TILTED GENERATOR MATCHING: ONE RL OBJECTIVE FOR EVERY DIFFUSION AND FLOW MODEL

Anonymous authors
Paper under double-blind review

ABSTRACT

Diffusion and flow models are a flexible framework for generative modeling: they can corrupt data in almost any way, live on almost any state space, and generate with almost any sampler. This flexibility, however, makes them hard to post-train. Reinforcement learning (RL) methods for diffusions and flows have been built one at a time, and none applies across this zoo of models. To fix this, we introduce *Tilted Generator Matching* (TGM), a single RL objective for every model expressible in the generator matching framework. This encapsulates nearly all standard generative models, such as flow matching, score-based diffusion, and discrete diffusion. Our key insight is that RL reduces to weighted pretraining: each step of KL-regularized RL is solved exactly by the model's own pretraining loss, tilted by exponentiated reward. TGM is therefore likelihood-free, works with any sampler, and on-policy. We study TGM on diffusion and flow language models (DLMs), which span discrete and continuous state spaces and masked, uniform, and Gaussian noise. TGM is the first RL method demonstrated on uniform-noise diffusion language models and the first designed for Gaussian DLMs, as well as for few-step flow-map language models. On Sudoku, GSM8K, and sentiment tasks, a single configuration improves every DLM family and matches or beats family-specific methods, in one case more than doubling accuracy ($17.0 \rightarrow 37.2$ on GSM8K for uniform DLMs). On flow maps, TGM-Map lifts the entire accuracy–NFE frontier, matching base accuracy with up to $36\times$ fewer sampling steps, and learns from rollouts that are up to $16\times$ cheaper to generate, at no cost in final accuracy.

1 INTRODUCTION

Diffusion and flow models generate data by simulating a Markov process—such as a stochastic or ordinary differential equation (SDE or ODE) or a continuous-time Markov chain (CTMC)—that carries a simple distribution to a target data distribution (Sohl-Dickstein et al., 2015; Ho et al., 2020; Song et al., 2021; Lipman et al., 2023). Across numerous data modalities, their designs vary enormously in state space, noising kernel, process, and training objective. Diffusion and flow language models (together, DLMs) exemplify this variety. Discrete DLMs (Austin et al., 2021; Lou et al., 2024) corrupt tokens by masking (Sahoo et al., 2024; Shi et al., 2024) or uniform replacement (Schiff et al., 2025; Sahoo et al., 2025), generate with a CTMC, and train with the evidence lower bound (ELBO). Continuous DLMs add Gaussian noise to token embeddings or one-hot vectors, generate with an SDE or ODE, and train with the ELBO (Gulrajani & Hashimoto, 2023; Yang et al., 2026) or cross-entropy (Dieleman et al., 2022; Lee et al., 2026a; Chen et al., 2026b). DLMs now rival autoregressive (AR) models at scale (Nie et al., 2025; Inception Labs et al., 2025; Google DeepMind, 2025; Fu et al., 2026a), and continuous DLMs match discrete ones in likelihood and scaling (Chen et al., 2026b; Yang et al., 2026).

Reinforcement learning (RL) has not kept pace with this zoo of models. Standard RL algorithms for AR models, such as PPO (Schulman et al., 2017) and GRPO (Shao et al., 2024), require likelihoods, which diffusion and flow models lack. Diffusion models trained with the ELBO can use it as a proxy; flow models have no cheap equivalent. RL methods have therefore been built *one model type at a time*: DDPO (Black et al., 2024) and DPOK (Fan et al., 2023) for image diffusion, Flow-GRPO (Liu et al., 2025a) and its variants (Xue et al., 2025; Li et al., 2025) for image flows, and diffu-GRPO (Zhao et al., 2025) and its successors (Tang et al., 2026a; Zhu et al., 2026; Wang et al., 2026a;b) for masked DLMs. No existing method spans the whole zoo, across discrete and contin-

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

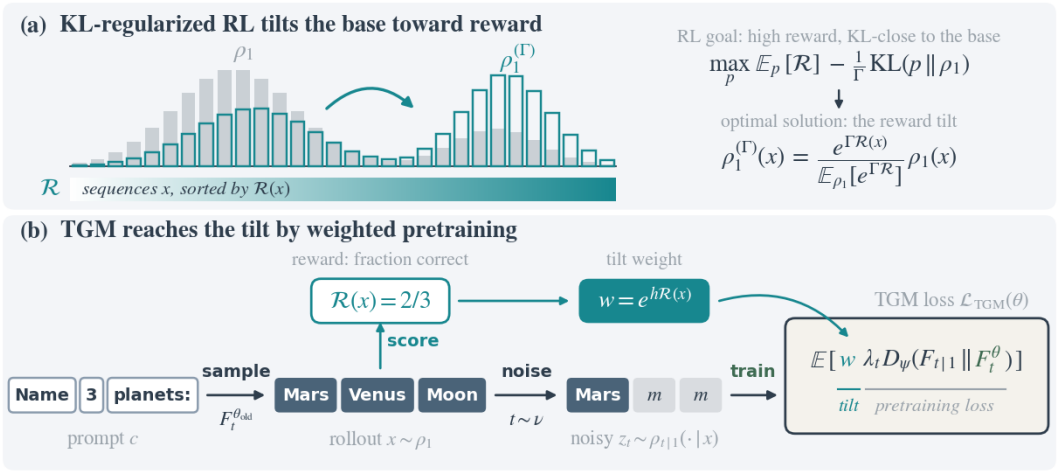

Figure 1: **TGM turns RL into weighted pretraining.** **(a)** KL-regularized RL targets the base distribution ρ_1 tilted toward high reward. **(b)** TGM reaches it with the pretraining loss alone: sample a rollout, score it, noise it with the pretraining kernel (here, masking), and weight it by $w = e^{h\mathcal{R}}$, where $\mathcal{R}$ is the reward and $h > 0$ the size of the tilt increment.

uous state spaces: RL for uniform-noise DLMs has been derived for general discrete processes but never demonstrated on one (Zekri & Boullé, 2025; Wan et al., 2026), and no method at all exists for Gaussian DLMs or for few-step flow-map language models (Lee et al., 2026a; Potaptchik et al., 2026c; Roos et al., 2026).

We close this gap with *Tilted Generator Matching* (TGM), one RL method for every model covered by *generator matching* (Holderrieth et al., 2025), a framework that unifies diffusion and flow models on any state space. RL amounts to tilting the model's distribution toward high reward (Figure 1a). Generator matching pretrains every such model with a simple regression loss on noised samples. Minimizing this loss yields a network that generates the distribution of its training samples. The key insight is that *tilting changes which clean samples are drawn, not how they are noised* (Bergmeister et al., 2026). Pretraining on the model's own samples, weighted by their exponentiated reward, is therefore equivalent to pretraining on the tilted distribution. TGM tilts in small increments to keep the weights stable and recovers tilt matching (Potaptchik et al., 2026a) and discrete tilt matching (Chen et al., 2026c) as special cases. *TGM-Map* extends TGM to few-step flow maps.

In practice, TGM samples rollouts, scores them, noises them, and minimizes the reward-weighted pretraining loss (Figure 1b). It is a *forward-process* RL method: it needs no likelihoods or reward gradients and works with any sampler, including higher-order and few-step ones. Unlike Flow-GRPO (Liu et al., 2025a), it stores only rollout endpoints, not trajectories, so it uses little memory. It inherits pretraining's scalability and stays on-policy when reusing rollouts across gradient steps.

Our primary contributions include:

- **One RL algorithm for all diffusion and flow models.** TGM is the first RL method *designed* for uniform and Gaussian DLMs and, through TGM-Map, the first demonstrated on flow-map language models; it also covers masked DLMs (Section 3).
- **RL as weighted pretraining.** For any model described by generator matching, we prove that each KL-regularized tilt increment is solved exactly by the model's own pretraining loss with rollouts weighted by $e^{h\mathcal{R}}$ (Section 3.1).
- **Accuracy improvements across DLMs.** On sudoku and GSM8K, one set of hyperparameters improves masked (MDLM), uniform (Duo) and Gaussian (FLM) DLMs on 10 of 12 family–task settings, and is the best RL method in its family on 9 of them (Section 4.2).
- **Efficient RL.** TGM-Map trains on few-step rollouts: NFE-8 rollouts match the accuracy of NFE-32 rollouts while generating 16× faster than the 128-NFE default (Section 4.2).

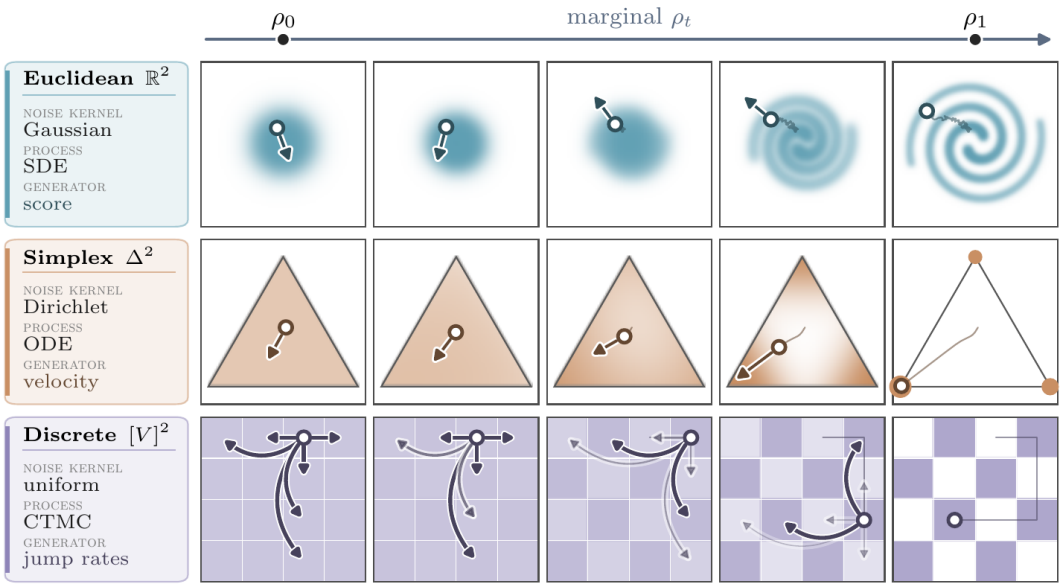

Figure 2: **Three Markov processes, one framework.** Generator matching describes each row by the same choices — state space, noising kernel, process, target — so although the three learn different quantities (score, velocity, jump rates), all are trained by the same regression. Marginals ρ_t carry ρ_0 to ρ_1; the ring shows where a sample is, and the arrows where the process sends it next (the drift, for the SDE). Continuous processes move in one direction, whereas the discrete process can jump to several states, one token at a time, with arrow width showing each jump's rate.

2 Background: Diffusion and Flows via Generator Matching

Generator matching (Holderrieth et al., 2025; Lipman et al., 2024) builds a generative model for a distribution ρ_1 over clean data, in pretraining the data distribution, through a series of choices: where the data lives, how it is noised, which Markov process removes the noise, what the network predicts, and how it is trained. A diffusion or flow model is one set of these choices, and they define it completely (Figure 2). Time runs from 0 (noise) to 1 (data).[1]

Data. A data point is a sequence $x = (x^1, \dots, x^L)$ of L elements, such as pixels or tokens, each in a *data space* $\mathcal{X}$. For images, $\mathcal{X} = \mathbb{R}^d$. For text, each element is a token from a vocabulary $\mathcal{V}$ of K tokens. Discrete diffusion language models (DLMs) keep tokens discrete, $\mathcal{X} = \mathcal{V}$; we write a token v as its one-hot vector $e_v \in \{0,1\}^K$ only so that kernels on $\mathcal{V}$ can be written linearly. Continuous DLMs map each token into Euclidean space $\mathbb{R}^d$ or onto the probability simplex Δ^{K-1} with a fixed injective encoder $\mathcal{E}$, applied at every position, which is either one-hot or a learned embedding; then $\mathcal{X} = \mathcal{E}(\mathcal{V})$. A deterministic decoder $\mathcal{D}$ with $\mathcal{D}(\mathcal{E}(v)) = v$, such as argmax or nearest embedding, maps generated elements back to tokens, on which rewards are computed (Appendix B.3).

Noising Kernel. A *noising kernel* $\kappa_t(\cdot \mid x^i)$ corrupts each element independently into a *state space* $\mathcal{S} \supseteq \mathcal{X}$. At $t = 0$ it forgets the element, $\kappa_0(\cdot \mid x^i) = \kappa_0$, and at $t = 1$ it returns it, $\kappa_1(\cdot \mid x^i) = \delta_{x^i}$. Common kernels are Gaussian noise, masking, and uniform resampling:

$$\mathcal{N}\big(\beta_t x^i, \alpha_t^2 I\big), \qquad \mathrm{Cat}\big(\beta_t x^i + \alpha_t e_{\mathrm{m}}\big), \qquad \mathrm{Cat}\big(\beta_t x^i + \tfrac{\alpha_t}{K}\mathbf{1}\big), \tag{1}$$

on $\mathcal{S} = \mathbb{R}^d$, $\mathcal{V} \cup \{\mathrm{m}\}$, and $\mathcal{V}$, respectively, with signal β_t rising from 0 to 1, noise α_t falling to 0, $\alpha_t = 1 - \beta_t$ for the discrete kernels, and mask token m (more in Table 3).

Probability Paths. The kernel determines the noise distribution $\rho_0 = \prod_i \kappa_0$ and the *conditional path* $\rho_{t|1}(z_t \mid x) = \prod_{i=1}^L \kappa_t(z_t^i \mid x^i)$, which noises a clean sequence x into a noisy sequence z_t.

[1] Diffusion runs time in reverse; we keep the flow matching direction, with clean x and noisy z_t.

Together with ρ_1, it determines the noisy marginal and the posterior,

$$\rho_t(z_t) = \mathbb{E}_{x\sim\rho_1}\big[\rho_{t|1}(z_t \mid x)\big], \qquad \rho_{1|t}(x \mid z_t) = \frac{\rho_1(x)\,\rho_{t|1}(z_t \mid x)}{\rho_t(z_t)}. \tag{2}$$

Pairs are cheap to sample: draw $x \sim \rho_1$, then $z_t \sim \rho_{t|1}(\cdot \mid x)$. The posterior couples all elements through ρ_1 and is intractable; we write $\mathbb{E}_{x|z_t}$ for expectations under it.

Markov Process. A model generates by simulating a Markov process from ρ_0. The process is specified by its *generator*, the instantaneous rate of change of $\mathbb{E}[f(z_t)]$ for test functions f, which is linear in the process's *coefficients* u_t: the velocity of an ODE, the drift of an SDE with a fixed diffusion coefficient, or the jump rates of a continuous-time Markov chain (CTMC) (Lipman et al., 2024).[2] The kernel fixes the marginals ρ_t but not the process that follows them: a Gaussian path is followed by an ODE or a family of SDEs, and a discrete path by many CTMCs. For each clean sequence x, one chooses *conditional coefficients* $u_{t|1}(z \mid x)$, in closed form, whose process follows $\rho_{t|1}(\cdot \mid x)$ and moves each element independently. Their posterior average

$$u_t(z) = \mathbb{E}_{x|z}\big[u_{t|1}(z \mid x)\big] \tag{3}$$

defines a process with marginals ρ_t, so simulating it from ρ_0 generates ρ_1; this is the generalized marginalization trick (Lipman et al., 2024, General Marginalization Trick). In practice, the process is simulated with a discretized sampler.

