RNN Generalization to Omega-Regular Languages

Charles Pert*, Dalal Alrajeh and Alessandra Russo

Imperial College London, UK

Abstract

Büchi automata (BAs) recognize ω-regular languages defined by formal specifications like linear temporal logic (LTL) and are commonly used in the verification of reactive systems. However, BAs face scalability challenges when handling and manipulating complex system behaviors. As neural networks are increasingly used to address these scalability challenges in areas like model checking, investigating their ability to generalize beyond training data becomes necessary. This work presents the first study investigating whether recurrent neural networks (RNNs) can generalize to ω-regular languages derived from LTL formulas. We train RNNs on ultimately periodic ω-word sequences to replicate target BA behavior and evaluate how well they generalize to out-of-distribution sequences. Through experiments on LTL formulas corresponding to deterministic automata of varying structural complexity, from 3 to over 100 states, we show that RNNs achieve high accuracy on their target ω-regular languages when evaluated on sequences up to $8\times$ longer than training examples, with 92.6% of tasks achieving perfect or near-perfect generalization. These results establish the feasibility of neural approaches for learning complex ω-regular languages, suggesting their potential as components in neurosymbolic verification methods.

Keywords

recurrent neural networks, omega-regular languages, linear temporal logic, büchi automata, length generalization

1. Introduction

Linear Temporal Logic (LTL) [1] formulas specify properties of system execution traces through ω-regular languages, which are composed of infinitely long sequences called ω-words. While regular languages are recognized by finite automata, ω-regular languages require Büchi automata (BAs) [2]. See Figure 1 for an example of a deterministic Büchi automaton (DBA). For a detailed introduction to these concepts, we refer the reader to [3].

While BAs provide exact solutions, they can become computationally expensive to handle and manipulate when representing complex behaviors. Recently, neurosymbolic methods have been used in verification contexts such as model checking [4], areas that traditionally rely on BAs. Characterizing whether neural networks can recognize ω-regular languages is a step toward enabling the development of additional approaches. Recent studies have shown that recurrent neural networks (RNNs) have the ability to generalize to the recognition of regular languages [5, 6, 7, 8], but this has not yet been shown specifically for ω-regular languages. Related works [9, 10] have used graph neural networks to analyze BA properties like emptiness checking and [11] demonstrated that Transformers [12] can generate satisfying ω-words for LTL formulas, yet the problem of generalization to the recognition of ω-regular languages remains open.

Extending the existing work on regular language generalization to ω-regular languages requires handling two practical challenges: (1) encoding: representing infinite sequences with finite-length sequences; (2) labeling sequences: computing acceptance labels for large batches of sequences. While finite automata accept a word when it terminates in an accepting state, BAs require ω-words to traverse accepting states infinitely often, meaning RNNs must learn to recognize eventually periodic behavior instead of just reaching an accepting state. We address challenge (1) by using ultimately periodic (UP) ω-words, which uniquely characterize their ω-regular languages [13]. This representation enables us to investigate whether RNNs can approximate the symbolic acceptance mechanisms of BAs.

OVERLAY 2025, 7th International Workshop on Artificial Intelligence and Formal Verification, Logic, Automata, and Synthesis, October 26th, 2025, Bologna, Italy

*Corresponding author.

charles.pert@imperial.ac.uk (C. Pert); dalal.alrajeh@imperial.ac.uk (D. Alrajeh); a.russo@imperial.ac.uk (A. Russo)

0009-0009-9790-6362 (C. Pert); 0000-0002-1365-8026 (D. Alrajeh); 0000-0002-3318-8711 (A. Russo)

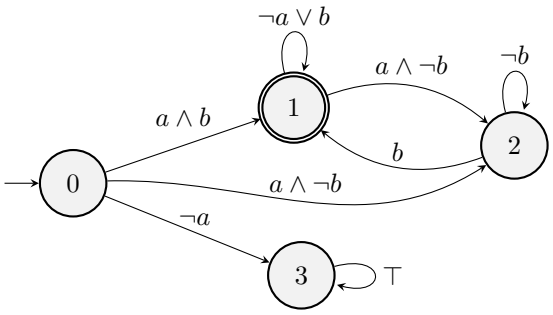


Figure 1: A DBA associated with the LTL formula $\mathbf{G}(a \rightarrow \mathbf{F}b) \wedge a$ (i.e., whenever a is true, it must eventually be followed by b, and a is initially true). Accepting states are double-circled. Transitions are labeled with propositional formulas. Input sequences, ω-words, are composed of symbols that represent assignments to the propositions. For example, the word $(a \wedge b)^\omega$ is accepted.

