Birdie: Advancing State Space Models
with Reward-Driven Objectives and Curricula

Sam Blouir[1], Jimmy T.H. 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), such as linear recurrent neural networks and linear attention variants, offer computational advantages over Transformers but struggle with tasks requiring long-range in-context retrieval-like text copying, associative recall, and question answering over long contexts. Previous efforts to address these challenges have focused on architectural modifications, often reintroducing computational inefficiencies. In this paper, we propose a novel training procedure, Birdie, that significantly enhances the in-context retrieval capabilities of SSMs without altering their architecture. Our approach combines bidirectional input processing with dynamic mixtures of specialized pre-training objectives, optimized via reinforcement learning. We introduce a new bidirectional SSM architecture that seamlessly transitions from bidirectional context processing to causal generation. Experimental evaluations demonstrate that Birdie markedly improves performance on retrieval-intensive tasks such as multi-number phone book lookup, long paragraph question-answering, and infilling. This narrows the performance gap with Transformers, while retaining computational efficiency. Our findings highlight the importance of training procedures in leveraging the fixed-state capacity of SSMs, offering a new direction to advance their capabilities. All code and pre-trained models are available at https://www.github.com/samblouir/birdie.

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., 2017b) 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, 2024a), and copying (Jelassi et al.,
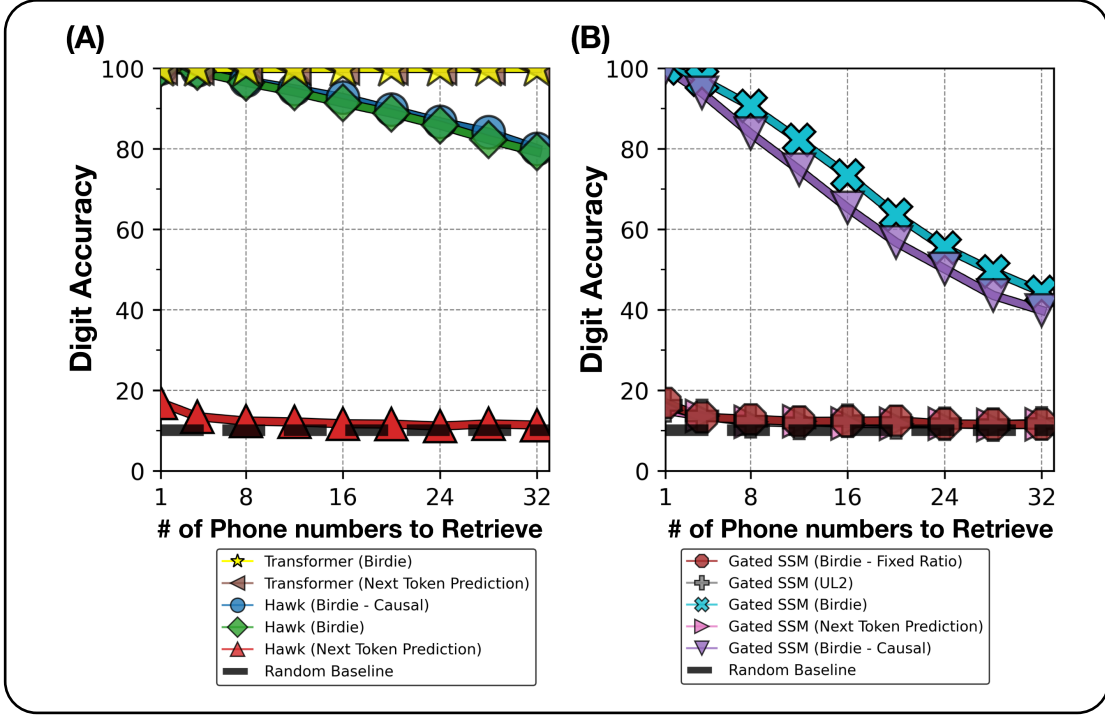
Figure 1: The **Multi-Phone Number Retrieval Task** entails finding and retrieving 1-32 phone numbers over a sequence length of 16,384. We demonstrate that State Space Models (SSMs) trained with Birdie significantly reduce their performance disparity with Transformers. For further details, please see section 4.3. **(A)** We conduct an ablation study comparing Hawk with Birdie, Birdie - Causal, and Next Token Prediction, alongside a Transformer using Birdie and Next Token Prediction. Hawk trained with Birdie and Birdie - Causal demonstrate significantly higher performance than when trained using Next Token Prediction. **(B)** An ablation that includes UL2 and the Fixed Ratio Mixture on our Gated SSM.

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 to address these weaknesses has been to formulate hybrid models that interleave SSM layers with global attention layers (Mehta et al., 2023; Fu et al., 2023a; Park et al., 2024; Poli et al., 2024), or sliding window attention (Beltagy et al., 2020; Arora et al., 2024a; De et al., 2024).[1] 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., 2024a; De et al., 2024).

The predominant focus on architecture to im-

prove performance on long-range in-context abilities misses an opportunity to investigate the role of the pre-training objectives and the potential interaction between the training procedure and model architecture. We note that prior work on generative SSMs exclusively utilizes Next Token Prediction for its 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 objec-

---

[1]Sliding window attention, introduced in Longformer (Beltagy et al., 2020), can be viewed as a type of fixed-state size method.

tive 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 automating much of 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.

Our new training procedure, which we call *Birdie*, significantly improves SSM performance on context-heavy and recall-intensive tasks, making this class of models more competitive with Transformers. While a performance gap with Transformers remains on certain tasks as retrieval demands increase (e.g., requiring more retrievals per example), Birdie reduces the severity of performance degradation in these scenarios and extends the range where these efficient models are effective. More broadly, our work underscores the importance of considering both learning dynamics and the inductive biases of SSM architectures to maximize the utility of their fixed state size.

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 shown:

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

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:

$$\mathbf{r}_t = \sigma(\mathbf{W}^a \mathbf{x}_t,)$$
$$\mathbf{i}_t = \sigma(\mathbf{W}^x \mathbf{x}_t),$$
$$\mathbf{a}_t = \sigma(\Lambda)^{cr_t}$$
$$\mathbf{h}_t = \mathbf{a}_t \odot \mathbf{h}_{t-1} + \sqrt{1 - \mathbf{a}_t^2} \odot (\mathbf{i}_t \odot \mathbf{x}_t)$$

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. (2024a) 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 Transformers (Biderman et al., 2023) to Mamba SSMs (Gu and Dao, 2023) 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., 2024a), 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 Next Token Prediction 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 Next Token Prediction, several alternative objectives have been proposed that can improve model performance in general language tasks (Tay et al., 2022, 2023b; Anil et al., 2023), code generation (Bavarian et al., 2022; Rozière et al., 2024a), and

multi-modal audio and vision Transformers (Chen et al., 2023).

For instance, *Masked Language Modeling* 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. *Infilling (Span Corruption)* extends this objective to generative models (Dong et al., 2019; Raffel et al., 2020). For a given input, several spans of tokens are each replaced with unique sentinel tokens. The model then generates the masked tokens and their respective sentinel tokens. In *Prefix language Modeling*, no loss is computed for the prefix, and the model can access the prefix bidirectionally. 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., 2023b) objectives combine Prefix language Modeling and Infilling (Span Corruption).

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 reinforcement learning 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

Encoder-Decoder models encode a prefix sequence into a single vector (or set of hidden states, for cross-attention) to generate a suffix from (Kalchbrenner and Blunsom, 2013; Cho et al., 2014; Sutskever et al., 2014; Raffel et al., 2020). The prefix-LM architecture, introduced in T5 and UniLM (Raffel et al., 2020; Dong et al., 2019),

simplifies this two-stage approach by instead only using self-attention layers with specific masking to allow for bidirectional processing in the prefix and causality in the suffix regions. We adapt this prefix-LM approach for recurrent models, which allows us to transfer information between the prefix and suffix regions for every layer, in contrast to the single-vector bottleneck with Encoder-Decoders.

This adaptation ensures our bidirectional SSMs maintain computational and parameter efficiency, enabling fair comparisons with unidirectional models.

To adapt prefix-LM to our models, we first split the recurrent state into forward and reverse components. The forward components are processed without modification, enabling our bidirectional layers to transmit information from the prefix to the suffix via the forward state dimensions. This contrasts with the bidirectional Encoder layers in Encoder-Decoder models, which are constrained to operate only within the prefix. The reverse components are modified in the suffix region to maintain causality; specifically, we zero out the forget gate (A_t) dimensions. This approach prevents the state from propagating information backward in causal areas. A mathematical description follows[2], and an additional example is included in Appendix Section A.4.

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

Compared to causal SSMs, our bidirectional SSMs trained with Birdie show greatly enhanced capabilities on question-answering retrieval with information-dense Wikipedia articles in SQuAD V2, shown in Figure 3. As we allocate half of the state for the reverse direction, we conservatively only allow half of Birdie's layers to be bidirectional. On the multi-phone number retrieval task, we only observe a small reduction In contexts where the information is Although Birdie - Causal allows for more information to flow forward

---

[2]Efficient implementations are provided in our codebase: https://www.github.com/samblouir/birdie

Performance is on-par with the Birdie - Causal model in the bandwidth [...] Our findings demonstrate that these bidirectional SSMs surpass their unidirectional equivalents in multiple such tasks, underscoring the utility of bidirectional processing.

We note that concurrent work, Arora et al. (2024b) also applies bidirectional processing to the prefix area with linear attention models and applies independent weights for the prefix and suffix regions. Additionally, they consider a fixed pre-training objective that mixes Prefix language Modeling with Masked Language Modeling, as in work such as UniLM and AlphaCode (Dong et al., 2019; Li et al., 2022). Our experimental results with the UL2 objective in Section 4 suggest that objectives such as this may be insufficient to improve SSM performance on retrieval-intensive tasks.

