Y-SHAPED GENERATIVE FLOWS

Anonymous authors
Paper under double-blind review

ABSTRACT

Modern continuous-time generative models typically induce *V-shaped* flows: each sample travels independently along a nearly straight trajectory from the prior to the data. Although effective, this independent movement overlooks the hierarchical structures that exist in real-world data. To address this, we introduce *Y-shaped generative flows*, a framework in which samples travel together along shared pathways before branching off to target-specific endpoints. Our formulation is theoretically justified, yet remains practical, requiring only minimal modifications to standard velocity-driven models. We implement this through a scalable, neural network-based training objective. Experiments on synthetic, image, and biological datasets demonstrate that our method recovers hierarchy-aware structures, improves distributional metrics over strong flow-based baselines, and reaches targets in fewer steps.

1 INTRODUCTION

Recent advances in generative modeling are heavily based on continuous-time flows. In this framework, a model learns a velocity field to transport the mass from a simple starting distribution to a complex target distribution (Ho et al., 2020; Lipman et al., 2022). Current state-of-the-art methods, such as flow matching, treat every data point as an isolated traveler moving along its own independent path (Liu et al., 2022; Tong et al., 2023). For example, moving a starting point x_0 to two different targets, y_1 and y_2, requires two completely separate trajectories that do not interact. We refer to this uncoordinated pattern as a *V-shaped flow*, as the objective does not provide a mechanism for the paths to merge.

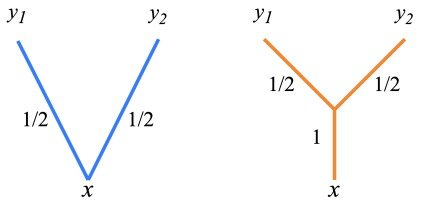


Figure 1: *Blueprint of flow shapes*. Conceptually in a V-flow, the mass separates and moves from source x along straight lines. In Y-flows, the mass moves together and then splits into targets.

However, we ask: *is a V-shaped flow always the most natural way to translate between real-world data distributions?* We argue that it often is not. Real-world data usually possess a hierarchical structure, groupings, and taxonomies that V-flows ignore. In applied mathematics, a closely related question was raised in branched optimal transport theory. This theory is inspired by natural and engineered systems designed for efficiency, such as vascular systems, trees, river basins, and urban supply networks (Bernot et al., 2005; Buttazzo & Stepanov, 2003; Xia, 2003). In these systems, it is more efficient to transport a large mass together along a main *trunk* before splitting it, rather than sending many small units individually. Translating this branching principle into generative modeling suggests that trajectories should be allowed to merge toward well-represented modes and then branch out to reach diverse targets. In contrast to conventional independent V-flows, we introduce *Y-shaped generative flows* (Figure 1).

This structural shift also addresses a second major challenge in current generative modeling research: efficiency. Many recent V-shaped approaches attempt to reduce the number of steps required to generate a sample using techniques such as mean flows (Geng et al., 2025), shortcut models (Frans et al., 2024), and *jumps* (Holderrieth et al., 2024). We argue that a Y-shaped framework is a more fundamental solution for faster generation. In a Y-flow, samples targeting central modes of a distribution share a common path and reach their destination quickly, while more unique or peripheral samples branch off later. This allows the model to naturally allocate steps based on complexity, rather than forcing a one-size-fits-all trajectory.

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

Contribution. We formally introduce *Y-shaped generative flows*, a continuous-time model that encourages shared transport before branching. Our method is based on a novel *concave velocity-power* action, and we prove (Theorem 1) that a low value of this objective forces delayed branching for arbitrary admissible flows. To implement this, we develop a scalable, neural network-based training objective. This objective balances the time-integrated branched action with a target reaching the boundary to ensure the flow matches the target data. Empirically, we demonstrate that our models learn branched trajectories on synthetic, image, and single-cell datasets, improving performance metrics (e.g., W_1/W_2/MMD) over strong baselines.

2 Background

Notation. Let $\Omega \subset \mathbb{R}^d$ be a compact and convex domain. We denote by $\mathcal{P}_2(\Omega)$ the space of probability measures on Ω with a finite second moment. When a measure μ admits a density relative to the Lebesgue measure, we denote its density by ρ (i.e., $d\mu(x) = \rho(x)dx$). Vectors are column vectors, $|\cdot|$ is the standard Euclidean norm, $\nabla\cdot$ is the divergence operator, and $|\cdot|_F$ is the Frobenius norm. The notation $d\mathcal{H}^1$ represents the differential element with respect to the *Hausdorff measure* in one dimension.

Continuous Normalizing Flows (CNFs) proposed by Chen et al. (2018) are generative models that define a probability density path through a neural ordinary differential equation (ODE) to morph one probability distribution, μ_0, into another, μ_1: $\frac{d}{dt}x_t = v_\theta(x_t, t)$, where $x_{t=0} \sim \mu_0$. Let Φ_t be the flow map associated with this ODE, which transports a particle from its initial condition at time 0 to its location at time t. The pushforward density $\rho_t = (\Phi_t)_\# \rho_0$ evolving under this dynamics *necessarily* satisfies *continuity equation* with the parameterized velocity field v_θ:

$$\partial_t \rho_t + \nabla \cdot (\rho_t v_t) = 0 \quad \text{on } \Omega \times (0,1), \qquad (1)$$
$$\rho_{t=0} = \rho_0, \quad \rho_{t=1} = \rho_1.$$

A key result is the instantaneous change of variables formula, which describes how the log density evolves along a trajectory: $\frac{d}{dt} \log \rho_t(x_t) = -\nabla \cdot v_\theta(x_t, t)$. This allows for a likelihood calculation by integrating this quantity over time. Training can be done by directly maximizing likelihood.

Monge–Kantorovich Optimal Transport (OT) seeks a way to morph one probability distribution with minimal effort, as quantified by a cost function $c(x, y)$. The Monge formulation seeks a deterministic map $T : \Omega \to \Omega$ that pushes μ_0 to μ_1 and minimizes the total cost $\int c(x, T(x))d\mu_0(x)$ (Villani, 2008). This problem can be ill-posed, its relaxation, the Kantorovich problem, searches over *couplings* (joint distributions) $\pi \in \Pi(\mu_0, \mu_1)$ with marginals μ_0 and μ_1: $\inf_{\pi \in \Pi(\mu_0,\mu_1)} \int_{\Omega\times\Omega} c(x, y)\, d\pi(x, y)$. For cost $c(x, y) = \|x - y\|^2$, the square root of the solution is the Wasserstein-2 distance.

Benamou–Brenier OT frames transportation as a continuous-time problem. For flow Φ_t, we have a pushforward density $\rho_t = (\Phi_t)_\# \rho_0$ evolving under the velocity field v. The Benamou–Brenier theorem states that the squared Wasserstein distance equals the minimal *kinetic energy* of such a flow:

$$W_2^2(\mu_0, \mu_1) \;=\; \inf_{\rho_t,\, v_t} \int_0^1 \int_\Omega |v_t(x)|^2 \, \rho_t(x)\, dx\, dt \quad \text{s.t. Eq. (1).} \qquad (2)$$

The minimizers of this problem are constant-speed geodesics in the Wasserstein space. When an optimal transport map T exists for the static problem (e.g., when μ_0 is absolutely continuous), the geodesic is given by *interpolation*: $\rho_t = ((1 - t)\mathrm{Id} + tT)_\# \rho_0$. The corresponding velocity field is constant along the trajectories, $v_t(x_t) = T(x_0) - x_0$.

3 On Shapes of Flows

What is the most natural way to control the shape of a generative flow? To approach this question, we draw inspiration from the mathematical domain that studies the organization of branching structures in nature and human-made systems, such as rivers, blood vessels, and traffic networks (Brasco et al., 2011; Xia, 2003). The core of all branched methods is the concave transport cost. Typically of the form $m^\alpha \ell$, where m is the mass and ℓ is the distance, and $\alpha \in (0, 1)$ is a branching coefficient. Since $(m_1 + m_2)^\alpha < m_1^\alpha + m_2^\alpha$ for $\alpha < 1$, this cost structure incentivizes the masses to travel together along shared paths. Transporting a combined mass along a single path is cheaper than transporting a mass m along a few separate parallel paths (Santambrogio, 2015).

In the branched transport formulation introduced by Xia (2003), the transport system is modeled as a flow network represented by F. Physically, F acts as the momentum or *mass flux*. It encodes both the

direction of movement and the amount of mass flowing at any given point. Transport is concentrated along a geometric network of 1-dimensional paths, denoted by Γ. In each segment of the path, let θ represent the amount (or thickness) of mass flowing. Mass conservation is strictly enforced by the continuity equation $\nabla \cdot F = \mu_0 - \mu_1$. Here, the divergence $\nabla \cdot F$ acts as a local accounting system, ensuring that the net outflow from any point matches the difference between the source distribution (μ_0) and the target distribution (μ_1). The goal is to minimize the transport cost, $E_\alpha(F)$, defined as:

$$E_\alpha(F) = \int_\Gamma \theta^\alpha \, d\mathcal{H}^1 \tag{3}$$

Unlike simple distance minimization, this integral calculates the mass cost (θ^α) multiplied by the distance traveled ($d\mathcal{H}^1$). Because θ^α is sub-additive, it is *cheaper* to group the mass into thick trunks rather than sending the particles along separate, thin paths. However, because this cost function is highly non-convex and defined over a complex space of singular fields, directly minimizing this functional is computationally intractable.

Another perspective on branched optimal transport is the Benamou–Brenier formulation studied by Brasco et al. (2011). The idea is to consider the transport problem (2), but with a *concave* cost function. With $\alpha \in (0, 1)$ being the branching parameter, we define the cost that measures the *work* of moving a mass m over a distance ℓ. Identifying the length ℓ with the velocity v and the mass m with the time-dependent probability density ρ_t, the dynamic branched transport is the following:

$$B^\alpha(\mu_0, \mu_1) \;=\; \inf_{\rho, v} \int_0^1 \sum_{i \in I_t} |v_{t,i}| \, (\rho_{t,i})^\alpha \, dt, \quad \text{s.t. Eq. (1).} \tag{4}$$

In this formulation, the cost (4) should be finite, the measure $\rho(t, \cdot)$ must be purely atomic for (almost every) time t. This means that the mass is concentrated at a countable set of points $\{x_{t,i}\}_{i \in I_t}$: $\rho(t, \cdot) = \sum_{i \in I_t} \rho_{t,i} \, \delta_{x_{t,i}}$, where $\rho_{t,i} = \rho(t, \{x_{t,i}\})$. The associated momentum is then $F = \rho v = \sum_{i \in I_t} v_{t,i} \, \rho_{t,i} \, \delta_{x_{t,i}}$, where $v_{t,i} = v(t, x_{t,i})$ is the velocity of the atom $x_{t,i}$.

