RWKV-7 "Goose" with Expressive Dynamic State Evolution

Bo Peng[1,2,*], **Ruichong Zhang**[3,*], **Daniel Goldstein**[2,4,*],
Eric Alcaide[2,5], **Xingjian Du**[6], **Haowen Hou**[7], **Jiaju Lin**[8], **Jiaxing Liu**[9], **Janna Lu**[4,10], **William Merrill**[11], **Guangyu Song**[2,12], **Kaifeng Tan**[13], **Saiteja Utpala**[2], **Nathan Wilce**[2,4], **Johan S. Wind**[14], **Tianyi Wu**[15], **Daniel Wuttke**[2,16], and **Christian Zhou-Zheng**[2]

[1]RWKV Project (under Linux Foundation AI & Data), [2]EleutherAI, [3]Tsinghua University, [4]Recursal AI, [5]Dalle Molle Institute for Artificial Intelligence USI-SUPSI, [6]University of Rochester, [7]Guangdong Laboratory of Artificial Intelligence and Digital Economy (SZ), [8]Pennsylvania State University, [9]Zhejiang University, [10]George Mason University, [11]New York University, [12]Tano Labs, [13]Shenzhen University, [14]University of Oslo, [15]Beijing Normal University, [16]Denigma

Abstract

We present RWKV-7 "Goose", a new sequence modeling architecture with constant memory usage and constant inference time per token. Despite being trained on dramatically fewer tokens than other top models, our 2.9 billion parameter language model achieves a new 3B SoTA on multilingual tasks and matches the current 3B SoTA on English language downstream performance. RWKV-7 introduces a newly generalized formulation of the delta rule with vector-valued gating and in-context learning rates, as well as a relaxed value replacement rule. We show that RWKV-7 can perform state tracking and recognize all regular languages, while retaining parallelizability of training. This exceeds the capabilities of Transformers under standard complexity conjectures, which are limited to TC^0. To demonstrate RWKV-7's language modeling capability, we also present an extended open source 3.1 trillion token multilingual corpus, and train four RWKV-7 models ranging from 0.19 billion to 2.9 billion parameters on this dataset.

To foster openness, reproduction, and adoption, we release our models[1] and dataset component listing[2] on Hugging Face, and our training and inference code[3] on GitHub; all under the Apache 2.0 License.

---

[*]Equal first authorship. Others listed alphabetically.

[1]Model weights at `https://huggingface.co/RWKV`

[2]Dataset components listed at `https://huggingface.co/RWKV`

[3]Source code at: `https://github.com/RWKV/RWKV-LM`

Contents

Name	State Evolution	Scalars	LS	FD	DD	GE
RWKV-4	$s_t = e^{-w} \odot s_{t-1} + e^{k_t} \odot v_t;$ $s'_t = e^{-w} \odot s'_{t-1} + e^{k_t}$		✗	✓	✗	✗
RetNet	$S_t = w S_{t-1} + v_t^T k_t$	w	✓	✗	✗	✗
RWKV-5	$S_t = S_{t-1}\text{diag}(w) + v_t^T k_t$		✓	✓	✗	✗
Mamba	$S_t = S_{t-1} \odot \exp(-(w_t^T 1) \odot \exp(A)) + (w_t \odot v_t)^T k_t$		✓	✓	✓	✗
RWKV-6 & GLA	$S_t = S_{t-1}\text{diag}(w_t) + v_t^T k_t$		✓	✓	✓	✗
HGRN-2	$S_t = S_{t-1}\text{diag}(w_t) + v_t^T(1 - w_t)$		✓	✓	✓	✗
Mamba-2	$S_t = w_t S_{t-1} + v_t^T k_t$	w_t	✓	✗	✓	✗
TTT [a]	$S_t = S_{t-1} - a_t \nabla l(S_{t-1}, k_t, v_t)$	a	✓	✗	✗	✓
Longhorn	$S_t = S_{t-1} \odot (I - a_t^T k_t^2) + (a_t x_t)^T k_t$		✓	✓	✓	✗
Gated DeltaNet	$S_t = w_t S_{t-1}(I - a_t k_t^T k_t) + a_t v_t^T k_t$	w_t, a_t	✓	✗	✓	✓
Titans [a]	$M_t = (1 - \alpha_t)M_{t-1} + S_t$ $S_t = w_t S_{t-1} - a_t \nabla l(M_{t-1}, k_t, v_t)$	w_t, a_t	✓	✗	✓	✓
Generalized Δ Rule	$S_t = S_{t-1}(\text{diag}(w_t) + z_t^T b_t) + v_t^T k_t$		✓	✓	✓	✓
RWKV-7 (ours)	$S_t = S_{t-1}(\text{diag}(w_t) - \hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t)) + v_t^T k_t$		✓	✓	✓	✓

Table 1: Recent RNN architectures used for language modeling.
LS (Large State): matrix-valued states, or state size at least 4 times larger than the model dimension.
FD (Flexible Decay): the dimension of the decay term w or w_t is not smaller than the model dimension.
DD (Dynamic Dependence): the decay term w_t is a function over the input x_t.
GE (Generalized Eigenvalue): evolution matrix admits eigenvalues outside of the interval $[0, 1]$.
[a] Shown with mini batch size 1 for simplicity.

1 Introduction

Autoregressive Transformers (Vaswani et al., 2023) have recently dominated sequence modeling tasks, enjoying excellent in-context processing and highly parallelizable training due to their use of softmax attention. However, softmax attention incurs quadratic computational complexity and memory usage with respect to sequence length due to its linearly expanding key-value cache. For short sequences, much of this cost can be covered by modern GPU parallelism techniques, but Transformer inference becomes increasingly costly as sequence lengths grow.

This limitation has inspired significant research into the design of recurrent neural network (RNN) architectures with compressive states that afford linear computational complexity and constant memory usage, while still allowing highly parallel training. Two of the most commonly proposed alternatives that satisfy these requirements are linear attention variant models (Katharopoulos et al., 2020b; Sun et al., 2023; Peng et al., 2024b; Yang et al., 2023a) and State Space Models (Gu & Dao, 2023). These architectures have grown more sophisticated, with many recent proposals incorporating some form of the delta rule, as embodied by parallelized DeltaNet (Schlag et al., 2021; Yang et al., 2024c). Such models have achieved impressive downstream performance results: since RWKV-4 (Peng et al., 2023), RNN models have shown increasing potential to rival Transformers when given equivalent model size and training compute, while dramatically reducing inference costs.

We present a new architecture, RWKV-7 "Goose", which generalizes the delta rule for use in sequence modeling. First, we add a vector-valued state gating mechanism, enhancing expressivity and providing implicit positional encoding. Second, we expand the in-context learning rate from a scalar to become vector-valued, allowing the model to selectively replace state data on a channel-wise basis. Third, we decouple the keys at which the delta rule removes from and adds to the state. Finally, we place these innovations within a modified RWKV-6 architecture, inheriting important features such as token-shift, bonus, and a ReLU^2 feedforward network. We also introduce an expanded 3.1 trillion token RWKV World v3 corpus designed for enhanced English, code, and multilingual task performance. We use this architecture and corpus to train new state-of-the-art open-source language models, upgraded from preexisting RWKV-5/RWKV-6 checkpoints.

Our main contributions are as follows:

- The **RWKV-7 "Goose" architecture**, which dramatically improves downstream benchmark performance over RWKV-6 and demonstrates state-of-the-art multilingual performance at 3B scale and near SoTA English language performance, despite being trained on many fewer tokens than the top models in its class.
- The **RWKV World v3 public dataset**, comprised of 3.1 trillion tokens of publicly available multilingual data.
- Public release of four **pre-trained RWKV-7 World v3 language models**, ranging from 0.19 to 2.9 billion parameters trained on 1.6 to 5.6 trillion tokens.
- Public release of three **pre-trained RWKV-7 Pile language models**, using the GPT-NeoX tokenizer (Black et al., 2022), ranging from 0.17 to 1.47 billion parameters, useful for comparative study with other architectures.
- **Proofs** that the generalized delta rule employed in RWKV-7 can **solve problems outside of** TC^0 under the widely held complexity conjecture that $\mathsf{TC}^0 \neq \mathsf{NC}^1$. This includes solving an S_5 state tracking problem known to be in NC^1 using only a single layer, and recognizing all regular languages using only a constant number of layers.
- A **method for upgrading the RWKV architecture without pre-training from scratch**, producing increasingly competitive trained models at reduced computational expense.

Larger datasets and RWKV-7 models are under active preparation and construction and will be released under the Apache 2 license whenever practical.

2   Background

Linear attention's major advantage over softmax attention is that it can be formulated as a RNN with constant running time per token and constant memory usage (Katharopoulos et al., 2020a), while softmax attention takes $O(N)$ time per token and $O(N)$ memory with regard to sequence length. Despite this dramatic efficiency improvement, linear attention has its own significant drawbacks (Schlag et al., 2021; Han et al., 2024; Fan et al., 2025).

One such issue is that linear attention numerically adds to the fixed-size state at every time-step: older state contents are never removed, only reduced by becoming a smaller proportion of the numerically increasing state. Due to limitations on the state size, eventually such a system must mix values together and muddy the outputs retrieved for a given key (Schlag et al., 2021; Yang et al., 2024b). Modern linear attention architectures like RWKV-6 (Peng et al., 2024b), RetNet (Sun et al., 2023), Gated Linear Attention (Yang et al., 2023a), and Mamba 2 (Dao & Gu, 2024) use per time-step decay to remove some portion of such older values from the state in a data-dependent manner. However, decay is a blunt tool that cannot remove only the values stored at specific keys.

Delta Rule.   DeltaNet (Schlag et al., 2021) sidesteps the problem of numerically increasing state by partially replacing the value stored at the current key with the same amount of a new value, allowing the model to both take away old memories and add new ones on a per-key basis. It reformulates the state update as an explicit online learning problem where the goal is to retrieve the correct value as output for a given key as input. DeltaNet was the first to apply the foundational Error Correcting Delta Rule (Widrow et al., 1960) to key-value compressive states, akin to those stored in the RNN formulation of linear attention. This update rule is equivalent to a single step of stochastic gradient descent, training the state S_t at test time to output the desired values v_t for the keys k_t as inputs using loss $\mathscr{L} = \frac{1}{2}\|(S_t k_t - v_t)\|^2$ and gradient $\frac{\partial \mathscr{L}}{\partial S} = S k^\top k - v^\top k$, leading to a recurrent update formula of $S_t = S_{t-1}(I - a k_t^T k_t) + a v_t^T k_t$, where a is a scalar learning rate. The ideas behind this internal state update can be traced back to fast weights (Schmidhuber, 1992).

There has been significant recent interest in improvements to DeltaNet, in order to bring its efficiency and downstream performance in line with Transformers while still capturing the speed and memory benefits of Linear Attention. Parallelizing DeltaNet (Yang et al., 2024c) showed that DeltaNet used diagonal plus low-rank (DPLR) state evolution like S4 (Gu et al., 2022), and could be parallelized across the time dimension, creating a path to efficiently train such models. Our work further extends that parallelization to cover the generalized delta rule formulation introduced herein, as well as the specific formula of RWKV-7.

Concurrent Work. Concurrent work with our own has focused on architectural improvements beyond DeltaNet while still using the delta rule or variations thereof. Longhorn (Liu et al., 2024) employs an update rule that approximates a closed-form solution to a globally optimal update objective, applied on an otherwise unchanged Mamba architecture. Gated Delta Networks (Yang et al., 2024a) applies gating to the DeltaNet state, essentially multiplying the transition matrix by a data-dependent scalar per head. This combines the DeltaNet update rule with the scalar decay found in some modern RNNs like RetNet and Mamba-2. The delta rule gradient descent formula with dynamic weight decay w_t and learning rate a_t becomes $S_t = S_{t-1} \left(\text{diag}(w_t) - k_t^\top k_t \, \text{diag}(a_t) \right) + v_t^\top k_t \, \text{diag}(a_t)$.

TTT (Test-Time Training) (Sun et al., 2024) and Titans (Behrouz et al., 2024) also both apply scalar decay, but eschew per-step gradient descent update rules in favor of a batched multi-timestep approach. Titans also adds momentum to the otherwise classical SGD update applied to the state.

Another concurrent work with our own, Unlocking State-Tracking in Linear RNNs Through Negative Eigenvalues (Grazzi et al., 2024), has demonstrated the potential for increased expressiveness that comes from allowing the state transition matrix to contain negative eigenvalues. We show a result significantly beyond this, proving that RWKV-7 and our generalized delta rule can recognize all regular languages using only a small constant number of layers.[4]

3 Architecture

Unlike the other work described above, RWKV-7 generalizes the delta update rule into an extended formula $S_t = S_{t-1}(\text{diag}(w_t) + z_t^T b_t) + v_t^T k_t$ to increase expressivity (see Table 1). This is still a diagonal plus rank one update rule, which admits efficient forms of parallelization (Yang et al., 2024c). Here, w_t is a more expressive data-dependent vector-valued decay, unlike the scalar decays featured in the other works previously described. Our use of z_t and b_t in this extended formula permits a flexible approach to state update, while retaining the important reduction in non-SRAM memory bandwidth usage that comes from using small data-dependent vectors instead of large matrices. One example of this flexibility is the ability to use a different removal key than replacement key. This extended delta rule is flexible enough that there may be other useful formulations that fit within it beyond the formulas we have chosen for calculating z_t and b_t in RWKV-7.

The original delta rule allowed a fixed scalar fraction of a value to be replaced from the state via its in-context learning rate parameter. RWKV-7's extended delta rule instead replaces data-dependent vector-valued amounts of the state, allowing each key channel in the state to vary independently. We parameterize $z_t = -\hat{\kappa}_t$ and $b_t = \hat{\kappa}_t \odot a_t$ where a_t has elements in $(0, 1)$. This keeps the update stable (see Appendix C), while maintaining increased expressivity. We demonstrate the improved performance of these design choices via ablations in Appendix K.

Previous research (Merrill et al., 2024) pointed out that Transformers and RNNs with a diagonal transition matrix could only represent functions in TC^0. RWKV-7, however, has a non-diagonal and input-dependent transition matrix, allowing it to represent more complex functions than its predecessors. In fact, we demonstrate that RWKV-7 possesses expressive power surpassing that of TC^0 under standard complexity conjectures and can recognize all regular languages. One new component of this power, not present in the original delta rule, is the ability to represent the "copy" state transition (Lemma 3). This is a key element in our proof that RWKV-7 can recognize all regular languages with a constant number of layers. See Appendix D for proof and details.

We replace the main RWKV-6 (Peng et al., 2024b) diagonal transition matrix with our extended delta rule and make several other changes to the RWKV architecture, observing significant modeling improvements. These include updates to the channel mixing module and the token shift module. We remove the data dependency of token-shift and the receptance gating of channel mixing, both of which contribute to faster training and inference. We increase the use of low-rank projections to generate more of our intermediate calculations, striking a balance between the total number of model parameters, training and inference speed, and downstream performance.

---

[4]For now, we choose to allow only part of the range of possible negative eigenvalues in our pre-trained large language models due to experimentally observed training instabilities.

Figure 1 presents the overall architecture of RWKV-7. Please refer to Appendix F for more details.
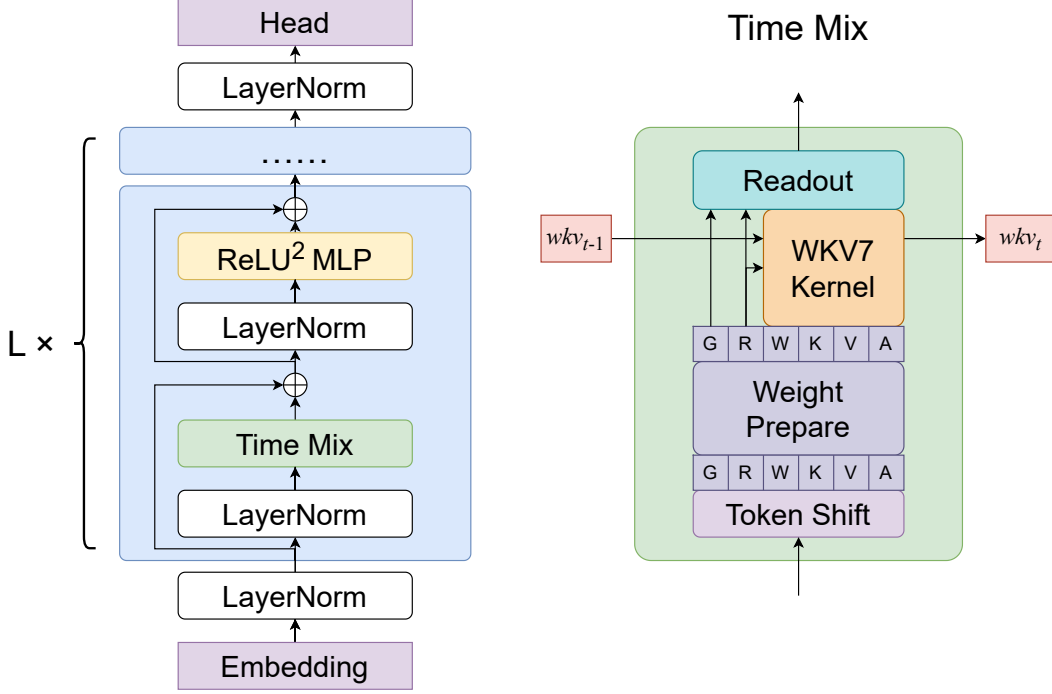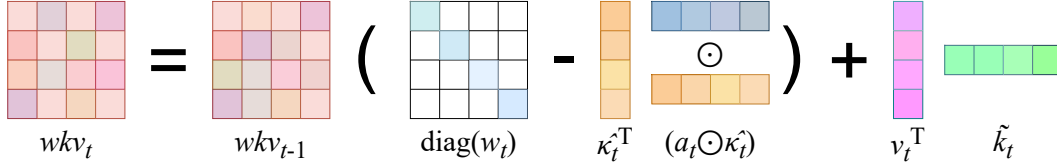

Figure 1: RWKV-7's overall architecture.

Figure 2: A simple illustration of the update mechanism of a single head of RWKV-7's state. Note that the actual state size is 64×64 per head, not 4×4.

4 Method

In this section, we use D to denote the model dimension. Bold capital letters represent trainable matrices, and vectors without a subscript t are trainable parameters. The first subscript denotes sequence position and second subscript denotes layer index, where necessary. We use the convention that all vectors are row vectors unless explicitly transposed, so all matrices operate on the right side, therefore $a^T b$ is an outer product and ab^T is an inner one. We use the square subscript to denote a placeholder for variable names and use the $\prod$ sign for cumulative matrix multiplication. See Appendix G for a pseudocode implementation of these formulas.

4.1 Time Mixing

Weight Preparation Along the lines of (Peng et al., 2024b), we introduce the following notation templates for common operators in the model, using the square subscript to denote a variable:

$$\mathrm{lerp}(a, b, x) = a + (b - a) \odot x, \tag{1}$$

$$\mathrm{loramlp}_\square(f, x, \mathrm{bias}) = f(x\boldsymbol{A}_\square)\boldsymbol{B}_\square + (\lambda_\square \text{ if bias else } 0), \tag{2}$$

Unless explicitly stated, all vectors appearing in this section are dimension D.

We extend the use of low-rank MLP (a 2-layer MLP with small hidden dimension compared to input and output), abbreviated as loramlp, to implement data dependency using minimal parameters.

The replacement key $\tilde{k}$, value v, decay w, removal key κ, in-context learning rate a, receptance r, and rwkv gate g parameters are computed as follows (outputs annotated with $\triangleright$):

$$x_t^\square = \text{lerp}(x_t, x_{t-1}, \mu_\square) \quad \square \in \{r, k, v, d, a, g\}, \qquad \text{token shifted inputs} \tag{3}$$

$$a_t = \text{sigmoid}(\text{loramlp}_a(\text{Identity}, x_t^a, \text{bias=True})), \qquad \triangleright\text{in-context learning rate} \tag{4}$$

$$k_t = x_t^k W_k, \qquad \text{key precursor} \tag{5}$$

$$\kappa_t = k_t \odot \xi, \qquad \triangleright\text{removal key} \tag{6}$$

$$\tilde{k}_t = k_t \odot \text{lerp}(1, a_t, \alpha), \qquad \triangleright\text{replacement key} \tag{7}$$

$$v_t = \text{sigmoid}(\text{loramlp}_v(\text{Identity}, x_t^v, \text{bias=True})), \qquad \text{value residual gate} \tag{8}$$

$$v'_{t,l} = x_t^v W_v, \qquad \text{value precursor} \tag{9}$$

$$v_t = \begin{cases} v'_{t,0}, & \text{layer } l = 0 \\ \text{lerp}(v'_{t,0}, v'_{t,l}, v_t), & \text{layer } l \geq 1 \end{cases}, \qquad \triangleright\text{value} \tag{10}$$

$$d_t = \text{loramlp}_d(\tanh, x_t^d, \text{bias=True}), \qquad \text{decay precursor} \tag{11}$$

$$w_t = \exp(-e^{-0.5}\text{sigmoid}(d_t)), \qquad \triangleright\text{decay} \tag{12}$$

$$r_t = x_t^r W_r, \qquad \triangleright\text{receptance} \tag{13}$$

$$g_t = \text{loramlp}_g(\text{sigmoid}, x_t^g, \text{bias=False}) \qquad \triangleright\text{rwkv gate} \tag{14}$$

ξ is a learned parameter representing the removal key multiplier, which transforms the original key into a version to be removed from the state. In practice, ξ lies in a range of approximately $[-5.3, 9.4]$.

α is a learned parameter representing the replacement rate booster, which adjusts the amount added back to the state after the transition matrix is applied.

Unlike r, k and v which are the main carriers of information, g, d, v and a act like gates which control the amount of information allowed to pass.

For comprehensive statistics of ξ, α and biases of d_t observed in the released RWKV-7 model, including extremum values, mean measurements, and distribution trends, see Appendix L.

For the computation of $x_t^\square$, we removed data dependency of linear interpolation from RWKV-6 to improve training speed.

We adapted the idea of Value Residual Learning Zhou et al. (2024) for the computation of v_t, which has shown to improve the final language modeling loss. v_t represents the value residual mix, which interpolates between the layer zero and current layer value precursors: $v_{t,0}$ and $v_{t,l}$.

We also updated the formula for computation of w_t, restricting all entries in $(\exp(-e^{-0.5}), 1)$ in favor of a smaller condition number for $\text{diag}(w_t)$, which maintains better training stability, and was beneficial to accuracy of the backward pass.

The $\tilde{k}_t$ in the formula can be regarded as a "normalized key", a design to ensure that the state of wkv contains columns of $O(1)$ size. Normally, we expect $\tilde{k}_t = k_t \odot (1 - w_t)$, as employed in RWKV-6c (see Appendix F), so that wkv_t rows are linear interpolations between wkv_{t-1} and $v_t^T k_t$ controlled by w_t. However, to further enhance expressivity, we decide to decouple w_t and a_t. We further decouple a_t from the amount actually added to the state, allowing the replacement rate booster α to interpolate the amount added between the normal in-context learning rate and 1.0. Importantly, all of these modifications operate on a per-channel basis. The numerical range of RWKV-7's wkv entries are generally stable as in RWKV-6c, unlike RWKV-6, where entries of the states can accumulate to thousands (see Appendix J for a state visualization).

The Weighted Key Value State Evolution   After weight preparation, we reshape $(r, w, \tilde{k}, v, \kappa, a)_t$, splitting them to h heads, with each head sized D/h. We always assume that h is a factor of D and heads are equally split. All operations in this section are shown per-head.

Before mixing in the time dimension, κ_t is normalized per head:

$$\hat{\kappa}_t = \kappa_t / \|\kappa_t\|_2 \tag{15}$$

The $\boldsymbol{wkv}$ (Weighted Key Value) is a multi-headed matrix-valued state of fast weights that undergoes dynamic evolution. The evolution of $\boldsymbol{wkv}$ is crucial for encoding context information by learning at test time to map keys to values. We start by defining the WKV time mixing as the recurrence relation

$$\boldsymbol{wkv}_0 = \boldsymbol{0}, \tag{16}$$

$$\boldsymbol{wkv}_t = \boldsymbol{wkv}_{t-1}\left(\mathrm{diag}(w_t) - \hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t)\right) + v_t^T \cdot \tilde{k}_t \tag{17}$$

Compared to RWKV-5 and RWKV-6, the $\boldsymbol{wkv}$ in this paper is transposed to ensure consistency with RWKV-7's code.