Target. The coefficients u_t are intractable because they average over the posterior. The network instead predicts a *target*. One chooses a *conditional target* $F_{t|1}(z \mid x)$, a closed-form function of (x, z) in which the conditional coefficients are affine, $u_{t|1}(z \mid x) = a_t(z) + B_t(z)\,F_{t|1}(z \mid x)$. It is fixed by the kernel and the process and does not depend on ρ_1. Posterior averaging preserves the affine map, so $u_t = a_t + B_t F_t$ with the *marginal target*

$$F_t(z) = \mathbb{E}_{x|z}\big[F_{t|1}(z \mid x)\big], \tag{4}$$

which depends on ρ_1 only through the posterior. For images, $F_{t|1}$ is the clean image, the noise, or the velocity (x-, ϵ-, or v-prediction), all equivalent up to affine maps (Lipman et al., 2024). For text, $F_{t|1}$ is the one-hot e_{x^i} of each clean token, whose marginal target is the posterior over tokens, $\pi(v \mid z, i) = \rho_{1|t}(x^i = v \mid z)$, output by a softmax. This holds for all kernels in Equation (1) and for one-hot and embedded tokens alike, since $\mathcal{E}$ is fixed.

Training by Regression. A network F_t^θ regresses the conditional target:

$$\mathcal{L}_{\text{pre}}(\theta) := \mathbb{E}_{t,x,z_t}\Big[\lambda_t\, D_\psi\big(F_{t|1} \,\big\|\, F_t^\theta\big)\Big], \tag{5}$$

with time $t \sim \nu$, data $x \sim \rho_1$, noisy sequence $z_t \sim \rho_{t|1}(\cdot \mid x)$, weight $\lambda_t(z_t) > 0$, and Bregman divergence $D_\psi(a\|b) = \psi(a) - \psi(b) - \langle \nabla\psi(b), a - b\rangle$ for a strictly convex ψ; we suppress the arguments of $F_{t|1}(z_t \mid x)$ and $F_t^\theta(z_t)$. Throughout, expectation subscripts list the random variables, drawn from these distributions unless a different one is written, as in $\mathbb{E}_{x\sim\rho_1^{(\gamma)}}$. Examples are $\mathrm{MSE}(a, b) = \frac{1}{2}\|a - b\|^2$ and the KL divergence, which reduces to $\mathrm{CE}(e_v, b) = -\log b_v$ for one-hot targets. As in conditional flow matching (Lipman et al., 2023), Equation (5) has the same gradients as regressing onto the intractable F_t. The network defines coefficients $u_t^\theta = a_t + B_t F_t^\theta$, and we write ρ_1^θ for the distribution their process generates from ρ_0.

Proposition 1 (Regression recovers the data distribution; Holderrieth et al., 2025)**.** *For any distribution ρ_1 over clean data, the minimizer of $\mathcal{L}_{\text{pre}}$ is the marginal target, $F_t^\theta = F_t$, so the model generates $\rho_1^\theta = \rho_1$. The proof is in Appendix C.1.*

A Bregman divergence is minimized by a conditional mean, so regressing onto $F_{t|1}$ returns its posterior mean F_t; by Equation (3), F_t generates ρ_1. This holds for any full-support ν and positive λ, so these choices do not move the minimizer; for particular weights, Equation (5) is also a likelihood bound. Table 3 lists these choices, and the resulting training loss, for each of the common diffusion and flow models.

[2] Generator matching calls u_t the linear parameterization and writes it F_t; we reserve F_t for what the network predicts, which equals u_t when the network predicts the coefficients directly.

3 TILTED GENERATOR MATCHING

TGM fine-tunes a diffusion or flow model with its own pretraining loss, on its own rollouts, weighting each rollout by a scalar function of its reward. We derive this from reward tilting (Section 3.1), give the estimator we use (Section 3.2), and extend it to flow maps (Section 3.3).

3.1 RL AS WEIGHTED PRETRAINING

Reward Tilting. Given a prompt c (suppressed below) and a reward $\mathcal{R}$, KL-regularized reward maximization, $\max_p \mathbb{E}_{x\sim p}[\mathcal{R}(x)] - \frac{1}{\Gamma} D_{\mathrm{KL}}(p \,\|\, \rho_1)$, is solved by the base distribution *tilted* by $\Gamma > 0$, $\rho_1^{(\Gamma)}(x) = e^{\Gamma\mathcal{R}(x)}\rho_1(x)/\mathbb{E}_{x'\sim\rho_1}[e^{\Gamma\mathcal{R}(x')}]$ (Peters & Schaal, 2007; Korbak et al., 2022; Rafailov et al., 2023). Reweighting base samples by $e^{\Gamma\mathcal{R}}$ has intractably high variance for large Γ, so, following TM and DTM (Potaptchik et al., 2026a; Chen et al., 2026c), we move in small increments. Tilts compose:

$$\rho_1^{(\gamma+h)}(x) = \frac{w(x)}{\mathbb{E}_{x'\sim\rho_1^{(\gamma)}}[w(x')]}\,\rho_1^{(\gamma)}(x), \qquad w(x) := e^{h\mathcal{R}(x)}. \tag{6}$$

We therefore reach Γ through $0 \to h \to \cdots \to \Gamma$. Each increment trains F_t^θ to generate $\rho_1^{(\gamma+h)}$ from rollouts of a frozen copy $F_t^{\theta_{\mathrm{old}}}$ that generates $\rho_1^{(\gamma)}$; then $\theta_{\mathrm{old}} \leftarrow \theta$.

Tilting Only Reweights Clean Data. The pretraining loss of Section 2 averages a *per-sequence loss* over clean data,

$$\mathcal{L}_{\mathrm{pre}}(\theta) = \mathbb{E}_{x\sim\rho_1}\big[\ell_\theta(x)\big], \qquad \ell_\theta(x) := \mathbb{E}_{t\sim\nu,\, z_t\sim\rho_{t|1}(\cdot|x)}\Big[\lambda_t\, D_\psi\big(F_{t|1}(z_t \mid x) \,\big\|\, F_t^\theta(z_t)\big)\Big]. \tag{7}$$

The noising kernel $\rho_{t|1}$ is fixed, so ℓ_θ does not depend on ρ_1: the data distribution only decides *which* sequences are noised. Pretraining on the next tilt means averaging ℓ_θ over $x \sim \rho_1^{(\gamma+h)}$. We cannot sample this distribution, but Equation (6) writes it as the current one times $w/\mathbb{E}[w]$, and ℓ_θ is the same function under both, so the average becomes a weighted average over rollouts of the current model:

$$\mathcal{L}_{\mathrm{pre}}^{(\gamma+h)}(\theta) = \mathbb{E}_{x\sim\rho_1^{(\gamma+h)}}\big[\ell_\theta(x)\big] = \frac{\mathbb{E}_{x\sim\rho_1^{(\gamma)}}\big[w(x)\,\ell_\theta(x)\big]}{\mathbb{E}_{x\sim\rho_1^{(\gamma)}}[w(x)]}. \tag{8}$$

Proposition 2 (Tilt increments are weighted pretraining)**.** *Let $x \sim \rho_1^{(\gamma)}$ be rollouts from any sampler. The unique minimizer of the TGM loss*

$$\mathcal{L}_{\mathrm{TGM}}^{(\gamma+h)}(\theta) := \mathbb{E}_{x\sim\rho_1^{(\gamma)}}\big[w(x)\,\ell_\theta(x)\big] \tag{9}$$

is the target $F_t^{(\gamma+h)}$ of the next tilt, which generates $\rho_1^{(\gamma+h)}$. The proof is in Appendix C.2.

RL thus becomes weighted pretraining (Algorithm 1). It needs no likelihood, and it places no requirement on the sampler, since rollouts enter only through x. It recovers TM and DTM with $c = 0$ (Potaptchik et al., 2026a; Chen et al., 2026c), and recovers the naive one-shot approach as the single increment $h = \Gamma$.

3.2 TGM IN PRACTICE

We optimize Equation (9) over minibatches of B prompts with G rollouts $x_{c,g}$ each, and make three changes. First, rewards differ in offset and scale across prompts, so we tilt by the advantage: $w_{c,g} = e^{hA_{c,g}}$ with $A_{c,g} = \mathrm{clip}\big((\mathcal{R}_{c,g} - \bar{\mathcal{R}}_c)/\sigma,\, -a,\, a\big)$, where $\bar{\mathcal{R}}_c$ is the group mean and σ the batch standard deviation (Liu et al., 2025b). Second, following DTM (Chen et al., 2026c), we add a control variate: we split $w = 1 + (w-1)$ and, in the reward-free part, replace the conditional target $F_{t|1}$ by its conditional mean given z_t, the current model $F_t^{\theta_{\mathrm{old}}}$. This leaves the expected gradient unchanged (Appendix C.3). Third, for non-verifiable rewards, one may add a KL term to

Algorithm 1 Tilted Generator Matching (ideal form)

Require: model F_t^θ, reward $\mathcal{R}$, increment h, target tilt Γ, path $\rho_{t|1}$, target $F_{t|1}$, divergence D_ψ

for $\gamma = 0, h, 2h, \ldots, \Gamma - h$ **do** ▷ one tilt increment
  $F_t^{\theta_{\text{old}}} \leftarrow F_t^\theta$ ▷ freeze the current model
  repeat
    $x \sim \rho_1^{(\gamma)}$ using $F_t^{\theta_{\text{old}}}$ ▷ generate a rollout with any sampler
    $w \leftarrow e^{h\mathcal{R}(x)}$ ▷ score reward and form tilt weight
    $t \sim \nu$; $z_t \sim \rho_{t|1}(\cdot \mid x)$ ▷ noise the rollout
    $\mathcal{L} \leftarrow w\, \lambda_t\, D_\psi\big(F_{t|1}(z_t \mid x) \,\|\, F_t^\theta(z_t)\big)$ ▷ weighted pretraining loss
    $\theta \leftarrow$ update θ with $\nabla_\theta \mathcal{L}$ ▷ gradient step
  until converged

the base model, which we omit by default (Yu et al., 2025; Liu et al., 2025b). With $t \sim \nu$ and $z_t \sim \rho_{t|1}(\cdot \mid x_{c,g})$ drawn per rollout, the minibatch loss is

$$\hat{\mathcal{L}}(\theta) = \frac{1}{BG} \sum_{c,g} \lambda_t \Big[\underbrace{(w_{c,g} - 1)\, D_\psi\big(F_{t|1} \,\|\, F_t^\theta\big)}_{\text{advantage-weighted pretraining}} + \underbrace{D_\psi\big(F_t^{\theta_{\text{old}}} \,\|\, F_t^\theta\big)}_{\text{anchor to current model}} + \underbrace{\eta\, D_\psi\big(F_t^{\theta_{\text{base}}} \,\|\, F_t^\theta\big)}_{\text{anchor to base}} \Big]. \quad (10)$$

Interpreting the Terms. The three terms of Equation (10) mirror other forward-process RL methods, which pair a reward-weighted regression with anchors to the rollout and reference models (Zheng et al., 2026; Xue et al., 2026; Bergmeister et al., 2026; Kveton et al., 2026; Choi et al., 2026). The *advantage-weighted pretraining* term is the only one that sees the reward. The *anchor to the current model* regresses the model onto its own current prediction, the average conditional target of its rollouts; for cross-entropy it is a KL divergence to $F_t^{\theta_{\text{old}}}$, up to a constant, and it has zero gradient at $\theta = \theta_{\text{old}}$. It keeps each update local and keeps the loss well-posed when the first term's coefficient is negative. The *anchor to the base model* guards against reward hacking in the case where the reward is imperfect.

Positive and Negative Samples. The coefficient $w_{c,g} - 1$ has the sign of the advantage. An above-average rollout pulls the model toward its conditional target, a below-average rollout pushes the model away from it, and an average rollout leaves the model unchanged. Pushing away from failures matters for RL (Zhu et al., 2025a; Chen et al., 2026a; Zheng et al., 2026). DTM tilts by raw nonnegative rewards, so $w \geq 1$ and no rollout is ever pushed away (Chen et al., 2026c); centering the reward lets below-average rollouts repel without moving the minimizer.

Sample Reuse. Rollouts cost far more than loss evaluations, so we take μ gradient steps on each batch of rollouts, re-noising them at every step. The objective, the next tilt of the rollouts' distribution, is fixed during the increment, so reuse needs no likelihood ratio or clipping. The full algorithm, with all of these choices in place, is Algorithm 2.

3.3 TGM-Map: Tilting Flow Maps

A flow map $X_{s,t}$ jumps along the solution of an ODE from time s to time t (Boffi et al., 2025). It is not a generator, so Proposition 2 does not apply to it directly, but its training loss splits into a pretraining loss and a reward-free consistency term. Flow map language models (Lee et al., 2026a) parameterize $X_{s,t}$ by a two-time head $\delta_{s,t}^\theta$ whose diagonal $\delta_{t,t}^\theta$ is an ordinary categorical head F_t^θ, driving the flow of Section 2. They train it with $\mathcal{L}_{\text{map}} = \mathcal{L}_{\text{pre}} + \mathcal{L}_{\text{sg}}$, where $\mathcal{L}_{\text{pre}}$ acts on the diagonal and $\mathcal{L}_{\text{sg}} = \mathbb{E}\big[D_{\text{KL}}(\bar{\delta}_{s,t} \,\|\, \delta_{s,t}^\theta(z_s))\big]$ enforces the semigroup condition for $s < t$ through a stop-gradient teacher $\bar{\delta}_{s,t}$. The two terms train disjoint parts of δ: the diagonal alone fixes the dynamics, and, given the diagonal, the unique critical point of $\mathcal{L}_{\text{sg}}$ is the flow map of those dynamics whenever z_s has full support (Lee et al., 2026a). TGM-Map tilts the diagonal and keeps the consistency term,

$$\mathcal{L}_{\text{TGM-Map}}^{(\gamma+h)}(\theta) = \mathcal{L}_{\text{TGM}}^{(\gamma+h)}(\theta) + \mathcal{L}_{\text{sg}}(\theta), \quad (11)$$

with both terms evaluated on noised rollouts.

Table 1: **RL across DLM families.** Accuracy (%) on Sudoku and GSM8K; held-out $P(\text{pos})$ (%) on sentiment. Bold: best RL method per family; AR is shown for context and is not bolded. –: omitted, as the FLM base solves almost no GSM8K problems. Sentiment is at a matched budget of **100 gradient updates**; Sudoku and GSM8K run to convergence.

		Sudoku (acc.)			GSM8K	Sentiment
Family	Method	Hard	Extreme	Evil	Acc.	$P(\text{pos})$
AR	AR	0.4	0.1	0.0	61.5	34.9
	+ RAFT	0.5	0.1	0.0	63.8	34.2
	+ GRPO	0.4	0.1	0.0	68.5	46.3
Masked	MDLM	34.9	10.7	3.4	12.9	33.2
	+ RAFT	35.4	9.1	2.8	12.6	31.0
	+ diffu-GRPO	39.1	12.4	**4.3**	16.9	38.6
	+ TGM (ours)	**43.5**	**15.1**	3.1	**20.7**	**39.4**
Uniform	Duo	55.8	26.1	15.5	17.0	29.3
	+ RAFT	59.8	31.4	16.6	20.5	32.7
	+ TGM (ours)	**66.0**	**39.8**	**20.2**	**37.2**	**35.1**
Gaussian	FLM	49.4	14.7	8.4	0.2	32.7
	+ RAFT	53.5	15.5	8.5	–	55.2
	+ Flow-GRPO	51.7	14.9	8.4	–	35.5
	+ TGM (ours)	**58.9**	**18.3**	**11.8**	–	**59.3**

Proposition 3 (TGM-Map)**.** *At the unique critical point of Equation* (11)*,* $\delta^\theta_{t,t} = F_t^{(\gamma+h)}$ *and* $X^\theta_{s,t}$ *is the flow map of the ODE it drives, so* $X^\theta_{0,1}$ *generates* $\rho_1^{(\gamma+h)}$*. The proof is in Appendix C.5.*

The reward enters only through the diagonal; the consistency term carries the tilt across all intervals. This is the construction TM suggests for flow maps (Potaptchik et al., 2026a), here with cross-entropy on the two-time head. Rollouts cost a few network evaluations each, so the RL loop costs little more than the updates themselves (Section 4). Like TGM, TGM-Map runs in the ideal form of Algorithm 1 or the practical form of Algorithm 2: add $\mathcal{L}_{\text{sg}}$ to the loss each computes.

