Co-Evolving Paths and Flows via Path–Flow Alignment

Zeyu Michael Li[*] Duke University
William Xingxu Chen[*] Duke University
Xiang Cheng Duke University

Abstract

We study path–flow alignment as a unified training objective for flow matching. Instead of fixing the interpolation path and learning only the velocity field, we jointly train an endpoint-preserving path network and a flow network using the same alignment loss: the flow learns to match the path velocity, and the path learns to align its velocity to the current flow. Although every fixed learned path defines a valid flow-matching objective, the alignment loss alone is not a reliable criterion for path learning. We identify *path overfitting*, a failure mode in which the alignment loss decreases while sample quality worsens. We find that this failure is associated with low-entropy bottlenecks in the induced probability path, where the learned path routes samples through overly concentrated intermediate marginals. Motivated by this diagnosis, we introduce a stochastic path regularizer that hides part of the source information from the path network while preserving exact endpoints. The resulting regularization gives an explicit entropy floor for the stochastic training-path marginals and empirically suppresses the bottleneck in the learned sampler, making joint path–flow training effective. On ImageNet-256×256 with SiT backbones, our method consistently improves FID across model scales, extends to model-guidance training, and leaves the inference-time architecture and sampler unchanged. Code is available at `https://github.com/lizeyu090312/traj_opt_paper`.

1 Introduction

Flow matching learns a generative ODE through a supervised regression problem: sample a source point and a data point, interpolate between them, and train a velocity field to match the tangent of that interpolation. This view underlies flow matching and stochastic interpolants [11, 2, 1], and scales to modern image backbones such as SiT [13]. In most large-scale image models this interpolation is fixed in advance, often as the linear path

$$\gamma_{\mathrm{lin}}(t; x_0, x_1) = (1-t)x_0 + tx_1.$$

The interpolating path is not a passive implementation detail. It determines both the states on which the model is trained and the vector targets it is asked to predict. Prior work has improved flow matching by changing endpoint couplings, straightening trajectories, using metric geometry, or learning curved paths [12, 19, 15, 8, 17]. We ask a complementary question:

What makes a path good, and can we learn it jointly with the flow?

A natural answer is to *use the flow-matching objective itself as a self-supervising signal for both the path and the flow* – a good path should make the flow-matching regression problem easier. Motivated by this, we propose *path–flow alignment*: we parameterize a path network γ_ψ, and jointly train the path network and the flow network on the same path–flow alignment loss.

[*]These authors contributed equally to this work.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

For any fixed learned path, we verify that this remains a valid flow-matching problem. Empirically, we observe that this approach quickly improves the flow-matching loss. However, the generation quality soon begins to degrade even as flow-matching loss continues to improve.

We call this *path overfitting*. In our sweeps, such overfitting is consistently accompanied by *entropy collapse* of intermediate marginals – intuitively, the path network γ_ψ is routing most of the flow through a series of low-entropy bottlenecks. This proves our first answer to be incomplete: a good path should not only reduce flow-matching loss, but *do so while avoiding entropy collapse.*

Motivated by this diagnosis, we introduce a simple regularization scheme that applies a Gaussian perturbation to the input of the path network. Theoretically, we show that, under the independent endpoint coupling used here, this regularization establishes an entropy floor on the stochastic training-path marginals. Empirically, we show that this stochastic-regularized path–flow alignment effectively improves the generation quality on the challenging image generation tasks. We also show that the same path–flow alignment template extends beyond standard flow matching by applying it to model-guidance training [18]. Overall, our results demonstrate the promise of joint training of path and flow networks, assuming proper regularization.

1.1 Contributions

Our main contributions are as follows.

1. **Learning interpolation paths for flow matching.** We introduce *path–flow alignment* (Algorithm 1), a training-time method for learning endpoint-preserving neural paths jointly with a flow model. Section 3 defines the construction; Proposition 1 gives the correctness guarantee.
2. **Identifying Path overfitting and low entropy bottlenecks.** We identify a failure mode of naive path–flow alignment, which we call *path overfitting*. As the flow-matching loss continues to improve, generation quality initially improves but can later degrade. Sections 3.5 and 4.1 empirically associate this failure with a phenomenon that we call a *low-entropy bottleneck*; Proposition 2 gives an exact entropy-rate interpretation of the divergence diagnostic.
3. **Stochastic Gaussian regularization with entropy guarantees.** We design a simple regularization scheme based on perturbing the source input seen by the learned path network. Under independent endpoint coupling, Theorem 1 lower-bounds the entropy of the stochastic training-path marginals. Proposition 3 further lower bounds their instantaneous rate of entropy decrease.
4. **ImageNet-scale validation and extension to model guidance.** On ImageNet-256×256 with SiT backbones, our method consistently improves FID across model sizes while keeping sampling cost unchanged (Table 1). Our alignment template also composes with model-guidance [18] training, and shows significant improvement over the model-guidance baseline (Table 1).

2 Related Work

Flow matching trains continuous-time generative models by regressing a velocity field to tangents of a prescribed probability path [11]. Conditional flow matching, optimal-transport variants, and stochastic interpolants show that the path, coupling, and induced velocity target are central modeling choices [19, 2, 1]. Our work stays within this framework, and focuses on the concrete question of how to train the path. Prior work improves flow matching by changing endpoint couplings or weights [19, 15, 4, 10], straightening trajectories through reflow [12], or replacing Euclidean lines with geometric paths [8]. Closest to us are methods that learn paths or forward processes, including learned curved paths for straighter flows [17], Neural Flow Diffusion Models [3], ALI-CFM for observed multi-marginal snapshots [9], and Curly-FM for non-gradient dynamics from population snapshots [14]. Our method differs in purpose and use: after endpoints are sampled, we learn an endpoint-conditioned residual path as a training-time supervision mechanism. Rather than optimizing straightness, likelihood, or matching observed intermediate marginals, we optimize for path–flow alignment, and regularize the intermediate marginals via stochastic regularization. Classifier-free guidance modifies sampling by combining conditional and unconditional predictions [7], while model guidance folds a guided direction into training so that sampling requires a single conditional evaluation [18]. Our main experiments use standard CFG evaluation (Section 6); Section 5 shows that the same path–flow alignment template also applies to model-guided velocity objectives.

3 Flow-Guided Interpolation

This section defines the learned training path. We first recall the population correctness statement for arbitrary endpoint-preserving paths, then introduce our residual parameterization and the alternating path–flow alignment algorithm.

3.1 Problem setup and notation

Let $p_0 = \mathcal{N}(0, I_d)$ be the latent source distribution. For each label $y \in \mathcal{Y}$, let $p_1(\cdot \mid y)$ denote the target latent distribution. For simplicity, this paper considers only independently coupled (x_0, x_1), with joint probability $(x_0, x_1 \mid y) \sim p_0(x_0)p_1(x_1 \mid y)$. The path–flow alignment algorithm and the population flow-matching correctness results below extend to more general endpoint couplings under their stated regularity conditions. However, the Gaussian-smoothing and entropy guarantees in Section 4.2 require an additional conditional-independence condition that is satisfied by the independent coupling above but need not hold for general optimal-transport couplings [19, 10]. For any given endpoints (x_0, x_1), define the standard linear interpolation

$$\gamma_{\text{lin}}(t; x_0, x_1) = (1-t)x_0 + tx_1, \qquad t \in [0,1]. \tag{1}$$

The linear path has constant velocity $x_1 - x_0$. Standard conditional flow matching trains a velocity network $v_\phi(x, t, y)$ by matching $\dot\gamma_{\text{lin}}(t; x_0, x_1) = x_1 - x_0$ at the intermediate state $\gamma_{\text{lin}}(t; x_0, x_1)$.

More generally, one can replace γ_{lin} by any sufficiently regular random path [11, 2, 1]. Under the regularity conditions of Proposition 1, an endpoint-preserving path induces an ideal ODE transport from source to target. We write a generic path as $\gamma(t; x_0, x_1, y, \xi)$, where ξ captures auxiliary randomness.

3.2 Flow matching and induced transport

Given any path γ, let $p_{\gamma,t}(\cdot \mid y)$ be the distribution of $\gamma(t; x_0, x_1, y, \xi)$ conditional on y, where ξ is sampled from its prescribed auxiliary distribution. The flow-matching loss is

$$\mathcal{L}_{\text{flow}}(v_\phi \mid \gamma) = \mathbb{E}_{y,x_1,x_0,\xi,t}\Big[\, \|v_\phi(\gamma(t; x_0, x_1, y, \xi), t, y) - \dot\gamma(t; x_0, x_1, y, \xi)\|_2^2 \,\Big]. \tag{2}$$

If $\gamma(t; x_0, x_1, y, \xi) = \gamma_{\text{lin}}(t; x_0, x_1)$, then $\dot\gamma(t; x_0, x_1, y, \xi) = x_1 - x_0$, and (2) reduces to the standard conditional flow-matching loss $\mathbb{E}_{y,x_1,x_0,t}\Big[\, \|v_\phi(\gamma_{\text{lin}}(t; x_0, x_1), t, y) - (x_1 - x_0)\|_2^2 \,\Big]$.

We call γ *endpoint-preserving* if, almost surely,

$$\gamma(0; x_0, x_1, y, \xi) = x_0, \qquad \gamma(1; x_0, x_1, y, \xi) = x_1. \tag{3}$$

Proposition 1 (Population target and induced ODE transport)**.** *Assume*

$$\mathbb{E}\int_0^1 \|\dot\gamma(t; x_0, x_1, y, \xi)\|_2^2 \, dt < \infty.$$

For any fixed path γ, the population minimizer of (2) *is*

$$v_\gamma^\star(x, t, y) = \mathbb{E}[\dot\gamma(t; x_0, x_1, y, \xi) \mid \gamma(t; x_0, x_1, y, \xi) = x,\ t,\ y]\,. \tag{4}$$

Assume in addition that $t \mapsto \gamma(t; x_0, x_1, y, \xi)$ is almost surely absolutely continuous, that γ is endpoint-preserving, and that $x_0 \sim p_0$ and $x_1 \sim p_1(\cdot \mid y)$. For each fixed y, assume that $v_\gamma^\star$ admits a jointly Borel version and, for almost every t, satisfies

$$\big\|v_\gamma^\star(x, t, y) - v_\gamma^\star(x', t, y)\big\|_2 \le L_y(t)\,\|x - x'\|_2\,,$$
$$\big\|v_\gamma^\star(x, t, y)\big\|_2 \le A_y(t)\big(1 + \|x\|_2\big),$$

for all $x, x' \in \mathbb{R}^d$, where $L_y, A_y \in L^1([0,1])$. Then the ODE

$$\frac{d}{dt}z(t) = v_\gamma^\star(z(t), t, y), \qquad z(0) \sim p_0, \tag{5}$$

has a unique Carathéodory flow whose marginals match those of $\gamma(t; x_0, x_1, y, \xi)$ for all $t \in [0,1]$. In particular,

$$z(1) \sim p_1(\cdot \mid y).$$

We defer the proof to Appendix E.1. Intuitively, (4) averages the tangents of all paths passing through the same state, time, and label. Therefore the ODE driven by $v_\gamma^\star$ moves probability mass in the same way as the path marginals.

3.3 Endpoint-preserving neural paths

Having verified the correctness of the flow-matching objective (2) for general paths, we now discuss how to use a residual path network $R_\psi(t; x_0, x_1, y) : [0,1] \times \mathbb{R}^d \times \mathbb{R}^d \times \mathcal{Y} \to \mathbb{R}^d$. to parameterize a general learnable path:

$$\gamma_\psi(t; x_0, x_1, y) = \gamma_{\text{lin}}(t; x_0, x_1) + t(1-t)R_\psi(t; x_0, x_1, y). \tag{6}$$

We describe the detailed implementation of R_ψ in Appendix C; all our results are agnostic to the parameterization of R_ψ. The envelope $t(1-t)$ allows the residual to curve the interior of the path while enforcing endpoint preservation (3). This envelope has previously been used in [17, 14, 3]. By observing that $t(1-t) = 0$ at $t = 0$ and $t = 1$, we immediately verify that

$$\gamma_\psi(0; x_0, x_1, y) = x_0, \qquad \gamma_\psi(1; x_0, x_1, y) = x_1. \tag{7}$$

3.4 Flow-guided path alignment and alternating optimization

Path and flow are trained on the same objective. For a fixed path γ_ψ, the flow phase minimizes

$$\mathcal{L}_{\text{flow}}(\phi \mid \psi) = \mathbb{E}_{y,x_1,x_0,t}\Big[\, \|v_\phi(\gamma_\psi(t; x_0, x_1, y), t, y) - \dot\gamma_\psi(t; x_0, x_1, y)\|_2^2 \,\Big]. \tag{8}$$

The path phase uses the same objective, but freezes v_ϕ and optimizes γ_ψ:

$$\begin{aligned}\mathcal{L}_{\text{path}}(\psi \mid \phi) =& \mathbb{E}_{y,x_1,x_0} \int_0^1 \|v_\phi(\gamma_\psi(t; x_0, x_1, y), t, y) - \dot\gamma_\psi(t; x_0, x_1, y)\|_2^2 \, dt \\ &+ \beta \mathbb{E}_{y,x_1,x_0} \int_0^1 \|\dot\gamma_\psi(t; x_0, x_1, y)\|_2^2 \, dt, \qquad \beta \geq 0.\end{aligned} \tag{9}$$

The first term of (9) is identical to (8). Thus the path phase does not introduce a separate supervision signal; it uses the current flow model to align the path. The second term of (9) is the kinetic energy of the path, scaled by β. This term regularizes the path by driving it toward the straight line; it is also important later for controlling the entropy of the induced path marginals (see Proposition 3 of Section 4.2).

