Beyond What Seems Necessary: Hidden Gains from Scaling Training-Time Reasoning Length under Outcome Supervision

Yihao Xue **Allan Zhang** **Jianhao Huang** **Amit Sahai** **Baharan Mirzasoleiman**
University of California, Los Angeles

Abstract

Training LLMs to think and reason for longer has become a key ingredient in building state-of-the-art models that can solve complex problems previously out of reach. Recent efforts pursue this in different ways, such as RL fine-tuning to elicit long CoT or scaling latent reasoning through architectural recurrence. This makes reasoning length an important scaling knob. In this work, we identify a novel phenomenon (both theoretically and experimentally): under outcome-only supervision, out-of-distribution (OOD) performance can continue improving as training-time reasoning length (e.g., the token budget in RL, or the loop count in looped Transformers) increases, even after in-distribution (ID) performance has saturated. This suggests that robustness may require a larger budget than ID validation alone would indicate. We provide theoretical explanations via two mechanisms: (i) self-iteration can induce a stronger inductive bias in the hypothesis class, reshaping ID-optimal solutions in ways that improve OOD generalization; and (ii) when shortcut solutions that work for ID samples but not for OOD samples persist in the hypothesis class, regularization can reduce the learned solution's reliance on these shortcuts as the number of self-iterations increases. We complement the theory with empirical evidence from two realizations of scaling training-time reasoning length: increasing the number of loops in looped Transformers on a synthetic task, and increasing token budgets during RL fine-tuning of LLMs on mathematical reasoning, coding, and cross-representation tasks.

1 Introduction

Large language models (LLMs) are increasingly trained to *think longer*: they generate multiple intermediate steps before outputting the final answer. A prominent example is chain-of-thought (CoT) reasoning, where the model attempts to solve a problem step by step. In practice, such behavior is often induced through reinforcement learning (RL) fine-tuning [35, 36, 31, 38, 48], where the model is trained to generate substantial "thinking" tokens [13] during rollouts. Another line of research studies *latent* reasoning carried out in continuous embedding space, for example via architectural recurrence such as looped Transformers [12, 34] or continuous CoT [14]. Together, these developments make *reasoning length*, whether explicit (tokens) or latent (loops), a central scaling knob.

Despite growing efforts to understand how CoT or looping affects a model's expressivity [46, 23, 12], and work on inference-time scaling of reasoning [39, 45], the effects of scaling this knob *during training* remain underexplored. This question is particularly important under outcome-only supervision [24, 20]—for example, in RL the reward is often assigned solely based on the final answer, and looped Transformers similarly provide no explicit guidance about how intermediate computation should proceed. In such settings, intermediate behavior is learned implicitly, making it crucial to understand what kinds of computation emerge through training. Essentially, scaling training-time reasoning length amounts to allowing more *self-iteration* during training without specifying how those iterations should be carried out. We thus lack a learning-theoretic account of:

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

How does allowing the model to self-iterate more times during training (e.g., using a larger token budget during RL, or training looped Transformers with more loops) change the learned solution? Can increasing reasoning length lead models to solve the same task via a qualitatively different mechanism?

Understanding this question matters in realistic settings where training and validation data are limited and distribution shift is expected at deployment. In such cases, in-distribution (ID) validation performance can be misleading: models can be "right for the wrong reasons," and an apparent ID plateau can hide further gains in robust generalization. Indeed, we find that out-of-distribution (OOD) performance can continue improving with increased training-time reasoning length even after ID performance has saturated.

We provide theoretical explanations for these hidden OOD gains from a learning-theoretic perspective that views increasing training-time reasoning length as increasing the number of self-iterations, and analyzes how this changes the effective hypothesis class explored by the training algorithm. This shift can alter *which* ID-optimal solution is selected, even when many solutions achieve the same ID error. We formalize two mechanisms: (i) self-iteration can induce a stronger inductive bias, reshaping the set of ID-optimal solutions in ways that improve OOD generalization; and (ii) when shortcut solutions persist in the hypothesis class, regularization can reduce the learned solution's reliance on these shortcuts as the number of self-iterations increases.

We substantiate this phenomenon empirically through two realizations of training-time reasoning length. First, we scale *latent* reasoning in looped Transformers by increasing the loop count. Training on synthetic algorithmic tasks, we observe that ID accuracy saturates at small loop counts while OOD accuracy continues to improve gradually. Second, we scale *explicit* CoT reasoning by increasing the token budget during RL fine-tuning across math, code-to-math, and cross-representation (natural language to graph) shifts across `Qwen2.5-1.5B`, `Qwen3-4B`, `Qwen3-8B`, and `Olmo3-7B`; OOD accuracy keeps improving well past ID saturation, mediated by realized reasoning length with clear compute–performance trade-offs.

Overall, we provide theoretical justification for a practical recommendation: when training is outcome-supervised and distribution shift is expected after deployment, it can be beneficial to train with longer reasoning lengths (larger CoT token budgets in RL or more loops in looped Transformers) beyond what appears sufficient to saturate validation performance. In-distribution performance can be misleading, masking substantial OOD gains from further scaling.

2 Related Work

RL post-training for reasoning. A key recent development is the use of variants of RL algorithms [35, 36, 31, 38, 48] in post-training to produce models that generate longer CoT deliberation [44, 22] and thereby solve more complex problems. This paradigm is emphasized in prominent reasoning-model reports [18, 38, 13, 28]. In RL fine-tuning, training rollouts are typically capped by a maximum generation length, so the model learns to solve problems within a fixed token budget. The choice of this budget therefore becomes an important scaling knob: training with a larger budget typically yields more capable models [21, 3, 37].

Latent reasoning via structural recurrence. In parallel with efforts to elicit longer explicit CoT, structural recurrence offers another route to scaling reasoning via latent "thought". Looped Transformers apply the same Transformer backbone repeatedly with shared weights—an idea that can be traced back to [6]—and have recently attracted renewed interest due to their theoretical expressivity, including the ability to simulate arbitrary Turing machines [5, 12], express iterative algorithms [47, 8–10, 46], and improved length generalization [7]. Saunshi et al. [34] shows that looped Transformers can exhibit strong reasoning performance and highlights empirical and theoretical connections to CoT, suggesting that looping can support a form of "latent thought" reasoning. This perspective is also closely related to the idea of continuous CoT [14].

We connect these threads by viewing them as different realizations of *self-iteration*. We offer the first learning-theoretic perspective on how the induced hypothesis class evolves as we scale the number of self-iterations, shaping what is learned on one distribution and how it generalizes OOD.

3 Training-Time Self-Iteration Can Yield OOD Gains Without ID Gains

In this section, we develop theory to shed light on the phenomenon that *OOD performance can continue improving as we increase training-time reasoning length, even when there is no visible ID improvement*. Increasing reasoning length—whether explicit (e.g., via CoT tokens) or latent (e.g., via the number of loops in looped Transformers)—can be viewed as different realizations of increasing self-iteration. Our key observation is that permitting more iterations before emitting the output changes how the model can produce its final answer. From a learning-theoretic perspective, this corresponds to a change in the *hypothesis class* over which the training algorithm searches. A natural implication is that when the ID optimum can be achieved in multiple ways, changing the hypothesis class can steer learning toward a different ID-optimal solution that behaves differently under distribution shift. We will illustrate two mechanisms through which this effect can occur.

Notation. We study a learning problem in which the goal is to learn a model $f : \mathcal{X} \to \mathcal{Y}$ from a hypothesis class $\mathcal{F}_k$, with ground-truth target function $z : \mathcal{X} \to \mathcal{Y}$. Here k denotes the number of training-time iterations (e.g., the token budget for CoT or the number of loops in a looped transformer), and $\mathcal{F}_k$ denotes the resulting hypothesis class. We consider a population-level setting in which the model is trained with respect to a distribution $\mathcal{P}$ over $\mathcal{X}$ and is subsequently evaluated on a different distribution $\mathcal{Q}$.

Understanding our phenomenon therefore requires analyzing how $\mathcal{F}_k$ evolves as k increases, and how this evolution interacts with the learning algorithm and the data distribution. Our theoretical settings below are deliberately stylized: rather than modeling full LLM RL optimization dynamics, they rigorously show that the phenomenon arises even at the population optimum and isolate two mechanisms—hypothesis-class reshaping and shortcut suppression under regularization—mirroring practical reasoning pipelines.

3.1 Self-Iteration Can Strengthen Inductive Bias in the Hypothesis Class

It is tempting to view self-iteration mainly as increasing expressivity, e.g., by enabling a broader class of serial computations [27, 23]. However, expressivity alone does not explain why self-iteration yields solutions with better OOD performance but the same ID performance. We find that self-iteration instead induces a favorable inductive bias in the hypothesis class, steering learning toward solutions better aligned with the target. In fact, when the base class $\mathcal{F}_1$ is already sufficiently expressive, as is common for large neural networks, self-iteration need not expand capacity: a k-fold self-iterate must admit a k-th compositional root. We next present two examples where self-iteration either shrinks or shifts the hypothesis space in ways that improve OOD generalization, while remaining optimal in-distribution.

3.1.1 Example: Self-Iterating the Full Hypothesis Class on a Cycle Task

We illustrate the effect of self-iteration through a deliberately extreme toy example. The key "magic" is that the base hypothesis class is maximally expressive: $\mathcal{F}_1$ contains *every* possible function and is therefore completely unconstrained, much like an overparameterized neural network that can fit the data in many incompatible ways. Nevertheless, when we restrict the learner to k-fold self-compositions of functions from this same class, the resulting hypothesis class $\mathcal{F}_k$ acquires a strong implicit structure—turning an unconstrained class into one that selects a uniquely correct solution and generalizes perfectly to the OOD data.

A cycle task. Fix an integer $n \geq 3$ and define the domain $\mathbb{S} \coloneqq \{0, 1, \ldots, n-1\}$. Fix any held-out point $q \in \mathbb{S}$, and let $\mathbb{Q} \coloneqq \{q\}$ and $\mathbb{P} \coloneqq \mathbb{S} \setminus \mathbb{Q}$. Let $\mathcal{P}$ be the uniform distribution over $\mathbb{P}$ and $\mathcal{Q}$ be the uniform distribution over $\mathbb{Q}$ (a point mass at q). Let the ground-truth target $z : \mathbb{S} \to \mathbb{S}$ be some cyclic permutation on $\mathbb{S}$.

Given a function $f : \mathbb{S} \to \mathbb{S}$ and a distribution $\mathcal{D}$ over $\mathbb{S}$, define the 0-1 loss $L_{\mathcal{D}}(f) \coloneqq \mathbb{E}_{x \sim \mathcal{D}} \mathbf{1}_{\{f(x) \neq z(x)\}}$. Given a hypothesis class $\mathcal{H}$, we minimize the ID population loss $L_{\mathcal{P}}$, namely $\arg\min_{f \in \mathcal{H}} L_{\mathcal{P}}(f)$. We then evaluate the resulting minimizer on $\mathcal{Q}$ as a measure of OOD performance. We take the base hypothesis class $\mathcal{F}$ to be the *entire* class of endomorphisms on $\mathbb{S}$, and define $\mathcal{F}_k$ to be the class of k-fold self-compositions of functions in $\mathcal{F}$: $\mathcal{F} \coloneqq \{ f : \mathbb{S} \to \mathbb{S} \}, \mathcal{F}_k \coloneqq \mathcal{F}^{\circ k} = \{ f^{\circ k} : f \in \mathcal{F} \}$. Here $f^{\circ k}$ denotes the k-fold self-composition $f \circ f \circ \cdots \circ f$.

OOD failure at $k = 1$. The ID minimizer can fit $\mathcal{P}$ perfectly while behaving arbitrarily on $\mathcal{Q}$.

Proposition 3.1. *There exists* $\hat{f}_1 \in \arg\min_{f\in\mathcal{F}_1} L_{\mathcal{P}}(f)$ *s.t.* $L_{\mathcal{P}}(\hat{f}_1)=0$ *and* $L_{\mathcal{Q}}(\hat{f}_1)=1$.

Proof. $\mathcal{F}_1$ contains all functions $\mathbb{S}\to\mathbb{S}$. One can interpolate $\mathbb{P}$ and set the value at q arbitrarily. □

Most sufficiently large k enforce a unique OOD-optimal ID minimizer. We next show that, despite $\mathcal{F}$ being completely unconstrained, for a large set of k the iterated class $\mathcal{F}_k$ induces a much stronger inductive bias: the ID minimizer becomes *unique* and equals the ground-truth z, and hence is OOD-optimal. We quantify "most" via (lower) natural density: for a set $A \subseteq \mathbb{N}$, define $\mathrm{dens}(A) \coloneqq \liminf_{N\to\infty} \frac{|A\cap\{1,2,\ldots,N\}|}{N}$. Intuitively, it is the asymptotic fraction of integers in A.

Theorem 3.2. *For any $\varepsilon > 0$, there exists a prime n and a set of "good" values of k*

$$\mathcal{K} \coloneqq \{\, k \geq n-1 :\ \gcd(k,n)=1 \text{ and } \gcd(k,n-1)>1 \,\}$$

such that the following holds. (i) $\mathcal{K}$ has high density: $\mathrm{dens}(\mathcal{K}) \geq 1-\varepsilon$. *(ii) For every choice of held-out point $q \in \mathbb{S}$, every cyclic-permutation target z, and every $k \in \mathcal{K}$, the ID minimizer over $\mathcal{F}_k$ is unique and equals the ground truth:* $\arg\min_{f\in\mathcal{F}_k} L_{\mathcal{P}}(f) = \{z\}$. *In particular, the unique minimizer $\hat{f} = z$ achieves zero OOD error, i.e., $L_{\mathcal{Q}}(\hat{f}) = 0$.*

Proof road map. The proof of Thm. 3.2 is deferred to App. A. We first establish a structural uniqueness criterion for good interpolants at fixed n and k (Thm. A.5), using functional-graph properties of large iterates and a characterization of interpolants realizable as k-fold iterates. Intuitively, $\gcd(k,n)=1$ ensures that the target z lies in $\mathcal{F}_k$, while $\gcd(k,n-1)>1$ eliminates the only spurious interpolant, forcing uniqueness. We then use density calculations and Dirichlet's theorem on primes in arithmetic progressions to pick a prime n for which the gcd conditions hold for a set of exponents $k \geq n-1$ with density at least $1-\varepsilon$.

As an example, for $\epsilon = 0.25$, we can take $n = 2311$. Then Thm 3.2 implies that for at least a 0.75 fraction of integers $k \geq 2310$, learning over $\mathcal{F}_k$ achieves zero error both ID and OOD. In contrast, when $k=1$, Prop. 3.1 shows that there exists an ID minimizer with maximal OOD error $L_{\mathcal{Q}}=1$.

3.1.2 Example: In-context coefficient prediction with a linear transformer

We present an example of a linear transformer trained to perform in-context coefficient prediction [17] for linear regression, trained with multiple self-iterations before producing its final prediction. We show how increasing the number of self-iterations shifts the induced hypothesis class toward a generic (in-context) gradient descent solver, yielding OOD improvements even when the ID loss remains zero. We note that our notation is tailored to mimic autoregressive generation, but in this continuous token-space setting it can be equivalently rewritten in a form more reminiscent of looped transformers [9].

Input sequence structure. Each input sequence encodes a linear regression problem identified by a tuple $(\boldsymbol{\Sigma}, \boldsymbol{\beta})$, where $\boldsymbol{x} \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ and $y = \boldsymbol{x}^\top \boldsymbol{\beta}$. Given $(\boldsymbol{\Sigma}, \boldsymbol{\beta})$, we sample n i.i.d. pairs $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ and assemble them into an input matrix $\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \in \mathbb{R}^{d_e\times(n+1)}$ with $d_e = 2d+2$ as follows. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d\times n}$ and $\boldsymbol{y} = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$. Define the following $d_e \times (n+1)$ matrix:

$$\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \coloneqq S(\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n) \coloneqq \begin{bmatrix} n^{-1/2}\boldsymbol{X}^\top & n^{-1/2}\boldsymbol{y} & \mathbf{0}_{n\times d} & \mathbf{0}_{n\times 1} \\ \mathbf{0}_{1\times d} & 0 & \mathbf{0}_{1\times d} & 1 \end{bmatrix}^\top. \tag{1}$$

The last column is a "query" token with a constant 1 in the final coordinate. Note that $\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}}$ is a random variable conditioned on $(\boldsymbol{\Sigma}, \boldsymbol{\beta})$, while $S(\cdot)$ is a deterministic function of the sampled dataset. The goal is to take $\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}}$ as the prompt and infer $\boldsymbol{\beta}$ from the in-context examples.

