# Scaling Flow Matching Models at Inference-Time by Search and Path Exploration

**Anonymous Author(s)**
Affiliation
Address
email

## Abstract

Inference-time compute scaling has become a powerful means of improving discrete generative models, yet its counterpart for continuous-time generative models remains under-explored. We close this gap for flow-matching (FM) models, whose deterministic ordinary-differential (ODE) and stochastic differential (SDE) samplers underpin state-of-the-art continuous generative models in domains such as image-generation and protein-folding. We introduce an inference-time strategy that injects analytically constructed divergence-free perturbations into the learned velocity field during ODE sampling. The perturbations preserve the method's defining linear interpolation path and exactly maintain the continuity equation, ensuring that probability mass is conserved while enabling the sampler to explore high quality trajectories, without modifying trained parameters. We additionally include SDE sampling as an alternative approach. Experiments on ImageNet and FoldFlow demonstrate our methods ability to trade-off computation time with sampling quality over multiple key metrics for each domain. Our work positions inference-time scaling as a principled, training-free lever for enhancing flow-matching models and invites future exploration across diverse continuous generative tasks.

## 1 Introduction

Inference-time compute scaling has emerged as a transformative paradigm in generative modeling, enabling performance improvements through increased computational resources at test time without requiring model retraining. While this approach has shown remarkable success in discrete domains—from language models like OpenAI's O1 **?** to diffusion models for image generation **?**—its application to continuous-time generative models remains largely unexplored.

Flow Matching (FM) **?** has established itself as a leading framework for continuous generative modeling, offering deterministic sampling through ordinary differential equations (ODEs) and achieving state-of-the-art results across diverse domains including image generation **?** and protein design **?**. However, the deterministic nature of FM sampling presents a fundamental challenge for inference-time scaling: unlike diffusion models that naturally provide stochasticity through noise injection, FM models follow a single, predetermined trajectory from noise to data, leaving limited opportunities for exploring alternative generation paths.

This work introduces the first principled approach to inference-time scaling for flow matching models. Our key insight is that we can inject carefully constructed divergence-free perturbations into the learned velocity field during sampling, creating a family of alternative ODE trajectories while preserving the fundamental mathematical properties of the original model. Specifically, our perturbations maintain the continuity equation at the trajectory level, ensuring exact conservation of probability mass throughout the generation process.

Submitted to 39th Conference on Neural Information Processing Systems (NeurIPS 2025). Do not distribute.

We make several key contributions:

**Theoretical Framework:** We prove that adding divergence-free perturbations to the velocity field preserves the continuity equation, providing mathematical guarantees that our modifications do not violate the underlying probability flow. Our approach maintains the linear interpolation paths and ODE structure that define flow matching, unlike concurrent work that converts to SDE sampling.

**Practical Algorithms:** We develop two complementary strategies: (1) divergence-free path exploration that generates multiple trajectories from a single initial condition, and (2) a two-stage approach combining random search over initial conditions with divergence-free exploration around promising candidates. This mirrors the exploration-exploitation tradeoff in reinforcement learning.

**Empirical Validation:** We demonstrate the effectiveness of our approach across two distinct domains. On ImageNet 256×256 using SiT-XL/2, our methods achieve significant improvements in FID, Inception Score, and DINO accuracy as compute budget increases. On protein design using FoldFlow, we show substantial improvements in structural quality (TM-score) from 0.743 to 0.868, establishing the cross-domain applicability of our approach.

**Methodological Insights:** Our experiments reveal that different inference-time scaling strategies excel under different conditions: random search provides broad exploration particularly effective for discrete sequence-structure relationships, while divergence-free perturbations enable focused local search around high-quality regions. The combination of both approaches consistently outperforms either method in isolation.

Our work establishes inference-time scaling as a viable and principled approach for enhancing flow matching models, opening new directions for improving continuous generative models across scientific and creative applications without the computational cost of retraining.

## 2 Preliminaries

**Notation.** Random variables are uppercase (e.g., $X$), and realizations are lowercase (e.g., $x$). Distributions at time $t \in [0,1]$ are denoted $\pi_t$ with density $p_t$; endpoints are $\pi_0 = \pi_{\mathrm{ref}}$ and $\pi_1 = \pi_{\mathrm{data}}$. Vector fields $v :^d \times[0,1] \to^d$ are time-dependent and (unless stated otherwise) Lipschitz in $x$ and measurable in $t$. The continuity equation is

$$\partial_t p_t(x) + \nabla \cdot \big(p_t(x)\, v(x,t)\big) = 0, \qquad p_0, p_1 \text{ given.} \tag{1}$$

### 2.1 Flow Matching

Flow Matching (FM) **?** defines a continuous bridge between a reference distribution $\pi_{\mathrm{ref}}$, typically a standard Gaussian, and a data distribution $\pi_{\mathrm{data}}$, by modeling trajectories $x_t$ that interpolate between samples $x_0 \sim \pi_{\mathrm{ref}}$ and $x_1 \sim \pi_{\mathrm{data}}$. A common choice is the linear path $x_t = (1-t)x_0 + tx_1$, though other conditional paths are possible and lead to generalized or conditional flow matching **???**. This is done via a learned velocity field $v_\theta(x,t)$ that satisfies the probability-flow ODE:

$$\frac{dx_t}{dt} = v_\theta(x_t, t), \quad t \in [0,1].$$

To train $v_\theta$, a supervised loss is used where the ground truth velocity is known analytically:

$$v^\star(x_t, t) = x_1 - x_0.$$

This target arises because $\frac{dx_t}{dt} = x_1 - x_0$ under linear interpolation. The training loss is then

$$\mathcal{L}_{\mathrm{FM}} = \mathbb{E}_{x_0 \sim \pi_{\mathrm{ref}}, x_1 \sim \pi_{\mathrm{data}}, t \sim \mathcal{U}[0,1]} \left[ \left\| v_\theta(x_t, t) - (x_1 - x_0) \right\|^2 \right].$$

Unlike diffusion models that rely on stochastic sampling from SDEs or discrete Markov chains, FM offers a deterministic, fast, and interpretable sampling process. Because the interpolant is linear and the learned dynamics are smooth, FM enables fewer sampling steps while maintaining sample quality.

### 2.2 Minibatch Optimal Transport Flow Matching (OT-FM)

While FM defines its objective using i.i.d. sample pairs $(x_0, x_1)$, such pairs are typically poorly coupled in high-dimensional space. Minibatch OT-FM **?** addresses this by using an optimal transport

(OT) plan computed within each minibatch to generate more meaningful pairs. That is, given batches $\{x_{0,i}\}_{i=1}^{B}$ from $\pi_{\text{ref}}$ and $\{x_{1,j}\}_{j=1}^{B}$ from $\pi_{\text{data}}$, an optimal permutation matrix $\pi^\star \in \Pi_B$ is computed to minimize a transport cost:

$$\pi^\star = \arg\min_{\pi \in \Pi_B} \sum_{i,j} \pi_{ij} \cdot c(x_{0,i}, x_{1,j}), \quad c(\cdot,\cdot) = ||x_{0,i} - x_{1,j}||^2$$

Computing exact OT scales cubically in batch size and quadratically in memory, so minibatch approximations are necessary for large datasets. Although such plans only approximate the global coupling, prior work has shown minibatch OT to be effective in practice for generative modeling and domain adaptation **?**. These better-aligned pairs shorten transport paths, stabilize training, and improve generalization.