The $\boldsymbol{wkv}_t$ attention calculation can alternatively be written in a parallel manner:

$$\boldsymbol{wkv}_t = \sum_{i=1}^{t}\left(v_i^T \tilde{k}_i \prod_{j=i+1}^{t}\left(\mathrm{diag}(w_j) - \hat{\kappa}_j^T(a_j \odot \hat{\kappa}_j)\right)\right) \in \mathbb{R}^{(D/h)\times(D/h)} \tag{18}$$

The recurrent transition design has parallels with Schlag et al. (2021), but crucially the transition matrix

$$G_t = \mathrm{diag}(w_t) - \hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t) = \left(\boldsymbol{I} - \hat{\kappa}_t^T(\frac{a_t}{w_t} \odot \hat{\kappa}_t)\right)\mathrm{diag}(w_t) \approx \left(\boldsymbol{I} - 2\hat{\kappa}_t^T\hat{\kappa}_t\right)\mathrm{diag}(w_t) \tag{19}$$

is no longer a Householder matrix but a scaled approximation of it, as $\hat{\kappa}_t \neq \frac{a_t}{w_t}\hat{\kappa}_t$. This mimics a Householder matrix but with expanded dynamics, while still having all eigenvalues in a stable range of $[-1, 1]$ and allows the network to decay information in all subspaces if necessary. It contrasts with the case of a Householder-like matrix with learning rate $(I - av^T v)$, $a \in [0, 1]$, as used in Schlag et al. (2021); Yang et al. (2024c) where all eigenvalues are one except for the last one corresponding to $1 - a$. Given these properties, we refer to w_t as "in-context weight decay" and to a_t as "in-context learning rate" (ICLR). The RWKV-7 transition matrix, therefore, allows for both dynamic state evolution and approximation to a forget gate at the same time. See Appendix C for the details on the eigenvalue of the transition matrix, and when the transition matrix is guaranteed to be stable.

The original delta rule in Schlag et al. (2021) allows partial or full removal of pre-existing values from the state at each time-step, with the amount removed being equal to the scalar a. Our formulation extends this ability by making a a vector, allowing for different removal amount per state column.

WKV Bonus and Output   All operations in this section are shown per-head unless otherwise specified.

Receptance, which acts like the query found in transformers, is applied to the WKV state, and the result is normalized. An added bonus, the amount of which is weighted by ρ, allows the model to place extra attention on the current shifted input token without requiring it to store that token in the state.

$$u_t = \left(r_t \cdot (\rho \odot \tilde{k}_t)^T\right)v_t \qquad\qquad\qquad \text{bonus} \tag{20}$$

$$p_t = \mathrm{LayerNorm}(r_t \boldsymbol{wkv}_t^T) + u_t \qquad\qquad \triangleright\text{attention result} \tag{21}$$

Finally, the heads are recombined via reshaping so that $p_t \in \mathbb{R}^D$, gated, and transformed into the output as follows:

$$o_t = (g_t \odot p_t)\boldsymbol{W}_o \in \mathbb{R}^D \tag{22}$$

4.2   MLP

The MLP module of RWKV-7 is no longer identical to the Channel Mixing module of previous RWKV-4,5,6 architectures (Peng et al., 2024b). We remove the gating matrix $\boldsymbol{W}_r$, making it a two-layer MLP. In compensation for the removed gating parameters to satisfy the equi-parameter condition, we set the hidden dimension to be 4 times the size of model dimension.

$$k_t' = \mathrm{lerp}(x_t', x_{t-1}', \mu_k')\boldsymbol{W}_{k'} \in \mathbb{R}^{4D} \tag{23}$$

$$o_t' = \mathrm{ReLU}(k_t')^2\boldsymbol{W}_{v'} \in \mathbb{R}^D \tag{24}$$

5   RWKV World v3 Dataset

We train our models on the new **RWKV World v3 Dataset**, a new multilingual 3.119 trillion token dataset drawn from a wide variety of publicly available data sources. This dataset aims to help close the gap with the amount of data used to train modern LLMs, which may consume as many as 15 - 18 trillion tokens (Qwen et al., 2025; Grattafiori et al., 2024). We select the data to approximate the distribution of our previous World datasets, including English, multilingual, and code, while slightly enhancing Chinese novels.We describe the composition of our dataset in Appendix B.

6   Pre-Trained Models

We have pre-trained and publicly released seven Apache 2.0 licensed RWKV-7 models:

1. Trained on Pile: **RWKV7-Pile** of sizes **0.1B, 0.4B, and 1.4B**
2. Trained on RWKV World V3: **RWKV7-World-3** of sizes **0.1B, 0.4B, 1.5B, and 2.9B**

See Appendix E for detailed configurations.

The RWKV-7 Pile models all use the GPT-NeoX-20B tokenizer (Black et al., 2022), and were all trained from scratch on the Pile dataset, which has 332 billion tokens.

All RWKV World dataset models use the RWKV World Tokenizer. Due to compute budget constraints, the Goose World 3 0.1B and 0.4B models were trained from pre-existing RWKV-5 World v1 and v2 checkpoints, and the Goose World 3 1.5B and 2.9B models were trained from pre-existing RWKV-6 World v2.1 checkpoints. These checkpoints' parameters were then converted to the RWKV-7 format via a process described below. Once in the new format, the models are trained on either the additional full 3.1 trillion tokens of the World v3 corpus, or an equally weighted sub-sampling of it. Under this methodology, some documents were seen two or even three times.

The World v1, v2, v2.1, and v3 corpora contain 0.6, 1.1, 1.4, and 3.1 trillion tokens, respectively. The amounts of training in each stage at with each successive model architecture and corpus are shown in Table 2.

Model	World v1	World v2	World v2.1	World v3	Total
RWKV7-World3-0.1B	0.6 (RWKV-5)			1.0 (RWKV-7)	1.6
RWKV7-World3-0.4B		1.1 (RWKV-5)		2.0 (RWKV-7)	3.1
RWKV7-World3-1.5B		1.1 (RWKV-6)	1.4 (RWKV-6)	3.1 (RWKV-7)	5.6
RWKV7-World3-2.9B		1.1 (RWKV-6)	1.4 (RWKV-6)	3.1 (RWKV-7)	5.6

Table 2: Total trillions of tokens trained for all RWKV-7 World 3 models

Our model format conversion process involves removing the token-shift low-rank MLPs, rescaling by half the embeddings, wkv receptance, wkv output matrix weights, and Layernorm and Groupnorm bias values. Layernorm and Groupnorm weights are clamped above zero and square rooted. We widen the FFN MLP from 3.5x (in RWKV-6) to 4x and add new small (1×10^{-3}) uniform initalizations in the new regions, removing the RWKV-6 FFN receptance weights. We widen the time decay Low-rank MLP and add new small (1×10^{-4}) uniform initializations in the new regions. We replace the gate weights with a LoRA obtained through singular value decomposition and rescaling by half.

7   Language Modeling Experiments

7.1   LM Evaluation Harness Benchmarks

RWKV-7 models are evaluated on a series of common English-focused and multilingual benchmarks using *LM Evaluation Harness* (Gao et al., 2023) as shown in Tables 3 and 4. We benchmarked RWKV-7 along with several new open models which are state-of-the-art in their parameter count ranges. All numbers are evaluated under fp32 precision with lm-eval v0.4.8 using 0-shot, except for MMLU in which case 5-shot was used.

(a) FLOPS vs. Average Benchmark Accuracy   (b) Active Parameters vs. Average Benchmark Accuracy
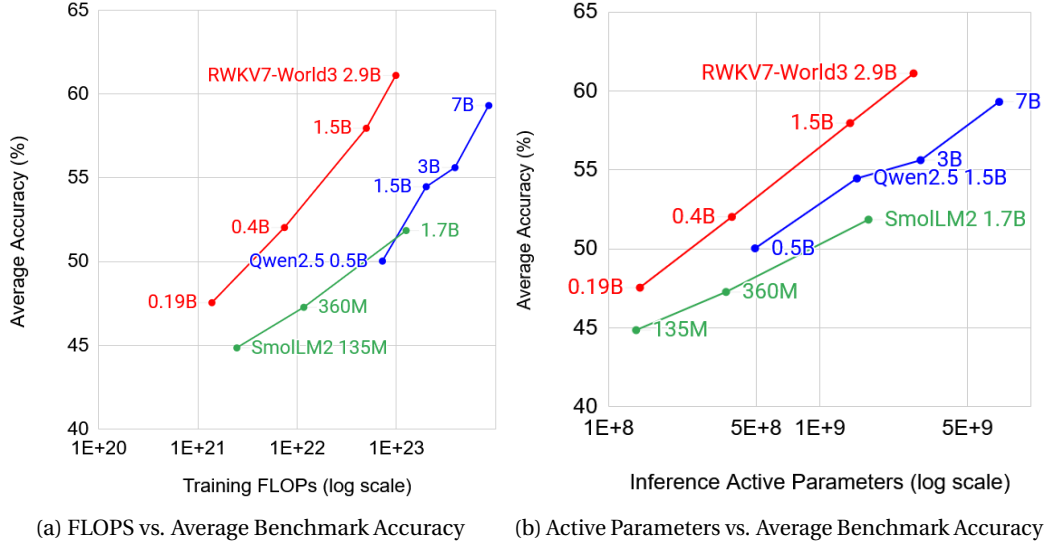
Figure 3: Model Comparisons across Multilingual Benchmarks

We find that RWKV-7 is generally able to match the English performance of Qwen2.5 (Qwen et al., 2025) with less than one third as many training tokens. Interestingly, we found that RWKV-7 models have shown giant leaps in MMLU performance compared to RWKV-6. We also find that RWKV-7-World models expand upon RWKV-6-World models already strong capabilities on multilingual benchmarks, outperforming SmolLM2 (Allal et al., 2025), Llama-3.2 (Grattafiori et al., 2024), and Qwen-2.5 (Qwen et al., 2025) by a significant margin.

In Figures 3a and 4a we plot FLOPs used to train several open models versus average accuracy across the same sets of common english and multi-lingual benchmarks. The multilingual evals show a very dramatic Pareto improvement versus the transformer models. Also note the similar english-language eval scores, but dramatically lower total FLOPs usage of RWKV7-World models versus other highly trained open transformer models. We theorize that if we were less constrained by compute and were able to train these models from scratch with the same amount of total tokens instead of from pre-trained checkpoints of earlier RWKV versions, the difference would be even more dramatic. Note that we did not plot the Llama 3.2 series of models, as they have no corresponding FLOPs amounts due to having been created via pruning and distillation from larger models.

7.2 Recent Internet Data Evaluation

Modern large language models are trained on massive datasets. Despite careful data cleaning, benchmark data leakage remains a challenge, compromising the validity of these evaluations. To complement traditional benchmarks, we evaluated RWKV-7 Goose and other leading open-source models using temporally novel internet data, generated after the models' training periods; this data could not have appeared in the training sets, removing data leakage concerns.

Specifically, we collected new data created after January 2025, including: newly submitted computer science and physics papers on arXiv, newly created Python/C++ open-source repositories on GitHub, recently published Wikipedia entries, new fiction on Archive of Our Own (Various, 2025), and recent news articles. Inspired by Delétang et al. (2024); Li et al. (2024b), we used compression rate as our evaluation metric. See Table 5 for details.

Remarkably, despite being trained on significantly less data than other top models, RWKV-7 Goose showed competitive performance on this temporally novel data.
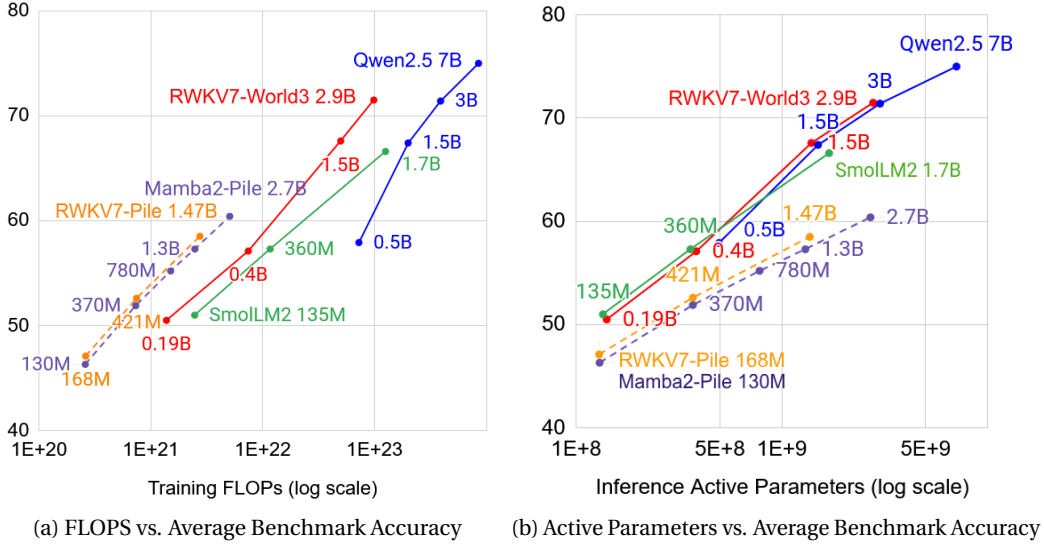
| (a) FLOPS vs. Average Benchmark Accuracy | (b) Active Parameters vs. Average Benchmark Accuracy |

Figure 4: Model Comparisons across English-Language Benchmarks

Model (Name)	Tokens (T)	lmb.o acc↑	hella acc_n↑	piqa acc↑	arcE acc↑	arcC acc↑	glue [a]acc↑	WG acc↑	sciq acc↑	mmlu acc↑	avg acc↑
RWKV5-World1-0.1B	0.6	38.4	31.9	61.4	44.2	19.9	45.5	52.9	76.3	23.1	43.7
SmolLM2-135M	2.0	42.9	**43.1**	**68.4**	**64.4**	**28.1**	**49.0**	**53.0**	84.0	**25.8**	**51.0**
RWKV7-World2.8-0.1B	1.6	**48.1**	42.1	67.3	59.3	25.5	48.1	52.7	**86.3**	25.4	50.5
RWKV5-World2-0.4B	1.1	54.0	40.9	66.5	54.0	24.0	50.0	53.2	86.9	23.8	50.4
SmolLM2-360M	4.0	53.8	56.4	72.1	70.4	36.5	**50.7**	59.0	91.2	26.3	57.4
Qwen2.5-0.5B	18.0	52.5	52.1	70.2	64.6	29.5	54.7	56.4	**93.1**	**47.8**	**57.9**
RWKV7-World2.9-0.4B	3.1	**58.6**	56.8	**72.9**	68.7	31.9	49.4	**59.9**	89.7	26.1	57.1
RWKV6-World2.1-1.6B	2.5	67.4	61.1	74.4	64.3	31.0	51.0	60.7	89.5	25.1	58.3
Llama3.2-1B	[b]15.0	63.0	63.7	74.5	65.5	31.3	49.7	60.7	91.4	32.1	59.1
SmolLM2-1.7B	11.0	67.7	**71.5**	**77.0**	77.7	**44.7**	51.5	66.1	93.3	50.3	66.6
Qwen2.5-1.5B	18.0	63.0	67.7	75.8	75.5	41.2	**65.0**	63.4	94.2	**61.0**	67.4
RWKV7-World3-1.5B	5.6	**69.5**	70.8	77.1	**78.1**	44.5	62.4	**68.2**	**94.3**	43.3	**67.6**
RWKV6-World2.1-3B	2.5	71.7	68.4	76.4	71.2	35.6	56.3	66.3	92.2	28.3	62.9
Llama3.2-3B	[b]15.0	70.5	73.6	76.7	74.5	42.2	50.7	69.9	95.7	56.5	67.8
Qwen2.5-3B	18.0	67.1	73.5	78.6	77.4	45.0	**70.2**	68.5	**96.2**	**65.7**	71.4
RWKV7-World3-2.9B	5.6	**73.4**	**76.4**	**79.7**	**81.0**	**48.7**	61.8	**72.8**	95.0	55.0	**71.5**

[a] **glue** is the average accuracy of 8 subtasks: **mnli**, **mnli_mismatch**, **mrpc**, **qnli**, **qqp**, **rte**, **sst2** and **wnli**
[b] Llama3.2-1B and 3B were pruned and distilled from Llama3.1-8B (Grattafiori et al., 2024)

Table 3: English Focused Benchmarks, including LAMBADA (**lmb.o**) (Paperno et al., 2016), Hellswag (**hella**) (Hampel, 1974), PIQA (Bisk et al., 2020), AI2 ARC (**arcE**, **arcC**) (Bhakthavatsalam et al., 2021), GLUE (Wang et al., 2018), Winogrande (**WG**) (Sakaguchi et al., 2021), SciQ (Welbl et al., 2017), MMLU (Hendrycks et al., 2021).

7.3 Associative Recall

Associative recall (AR) (Arora et al., 2023) evaluates the ability of the model to recall previously encountered information within a given context. Research indicates that a model's capacity for AR can reflect its effectiveness in learning from context (Elhage et al., 2021; Olsson et al., 2022). Consequently, AR has become a standard benchmark for developing new architectural designs in language models (Fu et al., 2023; Poli et al., 2023; Lutati et al., 2023).

We train two-layer RWKV-7 with MQAR and increased the difficulty by scaling the sequence length to as long as 2048. We use the RWKV-7 specific initialization, and set the ϵ of AdamW to 1×10^{-18} to stabilize learning in later stages. Weight decay of 0.1 is only applied to weight matrices, preventing the degeneration of certain modules (such as weights and biases of LayerNorm).

Model (Name)	Tokens (T)	lmb.m [a]ppl↓	lmb.m acc↑	pawsx acc↑	xcopa acc↑	xnli acc↑	xsClz acc↑	xwin acc↑	avg acc↑
RWKV5-World1-0.1B	0.6	270	22.0	48.6	53.0	36.1	51.7	59.5	45.1
SmolLM2-135M	2.0	1514	18.6	**51.2**	52.2	34.9	50.6	61.7	44.9
RWKV7-0.1B	1.6	**114**	**31.6**	46.1	**53.3**	**37.6**	**52.6**	**64.1**	**47.5**
RWKV5-World2-0.4B	1.1	66	36.8	49.5	54.0	38.5	54.1	65.6	49.8
SmolLM2-360M	4.0	389	25.8	51.4	51.7	36.0	51.2	67.8	47.3
Qwen2.5-0.5B	18.0	108	32.9	**52.6**	54.4	38.6	53.9	67.8	50.0
RWKV7-World3-0.4B	3.1	**52**	**39.6**	48.7	**55.4**	**40.3**	**55.3**	**72.9**	**52.0**
RWKV6-World2.1-1.6B	2.5	28	47.2	52.5	58.1	41.4	58.2	76.5	55.7
Llama3.2-1B	[b]15.0	52	39.0	53.9	55.3	41.2	56.6	72.2	53.0
SmolLM2-1.7B	11.0	85	37.1	**56.5**	53.1	38.1	54.1	72.8	52.0
Qwen2.5-1.5B	18.0	49	40.0	55.3	57.4	40.6	57.7	75.8	54.5
RWKV7-World3-1.5B	5.6	**25**	**48.4**	54.8	**59.7**	**43.7**	**61.4**	**79.8**	**58.0**
RWKV6-World2.1-3B	2.5	21	51.0	53.4	60.2	42.7	61.3	78.8	57.9
Llama3.2-3B	[b]15.0	30	45.9	**59.9**	58.5	44.2	60.6	79.2	58.1
Qwen2.5-3B	18.0	36	43.5	53.3	59.0	38.5	59.6	79.8	55.6
RWKV7-World3-2.9B	5.6	**18**	**52.9**	58.2	**63.1**	**45.4**	**64.7**	**82.4**	**61.1**

[a] The perplexity is the geometric mean, rather than arithmetic average, across 5 languages

[b] Llama3.2-1B and 3B were pruned and distilled from Llama3.1-8B (Grattafiori et al., 2024)

Table 4: Multilingual Benchmarks, including LAMBADA Multilingual (**lmb.m**) (Gao et al., 2023), XCOPA (Ponti et al., 2020), XNLI (Conneau et al., 2018),XStoryCloze (**xsClz**) (Lin et al., 2022), xWinogrande (**xwin**) (Tikhonov & Ryabinin, 2021).

Interestingly, with only a WKV size of 8192, RWKV-7 is able to recall 72.93% at the setting of 256 Key-value pairs. This suggests that a total of roughly $256 \times 0.7293 \times \log_2$ (number of key tokens × number of value tokens) = $186.2 \times 2 \times \log_2(4096) = 4480.8$ bits of information, is stored in a 8192 dimensional state, yielding an information density of 0.547 bits per dimension.

7.4 Mechanistic Architecture Design

We evaluate RWKV-7 on the Mechanistic Architecture Design (MAD) benchmark (Poli et al., 2024), a suite of synthetic token manipulation tasks designed to probe architectural capabilities in sequence modeling, as shown in Table 7.

RWKV-7 achieves the highest average score across all six tasks, outperforming previous architectures. It demonstrates perfect accuracy on In-Context and Noisy Recall tasks, matching DeltaNet while setting a new state-of-the-art for Fuzzy Recall. RWKV-7 also shows strong performance in memorization and selective copying, suggesting effective combination of attention-based and recurrent model strengths.

7.5 Long Context Experiments

To evaluate the ability of RWKV models to retain information over long sequences, we measured loss versus sequence position (we select tokens in range $[L/2 - 16384, L/2 + 16384)$ for document length L) on the PG19 test set (Rae et al., 2019) for two types of RWKV7 models and their predecessors trained on either The Pile dataset or World dataset. Despite sharing the same architecture and being pretrained on 4k context windows, models trained on different datasets exhibited different behaviors. The Pile-trained RWKV7 showed more significant loss reduction on long contexts compared to its predecessors, demonstrating effective long-context extrapolation (see Figure 5). Surprisingly, for RWKV7 trained on the World dataset, when processing contexts longer than 10k, the loss began to show an increasing trend (see Figure 6). We speculate this is because the larger dataset and model size created inductive biases that caused overfitting to specific context lengths. Further experiments showed that fine-tuning on long contexts can restore its long context capabilities.

To further test RWKV-7 long-context retrieval abilities, we conduct a pass-key retrieval evaluation following the approach of (Chen et al., 2024b) and plot the results in Figure 7. In this evaluation,

Model	arXiv CS ↓	arXiv Phys. ↓	Github Python ↓	Github C++ ↓	AO3 Eng ↓	BBC news ↓	Wiki Eng ↓	average ↓
Qwen2.5-1.5B	**8.12**	8.65	**4.42**	**4.40**	11.76	9.58	9.49	**8.06**
RWKV-7 1.5B	8.25	8.77	5.57	5.29	**10.93**	**9.34**	**8.97**	8.16
Llama-3.2-1B	8.37	8.76	5.18	5.16	11.69	**9.34**	9.07	8.23
SmolLM2-1.7B	8.38	9.04	5.17	4.94	11.20	9.40	9.46	8.23
Index-1.9B	8.34	**8.59**	5.65	5.29	11.49	9.51	9.23	8.30
stablelm-2-1.6b	8.58	9.08	5.54	5.45	11.42	9.24	9.06	8.34
RWKV-6 1.5B	8.62	9.00	6.06	5.80	11.09	9.57	9.30	8.49
RWKV-5 1.5B	8.77	9.11	6.20	5.92	11.25	9.75	9.50	8.64
mamba2-1.3b	8.74	8.74	6.32	5.71	11.63	9.74	9.86	8.68
MobileLLM-1.5B	8.82	9.29	6.79	6.29	11.59	**9.15**	9.22	8.73
mamba-1.4b-hf	8.88	8.86	6.43	5.81	11.70	9.83	9.97	8.78
Zamba2-1.2B	8.57	9.21	6.91	7.08	11.39	9.38	9.26	8.83
SmolLM-1.7B	8.38	9.02	5.76	6.55	12.68	9.85	9.89	8.88
MobileLLM-1B	9.03	9.57	7.03	6.53	11.86	9.35	9.43	8.97
RWKV-4 1.5B	9.34	9.80	6.54	6.16	11.33	10.00	9.82	9.00
pythia-1.4b-v0	9.12	9.20	6.79	6.15	12.19	10.20	10.43	9.15
Falcon3-1B-Base	8.60	9.20	6.92	7.16	13.04	10.45	10.75	9.45
Llama-3.2-3B	**7.78**	**8.10**	4.15	4.59	10.90	**8.70**	**8.28**	**7.57**
Qwen2.5-3B	7.79	8.25	**4.15**	**4.12**	11.23	9.15	8.96	7.66
RWKV-7 2.9B	7.90	8.34	5.16	4.88	**10.48**	8.92	8.47	7.74
stablelm-3b-4e1t	8.15	8.50	5.28	4.85	10.89	8.82	8.51	7.86
Minitron-4B-Base	8.09	8.70	5.13	4.74	11.05	9.08	8.90	7.96
recurrentgemma-2b	8.24	8.52	5.22	4.80	11.30	8.94	8.88	7.99
RWKV-6 3B	8.27	8.58	5.66	5.39	10.67	9.17	8.82	8.08
gemma-2-2b	8.39	8.81	5.36	5.01	11.35	8.90	9.03	8.12
mamba2attn-2.7b	8.33	8.29	5.78	5.22	11.13	9.28	9.26	8.18
RWKV-5 3B	8.42	8.70	5.78	5.51	10.83	9.36	9.00	8.23
mamba2-2.7b	8.43	8.37	5.93	5.34	11.21	9.37	9.38	8.29
Zamba2-2.7B	8.17	8.70	6.30	6.39	10.97	8.95	8.74	8.32
mamba-2.8b-hf	8.57	8.52	6.03	5.46	11.31	9.49	9.53	8.41
RWKV-4 3B	8.90	9.27	6.07	5.67	10.90	9.57	9.30	8.53
pythia-2.8b-v0	8.72	8.73	6.29	5.71	11.66	9.74	9.82	8.67

Table 5: Compression rate (unit: %) compared across different language models on various data sources, including arXiv papers, GitHub repositories, AO3 fiction, and news articles created after January 2025.

