Birdie: Advancing State Space Language Modeling with Dynamic Mixtures of Training Objectives

Sam Blouir[1], Jimmy Smith[2,3], Antonios Anastasopoulos[1,4], Amarda Shehu[1]

[1]Department of Computer Science, George Mason University, Fairfax, VA
[2]Stanford University, Stanford, CA [3]Liquid AI, Palo Alto, CA
[4]Archimedes AI, Athena RC, Athens, Greece
{sblouir,antonis,amarda}@gmu.edu, jsmith14@stanford.edu

Abstract

Efficient state space models (SSMs), including linear recurrent neural networks and linear attention variants, have emerged as potential alternative language models to Transformers. While efficient, SSMs struggle with tasks requiring in-context retrieval, such as text copying and associative recall, limiting their usefulness in practical settings. Prior work on how to meet this challenge has focused on the internal model architecture and not investigated the role of the training procedure. This paper proposes a new training procedure that improve the performance of SSMs on retrieval-intensive tasks. This novel pre-training procedure combines a bidirectional processing of the input with dynamic mixtures of pre-training objectives to improve the utilization of the SSM's fixed-size state. Our experimental evaluations show that this procedure significantly improves performance on retrieval-intensive tasks that challenge current SSMs, such as phone book lookup, long paragraph question-answering, and infilling tasks. Our findings offer insights into a new direction to advance the training of SSMs to close the performance gap with Transformers.[1]

1 Introduction

Due to their scaling properties (Hoffmann et al., 2022) and in-context learning (Garg et al., 2023), large Transformer models using attention (Bahdanau, 2014; Vaswani et al., 2017) are now prominent in natural language processing (NLP) and achieve effective performance in natural language generation tasks (NLG), including language modeling, machine translation, and question and answering (Q&A) (Yue et al., 2022; Xie et al., 2022; Kumar et al., 2021). However, the softmax attention mechanism cost scales quadratically with sequence length during training, and its key-value (KV) cache grows linearly with sequence length during inference. This leads to increasing costs for training and deployment as model providers continue to increase the context length (Dubey et al., 2024; Reid et al., 2024).

This trend in increasing context length has sparked a strong interest in developing efficient alternative sequence models. The goal is to maintain high performance while scaling effectively with longer sequences. Recent work has focused on recurrent models which offer two key advantages: subquadratic scaling for parallel processing and a fixed state size (in contrast to the growing KV cache in Transformer models) that enables constant-cost inference per step. These models come in different forms, ranging from state space model (SSM)-based methods, such as S4 (Gu et al., 2022), S5 (Smith et al., 2023), or Mamba (Gu and Dao, 2023)), to linear RNNs, such as RWKV (Peng et al., 2023), HGRU (Qin et al., 2023), and Hawk (De et al., 2024), to linear attention variants, such as RetNet (Sun et al., 2023) and GLA (Yang et al., 2024). These different methods vary in their exact parameterization and parallel computation, but all have an efficient, fixed-state size recurrence for inference. For brevity, we will generally refer to all of these methods as SSMs regardless of their exact parameterization or parallel computation path.

While some studies have shown the ability of SSMs to match Transformers in perplexity and some public benchmarks, an increasing line of work shows that current SSMs struggle on tasks that require long-range in-context abilities (Park et al., 2024), such as long-range retrieval (Wen et al., 2024), multi-query associative recall (Arora et al., 2023, 2024), and copying (Jelassi et al., 2024). These tasks are critical in NLP, where the ability to maintain and manipulate long-term dependencies is key to generating coherent text, following directions, copying sequences, and responding accurately to multiple queries. A typical approach

[1]All code and pre-trained models are available at https://www.github.com/samblouir/birdie, with support for JAX and PyTorch.

Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing, pages 9679–9705
November 12-16, 2024 ©2024 Association for Computational Linguistics

to address these weaknesses has been to formulate hybrid models (Poli et al., 2024) that interleave SSM layers with global attention layers (Mehta et al., 2023; Fu et al., 2023a; Park et al., 2024), or sliding window attention (Beltagy et al., 2020; Arora et al., 2024; De et al., 2024).[2] However, models with global attention layers still scale quadratically with sequence length and have a growing KV cache. Models that rely on sliding window attention also fail to perform in-context retrieval outside of the sliding window length (Arora et al., 2024; De et al., 2024).

The predominant focus on architecture to improve performance on long-range in-context abilities misses an opportunity to investigate the role of the pre-training objective(s) and potential interaction with model architecture. We note that prior work on SSMs predominantly utilizes a causal language modeling (CLM) pre-training objective.

In this paper we argue (and show) that in the presence of a fixed state size, a mixture of pre-training objectives can bias learning towards pertinent long-range interactions and that bidirectional processing of the context allows better utilization of the fixed state for such interactions. This paper makes the following key methodological contributions:

(1) We develop **novel pre-training objective mixtures** that confer SSMs strong performance on both standard downstream public benchmarks and recall and copying-intensive tasks where SSMs typically struggle, such as phone book retrieval tasks, infilling, and long paragraph Q&A.

(2) We show that **bidirectional processing of the context** combined with the pre-training objective mixtures can further boost performance. In addition, we develop a new bidirectional architecture for SSMs that allows a seamless transition from bidirectional processing of the context to causal generation of the response.

(3) To improve the practical ability to experiment with new pre-training objectives in the mixture, we propose **a dynamic mixture of pre-training objectives via reinforcement learning (RL)**. This allows for maximizing performance while simplifying the objective selection process.

The result is a new training procedure that significantly improves the performance of SSMs on recall-intensive tasks, making them more competitive with Transformers. We refer to this procedure as *Birdie*. While we do still observe a performance gap with Transformers on some tasks as the retrieval requirement becomes more difficult (e.g. increasing the number of retrievals required per example), our procedure makes the SSM performance degradation in these scenarios much less severe and expands the regime where these efficient methods can be useful. More broadly, our work points to considering the learning dynamics along with the inductive biases of SSM architectures in order to make better use of the fixed state size.

[2] The sliding window attention, introduced in Longformer (Beltagy et al., 2020), can be viewed as a form of a fixed-state size method.

2 Background and Related Work

This section relates background and prior work.

2.1 State Space Models

Given a length L sequence of inputs $\mathbf{x}_{1:L} \in \mathbb{R}^{L\times D}$, a general class of linear recurrences with hidden states $\mathbf{h}_{1:L} \in \mathbb{R}^{L\times N}$ and outputs $\mathbf{y}_{1:L} \in \mathbb{R}^{L\times D}$ can be computed as

$$\begin{aligned}\mathbf{h}_k &= \mathbf{A}_k\mathbf{h}_{k-1} + \mathbf{B}_k\mathbf{x}_k\\ \mathbf{y}_k &= \mathbf{g}(\mathbf{h}_k, \mathbf{x}_k)\end{aligned}$$

with state transition matrix $\mathbf{A}_k \in \mathbb{R}^{N\times N}$, input matrix $\mathbf{B}_k \in \mathbb{R}^{N\times U}$ and output function $\mathbf{g}(\cdot)$ to transform the hidden state into an output.

Many recent recurrent models fall within this SSM framework. Some are time-invariant, such that the dynamics parameters are static across time, i.e. $\mathbf{A}_k = \mathbf{A}$ and $\mathbf{B}_k = \mathbf{B}\ \forall k$. This includes state space layer/linear RNN variants such as S4 (Gu et al., 2022), S5 (Smith et al., 2023) and LRU (Orvieto et al., 2023) and as well as linear attention variants such as linear transformer (Katharopoulos et al., 2020) and RetNet (Sun et al., 2023). Other linear recurrent models have input-varying dynamics; these include state space layer/linear RNN variants such as Liquid-S4 (Hasani et al., 2022), HGRU (Qin et al., 2023), Mamba (Gu and Dao, 2023), Hawk (De et al., 2024), gated linear attention (Yang et al., 2024) methods, and prior work in linear RNNS (Balduzzi and Ghifary, 2016; Martin and Cundy, 2018; Bradbury et al., 2016; Lei et al., 2018). The linear (or conditionally linear) dependencies between time steps allow for efficient parallelization across the sequence via Fast Fourier Transforms (Gu et al., 2022; Fu et al., 2023b), parallel scans (Blelloch, 1990; Martin and Cundy, 2018; Smith et al., 2023) or other structured matrix operations (Yang et al., 2024) while also allowing for fast recurrences at inference.

In this work, we focus on input-varying SSMs, as they have provided better performance on language (Gu and Dao, 2023; De et al., 2024; Yang et al., 2024) compared to their time-invariant counterparts. This is generally attributed to their ability to ignore or forget contextually-irrelevant information. As an example, consider the Hawk model (De et al., 2024) which showed strong performance for attention-free methods on common max-likelihood evaluations. At its core, Hawk is powered by the Real-Gated LRU (RG-LRU), an input-dependent version of LRU. The mathematical formulation of the RG-LRU is:

$$\begin{aligned}
\mathbf{r}_t &= \sigma(\mathbf{W}^a \mathbf{x}_t,) \\
\mathbf{i}_t &= \sigma(\mathbf{W}^x \mathbf{x}_t), \\
\mathbf{a}_t &= \sigma(\Lambda)^{c r_t} \\
\mathbf{h}_t &= \mathbf{a}_t \odot \mathbf{h}_{t-1} + \sqrt{1 - \mathbf{a}_t^2} \odot (\mathbf{i}_t \odot \mathbf{x}_t)
\end{aligned}$$

where σ denotes the logistic-sigmoid function, Λ is a learnable parameter, and the constant c is set to 8.

2.2 Weaknesses of Current SSMs

While the fixed state size allows for efficient deployment at inference time, this limited state capacity also creates a tradeoff in how much information can be stored and used for in-context retrieval. These limitations have been characterized both theoretically (Arora et al., 2023; Jelassi et al., 2024; Wen et al., 2024) for simple tasks and empirically on both synthetic and more realistic tasks.

Park et al. (2024) and Arora et al. (2024) show that recurrent models struggle to perform synthetic multi-query associative recall (MQAR) (Arora et al., 2023) even when trained directly on the task. Jelassi et al. (2024) compared Pythia (Biderman et al., 2023) Transformers to Mamba (Gu and Dao, 2023) SSMs pre-trained on the same dataset and found that Mamba models significantly underperformed the Transformer baselines on retrieval tasks, such as phone-book lookup and long paragraph question-answering. Similarly, De et al. (2024) show that Hawk can perform phone-book retrieval for short lengths but fails to recall the correct phone number as the length grows. In the same work, even the Griffin model, which adds sliding window attention to Hawk struggles to retrieve phone numbers when the task exceeds the sliding window length. This phenomenon is also observed for Based (Arora et al., 2024), a hybrid of linear attention and sliding window attention on synthetic MQAR tasks.

Despite their computational appeal, current SSMs display significant weaknesses on the important skill of in-context retrieval. This limits how useful these models can be for practical deployment. We note that these prior works all train models with a simple CLM objective. These observations lead us in this work to question the standard training procedure and rethink it as a potential avenue for better utilization of the fixed state size and improved performance on in-context retrieval tasks.