3.2 Pre-training Objectives for SSMs

We hypothesize that Next Token Prediction does not strongly necessitate in-context retrieval capabilities within SSMs. For most of the pre-training corpus, much of this objective can be achieved by leveraging information from local tokens alone (Xiao et al., 2024). Additionally, common pre-training data preprocessing techniques eliminate repeated or duplicated data in individual training samples (Raffel et al., 2020; Xue et al., 2021; Groeneveld et al., 2024), further reducing the model's need to learn dense copying or long-range retrieval mechanisms. These factors collectively hinder the model's abilities to retrieve information over long distances. Although Next Token Prediction does not prevent Transformers from developing long-range retrieval capabilities, SSM architectures inherently possess different inductive biases.

To enhance the in-context retrieval abilities of SSMs in downstream tasks, we design novel objective mixtures that explicitly train models to learn long-range and high-density retrieval abilities throughout the pre-training process. We list these objectives and mixtures that we investigate in Table 1, and briefly describe several objectives that are core to our new methods:

Selective Copying:

We introduce a novel pre-training task, termed Selective Copying, in which the model is trained to retrieve specified spans within a given context, located between designated start and end tokens. An example is provided in Figure 2, and a detailed explanation of the task's format is included in Appendix Section A.5. This objective enables SSMs

to perform zero-shot text retrieval after pre-training. Figure 5 in the Appendix demonstrates model performance on this task using validation data from The Pile, which includes sources such as emails and Wikipedia articles. The design of this pre-training objective is inspired by the work of Olsson et al. (2022), which explore similar synthetic induction head tasks.

Figure 2. Selective Copying Example

Original Input: Birds sing in the morning.
Processed Input: [COPY] [START] Birds [END] the   [CONTEXT] Birds sing in the morning.
Target: sing in

Figure 2: **Models retrieve text found between special start and end tokens** in our self-supervised Selective Copying pre-training task. Please see Section 3.2 and Appendix Section A.5 for more details. We show model performance on this task in the "Selective Copying" column of Figure 5.

Copying: This is a straightforward recreation of the input sequence, where the model must generate a given input sequence verbatim. This objective is inspired by recent studies (Jelassi et al., 2024) that discuss the difficulties SSMs face with copying and retrieval tasks.

Deshuffling: The model is presented with a sequence where the tokens are shuffled. The challenge is to reorganize these tokens to restore the original sequence. We implement two variations: one where 50% of the tokens are shuffled, and another where the entire sequence is shuffled.

Autoencoding: The input sequence is noised using masked spans of tokens, and the model is tasked with generating the original input sequence. This process involves both copying the unmasked tokens from the input and generating new, denoised spans to replace the masked sections. This approach can be seen akin to BERT's Masked Language Modeling task. However, instead of predicting and infilling masked tokens in their original positions, Autoencoding reconstructs the entire sequence. An equivalent objective is found in BART's pre-training objectives (Lewis et al., 2019), where the model learns to reconstruct the original text from a corrupted version. Additionally, a similar, albeit masking tokens, rather than

spans, strategy is found in T5's ablations (Raffel et al., 2020), referred to as "Bert-style".

Autoencoding with deshuffling: This builds on the Autoencoding objective by also shuffling the non-corrupted spans. Effectively, this merges infilling, copying, and de-shuffling into one objective. We hypothesize this may promote transfer learning between objectives. This objective is nearly equivalent to the "*Text Infilling + Sentence Shuffling*" objective found in BART's ablations (Lewis et al., 2019), however, we shuffle sequences between our unmasked spans without any regard to the location of sentence stop words.

Fixed Ratio Mixture: A mixture of all the objectives listed in Table 1 at fixed ratios found by ablation on several downstream tasks using the EleutherAI LM Harness. We provide more details in Appendix Section 2.3. We discuss dynamic ratios in Section 3.3.

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 reward 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 Masked Language Modeling (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 average the top 8 actions with greatest predicted reward.

We visualize loss, greedy-decoding accuracy, and sampling probabilities for training objective categories in Figure 5 in Appendix A. We observe

Table 1. Objectives and Mixtures

`Input: Bird songs fill the early morning air.`

Objectives	Example
Infilling (Span Corruption)	In: Bird [mask] the early [mask] Tgt: songs fill [mask] air [mask]
Next Token Prediction	In: Bird songs fill the early morning Tgt: songs fill the early morning air
Prefix Language Modeling	In: Bird songs fill Tgt: the early morning air
Copying	In: Bird songs fill the early morning air 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
Autoencoding	In: Bird [mask] the early [mask] Tgt: Bird songs fill the early morning air
Autoencoding + Deshuffling	In: the early [mask] Bird [mask] Tgt: Bird songs fill the early morning air
Selective Copying	Please see Figure 2 for an example.

Mixtures	Description
Birdie	Dynamic mixture of above objectives using a reward model for controlling parameterization and sampling ratios.
Fixed Ratio Mixture	A mixture of all objectives using a fixed ratio found via ablations on max likelihood tasks.
UL2	A Fixed Ratio Mixture consisting of Prefix language Modeling and Infilling. Described further in Appendix Section A.6.

Table 1: This table presents the training objectives and mixtures used in our paper. Birdie effectively parameterizes these objectives, allowing for independent control of multiple factors, a capability that is further elaborated in Appendix Section A.2. Model performance for each objective configuration is detailed in Appendix Section 5. 'In' denotes the input text; 'Tgt' refers to the target output.

several trends, such as that training on the Autoencoding objective appears to boost both the 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 regardless of sampling rate after the model has seen sufficient amounts of these samples.

We integrate our automated approach with the new objectives introduced in Section 3.2 and the bidirectional processing from Section 3.1 and Appendix A.3, resulting in a method we call *Birdie*. As shown in Section 4, Birdie consistently enhances SSM performance across a variety of downstream tasks.

3.4 Gated SSM baseline

We define a generic Gated SSM baseline to also test our methods on other general SSMs. The recurrence equations are:

$$\mathbf{i}_t = \sigma(\mathbf{W}^i \mathbf{x}_t) \in \mathbb{R}^N,$$
$$\mathbf{z}_t = \mathbf{W}^z \mathbf{x}_t \in \mathbb{R}^N,$$
$$\mathbf{o}_t = \mathsf{GeLU}(\mathbf{W}^o \mathbf{x}_t) \in \mathbb{R}^N,$$
$$\mathbf{f}_t = \sigma(\mathbf{W}^f \mathbf{x}_t) \in \mathbb{R}^N,$$

$$\mathbf{h}_t = \mathbf{f}_t \odot \mathbf{h}_{t-1} + \mathbf{i}_t \odot \mathbf{z}_t,$$
$$\mathbf{y}_t = \mathbf{W}^{out}(\mathbf{o}_t \odot \mathbf{h}_t),$$

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 is done in Gated State Spaces and Mamba (Mehta et al., 2023; Gu and Dao, 2023). We find this simple baseline performs comparably with state-of-the-art SSMs, such as Hawk, on max-likelihood tasks, but does not perform as well when

asked to retrieve multiple phone numbers simultaneously, or when generating responses to questions about Wikipedia excerpts in SQuAD V2.

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 versions of Hawk, a state-of-the-art SSM, using either Next Token Prediction or the Birdie training procedure described in Section 3.3, with its bidirectional prefix processing and dynamic mixture selection. We also include a version without bidirectional prefix processing we refer to as Birdie - Causal. In addition, we train versions of a modern Transformer architecture using either Next Token Prediction or Birdie[3].

To show our Birdie training procedure is more broadly applicable to other model architectures, we also train several simple 1.4B Gated SSM baseline models, described in Section 3.4 to serve as generalized recurrent models. In addition, to ablate different aspects of the Birdie design, we train these Gated SSMs using additional objectives and mixtures as described in Table 1: Next Token Prediction objective, UL2, Fixed Ratio Mixture, Birdie and Birdie - Causal. Further pre-training details can be found in Appendix A.11.

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., 2023b). For FLAN, we extend the maximum sequence length to 4096 and further increase it to 8192 for Open-Hermes 2.5 (Teknium, 2023). More details on fine-tuning can be found in Sections A.12.

Evaluations: First, we evaluate our models across 21 tasks using the EleutherAI LM Har-

---

ness (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.14 and show per-task performance in Appendix Table 10. We then stress test in-context retrieval abilities of our models by evaluating on tasks previously shown to be difficult for SSMs (Jelassi et al., 2024) such as a multi-number phone book lookup task and SQuAD V2 paragraph Q&A. We also introduce a new infilling dataset to test the models' abilities to comprehend the full context of a story and infill a missing segment.

4.2 Comparative Performance and Ablation Study on Maximum-Likelihood Tasks

We report the average accuracy across 21 unique tasks in Table 2, with specific task-level metrics provided in Appendix Table 10. Our findings indicate that models trained using the Birdie procedure perform comparably to those using the Next Token Prediction objective, demonstrating that Birdie-trained models effectively maintain the knowledge and reasoning skills assessed by these benchmarks.

Table 2. Average Accuracy on 21 Tasks

Model	Training Procedure	Accuracy (%)
Hawk	**Birdie**	**41.4**
	Birdie (Causal)	40.8
	Next Token Prediction	39.6
Transformer	Birdie	39.7
	Next Token Prediction	**40.4**

(A) Instruct Models

Model	Training Procedure	Accuracy (%)
Hawk	Birdie	38.3
	Birdie (Causal)	**39.0**
	Next Token Prediction	**39.1**
Transformer	Birdie	38.5
	Next Token Prediction	**39.9**

(B) Base Models

Table 2: Average accuracy (%) on 21 tasks, including ARC, MMLU, and LogiQA. Models trained using Birdie perform comparably to Next Token Prediction. More details and ablations can be found in Appendix section 10. (A) Instruction-tuned models fine-tuned on FLAN and OpenHermes 2.5 after pre-training. (B) Base models after pre-training.