4 Experiments

We ask two questions. **(Q1)** Does one RL objective, TGM, improve every DLM family, and how does it compare with family-specific methods (Table 1)? **(Q2)** For few-step flow maps, does TGM-Map move the accuracy–NFE Pareto frontier, and can RL itself be made faster by training on few-step rollouts (Figure 3 and Table 2)?

4.1 Setup

Tasks (Appendix E.1). *Sudoku* and *GSM8K* follow S-FLM (Deschenaux & Gulcehre, 2026). S-FLM's Sudoku tiers (40 / 35 / 30 of 81 cells given) are too easy for our base models, so we use harder ones: hard / extreme / evil, with 30 / 25 / 20 given. The reward is the fraction of blanks filled correctly, and we report the exact solve rate. On GSM8K (Cobbe et al., 2021), models are pretrained on TinyGSM (Liu et al., 2023), which writes solutions as Python programs; the reward is 1 if the program returns the correct answer, and we report zero-shot pass@1. *Sentiment* is unconditional OpenWebText generation rewarded by a sentiment classifier; to detect reward hacking (Gao et al., 2023), we report the positive probability under a *held-out* sentiment classifier, used neither in training nor in checkpoint selection.

Models (Appendix E.2). Following S-FLM, each model is trained from scratch on one task, with one backbone size per task. We cover masked (MDLM; Sahoo et al., 2024), uniform (Duo; Sahoo et al., 2025) and Gaussian (FLM; Lee et al., 2026a) DLMs, the few-step flow map FMLM+ (Agarwal et al., 2026), and an AR reference.

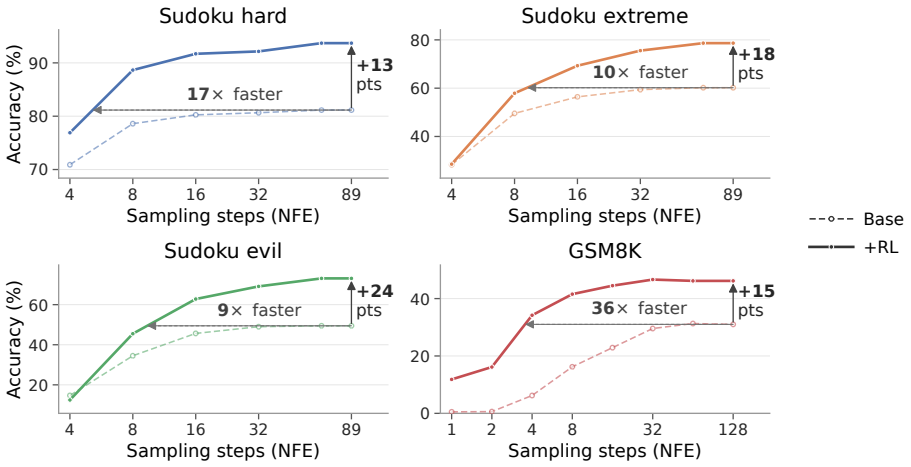

Figure 3: **TGM-Map moves the accuracy–NFE frontier of FMLM+.** Vertical arrows: gain at the largest budget. Horizontal arrows: the NFE reduction at which RL first matches the base's best accuracy over the grid (Appendix E.5).

RL baselines (Appendix E.3). **No existing RL method applies to every DLM family** (Table 4). A simple baseline is RAFT (Dong et al., 2023), a supervised fine-tuning method that trains on each group's highest-reward rollout with the model's pretraining loss. For DLMs, family-specific RL methods exist only for masked DLMs, such as diffu-GRPO (Zhao et al., 2025), which we compare against. For Gaussian FLMs, we adapt Flow-GRPO (Liu et al., 2025a), developed for images, to text for the first time by making FLM's Euler sampler stochastic. For uniform DLMs, RL has been derived for general or non-masked discrete processes (Zekri & Boullé, 2025; Wan et al., 2026) but, to our knowledge, never demonstrated on a uniform-noise language model, so we report the first such result rather than claiming the first method. For flow-map language models no RL method exists: RL for flow maps has been developed only for continuous image and video generators (Oertell et al., 2024; Li et al., 2026b; Huang et al., 2026; Mammadov et al., 2026), never for language. The AR reference uses GRPO (Shao et al., 2024). We compare these methods against TGM, and TGM-Map for flow maps, which apply unchanged to every family. Across methods we match the prompts, the rollouts per prompt, the gradient updates and the evaluation sampler.

Sampling and NFE (Appendix E.4). Each model uses its published sampler, identical across RL methods. We measure cost by the number of function evaluations (NFE), i.e. network forward passes. Within a model class every forward pass costs the same, so **NFE is a hardware-agnostic proxy for sampling time**. Unless stated otherwise, DLMs use 89 NFE on Sudoku (the solution length), 128 on GSM8K and 1023 on sentiment. AR models take one NFE per token: on Sudoku and sentiment they generate the same fixed length, and on GSM8K they stop at the end-of-text token, after 159 generated tokens on average.

4.2 Results

TGM improves every DLM family. Table 1 shows that **TGM is the best RL method in nearly all family–task settings**. It outperforms RAFT, the SFT-inspired method and the only other one that applies to every family, across the board. On average across models, TGM improves accuracy on Sudoku hard / extreme / evil by $+9.4$ / $+7.2$ / $+2.6$ points, while RAFT achieves $+2.9$ / $+1.5$ / $+0.2$; on GSM8K, TGM improves by $+14.0$ versus $+1.6$ (excluding FLM). As in S-FLM (Deschenaux & Gulcehre, 2026), the pretrained FLM solves almost no GSM8K problems, leaving no learning signal for any RL method. On sentiment, it raises held-out positive probability by $+6.0$ versus $+0.6$, again excluding FLM. TGM also beats family-specific RL: on MDLM, its gains are roughly double those of diffu-GRPO on every task except Sudoku evil; on FLM, they are at least $4\times$ those of our adapted Flow-GRPO on every Sudoku tier. On Duo, where no RL method existed, TGM more than doubles GSM8K accuracy ($17.0 \rightarrow 37.2$). **Overall, TGM outperforms both universal SFT-style methods and family-specific RL, while enabling RL on model families that had no method before.**

Table 2: **Few-step rollouts** (FMLM+ with TGM-Map, GSM8K). Rollout NFE varies; evaluation NFE is fixed at 32. Earlier configuration, all arms at 800 optimizer steps (Appendix E.6); speedup is relative to 128-NFE rollouts.

Rollout NFE	Acc. (%)	Samples/s	Speedup
base (no RL)	31.6	—	—
32	$41.6_{(+10.0)}$	9.9	$4.1\times$
16	$42.0_{(+10.4)}$	19.6	$8.1\times$
8	$41.6_{(+10.0)}$	39.0	$16.2\times$
4	$38.7_{(+7.1)}$	75.9	$31.5\times$
2	$29.3_{(-2.3)}$	130.3	$54.2\times$

TGM-Map improves the accuracy–NFE Pareto frontier. Flow maps sample from DLMs in few steps, potentially unlocking large inference speedups, yet no published RL method applies to them out of the box. TGM-Map fills this gap: it generates rollouts at a fixed NFE, scores them, and updates the model with the weighted pretraining objective. We then evaluate the fine-tuned FMLM+ across a sweep of NFE (Figure 3). **RL shifts the Pareto frontier up and to the left.** At the default NFE, TGM-Map adds $\mathbf{+13}$ / $\mathbf{+18}$ / $\mathbf{+24}$ **points** on Sudoku hard / extreme / evil and $\mathbf{+15}$ **points** on GSM8K. TGM-Map also enables order-of-magnitude speedups, matching the base's best accuracy with $\mathbf{17\times}$ / $\mathbf{10\times}$ / $\mathbf{9\times}$ **fewer NFE** on Sudoku and $\mathbf{36\times}$ **fewer** on GSM8K. On GSM8K, our post-trained model fills a 447-token canvas in 4 NFE and outperforms the base at 128 NFE, reaching 34.2% accuracy on challenging reasoning problems. An example generation is in Appendix F.

Few-step rollouts make RL faster. Rollout generation in post-training is expensive and is often the bottleneck in RL systems. DLMs, and flow maps in particular, turn this cost into a knob: their rollout NFE can be set independently of the NFE used at deployment. Table 2 tests this by varying the rollout NFE during RL training while evaluating at a fixed NFE. Cutting rollout NFE from 32 to 8 costs no accuracy ($+10.0$ to $+10.4$ points), so **rollouts can be $\mathbf{16\times}$ cheaper than the default 128 NFE at no loss in accuracy**. Below that, it degrades gracefully, keeping 71% of the gain at 4 NFE ($32\times$ cheaper), and collapses only at 2 NFE, where the policy falls below the base. Thus **TGM-Map makes RL dramatically faster by post-training directly on few-step flow maps.**

5 Discussion

TGM turns RL for any diffusion or flow model into weighted pretraining: the model regresses its own conditional target on its own rollouts, re-noised by its own kernel, with each rollout weighted by its exponentiated reward. One objective improves masked, uniform and Gaussian DLMs, needs no likelihood, ratio or stored trajectory, and extends to flow maps, where training on few-step rollouts makes RL itself cheap.

Relation to prior work. That a KL-regularized RL step is solved by reward-weighted maximum likelihood is classical (Peters & Schaal, 2007; Abdolmaleki et al., 2018; Norouzi et al., 2016), and it has recently been made exact for autoregressive LMs, where Boltzmann-weighted fine-tuning recovers the KL-regularized RLVR optimum (Shu et al., 2026). For diffusion and flow models, forward-process methods use the same fact that the tilt leaves the noising kernel unchanged (Potaptchik et al., 2026a; Bergmeister et al., 2026; Chen et al., 2026c), but each is derived for one model family: Gaussian flows and SDEs, or masked diffusion. Our contribution is the scope. Stating the identity at the level of generator matching makes it exact for every kernel, process and state space at once, and lets a single objective reach families that had no RL method: Gaussian and uniform DLMs and flow-map language models. TGM also sidesteps the two constraints of the popular alternatives. GRPO-style methods for DLMs need a likelihood proxy that exists only for some kernels (Zhao et al., 2025; Wang et al., 2026a), and reverse-process methods, including recent ones for general discrete flows (Liu et al., 2025a; Wan et al., 2026), need a stochastic sampler with tractable steps and stored trajectories. Appendix A gives a fuller account, including concurrent unifications of reward fine-tuning for continuous diffusion (Xu et al., 2026; Li et al., 2026a).

AI USE STATEMENT

In this work, we used generative AI tools throughout the implementation and writing of the paper. AI assistants were used to write and refactor research code, to orchestrate and monitor experiments on our cluster, to aggregate and cross-check experimental results, to search and verify the related literature, and to draft and edit the text of this paper, including its tables and the code that produces its figures. We did not use generative AI to produce any experimental result: every number reported here is the output of a training or evaluation run of the models described in Appendix E, and no reported value was generated, imputed or interpolated by a language model. Likewise, the core theoretical contribution — the tilted objective of Section 3.1 and the results proved in Appendix C — was developed by the authors; AI assistance there was limited to checking derivations and prose.

We have reviewed all AI-assisted work. AI-written code was reviewed by the authors and exercised by the project's test suite before use; claims about prior work were verified against primary sources (publisher pages and the arXiv record) rather than accepted from a model's summary; and every number appearing in a table or figure was traced back to the run that produced it. We take responsibility for the final content of this work, including all text, claims and artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work uses only public, standard benchmarks — GSM8K (Cobbe et al., 2021), TinyGSM (Liu et al., 2023), OpenWebText (Gokaslan & Cohen, 2019) and programmatically generated Sudoku puzzles — as released. No human subjects were involved and no new data was collected. The models are research artifacts of roughly 170M parameters, trained from scratch on single tasks, and are not general-purpose generators. We are not aware of specific Code of Ethics concerns raised by this submission.

REPRODUCIBILITY STATEMENT

The assumptions behind our theoretical claims and the complete proofs are in Appendix C; the statements as used in the main text appear in Section 3.1 and Section 3.2, and the corresponding proofs in Appendix C. The training procedure is given as pseudocode in Appendix D. For the experiments, Appendix E.1 specifies each task's data, prompt format, reward and evaluation metric, including the exact answer-extraction and scoring rules; Appendix E.2 and Table 5 give the model architectures and the pretraining settings for every checkpoint we start from; Appendix E.3 and Table 6 give the RL hyperparameters, which are shared across methods within a task, together with the method-specific settings for each baseline; and Appendix E.4 specifies the sampler and the NFE accounting used at evaluation, which differ across model families and materially affect absolute numbers. Where an experiment departs from the shared protocol we say so rather than smoothing it over: the departures are listed in Appendix E.3, and the rollout-NFE ablation, which predates the main experiments and uses a different configuration, is described separately in Appendix E.6. We will publicly release the source code for the models, the training procedure and the evaluation harness upon publication.

REFERENCES

Abbas Abdolmaleki, Jost Tobias Springenberg, Yuval Tassa, Remi Munos, Nicolas Heess, and Martin Riedmiller. Maximum a posteriori policy optimisation. *arXiv preprint arXiv:1806.06920*, 2018.

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.

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

Arindam Banerjee, Srujana Merugu, Inderjit S. Dhillon, and Joydeep Ghosh. Clustering with Bregman divergences. *Journal of Machine Learning Research*, 6:1705–1749, 2005.

Loubna Ben Allal, Anton Lozhkov, Elie Bakouch, Gabriel Martín Blázquez, Guilherme Penedo, Lewis Tunstall, Andrés Marafioti, Hynek Kydlíček, Agustín Piqueres Lajarín, Vaibhav Srivastav, Joshua Lochner, Caleb Fahlgren, Xuan-Son Nguyen, Clémentine Fourrier, Ben Burtenshaw, Hugo Larcher, Haojun Zhao, Cyril Zakka, Mathieu Morlon, Colin Raffel, Leandro von Werra, and Thomas Wolf. SmolLM2: When smol goes big — data-centric training of a fully open small language model. In *Conference on Language Modeling (COLM)*, 2025.

Andreas Bergmeister, Stefanie Jegelka, Nikolas Nüsken, Carles Domingo-Enrich, and Jakiw Pidstrigach. Reinforce adjoint matching: Scaling RL post-training of diffusion and flow-matching models. *arXiv preprint arXiv:2605.10759*, 2026.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. Training diffusion models with reinforcement learning. In *International Conference on Learning Representations (ICLR)*, 2024.

Nicholas M. Boffi, Michael S. Albergo, and Eric Vanden-Eijnden. Flow map matching with stochastic interpolants: A mathematical framework for consistency models. *Transactions on Machine Learning Research*, 2025.

Andrew Campbell, Joe Benton, Valentin De Bortoli, Tom Rainforth, George Deligiannidis, and Arnaud Doucet. A continuous time framework for discrete denoising models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Andrew Campbell, Jason Yim, Regina Barzilay, Tom Rainforth, and Tommi Jaakkola. Generative flows on discrete state-spaces: Enabling multimodal flows with applications to protein co-design. In *International Conference on Machine Learning (ICML)*, 2024.