We formally state the procedure in Algorithm 1 alternates between a **path training phase** and a **flow training phase**. In the simple case of $\rho_t \equiv 1$, $\hat{x}_0(t) = x_0$, and Algorithm 1 is ordinary alternating minimization: the path phase aligns γ_ψ to the current flow, and the flow phase trains v_ϕ on the current path. The optional stochastic-regularization case ($\rho_t < 1$) is discussed later in Section 4.2.

Correctness of flow-matching objective. Proposition 1 and (7) together imply that every updated path remains an exact interpolation from p_0 to $p_1(\cdot \mid y)$, and its population flow-matching target induces the corresponding correct ODE transport. Therefore, after each path update, the subsequent flow network phase is still a valid flow-matching problem for the current path.

3.5 Path overfitting

We first evaluate whether alternating optimization of γ_ψ and v_ϕ improves the downstream flow. Figure 1 shows the flow-matching loss after successive cycles of Algorithm 1, and Figure 2 shows the corresponding FID. We ablate the kinetic energy regularization strength β. For reference, the figure also includes the stochastic source-regularized variant introduced later in Section 4. Appendix D provides additional scatter-plot diagnostics for the same SiT-S sweep, relating FID to late divergence, average speed, and average acceleration of both the learned flow and the learned path.

The two plots reveal a clear path-overfitting effect. In the $\rho_t \equiv 1$ Algorithm 1 runs, the flow-matching loss continues to decrease as training proceeds. This means that the learned path and the learned flow become increasingly well aligned under the training objective. However, the corresponding FID does not continue to improve. For the unregularized runs, FID initially improves after the first cycle, but then rapidly deteriorates even as $\mathcal{L}_{\text{flow}}$ keeps decreasing.

This demonstrates that a low flow-matching loss is not, by itself, a sufficient condition for a good learned path. Proposition 1 guarantees that the population target for an endpoint-preserving path induces a correct transport, but the finite trained network can still be poor for generation in practice. In other words, the path objective can overfit: it can improve the local regression problem without preserving the distributional properties needed for robust sampling.

Algorithm 1 Alternating path–flow alignment with optional stochastic regularization

Require: Path parameters ψ, flow parameters ϕ, cycles C, path steps K_ψ, flow steps K_ϕ, kinetic weight β, optional perturbation schedule $t \mapsto \rho_t$

for $c = 1, \dots, C$ **do**
  for $k = 1, \dots, K_\psi$ **do**
    Sample $y, x_1 \sim p_1(\cdot \mid y)$, $x_0 \sim p_0$, $t \sim \mathrm{Unif}[0,1]$, $\epsilon \sim \mathcal{N}(0, I_d)$
    $\hat{x}_0(t) = \rho_t x_0 + \sqrt{1-\rho_t^2}\epsilon$
    $\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) = (1-t)x_0 + t x_1 + t(1-t)R_\psi(t; \hat{x}_0(t), x_1, y)$.
    $\mathcal{J}_\psi \leftarrow \|\partial_t \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) - v_\phi(\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon), t, y)\|_2^2 + \beta \|\partial_t \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon)\|_2^2$
    Update ψ using $-\nabla_\psi \mathcal{J}_\psi$, with ϕ frozen
  end for
  for $k = 1, \dots, K_\phi$ **do**
    Sample $y, x_1 \sim p_1(\cdot \mid y)$, $x_0 \sim p_0$, $t \sim \mathrm{Unif}[0,1]$, $\epsilon \sim \mathcal{N}(0, I_d)$
    $\hat{x}_0(t) = \rho_t x_0 + \sqrt{1-\rho_t^2}\epsilon$
    $\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) = (1-t)x_0 + t x_1 + t(1-t)R_\psi(t; \hat{x}_0(t), x_1, y)$.
    $\mathcal{J}_\phi \leftarrow \|\partial_t \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) - v_\phi(\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon), t, y)\|_2^2$
    Update ϕ using $-\nabla_\phi \mathcal{J}_\phi$, with ψ frozen
  end for
end for

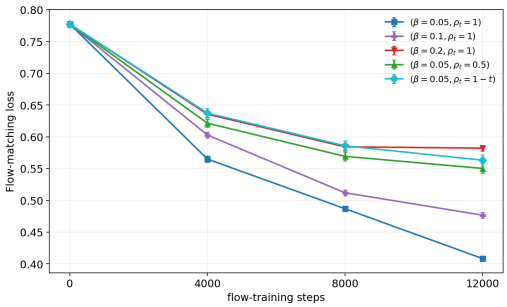


Figure 1: Flow-matching loss across alternating training cycles on the SiT-S model.

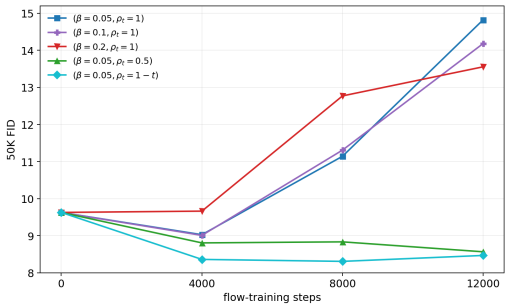


Figure 2: FID across alternating training cycles on the SiT-S model.

4 Entropy Bottlenecks and Stochastic Regularization

Section 3.5 shows that endpoint correctness and low flow-matching loss do not by themselves guarantee good generation. The learned path can become well aligned with the current flow on its own training states while the resulting sampler develops unstable distributional behavior. In this section, we identify this behavior as an entropy bottleneck and introduce a stochastic regularizer motivated by this diagnosis.

The key diagnostic is average flow divergence. Along any probability flow, average divergence equals the rate of change of differential entropy. Thus a late positive-divergence spike means that the sampler is rapidly expanding entropy near the endpoint. If this expansion compensates for earlier over-contraction, the flow must have passed through an overly concentrated intermediate marginal. We call this a low-entropy bottleneck.

We next introduce stochastic source perturbation: by perturbing the x_0 input to the residual path network, we effectively convolve the intermediate path marginals by an independent Gaussian, which in turn lower bounds entropy. The same smoothing also bounds Fisher information; combined with path kinetic energy, this gives an instantaneous lower bound on entropy decrease (equivalently average flow divergence).

To keep the notation separate, Section 4.1 diagnoses the probability flow generated by the learned sampler v_ϕ, whose marginals are denoted $p_{\phi,t}$. Section 4.2 analyzes the stochastic-regularized training path $\hat{\gamma}_\psi$, whose marginals are denoted $\hat{p}_{\psi,t}$. In the population limit where the trained flow

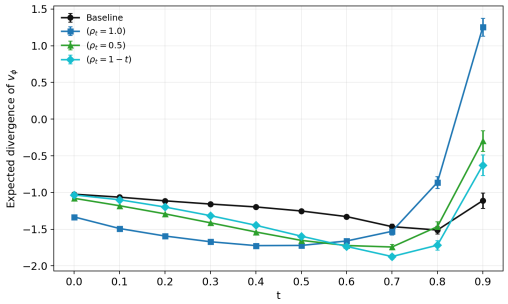


Figure 3: Per-dimensional binned divergence of the learned SiT-S flow network along its induced probability flow.

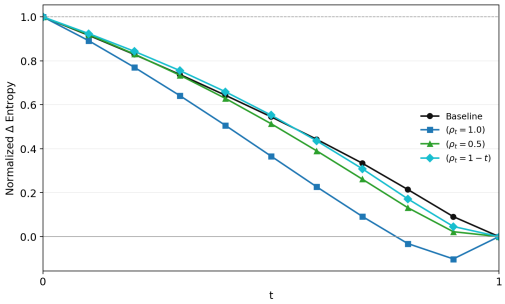


Figure 4: Cumulative normalized divergence. Values below zero indicate an entropy bottleneck relative to the generated endpoint distribution.

equals the path-induced target, $v_\phi = \hat{v}^\star_\psi$, the sampler marginals $p_{\phi,t}$ coincide with the regularized path marginals $\hat{p}_{\psi,t}$.

The distinction between these marginals is important for the scope of our results. Theorem 1 and Proposition 3 below control the stochastic training-path marginals $\hat{p}_{\psi,t}$, whereas Proposition 2 applies directly to the finite learned-sampler marginals $p_{\phi,t}$. The two are guaranteed to coincide when $v_\phi = \hat{v}^\star_\psi$, but may differ under finite optimization and model capacity. For a fixed path, population excess flow-matching risk controls an L^2 velocity discrepancy, while entropy evolution depends on velocity derivatives through divergence; velocity-level control alone therefore does not transfer the path-level entropy guarantee to the learned sampler. We accordingly interpret the divergence measurements below as empirical evidence about the learned sampler rather than as a consequence of the path-level guarantee.

4.1 Divergence spikes reveal entropy bottlenecks

An important diagnostic for the path-overfitting phenomenon in Section 3.5 is the average divergence of the learned velocity field v_ϕ, which has a direct connection to entropy: for any probability flow, the expected divergence of the velocity field equals the time derivative of differential entropy. Thus a large positive-divergence spike near the endpoint indicates that the learned flow is rapidly increasing entropy at late time. If this late expansion follows an earlier over-contraction, then the probability flow has passed through a low-entropy bottleneck.

For each trained model, let $z_\phi(t)$ denote the ODE trajectory generated by the learned flow,

$$\frac{d}{dt} z_\phi(t) = v_\phi(z_\phi(t), t, y), \qquad z_\phi(0) \sim p_0, \tag{10}$$

and let $p_{\phi,t}(\cdot \mid y)$ be the law of $z_\phi(t)$ conditional on y. We divide the time interval $[0, 1]$ into ten bins. For a bin $[a, b]$, we estimate the expected flow divergence

$$\bar{D}_\phi([a, b]) = \frac{1}{d(b-a)} \int_a^b \mathbb{E}_y \mathbb{E}_{x \sim p_{\phi,t}(\cdot|y)} \left[\nabla_x \cdot v_\phi(x, t, y)\right] dt, \tag{11}$$

using a Hutchinson trace estimator. The factor $1/d$ gives the per-dimensional divergence. Figure 3 plots this expected flow divergence against the time, for flows v_ϕ trained using baseline flow-matching, as well as using Algorithm 1 with different values of ρ_t. The overfit $\rho_t = 1$ run produces a pronounced positive-divergence spike near $t = 1$, in sharp contrast to the baseline and the stochastic source-regularized run introduced in Section 4.2.

The following proposition relates flow-divergence to entropy-change:

Proposition 2 (Expected divergence equals entropy rate)**.** *Let $p_t(\cdot \mid y)$ evolve under the continuity equation $\partial_t p_t(x \mid y) + \nabla_x \cdot (p_t(x \mid y) u_t(x, y)) = 0$, with sufficient smoothness, decay at infinity, and finite differential entropy. Equivalently, when the ODE is well posed, $p_t(\cdot \mid y)$ is the law of the solution to $\frac{d}{dt} z(t) = u_t(z(t), y)$. Then*

$$\frac{d}{dt} h(p_t(\cdot \mid y)) = \mathbb{E}_{x \sim p_t(\cdot|y)} \left[\nabla_x \cdot u_t(x, y)\right]. \tag{12}$$

Consequently, for any time bin $0 \le a < b \le 1$,

$$\frac{1}{b-a}\int_a^b \mathbb{E}_{x\sim p_t(\cdot|y)}\left[\nabla_x \cdot u_t(x,y)\right] dt = \frac{h(p_b(\cdot \mid y)) - h(p_a(\cdot \mid y))}{b-a}. \tag{13}$$

We defer the proof to Appendix E.2. We apply Proposition 2 directly to the learned flow by taking $u_t(\cdot, y) = v_\phi(\cdot, t, y), p_t(\cdot \mid y) = p_{\phi,t}(\cdot \mid y)$. This gives the exact identity

$$\frac{d}{dt}h(p_{\phi,t}(\cdot \mid y)) = \mathbb{E}_{x\sim p_{\phi,t}(\cdot|y)}\left[\nabla_x \cdot v_\phi(x,t,y)\right]. \tag{14}$$

Thus the divergence curves in Figure 3 are not merely local geometric measurements of v_ϕ; they estimate the entropy-rate profile of the probability flow generated by the learned sampler. If, in the population limit, v_ϕ equals the ideal path-induced velocity $v_\psi^\star$ from Proposition 1, then $p_{\phi,t} = p_{\psi,t}$ by the induced ODE transport statement, and the generated endpoint is $p_1(\cdot \mid y)$.

