FIXED-POINT RNNS: FROM DIAGONAL TO DENSE IN A FEW ITERATIONS

Sajad Movahedi[*1,2], **Felix Sarnthein**[*1,2,3], **Nicola Muça Cirone**[4], **Antonio Orvieto**[1,2]
[1] ELLIS Institute Tuebingen, [2] MPI for Intelligent Systems, Tuebingen AI Center
[3] Department of Computer Science, ETH Zurich
[4] Department of Mathematics, Imperial College London
{sajad.movahedi, felix.sarnthein}@tue.ellis.eu

ABSTRACT

Linear recurrent neural networks (RNNs) and state-space models (SSMs) such as Mamba have become promising alternatives to softmax-attention as sequence mixing layers in Transformer architectures. Current models, however, do not exhibit the full state-tracking expressivity of RNNs because they rely on channel-wise (i.e. diagonal) sequence mixing. In this paper, we propose to compute a dense linear RNN as the fixed-point of a parallelizable diagonal linear RNN in a single layer. We explore mechanisms to improve its memory and state-tracking abilities in practice, and achieve state-of-the-art results on the commonly used toy tasks A_5, S_5, copying, and modular arithmetics. We hope our results will open new avenues to more expressive and efficient sequence mixers.

1 INTRODUCTION

State-space models (SSMs) and other new efficient recurrent token mixers are becoming a popular alternative to softmax attention in language modeling (Gu & Dao, 2023) as well as other applications such as vision (Liu et al., 2024), audio (Goel et al., 2022) and DNA processing (Nguyen et al., 2024). Inspired by linear input-controlled filtering, these models can be expressed as carefully parametrized linear recurrent neural networks (RNNs) with input-dependent, diagonal state transition: $h_t = \text{diag}(\mathbf{a}_t)h_{t-1} + \mathbf{B}_t x_t$.

Compared to classical RNNs such as LSTMs, the recurrence from the previous hidden state h_{t-1} to the current h_t is linear and its coefficient $\mathbf{a}_t$ does not depend on the hidden states. These factors allow SSMs such as Mamba to be computed through efficient parallel methods during training. At test time, they are faster than classical Transformers on long sequences due to their recurrent nature.

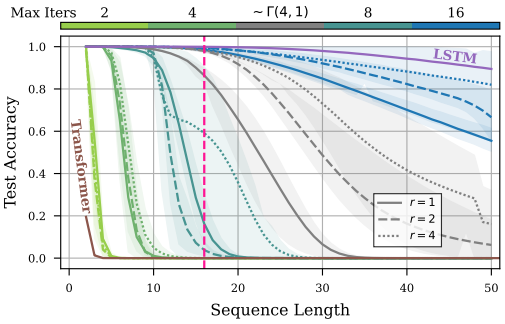

Figure 1: *Sequence length generalization at training length 16 (pink) for state-tracking on A_5, with Transformer (brown) and LSTM (purple) as lower/upper bounds. Our Fixed-Point RNN (Bedouin) is trained at different maximum number of fixed-point iterations ℓ_{max}: between 2 (green) and 16 (blue), or sampled from $\Gamma(4,1)$ with mean 4 (gray).* ***Increasing the number of fixed-point iterations turns the linear RNN from diagonal to dense in a few iterations.***

Though modern linear RNNs have shown promise in practice, recent theoretical studies suggest that using dense transition matrices (i.e. replacing $\text{diag}(\mathbf{a}_t)$ with a dense $\mathbf{A}_t$) could present an opportunity to improve expressivity and unlock performance on challenging tasks. In particular, Cirone et al. (2024b) prove that dense linear SSMs/RNNs are endowed with the theoretical expressivity of classical non-linear RNNs such as LSTMs. Furthermore, Merrill et al. (2024) show that this is particularly useful in state-tracking applications where models are expected to maintain and extrapolate a complex state of the world. While state-tracking is naturally expressed in non-linear RNNs, it is known to be unavailable to channel-wise sequence mixers such as SSMs or Transformers (Merrill & Sabharwal, 2023). Finally, DeltaNet demonstrated that controlled, non-diagonal state updates significantly improve learning of compressed associative memories (Schlag et al., 2021a; Yang et al., 2024b).

[*]Equal contribution.

Although linear recurrences are parallelizable across sequence length (Martin & Cundy, 2018), parallelizing dense RNNs efficiently is not trivial. Because channels and tokens need to be mixed jointly, structured transition matrices such as head-wise WY-representations (Yang et al., 2024b), are required for efficient implementations. But mixing only within heads reduces expressivity. *In this paper, we present a general recipe to parametrize a large class of selective dense linear RNNs as fixed-points of diagonal linear RNNs.*

Fixed-point RNNs. In order to understand the relationship between dense linear RNNs and the fixed point of diagonal linear RNNs, consider the recurrent form of a dense RNN

$$h_t^* = \mathbf{A}_t h_{t-1}^* + \mathbf{B}_t x_t, \tag{1}$$

where $\mathbf{A}_t \in \mathbb{R}^{d\times d}$ corresponds to the state transition matrix, $\mathbf{B}_t \in \mathbb{R}^{d\times d}$ corresponds to the input transition matrix, $h_t^* \in \mathbb{R}^d$ corresponds to the hidden state, and $x_t \in \mathbb{R}^d$ corresponds to the input for $t < T$ steps.

We parametrize the state transition matrix $\mathbf{A}_t$ as the product of a diagonal matrix $\Lambda_t \in \mathbb{R}^{d\times d}$, and a non-diagonal invertible mixing matrix $\mathbf{Q}_t \in \mathbb{R}^{d\times d}$

$$h_t^* = \mathbf{Q}_t^{-1} \Lambda_t h_{t-1}^* + \mathbf{B}_t x_t. \tag{2}$$

For the diagonal linear RNN $f_\theta(x,h)$, we use Λ_t as the diagonal state transition matrix, and $\mathbf{Q}_t\mathbf{B}_t$ resp. $(\mathbf{I}-\mathbf{Q}_t)$ as transition matrices for its two input sequences x and h:

$$f_\theta(x,h)_t = \Lambda_t f_\theta(x,h)_{t-1} + \mathbf{Q}_t\mathbf{B}_t x_t + (\mathbf{I}-\mathbf{Q}_t) h_t. \tag{3}$$

Assuming that the diagonal linear RNN has a fixed-point in depth $h^* = f_\theta(x,h^*) \in \mathbb{R}^{T\times d}$, we can first reformulate

$$\begin{aligned} h_t^* &= \Lambda_t h_{t-1}^* + \mathbf{Q}_t\mathbf{B}_t x_t + (\mathbf{I}-\mathbf{Q}_t) h_t^* \\ \mathbf{Q}_t h_t^* &= \Lambda_t h_{t-1}^* + \mathbf{Q}_t\mathbf{B}_t x_t, \end{aligned}$$

and than multiply both sides by $\mathbf{Q}_t^{-1}$ to obtain equation (2). This means that the states of the dense linear RNN are implicitly described by the fixed-point of $f_\theta(x,h)$ if it exists. Therefore, we can compute a *dense linear RNN* by finding the fixed-point of the corresponding *diagonal linear RNN*.

Motivated by this insight, we carefully parametrize the diagonal RNN and its *channel mixer* $\mathbf{Q}_t$ such that a fixed-point iteration converges towards the hidden states of the dense RNN. Intuitively, this introduces an iteration in depth of interleaved channel mixing with $\mathbf{Q}_t$ and sequence mixing with parallelizable linear recurrences (Martin & Cundy, 2018).

Summary. In this work, we propose a recipe to design a general class of dense linear RNNs as fixed points of corresponding diagonal linear RNNs. Our contributions are: **(1)** We develop a framework for Fixed-Point RNNs and establish its inherent compatibility with linear attention. **(2)** Fixed-Point RNNs trade parallelism for expressivity with the number of fixed point iterations (Figure 1). **(3)** Fixed-Point RNNs unite previously isolated capabilities of recurrent computation and memory (Figure 2).

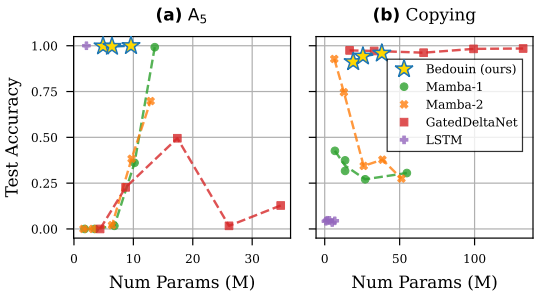

Figure 2: ***(a)*** *State-tracking on A_5 at sequence length 16, and* ***(b)*** *character accuracy of copying at $2\times$ sequence length generalization, trained on lengths $\in [5,50]$. Our single layer Fixed-Point RNN (Bedouin) with mixer reflections $r \in \{1,2,4\}$ is compared to baselines of increasing depth $\in \{1,2,4,6,8\}$.* ***Bedouin is the only model capable of solving both the state-tracking and the copy task.***

2 Fixed-Points as an RNN Layer

In this section, we first parametrize a diagonal linear RNN that is guaranteed to have an attracting fixed-point (Sec. 2.1). Then, we show that for the backward pass, it suffices to compute gradients only at the fixed-point h^* itself (Sec. 2.2). Finally we introduce matrix hidden states to the recurrence (Sec. 2.3)

2.1 COMPUTING THE FIXED POINT

Solving fixed-point equations such as $h = f_\theta(x,h)$, as needed from our discussion of equation 3, is perhaps one of the most well-studied problems in mathematics (Granas et al., 2003). Various methods discuss this in the context of deep learning, where the efficiency of forward and backward passes is a particularly important issue (Bai et al., 2019; 2021; Martens, 2020). However, a straightforward and yet effective method to compute the fixed point of an operator in the forward pass is simply to roll out the fixed point iteration. In the context of solving $h^* = f_\theta(x,h^*)$, this corresponds to introducing an iteration in depth $h^\ell = f_\theta(x,h^{\ell-1})$. Then, denoting as ℓ the current iteration in depth (i.e., over the layer dimension), and t the index of the current token (i.e., over the sequence dimension), one can start from $h_t^0 = 0$ and compute

$$h_t^\ell = \Lambda_t h_{t-1}^\ell + \mathbf{Q}_t \mathbf{B}_t x_t + (\mathbf{I} - \mathbf{Q}_t) h_t^{\ell-1}. \tag{4}$$

The difficulty with such an iteration in time and depth is that the dynamics could explode without proper stabilization. While the recurrence in time can be stabilized with RNN techniques such as an input gate $\mathbf{I} - \Lambda$, the iteration in depth, however, could still diverge if $f_\theta(x,h)$ does not have an attracting fixed-point Granas et al. (2003). Banach's theorem provides sufficient conditions for an operator to have an attracting fixed point (Banach, 1922). In the context of solving $h^* = f_\theta(x,h^*)$, the theorem states that $f_\theta(x,h)$ has an attracting fixed-point w.r.t. h if it has a Lipschitz constant < 1 w.r.t. h. To that end, we present the following theorem to stabilize the fixed-point iteration in depth over diagonal linear RNNs:

Theorem 2.1. *Let $f_\theta(x,h)$ be the diagonal linear RNN*

$$f_\theta(x,h)_t = \Lambda f_\theta(x,h)_{t-1} + (\mathbf{I} - \Lambda)(\mathbf{Q}\mathbf{B}_t x_t + (\mathbf{I} - \mathbf{Q}) h_t),$$

where Λ and $\mathbf{Q}$ are input-independent matrices and Λ is diagonal. If Λ and $\mathbf{I} - \mathbf{Q}$ are contractive (i.e. $\|\cdot\|_2 < 1$), then $f_\theta(x,h)$ has a Lipschitz constant < 1 in h. Proof in App. F.1.

