GENERATIVE MODELING WITH PROBABILITY-PRESERVING DYNAMICS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern generative modeling usually produces samples by transporting a simple noise prior to the data distribution. We study a complementary setting: once a useful state already exists, can a pretrained flow be reused as a stochastic transition rather than restarted from noise? We construct two reusable kernels directly from pretrained flow checkpoints, without retraining: a denoiser-based Metropolis kernel (dMALA), whose ideal target is the Gaussian-smoothed law p_σ, and a predictor–corrector kernel whose ideal target is p. We also characterize how one-step model and numerical errors accumulate under a local contraction assumption. On ImageNet-256, dMALA repairs deliberately weak generated states from FID 10.70 to 3.19 while retaining high similarity to the input (DINO 0.80). The same dynamics also support controlled variation: as the chain length increases, samples move progressively farther from their inputs, providing a controllable transition from local variation to broader exploration. Experiments on a synthetic Swiss-roll target, ImageNet-256, and Oxford Flowers-102 show that these dynamics can reuse existing flow models when a useful initial state is already available.

1 INTRODUCTION

Generative modeling has been dominated by methods that construct a transport from a simple reference law, typically Gaussian noise, to a complex data distribution (Song et al., 2021b; Ruthotto & Haber, 2021). Score-based diffusion models learn score fields over noisy intermediate distributions (Batzolis et al., 2021; Song et al., 2021a), whereas flow matching learns a velocity field that transports a reference distribution to data (Lipman et al., 2022). Sampling then follows the corresponding learned dynamics (Anderson, 1982; Song & Ermon, 2019; Dhariwal & Nichol, 2021; Lipman et al., 2022).

In this paper, we study a different mode of data generation. Particularly, instead of learning a global transport from noise to data, we seek to construct a discrete stochastic dynamics

$$x_0, x_1, x_2, \dots \tag{1}$$

such that

$$x_0 \sim p, \qquad x_j \sim p \quad \text{for all generation steps } j \geq 1, \tag{2}$$

where p denotes the data distribution. In other words, our approach states that rather than learning how to move *toward* the data distribution from an initial prior, we learn how to move *within* the data distribution while preserving it at every step. Within this mode of generation, the goal shifts from *transport* to *invariance*: rather than transforming noise into data, the sampler maintains the data distribution under a learned probability-preserving dynamics. Figure 1 illustrates the setting.

This viewpoint is particularly natural in modern augmented retrieval systems (Lewis et al., 2020) and latent generative pipelines (Rombach et al., 2022). In such systems, generation does not begin from an information-less state. Instead, one has access to existing valid examples, nearest neighbors, memory entries, or latent codes that already lie near the target distribution of interest (Blattmann et al., 2022). We refer to such existing examples or latent codes, which already lie on or near the target distribution rather than being fresh Gaussian noise, as *data-supported states*. In such settings, starting from pure noise is not the only viable option, and may not even be the most natural one. If one can start the generation process from data-supported states, then a probability-preserving dynamics becomes a natural complement to noise-to-data generation.

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.

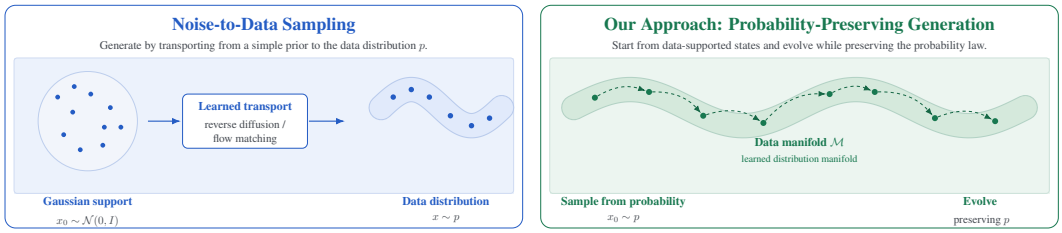


Figure 1: Two ways to use a pretrained generative model. Standard sampling starts from Gaussian noise and transports it to the data distribution. Our setting starts from an existing state near the target distribution and applies a stochastic transition designed to preserve the relevant law. Repeated application can produce progressively larger variations of the starting state.

Langevin dynamics, which is at the heart of modern generative models (Ho et al., 2020; Song & Ermon, 2019), provides precisely this intuition (Bussi & Parrinello, 2007): in continuous time it preserves p as an invariant measure (Roberts & Tweedie, 1996), but its finite-step discretization does not, and the discrepancy breaks the central requirement in equation 2 (Vempala & Wibisono, 2019); Section 2.1 makes this precise. Our starting point is that a generative model based on stationary, probability-preserving dynamics should address this issue directly.

We therefore develop two probability-preserving mechanisms that start from samples already drawn from the density of interest. The *first* is a corrected Langevin dynamics based on a conservative score, or equivalently a denoiser. In its score-based form, the correction restores invariance of p; in its denoiser-based form, it naturally acts on a Gaussian-smoothed law p_σ and therefore preserves p_σ. The *second* is a predictor–corrector flow construction: starting from $x \sim p$, one first adds a controlled amount of noise to move the sample to an intermediate bridge distribution, and then uses a learned flow or direct solution map to return it to p. Thus we present two distinct but related ways to probe a probability density when initialization already lies on, or near, that density. Notably, in both cases, the required score, denoiser, or flow, can be obtained directly from a pretrained flow-matching model, so neither mechanism requires training a new generative model from scratch.

Starting a stochastic process from data-supported states is an established idea, for example, persistent chains and replay buffers in energy-based model training (Tieleman, 2008; Du & Mordatch, 2019), the noise-and-reverse editing of SDEdit (Meng et al., 2021), and Energy Matching (Balcerak et al., 2025), which trains a time-independent potential whose Langevin dynamics can traverse the data manifold. What is new here is the construction from a pretrained flow checkpoint without retraining, the residual acceptance rule, and the approximation analysis; Section A expands the discussion.

A key practical advantage is that our construction reuses pretrained flow models *without further training*. Whereas manifold-aware perturbations modify the training distribution (Keegan & Ruthotto, 2026), we modify how an existing flow is sampled. Recent work has also emphasized the importance of conservation in score-based models and explored Metropolis-type corrections based on line integration of the score (Chao et al., 2023; Sjöberg et al., 2023); we build on these ideas to construct, from pretrained flow checkpoints, transition kernels whose organizing principle is not reverse noising, but stationary probing of the data distribution itself.

Our contribution is a checkpoint-to-kernel construction that reuses a pretrained flow model without retraining. First, we derive two stochastic transition kernels with explicit ideal target laws: a denoiser-based Metropolis transition (dMALA), which targets p_σ in the conservative-score, exact-line-integration limit, and a predictor–corrector transition that targets p under an exact flow. Second, we use a Wasserstein perturbation analysis to separate initialization mismatch from accumulated one-step error. Third, we characterize the resulting dynamics empirically through generated-state repair, controlled variation as the chain length grows, repeated use, initialization controls, compute, and run-to-run variability. The construction complements rather than replaces unconditional generation from noise. In the ideal setting of an exact conservative score and exact line integration, the first mechanism preserves the appropriate target law exactly; in the ideal setting of an exact bridge flow, the second mechanism preserves p exactly. In practice, the quality of these approximations depends on the learned score, denoiser, or flow model, and on the numerical approximation used in the correction step.

2 METHOD

We seek generative dynamics that leave a target law invariant. Throughout, p denotes the target data density on $\mathbb{R}^d$, and we assume access to samples $x \sim p$. We distinguish the original target p from its Gaussian-smoothed law

$$p_\sigma = p * \mathcal{N}(0, \sigma^2 I).$$

Exact score-based Metropolis correction preserves the law whose score is used. In Section 2.2, the score-based construction preserves p, while the denoiser formulation preserves the smoothed law p_σ. The exact predictor–corrector construction instead preserves p.

2.1 ON THE NECESSITY OF LANGEVIN DYNAMICS CORRECTION

A natural starting point is overdamped Langevin dynamics

$$dX_t = s(X_t)\, dt + \sqrt{2}\, dW_t, \qquad s(x) = \nabla \log p(x), \tag{3}$$

which preserves p in continuous time under standard regularity assumptions (Roberts & Tweedie, 1996). Thus, at the infinitesimal level, Langevin dynamics is probability-preserving.

The difficulty appears after discretization. The Euler–Maruyama step is

$$y = x + h\, s(x) + \sqrt{2h}\, z, \qquad z \sim \mathcal{N}(0, I), \tag{4}$$

which is the **U**nadjusted **L**angevin **A**lgorithm (ULA) (Durmus & Moulines, 2017). Although Equation 4 has the correct local geometry, it does not preserve the target law exactly for finite step size.

A Gaussian example makes this explicit - for $p = \mathcal{N}(0, I)$, so that $s(x) = -x$, one step of equation 4 maps $x \sim \mathcal{N}(0, I)$ to $y = (1-h)x + \sqrt{2h}\, z \sim \mathcal{N}(0, (1+h^2)I)$, which differs from p for every $h > 0$.

This is the basic issue: Langevin dynamics is correct in continuous time, but its explicit discretization changes the law. The first part of our method corrects this finite-step error.

Figure 2 illustrates this effect on a Swiss-roll-like target.

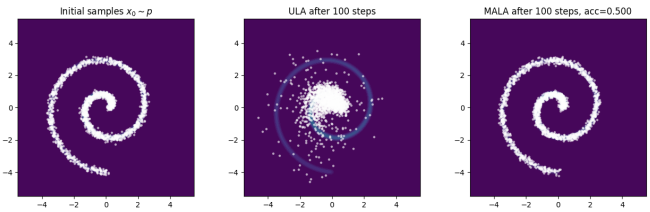


Figure 2: Comparison between unadjusted Langevin dynamics (ULA) and the corrected dynamics (MALA) on a Swiss-roll-like distribution. ULA drifts away from the target, while the corrected dynamics preserves it.

2.2 CORRECTED LANGEVIN DYNAMICS AND A DENOISER FORMULATION

Starting from the proposal, Equation 4, define the deterministic part of the update by

$$D_h(x) := x + h\, s_\theta(x), \tag{5}$$

so that the proposal can be written as

$$y = D_h(x) + \sqrt{2h}\, z, \qquad z \sim \mathcal{N}(0, I), \tag{6}$$

with proposal density

$$q_h(y \mid x) = \mathcal{N}(y;\, D_h(x),\, 2h\, I)\,. \tag{7}$$

When p is known up to a normalizing constant, the standard correction is **M**etropolis-**A**djusted **L**angevin **A**lgorithm (MALA) (Roberts & Tweedie, 1996; Roberts & Rosenthal, 1998). Its Metropolis–Hastings log-ratio is

$$\log r(x, y) = \log p(y) - \log p(x) + \log q_h(x \mid y) - \log q_h(y \mid x), \tag{8}$$

and the proposal is accepted with probability

$$\alpha(x, y) = \min\{1, \exp(\log r(x, y))\}. \tag{9}$$

This correction restores invariance of p.

In generative modeling, however, p is not available explicitly. To resolve this problem, we assume access to a learned score field s_θ. If $s_\theta = \nabla \log \widetilde{p}$ for a normalizable density $\widetilde{p}$, line integration recovers its log-density difference; this is the desired difference for p when the score is exact:

$$\Delta_{D_h}(x, y) := \frac{1}{h} \int_0^1 \Big\langle D_h\big(x + t(y - x)\big) - \big(x + t(y - x)\big),\, y - x \Big\rangle\, dt. \tag{10}$$

The corrected acceptance ratio is therefore

$$\log r_{D_h}(x, y) = \Delta_{D_h}(x, y) + \log q_h(x \mid y) - \log q_h(y \mid x), \tag{11}$$

and the proposal is accepted with probability

$$\alpha_{D_h}(x, y) = \min\big\{1, \exp\big(\log r_{D_h}(x, y)\big)\big\}. \tag{12}$$

When $D_h(x) = x + h\nabla \log p(x)$ and the integral Equation 10 is evaluated exactly, this reduces to MALA and preserves p.

This notation becomes especially useful when D_h is obtained from a denoiser. Let p_σ be the Gaussian-smoothed density, and let $D_\sigma(y) = \mathbb{E}[X \mid X + \sigma Z = y]$ be the exact conditional-mean denoiser, with $X \sim p$ and independent $Z \sim \mathcal{N}(0, I)$. Tweedie's identity gives

$$D_\sigma(y) = y + \sigma^2 \nabla \log p_\sigma(y). \tag{13}$$

Thus D_σ is exactly of the form Equation 5 with $h = \sigma^2$ and $s_\sigma(y) = \nabla \log p_\sigma(y)$. The proposal becomes

$$y = D_\sigma(x) + \sqrt{2}\, \sigma\, z, \qquad z \sim \mathcal{N}(0, I), \tag{14}$$

with proposal density

$$q_\sigma(y \mid x) = \mathcal{N}\big(y;\, D_\sigma(x),\, 2\sigma^2 I\big). \tag{15}$$

The corresponding line integral is

$$\Delta_{D_\sigma}(x, y) = \frac{1}{\sigma^2} \int_0^1 \Big\langle D_\sigma\big(x + t(y - x)\big) - \big(x + t(y - x)\big),\, y - x \Big\rangle\, dt, \tag{16}$$

so the exact correction leaves the smoothed density p_σ invariant. This point is important: the denoiser-based Metropolis construction preserves p_σ, not the original target law p.

A further simplification is obtained by approximating Equation 16 with the trapezoidal rule. Define the residual

$$r_\sigma(x) := D_\sigma(x) - x. \tag{17}$$

Then,

$$\Delta_{D_\sigma}(x, y) \approx \frac{1}{2\sigma^2} \big\langle r_\sigma(x) + r_\sigma(y),\, y - x \big\rangle. \tag{18}$$

Combining this with the Gaussian proposal ratio, the cross terms cancel and the approximate log-acceptance ratio becomes

$$\log r_{D_\sigma}(x, y) \approx \frac{1}{4\sigma^2} \Big(\|D_\sigma(x) - x\|^2 - \|D_\sigma(y) - y\|^2 \Big). \tag{19}$$

This yields a particularly simple denoiser-based sampler requiring only two evaluations of D_σ per step. We refer to this sampler as **d**enoising MALA, or **dMALA**, in short, because it reduces to the standard Metropolis-adjusted Langevin algorithm in the conservative-score, exact-line-integration limit.