Huayu Chen, Kaiwen Zheng, Qinsheng Zhang, Ganqu Cui, Lifan Yuan, Yin Cui, Haotian Ye, Tsung-Yi Lin, Ming-Yu Liu, Jun Zhu, and Haoxiang Wang. NFT: Bridging supervised learning and reinforcement learning in math reasoning. In *International Conference on Learning Representations (ICLR)*, 2026a.

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

Yuyuan Chen, Shiyi Wang, Peter Potaptchik, Jaeyeon Kim, and Michael S. Albergo. Discrete tilt matching. *arXiv preprint arXiv:2604.18739*, 2026c.

Jaemoo Choi, Wei Guo, Yuchen Zhu, Arash Vahdat, Molei Tao, Julius Berner, and Yongxin Chen. Scaling reinforcement learning for diffusion models via velocity matching. *arXiv preprint arXiv:2608.23664*, 2026.

Kevin Clark, Paul Vicol, Kevin Swersky, and David J Fleet. Directly fine-tuning diffusion models on differentiable rewards. In *International Conference on Learning Representations (ICLR)*, 2024.

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.

Sanjit Dandapanthula and Nicholas M. Boffi. Are we really tilting? the mechanics of reward guidance in flow and diffusion models. *arXiv preprint arXiv:2606.02884*, 2026.

Peter Dayan and Geoffrey E. Hinton. Using expectation-maximization for reinforcement learning. *Neural Computation*, 9(2):271–278, 1997.

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

Sander Dieleman, Laurent Sartran, Arman Roshannai, Nikolay Savinov, Yaroslav Ganin, Pierre H. Richemond, Arnaud Doucet, Robin Strudel, Chris Dyer, Conor Durkan, Curtis Hawthorne, Rémi Leblond, Will Grathwohl, and Jonas Adler. Continuous diffusion for categorical data. *arXiv preprint arXiv:2211.15089*, 2022.

DiffusionGemma Team. DiffusionGemma technical report. *arXiv preprint arXiv:2608.00146*, 2026.

Carles Domingo-Enrich, Michal Drozdzal, Brian Karrer, and Ricky T. Q. Chen. Adjoint matching: Fine-tuning flow and diffusion generative models with memoryless stochastic optimal control. *arXiv preprint arXiv:2409.08861*, 2024.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. RAFT: Reward rAnked FineTuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023.

Fredrik Esscher. On the probability function in the collective theory of risk. *Skandinavisk Aktuarietidskrift*, 15:175–195, 1932.

Ying Fan, Olivia Watkins, Yuqing Du, Hao Liu, Moonkyung Ryu, Craig Boutilier, Pieter Abbeel, Mohammad Ghavamzadeh, Kangwook Lee, and Kimin Lee. DPOK: Reinforcement learning for fine-tuning text-to-image diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Yonggan Fu, Lexington Whalen, Abhinav Garg, Chengyue Wu, Maksim Khadkevich, et al. Nemotron-Labs-Diffusion: A tri-mode language model unifying autoregressive, diffusion, and self-speculation decoding. *arXiv preprint arXiv:2607.05722*, 2026a.

Zheming Fu, Ruizhe He, Wei Shang, Xiaoxiao Ma, Lei Wang, Chang Liu, and Siming Fu. FlowAWR: Online adaptive flow reinforcement via advantage-weighted rectification. *arXiv preprint arXiv:2606.30376*, 2026b.

Leo Gao, John Schulman, and Jacob Hilton. Scaling laws for reward model overoptimization. In *International Conference on Machine Learning (ICML)*, 2023.

Itai Gat, Tal Remez, Neta Shaul, Felix Kreuk, Ricky T. Q. Chen, Gabriel Synnaeve, Yossi Adi, and Yaron Lipman. Discrete flow matching. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2024.

Aaron Gokaslan and Vanya Cohen. OpenWebText corpus. http://Skylion007.github.io/OpenWebTextCorpus, 2019.

Shansan Gong, Ruixiang Zhang, Huangjie Zheng, Jiatao Gu, Navdeep Jaitly, Lingpeng Kong, and Yizhe Zhang. DiffuCoder: Understanding and improving masked diffusion models for code generation. In *International Conference on Learning Representations (ICLR)*, 2026.

Google DeepMind. Gemini diffusion. https://deepmind.google/models/gemini-diffusion/, 2025.

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

Ishaan Gulrajani and Tatsunori B. Hashimoto. Likelihood-based diffusion language models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2020.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky T. Q. Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary Markov processes. In *International Conference on Learning Representations (ICLR)*, 2025.

Peter Holderrieth, Douglas Chen, Luca Eyring, Ishin Shah, Giri Anantharaman, Yutong He, Zeynep Akata, Tommi Jaakkola, Nicholas Matthew Boffi, and Max Simchowitz. Diamond maps: Efficient reward alignment via stochastic flow maps. *arXiv preprint arXiv:2602.05993*, 2026.

Yushi Huang, Xiangxin Zhou, Jun Zhang, Liefeng Bo, and Tianyu Pang. MeanFlowNFT: Bringing forward-process RL to average-velocity generators. *arXiv preprint arXiv:2607.15273*, 2026.

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

Muhammad Khalifa, Hady Elsahar, and Marc Dymetman. A distributional approach to controlled text generation. In *International Conference on Learning Representations (ICLR)*, 2021.

Diederik P. Kingma, Tim Salimans, Ben Poole, and Jonathan Ho. Variational diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2021.

Tomasz Korbak, Hady Elsahar, German Kruszewski, and Marc Dymetman. On reinforcement learning and distribution matching for fine-tuning language models with no catastrophic forgetting. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2022.

Branislav Kveton, Anup Rao, Subhojyoti Mukherjee, Krishna Kumar Singh, and Viet Dac Lai. AdvantageFlow: Advantage-weighted least squares for RL in flow models. *arXiv preprint arXiv:2605.26013*, 2026.

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

Jaewoo Lee, Hyeongyu Kang, Dohyun Kim, Kyuil Sim, Woocheol Shin, Minsu Kim, Taeyoung Yun, Jeongjae Lee, Sanghyeok Choi, Tabitha Edith Lee, Jong Chul Ye, and Jinkyoo Park. Aligning few-step generative models by amortizing sample-based variational inference. *arXiv preprint arXiv:2605.26552*, 2026b.

Junzhe Li, Yutao Cui, Tao Huang, Weijie Kong, Yiming Cheng, Chuxuan Zeng, Yinping Ma, Chun Fan, Miles Yang, Zhao Zhong, and Liefeng Bo. MixGRPO: Unlocking flow-based GRPO efficiency with mixed ODE-SDE. *arXiv preprint arXiv:2507.21802*, 2025.

Zeyang Li, Yunan Wang, Paolo Giaretta, and Navid Azizan. Newton matching for generative modeling: A unified framework for fine-tuning and sampling. *arXiv preprint arXiv:2609.05727*, 2026a.

Zhiqi Li, Wen Zhang, and Bo Zhu. Flow-map GRPO: Reinforcement learning for few-step flow-map generators via anchored stochastic composition. *arXiv preprint arXiv:2607.00535*, 2026b.

Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. In *International Conference on Learning Representations (ICLR)*, 2023.

Yaron Lipman, Marton Havasi, Peter Holderrieth, Neta Shaul, Matt Le, Brian Karrer, Ricky T. Q. Chen, David Lopez-Paz, Heli Ben-Hamu, and Itai Gat. Flow matching guide and code. *arXiv preprint arXiv:2412.06264*, 2024.

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

Jie Liu, Gongye Liu, Jiajun Liang, Yangguang Li, Jiaheng Liu, Xintao Wang, Pengfei Wan, Di Zhang, and Wanli Ouyang. Flow-GRPO: Training flow matching models via online RL. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding R1-Zero-like training: A critical perspective. In *Conference on Language Modeling (COLM)*, 2025b.

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

Daniel Loureiro, Francesco Barbieri, Leonardo Neves, Luis Espinosa Anke, and Jose Camacho-Collados. TimeLMs: Diachronic language models from Twitter. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (ACL): System Demonstrations*, 2022.

Abbas Mammadov, Jerry Y. Huang, Justin Lin, Partha Kaushik, Sheel Shah, Kartik Nair, Yee Whye Teh, and Nicholas M. Boffi. WTF?! simulation-free reinforcement learning with Wasserstein-tilted flow maps. *arXiv preprint arXiv:2609.27033*, 2026.

David McAllister, Songwei Ge, Brent Yi, Chung Min Kim, Ethan Weber, Hongsuk Choi, Haiwen Feng, and Angjoo Kanazawa. Flow matching policy gradients. In *International Conference on Learning Representations (ICLR)*, 2026.

Shen Nie, Fengqi Zhu, Zebin You, Xiaolu Zhang, Jingyang Ou, Jun Hu, Jun Zhou, Yankai Lin, Ji-Rong Wen, and Chongxuan Li. Large language diffusion models. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Mohammad Norouzi, Samy Bengio, Zhifeng Chen, Navdeep Jaitly, Mike Schuster, Yonghui Wu, and Dale Schuurmans. Reward augmented maximum likelihood for neural structured prediction. *arXiv preprint arXiv:1609.00150*, 2016.

Owen Oertell, Jonathan D. Chang, Yiyi Zhang, Kianté Brantley, and Wen Sun. RL for consistency models: Faster reward guided text-to-image generation. *arXiv preprint arXiv:2404.03673*, 2024.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.

Xue Bin Peng, Aviral Kumar, Grace Zhang, and Sergey Levine. Advantage-weighted regression: Simple and scalable off-policy reinforcement learning. *arXiv preprint arXiv:1910.00177*, 2019.

Jan Peters and Stefan Schaal. Reinforcement learning by reward-weighted regression for operational space control. In *International Conference on Machine Learning (ICML)*, 2007.

Peter Potaptchik, Cheuk-Kit Lee, and Michael S. Albergo. Tilt matching for scalable sampling and fine-tuning. In *International Conference on Machine Learning (ICML)*, 2026a.

Peter Potaptchik, Adhi Saravanan, Abbas Mammadov, Alvaro Prat, Michael S. Albergo, and Yee Whye Teh. Meta flow maps enable scalable reward alignment. *arXiv preprint arXiv:2601.14430*, 2026b.

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

Mihir Prabhudesai, Anirudh Goyal, Deepak Pathak, and Katerina Fragkiadaki. Aligning text-to-image diffusion models with reward backpropagation. *arXiv preprint arXiv:2310.03739*, 2023.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2023.

Jarrid Rector-Brooks, Mohsin Hasan, Zhangzhi Peng, Zachary Quinn, Chenghao Liu, Sarthak Mittal, Nouha Dziri, Michael Bronstein, Yoshua Bengio, Pranam Chatterjee, Alexander Tong, and Avishek Joey Bose. Steering masked discrete diffusion models via discrete denoising posterior prediction. *arXiv preprint arXiv:2410.08134*, 2024.

Daan Roos, Oscar Davis, Floor Eijkelboom, Michael Bronstein, Max Welling, İsmail İlkan Ceylan, Luca Ambrogioni, and Jan-Willem van de Meent. Categorical flow maps. *arXiv preprint arXiv:2602.12233*, 2026.

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

Subham Sekhar Sahoo, Justin Deschenaux, Aaron Gokaslan, Guanghan Wang, Justin Chiu, and Volodymyr Kuleshov. The diffusion duality. In *International Conference on Machine Learning (ICML)*, 2025.

Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. DistilBERT, a distilled version of BERT: Smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019.

Yair Schiff, Subham Sekhar Sahoo, Hao Phung, Guanghan Wang, Sam Boshar, Hugo Dalla-torre, Bernardo P. de Almeida, Alexander Rush, Thomas Pierrot, and Volodymyr Kuleshov. Simple guidance mechanisms for discrete diffusion models. In *International Conference on Learning Representations (ICLR)*, 2025.

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

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. DeepSeekMath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*, 2024.

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

Yao Shu, Chenxing Wei, Hongbin Lin, Shuang Qiu, and Hui Xiong. Reference-sampled boltzmann projection for KL-Regularized RLVR: Target-matched weighted SFT, finite one-shot gaps, and policy mirror descent. *arXiv preprint arXiv:2605.02469*, 2026.

Avi Singh, John D. Co-Reyes, Rishabh Agarwal, Ankesh Anand, Piyush Patil, Xavier Garcia, Peter J. Liu, James Harrison, Jaehoon Lee, Kelvin Xu, Aaron Parisi, Abhishek Kumar, Alex Alemi, Alex Rizkowsky, Azade Nova, Ben Adlam, Bernd Bohnet, Gamaleldin Elsayed, Hanie Sedghi, Igor Mordatch, Isabelle Simpson, Izzeddin Gur, Jasper Snoek, Jeffrey Pennington, Jiri Hron, Kathleen Kenealy, Kevin Swersky, Kshiteej Mahajan, Laura Culp, Lechao Xiao, Maxwell L. Bileschi, Noah Constant, Roman Novak, Rosanne Liu, Tris Warkentin, Yundi Qian, Yamini Bansal, Ethan Dyer, Behnam Neyshabur, Jascha Sohl-Dickstein, and Noah Fiedel. Beyond human data: Scaling self-training for problem-solving with language models. *Transactions on Machine Learning Research*, 2023.

Oswin So, Brian Karrer, Chuchu Fan, Ricky T. Q. Chen, and Guan-Horng Liu. Discrete adjoint matching. In *International Conference on Learning Representations (ICLR)*, 2026.

Jascha Sohl-Dickstein, Eric A. Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International Conference on Machine Learning (ICML)*, 2015.

Yang Song, Jascha Sohl-Dickstein, Diederik P. Kingma, Abhishek Kumar, Stefano Ermon, and Ben Poole. Score-based generative modeling through stochastic differential equations. In *International Conference on Learning Representations (ICLR)*, 2021.

Hannes Stark, Bowen Jing, Chenyu Wang, Gabriele Corso, Bonnie Berger, Regina Barzilay, and Tommi Jaakkola. Dirichlet flow matching with applications to DNA sequence design. In *International Conference on Machine Learning (ICML)*, 2024.

Xiaohang Tang, Rares Dolga, Sangwoong Yoon, and Ilija Bogunovic. wd1: Weighted policy optimization for reasoning in diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2026a.

Xiaohang Tang, Keyue Jiang, Che Liu, Qifang Zhao, Xiaoxiao Xu, Sangwoong Yoon, and Ilija Bogunovic. GDSD: Reinforcement learning as guided denoiser self-distillation for diffusion language models. *arXiv preprint arXiv:2605.29398*, 2026b.

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

Masatoshi Uehara, Yulai Zhao, Tommaso Biancalani, and Sergey Levine. Understanding reinforcement learning-based fine-tuning of diffusion models: A tutorial and review. *arXiv preprint arXiv:2407.13734*, 2024a.

Masatoshi Uehara, Yulai Zhao, Kevin Black, Ehsan Hajiramezanali, Gabriele Scalia, Nathaniel Lee Diamant, Alex M Tseng, Tommaso Biancalani, and Sergey Levine. Fine-tuning of continuous-time diffusion models as entropy-regularized control. *arXiv preprint arXiv:2402.15194*, 2024b.

Zhengyan Wan, Yidong Ouyang, Panwen Hu, and Qiang Sun. dFlowGRPO: Rate-aware policy optimization for discrete flow models. *arXiv preprint arXiv:2605.09291*, 2026.