Theorem 2.1 provides a way to parametrize linear RNNs with input-independent transition matrices in order to ensure that $f_\theta(x,h)$ has Lipschitz constant < 1: in addition to stabilizing the recurrence in time with a contractive state transition matrix Λ and an input normalization $\mathbf{I} - \Lambda$, the recurrence in depth can be stabilized with a contractive input transition matrix for the input h_t. Together, these two components guarantee that throughout the entire fixed-point iteration, all sequences h^ℓ up to h^* do not explode.

2.2 COMPUTING THE GRADIENT

One advantage of an explicit fixed-point parameterization, such as the one derived in Theorem 2.1, lies in the gradient computation. As described by Bai et al. (2019), back-propagation through the fixed-point iteration can be avoided once a fixed-point is found. Following the results of App. B, the gradient w.r.t. the input x and parameters θ can be computed at the fixed-point with the cost of solving $(\mathbf{I} - \mathbf{J}_{h^*})^{-1}$, where $\mathbf{J}_{h^*}$ is the Jacobian of $f_\theta(x,h^*)$ w.r.t. h^*. Bai et al. (2021) and Schöne et al. (2025) approximate this inverse using the first terms of the Neumann series, which leads to a truncated backpropagation formulation with sequential overhead. We avoid this inversion altogether with the following workaround:

Theorem 2.2. *Let $f_\theta(x,h)$ be a diagonal linear RNN, with fixed-point h^* and Lipschitz constant < 1 in h. Let further $\mathcal{L}(\cdot,\cdot)$ be a loss and y a target. If the Jacobians $\mathbf{J}_x(h) = \frac{\partial f_\theta}{\partial x}(x,h)$ and $\mathbf{J}_h = \frac{\partial f_\theta}{\partial h}(x,h)$ are equal, then the gradient $\nabla_\theta \mathcal{L}(f_\theta(x,h),y)$ computed at the fixed point $h = h^*$ will be a descent direction of $\mathcal{L}(f_\theta(x,h^*),y)$. Proof in App. F.2.*

Theorem 2.2 shows that parametrizing $f_\theta(x,h)$ such that $\mathbf{J}_x(h) = \mathbf{J}_h$ guarantees optimization progress even if the gradient is computed only at the fixed-point. In practice, we compute the gradients at the fixed-point instead of unrolling the backward pass, and observe that adhering to this condition speeds-up the convergence of the model during training.

2.3 INTRODUCING MATRIX HIDDEN STATES

Memory capacity is an important consideration in RNNs. In preliminary experiments, we notice a clear gap between the performance of a Fixed-Point RNN and Mamba (Dao & Gu, 2024) in terms of copying ability. We attribute this difference in performance to Mamba's state-expansion which endows it with matrix hidden states similar to linear attention, DeltaNet, or mLSTM (Katharopoulos et al., 2020; Schlag et al., 2021b; Beck et al., 2024). In simple terms, these models use an outer product of an input-dependent

vector $\mathbf{B}_t \in \mathbb{R}^{d_{\text{state}}}$ and the input $x_t \in \mathbb{R}^{d_{\text{inner}}}$ as a matrix-valued input to a recurrence $h_t = \lambda_t \odot h_{t-1} + \mathbf{B}_t x_t^\top$ with matrix-valued hidden state and transition gate $h_t, \lambda_t \in \mathbb{R}^{d_{\text{state}} \times d_{\text{inner}}}$. The hidden state is then contracted with another input-dependent vector $\mathbf{C}_t \in \mathbb{R}^{d_{\text{state}}}$ to get the output $y_t^\top = \mathbf{C}_t^\top h_t \in \mathbb{R}^{d_{\text{inner}}}$.

The matrix-valued recurrence introduces some challenges to our fixed-point framework. Specifically, in order to mix all the channels over the entirety of the state elements, the mixer $\mathbf{Q}_t$ has to be a fourth-order tensor $\in \mathbb{R}^{d_{\text{state}} \times d_{\text{inner}} \times d_{\text{state}} \times d_{\text{inner}}}$ in the recurrence $h_t^\ell = \lambda_t \odot h_{t-1}^\ell + \mathbf{Q}_t \otimes \mathbf{B}_t x_t^\top + (\mathbf{I}_{d^4} - \mathbf{Q}_t) \otimes h_t^{\ell-1}$, where $\otimes$ denotes the fourth-order tensor product and $\mathbf{I}_{d^4}$ a fourth-order identity tensor of the same shape as $\mathbf{Q}_t$. Certainly computing this recurrence is very challenging both in terms of computation and memory. As we confirm in section 3, one solution is to pass the contracted output y_t between fixed-point iterations

$$h_t^\ell = \lambda_t \odot h_{t-1}^\ell + \mathbf{B}_t (\mathbf{Q}_t x_t)^\top + \mathbf{B}_t \big((\mathbf{I} - \mathbf{Q}_t) y_t^{\ell-1}\big)^\top. \tag{5}$$

This implicitly factorizes the mixer $\mathbf{Q}_t$ into separately mixing along dimension d_{inner} which is used for better expressivity, and dimension d_{state} which is used for better memory.

3 Evaluation

We integrate the findings from the previous sections into a dense variant of Mamba (Gu & Dao, 2023) which we call Bedouin. Specifically, we combine the Griffin's normalization term Δ_t (De et al., 2024) and the fixed-point mechanism for Mamba's matrix state RNN (5) into the following fixed-point iteration:

$$h_t^\ell = \lambda_t \odot h_{t-1}^\ell + \bar{\mathbf{B}}_t^\ell \big(\Delta_t \mathbf{Q}_t^\ell x_t\big)^\top + \bar{\mathbf{B}}_t^\ell \big(\Delta_t (\mathbf{I} - \mathbf{Q}_t^\ell) y_t^{\ell-1}\big)^\top, \qquad y_t^{\ell\top} = (\bar{\mathbf{C}}_t^\ell)^\top h_t^\ell. \tag{6}$$

For the exact parametrization of Bedouin, we refer to App. D. In the following, we provide experimental results for Bedouin. A detailed summary of the experiment setup is available in App. E.

We compare Bedouin on the three tasks introduced above to Mamba (Gu & Dao, 2023), Mamba-2 (Dao & Gu, 2024), Gated DeltaNet (Yang et al., 2025) and the original LSTM (Hochreiter & Schmidhuber, 1997). In order to keep the number of layers at the same order of magnitude, we use two layers (2L) for the diagonal linear RNN baselines and one layer (1L) for Bedouin and LSTM. This keeps the number of parameters for the Bedouin comparable to the baselines (see Figure 2), and smaller than Gated DeltaNet for all experiments. For Bedouin, we report results for $r \in \{1,2,4\}$ Householder reflections of $\mathbf{Q}_t$ and a maximum number of 100 fixed-point iterations. However, we also investigate the effect of limiting the number of fixed-point iterations allowed during training to $\ell_{\max} \in \{2,4,8,16\}$ in Fig. 1. In order to reduce the average number of iterations, we also evaluate a randomization scheme where we sample from the Gamma distribution $\ell_{\max} \sim \Gamma(4,1)$ with mean 4 and mode 3.

State Tracking. In Figure 3, we compare Bedouin for $r \in \{1,2,4\}$ reflections and a maximum number of fixed-point iterations $\ell_{\max} = 16$ to the baselines on the A_5 and S_5 tasks with sequence length 16. As expected, the LSTM solves A_5 and S_5, while Mamba and Mamba-2 are not able to learn it at the training sequence length.

Copying. In Figure 4, we evaluate length generalization on the copying task. Both the Mamba and Mamba-2 models struggle with x2 generalization, which proves the effectiveness of our proposed modifications for better memory. The best-performing baseline is Gated DeltaNet, which is specifically designed to do well on the associative recall task (Yang et al., 2025) and is in fact a linear Transformer variant with about $\times 2$ parameters as Bedouin with mixer rank 1. Note that the number of iterations required by Bedouin to reach the fixed point (gray vertical line) is well below the maximum sequence length of the data.

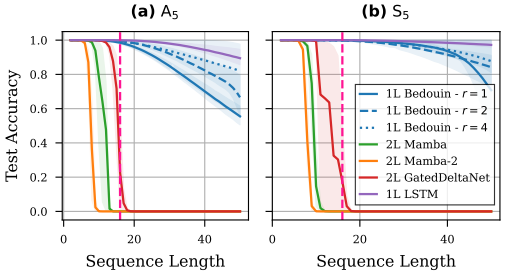

Figure 3: *Sequence length generalization on* ***(a)*** A_5 *and* ***(b)*** S_5. *We compare a 1-layer Bedouin with a mixer* $\mathbf{Q}_t$ *of* r *reflections to baselines with* L *layers. The pink line denotes the train sequence length.*

The Chomsky Hierarchy. In Table 1, we evaluate modular arithmetic with brackets. We observe that a 2-layer Bedouin outperforms the baselines reported by Grazzi et al. (2024) with a comparable number of parameters. In Figure 6, we plot the validation accuracy as a function of the number of fixed-point iterations. We observe that the accuracy plateaus at 20 iterations, which is significantly less than the shortest

and longest sequence in the validation set. Therefore, the number of iterations required by the Bedouin to reach its fixed point clearly does not scale with the sequence length in this task.

3.1 Number of Fixed-Point Iterations

A fixed-point iteration inevitably introduces sequential overhead to the computation of a model. While this might be acceptable for sequential generation at test time, reduced parallelism can be inhibiting at training time. In Figure 1, we therefore evaluate Bedouin on A_5 with a limited number of fixed-point iterations at training time ℓ_{max}. We observe that the performance decreases once ℓ_{max} is lower than the training sequence length of 16. Randomly sampling ℓ_{max} during training, however, allows to reduce the average sequential steps to 4 with good sequence length generalization at test time. We want to highlight that we do not observe this increase in the number of fixed-point iterations in the other tasks as discussed earlier.

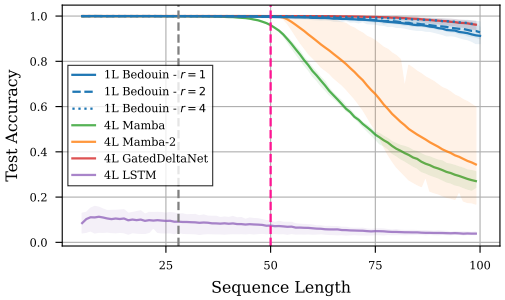

Figure 4: *Sequence length generalization on the copy task. We compare a 1-layer Bedouin with a mixer $\mathbf{Q}_t$ of r reflections to baselines with L layers. The gray line denotes the median number of fixed-point iterations at test time. The pink line denotes the maximum train sequence length.*

4 Discussion

A fixed-point mechanism, such as the one introduced in this paper, endows a parallelizable, diagonal linear RNN with the ability to dynamically increase the sequential computation per token and describe a dense linear RNN in the limit. Our results show that such a paradigm can enable both strong state-tracking and memory capabilities with a constant number of parameters in a combined sequence and channel mixing layer (Figure 2). In fact, the fixed-point iteration gradually transforms a diagonal (i.e., channel-wise) RNN into a dense (i.e., channel-mixing) RNN, thereby allowing to trade parallel computation for expressivity (Figure 1).

