FOR SMALL LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

The transformative capabilities of language models (LMs) have intensified the demand for their deployment on everyday devices, necessitating efficient processing for on-device language tasks. To address this, we propose Hymba, a new family of small language models featuring a hybrid-head architecture that strategically integrates attention mechanisms with state space models (SSMs). This architecture leverages the strengths of both systems: attention heads provide high-resolution recall, akin to snapshot memories in the human brain, while SSM heads offer efficient context summarization, similar to fading memories. To further enhance Hymba's performance, we introduce learnable meta tokens that are prepended to input sequences and jointly trained with model weights during pretraining. These meta tokens act as a learned cache initialization during inference, modulating all subsequent tokens within the hybrid heads and boosting the model's focus on salient information, similar to metamemory. Extensive experiments and ablation studies demonstrate that Hymba sets new state-of-the-art results for small LMs across various benchmarks and advances the accuracy-efficiency trade-offs of small LMs. For instance, Hymba-1.5B achieves comparable commonsense reasoning accuracy to Llama 3.2 3B while being 3.49× faster and offering a 14.72× reduction in cache size. All codes and models will be released upon acceptance.

1 INTRODUCTION

Transformers, with their attention-based architecture, have become the dominant architecture for Language Models (LMs) due to their impressive language modeling capabilities, efficient parallelization, and robust long-term recall enabled by token-level key-value (KV) caches (Vaswani, 2017). However, their quadratic computational cost and substantial memory requirements for storing KV caches pose significant efficiency challenges. In parallel, state space models (SSMs), e.g., Mamba (Gu & Dao, 2023) and Mamba-2 (Dao & Gu, 2024), have emerged as efficient alternatives, offering constant computational and memory complexity during inference and training efficiency with hardware-aware optimizations. Despite their advantages, SSMs often fall short in memory recall tasks compared to their Transformer counterparts, impacting their performance on general benchmarks and recall-intensive tasks (Waleffe et al., 2024; Arora et al., 2024a).

Recent hybrid models that combine attention and SSM layers have shown promising improvements over standalone architectures by sequentially interleaving these layers to capitalize on their respective strengths (Lieber et al., 2024; Ren et al., 2024). However, these existing hybrid models can lead to information bottlenecks when a layer type poorly suited for a specific task cannot effectively process the information, necessitating compensation from subsequent layers.

To address these limitations, we propose Hymba, a novel LM architecture that integrates attention heads and SSM heads within the same layer, offering parallel and complementary processing of the same inputs. This hybrid-head approach allows each layer to simultaneously harness both the high-resolution recall of attention and the efficient context summarization of SSMs, increasing the model's flexibility and expressiveness in handling various types of information flows and memory access patterns. Furthermore, to enhance the achievable performance of Hymba, we introduce meta tokens that are prepended to the input sequences and interact with all subsequent tokens. These meta tokens act as learnable cache initialization, enhancing the capabilities of SSM heads by providing a dynamic initial state that evolves with the model, and mitigating the "softmax attention cannot

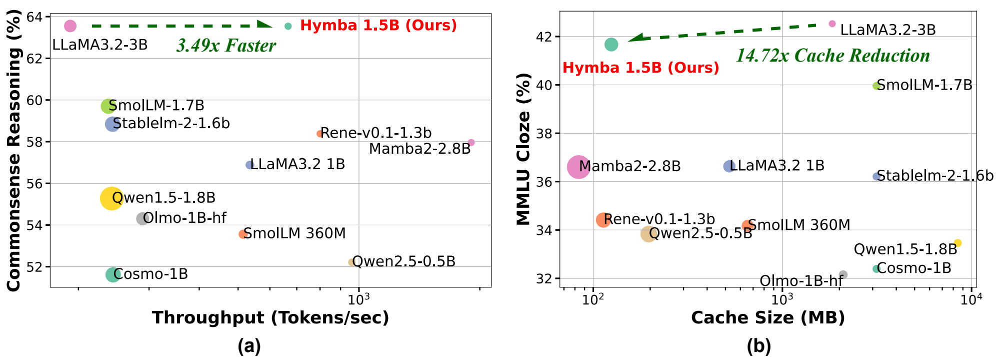

Figure 1: Visualize the trade-off between (a) commonsense reasoning accuracy (averaged over six tasks: ARC-C, ARC-E, PIQA, Hellaswag, OBQA, and Winogrande using (Fourrier et al., 2023)) and throughput, with cache size represented by the point size of different models, and (b) MMLU (cloze) accuracy and cache size, with throughput represented by the point size of different models.

attend to nothing" problem (Bondarenko et al., 2023; Miller; Xiao et al., 2023) for attention heads, improving performance across both general and recall-intensive tasks.

Comprehensive evaluations and ablation studies validate that Hymba not only establishes new state-of-the-art (SOTA) benchmarks across a wide range of representative tasks but also achieves greater efficiency than Transformers, standalone SSMs, or previous hybrid models, as shown in Fig. 1. In commonsense reasoning tasks, for example, Hymba-1.5B matches the performance on commonsense reasoning of Llama-3.2-3B but is $3.49\times$ faster and requires $14.72\times$ smaller cache size.

To effectively leverage Hymba for on-device language tasks, we perform post-training on our base model using supervised finetuning and direct preference optimization (Rafailov et al., 2024) to align the model with downstream tasks. Our instruction-tuned model, Hymba-1.5B-Instruct, has achieved best-in-class performance on GSM8K, GPQA, and the Berkeley function-calling leaderboard, surpassing the previous SOTA sub-2B instruct model Llama-3.2-1B. We also demonstrate that, with parameter-efficient finetuning techniques, our model shows great potential for on-device tasks. For example, with a specialized finetuned version, Dora (Liu et al., 2024c), Hymba-1.5B outperforms Llama3.1-8B-Instruct by 2.4% on RoleBench (Wang et al., 2023).

2 Hymba: The Proposed Hybrid-Head Architecture

In this section, we first outline the design roadmap in Sec. 2.1 and Tab. 1, followed by an in-depth explanation of each component in Sec. 2.2 through Sec. 2.4.

2.1 Hymba: Design Roadmap

SSMs like Mamba (Gu & Dao, 2023) were introduced to address the quadratic complexity and large inference-time KV cache of Transformers. However, due to their low-resolution memory, SSMs struggle with memory recall and performance (Waleffe et al., 2024; Jelassi et al., 2024; Arora et al., 2024a). To overcome these limitations, our roadmap for developing efficient and high-performing small LMs is summarized in Tab. 1 and outlined as follows:

Step ①: Develop fused hybrid modules. We explore different strategies for fusing attention and SSM heads, aiming to combine the recall capabilities of attention with the processing efficiency of SSMs. As shown in Sec. 2.2 and Tab. 1 (B), fusing attention and SSM heads in parallel within a hybrid-head module outperforms sequential stacking (Tab. 1 (A)). This approach allows both types of heads to process the same information simultaneously, leading to improved reasoning and recall accuracy by leveraging the strengths of both components.

Step ②: Reduce compute and KV cache overhead from attention heads. While attention heads improve task performance, they increase KV cache requirements and reduce throughput. To mitigate this, we optimize the hybrid-head module by combining local and global attention and employing cross-layer KV cache sharing, as shown in Tab. 1 (C) and (D). These strategies, detailed in Sec. 2.3, effectively reduce memory costs while maintaining task performance.

Table 1: Design roadmap of our Hymba model. We evaluate the models' (1) commonsense reasoning accuracy, averaged over 8 tasks, which requires the ability to apply everyday knowledge and logic to answer questions, and (2) recall accuracy, averaged over 2 tasks, which corresponds to the ability to retrieve relevant information from past input. The task lists are the same as those in Tab. 3. The throughput is measured with a 8k sequence length and a 128 batch size on an NVIDIA A100 GPU. The cache size is measured with a 16k sequence length, assuming the FP16 format.

Configuration	Commonsense Reasoning (%)	Recall (%)	Throughput (token/sec)	Cache Size (MB)	Design Reason
Ablations on 300M model size and 100B training tokens					
Transformer (LLaMA)	44.08	39.98	721.1	829.4	Accurate recall while inefficient
State Space Models (Mamba)	42.98	19.23	4720.8	1.9	Efficient while inaccurate recall
A. + Attention heads (sequential)	44.07	45.16	776.3	311.9	Enhance recall capabilities
B. + Multi-head structure (parallel)	45.19	49.90	876.7	295.7	Better balance of two modules
C. + Local / global attention	44.56	48.79	2399.7	78.1	Boost compute/cache efficiency
D. + KV cache sharing	45.16	48.04	2756.5	76.3	Cache efficiency
E. + Meta tokens	45.59	51.79	2695.8	76.9	Learned memory initialization
Scaling to 1.5B model size and 1.3T training tokens					
F. + Size / data	60.45	67.61	666.1	124.7	Further boost task performance

Step ③: Further boost task performance via integrating meta tokens. To better modulate the attention mechanism and SSM updates for each token within our hybrid model, we introduce meta tokens—pretrained learnable embeddings that are prepended to input sequences. These tokens serve as a learned cache initialization, enhancing the subsequent tokens' focus on relevant input, thereby boosting recall and reasoning accuracy, as shown in Tab. 1 (E). More details are provided in Sec. 2.4.

Step ④: Scale up data and model sizes. Finally, we scale up the size of both pretraining data (1.3T tokens) and model parameters (1.5B) and deliver our Hymba model family (see Tab. 1 (F)) in Sec. 2.5, which establishes new SOTA performance for small LMs across benchmarks, as extensively demonstrated in Sec. 3.2.

2.2 Hymba-Step ①: Develop A Fused Hybrid-Head Module

Target problem of this step. SSM models are efficient but suffer from limited recall capabilities and task performance (Waleffe et al., 2024; Jelassi et al., 2024; Arora et al., 2024a; Ben-Kish et al., 2024). Given the high recall resolution of attention, in this step we aim to (1) combine the processing efficiency and context summarization capabilities of SSMs with the high recall resolution of attention, and (2) develop a fused building block to achieve this goal, so it can serve as a fundamental component for constructing future foundation models.