While existing model checking tools like Spot [14] can compute acceptance labels for specific sequences, this approach quickly becomes impractical for the large datasets required for neural network training and does not address sequence generation. Instead, we use Spot to construct a BA from each LTL formula, generate sequences using the BA representation and simulate acceptance by directly processing the generated sequences through the BA. We restrict our approach to DBAs to simplify this acceptance check, limiting our study to recurrence properties [15]. While this means that this study does not cover persistence properties, recurrence properties cover a large number of properties and are commonly used in verification. Using BAs as data generators provides some control over sampling diversity, which is essential since random sampling can induce data imbalance problems when accepted (or rejected) traces are rare.

This work establishes the feasibility of RNNs generalizing to the recognition of ω-regular languages while identifying challenges for future research. Our investigation complements ongoing neurosymbolic advances in verification, for example, neural certificates for model checking [4], neural circuit synthesis [16], as well as neural specification mining, learning finite automata or LTL$_f$ [17] from traces [18, 19, 20, 21].

The main contributions of this work are:

- We present the first empirical evidence that RNNs achieve high accuracy on ω-regular languages when evaluated at lengths up to $8\times$ longer than the sequences in their training distribution.[1]
- We provide an analysis of RNN generalization, showing that generalization is not limited to toy ω-regular languages and is robust across DBAs with over 100 states. We also show that the model complexity is correlated with the complexity of the BA recognizing the ω-regular language.

2. Method

Practical training of RNNs on ω-regular languages requires a finite representation of ω-words and an efficient method for computing acceptance labels for generated sequences given the infinite acceptance condition of BAs. Our approach resolves the finite encoding problem by only using UP ω-words of an ω-regular language, uv^ω, where u is a finite prefix and v is an infinitely repeating suffix; these ω-words uniquely characterize the language [13]. We encode uv^ω as $u\$v$ [13]. The alphabet size of a DBA is $2^{|P|} + 1$, where $|P|$ is the number of propositions present in the LTL formula used to construct the DBA. Each symbol in the alphabet represents either an assignment to all propositions or the separator symbol \$.

Importantly, the encoding $u\$v$ establishes a bijection between UP ω-words in the target ω-regular language and words in a derived regular language. The DBA can be algorithmically reconstructed from

[1] Code available at: https://github.com/pertcj/omega-generalization

the finite automaton recognizing this regular language [13]. Therefore, while our RNNs are learning regular languages (and leveraging their established ability [6, 7, 5]), they are learning canonical regular representations of the target DBA that preserve all structural information for reconstruction.

To determine the label of a $u\$v$ sequence, we simulate DBA behavior. Simulating the finite prefix u yields the state in which u terminates. For the suffix v, we compute the state transition matrix induced by reading v and use matrix exponentiation to determine reachability. The sequence is accepted if repeated application of v can reach a cycle containing an accepting state.

We illustrate how we sample $u\$v$ sequences with fixed length n from a DBA. We first sample the position k of $\$$ uniformly between 1 and $n-1$ and then sample u with length $k-1$ and v with length $n-k$. We sample u and v by uniformly sampling valid paths in the DBA. Many DBAs exhibit strong acceptance biases that skew random sampling toward rejection or acceptance. For example, the presence of accepting or rejecting sink states (states that can only transition back to themselves) can dominate uniform sampling. We address this through targeted sampling strategies to improve the balance of our sampled sequences: (1) we oversample sequences, determine their labels, and selectively filter them, targeting a balanced class distribution; (2) when sampling accepted sequences, we exclude transitions to rejecting sink states during sampling (and vice versa for rejected sequences); (3) when sampling rejected sequences, we prevent transitions to accepting states within the suffix v because if v contains a state transition to an accepting state, the resulting ω-word is likely to be accepted. It is still possible to sample equivalent rejected sequences as this constraint is not applied to u. By controlling the sampling of accepted and rejected sequences separately, the resulting dataset is more balanced compared to uniform sampling of sequences.

3. Experiments

Our experiments aim to answer the following research questions:

RQ1: Can RNNs generalize to the recognition of sequences from ω-regular languages when trained only on short UP ω-words?

RQ2: Do structural properties of the underlying DBAs influence (a) generalization performance and (b) learned model complexity?