Model	Accuracy
2L Transformer	0.025
2L mLSTM	0.034
2L sLSTM	**0.173**
2L Mamba	0.136
2L DeltaNet	**0.200**
2L Bedouin ($r=4$)	**0.280**

Table 1: *Accuracies of various models on modular arithmetic with brackets, where random guessing is normalized to* 0.0*. We adopt the reported numbers by Grazzi et al. (2024) who evaluate baselines with extended eigenvalue range.*

For Fixed-Point RNNs to become competitive in practice, it is paramount to understand the trade-offs between parallel and sequential computation. In the worst case, if the sequential overhead is linear in the sequence length T, the model essentially behaves like a traditional, non-linear RNN with quadratic runtime $O(T^2)$. This, however, is not necessarily a disadvantage if the model is capable of adapting its sequential steps to the difficulty of the task, with negligible cost for the less demanding tasks. In this paper, we focus on introducing and parameterizing the framework for Fixed-Point RNNs. Therefore, we leave the analysis and improvement of fixed-point convergence speeds beyond our preliminary results (Figure 1, 4, 6) to future work.

Fixed-Point RNNs present an interesting opportunity to be fused into a single GPU kernel with reduced memory I/O, This is an inherent advantage from performing repeated computations on the same operands. Several open problems need to be solved to achieve that: (1) different implementations should converge to the same fixed-point, wether they compute the recurrence sequentially, in parallel, or in chunks, (2) the input-dependence and memory footprint of the fixed-point iteration should be reduced to satisfy current hardware limitations, and (3) alternative structures for sequence or channel mixers $\mathbf{Q}_t$ could reduce the computational burden. Future progress on these problems could enable significant speed-ups in practical implementations of Fixed-Point RNNs.

Conclusion In this paper, we presented a framework to cast a general class of dense linear RNNs as fixed-points of corresponding diagonal linear RNNs. We then show a seamless adaptation of the proposed Fixed-Point RNN framework into the linear attention family of architectures. The proposed framework provides a mechanism to trade expressivity with computation complexity while uniting the expressivity of recurrent models with the improved memory of linear attention models. Following encouraging results on toy tasks specifically designed to assess these capabilities, we hope this paper enables more expressive sequence mixing models without sacrificing memory capabilities.

ACKNOWLEDGMENTS

We would like to thank Riccardo Grazzi and Julien Siems for the helpful discussions and comments. Antonio Orvieto, Felix Sarnthein and Sajad Movahedi acknowledge the financial support of the Hector Foundation. Felix Sarnthein would also like to acknowledge the financial support from the Max Planck ETH Center for Learning Systems (CLS).

REFERENCES

Simran Arora, Sabri Eyuboglu, Michael Zhang, Aman Timalsina, Silas Alberti, James Zou, Atri Rudra, and Christopher Ré. Simple linear attention language models balance the recall-throughput tradeoff. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Shaojie Bai, J. Zico Kolter, and Vladlen Koltun. Deep equilibrium models. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'AlcheBuc, Emily B. Fox, and Roman Garnett (eds.), *Advances in Neural Information Processing Systems*, pp. 688–699, 2019.

Shaojie Bai, Vladlen Koltun, and J. Zico Kolter. Stabilizing equilibrium models by jacobian regularization. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 554–565. PMLR, 2021.

Stefan Banach. Sur les opérations dans les ensembles abstraits et leur application aux équations intégrales. *Fundamenta mathematicae*, 3(1):133–181, 1922.

Maximilian Beck, Korbinian Pöppel, Markus Spanring, Andreas Auer, Oleksandra Prudnikova, Michael Kopp, Günter Klambauer, Johannes Brandstetter, and Sepp Hochreiter. xlstm: Extended long short-term memory. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Yifan Chen, Qi Zeng, Heng Ji, and Yun Yang. Skyformer: Remodel self-attention with Gaussian kernel and Nystrom method. *Advances in Neural Information Processing Systems*, 2021.

Krzysztof Marcin Choromanski, Valerii Likhosherstov, David Dohan, Xingyou Song, Andreea Gane, Tamas Sarlos, Peter Hawkins, Jared Quincy Davis, Afroz Mohiuddin, Lukasz Kaiser, et al. Rethinking attention with performers. In *International Conference on Learning Representations*, 2020.

Nicola Muca Cirone, Jad Hamdan, and Cristopher Salvi. Graph expansions of deep neural networks and their universal scaling limits, 2024a.

Nicola Muca Cirone, Antonio Orvieto, Benjamin Walker, Cristopher Salvi, and Terry J. Lyons. Theoretical foundations of deep selective state-space models. *CoRR*, abs/2402.19047, 2024b. doi: 10.48550/ARXIV.2402.19047.

Tri Dao and Albert Gu. Transformers are ssms: Generalized models and efficient algorithms through structured state space duality. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Soham De, Samuel L. Smith, Anushan Fernando, Aleksandar Botev, George-Cristian Muraru, Albert Gu, Ruba Haroun, Leonard Berrada, Yutian Chen, Srivatsan Srinivasan, Guillaume Desjardins, Arnaud Doucet, David Budden, Yee Whye Teh, Razvan Pascanu, Nando de Freitas, and Caglar Gulcehre. Griffin: Mixing gated linear recurrences with local attention for efficient language models. *CoRR*, abs/2402.19427, 2024. doi: 10.48550/ARXIV.2402.19427.

Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Lukasz Kaiser. Universal transformers. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net, 2019.

Jeffrey L. Elman. Finding structure in time. *Cogn. Sci.*, 14(2):179–211, 1990. doi: 10.1207/S15516709COG1402_1.

Laurent El Ghaoui, Fangda Gu, Bertrand Travacca, Armin Askari, and Alicia Y. Tsai. Implicit deep learning. *SIAM J. Math. Data Sci.*, 3(3):930–958, 2021. doi: 10.1137/20M1358517.

Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D. Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *International Conference on Machine Learning, ICML 2023, 23-29 July 2023, Honolulu, Hawaii, USA*, volume 202 of *Proceedings of Machine Learning Research*, pp. 11398–11442. PMLR, 2023.

Karan Goel, Albert Gu, Chris Donahue, and Christopher Ré. It's raw! audio generation with state-space models. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvári, Gang Niu, and Sivan Sabato (eds.), *International Conference on Machine Learning, ICML 2022, 17-23 July 2022, Baltimore, Maryland, USA*, volume 162 of *Proceedings of Machine Learning Research*, pp. 7616–7633. PMLR, 2022.

Xavier Gonzalez, Andrew Warrington, Jimmy T. H. Smith, and Scott W. Linderman. Towards scalable and stable parallelization of nonlinear rnns. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Andrzej Granas, James Dugundji, et al. *Fixed point theory*, volume 14. Springer, 2003.

Alex Graves. Adaptive computation time for recurrent neural networks. *CoRR*, abs/1603.08983, 2016.

Riccardo Grazzi, Julien Siems, Jörg K. H. Franke, Arber Zela, Frank Hutter, and Massimiliano Pontil. Unlocking state-tracking in linear rnns through negative eigenvalues. *CoRR*, abs/2411.12537, 2024. doi: 10.48550/ARXIV.2411.12537.

Albert Gu and Tri Dao. Mamba: Linear-time sequence modeling with selective state spaces. *CoRR*, abs/2312.00752, 2023. doi: 10.48550/ARXIV.2312.00752.

Albert Gu, Karan Goel, and Christopher Ré. Efficiently modeling long sequences with structured state spaces. In *The Tenth International Conference on Learning Representations, ICLR 2022, Virtual Event, April 25-29, 2022*. OpenReview.net, 2022.

Joshua Hanson and Maxim Raginsky. Universal simulation of stable dynamical systems by recurrent neural nets. In *Learning for Dynamics and Control*. PMLR, 2020.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8):1735–1780, 1997. doi: 10.1162/NECO.1997.9.8.1735.

Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, et al. Gradient flow in recurrent nets: the difficulty of learning long-term dependencies. *A Field Guide to Dynamical Recurrent Neural Networks*, 2001.

J J Hopfield. Neural networks and physical systems with emergent collective computational abilities. *Proceedings of the National Academy of Sciences*, 79(8):2554–2558, 1982. doi: 10.1073/pnas.79.8.2554.

Herbert Jaeger. The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *German National Research Center for Information Technology GMD Technical Report*, 2001.

Samy Jelassi, David Brandfonbrener, Sham M. Kakade, and Eran Malach. Repeat after me: Transformers are better than state space models at copying. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International Conference on Machine Learning*. PMLR, 2020.

Samuel A Korsky. *On the computational power of RNNs*. PhD thesis, Massachusetts Institute of Technology, 2019.

Yi Heng Lim, Qi Zhu, Joshua Selfridge, and Muhammad Firmansyah Kasim. Parallelizing non-linear sequential models over the sequence length. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024.

Yue Liu, Yunjie Tian, Yuzhong Zhao, Hongtian Yu, Lingxi Xie, Yaowei Wang, Qixiang Ye, Jianbin Jiao, and Yunfan Liu. Vmamba: Visual state space model. In Amir Globersons, Lester Mackey, Danielle Belgrave, Angela Fan, Ulrich Paquet, Jakub M. Tomczak, and Cheng Zhang (eds.), *Advances in Neural Information Processing Systems 38: Annual Conference on Neural Information Processing Systems 2024, NeurIPS 2024, Vancouver, BC, Canada, December 10 - 15, 2024*, 2024.

Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*, 2017.

James Martens. New insights and perspectives on the natural gradient method. *J. Mach. Learn. Res.*, 21: 146:1–146:76, 2020.

Eric Martin and Chris Cundy. Parallelizing linear recurrent neural nets over sequence length. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*. OpenReview.net, 2018.

William Merrill and Ashish Sabharwal. The parallelism tradeoff: Limitations of log-precision transformers. *Trans. Assoc. Comput. Linguistics*, 11:531–545, 2023. doi: 10.1162/TACL_A_00562.

William Merrill, Jackson Petty, and Ashish Sabharwal. The illusion of state in state-space models. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024.

Takeru Miyato, Sindy Löwe, Andreas Geiger, and Max Welling. Artificial kuramoto oscillatory neurons. In *The Thirteenth International Conference on Learning Representations*, 2025.

Eric Nguyen, Michael Poli, Matthew G Durrant, Brian Kang, Dhruva Katrekar, David B Li, Liam J Bartie, Armin W Thomas, Samuel H King, Garyk Brixi, et al. Sequence modeling and design from molecular to genome scale with Evo. *Science*, 2024.

Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International Conference on Machine Learning*, 2013.

Bo Peng, Daniel Goldstein, Quentin Anthony, Alon Albalak, Eric Alcaide, Stella Biderman, Eugene Cheah, Xingjian Du, Teddy Ferdinan, Haowen Hou, et al. Eagle and Finch: RWKV with matrix-valued states and dynamic recurrence. *arXiv preprint arXiv:2404.05892*, 2024.

Zhen Qin, Songlin Yang, Weixuan Sun, Xuyang Shen, Dong Li, Weigao Sun, and Yiran Zhong. HGRN2: Gated linear RNNs with state expansion. *arXiv preprint arXiv:2404.07904*, 2024.

David E Rumelhart, Paul Smolensky, James L McClelland, and G Hinton. Sequential thought processes in pdp models. *Parallel Distributed Processing: Explorations in the Microstructures of Cognition*, 1986.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In Marina Meila and Tong Zhang (eds.), *Proceedings of the 38th International Conference on Machine Learning, ICML 2021, 18-24 July 2021, Virtual Event*, volume 139 of *Proceedings of Machine Learning Research*, pp. 9355–9366. PMLR, 2021a.