2.2.1 The Proposed Hybrid-Head Module

Previous hybrid models (Ren et al., 2024; Glorioso et al., 2024; Lieber et al., 2024) often combine attention and SSMs in a sequential manner. This strategy may lead to information bottlenecks when a layer type that is poorly suited for a specific task cannot effectively process the information. Motivated by the multi-head attention structure in the vanilla Transformer (Vaswani, 2017), where different heads undertake different roles and focus on different contexts (Lv et al., 2024; Merullo et al., 2024), we propose an alternative approach: *fusing attention and SSMs in parallel into a hybrid-head module*, as shown in Fig. 2 (a). The advantage of this design is that different attention and SSM heads can store, retrieve, and process the same piece of information in distinct ways, thereby inheriting the strengths of both operators.

Design formulation. We show that the hybrid-head module can be represented by a unified and symmetric formulation. As shown in Fig. 2 (a), given the input sequence $\tilde{X}$, which is the original input sequence X prepended with meta tokens introduced in Sec. 2.4, the input projection $W_{\text{in\_proj}} = [W^Q, W^K, W^V, W^{SSM}, W^G]$ projects $\tilde{X}$ to the query, key, and value of the attention heads using W^Q, W^K, and W^V, respectively, as well as the input features and gates of the SSM heads using W^{SSM} and W^G, respectively.

Following (Vaswani, 2017), the output of attention heads Y_{attn} can be formulated as:

$$Y_{\text{attn}} = \text{softmax}(QK^T)\, W^V \tilde{X} = M_{\text{attn}} \tilde{X} \tag{1}$$

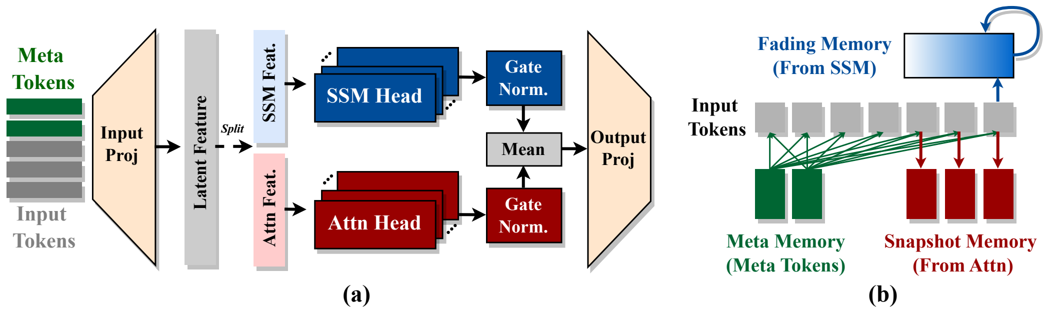

Figure 2: (a) Visualize the hybrid-head module in Hymba; (b) Interpret from the memory aspect.

where $M_{\text{attn}} = \text{softmax}(QK^T)\,W^V$ and $Q = W^Q\tilde{X}$, $K = W^K\tilde{X}$. Note that in this formulation we omit the scaling factor in the attention mechanism for simplicity of illustration.

Similar to the attention heads, the SSM heads in our model, for which we adopt Mamba (Gu & Dao, 2023), can also be represented using a data-controlled linear operator M_{ssm}, following (Ali et al., 2024; Ben-Kish et al., 2024). Specifically, the SSM head output Y_{ssm} can be formulated as:

$$\alpha^{i,j} = C_i \left(\prod_{k=j+1}^{i} \exp(A\Delta_k) \right) B_j \Delta_j,$$
$$Y_{\text{ssm}} = G \odot \alpha(A,B,C,\Delta)\; W^{SSM}\tilde{X} = M_{\text{ssm}}\tilde{X}, \tag{2}$$

where $M_{\text{ssm}} = G \odot \alpha(A,B,C,\Delta)\; W^{SSM}$, $G = W^G\tilde{X}$ is an output gate, and A, B, C, Δ are the SSM parameters following the definition in (Gu & Dao, 2023). More specifically, A is a learnable matrix, $B = W_B X_{ssm}$, $C = W_C X_{ssm}$, and $\Delta = \text{Softplus}(W_\Delta X_{ssm})$ with $X_{ssm} = W^{SSM}\tilde{X}$.

We observed that the output magnitudes of the SSM heads, Y_{ssm}, are consistently larger than those of the attention heads, Y_{attn}, as visualized in Fig. 7 in Append. B. To ensure effective fusion, we normalize and re-scale them using learnable vectors to improve training stability, and then average the outputs, followed by a final output projection. The overall formulation of our fused module can be represented symmetrically:

$$Y = W_{\text{out\_proj}} \left(\beta_1 \text{norm}(M_{\text{attn}}\tilde{X}) + \beta_2 \text{norm}(M_{\text{ssm}}\tilde{X}) \right) \tag{3}$$

where β_1 and β_2 are learnable vectors that re-scale each channel of the outputs from the attention and SSM heads, respectively. We further explore the optimal ratio of SSMs and attention in hybrid heads, along with their fusion strategy, in Append. B.

Note that the key design principle of our hybrid-head module is to process the same piece of information in parallel using hybrid operators, thereby benefiting from the complementary roles of both operators. In this work, we adopt Mamba Gu & Dao (2023) as SSM heads, while more advanced SSMs or linear attention Sun et al. (2023); Yang et al. (2023); Qin et al. (2024); Yang et al. (2024) can be integrated into our hybrid-head structure to seek further performance improvements.

Interpretation from the memory aspect. The components in the hybrid-head module can be interpreted as analogous to human brain functions. Specifically, as shown in Fig. 2 (b), the attention heads provide high recall resolution and thus act like snapshot memories in the human brain, storing detailed recollections of a moment or event. In contrast, the SSM heads summarize the context through a constant cache and thus function as fading memories, which gradually forget the details of past events while retaining their core or gist. As shown in Tab. 9 in Append. B, in our Hymba, the summarized global context from fading memories enables allocating more snapshot memories for memorizing local information while maintaining recall capabilities. This is achieved by replacing most global attention with local attention, thus improving memory efficiency.

2.2.2 Hybrid Module Design: Parallel vs. Sequential

We compare the hybrid-head module with a sequential counterpart, which interleaves local attention and Mamba layers as adopted by (Ren et al., 2024), by calculating the models' effective receptive

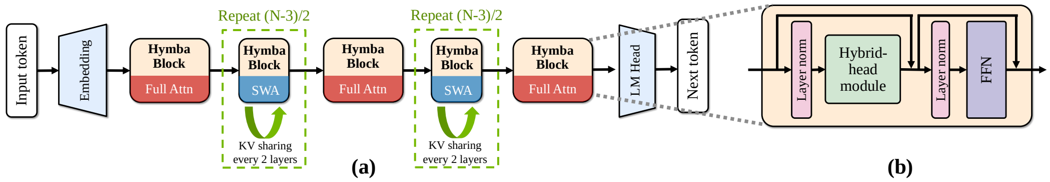

Figure 4: (a) The overall architecture of our Hymba model; (b) The building block of Hymba.

field (ERF) and their overall cache size. All the compared models have the same parameter size and are training from scratch using exactly the same training recipe. ERF is an empirical measure of the averaged distance among tokens that allows effective information propagation (Ben-Kish et al., 2024; Dosovitskiy, 2020) defined as the following,

$$ERF \approx \sum_{n \leq N} \sum_{h \leq H} \sum_{s \leq S} \frac{2M^h(S,s) \cdot (S-s) \cdot (N-n+1)}{HN(N+1)},$$

where S is index of the last token in the sequence, N is index of the last layer in the model, and $M^h(S,s)$ is the normalized attention score between token s and the last token in head h.

As shown in Fig. 3, we observe that (1) in line with common intuitions, Llama3 exhibits a notably larger ERF compared to Mamba due to its higher recall resolution, albeit at the cost of a larger cache size; (2) our multi-head structure demonstrates the best ERF across the four designs, with an order of magnitude larger ERF while maintaining a cache size comparable to the sequential structure. This suggests that the parallel structure can better leverage the limited cache size to capture longer and more complex relationships among tokens compared to the sequential one. The differences in ERF are also reflected in task accuracy: According to Tab. 1, the multi-head design (Tab. 1 (B)) improves commonsense reasoning and recall accuracy by +1.08% and 4.74%, respectively, over the sequential design (Tab. 1 (A)). Based on this benchmarking and analysis, we adopt the hybrid-head module as our basic building block.

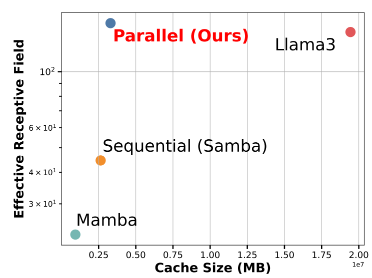

Figure 3: Visualize the ERF and cache size trade-off.

2.3 HYMBA-STEP ②: FURTHER KV CACHE OPTIMIZATION

Target problem of this step. Our hybrid-head module improves recall and reasoning capabilities but reduces memory and throughput efficiency due to the KV cache of the attention heads. To address this, we aim to reduce the KV cache while maintaining comparable task performance.

Combine global and local attention. Local attention, also known as Sliding Window Attention (SWA) (Beltagy et al., 2020), offers a more efficient alternative to global full attention, though it risks losing global context. However, with the presence of SSM heads in our hybrid-head module, which already summarize global context, we can more aggressively replace global full attention with local attention, achieving a better balance between efficiency and performance.

Exploring the ratio of local attention and global attention. As shown in Tab. 9 in Append. B, we initially replace global attention in all layers with SWA, which results in a significant degradation in recall capabilities, with accuracy dropping by over 20% on recall-intensive tasks. In response, we progressively reinstate global attention in some layers. Interestingly, as shown in Tab. 1 (C), we find that using global attention in just three layers (i.e., the first, middle, and last layers) is sufficient to recover recall-intensive accuracy while maintaining comparable commonsense reasoning accuracy. In turn, this strategy achieves 2.74× throughput and 3.79× cache reduction. This configuration is employed in all our delivered models.

Interpretation from the memory aspect. From the memory perspective, after introducing local attention, our hybrid-head module utilizes three types of memory systems that complement each other with varying costs and access patterns: (1) a limited number of expensive global memories from the full attention heads, (2) lower-cost local memories from the SWA heads, and (3) cheap but fading recurrent memories from the SSM heads.