2.3 Pre-training Objectives

Pre-training "instills" general-purpose knowledge and abilities (Raffel et al., 2020). While the default choice in NLP for a pre-training objective is CLM, or "next word prediction," several alternative objectives have been proposed that can improve model performance in general language tasks (Tay et al., 2022, 2023; Anil et al., 2023), code generation (Bavarian et al., 2022; Rozière et al., 2024), and multi-modal audio and vision Transformers (Chen et al., 2023).

For instance, *masked language modeling* (MLM) includes objectives where a limited number of tokens are replaced with a mask token, and the model must predict the original tokens. In its original conception with BERT (Devlin et al., 2019), each mask token represented one obfuscated input token. *Span corruption* (SC) extends the MLM objective to generative models (Guo et al., 2022). For a given input, several spans of tokens are replaced with unique sentinel tokens. The model then generates the masked tokens and their respective sentinel tokens. *Prefix language modeling* (PLM) does not calculate a loss on the prefix, and the model is allowed a bidirectional view of the context. During pre-training, input sequences are randomly split in two, with the prefix serving as context and the suffix as the target for the direct loss computation (Raffel et al., 2020). The *UL2* (Tay et al., 2023) objectives combine PLM and SC.

In this paper, we consider and build on the above representative pre-training objectives. As described in Section 3, we introduce new objectives and *dynamic* mixtures.

3 Methods

We propose two key methodological components to reduce the gap between SSMs and Transformers on in-context retrieval tasks: bidirectional processing of the input prompt or prefix and new mixtures of pre-training objectives designed to improve the ability of SSMs to perform retrieval. We then offer a new pre-training procedure that leverages RL for dynamic sampling of the pre-training objectives to reduce the burden of pre-selecting the optimal mixture ahead of time. We combine these components to define the *Birdie* training procedure. In the final part of this section, we also describe a baseline Gated SSM that allows for a simple implementation to test our methods.

3.1 Bidirectional processing

Bidirectional processing has shown advantages in generative Transformers with prefix language modeling objectives (Tay et al., 2023). Given the fixed state size of SSMs, we propose that bidirectionality could be even more advantageous, enabling SSMs to better triage state capacity, crucial for retrieval-intensive tasks. Our results indicate that bidirectional SSMs outperform their unidirectional counterparts on several such tasks. However, bidirectional processing in SSMs is not trivial. One needs to maintain their strict temporal coherence and causality. The state in an SSM represents the cumulative information up to the current time step. Incorporating information from the end of the context necessitates a careful approach to defining and updating the state. Implementation efficiency is also critical when incorporating bidirectional processing into a generative SSM.

We introduce a bidirectional architecture that addresses these challenges and matches a standard causal configuration in both compute and parameter count during pre-training. We divide the recurrent state into forward and reverse components. For the reverse state dimensions, we preserve causality in the causal/decoding region by masking out the forget gate dimensions that determine the reverse dynamics. This causes information traveling backwards in causal regions to never enter the state. We provide a mathematical description below[3] and provide an example in appendix section E.1.

$$
\begin{aligned}
x_t^{\text{forward}} &= x_{t,D_{\text{forward}}} \\
h_t^{\text{forward}} &= A_t \cdot h_{t-1}^{\text{forward}} + x_t^{\text{forward}} \\
x_t^{\text{rev}} &= x_{t,D_{\text{rev}}} \\
h_t^{\text{rev, prefix area}} &= A_t \cdot h_{t+1}^{\text{rev}} + x_t^{\text{rev}} \\
h_t^{\text{rev, causal area}} &= \mathbf{0} \cdot A_t \cdot h_{t+1}^{\text{rev}} + x_t^{\text{rev}} \\
h_t^{\text{rev}} &= [h_t^{\text{rev, prefix area}} \oplus h_t^{\text{rev, causal area}}] \\
h_t &= [h_t^{\text{forward}} \oplus h_t^{\text{rev}}]
\end{aligned}
$$

3.2 Pretraining Objectives for SSMs

We hypothesize CLM does not allow an SSM to learn to fully utilize its state for in-context retrieval. For the majority of the pre-training corpus, much of the "next-token prediction" loss can be substantially reduced by using information from local tokens. This may prevent the model from learning to retrieve. Note that while CLM pre-training does not seem to prevent Transformers from learning general retrieval skills, SSMs have a different inductive bias due to their relatively limited state. To improve utilization of this limited state capacity, we design pre-training objectives that attempt to force the SSM to learn to compress and retrieve throughout the pre-training process to improve in-context retrieval ability downstream.

We list the objectives and mixtures that we investigate in Table 1. We first briefly describe several previously proposed objectives that are core to our new methods:

Full Span Corruption (FSC): The model must generate the entire de-noised sequence. This is similar to BERT's MLM task, but the model generates the entire sequence rather than filling in masked tokens in-place. The same objective, albeit masking single tokens rather than spans, was included in an ablation in T5 and was referred to as BERT-style (Raffel et al., 2020). This tasks the model with maintaining a state where it can simultaneously copy from a context while generating new text conditioned on the context.

Deshuffling: The model is given an input sequence with shuffled tokens. The model must deshuffle the tokens to recreate the original sequence. We use two variations: one where 50% of the input tokens are shuffled, and another where all inputs tokens are shuffled.

Copying: We include copying tasks that do not involve denoising an input, inspired by recent work (Jelassi et al., 2024) that highlights challenges

[3] We provide efficient implementations of this in our codebase: https://github.com/samblouir/birdie

with SSMs in copying tasks. In Copying, the model must recreate the input sequence.

We build on these prior objectives and propose the following new pre-training objectives:

FSC with Deshuffling (FSC-D): This builds on FSC by also shuffling the non-corrupted spans, fusing span corruption, copying, and de-shuffling into one objective. We hypothesize this may help thwart over-fitting to unnatural traits of span corruption, such as always generating masked spans in the order they were shown.

Text: `Bird songs fill the early morning air`

Objectives	**Example**
CLM	In: – Tgt: `Bird songs fill the early morning air`
PLM	In: `Bird songs fill` Tgt: `the early morning air`
SC	In: `Bird [mask] the early [mask]` Tgt: `songs fill [mask] air [mask]`
FSC	In: `Bird [mask] the early [mask]` Tgt: `Bird songs fill the early morning air`
Deshuffling	In: `morning air early fill Bird songs the` Tgt: `Bird songs fill the early morning air`
Copying	In: `Bird songs fill the early morning air` Tgt: `Bird songs fill the early morning air`
FSC-D	In: `the early [mask] Bird [mask]` Tgt: `Bird songs fill the early morning air`
Selective Copying	In: `[start] C D [end] H [context] A B C D E F G H I` Tgt: `E F G [done]`
Mixtures	
BFR	Birdie-Fixed Ratio: A new mixture of the objectives above at fixed ratios found via ablations.
UL2	Fixed-ratio mixture of PLM and SC (Tay et al., 2023).

Table 1: CLM: Causal language modeling. PLM: Prefix language modeling. SC: Span corruption. FSC: Full SC. FSC-D: FSC with deshuffling. In: input; Tgt: target. New pre-training objectives and mixtures are **bolded**.

Selective Copying: We introduce here a novel variation, Selective Copying, in which the model is given beginnings and endings of spans in the context. The model must find and copy these spans to the output. This task strongly differs from standard copying - not all text is copied, and the spans to copy are not necessarily found in order. This can be seen as analog to the downstream phone book lookup task. The Selective Copying pre-training objective proposed here is inspired by Olsson et al. (2022), which introduces a similar version as a synthetic induction head task.

BFR: A mixture of all the objectives listed in Table 1 (except the UL2 mixturet) at fixed ratios. We discuss dynamic ratios in the next section.

3.3 Optimal Mixtures with Objective Sampling via Reinforcement Learning

Although we observed promising results in pilot runs, we found it difficult to pre-select optimal task mixture ratios. We also observed that seemingly optimal ratios can change during training, and different model architectures benefit from specialized ratios. Similar challenges in optimally scheduling and adjusting mixtures rates has been noted in Tay et al. (2022).

To address this, we propose a dynamic, automated curriculum that adapts pre-training task mixtures according to the evolving needs of the model. Our approach utilizes a critic model, which we use to predict rewards for proposed actions, given previous actions and observed outcomes. We define actions as training objectives along with their probabilities of being sampled or applied to incoming training data during training. Overall, this forms a classic multi-armed bandit framework and is related to a recent Gaussian Process approach for dynamic masking rates in MLM (Urteaga et al., 2023), which we found unable to model our diverse objectives and needs. We adopt a four-layer Gated SSM model (See Section 3.4) to directly predict per-objective rewards based on historical training data. We generate random actions and pick the action with greatest predicted reward.

We visualize loss, greedy-decoding accuracy, and sampling probabilities for training objective categories in Appendix A Figure 3. We observe trends, such as the observation that training on FSC appears to boost Copying and Deshuffling objectives to the extent that their sampling can be nearly shut-off. Other behaviors emerge, such as the selective copying ability continuing to form once the model sees sufficient amounts of these samples.

This approach, *Birdie*, which combines the new objectives described in Section 3.2 and the bidirectional processing described above in Section 3.1 consistently improves SSM performance on a variety of downstream tasks, as related in Section 4.

3.4 Gated SSM baseline

We define a generic Gated SSM baseline to verify that improvements from our training methods are not due to specifics of the SSM itself. The recurrence equations are:

$$
\begin{aligned}
\mathbf{i}_t &= \sigma(\mathbf{W}^i \mathbf{x}_t) \in \mathbb{R}^N, \\
\mathbf{z}_t &= \mathbf{W}^z \mathbf{x}_t \in \mathbb{R}^N, \\
\mathbf{o}_t &= \mathsf{GeLU}(\mathbf{W}^o \mathbf{x}_t) \in \mathbb{R}^N, \\
\mathbf{f}_t &= \sigma(\mathbf{W}^f \mathbf{x}_t) \in \mathbb{R}^N, \\
\\
\mathbf{h}_t &= \mathbf{f}_t \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \mathbf{z}_t, \\
\mathbf{y}_t &= \mathbf{W}^{out}(\mathbf{o}_t \odot \mathbf{h}_t),
\end{aligned}
$$

where σ is the logistic sigmoid function, $\mathbf{x}_t$ is the normalized input at time t, and $\mathbf{y}_t$ is the output that feeds into a residual connection. The operator $\odot$ represents element-wise multiplication. We note that this generic Gated SSM is closely related to a parallelizable version of an LSTM (Hochreiter et al., 1997) with the state dependency removed.

In our basic Gated SSM above, we fuse the SSM and MLP blocks as done in Mamba (Gu and Dao, 2023). However, in later experiments we observed that using a separate MLP block as well as a short 1D convolution to process the inputs as in prior work (Poli et al., 2023; Gu and Dao, 2023; De et al., 2024) improved performance on some retrieval tasks when trained with the Birdie procedure. For clarity, we refer to this final model in the results related in Section 4 below as Gated SSM+[4]. We find this simple baseline performs comparably with state-of-the-art SSMs, such as Hawk.