### 2.3 Stochastic Interpolants: A Unifying Framework

The stochastic interpolants framework **?** shows that both FM and diffusion models can be described using the same general family of interpolations. A stochastic interpolant is defined by

$$x_t = a(t)x_0 + b(t)x_1 + \sigma(t)\epsilon, \quad \epsilon \sim \mathcal{N}(0, I),$$

where $a(t), b(t)$, and $\sigma(t)$ are schedule functions satisfying boundary conditions: $a(0) = 1$, $b(1) = 1$, $a(1) = b(0) = 0$, and $\sigma(0) = \sigma(1) = 0$. This framework can express linear FM, denoising diffusion models, and their hybrids.

Crucially, this allows inference-time reinterpretation of trained models by simply modifying the interpolant schedule. For example, a model trained using FM with linear schedules $(a(t), b(t)) = (1-t, t)$ and $\sigma(t) = 0$ can be reinterpreted at test time as a VP diffusion model by using $(a(t), b(t)) = (\cos(\alpha t), \sin(\alpha t))$ and $\sigma(t) \neq 0$. This reveals a continuum of models and motivates inference-time strategies that leverage the same learned model parameters.

### 2.4 Inference-Time Compute Scaling in Generative AI

Inference-time scaling refers to performance improvements achieved by increasing computational resources *after training*, without modifying model parameters. While generative models have traditionally scaled performance by increasing data, model size, or training compute, recent research has explored whether additional computation at inference can yield better outputs.

In discrete domains such as language modeling, inference scaling has been explored through methods that allocate more compute per query. These include using longer reasoning chains, recursive planning, or verifier-guided editing **????**. Notably, OpenAI's O1 and O3 models **?** and DeepSeek R1 **?** do not use branching across multiple sampled responses. Instead, they increase inference-time compute by allowing more function evaluations per output, for example via deeper chains-of-thought or tree-based planning structures.

In contrast, recent work in diffusion models has implemented inference-time scaling via **search over generation trajectories** **?**. Diffusion models naturally provide stochasticity through their noise-injection process. The inference-time scaling diffusion framework identifies that *some initial noises are better than others*, and proposes to search over this noise space using verifiers to guide sample selection. This turns the generation problem into a two-axis search: one axis governs the verifier that provides feedback (e.g., Inception Score, CLIP, DINO), and the other defines the search algorithm (e.g., random search, zero-order optimization, path-space exploration). This approach allows performance to continue improving beyond what is achievable by simply increasing denoising steps.

These developments motivate the need for inference-time scaling in **continuous-time models** such as Flow Matching (FM), which lack natural sources of stochasticity or branching. The key challenge is that FM models sample deterministically via a single integration of the learned velocity field along a fixed interpolant. Our method addresses this by introducing divergence-free perturbations to the velocity field, thereby defining a family of ODEs that maintain the continuity equation but diverge in geometry. This enables principled sampling diversity and verifier-guided branching in FM, filling a crucial gap in inference-time scaling for continuous-time generative models.

# 3 Related Work

## 3.1 Inference-Time Scaling for Flow Matching Models

A concurrent line of work proposes inference-time scaling for flow models by introducing stochasticity and path diversity into the otherwise deterministic sampling process **?**. This is achieved through a two-step transformation of the learned model: (1) converting the velocity field from the probability flow ODE to a score-based SDE sampler, and (2) replacing the standard linear interpolant with a variance-preserving (VP) interpolant path to enhance exploration.

Specifically, the learned velocity $u_t(x)$ is used to define the drift term of a reverse-time SDE:

$$dx_t = f_t(x_t)\,dt + g_t\,dW_t, \quad \text{where } f_t(x_t) = u_t(x_t) - \frac{g_t^2}{2}\nabla \log p_t(x_t).$$

The score function $\nabla \log p_t(x_t)$ is estimated analytically from $u_t$ using the stochastic interpolants framework **?**:

$$\nabla \log p_t(x_t) = \frac{1}{\sigma_t} \cdot \frac{\alpha_t u_t(x_t) - \dot{\alpha}_t x_t}{\dot{\alpha}_t \sigma_t - \alpha_t \dot{\sigma}_t}.$$

In parallel, the interpolation path is converted from a linear interpolant $x_t = (1-t)x_0 + tx_1$ to a VP interpolant, such as $x_t = \alpha_t x_0 + \sigma_t x_1$ where $\alpha_t^2 + \sigma_t^2 = 1$. This conversion requires transforming the original velocity field into one compatible with the new interpolant **?**:

$$\bar{u}_s(\bar{x}_s) = \frac{\dot{c}_s}{c_s}\bar{x}_s + c_s \dot{t}_s u_{t_s}(\bar{x}_s/c_s),$$

where $c_s = \bar{\sigma}_s/\sigma_{t_s}$ and $t_s = \rho^{-1}(\bar{\rho}(s))$ is defined via signal-to-noise ratio schedules $\rho(t) = \alpha_t/\sigma_t$, $\bar{\rho}(s) = \bar{\alpha}_s/\bar{\sigma}_s$.

This approach enables the use of particle sampling strategies originally developed for diffusion models, which benefit from diverse sample paths and stochastic exploration. However, by transforming both the dynamics and the interpolant, the method loses the key benefits of flow matching: fast sampling via few deterministic steps and linear interpolants.

While this work is concurrent to our own, we still differentiate ourselves as our method preserves the original FM structure. We maintain the ODE form of the sampler and the linear interpolant, and introduce diversity solely by injecting divergence-free velocity perturbations during sampling. This guarantees that probability mass is conserved through the continuity equation, while allowing geometrically distinct trajectories for verifier-guided search.

## 3.2 Noise Injection while Preserving the Continuity Equation

A distinct thread of work aims to inject stochasticity during inference without violating the underlying continuity equation that defines the model's evolution. This objective is shared by our divergence-free perturbation strategy, but is also addressed through Langevin-style diffusion sampling schemes.

The most widely adopted approach in this category is the so-called "churn" strategy **?**, used extensively in diffusion models. The idea is to introduce noise in a way that preserves the marginal densities $p_t(x)$ *in expectation*, rather than at the level of individual sample trajectories. Specifically, the sampler follows an SDE of the form:

$$dx_t = u_t(x_t) - \beta(t)\,\sigma^2(t)\,\nabla \log p_t(x_t)\,dt + \sqrt{2\beta(t)}\,\sigma(t)\,dW_t,$$

where $\beta(t)$ is a user-defined schedule that governs the amount of stochasticity injected at each timestep. The drift term and noise are precisely scaled so that they cancel out in the associated Fokker–Planck equation, thereby preserving $p_t$.

Unlike our method, which satisfies the continuity equation path-wise by ensuring $\nabla_x \cdot (p_t w_t) = 0$ at every point, the EDM method preserves the distribution only in expectation over trajectories. As a result, individual sample paths do not strictly conserve mass. This difference becomes critical when injecting higher magnitudes of noise: EDM samplers tend to lose image quality earlier than our divergence-free ODE approach, as shown in our empirical study in Section **??**.

We include this method as a strong baseline in our experiments for its conceptual proximity to our work.

