DIAGNOSING INEFFICIENT TRAINING OF DIFFUSION LANGUAGE MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion language models (DLMs) need to be trained far longer than autoregressive models (ARMs) to reach the same held-out loss, despite their training objectives sharing a Bayes-optimal value. Proposed explanations include gradient noise, competition among corruptions and hard prediction problems; we test all three. We diagnose this training efficiency gap with a noisy quadratic model of training dynamics and find that it is largely explained not by gradient noise but by the features the DLM has learned, which permit far less loss reduction than an ARM's features taken at the same loss. This deficiency does not arise from averaging over corruptions, whose gradients point in the same direction, but from corruptions at high rates, which carry most of the DLM's loss in errors that its features can do little to correct. To distinguish limited expressivity from a failure to learn good features, we design Surface Sudoku, a Sudoku variant whose digits are latent variables. DLM training stalls at a loss far above an ARM's, while an auxiliary objective that supervises the latent digits helps close the efficiency gap. Our findings challenge the prevailing beliefs about the cause of DLM training inefficiency and suggest that closing the gap requires learning better features, through methods such as auxiliary objectives or adapting ARMs.

1 INTRODUCTION

Diffusion language models (DLMs) have emerged as an alternative to autoregressive (AR) generation (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024). A DLM generates text by iteratively denoising an entire sequence rather than producing one token at a time. Each network evaluation outputs multiple tokens, shifting small-batch generation from bandwidth-bound toward compute-bound and improving latency (Khanna et al., 2025; Taïga et al., 2026).

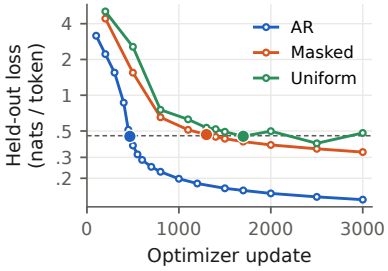


Figure 1: Held-out loss curves for ARMs and two types of DLMs. Filled markers indicate the checkpoints we analyze.

A key obstacle to DLM adoption is the cost of training them (von Rütte et al., 2026; Sahoo et al., 2026). Although the negative evidence lower bound (ELBO) that DLMs optimize and the negative log-likelihood that ARMs optimize share a Bayes-optimal value (Jeon et al., 2025; Casado Noguerales et al., 2026), DLMs require an order of magnitude more compute to reach the same loss on the same training data as an ARM (von Rütte et al., 2026; Sahoo et al., 2026); Figure 1 shows the gap for the models we analyze. So why is DLM training so inefficient? Prior work has proposed several explanations:

- **Gradient Noise** The DLM objective yields noisier gradients than the autoregressive objective (Arriola et al., 2025; Sahoo et al., 2026; Jia et al., 2026).
- **Corruption Averaging** DLM training wastes compute because different corruptions compete for the limited capacity of a single model (Wang et al., 2026; Shih et al., 2022; Xue et al., 2026).
- **Hard Prediction Problems** Denoising is harder, posing tasks which require more computation or expressivity than autoregression (Kim et al., 2025; Lee et al., 2026; von Rütte et al., 2026).

Reviewers: please read the Reviewer Guidelines (iclr.cc/Conferences/2027/ReviewerGuidelines) and the AI Policy for Reviewers (iclr.cc/Conferences/2027/AIPolicyForReviewers). If you used AI to expand, edit, or polish your review, please provide the input text to the LLM. Better still, consider skipping the LLM and submitting your original text: we, and the authors, are much more interested in your unedited thoughts than in what an LLM has to say. AI-assisted or not, you are putting your name and reputation behind your review: LLM-generated falsehoods, hallucinations or misrepresentations are subject to disciplinary action, which may include desk-rejecting all papers you have authored.

Key Contributions

1. We identify the features that diffusion models learn, rather than gradient noise, as the main source of the efficiency gap. With these features held fixed, a noisy quadratic model (Zhang et al., 2019) retains the gap, which does not narrow as training continues, and the gap persists even without the effects of gradient noise (Section 3).
2. We show that competition among corruptions, whether across rates or among draws at the same rate, does not cause diffusion's smaller attainable decrease. Instead, high corruption rates carry most of diffusion's loss, which the checkpoint's features can do little to reduce (Section 4).
3. We introduce Surface Sudoku, a variant of Sudoku designed to serve as a better proxy for language. We find that methods promoting feature learning, such as adding an auxiliary loss, accelerate training of masked diffusion without changing model capacity, suggesting that the efficiency gap can be due to a feature learning deficiency. Adapting a model which has already learned good features, such as an ARM, also significantly helps close the efficiency gap (Section 5).

2 Background

Objectives Each training sequence s is a prompt followed by a target $y := (y_1, \ldots, y_{L_s})$; the prompt is never corrupted or predicted, and we omit it from the notation. At each position p the network, with parameters $\theta \in \mathbb{R}^n$, outputs logits $z_p(\theta) \in \mathbb{R}^V$ over V tokens, scored against the clean token y_p by the cross-entropy $\ell(z, y) := -\log \mathrm{softmax}(z)_y$. A corruption ω consists of a sampled rate, the positions it corrupts at that rate, and, for uniform diffusion, the tokens it inserts. Each objective averages over positions p of a corrupted sequence with weights w_p, over corruptions ω applied to a sequence s, and over sequences:

$$\mathcal{L}(\theta) := \mathbb{E}_s \, \mathbb{E}_{\omega|s} \, \ell_{s\omega}(\theta), \qquad \ell_{s\omega}(\theta) := \frac{1}{\bar{L}} \sum_{p=1}^{L_s} w_p \, \ell\big(z_p(\theta), y_p\big), \tag{1}$$

where normalizing by $\bar{L} := \mathbb{E}_s L_s$ expresses every loss we report in nats per target token. The subscripts of a quantity indicate the averages that have not yet been taken: $\ell_{s\omega}$ is the loss of a corrupted sequence, $\ell_s := \mathbb{E}_{\omega|s}\ell_{s\omega}$ the expected loss of a clean sequence, and $\mathcal{L} = \mathbb{E}_s \ell_s$ the population loss. *AR* predicts each token from its prefix $y_{<p}$ with $w_p = 1$ and no corruption, so $\ell_s = \ell_{s\omega}$ and Eq. 1 is the negative log-likelihood. *Masked diffusion* (Austin et al., 2021; Sahoo et al., 2024; Shi et al., 2024) draws a rate $\tau \sim \mathcal{U}(0, 1)$, masks each token with probability τ, and predicts from the masked sequence y^M with $w_p = M_p/\tau$, where M_p indicates that token p is masked; Eq. 1 is then the negative ELBO, an upper bound on the negative log-likelihood. *Uniform diffusion* (Austin et al., 2021; Lou et al., 2024) replaces each token with probability τ by a uniformly random one and predicts from the corrupted sequence y^τ and the rate τ, with $w_p = 1$. We train it on this denoising cross-entropy, which, unlike the negative uniform-diffusion ELBO (Von Rütte et al., 2025), is convex in the logits, as the quadratic model of Section 3 requires. We recover the negative ELBO from the denoiser by the leave-one-out conversion of Gourevitch et al. (2026) (Appendix A.1).

Analyzing the ELBO We adopt the approach of recent scaling-law work (Sahoo et al., 2026; von Rütte et al., 2026) in comparing ARMs and DLMs by their training objectives. AR's negative log-likelihood and the diffusion ELBOs share their Bayes-optimal value, the entropy of the data (Jeon et al., 2025; Casado Noguerales et al., 2026), implying that at equal loss the objectives have the same excess above their common optimum, and the efficiency gap is the relative difference in how quickly they reduce this excess loss. Away from the optimum, the negative ELBO only upper-bounds a DLM's negative log-likelihood, but scoring masked diffusion models by their exact likelihood under a deterministic unmasking order closes at most a third of their gap to an ARM (Turok et al., 2026). Held-out loss is also seen as a meaningful predictor of downstream performance (Chen et al., 2025).

Setup We train transformers following von Rütte et al. (2026)'s architecture on TinyGSM (Liu et al., 2023), which has been studied in prior work (Kim et al., 2026; Deschenaux & Gulcehre, 2026), and has the useful property that its answers are nearly uniquely determined by the prompts, implying a near-0 optimal loss and eliminating the argument of irreducible loss as a cause of inefficient training made by Hong et al. (2026). We train with the ADOPT optimizer (Taniguchi et al., 2024), whose second-moment estimate excludes the current batch, so that each preconditioned batch gradient is an unbiased estimate of the preconditioned population gradient, as the quadratic analysis requires. See Appendix A for more details.

3 A NOISY QUADRATIC ACCOUNT

Motivation A commonly proposed explanation for the slow training of DLMs is the gradient noise resulting from the variance of Monte Carlo estimates of the ELBO over corruptions (Jia et al., 2026; Arriola et al., 2025). Prior work reduces it by designing Rao-Blackwellized objectives (Sahoo et al., 2024), training curricula (Sahoo et al., 2025), and antithetic sampling methods (Jia et al., 2026). We measure its contribution to the gap with the noisy quadratic model, which prior work has used to model the effects of gradient noise on training (Zhang et al., 2019; McCandlish et al., 2018) and the training dynamics of language models (Meterez et al., 2026). In 3.2 we decompose training in this model into the largest loss decrease the model allows, the part of it achieved within a finite training horizon, and the part lost to gradient noise. The model reproduces the gap, and the gap stems mostly in the first part rather than from the gradient noise.

3.1 THE QUADRATIC MODEL

Features Near a checkpoint θ_0, an update $\theta - \theta_0$ changes the logits at position p by $J_p(\theta - \theta_0)$ to first order, where $J_p := \partial z_p / \partial \theta \,|_{\theta_0} \in \mathbb{R}^{V \times n}$. The rows of J_p, the gradients of the logits, are the network's tangent features, whose inner products form the neural tangent kernel (Jacot et al., 2018), and we call the Jacobians J_p the checkpoint's *features*. Holding them fixed gives the linearized, or tangent, model $z_p(\theta_0) + J_p(\theta - \theta_0)$ (Chizat et al., 2019), which describes training from a checkpoint taken after the kernel's early rapid change (Fort et al., 2020); *feature learning* is the change of J_p during training.

Gradient and Curvature With the features fixed, the logits are affine in θ and the loss is convex, with gradient $r_p = \pi_p - \mathbf{1}_{y_p}$ and Hessian $\Lambda_p = \operatorname{diag}(\pi_p) - \pi_p \pi_p^\top \succeq 0$ in the logits, where $\pi_p := \operatorname{softmax}(z_p)$ and $\mathbf{1}_{y_p}$ is the one-hot target. With ADOPT's positive diagonal preconditioner D frozen, a continuation is heavy-ball momentum (Polyak, 1964) on $\delta := D^{-1/2}(\theta - \theta_0) \in \mathbb{R}^n$ (Appendix B.2); from here on, J_p denotes the features in these coordinates, $J_p D^{1/2}$. The tangent model's population gradient and Hessian in δ are then

$$g := \mathbb{E}_s \mathbb{E}_{\omega|s} \frac{1}{\overline{L}} \sum_p w_p J_p^\top r_p, \qquad G := \mathbb{E}_s \mathbb{E}_{\omega|s} \frac{1}{\overline{L}} \sum_p w_p J_p^\top \Lambda_p J_p, \tag{2}$$

where the Gauss–Newton matrix $G \succeq 0$ is the Gram matrix of the features weighted by Λ_p.

Quadratic Minimum The quadratic model is the second-order expansion $F(\delta) := \mathcal{L}(\theta_0) + g^\top \delta + \frac{1}{2}\delta^\top G \delta$ of the tangent model's loss, every term of which is measurable at the checkpoint; its minimum approximates the largest decrease that training with the checkpoint's features can achieve. Since F is convex and $g \in \operatorname{range} G$ (Appendix B.1), it is minimized by $\delta_* := -G^\dagger g$, with $G^\dagger$ the Moore–Penrose pseudoinverse, so that $\nabla F(\delta) = g + G\delta = G(\delta - \delta_*)$ and, with the eigendecomposition $G = \sum_i \lambda_i u_i u_i^\top$,

$$F(\delta) - F(\delta_*) = \frac{1}{2}(\delta - \delta_*)^\top G\,(\delta - \delta_*) = \frac{1}{2}\sum \lambda_i \big(u_i^\top (\delta - \delta_*)\big)^2. \tag{3}$$

With $g_i := u_i^\top g$ and the checkpoint's displacement $x_{i,0} := -u_i^\top \delta_*$ from the minimizer, projecting $g = -G\delta_*$ onto u_i gives $g_i = \lambda_i x_{i,0}$, so $\lambda_i^\dagger g_i^2 = \lambda_i x_{i,0}^2$, where $\lambda_i^\dagger$ is $1/\lambda_i$ if $\lambda_i > 0$ and 0 otherwise.

