A Data-Scaling Sweet Spot in Structured Algorithmic Learning

Shin So
Yonsei University

Kyelim Lee
Yonsei University

Albert No ALBERTNO@YONSEI.AC.KR
Yonsei University

Abstract

Data scaling is usually framed through an onset question: how much data is needed before a model generalizes? We study a different, update-based scaling question: after generalization is possible, which dataset size reaches a fixed validation exact-match threshold in the fewest optimizer updates under a fixed training protocol? In Needleman–Wunsch (NW) matrix generation, a structured-output dynamic-programming task, small Transformers reach high validation exact-match accuracy in the fewest optimizer updates at an intermediate dataset size, not at the largest one. Past this sweet spot, larger datasets still generalize but require more gradient updates. A random-suffix probe strengthens the interpretation: when an unstructured random suffix is appended to each NW target, the rule-governed NW component reaches high training accuracy earlier than the arbitrary suffix, and this structured–random gap widens in the same regimes where validation improves. Together with the observation that, near the onset of weak validation competence, increasing data can reduce the updates needed to reach high training exact-match accuracy, this shows that data scaling changes both rule discovery and residual exact fitting. The result separates the critical data size for making generalization attainable from the update-optimal data size for reaching a fixed validation exact-match threshold.

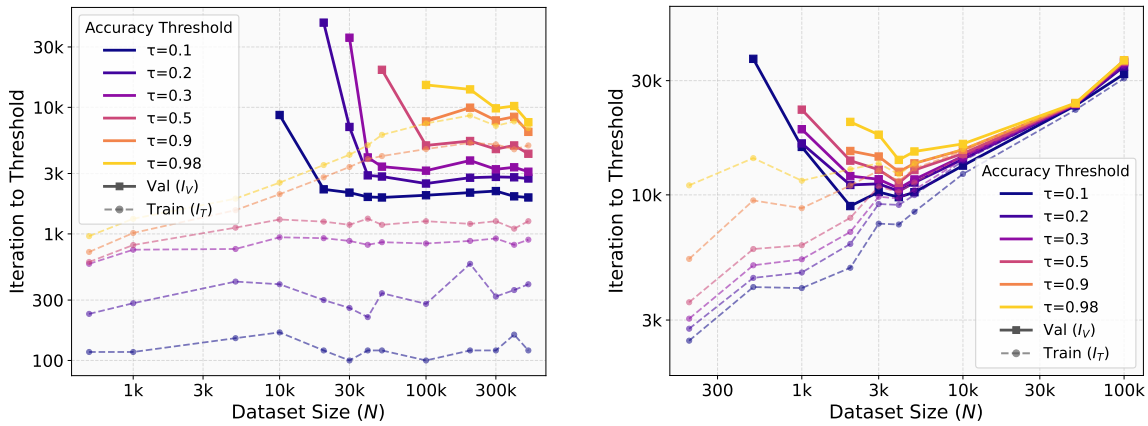
1. Introduction

Empirical scaling laws usually treat training data as a resource: with suitable compute and model scaling, additional data improves loss, accuracy, or compute allocation [4–6]. Grokking and critical-data-size studies give a sharper version of this view in controlled algorithmic settings. Below a task-dependent threshold, a model may fit the training set without discovering the rule; above it, validation accuracy becomes attainable and is typically expected to require fewer optimization steps as data increases [8, 11, 15, 19]. These works primarily address an *onset* question: when does generalization become possible?

We ask a complementary data-scaling question: *after generalization is possible, which dataset size reaches a fixed validation exact-match threshold in the fewest optimizer updates?* This distinction matters because update count is the unit directly controlled by most fixed-budget training protocols. A larger dataset may provide better coverage of the rule, but it also changes the optimization problem faced at every update. The dataset size that first permits generalization need not be the dataset size with the shortest threshold-reaching time.

We study this question in Needleman–Wunsch (NW) matrix generation, a structured-output task derived from sequence alignment [14]. Given two short strings, the NW dynamic program produces

© S. So, K. Lee & A. No.



(a) Three-digit multiplication

(b) NW matrix generation, $L = 5$

Figure 1: **A data-scaling sweet spot appears only for the structured-output NW task.** Each curve reports the first optimizer update at which exact-match accuracy reaches threshold τ; lower is faster. Solid lines show validation threshold-reaching iterations $I_V(\tau; N)$ and dashed lines show training threshold-reaching iterations $I_T(\tau; N)$. In the matched multiplication baseline, larger datasets do not create a pronounced post-onset slowdown under this analysis. In NW matrix generation, validation becomes fastest at an intermediate N: small datasets are structurally insufficient, intermediate datasets reach validation thresholds in the fewest updates, and larger datasets still generalize but require more updates. The dashed NW curves also show that, near the onset of weak validation competence, more data can make high training exact match easier rather than harder.

a score matrix whose entries recursively depend on neighboring cells. A decoder-only Transformer is trained to generate the entire matrix, and exact-match accuracy counts an example as correct only when all matrix entries are correct. NW is useful here not because biological alignment is our application, but because it is a compact high-dimensional output testbed: a short input induces a longer object with many mutually constrained cells.

Figure 1 previews the central phenomenon. In our matched three-digit multiplication baseline, once validation generalization is attainable, we do not observe a pronounced large-data slowdown under the same threshold-reaching analysis. NW behaves differently. Validation threshold-reaching times improve from the insufficient-data regime to an intermediate dataset size, then become slower again at larger dataset sizes. We call this interior optimum a *dataset-size sweet spot*. The finding is update-based: it concerns gradient updates under a fixed optimizer, model family, batch size, and evaluation rule, not asymptotic accuracy or epoch-normalized sample efficiency.

Our contributions are threefold. First, we identify a post-onset data-scaling effect in NW matrix generation: the update-optimal validation dataset size is interior rather than largest. Second, we show that the same data range changes training dynamics: near the onset of weak validation competence, increasing data can reduce the updates needed to reach high training exact-match accuracy. Third, we introduce a random-suffix probe that provides behavioral evidence for rule-mediated learning: the structured NW component separates from an appended random component precisely when validation improves. By rule-mediated learning, we mean improvement on reusable structure induced by the

NW recurrence, rather than only fitting independent input-output strings. These findings support a two-pressure account of data scaling in which additional examples first improve structural coverage, but after this benefit saturates, residual exact-match fitting costs can dominate.