Another formulation inspired by the Modica–Mortola (MM) framework was proposed by Oudet & Santambrogio (2011). It replaces the singular, non-smooth branched transport problem with a sequence of regularized, elliptic energy functionals M_λ^α defined over the more regular space of $\mathcal{H}^1$ vector fields. For a given $F(x)$, the approximating functional is as follows:

$$M_\lambda^\alpha(F) = \lambda^{\gamma_1} \int_\Omega |F(x)|^\alpha dx + \lambda^{\gamma_2} \int_\Omega |\nabla F(x)|^2 dx, \tag{5}$$

where $\lambda > 0$ is a small parameter. The exponents α, γ_1, and γ_2 are derived directly from the transport dimension d and the cost exponent α. But this method is practically unstable and difficult to scale for real-world generative modeling applications. Please see Appendix B.

***Takeaway**: Existing branched methods bottleneck generative modeling: the discrete formulation of Xia (2003) requires intractable optimization over 1-dimensional paths, the continuous formulation of Brasco et al. (2011) involves singular measures ill-suited for sampling and Monte-Carlo estimation, and the Modica-Mortola approximation is numerically unstable and restricted to a static field.*

Collectively, these failures motivate our proposal: a simpler, robust framework capable of generating branching structures without these optimization pitfalls.

4 METHOD

In this section, we propose an alternative approach that is simpler than the existing methods. In our method, we are not solving BOT exactly. We borrow the branching preference and propose a novel velocity-field formulation that is compatible with neural ODEs. We begin by examining the properties of the flux-based branched formulation. Then we show that similar functionals in the velocity space can provide branching structures, too. Let us first decompose the components of the MM approximation (5):

Concave Flux Term ($\int |F|^\alpha$). This term forces the transport density to concentrate on a lower-dimensional set, promoting sparsity rather than diffusing throughout the space. Minimizing a

concave power encourages F to be 0 (empty space) or concentrated on a lower-dimensional manifold, effectively forming the transport network.

Dirichlet Regularizer ($\int |\nabla F|^2$). This term penalizes sharp transitions, preventing paths from collapsing into infinitely thin Dirac masses. Forces the flux to spread over a strip of finite width A, creating a *corridor*. Without this regularization, the concave flux term would prefer infinitely thin, infinitely dense paths, which are ill-defined in the functional space.

The Combined Effect. The concave term compresses the mass into singular lines, while the Dirichlet term acts as an opposing *pressure* that expands them into smooth channels; the equilibrium is a stable branched network with branches of finite thickness.

Motivated by this framework (conceptually rather than algebraically; see Appendix D), we propose a velocity-driven objective that shifts the nonlinearity from the flux field to the velocity field.

4.1 Y-Shaped Velocity Field

We translate the branching requirements into an optimization problem over dynamic flow trajectories. Unlike static path-based formulations, our objective depends explicitly on the instantaneous velocity field v_t and density ρ_t. The loss function is defined as:

$$\mathcal{L}(\rho, v) = \inf_{\rho, v} \underbrace{\int_0^1 \int_\Omega \rho_t(x)|v_t(x)|^\alpha \, dxdt}_{\mathcal{T}(\rho,v)} + \lambda \underbrace{\int_0^1 \int_\Omega \rho_t(x)\|\nabla v_t(x)\|^2 \, dxdt}_{\mathcal{C}(\rho,v)}. \tag{6}$$

This formulation is fully compatible with standard Neural ODE solvers, which naturally satisfy the continuity equation by advecting particles through the learned field v_t. Although distinct from the mass-flux approach, this objective drives the emergence of Y-shaped structures through the interplay of two competing dynamic mechanisms: *Cohesion* and *time* efficiency. We analyze this interplay by addressing three fundamental questions regarding the flow behavior.

1. What mechanism enforces the initial collective motion?

The formation of a shared *trunk* is driven by the *cohesion* term $\mathcal{C}$, which enforces regularity by penalizing spatial changes in the velocity field (∇v).

Proposition 1 (Zero-Cost of Uniform Motion)**.** *If the velocity field $v(x,t)$ represents a spatially uniform translation (rigid body motion without rotation) on the support of ρ_t, the Cohesion Cost density is zero.*

The gradient ∇v measures the difference in velocity between neighboring points. When the entire probability mass moves with the exact same velocity vector, this gradient vanishes. Consequently, the objective function assigns zero penalty to this state, creating a strong preference for the mass to remain aggregated and translate as a single unit for as long as possible.

2. What acts as the barrier to premature separation?

To satisfy the boundary conditions, the mass must eventually separate to reach different targets (e.g., y_1 and y_2). However, separating a continuous density requires the velocity field to diverge in opposing directions.

Proposition 2 (The Cost of Separation)**.** *If a continuous velocity field separates the mass into distinct directions over a transition region of width ϵ, the Cohesion Cost scales as $\mathcal{O}(1/\epsilon)$.*

To split the flow, velocity vectors at the same time step must point in different directions. If this change in direction occurs over a very short distance ϵ, it creates a steep slope, resulting in a large gradient magnitude. Since the cost function effectively integrates the squared slope ($\|\nabla v\|^2$), narrowing the gap ϵ causes the energy cost to explode. This acts as a barrier that prevents the trajectories from splitting until they are spatially distant enough to do so smoothly.

3. How does the transport cost enable delayed branching?

If the system delays splitting to avoid the cohesion penalty, the particles must traverse the remaining distance in a reduced time interval. This requires traveling at higher speeds.

Proposition 3 (Time-Compression). *For $\alpha \in (0,1)$, the cost of traveling a fixed distance decreases as the duration of the travel decreases. Specifically, the functional favors high-velocity, short-duration transport over constant-velocity transport.*

The transport cost $|v|^\alpha$ increases sub-linearly with speed when $\alpha < 1$. This implies an economy of scale with respect to time: doubling the speed increases the instantaneous cost by a factor less than two, while halving the travel time. Therefore, the total integral over time is lower for fast, short bursts than for slow, continuous motion. We term this *time-compression*, as it allows the system to offset the cost of waiting in the trunk.

Summary. The system minimizes total energy by balancing two regimes: the zero-cost uniform motion favored by $\mathcal{C}$ (Prop. 1) and the high cost of splitting (Prop. 2). Branching occurs only when the energetic gain from time-compression (Prop. 3) sufficiently dominates the splitting penalty. This trade-off defines a critical branching time $\tau^* > 0$.

Lemma 1 (Optimal branching time in a low-dimensional illustration). *In the symmetric two-target setting of Appendix C, for a sufficient cohesion weight λ and $\alpha \in (0,1)$, the optimal branching time satisfies $\tau^* > 0$, implying that a Y-shaped trajectory is energetically superior to a V-shaped trajectory ($\tau = 0$).* See Appendix C for the proof.

Lemma 1 concerns a prescribed family in the plane; our main result removes this restriction. Group particles by the terminal region reached at $t = 1$, let $B(t)$ be the weighted variance of the groups' conditional means (how far the future branches have separated), and let $\mathcal{A}_\alpha(t)$, $\mathcal{C}(t)$ be the two terms of (6) accumulated up to time t.

Theorem 1 (Delayed separation; informal). *Let $\alpha \in (0,1)$, $\|v_t\| \le v_{\max}$, and let ρ_t satisfy a weighted Poincaré inequality with constant C_P. Then every admissible flow satisfies $\big(\sqrt{B(t)} - \sqrt{B(0)}\big)_+^2 \le t \min\{v_{\max}^{2-\alpha}\mathcal{A}_\alpha(t),\, C_P\,\mathcal{C}(t)\}$, and a flow with objective (6) at most E reaches a fraction q of its final separation no earlier than $\tau_q \ge (\sqrt{B_q} - \sqrt{B(0)})^2(v_{\max}^{\alpha-2} + \lambda/C_P)/E$.*

Separation thus costs both transport action and cohesion energy, so a low-energy flow cannot branch immediately. Precise statement and proof: Appendix E; relation to branched transport: Appendix D.

5 Practical Implementation

We now describe a simple and scalable training scheme based on neural ODEs and a Monte Carlo (MC) approximation of the velocity-power action, now augmented with the cohesion regularization term. Throughout this section, we fix the time horizon to $[0,1]$.

We parameterize the velocity field by a neural network $v_\theta : \Omega \times [0,1] \to \mathbb{R}^d$ and evolve samples via the ODE from Section 2: $\frac{dx_t}{dt} = v_\theta(x_t, t), \quad x_{t=0} \sim \mu_0$. Let Φ_t denote the flow map. The time-t distribution of particles is the push-forward $\rho_t = (\Phi_t)_\# \mu_0$; i.e., if $x_0 \sim \mu_0$ then $x_t = \Phi_t(x_0)$ is a draw from ρ_t. This simple fact justifies using equal-weight particles to approximate expectations under ρ_t; no importance weights are introduced or needed.

Objective. The total action we minimize combines the sub-linear transport cost with the cohesion regularizer, as motivated in Section 4. We minimize:

$$Y^\alpha(\theta) = \int_0^1 \mathbb{E}_{x\sim\rho_t}\left[|v_\theta(x,t)|_2^\alpha + \lambda |\nabla_x v_\theta(x,t)|^2\right] dt, \tag{7}$$

where $\alpha \in (0,1)$ is the branching exponent and $\lambda > 0$ is the cohesion weight that matches the formulation of the method. The term $\|\nabla_x v_\theta\|_F^2$ represents the Dirichlet energy of the velocity field, which is efficiently computed using automatic differentiation.

Time discretizations and MC estimator. Let $0 = t_0 < t_1 < \cdots < t_K = 1$ be a time grid with steps $\Delta t_k = t_k - t_{k-1}$ (in the uniform case, $\Delta t_k \equiv \Delta t = 1/K$). For a mini-batch $\{x_0^{(i)}\}_{i=1}^N \sim \mu_0$, we integrate our neural ODE along the grid to obtain states $x_{t_k}^{(i)} = \Phi_{t_k}(x_0^{(i)})$. The consistent estimator of Equation (7) is then:

$$\hat{Y}^\alpha(\theta) = \frac{1}{N}\sum_{i=1}^N \sum_{k=1}^K \Delta t_k \, |v_\theta(x_{t_{k-1}}^{(i)}, t_{k-1})|_2^\alpha + \lambda \frac{1}{N}\sum_{i=1}^N \sum_{k=1}^K \Delta t_k \, |\nabla_x v_\theta(x_{t_{k-1}}^{(i)}, t_{k-1})|^2. \tag{8}$$

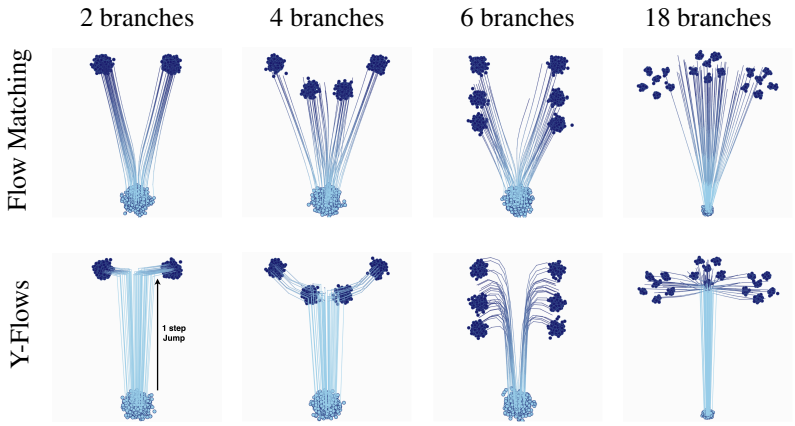


Figure 2: Comparison on low-dimensional Gaussian mixture illustrations. Each **column** is a task (2, 4, 6, and 18 branches). Each **row** is a method: top = Flow Matching (FM), bottom = Y-Flows (ours). The color gradient represents the flow steps. A monotone color indicates that the number of steps on this region was equal to 1. As shown in the 2, 4, and 18-branch cases, our model initially made a significant jump toward the target before splitting the mass. Subsequent movement involved reaching the targets via small almost zero size steps.