Definition 3.1 (Attainable decrease). The *attainable decrease* Δ is the height of the checkpoint above the minimum of F. It does not depend on D, since an invertible change of coordinates leaves $\min F$ unchanged (Appendix B.1), and it is formulated as follows:

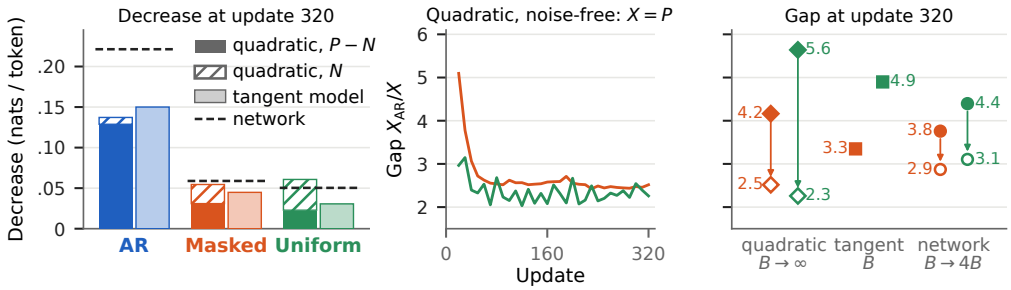
$$\Delta := F(0) - F(\delta_*) = \frac{1}{2}\delta_*^\top G\, \delta_* = \frac{1}{2}\sum_i \lambda_i x_{i,0}^2 = \frac{1}{2}\sum_i \lambda_i^\dagger g_i^2 = \frac{1}{2} g^\top G^\dagger g. \tag{4}$$

Direction i contributes Δ_i, and a curvature cutoff $\kappa \geq 0$ gives the cumulative value $\Delta_{\leq \kappa}$:

$$\Delta_i := \frac{1}{2}\lambda_i x_{i,0}^2 = \frac{1}{2}\lambda_i^\dagger g_i^2, \qquad \Delta_{\leq \kappa}(g) := \sum_{i:\, \lambda_i \leq \kappa} \Delta_i, \qquad \Delta = \Delta_{\leq \infty}(g). \tag{5}$$

We measure $\frac{1}{2}\lambda_i^\dagger g_i^2$ from g and G at the checkpoint, while the optimizer removes $\frac{1}{2}\lambda_i x_{i,0}^2$ by driving the displacement to zero (Figure 2).

3.2 DECOMPOSING TRAINING PROGRESS



(a) (b) (c)

Figure 3: **Without its noise cost, diffusion's decrease remains well below AR's.** (a) Loss decrease over 320 updates: the quadratic model's $P - N$ (solid) and N (hatched), the tangent model (light) and the network (dashed). (b) The quadratic model's P_{AR}/P by update. (c) X_{AR}/X at update 320, at batch B (filled) and with less gradient noise (open). The quadratic model's P and N follow from runs with one and four gradient batches per update (Appendix A.2).

Training Dynamics Δ assumes unlimited updates and exact gradients; we now ask how much of it T noisy updates realize. Let δ_t be the iterate after t updates of the checkpoint's optimizer on F, started from the checkpoint's momentum buffer and driven by batch gradients $\widehat{g}_t = \nabla F(\delta_t) + \xi_t$, and let $x_{i,t} := u_i^\top(\delta_t - \delta_*)$ be its displacement along u_i. By Eqs. 3 and 4, the loss removed after T updates is

$$F(0) - F(\delta_T) = \sum_i \frac{1}{2}\lambda_i\big(x_{i,0}^2 - x_{i,T}^2\big). \qquad (6)$$

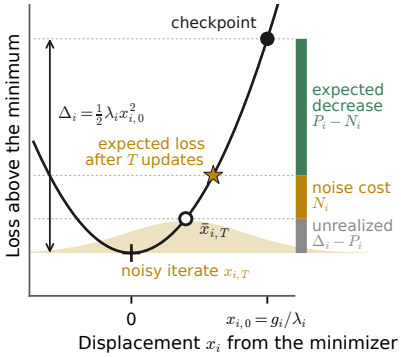


Figure 2: A single direction of Eq. 7. The checkpoint starts Δ_i above the minimum. After T updates, the height of the noise-free iterate $\bar{x}_{i,T}$ (open circle) above the minimum is the unrealized decrease, the spread of the noisy iterate incurs the noise cost N_i, and the remainder is the expected decrease.

The sampling errors ξ_t, the model's only randomness, have zero mean and are independent across updates. For a batch of B independently corrupted sequences their covariance is $\Sigma_B = \Sigma_1/B$, where Σ_1 is the covariance at the checkpoint of the gradient $g_{s\omega}$ of one corrupted sequence, the sum in Eq. 2 without its outer expectations. We assume that Σ_B stays fixed along the trajectory (*additive* noise), which Appendix C.6 checks along the network's path. Training then falls short of Δ for two reasons: within T updates the mean iterate need not reach the minimizer, and the sampling errors scatter the iterate around that mean.

Decomposition With the mean $\bar{x}_{i,t} := \mathbb{E}_\xi x_{i,t}$ over the sampling errors $\xi_0, \ldots, \xi_{T-1}$ and $\mathbb{E}_\xi x_{i,T}^2 = \bar{x}_{i,T}^2 + \mathrm{Var}_\xi(x_{i,T})$, the expectation of Eq. 6, the *expected decrease*, is

$$F(0) - \mathbb{E}_\xi F(\delta_T) = \sum_i (P_i - N_i), \qquad P_i := \frac{1}{2}\lambda_i\big(x_{i,0}^2 - \bar{x}_{i,T}^2\big), \qquad N_i := \frac{1}{2}\lambda_i \,\mathrm{Var}_\xi(x_{i,T}). \quad (7)$$

Since $\sum_i \frac{1}{2}\lambda_i x_{i,0}^2 = \Delta$, the attainable decrease splits, at the horizon T, into the *expected decrease* $P - N$, the *noise cost* $N = \sum_i N_i$ and the *unrealized decrease* $\Delta - P = \sum_i \frac{1}{2}\lambda_i \bar{x}_{i,T}^2$ (Figure 2), where $P = \sum_i P_i$ is the *noise-free decrease* of the mean iterate. Momentum is linear in the gradients it receives, so $\bar{x}_{i,T}$ is the noise-free iterate and $N \propto \Sigma_B \propto 1/B$; Appendix B.2 gives P and N in closed form, with the stationary limits of Zhang et al. (2019, Thm. 1). This decomposes the efficiency gap: an inefficient optimizer produces large $\Delta - P$, gradient noise a large N, and hard prediction problems or deficient features yield a small Δ.

Path Span The directions u_i cannot be computed in all n parameters, so we restrict F to the span $\mathcal{S}$ of 120 iterates of the quadratic model, run in all parameters for 640 updates; training restricted to such a subspace can match full training (Li et al., 2023). With Q an orthonormal basis of $\mathcal{S}$, the *attainable decrease within* $\mathcal{S}$ is $\Delta_{\mathcal{S}} := F(0) - \min_{\delta \in \mathcal{S}} F(\delta) \le \Delta$. From here on, λ_i and $u_i := Q v_i$ denote the curvatures and directions within $\mathcal{S}$, where (λ_i, v_i) are the eigenpairs of $Q^\top G Q$ and $g_i := u_i^\top g$. As in Eq. 5,

$$G_{\le\kappa} := \sum_{i:\,\lambda_i \le \kappa} \lambda_i u_i u_i^\top, \qquad \Delta_{\mathcal{S},\le\kappa}(g) := \frac{1}{2} g^\top G_{\le\kappa}^\dagger g = \sum_{i:\,\lambda_i \le \kappa} \frac{1}{2}\lambda_i^\dagger g_i^2, \qquad (8)$$

with $\Delta_{\mathcal{S}} = \Delta_{\mathcal{S},\leq\infty}$. The noise cost needs no subspace: stochastic Lanczos quadrature (Ubaru et al., 2017), started from samples of the gradient noise, gives N over all parameters (Appendix C).

3.3 Results

Fixed Features Because Eq. 7 describes training with fixed features, it is a baseline rather than a forecast of the network's trajectory, and it can account for the gap only if the gap persists when the features are fixed. We write X_{AR} for AR's value of a quantity X and measure the gap by ratios such as P_{AR}/P. Figure 3a compares the decrease over $T = 320$ updates of the quadratic model run in all parameters, of the tangent model, which keeps the features fixed but not the quadratic approximation, and of the network. The quadratic model achieves about half of each network's decrease and the tangent model three fifths to three quarters, yet the quadratic model's gap is at least as large as the network's and the tangent model's is comparable (Figure 3c; Table 2). The part of each network's decrease that requires feature change, the difference between the network and the tangent model in Figure 3a, is three to five times larger for AR than for either diffusion objective.

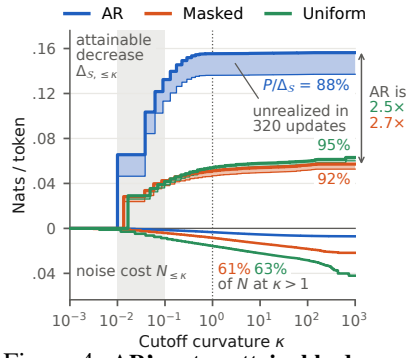


Figure 4: **AR's extra attainable decrease lies at low curvature, diffusion's noise cost at high curvature.** Above zero: $\Delta_{\mathcal{S},\leq\kappa}$, with the part not realized by update 320 without noise, $\Delta_{\mathcal{S},\leq\kappa} - P_{\leq\kappa}$, filled; labels give $P/\Delta_{\mathcal{S}}$ and, at the right edge, $\Delta_{\mathcal{S},\mathrm{AR}}/\Delta_{\mathcal{S}}$. Below zero, on the same scale: $N_{\leq\kappa}$ over all parameters at update 320; labels give the share of N at curvatures above 1. Shading: $.01 \leq \kappa \leq .1$ (Table 3).

Gradient Noise If gradient noise caused the gap, the gap would close as $B \to \infty$, where $N \propto 1/B$ vanishes. In the quadratic model, removing the noise lowers the gap from $(P - N)_{\mathrm{AR}}/(P - N) = 4.2$ and 5.6 to $P_{\mathrm{AR}}/P = 2.5$ and 2.3 for masked and uniform diffusion (Figure 3c; Table 1): noise costs diffusion $N/P = .43$ and .62 of its noise-free decrease, against .06 for AR, yet without it AR's decrease is still more than twice diffusion's. The network agrees: quadrupling the batch lowers its gap from 3.8 and 4.4 to 2.9 and 3.1.

Optimizer Speed Nor does the optimizer cause the gap. Without noise, the displacement along each direction decays geometrically, as about $(1 - \eta\lambda_i)^T$ below curvature .025 (Appendix B.2), so $P_{\mathrm{AR}}/P \to \Delta_{\mathrm{AR}}/\Delta$ as $T \to \infty$, and an optimizer that left diffusion relatively more decrease unrealized would make P_{AR}/P fall as T grows. It does not fall: from update 60 to 320 it stays between 2.4 and 2.7 for masked and between 2.0 and 2.7 for uniform diffusion (Figure 3b), and at update 630 it is at least 2.4 and 2.3 (Appendix A.2). Within $\mathcal{S}$, the optimizer realizes more of diffusion's attainable decrease than of AR's, $P/\Delta_{\mathcal{S}} = .92$ and .95 against .88 (Figure 4).

Attainable Decrease The gap therefore lies in the attainable decrease: $\Delta_{\mathcal{S},\mathrm{AR}}/\Delta_{\mathcal{S}} = 2.7$ and 2.5 (right edge of Figure 4), with almost all of AR's excess at curvatures $.01 \leq \lambda \leq .1$, the shaded band of Figure 4. Most of diffusion's N lies at $\lambda > 1$, where $\Delta_{\mathcal{S}}$ is small (lower half of Figure 4; Table 3). Section 4 asks whether diffusion's deficit at low curvature arises from the average over corruptions.

Key Takeaways The training efficiency gap between ARMs and DLMs is present even in a noisy quadratic model of training. Gradient noise only partially accounts for the gap. Instead, most of the gap can be attributed to the largest loss reduction that training can achieve under the quadratic approximation, a result of the model's features.

4 Averaging over Corruptions

Motivation A second explanation for the slow training of DLMs places the cost in the average over corruptions: the gradients of different corruptions may compete for one network's finite capacity (Wang et al., 2026; Shih et al., 2022; Xue et al., 2026). Training takes one step for all corruptions; if they demand different steps, part of the attainable decrease, which drives the gap (Section 3), is lost in the average. We measure whether different corruptions demand different steps and, finding that they do not, ask instead how much of each corruption's loss the checkpoint's features can reduce: corruptions at high rates carry most of diffusion's loss, which the features struggle to reduce.

4.1 CORRUPTION GRADIENTS

Corruptions as Tasks Each corruption ω, such as one set of masked positions at rate .3, defines a task. A corruption is drawn independently of the sequence it corrupts, $\mathbb{E}_{\omega|s} = \mathbb{E}_\omega$, so it can be applied to every sequence: $\ell_\omega := \mathbb{E}_s \ell_{s\omega}$ and $g_\omega := \mathbb{E}_s g_{s\omega}$ are the loss and gradient that training would see if every sequence were corrupted by ω alone, and $\mathcal{L} = \mathbb{E}_\omega \ell_\omega$, with its gradient $g = \mathbb{E}_\omega g_\omega$.

Two Factors We ask which of two things makes diffusion's attainable decrease smaller than AR's: the tasks' gradients cancel in their average g, or most of the loss lies in tasks whose loss the checkpoint's features cannot effectively reduce. For $\kappa = .1$, where we found the attainable decrease gap (Section 3.3), the two correspond to the factors of

$$\frac{\Delta_{\mathcal{S},\le\kappa}}{\Delta_{\mathcal{S},\le\kappa,\mathrm{AR}}} = c_{\le\kappa} \left(\mathbb{E}_\omega \Big[\frac{\ell_\omega}{\mathcal{L}} \, \sigma_\omega \Big] \right)^2, \tag{9}$$

the *alignment* $c_{\le\kappa}$, the fraction of the decrease that survives the average, and the *contribution per nat* σ_ω of each task relative to AR, weighted by the task's share of the loss. If competition among corruptions caused the gap, $c_{\le\kappa}$ would be well below 1. We now define the two factors.