Linear transformer with reduced weights. We consider a linear self-attention module with a residual connection and reduced weights adapted from [17]. For any input matrix $\boldsymbol{M} \in \mathbb{R}^{d_e\times\ell}$, define $T(\boldsymbol{M} \mid \boldsymbol{V}, \boldsymbol{W}) = \boldsymbol{M} + \tilde{\boldsymbol{V}}\,\boldsymbol{M}\boldsymbol{M}^\top\tilde{\boldsymbol{W}}\,\boldsymbol{M}$, where the only trainable blocks are $\boldsymbol{V}, \boldsymbol{W} \in \mathbb{R}^{d\times d}$ with

$$\tilde{\boldsymbol{V}} = \begin{bmatrix} \mathbf{0}_{(d+1)\times d} & \mathbf{0}_{(d+1)\times(d+2)} \\ \boldsymbol{V} & \mathbf{0}_{d\times(d+2)} \\ \mathbf{0}_{1\times d} & \mathbf{0}_{1\times(d+2)} \end{bmatrix}, \quad \tilde{\boldsymbol{W}} = \begin{bmatrix} \mathbf{0}_{d\times(d+1)} & \boldsymbol{W} & \mathbf{0}_{d\times 1} \\ \mathbf{0}_{1\times(d+1)} & \mathbf{0}_{1\times d} & -1 \\ \mathbf{0}_{(d+1)\times(d+1)} & \mathbf{0}_{(d+1)\times d} & \mathbf{0}_{(d+1)\times 1} \end{bmatrix}.$$

We write the last token (last column) as: $T(\boldsymbol{M} \mid \boldsymbol{V}, \boldsymbol{W})_{[:,\ell]} = \boldsymbol{M}_{[:,\ell]} + \tilde{\boldsymbol{V}}\,\boldsymbol{M}\boldsymbol{M}^\top\tilde{\boldsymbol{W}}\,\boldsymbol{M}_{[:,\ell]}$.

Autoregressive generation and extraction. We define a one-step autoregressive extension by appending the updated last token. Given $(\boldsymbol{V}, \boldsymbol{W})$, for any $\boldsymbol{M} \in \mathbb{R}^{d_e \times \ell}$: $\mathrm{Auto}(\boldsymbol{M} \mid \boldsymbol{V}, \boldsymbol{W}) = \begin{bmatrix}\boldsymbol{M} & T(\boldsymbol{M} \mid \boldsymbol{V}, \boldsymbol{W})_{[:,\ell]}\end{bmatrix}$. We define an extraction map $\mathrm{Extract}(\cdot)$, which, conceptually, abstracts how a model's final answer is obtained from its generated sequence, enabling direct comparison with the reference answer. In our case, we specialize it to $\mathrm{Extract}(\boldsymbol{M}) = \boldsymbol{M}_{[d+2:2d+1,\,\ell]} \in \mathbb{R}^d$, i.e., we extract rows $d+2, \ldots, 2d+1$ of the last token. Given a generation budget k, the induced predictor is

$$f_k(\cdot \mid \boldsymbol{V}, \boldsymbol{W}) \;=\; \mathrm{Extract} \circ \mathrm{Auto}^{\circ k}(\cdot \mid \boldsymbol{V}, \boldsymbol{W}), \tag{2}$$

whose output lies in the same space as $\boldsymbol{\beta}$, the target coefficients we aim to predict.

In-context GD-implementing parameters. Given a dataset $\mathcal{C} = \{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$, consider the squared loss $R(\boldsymbol{w}; \mathcal{C}) := \frac{1}{n}\sum_{i=1}^n (\boldsymbol{x}_i^\top \boldsymbol{w} - y_i)^2$. For a step size $\eta > 0$, one-step gradient descent (GD) on R is $\boldsymbol{w}_{t+1} = \boldsymbol{w}_t - \eta \nabla R(\boldsymbol{w}_t; \mathcal{C})$. On the other hand, given $\mathcal{C}$, we form the input sequence $S(\mathcal{C})$ as in equation 1 and run the autoregressive model. For parameters $(\boldsymbol{V}, \boldsymbol{W})$, define the extracted prediction at generation step t by $\hat{\boldsymbol{w}}_t(\mathcal{C}; \boldsymbol{V}, \boldsymbol{W}) := \mathrm{Extract}\big(\mathrm{Auto}^{\circ t}\big(S(\mathcal{C}) \mid \boldsymbol{V}, \boldsymbol{W}\big)\big)$. We say that $(\boldsymbol{V}, \boldsymbol{W})$ *implement GD updates* if there exists a step size $\eta > 0$ such that, for every dataset $\mathcal{C}$ and every $t \geq 0$,

$$\hat{\boldsymbol{w}}_{t+1}(\mathcal{C}; \boldsymbol{V}, \boldsymbol{W}) = \hat{\boldsymbol{w}}_t(\mathcal{C}; \boldsymbol{V}, \boldsymbol{W}) - \eta \nabla R(\hat{\boldsymbol{w}}_t(\mathcal{C}; \boldsymbol{V}, \boldsymbol{W}); \mathcal{C})\,.$$

Let Θ_{GD} denote the set of such *generic* GD-implementing $(\boldsymbol{V}, \boldsymbol{W})$ pairs (i.e., the above equality holds uniformly over all datasets). App. B.2 shows that this is equivalent to $\Theta_{\mathrm{GD}} = \Big\{\; (-\eta \boldsymbol{I}_d,\; \boldsymbol{I}_d) \mid \eta > 0 \;\Big\}$.

Infinite-context limit and loss. For each n, consider the model's prediction $f_k(\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \mid \boldsymbol{V}, \boldsymbol{W})$ on the input sequence $\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}}$. We study the infinite-context limit $n \to \infty$ for any finite k. Under our data model, the empirical moments in $\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \boldsymbol{S}^{(n)\top}_{\boldsymbol{\Sigma},\boldsymbol{\beta}}$ converge to their population counterparts, implying that $f_k(\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \mid \boldsymbol{V}, \boldsymbol{W})$ converges almost surely to a deterministic limit conditioned on $(\boldsymbol{\Sigma}, \boldsymbol{\beta})$ (see App. B.1). For a distribution $\mathcal{D}$ over $(\boldsymbol{\Sigma}, \boldsymbol{\beta})$, we then define the infinite-context population loss as

$$L_{\mathcal{D}}(f_k) \;:=\; \mathbb{E}_{(\boldsymbol{\Sigma},\boldsymbol{\beta})\sim\mathcal{D}} \,\Big\| \lim_{n\to\infty} f_k(\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \mid \boldsymbol{V}, \boldsymbol{W}) - \boldsymbol{\beta}\Big\|^2.$$

Function class. For each generation budget k, define $\mathcal{F}_k := \big\{f_k(\cdot \mid \boldsymbol{V}, \boldsymbol{W}) \,:\, \mathrm{SimDiag}_{\mathbb{R}}(\boldsymbol{V}, \boldsymbol{W})\big\}$, where $\mathrm{SimDiag}_{\mathbb{R}}(\boldsymbol{V}, \boldsymbol{W})$ means that $\boldsymbol{V}, \boldsymbol{W} \in \mathbb{R}^{d\times d}$ are simultaneously orthogonally diagonalizable over $\mathbb{R}$. We impose this restriction to simplify the analysis. Note that this restriction still contains the entire GD-implementing family.

Data distributions. $\mathcal{P}$ and $\mathcal{Q}$ are different distributions over $(\boldsymbol{\Sigma}, \boldsymbol{\beta})$. We assume $\boldsymbol{\Sigma}$ and $\boldsymbol{\beta}$ are independent under both distributions, and $\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_d)$. Under $\mathcal{P}$, $\boldsymbol{\Sigma}$ is uniformly distributed over $\{\gamma_1 \boldsymbol{I}_d, \gamma_2 \boldsymbol{I}_d\}$ with $\gamma_1, \gamma_2 > 0$ and $\gamma_1 \neq \gamma_2$. Under $\mathcal{Q}$, $\boldsymbol{\Sigma} = \gamma_{\mathcal{Q}} \boldsymbol{I}_d$ for some $\gamma_{\mathcal{Q}} \in (0, \gamma_1+\gamma_2)\backslash\{\gamma_1, \gamma_2\}$.

A mechanistic explanation: self-iteration steers learning toward an in-context GD solver. We focus on even k. For the ID task, we can always achieve a perfect fit for any even $k \geq 2$. However, the resulting OOD loss decreases strictly as k increases, so larger k provides a real benefit that is not visible from ID performance alone. Our analysis also provides a mechanistic explanation: as k increases, ID minimizers behave increasingly like a generic GD solver, illustrating how self-iteration shapes the hypothesis class to favor more generalizable behavior. See proof in App. B.2.

Theorem 3.3. *We consider even $k \geq 2$. In the above setting, let $f_k^\star \in \arg\min_{f\in\mathcal{F}_k} L_{\mathcal{P}}(f)$ be any minimizer on $\mathcal{P}$. Let $a := \frac{\gamma_1+\gamma_2}{2}$, $b := \frac{|\gamma_1-\gamma_2|}{2}$. Then: (1)* ***(ID fit is always perfect)*** *$L_{\mathcal{P}}(f_k^\star) = 0$. (2)* ***(Increasing k improves OOD loss)*** *We have $L_{\mathcal{Q}}(f_k^\star) = d\left(\frac{(a-\gamma_{\mathcal{Q}})^k - b^k}{a^k - b^k}\right)^2$, which decreases strictly as k increases over even integers. (3)* ***(ID minimizers become increasingly similar to an in-context GD solver as k grows)*** *Define the parameter distance $\mathrm{dist}\big((\boldsymbol{V}, \boldsymbol{W}), \Theta_{\mathrm{GD}}\big) := \inf_{(\boldsymbol{V}', \boldsymbol{W}')\in\Theta_{\mathrm{GD}}} \Big(\|\boldsymbol{V} - \boldsymbol{V}'\|_F^2 + \|\boldsymbol{W} - \boldsymbol{W}'\|_F^2\Big)$. Then, for every ID minimizer $f_k^\star$ with parameters $(\boldsymbol{V}^\star, \boldsymbol{W}^\star)$, $\mathrm{dist}\big((\boldsymbol{V}^\star, \boldsymbol{W}^\star), \Theta_{\mathrm{GD}}\big) = d\left(\frac{b}{a}\right)^{2k}$, which decreases as k increases over even integers.*

3.2 Regularization Can Yield OOD Gains When Shortcuts Persist in the Hypothesis Class

When discussing OOD generalization, a commonly invoked concept is that of a *shortcut*. In general, a shortcut refers to a pattern that a model can exploit to fit the target on the training distribution

$\mathcal{P}$, but which is not reliable or exploitable on an unseen distribution $\mathcal{Q}$ [11]. From this perspective, in the setting analyzed in Sec. 3.1, one can view shortcut solutions as eventually disappearing as we increase k. However, shortcuts may *persist* in the hypothesis class even as we iterate. We therefore characterize another mechanism that can still yield OOD gains even when ID performance does not improve. The key requirement is that the learning algorithm employs an appropriate regularization—one that penalizes shortcut solutions more strongly than the structured, generalizable component. We formalize this mechanism below.

Let $\mathcal{P}, \mathcal{Q}$ be distributions over $\mathcal{X}$, and let $z : \mathcal{X} \to \mathcal{Y}$ be a fixed target function, where $\mathcal{Y}$ is a (real) Hilbert space. For any measurable $f : \mathcal{X} \to \mathcal{Y}$ and any distribution $\mathcal{D}$ over $\mathcal{X}$, define $L_{\mathcal{D}}(f) := \mathbb{E}_{x\sim\mathcal{D}}\|f(x) - z(x)\|^2$. Assume the hypothesis class admits a decomposition into a structured part $\mathcal{G}_k$ and a shortcut part $\mathcal{H}_k$: $\mathcal{F}_k = \{\, g + h \,:\, g \in \mathcal{G}_k,\ h \in \mathcal{H}_k \,\}$, with $0 \in \mathcal{G}_k$, $0 \in \mathcal{H}_k$. We learn on $\mathcal{P}$ with regularization R:

$$(\hat{g}, \hat{h}) \in \operatorname*{arg\,min}_{g\in\mathcal{G}_k,\ h\in\mathcal{H}_k} \Big(L_{\mathcal{P}}(g + h) + R(g, h) \Big), \tag{3}$$

with $R(g, h) := R_G(g) + R_H(h)$ [1] Here, R_G, R_H are nonnegative functionals with $R_G(0) = R_H(0) = 0$. We evaluate the learned predictor $\hat{f} := \hat{g} + \hat{h}$.

Assumption 3.4. We formalize $\mathcal{G}_k$ as a structured, well-constrained family that behaves similarly on ID and OOD data, and $\mathcal{H}_k$ as a weakly constrained family that can exploit shortcuts. We also make assumptions on the regularization.

Well-structured part: distribution shift control on $\mathcal{G}_k$. There exists $\gamma \geq 1$ such that for all $g \in \mathcal{G}_k$,

$$\frac{1}{\gamma} L_{\mathcal{P}}(g) \ \leq\ L_{\mathcal{Q}}(g) \ \leq\ \gamma L_{\mathcal{P}}(g), \tag{4}$$

capturing the idea that $\mathcal{G}_k$ contains transferable mechanisms.

Shortcut part: failure of OOD residual fit in $\mathcal{H}_k$. For any minimizer $(\hat{g}, \hat{h})$ of equation 3, define the residual $\hat{r} := z - \hat{g}$. We assume there exists $C > 0$ such that

$$\mathbb{E}_{x\sim\mathcal{Q}}\|\hat{h}(x) - \hat{r}(x)\|^2 \ \geq\ C\, \mathbb{E}_{x\sim\mathcal{Q}}\|\hat{r}(x)\|^2. \tag{5}$$

Intuitively, $\mathcal{H}_k$ captures non-transferable "shortcut" mechanisms: although $\hat{h}$ may help reduce the objective on $\mathcal{P}$, it cannot predict the remaining target residual $\hat{r} = z - \hat{g}$ on $\mathcal{Q}$ beyond a constant-factor improvement over doing nothing.

Regularization with the right bias. In practice, learning pipelines induce regularization via (1) *explicit* mechanisms—weight decay, KL penalties, objective clipping, trust-region updates, and LoRA's low-rank constraint [16]—and (2) *implicit* optimization bias [40, 1, 29, 32, 26, 49, 2]. We hypothesize that these effects bias learning toward more structured solutions, in the sense that they effectively control the shortcut component without overly penalizing the structured component.

(i) Norm control for the shortcut regularizer. $\exists$ constants $a_{\mathcal{P}}, a_{\mathcal{Q}} \geq 0$ such that for all $h \in \mathcal{H}_k$,

$$\mathbb{E}_{x\sim\mathcal{D}}\|h(x)\|^2 \leq a_{\mathcal{D}}\, R_H(h), \qquad \mathcal{D} \in \{\mathcal{P}, \mathcal{Q}\}. \tag{6}$$

One example is norm-based regularization, e.g., $R_H(h) \propto \|h\|^2_{\mathcal{H}}$ for an appropriate function norm.

(ii) Bounded regularization for a structured fit. The regularization does not overly penalize the structured component $\mathcal{G}_k$: there exists $\rho_k \geq 0$ and $g_k^\star \in \arg\min_{g\in\mathcal{G}_k} L_{\mathcal{P}}(g)$ s.t.

$$R_G(g_k^\star) \leq \rho_k, \tag{7}$$

Decoupled scaling of ID and OOD performance. In this decomposition view, ID performance is governed by how well the shortcut component can fit the training distribution, whereas OOD performance is governed—both from below and above—by how well the structured one can fit, as shown below.

Theorem 3.5. *Under Assumption 3.4, define $\epsilon_k := \min_{g\in\mathcal{G}_k} L_{\mathcal{P}}(g)$, which measures how well the structured component $\mathcal{G}_k$ can fit the target in-distribution, and define $\delta_k := \min_{h\in\mathcal{H}_k}\Big(L_{\mathcal{P}}(h) +$*

[1] For a clean presentation, we assume the regularizer decomposes additively. More generally, one can allow a bounded cross-term and obtain analogous results with extra terms scaling with its bound..

$R_H(h)\Big)$, *which measures the best regularized ID objective achievable using only the shortcut component. For any ID minimizer* $\hat{f}$ *(equation 3): (1)* ***ID upper bound.*** $L_{\mathcal{P}}(\hat{f}) \le \delta_k$. *(2)* ***OOD sandwich bound.*** $C_1\,\epsilon_k \;\le\; L_{\mathcal{Q}}(\hat{f}) \;\le\; C_2\,\big(\epsilon_k + \rho_k\big)$, *where* $C_1 := \frac{C}{\gamma}$ *and* $C_2 := 4\gamma + 4\gamma a_{\mathcal{P}} + 2a_{\mathcal{Q}}$.

Consider a regime in which the shortcut channel $\mathcal{H}_k$ is highly expressive and achieves a small regularized ID objective even for small k, so that δ_k decays rapidly. In contrast, the structured channel $\mathcal{G}_k$ may require larger k for its expressive power to manifest, so ϵ_k decays much more slowly. In this regime, the theorem implies that the ID error—upper bounded by δ_k—can decrease quickly and plateau early, while the OOD error—lower bounded by ϵ_k up to constants—can improve no faster than the rate at which ϵ_k decreases. Note that Assumption 3.4 only specifies that $\mathcal{H}_k$ is non-transferable and norm-controlled while $\mathcal{G}_k$ transfers across shifts, without assuming OOD improvement with k: improvement further requires ϵ_k to decrease with k while ρ_k stays controlled.

Role of regularization in the competition between $\mathcal{G}_k$ and $\mathcal{H}_k$. The tightness of the OOD upper bound in Thm. 3.5 is affected by C_2 and ρ_k. A tighter bound favors (i) sufficiently strong regularization on the shortcut channel, reflected by reasonable norm-control constants $a_{\mathcal{P}}, a_{\mathcal{Q}}$, and (ii) mild regularization on the structured channel, reflected by a small ρ_k (i.e., a small penalty on the structured fit). Mechanistically, as k grows and $\mathcal{G}_k$ becomes expressive enough to compete with shortcut fits, this regularization profile suppresses the shortcut component $\hat{h}$ while leaving a low-complexity structured fit available, biasing minimizers toward $\mathcal{G}_k$. As a result, reliance on $\mathcal{H}_k$ decreases and OOD performance can continue to improve even after the ID error has plateaued.

