REASONING WITH CONTINUOUS LATENT DIFFUSION

Anonymous authors
Paper under double-blind review

ABSTRACT

Continuous diffusion generates complete reasoning solutions through iterative refinement in latent space. We introduce Latent Flow Reasoning Models (LFRMs), an ELF-based training and inference recipe. Our experiments show that accurate decoding alone does not ensure strong reasoning performance. We therefore learn compact representations from multiple layers of a strong autoregressive teacher. Their decomposition also enables asynchronous denoising at different rates. We show that prompt encodings need only preserve the information required for the correct text-conditional score, rather than exactly match teacher features, and use a staged curriculum to learn a compact prompt encoder that replaces the teacher Transformer at inference. We adapt DiffusionNFT to learned self-conditioning guidance and incorporate gold-solution endpoints to supplement sparse rewards. With 90M and 638M denoising backbones, our supervised models outperform reported continuous-diffusion baselines at comparable backbone scales on mathematical reasoning and HumanEval code generation. NFT further improves single-sample accuracy on math and code, and majority-vote performance on mathematics. Code will be available at: `https://anonymous.4open.science/r/LFRM-98B5/`.

1 INTRODUCTION

Autoregressive language models have made substantial progress on mathematical reasoning (Lewkowycz et al., 2022; DeepSeek-AI et al., 2025). Continuous diffusion offers a complementary approach, generating complete solutions through iterative refinement of all answer positions. Whereas autoregressive and discrete-diffusion language models typically generate in token space, continuous latent diffusion makes the denoising representation an additional design choice (Hu et al., 2026; Nie et al., 2025). Reasoning requires this representation to preserve exact quantities and dependencies between deductions. We therefore ask: *what representation makes complex reasoning solutions amenable to generation through denoising?* Token representations span a spectrum of complexity: tokenwise embedding tables, contextual encodings from bidirectional networks such as T5, and hidden activations of powerful autoregressive models (Gulrajani & Hashimoto, 2023; Raffel et al., 2019; Yang et al., 2025). We find that strong clean-token recovery can coexist with weak reasoning generation, motivating representation design for both decoding and generation (Figure 2; Appendix C.2).

We introduce Latent Flow Reasoning Models (LFRMs), based on the simple continuous flow formulation of Embedded Language Flows (ELF; Hu et al., 2026). Our pipeline learns compact answer representations from multiple layers of a strong autoregressive teacher (Figure 1). To avoid retaining the teacher for prompt conditioning, we separate the answer-generation target from the prompt encoding. Conditioning on prompt *text* makes that encoding an internal, learnable interface: it need only preserve the information required for denoising, without exact teacher-feature matching (Lemma 1). We use this flexibility to jointly train a compact prompt encoder and denoiser through the generative objective.

Remaining close to standard continuous diffusion lets us apply DiffusionNFT (Zheng et al., 2025) with verifiable rewards, after adapting it to ELF's learned guidance. Our supervised models outperform comparable-scale continuous-diffusion baselines on mathematical reasoning and HumanEval code generation; NFT further improves accuracy on both math and code.

Contributions. Our complete reasoning pipeline comprises the following key contributions:

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.

1. **A complete recipe for competitive continuous-diffusion reasoning.** We develop an LFRM pipeline spanning representation learning, conditional flow training, and inference (Figure 1). Our models demonstrate strong performance on math and coding tasks. Our supervised models outperform reported continuous-diffusion baselines at comparable backbone scales on mathematical reasoning across the evaluated denoising budgets (Tables 2–3); our supervised and post-NFT models also outperform reported PlaidQ results on HumanEval(+) (Table 4).
2. **Representation design shapes both reasoning accuracy and denoising dynamics.** We analyze representations for both ease of decoding (Appendix C.2) and ease of generation, motivating a learned multilayer representation that improves reasoning accuracy (Figure 2). Its layer decomposition enables asynchronous denoising across components, improving accuracy over synchronous denoising in our matched schedule comparison (Section 4.3 and Table 1). Thus the latent space shapes both what the model learns and how it generates.
3. **Learning compact prompt encoders through staged adaptation.** We show that *information preservation* for denoising is sufficient for an ideal downstream score network to recover the correct text-conditional score, without exact MSE matching to teacher prompt features (Lemma 1). This permits a compact prompt encoder to replace the large teacher. Our staged curriculum first learns the flow under fixed teacher conditioning, then fits the encoder and jointly adapts both networks (Figure 3), addressing the difficulty of learning them together from initialization (Figure 2).
4. **Guidance-compatible diffusion reinforcement learning for reasoning.** We propose a method to reconcile DiffusionNFT's CFG-free optimization (Zheng et al., 2025) with ELF's learned self-conditioning guidance. We optimize guidance-corrected fields while retaining guided rollouts, jointly updating the flow and prompt encoder as a text-conditioned policy (Section 5.1). Gold-solution endpoints supplement sparse correctness rewards, with their anchoring effect characterized in Lemma 2. NFT improves single-sample accuracy on math and code, and majority-vote performance on mathematics (Figure 4; Table 4).

2 Related work

Diffusion language models. Language diffusion operates in continuous representations (Gulrajani & Hashimoto, 2023; Tae et al., 2025; Deschenaux & Gulcehre, 2026) or discrete token spaces (Sahoo et al., 2024; Nie et al., 2025; Tang & Wang, 2026), with recent continuous models targeting reasoning, coding, and few-step generation (Agarwal et al., 2026; Azangulov et al., 2026; Peng et al., 2026b). We build on ELF (Hu et al., 2026), which denoises contextual representations and shares a backbone between denoising and token decoding. Concurrent to our paper, ELF-REG (Li et al., 2026) augments ELF with teacher-feature alignment and a jointly denoised global representation. LFRM learns its answer representation and replaces external prompt encoders with a compact trainable network. **Representation design and denoising order.** Related work learns or aligns diffusion representations in vision and language (Yu et al., 2024; Jiang et al., 2026; Meshchaninov et al., 2026), and explores separate denoising clocks for representation components in vision (Pan et al., 2025; Baade et al., 2026). We study learned multilayer language representations for both decoding and generation, and use their structure for asynchronous inference. **Diffusion reinforcement learning.** Reward-based post-training has also been explored for diffusion reasoning (Zhao et al., 2025; Zhu et al., 2025; Kang et al., 2026). We adapt DiffusionNFT (Zheng et al., 2025), which optimizes rewarded endpoints through forward-process regression. Our adaptation accommodates ELF's learned self-conditioning guidance, adds gold-solution anchoring, and jointly updates the flow model and prompt encoder.

3 Preliminaries

Tokens, encodings, and latents. Let $q = (q_1, \ldots, q_{L_q})$ and $a = (a_1, \ldots, a_{L_a})$ be prompt and answer tokens in vocabulary $\mathcal{V}$, with the answer including its terminal token. Their padded canvas $y = (q, a, \text{padding})$ has length L; canvas index i and answer index j satisfy $y_{L_q+j} = a_j$. An answer encoder gives $z_{\text{clean}} = \mathcal{E}(q, a) \in \mathbb{R}^{L_a \times d}$, and a prompt-only encoder gives $c_\phi(q) = P_\phi(q) \in \mathbb{R}^{L_q \times d}$; rows are token vectors. Unlike the teacher's tokenwise lookup $B_\psi[q_i]$, P_ϕ uses prompt context. Parameters ψ are frozen teacher weights, ϕ the prompt encoder, and θ the shared denoising/decoding

network, with $\Theta = (\theta, \phi)$. Teacher layer ℓ contributes width d_ℓ to total latent width d. Uppercase letters denote random variables.

Flow matching. Flow matching transports noise to data (Lipman et al., 2022). For noise scale $\sigma > 0$, our linear Gaussian path runs from noise at $t = 0$ to clean latents at $t = 1$:

$$z_t = tz_{\text{clean}} + (1-t)\sigma\epsilon, \qquad \epsilon \sim \mathcal{N}(0, I), \qquad u_t := \frac{\mathrm{d}z_t}{\mathrm{d}t} = z_{\text{clean}} - \sigma\epsilon. \tag{1}$$

Treating text q as the external condition and $c_\phi(q)$ as its internal encoding, write $v_\Theta(z, t; q) := v_\theta(z, t; c_\phi(q))$. The conditional objective is

$$\mathcal{L}_{\text{FM}}^{\text{basic}}(\Theta) = \mathbb{E}\Big[\big\|v_\theta\big(z_t, t; c_\phi(q)\big) - u_t\big\|_F^2\Big]. \tag{2}$$

The expectation samples training pairs (q, a), times $t \sim \pi(t)$, and independent Gaussian noise.

Text and encoded conditioning. Write $p_t(z \mid q)$ for the forward-path density and $s^\star(z, t \mid q) = \nabla_z \log p_t(z \mid q)$ for its conditional score. For $0 < t < 1$, the population-optimal velocity is

$$v^\star(z, t \mid q) = \mathbb{E}[u_t \mid z_t = z, q] = \frac{z + \sigma^2(1-t)s^\star(z, t \mid q)}{t}. \tag{3}$$

Conditioning on $c_\phi(q)$ may merge distinct prompts; Section 4.2 characterizes when the encoding preserves the text-conditional score.

Sampling under a different clock. At inference, let $\tau \in [0, 1]$ be solver time and $t = f(\tau)$ an increasing differentiable clock with $f(0) = 0$, $f(1) = 1$. Starting from $\bar{z}_0 \sim \mathcal{N}(0, \sigma^2 I)$, sampling obeys $\mathrm{d}\bar{z}_\tau/\mathrm{d}\tau = f'(\tau)v_\Theta(\bar{z}_\tau, f(\tau); q)$. For $f(\tau) = \tau^2$, the multiplier is 2τ. Exact integration preserves the endpoint, while finite-step accuracy can change. Section 4.3 extends this to group-specific clocks.

4 Continuous Diffusion Reasoning

For a general, fixed answer representation $\mathcal{E}(q, a)$ and prompt encoding $c = c(q)$, LFRM follows ELF's denoising and token-decoding formulation (Hu et al., 2026) (Figure 1). We use canvas length $L = 1024$, latent width $d = 1024$, and noise scale $\sigma = 2$.

Continuous denoising. ELF's bidirectional backbone predicts $\widehat{z}_{\text{clean}} = F_\theta(z_t, t; c, \widehat{z}_{\text{sc}}, g)$ from the clean prompt and noisy answer at a shared training time t. Here $\widehat{z}_{\text{sc}}$ is an earlier clean prediction and g the guidance scale. We apply Eq. (2) with velocity $v_\theta = (\widehat{z}_{\text{clean}} - z_t)/\Delta_t$ and stabilized target $u_t^{\text{stab}} = (z_{\text{clean}} - z_t)/\Delta_t$, with $\Delta_t = \max(1 - t, 0.05)$; outside the clamp, $u_t^{\text{stab}} = u_t$. Appendix B gives valid-token normalization.

Self-conditioning and SCCFG. ELF's self-conditioning classifier-free guidance (SCCFG) learns to amplify answer self-conditioning at scale g. It compares predictions with and without *answer self-conditioning*, retaining the prompt in both; Appendix B gives the detached guidance-adjusted target. Guidance is internal to the field, requiring one denoiser call per step.

Recovering discrete tokens. A vocabulary head on the same backbone predicts same-position answer tokens under a separate decoder corruption z^{dec}:

$$\mathcal{L}_{\text{CE}} = \mathbb{E}_{\text{valid}}\big[-\log p_{\theta,j}^{\text{tok}}(a_j \mid z^{\text{dec}}, c(q))\big]. \tag{4}$$

The expectation averages valid answer tokens, including the terminal token; Appendix B specifies the exact batch weighting, decoder corruption, and mode mixing. At inference, encode the prompt once and keep it clean through S Euler steps with recurrent self-conditioning; one additional backbone call decodes the answer in parallel and is excluded from denoising NFE.

4.1 Learning the denoising representation

The denoising representation must support generation and token recovery. We extract **teacher-forced Qwen3-4B-Instruct-2507 activations** (Qwen Team, 2025) in one forward pass; although these features are causal, our bidirectional denoiser jointly generates answer representations.

Latent construction. Let $h_j^{(\ell)}$ be the frozen teacher's post-block activation at answer position j. We concatenate centered linear projections as $z_{\text{clean},j} = \big[(h_j^{(\ell)} - \mu_\ell)W_\ell^{\text{enc}}\big]_{\ell \in \mathcal{S}}$ for $\mathcal{S} = \{16, 24, 32\}$,

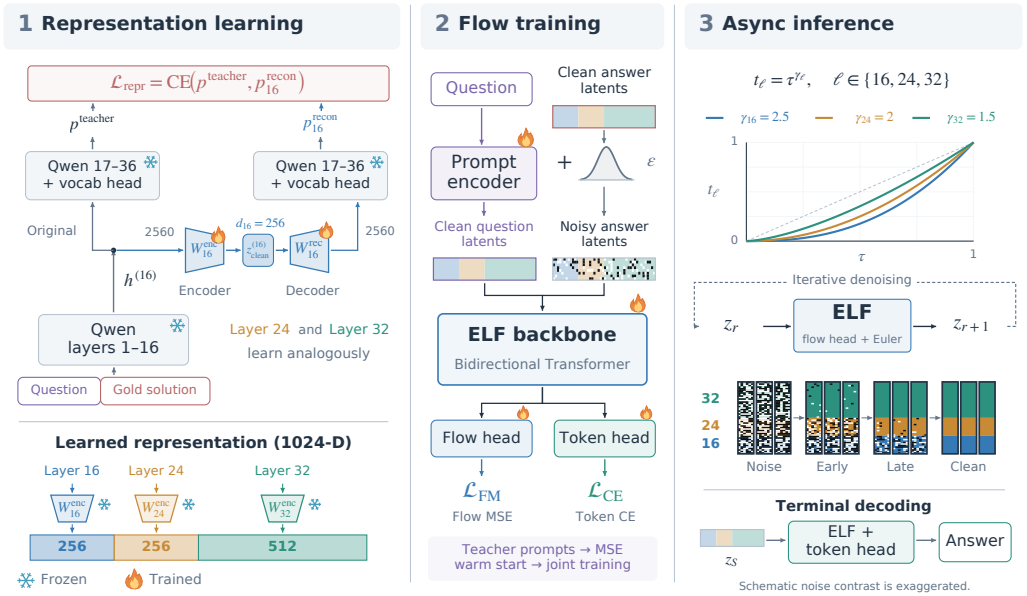

Figure 1: **The LFRM training and inference pipeline. Left:** Learn projections of Qwen layers 16/24/32 through teacher cross-entropy, then concatenate their outputs (Section 4.1). **Center:** Train the ELF backbone with flow and token-decoding losses and staged prompt learning (Sections 4 and 4.2). **Right:** Denoise the representation groups under asynchronous clocks, then decode the answer in parallel (Section 4.3).

with widths $(d_{16}, d_{24}, d_{32}) = (256, 256, 512)$ and fixed means μ_ℓ. These maps initially also encode prompts; answer representations stay fixed during flow training and NFT.

Learning the projectors. Linear decoders reconstruct $\widehat{h}_j^{(\ell)} = \mu_\ell + z_{\text{clean},j}^{(\ell)} W_\ell^{\text{rec}}$ (Figure 1, left). Replacing layer-ℓ answer activations with these reconstructions and running the frozen Qwen suffix yields next-token predictions $p_{\ell,j}^{\text{recon}}$. We preserve the teacher's predictions through $\mathcal{L}_{\text{repr}} = \sum_{\ell \in \mathcal{S}} \mathbb{E}_{q,a,j}\ \text{CE}(p_j^{\text{teacher}}, p_{\ell,j}^{\text{recon}})$, where CE is full-vocabulary soft cross-entropy. Only W_ℓ^{enc} and W_ℓ^{rec} are learned; the reconstruction decoders are used only at this stage. Appendix C gives initialization, whitening, and scoring details.

The first three arms in Figure 2 compare 1,024-dimensional whitened representations under matched six-epoch LFRM-B training and evaluation; Table 10 gives four-seed endpoint results.

Decodability and representation choice. The fixed 1,024-dimensional layer-32 PCA representation supports **99.24%** held-out same-position token recovery after two epochs of training a separate decoder (Appendix C.2). Yet LFRM-B using this same representation achieves only **11.20%** GSM8K accuracy after six epochs. Accurate clean-token recovery alone is therefore an insufficient criterion for choosing a denoising representation.

Layer selection. We choose layers 16, 24, and 32 based on centered kernel alignment (CKA; Kornblith et al., 2019) of Qwen activations; Appendix C.3 gives the analysis. Holding the projection method and total latent width fixed, multilayer PCA raises accuracy to **28.43%** (Figure 2). The resulting layer groups also enable distinct denoising clocks (Section 4.3).

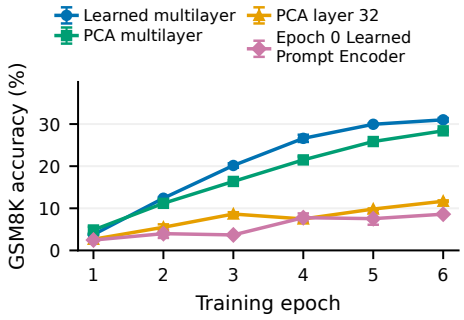

Figure 2: **Representation and conditioning affect reasoning.** LFRM-B learning curves on GSM8K, with 32 steps and $t = \tau^2$. The first three variants retain Qwen conditioning; the fourth jointly trains a randomly initialized prompt encoder. Protocol: Appendix C.4; values and SDs: Table 10 (Appendix C.4).

Learned versus PCA. PCA preserves high-variance directions, which need not preserve the teacher's predictive behavior. Holding layers and latent widths fixed, learning the projectors through teacher cross-entropy raises accuracy from **28.43% to 30.76%** (Figure 2).

4.2 Learning the conditional encoding in three stages

ELF and ELF-REG use external contextual prompt encoders (Hu et al., 2026; Li et al., 2026). We instead replace the Qwen teacher Transformer with a compact trainable encoder, retaining its frozen token lookup. For LFRM-L, the contextual network has **121M parameters**, compared with 3.63B in the full teacher Transformer (both excluding vocabulary input/output maps). The student uses six bidirectional attention blocks and encodes each prompt once; see Appendix A for its architecture.

The prompt as the condition. The text-conditioned field v_Θ of Section 3 includes self-conditioning and guidance as auxiliary inputs. We jointly optimize θ and ϕ, keeping the prompt text, answer representation, and forward corruption fixed. The following lemma characterizes what the encoding must preserve.

Lemma 1 (Preserving the text-conditional score)**.** *Let Z_{clean} and Q denote the clean answer latents and prompt, with $\mathbb{E}\|Z_{\text{clean}}\|_F^2 < \infty$, and fix a deterministic encoder $C = P_\phi(Q)$. Let $s^\star(z, t \mid c)$ denote the score conditioned on $C = c$. Under the independent Gaussian corruption in Eq.* (1)*, for $0 < t < 1$ and $c = P_\phi(q)$,*

$$s^\star(z,t \mid q) - s^\star(z,t \mid c) = \frac{t}{\sigma^2(1-t)^2}\big(\mathbb{E}[Z_{\text{clean}} \mid Z_t = z, Q = q] - \mathbb{E}[Z_{\text{clean}} \mid Z_t = z, C = c]\big). \quad (5)$$

Thus an encoder preserves the full-prompt score exactly when it preserves the posterior mean clean answer. If this holds along the path, the optimal unguided field also agrees by Eq. (3)*, giving the same conditional endpoint law under exact integration whenever the flow is well-defined.*

The encoder need not reproduce the teacher's coordinates: it must preserve the information needed for the posterior mean answer. Appendix B.2 proves the lemma and gives an information-preserving encoding with arbitrarily large teacher-imitation MSE. MSE matching supplies a compatible initialization, while downstream training can adapt the encoder and denoiser jointly. We use three stages, keeping answer representations and token embeddings fixed throughout (details in Appendix A):

1. Frozen conditioning. Train the flow model for 12 epochs using fixed Qwen prompt representations and the flow/decoder objective. **2. Prompt imitation.** Freeze the flow model and fit the compact encoder to the teacher prompt representations using MSE. **3. Joint adaptation.** Substitute the learned encoder and optimize both networks through the flow/decoder objective, without an additional prompt-MSE loss, to epoch 18 for LFRM-B or epoch 21 for LFRM-L on the mathematical reasoning tasks.

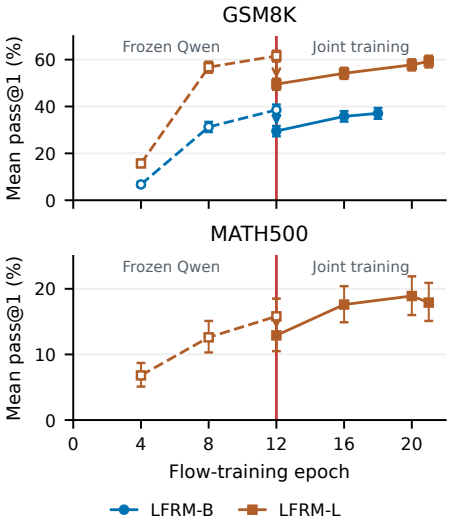

Figure 3: **From teacher conditioning to learned prompts.** Top: GSM8K; bottom: MATH500. Circles/squares denote LFRM-B/L; open/filled markers denote Qwen/learned prompt encoding. Red line: epoch-12 switch; arrows: immediate MSE-swap loss. Values and settings: Appendix B.3.