Competition Let $\Delta_{\mathcal{S},\le\kappa}(g_\omega)$, Eq. 8 at g_ω, be the decrease that the quadratic model would allow if g_ω were the whole gradient; the best step for task ω alone is $-G^\dagger_{\le\kappa} g_\omega$. The best step for the average, $-G^\dagger_{\le\kappa} g$, lowers ℓ_ω by $g_\omega^\top G^\dagger_{\le\kappa} g$, and since $g = \mathbb{E}_{\omega'} g_{\omega'}$, with ω' drawn independently of ω,

$$\Delta_{\mathcal{S},\le\kappa} = \frac{1}{2} \, \mathbb{E}_\omega \, g_\omega^\top G^\dagger_{\le\kappa} g = \frac{1}{2} \, \mathbb{E}_{\omega,\omega'} \, g_\omega^\top G^\dagger_{\le\kappa} g_{\omega'}. \tag{10}$$

Each term $g_\omega^\top G^\dagger_{\le\kappa} g_{\omega'}$ is the first-order decrease of ℓ_ω along $-G^\dagger_{\le\kappa} g_{\omega'}$, the best step for task ω' alone; it is negative when that step raises ℓ_ω. We compare two tasks' best steps by their cosine in the curvature metric G, defined as:

$$\cos_{\le\kappa}(g_\omega, g_{\omega'}) := \frac{g_\omega^\top G^\dagger_{\le\kappa} g_{\omega'}}{\big(2 \, \Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2} \Delta_{\mathcal{S},\le\kappa}(g_{\omega'})^{1/2}\big)}.$$

Since $G^\dagger_{\le\kappa} \succeq 0$, the Cauchy–Schwarz inequality gives $\cos_{\le\kappa}(g_\omega, g_{\omega'}) \le 1$, with equality exactly when $-G^\dagger_{\le\kappa} g_\omega$ and $-G^\dagger_{\le\kappa} g_{\omega'}$ point the same way; the tasks *compete* when it is below 1. Substituting the cosine into Eq. 10 and using the independence of ω and ω',

$$\Delta_{\mathcal{S},\le\kappa} = \mathbb{E}_{\omega,\omega'} \big[\Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2} \Delta_{\mathcal{S},\le\kappa}(g_{\omega'})^{1/2} \cos_{\le\kappa}(g_\omega, g_{\omega'}) \big] \le \big(\mathbb{E}_\omega \Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2} \big)^2. \tag{11}$$

The bound is reached exactly when no two tasks compete; the attainable decrease then is the square of the mean of the tasks' *contributions* $\Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2}$.

Definition 4.1 (Alignment). The *alignment* is the fraction of the bound in Eq. 11 that the average retains,

$$c_{\le\kappa} := \frac{\Delta_{\mathcal{S},\le\kappa}}{\big(\mathbb{E}_\omega \Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2} \big)^2} \le 1, \tag{12}$$

the mean of $\cos_{\le\kappa}(g_\omega, g_{\omega'})$ weighted by the product of the two tasks' contributions. It equals 1 exactly when no two tasks compete (Appendix B.3). If competition among corruptions was responsible for the efficiency gap, c would be well below 1.

Contribution per Nat If $c_{\le\kappa} = 1$, the contributions determine the attainable decrease, $\Delta_{\mathcal{S},\le\kappa} = \big(\mathbb{E}_\omega \Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2} \big)^2$. The DLM and ARM objectives are compared at equal loss, $\mathcal{L} = \mathbb{E}_\omega \ell_\omega$, so we compare each contribution with its corruption's loss, relative to AR at the same loss,

$$\sigma_\omega := \frac{\Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2} / \ell_\omega}{\Delta^{1/2}_{\mathcal{S},\le\kappa,\mathrm{AR}} / \mathcal{L}}, \qquad \text{so that} \qquad \frac{\Delta_{\mathcal{S},\le\kappa}}{\Delta_{\mathcal{S},\le\kappa,\mathrm{AR}}} = c_{\le\kappa} \left(\mathbb{E}_\omega \Big[\frac{\ell_\omega}{\mathcal{L}} \, \sigma_\omega \Big] \right)^2 \tag{13}$$

by Eq. 12. With $c_{\le\kappa} = 1$, DLM's attainable decrease thus falls short of AR's exactly insofar as its loss lies in corruptions with $\sigma_\omega < 1$, whose loss the checkpoint's features cannot reduce as effectively as ARMs features.

Design Measuring g_ω requires applying one corruption to many sequences, which training does not do. For each diffusion objective we draw one rate in each of 32 equal-probability strata of the

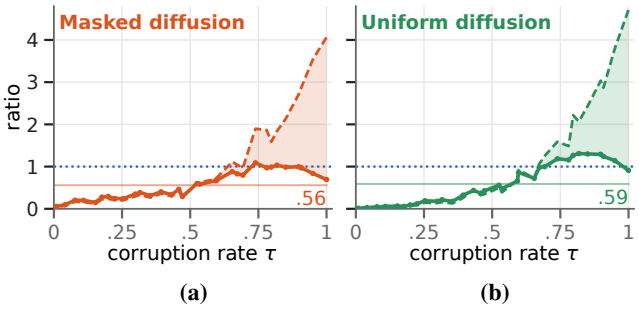


Figure 5: **At high corruption rates, the loss grows far faster than the contribution.** Per rate, averaged over its two corruptions: loss relative to the population's, $\ell_\omega/\mathcal{L}$ (dashed), and contribution relative to AR's (solid), with AR at 1 (dotted), for (a) masked and (b) uniform diffusion (denoising loss). The ratio of solid to dashed is σ_ω (Eq. 13); the labelled line, the mean of the solid curve, is $(\Delta_{\mathcal{S},\leq.1}/\Delta_{\mathcal{S},\leq.1,\mathrm{AR}})^{1/2}$, since $c_{\leq.1} = 1$. The loss is nearly all reducible at every rate, since the prompt, never corrupted, nearly determines the answer (Section 2).

rate distribution and 2 corruptions at each rate, and we apply all 64 corruptions to the same 1,024 held-out sequences, on which we also measure AR (Appendix D). Quadratic forms of averages are estimated without bias from products of the gradients of distinct sequences under distinct corruptions (Appendix B.4). All gradients are projected onto $\mathcal{S}$, with the curvatures of Section 3.2.

4.2 Results

Alignment Below curvature .1, where the gap lies, the tasks do not compete: $\cos_{\leq.1}(g_\omega, g_{\omega'}) \geq .97$ for every pair of the 64 corruptions, at the same rate or at different rates, and $c_{\leq.1} = 1.00$ for both objectives (Appendix D). Eq. 9 thus reduces to $(\Delta_{\mathcal{S},\leq.1}/\Delta_{\mathcal{S},\leq.1,\mathrm{AR}})^{1/2} = \mathbb{E}_\omega[(\ell_\omega/\mathcal{L})\,\sigma_\omega] = .56$ and .59: diffusion's smaller attainable decrease is entirely attributable to σ_ω (Figure 5).

Contribution per Nat The loss-weighted mean $\mathbb{E}_\omega[\frac{\ell_\omega}{\mathcal{L}}\sigma_\omega]$ is low because most of the loss lies at high rates, where σ_ω is small. As $\tau \to 1$, $\ell_\omega/\mathcal{L}$ rises above 4 while $\Delta_{\mathcal{S},\leq.1}(g_\omega)^{1/2}/\Delta^{1/2}_{\mathcal{S},\leq.1,\mathrm{AR}}$ stays near 1, so σ_ω falls to about .2 (Figure 5). Corruptions at high rates, which leave little uncorrupted context to predict from, therefore hold most of diffusion's loss in errors that the checkpoint's features cannot correct.

Key Takeaways Averaging over corruptions does not cause the training efficiency gap. The gradients of different corruptions could pull in different directions, so that no single training step serves them all, but, we find this is not the case. Instead, the efficiency gap can be attributed to DLM features struggling to reduce loss when there is heavy corruption, which also happens to be where most of the loss lies.

5 Surface Sudoku: A Proxy for Feature Learning

Motivation Sections 3 and 4 locate the gap in the checkpoints' features: a quadratic model that keeps them fixed removes far less of diffusion's loss than of AR's, and the deficit lies in the gradients of single corruptions. At a fixed checkpoint, two accounts predict this deficit equally well. One would say that training has not yet learned features that the network could represent that would make more loss accessible; the other would say that the corruptions pose prediction tasks that the network cannot solve, either because the tasks are computationally intractable (Kim et al., 2025), demand deeper computational graphs (Lee et al., 2026), or general expressivity associated with model size (von Rütte et al., 2026). We separate the two on a task whose missing representation is known and can be supplied during training.

5.1 Task and Test

Task Design Sudoku is a standard testbed on which DLMs outperform ARMs (Ye et al., 2025a; Kim et al., 2025; Ye et al., 2025b; Wang et al., 2026), but its constraints act directly on the observed digits. In text, by contrast, the constraints that link distant tokens often concern what the tokens refer to: a TinyGSM solution is a Python program that must refer to each quantity of the problem by the same variable name throughout, but the name chosen for a quantity changes from one solution to the next. The Random Hierarchy Model builds this structure into a model of language: each hidden

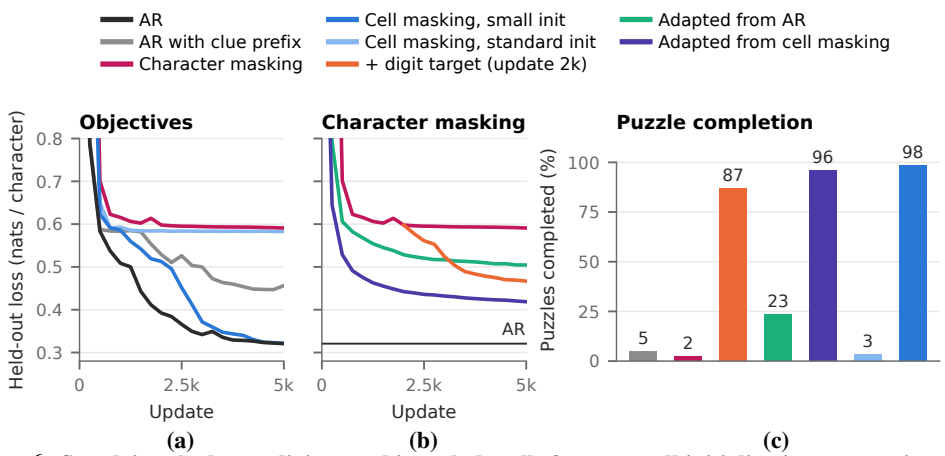


Figure 6: **Supplying the latent digits, masking whole cells from a small initialization, or starting from a trained ARM or cell-masking model lowers the loss below the character-masking plateau.** Held-out loss per character. (a) AR and character masking at the standard initialization, cell masking at both initializations, and the ARM that reads the clues as a prefix, measured with an all-mask prefix. (b) Character masking with and without the digit target added at update 2,000, and adapted from the ARM and from the small-initialization cell-masking model of (a). (c) Held-out puzzles, out of 128, completed into a valid board that keeps every clue; the AR bar is the prefix ARM of (a), it's under-performance relative to loss likely stems from order in which cells are filled largely determining Sudoku accuracy (Shah et al., 2024; Kim et al., 2025).

symbol is rewritten as one of several tuples of lower-level symbols, down to the observed tokens (Cagnetta et al., 2024; Cagnetta & Wyart, 2024). Surface Sudoku builds the same structure into Sudoku. It writes the digit $d \in \{0, \dots, 8\}$ of each cell as three characters, (a, b, c), with a cipher (r, q) that is fixed within each of the board's nine 3×3 boxes and differs between them (Figure 8):

$$a = (d + r) \bmod 9, \qquad b = (a - q) \bmod 9, \qquad c = (d - a - b) \bmod 9. \tag{14}$$

The Sudoku constraints therefore act on digits that the input never states. A complete cell reveals its digit, $d = (a + b + c) \bmod 9$, whatever the cipher, just as a tuple of the Random Hierarchy Model identifies the symbol that generated it; the characters of a partially visible cell do not.

Objectives All models use the same 16-layer transformer over the board's $81 \times 3 = 243$ characters and differ in their attention and readout (Appendix E). The ARM attends causally and predicts each character from its prefix. The masking models attend bidirectionally: character masking masks each character independently at a sampled rate and minimizes the negative ELBO $\mathcal{L}_{\text{char}}$, while cell masking masks the three characters of a cell together and predicts them jointly with a 729-way readout, so that every visible cell is complete. The digit target adds to $\mathcal{L}_{\text{char}}$ the cross-entropy of each masked character's digit, predicted by a head on the final hidden state that is discarded at evaluation (Eq. 20). Each objective is trained from a standard and a smaller initialization scale; losses are in nats per character, and $\mathcal{L}_{\text{char}}$ never includes the auxiliary term.

Test Let $\Theta \subset \mathbb{R}^n$ be the space of parameter values of the character-masking network and $\mathcal{L}^\star := \inf_{\theta \in \Theta} \mathcal{L}_{\text{char}}(\theta)$ the lowest loss it can attain. The hard-problem accounts place a plateau of $\mathcal{L}_{\text{char}}$ at $\mathcal{L}^\star$: the network cannot represent or compute a better solution. An auxiliary target changes which $\theta \in \Theta$ training reaches, not Θ, so a run with one that ends below the plateau bounds $\mathcal{L}^\star$ below it; the plateau is then a failure to learn features that the network can represent.

5.2 Results

Plateau With the same boards, optimizer and number of characters processed, the ARM reaches a loss of .321, whereas character masking plateaus at .592 at both initializations and completes $3/128$ held-out puzzles (Figure 6a,c; Table 6). This is the plateau that the hard-problem accounts predict.