### 3.3 Inference-Time Scaling for Diffusion Models

Inference-time scaling in diffusion models has recently been advanced through optimization in the latent noise space **?**. The core idea is to treat the initial noise vector $z \sim \mathcal{N}(0, I)$ as a controllable input, and to iteratively refine it using a verifier score $r(\cdot)$ that evaluates the final generated sample. The method begins with a population of candidate noise vectors $\{z_i\}$, denoises each to obtain $x_0^{(i)}$, scores them, and retains high-scoring candidates. New proposals are then generated by applying small perturbations to the best $z_i$, repeating the procedure over multiple rounds.

In addition to this search in noise space, the method proposes a "search over paths" strategy, in which denoised samples are partially re-noised and then denoised again. That is, after reaching an intermediate timestep $t$, the top samples are re-noised forward to $t' > t$, and denoising resumes from $t'$ to $t = 0$. This is intended to explore local perturbations in trajectory space around high-scoring samples.

However, the actual implementation uses hyperparameters such that the search begins at $t = 0.11$, and each re-noising step applies a forward process to $t' = 0.89$. This implies that 89% of the diffusion process is re-applied after a brief denoising. Because most of the signal is lost at this noise level, the re-noised samples are nearly indistinguishable from fresh random samples from the prior. Consequently, while this method technically performs a limited search over paths, in practice it behaves similarly to a best-of-$N$ strategy (often termed random search) conducted over initial noise vectors.

These strategies are highly effective in the diffusion setting, but they rely fundamentally on starting generation from a known distribution, typically $\mathcal{N}(0, I)$. In contrast, our approach applies to flow matching models trained with arbitrary or learned base distributions $\pi_{\text{ref}}$, which may not admit tractable sampling via random search.

### 3.4 Other Approaches to Efficient or Accurate Continuous Generative Models

Several approaches aim to improve the efficiency or quality of continuous-time generative models. Second-order ODE solvers **?** accelerate sampling by reducing discretization error. Curvature-controlled interpolants **?** introduce smoothness constraints on the sampling path. These methods are complementary to inference-time compute scaling and can be combined with divergence-free branching strategies for further performance improvements.

## 4 Inference Time Scaling for Flow Matching while Preserving the ODE

Our approach to inference-time scaling for flow matching models operates along two independent axes: **noise injection methods** (Sections 4.1-4.3) that introduce stochasticity during sampling, and **inference algorithms** (Section 4.4) that leverage this stochasticity for verifier-guided search. This separation allows for flexible combinations of noise types with different search strategies, enabling principled exploration of the compute-quality tradeoff space.

### 4.1 Divergence-Free Noise

To enhance sample diversity without altering the trained density path $p_t(x)$, we inject a small, divergence-free perturbation $w_t(x)$ into the learned velocity field $u_t(x)$. This preserves the ODE-based nature of the sampler and avoids departing from the continuity equation satisfied during training. We control the influence of the perturbation using a scalar hyperparameter $\lambda$, which scales the amount of injected noise.

### 4.2 Proof: Adding Divergence-Free Noise at Inference Preserves the Continuity Equation and the Probability Flow ODE

In flow-matching models, the learned velocity $u_t(x)$ satisfies the continuity equation:

$$\partial_t p_t(x) \; + \; \nabla_x \cdot \big(p_t(x)\, u_t(x)\big) \; = \; 0, \tag{CE}$$

ensuring that the evolution of densities $\{p_t\}$ is consistent with an underlying deterministic ODE. We aim to add a perturbation $w_t(x)$ (e.g. divergence-free "swirl") to improve diversity at inference time,

and want to confirm that the modified flow still respects the continuity equation. This is essential for ensuring that samples remain consistent with the trained marginal densities $p_t(x)$.

**Proposition.** Let $p_t$ and $u_t$ satisfy (CE). If a vector field $w_t$ satisfies

$$\nabla_x \cdot \big(p_t\, w_t\big) = 0 \quad \text{for all } t \in [0,1], \tag{1}$$

then the modified drift $\tilde{u}_t := u_t + \lambda w_t$, where $\lambda \in \mathbb{R}$ is a scalar hyperparameter controlling the amount of added noise, yields the same continuity equation:

$$\partial_t p_t \;+\; \nabla_x \cdot \big(p_t\, \tilde{u}_t\big) = 0.$$

*Proof.* Plug $\tilde{u}_t = u_t + \lambda w_t$ into (CE):

$$\partial_t p_t + \nabla_x \cdot \big(p_t(u_t + \lambda w_t)\big) = \underbrace{\big[\partial_t p_t + \nabla_x \cdot (p_t u_t)\big]}_{=0 \text{ by (CE)}} + \lambda\, \nabla_x \cdot \big(p_t w_t\big).$$

The second term vanishes by assumption (1), hence the entire expression equals zero and the modified drift preserves the same continuity equation. ■

**Remark (local criterion).** Using the identity

$$\nabla_x \cdot (p_t w_t) = p_t\big(\nabla_x \cdot w_t + s_t \cdot w_t\big), \quad s_t := \nabla_x \log p_t,$$

we see that condition (1) is equivalent to the pointwise or expected constraint:

$$\boxed{\nabla_x \cdot w_t + s_t \cdot w_t = 0.}$$

In practice, this is satisfied (in expectation) by choosing

$$w_t(x) = \big(I - \hat{s}_t \hat{s}_t^\top\big)\, \varepsilon, \quad \hat{s}_t = \frac{s_t}{\|s_t\|}, \quad \varepsilon \sim \mathcal{N}(0, I),$$

which projects Gaussian noise onto the subspace orthogonal to the score direction. This guarantees that $s_t \cdot w_t = 0$ exactly, while $\nabla_x \cdot w_t = 0$ holds in expectation because $w_t$ is a linear transformation of $\varepsilon$ with coefficients that are independent of the spatial variable $x$. Thus, $\nabla_x w_t = 0$ and the continuity equation is preserved.

### 4.3 Baseline Noise Injection Methods

While our primary contribution focuses on divergence-free noise injection, we also evaluate alternative approaches for introducing stochasticity into flow matching sampling as baseline comparisons:

**Standard SDE Sampling:** The most straightforward approach adds Gaussian noise directly to the velocity field: $dx_t = u_t(x_t)\, dt + \sigma\, dW_t$, where $\sigma$ controls the noise magnitude. While simple to implement, this approach does not preserve the continuity equation and can lead to rapid degradation of sample quality as noise levels increase.

**Score SDE Sampling:** Following the stochastic interpolants framework **?**, we can convert the learned velocity field to a score-based SDE formulation with drift $f_t(x_t) = u_t(x_t) - \frac{g_t^2}{2} \nabla \log p_t(x_t)$ that incorporates the analytically computed score function. This method preserves the marginal distributions in expectation but may lose trajectory-level mass conservation.

These baseline methods enable us to demonstrate the advantages of our divergence-free approach in maintaining sample quality while introducing beneficial stochasticity.

### 4.4 Inference Algorithms

Given any of the above noise injection methods, we can employ different search strategies to leverage the introduced stochasticity for improved sample quality. We present five inference algorithms that operate independently of the chosen noise injection method, enabling flexible combinations for different computational budgets and quality requirements.

#### 4.4.1 Best-of-N Sampling

The simplest inference scaling approach generates $N$ independent samples from different initial conditions and selects the top-$K$ samples based on a verifier function $r(\cdot)$. This method provides broad exploration of the initial condition space and scales linearly with computational budget. See Algorithm **??** for implementation details.