Imanol Schlag, Kazuki Irie, and Jürgen Schmidhuber. Linear transformers are secretly fast weight programmers. In *International Conference on Machine Learning*, 2021b.

Avi Schwarzschild, Eitan Borgnia, Arjun Gupta, Furong Huang, Uzi Vishkin, Micah Goldblum, and Tom Goldstein. Can you learn an algorithm? generalizing from easy to hard problems with recurrent networks. In Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems 2021, NeurIPS 2021, December 6-14, 2021, virtual*, pp. 6695–6706, 2021.

Mark Schöne, Babak Rahmani, Heiner Kremer, Fabian Falck, Hitesh Ballani, and Jannes Gladrow. Implicit language models are rnns: Balancing parallelization and expressivity, 2025.

Hava T Siegelmann and Eduardo D Sontag. On the computational power of neural nets. In *Proceedings of the fifth Annual Workshop on Computational Learning Theory*, 1992.

Julien Siems, Timur Carstensen, Arber Zela, Frank Hutter, Massimiliano Pontil, and Riccardo Grazzi. Deltaproduct: Increasing the expressivity of deltanet through products of householders, 2025.

Jimmy T.H. Smith, Andrew Warrington, and Scott Linderman. Simplified state space layers for sequence modeling. In *International Conference on Learning Representations*, 2023.

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

Yi Tay, Mostafa Dehghani, Samira Abnar, Yikang Shen, Dara Bahri, Philip Pham, Jinfeng Rao, Liu Yang, Sebastian Ruder, and Donald Metzler. Long range arena: A benchmark for efficient transformers. In *International Conference on Learning Representations*, 2020.

Asher Trockman, Hrayr Harutyunyan, J. Zico Kolter, Sanjiv Kumar, and Srinadh Bhojanapalli. Mimetic initialization helps state space models learn to recall. *CoRR*, abs/2410.11135, 2024. doi: 10.48550/ARXIV.2410.11135.

Jesús Urías. Householder factorizations of unitary matrices. *Journal of mathematical physics*, 51(7), 2010.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in Neural Information Processing Systems*, 2017.

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

Sinong Wang, Belinda Z Li, Madian Khabsa, Han Fang, and Hao Ma. Linformer: Self-attention with linear complexity. *arXiv preprint arXiv:2006.04768*, 2020.

Songlin Yang, Bailin Wang, Yikang Shen, Rameswar Panda, and Yoon Kim. Gated linear attention transformers with hardware-efficient training. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net, 2024a.

Songlin Yang, Bailin Wang, Yu Zhang, Yikang Shen, and Yoon Kim. Parallelizing linear transformers with the delta rule over sequence length. *CoRR*, abs/2406.06484, 2024b. doi: 10.48550/ARXIV.2406.06484.

Songlin Yang, Jan Kautz, and Ali Hatamizadeh. Gated delta networks: Improving mamba2 with delta rule. In *The Thirteenth International Conference on Learning Representations*, 2025.

Appendices

A Related Work **10**

B Gradient at the Fixed-Point h^* **11**

C Memory in Fixed-Point RNNs **12**

C.1 Dependence on h_{t-1} in practice . . . 12

C.2 Dependence on h_{t-1} in theory . . . 12

C.3 Expressing linear attention with $\lambda_t \approx \mathbf{1}\ \forall t$ . . . 13

D The Bedouin **13**

D.1 Mamba: Selective SSMs . . . 13

D.2 The Bedouin iteration . . . 14

D.3 Parameterization . . . 14

D.3.1 The transition matrices . . . 14

D.3.2 The channel mixer . . . 15

E Experimental setup **15**

E.1 Tasks . . . 15

E.2 Training & Evaluation Details . . . 17

F Proofs **18**

F.1 Proof for Theorem 2.1 . . . 18

F.2 Proof for Theorem 2.2 . . . 19

F.3 Proof for Theorem C.1 . . . 19

G Low-Rank Expressiveness **19**

A Related Work

Since their introduction (Rumelhart et al., 1986; Elman, 1990), RNNs have significantly contributed to the evolution of machine learning methods for sequential data, marked by key innovations such as the LSTM (Hochreiter & Schmidhuber, 1997) and Echo-State Networks (Jaeger, 2001). However, two significant challenges lead to the widespread adoption of the Transformer architecture (Vaswani et al., 2017): first, GPU hardware is optimized for large-scale matrix multiplications. Second, recurrent models are notoriously difficult to train due to vanishing and exploding gradients (Hochreiter et al., 2001; Pascanu et al., 2013).

Beyond softmax attention. The quadratic runtime complexity of Transformers motivated research on the linearization of its attention mechanism (Wang et al., 2020; Chen et al., 2021; Choromanski et al., 2020) – a technique that inevitably brings the sequence mixing mechanism closer to RNN-like processing (Katharopoulos et al., 2020; Schlag et al., 2021b). Recently, improvements on the long-range-arena benchmark (Tay et al., 2020) with state-space models (Gu et al., 2022; Smith et al., 2023) sparked

a renewed interest in recurrent models (Gu & Dao, 2023; Sun et al., 2023; De et al., 2024; Qin et al., 2024; Peng et al., 2024; Yang et al., 2024a). New efficient token mixing strategies such as Mamba (Gu & Dao, 2023) showcase impressive results in language modeling (Waleffe et al., 2024) while offering linear runtime complexity. These models are fundamentally diagonal linear RNNs, which enables parallel algorithms such as parallel scans (Martin & Cundy, 2018) and fast linear attention based implementations (Yang et al., 2024b; Dao & Gu, 2024).

Expressivity of Diagonal vs. Dense RNNs. It was recently pointed out by Cirone et al. (2024b) that the diagonality in the hidden-to-hidden state transition inevitably causes expressivity issues, showcasing a stark distinction with classic dense nonlinear RNNs, known to be Turing-complete (Siegelmann & Sontag, 1992; Korsky, 2019) and fully expressive in a dynamical systems sense (Hanson & Raginsky, 2020). Merrill et al. (2024) pointed at a similar issue with diagonality using tools from circuit complexity: in contrast to e.g. LSTMs, diagonal linear RNNs can not express state-tracking algorithms. This issue sparked interest in designing fast non-diagonal recurrent mechanisms and, more generally, in providing architectures capable of solving state-tracking problems. The first example of such an architecture is DeltaNet (Yang et al., 2024b) employing a parallelizable Housholder reflection as a state transition matrix. Endowing this matrix with negative eigenvalues improves tracking in SSMs (Grazzi et al., 2024). In concurrent work, Siems et al. (2025) show that adding more reflections improves state-tracking.

Toy tasks. Several works propose toy tasks to identify specific shortcomings of modern architectures. Specifically, Beck et al. (2024) use the Chomsky hierarchy to organize formal language tasks, of which a modular arithmetic task remains unsolved. With similar motivations, Merrill & Sabharwal (2023) introduce a set of word-problems for assessing state-tracking capabilities, among which the A_5 and S_5 tasks remain unsolved by Transformers and SSMs. Motivated by Transformers outperforming RNNs in memory capabilities, Jelassi et al. (2024) introduce a copying task as a fundamental benchmark for memory. We focus on these tasks to evaluate our Fixed-Point RNN framework.

Recurrence in Depth. Machine learning models that reduce an intrinsic energy through iterations have been an object of interest for decades (Hopfield, 1982; Miyato et al., 2025). For example, recurrence in depth can increase the expressivity of Transformers (Dehghani et al., 2019; Schwarzschild et al., 2021; Giannou et al., 2023) and is sometimes also understood as adaptive compute time (Graves, 2016). Under certain assumptions, iterated blocks can converge to an equilibrium point where they implicitly describe an expressive function (Bai et al., 2019; Ghaoui et al., 2021). Recently, this technique has been used to approximate non-linear RNNs with a fixed-point iteration of parallelizable linear RNNs (Lim et al., 2024; Gonzalez et al., 2024). In concurrent work to ours, Schöne et al. (2025) apply an iteration in depth to Mamba-2 and Llama blocks to increase expressivity and show promising results of their *implicit language models*. In contrast, we derive an explicit fixed-point iteration towards a dense linear RNN with a theoretically motivated parameterization, and focus on theoretical toy tasks.

B Gradient at the Fixed-Point h^*

One advantage of an explicit fixed-point parameterization, such as the one derived in Theorem 2.1, lies in the gradient computation. As described by Bai et al. (2019), back-propagation through the fixed-point iteration can be avoided once a fixed-point is found. To see this, consider the Jacobian across ℓ iterations $\mathbf{J}_x^\ell = \frac{\partial f_\theta}{\partial x}(x,h^{\ell-1})$. Since $h^{\ell-1}$ depends on x as well, we can recursively express $\mathbf{J}_x^\ell$ in terms of $\mathbf{J}_x^{\ell-1}$ and the Jacobians of a single iteration $\mathbf{J}_x(h) = \frac{\partial f_\theta}{\partial x}(x,h)$ and $\mathbf{J}_h = \frac{\partial f_\theta}{\partial h}(x,h)$ by applying the chain rule

$$\mathbf{J}_x^\ell = \mathbf{J}_x(h^{\ell-1}) + \mathbf{J}_{h^{\ell-1}} \cdot \mathbf{J}_x^{\ell-1}.$$

Instead of unrolling, we can implicitly differentiate $h^* = f_\theta(x,h^*)$ w.r.t. x, which yields $\mathbf{J}_x^* = \mathbf{J}_x(h^*) + \mathbf{J}_{h^*} \cdot \mathbf{J}_x^*$. Given the conditions on the Lipschitz constant of $f_\theta(x,h)$ in h, we can assume $\mathbf{J}_{h^\ell}$ to be contractive and therefore $(\mathbf{I} - \mathbf{J}_{h^\ell})$ to be positive definite and invertible. This allows to reformulate as

$$\mathbf{J}_x^* = (\mathbf{I} - \mathbf{J}_{h^*})^{-1} \cdot \mathbf{J}_x(h^*). \tag{7}$$

The case for $\mathbf{J}_\theta^*$ works analogously. This means that the gradient w.r.t. the input x and parameters θ can be computed at the fixed-point with the cost of solving $(\mathbf{I} - \mathbf{J}_{h^*})^{-1}$. Bai et al. (2021) and Schöne et al. (2025) approximate this inverse using the first terms of the Neumann series, which leads to a truncated backpropagation formulation or *phantom gradients*, incurring sequential overhead. We avoid this inversion altogether with the workaround in 2.2.

C MEMORY IN FIXED-POINT RNNS

In this section, we identify conditions which improve the memory capabilities of Fixed-Point RNNs. To that end, we evaluate Fixed-Point RNNs on the copy task introduced by Jelassi et al. (2024) as a fundamental benchmark for memory.

C.1 DEPENDENCE ON h_{t-1} IN PRACTICE

Unfortunately, even the Fixed-Point RNN with input-dependent parameters and matrix state akin to Mamba-1 is outperformed by Mamba-2 or DeltaNet (Dao & Gu, 2024; Yang et al., 2024b) on the copy task. Inspired by the short convolution in Mamba, we investigate the effect of augmenting the input-dependence of parameters λ_t^ℓ, $\mathbf{B}_t^\ell$, $\mathbf{C}_t^\ell$, and $\mathbf{Q}_t^\ell$ at iteration ℓ with a shifted hidden-state dependence. In practice, this means that these are linear functions of x_t as well as the shifted previous iterate (layer) $h_{t-1}^{\ell-1}$. We refer the reader to Sec. D.3 for the exact formulation of the dependency.