Digit Supervision Added at update 2,000, once the plateau has formed, the digit target lowers $\mathcal{L}_{\text{char}}$ to .469, and the model completes 111 of 128 puzzles (Figure 6b,c). Hence $\mathcal{L}^\star \leq .469$: at its original width and depth, the network holds a far better solution of the unchanged objective, which training without the target does not find.

Corruption Structure Masking whole cells exposes the digits of the visible cells without supervising them. From the smaller initialization, cell masking reaches the ARM's loss of .321 and completes 126 of 128 puzzles, but from the standard initialization it plateaus at .582 (Figure 6a,c). The initialization scale, which leaves plain character masking at .592, likewise decides whether the digit target from update 0 reaches .445 or .503 (Table 6), in line with its known role in whether training learns features (Chizat et al., 2019). Exposing the latent is therefore not enough: training must also find the solution that uses it.

AR Adaptation The digit target needs labels for the latent, and cell masking needs to know which characters encode each digit; language supplies neither. Adapting a pretrained ARM, as several of the largest DLMs do (Gong et al., 2025; Taïga et al., 2026), needs neither. Initialized from the ARM of Figure 6a with the recipe of Gong et al. (2025), character masking reaches .506, below the plateau though above the .469 of the digit target, and completes 30 of 128 puzzles (Figure 6b,c). Initialized from the small-initialization cell-masking model of Figure 6a, character masking reaches .420 and completes 123 of 128 puzzles when each character is predicted as the marginal of the donor's joint distribution over its cell's triplets (Appendix E), but stays at .585 with the character model's own readout (Table 6). Features learned by either model thus carry over to character masking.

Key Takeaways On Surface Sudoku, where the missing representation is known, the character-masking plateau is a failure of feature learning, not a limit of the network: taught the hidden digits, the same network falls far below it on the same objective. Changing the loss (a digit target), the corruption (whole cells masked, from a small initialization) or the starting point (a trained ARM or cell-masking model) each breaks the plateau.

6 Conclusions and Limitations

Conclusions Our findings challenge the commonly held beliefs and intuitions about what makes Diffusion Language Model training so expensive compared to training Autoregressive Models. We showed that the training efficiency gap—the gap in the rate at which ARMs and DLMs reduce loss—can be recovered with a closed form approximation—the noisy quadratic model—of training dynamics. Under this model, we are able to isolate the dominant source of the gap to the maximum reducible loss under the quadratic, a quantity set by the models features, rather than the commonly blamed gradient noise. Moreover, we find that this deficiency is also not attributable to competing demands from all of the different possible corruptions of the data; quite the opposite. Instead, DLMs simply lack the features to effectively reduce loss for any heavy corruptions of the data. Furthermore, our results on the Surface Sudoku dataset we design provide further evidence of this, as training modifications intended to boost feature learning can lead the same DLM to vastly better performance, suggesting that the deficient features are not necessarily due to limited model expressivity. Based on our findings, future work aiming to close the training efficiency gap between ARMs and DLMs should explore methods aimed at improving feature learning, such as auxiliary objectives, or adapting DLMs from models which have already learned good features, such as ARMs, rather than hope that DLMs themselves will efficiently learn good features. Additionally, theoretical models explaining *why* DLMs are inefficient at learning good features, perhaps through deep linear models (Kunin et al., 2025; Abbe et al., 2023) could offer deeper insight into the factors affecting this feature learning deficiency.

Limitations We measure attainable and noise-free decreases by curvature only in the span of the continuation's path. Restricted to this span, the quadratic model reproduces the noise-free decrease, but its attainable decrease is, up to estimation error, a lower bound on the attainable decrease in all parameters. Thus, we have not fully ruled out the possibility that DLMs are just significantly more ill-conditioned and the efficiency gap is an optimizer failure; however, recent work (Bolatov et al., 2026) failed to find any optimizers significantly outperforming an AdamW (Loshchilov & Hutter, 2019) baseline. Nevertheless, our experiments did use the uncommonly used ADOPT optimizer and results are not guaranteed to be the same for AdamW. Compute costs led to us running experiments on a small ($\approx$ 6-million parameter model) on only a single dataset and a single stage in training.

AI USE STATEMENT

In this work we used Claude Opus and Fable models to help formalize mathematical claims and refine hypotheses, particularly for section 4, as well as provide feedback and suggestions for refining experimental methodology (the suggestion to construct a basis using iterates along the training path, rather than using Lanczos on a fixed subset of the data, helped get a more accurate quadratic approximation to the real training dynamics). The models also assisted significantly in experiment implementation, creating figures, and in drafting and polishing the paper for readability.

We did not use models to directly generate synthetic data, however, the implementation of the Surface Sudoku dataset was done by generative AI with human oversight and specification.

We have reviewed all AI-assisted work carefully. All claims and text written by an LLM in the main body has been very carefully reviewed and revised. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

Our work analyzes why diffusion language models train more slowly than autoregressive models. We trained only on synthetic datasets, and do not introduce any methods or applications that have negative ethical consequences.

REPRODUCIBILITY STATEMENT

Appendix A gives the training and measurement protocols and the compute used, and Algorithm 1 summarizes how every quantity of Sections 3 and 4 is measured. Appendix B derives the quadratic model's decomposition and the unbiased estimators we use; Appendix C describes the measurements by curvature, Appendix D the crossed corruption design, and Appendix E the Surface Sudoku task, its data and every run. TinyGSM and the Sudoku solutions that Surface Sudoku is built from are public. The code will be made public upon acceptance.

REFERENCES

Emmanuel Abbe, Enric Boix Adserà, and Theodor Misiakiewicz. Sgd learning on neural networks: leap complexity and saddle-to-saddle dynamics. In Gergely Neu and Lorenzo Rosasco (eds.), *Proceedings of Thirty Sixth Conference on Learning Theory*, volume 195 of *Proceedings of Machine Learning Research*, pp. 2552–2623. PMLR, 12–15 Jul 2023. URL https://proceedings.mlr.press/v195/abbe23a.html.

Marianne Arriola, Aaron Gokaslan, Justin T. Chiu, Zhihan Yang, Zhixuan Qi, Jiaqi Han, Subham Sekhar Sahoo, and Volodymyr Kuleshov. Block diffusion: Interpolating between autoregressive and diffusion language models. In *International Conference on Learning Representations*, 2025.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In *Advances in Neural Information Processing Systems*, 2021.

Arman Bolatov, Egor Shulgin, David Li, Abduragim Shtanchaev, Sebastian U. Stich, Maxim Panov, Eric Moulines, Peter Richtárik, and Martin Takáč. Optimizers for diffusion models: A controlled benchmark, 2026. URL https://arxiv.org/abs/2609.23055.

Francesco Cagnetta and Matthieu Wyart. Towards a theory of how the structure of language is acquired by deep neural networks. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Francesco Cagnetta, Leonardo Petrini, Umberto M. Tomasini, Alessandro Favero, and Matthieu Wyart. How deep neural networks learn compositional data: The random hierarchy model. *Physical Review X*, 14:031001, 2024. doi: 10.1103/PhysRevX.14.031001.

Rodrigo Casado Noguerales, Bernhard Schölkopf, Thomas Hofmann, and Aran Raoufi. What does a discrete diffusion model learn? *arXiv:2607.05381*, 2026.

Yangyi Chen, Binxuan Huang, Yifan Gao, Zhengyang Wang, Jingfeng Yang, and Heng Ji. Scaling laws for predicting downstream performance in LLMs. *Transactions on Machine Learning Research*, 2025. ISSN 2835-8856. URL https://openreview.net/forum?id=PJUbMDkQVY.

Lénaïc Chizat, Edouard Oyallon, and Francis Bach. On lazy training in differentiable programming. In *Advances in Neural Information Processing Systems*, 2019.

Justin Deschenaux and Caglar Gulcehre. Language modeling with hyperspherical flows, 2026. URL https://arxiv.org/abs/2605.11125.

Stanislav Fort, Gintare Karolina Dziugaite, Mansheej Paul, Sepideh Kharaghani, Daniel M. Roy, and Surya Ganguli. Deep Learning versus Kernel Learning: an Empirical Study of Loss Landscape Geometry and the Time Evolution of the Neural Tangent Kernel. In *Advances in Neural Information Processing Systems*, volume 33, pp. 5850–5861, 2020.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Hao Peng, Jiawei Han, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *International Conference on Learning Representations*, 2025.

Samson Gourevitch, Yazid Janati, Dario Shariatian, Umut Simsekli, Eric Moulines, Eric P. Xing, and Alain Durmus. Uniform diffusion models revisited: Leave-one-out denoiser and absorbing state reformulation. *arXiv:2605.22765*, 2026.

Chunsan Hong, Sanghyun Lee, Chieh-Hsin Lai, Satoshi Hayakawa, Yuhta Takida, Yuki Mitsufuji, Seungryong Kim, and Jong Chul Ye. Understanding and accelerating the training of masked diffusion language models, 2026. URL https://arxiv.org/abs/2605.13026.

Arthur Jacot, Franck Gabriel, and Clément Hongler. Neural tangent kernel: Convergence and generalization in neural networks. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Moongyu Jeon, Sangwoo Shin, Dongjae Jeon, and Albert No. Information-theoretic discrete diffusion. In *Advances in Neural Information Processing Systems*, 2025.

Mengni Jia, Mengyu Zhou, Yihao Liu, Xiaoxi Jiang, and Guanjun Jiang. Bringing stability to diffusion: Decomposing and reducing variance of training masked diffusion models. In *International Conference on Learning Representations*, volume 2026, pp. 109393–109428, 2026.

Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, Stefano Ermon, Aditya Grover, and Volodymyr Kuleshov. Mercury: Ultra-Fast Language Models Based on Diffusion. *arXiv:2506.17298*, 2025.

Jaeyeon Kim, Kulin Shah, Vasilis Kontonis, Sham Kakade, and Sitan Chen. Train for the worst, plan for the best: Understanding token ordering in masked diffusions. In *International Conference on Machine Learning*, 2025.

Jaeyeon Kim, Jonathan Geuter, David Alvarez-Melis, Sham M. Kakade, and Sitan Chen. Stop training for the worst: Progressive unmasking accelerates masked diffusion training. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=55tjQADd7K.

Daniel Kunin, Giovanni Luca Marchetti, Feng Chen, Dhruva Karkada, James Simon, Michael Deweese, Surya Ganguli, and Nina Miolane. Alternating gradient flows: A theory of feature learning in two-layer neural networks. In D. Belgrave, C. Zhang, H. Lin, R. Pascanu, P. Koniusz, M. Ghassemi, and N. Chen (eds.), *Advances in Neural Information Processing Systems*, volume 38, Main Conference, pp. 4377–4424. Curran Associates, Inc., 2025. doi: 10.52202/085713-0156. URL https://proceedings.neurips.cc/paper_files/paper/2025/file/06cbd2e81dfbd3bb4cb0abce95b32584-Paper-Conference.pdf.

Sanghyun Lee, Chunsan Hong, Seungryong Kim, Jonghyun Lee, Jongho Park, and Dongmin Park. Looped diffusion language models. *arXiv:2605.26106*, 2026.

Tao Li, Lei Tan, Zhehao Huang, Qinghua Tao, Yipeng Liu, and Xiaolin Huang. Low dimensional trajectory hypothesis is true: DNNs can be trained in tiny subspaces. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 45(3):3411–3420, 2023. doi: 10.1109/TPAMI.2022.3178101.

Bingbin Liu, Sébastien Bubeck, Ronen Eldan, Janardhan Kulkarni, Yuanzhi Li, Anh Nguyen, Rachel Ward, and Yi Zhang. TinyGSM: Achieving >80% on GSM8k with small language models. *arXiv:2312.09241*, 2023.

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

Aaron Lou, Chenlin Meng, and Stefano Ermon. Discrete diffusion modeling by estimating the ratios of the data distribution. In *International Conference on Machine Learning*, 2024.

Sam McCandlish, Jared Kaplan, Dario Amodei, and OpenAI Dota Team. An empirical model of large-batch training. *arXiv:1812.06162*, 2018.

Alexandru Meterez, Pranav Ajit Nair, Depen Morwani, Cengiz Pehlevan, Sham Kakade, and Alex Damian. A defense of the quadratic model. *arXiv:2607.21716*, 2026.

Thomas Pethick, Wanyun Xie, Roman Machacek, and Volkan Cevher. Optimistic dual averaging unifies modern optimizers. *arXiv:2605.11172*, 2026.

Boris T. Polyak. Some methods of speeding up the convergence of iteration methods. *USSR Computational Mathematics and Mathematical Physics*, 4(5):1–17, 1964.

Ritvik19. Sudoku-Dataset. Hugging Face Datasets, 2024. URL `https://huggingface.co/datasets/Ritvik19/Sudoku-Dataset`. Accessed September 2026.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T. Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *Advances in Neural Information Processing Systems*, 2024.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267. PMLR, 13–19 Jul 2025. URL `https://proceedings.mlr.press/v267/sahoo25a.html`.

Subham Sekhar Sahoo, Jean-Marie Lemercier, Zhihan Yang, Justin Deschenaux, Jingyu Liu, John Thickstun, and Ante Jukić. Scaling beyond masked diffusion language models. In *International Conference on Machine Learning*, 2026.

Kulin Shah, Nishanth Dikkala, Xin Wang, and Rina Panigrahy. Causal language modeling can elicit search and reasoning capabilities on logic puzzles. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis K. Titsias. Simplified and generalized masked diffusion for discrete data. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Andy Shih, Dorsa Sadigh, and Stefano Ermon. Training and inference on any-order autoregressive models the right way. In *Advances in Neural Information Processing Systems*, volume 35, pp. 2762–2775, 2022.