To answer these questions, we use the ω-regular languages associated with two well-known LTL benchmarks. The first benchmark (**alaska_lift**) [22] consists of two encodings of safe lift behaviors (we use the variants with bug-fixes [23]) parameterized by the number of floors: encoding (a) uses a linear number of propositions per floor and encoding (b) uses a logarithmic number of propositions per floor. The second benchmark (**acacia_example**) [24] consists of 25 formulas specifying the behavior of arbiters and traffic light controllers [25]. For this benchmark, we use the negated versions of the formulas, which in our setting only flip the labels of the generated sequences.

For each LTL formula, we generate its corresponding DBA using Spot [14]. During training, we generate sequences on-the-fly with uniformly sampled lengths between 2 and 64. Once sampled, we transform sequences into one-hot encoded symbol embeddings for RNN processing. Our test data consists of 512 sampled sequences for each length between 2 and 512. We apply a 10-minute timeout for the construction of each DBA. We excluded automata with more than 200 states due to computationally expensive sequence generation, not RNN training. This constraint caused the exclusion of lift formulas for 4 or more floors and 2 formulas from **acacia_example**. The test data was balanced for the lift formulas and for 20 of the **acacia_example** formulas. However, the training data was not balanced for the lift formulas and the 3 **acacia_example** formulas with unbalanced test data.

Each experiment trains a single-layer vanilla RNN [26] with a hidden dimension of 256 (consistent with [6]), batch size 256 for 1×10^5 steps. We use linear warmup for the learning rate [27] from 1×10^{-8} to 1×10^{-3} at 20% of the training steps, the AMSGrad optimizer [28], and L_2 regularization with weight 5×10^{-4}. We minimize cross-entropy loss. Results are from a single seed due to computational budget. All experiments were conducted using an NVIDIA RTX 6000 Ada GPU. We measure in-distribution (ID)

accuracy, the mean accuracy of the test data in the training length range, and out-of-distribution (OOD) accuracy, the mean accuracy of the test data outside of the training length range.

To address RQ1, we assess whether RNNs can learn to classify ω-words by measuring performance on the test data, reporting ID accuracy (lengths 2-64) and OOD accuracy (lengths 65-512) in a summary table. To address RQ2a, we plot the relationship between the number of states in the DBAs and generalization performance. To address RQ2b, we plot the number of states against the parameter norm (L_2 norm) of the trained models (post-training). We report the Pearson correlation coefficients [29] and their statistical significance between the number of states and both OOD accuracy and the trained models' parameter norms to assess their linear correlations.

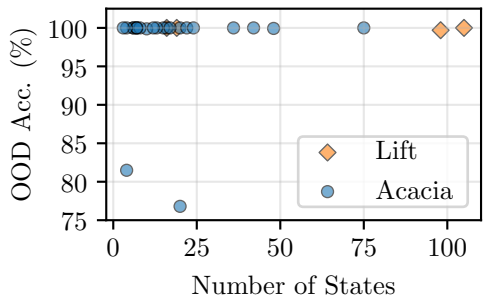
3.1. Results

In Table 1, we present the results answering RQ1. The majority of tasks (92.6%) had perfect or near-perfect generalization. Two tasks had OOD accuracies of 81.5% and 76.8%.

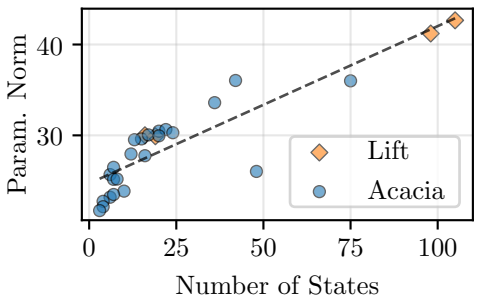
Table 1
Summary of RNN generalization performance on ω-regular language recognition tasks.

Performance Category	Tasks	Proportion	Mean ID Acc.	Mean OOD Acc.
Perfect ($> 99.9\%$)	23	85.2%	100.0%	100.0%
Near-Perfect ($98 - 99.9\%$)	2	7.4%	100.0%	99.8%
Good ($95 - 98\%$)	0	0%	–	–
Moderate ($90 - 95\%$)	0	0%	–	–
Poor ($< 90.0\%$)	2	7.4%	100.0%	79.1%
Overall	**27**	**100%**	**100.0%**	**98.4%**

In Figure 2, we present the plots comparing the number of states with OOD accuracy (Figure 2a) and the models' parameter norms (Figure 2b). Figure 2a reveals that generalization performance remains consistently high across automata of varying complexity, with 85.2% of tasks achieving perfect accuracy regardless of state count, indicating that RNN capacity is robust to the structural complexity of the underlying BA. OOD accuracy showed no significant correlation with the number of states in the DBA ($r = 0.115, p = 0.567$), suggesting that generalization performance does not necessarily degrade with increasing DBA complexity. Figure 2b shows a strong positive correlation between the number of states in the DBA and the parameter norm of the trained models ($r = 0.880, p < 0.001$), indicating that model complexity aligns with the complexity of the target ω-regular languages.



