LARGE LANGUAGE CONTINUOUS DIFFUSION MODELS

Anonymous authors
Paper under double-blind review

ABSTRACT

Despite success of discrete diffusion language models (dLMs) for fast parallel decoding, their non-smooth, high-dimensional space hinders trajectory steering for reasoning and inference acceleration. To overcome this, we present *Sigma*, the first large-scale (3B/8B) continuous dLM built on steerable, low-dimensional ODE/SDE latent trajectories. Trained blockwise via likelihood optimization, *Sigma* jointly denoises Gaussian-corrupted tokens while learning an optimal embedding geometry. To scale training, *Sigma* leverages pre-trained weights from existing autoregressive (AR) models for warm-starting. During inference, we identify classifier-free guidance and score temperature as essential for high-fidelity reasoning and coding. Across comprehensive math reasoning and coding evaluations against state-of-the-art discrete counterparts (masked dLMs and AR baselines), *Sigma* achieves competitive performance with discrete models on standard benchmarks (e.g., GSM8K, Minerva, HumanEval, MBPP) after pre-training and on challenging reasoning tasks (e.g., MATH-500, AIME) after supervised fine-tuning. Beyond performance parity, we uncover key structural properties unique to continuous dLMs: (*i*) embedding-space steering effectively governs the quality-diversity trade-off, yielding strong pass@k performance and (*ii*) continuous trajectories enable graceful degradation for low NFEs and efficient distillation. These establish continuous dLMs as a promising paradigm for efficient language generation.

1 INTRODUCTION

While diffusion language models (dLMs) enable fast parallel decoding compared to autoregressive (AR) baselines (Inception Labs et al., 2025; Song et al., 2025; Fu et al., 2026), dominant discrete dLMs operate in high-dimensional, categorical state spaces (e.g., $V \geq 100$K) that suffer under limited step budgets and impede trajectory control for complex reasoning and coding tasks. Continuous diffusion offers a principled way to overcome these structural limitations through smooth latent geometries that natively support structured editing (Meng et al., 2022), flexible ODE trajectory steering (Lu et al., 2022), and efficient few-step distillation (Salimans & Ho, 2022; Song et al., 2023).

Recently, there has been growing interest in continuous diffusion and flow matching for text generation (Gulrajani & Hashimoto, 2023; Dieleman et al., 2022; Hu et al., 2026; Lee et al., 2026; Guo et al., 2026; Liang et al., 2026; Yang et al., 2026); see App. A. However, existing methods remain limited in scale ($<$ 2B) or focus on *unconditional* generation, lacking evaluations on complex reasoning tasks. Whether continuous dLMs scale to complex conditional reasoning remains open. This is a non-trivial hurdle because continuous models update block embeddings globally, in contrast to the factorized categorical denoising of discrete dLMs that leads to semi-autoregressive token commitment per block.

To overcome these limitations, we present *Sigma* (3B/8B): **s**cal**i**ng up **g**aussian diffusion **m**odels for l**a**nguage. We propose a family of continuous dLMs operating over a 16-dimensional embedding space trained blockwise via likelihood optimization. Within each block, the model learns to jointly denoise Gaussian-corrupted token embeddings and to structure an optimal embedding geometry. To scale training, we warm-start the model from existing pre-trained AR weights. For inference, we explore diffusion trajectory sampling, identifying embedding-space Classifier-Free Guidance (CFG) and Score Temperature (ST) as crucial ingredients for high-fidelity reasoning and coding.

Through unified evaluations against state-of-the-art discrete dLMs (e.g., Nemotron-Labs-Diffusion, LLaDA, Dream), we show that *Sigma* achieves competitive performance across standard math and coding benchmarks (e.g., GSM8K, Minerva, HumanEval, MBPP) and extends well to harder reasoning tasks after supervised fine-tuning (e.g., MATH-500, AIME). Beyond raw accuracy, we

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.

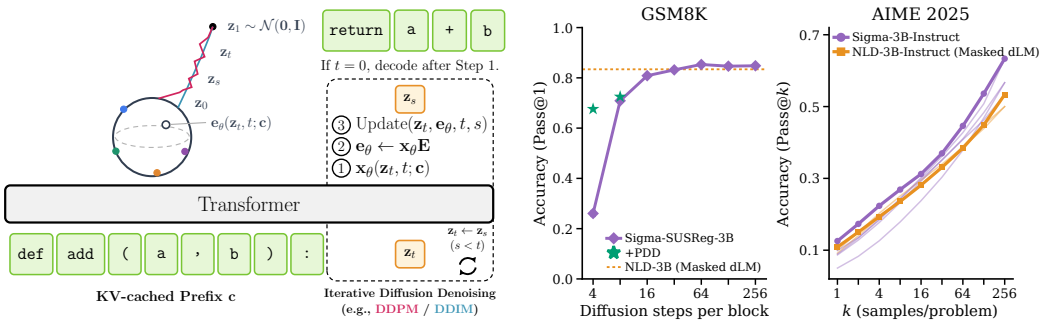


Figure 1: **Left: Inference pipeline.** *Sigma* generates text block-wise via iterative diffusion denoising over 16D continuous latents ($\mathbf{z}_t$) conditioned on a KV-cached prefix ($\mathbf{c}$), followed by categorical readout. Predicted vocabulary probabilities $\mathbf{x}_\theta(\mathbf{z}_t, t; \mathbf{c})$ map to mean predictions $\mathbf{e}_\theta(\mathbf{z}_t, t; \mathbf{c})$ in 16D space via categorical reparameterization. Clean embeddings (dots) lie on the 16D unit sphere, while $\mathbf{e}_\theta$ lies on or within it. **Middle: Smooth trajectory enables low-NFE reasoning.** GSM8K accuracy vs. NFEs using the optimal sampler per step budget (PDD: Parallel Decoding Distillation). **Right: Scaling inference-time compute.** Pass@k vs. samples per problem on AIME 2025 (solid curves: fixed config across k; thin curves: alternative tested configs).

highlight two structural properties unique to continuous language diffusion: (*i*) flexible pass@k exploration–exploitation control via CFG and ST hyperparameters; and (*ii*) graceful performance degradation under low NFE regimes (16/32) and efficient trajectory distillation for very low NFE regimes (4/8). Together, these properties position continuous diffusion as a promising foundation for fast inference in modern agentic AI. Our key contributions are summarized as follows:

- **First Large-Scale Continuous dLM:** We present *Sigma* (3B/8B), efficiently trained by warm-starting from pre-trained AR weights, and optimizing a regularized continuous diffusion ELBO.
- **Competitive Math & Coding:** Across six base and ten instruct benchmarks, *Sigma* achieves competitive performance with state-of-the-art discrete dLMs/ARs on reasoning and coding.
- **Inference Steering (CFG & ST):** By elucidating the inference design space, we show that embedding-space CFG and ST effectively govern the quality-diversity trade-off with minimal latency overhead via single-pass block-causal attention packing.
- **Low-NFE Robustness:** With embedding regularization, *Sigma* sustains strong quality down to 16 NFEs per block, while trajectory distillation noticeably improves performance at 4–8 NFEs.

2 Background

Notation. Let $\mathbf{x}$ be a sequence of length L over a vocabulary $\mathcal{V}$ of size V. We identify $\mathbf{x}$ with a matrix in $\{0,1\}^{L\times V}$ whose rows $\mathbf{x}^1, ..., \mathbf{x}^L$ are one-hot vectors in $\mathbb{R}^V$. Δ^V denotes the probability simplex over $\mathcal{V}$. $\langle \cdot, \cdot \rangle$ denotes the inner product. NELBO stands for negative evidence lower bound.

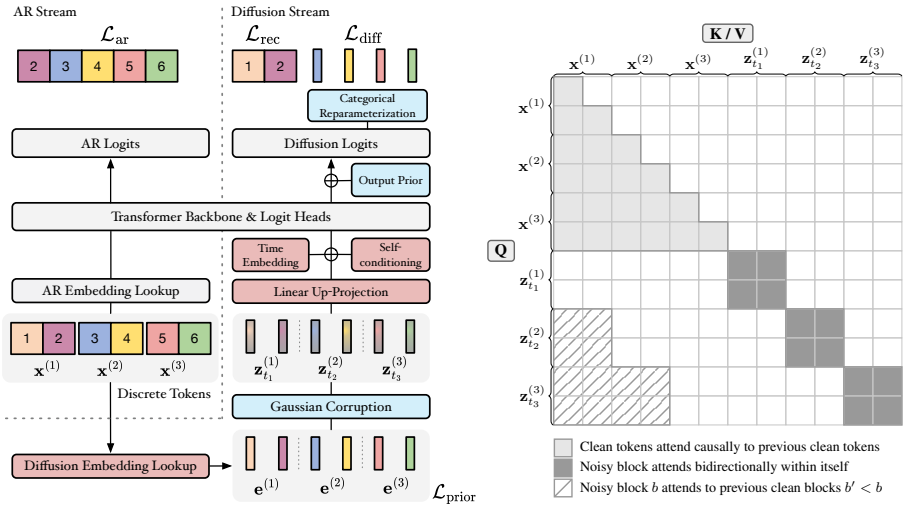
Plaid. Continuous dLMs lift discrete tokens into a low-dimensional Euclidean space and run Gaussian diffusion. There have been multiple proposed design choices, see App. A for an overview. We adopt the embedding and diffusion process from Plaid (Gulrajani & Hashimoto, 2023), a Variational Diffusion Model (VDM, Kingma et al. (2021)) over token embeddings. Let $\mathbf{E} \in \mathbb{R}^{V\times d_e}$ be a learnable embedding matrix whose rows lie on the unit sphere, with $d_e \ll V$. The embedding of $\mathbf{x}$ is given by $\mathbf{e} := \mathbf{x}\mathbf{E} \in \mathbb{R}^{L\times d_e}$ (i.e., standard embedding lookup). The forward process applies variance-preserving Gaussian noise on $\mathbf{e}$:

$$q(\mathbf{z}_t \mid \mathbf{x}) = \mathcal{N}(\alpha_t \mathbf{e},\ \sigma_t^2 \mathbf{I}), \qquad \alpha_t, \sigma_t > 0, \qquad \alpha_t^2 + \sigma_t^2 = 1, \qquad t \in [0,1], \tag{1}$$

with $\mathrm{SNR}(t) = \alpha_t^2/\sigma_t^2$ decreasing monotonically over $t \in [0,1]$. A denoiser $\mathbf{x}_\theta(\mathbf{z}_t, t) \in (\Delta^V)^L \subset \mathbb{R}^{L\times V}$ with output prior returns a categorical distribution over the vocabulary at every position. The reverse process is the exact Gaussian posterior $q(\mathbf{z}_s \mid \mathbf{z}_t, \mathbf{x})$ (Kingma et al., 2021, App. A.2) with $\mathbf{x}$ replaced by $\mathbf{x}_\theta(\mathbf{z}_t, t)$. Its NELBO splits into 3 terms: prior KL $\mathcal{L}_{\text{prior}}$, reconstruction loss $\mathcal{L}_{\text{rec}}$, and diffusion loss $\mathcal{L}_{\text{diff}}$:

$$\begin{aligned} -\log p_\theta(\mathbf{x}) \le \mathcal{L}_{\text{NELBO}}(\mathbf{x}) := \mathrm{KL}(q(\mathbf{z}_1 \mid \mathbf{x}) \,\|\, p(\mathbf{z}_1)) + \mathbb{E}_{\mathbf{z}_0\sim q(\mathbf{z}_0|\mathbf{x})}\left[\textstyle\sum_{l=1}^{L} - \log\langle \mathbf{x}_\theta^l(\mathbf{z}_0, 0), \mathbf{x}^l\rangle\right] \\ - \tfrac{1}{2}\ \mathbb{E}_{t\sim\mathrm{Unif}[0,1]}\, \mathbb{E}_{\mathbf{z}_t\sim q(\mathbf{z}_t|\mathbf{x})}[\mathrm{SNR}'(t) \|\mathbf{e}_\theta(\mathbf{z}_t, t) - \mathbf{e}\|^2], \end{aligned} \tag{2}$$

where $p(\mathbf{z}_1) = \mathcal{N}(\mathbf{0}, \mathbf{I})$ and the *categorical reparameterization* $\mathbf{e}_\theta(\mathbf{z}_t, t) := \mathbf{x}_\theta(\mathbf{z}_t, t)\mathbf{E} \in \mathbb{R}^{L\times d_e}$ predicts the posterior mean embedding $\mathbb{E}[\mathbf{e} \mid \mathbf{z}_t, t]$ corresponding to $\mathbf{x}_\theta(\mathbf{z}_t, t)$.



(a) Training procedure of *Sigma*. (b) Attention mask for training.

Figure 2: **Training procedure of *Sigma* (a) and its transformer attention mask (b)** for an example sequence $\mathbf{x}$ of length $L = 6$ with block size $L' = 2$. At pre-training, we feed the clean discrete tokens $\mathbf{x}$ to the AR stream (a, left part) and the noisy continuous embeddings $\mathbf{z}$ to the diffusion stream (a, right part). For the diffusion stream, each block is randomly chosen to either compute the reconstruction loss (1st block, $t = 0$) or the diffusion loss (2nd & 3rd blocks, $t \sim \text{Unif}[0, 1]$); see (5). Reconstruction loss is the cross entropy between the logits and the ground truth tokens, while for diffusion loss we use categorical reparameterization to obtain the mean embedding prediction and compute the mean squared error against the ground truth diffusion embeddings.

Learned noise schedule. The schedule admits $\gamma(t) = \gamma_0 + (\gamma_1 - \gamma_0)\tilde{\gamma}(t)$ with learnable endpoints $\gamma_0 < \gamma_1$ and a monotone interior shape $\tilde{\gamma} \in [0, 1]$, giving $\alpha_t^2 = \sigma(-\gamma(t))$, $\sigma_t^2 = \sigma(\gamma(t))$ and $\text{SNR}(t) = \mathrm{e}^{-\gamma(t)}$.[1] The diffusion term depends on the endpoints alone and is invariant to $\tilde{\gamma}(t)$, which instead controls its Monte-Carlo (MC) variance (Kingma et al., 2021), so they are learned for different reasons. Learning the interior-shape distributes denoising difficulty over t (Yang et al., 2026).

Learned embeddings. The NELBO (2) depends on $\mathbf{E}$ via (1), where embeddings become likelihood-trained parameters, which separates Plaid from continuous dLMs built on fixed (e.g., one-hot) encodings. Not learning embeddings significantly worsens Plaid's performance (Yang et al., 2026).

Self-conditioning (SC). Proposed in Chen et al. (2023), SC lets the denoiser see its own previous estimate. Let $\mathbf{x}_\theta(\mathbf{z}_t, t; \mathbf{e}_{\text{sc}})$ denote a denoiser taking an additional predicted clean embedding $\mathbf{e}_{\text{sc}} \in \mathbb{R}^{L \times d_e}$, and $\mathbf{0}$ for the empty condition (omitted by default in our notation). During training, 25% of train samples are self-conditioned: a first pass without gradient tracking produces $\mathbf{e}_{\text{sc}} := \mathbf{x}_\theta(\mathbf{z}_t, t; \mathbf{0})\mathbf{E}$, and only the second pass, $\mathbf{x}_\theta(\mathbf{z}_t, t; \text{sg}(\mathbf{e}_{\text{sc}}))\mathbf{E}$, receives gradients. The remaining directly train on $\mathbf{0}$, so the denoiser learns both branches. During validation, all samples enable SC. Since $\mathbf{e}_{\text{sc}}$ is itself a function of $(\mathbf{z}_t, t)$, the NELBO remains valid. Notice that during training, 25% SC increases the non-embedding FLOPs by only about 8%, which is minimal. Furthermore, during inference one directly feeds the previous-step's prediction, which incurs no overhead.

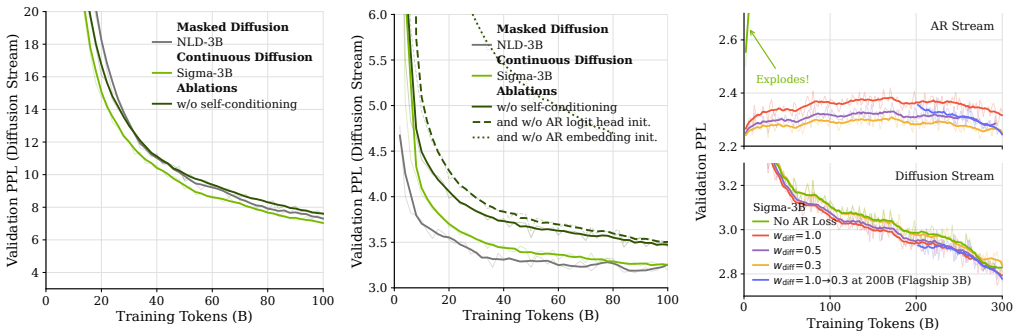
Scaling. RePlaid (Yang et al., 2026) realigns Plaid with the architectures of modern discrete dLMs/ARs and finds that, once aligned, continuous diffusion scales competitively with masked diffusion.

3 Large Language Continuous Diffusion Models

3.1 Blockwise Probabilistic Model

Inspired by Nemotron-Labs-Diffusion (NLD, Fu et al. (2026)), a state-of-the-art open-sourced masked dLM (see App. C for background), we adopt a block-wise formulation for our continuous diffusion language model: the sequence $\mathbf{x}$ is partitioned into B contiguous blocks $\{\mathbf{x}^{(b)}\}_{b=1}^{B}$ of size $L' = L/B$, where the model denoises one block at a time given its clean prefix. As in Arriola et al. (2025), we

[1] $\sigma(t) = 1/(1 + \mathrm{e}^{-t})$ is the sigmoid function.



(a) From scratch + no AR regularization Block size: 64

(b) From AR init. + no AR regularization Block size: 64

(c) From AR init. + AR regularization Block size: 32

Figure 3: **Pre-training *Sigma*-3B** from scratch (a), from AR initialization (b), and from AR initialization with AR regularization (c). PPL for AR and diffusion streams is computed via $\mathcal{L}_{\text{ar}}(\mathbf{x})$ (6) and $\mathcal{L}_{\text{BD}}(\mathbf{x})$ (3), respectively. Early investigations (a, b) used block size 64; our flagship 3B model (c) used block size 32 to match the inference block size of NLD. Proper AR initialization/regularization leads to the best *Sigma* model.

factorize the likelihood autoregressively *across* blocks and then bound each factor:

$$-\log p_\theta(\mathbf{x}) = \sum_{b=1}^{B} -\log p_\theta(\mathbf{x}^{(b)} \mid \mathbf{x}^{(<b)}) \le \sum_{b=1}^{B} \mathcal{L}_{\text{NELBO}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)}) =: \mathcal{L}_{\text{BD}}(\mathbf{x}), \tag{3}$$

where we obtain $\mathcal{L}_{\text{NELBO}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)})$ by conditioning the denoiser on $\mathbf{x}^{(<b)}$ as follows:

$$\begin{aligned}\mathcal{L}_{\text{NELBO}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)}) &= \mathrm{KL}(q(\mathbf{z}_1^{(b)} \mid \mathbf{x}^{(b)}) \,\|\, p(\mathbf{z}_1^{(b)})) \\ &\quad + \mathbb{E}_{q(\mathbf{z}_0^{(b)}|\mathbf{x}^{(b)})}\left[\sum_{l=1}^{L'} -\log\langle \mathbf{x}_\theta^l(\mathbf{z}_0^{(b)}, 0; \mathbf{x}^{(<b)}), \mathbf{x}^{(b,l)}\rangle\right] \\ &\quad - \tfrac{1}{2}\,\mathbb{E}_{t\sim\text{Unif}[0,1]}\,\mathbb{E}_{q(\mathbf{z}_t^{(b)}|\mathbf{x}^{(b)})}[\mathrm{SNR}'(t)\|\mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{x}^{(<b)}) - \mathbf{e}^{(b)}\|^2].\end{aligned} \tag{4}$$

Training. The reconstruction loss at $t = 0$ and diffusion loss at $t \in (0, 1)$ do not share a single forward pass. RePlaid evaluates one of them per $\mathbf{x}^{(b)}$ via Horvitz–Thompson estimator (Horvitz & Thompson, 1952) with $R \sim \text{Bern}(p_{\text{rec}})$, i.e., $\mathbf{x}^{(b)}$ is assigned to $\mathcal{L}_{\text{rec}}$ w.p. $p_{\text{rec}} \in (0, 1)$:

$$\mathcal{L}_{\text{NELBO}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)}) = \mathbb{E}_R\left[\mathcal{L}_{\text{prior}}(\mathbf{x}^{(b)}) + \tfrac{R}{p_{\text{rec}}}\mathcal{L}_{\text{rec}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)}) + \tfrac{1-R}{1-p_{\text{rec}}}\mathcal{L}_{\text{diff}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)})\right]. \tag{5}$$

Notice that this is an unbiased estimator if $\mathcal{L}_{\text{diff}}$ and $\mathcal{L}_{\text{rec}}$ are unbiased. We set $p_{\text{rec}} = 0.1$, empirically found to minimize the training objective variance of RePlaid.

To construct the unbiased estimator for $\mathcal{L}_{\text{BD}}(\mathbf{x})$ in (3) for training, we sample $R^{(b)}$ i.i.d. for each block. If $R^{(b)} = 0$, we sample $t^{(b)}$ and $\mathbf{z}_{t^{(b)}}^{(b)}$; otherwise, we set $t = 0$ and sample $\mathbf{z}_0^{(b)}$. The noise schedule endpoints are learned to optimize $\mathcal{L}_{\text{NELBO}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)})$ while the schedule interior shape is learned to optimize its MC variance, with both shared across blocks. As mentioned before, we apply SC only to 25% of samples (full sequence with all blocks). See Alg. 1 for the training algorithm.

3.2 Pre-Training

NLD utilizes AR initialization and regularization to enhance downstream performance (Fu et al., 2026). However, adapting these techniques to continuous dLMs like *Sigma* is non-trivial, as the activation dynamics of continuous and discrete models differ fundamentally; it is thus unclear whether discrete AR initialization remains as effective for continuous representation space. To establish a rigorous baseline, we first align *Sigma*'s data, architecture, and optimization with discrete dLMs for a controlled comparison at scale (Sec. 3.2.1). In Sec. 3.2.2, we evaluate the impact of AR initialization when pretraining *Sigma*-3B with (3), followed by an investigation of AR regularization for training our flagship 3B/8B models.

3.2.1 Unified Pretraining of Continuous and Discrete dLMs

Data. As with NLD, we adopt the open-sourced pretraining data from Basant et al. (2025). We use the Mistral-NeMo tokenizer ($V = 131{,}072$) (Mistral AI, 2024), batch size 512, and $L = 4096$.

Architecture. *Sigma* adopts the transformer backbone of NLD and its AR baseline, ensuring the three families are parameter- and FLOP-matched. As with NLD, *Sigma* uses a block causal attention mask (Fig. 2b). *Sigma* uniquely adds a unit-norm 16-dimensional diffusion embedding table with a linear up-projection, a linear down-projection mapping logits to soft embeddings, a self-conditioning linear projection, Fourier time conditioning, and a learnable noise schedule (see Sec. 2). Together, these account for less than 1% of the total parameters. We provide thorough details in App. E.

Training. We train our models for 150K steps, following the optimization procedure detailed in App. E. Given a batch size of 512, 150K steps correspond to over 300B tokens. Each training consumed ~9,600 and ~19,200 NVIDIA GB200 GPU hours for the 3B and 8B models, respectively.

3.2.2 AR Initialization and Regularization

AR Initialization. To evaluate the impact of initialization, we compare pretraining *Sigma* (*i*) from scratch and (*ii*) from an AR baseline pretrained on 1T tokens. Due to the dimension mismatch between high-dimensional AR embeddings and *Sigma*'s 16-dimensional space, we randomly initialize the diffusion embedding layer. As shown in Fig. 3(b), *Sigma*-3B (L'=64) with self-conditioning achieves competitive perplexity relative to NLD-3B (L'=64) under both regime (i) and (ii). However, AR initialization significantly accelerates convergence and yields lower final training loss.

AR Regularization. Continuous diffusion training inherently suffers from high gradient variance and slow convergence. To mitigate variance and accelerate optimization, we follow NLD by adding AR regularization, optimizing the joint objective $\mathcal{L}_{\text{ar}} + w_{\text{diff}}\mathcal{L}_{\text{BD}}$ (Fig. 3c; see App. E for the w_{diff} weighting schedule), where

$$\mathcal{L}_{\text{ar}} = \sum_{l=2}^{L} -\log\langle \mathbf{x}_\theta^l(\mathbf{x}^{<l}), \mathbf{x}^l \rangle \tag{6}$$

is computed over the AR stream using causal attention (Fig. 2). This architecture employs two distinct readout heads, with the AR regularization applied exclusively to the AR head. While this dual-head formulation improves auxiliary AR decoding capability, our primary objective remains maximizing diffusion decoding performance.

Finally, input sequences utilize AR embeddings for the AR stream and 16D diffusion embeddings for the diffusion stream (Fig. 2), facilitating AR initialization and efficient KV-caching. As shown in Fig. 3(c), AR regularization significantly improves training stability; with a properly tuned w_{diff}, it even outperforms pure perplexity optimization for the diffusion stream. This confirms that AR regularization stabilizes training while yielding a higher-quality model. Consequently, we adopt AR initialization, AR regularization, and a block size of L'=32 for all subsequent *Sigma* experiments.

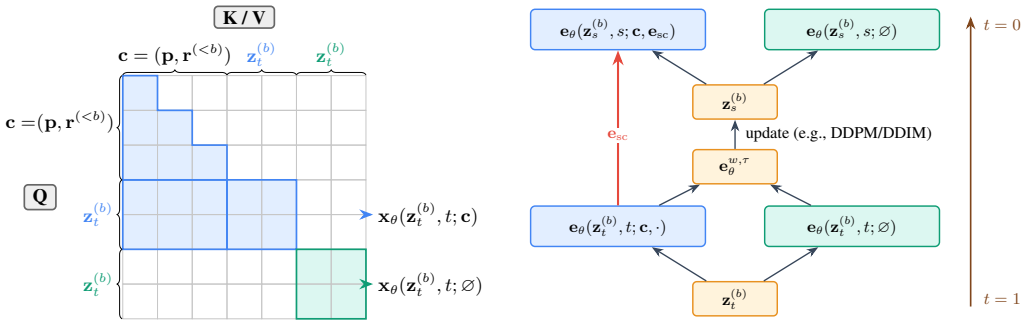
4 Language Generation via Continuous Diffusion Sampling

Drawing on the success of continuous diffusion samplers using SDE/ODE solvers (Song et al., 2021b), we systematically explore decoding samplers for continuous dLMs, evaluating performance across standard mathematical reasoning and coding benchmarks.

4.1 Block diffusion sampling

To generate a response $\mathbf{r}$ given a prompt $\mathbf{p}$, we denoise one block at a time. Let $\mathbf{c} := (\mathbf{p}, \mathbf{r}^{(<b)})$ be the clean context available when generating block b, stored in an attention key-value (KV) cache using the AR embeddings. An offset block-causal mask determines who can see whom (Fig. 4a). Positions in $\mathbf{c}$ use causal attention, so their KV can be cached and reused across diffusion steps, regardless of whether the prompt length is a multiple of L'. Positions in the noisy block $\mathbf{z}_t^{(b)}$ attend bidirectionally within the block and causally to the entire prefix $\mathbf{c}$. After a block is decoded, the AR embeddings of its committed tokens are appended to the causal KV cache for the next block during the next block's first diffusion step, without incurring an extra NFE.

Sampling block b starts from $\mathbf{z}_1^{(b)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ and steps from t to $s < t$ along a discretization of $[1, 0]$. At each step the denoiser yields a predicted clean embedding $\hat{\mathbf{e}}_\theta := \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c}) = \mathbf{x}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c})\mathbf{E}$ and the implied noise prediction $\hat{\boldsymbol{\epsilon}}_\theta := \frac{1}{\sigma_t}(\mathbf{z}_t^{(b)} - \alpha_t \hat{\mathbf{e}}_\theta)$. Below we introduce two canonical samplers for the reverse process. For both samplers, after the final step, tokens are read out as argmax of the conditional logit output $\mathbf{x}_\theta(\mathbf{z}_0^{(b)}, 0; \mathbf{c})\mathbf{E}$ and appended to $\mathbf{c}$. See Alg. 2 for the sampling algorithm.



(a) Packed attention mask while generating block b ($L' = 2$). Prefix $\mathbf{c}$ is causal (KV-cached). Conditional $\mathbf{z}_t^{(b)}$ attends to all of $\mathbf{c}$ and bidirectionally within the block; unconditional copy attends only to itself.

(b) One reverse step $t \to s$. CFG and ST mix the two denoisers into $\mathbf{e}_\theta^{w,\tau}$; an update yields $\mathbf{z}_s^{(b)}$. SC (red) feeds the previous conditional mean (not sharpened by CFG or ST) to the next conditional branch.

Figure 4: Inference for block-wise continuous diffusion: packed CFG attention (a) and one reverse step (b).

DDPM. We use the ancestral kernel of VDM (Kingma et al., 2021, App. A.4). It draws $\mathbf{z}_s^{(b)}$ from the closed-form Gaussian posterior $q(\mathbf{z}_s^{(b)} \mid \mathbf{z}_t^{(b)}, \mathbf{x}^{(b)})$ with the clean embedding replaced by $\hat{\mathbf{e}}_\theta$,

$$\mathbf{z}_s^{(b)} = (1-c)\tfrac{\alpha_s}{\alpha_t}\mathbf{z}_t^{(b)} + c\,\alpha_s\hat{\mathbf{e}}_\theta + \sqrt{c(1-\alpha_s^2)}\,\boldsymbol{\epsilon}, \quad \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}),\ c = 1 - \mathrm{e}^{\gamma(s)-\gamma(t)}. \tag{DDPM}$$

DDIM. One can also leverage the predictions $\hat{\mathbf{e}}_\theta$ and $\hat{\boldsymbol{\epsilon}}_\theta$ at $\mathbf{z}_t^{(b)}$ to deterministically update the state (Song et al., 2021a):

$$\mathbf{z}_s^{(b)} = \alpha_s\hat{\mathbf{e}}_\theta + \sigma_s\hat{\boldsymbol{\epsilon}}_\theta. \tag{DDIM}$$

4.2 Trajectory Steering

To sharpen model predictions, we incorporate two reverse-process steering methods to our block-wise continuous diffusion model without impacting its training and inference efficiency.

Classifier-free guidance (CFG). CFG (Ho & Salimans, 2022) is the standard tool for class-conditional or text-to-image diffusion models. In CFG, one queries the model twice to obtain a conditional and an unconditional prediction and then linearly extrapolates them. For our block diffusion model, the clean context $\mathbf{c}$ serves as the natural conditioning information, and CFG extrapolates away from the unconditional prediction for the current block b:

$$\mathbf{e}_\theta^w(\mathbf{z}_t^{(b)}, t; \mathbf{c}) := \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \varnothing) + w\left[\mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c}) - \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \varnothing)\right], \tag{CFG}$$

where $\varnothing$ drops the entire clean context $\mathbf{c}$ (so $\mathbf{z}_t^{(b)}$ attends only within its own block) and $w \geq 1$ is the guidance strength ($w = 1$ falls back to the plain conditional prediction). We bring two technical considerations. First, at batch size 1 transformer inference is memory-bound, so we pack conditional and unconditional predictions into a single forward pass at almost no wall-clock cost (Liu et al., 2026b), using the packed mask of Fig. 4a. We measure this overhead to be less than 5% per forward pass at block size 32 and 64 (App. G.10). Second, unlike in image diffusion models and existing DLMs that require train-time context dropout (Potaptchik et al., 2026; Hu et al., 2026), **our CFG is a pure inference-time trick**: due to sequence packing in pre-training, the first block typically starts in the middle of a document and naturally sees no context. We discuss existing approaches in App. A.

Score temperature (ST). ST (Dieleman et al., 2022) is a heuristic that sharpens the conditional score itself, rescaling the score $\nabla \log p_t(\mathbf{z}_t^{(b)}|\mathbf{c})$ (or equivalently, the noise prediction) by $\frac{1}{\tau}$ for $\tau \in (0, 1]$. One can show that it corresponds to the following operation on the mean prediction.

$$\mathbf{e}_\theta^\tau(\mathbf{z}_t^{(b)}, t; \mathbf{c}) := \tfrac{\mathbf{z}_t^{(b)}}{\alpha_t} + \tfrac{1}{\tau}\left[\mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c}) - \tfrac{\mathbf{z}_t^{(b)}}{\alpha_t}\right]. \tag{ST}$$

Intuitively, since $\mathbb{E}[\mathbf{z}_t^{(b)}|\mathbf{x}^{(b)}] = \alpha_t\mathbf{e}^{(b)}$, $\frac{1}{\alpha_t}\mathbf{z}_t^{(b)}$ serves as the naive bad prediction to extrapolate from. ST does not incur further model queries, and $\tau = 1$ leaves the sampler unchanged. We also recognize ST as a special case of Temporal Score Rescaling proposed in (Xu et al., 2026).

Table 1: **Performance of *Sigma*-3B (base model; block size: 32)** on math reasoning and coding benchmarks. Accuracy (%) is pass@1. The *Sigma* diffusion rows are reported as mean (over 10 seeds) ± two standard errors. **Bold**: the best value across all methods; underline: the best value within each mode.