The identity also gives a precise meaning to a low-entropy bottleneck. For any $s < 1$, applying (14) on $[s, 1]$ gives $h(p_{\phi,1}(\cdot \mid y)) - h(p_{\phi,s}(\cdot \mid y)) = \int_s^1 \mathbb{E}_{x\sim p_{\phi,t}(\cdot|y)}\left[\nabla_x \cdot v_\phi(x,t,y)\right] dt$. Therefore,

$$\int_s^1 \mathbb{E}_{x\sim p_{\phi,t}(\cdot|y)}\left[\nabla_x \cdot v_\phi(x,t,y)\right] dt > 0 \quad \Longleftrightarrow \quad h(p_{\phi,s}(\cdot \mid y)) < h(p_{\phi,1}(\cdot \mid y)). \tag{15}$$

We call this event a *low-entropy bottleneck*: the learned probability flow has contracted below the entropy of its own endpoint distribution $p_{\phi,1}$ and must subsequently expand entropy before $t = 1$. In the ideal endpoint-preserving case $p_{\phi,1} = p_1$, so the same condition says that the intermediate marginal has entropy below the target entropy.

Figure 4 visualizes this effect through cumulative divergence. Let $\bar{D}_\phi(t)$ denote the piecewise-constant, per-dimensional binned estimate from (11). We plot $\widehat{G}_\phi(t) = 1 - (\int_0^t \bar{D}_\phi(r)\, dr)/(\int_0^1 \bar{D}_\phi(r)\, dr)$. For the exact entropy-rate curve, this quantity equals the normalized excess entropy above the endpoint,

$$G_\phi(t; y) = \frac{h(p_{\phi,t}(\cdot \mid y)) - h(p_{\phi,1}(\cdot \mid y))}{h(p_0) - h(p_{\phi,1}(\cdot \mid y))} = 1 - \frac{\int_0^t \mathbb{E}_{x\sim p_{\phi,r}(\cdot|y)}[\nabla_x \cdot v_\phi(x,r,y)]\, dr}{\int_0^1 \mathbb{E}_{x\sim p_{\phi,r}(\cdot|y)}[\nabla_x \cdot v_\phi(x,r,y)]\, dr}. \tag{16}$$

The plotted statistic estimates the divergences with Hutchinson traces. A value near 1 corresponds to source-side entropy, a value near 0 corresponds to reaching the endpoint entropy, and a value below 0 indicates entropy below the endpoint entropy. The overfit run dips below zero around $t \approx 0.9$, matching the large positive-divergence spike in Figure 3. This supports the interpretation that the overfit flow passes through a low-entropy bottleneck and then requires a compensating expansion.

4.2 Regularization via Gaussian perturbation

Section 4.1 suggests that path overfitting is associated with entropy bottlenecks: the learned path can reduce its local regression objective while routing probability mass through overly concentrated intermediate marginals. We therefore regularize the residual path network by hiding part of the source noise from the residual input. The key design choice is that the perturbation is applied only to the input of R_ψ; the endpoint interpolation itself still uses the true endpoints (x_0, x_1).

Let $\rho_t \in [0, 1]$ be a deterministic perturbation schedule and let $\epsilon \sim \mathcal{N}(0, I_d)$ be independent of (x_0, x_1, y). Define

$$\hat{x}_0(t) = \rho_t x_0 + \sqrt{1-\rho_t^2}\,\epsilon. \tag{17}$$

For each fixed t, the perturbed source input is still marginally standard Gaussian, $\hat{x}_0(t) \sim \mathcal{N}(0, I_d)$, but its correlation with the true source endpoint is reduced from 1 to ρ_t. Thus the residual network receives a source-like input with less information about the particular endpoint noise x_0 that appears in the actual interpolation. The stochastic-regularized path is

$$\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) = (1-t)x_0 + tx_1 + t(1-t)R_\psi(t; \hat{x}_0(t), x_1, y). \tag{18}$$

This is the stochastic path used in Algorithm 1. Setting $\rho_t \equiv 1$ gives $\hat{x}_0(t) = x_0$ and recovers the unregularized path γ_ψ. Setting $\rho_t < 1$ hides part of x_0 from the residual network, while leaving the

true linear endpoint term unchanged. When ρ_t is time-dependent, the tangent $\partial_t \hat{\gamma}_\psi$ differentiates through both the envelope $t(1-t)$ and the schedule ρ_t in (17). The perturbation does not change the endpoints. For every schedule ρ_t, every ϵ, and every residual network R_ψ,

$$\hat{\gamma}_\psi(0; x_0, x_1, y, \epsilon) = x_0, \qquad \hat{\gamma}_\psi(1; x_0, x_1, y, \epsilon) = x_1. \tag{19}$$

Only the residual term sees the corrupted source input, and this residual term is multiplied by $t(1-t)$, which vanishes at both endpoints. When path learning is disabled, $R_\psi \equiv 0$, so (18) reduces to the standard linear path independently of ρ_t; stochastic source perturbation therefore has no standalone effect without a learned residual path.

Let $\hat{p}_{\psi,t}(\cdot \mid y)$ denote the distribution of $\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon)$ conditional on y, where (x_0, x_1) is sampled from the chosen endpoint coupling and $\epsilon \sim \mathcal{N}(0, I_d)$. The corresponding population flow-matching target is

$$\hat{v}^\star_\psi(x, t, y) = \mathbb{E}\left[\partial_t \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) \mid \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) = x,\ t,\ y\right]. \tag{20}$$

Thus Gaussian perturbation changes the training path distribution, but it does not make the flow-matching target biased or approximate: the population target remains the exact marginal velocity of the perturbed path.

Corollary 1 (Correctness of stochastic source perturbation)**.** *Assume that the stochastic path* (18) *satisfies the assumptions of Proposition 1. Then the flow-matching loss trained on pairs* $(\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon), \partial_t \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon))$ *has population minimizer* (20)*. Moreover, the probability flow generated by* $\hat{v}^\star_\psi$ *transports* p_0 *to* $p_1(\cdot \mid y)$*.*

Proof. The additional Gaussian variable ϵ is simply auxiliary randomness in the sense of Section 3.2. Endpoint preservation follows from (19). The Corollary follows immediately from Proposition 1. □

We now explain why the stochastic source perturbation in Section 4.2 directly regularizes the intermediate marginals: First, at any fixed time t away from the endpoint, the part of x_0 hidden from the residual network can be rewritten as independent additive Gaussian noise in the state. Thus $\hat{p}_{\psi,t}(\cdot \mid y)$ is a Gaussian-smoothed distribution and has an explicit entropy floor. Second, the same smoothing gives a Fisher-information bound. Combining this Fisher-information bound with the pointwise kinetic energy of the path yields an instantaneous lower bound on the entropy rate, or equivalently on the average divergence of the ideal marginal velocity.

The correctness results above do not require the independent endpoint coupling. For the Gaussian-smoothing result below, however, we use the independent coupling from Section 3.1: conditional on y, $x_0 \sim \mathcal{N}(0, I_d)$ and $x_1 \sim p_1(\cdot \mid y)$ are independent, and $\epsilon \sim \mathcal{N}(0, I_d)$ is independent of both. More generally, a sufficient condition for the argument below is that η_t in (23) is conditionally standard Gaussian and independent of $(\hat{x}_0(t), x_1)$ given y. A general optimal-transport coupling need not satisfy this condition.

Theorem 1 (Source randomization induces Gaussian smoothing)**.** *Fix a time* $t \in [0, 1)$ *such that* $\rho_t < 1$*, and define*

$$\sigma_t = (1-t)\sqrt{1 - \rho_t^2}. \tag{21}$$

Let $\hat{p}_{\psi,t}(\cdot \mid y)$ *be the law of the stochastic path* (18) *conditional on* y*. Then there exists a distribution* $q_{\psi,t}(\cdot \mid y)$ *such that*

$$\hat{p}_{\psi,t}(\cdot \mid y) = q_{\psi,t}(\cdot \mid y) * \mathcal{N}(0, \sigma_t^2 I_d). \tag{22}$$

More explicitly, if

$$\eta_t = \frac{x_0 - \rho_t \hat{x}_0(t)}{\sqrt{1 - \rho_t^2}}, \tag{23}$$

then, conditional on y*,* $\eta_t \sim \mathcal{N}(0, I_d)$ *and* η_t *is independent of* $(\hat{x}_0(t), x_1)$*, and*

$$\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) = (1-t)\rho_t \hat{x}_0(t) + t x_1 + t(1-t) R_\psi(t; \hat{x}_0(t), x_1, y) + \sigma_t \eta_t. \tag{24}$$

The distribution $q_{\psi,t}(\cdot \mid y)$ *in* (22) *is the law, conditional on* y*, of the non-Gaussian term*

$$(1-t)\rho_t \hat{x}_0(t) + t x_1 + t(1-t) R_\psi(t; \hat{x}_0(t), x_1, y). \tag{25}$$

Consequently, whenever the differential entropy is well-defined,

$$h(\hat{p}_{\psi,t}(\cdot \mid y)) \geq \frac{d}{2}\log\big(2\pi e\,\sigma_t^2\big) = \frac{d}{2}\log\big(2\pi e(1-t)^2(1-\rho_t^2)\big)\,. \tag{26}$$

Moreover, the Fisher information of the smoothed marginal satisfies

$$I(\hat{p}_{\psi,t}(\cdot \mid y)) := \mathbb{E}_{x\sim\hat{p}_{\psi,t}(\cdot|y)}\left[\|\nabla_x \log \hat{p}_{\psi,t}(x \mid y)\|_2^2\right] \leq \frac{d}{\sigma_t^2} = \frac{d}{(1-t)^2(1-\rho_t^2)}. \tag{27}$$

We defer the proof to Appendix E.3. The entropy floor in (26) is a state-level obstruction to arbitrarily deep entropy collapse. The bound becomes vacuous as $t \to 1$, which is unavoidable: endpoint preservation requires the added smoothing to vanish at the target endpoint.

This smoothing result also gives the bin-level intuition behind the divergence diagnostics above. Applying Proposition 2 to the ideal velocity $\hat{v}_\psi^\star$ gives, for any $0 \leq a < b < 1$,

$$\int_a^b \mathbb{E}_{x\sim\hat{p}_{\psi,t}(\cdot|y)}[\nabla_x \cdot \hat{v}_\psi^\star(x,t,y)]\,dt = h(\hat{p}_{\psi,b}(\cdot \mid y)) - h(\hat{p}_{\psi,a}(\cdot \mid y)).$$

Thus binned average divergence is a finite-difference entropy rate. Since (26) lower-bounds the entropy at the end of any bin with $\sigma_b > 0$, Gaussian perturbation prevents arbitrarily deep entropy collapse on such a bin unless the entropy at the beginning of the bin is correspondingly large.

We next give an instantaneous version of this control:

Proposition 3 (Entropy-rate lower bound under kinetic-energy control)**.** *Assume the regularity conditions needed for Proposition 2 and the coupling and regularity conditions of Theorem 1. Define the pointwise stochastic-path kinetic energy*

$$L_{\mathrm{euc}}^{\rho,\psi}(t,y) := \mathbb{E}\left[\|\partial_t \hat{\gamma}_\psi(t;x_0,x_1,y,\epsilon)\|_2^2 \;\middle|\; y\right], \tag{28}$$

where the superscript ρ denotes the schedule $t \mapsto \rho_t$. Then, for every $t < 1$ with $\rho_t < 1$,

$$\frac{d}{dt}h(\hat{p}_{\psi,t}(\cdot \mid y)) \geq -\frac{\sqrt{d\,L_{\mathrm{euc}}^{\rho,\psi}(t,y)}}{(1-t)\sqrt{1-\rho_t^2}} = -\frac{d}{1-t}\sqrt{\frac{L_{\mathrm{euc}}^{\rho,\psi}(t,y)}{d(1-\rho_t^2)}}. \tag{29}$$

We defer the proof to Appendix E.4. Proposition 3 shows that stochastic perturbation and kinetic regularization play complementary roles. The smoothing scale $\sigma_t = (1-t)\sqrt{1-\rho_t^2}$ controls the Fisher information of the marginal: larger smoothing makes the score smaller and prevents sharper entropy collapse. The kinetic term controls the size of the marginal velocity acting on this smoothed distribution. Thus L_{euc} is not only a geometric straightness penalty; in the presence of Gaussian perturbation, it also bounds how fast the path-induced marginal can decrease entropy.