(a) OOD accuracy (acc.) remains consistently high across DBAs with large numbers of states.

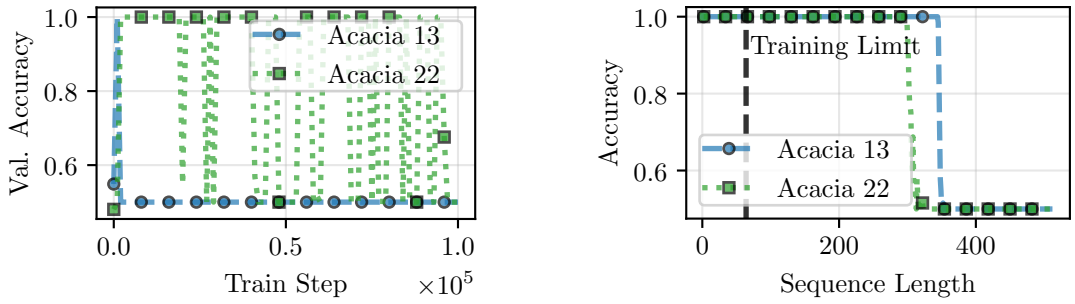


(b) Parameter (param.) norm of the trained RNNs shows a strong positive correlation with the number of states in the DBAs.

Figure 2: Diamond and circle markers distinguish between **alaska_lift** and **acacia_example** benchmarks.

We now examine the two tasks that exhibited poor generalization to understand their failure modes. During training, both tasks demonstrate unstable validation patterns that predict their generalization

failures (see Figure 3a). *Acacia 13* achieves 100% validation accuracy initially but then degrades to random chance (50%), while *Acacia 22* oscillates between 100% and 50% throughout training, indicating that neither model converges to a stable representation of the ground truth ω-regular language. This instability corresponds to poor length generalization, as shown in Figure 3b, where both tasks achieve perfect accuracy on shorter OOD sequences before experiencing sharp degradation to 50% at longer lengths (degradation begins at 345 and 300 for *Acacia 13* and *Acacia 22*, respectively). This degradation suggests that these models converged to local minima, achieving perfect in-distribution accuracy without learning the ground truth ω-regular language. The DBAs of both failure cases contained accepting sink states. If a sequence reaches such a state, the RNN must encode this information when processing all subsequent symbols. Further investigation is needed to confirm this hypothesis.



(a) Each model's validation (val.) accuracy on a set of 1024 sampled sequences of length 512 during training.

(b) Range evaluation displaying each model's accuracy on test data across sequence lengths.

Figure 3: Validation accuracy and length generalization for tasks with poor generalization.

4. Conclusion and Future Work

Our experiments demonstrate that RNNs generalize to the recognition of ω-regular languages from short UP ω-words. Across 27 tasks with a diverse range of system behaviors, we achieved perfect or near-perfect generalization in 92.6% of cases when testing on sequences up to 8 times longer than training examples. This finding remained consistent across automata ranging from 3 to 105 states, demonstrating that the approach scales to realistic verification problems. These results provide a foundation for developing *differentiable Büchi automata*, components within neurosymbolic systems. These components might behave as monitors in reinforcement learning or enable gradient-based search in model checking.

This work has several limitations that future research should address. Sampling sequences from complex automata (>200 states) proved impractical despite our attempts to speed up the process. Our experiments were also restricted to DBAs as a first step; notably, DBAs cannot represent all ω-regular languages [3]. Further investigation of the failure cases is necessary to develop improved training methods. Despite these limitations, this work provides strong evidence that neural approaches could further enhance neurosymbolic methods by offering more scalable, differentiable alternatives to traditional automata-theoretic methods. Exploring interpretability by adapting existing automata extraction techniques (e.g. [30, 31]) may enable verification of the learned representations required for safety-critical applications.

Acknowledgments

This work was supported by the UK EPSRC grant 2760033. The authors would like to thank Frederik Kelbel for reading the paper and the reviewers for their constructive feedback.