3.2.1 A concrete example with slow structured gains and fast shortcuts

For the purpose of illustrating the general result in Thm. 3.5, we present an admittedly stylized instantiation designed to make the structured/shortcut decomposition and the role of regularization transparent. Although analyzing highly complex models remains challenging at present, the same qualitative behavior is empirically supported in Sec. 4.

For a sequence $s = (e_1, \ldots, e_T)$, let $s[i]$ denote its i-th element and $s[-1]$ its last element; when $T = 1$, we identify s with e_1 when the context is clear. Each element is a token represented as a vector in $\mathbb{R}^d$, and the prompt is a single token $x \in \mathbb{R}^d$. For a vector $\mathbf{v} \in \mathbb{R}^d$, let $\mathbf{v}_{i:j}$ denote coordinates i through j.

An autoregressive model. We define $\mathrm{Auto}(s) := \mathrm{Append}(s,\, N(s))$, where N is a next-token generator that maps the current sequence to the next token. After k steps, the model outputs $f_k(x) := \mathrm{Extract}(\mathrm{Auto}^{\circ k}(x))$, where $\mathrm{Extract}(\cdot)$ maps a token sequence to $\mathcal{Y}$.

An iterative channel and a non-iterative channel. Next, we specify the next-token generator N. Let $a \in \mathcal{A}$ be a function $\mathbb{R}^p \to \mathbb{R}^p$ and $b \in \mathcal{B}$ be a function $\mathbb{R}^p \to \mathbb{R}^l$. Assume

$$N(s)_{1:p} = a\big(s[-1]_{1:p}\big),\; N(s)_{p+1:p+l} = b\big(s[1]_{1:p}\big). \tag{8}$$

with all other coordinates arbitrary as they play no role in the construction. Thus a is iterated across steps (so its expressivity can grow with k), while b depends only on the initial token and does not benefit from additional iterations. They write to disjoint coordinate blocks, so they do not interfere.

Resulting function class decomposition. Let $\psi_1 : \mathbb{R}^p \to \mathcal{Y}$ and $\psi_2 : \mathbb{R}^l \to \mathcal{Y}$. We define the extraction rule $\mathrm{Extract}(s) := \psi_1\big(s[-1]_{1:p}\big) + \psi_2\big(s[-1]_{p+1:p+l}\big)$. The prediction induced by the model is then given by

$$f_k(x) = \psi_1\Big(a^{\circ k}\big(x_{1:p}\big)\Big) + \psi_2\Big(b\big(x_{1:p}\big)\Big). \tag{9}$$

The induced hypothesis class decomposes as $\mathcal{G}_k = \{\, \psi_1 \circ a^{\circ k} \circ \pi : a \in \mathcal{A} \,\},\; \mathcal{H}_k = \{\, \psi_2 \circ b \circ \pi : b \in \mathcal{B} \,\}$.

Data. Let $\mathcal{Y} = \mathbb{R}$ and $l = 1$. Let the target function z be linear in the first p coordinates: $z(x) = \pi(x)^\top \theta$, $\theta = (\theta_1, \ldots, \theta_p)^\top \in \mathbb{R}^p_{\ge 0}$. Let $\mathcal{U} := \{\pm 1\}^p$ be the hypercube, with two disjoint subsets $\mathcal{U}_P := \{u \in \mathcal{U} :\; u_1 = +1\}$, $\mathcal{U}_Q := \{u \in \mathcal{U} :\; u_1 = -1\}$. We define distributions on inputs $x \in \mathbb{R}^d$ by first sampling $u \in \mathbb{R}^p$, then setting $x_{1:p} = u$; the remaining coordinates can be arbitrary and play no role in the construction. Let $\mathcal{P}$ be the distribution obtained by sampling $u \sim \mathrm{Unif}(\mathcal{U}_P)$, and let $\mathcal{Q}$ be obtained by sampling $u \sim \mathrm{Unif}(\mathcal{U}_Q)$.

Functions. Let $\psi_1(v) = \mathbf{1}^\top v$ for $v \in \mathbb{R}^p$ and $\psi_2(t) = t$ for $t \in \mathbb{R}$. We define $\mathcal{A}$ and $\mathcal{B}$ as follows.

(i) A structured channel that improves with k. Define the structured class $\mathcal{A}$ by diagonal linear maps:

$$\mathcal{A} := \Big\{ a(u) = \mathrm{diag}(w_1, \ldots, w_p)\, u \;\Big|\; 0 \le w_i \le \tau \text{ for all } i \Big\},$$

where $\tau > 1$. $\mathcal{A}$ represents a gradually improving structured mechanism: it is well-aligned with the target z (both are linear maps), but may be unable to represent z accurately under a small k. As k increases, repeated composition increases τ^k and enables increasingly accurate approximation of z.

(ii) An overly expressive shortcut channel. Let $\mathcal{B}$ be the RKHS on $\mathcal{U}$ induced by the Kronecker delta kernel $K(u, u') = \mathbf{1}\{u = u'\}$. This shortcut channel can produce essentially arbitrary labels on the finite support of $\mathcal{P}$ but provides no meaningful control on unseen inputs from $\mathcal{Q}$ when the supports are disjoint. It can be viewed as an analogue of the unconstrained capacity of a heavily overparameterized neural network: for instance, when trained on math data, an LLM may internalize a task-specific computation and produce answers with little step-by-step symbolic reasoning. This can achieve low ID error but need not transfer reliably to unseen problems. We observe related behaviors empirically in Sec. 4.2.

Regularization. We regularize the next-token generator N (Eq. 8) by its squared functional norm. Concretely, for any induced predictor $f_k = g + h$ of the form in Eq. 9, we define $R(g, h) := \lambda \|N\|^2$, $\|N\|^2 := \|a\|^2 + \|b\|_{\mathcal{B}}^2$, for some $\lambda > 0$. $\|a\|^2$ is the squared Frobenius norm of the induced linear map, i.e., $\sum_{i=1}^{p} w_i^2$, and $\|b\|_{\mathcal{B}}^2$ is the squared RKHS norm under the kernel defining $\mathcal{B}$. Equivalently, this corresponds to $R_G(g) := \lambda \|a\|^2$ and $R_H(h) := \lambda \|b\|_{\mathcal{B}}^2$.

Finally, we verify that the above satisfies Assumption 3.4:

Proposition 3.6. *Letting $M := 2^{p-1}$, Assumption. 3.4 holds with $\gamma = 1$, $C = 1$, $a_{\mathcal{P}} = a_{\mathcal{Q}} = 1/(\lambda M)$, $\rho_k = \lambda \sum_{i=1}^{p} \theta_i^{2/k}$, $\epsilon_k = \sum_{i=1}^{p} ((\theta_i - \tau^k)_+)^2$, and $\delta_k = \frac{\lambda M}{1+\lambda M} \|\theta\|_2^2$.*

To illustrate the theorem numerically, let's take $p = 8$, $\theta = [0, 2, 0, 0 \ldots]^\top$, $\tau = 1.1$, and $\lambda = 10^{-5}$. Then $\rho_k = \frac{\sqrt[k]{4}}{10^5}$, $\epsilon_k = ((2 - 1.1^k)_+)^2$, $\delta_k \approx 0.005$. By Thm. 3.5, the ID loss satisfies $L_{\mathcal{P}}(\hat{f}) \le \delta_k \le 0.006$ for all k, so the ID error is small and plateaus right from the beginning. In contrast, the OOD loss is bounded as $((2 - 1.1^k)_+)^2 \le L_{\mathcal{Q}}(\hat{f}) \le 4692\left(((2 - 1.1^k)_+)^2 + \frac{\sqrt[k]{4}}{10^5}\right)$. In particular, when $k = 1$, $L_{\mathcal{Q}}(\hat{f}) \ge 0.81$. When $k \ge 8$, the upper bound gives $L_{\mathcal{Q}}(\hat{f}) \le 0.056$. Thus increasing k from 1 to 8 provably reduces the OOD loss by at least 0.754, despite almost no ID improvement.

4 Experiments

4.1 Scaling Latent Reasoning via Looped Transformers on Synthetic Algorithmic Tasks

Looped Transformers have recently emerged as a new way to scale reasoning length via additional "latent thoughts" [47, 12, 34]. Conceptually, a k-loop model can be viewed as the k-fold self-composition of a single Transformer block: the same backbone (with shared weights) is applied repeatedly, rather than stacking k distinct layers with different parameters. See the model and training details in App. E.1.2.

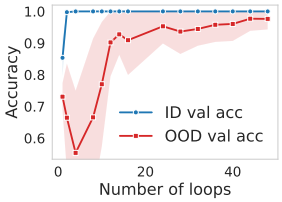


Figure 1: Looped Transformers trained on 4-hop problems. Results averaged over 8 random seeds. Increasing the loop count yields OOD gains up to around 44 loops, while ID accuracy saturates near 100% by around 2 loops.

p-hop induction. We consider the problem studied in [33, 34]. In this task, an input is a sequence over a finite alphabet. Starting from the last position, the model repeatedly "backtracks" by jumping to the position after the most recent earlier occurrence of the current symbol; after p such steps, it must output the symbol at the resulting location. See App. E.1.1 for the formal definition. This probes a form of multi-step retrieval that is reminiscent of the iterative reasoning required in certain reading-comprehension-style settings.

ID/OOD split. Let $i_1, \ldots, i_p$ denote the hop indices (i.e., the positions visited during backtracking). We assign an instance to ID if $\sum_{t=1}^{p} i_t$ is even, and to OOD otherwise, creating a strictly disjoint-support shift while unseen even-parity instances serve as a same-distribution ID control.

Fig. 1 summarizes the results. As we increase the number of loops, ID validation accuracy quickly saturates to nearly 100% (around 2 loops), while OOD validation accuracy improves much more slowly and continues to increase even at very large loop counts.

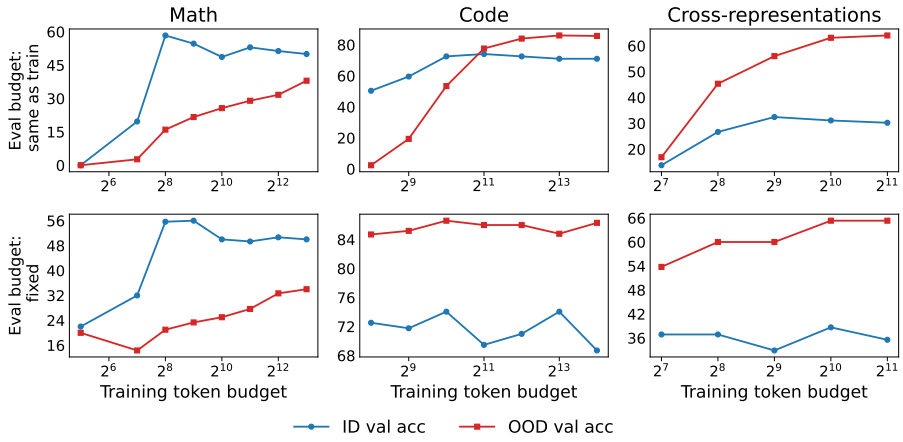


Figure 2: RL fine-tuning across three shifts with varying training token budgets (Pass@10). **Left (Math):** `Qwen2.5-1.5B-Instruct` (`Polynomial Roots` → `Mixed Operation`, 3 seeds). **Middle (Code):** `Qwen3-8B` (`LiveCodeBench v6` → `DAPO-Math-17K`). **Right (Cross-representations):** `Qwen2.5-1.5B-Instruct` (NL → graph transfer). **Top / Bottom:** matched vs. fixed max evaluation budget (8192, 16384, 2048). OOD improves steadily on math and cross-representations; on code, matched-budget OOD climbs while fixed-budget OOD sits near ceiling ($\approx$ 84–85%).

4.2 Scaling Token Budgets in RL Fine-Tuning of LLMs

We next fine-tune LLMs across multiple domains with RL, varying the training-time token budget that caps the maximum allowed CoT reasoning length.

Datasets and distribution shifts. Spanning changes in procedure, domain, format, and representation, we evaluate three distinct shifts (complementing the disjoint-support parity split in Sec. 4.1): **(1) Math (within-domain task shift):** `Polynomial Roots` (ID, adapted from Sun et al. [41] for 1.5B solvability; App. E.2) to `Mixed Operation` [41] (OOD). **(2) Code (cross-domain/format shift):** `LiveCodeBench v6` [19] (ID) to `DAPO-Math-17K` [48] (OOD, integer-answer math). **(3) Cross-representations (representation shift):** natural-language (ID) to graph (OOD) variants of isomorphic tasks from Lin et al. [25], fixing the underlying procedure. Training uses the ID train split and evaluates on unseen ID and OOD validation splits, providing a direct same-distribution control.

Models and evaluation setups. Using GRPO [13] with correctness-only rewards and LoRA [16], we fine-tune `Qwen2.5-1.5B-Instruct` [42], `Qwen3-8B`, `Qwen3-4B` [43], and `Olmo3-7B` [30] (App. E.2). To separate training budgets from test-time truncation, we evaluate under a *matched* budget equal to the training cap (Fig. 2, top) and a *fixed* budget set to each sweep's maximum cap (Fig. 2, bottom). Under both setups, ID accuracy plateaus by budget 256 (even slightly dropping thereafter), while OOD accuracy improves steadily on math and cross-representations (Pass@1 in App. E.2.2). On code, the OOD trend is visible under the matched evaluation budget. However, under the fixed evaluation budget the OOD metric sits near ceiling at every training budget ($\approx$ 84–85%), so that panel has no headroom and we do not treat it as evidence for the phenomenon.

Scaling across model capacity and lineages. Since Thm. 3.5 fixes the base class, it prescribes no parameter-scaling law: extra capacity can aid either shortcuts ($\delta_k \downarrow$) or structured fits ($\epsilon_k, \rho_k \downarrow$), and cross-size comparisons confound pre-training quality. Empirically, Fig. 3 confirms that our findings persist beyond 1.5B and Qwen: on `Qwen3-4B` (harder `Mixed Operation` OOD to avoid ceiling), fixed-8192 ID plateaus (86–91%) while OOD Pass@10 gains +27.5 points (45.5% → 73.0%); on independent open-data `Olmo3-7B` [30] (Pass@1, as Pass@10 saturates), matched-budget OOD climbs 0.3% → 75.8% while fixed-budget OOD sits near ceiling (75.2–76.3%), reflecting benchmark headroom rather than Qwen-specificity.

Reasoning dynamics and realized length. In Fig. 6 (App. E.2.2), small budgets (128/256) show near-zero reward until response length abruptly collapses to minimal reasoning, whereas large budgets (4096) grow length throughout training (examples in App. E.2.2). Table 1a further separates the training cap from uncensored test-time length on `Qwen3-4B` (eval cap 8192 $\gg$ 1743 max mean). Realized length grows with the cap except at 512 → 1024 (1038 → 962), the *only* step where OOD dips on both Pass@1 (26.9 → 24.2) and Pass@10 (49.0 → 46.5). Across the $n = 7$ budgets, OOD rank

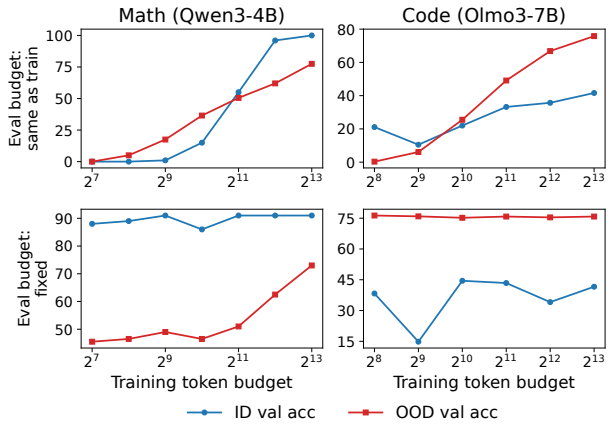


Figure 3: **Generalizability across model scale and lineage** (numbers in Table 2, App. E.2.2). **Left:** `Qwen3-4B` on harder `Mixed Operation` OOD (Pass@10); at fixed 8192 cap (bottom), ID plateaus (86–91%) while OOD gains +27.5 points (45.5% → 73.0%). **Right:** `Olmo3-7B` on code (Pass@1), reproducing the matched-budget OOD climb (0.3% → 75.8%, top) while fixed-budget OOD sits near ceiling (75.2–76.3%, bottom).

Table 1: **Realized reasoning length and compute trade-offs:** *(a)* uncensored OOD length and accuracy on `Qwen3-4B` (eval @ 8192); *(b)* estimated training FLOPs vs. OOD Pass@10 on `Qwen2.5-1.5B-Instruct`.

(a) Realized length (`Qwen3-4B`, eval @ 8192)

Cap	Len. (tok)	Pass@1	Pass@10
128	803	25.7	45.5
256	925	26.2	46.5
512	1038	26.9	49.0
1024	962	24.2	46.5
2048	1068	28.1	51.0
4096	1336	34.8	62.5
8192	1743	43.1	73.0

(b) Compute trade-off (`Qwen2.5-1.5B`, Math OOD)

Cap	FLOPs	vs. min	OOD	ΔOOD	$\Delta/10^{18}$
32	4.8×10^{17}	1.0×	0.0	—	—
128	5.7×10^{17}	1.2×	2.7	+2.7	30
256	6.8×10^{17}	1.4×	16.0	+13.3	121
512	9.0×10^{17}	1.9×	21.7	+5.7	26
1024	1.4×10^{18}	2.8×	25.7	+4.0	8.7
2048	2.3×10^{18}	4.7×	29.0	+3.3	3.7
4096	4.1×10^{18}	8.5×	31.7	+2.7	1.5
8192	7.7×10^{18}	16.0×	38.0	+6.3	1.7

correlation is higher with realized length than with the cap (0.89 vs. 0.79 on Pass@1; 0.99 vs. 0.94 on Pass@10), with partial $r = 0.95$ ($p < 0.001$) on Pass@1 controlling for the cap—supporting the cap as an instrument for realized self-iteration k despite collinearity (KL $= 0$ ablation in App. E.2.2).