---

[3]The Transformer - Birdie variant uses unmasked attention on the prefix, equivalent to the Prefix language Modeling architecture described in (Raffel et al., 2020)
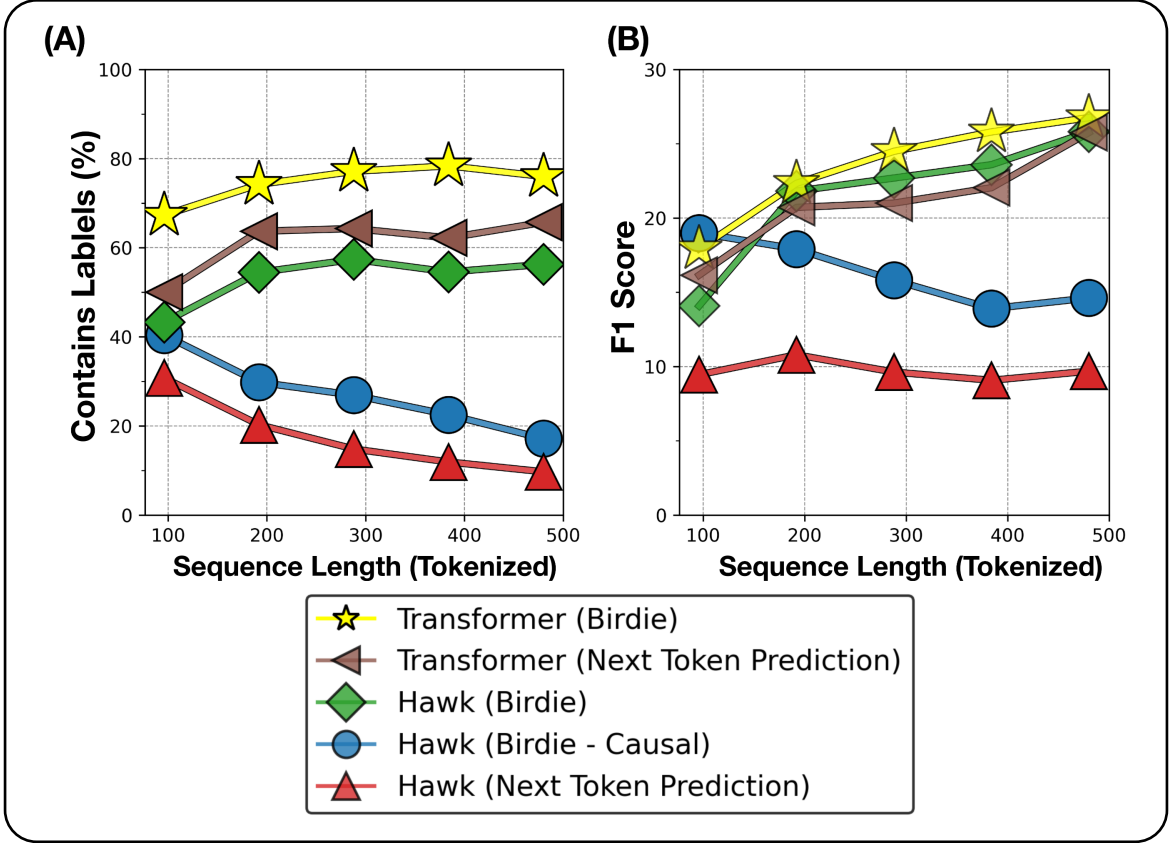
Figure 3: SQuAD V2 (Question Answering)

Figure 3: SQuAD V2 Question-Answering results with instruction-tuned models. Training with the Birdie procedure strongly improves SSM performance, compared to Next Token Prediction. Average results are shown in Table 3. Further details and ablations are available in Section 4.4 and Appendix Section A.15. (A) Answer Contains Label measures when a label is produced by the model verbatim. (B) The F1 Score awards partial credit for matching words in the label and also penalizes models for generating words not in labels.

4.3 Analysis on Multi-Phone Number Retrieval

Next, we explore the phone number retrieval task that previous works have found SSMs trained at various scales struggle on (Jelassi et al., 2024; De et al., 2024; Waleffe et al., 2024). In addition, to make it more challenging and further stress-test retrieval capacity, we also measure the simultaneous retrieval of up to 32 numbers from phone books containing 750-800 entries.

All models underwent minimal fine-tuning from their base configurations, primarily to extend the positional encodings of Transformers to handle longer sequence lengths-from 2,048 to 16,384 tokens. The fine-tuning process spanned 1,000 steps with a batch size of 64, utilizing training samples containing a uniformly sampled number of entries ranging from 200 to 800. For additional details, please see Appendix Section A.9.

We summarize the main multi-phone number

retrieval results in Figure 1A. We note that, as expected, the Transformer models achieve high performance regardless of the quantity of phone numbers being retrieved, and whether they are trained with either Birdie or Next Token Prediction. However, the Birdie trained Transformer reaches near perfect accuracy much sooner than the Next Token Prediction model.

We observe that SSMs trained with Next Token Prediction perform poorly, even when asked to retrieve only a single phone number and after thorough hyperparameter ablations. In contrast, we see that SSMs trained with the Birdie procedure significantly outperforms the Next Token Prediction SSM across the regime of different amounts of phone numbers to be retrieved. In fact, the Birdie SSMs achieve 100% accuracy across 1,024 unique phone books when retrieving a single number and overall significantly reduces the performance gap with the Transformer baselines. While we do observe the

performance of the Birdie SSMs degrades as the task complexity increases (i.e. increasing the quantity of phone numbers to be retrieved to 32), the Birdie procedure creates a regime in which SSMs can perform these types of retrievals. We theorize that a larger 7B parameter SSM trained using Birdie may match the Transformer's performance on this task.

Next, we examine the phone number retrieval task, a challenge which previous works have shown SSMs at various scales struggle with (Jelassi et al., 2024; De et al., 2024; Waleffe et al., 2024). To increase the task's difficulty and further stress-test retrieval capacity, we also measure the simultaneous retrieval of up to 32 numbers from phone books containing between 750–800 entries across 16,384 tokens.

All models underwent minimal fine-tuning from their base configurations, mainly to extend the positional encodings of Transformers to handle longer sequence lengths, from 2,048 to 16,384 tokens. The fine-tuning process spanned 1,000 steps with a batch size of 64, using training samples that contained a uniformly sampled number of entries ranging from 200 to 800. For additional details, please refer to Appendix Section A.9

We summarize the main multi-phone number retrieval results in Figure 1A. As expected, Transformer models achieve high performance regardless of the quantity of phone numbers being retrieved, whether trained with Birdie or Next Token Prediction. However, the Transformer trained with Birdie achieves near-perfect accuracy with far fewer steps needed than the Next Token Prediction model.

Additionally, we observe that SSMs trained with Next Token Prediction perform poorly, even when retrieving a single phone number, despite thorough hyperparameter tuning[4]. In contrast, SSMs trained with the Birdie procedure significantly outperform the Next Token Prediction SSM across varying retrieval demands. Birdie-trained SSMs achieve 100% accuracy across 1,024 unique phone books when retrieving a single number, eliminating the performance gap with Transformer baselines. While Birdie-trained SSM performance decreases as task complexity increases (e.g., retrieving up to 32 phone numbers), the Birdie procedure enables SSMs to perform these types of retrievals in the

---

[4]We describe the hyperparameter ablations done to improve non-Birdie models on the multi-phone number retrieval task in Appendix Section A.9

first place. As our models only have 1.4B parameters, we hypothesize that a larger 7B parameter SSM trained with Birdie may be able to match Transformer performance on this task.

Table 3: SQuAD V2 Question-Answering

Model	Training Procedure	Contains Labels (%)
Hawk	**Birdie**	**55.8**
	Birdie (Causal)	27.2
	Next Token Prediction	16.6
Transformer	**Birdie**	**76.1**
	Next Token Prediction	63.6

(A) Contains Labels (%)

Model	Training Procedure	F1
Hawk	**Birdie**	**22.4**
	Birdie (Causal)	16.3
	Next Token Prediction	10.0
Transformer	**Birdie**	**23.7**
	Next Token Prediction	21.0

(B) F1 Scores

Table 3: Results for instruction-tuned models on SQuAD V2 Question-Answering, where models are given a Wikipedia excerpt and then asked questions about it. This detailed breakdown shows different aspects of model performance. Plots versus context length shown in Figure 3. Details and ablations are available in Section 4.4 and Appendix Section A.15. (A) Answer Contains Label measures when a label is produced by the model verbatim. (B) The F1 Score awards partial credit for matching words in the label and also penalizes models for generating words not in labels.

Ablations We also ablate variations of the Birdie training procedure on this task using the basic Gated SSM model. We show these results in Figure 1B. The Birdie procedure significantly enhances performance across all tasks compared to all other training procedures considered. We observe a slight but consistent performance boost of the Birdie trained model over the Birdie - Causal trained model, indicating the usefulness of the bidirectional processing of the prefix for the Gated SSM. We also observe the UL2 and Fixed Ratio Mixture procedure (which also uses bidirectional processing of the prefix) does not appear to induce the retrieval abilities necessary for phone number retrieval. In addition, the Fixed Ratio Mixture's lack of improvement provides evidence of the importance of Birdie's dynamic mixtures for superior training. These same trends generally hold across

the Gated SSM ablations for the infilling task (Appendix Section A.16) and SQuAD V2 Question-Answering (Appendix Table A.15).

4.4 Question-Answering

We next evaluate performance on the SQuAD V2 Question-Answering task (Rajpurkar et al., 2018). Using greedy decoding for up to 128 tokens on all answerable questions, we format inputs as ``{context}\n\n{question}'' without including any few-shot examples. We report an "Answer Contains Label", where a question is considered correct if any of the labels are found in the response, as well as the classic F1 score.

Table 3 presents the average results and Figure 3 shows the results as a function of sequence length. The performance of Next Token Prediction-trained SSMs 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. We note that for this task, unlike the Phone Number Retrieval task in the previous section or the Infilling task in the next section, there is a more meaningful gap between the Birdie and Birdie - Causal trained SSMs. This indicates the bidirectional processing of the prefix may be particularly helpful for this task. Further ablations, tables, and details are available in Appendix Section A.15.