Dim	WKV state dim	$(64, 4)$	$(128, 8)$	$(256, 16)$	$(512, 64)$	$(1024, 128)$	$(2048, 256)$
64	8192	✓	✓	✓	98.43	95.01	72.93
128	16384	✓	✓	✓	✓	✓	94.97
256	32768	✓	✓	✓	✓	✓	98.97
512	65536	✓	✓	✓	✓	✓	✓

Table 6: RWKV-7 MQAR test results. We use (a, b) to denote the sequence length and number of KV pairs respectively. A check mark ✓ indicates that the model achieves over 99% accuracy. Results are maxed over 3 different learning rate settings.

a single sentence is repeated multiple times within a long context window, with a key phrase embedded at different positions. RWKV7-World3-1.5B achieves perfect accuracy up to a context length of 19600 tokens but exhibits degradation beyond 20600 tokens. The larger RWKV7-World3-2.9B extends perfect retrieval up to 35000 tokens, highlighting the benefits of scaling. However, performance begins to degrade beyond this point.

To explore potential improvements, we fine-tuned RWKV7-World3-1.5B and RWKV7-World3-2.9B on packed training sequences of length 128k tokens from a specially constructed dataset, which leads to further improvements in retrieval accuracy. With this fine-tuning, RWKV-7 (1.5B) reliably retrieves key phrases up to 29k tokens, and degradation is observed only around 40k tokens. RWKV-7 (2.9B) reliably retrieves the pass key up to 30k tokens, and degrades around 50k tokens.

Our context length extension dataset is comprised of both public and custom sources listed in Table 8. We employed a document length-based weighting scheme to prioritize longer contexts

Model	Compress	Fuzzy Recall	In-Context Recall	Memorize	Noisy Recall	Selective Copy	Avg
RWKV-7	44.5	**43.2**	**100**	89.1	**100**	98.8	**79.3**
Transformer	51.6	29.8	94.1	85.2	86.8	99.6	74.5
Multihead Hyena	44.8	14.4	99.0	89.4	98.6	93.0	73.2
DeltaNet	42.2	35.7	**100**	52.8	**100**	**100**	71.8
Mamba	**52.7**	6.7	90.4	**89.5**	90.1	86.3	69.3
Hyena	45.2	7.9	81.7	89.5	78.8	93.1	66.0
GLA	38.8	6.9	80.8	63.3	81.6	88.6	60.0

Results for comparison models from Yang et al. (2024c)

Table 7: Results on the MAD benchmark
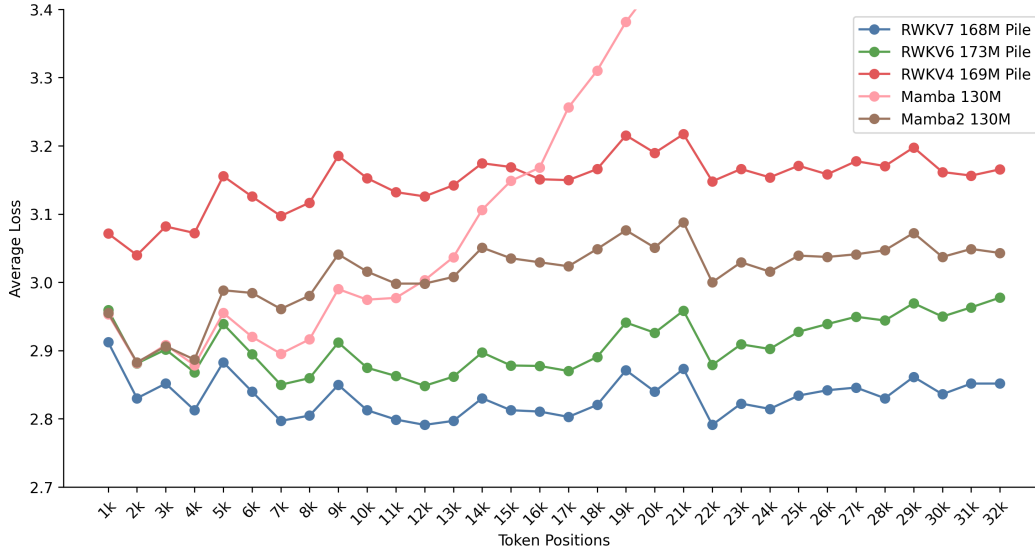
Figure 5: PG19 loss versus sequence position for RWKV and Mamba models trained on The Pile datasets.

during training. To approximate document lengths of 128,000 tokens, we used character counts of 512,000. Documents of less than 32,768 characters were assigned a weight of 1.0, while longer documents were assigned linearly increasing weights between 2.0 and 3.0, with a cap of 3.0 beyond 512,000. This method increases the inclusion of longer documents to bolster the model's handling of extended contexts while retaining shorter documents for diversity.

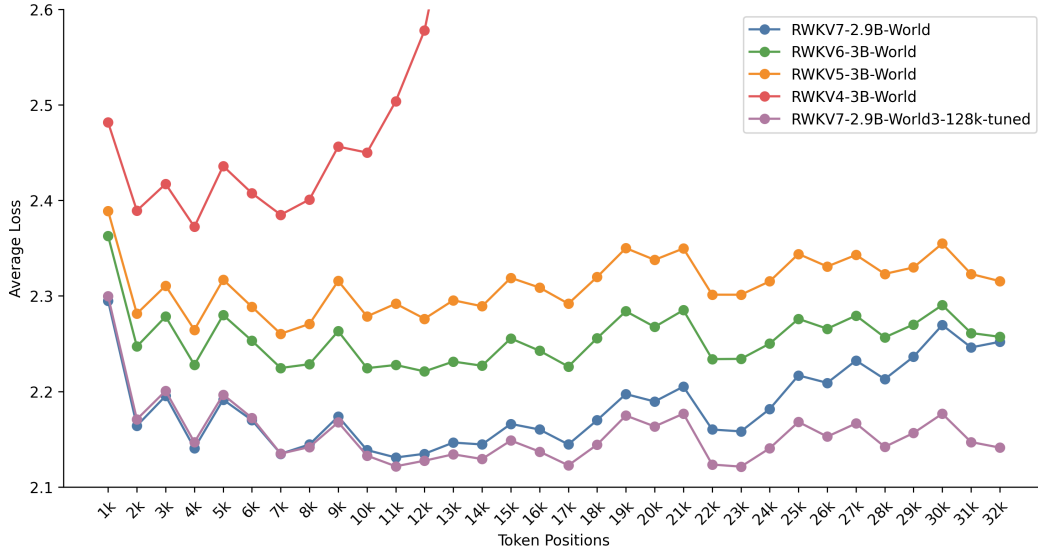| Dataset	Type	Amount
dclm-baseline-1.0	Public	25%
fineweb-edu	Public	15%
fineweb	Public	5%
codeparrot/github-code	Public	10%
arXiv-CC0-v0.5	Custom	10%
SuperWikiNEXT-32B	Custom	10%
public domain books	Custom	15%
the-stack (filtered)	Custom	10%

Table 8: Context Length Extension Dataset Components

Figure 6: PG19 loss versus sequence position for RWKV7 models and predecessors trained on the World dataset.

7.6 Evaluating State Tracking Using Group Multiplication

We adopt the experimental setting from Merrill et al. (2024) to evaluate the state-tracking capabilities of RWKV7 in comparison to Transformer, Mamba, S4, and classical RNN models. Given a sequence $g_0, g_1, g_2, \ldots, g_n$ drawn from A_5, $A_4 \times \mathbb{Z}_5$, or $\mathbb{Z}_{60}$, each step i is labeled with the cumulative product of the first i elements.

We plot the minimum number of layers required to achieve over 95% validation accuracy on group multiplication tasks, as a function of sequence length and group structure. The results are shown in Figure 8. Our findings indicate that RWKV-7 exhibits stronger state-tracking capabilities than Transformers, Mamba, and S4, though slightly weaker than classical RNNs. Figure 8 also aligns with our theory from Appendix D.2, which predicts that RWKV-7 can perform state tracking and recognize any regular language with a constant number of layers. RWKV-7 has no expressivity advantage for state tracking compared to classical RNNs, which can recognize any regular language in a single layer. However, classical RNNs, while being theoretically expressive, typically suffer from gradient vanishing and memorization problems (Zucchet & Orvieto, 2024) and cannot be parallelized efficiently, unlike RWKV-7.

8 Speed and Memory Usage

We compare the training speed and memory usage of the RWKV-7 attention-like kernel with the RWKV-6 kernel and Flash Attention v3 (Shah et al., 2024). The "RWKV-7" kernel accelerates bfloat16 matrix multiplications with modern CUDA instructions. We also include the "RWKV-7 fp32" kernel, which is simpler and performs all its internal calculations using float32. Although the bfloat16 kernel is faster, to maximize precision, the RWKV-7 fp32 kernel was used to train the RWKV-7 World models.

Our CUDA kernels are tuned for head dimension 64, as used in the RWKV-7-World models. The kernels still perform well for head dimension 128, but their efficiency drops off at larger head dimensions. There exist other RWKV-7 implementations which focus on head dimensions greater than 128. A key example is the Flash Linear Attention library (Yang & Zhang, 2024), which offers a Triton-based implementation designed for these larger configurations.

Speed In Figure 9, we time the forward + backward pass of each kernel for batch size 8, head dimension 64 and model dimension 4096 (64 wkv heads) on an H100 SXM GPU, for varying

(a) 1.5B   (b) 3B

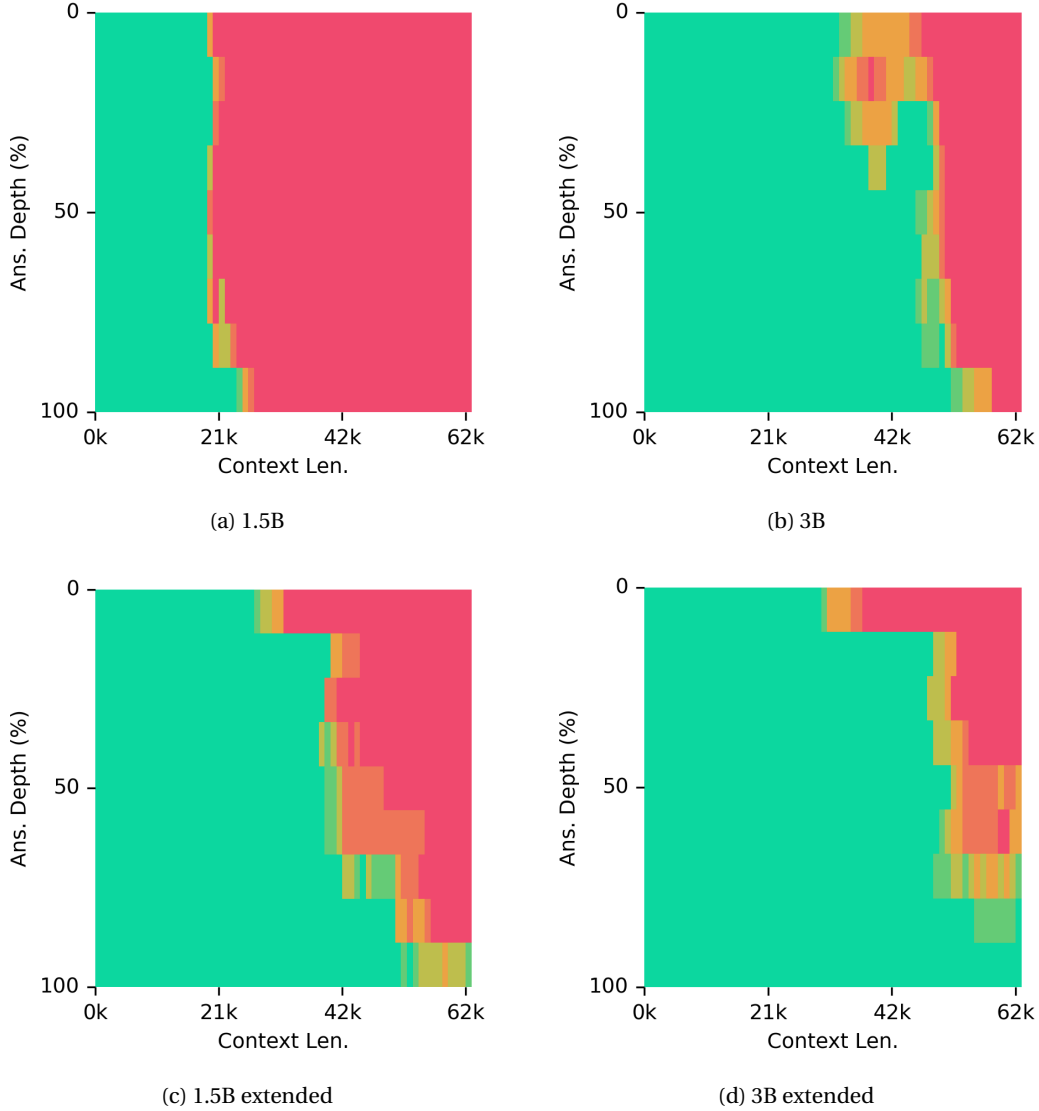(c) 1.5B extended   (d) 3B extended

Figure 7: RWKV7-World3 pass-key retrieval evaluation

sequence lengths. Although Flash Attention v3 is heavily optimized for the H100 GPU, it scales quadratically with sequence length, while the RWKV models scale linearly. This makes the RWKV models faster than attention for large sequence lengths. Furthermore, the optimized RWKV-7 kernel is about three times faster than the official RWKV-6 kernel.

The forward pass of RWKV-7 is about twice as fast as the backward pass. For inference, the forward pass does not need to store the wkv state, making it faster. For example, for sequence length 16k, the forward pass without storing state takes 7.9 ms, while the forward pass with storing state takes 11.2 ms, the backward pass takes 22.5 ms, and the Flash Attention v3 forward pass takes 33.9 ms.

Memory   The peak training memory usages of the tested models are well described by the formulas derived below. For example, the runs in Figure 9 required peak memory within 2% of the estimates.

In the tested kernels, the memory required per stored variable (e.g. q, k, or v in attention) is

$$\text{batch size} \times \text{model dimension} \times \text{sequence length} \times 2 \text{ bytes for bfloat16.}$$
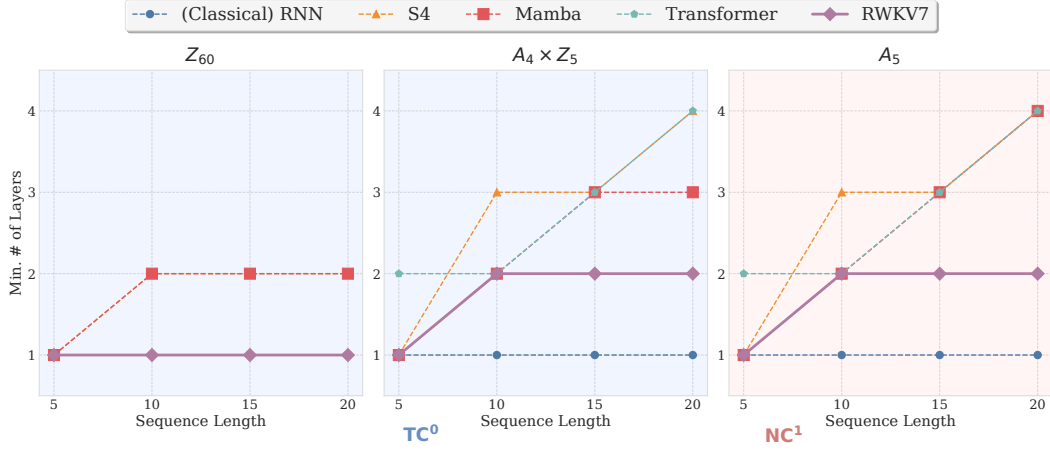
16

Figure 8: Minimum number of layers (lower is better) required to attain > 95% validation accuracy on group multiplication problems by sequence length and group.
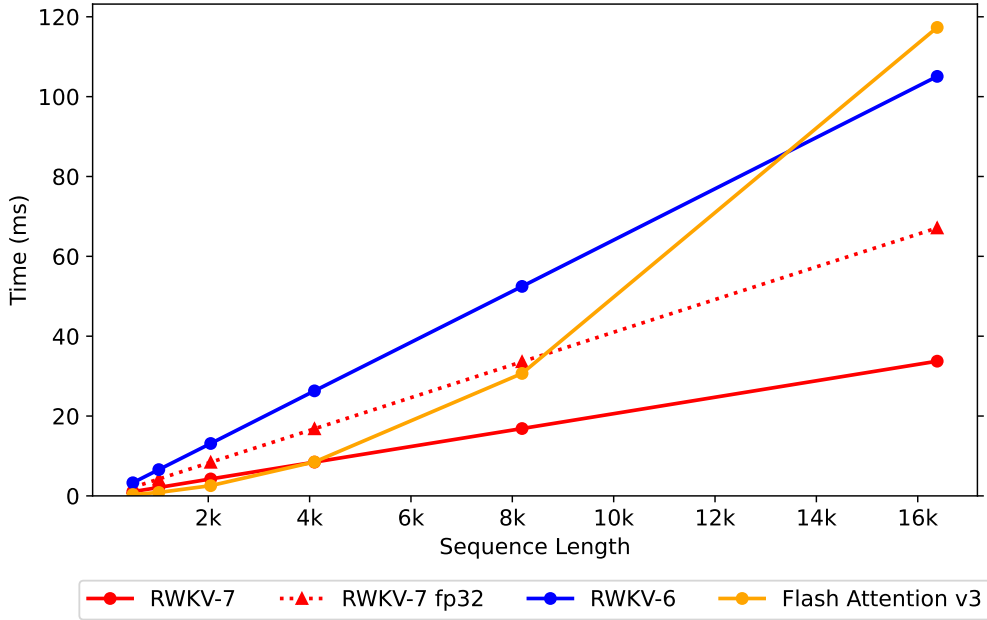

Figure 9: Time vs. Sequence Length (H100)

For sequence length 1024, this is 64MB per variable. To calculate memory usage, we may use Flash Attention v3: 10 variables, RWKV-6: 10 variables, RWKV-7: 18 variables, RWKV-7 fp32: 24 variables.

Flash Attention v3 requires 4 variables q, k, v and output for the forward pass, and the corresponding 4 gradients. Finally, the backward pass uses a temporary variable to accumulate the gradient of q in float32, yielding 2 variables worth of addition memory, for a total of 4+4+2 = 10.

RWKV-6 requires 5 variables r, w, k, v, and output for the forward pass and also the corresponding 5 gradients for the backward pass, for a total of 10.

RWKV-7 uses 7 variables in the forward pass (r, w, k, v, $-\hat{\kappa}$, $\hat{\kappa} \odot a$, and output), and the corresponding 7 gradients. Additionally, it stores the wkv state every 16 timesteps. At head size 64, the state contributes the equivalent of 4 variables, for a total of 18. "RWKV-7 fp32" has the same 14 forward

17

variables and gradients, but uses more memory to store the states in float32, for a total of 24 variable equivalents.

Memory usage is constant for single token inference and follows the formulas above, minus the gradients and state storage. Pre-fill can easily be accomplished in a chunked manner, with memory usage growing linearly with regard to chunk size. This allows an easy trade-off for fast parallelized pre-fill, with user selectable maximum memory usage.

9   Multimodal Experiments

In this section, we explore the capabilities of Goose when extended to handle multimodal tasks, where the model processes and integrates textual inputs with inputs from a different domain.
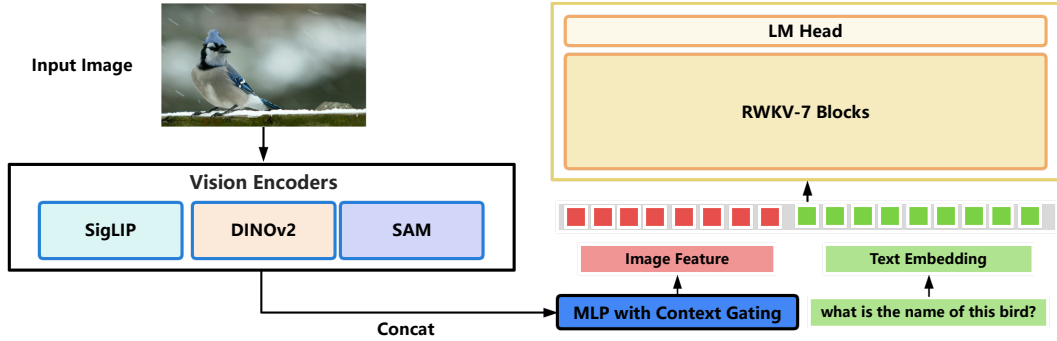

Figure 10: The architecture of VisualRWKV-7. The input image is processed by three vision encoders, and the obtained features are concatenated. Afterward, they are projected through an MLP with context gating to align with the dimensions of the RWKV-7 block. Finally, the image features are concatenated with the text embeddings and fed into the RWKV-7 LLM.

RWKV for Image Understanding   To demonstrate the modeling capabilities of RWKV-7, we constructed VisualRWKV-7 (Figure 10), a visual language model based on the RWKV-7 block, to evaluate the image understanding capabilities of RWKV-7. VisualRWKV-6 (Hou et al., 2024) used the CLIP encoder, which focused on processing low-resolution images and achieved good results. VisualRWKV-7 replaces the CLIP encoder with SigLIP and DINO visual encoders, and introduced a new high-resolution SAM vision encoder, which enhances the model's supported resolution to 1024 x 1024.

Method	Vision Encoder	LLM	VQA	SQA	TQA	GQA
VisualRWKV-6	SigLIP+DINOv2+SAM	RWKV6-1.6B	73.6	57.0	48.7	58.2
VisualRWKV-6	SigLIP+DINOv2+SAM	RWKV6-3.1B	79.1	62.9	52.7	61.0
VisualRWKV-7	SigLIP+DINOv2+SAM	RWKV7-0.1B	75.2	50.6	37.9	59.9
VisualRWKV-7	SigLIP+DINOv2+SAM	RWKV7-0.4B	77.9	55.0	41.1	62.3
VisualRWKV-7	SigLIP+DINOv2+SAM	RWKV7-1.5B	79.8	59.7	49.5	63.2
VisualRWKV-7	SigLIP+DINOv2+SAM	RWKV7-2.9B	**80.5**	**63.4**	**58.0**	**63.7**

Table 9: A comparison of VisualRWKV-7 to other Visual Language Models across 4 distinct benchmarks. We evaluate these models on benchmarks: GQA(Hudson & Manning, 2019), SQA(Lu et al., 2022), TQA(Singh et al., 2019) and VQA(Li et al., 2023b).

The experimental results of VisualRWKV-7 are shown in Table 9. The vision encoders used in both VisualRWKV-7 and VisualRWKV-6 are identical, and the training data remains consistent, aligned with the training data of LLaVA-1.5. The first stage consists of 558k alignment data, while the second stage includes 665k SFT data.

VisualRWKV-7 0.1B and 0.4B outperform VisualRWKV-6 1.6B on the in-domain benchmarks VQAv2 and GQA and rapidly approach VisualRWKV-6 1.6B on two other benchmarks. The experimental

results are highly compelling. With only 1/4 of the parameters (1.6B vs. 0.4B), VisualRWKV-7 surpasses VisualRWKV-6 on the VQAv2 and GQA benchmarks, demonstrating the powerful modeling capabilities of RWKV-7.

On the out-of-domain benchmark SQA, VisualRWKV-7 2.9B also outperforms VisualRWKV-6 3.1B, indicating that VisualRWKV-7 possesses strong generalization ability. In the TextQA (TQA) benchmark, which assesses a model's associative recall, VisualRWKV-7 2.9B achieves a 5.3-point improvement over VisualRWKV-6 3.1B, further proving its superior associative recall capabilities.

RWKV for Audio Modeling   To investigate the effectiveness of RWKV-7 for audio modeling, we introduce AudioRWKV-7, a novel adaptation of RWKV-7 for audio embedding analysis. We use a bi-directional modification to RWKV-7, similar to the modification used by Duan et al. (2024). We employ this approach to interpret and process complex, high-dimensional spectrogram features. To further capture acoustic and temporal characteristics, an mel-spectrogram is divided into patch tokens using a Patch-Embed CNN with a kernel size of $(P \times P)$ and sequentially fed into the model. The width and height of an audio mel-spectrogram represent the time and frequency bins, respectively. Typically, the time dimension is significantly longer than the frequency dimension. To effectively capture relationships among frequency bins within the same time frame, the mel-spectrogram is first segmented into patch windows $w_1, w_2, ..., w_n$, followed by further division of patches within each window. The token sequence follows the order: *time → frequency → window*. This arrangement ensures that patches corresponding to different frequency bins within the same time frame are positioned adjacent to each other in the input sequence.