Training the denoiser is straightforward. For σ fixed or sampled from a small interval close to 0, one minimizes

$$\mathcal{L}_{\text{denoise}} = \mathbb{E}_{x,\epsilon,\sigma}\left\|D_\sigma(x+\sigma\epsilon) - x\right\|^2, \qquad \epsilon \sim \mathcal{N}(0, I). \tag{20}$$

A training-time regularizer that encourages a conservative score field is given in Section C.5; the pretrained backbones used in this paper are taken as-is, and Section 2.4 examines the resulting approximation effects.

Pretrained flow models provide practical denoiser approximations without additional training. A conditional velocity determines the exact posterior-mean denoiser when that velocity is exact; a direct endpoint map instead supplies a denoising surrogate. Section B derives both constructions and distinguishes their approximation errors.

2.3 A PREDICTOR–CORRECTOR APPROACH FOR PROBABILITY-INVARIANT FLOW

The previous subsection used a local stochastic proposal together with a Metropolis correction. In the ideal denoiser formulation, this yields a Markov chain (Chung, 1967) whose invariant law is the smoothed distribution p_σ. We now describe a different construction whose ideal target is the original law p directly.

The flow-based approach considered here follows a different principle. Instead of using a local proposal followed by a Metropolis accept/reject step, we first allow the sample to move a controlled distance away from the target distribution by adding a small amount of noise, and then use a learned flow to transport it back toward the data distribution. This predictor–corrector mechanism can make substantial moves even when the geometry of the manifold is highly nonlinear, while preserving the target law in the exact-flow limit.

Let $p_1 = p$ denote the target law, let $p_0 = \mathcal{N}(0, I)$, and consider the linear bridge

$$x_t = (1-t)z + tx, \qquad x \sim p_1, \quad z \sim p_0 \quad \text{independently.} \tag{21}$$

We denote by p_t the law of x_t. Thus p_t interpolates between noise at $t = 0$ and data at $t = 1$.

Suppose now that we have trained a flow model associated with this bridge. This may be represented either by a velocity field

$$\dot{x} = v_\theta(x, t), \qquad x(\tau) = x_\tau, \qquad t \in [\tau, 1], \tag{22}$$

with flow map $\Phi_{\theta,t\to s}$, or by a direct solution map

$$f_\theta(x_t, t, s) \approx \Phi_{t\to s}(x_t). \tag{23}$$

For SoFlow, the direct solution map avoids numerical ODE integration at sampling time; for SiT-REPA we realize the return map by integrating its velocity field.

Starting from $x \sim p_1$, choose $\tau \in (0, 1)$, which controls the amount of refreshment, and draw

$$\hat{x} = (1-\tau)z + \tau x, \qquad z \sim \mathcal{N}(0, I). \tag{24}$$

Since $\hat{x} \sim p_\tau$, this predictor step moves the sample slightly away from the target distribution while remaining on the bridge. We then apply the learned flow from time τ back to time 1:

$$x^+ = \Phi_{\theta,\tau\to 1}(\hat{x}) \qquad \text{or} \qquad x^+ = f_\theta(\hat{x}, \tau, 1). \tag{25}$$

The following proposition uses the exact bridge flow $\Phi_{t\to s}$, not its learned approximation. The notation $(\Phi_{t\to s})_{\#}p_t$ denotes the distribution obtained by applying $\Phi_{t\to s}$ to a sample with law p_t.

Proposition 1 (Probability invariance of the predictor–corrector kernel)**.** *Assume that for every* $0 \le t \le s \le 1$,

$$(\Phi_{t\to s})_{\#}p_t = p_s. \tag{26}$$

Let $x \sim p_1$*, draw* $z \sim p_0$ *independently, form* $\hat{x}$ *by Equation 24, and set* $x^+ = \Phi_{\tau\to 1}(\hat{x})$*. Then* $x^+ \sim p_1 = p$*.*

Proof. By construction, $\hat{x} \sim p_\tau$. Applying the exact flow map from τ to 1 gives

$$x^+ = \Phi_{\tau\to 1}(\hat{x}) \sim (\Phi_{\tau\to 1})_{\#}p_\tau = p_1. \qquad \square$$

For the exact return map, this yields the invariant Markov kernel

$$K_\tau(x,\cdot): \quad x \mapsto x^+ = \Phi_{\tau\to 1}\big((1-\tau)z + \tau x\big), \qquad z \sim \mathcal{N}(0, I). \tag{27}$$

Replacing the exact map by a learned or numerically approximated map gives the approximate return-map kernel considered below. For τ close to 1, this is a local move; smaller values give stronger refreshment but require an accurate return map over a longer bridge segment. If the sampler is restricted to $\tau \geq \tau_{\min} > 0$, the return map is queried only on $t \in [\tau_{\min}, 1]$ rather than on the full interval $[0, 1]$.

Conceptually, this construction replaces Metropolis rejection by a learned return map to the target distribution. The predictor adds a controlled amount of noise, and the corrector follows the learned probability flow back to p. This provides a second route to invariance, now based on flow matching rather than corrected Langevin dynamics.

2.4 Approximate kernels

The exact-model constructions identify the target laws. We now separate initialization error from error in an approximate transition.

Proposition 2 (Approximate kernels). *Let W_r, $r \geq 1$, be the Wasserstein distance of order r, and let P have invariant law π. All laws below have finite rth moments. On $\mathcal{B}_R = \{\mu : W_r(\mu, \pi) \leq R\}$, assume $W_r(\mu P, \pi) \leq \rho W_r(\mu, \pi)$ and $W_r(\mu\widehat{P}, \mu P) \leq \gamma$, where $0 \leq \rho < 1$ and $\gamma \leq (1-\rho)R$. Then, for every $\mu_0 \in \mathcal{B}_R$ and integer $K \geq 0$,*

$$W_r(\mu_0 \widehat{P}^K, \pi) \leq \rho^K W_r(\mu_0, \pi) + \gamma \frac{1-\rho^K}{1-\rho}. \tag{28}$$

Any invariant law of $\widehat{P}$ in $\mathcal{B}_R$ is within $\gamma/(1-\rho)$ of π.

The bound separates the initial mismatch, whose upper bound decays with K, from accumulated one-step error. It follows from a standard Wasserstein perturbation argument (Rudolf & Schweizer, 2018); Section C.4 gives the proof and conditional one-step estimates for the Gaussian-proposal dMALA and predictor–corrector kernels. The ImageNet variant in Algorithm 1 also changes the proposal and applies per-class normalization; these changes belong to the total one-step error γ.

We examine approximation effects in Section E.12. The median relative Jacobian asymmetry of the extracted denoiser is 0.017 on the Swiss roll and 0.18 on ImageNet latents. On the Swiss roll, the trapezoidal estimate of the log-density difference has correlation 0.979 with the exact value, versus 0.982 for 8-point Gaussian quadrature. These diagnostics examine the learned field and the numerical integration; they do not estimate γ or ρ on ImageNet.

3 Experiments

We organize the experiments around four questions: **(Q1)** Do the proposed dynamics remain close to their target laws in a setting where the density is known exactly? **(Q2)** Can a flow-derived kernel repair a weak generated state without discarding its identity? **(Q3)** How do the kernels behave under repeated use, and how does chain length control the transition from local variation to broader exploration? **(Q4)** What quality–compute trade-offs do the kernels provide when they reuse pretrained ImageNet flow models?

Baselines. For ImageNet-256, we use the official 1-NFE SoFlow-XL/2-cond sampler (Luo et al., 2025) and no-CFG Euler sampling with SiT-REPA-XL (Yu et al., 2024) as backbone reference points. For the synthetic Swiss-roll experiment, the relevant baseline is unadjusted Langevin dynamics (ULA) (Durmus & Moulines, 2017). Unless stated otherwise, ImageNet distributional metrics use 50,000 samples against the ADM ImageNet-256 reference batch. We use neural-network function evaluations (NFEs) as the common measure of sampling cost. Additional seed and retrieval controls, together with the full NFE accounting, are reported in the appendix.

3.1 Probability preservation on a synthetic target

We compare the two probability-preserving mechanisms of Section 2 on a two-dimensional Swiss-roll-like target density given by a Gaussian mixture supported along a spiral. This problem is useful

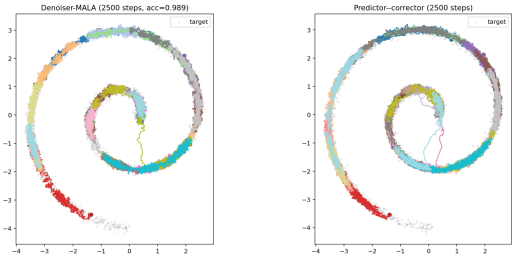


Figure 3: Side-by-side comparison. Left: dMALA trajectories on p_σ. Right: predictor–corrector flow on p. Both use 32 particles for 2500 steps and continue to move along the roll while remaining on the correct law.

since the target density is known exactly, so we can evaluate negative log-likelihood under the true law, while the geometry is sufficiently nonlinear to make long-term mixing visually and quantitatively informative. In all experiments below, both methods use the same pretrained SoFlow model (Luo et al., 2025). dMALA is implemented through the denoiser extracted from that flow model, while the predictor–corrector method uses the direct solution map itself.

A key distinction between the two methods is the law they target. We use p_t for the time-t marginal of the flow bridge, whereas $p_\sigma = p * \mathcal{N}(0, \sigma^2 I)$ denotes the Gaussian-smoothed target, with σ a smoothing scale rather than a time variable. dMALA is evaluated against p_σ, while the predictor–corrector construction targets the original distribution p directly.

Results. Figure 3 shows that both kernels continue to move along the spiral over 2,500 steps while staying close to the law they are designed to preserve. The ablations in Tables 4 and 5 show the expected movement–fidelity trade-off. For dMALA, reducing the smoothing scale σ from 0.111 to 0.020 reduces the mean move from 1.508 to 0.370, while MMD remains 0.013–0.017. For predictor–corrector, increasing τ from 0.70 to 0.95 reduces MMD from 0.057 to 0.020, while the mean move decreases from 2.811 to 0.568. Thus both mechanisms provide controllable movement while remaining close to their respective target laws.

3.2 ImageNet-256 generation

We evaluate the proposed kernels on ImageNet-256 (Deng et al., 2009) using two pretrained latent flow backbones: SoFlow-XL/2-cond (Luo et al., 2025) and SiT-REPA-XL (Yu et al., 2024). For SoFlow, the reference sampler is the official 1-NFE solution operator; for SiT-REPA, we use the no-CFG Euler sampler from our implementation. Unless stated otherwise, distributional metrics are computed on 50,000 samples against the ADM ImageNet-256 reference batch.

Throughout this subsection, a *seed* means the initial image or latent state from which a chain starts, not a random-number seed.

Repairing generated states. To test repair rather than replacement, we generate imperfect initial states using 25-step SiT Euler sampling without classifier-free guidance, obtaining FID 10.70. Table 1 compares repairing these states with simply drawing a fresh sample from SoFlow. After $K = 10$ steps, dMALA reduces FID to 3.19, close to the 2.96 of fresh resampling, while retaining substantially more of the particular input: DINO similarity is 0.80 versus 0.43, and LPIPS is 0.32 versus 0.72. Label agreement is also similar (0.85 versus 0.87). Thus dMALA repairs a weak generated state without simply replacing it with another sample from the same class.

Controlled variation from data-supported states. We next initialize both kernels from the same set of 50,000 ImageNet VAE latents. Decoding the seeds provides a zero-movement baseline: it gives FID 1.27, but does not produce a new state. We therefore interpret the seeded results jointly in terms of distributional quality and displacement from the input. At $K = 10$, dMALA gives FID 1.70 and predictor–corrector gives FID 2.28 (Table 2), while both move away from the seed; the corresponding LPIPS displacements are 0.26 and 0.29 (Table 11). Replacing training seeds with validation seeds gives FID 2.43 and 2.54, respectively (Table 10), showing that train/reference overlap contributes to the lowest values but does not account for the full behavior. Renormalization

Table 1: Repair versus fresh resampling on weak generated ImageNet-256 states. dMALA starts from the same $50{,}000$ weak SiT inputs; the fresh same-class row instead draws a new SoFlow sample with the same conditioning label. Added NFE counts network evaluations after the inputs are available. Distribution metrics use $50{,}000$ images. LPIPS and DINO measure similarity to the particular input; LPIPS uses a fixed $2{,}048$-pair subset, while DINO and label agreement use all $50{,}000$ pairs. Label agreement is the fraction of outputs whose predicted ImageNet class matches the conditioning label. Reference values are DINO 0.08 for unrelated images and LPIPS 0.73 to the nearest distinct training image.

		Distribution quality				Similarity to input		
State / operation	Added NFE	FID ↓	IS ↑	Prec. ↑	Rec. ↑	LPIPS ↓	DINO ↑	Label agr. ↑
Weak inputs (SiT, 25 steps, no CFG)	0	10.70	131.0	0.66	0.67	0.00	1.00	0.71
Fresh same-class sample (SoFlow, 1 NFE)	1	2.96	238.53	0.795	0.564	0.72	0.43	0.87
dMALA repair (K=10)	20	3.19	218.2	0.78	0.61	0.32	0.80	0.85

Table 2: Controlled variation from data-supported ImageNet-256 states. Gray rows are reference points: the noise-to-data sampler and, for SoFlow, the zero-movement decode baseline. "FID, 3 runs" reports mean ± sample standard deviation over three runs; supporting initialization, displacement, and run-level results are in Sections E.6 to E.8 and E.10.