2. Problem and Metrics

Needleman–Wunsch matrix generation. Let $\Sigma = \{A, C, G, T\}$ and $x, y \in \Sigma^L$. The target is a dynamic-programming matrix $M(x, y) \in \mathbb{Z}^{(L+1)\times(L+1)}$ with boundary values $M_{0,0} = 0$, $M_{i,0} = ig$, $M_{0,j} = jg$, and recurrence

$$M_{i,j} = \max\{M_{i-1,j-1} + s(x_i, y_j),\ M_{i-1,j} + g,\ M_{i,j-1} + g\}. \tag{1}$$

We use match score 5, mismatch score -4, gap penalty $g = -5$, and main length $L = 5$, giving a 6×6 target matrix. The matrix is serialized row-major and predicted autoregressively. Exact-match accuracy is the fraction of examples for which the full matrix is generated correctly.

Threshold-reaching metrics. Let $A_T(t; N)$ and $A_V(t; N)$ be training and validation exact-match accuracy after t optimizer updates. For threshold τ, define

$$I_T(\tau; N) = \min\{t : A_T(t; N) \geq \tau\}, \qquad I_V(\tau; N) = \min\{t : A_V(t; N) \geq \tau\}. \tag{2}$$

Runs that do not reach the threshold within the budget are treated as censored. We summarize the data-scaling curve by two quantities: the empirical critical data size $N_c(\tau)$, the smallest observed N with $I_V(\tau; N)$, and the update-optimal size $N^\star(\tau) = \arg\min_N I_V(\tau; N)$ among uncensored runs. The usual critical-data-size story concerns N_c; our main claim is that $N^\star$ can be distinct and interior.

3. Data-Scaling Results

3.1. NW has a post-onset validation sweet spot

Figure 2 expands the NW sweep across depths. Across model sizes and thresholds, the validation threshold-reaching curve is non-monotonic in dataset size. In the small-data regime, validation threshold-reaching times are absent or delayed. Around intermediate sizes, validation thresholds are reached in fewer optimizer updates. At larger sizes, validation still becomes possible, but the number of updates needed to reach the same exact-match threshold increases. Thus, for fixed high exact-match thresholds, the empirical relationship is

$$N_c(\tau) < N^\star(\tau) < N_{\max},$$

where $N_{\max}$ is the largest dataset size in the sweep.

This is not a claim that larger datasets lower final attainable accuracy. Nor is it a claim about epoch-measured sample efficiency. It is a statement about update-based convergence under a fixed training protocol. If one counts passes through the dataset, larger datasets naturally look more favorable because each epoch contains more examples. Data-scaling analyses should therefore specify which resource axis is being studied: examples, epochs, compute, or optimizer updates.

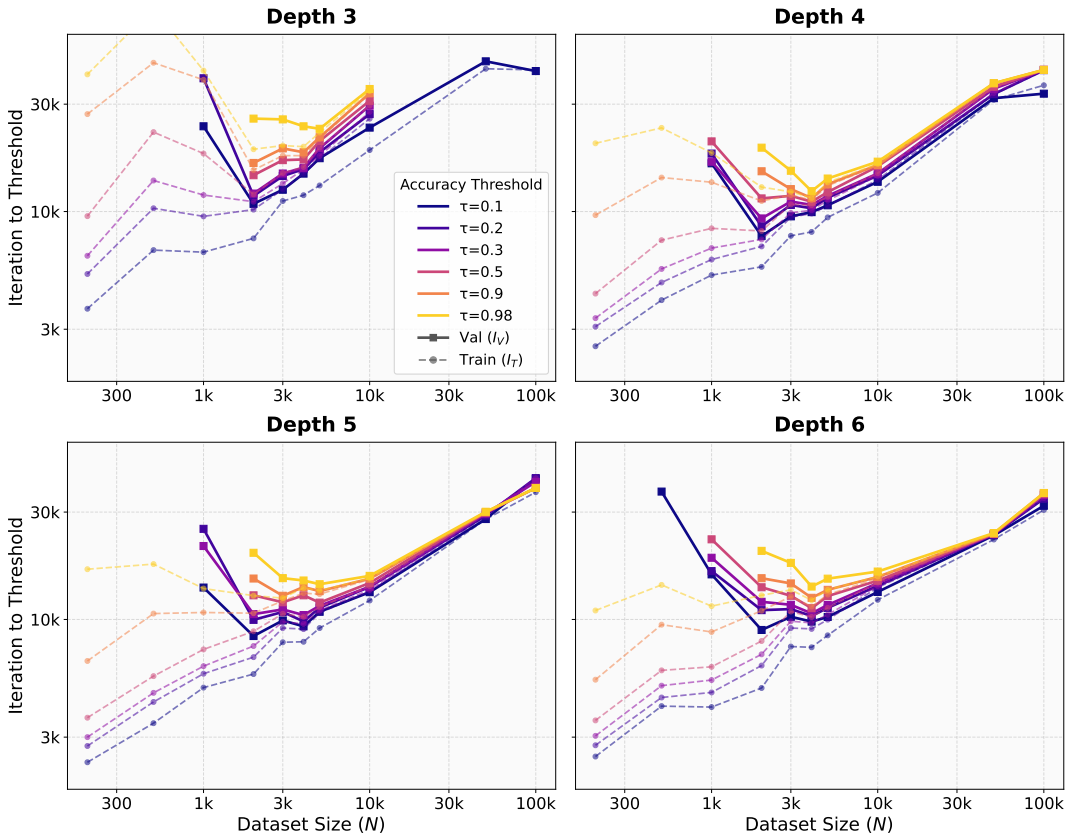
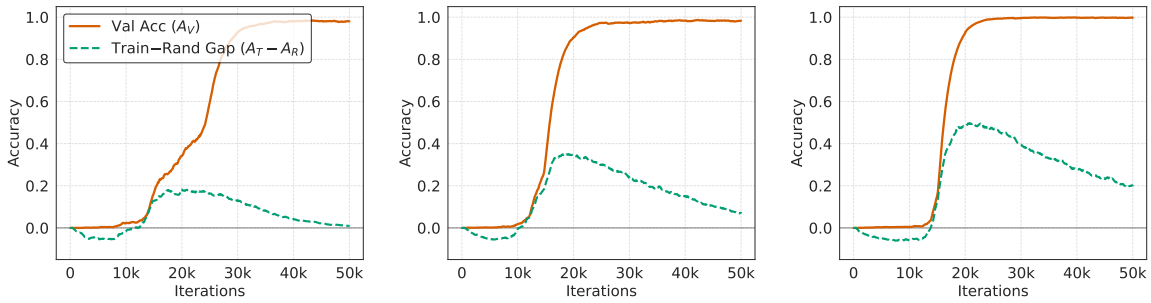


Figure 2: **NW validation convergence is non-monotonic across depths.** Multi-threshold NW sweeps for depths $D \in \{3, 4, 5, 6\}$ at $L = 5$. Colors denote exact-match thresholds τ, solid lines show validation threshold-reaching iterations $I_V(\tau; N)$, and dashed lines show training threshold-reaching iterations $I_T(\tau; N)$. Across depths, validation reaches strict thresholds fastest at intermediate dataset sizes, while the dashed curves show that training threshold-reaching can improve near the onset of weak validation competence.