#### 4.4.2 Path Exploration

Path exploration generates multiple trajectories from the same initial condition by introducing stochastic branching during the integration process. At predetermined timesteps, we create multiple branches by applying different noise realizations, simulate each branch to completion, score the final samples, and continue with the highest-scoring branches. This enables focused exploration around promising trajectories.

**Computational Consideration:** Path exploration requires simulating intermediate samples forward to completion ($t = 1$) for scoring, which adds significant computational overhead beyond the reported compute budgets. This look-ahead scoring ensures accurate quality assessment but substantially increases the actual computational cost compared to other methods. See Algorithm **??** for implementation details.

#### 4.4.3 Best-of-N + Path Exploration

This two-stage approach combines the benefits of initial condition search with trajectory-level exploration. Stage 1 performs Best-of-N sampling to identify promising initial conditions. Stage 2 applies path exploration starting from the top-$K$ initial conditions identified in Stage 1. This mirrors the exploration-exploitation tradeoff in reinforcement learning. See Algorithm **??** for implementation details.

#### 4.4.4 Noise Search

Noise search generates multiple samples from the same initial condition by applying different noise realizations throughout the entire sampling process. Unlike path exploration, this method does not require intermediate scoring and branching, making it computationally efficient while still enabling local exploration around a given starting point. See Algorithm **??** for implementation details.

#### 4.4.5 Best-of-N + Noise Search

Similar to the path exploration variant, this two-stage method first identifies promising initial conditions via Best-of-N sampling, then applies noise search around each selected starting point. This approach provides both broad initial exploration and focused local search while maintaining computational efficiency. See Algorithm **??** for implementation details.

#### 4.4.6 Multi-Round Noise Search

The noise search algorithms can be extended to multiple rounds of progressively focused exploration. In Round 1, we apply noise search starting from $t = 0$ and select the top-$K$ samples. In Round 2, we continue noise search from these $K$ samples at an intermediate timestep (e.g., $t = 0.5$), enabling further exploration around high-scoring regions. Round 3 continues from $t = 0.75$ with the best samples from Round 2. This progressive refinement allows for increasingly targeted search while maintaining computational tractability.

#### 4.4.7 Connection to Particle Sampling

Our inference algorithms share conceptual similarities with particle sampling methods, where multiple "particles" (sample trajectories) evolve through the state space and are periodically reweighted or resampled based on fitness criteria. The path exploration method particularly resembles particle filtering approaches, where particles branch and are selected based on likelihood estimates. However, our methods operate specifically within the flow matching framework and leverage verifier functions rather than likelihood-based reweighting.

### 4.5 Empirical Demonstration of Diversity Without Reducing Quality

To validate this approach, we evaluate how the injection of noise at inference time affects generation quality across several methods. We compare our divergence-free ODE method to a deterministic baseline, a simple Euler-Maruyama SDE, and a Langevin-style stochastic sampler derived from EDM **?**. All experiments are conducted using the pretrained SiT-XL/2 flow-matching model on ImageNet 256×256. For each configuration, we generate 1,000 samples and compute Fréchet Inception Distance (FID), Inception Score (IS), and sample diversity.

**Setup.** The $x$-axis in all figures represents the average magnitude of injected noise per step, normalized to the average magnitude of the velocity field $u_t(x)$. The $y$-axis reports sample diversity, FID, or IS. We compare four sampling strategies:

- **ODE-divfree**: Our proposed method, which injects divergence-free perturbations $w_t(x)$ that preserve the continuity equation at the level of individual trajectories:

$$dx_t = (u_t(x) + \lambda\, w_t(x))\, dt,$$

  where $\lambda$ controls the noise scale.

- **SDE**: Samples are generated from a simple Euler-Maruyama SDE:

$$dx_t = u_t(x)\, dt + \sigma\, dW_t,$$

  where $dW_t \sim \mathcal{N}(0, dt)$, and $\sigma$ scales the noise.

- **EDM-SDE**: A stochastic method that adjusts both drift and diffusion to preserve $p_t(x)$ in expectation:

$$dx_t = u_t(x)\, dt - \beta(t)\, \sigma^2(t)\, \nabla \log p_t(x)\, dt + \sqrt{2\beta(t)}\, \sigma(t)\, dW_t,$$

  where $\beta(t)$ is a user-defined schedule. See Related Work for details (Section **??**).

- **Score-SDE**: Uses the Score SDE formulation with analytically computed score functions from the stochastic interpolants framework.

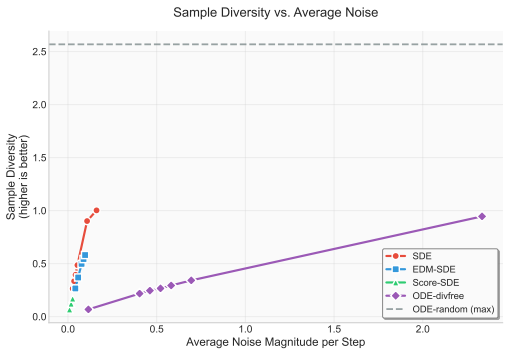


Figure 1: Sample diversity across increasing noise levels. Higher is better.

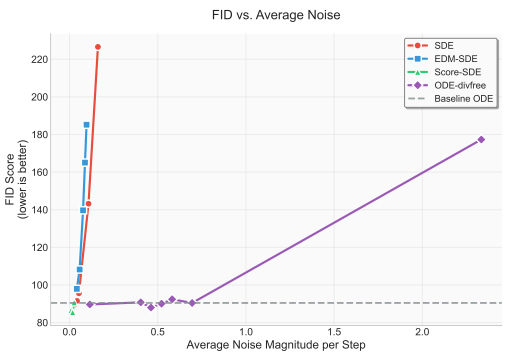


Figure 2: FID across increasing noise levels. Lower is better.

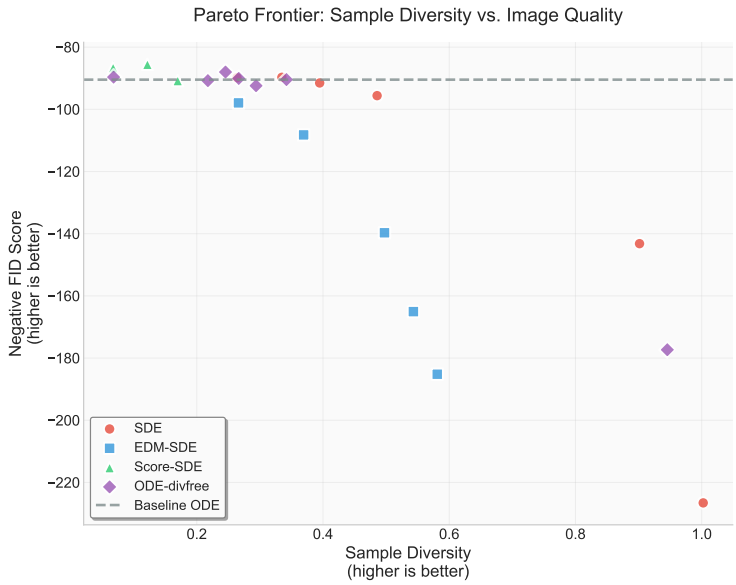


Figure 3: Pareto frontier analysis showing the trade-off between sample diversity and image quality (negative FID, where higher is better for both axes). Our divergence-free method achieves favorable positions on the frontier, maintaining high image quality while enabling substantial diversity gains compared to other stochastic sampling approaches.