Method / setting	NFE	FID ↓	FID, 3 runs	sFID ↓	IS ↑	Prec. ↑	Rec. ↑
SoFlow-XL/2-cond backbone							
Noise-to-data reference (1-NFE)	1	2.96		4.60	238.53	0.795	0.564
Seed-only baseline (decode; zero movement)	0	1.27		4.00	295.3	0.73	0.73
Pred.–Corr., K=10, t_{noise}=0.3	10	2.28	2.08 ± 0.17	5.47	297.37	0.79	0.62
dMALA, K=10, σ=0.3	20	1.70	1.72 ± 0.02	5.05	300.04	0.79	0.65
Pred.–Corr., K=20, t_{noise}=0.5	20	4.67	4.53 ± 0.12	5.33	349.84	0.87	0.50
dMALA, K=20, σ=0.3	40	2.17	2.19 ± 0.03	5.81	306.25	0.80	0.62
SiT-REPA-XL backbone							
Noise-to-data reference (no CFG, 400 Euler steps)	400	6.61		5.62	152.52	0.68	0.69
Pred.–Corr., K=10, t_{noise}=0.3	100	3.11	3.14 ± 0.03	6.06	220.35	0.73	0.68
dMALA, K=10, σ=0.3	200	1.93	1.93 ± 0.01	5.62	241.20	0.74	0.70
Pred.–Corr., K=20, t_{noise}=0.5	400	3.06	3.07 ± 0.02	6.24	199.50	0.72	0.69
dMALA, K=20, σ=0.3	800	1.79	1.82 ± 0.03	5.14	236.03	0.74	0.71

is described in Algorithm 1, displacement statistics are in Section E.8, and run-to-run variation is reported in Section E.10.

Behavior under repeated use. We compare repeated transitions in a $10{,}000$-sample study using weak SiT-generated initial states (Table 13). The proposal-only baseline repeatedly applies the perturb-and-return proposal used inside dMALA, without the full correction. It reaches FID 5.84 at $K = 10$, after which further iteration reduces coverage. At 40 NFE, proposal-only has FID 7.31 and Recall 0.575, compared with 5.83 and 0.666 for dMALA. Thus the full kernel gives better distributional quality and coverage in the longer repeated-use regime.

Chain length controls exploration. Increasing K moves the state progressively farther from its seed (Figure 4 and section E.8). From $K = 10$ to $K = 200$, LPIPS grows from 0.26 to 0.48 and the median latent distance grows from 33.6 to 52.8, approaching the 55.6 spacing to a nearest distinct training image. Over the same range, the fraction of chains whose nearest image among the 50,000 starting latents is still their own starting point falls from 99.8% to 70.1%. This movement has a cost: FID changes from 1.70 to 5.61, while Precision remains between 0.79 and 0.82. Thus the same dynamics support two related uses: short chains retain substantial similarity to the input and provide local variation, while longer chains move progressively farther from the starting state and provide broader exploration. We therefore view K as a variation-exploration dial, not a parameter that should simply be made as large as possible.

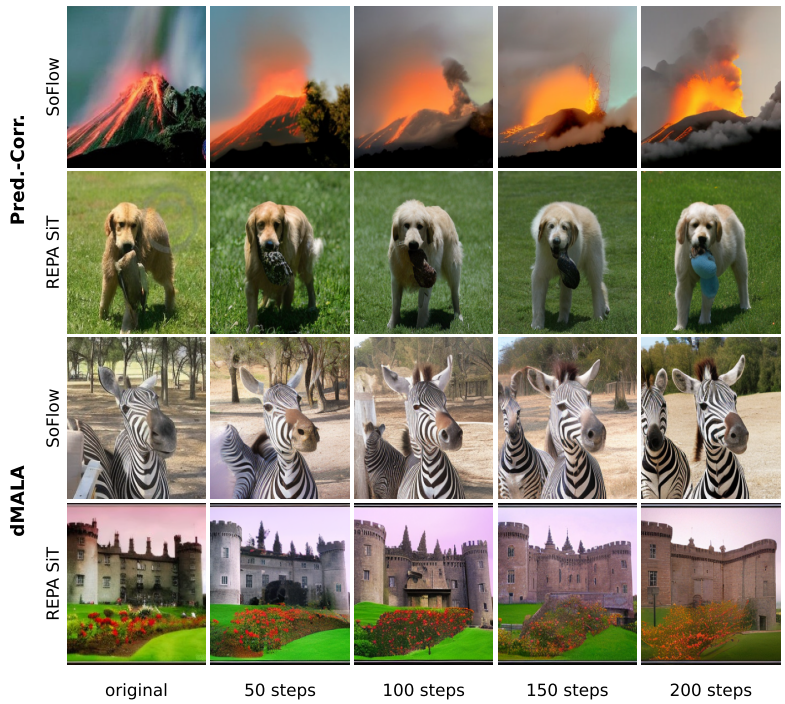


Figure 4: Increasing variation with chain length on ImageNet-256. Each row follows one initial state as K increases, showing progressively larger changes from the starting state. Additional examples are in Appendix G.

Computational efficiency. Table 8 compares the two operating modes using the same SiT-REPA checkpoint. Predictor–corrector reaches FID 4.91 at 200 NFE, compared with 6.51 for 800-step Euler integration; dMALA reaches FID 1.93 at 200 NFE and 1.79 at 800 NFE. These figures quantify quality per backbone evaluation in the state-supported setting; direct Euler generation instead starts from noise.

4 Discussion and Conclusion

We introduced two reusable transition kernels, dMALA and predictor–corrector, that are obtained from a pretrained flow checkpoint without retraining. Their ideal invariant laws are p_σ and p, respectively, while Proposition 2 characterizes how model and numerical errors perturb this ideal behavior.

The experiments show that the same dynamics support both repair and controlled variation. On ImageNet-256, dMALA repairs deliberately weak generated states from FID 10.70 to 3.19 while retaining high input similarity (DINO 0.80). Increasing the chain length then moves valid states progressively farther from their inputs, providing a controllable transition from local variation to broader exploration. Together, these results show how pretrained flows can be reused when a useful initial state is already available.

Limitations. Both kernels inherit approximation error from the pretrained flow. dMALA targets the smoothed law p_σ rather than p, while the predictor–corrector kernel depends on the accuracy of the learned return map. Longer chains can therefore accumulate model error.

AI USE STATEMENT

In this work, we used generative AI tools for grammar correction and improving the phrasing and clarity of the manuscript. We have not used generative AI tools for developing scientific ideas, designing the methodology or experiments, analyzing results, or generating scientific claims. We reviewed all AI-assisted edits and verified that they preserved the intended technical meaning. We take responsibility for the final content of this work, including text, claims or artifacts produced with the aid of generative AI.

ETHICS STATEMENT

This work develops general-purpose methods for generative modeling and does not involve human subjects, personally identifiable information, or sensitive data. Our experiments use standard benchmark datasets and synthetic data. As with generative modeling methods more broadly, the proposed approach could potentially be used in applications that generate misleading or harmful content; however, such applications are outside the scope of this work. We are not aware of additional ethical concerns specific to the proposed methodology.

REPRODUCIBILITY STATEMENT

We provide the technical details needed to reproduce our results in the appendices. Appendix C gives pseudocode and derivations for the proposed samplers, including the implemented dMALA proposal and the proof of Proposition 2; Appendix D specifies the datasets, model architectures, pretrained backbones, implementation details and computational setup; Appendix E states the seed protocol, the NFE accounting of every row and run-to-run error bars with their seed semantics. All ImageNet-256 experiments use publicly available checkpoints and the ADM evaluation suite, and no model was trained or fine-tuned for ImageNet-256. The code will be made available to the public upon acceptance.

REFERENCES

Michael S Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants. *arXiv preprint arXiv:2209.15571*, 2022.

Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions. *arXiv preprint arXiv:2303.08797*, 2023.

Brian D. O. Anderson. Reverse-time diffusion equation models. *Stochastic Processes and their Applications*, 12(3):313–326, 1982.

Michal Balcerak, Tamaz Amiranashvili, Antonio Terpin, Suprosanna Shit, Lea Bogensperger, Sebastian Kaltenbach, Petros Koumoutsakos, and Bjoern Menze. Energy matching: Unifying flow matching and energy-based models for generative modeling. In *Advances in Neural Information Processing Systems*, 2025.

Georgios Batzolis, Jan Stanczuk, Carola-Bibiane Schönlieb, and Christian Etmann. Conditional image generation with score-based diffusion models. *arXiv preprint arXiv:2111.13606*, 2021.

Yoshua Bengio, Aaron Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence*, 35(8):1798–1828, 2013.

Andreas Blattmann, Robin Rombach, Kaan Oktay, Jonas Müller, and Björn Ommer. Retrieval-augmented diffusion models. In *Advances in Neural Information Processing Systems*, 2022.

Giovanni Bussi and Michele Parrinello. Accurate sampling using Langevin dynamics. *Physical Review E—Statistical, Nonlinear, and Soft Matter Physics*, 75(5):056707, 2007.

John Charles Butcher. *Numerical Methods for Ordinary Differential Equations*. John Wiley & Sons, 3rd edition, 2016.

Chen-Hao Chao, Wei-Fang Sun, Bo-Wun Cheng, and Chun-Yi Lee. On investigating the conservative property of score-based generative models. In *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pp. 4076–4095, 2023.

Ricky T. Q. Chen, Yulia Rubanova, Jesse Bettencourt, and David K. Duvenaud. Neural ordinary differential equations. In *Advances in Neural Information Processing Systems*, volume 31, 2018.

Kai Lai Chung. *Markov Chains with Stationary Transition Probabilities*, volume 104 of *Die Grundlehren der mathematischen Wissenschaften*. Springer-Verlag, New York, 2nd edition, 1967.

Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. ImageNet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 248–255. IEEE, 2009.

Prafulla Dhariwal and Alexander Nichol. Diffusion models beat GANs on image synthesis. In *Advances in Neural Information Processing Systems*, volume 34, pp. 8780–8794, 2021.

Laurent Dinh, Jascha Sohl-Dickstein, and Samy Bengio. Density estimation using Real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

Yilun Du and Igor Mordatch. Implicit generation and modeling with energy based models. In *Advances in Neural Information Processing Systems*, 2019.

Alain Durmus and Éric Moulines. Nonasymptotic convergence analysis for the unadjusted Langevin algorithm. *The Annals of Applied Probability*, 27(3):1551–1587, 2017.

Ian J Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014.

Will Grathwohl, Ricky TQ Chen, Jesse Bettencourt, Ilya Sutskever, and David Duvenaud. FFJORD: Free-form continuous dynamics for scalable reversible generative models. *arXiv preprint arXiv:1810.01367*, 2018.

Amir Hertz, Ron Mokady, Jay Tenenbaum, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Prompt-to-prompt image editing with cross attention control. *arXiv preprint arXiv:2208.01626*, 2022.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. In *Advances in Neural Information Processing Systems*, volume 33, pp. 6840–6851, 2020.

Tero Karras, Samuli Laine, and Timo Aila. A style-based generator architecture for generative adversarial networks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 4401–4410, 2019.

Bahjat Kawar, Michael Elad, Stefano Ermon, and Jiaming Song. Denoising diffusion restoration models. *Advances in neural information processing systems*, 35:23593–23606, 2022.

Katherine Keegan and Lars Ruthotto. Manifold-aware perturbations for constrained generative modeling. *arXiv preprint arXiv:2601.23151*, 2026.

Diederik P Kingma and Max Welling. Auto-encoding variational Bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks. In *Advances in Neural Information Processing Systems*, volume 33, pp. 9459–9474, 2020.

Yaron Lipman, Ricky TQ Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling. *arXiv preprint arXiv:2210.02747*, 2022.

Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow. *arXiv preprint arXiv:2209.03003*, 2022.

Tianze Luo, Haotian Yuan, and Zhuang Liu. SoFlow: Solution flow models for one-step generative modeling. *arXiv preprint arXiv:2512.15657*, 2025.

Jerrold E. Marsden and Anthony J. Tromba. *Vector Calculus*. W. H. Freeman, 6 edition, 2011.

Chenlin Meng, Yutong He, Yang Song, Jiaming Song, Jiajun Wu, Jun-Yan Zhu, and Stefano Ermon. SDEdit: Guided image synthesis and editing with stochastic differential equations. *arXiv preprint arXiv:2108.01073*, 2021.

Ron Mokady, Amir Hertz, Kfir Aberman, Yael Pritch, and Daniel Cohen-Or. Null-text inversion for editing real images using guided diffusion models. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 6038–6047, 2023.

Kirill Neklyudov, Rob Brekelmans, Daniel Severo, and Alireza Makhzani. Action matching: Learning stochastic dynamics from samples. In *International conference on machine learning*, pp. 25858–25889. PMLR, 2023.

Alexander Quinn Nichol and Prafulla Dhariwal. Improved denoising diffusion probabilistic models. In *International conference on machine learning*, pp. 8162–8171. PMLR, 2021.

Maria-Elena Nilsback and Andrew Zisserman. Automated flower classification over a large number of classes. In *Sixth Indian Conference on Computer Vision, Graphics & Image Processing*, pp. 722–729. IEEE, 2008.

George Papamakarios, Eric Nalisnick, Danilo Jimenez Rezende, Shakir Mohamed, and Balaji Lakshminarayanan. Normalizing flows for probabilistic modeling and inference. *Journal of Machine Learning Research*, 22(57):1–64, 2021.

William Peebles and Saining Xie. Scalable diffusion models with transformers. In *Proceedings of the IEEE/CVF international conference on computer vision*, pp. 4195–4205, 2023.

Danilo Jimenez Rezende, Shakir Mohamed, and Daan Wierstra. Stochastic backpropagation and approximate inference in deep generative models. In *International conference on machine learning*, pp. 1278–1286. PMLR, 2014.

Gareth O. Roberts and Jeffrey S. Rosenthal. Optimal scaling of discrete approximations to Langevin diffusions. *Journal of the Royal Statistical Society: Series B*, 60(1):255–268, 1998.

Gareth O. Roberts and Richard L. Tweedie. Exponential convergence of Langevin distributions and their discrete approximations. *Bernoulli*, 2(4):341–363, 1996.

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, and Björn Ommer. High-resolution image synthesis with latent diffusion models. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10684–10695, 2022.

Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *International Conference on Medical image computing and computer-assisted intervention*, pp. 234–241. Springer, 2015.

Daniel Rudolf and Nikolaus Schweizer. Perturbation theory for Markov chains via Wasserstein distance. *Bernoulli*, 24(4A):2610–2639, 2018. doi: 10.3150/17-BEJ938.