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 ordered story entries, one of which is made blank. Models then predict the most appropriate option to fill this blank. As on other tasks, we observe that the Birdie procedure allows the SSM models to perform more closely to the Transformer baselines. The Transformer trained with Birdie also improves its performance. Table 4 relates the main results. More results, details, and an example can be found in Appendix A.16.

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

Table 4: Story Infilling

Model	Training Procedure	Accuracy (%)
Hawk	**Birdie**	**42.5**
	Birdie (Causal)	41.5
	Next Token Prediction	33.1
Transformer	**Birdie**	**42.2**
	Next Token Prediction	41.9

(A) Instruct Models

Model	Training Procedure	Accuracy (%)
Hawk	Birdie	36.6
	Birdie (Causal)	**38.5**
	Next Token Prediction	29.4
Transformer	Birdie	39.8
	Next Token Prediction	**40.5**

(B) Base Models

Table 4: Average accuracy (%) on the new infilling dataset, where models complete story segments. Birdie-trained SSMs surpass Next Token Prediction-trained SSMs. For data samples and more, please see Appendix section A.16. (A) Instruction-tuned models fine-tuned on FLAN and OpenHermes 2.5 after pre-training. (B) Base models after pre-training.

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 Next Token Prediction, 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

It is important to note that these experiments were constrained by an academic budget. While our 1.4B models, trained on 32B tokens, are sufficiently large for specific tasks-such as extracting multiple text spans simultaneously-the scalability of these results with larger models and additional data remains uncertain. Although the 8B Mamba and Mamba-2 models, trained on 3.5 trillion tokens with the Next Token Prediction objective, struggle with tasks like phonebook lookup, which our models appear capable of handling (Waleffe et al., 2024), we did not evaluate whether these larger models could be fine-tuned for such tasks. Initial attempts at a 'second-stage' pre-training with our 1.4B models were also unsuccessful.

The simplicity of the Next Token Prediction objective is difficult to surpass in terms of implementation. In contrast, training setups that employ a mixture of objectives require careful tuning to ensure correct implementation.

Another limitation is the availability of long-context evaluations for LLMs. Tasks that cleanly separate parametric knowledge from true in-context reasoning abilities are scarce (Hsieh et al., 2024). This is especially challenging in realistic question-answering tasks, where the knowledge required may already be memorized from training data. While our long-paragraph question-answering and infilling tasks may be susceptible to this issue, synthetic tasks like phonebook retrieval can more reliably assess in-context reasoning, though their practical relevance is often questioned. Ongoing innovation in long-context evaluation methods is crucial for enhancing language models' long-context capabilities, independent of architecture.

Finally, we observe that SSMs' performance on retrieval tasks degrades faster than Transformer baselines. We do not claim to have fully solved the retrieval problem, and there are likely other limitations of SSMs that were not captured by the tasks explored in our study.

Acknowledgments

References

AI across google: PaLM 2. https://ai.google/discover/palm2/. Accessed: 2023-05-28.

Roee Aharoni, Melvin Johnson, and Orhan Firat. 2019. Massively multilingual neural machine translation. 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 3874–3884.

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.

Ido Amos, Jonathan Berant, and Ankit Gupta. 2024. Never train from scratch: Fair comparison of long-sequence models requires data-driven priors. *Preprint*, arXiv:2310.02980.

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

Martin Arjovsky, Amar Shah, and Yoshua Bengio. 2016. Unitary evolution recurrent neural networks. *Preprint*, arXiv:1511.06464.

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. 2024a. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-first International Conference on Machine Learning*.

Simran Arora, Aman Timalsina, Aaryan Singhal, Benjamin Spector, Sabri Eyuboglu, Xinyi Zhao, Ashish Rao, Atri Rudra, and Christopher Ré. 2024b. Just read twice: closing the recall gap for recurrent language models. *arXiv preprint arXiv:2407.05483*.

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.

Maarten Bosma and Jason Wei. Introducing FLAN: More generalizable language models with instruction fine-tuning. https://ai.googleblog.com/2021/10/introducing-flan-more-generalizable.html. Accessed: 2023-05-28.

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.

Rewon Child, Scott Gray, Alec Radford, and Ilya Sutskever. 2019. Generating long sequences with sparse transformers. *Preprint*, arXiv:1904.10509.

Kyunghyun Cho, Bart van Merrienboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *Preprint*, arXiv:1406.1078.

Krzysztof Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, et al. 2021. Rethinking attention with performers. In *ICLR*.

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

Hyung Won Chung, Noah Constant, Xavier Garcia, Adam Roberts, Yi Tay, Sharan Narang, and Orhan Firat. 2023. Unimax: Fairer and more effective language sampling for large-scale multilingual pretraining. *Preprint*, arXiv:2304.09151.

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.

Zihang Dai, Zhilin Yang, Yiming Yang, Jaime Carbonell, Quoc V. Le, et al. 2019. Transformer-xl: Attentive language models beyond a fixed-length context. In *ACL*, page 2978–2988.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *Preprint*, arXiv:2307.08691.

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.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *CoRR*, abs/1905.03197.

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*.

N. Benjamin Erichson, Omri Azencot, Alejandro Queiruga, Liam Hodgkinson, and Michael W. Mahoney. 2021. Lipschitz recurrent neural networks. In *ICLR*.

Fahim Faisal and Antonios Anastasopoulos. 2021. Investigating post-pretraining representation alignment for cross-lingual question answering. In *Workshop on Machine Reading for Question Answering (MRQA)*, Online. Association for Computational Linguistics.

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.

Xavier Garcia, Yamini Bansal, Colin Cherry, George Foster, Maxim Krikun, Fangxiaoyu Feng, Melvin Johnson, and Orhan Firat. 2023. The unreasonable effectiveness of few-shot learning for machine translation. *Preprint*, arXiv:2302.01398.

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.

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

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

Albert Gu, Tri Dao, Stefano Ermon, Atri Rudra, and Christopher Re. 2020. Hippo: Recurrent memory with optimal polynomial projections. *Preprint*, arXiv:2008.07669.

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.

Ankit Gupta, Albert Gu, and Jonathan Berant. 2022. Diagonal state spaces are as effective as structured state spaces. *Preprint*, arXiv:2203.14343.

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, Andrew Critch, Jerry Li, Dawn Song, and Jacob Steinhardt. 2021a. Aligning ai with shared human values. *Proceedings of the International Conference on Learning Representations (ICLR)*.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021b. 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.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1700–1709. ACL.

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.

Vid Kocijan, Oana-Maria Camburu, Ana-Maria Cretu, Yordan Yordanov, Phil Blunsom, and Thomas Lukasiewicz. 2019. WikiCREM: A large unsupervised corpus for coreference resolution. In *EMNLP-IJCNLP*, pages 4303–4312, Hong Kong, China. Association for Computational Linguistics.

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.

Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard Hovy. 2017. RACE: Large-scale ReAding comprehension dataset from examinations. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 785–794, Copenhagen, Denmark. 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.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *Preprint*, arXiv:1910.13461.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

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.

Xuezhe Ma, Chunting Zhou, Xiang Kong, Junxian He, Liangke Gui, Graham Neubig, Jonathan May, and Luke Zettlemoyer. 2023. Mega: Moving average equipped gated attention. *Preprint*, arXiv:2209.10655.

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.

OpenAI. 2023. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

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, Stanislaw 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.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. *Preprint*, arXiv:2305.18290.

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.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2023. Exploring the limits of transfer learning with a unified text-to-text transformer. *Preprint*, arXiv:1910.10683.

Altaf Rahman and Vincent Ng. 2012. Resolving complex cases of definite pronouns: The Winograd schema challenge. In *EMNLP-IJNLP*, pages 777–789, Jeju Island, Korea. Association for Computational Linguistics.

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*.

Reka Team, Aitor Ormazabal, Che Zheng, Cyprien de Masson d'Autume, Dani Yogatama, Deyu Fu, Donovan Ong, Eric Chen, Eugenie Lamprecht, Hai Pham, Isaac Ong, Kaloyan Aleksiev, Lei Li, Matthew Henderson, Max Bain, Mikel Artetxe, Nishant Relan, Piotr Padlewski, Qi Liu, Ren Chen, Samuel Phua, Yazheng Yang, Yi Tay, Yuqi Wang, Zhongkai Zhu, and Zhihui Xie. 2024. Reka core, flash, and edge: A series of powerful multimodal language models. *Preprint*, arXiv:2404.12387.

David W Romero, Anna Kuzinna, Erik J Bekkers, Jakub M Tomczak, and Mark Hoogendoorn. 2022. Ckconv: Continuous kernel convolutions for sequential data. In *ICLR*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions ACL*, 9:53–68.

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. 2024a. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

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. 2024b. Code llama: Open foundation models for code. *Preprint*, arXiv:2308.12950.

T. Konstantin Rusch and Siddhartha Mishra. 2021. Unicornn: A recurrent model for learning very long time dependencies. In *ICML*, volume 139 of *Mach Learn Res*, pages 9168–9178. PMLR.

Alexander Rush and Sidd Karamcheti. The annotated S4. https://srush.github.io/annotated-s4/. Accessed: 2023-05-28.

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

Noam Shazeer. 2020. Glu variants improve transformer. *Preprint*, arXiv:2002.05202.

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.

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

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*.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to sequence learning with neural networks. In *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, page 3104–3112, Cambridge, MA, USA. MIT Press.

Yi Tay, Mostafa Dehghani, Samira Abnar, Hyung Chung, William Fedus, Jinfeng Rao, Sharan Narang, Vinh Tran, Dani Yogatama, and Donald Metzler. 2023a. Scaling laws vs model architectures: How does inductive bias influence scaling? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12342–12364, Singapore. Association for Computational Linguistics.