In the case of the common uniform grid, this simplifies the outer time summation to $\frac{1}{K}\sum_k$. If an adaptive ODE solver is used, we evaluate its internal time points and accumulate the corresponding Δt_k. Under exact integration, independence of $\{x_0^{(i)}\}$ and mesh refinement ($N, K \to \infty$), $\hat{Y}^\alpha(\theta) \to Y^\alpha(\theta)$ by the law of large numbers and Riemann sums.

Boundary Matching. To enforce $\rho_{t=1} = \mu_1$, we add a differentiable boundary loss that compares the empirical law of the final particles with μ_1. Given mini-batches $X_1 = \{x_{t=1}^{(i)}\}_{i=1}^{N_X}$ (push-forward samples) and $Y = \{y^{(j)}\}_{j=1}^{N_Y} \sim \mu_1$, we use the entropic Sinkhorn divergence (bias-corrected) (Cuturi, 2013; Feydy et al., 2019), with cost $c(x, y) = \|x-y\|_2^2$. Where OT_ϵ is the entropic OT cost computed by Sinkhorn iterations (regularization $\epsilon > 0$). **Final Loss.** Our total training objective is:

$$\mathcal{L}(\theta) = \hat{Y}^\alpha(\theta) + \lambda_{sink}\mathcal{L}_{sinkhorn}(\theta), \tag{9}$$

where $\lambda_{sink} > 0$ balances the boundary matching constraint against the trajectory action. We differentiate through the ODE solver (using standard backpropagation or the adjoint method) and optimize θ with stochastic gradient methods using fresh mini-batches from μ_0 and μ_1. In practice, for efficient Jacobian computation, the Hutchinson trick was applied (Grathwohl et al., 2018). The Sinkhorn term is one possible endpoint loss; Appendix F describes a more scalable alternative based on a learned Kantorovich dual potential (YF-KD), used for the pixel-space experiments in Appendix I.

6 Related Work

Branching transport problems were studied in discrete settings. Classical approaches include the Euclidean Steiner Tree (ESTP), where branch points are geometrically optimized under flow-independent costs, and early geometric constructions for single-source branched optimal transport (BOT) (Bernot et al., 2005; Buttazzo & Stepanov, 2003; Maddalena et al., 2003). More recently, Lippmann et al. (2022) introduced an approximate solver that decouples topology search from geometry optimization: given a tree topology, the positions of the branches are optimized through leaf-elimination in $O(nd)$ time, while a rewiring procedure explores the topology. Their method works only for *discrete* settings.

Beyond discrete graph solvers, continuous stochastic frameworks have also been proposed. Tang et al. (2025) extend Schrödinger Bridge Matching (BSBM) to the *branched* case by learning divergent drift and growth fields that transport mass from a common source to multiple targets. This method produces continuous branched trajectories, but is not based on branched transport theory.

7 EXPERIMENTS

7.1 BRANCHING ILLUSTRATIONS

Gaussian Mixtures. We compare how two flow-based methods transport mass from a single source distribution to a highly multi-modal target. The target is a mixture of K groups ("branches") arranged at the top; the source is a single group at the bottom. We train the same-size models with (i) standard FM and (ii) Y-Flows. Each curve shows the trajectory of a sample from source (light) to destination cluster (dark). Both models use the same time-conditioned MLP $v_\theta(x, t)$: 64-d time embedding $\rightarrow$ 3×256 SiLU layers $\rightarrow$ 2-D output; integrated with 10 fixed Euler steps over $t \in [0, 1]$. Training is identical: Adam ($lr = 10^{-3}$, batch 256, 10k iterations, seed 42). FM uses the standard flow-matching objective along linear source–target couplings. Y-Flows maintains the same backbone, but increases the loss with a branched-transport prior: endpoint OT/Sinkhorn term weight $\lambda_{sink} \approx 5$. For the results, see Figure 2. As shown, Y-Flows discovers a shared *trunk* that later splits into branches, producing short, structured, tree-like transport.

To take advantage of the fast convergence properties, we propose an early stopping mechanism for the neural ODE solver. Specifically, we terminate the integration if the displacement norm between consecutive steps, $\|x_{t+1} - x_t\|_2$, falls below a threshold of $\delta = 1 \times 10^{-3}$. This criterion effectively detects when the particles have settled in their target modes, allowing the model to dynamically adapt its computational budget. Empirically, we observe that on the Gaussian mixtures, Y-Flows converged in an average of 3 steps for the 2-branch case, 5 steps for 4 branches, 10 steps for 6 branches, and 4.5 steps for the 18-branch setting. In contrast, the Flow Matching baseline consistently utilized a fixed budget of 10 integration steps to achieve comparable transport, highlighting our method's ability to concentrate mass movement into fewer updates.

LiDAR Surface Navigation. The purpose of this experiment is to test whether a branched OT model *can learn when and where to split while staying confined to a real 3D surface*. For this experiment, we include a potential energy penalty on our loss as in (Liu et al., 2023; Tang et al., 2025). We transport a single source to four target clouds laid out on an airborne LiDAR terrain, which adds curvature, uneven sampling, and natural surface corridors that synthetic mixtures do not capture.

Each sample is a 3D point (x, y, z) with optional attributes (intensity, return number, total returns, scan angle, class). After filtering non-ground classes, removing outliers with a radius/k-NN check, and Poisson-disk (or voxel-grid) subsampling, we retain 5k points that preserve ridges, slopes, and basins. Coordinates are centered and scaled to a unit box (optionally PCA-aligned). We estimate per-point normals via local PCA and build a k-NN surface graph ($k \in [8, 16]$) with Euclidean edge weights as a geodesic proxy. During optimization, states are projected back to the nearest tangent patch to stay on-manifold. A single source distribution sits on a lower slope; four target clouds are disjoint regions on ridges/basins (indices provided for reproducibility).

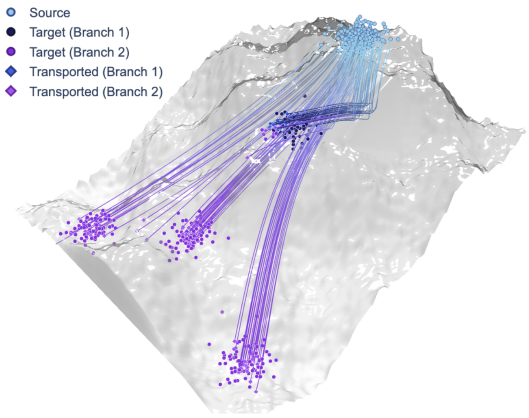
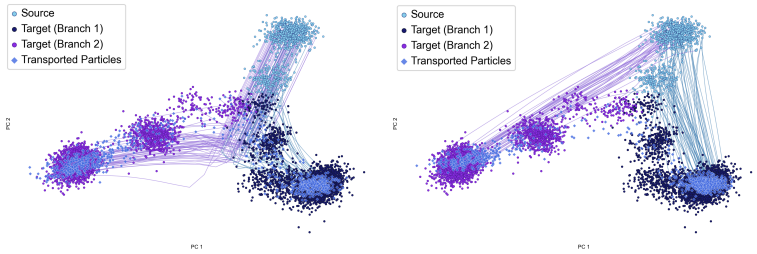


Figure 3: *Result of Y-Flows on LiDAR dataset.*

The method infers a three-junction topology, places the splits at terrain transitions (ridge breaks/valley entries), and keeps trajectories on-surface, reconstructing all four targets with tight endpoint fits. This shows that the approach handles surface-constrained, branched transport and is applicable in real-world tasks; see Figure 3 for details.

7.2 BIOLOGY DATA

Cellular Differentiation. To further validate our method, we use the Tedsim dataset (Pan et al., 2022) as a controlled reference point with known ground truth dynamics. Tedsim simulates a cellular differentiation process by modeling cell division from a root cell, generating both gene expression profiles and heritable lineage barcodes. This provides a complete record of a branching differentiation process.



(a) $\alpha = 0.5$. (b) $\alpha = 1.0$.

Figure 4: *PCA projected results of Y-Flows on Tedsim dataset (50D).*

Table 1: Comparison across methods on Tedsim dataset across embedding dimensions.

	Metric	BSBM	CNF	CFM	FM	Y-Flows
50D	W_1	17.72	13.76	12.33	12.09	**10.48**
	W_2	17.96	13.81	12.44	12.17	**10.72**
	MMD	0.63	0.51	0.11	0.15	**0.10**
150D	W_1	20.34	18.84	17.45	18.60	**14.29**
	W_2	20.51	18.88	17.54	19.79	**14.42**
	MMD	0.48	0.29	0.13	0.14	**0.13**
250D	W_1	23.47	22.48	20.07	22.80	**17.37**
	W_2	23.71	22.51	20.14	22.94	**17.50**
	MMD	0.55	0.18	0.12	0.16	**0.11**
500D	W_1	29.46	28.88	22.60	24.97	**20.99**
	W_2	29.63	28.91	22.68	25.04	**21.05**
	MMD	0.51	0.15	0.12	0.13	**0.12**

Table 2: Comparison across methods on Single-Cell RNA dataset across embedding dimensions.

	Metric	BSBM	CNF	CFM	FM	Y-Flows
50D	W_1	17.65	8.69	8.22	8.16	**7.06**
	W_2	18.11	9.14	8.52	8.41	**7.28**
	MMD	0.47	0.17	0.13	0.14	**0.12**
150D	W_1	25.68	16.73	13.90	13.81	**12.90**
	W_2	26.13	17.13	14.23	14.17	**12.68**
	MMD	0.35	0.15	0.10	0.12	**0.09**
250D	W_1	30.36	22.13	17.67	18.07	**16.27**
	W_2	30.77	22.49	17.95	18.36	**16.46**
	MMD	0.31	0.15	0.11	0.12	**0.11**
685D	W_1	33.81	33.45	25.26	25.68	**23.28**
	W_2	34.07	33.56	25.45	25.88	**23.44**
	MMD	0.39	0.20	0.14	0.16	**0.11**

We apply our method to a specific Tedsim scenario, modeling the transport from a single progenitor cell to two distinct terminal states. This controlled experiment allows us to quantitatively assess our method's ability to accurately reconstruct branching trajectories where the true pathways are known beforehand. See Figure 4 and Table 1. The experiments highlight our model's effectiveness in reconstructing branching trajectories. We observed that sub-linearity exponents of $\alpha = 0.5$ consistently produced well-defined bifurcating structures. We compared different approaches using the following metrics: W1/W2: Wasserstein distance with ground metrics, see Section 2. MMD: Maximum Mean Discrepancy using a Radial Basis Function kernel with a median heuristic bandwidth (Gretton et al., 2012).