Lars Ruthotto and Eldad Haber. An introduction to deep generative modeling. *GAMM-Mitteilungen*, 44(2):e202100008, 2021.

Anders Sjöberg, Jakob Lindqvist, Magnus Önnheim, Mats Jirstrand, and Lennart Svensson. MCMC-correction of score-based diffusion models for model composition. *arXiv preprint arXiv:2307.14012*, 2023.

Jascha Sohl-Dickstein, Eric Weiss, Niru Maheswaranathan, and Surya Ganguli. Deep unsupervised learning using nonequilibrium thermodynamics. In *International conference on machine learning*, pp. 2256–2265. pmlr, 2015.

Jiaming Song, Chenlin Meng, and Stefano Ermon. Denoising diffusion implicit models. *arXiv preprint arXiv:2010.02502*, 2020.

Yang Song and Stefano Ermon. Generative modeling by estimating gradients of the data distribution. In *Advances in Neural Information Processing Systems*, 2019.

Yang Song and Diederik P. Kingma. How to train your energy-based models. *arXiv preprint arXiv:2101.03288*, 2021.

Yang Song, Conor Durkan, Iain Murray, and Stefano Ermon. Maximum likelihood training of score-based diffusion models. *Advances in neural information processing systems*, 34:1415–1428, 2021a.

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

Tijmen Tieleman. Training restricted Boltzmann machines using approximations to the likelihood gradient. In *Proceedings of the 25th International Conference on Machine Learning*, pp. 1064–1071, 2008.

Aaron van den Oord, Nal Kalchbrenner, Lasse Espeholt, Koray Kavukcuoglu, Oriol Vinyals, and Alex Graves. Conditional image generation with PixelCNN decoders. In *Advances in Neural Information Processing Systems*, volume 29, 2016.

Santosh Vempala and Andre Wibisono. Rapid convergence of the unadjusted Langevin algorithm: Isoperimetry suffices. *Advances in neural information processing systems*, 32, 2019.

Yinhuai Wang, Jiwen Yu, and Jian Zhang. Zero-shot image restoration using denoising diffusion null-space model. *arXiv preprint arXiv:2212.00490*, 2022.

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

APPENDIX OUTLINE

The appendices are organized as follows:

- Section A discusses related work relevant to our methods.
- Section B derives the denoiser used at sampling time from a pretrained flow-matching model, with a general formula for any linear bridge $x_t = \kappa(t)x + \sigma(t)z$ and an explicit specialization to the linear interpolant $x_t = tx + (1-t)z$.
- Section C provides explicit pseudocode for one step of the Denoising MALA (dMALA) sampler (Algorithm 1) and the Predictor–Corrector sampler (Algorithm 2), together with the cancellation argument that takes the trapezoidal line-integral acceptance ratio of Section 2.2 into the simple two-evaluation form used in our experiments.
- Section C.4 proves Proposition 2, and Section C.5 gives the training-time conservativity regularizer.
- Section D describes the datasets, pretrained backbones (SoFlow-XL/2-cond and SiT-REPA-XL), the Flowers-102 U-Net architecture, and the compute used.
- Section E reports additional quantitative results: an evaluation on Oxford Flowers-102 and a dMALA ablation over smoothing levels σ and chain lengths K on ImageNet-256 with the SoFlow backbone, in addition to a study on the computational efficiency of our method.
- Section E.5 to Section E.12 report the long-chain study, the initialization controls, the additional data-supported controls, the exploration diagnostics, the full repeated-use study, run-to-run error bars, the bridge-noise sweep with the terminal-denoising check, and the approximation diagnostics.
- Section F discusses broader societal impact.
- Section G shows generated-sample visualizations across step counts on Flowers-102 and ImageNet-256.

A RELATED WORK

Generative Modeling. Deep generative models have progressed from autoregressive networks (van den Oord et al., 2016), exact-likelihood normalizing flows (Dinh et al., 2016; Papamakarios et al., 2021), and variational autoencoders (Kingma & Welling, 2013; Rezende et al., 2014) to generative adversarial networks capable of high-resolution synthesis (Goodfellow et al., 2014; Karras et al., 2019). More recently, denoising diffusion probabilistic models (Sohl-Dickstein et al., 2015; Ho et al., 2020; Nichol & Dhariwal, 2021) have achieved state-of-the-art sample quality by learning to reverse a gradual noising process. While these foundational models predominantly focus on mapping a simple noise prior to the target data distribution, our work provides a complementary framework that operates directly within the data distribution itself.

Continuous-Time Transport and Flows. The generative processes of diffusion can be elegantly unified through the lens of continuous-time differential equations (Chen et al., 2018; Grathwohl et al., 2018). Recent formulations such as stochastic interpolants (Albergo & Vanden-Eijnden, 2022; Albergo et al., 2023), rectified flows (Liu et al., 2022), and action matching (Neklyudov et al., 2023) learn ordinary differential equations (ODEs) to smoothly transport probability mass from a base distribution to the target data. While traditionally optimized for efficient, straight-path generation from pure noise, the vector fields learned by these models provide the fundamental continuous-time dynamics that our predictor-corrector mechanism repurposes for local probability-preserving exploration.

Data-Supported Initialization. Beyond unconditional generation from noise, initializing stochastic processes from data-supported states is widely used in inverse problems (Kawar et al., 2022; Wang et al., 2022) and guided image editing. Methods like SDEdit (Meng et al., 2021) inject controlled noise into an existing sample and reverse the process to produce realistic modifications, while deterministic inversion techniques map real data into latent spaces for localized semantic editing (Song et al., 2020; Hertz et al., 2022; Mokady et al., 2023). Although these methods primarily aim to balance user-guided constraints with sample realism, they share the underlying principle of exploring

the local geometry around a given data manifold, a property we explicitly formalize as a stationary generative kernel.

The resulting approach differs from both standard diffusion models and classical energy-based modeling. Diffusion models learn a reverse transport from noise through a family of perturbed distributions (Song & Ermon, 2019; Song et al., 2021b). This typically requires learning a time-dependent score field $s(x, t)$ across many noise levels. In contrast, our first method does not require such a family of scores: because generation is initialized from data-supported states and proceeds by local probability-preserving moves, it is sufficient to learn a single conservative score field, or one associated with a small fixed noise level. Energy-based models, by contrast, typically learn a scalar energy and then sample by approximate Langevin dynamics, often from noise or from a replay mechanism (Du & Mordatch, 2019). Our approach instead begins from data-supported states and then either uses a corrected score/denoiser-based move or a predictor–corrector flow move.

Persistent chains and replay buffers in energy-based model training keep Langevin samples alive across parameter updates (Tieleman, 2008; Du & Mordatch, 2019); see Song & Kingma (2021) for a review of MCMC-based training. Energy Matching (Balcerak et al., 2025) trains a single time-independent scalar potential whose gradient transports samples to the data manifold and, near it, defines a Boltzmann law that Langevin dynamics can iterate indefinitely; it thus also treats the data manifold as something to be traversed rather than reached once, but requires training the potential. SDEdit (Meng et al., 2021) noises an existing image and reverses a diffusion model to edit it. The kernels here are not new in principle: what is new is their construction from a pretrained flow checkpoint without retraining, the residual acceptance rule, and the approximation analysis of Section 2.4.

The role of retrieval is then conceptually simple. Generation is performed by initializing the sampler from retrieved data points or retrieved latent codes, and then stepping forward with one of these probability-preserving mechanisms. When initialization follows the relevant invariant law, no burn-in is required; more generally, retrieved initializations place the sampler in a high-probability region of the relevant data manifold. This makes the framework especially attractive in latent spaces, where retrieval is natural, geometry is smoother, and local stochastic exploration can produce meaningful variation at lower computational cost (Bengio et al., 2013).

B Estimating a Denoiser from a Flow Model

We describe two ways to reuse a pretrained flow without training a separate denoiser: a conditional-mean estimate obtained from its velocity field, and a denoising surrogate obtained from its direct endpoint map. Both use the bridge on which the model was trained, but the two outputs need not coincide.

Let κ and σ be continuously differentiable, and consider the general linear bridge

$$x_t = \kappa(t)\, x + \sigma(t)\, z, \qquad x \sim p, \quad z \sim \mathcal{N}(0, I) \quad \text{independently}, \tag{29}$$

where $t \in [0, 1]$, $\kappa(1) = 1$, and $\sigma(1) = 0$. At the times used below, assume $\kappa(t) > 0$, $\sigma(t) > 0$, and $\kappa(t)\sigma'(t) - \kappa'(t)\sigma(t) \neq 0$. To connect this bridge to denoising, define a noisy variable

$$x_\eta = x + \eta z. \tag{30}$$

If we choose

$$\eta = \frac{\sigma(t)}{\kappa(t)}, \tag{31}$$

then

$$\kappa(t)\, x_\eta = \kappa(t)\, x + \kappa(t)\eta z = \kappa(t)\, x + \sigma(t)\, z = x_t. \tag{32}$$

Thus the bridge variable x_t is simply a scaled version of the noisy sample x_η.

Suppose first that the pretrained flow model provides a direct solution map

$$f_\theta(x_t, t, 1) \approx \Phi_{t\to 1}(x_t). \tag{33}$$

This map approximates transport to the data endpoint. Evaluating it at the scaled noisy input defines a flow-based denoising surrogate:

$$\widehat{D}^{\mathrm{map}}_\eta(y) := f_\theta\big(\kappa(t)y, t, 1\big), \qquad \eta = \frac{\sigma(t)}{\kappa(t)}. \tag{34}$$

An exact endpoint transport is not generally the conditional mean $D_\eta(y) = \mathbb{E}[X \mid X + \eta Z = y]$. Thus exact transport alone does not imply Tweedie's identity for $\widehat{D}_\eta^{\mathrm{map}}$. The construction still queries the flow at the correctly scaled point $\kappa(t)y$.

If instead the pretrained model provides only a velocity field

$$v_\theta(x_t, t) \approx \mathbb{E}[\dot{x}_t \mid x_t], \tag{35}$$

then the denoiser can still be recovered. Differentiating Equation 29 gives

$$\dot{x}_t = \kappa'(t)\, x + \sigma'(t)\, z. \tag{36}$$

Together, Equation 29 and Equation 36 form a 2×2 linear system in x and z. Solving for x gives

$$x = \frac{\sigma'(t)\, x_t - \sigma(t)\, \dot{x}_t}{\kappa(t)\sigma'(t) - \kappa'(t)\sigma(t)}. \tag{37}$$

Taking conditional expectations and substituting the learned velocity gives the denoiser estimate

$$\widehat{D}_\eta^{\mathrm{vel}}(y) := \frac{\sigma'(t)\kappa(t)y - \sigma(t)v_\theta(\kappa(t)y, t)}{\kappa(t)\sigma'(t) - \kappa'(t)\sigma(t)}, \qquad \eta = \frac{\sigma(t)}{\kappa(t)}. \tag{38}$$

For the exact conditional velocity, this expression equals $D_\eta(y)$. For a learned velocity, it is an approximation to that conditional mean.

For the linear bridge used in our experiments,

$$x_t = t\, x + (1 - t)\, z, \tag{39}$$

we have $\kappa(t) = t$, $\sigma(t) = 1 - t$, and therefore

$$\eta = \frac{1 - t}{t}. \tag{40}$$

When this flow-derived denoiser is used in dMALA, we identify this effective Gaussian noise scale with the dMALA smoothing parameter, $\sigma = \eta$. The direct-map surrogate becomes

$$\widehat{D}_\eta^{\mathrm{map}}(y) := f_\theta(ty, t, 1), \qquad t = \frac{1}{1 + \eta}. \tag{41}$$

The velocity-based conditional-mean estimate becomes

$$\widehat{D}_\eta^{\mathrm{vel}}(y) := ty + (1 - t)v_\theta(ty, t), \qquad t = \frac{1}{1 + \eta}. \tag{42}$$

Both constructions reuse a pretrained flow at bridge-consistent inputs. The velocity formula recovers the conditional-mean denoiser when the velocity is exact, whereas the direct-map construction is a denoising surrogate. Any discrepancy from the target score belongs to the approximation error of the resulting dMALA kernel.

C Algorithmic Details

This appendix gives explicit pseudocode for one step of each of the two probability-preserving samplers introduced in Section 2, and the short cancellation argument that turns the line-integral acceptance ratio of Section 2.2 into the simple two-evaluation form actually used in our experiments.

C.1 Denoising MALA (dMALA) Sampler

The reference dMALA kernel combines the proposal Equation 14 with the simplified acceptance ratio Equation 19. The ImageNet experiments use the proposal variant specified in Algorithm 1; in either case, the acceptance statistic uses one denoiser evaluation at the current state and one at the proposal.

Algorithm 1 One step of denoising MALA (dMALA).

Require: current state x_j, denoiser D_σ, smoothing level $\sigma > 0$

1: Draw $z \sim \mathcal{N}(0, I)$.

2: Form the reference proposal

$$y = D_\sigma(x_j) + \sqrt{2}\,\sigma z. \qquad \text{(cf. Equation 14)}$$

3: *ImageNet proposal variant.* For the ImageNet experiments, replace the preceding proposal by

$$\tilde{D}_\sigma(x) = f_\theta((1-\sigma)x + \sigma\varepsilon,\, \sigma \to 0)\,, \qquad \varepsilon \sim \mathcal{N}(0, I),$$

with fresh ε at every evaluation, and set $y = \tilde{D}_\sigma(x_j)$ with no additional Gaussian term. Use $\tilde{D}_\sigma$ in place of D_σ in the acceptance statistic below.

4: Compute the approximate log-acceptance ratio

$$\log r(x_j, y) \approx \frac{1}{4\sigma^2}\left(\|D_\sigma(x_j) - x_j\|^2 - \|D_\sigma(y) - y\|^2\right), \qquad \text{(cf. Equation 19)}$$

where D_σ denotes $\tilde{D}_\sigma$ for the ImageNet variant.

5: Set

$$\alpha = \min\{1, \exp(\log r(x_j, y))\}.$$

6: Draw $u \sim \mathrm{Unif}(0, 1)$.