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. 2020. Long range arena: A benchmark for efficient transformers. *Preprint*, arXiv:2011.04006.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, Xuezhi Wang, Hyung Won Chung, Dara Bahri, Tal Schuster, Steven Zheng, Denny Zhou, Neil Houlsby, and Donald Metzler. 2023b. UL2: Unifying language learning paradigms. In *The Eleventh International Conference on Learning Representations*.

Yi Tay, Mostafa Dehghani, Vinh Q. Tran, Xavier Garcia, Jason Wei, et al. 2023c. 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*.

Teknium. 2023. Openhermes 2.5: An open dataset of synthetic data for generalist llm assistants.

W. R. Thompson. 1933. On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. *Biometrika*, 25(3-4).

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, Ł ukasz Kaiser, and Illia Polosukhin. 2017a. Attention is all you need. In *NeurIPS*, volume 30. Curran Associates, Inc.

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

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph attention networks. In *ICLR*.

Roger Waleffe, Wonmin Byeon, Duncan Riach, Brandon Norick, Vijay Korthikanti, Tri Dao, Albert Gu, Ali Hatamizadeh, Sudhakar Singh, Deepak Narayanan, Garvit Kulshreshtha, Vartika Singh, Jared Casper, Jan Kautz, Mohammad Shoeybi, and Bryan Catanzaro. 2024. An empirical study of mamba-based language models. *Preprint*, arXiv:2406.07887.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, et al. 2019. Superglue: A stickier benchmark for general-purpose language understanding systems. *NeurIPS*, 32.

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.

Boshi Wang, Xiang Yue, Yu Su, and Huan Sun. 2024. Grokked transformers are implicit reasoners: A mechanistic journey to the edge of generalization. *Preprint*, arXiv:2405.15071.

Junxiong Wang, Jing Nathan Yan, Albert Gu, and Alexander M. Rush. 2023a. Pretraining without attention. *Preprint*, arXiv:2212.10544.

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. 2023b. How far can camels go? exploring the state of instruction tuning on open resources. *Preprint*, arXiv:2306.04751.

Jason Wei, Maarten Bosma, Vincent Y. Zhao, Kelvin Gu, Adams Wei Yu, et al. 2022. Finetuned language models are zero-shot learners.

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.

Shijie Wu and Mark Dredze. 2020. Are all languages created equal in multilingual BERT? In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 120–130, Online. Association for Computational Linguistics.

Guangxuan Xiao, Jiaming Tang, Jingwei Zuo, Junxian Guo, Shang Yang, Haotian Tang, Yao Fu, and Song Han. 2024. Duoattention: Efficient long-context llm inference with retrieval and streaming heads. *arXiv preprint arXiv:2410.10819*.

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*.

Linting Xue, Aditya Barua, Noah Constant, Rami Al-Rfou, Sharan Narang, et al. 2022. ByT5: Towards a token-free future with pre-trained byte-to-byte models. *Transactions ACL*, 10:291–306.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

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*.

Manzil Zaheer, Guru Guruganesh, Avinava Dubey, Joshua Ainslie, Chris Alberti, et al. 2020. Big bird: Transformers for longer sequences. In *NeurIPS*.

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*.

Biao Zhang and Rico Sennrich. 2019. Root mean square layer normalization. *Preprint*, arXiv:1910.07467.

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 Reinforcement Learning for Objective Sampling

We propose Birdie, a reinforcement learning-based approach to dynamically adjust the sampling ratios and configurations (parameters) of multiple training objectives during model training. Our goal is to maximize the overall reduction in loss across various objective classes, under the assumption that each class is equally important to minimize. This method, described below, enables the model to learn which objectives and configurations are most beneficial at different stages of training. Critically, Birdie takes into account the interactions between training objectives.

A.1.1 Objective Classes

We train our models using a variety of objectives, which each have several configurations or parameterizations. These objective encourage the model to learn various aspects of language, such as next token prediction, Span Corruption, and sequence reordering, or tasks, like deshuffling and selective copying. The objectives we consider are:

- Next Token Prediction

- Prefix language Modeling

- Selective Copying

- Copying

- Deshuffling

- Infilling (Span Corruption)

- Autoencoding with Deshuffling

Examples of these can be found in Table 1, and the configurations are shown in Section A.2.

A.1.2 Actions and Configurations

In Birdie's framework, an **action** is a probability vector representing the sampling frequency for each training objective and configuration. To give Birdie additional control over the training process, we create multiple configurations for each objective class by varying parameters such as context sequence length and masking percentage. Each unique configuration is treated as a separate objective in the action space.

For instance, the *Deshuffling* objective includes configurations with varying percentages of shuffled tokens (50% or 100%) and sequence length ranges (between 64–512 or 512–1024 tokens). This allows the model to learn not only which objectives are beneficial to train on, as well as which specific parameter settings are most effective.

A.1.3 Reward Function

A reward vector, with elements corresponding to each possible objective parameterization, is calculated based on the change in loss achieved by that configuration. The reward function is designed to:

- **Reduce noise**: Small, insignificant changes in loss are scaled down.

- **Maintain scale**: Rewards are normalized to the range $[-1, 1]$ to stabilize learning. Negative rewards provide an intuitive interface for discouraging undesirable actions.

- **Focus on improvement areas**: Diminishing rewards for already low losses prevent over-focusing on well-performing objectives.

The reward for each objective configuration is calculated as:

$$\Delta L = \frac{\text{loss}_{\text{old}} - \text{loss}_{\text{new}}}{\text{loss}_{\text{old}}}, \quad S = \sqrt{\text{loss}_{\text{old}} \cdot \text{loss}_{\text{new}}},$$

$$\text{Reward} = -r \cdot 100 \cdot \tanh\left(r \cdot S \cdot (\Delta L)^3\right)$$

$$\text{Reward} = \text{clip}(\text{Reward}, -1, 1),$$

where:

- ΔL is the normalized change in loss.

- S is the geometric mean of the old and new losses.

- r is a sensitivity hyperparameter (by default, we use Euler's number e).

This function balances improvements across different loss scales. For example, reducing a loss from 4.5 to 4.2 yields a similar reward to reducing a loss from 0.6 to 0.5207, despite the difference in percentage changes (-7% vs -13%).

To normalize for certain objective classes having greater numbers of configurations than others, we compute the *Class Reward* R_c for each objective class c by averaging the rewards of all configurations within that class:

$$R_c = \frac{1}{N_c} \sum_{i \in c} R_i,$$

where N_c is the number of configurations within class c and R_i is the reward for configuration i. The *Total Reward* is then the sum of all Class Rewards:

$$R_{\text{total}} = \sum_c R_c.$$

A.1.4 Choosing an Action

At each evaluation iteration, the Birdie's action selection procedure involves the following steps. Please see the pseudocode in Appendix Figure 4 for added clarity.

1. **Prepare historical data: losses and action**: At each time step, we have two vectors. One represents the loss across objective configurations, and the other represents the action at that time, or sampling probability for each objective configuration. We stack and concatenate these, creating an input of shape (number of time steps, number of objective configurations * 2)

2. **Generate candidate actions**: Since we have prepared our historical data, we now pick an action for our current state. We randomly sample 2,048 probability vectors representing potential actions, and concatenate them with the current loss vector. This creates an input of shape (2048, number of objective configurations * 2).

3. **Predict rewards**: After concatenating the proposed actions with the current loss, we repeat the 2D array containing our historical losses and actions 2048 times. We concatenate these once more to creates an input of shape (2048, number of time steps+1, number of objective configurations * 2). We then process this using the reward model to obtain our predicted rewards per action.

4. **Select top actions**: We choose the 8 actions with the highest predicted rewards.

5. **Average actions**: Average these top actions and take this final action until the next evaluation.

Birdie is trained on historical loss, action, and uses observed reward data as labels, enabling it to dynamically estimate optimal actions based on past performance. Due to the limited amount of training samples, we find the best results by Grokking Birdie, training for 200 steps at every evaluation period rather than reaching a target loss.

Training begins with a warm-up phase where the language model is trained with uniform sampling across all objectives for the first 250 steps. Early evaluations run at 10, 50 and 250 steps to create initial training data for Birdie. Birdie is permanently given control of actions at 250 steps. Further evaluations then occur at steps 500, 1000, 1500, 2000, and every 1000 steps thereafter.

A.1.5 Reward Model Architecture

The reward model utilizes a Gated SSM architecture (Section 3.4) with four layers, each with a hidden size of 256. This model takes as input a sequence of historical losses and actions and predicts the future reward for a given action and current losses. We apply independent RMSNorm layers for the loss and action input dimensions.

A.2 Birdie's controls

Here, we describe the parameters, or configurations, for our objectives. For a given objective, its configurations are applied as the superset of available settings. We then allow Birdie to adaptively set the sampling probabilities for each configuration independently. When calculating the total reward, we normalize the reward by scaling each configuration's reward by the number of configurations for that objective class. A plot of the average performance per class is shown in Appendix Figure 5.

Infilling (Span Corruption):

1. The length of the sequence (between 128-256, 256-512, 512-1024, or 1024-2048 tokens long).

2. The total percentage of the sequence to be masked (5%, 15%, or 50%).

3. The mean length of spans (3, 8, or 32 tokens long).

Next Token Prediction and Prefix language Modeling: Due to an implementation limitation, we allow for no controls other than how often these objectives are sampled.

Selective Copying:

1. How many spans to find and copy at once (1, 2, 3, or 4 spans).

2. How often the context is presented either before or after the START and END segments to find appears. (For example, one style presents a phone book and then asks for a specific person's number. The other style first asks for the person's phone number, and then provides the book.)

3. The length of the context (between 384-768 or 768-1536 tokens long).

Copying: We allow Birdie to control the length of the sequence to copy, from between 64-256 or 256-2014 tokens to copy.