3.2. More data can reduce training threshold-reaching time

The dashed curves in Figure 1 show a second effect. Near the dataset sizes where weak validation thresholds first become reachable, increasing N can reduce $I_T(\tau; N)$ for high training thresholds. This is surprising under a purely example-wise memorization view: if fitting the training set only required storing independent input-output pairs, more examples should make high-threshold training exact match slower in update count.

A more plausible interpretation is rule-mediated fitting. Once the model learns part of the NW recurrence, that structure can explain many training examples and many output cells at once. Training exact match is then not merely memorization followed by generalization; partial generalization can reduce the cost of fitting the training set. This aligns with mechanistic accounts of grokking in which memorizing and generalizing solutions differ in efficiency [9, 13, 16].



$N = 4$k $N = 5$k $N = 10$k

Figure 3: **Random suffixes separate rule-mediated progress from arbitrary fitting.** Each panel trains on NW targets augmented with an independently sampled four-bit random suffix. Orange shows NW validation accuracy A_V, and dashed green shows $G = A_T - A_R$, where A_T is training accuracy on the structured NW matrix and A_R is training accuracy on the random suffix. Since the suffix is input-independent, A_R measures arbitrary example-specific fitting. Validation begins to rise when $G > 0$, and initially G tracks A_V, equivalently $A_T \approx A_R + A_V$. Thus training accuracy decomposes into suffix-like memorization plus rule-mediated fitting; later separation suggests residual memorization after the NW rule is largely available.

3.3. Random suffixes isolate the structured component

The sweet spot could be misread as ordinary memorization difficulty: perhaps larger datasets simply require more arbitrary examples to be stored, a concern motivated by the ability of neural networks to fit arbitrary or noisy targets [1, 3, 18]. We therefore append an independently sampled four-bit random suffix to each NW target. Let A_T be training accuracy on the structured NW matrix, A_V validation accuracy on the NW matrix, and A_R training accuracy on the random suffix. Since the suffix is input-independent, A_R measures arbitrary example-specific fitting, while $G = A_T - A_R$ measures NW fitting beyond this random baseline.

Figure 3 shows that validation begins to rise when $G > 0$. During the early transition, G closely tracks A_V, equivalently $A_T \approx A_R + A_V$. Thus training accuracy decomposes into suffix-like memorization plus rule-mediated fitting: the same NW rule that improves validation also helps fit the training matrices. When the curves later separate, the NW rule is largely available, and the remaining exact-match burden is increasingly residual memorization rather than new transferable structure.

3.4. The effect is task-dependent

The multiplication baseline in Figure 1 is important because it uses the same style of threshold-reaching analysis and exact-match evaluation, but in our matched three-digit multiplication setting we do not observe the same pronounced large-data slowdown. This suggests that the sweet spot is not an artifact of plotting threshold-reaching times. NW differs because each example contains a full dynamic-programming table. After a useful rule is available, fixed-threshold exact match still requires many coordinated cell-level decisions to be correct. Short-output arithmetic tasks expose less of this residual exact-fitting burden, although prior arithmetic studies show that formatting and number-theoretic structure can still strongly affect transformer learning [2, 7].

4. Interpretation: Two Pressures in Data Scaling

The results suggest a two-pressure account. Additional data initially improves *structural coverage*: the model sees enough local configurations of the task rule to acquire a reusable partial solution. This accounts for the left side of the sweet spot, where validation and sometimes training threshold-reaching times improve as N grows. Figure 4 places the effect beside two short-output arithmetic controls under the same threshold-reaching analysis. The detailed experimental setup is in Appendix A.3.

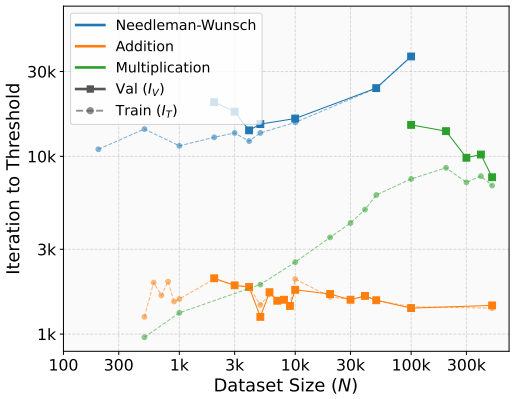


Figure 4: Task-scope controls at $\tau = 0.98$.

The three task families illustrate how the balance changes. In decimal addition, the output is short and the reusable carry rule is quickly exposed; once the model has the rule, train and validation threshold-reaching remain close, so additional data mostly supplies redundant coverage. In three-digit multiplication, the rule is harder because many digit interactions and carries must be coordinated, but the output is still short; over the evaluated range, more data continues to help or at least does not create the pronounced post-onset slowdown seen in NW. In NW matrix generation, each example contains a full dynamic-programming table. A partially learned recurrence can explain many cells and examples at once, reducing training threshold-reaching near the onset of weak validation competence. But strict full-matrix exact match later requires eliminating many residual cell-level errors, so the optimization bottleneck can shift from discovering the shared rule to completing exact fitting.

We call this latter burden *residual exact fitting*. The random-suffix result makes the account diagnostic. Useful data scaling is not a generic improvement in fitting any target sequence; the model improves more on the recurrence-governed NW component than on an arbitrary suffix. Thus $N_c(\tau)$ marks when a generalizing rule first becomes learnable, whereas $N^\star(\tau)$ marks the update-optimal balance between structural coverage and residual exact fitting. This differs from double-descent sample non-monotonicity, where more samples can worsen test performance near an effective-complexity threshold [12]: here final generalization remains attainable, and the non-monotonicity is in update time.

5. Conclusion and Limitations

We studied data scaling through the update time required to reach fixed exact-match thresholds. In NW matrix generation, the update-optimal dataset size is intermediate: beyond the onset of generalization, additional data can still slow validation threshold-reaching. Together with the random-suffix probe, this supports a two-pressure view in which data first accelerates reusable rule learning but later increases residual exact-fitting cost.