We evaluated the performance of AudioRWKV-7 across multiple model scales and architectures, using the AudioSet dataset (Gemmeke et al., 2017). Detailed experimental outcomes are presented in Table 10. From the results we find that our model achieves comparable performance with a much smaller parameter count when compared with CNN, Transformer and Mamba based architectures, and exceeds the performance of AudioRWKV-6. These findings demonstrate the robustness and versatility of AudioRWKV. Note that to ensure a fair comparison, we retrained AudioRWKV-6 without ensembling models with different patch settings, which accounts for the difference in results from (Peng et al., 2024a).

Model	#Parameters	Architecture	mAP ↑
DeepRes (Ford et al., 2019)	26M	CNN	0.392
HST-AT	88.5M	Transformer	0.433*
HST-AT pretrained(Chen et al., 2022)	88.5M	Transformer	0.429*
MambaOut(Yu & Wang, 2024)	101.3M	Mamba	0.397
AudioRWKV-6	8.9M	RWKV6	0.381
AudioRWKV-6	19.8M	RWKV6	0.426
AudioRWKV-7	8.9M	RWKV7	0.392
AudioRWKV-7	19.8M	RWKV7	0.431

Table 10: A comparison of mean Average Precision (mAP) among AudioRWKV7 and other baselines on AudioSet dataset. HST-AT pretrained is a variation that uses vision transformer to initialize the weights. *Results reproduced by ourselves.

10   Conclusions

We introduced RWKV-7, a novel RNN architecture that pushes the boundaries of recurrent neural networks to new heights. RWKV-7 achieves state-of-the-art performance for its size across a wide range of benchmarks, demonstrating its potential to rival even highly optimized models such as Qwen2.5 despite being trained on many fewer tokens. As an RNN, RWKV-7 maintains high parameter efficiency, linear time complexity, and constant memory usage, offering a compelling alternative to traditional Transformer-based architectures.

10.1   Limitations

Despite its strengths, the RWKV-7 architecture and models face certain limitations yet to be mitigated in future work.

Numerical Precision. We observed that some operators, particularly the WKV7 kernel, are sensitive to the numerical precision of the implementation. This highlights the need for careful handling of numerical precision during model deployment. We also observed differences in training dynamics when using different kernels, which implies that the correct handling of precision while calculating and applying state updates is of utmost importance in this architecture.

Lack of Instruction Tuning and Alignment. All RWKV-7 models presented in this work are pretrained base models and have not undergone the phase of Supervised Fine-Tuning (SFT) for instruction following nor alignment with human preferences (RLHF). Future efforts should focus on incorporating these capabilities to enhance the model's usability in real-world applications.

Prompt Sensitivity. We found that the absence of the special token `<|endoftext|>` results in degraded performance of RWKV-7 models, e.g. inability to remember the first token of the input. See Appendix M for details.

Compute Resources. Due to computational budget constraints, our training was limited on at most $12 \times 8 = 96$ Nvidia H800 GPUs. This falls short of the resources required for recent large-scale training efforts, such as DeepSeek-V3 (DeepSeek-AI et al., 2025). Additionally, we are forced to continue training from pre-existing checkpoints of earlier RWKV architectures and therefore re-use some parts of our dataset. This may limit the capabilities of our models versus pre-training from scratch. Scaling up RWKV-7 to larger sizes and datasets will require additional computational resources.

10.2 Future Work

In addition to training larger RWKV-7 models with more tokens in the future, we also aim to explore several promising directions to further enhance the architecture and its capabilities.

Speedup Techniques. A variety of speed optimization techniques were highlighted in the technical report of DeepSeek-V3 (DeepSeek-AI et al., 2025), including Dual Pipelining Mechanism, Mixture-of-Experts, Multi-Token Prediction, and FP8 Training. We are aware that many of these techniques are orthogonal to RWKV-7's architectural optimizations, therefore could be integrated to further accelerate training in later RWKV models. However, RWKV-7, like its predecessors, has been trained completely without pipeline parallelism. We also noticed that there is room for speed optimization of RWKV-7 kernels and operators. We will explore both kernel-level optimizations and distributed training strategies in the future.

Incorporating Chain-of-Thought Reasoning. We believe that RWKV-7, as a linear RNN, is well-suited for efficient Chain-of-Thought reasoning (Wei et al., 2022). However, this capability has been barely explored due to the lack of suitable reinforcement learning pipelines. In future work, we plan to incorporate deep thinking abilities into RWKV-7, enabling it to excel in tasks requiring multi-step logical reasoning and complex problem-solving.

References

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. Smollm2: When smol goes big – data-centric training of a small language model, 2025. URL https://arxiv.org/abs/2502.02737.

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

Zhangir Azerbayev, Hailey Schoelkopf, Keiran Paster, Marco Dos Santos, Stephen Marcus McAleer, Albert Q. Jiang, Jia Deng, Stella Biderman, and Sean Welleck. Llemma: An open language model for mathematics. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=4WnqRR915j.

David A. Barrington. Bounded-width polynomial-size branching programs recognize exactly those languages in nc1. *Journal of Computer and System Sciences*, 38(1):150–164, 1989. URL https://www.sciencedirect.com/science/article/pii/0022000089900378.

Ali Behrouz, Peilin Zhong, and Vahab Mirrokni. Titans: Learning to memorize at test time, 2024.

Loubna Ben Allal, Anton Lozhkov, Guilherme Penedo, Thomas Wolf, and Leandro von Werra. Cosmopedia, February 2024a. URL https://huggingface.co/datasets/HuggingFaceTB/cosmopedia.

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

Sumithra Bhakthavatsalam, Daniel Khashabi, Tushar Khot, Bhavana Dalvi Mishra, Kyle Richardson, Ashish Sabharwal, Carissa Schoenick, Oyvind Tafjord, and Peter Clark. Think you have solved direct-answer question answering? try arc-da, the direct-answer ai2 reasoning challenge. *arXiv preprint arXiv:2102.03315*, 2021.

Yonatan Bisk, Rowan Zellers, Jianfeng Gao, Yejin Choi, et al. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pp. 7432–7439, 2020.

Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usvsn Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé (eds.), *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pp. 95–136, virtual+Dublin, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL https://aclanthology.org/2022.bigscience-1.9.

Ke Chen, Xingjian Du, Bilei Zhu, Zejun Ma, Taylor Berg-Kirkpatrick, and Shlomo Dubnov. Hts-at: A hierarchical token-semantic audio transformer for sound classification and detection. In *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 646–650. IEEE, 2022.

Yingfa Chen, Xinrong Zhang, Shengding Hu, Xu Han, Zhiyuan Liu, and Maosong Sun. Stuffed mamba: State collapse and state capacity of rnn-based long-context modeling, 2024a. URL https://arxiv.org/abs/2410.07145.

Yukang Chen, Shengju Qian, Haotian Tang, Xin Lai, Zhijian Liu, Song Han, and Jiaya Jia. Longlora: Efficient fine-tuning of long-context large language models, 2024b. URL https://arxiv.org/abs/2309.12307.

Alexis Conneau, Ruty Rinott, Guillaume Lample, Adina Williams, Samuel Bowman, Holger Schwenk, and Veselin Stoyanov. Xnli: Evaluating cross-lingual sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pp. 2475–2485, 2018.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality, 2024. URL https://arxiv.org/abs/2405.21060.

Dao AI Lab. Causal conv1d. https://github.com/Dao-AILab/causal-conv1d, 2023. Accessed: 2025-02-26.

DeepSeek-AI, Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, Damai Dai, Daya Guo, Dejian Yang, Deli Chen, Dongjie Ji, Erhang Li, Fangyun Lin, Fucong Dai, Fuli Luo, Guangbo Hao, Guanting Chen, Guowei Li, H. Zhang, Han Bao, Hanwei Xu, Haocheng Wang, Haowei Zhang, Honghui Ding, Huajian Xin, Huazuo Gao, Hui Li, Hui Qu, J. L. Cai, Jian Liang, Jianzhong Guo, Jiaqi Ni, Jiashi Li, Jiawei Wang, Jin Chen, Jingchang Chen, Jingyang Yuan, Junjie Qiu, Junlong Li, Junxiao Song, Kai Dong, Kai Hu, Kaige Gao, Kang Guan, Kexin Huang, Kuai Yu, Lean Wang, Lecong Zhang, Lei Xu, Leyi Xia, Liang Zhao, Litong Wang, Liyue Zhang, Meng Li, Miaojun Wang, Mingchuan Zhang, Minghua Zhang, Minghui Tang, Mingming Li, Ning Tian, Panpan Huang, Peiyi Wang, Peng Zhang, Qiancheng Wang, Qihao Zhu, Qinyu Chen, Qiushi Du, R. J. Chen, R. L. Jin, Ruiqi Ge, Ruisong Zhang, Ruizhe Pan, Runji Wang, Runxin Xu, Ruoyu Zhang, Ruyi Chen, S. S. Li, Shanghao Lu, Shangyan Zhou, Shanhuang Chen, Shaoqing Wu, Shengfeng Ye, Shengfeng Ye, Shirong Ma, Shiyu Wang, Shuang Zhou, Shuiping Yu, Shunfeng Zhou, Shuting Pan, T. Wang, Tao Yun, Tian Pei, Tianyu Sun, W. L. Xiao, Wangding Zeng, Wanjia Zhao, Wei An, Wen Liu, Wenfeng Liang, Wenjun Gao, Wenqin Yu, Wentao Zhang, X. Q. Li, Xiangyue Jin, Xianzu Wang, Xiao Bi, Xiaodong Liu, Xiaohan Wang, Xiaojin Shen, Xiaokang Chen, Xiaokang Zhang, Xiaosha Chen, Xiaotao Nie, Xiaowen Sun, Xiaoxiang Wang, Xin Cheng, Xin Liu, Xin Xie, Xingchao Liu, Xingkai Yu, Xinnan Song, Xinxia Shan, Xinyi Zhou, Xinyu Yang, Xinyuan Li, Xuecheng Su, Xuheng Lin, Y. K. Li, Y. Q. Wang, Y. X. Wei, Y. X. Zhu, Yang Zhang, Yanhong Xu, Yanhong Xu, Yanping Huang, Yao Li, Yao Zhao, Yaofeng Sun, Yaohui Li, Yaohui Wang, Yi Yu, Yi Zheng, Yichao Zhang, Yifan Shi, Yiliang Xiong, Ying He, Ying Tang, Yishi Piao, Yisong Wang, Yixuan Tan, Yiyang Ma, Yiyuan Liu, Yongqiang Guo, Yu Wu, Yuan Ou, Yuchen Zhu, Yuduan Wang, Yue Gong, Yuheng Zou, Yujia He, Yukun Zha, Yunfan Xiong, Yunxian Ma, Yuting Yan, Yuxiang Luo, Yuxiang You, Yuxuan Liu, Yuyang Zhou, Z. F. Wu, Z. Z. Ren, Zehui Ren, Zhangli Sha, Zhe Fu, Zhean Xu, Zhen Huang, Zhen Zhang, Zhenda Xie, Zhengyan Zhang, Zhewen Hao, Zhibin Gou, Zhicheng Ma, Zhigang Yan, Zhihong Shao, Zhipeng Xu, Zhiyu Wu, Zhongyu Zhang, Zhuoshu Li, Zihui Gu, Zijia Zhu, Zijun Liu, Zilin Li, Ziwei Xie, Ziyang Song, Ziyi Gao, and Zizheng Pan. Deepseek-v3 technical report, 2025. URL https://arxiv.org/abs/2412.19437.

Grégoire Delétang, Anian Ruoss, Paul-Ambroise Duquenne, Elliot Catt, Tim Genewein, Christopher Mattern, Jordi Grau-Moya, Li Kevin Wenliang, Matthew Aitchison, Laurent Orseau, Marcus Hutter, and Joel Veness. Language modeling is compression, 2024. URL https://arxiv.org/abs/2309.10668.

Yuchen Duan, Weiyun Wang, Zhe Chen, Xizhou Zhu, Lewei Lu, Tong Lu, Yu Qiao, Hongsheng Li, Jifeng Dai, and Wenhai Wang. Vision-rwkv: Efficient and scalable visual perception with rwkv-like architectures. *arXiv preprint arXiv:2403.02308*, 2024.

Nelson Elhage, Neel Nanda, Catherine Olsson, Tom Henighan, Nicholas Joseph, Ben Mann, Amanda Askell, Yuntao Bai, Anna Chen, Tom Conerly, Nova DasSarma, Dawn Drain, Deep Ganguli, Zac Hatfield-Dodds, Danny Hernandez, Andy Jones, Jackson Kernion, Liane Lovitt, Kamal Ndousse, Dario Amodei, Tom Brown, Jack Clark, Jared Kaplan, Sam McCandlish, and Chris Olah. A mathematical framework for transformer circuits. *Transformer Circuits Thread*, 2021. https://transformer-circuits.pub/2021/framework/index.html.

Qihang Fan, Huaibo Huang, and Ran He. Breaking the low-rank dilemma of linear attention, 2025. URL https://arxiv.org/abs/2411.07635.

Logan Ford, Hao Tang, François Grondin, and James R Glass. A deep residual network for large-scale acoustic scene analysis. In *InterSpeech*, pp. 2568–2572, 2019.

Daniel Y. Fu, Tri Dao, Khaled K. Saab, Armin W. Thomas, Atri Rudra, and Christopher Re. Hungry hungry hippos: Towards language modeling with state space models, 2023.

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. The pile: An 800gb dataset of diverse text for language modeling, 2020.

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

Jort F Gemmeke, Daniel PW Ellis, Dylan Freedman, Aren Jansen, Wade Lawrence, R Channing Moore, Manoj Plakal, and Marvin Ritter. Audio set: An ontology and human-labeled dataset for audio events. In *2017 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 776–780. IEEE, 2017.

Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab Al-Badawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit

Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers, Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL https://arxiv.org/abs/2407.21783.

Riccardo Grazzi, Julien Siems, Jörg K. H. Franke, Arber Zela, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues, 2024. URL https://arxiv.org/abs/2411.12537.

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

Albert Gu, Karan Goel, and Christopher Re. Efficiently modeling long sequences with structured state spaces, 2022.

Frank R Hampel. The influence curve and its role in robust estimation. *Journal of the american statistical association*, 69(346):383–393, 1974.

Dongchen Han, Yifan Pu, Zhuofan Xia, Yizeng Han, Xuran Pan, Xiu Li, Jiwen Lu, Shiji Song, and Gao Huang. Bridging the divide: Reconsidering softmax and linear attention, 2024. URL https://arxiv.org/abs/2412.06590.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021. URL https://arxiv.org/abs/2009.03300.

Roger A Horn and Charles R Johnson. *Matrix analysis.* Cambridge university press, 2012.

Haowen Hou, Peigen Zeng, Fei Ma, and Fei Richard Yu. Visualrwkv: Exploring recurrent neural networks for visual language models. *ArXiv*, abs/2406.13362, 2024. URL https://api.semanticscholar.org/CorpusID:270620870.

Drew A. Hudson and Christopher D. Manning. Gqa: A new dataset for real-world visual reasoning and compositional question answering. *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 6693–6702, 2019. URL https://api.semanticscholar.org/CorpusID:152282269.

Jean Kaddour. The minipile challenge for data-efficient language models, 2023.

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

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. *Proceedings of the 37 th International Conference on Machine Learning*, 2020b.

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

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, Qian Liu, Evgenii Zheltonozhskii, Terry Yue Zhuo, Thomas Wang, Olivier Dehaene, Mishig Davaadorj, Joel Lamy-Poirier, João Monteiro, Oleh Shliazhko, Nicolas Gontier, Nicholas Meade, Armel Zebaze, Ming-Ho Yee, Logesh Kumar Umapathi, Jian Zhu, Benjamin Lipkin, Muhtasham Oblokulov, Zhiruo Wang, Rudra Murthy, Jason Stillerman, Siva Sankalp Patel, Dmitry Abulkhanov, Marco Zocca, Manan Dey, Zhihan Zhang, Nour Fahmy, Urvashi Bhattacharyya, Wenhao Yu, Swayam Singh, Sasha Luccioni, Paulo Villegas, Maxim Kunakov, Fedor Zhdanov, Manuel Romero, Tony Lee, Nadav Timor, Jennifer Ding, Claire Schlesinger, Hailey Schoelkopf, Jan Ebert, Tri Dao, Mayank Mishra, Alex Gu, Jennifer Robinson, Carolyn Jane Anderson, Brendan Dolan-Gavitt, Danish Contractor, Siva Reddy, Daniel Fried, Dzmitry Bahdanau, Yacine Jernite, Carlos Muñoz Ferrandis, Sean Hughes, Thomas Wolf, Arjun Guha, Leandro von Werra, and Harm de Vries. Starcoder: may the source be with you!, 2023a.

Yifan Li, Yifan Du, Kun Zhou, Jinpeng Wang, Wayne Xin Zhao, and Ji rong Wen. Evaluating object hallucination in large vision-language models. In *Conference on Empirical Methods in Natural Language Processing*, 2023b. URL https://api.semanticscholar.org/CorpusID:258740697.

Yucheng Li, Yunhao Guo, Frank Guerin, and Chenghua Lin. Evaluating large language models for generalization and robustness via data compression, 2024b. URL https://arxiv.org/abs/2402.00861.

Xi Victoria Lin, Todor Mihaylov, Mikel Artetxe, Tianlu Wang, Shuohui Chen, Daniel Simig, Myle Ott, Naman Goyal, Shruti Bhosale, Jingfei Du, et al. Few-shot learning with multilingual generative language models. In *Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing*, pp. 9019–9052, 2022.

Bo Liu, Rui Wang, Lemeng Wu, Yihao Feng, Peter Stone, and Qiang Liu. Longhorn: State space models are amortized online learners, 2024. URL https://arxiv.org/abs/2407.14207.

Anton Lozhkov, Loubna Ben Allal, Leandro von Werra, and Thomas Wolf. Fineweb-edu: the finest collection of educational content, 2024. URL https://huggingface.co/datasets/HuggingFaceFW/fineweb-edu.

Pan Lu, Swaroop Mishra, Tony Xia, Liang Qiu, Kai-Wei Chang, Song-Chun Zhu, Oyvind Tafjord, Peter Clark, and A. Kalyan. Learn to explain: Multimodal reasoning via thought chains for science question answering. *ArXiv*, abs/2209.09513, 2022. URL https://api.semanticscholar.org/CorpusID:252383606.

Shahar Lutati, Itamar Zimerman, and Lior Wolf. Focus your attention (with adaptive iir filters), 2023.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *ArXiv*, abs/1812.06162, 2018. URL https://api.semanticscholar.org/CorpusID:56262183.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. *ArXiv*, abs/2404.08819, 2024. URL https://api.semanticscholar.org/CorpusID:269149086.

Igor Molybog, Peter Albert, Moya Chen, Zachary DeVito, David Esiobu, Naman Goyal, Punit Singh Koura, Sharan Narang, Andrew Poulton, Ruan Silva, Binh Tang, Diana Liskovich, Puxin Xu, Yuchen Zhang, Melanie Kambadur, Stephen Roller, and Susan Zhang. A theory on adam instability in large-scale machine learning, 2023. URL https://arxiv.org/abs/2304.09871.

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. In-context learning and induction heads, 2022.

Denis Paperno, Germán Kruszewski, Angeliki Lazaridou, Quan Ngoc Pham, Raffaella Bernardi, Sandro Pezzelle, Marco Baroni, Gemma Boleda, and Raquel Fernández. The lambada dataset: Word prediction requiring a broad discourse context. *arXiv preprint arXiv:1606.06031*, 2016.

Keiran Paster, Marco Dos Santos, Zhangir Azerbayev, and Jimmy Ba. Openwebmath: An open dataset of high-quality mathematical web text, 2023.

Bo Peng, Eric Alcaide, Quentin Anthony, Alon Albalak, Samuel Arcadinho, Stella Biderman, Huanqi Cao, Xin Cheng, Michael Chung, Leon Derczynski, Xingjian Du, Matteo Grella, Kranthi Gv, Xuzheng He, Haowen Hou, Przemyslaw Kazienko, Jan Kocon, Jiaming Kong, Bartłomiej Koptyra, Hayden Lau, Jiaju Lin, Krishna Sri Ipsit Mantri, Ferdinand Mom, Atsushi Saito, Guangyu Song, Xiangru Tang, Johan Wind, Stanisław Woźniak, Zhenyuan Zhang, Qinghua Zhou, Jian Zhu, and Rui-Jie Zhu. RWKV: Reinventing RNNs for the transformer era. In Houda Bouamor, Juan Pino, and Kalika Bali (eds.), *Findings of the Association for Computational Linguistics: EMNLP 2023*, pp. 14048–14077, Singapore, December 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-emnlp.936. URL https://aclanthology.org/2023.findings-emnlp.936.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, Przemysław Kazienko, Kranthi Kiran GV, Jan Kocoń, Bartłomiej Koptyra, Satyapriya Krishna, Ronald McClelland Jr., Jiaju Lin, Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Cahya Wirawan, Stanisław Woźniak, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Peng Zhou, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: Rwkv with matrix-valued states and dynamic recurrence, 2024a. URL https://arxiv.org/abs/2404.05892.

Bo Peng, Daniel Goldstein, Quentin Gregory Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Teddy Ferdinan, Kranthi Kiran GV, Haowen Hou, Satyapriya Krishna, Ronald McClelland Jr., Niklas Muennighoff, Fares Obeid, Atsushi Saito, Guangyu Song, Haoqin Tu, Ruichong Zhang, Bingchen Zhao, Qihang Zhao, Jian Zhu, and Rui-Jie Zhu. Eagle and finch: RWKV with matrix-valued states and dynamic recurrence. In *First Conference on Language Modeling*, 2024b. URL https://openreview.net/forum?id=soz1SEiPeq.

Michael Poli, Stefano Massaroli, Eric Nguyen, Daniel Y Fu, Tri Dao, Stephen Baccus, Yoshua Bengio, Stefano Ermon, and Christopher Ré. Hyena hierarchy: Towards larger convolutional language models. In *International Conference on Machine Learning*, pp. 28043–28078. PMLR, 2023.

Michael Poli, Armin W Thomas, Eric Nguyen, Pragaash Ponnusamy, Björn Deiseroth, Kristian Kersting, Taiji Suzuki, Brian Hie, Stefano Ermon, Christopher Ré, Ce Zhang, and Stefano Massaroli. Mechanistic design and scaling of hybrid architectures, 2024. URL https://arxiv.org/abs/2403.17844.

Edoardo Maria Ponti, Goran Glavaš, Olga Majewska, Qianchu Liu, Ivan Vulić, and Anna Korhonen. XCOPA: A multilingual dataset for causal commonsense reasoning. In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2362–2376, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.185. URL https://aclanthology.org/2020.emnlp-main.185.

Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL https://arxiv.org/abs/2412.15115.

Jack W. Rae, Anna Potapenko, Siddhant M. Jayakumar, and Timothy P. Lillicrap. Compressive transformers for long-range sequence modelling, 2019. URL https://arxiv.org/abs/1911.05507.

Mark Rudelson and Roman Vershynin. Sampling from large matrices: An approach through geometric functional analysis. *J. ACM*, 54(4):21–es, July 2007. ISSN 0004-5411. doi: 10.1145/1255443.1255449. URL https://doi.org/10.1145/1255443.1255449.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. *Communications of the ACM*, 64(9):99–106, 2021.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers, 2021. URL https://arxiv.org/abs/2102.11174.

Jürgen Schmidhuber. Learning to control fast-weight memories: An alternative to dynamic recurrent networks. *Neural Computation*, 4(1):131–139, 1992. doi: 10.1162/neco.1992.4.1.131.

John Schultz, Jakub Adamek, Matej Jusup, Marc Lanctot, Michael Kaisers, Sarah Perrin, Daniel Hennes, Jeremy Shar, Cannada Lewis, Anian Ruoss, Tom Zahavy, Petar Veličković, Laurel Prince, Satinder Singh, Eric Malmi, and Nenad Tomašev. Mastering board games by external and internal planning with language models, 2024. URL https://arxiv.org/abs/2412.12119.

Jay Shah, Ganesh Bikshandi, Ying Zhang, Vijay Thakkar, Pradeep Ramani, and Tri Dao. Flash-attention-3: Fast and accurate attention with asynchrony and low-precision. *arXiv preprint arXiv:2407.08608*, 10, 2024.

Amanpreet Singh, Vivek Natarajan, Meet Shah, Yu Jiang, Xinlei Chen, Dhruv Batra, Devi Parikh, and Marcus Rohrbach. Towards vqa models that can read, 2019.

Samuel L. Smith, Pieter-Jan Kindermans, and Quoc V. Le. Don't decay the learning rate, increase the batch size. In *International Conference on Learning Representations*, 2018. URL https://openreview.net/forum?id=B1Yy1BxCZ.