In Table 2, we ablate the hidden-state dependence for various combinations of λ_t, $\mathbf{B}_t$, $\mathbf{C}_t$, and $\mathbf{Q}_t$. Observe that the dependence of $\mathbf{B}_t$ and $\mathbf{C}_t$ is crucial to enable the model to copy. Furthermore, if additionally λ_t and $\mathbf{Q}_t$ depend on $h_{t-1}^{\ell-1}$, the copy task is essentially solvable at $\times 2$ length generalization.

The dense matrix $\mathbf{A}_t$ of the corresponding dense RNN in 1 at the fixed-point h^* now also depends on the hidden state h_{t-1}^*, akin to traditional RNNs. This means that the fixed-point iteration is no longer-convex and a solution h^* may not be unique. Therefore different fixed-point methods are not guaranteed to converge to the same result. In particular, fixed-points found in parallel (or chunked) form at training time or sequentially at inference time could not be equivalent anymore. While we observe that this is indeed the case at initialization, during training the two methods of finding fixed-points become gradually closer until they produce the same value when the model is fully trained. We provide empirical evidence for this claim in Appendix, Figure 7.

Dependence on $h_{t-1}^{\ell-1}$				**Test Accuracy**
λ_t	$\mathbf{Q}_t$	$\mathbf{B}_t$	$\mathbf{C}_t$	
				0.11 ± 0.00
✓				0.53 ± 0.02
	✓			0.45 ± 0.05
✓	✓			0.55 ± 0.05
		✓	✓	0.81 ± 0.01
✓		✓	✓	0.88 ± 0.01
	✓	✓	✓	0.86 ± 0.02
✓	✓	✓	✓	0.94 ± 0.03

Table 2: Copying at $2\times$ length generalization for Bedouin (refer to Sec. D for model details). Each column determines which components of the recurrence from (6) depends on the previous hidden state h_{t-1}. Performance is unlocked by including a dependency on $\mathbf{B}_t$ and $\mathbf{C}_t$.

C.2 DEPENDENCE ON h_{t-1} IN THEORY

We hypothesize that the dependence of the matrices λ_t, $\mathbf{B}_t$, $\mathbf{C}_t$, and $\mathbf{Q}_t$ may provide a mechanism for the model to retain and manipulate positional information over the sequence. Jelassi et al. (2024) and Trockman et al. (2024) show that position embeddings could play a crucial role in copy tasks by acting similar to hashing keys in a hashing table. We extend their mechanistic approach to understand why two-layers of linear attention could need $h_{t-1}^{\ell-1}$ to generate appropriate position embeddings for the hashing mechanism.

Specifically consider $y_t = \mathbf{C}_t^\top h_t$ with $h_t = h_{t-1} + \mathbf{B}_t x_t^\top$, assuming that a linear RNN with matrix-state can express linear attention by setting $\lambda_t \approx \mathbf{1}\ \forall t$. Upon receiving an input sequence $\{x_1, x_2, \dots, x_\delta\}$ of length δ followed by a delimiter element $x_\mathbf{s}$, the model is expected to copy the input sequence autoregressively, i.e. to start producing $\{x_1, x_2, \dots, x_\delta\}$ at output positions $\delta+1$ to 2δ. Following (Arora et al., 2024), the second layer could use position embeddings as hashing keys to detect and copy each token. More concretely, if the first layer receives a sequence $\{x_1, x_2, \dots, x_\delta, x_\mathbf{s}, x_1, x_2, \dots, x_{\delta-1}\}$ of size 2δ and augments it with shifted position embeddings $\{p_i\}_{i=1}^\delta$ to produce the hidden sequence $\{x_1 + p_1, x_2 + p_2, \dots, x_\delta + p_\delta, x_\mathbf{s} + p_1, x_1 + p_2, \dots, x_{\delta-1} + p_\delta\}$, then a second layer can act as a linear transformer and produce the sequence $\{x_1, x_2, \dots, x_\delta\}$ at output positions $\delta+1$ to 2δ. In the following, we focus on the conditions for the first layer to produce the shifted position embeddings.

We start by assuming that the first layer has a skip-connection $y_t^\top = \mathbf{C}_t^\top h_t + x_t^\top$. In this case, the inputs can be augmented if the recurrence is able to produce shifted encodings $p_{t-\delta} = p_t$ for $\delta < t$ using $p_t^\top = \mathbf{C}_t^\top h_t$.

This condition can be unrolled as

$$\mathbf{C}_{t-\delta}^\top h_{t-\delta} \stackrel{!}{=} \mathbf{C}_t^\top h_{t-\delta} + \mathbf{C}_t^\top \sum_{\tau=t-\delta+1}^{t} \mathbf{B}_\tau x_\tau^\top \qquad \forall \delta < t.$$

and is satisfied if the equations $\mathbf{C}_{t-\delta}^\top h_{t-\delta} \stackrel{!}{=} \mathbf{C}_t^\top h_{t-\delta}$ and

$$\mathbf{C}_t^\top \sum_{\tau=t-\delta+1}^{t-1} \mathbf{B}_\tau x_\tau^\top \stackrel{!}{=} -\mathbf{C}_t^\top \mathbf{B}_t x_t^\top$$

hold. Such conditions could only be true if $\mathbf{B}_t$ and $\mathbf{C}_t$ are a function of the previous hidden state h_{t-1} because they need to be able to retain information about $\{x_i\}_{i=t-\delta+1}^{t-1}$. While not an explicit mechanism for copying, this derivation provides insight into why a dependency on h_{t-1} could be helpful.

C.3 Expressing linear attention with $\lambda_t \approx \mathbf{1}\ \forall t$

Recent parameterizations of transition matrices λ_t use the exponential of a negative number as opposed to the sigmoid function due to the saturation of the sigmoid (Gu et al., 2022). However, this parameterization still does not provide a mechanism to express linear attention with $\lambda_t \approx \mathbf{1}$ in a controlled way. To that end, De et al. (2024) propose to separate the lower-bound of λ_t from its selective component Δ_t by setting $\lambda_t = \exp(-c \cdot \text{softplus}(\omega) \cdot \Delta_t)$. With a selective parameter $0 \preceq \Delta_t \preceq \mathbf{1}$, an input independent component ω, a temperature c, and the softplus function, the transition matrix λ_t obeys the lower-bound $\lambda_t \succeq \exp(-c \cdot \text{softplus}(\omega))$. While the selective component Δ_t seems to be a crucial element for certain tasks in a linear RNN (Gu & Dao, 2023), we believe it also introduces a recency bias to the recurrence. We present the following theorem as evidence for this claim.

Theorem C.1. *Let $h_t = \lambda_t \cdot h_{t-1} + x_t$ define a 1-dimensional RNN parameterized by ω, b as $\lambda_t = \omega^\top x_t + b$. Let the input $x_t \sim \mathcal{N}(0,1)$, the bias term positive, and the gate be parameterized as $\lambda_t = \sigma(\omega^\top x_t + b)$. We define the expected memory of the RNN model $\mathcal{M}(\omega, b)$ as the expected length of sequence $\mathcal{T}$ in which we have $\lambda_t > 0.99$ for $t \in (0, \mathcal{T}]$. Then we have: $\mathcal{M}(\omega, b) \propto \exp\left(\|\omega\|_2^2\right)^{-1}$. Proof in App. F.3.*

Theorem C.1 shows an exponential decrease in memory capacity as the selective component Δ_t becomes more prominent. This means that the weights mapping an input x_t to Δ_t need to be initialized to 0 in order facilitate $\lambda_t \approx \mathbf{1}\ \forall t$. In that way, the model avoids the *recency bias* linked to selectivity and empirically improves the performance on the copy task.

D The Bedouin

In this section, we combine the findings from the previous sections into a dense variant of Mamba (Gu & Dao, 2023). After a short recap of Mamba, we introduce the fixed-point iteration and parametrization of our new model which we call The Bedouin. A diagram is available in Figure 5.

D.1 Mamba: Selective SSMs

Mamba is a multi-layer network, with an embedding size of d_{model}. A Mamba block is a matrix state diagonal linear RNN which first expands a sequence of embeddings by a factor of e to size $d_{\text{inner}} = e \times d_{\text{model}}$, and then computes an element-wise recurrence on the matrix hidden states $h_t \in \mathbb{R}^{d_{\text{state}} \times d_{\text{inner}}}$ as

$$h_t = \lambda_t \odot h_{t-1} + \mathbf{B}_t (\Delta_t x_t)^\top, \tag{8}$$

where $\lambda_t \in \mathbb{R}^{d_{\text{state}} \times d_{\text{inner}}}$ is an input-dependent state transition vector, $\mathbf{B}_t \in \mathbb{R}^{d_{\text{state}}}$ an input transition vector, $x_t \in \mathbb{R}^{d_{\text{inner}}}$ the input, and $\Delta_t \in \mathbb{R}^{d_{\text{inner}} \times d_{\text{inner}}}$ a diagonal matrix which acts an input normalization term. The matrices are parameterized as:

$$\lambda_t = \exp(-\lambda_{\log} \Delta_t), \qquad \lambda_{\log} = \exp(\omega),$$
$$\Delta_t = \text{diag}(\text{softplus}(\mathbf{W}_\Delta x_t + b_\Delta)), \qquad \mathbf{B}_t = \mathbf{W}_\mathbf{B} x_t,$$

with $\omega \in \mathbb{R}^{d_{\text{state}} \times d_{\text{inner}}}$, $\mathbf{W}_\Delta \in \mathbb{R}^{d_{\text{inner}} \times d_{\text{inner}}}$, $\mathbf{W}_\mathbf{B} \in \mathbb{R}^{d_{\text{state}} \times d_{\text{inner}}}$, and $b_\Delta \in \mathbb{R}^{d_{\text{inner}}}$. The output of a Mamba block $y_t \in \mathbb{R}^{d_{\text{inner}}}$ is a contraction of the matrix hidden state with $\mathbf{C}_t \in \mathbb{R}^{d_{\text{state}}}$

$$y_t^\top = \mathbf{C}_t^\top h_t, \quad \mathbf{C}_t = \mathbf{W}_\mathbf{C} x_t,$$

for $\mathbf{W_C} \in \mathbb{R}^{d_{\text{state}} \times d_{\text{inner}}}$. Note that Mamba proposes a skip connection of $y_t + \mathbf{D} \odot x_t$, where $\mathbf{D} \in \mathbb{R}^{d_{\text{inner}}}$ is an input-independent vector. Finally, the model output is usually scaled by a gated linear unit (GLU) as $\tilde{y}_t = \mathbf{G}_t \odot y_t$, where $\mathbf{G}_t = \text{SiLU}(\mathbf{W}_G x_t)$ is a non-linear function of the input.