7: **if** $u \le \alpha$ **then**

8: Set $x_{j+1} = y$. {accept}

9: **else**

10: Set $x_{j+1} = x_j$. {reject}

11: **end if**

12: *ImageNet only.* Normalize x_{j+1} per class using the fixed channelwise mean and standard deviation of the corresponding real latents.

Ensure: x_{j+1}

C.2 Predictor–Corrector Sampler

The predictor–corrector sampler combines a single Gaussian noising step at the bridge time τ with the application of the learned flow map $\Phi_{\theta,\tau\to1}$ back to the data end of the bridge (Algorithm 2). Any procedure that realizes this flow map suffices, and our framework is agnostic to the choice. When the pretrained model exposes a direct solution map $f_\theta(\cdot, \tau, 1)$ as for SoFlow (Luo et al., 2025), the corrector is a single network function evaluation (NFE). When the model is given as a velocity field v_θ as for SiT-REPA-XL (Yu et al., 2024), the flow map is realized by numerically integrating $\dot{x} = v_\theta(x, t)$ from $t = \tau$ to $t = 1$ using any ODE integrator of choice; in our experiments we use forward Euler with n_{steps} steps (Table 8), but Runge–Kutta variants, DPM-Solver, or any other integrator would equally well realize the same flow map. The total cost per probe iteration is therefore 1 NFE for a direct solution map and n_{steps} NFEs for a numerically integrated velocity field.

Algorithm 2 One step of the Predictor–Corrector sampler.

Require: current state x_j, bridge time $\tau \in (0, 1)$, flow map $\Phi_{\theta,\tau\to1}$

1: **Predictor:** draw $z \sim \mathcal{N}(0, I)$ and set $\hat{x} \leftarrow (1-\tau)\, z + \tau\, x_j$. Equation 24

2: **Corrector:** set $x_{j+1} \leftarrow \Phi_{\theta,\tau\to1}(\hat{x})$.

Ensure: x_{j+1}

C.3 Derivation of the Trapezoidal Acceptance Ratio

We give the cancellation argument that takes the trapezoidal approximation of the line integral in Equation 18 into the simple two-evaluation form used in Equation 19.

Recall that the corrected log-acceptance ratio is

$$\log r_{D_\sigma}(x, y) \;=\; \Delta_{D_\sigma}(x, y) + \log q_\sigma(x \mid y) - \log q_\sigma(y \mid x), \qquad (43)$$

with proposal density $q_\sigma(y \mid x) = \mathcal{N}\big(y;\, D_\sigma(x),\, 2\sigma^2 I\big)$. The Gaussian proposal log-ratio is

$$\log q_\sigma(x \mid y) - \log q_\sigma(y \mid x) \;=\; \frac{1}{4\sigma^2}\Big(\big\|y - D_\sigma(x)\big\|^2 - \big\|x - D_\sigma(y)\big\|^2\Big). \tag{44}$$

Writing $r_\sigma(x) := D_\sigma(x) - x$, expand each squared norm:

$$\|y - D_\sigma(x)\|^2 = \|(y-x) - r_\sigma(x)\|^2 \;=\; \|y-x\|^2 - 2\,\langle y - x,\, r_\sigma(x)\rangle + \|r_\sigma(x)\|^2, \tag{45}$$

$$\|x - D_\sigma(y)\|^2 = \|(x-y) - r_\sigma(y)\|^2 \;=\; \|y-x\|^2 - 2\,\langle x - y,\, r_\sigma(y)\rangle + \|r_\sigma(y)\|^2. \tag{46}$$

Subtracting and using $\langle x-y,\, r_\sigma(y)\rangle = -\langle y-x,\, r_\sigma(y)\rangle$,

$$\|y - D_\sigma(x)\|^2 - \|x - D_\sigma(y)\|^2 \;=\; -2\,\langle y-x,\, r_\sigma(x) + r_\sigma(y)\rangle + \|r_\sigma(x)\|^2 - \|r_\sigma(y)\|^2. \tag{47}$$

Substituting Equation 47 into Equation 44 and combining with the trapezoidal approximation $\Delta_{D_\sigma}(x,y) \approx \frac{1}{2\sigma^2}\,\langle r_\sigma(x) + r_\sigma(y),\, y - x\rangle$, the inner-product terms cancel and only the squared-residual difference survives:

$$\log r_{D_\sigma}(x,y) \;\approx\; \frac{1}{4\sigma^2}\Big(\|D_\sigma(x) - x\|^2 - \|D_\sigma(y) - y\|^2\Big), \tag{48}$$

which is Equation 19. The sampler therefore needs only two evaluations of D_σ per step, one at the current state and one at the proposal.

C.4 Proof of Proposition 2

We prove Proposition 2 using a standard Wasserstein perturbation argument (Rudolf & Schweizer, 2018), and then give one-step estimates for the two mathematical kernels. Here $P^\star$ denotes the ideal kernel called P in the main text. For $r \geq 1$, W_r is the Wasserstein distance of order r on laws with finite rth moments, and $(\mu P)(A) = \int P(x, A)\,\mu(dx)$.

C.4.1 A perturbed-contraction lemma

Assumption 1 (Local contraction of the ideal kernel)**.** *Let $P^\star$ be a Markov kernel with invariant law π, and let $\mathcal{B}_R = \{\mu : W_r(\mu, \pi) \leq R\}$. There exists $\rho \in [0, 1)$ with*

$$W_r(\mu P^\star, \pi) \leq \rho W_r(\mu, \pi) \qquad \textit{for every } \mu \in \mathcal{B}_R.$$

Assumption 2 (One-step implementation error)**.** *The implemented kernel $\widehat{P}$ satisfies $W_r(\mu\widehat{P}, \mu P^\star) \leq \gamma$ for all $\mu \in \mathcal{B}_R$.*

Lemma 1 (Perturbed contraction)**.** *Under Assumptions 1 and 2 with $\gamma \leq (1-\rho)R$, for every $\mu_0 \in \mathcal{B}_R$ the iterates $\mu_k = \mu_0 \widehat{P}^{\,k}$ remain in $\mathcal{B}_R$ and*

$$W_r\big(\mu_K, \pi\big) \;\leq\; \rho^K\, W_r(\mu_0, \pi) \;+\; \gamma\,\frac{1-\rho^K}{1-\rho} \qquad (K \geq 0), \tag{49}$$

which is equation 28 of the main text. Moreover, any invariant law $\widehat{\pi} \in \mathcal{B}_R$ of $\widehat{P}$ satisfies $W_r(\widehat{\pi}, \pi) \leq \gamma/(1-\rho)$.

Proof. Write $d_k = W_r(\mu_k, \pi)$ and use $\pi P^\star = \pi$:

$$d_{k+1} = W_r\big(\mu_k\widehat{P},\, \pi P^\star\big) \;\leq\; \underbrace{W_r\big(\mu_k\widehat{P}, \mu_k P^\star\big)}_{\leq\,\gamma\ \text{(Assumption 2)}} + \underbrace{W_r\big(\mu_k P^\star, \pi P^\star\big)}_{\leq\,\rho\, d_k\ \text{(Assumption 1)}}.$$

If $d_k \leq R$ then $d_{k+1} \leq \gamma + \rho R \leq R$ by the standing condition $\gamma \leq (1-\rho)R$, so the recursion is well defined; unrolling $d_{k+1} \leq \gamma + \rho d_k$ yields equation 49. For an invariant $\widehat{\pi} \in \mathcal{B}_R$, $W_r(\widehat{\pi}, \pi) = W_r(\widehat{\pi}\widehat{P}, \pi P^\star) \leq \gamma + \rho\, W_r(\widehat{\pi}, \pi)$, and rearranging gives the bound. □

The accumulated-error term is $\gamma \sum_{j=0}^{K-1} \rho^j$; its upper bound grows approximately as $K\gamma$ when $K(1-\rho)$ is small and tends to $\gamma/(1-\rho)$. The initialization term vanishes when the chain starts with exactly its own invariant law, $\mu_0 = \pi$. For predictor–corrector this means $\mu_0 = p$; for dMALA it means $\mu_0 = p_\sigma$. Starting dMALA from clean data distributed as p retains the term $\rho^K W_1(p, p_\sigma)$. Under the coupling $X \sim p$, $Z \sim \mathcal{N}(0, I)$ independent,

$$W_1(p, p_\sigma) \leq \sigma\, \mathbb{E}\|Z\| \leq \sigma\sqrt{d}.$$

This example requires finite first moments and the initialization condition of Proposition 2. It does not identify an empirical finite set of retrieved states with an exact target law.

C.4.2 SPECIALIZATIONS TO THE TWO KERNELS

We use $P^\star$ for the ideal kernel denoted by P in the main text.

Predictor–corrector. Let $K_\tau^\star$ use the exact map $\Phi_{\tau\to 1}$ and let $\widehat{K}_\tau$ use an approximate return map $\widehat{\Phi}_{\tau\to 1}$, including any numerical integration error. Couple both updates using the same $X \sim \mu$ and independent $Z \sim \mathcal{N}(0, I)$, and write $\widehat{X}_\tau = \tau X + (1-\tau)Z$. If

$$\sup_{\mu\in\mathcal{B}_R} \left(\mathbb{E} \left\| \widehat{\Phi}_{\tau\to 1}(\widehat{X}_\tau) - \Phi_{\tau\to 1}(\widehat{X}_\tau) \right\|^2 \right)^{1/2} \le \varepsilon_\Phi,$$

then $W_2(\mu\widehat{K}_\tau, \mu K_\tau^\star) \le \varepsilon_\Phi$ for every $\mu \in \mathcal{B}_R$. A uniform pointwise map-error bound of ε_Φ is sufficient for this condition. Thus the one-step assumption holds with $\gamma = \varepsilon_\Phi$.

Let N_τ denote the noising kernel $x \mapsto \tau x + (1-\tau)Z$. Coupling two initial laws and using the same Z gives

$$W_2(\mu N_\tau, \nu N_\tau) \le \tau W_2(\mu, \nu).$$

Consequently, if the exact return map is L_Φ-Lipschitz,

$$W_2(\mu K_\tau^\star, \nu K_\tau^\star) \le \tau L_\Phi W_2(\mu, \nu).$$

When $\tau L_\Phi < 1$, we may use $\rho = \tau L_\Phi$ in Proposition 2. Otherwise this particular estimate does not establish strict contraction. Proposition 1 supplies the invariant law p independently of this contraction test.

Gaussian-proposal dMALA. The next lemma concerns a deterministic denoiser and the additive Gaussian proposal, not the stochastic proposal and normalization used in the ImageNet variant. It makes the required moment bounds explicit.

Lemma 2 (One-step error for the Gaussian-proposal kernel)**.** *Let $\pi = p_\sigma$ be a positive density with globally L-Lipschitz score $s = \nabla \log \pi$, and put $h = \sigma^2 > 0$. Let $\widehat{s}$ be a continuous deterministic score approximation satisfying $\sup_x \|\widehat{s}(x) - s(x)\| \le \varepsilon \le 1$. Define*

$$D(x) = x + hs(x), \qquad \widehat{D}(x) = x + h\widehat{s}(x),$$
$$q^\star(y \mid x) = \mathcal{N}(y; D(x), 2hI), \quad \widehat{q}(y \mid x) = \mathcal{N}(y; \widehat{D}(x), 2hI).$$

The ideal kernel $P^\star$ uses $q^\star$ and the exact Metropolis–Hastings log-ratio for π. The approximate kernel P_{G} uses $\widehat{q}$ and the log-ratio

$$\widehat{a}(x, y) = T_{\widehat{s}}(x, y) + \log \widehat{q}(x \mid y) - \log \widehat{q}(y \mid x),$$
$$T_{\widehat{s}}(x, y) = \tfrac{1}{2}\langle \widehat{s}(x) + \widehat{s}(y), y - x\rangle.$$

This is the trapezoidal acceptance rule, equivalently the squared-residual rule derived in Section 2.2. For a starting law μ with finite second moment, take $X \sim \mu$ and independent $Z \sim \mathcal{N}(0, I)$, and set

$$Y^\star = D(X) + \sqrt{2h}\, Z, \qquad \widehat{Y} = \widehat{D}(X) + \sqrt{2h}\, Z, \qquad V = \|Y^\star - X\|.$$

Let

$$Q = \left| T_{\widehat{s}}(X, \widehat{Y}) - \int_0^1 \langle \widehat{s}(X + t(\widehat{Y} - X)), \widehat{Y} - X\rangle \, dt \right|,$$

and assume the following quantities are finite:

$$M_\mu = (\mathbb{E}V^2)^{1/2}, \qquad H_\mu = (\mathbb{E}\|s(Y^\star)\|^2)^{1/2}, \qquad \delta_\mu = (\mathbb{E}Q^2)^{1/2}.$$

Define $c = 2 + hL$, $a = 1 + c/2$, $b = 1 + hL/2 + c^2/4$, and

$$C_\mu = h + aM_\mu^2 + haM_\mu H_\mu + hbM_\mu.$$

Then

$$W_1(\mu P_{\mathrm{G}}, \mu P^\star) \le C_\mu \varepsilon + M_\mu \delta_\mu.$$

Proof. Use the same X, Z for both proposals and the same independent uniform random variable for their acceptance decisions. The proposal difference satisfies $\|\widehat{Y} - Y^\star\| \le h\varepsilon$. Let $a^\star(X, Y^\star)$ be the exact log-ratio. The difference between the trapezoidal target term and the exact target term is at most

$$Q + \varepsilon V + h\varepsilon^2 + h\varepsilon\|s(Y^\star)\| + \tfrac{1}{2}Lh^2\varepsilon^2.$$

Here the first two error sources are quadrature and score error along the integration segment. The last two terms follow by integrating the L-Lipschitz exact score between $Y^\star$ and $\widehat{Y}$.