Chenyu Wang, Masatoshi Uehara, Yichun He, Amy Wang, Tommaso Biancalani, Avantika Lal, Tommi Jaakkola, Sergey Levine, Hanchen Wang, and Aviv Regev. Fine-tuning discrete diffusion models via reward optimization with applications to DNA and protein design. In *International Conference on Learning Representations (ICLR)*, 2025.

Chenyu Wang, Paria Rashidinejad, DiJia Su, Song Jiang, Sid Wang, Siyan Zhao, Cai Zhou, Shannon Zejiang Shen, Feiyu Chen, Tommi Jaakkola, Yuandong Tian, and Bo Liu. SPG: Sandwiched policy gradient for masked diffusion language models. In *International Conference on Learning Representations (ICLR)*, 2026a.

Guanghan Wang, Gilad Turok, Yair Schiff, Marianne Arriola, and Volodymyr Kuleshov. d2: Improving reasoning in diffusion language models via trajectory likelihood estimation. In *International Conference on Machine Learning (ICML)*, 2026b.

Jiaqi Wang, Haoge Deng, Ting Pan, Yang Liu, Chengyuan Wang, Fan Zhang, Yonggang Qi, and Xinlong Wang. UDM-GRPO: Stable and efficient group relative policy optimization for uniform discrete diffusion models. In *International Conference on Machine Learning (ICML)*, 2026c.

Chenxing Wei, Jiazhen Kang, Hong Wang, Jianqing Zhang, Hao Jiang, Xiaolong Xu, Ningyuan Sun, Ying He, F. Richard Yu, Yao Shu, and Bo Jiang. LFPO: Likelihood-free policy optimization for masked diffusion models. *arXiv preprint arXiv:2603.01563*, 2026.

Dongyeop Woo, Minsu Kim, Minkyu Kim, Kiyoung Seong, and Sungsoo Ahn. Energy-based generator matching: A neural sampler for general state space. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Wei Xiong, Jiarui Yao, Yuhui Xu, Bo Pang, Lei Wang, Doyen Sahoo, Junnan Li, Nan Jiang, Tong Zhang, Caiming Xiong, and Hanze Dong. A minimalist approach to LLM reasoning: from rejection sampling to reinforce. *arXiv preprint arXiv:2504.11343*, 2025.

Yixian Xu, Yuanrui Zhang, Shengjie Luo, Liwei Wang, and Di He. Designing reinforcement learning for diffusion models: A unified path-space view. *arXiv preprint arXiv:2608.14430*, 2026.

Shuchen Xue, Chongjian Ge, Shilong Zhang, Yichen Li, and Zhi-Ming Ma. Advantage weighted matching: Aligning RL with pretraining in diffusion models. In *International Conference on Machine Learning (ICML)*, 2026.

Zeyue Xue, Jie Wu, Yu Gao, Fangyuan Kong, Lingting Zhu, Mengzhao Chen, Zhiheng Liu, Wei Liu, Qiushan Guo, Weilin Huang, and Ping Luo. DanceGRPO: Unleashing GRPO on visual generation. *arXiv preprint arXiv:2505.07818*, 2025.

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

Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, et al. DAPO: An open-source LLM reinforcement learning system at scale. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*, 2023.

Oussama Zekri and Nicolas Boullé. Fine-tuning discrete diffusion models with policy gradient methods. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

Zikun Zhang, Jiayuan Sheng, David D. Yao, and Wenpin Tang. A continuous-time reinforcement learning framework for fine-tuning discrete diffusion models. *arXiv preprint arXiv:2607.14522*, 2026.

Siyan Zhao, Devaansh Gupta, Qinqing Zheng, and Aditya Grover. d1: Scaling reasoning in diffusion large language models via reinforcement learning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025.

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. In *International Conference on Learning Representations (ICLR)*, 2026.

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. In *Proceedings of the 64th Annual Meeting of the Association for Computational Linguistics (ACL)*, 2026.

Xinyu Zhu, Mengzhou Xia, Zhepei Wei, Wei-Lin Chen, Danqi Chen, and Yu Meng. The surprising effectiveness of negative reinforcement in LLM reasoning. In *Advances in Neural Information Processing Systems (NeurIPS)*, 2025a.

Yuchen Zhu, Wei Guo, Jaemoo Choi, Petr Molodyk, Bo Yuan, Molei Tao, and Yongxin Chen. Enhancing reasoning for diffusion LLMs via distribution matching policy optimization. *arXiv preprint arXiv:2510.08233*, 2025b.

CONTENTS

1 Introduction **1**

2 Background: Diffusion and Flows via Generator Matching **3**

3 Tilted Generator Matching **5**

3.1 RL as Weighted Pretraining 5

3.2 TGM in Practice 5

3.3 TGM-Map: Tilting Flow Maps 6

4 Experiments **7**

4.1 Setup 7

4.2 Results 8

5 Discussion **9**

A Related Work **20**

B Generator Matching **22**

B.1 Conditional Coefficients Are Affine in the Target 22

B.2 Pretraining Losses of Common Models 22

B.3 Continuous-State Diffusion Language Models 23

B.4 RL Methods Across Models 23

C Proofs **26**

C.1 Proof of Proposition 1 26

C.2 Proof of Proposition 2 26

C.3 Control Variate 26

C.4 Invariance to Centering 27

C.5 Proof of Proposition 3 27

C.6 Relation to Tilt Matching and Discrete Tilt Matching 28

D Algorithm **29**

E Experimental Details **30**

E.1 Tasks 30

E.2 Models 30

E.3 RL methods and baselines 31

E.4 Sampling and NFE 32

E.5 Accuracy–NFE frontier 32

E.6 Rollout-NFE ablation 33

F Qualitative Samples 34

A Related Work

We first place TGM's central identity in its lineage, then review RL for diffusion and flow models by what each family needs from the model: a likelihood, a tractable reverse trajectory, or only a training loss. Table 4 summarizes which methods apply to which models.

RL as reward-weighted maximum likelihood. When RL is regularized by a KL divergence to a reference, its optimum is the reference tilted by the exponentiated reward (Korbak et al., 2022), and fitting that tilt is a weighted maximum-likelihood problem. This view underlies EM-based policy search (Dayan & Hinton, 1997; Peters & Schaal, 2007), advantage-weighted regression (Peng et al., 2019), and MPO, whose M-step fits the tilted policy by weighted maximum likelihood (Abdolmaleki et al., 2018); for sequence models, RAML trains against an exponentiated-reward target (Norouzi et al., 2016), and distributional approaches fit a tilted LM by importance-weighted cross-entropy (Khalifa et al., 2021). For LLMs, reward-filtered fine-tuning (Dong et al., 2023; Yuan et al., 2023; Singh et al., 2023) is a hard-threshold version, competitive with GRPO once corrected (Xiong et al., 2025), and recent work shows that Boltzmann-weighted fine-tuning on reference samples recovers the KL-regularized RLVR optimum exactly, with refreshed samples giving policy mirror descent (Shu et al., 2026). All of these fit a model whose likelihood, and hence maximum-likelihood loss, is tractable. TGM carries the identity to models trained by generator matching, whose pretraining loss is a Bregman regression onto a conditional target rather than a log-likelihood, and whose annealed increments play the role of the refreshed samples.

Likelihood-based RL for DLMs. GRPO and PPO (Shao et al., 2024; Schulman et al., 2017) need sequence likelihoods or their ratios, which DLMs lack: exact likelihoods require marginalizing over generation orders, and are tractable only for special samplers (Turok et al., 2026). Masked DLMs therefore substitute one-step or ELBO surrogates (Zhao et al., 2025; Gong et al., 2026; Zhu et al., 2026; Tang et al., 2026a; Wang et al., 2026a), whose ratio noise is a known source of instability; image diffusion models do the same with the denoising loss (Xue et al., 2026). The surrogate exists where a variational bound does, which is looser for uniform noise and absent for flow and flow-map models trained by regression.

Reverse-process RL. The dominant approach for image models treats the sampling trajectory as a Markov decision process and applies policy gradients to the Gaussian transitions of a stochastic sampler (Black et al., 2024; Fan et al., 2023; Liu et al., 2025a; Xue et al., 2025; McAllister et al., 2026); masked DLMs have trajectory-level counterparts (Wang et al., 2026b). For discrete models, several frameworks are derived for a general CTMC rather than for masking (Zekri & Boullé, 2025; So et al., 2026; Zhang et al., 2026; Wan et al., 2026), and UDM-GRPO targets uniform discrete diffusion (Wang et al., 2026c). Their experiments use masked models, image tokens, DNA, or a multimodal non-masked flow; none is demonstrated on a uniform-noise language model, and none covers continuous-state models. Reverse-process RL ties training to one stochastic sampler with tractable steps, stores trajectories, and must stochasticize flows, which excludes deterministic, higher-order and few-step samplers.

Forward-process RL and reward tilting. Under KL-regularized reward maximization, the optimal process tilts the clean-sample distribution and leaves the noising kernel unchanged (Bergmeister et al., 2026). Forward-process methods exploit this: they sample rollouts with any sampler, noise them as in pretraining, and regress the pretraining target with reward-dependent weights or targets (Zheng et al., 2026; Xue et al., 2026; Kveton et al., 2026; Fu et al., 2026b; Choi et al., 2026). Tilt matching derives such updates exactly by annealing the tilt for Gaussian flows (Potaptchik et al., 2026a), and discrete tilt matching does so for masked diffusion (Chen et al., 2026c); TGM recovers both as special cases (Appendix C.6). A parallel stochastic-optimal-control line reaches the tilt through adjoint and entropy-regularized formulations (Domingo-Enrich et al., 2024; Uehara et al., 2024b; Bergmeister et al., 2026), including for discrete diffusion (Wang et al., 2025; So et al., 2026). For masked DLMs, several likelihood-free methods match the reward-tilted posterior with cross-entropy or logit regression (Rector-Brooks et al., 2024; Zhu et al., 2025b; Tang et al., 2026b; Wei et al., 2026). Each of these is derived for one state space or one kernel. Concurrent work unifies reward fine-tuning within continuous diffusion and flow models: as one path-space principle (Xu et al., 2026), or as Newton iterations over conditional-matching minimizers (Li et al., 2026a). Energy-based generator matching (Woo et al., 2025) reweights the generator-matching loss to sam-

ple from an energy without data. TGM instead proves the tilt identity for any generator, discrete or continuous, as KL-regularized RL around a pretrained model.

RL for few-step generators and flow maps. RL for consistency models and flow maps has been developed for images and video (Oertell et al., 2024; Li et al., 2026b; Huang et al., 2026; Mammadov et al., 2026; Lee et al., 2026b), and flow maps make value estimation and reward guidance cheap (Holderrieth et al., 2026; Potaptchik et al., 2026b). Tilt matching notes that it applies to flow maps (Potaptchik et al., 2026a); TGM-Map makes this precise for two-time heads trained with cross-entropy (Proposition 3) and is, to our knowledge, the first RL method demonstrated on flow-map language models.

Other formulations. Differentiable-reward methods backpropagate reward gradients through the sampler (Clark et al., 2024; Prabhudesai et al., 2023), which verifiable rewards do not provide. Beyond KL, the prior can be enforced by optimal-transport or other divergences (Mammadov et al., 2026), and whether reward-guided samplers attain the tilt at all has been questioned (Dandapanthula & Boffi, 2026). For a broader survey of diffusion RL, see Uehara et al. (2024a).

B Generator Matching

This appendix supports Section 2. We show that the conditional coefficients of every model we consider are affine in its conditional target (Appendix B.1), list the choices and pretraining losses of common models (Appendix B.2), show that encoding tokens into a continuous space changes nothing (Appendix B.3), and mark which RL methods apply to which models (Appendix B.4). For general background on generators we refer to Holderrieth et al. (2025) and Lipman et al. (2024).

B.1 Conditional Coefficients Are Affine in the Target

Section 2 assumes that the conditional coefficients are affine in the conditional target, $u_{t|1}(z \mid x) = a_t(z) + B_t(z)F_{t|1}(z \mid x)$. Because the kernel factorizes, the conditional process moves each element independently, so the coefficients of element i depend on x only through x^i. Dots denote time derivatives.

Tokens. For token data, the assumption holds with the one-hot target $F_{t|1} = e_{x^i}$ for *any* kernel and process, because every function of a token is linear in its one-hot vector:

$$g(x^i, z) = \sum_{v \in \mathcal{V}} g(v, z)\, [e_{x^i}]_v. \tag{12}$$

Taking $g = u_{t|1}$ gives $a_t = 0$ and $B_t(z) = [\, u_{t|1}(z \mid v)\,]_{v \in \mathcal{V}}$, so the marginal target is the posterior over tokens, $F_t(z) = \pi(\cdot \mid z, i)$, and $u_t(z) = \sum_v \pi(v \mid z, i)\, u_{t|1}(z \mid v)$. The same holds for any other conditional target that is a function of the token, such as the time-reversal rates regressed by uniform-noise ELBOs (Appendix B.2). For example, the masked kernel is followed by the CTMC that unmasks element i at rate $\dot\beta_t/\alpha_t$ and jumps to its clean token, $u_{t|1}(v \mid z^i{=}\mathrm{m}, x^i) = \frac{\dot\beta_t}{\alpha_t}[e_{x^i}]_v$, and the uniform kernel by $u_{t|1}(v \mid z^i, x^i) = \frac{\dot\beta_t}{\alpha_t}\big([e_{x^i}]_v - \delta_{z^i}(v)\big)$ for $v \neq z^i$ (Gat et al., 2024; Campbell et al., 2024).

Gaussian kernels. For images, the target must be affine in $x^i \in \mathbb{R}^d$ itself, and for encoded tokens in $\mathcal{E}(x^i)$. For $\kappa_t(\cdot \mid x^i) = \mathcal{N}(\beta_t x^i, \alpha_t^2 I)$, every process that follows the marginal path solves

$$dz_t = \Big(b_t(z_t) + \tfrac{\epsilon_t}{2} s_t(z_t)\Big)dt + \sqrt{\epsilon_t}\, dW_t, \qquad z_0 \sim \mathcal{N}(0, I), \tag{13}$$

for a chosen diffusion schedule $\epsilon_t \geq 0$: $\epsilon_t \equiv 0$ is the probability-flow ODE and any $\epsilon_t > 0$ a diffusion. The velocity b_t and score $s_t = \nabla \log \rho_t$ are posterior averages of

$$b_{t|1}(z \mid x) = \frac{\dot\alpha_t}{\alpha_t} z^i + \Big(\dot\beta_t - \frac{\dot\alpha_t \beta_t}{\alpha_t}\Big) x^i, \qquad s_{t|1}(z \mid x) = -\frac{z^i - \beta_t x^i}{\alpha_t^2}, \tag{14}$$

which are both affine in x^i, so every drift in Equation (13) is affine in the clean-data target and hence in the noise and velocity targets, which are affine in it given z (Lipman et al., 2024). A trained network can therefore drive the ODE or any SDE of the same path: flow versus diffusion is a sampling choice. For encoded tokens, $x^i = \mathcal{E}(v)$ and the posterior mean of x^i is $\mathcal{E}(\pi(\cdot \mid z, i))$ with $\mathcal{E}$ extended linearly to distributions, so a single categorical head determines both fields.