D.2 The Bedouin iteration

Let us introduce the fixed-point iteration to the Mamba architecture. We represent the hidden state as h_t^ℓ, where t is the token index (i.e., indexing over the sequence dimension), and ℓ corresponds to the fixed-point iteration index (i.e., indexing over the depth dimension). The same notation is used for other variables to emphasize when they depend on the input and hidden state of the current iteration. We propose the following iteration to adapt Mamba (8) to the fixed-point mechanism for matrix state RNNs (5):

$$\begin{aligned} h_t^\ell &= \lambda_t \odot h_{t-1}^\ell + \bar{\mathbf{B}}_t^\ell \big(\Delta_t \mathbf{Q}_t^\ell x_t\big)^\top + \bar{\mathbf{B}}_t^\ell \big(\Delta_t \big(\mathbf{I} - \mathbf{Q}_t^\ell\big) y_t^{\ell-1}\big)^\top, \\ {y_t^\ell}^\top &= (\bar{\mathbf{C}}_t^\ell)^\top h_t^\ell. \end{aligned} \tag{6}$$

In order to limit the Lipschitz constant according to Theorem 2.1, we use L2 normalized $\bar{\mathbf{B}}_t^\ell$ and $\bar{\mathbf{C}}_t^\ell$. Furthermore, we replace the normalization term $(\mathbf{1} - \lambda_t)$ with Δ_t to stay compatible with the Mamba implementation. Expanding $y_t^{\ell-1}$ yields the recurrence on the matrix state

$$h_t^\ell = \lambda_t \odot h_{t-1}^\ell + \text{const} + \bar{\mathbf{B}}_t^\ell (\bar{\mathbf{C}}_t^{\ell-1})^\top h_t^{\ell-1} (\mathbf{I} - \mathbf{Q}_t^\ell)^\top \Delta_t,$$

where the last term nicely illustrates the two components which mix the channels of the hidden states: the low-rank matrix $\bar{\mathbf{B}}_t^\ell (\bar{\mathbf{C}}_t^{\ell-1})^\top$ mixes over the dimension d_{state}, while $(\mathbf{I} - \mathbf{Q}_t^\ell)^\top$ mixes over the dimension d_{inner}. This factorization significantly simplifies the fourth-order tensor mixer formulation introduced in (5) and performs well in practice.

Finally, Bedouin (6) can be expressed as Mamba (8)

$$h_t^\ell = \lambda_t \odot h_{t-1}^\ell + \bar{\mathbf{B}}_t^\ell \big(\Delta_t \tilde{x}_t^\ell\big)^\top, \tag{9}$$

with an adjusted input $\tilde{x}_t^\ell = \mathbf{Q}_t^\ell \big(x_t - y_t^{\ell-1}\big) + y_t^{\ell-1}$. In other words, one fixed-point step consists of a channel mixing using $\mathbf{Q}_t$, followed by a sequence mixing using Mamba. This separation of concerns allows to speed up the parallel recurrence in time using the Mamba implementation. To find a fixed-point, the two phases are repeated until convergence, i.e. $\frac{\|y^\ell - y^{\ell-1}\|_\infty}{\|y^\ell\|_\infty} < 0.1$. For a visual summary of the complete fixed-point iteration, please refer to Figure 5.

After convergence to a fixed-point, h_t^* and y_t^* present the hidden state and output of the dense matrix-valued RNN. Similar to Mamba, we apply a gated linear unit to the output

$$\tilde{y}_t^* = \mathbf{G}_t \odot \text{SiLU}(y_t^*)$$

using $\mathbf{G}_t \in \mathbb{R}^{d_{\text{inner}}}$ and the SiLU activation function.

D.3 Parameterization

Now, we will provide the details for the parameterization of each component in the Bedouin.

D.3.1 The transition matrices

Following the analyses in Sec. C, we propose some changes to the input-dependent parameters. Specifically, we adopt the definition of λ_t from Griffin (De et al., 2024)

$$\begin{aligned} \lambda_t &= \exp(-c \cdot \lambda_{\log} \cdot \Delta_t), & \lambda_{\log} &= \text{softplus}(\omega), \\ \Delta_t &= \text{diag}(\sigma(\mathbf{W}_\Delta^x x_t + b_\Delta)), & \mathbf{B}_t^\ell &= \mathbf{W}_\mathbf{B}^x x_t + \mathbf{W}_\mathbf{B}^y y_{t-1}^{\ell-1}, \end{aligned}$$

with their proposed hyperparameter choice $c = 8$, and model the dependence on the previous output with $\mathbf{C}_t^\ell = \mathbf{W}_\mathbf{C}^x x_t + \mathbf{W}_\mathbf{C}^y y_{t-1}^{\ell-1}$ and $\mathbf{G}_t = \mathbf{W}_\mathbf{G}^x x_t + \mathbf{W}_\mathbf{G}^y y_{t-1}^*$, respectively. Finally, we keep the skip connection in Mamba, but remove the short convolution due to the previous state dependency.

D.3.2 THE CHANNEL MIXER

In Sec. 2, we showed how to design diagonal linear recurrences that converge to a dense linear RNN via fixed-point iterations and how to train them. In this section, we turn our attention to the fixed-point dense object, and discuss a choice for $\mathbf{Q}_t$, where $\mathbf{A}_t = \mathbf{Q}_t^{-1}\Lambda_t$ as in equation 2, striking a balance between parameter efficiency and expressivity.

According to (Cirone et al., 2024b; Merrill et al., 2024), a key factor to increase expressivity in dense linear RNNs lies in effectively mixing information through the hidden state's dimensions. While using a non-structured input-dependent state transition matrix would be prohibitive both computationally and in terms of required parameters with $\mathcal{O}(d^3)$ cost, certain structures such as circulant matrices do not improve the expressivity due to being co-diagonalizable (Cirone et al., 2024b). In the following theorem, we start with the observation that a simple low-rank parameterization could provide the necessary expressivity:

Theorem D.1 (Informal)**.** *While diagonal transition RNNs are confined to learning linear filters over point-wise transformations of the input path, RNNs with hidden dimension N and input-dependent transition matrix of rank $\sim\log(N)$ achieve expressive universality: they can approximate* any *Path-to-Vector function arbitrarily well on compact domains, when N is sufficiently large. Proof in App. G.*

Inspired by Theorem D.1, we start with a simple low-rank form for the mixer $\mathbf{Q}_t = \sum_{i=1}^{r}\alpha_{it}\bar{\mathbf{u}}_{it}\bar{\mathbf{u}}_{it}^\top$, where r is the rank, α_{it} are scalar coefficients, and $\bar{\mathbf{u}}_{it}$ are unitary vectors. This structure has two benefits over a general input-dependent mixer: (1) the input-to-mixer mapping $x_t \to \mathbf{Q}_t$ requires only $\mathcal{O}(dr)$ instead of $\mathcal{O}(d^3)$ parameters, and (2) the mixing operation has a complexity of $\mathcal{O}(dr)$ instead of $\mathcal{O}(d^2)$.

While extremely simple and parallelizable, this parameterization requires further regularization. Following Theorem 2.1, $\mathbf{I}-\mathbf{Q}_t$ needs to be contractive which is satisfied if $\mathbf{Q}_t$ is either orthogonal or contractive as well. This can be controlled ensuring $\sum_i \alpha_{it} \leq 1$. Beyond that, we observe a problem of rank collapse: since the derivatives of the mixer w.r.t. the $\mathbf{u}_{it}$s are independent of each other, gradient-based optimization guides them in the same direction, resulting in a collapsed parameterization. To avoid that, one could either orthogonalize $\mathbf{u}_{it}$s or directly parametrize $\mathbf{Q}_t$ with orthogonal components using the following theorem:

Theorem D.2. ***(Householder factorization of unitary matrices (Urías, 2010))*** *A matrix $\mathbf{U} \in \mathbb{C}^{N\times N}$ is unitary if and only if for every vector $y \in \mathbb{C}^N$ there exists a set of Householder matrices $\{\mathbf{H}_1, \mathbf{H}_2, ..., \mathbf{H}_\ell\}$ with $1 \leq \ell \leq N$ such that $\mathbf{U}y = z\mathbf{H}_1\mathbf{H_2}...\mathbf{H}_\ell y$, where z is a phase factor.*

Following Theorem D.2, we propose to parametrize $\mathbf{Q}_t$ as the product of a number of generalized Householder matrices

$$\mathbf{Q}_t = \prod_{i=1}^{r}\left(\mathbf{I} - 2\alpha_{it}\cdot\bar{\mathbf{u}}_{it}\bar{\mathbf{u}}_{it}^\top\right), \tag{10}$$

where $0 < \alpha_{it} < 1$ and $\bar{\mathbf{u}}_{it}$s are a unit vectors. This avoids rank collapse by forcing $\mathbf{Q}_t$ to be full-rank, while $\mathbf{I}-\mathbf{Q}_t$ of rank r remains contractive and stabilizes the fixed-point iteration. $\mathbf{Q}_t$ also has negative eigenvalues as in (Grazzi et al., 2024). Therefore, we use the formulation based on the product of generalized Householder matrices and parameterize the $i = 1, ..., r$ reflections in (10) with

$$\mathbf{u}_{it}^\ell = \mathbf{W}_{\mathbf{u}_i}^x x_t + \mathbf{W}_{\mathbf{u}_i}^y y_{t-1}^{\ell-1}, \qquad \bar{\mathbf{u}}_{it}^\ell = \frac{\mathbf{u}_{it}^\ell}{\left\|\mathbf{u}_{it}^\ell\right\|_2}, \qquad \alpha_{it}^\ell = \sigma\left(\left(\mathbf{w}_{\alpha_i}^x\right)^\top x_t + \left(\mathbf{w}_{\alpha_i}^y\right)^\top y_{t-1}^{\ell-1}\right).$$

Comparision to DeltaNet. We note the following difference with (Yang et al., 2024b): DeltaNet uses a single generalized Householder reflection as a state transition matrix within one head. While more reflections can be added by introducing zeros into the sequence (Siems et al., 2025), the interactions between channels remain constrained to a single head. Our framework allows for a principled parametrization of dense linear RNNs as fixed-points of diagonal linear RNNs with various matrix structures $\mathbf{Q}_t$ to mix across all available channels.

E EXPERIMENTAL SETUP

In this section, we provide the experimental setup for the state tracking, copying, and mod arithmetic tasks.

E.1 TASKS

State Tracking The task of tracking state in the alternating group on five elements (A_5) is one of the tasks introduced in (Merrill et al., 2024) to show that linear RNNs and SSMs cannot solve state-tracking