4 Experiments and Results

Here, we present our experimental setup and main findings.

4.1 Experimental Setup

We pre-train and instruction-tune a series of 1.4B parameter SSM and Transformer models to investigate the proposed methods. This size allows us to achieve non-trivial performance on popular public benchmarks while making it feasible to ablate a number of design choices.

Pre-training: We train several 1.4B versions of the Gated SSM baseline described in Section 3.4. We include ablations using the standard CLM objective, UL2, and our novel Birdie training procedure described in Section 3.3, with its bidirectional prefix processing and dynamic mixture selection. We also include two variations for the Gated SSM: a causal-only model and a non-dynamic, fixed ratio mixture we refer to as Birdie - Causal and Birdie - Fixed Ratio respectively. To show our methods are more broadly applicable to other SSMs, we also train Hawk, a state-of-the-art SSM, and a modern Transformer architecture using CLM and the Birdie training procedure[5]. Key results and comparative analysis of these models are discussed in Section 4.3. Additional pre-training details can be found in Appendix A.1.

Instruction Tuning: For all models, we loosely follow the progressive learning fine-tuning procedure from Orca 2 (Mitra et al., 2023) and integrate common instruction-tuning procedures from FLAN (Longpre et al., 2023), Zephyr (Tunstall et al., 2023), and Tulu (Wang et al., 2023). We extend the sequence length to 4096 and 8192. More details on fine-tuning can be found in Sections A.2.

Evaluations: First, we evaluate our models across 21 tasks using the EleutherAI LM Harness (Gao et al., 2023) to test general knowledge and reasoning abilities and ensure that the Birdie training procedure maintains performance here. We describe these tasks further in Appendix A.4. To stress test in-context retrieval abilities of the models, we evaluate on the phone-book lookup task, as well as SQuAD V2 for paragraph Q&A tasks that challenge SSMs (Jelassi et al., 2024). We also introduce a new infilling dataset to test the models' abilities to comprehend the context of a story.

4.2 Comparative Performance and Ablation Study on Max-likelihood Tasks

We show the detailed performance of base and instruction-tuned models using the LM Eval harness in the Appendix Table 5. In Table 2 we relate the average accuracy over all 21 tasks. The results show that the models trained with the Birdie procedure perform comparably to models trained with the standard CLM objective, especially after instruction-tuning. This shows that the Birdie training procedure does not harm the general short context knowledge and reasoning abilities these benchmarks test.

4.3 Analysis on Phone Number Retrieval

Next, we explore a phone number retrieval task, previously identified as challenging for SSMs (Je-

[4]An implementation is available in Appendix E.2 and https://github.com/samblouir/birdie.

[5]The Transformer-Birdie variant uses unmasked attention on the prefix, equivalent to the prefix-LM architecture described in (Raffel et al., 2020)

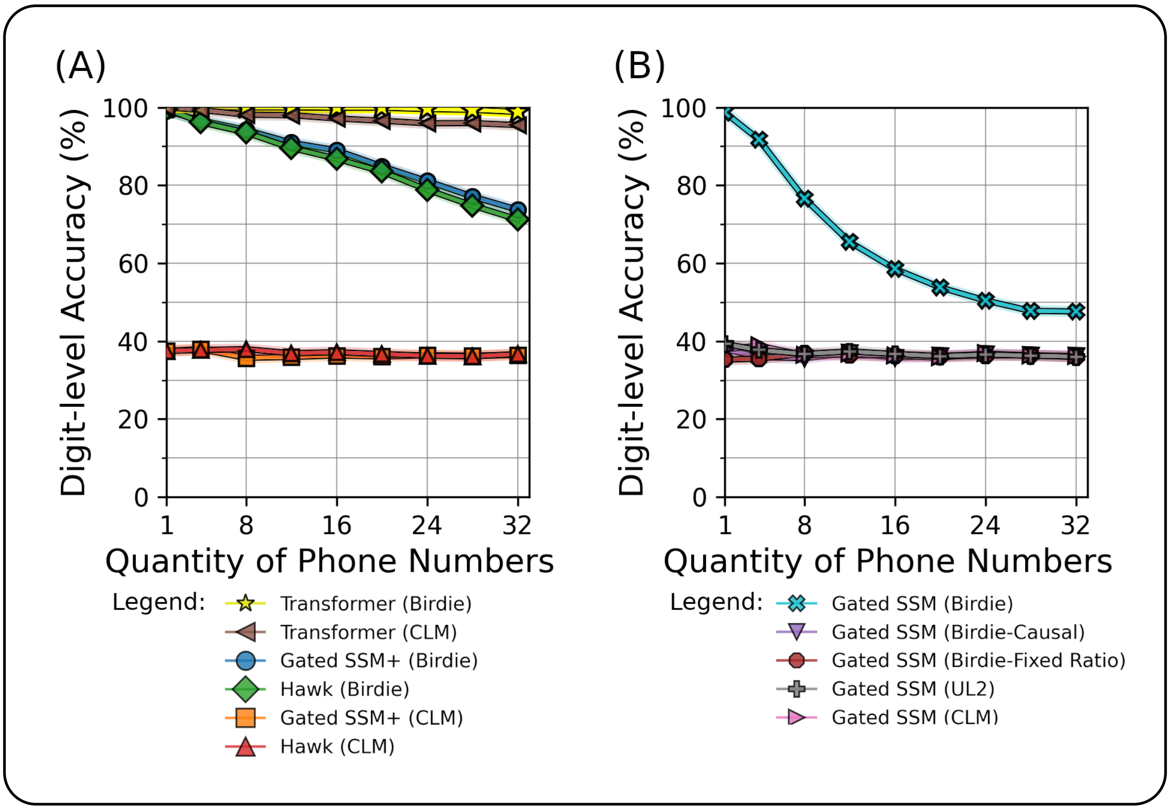

Figure 1: Phone number retrieval task performance, using a 16,384 sequence length. SSMs trained with Birdie significantly close the gap with Transformers. (A) Comparison between Gated SSM+, Hawk, and Transformer trained with CLM and Birdie. The Birdie procedure significantly improves SSM performance. (B): Controlled ablation using various pre-training approaches for Gated SSM. More details are available in Appendix Section D.

Model	Training Procedure	Accuracy (%)
	Instruct Models	
Gated SSM+	Birdie	42.0
Hawk	Birdie	41.5
Transformer	CLM	40.9
Hawk	CLM	40.9
Transformer	Birdie	40.2
Gated SSM+	CLM	40.0
	Base Models	
Hawk	CLM	40.5
Transformer	CLM	40.4
Gated SSM+	CLM	39.5
Transformer	Birdie	39.1
Hawk	Birdie	38.7
Gated SSM+	Birdie	38.5

Table 2: Average accuracy (%) across 21 EleutherAI tasks, including ARC, MMLU, and LogiQA. The Birdie procedure performs comparably to CLM on these tasks. More ablations are related in detail in Appendix 5.

lassi et al., 2024; De et al., 2024). We introduce a more complex variant by expanding the phone book from 200 to approximately 800 entries that can require retrieving up to 32 numbers simultaneously. All models were fine-tuned from their base configurations for 250 steps, adapting to entry counts ranging from 8 to 800. This mild fine-tuning primarily aimed to extend positional encodings and accommodate the models to new lengths. For additional details, please refer to Appendix D.

Ablations We evaluate variations of the Birdie training procedure using the basic Gated SSM model on the phone book task, related in Figure 1B. The full Birdie procedure significantly enhances performance across all tasks. Notably, Birdie-Causal's minimal improvement over the CLM baseline highlights the role of bidirectional processing. However, UL2's minor performance boost over CLM suggests that bidirectional processing alone does not account for the gains. Similarly, Birdie-Fixed Ratio's lack of improvement provides evidence of the importance of Birdie's dynamic mixtures. These trends hold across other tasks (see the infilling task in Appendix C or SQuAD in Figure 2). We initially observed that Hawk trained with Birdie outperformed Gated SSM on the phone book task. This led us to develop and focus on the Gated-SSM+ for subsequent results.

General Results Figure 1A compares the Transformer, Hawk, and Gated SSM+ models, trained using either Birdie or the CLM objective on the phone book task. This task is easy for the Transformers, which achieve high performance regardless of the number of phone numbers retrieved. However, we observe training the Transformer with the Birdie procedure leads to a slight boost in performance. We also observe that the Hawk and Gated SSM+ baselines trained with the CLM objective perform poorly, even when asked to retrieve only a single phone number. In contrast, we see that both the Hawk and Gated SSM+ models trained with the Birdie procedure significantly reduce the performance gap with the Transformer baselines. While the performance of the SSMs degrades as the task complexity increases (e.g., increasing the number of phone numbers), the Birdie procedure significantly extends the regime in which the SSMs can perform the retrieval.

4.4 Question-Answering

We evaluate our models on the SQuAD-V2 Q&A task (Rajpurkar et al., 2018). Using greedy decoding (max 64 tokens) on answerable questions, we format inputs as "contextquestion" without few-shot examples. We measure "Answer Contains Label" and F1 score. Figure 2 shows our results, and further ablations and details are available in Appendix Section B. CLM-trained SSMs' performance strongly degrades with increasing context length, as noted by Jelassi et al. (2024). However, Birdie-trained SSMs maintain performance comparable to Transformers across all available sequence lengths.

4.5 Infilling Results

Finally, we introduce a new infilling task to assess models' capabilities in copying, retrieval, and context comprehension. Models are presented with a story containing 3-7 causal entries, one of which is blank. Models predict the most appropriate option to fill this blank. As with other tasks, we observe that the Birdie procedure allows the SSM models to perform more closely to the Transformer baselines. Table 3 relates the main results. More results and details can be found in Appendix C.

Model	Training Procedure	Accuracy
Instruct Models		
Gated SSM+	Birdie	42.5%
Transformer	Birdie	42.0%
Transformer	CLM	42.0%
Hawk	Birdie	40.4%
Hawk	CLM	34.0%
Gated SSM+	CLM	32.8%
Base Models		
Transformer	CLM	40.4%
Hawk	Birdie	39.7%
Transformer	Birdie	39.7%
Gated SSM+	Birdie	36.6%
Hawk	CLM	29.6%
Gated SSM+	CLM	29.5%

Table 3: Average accuracy on the new infilling dataset, where models complete story segments. Birdie-trained SSMs surpass CLM-trained SSMs. For data samples and more, please see Appendix section C.