Daria Soboleva, Faisal Al-Khateeb, Robert Myers, Jacob R Steeves, Joel Hestness, and Nolan Dey. SlimPajama: A 627B token cleaned and deduplicated version of RedPajama. https://www.cerebras.net/blog/slimpajama-a-627b-token-cleaned-and-deduplicated-version-of-redpajama, June 2023. URL https://huggingface.co/datasets/cerebras/SlimPajama-627B.

Luca Soldaini, Rodney Kinney, Akshita Bhagia, Dustin Schwenk, David Atkinson, Russell Authur, Ben Bogin, Khyathi Chandu, Jennifer Dumas, Yanai Elazar, Valentin Hofmann, Ananya Harsh Jha, Sachin Kumar, Li Lucy, Xinxi Lyu, Nathan Lambert, Ian Magnusson, Jacob Morrison, Niklas Muennighoff, Aakanksha Naik, Crystal Nam, Matthew E. Peters, Abhilasha Ravichander, Kyle Richardson, Zejiang Shen, Emma Strubell, Nishant Subramani, Oyvind Tafjord, Pete Walsh, Luke Zettlemoyer, Noah A. Smith, Hannaneh Hajishirzi, Iz Beltagy, Dirk Groeneveld, Jesse Dodge, and Kyle Lo. Dolma: an Open Corpus of Three Trillion Tokens for Language Model Pretraining Research. *arXiv preprint*, 2024.

Yu Sun, Xinhao Li, Karan Dalal, Jiarui Xu, Arjun Vikram, Genghan Zhang, Yann Dubois, Xinlei Chen, Xiaolong Wang, Sanmi Koyejo, Tatsunori Hashimoto, and Carlos Guestrin. Learning to (learn at test time): Rnns with expressive hidden states, 2024.

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

Alexey Tikhonov and Max Ryabinin. It's all in the heads: Using attention heads as a baseline for cross-lingual transfer in commonsense reasoning. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pp. 3534–3546, 2021.

Oguzhan Topsakal, Colby Jacob Edell, and Jackson Bailey Harper. Evaluating large language models with grid-based game competitions: An extensible llm benchmark and leaderboard, 2024. URL https://arxiv.org/abs/2407.07796.

Various. Archive of our own, 2025. URL https://archiveofourown.org.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need, 2023.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 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*, pp. 353–355, 2018.

Jason Wei, Maarten Bosma, Vincent Y. Zhao abd Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V. Le. Finetuned language models are zero-shot learners, 2021.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=_VjQlMeSB_J.

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

Bernard Widrow, Marcian E Hoff, et al. Adaptive switching circuits. In *IRE WESCON convention record*, volume 4, pp. 96–104. New York, 1960.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. Magpie: Alignment data synthesis from scratch by prompting aligned llms with nothing, 2024. URL https://arxiv.org/abs/2406.08464.

Takuto Yamana. Egaroucid, January 2025. URL https://www.egaroucid.nyanyan.dev/.

Songlin Yang and Yu Zhang. Fla: A triton-based library for hardware-efficient implementations of linear attention mechanism, January 2024. URL https://github.com/fla-org/flash-linear-attention.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2023a.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule, 2024a.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training, 2024b. URL https://arxiv.org/abs/2312.06635.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length, 2024c. URL https://arxiv.org/abs/2406.06484.

Xiaocong Yang, James Y. Huang, Wenxuan Zhou, and Muhao Chen. Parameter-efficient tuning with special token adaptation, 2023b. URL https://arxiv.org/abs/2210.04382.

Weihao Yu and Xinchao Wang. Mambaout: Do we really need mamba for vision? *arXiv preprint arXiv:2405.07992*, 2024.

Lifan Yuan, Ganqu Cui, Hanbin Wang, Ning Ding, Xingyao Wang, Jia Deng, Boji Shan, Huimin Chen, Ruobing Xie, Yankai Lin, Zhenghao Liu, Bowen Zhou, Hao Peng, Zhiyuan Liu, and Maosong Sun. Advancing llm reasoning generalists with preference trees, 2024.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. Mammoth2: Scaling instructions from the web. *Advances in Neural Information Processing Systems*, 2024.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. Hellaswag: Can a machine really finish your sentence?, 2019. URL https://arxiv.org/abs/1905.07830.

Yifan Zhang, Yifan Luo, Yang Yuan, and Andrew Chi-Chih Yao. Training and evaluating language models with template-based data generation. *arXiv preprint arXiv:2411.18104*, 2024.

Zhanchao Zhou, Tianyi Wu, Zhiyun Jiang, and Zhenzhong Lan. Value residual learning for alleviating attention concentration in transformers, 2024. URL https://arxiv.org/abs/2410.17897.

Alex Zhuang, Ge Zhang, Tianyu Zheng, Xinrun Du, Junjie Wang, Weiming Ren, Stephen W. Huang, Jie Fu, Xiang Yue, and Wenhu Chen. Structlm: Towards building generalist models for structured knowledge grounding, 2024.

Nicolas Zucchet and Antonio Orvieto. Recurrent neural networks: vanishing and exploding gradients are not the end of the story, 2024. URL https://arxiv.org/abs/2405.21064.

A Author Contributions

Bo Peng Original RWKV-7 ideas, original code, performance optimizations, original experiments, dataset composition, and trained models from 0.1B to 2.9B.

Ruichong Zhang Personnel organization, wrote Sections 3, 4, 7.1, 7.3, 10 and Appendices C, E, H, J, K, M, Figures 1, 2, 11, 12, 15, 16 and Tables 1, 3, 4, 6, 14, 15, 16, 17, 18, 20. Additional contributions on implementing RWKV-7 for Flash-linear-attention and converting RWKV-7 models on HuggingFace.

Daniel Goldstein Manuscript organization, initial draft sections 1, 2, 3, 4, 5, FLOPS portion of 7.1, figures 3a and 4a, and appendices B, F, G. Proofreading and revisions of full manuscript. Oversaw and chose experiments for appendix K and for pass-key in subsection 7.5. Assistance with appendix D revisions and ideas. Developed and tested initial RWKV-7 Hugging Face model code.

Eric Alcaide Section 3, validation of CUDA kernels for scalable training, and manuscript proofreading.

Xingjian Du Experiments and writing of audio modeling in section 9.

Haowen Hou Wrote Section 9, covering architectural design, coding, model training, experimental evaluation, as well as figure (Figure 10), table (Table 9), and text writing.

Jiaju Lin Experiments and writing audio modeling in section 9.

Jiaxing Liu Experiments and writing audio modeling in section 9.

Janna Lu Needle-in-haystack evaluations for Section 7.5, experiments for figure 3a and 3b, figure 4a and 4b, figure 7, table 3, and table 18. Edits to Section 1, G, and abstract.

William Merrill Developmental discussion, proofreading and revisions for appendix D.

Guangyu Song Section 7.4. Experiments for 7.4.

Kaifeng Tan Section 7.2, Figures 5, 6. Appendix I on board game modeling with Othello/Reversi, including training data design, model implementation, experiments, and result analysis.

Saiteja Utpala Section 7.6 and state tracking experiments for Figure 8.

Nathan Wilce Extended context-length dataset development and extended-length model training. Description of extended context-length dataset in 7.5.

Johan S. Wind Main author of RWKV-7 CUDA kernel implementations. Experiments for Figure 9. Section 8 and Appendix D. Contributions to Appendix C.

Tianyi Wu Appendix D and Appendix L. Contributions to section 4 and Appendix C (proofreading and revisions).

Daniel Wuttke Constructed an itemized table of v2-v3 world datasets. Proofreading and revision of the manuscript. Contributed to Abstract, Sections 1 and B. Contributed to Tables 11, 12, 13, and 16 Performed evaluations of RWKV-7 world models and reference base models (Table 3 and 4).

Christian Zhou-Zheng Experiments and writing for Appendix K and Table 19. Contributions to Sections 1 and 2. Proofreading and revisions of full manuscript.

Dataset	Domain	Dataset	Domain
slimpajama C4	Web	Llama-3-Magpie-Pro-1M-v0.1	Align
dolma v1.6 (reddit only)[a]	Forums	Magpie-Pro-MT-300K-v0.1	Align
glaive-code-assistant-v3	Code	Magpie-Air-MT-300K-v0.1	Align
m-a-p_Code-Feedback	Code	Magpie-Qwen2-Pro-1M-v0.1	Align
cosmopedia-v0.1	Synthetic	Magpie-Phi3-Pro-300K-Filtered-v1	Align
SystemChat-2.0	Instruct		
Tess-v1.5	Instruct	Magpie-Gemma2-Pro-200K-Filtered-v0.1	Align
UltraInteract_sft	Instruct		

Table 11: Components added into the RWKV World v2.1 dataset, their source links, and their domains.
[a]We added only the reddit datasets from dolma v1.6

[b] DM_math as part of The Pile was in World v2 but missed being mentioned explicitly in (Peng et al., 2024b)

Dataset	Domain	Dataset	Domain
REMOVED slimpajama parts[a]	Web	StarCoder[c]	Code
dclm-baseline-10-of-10[b]	Web	python-edu	Code
ccnews	Web	cosmopedia-v0.2	Synthetic
fineweb-edu	Web Edu	WebInstructSub	Forums
TemplateGSM	Math	Buzz-v1.2	Instruct
open-web-math	Math	SKGInstruct	Instruct
algebraic-stack	Math	FLAN	Instruct

Table 12: Components added into the RWKV World v3 dataset, their source links, and their domains.
[a]We removed the CC and C4 components of SlimPajama from the corpus for World v3
[b]For DCLM-baseline, we include only global-shard_10_of_10
[c]For StarCoder, we now include all datasets, instead of just those datasets with at least 10 stars

B   Training Dataset Details

The RWKV World v3 corpus builds upon the RWKV World v2 corpus (Peng et al., 2024b) in two steps that we describe separately here for the purposes of reproducibility for the Goose training runs: World v2.1 adds the entries listed in Table 11 to sum to a total of approximately 1.4 trillion RWKV World Tokenizer tokens. World v3 adds more entries, listed in Table 12, to sum to a total of approximately 3.1 trillion tokens. In the combined corpora, all tokens are given equal weighting unless otherwise noted.

SlimPajama	Soboleva et al. (2023)
StarCoder	Li et al. (2023a)
Cosmopedia	Ben Allal et al. (2024a)
Dolma	Soldaini et al. (2024)
UltraInteract	Yuan et al. (2024)
Magpie	Xu et al. (2024)
FineWeb	Lozhkov et al. (2024)
DataComp LM (DCLM)	Li et al. (2024a)
WebInstructSub	Yue et al. (2024)
StructLM	Zhuang et al. (2024)
TemplateGSM	Zhang et al. (2024)
SmolLM Corpus	Ben Allal et al. (2024b)
FLAN	Wei et al. (2021)
OpenWebMath	Paster et al. (2023)
Algebraic-Stack	Azerbayev et al. (2024)

Table 13: RWKV World v3 dataset component citations

Category	Tokens (B)
Web	1945.2
Books	337.2
Code	258.4
Science & Wiki	222.7
Fiction	192.6
Chat & QA & Instruction	110.0
Math	32.3
Law & Government	19.0
Poetry & Lyrics	1.7
Total	3119.2

Table 14: RWKV World v3 dataset categories

Most of the component data sources for the RWKV World v3 dataset are used intact, with no up- or down-sampling done so that all tokens are given equal weighting. Some sub-sampling is done for over-represented languages within a few data sources in the original World v2 corpus. All newly added tokens in v2.1 and v3 are given equal weighting.

C Transition Matrix Eigenvalues and Stability

We are interested in all the eigenvalues of the transition matrix $A_t = \text{diag}(w_t) - \hat{\kappa}_t^T (a_t \odot \hat{\kappa}_t)$, and when it can be stable. We drop the subscript t for statements which hold at all timesteps, to avoid clutter.

Theorem 1. *Let $A = \text{diag}(w) - c\hat{\kappa}^T (a \odot \hat{\kappa}) \in M_m(\mathbb{R})$ be a matrix, where all entries of w belong to $(u, 1)$, where $u = \exp(-e^{-1/2}) = 0.5452\cdots$ is the clamping lower bound. Also, all entries of a are located in $(0, 1)$, and $\hat{\kappa}$ is a unit row vector. When $c \in (0, 1 + u)$, the following holds:*

1. *The matrix A is similar to a symmetric matrix, hence similar to a diagonal matrix.*
2. *All eigenvalues of A lie in the interval $(-1, 1)$.*
3. *The matrix A admits at most one negative eigenvalue.*
4. *If further assumed that a_t is time-independent, then the update formula is guaranteed to be stable, i.e. there exists a time-independent constant K such that*

$$\left\| \prod_{t=1}^{T} A_t \right\|_2 \leq K,$$

where $\|\cdot\|_2$ denotes the spectral norm.

This hyperparameter c in the formula can be regarded as a "Global ICLR Multiplier". This hyperparameter c is set to 1 in the current implementations of RWKV-7 language modeling.

Proof.

1. We notice that
$$A = \text{diag}(w) - c\hat{\kappa}^T (a \odot \hat{\kappa}) = \text{diag}(w) - c\hat{\kappa}^T \hat{\kappa} \text{diag}(a).$$

The matrix A itself is not necessarily a symmetric matrix. However, we can use the fact that $\text{diag}(a)$ is positive definite, so we can compute its square root. This allows us to rewrite

$$\text{diag}(a)^{1/2} A \, \text{diag}(a)^{-1/2} = \text{diag}(a)^{1/2}\text{diag}(w)\text{diag}(a)^{-1/2} - c\,\text{diag}(a)^{1/2}\hat{\kappa}^T \hat{\kappa} \, \text{diag}(a)\text{diag}(a)^{-1/2}$$

$$= \text{diag}(w) - c\left(\hat{\kappa}\text{diag}(a)^{1/2}\right)^T \left(\hat{\kappa}\text{diag}(a)^{1/2}\right),$$

which is a symmetric matrix. We denote this matrix by B. It has exactly the same eigenvalues as A, since it is formed by a similarity transformation of A.

32

2. Since $B = \text{diag}(a)^{1/2} A \text{diag}(a)^{-1/2}$ is symmetric, all eigenvalues of B (and hence A) are real and located on the interval of

$$\left[\min_{|\hat{s}|=1} \hat{s} B \hat{s}^T, \ \max_{|\hat{s}|=1} \hat{s} B \hat{s}^T \right].$$

It suffices to show that $\hat{s} B \hat{s}^T \in (-1, 1)$. This can be proved via direct expansion of B by

$$\hat{s} B \hat{s}^T = \hat{s}\,\text{diag}(w)\hat{s}^T - c\,\hat{s}(\hat{\kappa}\,\text{diag}(a)^{1/2})^T(\hat{\kappa}\,\text{diag}(a)^{1/2})\hat{s}^T$$

$$\geq u\hat{s}\hat{s}^T - c\left|(\hat{\kappa}\,\text{diag}(a)^{1/2})\hat{s}^T\right|^2$$

$$\geq u - c$$

$$> -1.$$

Similarly,

$$\hat{s} B \hat{s}^T = \hat{s}\,\text{diag}(w)\hat{s}^T - c\,\hat{s}(\hat{\kappa}\,\text{diag}(a)^{1/2})^T(\hat{\kappa}\,\text{diag}(a)^{1/2})\hat{s}^T$$

$$< \hat{s}\hat{s}^T - c\left|(\hat{\kappa}\,\text{diag}(a)^{1/2})\hat{s}^T\right|^2$$

$$\leq \hat{s}\hat{s}^T$$

$$= 1,$$

which completes this part.

3. Recall $B = \text{diag}(w) - c\left(\hat{\kappa}\,\text{diag}(a)^{1/2}\right)^T\left(\hat{\kappa}\,\text{diag}(a)^{1/2}\right)$. Then B is congruent with

$$\hat{B} = \text{diag}(w)^{-1/2} B \,\text{diag}(w)^{-1/2} = I - u^T u,$$

where $u = \sqrt{c}\,\hat{\kappa}\,\text{diag}(a)^{1/2}\text{diag}(w)^{-1/2}$.

Clearly, $\hat{B}$ has at most one negative eigenvalue with value $1 - \|u\|^2$, and all other eigenvalues equal to 1. By Sylvester's law of inertia (Horn & Johnson, 2012), congruency preserves the number of negative eigenvalues. Hence, B also has at most one negative eigenvalue.

4. We drop the subscript in the time-invariant a_t.

Since $B_t = \text{diag}(a)^{1/2} A_t \text{diag}(a)^{-1/2}$ is symmetric, the spectral norm of B_t is equal to the largest absolute value for eigenvalues of B_t. We proved previously that the eigenvalues lie in $(-1, 1)$, so $\|B_t\|_2 \leq 1$. That is, B_t is a contraction matrix.

Furthermore, we have

$$\prod_{t=1}^{T} A_t = \prod_{i=t}^{T} \text{diag}(a)^{-1/2} B_t \text{diag}(a)^{1/2}$$

$$= \text{diag}(a)^{-1/2}\left(\prod_{i=1}^{t} B_t\right)\text{diag}(a)^{1/2}.$$

Then

$$\left\|\prod_{t=1}^{T} A_t\right\|_2 \leq \left\|\text{diag}(a)^{-1/2}\right\|_2 \left\|\prod_{t=1}^{T} B_t\right\|_2 \left\|\text{diag}(a)^{1/2}\right\|_2$$

$$\leq \min(a)^{-1/2} \cdot 1 \cdot 1.$$

We can set $K = \min(a)^{-1/2} < \infty$, which is time-independent.

$\square$

While our stability proof only holds for time-independent a_t, we do not observe any problems with time-varying a_t in practice. We therefore put greater emphasis on the expressivity of RWKV-7, and include time-varying a_t.

D Expressivity of RWKV-7

We show that the RWKV-7 architecture can express NC^1-complete state tracking problems that cannot be expressed by transformers or other recurrent architectures such as S4 and Mamba, under standard complexity conjectures. We first show a particular NC^1-complete problem that can be expressed by RWKV-7 in Section D and then generalize the argument to show that any regular language can be recognized by RWKV-7 in Section D.2. As regular language recognition can be understood to formalize finite state tracking problems, this suggests an expressivity advantage of RWKV-7 on state-tracking problems.

D.1 Warmup: Expressivity Beyond TC^0

Recall that the RWKV-7 wkv state is updated, at each token, by multiplication with $A_t = \mathrm{diag}(w_t) - c\hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t)$, where $c = 1$. In the following, we will consider $c = 2$, and show that this yields expressivity beyond TC^0 (unless $\mathsf{TC}^0 = \mathsf{NC}^1$). TC^0 is the complexity class which includes transformers as well as all non-gated or diagonal SSMs (Merrill et al., 2024; Barrington, 1989).

Theorem 2. *RWKV-7 can solve a problem which is NC^1-complete under AC^0 reductions.*

Proof. RWKV-7 can, by Lemma 2, solve the problem of tracking swaps on five elements. This problem is NC^1-complete under AC^0 reductions (Merrill et al., 2024). $\square$

Lemma 1. *The RWKV-7 transition matrix can represent an arbitrary swap matrix, where a swap matrix is an identity matrix with two of its rows swapped.*

Proof. Given indices x and y, let

$$w_t = 1, \quad c = 2, \quad \hat{\kappa}_t = (e_x - e_y)/\sqrt{2}, \text{ and } a_t = 1.$$

Here e_i denotes the vector with 1 at position i and 0 elsewhere.

Then the transition matrix becomes $A_t = I - e_x^T e_x - e_y^T e_y + e_x^T e_y + e_y^T e_x$, which is the permutation matrix that swaps indices x and y. $\square$

Lemma 2 (RWKV-7 can track swaps on 5 elements)**.** *Let a sequence of swaps on 5 elements be encoded in the format*

$$\#[x_1 \leftrightarrow y_1][x_2 \leftrightarrow y_n]\dots,$$

where # is a special beginning-of-sequence token, and $[x_i \leftrightarrow y_i]$ is a token that denotes a swap between elements x_i and y_i. Then there exists a one-layer RWKV-7 model which outputs 1 if the sequence of swaps encode the identity permutation, and outputs 0 otherwise.

Proof. Let the RWKV-7 model have 5 wkv heads of head dimension 5. The embedding weights and "weight preparation" part of the time mixing layer are set such that the following two properties hold:

Firstly, when the model sees the special beginning-of-sequence token, the ith wkv head receives

$$w = 1, \ c = 2, \ \hat{\kappa} = e_i, \ a = 0, \text{ and } v = \tilde{k} = e_i.$$

Here e_i denotes the vector with 1 at position i and 0 elsewhere. This sets the state of the ith wkv head to $\mathbf{wkv} = e_i^T e_i$, which represents state i.

Secondly, when token represents a swap between tokens $1 \le x < y \le 5$. In this case, using Lemma 1, all wkv heads receive

$$w = 1, \ c = 2, \ \hat{\kappa} = (e_x - e_y)/\sqrt{2}, \ a = 1, \text{ and } v = \tilde{k} = 0.$$

This changes the state to y if it was x, or x if it was y, or keeps it unchanged otherwise.

To calculate the output, the ith wkv head checks whether it represents state i, by applying receptance $r = e_i$. Finally, the MLP layer combines these outputs to check if all 5 heads agree with the identity permutation, and the output head outputs 1 if this is true, and 0 otherwise. $\square$

D.2 Main Result: RWKV-7 Can Recognize Any Regular Language

Moreover, we are able to demonstrate that RWKV-7 has the capability to recognize any regular language. The regular languages are precisely those which can be recognized by a deterministic finite automaton (DFA). Therefore, it is sufficient to show that RWKV-7 can simulate any DFA. We define DFAs in the usual way:

Definition 1. *Classically, a DFA is a tuple $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$ where Q is a finite set of states, Σ is a finite vocabulary of tokens, $\delta_\sigma : Q \to Q$ is a transition function for each token $\sigma \in \Sigma$, $q_0 \in Q$ is the initial state, and $F \subseteq Q$ is a set of accepting states.*

Equivalently, the DFA's computation on $w \in \Sigma^*$ can be represented by matrix computations. Each δ_σ can be represented by a boolean matrix $M_\sigma \in \{0, 1\}^{|Q| \times |Q|}$, where $M_w(i, j) = 1$ iff $\delta_w(q_j) = q_i$. The initial state q_0 can be represented as a one-hot vector $\alpha \in \{0, 1\}^{|Q|}$, and the set of accepting states F can be represented as a multi-hot vector $\omega \in \{0, 1\}^{|Q|}$.

For a given string $w_1 \cdots w_T$, the DFA computes

$$\alpha \cdot M_{w_1} \cdots M_{w_T} \cdot \omega^\top \tag{25}$$

We say that $w \in L$ if and only if this expression evaluates to 1.

Having defined regular languages, we are in a position to show our main result:

Theorem 3. *For any regular language, there exists a 4-layer RWKV-7 model that recognizes it.*

Proof. We demonstrate that the RWKV-7 architecture can recognize strings in an arbitrary regular language L. Consider a string $w \in \Sigma^*$ and its membership in L. There exists a DFA (Definition 1) which recognizes L by evaluating (25). To prove that RWKV-7 can recognize any regular language, it is therefore sufficient to construct a RWKV-7 that evaluates (25).

Construction overview. A standard way to recognize a regular language would be to construct the transition matrices for each token and then multiply them to compute the final state. However, this does not work for RWKV-7 because because an arbitrary DFA transition can have rank $n = |Q|$ (i.e., the number of states), whereas a wkv head can only implement a simple elementary transition matrix per input token. A natural idea is to factor each DFA transition into *n elementary transition matrices* (Lemma 3), each of which can be directly implemented by wkv heads (Lemma 4). However, expanding DFA transitions in this way gives more elementary transition matrices than tokens, which means it cannot be directly implemented by RWKV-7.

Fortunately, a simple modification of this idea will allow us to implement regular language recognition in RWKV-7. Rather than expanding each transition matrix to n elementary matrices, we use the first three layers to convert blocks of n DFA transitions to products of n elementary matrices with the same product. The final layer then multiplies these elementary matrices to obtain the final state.

Details: information routing via residual stream and Layernorm. The output of each layer is stored in the residual stream of the architecture, in independent subspaces, which makes them all available to deeper layers. Since the outputs come from a finite set, they may be represented by one-hot encoding.

The input to each time mixing block therefore contains the outputs of all previous layers. First, a Layernorm is applied. Note that one-hot encodings have constant norm, which ensures that the Layernorm preserves the encoded information.

Next is the "weight preparation" part of the time mixing block, which takes input x_t encoding the output from previous layers, and constructs $r_t, w_t, \tilde{k}_t, v_t, \hat{\kappa}_t$ and a_t for the wkv heads. The weight preparation is sufficiently expressive to allow each of the 6 output variables to be an arbitrary linear transform of x_t. This allows selecting $r_t, w_t, \tilde{k}_t, v_t, \hat{\kappa}_t$ and a_t based on arbitrary outputs from previous layers. Therefore, the wkv heads are the main part of the construction.

WKV head construction. For any $1 \le t \le T$, the current position t can be split into $t = ln + \hat{t}$, for integers $0 \le l$ and $1 \le \hat{t} \le n$. We view the input sequence as blocks of length n, indexed by l.