Compute trade-offs and shift sensitivity. Table 1b tracks OOD Pass@10 vs. estimated training compute on `Qwen2.5-1.5B-Instruct` ($\approx 10N$ FLOPs/token, $N = 1.54 \times 10^9$, 57,344 sequences): OOD rises up to 8192 (16× compute), with marginal gains diminishing from 121 and 26 points per 10^{18} FLOPs at 256–512 to 1.5–1.7 at 4096–8192. On distant benchmarks—`MATH500` [15, 24] and `GSM8K` [4]—checkpoints show small, stable gains (+3.7 and +1.7 points), indicating that OOD gains scale with shared procedural structure while preserving general math ability.

5 Conclusion, Limitations, and Discussion

We study how scaling training-time (explicit or latent) reasoning length affects OOD generalization under outcome supervision. Viewing reasoning as self-iteration, we show theoretically that increasing it can improve OOD performance even after ID saturation by reshaping the induced hypothesis class and interacting with regularization to suppress shortcuts, and corroborate this in looped Transformers and RL fine-tuning across multiple shifts and model lineages.

Limitations. (i) *Stylized theory:* our formal results analyze population optima in tractable constructions rather than LLM RL dynamics, holding the base class fixed without parameter-scaling laws. (ii) *Scale, compute, and headroom:* our RL sweeps span models up to 8B (`Qwen2.5/3`, `Olmo3`), show diminishing OOD gains per FLOP (Table 1), and require benchmark headroom (fixed-budget code OOD sits near ceiling). (iii) *Shift sensitivity and shortcuts:* gains rely on shared procedural structure (modest on `MATH500/GSM8K`) and do not rule out "seemingly natural" shortcuts escaping regularization. Rather than claiming universality, we clarify when hidden OOD gains arise.

Acknowledgments and Disclosure of Funding

This research was supported by NSF CAREER Award 2146492, NSF-Simons AI Institute for Cosmic Origins (CosmicAI), NSF AI Institute for Foundations of Machine Learning (IFML), and an Apple Research Award.

References

[1] Sanjeev Arora, Nadav Cohen, Wei Hu, and Yuping Luo. Implicit regularization in deep matrix factorization. *Advances in Neural Information Processing Systems*, 32, 2019. URL https://arxiv.org/abs/1905.13655.

[2] David G. T. Barrett and Benoit Dherin. Implicit gradient regularization. In *International Conference on Learning Representations*, 2021. URL https://arxiv.org/abs/2009.11162.

[3] Locke Cai and Ivan Provilkov. Escaping the verifier: Learning to reason via demonstrations. *arXiv preprint arXiv:2511.21667*, 2025.

[4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021. URL https://arxiv.org/abs/2110.14168.

[5] Artur Back De Luca and Kimon Fountoulakis. Simulation of graph algorithms with looped transformers. *arXiv preprint arXiv:2402.01107*, 2024.

[6] Mostafa Dehghani, Stephan Gouws, Oriol Vinyals, Jakob Uszkoreit, and Łukasz Kaiser. Universal transformers. *arXiv preprint arXiv:1807.03819*, 2018.

[7] Ying Fan, Yilun Du, Kannan Ramchandran, and Kangwook Lee. Looped transformers for length generalization. *arXiv preprint arXiv:2409.15647*, 2024.

[8] Yihang Gao, Chuanyang Zheng, Enze Xie, Han Shi, Tianyang Hu, Yu Li, Michael K Ng, Zhenguo Li, and Zhaoqiang Liu. On the expressive power of a variant of the looped transformer. *CoRR*, 2024.

[9] Khashayar Gatmiry, Nikunj Saunshi, Sashank J Reddi, Stefanie Jegelka, and Sanjiv Kumar. Can looped transformers learn to implement multi-step gradient descent for in-context learning? *arXiv preprint arXiv:2410.08292*, 2024.

[10] Khashayar Gatmiry, Nikunj Saunshi, Sashank J Reddi, Stefanie Jegelka, and Sanjiv Kumar. On the role of depth and looping for in-context learning with task diversity. *arXiv preprint arXiv:2410.21698*, 2024.

[11] Robert Geirhos, Jörn-Henrik Jacobsen, Claudio Michaelis, Richard Zemel, Wieland Brendel, Matthias Bethge, and Felix A Wichmann. Shortcut learning in deep neural networks. *Nature Machine Intelligence*, 2(11):665–673, 2020.

[12] Angeliki Giannou, Shashank Rajput, Jy-yong Sohn, Kangwook Lee, Jason D Lee, and Dimitris Papailiopoulos. Looped transformers as programmable computers. In *International Conference on Machine Learning*, pages 11398–11442. PMLR, 2023.

[13] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, Xiao Bi, et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025.

[14] Shibo Hao, Sainbayar Sukhbaatar, DiJia Su, Xian Li, Zhiting Hu, Jason Weston, and Yuandong Tian. Training large language models to reason in a continuous latent space. *arXiv preprint arXiv:2412.06769*, 2024.

[15] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. In *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track*, 2021. URL https://arxiv.org/abs/2103.03874.

[16] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. Lora: Low-rank adaptation of large language models. *ICLR*, 1 (2):3, 2022.

[17] Jianhao Huang, Zixuan Wang, and Jason D Lee. Transformers learn to implement multi-step gradient descent with chain of thought. *arXiv preprint arXiv:2502.21212*, 2025.

[18] Aaron Jaech, Adam Kalai, Adam Lerer, Adam Richardson, Ahmed El-Kishky, Aiden Low, Alec Helyar, Aleksander Madry, Alex Beutel, Alex Carney, et al. Openai o1 system card. *arXiv preprint arXiv:2412.16720*, 2024.

[19] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. LiveCodeBench: Holistic and contamination free evaluation of large language models for code. *arXiv preprint arXiv:2403.07974*, 2024. URL `https://arxiv.org/abs/2403.07974`.

[20] Zeyu Jia, Alexander Rakhlin, and Tengyang Xie. Do we need to verify step by step? rethinking process supervision from a theoretical perspective. *arXiv preprint arXiv:2502.10581*, 2025.

[21] Devvrit Khatri, Lovish Madaan, Rishabh Tiwari, Rachit Bansal, Sai Surya Duvvuri, Manzil Zaheer, Inderjit S Dhillon, David Brandfonbrener, and Rishabh Agarwal. The art of scaling reinforcement learning compute for llms. *arXiv preprint arXiv:2510.13786*, 2025.

[22] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.

[23] Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*, 1, 2024.

[24] Hunter Lightman, Vineet Kosaraju, Yuri Burda, Harrison Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let's verify step by step. In *The Twelfth International Conference on Learning Representations*, 2023.

[25] Fangru Lin, Valentin Hofmann, Xingchen Wan, Weixing Wang, Zifeng Ding, Anthony G. Cohn, and Janet B. Pierrehumbert. Can large language models generalize procedures across representations? In *Proceedings of the 43rd International Conference on Machine Learning*, 2026.

[26] Kaifeng Lyu, Zhiyuan Li, Runzhe Wang, and Sanjeev Arora. Gradient descent on two-layer nets: Margin maximization and simplicity bias. *Advances in Neural Information Processing Systems*, 34, 2021. URL `https://arxiv.org/abs/2110.13905`.

[27] William Merrill and Ashish Sabharwal. The expressive power of transformers with chain of thought. *arXiv preprint arXiv:2310.07923*, 2023.

[28] Niklas Muennighoff, Zitong Yang, Weijia Shi, Xiang Lisa Li, Li Fei-Fei, Hannaneh Hajishirzi, Luke Zettlemoyer, Percy Liang, Emmanuel Candès, and Tatsunori B Hashimoto. s1: Simple test-time scaling. In *Proceedings of the 2025 Conference on Empirical Methods in Natural Language Processing*, pages 20286–20332, 2025.

[29] Preetum Nakkiran, Gal Kaplun, Dimitris Kalimeris, Tristan Yang, Benjamin L. Edelman, Fred Zhang, and Boaz Barak. SGD on neural networks learns functions of increasing complexity. *Advances in Neural Information Processing Systems*, 32, 2019. URL `https://arxiv.org/abs/1905.11604`.

[30] Team Olmo, Allyson Ettinger, Amanda Bertsch, Bailey Kuehl, David Graham, David Heineman, Dirk Groeneveld, Faeze Brahman, Finbarr Timbers, Hamish Ivison, et al. Olmo 3. *arXiv preprint arXiv:2512.13961*, 2025. URL `https://arxiv.org/abs/2512.13961`.

[31] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.

[32] Noam Razin and Nadav Cohen. Implicit regularization in deep learning may not be explainable by norms. *Advances in Neural Information Processing Systems*, 33, 2020. URL https://arxiv.org/abs/2005.06398.

[33] Clayton Sanford, Daniel Hsu, and Matus Telgarsky. Transformers, parallel computation, and logarithmic depth. *arXiv preprint arXiv:2402.09268*, 2024.

[34] Nikunj Saunshi, Nishanth Dikkala, Zhiyuan Li, Sanjiv Kumar, and Sashank J Reddi. Reasoning with latent thoughts: On the power of looped transformers. *arXiv preprint arXiv:2502.17416*, 2025.

[35] John Schulman, Sergey Levine, Pieter Abbeel, Michael Jordan, and Philipp Moritz. Trust region policy optimization. In *International conference on machine learning*, pages 1889–1897. PMLR, 2015.

[36] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*, 2017.

[37] Amrith Setlur, Matthew YR Yang, Charlie Snell, Jeremy Greer, Ian Wu, Virginia Smith, Max Simchowitz, and Aviral Kumar. e3: Learning to explore enables extrapolation of test-time compute for llms. *arXiv preprint arXiv:2506.09026*, 2025.

[38] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Yang Wu, et al. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

[39] Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

[40] Daniel Soudry, Elad Hoffer, Mor Shpigel Nacson, Suriya Gunasekar, and Nathan Srebro. The implicit bias of gradient descent on separable data. *The Journal of Machine Learning Research*, 19(1):2822–2878, 2018.

[41] Yiyou Sun, Shawn Hu, Georgia Zhou, Ken Zheng, Hannaneh Hajishirzi, Nouha Dziri, and Dawn Song. Omega: Can llms reason outside the box in math? evaluating exploratory, compositional, and transformative generalization. *arXiv preprint arXiv:2506.18880*, 2025.

[42] Qwen Team. Qwen2.5: A party of foundation models, September 2024. URL https://qwenlm.github.io/blog/qwen2.5/.

[43] Qwen Team. Qwen3 technical report, 2025. URL https://arxiv.org/abs/2505.09388.

[44] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

[45] Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. Inference scaling laws: An empirical analysis of compute-optimal inference for problem-solving with language models. *arXiv preprint arXiv:2408.00724*, 2024.

[46] Kevin Xu and Issei Sato. On expressive power of looped transformers: Theoretical analysis and enhancement via timestep encoding. *arXiv preprint arXiv:2410.01405*, 2024.

[47] Liu Yang, Kangwook Lee, Robert Nowak, and Dimitris Papailiopoulos. Looped transformers are better at learning learning algorithms. *arXiv preprint arXiv:2311.12424*, 2023.

[48] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. Dapo: An open-source llm reinforcement learning system at scale. *arXiv preprint arXiv:2503.14476*, 2025.

[49] Chiyuan Zhang, Samy Bengio, Moritz Hardt, Benjamin Recht, and Oriol Vinyals. Understanding deep learning (still) requires rethinking generalization. *Communications of the ACM*, 64(3): 107–115, 2021.

Contents

1 Introduction **1**

2 Related Work **2**

3 Training-Time Self-Iteration Can Yield OOD Gains Without ID Gains **3**

3.1 Self-Iteration Can Strengthen Inductive Bias in the Hypothesis Class 3

3.1.1 Example: Self-Iterating the Full Hypothesis Class on a Cycle Task 3

3.1.2 Example: In-context coefficient prediction with a linear transformer 4

3.2 Regularization Can Yield OOD Gains When Shortcuts Persist in the Hypothesis Class 5

3.2.1 A concrete example with slow structured gains and fast shortcuts 7

4 Experiments **8**

4.1 Scaling Latent Reasoning via Looped Transformers on Synthetic Algorithmic Tasks 8

4.2 Scaling Token Budgets in RL Fine-Tuning of LLMs 9

5 Conclusion, Limitations, and Discussion **10**

A Proof of Theorem 3.2 **15**

A.1 Overview 15

A.2 A structural uniqueness criterion for fixed n and k 15

A.2.1 Functional-graph facts for large iterates 15

A.2.2 The only height-≤ 1 interpolants 16

A.2.3 Which interpolants are k-fold iterates? 16

A.2.4 Uniqueness for fixed n, k 18

A.3 Number theory: densities, totients, and Dirichlet 18

A.4 High-density uniqueness for the cycle task via large iterates 19

B Proof of Theorem 3.3 **20**

B.1 Explicit Form of the Infinite-Context Limit 20

B.2 Characterizing in-distribution minimizers and convergence to GD 21

C Proof of Theorem 3.5 **23**

D Proof of Proposition 3.6 **24**

E Experimental Details **26**

E.1 Additional Details for Section 4.1 26

E.1.1 Description of the p-hop Problem 26

E.1.2 Model and Training Details 26

E.1.3 Additional Experiments with Loop Embedding 26

E.2 Additional Details for Section 4.2 27

E.2.1 Data and Training Details 27
E.2.2 Additional Results and Regularization Ablation 28

A Proof of Theorem 3.2

A.1 Overview

We study the "cycle task" on a finite set $\mathbb{S}$ of size n, where the ground-truth target $z : \mathbb{S} \to \mathbb{S}$ is a cyclic permutation (a single directed n-cycle). The training distribution is uniform on all but one held-out point $q \in \mathbb{S}$. We consider the hypothesis class consisting of k-fold self-iterates $f^{\circ k}$ of arbitrary functions $f : \mathbb{S} \to \mathbb{S}$.

The main goal is a *high-density* uniqueness statement: for any desired accuracy level $\varepsilon > 0$, we will choose a suitable (large) prime n so that for at least a $(1 - \varepsilon)$ fraction of all integers $k \geq n - 1$, the empirical risk minimizer over $\mathcal{F}_k = \mathcal{F}^{\circ k}$ is *unique* and equals the true cycle map z.

A WLOG reduction to the canonical cycle. Because z is a single n-cycle, there exists a relabeling bijection $\pi : \mathbb{S} \to \mathbb{S}$ such that $\pi \circ z \circ \pi^{-1} = z_0$, where $z_0(i) = i + 1 \pmod n$. Conjugation preserves k-fold iterates: $(\pi f \pi^{-1})^{\circ k} = \pi f^{\circ k} \pi^{-1}$, and it preserves which hypotheses interpolate $\mathbb{P}$ (up to relabeling the held-out point). Thus it suffices to prove the theorem for the canonical target $z(i) = i + 1 \pmod n$. We do so below, and the general statement follows by relabeling.

Henceforth, assume

$$z(i) \coloneqq i + 1 \pmod n.$$

Natural density. For a set $A \subseteq \mathbb{N}$, define its (lower) natural density by

$$\operatorname{dens}(A) \coloneqq \liminf_{N \to \infty} \frac{|A \cap \{1, 2, \ldots, N\}|}{N}.$$

(All sets we use below are periodic modulo some integer, hence in fact have a true limit density.) When we say "a $(1 - \varepsilon)$ fraction of integers $k \geq m$", we mean that

$$\liminf_{N \to \infty} \frac{|\{k \in \{m, \ldots, N\} : k \in A\}|}{N - m + 1} \geq 1 - \varepsilon,$$

equivalently $\operatorname{dens}(A) \geq 1 - \varepsilon$ since removing finitely many integers does not change density.

A.2 A structural uniqueness criterion for fixed n and k

A.2.1 Functional-graph facts for large iterates

Functional-graph terminology. Any map $g : \mathbb{S} \to \mathbb{S}$ can be viewed as a directed graph with outdegree 1 at each vertex. A point x is *periodic for* g if $g^{\circ t}(x) = x$ for some $t \geq 1$. The *height* $\operatorname{ht}(g)$ is the maximum, over all $x \in \mathbb{S}$, of the length of the shortest directed path from x to the set of periodic points of g. Equivalently,

$$\operatorname{ht}(g) \leq 1 \quad \Longleftrightarrow \quad \text{for every } x \in \mathbb{S},\ \ g(x) \text{ is periodic for } g.$$

Lemma A.1 (Large iterates collapse height)**.** *Let $|\mathbb{S}| = n$ and let $k \geq n - 1$. For every function $f : \mathbb{S} \to \mathbb{S}$, the iterate $g \coloneqq f^{\circ k}$ satisfies $\operatorname{ht}(g) \leq 1$.*

Proof. Fix $x \in \mathbb{S}$ and consider the forward orbit under f:

$$x,\ f(x),\ f^{\circ 2}(x),\ \ldots.$$

Among the $n + 1$ points $x, f(x), \ldots, f^{\circ n}(x)$ there must be a repetition, so there exist indices $0 \leq i < j \leq n$ such that $f^{\circ i}(x) = f^{\circ j}(x)$. Let $y \coloneqq f^{\circ i}(x)$. Then $f^{\circ (j-i)}(y) = y$, so y is periodic for f, and moreover $i \leq n - 1$.