B.2 Pretraining Losses of Common Models

Table 3 lists the choices behind common diffusion and flow models and the per-element loss inside Equation (5). By Proposition 1, the time distribution ν and weight λ_t do not move the minimizer, so models that share a kernel, process and target share it and differ only in how they weight errors across noise levels. Particular weights make Equation (5) a likelihood bound: for Gaussian noise, the clean-data MSE with $\lambda_t = \frac{1}{2}\dot{\mathrm{SNR}}_t$, $\mathrm{SNR}_t = \beta_t^2/\alpha_t^2$, up to endpoint terms (Kingma et al., 2021; Yang et al., 2026); for masked noise, the cross-entropy with $\lambda_t = \dot\beta_t/\alpha_t$ (Sahoo et al., 2024). For uniform noise, the ELBO instead regresses the time-reversal rates with the generalized KL divergence $\mathrm{gKL}(a, b) = \sum_{v \neq z^i} \big(a_v \log \frac{a_v}{b_v} - a_v + b_v\big)$ (Campbell et al., 2022; Sahoo et al., 2025). The cross-entropy losses on Gaussian and Dirichlet paths have no known ELBO. Our Duo runs fine-tune with the cross-entropy on the posterior instead of the gKL loss (Appendix E.2).

B.3 Continuous-State Diffusion Language Models

A continuous DLM generates in $\mathbb{R}^d$ or Δ^{K-1} and decodes to tokens, while the reward is defined on tokens. The results of Section 3 apply unchanged under two conditions, satisfied by every continuous model in Table 3 once its embeddings are frozen.

(C1) *Fixed, injective, per-token encoder.* $\mathcal{E} : \mathcal{V} \to \mathbb{R}^d$ acts on one token at a time, is not trained with θ, and maps distinct tokens to distinct points. Per-token makes $\mathcal{N}(\beta_t \mathcal{E}(v), \alpha_t^2 I)$ a kernel on tokens, so the path factorizes over positions and the head is a per-position posterior. Fixed keeps the conditional path, and hence ℓ_θ of Equation (7), independent of θ; models that train embeddings jointly during pretraining (Dieleman et al., 2022; Gulrajani & Hashimoto, 2023) freeze them before RL.

(C2) *Deterministic decoder.* $\mathcal{D}$ with $\mathcal{D} \circ \mathcal{E} = \mathrm{id}$, applied per position, such as nearest embedding or the argmax of the head at $t = 1$. It makes the reward a function of the generated element, $\mathcal{R} \circ \mathcal{D}$.

Lemma 4 (Tilting commutes with encoding)**.** *Under (C1)–(C2), let $\mathcal{E}_\# \rho_1$ be the distribution of $\mathcal{E}(x)$ for $x \sim \rho_1$. Then $\mathcal{E}_\#\big(\rho_1^{(\Gamma)}\big) = \big(\mathcal{E}_\# \rho_1\big)^{(\Gamma)}$, where the right-hand tilt uses the reward $\mathcal{R} \circ \mathcal{D}$.*

Proof. The weight $e^{\Gamma \mathcal{R}(x)} = e^{\Gamma(\mathcal{R} \circ \mathcal{D})(\mathcal{E}(x))}$ is a function of $\mathcal{E}(x)$, so reweighting and pushing forward commute, and the normalizers agree. □

The tilt problem on tokens and on encoded tokens is therefore the same problem, and Propositions 1 to 3 apply verbatim. One practical rule follows: if a sampler outputs an encoded element rather than a token, the clean sample entering Equation (9) is the decoded token sequence $\mathcal{D}(z_1)$, re-encoded, since z_1 need not lie in $\mathcal{E}(\mathcal{V})^L$.

B.4 RL Methods Across Models

Table 4 marks which RL methods apply to the models of Table 3. Likelihood-based methods need the sequence likelihood or a bound on it, which exists only for some kernels. Reverse-process methods need the transition likelihood of each sampler step, which ties them to stochastic samplers with tractable steps. Forward-process methods, including TGM, need only a training loss. A mark records whether a model satisfies the method's stated requirement, not whether the method would work well there.

Table 3: **Diffusion and flow models in generator matching.** Per element: the state space $\mathcal{S}$, noise distribution κ_0 and kernel κ_t; the process and the coefficients it is linear in; the conditional target $F_{t|1}$; and the loss $\lambda_t D_\psi$ inside Equation (5), with network output ϵ^θ, v^θ, x^θ, posterior $\pi^\theta(\cdot \mid z, i)$, concrete score s^θ, rates R^θ, or two-time head $\delta^\theta_{s,t}$. Gaussian kernels have signal β_t and noise α_t; discrete kernels have $\alpha_t = 1 - \beta_t$; Dirichlet has $\beta_t : 0 \to \infty$; $\epsilon = (z^i - \beta_t x^i)/\alpha_t$; $R_{t|1}$ are the time-reversal rates out of z^i. ‡Embeddings trained jointly during pretraining. *A flow map is not a generator; its diagonal is (Section 3.3).

Model	**State** $\mathcal{S}$	κ_0	**Kernel** $\kappa_t(\cdot \mid x^i)$	**Process** (u_t)	**Target** $F_{t\|1}$	**Loss** $\lambda_t D_\psi$
Continuous data						
DDPM (Ho et al., 2020)	$\mathbb{R}^d$	$\mathcal{N}(0, I)$	$\mathcal{N}(\beta_t x^i, \alpha_t^2 I)$	SDE (drift)	ϵ	$\|\epsilon^\theta - \epsilon\|^2$
Flow matching (Lipman et al., 2023)	$\mathbb{R}^d$	$\mathcal{N}(0, I)$	$\mathcal{N}(t\, x^i, (1-t)^2 I)$	ODE (velocity)	$(x^i - z^i)/(1-t)$	$\|v^\theta - F_{t\|1}\|^2$
VDM (Kingma et al., 2021)	$\mathbb{R}^d$	$\mathcal{N}(0, I)$	$\mathcal{N}(\beta_t x^i, \alpha_t^2 I)$	SDE (drift)	x^i	$\frac{1}{2}\mathrm{S\dot{N}R}_t \|x^\theta - x^i\|^2$
Discrete tokens						
MDLM, LLaDA (Sahoo et al., 2024; Nie et al., 2025)	$\mathcal{V} \cup \{\mathrm{m}\}$	δ_{e_m}	$\mathrm{Cat}(\beta_t x^i + \alpha_t e_\mathrm{m})$	CTMC (rates)	e_{x^i}	$-\frac{\dot\beta_t}{\alpha_t}\mathbf{1}[z^i{=}\mathrm{m}] \log \pi^\theta(x^i \mid z, i)$
SEDD (Lou et al., 2024)	$\mathcal{V} \cup \{\mathrm{m}\}$	δ_{e_m}	$\mathrm{Cat}(\beta_t x^i + \alpha_t e_\mathrm{m})$	CTMC (rates)	$r_y = \frac{\kappa_t(y\|x^i)}{\kappa_t(z^i\|x^i)}$	$\sum_{y \neq z^i} w_{t,y}\left(s^\theta_y - r_y \log s^\theta_y\right)$
Discrete FM (Gat et al., 2024)	$\mathcal{V}$ $(\cup\{\mathrm{m}\})$	Unif / δ_{e_m}	$\mathrm{Cat}(\beta_t x^i + \alpha_t \kappa_0)$	CTMC (rates)	e_{x^i}	$-\log \pi^\theta(x^i \mid z, i)$
DiffusionGemma (DiffusionGemma Team, 2026)	$\mathcal{V}$	Unif	$\mathrm{Cat}(\beta_t x^i + \frac{\alpha_t}{K}\mathbf{1})$	CTMC (rates)	e_{x^i}	$-\log \pi^\theta(x^i \mid z, i)$
UDLM, Duo (Schiff et al., 2025; Sahoo et al., 2025)	$\mathcal{V}$	Unif	$\mathrm{Cat}(\beta_t x^i + \frac{\alpha_t}{K}\mathbf{1})$	CTMC (rates)	$R_{t\|1}(\cdot \mid z^i, x^i)$	$\mathrm{gKL}\big(R_{t\|1}, R^\theta\big)$
Continuous representations of tokens						
Dirichlet FM (Stark et al., 2024)	Δ^{K-1}	$\mathrm{Dir}(\mathbf{1})$	$\mathrm{Dir}(\mathbf{1} + \beta_t x^i)$	ODE (velocity)	e_{x^i}	$-\log \pi^\theta(x^i \mid z, i)$
Plaid (Gulrajani & Hashimoto, 2023)	$\mathbb{R}^d$	$\mathcal{N}(0, I)$	$\mathcal{N}(\beta_t \mathcal{E} x^i, \alpha_t^2 I)^\ddagger$	SDE (drift)	$\mathcal{E} x^i$	$\frac{1}{2}\mathrm{S\dot{N}R}_t \|x^\theta - \mathcal{E} x^i\|^2$
CDCD (Dieleman et al., 2022)	$\mathbb{R}^d$	$\mathcal{N}(0, I)$	$\mathcal{N}(\beta_t \mathcal{E} x^i, \alpha_t^2 I)^\ddagger$	ODE (velocity)	e_{x^i}	$-\log \pi^\theta(x^i \mid z, i)$
LangFlow (Chen et al., 2026b)	$\mathbb{R}^d$	$\mathcal{N}(0, I)$	$\mathcal{N}(\beta_t \mathcal{E} x^i, \alpha_t^2 I)$	ODE (velocity)	e_{x^i}	$-\log \pi^\theta(x^i \mid z, i)$
FLM (Lee et al., 2026a)	$\mathbb{R}^K$	$\mathcal{N}(0, I)$	$\mathcal{N}(\beta_t x^i, \alpha_t^2 I)$	ODE (velocity)	e_{x^i}	$-\log \pi^\theta(x^i \mid z, i)$
FMLM* (Lee et al., 2026a)	$\mathbb{R}^K$	$\mathcal{N}(0, I)$	$\mathcal{N}(\beta_t x^i, \alpha_t^2 I)$	flow map $(\delta_{s,t})$	e_{x^i}, semigroup	$-\log \delta^\theta_{t,t}(x^i \mid z) + D_\mathrm{KL}\big(\bar\delta_{s,t} \,\|\, \delta^\theta_{s,t}\big)$

Table 4: **Post-training methods across diffusion and flow models** (models as in Table 3). RAFT is grouped separately because it is reward-filtered supervised fine-tuning rather than RL: it trains on the best rollout of each group and never pushes away from a poor one. ✓: demonstrated in the method's paper; (✓): the method's requirement holds but it was not demonstrated; ×: the requirement fails. Requirements: diffu-GRPO, a masked mean-field likelihood; SPG, a masked ELBO and EUBO; AWM, an ELBO surrogate for the likelihood; Flow-GRPO, Gaussian SDE transitions; RAM, a memoryless Gaussian SDE; DiffusionNFT, Gaussian velocity regression; TM, velocity regression; DTM, a masked CTMC; RAFT, a pretraining loss; TGM, a generator-matching or flow-map loss. †Only rollouts with negative raw reward.

	Likelihood / ELBO			**Reverse process**		**Forward process**			**SFT**	
Model	*diffu-GRPO*	*SPG*	*AWM*	*Flow-GRPO*	*RAM*	*DiffusionNFT*	*TM*	*DTM*	*RAFT*	***TGM (ours)***
Continuous data										
DDPM (Ho et al., 2020)	×	×	(✓)	(✓)	(✓)	(✓)	✓	×	✓	(✓)
Flow matching (Lipman et al., 2023)	×	×	✓	✓	✓	✓	(✓)	×	(✓)	(✓)
VDM (Kingma et al., 2021)	×	×	(✓)	(✓)	(✓)	(✓)	(✓)	×	(✓)	(✓)
Discrete tokens										
MDLM, LLaDA (Sahoo et al., 2024; Nie et al., 2025)	✓	✓	(✓)	×	×	×	×	✓	(✓)	✓
SEDD (Lou et al., 2024)	(✓)	(✓)	(✓)	×	×	×	×	(✓)	(✓)	(✓)
Discrete FM (Gat et al., 2024)	(✓)	(✓)	(✓)	×	×	×	×	(✓)	(✓)	(✓)
DiffusionGemma (DiffusionGemma Team, 2026)	×	×	×	×	×	×	×	(✓)	(✓)	(✓)
UDLM (Schiff et al., 2025)	×	×	(✓)	×	×	×	×	(✓)	(✓)	(✓)
Duo (Sahoo et al., 2025)	×	×	(✓)	×	×	×	×	(✓)	(✓)	✓
Continuous representations of tokens										
Dirichlet FM (Stark et al., 2024)	×	×	×	×	×	×	(✓)	×	(✓)	(✓)
Plaid (Gulrajani & Hashimoto, 2023)	×	×	(✓)	(✓)	(✓)	(✓)	(✓)	×	(✓)	(✓)
CDCD (Dieleman et al., 2022)	×	×	(✓)	(✓)	(✓)	(✓)	(✓)	×	(✓)	(✓)
LangFlow (Chen et al., 2026b)	×	×	(✓)	(✓)	(✓)	(✓)	(✓)	×	(✓)	(✓)
FLM (Lee et al., 2026a)	×	×	(✓)	(✓)	(✓)	(✓)	(✓)	×	(✓)	✓
FMLM (Lee et al., 2026a)	×	×	×	×	×	×	(✓)	×	(✓)	✓
Likelihood-free	×	×	×	×	✓	✓	✓	✓	✓	✓
Repels below-average rollouts	✓	✓	✓	✓	✓	✓	×†	×†	×	✓

C PROOFS

Throughout, $\mathbb{E}_{x|z}$ averages over the current posterior $\rho_{1|t}^{(\gamma)}(\cdot \mid z)$, $F_t^{(\gamma)}$ and $F_t^{(\gamma+h)}$ are the marginal targets Equation (4) of the current and next tilt, $w = e^{h\mathcal{R}}$, and $Z := \mathbb{E}_{x\sim\rho_1^{(\gamma)}}[w(x)]$. We use two facts. The first is the Bregman identity: for any random target Y and fixed b,

$$\mathbb{E}\big[D_\psi(Y \,\|\, b)\big] = D_\psi\big(\mathbb{E}[Y] \,\big\|\, b\big) + \mathbb{E}[\psi(Y)] - \psi\big(\mathbb{E}[Y]\big), \tag{15}$$

since $D_\psi(a\|b)$ is affine in a apart from $\psi(a)$, which does not depend on b; the right-hand side is uniquely minimized at $b = \mathbb{E}[Y]$ because ψ is strictly convex (Banerjee et al., 2005). The second is the form of the tilted posterior.

Lemma 5 (Tilted posterior). *For every t, $\rho_{1|t}^{(\gamma+h)}(x \mid z) = w(x)\,\rho_{1|t}^{(\gamma)}(x \mid z)/\mathbb{E}_{x|z}[w]$, and hence*

$$F_t^{(\gamma+h)}(z) = \frac{\mathbb{E}_{x|z}\big[w\, F_{t|1}\big]}{\mathbb{E}_{x|z}[w]}. \tag{16}$$

Proof. By Equation (6) and the fixed conditional path, the joint distributions of (x, z_t) satisfy $\rho_1^{(\gamma+h)}(x)\rho_{t|1}(z \mid x) = \frac{w(x)}{Z}\rho_1^{(\gamma)}(x)\rho_{t|1}(z \mid x)$. Summing (or integrating) over x gives $\rho_t^{(\gamma+h)}(z) = \mathbb{E}_{x|z}[w]\,\rho_t^{(\gamma)}(z)/Z$; dividing the joint by it gives the posterior, and averaging $F_{t|1}$ under it gives Equation (16). □

Equation (16) is the conditional Esscher transform (Esscher, 1932) that TM and DTM derive for velocities and unmasking posteriors (Potaptchik et al., 2026a; Chen et al., 2026c); TGM never evaluates it, since the regression Equation (9) computes it. Since $w > 0$, the lemma also shows that $\rho_t^{(\gamma)}$ and $\rho_t^{(\gamma+h)}$ have the same null sets.