Method	Math		Coding				Avg.
	GSM8K	Minerva Math	HumanEval	HumanEval+	MBPP	MBPP+	
AR initialization	83.70	**66.28**	51.22	48.78	60.60	77.78	64.73
AR mode							
Nemotron-Labs-Diffusion-3B	83.78	65.06	53.05	47.56	**60.80**	**78.84**	64.85
***Sigma*-3B (Ours)**	83.40	65.10	57.32	51.83	59.00	76.19	**65.47**
Diffusion mode							
LLaDA-8B (Nie et al., 2025b)	70.96	27.30	32.32	27.44	40.80	51.85	41.78
Dream-7B (Ye et al., 2025)	77.18	39.60	54.88	49.39	56.80	74.60	58.74
Nemotron-Labs-Diffusion-3B	83.40	63.96	56.10	**53.05**	57.60	74.34	64.74
***Sigma*-3B (Ours)**							
DDPM 256 steps / block (w=2, τ=0.1)	**84.50** (± 0.31)	64.60 (± 0.28)	**59.45** (± 1.30)	**53.84** (± 1.29)	52.32 (± 0.73)	71.32 (± 0.91)	64.34
DDIM 32 steps / block (w=3, τ=0.3)	81.91 (± 0.32)	62.76 (± 0.18)	56.59 (± 1.31)	50.67 (± 1.00)	47.26 (± 0.46)	63.49 (± 0.77)	60.45
DDIM 16 steps / block (w=3, τ=0.5)	78.08 (± 0.34)	60.16 (± 0.21)	47.50 (± 1.14)	43.84 (± 0.94)	41.84 (± 1.00)	59.58 (± 0.98)	55.17

Finally, we remark that due to the commutativity of (CFG) and (ST), we can combine them into a single operation $\mathbf{e}_\theta^{w,\tau}(\mathbf{z}_t^{(b)}, t; \mathbf{c})$, which is essentially using the score of the composed distribution $(p_t(\cdot \mid \mathbf{c})^w p_t(\cdot \mid \varnothing)^{1-w})^{1/\tau}$ (Prop. 1). We observed that using either knob is weaker than using both of them, see Fig. 5.

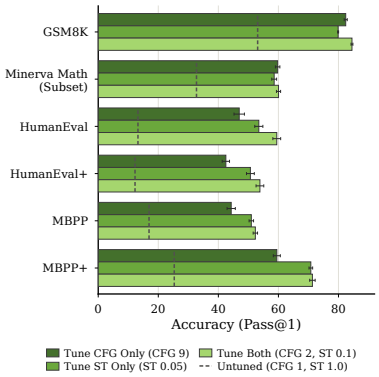


Figure 5: **CFG and ST tuning** on *Sigma*-3B for DDPM sampling. Combining both techniques yields significant gains over applying either in isolation, and markedly outperforms generation without CFG or ST (see Fig. 10).

Self-conditioning (SC). At inference, SC is applied only to the conditional prediction: when querying the model at time s, the previous step's conditional prediction $\mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c})$ (before applying CFG and ST) serves as the SC input for the conditional prediction, while the unconditional prediction receives $\mathbf{0}$ SC input (Fig. 4b).

Sharpening in logit space. While both knobs above act on the mean embedding prediction obtained after the categorical reparameterization, we can also sharpen in logit space by applying the same two ideas before the softmax. In App. G.3, we show that the logit-space CFG and temperature do provide sharpening, but the embedding-space knobs provide stronger performance, hence we keep (CFG) and (ST) as the default option.

4.3 Mathematical Reasoning and Coding Performance

We evaluate on two math and four coding benchmarks: GSM8K, Minerva Math, HumanEval, HumanEval+, MBPP, and MBPP+. Results are reported in Tab. 1. In diffusion mode, *Sigma*-3B uses DDPM sampling with 256 steps per block with EDM $\rho = 7$ schedule, $w = 2, \tau = 0.1$ and SC. See App. G.1 for details on the harness, scoring pipeline, error bars, and the samplers for baselines.

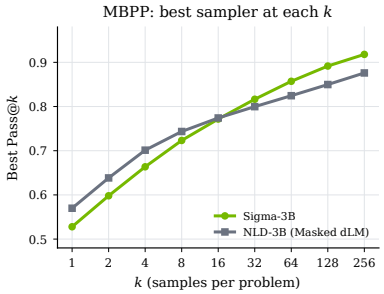


Figure 6: **Pass@k accuracy on MBPP.** As the sampling budget grows, *Sigma* prefers higher ST and overtakes NLD (masked dLM) after $k = 32$.

Results. In AR mode, *Sigma*-3B attains the best six-task average across all methods; in diffusion mode, our DDPM-256 run is **comparable to Nemotron-Labs-Diffusion-3B on average and sets the best base-model numbers on GSM8K, HumanEval, and HumanEval+**; it also outperforms LLaDA-8B and Dream-7B by 22.6 and 5.6 points on average respectively, despite being 2× smaller in parameters. Empirically, we observe *less* left-to-right bias within a block for *Sigma* compared with the confidence-based masked DLM sampling (Fig. 20). To confirm the benefit of AR regularization (Sec. 3.2.2), we also trained *Sigma*-3B with just $\mathcal{L}_{\text{BD}}$ (3) without $\mathcal{L}_{\text{ar}}$ (6); results are uniformly worse across all tasks (Fig. 11). We refer readers to Tab. 8 for the same evaluation on 8B base models.

Pass@k analysis. While pass@1 accuracy in Tab. 1 captures single-draw quality, test-time search and RL draw multiple rollouts for the same prompt, where diverse rollouts are needed for richer

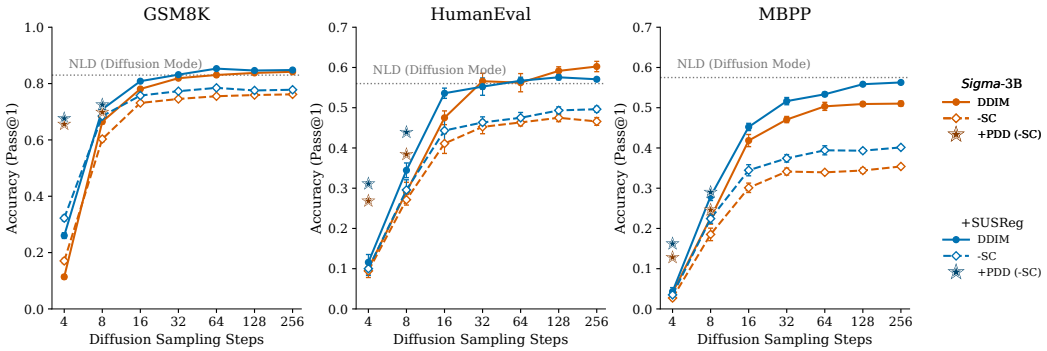


Figure 7: **Performance vs. diffusions steps of *Sigma*-3B and *Sigma*-SUSReg-3B (block size: 32).** All base models use DDIM sampling with uniform-γ time discretization, with best per-NFE (CFG, ST) shared across the three tasks. PDD models use their own DDIM-based sampling and a common (CFG, ST) setting inherited at distillation, with self-conditioning disabled in both the student and teacher. Results on all six base tasks: Tab. 7.

learning signals and increase the chance of discovering a correct solution. Pass@k does not directly measure diversity, but tests whether additional samples expand correct-solution coverage. We sweep CFG and ST for *Sigma*-3B and confidence threshold and sampling temperature for NLD, reporting each model's best tested config at each budget k. On MBPP, *Sigma* overtakes NLD at $k = 32$ and reaches 91.8% versus 87.6% at $k = 256$ (Fig. 6). See App. G.8 for the full study.

5 Few-Step Generation

5.1 ODE Sampler and Time Discretization

In previous results, we run 256 DDPM steps per block to establish the strongest performance. In low-NFE regime, it is generally known that deterministic samplers paired with well-tuned time discretization lead to better performance. To investigate the tradeoffs for *Sigma*-3B, we choose two representative tasks (GSM8K and HumanEval), and vary the sampler (DDPM or DDIM) and the time-discretization across various diffusion sampling budgets $T \in \{4, 8, 16, 32, 64, 128, 256\}$. For each sampler-discretization-budget combination, we sweep CFG and ST; we then report the CFG-ST configuration that achieves the best average accuracy over the two tasks (Figs. 13 and 14).

We found that **DDIM substantially outperforms DDPM at low NFEs** while matching DDPM at high NFEs. The uniform-γ schedule (App. G.6) performs the best for a wide range of sampling budgets. Surprisingly, we found higher-order ODE solvers originally proposed for image diffusion models (e.g., Zhang & Chen (2023); Lu et al. (2022)) perform worse than DDIM, see App. G.7 for the full study. Using the same optimal hyperparameters from the two tasks, we evaluate the 16/32-step performance across all six tasks in Tabs. 1 and 8; Tab. 7 shows performance at other budgets.

5.2 Embedding Regularization

We observe that during training, embeddings collapse to 4 effective dimensions in terms of principal components—a phenomenon not typically seen in standard autoregressive language model embeddings. While the current model achieves strong high-NFE performance, we hypothesize that regularizing the embeddings to be more uniformly distributed on the unit sphere will improve low-NFE performance. Our intuition is that representation collapse exacerbates token ambiguity, particularly when coarse numerical precision and low-NFE sampling introduce prediction errors.

To mitigate this, we apply Sketched Uniform Spherical Regularization (SUSReg, Nicollier et al. (2026)) to the embedding matrix $\mathbf{E}$ during pre-training, penalizing the deviation between its empirical distribution and a uniform distribution on the unit sphere (see App. E.1). The resulting *Sigma*-SUSReg-3B **improves few-step and distillation results** compared to *Sigma*-3B (Fig. 7 and Tab. 7).

5.3 Parallel Decoding Distillation for Few-step Generation

Although DDIM has better performance than DDPM at reduced denoising steps, the pretrained model still loses quality at very low NFEs with a clear drop-off below 16 NFEs (Fig. 7). We therefore distill the many-step trajectory of *Sigma*-3B into a few-step student. Our method adapts Parallel Decoding

Distillation (PDD, Shaul et al. (2026)) to blockwise continuous language diffusion, see App. I for full details. Due to the parallel nature of the student PDD heads, they cannot utilize recurrent computation, so we distill from the teacher without self-conditioning (SC-off). During distillation, we amortize CFG into our student by training the single conditional student call to match the two-call guided teacher directly, which reduces FLOPs during sampling. As shown in Fig. 7 and Tab. 7, **PDD is most helpful at 4 diffusion NFEs** and **pre-training with SUSReg improves PDD performance**.

6 Supervised Fine-Tuning

We apply supervised fine-tuning (SFT) to the pre-trained base *Sigma* with the same training objective used during pre-training, restricting to data assistant responses. The complete data processing, optimization recipe, and continuous-diffusion-specific controls are given in App. H. As shown in Tabs. 2 and 3, the average performance of *Sigma*-Instruct is **on par with NLD at both 3B and 8B scale**, and **surpasses other open-source masked diffusion models like Dream and LLaDA at the 8B scale**. For each sampler-step combination, we re-use the best CFG-ST configuration found in Sec. 4 and Sec. 5, indicating sampling hyperparameters are reasonably robust to SFT and scale.

Table 2: **Performance of *Sigma*-3B-Instruct (block size: 32)**. Accuracy (%) is pass@1. The *Sigma* diffusion rows are reported as mean (over 10 seeds) ± two standard errors. The four reference models at the top are AR models. Baseline numbers are from Fu et al. (2026). Within *diffusion mode*, underline marks best value per task.

	Math				Coding			Science & Knowledge		IF	
Method	GSM8K	Math500	AIME24	AIME25	HumanEval	MBPP	LCB-CPP	MMLU	GPQA	IFEval	Avg.
Llama-3.2-3B-Instruct	62.09	32.60	0.00	0.00	65.85	66.93	9.25	62.45	27.78	69.69	39.66
Phi-4-mini-Instruct	76.57	70.80	10.00	6.67	77.44	75.13	9.91	53.37	40.91	71.16	49.20
Ministral3-3B-Instruct-2512	87.41	71.90	12.50	12.71	64.79	68.39	12.50	63.21	28.79	57.55	47.98
Qwen3-4B	92.19	68.40	10.63	10.63	75.91	72.49	15.58	77.05	37.18	72.20	53.23
AR mode											
Nemotron-Labs-Diffusion-3B-Instruct	87.87	77.60	23.33	16.67	76.22	71.16	21.37	71.70	39.39	69.39	55.47
***Sigma*-3B-Instruct (Ours)**	87.11	73.00	10.00	6.67	72.56	72.75	13.66	71.24	38.38	66.51	51.19
Diffusion mode											
Nemotron-Labs-Diffusion-3B-Instruct	88.40	74.80	16.67	10.00	74.39	73.54	16.30	72.06	33.84	68.93	52.89
***Sigma*-3B-Instruct (Ours)**											
DDPM (256 steps, w=2, τ=0.1)	89.18 ± 0.24	70.44 ± 0.79	17.00 ± 3.05	11.00 ± 2.12	77.62 ± 1.31	73.10 ± 0.75	11.17 ± 0.28	71.28 ± 0.08	34.70 ± 1.14	66.04 ± 0.60	52.15
DDIM (32 steps, w=3, τ=0.3)	87.86 ± 0.29	68.30 ± 0.67	9.67 ± 2.20	14.00 ± 2.46	72.74 ± 1.22	68.84 ± 0.78	10.11 ± 0.31	69.15 ± 0.11	31.82 ± 1.82	66.09 ± 0.22	49.86

Table 3: **Performance of *Sigma*-8B-Instruct (block size: 32)**. Accuracy (%) is pass@1. The *Sigma* diffusion rows are reported as mean (over 10 seeds) ± two standard errors. The three reference models at the top are AR models. NLD-8B-Instruct's diffusion mode results are reproduced under aligned protocol, while other baseline numbers are from Fu et al. (2026, Tab. 5). Within *diffusion mode*, underline marks best value per task.

	Math				Coding			Science & Knowledge		IF	
Method	GSM8K	Math500	AIME24	AIME25	HumanEval	MBPP	LCB-CPP	MMLU	GPQA	IFEval	Avg.
Qwen2.5-7B-Instruct	91.89	75.10	13.75	6.88	77.44	81.55	12.33	74.86	37.12	74.58	54.55
Qwen3-8B	92.42	84.80	30.21	22.08	81.71	81.88	21.09	76.66	49.24	87.38	62.75
Ministral3-8B-Instruct-2512	92.42	83.60	27.71	24.58	71.04	78.97	20.76	73.90	42.87	64.31	58.02
AR mode											
Nemotron-Labs-Diffusion-8B-Instruct	94.01	88.00	33.33	33.33	80.49	85.19	28.85	79.85	44.44	68.65	63.61
***Sigma*-8B-Instruct (Ours)**	91.96	78.60	16.67	13.33	76.22	83.07	21.81	77.89	43.94	80.46	58.39
Diffusion mode											
LLaDA-8B Instruct (Nie et al., 2025b)	79.91	39.20	0.00	0.00	49.40	41.00	4.19	65.50	33.30	59.90	37.24
Dream-7B Instruct (Ye et al., 2025)	81.00	43.00	0.00	3.33	55.50	58.80	1.25	67.00	33.00	62.50	40.54
SDAR-8B Chat (Cheng et al., 2026)	91.30	78.60	16.67	10.00	78.70	72.00	13.44	78.60	40.20	61.40	54.09
Nemotron-Labs-Diffusion-8B-Instruct	91.58	84.60	30.00	23.33	83.54	82.01	21.15	76.65	33.33	73.02	59.92
***Sigma*-8B-Instruct (Ours)**											
DDPM (256 steps, w=2, τ=0.1)	92.49 ± 0.34	78.88 ± 0.50	23.67 ± 3.05	16.33 ± 3.05	79.76 ± 1.06	81.98 ± 0.87	18.30 ± 0.41	77.46 ± 0.12	44.70 ± 1.74	79.14 ± 0.63	59.27
DDIM (32 steps, w=3, τ=0.3)	92.14 ± 0.19	77.38 ± 0.48	25.00 ± 3.92	13.67 ± 1.48	74.76 ± 1.18	76.56 ± 0.84	15.66 ± 0.47	76.38 ± 0.09	43.54 ± 0.57	74.99 ± 0.59	57.01

7 Conclusions

We introduced *Sigma*, a blockwise continuous dLM via efficient continual pretraining from an AR checkpoint, together with a reverse-diffusion sampling recipe that combines DDPM/DDIM samplers with CFG, ST, and SC. *Sigma* (3B/8B) attains the best six-task average in AR mode and, in diffusion mode, performs comparably with the same-size masked dLM (Nemotron-Labs-Diffusion) trained under an aligned pipeline. The same conclusions generally transfers to post-SFT. Moreover, we adapt a sphere-normalized embedding regularization technique and few-step distillation to *Sigma*, which together enable non-trivial 4/8-step results. We discuss limitations and future directions in App. B.

AI USE STATEMENT

In this work, we used generative AI tools for designing or providing feedback on research methodology or experiments, implementing methods, and interpreting results. We have not used generative AI tools for helping develop theoretical models or conceptual frameworks, formulating mathematical claims, providing critical ingredients for proving mathematical claims, assisting in the writing of proofs, proposing or refining hypotheses, and supporting qualitative and thematic data analysis; generating synthetic data sets, assisting with translation, cleaning and reformatting dataset are not applicable to this work. Additionally, we used generative AI tools for creating or modifying scientific figures or images, creating or editing software code, drafting parts of the paper, summarizing or analyzing existing literature, and editing the paper to improve readability. We have reviewed all AI-assisted work. We manually checked the AI-written literature review and AI-implemented experimental results, and AI-generated code was verified and tested for correctness. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

REFERENCES

Niket Agarwal, Arslan Ali, Maciej Bala, Yogesh Balaji, Erik Barker, Tiffany Cai, Prithvijit Chattopadhyay, Yongxin Chen, Yin Cui, Yifan Ding, et al. Cosmos world foundation model platform for physical AI. *arXiv preprint arXiv:2501.03575*, 2025.

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 *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=tyEyYT267x.

Jacob Austin, Daniel D. Johnson, Jonathan Ho, Daniel Tarlow, and Rianne van den Berg. Structured denoising diffusion models in discrete state-spaces. In M. Ranzato, A. Beygelzimer, Y. Dauphin, P.S. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 17981–17993. Curran Associates, Inc., 2021a. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/958c530554f78bcd8e97125b70e6973d-Paper.pdf.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, et al. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021b.

Pavel Avdeyev, Chenlai Shi, Yuhao Tan, Kseniia Dudnyk, and Jian Zhou. Dirichlet diffusion score model for biological sequence generation. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 1276–1301. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/avdeyev23a.html.

Aarti Basant, Abhijit Khairnar, Abhijit Paithankar, Abhinav Khattar, Adithya Renduchintala, Aditya Malte, Akhiad Bercovich, Akshay Hazare, Alejandra Rico, Aleksander Ficek, et al. Nvidia nemotron nano 2: An accurate and efficient hybrid mamba-transformer reasoning model. *arXiv preprint arXiv:2508.14444*, 2025.

Georgios Batzolis, Mark Girolami, and Luca Ambrogioni. CoBit: Language modeling with bitstream diffusion. *arXiv preprint arXiv:2605.07013*, 2026.

Tiwei Bie, Maosong Cao, Kun Chen, Lun Du, Mingliang Gong, Zhuochen Gong, Yanmei Gu, Jiaqi Hu, Zenan Huang, Zhenzhong Lan, et al. LLaDA2.0: Scaling up diffusion language models to 100B. *arXiv preprint arXiv:2512.15745*, 2025.

Yonatan Bisk, Rowan Zellers, Ronan Le Bras, Jianfeng Gao, and Yejin Choi. Piqa: Reasoning about physical commonsense in natural language. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 7432–7439, 2020.

Aaron Blakeman, Aaron Grattafiori, Aarti Basant, Abhibha Gupta, Abhinav Khattar, Adi Renduchintala, Aditya Vavre, Akanksha Shukla, Akhiad Bercovich, Aleksander Ficek, et al. Nemotron 3 Nano: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2512.20848*, 2025.

Nicholas Matthew Boffi, Michael Samuel Albergo, and Eric Vanden-Eijnden. How to build a consistency model: Learning flow maps via self-distillation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Di5apl8HSH.

Aakshita Chandiramani, Aaron Blakeman, Abdullahi Olaoye, Abhibha Gupta, Abhilash Somasamudramath, Abhinav Khattar, Adeola Adesoba, Adi Renduchintala, Adil Asif, Aditya Agrawal, et al. Nemotron 3 super: Open, efficient mixture-of-experts hybrid mamba-transformer model for agentic reasoning. *arXiv preprint arXiv:2604.12374*, 2026.

Huiwen Chang, Han Zhang, Lu Jiang, Ce Liu, and William T Freeman. MaskGIT: Masked generative image transformer. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2022.

Jannis Chemseddine, Gregor Kornhardt, and Gabriele Steidl. Spherical flows for sampling categorical data. *arXiv preprint arXiv:2605.05629*, 2026.

Boyuan Chen, Diego Martí Monsó, Yilun Du, Max Simchowitz, Russ Tedrake, and Vincent Sitzmann. Diffusion forcing: Next-token prediction meets full-sequence diffusion. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=yDo1ynArjj.

Hansheng Chen, Kai Zhang, Hao Tan, Leonidas Guibas, Gordon Wetzstein, and Sai Bi. pi-Flow: Policy-based few-step generation via imitation distillation. In *The Fourteenth International Conference on Learning Representations*, 2026a. URL https://openreview.net/forum?id=1vAte8dsap.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

Ting Chen, Ruixiang Zhang, and Geoffrey Hinton. Analog bits: Generating discrete data using diffusion models with self-conditioning. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=3itjR9QxFw.

Yuxin Chen, Chumeng Liang, Hangke Sui, Ruihan Guo, Chaoran Cheng, Jiaxuan You, and Ge Liu. LangFlow: Continuous diffusion rivals discrete in language modeling. *arXiv preprint arXiv:2604.11748*, 2026b.

Chaoran Cheng, Jiahan Li, Jian Peng, and Ge Liu. Categorical flow matching on statistical manifolds. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=5fybcQZ0g4.

Shuang Cheng, Yihan Bian, Dawei Liu, Yuhua Jiang, Yihao Liu, Linfeng Zhang, Qian Yao, Zhongbo Tian, Wenhai Wang, Qipeng Guo, Kai Chen, Biqing Qi, and Bowen Zhou. SDAR: A synergistic diffusion-AutoRegression paradigm for scalable sequence generation. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Findings of the Association for Computational Linguistics: ACL 2026*, pp. 22058–22075, San Diego, California, United States, July 2026. Association for Computational Linguistics. ISBN 979-8-89176-395-1. doi: 10.18653/v1/2026.findings-acl.1110. URL https://aclanthology.org/2026.findings-acl.1110/.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try ARC, the AI2 reasoning challenge. *arXiv preprint arXiv:1803.05457*, 2018.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Oscar Davis, Samuel Kessler, Mircea Petrache, Ismail Ilkan Ceylan, Michael M. Bronstein, and Joey Bose. Fisher flow matching for generative modeling over discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=6jOScqwdHU.

Oscar Davis, Anastasiia Filippova, Pierre Ablin, Victor Turrisi, Amitis Shidani, Marco Cuturi, and Louis Béthune. Scaling categorical flow maps. *arXiv preprint arXiv:2605.07820*, 2026.

Justin Deschenaux and Caglar Gulcehre. Language modeling with hyperspherical flows. *arXiv preprint arXiv:2605.11125*, 2026.

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, et al. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, Nicolai Oswald, Enze Xie, Daniel Egert, Sharath Turuvekere Sreenivas, Shizhe Diao, et al. Nemotron-labs-diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv preprint arXiv:2607.05722*, 2026.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=uWj4s7rMnR.

Shansan Gong, Shivam Agarwal, Yizhe Zhang, Jiacheng Ye, Lin Zheng, Mukai Li, Chenxin An, Peilin Zhao, Wei Bi, Jiawei Han, Hao Peng, and Lingpeng Kong. Scaling diffusion language models via adaptation from autoregressive models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=j1tSLYKwg8.

Ishaan Gulrajani and Tatsunori Hashimoto. Likelihood-based diffusion language models. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=e2MCL6hObn.

Hongcan Guo, Qinyu Zhao, Yian Zhao, Shen Nie, Rui Zhu, Qiushan Guo, Feng Wang, Tao Yang, Hengshuang Zhao, Guoqiang Wei, et al. Continuous latent diffusion language model. *arXiv preprint arXiv:2605.06548*, 2026.

Xiaochuang Han, Sachin Kumar, and Yulia Tsvetkov. SSD-LM: Semi-autoregressive simplex-based diffusion language model for text generation and modular control. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11575–11596, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.647. URL https://aclanthology.org/2023.acl-long.647/.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=d7KBjmI3GmQ.

Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022. URL https://arxiv.org/abs/2207.12598.

Jonathan Ho, Tim Salimans, Alexey Gritsenko, William Chan, Mohammad Norouzi, and David Fleet. Video diffusion models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 8633–8646. Curran Associates, Inc., 2022. doi: 10.52202/068431-0628. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/39235c56aef13fb05a6adc95eb9d8d66-Paper-Conference.pdf.

Emiel Hoogeboom, Didrik Nielsen, Priyank Jaini, Patrick Forré, and Max Welling. Argmax flows and multinomial diffusion: Learning categorical distributions. In A. Beygelzimer, Y. Dauphin, P. Liang, and J. Wortman Vaughan (eds.), *Advances in Neural Information Processing Systems*, 2021. URL https://openreview.net/forum?id=6nbpPqUCIi7.

Daniel G Horvitz and Donovan J Thompson. A generalization of sampling without replacement from a finite universe. *Journal of the American statistical Association*, 47(260):663–685, 1952.

Keya Hu, Linlu Qiu, Yiyang Lu, Hanhong Zhao, Tianhong Li, Yoon Kim, Jacob Andreas, and Kaiming He. ELF: Embedded language flows. *arXiv preprint arXiv:2605.10938*, 2026.

Xun Huang, Zhengqi Li, Guande He, Mingyuan Zhou, and Eli Shechtman. Self forcing: Bridging the train-test gap in autoregressive video diffusion. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=mSiN7i0BYH.

Inception Labs, Samar Khanna, Siddhant Kharbanda, Shufan Li, Harshit Varma, Eric Wang, Sawyer Birnbaum, Ziyang Luo, Yanis Miraoui, Akash Palrecha, et al. Mercury: Ultra-fast language models based on diffusion. *arXiv preprint arXiv:2506.17298*, 2025.

Jaehyeong Jo and Sung Ju Hwang. Continuous diffusion model for language modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=VGv5y60sXC.

Tero Karras, Miika Aittala, Timo Aila, and Samuli Laine. Elucidating the design space of diffusion-based generative models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=k7FuTOWMOc7.

Tero Karras, Miika Aittala, Tuomas Kynkäänniemi, Jaakko Lehtinen, Timo Aila, and Samuli Laine. Guiding a diffusion model with a bad version of itself. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=bg6fVPVs3s.

Diederik Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. *Advances in neural information processing systems*, 34:21696–21707, 2021. URL https://openreview.net/forum?id=2LdBqxc1Yv.

Chanhyuk Lee, Jaehoon Yoo, Manan Agarwal, Sheel Shah, Jerry Huang, Aditi Raghunathan, Seunghoon Hong, Nicholas M Boffi, and Jinwoo Kim. Flow map language models: One-step language modeling via continuous denoising. *arXiv preprint arXiv:2602.16813*, 2026.

Aitor Lewkowycz, Anders Johan Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Venkatesh Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. Solving quantitative reasoning problems with language models. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=IFXTZERXdM7.

Pengxiang Li, Shilin Yan, Jiayin Cai, Renrui Zhang, Ruichuan An, Ziyu Guo, and Xiaowei Gao. Adaptive classifier-free guidance via dynamic low-confidence masking. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=0EcomthIKh.

Xiang Lisa Li, John Thickstun, Ishaan Gulrajani, Percy Liang, and Tatsunori Hashimoto. Diffusion-LM improves controllable text generation. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=3s9IrEsjLyk.

Jiajun Liang, Yucheng Liao, Yukang Cao, Jiazhe Wei, Ken Li, Wende Tan, Jiankun Zhang, ZY Cui, Jingkang Yang, Liucheng Guo, et al. AURORA-LM: Autoencoding unified representation for continuous-latent diffusion language modeling. *arXiv preprint arXiv:2608.02602*, 2026.

Alexander H Liu, Kartik Khandelwal, Sandeep Subramanian, Victor Jouault, Abhinav Rastogi, Adrien Sadé, Alan Jeffares, Albert Jiang, Alexandre Cahill, Alexandre Gavaudan, et al. Ministral 3. *arXiv preprint arXiv:2601.08584*, 2026a.

Jingyu Liu, Xin Dong, Zhifan Ye, Rishabh Mehta, Yonggan Fu, Vartika Singh, Ce Zhang, and Pavlo Molchanov. TiDAR: Think in diffusion, talk in autoregression. In A. Chowdhery and Z. Jia (eds.), *Proceedings of Machine Learning and Systems*, volume 8, pp. 748–762. MLSys, 2026b. URL https://proceedings.mlsys.org/paper_files/paper/2026/file/1367d856028f65a9555b0274db09e608-Paper-Conference.pdf.

Justin Lovelace, Varsha Kishore, Chao Wan, Eliot Seo Shekhtman, and Kilian Q Weinberger. Latent diffusion for language generation. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=NKdtztladR.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-solver: A fast ODE solver for diffusion probabilistic model sampling in around 10 steps. In Alice H. Oh, Alekh Agarwal, Danielle Belgrave, and Kyunghyun Cho (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=2uAaGwlP_V.

Cheng Lu, Yuhao Zhou, Fan Bao, Jianfei Chen, Chongxuan Li, and Jun Zhu. DPM-Solver++: Fast solver for guided sampling of diffusion probabilistic models. *Machine Intelligence Research*, 22 (4):730–751, 2025.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=aBsCjcPu_tE.

Viacheslav Meshchaninov, Egor Chimbulatov, Alexander Shabalin, Aleksandr Abramov, and Dmitry Vetrov. Cosmos: Compressed and smooth latent space for text diffusion modeling. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=Rv6Lz84FlZ.

Viacheslav Meshchaninov, Alexander Shabalin, Egor Chimbulatov, Nikita Gushchin, Ilya Koziev, Alexander Korotin, and Dmitry Vetrov. How to train your latent diffusion language model jointly with the latent space. *arXiv preprint arXiv:2605.07933*, 2026.

Mistral AI. Mistral NeMo. https://mistral.ai/news/mistral-nemo/, July 2024. Accessed: 2026-09-14.

Léo Nicollier, Max Dunitz, Marc Pic, Pablo Musé, Enric Meinhardt-Llopis, and Gabriele Facciolo. SPHERE-JEPA: Spherical prediction with homogeneous embeddings. *arXiv preprint arXiv:2605.26900*, 2026.

Shen Nie, Fengqi Zhu, Chao Du, Tianyu Pang, Qian Liu, Guangtao Zeng, Min Lin, and Chongxuan Li. Scaling up masked diffusion models on text. In *The Thirteenth International Conference on Learning Representations*, 2025a. URL https://openreview.net/forum?id=WNvvwK0tut.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025b. URL https://openreview.net/forum?id=KnqiC0znVF.

Hunter Nisonoff, Junhao Xiong, Stephan Allenspach, and Jennifer Listgarten. Unlocking guidance for discrete state-space diffusion and flow models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=XsgHl54yO7.

Jingyang Ou, Shen Nie, Kaiwen Xue, Fengqi Zhu, Jiacheng Sun, Zhenguo Li, and Chongxuan Li. Your absorbing discrete diffusion secretly models the conditional distributions of clean data. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=sMyXP8Tanm.

Peter Potaptchik, Jason Yim, Adhi Saravanan, Peter Holderrieth, Eric Vanden-Eijnden, and Michael S Albergo. Discrete flow maps. *arXiv preprint arXiv:2604.09784*, 2026.

Herbert E. Robbins. *An Empirical Bayes Approach to Statistics*, pp. 388–394. Springer New York, New York, NY, 1992. ISBN 978-1-4612-0919-5. doi: 10.1007/978-1-4612-0919-5_26. URL https://doi.org/10.1007/978-1-4612-0919-5_26.

Amirmojtaba Sabour, Sanja Fidler, and Karsten Kreis. Align your flow: Scaling continuous-time flow map distillation. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=pzHuesCvcO.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Mariano Marroquin, Justin T Chiu, Alexander M Rush, and Volodymyr Kuleshov. Simple and effective masked diffusion language models. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=L4uaAR4ArM.

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin T Chiu, and Volodymyr Kuleshov. The diffusion duality. In *Forty-second International Conference on Machine Learning*, 2025. URL https://openreview.net/forum?id=9P9Y8FOSOk.

Keisuke Sakaguchi, Ronan Le Bras, Chandra Bhagavatula, and Yejin Choi. Winogrande: An adversarial winograd schema challenge at scale. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 34, pp. 8732–8740, 2020.

Tim Salimans and Jonathan Ho. Progressive distillation for fast sampling of diffusion models. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=TIdIXIpzhoI.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P de Almeida, Alexander M Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=i5MrJ6g5G1.

Neta Shaul, Chao Liu, Arash Vahdat, and Julius Berner. Parallel decoding distillation for fast image and video generation. *arXiv preprint arXiv:2607.26004*, 2026.

Jiaxin Shi, Kehang Han, Zhe Wang, Arnaud Doucet, and Michalis Titsias. Simplified and generalized masked diffusion for discrete data. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=xcqSOfHt4g.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. In *International Conference on Learning Representations*, 2021a. URL https://openreview.net/forum?id=St1giarCHLP.

Yang Song, Jascha Sohl-Dickstein, Diederik P Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations*, 2021b. URL https://openreview.net/forum?id=PxTIG12RRHS.