Single-cell RNA. We now evaluate the performance of our method on the 50D Paul15 dataset. This dataset (Paul et al., 2015) contains single-cell RNA sequencing data from approximately 2,730 myeloid progenitor cells, with expression profiles for roughly 1,000 highly variable genes. The dataset captures various cell types in the hematopoietic system, including early progenitor cells, monocytes, and neutrophils. This dataset represents cellular states during myeloid differentiation and serves as a benchmark for studying developmental trajectories and cell fate decisions in hematopoiesis. We model transport from early progenitor cells to mature, differentiated cell types (monocytes and neutrophils). For comparison with Branched SBM (Tang et al., 2025) and FM (Lipman et al., 2022), see Table 1 and 2.

8 Image Data

FFHQ dataset. To test our model in higher dimensions we run experiments in 512D latent space of a pretrained ALAE model on FFHQ 1024x1024 dataset. The ALAE is used *only* for decoding latents to images during evaluation and remains frozen at all times. We load latent vectors paired with gender labels and split them into 60k training and 10k test samples. Let $\mathcal{D}$ denote the latent dimensionality inferred from the data. **Settings**: **1)** *Unconditional generation*: learn a vector field that transports samples from a standard Gaussian prior $\mu_0 \sim \mathcal{N}(0, I_{\mathcal{D}})$ to the empirical distribution of latent data μ_1. **2)** *Domain translation*: learn class-conditional transport from source-class latents to target-class latents (e.g. female $\rightarrow$ male); mini-batches for μ_0 and μ_1 are sampled from the respective class-specific training subsets.



Figure 5: *Female-to-male domain translation in the ALAE latent space. Two-step ODE results. Our method produces compatible results in a higher-dimensional latent space.*

Architecture: v_θ is an MLP with Tanh activations and hidden width 1024. A scalar time embedding (linear layer) is concatenated to x before the MLP. We use the same architecture for fair comparison for all types of model, except for Mean Flows which requires additional input r to encode the start of sampling trajectory. **Baselines**: We compare with Flow Matching (FM) and Mean Flows (MF) (Geng et al., 2025). All methods are trained for 100k iterations with a learning rate of 10^{-4} using the Adam optimizer.

Results: Figure 5 visualizes the domain translation results. Top row: samples from the source distribution μ_0 (female faces). Subsequent rows show the corresponding translations in the target distribution μ_1 (male faces) using FM, MF, and Y-Flows. Qualitatively, Y-Flows better preserves key attributes such as apparent age and skin tone. Quantitatively, we report the Fréchet Distance (**FD**) on FFHQ latents. As shown in Table 3, Y-Flows achieves the best scores and achieves high-quality translations in as few as two steps. See Figure 7 for the unconditional generation. This justifies our theoretical assumption of compression of time in our method. Appendix I reports FD-versus-steps curves (1–128 steps, three seeds), ablations of α, λ, the endpoint loss, speed-bounding remedies and the Jacobian estimator, a branching metric, and raw-pixel MNIST (784-D) and CelebA-64 (12,288-D) experiments: with the Kantorovich-dual endpoint, Y-Flows reach near-converged quality in 2–4 steps, while FM attains a slightly lower FD at convergence.

Table 3: Fréchet distance **val** vs. steps by model (↓) on FFHQ.

Steps	FM	Mean Flows	Y-Flows
1	35.17	35.87	35.27
2	26.33	23.56	**23.45**

Run-time: All experiments were conducted on a single NVIDIA RTX A6000 GPU. The average training time per experiment takes $\sim$ 1 hour. To *reproduce* our experiments, refer to the supplementary materials. The code will be open-source.

9 Broader Impact and Limitations

Y-Flows can improve efficient and interpretable generative modeling for structured data such as biological, geometric, and image domains. However, the method depends on sublinear velocity costs and deterministic ODE dynamics, and may require extra regularization for stability in some settings (see Appendix G).

10 Conclusion

In this work, we introduced Y-shaped generative flows, a continuous-time framework that addresses the structural limitations of standard V-shaped transport by rewarding shared movement before branching. By minimizing a novel velocity-power action with a sublinear exponent $\alpha \in (0, 1)$, we established a theoretical link between scalable neural ODEs and branched structures. We demonstrated that this concave velocity dependence induces a time-compression effect, favoring fast, shared transport along trunks followed by efficient branching. Empirically, our method recovers interpretable lineage structures in biological data, respects geometric constraints in LiDAR navigation, and achieves high-quality image generation with significantly fewer integration steps. Ultimately, Y-Flows offer a theoretically grounded and practically effective approach to learning hierarchy-aware generative trajectories that adapt computational effort to the underlying data structure.

AI USE STATEMENT

We used an LLM only for grammar and language editing of the manuscript. It was not used for generating research ideas, methods, code, experiments, or analysis. All AI-assisted edits were reviewed by the authors. We take full responsibility for the final content of this work.

REFERENCES

Marc Bernot, Vicent Caselles, and Jean-Michel Morel. Traffic plans. *Publicacions Matemàtiques*, pp. 417–451, 2005.

Lorenzo Brasco, Giuseppe Buttazzo, and Filippo Santambrogio. A benamou–brenier approach to branched transport. *SIAM journal on mathematical analysis*, 43(2):1023–1040, 2011.

Giuseppe Buttazzo and Eugene Stepanov. Optimal transportation networks as free dirichlet regions for the monge-kantorovich problem. *Annali della Scuola Normale Superiore di Pisa-Classe di Scienze*, 2(4):631–678, 2003.

Ricky TQ Chen, Yulia Rubanova, Jesse Bettencourt, and David K Duvenaud. Neural ordinary differential equations. *Advances in neural information processing systems*, 31, 2018.

Marco Cuturi. Sinkhorn distances: Lightspeed computation of optimal transport. In *Advances in neural information processing systems*, pp. 2292–2300, 2013.

Jean Feydy, Thibault Séjourné, François-Xavier Vialard, Shun-ichi Amari, Alain Trouvé, and Gabriel Peyré. Interpolating between optimal transport and mmd using sinkhorn divergences. In *The 22nd international conference on artificial intelligence and statistics*, pp. 2681–2690. PMLR, 2019.

Kevin Frans, Danijar Hafner, Sergey Levine, and Pieter Abbeel. One step diffusion via shortcut models. *arXiv preprint arXiv:2410.12557*, 2024.

Zhengyang Geng, Mingyang Deng, Xingjian Bai, J Zico Kolter, and Kaiming He. Mean flows for one-step generative modeling. *arXiv preprint arXiv:2505.13447*, 2025.

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

Arthur Gretton, Karsten M. Borgwardt, Malte J. Rasch, Bernhard Schölkopf, and Alexander J. Smola. A kernel two-sample test. *J. Mach. Learn. Res.*, 13:723–773, 2012. URL http://dl.acm.org/citation.cfm?id=2188410.

Nikita Gushchin, Alexander Kolesov, Alexander Korotin, Dmitry P Vetrov, and Evgeny Burnaev. Entropic neural optimal transport via diffusion processes. *Advances in Neural Information Processing Systems*, 36:75517–75544, 2023.

Jonathan Ho, Ajay Jain, and Pieter Abbeel. Denoising diffusion probabilistic models. *Advances in neural information processing systems*, 33:6840–6851, 2020.

Peter Holderrieth, Marton Havasi, Jason Yim, Neta Shaul, Itai Gat, Tommi Jaakkola, Brian Karrer, Ricky TQ Chen, and Yaron Lipman. Generator matching: Generative modeling with arbitrary markov processes. *arXiv preprint arXiv:2410.20587*, 2024.

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

Peter Lippmann, Enrique Fita Sanmartín, and Fred A Hamprecht. Theory and approximate solvers for branched optimal transport with multiple sources. *Advances in Neural Information Processing Systems*, 35:267–279, 2022.

Guan-Horng Liu, Yaron Lipman, Maximilian Nickel, Brian Karrer, Evangelos A Theodorou, and Ricky TQ Chen. Generalized schrödinger bridge matching. *arXiv preprint arXiv:2310.02233*, 2023.

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

Francesco Maddalena, Sergio Solimini, and Jean-Michel Morel. A variational model of irrigation patterns. *Interfaces and Free Boundaries*, 5(4):391–415, 2003. doi: 10.4171/IFB/85.

Edouard Oudet and Filippo Santambrogio. A modica-mortola approximation for branched transport and applications. *Archive for rational mechanics and analysis*, 201(1):115–142, 2011.

Xinhai Pan, Hechen Li, and Xiuwei Zhang. TedSim: temporal dynamics simulation of single-cell RNA sequencing data and cell division history. *Nucleic Acids Research*, 50(8):4272–4288, 2022. doi: 10.1093/nar/gkac235.

Franziska Paul, Yaara Arkin, Amir Giladi, Diego Adhemar Jaitin, Ephraim Kenigsberg, Hadas Keren-Shaul, Deborah Winter, David Lara-Astiaso, Meital Gury, Assaf Weiner, Eyal David, Nadav Cohen, Felicia Kathrine Lauridsen, Simon Haas, Andreas Schlitzer, Alexander Mildner, Florent Ginhoux, Steffen Jung, Andreas Trumpp, Bo Torben Porse, Amos Tanay, and Ido Amit. Transcriptional heterogeneity and lineage commitment in myeloid progenitors. *Cell*, 163(7):1663–1677, 2015.

Filippo Santambrogio. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*, volume 87 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser, 2015.

Sophia Tang, Yinuo Zhang, Alexander Tong, and Pranam Chatterjee. Branched schrödinger bridge matching. *arXiv preprint arXiv:2506.09007*, 2025.

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. *arXiv preprint arXiv:2302.00482*, 2023.

Cédric Villani. *Optimal transport: old and new*, volume 338. Springer Science & Business Media, 2008.

Qinglan Xia. Optimal paths related to transport problems. *Communications in Contemporary Mathematics*, 5(02):251–279, 2003.