**Results.** As shown in the figures, our divergence-free ODE approach allows substantially higher levels of noise to be injected without degrading sample quality. Sample diversity increases smoothly with noise level (Fig. **??**), while FID remains stable across a wide noise range (Fig. **??**). The Pareto frontier analysis (Fig. **??**) demonstrates that our method achieves superior trade-offs between diversity and quality compared to alternative stochastic approaches. In contrast, standard SDE sampling degrades rapidly as noise increases. EDM sampling performs better than the naïve SDE—consistent with its theoretical guarantee of preserving $p_t(x)$ in expectation—but still deteriorates earlier than our method. Similar analysis for Inception Score shows comparable results (see Appendix). This suggests that satisfying the continuity equation per trajectory, as our approach does, offers stronger robustness to stochasticity during sampling.

# 5 Experiments

We evaluate the effectiveness of our inference-time scaling framework across two domains: ImageNet 256×256 image generation and protein structure design. Our experiments systematically explore combinations of noise injection methods with inference algorithms to demonstrate the versatility and effectiveness of our approach.

## 5.1 Evaluated Method Combinations

We evaluate seven distinct combinations of noise injection methods and inference algorithms across both experimental domains. Each method is assigned a shorthand name for consistent reference throughout our results:

- **Best-of-N**: Random search over initial conditions using deterministic ODE sampling
- **Path-DivFree**: Path exploration with divergence-free noise injection (Algorithm **??**)
- **Path-SDE**: Path exploration with standard SDE noise injection
- **Path-ScoreSDE**: Path exploration with score SDE noise injection
- **BestN+Path-DivFree**: Two-stage Best-of-N followed by divergence-free path exploration (Algorithm **??**)

- **NoiseSearch-3R-DivFree**: Multi-round (3 rounds) noise search with divergence-free noise injection (Algorithm **??**)
- **BestN+NoiseSearch-3R-DivFree**: Two-stage Best-of-N followed by 3-round divergence-free noise search (Algorithm **??**)

All methods are evaluated across compute budgets of 1×, 2×, 4×, and 8×, where the compute factor corresponds to the number of final samples retained. The 1× baseline uses deterministic ODE sampling with 1,000 samples for ImageNet and 64 samples for FoldFlow.

## 5.2 ImageNet 256×256: Inference-Time Scaling with SiT-XL/2

We demonstrate the effectiveness of our inference-time scaling approach on ImageNet 256×256 using the pretrained SiT-XL/2 flow-matching model.

### 5.2.1 Experimental Setup

We evaluate all seven method combinations described in Section 5.1 on ImageNet 256×256 using the pretrained SiT-XL/2 flow-matching model. All methods generate 1,000 samples for robust statistical evaluation.

### 5.2.2 Evaluation Metrics and Verifiers

We evaluate performance using four metrics: FID (lower is better), Inception Score (higher is better), DINO Top-1 accuracy (higher is better), and DINO Top-5 accuracy (higher is better). Inception Score measures the quality and diversity of generated images based on a pretrained ImageNet classifier **?**. DINO scores evaluate semantic quality using self-supervised vision transformer features that capture richer visual representations **?**. FID computes the Fréchet distance between generated and real image feature distributions, providing a comprehensive measure of sample quality and diversity **?**. Each experiment uses two different verifier functions for sample selection: Inception Score-based scoring and DINO-based scoring.

### 5.2.3 Results: Inception Score-Guided Scaling

Figure **??** shows the results when using Inception Score as the verifier for sample selection. We report Inception Score and DINO Top-1 accuracy as the primary metrics, as these capture different aspects of sample quality.

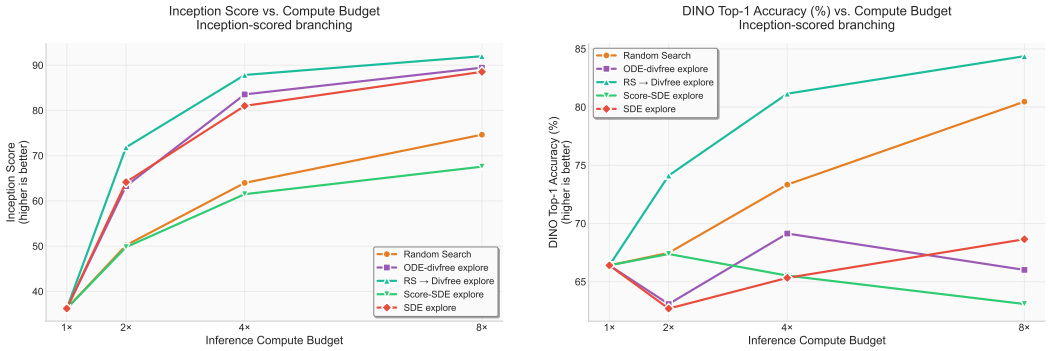


Figure 4: Inference-time scaling results using Inception Score-guided selection. Left: Inception Score vs. compute budget. Right: DINO Top-1 accuracy vs. compute budget.

The RS → Divfree explore method demonstrates the strongest performance, achieving the highest Inception Scores at 4× and 8× compute budgets. This two-stage approach effectively combines the benefits of searching over initial conditions with path-space exploration. The ODE-divfree explore method shows steady improvement but is limited by starting from a single initial condition. Random search provides consistent but modest gains, while the SDE-based methods show more variable performance.

### 5.2.4 Results: DINO-Guided Scaling

Figure **??** presents results when using DINO-based scoring for sample selection. We focus on FID, Inception Score, and DINO Top-1 accuracy as key performance indicators.

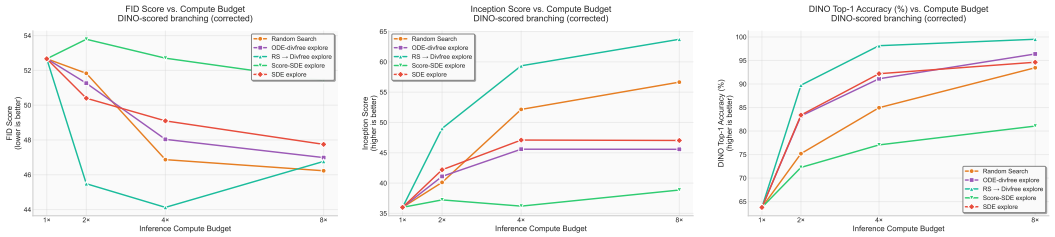


Figure 5: Inference-time scaling results using DINO-guided selection. Left: FID vs. compute budget. Center: Inception Score vs. compute budget. Right: DINO Top-1 accuracy vs. compute budget.

Under DINO-guided selection, the RS $\rightarrow$ Divfree explore method again shows superior performance, achieving the best FID scores and DINO Top-1 accuracy at higher compute budgets. Notably, DINO-guided selection produces more substantial improvements in DINO Top-1 accuracy compared to Inception-guided selection, demonstrating the importance of verifier-method alignment. The ODE-divfree explore method shows consistent improvement across all metrics, while random search and SDE methods provide more modest gains.

## 5.3 FoldFlow: Protein Design

We evaluate our inference-time scaling methods on protein structure generation using the pretrained FoldFlow model **?**. FoldFlow is a continuous normalizing flow model that generates protein backbone structures by learning the conditional distribution of protein coordinates given sequence information. This provides an important validation of our approach in a different scientific domain with distinct evaluation metrics.