First, we describe $l \ge 1$. Consider the product of DFA transitions $\tilde{M}_l = M_{w_{(l-1)n+1}} \ldots M_{w_{(l-1)n+n}}$. This product is also a DFA transition matrix (i.e., it has a single 1 per column). Hence, Lemma 3 allows us to factor $\tilde{M}_l = G_{l,1} G_{l,2} \ldots G_{l,n}$, where $G_{l,1}, \ldots, G_{l,n}$ are elementary transition matrices. Fix one such factorization for each possible DFA transition matrix. At position $t = ln + \hat{t}$, the wkv state in the fourth (final) layer is right-multiplied by the elementary transition matrix $G_{l,\hat{t}}$. Since $G_{l,\hat{t}}$ is uniquely defined by the last $2n$ tokens and position modulo $2n$, it can be computed in the third layer by Lemma 7, and fed to the fourth layer.

The first block $l = 0$ is handled as a special case. Again, we consider the fourth layer. At the first token, set $v_1 = e_1$ and $\tilde{k}_1 = \alpha$. All subsequent transitions $t \geq 2$ set $v_t = \tilde{k}_t = \mathbf{0}$, and implement identity transition matrices from Lemma 4. Then $\mathbf{wkv}_t = e_1^T \alpha$ for $1 \leq t \leq n$.

With this construction, for all $1 \leq t \leq T$, the fourth layer's wkv state becomes

$$\mathbf{wkv}_t = e_1^T \hat{\alpha}_t, \text{ where } \hat{\alpha}_t = \alpha M_{w_1} M_{w_2} \dots M_{w_{(l-1)n}} G_{l,1} G_{l,2} \dots G_{l,\hat{t}},$$

where as usual, empty products (such as for $l \leq 1$) evaluate to identity matrices.

Duplicate this construction into n wkv heads, where the ith applies receptance e_i. In combination, the wkv heads read out the whole vector $\hat{\alpha}_t$.

Final tokens. Consider $t = T$. To match the DFA evaluation formula from (25), $\hat{\alpha}_T$ can be multiplied by

$$\hat{\omega}_T^T = G_{l,\hat{t}+1} \dots G_{l,n} M_{w_{ln+1}} M_{w_{ln+2}} \dots M_{w_{ln+\hat{t}}} \omega^T.$$

Fortunately, this expression is a fixed function of the current position modulo $2n$ and the last $2n$ tokens, and can hence be found by a lookup in the third layer by Lemma 7. The final MLP on the final token can thus output the scalar product $\hat{\alpha}_T \hat{\omega}_T^T$, which is equal to (25), thus completing the construction. $\qquad \square$

Our construction uses MLPs to implement lookup tables with sizes on the order of $|\Sigma|^{2n}$, which may require MLP layers that are exponentially wide in the number of states of the original DFA.

D.3 Lemmas for Theorem 3

The proof for Theorem 3 requires many different lemmas, which are stated and proved in the following.

A single RWKV-7 wkv state transition cannot directly implement an arbitrary DFA transition. However, DFA transition matrices can be decomposed into a product of *elementary transition matrices* that can be directly simulated by wkv state transitions (cf. Lemma: 1):

Lemma 3. *Let M be a DFA transition matrix. I.e., M has shape $n \times n$, and contains a single 1 in each column. Then M can be factored into a product of n elementary transition matrices $G_1, \dots, G_n$. Specifically,*

$$M = G_1 G_2 \dots G_n,$$

where each of $G_1, \dots, G_n$ has one of the following forms:

1. *Identity matrix.*
2. *Swap matrix $x \leftrightarrow y$; an identity matrix with rows x and y swapped.*
3. *Copy matrix $x \rightarrow y$; an identity matrix with column y replaced by a copy of column x.*

Proof. We will greedily build M from the identity matrix by right-multiplying elementary transition matrices. Right-multiplying by the identity matrix does nothing, right-multiplying with a swap matrix $x \leftrightarrow y$ swaps columns x and y, and right-multiplying with a copy matrix $x \rightarrow y$ replaces column x by a copy of column y.

We use X to denote the current partial product of elementary transition matrices. Initially, X is the identity matrix, and the goal is to apply n transitions to make $X = M$. We proceed greedily in three stages.

1. Find a column c of X that differs from column c of M, but which matches a different column c' of M. If no such position exists, proceed to the next stage. Otherwise, right-multiply X by a swap matrix that swaps columns c and c'.
   Note that M and X differed in columns c and c' before the swap, but match in column c after the swap.
2. Find a column c where M and X differ. If no such column exists, proceed to the next stage. Otherwise, the previous stage has ensured that there exists a column c' of X, necessarily different from c, which contains a column identical to column c of M.
   Right-multiply by a copy matrix that replaces column c of X with column c' of X. Note that M and X differed in column c before the move, while they agree afterwards.

36

3. Right-multiply by identity matrices until X is the product of n elementary transition matrices. Initially, M and X differed in at most n columns. Each subsequent right-multiplication by an elementary transition matrix reduced the number of differing columns by at least one. Thus, stages 1 and 2 required at most n multiplications.

$\square$

Recall that the RWKV-7 wkv state is at token index t updated by multiplication with

$$A_t = \mathrm{diag}(w_t) - c\hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t),$$

with $c = 1$. To simplify the presentation of the core idea, we will instead present a construction with $c = 2$, and then show how to remove this assumption in Section D.3.

Lemma 4. *For any elementary transition matrix G (in the sense of Lemma 3), there exist n-dimensional vectors $\hat{\kappa}$ and $\vec{a}$, where $\|\hat{\kappa}\| = 1$, $\vec{a}$ has elements in $\{0, 1\}$, and*

$$G = \mathrm{diag}(w) - c\hat{\kappa}^T(\vec{a} \odot \hat{\kappa}),$$

where $c = 2$ and $w = \mathbf{1}$.

Proof. We use e_i to denote the n-dimensional vector with 1 at position i and 0 elsewhere.

1. Identity matrix: Select any length one vector $\hat{\kappa}$, for example $\hat{\kappa} = e_1$, and $\vec{a} = \mathbf{0}$.
2. Swap matrix; an identity matrix with rows x and y swapped: Select $\hat{\kappa} = (e_x - e_y)/\sqrt{2}$ and $\vec{a} = \mathbf{1}$.
3. Copy matrix; an identity matrix with column y replaced by a copy of column x: Select $\hat{\kappa} = (e_x - e_y)/\sqrt{2}$ and $\vec{a} = e_x$.

$\square$

We now move on to the explicit construction of the first three layers.

Lemma 5. *There is a 1-layer RWKV-7 which outputs whether the current position is first, and whether the current position is even or odd.*

Proof. RWKV-7 performs a token-shift operation before the wkv heads are reached. This token shift takes as input the last token, and can therefore detect whether a previous token exists.

The first layer's wkv state can track position parity by selecting $\tilde{k}_1 = v_1 = e_1$ for the first position $t = 1$, and subsequently letting $c = 2$, $w_t = a_t = \mathbf{1}$, $\hat{\kappa}_t = e_1$ and $\tilde{k}_t = v_t = \mathbf{0}$ for $t \geq 2$. This leads to $\mathbf{wkv}_1 = e_1^T e_1$ and subsequently $\mathbf{wkv}_t = \mathbf{wkv}_{t-1}(I - 2e_1^T e_1)$ for $t \geq 2$. Then $\mathbf{wkv}_t = e_1^T e_1$ for odd t and $\mathbf{wkv}_t = -e_1^T e_1$ for even t.

Receptance $r = e_1$ can then be used to read out the sign of the wkv state, which encodes the current position's parity.

The subsequent MLP can transform these two boolean outputs to an arbitrary format. $\square$

Lemma 6. *For any positive integer n, there is a 2-layer RWKV-7 that outputs the position modulo $2n$.*

Proof. We use Lemma 5 for the first layer, which tells the second layer whether the current position is first, and the parity of the current position.

At the first position, set $\tilde{k}_1 = v_1 = e_1$, such that $\mathbf{wkv}_1 = e_1^T e_1$. For all subsequent positions $t \geq 2$, set $c = 2$, $w_t = a_t = \mathbf{1}$ and $\tilde{k}_t = v_t = \mathbf{0}$. Furthermore, set $\hat{\kappa}_t = e_1$ for even t and $\hat{\kappa}_t = \cos(\pi/n)e_1 + \sin(\pi/n)e_2$ at odd t. Then $\mathbf{wkv}_T = e_1^T e_1(I - 2\hat{\kappa}_2^T \hat{\kappa}_2)\dots(I - 2\hat{\kappa}_T^T \hat{\kappa}_T)$. Note that for even $t \geq 2$, the matrix $(I - 2\hat{\kappa}_t^T \hat{\kappa}_t)(I - 2\hat{\kappa}_{t+1}^T \hat{\kappa}_{t+1})$ rotates the first two coordinates by an angle $2\pi/n$. Thus,

$$\mathbf{wkv}_t = \begin{cases} e_1^T\left(\cos(\pi(t-1)/n)e_1 + \sin(\pi(t-1)/n)e_2\right), & \text{if } t \text{ is odd} \\ e_1^T\left(-\cos(\pi(t-2)/n)e_1 + \sin(\pi(t-2)/n)e_2\right), & \text{if } t \text{ is even} \end{cases}.$$

The wkv heads are immediately followed by group normalization, which discards information about magnitudes. We therefore use $2n$ wkv heads of the type above, where the kth head applies receptance $r = \cos(\pi k/n)e_1 + \sin(\pi k/n)e_2$. The signs of these readouts, along with the parity from the first layer, can then be combined in the subsequent MLP layer to deduce the current position modulo $2n$. $\square$

Lemma 7. *Let $\tilde{t} \equiv t$ modulo $2n$ be the current position modulo $2n$, and let $w_t, w_{t-1}, \ldots, w_{t-(2n-1)}$ be the last $2n$ tokens. Define $w_t = |\Sigma| + 1$ for before-sequence tokens $t \leq 0$. Let $\Xi[\tilde{t}, w_t, w_{t-1}, \ldots, w_{t-(2n-1)}]$ be a lookup table that takes as key the current position modulo $2n$ and the $2n$ most recent tokens.*

Given inputs $\tilde{t}$ and $w_t, w_{t-1}, \ldots, w_{t-(2n-1)}$, there is a layer of RWKV-7 that simulates Ξ.

Moreover, there is a 3-layer RWKV-7 that simulates Ξ.

Proof. Recall that the wkv state is initialized to all zeros. We apply the wkv state update

$$\mathbf{wkv}_t = \mathbf{wkv}_{t-1}(I - e_{\tilde{t}}^T e_{\tilde{t}}) + e_{w_t}^T e_{\tilde{t}}.$$

This can be achieved by selecting $c \geq 1$, $w_t = \mathbf{1}$, $a_t = \frac{1}{c}\mathbf{1}$, $\tilde{k}_t = \hat{\kappa}_t = e_{\tilde{t}}$ and $v = e_{w_t}$.

In words, the state update replaces the $\tilde{t}$th column of the wkv state with e_{w_t}. Hence, the wkv state stores the last $2n$ tokens.

We make n such wkv heads, where the ith wkv head applies receptance $r = e_i$. This reads out the full state, which contains the last $2n$ tokens. The state and $\tilde{t}$ are fed into the subsequent MLP layer, which performs the lookup into Ξ.

Note that by Lemma 6, the first two layers can compute current position modulo $2n$ and the $2n$ most recent tokens. Hence, a 3-layer RWKV can compute Ξ. $\square$

Removing the assumption $c = 2$ Some of our constructions use $c = 2$, while the actual model uses $c = 1$. However, since the transition matrix is $A_t = \text{diag}(w_t) - c\hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t)$, halving both c and w_t simply causes A_t to be halved. This causes the wkv state to halve in magnitude at each token. However, since the wkv heads are immediately followed by group normalizations, the magnitude of the wkv state does not affect subsequent calculations. Additionally, since floating point numbers store a separate exponent, this rescaling only requires log-precision. The shrinking state could in principle be mismatched with the scales of v_t and $\tilde{k}_t$, but our constructions always satisfy $v_t = \tilde{k}_t = \mathbf{0}$ whenever $c \neq 1$ is required.

E   Additional Architectural and Training Details

Architecture Diagram   We provide a comprehensive architecture diagram (Figure 11) in order to help readers thoroughly understand our architecture design.

Parameters and Dimensions   Throughout this section, we denote by D the model dimension, L the number of layers, $h = D/D_h$ the number of heads, and V the vocabulary size. All models are trained with head size $D_h = 64$, i.e., each time-mixing has $h = D/64$ heads with dimension 64×64.

Pile models are trained with $V = 50304$ with the GPT-NeoX 20B tokenizer. World models are trained with $V = 65536$ with RWKV World tokenizer.

Model Name	L	D	State Size (WKV + Shift)	Parameters
RWKV7-World3-0.1B	12	768	589 824 + 18 432	191 034 624
RWKV7-World3-0.4B	24	1024	1 572 864 + 49 152	450 767 872
RWKV7-World3-1.5B	24	2048	3 145 728 + 98 304	1 527 404 544
RWKV7-World3-2.9B	32	2560	5 242 880 + 163 840	2 947 735 040

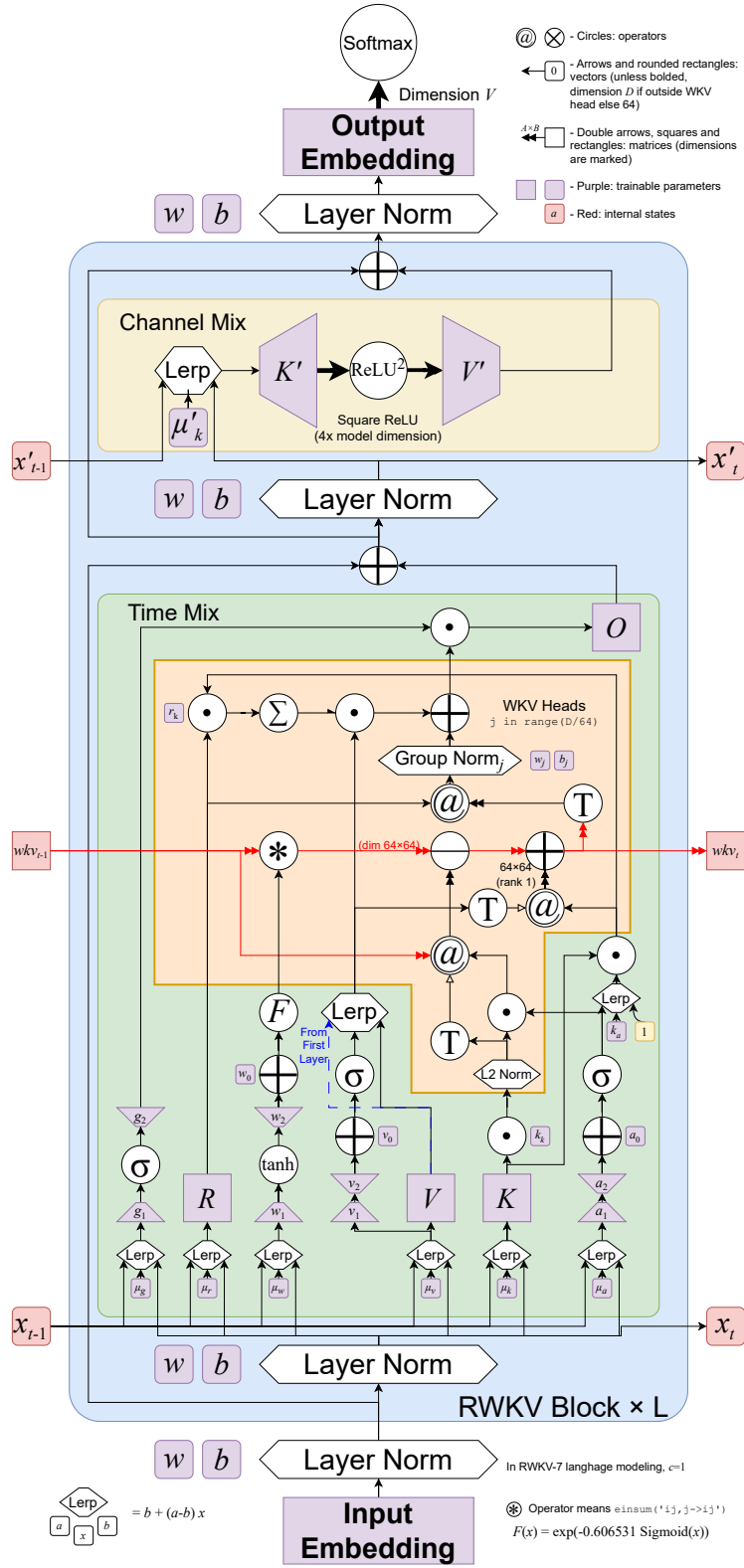Table 15: Released Goose models with parameters and state size.

Figure 11: The architecture of RWKV-7, drawn in detail.

RWKV-7 uses four low-rank MLPs for decay w, value residual v, in-context learning rate a and gate g respectively. The intermediate dimensions are listed in Table 16. These values are based on our mere speculation of how much information can be passed through.

Dimension (D)	d_w	d_a	d_v	d_g
768	64	64	32	128
1024	64	64	32	128
2048	96	96	64	256
2560	96	96	64	320
4096	128	128	96	480
6144	128	128	96	640

Table 16: Suggested Intermediate Dimensions for the low-rank MLPs for RWKV-7 models

The number of parameters for all RWKV-7 models can be computed by the formula:

$$\#(\text{Params}) = 2DV + 4D + LD\left(12D + 2\left(d_w + d_a + d_v + d_g\right) + 19\right) - (2Dd_v + D). \tag{26}$$

Where:

- The weights of the embeddings and head, and the Layernorms beside them, yield $2DV + 4D$ parameters;
- The weights of each layer yield $D\left(12D + 2\left(d_w + d_a + d_v + d_g\right) + 19\right)$ parameters, except for the first layer;
- The low-rank MLP for the value residual is not present in the first layer, subtracting $(2Dd_v + D)$ parameters.

Parameter Initializations Proper parameter initialization is crucial for ensuring training stability and achieving optimal performance for language models. RWKV-7 employs a carefully designed initialization strategy tailored to its architecture. The detailed initialization scheme is beyond the scope here but can be found in the official code repository. We emphasize that using the recommended initialization is essential for replicating the results in this paper. Deviations from the prescribed initialization may lead to performance degradation.

Dataset Loading The dataset used for pretraining consists of 3 119 194 079 123 tokens stored on disk, which are memory-mapped using the `mmap` mechanism. To ensure a diverse and pseudo-random sampling of training sequences, we employ a custom data loading strategy based on a mathematical function with desirable properties. Specifically, we utilize a pseudo-random number generator defined by the function $f(x) = ax^3$ over the finite field $\mathbb{Z}/p\mathbb{Z}$, where p is a prime number of the form $3n + 2$. This function is chosen because it is a bijection (full map) in $\mathbb{Z}/p\mathbb{Z}$, ensuring that all possible indices are eventually accessed exactly once within one epoch.

For pretraining with a sequence length of 4096, the relative address of the k-th sample is determined as:

$$\text{start\_address} = 4096 \cdot (ak^3 \mod p), \quad \text{end\_address} = \text{start\_address} + 4097.$$

Here, p is chosen as the largest prime of the form $3n + 2$ smaller than $\lfloor \text{dataset\_size}/4096 \rfloor$, yielding $p = 761 521 949$. The parameter a is set to an integer close to $0.618p$ that ensures good mixing properties of the generated addresses.

This approach guarantees both simple calculation and uniform access to the dataset while maintaining pseudo-randomness. By leveraging the properties of modular arithmetic and cubic mappings, we achieve a balance between computational efficiency and data diversity during pretraining.

Training Details All RWKV-7 models were trained under `bfloat16` format on nodes of 8× Nvidia H800. The AdamW optimizer was configured with $\beta_1 = 0.9$, $\beta_2 = 0.99$, $\epsilon = 1 \times 10^{-18}$, and a weight decay of 0.1 applied exclusively to linear layers and embedding weights. The choice of such a small ϵ value is motivated by the theory proposed by Molybog et al. (2023), which suggests that

Model	Phase	Nodes	Batch size	Proposed Initial LR	Final Loss
RWKV7-World3-0.1B	1	1	240×4096	6×10^{-4}	2.5290
RWKV7-World3-0.4B	1	1	240×4096	5×10^{-4}	
	2	2	480×4096	6×10^{-4}	2.2580
RWKV7-World3-1.5B	1	3	480×4096	4×10^{-4}	
	2	4	672×4096	4.5×10^{-4}	
	3	6	1152×4096	6.1×10^{-4}	1.9970
RWKV7-World3-2.9B	1	4	640×4096	4×10^{-4}	
	2	6	1008×4096	5×10^{-4}	
	3	7	1120×4096	5.4×10^{-4}	
	4	12	2016×4096	8×10^{-4}	1.8745

Table 17: Training Schedules and Batch Sizes

reducing ϵ can help stabilize training in large-scale models by ensuring that intermediate layers remain in a regime of active updates, thus mitigating sudden loss spikes and promoting smoother convergence.

The context length for pretraining was 4096 tokens. The base decay rate w_0 parameters are placed into a special 2x learning rate multiplier grouping.

Besides the traditional cosine learning rate decay schedule, we used a phased dynamic batch size scaling strategy inspired by the concept of critical batch size proposed by McCandlish et al. (2018) and similar to the approaches in Smith et al. (2018). Our strategy involves progressively increasing the batch size during training, accompanied by corresponding adjustments to the learning rate.

The detailed training schedules for different model sizes are listed in Table 17.

The learning rate undergoes a cosine decay schedule from the proposed initial learning rate at the beginning of the entire training run to the expected final learning rate of 1×10^{-5} at the end of the entire run, but the implied initial rate varies across phases.

This approach not only enhances training efficiency but also utilizes GPU resources economically. After smaller models complete their training, additional GPU resources become available for the later stages of training larger models. This cascading resource allocation ensures that computational power is dynamically reallocated, maximizing hardware utilization and reducing idle time.

We observe extremely stable training without any loss spikes in all four runs, indicating that the likelihood of encountering such spikes during the training of a very large RWKV-7 model is minimal.

See Figure 12 for the resulting learning rates and observed loss curves.

Despite the general stability of our loss curves, we did sometimes observe NaN loss across a single training step, which we theorize may be due to our use of such an extremely low AdamW ϵ. When this occurs, we rewind the training to the prior checkpoint, clear optimizer states, and continue from that point.

F   Additional Architecture Discussion

The following is a general overview of the RWKV-7 architecture and selected rationale for its design choices:

The RWKV-7 Time Mixer begins with a feature that has been in RWKV since its inception: token shift. Token shift is a variety of 1D short convolution that is intended to allow the model to create induction heads within a single layer. At its core, token shift is just a linear interpolation between the current and prior tokens on a per channel basis, where the amount per channel is a learned parameter. Many other modern models (e.g. Mamba) have begun including short convolutions

(a) 0.1B   (b) 0.4b
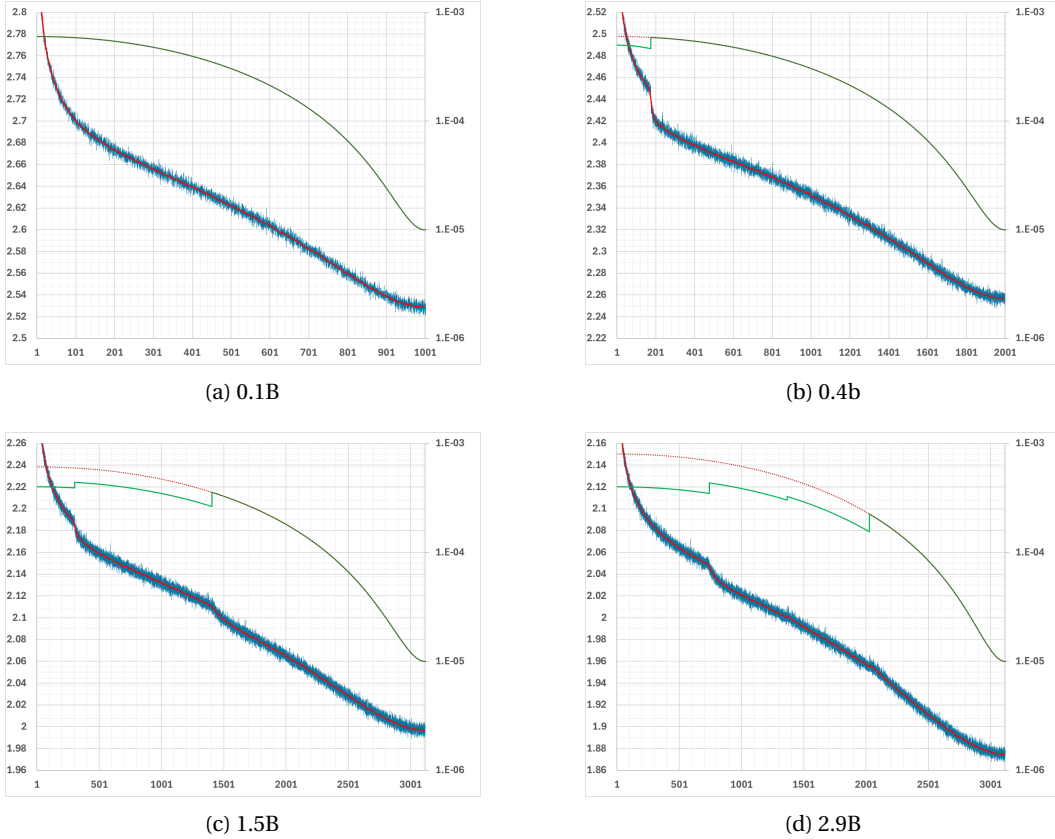
(c) 1.5B   (d) 2.9B

Figure 12: Training loss curve for RWKV-7 World models. Blue line: loss; Red line: smoothed loss; Green line: actual learning rate; Dotted red line: learning rate schedule.

before attention replacements (See also Dao AI Lab (2023)). RWKV-6 introduced an advanced new form of token shift that was data dependent. While we found that this was beneficial in terms of loss decrease per step, we made the judgement call that the improvement in training and inference efficiency was not worthwhile. Therefore, RWKV-7 includes only the simple token shift found in RWKV-4 and RWKV-5.