5 Conclusion

In this work, we investigated the significant impact of the training procedure on the downstream capabilities of State Space Models (SSMs). While prior research highlighted major weaknesses of SSMs on in-context retrieval tasks, we demonstrated that refining the training process can enhance their performance in these areas. Specifically, we proposed a novel combination of bidirectional processing of the prefix with mixtures of specialized pre-training objectives designed to improve infilling, copying, and handling of long-range dependencies. Additionally, we introduced an RL-based dynamic sampling procedure that adaptively selects optimal objective mixtures throughout training. As a result, the Birdie training procedure strongly improves a model's ability to tackle retrieval-heavy tasks where previous SSM methods have struggled. This finding suggests that, despite the simplicity of the popular CLM objective, this objective may not align optimally with the inductive biases inherent in SSM architectures.

Our work posits that SSMs can achieve enhanced performance through careful selection and design of training objectives, offering a novel pathway for improvement beyond architectural modifications. By showcasing substantial performance gains achievable through this approach, we advocate for a broader reconsideration of how SSMs are developed and optimized. The introduction of Birdie exemplifies the benefits this methodology can bring, pointing toward new directions for future research. We hope that our findings will inspire further exploration of pre-training objectives as a critical factor in advancing SSMs and their application to complex NLP challenges.

6 Limitations

Our experiments were limited by an academic budget. While the 1.4B models we trained and studied are large enough to provide stronger confidence

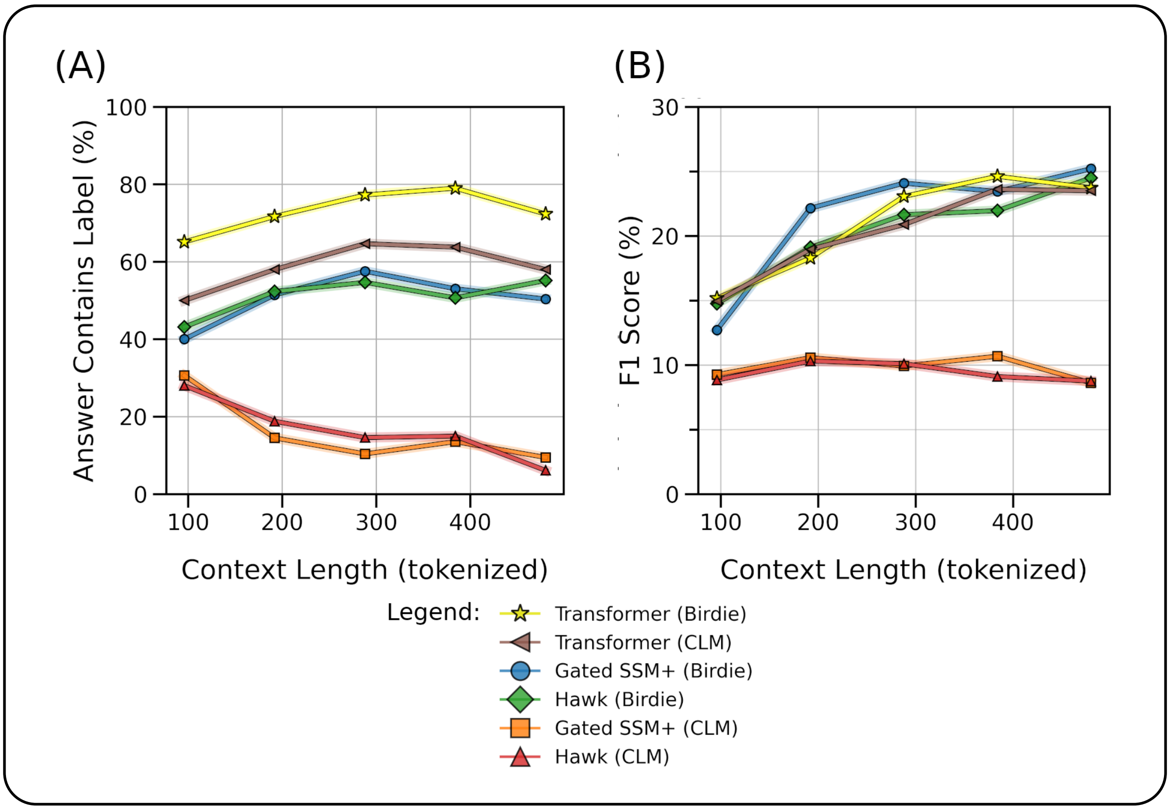

Figure 2: SQuAD-V2 results with instruction-tuned models. Training with the Birdie procedure strongly improves SSM performance, compared to CLM. Further ablations and details are available in Appendix Section B.

compared to studies that can only analyze models with less than 1B parameters, there is a lingering question of how results scale with larger models and more data.

It is hard to beat the simplicity of the CLM objective. The training setup required for the mixture of objectives approach requires more care to implement correctly.

The availability of long context evaluations of LLMs is challenging. It is often difficult to find tasks that separate out the use of parametric knowledge from true in-context reasoning abilities (Hsieh et al., 2024). This can be particularly true in tasks using realistic data, since the knowledge required to solve the task may have been present in the training data. It is possible our long paragraph question-answering and infilling tasks slighly suffer from this issue. On the other hand, synthetic tasks, such as the phone book retrieval task, can make it easier to ensure the task requires true in-context reasoning, albeit it is easy to question the usefulness and applicability of such synthetic tasks. Continued innovation in long context evaluations is crucial to stronger long context abilities in language models (agnostic of architecture).

In our experiments we observe that the performance of SSMs on retrieval tasks still starts to degrade more quickly than the Transformer baselines. We do not claim to have completely solved the retrieval problem, and there may well be other weaknesses of SSMs that are not captured in the tasks considered in our work.

Acknowledgments

This work was supported in part by the National Science Foundation Grant No. 2310113 to Amarda Shehu. Antonios Anastasopoulos is also supported by the NSF under award IIS-2327143. Computations were run on Hopper, a research computing cluster provided by the Office of Research Computing at George Mason University, VA (URL: http://orc.gmu.edu) and on Cloud TPUs provided by Google's TPU Research Cloud (TRC) program. We thank the anonymous reviewers for their constructive feedback.

References

Aida Amini, Saadia Gabriel, Shanchuan Lin, Rik Koncel-Kedziorski, Yejin Choi, and Hannaneh Hajishirzi. 2019. MathQA: Towards interpretable math word problem solving with operation-based formalisms. 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 2357–2367, Minneapolis, Minnesota. Association for Computational Linguistics.

Rohan Anil, Andrew M. Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, et al. 2023. Palm 2 technical report. *Preprint*, arXiv:2305.10403.

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

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Re. 2024. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-first International Conference on Machine Learning*.

Dzmitry Bahdanau. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

David Balduzzi and Muhammad Ghifary. 2016. Strongly-typed recurrent neural networks. *Preprint*, arXiv:1602.02218.

Mohammad Bavarian, Heewoo Jun, Nikolas Tezak, John Schulman, Christine McLeavey, Jerry Tworek, and Mark Chen. 2022. Efficient training of language models to fill in the middle. *Preprint*, arXiv:2207.14255.

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

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. 2023. Pythia: A suite for analyzing large language models across training and scaling. In *International Conference on Machine Learning*, pages 2397–2430. PMLR.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. 2020. Piqa: Reasoning about physical commonsense in natural language. In *Thirty-Fourth AAAI Conference on Artificial Intelligence*.

Guy E. Blelloch. 1990. Prefix sums and their applications. Technical Report CMU-CS-90-190, School of Computer Science, Carnegie Mellon University.

James Bradbury, Stephen Merity, Caiming Xiong, and Richard Socher. 2016. Quasi-recurrent neural networks. *Preprint*, arXiv:1611.01576.

Xi Chen, Josip Djolonga, Piotr Padlewski, Basil Mustafa, Soravit Changpinyo, Jialin Wu, Carlos Riquelme Ruiz, Sebastian Goodman, Xiao Wang, Yi Tay, Siamak Shakeri, Mostafa Dehghani, Daniel M. Salz, Mario Lucic, Michael Tschannen, Arsha Nagrani, Hexiang Hu, Mandar Joshi, Bo Pang, Ceslee Montgomery, Paulina Pietrzyk, Marvin Ritter, A. J. Piergiovanni, Matthias Minderer, Filip Pavetic, Austin Waters, Gang Li, Ibrahim M. Alabdulmohsin, Lucas Beyer, Julien Amelot, Kenton Lee, Andreas Steiner, Yang Li, Daniel Keysers, Anurag Arnab, Yuanzhong Xu, Keran Rong, Alexander Kolesnikov, Mojtaba Seyedhosseini, Anelia Angelova, Xiaohua Zhai, Neil Houlsby, and Radu Soricut. 2023. Pali-x: On scaling up a multilingual vision and language model. *ArXiv*, abs/2305.18565.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, et al. 2022. Palm: Scaling language modeling with pathways. *Preprint*, arXiv:2204.02311.

Christopher Clark, Kenton Lee, Ming-Wei Chang, Tom Kwiatkowski, Michael Collins, and Kristina Toutanova. 2019. 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, Minneapolis, Minnesota. Association for Computational Linguistics.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George Cristian-Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando De Freitas, and Caglar Gulcehre. 2024. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *Preprint*, arXiv:2402.19427.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. *Preprint*, arXiv:1810.04805.

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

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, et al. 2023a. Hungry hungry hippos: Towards language modeling with state space models. In *ICLR*.

Daniel Y. Fu, Hermann Kumbong, Eric Nguyen, and Christopher Ré. 2023b. Flashfftconv: Efficient convolutions for long sequences with tensor cores. *Preprint*, arXiv:2311.05908.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, Shawn Presser, and Connor Leahy. 2020. The pile: An 800gb dataset of diverse text for language modeling. *Preprint*, arXiv:2101.00027.

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. 2023. A framework for few-shot language model evaluation.

Shivam Garg, Dimitris Tsipras, Percy Liang, and Gregory Valiant. 2023. What can transformers learn in-context? a case study of simple function classes. In *NeurIPS*.

Andrew Gordon, Zornitsa Kozareva, and Melissa Roemmele. 2012. SemEval-2012 task 7: Choice of plausible alternatives: An evaluation of commonsense causal reasoning. In **SEM 2012: The First Joint Conference on Lexical and Computational Semantics – Volume 1: Proceedings of the main conference and the shared task, and Volume 2: Proceedings of the Sixth International Workshop on Semantic Evaluation (SemEval 2012)*, pages 394–398, Montréal, Canada. Association for Computational Linguistics.

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

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

Mandy Guo, Joshua Ainslie, David Uthus, Santiago Ontanon, Jianmo Ni, Yun-Hsuan Sung, and Yinfei Yang. 2022. Longt5: Efficient text-to-text transformer for long sequences. *Preprint*, arXiv:2112.07916.

Ramin Hasani, Mathias Lechner, Tsun-Hsuan Wang, Makram Chahine, Alexander Amini, and Daniela Rus. 2022. Liquid structural state-space models. *Preprint*, arXiv:2209.12951.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Sepp Hochreiter, J urgen Schmidhuber, and Corso Elvezia. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, et al. 2022. Training compute-optimal large language models. In *NeurIPS*.