Cross-layer KV sharing. Recent works (Liu et al., 2024a) observe that KV cache shares a high similarity between adjacent layers, suggesting that using separate KV caches for each layer leads to both cache and parameter redundancy. In light this, we employ cross-layer KV sharing (Brandon et al., 2024), where keys and values are shared between consecutive layers (e.g., every two layers share the same KV cache). This strategy reduces both KV memory usage and model parameters, allowing the saved parameters to be reallocated to other model components. As shown in Tab. 1 (D), cross-layer KV sharing improves throughput by $1.15\times$ while maintaining comparable recall accuracy and boosting commonsense accuracy by +0.60%.

After the above optimization, Hymba's overall architecture is visualized in Fig. 4.

2.4 HYMBA-STEP ③: INTEGRATION OF META TOKENS

Target problem of this step. We observed that the initial tokens, though not semantically important, often receive significant attention scores from subsequent token, similar to observations by prior work (Xiao et al., 2023; Han et al., 2024). We hypothesize that drawing excessive attention to these semantically unimportant tokens does not benefit attention mechanisms. Therefore, in this step, we aim to guide the attention to focus more on tokens that meaningfully contribute to task performance.

Introducing meta tokens. Our key insight is that, rather than drawing excessive attention to semantically unimportant initial tokens, learning these critical tokens jointly with model weights could better shape the attention distribution. Specifically, we introduce a set of learnable meta tokens $R = [r_1, r_2, \ldots, r_m]$ to serve as the initial tokens. Given the input sequence $X = [x_1, x_2, \ldots, x_n]$, these meta tokens are prepended to the input sequence, forming the modified input sequence:

$$\tilde{X} = [R, X] = [r_1, r_2, \ldots, r_m, x_1, x_2, \ldots, x_n] \tag{4}$$

where $\tilde{X}$ represents the new input sequence for our model. At inference time, since the meta tokens are fixed and appear at the beginning of any input sequences, their computation can be performed offline. Thus, the role of meta tokens at inference can also be viewed as *learned cache initialization* to modulate the subsequent tokens, allowing subsequent tokens to focus more on those that contribute meaningfully to task performance.

Interpretation from the memory aspect. Similar to the analogy in Sec. 2.2, the meta tokens participate in the attention and SSM calculations of all subsequent tokens, analogous to metamemory in the human brain, which helps recognize where to locate needed information in other memories. We further analyze others roles of meta tokens and their connections with related works in Append. D.

Meta tokens boost recall capabilities and commonsense reasoning accuracy. To analyze the impact of meta tokens on the attention mechanism, we visualize the entropy of the attention map for both the attention and SSM heads (Ali et al., 2024; Ben-Kish et al., 2024) before and after introducing meta tokens. Specifically, the attention map entropy reflects the distribution of attention scores across tokens, where lower entropy indicates stronger retrieval effects (Ren et al., 2024), as the attention scores are concentrated around a smaller subset of tokens, and vice versa.

We provide the visualization in Fig. 9 in Append. D, where we observe that, after introducing meta tokens, both the attention and SSM heads exhibit an overall reduction in entropy. Combined with the improved reasoning and recall capabilities shown in Tab. 1 (E), this suggests that meta tokens may help both the attention and SSM heads focus more on a subset of important tokens that contribute most to task performance.

2.5 HYMBA-STEP ④: DELIVER THE MODEL FAMILY WITH SCALED MODEL AND DATA SIZE

Building on the design insights explored above, we scale up the model sizes and training tokens to deliver the Hymba model family, which includes a 125M model, a 350M model, and a 1.5B model.

We train Hymba-125M/350M/1.5B models using a mix of DCLM-Baseline-1.0 (Li et al., 2024), SmoLM-Corpus (Ben Allal et al., 2024), and a proprietary high-quality dataset, with 1T, 250B, and 50B tokens, respectively. We combine the Warmup-Stable-Decay (WSD) learning rate scheduler (Hu et al., 2024), with maximum and minimum learning rates of 3e-3 and 1e-5, and the data annealing technique (Dubey et al., 2024; Shen et al., 2024) to ensure stable pretraining. Throughout the training process, we use a sequence length of 2k and a batch size of 2M tokens. More pretraining details are provided in Append. F.

Table 2: Benchmark Hymba with SOTA small LMs. All models have less than 2B parameters. MMLU close and all other results are obtained through HUGGINGFACE/LIGHTEVAL (Allal et al., 2024) and LM-EVALUATION-HARNESS (Gao et al., 2023), respectively. SQuAD-C (SQuAD-Completion) indicates a variant of the SQuAD question answering task proposed by Arora et al. (2024b). The throughput is measured with a 8k sequence length and a 128 batch size on an NVIDIA A100 GPU. For models encountering out-of-memory (OOM) issues during throughput measurement, we halve the batch size until the OOM is resolved. This approach is used to measure the maximum achievable throughput for efficient batch generation without OOM.

Model	#Params.	Throughput	MMLU cloze / 5-shot	ARC-E	ARC-C 25-shot	PIQA	WinoGrnde	HellaSwag	SQuAD-C 1-shot	Avg.
OpenELM-1	1.1B	-	- / 27.06	62.37	33.87	74.76	61.80	48.37	-	51.37
Llama-3.2-1B	1.2B	534.99	36.63 / 32.12	65.49	32.8	74.48	60.69	47.72	49.20	49.89
Rene-v0.1	1.3B	800.15	34.41 / 32.94	67.05	36.95	76.49	62.75	51.16	48.36	51.26
Phi-1.5	1.3B	241.03	33.55 / 42.56	76.18	49.40	76.56	**72.85**	48.00	30.09	53.65
RWKV6	1.6B	1498.91	32.10 / 25.92	60.69	34.13	73.61	60.62	46.45	45.01	47.32
SmolLM	1.7B	237.67	39.96 / 27.06	**76.47**	46.67	75.79	60.93	49.58	45.81	52.78
Cosmo	1.8B	244.21	32.39 / 26.10	62.42	34.81	71.76	55.80	42.90	38.51	45.59
h2o-danube2	1.8B	271.27	34.57 / 40.05	70.66	40.61	76.01	66.93	53.70	49.03	53.95
Llama-3.2-3B	3.0B	190.94	**42.53 / 56.03**	74.54	46.50	76.66	69.85	**55.29**	43.46	58.11
Hymba	1.5B	666.13	41.53 / 52.78	76.18	**51.96**	**77.53**	65.59	53.95	**55.46**	**59.37**

3 EXTENSIVE EVALUATION OF OUR HYMBA MODEL FAMILY

3.1 EXPERIMENT SETTINGS

Baselines. Our baselines include popular (small) LMs with quadratic attention (e.g., Llama 3.2 (AI, 2024c), SmolLM (Allal et al., 2024), StableLM (Bellagente et al., 2024), Olmo (Groeneveld et al., 2024), and Cosmo (Huggingface, 2024)), linear recurrence models (e.g., Mamba2 (Dao & Gu, 2024)), and hybrid models (e.g., Rene (AI, 2024a)).

Benchmark settings. We adopt two benchmark settings: (1) In Sec. 3.2, we directly benchmark our delivered Hymba against SOTA public small LMs, and (2) in Sec. 3.3, we train different architectures from scratch with the same dataset, number of layers, model size, and training recipes.

Benchmark tasks. In addition to evaluating language modeling, commonsense reasoning, and recall-intensive tasks on our base models, we also evaluate our instruction-tuned models on downstream tasks such as math, function calling, and role-playing in Sec. 3.4.

3.2 BENCHMARK WITH SOTA SMALL LMS

We present the benchmark results of our Hymba models with parameter sizes of 125M, 350M, and 1.5B, compared to SOTA small language models within the same size range. In addition to the commonly adopted 5-shot MMLU, we further follow the evaluation setup in (Allal et al., 2024; Cosmopedia/evaluation) to report MMLU cloze. Specifically, the default leaderboard MMLU task uses "A", "B", "C", "D", etc., as answer targets, which generally yields random results on small and non-instructed models. In contrast, MMLU cloze uses the full MMLU answer as the target, resulting in more stable and less biased outcomes (Alzahrani et al., 2024).

As highlighted in Tab. 2, Hymba-1.5B models perform the best on seven out of eight tasks using only 1.3T pretraining tokens. At the same time, Hymba-1.5B maintains high throughput, being about 1.2$\times$ to 2.8$\times$ faster than other Transformer-based models at an 8K sequence length. This speedup becomes even more pronounced as the sequence length increases.

Our delivered tiny LMs, Hymba-125M/350M, consistently outperform all LMs of comparable model size, as summarized in Tab. 6 and Tab. 7 in Append. A.

3.3 BENCHMARK DIFFERENT ARCHITECTURES UNDER THE SAME SETTING

General and recall-intensive tasks performance comparison. We do a comprehensive comparison between Hymba and other model architectures, including standard Transformer (Llama3 (AI,

Table 3: Apple-to-apple comparison of our Hymba, pure Mamba2 (Dao & Gu, 2024), Mamba2 with FFN, Llama3 (Dubey et al., 2024) style, and Samba- (Ren et al., 2024) style (Mamba-FFN-Attn-FFN) architectures. All models have 1B parameters and are trained from scratch for 100B tokens from SmolLM-Corpus (Ben Allal et al., 2024) with exactly the same training recipe. All results are obtained through LM-EVALUATION-HARNESS (Gao et al., 2023). The best and second best results are highlighted in bold and underline, respectively.