RWKV-7 time mixing follows the overall form of delta rule, applying an SGD-like mechanism to train the state at test time, such that when presented with a key it will respond with an appropriately matching value, like those it has been shown previously. This can be conceptualized as a simple order of operations: 1) decay the state 2) remove part of the old value located at the current key 3) add part of the new value into the current key. Then, a query (we call it receptance) is applied to the state in order to request the value located at a specific key. After that, the remaining operations normalize the returned values to keep numerical sizing consistent, apply gating, and recombine the heads to obtain the output.

In SGD terms, the decay found in models like RWKV-7 and Gated DeltaNet can be thought of as similar to weight decay. RWKV models since RWKV-4 have employed some form of vector-valued decay on the state in place of positional encoding. We continue this tradition with RWKV-7. While vector valued decay is harder to compute efficiently than its scalar valued equivalent, it brings a significant improvement in loss per step. Many other models, like Mamba-2, make the choice to maximize training efficiency and simplify their kernels by using scalar decay, purposely trading quality for speed.

We are forced to limit the maximum decay that can occur in a given time-step both in order to maintain training stability and to assist the creation of fast but numerically stable kernel implementations. We have many varieties of kernel available today, and are still working on new designs with enhanced efficiency and accuracy. Please see Parallelizing Linear Transformers with

42

the Delta Rule over Sequence Length (Yang et al., 2024c) and its accompanying blog post series for insightful details on many components of such algorithms. We hope to be able to reduce the decay limit further in future revisions.

This lower bound on the decay multiplier is expressed by the $\exp(-e^{-0.5}\sigma(d_t))$ formula for decay. It is a rearrangement of an original source formula $\exp(-\exp(-0.5 - \text{softplus}(d_t)))$. The outer $\exp(-\exp(x))$ in the original is nearly a flipped version of sigmoid, but with a better gradient. The $-0.5 - \text{softplus}(d_t)$ is a way of limiting the inputs to this sigmoid-like function to be less than -0.5, which results in the final decay being greater than 0.545. This means that decay can remove at most 45.5% of the pre-existing values in the wkv state from one timestep to the next. More can be then removed by the delta rule mechanism.

The in-context learning rate is usually equivalent to the learning rate in SGD. Ours is a bit less restrictive than the traditional delta rule or SGD would allow. We also make this data dependent and extend it to be vector-valued instead of being merely a scalar. This is allows each key channel to have its own learning rate, dependent on the current token. Note that in TTT, Gated DeltaNet, and Titans although the in-context learning rates are data dependent, they are only scalar valued.

Some of the design of RWKV-7 with regard to the in-context learning rate is theoretically motivated but may not be apparent from the equations. RWKV-6c, a later v6 sub-version with no trained models but which worked well experimentally, kept its state somewhat normalized using a new design. As mentioned early in this paper and consistent with the observations in Yang et al. (2024b); Chen et al. (2024a), there is a fundamental issue with linear attention numerically growing in an unbounded fashion, and the base RWKV-6 revision has this problem. RWKV-6c, however, defeats this issue by ensuring that the amount of key added to the state is never more than the amount removed by decay. It accomplishes this by pre-multiplying the key by one minus the decay before adding it into the wkv state.

Early versions of RWKV-7 attempted to use similar mathematical formulations to keep everything normalized. But experimentally we found that the model both did best and was most efficient when we allowed it enough mathematical leeway to make these decisions on its own, rather than enforcing them. So instead of pre-multiplying the key, we give the replacement key latitude in its learning rate via the replacement rate booster.

Similarly, the removal key is decoupled from the replacement key. Note that the removal key is normalized. This is important in delta rule variations, because otherwise the outer product of removal key with itself would cause unwanted changes in the amount removed due to the implicit squaring of its length during that process.

Another point that may not be clear upon first examination is the importance of RWKV-7 being outside of the TC^0 complexity class in which transformers, linear attention, and many varieties of SSMs lie. Consider a single layer of the Key Value Cache of a transformer. It is appended to upon each new token processed, and can never be changed. RWKV-7, however, can and does edit its state at each subsequent token. This can include simple operations like replacement, which can be viewed as functioning similarly to a transformer losing a KV Cache entry via some sparsity mechanism e.g. a sliding window. But it can also include complex operations like swapping two entries in the state; operations a transformer cannot do within its immutable KV Cache. These kinds of operations can be used to enact computations on the state itself, which lends greater computational abilities to RWKV-7 when processing a fixed set of inputs. You might think of the RWKV-7 state as being like an internal scratchpad.

There are easy problems that simply cannot be solved by transformers on fixed inputs because they lack this ability. One example is that if you give a transformer an ordered set of items and a list of which ones swap places, it will not be able to tell you which item ends up in which position at the end. Both architectures gain even more power when they are allowed extra outputs (e.g. chain of thought), becoming Turing complete. But this requires costly additional inference-time computation, whereas the RWKV-7 state does not.

A possible way to interpret the designation of RWKV-7 is training a model to learn (how to train an internal model). A WKV head can be viewed as a linear transformation that takes the input (receptance) r and outputs o. Every WKV head can be regarded as a linear model that can update

its weights as the context progresses. The entries of WKV states are exactly the parameters of these models.

After receptance is applied to the WKV state, we normalize the result on a per head basis. This has become common across many linear attention and post-linear-attention architectures. It is a way of ensuring that change in the numerical size of the state over time does not impact the model's ability to use the state. The original formulations of RWKV-4 used a denominator in place of normalization to achieve a similar effect, but it is costly, harder to code, and uses more memory.

The readout part of RWKV-7 differ from RWKV-6 by the addition of the per-head $\left(r_t(\rho \odot \tilde{k}_t)^T\right)$ scalar gating of v_t. The trainable parameter ρ resembles the design of "time-first" u term existing from RWKV-4 to RWKV-6, under the belief that the information of the current token deserves special treatment. The "time-first" term was fused inside the WKV kernel in previous architectures of RWKV, but we decide to extract this term out of the kernel to simplify the implementation of RWKV-7.

G Pseudocode For RWKV-7

```python
import math
import torch as th
import torch.nn.functional as F

def rwkv_timemix(params, x, vprime_0, shift_state, wkv_state, layer_id):
    B, T, C = x.shape        # batch_size, sequence_length, d_model
    N = wkv_state.shape[-1]  # head size
    H = C // N               # head count

    # weight preparation
    x_shifted   = th.cat([shift_state, x[:, :-1, :]], dim=1)
    shift_state = x[:, -1:, :]

    x_receptance = th.lerp(x, x_shifted, params.mu_r)
    x_decay      = th.lerp(x, x_shifted, params.mu_d)
    x_key        = th.lerp(x, x_shifted, params.mu_k)
    x_value      = th.lerp(x, x_shifted, params.mu_v)
    x_iclr       = th.lerp(x, x_shifted, params.mu_a)
    x_gate       = th.lerp(x, x_shifted, params.mu_g)

    r     = x_receptance @ params.W_receptance # BTC
    d     = params.decay_lora(x_decay)         # BTC
    k     = x_key @ params.W_key               # BTC
    vprime = x_value @ params.W_value          # BTC
    gate  = params.gate_lora(x_gate)           # BTC
    iclr  = params.iclr_lora(x_iclr).sigmoid() # BTC

    # 1st layer: return value to use in later layers, no interpolation
    if layer_id == 0:
        v = vprime_0 = vprime
    else:
        value_residual_gate = th.sigmoid(params.nu_lora(x_value))
        v = th.lerp(vprime, vprime_0, value_residual_gate)

    decay = th.exp(-math.exp(-0.5) * d.to(th.float).sigmoid())
    removal_k = k * params.removal_key_multiplier
    removal_k = F.normalize(removal_k.view(B,T,H,-1), dim=-1).view(B,T,C)
    replacement_k = th.lerp(k, k * iclr, params.iclr_mix_amt)

    # recurrence relation
    out = th.empty_like(x).view(B,T,C)
    for t in range(T):
        # single step of wkv state transition
        decay_t          = decay[:, t].view(B, H, N, 1)
        iclr_t           = iclr[:, t].view(B, H, N, 1)
```

44

```python
        removal_k_t     = removal_k[:, t].view(B, H, N, 1)
        replacement_k_t = replacement_k[:, t].view(B, H, N, 1)
        v_t             = v[:, t].view(B, H, N, 1)
        r_t             = r[:, t].view(B, H, N, 1)

        wkv_state = wkv_state * decay_t.mT - wkv_state @ removal_k_t @ (
                                    iclr_t * removal_k_t).mT
        wkv_state = wkv_state + v_t @ replacement_k_t.mT
        y = wkv_state @ r_t  # BHVK @ BHK1 = BHV1
        out[:,t] = y.view(B,C) # recombine heads

    # normalization
    y = F.group_norm(y.view(B * T, -1), num_groups=H, params.ln_x.weight,
                                    params.ln_x.bias, eps = H * 1e-5).
                                    view(B, T, -1)

    # bonus and output
    bonus = ((r*k*params.bonus_multiplier).sum(dim=-1, keepdim=True) * v)
    bonus = bonus.view(B,T,C)    # recombine heads
    out = out + bonus
    out = (out * gate) @ params.W_output # BTC

    return out, v0, shift_state, wkv_state


def rwkv_channelmix(x, shift_state):
    x_shifted = th.cat([shift_state, x[:, :-1, :]], dim=1)
    shift_state = x[:, -1:, :]
    xk = th.lerp(x, x_shifted, params.mu_x)
    k = params.W_k @ xk
    v = params.W_v @ th.relu(k).square()
    return v, shift_state


def rwkv_model(params, input_ids, state):
    x = params.embedding(input_ids)
    x = params.layer_norm_pre(x)

    v0 = None
    layer_id = -1
    for layer in params.layers:
        layer_id = layer_id + 1
        dx, v0, state.timemix_shiftstate, state.timemix_wkvstate = \
                                    rwkv_timemix(
            layer.time_mix,
            layer.layer_norm_timemix(x),
            v0,
            state.timemix_shiftstate,
            state.timemix_wkvstate,
            layer_id
        )
        x = x + dx
        dx, state.chanmix_shiftstate = rwkv_chanmix(
            layer.channel_mix,
            layer.layer_norm_chanmix(x),
            state.chanmix_shiftstate
        )
        x = x + dx

    x = params.layer_norm_out(x)
    logits = params.head(x)
    return logits, state
```

H   PyTorch code For Naive WKV7 Kernel (Forward and Backward)

```python
class WKV7_Kernel(nn.Module):
    def __init__(self):
        super().__init__()

    def forward(self, r, w, k, v, a, b, state):
        r = r.view(B, T, H, N)
        k = k.view(B, T, H, N)
        v = v.view(B, T, H, N)
        a = a.view(B, T, H, N)
        b = b.view(B, T, H, N)
        self.state_cache = torch.zeros((B, T+1, H, N, N))
        self.state_cache[:, 0, :] = state
        w = torch.exp(-torch.exp(w.view(B, T, H, N)))
        out = torch.zeros((B, T, H, N))
        for t in range(T):
            kk = k[:, t, :]
            rr = r[:, t, :]
            vv = v[:, t, :]
            aa = a[:, t, :]
            bb = b[:, t, :]
            state = (
                state * w[: , t, :, None, :]
                + torch.einsum('bhik,bhk,bhj->bhij', state, aa, bb)
                + torch.einsum('bhj,bhi->bhij', kk, vv)
            )
            self.state_cache[:, t+1, :] = state
            out[:, t, :] = torch.einsum('bhj,bhij->bhi', rr, state)
        return out, state

    def backward(self, r, w0, k, v, a, b, gout, gstate):
        gout = gout.view(B, T, H, N)
        gr = torch.zeros((B, T, H, N))
        gw = torch.zeros((B, T, H, N))
        gk = torch.zeros((B, T, H, N))
        gv = torch.zeros((B, T, H, N))
        ga = torch.zeros((B, T, H, N))
        gb = torch.zeros((B, T, H, N))
        w = torch.exp(-torch.exp(w0.view(B, T, H, N)))
        for t in range(T-1, -1, -1):
            gr[:, t, :] = torch.matmul(
            gout[:,t,:,None,:], self.state_cache[:,t+1,:]).squeeze(-2)
            gstate.add_(torch.matmul(
            gout[:,t,:,:,None], r[:,t,:,None,:]))
            gk[:, t, :] = torch.matmul(
            v[:,t,:,None,:], gstate).squeeze(-2)
            gv[:, t, :] = torch.matmul(
            gstate, k[:,t,:,:,None]).squeeze(-1)
            ga[:, t, :] = torch.einsum(
            'bhik,bhj,bhij->bhk',
            self.state_cache[:, t, :], b[:, t, :], gstate)
            gb[:, t, :] = torch.einsum(
            'bhik,bhk,bhij->bhj',
            self.state_cache[:, t, :], a[:, t, :], gstate)
            gw[:, t, :] = torch.einsum(
            'bhij,bhij->bhj',
            self.state_cache[:, t, :], gstate)
            gstate      = torch.einsum(
            'bhj,bhij->bhij', w[:, t, :], gstate) \
            + torch.einsum(
            'bhk,bhj,bhij->bhik', a[:, t, :], b[:, t, :], gstate)
        gw = -torch.exp(w0-torch.exp(w0)) * gw
```

46

```
63        return gr, gw, gk, gv, ga, gb, gstate
```

I Board Game Modeling

Previous research (Schultz et al. (2024); Topsakal et al. (2024)) has shown that board games can serve as potentially effective tools for evaluating a model's capabilities. As an RNN with powerful state tracking abilities, RWKV-7 is highly suitable for board game modeling and conducting extended searches directly within context to find better strategies. As an early exploration, we tested RWKV-7's board game modeling capabilities and its ability to perform in-context search on the game of Othello (Reversi). We found an expanded formulation of RWKV-7 to be useful for this task, which we designate as RWKV-7a: $S_t = S_{t-1}\text{diag}(w_t)(I - c\hat{\kappa}_t^T(a_t \odot \hat{\kappa}_t)) + v_t^T k_t$ This formula allows the full range of (-1,1) eigenvalues when $c = 2$.

Data We designed training data that guides the model to predict legal moves, evaluate these moves, and perform Alpha-Beta pruning. Each training sample consists of three components:

- **Input section**: Captures the game state, including the current board position, active player, and search parameters (tree depth and width settings).
- **Reasoning section**: Varies with search settings:
    - Without search (depth or width = 1): Lists legal moves and their evaluations
    - With search (depth and width > 1): Performs Alpha-Beta pruning to find optimal moves

    All move evaluations in the reasoning section were generated using the Egaroucid engine (Yamana, 2025).
- **Output section**: This contains the final move decision and displays the resulting board position after implementing the move.

An example of our training data is shown in sample 1.

```
1  <input>
2  . . O O * * . .
3  * . O O O * * .
4  * * * O * * O O
5  * * * * * O O O
6  * O * O O * O O
7  * * O O O * O .
8  * . O O O O . .
9  . . . . O O . .
10 NEXT o
11 MAX_WIDTH-2
12 MAX_DEPTH-2
13 </input>
14 <reasoning>
15 Possible moves and score: g1 -19 h1 -01 b2 -08 h2 -23 b7 -12 g7 -09
16 <stack>
17 Remaining_Depth:2
18 Max_Node Alpha: -in Beta: +in Best: -- Current: h1 -01 Unexplored: b2 -08
19 </stack>
20 => Search next node
21 [Depth limit not reached]
22 <board>
23 . . O O * * . O
24 * . O O O * O .
25 * * * O * O O O
26 * * * * O O O O
27 * O * O O * O O
28 * * O O O * O .
29 * . O O O . . .
30 . . . . O O . .
31 </board>
```

```
32 NEXT *
33 Possible moves and score: b1 +02 b2 +05 h2 +10 h6 +03 h7 +08 c8 +06 d8
       +01 g8 +09
34 [Current player has legal moves]
35 [Internal node - expand]
36 <stack>
37 Remaining_Depth:1
38 Min_Node Alpha: -in Beta: +in Best: -- Current: d8 +01 Unexplored: b1 +02
39 Max_Node Alpha: -in Beta: +in Best: -- Current: h1 -01 Unexplored: b2 -08
40 </stack>
41 => Search next node
42 [Depth limit reached - evaluate all leaves]
43 [Updated stack]
44 <stack>
45 Remaining_Depth:1
46 Min_Node Alpha: -in Beta: +01 Best: d8 Current: -- --- Unexplored:
47 Max_Node Alpha: +01 Beta: +in Best: h1 Current: b2 -08 Unexplored:
48 </stack>
49 => Search next node
50 [Depth limit not reached]
51 <board>
52 . . O O * * . .
53 * O O O O * * .
54 * O O O * * O O
55 * O * O * O O O
56 * O * O O * O O
57 * * O O O * O .
58 * . O O O O . .
59 . . . . O O . .
60 </board>
61 NEXT *
62 Possible moves and score: a1 -07 b1 +13 h6 -01 b7 -08 g7 +08 h7 -02 c8
       +01 d8 -03 g8 +04
63 [Current player has legal moves]
64 [Internal node - expand]
65 <stack>
66 Remaining_Depth:1
67 Min_Node Alpha: +01 Beta: +in Best: -- Current: b7 -08 Unexplored: a1 -07
68 Max_Node Alpha: +01 Beta: +in Best: h1 Current: b2 -08 Unexplored:
69 </stack>
70 => Search next node
71 [Depth limit reached - evaluate all leaves]
72 [Updated stack]
73 <stack>
74 Remaining_Depth:1
75 Min_Node Alpha: +01 Beta: -08 Best: b7 Current: -- --- Unexplored:
76 Max_Node Alpha: +01 Beta: +in Best: h1 Current: -- --- Unexplored:
77 </stack>
78 [End of search]
79 > Playing h1
80 </reasoning>
81 <output>
82  h1
83 . . O O * * . O
84 * . O O O * O .
85 * * * O * O O O
86 * * * * O O O O
87 * O * O O * O O
88 * * O O O * O .
89 * . O O O O . .
90 . . . . O O . .
91 </output>
```

Code Listing 1: Training Data Example

Training We trained RWKV-7 models with 9M and 26M parameters respectively on 6 million samples. By tracking the loss across different token types during training (figure 13), we noticed that the model first mastered output formatting, then developed board state tracking capability, and continuously improved its evaluation accuracy throughout training.
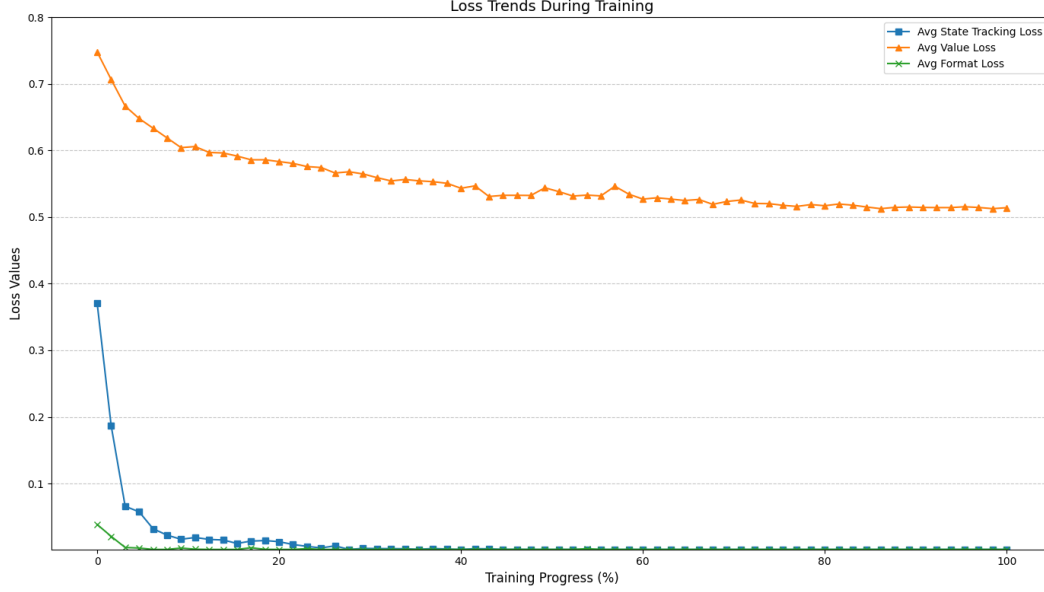

Figure 13: Reversi Training loss for different token types

Evaluation We control the model's thinking budget by setting the width and depth of Alpha-beta pruning, and test the win rate against baseline a model (depth=1, width=1) under different budgets. As shown in figure 14, by increasing the testing budget, RWKV-7 can effectively search for better strategies, demonstrating a positive test-time scaling law on this task.
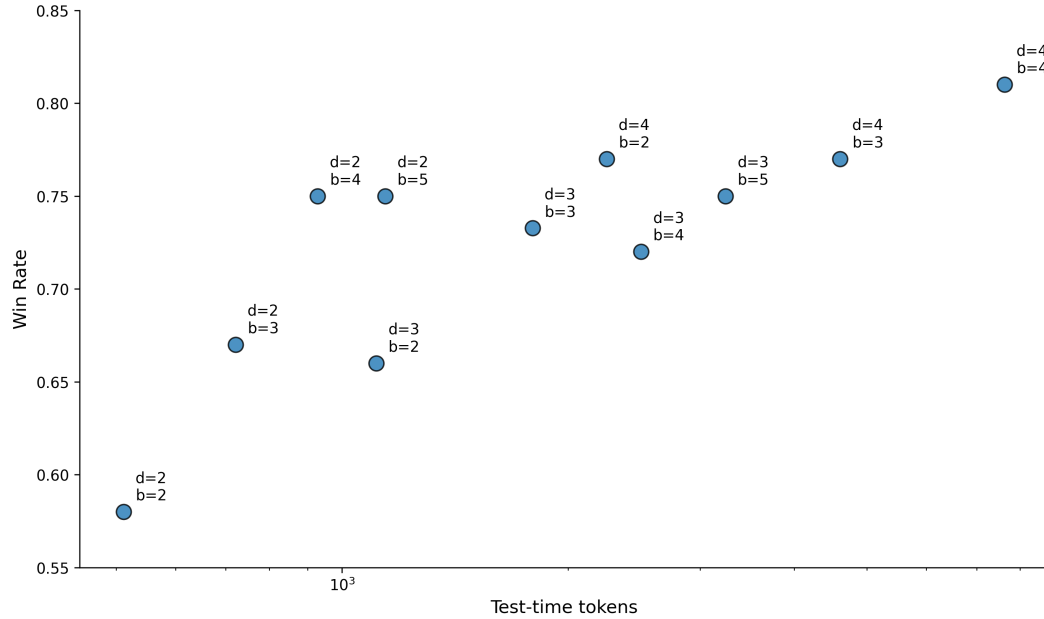

Figure 14: Reversi Token consumption and win rates under different search configurations

J State Inspections

We inspected the WKV matrix states of RWKV-7 and compared them with those of RWKV-5 and RWKV-6. We analyzed the following aspects:

1. Visualization example of WKV state matrices, to better understand the structure and behavior of the matrices.
2. Root Mean Square (RMS) of matrix elements, to assess the numerical stability of those WKV matrices.
3. Stable Rank (SR) of the matrices (Rudelson & Vershynin, 2007), defined as the square of (the Frobenius norm divided by the spectral norm):

$$\text{SR}(A) := \left(\frac{\|A\|_F}{\|A\|_2} \right)^2.$$

This serves as a rough measure to the effective amount of information of the states.