Since $k \geq n - 1 \geq i$, we may write

$$g(x) = f^{\circ k}(x) = f^{\circ (k-i)}(y).$$

Because y lies on a directed cycle of f, every forward iterate of y also lies on that cycle; in particular $g(x)$ is periodic for f.

Finally, if a point u is periodic for f, say $f^{\circ m}(u) = u$, then it is also periodic for $g = f^{\circ k}$ because

$$g^{\circ m}(u) = (f^{\circ k})^{\circ m}(u) = f^{\circ(km)}(u) = u.$$

Applying this to $u = g(x)$ shows that $g(x)$ is periodic for g. Since x was arbitrary, every point reaches a periodic point of g in at most one step, i.e. $\mathrm{ht}(g) \leq 1$. □

A.2.2 The only height-≤ 1 interpolants

Define the "skip-one" map $\tilde{z} : \mathbb{S} \to \mathbb{S}$ by

$$\tilde{z}(x) \coloneqq \begin{cases} z(x), & x \neq q, \\ z^{\circ 2}(q), & x = q. \end{cases} \tag{10}$$

Thus $\tilde{z}$ agrees with z on the training set $\mathbb{P}$ but sends the held-out point q forward by two steps.

Lemma A.2 (Only two interpolants can have height at most 1)**.** *Let $g : \mathbb{S} \to \mathbb{S}$ satisfy $g(x) = z(x)$ for all $x \in \mathbb{P}$. If $\mathrm{ht}(g) \leq 1$, then g must equal either z or $\tilde{z}$.*

Proof. Because z is an n-cycle, there is a unique integer $t \in \{1, 2, \ldots, n\}$ such that

$$g(q) = z^{\circ t}(q),$$

where $t = n$ corresponds to $g(q) = q$.

If $t = 1$, then $g(q) = z(q)$ and hence $g = z$.

If $t = 2$, then g agrees with z on $\mathbb{P}$ and sends q to $z^{\circ 2}(q)$, so $g = \tilde{z}$.

It remains to show that $t \geq 3$ is impossible under $\mathrm{ht}(g) \leq 1$.

Let $r \coloneqq z(q) \in \mathbb{P}$. Then $g(r) = z(r) = z^{\circ 2}(q)$. We claim that $g(r)$ is *not* periodic for g when $t \geq 3$.

Indeed, for each $j \in \{0, 1, \ldots, t-3\}$, the point $z^{\circ(j+2)}(q)$ is not equal to q, so along this segment g acts exactly as z:

$$g^{\circ j}(g(r)) = g^{\circ j}(z^{\circ 2}(q)) = z^{\circ(j+2)}(q).$$

In particular, taking $j = t - 3$ gives $g^{\circ(t-3)}(g(r)) = z^{\circ(t-1)}(q)$. Since $t \in \{3, \ldots, n\}$ implies $z^{\circ(t-1)}(q) \neq q$, one more application of g still follows z, hence

$$g^{\circ(t-2)}(g(r)) = g\big(z^{\circ(t-1)}(q)\big) = z^{\circ t}(q) = g(q).$$

The point $g(q)$ lies on the unique directed cycle of g:

- If $t < n$, then $q \to z^{\circ t}(q) \to z^{\circ(t+1)}(q) \to \cdots \to z^{\circ(n-1)}(q) \to q$ is a directed cycle.
- If $t = n$, then $g(q) = q$ and q is a fixed point.

When $t \geq 3$, the point $g(r) = z^{\circ 2}(q)$ is not on that cycle; it flows into the cycle after at least one step and therefore cannot be periodic.

Thus there exists $x = r$ such that $g(x) = g(r)$ is *not* periodic, which contradicts $\mathrm{ht}(g) \leq 1$. Therefore $t \not\geq 3$, and the only possibilities are $t \in \{1, 2\}$, giving $g \in \{z, \tilde{z}\}$. □

A.2.3 Which interpolants are k-fold iterates?

Lemma A.3 (z is a k-fold iterate iff $\gcd(k, n) = 1$)**.** *Let $|\mathbb{S}| = n$ and $k \geq 1$. Then*

$$z \in \mathcal{F}_k \quad \Longleftrightarrow \quad \gcd(k, n) = 1.$$

Proof. ($\Rightarrow$) Suppose $z = f^{\circ k}$ for some $f : \mathbb{S} \to \mathbb{S}$. Since z is bijective, $f^{\circ k}$ is bijective, hence f is injective: if $f(x) = f(y)$ then applying $f^{\circ(k-1)}$ gives $f^{\circ k}(x) = f^{\circ k}(y)$, and bijectivity of $f^{\circ k}$ implies $x = y$. On a finite set, injective implies bijective, so f is a permutation of $\mathbb{S}$.

Moreover,

$$f \circ z = f \circ f^{\circ k} = f^{\circ(k+1)} = f^{\circ k} \circ f = z \circ f,$$

so f commutes with z. Let $a \in \{0, 1, \ldots, n-1\}$ be such that $f(0) = a = z^{\circ a}(0)$. Then for every $i \in \mathbb{S}$,

$$f(i) = f(z^{\circ i}(0)) = z^{\circ i}(f(0)) = z^{\circ i}(a) = z^{\circ a}(i),$$

so $f = z^{\circ a}$. Now $f^{\circ k} = z^{\circ(ak)} = z$ implies $ak \equiv 1 \pmod{n}$, which is solvable if and only if $\gcd(k, n) = 1$.

($\Leftarrow$) If $\gcd(k, n) = 1$, let a be the inverse of k modulo n, i.e. $ak \equiv 1 \pmod{n}$. Set $f \coloneqq z^{\circ a}$. Then

$$f^{\circ k} = z^{\circ(ak)} = z^{\circ 1} = z,$$

so $z \in \mathcal{F}_k$. □

Lemma A.4 ($\tilde{z}$ is a k-fold iterate iff $\gcd(k, n-1) = 1$)**.** *Let $\tilde{z}$ be defined by equation 10 and set $L \coloneqq n - 1$. Then for any $k \geq 1$,*

$$\tilde{z} \in \mathcal{F}_k \quad \iff \quad \gcd(k, L) = 1.$$

Proof. Let $r \coloneqq z(q)$ and let $C \coloneqq \mathbb{S} \setminus \{r\}$, so $|C| = L = n - 1$. By construction, $\tilde{z}$ restricts to a single directed cycle on C of length L (it is the z-cycle with the point r removed), and r maps into that cycle:

$$\tilde{z}(r) = z^{\circ 2}(q) \in C.$$

In particular, the periodic points of $\tilde{z}$ are exactly the points of C.

Fact: periodic points are preserved under iterates. For any map f and any $k \geq 1$, a point x is periodic for f if and only if it is periodic for $f^{\circ k}$:

$$f^{\circ m}(x) = x \;\Rightarrow\; (f^{\circ k})^{\circ m}(x) = f^{\circ(km)}(x) = x,$$

and conversely

$$(f^{\circ k})^{\circ m}(x) = x \;\Rightarrow\; f^{\circ(km)}(x) = x.$$

Thus f and $f^{\circ k}$ have exactly the same periodic points.

($\Rightarrow$) Necessity. Assume $\tilde{z} = f^{\circ k}$ for some f. By the fact above, f and $\tilde{z}$ have the same periodic points, so the periodic points of f are exactly C. In particular $f(C) \subseteq C$ and every point of C is periodic for f, so $f|_C$ is a permutation of C.

Since $\tilde{z}|_C = (f|_C)^{\circ k}$ is a *single* cycle of length L, the permutation $f|_C$ must itself be a single cycle of length L (taking powers of a permutation cannot merge distinct cycles). On a cycle of length L, taking the kth power splits the cycle into $\gcd(L, k)$ disjoint cycles. Because $(f|_C)^{\circ k} = \tilde{z}|_C$ is a single cycle, we must have $\gcd(L, k) = 1$.

($\Leftarrow$) Sufficiency. Assume $\gcd(k, L) = 1$. Enumerate the cycle C in the order induced by $\tilde{z}$ as

$$e_0 \coloneqq q, \quad e_1 \coloneqq z^{\circ 2}(q), \quad e_2 \coloneqq z^{\circ 3}(q), \; \ldots, \quad e_{L-1} \coloneqq z^{\circ(n-1)}(q),$$

so that $\tilde{z}(e_i) = e_{i+1 \bmod L}$ for all i.

Let a be the inverse of k modulo L, i.e. $ak \equiv 1 \pmod{L}$, and define f on C by

$$f(e_i) \coloneqq e_{i+a \bmod L}.$$

Then for each $e_i \in C$,

$$f^{\circ k}(e_i) = e_{i+ak \bmod L} = e_{i+1 \bmod L} = \tilde{z}(e_i).$$

It remains to define f on the remaining point $r \notin C$. We want $f^{\circ k}(r) = \tilde{z}(r) = e_1$. Since $f(r)$ will lie in C and f acts on C by $e_i \mapsto e_{i+a}$, it suffices to choose $f(r) = e_j$ so that

$$f^{\circ(k-1)}(e_j) = e_1.$$

But $f^{\circ(k-1)}(e_j) = e_{j+a(k-1) \bmod L}$, so we need

$$j + a(k-1) \equiv 1 \pmod{L}.$$

There is a unique $j \in \{0, 1, \ldots, L-1\}$ satisfying this congruence; fix that j and set $f(r) \coloneqq e_j$. Then

$$f^{\circ k}(r) = f^{\circ(k-1)}(f(r)) = f^{\circ(k-1)}(e_j) = e_{j+a(k-1)} = e_1 = \tilde{z}(r).$$

Thus $f^{\circ k} = \tilde{z}$ on all of $\mathbb{S}$, proving $\tilde{z} \in \mathcal{F}_k$. □

A.2.4 Uniqueness for fixed n, k

Theorem A.5 (Uniqueness of the good interpolant for large iterates)**.** *Fix $n \geq 3$ and a held-out point $q \in \mathbb{S}$. Let $k \geq n-1$ satisfy*

$$\gcd(k, n) = 1 \quad \textit{and} \quad \gcd(k, n-1) > 1.$$

Then

$$\arg\min_{g \in \mathcal{F}_k} L_{\mathcal{P}}(g) = \{z\},$$

and consequently the unique minimizer $\hat{g} = z$ has perfect held-out performance:

$$L_{\mathcal{Q}}(\hat{g}) = 0.$$

Proof. Because $\gcd(k, n) = 1$, Lemma A.3 implies $z \in \mathcal{F}_k$, hence the minimum possible training loss over $\mathcal{F}_k$ is at most $L_{\mathcal{P}}(z) = 0$ and therefore equals 0.

Now let $g \in \mathcal{F}_k$ satisfy $L_{\mathcal{P}}(g) = 0$. Then $g(x) = z(x)$ for all $x \in \mathbb{P}$. Since $g \in \mathcal{F}_k$ and $k \geq n-1$, Lemma A.1 gives $\mathrm{ht}(g) \leq 1$. Lemma A.2 then forces $g \in \{z, \tilde{z}\}$.

Finally, because $\gcd(k, n-1) > 1$, Lemma A.4 implies $\tilde{z} \notin \mathcal{F}_k$. Thus the only hypothesis in $\mathcal{F}_k$ with zero training loss is z, so the minimizer is unique and equals z. Since $\mathcal{Q}$ is the point mass at q, we have $L_{\mathcal{Q}}(z) = \mathbf{1}\{z(q) \neq z(q)\} = 0$. □

A.3 Number theory: densities, totients, and Dirichlet

Lemma A.6 (Densities of multiples and coprimes)**.** *Let $m \geq 1$ be an integer.*

1. *The set $\{k \in \mathbb{N} : m \mid k\}$ has density $1/m$.*
2. *The set $\{k \in \mathbb{N} : \gcd(k, m) = 1\}$ has density $\varphi(m)/m$, where φ is Euler's totient function.*

Proof. Both statements follow from periodicity modulo m. For (1), in each block of m consecutive integers there is exactly one multiple of m. For (2), among the residues $\{1, 2, \ldots, m\}$ there are exactly $\varphi(m)$ residues that are coprime to m, and each block of length m repeats the same pattern. □

Lemma A.7 (Divisibility can only decrease $\varphi(\cdot)/(\cdot)$)**.** *If $M \mid N$, then*

$$\frac{\varphi(N)}{N} \leq \frac{\varphi(M)}{M}.$$

Proof. Using the prime-factor formula,

$$\frac{\varphi(T)}{T} = \prod_{p \mid T} \left(1 - \frac{1}{p}\right).$$

If $M \mid N$, then every prime divisor of M is also a prime divisor of N, so the product defining $\varphi(N)/N$ contains all factors from $\varphi(M)/M$, and possibly additional factors of the form $(1 - 1/p) \leq 1$. Hence $\varphi(N)/N \leq \varphi(M)/M$. □

Lemma A.8 (Making $\varphi(M)/M$ arbitrarily small)**.** *For every $\delta > 0$ there exists an integer $M \geq 2$ such that*

$$\frac{\varphi(M)}{M} < \delta.$$

Proof. Let $p_1 < p_2 < \cdots$ be the primes and define the primorial

$$M_t \coloneqq \prod_{i=1}^{t} p_i.$$

Then

$$\frac{\varphi(M_t)}{M_t} = \prod_{i=1}^{t} \left(1 - \frac{1}{p_i}\right).$$

Using $\log(1-x) \le -x$ for $x \in (0,1)$,

$$\log\left(\prod_{i=1}^{t}\left(1-\frac{1}{p_i}\right)\right) = \sum_{i=1}^{t} \log\left(1-\frac{1}{p_i}\right) \le -\sum_{i=1}^{t}\frac{1}{p_i}.$$

A classical theorem of Euler states that $\sum_{p \text{ prime}} \frac{1}{p}$ diverges, so the RHS tends to $-\infty$, forcing $\prod_{i=1}^{t}(1-1/p_i) \to 0$. Hence for large enough t, $\varphi(M_t)/M_t < \delta$; set $M \coloneqq M_t$. □

Theorem A.9 (Dirichlet's theorem on arithmetic progressions)**.** *If $a, m \in \mathbb{N}$ satisfy $\gcd(a,m) = 1$, then there exist infinitely many primes p such that $p \equiv a \pmod{m}$.*

Lemma A.10 (A density lower bound for "good" exponents)**.** *Let $n \ge 3$ be prime and define*

$$G_n \coloneqq \{k \in \mathbb{N} : \gcd(k,n) = 1 \text{ and } \gcd(k, n-1) > 1\}.$$

Then G_n has density at least

$$\operatorname{dens}(G_n) \;\ge\; 1 - \frac{1}{n} - \frac{\varphi(n-1)}{n-1}.$$

Proof. Because n is prime, $\gcd(k,n) = 1$ fails exactly when $n \mid k$, which has density $1/n$ by Lemma A.6(1). Also $\gcd(k, n-1) > 1$ fails exactly when $\gcd(k, n-1) = 1$, which has density $\varphi(n-1)/(n-1)$ by Lemma A.6(2).

Thus the complement G_n^c is contained in the union

$$\{k : n \mid k\} \;\cup\; \{k : \gcd(k, n-1) = 1\}.$$

Taking densities and using subadditivity (a union bound) yields

$$\operatorname{dens}(G_n^c) \;\le\; \frac{1}{n} + \frac{\varphi(n-1)}{n-1}.$$

Therefore $\operatorname{dens}(G_n) = 1 - \operatorname{dens}(G_n^c) \ge 1 - \frac{1}{n} - \frac{\varphi(n-1)}{n-1}$. □

A.4 High-density uniqueness for the cycle task via large iterates

Fix $\varepsilon > 0$. We will choose a prime n such that

$$\frac{1}{n} < \frac{\varepsilon}{2} \qquad \text{and} \qquad \frac{\varphi(n-1)}{n-1} < \frac{\varepsilon}{2}. \tag{11}$$

Then Lemma A.10 will imply $\operatorname{dens}(G_n) \ge 1 - \varepsilon$. Finally, Theorem A.5 will transfer this density statement to the uniqueness property.

Step 1: choose M with $\varphi(M)/M < \varepsilon/2$. By Lemma A.8, choose an integer $M \ge 2$ such that

$$\frac{\varphi(M)}{M} < \frac{\varepsilon}{2}.$$

Step 2: use Dirichlet to pick a large prime $n \equiv 1 \pmod{M}$. By Dirichlet's theorem (Theorem A.9) with $a = 1$ and modulus $m = M$, there exist infinitely many primes n with

$$n \equiv 1 \pmod{M}.$$

Choose such a prime n large enough that $1/n < \varepsilon/2$. Then $M \mid (n-1)$.

Step 3: bound $\varphi(n-1)/(n-1)$. Since $M \mid (n-1)$, Lemma A.7 gives

$$\frac{\varphi(n-1)}{n-1} \leq \frac{\varphi(M)}{M} < \frac{\varepsilon}{2}.$$

Together with $1/n < \varepsilon/2$, this establishes equation 11.