Replacing Qwen with the MSE-trained encoder immediately reduces accuracy, despite unchanged flow-model weights (Figure 3). Subsequent joint training recovers much of this loss on GSM8K and exceeds the original teacher-conditioned checkpoint on MATH500. This recovery reflects adaptation of *both* networks, showing the value of learning the conditioning interface through the generative objective.

Importance of fixing initial conditioning. Joint learning from initialization makes the conditioning interface change while the flow model is still learning to interpret it. Under the matched six-epoch comparison, this approach reaches only **8.70%** GSM8K accuracy, versus **30.76%** with frozen teacher conditioning (Figure 2). This motivates stabilizing the condition during early learning and adapting it afterward.

4.3 Asynchronous inference with local clocks

Our multilayer representation allows different feature groups to become clean at different rates, potentially letting one group use cleaner information from another. We extend the shared clock of Section 3 to layer-specific clocks $t_\ell = f_\ell(\tau) = \tau^{\gamma_\ell}$ for $\ell \in \{16, 24, 32\}$ at inference, while retaining synchronous training. Equal exponents recover synchronous inference; unequal exponents give different noise levels within a single denoiser call.

Adapting a synchronously trained denoiser. Although training uses a shared time, inference applies the same learned time-embedding function to each local time and averages its outputs, $\bar{e}^{\text{time}}_\theta(\boldsymbol{t}) = \frac{1}{3}\sum_\ell e^{\text{time}}_\theta(t_\ell)$. This needs no retraining and recovers the original embedding when local times agree.

For solver state $z_r = \bar{z}_{\tau_r}$, predicted clean endpoint $\widehat{z}_{\text{clean},r}$, and local times $t_{\ell,r} = f_\ell(\tau_r)$, the groupwise Euler update is

$$z^{(\ell)}_{r+1} = z^{(\ell)}_r + (\tau_{r+1} - \tau_r)\, f'_\ell(\tau_r) \frac{\widehat{z}^{(\ell)}_{\text{clean},r} - z^{(\ell)}_r}{\Delta_{t_{\ell,r}}}, \tag{6}$$

with Δ_t defined in Section 4. All groups update from the same denoiser call, so asynchrony does not increase NFE. Unlike shared-clock reparameterization, this adapts the model to unequal group times absent from synchronous training; implementation details appear in Appendix D.

Schedule choice and reasoning accuracy. Our headline results use $(\gamma_{16}, \gamma_{24}, \gamma_{32}) = (2.5, 2, 1.5)$, consistent with the highest observed accuracy in the reported MATH validation comparison (Appendix D). The shared-square clock improves on the identity clock, and $(2.5, 2, 1.5)$ achieves the highest mean among the four schedules in Table 1 on both benchmarks. The validation and lower-NFE comparisons show that the ordering depends on the evaluation setting (Appendix D).

Table 1: **Inference clocks change accuracy without retraining.** LFRM-L epoch 21, 32 steps; mean pass@1 (%). Protocol and full sweep: Appendix D.

$(\gamma_{16}, \gamma_{24}, \gamma_{32})$	GSM8K	MATH500
$(1, 1, 1)$	52.96 ($\pm$1.13)	15.50 ($\pm$1.27)
$(2, 2, 2)$	57.51 ($\pm$0.80)	17.40 ($\pm$1.41)
$(1.5, 2, 2.5)$	58.07 ($\pm$0.64)	17.60 ($\pm$1.98)
$(2.5, 2, 1.5)$	**59.25** ($\pm$0.16)	**17.90** ($\pm$1.27)

Reconstruction probes reveal distinct sensitivities across layer groups, with weaker contrasts under random coordinate grouping (Figure 7). These findings motivate studying coordinated denoising across representations, but do not establish the mechanism behind the sampling gains.

5 Reinforcement learning with self-conditioned flows

DiffusionNFT (Zheng et al., 2025) improves continuous generators through rewarded endpoint regression, without differentiating through sampling trajectories. We adapt it to LFRM's learned self-conditioning guidance and supplement sparse correctness rewards with gold-solution endpoints.

5.1 NFT with learned self-conditioning guidance

The velocity field optimized by standard NFT. Let v_{cur} and v_{old} denote the current and old policy velocity fields, u a sample's forward velocity target, and $\rho \in [0, 1]$ its normalized optimality coefficient. NFT forms implicit positive and negative velocity fields, $v^+ = (1-\beta)v_{\text{old}} + \beta v_{\text{cur}}$ and $v^- = (1+\beta)v_{\text{old}} - \beta v_{\text{cur}}$. For current and old velocity-field values f, h, the pointwise loss is

$$\ell_\beta(f, h; u, \rho) = \rho\|(1-\beta)h + \beta f - u\|_2^2 + (1-\rho)\|(1+\beta)h - \beta f - u\|_2^2. \tag{7}$$

Standard NFT applies this loss to $f = v_{\text{cur}}$ and $h = v_{\text{old}}$, holding the old field fixed; its original experiments use CFG-free optimization (Zheng et al., 2025). LFRM instead learns a guidance-conditioned field, motivating a correction before applying NFT's regression objective.

Constructing the SCCFG-compatible field. For each policy $p \in \{\text{cur}, \text{old}, \text{ref}\}$, forward-noise an endpoint and obtain velocity v^0_p from a detached bootstrap without answer self-conditioning. The main prediction v^{main}_p uses the bootstrap clean endpoint when $b_{\text{sc}} = 1$ and zero answer self-conditioning otherwise; both passes retain the question. Following ELF's guidance construction, define

$$\widetilde{v}_p = v^{\text{main}}_p - \text{sg}\big[b_{\text{sc}}(1 - g^{-1})(v^{\text{main}}_p - v^0_p)\big], \tag{8}$$

where $g > 0$ is the SCCFG scale. Our update applies Eq. (7) to $f = \widetilde{v}_{\mathrm{cur}}$, $h = \mathrm{sg}(\widetilde{v}_{\mathrm{old}})$, and $u = u_t^{\mathrm{stab}}$ from Section 4. We use $\beta = 1$ and add a squared-velocity penalty toward the corrected field $\widetilde{v}_{\mathrm{ref}}$ of the fixed initial reference. With self-conditioning enabled, the corrected value is $v_p^0 + (v_p^{\mathrm{main}} - v_p^0)/g$. Detaching the correction preserves the main-pass Jacobian rather than scaling it by $1/g$ (Appendix E).

Guided rollouts and forward-process updates. Collection retains recurrent self-conditioning and SCCFG guidance. We reward decoded answers but optimize their saved continuous endpoints, using fresh forward corruption and bootstrap predictions rather than replaying rollout histories. Thus rollouts remain guided while NFT optimizes the corrected fields in Eq. (8).

5.2 Gold-anchored reasoning updates

Each question contributes 15 old-policy generations and one gold latent endpoint, with raw rewards 0.75 for correct generated answers, 0 for incorrect answers, and 1 for gold. The gold endpoint supplies a preferred target even when all generated answers are incorrect. We discard groups whose generated answers are all correct, concentrating updates on observed failures; retained rewards are normalized into ρ as described in Appendix E. This recipe uses reference solutions as well as verifiable rewards. Gold raw reward one need not give $\rho_G = 1$; the following lemma characterizes its contribution.

Lemma 2 (Gold endpoints as field anchors)**.** *For a gold endpoint, let $\mathcal{H}$ contain the noisy state, time, question, coefficient $\rho_G \in [1/2, 1]$, and any auxiliary inputs held fixed for the update. Let U denote the random stabilized target u_t^{stab}, assume its conditional second moment is finite, and set $m_G = \mathbb{E}[U \mid \mathcal{H}, \mathrm{gold}]$. For $\mathcal{H}$-measurable current and old velocity-field values f, h, define $a_G = (2\rho_G - 1)m_G + 2(1 - \rho_G)h$. The beta-one NFT loss satisfies*

$$\mathbb{E}[\ell_1(f, h; U, \rho_G) \mid \mathcal{H}, \mathrm{gold}] = \|f - a_G\|_2^2 + C_{\mathcal{H}}, \tag{9}$$

where $C_{\mathcal{H}}$ is independent of f with the conditioning information and auxiliary fields fixed. Thus the gold term attracts the current field toward a convex combination of the gold posterior target and the old field.

At $\rho_G = 1$, the gold term directly regresses toward m_G. Under an ideal reference model matching that gold posterior target, the anchor interpolates between the reference and old fields; Appendix E.4 gives the proof and scope. This gold-induced anchor complements the explicit reference penalty.

Because the external condition is the question text, NFT jointly updates the prompt encoder and flow model as one conditional policy (Lemma 1), without a prompt-imitation objective. Training uses synchronous identity clocks; asynchronous evaluation follows Section 4.3. Appendix A gives optimization details, and Sections 6.1 and 6.2 discuss the empirical gains.

6 Empirical Evaluation

We evaluate mathematical reasoning on GSM8K and MATH500 (Cobbe et al., 2021; Hendrycks et al., 2021; Hugging Face H4, 2024), and code generation in Section 6.2. Math evaluations use zero-shot prompts, async clocks $(2.5, 2, 1.5)$, numerical-answer scoring for GSM8K, and Math-Verify for MATH500 (Kydlíček, 2025). Headline pass@1 averages eight sampling seeds at 64 denoising steps. NFE counts denoising calls, excluding prompt encoding and terminal decoding. Appendices F and G.2 give checkpoint settings and seed standard deviations.

6.1 Mathematical reasoning

Reasoning accuracy across denoising budgets. At comparable denoising-backbone sizes, supervised LFRM already exceeds the reported continuous-diffusion baselines, and NFT gives our strongest results (Table 2). On MATH500, supervised LFRM-L reaches 21.18%, compared with ELF-REG-L's 13.39% at 127 denoising calls (Li et al., 2026). Our 90.4M/638M denoisers use compact learned prompt encoders, without executing the teacher Transformer during generation. The autoregressive and discrete-diffusion rows provide broader performance context; Table 13 in Appendix G.1 adds comparisons with larger models.

Both supervised and post-NFT LFRM retain this advantage at 8, 16, 32, and 64 NFE within each backbone-size group (Table 3). At eight steps, supervised LFRM-L reaches 15.23% on MATH500,

Table 2: Mathematical reasoning, mean pass@1 (%). Parameter counts include frozen weights; Embedding covers vocabulary input/output maps, and $(+X)$ denotes parameters used once per response. Bold/underline mark best/second within each continuous-backbone group. $^{*}N$ denotes $N-1$ denoising calls; slashes separate GSM8K/MATH500 budgets. †Python scoring. Full comparisons: Appendix G.1; protocols: Appendix F; seed SDs: Appendix G.2.

Model	Embedding Params (M)	Other Params (M)	NFE	GSM8K	MATH500
Autoregressive					
Qwen3-0.6B-Base (Yang et al., 2025)	156	440	—	59.59	—
Qwen3-0.6B, thinking (Yang et al., 2025)	156	440	—	—	77.60
Discrete diffusion / blockwise diffusion					
SDAR-1.7B-Chat (Cheng et al., 2025)	622	1,410	4/block	80.10	63.20
Continuous diffusion / flow: denoising backbone below 200M					
$\mathbb{S}$-FLM (top-1) (Deschenaux & Gulcehre, 2026)	75.5	87.5	1,024	18.00†	—
FMLM+ Init (Agarwal et al., 2026)	75.5	92	64	33.60†	—
ELF-REG-B (Li et al., 2026)	311	104 $(+441)$	$^{*}64$	38.11	—
LFRM-B, pre-NFT (ours)	545	90.4 $(+45.8)$	64	<u>39.17</u>	—
LFRM-B, post-NFT (ours)	545	90.4 $(+45.8)$	64	**42.16**	—
Continuous diffusion / flow: denoising backbone 200M–1B					
ELF-REG-L (Li et al., 2026)	311	652 $(+442)$	$^{*}64$ / $^{*}128$	55.96	13.39
LFRM-L, pre-NFT (ours)	545	638 $(+123)$	64	<u>58.67</u>	<u>21.18</u>
LFRM-L, post-NFT (ours)	545	638 $(+123)$	64	**63.74**	**24.68**

Table 3: Reasoning at lower NFE, mean pass@1 (%). Table 2 gives model parameter counts. Bold/underline mark first/second within each benchmark and backbone-scale block; * marks one fewer denoising call than the heading. External results follow Agarwal et al. (2026); Li et al. (2026); Appendix F gives source and protocol details.

	GSM8K (∼100M)				GSM8K (∼650M)				MATH500 (∼650M)			
Method / NFE	8	16	32	64	8	16	32	64	8	16	32	64
FMLM+ Init	15.10	26.10	31.80	33.60	—	—	—	—	—	—	—	—
ELF-REG	*26.20	*34.42	*37.15	*38.11	*47.34	*52.89	*55.04	*55.96	*9.76	*11.53	*12.40	*13.31
LFRM, pre-NFT	<u>26.40</u>	<u>34.48</u>	<u>37.80</u>	<u>39.17</u>	<u>49.41</u>	<u>55.33</u>	<u>57.88</u>	<u>58.67</u>	<u>15.23</u>	<u>17.93</u>	<u>19.65</u>	<u>21.18</u>
LFRM, post-NFT	**30.08**	**37.77**	**41.13**	**42.16**	**54.84**	**60.55**	**62.47**	**63.74**	**17.45**	**20.93**	**22.60**	**24.68**

Our checkpoints are fixed across columns, with SCCFG 2 for GSM8K and 3 for MATH500. Sampling-seed counts and SDs: Table 14 in Appendix G.2.

exceeding ELF-REG-L's 13.39% at 127 denoising calls. Useful reasoning therefore emerges without training a separate few-step student. These are reported operating points with different training and evaluation protocols; NFE does not equate FLOPs across methods. The MATH500 headline tables use SCCFG 3; GSM8K and all inference-allocation curves retain SCCFG 2.

Refining an answer or drawing more answers. For k samples with S denoising steps each, we use kS as the inference budget. Oracle pass@k measures whether any sample is correct; majority vote selects the most frequent final answer after grouping equivalent answers, without consulting the reference. For selectors returning one of the generated answers, voting provides an achievable *lower bound* on optimal selection accuracy, while oracle pass@k supplies its *upper bound*. We compare $S \in \{8, 16, 32, 64\}$ up to a budget of 512 calls; Appendix G.4 gives aggregation details.

Figure 4 shows that oracle selection generally favors more attempts with fewer steps, whereas voting generally favors higher-NFE samples. On MATH500 after NFT, 64 eight-step samples reach 67.00% oracle accuracy, versus 52.60% for eight 64-step samples; voting reverses this preference, at 25.79% versus 32.42%. Intermediate depths remain competitive for voting. The voting–oracle gap motivates better answer selection, potentially using latent-space or hidden-state classifiers.

What changes after NFT? NFT improves single-sample accuracy at every reported denoising depth (Tables 2–3). At 64 steps with SCCFG 2, the paired LFRM-L gains are $+5.07$ percentage points on GSM8K and $+4.75$ on MATH500. These results support the promise of diffusion RL for continuous reasoning models. The benefit is less uniform for high-k oracle coverage: at eight steps on

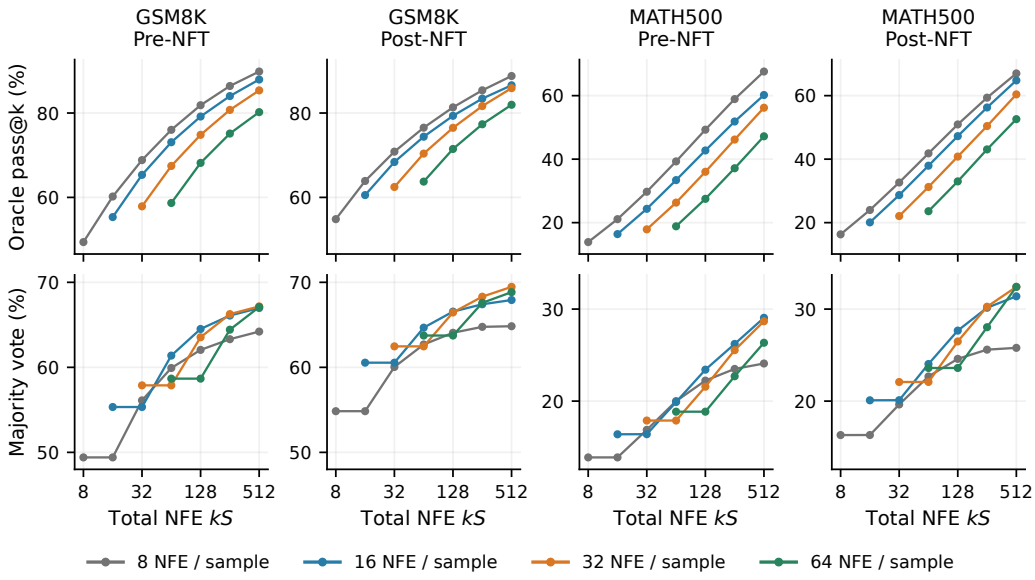

Figure 4: LFRM-L before and after NFT on GSM8K and MATH500, with SCCFG 2. Top: oracle pass@k; bottom: majority-vote accuracy. Colors denote denoising NFE per sample; the budget is kS. Numerical values appear in Appendix G.4, and LFRM-B curves in Appendix G.3.

MATH500, NFT raises pass@1 from 13.88% to 16.31%, while pass@64 changes from 67.60% to 67.00%. Voting shows clearer gains: with eight 64-step samples, MATH500 accuracy rises from 26.34% to 32.42% (+6.07 points). Voting benefits from the correct answer's frequency relative to competing answers, whereas oracle selection needs only one successful draw. These patterns do not establish reduced reasoning diversity after NFT; Appendix G.4 gives further comparisons and interpretation. Appendices I.1–I.2 illustrate complete solutions and intermediate generations.

6.2 Code Generation

On OpenCodeInstruct (Ahmad et al., 2025), LFRM-L uses 12 epochs of frozen-Qwen flow training, ten prompt-MSE epochs, one joint epoch, and 100 NFT updates with execution-based rewards (Appendix A). Table 4 uses eight seeds, 128 denoising calls, async $(2.5, 2, 1.5)$, and SCCFG 3.

Table 4: **Code generation without function-name repair.** Mean pass@1 (%). HE/HE+ use base/full extended tests; both MBPP columns use 378 tasks. PlaidQ's "MBPP+" uses base tests. Frozen Qwen: epoch 12; pre-NFT: joint epoch 13; post-NFT: update 100. NFE excludes terminal decoding; PlaidQ+CFG uses two calls per step. Protocols: Appendix H; seed SDs: Table 20.

Model / setting	Denoising NFE	HE	HE+	MBPP-378 base	MBPP+ full
PlaidQ (Peng et al., 2026b)	128	19.02	18.05	22.18	—
PlaidQ+CFG (Peng et al., 2026b)	256	22.04	19.70	24.58	—
LFRM-L / frozen Qwen	128	33.69	30.87	38.72	33.99
LFRM-L, pre-NFT (ours)	128	29.57	26.91	18.45	16.40
LFRM-L, post-NFT (ours)	128	32.85	30.18	22.26	19.61

Pre- and post-NFT LFRM exceed reported PlaidQ results on HumanEval(+), but trail guided PlaidQ on MBPP-378 base. Under matched function-name aliasing, they also exceed ELF-REG-L (Appendix H). NFT raises pass@1 on all four coding metrics, including HumanEval+ by 3.28 points and MBPP+ by 3.21. HumanEval(+) approaches frozen-Qwen conditioning, while a substantial MBPP gap remains. Higher-pass oracle gains are smaller, as in mathematics; paired comparisons appear in Appendix H.

REPRODUCIBILITY STATEMENT

Appendix A specifies model architectures, training populations, stage durations, optimization settings, and checkpoint selection. Appendices B–E provide the supervised objectives, representation construction, sampling procedures, and NFT implementation. The assumptions and proofs of Lemmas 1 and 2 are given in Appendices B.2 and E.4. Evaluation protocols, sampling-seed counts, EMA selectors, scoring conventions, and uncertainty estimates are documented in Appendices F and H; additional numerical comparisons and ablations are collected in Appendices G–H. Code will be available at `https://anonymous.4open.science/r/LFRM-98B5/`.

AI USE STATEMENT

We used generative AI tools for literature search, planning, executing and logging experiment results, cleaning, organizing and shortening the paper draft, organizing and formatting the code for public release. We have reviewed all AI-assisted work. We checked compiled results against experiment outputs and verified AI-generated code. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI

REFERENCES

Manan Agarwal, Sheel Shah, Chanhyuk Lee, Jaehoon Yoo, Jerry Huang, Seunghoon Hong, Aditi Raghunathan, Jinwoo Kim, and Nicholas M. Boffi. Posterior refinement: Fast language generation via any-order flow maps. *arXiv preprint arXiv:2606.24773*, 2026. URL https://arxiv.org/abs/2606.24773.