Yang Song, Prafulla Dhariwal, Mark Chen, and Ilya Sutskever. Consistency models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan Sabato, and Jonathan Scarlett (eds.), *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 32211–32252. PMLR, 23–29 Jul 2023. URL https://proceedings.mlr.press/v202/song23a.html.

Yuxuan Song, Zheng Zhang, Cheng Luo, Pengyang Gao, Fan Xia, Hao Luo, Zheng Li, Yuehang Yang, Hongli Yu, Xingwei Qu, et al. Seed diffusion: A large-scale diffusion language model with high-speed inference. *arXiv preprint arXiv:2508.02193*, 2025.

Robin Strudel, Corentin Tallec, Florent Altché, Yilun Du, Yaroslav Ganin, Arthur Mensch, Will Grathwohl, Nikolay Savinov, Sander Dieleman, Laurent Sifre, et al. Self-conditioned embedding diffusion for text generation. *arXiv preprint arXiv:2211.04236*, 2022.

Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. TESS 2: A large-scale generalist diffusion language model. pp. 21171–21188, July 2025. doi: 10.18653/v1/2025.acl-long.1029. URL https://aclanthology.org/2025.acl-long.1029/.

Yunshu Wu, Jiayi Cheng, Longxuan Yu, Partha Thakuria, Rob Brekelmans, Evangelos E Papalexakis, and Greg Ver Steeg. Discrete stochastic localization for non-autoregressive generation. *arXiv preprint arXiv:2602.16169*, 2026.

Jinheng Xie, Weijia Mao, Zechen Bai, David Junhao Zhang, Weihao Wang, Kevin Qinghong Lin, Yuchao Gu, Zhijie Chen, Zhenheng Yang, and Mike Zheng Shou. Show-o: One single transformer to unify multimodal understanding and generation. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=o6Ynz6OIQ6.

Yanbo Xu, Yu Wu, Sungjae Park, Zhizhuo Zhou, and Shubham Tulsiani. Temporal score rescaling for temperature sampling in diffusion and flow models. In *Forty-third International Conference on Machine Learning*, 2026. URL https://openreview.net/forum?id=WbfRe0cheB.

Ling Yang, Ye Tian, Bowen Li, Xinchen Zhang, Ke Shen, Yunhai Tong, and Mengdi Wang. MMaDA: Multimodal large diffusion language models. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025. URL https://openreview.net/forum?id=wczmXLuLGd.

Zhihan Yang, Wei Guo, Shuibai Zhang, Subham Sekhar Sahoo, Yongxin Chen, Arash Vahdat, Morteza Mardani, and John Thickstun. Continuous diffusion scales competitively with discrete diffusion for language. *arXiv preprint arXiv:2605.18530*, 2026.

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

Tianwei Yin, Michaël Gharbi, Taesung Park, Richard Zhang, Eli Shechtman, Fredo Durand, and William T. Freeman. Improved distribution matching distillation for fast image synthesis. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*, 2024. URL https://openreview.net/forum?id=tQukGCDaNT.

Tianwei Yin, Qiang Zhang, Richard Zhang, William T. Freeman, Fredo Durand, Eli Shechtman, and Xun Huang. From Slow Bidirectional to Fast Autoregressive Video Diffusion Models . In *2025 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 22963–22974, Los Alamitos, CA, USA, June 2025. IEEE Computer Society. doi: 10.1109/CVPR52734.2025.02138. URL https://doi.ieeecomputersociety.org/10.1109/CVPR52734.2025.02138.

Zebin You, Shen Nie, Xiaolu Zhang, JUN ZHOU, Zhiwu Lu, Ji-Rong Wen, and Chongxuan Li. LLaDA-V: Large language diffusion models with visual instruction tuning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10093–10105, June 2026.

Longxuan Yu, Yunshu Wu, Yu Fu, Siheng Xiong, Rob Brekelmans, Hui Liu, Yue Dong, and Greg Ver Steeg. DSL-LLaDA: Scaling continuous denoising to 8B masked diffusion LMs. *arXiv preprint arXiv:2606.01024*, 2026.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. HellaSwag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pp. 4791–4800, Florence, Italy, July 2019. Association for Computational Linguistics. URL https://aclanthology.org/P19-1472/.

Qinsheng Zhang and Yongxin Chen. Fast sampling of diffusion models with exponential integrator. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Loek7hfb46P.

Chunting Zhou, LILI YU, Arun Babu, Kushal Tirumala, Michihiro Yasunaga, Leonid Shamis, Jacob Kahn, Xuezhe Ma, Luke Zettlemoyer, and Omer Levy. Transfusion: Predict the next token and diffuse images with one multi-modal model. In *The Thirteenth International Conference on Learning Representations*, 2025. URL https://openreview.net/forum?id=SI2hI0frk6.

Fengqi Zhu, Zebin You, Yipeng Xing, Zenan Huang, Lin Liu, Yihong Zhuang, Guoshan Lu, Kangyu Wang, Xudong Wang, Lanning Wei, et al. LLaDA-MoE: A sparse MoE diffusion language model. *arXiv preprint arXiv:2509.24389*, 2025.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-reduced preference optimization for large language diffusion models. In Maria Liakata, Viviane P. Moreira, Jiajun Zhang, and David Jurgens (eds.), *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 11425–11460, San Diego, California, United States, July 2026a. Association for Computational Linguistics. ISBN 979-8-89176-390-6. doi: 10.18653/v1/2026.acl-long.524. URL https://aclanthology.org/2026.acl-long.524/.

Yuchen Zhu, Jing Shi, Chongjian Ge, Hao Tan, Yiran Xu, Wanrong Zhu, Jason Kuen, Koustava Goswami, Rajiv Jain, Yongxin Chen, et al. FLARE: Diffusion for hybrid language model. *arXiv preprint arXiv:2606.01774*, 2026b.

Shaobin Zhuang, Yuang Ai, Jiaming Han, Xiaohui Li, Huaibo Huang, Xiangyu Yue, Xuefeng Hu, Kun Xu, Yali Wang, and Hao Chen. BitLM: Unlocking multi-token language generation with bitwise continuous diffusion. *arXiv preprint arXiv:2605.11577*, 2026.

CONTENTS

A Related Work **20**

B Limitations and Future Work **22**

C Additional Background: Masked Diffusion Language Models **23**

D Training and Inference Algorithms **26**

E Implementation Details and Additional Results for Pre-training **28**

E.1 Sketched Uniform Spherical Regularization (SUSReg) 28

E.2 Ablation Study: MDLM Initialization for Pre-training 29

F Implementation Details and Additional Results for Likelihood-based Benchmarks **31**

G Implementation Details and Additional Results for Inference **33**

G.1 Reasoning Benchmark Setup 33

G.2 Sweeping Classifier-Free Guidance and Score Temperature 34

G.3 Ablation Study: Steering in Logit Space Instead of Embedding Space 34

G.4 Ablation Study: Choosing the Bad Prediction for CFG with Self-conditioning . . . 38

G.5 Ablation Study: Commit-Stride Sampling 39

G.6 Ablation Study: Time Discretization 41

G.7 Higher-order ODE Solvers 42

G.8 Pass@k Evaluation 45

G.8.1 Base Model: MBPP 45

G.8.2 Base Model: Minerva Geometry 46

G.8.3 Base Model: GSM8K 47

G.8.4 Instruct Models: AIME 2025 48

G.9 Reasoning Trajectory Analysis 49

G.10 Wall-clock Cost of the Packed CFG Forward 49

G.11 Proofs 49

H Implementation Details and Additional Results for Supervised Fine-Tuning **53**

H.1 Supervised Fine-Tuning Setup 53

H.2 Evaluation Setup and Frozen-Parameter Ablation 54

I Implementation Details and Additional Results for Parallel Decoding Distillation **56**

I.1 Parallel Predictions and Intra-Segment Rollout 56

I.2 Residual Parallel Heads 56

I.3 Distillation Objective and Optimization 57

I.4 Few-Step Inference 58

I.5 Training and Parameterization Details 58

J Example Outputs **59**

J.1 *Sigma*-Base-3B 59

J.2 *Sigma*-3B-Instruct 60

A Related Work

[Back to Sec. 1 or Sec. 2 in the main paper.]

This appendix provides a literature review on continuous diffusion language models, the family to which *Sigma* belongs.

Continuous diffusion language models. Continuous dLMs lift discrete tokens into a Euclidean space and run a Gaussian (or Riemannian) diffusion there. A central attraction for doing so is that the machinery developed for image diffusion, such as efficient ODE solvers, guidance, and distillation techniques, transfers with little modification. Existing methods differ mainly in *where* the diffusion process lives:

- Diffusing *token embeddings*. Diffusion-LM (Li et al., 2022) pioneered this approach with Gaussian diffusion over learned word embeddings and a rounding step back to tokens; CDCD (Dieleman et al., 2022) refined the objective with score interpolation and a categorical cross-entropy loss over softmax logits, and self-conditioned embedding diffusion (Strudel et al., 2022) added self-conditioning on the previous denoising estimate. Plaid (Gulrajani & Hashimoto, 2023) tied these ideas together around a likelihood target by pairing low-dimensional sphere-normalized embeddings with a learned noise schedule and categorical reparameterization; RePlaid (Yang et al., 2026) aligns this recipe with modern DiT architectures and shows it scales competitively with discrete dLMs. Meanwhile, LangFlow (Chen et al., 2026b) follows the same learnable-embedding but uses cross-entropy training loss, while Wu et al. (2026) adopts a stochastic-localization formulation. $\mathbb{S}$-FLM (Deschenaux & Gulcehre, 2026) and Spherical Flows (Chemseddine et al., 2026) likewise use learnable token embeddings but further constrain the diffusion trajectory itself to the embedding hypersphere. A parallel strand fixes the embedding to be *one-hot* rather than learned: FLM (Lee et al., 2026) and DFM (Potaptchik et al., 2026) run flow matching from a Gaussian source to one-hot vocabulary targets in $\mathbb{R}^V$, and further distill the flow into a few-step flow map for fast generation.
- Diffusing an *autoencoded latent*. Here, a text autoencoder produces a compact continuous latent variable and diffusion is run in that latent space. LD4LG (Lovelace et al., 2023) runs latent diffusion on a frozen encoder–decoder (BART and T5); COSMOS (Meshchaninov et al., 2025), ELF (Hu et al., 2026) Cola-DLM (Guo et al., 2026), and AURORA-LM (Liang et al., 2026) scale this axis with different latent designs and training objectives; LDLM (Meshchaninov et al., 2026) learns the encoder and the diffusion prior jointly rather than sequentially.
- Diffusing on the *simplex or other statistical manifolds*. These methods parameterize a categorical distribution over the vocabulary and diffuse its parameters directly, keeping the noise process on-manifold and avoiding a rounding step. SSD-LM (Han et al., 2023) runs simplex diffusion in a semi-autoregressive block structure, Dirichlet diffusion (Avdeyev et al., 2023) uses a Jacobi diffusion process on the probability simplex, and later variants (Cheng et al., 2024; Davis et al., 2024; Jo & Hwang, 2025) extend the geometry to Fisher–Rao and Riemannian flow matching.
- Reaching discrete data through *dequantization or a Gaussian–discrete duality*. Argmax flows (Hoogeboom et al., 2021) lift categorical data into a continuous space via argmax-based variational dequantization, so continuous flows can be trained under a discrete likelihood; analog bits (Chen et al., 2023), BitLM (Zhuang et al., 2026), and CoBit (Batzolis et al., 2026) instead map each token to a bitstring and diffuse from there; Duo (Sahoo et al., 2025) establishes a duality between continuous Gaussian and uniform-state discrete diffusion, letting the two families share training and sampling machinery.

Sigma sits in the token-embedding class, inheriting the learnable embedding and variational diffusion process of Plaid and RePlaid.

Scaling continuous dLMs to billions of parameters. Recent work pushes continuous dLMs into the billion-parameter regime, but along axes different from ours: for example, Cola-DLM (Guo et al., 2026) ($\sim$2B, hierarchical latent-prior modeling), AURORA-LM (Liang et al., 2026) (1B, flow matching over an autoencoded latent), and Categorical Flow Maps (Davis et al., 2026) (1.7B, self-distilled for few-step generation). A separate line changes the base representation and scales through an existing LM backbone: BitLM (Zhuang et al., 2026) places a bitwise diffusion head atop a causal LM up to 8B, TESS 2 (Tae et al., 2025) adapts an autoregressive Mistral-7B into a generalist diffusion model with a simplex cross-entropy loss, and DSL-LLaDA (Yu et al., 2026) scales

continuous denoising via discrete stochastic localization (Wu et al., 2026) on top of the LLaDA-8B masked-diffusion backbone (Nie et al., 2025b). In contrast, *Sigma* is, to our knowledge, the first likelihood-trained continuous dLM at 3B/8B, and the first evaluated head-to-head with a matched masked dLM (Fu et al., 2026) on conditional reasoning and code.

CFG in dLMs. Continuous dLMs realize the guidance condition in two ways. When there is no prompt (e.g., unconditional generation), several works repurpose the self-conditioning signal as the condition: ELF (Hu et al., 2026) and AURORA-LM (Liang et al., 2026) extrapolate between predictions made with and without the previous-step estimate. For conditional generation, DFM (Potaptchik et al., 2026), AURORA-LM (Liang et al., 2026), Cola-DLM (Guo et al., 2026), and BitLM (Zhuang et al., 2026) extrapolate between a prompt-conditioned and an unconditional prediction, as in image diffusion models. A cost for this is that enabling it requires dropping the prompt context during training. FLM (Lee et al., 2026) also leverages autoguidance (Karras et al., 2024) for prediction sharpening. For discrete dLMs, LLaDA (Nie et al., 2025b) popularized an "unsupervised" CFG that runs one forward with full prompt and one with the prompt masked, then linearly extrapolates the two logit vectors before renormalizing; this has since been refined by more general discrete guidance mechanisms (Schiff et al., 2025; Nisonoff et al., 2025). Our steering (Sec. 4.2) differs on three points: (*i*) the guidance condition is the clean prefix itself, and the unconditional branch simply drops that prefix so the noisy block attends only within itself; (*ii*) the extrapolation acts on the predicted clean embedding (equivalently the score), not on logits; (*iii*) it is a pure inference-time operation that requires no train-time dropout because sequence packing leaves the first block without context.

Autoregressive and block video diffusion. Our block-causal factorization (i.e., autoregressive across blocks, bidirectional Gaussian denoising within a block over a KV-cacheable clean prefix) closely mirrors recent video generation models that couple causal structure across frames with denoising within them (Chen et al., 2024; Yin et al., 2025; Huang et al., 2025). The bidirectional video-diffusion lineage (Ho et al., 2022) was the original target of this conversion, and the autoregressive arm of Cosmos (Agarwal et al., 2025) likewise structures video generation causally across frames. Diffusion Forcing (Chen et al., 2024) trains a causal model with independent per-frame noise levels, unifying next-token prediction with full-sequence diffusion and enabling stable rollouts past the training horizon. CausVid (Yin et al., 2025) and Self Forcing (Huang et al., 2025) convert bidirectional video diffusion into causal, KV-cached autoregressive generators for fast streaming synthesis while curbing error accumulation along the rollout. These works share our central design tension (i.e., coupling causal, autoregressive structure with within-block diffusion) and several of their techniques (per-block noise schedules, rolling KV caches, distribution-matching distillation) transfer naturally to the language setting.

B Limitations and Future Work

[Back to Sec. 7 in the main paper.]

Despite the promising performance of continuous dLMs for language reasoning, several limitations remain. In this section, we outline these key challenges alongside promising directions for future research.

Inference Speed and Few-Step Trajectory Distillation. Continuous dLM decoding remains slower per generated token at high step counts compared to fully optimized autoregressive baselines. Under low step budgets, DDIM-4/8 exhibits a major performance drop relative to DDPM-256 (Tab. 1). We observe that neither higher-order ODE solvers (App. G.7) nor commit-stride sampling (App. G.5) fully bridge this gap. To address this, we explore preliminary on-policy and off-policy Parallel Decoding Distillation (PDD), observing encouraging performance gains. Significant headroom remains in optimizing PDD head parameterizations, training objectives, and policy learning strategies. Given that standard DDIM sampling with 16 NFEs maintains decent performance, effective distillation holds substantial potential for $4\times$–$8\times$ acceleration, leveraging insights from continuous trajectory distillation in visual domains. Beyond PDD, extending trajectory distillation to consistency and flow-matching frameworks provides a promising path toward real-time continuous dLM generation. Finally, since our model involves self-conditioning, how to distill it into the few-step generative model remains an open question.

Encoder–Decoder Robustness. We observe that diffusion-based ELBO training tends to collapse the continuous embedding geometry because the readout decoder (the linear logit projection) is trained predominantly to reconstruct clean, final-step token embeddings. Consequently, the decoder becomes hypersensitive to subtle continuous perturbations. This sensitivity becomes severe when latent representations fall out-of-distribution (OOD), which occurs frequently during early denoising stages, low-NFE sampling, trajectory distillation, or aggressive score steering via CFG and ST (Fig. 15). This latent-to-discrete mismatch might induce decoding errors that bottleneck reasoning performance, especially in code generation where a single syntax error yields complete benchmark failure. To mitigate embedding collapse and strengthen decoder robustness, future research could explore auxiliary cross-entropy losses evaluated across all diffusion timesteps or robustify the readout decoder via VAE-style pre-training prior to diffusion alignment.

Memory Overhead and Blockwise KV-Caching Dynamics. Operating blockwise continuous diffusion requires evaluating the transformer denoiser across N reverse steps per continuous block. Although blockwise generation offers parallel decoding potential, repeatedly updating and querying key-value (KV) states across intermediate noise timesteps increases peak memory bandwidth demand compared to standard single-pass autoregressive KV-caching. Optimizing persistent KV-cache management, state-reuse across intermediate diffusion steps, and extending blockwise continuous diffusion to ultra-long context windows ($>$ 32k tokens) remain important systems challenges for future hardware deployments.

Reward-Driven Inference-Time Scaling and Post-Training. Both inference-time reward steering and reward-driven post-training remain largely unexplored for continuous dLMs. At inference, our CFG + ST framework extends naturally to reward-model guidance, sequential Monte Carlo over the denoising SDE, and tilted sampling toward verifier-preferred regions. For post-training, continuous dLMs can in principle inherit the RL toolbox developed for image and video diffusion; whether it transfers from perceptual rewards to language verifiers (math checkers, unit tests, preference models) is an open direction.

Multimodal Reasoning and Unified Latents. Because the probability-flow ODE is natively well-defined in our continuous formulation, standard continuous image-diffusion techniques such as trajectory matching (Song et al., 2023; Geng et al., 2025; Sabour et al., 2025; Boffi et al., 2025; Chen et al., 2026a; Shaul et al., 2026) and distribution matching (Yin et al., 2024) can be directly imported into language modeling. Continuous diffusion is a natural fit for unified multimodal generation architectures (Zhou et al., 2025; Xie et al., 2025; Yang et al., 2025) that bridge text with images, video, or audio. Extending *Sigma* to joint multimodal continuous trajectories presents a compelling path forward.

C Additional Background: Masked Diffusion Language Models

[Back to Sec. 3.1 in the main paper.]

Notation. We restate masked diffusion models (MDMs, Austin et al. (2021a); Sahoo et al. (2024); Shi et al. (2024); Ou et al. (2025)) in the notation of Secs. 2 and 3.1. Many notations are shared and carry their usual meaning: the clean sequence $\mathbf{x}$, identified with its one-hot matrix in $\{0,1\}^{L\times V}$ with rows $\mathbf{x}^1,\dots,\mathbf{x}^L$; the vocabulary $\mathcal{V}$ of size V (now including mask token $\mathbf{m}$); the simplex Δ^V; and the denoiser $\mathbf{x}_\theta$, which in either paradigm maps a noisy state to a per-position categorical distribution in $(\Delta^V)^L \subset \mathbb{R}^{L\times V}$. However, there are a few key differences:

- The latent $\mathbf{z}_t$. Continuous: a Gaussian-noised embedding $\mathbf{z}_t \in \mathbb{R}^{L\times d_e}$ (1). Masked: a partially masked token sequence $\mathbf{z}_t \in \mathcal{V}^L$ (7), identified with its one-hot matrix like $\mathbf{x}$. The denoiser $\mathbf{x}_\theta(\mathbf{z}_t, t)$ and reverse chain p_θ consume whichever of the two is in force.
- The noise schedule α_t. Continuous: the variance-preserving signal scale, $\alpha_t^2 + \sigma_t^2 = 1$, learned through $\gamma(t)$ (see (1)). Masked: the probability that a position is still kept unmasked (i.e., clean) at time t, decreasing from $\alpha_0 = 1$ to $\alpha_1 = 0$ and fixed to be the linear schedule $\alpha_t = 1 - t$ by default.
- The endpoints $t = 0$ and $t = 1$. Masked: exact. $\alpha_0 = 1$ gives the clean sequence $\mathbf{x}$ and $\alpha_1 = 0$ the fully masked one, so the bound needs neither a prior nor a reconstruction term. Continuous: approximate. The learned endpoints $\gamma_0 < \gamma_1$ leave a finite SNR at both ends, so $\mathbf{z}_0$ is a minimally noisy latent that still has to be decoded and $\mathbf{z}_1$ is only approximately $\mathcal{N}(\mathbf{0}, \mathbf{I})$.

The embedding objects $\mathbf{e}$, $\mathbf{E}$, d_e and the signal-to-noise schedule $\mathrm{SNR}(t)$ have no masked counterpart and do not appear below.

Forward process. MDMs construct the noising process by random masking and generate samples by iteratively unmasking from a fully masked sequence. The marginal $q(\mathbf{z}_t \mid \mathbf{x})$ factorizes across positions and independently keeps each token clean with probability α_t or replaces it with the mask token $\mathbf{m}$:

$$q(\mathbf{z}_t^l \mid \mathbf{x}^l) = \mathrm{Cat}(\mathbf{z}_t^l; \alpha_t\, \mathbf{x}^l + (1-\alpha_t)\, \mathbf{m}). \tag{7}$$

The process is *absorbing*: once a position is masked, it remains masked for the rest of the process. Combined with (7), this pins down the two-time conditional for any $0 \le s < t \le 1$:

$$q(\mathbf{z}_s^l \mid \mathbf{z}_t^l, \mathbf{x}^l) = \begin{cases} \mathrm{Cat}\,(\mathbf{z}_s^l; \mathbf{z}_t^l), & \mathbf{z}_t^l \neq \mathbf{m}, \\ \mathrm{Cat}\left(\mathbf{z}_s^l; \frac{\alpha_s - \alpha_t}{1-\alpha_t}\, \mathbf{x}^l + \frac{1-\alpha_s}{1-\alpha_t}\, \mathbf{m}\right), & \mathbf{z}_t^l = \mathbf{m}, \end{cases} \tag{8}$$

where the first case copies an already-unmasked position unchanged.

Denoiser and sequence-level NELBO. As in Sec. 2, a denoiser $\mathbf{x}_\theta(\mathbf{z}_t, t) \in (\Delta^V)^L \subset \mathbb{R}^{L\times V}$ returns a categorical distribution over the vocabulary at each masked position, the only difference being that $\mathbf{z}_t$ is now the masked state of (7) rather than a noisy embedding. Since it predicts *clean* tokens, it places no mass on the mask token, $\langle \mathbf{x}_\theta^l(\mathbf{z}_t, t), \mathbf{m}\rangle = 0$, so it can be substituted for $\mathbf{x}^l$ wherever the latter appears. Substituting $\mathbf{x}_\theta$ for the one-hot $\mathbf{x}$ in (8) to define the reverse chain p_θ and, by taking the continuous-time limit, one obtains the NELBO as a weighted cross-entropy on the masked positions (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025):

$$-\log p_\theta(\mathbf{x}) \le \mathcal{L}_{\mathrm{MDLM}}(\mathbf{x}) := \mathbb{E}_{t\sim\mathrm{Unif}[0,1]}\, \mathbb{E}_{q(\mathbf{z}_t\mid\mathbf{x})} \left[\frac{\alpha_t'}{1-\alpha_t} \sum_{l=1}^{L} \mathbf{1}_{\mathbf{z}_t^l = \mathbf{m}} \log \langle \mathbf{x}_\theta^l(\mathbf{z}_t, t), \mathbf{x}^l \rangle \right]. \tag{9}$$

Given $\alpha_t = 1 - t$, the weight becomes $-\frac{1}{t}$. The loss is invariant to the shape of α_t and only depends on the endpoints $\alpha_0 = 1$ and $\alpha_1 = 0$ (Kingma et al., 2021; Sahoo et al., 2024). Interestingly, among several choices, the linear schedule minimizes the variance of the loss (Sahoo et al., 2024, E.1.1).

Optimal denoiser. Let $\theta^\star$ be the minimizer of (9). The bound is a cross entropy, so $\mathbf{x}_{\theta^\star}$ is the posterior over the clean token given the partially masked state (Sahoo et al., 2024; Shi et al., 2024; Ou et al., 2025); in the one-hot notation this posterior *is* a conditional mean, exactly as $\mathbf{e}_\theta$ predicts $\mathbb{E}[\mathbf{e} \mid \mathbf{z}_t, t]$ in Sec. 2:

$$\mathbf{x}_{\theta^\star}^l(\mathbf{z}_t, t) = \mathbb{E}\left[\mathbf{x}^l \mid \mathbf{z}_t\right] \in \Delta^V. \tag{10}$$

The NELBO scores masked positions only, so it constrains $\mathbf{x}^l_{\theta^\star}$ only where $\mathbf{z}^l_t = \mathbf{m}$. At the remaining positions $\mathbf{z}^l_t$ already reveals $\mathbf{x}^l$, so the same expression returns $\mathbb{E}[\mathbf{x}^l \mid \mathbf{z}_t] = \mathbf{z}^l_t$: the denoiser copies its input, which is what the absorbing property of (8) assumes. The right-hand side of optimal denoiser (10) does not explicitly depend on t and involves it only through $\mathbf{z}_t$, i.e., conditioned on the masked state, the time carries no further information about $\mathbf{x}$(Ou et al., 2025). Models in this family accordingly take the state alone as input, and we write $\mathbf{x}_\theta(\mathbf{z}_t)$, dropping the time argument, from here on.

Ancestral reverse sampling. Sampling starts from the fully masked sequence $\mathbf{z}_1 = (\mathbf{m}, \dots, \mathbf{m})$ and steps along a discretization $1 = t_0 > t_1 > \cdots > t_N = 0$ by iterating (8) with $\mathbf{x}_\theta$ in place of $\mathbf{x}$:

$$p_\theta(\mathbf{z}^l_s \mid \mathbf{z}_t) = \begin{cases} \mathrm{Cat}(\mathbf{z}^l_s; \mathbf{z}^l_t), & \mathbf{z}^l_t \neq \mathbf{m}, \\ \mathrm{Cat}\left(\mathbf{z}^l_s; \frac{\alpha_s - \alpha_t}{1-\alpha_t}\, \mathbf{x}^l_\theta(\mathbf{z}_t) + \frac{1-\alpha_s}{1-\alpha_t}\, \mathbf{m}\right), & \mathbf{z}^l_t = \mathbf{m}. \end{cases} \tag{11}$$

In other words, at each step and each masked position l, independently, one decides whether to unmask this position with probability $\frac{\alpha_s - \alpha_t}{1-\alpha_t}$. If decided to unmask, the token is sampled from the distribution $\mathbf{x}^l_\theta(\mathbf{z}_t) \in \Delta^V$. Once unmasked, a token is frozen for the rest of the trajectory.

Confidence-based (greedy) sampling. Ancestral sampling with a large vocabulary and few steps often commits low-probability tokens early and cannot revisit them. *Confidence-based* samplers (Chang et al., 2022; Nie et al., 2025b; Ye et al., 2025; Fu et al., 2026) choose which positions to unmask by predicted confidence, and a temperature $T \in [0, 1]$ (typically zero, i.e., deterministic argmax decoding) when sampling from the categorical distribution. Two variants are commonly used:

- *Top-k per step.* Fix a budget k_i at step i so that their sum is the sequence length. At step i, commit the k_i masked positions with the highest confidence $c^l := \|\mathbf{x}^l_\theta(\mathbf{z}_{t_i})\|_\infty$, and sample these positions from the probability distribution $\mathbf{x}^l_\theta(\mathbf{z}_{t_i})$ with temperature T. When $T = 0$, this is the default sampler of LLaDA (Nie et al., 2025b) and Dream (Ye et al., 2025). The LLaDA and Dream baselines reported in Tabs. 3 and 8 are generated with $k_1 = \ldots = k_L = 1$ (i.e., unmasking one position at a time with the highest confidence).
- *Threshold per step.* Fix a confidence threshold $\vartheta \in (0, 1)$. At step i, commit every masked position with $c^l \geq \vartheta$, and sample these positions from the probability distribution $\mathbf{x}^l_\theta(\mathbf{z}_{t_i})$ with temperature T. When $T = 0$, this is the greedy sampler of Nemotron-Labs-Diffusion (Fu et al., 2026) and defines our masked baseline in Tabs. 1 to 3 and 8 using $\vartheta \in \{0.8, 0.9\}$ as the threshold.

Like ancestral sampling, these samplers preserve the absorbing property that an unmasked token stays fixed for the rest of the trajectory.

Block diffusion. Similar to Sec. 3.1, the sequence $\mathbf{x}$ is partitioned into B contiguous blocks $\{\mathbf{x}^{(b)}\}_{b=1}^B$ of length $L' = L/B$, and the model denoises one block at a time given its clean prefix. Exactly as in (3), Arriola et al. (2025) factorizes the likelihood autoregressively *across* blocks and then bounds each factor, now with the masked bound in place of the continuous one:

$$-\log p_\theta(\mathbf{x}) = \sum_{b=1}^{B} -\log p_\theta(\mathbf{x}^{(b)} \mid \mathbf{x}^{(<b)}) \leq \sum_{b=1}^{B} \mathcal{L}_{\text{MDLM}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)}) =: \mathcal{L}_{\text{BD-MDLM}}(\mathbf{x}),$$

where we obtain $\mathcal{L}_{\text{MDLM}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)})$ by noising block b alone and conditioning the denoiser in (9) on the clean prefix $\mathbf{x}^{(<b)}$:

$$\begin{aligned} &\mathcal{L}_{\text{MDLM}}(\mathbf{x}^{(b)}; \mathbf{x}^{(<b)}) \\ &= \mathbb{E}_{t \sim \text{Unif}[0,1]}\, \mathbb{E}_{q(\mathbf{z}^{(b)}_t \mid \mathbf{x}^{(b)})} \left[\frac{\alpha'_t}{1-\alpha_t} \sum_{l=1}^{L'} \mathbf{1}_{\mathbf{z}^{(b,l)}_t = \mathbf{m}} \log \langle \mathbf{x}^l_\theta(\mathbf{z}^{(b)}_t; \mathbf{x}^{(<b)}), \mathbf{x}^{(b,l)} \rangle \right]. \end{aligned} \tag{12}$$

Training objective of NLD. A single Monte-Carlo sample of (12) per block (one time $t^{(b)} \sim \text{Unif}[0, 1]$ and a correspondingly sampled mask assignment, abbreviated as $\mathbf{z}^{(b)} := \mathbf{z}^{(b)}_{t^{(b)}}$) gives the NELBO estimator

$$\widehat{\mathcal{L}}_{\text{BD-MDLM}}(\mathbf{x}) = \sum_{b=1}^{B} \frac{\alpha'_{t^{(b)}}}{1-\alpha_{t^{(b)}}} \sum_{l=1}^{L'} \mathbf{1}_{\mathbf{z}^{(b,l)} = \mathbf{m}} \log \langle \mathbf{x}^l_\theta(\mathbf{z}^{(b)}; \mathbf{x}^{(<b)}), \mathbf{x}^{(b,l)} \rangle, \tag{13}$$

whose weight is $-\frac{1}{t^{(b)}}$, so each masked token contributes its cross entropy divided by its masking probability. NLD (Fu et al., 2026) instead draws one t per $\mathbf{x}$ and shares it across the B blocks rather than drawing $t^{(b)}$ independently. For validation we want the NELBO per token, whose exponential is the reported perplexity; over a batch $\mathbf{x}_1, \ldots, \mathbf{x}_N$ this normalizes by the total number of tokens:

$$\mathcal{L}_{\text{BD-VAL}} \;=\; \frac{1}{NL} \sum_{n=1}^{N} \widehat{\mathcal{L}}_{\text{BD-MDLM}}(\mathbf{x}_n). \tag{14}$$

NLD, however, trains with a different objective: the *global loss average* normalizes by the number of masked tokens actually drawn ($N_{\text{mask}}(\mathbf{x}) \;=\; \sum_{b=1}^{B} \sum_{l=1}^{L'} \mathbf{1}_{\mathbf{z}^{(b,l)}=\mathbf{m}}$), giving the NELBO per *masked* token, which is not a bound on the per-token likelihood:

$$\mathcal{L}_{\text{BD-GLA}} \;=\; \frac{\sum_{n=1}^{N} \widehat{\mathcal{L}}_{\text{BD-MDLM}}(\mathbf{x}_n)}{\sum_{n=1}^{N} N_{\text{mask}}(\mathbf{x}_n)}. \tag{15}$$

The two share a numerator and differ only by the batch's realized masking rate,

$$\mathcal{L}_{\text{BD-VAL}} \;=\; \mathcal{L}_{\text{BD-GLA}} \cdot \frac{\sum_{n=1}^{N} N_{\text{mask}}(\mathbf{x}_n)}{NL}, \qquad \mathbb{E}\left[\frac{N_{\text{mask}}(\mathbf{x})}{L}\right] = \mathbb{E}_{t\sim\text{Unif}[0,1]}(1-\alpha_t) = \frac{1}{2},$$

the last equality under the linear schedule. So $\mathcal{L}_{\text{BD-GLA}}$ overstates the per-token NELBO by roughly a factor of two. Both denominators are constant with respect to θ, so the two objectives differ only in scale: that scale is the fixed NL for $\mathcal{L}_{\text{BD-VAL}}$ and the random $\sum_n N_{\text{mask}}(\mathbf{x}_n)$ for $\mathcal{L}_{\text{BD-GLA}}$.