Deshuffling:

1. The length of the sequence (between 128-256 or 512-1024 tokens long).

2. How much of the sequence is shuffled (50% or 100% tokens are shuffled).

Autoencoding:

1. The length of the sequence (between 192-384, 384-768, or 768-1536 tokens long).

2. The total percentage of the sequence to be masked (15% or 85%).

3. The average length of masked spans for a given sequence (3, 8, or 32 mean span width)

4. Whether to shuffle the non-masked spans or not.

```
# Pseudocode for sampling an action from Birdie

# Prepare action and loss histories
# action_history: (time_steps, num_configs)
# loss_history:   (time_steps, num_configs)
history = concatenate(action_history, loss_history, axis=-1)
# history.shape: (time_steps, num_configs * 2)

# Generate 2048 proposed actions
# current_losses: (num_configs,)
proposed_actions = random_uniform(2048, num_configs)
proposed_actions /= sum(proposed_actions, axis=-1, keepdims=True)
# proposed_actions.shape: (2048, num_configs)

# Prepare inputs for Birdie
current_losses_expanded = repeat(current_losses[None], (2048, 1))
# current_losses_expanded.shape: (2048, num_configs)
input_actions = concatenate(current_losses_expanded, proposed_actions, axis=-1)
# input_actions.shape: (2048, num_configs * 2)

# Repeat history for batch processing
history_expanded = tile(history[np.newaxis, :, :], (2048, 1, 1))
input_sequence = concatenate(history_expanded, input_actions[:, np.newaxis, :],
    axis=1)
# input_sequence.shape: (2048, time_steps + 1, num_configs * 2)

# Predict rewards using Birdie
estimated_rewards = Birdie(input_sequence)
# estimated_rewards.shape: (2048, time_steps + 1, num_configs)

# Extract rewards for proposed actions
estimated_rewards = estimated_rewards[:, -1, :]
# Apply per-configuration scaling.
# For each objective's configurations, set the elements of scaling_vector to equal
    (1/num_configurations_for_the_objective).
# Otherwise, objectives with more configurations are over-represented in the total
    reward below.
estimated_rewards *= scaling_vector

# Compute total rewards
total_rewards = sum(estimated_rewards, axis=-1)
# Select top actions
top_indices = argsort(total_rewards)[-8:]
top_actions = proposed_actions[top_indices]

# Final action is the average of top actions
final_action = mean(top_actions, axis=0)
```

Figure 4: Pseudocode for sampling an action from Birdie.

A.3 Bidirectional Processing

A.3.1 Implementation Details

To efficiently implement bidirectional processing in SSMs, we adapt the prefix-LM architecture used with Transformers (Dong et al., 2019; Raffel et al., 2020; Tay et al., 2023c) to create a simple mechanism that enables bidirectionality in the prefix (inputs) while enforcing causality in the suffix (outputs). We use a careful construction of the input sequences and corresponding masks, shown below. Assuming masked sequence packing for efficient training, our approach is compute-matched with a causal scan operation.

Input Sequence Processing: Consider an example where we have the original input tokens $\{4, 5, 6\}$ and corresponding labels $\{7, 8, 9\}$. We construct a teacher-forced input to the model by concatenating the inputs and labels, with a special token (in our case, 1) inserted to indicate the beginning of the generation phase. These processed inputs become $\{4, 5, 6, 1, 7, 8\}$. The processed labels to calculate the loss on are $\{-, -, -, 7, 8, 9\}$, where the hyphens indicate positions without any loss (i.e., the model is not trained to predict these tokens).

Reset Mask for Sequence Packing and Bidirectionality: When packing samples into our training sequences, we reset the SSM's state to block the model from mixing unrelated samples. We do this by creating a "reset mask" that marks the start of each new sample. At these marks, we reset the state to 0. To manage the reverse flow of information in our SSMs, we re-use the same reset mask used for sequence packing to control the state information flow, in both reverse and forward directions. Extending on the example given above, the reset mask is $\{1, 0, 0, 2, 2, 2\}$, where the value '2' indicates positions where the reverse state components are forcibly reset to enforce causality in the suffix region, and 1 represents where to reset the state as a new sample begins.

State Partitioning and Concatenation: In our SSMs, we then partition the state dimensions into forward and reverse components. Let f_t represent the state at time step t, with a total dimension of D_{state}. We split f_t into two halves, shown below using NumPy's syntax:

$$f_t^{\text{forward}} = f_t[..., : D_{\text{state}}/2]$$
$$f_t^{\text{reverse}} = f_t[..., D_{\text{state}}/2 :]$$

Similarly, we split the input x_t into forward and reverse components:

$$x_t^{\text{forward}} = x_t[..., : D_{\text{state}}/2]$$
$$x_t^{\text{reverse}} = x_t[..., D_{\text{state}}/2 :]$$

We then apply the reset mask to the forward and reverse state components as shown below, to prevent backward information flow and prevent inter-sample state interference:

$$f_t^{\text{forward}} = \begin{cases} f_t^{\text{forward}} & \text{if reset\_mask}[t] \neq 1 \\ 0 & \text{if reset\_mask}[t] = 1 \end{cases}$$

$$f_t^{\text{reverse}} = \begin{cases} f_t^{\text{reverse}} & \text{if reset\_mask}[t] \neq 2 \\ 0 & \text{if reset\_mask}[t] = 2 \end{cases}$$

With the masked reverse state components, we proceed to compute the forward and reverse recurrent states using the recurrence functions:

$$h_t^{\text{forward}} = \text{RecurrenceForward}(f_t^{\text{forward}}, x_t^{\text{forward}})$$
$$h_t^{\text{reverse}} = \text{RecurrenceReverse}(f_t^{\text{reverse}}, x_t^{\text{reverse}})$$

After processing the recurrences, we concatenate the forward and reverse recurrent states along the state dimension to obtain the complete state at time step t:

$$h_t = [h_t^{\text{forward}} \oplus h_t^{\text{reverse}}]$$

As we segmented the state into forward and reverse portions earlier, this final, concatenated h_t is equivalently sized to a state that would have resulted from using the same SSM fully causally, allowing our parameter count to remain the same. Additionally, since an equal number of state dimensions are traveling through the sequence, this state segmentation also allows us to compute match our the causal models. Empirically, we find this bidirectional approach provides benefits even when compute and parameter matched.

State Utilization: By utilizing the reset mask to partitioning the state in this manner, we ensure that bidirectionality is available in the prefix region while maintaining causality in the suffix region, as well as preventing interference among sequence packed samples. This bidirectional encoding of the input sequence can enhance the ability of the SSMs to handle varied inputs without violating the causal constraints necessary for generation, with only a minor reduction in state components traveling forwards. In contrast to an Encoder-Decoder setup, which restricts bidirectional layers to only process tokens in the prefix area, in our bidirectional layers, f_t^{forward} runs along the entire sequence, just as a standard causal recurrent layer does.

A.4 Bidirectional Python Implementation Example:

We provide an efficient implementation of our approach on our Github page in PyTorch and JAX at https://www.github.com/samblouir/birdie. To further clarify our approach, consider the following pseudocode using Python:

```
Prefix language Modeling example:

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

Assuming masked sequence packing for
    efficient training, this approach is
    compute-matched with a causal scan
    operation.

Example:
# We want to prepare a reset_mask for a
# given input and label

Original inputs: [4, 5, 6]
Original labels: [7, 8, 9]

# We concatenate these for
# our decoder-only models.
Processed inputs: [4, 5, 6, 1, 7, 8]
Processed labels: [-, -, -, 7, 8, 9]
# (1 acts as the "begin generating" token.)

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

# (For reference, here is a
# standard Attention mask for
# a Prefix language Modeling Transformer.
# This designates the
# bidirectional/encoder area.)
Processed attn_mask: [1, 1, 1, 0, 0, 0]

# We now can use the reset_mask in our model.
```

```
Equivalent abbreviated SSM code:

# First, we partition f_t into two halves.
split_location = (state_size // 2)

# The shape of f_t is (batch size, length,
    state_dims)
f_t_forward = f_t[..., :split_location]
f_t_reverse = f_t[..., split_location:]

# The shapes of f_t_forward and f_t_reverse
    are:
#   (batch size, length, state_dims//2)

# We also split x_t (which is (i_t * z_t))
x_t_forward = x_t[..., :split_location]
x_t_reverse = x_t[..., split_location:]

# The shapes of x_t_forward and x_t_reverse
    are:
#   (batch size, length, state_dims//2)

# Now we use our reset_mask to mask
    f_t_reverse in causal areas.
f_t_reverse = np.where(
            reverse_mask == 2,
            0,
            f_t_reverse,
        )

# We can then run the recurrence as usual.
h_t_fwd = \
    recurrence_func(
        f_t_forward,
        x_t_forward,
    )
h_t_rev = \
    reverse_recurrence_func(
        f_t_reverse,
        x_t_reverse,
    )

# Finally, we concatenate along the last axis
h_t = concatenate(xs_fwd, xs_rev)
```

A.5 Selective Copying

A.5.1 Example Illustration

Consider the input sequence "ABCDEF". We use the following variables with randomly selected values:

- **Selected Span:** "DE"

- **Start Delimiter Length:** "2"

- **End Delimiter Length:** "1"

These arguments result in the following selected inputs and labels for the model:

Processed Input:

$$[\text{CONTEXT}] \ \text{ABCDEF} \ [\text{COPY}] \ [\text{START}] \ BC \ [\text{END}] \ F$$

Processed Label:

$$\text{DE} \ [\text{DONE}]$$

A.5.2 Detailed Instructions

To construct a Selective Copying instance involving a single span, follow the procedure outlined below:

1. **Sample Loading**: Load an input string from the dataset and tokenize it. This tokenized string is referred to as the "context." The model will extract one or more spans from this context.

2. **Span Selection**: Randomly select at least one contiguous span from the context, with a length between 4 to 32 tokens. If multiple spans are selected, ensure they do not overlap.

3. **Delimiter Identification**: For each selected span, randomly determine the lengths of the start and end delimiters (ranging from 1 to 8 tokens). Extract the specified number of tokens immediately before the span as the start delimiter and the specified number of tokens immediately after the span as the end delimiter.