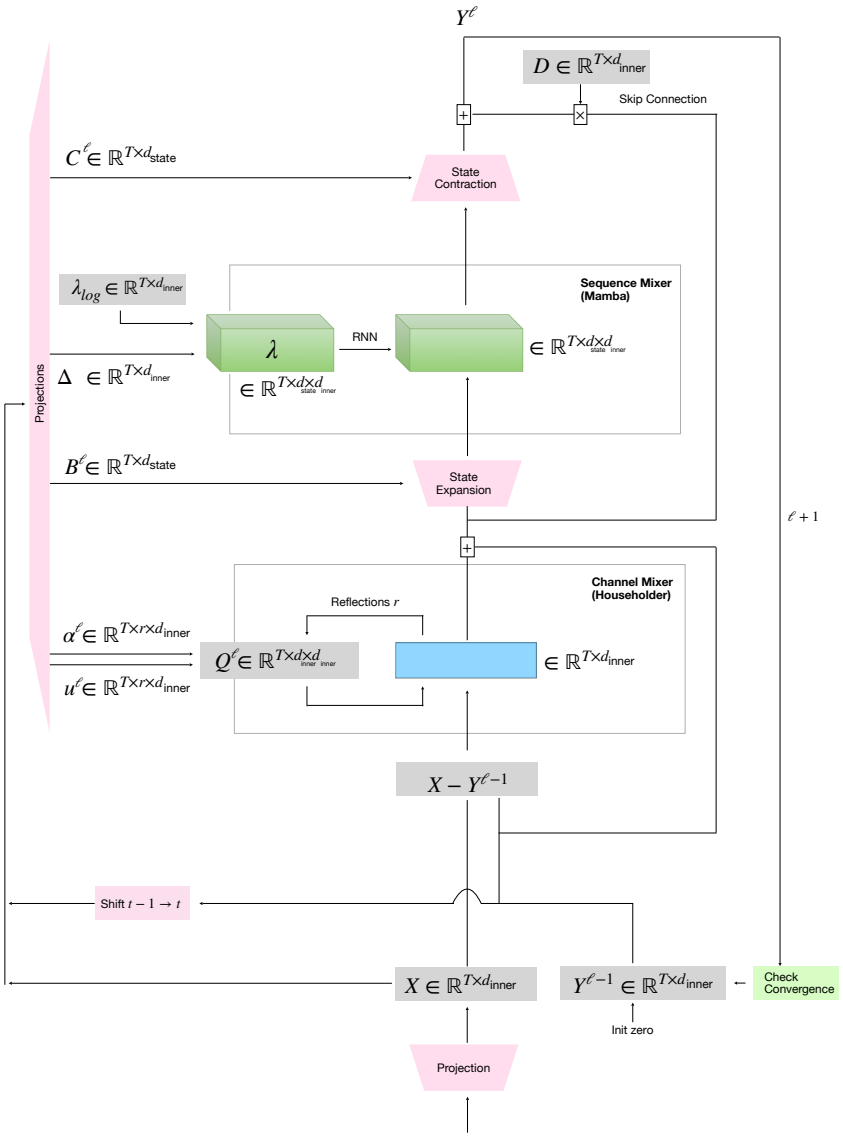

Figure 5: Sketch of our Beduin architecture

problems. A_5 is the simplest subset of S_5, the word problem involving tracking the permutation of five elements. In these tasks, a model is presented with an initial state and a sequence of permutations. As the output, the model is expected to predict the state that results from applying the permutations to the initial state. Solving these task with an RNN requires either a dense transition matrix or the presence of non-linearity in the recurrence. It is therefore a good proxy to verify the state-tracking ability of Bedouin. In order to investigate the out-of-distribution generalization ability of the model, we train the model with a smaller train sequence length and evaluate for larger (more than $\times 3$) sequence lengths.

Copying We use the copy task (Jelassi et al., 2024) in order to assess the memory capabilities of Bedouin. In this task, the model is presented with a fixed-size sequence of elements, and expected to copy a subsequence of it after receiving a special token signaling the start of the copying process. In order to investigate the out-of-distribution generalization ability of the model, we train the models with sequence length < 50, and assess the x2 length generalization following Jelassi et al. (2024) and Trockman et al. (2024).

The Chomsky Hierarchy Following Grazzi et al. (2024), we also evaluate Bedouin on the remaining unsolved task of the Chomsky Hierarchy of language problems introduced by Beck et al. (2024). Specif-

ically, we focus on the mod arithmetic task with brackets, for which the best performance reported so far according to Grazzi et al. (2024) is an accuracy of 0.2. Following the setup of Grazzi et al. (2024), we train on sequence lengths 3 to 40 and report scaled accuracies on test sequences of lengths 40 to 256. For Bedouin, we use a 2-layer model with $r = 4$ reflections, i.e. the best performing model in the A_5 experiment.

E.2 Training & Evaluation Details

State tracking. We train all models for 5 epochs, with a batch size of 128, 3 different random seeds, learning rate set to 0.0001, weight decay set to 0.01, gradient clipping 1.0, and the AdamW optimizer (Loshchilov & Hutter, 2017). We sample 1.6M samples from all the possible permutations, and split the data with a ratio of 4 to 1. We use the implementation and the hyperparameters provided by Merrill et al. (2024). We train the model for sequence length 16, and evaluate for sequence length 2 through 50.

Copying. We train all models for 10000 iterations, batch size 128, 3 different random seeds, learning rate 0.00001, weight decay 0.1, gradient clipping 1.0, the AdamW optimizer, and with linear learning rate decay after a 300 iterations warmup. The data is sampled randomly at the start of the training/evaluation. We use a vocab size of 29, a context length of 256, and train the model for copy sequence length in the range 5 to 50, and evaluate for the range 5 to 100. we use the implementation and the hyperparameters provided by Jelassi et al. (2024).

Mod arithmetic. Our models are trained for 100000 iterations, batch size 256, learning rate 0.001, weight decay 0.1, and no gradient clipping. The learning rate is decayed using a cosine scheduling by a factor of 0.001 after 10000 iterations of warmup. The data is randomly sampled at the start of training/evaluation. We use a vocab size of 12, with context length 256, and train data sequence length in the range 3 to 40, and the test/evaluation data in the range 40 to 256. We use the implementation and the hyperparameters provided by Beck et al. (2024) and Grazzi et al. (2024), which are the same hyperparameters used for training and evaluating the baselines.

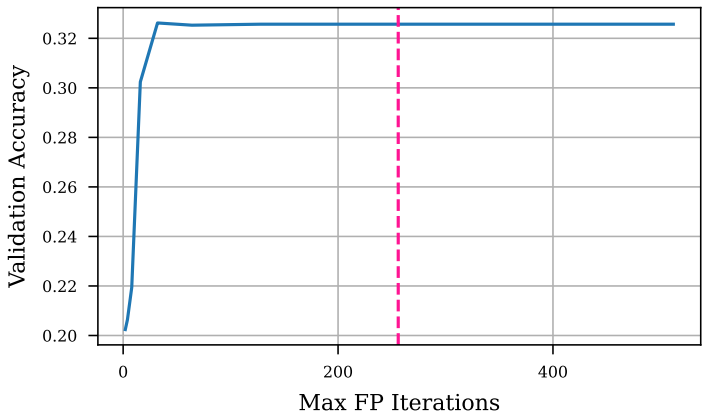

Figure 6: *Number of fixed-point iterations on the modular arithmetic task at test time. We report the validation accuracy after convergence for the number of fixed-point iterations caped at various values ranging from* 2 *to* 512*. The pink dashed line denotes the maximum sequence length during validation.*

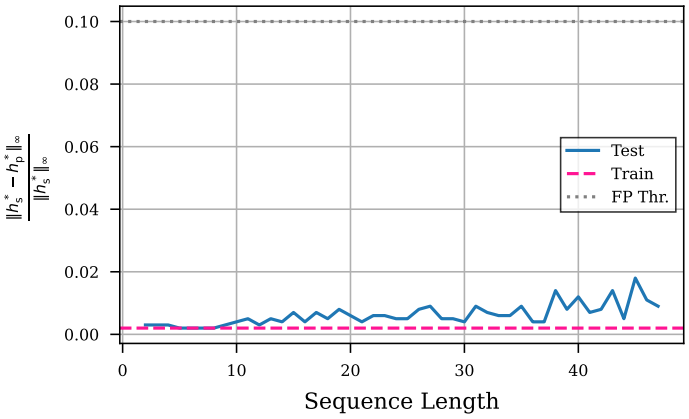

Figure 7: The difference between the fixed-point computed sequentially (i.e., computing the fixed-point for each token separately) and the fixed-point computed in parallel (i.e., computed through (6)) on the A_5 task trained on sequence length 16 to convergence. The x-axis denotes the test sequence length, and the y-axis the normalized difference. The dashed gray line denotes the threshold for stopping the fixed-point iterations.

F PROOFS

F.1 PROOF FOR THEOREM 2.1

We start the proof with the unrolled form of the linear RNN

$$f_\theta(x,h)_t = \sum_{\tau=0}^{t} \Lambda^{t-\tau}(\mathbf{I}-\Lambda)(\mathbf{Q}\mathbf{B}_\tau x_\tau + (\mathbf{I}-\mathbf{Q})h_\tau).$$

Note that in order to prove the theorem, we need to show that

$$\|f_\theta(x,h)_t - f_\theta(x,h')_t\|_2 < \|h-h'\|_2,$$

where h and h' are two arbitrary hidden states. From the unrolled form, this is equivalent to

$$\left\| \sum_{\tau=0}^{t} \Lambda^{t-\tau}(\mathbf{I}-\Lambda)(\mathbf{I}-\mathbf{Q})(h_\tau - h'_\tau) \right\|_2 < \|h-h'\|_2. \tag{11}$$

From the Cauchy-Schwarz inequality, we can upper-bound the LHS of (11) as

$$\left\| \sum_{\tau=0}^{t} \Lambda^{t-\tau}(\mathbf{I}-\Lambda)(\mathbf{I}-\mathbf{Q})(h_\tau - h'_\tau) \right\|_2 \leq \left\| \sum_{\tau=0}^{t} \Lambda^{t-\tau} \right\|_2 \cdot \|\mathbf{I}-\Lambda\|_2 \cdot \|\mathbf{I}-\mathbf{Q}\|_2 \cdot \left\| h_{\leq t} - h'_{\leq t} \right\|_2,$$

where $h_{\leq t}$ corresponds to the concatenation of the hidden states h_τ for $\tau \leq t$. Now to prove this product is $< \|h-h'\|_2$, consider the terms individually. Since $\left\| h_{\leq t} - h'_{\leq t} \right\|_2 \leq \|h-h'\|_2$, the remaining terms need to be <1. Assuming Λ is contractive, we use the Neumann series $\sum_{\tau=0}^{t} \Lambda^{t-\tau} \leq \mathbf{I}-\Lambda$ and get

$$\left\| \sum_{\tau=0}^{t} \Lambda^{t-\tau} \right\|_2 \cdot \|\mathbf{I}-\Lambda\|_2 \leq 1.$$

Finally, it remains to show that

$$\|\mathbf{I}-\mathbf{Q}\|_2 < 1.$$

This condition can be satisfied if $\mathbf{I}-\mathbf{Q}$ is contractive. This completes our proof. □

F.2 PROOF FOR THEOREM 2.2

We start the proof by setting $\delta := \frac{\partial \mathcal{L}}{\partial f}$ and $\mathbf{J}_x := \mathbf{J}_x(h^*)$. Then, we can write the backward propagation as $\frac{\partial \mathcal{L}}{\partial x} = (\mathbf{J}_x^*)^\top \delta$. In order to prove that the gradient computed at the fixed-point is a descent direction, we need to show that $\mathbf{J}_x^\top \delta$ is in the direction of $(\mathbf{J}_x^*)^\top \delta$, or in other words, we have $\delta^\top \mathbf{J}_x^* \mathbf{J}_x^\top \delta \geq 0$. This is equivalent to showing that the symmetric part of the matrix $\mathbf{J}_x^* \mathbf{J}_x^\top$ is positive semi-definite.