Scaling. Following the scaling study of masked dLM on text by Nie et al. (2025a), the field has scaled along two lines. The *from-scratch* line trains a masked dLM directly on raw data: LLaDA-8B (Nie et al., 2025b) rivals AR models of the same size, and LLaDA-MoE (Zhu et al., 2025) pushes the objective into a sparse mixture-of-experts backbone at 1.4B active parameters. The LLaDA-8B backbone has been further extended along orthogonal axes without additional pre-training scaling: LLaDA 1.5 (Zhu et al., 2026a) adds variance-reduced preference alignment and LLaDA-V (You et al., 2026) adds visual instruction tuning. The *AR-adaptation* line instead starts from a pretrained AR checkpoint and performances continual pre-training to turn it into a masked dLM. Dream-7B (Ye et al., 2025) converts Qwen2.5-7B into a masked dLM via the DiffuLLaMA shift-operation trick (Gong et al., 2025). The remainder of this line instead converts the AR checkpoint into a blockwise masked dLM (Arriola et al., 2025) that preserves KV caching across blocks, a conversion far cheaper than pre-training from scratch: Nemotron-Labs-Diffusion (Fu et al., 2026), FLARE (Zhu et al., 2026b), SDAR (Cheng et al., 2026) at 1.7∼30B (including a 30B MoE), and most recently LLaDA 2.0-mini/flash (Bie et al., 2025) at 16B and 100B (MoE), the largest open masked dLM to date. Closed-source frontier deployments such as Mercury (Inception Labs et al., 2025) and Seed Diffusion (Song et al., 2025) also fall into this block masked dLM paradigm.

D TRAINING AND INFERENCE ALGORITHMS

[Back to Sec. 3.2 or Sec. 4 in the main paper.]

We collect the two procedures described in the main paper:

- Alg. 1 is one optimization step of the block-wise NELBO (3) and (4) under the Horvitz–Thompson estimator (5) (Sec. 3.1);
- Alg. 2 is block-wise reverse sampling (Sec. 4.1) with CFG, ST, SC, and commit stride (Sec. 4.2).

Algorithm 1 *Sigma* training step for one sequence

Require: sequence $\mathbf{x} \in \{0,1\}^{L\times V}$; block size L', number of blocks $B = L/L'$; backbone parameters θ; embedding matrix $\mathbf{E} \in \mathbb{R}^{V\times d_e}$; noise schedule endpoints $\gamma_0 < \gamma_1$ and monotone interior shape $\tilde{\gamma}_\psi : [0,1] \to [0,1]$, giving $\gamma(t) = \gamma_0 + (\gamma_1 - \gamma_0)\tilde{\gamma}_\psi(t)$; reconstruction probability $p_{\text{rec}} \in (0,1)$ (default 0.1); self-conditioning probability $p_{\text{sc}} \in (0,1)$ (default 0.25); diffusion loss weight $w_{\text{diff}} > 0$ and optional AR-loss weight $w_{\text{ar}} \geq 0$; optional SUSReg weight $\lambda_{\text{susreg}} \geq 0$ with M slices and P Epps–Pulley quadrature points on $[-t_{\max}, t_{\max}]$ (defaults in App. E.1)

$\mathbf{e} \leftarrow \mathbf{x}\mathbf{E} \in \mathbb{R}^{L\times d_e}$ ▷ Embedding representation, each row has unit-norm
for $b = 1, \ldots, B$ **do** ▷ independent noise level and estimator branch per block
  $R^{(b)} \sim \text{Bern}(p_{\text{rec}})$
  $t^{(b)} \leftarrow 0$ **if** $R^{(b)} = 1$ **else** $t^{(b)} \sim \text{Unif}[0,1]$
  $\boldsymbol{\epsilon}^{(b)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$, $\mathbf{z}^{(b)} \leftarrow \alpha_{t^{(b)}}\mathbf{e}^{(b)} + \sigma_{t^{(b)}}\boldsymbol{\epsilon}^{(b)}$ ▷ Gaussian noising (1)
end for
$\mathbf{e}_{\text{sc}} \leftarrow \mathbf{0}$
if $u \sim \text{Unif}[0,1] < p_{\text{sc}}$ **then** ▷ SC pass carries no gradient
  $\mathbf{e}_{\text{sc}} \leftarrow \text{sg}\left(\mathbf{x}_\theta(\{\mathbf{z}^{(b)}\}, \{t^{(b)}\}; \mathbf{x}, \mathbf{0})\,\mathbf{E}\right)$
end if
$\mathbf{x}_\theta \leftarrow \mathbf{x}_\theta(\{\mathbf{z}^{(b)}\}, \{t^{(b)}\}; \mathbf{x}, \mathbf{e}_{\text{sc}})$, $\mathbf{e}_\theta \leftarrow \mathbf{x}_\theta\mathbf{E}$ ▷ one forward for all blocks; block-causal mask
for $b = 1, \ldots, B$ **do**
  $\mathcal{L}^{(b)}_{\text{rec}} \leftarrow -\sum_{l=1}^{L'} \log\langle \mathbf{x}^{(b,l)}_\theta, \mathbf{x}^{(b,l)}\rangle$ ▷ used only when $R^{(b)} = 1$, i.e., $t^{(b)} = 0$
  $\mathcal{L}^{(b)}_{\text{diff}} \leftarrow -\frac{1}{2}\,\text{SNR}'(t^{(b)})\,\|\mathbf{e}^{(b)}_\theta - \mathbf{e}^{(b)}\|^2$ ▷ used only when $R^{(b)} = 0$
  $\mathcal{L}^{(b)}_{\text{prior}} \leftarrow \frac{L'}{2}\left[d_e(\sigma_1^2 - 1 - \log\sigma_1^2) + \alpha_1^2\right]$ ▷ closed form of $\text{KL}\left(q(\mathbf{z}_1^{(b)} \mid \mathbf{x}^{(b)}) \,\|\, p(\mathbf{z}_1^{(b)})\right)$
  $\mathcal{L}^{(b)} \leftarrow \frac{1-R^{(b)}}{1-p_{\text{rec}}}\mathcal{L}^{(b)}_{\text{diff}} + \frac{R^{(b)}}{p_{\text{rec}}}\mathcal{L}^{(b)}_{\text{rec}} + \mathcal{L}^{(b)}_{\text{prior}}$ ▷ Horvitz–Thompson estimator (5)
end for
$\mathcal{L}_{\text{BD}} \leftarrow \sum_{b=1}^{B}\mathcal{L}^{(b)}$ ▷ block-diffusion NELBO estimator (3)
if $\lambda_{\text{susreg}} > 0$ **then**
  $\mathcal{L}_{\text{BD}} \leftarrow \mathcal{L}_{\text{BD}} + \lambda_{\text{susreg}}\,L\,s(\mathbf{E})$ ▷ SUSReg loss (16)
end if
$\mathcal{L} \leftarrow w_{\text{diff}}\,\mathcal{L}_{\text{BD}} + w_{\text{ar}}\,\mathcal{L}_{\text{ar}}(\mathbf{x})$ ▷ joint objective (Sec. 3.2.2)
$\mathcal{L}_{\text{var}} \leftarrow \sum_{b\,:\,R^{(b)}=0}(\mathcal{L}^{(b)}_{\text{diff}})^2$ ▷ Noise schedule interior shape learned to minimize MC variance
update θ with $\nabla_\theta\mathcal{L}$ ▷ backbone group, peak lr 10^{-5}
update $(\mathbf{E}, \gamma_0, \gamma_1)$ with $\nabla_{(\mathbf{E},\gamma_0,\gamma_1)}\mathcal{L}$ ▷ diffusion group, peak lr 10^{-2}, no weight decay
update ψ with $\nabla_\psi\mathcal{L}_{\text{var}}$ ▷ same as diffusion group

At training, all B blocks are denoised in a *single* forward pass: the offset block-causal mask of Fig. 2b makes block b's noisy latent attend to the clean prefix $\mathbf{x}^{(<b)}$ and bidirectionally within itself, so a per-block time $t^{(b)}$ and a per-block Horvitz–Thompson coin $R^{(b)}$ cost no extra compute. At inference, the conditional and unconditional branches are likewise packed into one forward as in Fig. 4a, so the guidance in (28) costs slightly more wall-clock time per step (Tab. 20) at batch size one.

We further clarify on the learning of the noise schedule, since its two parts are optimized for two different objectives. The endpoints γ_0, γ_1 enter $\mathcal{L}_{\text{diff}}$ through $\text{SNR}'(t)$ and (α_t, σ_t), so they receive an ordinary NELBO gradient and are trained to tighten the bound while the interior shape $\tilde{\gamma}_\psi$ does not. As noted in Sec. 2, the diffusion term is invariant to it in expectation, so $\nabla_\psi\mathcal{L}$ is zero on average and carries only noise. Following Kingma et al. (2021), ψ is instead trained to minimize the Monte-Carlo variance of the diffusion estimator, which for a mean-invariant estimator reduces to minimizing its

second moment $\mathcal{L}_{\text{var}}$ in Alg. 1. Both parts sit in the same separate optimizer group as the diffusion embedding, at peak learning rate 10^{-2} without weight decay (Sec. 3.2.2).

Algorithm 2 Block-wise reverse sampling of *Sigma* with CFG, ST, SC, and commit stride

Require: prompt $\mathbf{p}$; block size L'; commit stride $S \in \{1, \dots, L'\}$ ($S = L'$ recovers plain block decoding, App. G.5); total generation length T_{gen}; steps N with discretization $1 = t_0 > t_1 > \cdots > t_N = 0$ (App. G.6); guidance weight $w \geq 1$; score temperature $\tau \in (0, 1]$; sampler $\in \{\text{DDPM}, \text{DDIM}\}$

$\mathbf{c} \leftarrow \mathbf{p}$; populate the causal KV cache from $\mathbf{p}$; $b \leftarrow 0$
while $|\mathbf{r}| < T_{\text{gen}}$ **do** ▷ $|\mathbf{r}|$: number of tokens committed so far
  $b \leftarrow b + 1$
  $\mathbf{z}_{t_0}^{(b)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I}) \in \mathbb{R}^{L' \times d_e}$, $\mathbf{e}_{\text{sc}} \leftarrow \mathbf{0}$
  for $i = 1, \dots, N$ **do**
    $t \leftarrow t_{i-1}$, $s \leftarrow t_i$
    $(\mathbf{e}_{\text{c}}, \mathbf{e}_{\varnothing}) \leftarrow (\mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c}, \mathbf{e}_{\text{sc}}),\ \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \varnothing, \mathbf{0}))$ ▷ one packed forward, Fig. 4a
    $\mathbf{e}_{\text{sc}} \leftarrow \mathbf{e}_{\text{c}}$ ▷ SC reuses the *unsteered* conditional mean, Fig. 4b
    $\hat{\mathbf{e}}_\theta \leftarrow \frac{w}{\tau}\mathbf{e}_{\text{c}} + \frac{1-w}{\tau}\mathbf{e}_{\varnothing} + \left(1 - \frac{1}{\tau}\right)\frac{\mathbf{z}_t^{(b)}}{\alpha_t}$ ▷ Embedding mean prediction after CFG & ST, (28)
    if sampler is DDIM **then**
      $\hat{\boldsymbol{\epsilon}}_\theta \leftarrow \frac{1}{\sigma_t}(\mathbf{z}_t^{(b)} - \alpha_t \hat{\mathbf{e}}_\theta)$, $\mathbf{z}_s^{(b)} \leftarrow \alpha_s \hat{\mathbf{e}}_\theta + \sigma_s \hat{\boldsymbol{\epsilon}}_\theta$ ▷ (DDIM)
    else
      $\kappa \leftarrow 1 - \mathrm{e}^{\gamma(s) - \gamma(t)}$, $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
      $\mathbf{z}_s^{(b)} \leftarrow (1 - \kappa)\frac{\alpha_s}{\alpha_t}\mathbf{z}_t^{(b)} + \kappa\, \alpha_s \hat{\mathbf{e}}_\theta + \sqrt{\kappa(1 - \alpha_s^2)}\, \boldsymbol{\epsilon}$ ▷ (DDPM)
    end if
  end for
  $k_b \leftarrow \min(S,\ T_{\text{gen}} - |\mathbf{r}|)$ ▷ tokens to commit: the last window may commit fewer than S
  $\mathbf{r}^{(b,l)} \leftarrow \operatorname{argmax}_v \mathbf{x}_\theta^{l,v}(\mathbf{z}_{t_N}^{(b)}, t_N; \mathbf{c}, \mathbf{e}_{\text{sc}}),\ \ l \in [k_b]$ ▷ argmax decoding
  $\mathbf{c} \leftarrow (\mathbf{c}, \mathbf{r}^{(b)})$; append the KV of $\mathbf{r}^{(b)}$ to the causal cache; discard the tail ▷ the $L' - k_b$ tail positions are re-drawn as fresh noise in the next window
end while
return $\mathbf{r} = (\mathbf{r}^{(1)}, \dots, \mathbf{r}^{(b)})$

The outer loop of Alg. 2 runs $B := \lceil T_{\text{gen}}/S \rceil$ times, and each window costs $N + 1$ denoiser forward passes (N diffusion steps plus one decoding readout), for a total of $B(N + 1)$ forward passes. The effective NFE per committed token is thus $\approx (N + 1)/S$, see App. G.5 for a full compute-vs-quality sweep. Setting $w = 1$ disables CFG, $\tau = 1$ disables ST, and never updating $\mathbf{e}_{\text{sc}}$ (i.e., leaving it as $\mathbf{0}$) disables SC; each of these recovers a strictly weaker sampler, as shown in Fig. 10.

E Implementation Details and Additional Results for Pre-training

[Back to Sec. 3.2 in the main paper.]

Architecture. Following NLD (Fu et al., 2026), we adopt the transformer architecture of Ministral (Liu et al., 2026a):

- 3B: 26 layers, width 3072, FFN 9216, 32 query and 8 key/value heads (head dimension 128), SwiGLU, RMSNorm, YaRN-scaled rotary embeddings ($\theta = 10^6$), and untied embeddings. This adds up to 3.0B non-embedding and 3.8B total parameters.
- 8B: 34 layers, width 4096, FFN 14336, same 32 query and 8 key/value heads (head dimension 128) and same SwiGLU, RMSNorm, YaRN rotary embeddings and untied embeddings. This adds up to 7.4B non-embedding and 8.5B total parameters.

Optimization. For $\tilde{\gamma}$ and $\mathbf{E}$, we use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, no weight decay (rows of $\mathbf{E}$ unit-norm), and peak learning rate $\eta_{\text{peak}} = 1 \times 10^{-2}$; γ_0, γ_1 use the same setting but with weight decay 0.1 (Yang et al., 2026). For other parameters, we follow NLD and use AdamW with $\beta_1 = 0.9$, $\beta_2 = 0.95$, weight decay 0.1, and $\eta_{\text{peak}} = 1 \times 10^{-5}$. For all parameters, we use gradient clipping value 1.0. The learning rate schedule is Warmup-Stable-Decay over 150K steps: 1% linear warmup to η_{peak}, constant through step 120K, then exponential decay to 10% of η_{peak}.

Self-conditioning (SC). SC injects the model's own detached clean-embedding prediction into the projected noised input stream with a bias-free, zero-initialized linear layer mapping $\mathbb{R}^{d_e}$ to the hidden width $\mathbb{R}^d$: a first pass with $\mathbf{e}_{\text{sc}} = \mathbf{0}$, then a second pass with $\sqrt{d_e}\ \text{sg}(\mathbf{e}_{\text{sc}})$ through that linear layer.

Time conditioning. For diffusion stream, a single γ is sampled per block and shared across its tokens. We encode γ using fixed sinusoidal features with frequencies $\exp(\text{linspace}(-5, 5, 32))$, concatenating $[\sin, \cos]$ into 64 dimensions. A bias-free, zero-initialized linear layer projects this encoding to the hidden width, where it is added to the projected noised input.

Numerical precision. All parameters are stored in `BF16`, and the transformer also computes in `BF16` through the forward and backward pass, its linear logit heads included; gradient accumulation and the data-parallel all-reduce run in `FP32`. The small modules outside the transformer upcast their weights at use. Among these, most computes in `FP32`: the diffusion embeddings and their Gaussian corruption and up-projection, the sinusoidal γ features and their up-projection, and the SC up-projection. The noise schedule computes in `FP64`, covering the endpoints γ_0, γ_1, the minus log-SNR γ and its derivative, and the α_t, σ_t derived from them. These drop `FP64` at three points: γ before the sinusoidal features, α_t, σ_t at the corruption, and the variance rescaling before the up-projection for $\mathbf{z}_t$. These modules incur negligible FLOPs, so the extra precision is a cheap precaution followed from Gulrajani & Hashimoto (2023); Yang et al. (2026). The optimizer keeps `FP32` master weights and Adam moments, sharded across data-parallel ranks (distributed optimizer).

AR regularization. We searched over $w_{\text{diff}} \in \{0.3, 1.0\}$. Interestingly, when $w_{\text{diff}} = 1.0$, adding the AR regularization even improved diffusion PPL (Fig. 3c). However, we found that switching to $w_{\text{diff}} = 0.3$ at 200B tokens offers the best convergence for both losses (Fig. 3c); this is our flagship 3B *Sigma* model which we evaluate in subsequent sections. We also trained our 8B *Sigma* model using this configuration.

E.1 Sketched Uniform Spherical Regularization (SUSReg)

[Back to Sec. 5.2 in the main paper.]

To avoid the $V = 131072$ embedding vectors in $d_e = 16$ dimensional unit sphere being too tightly clustered for accurate decoding (as observed in their low-rank structure in Yang et al. (2026)), we propose to add an optional regularization term based on SUSReg (Nicollier et al., 2026) to the embeddings, which regularizes their distribution $p_{\mathbf{E}} := \frac{1}{V}\sum_{v=1}^{V} \delta_{\mathbf{e}_v}$ toward $\text{Unif}(\mathbb{S}^{d_e-1})$, the uniform distribution on the unit sphere in $\mathbb{R}^{d_e}$.

For M directions $\mathbf{a}_m \sim \mathrm{Unif}(\mathbb{S}^{d_e-1})$, we compare the empirical characteristic function of each one-dimensional projection,

$$\phi_{\mathbf{a}_m}(t) = \frac{1}{V}\sum_v \mathrm{e}^{\mathrm{i}t\langle \mathbf{a}_m, \mathbf{e}_v\rangle},$$

to the closed-form target characteristic function of one-dimensional projection of $\mathrm{Unif}(\mathbb{S}^{d_e-1})$,

$$\phi_{d_e}(t) = \int_{-1}^{1} \cos(tx)\rho_{d_e}(x)\,\mathrm{d}x, \quad \rho_{d_e}(x) = \frac{\Gamma(\frac{d_e}{2})}{\sqrt{\pi}\Gamma(\frac{d_e-1}{2})}(1-x^2)^{\frac{d_e-3}{2}}\mathbf{1}_{|x|\le 1},$$

under an Epps–Pulley Gaussian-weighted L^2 metric:

$$s(\mathbf{E}) := \frac{V}{M}\sum_{m=1}^{M}\int_{-t_{\max}}^{t_{\max}} \left|\phi_{\mathbf{a}_m}(t) - \phi_{d_e}(t)\right|^2 \mathrm{e}^{-t^2/2}\,\mathrm{d}t. \tag{16}$$

The multiplier V ensures that $\mathbb{E}_{\mathbf{a}_{1:M}}[s(\mathbf{E})]$, as $V \to \infty$, converges to a limit that is independent of V and only depends on d_e. We add $\lambda_{\text{susreg}} s(\mathbf{E})$ *per token* to the block NELBO $\mathcal{L}_{\text{BD}}(\mathbf{x})$ (3) on the diffusion arm before combining with the AR auxiliary, so the SUSReg-to-ELBO gradient ratio on $\mathbf{E}$ is exactly λ_{susreg} regardless of the joint weights $w_{\text{diff}}, w_{\text{ar}}$ (Alg. 1). At the values used in our 3B pre-training run, $\lambda_{\text{susreg}} = 0.005$, $M = 512$ directions, and the t-integral discretized on $P = 17$ trapezoidal points over $[-5, 5]$; the resulting SUSReg contribution to the reported loss is $w_{\text{diff}}\lambda_{\text{susreg}} s(\mathbf{E})$. We remark that the cost for computing $s(\mathbf{E})$, $O(VM(d_e + P))$, is negligible compared with the cost of transformer forward pass.

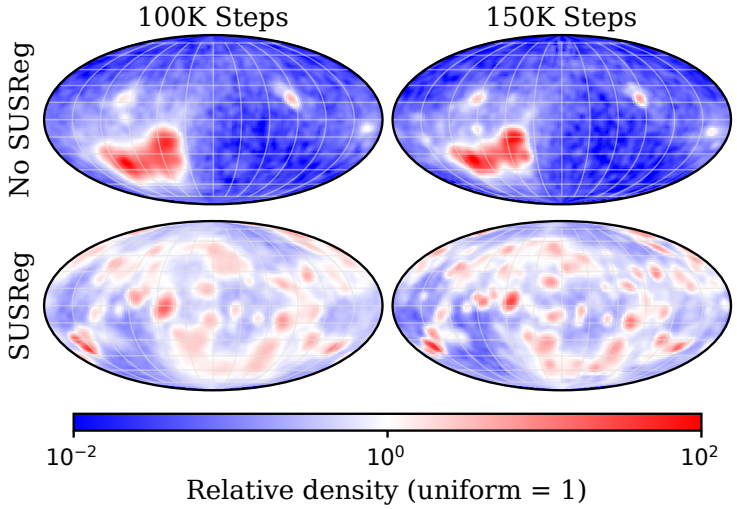


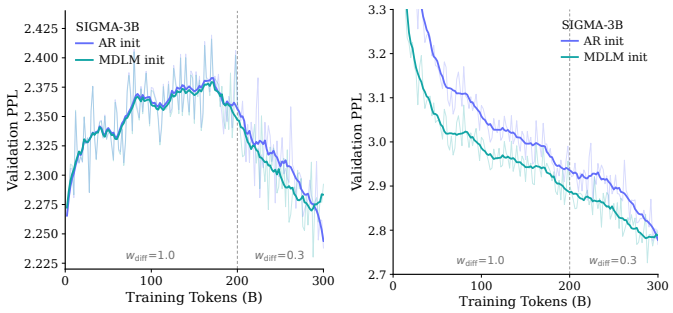
Figure 8: Density plot of a random $\mathbb{S}^{15} \to \mathbb{S}^2$ Mollweide projection of all learned embeddings living on $\mathbb{S}^{15}$.

E.2 Ablation Study: MDLM Initialization for Pre-training

[Back to Sec. 3.2 in the main paper.]

Sigma-3B featured in our main results is warm-started from an AR checkpoint ("AR initialization" in Tab. 1) and then trained for 300B tokens under the joint diffusion + AR objective (Sec. 3.2.2). We ablate that choice against a masked diffusion language model (MDLM) warm start: identical 300B-token recipe, run instead from the released Nemotron-Labs-Diffusion-3B checkpoint of Fu et al. (2026), which is our main baseline in Tab. 1. All other knobs are held fixed.

Evaluation setup. In Tab. 4, we evaluate the performance of the two trained *Sigma*-3B models at the two CFG-ST configurations of Tab. 1, both with self-conditioning enabled: DDPM (256 steps, EDM ρ=7 schedule, $w = 2$, $\tau = 0.1$) and DDIM (32 steps, uniform-γ schedule, $w = 3$, $\tau = 0.3$). Numbers are mean $\pm$ two standard errors over 10 sampling lanes, forming a 95% confidence interval of the mean.



(a) Validation Loss (AR) (b) Validation Loss (Diffusion)

Figure 9: **Pre-training *Sigma*-3B** with block size 32 and AR regularization from AR initialization (AR init) and from MDLM initialization (MDLM init). PPL for AR stream (a) and diffusion stream (b) are computed using $\mathcal{L}_{\text{ar}}(\mathbf{x})$ ((6)) and $\mathcal{L}_{\text{BD}}(\mathbf{x})$ (3), respectively. MDLM initialization has an initial advantage that dies off during learning rate decay.

Table 4: **Diffusion initialization ablation for *Sigma*-3B pre-training.** Same six generative reasoning tasks as Tab. 1. The *AR init* rows are copied verbatim from Tab. 1. The *MDLM init* rows use the same 300B-token training recipe but warm-started from the Nemotron-Labs-Diffusion-3B checkpoint of Fu et al. (2026). Δ = MDLM − AR. All metrics are mean ± two standard errors over 10 sampling lanes. **Avg.** is the unweighted mean over the six tasks.

Sampler	Init	GSM8K	Minerva Math	HumanEval	HumanEval+	MBPP	MBPP+	Avg.
DDPM (256 steps, w=2, τ=0.1)	AR	84.50 ± 0.31	64.60 ± 0.28	59.45 ± 1.30	53.84 ± 1.29	52.32 ± 0.73	71.32 ± 0.91	64.34
	MDLM	84.45 ± 0.35	65.37 ± 0.12	57.13 ± 1.05	52.87 ± 1.35	55.56 ± 0.65	72.62 ± 0.56	64.67
	Δ	-0.05	+0.77	-2.32	-0.97	+3.24	+1.30	+0.33
DDIM (32 steps, w=3, τ=0.3)	AR	81.91 ± 0.32	62.76 ± 0.18	56.59 ± 1.31	50.67 ± 1.00	47.26 ± 0.46	63.49 ± 0.77	60.45
	MDLM	83.71 ± 0.17	64.23 ± 0.28	52.99 ± 0.59	50.18 ± 0.68	51.24 ± 0.51	67.46 ± 0.58	61.64
	Δ	+1.80	+1.47	-3.60	-0.49	+3.98	+3.97	+1.19

Results. MDLM initialization has a small but consistent win on average at both configurations, but the sign of the change is task-dependent rather than uniform. MBPP and MBPP+ strongly favor MDLM initialization, while HumanEval and HumanEval+ show opposite trends.

F Implementation Details and Additional Results for Likelihood-based Benchmarks

[Back to Sec. 3.2 in the main paper.]

Table 5: **Likelihood-based multiple-choice benchmarks (3B Base).** Accuracy (%) per task. *AR mode* ranks candidates with the AR likelihood from the AR stream; *diffusion mode* ranks them by the variational bound of the diffusion stream: (17) for *Sigma*, the count-based masked diffusion NELBO (19) for Nemotron-Labs-Diffusion. Underline: the best within each mode. **Bold**: the best value across all methods.

Method	ARC-e	ARC-c	HellaSwag	PIQA	WinoGrande	MMLU	Avg.
Chance	*25.00*	*25.00*	*25.00*	*50.00*	*50.00*	*25.00*	*33.33*
AR initialization	**80.39**	**57.42**	**71.74**	78.18	**70.64**	**70.74**	**71.52**
AR mode							
Nemotron-Labs-Diffusion-3B	78.70	55.46	70.39	**78.24**	70.32	69.61	70.46
***Sigma*-3B (Ours)**	79.29	55.29	69.98	77.64	69.46	68.50	70.03
Diffusion mode							
Nemotron-Labs-Diffusion-3B	78.07	52.99	64.90	73.78	68.75	69.70	68.03
***Sigma*-3B (Ours)**	78.96	54.61	65.64	73.61	69.93	67.66	68.40

Table 6: **Likelihood-based multiple-choice benchmarks (8B Base).** Accuracy (%) per task. *AR mode* ranks candidates with the AR likelihood from the AR stream; *diffusion mode* ranks them by the variational bound of the diffusion stream: (17) for *Sigma*, the count-based masked diffusion NELBO (19) for Nemotron-Labs-Diffusion. Underline: the best within each mode. **Bold**: the best value across all methods.

Method	ARC-e	ARC-c	HellaSwag	PIQA	WinoGrande	MMLU	Avg.
Chance	*25.00*	*25.00*	*25.00*	*50.00*	*50.00*	*25.00*	*33.33*
AR initialization	**83.80**	**61.52**	**77.05**	80.03	**76.48**	**76.84**	**75.95**
AR mode							
Nemotron-Labs-Diffusion-8B	82.32	59.73	76.09	79.76	75.93	74.88	74.78
***Sigma*-8B (Ours)**	82.28	60.67	75.70	**80.41**	**76.48**	74.58	75.02
Diffusion mode							
Nemotron-Labs-Diffusion-8B	80.81	58.62	71.47	77.31	73.88	76.01	73.02
***Sigma*-8B (Ours)**	80.85	58.19	69.03	76.44	75.22	73.96	72.28

Benchmarks. We evaluate on ARC-e/c (Clark et al., 2018), HellaSwag (Zellers et al., 2019), PIQA (Bisk et al., 2020), WinoGrande (Sakaguchi et al., 2020), and MMLU (Hendrycks et al., 2021). Per Open LLM Leaderboard convention we report `acc_norm` (normalized by character length) on the four length-varying 0-shot tasks and plain `acc` on WinoGrande and MMLU. These benchmarks score a candidate response $\mathbf{r}$ against a prompt $\mathbf{p}$ by comparing $\log p_\theta(\mathbf{r} \mid \mathbf{p})$ across candidates.

Scoring our continuous method. *Sigma* does not give likelihood in closed form, but a surrogate based on the NELBO: taking the prompt as the clean prefix and the candidate as the block to be denoised, and then applying (4) verbatim with $\mathbf{p}$ as $\mathbf{x}^{(<b)}$ and $\mathbf{r}$ as $\mathbf{x}^{(b)}$:

$$-\log p_\theta(\mathbf{r} \mid \mathbf{p}) \le \mathcal{L}_{\text{NELBO}}(\mathbf{r};\mathbf{p}) = \mathrm{KL}(q(\mathbf{z}_1 \mid \mathbf{r}) \,\|\, p(\mathbf{z}_1)) + \mathbb{E}_{q(\mathbf{z}_0|\mathbf{r})}\left[\textstyle\sum_{l=1}^{|\mathbf{r}|} - \log\langle \mathbf{x}_\theta^l(\mathbf{z}_0, 0; \mathbf{p}), \mathbf{r}^l\rangle\right] - \tfrac{1}{2}\, \mathbb{E}_{t\sim\mathrm{Unif}[0,1]}\, \mathbb{E}_{q(\mathbf{z}_t|\mathbf{r})}[\mathrm{SNR}'(t)\|\mathbf{e}_\theta(\mathbf{z}_t, t; \mathbf{p}) - \mathbf{e}\|^2]. \tag{17}$$

In practice, we evaluate (17) with 128 stratified samples $t_k = \frac{k+u}{128}$, $k = 0, \ldots, 127$, $u \sim \mathrm{Unif}[0, 1]$ and 8 reconstruction samples.

Scoring the masked baseline via count-based NELBO. Here we use the notation for masked diffusion defined in App. C. Let $\mathbf{p}$ be the prompt and $\mathbf{r}$ be a candidate response. We generate the masked states by the *mask set*, instead of randomly masking each position with probability t: for $M \subseteq \{1, \ldots, L_\mathbf{r}\}$, where $L_\mathbf{r} := |\mathbf{r}|$, we obtain $\mathbf{z}_M$ from masking indices in M of $\mathbf{r}$, and denote $C(M) := \sum_{l\in M} - \log\langle \mathbf{x}_\theta^l(\mathbf{z}_M; \mathbf{p}), \mathbf{r}^l\rangle$ their cross entropy. Under $\alpha_t = 1 - t$, (9) gives the continuous-time bound; integrating t out yields an average over the *number* of masked tokens (Ou

et al., 2025; Nie et al., 2025b):

$$-\log p_\theta(\mathbf{r} \mid \mathbf{p}) \;\le\; \mathcal{L}_{\text{MDLM}}(\mathbf{r};\mathbf{p}) \;=\; \mathbb{E}_{t\sim\text{Unif}[0,1]}\left[\frac{1}{t}\,\mathbb{E}_{M\sim\text{Bern}(t)^{\otimes L_\mathbf{r}}}\,C(M)\right] \tag{18}$$

$$=\; \mathbb{E}_{n\sim\text{Unif}\{1,\ldots,L_\mathbf{r}\}}\left[\frac{L_\mathbf{r}}{n}\,\mathbb{E}_{M\sim\text{Unif}\binom{[L_\mathbf{r}]}{n}}\,C(M)\right], \tag{19}$$

where $\binom{[L_\mathbf{r}]}{n}$ means the set of all n-element subsets of $[L_\mathbf{r}] := \{1, 2, ..., L_\mathbf{r}\}$. (19) is a Rao–Blackwellized low-variance estimator of (18), manifested in the following three major differences: *(i)* The $\frac{1}{t}$ weight becomes $\frac{L_\mathbf{r}}{n} \le L_\mathbf{r}$, so rare small-$t$ draws that leave one or two masked tokens would not cause the variance of the estimator to blow up. *(ii)* Bernoulli masking leaves $\mathbf{r}$ wholly unmasked with probability $(1-t)^{L_\mathbf{r}}$, which may be common for short responses and wastes a forward pass, while the $n \ge 1$ constraint avoids this. *(iii)* As n is finite, we can similarly use stratified sampling as the t-grid above, spreading counts evenly over $\{1, \ldots, L_\mathbf{r}\}$ from one uniform offset, so every batch covers all masking rates.