Declaration on Generative AI

During the preparation of this work, the authors used Claude Sonnet 4 in order to: Paraphrase and reword. After using this tool, the authors reviewed and edited the content as needed and take full responsibility for the publication's content.

References

[1] A. Pnueli, The Temporal Logic of Programs, in: 18th Annual Symposium on Foundations of Computer Science (sfcs 1977), 1977, pp. 46–57. doi:10.1109/SFCS.1977.32.

[2] J. R. Büchi, On a Decision Method in Restricted Second Order Arithmetic, 1990, pp. 425–435. doi:10.1007/978-1-4613-8928-6_23.

[3] Baier, Christel and Katoen, Joost-Pieter, Principles of Model Checking, 2008. URL: https://mitpress.mit.edu/9780262026499/principles-of-model-checking/.

[4] Mirco Giacobbe and Daniel Kroening and Abhinandan Pal and Michael Tautschnig, Neural Model Checking, in: The Thirty-eighth Annual Conference on Neural Information Processing Systems, 2024. URL: https://openreview.net/forum?id=dJ9KzkQ0oH.

[5] Alexandra Butoi and Ghazal Khalighinejad and Anej Svete and Josef Valvoda and Ryan Cotterell and Brian DuSell, Training Neural Networks as Recognizers of Formal Languages, in: The Thirteenth International Conference on Learning Representations, 2025. URL: https://openreview.net/forum?id=aWLQTbfFgV.

[6] G. Delétang, A. Ruoss, J. Grau-Moya, T. Genewein, L. K. Wenliang, E. Catt, C. Cundy, M. Hutter, S. Legg, J. Veness, P. A. Ortega, Neural Networks and the Chomsky Hierarchy, in: 11th International Conference on Learning Representations, 2023.

[7] Svete, Anej and Chan, Robin and Cotterell, Ryan, On Efficiently Representing Regular Languages as RNNs, in: Findings of the Association for Computational Linguistics: ACL 2024, Association for Computational Linguistics, 2024, pp. 4118–4135. URL: https://aclanthology.org/2024.findings-acl.244/. doi:10.18653/v1/2024.findings-acl.244.

[8] Merrill, William and Weiss, Gail and Goldberg, Yoav and Schwartz, Roy and Smith, Noah A. and Yahav, Eran, A Formal Hierarchy of RNN Architectures, in: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, Association for Computational Linguistics, 2020, pp. 443–459. URL: https://aclanthology.org/2020.acl-main.43/. doi:10.18653/v1/2020.acl-main.43.

[9] Stammet, Christophe and Dotti, Prisca and Ultes-Nitsche, Ulrich and Fischer, Andreas, Analyzing Büchi Automata with Graph Neural Networks, arXiv preprint arXiv:2206.09619 (2022).

[10] Stammet, Christophe and Ultes-Nitsche, Ulrich and Fischer, Andreas, Universality of Büchi Automata: Analysis with Graph Neural Networks, IEEE Access 11 (2023).

[11] C. Hahn, F. Schmitt, J. U. Kreber, M. N. Rabe, B. Finkbeiner, Teaching Temporal Logics to Neural Networks, in: International Conference on Learning Representations, 2021. URL: https://openreview.net/forum?id=dOcQK-f4byz.

[12] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, I. Polosukhin, Attention Is All You Need, in: Advances in Neural Information Processing Systems, volume 30, 2017. URL: https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

[13] Calbrix, Hugues and Nivat, Maurice and Podelski, Andreas, Ultimately Periodic Words of Rational ω-Languages, in: Mathematical Foundations of Programming Semantics, 1994, pp. 554–566.

[14] A. Duret-Lutz, E. Renault, M. Colange, F. Renkin, A. G. Aisse, P. Schlehuber-Caissier, T. Medioni, A. Martin, J. Dubois, C. Gillard, H. Lauko, From Spot 2.0 to Spot 2.10: What's New?, in: Proceedings of the 34th International Conference on Computer Aided Verification (CAV'22), volume 13372 of *Lecture Notes in Computer Science*, 2022, pp. 174–187. doi:10.1007/978-3-031-13188-2_9.

[15] Z. Manna, A. Pnueli, A Hierarchy of Temporal Properties (invited paper, 1989), in: Proceedings of

the Ninth Annual ACM Symposium on Principles of Distributed Computing, PODC '90, 1990, p. 377–410. doi:10.1145/93385.93442.