Cheng-Ping Hsieh, Simeng Sun, Samuel Kriman, Shantanu Acharya, Dima Rekesh, Fei Jia, and Boris Ginsburg. 2024. Ruler: What's the real context size of your long-context language models? *arXiv preprint arXiv:2404.06654*.

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

Qiao Jin, Bhuwan Dhingra, Zhengping Liu, William W. Cohen, and Xinghua Lu. 2019. Pubmedqa: A dataset for biomedical research question answering. *Preprint*, arXiv:1909.06146.

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

Sachin Kumar, Antonios Anastasopoulos, Shuly Wintner, and Yulia Tsvetkov. 2021. Machine translation into low-resource language varieties. In *ACL-IJNLP*, Online. Association for Computational Linguistics.

Tao Lei, Yu Zhang, Sida I. Wang, Hui Dai, and Yoav Artzi. 2018. Simple recurrent units for highly parallelizable recurrence. *Preprint*, arXiv:1709.02755.

Hector J. Levesque, Ernest Davis, and Leora Morgenstern. 2012. The winograd schema challenge. In *13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012*, Proceedings of the International Conference on Knowledge Representation and Reasoning, pages 552–561. Institute of Electrical and Electronics Engineers Inc. 13th International Conference on the Principles of Knowledge Representation and Reasoning, KR 2012 ; Conference date: 10-06-2012 Through 14-06-2012.

Wing Lian, Guan Wang, Bleys Goodson, Eugene Pentland, Austin Cook, Chanvichet Vong, "Teknium", and Nathan Hoos. 2023. Slimorca dedup: A deduplicated subset of slimorca.

Jian Liu, Leyang Cui, Hanmeng Liu, Dandan Huang, Yile Wang, and Yue Zhang. 2020. Logiqa: A challenge dataset for machine reading comprehension with logical reasoning. *Preprint*, arXiv:2007.08124.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning. *Preprint*, arXiv:2301.13688.

Eric Martin and Chris Cundy. 2018. Parallelizing linear recurrent neural nets over sequence length. *Preprint*, arXiv:1709.04057.

Harsh Mehta, Ankit Gupta, Ashok Cutkosky, and Behnam Neyshabur. 2023. Long range language modeling via gated state spaces. In *The Eleventh International Conference on Learning Representations*.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *EMNLP*.

Arindam Mitra, Luciano Del Corro, Shweti Mahajan, Andres Codas, Clarisse Simoes, Sahaj Agarwal, Xuxi Chen, Anastasia Razdaibiedina, Erik Jones, Kriti Aggarwal, Hamid Palangi, Guoqing Zheng, Corby Rosset, Hamed Khanpour, and Ahmed Awadallah. 2023. Orca 2: Teaching small language models how to reason. *Preprint*, arXiv:2311.11045.

Catherine Olsson, Nelson Elhage, Neel Nanda, Nicholas Joseph, Nova DasSarma, Tom Henighan, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Scott Johnston, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. 2022. In-context learning and induction heads. *Preprint*, arXiv:2209.11895.

Antonio Orvieto, Samuel L Smith, Albert Gu, Anushan Fernando, Caglar Gulcehre, Razvan Pascanu, and Soham De. 2023. Resurrecting recurrent neural networks for long sequences. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org.

Ankit Pal, Logesh Kumar Umapathi, and Malaikannan Sankarasubbu. 2022. Medmcqa : A large-scale multi-subject multi-choice dataset for medical domain question answering. *Preprint*, arXiv:2203.14371.

Jongho Park, Jaeseung Park, Zheyang Xiong, Nayoung Lee, Jaewoong Cho, Samet Oymak, Kangwook Lee, and Dimitris Papailiopoulos. 2024. Can mamba learn how to learn? a comparative study on in-context learning tasks. In *Forty-first International Conference on Machine Learning*.

Anselmo Peñas, Eduard H. Hovy, Pamela Forner, Álvaro Rodrigo, Richard F. E. Sutcliffe, and Roser Morante. 2013. Qa4mre 2011-2013: Overview of question answering for machine reading evaluation. In *CLEF*.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Huanqi Cao, Xin Cheng, Michael Chung, Matteo Grella, Kranthi Kiran GV, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartlomiej Koptyra, Hayden Lau, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Xiangru Tang, Bolun Wang, Johan S. Wind, Stansilaw Wozniak, Ruichong Zhang, Zhenyuan Zhang, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. 2023. Rwkv: Reinventing rnns for the transformer era. *Preprint*, arXiv:2305.13048.

Mohammad Taher Pilehvar and José Camacho-Collados. 2018. Wic: 10, 000 example pairs for evaluating context-sensitive representations. *CoRR*, abs/1808.09121.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y. Fu, Tri Dao, et al. 2023. Hyena hierarchy: Towards larger convolutional language models. *Preprint*, arXiv:2302.10866.

Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Re, et al. 2024. Mechanistic design and scaling of hybrid architectures. In *Forty-first International Conference on Machine Learning*.

Zhen Qin, Songlin Yang, and Yiran Zhong. 2023. Hierarchically gated recurrent neural network for sequence modeling. *Preprint*, arXiv:2311.04823.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 784–789.

Machel Reid, Nikolay Savinov, Denis Teplyashin, Dmitry Lepikhin, Timothy Lillicrap, Jean-baptiste Alayrac, Radu Soricut, Angeliki Lazaridou, Orhan Firat, Julian Schrittwieser, et al. 2024. Gemini 1.5: Unlocking multimodal understanding across millions of tokens of context. *arXiv preprint arXiv:2403.05530*.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. 2019. WINOGRANDE: an adversarial winograd schema challenge at scale. *CoRR*, abs/1907.10641.

Jimmy T. H. Smith, Andrew Warrington, and Scott W. Linderman. 2023. Simplified state space layers for sequence modeling. In *ICLR*.

Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Ng, and Christopher Potts. 2013. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages

1631–1642, Seattle, Washington, USA. Association for Computational Linguistics.

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

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, et al. 2023. Ul2: Unifying language learning paradigms. In *ICLR*.

Yi Tay, Jason Wei, Hyung Won Chung, Vinh Q. Tran, David R. So, Siamak Shakeri, Xavier García, Huaixiu Steven Zheng, Jinfeng Rao, Aakanksha Chowdhery, Denny Zhou, Donald Metzler, Slav Petrov, Neil Houlsby, Quoc V. Le, and Mostafa Dehghani. 2022. Transcending scaling laws with 0.1% extra compute. In *Conference on Empirical Methods in Natural Language Processing*.

Lewis Tunstall, Edward Beeching, Nathan Lambert, Nazneen Rajani, Kashif Rasul, Younes Belkada, Shengyi Huang, Leandro von Werra, Clémentine Fourrier, Nathan Habib, Nathan Sarrazin, Omar Sanseviero, Alexander M. Rush, and Thomas Wolf. 2023. Zephyr: Direct distillation of lm alignment. *Preprint*, arXiv:2310.16944.

Iñigo Urteaga, Moulay-Zaïdane Draïdia, Tomer Lancewicki, and Shahram Khadivi. 2023. Multi-armed bandits for resource efficient, online optimization of language model pre-training: the use case of dynamic masking. *Preprint*, arXiv:2203.13151.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Preprint*, arXiv:1706.03762.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Yizhong Wang, Hamish Ivison, Pradeep Dasigi, Jack Hessel, Tushar Khot, Khyathi Raghavi Chandu, David Wadden, Kelsey MacMillan, Noah A. Smith, Iz Beltagy, and Hannaneh Hajishirzi. 2023. How far can camels go? exploring the state of instruction tuning on open resources. *Preprint*, arXiv:2306.04751.

Johannes Welbl, Nelson F. Liu, and Matt Gardner. 2017. Crowdsourcing multiple choice science questions. In *Proceedings of the 3rd Workshop on Noisy User-generated Text*, pages 94–106, Copenhagen, Denmark. Association for Computational Linguistics.

Kaiyue Wen, Xingyu Dang, and Kaifeng Lyu. 2024. Rnns are not transformers (yet): The key bottleneck on in-context retrieval. *Preprint*, arXiv:2402.18510.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.

Tianbao Xie, Chen Henry Wu, Peng Shi, Ruiqi Zhong, Torsten Scholak, et al. 2022. Unifiedskg: Unifying and multi-tasking structured knowledge grounding with text-to-text language models. In *EMNLP*.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. 2024. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning*.

Xiang Yue, Ziyu Yao, and Huan Sun. 2022. Synthetic question value estimation for domain adaptation of question answering. In *ACL*.

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

Ben Zhou, Daniel Khashabi, Qiang Ning, and Dan Roth. 2019. "going on a vacation" takes longer than "going for a walk": A study of temporal commonsense understanding. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3363–3369, Hong Kong, China. Association for Computational Linguistics.

A Appendix

A.1 Pretraining

We train all models on the same data pipeline using The Pile.(Gao et al., 2020)[6]. The Pile is a collection of several datasets, and includes books, code, web scrapes, emails, and question-answer instruction formatted examples.

During all training and fine-tuning, we always use sequence packing and proper masking for all models, preventing samples from interfering with each other. For Hawk and Gated SSM+, we add spacing between samples to prevent the Conv1D layer from leaking out information. All models use this spacing to normalize the samples seen during evaluation periods and, therefore, reduce external noise when comparing models trained using Birdie's reinforcement learning setup.

Models trained for 32,000 steps, with a batch size of 520. We train all models on The Pile (Gao et al., 2020) dataset for 32B tokens using sequence packing and proper masking to prevent sample interference. All models were pre-trained with a sequence length of 2048. Following recommendations by Chowdhery et al. (2022), we pre-train slightly over Chinchilla optimal scaling laws (Hoffmann et al., 2022) – 20-25x tokens per parameter. We provide a comparison of compute costs and resources in Table 4. We count both context and target tokens as tokens "seen" by the model. This provides a fair comparison among different pre-training objectives. This diverges from other approaches, which do not always consider context tokens in their total count of tokens on which the model was trained (Tay et al., 2023). This means that the Copying task, for example, results in an actual reduction in the total count of unique training tokens seen by the model. This is because the training budget is for a number of tokens. With copying, the same tokens appear twice: once as an input, and once as a label.

We use the same hyperparameters for all models, using the same settings, such as learning rates and batch sizes, as models found in Mamba (Gu and Dao, 2023). We use the official settings for Hawk - gradient clipping on Beta and no weight decay on RG-LRU layers. Our Transformer baselines use Llama 2 Long's positional encodings.

[6]We use the full version of The Pile, last available mid-2023

A.2 Instruction Tuning

For 1.4B parameter models, we largely follow the progressive learning fine-tuning procedure from Orca 2 (Mitra et al., 2023), as immediately jumping into relatively difficult, small datasets, such as SlimOrca-Dedup (Lian et al., 2023) ended up hurting performance. We follow common instruction-tuning procedures from FLAN (Longpre et al., 2023), Zephyr (Tunstall et al., 2023), and Tulu (Wang et al., 2023) with dropout, cosine decay learning rate, and no weight decay. We use all training, validation, and test sets as provided by the original authors.