In practice, we use (19) with 128 low-discrepancy draws n_k and standard random sampling for M_k for Nemotron-Labs-Diffusion (Fu et al., 2026) in Tabs. 5 and 6. On MMLU candidates are single tokens, so $L_\mathbf{r} = n = 1$ and (19) is exact after one forward pass, while (18) would still be averaging $\frac{1}{t}$. We found (19) to work better than (18) in early experiments.

Length generalization. Diffusion training denoises blocks of exactly L' tokens from packed sequences of length L, whereas candidates are typically short and occasionally longer than L'. For each $(\mathbf{p}, \mathbf{r})$ pair, we apply causal attention over $\mathbf{p}$ and bidirectional attention over $\mathbf{r}$, assuming that the bidirectional attention can generalize to response length other than L'. The attention map follows the conditional branch in Fig. 4a.

G Implementation Details and Additional Results for Inference

[Back to Secs. 4 and 5 in the main paper.]

Table 7: **Low-NFE performance of *Sigma*-3B (base model; block size 32) with DDIM and SUSReg embedding regularization** on math reasoning and coding benchmarks. Accuracy (%) is Pass@1. The *Sigma* rows are reported as mean (over 10 seeds) ± two standard errors. All results use the uniform-γ schedule. **Bold** marks the best value per task; underline marks the better of *Sigma*-3B and + SUSReg within each step budget. For DDIM 256 steps, guidance settings were tuned according to the best 6-task average accuracy. For other runs (except PDD), settings were selected by tuning on 3-task average using GSM8K, HumanEval and MBPP. Within this table, all DDIM runs use SC; PDD does not use SC due to limitations discussed in Sec. 5.3.

	Math		Coding					
Method (Diffusion Mode)	GSM8K	Minerva Math	HumanEval	HumanEval+	MBPP	MBPP+	Avg.	Δ Avg.
Nemotron-Labs-Diffusion-3B	83.40	63.96	56.10	53.05	**57.60**	**74.34**	**64.74**	
DDIM 256 steps / block								
Sigma-3B (w=3, τ=0.2)	**84.18** ± 0.18	**64.00** ± 0.14	**60.24** ± 0.84	**54.02** ± 0.80	51.02 ± 0.42	69.79 ± 0.16	63.88	
+ SUSReg (w=3, τ=0.2)	84.12 ± 0.14	63.68 ± 0.18	57.32 ± 0.67	52.20 ± 0.74	56.08 ± 0.39	72.65 ± 0.31	64.34	+0.46
DDIM 32 steps / block								
Sigma-3B (w=3, τ=0.3)	81.91 ± 0.32	62.76 ± 0.18	56.59 ± 1.30	50.67 ± 1.00	47.06 ± 0.52	63.49 ± 0.77	60.41	
+ SUSReg (w=4, τ=0.4)	83.17 ± 0.30	63.28 ± 0.15	55.24 ± 1.36	50.91 ± 0.99	51.64 ± 0.57	66.88 ± 0.29	61.85	+1.44
DDIM 16 steps / block								
Sigma-3B (w=3, τ=0.5)	78.08 ± 0.34	60.16 ± 0.21	47.50 ± 1.14	43.84 ± 0.94	41.84 ± 1.00	59.58 ± 0.98	55.17	
+ SUSReg (w=3, τ=0.4)	80.85 ± 0.31	61.73 ± 0.24	53.60 ± 0.80	48.84 ± 0.90	45.22 ± 0.57	60.79 ± 0.66	58.50	**+3.33**
DDIM 8 steps / block								
Sigma-3B (w=3, τ=0.8)	66.65 ± 0.40	45.13 ± 0.23	28.54 ± 1.53	25.67 ± 1.35	23.58 ± 0.65	32.80 ± 0.83	37.06	
+ SUSReg (w=3, τ=0.7)	70.88 ± 0.43	52.12 ± 0.40	34.45 ± 1.15	31.52 ± 1.29	27.98 ± 0.66	39.63 ± 0.99	42.76	**+5.70**
DDIM 4 steps / block								
Sigma-3B (w=2, τ=0.9)	11.64 ± 0.57	4.94 ± 0.11	9.88 ± 0.69	9.15 ± 0.79	4.24 ± 0.53	6.59 ± 0.52	7.74	
+ SUSReg (w=2, τ=0.9)	26.04 ± 0.64	9.64 ± 0.20	11.59 ± 1.22	10.85 ± 1.09	4.32 ± 0.62	8.33 ± 0.84	11.80	**+4.06**
PDD 8 steps / block								
Sigma-3B (w=3, τ=0.3)	69.83 ± 0.16	49.62 ± 0.16	38.41 ± 0.40	35.37 ± 0.41	24.60 ± 0.39	34.92 ± 0.31	42.12	
+ SUSReg (w=3, τ=0.3)	72.48 ± 0.15	51.82 ± 0.16	43.90 ± 0.40	39.63 ± 0.34	29.00 ± 0.28	43.92 ± 0.61	46.79	**+4.67**
PDD 4 steps / block								
Sigma-3B (w=4, τ=0.4)	65.58 ± 0.26	41.16 ± 0.11	26.83 ± 0.56	25.00 ± 0.61	12.80 ± 0.35	19.05 ± 0.58	31.74	
+ SUSReg (w=4, τ=0.4)	67.60 ± 0.34	45.18 ± 0.13	31.10 ± 0.44	28.05 ± 0.42	16.20 ± 0.34	24.60 ± 0.35	35.45	**+3.71**

G.1 Reasoning Benchmark Setup

[Back to Secs. 4 and 5 in the main paper.]

We describe the harness, scoring pipeline, and sampler that produce Tabs. 1, 7 and 8. The scoring pipeline is consistent for all base models and generation modes.

Benchmarks and few-shot budgets. We use six tasks from `lm-evaluation-harness` following Fu et al. (2026): GSM8K (8-shot, 1319 questions) (Cobbe et al., 2021), Minerva Math (4-shot, 5000 questions[2]) (Lewkowycz et al., 2022), HumanEval and HumanEval+ (0-shot, 164 questions) (Chen et al., 2021), and MBPP and MBPP+ (3-shot, 500 and 378 questions, respectively) (Austin et al., 2021b). The per-task generation budget is 256 tokens for GSM8K and 512 tokens for the other five tasks.

Scoring. GSM8K is scored by exact-match on the numerical answer, extracted from the model's response by a permissive regex that keeps the last number appearing in the completion (stripping separators and trailing punctuation). Minerva Math is scored by symbolic equivalence after canonical-

[2]Number of questions in each sub-domain: algebra 1187, counting/prob 474, geometry 479, intermediate algebra 903, number theory 540, prealgebra 871, precalc 546.

Table 8: **Performance of *Sigma*-8B (base model; block size: 32)** on math reasoning and coding benchmarks. Accuracy (%) is pass@1. The *Sigma* diffusion rows are reported as mean (over 10 seeds) ± two standard errors. **Bold**: the best value across all methods; underline: the best value within each mode.

	Math		Coding				
Method	GSM8K	Minerva Math	HumanEval	HumanEval+	MBPP	MBPP+	Avg.
AR initialization	85.90	**69.18**	59.15	53.05	68.20	**83.33**	69.80
AR mode							
Llama-3.1-8B	54.06	18.22	35.37	28.66	48.80	61.90	41.17
Ministral3-8B	80.21	44.58	42.68	38.41	61.60	76.98	57.41
Qwen3-8B	86.73	52.94	**64.63**	**56.71**	**69.40**	83.07	68.91
Nemotron-Labs-Diffusion-8B	88.25	66.00	60.37	53.05	68.20	82.54	69.73
***Sigma*-8B (Ours)**	86.88	66.72	51.83	48.78	64.00	80.95	66.53
Diffusion mode							
LLaDA-8B	70.96	27.30	32.32	27.44	40.80	51.85	41.78
Dream-7B	77.18	39.60	54.88	49.39	56.80	74.60	58.74
Nemotron-Labs-Diffusion-8B	87.26	65.16	62.80	**57.32**	67.00	81.75	**70.22**
***Sigma*-8B (Ours)**							
DDPM (256 steps, $w = 2, \tau = 0.1$)	**89.07** ± 0.28	67.47 ± 0.20	61.40 ± 1.73	**57.44** ± 1.62	64.78 ± 0.54	77.83 ± 0.72	69.67
DDIM (32 steps, $w = 3, \tau = 0.3$)	87.78 ± 0.33	67.13 ± 0.15	58.54 ± 1.04	53.17 ± 1.27	59.54 ± 0.57	74.15 ± 0.43	66.72

izing boxed and inline expressions. HumanEval and HumanEval+ report post-processed pass@1: the completion is stripped out of the Markdown code fence, an AST-based longest-valid-code extraction is applied, and the resulting program is executed against the task test cases in the container. MBPP and MBPP+ report the harness pass@1 directly without this post-processing step, matching the convention of prior work.

Decoding configuration. Unless overridden in the corresponding row of Tab. 1, all *Sigma* diffusion-mode runs use block size $L' = 32$, `BF16` model weights at inference time, argmax readout at the final diffusion step, and operation of SC follows Fig. 4b. The DDPM-256 row uses the ancestral sampler of (DDPM) on the EDM $\rho = 7$ schedule (App. G.6); the DDIM-32 row uses the deterministic sampler of (DDIM) on the uniform-γ schedule; the row-specific guidance strength w and score temperature τ appear in the table caption. Commit stride is the block size unless otherwise specified.

Seed error bars. Each *Sigma* diffusion row aggregates 10 independent sampling lanes as follows. For question i, we independently draw 10 responses under the same decoder with 10 different seeds. Let $s_{i,j}$ be the score of the j-th response to question i. We first average over questions to obtain 10 per-lane scores $s_j = \mathrm{mean}_i(s_{i,j})$ for $j = 1, \ldots, 10$, and then obtain $\bar{s} = \mathrm{mean}_j(s_j)$ together with the across-lane standard deviation $\hat{\sigma} = \mathrm{std}_j(s_j)$. The half-widths in Tab. 1 are $\frac{2}{\sqrt{10}}\hat{\sigma}$, an approximate 95% confidence interval for the mean.

Baseline decoders. The AR-mode for NLD and *Sigma* row uses greedy argmax at each step; NLD's diffusion-mode row uses the confidence-threshold greedy sampler of Fu et al. (2026) (block size $L' = 32$; argmax readout at every position whose top-1 probability exceeds a threshold; we report the better of thresholds 0.8 and 0.9 per task). For LLaDA-8B and Dream-7B we quote each model's best diffusion-mode number without the confidence-threshold rule: at each denoising step the single position with the highest top-1 confidence is unmasked. We refer readers to App. C for details of inference algorithms for masked diffusion models.

G.2 Sweeping Classifier-Free Guidance and Score Temperature

[Back to Sec. 4.2 or Sec. 4.3 in the main paper.]

In Figs. 10 and 11, we present the full tables for CFG and ST sweep for our *Sigma*-3B models trained with and without AR regularization, respectively.

G.3 Ablation Study: Steering in Logit Space Instead of Embedding Space

[Back to Sec. 4.2 in the main paper.]

Both steering knobs introduced in Sec. 4.2 act on the predicted clean embedding $\mathbf{e}_\theta \in \mathbb{R}^{L' \times d_e}$, equivalently on the score or velocity. Recall that our denoiser leverages *categorical reparameterization*, i.e.,

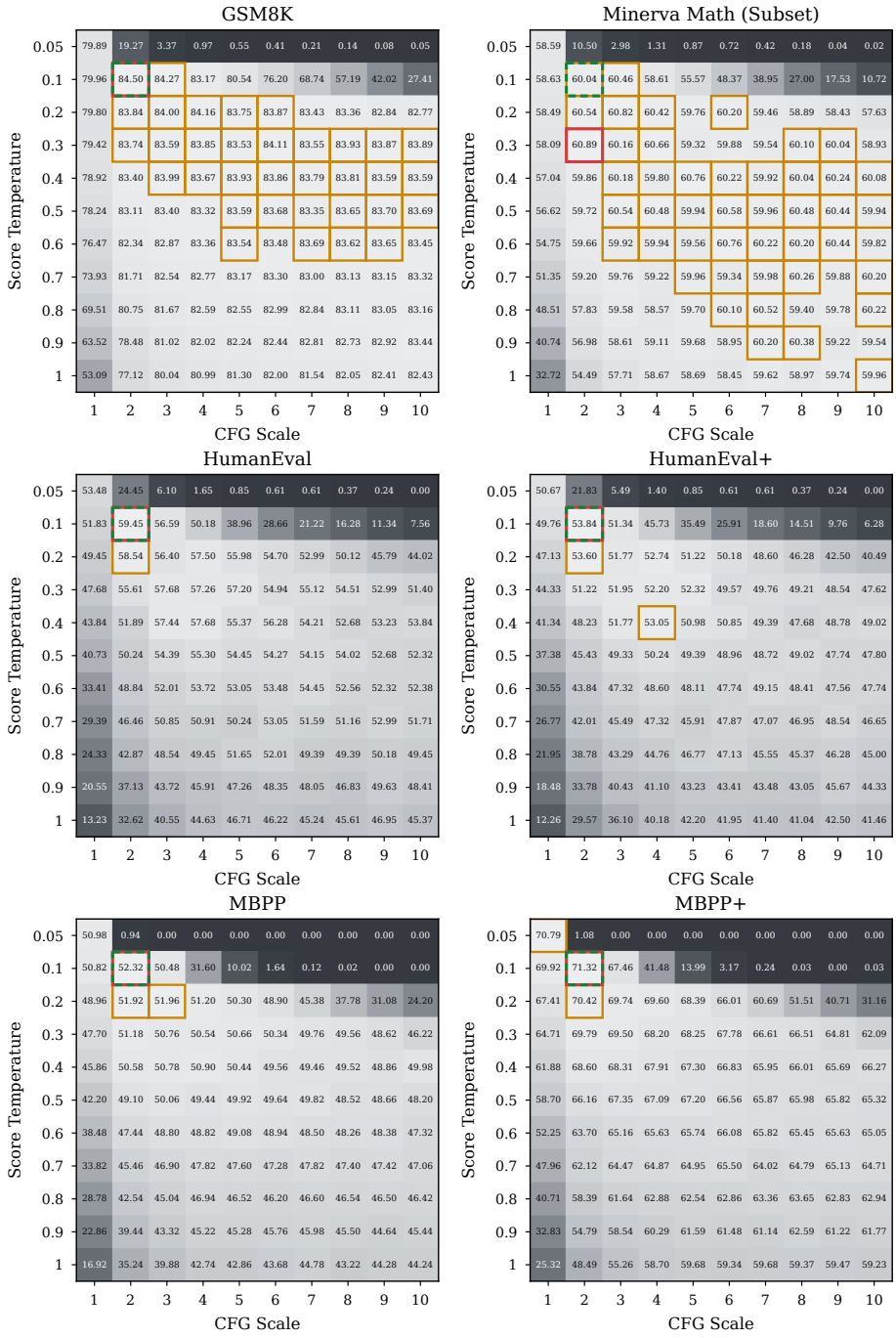


Figure 10: Sweeping CFG and ST for *Sigma*-3B trained with AR regularization. Accuracies are averaged over 10 random seeds. Red: per-task best; orange: within 1 point of per-task best; green: best average across 6 tasks. **For this model, per-task best and best average align on 5 out of 6 tasks.** In the exception of Minerva Math (subset; random 10% from each subtask), per-task best is still within 1 point from best average. We report best average (CFG $= 2$, ST $= 0.1$) in Fig. 5 and Tab. 1.

$\mathbf{e}_\theta = \mathrm{softmax}(\boldsymbol{\ell}_\theta)\mathbf{E} = \mathbf{x}_\theta\mathbf{E}$, where $\boldsymbol{\ell}_\theta \in \mathbb{R}^{L' \times V}$ are the predicted logits, and $\mathrm{softmax}$ operates in the vocabulary dimension. Thus, a natural alternative is to apply the same operations to $\boldsymbol{\ell}_\theta$ before the $\mathrm{softmax}$. This is the form of guidance used by discrete dLMs, which does not have embedding-space prediction to steer (Nie et al., 2025b; Li et al., 2025; Schiff et al., 2025) (App. A).

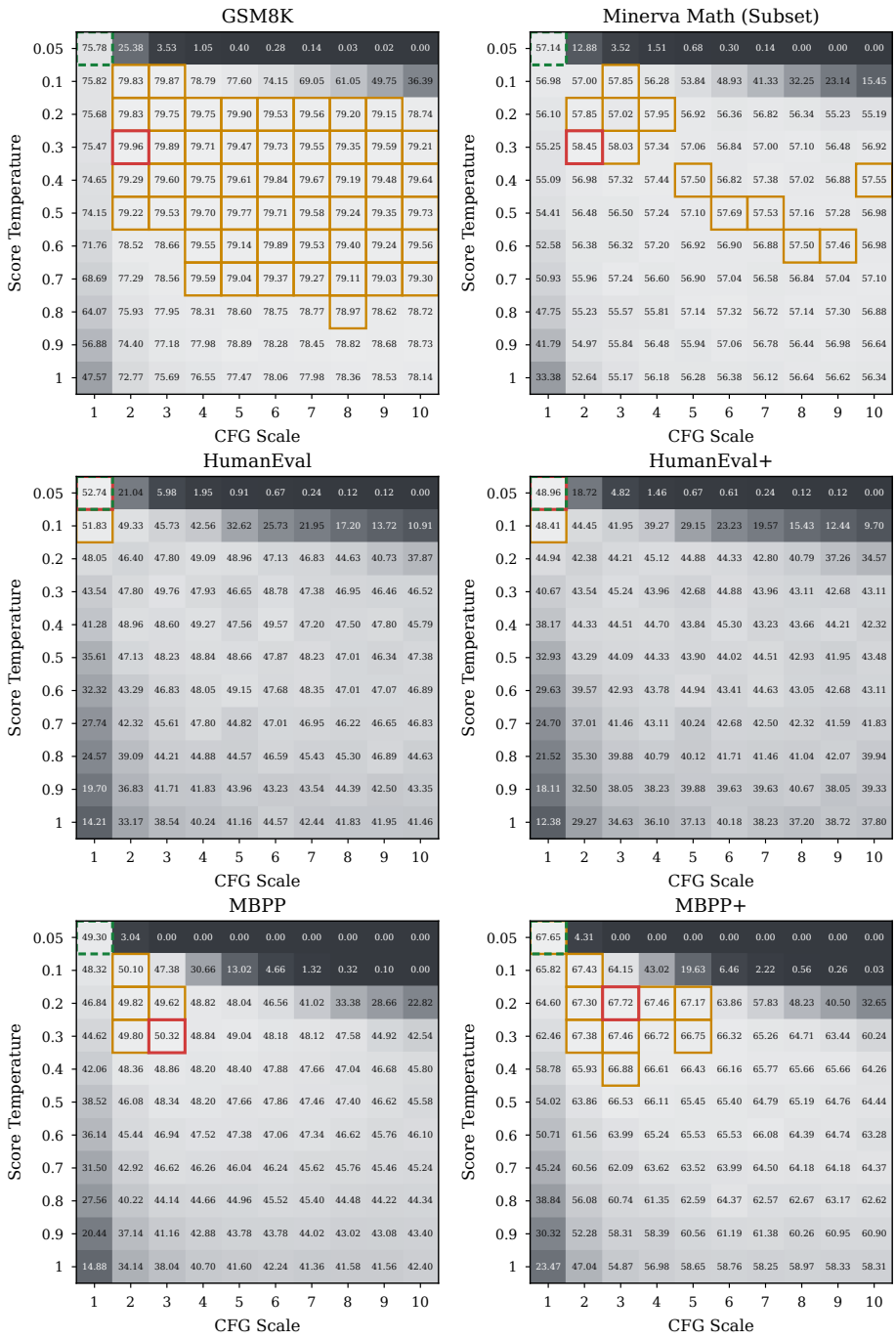


Figure 11: Sweeping CFG and ST for *Sigma*-3B trained *without* AR regularization. Accuracies are averaged over 10 random seeds. Red: per-task best; orange: within 1 point of per-task best; green: best average across 6 tasks. Both per-task best and best average are worse than *Sigma*-3B trained AR regularization (Fig. 10).

Logit-space CFG (LCFG). Reusing the notations in App. G.4, with the same fully unconditional bad prediction, we do softmax on the extrapolated logits:

$$\mathbf{e}_\theta^{w,\text{logit}} := \text{softmax}(\boldsymbol{\ell}_\theta^{\varnothing,\mathbf{0}} + w\,[\boldsymbol{\ell}_\theta^{+} - \boldsymbol{\ell}_\theta^{\varnothing,\mathbf{0}}])\mathbf{E}, \qquad w \geq 1,$$

against our default $\mathbf{e}_\theta^{w} = \mathbf{e}_\theta^{\varnothing,\mathbf{0}} + w[\mathbf{e}_\theta^{+} - \mathbf{e}_\theta^{\varnothing,\mathbf{0}}]$ of (CFG), which extrapolates after the softmax. The two coincide at $w = 1$ (i.e., no guidance).

Logit temperature (LT). This is the analogue of ST in logit space, which rescales the conditional logits themselves:

$$\mathbf{e}_\theta^T := \mathrm{softmax}\left(\frac{\boldsymbol{\ell}_\theta^+}{T}\right)\mathbf{E}, \qquad T \in [0, 1],$$

where $T = 1$ recovers the conditional mean embedding prediction (i.e., no guidance), and smaller T sharpens. The $T \to 0$ limit is the maximum a posteriori (MAP) denoiser $\mathbf{e}_\theta^0 = \mathbf{E}[\mathrm{argmax}\, \boldsymbol{\ell}_\theta^+]$, which snaps the prediction onto a single vocabulary embedding.

We remark that both operations change the trajectory only; tokens are still read out as the argmax of the conditional logits at the final step $t = 0$, exactly as in Sec. 4.1 and Alg. 2. Also, LCFG does not cost extra compute compared with the embedding-space counterpart, since we need to get both conditional and unconditional logit predictions.

Experimental setup. We use the setting identical to the default in App. G.4. We use the same *Sigma*-3B checkpoint with block length 32, the full GSM8K and HumanEval evaluation sets, and generate 10 responses per problem. DDIM uses 64 steps/block on a uniform-γ schedule with $\tau = 0.3$, and DDPM uses 256 steps/block on the EDM $\rho = 7$ schedule and $\tau = 1$.

Table 9: Logit-space versus embedding-space CFG. GSM8K is accuracy (%) and HumanEval is execution-based pass@1 (%). Entries are mean ± two standard errors over the 10 generations per prompt, an approximate 95% confidence interval for the mean, and **bold** marks the best entry in each row. The last column is the best value over the *full* sweep of Tabs. 12 and 13, which continues past $w = 5$ for the embedding rows. The logit rows were swept only to $w = 5$ as they already peak at $w = 2$ and fall monotonically after that.

Setting	Guidance space	$w = 1$	$w = 2$	$w = 3$	$w = 4$	$w = 5$	Tuned best
GSM8K, DDIM 64 steps, $\tau = 0.3$	Logit	68.13 ± 0.38	**83.49** ± 0.22	81.55 ± 0.41	77.08 ± 0.41	67.78 ± 0.53	**83.49** ± 0.22 ($w = 2$)
	Embedding (ours)	68.13 ± 0.38	82.52 ± 0.21	82.84 ± 0.28	83.04 ± 0.20	**83.19** ± 0.18	83.19 ± 0.18 ($w = 5$)
GSM8K, DDPM 256 steps, $\tau = 1$	Logit	53.09 ± 0.85	**76.76** ± 0.50	75.16 ± 0.79	69.48 ± 0.59	60.77 ± 0.99	76.76 ± 0.50 ($w = 2$)
	Embedding (ours)	53.09 ± 0.85	77.12 ± 0.66	80.04 ± 0.39	80.99 ± 0.48	**81.30** ± 0.41	**82.43** ± 0.50 ($w = 10$)
HumanEval, DDIM 64 steps, $\tau = 0.3$	Logit	42.93 ± 0.66	**53.78** ± 1.09	39.15 ± 1.13	20.67 ± 0.84	12.68 ± 0.47	53.78 ± 1.09 ($w = 2$)
	Embedding (ours)	42.93 ± 0.66	53.96 ± 0.95	56.04 ± 1.16	**56.22** ± 0.83	54.09 ± 1.22	**56.22** ± 0.83 ($w = 4$)
HumanEval, DDPM 256 steps, $\tau = 1$	Logit	13.23 ± 1.50	**29.70** ± 2.69	22.68 ± 1.36	14.51 ± 2.14	9.76 ± 0.48	29.70 ± 2.69 ($w = 2$)
	Embedding (ours)	13.23 ± 1.50	32.62 ± 1.98	40.55 ± 1.79	44.63 ± 1.07	**46.71** ± 1.82	**46.95** ± 1.73 ($w = 9$)

Table 10: Logit temperature, GSM8K accuracy (%) with DDPM, 256 steps/block, EDM $\rho = 7$ schedule, no CFG. Score entries are mean ± two standard errors over the 10 generations per prompt, an approximate 95% confidence interval for the mean. $T = 1$ is the unsharpened MMSE denoiser and is the same configuration as the $w = 1$ logit-space entry of Tab. 9. The bottom row is LCFG at the matched weight $w = \frac{1}{T}$, repeated from Tab. 9; the two are the same operation if the unconditional logits are uniform over the vocabulary, so the gap isolates what logit-space guidance adds over plain sharpening.

T	1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{4}$	$\frac{1}{5}$	$\frac{1}{10}$	0 (MAP)
LT at T	52.44 ± 1.00	76.88 ± 0.62	79.07 ± 0.38	78.97 ± 0.50	**79.11** ± 0.32	78.74 ± 0.23	78.61 ± 0.33
LCFG at $w = \frac{1}{T}$	52.44 ± 1.00	76.76 ± 0.50	75.16 ± 0.79	69.48 ± 0.59	60.77 ± 0.99	—	—
Δ (LT − LCFG)	0.00	+0.12	+3.91	+9.49	+18.34	—	—

LCFG peaks early and tops out lower. Tab. 9 shows that LCFG peaks at $w = 2$ in all four settings and then declines monotonically, while embedding-space CFG keeps improving out to $w = 4 \sim 10$. Comparing each at its own optimum, logit space gives a marginal gain of $+0.30$ on GSM8K/DDIM but loses by 5.68, 2.44, and 17.26 points in the other three settings.

Table 11: Composing logit temperature with the other two knobs: GSM8K accuracy (%) with DDIM, 64 steps/block, uniform-γ schedule, ST $\tau = 0.3$. Entries are mean $\pm$ two standard errors over the 10 generations per prompt, an approximate 95% confidence interval for the mean. **Bold** marks the best entry in each row.

T	1	$\frac{1}{2}$	$\frac{1}{3}$	$\frac{1}{5}$	0 (MAP)
No guidance ($w = 1$)	68.18 $\pm$ 0.33	**77.60** $\pm$ 0.23	77.42 $\pm$ 0.35	76.13 $\pm$ 0.31	71.83 $\pm$ 0.24
Embedding CFG at $w = 5$	**83.19** $\pm$ 0.18	81.44 $\pm$ 0.19	81.49 $\pm$ 0.25	79.96 $\pm$ 0.31	76.84 $\pm$ 0.44

Logit temperature sharpens effectively, and the MAP denoiser does not collapse. Tab. 10 shows sharpening alone is worth a great deal on DDPM, lifting GSM8K from 52.44 at $T = 1$ to 79.11 at $T = \frac{1}{5}$. The curve then stays flat all the way to the hard endpoint: the entire range from $T = \frac{1}{3}$ to $T = 0$ spans only 0.5 points, and $T = 0$ itself scores 78.61. Committing by argmax at high noise therefore costs nothing measurable here, and the on-manifold denoiser is a viable operating point rather than a degenerate one.

What logit-space guidance adds over sharpening is neutral at best. Note that if the unconditional branch predicted zero (i.e., a uniform distribution over the vocabulary), logit CFG at weight w will be equivalent to logit temperature at $T = \frac{1}{w}$. The gap between the two therefore measures the contribution of the structured part of the unconditional logits, which is the entire content of logit-space guidance. The bottom rows of Tab. 10 show that contribution is negligible at $w = 2$ and then strictly harmful at larger w. So logit-space CFG is best understood as a more expensive but still poorly behaved sharpener.

Logit temperature composes with ST, but adds nothing on top of CFG. LT and ST sharpen different objects: LT tempers the categorical posterior and then takes its mean, so its prediction stays inside the convex hull of $\mathbf{E}$ and saturates at the MAP vertex, whereas ST extrapolates the mean prediction away from $\frac{\mathbf{z}_t}{\alpha_t}$ and can go outside the unit ball. Tab. 11 confirms they do compose: on DDIM, where ST $= 0.3$ is already active, adding LT at $T = \frac{1}{2}$ lifts GSM8K for 9.42 points that score temperature alone does not reach. However, when CFG and ST are already on, adding a third sharpener LT harms accuracy.

Conclusion. Logit-space steering does sharpen, and logit temperature in particular is a simple knob requiring no second branch as in Fig. 4a. However, it does not outperform the default embedding-space CFG in general. We therefore keep the embedding-space formulation of (CFG) and (ST) throughout.

G.4 Ablation Study: Choosing the Bad Prediction for CFG with Self-Conditioning

[Back to Sec. 4.2 in the main paper.]

Standard CFG (Ho & Salimans, 2022), also following the idea of auto-guidance (Karras et al., 2024), extrapolates a conditional prediction away from a "bad" prediction to sharpen the model's prediction. Self-conditioning introduces a second source of conditioning, as is used in existing continuous DLMs (e.g., Lee et al. (2026); Potaptchik et al. (2026); Hu et al. (2026); see App. A for detailed discussion). We define the following notations in this subsection:

$$\mathbf{e}_\theta^{+} := \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c}, \mathbf{e}_{\text{sc}}),$$
$$\mathbf{e}_\theta^{\mathbf{c},\mathbf{0}} := \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c}, \mathbf{0}),$$
$$\mathbf{e}_\theta^{\varnothing,\mathbf{0}} := \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \varnothing, \mathbf{0}),$$

where the good prediction $\mathbf{e}_\theta^{+}$ is the conditional prediction that uses both the clean context and SC; for either bad prediction $\mathbf{e}_\theta^{-} \in \{\mathbf{e}_\theta^{\mathbf{c},\mathbf{0}}, \mathbf{e}_\theta^{\varnothing,\mathbf{0}}\}$, the guided mean embedding prediction is

$$\mathbf{e}_\theta^{w} = \mathbf{e}_\theta^{+} + (w-1)(\mathbf{e}_\theta^{+} - \mathbf{e}_\theta^{-}).$$

The *context-preserving* baseline keeps the prompt and previously generated blocks in both branches and sets $\mathbf{e}_\theta^{-} \leftarrow \mathbf{e}_\theta^{\mathbf{c},\mathbf{0}}$; it therefore extrapolates only away from the prediction made without SC. Our

proposed method in Sec. 4.2 instead uses the fully unconditional branch $\mathbf{e}_\theta^- \leftarrow \mathbf{e}_\theta^{\varnothing,\mathbf{0}}$, removing both clean context and SC.

Experimental setup. We compare the two choices using the same *Sigma*-3B checkpoint with block length 32, full GSM8K and HumanEval evaluation sets, and 10 samples per problem. The DDIM sweep uses 64 steps per block, uniform-γ schedule, and ST $\tau = 0.3$; the DDPM sweep uses 256 steps per block, EDM $\rho = 7$ schedule, and ST $\tau = 1$. We remark that the guidance is inactive at $w = 1$ and the two choices coincide.

Table 12: GSM8K accuracy (%) when choosing the bad prediction for CFG with SC. "Context/no-SC" retains clean context but removes SC (i.e., $\mathbf{e}_\theta^- \leftarrow \mathbf{e}_\theta^{\mathbf{c},\mathbf{0}}$); ours removes both (i.e., $\mathbf{e}_\theta^- \leftarrow \mathbf{e}_\theta^{\varnothing,\mathbf{0}}$). Score entries are mean ± two standard errors over the 10 sampling lanes, an approximate 95% confidence interval for the mean; Δ is ours − context/no-SC.

DDIM, 64 steps/block, uniform-γ schedule, $\tau = 0.3$

Guidance w	1	2	3	4	5	6	8	12	16
Context/no-SC	68.13 ± 0.38	72.48 ± 0.27	73.99 ± 0.47	74.37 ± 0.35	75.12 ± 0.51	76.13 ± 0.41	**76.19** ± 0.44	75.60 ± 0.56	74.09 ± 0.52
Ours	68.13 ± 0.38	82.52 ± 0.21	82.84 ± 0.28	83.04 ± 0.20	**83.19** ± 0.18	83.01 ± 0.30	82.78 ± 0.40	81.26 ± 0.40	78.00 ± 0.32
Δ	0.00	+10.05	+8.85	+8.67	+8.07	+6.88	+6.59	+5.66	+3.91

DDPM, 256 steps/block, EDM $\rho = 7$ schedule, $\tau = 1$

Guidance w	1	2	3	4	5	6	7	8	9	10
Context/no-SC	53.09 ± 0.85	62.43 ± 0.97	65.44 ± 0.77	67.53 ± 0.58	**68.45** ± 0.77	**68.45** ± 0.56	67.66 ± 0.61	67.82 ± 1.17	67.09 ± 0.48	66.76 ± 0.56
Ours	53.09 ± 0.85	77.12 ± 0.66	80.04 ± 0.39	80.99 ± 0.48	81.30 ± 0.41	82.00 ± 0.47	81.54 ± 0.64	82.05 ± 0.29	82.41 ± 0.51	**82.43** ± 0.50
Δ	0.00	+14.69	+14.60	+13.46	+12.85	+13.55	+13.88	+14.23	+15.32	+15.67

Table 13: HumanEval execution-based pass@1 (%) when choosing the bad prediction for CFG with SC. "Context/no-SC" retains clean context but removes SC (i.e., $\mathbf{e}_\theta^- \leftarrow \mathbf{e}_\theta^{\mathbf{c},\mathbf{0}}$); ours removes both (i.e., $\mathbf{e}_\theta^- \leftarrow \mathbf{e}_\theta^{\varnothing,\mathbf{0}}$). Score entries are mean ± two standard errors over the 10 sampling lanes, an approximate 95% confidence interval for the mean; Δ is ours − context/no-SC.