4. **Formatting the Span and Delimiters**: Concatenate the delimiters with the following tokens:

   [START] <start_delim> [END] <end_delim>

   Prepend this sequence with the [COPY] token to indicate a copying task:

   [COPY] [START] <start_delim> [END] <end_delim>

5. **Concatenating the Context**: Prepend the context with the [CONTEXT] token:

   [CONTEXT] ABCDEF

   Combine this with the formatted delimiters either by prepending or appending:

   - **Prepend:**

     [COPY] [START] BC [END] F [CONTEXT] ABCDEF

   - **Append:**

     [CONTEXT] ABCDEF [COPY] [START] BC [END] F

6. **Sampling Strategy**: The control system, Birdie, determines the frequency of prepending or appending the delimiters, the number of spans to selectively copy, and sets the maximum length of the context.

A.5.3 Detailed Example

Let's revisit the sequence "ABCDEF" with the span "DE" selected:

- **Start Delimiter Length**: 2 tokens (BC)

- **End Delimiter Length**: 1 token (F)

Formatted Delimiters:

[COPY] [START] BC [END] F

Final Concatenated Input (Prepend Example):

[COPY] [START] BC [END] F [CONTEXT] ABCDEF

Final Concatenated Input (Append Example):

[CONTEXT] ABCDEF [COPY] [START] BC [END] F

A.6 UL2

UL2, introduced by Tay et al. (Tay et al., 2023b), is a *mixture-of-denoisers* (MoD) pre-training strategy originally developed and ablated for Transformer-based language models. Rather than relying solely on a causal (left-to-right) language modeling objective, UL2 interleaves multiple forms of denoising tasks, such as infilling and prefix language modeling, to produce more robust representations.

In its original formulation, UL2 labels each sequence with a "paradigm token" (e.g., "R" or "X") to indicate which denoising objective was used. Table 5 summarizes the full set of infilling and prefix modeling objectives in UL2, including their mean span widths and the percentage of tokens masked. These paradigm tokens were ablated off by continuing pre-training without them, for a significant number of steps. Garcia et al. (2023) published an updated and specialized variant for decoder-only models from the authors of UL2. This variant augments UL2 with a pure next token prediction objective (standard causal language modeling) and simplifies the amount of infilling configurations. Table 6 lists these modified denoisers, alongside their sampling ratios. In this variant, "R" denotes a short-span infilling objective, "X" denotes a more extreme infilling setup with higher mask percentages or larger spans, and "S" indicates a prefix language modeling task. Notably, the pure causal language modeling objective is included without a paradigm token.

Empirical results in both the original and specialized versions show that mixing multiple denoising tasks can outperform training solely on next token prediction for many downstream tasks. In cases where UL2 does not surpass standard causal modeling, the authors noted performance differences tend to be small (Tay et al., 2023b).

A.7 Denoising Objectives and Birdie Motivation

Tables 5 and 6 provide an overview of the denoising strategies used in UL2. The objective categories, "R," "X," "S," (as well as next token prediction), are used to group specific noising objective and configuration classes that are applied to input samples. For instance, "R" often denotes standard span infilling at normal masking probabilities, while "X" can span larger regions of text and/or mask up to 50% of tokens. In the case of "S" (prefix language modeling), the sequence is partially visible at the beginning, requiring the model to continue predicting from a given prefix.

In our approach with Birdie, we follow a similar paradigm-token strategy, but introduce two key modifications:

1. **Appending Paradigm Tokens** Instead of always prepending the paradigm token, Birdie appends it 50% of the time. This design choice is intended to familiarize the model with varied token placements and reduce reliance on position-based cues.

2. **Dynamic Mixture Ratios.** While UL2 maintains fixed ratios of denoising tasks throughout training, Birdie adaptively updates the ratio of objectives (e.g., infilling vs. prefix vs. causal language modeling) based on model feedback signals. This avoids exhaustive ablation studies and aims to optimize performance for different architectures and tasks.

These modifications address several limitations we observed when applying UL2 to *state-space models* (SSMs):

1. **Transformer-Centric Ablations.** UL2's reported ablations focus primarily on Transformers. Our experiments indicate that SSM-based models (e.g., Gated SSMs) respond differently to certain denoising objectives, necessitating an approach tailored to SSMs.

2. **Weak Retrieval/Copy Abilities.** Although UL2 excels on many downstream tasks, we found that SSMs trained with UL2 did not improve on exact-retrieval tasks (e.g., phone number copying) over a standard next-token predictor. This suggests the need for adaptive sampling of denoisers that more strongly emphasize copying skills for SSM architectures.

3. **Fixed Objective Ratios.** UL2 uses a constant mixture of objectives across the entire training process. Empirically, we observed that certain objectives (e.g., infilling) may require a higher sampling frequency for SSMs, whereas Transformers are able to perform this task relatively easy. Dynamic ratio adjustments in Birdie allow the model to re-balance objectives on the fly, rather than relying on predefined proportions.

Our approach is able to maintain the benefits of mixture-of-denoiser pre-training, while addressing

Paradigm Token	Mean Span Width	Masked Input %
"R" Denoisers	3 tokens	15%
	8 tokens	15%
"X" Denoisers	3 tokens	50%
	8 tokens	50%
	64 tokens	15%
	64 tokens	50%
"S" Denoisers	Prefix language Modeling, sequence split 75%	

Table 5: Summary of Denoising Objectives Used in UL2

Paradigm Token	Mean Span Width	Masked Input %
"R" dDenoiser	3 tokens	15%
"X" dDenoiser	32 tokens	50%
"S" dDenoiser	Prefix language Modeling, sequence split 75%	
-	Next Token Prediction (also referred to as Causal language Modeling)(	

Table 6: Summary of Denoising Objectives used in a specialized variant of UL2 for decoder-only models, as reported in (Garcia et al., 2023). The objectives are sampled at a ratio of 20% prefix language modeling, 60% next token prediction, and 20% infilling. Presumably, this translates to a 10% sampling probability of the "R" dDenoiser and a 10% sampling probability for the "S" denoiser. **Note:** a denoiser paradigm token was not provided for the next token prediction task, so. Thus, we do not include one here. More details are provided in subsection A.6

our practical compute constraints and unique needs
posed by SSMs.

A.8 Hyperparameters

Model	HD	State	Block	MLP Exp.	Attn.	1D Conv	Bidir.
Gated SSM	2048	2560	48*	–	–	–	Repeating: (50% bidir., causal)
Hawk	2048	2560	20	$\frac{8}{3} \cdot 2048$	–	Size 4	Repeating: (50% bidir., causal)
Transformer	2048	–	24		MHA 16H/128D	–	Every layer

* Since Gated SSM uses a fused recurrence and MLP layer, similar parameter count is maintained.

Table 7: Comparison of Model Characteristics

SSMs did not have weight decay applied to their W^f weights, and Hawk does not have any weight decay applied to its RG-LRU parameters or biases.

A.9 Phone Number Task

Hyperparameters Models are fine-tuned for 1000 steps with no weight decay and a batch size of 64. Training samples range from 200 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. Hawk and Transformer models are trained with a fixed learning rate of 5×10^{-5}.

Worst Baselines Phonebook Sweep In an unsuccessful attempt to improve the non-converging models (Next Token Prediction, UL2, and Fixed Ratio Mixture), we ran extensive hyperparameter sweeps. Our best settings were a 0.0 global weight decay, 1e-5 learning rate, and a batch size of 64. Many settings achieved similar results, but others resulted in the accuracy collapsing to near 0%. Specifically, we tried the power set of the following hyperparameters:

- **Global weight decay:** 0.0, 0.01, 0.1

- **Learning rate:** $1 \times 10^{-7}, 5 \times 10^{-7}, 1 \times 10^{-6}, 5 \times 10^{-6}, 1 \times 10^{-5}, 5 \times 10^{-5}, 1 \times 10^{-4}, 5 \times 10^{-4}, 1 \times 10^{-3}$

- **Batch size:** 16, 32, 64, 128, 256

A.10 Birdie pre-training Metrics

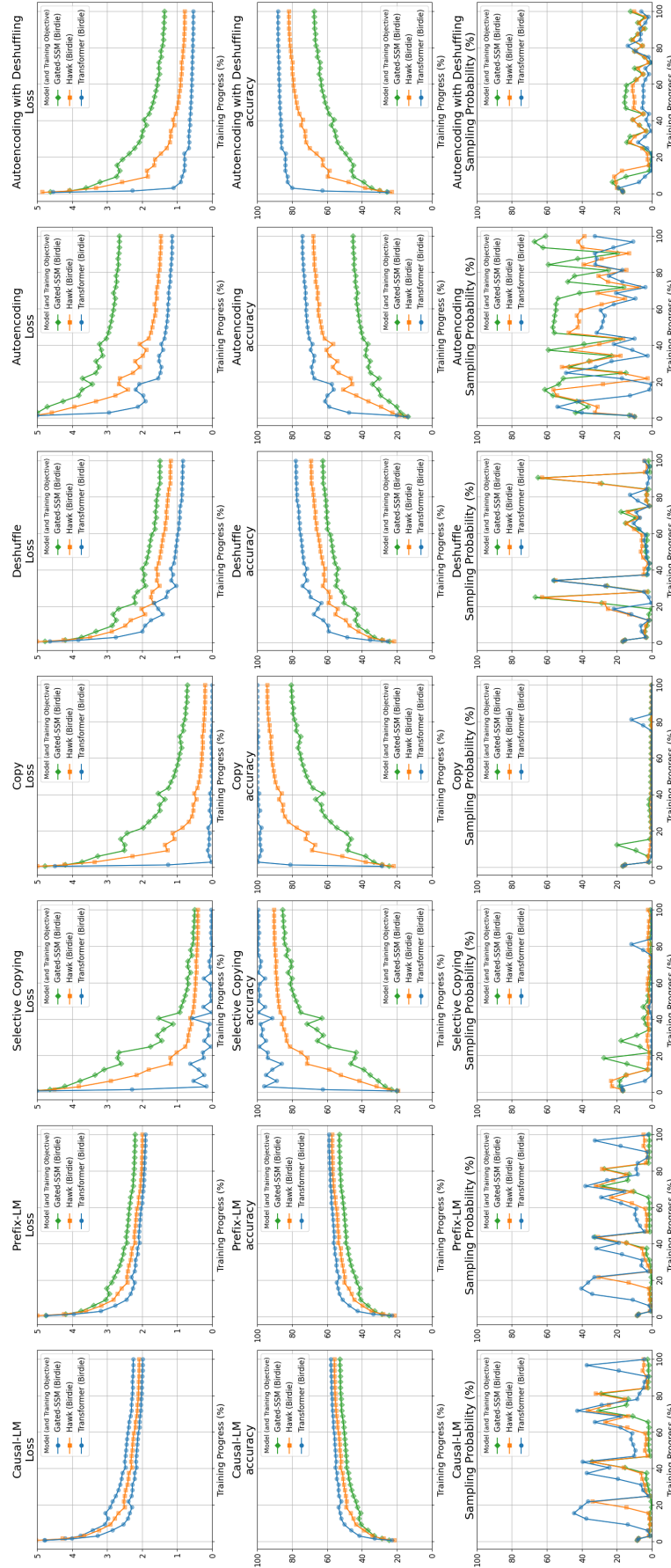Figure 5: These plots show how several metrics evolve during training. Loss and Accuracy are on Validation data from The Pile. Accuracy denotes greedy decoding accuracy. Sampling Probability (%) denotes the probability that an objective in a class is selected for each segmented sample from the training dataloader, as selected by Birdie. The parameterizations for each objective are described in Appendix Section A.2.