We change hyperparameters from FLAN's paper since we use AdamW and not AdaFactor. We need a different learning rate to compensate for the lack of AdaFactor's parameter-scaled updates. We use a gentle 3e-4 peak cosine LR as in Zephyr (Tunstall et al., 2023) over 4 epochs. For FLAN, we extend the sequence length to 4096 (from 2048 during pre-training) and use a batch size of 20. This keeps the number of tokens per batch equal with the original publication.

A.3 Hardware and Experimental Setup

We fully-train 11 models, each containing 1.4 billion parameters. The primary training is conducted on 5 machines, each equipped with 4 Nvidia A100 GPUs (80GB). Additionally, fine-tuning and evaluation was split among Google TPUv3-8 and TPUv4-32 units, generously provided through Google's TPU Research Cloud, for which we are sincerely grateful. The fixed ratios of BFR was found by training small 110M Gated SSM and Transformers models with random mixtures and hand-tuning sampling rates. This took over 50 iterations of training the 110M model, which took roughly 5 hours each.

Table 4 relates compute cost between models for the hardware we used for pre-training.

Backend	**Model**	**GPU Hrs (A100)**	**Sec / Step**	**Seq Length**	**Tokens / sec / A100**
Torch	Birdie	3,200	2.0	N/A	26,148
Torch	Flash Attn. 2	7,011	4.4	2048	12,152
JAX	Gated-SSM	5,600	3.5	N/A	15,214
JAX	Gated-SSM+	6,480	4.05	N/A	13,148
JAX	Hawk	7,680	4.8	N/A	11,093
JAX	Transformer	10,016	6.3	2048	8,506

Table 4: Comparison of observed model training speeds on our multi-node A100 setup.

A.4 The EleutherAI LM Harness Tasks

Task	Description
arc_easy	The 'Easy' portion of a multiple-choice question-answering dataset, containing questions from science exams from grade 3 to 9 (Clark et al., 2018).
arc_challenge	The Challenge portion of the dataset, containing the more difficult questions that require reasoning (Clark et al., 2018).
boolq	A question answering dataset for Yes/No questions containing 15942 examples; each example is a triplet of (question, passage, answer), with the title of the page (from google search engine where questions are collected) as optional additional context (Clark et al., 2019).
copa	The Choice Of Plausible Alternatives (COPA) dataset consists of 1000 questions composed of a premise and two alternatives, with the task being to select the alternative that more plausibly has a causal relation with the premise (Gordon et al., 2012).
HellaSwag	A dataset designed to test common sense reasoning and grounded situations, presenting contexts from video and text with multiple-choice endings where a model must predict the most likely continuation (Zellers et al., 2019).
logiQA	A question answering dataset derived from logical reasoning examination questions, aimed at evaluating the deep logical reasoning capability of models (Liu et al., 2020).
mathqa	A large-scale dataset of math word problems (Amini et al., 2019).
mc_taco	13K question-answer pairs that require temporal commonsense comprehension on (1) duration of an event, (2) order of events, (3) time when event occurs, (4) event frequency, and (5) stationarity (whether a state is maintained for a very long time or indefinitely). (Zhou et al., 2019)
medmcqa	A large-scale, Multiple-Choice Question Answering (MCQA) dataset designed to address real-world medical entrance exam questions (Pal et al., 2022).
mmlu	The Massive Multitask Language Understanding (MMLU) dataset, consisting of questions spanning multiple subjects and domains, designed to test models on a broad range of knowledge and reasoning skills (Hendrycks et al., 2021).
mnli	Often also referred to as multi-nl, this Multi-Genre Natural Language Inference (MultiNLI) corpus is a dataset to test sentence understanding; it offers data from ten distinct genres of written and spoken English–enabling evaluation on nearly the full complexity of the language and on cross-genre domain adaptation. (Williams et al., 2018)
OpenBookQA	A dataset that consists of 5,957 multiple-choice questions that necessitate the use of both reasoning and additional broad common sense or scientific knowledge not contained in the question itself (Mihaylov et al., 2018).
piqa	The Physical Interaction Question Answering dataset, focusing on reasoning about physical properties of objects and the actions taken upon them (Bisk et al., 2020).
pubmedqa	A Yes/No biomedical question answering dataset collected from PubMed abstracts (Jin et al., 2019).
qa4mre	The Question Answering for Machine Reading Evaluation dataset is designed for the annual competition, consisting of a series of questions based on a single document with multiple-choice answers (Peñas et al., 2013).
qnli	The Question-answering Natural Language Inference dataset is automatically derived from the Stanford Question Answering Dataset v1.1 (SQuAD) of question-paragraph pairs, where one of the sentences in the paragraph (drawn from Wikipedia) contains the answer to the corresponding question (written by an annotator). (Wang et al., 2018).
race	A large-scale reading comprehension dataset collected from English exams, featuring questions with multiple-choice answers that demand high-level reasoning abilities (Lei et al., 2018).
sciq	Crowd-sourced science exam questions about Physics, Chemistry, Biology, etc, in multiple-choice format with 4 answer options and an evidence-supporting paragraph for the correct answer for most questions (Welbl et al., 2017).
sst2	The Stanford Sentiment Treebank, a corpus with fully labeled parse trees for a complete analysis of the compositional effects of sentiment in language (Socher et al., 2013).
wic	A large-scale Word in Context dataset based on annotations curated by experts for generic evaluation of context-sensitive representations (Pilehvar and Camacho-Collados, 2018).
winogrande	A large-scale dataset of 44k problems, inspired by the original Winograd Schema Challenge (WSC) design (Levesque et al., 2012), but adjusted to improve both the scale and the hardness of the dataset (Sakaguchi et al., 2019).

Table 5: Performance on EleutherAI LM Harness Tasks

Instruct Models Model	ARC Challenge and ARC Easy	BoolQ	COPA	HellaSwag	LogiQA	MathQA	MC-TACO	MedMCQA	MMLU	MNLI	OpenBookQA	PIQA	PubMedQA	QA4MRE	QNLI	race	SciQ	SST-2	TruthfulQA	WiC	Winogrande	Average
Gated SSM+ (Birdie)	28.6%	61.5%	43.8%	30.0%	31.6%	26.2%	63.7%	29.8%	22.1%	32.3%	29.4%	62.9%	53.6%	31.4%	57.3%	29.8%	43.6%	74.9%	28.8%	49.8%	51.1%	42.0%
Gated SSM (Birdie)	28.5%	58.7%	47.6%	29.0%	28.6%	25.3%	64.7%	31.7%	21.6%	31.8%	30.4%	60.6%	54.2%	28.9%	53.4%	27.9%	43.9%	82.1%	28.2%	51.3%	50.0%	41.8%
Gated SSM (Birdie - Causal)	29.2%	52.0%	45.8%	28.8%	26.1%	25.7%	65.3%	32.2%	21.7%	31.9%	31.0%	62.2%	52.6%	28.2%	50.9%	29.0%	43.7%	87.5%	30.8%	50.0%	50.2%	41.7%
Hawk (Birdie)	29.8%	62.3%	47.4%	30.6%	30.6%	26.3%	33.8%	31.4%	21.5%	31.8%	31.8%	63.3%	54.8%	30.1%	49.5%	30.0%	48.9%	84.9%	30.2%	50.0%	52.0%	41.5%
Transformer (CLM)	29.8%	50.1%	46.6%	33.8%	32.0%	26.7%	41.0%	25.8%	26.7%	31.8%	30.8%	63.9%	47.8%	32.8%	51.0%	30.2%	42.7%	87.6%	26.2%	50.6%	50.9%	40.9%
Hawk (CLM)	29.6%	47.6%	46.9%	28.2%	28.3%	25.5%	65.0%	28.2%	23.8%	34.0%	31.2%	61.2%	33.8%	28.0%	53.7%	30.3%	46.2%	85.8%	25.5%	52.8%	52.6%	40.9%
Gated SSM + MLP (Birdie)	28.1%	57.4%	43.5%	29.2%	28.6%	25.0%	48.4%	23.4%	23.5%	31.8%	29.8%	62.6%	54.4%	28.7%	50.2%	28.5%	46.1%	82.0%	31.7%	50.2%	51.7%	40.7%
Transformer (Birdie)	28.0%	62.5%	45.7%	31.4%	30.6%	25.7%	36.0%	31.7%	22.6%	31.8%	32.8%	62.2%	54.4%	33.9%	49.8%	29.6%	54.2%	50.9%	29.1%	50.0%	51.9%	40.2%
Gated SSM (UL2)	28.5%	52.3%	45.9%	28.6%	26.3%	25.1%	62.1%	30.5%	21.5%	31.8%	31.0%	61.5%	44.8%	29.4%	48.8%	26.0%	40.8%	79.8%	29.7%	49.2%	50.4%	40.2%
Gated SSM+ (CLM)	29.1%	62.1%	46.9%	28.5%	27.6%	26.5%	62.1%	26.9%	22.9%	33.2%	31.0%	60.9%	52.2%	25.0%	49.9%	27.6%	45.2%	54.9%	27.5%	50.3%	50.7%	40.0%
Gated SSM (Birdie - Fixed Ratio)	29.2%	61.4%	43.3%	29.5%	29.0%	25.2%	42.5%	32.0%	21.2%	31.8%	30.2%	62.3%	23.2%	25.9%	53.2%	29.4%	42.0%	77.6%	29.3%	52.4%	52.6%	39.2%
Gated SSM (CLM)	29.0%	61.5%	47.7%	31.2%	27.0%	25.8%	34.1%	28.5%	21.7%	31.8%	30.6%	61.9%	54.6%	27.8%	49.5%	28.5%	43.6%	54.5%	31.9%	50.2%	49.8%	39.1%