This claim is deliberately scoped. We do not suggest that more data is generally harmful, nor that the same sweet spot should appear in all algorithmic tasks. Our result is update-based, tied to a fixed optimization protocol, and most visible in a structured-output setting; broader task families and compute-scaled protocols remain important future work.

References

[1] Devansh Arpit, Stanislaw Jastrzebski, Nicolas Ballas, David Krueger, Emmanuel Bengio, Maxinder S. Kanwal, Tegan Maharaj, Asja Fischer, Aaron Courville, Yoshua Bengio, and Simon Lacoste-Julien. A closer look at memorization in deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, 2017.

[2] François Charton. Learning the greatest common divisor: Explaining transformer predictions. In *International Conference on Learning Representations*, 2024.

[3] Vitaly Feldman and Chiyuan Zhang. What neural networks memorize and why: Discovering the long tail via influence estimation. In *Advances in Neural Information Processing Systems*, 2020.

[4] Joel Hestness, Sharan Narang, Newsha Ardalani, Gregory Diamos, Heewoo Jun, Hassan Kianinejad, Md. Mostofa Ali Patwary, Yang Yang, and Yanqi Zhou. Deep learning scaling is predictable, empirically. *arXiv preprint arXiv:1712.00409*, 2017.

[5] Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*, 2022.

[6] Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. Scaling laws for neural language models. *arXiv preprint arXiv:2001.08361*, 2020.

[7] Nayoung Lee, Kartik Sreenivasan, Jason D. Lee, Kangwook Lee, and Dimitris Papailiopoulos. Teaching arithmetic to small transformers. *arXiv preprint arXiv:2307.03381*, 2023.

[8] Ziming Liu, Ouail Kitouni, Niklas Nolte, Eric J. Michaud, Max Tegmark, and Mike Williams. Towards understanding grokking: An effective theory of representation learning. In *Advances in Neural Information Processing Systems*, 2022.

[9] Ziming Liu, Eric J. Michaud, and Max Tegmark. Omnigrok: Grokking beyond algorithmic data. In *International Conference on Learning Representations*, 2023.

[10] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *International Conference on Learning Representations*, 2019.

[11] John X. Morris, Chawin Sitawarin, Chuan Guo, Narine Kokhlikyan, G. Edward Suh, Alexander M. Rush, Kamalika Chaudhuri, and Saeed Mahloujifar. How much do language models memorize? *arXiv preprint arXiv:2505.24832*, 2025.

[12] Preetum Nakkiran, Gal Kaplun, Yamini Bansal, Tristan Yang, Boaz Barak, and Ilya Sutskever. Deep double descent: Where bigger models and more data hurt. *arXiv preprint arXiv:1912.02292*, 2019.

[13] Neel Nanda, Lawrence Chan, Tom Lieberum, Jess Smith, and Jacob Steinhardt. Progress measures for grokking via mechanistic interpretability. In *International Conference on Learning Representations*, 2023.

[14] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3): 443–453, 1970.

[15] Alethea Power, Yuri Burda, Harri Edwards, Igor Babuschkin, and Vedant Misra. Grokking: Generalization beyond overfitting on small algorithmic datasets. *arXiv preprint arXiv:2201.02177*, 2022.

[16] Vikrant Varma, Rohin Shah, Zachary Kenton, János Krámár, and Ramana Kumar. Explaining grokking through circuit efficiency. *arXiv preprint arXiv:2309.02390*, 2023.

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

[18] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning requires rethinking generalization. In *International Conference on Learning Representations*, 2017.

[19] Xuekai Zhu, Yao Fu, Bowen Zhou, and Zhouhan Lin. Critical data size of language models from a grokking perspective. *arXiv preprint arXiv:2401.10463*, 2024.

AI Assistance Disclosure

The authors used a large language model for language polishing, restructuring suggestions, and editing support. The authors were responsible for all scientific content, experimental design, analysis, claims, and final manuscript decisions.

Appendix A. Expanded Experimental Details

A.1. Needleman–Wunsch task construction

We use fixed-length Needleman–Wunsch matrix prediction as a controlled structured-output task. Each input consists of two sequences $x = (x_1, \ldots, x_m)$ and $y = (y_1, \ldots, y_m)$ over a finite alphabet $\mathcal{A}$. The target is the full dynamic-programming score matrix $M \in \mathbb{Z}^{(m+1)\times(m+1)}$. In the main experiments, $m = 5$ and $|\mathcal{A}| = 4$.

The score matrix is initialized by

$$M_{0,0} = 0, \qquad M_{i,0} = i \cdot g, \qquad M_{0,j} = j \cdot g,$$

where g is the gap penalty. For $i, j \geq 1$, the recurrence is

$$M_{i,j} = \max\{M_{i-1,j-1} + s(x_i, y_j),\ M_{i-1,j} + g,\ M_{i,j-1} + g\}.$$

We use $s(a, b) = 5$ for a match, $s(a, b) = -4$ for a mismatch, and $g = -5$ throughout the main experiments.

The model predicts the score matrix as a sequence of discrete output tokens. Unless otherwise stated, we evaluate exact full-matrix accuracy: an example is counted as correct only when all predicted matrix entries match the Needleman–Wunsch target matrix. Cell-level accuracy is used only for diagnostic analyses and is not the main convergence metric.

A.2. Dataset sweep and model.

For $L = 5$, the finite task universe contains $|\Sigma|^{2L} = 4^{10} = 1{,}048{,}576$ input pairs. We sample training sets of size

$$N \in \{200, 500, 1\text{k}, 2\text{k}, 3\text{k}, 4\text{k}, 5\text{k}, 10\text{k}, 50\text{k}, 100\text{k}\}$$

and evaluate on held-out examples from the same universe. For each dataset size N, we sample N training examples from the finite task universe and evaluate on a held-out validation set. Unless otherwise stated, the validation set is held fixed across runs for the same task configuration.

We train decoder-only Transformers [17]. A depth-D model has D layers, D attention heads, embedding dimension $64D$, and feed-forward width $256D$; the main sweep uses $D \in \{3, 4, 5, 6\}$. We use AdamW [10], cosine learning-rate decay from 10^{-3} to 10^{-4}, weight decay 0.1, effective batch size 160, and at most $50{,}000$ optimizer updates. Each configuration is run with five random seeds unless otherwise stated.

A.3. Addition and multiplication baselines

We use decimal addition and multiplication as short-output algorithmic baselines. These tasks test whether exact-match thresholding alone produces an interior dataset-size optimum. The main claims of the paper concern Needleman–Wunsch matrix generation; the arithmetic tasks serve as contrast controls.