Task Type	Arch. Style (1B)	Mamba2	Mamba2 w/ FFN	Llama3	Samba	Hymba
Language	Wiki. ppl. ↓	<u>19.17</u>	20.42	19.28	19.91	**18.62**
	LMB. ppl. ↓	<u>12.59</u>	14.43	13.09	12.65	**10.38**
Recall Intensive	SWDE ↑	50.24	26.43	**75.95**	30.00	<u>60.71</u>
	SQuAD-C ↑	36.43	31.40	18.70	<u>42.33</u>	**44.93**
	Avg. ↑	43.34	28.92	<u>47.33</u>	36.17	**52.82**
Common-sense Reasoning and Question-answering	Lambda ↑	47.51	44.54	47.95	<u>49.08</u>	**52.84**
	PIQA ↑	<u>73.94</u>	73.07	73.45	73.23	**74.97**
	ARC-C ↑	38.91	37.03	<u>39.68</u>	39.59	**41.72**
	ARC-E ↑	70.96	71.00	<u>73.74</u>	73.36	**74.12**
	Hella. ↑	57.73	55.83	57.64	<u>58.49</u>	**60.05**
	Wino. ↑	**58.48**	55.56	56.20	57.54	<u>57.85</u>
	TruthfulQA ↑	30.75	29.86	<u>31.64</u>	28.84	**31.76**
	SIQA ↑	41.86	42.22	42.22	<u>42.48</u>	**43.24**
	Avg. ↑	52.52	51.14	52.82	<u>52.83</u>	**54.57**

2024b)), pure Mamba (Gu & Dao, 2023; Dao & Gu, 2024), Mamba with FFN and hybrid architecture with sequential layer stacking (Samba (Ren et al., 2024)) on several downstream tasks. All models have the same number of layers and total parameters to facilitate equal comparison. Models are trained on the same data with the same hyperparameters under the same codebase. To ensure the results are generalizable, we run comparison experiments at different scales (1B and 300M) and different training datasets (SmolLM-corpus (Ben Allal et al., 2024) and FineWeb (Penedo et al., 2024)) in Tab. 3 and Tab. 8, respectively. We evaluate the models on language modeling, real-world recall-intensive, and general common-sense reasoning, and question-answering tasks.

As shown in Tab. 3, our Hymba model consistently outperforms other 1B architectures across most tasks, e.g., achieving an average score 1.45% higher than the second-best model at the 300M scale and 1.74% higher at the 1B scale. The ablation study for the 300M scale is in Append. A.

In addition, considering that Mamba models suffer from limited recall capabilities due to their constant-size cache and recurrent nature (Ben-Kish et al., 2024; Arora et al., 2024a; Jelassi et al., 2024), we test the models on two real-world recall-intensive tasks, SWDE (Arora et al., 2024a; Lockard et al., 2019) and SQuAD (Arora et al., 2024a; Rajpurkar et al., 2018), where the former is to to extract semi-structured relations from given raw HTML websites and the latter is to extract answers from a given context passages. Echoing the previous findings, Mamba2 and Mamba2 with FFN architectures under-perform the Transformer model (i.e., Llama3) on these tasks (see Tab. 3). Our Hymba model augments the Mamba heads with attention heads, which allows the model to have a large ERF to establish long-range dependencies and high-resolution memory to store and retrieve key information in all layers. As a result, our Hymba model outperforms the Transformer model and Samba architecture (that stacks Mamba and attention layers sequentially).

Needle-in-the-Haystack performance comparison. We further do an apple-to-apple comparison between Hymba, Mamba2 and Llama3 on the synthetic retrieval task, needle-in-the-haystack. A random and informative sentence (i.e., needle) is inserted into a long document (i.e., haystack) and the model is required to retrieve the needle from the haystack to answer the questions. All models are of size 1B and trained with the same setting: i. pretrain is done with 1k sequence length; ii. finetune with 4k sequence length; iii. test with up to 16k sequence length. If models have ROPE, we adjust the ROPE base on (Liu et al., 2023) during finetuning. As shown in Fig. 5, Hymba model significantly outperforms the Mamba2 and Llama3 models. While the Mamba2 model has

Table 4: The comparison between lightweight instruction-tuned models. The best and second-best results are highlighted in bold and underlined, respectively. * OpenELM and SmolLM cannot understand function calling, leading to 0 accuracy in most categories. We also included the results of the Llama3.2-3B model as a reference, but since it is larger than 3B, it has been marked in gray.

Model	#Params	MMLU ↑	IFEval ↑	GSM8K ↑	GPQA ↑	BFCLv2 ↑	Avg. ↑
SmolLM	1.7B	27.80	25.16	1.36	25.67	-*	20.00
OpenELM	1.1B	25.65	6.25	56.03	21.62	-*	27.39
Llama-3.2	1.2B	44.41	**58.92**	42.99	24.11	20.27	38.14
Gemma-2	2.6B	**56.87**	28.47	52.16	25.89	12.49	35.18
Llama-3.2-3B	3.2B	61.22	77.40	77.26	29.69	45.65	58.24
Hymba	1.5B	53.36	**64.61**	**57.62**	**27.90**	**48.07**	**50.31**

good extrapolation capabilities when the needle is inserted in the end of the haystack, it struggles to retrieve the needle when the needle is in the beginning or middle of the haystack. In contrast, Llama3 model has limited extrapolation capabilities (Peng et al., 2023b; Liu et al., 2023; Zhang et al., 2024) and struggles to the "lost in the middle" (Liu et al., 2024b) scenario.

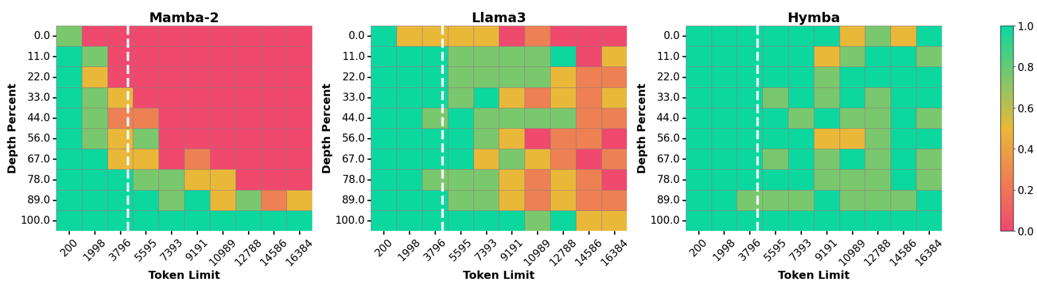

Figure 5: Needle-in-the-haystack performance comparison across different architecture under apple-to-apple setting. The white vertical line represents the finetuning sequence length (4K).

3.4 Evaluation on Instruction-tuned Benchmarks

Implementation details of post-training. We post-trained Hymba-1.5B base model with a two-stage strategy: the first full-finetuning (FFT) stage and another direct preference optimization (DPO) (Rafailov et al., 2024) training. The learning rates are 5e-5, and 3e-6 for FFT and DPO, respectively. To accelerate training, we follow the training recipe (Tunstall et al., 2023; Diao et al., 2024; Dong et al., 2024) to pack the samples and use a block size of 2048. We compare Hymba-1.5B-Instruct with competitive lightweight instruction-tuned models, i.e., Llama-3.2-1B-Instruct (AI, 2024c), Gemma-2-2B-Instruct (Team et al., 2024), OpenELM-1-1B-Instruct (Mehta et al., 2024), and SmolLM-1.7B-Instruct (Allal et al., 2024). We test the instruction-tuned models on MMLU (5-shot), IFEval, GSM8K (5-shot), GPQA (0-shot), and Berkeley Function-Calling Leaderboard v2 (BFCLv2) (Yan et al., 2024). More details about the experimental settings, baseline models, and evaluation tasks are shown in Append. F.

Evaluation results. The evaluation results are shown in Tab. 4. In general, Hymba-1.5B-Instruct achieves the highest performance on an average of all tasks, outperforming the previous SoTA model, Llama-3.2-1B-Instruct, by around 12%. It demonstrates a great ability on math, reasoning, and function calling, with the best-in-class performance.

Evaluation on role-play tasks. In addition to full-finetuning, we con-

Model	#Params	Instruction Generalization	Role Generalization
Llama-7B	7B	19.2	19.3
Aplaca-7B	7B	25.6	24.5
Vicuna-13B	13B	25.0	24.3
Llama2-7B-chat	7B	18.8	20.5
RoleLlama-7B	7B	35.5	33.5
Hymba-DoRA	1.5B	**40.0**	**37.9**

Table 5: The comparison between DoRA-finetuned Hymba and baselines on RoleBench. All baseline results are from Wang et al. (2023).

duct experiments to evaluate whether Hymba is compatible with DoRA (Liu et al., 2024c), a parameter-efficient finetuning method that updates pretrained models using a minimal set of parameters. This approach is especially well-suited for on-device finetuning scenarios where computational resources are constrained. Additionally, DoRA significantly reduces storage requirements for saving multiple downstream models, as it only requires storing the finetuned DoRA parameters, which constitute less than 10% of the original model's total parameters. Specifically, we further finetune the post-trained Hymba on RoleBench (Wang et al., 2023) using DoRA to enhance its role-playing capabilities. The training set of RoleBench is used for training, and the model is evaluated on two sub-tasks: instruction generalization (Inst. Gene.) and role generalization (Role. Gene.). As shown in the Tab. 5, our Hymba-DoRA significantly outperforms larger models. For instance, DoRA finetuned Hymba achieves scores of 40.0% / 37.9% on instruction generalization/role generalization, outperforming RoleLlama-7B (Wang et al., 2023) by 4.5%, and 4.4% respectively. This indicates the strong generalization of our model and the effectiveness of using parameter-efficient finetuning techniques to further enhance its performance.

4 Related Works

Large language models. Prior to the rise of LLMs, transformer-based models (Vaswani, 2017; Devlin et al., 2018; Raffel et al., 2020; Roberts et al., 2022) proved highly effective at capturing relationships between tokens in complex sequences through the use of the attention mechanism (Vaswani, 2017). These models also demonstrated considerable scalability (Qin et al., 2023; Kaplan et al., 2020; Biderman et al., 2023) in terms of both model size and the volume of pretraining data. This scalability paved the way for the development of LLMs, such Mistral (Jiang et al., 2023), the Llama (Touvron et al., 2023; AI, 2024b), Gemma (Team et al., 2024), and GPT-4 (Achiam et al., 2023), which showcase remarkable zero-shot and few-shot in-context learning abilities.

Efficient language models. Despite the promise of transformer-based LMs, the quadratic computational complexity and the linearly increasing KV cache size of attention modules with longer sequences limit their processing efficiency. To address this, efficient LMs featuring sub-quadratic complexity in sequence length and strong scaling properties have emerged (Peng et al., 2023a; Sun et al., 2023; Gu & Dao, 2023; Dao & Gu, 2024; Yang et al., 2023; Katharopoulos et al., 2020). As pointed out by (Gu & Dao, 2023), popular efficient LM architectures such as RWKV (Peng et al., 2023a) and RetNet (Sun et al., 2023) can be viewed as variants of SSMs (Gu et al., 2021a;b). Mamba(Gu & Dao, 2023), one of the most widely used SSMs, improves upon previous SSMs by selectively propagating or forgetting information along the sequence length in an input-dependent manner. Follow-up works such as Mamba2 (Dao & Gu, 2024) and GLA (Yang et al., 2023) introduce more hardware-friendly gating mechanisms to enhance training throughput over Mamba.