Wasi Uddin Ahmad, Aleksander Ficek, Mehrzad Samadi, Jocelyn Huang, Vahid Noroozi, Somshubra Majumdar, and Boris Ginsburg. OpenCodeInstruct: A large-scale instruction tuning dataset for code LLMs. *arXiv preprint arXiv:2504.04030*, 2025. URL https://arxiv.org/abs/2504.04030.

Jacob Austin, Augustus Odena, Maxwell Nye, Maarten Bosma, Henryk Michalewski, David Dohan, Ellen Jiang, Carrie Cai, Michael Terry, Quoc Le, and Charles Sutton. Program synthesis with large language models. *arXiv preprint arXiv:2108.07732*, 2021. URL https://arxiv.org/abs/2108.07732.

Iskander Azangulov, Kianoosh Ashouritaklimi, Leo Zhang, Simon Vary, and Patrick Rebeschini. Masked language flow models. *arXiv preprint arXiv:2606.27617*, 2026. URL https://arxiv.org/abs/2606.27617.

Alan Baade, Eric Ryan Chan, Kyle Sargent, Changan Chen, Justin Johnson, Ehsan Adeli, and Li Fei-Fei. Latent forcing: Reordering the diffusion trajectory for pixel-space image generation. *arXiv preprint arXiv:2602.11401*, 2026. URL https://arxiv.org/abs/2602.11401.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021. URL https://arxiv.org/abs/2107.03374.

Shuang Cheng, Yihan Bian, Dawei 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. *arXiv preprint arXiv:2510.06303*, 2025. URL https://arxiv.org/abs/2510.06303.

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

DeepSeek-AI, Daya Guo, Dejian Yang, Haowei Zhang, et al. DeepSeek-R1: Incentivizing reasoning capability in LLMs via reinforcement learning. *arXiv preprint arXiv:2501.12948*, 2025. URL https://arxiv.org/abs/2501.12948v1.

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

Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-Based Diffusion Language Models. *arXiv preprint arXiv:2305.18619*, 2023. URL https://arxiv.org/abs/2305.18619.

Marton Havasi, Brian Karrer, Itai Gat, and Ricky T. Q. Chen. Edit flows: Flow matching with edit operations. *arXiv preprint arXiv:2506.09018*, 2025. URL https://arxiv.org/abs/2506.09018.

Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the MATH dataset. *arXiv preprint arXiv:2103.03874*, 2021. URL https://arxiv.org/abs/2103.03874.

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. URL https://arxiv.org/abs/2605.10938.

Hugging Face H4. MATH-500, 2024. URL https://huggingface.co/datasets/HuggingFaceH4/MATH-500.

Jiaxiu Jiang, Jingjing Ren, Wenbo Li, Bo Wang, Haoze Sun, Yijun Yang, Jianhui Liu, Yanbing Zhang, Shenghe Zheng, Yuan Zhang, Haoyang Huang, Nan Duan, and Wangmeng Zuo. TextLDM: Language modeling with continuous latent diffusion. *arXiv preprint arXiv:2605.07748*, 2026. URL https://arxiv.org/abs/2605.07748.

Keller Jordan. Muon: An optimizer for hidden layers in neural networks. Software, 2024. URL https://github.com/KellerJordan/Muon.

Haoqiang Kang, Yizhe Zhang, Nikki Lijing Kuang, Yi-An Ma, and Lianhui Qin. LaDi-RL: Latent Diffusion Reasoning Prevents Entropy Collapse in Reinforcement Learning. *arXiv preprint arXiv:2602.01705*, 2026. URL https://arxiv.org/abs/2602.01705.

Simon Kornblith, Mohammad Norouzi, Honglak Lee, and Geoffrey Hinton. Similarity of neural network representations revisited. *arXiv preprint arXiv:1905.00414*, 2019. URL https://arxiv.org/abs/1905.00414.

Hynek Kydlíček. Math-Verify: Math Verification Library, 2025. URL https://github.com/huggingface/Math-Verify.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay 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. *arXiv preprint arXiv:2206.14858*, 2022. URL https://arxiv.org/abs/2206.14858.

Zeyu Michael Li, William Xingxu Chen, Bingshuo Qian, Jiayin Liu, and Xiang Cheng. ELF-REG: Scaling continuous diffusion language models to reasoning tasks. *arXiv preprint arXiv:2609.29102*, 2026. URL https://arxiv.org/abs/2609.29102.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022. URL https://arxiv.org/abs/2210.02747.

Jiawei Liu, Chunqiu Steven Xia, Yuyao Wang, and Lingming Zhang. Is your code generated by ChatGPT really correct? Rigorous evaluation of large language models for code generation. *arXiv preprint arXiv:2305.01210*, 2023. URL https://arxiv.org/abs/2305.01210.

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. URL https://arxiv.org/abs/2605.07933.

Arindam Mitra, Hamed Khanpour, Corby Rosset, and Ahmed Awadallah. Orca-Math: Unlocking the potential of SLMs in grade school math. *arXiv preprint arXiv:2402.14830*, 2024. URL https://arxiv.org/abs/2402.14830.

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. *arXiv preprint arXiv:2502.09992*, 2025. URL https://arxiv.org/abs/2502.09992.

Yueming Pan, Ruoyu Feng, Qi Dai, Yuqi Wang, Wenfeng Lin, Mingyu Guo, Chong Luo, and Nanning Zheng. Semantics lead the way: Harmonizing semantic and texture modeling with asynchronous latent diffusion. *arXiv preprint arXiv:2512.04926*, 2025. URL https://arxiv.org/abs/2512.04926.

Fred Zhangzhi Peng, Alexis Fox, Anru R. Zhang, and Alexander Tong. Don't retrain, align: Adapting autoregressive LMs to diffusion LMs via representation alignment. *arXiv preprint arXiv:2605.06885*, 2026a. URL https://arxiv.org/abs/2605.06885.

Fred Zhangzhi Peng, Kaiwen Zheng, and Anru R. Zhang. Distilled Continuous Diffusion Language Models Can Write Code in Few Steps—or One. *arXiv preprint arXiv:2609.04531v1*, 2026b. URL https://arxiv.org/abs/2609.04531v1. Version 1; subsequently withdrawn by the authors.

Qwen Team. Qwen3-4B-Instruct-2507 model card, 2025. URL https://huggingface.co/Qwen/Qwen3-4B-Instruct-2507.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *arXiv preprint arXiv:1910.10683*, 2019. URL https://arxiv.org/abs/1910.10683.

Subham Sekhar Sahoo, Marianne Arriola, Yair Schiff, Aaron Gokaslan, Edgar Marroquin, Justin T Chiu, Alexander Rush, and Volodymyr Kuleshov. Simple and Effective Masked Diffusion Language Models. *arXiv preprint arXiv:2406.07524*, 2024. URL https://arxiv.org/abs/2406.07524.

Jaesung Tae, Hamish Ivison, Sachin Kumar, and Arman Cohan. TESS 2: A Large-Scale Generalist Diffusion Language Model. *arXiv preprint arXiv:2502.13917v2*, 2025. URL https://arxiv.org/abs/2502.13917v2.

Sophia Tang and Shiyi Wang. Discrete Beckmann transport models for one-step language modeling and reasoning. *arXiv preprint arXiv:2609.15903*, 2026. URL https://arxiv.org/abs/2609.15903.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. OpenMathInstruct-2: Accelerating AI for math with massive open-source instruction data. *arXiv preprint arXiv:2410.01560*, 2024. URL https://arxiv.org/abs/2410.01560.

An Yang, Anfeng Li, Baosong Yang, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025. URL https://arxiv.org/abs/2505.09388.

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. URL https://arxiv.org/abs/2508.15487.

Longhui Yu, Weisen Jiang, Han Shi, Jincheng Yu, Zhengying Liu, Yu Zhang, James T. Kwok, Zhenguo Li, Adrian Weller, and Weiyang Liu. MetaMath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023. URL https://arxiv.org/abs/2309.12284.

Sihyun Yu, Sangkyung Kwak, Huiwon Jang, Jongheon Jeong, Jonathan Huang, Jinwoo Shin, and Saining Xie. Representation alignment for generation: Training diffusion transformers is easier than you think. *arXiv preprint arXiv:2410.06940*, 2024. URL https://arxiv.org/abs/2410.06940.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling Reasoning in Diffusion Large Language Models via Reinforcement Learning. *arXiv preprint arXiv:2504.12216*, 2025. URL https://arxiv.org/abs/2504.12216.

Kaiwen Zheng, Huayu Chen, Haotian Ye, Haoxiang Wang, Qinsheng Zhang, Kai Jiang, Hang Su, Stefano Ermon, Jun Zhu, and Ming-Yu Liu. DiffusionNFT: Online diffusion reinforcement with forward process. *arXiv preprint arXiv:2509.16117*, 2025. URL https://arxiv.org/abs/2509.16117.

Fengqi Zhu, Rongzhen Wang, Shen Nie, Xiaolu Zhang, Chunwei Wu, Jun Hu, Jun Zhou, Jianfei Chen, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. LLaDA 1.5: Variance-Reduced Preference Optimization for Large Language Diffusion Models. *arXiv preprint arXiv:2505.19223*, 2025. URL https://arxiv.org/abs/2505.19223.

A TRAINING RECIPE AND MODEL ARCHITECTURES

This section collects the architectures, training stages, and optimization settings. All tasks use the same representation and conditioning design; Table 6 specifies their training durations. Detailed representation-learning equations, supervised losses, and NFT updates appear in Appendices C, B, and E, respectively.

A.1 ARCHITECTURES AND FIXED REPRESENTATIONS

The teacher is Qwen3-4B-Instruct-2507, with hidden width 2,560 and a 151,936-token vocabulary. We learn task-specific projections of its post-block activations at layers 16, 24, and 32, with output widths 256, 256, and 512. Concatenating these covariance-whitened features gives the 1,024-dimensional representation used for both teacher prompt conditions and answer targets. Projectors are learned before flow training and then fixed; their reconstruction decoders and frozen Qwen suffixes are used only during representation learning (Appendix C). Teacher activations and projected targets can be precomputed, so the later training stages do not require live Qwen Transformer calls.

Table 5: LFRM architectures. Both flow models use the ELF backbone, shared between denoising and token decoding. The prompt encoder is a separate bidirectional Transformer; its Qwen vocabulary lookup is frozen from initialization onward.

Component	LFRM-B	LFRM-L
ELF backbone: blocks / width / heads	12 / 768 / 12	32 / 1,280 / 16
ELF input bottleneck	512	512
Prompt Transformer: blocks / width / heads	6 / 768 / 12	6 / 1,280 / 16
Prompt frontend	$2560 \to 512 \to 768$	$2560 \to 512 \to 1280$
Prompt output width	1,024	1,024
Frozen Qwen token lookup	$151{,}936 \times 2{,}560$	
Teacher layers / projected widths	16/24/32 / 256/256/512	
Canvas length / latent width	1,024 / 1,024	

ELF backbone and token head. Table 5 gives the model sizes. Both backbones use bidirectional attention, rotary position encoding, and SwiGLU blocks with MLP ratio four. The flow head predicts 1,024-dimensional clean latents. A separate token head maps the backbone output through 1,024 decoder features to the native vocabulary, sharing the Transformer with denoising mode. Answer self-conditioning supplies a second latent channel. Time, mode, and SCCFG embeddings condition the backbone; the inference-only duplication of time embeddings is described in Section 4.3.

Compact prompt encoder. The encoder first looks up each question token in Qwen's frozen $151{,}936 \times 2{,}560$ input table. Two learned linear maps, $2560 \to 512 \to d_{\text{model}}$, have no intervening nonlinearity. Six bidirectional ELF-style blocks use rotary positions and MLP ratio four, followed by RMSNorm and a 1,024-dimensional output projection. The first frontend map has no bias; the second and output maps have biases. The LFRM-L contextual encoder contains 121,333,180 trainable parameters, excluding the frozen lookup. The full 36-block Qwen Transformer contains 3,633,511,936 parameters excluding its tied vocabulary maps, which contain 388,956,160 parameters. Only question-prefix tokens enter the encoder; padding is masked and padded outputs are zero. The lookup stays fixed during prompt imitation, joint training, and NFT, and is excluded from optimizers and prompt EMAs. Learned conditioning thus executes the compact encoder and vocabulary lookup once per response, without the Qwen Transformer.

A.2 TRAINING STAGES AND DURATIONS

Training populations and epoch units. The GSM8K reasoning runs share 2,434,330 answer rows and 275,661 unique prompts. The MATH runs use the MATH/augmented-MATH portion of OpenMathInstruct-2's five-million-row split (Toshniwal et al., 2024), yielding 3,967,527 eligible answer rows and 508,953 unique prompts. Coding uses 4,893,784 eligible OpenCodeInstruct answer rows and 4,772,508 unique prompts. Different solutions to one question remain separate

Table 6: Training stages by task. Flow-training epochs count passes over answer rows; prompt-MSE epochs count passes over unique prompts and do not advance the flow-training epoch. Joint epochs are additional to the first 12 flow epochs. The last column gives the NFT checkpoint used for reporting, in successful optimizer updates, rather than a further epoch count.

Task / model	Frozen-Qwen flow epochs	Prompt-MSE epochs	Joint epochs (flow endpoint)	Reported NFT update
GSM8K / LFRM-B	12	60	6 (18)	300
GSM8K / LFRM-L	12	30	9 (21)	500
MATH / LFRM-L	12	10	9 (21)	600
OpenCodeInstruct / LFRM-L	12	10	1 (13)	100

flow-training rows, whereas standalone prompt imitation weights each unique prompt equally. Native Qwen chat sequences must fit the 1,024-token canvas; overlength rows are excluded rather than truncated. Supervised losses include answer content and one terminal token, excluding prompt and padding positions.

GSM8K training-data provenance. Both GSM8K models use the same mixture of the original `train` splits of OpenMathInstruct-2 (Toshniwal et al., 2024), Orca-Math (Mitra et al., 2024), and MetaMathQA (Yu et al., 2023). OpenMathInstruct-2 contributes only rows whose `problem_source` is `gsm8k` or `augmented_gsm8k`; MetaMathQA contributes only types beginning with `GSM`. Orca-Math contributes all eligible rows. Table 7 gives the final training counts. Solutions to the same question remain separate training rows, with no additional source reweighting.

Table 7: GSM8K training mixture after filtering and question-group splitting. Counts are answer rows, shared by LFRM-B and LFRM-L. All sources originate from their original training splits.

Source	Retained subset	Training rows
OpenMathInstruct-2	`gsm8k`	292,485
	`augmented_gsm8k`	1,841,419
Orca-Math	Eligible train rows	147,464
MetaMathQA	`GSM` types	152,962
Total		2,434,330

Filtering and formatting. Corpus construction requires a nonempty question and solution and an extractable numerical final answer, allowing signed integers, decimals, and fractions. OpenMathInstruct-2 supplies its expected answer; Orca-Math and MetaMathQA answers are extracted from the solution. The builder stores a standardized response ending in one `#### answer` line for the initial length filter. This filter uses the `inclusionAI/LLaDA-MoE-7B-A1B-Base` tokenizer, requiring at most 256 prompt tokens and at most 1,024 prompt-plus-response tokens, including EOS. For Qwen representation extraction, the retained `problem` and `solution` fields are rendered with its native chat template; the question instruction requests a boxed final answer. All 2,434,330 rows fit: the saved Qwen token cache has maximum length 1,019 and zero truncated rows.

Splitting and benchmark screening. Question keys normalize Unicode, case, quotation marks, and whitespace; MetaMathQA also links rewrites through `original_question`. Exact matches and near-duplicate candidates retrieved by MinHash/LSH are merged across sources. Near-duplicate splitting uses substring coverage at least 0.90, or word 5-/8-gram Jaccard similarity at least 0.55/0.40. A deterministic hash with seed 20260702 selects 10% of source-local question keys for validation, and their entire merged groups are held out. The resulting split has 171,556 training groups and 25,949 validation groups (568,999 answer rows); the training population contains 275,661 distinct Qwen prompt prefixes. A 10% training-row cap on MetaMathQA removes no further rows, since its retained share is 6.28%.

Before splitting, the builder screens question texts against all 1,319 official GSM8K test questions, using normalized exact matching, substring coverage at least 0.85, and the same 5-/8-gram thresholds over candidates retrieved through shared n-grams. This removes 128 OpenMathInstruct-2, two Orca-Math, and 92 MetaMathQA candidate rows. The final Qwen-cache audit records zero

whitespace-normalized exact question matches to GSM8K test; the saved training and validation group IDs are disjoint. These checks establish the stated lexical screening, rather than semantic decontamination or absence of overlap in the teacher's pretraining data.

Stage 1: flow training with frozen teacher conditioning. Initialize the ELF backbone and token decoder with seed 42 and train for 12 answer-row epochs using fixed Qwen prompt representations. Both answer targets and prompt conditions use the learned projectors. Optimize the mixed flow/decoder-CE objective, choosing decoder mode with probability 0.2 and flow mode otherwise. All representation groups share a synchronous scalar time $t = \mathrm{sigmoid}(-1.5 + 0.8\xi)$, $\xi \sim \mathcal{N}(0, 1)$, with flow noise scale two. Prompt positions remain clean; bootstrap and SCCFG auxiliary targets are detached. Appendix B gives the exact corruptions, masks, guidance distribution, and loss reductions.

Stage 2: standalone prompt imitation. Hold the flow model and representation fixed and initialize the compact prompt encoder's trainable layers with seed 42, using the frozen Qwen token table from the outset. Writing the projected question-only teacher states as $c_{\mathrm{T}}(q)$, minimize

$$\mathcal{L}_{\mathrm{prompt}} = \mathbb{E}_q \left[\frac{1}{L_q d} \sum_{i=1}^{L_q} \| c_{\phi,i}(q) - c_{\mathrm{T},i}(q) \|_2^2 \right]. \tag{10}$$

The loss first averages valid positions and latent coordinates within a prompt, then weights prompts equally. Prompt-MSE training has its own epoch counter and does not change the epoch-12 flow checkpoint. Table 6 gives the task-specific durations; prompt EMA 0.999 initializes the joint stage.

Stage 3: joint adaptation. Replace the teacher condition with the learned encoder and continue flow training with differentiable question-prefix latents and fixed answer targets. Preserve the flow model's raw weights, optimizer state, and independent EMAs; initialize the prompt model from its selected standalone EMA and start fresh prompt optimizers. The flow/decoder objective updates both trainable networks through the main conditioned forward, while auxiliary self-conditioning and guidance targets remain detached. Mathematical reasoning uses no additional prompt-MSE loss: LFRM-B trains six more epochs to epoch 18, and LFRM-L trains nine to epoch 21. Coding trains one more epoch to epoch 13 and uses the regularizer below. All three stages retain the same prompt architecture and frozen token table.

Supervised optimization. Flow training, standalone prompt imitation, and joint adaptation use effective batch 512, constant learning rate 0.002, zero weight decay, and gradient clipping at norm one, applied separately to the flow and prompt models during joint training. Eligible matrices use Muon (Jordan, 2024) with momentum 0.95 and five FP32 Newton–Schulz iterations. Our implementation adapts the `muon-optimizer` package, with FP32 orthogonalization and Optax-compatible momentum bias correction and shape scaling. Remaining flow-model parameters use the auxiliary Nesterov-Adam optimizer; remaining prompt parameters use ordinary AdamW, with betas $(0.9, 0.999)$ and epsilon 10^{-8}. Trainable parameters and optimizer state use FP32, with compiled BF16 forward computation. Each trainable component maintains independent raw weights and EMAs with decays 0.99, 0.999, and 0.9999, updated at optimizer boundaries. Evaluation selects flow and prompt states independently; Appendices F and H specify the selectors for the reported results.

A.3 Coding joint-training regularizer

The coding epoch-13 checkpoint follows one additional joint epoch (9,558 optimizer updates), starting from the epoch-12 flow checkpoint and the prompt-epoch-10 EMA 0.999 state. Unlike the math joint stage, coding retains an auxiliary teacher-prompt MSE. Targets are the fixed, cached Qwen prompt representations; the loss averages valid prompt positions and latent coordinates within each training row, then weights rows equally. Only the prompt encoder receives this auxiliary gradient; answer representations and the vocabulary lookup remain frozen.

At each optimizer update, let g_{down} and g_{MSE} denote the accumulated, data-parallel-mean prompt gradients from the downstream and auxiliary objectives. We combine them as $g_\phi = g_{\mathrm{down}} + \lambda g_{\mathrm{MSE}}$,

with $\lambda = 2\,\overline{\|g_{\text{down}}\|_2}/\max(\overline{\|g_{\text{MSE}}\|_2}, 10^{-12})$, where bars denote exponential moving averages with decay 0.99. The target 2:1 ratio concerns smoothed gradient norms, not scalar loss values. The combined prompt gradient is clipped at norm one. Joint training otherwise retains the synchronous clocks and supervised optimization settings above. The subsequent coding NFT stage uses no auxiliary prompt MSE.