[16] F. Schmitt, C. Hahn, M. N. Rabe, B. Finkbeiner, Neural Circuit Synthesis from Specification Patterns, in: Advances in Neural Information Processing Systems, volume 34, 2021, pp. 15408–15420. URL: https://proceedings.neurips.cc/paper_files/paper/2021/file/8230bea7d54bcdf99cdfe85cb07313d5-Paper.pdf.

[17] G. De Giacomo, M. Y. Vardi, Linear Temporal Logic and Linear Dynamic Logic on Finite Traces, in: Proceedings of the Twenty-Third International Joint Conference on Artificial Intelligence, IJCAI '13, 2013, p. 854–860.

[18] Luo, Weilin and Han, Tingchen and Qiu, Junming and Wan, Hai and Du, Jianfeng and Peng, Bo and Xiao, Guohui and Liu, Yanan, NADA: Neural Acceptance-Driven Approximate Specification Mining, Proc. ACM Softw. Eng. 2 (2025). URL: https://doi.org/10.1145/3728956. doi:10.1145/3728956.

[19] H. Wan, P. Liang, J. Du, W. Luo, R. Ye, B. Peng, End-to-End Learning of LTLf Formulae by Faithful LTLf Encoding, Proceedings of the AAAI Conference on Artificial Intelligence 38 (2024). URL: https://ojs.aaai.org/index.php/AAAI/article/view/28757. doi:10.1609/aaai.v38i8.28757.

[20] W. Luo, P. Liang, J. Du, H. Wan, B. Peng, D. Zhang, Bridging LTLf Inference to GNN Inference for Learning LTLf Formulae, Proceedings of the AAAI Conference on Artificial Intelligence 36 (2022). URL: https://ojs.aaai.org/index.php/AAAI/article/view/21221. doi:10.1609/aaai.v36i9.21221.

[21] H. Walke, D. Ritter, C. Trimbach, M. Littman, Learning Finite Linear Temporal Logic Specifications with a Specialized Neural Operator, arXiv preprint arXiv:2111.04147 (2021).

[22] M. D. Wulf, L. Doyen, N. Maquet, J. Raskin, Antichains: Alternative Algorithms for LTL Satisfiability and Model-Checking, in: Tools and Algorithms for the Construction and Analysis of Systems, volume 4963 of *Lecture Notes in Computer Science*, 2008, pp. 63–77. doi:10.1007/978-3-540-78800-3_6.

[23] V. Schuppan, Towards a Notion of Unsatisfiable and Unrealizable Cores for LTL, Science of Computer Programming - SCP 77 (2012). doi:10.1016/j.scico.2010.11.004.

[24] Filiot, Emmanuel and Jin, Naiyong and Raskin, Jean-François, An Antichain Algorithm for LTL Realizability, in: Proceedings of the 21st International Conference on Computer Aided Verification, CAV '09, 2009, p. 263–277. doi:10.1007/978-3-642-02658-4_22.

[25] Schuppan, Viktor and Darmawan, Luthfi, Evaluating LTL Satisfiability Solvers, in: Automated Technology for Verification and Analysis, 2011, pp. 397–413.

[26] J. L. Elman, Finding Structure in Time, Cognitive Science 14 (1990) 179–211. doi:https://doi.org/10.1207/s15516709cog1402_1.

[27] D. S. Kalra, M. Barkeshli, Why Warmup the Learning Rate? Underlying Mechanisms and Improvements, in: Advances in Neural Information Processing Systems, volume 37, 2024, pp. 111760–111801. URL: https://proceedings.neurips.cc/paper_files/paper/2024/file/ca98452d4e9ecbc18c40da2aa0da8b98-Paper-Conference.pdf.

[28] S. J. Reddi, S. Kale, S. Kumar, On the Convergence of Adam and Beyond, in: International Conference on Learning Representations, 2018. URL: https://openreview.net/forum?id=ryQu7f-RZ.

[29] D. Freedman, R. Pisani, R. Purves, Statistics, 4th ed., 2007. URL: https://wwnorton.co.uk/books/9780393930436-statistics.

[30] W. Merrill, N. Tsilivis, Extracting Finite Automata from RNNs Using State Merging, CoRR abs/2201.12451 (2022). URL: https://arxiv.org/abs/2201.12451.

[31] Weiss, Gail and Goldberg, Yoav and Yahav, Eran, Extracting Automata from Recurrent Neural Networks Using Queries and Counterexamples, in: Proceedings of the 35th International Conference on Machine Learning, PMLR, 2018, pp. 5247–5256. URL: https://proceedings.mlr.press/v80/weiss18a.html.