Hybrid language models. To combine the processing efficiency of SSMs with the recall capabilities of transformers, an emerging trend is the creation of hybrid models that incorporate both types of operators. Specifically, (Park et al., 2024) proposes a hybrid model called MambaFormer, which interleaves Mamba and attention modules to improve in-context learning capabilities. Jamba (Lieber et al., 2024) and Zamba (Glorioso et al., 2024) also develop sequentially stacked Mamba-Attention hybrid models. Samba (Ren et al., 2024) introduces a structure that sequentially stacks Mamba, SWA, and MLP layers by repeating the Mamba-MLP-SWA-MLP structure, achieving constant throughput as sequence lengths increase. Other recent work has also explored hybrid models that mix either linear RNNs or convolutions with attention (De et al., 2024; Pilault et al., 2024; Saon et al., 2023; Yang et al., 2024).

5 Conclusion

In this work, we present Hymba, a new family of small LMs featuring a hybrid-head architecture that combines the high-resolution recall capabilities of attention heads with the efficient context summarization of SSM heads. To further optimize the performance of Hymba, we introduce learnable meta tokens, which act as a learned cache for both attention and SSM heads, enhancing the model's focus on salient information. Through the roadmap of Hymba, comprehensive evaluations, and ablation studies, we demonstrate that Hymba sets new SOTA performance across a wide range of tasks, achieving superior results in both accuracy and efficiency. Additionally, our work provides valuable insights into the advantages of hybrid-head architectures, offering a promising direction for future research in efficient LMs.

REFERENCES

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.

Cartesia AI. The On-Device Intelligence Update. 2024a. URL `https://cartesia.ai/blog/on-device`.

Meta AI. Introducing Meta Llama 3: The most capable openly available LLM to date. 2024b. URL `https://ai.meta.com/blog/meta-llama-3/`.

Meta AI. Llama 3.2: Revolutionizing edge AI and vision with open, customizable models. 2024c. URL `https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/`.

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

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Leandro von Werra, and Thomas Wolf. Smollm - blazingly fast and remarkably powerful, 2024.

Norah Alzahrani, Hisham Abdullah Alyahya, Yazeed Alnumay, Sultan Alrashed, Shaykhah Alsubaie, Yusef Almushaykeh, Faisal Mirza, Nouf Alotaibi, Nora Altwairesh, Areeb Alowisheq, et al. When benchmarks are targets: Revealing the sensitivity of large language model leaderboards. *arXiv preprint arXiv:2402.01781*, 2024.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. *arXiv preprint arXiv:2402.18668*, 2024a.

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, Dylan Zinsley, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff, 2024b.

Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, et al. Stable lm 2 1.6 b technical report. *arXiv preprint arXiv:2402.17834*, 2024.

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

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Smollm-corpus, 2024. URL `https://huggingface.co/datasets/HuggingFaceTB/smollm-corpus`.

Assaf Ben-Kish, Itamar Zimerman, Shady Abu-Hussein, Nadav Cohen, Amir Globerson, Lior Wolf, and Raja Giryes. Decimamba: Exploring the length extrapolation potential of mamba, 2024. URL `https://arxiv.org/abs/2406.14528`.

Stella Biderman, Hailey Schoelkopf, Quentin Gregory Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, et al. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pp. 2397–2430. PMLR, 2023.

Yelysei Bondarenko, Markus Nagel, and Tijmen Blankevoort. Quantizable transformers: Removing outliers by helping attention heads do nothing. *Advances in Neural Information Processing Systems*, 36:75067–75096, 2023.

William Brandon, Mayank Mishra, Aniruddha Nrusimha, Rameswar Panda, and Jonathan Ragan Kelly. Reducing transformer key-value cache size with cross-layer attention. *arXiv preprint arXiv:2405.12981*, 2024.

Mikhail S Burtsev, Yuri Kuratov, Anton Peganov, and Grigory V Sapunov. Memory transformer. *arXiv preprint arXiv:2006.11527*, 2020.

Cosmopedia/evaluation. Cosmopedia/evaluation at main · huggingface/cosmopedia. URL `https://github.com/huggingface/cosmopedia/tree/main/evaluation`.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. *arXiv preprint arXiv:2405.21060*, 2024.

Timothée Darcet, Maxime Oquab, Julien Mairal, and Piotr Bojanowski. Vision transformers need registers. *arXiv preprint arXiv:2309.16588*, 2023.

Soham De, Samuel L Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, et al. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *arXiv preprint arXiv:2402.19427*, 2024.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

Shizhe Diao, Rui Pan, Hanze Dong, Kashun Shum, Jipeng Zhang, Wei Xiong, and Tong Zhang. Lmflow: An extensible toolkit for finetuning and inference of large foundation models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 3: System Demonstrations)*, pp. 116–127, 2024.

Charles Dickens. *The Adventures of Oliver Twist*. Ticknor and Fields, 1868.

Hanze Dong, Wei Xiong, Bo Pang, Haoxiang Wang, Han Zhao, Yingbo Zhou, Nan Jiang, Doyen Sahoo, Caiming Xiong, and Tong Zhang. Rlhf workflow: From reward modeling to online rlhf. *arXiv preprint arXiv:2405.07863*, 2024.

Alexey Dosovitskiy. An image is worth 16x16 words: Transformers for image recognition at scale. *arXiv preprint arXiv:2010.11929*, 2020.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Clémentine Fourrier, Nathan Habib, Thomas Wolf, and Lewis Tunstall. Lighteval: A lightweight framework for llm evaluation, 2023. URL `https://github.com/huggingface/lighteval`.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 12 2023. URL `https://zenodo.org/records/10256836`.

Leo Gao, Jonathan Tow, Baber Abbasi, Stella Biderman, Sid Black, Anthony DiPofi, Charles Foster, Laurence Golding, Jeffrey Hsu, Alain Le Noac'h, Haonan Li, Kyle McDonell, Niklas Muennighoff, Chris Ociepa, Jason Phang, Laria Reynolds, Hailey Schoelkopf, Aviya Skowron, Lintang Sutawika, Eric Tang, Anish Thite, Ben Wang, Kevin Wang, and Andy Zou. A framework for few-shot language model evaluation, 07 2024. URL `https://zenodo.org/records/12608602`.

Paolo Glorioso, Quentin Anthony, Yury Tokpanov, James Whittington, Jonathan Pilault, Adam Ibrahim, and Beren Millidge. Zamba: A compact 7b ssm hybrid model. *arXiv preprint arXiv:2405.16712*, 2024.

Dirk Groeneveld, Iz Beltagy, Pete Walsh, Akshita Bhagia, Rodney Kinney, Oyvind Tafjord, Ananya Harsh Jha, Hamish Ivison, Ian Magnusson, Yizhong Wang, Shane Arora, David Atkinson, Russell Authur, Khyathi Chandu, Arman Cohan, Jennifer Dumas, Yanai Elazar, Yuling Gu, Jack Hessel, Tushar Khot, William Merrill, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Valentina Pyatkin, Abhilasha Ravichander, Dustin Schwenk, Saurabh Shah, Will Smith, Nishant Subramani, Mitchell Wortsman, Pradeep Dasigi, Nathan Lambert, Kyle Richardson, Jesse Dodge, Kyle Lo, Luca Soldaini, Noah A. Smith, and Hannaneh Hajishirzi. Olmo: Accelerating the science of language models. *Preprint*, 2024.

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

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. *arXiv preprint arXiv:2111.00396*, 2021a.

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, 2021b.

Yuxian Gu, Xu Han, Zhiyuan Liu, and Minlie Huang. Ppt: Pre-trained prompt tuning for few-shot learning. *arXiv preprint arXiv:2109.04332*, 2021c.

Chi Han, Qifan Wang, Hao Peng, Wenhan Xiong, Yu Chen, Heng Ji, and Sinong Wang. Lm-infinite: Zero-shot extreme length generalization for large language models. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pp. 3991–4008, 2024.

Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.

Huggingface. HuggingFaceTB/cosmo-1b. 2024. URL `https://huggingface.co/HuggingFaceTB/cosmo-1b`.

Samy Jelassi, David Brandfonbrener, Sham M Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. *arXiv preprint arXiv:2402.01032*, 2024.

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.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pp. 5156–5165. PMLR, 2020.

Brian Lester, Rami Al-Rfou, and Noah Constant. The power of scale for parameter-efficient prompt tuning. *arXiv preprint arXiv:2104.08691*, 2021.

Jeffrey Li, Alex Fang, Georgios Smyrnis, Maor Ivgi, Matt Jordan, Samir Gadre, Hritik Bansal, Etash Guha, Sedrick Keh, Kushal Arora, Saurabh Garg, Rui Xin, Niklas Muennighoff, Reinhard Heckel, Jean Mercat, Mayee Chen, Suchin Gururangan, Mitchell Wortsman, Alon Albalak, Yonatan Bitton, Marianna Nezhurina, Amro Abbas, Cheng-Yu Hsieh, Dhruba Ghosh, Josh Gardner, Maciej Kilian, Hanlin Zhang, Rulin Shao, Sarah Pratt, Sunny Sanyal, Gabriel Ilharco, Giannis Daras, Kalyani Marathe, Aaron Gokaslan, Jieyu Zhang, Khyathi Chandu, Thao Nguyen, Igor Vasiljevic, Sham Kakade, Shuran Song, Sujay Sanghavi, Fartash Faghri, Sewoong Oh, Luke Zettlemoyer, Kyle Lo, Alaaeldin El-Nouby, Hadi Pouransari, Alexander Toshev, Stephanie Wang, Dirk Groeneveld, Luca Soldaini, Pang Wei Koh, Jenia Jitsev, Thomas Kollar, Alexandros G. Dimakis, Yair Carmon, Achal Dave, Ludwig Schmidt, and Vaishaal Shankar. Datacomp-lm: In search of the next generation of training sets for language models, 2024.