For this analysis, we selected 10 samples from the validation set of the PG19 dataset (Rae et al., 2019), ensuring that each sample had a sequence length of at least 8192 tokens. We tested the 1.5B parameter versions of RWKV-5, RWKV-6, and RWKV-7, plotting on the appearance, stable rank and RMS values of their WKV matrices.
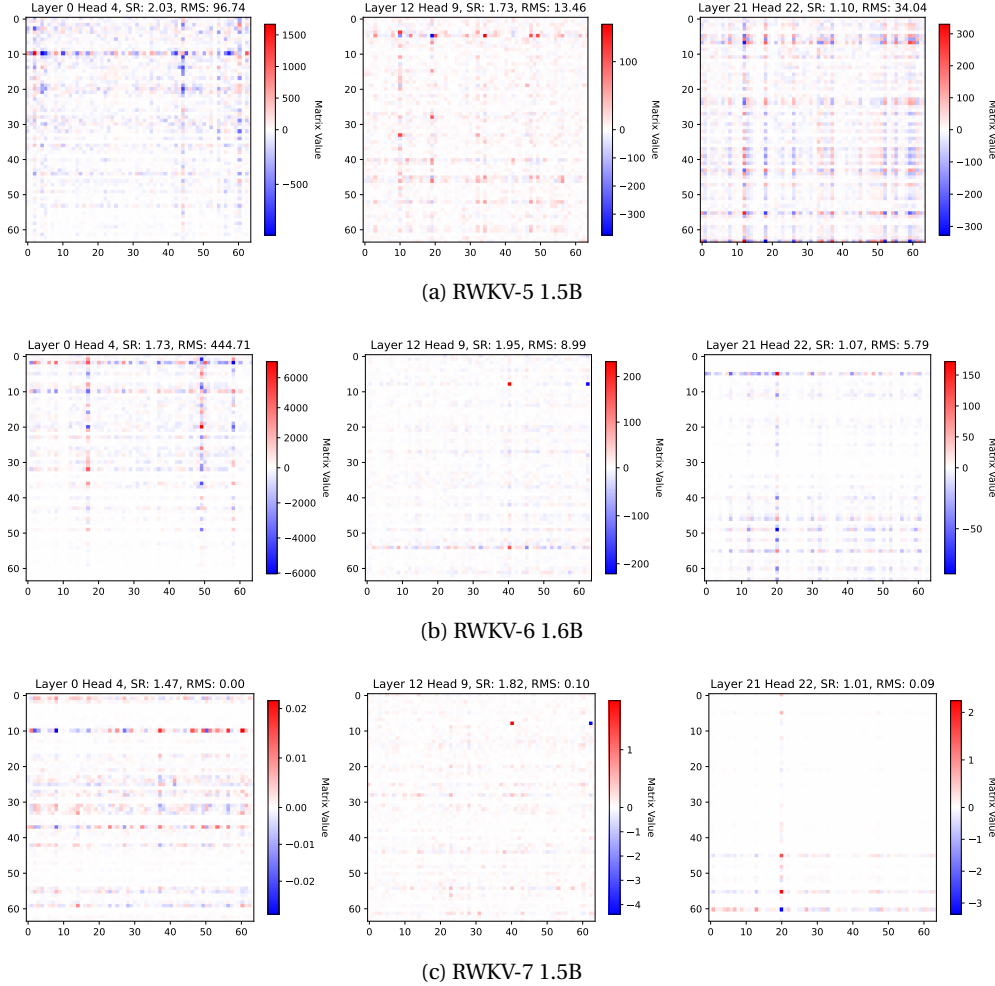

(a) RWKV-5 1.5B

(b) RWKV-6 1.6B

(c) RWKV-7 1.5B

Figure 15: Visualization example of RWKV's WKV matrices.

We observed that the WKV states of RWKV-7 had significantly smaller RMS values compared to RWKV-5 and RWKV-6. The entries of the WKV matrix in RWKV-7 were consistently of order $O(1)$

(i.e., no outliers, and does not grow over context length), whereas RWKV-5 and RWKV-6 constantly produced outliers on the order of thousands (see Figures 15 and 16). This indicates that RWKV-7 has better numerical stability during training and inference.
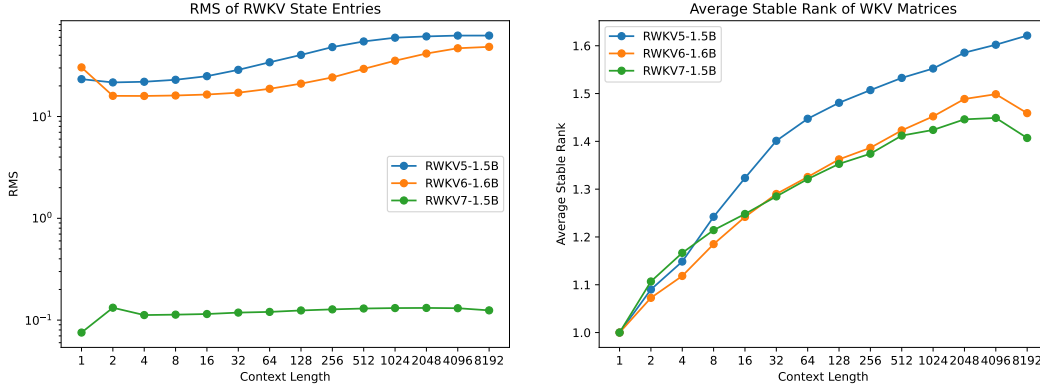

Figure 16: The global RMS and average stable rank of WKV matrices, plotted over sequence length.

Interestingly, the stable rank of the WKV matrix in RWKV-7 has shown to be lower than that of RWKV-5 and RWKV-6 for context longer than 32. A lower stable rank typically suggests that the matrix contains less information or has a more compressed representation. However, this observation appears to contradict the experimental results showing that RWKV-7 performs better on tasks requiring long-term memory. We hypothesize that this contradiction can be explained by RWKV-7's enhanced state evolution mechanism, which enables RWKV-7 to achieve stronger information compression and utilization capabilities and allowing it to maintain important information in a more compact form. This dual capability likely contributes to both the reduced stable rank and the improved performance on memory-intensive tasks.

K Ablation Experiments

The Pile   To demonstrate the architectural advantages of RWKV-7, we conducted ablation experiments by training models of three different sizes—168M, 421M, and 1.47B parameters—on the full Pile dataset (Gao et al., 2020).

Model	Tokens (B)	lmb.o ppl ↓	lmb.o acc ↑	hella acc_n ↑	piqa acc ↑	arcE acc ↑	arcC acc ↑	glue acc ↑	WG acc ↑	sciq acc ↑	avg acc ↑
RWKV4-169M-Pile	332	29.2	33.2	32.2	64.8	47.1	19.9	47.6	51.2	77.6	46.7
Pythia-160M	300	37.3	35.4	30.3	62.3	43.6	19.5	46.5	51.3	75.4	45.5
Mamba-130M	300	16.0	44.3	35.3	64.5	48.0	19.7	48.5	52.1	78.2	48.8
Mamba2-130M	300	16.8	43.9	35.3	64.9	47.4	**20.9**	45.8	**52.6**	81.0	49.0
RWKV6-173M-Pile	332	16.0	44.5	34.9	64.4	**48.3**	19.7	48.9	51.9	80.6	49.2
RWKV7-168M-Pile	332	**14.2**	**45.7**	**36.9**	**65.5**	47.9	19.7	**49.1**	52.4	**81.6**	**49.8**
RWKV4-430M-Pile	332	13.1	45.1	40.8	67.7	52.8	24.1	49.4	52.0	80.7	51.6
Pythia-410M	300	10.8	51.6	40.6	66.7	51.9	21.4	44.1	53.3	81.5	51.4
Mamba-370M	300	8.1	55.6	46.5	69.5	55.0	25.0	46.8	55.5	84.5	54.8
Mamba2-370M	300	8.0	55.9	46.9	**70.5**	54.8	**25.1**	48.1	55.4	85.3	55.2
RWKV7-421M-Pile	332	**7.2**	**57.9**	**48.0**	69.3	**56.3**	23.5	50.3	**56.4**	**85.9**	**56.0**
RWKV4-1.5B-Pile	332	7.1	56.4	52.8	72.2	60.7	24.9	**50.5**	54.3	85.8	57.2
Pythia-1.4B	300	6.1	61.7	52.0	70.8	60.5	26.1	47.7	57.5	86.6	57.9
Mamba-1.4B	300	5.0	65.0	59.1	**74.2**	**65.5**	29.8	46.2	61.4	87.3	61.1
Mamba2-1.3B	300	5.0	65.6	59.9	73.2	64.2	29.9	46.1	61.0	89.8	61.2
RWKV7-1.47B-Pile	332	**4.8**	**67.0**	**61.8**	73.6	64.9	**30.2**	48.0	**64.4**	**91.1**	**62.6**

Table 18: English Focused Benchmarks, including LAMBADA (**lmb.o**) (Paperno et al., 2016), Hellaswag (**hella**) (Zellers et al., 2019), PIQA (Bisk et al., 2020), AI2 ARC (**arcE, arcC**) (Bhakthavatsalam et al., 2021), GLUE (Wang et al., 2018), Winogrande (**WG**) (Sakaguchi et al., 2021), SciQ (Welbl et al., 2017).

These results highlight the consistent improvements brought by the RWKV-7 architecture over earlier RWKV models, even when trained on the same dataset. As all RWKV models shown were trained under identical configurations and dataset, this underscores the inherent architectural advantages of RWKV-7 over its predecessors. Notably, the performance gap sustains as the model size increases, suggesting that RWKV-7 may scale more effectively than its predecessors.

Architecture Choice Ablations   We ran a series of ablation experiments to validate the various improvements made in RWKV-7 versus some of the more restrictive choices seen in other DeltaNet and post-DeltaNet related work. These improvements were:

- using a vector-valued decay w instead of a scalar-valued decay;
- using a vector-valued in-context learning rate a instead of a scalar-valued rate;
- using different removal κ and replacement $\tilde{k}$ keys, instead of the same for both; and
- adding the bonus term $u_{t,j} v_{t,j}$ to the output step (Equation 20).

We trained a small 6-layer, $d_{model} = 768$ Goose model on the 1.6B token minipile (Kaddour, 2023) dataset at context length 512 and obtained the loss results shown in Table 19.

Model	Training Loss	Validation Loss
Goose	2.834	2.541
Goose, scalar decay	2.873	2.609
Goose, scalar in-context learning rate	2.843	2.591
Goose, same removal/replacement keys	2.840	2.560
Goose, no bonus term	2.841	2.588

Table 19: Ablation Results for 6 layer 768 dimension Goose model

L   Parameters Statistics

In Section 4, the actual ranges and statistical metrics of the parameters within the trained model are not specified. To facilitate a better understanding of the role of these parameters in the practical models, this appendix provides empirical statistical metrics of selected parameters from the released RWKV-7 model.
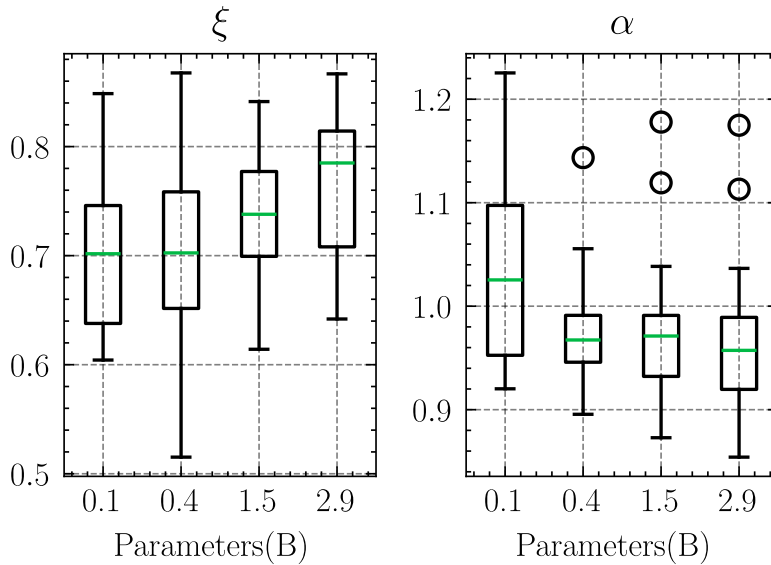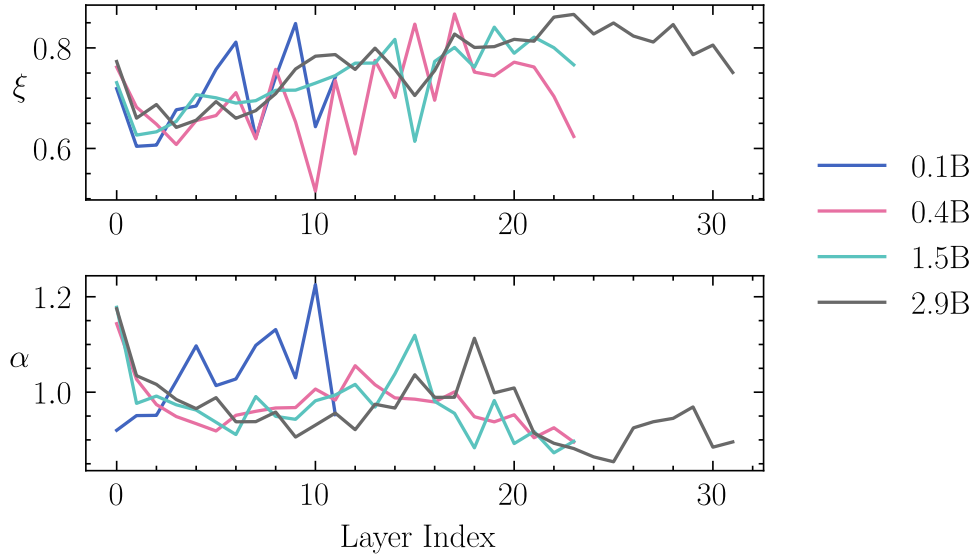

Figure 17: Box Plots of ξ and α Across Models

Figure 18: Mean ξ and α Across Layers for Different Models
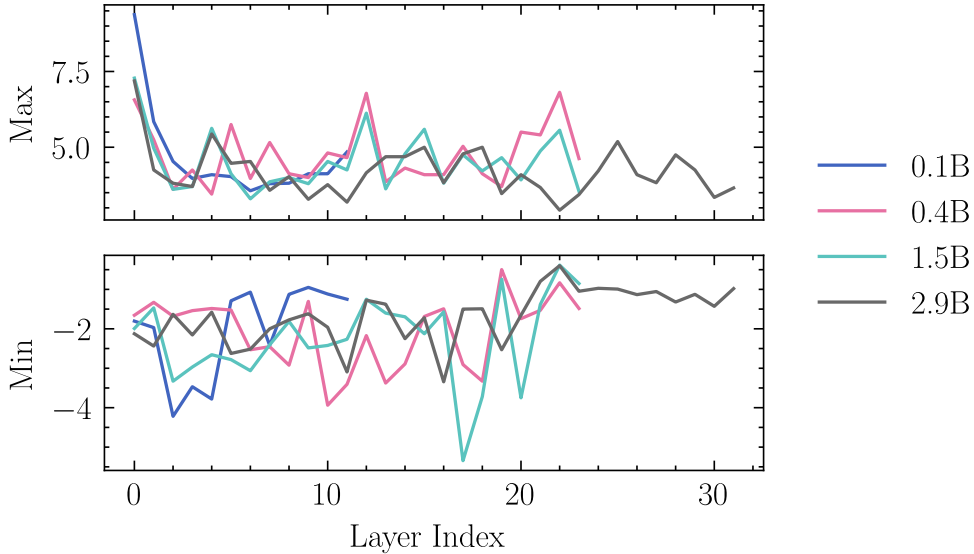

Figure 19: Maximum and Minimum of ξ Across Layers in Different Models

M Initial Token Sensitivity

In our evaluation of LAMBADA (Paperno et al., 2016), we found that RWKV-7's performance varies significantly under different settings. After ruling out precision issues, we investigated the consistency of the input and discovered that omitting the special token `<|endoftext|>` at the beginning of the input caused substantial and statistically significant differences in perplexity (PPL) and accuracy (ACC) for some RWKV-7 models.

Previous research highlights that Transformer models are sensitive to special tokens, and fine-tuning these tokens can yield notable improvements (Yang et al., 2023b). However, to our knowledge, no quantitative study has examined this effect systematically.

We analyzed the cases with the largest performance discrepancies, and identified a key pattern:
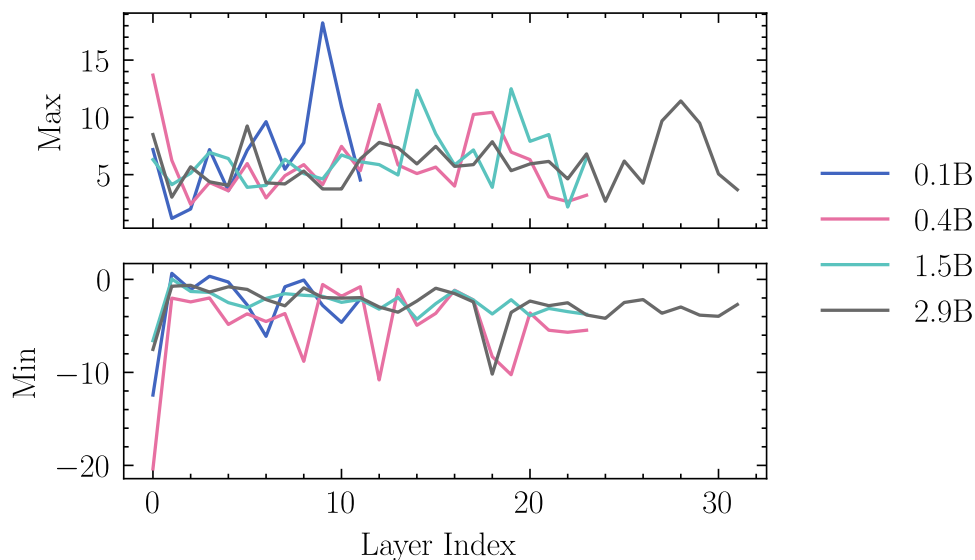
Figure 20: Maximum and Minimum of α Across Layers in Different Models
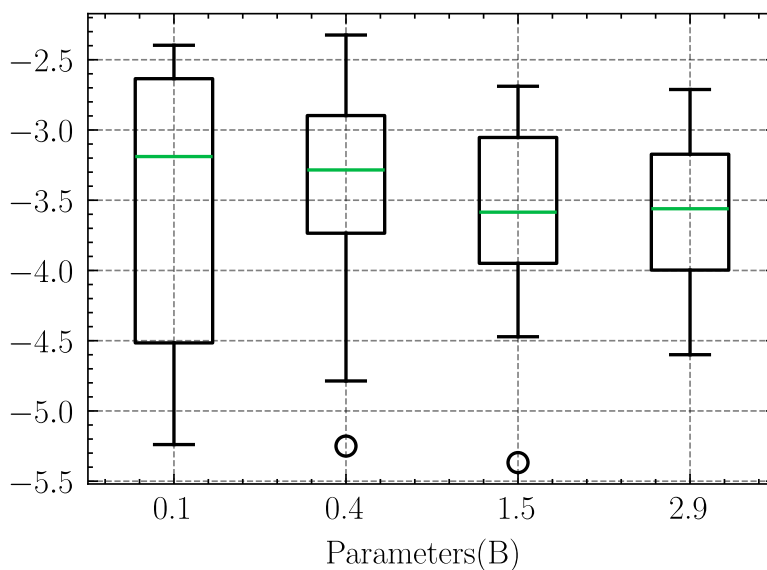

Figure 21: Box Plots of biases of d_t Across Models

- The answer appears as the first word of the paragraph.
- This first word does not reappear elsewhere in the text.

This suggests that the model may struggle to retain the first token in memory.

In the LAMBADA test set, we identified 142 such examples out of a total of 5153 questions. One example is:

> Beth smoothed her wiry half-black, half-gray hair from her makeup-free face. In New Mexico, the natural look was common. Standing next to Cindy Fanucci, she felt like a disaster. She hid her ragged nails under the sleeves of her sweatshirt.
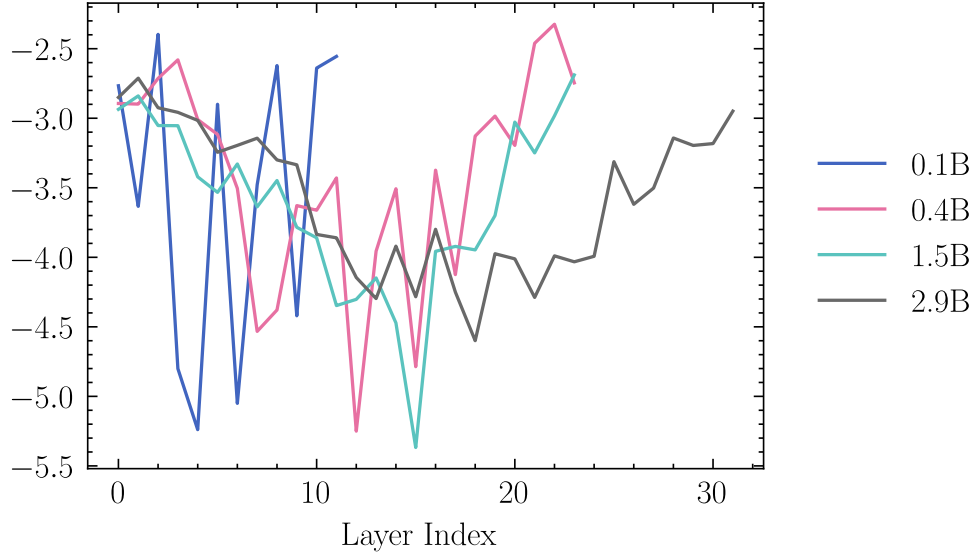> "Hi, I'm Cindy. It's so nice to meet you, **Beth.**"

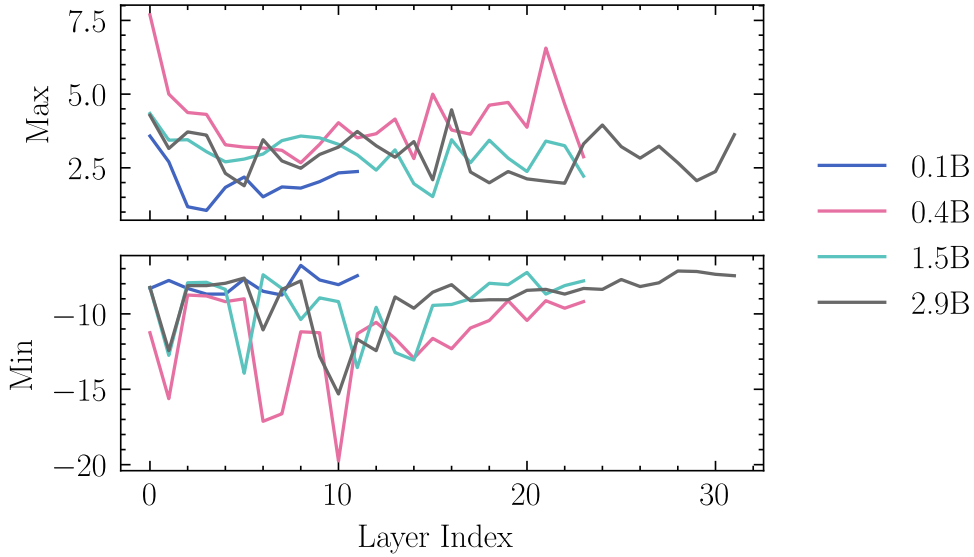Figure 22: Mean biases of d_t Across Layers for Different Models

Figure 23: Maximum and Minimum of biases of d_t Across Layers in Different Models

The following table summarizes the performance of different models with and without the padding token `<|endoftext|>`:

The results indicate that the inclusion of the `<|endoftext|>` token improves the performance of RWKV-7 very significantly, especially for small models (e.g., RWKV-7 0.1B). This finding highlights the importance of proper state initialization and context setting for RNN-based architectures like RWKV. Unexpectedly, we also found that two consecutive `<|endoftext|>` tokens at the beginning can further improve the performance of RWKV-7, despite that `<|endoftext|>` never appears consecutively in the training corpus.

However, for Transformer-based models such as Qwen2.5-0.5B, we observe that the impact of the `<|endoftext|>` token is less pronounced, suggesting that these models may have better mechanisms for attending to the initial token.

Model	EOS padding	PPL	ACC (%)	Significance
RWKV7 World 0.1B	0	357	9.2	***
	1	16.4	36.6	
RWKV7 World 0.4B	0	42.7	28.9	***
	1	7.25	48.6	
SmolLM2 360M	0	21.1	39.4	*
	1	9.17	49.3	
Qwen2.5 0.5B	0	12.2	47.9	NS
	1	7.97	54.9	

Table 20: Performance comparison of different models with and without the `<|endoftext|>` token in the partial set of 142 samples from LAMBADA. Significance levels: $^{*}p < 0.05$, $^{**}p < 0.01$, $^{***}p < 0.001$, NS = Not Significant.

As a result, for optimal performance when prompting RWKV-7, we recommend always including the `<|endoftext|>` token at the beginning of the prompt. For example, if you plan to use RWKV-7 as a chat assistant, consider the following structured prompt format:

```
<|endoftext|>User: <Your Question>

Assistant: <Assistant Answer>

User: <Another Question>

Assistant:
```