A Background (Extended)

Continuous Normalizing Flow A Continuous Normalizing Flow (CNF) (Chen et al., 2018) is a generative model that defines a probability density path through a Neural Ordinary Differential Equation (ODE):

$$\frac{d}{dt}x_t = v_\theta(x_t, t), \qquad x_{t=0} \sim \mu_0.$$

Let Φ_t be the flow map associated with this ODE, which transports a particle from its initial condition at time 0 to its location at time t. The pushforward density $\rho_t = (\Phi_t)_{\#}\rho_0$ that evolves under this dynamics *necessarily* satisfies the *continuity equation* with the parameterized velocity field v_θ. The continuity equation encodes the law of mass conservation:

$$\partial_t \rho_t + \nabla \cdot (\rho_t v_t) = 0 \quad \text{on } \Omega \times (0,1),$$
$$\rho_{t=0} = \rho_0, \quad \rho_{t=1} = \rho_1.$$

A key result is the instantaneous change of variables formula, which describes how the log density evolves along a trajectory:

$$\frac{d}{dt}\log \rho_t(x_t) = -\nabla \cdot v_\theta(x_t, t).$$

This allows for a likelihood calculation by integrating this quantity over time. Training can be done by directly maximizing likelihood.

Flow Matching. The core idea of Flow Matching (FM)(Lipman et al., 2022; Tong et al., 2023) is to train a CNF by directly regressing its velocity field v_θ towards a target vector field u_t that generates a desired probability path. The Flow Matching objective is: $\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, x\sim\rho_t}\left[\|v_\theta(x,t) - u_t(x)\|^2\right]$.

A critical challenge is that sampling $x \sim \rho_t$ from the marginal path at arbitrary times is typically intractable. Conditional Flow Matching (CFM) (Lipman et al., 2022) provides a solution by constructing the marginal path as a mixture of simpler and tractable conditional paths. Let z be a conditioning variable with distribution $q(z)$. We define the marginal path as follows: $\rho_t(x) = \int \rho_t(x \mid z)q(z)\,dz$, where each conditional path $\rho_t(x|z)$ is generated by a corresponding conditional vector field $u_t(x \mid z)$. The marginal field $u_t(x)$ that generates ρ_t is then given by:

$$u_t(x) \;=\; \mathbb{E}_{z\sim q(z|x)}[u_t(x \mid z)] = \mathbb{E}_{z\sim q}\left[\frac{\rho_t(x \mid z)}{\rho_t(x)}\, u_t(x \mid z)\right],$$

where $q(z \mid x)$ is the posterior. The key theorem is to minimize the following *Conditional Flow Matching* objective:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim\mathcal{U}[0,1],\, z\sim q,\, x\sim\rho_t(\cdot|z)}\left[\|v_\theta(x,t) - u_t(x \mid z)\|^2\right] \tag{10}$$

yields the same gradient for θ as minimizing the intractable $\mathcal{L}_{\text{FM}}(\theta)$. This makes CFM a practical objective, as it only requires sampling from conditional paths $\rho_t(x|z)$ and knowing their closed-form drifts $u_t(x|z)$.

The flexibility of CFM lies in the choice of conditional paths. The coupling $q(z)$ is the independent joint distribution $q(x_0)q(x_1)$, so $z = (x_0, x_1)$. A common conditional path is a Gaussian bridge: $\rho_t(x \mid z) = \mathcal{N}(x \mid \mu_t, \sigma^2 I)$, where $\mu_t = (1-t)x_0 + tx_1$ is a linear interpolation. The simple constant drift that generates this path is $u_t(x \mid z) = x_1 - x_0$. The optimal transport CFM (OT-CFM) (Tong et al., 2023) method uses an optimal coupling to define conditionals. Here, $z = (x_0, x_1)$ is sampled from an OT plan π between μ_0 and μ_1 (Section 2). In practice, the OT plan π is efficiently approximated using mini-batch OT, which has been shown to work well empirically.

B Modica-Mortola Branched Flows

Oudet & Santambrogio (2011) proposed a theoretically grounded approximation inspired by the Modica-Mortola framework used in phase-field modeling. This framework replaces the singular, non-smooth branched transport problem with a sequence of regularized, elliptic energy functionals

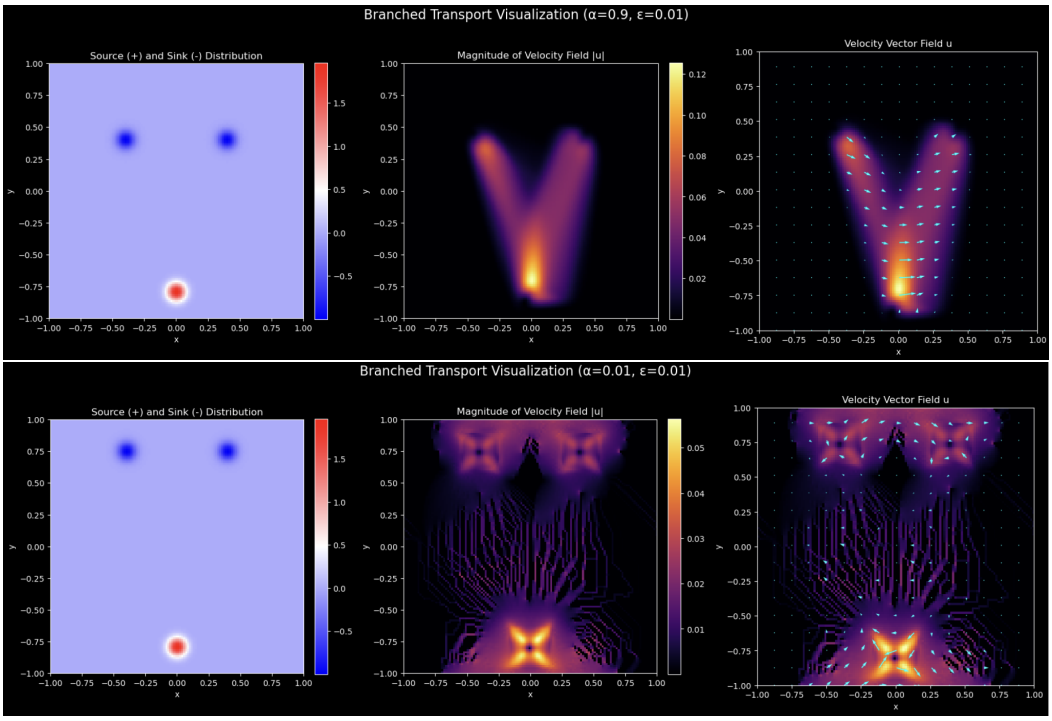


Figure 6: Modica-Mortola solver results using a neural network-based field parametrization. The difference between the top and bottom rows illustrates the sensitivity of the method to optimization parameters, specifically the learning rate. We observe that the method is highly sensitive to initialization and optimization settings.

M_λ^α, defined over the more regular space of vector fields $\mathcal{H}^1$. For a given vector field $F(x)$, the approximating functional is defined as:

$$M_\lambda^\alpha(F) = \lambda^{\gamma_1} \int_\Omega |F(x)|^\alpha dx + \lambda^{\gamma_2} \int_\Omega |\nabla F(x)|^2 dx, \tag{11}$$

where $\lambda > 0$ is a small regularization parameter. The exponents γ_1 and γ_2 are derived from the transport dimension d and the cost exponent α to ensure correct scaling behavior. As described by Oudet & Santambrogio (2011), the functional in (11) consists of two competing terms:

1. **A concave potential term** ($\int |F|^\alpha dx$): This term acts analogously to a double-well potential, encouraging the magnitude of the vector field to be either zero or arbitrarily large, thereby promoting the formation of sparse high-intensity pathways.
2. **A Dirichlet energy term** ($\int |\nabla F|^2 dx$): This Sobolev penalty penalizes spatial variations, enforcing smoothness and regularity on the vector field.

The central theoretical guarantee is that as $\lambda \to 0$, the functionals M_λ^α Γ-converge to the original branched transport energy M^α. This ensures that minimizers of the regularized problem converge to a minimizer of the intractable branched transport problem.

Although the original implementation by Oudet & Santambrogio (2011) relied on a staggered grid discretization and FFT-based projections to enforce the divergence constraint, we reformulated this variational principle within a deep learning framework.

In our experiments, we parameterized the vector field F as a coordinate-based neural network (an MLP) $F_\theta : \Omega \to \mathbb{R}^d$, implemented in PyTorch. Unlike grid-based methods, this allows for a mesh-free representation of the transport density. The optimization objective $\mathcal{L}(\theta)$ is composed of the relaxed energy functional and a soft penalty for the divergence constraint:

$$\mathcal{L}(\theta) = M_\lambda^\alpha(F_\theta) + \beta \|\nabla \cdot F_\theta - (\mu_0 - \mu_1)\|^2, \tag{12}$$

where β is a penalty weight (In practice, we find it important to put this value high, in our experiments, the method worked only with $\beta > 10k$). We utilized PyTorch's automatic differentiation

(`torch.autograd`) to compute the spatial Jacobian ∇F_θ for the Dirichlet term and the divergence $\nabla \cdot F_\theta$ for the constraint term exact to the network precision. The integrals were approximated by Monte Carlo sampling in the domain Ω.

We evaluated this approach in a simple setting with a single source and two symmetric targets. Although the method successfully generated the branching structures shown in Figure 6, we found that the training dynamics is notoriously difficult to stabilize. The optimization landscape is highly sensitive to hyperparameters; for instance, training diverged significantly at a learning rate of 1e-4, yet converged to a reasonable solution at 1e-5.

We attribute this difficulty to the inherent nature of the flux variable in the Modica-Mortola formulation. Since F represents a mass flux (density $\times$ velocity), optimal solutions tend toward singularity (Dirac structures) as $\lambda \to 0$. Neural networks often struggle with such high-frequency spectral components (spectral bias), making the simultaneous minimization of the concave potential and the divergence constraint numerically unstable without careful scheduling of the regularization parameter λ.

C Low-Dimensional Illustration: Optimal Branching Time

We consider a dynamic problem on a domain $\Omega \subset \mathbb{R}^d$ over a time interval $t \in [0, 1]$. The goal is to transport a source distribution ρ_0 to a target ρ_1 while minimizing a functional composite energy. The density ρ_t evolves according to the continuity equation $\partial_t \rho + \nabla \cdot (\rho v) = 0$. The total energy functional is given by:

$$J(\rho, v) = \underbrace{\int_0^1 \int_\Omega \rho_t(x) \|v_t(x)\|^\alpha \, dx \, dt}_{\mathcal{T}(\rho,v)} + \lambda \underbrace{\int_0^1 \int_\Omega \rho_t(x) \|\nabla v_t(x)\|_F^2 \, dx \, dt}_{\mathcal{C}(\rho,v)} \tag{13}$$

where $\alpha \in (0, 1)$ is the sub-linear transport exponent, $\lambda > 0$ is the cohesion weight, and $\| \cdot \|_F$ denotes the Frobenius norm.