DDIM, 64 steps/block, uniform-γ schedule, $\tau = 0.3$

Guidance w	1	2	3	4	5	6	8	12	16
Context/no-SC	42.93 ± 0.66	46.65 ± 1.08	50.00 ± 0.81	50.85 ± 0.61	**50.91** ± 0.86	50.43 ± 0.66	49.39 ± 1.30	43.23 ± 1.20	40.85 ± 2.07
Ours	42.93 ± 0.66	53.96 ± 0.95	56.04 ± 1.16	**56.22** ± 0.83	54.09 ± 1.22	54.51 ± 0.89	54.02 ± 0.78	43.66 ± 1.52	31.22 ± 1.62
Δ	0.00	+7.31	+6.04	+5.37	+3.18	+4.08	+4.63	+0.43	-9.63

DDPM, 256 steps/block, EDM $\rho = 7$ schedule, $\tau = 1$

Guidance w	1	2	3	4	5	6	7	8	9	10
Context/no-SC	13.23 ± 1.50	20.85 ± 2.29	25.73 ± 1.51	27.93 ± 1.66	29.57 ± 1.03	29.76 ± 1.75	31.40 ± 2.07	32.38 ± 1.88	**32.50** ± 2.26	30.00 ± 1.35
Ours	13.23 ± 1.50	32.62 ± 1.98	40.55 ± 1.79	44.63 ± 1.07	46.71 ± 1.82	46.22 ± 2.12	45.24 ± 1.49	45.61 ± 1.03	**46.95** ± 1.73	45.37 ± 1.63
Δ	0.00	+11.77	+14.82	+16.70	+17.14	+16.46	+13.84	+13.23	+14.45	+15.37

Results. As shown in Tabs. 12 and 13, the fully unconditional bad prediction (ours) is consistently stronger in the useful guidance range: it almost wins every matched CFG weight in sweep on both tasks and both samplers, and, tuned over the sweep, reaches a higher ceiling in each task. Thus, we conclude that the advantage shown here is not an artifact of using different guidance search ranges.

G.5 Ablation Study: Commit-Stride Sampling

[Back to Sec. 4 in the main paper.]

Beyond block-wise decoding of L' tokens per denoising window, we further introduce a *commit stride* $S \in \{1, \ldots, L'\}$. Each denoising window still runs the same block-wise reverse process over a block of fresh Gaussian noise with length $L' = 32$, but only the leading $S \in \{8, 16, 32\}$ tokens are committed back into the causal KV cache; the remaining $L' - S$ tail positions are discarded and re-drawn as fresh noise in the next window (which is shifted forward by S absolute positions). Setting $S \leftarrow L'$ recovers the default block-decoding method. The pseudocode for commit stride can be found in Alg. 2. This is a pure inference-time technique that further improves the output quality even under matched compute budgets, as we will see below.

Setup. We evaluate on the base models of Tabs. 1 and 8: *Sigma*-3B and *Sigma*-8B, both with block size $L' = 32$. Decoding invariants match the diffusion-mode rows of Tab. 1. For each stride, CFG weight w and ST τ are chosen based on the number of DDIM steps per window: $(w, \tau) = (3, 0.2)$ at ≥ 64 steps, $(3, 0.3)$ at 32 steps, $(3, 0.5)$ at 16 steps, and $(3, 0.8)$ at 8 steps, following the recommendation in Fig. 14. All cells generate 10 samples for each prompt and report mean $\pm$ two standard errors (App. G.1).

Nominal vs. true NFE. For simplicity, we match compute cost by *nominal* NFE, defined as the number of denoiser forward passes required to commit $L' = 32$ tokens under the given (steps, stride) pair:

$$\text{Nominal NFE} := \frac{NL'}{S},$$

where N is the number of diffusion steps per denoising window and the $+1$ endpoint argmax readout is dropped so that the count is exactly proportional to the diffusion-step budget. Since every denoising window operates on the *same* L'-wide noisy block appended to the same-length clean prefix, one forward pass costs the same FLOPs regardless of S, and two rows sharing a nominal-NFE bucket in Tabs. 14 and 15 are FLOP-matched (up to a $1/N$ correction from the shared readout).

Table 14: **Commit-stride ablation on *Sigma*-3B.** Same six generative reasoning tasks as Tab. 1. *Nom. NFE* is the nominal number of forward passes per 32 committed tokens (i.e., steps/stride $\times$ 32). All DDIM rows use the uniform-γ schedule; the DDPM row uses the EDM $\rho = 7$ schedule. Rows within each nominal-NFE bucket are directly compute-comparable. Mean $\pm$ two standard errors across 10 sampling lanes. **Bold** marks the best value per column across the stride sweep.

Sampler	Nom. NFE	Steps	Stride	GSM8K	Minerva Math	HumanEval	HumanEval+	MBPP	MBPP+	Avg.
DDPM (w=2, τ=0.1)	256	256	32	84.50 ± 0.31	64.60 ± 0.28	59.45 ± 1.30	53.84 ± 1.29	52.32 ± 0.73	71.32 ± 0.91	64.34
DDIM (w=3, τ=0.2)	256	128	16	85.70 ± 0.15	64.68 ± 0.20	60.67 ± 0.80	54.88 ± 0.84	55.52 ± 0.36	72.14 ± 0.34	65.60
DDIM (w=3, τ=0.2)	256	64	8	84.94 ± 0.17	**64.73** ± 0.18	61.52 ± 0.76	56.10 ± 0.62	**58.86** ± 0.35	**75.87** ± 0.40	**67.00**
DDIM (w=3, τ=0.2)	128	128	32	83.78 ± 0.11	63.89 ± 0.13	59.15 ± 0.68	53.17 ± 0.87	50.48 ± 0.21	68.12 ± 0.51	63.10
DDIM (w=3, τ=0.2)	128	64	16	**85.82** ± 0.09	64.24 ± 0.14	60.73 ± 0.83	54.63 ± 0.85	54.06 ± 0.30	71.56 ± 0.44	65.17
DDIM (w=3, τ=0.3)	128	32	8	84.17 ± 0.29	64.44 ± 0.22	**62.13** ± 0.63	**56.34** ± 0.49	57.56 ± 0.61	74.68 ± 0.50	66.55
DDIM (w=3, τ=0.2)	64	64	32	82.93 ± 0.15	63.93 ± 0.13	56.89 ± 0.73	50.91 ± 0.66	49.26 ± 0.27	66.35 ± 0.35	61.71
DDIM (w=3, τ=0.3)	64	32	16	85.21 ± 0.25	63.44 ± 0.16	59.82 ± 0.92	54.27 ± 1.16	51.88 ± 0.62	69.87 ± 0.49	64.08
DDIM (w=3, τ=0.5)	64	16	8	83.95 ± 0.28	62.97 ± 0.23	59.63 ± 1.06	53.66 ± 0.94	54.90 ± 0.50	73.10 ± 0.57	64.70
DDIM (w=3, τ=0.3)	32	32	32	81.91 ± 0.32	62.76 ± 0.18	56.59 ± 1.31	50.67 ± 1.00	47.26 ± 0.46	63.49 ± 0.77	60.45
DDIM (w=3, τ=0.5)	32	16	16	82.40 ± 0.38	62.37 ± 0.21	56.77 ± 0.82	51.52 ± 0.90	50.12 ± 0.71	67.72 ± 0.46	61.82
DDIM (w=3, τ=0.8)	32	8	8	80.60 ± 0.47	59.48 ± 0.22	52.56 ± 1.94	46.95 ± 1.86	46.50 ± 0.80	60.50 ± 1.02	57.76

We present the results for commit-stride sampling for *Sigma*-3B and *Sigma*-8B in Tabs. 14 and 15, respectively. The tables sweep all three strides $S \in \{8, 16, 32\}$ within each nominal-NFE bucket $\{256, 128, 64, 32\}$. At the top NFE (nominal 256), the DDIM 64 steps / stride 8 row is the best on both models: it beats the tuned DDPM-256 baseline of Tabs. 1 and 8 by $+2.66$ average points on 3B and $+0.76$ on 8B, with the largest per-task gains on the coding benchmarks such as MBPP. The same ordering holds down to around nominal 64: committing fewer tokens per window (i.e., smaller stride S) improves the average monotonically. At nominal 128 NFE, DDIM 32 steps / stride 8 reaches within a point of the top budget on both sizes, and still above the full-stride DDPM-256 baselines. The benefit of a smaller stride disappears once it forces the per-window step count below roughly 16:

Table 15: **Commit-stride ablation on *Sigma*-8B.** Same six tasks as Tab. 8; conventions match Tab. 14. **Bold** marks the best value per column across the stride sweep.

Sampler	Nom. NFE	Steps	Stride	GSM8K	Minerva Math	HumanEval	HumanEval+	MBPP	MBPP+	Avg.
DDPM (w=2, τ=0.1)	256	256	32	89.07 ± 0.28	67.47 ± 0.20	**61.40** ± 1.73	**57.44** ± 1.62	64.78 ± 0.54	77.83 ± 0.72	69.66
DDIM (w=3, τ=0.2)	256	128	16	88.64 ± 0.11	67.59 ± 0.10	60.91 ± 0.64	55.67 ± 0.60	67.44 ± 0.35	**81.75** ± 0.33	70.33
DDIM (w=3, τ=0.2)	256	64	8	**90.17** ± 0.15	**68.82** ± 0.20	60.49 ± 0.98	53.96 ± 0.93	**67.72** ± 0.21	81.35 ± 0.28	**70.42**
DDIM (w=3, τ=0.2)	128	128	32	88.70 ± 0.14	67.28 ± 0.09	58.48 ± 1.00	52.68 ± 1.00	63.22 ± 0.28	76.85 ± 0.31	67.87
DDIM (w=3, τ=0.2)	128	64	16	88.76 ± 0.17	67.63 ± 0.17	59.33 ± 1.06	53.54 ± 0.94	66.62 ± 0.42	80.90 ± 0.31	69.46
DDIM (w=3, τ=0.3)	128	32	8	89.61 ± 0.24	68.11 ± 0.15	61.22 ± 0.55	55.18 ± 0.69	66.22 ± 0.49	80.19 ± 0.34	70.09
DDIM (w=3, τ=0.2)	64	64	32	88.68 ± 0.19	67.30 ± 0.18	58.90 ± 0.71	52.56 ± 0.54	61.96 ± 0.36	76.35 ± 0.28	67.62
DDIM (w=3, τ=0.3)	64	32	16	88.15 ± 0.33	67.58 ± 0.21	58.05 ± 1.09	52.80 ± 0.98	65.32 ± 0.41	78.10 ± 0.59	68.33
DDIM (w=3, τ=0.5)	64	16	8	88.45 ± 0.31	67.21 ± 0.26	61.40 ± 1.73	55.67 ± 1.31	61.92 ± 0.77	75.21 ± 0.73	68.31
DDIM (w=3, τ=0.3)	32	32	32	87.78 ± 0.33	67.13 ± 0.15	58.54 ± 1.04	53.17 ± 1.27	59.54 ± 0.57	74.15 ± 0.43	66.72
DDIM (w=3, τ=0.5)	32	16	16	87.04 ± 0.35	67.14 ± 0.27	58.41 ± 1.22	53.29 ± 1.19	60.46 ± 0.39	73.41 ± 0.51	66.63
DDIM (w=3, τ=0.8)	32	8	8	85.91 ± 0.29	64.57 ± 0.22	55.98 ± 1.46	50.61 ± 1.75	52.74 ± 0.85	64.44 ± 0.87	62.38

at nominal 32, stride 8 leaves only 8 steps per window, and its coarse time grid collapses quality due to larger discretization error, falling below the stride-16 and stride-32 rows at the same budget.

G.6 Ablation Study: Time Discretization

[Back to Sec. 5.1 in the main paper.]

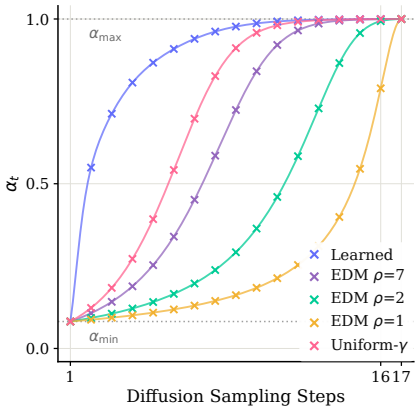


Figure 12: **Time discretization** for 16-step generation: 16 diffusion steps plus an extra decoding step.

We mainly consider the following discretization schemes. Let N be the total number of steps and assume the i-th step ($i = 1, ..., N$) goes from t_{i-1} to $t_i < t_{i-1}$.

- Learned: arithmetic progression in time: $t_i = 1 - \frac{i}{N}$.
- Uniform-γ: arithmetic progression in γ, the minus log-SNR: $\gamma(t_i) = \gamma_{\max} + \frac{i}{N}(\gamma_{\min} - \gamma_{\max})$.
- EDM ρ (Karras et al., 2022): first convert the VP schedule to VE by defining $\tilde{\sigma}_t := \frac{\sigma_t}{\alpha_t}$ as the essential standard deviation at time t. The grid points are chosen so that $\tilde{\sigma}_{t_i}^{1/\rho}$ is an arithmetic progression, i.e., $\tilde{\sigma}_{t_i} = \left(\tilde{\sigma}_{\max}^{1/\rho} + \frac{i}{N}(\tilde{\sigma}_{\min}^{1/\rho} - \tilde{\sigma}_{\max}^{1/\rho})\right)^{\rho}$. Note that as $\rho \to \infty$, this converges to the uniform-γ scheme (Prop. 2).

We visualize the three schemes in Fig. 12. Empirically, we find that for high NFE sampling, both the uniform-γ and EDM $\rho = 7$ schemes perform similarly, while uniform-γ generally performs the best

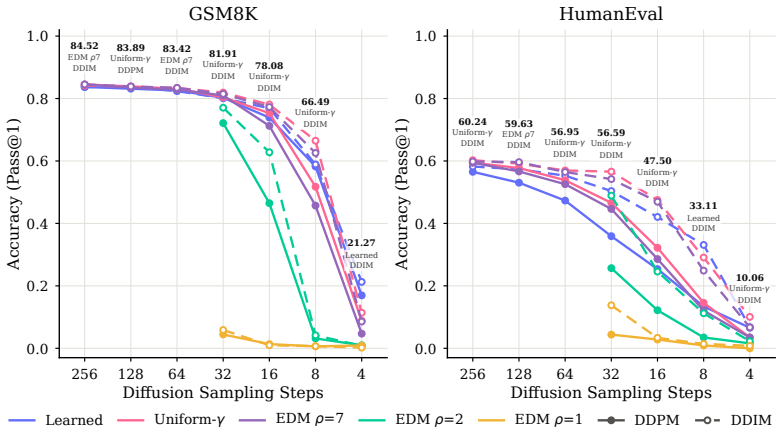


Figure 13: DDIM substantially outperforms DDPM at low NFEs.

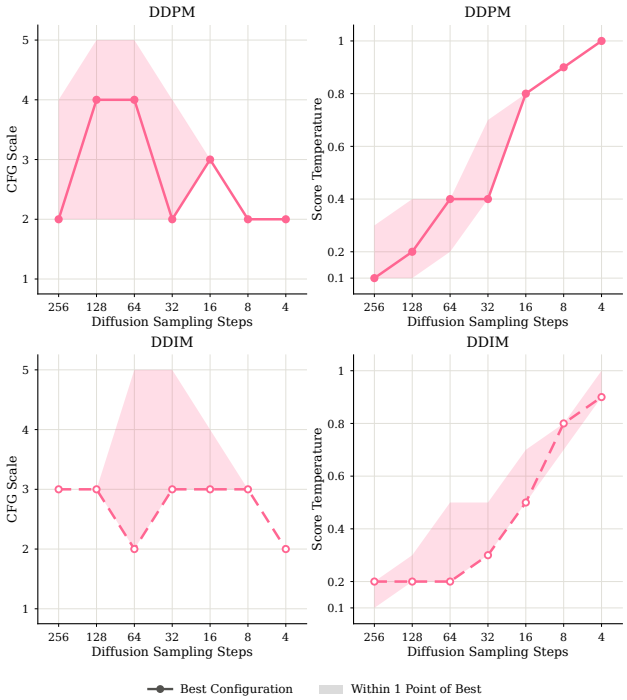


Figure 14: **Best CFG-ST configuration vs. diffusion steps** for the best average score of GSM8K & HumanEval under uniform-γ discretization.

for low NFE sampling. See Fig. 13 for the few-step reasoning accuracy of DDPM and DDIM over GSM8K and HumanEval.

G.7 Higher-order ODE Solvers

[Back to Sec. 5.1 in the main paper.]

In this subsection, we test if existing higher-order probability-flow ODE (PFODE) solvers improve low-NFE sampling. All methods use the same *Sigma*-3B checkpoint, block length 32, uniform-γ time discretization (App. G.6), self-conditioning on, and DDIM-selected CFG and ST configurations as in Fig. 14: at 8 steps, we use $w = 3, \tau = 0.8$; at 16 steps, we use $w = 3, \tau = 0.5$. As described in App. G.1, we generate 10 responses per problem and report the mean $\pm$ two standard errors. We match compute across methods at 8 or 16 sampling-loop denoiser calls (the "Steps" column of

Tab. 16); adding the final argmax decoding at $t_N = 0$ (Alg. 2) gives a total NFE of 9 or 17. Midpoint spends two evaluations per outer transition so performs $S/2$ transitions at S steps, while the other four methods perform S transitions.

Table 16: Higher-order solver comparison on *Sigma*-3B. "Steps" is the number of denoiser calls in the sampling loop, and NFE = Steps +1 is the total denoiser count including the final argmax decoding; both are matched across methods within each row. Each entry is mean ± two standard errors over 10 responses per question, forming an approximate 95% confidence interval. **Bold** marks the best method in each row.

Task	Steps	NFE	DDIM	DPM-Solver++(2M)	ρAB1-DEIS	ρAB2-DEIS	ρRK-DEIS
GSM8K	8	9	**66.49** ± 0.52	22.77 ± 0.92	61.78 ± 0.46	48.95 ± 1.04	2.88 ± 0.20
GSM8K	16	17	**78.08** ± 0.34	33.18 ± 0.60	48.96 ± 0.98	2.65 ± 0.37	0.86 ± 0.15
HumanEval	8	9	**29.09** ± 1.59	12.07 ± 1.57	24.45 ± 1.66	20.24 ± 1.37	3.72 ± 1.10
HumanEval	16	17	**47.50** ± 1.14	22.01 ± 1.49	33.48 ± 1.39	1.58 ± 0.58	1.77 ± 0.56
MBPP	8	9	**22.96** ± 1.06	0.48 ± 0.17	9.48 ± 0.61	4.66 ± 0.46	0.00 ± 0.00
MBPP	16	17	**42.36** ± 0.66	0.36 ± 0.18	6.58 ± 0.52	0.00 ± 0.00	0.00 ± 0.00

Update forms. We write the discretization time points as $1 = t_0 > t_1 > \cdots > t_N = 0$ and describe transition $i \to i+1$. For conciseness, let $\alpha_i = \alpha_{t_i}$, $\sigma_i = \sigma_{t_i}$, $\lambda_i = \log \frac{\alpha_i}{\sigma_i} = -\frac{\gamma(t_i)}{2}$, and define $\rho_i = \frac{\sigma_i}{\alpha_i}$. The model prediction at the current point is $\hat{\mathbf{e}}_i = \mathbf{e}_\theta(\mathbf{z}_i, t_i; \mathbf{c})$ with implied noise prediction $\hat{\boldsymbol{\epsilon}}_i = \frac{1}{\sigma_i}(\mathbf{z}_i - \alpha_i \hat{\mathbf{e}}_i)$. Unless stated otherwise, we assume both quantities already carry the CFG and ST transformations of (CFG) and (ST). The DDIM update in (DDIM) is

$$\mathbf{z}_{i+1}^{\mathrm{DDIM}} = \alpha_{i+1}\hat{\mathbf{e}}_i + \sigma_{i+1}\hat{\boldsymbol{\epsilon}}_i. \tag{20}$$

This is the zeroth-order exponential integral discretization of the PFODE, in either prediction form as described below:

- In the $\boldsymbol{\epsilon}$-prediction form, setting $\mathbf{y} := \frac{\mathbf{z}}{\alpha}$ gives $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\rho} = \hat{\boldsymbol{\epsilon}}_\theta$, hence

$$\mathbf{z}_{i+1} = \frac{\alpha_{i+1}}{\alpha_i}\mathbf{z}_i + \alpha_{i+1}\int_{\rho_i}^{\rho_{i+1}} \hat{\boldsymbol{\epsilon}}_\theta(\rho)\,\mathrm{d}\rho. \tag{21}$$

- In the $\mathbf{e}$-prediction form, setting $\mathbf{v} := \frac{\mathbf{z}}{\sigma}$ and $u = \frac{\alpha}{\sigma}$ gives $\frac{\mathrm{d}\mathbf{v}}{\mathrm{d}u} = \hat{\mathbf{e}}_\theta$, hence

$$\mathbf{z}_{i+1} = \frac{\sigma_{i+1}}{\sigma_i}\mathbf{z}_i + \sigma_{i+1}\int_{u_i}^{u_{i+1}} \hat{\mathbf{e}}_\theta(u)\,\mathrm{d}u. \tag{22}$$

Freezing either learned field on the interval recovers (20).

DPM-Solver++(2M) (Lu et al., 2025) uses the $\mathbf{e}$-prediction form and linearly extrapolates $\hat{\mathbf{e}}_\theta$ from the two most recent grid points in λ. With $h_i = \lambda_{i+1} - \lambda_i > 0$ and $r_i = \frac{h_{i-1}}{h_i}$,

$$\mathbf{d}_i = \begin{cases} \hat{\mathbf{e}}_i, & i = 0, \\ \left(1 + \frac{1}{2r_i}\right)\hat{\mathbf{e}}_i - \frac{1}{2r_i}\hat{\mathbf{e}}_{i-1}, & i > 0, \end{cases} \tag{23}$$

$$\mathbf{z}_{i+1}^{\mathrm{DPM}} = \frac{\sigma_{i+1}}{\sigma_i}\mathbf{z}_i + \alpha_{i+1}(1 - \mathrm{e}^{-h_i})\mathbf{d}_i. \tag{24}$$

Equivalently, this is $\mathbf{z}_{i+1}^{\mathrm{DDIM}} + \alpha_{i+1}(1 - \mathrm{e}^{-h_i})\frac{1}{2r_i}(\hat{\mathbf{e}}_i - \hat{\mathbf{e}}_{i-1})$ for $i > 0$.

The second-order ρAB1-DEIS solver (Zhang & Chen, 2023)[3] instead uses the natural quadrature coordinate in (21). Let $k_i = \rho_{i+1} - \rho_i$ and $p_i = \rho_i - \rho_{i-1}$. The first transition uses DDIM. After one history value is available,

$$\mathbf{z}_{i+1}^{\mathrm{AB2}} = \mathbf{z}_{i+1}^{\mathrm{DDIM}} + \frac{\alpha_{i+1}k_i^2}{2p_i}\left(\hat{\boldsymbol{\epsilon}}_i - \hat{\boldsymbol{\epsilon}}_{i-1}\right). \tag{25}$$

[3]DEIS indexes its Adams–Bashforth (AB) methods by the degree of the extrapolating polynomial, thus the r-th order polynomial provides convergence order $r + 1$.

The third-order ρAB2-DEIS solver (Zhang & Chen, 2023) is the corresponding three-point method for $\frac{\mathrm{d}\mathbf{y}}{\mathrm{d}\rho} = \hat{\boldsymbol{\epsilon}}_\theta$. It uses DDIM on the first transition, the ρAB1-DEIS rule in (25) on the second transition, and then integrates the quadratic Lagrange interpolant through the newest three noise predictions. With $q_i = \rho_{i-1} - \rho_{i-2}$,

$$b_{0,i} = 1 + \frac{(2p_i + q_i)k_i}{2p_i(p_i + q_i)} + \frac{k_i^2}{3p_i(p_i + q_i)},$$
$$b_{1,i} = -\frac{(p_i + q_i)k_i}{2p_i q_i} - \frac{k_i^2}{3p_i q_i},$$
$$b_{2,i} = \frac{p_i k_i}{2q_i(p_i + q_i)} + \frac{k_i^2}{3q_i(p_i + q_i)},$$

and

$$\mathbf{z}_{i+1}^{\mathrm{AB3}} = \mathbf{z}_{i+1}^{\mathrm{DDIM}} + \alpha_{i+1}k_i\left[(b_{0,i} - 1)\hat{\boldsymbol{\epsilon}}_i + b_{1,i}\hat{\boldsymbol{\epsilon}}_{i-1} + b_{2,i}\hat{\boldsymbol{\epsilon}}_{i-2}\right].$$

By construction, $b_{0,i} + b_{1,i} + b_{2,i} = 1$, so the update reduces to DDIM when $\hat{\boldsymbol{\epsilon}}_{i-2} = \hat{\boldsymbol{\epsilon}}_{i-1} = \hat{\boldsymbol{\epsilon}}_i$. On a ρ-uniform grid, the weights collapse to the classical $\left(\frac{23}{12}, -\frac{16}{12}, \frac{5}{12}\right)$.

Finally, ρRK-DEIS (Zhang & Chen, 2023) is explicit midpoint Runge-Kutta method in the same integrating-factor state $\mathbf{y} = \frac{\mathbf{z}}{\alpha}$. It uses the arithmetic midpoint $\rho_{m,i} = \frac{1}{2}(\rho_i + \rho_{i+1})$, converts it back to $\gamma_{m,i} = 2\log\rho_{m,i}$ and therefore to $(\alpha_{m,i}, \sigma_{m,i})$, predicts an intermediate state,

$$\mathbf{z}_{m,i} = \alpha_{m,i}\left(\frac{\mathbf{z}_i}{\alpha_i} + (\rho_{m,i} - \rho_i)\hat{\boldsymbol{\epsilon}}_i\right), \tag{26}$$

queries the denoiser at $(\mathbf{z}_{m,i}, t_{m,i})$ to obtain $\hat{\boldsymbol{\epsilon}}_{m,i}$, and completes the full interval as

$$\mathbf{z}_{i+1}^{\mathrm{midpoint}} = \alpha_{i+1}\left(\frac{\mathbf{z}_i}{\alpha_i} + (\rho_{i+1} - \rho_i)\hat{\boldsymbol{\epsilon}}_{m,i}\right) = \mathbf{z}_{i+1}^{\mathrm{DDIM}} + \alpha_{i+1}(\rho_{i+1} - \rho_i)(\hat{\boldsymbol{\epsilon}}_{m,i} - \hat{\boldsymbol{\epsilon}}_i). \tag{27}$$

Results. Tab. 16 shows that none of these higher-order updates improves accuracy over DDIM at the tested low-NFE sampling settings. The closest alternative is ρAB1-DEIS at 8 steps, but it still trails DDIM on GSM8K and HumanEval and is much worse on MBPP. Increasing the budget from 8 to 16 steps improves DDIM on every task, while the higher-order solvers do not improve reliably. ρAB2-DEIS and ρRK-DEIS become especially unstable at 16 steps.

Trajectory diagnosis. We perform a trajectory diagnosis by running different solvers from the same initial Gaussian noise $\mathbf{z}_1$ given the same prompt and few-shot examples, saving the complete block-0 trajectory. We use the first 32 questions of either GSM8K or HumanEval dataset, draw eight generations per question, and use three CFG–ST configurations $(w, \tau) \in \{(3, 0.4), (3, 0.6), (2, 0.7)\}$, each with self-conditioning on and off. This gives 2 tasks $\times$ 3 CFG–ST pairs $\times$ 2 self-conditioning configurations $= 12$ settings. For each setting, the initial eight-lane noise tensor is identical across solvers and step budgets. We evaluate runs with $8, 16, 32, 64, 128$, and 256 sampling steps together with a paired DDIM-512 high-resolution reference as ground truth.

Recall that all embedding vectors in $\mathbf{E}$ are unit-norm, and thus ideally our terminal state $\mathbf{z}_0$ should be close to the unit sphere. Panels (a) and (b) of Fig. 15 measure the geometry of the terminal latent states. Panel (a) averages $|\|\mathbf{z}_0\| - 1|$ within each setting and then equally across the 12 settings. For panel (b), we compute the nearest-embedding cosine similarity $\max_k \cos(\mathbf{z}_0, \mathbf{E}_k)$, take its fifth percentile within each setting, and then average the 12 setting-level percentiles. This tail statistic reveals endpoints stranded between vocabulary vertices. Error bars are standard errors across settings.

At 8 steps, the mean off-sphere distance is 0.014 for DDIM, but 1.29 for ρAB1-DEIS and 26.2 for DPM-Solver++(2M); their fifth-percentile nearest-embedding cosine is 0.994, 0.851, and 0.845, respectively. Thus, coarse higher-order extrapolation can move $\mathbf{z}_0$ far outside the clean embedding manifold. Because the reconstruction term is applied only at $t = 0$ (4), the model is not trained to decode such off-manifold terminal states, this causes a large decoding error. By 32 steps, all three methods return close to the sphere and to a vocabulary vertex.

Implications and future work. This negative result suggests that low-NFE inference for *Sigma* is not limited only by the formal order of the ODE integrator. Under strong CFG, ST, and SC, the effective denoising field can be less smooth than the trajectories for which standard image-diffusion

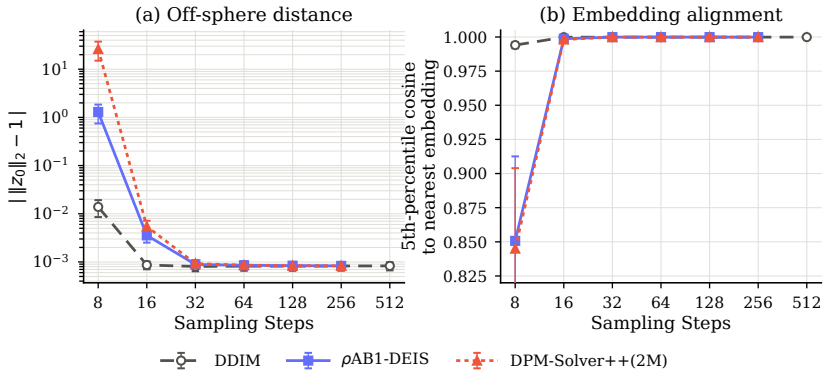


Figure 15: Endpoint geometry under low-step ODE solvers. (a) Terminal distance from the unit-sphere vocabulary manifold. (b) Fifth-percentile cosine similarity to the nearest vocabulary embedding. At 8 steps, higher-order extrapolation leaves endpoints far off-manifold and between vocabulary vertices, whereas these differences largely disappear by 32 steps.

solvers were designed. In this regime, formal integration order alone is not a reliable predictor of downstream quality, and DDIM remains the most robust choice among the solvers tested. We leave exploration of more stable and efficient higher-order ODE solvers for future work.

G.8 Pass@k Evaluation

[Back to Sec. 4.3 in the main paper.]

We study inference-time scaling in two distinct evaluation regimes. First, we compare the pretrained 3B models on MBPP, Minerva Geometry, and GSM8K under few-shot prompting. Second, we compare the instruct 3B models on AIME 2025 under the non-thinking protocol of Tab. 2.

Metric. For a problem with c correct answers among n generations, we use the standard unbiased estimator

$$\text{pass@}k = 1 - \frac{\binom{n-c}{k}}{\binom{n}{k}}$$

and average it over problems to obtain the pass@k accuracy on a evaluation dataset. Cross-family uncertainty is computed from paired per-problem differences and reported as a 95% confidence interval.

G.8.1 Base Model: MBPP

Setup. We compare the pretrained 3B *Sigma* and NLD (Fu et al., 2026) models on all 500 MBPP problems, drawing 256 generations per problem for every configuration. Both models use block size 32. *Sigma* uses DDPM with 256 steps and the EDM $\rho = 7$ schedule, sweeping CFG $w \in \{1, 2, 3\}$ and ST $\tau \in \{0.1, 0.4, 0.7, 1.0\}$; NLD uses 32 transfer steps per block, sweeping confidence threshold $\vartheta \in \{0.8, 0.9\}$ and temperature $T \in \{0.3, 0.6, 1.0\}$. Candidate draws use different random seeds while preserving identical prompts and few-shot examples across configurations, as in App. G.1.

Results. The left panel of Fig. 16 shows the best performance available in each tested sampler grid at every generation budget. As k grows, the *Sigma* setting carrying this curve moves from $\tau = 0.1$ to 0.4, 0.7, and finally 1.0, shifting from single-sample quality toward broader coverage. Because the tested grids differ in size, this is a best-of-grid comparison; the right panel provides the complementary fixed-configuration result. One fixed *Sigma* configuration reaches the same pass@256 value of 0.918, compared with 0.876 for NLD.