The scaling is comparable to the standard linear path. If $R_\psi \equiv 0$ and the target is a point mass, then the linear path has marginal $\mathcal{N}(0,(1-t)^2 I_d)$ and entropy rate exactly $-d/(1-t)$. The bound (29) recovers this order when ρ_t is bounded away from 1. Conversely, when ρ_t is close to 1, there is little smoothing and the bound appropriately becomes weak.

5 Generalizing Beyond Flow Matching

Path–flow alignment is not tied to the standard flow-matching velocity; it only needs a velocity-like target that is regressed to the path tangent. Model guidance [18] provides such a target by folding classifier-free guidance into training. For guidance scale ω, define

$$\tilde{v}_\phi^\omega(x,t,y) = v_\phi(x,t,y) - \omega\,\mathrm{sg}(v_\phi(x,t,y) - v_\phi(x,t,\emptyset))\,,$$

where sg denotes stop-gradient, and train with

$$\mathcal{L}_{\mathrm{MG}}(\phi \mid \gamma) = \mathbb{E}_{y,x_1,x_0,\xi,t}\left\|\tilde{v}_\phi^\omega(\gamma(t;x_0,x_1,y,\xi),t,y) - \dot{\gamma}(t;x_0,x_1,y,\xi)\right\|_2^2. \tag{30}$$

We use the same alternating updates as Algorithm 1, replacing v_ϕ by $\tilde{v}_\phi^\omega$ in both phases. Stochastic source perturbation and endpoint preservation are unchanged. Table 1 shows that this extension improves 50k-FID over standard MG fine-tuning on both SiT-B-MG and SiT-XL-MG.

6 Experiments

Unless otherwise stated, we use $\beta = 0.05$ and $\rho_t = c(1-t)$ with $c = 1$ for SiT-S/B/L and $c = 0.7$ for SiT-XL. We provide full training setups in Appendix F.

ImageNet results. We evaluate regularized path–flow alignment on class-conditional ImageNet-256×256 using SiT backbones [13, 5]. Each run starts from the same pretrained SiT checkpoint. The baseline continues standard flow-matching training for 12k steps; our method uses three alternating cycles, each with 3k path updates followed by 4k flow updates. Table 1 reports the best 50k-FID achieved within the 12k flow-update budget. Our method improves FID at every scale, including SiT-XL, where standard fine-tuning does not improve the starting checkpoint. The largest gains are on SiT-L and SiT-XL, with relative FID improvements of 19.1% and 12.7%, respectively. The full checkpoint-wise FID/IS table is in Appendix Table 3.

Model guidance. We also test the effective-velocity extension from Section 5. Starting from FID-converged SiT-B-MG and SiT-XL-MG checkpoints, we compare standard MG fine-tuning baseline against path–flow alignment with the MG objective. Table 1 shows that our method improves the best FID from 7.940 to 5.585 on SiT-B-MG and from 1.494 to 1.322 on SiT-XL-MG. Checkpoint-wise FID results are in Appendix Table 4.

The FID improvements persist against compute-matched continued finetuning (Appendix F.1) and across three disjoint generation-seed sets (Appendix F.2). The ImageNet results improve recall and Inception Score (Appendix F.2 and Table 3). We also find gains on FlowDCN-B (Appendix F.3) and compare with [17] (Appendix F.3). We demonstrate that path–flow alignment increases curvature of sampled trajectories on a toy setup and include qualitative comparisons showing improvements in image quality (Appendix F.4).

Table 1: ImageNet-256×256 summary. We report the best 50k-FID (↓) over the same 12k continuation budget; full checkpoint-wise FID/IS results are in Appendix Tables 3–4. Params denote the inference-time SiT flow model only; the path network is used only during training.

Objective	Model	Params	Init FID	Baseline best	Ours best	FID gain
FM	SiT-S	33M	9.109	9.022	**8.359**	7.3%
FM	SiT-B	130M	5.074	5.057	**4.415**	12.7%
FM	SiT-L	458M	3.424	3.390	**2.742**	19.1%
FM	SiT-XL	675M	2.091	2.091	**1.825**	12.7%
MG	SiT-B-MG	130M	8.090	7.940	**5.585**	29.7%
MG	SiT-XL-MG	675M	1.522	1.494	**1.322**	11.5%

7 Conclusion

We introduced path–flow alignment, which jointly learns an endpoint-preserving training path and flow model, and identified path overfitting as a failure mode of naive joint optimization. Stochastic source perturbation provides Gaussian-smoothing and entropy guarantees for the learned training path, while our experiments show that it also suppresses the associated late-time divergence bottleneck in finite learned samplers. Across SiT and FlowDCN models, the resulting post-training procedure improves generation quality under compute-matched comparisons without changing the inference architecture or sampler. Our theory does not establish a causal link between entropy bottlenecks and generation quality, nor does it guarantee transfer of the path-level entropy control to an imperfectly fitted flow; understanding this connection and extending joint path–flow training beyond pretrained image generators remain important directions.

Funding and Competing Interests

The authors received no third-party funding or support for this work, and declare no competing financial interests.

References

[1] Michael S. Albergo, Nicholas M. Boffi, and Eric Vanden-Eijnden. Stochastic interpolants: A unifying framework for flows and diffusions, 2025.

[2] Michael S. Albergo and Eric Vanden-Eijnden. Building normalizing flows with stochastic interpolants, 2023.

[3] Grigory Bartosh, Dmitry Vetrov, and Christian A. Naesseth. Neural flow diffusion models: Learnable forward process for improved diffusion modelling, 2025.

[4] Sergio Calvo-Ordonez, Matthieu Meunier, Alvaro Cartea, Christoph Reisinger, Yarin Gal, and Jose Miguel Hernandez-Lobato. Weighted conditional flow matching, 2026.

[5] 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*, pages 248–255, 2009.

[6] Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, and Sepp Hochreiter. Gans trained by a two time-scale update rule converge to a local nash equilibrium, 2018.

[7] Jonathan Ho and Tim Salimans. Classifier-free diffusion guidance, 2022.

[8] Kacper Kapuśniak, Peter Potaptchik, Teodora Reu, Leo Zhang, Alexander Tong, Michael Bronstein, Avishek Joey Bose, and Francesco Di Giovanni. Metric flow matching for smooth interpolations on the data manifold, 2024.

[9] Oskar Kviman, Kirill Tamogashev, Nicola Branchini, Víctor Elvira, Jens Lagergren, and Nikolay Malkin. Multi-marginal flow matching with adversarially learnt interpolants, 2026.

[10] Yexiong Lin, Yu Yao, and Tongliang Liu. Beyond optimal transport: Model-aligned coupling for flow matching, 2025.

[11] Yaron Lipman, Ricky T. Q. Chen, Heli Ben-Hamu, Maximilian Nickel, and Matt Le. Flow matching for generative modeling, 2023.

[12] Xingchao Liu, Chengyue Gong, and Qiang Liu. Flow straight and fast: Learning to generate and transfer data with rectified flow, 2022.

[13] Nanye Ma, Mark Goldstein, Michael S. Albergo, Nicholas M. Boffi, Eric Vanden-Eijnden, and Saining Xie. Sit: Exploring flow and diffusion-based generative models with scalable interpolant transformers, 2024.

[14] Katarina Petrović, Lazar Atanackovic, Viggo Moro, Kacper Kapuśniak, İsmail İlkan Ceylan, Michael Bronstein, Avishek Joey Bose, and Alexander Tong. Curly flow matching for learning non-gradient field dynamics, 2025.

[15] Aram-Alexandre Pooladian, Heli Ben-Hamu, Carles Domingo-Enrich, Brandon Amos, Yaron Lipman, and Ricky T. Q. Chen. Multisample flow matching: Straightening flows with minibatch couplings, 2023.

[16] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, and Xi Chen. Improved techniques for training gans, 2016.

[17] Shiv Shankar and Tomas Geffner. Learning straight flows by learning curved interpolants, 2025.

[18] Zhicong Tang, Jianmin Bao, Dong Chen, and Baining Guo. Diffusion models without classifier-free guidance, 2025.

[19] Alexander Tong, Kilian Fatras, Nikolay Malkin, Guillaume Huguet, Yanlei Zhang, Jarrid Rector-Brooks, Guy Wolf, and Yoshua Bengio. Improving and generalizing flow-based generative models with minibatch optimal transport, 2024.

[20] Shuai Wang, Zexian Li, Tianhui Song, Xubin Li, Tiezheng Ge, Bo Zheng, and Limin Wang. Flowdcn: Exploring dcn-like architectures for fast image generation with arbitrary resolution, 2024.

A Notation guide

Table 2 summarizes the main path, marginal, and velocity notation used throughout the paper. The generic notation γ and $v^\star_\gamma$ is used in the population flow-matching statements; the symbols γ_ψ and $\hat{\gamma}_\psi$ denote the two learned-path special cases used by the algorithm and the stochastic regularizer.

Table 2: Notation convention for paths, induced marginals, and velocity fields.

Symbol	Meaning
γ_{lin}	Standard linear path, $\gamma_{\mathrm{lin}}(t; x_0, x_1) = (1-t)x_0 + tx_1$.
γ	Generic sufficiently regular path, possibly with auxiliary randomness ξ.
γ_ψ	Unregularized learned residual path.
$\hat{\gamma}_\psi$	Stochastic-regularized learned residual path.
$p_{\gamma,t}$	Marginal distribution of a generic path $\gamma(t; x_0, x_1, y, \xi)$ at time t.
$p_{\psi,t}$	Marginal distribution of the unregularized learned path $\gamma_\psi(t; x_0, x_1, y)$.
$\hat{p}_{\psi,t}$	Marginal distribution of the stochastic-regularized path $\hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon)$.
$p_{\phi,t}$	ODE marginal induced by the learned sampler v_ϕ; this is not a path marginal unless v_ϕ equals the corresponding ideal path velocity.
$v^\star_\gamma$	Population flow-matching target for a generic path γ; it is the conditional mean of the path tangent given position, time, and label.
$v^\star_\psi$	Population flow-matching target for the unregularized learned path γ_ψ.
$\hat{v}^\star_\psi$	Population flow-matching target for the stochastic-regularized path $\hat{\gamma}_\psi$.
v_ϕ	Trained flow network used for sampling at inference time.
u_t	Generic velocity field used in entropy-rate identities.
ρ_t	Time-dependent source-perturbation schedule.
σ_t	Gaussian smoothing scale induced by source perturbation, $\sigma_t = (1-t)\sqrt{1-\rho_t^2}$.
$L^{\rho,\psi}_{\mathrm{euc}}(t, y)$	Pointwise kinetic energy of the stochastic-regularized path, defined in (28).

B Discussion and Limitations

Our experiments focus on the image generation task using FID as the key metric. While this is a challenging problem, flow-based generative models have been applied in a broad range of settings. It would be useful to investigate the efficacy of our method in these applications as well.

Path–flow alignment requires additional training compute and memory for the path network, and alternating path and flow optimization adds engineering complexity. The compute-matched results in Appendix F.1 show that the gains persist under comparable continuation compute, but do not remove this practical overhead. Our experiments study continued training from pretrained or converged checkpoints rather than joint training from scratch. Although Appendix F.3 broadens the architecture and dataset scope, the evaluation remains restricted to class-conditional image generation; text-to-image models and non-image domains remain untested. The sensitivity study over ρ_t and β is limited, the path network and subtractor design is not exhaustively ablated, and the multi-seed results in Appendix F.2 measure generation-seed variability rather than variability across independently trained models.

Theorem 1 and Proposition 3 control the stochastic training-path marginals and their path-induced population velocity, not directly the marginals produced by the finite learned sampler. These marginals coincide when the learned flow equals the path-induced population velocity, but may differ under finite optimization and model capacity; velocity-level flow-matching control alone does not provide the derivative-level control needed to transfer the entropy guarantee. Accordingly, the observed association between late-time divergence, entropy bottlenecks, and FID is empirical rather than causal. Finally, the Gaussian-convolution, entropy, and Fisher-information guarantees require the stated conditional-independence condition and do not automatically extend to general endpoint couplings such as optimal transport.

C Path network and subtractor implementation

Below we describe the implementation of the residual path network R_ψ used in the SiT experiments. The final sampler is still an ordinary SiT velocity model v_ϕ. The additional path machinery consists of two SiT-shaped networks used only to define the training path: a trainable path network P_ψ and a frozen subtractor network S. The role of the subtractor is to make the learned path start exactly as the linear path while still allowing P_ψ to learn endpoint-conditioned residual curvature.