The forward Gaussian residuals are identical: $\widehat{Y} - \widehat{D}(X) = Y^\star - D(X) = \sqrt{2h}\, Z$. For the reverse residuals,

$$\|\widehat{D}(\widehat{Y}) - D(Y^\star)\| \le h\varepsilon(2 + hL) = hc\varepsilon, \qquad \|X - D(Y^\star)\| \le V + h\|s(Y^\star)\|.$$

Expanding the squared reverse residuals therefore bounds their contribution to the log-ratio difference by

$$\frac{c}{2}\varepsilon\big(V + h\|s(Y^\star)\|\big) + \frac{hc^2}{4}\varepsilon^2.$$

Combining the bounds and using $\varepsilon \le 1$ gives

$$|\widehat{a}(X, \widehat{Y}) - a^\star(X, Y^\star)| \le Q + \varepsilon\big(aV + ha\|s(Y^\star)\| + hb\big).$$

The function $u \mapsto \min(1, e^u)$ is 1-Lipschitz. Under the shared uniform random variable, the conditional probability of different acceptance decisions is the absolute difference of the two acceptance probabilities. The output distance is bounded by $h\varepsilon$ plus V when the decisions differ. Hence

$$\begin{aligned}\mathbb{E}\|\widehat{X}^+ - X^{\star+}\| &\le h\varepsilon + \mathbb{E}[VQ] + \varepsilon a\mathbb{E}V^2 \\ &\quad + \varepsilon ha\mathbb{E}[V\|s(Y^\star)\|] + \varepsilon hb\mathbb{E}V \\ &\le C_\mu\varepsilon + M_\mu\delta_\mu.\end{aligned}$$

by Cauchy–Schwarz. This coupling bounds the Wasserstein distance. □

To obtain a uniform γ for Proposition 2, the quantities M_μ, H_μ, and δ_μ must be bounded uniformly over all laws to which that proposition is applied. If their bounds are M, H, δ, respectively, then

$$\gamma \le \big(h + aM^2 + haMH + hbM\big)\varepsilon + M\delta.$$

These are additional moment and quadrature assumptions; membership in a W_1 ball alone does not imply them. A uniform per-proposal quadrature bound of δ implies the root-mean-square bound used here. The estimate is for P_{G}; the next remarks distinguish it from the ImageNet implementation.

C.4.3 Exact instantiation and scope of the hypotheses

Example 1 (Gaussian target: exact contraction rate)**.** *Let $p = \mathcal{N}(0, I_d)$ under the linear bridge equation 21. Then $p_\tau = \mathcal{N}\big(0, s_\tau^2 I\big)$ with $s_\tau^2 = \tau^2 + (1-\tau)^2$, and the exact probability-flow map is the linear rescaling $\Phi_{\tau\to 1}(\hat{x}) = \hat{x}/s_\tau$ (solve $\dot{x} = \frac{d}{dt}(\log s_t)\, x$). The ideal kernel is the AR(1) map $x^+ = \frac{\tau}{s_\tau}x + \frac{1-\tau}{s_\tau}z$, whose stationary law is exactly $\mathcal{N}(0, I)$ and whose W_2-contraction rate is*

$$\rho(\tau) = \frac{\tau}{\sqrt{\tau^2 + (1-\tau)^2}} < 1 \quad \textit{for all } \tau \in (0, 1),$$

e.g. $\rho(0.70) \approx 0.92$, $\rho(0.85) \approx 0.98$, $\rho(0.95) \approx 0.999$. Assumption 1 thus holds globally with an explicit rate in this case, and the rate degrades monotonically as $\tau \to 1$, consistent with the smaller mean displacement and slower mixing observed at larger τ in Table 5.

Remark 1 (Locality of the contraction hypothesis). The set $\mathcal{B}_R$ is a neighborhood of π in the space of probability laws, not a bounded region of sample space. Restricting initialization to this set does not by itself prove contraction or bound all proposal moments. Example 1 verifies the contraction assumption for the Gaussian predictor–corrector kernel. For the multimodal targets considered experimentally, contraction remains a hypothesis of Proposition 2; it is not inferred from sample quality or from decreasing FID increments.

Remark 2 (Diagnostics and the constants). The diagnostics in Section E.12 examine the extracted field and the quadrature rule. Jacobian asymmetry diagnoses local non-conservativity. Comparing quadrature rules probes numerical sensitivity; comparison with exact log-density differences assesses the combined score and integration error. Neither diagnostic supplies a uniform score-error bound or estimates γ and ρ on ImageNet. In particular, the Wasserstein estimate is not an FID bound.

Remark 3 (Deployed proposal variant). Write P_{G} for the deterministic-denoiser Gaussian-proposal kernel of Lemma 2, and P_{img} for the ImageNet variant in Algorithm 1. The latter uses a stochastic flow-based proposal and per-class normalization. For each starting law μ, the triangle inequality gives

$$W_1(\mu P_{\mathrm{img}}, \mu P^\star) \le W_1(\mu P_{\mathrm{img}}, \mu P_{\mathrm{G}}) + W_1(\mu P_{\mathrm{G}}, \mu P^\star).$$

Lemma 2 bounds the second term under its hypotheses. It does not bound the first term, which includes the proposal change, internal randomness, and normalization. Proposition 2 applies to P_{img} conditionally on a bound for this total one-step error and on the stated contraction assumption.

Remark 4 (The τ trade-off, quantified). For fixed K, the accumulation factor $S_K(\rho) = \sum_{j=0}^{K-1} \rho^j$ increases with ρ and tends to K as $\rho \to 1$. In Example 1, the contraction factor increases as τ approaches the data endpoint. The return-map error may also depend on τ, so the relevant bound is $\varepsilon_\Phi(\tau) S_K(\rho(\tau))$. No monotonic dependence of ε_Φ on τ follows from invariance alone.

C.5 Training-time regularizer

In order to encourage a conservative score field, we use the classical fact from vector calculus (Marsden & Tromba, 2011) that if $s : \Omega \subset \mathbb{R}^d \to \mathbb{R}^d$ is C^1, Ω is simply connected, and its Jacobian $\nabla s(x)$ is symmetric for every $x \in \Omega$, then s is a gradient field; that is, there exists a scalar potential ϕ such that

$$s(x) = \nabla \phi(x).$$

Since

$$J_{D_\sigma}(x) = I + \sigma^2 J_{s_\sigma}(x), \tag{50}$$

the denoiser Jacobian $J_{D_\sigma}(x)$ is symmetric if and only if the score Jacobian $J_{s_\sigma}(x)$ is symmetric. Therefore, to encourage a conservative score field, it is sufficient to regularize the denoiser Jacobian. We do so through

$$\mathcal{L}^D_{\mathrm{sym}}(x) = \mathbb{E}_v \left\| J_{D_\sigma}(x) v - J_{D_\sigma}(x)^\top v \right\|^2, \tag{51}$$

which is numerically more stable than regularizing the score directly.

C.6 Illustration on the Swiss roll

Example 2 (ULA versus corrected Langevin (MALA) on a Swiss-roll-like target)**.** *To illustrate the need for correction, we consider a two-dimensional Swiss-roll-like target density and initialize particles from the target distribution. We then evolve them using either the unadjusted Langevin algorithm (ULA) or the corrected dynamics above. Since both methods start from $x_0 \sim p$, an ideal probability-preserving process should satisfy $x_k \sim p$ for all k. In practice, ULA drifts away from the target, while the corrected dynamics remains close to the original distribution. See Figure 2.*

D Experimental Details

D.1 Datasets

Oxford Flowers-102. We use the Oxford 102 Category Flower dataset (Nilsback & Zisserman, 2008), which comprises 8,189 images spanning 102 fine-grained flower species. All three official splits (train, validation, and test) are merged into a single evaluation pool, following prior unconditional generation work. All images are resized to 256×256 pixels.

ImageNet-256. For large-scale experiments we use ImageNet 2012 (Deng et al., 2009) at 256×256 resolution with 1,000 classes. Generation operates in the latent space of a variational autoencoder (Rombach et al., 2022), following the convention of recent latent generative models such as REPA (Yu et al., 2024).

D.2 Generative Models

All three backbones share the same variational autoencoder (Rombach et al., 2022), a KL-regularized encoder–decoder trained to produce 4-channel latents at $1/8$ the spatial resolution of the input image (i.e. 32×32 for 256×256 inputs). Latents are scaled by $\lambda = 0.18215$ before and after the generative model, following the convention of the Stable Diffusion codebase.

D.2.1 Oxford Flowers-102

For the Flowers-102 experiments we train a latent flow matching model from scratch using a U-Net (Ronneberger et al., 2015) that operates directly on the 4-channel latent tensors. The architecture is summarized in Table 3. The model is trained with a cosine-path interpolant

$$x_t = \cos\left(\tfrac{\pi}{2}t\right)\, x + \sin\left(\tfrac{\pi}{2}t\right)\, \varepsilon, \qquad x \sim p,\ \varepsilon \sim \mathcal{N}(0, I), \tag{52}$$

so that $t = 0$ corresponds to pure data and $t = 1$ to pure noise. Generation from noise integrates the learned velocity backward from $t = 1$ to $t = 0$ using the second-order Runge–Kutta (RK2) integrator (Butcher, 2016).

Table 3: Flow Matching U-Net architecture (Flowers-102).

Hyperparameter	Value
Latent channels	4
Latent resolution	32×32
Residual blocks per level	3
Channel multipliers	$[1,\ 2,\ 2,\ 2]$
Attention heads	4
Attention resolutions	16×16
Dropout	0.1
Interpolant	Cosine path
ODE integrator	RK2, $N_{\mathrm{ode}} = 10$

D.2.2 ImageNet-256

SoFlow (Luo et al., 2025) is a class-conditional latent flow-matching model built on the DiT-XL/2 backbone (Peebles & Xie, 2023). We use the publicly available checkpoint trained for 1.2×10^6 gradient steps with EMA weights. In the noise-to-data convention used in Equation (21), we write its linear interpolant as

$$x_t = (1 - t)z + tx, \qquad t \in [0, 1]. \tag{53}$$

The model operates on 32×32 four-channel latents conditioned on 1,000 ImageNet classes.

We additionally use the SiT-XL/2 checkpoint trained with REPA (Yu et al., 2024), a training objective that aligns a DiT's internal representations with a frozen vision encoder. We generate $50{,}000$ samples and evaluate using the REPA evaluator script. For both backbones, all generations reported in this paper are produced without classifier-free guidance.

D.3 Implementation Details

All experiments for Oxford Flowers-102 ran on a single NVIDIA RTX 4090 GPU. ImageNet-256 experiments used two NVIDIA RTX 4090 GPUs and one H100 GPU. We did not retrain or finetune any of the pretrained ImageNet backbones (SoFlow-XL/2-cond, SiT-REPA-XL); the 50,000-sample evaluation runs are the only ImageNet-scale compute consumed by our framework.

E Additional Experimental Results

E.1 Swiss-roll ablation tables

The two ablations referenced in Section 3.1: dMALA on the smoothed law p_σ and the predictor–corrector kernel on p, both for 200 steps.

Table 4: SoFlow-denoiser MALA ablation across smoothing levels σ. Each configuration is evaluated against the corresponding smoothed law p_σ.

σ	Steps	Acc. rate	NLL to p_σ	MMD to p_σ	Mean move
0.111	200	0.896	2.912	0.016	1.508
0.053	200	0.960	2.386	0.013	0.813
0.020	200	0.989	1.733	0.017	0.370

E.2 Oxford Flowers-102

We evaluate our approach on the Oxford Flower-102 dataset (Nilsback & Zisserman, 2008) (training details are in D.2.1). Table 6 compares the Predictor–Corrector and dMALA samplers against a standard flow-matching baseline generated from Gaussian noise, reporting Fréchet Inception Distance (FID), Precision, and Recall across step counts.

Table 5: Predictor–corrector flow ablation. This method targets the original law p.

tau	steps	NLL to p	MMD to p	Mean move
0.70	200	8.994	0.057	2.811
0.85	200	3.378	0.025	1.633
0.95	200	1.760	0.020	0.568

Table 6: FID, Precision, and Recall on Oxford Flowers-102, with $t_{\text{noise}} = 0.1$.

Method	**Steps**	**FID** $\downarrow$	**Precision** $\uparrow$	**Recall** $\uparrow$
Baseline FM (noise $\rightarrow$ data)	–	23.95	0.5213	0.2761
Predictor–Corrector	20	12.50	0.6013	0.6962
	40	15.90	0.4676	0.6313
	60	19.92	0.3859	0.5848
	80	25.38	0.3091	0.5524
dMALA	20	**10.92**	**0.6967**	**0.7825**
	40	12.64	0.5940	0.6936
	60	14.45	0.5179	0.6537
	80	16.34	0.4587	0.6210

E.3 ImageNet-256: dMALA Ablation

Table 7 reports the dMALA sampler on ImageNet-256 with the SoFlow-XL/2-cond backbone, sweeping the smoothing level σ at $\sigma \in \{0.3, 0.5\}$ and the number of Metropolis–Hastings iterations K. Smaller σ raises the acceptance rate and yields lower FID; larger σ trades FID against higher Inception Score and Precision, at the cost of Recall.

Table 7: dMALA on ImageNet-256 with SoFlow-XL/2-cond. The denoiser uses SoFlow's solution operator with noise level σ. K denotes the number of Metropolis–Hastings iterations; Acc. rate is the running mean acceptance probability. Metrics are computed on $50{,}000$ samples against the ADM reference batch. Reference baseline shown in gray.

	K	Acc. rate	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
Baseline (1-NFE)	–	–	2.96	4.60	238.53	0.80	0.56
$\sigma = 0.5$	50	0.68	5.50	6.57	356.05	0.87	0.47
	100	0.49	7.28	8.18	367.45	0.89	0.42
	150	0.49	8.29	9.49	372.80	0.89	0.39
	200	0.48	9.02	10.32	377.30	0.90	0.38
$\sigma = 0.3$	50	0.76	3.23	7.72	313.29	0.81	0.57
	100	0.54	4.34	10.40	308.11	0.82	0.52
	150	0.53	5.07	12.45	301.23	0.81	0.49
	200	0.53	5.61	14.26	294.47	0.81	0.47

E.4 Computational Efficiency

In the next experiment we test the computational efficiency of the methods and compare them for a transport generation that has a similar budget. Results are shown in Table 8.