### 5.3.1 Experimental Setup

We apply the same inference-time scaling framework from our ImageNet experiments to protein generation, evaluating all seven method combinations described in Section 5.1. We generate 64 protein samples of length 100 residues per configuration across compute budgets of 1×, 2×, 4×, and 8×. All methods are compared against the standard deterministic ODE sampling baseline (1× compute).

During inference-time scaling, we use self-consistency TM-score as the verifier function for sample selection, while our evaluation employs multiple metrics to comprehensively assess protein quality and designability.

### 5.3.2 Self-Consistency Evaluation

The self-consistency evaluation framework measures the designability of generated protein structures by testing whether they maintain structural integrity through a complete folding-refolding cycle. This process, following the FoldFlow methodology **?**, consists of four key steps:

**Structure Generation:** FoldFlow generates a protein backbone structure from noise, producing 3D coordinates for the protein chain.

**Inverse Folding:** The generated structure is fed to ProteinMPNN **?**, an inverse folding model that predicts amino acid sequences likely to fold into the given backbone structure. ProteinMPNN generates 8 different sequence candidates for each structure.

**Refolding:** Each of the 8 predicted sequences is then processed by ESMFold **?**, a structure prediction model that generates 3D structures from amino acid sequences.

**Structural Comparison:** The refolded structures are compared to the original generated structure using structural alignment metrics, specifically self-consistency RMSD (scRMSD) and self-consistency TM-score (scTM-score).

The self-consistency framework essentially asks: "If I generate a structure, predict sequences that should fold into it, and then fold those sequences, do I get back something similar to what I started with?" This tests whether the generated structure represents a realistic, foldable protein that follows the sequence-structure relationships observed in nature.

### 5.3.3 Evaluation Metrics

We evaluate protein generation quality using both continuous metrics and threshold-based designability measures:

**Continuous Metrics:** We report the average self-consistency RMSD (scRMSD) and self-consistency TM-score (scTM-score) across all generated proteins. For each protein, we take the best score among the 8 refolded structures. Lower scRMSD values indicate better structural preservation, while higher scTM-score values indicate better structural similarity.

**Designability Thresholds:** Following the FoldFlow evaluation framework, we measure the percentage of generated proteins that achieve specific self-consistency thresholds. We report the fraction of structures with scRMSD < 2.0Å, scRMSD < 1.5Å, and scRMSD < 1.0Å, representing increasingly stringent designability criteria. A protein is considered designable at a given threshold if at least one of its 8 refolded structures meets the criterion.

### 5.3.4 Results

Figure **??** shows the mean self-consistency TM-scores across different compute budgets for each inference-time scaling method. Additionally, we provide comprehensive evaluation through mean self-consistency RMSD scores and designability threshold analyses.

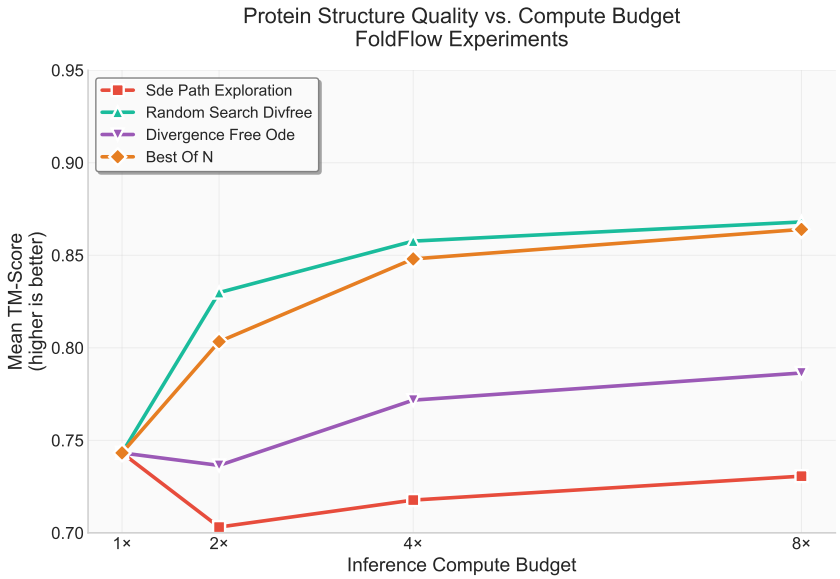


Figure 6: Protein structure quality (self-consistency TM-score) vs. compute budget for FoldFlow experiments. All methods start from the same baseline (standard ODE sampling at 1× compute) and demonstrate how additional inference compute improves protein designability.

The evaluation includes three complementary perspectives on protein generation quality. Mean self-consistency TM-score provides a continuous measure of structural similarity between generated and refolded structures. Mean self-consistency RMSD offers an alternative continuous metric focusing on coordinate-level accuracy. The designability threshold analysis reports the percentage of generated

proteins meeting increasingly stringent criteria: scRMSD < 2.0Å, scRMSD < 1.5Å, and scRMSD < 1.0Å. These threshold-based metrics align closely with the FoldFlow evaluation framework and provide insight into the practical utility of generated structures, as proteins meeting tighter thresholds are more likely to represent biologically viable designs.

# 6 Conclusion

We have introduced the first comprehensive framework for inference-time scaling in flow matching models, addressing a critical gap in continuous-time generative modeling. Our approach preserves the mathematical foundations of flow matching while enabling principled exploration of alternative generation trajectories through divergence-free velocity perturbations.

**Key Achievements:** Our theoretical analysis proves that divergence-free perturbations maintain the continuity equation at the trajectory level, providing stronger guarantees than methods that preserve distributions only in expectation. This mathematical rigor enables robust performance across diverse noise levels and ensures compatibility with the linear interpolation paths that define flow matching. Our experimental validation demonstrates substantial improvements across two distinct domains: ImageNet image generation and protein structure design, with our two-stage approach consistently achieving the best performance.

**Methodological Insights:** Our work reveals complementary strengths of different scaling strategies. Random search provides broad exploration particularly valuable for discrete relationships (e.g., protein sequence-structure mappings), while divergence-free path exploration enables focused exploitation around promising regions. The two-stage combination captures benefits of both approaches, suggesting that inference-time scaling benefits from balancing exploration and exploitation.

**Practical Impact:** The training-free nature of our approach makes it immediately applicable to existing flow matching models across domains. For protein design, improvements from 0.743 to 0.868 TM-score represent meaningful advances in structural quality with implications for drug discovery and protein engineering. For image generation, our method provides a scalable path to higher quality samples without architectural modifications or retraining.

**Limitations and Future Work:** While our approach shows diminishing returns at higher compute budgets (8×), suggesting practical limits to inference-time scaling, even modest compute increases (2×-4×) yield substantial quality improvements. Future work could explore adaptive scaling strategies that optimize compute allocation based on sample complexity, integration with more sophisticated verifiers for domain-specific applications, and extension to other continuous-time generative models beyond flow matching.

**Broader Implications:** Our work positions inference-time scaling as a fundamental capability for continuous generative models, complementing training-time scaling. As generative models become increasingly important for scientific discovery and creative applications, the ability to trade computational resources for sample quality at inference time provides practitioners with valuable flexibility. This paradigm shift from "one model, one quality level" to "adjustable quality through inference compute" opens new possibilities for interactive applications, quality-critical deployments, and resource-constrained scenarios.