A.4 NFT FINE-TUNING

Initialize current, old, and reference policies from the selected supervised endpoint, with fresh NFT optimizers. The current flow backbone and prompt body are trainable. The Qwen vocabulary lookup and representation projectors remain fixed, as do the decoder-specific mode tokens, output projection, and vocabulary-head parameters. The shared backbone continues to change, so freezing these decoder-specific parameters does not freeze the entire decoding function. Old and reference policies receive no optimizer gradients; each retains its own prompt encoder.

Both trainable components use constant learning rate 10^{-4}, zero weight decay, and separate norm-one gradient clipping. The flow backbone uses Muon with auxiliary Nesterov-Adam; the prompt body uses Muon/AdamW. Computation uses BF16. Collection uses 32 denoising steps, synchronous identity clocks, and SCCFG two; optimization also retains synchronous scalar-time conditioning. Each round collects 24 prompt groups with 15 generated endpoints and one gold endpoint per group, followed by eight independently corrupted records per retained endpoint. Filtering makes the update size variable, so the supervised batch size 512 does not apply to NFT. Appendix E specifies rewards, filtering, loss normalization, and old-policy updates. Table 6 lists the reported NFT endpoints. Evaluation uses flow/prompt EMA $0.99/0.99$ for math and $0.9/0.9$ for code, with asynchronous clocks only at inference.

Coding reward population. Coding NFT starts from joint epoch 13 with flow/prompt EMA $0.9999/0.9999$ and fresh optimizers, retaining the objective and optimization settings above. We select 4,096 unique OpenCodeInstruct training prompts and reserve 256 disjoint development prompts, with selection seed 20260926. Eligible rows explicitly specify the tested interface, have unchanged native assertion tests and successful reference metadata, and pass reference execution in the reward harness. Prompt deduplication and exact held-out benchmark prompt exclusion precede selection. The training tests supply binary execution rewards; HumanEval/MBPP evaluation tests are never used for training. Gold endpoints receive reward one, generated programs receive 0.75 if all retained assertions pass and zero otherwise. Synthetic training tests remain imperfect measures of correctness. Execution is isolated, network-disabled, and resource-limited, with a five-second timeout per assertion. Both flow and prompt models train jointly, without the supervised stage's prompt-MSE regularizer. We report successful NFT update 100.

NFT checkpoint selection. We compare updates 100/200/300/400 for GSM8K LFRM-B, 100/300/500 for GSM8K LFRM-L, and 200/400/600 for MATH LFRM-L. Selection uses mean accuracy over seeds 42/123 on fixed development panels of 1,000 GSM8K and 500 MATH problems (Appendix D), with async $(2.5, 2, 1.5)$, 32 steps, and SCCFG two. Update 300 for GSM8K LFRM-B and 500 for GSM8K LFRM-L were selected based on validation accuracy; Math-Verify selects update 600 for MATH LFRM-L. For coding, update 100 with flow/prompt EMA $0.9/0.9$ was selected under computation resource constraints.

B LFRM OBJECTIVE AND CONDITIONAL ENCODING

B.1 FLOW AND TOKEN-DECODING OBJECTIVES

Canvas and scoring positions. Training sequences satisfy $L_q + L_a \le L = 1024$, with latent width $d = 1024$. For example b, the mask M_{bi} equals one on $L_{q,b} < i \le L_{q,b} + L_{a,b}$ and zero elsewhere. It includes the answer's terminal token and excludes prompt and padding positions. Answer latents and velocities at answer index j occupy canvas position $i = L_{q,b} + j$.

Time and self-conditioning. For a flow row, sample $t = \mathrm{sigmoid}(-1.5 + 0.8\xi)$ with $\xi \sim \mathcal{N}(0,1)$ and use noise scale $\sigma = 2$. A detached bootstrap pass with zero answer self-conditioning produces $\widehat{z}_{\mathrm{boot}}$. With $b_{\mathrm{sc}} \sim \mathrm{Bernoulli}(0.5)$, the main pass receives $\widehat{z}_{\mathrm{sc}} = b_{\mathrm{sc}}\widehat{z}_{\mathrm{boot}}$ in answer self-conditioning positions. Prompt states are restored to the clean condition independently of b_{sc}. The endpoint predictor and velocity denominator are

$$\widehat{z}_{\mathrm{clean}} = F_\theta(z_t, t; c, \widehat{z}_{\mathrm{sc}}, g), \qquad \Delta_t = \max(1 - t, 0.05). \tag{11}$$

The velocity conversion in Section 4 uses this denominator for both prediction and target, so the executed target differs from $u_t = z_{\mathrm{clean}} - 2\epsilon$ where the clamp is active.

Masked flow regression. The normalized counterpart of Eq. (2), before the SCCFG target adjustment, is

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}\left[\frac{\sum_{bi} M_{bi}\|v_{\theta,bi} - u^{\mathrm{stab}}_{bi}\|_2^2}{d\sum_{bi} M_{bi}}\right]. \tag{12}$$

Here u^{stab}_{bi} is the target at example b's sampled time t_b. All latent coordinates contribute. SCCFG replaces this target by $\widetilde{u}_{bi}$ below.

SCCFG target. Draw the guidance scale g from $\log(1+g) \sim \mathrm{Uniform}(\log 1.5, \log 6)$, giving $g \in [0.5, 5]$. Let v^0 and v^1 denote auxiliary velocity predictions without and with answer self-conditioning. The main loss uses

$$\widetilde{u}_t = \mathrm{sg}\big[u^{\mathrm{stab}}_t + b_{\mathrm{sc}}(1 - g^{-1})(v^1 - v^0)\big]. \tag{13}$$

The question is present in both auxiliary branches. All auxiliary predictions and targets are detached; during joint training, gradients enter the prompt encoder through the main conditioned forward. At inference, the preceding denoising prediction supplies recurrent self-conditioning and the requested guidance scale is passed to the network. Guidance scale one retains recurrent self-conditioning. Prompt-drop probability is zero in supervised training, and all reported evaluations use prompt CFG one.

Decoder corruption and mixed objective. Each row independently draws $\delta^{\mathrm{dec}}_b \sim \mathrm{Bernoulli}(0.2)$ to choose decoder mode. At each position of a decoder row, sample

$$\lambda_{bi} = \mathrm{sigmoid}(0.8 + 0.8\xi_{bi}), \qquad z^{\mathrm{dec}}_{bi} = \lambda_{bi} z_{\mathrm{clean},bi} + (1 - \lambda_{bi})\epsilon'_{bi},$$

where ξ_{bi} and ϵ'_{bi} are standard Gaussian. Decoder noise scale is one. The model receives decoder mode, time one, zero answer self-conditioning, and the clean question prefix. Token CE predicts the original token at each position, without a next-token shift.

The valid-token expectation in Eq. (4) samples a training microbatch and then uniformly selects a valid answer position across its rows. With $c_b = c(q_b)$, its explicit form is

$$\mathcal{L}_{\mathrm{CE}} = \mathbb{E}\left[-\frac{\sum_{bi} M_{bi} \log p^{\mathrm{tok}}_{\theta,i}(y_{bi} \mid z^{\mathrm{dec}}_b, c_b)}{\sum_{bi} M_{bi}}\right]. \tag{14}$$

Here the answer-indexed output $p^{\mathrm{tok}}_{\theta,j}$ in the main text corresponds to canvas position $i = L_q + j$. This averages tokens within each microbatch, rather than weighting solutions equally. Decoder and flow rows use separate output heads on the same backbone, with mode probabilities 0.2 and 0.8, respectively.

For valid response mask M_{bi}, including the terminal token and excluding question/padding positions, a rank-local microbatch minimizes

$$\mathcal{L}_{\mathrm{LFRM}}^{\mathrm{micro}} = \frac{\sum_{bi} M_{bi} \left[(1-\delta_b^{\mathrm{dec}}) \|v_{\theta,bi} - \widetilde{u}_{bi}\|_2^2 / d - \delta_b^{\mathrm{dec}} \log p_{\theta,i}^{\mathrm{tok}}(y_{bi} \mid z_b^{\mathrm{dec}}, c_b) \right]}{\sum_{bi} M_{bi}}. \tag{15}$$

Here $\widetilde{u}_{bi}$ is the SCCFG-adjusted target at example b's sampled time. There is one denominator across both row types, with no additional normalization by mode probability. Averaging over mode assignments gives $0.8\mathcal{L}_{\mathrm{FM}}^{\mathrm{SCCFG}} + 0.2\mathcal{L}_{\mathrm{CE}}$ under this common normalization. The flow term averages all 1,024 coordinates, preserving the relative 256/256/512 group widths. Gradients are averaged over data-parallel ranks and accumulation microsteps. Both models share their backbone between flow and token-decoder modes (Table 5).

Sampling initialization and termination. Given q, compute $c(q)$ once and initialize all $L - L_q$ available answer positions independently from $\mathcal{N}(0, \sigma^2 I)$. The prompt remains clean throughout. For S denoising calls, Euler evaluates at τ_r, $r = 0, \ldots, S-1$, and advances to $\tau_S = 1$; Appendix D specifies the grid. Answer self-conditioning starts at zero and subsequently carries the preceding clean prediction. The requested SCCFG scale is supplied to each call. A separate call through the shared backbone and token head decodes all answer positions in parallel; the first terminal token determines the returned length. Denoising NFE counts only the S denoising calls.

The prompt architecture, imitation objective, and stage durations are collected in Appendix A; Eq. (10) defines the standalone prompt loss. Figure 3's evaluation cohorts are specified in Appendix B.3.

B.2 Conditional representation: proof and scope

Lemma 3 (General conditional regression)**.** *Let Q be the prompt, $W = (Z_t, t)$ the noisy answer and time, and U a square-integrable forward-regression target. Fix a deterministic encoder $C_\phi = P_\phi(Q)$ and the joint law of (Q, W, U). Define $m_Q = \mathbb{E}[U \mid W, Q]$ and $m_\phi = \mathbb{E}[U \mid W, C_\phi]$. Let $\mathcal{R}_Q^\star$ and $\mathcal{R}_\phi^\star$ be the minimum squared regression risks among all measurable predictors given (W, Q) and (W, C_ϕ), respectively. Then*

$$\mathcal{R}_\phi^\star - \mathcal{R}_Q^\star = \mathbb{E}\|m_Q - m_\phi\|_2^2. \tag{16}$$

In particular, the encoding attains the full-prompt optimum exactly when it preserves the posterior mean target almost surely.

Proof of Lemma 3. Because C_ϕ is a deterministic function of Q, the information in (W, C_ϕ) is contained in that of (W, Q). The tower property gives $m_\phi = \mathbb{E}[m_Q \mid W, C_\phi]$. For any square-integrable predictor $f(W, C_\phi)$, conditional-expectation orthogonality yields

$$\begin{aligned} \mathbb{E}\|U - f(W, C_\phi)\|_2^2 = \mathbb{E}\|U - m_Q\|_2^2 + \mathbb{E}\|m_Q - m_\phi\|_2^2 \\ + \mathbb{E}\|m_\phi - f(W, C_\phi)\|_2^2. \end{aligned} \tag{17}$$

The first term is $\mathcal{R}_Q^\star$. Minimizing over f removes the final term and gives Eq. (16). Equality of the two optimal risks is equivalent to $m_Q = m_\phi$ almost surely. □

Proof of Lemma 1. For fixed $0 < t < 1$, differentiating the Gaussian mixture induced by the forward path gives

$$s^\star(z, t \mid q) = \frac{t\,\mathbb{E}[Z_{\mathrm{clean}} \mid Z_t = z, Q = q] - z}{\sigma^2 (1-t)^2}. \tag{18}$$

The same identity holds with $C = P_\phi(Q)$ in place of Q. Subtracting gives Eq. (5); its scalar factor is nonzero on $(0, 1)$. Moreover, $Z_{\mathrm{clean}} - \sigma\epsilon = (Z_{\mathrm{clean}} - Z_t)/(1-t)$ on the forward path, so substituting Eq. (18) into its conditional expectation gives Eq. (3). Equal scores therefore give equal optimal unguided fields. Starting from the same Gaussian prior, their exact flows have the same endpoint law under the usual existence and uniqueness assumptions, interpreting the endpoint as a limit if necessary. Applying Lemma 3 to the endpoint or velocity target also identifies posterior-mean preservation with equality of the optimal regression risks. □

Posterior sufficiency and teacher imitation. For the random counterpart U of u_t^{stab}, namely $U = (Z_{\mathrm{clean}} - Z_t)/\Delta_t$, the posterior mean is $(\mathbb{E}[Z_{\mathrm{clean}} \mid Z_t, t, Q] - Z_t)/\Delta_t$. Thus preserving the posterior mean clean endpoint suffices for this regression target, including where the deterministic denominator clamp is active. Equality of the conditional distributions of Z_{clean} given Q and $P_\phi(Q)$ is a sufficient condition. The criterion is independent of the coordinate system used by the teacher: an invertible change of encoding coordinates preserves the available information and can be compensated by the denoiser. The finite network class may limit that compensation. Prompt MSE supplies a useful initialization for the existing denoiser, while subsequent training can adapt the interface through the downstream objective.

For a concrete example, suppose $c_{\mathrm{T}}(q)$ preserves the posterior mean. For any scalar α, define $c'_i(q) = c_{\mathrm{T},i}(q) + \alpha \mathbf{1}_d$ at each valid prompt position. Subtracting the known offset recovers the teacher encoding exactly, so c' preserves the same information and ideal conditional score, while its normalized teacher-imitation MSE in Eq. (10) is α^2 and can be arbitrarily large. This construction illustrates the information criterion; it does not assert that a fixed finite denoiser can compensate for arbitrary encoder errors without adaptation.

Noise, masks, and optimization. Conditioning Eq. (17) on time defines an excess risk $\Delta_\phi(t) = \mathbb{E}[\|m_Q - m_\phi\|_2^2 \mid t]$. The identity imposes no monotonicity on $\Delta_\phi(t)$. For a token-weighted objective, the same argument applies under the induced probability measure over scored positions, including the position index among the observed variables; all conditional expectations must use that same measure. The lemma assumes that the answer targets and corruption law do not depend on the encoder parameters, as in our joint stage. It justifies the conditional-model parameterization and characterizes its information requirement. It does not assert that joint training from initialization reaches the optimum, or identify a learned SCCFG sampling field with the unguided posterior field.

B.3 Prompt-learning curves and numerical results

Table 8 contains every point in Figure 3. Points are means over seeds 42/123, evaluated with async $(2.5, 2, 1.5)$, 32 denoising steps, SCCFG 2, and backbone EMA 0.9999. Prompt EMA is 0.999 at the MSE swap and 0.9999 during joint training. GSM8K uses numerical-answer scoring and MATH500 uses Math-Verify. Error bars show pointwise 95% intervals from 4,000 problem-bootstrap replicates; brackets in the table give their bounds. Epoch 12 appears twice because prompt substitution does not advance the flow-training epoch; the two rows have identical backbone weights. The axes start at zero, but no epoch-zero accuracy was measured or plotted.

The curve samples flow epochs 4, 8, and 12 before prompt substitution; epoch 12 immediately after substitution; and epochs 16/18 for LFRM-B or 16/20/21 for LFRM-L. All 20 points retain their saved predictions and scores. The joint-from-start comparison in Figure 2 instead uses the common $t = \tau^2$ clock and backbone EMA 0.999 for its six-epoch controls; its four-seed endpoint means are not points on these curriculum curves. Training durations are summarized in Table 6.

Table 8: Numerical values for Figure 3. Correct counts for seeds 42/123 and the benchmark size N specify the exact per-seed accuracies. Mean pass@1 and sample seed SD (parentheses) are percentages; brackets give the plotted 95% problem-bootstrap intervals.

Epoch	Prompt	Correct_{42}	Correct_{123}	Mean (SD)	95% CI
GSM8K, LFRM-B; $N = 1319$					
4	Qwen	92	86	6.75 (0.32)	$[5.76, 7.81]$
8	Qwen	401	424	31.27 (1.23)	$[29.11, 33.43]$
12	Qwen	514	503	38.55 (0.59)	$[36.28, 40.83]$
12	MSE swap	398	381	29.53 (0.91)	$[27.29, 31.69]$
16	Joint	482	462	35.78 (1.07)	$[33.55, 38.02]$
18	Joint	495	483	37.07 (0.64)	$[34.76, 39.39]$
GSM8K, LFRM-L; $N = 1319$					
4	Qwen	212	202	15.69 (0.54)	$[14.22, 17.21]$
8	Qwen	755	743	56.79 (0.64)	$[54.40, 59.14]$
12	Qwen	804	820	61.56 (0.86)	$[59.17, 63.80]$
12	MSE swap	647	660	49.55 (0.70)	$[47.12, 51.90]$
16	Joint	728	701	54.17 (1.45)	$[51.78, 56.52]$
20	Joint	761	763	57.77 (0.11)	$[55.38, 60.12]$
21	Joint	783	780	59.25 (0.16)	$[56.67, 61.68]$
MATH500, LFRM-L; $N = 500$					
4	Qwen	30	38	6.80 (1.13)	$[5.10, 8.70]$
8	Qwen	61	65	12.60 (0.57)	$[10.30, 15.10]$
12	Qwen	82	76	15.80 (0.85)	$[13.10, 18.50]$
12	MSE swap	64	65	12.90 (0.14)	$[10.50, 15.40]$
16	Joint	84	92	17.60 (1.13)	$[14.90, 20.40]$
20	Joint	98	91	18.90 (0.99)	$[16.00, 21.90]$
21	Joint	85	94	17.90 (1.27)	$[15.10, 20.90]$

C Representation construction and ablations

C.1 Construction and covariance whitening

Feature geometry. The frozen teacher's post-block residual is $h_j^{(\ell)} = h_\psi^{(\ell)}(q, a_{\le j}) \in \mathbb{R}^{d_{\text{teacher}}}$, with $d_{\text{teacher}} = 2560$. Each encoder $W_\ell^{\text{enc}} \in \mathbb{R}^{d_{\text{teacher}} \times d_\ell}$ produces $z_{\text{clean},j}^{(\ell)} = (h_j^{(\ell)} - \mu_\ell) W_\ell^{\text{enc}}$, and its reconstruction decoder has shape $W_\ell^{\text{rec}} \in \mathbb{R}^{d_\ell \times d_{\text{teacher}}}$. Brackets in Section 4.1 concatenate layer blocks in ascending order. The same maps initially encode prompt states, which cannot access subsequent answers under the teacher's causal mask.

Whitening and initialization. The linear encoder and reconstruction decoder admit changes of latent scale that leave reconstruction unchanged. We fix this freedom before applying isotropic diffusion noise. For each selected layer, estimate the mean μ_ℓ and covariance Σ_ℓ over training assistant-content tokens. Write $\Sigma_\ell = U_\ell \Lambda_\ell U_\ell^\top$, with eigenvalues in descending order, and define

$$\widetilde{\Sigma}_\ell = U_\ell \operatorname{diag}\big(\max(\lambda_{\ell i}, 10^{-5})\big) U_\ell^\top.$$

We parameterize the encoder by

$$Q_\ell = \operatorname{qf}(R_\ell), \qquad W_\ell^{\text{enc}} = \widetilde{\Sigma}_\ell^{-1/2} Q_\ell, \qquad (W_\ell^{\text{enc}})^\top \widetilde{\Sigma}_\ell W_\ell^{\text{enc}} = I_{d_\ell}, \tag{19}$$

where qf is a differentiable thin QR factorization and R_ℓ is trainable. This maintains a nondegenerate coordinate scale while the subspace changes. Initialize $Q_{\ell,0} = U_{\ell,:d_\ell}$, $W_{\ell,0}^{\text{enc}} = \widetilde{\Sigma}_\ell^{-1/2} Q_{\ell,0}$, and $W_{\ell,0}^{\text{rec}} = Q_{\ell,0}^\top \widetilde{\Sigma}_\ell^{1/2}$. The resulting reconstruction is ordinary rank-d_ℓ PCA:

$$\widehat{h}_{j,0}^{(\ell)} = \mu_\ell + (h_j^{(\ell)} - \mu_\ell) U_{\ell,:d_\ell} U_{\ell,:d_\ell}^\top.$$

Starting from this initialization, we optimize Eq. (21), passing gradients through the thin QR factorization while leaving the decoder unconstrained. The whitening identity in Eq. (19) refers to the fixed regularized training covariance; it does not imply identity covariance on every minibatch, on question features, or across concatenated layer groups.

Teacher-predictive reconstruction. For each layer, the linear decoder reconstructs the original hidden width:

$$\widehat{h}_j^{(\ell)} = \mu_\ell + z_{\text{clean},j}^{(\ell)} W_\ell^{\text{rec}}. \tag{20}$$

The three reconstruction branches run frozen Qwen blocks 17–36, 25–36, and 33–36, respectively, followed by final normalization and the vocabulary head. We intervene on all assistant-content states at a single layer in each branch, retaining clean prefix/control states. Full-vocabulary soft cross-entropy scores positions that predict the next assistant-content token; the first assistant token, terminal token, prompt, and padding are excluded as targets. The objective is

$$\mathcal{L}_{\text{repr}} = \sum_{\ell \in \mathcal{S}} \mathbb{E}_{q,a,j} \left[- \sum_{w \in \mathcal{V}} p_j^{\text{teacher}}(w) \log p_{\ell,j}^{\text{recon}}(w) \right]. \tag{21}$$

Each layer is intervened on independently. Both distributions predict a_{j+1} from the state at a_j, whereas LFRM's token decoder predicts the same-position token. The clean teacher distribution is detached; gradients through the frozen suffix train only W_ℓ^{enc} and W_ℓ^{rec}. Teacher-to-reconstruction KL differs from this objective only by the fixed teacher entropy. These reconstruction decoders are separate from LFRM's token decoder and, together with the Qwen suffixes, are used only to learn the representation.