Symmetric Low-Dimensional Setup

To analyze the branching behavior, we consider a simplified symmetric case:

- **Source:** A mass distribution concentrated at $P = (0, 0)$ at $t = 0$.
- **Targets:** Two symmetric targets $Q_1 = (-w, h)$ and $Q_2 = (w, h)$ in $t = 1$.
- **Particle Model:** We model the mass as a connected cloud of characteristic width ϵ.
- **Trajectory Strategy:** We parameterize the path by a branching time $\tau \in [0, 1]$.
  - For $t \in [0, \tau)$, the mass moves as a single cluster (Trunk).
  - For $t \in [\tau, 1]$, the mass splits and moves towards the respective targets (Branches).

The optimization problem reduces to finding τ that minimizes the trade-off between the **Transport Cost** (path length/speed efficiency) and the **Cohesion Cost** (velocity field regularity).

Proposition 1 (Zero-Cost of Uniform Motion)**.** *If the velocity field $v(x, t)$ represents a spatially uniform translation (rigid body motion without rotation) on the support of ρ_t, the Cohesion Cost density is zero.*

Proof. Let $\mathrm{supp}(\rho_t)$ denote the spatial support of the density at time t. Assume $v(x, t) = \mathbf{u}(t)$ for all $x \in \mathrm{supp}(\rho_t)$, where $\mathbf{u}(t)$ depends only on time. The Frobenius norm of the velocity gradient is defined as:

$$\|\nabla v\|_F^2 = \sum_{i,j} \left(\frac{\partial v_i}{\partial x_j} \right)^2 \tag{14}$$

Since v is constant with respect to the spatial coordinates x, the partial derivatives $\frac{\partial v_i}{\partial x_j}$ vanish identically. Consequently:

$$\int_{\mathrm{supp}(\rho_t)} \rho_t(x) \|\nabla v(x, t)\|_F^2 \, dx = \int \rho_t(x) \cdot 0 \, dx = 0 \tag{15}$$

Thus, the *Trunk* phase of the trajectory incurs zero penalty from the cohesion term. □

Proposition 2 (The Cost of Separation)**.** *If a continuous velocity field separates the mass into distinct directions over a transition region of width ϵ, the Cohesion Cost scales as $\mathcal{O}(1/\epsilon)$.*

Proof. Consider a 1D cross-section of the particle cloud at the moment of splitting, defined on the interval $x \in [-\epsilon, \epsilon]$. To achieve separation:

- For $x \in [-\epsilon, 0)$, $v(x) \approx -u$ (moving left).
- For $x \in (0, \epsilon]$, $v(x) \approx +u$ (moving right).

Assuming v is differentiable, it must transition from $-u$ to $+u$ over a distance of 2ϵ. By the Mean Value Theorem applied to each component, there exists a point x_0 in the interval where the gradient magnitude satisfies $|\nabla v(x_0)| \geq \frac{2u}{2\epsilon} = \frac{u}{\epsilon}$. The contribution to the energy integral approximates:

$$\int_{-\epsilon}^{\epsilon} \rho \|\nabla v\|^2 dx \approx \rho_{\text{avg}} \cdot (2\epsilon) \cdot \left(\frac{u}{\epsilon}\right)^2 \propto \frac{1}{\epsilon} \tag{16}$$

As $\epsilon \to 0$, this cost diverges, creating an infinite barrier against the immediate separation of a singular measure. □

Proposition 3 (Time-Compression)**.** *For $\alpha \in (0, 1)$, the transport cost of traveling a fixed distance D decreases as the duration of the travel T decreases.*

Proof. Consider a particle traveling a distance D over duration T with constant speed $|v| = D/T$. The Transport Cost is

$$C(T) = \int_0^T |v|^\alpha dt = T \cdot \left(\frac{D}{T}\right)^\alpha = D^\alpha T^{1-\alpha} \tag{17}$$

We analyze the sensitivity of the cost to the duration T:

$$\frac{dC}{dT} = D^\alpha (1-\alpha) T^{-\alpha} \tag{18}$$

Since $\alpha \in (0, 1)$, the term $(1 - \alpha)$ is strictly positive. Therefore, $\frac{dC}{dT} > 0$. This implies that reducing travel time T (and consequently increasing velocity) strictly reduces the total energy cost. This property favors "impulsive" motion: waiting in the trunk (where cost is low) and then bursting to the target in the remaining short time window. □

Lemma 1 (Optimal branching time in a low-dimensional illustration)**.** *For a sufficiently large cohesion weight λ and $\alpha \in (0, 1)$, the optimal branching time $\tau^* > 0$. This implies that a Y-shaped trajectory is energetically superior to a V-shaped trajectory (where $\tau = 0$).*

Proof. We minimize the total energy $E(\tau)$ with respect to the branching time $\tau \in [0, 1)$. Let the trunk length be τh and the remaining branch length be $L_B(\tau) = \sqrt{w^2 + h^2(1-\tau)^2}$.

The total energy is the sum of the trunk transport, branch transport, and branch cohesion costs:

$$E(\tau) = \underbrace{(\tau h)^\alpha \tau^{1-\alpha}}_{\text{Trunk } \mathcal{T}} + \underbrace{L_B(\tau)^\alpha (1-\tau)^{1-\alpha}}_{\text{Branch } \mathcal{T}} + \underbrace{\lambda K_{\text{split}}(1-\tau)}_{\text{Branch } \mathcal{C}} \tag{19}$$

Simplifying the terms (using $\tau^\alpha \tau^{1-\alpha} = \tau$):

$$E(\tau) = h^\alpha \tau + \left(\sqrt{w^2 + h^2(1-\tau)^2}\right)^\alpha (1-\tau)^{1-\alpha} + \lambda K_{\text{split}}(1-\tau) \tag{20}$$

We evaluate the derivative at $\tau = 0$ (the V-shaped configuration) to check for a descent direction.

$$E'(0) = h^\alpha + \left[\frac{d}{d\tau}\left(L_B(\tau)^\alpha (1-\tau)^{1-\alpha}\right)\right]_{\tau=0} - \lambda K_{\text{split}} \tag{21}$$

The dominant term in the bracket comes from differentiating $(1-\tau)^{1-\alpha}$. At $\tau = 0$, this derivative is $-(1-\alpha)$. The cohesion term contributes $-\lambda K_{\text{split}}$. Thus, the derivative has the form:

$$E'(0) \approx h^\alpha - (1-\alpha) L_B(0)^\alpha - \lambda K_{\text{split}} \tag{22}$$

Since $L_B(0) > h$ (the hypotenuse is longer than the vertical leg) and $\lambda K_{\text{split}} > 0$, for sufficiently large λ or small α, we have:

$$E'(0) < 0 \tag{23}$$

Since the derivative is negative at $\tau = 0$, the energy decreases as τ increases. Therefore, the minimum must occur at some $\tau^* > 0$, confirming the optimality of the Y-shape. □

D Relation to Branched Transport: A Density–Velocity Action

Y-Flows are *not* obtained by algebraically combining the branched Benamou–Brenier problem (4) with the Modica–Mortola functional (5); the two formulations motivate different design principles. Branched transport motivates an economy of shared transport, whereas Modica–Mortola motivates a balance between concentration and spatial regularity. To make the relation between these mechanisms and our objective explicit, consider the broader density–velocity action

$$\mathcal{A}_{\beta,\alpha}(\rho, v) = \int_0^1 \int_\Omega \rho_t(x)^\beta \, \|v_t(x)\|^\alpha \, dx \, dt, \qquad 0 < \beta \le 1, \quad 0 < \alpha \le 1, \tag{24}$$

together with the full objective $\mathcal{E}_{\beta,\alpha}(\rho, v) = \mathcal{A}_{\beta,\alpha}(\rho, v) + \lambda \mathcal{C}(\rho, v)$, where $\mathcal{C}$ is the cohesion term of (13). Written in terms of the flux $m_t = \rho_t v_t$, the transport term reads

$$\mathcal{A}_{\beta,\alpha}(\rho, m) = \int_0^1 \int_\Omega \rho_t(x)^{\beta-\alpha} \, \|m_t(x)\|^\alpha \, dx \, dt. \tag{25}$$

The two exponents control different effects: density concavity ($\beta < 1$) creates the classical *shared-mass economy* of branched transport, whereas velocity concavity ($\alpha < 1$) creates *temporal compression* by favouring fast motion over short intervals. Classical dynamic branched transport (4) is the slice $0 < \beta < 1$, $\alpha = 1$; the practical Y-Flows objective (6) is the mass-linear slice $\beta = 1$, $0 < \alpha < 1$.

Proposition 4 (Velocity concavity preserves the branched-transport geometry)**.** *Let Γ be a transport network along which a branch of mass θ and length ℓ is traversed by a path x_t with speed $\|\dot{x}_t\| \le v_{\max}$. Then for $0 < \alpha \le 1$,*

$$\theta^\beta \int \|\dot{x}_t\|^\alpha \, dt \;\ge\; \theta^\beta \, v_{\max}^{\alpha-1} \, \ell, \tag{26}$$

and for $0 < \alpha < 1$ equality holds if and only if $\|\dot{x}_t\| \in \{0, v_{\max}\}$ for almost every t. Consequently, minimizing the time parametrization of a fixed network gives $\inf \mathcal{A}_{\beta,\alpha} = v_{\max}^{\alpha-1} \int_\Gamma \theta^\beta \, d\mathcal{H}^1$, i.e., the branched-transport network cost (3) up to a constant factor.

Proof. Let $r(t) = \|\dot{x}_t\| \in [0, v_{\max}]$ and $\ell = \int r(t)\, dt$. For $r(t) > 0$ we have $r(t)^\alpha = r(t)\, r(t)^{\alpha-1} \ge r(t)\, v_{\max}^{\alpha-1}$, because $\alpha - 1 \le 0$ and $r(t) \le v_{\max}$; for $r(t) = 0$ both sides vanish. Integrating in time yields (26). For $\alpha < 1$ the inequality $r^{\alpha-1} \ge v_{\max}^{\alpha-1}$ is strict unless $r = v_{\max}$, so equality holds iff $r(t) \in \{0, v_{\max}\}$ a.e. Summing (26) over the branches of Γ gives the network cost. □

Proposition 4 shows that velocity concavity does not destroy the branched-transport structure when density concavity is present: the optimal network geometry remains that of branched transport, while $\alpha < 1$ additionally selects maximal-speed traversal over the shortest possible active interval (at $\alpha = 1$ every time parametrization has the same cost, so this preference disappears). A direct dynamic extension with $\beta < 1$, however, requires estimating the evolving density and drives the mass onto increasingly singular, lower-dimensional structures, which makes optimization difficult (see Appendix B). The practical objective therefore uses the tractable slice $\beta = 1$, which retains time compression but removes exact mass sub-additivity, and replaces the density-concave mechanism by the cohesion term $\mathcal{C}$.