For d-digit addition, each input is a pair of nonnegative decimal integers (a, b), where $a, b \in \{0, \ldots, 10^d - 1\}$, and the target is the exact decimal representation of $a+b$. For d-digit multiplication, the input is again a pair (a, b), with $a, b \in \{0, \ldots, 10^d - 1\}$, and the target is the exact decimal representation of $a \cdot b$. The multiplication baseline in the main text uses $d = 3$; the addition control uses $d = 10$.

Inputs are blank-padded to length d. Targets are first written in canonical decimal form without leading zeros and then blank-padded to the maximum target length for batching and autoregressive prediction. Exact-match accuracy is computed on the full padded target sequence.

For arithmetic baselines, training examples are sampled from the finite input universe and validation examples are held out from the same task family. The multiplication sweep uses

$$N \in \{500, 1\text{k}, 5\text{k}, 10\text{k}, 20\text{k}, 30\text{k}, 40\text{k}, 50\text{k}, 100\text{k}, 200\text{k}, 300\text{k}, 400\text{k}, 500\text{k}\}.$$

The addition sweep uses

$$N \in \{500, 600, 700, 800, 900, 1\text{k}, 5\text{k}, 10\text{k}, 20\text{k}, 30\text{k}, 40\text{k}, 50\text{k}, 100\text{k}, 200\text{k}, 300\text{k}, 400\text{k}, 500\text{k}\}.$$

All optimizer, model-size, threshold-reaching, and censoring conventions match the NW experiments unless otherwise stated. Figure 4 includes the three tasks in one grid.

A.4. Model, optimizer, and compute details

For a model with depth D, we use a decoder-only Transformer with D layers, D attention heads, embedding dimension $d_{\text{emb}} = 64D$, and feed-forward dimension $d_{\text{ff}} = 4d_{\text{emb}}$. All models are trained from random initialization. Unless otherwise stated, we use AdamW with learning rate decayed from 10^{-3} to 10^{-4} by cosine scheduling, $\beta_1 = 0.9$, $\beta_2 = 0.99$, and weight decay 0.1.

We use batch size 8 with gradient accumulation over 20 steps, corresponding to an effective batch size of 160. Each run is trained for at most $50{,}000$ optimizer updates. Train and validation metrics are evaluated every 100 optimizer updates. Unless otherwise stated, convergence time is measured in optimizer updates rather than epochs. Unless otherwise stated, each task and dataset-size configuration is run with five random seeds; shaded regions report one standard deviation across seeds.

If a run does not reach the threshold within $50{,}000$ updates, we treat it as censored and mark it separately in the corresponding plot. All experiments were run on NVIDIA RTX 3090 GPUs with 24 GB memory. A single NW run at $L = 5$ and depth $D = 6$ takes approximately 10 GPU-hours for 50,000 optimizer updates. The full set of reported experiments used approximately 2,600 GPU-hours, including the main dataset-size sweep, depth sweep, sequence-length robustness experiments, and random-suffix probes. Preliminary or failed exploratory runs required approximately 1,077 additional GPU-hours.

Appendix B. Robustness Across Depth and Normalization

We test whether the main findings are specific to the representative $D = 6$, $L = 5$ setting. The main text reports the multi-threshold trajectories for $D \in \{3, 4, 5, 6\}$ at $L = 5$. The qualitative pattern is preserved across depths: validation threshold-reaching time is non-monotonic in N, and in the partial-generalization band, training threshold-reaching time can decrease as N grows. The location of the update-optimal dataset size shifts modestly with depth, consistent with deeper models having a slightly larger structural-coverage threshold. Here we include the corresponding seed-variability and epoch-normalized views.

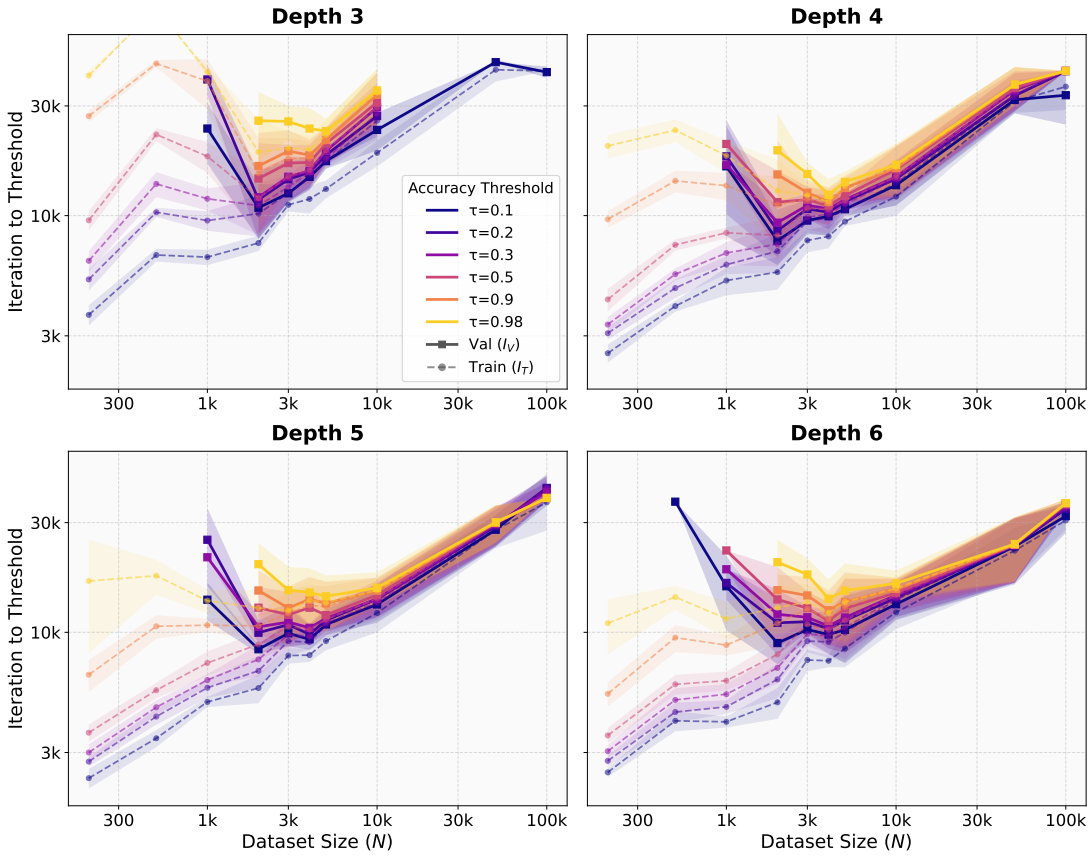


Figure 5: Seed variability for the $L = 5$ NW multi-threshold sweeps. Shaded regions denote one standard deviation across random seeds. Although exact threshold-reaching times vary, the qualitative pattern remains stable: validation threshold-reaching times are non-monotonic in N, and training threshold-reaching times improve near the onset of weak validation competence.