C.2 Clean-token decoding from the fixed PCA representation

We use the exact frozen, whitened top-1,024 PCA coordinates from Qwen layer 32 used by the single-layer generation arm in Figure 2. A separately trained decoder maps these coordinates through a learned $1{,}024 \to 1{,}024$ linear map, GELU, and a vocabulary projection to the same-position token. The PCA transform remains fixed and is applied in FP32 to cached BF16 teacher activations. Layer 32 is a deep-layer control within the teacher's 36 blocks.

The probe trains for two epochs (856 updates) on 218,857 rows, with seed 42, batch size 512, learning rate 0.002, Muon/auxiliary optimization, and FP32 vocabulary cross-entropy. Training uses four GPUs with gradients weighted by each rank's valid-token count. We evaluate final raw weights on 12,436 held-out solution rows from 4,647 prompt groups, comprising 2,112,672 assistant-content tokens. Training, development, and test splits are disjoint by prompt. Same-position targets include the first assistant token and exclude prompt, terminal, and padding tokens.

Table 9 reports token-weighted accuracy and cross-entropy. Intervals use 4,000 prompt-cluster bootstrap replicates (seed 20260920), retaining all rows and tokens within each resampled prompt; they quantify held-out sampling uncertainty, not training variability. These results demonstrate substantial clean-token recoverability from the actual PCA generation representation. They measure per-token reconstruction with a separate probe, not solution accuracy or the performance of LFRM's decoder, and do not isolate the cause of generation errors. The probe is not claimed to have converged after two epochs.

Table 9: Clean-token decoding from fixed PCA coordinates after two epochs, using final raw probe weights. Brackets give 95% prompt-cluster bootstrap intervals. The input is the representation used by the single-layer generation ablation.

Probe input	Token accuracy (%)	CE (nats/token)
Fixed layer-32 PCA	99.24 [99.19, 99.28]	0.0548 [0.0502, 0.0597]

C.3 Layer similarity and CKA

The CKA study uses disjoint discovery and confirmation panels of 4,096 problems each, totaling 8,192 equally weighted problems. For each problem, teacher-forced assistant activations are mean-pooled into eight relative-position bins and concatenated before computing linear CKA. Figure 5 shows the combined panel. The layer-16/17 CKA is 0.038; the selected-layer similarities are 0.124 (16/24), 0.543 (16/32), and 0.842 (24/32). We use the transition after layer 16 and regular spacing thereafter to motivate the representation, without claiming that CKA identifies an optimal triplet.

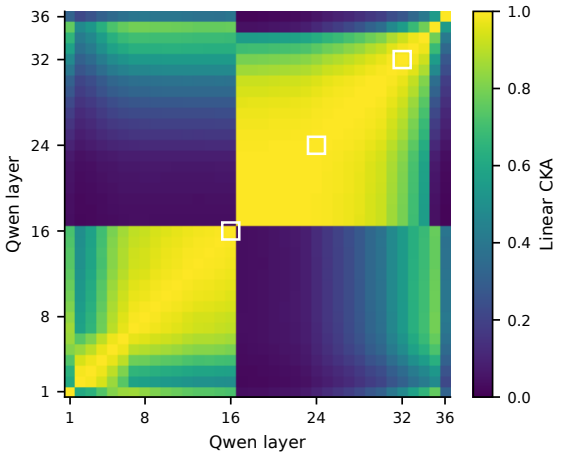

Figure 5: **Qwen layer structure.** Linear CKA of teacher-forced activations on the combined discovery and confirmation panels. White markers select layers 16, 24, and 32. The transition after layer 16 motivates retaining that layer, followed by regularly spaced deeper features.

C.4 Representation ablations and clock robustness

Figure 6 changes the evaluation clock to $t = \tau$, retaining the trained checkpoints, 32 steps, guidance, and EMA selectors; the identity-clock comparison shows that the epoch-six ordering persists under another schedule. The two-seed identity-clock means are 26.76%, 24.22%, 13.38%, and 8.68% for learned multilayer, PCA multilayer, PCA layer 32, and joint prompt training from initialization.

Learned multilayer leads PCA multilayer from epoch two onward under both clocks, and multilayer PCA leads layer-32 PCA at every epoch.

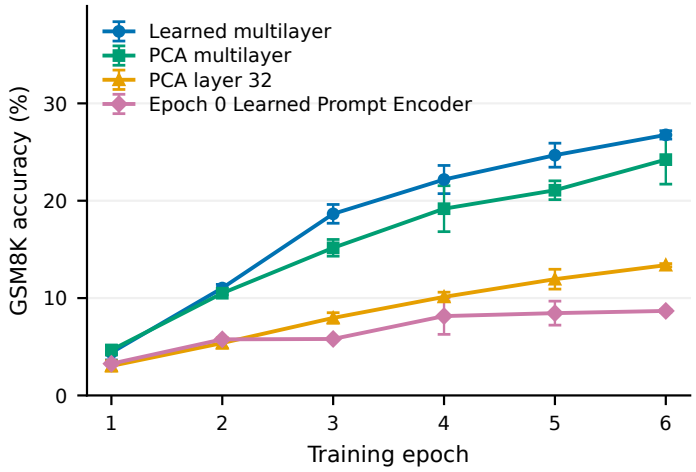

Figure 6: **Identity-clock robustness.** The four models from Figure 2, evaluated with $t = \tau$ and 32 steps. Points average seeds 42/123 (bars: seed SD). Each epoch uses one trained checkpoint per arm. Numerical values appear in Table 10.

All four arms start from the same seed-42 backbone initialization, use 2,434,330 training rows, and preserve effective batch 512. All answer representations are whitened and 1,024-dimensional. The first three arms use frozen Qwen prompt activations; the fourth uses learned multilayer answer latents and a randomly initialized prompt encoder trained jointly from epoch zero. All select backbone EMA 0.999, and the fourth selects prompt EMA 0.999. The PCA multilayer run changes from two to four GPUs during training while maintaining that batch size. Each arm has one training run; seed variation in the figures measures sampling randomness only. The comparisons isolate projector choice at fixed layers and layer choice under PCA, without estimating their interaction. Paired intervals in the main text use 4,000 problem-bootstrap replicates and the four matched epoch-six generation seeds.

Representation-ablation curves. Table 10 gives all points and error bars in Figures 2 and 6, followed by the four-seed epoch-six summary discussed in Sections 4.1 and 4.2. The curves use seeds 42/123; the endpoint summary adds 456/789. Error bars and parenthesized values are sample SDs across their respective generation seeds. Both use 32 steps and backbone EMA 0.999; The controls above apply to both clocks.

Table 10: Numerical learning curves and endpoint summary for Figures 2 and 6: mean GSM8K accuracy (%) with sample seed SD in parentheses. Curves use two seeds (42/123); the final block reports the four-seed epoch-six comparison (42/123/456/789). The final column trains the prompt encoder jointly from epoch zero; the other three retain Qwen prompt conditioning.

Epoch	Learned 16/24/32	PCA 16/24/32	PCA layer 32	Epoch-0 joint prompt training
$t = \tau^2$				
1	3.75 (0.05)	4.85 (0.75)	2.65 (0.11)	2.46 (0.05)
2	12.40 (0.27)	11.18 (0.16)	5.50 (0.70)	3.98 (1.02)
3	20.17 (0.54)	16.38 (0.32)	8.64 (0.43)	3.68 (0.16)
4	26.61 (0.86)	21.49 (0.80)	7.47 (0.05)	7.77 (1.02)
5	29.95 (0.21)	25.85 (0.21)	9.82 (0.16)	7.54 (1.45)
6	31.01 (0.43)	28.39 (0.27)	11.68 (0.21)	8.61 (0.05)
$t = \tau$				
1	4.40 (0.75)	4.66 (0.48)	3.03 (0.54)	3.26 (0.11)
2	11.03 (0.38)	10.50 (0.16)	5.38 (0.54)	5.76 (0.00)
3	18.65 (0.96)	15.16 (0.86)	7.96 (0.54)	5.80 (0.05)
4	22.18 (1.45)	19.18 (2.36)	10.12 (0.48)	8.15 (1.88)
5	24.68 (1.23)	21.08 (0.96)	11.94 (1.02)	8.45 (1.23)
6	26.76 (0.43)	24.22 (2.52)	13.38 (0.16)	8.68 (0.16)
Epoch-six summary: $t = \tau^2$, four seeds				
6	30.76 (0.39)	28.43 (0.19)	11.20 (0.56)	8.70 (0.33)

D INFERENCE CLOCKS AND RECONSTRUCTION DIAGNOSTICS

D.1 LOCAL CLOCKS, TIME EMBEDDINGS, AND SAMPLING GRID

Local corruption and time embeddings. The local clocks in Section 4.3 assign each group its own Gaussian corruption, $z_{\boldsymbol{t}}^{(\ell)} = t_\ell z_{\text{clean}}^{(\ell)} + (1 - t_\ell)\sigma\epsilon^{(\ell)}$. The three time embedders are identical copies of the selected scalar-time checkpoint weights:

$$\bar{e}_\theta^{\text{time}}(\boldsymbol{t}) = \frac{1}{3} \sum_{\ell \in \{16,24,32\}} e_{\theta,\ell}^{\text{time}}(t_\ell), \qquad e_{\theta,\ell}^{\text{time}} = e_\theta^{\text{time}}. \tag{22}$$

All other tensors are unchanged. The embedding outputs are weighted equally, irrespective of the 256/256/512 group widths, with the mean accumulated in FP64 before casting back to model dtype. The resulting embedding is a shared network condition, not a separate Transformer for each group. In particular, permuting the three exponents preserves this conditioning average while changing the coordinate groups to which the local velocity factors apply. Because the group widths differ, that permutation need not preserve total latent noise energy.

Sampling grid and conditioning. For S denoising steps, we set $\tau_0 = 0$, $\tau_S = 1$, and use the same logit-normal-quantile grid across schedules:

$$\tau_r = \text{sigmoid}\left[-1.5 + 0.8\Phi^{-1}(r/S)\right], \qquad 0 < r < S. \tag{23}$$

The base τ grid is fixed across schedule comparisons. We retain the full derivative $f'_\ell(\tau) = \gamma_\ell \tau^{\gamma_\ell - 1}$ in Eq. (6), with the endpoint denominator $\Delta_{t_{\ell,r}} = \max(1 - \tau_r^{\gamma_\ell}, 0.05)$. Question states remain clean, and recurrent self-conditioning carries the previous endpoint prediction to the next call. Changing the clock therefore alters both the integration grid in local time and the prediction history. One shared denoiser call updates all groups; the terminal token-decoding call remains separate.

D.2 SCHEDULE COMPARISONS

Tables 1 and 11 use LFRM-L epoch 21, 32 denoising steps, SCCFG 2, backbone/prompt EMA $0.9999/0.9999$, and seeds 42/123. Scores use numerical-answer accuracy for GSM8K and Math-Verify for MATH500 (Appendix F). Entries in Table 1 give mean pass@1 with sample SD across the two seeds; bold marks the highest mean among its four schedules.

Table 11 compares the four schedules from the main table on both validation and test sets. The 500-problem MATH validation set comes from the same original MATH test pool as MATH500, excluding its questions, whereas our GSM8K validation set comprises held-out original training questions. We therefore regard the MATH validation comparison as more directly matched to its benchmark's source distribution; the two validation sets need not favor the same schedule. The highest GSM8K validation mean instead uses $(1.5, 2, 2.5)$. Thus the main comparison supports the usefulness of time reparameterization and the chosen operating point, without identifying a universally optimal exponent vector.

Table 11: Four-schedule LFRM-L epoch-21 comparison: mean pass@1 (%) with sample seed SD in parentheses, over seeds 42/123 at 32 denoising steps and SCCFG 2. Validation sets contain 1,000 GSM8K and 500 MATH problems. Test columns include all entries and SDs from Table 1. Bold marks each column maximum, including ties.

	Validation		Test	
Clock powers	GSM8K	MATH	GSM8K	MATH500
$(1, 1, 1)$	65.00 (0.99)	15.40 (0.28)	52.96 (1.13)	15.50 (1.27)
$(2, 2, 2)$	72.70 (0.85)	17.80 (0.57)	57.51 (0.80)	17.40 (1.41)
$(1.5, 2, 2.5)$	**72.80** (1.13)	18.30 (0.14)	58.07 (0.64)	17.60 (1.98)
$(2.5, 2, 1.5)$	72.15 (0.35)	**18.70** (0.14)	**59.25** (0.16)	**17.90** (1.27)

For the 32-step test comparison, async $(2.5, 2, 1.5)$ minus sync $(2, 2, 2)$ gives +1.74 percentage points on GSM8K (paired 95% CI $[+0.72, +2.73]$) and +0.50 on MATH500 ($[-0.90, +1.90]$). Repeating the same two-seed comparison at 16 steps gives 55.57% versus 54.74% on GSM8K, a difference of +0.83 points ($[-0.15, +1.82]$), and 16.60% versus 16.90% on MATH500, a difference of

-0.30 points ($[-2.00, +1.40]$). All use SCCFG 2. These are pointwise paired problem-bootstrap intervals with 4,000 replicates, conditional on the fixed checkpoints and generation seeds; the standard deviations in the main table instead describe dispersion across full-benchmark seed accuracies.

D.3 Layer decomposition and reconstruction sensitivity

We probe a trained LFRM-B model by corrupting one group of clean gold-solution latents, predicting the clean representation, and decoding it to tokens. A matched control replaces the layer groups by eight random partitions of the same 256/256/512 widths, keeping the assigned times and injected Gaussian energy fixed. Figure 7 shows that the actual groups have different reconstruction sensitivities: perturbing layer-16 features is more damaging than perturbing deeper features, with the clearest contrast against layer 24. Random grouping attenuates these contrasts. The layer decomposition thus exposes structure that is obscured when coordinates are grouped arbitrarily.

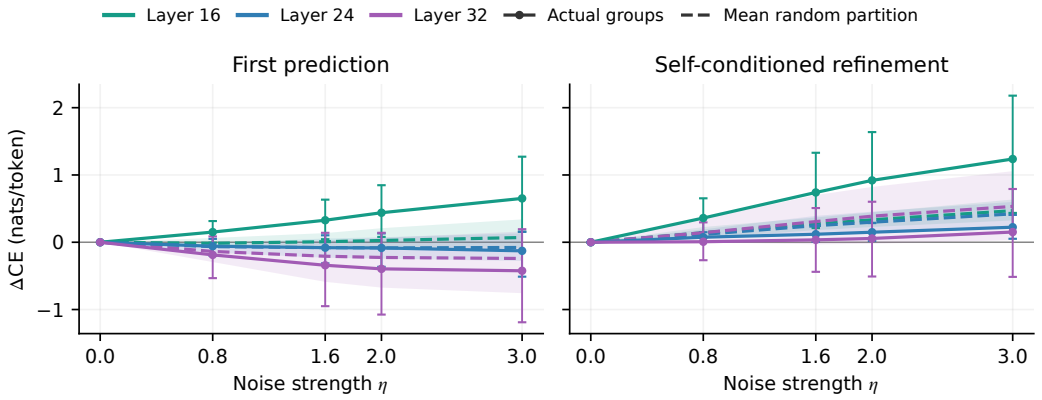

Figure 7: **Layer groups expose distinct reconstruction sensitivities.** LFRM-B epoch 18 on 64 GSM8K validation problems. Solid curves perturb actual layer groups; dashed curves average eight random partitions with matched widths, times, and injected Gaussian energy. Colors identify the corresponding layer or control slot. Decoder CE changes are relative to the same pass's clean-input prediction: first without answer self-conditioning (left), then after one refinement on the same corrupted input (right). Bars show 95% CIs; shading is the range of random-partition means, not a confidence interval. Table 12 gives every plotted value.

Reconstruction intervention. Figure 7 uses LFRM-B epoch 18, backbone/prompt EMA $0.9999/0.9999$, SCCFG 1, noise seeds 42/123, and a fixed 64-problem GSM8K validation panel. For a selected group ℓ, we set $t_\ell = 2/[2 + \eta\,\mathrm{RMS}(z^{(\ell)}_{\mathrm{clean}})]$ and replace its assistant-content coordinates by $t_\ell z^{(\ell)}_{\mathrm{clean}} + 2(1 - t_\ell)\epsilon^{(\ell)}$, for $\eta \in \{0.8, 1.6, 2, 3\}$. Other coordinates, question positions, terminal tokens, and padding remain clean. The model predicts the full clean latent, which its own token head decodes. The second pass reuses the same corrupted input with the first prediction as self-conditioning; it is not another integration step. CE averages gold-token losses within each problem, then noise seeds and problems equally. Negative changes mean lower CE than the clean-input *prediction* baseline, not a negative loss.

The random control uses eight fixed NumPy PCG64 partitions (seeds 20260918–20260925). Each control slot retains its true-group time and injected Gaussian energy; random-subset RMS is recorded but does not recalibrate the intervention. At $\eta = 3$, after self-conditioning, the true-group layer-16 minus layer-24 CE contrast exceeds the corresponding mean-random contrast by 0.967 nats/token (paired 95% CI $[0.168, 1.890]$). The layer-16 minus layer-32 excess is 1.149 ($[0.061, 2.241]$), with weaker evidence at lower noise and in the first prediction. Random partitions are averaged within problem before resampling; they are not additional independent problems.

Interpretation. For $0 < \tau < 1$, our chosen powers give $t_{16} < t_{24} < t_{32}$: layer 16 becomes clean relatively later, while all groups reach their endpoint together. One possible explanation is that the more sensitive layer-16 representation benefits from being reconstructed with cleaner deeper-layer

features available as context. This is a hypothesis about coordination between representations, rather than a consequence of the local probe. The probe motivates investigating layer-dependent denoising, but further experiments are needed to establish whether this proposed mechanism explains its sampling benefits.

Reconstruction-sensitivity curves. Table 12 reproduces the means, error bars, and shaded ranges in Figure 7. The zero-noise points are identically zero because each condition is compared with its own pass's clean-input baseline. Random-partition ranges describe the eight partition means, not uncertainty intervals.

Table 12: Numerical values for Figure 7, in nats/token. Actual-group entries give ΔCE and its 95% problem-bootstrap CI; random-group entries give the mean over eight matched partitions and the minimum/maximum partition means. Noise strength is η; the layer identifies the matched group width.

Layer	η	Actual group [95% CI]	Random mean [min, max]
First prediction			
16	0	$0.000\ [0.000, 0.000]$	$0.000\ [0.000, 0.000]$
16	0.8	$0.151\ [-0.006, 0.315]$	$-0.015\ [-0.053, 0.064]$
16	1.6	$0.327\ [0.045, 0.633]$	$0.008\ [-0.057, 0.136]$
16	2	$0.439\ [0.080, 0.848]$	$0.026\ [-0.075, 0.209]$
16	3	$0.651\ [0.158, 1.272]$	$0.071\ [-0.087, 0.340]$
24	0	$0.000\ [0.000, 0.000]$	$0.000\ [0.000, 0.000]$
24	0.8	$-0.059\ [-0.201, 0.050]$	$-0.065\ [-0.154, -0.010]$
24	1.6	$-0.081\ [-0.314, 0.103]$	$-0.080\ [-0.231, 0.055]$
24	2	$-0.085\ [-0.365, 0.131]$	$-0.084\ [-0.245, 0.075]$
24	3	$-0.128\ [-0.512, 0.154]$	$-0.078\ [-0.279, 0.134]$
32	0	$0.000\ [0.000, 0.000]$	$0.000\ [0.000, 0.000]$
32	0.8	$-0.188\ [-0.533, 0.087]$	$-0.136\ [-0.294, -0.050]$
32	1.6	$-0.342\ [-0.950, 0.140]$	$-0.208\ [-0.588, 0.028]$
32	2	$-0.395\ [-1.075, 0.140]$	$-0.226\ [-0.675, 0.065]$
32	3	$-0.424\ [-1.189, 0.194]$	$-0.241\ [-0.757, 0.160]$
Self-conditioned refinement			
16	0	$0.000\ [0.000, 0.000]$	$0.000\ [0.000, 0.000]$
16	0.8	$0.360\ [0.073, 0.653]$	$0.142\ [0.060, 0.226]$
16	1.6	$0.740\ [0.211, 1.330]$	$0.271\ [0.180, 0.369]$
16	2	$0.920\ [0.289, 1.637]$	$0.334\ [0.226, 0.446]$
16	3	$1.237\ [0.436, 2.179]$	$0.469\ [0.324, 0.636]$
24	0	$0.000\ [0.000, 0.000]$	$0.000\ [0.000, 0.000]$
24	0.8	$0.075\ [0.006, 0.150]$	$0.117\ [0.052, 0.192]$
24	1.6	$0.119\ [0.005, 0.237]$	$0.247\ [0.115, 0.392]$
24	2	$0.149\ [0.019, 0.285]$	$0.298\ [0.132, 0.455]$
24	3	$0.223\ [0.051, 0.406]$	$0.422\ [0.211, 0.600]$
32	0	$0.000\ [0.000, 0.000]$	$0.000\ [0.000, 0.000]$
32	0.8	$0.007\ [-0.267, 0.295]$	$0.142\ [0.001, 0.348]$
32	1.6	$0.036\ [-0.440, 0.508]$	$0.307\ [-0.031, 0.701]$
32	2	$0.056\ [-0.509, 0.602]$	$0.388\ [-0.023, 0.822]$
32	3	$0.152\ [-0.515, 0.792]$	$0.533\ [0.030, 1.057]$

E NFT IMPLEMENTATION AND GOLD-ANCHOR ANALYSIS

E.1 COLLECTION, REWARDS, AND FORWARD CORRUPTION

Each collection round draws 24 prompt groups from the applicable training population; coding uses the screened reward cohort in Appendix A.4. The old policy produces 15 candidates per group using 32 Euler denoising steps, synchronous identity clocks, the logit-normal-quantile grid of Appendix D, initial noise scale two, prompt CFG one, and SCCFG two. Self-conditioning is recurrent during collection. Terminal token decoding is greedy and uses the same SCCFG scale. We save the terminal latent endpoint and its decoded answer length; generated text is used for reward evaluation without

re-encoding it through Qwen. Each group additionally receives one gold endpoint from the fixed answer cache. No Qwen Transformer forward is required during NFT.