The cohesion term is a velocity-coherence regularizer. The term $\mathcal{C}$ should not be interpreted as a mass-concavity term but as a penalty on velocity disagreement. Under a weighted Poincaré inequality for ρ_t with constant C_P (Assumption 1 below), the between-group velocity disagreement of any partition of the mass into groups with weights p_k and mean velocities $u_k(t)$ satisfies

$$\sum_k p_k \, \|u_k(t) - \bar{u}(t)\|^2 \;\le\; C_P \int_\Omega \rho_t(x) \, \|\nabla v_t(x)\|_F^2 \, dx. \tag{27}$$

Hence assigning different velocities to future terminal groups while they are still spatially mixed requires Jacobian energy. The cohesion term does not single out a unique hierarchy; it discourages early velocity disagreement and thereby creates an inductive bias toward delayed separation. In this sense the final objective defines a learning-compatible notion of *Y-shaped optimality*, distinct from branched-transport optimality: time compression combined with delayed velocity differentiation. Appendix E makes this precise.

E Population-Level Characterization and Delayed-Separation Theorem

Appendix C is a low-dimensional illustration with a prescribed trunk-then-branch trajectory family. A deterministic Lipschitz ODE cannot make distinct trajectories merge exactly and later split, so a "Y-shaped flow" has to be understood at the population level rather than as a topological bifurcation of individual trajectories. Here we define it as *velocity-coherent shared motion followed by delayed terminal-group separation*, and prove, for arbitrary admissible velocity fields, that low values of the full objective (6) force delayed separation.

Setup. Let $X_0 \sim \mu_0$, $\dot{X}_t = v_t(X_t)$ and $X_t = \Phi_t(X_0)$, so that $\rho_t = \mathrm{Law}(X_t)$ solves the continuity equation (1). Assume the target is partitioned into terminal regions $A_1, \dots, A_K$ and define the terminal label $C = k \Leftrightarrow X_1 \in A_k$, with $p_k = \mathbb{P}(C = k) > 0$. Define the conditional means and velocities

$$m_k(t) = \mathbb{E}[X_t \mid C = k], \qquad u_k(t) = \mathbb{E}[v_t(X_t) \mid C = k],$$
$$\bar{m}(t) = \textstyle\sum_k p_k m_k(t) = \mathbb{E}[X_t], \qquad \bar{u}(t) = \textstyle\sum_k p_k u_k(t) = \mathbb{E}[v_t(X_t)],$$

and the population statistics

$$B(t) = \sum_{k=1}^{K} p_k \|m_k(t) - \bar{m}(t)\|^2, \qquad V(t) = \sum_{k=1}^{K} p_k \|u_k(t) - \bar{u}(t)\|^2, \qquad W(t) = \mathbb{E}\|X_t - \bar{m}(t)\|^2. \tag{28}$$

$B(t)$ measures terminal-group separation, $V(t)$ velocity disagreement between groups, and $W(t)$ the width of the whole population. For $q \in (0,1)$ define the *branching time*

$$\tau_q = \inf\{t \in [0,1] : \; B(t) - B(0) \geq q\,(B(1) - B(0))\}, \qquad B_q = B(0) + q\,(B(1) - B(0)). \tag{29}$$

Definition 1 (Y-shaped flow, population level; informal)**.** *A flow (ρ_t, v_t) is Y-shaped if $W(t)$, $B(t)$ and $V(t)$ remain small on an initial interval $[0, \tau)$ with $\tau > 0$, while $B(1) > B(0)$ because distinct terminal regions must eventually be reached.*

Assumption 1 (Weighted Poincaré inequality)**.** *There is $C_P < \infty$ such that for every $t \in [0,1]$ and every $f \in H^1(\rho_t; \mathbb{R}^d)$, $\mathrm{Var}_{\rho_t}(f) = \int_\Omega \rho_t \|f - \int \rho_t f\|^2 \, dx \leq C_P \int_\Omega \rho_t \|\nabla f\|_F^2 \, dx$.*

For $t \in [0,1]$ write the accumulated transport action, cohesion energy and total objective as

$$\mathcal{A}_\alpha(t) = \int_0^t \int_\Omega \rho_s \|v_s\|^\alpha \, dx \, ds, \quad \mathcal{C}(t) = \int_0^t \int_\Omega \rho_s \|\nabla v_s\|_F^2 \, dx \, ds, \quad \mathcal{J}_{\alpha,\lambda}(t) = \mathcal{A}_\alpha(t) + \lambda \mathcal{C}(t),$$

so that $\mathcal{J}_{\alpha,\lambda}(1)$ is the objective (6). Let $\Delta_B(t) = \big(\sqrt{B(t)} - \sqrt{B(0)}\big)_+$.

Theorem 1 (restated) (Delayed separation for arbitrary admissible flows)**.** *Let $0 < \alpha < 1$, let (ρ_t, v_t) be an admissible flow with $\|v_t(x)\| \leq v_{\max}$, and let Assumption 1 hold. Then for every $t \in [0,1]$:*

$$\Delta_B(t)^2 \leq t \, \min\big\{ \, v_{\max}^{2-\alpha} \, \mathcal{A}_\alpha(t), \; C_P \, \mathcal{C}(t) \, \big\}, \tag{30}$$

$$\Delta_B(t)^4 \leq C_P \, v_{\max}^{2-\alpha} \, t^2 \, \mathcal{A}_\alpha(t) \, \mathcal{C}(t), \tag{31}$$

$$\mathcal{J}_{\alpha,\lambda}(t) \geq \frac{\Delta_B(t)^2}{t} \Big(v_{\max}^{\alpha-2} + \frac{\lambda}{C_P} \Big). \tag{32}$$

Corollary 2 (Lower bound on the branching time)**.** *If a minimizer or ε-minimizer satisfies $\mathcal{J}_{\alpha,\lambda}(1) \leq E$ and $B(1) > B(0)$, then for every $q \in (0,1)$*

$$\tau_q \;\geq\; \frac{\big(\sqrt{B_q} - \sqrt{B(0)}\big)^2}{E} \Big(v_{\max}^{\alpha-2} + \frac{\lambda}{C_P} \Big). \tag{33}$$

In particular, a low-energy solution cannot separate the terminal groups immediately: rapid early separation requires both transport action and cohesion energy.

Proof of Theorem 1. Step 1 (separation is driven by velocity disagreement). Since $t \mapsto X_t$ is absolutely continuous with $\dot{X}_t = v_t(X_t)$ and $\|v_t\| \le v_{\max}$, we may differentiate under the (conditional) expectation to get $\dot{m}_k(t) = u_k(t)$ and $\dot{\bar{m}}(t) = \bar{u}(t)$. Hence

$$\begin{aligned}\dot{B}(t) &= 2\sum_k p_k \left\langle m_k(t) - \bar{m}(t),\, u_k(t) - \bar{u}(t) \right\rangle \\ &\le 2\Big(\sum_k p_k \|m_k - \bar{m}\|^2\Big)^{1/2} \Big(\sum_k p_k \|u_k - \bar{u}\|^2\Big)^{1/2} = 2\sqrt{B(t)}\sqrt{V(t)}\end{aligned}$$

by the Cauchy–Schwarz inequality for the weights p_k. Wherever $B > 0$ this gives $\frac{d}{dt}\sqrt{B(t)} \le \sqrt{V(t)}$; integrating (and using the positive part when B vanishes) yields

$$\Delta_B(t) \le \int_0^t \sqrt{V(s)}\, ds \le \sqrt{t}\,\Big(\int_0^t V(s)\, ds\Big)^{1/2}, \tag{34}$$

where the last inequality is Cauchy–Schwarz in time.

Step 2 (two complementary controls on V). The between-group variance is dominated by the total variance (law of total variance): $V(t) = \sum_k p_k \|u_k - \bar{u}\|^2 \le \mathbb{E}\|v_t(X_t) - \bar{u}(t)\|^2 = \mathrm{Var}_{\rho_t}(v_t)$. On the one hand, Assumption 1 applied to $f = v_t$ gives

$$V(t) \le C_P \int_\Omega \rho_t \|\nabla v_t\|_F^2\, dx. \tag{35}$$

On the other hand, the speed bound gives

$$V(t) \le \mathrm{Var}_{\rho_t}(v_t) \le \int_\Omega \rho_t \|v_t\|^2\, dx = \int_\Omega \rho_t \|v_t\|^{2-\alpha} \|v_t\|^\alpha\, dx \le v_{\max}^{2-\alpha} \int_\Omega \rho_t \|v_t\|^\alpha\, dx. \tag{36}$$

Step 3 (conclusion). Integrating (35) and (36) over $[0,t]$ gives $\int_0^t V \le C_P\, \mathcal{C}(t)$ and $\int_0^t V \le v_{\max}^{2-\alpha} \mathcal{A}_\alpha(t)$; inserting into (34) proves (30). Multiplying the two bounds in (30) proves (31). Finally, (30) gives the lower bounds $\mathcal{A}_\alpha(t) \ge \Delta_B(t)^2\, v_{\max}^{\alpha-2}/t$ and $\lambda\,\mathcal{C}(t) \ge \lambda\,\Delta_B(t)^2/(C_P\, t)$; adding them proves (32). □

Proof of Corollary 2. B is continuous, so the infimum in (29) is attained and $B(\tau_q) = B_q \ge B(0)$, whence $\Delta_B(\tau_q)^2 = (\sqrt{B_q} - \sqrt{B(0)})^2$. Since the integrands of $\mathcal{A}_\alpha$ and $\mathcal{C}$ are nonnegative, $\mathcal{J}_{\alpha,\lambda}(\tau_q) \le \mathcal{J}_{\alpha,\lambda}(1) \le E$. Applying (32) at $t = \tau_q$ and rearranging gives (33). □

Interpretation. Theorem 1 couples both terms of the objective. The Poincaré term controls how different the group velocities can be, while the velocity action controls how fast the conditional means can move; (31) shows that terminal-group separation requires expenditure of *both* quantities, and (32)–(33) bound the branching time in terms of the complete objective. The role specific to $0 < \alpha < 1$ is temporal compression: by Proposition 4, among all parametrizations of the same path the sublinear action favours maximal-speed motion over the shortest possible active interval, with waiting elsewhere. Thus the cohesion term suppresses early terminal-group separation, while the sublinear velocity term compresses the unavoidable group-specific motion into a short interval; together with the endpoint requirement $B(1) > B(0)$, this yields the population-level Y-shape of Definition 1: an initially velocity-coherent bundle followed by delayed, temporally concentrated separation. The same argument applied to $W(t)$ yields an analogous bound on the growth of the population width. The statement is about delayed, energy-controlled separation in the sense of Definition 1, which is the notion of Y-shapedness that the objective enforces; whether a particular source–target pair additionally produces a topological Y shape depends on the geometry of μ_0 and μ_1.