Table 8: Predictor–corrector sampling matches or beats direct flow integration at equal or smaller compute on ImageNet-256 with SiT-REPA-XL. The baseline rows integrate the learned velocity field over the full noise-to-data path using forward Euler with the indicated number of steps. Each predictor–corrector iteration instead integrates only its partial return segment using n_{steps} Euler steps. The total cost is NFE $= n_{\text{steps}}K$ for predictor–corrector and $2n_{\text{steps}}K$ for dMALA. Predictor–corrector uses $t_{\text{noise}} = 0.5$ and dMALA uses $\sigma = 0.3$. Metrics use $50{,}000$ samples against the ADM reference batch.

	NFE	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
Baseline (400 Euler steps)	400	6.61	5.62	152.52	0.68	0.69
Baseline (800 Euler steps)	800	6.51	5.58	153.08	0.68	0.69
Pred.–Corr n_{steps}=10, K=20 (ours)	200	4.91	12.07	183.82	0.70	0.66
Pred.–Corr n_{steps}=20, K=20 (ours)	400	3.06	6.24	199.50	0.72	0.69
dMALA n_{steps}=10, K=10 (ours)	200	1.93	5.62	241.20	0.74	0.70
dMALA n_{steps}=20, K=20 (ours)	800	1.79	5.14	236.03	0.74	0.71

E.5 Long-chain behavior

Table 9 reports the long-chain study referenced in Section 3.2. The per-50-step FID increments shrink ($+1.11$, $+0.73$, $+0.54$ for dMALA at $\sigma = 0.3$; $+1.22$, $+0.75$, $+0.53$ for the predictor–corrector kernel).

E.6 ImageNet-256: Effect of initialization

Table 10 runs both $K = 10$ SoFlow kernels from three common initializations—training latents, validation latents, and the 1-NFE sampler's own outputs—and additionally runs dMALA from the weak SiT outputs of Table 1. The validation-seed rows are the overlap-free values quoted in Section 3.2.

Table 9: Long-chain behavior of dMALA and Predictor-Corrector on ImageNet-256 with SoFlow-XL/2-cond. dMALA uses denoiser noise level $\sigma = 0.3$; Predictor-Corrector uses bridge noise $t_{\text{noise}} = 0.3$; K denotes the number of iterations. Acc. rate is the running mean Metropolis acceptance probability.

Method	K	Acc. rate	FID ↓	sFID ↓	IS ↑	Precision ↑	Recall ↑
Baseline (1-NFE)	–	–	2.96	4.60	238.53	0.80	0.56
dMALA	50	0.76	3.23	7.72	313.29	0.81	0.57
	100	0.54	4.34	10.40	308.11	0.82	0.52
	150	0.53	5.07	12.45	301.23	0.81	0.49
	200	0.53	5.61	14.26	294.47	0.81	0.47
Pred.–Corr.	50	–	3.96	8.90	313.48	0.82	0.53
	100	–	5.18	12.14	304.34	0.81	0.48
	150	–	5.93	14.64	290.25	0.80	0.46
	200	–	6.46	16.48	278.84	0.79	0.45

Table 10: Effect of initialization on the $K = 10$ SoFlow configurations. The rows use the same sampler settings and differ only in how the initial states are obtained.

Kernel	Seed source	FID ↓	IS ↑	Prec. ↑	Rec. ↑
dMALA, $\sigma = 0.3$	training latents	1.70	300.04	0.79	0.65
Pred.–Corr., $t_{\text{noise}} = 0.3$	training latents	2.28	297.37	0.79	0.62
dMALA	validation latents	2.43	247.1	0.78	0.58
Pred.–Corr.	validation latents	2.54	259.4	0.79	0.56
dMALA	the 1-NFE sampler's own outputs (FID 2.96)	3.04	284.5	0.85	0.52
Pred.–Corr.	the 1-NFE sampler's own outputs (FID 2.96)	3.34	292.1	0.86	0.50
dMALA	SiT 25-step no-CFG outputs (FID 10.70)	3.19	218.2	0.78	0.61

E.7 Additional data-supported controls

Table 11 reports additional controls on the same set of $50{,}000$ starting ImageNet latents, including decoding, Gaussian perturbation, nearest-neighbour retrieval, and the two proposed kernels at $K = 10$.

Table 11: Additional data-supported controls on the same 50,000 starting ImageNet latents, SoFlow-XL/2-cond, ADM reference; LPIPS on the fixed $2{,}048$-seed subset. Nearest-neighbour (NN) retrieval returns each seed's nearest training image (seed excluded), within the seed's class or over the full training set; the unrestricted search returns only $10{,}799$ distinct images across the $50{,}000$ queries and one image $1{,}296$ times (hub collapse), the class-restricted search $31{,}042$. [a] measured on a second run with the same settings.

Method	NFE	FID ↓	sFID ↓	IS ↑	Prec. ↑	Rec. ↑	LPIPS
Decode the seeds	0	1.27	4.00	295.3	0.73	0.73	0.00
Gaussian perturbation of the seeds, std 0.3	0	38.55	36.44	65.0	0.40	0.59	0.29
NN retrieval, same class	0	4.88	14.11	300.1	0.72	0.61	0.67
NN retrieval, unrestricted	0	62.6	128.80	112.2	0.61	0.61	0.74
Pred.–Corr., $K = 10$, $t_{\text{noise}} = 0.3$	10	2.28	5.47	297.37	0.79	0.62	0.29[a]
dMALA, $K = 10$, $\sigma = 0.3$	20	1.70	5.05	300.04	0.79	0.65	0.26

E.8 Exploration against chain length

Table 12 gives the displacement statistics quoted in Section 3.2 for the dMALA runs of Tables 2 and 9 at $K = 10$ to 200: LPIPS and latent distance to the seed, distance to the nearest of the $50{,}000$ starting latents, the fraction of chains whose nearest starting image is still their own initial state, and FID.

Table 12: dMALA ($\sigma = 0.3$) on SoFlow-XL/2-cond from the same $50{,}000$ training latents. LPIPS to the seed on the fixed $2{,}048$-pair subset; ℓ_2 to the seed and the distance to the nearest of the $50{,}000$ seeds (NN dist.) are raw ℓ_2 distances in the $4{,}096$-dimensional SD-VAE latent space, medians over $4{,}096$ chains; "Own seed is NN" is the fraction of chains whose nearest seed pool image is still the chain's own initial state. The reference row is the distance from a seed to its nearest distinct training image (self excluded; $8{,}192$ queries; 98.6% of these pairs are cross-class). FID from Table 2 and Table 9.

K	NFE	LPIPS	ℓ_2	NN dist.	Own seed is NN	FID
10	20	0.26	33.6	33.6	99.8%	1.70
50	100	0.36	45.0	45.0	97.4%	3.23
100	200	0.42	49.0	48.9	89.5%	4.34
200	400	0.48	52.8	52.2	70.1%	5.61
Nearest distinct training image (reference)		0.73	55.6			

E.9 Repeated use of the transition

Table 13 evaluates repeated use of the SoFlow-XL/2-cond-based transitions from weak SiT-generated inputs, using $10{,}000$ samples for each distributional-metric row. The reference, proposal-only, and cross-check rows use the same $10{,}000$ inputs drawn from the weak inputs of Table 1; the diagnostic and dMALA rows are independent runs generated by the same weak-input protocol. The fresh same-class reference is an independent 1-NFE SoFlow sample conditioned on the same ImageNet class as the corresponding weak input. Because FID depends on sample count, FIDs should be compared within the common $10{,}000$-sample protocol; the corresponding $50{,}000$-sample evaluation is reported in Table 1.

Table 13: Repeated use from weak generated ImageNet-256 inputs, with $n = 10{,}000$ for each distributional-metric row. The weak inputs are 25-step SiT Euler samples generated without classifier-free guidance, as in Table 1. The proposal-only baseline repeatedly applies the SoFlow perturb-and-return proposal. The no-Metropolis control uses the same deployed proposal as dMALA but omits the accept/reject step, while dMALA uses the full update. dMALA requires two network evaluations per step, so proposal-only $K = 20, 40$ and dMALA $K = 10, 20$ have matched NFE. LPIPS, DINO, and label agreement are shown only when the required input–output pairing or conditioning information was retained.

			Distribution quality				Input similarity		
Operation	K	NFE	FID ↓	IS ↑	Prec. ↑	Rec. ↑	LPIPS ↓	DINO ↑	Label agr. ↑
Reference rows									
Weak inputs, unchanged	0	0	13.72	130.3	0.662	0.747	0.000	1.000	0.709
Fresh SoFlow sample	–	1	5.80	239.7	0.798	0.650	0.723	0.431	0.868
Repeated proposal-only baseline									
Proposal only	1	1	10.22	161.4	0.696	0.727	0.174	0.910	0.770
	2	2	8.68	182.1	0.732	0.706	0.229	0.873	0.799
	5	5	6.63	219.0	0.775	0.695	0.303	0.815	0.841
	10	10	5.84	255.0	0.810	0.663	0.357	0.768	0.881
	20	20	6.16	292.6	0.845	0.614	0.409	0.721	0.910
	30	30	6.80	308.8	0.854	0.595			
	40	40	7.31	321.0	0.862	0.575			
No-Metropolis control	10	10	5.89	238.9	0.785	0.677			
dMALA	10	20	6.51	215.6	0.771	0.694			
	20	40	5.83	245.9	0.801	0.666			
Cross-check: subset of the Table 1 run									
dMALA	10	20	6.22	217.3	0.771	0.693	0.323	0.799	0.846

E.10 RUN-TO-RUN ERROR BARS

Table 14 lists the three runs behind the "FID, 3 runs" column of Table 2.

Table 14: FID over three complete sampling runs for every kernel configuration of Table 2. Run 0 is the tabulated run; runs 1 and 2 independently redraw the 50,000 starting latents, class labels, and chain randomness. Mean ± sample standard deviation (ddof = 1).

Backbone	Configuration	NFE	Run 0	Runs 1, 2	Mean ± std
SoFlow	dMALA, $K = 10$, $\sigma = 0.3$	20	1.70	1.74, 1.71	1.72 ± 0.02
SoFlow	Pred.–Corr., $K = 10$, $t_{\text{noise}} = 0.3$	10	2.28	2.00, 1.97	2.08 ± 0.17
SoFlow	dMALA, $K = 20$, $\sigma = 0.3$	40	2.17	2.23, 2.19	2.19 ± 0.03
SoFlow	Pred.–Corr., $K = 20$, $t_{\text{noise}} = 0.5$	20	4.67	4.45, 4.47	4.53 ± 0.12
SiT-REPA	dMALA, $K = 10$, $n_{\text{steps}} = 10$	200	1.93	1.93, 1.94	1.93 ± 0.01
SiT-REPA	Pred.–Corr., $K = 10$, $t_{\text{noise}} = 0.3$, $n_{\text{steps}} = 10$	100	3.11	3.15, 3.16	3.14 ± 0.03
SiT-REPA	dMALA, $K = 20$, $n_{\text{steps}} = 20$	800	1.79	1.80, 1.86	1.82 ± 0.03
SiT-REPA	Pred.–Corr., $K = 20$, $t_{\text{noise}} = 0.5$, $n_{\text{steps}} = 20$	400	3.06	3.05, 3.09	3.07 ± 0.02

E.11 BRIDGE-NOISE SWEEP AND TERMINAL DENOISING

Table 15 sweeps the bridge noise of the predictor–corrector kernel at $K = 10$ (left) and applies one explicit denoising step to the final states of the dMALA chain of Table 2 (right); the step leaves the metrics essentially unchanged (FID 1.70 to 1.77, Precision 0.791 in both cases).

Table 15: Left: predictor–corrector kernel on SoFlow-XL/2-cond, $K = 10$, 50,000 samples, bridge noise $t_{\text{noise}} = 1 - \tau$. Right: one explicit denoising step applied to the final states of the dMALA chain of Table 2 ($K = 10$, $\sigma = 0.3$), 50,000 samples.

t_{noise}	NFE	FID	IS	Prec.	Rec.
0.1	10	2.33	273.1	0.75	0.68
0.3	10	2.28	297.37	0.79	0.62
0.5	10	3.12	332.1	0.85	0.54

	FID	sFID	IS	Prec.	Rec.
dMALA, $K = 10$	1.70	5.05	300.0	0.791	0.650
with terminal denoising	1.77	5.05	302.7	0.791	0.642

E.12 APPROXIMATION DIAGNOSTICS

On the Swiss roll we use the pretrained flow of Section 3.1 with proposal noise 0.3 and the flow evaluated at bridge time 0.7, 4,096 chains of 50 steps (204,800 proposal pairs) and exact 2×2 Jacobians at 512 states. The two-point trapezoidal estimate of $\Delta \log p_\sigma$ has Pearson correlation 0.979 with the exact value, an 8-point Gauss rule 0.982, and the two rules correlate 0.974 with each other; the chains' mean NLL under p_σ is 1.090 (trapezoid) and 1.141 (Gauss) against 1.230 for exact samples of p_σ; the median relative Jacobian asymmetry of the extracted denoiser is 0.017. On ImageNet latents we use random-direction probes $\|(J - J^\top)v\|/\|(J + J^\top)v\|$ at 256 states of the dMALA chain of Table 2 ($K = 10$, $\sigma = 0.3$) and 8 directions each, in fp32 with the denoiser's internal noise frozen per state, $J^\top v$ by vector–Jacobian products and Jv by central differences: the median is 0.179 (interquartile range 0.154 to 0.201) for the denoiser and 0.105 for the induced score field. These are the values quoted in Section 2.4.

F BROADER IMPACT

This work develops a sampling framework that operates on top of existing pretrained flow-matching models. Two implications follow. On the positive side, because our framework re-uses existing

checkpoints rather than training new generative models, it can reduce the energy and compute footprint of deploying generative systems, and pairs naturally with retrieval-augmented pipelines that aim to ground generated content in reliable source material. On the negative side, because the framework inherits the behavior of the underlying pretrained model, it also inherits whatever biases, representational gaps, or potential for misuse the underlying model carries, including the risk of being applied to produce deepfakes or targeted misinformation. Practitioners deploying this framework should treat the underlying model's documentation and known failure modes as a baseline that the sampling framework does not modify.

G Sample Visualizations

G.1 ImageNet-256

We provide visualizations of both the Predictor–Corrector and the dMALA samplers on the ImageNet256 dataset (Deng et al., 2009). Figure 4 presents a combined overview comparing both methods across the SoFlow-XL/2 and SiT-XL/2 models over 200 iterations.