Step 4: conclude a density-$\geq 1-\varepsilon$ set of "good" exponents. Define

$$G_n = \{k \in \mathbb{N} : \gcd(k, n) = 1 \text{ and } \gcd(k, n-1) > 1\}.$$

By Lemma A.10 and equation 11,

$$\operatorname{dens}(G_n) \geq 1 - \frac{1}{n} - \frac{\varphi(n-1)}{n-1} \;\geq\; 1 - \varepsilon.$$

Since removing finitely many integers does not change density, the same bound holds if we restrict to $k \geq n-1$:

$$\operatorname{dens}(\{k \geq n-1 :\; k \in G_n\}) = \operatorname{dens}(G_n) \geq 1 - \varepsilon.$$

Step 5: transfer "good gcd" to uniqueness. For any $k \geq n-1$ with $k \in G_n$, we have $\gcd(k, n) = 1$, and $\gcd(k, n-1) > 1$. Therefore Theorem A.5 applies and yields

$$\arg\min_{g \in \mathcal{F}_k} L_{\mathcal{P}}(g) = \{z\}.$$

Consequently the unique minimizer is $\hat{g} = z$, and since $\mathcal{Q}$ is a point mass at q we have $L_{\mathcal{Q}}(\hat{g}) = L_{\mathcal{Q}}(z) = 0$.

Thus for at least a $(1-\varepsilon)$ fraction of integers $k \geq n-1$, the uniqueness property holds.

B Proof of Theorem 3.3

B.1 Explicit Form of the Infinite-Context Limit

For each n, let $\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \in \mathbb{R}^{d_e \times (n+1)}$ denote the input sequence constructed from n samples $\{(\boldsymbol{x}_i, y_i)\}_{i=1}^n$ via equation 1. Let $\boldsymbol{X} = [\boldsymbol{x}_1, \ldots, \boldsymbol{x}_n] \in \mathbb{R}^{d \times n}$ and $\boldsymbol{y} = (y_1, \ldots, y_n)^\top \in \mathbb{R}^n$. Define the extracted coefficient vector at generation step t by

$$\hat{\boldsymbol{w}}^{(n)}_t \;:=\; \operatorname{Extract}\Big(\operatorname{Auto}^{\circ t}\big(\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \mid \boldsymbol{V}, \boldsymbol{W}\big)\Big) \in \mathbb{R}^d,$$

so that $\hat{\boldsymbol{w}}^{(n)}_k = f_k(\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \mid \boldsymbol{V}, \boldsymbol{W})$ by definition of f_k.

A direct block-matrix calculation using the sparsity of $\tilde{\boldsymbol{V}}, \tilde{\boldsymbol{W}}$ shows that, for every $t \geq 0$,

$$\hat{\boldsymbol{w}}^{(n)}_{t+1} = \hat{\boldsymbol{w}}^{(n)}_t + \boldsymbol{V}\Big(\widehat{\boldsymbol{\Sigma}}^{(n)}\, \boldsymbol{W}\, \hat{\boldsymbol{w}}^{(n)}_t - \widehat{\mathbf{u}}^{(n)}\Big), \qquad \hat{\boldsymbol{w}}^{(n)}_0 = \mathbf{0}, \tag{12}$$

where

$$\widehat{\boldsymbol{\Sigma}}^{(n)} := \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i \boldsymbol{x}_i^\top = \frac{1}{n}\boldsymbol{X}\boldsymbol{X}^\top, \qquad \widehat{\mathbf{u}}^{(n)} := \frac{1}{n}\sum_{i=1}^n \boldsymbol{x}_i y_i = \frac{1}{n}\boldsymbol{X}\boldsymbol{y}.$$

In particular, conditioned on the dataset, the induced update depends on the samples only through the empirical moments $\widehat{\boldsymbol{\Sigma}}^{(n)}$ and $\widehat{\mathbf{u}}^{(n)}$.

Unrolling the recursion equation 12 yields the closed form

$$\hat{\boldsymbol{w}}^{(n)}_k = -\sum_{t=0}^{k-1} \Big(\boldsymbol{I} + \boldsymbol{V}\widehat{\boldsymbol{\Sigma}}^{(n)}\boldsymbol{W}\Big)^t \boldsymbol{V}\, \widehat{\mathbf{u}}^{(n)}. \tag{13}$$

For fixed $(\boldsymbol{V}, \boldsymbol{W})$ and k, define the deterministic map

$$g_k(\boldsymbol{\Sigma}, \mathbf{u}) \;:=\; -\sum_{t=0}^{k-1} \Big(\boldsymbol{I} + \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}\Big)^t \boldsymbol{V}\, \mathbf{u}, \qquad (\boldsymbol{\Sigma}, \mathbf{u}) \in \mathbb{R}^{d \times d} \times \mathbb{R}^d.$$

Then equation 13 can be written as

$$\hat{\boldsymbol{w}}_k^{(n)} = g_k\big(\widehat{\boldsymbol{\Sigma}}^{(n)}, \widehat{\mathbf{u}}^{(n)}\big).$$

Since k is fixed, g_k is obtained from its inputs via finitely many additions and multiplications (matrix products and sums in a fixed order); equivalently, each coordinate of $g_k(\boldsymbol{\Sigma}, \mathbf{u})$ is a multivariate polynomial in the entries of $(\boldsymbol{\Sigma}, \mathbf{u})$. In particular, g_k is continuous. Therefore, if $(\widehat{\boldsymbol{\Sigma}}^{(n)}, \widehat{\mathbf{u}}^{(n)}) \to (\boldsymbol{\Sigma}, \mathbf{u})$ almost surely, then $g_k(\widehat{\boldsymbol{\Sigma}}^{(n)}, \widehat{\mathbf{u}}^{(n)}) \to g_k(\boldsymbol{\Sigma}, \mathbf{u})$ almost surely (by the Continuous Mapping Theorem).

Under our data model $\boldsymbol{x}_i \sim \mathcal{N}(0, \boldsymbol{\Sigma})$ and $y_i = \boldsymbol{x}_i^\top \boldsymbol{\beta}$, we have almost surely

$$\widehat{\boldsymbol{\Sigma}}^{(n)} \to \boldsymbol{\Sigma}, \qquad \widehat{\mathbf{u}}^{(n)} = \frac{1}{n}\sum_{i=1}^{n} \boldsymbol{x}_i \boldsymbol{x}_i^\top \boldsymbol{\beta} \to \boldsymbol{\Sigma}\boldsymbol{\beta}, \qquad (n \to \infty).$$

Combining these facts yields the almost sure limit

$$\lim_{n\to\infty} f_k\big(\boldsymbol{S}^{(n)}_{\boldsymbol{\Sigma},\boldsymbol{\beta}} \mid \boldsymbol{V}, \boldsymbol{W}\big) = g_k(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}\boldsymbol{\beta}) = -\sum_{t=0}^{k-1} \Big(\boldsymbol{I} + \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}\Big)^t \boldsymbol{V}\,\boldsymbol{\Sigma}\,\boldsymbol{\beta}. \tag{14}$$

Equivalently, writing $\hat{\boldsymbol{w}}_k := g_k(\boldsymbol{\Sigma}, \boldsymbol{\Sigma}\boldsymbol{\beta})$ and letting $\boldsymbol{A} := \boldsymbol{I} + \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}$,

$$\hat{\boldsymbol{w}}_k - \boldsymbol{\beta} = -\Big(\sum_{t=0}^{k-1} \boldsymbol{A}^t\,\boldsymbol{V}\boldsymbol{\Sigma} + \boldsymbol{I}\Big)\boldsymbol{\beta}. \tag{15}$$

B.2 Characterizing in-distribution minimizers and convergence to GD

We repeatedly invoke the finite-n update equation 12 and the population-limit expression equation 14–equation 15 established in Subsection B.1.

Step 1: Characterizing generic GD-implementing parameters. Recall the definition of Θ_{GD} from equation 16: we say that $(\boldsymbol{V}, \boldsymbol{W})$ implements one-step GD if there exists $\eta > 0$ such that, for every dataset (equivalently, for all realizable empirical moments $(\widehat{\boldsymbol{\Sigma}}, \widehat{\mathbf{u}})$) and every iterate $\boldsymbol{w} \in \mathbb{R}^d$, the induced update equals the GD update.

By equation 12, the transformer-induced one-step update has the form

$$U_{\boldsymbol{V},\boldsymbol{W}}(\boldsymbol{w}; \widehat{\boldsymbol{\Sigma}}, \widehat{\mathbf{u}}) := \boldsymbol{w} + \boldsymbol{V}\big(\widehat{\boldsymbol{\Sigma}}\boldsymbol{W}\boldsymbol{w} - \widehat{\mathbf{u}}\big),$$

whereas one-step GD on the squared loss is

$$U_{\mathrm{GD},\eta}(\boldsymbol{w}; \widehat{\boldsymbol{\Sigma}}, \widehat{\mathbf{u}}) := \boldsymbol{w} - \eta\big(\widehat{\boldsymbol{\Sigma}}\boldsymbol{w} - \widehat{\mathbf{u}}\big).$$

If $U_{\boldsymbol{V},\boldsymbol{W}} \equiv U_{\mathrm{GD},\eta}$ holds for all $\boldsymbol{w}$ and all $(\widehat{\boldsymbol{\Sigma}}, \widehat{\mathbf{u}})$, then comparing the coefficients of $\widehat{\mathbf{u}}$ forces $-\boldsymbol{V} = \eta\boldsymbol{I}_d$, i.e., $\boldsymbol{V} = -\eta\boldsymbol{I}_d$. Substituting this into the coefficient of $\widehat{\boldsymbol{\Sigma}}\boldsymbol{w}$ yields

$$\boldsymbol{V}\widehat{\boldsymbol{\Sigma}}\boldsymbol{W} = -\eta\widehat{\boldsymbol{\Sigma}} \quad \text{for all } \widehat{\boldsymbol{\Sigma}},$$

which implies $\boldsymbol{W} = \boldsymbol{I}_d$. Hence

$$\Theta_{\mathrm{GD}} = \Big\{\, (-\eta\boldsymbol{I}_d,\ \boldsymbol{I}_d) \;\Big|\; \eta > 0 \,\Big\}. \tag{16}$$

Step 2: Infinite-context error under isotropic $\boldsymbol{\Sigma} = \gamma\boldsymbol{I}_d$. We now work in the infinite-context limit. By equation 15, for any covariance $\boldsymbol{\Sigma}$ we have

$$\hat{\boldsymbol{w}}_k - \boldsymbol{\beta} = -\Big(\sum_{t=0}^{k-1} \boldsymbol{A}^t\,\boldsymbol{V}\boldsymbol{\Sigma} + \boldsymbol{I}\Big)\boldsymbol{\beta}, \qquad \boldsymbol{A} := \boldsymbol{I} + \boldsymbol{V}\boldsymbol{\Sigma}\boldsymbol{W}.$$

Specializing to the isotropic case $\boldsymbol{\Sigma} = \gamma\boldsymbol{I}_d$ yields

$$\hat{\boldsymbol{w}}_k - \boldsymbol{\beta} = -\boldsymbol{A}_\gamma(\boldsymbol{V}, \boldsymbol{W})\,\boldsymbol{\beta}, \qquad \boldsymbol{A}_\gamma(\boldsymbol{V}, \boldsymbol{W}) := \sum_{t=0}^{k-1} (\boldsymbol{I} + \gamma\boldsymbol{V}\boldsymbol{W})^t\,\gamma\boldsymbol{V} + \boldsymbol{I}. \tag{17}$$

Since $\boldsymbol{\beta} \sim \mathcal{N}(0, \boldsymbol{I}_d)$, we have $\mathbb{E}_{\boldsymbol{\beta}}\|\boldsymbol{A}_\gamma\boldsymbol{\beta}\|^2 = \|\boldsymbol{A}_\gamma\|_F^2$. Therefore, under $\mathcal{P}$ (uniform over $\gamma \in \{\gamma_1, \gamma_2\}$),

$$L_{\mathcal{P}}(f_k) = \frac{1}{2}\|\boldsymbol{A}_{\gamma_1}(\boldsymbol{V}, \boldsymbol{W})\|_F^2 + \frac{1}{2}\|\boldsymbol{A}_{\gamma_2}(\boldsymbol{V}, \boldsymbol{W})\|_F^2. \tag{18}$$

Step 3: Reduce to a scalar problem via simultaneous orthogonal diagonalizability. Recall that we only consider $\boldsymbol{V}, \boldsymbol{W}$ that are simultaneously orthogonally diagonalizable over $\mathbb{R}$, i.e., there exists an orthogonal matrix $\boldsymbol{U}$ such that

$$\boldsymbol{V} = \boldsymbol{U}\mathbf{diag}(v_1, \ldots, v_d)\boldsymbol{U}^\top, \qquad \boldsymbol{W} = \boldsymbol{U}\mathbf{diag}(w_1, \ldots, w_d)\boldsymbol{U}^\top.$$

Since $\boldsymbol{A}_\gamma(\boldsymbol{V}, \boldsymbol{W})$ is a matrix polynomial in $\boldsymbol{V}$ and $\boldsymbol{W}$, it is diagonalized by the same $\boldsymbol{U}$:

$$\boldsymbol{A}_\gamma(\boldsymbol{V}, \boldsymbol{W}) = \boldsymbol{U}\mathbf{diag}(A_{\gamma,1}, \ldots, A_{\gamma,d})\boldsymbol{U}^\top,$$

where, for each $i \in [d]$,

$$A_{\gamma,i} = 1 + \gamma v_i \sum_{t=0}^{k-1} (1 + \gamma v_i w_i)^t = 1 + \frac{(1 + \gamma v_i w_i)^k - 1}{w_i}. \tag{19}$$

The last equation is for $w_i \neq 0$. Note that, if $w_i = 0$, then $A_{\gamma,i} = 1 + \gamma v_i \sum_{t=0}^{k-1} 1 = 1 + k\gamma v_i$, which cannot vanish simultaneously for $\gamma = \gamma_1$ and $\gamma = \gamma_2$ since $\gamma_1 \neq \gamma_2$. Hence any zero-loss solution must satisfy $w_i \neq 0$. Because $\boldsymbol{U}$ is orthogonal, $\|\boldsymbol{A}_\gamma\|_F^2 = \sum_{i=1}^d A_{\gamma,i}^2$, and equation 18 becomes

$$L_{\mathcal{P}}(f_k) = \frac{1}{2}\sum_{i=1}^d A_{\gamma_1,i}^2 + \frac{1}{2}\sum_{i=1}^d A_{\gamma_2,i}^2, \tag{20}$$

so the minimization decouples across coordinates.

Step 4: Solve the per-coordinate minimization problem (even k). Fix i and write $v = v_i$, $w = w_i$. Achieving $L_{\mathcal{P}}(f_k) = 0$ requires $A_{\gamma_1}(v, w) = A_{\gamma_2}(v, w) = 0$, which is equivalent to

$$(1 + \gamma vw)^k = 1 - w \quad \text{for } \gamma \in \{\gamma_1, \gamma_2\}.$$

In particular, $(1 + \gamma_1 vw)^k = (1 + \gamma_2 vw)^k$. Let $x := 1 + \gamma_1 vw$ and $y := 1 + \gamma_2 vw$. Since k is even and $x, y \in \mathbb{R}$, the equality $x^k = y^k$ implies $|x| = |y|$, hence $x = \pm y$.

If $x = y$, then $(\gamma_1 - \gamma_2)vw = 0$. Since $\gamma_1 \neq \gamma_2$, this forces $vw = 0$, and substituting into $(1 + \gamma vw)^k = 1 - w$ gives $1 = 1 - w$, i.e., $w = 0$. As noted in Step 3, the case $w = 0$ cannot yield zero loss at both γ_1 and γ_2. Therefore the case $x = y$ is impossible, and we must have $x = -y$, i.e.,

$$1 + \gamma_1 vw = -(1 + \gamma_2 vw).$$

Solving gives $vw = -1/a$, where $a := (\gamma_1 + \gamma_2)/2$. Writing $b := |\gamma_1 - \gamma_2|/2$ and using $\gamma_1 = a + b$ and $\gamma_2 = a - b$ then yields

$$w = 1 - \left(\frac{b}{a}\right)^k, \qquad v = -\frac{1}{a\,(1 - (b/a)^k)}.$$

Thus the pair (v_i, w_i) is uniquely determined and identical for all i, and consequently every in-distribution minimizer $f_k^\star$ must have parameters

$$\boldsymbol{W}^\star = \left(1 - (b/a)^k\right)\boldsymbol{I}_d, \qquad \boldsymbol{V}^\star = -\frac{1}{a\,(1 - (b/a)^k)}\,\boldsymbol{I}_d, \tag{21}$$

which proves item (1).

Step 5: Compute the OOD loss and monotonicity. Plugging equation 21 into equation 19 with $\gamma = \gamma_{\mathcal{Q}}$ gives

$$A_{\gamma_{\mathcal{Q}}} = \frac{(a - \gamma_{\mathcal{Q}})^k - b^k}{a^k - b^k}, \qquad L_{\mathcal{Q}}(f_k^\star) = d\,A_{\gamma_{\mathcal{Q}}}^2 = d\left(\frac{(a - \gamma_{\mathcal{Q}})^k - b^k}{a^k - b^k}\right)^2.$$

To show strict monotonicity over even k, define

$$r := \left|\frac{a - \gamma_{\mathcal{Q}}}{a}\right|, \qquad s := \frac{b}{a}.$$

Under the assumption $\gamma_{\mathcal{Q}} \in (0, 2a) \setminus \{\gamma_1, \gamma_2\}$ we have $r, s \in (0, 1)$ and $r \neq s$. For even k (so $r^k, s^k > 0$),

$$|A_{\gamma_{\mathcal{Q}}}| = \left| \frac{r^k - s^k}{1 - s^k} \right|.$$

It suffices to show that the function

$$\phi(t) := \left| \frac{r^t - s^t}{1 - s^t} \right|, \qquad t > 0,$$

is strictly decreasing. Without loss of generality assume $r > s$ (the case $r < s$ is symmetric by absolute value). Write $r = e^{-\alpha}$ and $s = e^{-\beta}$ with $0 < \alpha < \beta$. Then

$$\phi(t) = \frac{e^{-\alpha t} - e^{-\beta t}}{1 - e^{-\beta t}}.$$

A direct derivative calculation gives

$$\phi'(t) = -\frac{\alpha(e^{\beta t} - 1) - \beta(e^{\alpha t} - 1)}{(e^{\beta t} - 1)^2} \, e^{(\beta - \alpha)t}.$$

Since $x \mapsto \frac{e^{xt} - 1}{x}$ is strictly increasing for $x > 0$ (for any fixed $t > 0$), and $\beta > \alpha$, we have $\alpha(e^{\beta t} - 1) > \beta(e^{\alpha t} - 1)$, hence the numerator is positive and thus $\phi'(t) < 0$. Therefore $\phi(t)$ (and hence $|A_{\gamma_{\mathcal{Q}}}|$) decreases strictly with t, implying that $L_{\mathcal{Q}}(f_k^\star) = dA_{\gamma_{\mathcal{Q}}}^2$ decreases strictly as k increases over even integers.