Teacher-residualized path parameterization. Let

$$\gamma_{\mathrm{lin}}(t; x_0, x_1) = (1-t)x_0 + tx_1.$$

The implemented path is

$$\begin{aligned}\gamma_\psi^{\mathrm{impl}}(t; x_0, x_1, y, \epsilon_0) = \gamma_{\mathrm{lin}}(t; x_0, x_1) \\ + t(1-t)\Big[P_\psi(\hat{\gamma}_{\mathrm{lin}}(t), \hat{\Delta}(t), t, y) - S(\hat{\gamma}_{\mathrm{lin}}(t), t, y)\Big].\end{aligned} \tag{31}$$

thus (31) matches the residual form in (6) with

$$R_\psi(t; x_0, x_1, y) = P_\psi(\gamma_{\mathrm{lin}}(t; x_0, x_1), x_1 - x_0, t, y) - S(\gamma_{\mathrm{lin}}(t; x_0, x_1), t, y).$$

Perturbed endpoint inputs. The outer anchor path in (31) always uses the true endpoints (x_0, x_1). Perturbations are applied only to the inputs of P_ψ and S. Given schedules $\rho_t \in [0, 1]$, define

$$\hat{x}_0(t) = \rho_t x_0 + \sqrt{1-\rho_t^2}\,\epsilon_0. \tag{32}$$

The source perturbation ρ_t is the stochastic regularizer analyzed in Section 4.2. The actual inputs to the path network and subtractor are

$$\hat{\gamma}_{\mathrm{lin}}(t) = (1-t)\hat{x}_0(t) + tx_1, \qquad \hat{\Delta}(t) = x_1 - \hat{x}_0(t). \tag{33}$$

Thus stochastic regularization hides part of the endpoint information from the residual branch, while the endpoint identities of the learned path remain exact because the residual term is multiplied by the boundary envelope.

Subtractor network. The subtractor $S(x, t, y)$ is an unmodified SiT velocity network with the standard single patch-embedding stem, timestep embedding, class-label embedding, SiT transformer blocks, and final velocity head. In the main experiments, S is loaded from a pretrained SiT-S/2 checkpoint and frozen. During subsequent flow training, both the learned path network and the subtractor are frozen; together they define the training states and tangent targets for the flow model v_ϕ.

Dual-stem path network. The path network P_ψ uses the same SiT transformer body as S, but replaces the single spatial input stem with two stems. The first stem embeds the hatted linear state and the second stem embeds the hatted endpoint displacement:

$$z_{\mathrm{lin}} = \mathrm{Embed}_{\mathrm{lin}}(\hat{\gamma}_{\mathrm{lin}}(t)), \qquad z_\Delta = \mathrm{Embed}_\Delta(\hat{\Delta}(t)).$$

The token sequence entering the transformer is

$$z = z_{\mathrm{lin}} + z_\Delta + z_{\mathrm{pos}}, \tag{34}$$

where z_{pos} is the fixed sinusoidal positional embedding used by SiT. The usual SiT conditioning vector is

$$c = e_t(t) + e_y(y),$$

where e_t is the timestep embedding and e_y is the class-label embedding. When endpoint conditioning is enabled, the delta-stem tokens are also average pooled and passed through a small projector,

$$c = e_t(t) + e_y(y) + W_{\mathrm{end}}\,\mathrm{SiLU}\left(\mathrm{LayerNorm}\left(\frac{1}{N}\sum_{i=1}^{N} z_{\Delta,i}\right)\right). \tag{35}$$

This global endpoint-conditioning branch is the `EndpointConditionProjector` module, and the full path network is implemented as `DualStemPathSiT`. The transformer blocks and final head are otherwise the same as in the base SiT velocity model.

Initialization and exact linear startup. The path network P_ψ and subtractor S are initialized from the same pretrained SiT checkpoint. Loading the checkpoint into P_ψ proceeds as follows. The vanilla SiT input stem is copied into the linear stem $\mathrm{Embed}_{\mathrm{lin}}$. The timestep embedder, label embedder, positional embedding, transformer blocks, and final layer are copied unchanged. The new delta stem Embed_Δ is zero-initialized, and the final linear layer of the endpoint-conditioning projector is also zero-initialized. Consequently, at initialization,

$$P_\psi(\hat{\gamma}_{\mathrm{lin}}(t), \hat{\Delta}(t), t, y) = S(\hat{\gamma}_{\mathrm{lin}}(t), t, y), \tag{36}$$

provided both networks are loaded from the same checkpoint. Therefore the bracketed residual in (31) is exactly zero and

$$\gamma_\psi^{\mathrm{impl}}(t; x_0, x_1, y, \epsilon_0) = \gamma_{\mathrm{lin}}(t; x_0, x_1)$$

at the start of path training. This exact startup property is why we use the path network and subtractor pair rather than directly learning an unconstrained displacement from scratch.

D Diagnostic correlations for path overfitting

In Figures 1–2 and 5–7, we use the same settings as SiT-S in Table 1 other than changes to β and ρ noted in each figure's legends. We provide more detailed descriptions for Figures 1–2: curves with $\rho_t = 1$ remove stochastic source regularization and vary kinetic regularization through β, while $\rho_t = 0.5$ and $\rho_t = 1 - t$ are stochastic variants at $\beta = 0.05$. Stronger kinetic regularization alone reduces (but does not prevent) late degradation.

This appendix provides the scatter-plot analysis behind the diagnostic discussion in Sections 3.5 and 4.1. All points are SiT-S runs evaluated with 50k FID. Each plot contains checkpoints after 4k, 8k, and 12k flow-training steps, with marker shape indicating the step and color indicating the algorithmic variant. The variants sweep the kinetic energy regularization strength β, the source-perturbation schedule ρ_t, and an additional acceleration-regularized ablation. The acceleration-regularized run adds

$$\lambda\, \mathbb{E} \int_0^1 \left\| \partial_t^2 \gamma_\psi(t; x_0, x_1, y) \right\|_2^2 \, dt$$

to the path objective, with $\lambda = 2 \times 10^{-5}$. Larger values of λ performed poorly in our preliminary sweeps and are not included here. The vertical axis is truncated to the FID range $[8, 11]$, which is the range of interest; higher FID values correspond to extremely bad path-overfitting runs.

We emphasize these plots as qualitative diagnostics rather than as a statistical correlation study. The number of runs is small and the points are coupled through common checkpoints and training schedules. Nevertheless, the comparison is useful: among the measured quantities, the last-bucket divergence gives the clearest visual correlation with FID.

Figure 5 shows the strongest qualitative relationship. The unregularized or weakly regularized checkpoints that degrade in FID develop a large positive final-bin divergence, consistent with the entropy-bottleneck picture in Section 4.1. In contrast, the stochastic source-regularized checkpoints maintain low FID while avoiding a large positive late-time divergence spike. The acceleration-regularized baseline can reduce some geometric quantities, but it does not reliably remove this late positive-divergence behavior.

Taken together, Figures 5–7 highlight that the divergence spike appears to be a stronger predictor of path overfitting, compared to other geometric quantities such as speed and acceleration of the path or the flow. The late-time divergence statistic is more directly tied to the entropy-rate identity of Proposition 2, and in this sweep it is the clearest diagnostic of FID degradation.

E Proofs for Key Results

E.1 Proof of Proposition 1

Proof of Proposition 1. We first prove the population regression statement. All expectations are taken with respect to the chosen endpoint coupling conditional on y, the auxiliary randomness ξ, and $t \sim \mathrm{Unif}[0, 1]$. Let

$$\mathcal{G} = \sigma(\gamma(t; x_0, x_1, y, \xi), t, y)$$

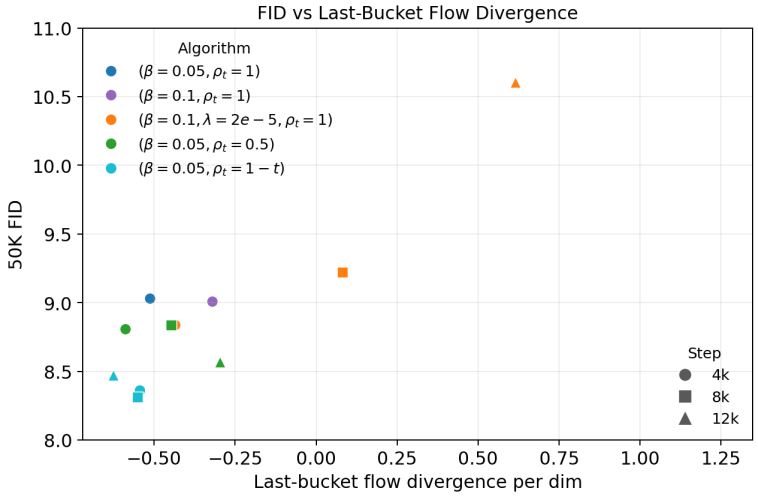
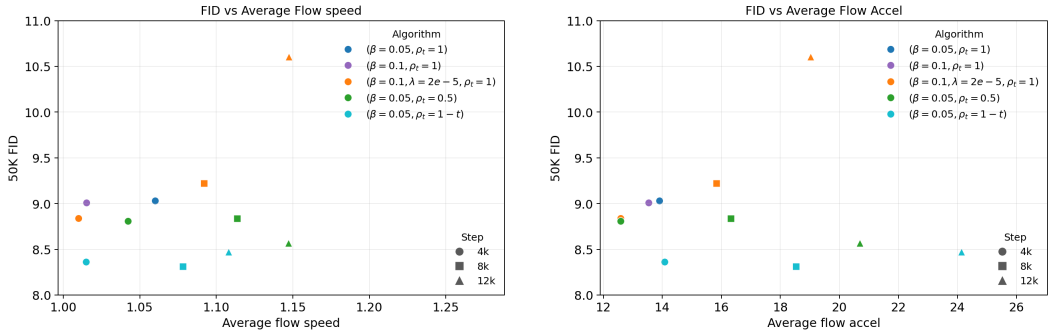


Figure 5: FID versus per-dimensional flow divergence in the last time bucket. The last bucket corresponds to the final bin of the ten-bin estimator in (11). Runs with large positive last-bucket divergence are precisely the high-FID overfit checkpoints in this sweep, while the best checkpoints keep the final-bin divergence negative or only mildly positive.



(a) Average flow Euclidean norm.

(b) Average flow acceleration.

Figure 6: FID versus global flow-geometry summaries. Unlike the last-bucket divergence in Figure 5, average flow speed and acceleration are not monotone indicators of sample quality. Some low-FID stochastic source-regularized checkpoints have comparable or larger average flow speed and acceleration than higher-FID unregularized checkpoints.

be the σ-algebra generated by the point on the path, the time, and the label. For any square-integrable vector field v_ϕ, the random variable

$$v_\phi(\gamma(t; x_0, x_1, y, \xi), t, y)$$

is $\mathcal{G}$-measurable. By the tower property,

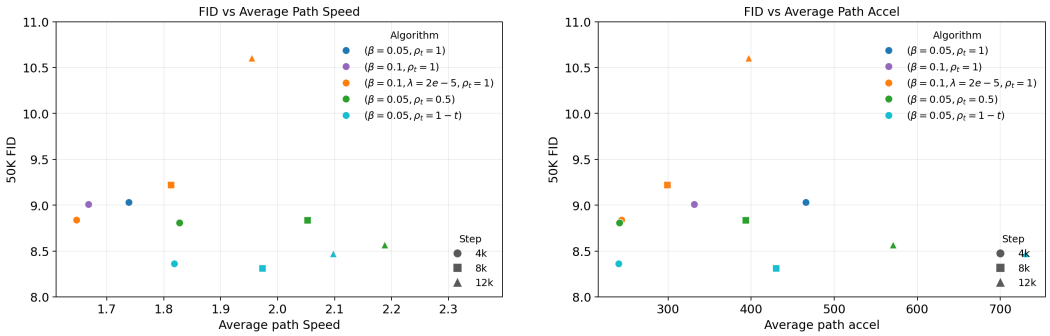
$$\begin{aligned} \mathcal{L}_{\text{flow}}(v_\phi \mid \gamma) &= \mathbb{E}\Big[\|v_\phi(\gamma(t; x_0, x_1, y, \xi), t, y) - \dot{\gamma}(t; x_0, x_1, y, \xi)\|_2^2 \Big] \\ &= \mathbb{E}\Big[\mathbb{E}\big[\|v_\phi(\gamma(t; x_0, x_1, y, \xi), t, y) - \dot{\gamma}(t; x_0, x_1, y, \xi)\|_2^2 \;\big|\; \mathcal{G} \big] \Big]. \end{aligned} \tag{37}$$

Fix a realization of $\mathcal{G}$, say

$$\gamma(t; x_0, x_1, y, \xi) = x, \qquad t = s, \qquad y = \bar{y}.$$

Conditional on this event, the value of the vector field is a deterministic vector $a = v_\phi(x, s, \bar{y})$. The conditional risk is

$$\mathbb{E}\Big[\|a - \dot{\gamma}(t; x_0, x_1, y, \xi)\|_2^2 \;\Big|\; \gamma(t; x_0, x_1, y, \xi) = x,\ t = s,\ y = \bar{y} \Big].$$



(a) Average path Euclidean norm.