To examine the short-term behavior of the samplers, Figure 5 and Figure 6 illustrate the original images alongside samples generated after a 20-step run using dMALA and Predictor–Corrector, respectively.

Furthermore, we provide extended long-run visualizations for Predictor–Corrector in Figure 7 and for dMALA in Figure 8. In these extended subfigures, the first row displays the original image (before any iterations), and each subsequent row corresponds to an additional 50 steps, reaching 200 steps in the final row. Across all experiments, both methods successfully preserve the class identity throughout the chain while progressively moving the sample along the data manifold. To further illustrate the step-by-step progression of the Predictor–Corrector across longer chains, Figure 9 and Figure 10 present generated samples using the SiT-REPA-XL and SoFlow-XL/2-cond backbones, respectively. In these figures, the leftmost column presents the original image, while the subsequent columns track the continuous evolution of the state across steps.



(a) SoFlow-XL/2 (b) SiT-XL/2

Figure 5: dMALA (20 steps). The first row shows the original image; the second row shows the generated sample.

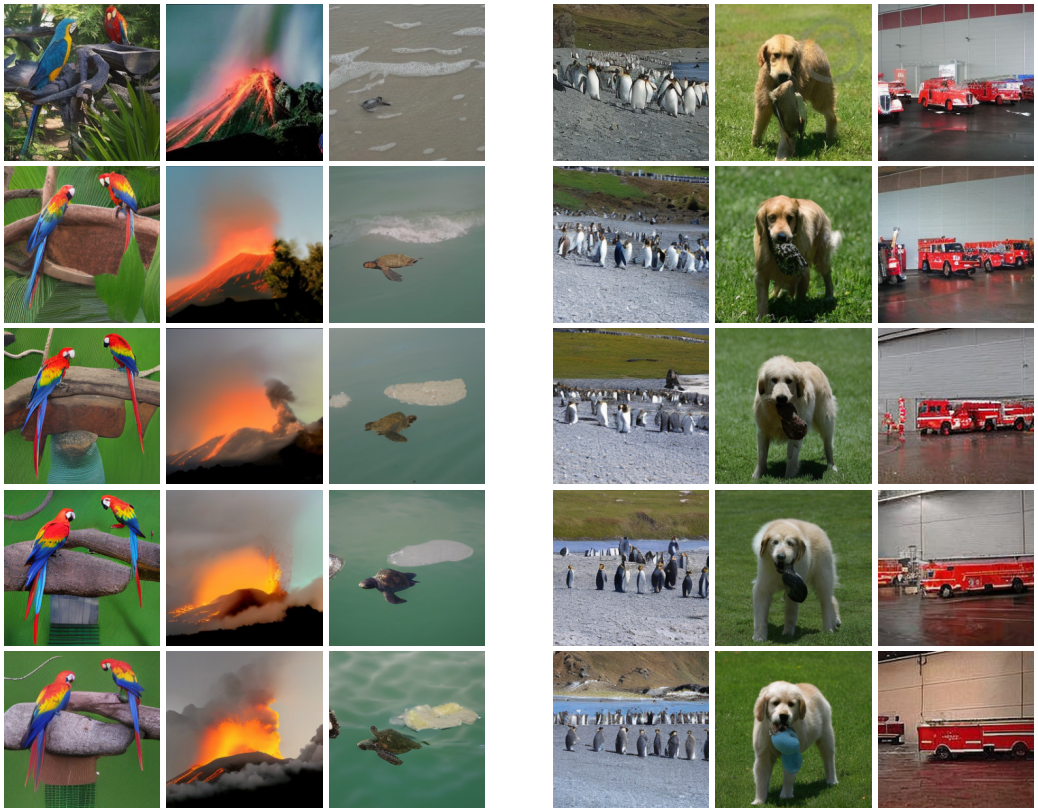
G.2 Oxford Flowers-102 Visualization

We further validate our approach on the Oxford Flowers dataset (Nilsback & Zisserman, 2008). Figure 11 shows results for the Predictor–Corrector and dMALA. In each subfigure the first row shows the original image, and each subsequent row shows the sample after an additional 20 iterations.



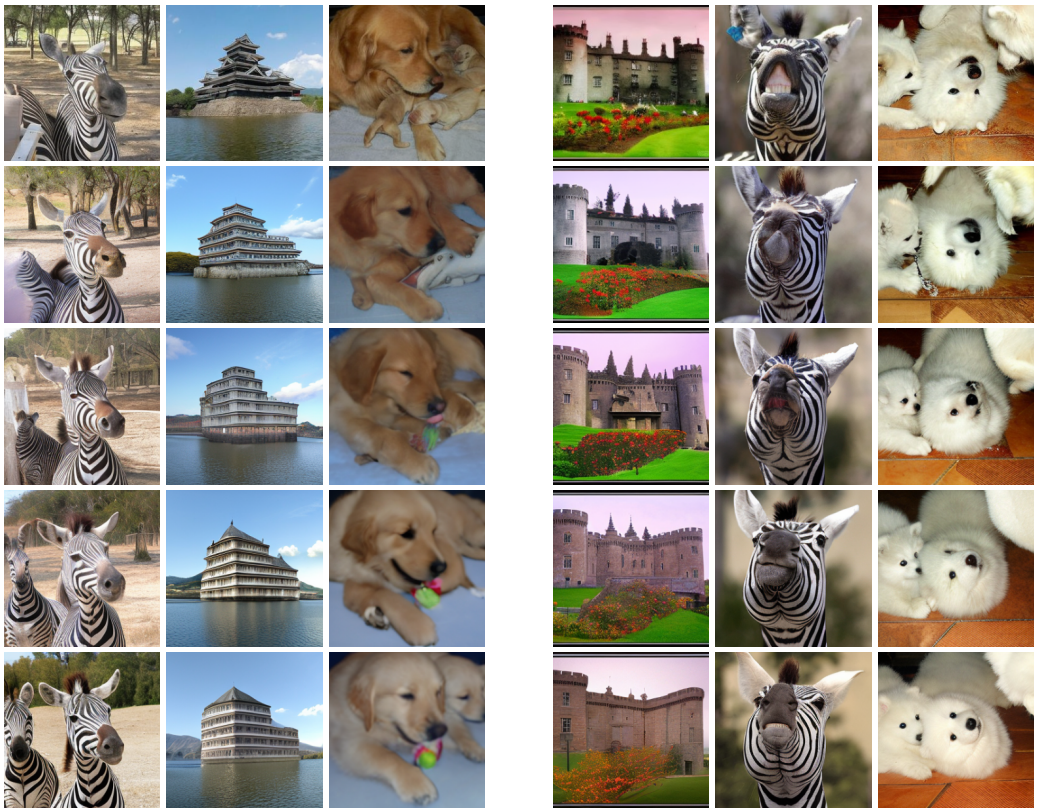
(a) SoFlow-XL/2 (b) SiT-XL/2

Figure 6: Pred.–Corr. (20 steps). The first row shows the original image; the second row shows the generated sample.



(a) SoFlow-XL/2 (b) SiT-XL/2

Figure 7: Predictor–Corrector on ImageNet256 over 200 iterations. The first row shows the original image; each subsequent row shows the sample after an additional 50 steps, up to a total of 200 steps.



(a) SoFlow-XL/2 (b) SiT-XL/2

Figure 8: dMALA on ImageNet256 over 200 iterations. The first row shows the original image; each subsequent row shows the sample after an additional 50 steps, up to a total of 200 steps.

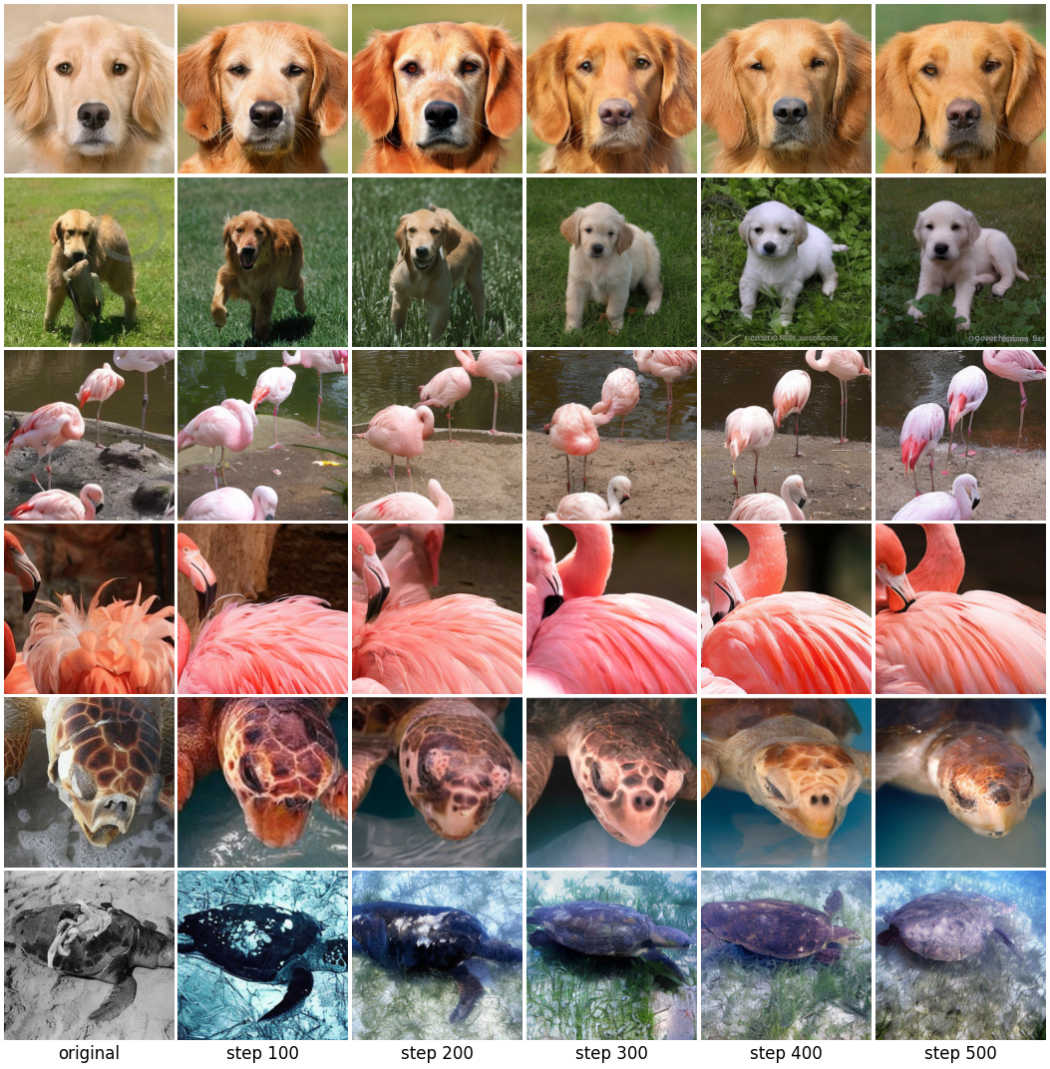


Figure 9: Pred.–Corr. ImageNet-256 samples generated with the SiT-REPA-XL backbone. The leftmost column shows the original image; subsequent columns show the state after the indicated number of probe iterations.

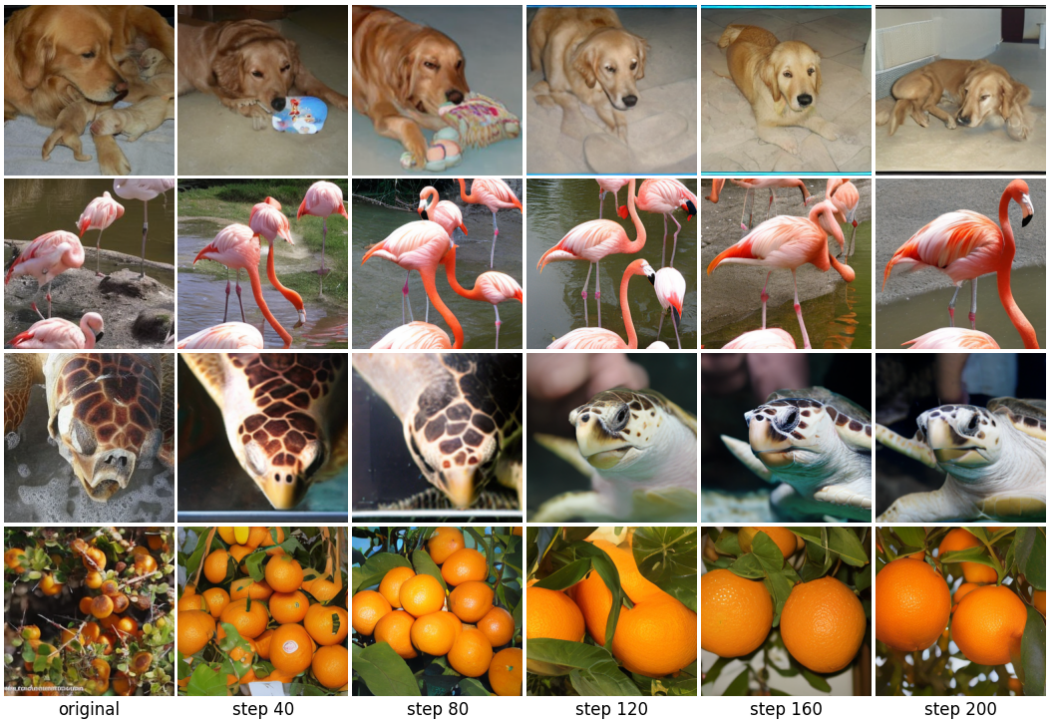


Figure 10: Pred.–Corr. ImageNet-256 samples generated with the SoFlow-XL/2-cond backbone. The leftmost column shows the original image; subsequent columns show the state after the indicated number of probe iterations.



(a) Predictor–Corrector (b) dMALA

Figure 11: Predictor–Corrector and dMALA on Oxford Flowers-102 over 80 iterations. The first row shows the original image; each subsequent row shows the sample after an additional 20 iterations, up to a total of 80 steps.