Opher Lieber, Barak Lenz, Hofit Bata, Gal Cohen, Jhonathan Osin, Itay Dalmedigos, Erez Safahi, Shaked Meirom, Yonatan Belinkov, Shai Shalev-Shwartz, et al. Jamba: A hybrid transformer-mamba language model. *arXiv preprint arXiv:2403.19887*, 2024.

Akide Liu, Jing Liu, Zizheng Pan, Yefei He, Gholamreza Haffari, and Bohan Zhuang. Minicache: Kv cache compression in depth dimension for large language models. *arXiv preprint arXiv:2405.14366*, 2024a.

Nelson F Liu, Kevin Lin, John Hewitt, Ashwin Paranjape, Michele Bevilacqua, Fabio Petroni, and Percy Liang. Lost in the middle: How language models use long contexts. *Transactions of the Association for Computational Linguistics*, 12:157–173, 2024b.

Shih-yang Liu, Chien-Yi Wang, Hongxu Yin, Pavlo Molchanov, Yu-Chiang Frank Wang, Kwang-Ting Cheng, and Min-Hung Chen. Dora: Weight-decomposed low-rank adaptation. In *Forty-first International Conference on Machine Learning*, 2024c.

Xiaoran Liu, Hang Yan, Shuo Zhang, Chenxin An, Xipeng Qiu, and Dahua Lin. Scaling laws of rope-based extrapolation. *arXiv preprint arXiv:2310.05209*, 2023.

Colin Lockard, Prashant Shiralkar, and Xin Luna Dong. OpenCeres: When open information extraction meets the semi-structured web. In Jill Burstein, Christy Doran, and Thamar Solorio (eds.), *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)*, pp. 3047–3056, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1309. URL https://aclanthology.org/N19-1309.

Ang Lv, Kaiyi Zhang, Yuhan Chen, Yulong Wang, Lifeng Liu, Ji-Rong Wen, Jian Xie, and Rui Yan. Interpreting key mechanisms of factual recall in transformer-based language models. *arXiv preprint arXiv:2403.19521*, 2024.

Sachin Mehta, Mohammad Hossein Sekhavat, Qingqing Cao, Maxwell Horton, Yanzi Jin, Chenfan Sun, Iman Mirzadeh, Mahyar Najibi, Dmitry Belenko, Peter Zatloukal, et al. Openelm: An efficient language model family with open-source training and inference framework. *arXiv preprint arXiv:2404.14619*, 2024.

Jack Merullo, Carsten Eickhoff, and Ellie Pavlick. Talking heads: Understanding inter-layer communication in transformer language models. *arXiv preprint arXiv:2406.09519*, 2024.

Evan Miller. Attention is off by one. URL https://www.evanmiller.org/attention-is-off-by-one.html.

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.

Guilherme Penedo, Hynek Kydlíček, Loubna Ben allal, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, and Thomas Wolf. The fineweb datasets: Decanting the web for the finest text data at scale, 2024. URL https://arxiv.org/abs/2406.17557.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, et al. Rwkv: Reinventing rnns for the transformer era. *arXiv preprint arXiv:2305.13048*, 2023a.

Bowen Peng, Jeffrey Quesnelle, Honglu Fan, and Enrico Shippole. Yarn: Efficient context window extension of large language models. *arXiv preprint arXiv:2309.00071*, 2023b.

Jonathan Pilault, Mahan Fathi, Orhan Firat, Chris Pal, Pierre-Luc Bacon, and Ross Goroshin. Block-state transformers. *Advances in Neural Information Processing Systems*, 36, 2024.

Zhen Qin, Dong Li, Weigao Sun, Weixuan Sun, Xuyang Shen, Xiaodong Han, Yunshen Wei, Baohong Lv, Fei Yuan, Xiao Luo, et al. Scaling transnormer to 175 billion parameters. *arXiv preprint arXiv:2307.14995*, 2023.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. Hgrn2: Gated linear rnns with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad, 2018.

Liliang Ren, Yang Liu, Yadong Lu, Yelong Shen, Chen Liang, and Weizhu Chen. Samba: Simple hybrid state space models for efficient unlimited context language modeling. *arXiv preprint arXiv:2406.07522*, 2024.

Adam Roberts, Hyung Won Chung, Anselm Levskaya, Gaurav Mishra, James Bradbury, Daniel Andor, Sharan Narang, Brian Lester, Colin Gaffney, Afroz Mohiuddin, Curtis Hawthorne, Aitor Lewkowycz, Alex Salcianu, Marc van Zee, Jacob Austin, Sebastian Goodman, Livio Baldini Soares, Haitang Hu, Sasha Tsvyashchenko, Aakanksha Chowdhery, Jasmijn Bastings, Jannis Bulian, Xavier Garcia, Jianmo Ni, Andrew Chen, Kathleen Kenealy, Jonathan H. Clark, Stephan Lee, Dan Garrette, James Lee-Thorp, Colin Raffel, Noam Shazeer, Marvin Ritter, Maarten Bosma, Alexandre Passos, Jeremy Maitin-Shepard, Noah Fiedel, Mark Omernick, Brennan Saeta, Ryan Sepassi, Alexander Spiridonov, Joshua Newlan, and Andrea Gesmundo. Scaling up models and data with `t5x` and `seqio`. *arXiv preprint arXiv:2203.17189*, 2022. URL `https://arxiv.org/abs/2203.17189`.

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)*, pp. 1–5. IEEE, 2023.

Yikang Shen, Zhen Guo, Tianle Cai, and Zengyi Qin. Jetmoe: Reaching llama2 performance with 0.1 m dollars. *arXiv preprint arXiv:2404.07413*, 2024.

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

Gemma Team, Morgane Riviere, Shreya Pathak, Pier Giuseppe Sessa, Cassidy Hardin, Surya Bhupatiraju, Léonard Hussenot, Thomas Mesnard, Bobak Shahriari, Alexandre Ramé, et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

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.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, et al. Zephyr: Direct distillation of lm alignment. *arXiv preprint arXiv:2310.16944*, 2023.

A Vaswani. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, et al. An empirical study of mamba-based language models. *arXiv preprint arXiv:2406.07887*, 2024.

Zekun Moore Wang, Zhongyuan Peng, Haoran Que, Jiaheng Liu, Wangchunshu Zhou, Yuhan Wu, Hongcheng Guo, Ruitong Gan, Zehao Ni, Man Zhang, et al. Rolellm: Benchmarking, eliciting, and enhancing role-playing abilities of large language models. *arXiv preprint arXiv:2310.00746*, 2023.

Guangxuan Xiao, Yuandong Tian, Beidi Chen, Song Han, and Mike Lewis. Efficient streaming language models with attention sinks. *arXiv preprint arXiv:2309.17453*, 2023.

Fanjia Yan, Huanzhi Mao, Charlie Cheng-Jie Ji, Tianjun Zhang, Shishir G. Patil, Ion Stoica, and Joseph E. Gonzalez. Berkeley function calling leaderboard. `https://gorilla.cs.berkeley.edu/blogs/8_berkeley_function_calling_leaderboard.html`, 2024.

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

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *arXiv preprint arXiv:2406.06484*, 2024.

Xinrong Zhang, Yingfa Chen, Shengding Hu, Zihang Xu, Junhao Chen, Moo Hao, Xu Han, Zhen Thai, Shuo Wang, Zhiyuan Liu, et al. Inf bench: Extending long context evaluation beyond 100k tokens. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 15262–15277, 2024.

Itamar Zimerman, Ameen Ali, and Lior Wolf. A unified implicit attention formulation for gated-linear recurrent sequence models. *arXiv preprint arXiv:2405.16504*, 2024.

A EXTENSIVE BENCHMARK FOR MORE HYMBA MODEL VARIANTS

A.1 COMPARISON WITH SOTA TINY LMS AT 350M AND 125M SCALES

Besides our 1.5B model, we also evaluate the 350M and 125M Hymba models on a diverse set of benchmarks in Tab. 6 and Tab. 7, respectively. Consistent with the results of our 1.5B model, Hymba-350M/125M models outperform the SOTA tiny LMs across most of tasks and achieve the best average score. This indicates that our Hymba scales effectively across different model sizes.

Table 6: Benchmark Hymba with SOTA tiny LMs, all of which have fewer than 200M parameters. All results are obtained through HUGGINGFACE/LIGHTEVAL, following Allal et al. (2024).

Model	#Params.	MMLU (cloze) ↑	ARC (c+e) ↑	PIQA ↑	Hella. ↑	OBQA ↑	Wino. ↑	Avg. ↑
Mamba-130m-hf	130M	27.41	33.01	63.33	33.86	30.40	51.54	42.43
Cerebras-GPT	111M	25.56	27.75	58.16	26.32	25.40	50.28	37.58
GPT-neo	125M	27.25	31.30	62.35	29.68	29.20	51.54	40.81
LaMini-GPT	124M	26.47	33.26	62.89	30.05	27.80	50.75	40.95
Opt	125M	25.67	31.25	61.97	31.04	29.00	53.20	41.29
GPT2	137M	26.29	31.09	62.51	29.76	29.40	49.72	40.50
Pythia	160M	26.68	31.92	61.64	29.55	27.80	49.49	40.08
MobileLM	125M	-	35.51	65.30	38.90	**39.50**	**53.10**	46.46
SmolLM	135M	30.23	43.99	**69.60**	42.30	33.60	52.70	48.44
Hymba	125M	**31.12**	**44.95**	68.50	**45.54**	35.52	52.25	**49.35**

Table 7: Benchmark Hymba with SOTA tiny LMs, all of which have fewer than 400M parameters. All results are obtained through HUGGINGFACE/LIGHTEVAL, following Allal et al. (2024).