String-diversity diagnostic. We further investigate exact-string repetition as a partial proxy for output diversity, after stripping leading and trailing whitespaces. At the selected fixed MBPP configurations, *Sigma* repeats 10.4% of its 256 output strings, compared with 53.1% for NLD. Although NLD produces more correct samples per problem on average (139.5 versus 91.5), *Sigma*

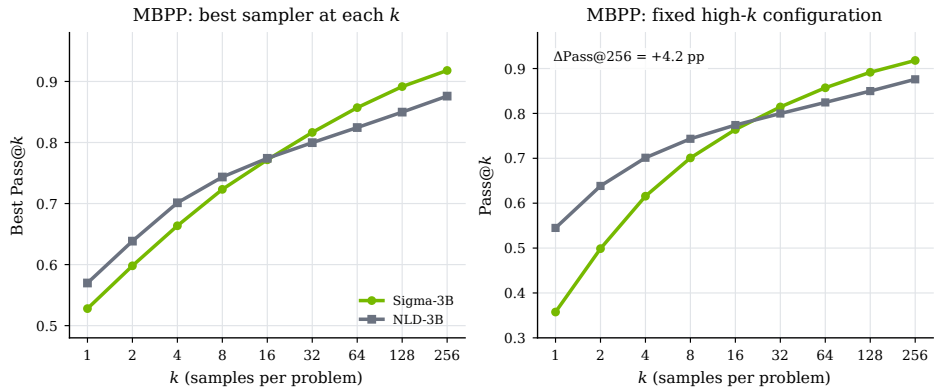


Figure 16: **Pass@k on MBPP.** *Left:* at each k, each family uses the best configuration in its tested sampler grid; *Sigma* overtakes NLD at $k = 32$ and reaches 0.918 versus 0.876 at $k = 256$. *Right:* one fixed high-coverage configuration per family, selected by the one reaching the highest pass@256 accuracy: *Sigma* $(w = 2, \tau = 1.0)$ and NLD $(\vartheta = 0.9, T = 1.0)$. The $+4.2$ point improvement at $k = 256$ has a paired 95% CI of $[+1.7, +6.7]$, with the interval excluding zero from $k = 64$ onward.

produces more distinct correct strings (71.4 versus 46.9), consistent with its later saturation and higher problem coverage.

G.8.2 Base Model: Minerva Geometry

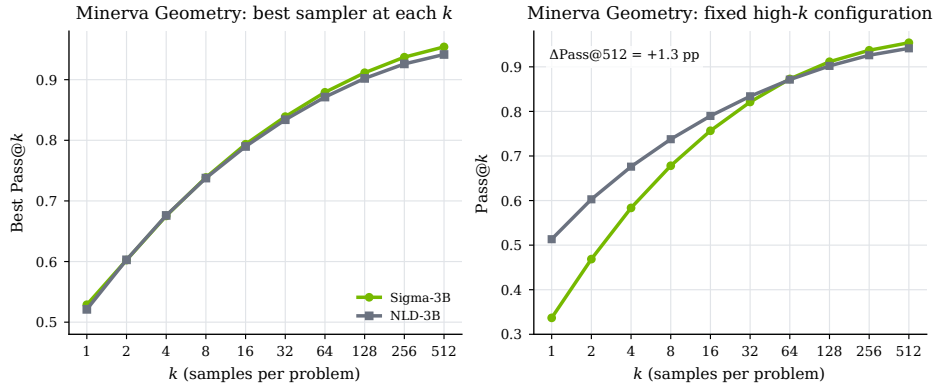


Figure 17: **Pass@k on Minerva Geometry.** *Left:* the best tested configuration in each family at every k; *Sigma* leads from $k = 8$ onward and reaches 0.954 versus 0.942 at $k = 512$. *Right:* fixed high-coverage configurations *Sigma* $(w = 2, \tau = 1.0)$ and NLD $(\vartheta = 0.9, T = 1.0)$. Their $+1.3$ point difference at $k = 512$ is directional but unresolved, with a paired 95% CI of $[-0.3, +2.8]$.

Setup. We next compare the same 3B base models on the 479 Minerva Geometry problems. Each tested configuration has 256 generations per problem, and the highlighted configurations are extended to 512. *Sigma* now uses DDIM with 32 steps and the uniform-γ schedule, sweeping $w \in \{1, 2, 3\}$ and $\tau \in \{0.4, 0.7, 1.0\}$ plus $(w, \tau) \in \{(4, 0.7), (5, 0.7)\}$; NLD uses 32 transfer steps with $\vartheta = 0.9$ and $T \in \{0.3, 0.6, 1.0\}$, as we have observed in the previous experiment on MBPP that $\vartheta = 0.9$ almost always outperforms $\vartheta = 0.8$ under the same temperature.

Results. In the left panel of Fig. 17, *Sigma* leads by 0.1 point at $k = 8$, and the gap grows monotonically to 1.3 points at $k = 512$. The setting carrying its best-tested curve moves from $\tau = 0.4$ to 0.7 and then 1.0. The fixed comparison in the right panel has the same direction at high k, but unlike MBPP, its confidence interval still spans zero.

Matched-pass@1 controls on Minerva Geometry. We next hold single-sample quality fixed on Geometry. Increasing CFG at $\tau = 0.7$ moves *Sigma* into the pass@1 band of NLD's best sampler

$(\vartheta = 0.9, T = 1)$, producing three matched operating points, including one whose pass@1 differs by only 0.0005. As shown in Tab. 17, all three *Sigma* settings trace essentially the same pass@k curve as best-tuned NLD: beyond pass@1, every difference is within 0.007.

Table 17: ***Sigma* ties the best-tuned NLD sampler** $(\vartheta = 0.9, T = 1)$ **at matched pass@1 on Minerva Geometry.** Entries are paired differences in pass@k (*Sigma* − NLD) with 512 independent generations per problem for every setting.

Sigma sampling configuration	$k = 1$	$k = 8$	$k = 32$	$k = 64$	$k = 128$	$k = 256$	$k = 512$
$w = 3, \tau = 0.7$	−0.0080	−0.002	+0.001	0.000	−0.003	−0.002	+0.006
$w = 4, \tau = 0.7$	−0.0025	−0.001	+0.002	+0.001	−0.001	−0.004	−0.004
$w = 5, \tau = 0.7$	+0.0005	+0.001	+0.005	+0.004	−0.001	−0.005	−0.004

Comparison with lower-temperature NLD samplers. At the same pass@1, the outcome is stronger against NLD configurations with lower sampling temperature. As Tab. 18 shows, *Sigma* improves pass@256 by 1.9 to 6.6 points across all five matched comparisons, with every paired interval excluding zero. In short, *Sigma* dominates NLD when the latter is poorly tuned for coverage and ties its best-tuned high-pass@k sampler.

Table 18: ***Sigma* outperforms lower-temperature NLD samplers at matched pass@1 on Minerva Geometry.** The final columns report the paired pass@256 difference (*Sigma* − NLD) and its 95% confidence interval.

Sigma vs. NLD	Δ pass@1	Δ pass@256	95% CI
$w = 4, \tau = 0.7$ vs. $\vartheta = 0.9, T = 0.3$	−0.0103	**+0.066**	[+0.044, +0.088]
$w = 5, \tau = 0.7$ vs. $\vartheta = 0.9, T = 0.3$	−0.0073	**+0.065**	[+0.042, +0.088]
$w = 3, \tau = 0.7$ vs. $\vartheta = 0.9, T = 0.6$	−0.0115	**+0.024**	[+0.008, +0.041]
$w = 4, \tau = 0.7$ vs. $\vartheta = 0.9, T = 0.6$	−0.0064	**+0.020**	[+0.003, +0.037]
$w = 5, \tau = 0.7$ vs. $\vartheta = 0.9, T = 0.6$	−0.0034	**+0.019**	[+0.002, +0.036]

G.8.3 Base Model: GSM8K

Setup and results. We repeat the comparison on GSM8K (1319 problems, 8-shot), drawing 256 generations per problem for every configuration; *Sigma* uses DDIM with 32 steps and the uniform-γ schedule, and NLD uses 32 transfer steps per block. We observe that GSM8K is nearly saturated for both families, so the family separation lives entirely in the hard tail. We report the full per-configuration results in Tab. 19 and the best-of-family comparison in Fig. 18. Fixing one high-coverage configuration per family, *Sigma* $(w = 2, \tau = 1.0)$ against NLD's best sampler $(\vartheta = 0.9, T = 1.0)$, reproduces the same pattern as MBPP and Geometry: NLD leads at small k, the curves cross near $k = 16$, and *Sigma* is ahead from $k = 32$ onward. The paired per-problem advantage is +0.7 points and its 95% confidence interval excludes zero at *every* $k \geq 32$ through $k = 256$ (+0.0072 [+0.003, +0.012] at $k = 32$; +0.0068 [+0.002, +0.012] at $k = 256$). *Sigma*'s selected configuration also solves 1314 of 1319 problems at least once versus 1305 for NLD.

Table 19: **Full pass@k results on GSM8K**, sorted by pass@256. "Solved" is the number of problems answered correctly at least once in 256 generations.

Model	Configuration	Pass@1	Pass@8	Pass@32	Pass@64	Pass@256	Solved
Sigma	w=2, τ=1.0	0.651	0.942	0.980	0.988	**0.996**	**1314**
NLD	ϑ=0.9, T=1.0	0.823	0.950	0.973	0.981	0.989	1305
Sigma	w=3, τ=0.7	0.806	0.938	0.965	0.974	0.987	1302
NLD	ϑ=0.9, T=0.3	0.836	0.913	0.937	0.946	0.964	1271
Sigma	w=2, τ=0.4	0.799	0.878	0.906	0.916	0.931	1228

Base-model summary. Across MBPP, Minerva Geometry, and GSM8K, CFG and ST expose a broad exploration–exploitation frontier. The high-k advantage survives with one fixed configuration on MBPP and GSM8K, while the matched-pass@1 Geometry controls show parity with NLD's best-tuned sampler and gains only against its lower-temperature settings.

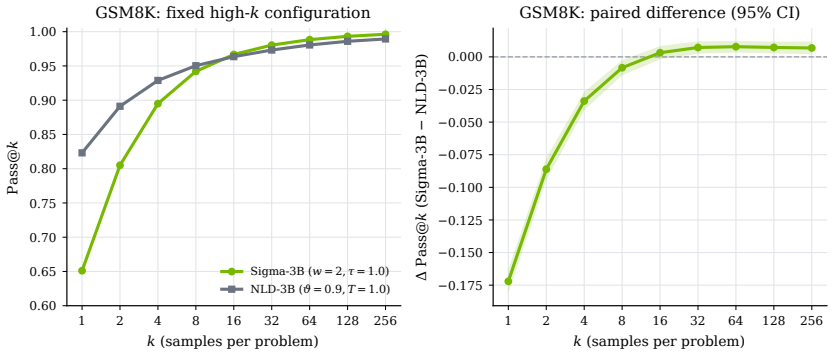


Figure 18: ***Sigma* overtakes NLD in the GSM8K hard tail.** Best-of-family fixed configuration: *Sigma* (w=2, τ=1.0) versus NLD's best sampler (ϑ=0.9, T=1.0). *Left:* the pass@k curves; GSM8K saturates high, so the curves cross near $k = 16$ and the gap is small. *Right:* the paired per-problem difference (*Sigma* − NLD) with its 95% CI, where the effect is legible: negative at $k \leq 8$, then positive with the interval excluding zero at every $k \geq 32$.

G.8.4 Instruct Models: AIME 2025

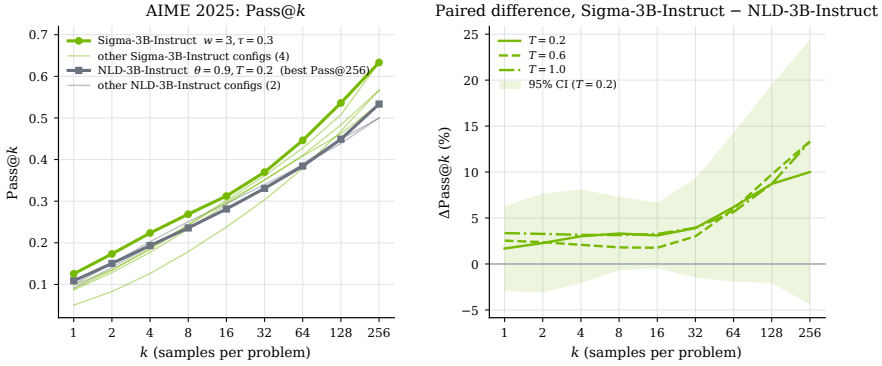


Figure 19: ***Sigma* leads NLD at every k on AIME 2025, with a margin that grows with k.** *Left:* all eight tested configurations at $n = 256$; the bold curve of each family is its best setting by pass@256. Four *Sigma* settings exceed every NLD setting at $k = 256$, and the fifth ties NLD's best ($\vartheta = 0.9, T = 0.2$). *Right:* paired per-problem differences against each NLD temperature, with the 95% interval of the highlighted pair shaded. The lead is consistent in direction at every k against all three NLD samplers.

The study above uses the pretrained checkpoints under few-shot prompting. Because pass@k is often read as a proxy for the headroom available to reinforcement learning, we repeat the comparison on the *instruct* models, under the same non-thinking protocol as the benchmark table in Tab. 2 (harness described in App. H.2), on AIME 2025.

Protocol and sampling settings. We use the same unbiased estimator as above, averaged over the 30 AIME 2025 problems, and draw 256 generations per problem for every configuration of both families. All cross-family comparisons are paired per problem and reported with a 95% confidence interval.

Both models use block size 32. *Sigma* uses DDIM with 32 steps and the uniform-γ schedule, sweeping the five settings $(w, \tau) \in \{(3, 0.3), (4, 0.3), (4, 0.7), (5, 0.7), (3, 1.0)\}$; NLD uses confidence-thresholded masked decoding with $\vartheta = 0.9$ and $T \in \{0.2, 0.6, 1.0\}$. The response budgets are 4096 tokens for *Sigma* and 16384 for NLD, but the shorter *Sigma* cap is not binding since the reasonable generations (i.e., the samples that are not degenerate, such as repeating the same pattern without emitting EOS token) are all within this length range for both models.

A consistent lead that widens with the budget. The left panel of Fig. 19 shows that the best *Sigma* setting $(w = 3, \tau = 0.3)$ leads NLD's best sampler $(\vartheta = 0.9, T = 0.2)$ at *every* k, from $+1.7$ points at $k = 1$ to $+10.0$ points at $k = 256$. It is also robust to the choice of *Sigma* setting: at $k = 256$, four

of the five sweep points, namely $(w, \tau) = (3, 0.3)$ and $(4, 0.7)$ at 0.633, $(4, 0.3)$ and $(3, 1.0)$ at 0.567, exceed every NLD point, and the fifth, $(5, 0.7)$ at 0.533, ties NLD's best setting $(\vartheta = 0.9, T = 0.2)$ at 0.533. Against NLD's other two temperatures the paired gap at $k = 256$ is $+13.3$ points in both cases. We remark that, unlike the base-model result, the gain here does not come at the cost of single-sample quality: *Sigma* leads on pass@1 as well.

Statistical resolution. The right panel of Fig. 19 shows the limit of this evidence. Of the 27 paired comparisons (nine values of k against three NLD samplers), only $(w = 3, \tau = 0.3)$ against $(\vartheta = 0.9, T = 0.6)$ at $k = 256$ excludes zero Since AIME 2025 has only 30 problems, the width of these intervals is set by the number of problems rather than the number of generations, and drawing more samples per problem cannot narrow them.

G.9 Reasoning Trajectory Analysis

[Back to Sec. 4.3 in the main paper.]

See Fig. 20 for the commit order analysis between continuous and discrete dLMs.

G.10 Wall-clock Cost of the Packed CFG Forward

[Back to Sec. 4.2 in the main paper.]

Table 20: Wall-clock cost per denoiser evaluation (NFE) of packing the conditional and unconditional CFG branches into one forward pass, at batch size 1. Times are in milliseconds on H100. Δ is the time cost of packed conditional and unconditional predictions (packed CFG) $-$ only conditional prediction (no CFG), reported as mean $\pm$ two standard errors over the 16 questions, an approximate 95% confidence interval for the mean. The *transformer forward* row shows that doubling the query length from L' to $2L'$ leaves both the KV cache and the weight traffic unchanged, so the extra work hides entirely under the memory-bandwidth floor. The residual cost is when passing through the logit head and computing the categorical reparameterization $\mathbf{e}_\theta = \mathrm{softmax}(\boldsymbol{\ell}_\theta)\mathbf{E}$, which is independent of how the two branches are batched.

Block Size	Stage	No CFG (ms)	Packed CFG (ms)	Δ	Δ (%)
$L' = 32$	Transformer forward	18.06	18.38	$+0.322 \pm 0.181$	$+1.8$
	Logit head + cat. reparam.	1.02	1.50	$+0.481 \pm 0.008$	$+47.2$
	Total per NFE	19.08	19.89	$+0.804 \pm 0.186$	$+4.2$
$L' = 64$	Transformer forward	18.23	18.21	-0.021 ± 0.159	-0.1
	Logit head + cat. reparam.	1.08	1.60	$+0.519 \pm 0.006$	$+48.3$
	Total per NFE	19.31	19.80	$+0.498 \pm 0.162$	$+2.6$

We measure the wall-clock cost of the packed CFG forward with attention map (Fig. 4a) at batch size 1 compared with the unguided forward. We decode the same set of 16 GSM8K questions (8-shot, prompt length $729 \sim 814$) with the DDIM sampler at 32 steps per block. To keep the comparison exact, each run generates only the *first* block, so that every timed forward sees a KV cache which is precisely the prompt. Each question therefore requires $32 + 1$ forward passes (Alg. 2). Our implementation evaluates the mean embedding prediction $\mathbf{e}_\theta = \mathbf{x}_\theta\mathbf{E}$ on every step (including the final one at $t = 0$ which is the argmax readout of the logits that does not require mean embedding prediction), so wall-clock cost is not diluted by that step. The very first denoiser query of each process involves a one-time CUDA initialization which takes a longer time, so is excluded to avoid biasing the measurement. All runs are performed on one H100, using *Sigma*-3B with precision `BF16`.

Tab. 20 confirms our claim. At $L' = 32$ the packed transformer forward costs $+1.8\%$ over the unguided one, and at $L' = 64$ it is free to within measurement noise. The per-NFE overhead is dominated by computing the second set of logits, doing softmax, and multiplying by the embedding matrix rather than by attention.

G.11 Proofs

[Back to Sec. 4.2 in the main paper.]

Proposition 1 (Guidance and temperature commute)**.** *Applying* (ST) *to each branch and then* (CFG) *yields the same mean prediction as applying* (CFG) *before* (ST)*,*

$$\mathbf{e}_\theta^{w,\tau}(\mathbf{z}_t^{(b)}, t; \mathbf{c}) := \frac{w}{\tau}\mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c}) + \frac{1-w}{\tau}\mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \varnothing) + \Big(1 - \frac{1}{\tau}\Big)\frac{\mathbf{z}_t^{(b)}}{\alpha_t}. \tag{28}$$

The associated score is that of $p_t(\mathbf{z}_t^{(b)} \mid \mathbf{c})^{\frac{w}{\tau}}\, p_t(\mathbf{z}_t^{(b)} \mid \varnothing)^{\frac{1-w}{\tau}}$.

Proof. Write $\mathbf{e}_{\mathrm{c}} := \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \mathbf{c})$, $\mathbf{e}_\varnothing := \mathbf{e}_\theta(\mathbf{z}_t^{(b)}, t; \varnothing)$, and $\mu_t := \frac{\mathbf{z}_t^{(b)}}{\alpha_t}$.

Rearranging (CFG) and (ST), we have the following affine maps for CFG and ST, respectively:

$$G(\mathbf{e}_{\mathrm{c}}, \mathbf{e}_\varnothing) := \mu_t + \big[w(\mathbf{e}_{\mathrm{c}} - \mu_t) + (1-w)(\mathbf{e}_\varnothing - \mu_t)\big], \qquad T(\mathbf{e}) := \mu_t + \frac{1}{\tau}(\mathbf{e} - \mu_t). \tag{29}$$

Note that $G(\mu_t, \mu_t) = \mu_t$ and $T(\mu_t) = \mu_t$. Therefore,

$$\begin{aligned} T(G(\mathbf{e}_{\mathrm{c}}, \mathbf{e}_\varnothing)) &= \mu_t + \frac{1}{\tau}\big[w(\mathbf{e}_{\mathrm{c}} - \mu_t) + (1-w)(\mathbf{e}_\varnothing - \mu_t)\big] \\ &= w\big[\mu_t + \frac{1}{\tau}(\mathbf{e}_{\mathrm{c}} - \mu_t)\big] + (1-w)\big[\mu_t + \frac{1}{\tau}(\mathbf{e}_\varnothing - \mu_t)\big] \\ &= w\,T(\mathbf{e}_{\mathrm{c}}) + (1-w)\,T(\mathbf{e}_\varnothing) \;=\; G(T(\mathbf{e}_{\mathrm{c}}), T(\mathbf{e}_\varnothing)), \end{aligned}$$

so CFG and ST commute. Expanding the middle line recovers (28).

Recall from (1) that $\mathbf{z}_t^{(b)} \mid \mathbf{e}^{(b)} \sim \mathcal{N}(\alpha_t \mathbf{e}^{(b)}, \sigma_t^2 \mathbf{I})$ is a Gaussian transition kernel. Tweedie's formula (Robbins, 1992) gives an affine correspondence between any mean prediction $\mathbf{e}$ and the marginal score $\mathbf{s} := \nabla_{\mathbf{z}_t^{(b)}} \log p_t(\mathbf{z}_t^{(b)} \mid \cdot)$:

$$\mathbf{s} = \frac{\alpha_t}{\sigma_t^2}\big(\mathbf{e} - \mu_t\big), \qquad \mathbf{e} = \mu_t + \frac{\sigma_t^2}{\alpha_t}\,\mathbf{s}.$$

Therefore, taking an affine combination of mean predictions in (29) is the same as taking the same affine combination of the corresponding scores, and scaling the offset by $\frac{1}{\tau}$ is the same as scaling the score by $\frac{1}{\tau}$. Write $\mathbf{s}_{\mathrm{c}} := \nabla_{\mathbf{z}_t^{(b)}} \log p_t(\mathbf{z}_t^{(b)} \mid \mathbf{c})$ and $\mathbf{s}_\varnothing$ analogously; then the score of (28) is

$$\begin{aligned} \frac{1}{\tau}\big(w\,\mathbf{s}_{\mathrm{c}} + (1-w)\,\mathbf{s}_\varnothing\big) &= \frac{w}{\tau}\nabla_{\mathbf{z}_t^{(b)}} \log p_t(\mathbf{z}_t^{(b)} \mid \mathbf{c}) + \frac{1-w}{\tau}\nabla_{\mathbf{z}_t^{(b)}} \log p_t(\mathbf{z}_t^{(b)} \mid \varnothing) \\ &= \nabla_{\mathbf{z}_t^{(b)}} \log\big(p_t(\mathbf{z}_t^{(b)} \mid \mathbf{c})^{\frac{w}{\tau}}\, p_t(\mathbf{z}_t^{(b)} \mid \varnothing)^{\frac{1-w}{\tau}}\big), \end{aligned}$$

and hence the claim on the associated score is proved. □

Proposition 2 (EDM-ρ schedule converges to uniform-γ schedule when $\rho \to \infty$)**.** *Fix* $N \geq 1$ *and endpoints* $0 < \tilde{\sigma}_{\min} < \tilde{\sigma}_{\max}$*, where* $\tilde{\sigma}_t = \frac{\sigma_t}{\alpha_t}$ *and the VP noise schedule follows Sec. 2. For every* $0 \leq i \leq N$*, the EDM-*ρ *and uniform-*γ *grid points of App. G.6 satisfy*

$$\tilde{\sigma}_{t_i}^{\mathit{EDM}(\rho)} := (\tilde{\sigma}_{\max}^{1/\rho} + \tfrac{i}{N}(\tilde{\sigma}_{\min}^{1/\rho} - \tilde{\sigma}_{\max}^{1/\rho}))^\rho \xrightarrow[\rho\to\infty]{} \tilde{\sigma}_{\max}^{1-i/N}\tilde{\sigma}_{\min}^{i/N} =: \tilde{\sigma}_{t_i}^{\mathit{uniform}\text{-}\gamma}. \tag{30}$$

In other words, the EDM-ρ *discretization converges pointwise to the uniform-*γ *discretization when* $\rho \to \infty$.

Proof. The right-hand side of (30) is exactly the uniform-γ grid on $[\tilde{\sigma}_{\min}, \tilde{\sigma}_{\max}]$: because $\tilde{\sigma}_t^2 = \frac{\sigma_t^2}{\alpha_t^2} = \frac{1}{\mathrm{SNR}(t)} = \mathrm{e}^{\gamma(t)}$, the definition $\gamma(t_i) = \gamma_{\max} + \frac{i}{N}(\gamma_{\min} - \gamma_{\max})$ is equivalent to $\log \tilde{\sigma}_{t_i}$ being the arithmetic progression $(1 - \frac{i}{N})\log\tilde{\sigma}_{\max} + \frac{i}{N}\log\tilde{\sigma}_{\min}$, which exponentiates to $\tilde{\sigma}_{\max}^{1-i/N}\,\tilde{\sigma}_{\min}^{i/N}$.

To prove the limit, let $u := \frac{i}{N} \in [0,1]$, $p := \frac{1}{\rho} > 0$, $a := \tilde{\sigma}_{\max}$, $b := \tilde{\sigma}_{\min}$. The EDM-$\rho$ grid point is the weighted power mean

$$\tilde{\sigma}_{t_i}^{\mathrm{EDM}(\rho)} = ((1-u)a^p + ub^p)^{1/p},$$

so it suffices to show that this converges to the weighted geometric mean $a^{1-u}b^u$ as $p \searrow 0$. Taking $\log$ and performing Taylor expansion $a^p = 1 + p \log a + O(p^2)$ around $p = 0$, we have

$$(1-u)a^p + ub^p = 1 + p((1-u)\log a + u \log b) + O(p^2),$$
$$\implies \log((1-u)a^p + ub^p) = p((1-u)\log a + u \log b) + O(p^2),$$

Dividing by p,

$$\log \tilde{\sigma}_{t_i}^{\mathrm{EDM}(\rho)} = (1-u)\log a + u \log b + O(p) \to \log(a^{1-u}b^u),$$

which is exactly the $\log$ of the right-hand side of (30). □

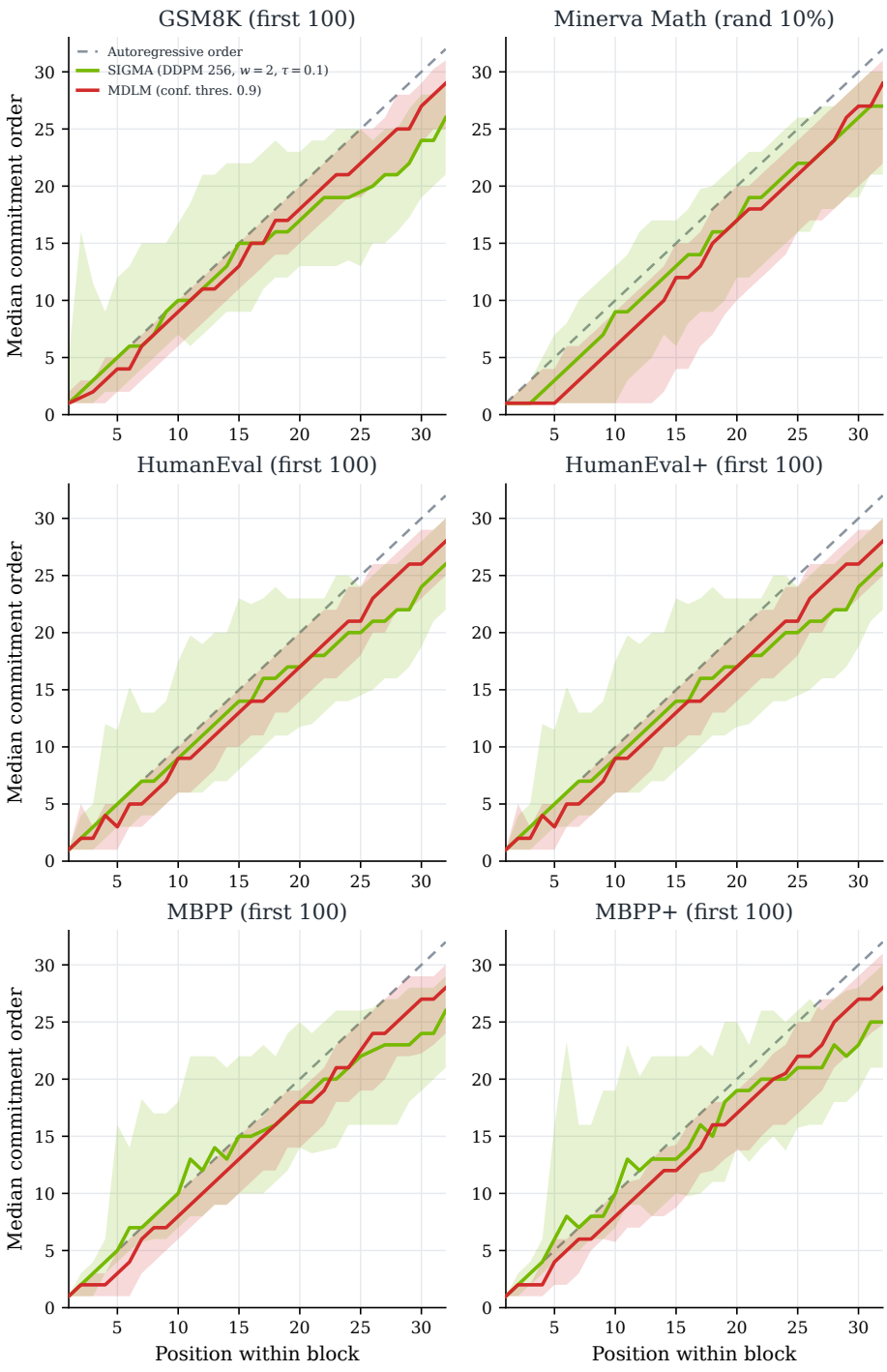


Figure 20: ***Sigma* tend to commit early positions of a block later than MDLM and the final positions earlier.** The sampler used is 256-step DDPM ($w = 2, \tau = 0.1$) as in Tab. 1. Generation order averaged across multiple documents and blocks. For *Sigma*, the commitment step t_i of position i is defined as the smallest t such that for every step $t' \geq t$ up to the final readout, (i) the argmax of $\mathbf{x}_\theta$ equals the finally decoded token, and (ii) that argmax is assigned probability at least 0.9. The commitment order of position i is the rank of t_i among $t_1, \ldots, t_{32}$ within its block, in ascending order. Positions sharing a commitment step share the lowest rank of the tie.

H Implementation Details and Additional Results for Supervised Fine-Tuning

[Back to Sec. 6 in the main paper.]

H.1 Supervised Fine-Tuning Setup

Our SFT pipeline follows the joint diffusion–AR design of Nemotron-Labs-Diffusion (NLD) (Fu et al., 2026): the prompt remains clean, supervision is restricted to the response, and the AR and diffusion objectives retain coefficients 1.0 and 0.3, respectively, following the training objective in Sec. 3.2.

Initialization and objective. We initialize from the 150K-step pretrained *Sigma*-{3B, 8B} checkpoints, which have block length $L' = 32$, separate diffusion and AR readout heads, and self-conditioning. We fine-tune the full transformer and both heads. Let $\mathcal{A}$ denote the supervised answer positions, including the terminal EOS padding described below. The SFT objective is

$$\mathcal{L}_{\text{sft}} = 0.3\,\frac{\sum_{i\in\mathcal{A}}\mathcal{L}_{\text{BD},i}}{|\mathcal{A}|} + \frac{\sum_{i\in\mathcal{A}}\mathcal{L}_{\text{ar},i}}{|\mathcal{A}|}, \tag{31}$$

where $\mathcal{L}_{\text{BD}}$ is the continuous block NELBO of (3) and (4), including its diffusion, reconstruction, and prior terms, and $\mathcal{L}_{\text{ar}}$ is next-token cross-entropy from the clean stream (6). Each numerator and denominator is summed globally over data-parallel ranks before the two terms are combined. The implementation maintains separate denominators for the two objectives; they are equal in this setup because position zero is an unsupervised system token and both objectives otherwise use the same answer mask.

As in pre-training (Sec. 3.2), we estimate the reconstruction and diffusion terms with the Horvitz–Thompson estimator in (5): independently for each block, we select reconstruction at $t = 0$ with probability $p_{\text{rec}} = 0.1$ and otherwise draw $t \sim \text{Unif}[0, 1]$. Gaussian noise is applied only in embedding space; the token IDs always remain clean. Prompt and padding positions have zero loss and are pinned to their clean embeddings; answer blocks see the clean context $\mathbf{c} = (\mathbf{p}, \mathbf{r}^{(<b)})$ of Sec. 4.1.

Data and preprocessing. Following NLD, our controlled experiment uses the same single-turn SFT dataset of Chandiramani et al. (2026). Every data sample has system, user, and assistant turns; the system message is empty. The corpus is the non-thinking variant: responses either begin with an empty `<think></think>` field (92.1% of the samples) or does not have think tags at all. We use the same Mistral-NeMo (Mistral AI, 2024) tokenizer ($V = 131{,}072$) and the Nemotron Nano chat template (Blakeman et al., 2025). One key difference between *Sigma* (continuous diffusion) and NLD (masked diffusion) is that *Sigma* requires one well-defined diffusion time per block, and thus cannot place prompt and answer tokens in the same noisy block. Therefore, during training, we *left-pad* each prompt by at most 31 unsupervised tokens so that the answer begins on the absolute 32-token block grid. We do not insert padding between prompt and answer, and inference does not pad the prompt either due to fully causal attention for the prompt. After the answer, supervised EOS tokens complete its final 32-token block; the batch is then right-padded with unsupervised PAD tokens to a multiple of 1024. This coarser batch alignment limits the number of compiled FlexAttention sequence shapes without changing the block factorization. Main SFT optimization configurations are listed in Tab. 21, across all model sizes.