Base Models Model	ARC Challenge and ARC Easy	BoolQ	COPA	HellaSwag	LogiQA	MathQA	MC-TACO	MedMCQA	MMLU	MNLI	OpenBookQA	PIQA	PubMedQA	QA4MRE	QNLI	race	SciQ	SST-2	TruthfulQA	WiC	Winogrande	Average
Hawk (CLM)	30.4%	51.7%	51.2%	34.3%	27.3%	26.3%	56.9%	32.0%	21.8%	31.7%	29.8%	62.6%	54.4%	28.7%	50.3%	28.4%	53.0%	54.2%	26.2%	48.9%	51.1%	40.5%
Transformer (CLM)	30.5%	62.1%	50.4%	40.1%	31.0%	26.4%	33.9%	24.1%	26.5%	31.8%	31.8%	65.5%	55.2%	25.5%	49.4%	30.8%	55.6%	50.0%	25.9%	50.0%	50.7%	40.4%
Gated SSM+ (CLM)	30.0%	55.4%	49.7%	34.6%	26.6%	25.1%	35.1%	29.9%	23.6%	32.0%	30.8%	61.8%	55.2%	29.1%	48.5%	26.7%	54.6%	55.5%	25.1%	49.4%	51.9%	39.5%
Gated SSM (CLM)	29.8%	62.0%	48.5%	35.5%	27.2%	24.3%	34.1%	32.2%	21.2%	31.8%	30.2%	64.6%	52.0%	27.5%	49.3%	29.3%	48.9%	56.9%	25.5%	50.0%	49.6%	39.5%
Gated SSM (Birdie - Fixed Ratio)	28.0%	39.1%	47.3%	29.6%	27.0%	24.0%	66.1%	26.9%	23.1%	31.8%	30.6%	59.6%	53.6%	28.7%	50.6%	27.4%	46.5%	59.2%	27.7%	50.0%	51.0%	39.4%
Transformer (Birdie)	25.9%	44.3%	48.4%	29.7%	33.0%	24.7%	66.1%	28.1%	25.3%	31.8%	22.2%	55.3%	37.2%	30.0%	51.7%	29.3%	62.5%	50.0%	27.7%	48.9%	49.6%	39.1%
Hawk (Birdie)	29.6%	42.3%	48.1%	32.3%	26.3%	24.8%	53.3%	22.8%	28.0%	31.8%	31.0%	62.9%	36.8%	26.4%	50.2%	27.5%	50.1%	58.3%	29.5%	50.9%	49.8%	38.7%
Gated SSM+ (Birdie)	29.4%	53.4%	46.2%	31.4%	27.3%	26.0%	39.0%	20.1%	23.1%	31.8%	31.0%	62.1%	55.2%	28.4%	50.6%	25.5%	46.9%	51.7%	27.7%	50.2%	51.1%	38.5%
Gated SSM (UL2)	28.8%	43.9%	44.2%	28.8%	25.5%	24.5%	53.3%	30.0%	22.9%	31.8%	32.2%	61.2%	40.8%	25.5%	49.8%	27.3%	49.9%	58.7%	27.1%	50.8%	50.8%	38.5%
Gated SSM (Birdie - Causal)	26.3%	49.9%	43.7%	28.5%	25.8%	24.4%	47.2%	28.8%	21.9%	32.1%	31.8%	61.2%	50.0%	24.1%	50.4%	24.6%	41.4%	53.7%	31.5%	50.0%	51.7%	38.0%
Gated SSM (Birdie)	27.6%	38.8%	46.5%	28.6%	26.1%	24.3%	62.8%	21.6%	23.5%	31.8%	30.4%	60.4%	34.4%	26.1%	52.1%	25.1%	42.1%	57.9%	29.7%	50.0%	50.1%	37.6%
Gated SSM + MLP (Birdie)	28.3%	45.6%	45.0%	29.0%	28.9%	25.2%	45.0%	20.0%	22.5%	31.8%	30.2%	61.6%	36.4%	25.0%	49.7%	25.6%	40.7%	59.7%	25.7%	50.0%	49.3%	36.9%

A.5 Birdie Pretraining Metrics

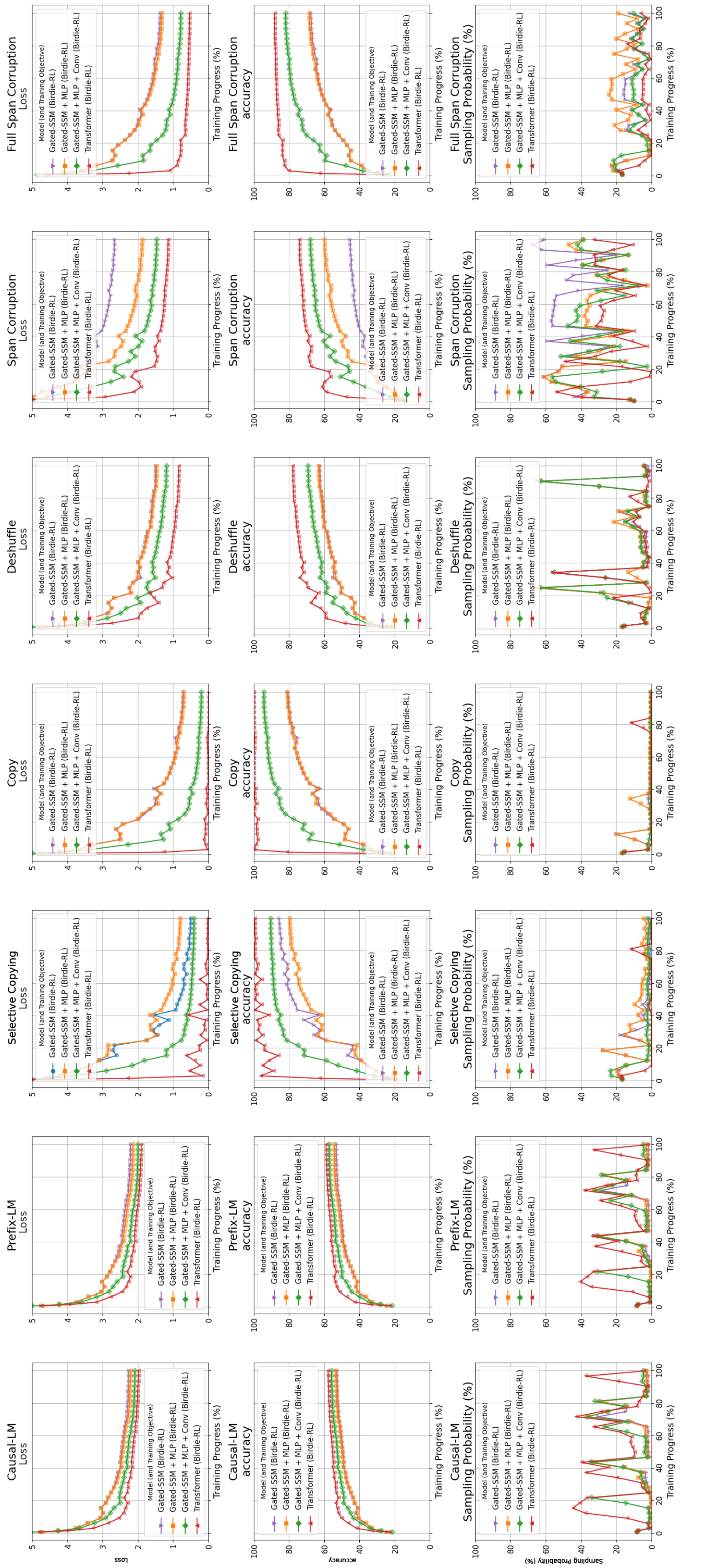

Figure 3: These plots show how Birdie's RL adjusts the pre-training objective mixtures in Gated SSM, building up to Gated SSM+ by adding an MLP, as well as a 1D Convolution layer. Objectives are arbitrarily grouped and averaged together.

B SQuAD V2

Task Description and Setup We evaluate our instruction-tuned models on SQuAD V2, a question-answering dataset. In SQuAD V2, models are given a Wikipedia excerpt and asked a question. Some questions have several acceptable labels, while others are purposefully unanswerable. Following prior work (Jelassi et al., 2024), we focus only on answerable questions. We do not fine-tune our models on this task.

While the standard SQuAD V2 metric (F1) penalizes models for generating additional words, our models are not trained for brevity. Since SQuAD predates modern conversational language models, we prioritize the "Answer Contains Label" metric. This metric awards full credit if any acceptable answer is present in the generated response, while the F1 score awards partial credit for word matches but penalizes verbosity.

Model Tag	Training Procedure	F1 (%)	Answer Contains Label (%)
Gated SSM	**Birdie**	**17.0**	**31.3**
Gated SSM	UL2	12.8	18.6
Gated SSM	Birdie - Fixed Ratio	11.3	18.5
Gated SSM	Birdie - Causal	11.3	15.0
Gated SSM	CLM	10.3	14.7

Table 6: Averaged SQuAD V2 results with instruction-tuned Gated SSM models. Training with the Birdie procedure strongly improves SSM performance compared to other training procedures. The best performing model and metrics are shown in bold.

Model Tag	Training Procedure	F1 (%)	Answer Contains Label (%)
Transformer	**Birdie**	21.4	**73.7**
Transformer	CLM	21.0	60.9
Gated SSM+	**Birdie**	**23.2**	54.4
Hawk	Birdie	20.9	52.6
Hawk	CLM	10.1	16.1
Gated SSM+	CLM	9.1	15.7

Table 7: Averaged SQuAD V2 results with instruction-tuned models. Training with the Birdie procedure strongly improves SSM performance, compared to CLM. The best performing models and metrics are shown in bold. These results are plotted by sequence length in Figure 2

.

C Story Infilling Task

Task Description We generate thousands of stories with blank sections using Mistral v0.1 Instruct (7B) with an unusually high temperature of 10.0 and use a min_p of 0.10 to keep text coherent. At the same time, we have the model generate four potential choices to fill in that story, with one of them being the intended best choice. Generally, the choices to fill in the stories are plausible. The model tends to generate at least one adversarial option that is very close to being the best answer, but is also not the best choice.

We filter questions using a Jaccard similarity of 0.85, so when at least two stories share at least 15% of their words, only one is kept and the rest are removed. Finally, we present each story and its choices to four language models, and ask if the intended label is truly the best choice. We remove questions that do not receive a majority vote from four language models. Specifically, these are the instruct versions of Mistral Nemo 2407 (12B), Gemma-2 (9B), Llama 3.1 (8B), and Mistral v0.3 (7B).

Model	Training Procedure	Accuracy
Instruct Models		
Gated SSM	Birdie	36.8%
Gated SSM	Birdie - Fixed Ratio (BFR)	36.2%
Gated SSM	UL2	34.7%
Gated SSM	Birdie - Causal	33.9%
Gated SSM	CLM	32.2%
Base Models		
Gated SSM	Birdie	36.8%
Gated SSM	Birdie - Causal	34.7%
Gated SSM	UL2	31.7%
Gated SSM	Birdie - Fixed Ratio (BFR)	29.6%
Gated SSM	CLM	27.5%

Table 8: Average accuracy over our new infilling dataset. Models fill in a missing part of a story by selecting the best possible option. Losses are normalized by target token length.

Dataset Example Below, we provide an example of a shorter entry and a longer entry from our new infilling dataset.

Short Entry:

```
Consider the following sequence of events, then select a choice that best fills
 in the missing entry:
1. A stranger hands a letter to Ellie on a rainy afternoon.
2. (blank)
3. As she gets closer to the island, the edges of the map feel warm.

Choices:
(A) The letter contains information about a secret meeting happening at the end
 of the week.
(B) She ignores the letter and throws it away.
(C) Ellie finds a hidden treasure map in the envelope.
(D) The letter leads her to an uncharted island.

Which choice best fills in the missing entry?
```

Label:

```
(D) The letter leads her to an uncharted island.
```

Figure 4: A short example from our new infilling task.