Step 6: Distance to the GD-implementing family. Let $(\boldsymbol{V}^\star, \boldsymbol{W}^\star)$ be as in equation 21. By equation 16, the closest point in Θ_{GD} matches $\boldsymbol{V}^\star$ (take $\eta = 1/[a(1 - (b/a)^k)]$) and sets $\boldsymbol{W}' = \boldsymbol{I}_d$, hence

$$\operatorname{dist}\left((\boldsymbol{V}^\star, \boldsymbol{W}^\star), \Theta_{\mathrm{GD}}\right) = \|\boldsymbol{W}^\star - \boldsymbol{I}_d\|_F^2 = d\left(\frac{b}{a}\right)^{2k},$$

proving item (3). □

C Proof of Theorem 3.5

Proof. Fix k. Let $(\hat{g}, \hat{h})$ be any minimizer of equation 3, and define $\hat{f} := \hat{g} + \hat{h}$.

(1) ID upper bound. Recall

$$\delta_k := \min_{h \in \mathcal{H}_k} \Big(L_{\mathcal{P}}(h) + R_H(h) \Big).$$

Since $0 \in \mathcal{G}_k$, for any $h \in \mathcal{H}_k$ the pair $(0, h)$ is feasible for equation 3, and $R(0, h) = R_G(0) + R_H(h) = R_H(h)$ because $R_G(0) = 0$. By optimality of $(\hat{g}, \hat{h})$,

$$L_{\mathcal{P}}(\hat{f}) + R(\hat{g}, \hat{h}) \;\leq\; L_{\mathcal{P}}(h) + R(0, h) \;=\; L_{\mathcal{P}}(h) + R_H(h) \qquad \forall\, h \in \mathcal{H}_k.$$

Taking the minimum over $h \in \mathcal{H}_k$ yields

$$L_{\mathcal{P}}(\hat{f}) + R(\hat{g}, \hat{h}) \leq \delta_k.$$

Since $R(\hat{g}, \hat{h}) = R_G(\hat{g}) + R_H(\hat{h}) \geq 0$, we conclude

$$L_{\mathcal{P}}(\hat{f}) \leq \delta_k.$$

(2) OOD lower bound. Define the residual associated with $\hat{g}$:

$$\hat{r} := z - \hat{g}.$$

Then

$$L_{\mathcal{Q}}(\hat{f}) = \mathbb{E}_{x \sim \mathcal{Q}} \|\hat{f}(x) - z(x)\|^2 = \mathbb{E}_{x \sim \mathcal{Q}} \|\hat{g}(x) + \hat{h}(x) - z(x)\|^2 = \mathbb{E}_{x \sim \mathcal{Q}} \|\hat{h}(x) - \hat{r}(x)\|^2.$$

By Assumption 3.4 (Shortcut part, equation 5), applied to the minimizer $(\hat{g}, \hat{h})$ of equation 3,

$$\mathbb{E}_{x \sim \mathcal{Q}} \|\hat{h}(x) - \hat{r}(x)\|^2 \;\geq\; C\, \mathbb{E}_{x \sim \mathcal{Q}} \|\hat{r}(x)\|^2 = C\, L_{\mathcal{Q}}(\hat{g}).$$

Using Assumption 3.4 (Well-structured part, equation 4), $L_{\mathcal{Q}}(\hat{g}) \geq \frac{1}{\gamma} L_{\mathcal{P}}(\hat{g})$. Finally, since $\hat{g} \in \mathcal{G}_k$, $L_{\mathcal{P}}(\hat{g}) \geq \min_{g \in \mathcal{G}_k} L_{\mathcal{P}}(g) = \epsilon_k$. Therefore,

$$L_{\mathcal{Q}}(\hat{f}) \geq \frac{C}{\gamma}\, \epsilon_k = C_1\, \epsilon_k.$$

(3) OOD upper bound. Let $g_k^\star \in \arg\min_{g\in\mathcal{G}_k} L_{\mathcal{P}}(g)$ be as in Assumption 3.4(ii), so $L_{\mathcal{P}}(g_k^\star) = \epsilon_k$ and $R_G(g_k^\star) \le \rho_k$. Since $0 \in \mathcal{H}_k$, the pair $(g_k^\star, 0)$ is feasible for equation 3, and $R(g_k^\star, 0) = R_G(g_k^\star) + R_H(0) = R_G(g_k^\star)$ because $R_H(0) = 0$. By optimality of $(\hat{g}, \hat{h})$,

$$L_{\mathcal{P}}(\hat{f}) + R_G(\hat{g}) + R_H(\hat{h}) \;\le\; L_{\mathcal{P}}(g_k^\star) + R_G(g_k^\star) \;\le\; \epsilon_k + \rho_k. \tag{22}$$

Dropping the nonnegative term $R_G(\hat{g})$ gives

$$L_{\mathcal{P}}(\hat{f}) \le \epsilon_k + \rho_k, \qquad R_H(\hat{h}) \le \epsilon_k + \rho_k. \tag{23}$$

We now bound $L_{\mathcal{Q}}(\hat{f})$. Using $\|u+v\|^2 \le 2\|u\|^2 + 2\|v\|^2$,

$$L_{\mathcal{Q}}(\hat{f}) = \mathbb{E}_{x\sim\mathcal{Q}}\|(\hat{g}(x) - z(x)) + \hat{h}(x)\|^2 \le 2L_{\mathcal{Q}}(\hat{g}) + 2\,\mathbb{E}_{x\sim\mathcal{Q}}\|\hat{h}(x)\|^2.$$

By equation 4, $L_{\mathcal{Q}}(\hat{g}) \le \gamma L_{\mathcal{P}}(\hat{g})$. Also,

$$L_{\mathcal{P}}(\hat{g}) = \mathbb{E}_{x\sim\mathcal{P}}\|(\hat{f}(x) - z(x)) - \hat{h}(x)\|^2 \le 2L_{\mathcal{P}}(\hat{f}) + 2\,\mathbb{E}_{x\sim\mathcal{P}}\|\hat{h}(x)\|^2.$$

By Assumption 3.4(i) (Norm control, equation 6),

$$\mathbb{E}_{x\sim\mathcal{P}}\|\hat{h}(x)\|^2 \le a_{\mathcal{P}} R_H(\hat{h}), \qquad \mathbb{E}_{x\sim\mathcal{Q}}\|\hat{h}(x)\|^2 \le a_{\mathcal{Q}} R_H(\hat{h}).$$

Combining these bounds,

$$\begin{aligned} L_{\mathcal{Q}}(\hat{f}) &\le 2\gamma\Big(2L_{\mathcal{P}}(\hat{f}) + 2a_{\mathcal{P}} R_H(\hat{h})\Big) \;+\; 2a_{\mathcal{Q}} R_H(\hat{h}) \\ &= 4\gamma\, L_{\mathcal{P}}(\hat{f}) + \big(4\gamma a_{\mathcal{P}} + 2a_{\mathcal{Q}}\big) R_H(\hat{h}). \end{aligned}$$

Applying equation 23 yields

$$L_{\mathcal{Q}}(\hat{f}) \le \big(4\gamma + 4\gamma a_{\mathcal{P}} + 2a_{\mathcal{Q}}\big)\,(\epsilon_k + \rho_k) = C_2\,(\epsilon_k + \rho_k).$$

Combining (1)–(3) completes the proof. □

D Proof of Proposition 3.6

Proof. We work on the support $\mathcal{U} = \{\pm 1\}^p$ via $u = \pi(x) = x_{1:p}$. Recall $\mathcal{P} = \mathrm{Unif}(\mathcal{U}_P)$, $\mathcal{Q} = \mathrm{Unif}(\mathcal{U}_Q)$, and $|\mathcal{U}_P| = |\mathcal{U}_Q| = M = 2^{p-1}$. The target is $z(u) = u^\top\theta$ with $\theta \in \mathbb{R}^p_{\ge 0}$.

1) Distribution-shift control on $\mathcal{G}_k$ with $\gamma = 1$, and the value of ϵ_k. For $a(u) = \mathrm{diag}(w_1, \dots, w_p)u$ with $0 \le w_i \le \tau$, we have $a^{\circ k}(u) = \mathrm{diag}(w_1^k, \dots, w_p^k)u$. With $\psi_1(v) = \mathbf{1}^\top v$, any $g \in \mathcal{G}_k$ can be written as

$$g(u) = \mathbf{1}^\top a^{\circ k}(u) = \sum_{i=1}^p \alpha_i u_i, \qquad \alpha_i := w_i^k \in [0, \tau^k].$$

Hence

$$(g(u) - z(u))^2 = \Big(\sum_{i=1}^p (\alpha_i - \theta_i)u_i\Big)^2.$$

Under both $\mathcal{P}$ and $\mathcal{Q}$, the coordinates satisfy $\mathbb{E}[u_i u_j] = 0$ for $i \ne j$ and $\mathbb{E}[u_i^2] = 1$ for all i (note $u_1 \equiv \pm 1$ but still $u_1^2 = 1$ and $\mathbb{E}[u_1 u_j] = \mathbb{E}[u_j] = 0$ for $j \ge 2$). Therefore,

$$L_{\mathcal{P}}(g) = \sum_{i=1}^p (\alpha_i - \theta_i)^2 \;=\; L_{\mathcal{Q}}(g),$$

so equation 4 holds with $\gamma = 1$.

To compute ϵ_k, we minimize $\sum_{i=1}^p (\alpha_i - \theta_i)^2$ subject to $\alpha_i \in [0, \tau^k]$, which yields $\alpha_i^\star = \min\{\theta_i, \tau^k\}$. Hence

$$\epsilon_k = \min_{g\in\mathcal{G}_k} L_{\mathcal{P}}(g) = \sum_{i=1}^p (\theta_i - \min\{\theta_i, \tau^k\})^2 = \sum_{i=1}^p \big((\theta_i - \tau^k)_+\big)^2.$$

2) Norm control on $\mathcal{H}_k$ with $a_{\mathcal{P}} = a_{\mathcal{Q}} = 1/(\lambda M)$. Here $l = 1$ and $\psi_2(t) = t$, so any $h \in \mathcal{H}_k$ is of the form $h(u) = b(u)$ for some $b \in \mathcal{B}$. For the Kronecker delta kernel $K(u, u') = \mathbf{1}\{u = u'\}$ on the finite set $\mathcal{U}$, the RKHS norm satisfies

$$\|b\|_{\mathcal{B}}^2 = \sum_{u \in \mathcal{U}} b(u)^2.$$

Moreover, the instantiation defines $R_H(h) \coloneqq \lambda \|b\|_{\mathcal{B}}^2$. Thus,

$$\mathbb{E}_{u \sim \mathcal{P}} |h(u)|^2 = \frac{1}{M} \sum_{u \in \mathcal{U}_P} b(u)^2 \leq \frac{1}{M} \sum_{u \in \mathcal{U}} b(u)^2 = \frac{1}{M} \|b\|_{\mathcal{B}}^2 = \frac{1}{\lambda M} R_H(h).$$

The same argument holds for $\mathcal{Q}$. Hence equation 6 holds with $a_{\mathcal{P}} = a_{\mathcal{Q}} = 1/(\lambda M)$.

3) Bounded regularization for a structured fit and the value of ρ_k. Under the instantiation, $R_G(g) \coloneqq \lambda \|a\|^2$ with $\|a\|^2 = \sum_{i=1}^p w_i^2$. Let $g_k^\star$ be the minimizer achieving $\alpha_i^\star = \min\{\theta_i, \tau^k\}$ above, and choose $w_i^\star \coloneqq (\alpha_i^\star)^{1/k} \in [0, \tau]$. Then $g_k^\star \in \mathcal{G}_k$ and $L_{\mathcal{P}}(g_k^\star) = \epsilon_k$. Moreover,

$$R_G(g_k^\star) = \lambda \sum_{i=1}^p (w_i^\star)^2 = \lambda \sum_{i=1}^p \min\{\theta_i^{2/k}, \tau^2\} \leq \lambda \sum_{i=1}^p \theta_i^{2/k}.$$

Thus Assumption 3.4(ii) holds with $\rho_k = \lambda \sum_{i=1}^p \theta_i^{2/k}$.

4) Value of δ_k. Recall $\delta_k = \min_{h \in \mathcal{H}_k} \big(L_{\mathcal{P}}(h) + R_H(h)\big)$. Write $h(u) = b(u)$. Since $L_{\mathcal{P}}$ depends only on values on $\mathcal{U}_P$, and R_H penalizes $\sum_{u \in \mathcal{U}} b(u)^2$, any minimizer satisfies $b(u) = 0$ for all $u \notin \mathcal{U}_P$. Thus the problem reduces to

$$\min_{\{b(u)\}_{u \in \mathcal{U}_P}} \frac{1}{M} \sum_{u \in \mathcal{U}_P} \big(b(u) - z(u)\big)^2 + \lambda \sum_{u \in \mathcal{U}_P} b(u)^2.$$

This decouples over $u \in \mathcal{U}_P$. For each u, minimizing $\frac{1}{M}(b - z)^2 + \lambda b^2$ gives $b^\star = \frac{z}{1 + \lambda M}$. Plugging back yields the optimal value

$$\delta_k = \frac{\lambda}{1 + \lambda M} \sum_{u \in \mathcal{U}_P} z(u)^2.$$

Finally, under $u \sim \mathrm{Unif}(\mathcal{U}_P)$, $\mathbb{E}[z(u)^2] = \theta^\top \mathbb{E}[uu^\top]\theta = \|\theta\|_2^2$, and hence $\sum_{u \in \mathcal{U}_P} z(u)^2 = M\|\theta\|_2^2$. Therefore,

$$\delta_k = \frac{\lambda M}{1 + \lambda M} \|\theta\|_2^2.$$

5) Shortcut failure on $\mathcal{H}_k$ with $C = 1$. Let $(\hat{g}, \hat{h})$ be any minimizer of equation 3, and write $\hat{h}(u) = \hat{b}(u)$. As above, the objective depends on $\hat{b}(u)$ for $u \notin \mathcal{U}_P$ only through the penalty term $\lambda \sum_{u \in \mathcal{U}} \hat{b}(u)^2$. Therefore, for any minimizer we must have $\hat{b}(u) = 0$ for all $u \notin \mathcal{U}_P$, and in particular $\hat{h}(u) = 0$ for all $u \in \mathcal{U}_Q$. Define $\hat{r} \coloneqq z - \hat{g}$. Then on $\mathcal{Q}$,

$$\hat{h}(u) - \hat{r}(u) = -\hat{r}(u), \qquad u \in \mathcal{U}_Q,$$

and hence

$$\mathbb{E}_{u \sim \mathcal{Q}} |\hat{h}(u) - \hat{r}(u)|^2 = \mathbb{E}_{u \sim \mathcal{Q}} |\hat{r}(u)|^2.$$

Thus equation 5 holds with $C = 1$.

Combining the above verifications establishes Assumption 3.4 with the stated constants and the claimed expressions for $\epsilon_k, \rho_k, \delta_k$. □

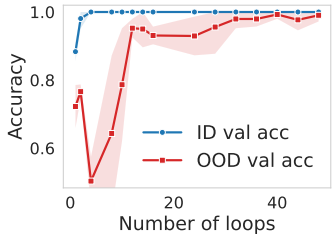


Figure 4: Results for looped Transformers on the 4-hop task with loop embeddings show a pattern consistent with Figure 1. ID accuracy saturates around 2 loops, while OOD accuracy continues improving up to around 36 loops.

E Experimental Details

E.1 Additional Details for Section 4.1

E.1.1 Description of the p-hop Problem

Formally, fix a finite alphabet Σ and a sequence $v \in \Sigma^n$. Define the *one-step backtracking* map

$$\mathrm{find}_1(v, i) \;\coloneqq\; \max\Big(\{0\} \cup \{\, j \le i : \; v_{j-1} = v_i \,\}\Big),$$

and for $p \ge 2$ define the p-step backtracking recursively by

$$\mathrm{find}_p(v, i) \;\coloneqq\; \mathrm{find}_1\big(v, \mathrm{find}_{p-1}(v, i)\big).$$

The p-hop target is then defined as $\mathrm{hop}_p(v) \coloneqq v_{\mathrm{find}_p(v,n)}$, and in our data generation we ensure that the p-hop always exists (i.e., $\mathrm{find}_p(v, n) \neq 0$). Intuitively, the task asks the model to recursively backtrack through the sequence for p steps according to the find_1 rule, and then output the symbol found at the resulting location.