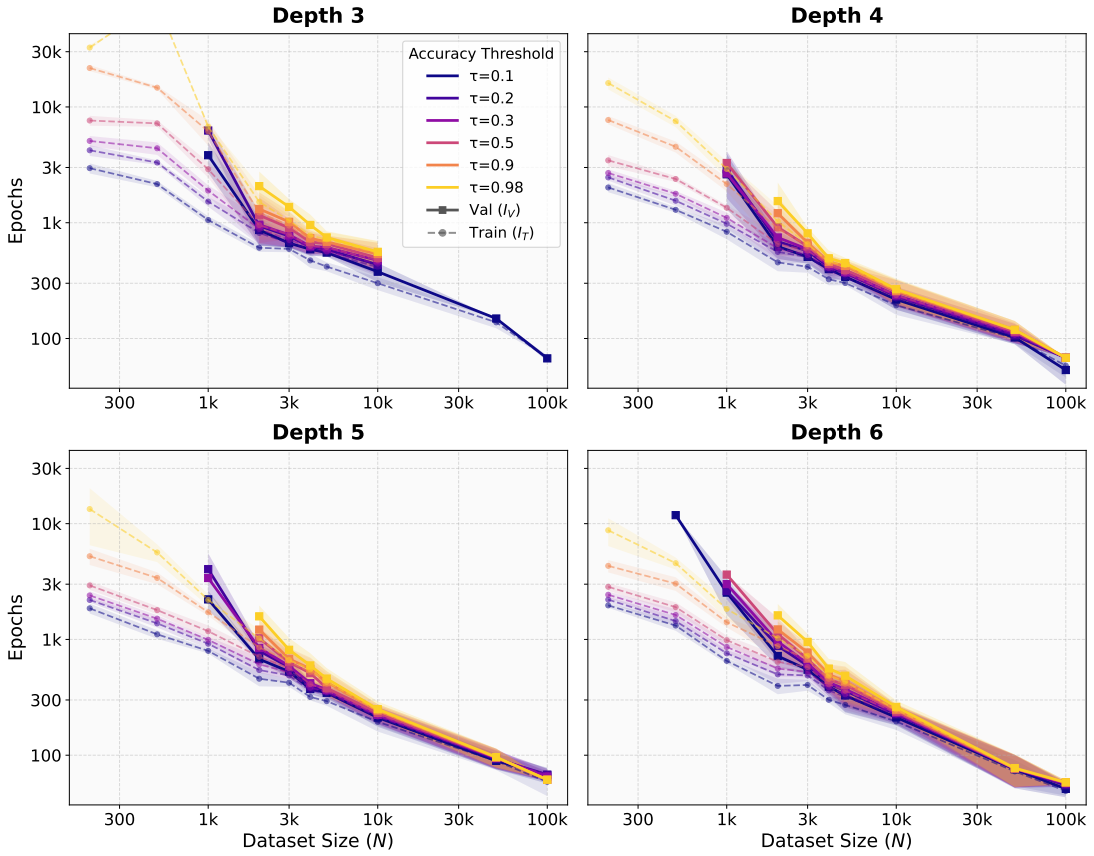


Figure 6: Epoch-normalized convergence for the $L = 5$ NW depth sweep. The same threshold-reaching iterations are measured in epochs rather than optimizer updates. Epoch normalization emphasizes data exposure and recovers the more conventional view in which larger datasets require fewer passes over the data. This confirms that the sweet spot in the main text is specifically an update-based convergence phenomenon.

Appendix C. Robustness Across Sequence Length

To test whether the timing pattern is specific to the $L = 5$ Needleman–Wunsch task, we repeat the same analysis at sequence length $L = 4$. The full task universe is smaller, but the qualitative structure is similar: validation convergence is non-monotonic in N, and the decrease of $I_T(\tau; N)$ appears around the onset of weak validation thresholds.