The generated reward is 0.75 for a correct final answer and zero otherwise, while the gold reward is one. GSM8K uses numerical-answer correctness. MATH uses Math-Verify 0.9.0 with a five-second timeout; unparsed predictions, exceptions, and timeouts receive zero. Coding correctness requires all retained native training assertions to pass (Appendix A.4). All-correct generated groups are excluded before computing normalization statistics. For retained group b and candidate m, define

$$A_{bm} = \mathrm{clip}\left(\frac{R_{bm} - \bar{R}_b}{\sigma_R + 10^{-4}}, -5, 5\right), \qquad \rho_{bm} = \mathrm{clip}\left(\frac{1}{2} + \frac{A_{bm}}{10}, 0, 1\right), \tag{24}$$

where $\bar{R}_b$ averages all 16 endpoints and σ_R is the population standard deviation of the retained *raw* rewards across the complete collection round. Discarded groups do not enter these statistics. All-incorrect groups remain informative because they include a gold endpoint. The GSM8K implementations additionally stop before an update if an entire round has zero correct generations; the MATH and coding implementations disable that round-level guard and permit gold-anchored updates. A round with no retained groups supplies no optimizer update.

Correct generations remain in mixed groups. Because gold has a higher raw reward than correct generations, even all-correct groups would supply a gold-versus-generation preference if retained. Filtering removes those groups to concentrate updates on observed failures. The higher gold reward favors the reference reasoning trace within the latent objective, whereas generated-solution rewards depend only on final-answer correctness.

For each retained endpoint, draw eight independent time/noise/self-conditioning records, yielding at most $24 \times 16 \times 8 = 3072$ records per update. Each record draws $t = \mathrm{sigmoid}(-1.5 + 0.8\xi)$, with $\xi \sim \mathcal{N}(0, 1)$, independently of the rollout grid. All representation groups use this same time, with unit group weights. Corrupt answer latents using Eq. (1) and convert predictions and endpoint targets using $\Delta_t = \max(1 - t, 0.05)$. Each policy restores its own clean prompt states, while endpoint, noise, answer states, and masks are shared across roles. Masks include the first EOS token and exclude prompt and subsequent padding positions.

E.2 SCCFG CONSTRUCTION AND EXACT LOSS

For each record, sample $b_{\mathrm{sc}} \sim \mathrm{Bernoulli}(0.5)$ and $\log(1 + g) \sim \mathrm{Uniform}(\log 1.5, \log 6)$. Use the same b_{sc}, g for current, old, and reference roles. Each role first predicts a clean endpoint under zero answer self-conditioning, restores its own clean prompt, and detaches that prediction. The main forward receives b_{sc} times the bootstrap prediction in answer positions; prompt positions remain clean regardless of b_{sc}. The guidance-scale input g is supplied to both passes. Convert both predictions to velocities and apply Eq. (8).

Only the current main forward carries gradients. Bootstrap predictions, the entire guidance correction, old/reference computations, sampled endpoints, and data targets are detached, giving $\nabla_\Theta \widetilde{v}_{\mathrm{cur}} = \nabla_\Theta v^{\mathrm{main}}_{\mathrm{cur}}$ under the implemented differentiation. The current prompt encoder receives the main-forward chain-rule gradient, including its use as the clean condition. Its output gradients are accumulated across records and propagated through the shared prompt graph. This implements the same detached construction as the ELF guidance-adjusted target: $\widetilde{v}_{\mathrm{cur}} - u^{\mathrm{stab}}_t$ equals $v^{\mathrm{main}}_{\mathrm{cur}} - \mathrm{sg}[u^{\mathrm{stab}}_t + b_{\mathrm{sc}}(1 - g^{-1})(v^{\mathrm{main}}_{\mathrm{cur}} - v^{0}_{\mathrm{cur}})]$ when u^{stab}_t is detached. Removing the stop-gradient from the correction would change the update.

Let b index retained training records and i full-canvas token positions. With $d = 1024$, define $N_{\mathrm{tok}} = \sum_{bi} M_{bi}$ across all ranks and microbatches of the update. Define

$$\mathcal{S}(e; \omega) = \sum_{bi} M_{bi} \omega_b \frac{\|e_{bi}\|_2^2}{d}. \tag{25}$$

With $e_c = \widetilde{v}_{\mathrm{cur}} - \mathrm{sg}(u^{\mathrm{stab}}_t)$ and $e_o = \mathrm{sg}(\widetilde{v}_{\mathrm{old}}) - \mathrm{sg}(u^{\mathrm{stab}}_t)$, the beta-one residuals are $e_+ = e_c$ and $e_- = 2e_o - e_c$. The executed loss is

$$\mathcal{L}_{\mathrm{NFT}} = \frac{5}{N_{\mathrm{tok}}}\big[\mathcal{S}(e_+; \rho) + \mathcal{S}(e_-; 1 - \rho)\big] + \frac{0.1}{N_{\mathrm{tok}}}\mathcal{S}\big(\widetilde{v}_{\mathrm{cur}} - \mathrm{sg}(\widetilde{v}_{\mathrm{ref}}); 1\big). \tag{26}$$

This averages coordinates and valid tokens, rather than averaging a separate mean for each solution. Gradients are summed across ranks using the same update-wide denominator. The reference term is a squared-velocity penalty. We omit the adaptive residual normalization of the original Diffusion-NFT implementation (Zheng et al., 2025); no decoder CE, prompt MSE, or separate correct-solution flow auxiliary loss is added.

E.3 Policy roles and old-policy updates

Appendix A.4 specifies initialization, trainable components, and optimizer settings.

The parameterization in Section 4.2 makes the question tokens the external condition for all roles. Thus the current, old, and reference policies use their respective encoders P_ϕ, $P_{\phi_{\text{old}}}$, and $P_{\phi_{\text{ref}}}$ while being compared under the same question q.

After successful update n, both old-policy components are updated by

$$\Theta_{\text{old}} \leftarrow \alpha_n \Theta_{\text{old}} + (1-\alpha_n)\Theta_{\text{cur}}, \qquad \alpha_n = \begin{cases} \min(0.001n, 0.5), & n < 400, \\ 0.4, & n \geq 400. \end{cases} \tag{27}$$

The reference remains at initialization. The evaluation EMAs specified in Appendix A.4 are distinct from this old-policy retention schedule.

E.4 Proof and interpretation of gold anchoring

Proof of Lemma 2. All expectations below condition on $\mathcal{H}$ and a gold endpoint, so f, h, ρ_G are fixed. Let $\sigma_G^2 = \mathbb{E}[\|U - m_G\|_2^2 \mid \mathcal{H}, \text{gold}]$. Conditional variance decomposition gives

$$\mathbb{E}[\ell_1(f, h; U, \rho_G) \mid \mathcal{H}, \text{gold}] = \rho_G \|f - m_G\|_2^2 + (1-\rho_G)\|f - (2h - m_G)\|_2^2 + \sigma_G^2. \tag{28}$$

For any vectors a, b, the identity $\omega\|f-a\|^2 + (1-\omega)\|f-b\|^2 = \|f - [\omega a + (1-\omega)b]\|^2 + \omega(1-\omega)\|a-b\|^2$ completes the square. Substituting $a = m_G$, $b = 2h - m_G$, and $\omega = \rho_G$ yields Eq. (9), with

$$C_{\mathcal{H}} = \sigma_G^2 + 4\rho_G(1-\rho_G)\|m_G - h\|_2^2. \tag{29}$$

The anchor weights $2\rho_G - 1$ and $2(1-\rho_G)$ are nonnegative and sum to one. □

Random coefficients and reference interpretation. In the executed reward mapping, the gold raw reward exceeds its group's mean, so $\rho_G \in (1/2, 1]$. Its value depends on the group's generations and the retained population. The conditioning information $\mathcal{H}$ therefore includes the realized coefficient and any retention information used to define the population; m_G is computed under that same conditional distribution. The lemma does not assume coefficient–target independence. If that independence holds given state, time, and question, the coefficient can be removed from the conditioning of m_G. Identifying m_G with a reference field requires the stronger idealization that the reference matches this gold posterior target under the same conditioning and corruption law. Correct final answers alone do not imply that equality. If the reference and old fields both equal m_G, the gold anchor equals the reference for every allowed ρ_G.

Relation to NFT and detached guidance. The moving old policy centers NFT's implicit positive/negative construction, while our fixed reference appears in the additional penalty of Eq. (26). Lemma 2 characterizes a third source of anchoring supplied by gold data: attraction toward the gold posterior target, interpolated with the old field. It holds for the field-level quadratic with auxiliary inputs fixed. In the SCCFG implementation, differentiation treats the guidance correction as fixed within each update, even though it is recomputed from the current network on subsequent forwards. The lemma consequently interprets the local corrected-field regression; it supplies no monotonic-improvement guarantee for the full recurrent sampler. Group filtering changes the training population, and the information in self-conditioning differs between collection and forward-noising updates. Our benchmark comparisons assess the resulting complete procedure empirically.

F EVALUATION PROTOCOLS AND UNCERTAINTY

F.1 BENCHMARKS, CHECKPOINTS, AND SCORING

Benchmarks and scoring. Each generation seed evaluates all 1,319 GSM8K test problems or all 500 MATH500 problems in fixed benchmark order. GSM8K accuracy compares extracted final answers with gold answers by numerical equality. Extraction prioritizes the last boxed number, then explicit answer markers, then the last number; normalization removes formatting and parses decimals and fractions. Math-Verify is the primary MATH scorer throughout, including the schedule comparison in Table 1; PRM800K scores are retained as a secondary diagnostic. Validation comparisons use the separate fixed 1,000-problem GSM8K and 500-problem MATH panels described in Appendix D. Checkpoint, EMA, clock, guidance, and seed choices are specified with each comparison.

Math checkpoints and generation settings. The headline and inference-allocation math evaluations use a 1,024-token canvas and async clocks $(2.5, 2, 1.5)$. Table 6 lists the supervised and NFT endpoints. Supervised evaluations select backbone/prompt EMA $.9999/.9999$; NFT evaluations select $.99/.99$. Table 2 uses eight sampling seeds at 64 denoising steps. Table 3 uses 64, 32, 16, and 8 seeds at 8, 16, 32, and 64 steps, respectively. Our rows in both tables use SCCFG 2 for GSM8K and SCCFG 3 for MATH500. The inference-allocation figures retain SCCFG 2, whose 64-step MATH500 pass@1 is 18.85% before NFT and 23.60% afterward.

F.2 EXTERNAL REPORTS AND PARAMETER ACCOUNTING

Tables 2–3 and 13 use the comparison papers' reported accuracies and sampling budgets, without re-evaluating their models. GSM8K uses 4-shot prompts for Qwen/LLaDA, 8-shot prompts for Dream/TESS 2, and zero-shot prompts for ours. FMLM+ grades Python solutions, whereas ELF-REG and LFRM produce natural-language reasoning. The low-NFE values come from FMLM+ Table 8 and ELF-REG Tables 13 and 17; ELF-REG uses early stopping (Agarwal et al., 2026; Li et al., 2026). The $\mathbb{S}$-FLM row uses its enhanced hyperspherical configuration with top-1 velocity decoding, rather than its basic sampler. MLFM's 31.24% result uses context-corrupted guidance and online token promotion: its reported 256 sampling steps need not equal 256 network calls, so Table 13 marks NFE as unestablished. SDAR uses four steps per four-token block. ELF-REG's reported budget includes one terminal decoder call, so our tables mark it with *; our NFE excludes prompt encoding and terminal decoding.

Parameter counts include frozen components. Our inherited Qwen vocabulary input/output maps contain 545M parameters and are called once per response; the denoising backbones contain 90.4M/638M parameters. The $(+X)$ counts include the learned prompt encoder and terminal projection, also used once per response. ELF-REG includes its frozen Qwen prompt encoder, and its reported non-decoder counts retain training-only REPA projectors. MLFM includes unmerged adapters; SDAR counts both untied vocabulary maps. These conventions describe stored parameters and invocation frequency, not equal inference cost across architectures.

F.3 ORACLE PASS@K AND MAJORITY VOTING

With N benchmark problems, n saved draws per problem, and C_b correct draws for problem b, the oracle estimator is

$$\widehat{\text{pass@}k} = \frac{1}{N}\sum_{b=1}^{N}\left[1 - \frac{\binom{n-C_b}{k}}{\binom{n}{k}}\right], \qquad \binom{n-C_b}{k} = 0 \quad \text{if } n - C_b < k. \tag{30}$$

Each depth $S \in \{8, 16, 32, 64\}$ has $n = 512/S$ saved draws per problem, giving $64, 32, 16, 8$ draws, respectively. Thus every available draw contributes to each supported k, rather than selecting one arbitrary group of k seeds. At $k = n$, this becomes the fraction of problems with any correct saved answer.

Voting and answer equivalence. We use "majority vote" for the plurality rule, which does not require more than half the samples to agree. We average voting accuracy over uniformly selected

k-subsets of the same draws. Numerical normalization groups GSM8K answers; the pinned mutual Math-Verify comparison groups MATH500 answers. Invalid extractions abstain, and an all-invalid subset fails. Ties among the largest answer groups are resolved uniformly, with a uniformly selected representative within the chosen group; the reported accuracy averages over these choices. Gold correctness labels score the selected representative but do not determine the selection. Equivalence partitions permit exact hypergeometric subset averaging. For nontransitive MATH comparisons, we recluster within each subset, enumerating up to 65,536 subsets and otherwise using a fixed 16,384-subset approximation. At $k = 1$, voting and oracle accuracy coincide; with two valid answers, uniform tie-breaking also prevents a systematic voting advantage over a single draw.

F.4 Uncertainty and reporting conventions

Sampling-seed variation. For seed accuracies $A_1, \ldots, A_n$, expressed as percentages, we report the mean $\bar{A}$ and sample standard deviation

$$\mathrm{SD}_{\mathrm{seed}} = \sqrt{\frac{1}{n-1}\sum_{m=1}^{n}(A_m - \bar{A})^2}.$$

These values describe generation randomness at a fixed checkpoint and test set. They are distinct from uncertainty across test problems and from variation across training runs.

Bars in the representation learning curves are ± 1 seed SD; small parenthesized values in the corresponding tables report the same statistic. External baselines retain author-reported accuracies; we do not impute missing SDs.

Problem-bootstrap uncertainty. Parentheses in Tables 15–17 and 20–21 report the standard deviation of the benchmark mean across 4,000 problem-bootstrap replicates, using bootstrap seed 20260918. Each replicate resamples whole problems with replacement while preserving their complete draw sets. These are bootstrap standard errors conditional on the checkpoints and saved generations, not the across-generation-seed SDs in Table 14. In particular, the largest-k point uses the entire draw pool and does not provide repeated independent k-sample groups from which to estimate a seed-group SD. The bootstrap does not add training-run variability or uncertainty from the fixed voting-subset approximation. All values are rounded independently from full precision.

Confidence intervals and paired comparisons. A 95% CI is a pointwise percentile interval from 4,000 problem-bootstrap replicates. This convention applies to the prompt-curriculum and reconstruction-probe figures. For pre/post comparisons, we resample the same problem indices in both stages and form the difference within each replicate. Pointwise 95% percentile intervals for changes therefore use paired differences. None of these measures estimates variation across independent training runs.

G ADDITIONAL MATHEMATICAL-REASONING RESULTS

This section collects the expanded mathematical-reasoning comparison, sampling-seed variation, and inference-allocation results. Appendix F defines the protocols and uncertainty measures.

G.1 FULL BASELINE COMPARISON

Table 13: Full mathematical-reasoning comparison, mean pass@1 (%). Parameters are in millions (three significant digits). Embedding Params count vocabulary input/output maps, tied weights once; Other Params count remaining weights, with $(+X)$ used once per response. Frozen components are included. ${}^{*}N$ denotes $N-1$ denoising calls for ELF-REG. Bold/underline mark first/second within each sub-billion continuous-backbone group.

Model	Embedding Params (M)	Other Params (M)	NFE	GSM8K	MATH500
Autoregressive					
Qwen3-0.6B-Base (Yang et al., 2025)	156	440	—	59.59	—
Qwen3-0.6B, thinking (Yang et al., 2025)	156	440	—	—	77.60
Discrete diffusion / blockwise diffusion					
LLaDA-8B-Base (Nie et al., 2025)	1,040	6,980	1,024	70.30	—
Dream-v0-Base-7B (Ye et al., 2025)	1,090	6,530	256	77.20	—
SDAR-1.7B-Chat (Cheng et al., 2025)	622	1,410	4/block	80.10	63.20
Continuous diffusion / flow: denoising backbone below 200M					
$\mathbb{S}$-FLM (top-1) (Deschenaux & Gulcehre, 2026)	75.5	87.5	1,024	18.00†	—
FMLM+ Init (Agarwal et al., 2026)	75.5	92	64	33.60†	—
ELF-REG-B (Li et al., 2026)	311	104 $(+441)$	${}^{*}64$	38.11	—
LFRM-B, pre-NFT (ours)	545	90.4 $(+45.8)$	64	<u>39.17</u>	—
LFRM-B, post-NFT (ours)	545	90.4 $(+45.8)$	64	**42.16**	—
Continuous diffusion / flow: denoising backbone 200M–1B					
ELF-REG-L (Li et al., 2026)	311	652 $(+442)$	${}^{*}64$ / ${}^{*}128$	55.96	13.39
LFRM-L, pre-NFT (ours)	545	638 $(+123)$	64	<u>58.67</u>	<u>21.18</u>
LFRM-L, post-NFT (ours)	545	638 $(+123)$	64	**63.74**	**24.68**
Continuous diffusion / flow: larger backbones (reference)					
MLFM (Azangulov et al., 2026)	123	1,220	NR§	31.24	—
TESS 2, GSM8K FT (Tae et al., 2025)	262	6,740	1,000	$\approx 68.9^{\ddagger}$	—

External results are author-reported; †Python scoring, ‡Approximate accuracy read from TESS 2 Figure 3b. §MLFM reports 256 sampling steps with guidance and online token promotion; the corresponding NFE is not established (NR). Slashes separate GSM8K/MATH500 budgets. Appendix F details prompting and parameter accounting; Table 14 in Appendix G.2 gives our seed SDs.

G.2 VARIATION ACROSS EVALUATION SEEDS

Table 14 reports standard deviations across independent sampling seeds for all our settings in Tables 2 and 3, together with the SCCFG 2 MATH500 settings used in Figure 4. Each seed generates one solution per problem across the full fixed test set. We use every completed sampling seed at each denoising budget: $n = 64, 32, 16, 8$ for NFE $8, 16, 32, 64$, respectively. Checkpoints, guidance, scorers, and benchmark populations match the main tables.

For the other main-body comparisons, Table 11 in Appendix D repeats Table 1's two-seed schedule results with SDs; Tables 20–21 in Appendix H give pass@1 seed SDs in their final columns for the eight-seed coding comparisons in Tables 4 and 18, respectively. Their pass@k columns additionally report problem-bootstrap SEs. Table 10 in Appendix C.4 also includes the four-seed epoch-six summary discussed in Sections 4.1 and 4.2. These seed statistics cover our evaluations; external baselines retain the authors' reported accuracies, without imputing missing SDs.

Table 14: Mean accuracy $\pm$ standard deviation across sampling seeds, in percentage points. Each seed evaluates the complete benchmark at the same fixed checkpoint. Columns use 64, 32, 16, and 8 seeds, respectively. The 64-NFE column supplies the results in Table 2; SCCFG 3 supplies the MATH500 rows in Tables 2 and 3. SCCFG 2 MATH500 rows correspond to Figure 4.