C.1 PROOF OF PROPOSITION 1

Conditioning Equation (5) on (t, z_t) and applying Equation (15) with $Y = F_{t|1}(z_t \mid x)$, $x \sim \rho_{1|t}(\cdot \mid z_t)$, gives

$$\mathcal{L}_{\text{pre}}(\theta) = \mathbb{E}_{t,z_t}\Big[\lambda_t\, D_\psi\big(F_t \,\big\|\, F_t^\theta\big)\Big] + C, \tag{17}$$

with C independent of θ, so the conditional and marginal losses have the same gradients. Since $\lambda_t > 0$ and ν has full support, $\mathcal{L}_{\text{pre}}$ is minimized if and only if $F_t^\theta(z) = F_t(z)$ for almost every t and ρ_t-almost every z. Then $u_t^\theta = a_t + B_t F_t^\theta = u_t$, and by the generalized marginalization trick Equation (3), the process with coefficients u_t has marginals ρ_t, so $\rho_1^\theta = \rho_1$. □

C.2 PROOF OF PROPOSITION 2

Here $\rho_1^{(\gamma)}$ is the distribution of the rollouts, whichever sampler produced them. By Equation (8), $\mathcal{L}_{\text{TGM}}^{(\gamma+h)}(\theta) = Z \cdot \mathcal{L}_{\text{pre}}^{(\gamma+h)}(\theta)$ for every θ, where $\mathcal{L}_{\text{pre}}^{(\gamma+h)}$ is the pretraining loss Equation (5) with ρ_1 replaced by $\rho_1^{(\gamma+h)}$. Since $Z > 0$ does not depend on θ, the two losses have the same minimizers. By Proposition 1 applied to $\rho_1^{(\gamma+h)}$, the unique minimizer is $F_t^{(\gamma+h)}$, which generates $\rho_1^{(\gamma+h)}$. The minimizer is identified for $\rho_t^{(\gamma+h)}$-almost every z, and by Proposition 5 these are the same states as for $\rho_t^{(\gamma)}$, which the noised rollouts cover. □

C.3 CONTROL VARIATE

Proposition 6 (Control variate). *If $F_t^{\theta_{\text{old}}} = F_t^{(\gamma)}$, then for every θ, with $x \sim \rho_1^{(\gamma)}$,*

$$\mathcal{L}_{\text{TGM}}^{(\gamma+h)}(\theta) = \mathbb{E}_{t,x,z_t}\Big[\lambda_t\Big((w-1)\, D_\psi\big(F_{t|1} \,\big\|\, F_t^\theta\big) + D_\psi\big(F_t^{\theta_{\text{old}}} \,\big\|\, F_t^\theta\big)\Big)\Big] + C, \tag{18}$$

with C independent of θ. The two sides have the same minimizers and the same gradients.

Proof. Write $w = 1 + (w-1)$ in Equation (9). By Equation (15) applied at each (t, z_t), the reward-free term $\mathbb{E}[\lambda_t D_\psi(F_{t|1} \| F_t^\theta)]$ equals $\mathbb{E}[\lambda_t D_\psi(F_t^{(\gamma)} \| F_t^\theta)]$ up to a constant, and $F_t^{(\gamma)} = F_t^{\theta_{\text{old}}}$ by assumption. □

The minibatch loss Equation (10) with $\eta = 0$ is the sample average of Equation (18). At each z, the θ-dependent part of the right-hand side of Equation (18) is $\mathbb{E}_{x|z}[w]\, D_\psi\big(F_t^{(\gamma+h)} \,\|\, F_t^\theta\big)$ by Equations (15) and (16), whose target lies in the domain of ψ. The loss is therefore well-posed in expectation even though individual coefficients $w - 1$ are negative.

Inexact current model. If $F_t^{\theta_{\text{old}}} \neq F_t^{(\gamma)}$, for instance when rollouts come from a few-step sampler, the same computation with Equation (16) shows that the minimizer of the right-hand side of Equation (18) at z is

$$m(z) = \frac{F_t^{\theta_{\text{old}}} + \mathbb{E}_{x|z}\big[(w-1)F_{t|1}\big]}{\mathbb{E}_{x|z}[w]} = F_t^{(\gamma+h)} + \frac{F_t^{\theta_{\text{old}}} - F_t^{(\gamma)}}{\mathbb{E}_{x|z}[w]}, \tag{19}$$

when $m(z)$ lies in the domain of ψ: the next target plus the current model's error, scaled by $1/\mathbb{E}_{x|z}[w] \approx 1$ for small h. Without the control variate, Equation (9) has no such bias.

C.4 Invariance to Centering

Fix a prompt c and treat the batch standard deviation σ as a constant; we do not clip advantages. Since $\mathcal{R}_{c,g} - \bar{\mathcal{R}}_c = \frac{G-1}{G}\big(\mathcal{R}_{c,g} - \bar{\mathcal{R}}_{c,-g}\big)$, where $\bar{\mathcal{R}}_{c,-g}$ is the mean reward of the other rollouts of prompt c, the tilt weight factors as $w_{c,g} = w'_{c,g} k_{c,g}$ with $w'_{c,g} = e^{h' \mathcal{R}_{c,g}}$, $k_{c,g} = e^{-h' \bar{\mathcal{R}}_{c,-g}}$ and $h' = \frac{G-1}{G}\frac{h}{\sigma}$. The rollouts of prompt c are independent given c, so $k_{c,g}$ is independent of the pair (x, z_t) of rollout g, and $\bar{k}_c := \mathbb{E}[k_{c,g} \mid c] > 0$ does not depend on g. Writing $D_1 = \lambda_t D_\psi(F_{t|1} \| F_t^\theta)$ and $D_{\text{old}} = \lambda_t D_\psi(F_t^{\theta_{\text{old}}} \| F_t^\theta)$, the expected loss of rollout g in Equation (10) is

$$\mathbb{E}\big[D_{\text{old}} + (w'_{c,g} k_{c,g} - 1) D_1 \,\big|\, c\big] = \bar{k}_c\, \mathbb{E}\big[w'_{c,g} D_1 \,\big|\, c\big] + \mathbb{E}\big[D_{\text{old}} - D_1 \,\big|\, c\big]. \tag{20}$$

The last term does not depend on θ by Proposition 6, and the first is $\bar{k}_c$ times the TGM loss Equation (9) of prompt c with increment h'. Since F_t^θ is a free function of (z_t, c), the factor $\bar{k}_c$ does not move the minimizer at any c, which is therefore the tilt of the current model by $h'\mathcal{R}$. Centering thus rescales the increment but not the target; clipping the advantage would replace the exponential tilt by a reshaped reward. □

C.5 Proof of Proposition 3

We work at the population level with a network that can represent any two-time head. Following Lee et al. (2026a), $X_{s,t}^\theta$ is an affine function of the two-time head $\delta_{s,t}^\theta$, whose diagonal $\delta_{t,t}^\theta$ is a categorical head driving the Gaussian flow with velocity $u_t^\theta = a_t + B_t\, \delta_{t,t}^\theta$ (Appendix B.1), and $\delta_{s,t}^\theta$ is an average of such heads along the trajectory from s to t.

Diagonal. The semigroup loss $\mathcal{L}_{\text{sg}}$ trains only $\delta_{s,t}$ with $s < t$; its teacher is stop-gradient. The diagonal therefore receives gradient only from $\mathcal{L}_{\text{TGM}}^{(\gamma+h)}$, and at a critical point it minimizes this loss, since $\mathcal{L}_{\text{TGM}}^{(\gamma+h)}$ is convex in the head by Equation (15). By Proposition 2, $\delta_{t,t}^\theta = F_t^{(\gamma+h)}$.

Off-diagonal. Given the diagonal, the unique critical point of the KL semigroup loss is the two-time head of the flow driven by the diagonal, provided the times (s, u, t) and the states z_s are drawn with full support (Lee et al., 2026a, semigroup proposition). Gaussian noise of any rollout has full support on $\mathbb{R}^K$ for $s < 1$, so evaluating $\mathcal{L}_{\text{sg}}$ on noised rollouts satisfies the condition. Hence $X_{s,t}^\theta$ is the flow map of the ODE with velocity $u_t^{(\gamma+h)} = a_t + B_t F_t^{(\gamma+h)}$.

Generated distribution. By Proposition 1 applied to $\rho_1^{(\gamma+h)}$, this ODE carries ρ_0 to $\rho_1^{(\gamma+h)}$, and so does its flow map $X_{0,1}^\theta$. □

C.6 Relation to Tilt Matching and Discrete Tilt Matching

TM and DTM characterize the next target by Equation (16) and regress onto targets with the right weighted mean. Any $T(x, z_t)$ with $\mathbb{E}_{x|z}[wT] = \mathbb{E}_{x|z}[wF_{t|1}]$ can replace $F_{t|1}$ in Equation (9), since by Equation (15) the θ-dependent part of $\mathbb{E}_{x|z}[wD_\psi(T\|b)]$ depends on T only through $\mathbb{E}_{x|z}[wT]$. If $F_t^{\theta_{\text{old}}} = F_t^{(\gamma)}$, this holds for $T_\zeta = F_{t|1} + \frac{\zeta(z_t)}{w}\big(F_t^{\theta_{\text{old}}} - F_{t|1}\big)$ with any ζ. With velocity targets and MSE, $\zeta \equiv 0$ is TM's weighted flow matching and $\zeta \equiv 1$ its implicit tilt matching (Potaptchik et al., 2026a); with one-hot targets, cross-entropy, and the masked kernel, $\zeta \equiv c$ is DTM's c-DTM objective (Chen et al., 2026c). Equation (9) is $\zeta \equiv 0$ and Proposition 6 is $\zeta \equiv 1$.

D ALGORITHM

Algorithm 2 combines the choices of Section 3.2. Each increment freezes a snapshot, generates and scores G rollouts for each of B prompts, turns their advantages into tilt weights, and takes μ gradient steps on minibatches drawn from the resulting buffer and noised afresh. The optional η term anchors to the base model; we do not use it in any experiment. For TGM-Map, add $\mathcal{L}_{\mathrm{sg}}$ to the loss on line 9.

Algorithm 2 Tilted Generator Matching (practical form)

Require: model F_t^θ, prompts $\mathcal{C}$, reward $\mathcal{R}$, increment h, target tilt Γ, path $\rho_{t|1}$, target $F_{t|1}$, divergence D_ψ, weight λ, time distribution ν, prompts per batch B, rollouts per prompt G, reuse steps μ, base-anchor weight η (optional)

1: **for** $\gamma = 0, h, 2h, \dots, \Gamma - h$ **do** ▷ one tilt increment
2: $F_t^{\theta_{\mathrm{old}}} \leftarrow F_t^\theta$ ▷ freeze the current model
3: draw $c_1, \dots, c_B \sim \mathcal{C}$ and G rollouts $x \sim \rho_1^{(\gamma)}(\cdot \mid c)$ each, using $F_t^{\theta_{\mathrm{old}}}$ ▷ any sampler
4: $A_{c,g} \leftarrow (\mathcal{R}_{c,g} - \bar{\mathcal{R}}_c)/\sigma$; $w_{c,g} \leftarrow e^{hA_{c,g}}$ ▷ tilt weights
5: $\mathcal{B} \leftarrow \{(c, x, w_{c,g})\}_{c,g}$ ▷ rollout buffer
6: **for** $j = 1, \dots, \mu$ **do** ▷ sample reuse
7: draw a minibatch of (c, x, w) from $\mathcal{B}$
8: $t \sim \nu$; $z_t \sim \rho_{t|1}(\cdot \mid x)$ ▷ fresh noise
9: $\mathcal{L} \leftarrow \lambda_t \big[(w-1) D_\psi(F_{t|1} \| F_t^\theta) + D_\psi(F_t^{\theta_{\mathrm{old}}} \| F_t^\theta) + \eta\, D_\psi(F_t^{\theta_{\mathrm{base}}} \| F_t^\theta)\big]$, averaged ▷ Equation (10)
10: $\theta \leftarrow$ update θ with $\nabla_\theta \mathcal{L}$

E Experimental Details

E.1 Tasks

Throughout, y is a sampled completion and $\mathcal{R}(y)$ its reward during RL.

Sudoku (following S-FLM (Deschenaux & Gulcehre, 2026)). *Data.* 9×9 puzzles with unique solutions in three tiers by number of given cells: hard (30 of 81), extreme (25) and evil (20). S-FLM's tiers give 40 / 35 / 30 cells; these are too easy for our base models, so we shift to harder tiers. RL trains one model per tier on 48,000 puzzles; we evaluate on 2,000 held-out puzzles per tier. *Format.* The canvas is 180 tokens laid out as `[BOS]` + puzzle + `[BOS]` + solution. The puzzle is written row by row with blanks as `0` and a separator token between rows, giving 81 digits and 8 separators (89 tokens); a second `BOS` marks the puzzle–solution boundary, so the prompt is 91 tokens and the model generates the remaining 89 in the same format. The prompt is never noised: its positions are held clean and re-pinned at every sampler step. *Extraction.* We keep the digit characters of y and pad or truncate to 81 cells, giving $\hat{g}$; there is no legality check or format term. *Reward.* With solution g and blank cells $\mathcal{B}$, $\mathcal{R}(y) = |\mathcal{B}|^{-1} \sum_{i \in \mathcal{B}} \mathbb{1}[\hat{g}_i = g_i] \in [0, 1]$, maximized only by the unique solution. *Metric.* Exact solve rate: the fraction of test puzzles with $\hat{g} = g$.

GSM8K (following S-FLM (Deschenaux & Gulcehre, 2026)). *Data.* Base models are trained on TinyGSM (Liu et al., 2023), which writes each solution as a Python function `simple_math_problem()` returning the answer. RL uses the 7,473 GSM8K (Cobbe et al., 2021) training questions as prompts only (no reference programs); we evaluate zero-shot on the 1,319 test problems. *Format.* `[BOS]` + question + the separator, tokenized with `SmolLM-135M` (Ben Allal et al., 2025), with no system prompt or few-shot examples. Following S-FLM (Deschenaux & Gulcehre, 2026), the separator is the literal two-character string `\n` (a backslash followed by `n`, two tokens), not a newline; this is the string the TinyGSM checkpoints were pretrained with. The canvas has 512 tokens, so $512 - |\text{prompt}|$ positions are generated. *Extraction.* Following posterior refinement (Agarwal et al., 2026), we take the text from the first `def` to the first end-of-text token, keep its longest compiling prefix, and execute it in a restricted namespace with a 5 s timeout. The gold answer a is the `####` field. *Reward.* $\mathcal{R}(y) = \mathbb{1}[\text{program returns } a]$, exact for integers and within 10^{-3} otherwise; parse errors, exceptions and timeouts score 0. No format reward or partial credit. *Metric.* Pass@1, one program per problem.

Sentiment. *Format.* Unconditional: the prompt is a single `BOS`, and the model generates the rest of a 1,024-token OpenWebText canvas (Gokaslan & Cohen, 2019). We score the first 128 generated tokens. *Reward.* The logit margin $\ell_{\text{pos}}(y) - \ell_{\text{neg}}(y)$ of the three-way classifier `cardiffnlp/twitter-roberta-base-sentiment-latest` (Loureiro et al., 2022), min–max scaled within each rollout group before entering the tilt weight Equation (10). *Metrics.* A policy can raise a fixed classifier's score without becoming more positive (Gao et al., 2023), so we report the mean positive-class probability under a held-out classifier, `distilbert-base-uncased-finetuned-sst-2-english` (Sanh et al., 2019), which is used neither in training nor in checkpoint selection. It is measured on 256 samples from a single checkpoint per arm.

E.2 Models

Diffusion models use a DiT backbone (Peebles & Xie, 2023) with rotary embeddings. AR references use the same backbone with causal attention and no time conditioning, and are slightly smaller.