Model	#Params.	MMLU (cloze) ↑	ARC (c+e) ↑	PIQA ↑	Hella. ↑	OBQA ↑	Wino. ↑	Avg. ↑
Bloom	560M	27.49	32.86	65.13	35.98	28.80	51.70	42.89
Cerebras-GPT-256M	256M	25.91	29.69	61.37	28.44	28.00	51.62	39.82
Cerebras-GPT-590M	590M	26.93	32.40	62.84	31.99	28.40	50.12	41.15
Opt	350M	26.57	31.94	64.36	36.09	27.80	52.57	42.55
Pythia	410M	28.94	35.05	66.92	39.21	28.40	52.80	44.48
GPT2-medium	380M	27.77	34.30	66.38	37.06	31.20	49.49	43.69
MobileLM	350M	-	43.65	68.60	49.60	**40.00**	57.60	51.89
SmolLM	360M	34.17	51.10	72.00	53.80	37.20	53.70	53.56
Hymba	350M	**34.54**	**52.46**	**72.91**	**55.08**	38.40	**57.85**	**55.34**

A.2 APPLE-TO-APPLE COMPARISON WITH OTHER ARCHITECTURES AT 300M AND 1B SCALE

In Sec. 3.3 of our main paper, we show the apple-to-apple architecture comparison under the same settings with a 1B model size. In addition to superior performance on both general and recall-intensive benchmarks, Hymba also has lower validation loss and more stable gradient norm during pre-training as shown in Fig. 6

We further validate the superiority of our architecture at the 300M size with a different training dataset to ensure the generalization of our findings. Specifically, we train different 300M model architectures on 100B tokens from FineWeb (Penedo et al., 2024). We set peak learning rates to 5e-4 and use warmup and cosine decay scheduler. The training sequence length is set to 1024. As shown in Tab. 8, Hymba achieves the best performance in almost all tasks (with a second-best result in one task), yielding an average accuracy boost of +1.45% compared to the strongest baseline.

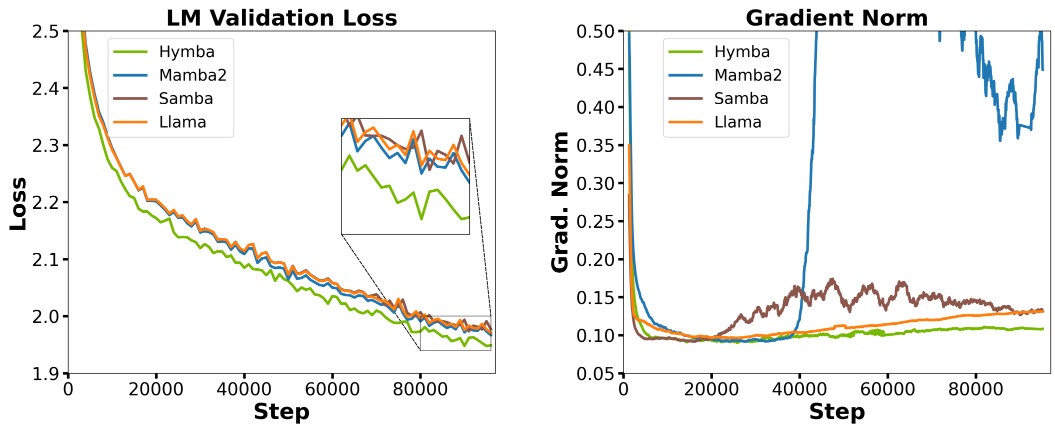

Figure 6: LM validation loss and gradient norm during pre-training. All models have the same scale (1B) and exactly the same training receipt.

Table 8: Apple-to-apple comparison of our Hymba, pure Mamba (Gu & Dao, 2023), Mamba with FFN, Llama3 (Dubey et al., 2024) style, and Samba- (Ren et al., 2024) style (Mamba-FFN-Attn-FFN) architectures. All models have 300M parameters and are trained for 100B tokens from FineWeb dataset (Penedo et al., 2024) with exactly the same training recipes. All results are obtained through LM-EVALUATION-HARNESS (Gao et al., 2023). The best and second best results are highlighted in bold and underline, respectively.

Task Type	Arch. Style (300M)	Mamba	Mamba w/ FFN	Llama3	Samba	Hymba
Language	Wiki. ppl. ↓	30.78	33.41	<u>30.04</u>	31.41	**28.53**
	LMB. ppl. ↓	19.95	23.64	20.53	<u>19.75</u>	**15.45**
Recall Intensive	SQuAD-C ↑	21.31	17.56	22.10	<u>39.88</u>	**45.24**
	SWDE ↑	17.14	13.10	<u>57.86</u>	22.14	**58.33**
	Avg. ↑	19.23	15.33	<u>39.98</u>	31.01	**51.79**
Common-sense Reasoning and Question-answering	Lambda ↑	38.95	36.37	40.15	<u>40.59</u>	**44.67**
	PIQA ↑	69.64	69.26	<u>70.29</u>	69.86	**70.73**
	ARC-C ↑	24.91	25.00	24.83	<u>25.76</u>	**26.28**
	ARC-E ↑	50.67	50.34	50.24	49.79	**53.20**
	Hella. ↑	44.95	44.08	45.69	<u>46.45</u>	**48.23**
	Wino. ↑	51.70	51.78	<u>52.64</u>	52.49	**53.35**
	TruthfulQA ↑	23.86	26.23	**28.97**	27.27	<u>27.87</u>
	SIQA ↑	39.20	39.53	39.66	<u>39.92</u>	**39.92**
	Avg.	42.98	42.82	<u>44.08</u>	44.02	**45.53**

B Ablation Studies of Our Hymba Architecture

We perform further ablation studies and analyses of the design factors in our Hymba.

The ratio of SSMs and attention in hybrid heads. To determine the proper number of attention heads, we start with a Mamba model and gradually replace Mamba's hidden dimensions with attention heads, maintaining the same overall model size. As shown in Tab. 9 (1)∼(4), we observe that model performance improves as the ratio of attention parameters increases and gradually saturates when the parameter ratio of attention to Mamba reaches 1:2.12. We stop introducing more attention heads, considering that adding more would bring increased memory overhead.

Table 9: Ablation study of the design choices of Hymba. The design finally adopted by Hymba is highlighted in **bold**. Specifically, the task lists are the same as those in Tab. 3. The throughput is measured with a 8k sequence length and a 128 batch size on an NVIDIA A100 GPU. The cache size is measured with a 16k sequence length, assuming the FP16 format.

Design Factor		Configuration	Param. Ratio Attn:Mamba	Avg. (General) ↑	Avg. (Recall) ↑	Throughput (Token/s) ↑	Cache (MB) ↓
Attn/Mamba Ratio	1)	Mamba Heads Only	0:1	42.98	19.23	4720.8	1.87
	2)	Mamba + 4 Attn Heads	1:8.48	44.20	44.65	3278.1	197.39
	3)	Mamba + 8 Attn Heads	1:4.24	44.95	52.53	1816.5	394.00
	4)	Mamba + 16 Attn Heads	1:2.12	45.08	56.46	656.6	787.22
	5)	**4) + GQA**	1:3.64	45.19	49.90	876.7	295.69
	6)	Attn Heads Only (LLaMA)	1:0	44.08	39.98	721.1	829.44
Sliding Window	7)	5) + All SWA's	1:3.64	44.42	29.78	4485.09	5.51
	8)	**5) + SWA's + Full Attn**	1:3.64	44.56	48.79	2399.7	76.28
	9)	**8) + Cross-layer KV sharing**	1:5.23	45.16	48.04	2756.5	76.30
	10)	6) + Same KV compression	1:0	43.60	28.18	3710.0	56.62
Fusion	11)	9) Replace Mean by Concat	1: 5.82	44.56	48.94	1413.9	76.30
Meta Tokens	12)	1) + Meta Tokens	0:1	44.01	19.34	4712.8	1.87
	13)	**9) + Meta Tokens**	1:5.23	45.53	51.79	2695.8	76.87

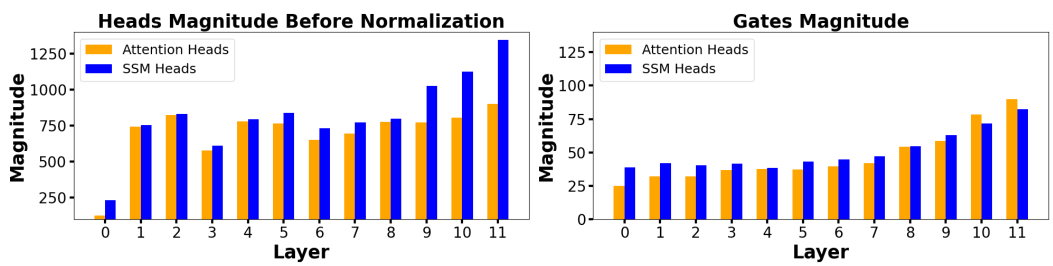

Figure 7: Left: visualization of output magnitudes of attention and SSM heads. SSM heads consistently have higher output magnitude than attention heads due to their structure. Right: visualization of attention and SSM heads' gate magnitudes. Through model learning, the relative magnitudes of attention and SSM gates vary across different layers.

There are two interesting observations: (1) Although the attention-only model outperforms the Mamba-only model, the hybrid model with both attention and Mamba heads achieves the best performance; (2) with further KV cache optimization, the ratio of attention heads decreases further. In our final model, attention heads occupy no more than 1/5 of the Mamba heads, yet significantly boost both recall and commonsense reasoning compared to the vanilla Mamba. This suggests that the hybrid model leverages the strengths and diversity of both attention and SSM heads, achieving a better trade-off between efficiency and performance.

The hybrid-head fusion strategy. We have explored two straightforward methods to fuse the outputs of attention and SSM heads: concatenation and mean. For concatenation, we combine the outputs of all heads and use a linear layer to project the concatenated output to the final output dimension. However, the parameter size of the linear layer increases with both the number of heads and the head dimensions. Additionally, based on the empirical comparison between Tab. 9 (9) and (11), the performance of concatenation fusion is not better than the simple mean fusion. Therefore, we adopt the mean fusion strategy in our final design.

Impact of KV cache optimization. After applying a series of KV cache optimization techniques, moving from Tab. 9 (5) to Tab. 9 (9), we observe that our Hymba maintains comparable recall and commonsense reasoning accuracy while being 2.74× faster. In contrast, applying the same KV cache optimization to a pure Transformer, as seen in the comparison between Tab. 9 (6) and (10), results in a recall accuracy drop of 10% or more and degraded commonsense reasoning accuracy. This supports our analysis in Sec. 2.3, showing that the presence of SSM heads in our hybrid-head module has already summarized the global context, allowing us to more aggressively replace global full attention with local attention in our hybrid model.