Shohei Taniguchi, Keno Harada, Gouki Minegishi, Yuta Oshima, Seong Cheol Jeong, Go Nagahara, Tomoshi Iiyama, Masahiro Suzuki, Yusuke Iwasawa, and Yutaka Matsuo. ADOPT: Modified Adam can converge with any β_2 with the optimal rate. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

Adrien Ali Taïga, James Assiene, Daniele Calandriello, Rahma Chaabouni, João Gante, Tamara von Glehn, Nate Keating, Chris Knutsen, Martin Kukla, Tianlin Liu, Ivan Lobov, Ofir Nabati, João Gabriel Oliveira, Nicolas Perez-Nieves, Nastasia Prutianova, Bobak Shahriari, Jean Tarbouriech, Pavel Tyletski, Çağlar Ünlü, Cindy Wu, Glenn Cameron, Jerome Connor, Sertan Girgin, Maarten Grootendorst, Alon Levkovitch, Eliya Nachmani, Omar Sanseviero, Piotr Stanczyk, Quentin Berthet, Andrew Campbell, Clément Crepy, Valentin De Bortoli, Arnaud Doucet, Romuald Elie, Alexandre Galashov, Klaus Greff, Alexis Jacq, David Ruhe, Yu-Han Wu, Sebastian Flennerhag, Brendan O'Donoghue, George Scrivener, and Shantanu Thakoor. DiffusionGemma Technical Report. *arXiv:2608.00146*, 2026.

Gilad Turok, Chris De Sa, and Volodymyr Kuleshov. DUEL: Exact likelihood for masked diffusion via deterministic unmasking. *arXiv:2603.01367*, 2026.

Shashanka Ubaru, Jie Chen, and Yousef Saad. Fast estimation of $\mathrm{tr}(f(A))$ via stochastic Lanczos quadrature. *SIAM Journal on Matrix Analysis and Applications*, 38(4):1075–1099, 2017.

Dimitri Von Rütte, Janis Fluri, Yuhui Ding, Antonio Orvieto, Bernhard Schölkopf, and Thomas Hofmann. Generalized interpolating discrete diffusion. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267. PMLR, 2025. URL https://proceedings.mlr.press/v267/von-rutte25a.html.

Dimitri von Rütte, Janis Fluri, Omead Pooladzandi, Bernhard Schölkopf, Thomas Hofmann, and Antonio Orvieto. Scaling behavior of discrete diffusion language models. In *International Conference on Learning Representations*, 2026.

Yuxiang Wang, Yu Xiang, Baojian Zhou, Qifang Zhao, Keyue Jiang, Yanghua Xiao, and Xiaoxiao Xu. On the trainability of masked diffusion language models via blockwise locality. *arXiv:2604.24832*, 2026.

Shuchen Xue, Tianyu Xie, Tianyang Hu, Zijin Feng, Jiacheng Sun, Kenji Kawaguchi, Zhenguo Li, and Zhi-Ming Ma. Any-order GPT as masked diffusion model: Decoupling formulation and architecture. In *International Conference on Machine Learning*, 2026.

Greg Yang, Edward J. Hu, Igor Babuschkin, Szymon Sidor, Xiaodong Liu, David Farhi, Nick Ryder, Jakub Pachocki, Weizhu Chen, and Jianfeng Gao. Tuning large neural networks via zero-shot hyperparameter transfer. In *Advances in Neural Information Processing Systems*, volume 34, pp. 17084–17097, 2021.

Jiacheng Ye, Jiahui Gao, Shansan Gong, Lin Zheng, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Beyond autoregression: Discrete diffusion for complex reasoning and planning. In *International Conference on Learning Representations*, 2025a.

Jiacheng Ye, Zhihui Xie, Lin Zheng, Jiahui Gao, Zirui Wu, Xin Jiang, Zhenguo Li, and Lingpeng Kong. Dream 7B: Diffusion large language models. *arXiv:2508.15487*, 2025b.

Guodong Zhang, Lala Li, Zachary Nado, James Martens, Sushant Sachdeva, George E. Dahl, Christopher J. Shallue, and Roger B. Grosse. Which algorithmic choices matter at which batch sizes? Insights from a noisy quadratic model. In *Advances in Neural Information Processing Systems*, volume 32, pp. 8196–8207, 2019.

A PROTOCOLS

Algorithm 1 summarizes how the quantities of Sections 3 and 4 are measured; the paragraphs below and Appendices A.2–D give the protocol of each step.

Algorithm 1 Measurements of Sections 3 and 4, carried out for each objective.

1: **Checkpoint.** Train with ADOPT for 3,000 updates, saving parameters and optimizer state, and take θ_0 at the matched held-out loss (Figure 1). From its frozen second moment and the group learning rates, form D, the coordinates $\delta = D^{-1/2}(\theta - \theta_0)$ and the inherited buffer $v_0 = \eta D^{-1/2} m_0$ (Eq. 15).

2: **Network and tangent model.** From θ_0, continue the network for $T = 320$ updates with batches of $B/2$, B and $4B$ sequences, and the tangent model $z_p(\theta_0) + J_p\delta$ with one and with four gradient batches per update (Tables 1 and 2).

3: **Quadratic model in all parameters.** Run Eq. 15 on F for T updates, with the batch gradient at update t equal to the checkpoint gradient on the network's minibatch plus $\widehat{G}_t\delta_t$, where $\widehat{G}_t$ averages the Gauss–Newton matrices of R fresh batches ($R = 4$ for AR, 16 for diffusion). With one and with four gradient batches per update, this gives P_B and P_{4B}, hence $N = \frac{4}{3}(P_{4B} - P_B)$ and $P = P_B + N$ (Appendix A.2).

4: **Basis of $\mathcal{S}$.** Extend the one-batch run to 640 updates and orthonormalize 120 of its iterates into Q. Estimate $Q^\top GQ$ on 8 fresh batches, taking the eigenvectors u_i from four of them and the curvatures λ_i from the other four, and estimate the coefficients g_i of g on 32 further batches (Appendices C.1–C.3).

5: **Attainable and noise-free decrease by curvature.** In the basis, $\Delta_{\mathcal{S},\leq\kappa} = \sum_{i:\,\lambda_i\leq\kappa} \frac{1}{2}\lambda_i^\dagger g_i^2$, and $P_{\leq\kappa}$ follows from Eq. 18 with $x_{i,0} = g_i/\lambda_i$ and $v_{i,0} = u_i^\top v_0$ (Figure 4).

6: **Noise cost by curvature.** From four probes $\xi = (\widehat{g} - \widehat{g}')/\sqrt{2}$, differences of independent batch gradients, run T Lanczos steps on the Gauss–Newton matrix of one fresh batch; Gauss quadrature gives $N_T = \mathrm{tr}[n_T(G)\Sigma_B]$ and its split by curvature (Appendix C.5).

7: **Corruption gradients.** Apply each of 64 corruptions, two at each of 32 stratified rates, to the same 1,024 held-out sequences, and compute AR's gradients on these sequences. Project every gradient onto Q, and estimate $\Delta_{\mathcal{S},\leq\kappa}(g_\omega)$, ℓ_ω, the pairwise cosines $\cos_{\leq\kappa}(g_\omega, g_{\omega'})$, $\Delta_{\mathcal{S},\leq\kappa}$ and $c_{\leq\kappa}$ in the metric $G^\dagger_{\leq\kappa}$ from products over distinct sequences and distinct corruptions (Eqs. 10–12; Appendix D).

Model and Training Models are 6.16M-parameter transformers with the architecture and initialization of von Rütte et al. (2026). They are trained on TinyGSM (Liu et al., 2023), in which a problem description precedes each target sequence and is neither corrupted nor predicted; the vocabulary is the 2,048-token TokenMonster vocabulary `englishcode-2048-clean-v1`[1] with beginning-of-sequence, end-of-sequence, padding and mask tokens added, 2,052 tokens in all. Runs use ADOPT (Taniguchi et al., 2024) with $\beta_1 = .98$, $\beta_2 = .999$, a batch of 4,096 sequences and a learning-rate parameter of .2, selected by a sweep of full training runs of masked and uniform diffusion. AR uses the same value without separate tuning, so any mistuning of AR makes the gap conservative. Learning rates are scaled per parameter group as in μP (Yang et al., 2021). Training uses bfloat16 mixed precision, a linear warm-up over 250 updates followed by a constant learning rate, and no gradient clipping. After warm-up it adds the weight-centering term of SODA (Pethick et al., 2026), which no continuation from the checkpoints uses. Uniform diffusion's network is conditioned on the corruption rate through an added embedding and predicts the clean token at every position; its optimizer, learning rates, warm-up and batch are those of masked diffusion. Every objective is trained for 3,000 updates.

Answers Given Prompts A training sequence is the beginning-of-sequence token, the problem and a separator, which form the prompt, followed by the target, the answer and the end-of-sequence token; sequences longer than 256 tokens are dropped rather than truncated. Grouping the 6,782,532 training sequences by their exact prompt, a prompt with k sequences and distinct targets of multiplicities $m_1, m_2, \ldots$ has empirical conditional entropy $\sum_j (m_j/k)\log(k/m_j)$. Its mean over training sequences is .073 nats per sequence, or .00053 nats per target token at the mean target length of 137.7 tokens. This is the minimum over predictors of AR's negative log-likelihood on the training data, and of the diffusion ELBOs, which are tight when the model equals the data distribution. Of the training sequences, 5.6% have a prompt with more than one distinct target. The measure counts exact duplicates, so it concerns the training data rather than the generator of TinyGSM on unseen prompts: among repeated prompts, 20% of target pairs are identical.

[1] `https://github.com/alasdairforsythe/tokenmonster`

Corruption Schedule Training draws rates uniformly from $[\tau_{\min}, 1 - \tau_{\min}]$ with $\tau_{\min} = \mathrm{sigmoid}(-9) \simeq 1.23 \times 10^{-4}$, which keeps the second moment of the masked weight $1/\tau$ finite; under $\tau \sim \mathcal{U}(0, 1)$ it is infinite.

Matched Checkpoints The analysis starts from one checkpoint per objective, saved with its optimizer state, at closely matched held-out loss (filled markers in Figure 1): AR at update 467, masked diffusion at update $1{,}300$ and uniform diffusion at update $1{,}700$, with losses of $.451$, $.467$ and $.451$ on the held-out set of $4{,}096$ sequences (for uniform diffusion, the negative ELBO); masked diffusion's higher loss makes the gap conservative.

Continuations Every network continuation runs for 320 updates from a checkpoint with ADOPT at the training learning rate $.2$ and $\beta_1 = .98$, the momentum buffer inherited from training, the second moment frozen at its checkpoint value, and fresh batches of $4{,}096$ training sequences with the training corruption. Uniform diffusion's continuations train on its denoising loss and also record its ELBO.

Batch-Size Continuations Three further network continuations, with batches of $2{,}048$, $4{,}096$ and $16{,}384$ sequences, half, one and four times the training batch B, measure how the network's decrease depends on the batch size. They are evaluated on the held-out set of $32{,}768$ sequences. If the network's noise cost were proportional to $1/B$, as the quadratic model's is (Section 3.2), the decreases at B and $4B$ would determine the decrease at $B/2$: $.2115$, $.0312$ and $.0193$ for AR, masked and uniform diffusion, against the measured $.2148$, $.0431$ and $.0287$.

Data Splits The $6{,}851{,}043$ sequences of the TinyGSM training set that fit in 256 tokens are split at random (seed 42) into a validation split of 1% and a training split of $6{,}782{,}532$ sequences. Held-out evaluations use three disjoint sets from the validation split, each sequence with one fixed corruption: $32{,}768$ sequences for the batch-size continuations, $4{,}096$ for the training runs' losses and the tangent model, and $8{,}192$ for the quadratic model. The crossed design of Section 4 applies 64 corruptions to each of $1{,}024$ sequences of the last set (Appendix D). The basis Q, the curvature estimates, the gradient means and the quadrature probes use fresh batches from the training split with the training corruption. No reported quantity reuses the sequences that selected its directions or estimated its curvatures.

Compute Every run uses one NVIDIA H100 GPU. Training a TinyGSM model for $3{,}000$ updates takes about 20 minutes, and a Surface Sudoku run of $5{,}000$ updates 11 to 25 minutes. The measurements of Sections 3 and 4 take roughly 25 GPU-hours for each diffusion objective and 10 for AR, most of them in the quadratic continuations in all parameters, which average the curvature of 16 batches per update for diffusion.

A.1 Uniform diffusion: evidence bound and reporting units

Evidence Bound At rate τ the forward process keeps a token with probability $1 - \tau$ and otherwise replaces it by one drawn uniformly from the vocabulary. The negative ELBO of Von Rütte et al. (2025) is minimized not by the denoising posterior $p(y_p \mid y^\tau, \tau)$ that our network learns but by the leave-one-out posterior $p(y_p \mid y^\tau_{\setminus p}, \tau)$, its unique minimizer for uniform noise (Gourevitch et al., 2026, Prop. 1), so evaluating the bound at the denoiser misreports it even for a perfect denoiser. Dividing the denoiser by the forward likelihood of the observed token and renormalizing gives the leave-one-out posterior; in logits, this lowers the observed token's logit by $\log\big(1 + V(1-\tau)/\tau\big)$ and leaves the others unchanged (Gourevitch et al., 2026, Prop. 5 and Eq. 13). We apply this conversion at every position and evaluate the bound on the result.

Reporting Units Every decrease and noise cost of uniform diffusion that is measured on its denoising loss is reported multiplied by 1.637, except in Table 2, which compares the models on the denoising loss itself. The factor is the slope of an affine fit of the ELBO against the denoising loss over the matched uniform run's nine saved checkpoints with denoising loss at most 1 (updates 800–$3{,}000$, on the held-out set of $4{,}096$ sequences); the intercept cancels in differences. Ratios such as $P/\Delta_{\mathcal{S}}$ and cosines carry no units, and curvatures remain those of the denoising loss, so the curvature cutoffs are unchanged. The network continuations' decreases are measured on the ELBO directly and are not converted.