(b) Average path acceleration.

Figure 7: FID versus global path-geometry summaries. The path-speed and path-acceleration summaries also do not explain the FID ranking on their own. In particular, good stochastic source-regularized checkpoints can have larger path speed or acceleration than worse unregularized checkpoints. This supports the view that the key failure mode is distributional—a late entropy-expansion spike—rather than simply excessive path length or curvature.

Let

$$m(x, s, \bar{y}) = \mathbb{E}[\dot{\gamma}(t; x_0, x_1, y, \xi) \mid \gamma(t; x_0, x_1, y, \xi) = x,\ t = s,\ y = \bar{y}]\,.$$

Then

$$\mathbb{E}\Big[\|a - \dot{\gamma}(t; x_0, x_1, y, \xi)\|_2^2 \;\Big|\; \gamma(t; x_0, x_1, y, \xi) = x,\ t = s,\ y = \bar{y}\Big]$$
$$= \|a - m(x, s, \bar{y})\|_2^2 + \mathbb{E}\Big[\|\dot{\gamma}(t; x_0, x_1, y, \xi) - m(x, s, \bar{y})\|_2^2 \;\Big|\; \gamma(t; x_0, x_1, y, \xi) = x,\ t = s,\ y = \bar{y}\Big]\,,$$

because the conditional mean of $\dot{\gamma}(t; x_0, x_1, y, \xi) - m(x, s, \bar{y})$ is zero under the same conditioning. The second term does not depend on a, while the first term is uniquely minimized by $a = m(x, s, \bar{y})$. Therefore the population minimizer is

$$v_\gamma^\star(x, t, y) = \mathbb{E}[\dot{\gamma}(t; x_0, x_1, y, \xi) \mid \gamma(t; x_0, x_1, y, \xi) = x,\ t,\ y]$$

for $p_{\gamma,t}(\cdot \mid y)\,dt$-almost every (x, t) and for the relevant labels y. This proves the first part of the proposition. Notice that no endpoint-preservation assumption was used.

The same conditional-expectation argument also gives the projection identity

$$\begin{aligned}\mathcal{L}_{\text{flow}}(v_\phi \mid \gamma) = \mathbb{E}\Big[\,\big\|v_\phi(\gamma(t; x_0, x_1, y, \xi), t, y) - v_\gamma^\star(\gamma(t; x_0, x_1, y, \xi), t, y)\big\|_2^2\,\Big] \\ + \mathbb{E}\Big[\,\big\|\dot{\gamma}(t; x_0, x_1, y, \xi) - v_\gamma^\star(\gamma(t; x_0, x_1, y, \xi), t, y)\big\|_2^2\,\Big].\end{aligned} \tag{38}$$

Indeed, expand

$$\begin{aligned}&\dot{\gamma}(t; x_0, x_1, y, \xi) - v_\phi(\gamma(t; x_0, x_1, y, \xi), t, y) \\ &\quad = \big(\dot{\gamma}(t; x_0, x_1, y, \xi) - v_\gamma^\star(\gamma(t; x_0, x_1, y, \xi), t, y)\big) \\ &\qquad + \big(v_\gamma^\star(\gamma(t; x_0, x_1, y, \xi), t, y) - v_\phi(\gamma(t; x_0, x_1, y, \xi), t, y)\big).\end{aligned}$$

The cross term has expectation zero because the second factor is $\mathcal{G}$-measurable and

$$\mathbb{E}\big[\dot{\gamma}(t; x_0, x_1, y, \xi) - v_\gamma^\star(\gamma(t; x_0, x_1, y, \xi), t, y) \mid \mathcal{G}\big] = 0.$$

This proves (38).

We now prove the ODE transport statement. Fix y and assume the additional hypotheses in the second part of Proposition 1. In particular, the Borel, Lipschitz, and linear-growth conditions imply that $v_\gamma^\star(\cdot, \cdot, y)$ generates a unique Carathéodory flow. Let $f \in C_c^\infty(\mathbb{R}^d)$ be a test function. Differentiating under the expectation gives

$$\frac{d}{dt}\mathbb{E}[f(\gamma(t; x_0, x_1, y, \xi)) \mid y] = \mathbb{E}[\langle \nabla_x f(\gamma(t; x_0, x_1, y, \xi)), \dot{\gamma}(t; x_0, x_1, y, \xi)\rangle \mid y].$$

Conditioning on $\gamma(t; x_0, x_1, y, \xi) = x$ and using the definition of $v^\star_\gamma$,

$$\frac{d}{dt}\int f(x)p_{\gamma,t}(x \mid y)\,dx = \int \left\langle \nabla_x f(x), v^\star_\gamma(x,t,y)\right\rangle p_{\gamma,t}(x \mid y)\,dx.$$

This is the weak form of the continuity equation associated with the velocity field $v^\star_\gamma$. The law of the Carathéodory flow

$$\frac{d}{dt}z(t) = v^\star_\gamma(z(t), t, y)$$

satisfies the same continuity equation. By endpoint preservation and $x_0 \sim p_0$, the path marginal at $t = 0$ is p_0, so the ODE and the path have the same initial law. Under the Lipschitz and linear-growth assumptions of Proposition 1, the associated probability solution is unique. Hence the ODE marginals coincide with the path marginals for every $t \in [0, 1]$. Finally, endpoint preservation and $x_1 \sim p_1(\cdot \mid y)$ imply that the path marginal at $t = 1$ is $p_1(\cdot \mid y)$, so $z(1) \sim p_1(\cdot \mid y)$. □

E.2 Proof of Proposition 2

Proof. The differential entropy is

$$h(p_t(\cdot \mid y)) = -\int p_t(x \mid y)\log p_t(x \mid y)\,dx.$$

Differentiating and using $\int \partial_t p_t(x \mid y)\,dx = 0$ gives

$$\frac{d}{dt}h(p_t(\cdot \mid y)) = -\int \partial_t p_t(x \mid y)\log p_t(x \mid y)\,dx.$$

Substituting $\partial_t p_t(x \mid y) + \nabla_x \cdot (p_t(x \mid y)u_t(x, y)) = 0$, yields

$$\frac{d}{dt}h(p_t(\cdot \mid y)) = \int \nabla_x \cdot (p_t u_t)\log p_t\,dx.$$

Integrating by parts and using the decay assumption,

$$\frac{d}{dt}h(p_t(\cdot \mid y)) = -\int p_t(x \mid y)u_t(x, y)\cdot\nabla_x \log p_t(x \mid y)\,dx = \int p_t(x \mid y)\nabla_x \cdot u_t(x, y)\,dx.$$

This proves (12). Integrating over $[a, b]$ proves (13). □

E.3 Proof of Theorem 1

Proof. For the fixed time t, the pair $(x_0, \hat{x}_0(t))$ is jointly Gaussian and satisfies

$$\mathrm{Cov}(x_0) = I_d, \qquad \mathrm{Cov}(\hat{x}_0(t)) = I_d, \qquad \mathrm{Cov}(x_0, \hat{x}_0(t)) = \rho_t I_d.$$

The variable η_t in (23) is therefore jointly Gaussian with $\hat{x}_0(t)$. A direct covariance calculation gives

$$\mathrm{Cov}(\eta_t) = I_d, \qquad \mathrm{Cov}(\eta_t, \hat{x}_0(t)) = 0.$$

Hence, conditional on y, $\eta_t \sim \mathcal{N}(0, I_d)$ and η_t is independent of $\hat{x}_0(t)$. Under the independent endpoint coupling, x_1 is independent of (x_0, ϵ) conditional on y, so η_t is conditionally independent of $(\hat{x}_0(t), x_1)$ given y. Rearranging (23) gives

$$x_0 = \rho_t \hat{x}_0(t) + \sqrt{1 - \rho_t^2}\,\eta_t.$$

Substituting this identity into (18) yields (24). Since the last term $\sigma_t\eta_t$ is Gaussian and independent of the non-Gaussian term (25), the conditional law of their sum is the convolution (22).

The entropy floor follows by conditioning on the non-Gaussian term in (25). After this conditioning, the only remaining randomness in the state is the independent Gaussian $\sigma_t\eta_t$, whose entropy is

$$\frac{d}{2}\log(2\pi e\sigma_t^2).$$

Removing the conditioning can only increase entropy by a nonnegative mutual-information term, giving (26).

It remains to explain the Fisher-information bound. Let φ_{σ_t} denote the density of $\mathcal{N}(0, \sigma_t^2 I_d)$. By the convolution representation,

$$\hat{p}_{\psi,t}(x \mid y) = \int \varphi_{\sigma_t}(x - r)\, dq_{\psi,t}(r \mid y).$$

Differentiating under the integral gives

$$\nabla_x \hat{p}_{\psi,t}(x \mid y) = -\frac{1}{\sigma_t^2} \int (x - r)\varphi_{\sigma_t}(x - r)\, dq_{\psi,t}(r \mid y).$$

Equivalently, using the representation $\hat{\gamma}_\psi(t) = r + \sigma_t \eta_t$ from (24),

$$\nabla_x \log \hat{p}_{\psi,t}(x \mid y) = -\frac{1}{\sigma_t} \mathbb{E}\left[\eta_t \mid \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) = x,\, y\right]. \tag{39}$$

Therefore Jensen's inequality gives

$$\begin{aligned} I(\hat{p}_{\psi,t}(\cdot \mid y)) &= \frac{1}{\sigma_t^2} \mathbb{E}\left[\|\mathbb{E}[\eta_t \mid \hat{\gamma}_\psi(t), y]\|_2^2 \mid y\right] \\ &\le \frac{1}{\sigma_t^2} \mathbb{E}[\|\eta_t\|_2^2] = \frac{d}{\sigma_t^2}. \end{aligned}$$

This proves (27). □

E.4 Proof of Proposition 3

Proof. By the definition of the stochastic-path population target in (20), the ideal marginal velocity is

$$\hat{v}_\psi^\star(x, t, y) = \mathbb{E}\left[\partial_t \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) \mid \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) = x,\, t,\, y\right].$$

Applying Proposition 2 to the probability flow generated by $\hat{v}_\psi^\star$ gives

$$\frac{d}{dt} h(\hat{p}_{\psi,t}(\cdot \mid y)) = \mathbb{E}_{x \sim \hat{p}_{\psi,t}(\cdot|y)}[\nabla_x \cdot \hat{v}_\psi^\star(x, t, y)].$$

Integrating by parts,

$$\begin{aligned} \frac{d}{dt} h(\hat{p}_{\psi,t}(\cdot \mid y)) &= -\mathbb{E}_{x \sim \hat{p}_{\psi,t}(\cdot|y)}\left[\left\langle \hat{v}_\psi^\star(x, t, y), \nabla_x \log \hat{p}_{\psi,t}(x \mid y) \right\rangle\right] \\ &\ge -\sqrt{\mathbb{E}_{x \sim \hat{p}_{\psi,t}(\cdot|y)} \left\| \hat{v}_\psi^\star(x, t, y) \right\|_2^2} \sqrt{I(\hat{p}_{\psi,t}(\cdot \mid y))}, \end{aligned}$$

where the inequality is Cauchy–Schwarz.

The first factor is controlled by the path kinetic energy. Indeed, by Jensen's inequality for conditional expectation,

$$\begin{aligned} \mathbb{E}_{x \sim \hat{p}_{\psi,t}(\cdot|y)} \left\| \hat{v}_\psi^\star(x, t, y) \right\|_2^2 &\le \mathbb{E}\left[\left\| \partial_t \hat{\gamma}_\psi(t; x_0, x_1, y, \epsilon) \right\|_2^2 \,\middle|\, y \right] \\ &= L_{\mathrm{euc}}^{\rho,\psi}(t, y). \end{aligned}$$

The second factor is controlled by Gaussian smoothing: Theorem 1 gives

$$I(\hat{p}_{\psi,t}(\cdot \mid y)) \le \frac{d}{(1 - t)^2 (1 - \rho_t^2)}.$$

Combining the last three displays proves (29). □

F Experimental details

We implement Algorithm 1 with flow v_ϕ initialized from already-trained SiT [13] checkpoints of the appropriate size and path (P_ψ and S) initialized from pretrained checkpoints. Our main experiments use ImageNet-256×256 [5]. Our setting is post-training of a pretrained flow model. For SiT-S/B/L, we finetune the pretrained checkpoints until FID plateaus and use the resulting checkpoint to initialize both the standard continuation baseline and our method. For SiT-XL, finetuning does not improve the pretrained checkpoint FID, so the pretrained checkpoint is the initialization for the flow network v_ϕ.