Model / stage	8 NFE	16 NFE	32 NFE	64 NFE
Sampling seeds n	64	32	16	8
GSM8K, LFRM-B, SCCFG 2				
Pre-NFT	26.40 ± 0.87	34.48 ± 0.94	37.80 ± 1.20	39.17 ± 0.63
Post-NFT	30.08 ± 0.87	37.77 ± 0.84	41.13 ± 0.94	42.16 ± 0.88
GSM8K, LFRM-L, SCCFG 2				
Pre-NFT	49.41 ± 0.86	55.33 ± 0.91	57.88 ± 1.03	58.67 ± 0.69
Post-NFT	54.84 ± 0.92	60.55 ± 0.69	62.47 ± 0.82	63.74 ± 0.83
MATH500, LFRM-L, SCCFG 3				
Pre-NFT	15.23 ± 1.16	17.93 ± 1.30	19.65 ± 1.37	21.18 ± 0.70
Post-NFT	17.45 ± 1.33	20.93 ± 1.26	22.60 ± 1.58	24.68 ± 1.27
MATH500, LFRM-L, SCCFG 2				
Pre-NFT	13.88 ± 1.19	16.39 ± 1.20	17.89 ± 1.31	18.85 ± 1.38
Post-NFT	16.31 ± 1.36	20.09 ± 1.23	22.08 ± 1.41	23.60 ± 1.79

G.3 LFRM-B INFERENCE ALLOCATION BEFORE AND AFTER NFT

Figure 8 complements the LFRM-L comparison in Figure 4. It uses the GSM8K supervised epoch-18 and NFT-update-300 checkpoints with async $(2.5, 2, 1.5)$ and SCCFG 2. Y-axis ranges are matched across training stages separately for oracle accuracy and voting.

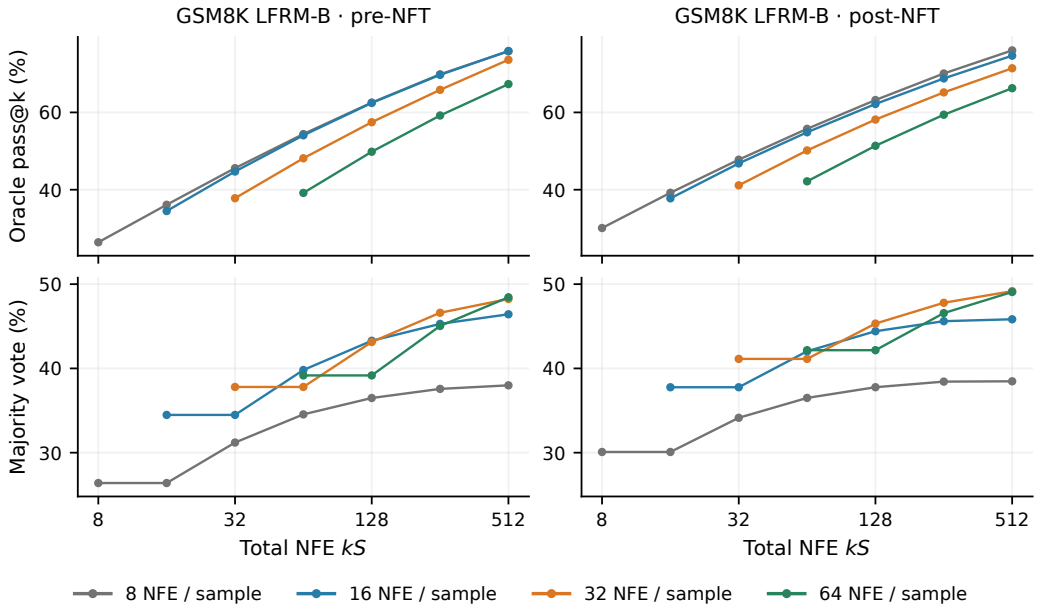

Figure 8: GSM8K LFRM-B before and after NFT. Upper panels show oracle pass@k; lower panels show majority-vote accuracy. Colors denote denoising NFE per sample. The same 64/32/16/8 sample cohorts and total budget of 512 denoising calls are used as in Figure 4. Table 17 gives every plotted value with problem-bootstrap SDs.

G.4 NUMERICAL INFERENCE-ALLOCATION RESULTS

Tables 15–17 give every oracle and voting point in Figures 4 and 8, at SCCFG 2 throughout. Each table pairs supervised and NFT checkpoints at identical denoising depths, sample counts, and ordered generation seeds. The underlying populations contain all 1,319 GSM8K or 500 MATH500 test problems.

Metrics and uncertainty are defined in Appendix F.

Paired NFT improvements. For the 64-step, SCCFG-2 LFRM-L comparisons discussed in Section 6.1, NFT improves pass@1 by 5.07 points on GSM8K ($[4.26, 5.90]$) and 4.75 on MATH500 ($[3.33, 6.15]$). At eight steps and 64 samples, the NFT-minus-supervised oracle differences are -1.06 points on GSM8K LFRM-L ($[-2.35, 0.23]$) and -0.60 on MATH500 ($[-3.80, 2.60]$). Both intervals include zero. At 64 steps and eight samples, voting improves by 1.75 points on GSM8K LFRM-L ($[0.22, 3.29]$) and 6.07 on MATH500 ($[3.04, 8.98]$). These intervals are conditional on the evaluated checkpoints and draws and are not adjusted for multiple comparisons.

Interpreting NFT gains across allocations. At eight steps on GSM8K LFRM-L, NFT raises pass@1 from 49.41% to 54.84%, while pass@64 changes from 89.84% to 88.78%. On MATH500 at the same allocation, voting rises from 24.09% to 25.79%; this smaller gain has paired 95% interval $[-0.45, 3.85]$. Thus voting improvements also depend on the allocation. For independent samples with problem-specific success probability p_i, oracle success is $1-(1-p_i)^k$: at large k, gains saturate on problems already solved frequently. Voting instead depends on the correct answer's frequency relative to competing answer groups. Changes in correctness probability, its distribution across problems, and competing-answer frequencies can therefore affect these metrics differently. The

curves motivate this interpretation but do not by themselves establish a loss of reasoning diversity after NFT.

Table 15: GSM8K, LFRM-L: every point in Figure 4, with SCCFG 2. Entries are accuracy percentages with problem-bootstrap SDs in parentheses, not across-seed SDs (Appendix F.4). Each denoising depth uses $n = 512/S$ saved draws per problem; k-subset averaging uses all n.

			Oracle pass@k		Majority vote	
NFE S	k	kS	Pre-NFT	Post-NFT	Pre-NFT	Post-NFT
8	1	8	49.41 (1.04)	54.84 (1.09)	49.41 (1.04)	54.84 (1.09)
8	2	16	60.21 (1.08)	63.89 (1.11)	49.41 (1.04)	54.84 (1.09)
8	4	32	68.84 (1.05)	70.85 (1.08)	56.11 (1.15)	60.04 (1.17)
8	8	64	76.01 (0.99)	76.53 (1.02)	59.92 (1.18)	62.69 (1.20)
8	16	128	81.85 (0.92)	81.34 (0.95)	62.05 (1.21)	64.07 (1.24)
8	32	256	86.41 (0.85)	85.39 (0.90)	63.31 (1.25)	64.76 (1.27)
8	64	512	89.84 (0.83)	88.78 (0.88)	64.21 (1.31)	64.84 (1.31)
16	1	16	55.33 (1.06)	60.55 (1.11)	55.33 (1.06)	60.55 (1.11)
16	2	32	65.34 (1.08)	68.39 (1.10)	55.33 (1.06)	60.55 (1.11)
16	4	64	73.08 (1.04)	74.39 (1.05)	61.38 (1.15)	64.67 (1.17)
16	8	128	79.17 (0.98)	79.34 (1.00)	64.51 (1.18)	66.55 (1.20)
16	16	256	84.02 (0.92)	83.42 (0.95)	66.09 (1.22)	67.42 (1.23)
16	32	512	87.95 (0.89)	86.58 (0.94)	66.98 (1.29)	67.92 (1.27)
32	1	32	57.88 (1.08)	62.47 (1.10)	57.88 (1.08)	62.47 (1.10)
32	2	64	67.45 (1.08)	70.38 (1.08)	57.88 (1.08)	62.47 (1.10)
32	4	128	74.83 (1.03)	76.49 (1.03)	63.54 (1.16)	66.45 (1.16)
32	8	256	80.73 (0.99)	81.65 (0.97)	66.26 (1.19)	68.31 (1.20)
32	16	512	85.37 (0.97)	85.90 (0.96)	67.17 (1.28)	69.47 (1.26)
64	1	64	58.67 (1.10)	63.74 (1.12)	58.67 (1.10)	63.74 (1.12)
64	2	128	68.17 (1.11)	71.48 (1.09)	58.67 (1.10)	63.74 (1.12)
64	4	256	75.13 (1.10)	77.34 (1.06)	64.43 (1.18)	67.59 (1.17)
64	8	512	80.21 (1.11)	81.96 (1.06)	67.09 (1.26)	68.84 (1.24)

Table 16: MATH500, LFRM-L: every point in Figure 4, with SCCFG 2. Entries are accuracy percentages with problem-bootstrap SDs in parentheses, not across-seed SDs (Appendix F.4). Each denoising depth uses $n = 512/S$ saved draws per problem; k-subset averaging uses all n.

			Oracle pass@k		Majority vote	
NFE S	k	kS	Pre-NFT	Post-NFT	Pre-NFT	Post-NFT
8	1	8	13.88 (0.99)	16.31 (1.11)	13.88 (0.99)	16.31 (1.11)
8	2	16	21.10 (1.29)	23.98 (1.40)	13.88 (0.99)	16.31 (1.11)
8	4	32	29.74 (1.55)	32.65 (1.63)	16.87 (1.24)	19.64 (1.37)
8	8	64	39.29 (1.73)	41.82 (1.79)	20.00 (1.44)	22.68 (1.56)
8	16	128	49.27 (1.86)	50.94 (1.90)	22.23 (1.58)	24.59 (1.69)
8	32	256	58.92 (1.96)	59.37 (1.98)	23.50 (1.69)	25.59 (1.80)
8	64	512	67.60 (2.10)	67.00 (2.12)	24.09 (1.87)	25.79 (1.93)
16	1	16	16.39 (1.11)	20.09 (1.26)	16.39 (1.11)	20.09 (1.26)
16	2	32	24.35 (1.42)	28.68 (1.55)	16.40 (1.11)	20.10 (1.26)
16	4	64	33.40 (1.68)	37.94 (1.75)	19.90 (1.37)	24.03 (1.52)
16	8	128	42.73 (1.87)	47.25 (1.89)	23.41 (1.57)	27.66 (1.70)
16	16	256	51.86 (2.01)	56.27 (1.99)	26.21 (1.74)	30.15 (1.85)
16	32	512	60.20 (2.20)	64.80 (2.15)	29.05 (2.00)	31.39 (2.03)
32	1	32	17.89 (1.18)	22.08 (1.33)	17.89 (1.18)	22.08 (1.33)
32	2	64	26.31 (1.48)	31.24 (1.61)	17.89 (1.18)	22.08 (1.33)
32	4	128	36.02 (1.74)	40.81 (1.81)	21.56 (1.45)	26.48 (1.61)
32	8	256	46.17 (1.96)	50.42 (1.93)	25.54 (1.69)	30.26 (1.83)
32	16	512	56.20 (2.21)	60.40 (2.15)	28.68 (1.97)	32.42 (2.04)
64	1	64	18.85 (1.26)	23.60 (1.42)	18.85 (1.26)	23.60 (1.42)
64	2	128	27.50 (1.60)	33.00 (1.71)	18.85 (1.26)	23.60 (1.42)
64	4	256	37.13 (1.89)	43.07 (1.96)	22.70 (1.57)	28.03 (1.70)
64	8	512	47.20 (2.23)	52.60 (2.24)	26.34 (1.91)	32.42 (2.03)

Table 17: GSM8K, LFRM-B: every point in Figure 8, with SCCFG 2. Entries are accuracy percentages with problem-bootstrap SDs in parentheses, not across-seed SDs (Appendix F.4). Each denoising depth uses $n = 512/S$ saved draws per problem; k-subset averaging uses all n.

			Oracle pass@k		Majority vote	
NFE S	k	kS	Pre-NFT	Post-NFT	Pre-NFT	Post-NFT
8	1	8	26.40 (0.86)	30.08 (0.95)	26.40 (0.86)	30.08 (0.95)
8	2	16	36.10 (1.02)	39.21 (1.08)	26.40 (0.86)	30.08 (0.95)
8	4	32	45.58 (1.11)	47.77 (1.14)	31.20 (1.02)	34.14 (1.09)
8	8	64	54.40 (1.15)	55.75 (1.16)	34.55 (1.11)	36.50 (1.16)
8	16	128	62.54 (1.15)	63.20 (1.15)	36.49 (1.18)	37.77 (1.22)
8	32	256	69.86 (1.15)	70.07 (1.14)	37.57 (1.24)	38.43 (1.27)
8	64	512	75.82 (1.17)	76.04 (1.16)	37.99 (1.32)	38.47 (1.33)
16	1	16	34.48 (0.98)	37.77 (1.05)	34.48 (0.98)	37.77 (1.05)
16	2	32	44.71 (1.09)	46.80 (1.14)	34.48 (0.98)	37.77 (1.05)
16	4	64	54.08 (1.13)	54.86 (1.17)	39.81 (1.12)	42.03 (1.17)
16	8	128	62.46 (1.14)	62.17 (1.17)	43.28 (1.19)	44.41 (1.23)
16	16	256	69.71 (1.13)	68.79 (1.16)	45.29 (1.26)	45.61 (1.29)
16	32	512	75.89 (1.16)	74.68 (1.19)	46.42 (1.36)	45.83 (1.36)
32	1	32	37.80 (1.01)	41.13 (1.09)	37.80 (1.01)	41.13 (1.09)
32	2	64	48.13 (1.11)	50.16 (1.16)	37.80 (1.01)	41.13 (1.09)
32	4	128	57.46 (1.14)	58.15 (1.19)	43.14 (1.14)	45.32 (1.19)
32	8	256	65.84 (1.15)	65.17 (1.20)	46.60 (1.23)	47.79 (1.26)
32	16	512	73.62 (1.20)	71.42 (1.24)	48.24 (1.33)	49.17 (1.35)
64	1	64	39.17 (1.04)	42.16 (1.11)	39.17 (1.04)	42.16 (1.11)
64	2	128	49.82 (1.15)	51.36 (1.19)	39.17 (1.04)	42.16 (1.11)
64	4	256	59.20 (1.21)	59.40 (1.24)	45.04 (1.19)	46.56 (1.22)
64	8	512	67.32 (1.29)	66.26 (1.32)	48.44 (1.32)	49.07 (1.34)

H CODE GENERATION: PROTOCOLS AND DETAILED RESULTS

Training stages and checkpoint selectors. Appendix A describes the coding architecture, three supervised stages, and NFT continuation; Appendix A.3 gives the supervised joint prompt-MSE regularizer. Supervised evaluations select backbone EMA 0.9999, with prompt EMA 0.999 for standalone MSE and 0.9999 for joint epoch 13. Post-NFT uses update 100 with independently selected backbone/prompt EMA $0.9/0.9$. Frozen-Qwen and standalone-MSE conditions share the epoch-12 backbone; joint training and NFT each update both trainable models. The coding checkpoint-selection procedure is given in Appendix A.4.

Generation and scoring. We use the benchmark task text in the native Qwen chat format, without assistant prefill, with batch size 32, CFG 1, SCCFG 3, and async $(2.5, 2, 1.5)$ throughout. Answers fill the remaining positions of the 1,024-token canvas and are decoded greedily. Denoising NFE excludes one terminal decoder call. The eight-seed cohort is $\{42, 123, 456, 789, 2026, 31415, 27182, 16180\}$. The shorter training curve uses 32 steps and its stated two- or four-seed cohorts.

HumanEval (Chen et al., 2021) contains 164 tasks; MBPP (Austin et al., 2021) uses the 378 tasks retained by EvalPlus (Liu et al., 2023). We report base and full extended tests on the same generated programs, requiring both suites to pass for a plus score. Standard scoring applies native EvalPlus sanitization without function-name repair. Execution uses a minimum timeout of 4 seconds, reference-runtime multiplier four, and 4 GiB per program; failures and timeouts count as incorrect. Our separate alias condition changes extraction only: if the expected binding is absent and exactly one top-level function is present, append `expected_name = generated_name` and sanitize again. Existing bindings and ambiguous cases are unchanged. This changes which programs the evaluator accepts without changing generation.

Mapping external reports. Our PlaidQ comparison uses the author-reported results from arXiv version 1, subsequently withdrawn by the authors to address institutional resource-disclosure requirements (Peng et al., 2026b). PlaidQ Appendix B.7 distinguishes original MBPP-500 from the 378-task subset evaluated using base assertions; its latter column is named "MBPP+" (Peng et al., 2026b). We map only that latter column to our MBPP-378 base metric. PlaidQ uses prefix completion with a 128-token response budget, benchmark-specific extraction, and a 15-second program timeout. Its Table 1 supplies our comparison values. ELF-REG Table 1 supplies its aliased HumanEval, HumanEval+, and MBPP-378 base results (Li et al., 2026); its Appendix B.3 documents the alias rule and larger execution limits (20-second minimum, reference multiplier ten, 6 GiB). Our alias rows match the entry-point repair and test-set definitions, while retaining our prompts and execution limits. These comparisons align scoring categories, rather than reproducing each paper's full generation protocol. Full extended MBPP+ values are not reported for either comparison paper. We retain their reported values without re-evaluating their models.

For additional discrete-model context, oDLM-0.6B reports HumanEval/HumanEval+ pass@1 of 17.87%/16.40% at 128 sampling steps with a 128-token response budget (Peng et al., 2026a, Table 1 and Appendix B.7). Edit Flow-1.3B reports 12.8%/10.4%, and its localized variant reports 14.0%/10.4% (Havasi et al., 2025, Table 3). Our learned-conditioning results without name repair are 27.97%/25.30% before joint training, 29.57%/26.91% at epoch 13, and 32.85%/30.18% after NFT. These comparisons use the authors' reported accuracies; training data, initialization, and sampling protocols differ. We restrict the additional comparison to HumanEval(+) rather than equating differently defined MBPP columns.

Comparison under function-name aliasing. Table 18 separates the ELF-REG-L comparison from standard scoring in Table 4. Under the same alias rule, all four LFRM-L variants exceed ELF-REG-L on all three reported metrics. For NFT update 100, aliasing raises MBPP-378 base accuracy from 22.26% to 38.96%, exceeding ELF-REG-L's 28.92%; without repair, it remains below guided PlaidQ's 24.58%. Mixing repaired and unrepaired scores would therefore obscure a substantial difference in acceptance criteria.

Prompt adaptation remains task dependent. At 32 steps, swapping in the encoder after one MSE epoch drops MBPP+ from 32.34% to 7.61%, much more than the math prompt

Table 18: **Code generation with single-function name aliasing.** Mean pass@1 (%) under the matched repair rule. ELF-REG-L values are author-reported; *128 denotes 127 denoising calls plus terminal decoding in its reported budget. Our rows use eight seeds: frozen/MSE conditioning at epoch 12; joint conditioning at epoch 13; and NFT update 100. Prompting and execution limits still differ across papers, as detailed above. Table 21 gives our seed SDs and higher-pass results.

Model / setting	Denoising NFE	HE	HE+	MBPP-378 base	MBPP+ full
ELF-REG-L (Li et al., 2026)	*128	22.56	21.46	28.92	—
LFRM-L / frozen Qwen (ours)	128	33.69	30.87	43.45	37.93
LFRM-L / prompt MSE (ours)	128	28.05	25.38	30.99	26.32
LFRM-L, pre-NFT (ours)	128	29.73	27.06	32.11	27.81
LFRM-L, post-NFT (ours)	128	33.16	30.26	38.96	33.76

swaps (Figure 3). Ten MSE epochs recover HumanEval+ to 24.70%, near Qwen conditioning's 25.61%, but MBPP+ reaches only 15.15%. Coding prompts average 233 tokens versus 84/89 for GSM8K/MATH; longer contexts may contribute to this harder compression problem, although length is not isolated. At 128 steps, one epoch of joint adaptation raises HumanEval(+) pass@1 by 1.60 points and HumanEval+ pass@8 from 42.07% to 47.56%, while standard MBPP-378 base/MBPP+ pass@1 decreases by 1.16/0.56 points. Joint adaptation therefore gives mixed gains at this early checkpoint.