Looking forward, we envision inference-time scaling becoming a standard tool in the continuous generative modeling toolkit, enabling researchers and practitioners to extract maximum value from trained models while adapting to diverse quality requirements and computational constraints.

# A Inference Algorithm Implementations

This section provides detailed algorithmic descriptions for all five inference methods introduced in Section 4.4.

### A.1 Best-of-N Sampling

**Algorithm 1** Best-of-N Sampling

**Require:** Flow matching model $v_\theta$, verifier function $r(\cdot)$, number of samples $N$, number to retain $K$
**Ensure:** Top-$K$ generated samples
**for** $i = 1$ to $N$ **do**
  $x_0^{(i)} \sim \pi_{\text{ref}}$ ▷ Sample initial condition
  $x_1^{(i)} \leftarrow \text{ODESolve}(v_\theta, x_0^{(i)})$ ▷ Deterministic sampling
  $s^{(i)} \leftarrow r(x_1^{(i)})$ ▷ Compute verifier score
**end for**
**return** Top-$K$ samples by score $\{s^{(i)}\}$

### A.2 Path Exploration

**Algorithm 2** Path Exploration with Noise Injection

**Require:** Flow matching model $v_\theta$, initial noise $x_0$, noise injection method $\mathcal{N}(\cdot)$, verifier function $r(\cdot)$, branching schedule $\{t_b\}$, branches per timestep $B$
**Ensure:** Generated sample $x_1$
trajectories $\leftarrow \{x_0\}$ ▷ Initialize with single trajectory
**for** each branching timestep $t_b$ **do**
  new_trajectories $\leftarrow \{\}$
  **for** each trajectory $x_{t_b}$ in trajectories **do**
    **for** $j = 1$ to $B$ **do**
      $x_1^{(j)} \leftarrow \text{NoisySample}(v_\theta, x_{t_b}, t_b, \mathcal{N})$ ▷ Sample to completion
      $s^{(j)} \leftarrow r(x_1^{(j)})$ ▷ Score final sample
      Add $(x_{t_b}, s^{(j)})$ to new_trajectories
    **end for**
  **end for**
  trajectories $\leftarrow$ Top scoring trajectories from new_trajectories
**end for**
**return** Best trajectory from final sampling round

### A.3 Best-of-N + Path Exploration

**Algorithm 3** Two-Stage Best-of-N + Path Exploration

**Require:** Flow matching model $v_\theta$, verifier function $r(\cdot)$, stage 1 samples $N$, top candidates $K$, noise injection method $\mathcal{N}(\cdot)$
**Ensure:** Top-$K$ generated samples
**Stage 1:** $\{x_0^*\} \leftarrow \text{BestOfN}(v_\theta, r, N, K)$ ▷ Algorithm **??**
**Stage 2:** results $\leftarrow \{\}$
**for** each $x_0^{(i)} \in \{x_0^*\}$ **do**
  $x_1^{(i)} \leftarrow \text{PathExploration}(v_\theta, x_0^{(i)}, \mathcal{N}, r)$ ▷ Algorithm **??**
  Add $x_1^{(i)}$ to results
**end for**
**return** Top-$K$ samples from results by verifier score

### A.4 Noise Search

**Algorithm 4** Multi-Round Noise Search

**Require:** Flow matching model $v_\theta$, initial noise $x_0$, noise injection method $\mathcal{N}(\cdot)$, verifier function $r(\cdot)$, rounds $R$, samples per round $N$, candidates per round $K$
**Ensure:** Generated sample $x_1$

candidates $\leftarrow \{x_0\}$ ▷ Initialize with single candidate
**for** round $i = 1$ to $R$ **do**
  $t_{\text{start}} \leftarrow (i-1)/R$ ▷ Starting timestep for round
  round_samples $\leftarrow \{\}$
  **for** each candidate $x_{t_{\text{start}}}$ in candidates **do**
    **for** $j = 1$ to $N$ **do**
      $x_1^{(j)} \leftarrow \text{NoisySample}(v_\theta, x_{t_{\text{start}}}, t_{\text{start}}, \mathcal{N})$
      $s^{(j)} \leftarrow r(x_1^{(j)})$
      Add $(x_{t_{\text{start}}}, x_1^{(j)}, s^{(j)})$ to round_samples
    **end for**
  **end for**
  **if** $i < R$ **then**
    candidates $\leftarrow$ Top-$K$ starting points by final score
  **else**
    **return** Best final sample by verifier score
  **end if**
**end for**

### A.5 Best-of-N + Noise Search

**Algorithm 5** Two-Stage Best-of-N + Noise Search

**Require:** Flow matching model $v_\theta$, verifier function $r(\cdot)$, stage 1 samples $N$, top candidates $K$, noise injection method $\mathcal{N}(\cdot)$, search rounds $R$
**Ensure:** Top-$K$ generated samples

**Stage 1:** $\{x_0^*\} \leftarrow \text{BestOfN}(v_\theta, r, N, K)$ ▷ Algorithm **??**
**Stage 2:** results $\leftarrow \{\}$
**for** each $x_0^{(i)} \in \{x_0^*\}$ **do**
  $x_1^{(i)} \leftarrow \text{NoiseSearch}(v_\theta, x_0^{(i)}, \mathcal{N}, r, R)$ ▷ Algorithm **??**
  Add $x_1^{(i)}$ to results
**end for**
**return** Top-$K$ samples from results by verifier score

# B Additional Experimental Results

This appendix contains supplementary figures and detailed results that support the main findings presented in the paper.

## B.1 Complete Noise and Diversity Study Results

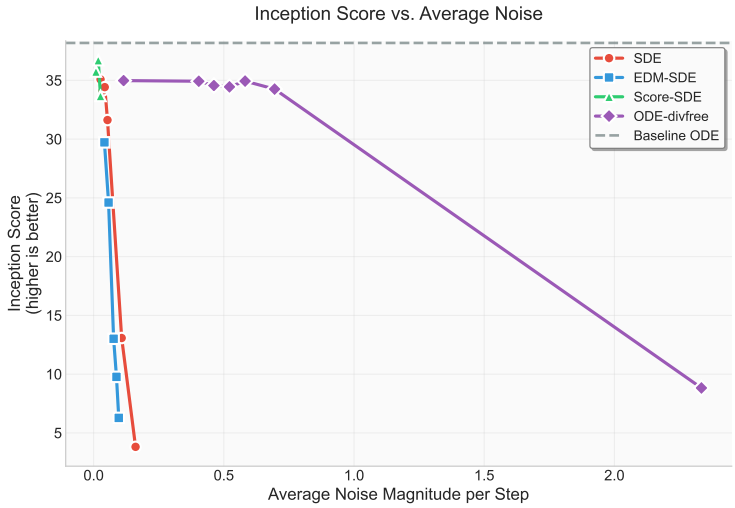


Figure 7: Inception Scores across increasing noise levels. Higher is better. This complements the diversity and FID analysis shown in the main text, demonstrating similar robustness patterns for our divergence-free method.