Our path network R_ψ is constructed from a pre-trained flow network, as described in Appendix C. For the FM experiments in Table 1, we use the pretrained SiT-S checkpoint to initialize P_ψ and S. For the MG experiments in Table 1, we use the pretrained SiT-B-MG checkpoint to initialize P_ψ and S.

Experimental details for FM experiments in Table 1. In each cycle we finetune path R_ψ using maximum learning rate 3×10^{-4} with cosine decay schedule for 3000 steps, batch size 280, $\beta = 0.05$, and $\rho_t = c \cdot (1 - t)$ where $c = 1$ for SiT-S,B,L and $c = 0.7$ for SiT-XL. In each cycle we finetune flow v_ϕ using constant learning rate 5×10^{-6} for 4000 steps with no EMA, batch size 768, and inherit the same ρ_t from the path tuning stage. To ensure a fair comparison, we apply standard finetuning to the pretrained baseline SiT models using both the default pretraining settings reported in SiT [13] and our settings for finetuning the flow network v_ϕ, and report the best FID for the baseline given a fixed finetuning budget. Applying standard finetuning to SiT-S,B,L gives improvements to the FID, whereas finetuning SiT-XL results in worse FID. Experiments are conducted on H200 GPUs. We compute FID [6] and Inception-Score (IS) [16] on 50k images for each checkpoint using the Heun ODE sampler at 250 NFE. We tune the classifier-free guidance strength (CFG) [7] separately for the SiT baseline and our method, generally noticing higher CFG compared to the baseline working better for Algorithm 1. We use CFG $= \{3, 2, 1.5, 1.5\}$ for baseline SiT-S,B,L,XL models, and CFG = $\{3.5, 2.5, 2.0, 1.7\}$ for our SiT-S,B,L,XL models. To ensure a fair comparison, we sweep CFG on the same grid for both the SiT baseline and our method, and report FID with best CFG.

Experimental details for MG experiments in Table 1. For the model-guidance experiments, Algorithm 1 uses three alternating cycles, each with 3k path-tuning steps followed by 4k MG-flow-tuning steps. Both SiT-B-MG and SiT-XL-MG use $\beta = 0.05$, flow EMA decay 0.999, and MG-flow tuning with batch size 768 and learning rate 5×10^{-6}. For both models, the path network P_ψ and subtractor S are initialized from the pretrained SiT-B-MG checkpoint and use batch size 140. The SiT-B-MG run uses a constant source-perturbation $\rho_t = 0.5$ and path learning rate 3×10^{-4}, while SiT-XL-MG uses $\rho_t = 0.7(1 - t)$ and the path learning rate is 5×10^{-5}. For a fair comparison, the standard MG fine-tuning baselines use the same batch size and learning rate as MG-flow tuning. All entries are evaluated with 50k generated samples using the Heun ODE sampler with 250 NFE and CFG $= 1.0$.

Table 3: Full ImageNet-256×256 results corresponding to Table 1. The baseline columns report the initial checkpoint and the best FID reached by standard fine-tuning within the 12k continuation budget. Our method is evaluated after each alternating cycle.

Model	Params	FID (baseline) ↓		FID (ours) ↓			IS (baseline) ↑	IS (ours) ↑		
		Init	Best	4k	8k	12k	Init	4k	8k	12k
SiT-S	33M	9.109	9.022	8.375	**8.359**	8.492	173.6	**188.9**	187.2	185.4
SiT-B	130M	5.074	5.057	4.703	4.480	**4.415**	199.0	**234.9**	229.0	225.5
SiT-L	458M	3.424	3.390	3.014	2.882	**2.742**	201.1	**274.8**	271.5	267.2
SiT-XL	675M	2.091	2.091	1.909	1.845	**1.825**	256.1	282.9	285.6	**288.6**

Table 4: Checkpoint-wise model-guidance FID results corresponding to Table 1. Both methods start from the same model-guidance checkpoint; lower is better.

Model	Method	0k	4k	8k	12k
SiT-B-MG (130M)	Ours	8.090	6.704	5.844	**5.585**
	MG baseline	8.090	8.011	7.940	7.950
SiT-XL-MG (675M)	Ours	1.522	1.336	**1.322**	1.333
	MG baseline	1.522	1.494	1.504	1.501

Table 5: Training efficiency measured in steps per second per H200 GPU.

Model	Alg 1 (Path Phase)	Alg 1 (Flow Phase)	Standard flow matching
SiT-S	1.86	1.39	4.65
SiT-B	1.70	1.02	2.08
SiT-L	0.85	0.51	0.71
SiT-XL	0.70	0.37	0.50

Efficiency. Table 5 compares the efficiency of training flow and path networks in Algorithm 1 to standard flow matching training. The additional cost of using R_ψ during flow training causes our flow training to be less efficient per step. However, the relative cost of R_ψ decreases at larger flow models because R_ψ has constant size across S/B/L/XL. All experiments are conducted on H200 GPUs, and our code supports multi-GPU training and image generation. Appendix F.1 reports complete GPU-hour- and FLOP-matched experiments.

F.1 Compute-matched baseline continuation

Using our method, three training cycles contain 9k path updates and 12k learned-path flow updates. For SiT-S/B/XL, these cycles cost 1.435/2.546/8.811 EFLOPs, whereas 12k updates of standard continuation cost 0.335/1.272/6.559 EFLOPs. We extend each baseline to the larger of the GPU-hour- and FLOP-matched budgets. GPU-hour matching gives the larger budget for all three models sizes.

Table 6: GPU-hour matching for three training cycles. GPU-hours are measured on H200 GPUs.

Model	Ours GPU-h (path + flow)	Matched baseline steps	Baseline GPU-h
SiT-S	3.574 (1.210 + 2.364)	61.1k	3.579
SiT-B	4.667 (1.468 + 3.199)	36.6k	4.677
SiT-XL	12.216 (3.546 + 8.670)	22.8k	12.243

Table 7: Compute-matched ImageNet-256×256 continuation. FID is computed from 50k images using the Heun sampler at 250 NFE. The reduction is relative to the stronger standard baseline.

Model	Initialization	Baseline best	Compute-matched	Ours	Reduction
SiT-S	9.109	9.022	9.025	**8.359**	7.3%
SiT-B	5.074	5.057	4.981	**4.415**	11.4%
SiT-XL	2.091	2.091	2.147	**1.825**	12.7%

Longer standard continuation improves only SiT-B beyond its original baseline, while path–flow training gives lower FID for all three model sizes. The inference architecture, NFE, and sampling cost are unchanged. We also evaluated the performance of 12k-step continuation with the original SiT hyperparameters. For SiT-B, our tuned baseline reaches 5.057 FID, compared with 5.533 under the original hyperparameters and 5.074 at initialization. For SiT-XL, the two settings reach 2.134 and 2.095, respectively, compared with 2.091 at initialization.

The cost of path–flow alignment applied to a pretrained checkpoint is small relative to pretraining: SiT-XL/2 pretraining uses an estimated 1275.4 EFLOPs, more than $140\times$ the 8.811 EFLOPs used by three cycles of path–flow training.

F.2 CFG selection and additional metrics

We select classifier-free guidance (CFG) independently for each baseline and path–flow checkpoint using 10k generated images and the adaptive Dopri5 sampler. Note that headline evaluations use 50k-FID.

Table 8: FID on the 10k-image CFG-selection sweeps. Bold and underline identify the selected value for each checkpoint.

	SiT-S			
	CFG 2	CFG 3	CFG 3.5	CFG 4
Baseline	18.742	**13.094**	13.099	13.967
Ours	20.364	11.463	**10.988**	11.439

	SiT-B			
	CFG 1	CFG 2	CFG 2.5	CFG 3
Baseline	31.481	**8.794**	9.157	11.126
Ours	43.778	9.576	**7.286**	8.051

	SiT-L			
	CFG 1	CFG 1.5	CFG 2	CFG 2.5
Baseline	17.612	**6.992**	7.745	10.828
Ours	24.455	8.050	**5.606**	7.666

	SiT-XL			
	CFG 1	CFG 1.5	CFG 1.7	CFG 2
Baseline	12.446	**4.977**	5.271	6.771
Ours	14.796	5.062	**4.787**	5.651

The sweeps select CFG 3/2/1.5/1.5 for the SiT-S/B/L/XL baselines and 3.5/2.5/2/1.7 for the corresponding path–flow checkpoints. We use the best CFG settings for each model size and method in our 50k-FID headline results.

In Table 9 we additionally report precision and recall corresponding to the SiT checkpoints used for headline results in Table 1. As baseline, we generate using the model with the best FID at every SiT model size.

Table 9: Precision and recall corresponding to the 50k-image headline FID results in Table 1, generated using Heun at 250 NFE. Bold marks the better value within each model and metric.

	SiT-S	
	Precision ↑	Recall ↑
Baseline	**0.874**	0.283
Ours	0.860	**0.297**

	SiT-B	
	Precision ↑	Recall ↑
Baseline	**0.850**	0.441
Ours	0.826	**0.474**

	SiT-L	
	Precision ↑	Recall ↑
Baseline	**0.819**	0.542
Ours	0.806	**0.564**

	SiT-XL	
	Precision ↑	Recall ↑
Baseline	**0.814**	0.581
Ours	0.791	**0.611**

Recall increases when comparing our method to the baseline, while precision decreases.

To examine how both metrics vary with guidance, Table 10 reports how precision and recall change with respect to CFG using 10k images. Bold marks the best value in each row, and underline marks the CFG selected by FID for the headline evaluation.

Table 10: Precision and recall across 10k-image CFG sweeps using the Dopri5 sampler.

		SiT-S			
Metric	Method	CFG 2	CFG 3	CFG 3.5	CFG 4
Precision ↑	Baseline	0.850	**0.875**	0.872	0.868
	Ours	0.831	**0.858**	0.857	0.852
Recall ↑	Baseline	0.360	**0.364**	0.354	0.332
	Ours	0.385	**0.433**	0.416	0.407
		SiT-B			
Metric	Method	CFG 1	CFG 2	CFG 2.5	CFG 3
Precision ↑	Baseline	0.766	**0.853**	0.848	0.840
	Ours	0.732	0.811	**0.824**	0.815
Recall ↑	Baseline	0.380	0.523	**0.525**	0.506
	Ours	0.326	0.559	0.578	**0.580**
		SiT-L			
Metric	Method	CFG 1	CFG 1.5	CFG 2	CFG 2.5
Precision ↑	Baseline	0.776	0.816	**0.822**	0.813
	Ours	0.745	0.788	**0.798**	0.794
Recall ↑	Baseline	0.518	0.606	**0.616**	0.604
	Ours	0.484	0.601	0.655	**0.663**
		SiT-XL			
Metric	Method	CFG 1	CFG 1.5	CFG 1.7	CFG 2
Precision ↑	Baseline	0.776	**0.811**	0.809	0.808
	Ours	0.752	0.789	**0.795**	0.790
Recall ↑	Baseline	0.564	0.653	0.663	**0.671**
	Ours	0.576	0.671	0.688	**0.693**

At the CFG selected for each path–flow checkpoint, recall changes from 0.354/0.525/0.616/0.663 to 0.416/0.578/0.655/0.688 for SiT-S/B/L/XL, while precision changes from 0.872/0.848/0.822/0.809 to 0.857/0.824/0.798/0.795. At the lowest tested CFG, path–flow training has lower precision for all four sizes; recall is lower for SiT-B/L and slightly higher for SiT-S/XL.

Multi-seed evaluation. In Table 11 we evaluate each fixed checkpoint on three sets of images generated from three disjoint sets of random seeds. The ranges for the random seeds are 0-49,999, 50k-99,999, 100k-149,999.

The mean FID is lower for path–flow training at every model size. Each gap is at least $5.8\times$ the larger sample standard deviation observed for that model.

Table 11: Mean $\pm$ sample standard deviation of 50k-image FID across three generation-seed ranges, using Heun at 250 NFE.

Model	Baseline best	Ours best
SiT-S	9.142 ± 0.13	$\mathbf{8.383 \pm 0.041}$
SiT-B	5.093 ± 0.041	$\mathbf{4.470 \pm 0.054}$
SiT-L	3.458 ± 0.061	$\mathbf{2.777 \pm 0.031}$
SiT-XL	2.103 ± 0.026	$\mathbf{1.830 \pm 0.015}$

F.3 Applying our method in more scenarios

Using our method on FlowDCN. To broaden the applicability of our method, we apply three cycles of path–flow training to a pretrained 120M-parameter FlowDCN-B model [20], which has a purely convolutional backbone. We select CFG using 10k-FID (dopri5 sampler, see Table 12), and compare with 25.9k steps of FLOP-matched standard continuation. Final FID is computed from 50k images using the Heun sampler at 250 NFE.

Table 12: FlowDCN-B 10k-image CFG sweep using the 100-step adaptive Dopri5 sampler.