A.2 QUADRATIC CONTINUATIONS IN ALL PARAMETERS

Protocol For each objective, the quadratic model is run in all n optimizer coordinates by the recursion of Eq. 15, with the network continuation's learning rate, horizon, frozen preconditioner and inherited momentum. Its batch gradient at each update is the checkpoint gradient on the network continuation's minibatch of $4{,}096$ training sequences at that update, plus $\widehat{G}_t\delta_t$, where $\widehat{G}_t$ is the mean Gauss–Newton matrix of R fresh batches of $4{,}096$ training sequences, drawn independently of the gradients and never reused. The run therefore carries the minibatches' sampling noise, and its decrease estimates $P - N$ rather than P. Each run tracks two values of R at once, 1 and 4 for AR and 4 and 16 for diffusion, the smaller using the first of the larger's curvature batches. Decreases are evaluated with the gradient and Gauss–Newton matrix of the held-out set of $8{,}192$ sequences, and each value is a single realization.

Curvature Averaging Sampling the curvature adds noise that lowers the expected decrease and that the model with fixed curvatures (Appendix C) does not have; suppressing it takes more averaging for diffusion. At update 320 the decrease rises with R for every objective: from $.124$ at $R = 1$ to $.129$ at $R = 4$ for AR, from divergence at $R = 1$ to $.027$ at $R = 4$ and $.031$ at $R = 16$ for masked diffusion, and from negative values up to $R = 8$ to $.023$ at $R = 16$ for uniform diffusion. We report each objective at the largest R measured, $R = 4$ for AR and $R = 16$ for diffusion.

Noise Removal The noise-free decrease P follows by extrapolation in the batch size. For each objective a second run averages, at every update, the gradients of four batches: the network continuation's batch and three further training batches. Each batch is normalized separately, so the average has a quarter of the gradient covariance at fixed parameters. When the two runs share their curvature draws, the iterate is, given these draws, affine in the sampled gradients, so the noise cost is proportional to $1/B$ and $P_B = P_{\infty,R} - k_R/B$ holds exactly, where $P_{\infty,R}$ retains the cost of curvature sampling at that R and k_R is a constant; the two batch sizes then determine $P_{\infty,R}$ (Table 1). On each objective's training loss, the quadratic model's gain from four gradient batches per update is $.85$, $.85$ and 1.03 times the network's gain from four times the batch at update 320 (Table 2).

Curvature Draws Only AR's two runs share their curvature draws. The diffusion objectives' four-batch runs draw their curvature batches independently of the single-batch runs, so for them the extrapolation holds only in expectation over the draws. For masked diffusion the difference is small: its two independent draws at $R = 4$ agree to within 1%, and at $R = 4$ a single-batch run that shares the four-batch run's draws gives $P = .0538$, against $.0545$ at $R = 16$. A check that holds the curvature fixed covers all three objectives: the noise-free decrease within $\mathcal{S}$ minus the noise cost by quadrature (Appendix C) gives $P - N = .130$, $.031$ and $.018$, against the continuations' $.129$, $.031$ and $.023$.

Horizon Both runs are evaluated every 10 updates, and the extrapolation gives P at each evaluation (Figure 3b). To span $\mathcal{S}$, the run with the larger curvature average is extended to 640 updates (Appendix C.1). Its decrease after u updates is estimated as $-g^\top\delta_u - \frac{1}{2}\delta_u^\top\bar{G}_u\delta_u$, with g the mean gradient of $131{,}072$ training sequences and $\bar{G}_u$ the mean Gauss–Newton matrix of the curvature batches drawn at update $u + 1$, which are independent of δ_u. At update 320 this estimate gives $.1287$, $.0289$ and $.024$, against $.1288$, $.0309$ and $.023$ on the held-out set. Averaged over updates 590–630 it gives $.146$, $.0335$ and $.017$, against $.127$, $.0286$ and $.020$ over updates 280–320. The noise cost N_T increases with T toward $N^\infty := \sum_i N_i^\infty$ (Eq. 19), and the quadrature gives $N_{320} \geq .83\, N^\infty$ for every objective (Appendix C.5), so after update 320 the noise cost grows by at most $N^\infty - N_{320} \leq .21\, N_{320}$. The growth of P from update 320 to 630 is therefore at least that of $P - N$ and exceeds it by at most $.21\, N_{320}$. With P and N at update 320 from Table 1 and the growth of $P - N$ from the averages above, AR's noise-free decrease at update 630 is at least 2.4 times masked diffusion's and 2.3 times uniform diffusion's.

Tangent Model The quadratic model is the second-order expansion, at the checkpoint, of the tangent model's cross-entropy (Section 3). We also continue the tangent model itself, whose features are fixed but whose loss is the exact cross-entropy, from each checkpoint with the continuation's optimizer, once with one gradient batch per update and once with the average of four, as in the quadratic model (Table 2). Its decrease exceeds the quadratic model's for every objective.

Table 1: Decrease at update 320 by batch size (Section 3.3), in nats per target token. Quadratic model: one and four gradient batches per update, the noise cost $N = \frac{4}{3}(P_{4B} - P_B)$ and the noise-free decrease $P = P_B + N$, which the model's $1/B$ law makes exact. Network: batches of 2,048, 4,096 and 16,384 sequences; its decrease does not follow $1/B$ (Appendix A), so it is not extrapolated. Uniform diffusion's network values are measured on the ELBO directly (Appendix A.1).

	$P_{B/2}$	P_B	P_{4B}	N	P
quadratic, AR ($R = 4$)	–	.1288	.1351	.0084	.1372
quadratic, masked ($R = 16$)	–	.0309	.0486	.0236	.0545
quadratic, uniform ($R = 16$)	–	.0228	.0513	.0379	.0607
network, AR	.2148	.2213	.2287	–	–
network, masked	.0431	.0589	.0796	–	–
network, uniform	.0287	.0503	.0737	–	–

Table 2: Fixed features with and without the quadratic approximation, at update 320, in nats per target token on each objective's training loss (for uniform diffusion, its denoising loss): the decrease with one gradient batch per update, and the gain from four (for the network, from a batch of 16,384 sequences). The tangent model, the quadratic model and the network are evaluated on the held-out sets of 4,096, 8,192 and 32,768 sequences.

	decrease			gain from four batches		
	AR	masked	uniform	AR	masked	uniform
tangent model	.1500	.0448	.0187	.0039	.0143	.0168
quadratic model	.1288	.0309	.0140	.0063	.0177	.0174
network	.2213	.0589	.0329	.0074	.0208	.0169

B Derivations

B.1 Curvature and the attainable decrease

Logit Cross-Entropy For logits $z \in \mathbb{R}^V$ and target y, the cross-entropy $\ell(z, y) = -z_y + \log \sum_k e^{z_k}$ has gradient $r = \pi - \mathbf{1}_y$ and Hessian $\Lambda = \operatorname{diag}(\pi) - \pi\pi^\top$ in z, with $\pi = \operatorname{softmax}(z)$; the Hessian does not depend on the target. For any $a \in \mathbb{R}^V$, $a^\top \Lambda a = \sum_k \pi_k a_k^2 - (\sum_k \pi_k a_k)^2 = \operatorname{Var}_{k\sim\pi}(a_k) \geq 0$, so $\Lambda \succeq 0$, and since every $\pi_k > 0$, $a^\top \Lambda a = 0$ exactly when a is constant across classes. With the features fixed, the logits $z_p(\theta_0) + J_p\delta$ are affine in δ, so $\nabla_\delta \ell = J_p^\top r_p$ and $\nabla_\delta^2 \ell = J_p^\top \Lambda_p J_p$ with no further term; averaging with the weights of Eq. 1 gives Eq. 2.

Quadratic Optimum For any $d \in \mathbb{R}^n$, the identity above with $a = J_p d$ gives

$$d^\top G d = \frac{1}{\bar{L}} \mathbb{E} \sum_p w_p \operatorname{Var}_{\pi_p}(J_p d) \geq 0.$$

If $d \in \ker G$, every term with positive weight vanishes almost surely, so $J_p d$ is constant across classes, and since the entries of r_p sum to zero, $g^\top d = \bar{L}^{-1} \mathbb{E} \sum_p w_p r_p^\top J_p d = 0$. Hence $g \perp \ker G$, so $g \in \operatorname{range} G$ and $\delta_* = -G^\dagger g$ solves $g + G\delta_* = 0$; by convexity it is a global minimizer, and every other minimizer differs from it by an element of $\ker G$. Since $G\delta_* = -g$,

$$F(\delta_*) - F(0) = g^\top \delta_* + \frac{1}{2}\delta_*^\top G \delta_* = \frac{1}{2} g^\top \delta_* = -\frac{1}{2} g^\top G^\dagger g.$$

Minimizing F over the span of the eigenvectors with $\lambda_i \leq \kappa$ gives $\Delta_{\leq\kappa}$ in the same way; directions with $\lambda_i = 0$ have $g_i = 0$ and contribute nothing. An invertible linear change of coordinates leaves $\min F$, and hence Δ, unchanged even when G is singular, but it changes the eigenvalues of G, and with them $\Delta_{\leq\kappa}$ and the dynamics over a finite horizon.

B.2 Momentum dynamics and gradient noise

Optimizer Coordinates ADOPT (Taniguchi et al., 2024) normalizes each gradient by a second-moment estimate that excludes the current batch, so each normalized gradient has the preconditioned population gradient as its conditional mean. With that estimate held at its checkpoint value and the learning-rate ratios of the parameter groups absorbed into the positive diagonal preconditioner D, a

continuation updates $m_{t+1} = \beta m_t + (1-\beta) D\,\widehat{g}_{\theta,t}$ and $\theta_{t+1} = \theta_t - \eta m_{t+1}$, with no active clipping or weight decay. In $\delta_t = D^{-1/2}(\theta_t - \theta_0)$, where the batch gradient is $\widehat{g}_t = D^{1/2}\widehat{g}_{\theta,t}$, and with $v_t := \eta D^{-1/2} m_t$, this is heavy-ball momentum,

$$v_{t+1} = \beta v_t + \alpha\,\widehat{g}_t, \qquad \delta_{t+1} = \delta_t - v_{t+1}, \qquad \alpha = \eta(1-\beta), \quad \delta_0 = 0. \tag{15}$$

Writing the batch gradient of the quadratic as its mean plus a sampling error, $\widehat{g}_t = \nabla F(\delta_t) + \xi_t$, gives the recursion solved next. For fresh batches the sampling error has zero mean given the past and is uncorrelated across updates, and the model holds its covariance fixed (Section 3.2).

Per-Direction Solution With $\nabla F(\delta) = G(\delta - \delta_*)$ (Eq. 3) and $u_i^\top G = \lambda_i u_i^\top$, projecting Eq. 15 onto u_i gives one recurrence per direction. Write $v_{i,t} := u_i^\top v_t$ and $\xi_{i,t} := u_i^\top \xi_t$, whose variance $\sigma_i^2 := u_i^\top \Sigma_B u_i$ we measure on training batches at the checkpoint (Appendix C.3). Then

$$v_{i,t+1} = \beta v_{i,t} + \alpha\,(\lambda_i x_{i,t} + \xi_{i,t}), \qquad x_{i,t+1} = x_{i,t} - v_{i,t+1}. \tag{16}$$

We write $P_{i,T}$ and $N_{i,T}$ for the terms of Eq. 7 after T updates; P_i and N_i in the main text are their values at $T = 320$, and $P_{i,T}/\Delta_i = 1 - \bar{x}_{i,T}^2/x_{i,0}^2$ is the fraction realized. Eliminating the buffer leaves, in one direction with its index dropped,

$$x_{t+1} = a\,x_t - \beta\,x_{t-1} - \alpha\,\xi_t, \qquad a = 1 + \beta - \alpha\lambda, \tag{17}$$

whose impulse response $h_0 = 1$, $h_1 = a$, $h_j = a\,h_{j-1} - \beta\,h_{j-2}$ is a combination of the j-th powers of the two roots of $\zeta^2 - a\zeta + \beta = 0$, the roots of the heavy-ball analysis of Zhang et al. (2019, Thm. 1); both lie inside the unit circle exactly when $0 < \alpha\lambda < 2(1+\beta)$. For $\alpha\lambda < (1-\sqrt{\beta})^2$, that is, below curvature .025 at $\alpha = .004$ and $\beta = .98$, the roots are real and the larger is about $1 - \alpha\lambda/(1-\beta) = 1 - \eta\lambda$; for $(1-\sqrt{\beta})^2 < \alpha\lambda < (1+\sqrt{\beta})^2$, which at $\beta = .98$ extends almost to the stability limit, they are complex with modulus $\sqrt{\beta}$. By superposition, with uncorrelated noise,

$$x_T = \bar{x}_T - \alpha \sum_{t<T} h_{T-1-t}\,\xi_t, \qquad \mathrm{Var}(x_T) = \alpha^2\sigma^2 \sum_{j<T} h_j^2,$$
$$\bar{x}_T = (h_T - \beta h_{T-1})\,x_0 - \beta h_{T-1}\,v_0, \tag{18}$$

where the noise-free trajectory $\bar{x}_T$ starts from the checkpoint's displacement x_0 and inherited buffer v_0; Zhang et al. (2019) start from a zero buffer. Figure 4 and Table 3 use the checkpoint buffer. The noise cost after T updates and its limit are

$$N_{i,T} = \frac{\alpha^2\lambda_i\sigma_i^2}{2} \sum_{j=0}^{T-1} h_{i,j}^2, \qquad N_i^\infty = \frac{\alpha\sigma_i^2(1+\beta)}{2(1-\beta)[2(1+\beta) - \alpha\lambda_i]}, \tag{19}$$

where N_i^∞ is the steady-state risk of Zhang et al. (2019, Thm. 1) in our notation (their learning rate is our α and their c_i/B our σ_i^2). Correlations between different directions do not enter the expected loss, because the quadratic is diagonal in this basis. Low-curvature directions take longer than the horizon to reach the stationary spread, so the tables use the finite-horizon $N_{i,T}$.