- **Sudoku.** 8 blocks, hidden size 512 (28.6M parameters; AR 25.2M), 180-token canvas, 20k steps. The three difficulty tiers share every pretraining setting and differ only in the puzzles they are trained on.
- **GSM8K.** 12 blocks, hidden size 768 (168M; AR 160.5M), trained on TinyGSM for 250k steps, `SmolLM-135M` tokenizer, sequences padded to 512.
- **Sentiment.** 12 blocks, hidden size 768 (169.6M; AR 162.2M), released checkpoints trained on OpenWebText for 1M steps, GPT-2 tokenizer, context 1,024.
- **Duo.** Duo's released head parameterizes the cavity posterior, paired with its bridge sampler (Gourevitch et al., 2026); we convert it to the posterior in closed form and fine-tune with the cross-entropy on the posterior (Appendix B.2).

- **FMLM+.** The flow map of posterior refinement (Agarwal et al., 2026), with the task's backbone and the task's pretraining budget. On GSM8K it is initialized from the TinyGSM MDLM and trained with the flow-map objective. On Sudoku it is trained from scratch on the puzzles with the same objective: no initialization from another model and no distillation.

Table 5: Pretraining settings. All models use AdamW ($\beta = (0.9, 0.999)$, $\epsilon = 10^{-8}$, weight decay 0), gradient clipping at 1.0, an EMA of 0.9999, and a constant learning rate after warmup. Every value is read from the checkpoint's own recorded training configuration.

Task	Recipe	Steps	Batch	LR	Warmup	Precision
Sudoku	S-FLM (Deschenaux & Gulcehre, 2026)	20k	256	3×10^{-4}	2.5k	bf16
GSM8K	S-FLM (Deschenaux & Gulcehre, 2026)	250k	512	3×10^{-4}	2.5k	bf16
Sentiment	MDLM (Sahoo et al., 2024)	1M	512	3×10^{-4}	2.5k	bf16
FMLM+	PR (Agarwal et al., 2026)	as its task's row above				

E.3 RL METHODS AND BASELINES

Matched protocol. Within a task all methods share the prompts, the rollouts per prompt G, the optimizer batch, the number of optimizer steps per phase and the evaluation sampler (Table 6). Learning rates are shared across methods and set per task: 1×10^{-5} on Sudoku and sentiment, 3×10^{-5} on GSM8K. They were not tuned per method; a learning-rate probe for the GRPO family over $\{1, 3\} \times 10^{-5}$ (AR and MDLM, GSM8K) and $\{1, 3\} \times 10^{-5}$, $\{1, 3\} \times 10^{-4}$ (FLM, Sudoku) found the larger rates strictly worse, so the shared value was kept. Three arms depart from the matched budget and we state them rather than smooth them over: AR + RAFT on Sudoku, FLM + TGM/RAFT on Sudoku (retrained at the published deterministic sampler, Appendix E.4) and FMLM+ + TGM-Map on Sudoku each take 100 optimizer steps per phase instead of 40.

TGM / TGM-Map. The tilt exponentiates the group-centred reward divided by the batch standard deviation, $\tilde{A} = (r - \bar{r}_g)/(\hat{\sigma} + 10^{-4})$, in the row form with gauge $c = 1$, which makes the per-phase step h task-independent and caps the head KL at $h^2/2$ nats per phase. The reference policy is a hard snapshot of the weights at the start of each phase, and the control variate is on for every TGM arm; there is no anchor to the pretrained weights ($\beta_0 = 0$). The step and dose are $h = 1$, $A = 30$ (Sudoku) and $h = 1$, $A = 60$ (GSM8K) for MDLM and Duo; $h = 1$, $A = 50$ (Sudoku) and $h = 1$, $A = 20$ (GSM8K) for FLM; $h = 0.25$, $A = 5$ on sentiment; and $h = 4$, $A = 120$ (Sudoku), $h = 2$, $A = 200$ (GSM8K) for TGM-Map, for $\mathrm{round}(A/h)$ phases. Times are drawn by a stratified sampler, i.i.d. on the GSM8K flow-map arms. Each phase's rollout buffer is reused μ times without replacement: $\mu = 2.5$ on Sudoku, 5 on GSM8K, 10 on sentiment. The optimizer is AdamW ($\beta = (0.9, 0.99)$, $\epsilon = 10^{-8}$, no weight decay, gradient clip 1.0) with a per-phase cycle of 5% linear warmup and linear decay to zero; the optimizer state is not reset between phases except on the GSM8K flow-map arms. The loss is reduced per token, except on MDLM where it is reduced per masked state.

GSM8K: two TGM-Map arms. The FMLM+ GSM8K number in Table 1 and the GSM8K panel of Figure 3 come from two different TGM-Map runs, and we report each where it was measured rather than carrying one across. They share the objective, tilt schedule and decoder, and differ in two knobs: the table's arm disables student dropout and takes 40 gradient steps per tilt phase, reaching 37.6% at NFE 128; the figure's arm enables dropout and takes 100 steps per phase, reaching 46.2%. Dropout alone does not account for the gap, since the step budget differs as well.

RAFT (Dong et al., 2023). Fine-tunes on the highest-reward rollout of each group with the family's pretraining loss (next-token cross-entropy for AR) at weight one, with no snapshot term. On Sudoku and GSM8K we keep the single argmax rollout per group; ties are broken uniformly at random from a seeded generator, so the choice is replayable. A group is used only if its maximum reward is positive: on GSM8K's binary reward a group with no correct rollout contributes nothing, while on Sudoku's dense reward a partially correct group is kept, which is the fair analogue of TGM seeing the same dense reward. Zero-variance groups are *not* removed: a group whose G rollouts all score alike still contributes its one uniformly chosen rollout when that score is positive. On sentiment, which is unconditional, we instead keep the top $1/G$ fraction of the batch by reward.

GRPO (Shao et al., 2024). For AR: exact token log-likelihoods, group-normalized advantages, $\mu = 2.5$ (Sudoku) / 5 (GSM8K) inner passes per rollout batch, clipping $\epsilon = 0.2$, per-sequence reduction, and a KL penalty to the frozen pretrained weights with $\beta = 0.04$ (the k_3 estimator), applied at the same value on every GRPO-family arm for parity.

diffu-GRPO (Zhao et al., 2025). For MDLM: its one-step likelihood estimator, with the completion fully masked and each prompt token masked i.i.d. with probability 0.15, one forward per update; the mask is shared by the current, snapshot and reference forwards, so the ratio is exactly one at the start of an update. $\mu = 2.5$ (Sudoku) / 5 (GSM8K), $\epsilon = 0.5$, advantages centred but not scaled, token-mean reduction, $\beta = 0.04$.

Flow-GRPO (Liu et al., 2025a). Developed for image flows; we adapt it to FLM, to our knowledge its first use on text. Rollouts replace the deterministic Euler sampler with the marginal-preserving SDE at noise level $\gamma = 1$ (full re-noise at every step, at unchanged NFE), so each step is a Gaussian with tractable likelihood. The final step is a deterministic argmax readout, leaving 88 stochastic transitions of the 89 Sudoku steps (127 of 128 on GSM8K); the loss uses one of them per trajectory, drawn uniformly, which is the FLOP-matched setting. Advantages are centred, divided by the batch standard deviation and clipped to ± 5; zero-variance groups are dropped; $\epsilon = 10^{-4}$ applies to the per-dimension mean log-density, and $\beta = 0.04$. Evaluation uses the deterministic sampler, as for every FLM row.

Table 6: RL hyperparameters of the main-results arms (Table 1), shared by every method within a task. Gradient updates are the full budget, $\mathrm{round}(A/h)$ phases of 40 optimizer steps; the reported checkpoint is selected over phases. μ is the number of passes over each phase's rollout buffer.

Task	G	Prompts / phase	Gradient updates	μ	LR	Method-specific
Sudoku	16	256	1,200 (30×40)	2.5	1×10^{-5}	ϵ: 0.2 / 0.5 / 10^{-4}, β=0.04
GSM8K	32	128	2,400 (60×40)	5	3×10^{-5}	ϵ: 0.2 / 0.5 / 10^{-4}, β=0.04
Sentiment	64	8	800 (20×40)	10	1×10^{-5}	ϵ: 0.2 / 0.5 / 10^{-4}, β=0.04

E.4 Sampling and NFE

Samplers. Each model uses its release's default, unchanged across RL methods and NFE: ancestral sampling at $T = 1$ for MDLM and Duo; Euler at $T = 1$, $\gamma = 0$ for FLM; greedy ($T = 0$) posterior refinement (Agarwal et al., 2026) with commit threshold $p = 0.999$ for FMLM+; temperature $T = 1$ with a KV cache for AR. The AR rows are therefore sampled rather than greedy: greedy decoding makes a group's G rollouts identical and leaves the advantage with no variance, so RL cannot run, and we evaluate with the same decoder the rollouts use.

NFE. One NFE is one network forward pass. Diffusion models denoise the whole canvas at a fixed NFE: 89 on Sudoku, 128 on GSM8K and 1023 on sentiment. For FMLM+, NFE is refinement rounds × flow steps per round; on both tasks we evaluate it on, it takes one flow step per round, so its NFE equals its round count. AR performs one pass per generated token: 89 on Sudoku, the full 1,024-token canvas on sentiment, and on GSM8K a variable number that stops at end-of-text (≈159 on average for the base, at most $512 - |\text{prompt}|$). S-FLM (Deschenaux & Gulcehre, 2026) reports the 512-token canvas for AR on GSM8K, which is an upper bound, not the number of passes. NFE compares cost only within a model class: per-pass cost differs across classes (e.g. AR uses a KV cache).

Comparing across families. Absolute accuracy reflects the decoder as much as the model: on GSM8K, FMLM+ (greedy posterior refinement) reaches about 31% where MDLM (ancestral) reaches 12.9%. Compare each base only with its own RL arms.

E.5 Accuracy–NFE frontier

For Figure 3 we evaluate one TGM-Map checkpoint per task and the base FMLM+ at every NFE on the grid shown, one sample per problem. The horizontal arrow marks where RL first reaches the base's best accuracy over the grid, interpolated in $\log_2$ NFE; the vertical arrow is the gain at the largest NFE. The GSM8K arm uses a different TGM-Map configuration from the Sudoku arms

(Appendix E.3). We omit sentiment, for two reasons. FMLM+ is not evaluated on sentiment at all, so the flow-map frontier has no sentiment arm to plot. More substantively, sentiment moves the opposite way from task accuracy as the budget is cut: on the masked arms for which we do have a two-budget comparison, the RL gain in held-out positive probability *grows* with NFE rather than shrinking, so a sentiment panel drawn in the same grammar would look like the others while telling the opposite story.

E.6 ROLLOUT-NFE ABLATION

Table 2 predates the main experiments and differs from them in four ways: (i) $G = 8$ rollouts per prompt (not 16) and an optimizer batch of 96 prompts (not 16); (ii) 640–800 prompts per phase (not 128), refreshed every 50 steps, a mechanism since removed; (iii) the pre-refactor fixed-length canvas instead of ragged spans above a floor; and (iv) a 512-prompt in-loop evaluation at NFE 32, not the full 1,319-problem test at NFE 128. The tilt is unchanged: $h = 2$, $A = 200$, learning rate 3×10^{-5}, summed semigroup loss, control variate on, and the same decoder ($p = 0.999$, one flow step, fresh noise).

Matched budget. The arms peaked between 900 and 7,800 optimizer steps, so their best checkpoints are not comparable. We report every arm at 800 steps, the largest budget the shortest arm reached. The table therefore measures learning *per optimizer step*, not accuracy at convergence: at its own peak (7,800 steps), the NFE-32 arm reaches 50.8%. The base is the shared pre-RL reading of the four 800-prompt arms; the NFE-32 arm drew from a 640-prompt pool and read 30.3%.

Timing. Samples/s is rollout throughput on one H100 NVL in bf16 on GSM8K training prompts, 32 sequences per device, after warm-up and with device synchronization, averaged over 3 runs.

F Qualitative Samples

The flow map of Section 3 is trained so that a handful of network calls suffice, and Figure 3 reports what that buys in aggregate. Figure 4 shows what it looks like on a single problem: an FMLM+ flow map, refined with TGM-Map, writing a correct GSM8K program in four network calls. The prompt is the problem text and nothing else (`[BOS]` + question + `\n`, no instruction preamble and no few-shot examples; Appendix E.1), so the docstring the model writes is that prompt copied back out. The interesting behaviour is not that the program is right but *where* the four calls go. The first call freezes 87 of the 112 program cells at once — signature and docstring, which the model can copy almost deterministically from the prompt — leaving the four arithmetic lines unresolved. The remaining three calls commit 21, 2 and 2 cells: they spend themselves on those four lines, and in particular on a coordination problem that a left-to-right decoder never faces. The name bound on the first body line and the name read on the second are separate canvas cells, committed out of order, and the sampler has to make them agree. Round 2 commits `= 16 - 3 - 4` around two digits that had already frozen in round 1, round 3 commits the `money` cells, and round 4 finally settles the variable name in both places at once. The name wanders while it is uncommitted — round 1's guess is already `eggs`, rounds 2 and 3 show `remaining` in both positions, and round 4 commits `eggs` again — which costs nothing, because an uncommitted cell is never a decision: the sampler discards and redraws all of them at every call. Reading a panel: `eggs` froze on this call, `money` froze earlier, and `result` is still free and is redrawn at the next call; a shaded blank is a whitespace cell that froze. Grey cells are not decisions — the sampler re-noises and redraws *every* uncommitted cell at *every* call — so grey text is only a current argmax and may change wholesale between panels (compare the body lines of rounds 1 and 2).

Prompt — a GSM8K test problem gold answer: 18

Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?

Round 1 of 4 **+87** frozen 87 / 112 program cells 417 / 443 canvas

```
def simple_math_problem() -> int:
    '''Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?'''

    eggs =  = 1 3 - 4 -4
    money = eggs_left
    result = money
    return result<|endoftext|>
```

Round 2 of 4 **+21** frozen 108 / 112 program cells 439 / 443 canvas

```
def simple_math_problem() -> int:
    '''Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?'''

    remaining = 16 - 3 - 4
    money = remaining * 2
    result = money
    return result<|endoftext|>
```

Round 3 of 4 **+2** frozen 110 / 112 program cells 441 / 443 canvas

```
def simple_math_problem() -> int:
    '''Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?'''

    remaining = 16 - 3 - 4
    money = remaining * 2
    result = money
    return result<|endoftext|>
```

Round 4 of 4 **+2** frozen 112 / 112 program cells 443 / 443 canvas

```
def simple_math_problem() -> int:
    '''Janet's ducks lay 16 eggs per day. She eats three for breakfast every morning and bakes muffins for her friends every day with four. She sells the remainder at the farmers' market daily for $2 per fresh duck egg. How much in dollars does she make every day at the farmers' market?'''

    eggs = 16 - 3 - 4
    money = eggs * 2
    result = money
    return result<|endoftext|>
```

Figure 4: **One GSM8K problem, solved in 4 NFE by an FMLM+ flow map.** Each panel is one round of the posterior-refinement sampler (Agarwal et al., 2026): the network is called once, every cell whose posterior clears the commit threshold freezes, and frozen cells are never revisited. Counters give cells frozen on that call, the running total in the 112-cell program window, and the total over the 443-cell span (`gen_len`). The top box is the *entire* prompt (Appendix E.1); the docstring restates it verbatim and freezes entirely on call 1, and italic marks prose from program. Everything else is as decoded; the program returns 18. *Provenance:* a different checkpoint and decoder from any reported number ($p = 0.9$ at $T = 0$, against $p = 0.999$ throughout; Appendix E.4), so it is illustrative, not a sample from a reported arm.