Now note that from (7) we have: $\mathbf{J}_x^* \mathbf{J}_x^\top = (\mathbf{I} - \mathbf{J}_h)^{-1} \mathbf{J}_x \mathbf{J}_x^\top$. From our assumption $\mathbf{J}_x = \mathbf{J}_h := \mathbf{J}$, we need to show that the symmetric part of the matrix $(\mathbf{I} - \mathbf{J})^{-1} \mathbf{J}\mathbf{J}^\top$ is positive semi-definite. Note that using the Neumann series, we can write:

$$(\mathbf{I} - \mathbf{J})^{-1} \mathbf{J} = \sum_{i=1}^{\infty} \mathbf{J}^i = \mathbf{J} \sum_{i=0}^{\infty} \mathbf{J}^i = \mathbf{J}(\mathbf{I} - \mathbf{J})^{-1}.$$

Therefore, we have $(\mathbf{I} - \mathbf{J})^{-1} \mathbf{J}\mathbf{J}^\top = \mathbf{J}(\mathbf{I} - \mathbf{J})^{-1} \mathbf{J}^\top$. Going back to the definition of positive semi-definiteness, we need to show that $\delta^\top \mathbf{J}(\mathbf{I} - \mathbf{J})^{-1} \mathbf{J}^\top \delta > 0$ for all δ. Setting $\omega = \mathbf{J}^\top \delta$, this is equivalent to having $\omega^\top (\mathbf{I} - \mathbf{J})^{-1} \omega$. Note that from our assumption for the Lipschitz constant of the function, we have $\|\mathbf{J}\|_2 < 1$, which means $(\mathbf{I} - \mathbf{J})$ has strictly positive eigenvalues. This completes our proof. □

F.3 PROOF FOR THEOREM C.1

Note that we can write $\mathcal{M}(\omega, b)$ as $\mathcal{M}(\omega, b) = \text{Prob}\big(\omega^\top x + b \leq \sigma^{-1}(0.99)\big)^{-1}$. Given the distribution of x, we can write $\text{Prob}\big(\omega^\top x + b \leq \sigma^{-1}(0.99)\big) = \Phi\left(\frac{\sigma^{-1}(0.99) - b}{\|\omega\|_2^2}\right)$. Given that $\Phi(\cdot)$ can be written as the integration of a quadratic exponential function w.r.t. the denominator of its argument, this completes our proof. □

G LOW-RANK EXPRESSIVENESS

In this section, we prove that SSMs with low-rank structure can be maximally expressive under weak assumptions on the growth of the rank with hidden dimension. To do this we first place ourselves in the general setting of (Cirone et al., 2024b), accordingly we consider models given by controlled differential equations of type[1]:

$$\mathrm{d}Y_s = \sum_{i=1}^{d_\omega} A^i Y_s \mathrm{d}\omega_s^i, \quad Y_0 \in \mathbb{R}^{d_Y} \tag{12}$$

Following the notation and methodology of (Cirone et al., 2024b)[B.4]), this can be written in terms of the Signature as

$$\mathbf{Y}((A^i)_i, Y_0, \omega)_t := Y_t = \sum_{I \in \mathbb{W}_{d_\omega}} (A^I Y_0)\, S^I(\omega)_{[0,t]} \tag{13}$$

where $\mathbb{W}_{d_\omega}$ is the set of words in the alphabet $[[d_\omega]] := \{1, \ldots, d_\omega\}$ (*i.e.* $\mathbb{W}_{d_\omega} = \bigcup_{n \geq 0} [[d_\omega]]^n$) and for a given word $I = i_1 \ldots i_n$ with $S^I(\omega)_{[0,t]}$ we refer to the Ith component of the *signature* tensor $S(\omega)_{[0,t]}$ *i.e.*

$$S^I(\omega)_{[0,t]} = \underbrace{\int \cdots \int}_{\substack{u_1 < \cdots < u_n \\ u_i \in [0,t]}} \mathrm{d}\omega_{u_1}^{i_1} \cdots \mathrm{d}\omega_{u_n}^{i_n}.$$

It follows directly from (13) that any linear readout of Y_t can be represented as a series in signature terms. As a result, these systems are fundamentally restricted to learning functions that closely approximate these convergent series.

Maximal expressivity is attained when *any* finite linear combination of signature terms can be approximated by a linear readout on Y_t via suitable configurations of the matrices A^i.

[1] For simplicity we have omitted the $d\xi$ term, as the results and proof change minimally in form but not in spirit.

Definition G.1. Fix a set of paths $\mathcal{X} \subseteq C^{1-var}([0,1];\mathbb{R}^d)$. We say that a sequence $(\mathcal{A}_N,\mathcal{Y}_N)_{N\in\mathbb{N}}$, where $\mathcal{Y}_N \subseteq \mathbb{R}^N$ and $\mathcal{A}_N \subseteq \mathbb{R}^{N\times N}$, achieves *maximal expressivity* for $\mathcal{X}$ whenever for any positive tolerance $\epsilon > 0$ and any finite linear combination coefficients $\alpha \in T(\mathbb{R}^d)$ there exist a choice of parameters $v,(A^i),Y_0$ in some $\mathbb{R}^N,\mathcal{A}_N,\mathcal{Y}_N$ in the sequence such that $v^\top \mathbf{Y}((A^i),Y_0,\omega).$ is uniformly close to $\langle \alpha, S(\omega)_{[0,\cdot]}\rangle$ up to an error of ϵ *i.e.*

$$\forall \epsilon > 0, \forall \alpha \in T(\mathbb{R}^d), \exists N \geq 0, \exists (v,(A^i),Y_0) \in \mathbb{R}^N \times \mathcal{A}_N^d \times \mathcal{Y}_N \text{ s.t.}$$
$$\sup_{(\omega,t)\in\mathcal{X}\times[0,1]} |\langle \alpha, S(\omega)_{[0,t]}\rangle - v^\top \mathbf{Y}((A^i),Y_0,\omega)_t| < \epsilon$$

If we are given a sequence of probabilities $\mathbb{P}_N$ on $\mathcal{A}_N^d \times \mathcal{Y}_N$ such that $\forall \epsilon > 0, \forall \alpha \in T(\mathbb{R}^d)$ it holds that

$$\lim_{N\to\infty} \mathbb{P}_N \left\{ \exists v \in \mathbb{R}^N \text{ s.t.} \sup_{(\omega,t)\in\mathcal{X}\times[0,1]} |\langle \alpha, S(\omega)_{[0,t]}\rangle - v^\top \mathbf{Y}((A^i),Y_0,\omega)_t| < \epsilon \right\} = 1 \tag{14}$$

then we say that $(\mathcal{A}_N,\mathcal{Y}_N,\mathbb{P}_N)_{N\in\mathbb{N}}$ achieves *maximal probabilistic expressivity* for $\mathcal{X}$.

As discussed in the main body of this work in (Cirone et al., 2024b) the authors prove that $(\mathbb{R}^{N\times N},\mathbb{R}^N,\mathbb{P}_N)$, where $\mathbb{P}_N$ is a Gaussian measure corresponding to the classical *Glorot* initialization scheme in deep learning, achieves *maximal probabilistic expressivity* for compact sets.

Albeit expressiveness is thus maximally attained the resulting matrices A_i are almost-surely dense, hence the models are not efficiently implementable. As the next result suggests, a possible alternative is given by low-rank matrices:

Proposition G.2. *The sequence of triplets* $(\mathbb{R}^{N\times N},\mathbb{R}^N,\mathbb{P}_N)$ *where* $\mathbb{P}_N$ *is such that*

- *the initial value has independent standard Gaussian entries* $[Y_0]_\alpha \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$,
- *the weight matrices are distributed as* $A^i \overset{\text{iid}}{\sim} \frac{1}{\sqrt{Nr_N}} W M^\top$ *with* W *and* M *independent* $N \times r_N$ *matrices having entries* $[W]_{\alpha,\beta},[M]_{\alpha,\beta} \overset{\text{iid}}{\sim} \mathcal{N}(0,1)$,
- *the rank parameter* r_N *satisfies* $r_N \to \infty$ *as* $N \to \infty$

achieves maximal probabilistic expressivity *for compact sets.*

Proof. Following (Cirone et al., 2024b)[B.3.5] we only need to prove a bound of type

$$\left\| \frac{1}{N} \langle A_I Y_0, A_J Y_0 \rangle_{\mathbb{R}^N} - \delta_{I,J} \right\|_{L^2(\mathbb{P}_N)} \leq (\kappa(|I|+|J|))!!\, o(1) \tag{15}$$

as in the full-rank Gaussian case.

We will place ourselves in the graphical setting of (Cirone et al., 2024a) and leverage the fact that (*c.f.* (Cirone et al., 2024a)[7.1]) their results and techniques naturally hold for rectangular matrices.

In our setting $\frac{1}{N}\langle A_I Y_0, A_J Y_0 \rangle_{\mathbb{R}^N}$ corresponds to a *product graph* $G_{I,J}$ corresponding to a ladder having $2|I|+2|J|$ edges as shown in figure 8. We can then use (Cirone et al., 2024a)[Prop. 2] to compute the square of the L^2 norm in equation (15), the only difference from the dense case is that half of the vertices (excluding the "middle" one) correspond to a space of dimension r_N while the rest to the standard N.

Since $r_N \to \infty$ and given the scaling $N^{-1}\big(Nr_N\big)^{-\frac{|I|+|J|}{2}}$, the admissible pairings of $G_{I,J}$ not of order $o(1)$ are only the leading ones. These correspond to product graphs with $\frac{|I|+|J|}{2}$ r_N-dimensional vertices and $\frac{|I|+|J|}{2}+1$ N-dimensional vertices. By the same reasoning as in the full-rank case, these are found to be just the identity pairings.

Moreover, all pairings of $G_{I,J} \sqcup G_{I,J}$ that do not result in an identity pairing in at least one of the two copies are $\mathcal{O}\big(\frac{1}{N\wedge r_N}\big)$ (instead of $\mathcal{O}\big(\frac{1}{N}\big)$). This follows as in the full-rank case.

Since the total number of admissible pairings of $G_{I,J} \sqcup G_{I,J}$ is $\big(4(|I|+|J|)\big)!!$, we conclude that equation (15) holds with $\kappa = 4$ and $o(1) := \mathcal{O}\big(\frac{1}{\sqrt{N\wedge r_N}}\big)$.

□

$$\frac{1}{N}\langle A_I Y_0,\ A_J Y_0\,\rangle \quad \equiv \quad \frac{1}{N}\,\frac{1}{(Nr_N)^{|I|+|J|}}$$

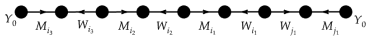

Figure 8: The *product graph* $G_{I,J}$ for $I=i_1 i_2 i_3$ and $J=j_1$.

Remark G.3. Following (Cirone et al., 2024a)[6.1] it's possible to prove that the W and M can be taken as having iid entries from a centred, symmetric but heavy tailed distribution given finiteness of even moments. This distributional choice comes useful in controlling the eigenvalues of $A=WM^\top$.

Remark G.4. While the proof crucially uses the assumption $r_N \to \infty$ as $N \to \infty$, at the same time we have not provided an argument against r_N not diverging. In figure 9 we present a counterexample, showing that if r_N does not diverge then the asymptotics differ from the dense ones, in particular some symmetries are "lost", impossible to recover due to unavoidable noise.

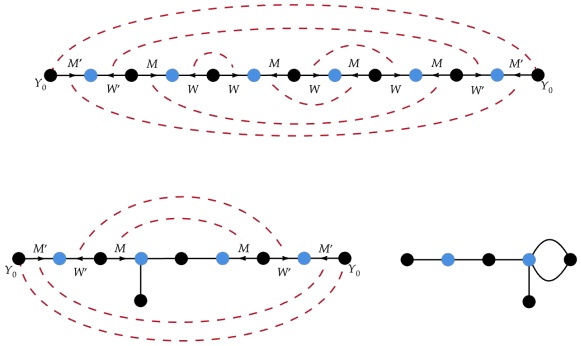

Figure 9: Admissible pairing different from the "identity" paring, but still leading to maximal asymptotic scaling in the bounded r_N case. Here, $I=12\neq 1112=J$, and we have highlighted in blue the vertices corresponding to the bounded dimension r_N. Recall that edges without arrows correspond to the matrix $\mathbf{I}$ (matrix of ones), and that two edges corresponding to matrices A and B which share direction and terminal vertices can be merged into the edge $A\odot B$.