A.11 Pre-training

We train all models on the same data pipeline using The Pile (Gao et al., 2020)[5]. 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, we add spacing between samples to prevent the Conv1D layer from leaking information between samples. 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 are 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 8. 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., 2023b). 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 - specific gradient clipping on Beta (shown visible in appendix figure A.16.2), no weight decay on RG-LRU layers, and keep parameters stored in bfloat16 precision. All others use float32, though we always cast intermediates to bfloat16 except when running the recurrence functions for the Gated SSM and

Hawk. Our Transformer baselines use a 250k decay rate for their RoPE positional encodings, following a suggestion on a Reddit post (Rozière et al., 2024b).

A.12 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., 2023b) 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 use a different learning rate to compensate for the lack of AdaFactor's parameter-scaled updates. We use a gentle 3e-4 peak cosine LR that decays down to 3e-5, similar to work 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. We finish instruction tuning by again resetting the optimizer state, and using a $3e-5$ to $3e-6$ cosine schedule over two epochs on the Open-Hermes dataset (Teknium, 2023). During this final phase, we extend the sequence length to 8192, although the longest sample in Open-Hermes is only around 6,000 tokens long.

---

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

A.13 Hardware and Experimental Setup

We train 11 models, each containing 1.4 billion parameters. The primary training is conducted over 5 machines, each equipped with 4 Nvidia A100 GPUs (80GB). Additionally, fine-tuning and evaluation was split among four NVIDIA H100 GPUs, five Google TPUv3-8, a TPUv4-32 slice, generously provided through Google's TPU Research Cloud, for which we are sincerely grateful. The fixed ratios of the Fixed Ratio Mixture was found by training small 110M Gated SSM and Transformers models with random mixtures and hand-tuning sampling rates to ensure no objective. This took over 50 iterations of training the 110M model, which took roughly 5 hours each.

Table 8 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	Gated SSM	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	Hawk	6,480	4.05	N/A	13,148
JAX	Transformer	10,016	6.3	2048	8,506

Table 8: Comparison of observed model training speeds on our multi-node A100 setup.

A.14 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., 2021b).
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).
SST-2	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 9 — Instruct Models / Base Models

Model	Procedure	Average	ARC Challenge	ARC Easy	BoolQ	COPA	HellaSwag	LogiQA	MathQA	MC-TACO	MedMCQA	MMLU	MNLI	OpenBookQA	PIQA	PubMedQA	QA4MRE	QNLI	race	SciQ	SST-2	TruthfulQA (MC1)	WiC	Winogrande
Instruct Models																								
Hawk	Birdie	41.4%	25.4%	31.8%	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%
Hawk	Birdie - Causal	40.8%	26.3%	32.3%	52.9%	45.4%	30.6%	31.3%	25.8%	55.4%	28.3%	22.9%	31.8%	31.2%	63.5%	48.6%	28.5%	55.2%	27.7%	43.8%	81.5%	30.8%	53.6%	49.9%
Hawk	Next Token Pred	39.6%	25.7%	32.5%	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%
Attention	Birdie	39.7%	25.3%	30.7%	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%
Attention	Next Token Pred	40.4%	26.5%	33.0%	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%
Base Models																								
Hawk	Birdie	38.3%	25.9%	33.2%	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%
Hawk	Birdie - Causal	39.0%	25.0%	34.6%	49.3%	45.9%	32.7%	30.0%	25.6%	40.9%	24.3%	26.0%	31.8%	29.4%	63.3%	48.2%	24.8%	49.4%	28.1%	60.0%	61.0%	28.0%	49.8%	50.5%
Hawk	Next Token Pred	39.1%	25.4%	34.6%	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%
Attention	Birdie	38.5%	23.0%	28.8%	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%
Attention	Next Token Pred	39.9%	26.0%	35.0%	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%

Table 9: Hawk and Transformer model performance comparisons for Instruct Models and Base Models.

Table 10 — Instruct Models / Base Models

Model	Procedure	Average	ARC Challenge	ARC Easy	BoolQ	COPA	HellaSwag	LogiQA	MathQA	MC-TACO	MedMCQA	MMLU	MNLI	OpenBookQA	PIQA	PubMedQA	QA4MRE	QNLI	race	SciQ	SST-2	TruthfulQA (MC1)	WiC	Winogrande
Instruct Models																								
Gated SSM	Birdie	41.2%	25.6%	31.4%	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%
Gated SSM	Birdie - Causal	41.1%	25.9%	32.4%	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%
Gated SSM	Fixed Ratio Mix	38.8%	25.7%	32.7%	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%
Gated SSM	Next Token Pred	38.7%	25.6%	32.4%	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%
Gated SSM	UL2	39.7%	25.3%	31.7%	52.5%	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%
Base Models																								
Gated SSM	Birdie	37.2%	25.0%	30.2%	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%
Gated SSM	Birdie - Causal	37.5%	24.7%	27.9%	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%
Gated SSM	Fixed Ratio Mix	38.9%	25.1%	30.9%	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%
Gated SSM	Next Token Pred	39.1%	25.5%	34.1%	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%
Gated SSM	UL2	38.0%	23.6%	33.9%	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%

Table 10: Gated SSM training procedure comparisons performance comparisons for Instruct Models and Base Models.

A.15 SQuAD V2: Question and Answering

Task Description and Setup We evaluate our instruction-tuned models on SQuAD V2, a question-answering dataset where models are provided with a Wikipedia excerpt and asked a question. Some questions have multiple acceptable answers, while others are intentionally unanswerable. Following previous work (Jelassi et al., 2024), we focus exclusively on answerable questions and do not fine-tune our models on this task.

The standard metric for SQuAD V2 (F1) penalizes models for verbosity by reducing scores when additional words are present. Since our models are not trained to prioritize brevity, and SQuAD V2 predates today's conversational language models, we place greater emphasis on the "Answer Contains Label" metric. This metric awards full credit if any acceptable answer fully appears in the model's response, whereas the F1 score grants partial credit for matching words but penalizes longer responses.

Model Tag	Training Procedure	F1 (%)	Answer Contains Label (%)
Transformer	**Birdie**	21.4	**73.7**
Transformer	Next Token Prediction	21.0	60.9
Hawk	**Birdie**	**23.2**	54.4
Hawk	Next Token Prediction	9.1	15.7

Table 11: Averaged SQuAD V2 results with instruction-tuned models. Training with the Birdie procedure strongly improves SSM performance, compared to Next Token Prediction. The best performing models and metrics are shown in bold. These results are plotted by sequence length in Figure 3

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

Table 12: Averaged SQuAD V2 Question-Answering 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.

A.16 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	Fixed Ratio Mixture	36.2%
Gated SSM	UL2	34.7%
Gated SSM	Birdie - Causal	33.9%
Gated SSM	Next Token Prediction	32.2%
Base Models		
Gated SSM	Birdie	36.8%
Gated SSM	Birdie - Causal	34.7%
Gated SSM	UL2	31.7%
Gated SSM	Fixed Ratio Mixture	29.6%
Gated SSM	Next Token Prediction	27.5%

Table 13: 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 three figures for the infilling task. Figure 6 shows a shorter entry in the dataset, and Figure 7 shows the longest entry from our new infilling dataset. Finally, we show the prompt given to language models to judge the validity of questions and labels.

```
                                      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 6: A short example from our new infilling task. For more details, see appendix section A.16.

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 7: A long example from our new infilling task. For more details, see appendix section A.16.

```
We are making a dataset. Please help us determine if the possible label is the best
    available choice from the given options.
Given a sequence of events with a missing entry, models are supposed to predict the
    best choice to fill in the blank line.

Here is the sequence of events:
1. A group of scientists embark on a mission to explore the outer edges of the
    universe.
2. (blank)
3. Their discoveries revolutionize our understanding of cosmic phenomena.

Here are the possible choices:
(A) The scientists fail to return to Earth due to an unknown celestial event.
(B) They discover a new planet that can sustain human life.
(C) They encounter another species of intelligent beings from beyond our galaxy.
(D) They uncover the secrets behind black holes and how they function.

Is "(D) They uncover the secrets behind black holes and how they function." the
    best choice out of the above for replacing the blank line?

Reply immediately with yes or no.
```

Figure 8: The prompt used to ask models if the question and label are suitable for the infilling task. For more details, see appendix section A.16.

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

A.16.1 Gated SSM Implementation

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

class GatedLinearRNN(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)


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

A.16.2 Hawk Implementation

```python
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
```