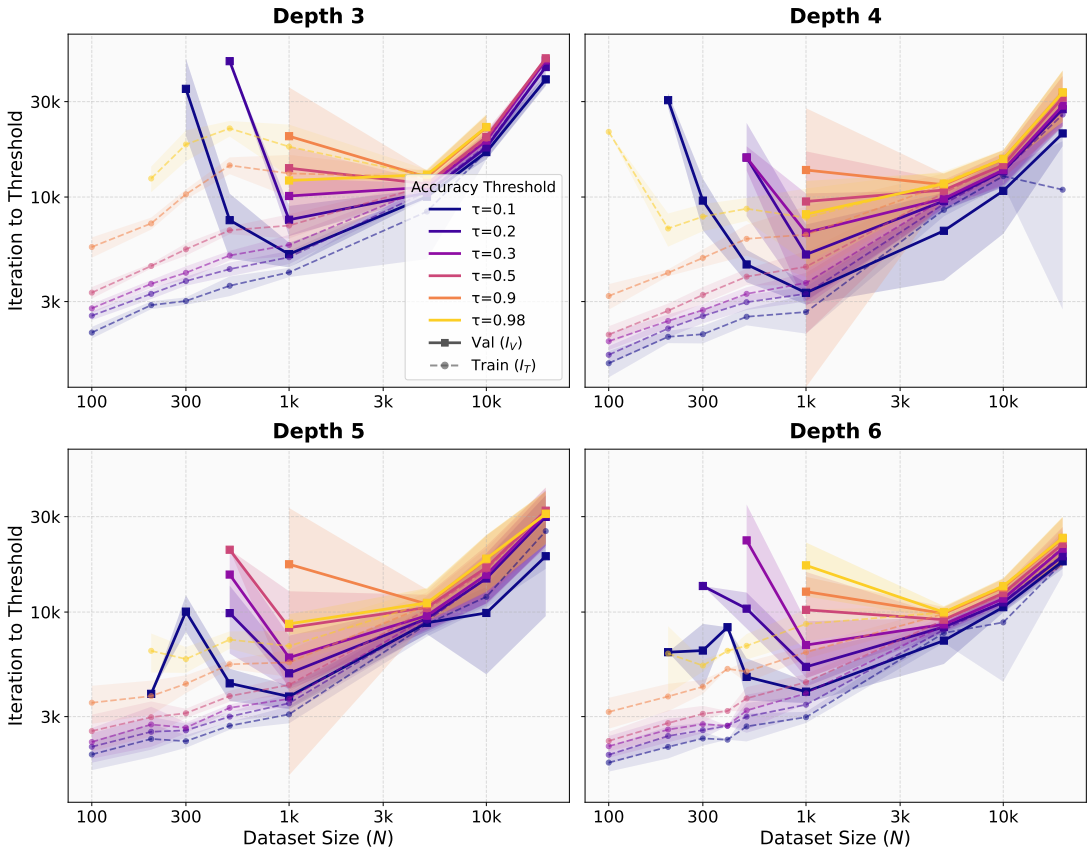


Figure 7: Seed variability for the $L = 4$ NW sweeps. Shaded regions denote one standard deviation across random seeds. Exact threshold-reaching times vary, but the intermediate-data regime again shows overlap between weak validation competence and faster training threshold-reaching times.

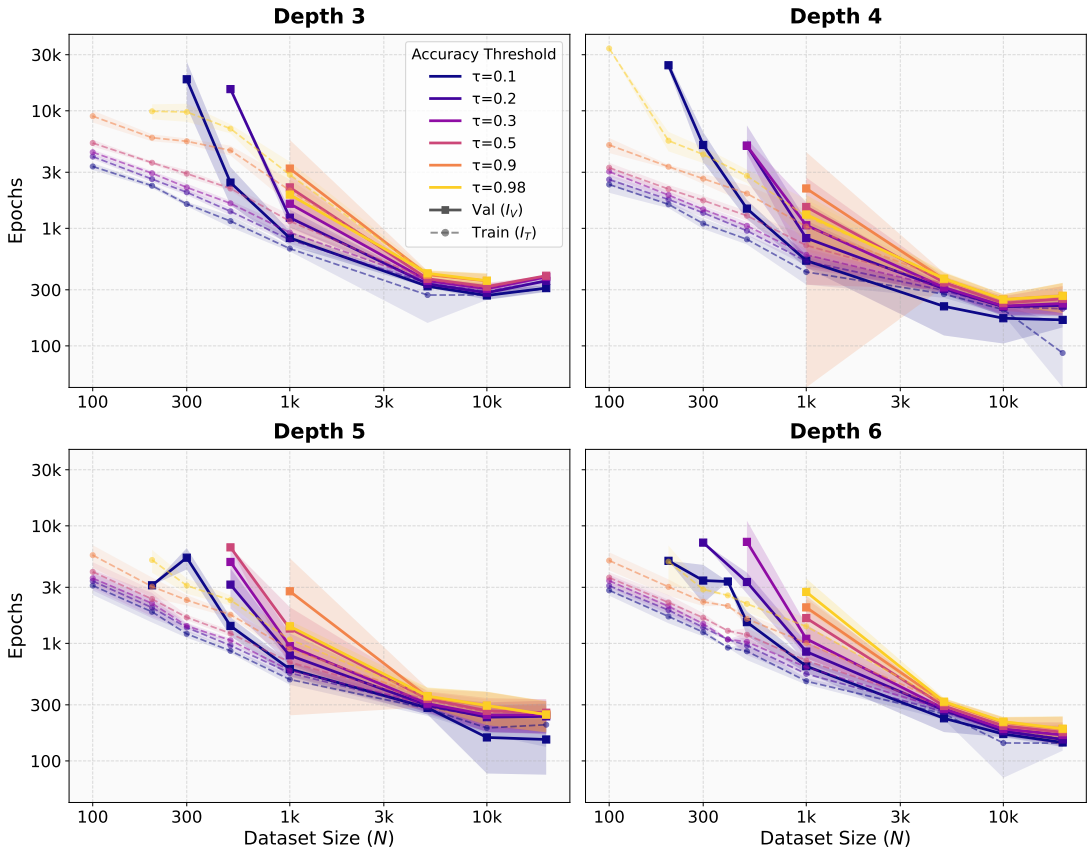


Figure 8: Epoch-normalized convergence for the $L = 4$ NW task. As in the $L = 5$ experiments, epoch normalization emphasizes data exposure rather than parameter-update efficiency and recovers a conventional monotone-data view. This reinforces that the sweet spot studied in the main text is update-based.

Appendix D. Expanded Random-Suffix Experiments

The main text uses the random-suffix probe as the central mechanism-oriented evidence. Here we provide the representative component trajectories and the full dataset-size sweep. Each NW target is augmented with a four-bit random binary suffix. We track training accuracy on the structured NW matrix A_T, validation accuracy on the NW matrix A_V, and training accuracy on the random suffix A_R. The NW matrix is rule-generated, while the suffix is example-specific and cannot be inferred from the input.