Long Entry:

```
        Consider the following sequence of events, then select a choice that
best fills in the missing entry:
        1. A young woman named Mia had a passion for baking. She enjoyed trying
 out new recipes and experimenting with different flavors. One day, as she was
perusing through a cookbook, she came across a recipe for a unique chocolate
cake that sounded both delicious and challenging to make. Determined to create
this masterpiece, Mia gathered all the necessary ingredients and began the
process.
        2. (blank)
        3. She added more flour to thicken the mixture and waited patiently for
 the result. When she took a small spoonful of the new mixture, it had finally
reached a consistency that resembled cake batter. Relieved, Mia continued with
her baking process, pouring the mixture into a round pan and placing it in the
oven.
        4. The aroma of freshly baked chocolate cake filled Mia's home as she
waited for the timer to go off. When the cake was finished, she carefully
removed it from the pan and placed it on a cooling rack. Once it had cooled
down enough to eat, Mia took a bite and smiled with satisfaction. Her
experimentation had paid off; she had created a delectable chocolate cake that
tasted as good as it smelled.
        5. Proud of her achievement, Mia shared the cake with her family. They
all raved about how moist and flavorful the cake was, with no one guessing the
troubles she had gone through to perfect the recipe. From that day on, this new
 chocolate cake recipe became a staple in Mia's kitchen, something that both
delighted her family and showcased her unwavering determination to succeed in
all things baking.

        Choices:
        (A) The chocolate cake mixture seemed too watery, so Mia added an
additional ingredient.
        (B) Mia decided that she did not need to adjust the recipe and
proceeded with it as written.
        (C) Mia gave up on her goal of creating the perfect chocolate cake.
        (D) Mia added more flour to thicken the mixture.

        Which choice best fills in the missing entry?
```

Label:

```
        (A) The chocolate cake mixture seemed too watery, so Mia added an
additional ingredient.
```

Figure 5: A long example from our new infilling task.

D Phone Number Task

Hyperparameters Models are fine-tuned for 250 steps using a learning rate of 5e-5, no weight decay, and a batch size of 32 for 250 steps. We find a batch size of 64 and just 100 steps brings similar results, but the Gated SSM had difficulty with this. Training samples range from 8 to 800 entries and from 1-32 phone numbers to retrieve. Ideally, this allows for our models to handle any phone book example given in this range. We use sequence packing to concatenate shorter training examples out to 16384 tokens. Sequences are packed. Evaluations are done using a sequence length of 16384. Since names vary in length, our implementations tries to get close to 800 total entries.

Inputs:

```
What are the phone numbers for Keven Meador, Stacey Krohn, Aubrey Wrenn, Eva
    Jurkovic, Gloria Job, Lamont Wilson, Emerald Hyman, Ali Hunsberger, Karsyn
    Jankowski, Alec Vinyard, Cole Pattison, Noe Pacheco, Trent Adamo, Greggory
    Chudnovsky, Yandel Funderburk, Scot Mitterer, Matthew Zeigler, Delvin Lerdal,
    Ellen Hickerson, Violet Lightbody, Ashlynn Buckingham, Pranav Blaisdell,
    Sheridan Lorentz, Levar Sharpe, Ramiro Vanlandingham, Yahir Leavitt, Cassius
    Mcguigan, Lillie Jetmore, Beatriz Jobe, Jamison Arruda, John Lovett, and Wade
    Anger? Find them in the phonebook below.

Phonebook:
Leonardo Rampone: 669-174-4914
Porter Wendell: 243-610-6940
Nicolle Journell: 612-425-4786
Tremayne Wcislo: 811-843-0927
[[~12 pages worth of phone entries go here]]
Elbert Foglesong: 345-541-6086
Matthew Zeigler: 417-648-0710
Patricia Queener: 174-489-9656
Kathryn Enrile: 472-553-8622

What are the phone numbers for Keven Meador, Stacey Krohn, Aubrey Wrenn, Eva
    Jurkovic, Gloria Job, Lamont Wilson, Emerald Hyman, Ali Hunsberger, Karsyn
    Jankowski, Alec Vinyard, Cole Pattison, Noe Pacheco, Trent Adamo, Greggory
    Chudnovsky, Yandel Funderburk, Scot Mitterer, Matthew Zeigler, Delvin Lerdal,
    Ellen Hickerson, Violet Lightbody, Ashlynn Buckingham, Pranav Blaisdell,
    Sheridan Lorentz, Levar Sharpe, Ramiro Vanlandingham, Yahir Leavitt, Cassius
    Mcguigan, Lillie Jetmore, Beatriz Jobe, Jamison Arruda, John Lovett, and Wade
    Anger? Find them in the phonebook above.
```

Labels:

```
337-743-1822, 487-090-9300, 261-549-5474, 239-751-7415, 899-328-4576, 500-199-0084,
     744-974-9713, 617-979-7448, 132-114-9918, 807-843-6708, 200-177-4367,
    800-256-6603, 276-090-4864, 174-449-8065, 107-912-1144, 367-994-8279,
    417-648-0710, 130-012-0838, 668-436-3798, 951-625-4252, 734-538-6288,
    952-422-8127, 209-140-8566, 252-088-9435, 956-578-5675, 355-111-4554,
    779-940-5640, 235-150-3054, 312-638-2822, 400-177-6943, 896-686-1785,
    330-123-2864
```

Figure 6: An abbreviated example of a 32 phone number retrieval sample with a 16,384 token length.

E Code

E.1 Bidirectional Example

This code is available on the Github page. URL: https://github.com/samblouir/birdie.

```
Prefix-LM example:


This enables bidirectionality on the inputs/
    context, and enforces causality on the
    labels.

Assuming sequence packing, it is compute-
    matched with a causal scan operation.


(i.e.: reverse_Lambda_elements = np.where(
    reset_mask == 2, 0.0,
    reverse_Lambda_elements))


Example:

Original inputs: [4, 5, 6]

Original labels: [7, 8, 9]

# The inputs.
# 1 acts as the "begin generating" token.
Processed inputs: [4, 5, 6, 1, 7, 8]

# The labels.
Processed labels: [-, -, -, 7, 8, 9]

# Marks which tokens to use for the loss
Processed loss_mask: [0, 0, 0, 1, 1, 1]

# Locations with "2" mark where
# to block state information flow
#  from the right/reverse-direction
Processed reset_mask: [0, 0, 0, 2, 2, 2]

# Marks the bidirectional tokens
# (aka encoder area) for Attention.
Processed attn_mask: [1, 1, 1, 0, 0, 0]


Here is a transposed view of the processed
    data:

idx, input, label, loss_m, attn_m, reset_mask

0, 4, 0, 0, 1, 0,

1, 5, 0, 0, 1, 0,

2, 6, 0, 0, 1, 0,

3, 1, 7, 1, 0, 2,

4, 7, 8, 1, 0, 2,
```

```
Equivalent abbreviated SSM code:


split_location = (state_size // 2)

Lambda_elements_forward = Lambda_elements
    [..., :split_location]

Lambda_elements_reverse = Lambda_elements
    [..., split_location:]


Bu_elements_forward = Bu_elements[..., :
    split_location]

Bu_elements_reverse = Bu_elements[...,
    split_location:]



h_t_fwd = scan(Lambda_elements_forward,
    Bu_elements_forward)

h_t_rev = scan(Lambda_elements_reverse,
    Bu_elements_reverse, reverse=True)

# Concatenate on the last axis
h_t = concatenate(xs_fwd, xs_rev)
```

E.2 Gated SSM Implementation

```
import jax
import jax.numpy as jnp
from jax.nn import sigmoid, gelu
import flax.linen as nn
from flax.linen import Module, Dense

class GatedLinearRNN(nn.Module):
  state_size: int
  hidden_size: int

  def setup(self):
    self.W_f = Dense(self.state_size)
    self.W_z_gate = Dense(self.state_size)
    self.W_z = Dense(self.state_size)
    self.W_out_gate = Dense(self.state_size)
    self.W_out = Dense(self.hidden-size)
    self.Conv1D = Conv(features=state_size,
        kernel_size=4)


  def __call__(self, x_t):
    out_gate = gelu(self.W_out_gate(x_t))

    x_t = self.Conv1D(x_t)
    f_t = sigmoid(self.W_f(x_t))
    z_t = self.W_z(x_t) * sigmoid(self.
        W_z_gate(x_t))

    h_t = ParallelScan(f_t, z_t)
    y_t = self.W_out(out_gate * h_t)
    return y_t
```

E.3 Hawk Implementation

```
import jax
import jax.numpy as jnp
from jax.nn import sigmoid, softplus
from jax import custom_vjp
import flax.linen as nn
from flax.linen import Module, Dense



""" Hawk is untrainable without aggressive
    gradient clipping (standard gradient
    norm clipping is insufficient).
This custom backwards pass implementation is
    directly from RG-LRU code in the
    RecurrentGemma codebase. """

@custom_vjp
def sqrt_bound_derivative(x, max_gradient):
    """ Computes a square root with a
        gradient clipped at 'max_gradient'.
        """
    return jnp.sqrt(x)

def stable_sqrt_fwd(x, max_gradient):
    return jnp.sqrt(x), (x, max_gradient)

def stable_sqrt_bwd(res, g):
    x, max_gradient = res
    x_clipped = jnp.maximum(x, 1 / (4 *
        max_gradient**2))
    return (g / (2 * jnp.sqrt(x_clipped)),)

sqrt_bound_derivative.defvjp(stable_sqrt_fwd,
     stable_sqrt_bwd)
%%%%%%


class HawkLayer(nn.Module):
    """Hawk Layer: This layer uses a Conv1D
        followed by an RG-LRU layer.

    Attributes:
        forget_base: Base forgetting factor.
        alpha_log_scale: "C" in the RG-LRU
            equation. Scaling factor for the
             alpha parameter.
        max_gradient: Maximum gradient for (
            NaN) gradient clipping in sqrt
            operation.
    """
    forget_base: float
    alpha_log_scale: float
    state_size: int
    d_model: int
    max_gradient: float = 1000.0

    def setup(self):
        self.W_a = Dense(self.state_size)
        self.W_x = Dense(self.state_size)
        self.W_input = Dense(self.state_size,
             use_bias=False)
        self.W_output = Dense(self.d_model,
            use_bias=False)
        self.W_gate = Dense(self.state_size,
            use_bias=False)
        self.Conv1D = Conv(features=
            state_size, kernel_size=4)

    def __call__(self, x_t):
        sidegate = gelu(self.W_gate(x_t))
        x_t = self.Conv1D(x_t)

        r_t = sigmoid(self.W_a(x_t))
        softplus_forget_base = softplus(self.
            forget_base)

        % Calculate a_t in log space for
            stability
        a_t = jnp.exp(self.alpha_log_scale *
            softplus_forget_base * r_t)
        log_a = -8.0 * gate_a * jax.nn.
            softplus(a_param)
        a = jnp.exp(log_a)


        a_squared = jnp.exp(2 * log_a)
        beta = sqrt_bound_derivative(1 -
            a_squared, self.max_gradient)
        i_t = (beta * sigmoid(self.W_x(x_t))
            * x_t)

        h_t = ParallelScan(a_t, i_t)
        y_t = self.W_output(sidegate * h_t)
        return y_t
```