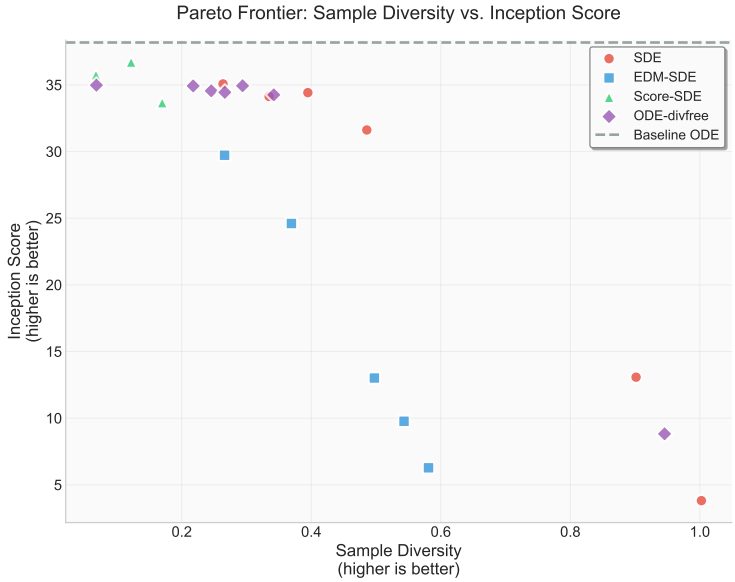


Figure 8: Pareto frontier analysis showing the trade-off between sample diversity and Inception Score (higher is better for both axes). This complements the FID-based Pareto analysis in the main text.

## B.2 Complete Inference-Time Scaling Results

### B.2.1 Inception Score-Guided Scaling (All Metrics)

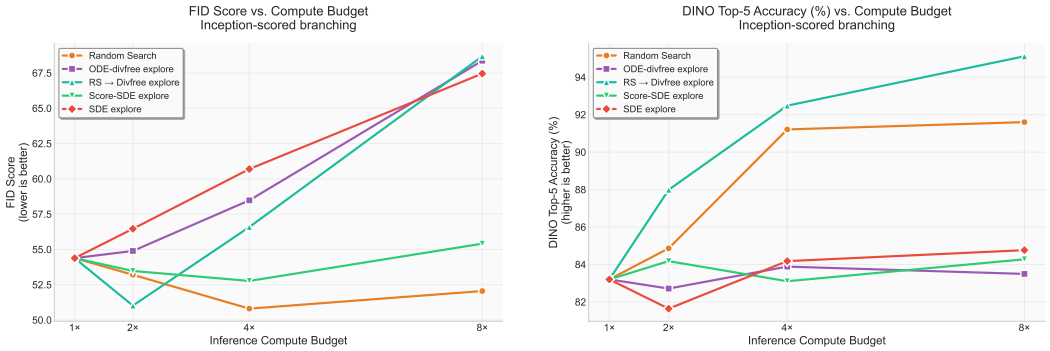


Figure 9: Additional Inception Score-guided scaling results. Left: FID vs. compute budget. Right: DINO Top-5 accuracy vs. compute budget.

### B.2.2 DINO-Guided Scaling (All Metrics)

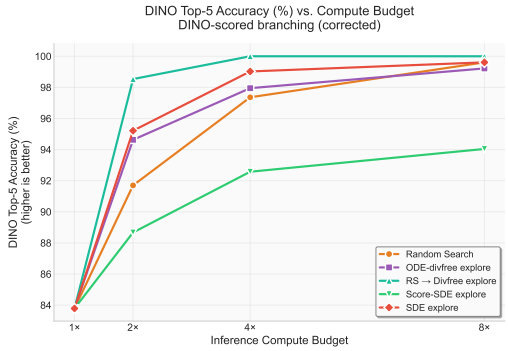


Figure 10: DINO Top-5 accuracy vs. compute budget for DINO-guided scaling experiments.

# C Implementation Details

## C.1 Look-Ahead Scoring for Branching Methods

Our path exploration methods (ODE-divfree explore, SDE explore, and the two-stage approach) employ a look-ahead scoring strategy for branch selection. At each branching point, multiple candidate trajectories are simulated to completion, and each generated sample is evaluated using the verifier function (Inception Score, DINO score, or TM-score). The scores are then used to select the most promising branches for continued exploration.

Alternative approaches exist for scoring intermediate branches, such as evaluating partially-noised samples at branching points or using a fixed computational budget for partial trajectory simulation. However, these methods introduce additional complexity in score interpretation and may not provide reliable quality estimates for incomplete generations. For simplicity and to ensure fair comparison across methods, we use complete trajectory simulation for all scoring decisions.

It is important to note that we do not include the computational cost of look-ahead scoring when reporting computational complexity or compute budgets. The reported compute factors (1×, 2×, 4×, 8×) reflect only the final number of samples retained, not the intermediate computational overhead required for branch evaluation. This choice enables cleaner comparison across methods while acknowledging that practical deployment would need to account for this additional computational cost.

### C.2 Experimental Hyperparameters

Table **??** summarizes the key hyperparameters used across all experiments. These values were selected based on preliminary experiments to balance sample quality and computational efficiency.

| Parameter | ImageNet | FoldFlow |
|---|---|---|
| Model | SiT-XL/2 | FoldFlow |
| Number of samples | 1,024 | 64 |
| Integration timesteps | 20 | 50 |
| Protein length | N/A | 50 residues |
| **Divergence-free methods** | | |
| Noise scale ($\lambda$) | 0.35 | 0.6 |
| Branch interval | N/A | 0.1 (every 5 steps) |
| **SDE methods** | | |
| Noise scale ($\sigma$) | 0.1 | 0.3 |
| Branch interval | N/A | 0.1 (every 5 steps) |
| **Compute budgets** | 1×, 2×, 4×, 8× | 1×, 2×, 4×, 8× |
| **Verifiers** | Inception, DINO | TM-score |

Table 1: Experimental hyperparameters for ImageNet and FoldFlow experiments. Branch interval controls the frequency of branching operations during path exploration.

### C.3 Domain-Specific Implementation Details

#### C.3.1 ImageNet Experiments

For ImageNet experiments, we use the pretrained SiT-XL/2 model with 20 integration timesteps, following the standard configuration for efficient sampling. The divergence-free noise scale of $\lambda = 0.35$ was selected to provide substantial diversity without degrading sample quality, as validated in our noise study (Section **??**). The SDE noise scale of $\sigma = 0.1$ provides a conservative baseline for stochastic sampling comparison.

All ImageNet experiments generate 1,024 samples per configuration to ensure robust statistical evaluation across the four metrics (FID, Inception Score, DINO Top-1, DINO Top-5). Verifier-based selection uses either Inception Score or DINO features, enabling comparison of verifier-method alignment effects.

#### C.3.2 FoldFlow Experiments

FoldFlow experiments focus on proteins of length 50 residues, providing a manageable complexity for systematic evaluation while representing realistic protein design scenarios. The model uses 50 integration timesteps by default, but performing branching operations at every timestep would be computationally prohibitive and unnecessary.

The branch interval parameter controls branching frequency: a value of 0.1 indicates branching every 10% of the trajectory. With 50 timesteps, this corresponds to branching every 5 timesteps, resulting in 10 total branching operations per trajectory. This provides sufficient exploration opportunities while maintaining computational tractability.

The higher noise scales for FoldFlow ($\lambda = 0.6$ for divergence-free, $\sigma = 0.3$ for SDE) reflect the different data characteristics and model sensitivities compared to image generation. These values were calibrated to provide meaningful exploration while preserving protein structural validity.

The smaller sample size (64 proteins) reflects the computational cost of protein generation and evaluation, while still providing sufficient data for reliable TM-score estimation and method comparison.

## D Additional Theoretical Analysis

TODO: Add extended theoretical analysis and proofs.