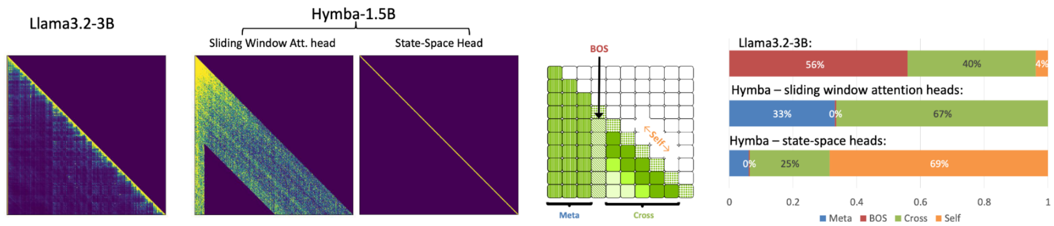

Figure 8: Sum of attention score from different categories (i.e., 'Meta', 'BOS', 'Self', 'Cross') in Llama-3.2-3B and Hymba-1.5B. Note that the parallel SSM and attention structure in the latter disentangles the attention map.

C Visualization and Analysis of Attention Maps

To better analyze the attention distributions, we categorize elements in the attention map into four types: (1) 'Meta': attention scores from all real tokens to meta tokens. This category reflects the model's preference for attending to meta tokens. In attention map, they are usually located in the first few columns (e.g., 128 for Hymba) if a model has meta tokens. (2) 'BOS': attention scores from all real tokens to the beginning-of-sequence token. In the attention map, they are usually located in the first column right after the meta tokens. (3) 'Self': attention scores from all real tokens to themselves. In the attention map, they are usually located in the diagonal line. (4) 'Cross': attention scores from all real tokens to other real tokens. In the attention map, they are usually located in the off-diagonal area.

In Fig. 8, we visualize the real attention maps from Llama-3.2-3B and Hymba-1.5B on texts from Oliver Twist Chapter 29 (Dickens, 1868) and sum up the attention scores from different categories. The summed scores are normalized by the context length. For SSM heads, we follow Ben-Kish et al. (Ben-Kish et al., 2024) and Zimerman et al. (Zimerman et al., 2024) to calculate their attention maps and normalize the attention maps to ensure each row sums to 1.

We observe that the attention pattern of Hymba is significantly different from the vanilla Transformers. In vanilla Transformers, attention scores are more concentrated on 'BOS', which is consistent with the findings in (Xiao et al., 2023). In addition, vanilla Transformers also have a higher proportion of 'Self' attention scores. In Hymba, meta tokens, attention heads and SSM heads work complimentary to each other, leading to a more balanced distribution of attention scores across different types of tokens. Specifically, meta tokens offload the attention scores from 'BOS', allowing the model to focus more on the real tokens. SSM heads summarize the global context, which focus more on current tokens (i.e., 'Self' attention scores). Attention heads, on the other hand, pay less attention to 'Self' and 'BOS' tokens, and more attention to other tokens (i.e., 'Cross' attention scores). This suggests that the hybrid-head design of Hymba can effectively balance the attention distribution across different types of tokens, potentially leading to better performance.

D Meta Tokens: More Analysis and Visualization

Relationship with prior works. Learnable tokens have also been leveraged in previous transformer-based models. Previous prompt tuning works (Lester et al., 2021; Gu et al., 2021c) prepend learnable prompts while keeping the model weights frozen during the task-specific tuning stage, aiming to adapt a pretrained LM to downstream tasks in a parameter-efficient manner. (Burtsev et al., 2020) introduces both learnable tokens and corresponding memory update modules to augment the memory mechanism in transformers. (Darcet et al., 2023) appends a set of learnable tokens called registers to the image patches of vision transformers (Dosovitskiy, 2020) to store global information and improve visual recognition. Our method combines ideas from all of these works in a more flexible manner. It optimizes the meta tokens jointly with model weights during the pretraining stage, is compatible with sliding window attention heads and other attention types or SSMs, and converts the meta tokens into KV-cache initialization during inference, without modifying the architecture.

Meta tokens reduce attention map entropy. We visualize the entropy of the attention map for both the attention and SSM heads (Ali et al., 2024; Ben-Kish et al., 2024) before and after intro-

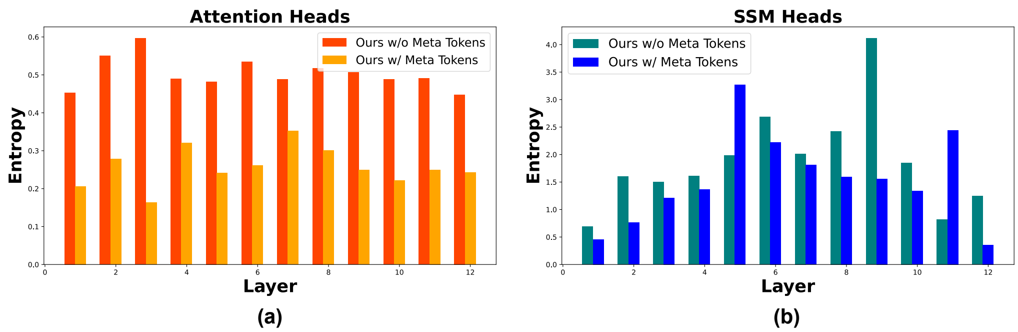

Figure 9: Visualize the layer-wise attention map entropy of (a) attention heads, and (b) SSM heads with and without meta tokens.

ducing meta tokens. As introduced in Sec. 2.4 of our main paper, the attention map entropy reflects the distribution of attention scores across tokens, where lower entropy indicates stronger retrieval effects (Ren et al., 2024), as the attention scores are concentrated around a smaller subset of tokens.

As shown in Fig. 9, we observe that after introducing meta tokens, both the attention and SSM heads exhibit an overall reduction in entropy. Specifically, entropy is significantly reduced in all attention heads and in 10 out of 12 layers of the SSM heads. This suggests that meta tokens can reduce attention map entropy, potentially helping both the attention and SSM heads focus more on a subset of important tokens that contribute most to task performance, as indicated by the boosted performance in Tab. 9.

E Illustration of Hymba's Attention Mask

As illustrated in Sec. 2 of our main paper, one Hymba layer is composed of SSM heads and global or local attention heads, augmented by meta tokens. To better understand the design, we visualize the attention mask of a Hymba layer with sliding window attention heads in Fig. 10. Specifically, since the meta tokens are visible to all subsequent tokens to modulate their attention mechanism, the first columns of the attention mask are set to ones; In addition, only nearby tokens falling within a sliding window are visible to the sliding window attention heads and SSM heads follow standard autoregressive attention patterns. As such, when applying all three attention masks together, the overall attention pattern is shown in the rightmost subfigure in Fig. 10.

F Pretraining and Post-training Implementation Details

Pretraining settings. We train Hymba-125M/350M/1.5B models on 1.3T tokens, using a mix of DCLM-Baseline-1.0 (Li et al., 2024), SmolLM-Corpus (Ben Allal et al., 2024), and an internal high-quality dataset for 1T, 250B, and 50B tokens, respectively. We adopt the WSD learning rate scheduler (Hu et al., 2024) with three phases: (1) warmup steps set to 1% of the total steps, (2) a stable phase maintaining the peak learning rate of 3e-3, and (3) a decay phase reducing the learning rate to 1e-5 over 20% of the total steps, while gradually annealing to smaller, higher-quality datasets

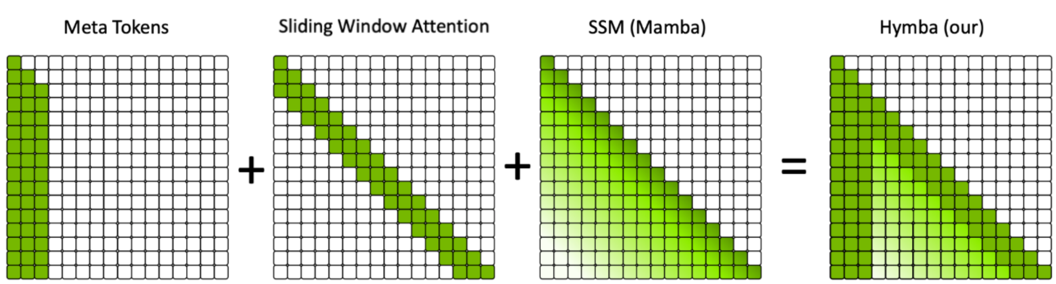

Figure 10: Visualize the attention mask of a Hymba layer with sliding window attention heads and SSM heads.

like SmolLM-Corpus and the internal dataset. We use a sequence length of 2048 and a batch size of 2M tokens throughout the training process, which is conducted on 128 NVIDIA A100 GPUs.

Implementation details of post-training. We post-trained our 1.5B base model with a two-stage strategy: the first full-finetuning (FFT) stage and another direct preference optimization (DPO) (Rafailov et al., 2024) training. The learning rates are 5e-5, and 3e-6 for FFT and DPO, respectively. Both FFT and DPO training are carried out for one epoch with a cosine scheduler. The global batch size is set to 1024. To accelerate training, we follow the training recipe (Tunstall et al., 2023; Diao et al., 2024; Dong et al., 2024) to pack the samples and use a block size of 2048. We implement the finetuning and DPO training with the LMFlow toolkit (Diao et al., 2024). In addition to full-finetuning, we also leverage Dora (Liu et al., 2024c) to do parameter-efficient finetuning.

Baselines and downstream tasks. We compare Hymba with competitive lightweight instruction-tuned models, i.e., Llama-3.2-1B-Instruct (AI, 2024c), Gemma-2-2B-Instruct (Team et al., 2024), OpenELM-1-1B-Instruct (Mehta et al., 2024), and SmolLM-1.7B-Instruct (Allal et al., 2024). We test the instruction-tuned models on MMLU (5-shot), IFEval, GSM8K (5-shot), GPQA (0-shot), and Berkeley Function-Calling Leaderboard v2 (BFCLv2) (Yan et al., 2024). For BFCLv2, we use the official code from Gorilla project (Yan et al., 2024) and evaluate the BFCLv2-live category, including *live_simple*, *live_multiple*, *live_parallel*, *live_parallel_multiple*, *live_relevance*. We exclude *live_irrelevance*, since we found some baseline models without function calling abilities, could achieve high in the *live_irrelevance* category (where the model is not required to call function) and very low in other tasks, but still got high overall accuracy although these models are not helpful at all. For the remaining tasks, we directly use the lm-evaluation-harness (Gao et al., 2024).