Following [34], we generate sequences by first sampling a valid p-hop chain and then filling the remaining positions with random *filler* symbols while respecting the p-hop constraints.

In our experiments, we set $p = 4$, the vocabulary size to 4, and the sequence length to 12.

E.1.2 Model and Training Details

Architecture. We use a looped Transformer classifier operating on fixed-length token sequences. Each input sequence is embedded using learned token embeddings and learned positional embeddings. The backbone consists of a single Transformer block, comprising multi-head self-attention followed by a two-layer MLP with GELU activations, residual connections, and layer normalization. To scale latent reasoning depth, the backbone is applied repeatedly for $k \in \{1, 2, 4, 8, 10, 12, 14, 16, 24, 28, 32, 36, 40, 44, 48\}$ loops, with parameters shared across loops. The final hidden state at the last sequence position is passed through a linear classification head to predict the output symbol. The classification head is linear and does not use bias. We use model dimension 256, with 8 attention heads and MLP hidden dimension 512.

Optimization. Models are trained using AdamW with learning rate 10^{-3}, cosine decay to a minimum learning rate of 10^{-4}, and a warmup of 1000 steps. We set weight decay to zero. Training is run for 20,000 steps with batch size 256. Gradients are clipped to norm 1.0. We use automatic mixed precision for efficiency.

E.1.3 Additional Experiments with Loop Embedding

In addition to the vanilla looped Transformer considered in Section 4.1, we also study a variant that incorporates *loop embeddings*. Concretely, at each loop iteration $\ell \in \{0, \ldots, k-1\}$, we add a learned embedding vector $e_k \in \mathbb{R}^{d_{\text{model}}}$ to the hidden states before applying the shared Transformer block. This provides the model with explicit information about the current iteration index, allowing the weight-tied backbone to implement iteration-dependent update rules. This design is analogous

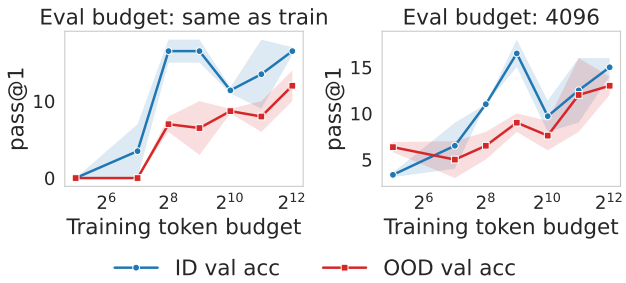


Figure 5: Pass@1 results show the same qualitative pattern: ID performance oscillates slightly more but exhibits no clear improvement beyond a token budget of 256, while OOD accuracy continues to improve steadily under both evaluation setups.

to timestep or iteration encodings used in recurrent and looped Transformer variants, such as the Universal Transformer [6], and more recent work on looped Transformers [46] and continuous CoT models [14]. Figure 4 reports results for looped Transformers with loop embeddings. We observe the same pattern as in Figure 1: ID validation accuracy saturates rapidly around 2 loops, while OOD accuracy continues to improve with increasing loop count, exhibiting clear gains up to around 36 loops. This indicates that the OOD improvements observed in Section 4.1 are robust to the inclusion of explicit loop-index conditioning and do not rely on strict homogeneity across iterations.

E.2 Additional Details for Section 4.2

E.2.1 Data and Training Details

Additional dataset details. For both the ID and OOD datasets, we use 900 examples for training and 100 examples for validation. We create a custom variant of the `Polynomial Roots` task because the original version is too difficult for the 1.5B base model (0% accuracy), making RL training uninformative. In our variant, each example presents a univariate polynomial $P(x)$ (with integer coefficients) and asks for the *largest rational root* written in lowest terms as n/m, followed by the derived target $|n| \cdot |m|$. Concretely, we construct polynomials by sampling a small number of rational roots and coefficient factors, forming linear terms of the form $(A_i x - b_i)$, and multiplying them using `sympy` to obtain $P(x)$, which is then used in the prompt. The ground-truth label is computed by identifying the largest rational root of $P(x)$, reducing it to coprime integers (n, m), and outputting $|n| \cdot |m|$. For example, one instance from our dataset is:

> Let $P(x) = 144x^3 - 276x^2 - 104x - 9$. Find the largest root n/m where n and m are coprime. What is $|n| \cdot |m|$?

The largest root is $9/4$, so the target is $|9| \cdot |4| = 36$. This construction yields instances that are solvable by the 1.5B model while still requiring multi-step reasoning.

Training and evaluation setup. On the math setting (Fig. 2, left), we fine-tune `Qwen2.5-1.5B-Instruct` using GRPO [13] with a correctness-only reward, and apply LoRA [16] with rank 32 and $\alpha = 64$. All experiments are run using the `verl` framework. We train for 4 epochs (224 total training steps) with AdamW, a learning rate of 3×10^{-6}, and a KL regularization coefficient of 0.001. The batch size is 16, with 16 generations sampled per prompt. Training is performed on 4 NVIDIA A6000 GPUs. At evaluation time, we decode with temperature 1. For the code setting (Fig. 2, middle), we apply GRPO fine-tuning to `Qwen3-8B` [43] on `LiveCodeBench v6` [19] and evaluate OOD on `DAPO-Math-17K` [48]; we also replicate this budget sweep on `Olmo3-7B` [30] (Fig. 3, right; Table 2) to verify robustness across an independent pre-training lineage. For the 4B math scaling experiment (Fig. 3, left; Table 2), we fine-tune `Qwen3-4B` [43] on `Polynomial Roots` and evaluate on a harder variant of `Mixed Operation` so that accuracy stays away from the 100% ceiling across token budgets, mirroring our difficulty adjustment of `Polynomial Roots` for the 1.5B model above.

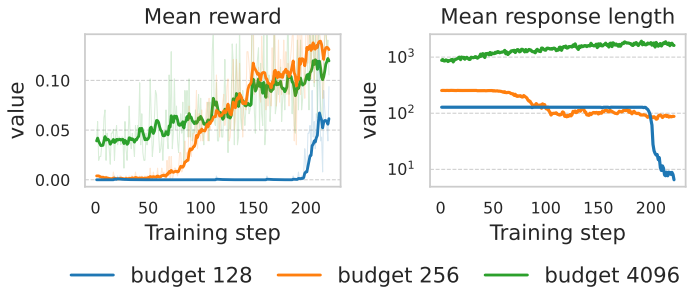


Figure 6: For small budgets (128/256), the mean reward (left) remains near 0 early in training and rises sharply when the mean response length (right) abruptly drops, indicating a transition to shorter, less step-by-step outputs. In contrast, with a large budget (4096), response length increases throughout training.

E.2.2 Additional Results and Regularization Ablation

The Pass@1 counterpart to Figure 2 for the experiment on the math dataset is shown in Figure 5. ID accuracy still peaks around a token budget of 256, while OOD accuracy continues to improve steadily under both evaluation setups. Figure 6 shows the budget-dependent reasoning dynamics discussed in Section 4.2. Table 2 reports the exact numerical values plotted in Figure 3 for `Qwen3-4B` (math, Pass@10) and `Olmo3-7B` (code, Pass@1) across all training token budgets.

Table 2: Exact ID and OOD validation accuracies (%) corresponding to Figure 3 across training-time token budgets. *Left:* `Qwen3-4B` on math (harder `Mixed Operation` OOD variant, Pass@10). *Right:* `Olmo3-7B` on code (`LiveCodeBench v6` → `DAPO-Math-17K`, Pass@1).

	`Qwen3-4B` **(Math, Pass@10)**				`Olmo3-7B` **(Code, Pass@1)**			
Train	Eval @ train		Eval @ 8192		Eval @ train		Eval @ 8192	
Budget	ID	OOD	ID	OOD	ID	OOD	ID	OOD
128	0.0	0.0	88.0	45.5	—	—	—	—
256	0.0	5.0	89.0	46.5	21.1	0.3	38.3	76.3
512	1.0	17.5	91.0	49.0	10.5	6.1	14.8	75.9
1024	15.0	36.5	86.0	46.5	22.0	25.5	44.5	75.2
2048	55.0	50.5	91.0	51.0	33.2	49.1	43.4	75.8
4096	96.0	62.0	91.0	62.5	35.7	66.8	34.1	75.4
8192	100.0	77.5	91.0	73.0	41.6	75.8	41.6	75.8

Exploratory ablation setting the KL coefficient to zero. To examine the role of explicit KL regularization in GRPO, we also ran an exploratory sweep training `Qwen2.5-1.5B-Instruct` on the same math setting with the KL penalty coefficient set to 0 (1 seed per budget due to compute constraints; Table 3). Under a fixed 8192-token evaluation cap, Pass@1 ID accuracy stays roughly flat ($7.1\% \rightarrow 8.4\%$, peaking at 9.3% at budget 1024) while OOD accuracy varies between 8.3% and 11.7% (9.8% at 128 vs. 11.3% at 8192). However, because this sweep uses a single seed per budget—and removing the KL term removes a stabilizing constraint on policy updates, increasing seed-to-seed variance—a non-monotonic ≈ 1.5-point difference cannot be separated from seed noise, and we treat this result as inconclusive. More broadly, setting the KL coefficient to zero ablates only one contributor rather than regularization as a whole: practical RL pipelines retain multiple other explicit and implicit regularizers that are integral to stable optimization, including objective clipping, weight decay, the low-rank update constraint of LoRA, and the implicit bias of gradient-based optimization [40, 1, 29, 32, 26, 49, 2].

Table 3: Exploratory KL $= 0$ sweep on math (`Qwen2.5-1.5B-Instruct`, Pass@1, 1 seed per budget). Because removing KL increases run-to-run variance and other explicit/implicit regularizers (clipping, LoRA, optimization bias) remain active, the small non-monotonic variations at a single seed are inconclusive.

Train Token Budget	ID (eval @ train)	OOD (eval @ train)	ID (eval @ 8192)	OOD (eval @ 8192)
128	0.0	0.0	7.1	9.8
256	0.0	2.4	8.2	9.3
512	0.0	8.0	8.4	9.5
1024	3.1	9.4	9.3	8.3
2048	8.9	11.9	9.0	10.1
4096	7.5	10.0	8.6	11.7
8192	8.1	10.9	8.4	11.3

Example outputs at different training-time token budgets

Question.

```
For 5x**3 + 13x*2 + 4x - 4 = 0, express the smallest root as n/m in lowest terms.
     Calculate the product of absolute values |n| * |m|?
```

Model trained with a token budget of 128.

```
\\[\n\\boxed{2}\n\\]
```

Model trained with a token budget of 256.

```
We first find the roots of the given cubic equation by factoring or using numerical
     methods. This gives us the roots -4, -2, and 1/2. The smallest root is -2,
     which is in the form of a fraction. Therefore, n=2 and m=1, and their product
     of absolute values is |n| * |m| = 2 * 1 = \\boxed{2}.
```

Model trained with a token budget of 4096.

(Continued on next page.)

```
To find the smallest root of the polynomial equation \\(5x^3 + 13x^2 + 4x - 4 =
    0\\), we can use the Rational Root Theorem, which suggests that any
   rational root of the polynomial equation is a factor of the constant term (-4)
       divided by a factor of the leading coefficient (5). The factors of 4 are
       \\(\\pm 1, \\pm 2,
   \\pm 4\\), and the factors of 5 are \\(\\pm 1, \\pm 5\\). Therefore, the
       possible rational roots are \\(\\pm 1, \\pm \\frac{1}{5}, \\pm 2, \\pm
       \\frac{2}{5}, \\pm 4, \\pm
   \\frac{4}{5}\\).\n\nWe will test the possible roots by substituting them into
       the polynomial:\n\nFirst, let's test \\(x = 1\\):\n\\[5(1)^3 + 13(1)^2 +
       4(1) - 4 = 5 + 13 + 4
    - 4 = 18 \\neq 0.\\]\nSo, \\(x = 1\\) is not a root.\n\nNext, let's test \\(x =
        -1\\):\n\\[5(-1)^3 + 13(-1)^2 + 4(-1) - 4 = -5 + 13 - 4 - 4 = -2 \\neq
        0.\\]\nSo, \\(x =
   -1\\) is not a root.\n\nNext, let's test \\(x =
       \\frac{1}{5}\\):\n\\[5\\left(\\frac{1}{5}\\right)^3 +
       13\\left(\\frac{1}{5}\\right)^2 + 4\\left(\\frac{1}{5}\\right) - 4 =
   5\\left(\\frac{1}{125}\\right) + 13\\left(\\frac{1}{25}\\right) +
       4\\left(\\frac{1}{5}\\right) - 4 = \\frac{5}{125} + \\frac{13}{25} +
       \\frac{4}{5} - 4.\\]\nConverting all
   terms to a common denominator (125):\n\\[\\frac{5}{125} + \\frac{13(5)}{125} +
       \\frac{80}{125} - \\frac{500}{125} = \\frac{5 + 65 + 80 - 500}{125} =
       \\frac{-350}{125} =
   -2.8 \\neq 0.\\]\nSo, \\(x = \\frac{1}{5}\\) is not a root.\n\nNext, let's test
       \\(x = -\\frac{1}{5}\\):\n\\[5\\left(-\\frac{1}{5}\\right)^3 +
   13\\left(-\\frac{1}{5}\\right)^2 + 4\\left(-\\frac{1}{5}\\right) - 4 =
       5\\left(-\\frac{1}{125}\\right) + 13\\left(\\frac{1}{25}\\right) +
       4\\left(-\\frac{1}{5}\\right) - 4
   = -\\frac{5}{125} + \\frac{13}{25} - \\frac{4}{5} - 4.\\]\nConverting all terms
       to a common denominator (125):\n\\[-\\frac{5}{125} + \\frac{13(5)}{125} -
       \\frac{80}{125} -
   \\frac{500}{125} = -\\frac{5}{125} + \\frac{65}{125} - \\frac{80}{125} -
       \\frac{500}{125} = -\\frac{580}{125} = -4.64 \\neq 0.\\]\nSo, \\(x =
       -\\frac{1}{5}\\) is not a
   root.\n\nNext, let's test \\(x = 2\\):\n\\[5(2)^3 + 13(2)^2 + 4(2) - 4 = 5
       \\cdot 8 + 13 \\cdot 4 + 4 \\cdot 2 - 4 = 40 + 52 + 8 - 4 = 96 \\neq
       0.\\]\nSo, \\(x = 2\\) is
   not a root.\n\nNext, let's test \\(x = -2\\):\n\\[5(-2)^3 + 13(-2)^2 + 4(-2) - 4
       = 5 \\cdot (-8) + 13 \\cdot 4 + 4 \\cdot (-2) - 4 = -40 + 52 - 8 - 4 =
       0.\\]\nSo, \\(x =
   -2\\) is a root.\n\nSince \\(x = -2\\) is a root, we can perform polynomial
       division of \\(5x^3 + 13x^2 + 4x - 4\\) by \\(x + 2\\). Using synthetic
   division:\n\n\\[\n\\begin{array}{r|rrrr}\n-2 & 5 & 13 & 4 & -4 \\\\\n & & -10 &
       -6 & 4 \\\\\n\\hline\n & 5 & 3 & -2 & 0 \\\\\n\\end{array}\n\\]\nThe
       quotient is
   \\(5x^2 + 3x - 2\\), so we have:\n\\[5x^3 + 13x^2 + 4x - 4 = (x + 2)(5x^2 + 3x -
       2).\\]\nNext, we solve the quadratic equation \\(5x^2 + 3x - 2 = 0\\) using
       the quadratic
   formula \\(x = \\frac{-b \\pm \\sqrt{b^2 - 4ac}}{2a}\\):\n\\[x = \\frac{-3 \\pm
       \\sqrt{3^2 - 4 \\cdot 5 \\cdot (-2)}}{2 \\cdot 5} = \\frac{-3 \\pm \\sqrt{9
       + 40}}{10} =
   \\frac{-3 \\pm \\sqrt{49}}{10} = \\frac{-3 \\pm 7}{10}.\\]\nThis gives us two
       solutions:\n\\[x = \\frac{-3 + 7}{10} = \\frac{4}{10} =
       \\frac{2}{5},\\]\n\\[x = \\frac{-3 -
   7}{10} = \\frac{-10}{10} = -1.\\]\nThe roots of the equation \\(5x^3 + 13x^2 +
       4x - 4 = 0\\) are \\(x = -2\\) and \\(x = \\frac{2}{5}\\) and \\(x = -1\\).
       The smallest root
    is \\(-2\\), so \\(n = -2\\) and \\(m = 1\\).\n\nThe product of the absolute
        values of \\(n\\) and \\(m\\) is:\n\\[|n| \\cdot |m| = |-2| \\cdot |1| = 2
        \\cdot 1 =
   2.\\]\nThus, the answer is:\n\\[\\boxed{2}.\\]
```

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   [Yes]

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   [Yes]

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.

- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   [Yes]

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.

- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

   [Yes]

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
   - The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
   - The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
   - The assumptions made should be given (e.g., Normally distributed errors).
   - It should be clear whether the error bar is the standard deviation or the standard error of the mean.
   - It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
   - For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
   - If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   [Yes]

   Justification: Compute resources, hardware (4 NVIDIA A6000 GPUs), and training FLOPs estimates across token budgets are detailed in Section 4.2 (Table 1) and Appendix E.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   [Yes]

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    [N/A].

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.

- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    [N/A].

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    [Yes]

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    [N/A].

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.

- At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    [N/A].

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    [N/A].

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    [N/A].

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.