Checkpoint	CFG 1.5	CFG 2.0	CFG 2.5
Baseline	11.890	**9.142**	11.230
Ours	15.734	**8.204**	8.765

The sweep selects CFG 2.0 for both checkpoints. At the chosen CFG, Table 13 compares the performance of the pretrained model, compute-matched continuation, and our method's three-cycle finetuned checkpoint. Path–flow training reaches 5.473 FID, a 7.0% improvement over the best FLOP-matched continuation result of 5.888, demonstrating that our path–flow alignment method can be applied to various architectures in addition to SiT [13].

Table 13: Performance of path–flow alignment using FlowDCN-B (Ours best) compared to continued finetuning (Baseline best), 50k-FID (Heun sampler 250 NFE).

Model	Params	Init FID	Baseline best	Ours best	FID gain
FlowDCN-B	120M	6.489	5.888	**5.473**	7.0%

Comparison to [17] on CIFAR-10. We compare with Learning Straight Flows [17] by initializing both our path–flow alignment method and [17]'s method from a SiT-CIFAR/2 checkpoint pretrained for 200k steps. This is a smaller variant of SiT adapted for CIFAR-10 that we pretrained. To ensure fair comparisons, we use matched training FLOPs: our path–flow alignment uses 32.888 PFLOPs, whereas our reproduction of Learning Straight Flows [17] and CIFAR-10 baseline continuation use at least as much FLOPs as was allocated to our method. We compare the methods using 10k-FID at CFG 2.0 and the ODE Euler sampler at 2 and 40 NFE.

Our implementation of Learning Straight Flows [17] includes its empirical target-field estimator, joint flow and straightness objective, and a four-block GLOW/PLU invertible interpolant. We use 100 queries and a balanced 500-image candidate pool. Since we are unable to access the original code and some CIFAR-10 hyperparameters are unavailable, we are unable to reproduce [17] exactly.

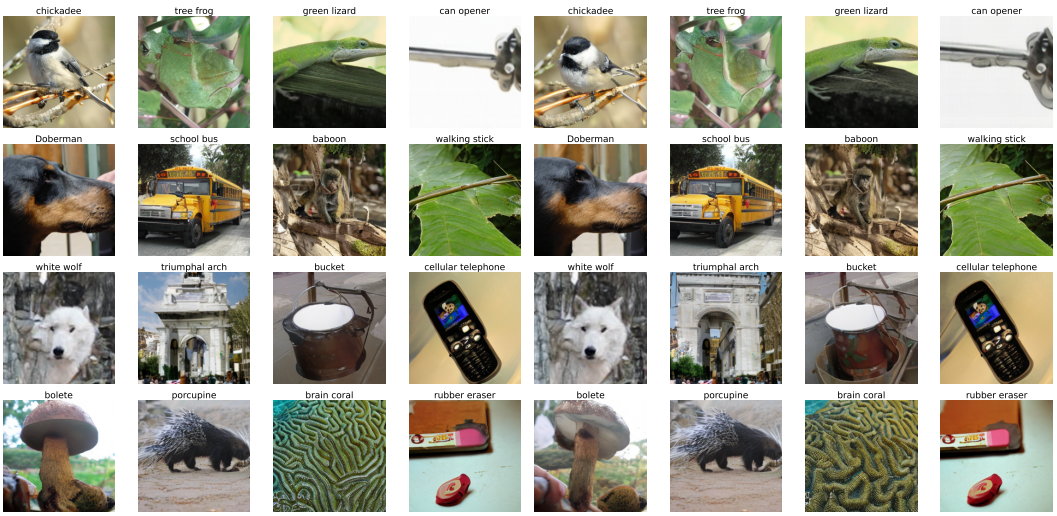
The standard-continuation row reports the best FID at each NFE achieved during continued finetuning. We report the FID at the end of training for our path–flow alignment and [17]. Learning Straight Flows gives the best result at 2 NFE, while path–flow alignment gives the best result at 40 NFE. Intuitively, our method learns a more curved trajectory (see Figure 9) whereas [17] penalizes the curvature of the resulting trajectory at the cost of high-NFE generation, leading our method to perform better at higher NFE.

Table 14: FLOP-matched CIFAR-10 comparison using 10k images from one generation seed. Lower FID is better.

Method	FID, 2 NFE ↓	FID, 40 NFE ↓
Our method (1 cycle)	170.845	**9.097**
Baseline continuation	100.518 (step 1,000)	10.339 (step 1,500)
[17], $\lambda_{\max} = 0.001$, GLOW LR 5×10^{-6}	**79.150**	11.667
[17], $\lambda_{\max} = 0.001$, GLOW LR 5×10^{-7}	84.230	11.568
[17], $\lambda_{\max} = 0.0003$, GLOW LR 5×10^{-7}	94.295	11.878

F.4 Qualitative comparisons with our method

Qualitative samples. Figure 8 compares images generated by the pretrained SiT-XL checkpoint with the cycle-3 checkpoint of our path–flow alignment. The grids are uncurated and matched by class and generation seed.



(a) Best SiT-XL baseline (b) Ours after three cycles

Figure 8: Uncurated SiT-XL samples matched by class and generation seed. Left: the best baseline, given by the framework initialization. Right: path–flow alignment after three cycles.

Most corresponding samples remain similar in class, composition, pose, and background, indicating that our path–flow alignment method refines the pretrained model rather than substantially changing its overall behavior.

Several samples nevertheless show **clearer local structure with our method**. At (1,3), the green lizard has more clearly defined feet and contact with the surface. At (2,3), the baboon has a more coherent head and body. At (3,1), the white wolf has a better-formed face. At (3,2), our result is recognizably a triumphal arch, whereas the baseline resembles an irregular building with a small opening. At (4,4), the rubber eraser is cleanly separated from the surrounding material rather than bleeding into it.

Visualizing learned trajectories. Figure 9 visualizes the trajectories from a toy experiment under standard flow matching, unregularized path learning, and stochastic path learning. Our method induces a trajectory with higher curvature compared to standard flow matching.

Our toy example transports an independently coupled two-component Gaussian mixture centered at $(-5, \pm 1)$ to one centered at $(5, \pm 1)$; each component has isotropic standard deviation 0.3. We first train a standard flow-matching teacher on linear paths, then initialize the $\rho = 1$ and $\rho_t = 1 - t$ branches from this teacher and run two cycles of alternating path and flow updates. The plotted

curves integrate the same 600 source samples with Heun over 240 steps, allowing their geometry to be compared directly.

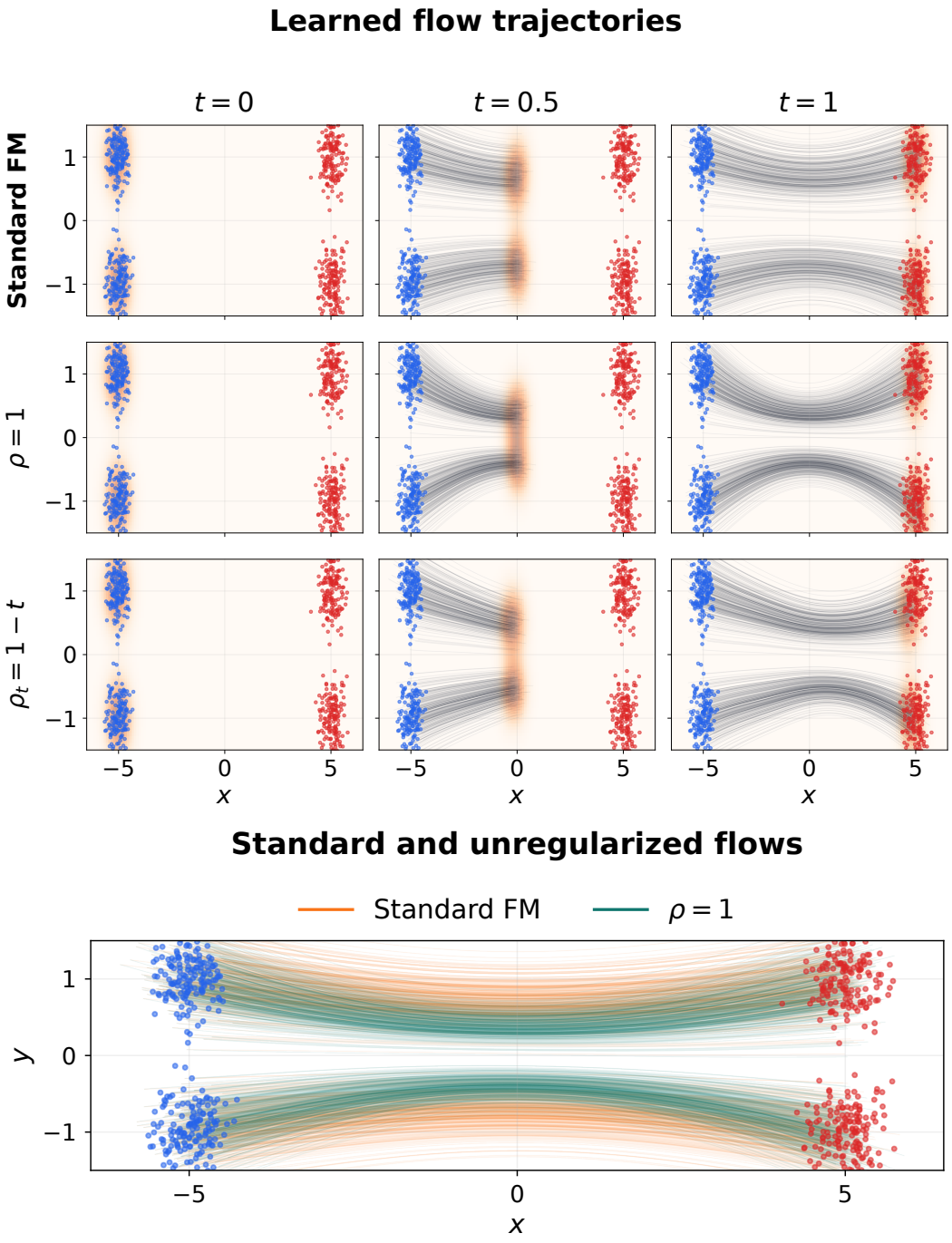


Figure 9: Learned trajectories in the toy experiment. Top: standard flow matching, unregularized path learning ($\rho = 1$), and stochastic path learning ($\rho_t = 1 - t$). Bottom: standard and unregularized trajectories overlaid.

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: Our abstract describes our contribution, while our introduction highlight our theoretical and experimental results. Theoretical contributions and proofs are found in the main body of the paper and appendix, and experimental results are presented in Section 6.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: See Section B where we discuss the limitations of our paper.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

Answer: [Yes]

Justification: Proofs of theorems are either found in the Appendix or the main body. Our theoretical results describe assumptions.

Guidelines:

- The answer [N/A] means that the paper does not include theoretical results.
- All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
- All assumptions should be clearly stated or referenced in the statement of any theorems.
- The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
- Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
- Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide a repository containing code with instructions on how to reproduce. We provide detailed experimental settings inside of Appendix F.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.
     (c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).
     (d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

Answer: [Yes]

Justification: We use an open source dataset (ImageNet [5]) and provide explanations of how we use other papers' code/resources in our code repository.

Guidelines:

- The answer [N/A] means that paper does not include experiments requiring code.
- Please see the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
- The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (`https://neurips.cc/public/guides/CodeSubmissionPolicy`) for more details.
- The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
- The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
- At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
- Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

Answer: [Yes]

Justification: They can be found in Appendix F and Section 6. Our code includes the hyperparameters and other setup information.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
- The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [No]

Justification: We report generation-seed variability for fixed checkpoints using three disjoint 50k-image seed sets in the Appendix. We do not report the variability across independently trained models due to the high computational costs associated with finetuning.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.

- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We detail the efficiency of our method in Appendix F. We used a mix of cloud providers and local GPU clusters.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics `https://neurips.cc/public/EthicsGuidelines`?

   Answer: [Yes]

   Justification: We use open source datasets and do not perform any experiments that contravene the Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.
   - If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
   - The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: We do not expect our paper to have any societal impact.

    Guidelines:

- The answer [N/A] means that there is no societal impact of the work performed.
- If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
- Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
- The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
- The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
- If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: Our paper does not present risks for misuse, since our class-conditioned image generators are trained on a standard labeled dataset.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We credit the datasets and code bases on which our code is built in the paper as well as our code repository. The ImageNet dataset is available for free to researchers for non-commercial use.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.

- The name of the license (e.g., CC-BY 4.0) should be included for each asset.
- For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
- If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
- For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
- If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: Our code is well-documented and all details are found inside of our code repository.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.
    - According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This paper does not involve crowdsourcing or research with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.

- Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
- We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
- For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: LLM is not used for the core methodology or the originality of the research. We use LLMs to polish text and polish code.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.