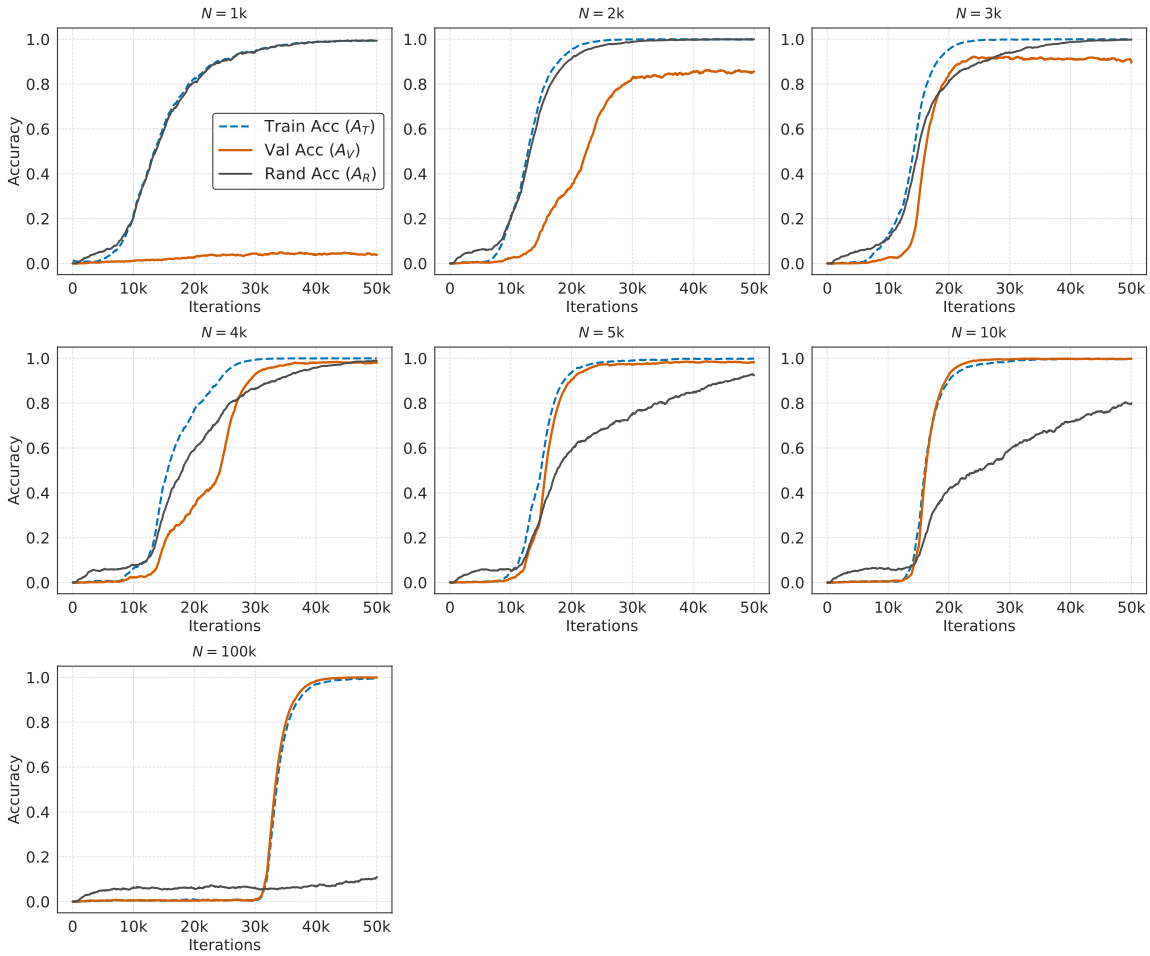


Figure 9: Full random-suffix component trajectories across dataset sizes. At small N, A_T and A_R rise together while A_V remains near zero. At larger N, the structured NW component reaches high training accuracy before the much shorter random suffix.

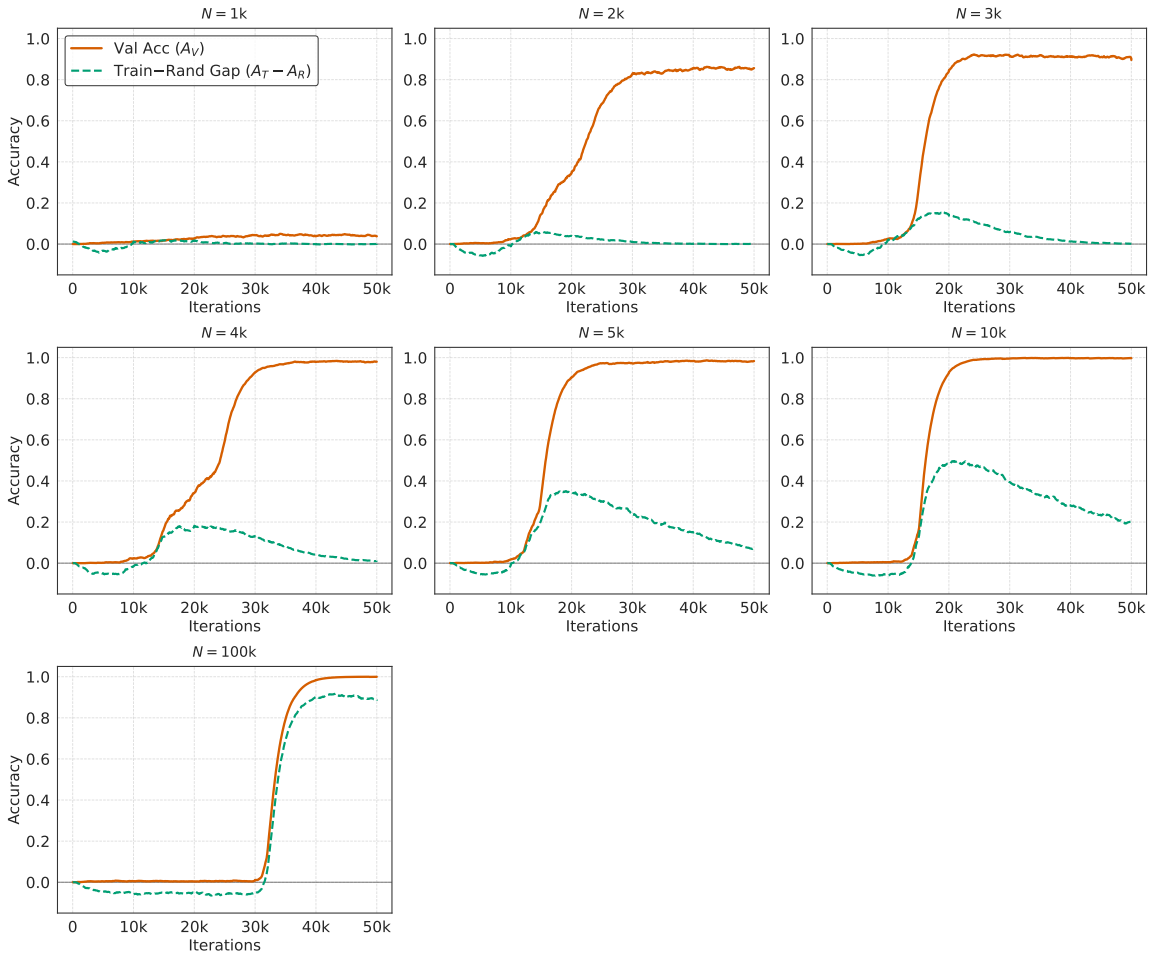


Figure 10: Full dataset-size sweep for the train-random gap. For each N, we compare NW validation accuracy A_V with $G = A_T - A_R$, where A_T is training accuracy on the NW matrix and A_R is training accuracy on the random suffix. At small N, the gap remains small and validation remains low; at larger N, the gap rises together with validation accuracy.

Appendix E. Broader Impact

This work is foundational and studies synthetic algorithmic datasets rather than deployed systems. Its main potential benefit is a clearer understanding of memorization–generalization dynamics under controlled conditions. We do not release pretrained models, scraped datasets, personal data, or user-facing systems. We therefore do not anticipate direct societal harms, although the general study of memorization may inform future work on privacy, robustness, and data efficiency.