B.3 Averaging over corruptions

Alignment By Eqs. 11 and 12, $c_{\le\kappa}$ is a mean of cosines weighted by the products $\Delta_{\mathcal{S},\le\kappa}(g_\omega)^{1/2}\Delta_{\mathcal{S},\le\kappa}(g_{\omega'})^{1/2} \ge 0$, and every cosine is at most 1, so $c_{\le\kappa} \le 1$, with equality exactly when $\cos_{\le\kappa}(g_\omega, g_{\omega'}) = 1$ for almost every pair of corruptions of nonzero contribution, that is, when the vectors $(G_{\le\kappa}^\dagger)^{1/2} g_\omega$ are positive multiples of one vector.

B.4 Estimation

Unbiased Estimators For independent, identically distributed gradients $\hat{h}_1, \ldots, \hat{h}_n$ with mean h and a fixed matrix $A \succeq 0$, the mean cross product $U_n := \frac{1}{n(n-1)} \sum_{a\ne b} \hat{h}_a^\top A \hat{h}_b$ satisfies $\mathbb{E}\,U_n = h^\top A h$, and $U_n = \bar{h}^\top A \bar{h} - \frac{1}{n}\,\mathrm{tr}(A\widehat{C})$, where $\bar{h}$ is the sample mean and $\widehat{C}$ the sample covariance; this is the variance correction of the gradient means (Appendix C.3). With $A = G_{\le\kappa}^\dagger$ and the gradients of distinct sequences under one corruption ω, $U_n/2$ estimates $\Delta_{\mathcal{S},\le\kappa}(g_\omega)$ without bias, and products of the gradients of distinct sequences under two corruptions $\omega \ne \omega'$ estimate $g_\omega^\top G_{\le\kappa}^\dagger g_{\omega'}$. For independent $\hat{h}, \hat{h}'$ with the same distribution, $(\hat{h} - \hat{h}')/\sqrt{2}$ has mean zero and the covariance

of $\hat{h}$, which gives the probes of the quadrature (Appendix C.5). These identities require A to be independent of the gradients; the curvatures are estimated on training batches disjoint from every readout (Appendix A).

Rate Stratification Let a gradient h be the mean over rates of its rate-conditional means h_τ, as the population gradient is, and let μ_k be the mean of h_τ over stratum k of $K = 32$ equal-probability rate strata, so that $h = K^{-1} \sum_k \mu_k$ and, for any fixed $A \succeq 0$,

$$h^\top A h = \frac{1}{K^2} \Big[\sum_k \mu_k^\top A \mu_k + \sum_{k \neq l} \mu_k^\top A \mu_l \Big].$$

Products of the gradients of corruptions from different strata estimate the terms with $k \neq l$ without bias. The crossed design has one rate per stratum, so products of the gradients of distinct corruptions within stratum k estimate $\mu_k^\top A \mu_k + \mathbb{E}_{\tau \in k} (h_\tau - \mu_k)^\top A (h_\tau - \mu_k)$.

C Measurement by Curvature

C.1 Basis of the path span

The quadratic continuation in all parameters (Appendix A.2) with the larger curvature average, $R = 4$ for AR and $R = 16$ for diffusion, is extended from 320 to 640 updates with fresh training batches. Its iterates at updates $4, 8, \ldots, 320$ and $328, 336, \ldots, 640$, 120 in all, span the basis Q, which the eigendecomposition of their Gram matrix orthonormalizes. Each basis vector is multiplied by the Gauss–Newton matrices of 8 fresh batches of 4,096 training sequences, drawn independently of the path, so every entry of the projected curvature $Q^\top G Q$ is estimated without bias.

C.2 Held-out curvatures

The directions are the eigenvectors of the projected curvature of the first four batches, and each direction's curvature is its Rayleigh quotient under the other four: an eigenvalue read from the batches that chose its direction is biased low at the flat end. The quadratic model in the basis is therefore a cross-fitted diagonal approximation to $Q^\top G Q$. The held-out curvatures range from .010 to 166 for AR, .014 to 163 for masked and .017 to 633 for uniform diffusion, all positive and below the stability limit $\alpha \lambda_i < 2(1 + \beta)$ of Appendix B.2; 15 of each objective's 120 directions lie below curvature 1.

C.3 Gradient means and noise

Gradient coefficients use 32 training batches of 4,096 sequences with the training corruption, independent of the curvature batches. The squared mean coefficient subtracts the variance of the batch mean, $\hat{g}_i^2 = \max(\bar{g}_i^2 - \hat{\sigma}_i^2/32, 0)$ (Appendix B.4), where the variance $\hat{\sigma}_i^2$ of the coefficient across batches estimates the per-update noise variance σ_i^2. The buffer's coefficients are the projections of the checkpoint's momentum buffer. The noise cost comes not from these variances but from the quadrature of Appendix C.5, which covers all parameters.

C.4 What the path span holds

Restricted to the basis, the quadratic model gives $P = .1371$, $.0527$ and $.0601$ at update 320, against .1372, .0545 and .0607 in all parameters (Table 1). Its attainable decrease $\Delta_{\mathcal{S}}$ is a lower bound on Δ up to the estimation error of the cross-fitted diagonal curvature. Keeping one position every 32 updates retains 97–99.5% of $\Delta_{\mathcal{S}}$, so the length of the path, not the density of its positions, limits it. The basis holds 13%, 24% and 63% of the squared norm of the population gradient. Since the basis reproduces P, the remainder contributes little to the decrease within the horizon.

C.5 Noise cost by quadrature

The noise cost after T updates is a trace, $N_T = \mathrm{tr}[n_T(G)\Sigma_B]$, where $n_T(\lambda) = \frac{1}{2}\alpha^2 \lambda \sum_{j<T} h_j(\lambda)^2$, the cost of unit noise variance at curvature λ (Eq. 19), is a polynomial of degree $2T - 1$ in λ. For a probe ξ with covariance Σ_B, $\mathbb{E}\, \xi^\top n_T(G) \xi = N_T$, and T Lanczos steps on G started from ξ evaluate

$\xi^\top n_T(G)\xi$ exactly, because Gauss quadrature with T nodes is exact for polynomials of degree below $2T$ (Ubaru et al., 2017, Sec. 4.1). Unlike their isotropic probes, ours have the covariance of the gradient noise. Each probe is the difference of the gradients of two independent batches of $4{,}096$ training sequences divided by $\sqrt{2}$, which has covariance Σ_B (Appendix B.4), and G is estimated on one further batch of $4{,}096$ training sequences, the same for every probe. We run $T = 320$ steps from four probes per objective, reorthogonalizing every Lanczos vector. The quadrature needs no basis, and its nodes and weights also split the noise power and cost by curvature (Table 3) and give the stationary value N^∞. It gives $N_{320} = .0070$, $.0217$ and $.042$ (standard errors across probes $.0001$, $.0009$ and $.0015$), 84%, 92% and 111% of the noise cost of the continuations in all parameters (Table 1), which average 4 or 16 curvature batches per update instead of holding one fixed. By update 320 each objective has incurred between 83% and 89% of N^∞.

C.6 Gradient noise along the network's path

The model of Section 3.2 holds the noise covariance at its checkpoint value, which we test on the path the network takes. For 32 training batches of $4{,}096$ sequences, independent of the run's own batches, we compute each batch's gradient in the optimizer's coordinates (Eq. 15) at the checkpoint and at the network continuation's states after 80, 160 and 320 updates, with the same sequences and corruptions at every state. The noise power of a state, the trace over all parameters of the covariance of these gradients across batches, does not grow with the displacement: after 80, 160 and 320 updates it is $.80$, $.78$ and $.84$ of its checkpoint value for AR, $.92$, $.90$ and $.83$ for masked and $.84$, $.79$ and $.82$ for uniform diffusion.

C.7 The quadratic model by curvature band

Each direction u_i of $\mathcal{S}$ follows Eq. 16 from the checkpoint's displacement and buffer (Algorithm 1), and the noise cost of the quadrature (Appendix C.5) is assigned to bands by the curvature of its nodes. Table 3 splits the totals of Figure 4 into six curvature bands. $\mathcal{S}$ has no direction below curvature $.01$, so the lowest band holds only noise, and only in the band from $.01$ to $.1$ does momentum leave more than a few percent of $\Delta_{\mathcal{S}}$ unrealized within 320 updates.

Table 3: The quadratic model by curvature band, in nats per target token: attainable decrease $\Delta_{\mathcal{S}}$ and fraction realized in 320 noise-free updates $P/\Delta_{\mathcal{S}}$ within $\mathcal{S}$, and finite-horizon noise cost N over all parameters from the quadrature (Eq. 7); the totals are those of Figure 4.

	attainable Δ			realized, P/Δ			noise cost N (10^{-3})		
curvature	AR	masked	uniform	AR	masked	uniform	AR	masked	uniform
below .01	–	–	–	–	–	–	.25	1.08	1.19
.01–.1	.1324	.0425	.0418	.86	.90	.93	1.13	2.94	7.46
.1–1	.0231	.0088	.0121	1.00	1.00	1.00	1.98	4.37	6.86
1–10	.0003	.0027	.0040	.97	.99	1.00	2.04	5.01	6.63
10–100	.0005	.0023	.0020	1.00	1.00	1.00	1.24	4.77	6.14
above 100	.0001	.0009	.0032	1.00	1.00	1.00	.41	3.56	13.79
all	.1563	.0572	.0631	.88	.92	.95	7.05	21.73	42.08

D Averaging over Corruptions

Crossed Design The crossed design uses $1{,}024$ sequences of the held-out set of $8{,}192$ sequences. One rate is drawn in each of 32 equal-probability strata of the truncated rate schedule, the same 32 rates, from $.009$ to 1.00, for both diffusion objectives, and two corruptions are drawn at each rate: a set of masked positions for masked diffusion, and a set of positions with the tokens that replace them for uniform diffusion. Each of the 64 corruptions is applied to all $1{,}024$ sequences. Half the mean of $g_{s\omega}^\top G_{\le\kappa}^\dagger g_{t\omega}$ over distinct sequences $s \neq t$ estimates $\Delta_{\mathcal{S},\le\kappa}(g_\omega)$, and half the stratified mean over distinct sequences under distinct corruptions estimates $\Delta_{\mathcal{S},\le\kappa}$ (Appendix B.4). The pairwise cosines and the contributions in $c_{\le\kappa}$ use the square roots of the per-corruption estimates. On the same sequences, each with 32 independent corruptions, the objective's own estimate of $\Delta_{\mathcal{S},\le\kappa}$ is within 6% of the crossed design's below curvatures $.1$ and 1 (Table 4).

Table 4: The crossed design below three curvature cutoffs, with bootstrap standard errors: its attainable decrease $\Delta_{\mathcal{S},\leq\kappa}$, in nats per target token; the objective's own $\Delta_{\mathcal{S},\leq\kappa}$, measured on the same sequences with independent corruptions; and the alignment $c_{\leq\kappa}$ of the corruptions' gradients (Eq. 12). Uniform diffusion's decreases are scaled by 1.637 (Appendix A.1). AR's $\Delta_{\mathcal{S},\leq\kappa}$ is $.123 \pm .007$, $.145 \pm .008$ and $.146 \pm .008$.

	masked			uniform		
κ	$\Delta_{\mathcal{S},\leq\kappa}$	own $\Delta_{\mathcal{S},\leq\kappa}$	$c_{\leq\kappa}$	$\Delta_{\mathcal{S},\leq\kappa}$	own $\Delta_{\mathcal{S},\leq\kappa}$	$c_{\leq\kappa}$
.1	$.039 \pm .002$	$.041 \pm .002$	$1.00 \pm .00$	$.043 \pm .002$	$.042 \pm .002$	$1.00 \pm .00$
1	$.047 \pm .002$	$.050 \pm .002$	$1.00 \pm .00$	$.056 \pm .002$	$.054 \pm .002$	$.99 \pm .00$
∞	$.052 \pm .002$	$.056 \pm .002$	$.94 \pm .01$	$.065 \pm .003$	$.063 \pm .003$	$.85 \pm .00$

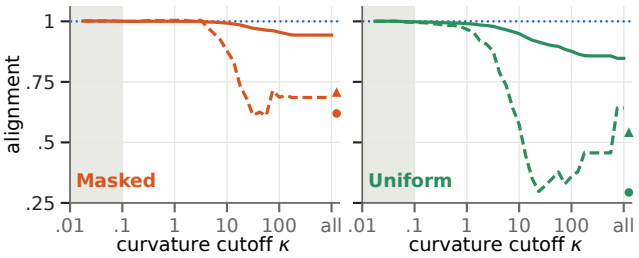


Figure 7: The alignment of the 64 corruptions' gradients against the curvature cutoff κ for (a) masked and (b) uniform diffusion: $c_{\leq\kappa}$ in the metric $G^{\dagger}_{\leq\kappa}$ (solid; Eq. 12) and in the identity metric (dashed), both within $\mathcal{S}$, and in the identity metric over all parameters at the right edge, in δ (circle) and θ (triangle). $\mathcal{S}$ has no direction below curvature .01. Shading: $.01 \leq \kappa \leq .1$, as in Figure 4.