Frozen-special-parameter default and ablation. In the main SFT run, we freeze the learned noise-schedule network $\tilde{\gamma}_\psi(t)$ and its endpoints γ_0, γ_1, as well as the diffusion embedding matrix $\mathbf{E}$. This freezes only the continuous-diffusion geometry; the transformer, input and conditioning projections, self-conditioning projection, and both output heads remain trainable at the base learning rate. Our ablation unfreezes the schedule, endpoints, and diffusion embedding matrix while holding all other settings fixed. Following pretraining (App. E), these parameters use a $1000\times$ learning-rate multiplier, giving a peak rate of 2.5×10^{-3} under the same scheduler; weight decay is zero for the schedule network and diffusion embedding matrix, while the two scalar endpoints retain the base weight decay of 0.1.

Table 21: SFT optimization settings for *Sigma*.

Setting	Value
Sequence length / block length	4096 / 32
Global / micro batch size	256 / 1 sequence
Training duration	8000 steps (≈ 2.4 epochs)
Optimizer	AdamW, $(\beta_1, \beta_2) = (0.9, 0.98)$, $\epsilon = 10^{-5}$
Learning rate / weight decay	2.5×10^{-6} / 0.1
Schedule	50-step linear warmup, then constant
Gradient clipping / precision	1.0 / `BF16`
Self-conditioning	25% of microbatches each step following pretraining
Parallelism	FSDP2; TP = CP = 1; no sequence packing

H.2 Evaluation Setup and Frozen-Parameter Ablation

The instruct evaluations in Tabs. 2 and 3 share the diffusion decoder, answer extraction, and seed-error-bar procedure of the base-model harness (App. G.1); we describe here only what differs at the instruct operating point.

Benchmarks. We evaluate on ten tasks (numbers in brackets represent the number of problems): GSM8K (1319), MATH-500 (500), AIME 2024 and AIME 2025 (30 each), HumanEval (164), MBPP (378), LiveCodeBench-CPP (454), MMLU (14042), GPQA (198), and IFEval (541). Prompts follow the Nemotron Nano chat template used in SFT training, and the assistant turn is generated in diffusion mode with block length $L' = 32$.

Scoring. Math tasks (GSM8K, MATH-500, AIME) are scored by symbolic equivalence on the final `\boxed{}` answer; GSM8K additionally accepts the last number in the completion. HumanEval and MBPP report executed pass@1 on the base test suites (not the + variants). LiveCodeBench-CPP compiles and runs each extracted C++ program against its tests. GPQA and MMLU are scored by the extracted multiple-choice letter; both are prompted for free-form reasoning ending in `Answer:` `\boxed{A/B/C/D}`, and the grader reads the *last* `\boxed{}` in the completion, falling back to a regular expression when the model emits none. IFEval reports the instruction-level loose-following rate from the `instruction_following_eval` checkers, which is the metric the reference models in Tabs. 2 and 3 report.

Baselines. All baseline numbers in Tabs. 2 and 3 are copied from Fu et al. (2026, Tabs. 5, 7), except that NLD-8B's diffusion mode results are our reproduction under aligned protocol (without format-injection tricks). Fu et al. (2026) report SDAR-8B Chat at two confidence-threshold operating points (TPF = 1.00 and TPF = 1.75); we use the former as it is stronger on 8 of the 10 tasks (the two exceptions, MMLU and HumanEval, differ by ≤ 0.6 points).

Decoding algorithms. We report two operating points, both selected on the base model (Tab. 1) and both with self-conditioning enabled. The DDPM-256 rows use the ancestral sampler on the EDM $\rho = 7$ schedule with guidance $w = 2$ and score temperature $\tau = 0.1$; the DDIM-32 rows use the deterministic sampler on the uniform-γ schedule with $w = 3$ and $\tau = 0.3$.

Generation budget. Unlike in Fu et al. (2026) who used the same 16834 budget for all tasks, the maximum number of generated tokens for *Sigma* is set per benchmark rather than uniformly: 1536 for GSM8K, 2560 for HumanEval, MBPP, LiveCodeBench-CPP, and IFEval, 2560 for MMLU, and 4096 for AIME 2024, AIME 2025, MATH-500, and GPQA. These budgets are sized to the per-lane graded-answer-length distribution: on our small-scale smoke runs, no correct answer used a budget beyond 4096, and the long tail of length is dominated by degenerate repetition that a larger budget does not convert into correct answers.

Seed error bars. As in the base-model harness, each row draws 10 independent sampling lanes, averages over questions within each lane to obtain 10 per-lane scores, and reports the mean with half-width $\frac{2}{\sqrt{10}}\hat{\sigma}$, where $\hat{\sigma}$ is the across-lane standard deviation.

Frozen-versus-unfrozen ablation. Tab. 22 compares the frozen default of Tab. 2 against the unfrozen variant of the previous paragraph, which trains the noise schedule, its γ_0, γ_1 endpoints, and the diffusion embedding matrix at a $1000\times$ learning-rate multiplier as in pretraining (App. E). We run the

comparison at both sampler operating points of Tab. 2. Under DDPM-256, the two configurations are within the seed interval on seven of ten tasks; under DDIM-32, the ordering of the average reverses, with the unfrozen model ahead. As the two configurations are close in performance in general, we keep the frozen configuration as the simpler default for 8B.

Table 22: **Freezing the continuous-diffusion geometry during SFT.** 3B instruct performance, mean ± two standard errors over 10 sampling lanes, under both sampler operating points of Tab. 2. *Frozen* is the default of Tab. 2: the noise schedule, its endpoints, and the diffusion embedding matrix are held fixed. *Unfrozen* trains all three at a 1000× learning-rate multiplier. **Avg.** is the unweighted mean over the ten tasks. The frozen/unfrozen ordering of the average is not stable across the two samplers, and in general the gap is within the seed interval.

	Math				**Coding**			**Multiple Choice**		**IF**	
Configuration	GSM8K	Math500	AIME24	AIME25	HumanEval	MBPP	LCB-CPP	MMLU	GPQA	IFEval	Avg.
DDPM (256 steps, w=2, τ=0.1)											
Frozen (default)	89.18 ± 0.24	70.44 ± 0.79	17.00 ± 3.05	11.00 ± 2.12	77.62 ± 1.31	73.10 ± 0.75	11.17 ± 0.28	71.28 ± 0.08	34.70 ± 1.14	66.04 ± 0.60	52.15
Unfrozen (1000× LR)	88.39 ± 0.21	72.62 ± 0.88	13.00 ± 2.57	11.67 ± 2.36	75.61 ± 1.42	72.33 ± 0.92	11.37 ± 0.40	72.00 ± 0.15	33.28 ± 1.27	66.22 ± 0.37	51.65
DDIM (32 steps, w=3, τ=0.3)											
Frozen (default)	87.86 ± 0.29	68.30 ± 0.67	9.67 ± 2.20	14.00 ± 2.46	72.74 ± 1.22	68.84 ± 0.78	10.11 ± 0.31	69.15 ± 0.11	31.82 ± 1.82	66.09 ± 0.22	49.86
Unfrozen (1000× LR)	87.29 ± 0.33	68.64 ± 0.61	14.00 ± 3.63	15.00 ± 2.16	72.26 ± 1.11	70.29 ± 0.62	9.47 ± 0.54	70.44 ± 0.14	33.69 ± 1.55	67.35 ± 0.29	50.84

I Implementation Details and Additional Results for Parallel Decoding Distillation

[Back to Sec. 5.3 in the main paper.]

I.1 Parallel Predictions and Intra-Segment Rollout

PDD reduces the number of expensive denoiser evaluations without discarding the teacher's fine integration grid. Following Shaul et al. (2026), we let $1 = t_0 > t_1 > \cdots > t_G = 0$ be a decreasing time grid. A conventional DDIM sampler queries the denoiser once per interval and therefore requires G NFEs. For a target budget $M < G$, PDD instead partitions the grid into M *segments*,[4] each spanning $\Delta = G/M$ fine intervals. At the start of segment k, where $i_k = k\Delta$, the student evaluates its transformer backbone once and emits Δ raw clean-embedding predictions required within that segment:

$$\mathbf{e}_\phi(\mathbf{z}_{i_k}, t_{i_k}; \mathbf{x}^{(<b)}) = \{\hat{\mathbf{e}}^{S,\text{raw}}_{i_k,j}\}_{j=0}^{\Delta-1}. \tag{32}$$

Here j is the local offset within the segment, while the corresponding prediction head is indexed by the absolute fine-grid position $i_k + j$; we describe this parameterization in App. I.2. All Δ outputs share the same backbone representation and differ only through lightweight output heads. Because these heads are evaluated in parallel, they cannot consume recurrent self-conditioning from earlier local nodes; we therefore *disable self-conditioning* for both teacher and student during distillation.

Starting from the segment input, these predictions are then composed through the ordinary DDIM recursion (DDIM):

$$\begin{aligned} \tilde{\mathbf{z}}_{i_k,0} &= \mathbf{z}_{i_k}, \\ \tilde{\mathbf{z}}_{i_k,j+1} &= \alpha_{i_k,j+1}\hat{\mathbf{e}}^{S}_{i_k,j} + \sigma_{i_k,j+1} \cdot \frac{\tilde{\mathbf{z}}_{i_k,j} - \alpha_{i_k,j}\hat{\mathbf{e}}^{S}_{i_k,j}}{\sigma_{i_k,j}}, \; j = 0, 1, ..., \Delta - 1, \end{aligned} \tag{33}$$

where $\alpha_{i_k,j} = \alpha_{t_{i_k+j}}$, $\sigma_{i_k,j} = \sigma_{t_{i_k+j}}$, and $\hat{\mathbf{e}}^{S}_{i_k,j}$ denotes the guided student prediction, constructed as follows. The raw prediction in (32) includes the CFG behavior distilled from the teacher but precedes score temperature (ST). Before each solver update, we apply ST analytically using the state $\tilde{\mathbf{z}}_{i_k,j}$ reached at that local node according to the local DDIM rollout (33),

$$\hat{\mathbf{e}}^{S}_{i_k,j} = \frac{1}{\tau}\hat{\mathbf{e}}^{S,\text{raw}}_{i_k,j} + \left(1 - \frac{1}{\tau}\right)\frac{\tilde{\mathbf{z}}_{i_k,j}}{\alpha_{i_k,j}}. \tag{34}$$

Thus, PDD parallelizes the expensive denoiser predictions, not the inexpensive state updates within a segment. Only the DDIM endpoint $\tilde{\mathbf{z}}_{i_k,\Delta}$ is passed to the next transformer evaluation as $\mathbf{z}_{i_{k+1}}$.

I.2 Residual Parallel Heads

Learning to predict an independent clean embedding at every fine-grid node would discard the strong prediction already produced by the pretrained denoiser at each segment start. We instead retain the original diffusion readout as an anchor and parameterize every parallel prediction as a residual transformation of it. Following Shaul et al. (2026), each residual head is associated with an absolute fine-grid index. Segment k therefore selects

$$R_{i_k}, R_{i_k+1}, \ldots, R_{i_k+\Delta-1}. \tag{35}$$

Let $H_\phi(\mathbf{z}_{i_k}, t_{i_k}; \mathbf{x}^{(<b)})$ be the shared transformer representation from (32). Its original diffusion readout, which serves as the anchor, is

$$\hat{\mathbf{e}}^{S,\text{anchor}}_{i_k} = \text{softmax}(W_{\text{out}} H_\phi(\mathbf{z}_{i_k}, t_{i_k}; \mathbf{x}^{(<b)}) + \boldsymbol{\ell}^{\text{prior}}(\mathbf{z}_{i_k}, t_{i_k}))\mathbf{E}, \tag{36}$$

where W_{out} is the vocabulary projection and $\boldsymbol{\ell}^{\text{prior}}(\mathbf{z}_{i_k}, t_{i_k})$ denotes the output-prior logits. For local offset j, the corresponding absolute head produces

$$\begin{aligned} (\mathbf{s}_{i_k,j}, \mathbf{r}_{i_k,j}) &= R_{i_k+j}(H_\phi(\mathbf{z}_{i_k}, t_{i_k}; \mathbf{x}^{(<b)})), \\ \mathbf{q}_{i_k,j} &= \exp(2\tanh(\mathbf{s}_{i_k,j})), \\ \hat{\mathbf{e}}^{S,\text{raw}}_{i_k,j} &= \mathbf{q}_{i_k,j} \odot \hat{\mathbf{e}}^{S,\text{anchor}}_{i_k} + \mathbf{r}_{i_k,j}. \end{aligned} \tag{37}$$

[4] We use *segment* rather than the term *block* from Shaul et al. (2026) to avoid confusion with the language-model diffusion blocks.

Each R_i is an embedding-space MLP whose final layer is initialized at zero, so every parallel head initially reproduces the pretrained anchor prediction.

The student therefore begins from the pretrained denoiser rather than from randomly initialized parallel predictions, and each residual head only needs to model how the clean prediction evolves along the local trajectory. The additive residual and scaling also allow predictions to leave the convex hull of all token embeddings, which is necessary for matching extrapolated CFG-guided teacher predictions.

I.3 Distillation Objective and Optimization

Throughout this subsection, every teacher evaluation is treated as a fixed target: the teacher parameters are frozen, and any student-generated input state is detached at the teacher-query boundary. We omit stop-gradient operators from the notation for simplicity. In contrast, the student evaluation and its intra-segment rollout are constructed with gradients enabled.

We distill CFG into the conditional student, while leaving ST as the analytic transformation in (34). At a student-visited state $\tilde{\mathbf{z}}_{i_k,j}$, the frozen teacher's CFG-guided prediction before ST is

$$\hat{\mathbf{e}}^{T,\text{raw}}_{i_k,j} = \mathbf{e}_T(\tilde{\mathbf{z}}_{i_k,j}, t_{i_k,j}; \varnothing) + w[\mathbf{e}_T(\tilde{\mathbf{z}}_{i_k,j}, t_{i_k,j}; \mathbf{x}^{(<b)}) - \mathbf{e}_T(\tilde{\mathbf{z}}_{i_k,j}, t_{i_k,j}; \varnothing)]. \tag{38}$$

The student receives only the conditional prefix $\mathbf{x}^{(<b)}$, so matching this target amortizes the teacher's conditional and unconditional evaluations into a single student call. Crucially, both $\hat{\mathbf{e}}^{S,\text{raw}}_{i_k,j}$ and $\hat{\mathbf{e}}^{T,\text{raw}}_{i_k,j}$ denote predictions *before* ST; ST affects the states visited by the local rollout but is not part of the regression target.

For each language-model diffusion block b, we sample a segment start t_{i_k} uniformly from the candidate starts and draw $\mathbf{z}_{i_k}$ from the VP forward marginal conditioned on the clean data embedding. One student evaluation produces the parallel predictions in (32), from which we construct the differentiable local rollout in (33). We always supervise the first local node, $j_1 = 0$, to anchor the segment-start prediction, and sample the remaining $P-1$ indices $\{j_2, \dots, j_P\}$ uniformly without replacement from $\{1, \dots, \Delta - 1\}$. The PDD objective compares the raw student and CFG-guided teacher predictions at these student-visited states:

$$\mathcal{L}_{\text{PDD}} = \frac{1}{P} \sum_{p=1}^{P} \mathbb{E}[\|\hat{\mathbf{e}}^{S,\text{raw}}_{i_k,j_p} - \hat{\mathbf{e}}^{T,\text{raw}}_{i_k,j_p}\|^2]. \tag{39}$$

This construction is intra-segment on-policy: although the segment start is sampled from the forward marginal, all later supervised states within the segment are generated by the current student. The teacher therefore corrects the student where it locally visits, rather than only on teacher or forward-process states.

Because trajectory distillation supervises only the trajectory and only on a sparse set of guided states, we additionally match the teacher's categorical reconstruction and autoregressive distributions to preserve the pretrained model's behavior. We use forward KL divergences on the corresponding logits:

$$\mathcal{L}_{\text{teacher-rec}} = \mathbb{E}_{\mathbf{z}_0^{(b)} \sim q(\mathbf{z}_0^{(b)} | \mathbf{x}^{(b)})} \sum_{l=1}^{L'} \text{KL}(\mathbf{x}^l_T(\mathbf{z}_0^{(b)}, 0; \mathbf{x}^{(<b)}) \,\|\, \mathbf{x}^l_\phi(\mathbf{z}_0^{(b)}, 0; \mathbf{x}^{(<b)})). \tag{40}$$

$$\mathcal{L}_{\text{teacher-ar}} = \sum_{l=1}^{L'} \text{KL}(\mathbf{x}^l_{T^T_{\text{AR}}}(\mathbf{x}^{(b,<l)}; \mathbf{x}^{(<b)}) \,\|\, \mathbf{x}^l_{\phi_{\text{AR}}}(\mathbf{x}^{(b,<l)}; \mathbf{x}^{(<b)})). \tag{41}$$

By construction, these losses are zero as distillation begins. Together, this yields the combined objective:

$$\mathcal{L}_{\text{distill}} = \lambda_{\text{PDD}} \mathcal{L}_{\text{PDD}} + \lambda_{\text{teacher-rec}} \mathcal{L}_{\text{teacher-rec}} + \lambda_{\text{teacher-ar}} \mathcal{L}_{\text{teacher-ar}}, \tag{42}$$

Teacher and student are initialized from the same checkpoint, after which the teacher remains frozen. We also freeze the embedding matrix and noise schedule so that the teacher trajectory retains a fixed meaning throughout distillation. The student transformer is adapted through LoRA, while the residual heads are optimized directly with a separate learning rate. This separation allows the new heads to learn local corrections without destabilizing the pretrained backbone or diffusion calibration. Full training details and hyperparameters are given in App. I.5.

I.4 Few-Step Inference

For a requested NFE budget M, we evaluate the student at M segment starts in sequence. At segment k, the model uses the absolute heads $R_{i_k}, \ldots, R_{i_k+\Delta_M-1}$, where $\Delta_M = G/M$, and advances the state through (33) and (34). Only the segment endpoint is passed to the next transformer evaluation. The student therefore reduces the sequential neural-network depth from G evaluations to M, while retaining fine-grid DDIM integration between evaluations.

I.5 Training and Parameterization Details

Training details. We initialize the student from base *Sigma*-3B (with or without SUSReg regularization, ablation results are shown in Fig. 7), which also serves as the frozen distillation teacher. The embedding lookup, vocabulary projection and noise schedule (including terminal points) are frozen. The transformer backbone is adapted with LoRA of rank 128 and scaling factor 256 using learning rate 10^{-6}. We use a global bank of 32 parallel residual heads parameterized by three-layer MLPs of width 384 with standard LayerNorm, which use a separate parameter group with learning rate 2.5×10^{-6}.

We use AdamW with weight decay 1.0, $\beta_1 = 0.9$, $\beta_2 = 0.999$ and a batch size of 256. For all parameters, we use gradient clipping value 1.0. The learning rate schedule is Warmup-Stable-Decay over 10K steps: 5% linear warmup to the nominal learning rate value, constant through step 8000, then exponential decay to 10% of nominal learning rate value. We use a fixed clean context of 512 tokens, and train on a signal block of 32 tokens. We noticed that this hardly impacts distillation results compared to training on all possible context lengths $\{0, 32, \ldots, 4064\}$ used for pretraining, while simplifying and significantly accelerating distillation. Our student is trained flexibly at $M_{\min} = 4$ and $M_{\max} = 8$ NFE.

For NFE-4 results in Fig. 7 in the main paper, the teacher trajectory uses classifier-free guidance $w = 4$ and score temperature $\tau = 0.4$ swhile for NFE-8 results, the best guidance setting with $w = 3$ and $\tau = 0.3$. These were the only guidance settings investigated and were informed by the self-conditioning-disabled teacher DDIM performance at 32 NFEs. We supervise $P = 8$ sampled downstream nodes per segment. In $\mathcal{L}_{\text{distill}}$ (42), the PDD objective component has weight $\lambda_{\text{PDD}} = 5.0$, the teacher-KL regularization on diffusion stream has weight $\lambda_{\text{teacher-rec}} = 0.1$ and teacher-KL regularization on the clean AR stream has weight $\lambda_{\text{teacher-ar}} = 0.3$. These weights were chosen to balance the respective gradient norms.

Flexible-NFE head sharing. Following Shaul et al. (2026), we train one student for a set of inference budgets

$$\mathcal{M} = \{M_{\min}, 2M_{\min}, \ldots, M_{\max}\}, \qquad M_{\max} = 2^a M_{\min}. \tag{43}$$

For budget $M \in \mathcal{M}$, one student evaluation advances $\Delta_M = G/M$ fine-grid intervals before the transformer is evaluated again. Because the residual heads are indexed by absolute fine-grid position, segment k selects the contiguous slice $R_{i_k}, \ldots, R_{i_k+\Delta_M-1}$. Changing M changes the segment boundaries and slice lengths while reusing the same backbone and bank of absolute heads. A single student can therefore be evaluated at several NFE budgets without an additional NFE embedding or a separately distilled model for each budget.

During training, NFE budgets are stratified across global microbatches so that all values in $\mathcal{M}$ receive comparable supervision. This exposes the model both to longer, more difficult intra-segment rollouts at low NFE and to shorter rollouts at higher NFE.

Additional results We synthesize the results in Fig. 7 in the main paper for GSM8K, HumanEval and MBPP. All-task results are displayed in Tab. 7. PDD achieves a clear 2x acceleration from NFE-16 to NFE-8 relative to its self-conditioning-disabled teacher. Compared to the self-conditioning-enabled *Sigma*-3B-SUSReg model, which obtains the best results for NFE>16, PDD achieves a 2× acceleration from NFE-8 to NFE-4, where it boasts a 68% GSM8K flexible-extract accuracy at NFE=4 while the non-distilled models achieve at most 32%. Coding is empirically more NFE-sensitive: PDD allows a more graceful decay as NFE decreases, but NFE-4 pass@1 still falls short of the performance at NFE>16.

Importantly, initializing the student and frozen teacher from a checkpoint regularized with SUSReg yields much better distillation performance: +4.67pp at NFE-8 and +3.71pp at NFE-4. This confirms the importance of regularizing the embedding geometry in such a few-NFE setup.

J EXAMPLE OUTPUTS

Below, we present example outputs from the pretrained and instruct 3B models to showcase the reasoning capabilities of the models across multiple tasks. We only make minimal modifications to the original outputs (e.g., correcting the unpaired LaTeX brackets that otherwise break the compilation, and changing markdown items to LaTeX itemize environment for better visual appearance).

J.1 *Sigma*-BASE-3B

GSM8K (8 shot): DDIM, 8 steps/block, CFG 2, ST 0.7

Q: There are 66 fish in the fish tank. One-third of the fish have red stripes, and 5/11 of the remaining fish have blue stripes. Altogether, how many fish have red stripes and blue stripes?
A: There are 66 fish originally. One-third of them have red stripes, which is 66 * (1/3) = 22 fish. That leaves 66 - 22 = 44 fish remaining. 5/11 of the remaining fish have blue stripes, which is 44 * (5/11) = 20 fish. So there are 22 + 20 = 42 fish with red or blue stripes. The answer is 42.

MBPP (3 shot): DDIM, 8 steps/block, CFG 2, ST 0.7

Q: Write a function to count all the distinct pairs having a difference of k in any array.
A:

```
def count_pairs(nums, n, k):
    count = 0
    pairs = set()
    for num in nums:
      if num + k in pairs:
        count += 1
      if num - k in pairs:
        count += 1
      pairs.add(num)
    return count
```

HumanEval (0 shot): DDIM, 32 steps/block, CFG 2, ST 0.7

Q:

```
from typing import List


def has_close_elements(numbers: List[float], threshold: float) -> bool:
    """ Check if in given list of numbers, are any two numbers closer to each other than
    given threshold.
    >>> has_close_elements([1.0, 2.0, 3.0], 0.5)
    False
    >>> has_close_elements([1.0, 2.8, 3.0, 4.0, 5.0, 2.0], 0.3)
    True
    """
```

A:

```
    for i in range(len(numbers)):
        for j in range(i + 1, len(numbers)):
            if abs(numbers[i] - numbers[j]) < threshold:
                return True
    return False
```

Minerva math (4 shot): DDIM, 32 steps/block, CFG 2, ST 0.7

Q: In right triangle ABC, shown below, $\cos B = \frac{6}{10}$. What is $\tan C$?

```
[asy]
draw((0,0)--(10,0)--(3.6,4.8)--cycle,black+linewidth(1));
draw(rightanglemark((0,0),(3.6,4.8),(10,0),20),black+linewidth(1));
label("$C$",(10,0),E);
label("$A$",(3.6,4.8),N);
label("$B$",(0,0),W);
label("10",(0,0)--(10,0),S);
[/asy]
```

A: In right triangle ABC with $\cos B = \frac{6}{10}$, we first determine $\sin B$ using the Pythagorean identity:

$$\sin^2 B = 1 - \cos^2 B = 1 - \left(\frac{6}{10}\right)^2 = 1 - \frac{36}{100} = \frac{64}{100}.$$

Taking the positive square root (since B is in the first quadrant), we get:

$$\sin B = \frac{8}{10} = \frac{4}{5}.$$

Next, $\tan B = \frac{\sin B}{\cos B} = \frac{\frac{4}{5}}{\frac{6}{10}} = \frac{4}{5} \times \frac{10}{6} = \frac{40}{30} = \frac{4}{3}$.
In a right triangle, $\tan C = \cot B = \frac{1}{\tan B} = \frac{1}{\frac{4}{3}} = \frac{3}{4}$.
Thus, the value of $\tan C$ is $\boxed{\frac{3}{4}}$.
Final Answer: The final answer is $\frac{3}{4}$. I hope it is correct.

J.2 *Sigma*-3B-INSTRUCT

AIME 24: DDIM, 32 steps/block, CFG 3, ST 0.3

Q: Jen enters a lottery by picking 4 distinct numbers from $S = \{1, 2, 3, \cdots, 9, 10\}$. 4 numbers are randomly chosen from S. She wins a prize if at least two of her numbers were 2 of the randomly chosen numbers, and wins the grand prize if all four of her numbers were the randomly chosen numbers. The probability of her winning the grand prize given that she won a prize is $\frac{m}{n}$ where m and n are relatively prime positive integers. Find $m + n$.
A: The probability of winning the grand prize given that Jen won a prize is calculated as follows.
Jen wins a prize if at least two of her chosen numbers are among the four randomly selected numbers. The grand prize is won if all four of her chosen numbers are among the four chosen numbers.
The total number of ways to choose four numbers from the set $S = \{1, 2, 3, \ldots, 10\}$ is $\binom{10}{4} = 210$.
The event that Jen wins a prize occurs when at least two of her four chosen numbers are in the four chosen numbers. The number of favorable outcomes for winning a prize is found by considering the cases where at least two of her four chosen numbers are included in the four chosen numbers. This is equivalent to the number of ways to choose the four chosen numbers such that at least two of Jen's four chosen numbers are in the four chosen numbers.
The number of ways to choose the four chosen numbers with at least two of Jen's four chosen numbers is calculated by summing the cases for exactly a numbers of Jen's numbers being included, where $a = 2, 3, 4, 4$:

- For $a = 2$: $\binom{4}{2} \times \binom{6}{2} = 6 \times 15 = 90$
- For $a = 3$: $\binom{4}{3} \times \binom{6}{1} = 4 \times 6 = 24$
- For $a = 4$: $\binom{4}{4} \times \binom{6}{0} = 1 \times 1 = 1$

Summing these gives the total number of outcomes where Jen wins a prize: $90 + 24 + 1 = 115$.
The event that Jen wins the grand prize occurs when all four of her chosen numbers are in the four chosen numbers, which happens only when $a = 4$. The number of such outcomes is 1.
The conditional probability of winning the grand prize given that she won a prize is the ratio of the number of outcomes for winning the grand prize to the number of outcomes for winning a prize, which is $\frac{1}{115}$.
The fraction $\frac{1}{115}$ is in simplest terms, as 1 and 115 are relatively prime. Thus, $m = 1$ and $n = 115$, and $m + n = 116$.
$\boxed{116}$

AIME 25: DDIM, 32 steps/block, CFG 3, ST 0.3

Q: Six points A, B, C, D, E, and F lie in a straight line in that order. Suppose that G is a point not on the line and that $AC = 26, BD = 22, CE = 31, DF = 33, AF = 73, CG = 40$, and $DG = 30$. Find the area of $\triangle BGE$.
A: The points A, B, C, D, E, and F lie on a straight line in that order. Assign coordinates to the points such that A is $a = 0$, B b, C c, D at d, E at e, and F at f, with $0 < b < c < d < e < f$.
Given:

- $AC = 26$, so $c = 26$.
- $AF = 73$, so $f = 73$.

- $BD = 22$, so $d - b = 22$.
- $CE = 31$, so $e - c = 31$, giving $e = 26 + 31 = 57$.
- $DF = 33$, so $f - d = 33$, giving $d = 73 - 33 = 40$.
- $CG = 40$ and $DG = 30$, where G is not on the line.

From $d = 40$ and $d - b = 22$, solve for $b = 40 - 22 = 18$. From $e = 57$ and $f = 73$, solve for $f = 73 - 57 = 16$. Thus, the coordinates are $A = 0$, $B = 18$, $C = 26$, $D = 40$, $E = 57$, $F = 73$.
Let G be at (x, y). The distances give:

- $CG = 40$: $(x - 26)^2 + y^2 = 1600$.
- $DG = 30$: $(x - 40)^2 + y^2 = 900$.

Subtract the equations:

$$(x - 26)^2 + y^2 - [(x - 40)^2 + y^2] = 1600 - 900$$
$$(x^2 - 52x + 676) - (x^2 - 80x + 1600) = 700$$
$$28x - 924 = 700$$
$$28x = 1624$$
$$x = \frac{1624}{28} = \frac{406}{7}.$$

Substitute $x = \frac{406}{7}$ into $(x - 40)^2 + y^2 = 900$:

$$\left(\frac{406}{7} - 40\right)^2 + y^2 = 900$$
$$\left(\frac{406}{7} - \frac{280}{7}\right)^2 + y^2 = 900$$
$$\left(\frac{126}{7}\right)^2 + y^2 = 900$$
$$\left(\frac{126}{7}\right)^2 = 18^2 = 324$$
$$324 + y^2 = 900$$
$$y^2 = 576$$
$$y = \pm 24 \quad \text{(since area depends on } |y|\text{)}.$$

The area of $\triangle BGE$ with vertices $B(18, 0)$, $G\left(\frac{406}{7}, y\right)$, and $E(57, 0)$ is found using the formula for the area of a triangle with base BE and height $|y|$:

- Base $BE = |57 - 18 = 39$.
- Height is $|y| = 24$.
- Area = $\frac{1}{2} \times 39 \times 24 = \frac{1}{2} \times 936 = 468$.

Alternatively, using the shoelace formula:

$$\text{Area} = \frac{1}{2}\left|18\,(y - 0) + \frac{406}{7}(0 - 0) + 57(0 - y)\right| = \frac{1}{2}\,|18y - 57y| = \frac{1}{2}|-39y| = \frac{1}{2}\times 39|y| = \frac{39}{2}\times 24 = 468,$$

since $|y| = 24$.
The base of $\triangle BGE$ is $|57 - 18| = 39$, and the height is $|y| = 24$, so the area is $\frac{1}{2} \times 39 \times 24 = 468$.
The area is consistent regardless of the sign of y, and all conditions are satisfied with $y = 24$ or $y = -24$.
Thus, the area of $\triangle BGE$ is 468.
$\boxed{468}$

GPQA (Quantum Mechanics, Hard graduate level): DDIM, 32 steps/block, CFG 3, ST 0.3

Q: A common approximation made in many-body nuclear calculations of some interaction (for example an electroweak interaction) is to suppose that nucleons in a nucleus behave in the same way as free nucleons. Which of the following four assumptions jointly imply this "impulse" approximation?

1. The interaction current only interacts with individual nucleons.
2. The nucleus is transparent apart from the selected nucleon.
3. The quarks internal to the selected nucleon are non-relativistic.

4. The interaction proceeds as if the selected nucleon experiences no binding forces.

A) 1, 2, 3
B) 1, 3, 4
C) 2, 3, 4
D) 1, 2, 4

A: The "impulse" approximation in many-body nuclear calculations assumes that the behavior of a selected nucleon in a nucleus is similar to that of a free nucleon. This approximation is based on the idea that the interaction on the selected nucleon is not significantly affected by the binding forces from the other particles in the nucleus.
To justify this approximation, the following assumptions must ensure that the interaction on the selected nucleon is not influenced by the binding forces from the other nucleons:

- **Assumption 1**: The interaction current only interacts with individual nucleons. This ensures that the interaction does not depend on the collective state of the nucleus but rather only on the individual nucleons.
- **Assumption 2**: The nucleus is transparent apart from the selected nucleon. This means that the interaction on the selected nucleon is not altered by the presence of the other nucleons, allowing it to be treated as a free nucleon.
- **Assumption 4**: The interaction proceeds as if the selected nucleon experiences no binding forces. This directly states that the binding forces from other nucleons are ignored, allowing the selected nucleon to behave like a free nucleon.
- **Assumption 3** ("quarks internal to the selected nucleon are non-relativistic") is not relevant to this approximation, as it pertains to the internal structure of the nucleon, not the external environment in the nucleus.

Thus, the assumptions that jointly imply the impulse approximation are **1, 2, and 4**.
Answer: D