Prompt lengths and encoder size. The recorded mean prefix lengths over unique training prompts are 84.17 tokens for the GSM8K pipeline (275,661 prompts), 89.15 for MATH (508,953), and 233.44 for OpenCodeInstruct (4,772,508). These are native formatted training prefixes, including the assistant header, rather than benchmark question lengths. Code prefixes average 2.77 and 2.62 times the GSM8K and MATH lengths; their median is 216 and 99th percentile 506. The length difference motivates the conditioning hypothesis but is not a controlled length ablation. Likewise, the cross-task prompt-swap comparison uses each task's selected MSE checkpoint (GSM8K LFRM-L epoch 30, MATH epoch 10, code epoch 1) and the stated seed cohorts. At 32 steps, the code MBPP+ swap loss is 24.74 percentage points, compared with 12.02 points on GSM8K LFRM-L and 2.90 on MATH500. The 121,333,180 parameters of our LFRM-L contextual encoder exclude the frozen $151{,}936 \times 2{,}560$ vocabulary lookup. ELF-REG's Qwen3-0.6B-Base contextual body has approximately 440M parameters, excluding its approximately 156M lookup; its additional Qwen3-1.7B-Base alignment teacher is used only during training. Our frozen-Qwen reference instead uses a larger 4B teacher, so the cross-paper comparison does not isolate encoder size.

Training trajectory and prompt substitution. Table 19 keeps NFE and guidance fixed across flow-training epochs 4/8/12 and the prompt-MSE substitution at the fixed epoch-12 backbone checkpoint. The prompt epoch counter starts afresh; prompt epoch 1 is already MSE-trained, not a random encoder. At 32 steps, the four-seed HumanEval+ gap between prompt epoch 10 and original Qwen is -0.91 points (paired 95% problem-bootstrap interval $[-5.03, 3.20]$); MBPP+ retains a -17.20 point gap ($[-20.24, -14.22]$). These comparisons isolate the conditioner because the backbone weights are identical.

Table 19: **Coding training and prompt-substitution results at 32 denoising steps.** Mean pass@1 (%) with generation-seed SD in parentheses; no aliases. Flow-training epochs 4/8 use seeds 42/123; epoch 12 and both prompt endpoints add 456/789. All rows use SCCFG 3. The prompt-MSE rows retain the same epoch-12 backbone weights.

Checkpoint / conditioner	Seeds	HE	HE+	MBPP-378 base	MBPP+
Epoch 4 / Qwen	2	23.17 (3.45)	21.04 (2.16)	26.85 (0.19)	23.68 (0.19)
Epoch 8 / Qwen	2	27.74 (6.47)	25.61 (5.17)	34.92 (2.24)	30.42 (1.87)
Epoch 12 / Qwen	4	27.90 (1.75)	25.61 (2.49)	37.76 (0.70)	32.34 (0.79)
Epoch 12 / prompt MSE 1	4	19.82 (2.08)	18.60 (2.31)	8.93 (1.02)	7.61 (1.23)
Epoch 12 / prompt MSE 10	4	26.37 (1.44)	24.70 (2.08)	17.59 (2.61)	15.15 (2.16)

Eight-seed variation and multiple attempts. Tables 20–21 give oracle pass@1/2/4/8 with problem-bootstrap standard errors in parentheses, and retain the pass@1 generation-seed SD in a separate column. For each problem with c successful samples among eight, pass@k uses $1 - \binom{8-c}{k}/\binom{8}{k}$, averaged across problems. Eight samples do not support pass@10, so the main cross-paper comparison uses pass@1. Bootstrap uncertainty uses 4,000 problem resamples with seed 20260918, retaining all eight generations per problem and pairing compared conditions. The bootstrap SE measures uncertainty across problems conditional on the saved generations; the seed SD describes sampling variation over the fixed benchmark. Neither estimates training variability (Appendix F.4).

Before joint training, at 128 steps, aliasing improves MSE-prompt MBPP-378 base by 11.38 points (95% interval $[9.19, 13.69]$), compared with 4.73 ($[3.41, 6.18]$) for frozen Qwen. With the MSE encoder, full MBPP+ improves by 9.36 points under repair, but remains 11.61 points below re-paired frozen-Qwen performance ($[9.13, 14.25]$). Thus entry-point naming accounts for part of the measured conditioning gap; correcting it does not recover the teacher-conditioned result. Standard HumanEval+ also retains a 5.56 point gap at this longer budget ($[1.75, 9.38]$), so the near-recovery at 32 steps should not be interpreted as equivalence at every inference budget.

Table 20: **Standard scoring without aliases.** LFRM-L with 128 denoising calls, SCCFG 3, eight seeds. Frozen Qwen and prompt MSE use epoch 12; pre-NFT is joint epoch 13; post-NFT is update 100. Pass@k entries are accuracies (%) with problem-bootstrap standard errors in parentheses; the final column gives the separate generation-seed SD of pass@1. Both uncertainty measures are in percentage points (Appendix F.4). Higher-k values average all seed subsets.

Stage	Benchmark	pass@1	pass@2	pass@4	pass@8	pass@1 seed SD
Frozen Qwen	HumanEval	33.69 (2.78)	44.66 (3.21)	54.13 (3.46)	62.20 (3.75)	2.11
Frozen Qwen	HumanEval+	30.87 (2.70)	41.53 (3.18)	50.82 (3.45)	59.15 (3.77)	2.09
Frozen Qwen	MBPP-378 base	38.72 (2.01)	49.17 (2.21)	58.26 (2.32)	66.40 (2.48)	1.37
Frozen Qwen	MBPP+	33.99 (2.00)	43.26 (2.25)	51.46 (2.40)	58.73 (2.60)	1.62
Prompt MSE	HumanEval	27.97 (2.84)	35.98 (3.27)	42.87 (3.58)	48.78 (3.90)	1.44
Prompt MSE	HumanEval+	25.30 (2.81)	32.19 (3.25)	37.65 (3.55)	42.07 (3.85)	1.84
Prompt MSE	MBPP-378 base	19.61 (1.54)	27.44 (1.90)	35.41 (2.17)	43.92 (2.53)	1.85
Prompt MSE	MBPP+	16.96 (1.47)	23.72 (1.84)	30.30 (2.14)	36.77 (2.48)	1.11
Pre-NFT	HumanEval	29.57 (2.74)	39.07 (3.23)	47.23 (3.55)	54.27 (3.87)	2.33
Pre-NFT	HumanEval+	26.91 (2.73)	35.26 (3.22)	42.22 (3.58)	47.56 (3.92)	1.99
Pre-NFT	MBPP-378 base	18.45 (1.53)	25.77 (1.90)	32.95 (2.20)	39.95 (2.53)	1.23
Pre-NFT	MBPP+	16.40 (1.45)	23.20 (1.84)	29.81 (2.15)	35.98 (2.48)	1.30
Post-NFT	HumanEval	32.85 (2.97)	41.59 (3.35)	49.08 (3.65)	54.27 (3.89)	1.91
Post-NFT	HumanEval+	30.18 (2.97)	37.83 (3.33)	44.48 (3.64)	49.39 (3.92)	1.79
Post-NFT	MBPP-378 base	22.26 (1.71)	29.62 (2.02)	36.76 (2.27)	43.39 (2.54)	1.50
Post-NFT	MBPP+	19.61 (1.63)	26.31 (1.96)	32.54 (2.24)	37.83 (2.49)	1.36

One epoch of joint adaptation. Table 22 compares joint epoch 13 with the epoch-12 backbone and prompt-MSE-epoch-10 encoder, matching all eight seeds, 128 denoising steps, guidance, prompts, and scoring. Both learned components change, and the prompt EMA selector changes from 0.999 to 0.9999; this compares the reported training stages rather than isolating the MSE regularizer's effect. Standard HumanEval(+) pass@1 rises by 1.60 points, with paired intervals including zero. HumanEval+ pass@8 rises by 5.49 points, while standard MBPP-378 base pass@8 falls by 3.97 points. Under alias scoring, MBPP+ pass@1 rises by 1.49 points; under standard scoring it falls by 0.56 points. These differing outcomes reinforce the need to separate entry-point repair from the full coding task.

NFT with execution-based rewards. Table 23 compares NFT update 100 against joint epoch 13 under matched prompts, ordered seeds, 128 denoising calls, SCCFG 3, arithmetic, and scorers. Backbone/prompt evaluation EMAs change from $0.9999/0.9999$ to $0.9/0.9$, so this comparison measures the reported training-stage change, including its EMA selection. Standard pass@1 improves on all four benchmarks, with positive paired intervals: HumanEval and HumanEval+ gain

Table 21: **Scoring with single-function aliases.** LFRM-L with 128 denoising calls, SCCFG 3, eight seeds. Frozen Qwen and prompt MSE use epoch 12; pre-NFT is joint epoch 13; post-NFT is update 100. Pass@k entries are accuracies (%) with problem-bootstrap standard errors in parentheses; the final column gives the separate generation-seed SD of pass@1. Both uncertainty measures are in percentage points (Appendix F.4). Higher-k values average all seed subsets.

Stage	Benchmark	pass@1	pass@2	pass@4	pass@8	pass@1 seed SD
Frozen Qwen	HumanEval	33.69 (2.78)	44.66 (3.21)	54.13 (3.46)	62.20 (3.75)	2.11
Frozen Qwen	HumanEval+	30.87 (2.70)	41.53 (3.18)	50.82 (3.45)	59.15 (3.77)	2.09
Frozen Qwen	MBPP-378 base	43.45 (2.06)	53.97 (2.22)	62.90 (2.28)	70.63 (2.39)	1.21
Frozen Qwen	MBPP+	37.93 (2.07)	47.21 (2.28)	55.07 (2.40)	61.90 (2.55)	1.43
Prompt MSE	HumanEval	28.05 (2.84)	36.11 (3.27)	43.05 (3.59)	48.78 (3.90)	1.38
Prompt MSE	HumanEval+	25.38 (2.81)	32.32 (3.25)	37.82 (3.56)	42.07 (3.85)	1.78
Prompt MSE	MBPP-378 base	30.99 (1.88)	40.54 (2.18)	48.72 (2.37)	55.82 (2.57)	1.48
Prompt MSE	MBPP+	26.32 (1.81)	34.45 (2.14)	41.41 (2.38)	47.09 (2.61)	1.54
Pre-NFT	HumanEval	29.73 (2.76)	39.16 (3.23)	47.24 (3.55)	54.27 (3.87)	2.34
Pre-NFT	HumanEval+	27.06 (2.74)	35.39 (3.23)	42.31 (3.59)	47.56 (3.92)	2.14
Pre-NFT	MBPP-378 base	32.11 (1.91)	41.66 (2.20)	49.82 (2.37)	57.14 (2.57)	1.72
Pre-NFT	MBPP+	27.81 (1.86)	36.19 (2.18)	43.16 (2.39)	49.21 (2.60)	1.88
Post-NFT	HumanEval	33.16 (2.97)	42.03 (3.34)	49.64 (3.64)	54.88 (3.88)	2.06
Post-NFT	HumanEval+	30.26 (2.97)	37.87 (3.34)	44.48 (3.64)	49.39 (3.92)	1.75
Post-NFT	MBPP-378 base	38.96 (2.10)	47.68 (2.32)	54.42 (2.42)	60.32 (2.55)	1.60
Post-NFT	MBPP+	33.76 (2.07)	41.21 (2.33)	46.55 (2.47)	50.79 (2.61)	1.43

Table 22: **Joint epoch 13 minus pre-joint prompt MSE.** Differences in percentage points at 128 denoising calls and SCCFG 3, using the same eight seeds. Brackets give paired 95% problem-bootstrap intervals from 4,000 replicates. Both models use backbone EMA 0.9999; prompt EMA is 0.999 before joint training and 0.9999 afterward. Differences are computed before rounding the stage accuracies.

Scoring	Benchmark	Δ pass@1 [95% CI]	Δ pass@8 [95% CI]
Standard	HumanEval	$+1.60$ $[-0.61, +3.73]$	$+5.49$ $[+0.00, +11.59]$
Standard	HumanEval+	$+1.60$ $[-0.53, +3.58]$	$+5.49$ $[+0.61, +10.98]$
Standard	MBPP-378 base	-1.16 $[-2.38, +0.03]$	-3.97 $[-7.67, -0.53]$
Standard	MBPP+	-0.56 $[-1.69, +0.53]$	-0.79 $[-4.23, +2.38]$
Alias	HumanEval	$+1.68$ $[-0.53, +3.81]$	$+5.49$ $[+0.00, +11.59]$
Alias	HumanEval+	$+1.68$ $[-0.46, +3.73]$	$+5.49$ $[+0.61, +10.98]$
Alias	MBPP-378 base	$+1.12$ $[-0.33, +2.65]$	$+1.32$ $[-2.38, +4.76]$
Alias	MBPP+	$+1.49$ $[+0.17, +2.88]$	$+2.12$ $[-1.06, +5.03]$

3.28 points, MBPP-378 base 3.80, and MBPP+ 3.21. Under alias scoring, the corresponding gains are 3.43, 3.20, 6.85, and 5.95 points. Aliasing remains a separate evaluation convention, rather than part of standard scoring.

Higher-pass gains are less pronounced. Standard HumanEval pass@8 remains 54.27%, and HumanEval+ rises from 47.56% to 49.39%; neither paired interval excludes zero. The MBPP pass@8 intervals also include zero, including the base-test interval whose lower endpoint rounds to zero. This mirrors the mathematics results: NFT improves single-sample reliability more clearly than oracle coverage, without establishing a decrease in diversity. HumanEval(+) pass@1 approaches the frozen-Qwen reference, but a substantial MBPP gap remains. These are conditional comparisons of fixed checkpoints and saved draws, not estimates of training-run or checkpoint-selection uncertainty.

Table 23: **NFT update 100 minus joint epoch 13.** Differences in percentage points with paired 95% problem-bootstrap intervals, over eight matched seeds at 128 denoising calls and SCCFG 3. Backbone/prompt EMAs are $0.9999/0.9999$ before NFT and $0.9/0.9$ afterward. Intervals use 4,000 problem resamples and condition on the selected endpoints.

Scoring	Benchmark	Δ pass@1 [95% CI]	Δ pass@8 [95% CI]
Standard	HumanEval	$+3.28$ $[+1.22, +5.34]$	$+0.00$ $[-5.49, +5.49]$
Standard	HumanEval+	$+3.28$ $[+1.30, +5.34]$	$+1.83$ $[-3.05, +6.71]$
Standard	MBPP-378 base	$+3.80$ $[+2.61, +5.09]$	$+3.44$ $[+0.00, +7.14]$
Standard	MBPP+	$+3.21$ $[+2.05, +4.43]$	$+1.85$ $[-1.32, +5.03]$
Alias	HumanEval	$+3.43$ $[+1.30, +5.56]$	$+0.61$ $[-4.27, +6.10]$
Alias	HumanEval+	$+3.20$ $[+1.14, +5.34]$	$+1.83$ $[-3.05, +6.71]$
Alias	MBPP-378 base	$+6.85$ $[+5.22, +8.40]$	$+3.17$ $[+0.00, +6.61]$
Alias	MBPP+	$+5.95$ $[+4.46, +7.47]$	$+1.59$ $[-1.32, +4.76]$

I GENERATION EXAMPLES AND TRAJECTORIES

I.1 GENERATED SOLUTIONS

Figure 9 presents one correct GSM8K generation and one correct MATH500 generation from the post-NFT LFRM-L models. The solutions combine intermediate calculations into a final answer: successive multiplicative changes in the GSM8K example, and substitution followed by rational arithmetic in the MATH500 example. Both are drawn from saved benchmark generations and retain the original model text.

GSM8K: reasoning across successive changes

Problem. A new program had 60 downloads in the first month. The number of downloads in the second month was three times as many as the downloads in the first month, but then reduced by 30% in the third month. How many downloads did the program have total over the three months?

Generated solution.

Let's break down the downloads for the program month by month.
In the first month, the program had 60 downloads.
In the second month, the program had three times as many as the downloads in the first month, so it had 60 * 3 = 180 downloads.
In the third month, the downloads were reduced by 30%, which means it had 180 - (180 * 30 / 100) = 180 - 54 = 126 downloads.
To find the total number of downloads over the three months, we add up the downloads from each month: 60 + 180 + 126 = 240 + 126 = 366
Thus, the program had a total of $\boxed{366}$ downloads over the three months.

MATH500: evaluating a rational function

Problem. If $f(x) = \frac{3x-2}{x-2}$, what is the value of $f(-2) + f(-1) + f(0)$? Express your answer as a common fraction.

Generated solution.

To find the value of $f(-2) + f(-1) + f(0)$ we need to substitute $x = -2, x = -1$, and $x = 0$ into the function $f(x)$.
For $f(-2)$: $f(-2) = \frac{3(-2)-2}{-2-2} = \frac{-6-2}{-4} = \frac{-8}{-4} = 2$
For $f(-1)$: $f(-1) = \frac{3(-1)-2}{-1-2} = \frac{-3-2}{-3} = \frac{-5}{-3} = \frac{5}{3}$
For $f(0)$: $f(0) = \frac{3(0)-2}{0-2} = \frac{-2}{-2} = 1$
Now we can add these together to find the sum: $f(-2) + f(-1) + f(0) = 2 + \frac{5}{3} + 1 = \frac{11}{3} + 1 = \frac{11}{3} + \frac{3}{3} = \frac{14}{3}$ $\boxed{\frac{14}{3}}$

Figure 9: Correct generations from LFRM-L at 64 denoising NFEs, async $(2.5, 2, 1.5)$, SCCFG 2, and seed 42. GSM8K uses NFT update 500; MATH500 uses NFT update 600. Displayed equations have been moved inline to save space; model wording and calculations are unchanged. Both answers and the intermediate calculations are correct.

I.2 INTERMEDIATE GENERATION TRAJECTORY

Figure 10 shows steps 20, 40, 49, 60, 62, and 64 of the GSM8K example in Figure 9. After sampling, we apply the terminal decoder (time 1, SCCFG 2, zero decoder self-conditioning) to each saved state. These premature readouts neither feed back into sampling nor represent intermediate clean-latent predictions. The final text exactly reproduces the archived generation.

Light blue marks exact token-ID equality at the same position in the final answer, before its EOS; matches need not persist. Panels label the local clocks (t_{16}, t_{24}, t_{32}): 0 is noise and 1 is clean. Each readout retains its native extraction and EOS; only whitespace layout changes.

Step 20 / 64 Layer 16: 0.006 Layer 24: 0.017 Layer 32: 0.047

```
8888 8 8            8   8   88 8    88  88  8 8 88888  8   888888 8888 88  8888 8
    8  88888 88  88888 888 888 88 8888 888888 88888888  8888 8888  8888 888  8888
8888 88888888 88888 888 8888888888888 8888888 8  8888          8  8         88
    8   8 8         8 88           888      8    8        888    8           8  8
               8

 8   8     8 8     8  8          8    88        8       88  8        88   88
      8   8               8  8           8  8       8
    8
            88   8
```

Step 40 / 64 Layer 16: 0.024 Layer 24: 0.050 Layer 32: 0.106

```
8878 8 8         88 888   8  88 8888 888 888 8888888 88 888 888888 8888 88
888888887 8888 78   827 2880   817  888888 88   8888zing the7    we... $60 + 180 +
108 = 271 + 188 = 3Q6$87 the78 the8  \boxed...778887 the 8.
```

Step 49 / 64 Layer 16: 0.043 Layer 24: 0.081 Layer 32: 0.152

```
8 the break the the7 8              8  the 88 the88 888888 8 88 8  88   8
888888888 8888888    8... $188 - the181 * 30 / 100... = 180 - 54 = 12648To the the
  8   888   the2  the8: 68 + 188 + 127 = 271 + 126 = 376$87 the78 the8  \boxed
876878 the 8.
```

Step 60 / 64 Layer 16: 0.123 Layer 24: 0.187 Layer 32: 0.284

```
8 the7 the the2    8 81 the 88 the88 88 81 the 88 the88 2  2 the2  the 88  8 888
37 188 81 the  8 the877 by 8487  7 188 - 181lin 78  184... = 188 - 54 = 126 8To
the the   27 the  87   the28 the...: 88 + 188 + 127 = 27 the + 127 = 37687 the
the88 the   \boxed 876888 the 8.
```

Step 62 / 64 Layer 16: 0.175 Layer 24: 0.248 Layer 32: 0.351

```
Let the break the the2  the8 8 81 the red8 the8  88281 the the88 the2  2  2 the2
the     8 888 8  188 82 the the88 the2788 888 2   188 - 188  38  181...  188 - 577
126 8... find the 8 2  the  8    the28 the8: 88 + 188 + 127  27 the + 127  77 87
the the88 the   \boxed 8768   the 8.
```

Step 64 / 64 (final) Layer 16: 1.000 Layer 24: 1.000 Layer 32: 1.000

```
Let's break down the downloads for the program month by month.
In the first month, the program had 60 downloads.
In the second month, the program had three times as many as the downloads in the
first month, so it had 60 * 3 = 180 downloads.
In the third month, the downloads were reduced by 30%, which means it had 180 -
(180 * 30 / 100) = 180 - 54 = 126 downloads.
To find the total number of downloads over the three months, we add up the
downloads from each month: 60 + 180 + 126 = 240 + 126 = 366
Thus, the program had a total of \boxed{366} downloads over the three months.
```

Figure 10: **GSM8K generation trajectory.** Steps 20, 40, 49, 60, 62, and 64, from top to bottom, for the GSM8K example in Figure 9. LFRM-L NFT update 500; backbone/prompt EMA $.99/.99$; seed 42; async $(2.5, 2, 1.5)$; 64 denoising steps; CFG 1 and SCCFG 2. Light blue identifies tokens matching the final answer at the same position.