Alignment by Curvature The pairwise cosines $\cos_{\leq\kappa}(g_\omega, g_{\omega'})$ are at least .97 below curvature .1 for both objectives, and at least .94 for masked and .88 for uniform diffusion below curvature 1. The alignment $c_{\leq\kappa}$ is at least .99 up to curvature 1, which includes the band $.01 \leq \kappa \leq .1$ where AR's extra attainable decrease lies, and falls only at higher curvatures, to .94 for masked and .85 for uniform diffusion over all curvatures (solid in Figure 7). The alignment is not an artifact of the metric: with the identity in place of $G^{\dagger}_{\leq\kappa}$, $c_{\leq\kappa} = g^\top g / \big(\mathbb{E}_\omega (g_\omega^\top g_\omega)^{1/2}\big)^2$ over the directions of $\mathcal{S}$ with curvature at most κ, it is again 1.00 below curvature .1 for both objectives (dashed), and it falls at high curvatures, to .69 and .64 over all of $\mathcal{S}$, because the identity weights each stiff direction by 1 rather than $1/(2\lambda_i)$. Over all parameters, the identity gives .62 and .29 in the coordinates δ of the quadratic model and .71 and .54 for the raw gradients in θ, without the preconditioner; these two are computed from sums over the 1,024 sequences and have no bootstrap. The corruptions' gradients therefore differ along the stiff directions, where $\Delta_{\mathcal{S}}$ is small (Figure 4), and agree along the directions where the gap lies.

Uncertainty Standard errors come from 200 bootstrap resamples of the sequences, entered as multiplicity weights so that no cross product pairs a sequence with a copy of itself. They do not include the sampling of the 64 corruptions.

Corruption Rates Table 5 summarizes the corruptions by rate quartile below curvature .1: their attainable decreases $\Delta_{\mathcal{S},\leq.1}(g_\omega)$, their losses ℓ_ω and their contributions per nat σ_ω.

E Surface-Sudoku Setup

Encoding and Data The cipher $(r, q) \in \{0, \ldots, 8\} \times \{0, 1\}$ of each box of each board is set by a deterministic hash of the complete board and the box index. A complete triplet (a, b, c) of Eq. 14 reveals both its digit, $d = (a + b + c) \bmod 9$, and its box's cipher, $r = (a - d) \bmod 9$ and $q = (a - b) \bmod 9$. The input is the resulting 243-character string, with no digit or cipher labels; only the digit-target runs receive digit labels, and only in training.

We take the first 1,000,000 solutions of the first training file and the 49,000 solutions of the first validation file of the `Ritvik19/Sudoku-Dataset` (Ritvik19, 2024); no validation solution appears in this training set. The first 8,192 validation boards give the learning curves and the next 32,768 the held-out evaluation. Every comparison of endpoints uses update 5,000, without selecting an earlier checkpoint.

Table 5: Single corruptions by rate quartile, below curvature .1: $\Delta_{\mathcal{S},\leq.1}(g_\omega)$ in nats per target token, averaged over the quartile's 16 corruptions; the mean loss ℓ_ω and the quartile's share of the total loss; and σ_ω (Eq. 13), the quartile's mean contribution $\Delta_{\mathcal{S},\leq.1}(g_\omega)^{1/2}$ divided by its mean loss, relative to $\Delta^{1/2}_{\mathcal{S},\leq.1,\mathrm{AR}}/\mathcal{L}$, so that the mean of σ_ω over the quartiles, weighted by their losses, is $(\Delta_{\mathcal{S},\leq.1}/\Delta_{\mathcal{S},\leq.1,\mathrm{AR}})^{1/2} = .56$ and .59. Uniform diffusion's decreases and losses are measured on its denoising loss and scaled by 1.637 (Appendix A.1). AR's gradient has $\Delta_{\mathcal{S},\leq.1} = .123$ at a loss of .44 nats per target token.

	masked				uniform			
rates	decrease	loss	share	σ_ω	decrease	loss	share	σ_ω
1.00–.78	.109	1.15	65%	.36	.180	1.70	71%	.43
.74–.52	.075	.43	24%	.79	.088	.54	22%	.91
.47–.25	.016	.13	8%	1.14	.012	.15	6%	1.19
.22–.01	.005	.07	4%	1.24	.001	.02	1%	1.43

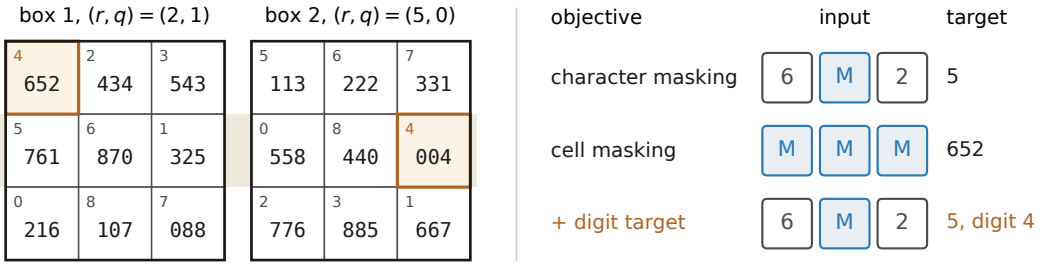


Figure 8: **The cipher changes from box to box, while the Sudoku constraints act across boxes.** Left: two boxes of one solved board, each cell showing its digit (small) and its three characters (large); the digit 4 is written as 652 in box 1 and as 004 in box 2, and the shaded row runs across both boxes. Right: what each objective supervises in the first cell of box 1 (M: mask).

Corruption and Prediction Both masking objectives draw one rate τ per board, stratified uniformly on $[.001, 1]$ across the batch. Character masking then masks each of the 243 characters independently with probability τ, and cell masking each of the 81 cells, hiding its three characters together. Each loss sums the cross-entropies of the masked targets, weighted by $1/\tau$, and divides by 243, so both are negative ELBOs per character. The character model's nine-way readout is tied to the character embeddings. The cell model also reads the 243 characters as separate tokens, with no cell embeddings; its output head concatenates a cell's three final hidden states, applies LayerNorm and predicts a joint distribution over the 729 triplets, and its cross-entropy is counted once per masked cell.

Architecture and Training Every model is a 16-layer transformer of width 256. The standard initialization draws each weight with standard deviation $1/\sqrt{\text{fan-in}}$, and the small initialization uses the GPT-2 constant .02; the two initializations of an objective start from the same random matrices up to scale and see the same corruptions. Every run uses AdamW with learning rate .001, weight decay .1 and batches of 512 boards, and processes the same board batches for 5,000 updates.

AR Reference The ARM reads the same 243-character strings with causal attention and predicts each character from its prefix in row-major order; its loss is the exact negative log-likelihood per character. It is trained with the optimizer, board stream and number of characters of the masking runs.

Digit Supervision The auxiliary head a_ψ applies LayerNorm and an affine nine-way classifier to each final hidden state h_j and is used only in training. With M_j indicating that character j is masked and $d_{c(j)}$ the clean digit of its cell, training minimizes

$$\mathcal{L}_{\mathrm{char}} + \frac{\sum_j M_j \ \mathrm{CE}(a_\psi(h_j), d_{c(j)})}{\max(1, \sum_j M_j)}. \tag{20}$$

with sums over the training batch: the auxiliary term has coefficient one and no inverse-rate weight, and every masked character of a cell receives its cell's digit as target. The digit is a function of the

clean triplet, the sum of its characters modulo 9, so the target adds no information beyond the clean board; it supervises the latent directly. Evaluation reports $\mathcal{L}_{\text{char}}$ alone.

Digit-Target Runs The runs with the digit target from update 0 share the random matrices and the board and mask streams of the plain run at the same initialization scale. The branched run resumes the standard-initialization character-masking run at update 2,000, once its plateau has formed, with the full parameter and AdamW state, including the step count. It adds the digit head, with fresh moments for the head only, and the loss of Eq. 20, and continues on the same board and mask stream to update 5,000 at learning rate .001 without a new warm-up. A branch resumed in the same way without the head reproduces the original run exactly at all 12 monitoring evaluations, so the original run serves as the control. On the 32,768 evaluation boards with paired corruptions, the branched run's endpoint is lower by .1232 nats per character (95% interval $[.1213, .1251]$), or 20.8%.

Adaptation from AR The adapted run copies every parameter of the standard-initialization ARM at update 5,000 into the character-masking model and follows Gong et al. (2025) in two respects. The readout is shifted, so that the output at position i predicts character $i+1$ and the adapted model at update 0 reproduces the ARM's next-character predictions; and attention is annealed from causal to bidirectional. Where they unmask a random, linearly growing share of future positions, we use a band: query i attends to keys $j \le i + w$, with w growing linearly from 0 to the full length over the first 1,000 updates. The ARM never reads the mask token, so we set the mask embedding to the mean of the nine character embeddings. Training otherwise matches the plain character runs: fresh AdamW state, the same schedule, and the same board and mask streams for 5,000 updates. Paired with the plain standard-initialization run, the adapted endpoint is lower by .0858 nats per character (95% interval $[.0845, .0871]$).

Adaptation from Cell Masking This run copies every parameter of the small-initialization cell-masking model at update 5,000, including its joint head and its trained mask embedding, and predicts each character as a marginal of the joint head. For cell c, let $p_c(t)$ be the head's distribution over the triplets $t = (t_1, t_2, t_3) \in \{0, \dots, 8\}^3$ and V_c the set of triplets that agree with the cell's visible characters. Character k of the cell is predicted as

$$q_{c,k}(v) = \frac{\sum_{t \in V_c:\, t_k = v} p_c(t)}{\sum_{t \in V_c} p_c(t)}, \qquad v \in \{0, \dots, 8\}. \tag{21}$$

For a cell whose three characters are all masked, V_c contains every triplet and $q_{c,k}$ is the donor's own marginal. The loss is the character-masking loss of these predictions, so the run differs from the plain character runs only in its readout, the donor's joint head; the optimizer, with fresh AdamW state, the schedule and the board and mask streams are theirs, for 5,000 updates. With the character model's nine-way readout, tied to the character embeddings, in place of the joint head, the same parameters reach only .585 and complete 5 of 128 puzzles. Paired with the plain small-initialization run, the adapted endpoint is lower by .1716 nats per character (95% interval $[.1689, .1743]$).

Generation and Solving Each model generates 128 boards without clues, and each diffusion model completes the same 128 held-out puzzles from their clues. A valid board satisfies the Sudoku constraints with one consistent cipher in every box; a valid completion also keeps every clue, and any valid solution counts. The diffusion decoders reveal one unit, a character or a cell, per forward pass: the most confident remaining unit, with its most likely value for completion and, for generation, the most confident of the proposals sampled at temperature one for every remaining unit. The ARM samples ancestrally at temperature one. Since it cannot condition on clues later in its order, we train a second ARM of the same architecture for completion, which reads a masked copy of the board as an unscored prefix: each cell of the copy is kept with probability $1 - \tau$, with $\tau \sim \mathcal{U}(0, 1)$ drawn per board, and the other cells' characters are the mask token. Its input is the prefix followed by the board, its loss is the next-character loss of the board alone, and its recipe and board stream are those of the standard-initialization ARM.

AR Losses and Completion Figure 6c reports completion from clues, since without clues most models generate few valid boards (Table 6), and its AR bar comes from a different ARM than the AR loss of Figure 6a,b. The ARM of the loss cannot condition on clues later in its order, so completion uses a second ARM that reads the clues as a prefix (Generation and Solving, above); neither completes more than a few puzzles. Equal loss therefore does not imply equal completion: the plain ARM and cell masking from the small initialization both reach .321, but the ARMs decode

Model	Loss	Valid generation	Valid completion
Objectives and initialization scales			
Character masking: standard init	0.592	0/128	3/128
Character masking: small init	0.592	0/128	2/128
Cell masking: standard init	0.582	0/128	4/128
Cell masking: small init	0.321	101/128	126/128
Autoregressive: standard init	0.321	5/128	2/128
Autoregressive: small init	0.328	0/128	—
Character masking with a digit target or adapted from a trained model			
Digit target from update 2,000: standard init	0.469	13/128	111/128
Digit target from update 0: small init	0.445	34/128	123/128
Digit target from update 0: standard init	0.503	0/128	53/128
Adapted from the standard-init ARM	0.506	0/128	30/128
Adapted from the small-init cell-masking model	0.420	68/128	123/128
the same with the character readout	0.585	0/128	5/128
Adapted from the character-prefix ARM	0.544	0/128	6/128
ARM reading the clue-masked board as a prefix			
Autoregressive, cell prefix: standard init	0.456	0/128	6/128
Autoregressive, character prefix: standard init	0.435	0/128	4/128

Table 6: Endpoint loss at update $5{,}000$ (nats per character, $32{,}768$ evaluation boards) and valid-board counts. Character- and cell-masking losses are negative ELBOs and exclude any auxiliary term; the AR loss is the exact negative log-likelihood. Completion preserves the supplied clues and accepts any valid solution. Diffusion models decode in confidence order and the ARMs sample left to right: the plain ARM with each clue written into the board when its position is reached, and the prefix ARMs, whose losses are measured with an all-mask prefix, with the clues as the prefix. A dash marks a model without the evaluation.

in a fixed order and the diffusion models in order of confidence, and Sudoku accuracy depends strongly on the order in which cells are filled. An adaptive order that unmasks by probability margin raises the accuracy of masked diffusion models from under 7% to about 90% (Kim et al., 2025), and ARMs trained on the order in which a solver fills the cells solve 94% of puzzles but fail to learn Sudoku without that order (Shah et al., 2024).