F Kantorovich-Dual Endpoint Objective (YF-KD)

The Sinkhorn divergence in (9) is only one possible choice for the boundary-matching term. For pixel-space experiments we use a more scalable alternative inspired by the neural dual-potential

formulation of entropic neural optimal transport (ENOT) (Gushchin et al., 2023): the Y-shaped action is retained and the terminal marginal is enforced through a learned Kantorovich potential f_ψ via the saddle-point problem

$$\min_{\theta} \max_{\|f_\psi\|_{\mathrm{Lip}} \le 1} \hat{Y}^{\alpha}(\theta) + \lambda_{\mathrm{end}} \Big[\mathbb{E}_{X_1 \sim \rho_1^\theta} f_\psi(X_1) - \mathbb{E}_{Y \sim \mu_1} f_\psi(Y) \Big], \tag{37}$$

where $\rho_1^\theta = (\Phi_1^\theta)_{\#}\mu_0$ and $\lambda_{\mathrm{end}} > 0$ plays the role of λ_{sink} in (9). The potential enforces $\rho_1^\theta \approx \mu_1$ while the velocity field continues to minimize the Y-shaped action; this is the deterministic analogue of the ENOT training mechanism, with a deterministic neural ODE and a sublinear velocity cost in place of stochastic diffusion dynamics. The Lipschitz constraint is enforced with a standard gradient penalty evaluated at samples $\widehat{X}$, $\mathcal{L}_{\mathrm{GP}} = \lambda_{\mathrm{gp}} \mathbb{E}_{\widehat{X}} \big[(\|\nabla_{\widehat{X}} f_\psi(\widehat{X})\|_2 - 1)^2 \big]$. We refer to this variant as **YF-KD** and to the Sinkhorn variant of Section 5 as **YF-S**. Only the endpoint penalty changes; the critic is not needed at inference time.

MNIST details. The velocity U-Net has 1.67M parameters; the potential f_ψ has 0.43M parameters and uses three strided convolutional layers with LeakyReLU slope 0.2, trained with Adam at 2×10^{-4}, $\beta = (0.0, 0.9)$. The velocity model uses batch size 256, Adam at 10^{-3}, 12,000 iterations, EMA 0.999, early-stopping threshold $\delta = 10^{-3}$, one Hutchinson probe ($P = 1$), and $K = 4$ explicit Euler steps during training.

G Discussion and Limitations

Well-posedness for $\alpha \in (0, 1)$. For $\alpha \in (0, 1)$, the map $v \mapsto \|v\|^\alpha$ is sublinear, which induces the time-compression effect, but can also cause the continuous-time problem to degenerate if arbitrarily large speeds are allowed. A standard remedy is to enforce at least one of the following:

- a hard speed bound $\|v(t, x)\| \le v_{\max}$ almost everywhere, or
- a small convex stabilizer, e.g. $+\mu \int_0^1 \int_\Omega \rho \, \|v\|^2 \, dx \, dt$ with $\mu > 0$, or
- a smooth surrogate $\big(\|v\|^2 + \eta^2\big)^{\alpha/2}$ with $\eta > 0$. This can be useful for stable gradients near $v = 0$.

Any of these prevents "infinite-speed, zero-time" collapse while preserving the qualitative preference for bursty motion. In our experiments the unconstrained objective trained stably without these safeguards (Appendix I.2, Table 9, quantifies the resulting velocity distribution); they remain available when stronger control is required.

Density-weighting caveat. Because $\mathcal{C}$ is weighted by ρ, sharp gradients in v are cheap, where ρ is close to zero. If the model is free to create low-density *gaps* in the transition layer, it can partially bypass the separation barrier. In practice, one can mitigate this with density regularization (e.g. entropy) or by constraining how ρ is represented.

Deterministic ODE caveat (literal splitting of a point mass). A deterministic flow map sends a Dirac mass to a Dirac mass; it cannot turn one particle into two. In particle implementations, "branching" should therefore be interpreted as *different particles* (initially distributed under ρ_0) peeling into distinct modes, rather than a literal split of a single trajectory.

In general, it is important to note that if $\alpha \ge 1$, the time-compression advantage disappears (the speed penalty is no longer sublinear), weakening the incentive for delayed bursting.

Relation to Flow Matching and training stability. The target distribution is *not* learned: as in Flow Matching, μ_1 is fixed and represented by data samples. The difference is that Flow Matching prescribes the complete probability path and regresses its velocity, whereas Y-Flows leave the intermediate path unconstrained and impose the endpoint condition through a penalty (Sinkhorn or a learned dual potential, Appendix F); the endpoint discrepancy decreases as the penalty weight increases, subject to the usual approximation and compactness assumptions. Training is therefore more expensive per iteration than Flow Matching (ODE rollout, Jacobian estimate, endpoint loss; see Table 10), which is compensated by substantially fewer inference steps (Appendix I). Multi-seed variability and hyperparameter sensitivity are reported in Tables 6–8.



Figure 7: Image generation. Each row depicts a result of a generative model predicting a sample from the target distribution μ_1. Samples from the source distribution μ_0 are fixed and are the same for all models.

When should a Y-shaped flow be preferred? The Y-shaped inductive bias is most appropriate when (i) the source distribution is sufficiently concentrated or connected for coherent initial motion to be plausible, (ii) the target contains distinct terminal groups or an underlying hierarchy, and (iii) the application benefits from few-step or adaptive inference. These conditions are natural in generative modeling with a Gaussian source and a multimodal target. For targets without a meaningful grouped or hierarchical structure, the cohesion bias may provide little benefit or delay separation unnecessarily; the cohesion weight λ should then be reduced. Empirically (Appendix I), the advantage of Y-Flows lies in the low-step regime: they reach near-converged quality within a few steps, while Flow Matching attains a slightly lower FD at its converged operating point on FFHQ latents.

H Additional Experiments and Illustrations

H.1 LiDAR

Airborne LiDAR (Liu et al., 2023) terrain tile with ground and low-vegetation; each sample is a 3D point (x, y, z) with optional attributes (intensity, return number, total returns, scan angle, class). After filtering non-ground classes, removing outliers with a radius/k-NN check, and Poisson-disk (or voxel-grid) subsampling, we retain 5k points that preserve ridges, slopes, and basins. Coordinates are centered and scaled to a unit box (optionally PCA-aligned). We estimate per-point normals via local PCA and build a k-NN surface graph ($k \in [8, 16]$) with Euclidean edge weights as a geodesic proxy; during optimization, states are projected back to the nearest tangent patch to stay on-manifold. A single source distribution sits on a lower slope; four target clouds are disjoint regions on ridges/basins (indices provided for reproducibility).

H.2 Image Data

The implementation of *Flow Matching* was based on:
`facebookresearch/flow_matching`.

The implementation *Mean Flows* was based on:
`Gsunshine/py-meanflow`.

The experiment on *Domain Translation* was inspired by:
milenagazdieva/LightUnbalancedOptimalTransport.

FFHQ *encoded dataset and decoder* are taken from:
podgorskiy/ ALAE

I Extended Experiments: Step Curves, Ablations, and Raw-Pixel Benchmarks

This appendix extends the evaluation of Section 8 with (i) complete Fréchet-distance-versus-steps curves over multiple seeds, (ii) ablations of α, the cohesion weight, the endpoint objective, the well-posedness remedies of Appendix G and the Jacobian estimator, (iii) a direct quantitative branching metric, and (iv) experiments in raw pixel space on MNIST (784-D) and CelebA-64 (12,288-D), without any autoencoder or latent representation. YF-S denotes Y-Flows with the Sinkhorn endpoint loss (Section 5); YF-KD denotes the Kantorovich-dual variant of Appendix F. All methods share the same architecture, training budget and inference stopping criterion. Tables 4–6 and Table 11 report mean $\pm$ standard deviation over three random seeds; the ablation tables are single-seed. We write λ_{coh} for the cohesion weight λ of (7) when it appears next to other weights.

I.1 FFHQ Latents: Fréchet Distance versus Integration Steps

Table 4 reports FD in the ALAE latent space (female $\to$ male translation) from 1 to 128 integration steps together with the best observed value per method. YF-KD is strongest in the low-step regime: after only two to four steps its FD is roughly 3–4$\times$ lower than Mean Flows and about 4$\times$ lower than Flow Matching at the same budget (Table 5). Mean Flows plateaus at a substantially higher FD than both YF-KD and FM. At their respective best operating points, YF-KD reaches 5.22 ± 0.22 and FM 4.56 ± 0.41: Y-Flows deliver their gains as inference efficiency, reaching near-converged quality within a few steps, while Flow Matching attains a slightly lower FD once converged.

Table 4: FFHQ latents: Fréchet distance (↓) as a function of the number of integration steps. Mean $\pm$ std over three seeds. Best per column in **bold**.

Method	1	2	4	8	16	32	64	128	Best
FM	35.74±4.15	26.33±3.46	20.57±1.66	21.13±0.53	14.90±0.15	**4.56±0.41**	12.76±0.58	**4.97±0.68**	**4.56±0.41**
MF	35.24±10.9	23.56±1.52	16.98±1.75	13.43±2.04	11.74±2.13	11.05±2.11	10.77±2.08	10.66±2.05	10.66±2.05
YF-S	**35.23±13.7**	23.45±1.40	20.21±1.39	20.82±0.77	19.58±0.67	19.04±0.64	18.78±0.63	16.64±0.63	12.64±0.63
YF-KD	38.71±16.4	**6.54±1.19**	**5.36±0.28**	**5.37±0.42**	**5.39±0.49**	5.41±0.52	**5.42±0.54**	5.43±0.55	5.22±0.22

Table 5: FFHQ latents: representative matched inference budgets (FD, ↓).

Inference budget	FM	MF	YF-KD	Comparison
2 steps	26.33	23.56	**6.54**	YF-KD is 3.6$\times$ lower than MF and 4.0$\times$ lower than FM
4 steps	20.57	16.98	**5.36**	YF-KD is 3.2$\times$ lower than MF and 3.8$\times$ lower than FM
8 steps	21.13	13.43	**5.37**	YF-KD is 2.5$\times$ lower than MF and 3.9$\times$ lower than FM
Best observed	**4.56**	10.66	5.22	YF-KD is 2.0$\times$ lower than MF; FM is 1.1$\times$ lower than YF-KD

Seed variability. Table 6 lists the converged FD per seed. Mean Flows is the least reproducible method in this comparison; YF-KD has the smallest standard deviation, although three seeds are insufficient to establish strong statistical significance.

Table 6: FFHQ latents: converged FD over three seeds.

Method	Seed 0	Seed 1	Seed 2	Mean $\pm$ SD	Max/min spread
FM	5.02	4.27	4.38	4.56 ± 0.41	1.18$\times$
MF	9.60	13.02	9.35	10.66 ± 2.05	1.39$\times$
YF-KD	5.02	5.45	5.20	$\mathbf{5.22 \pm 0.22}$	1.09$\times$

I.2 FFHQ LATENTS: ABLATIONS

Branching exponent and cohesion weight. Table 7 sweeps α and λ_{coh} and reports the best FD over the inference-step sweep. Table 8 shows the step dependence of representative configurations and reveals two regimes: configurations without the cohesion term ($\lambda = 0$) encode strongly front-loaded trajectories but are not effective at extremely low integration budgets, whereas configurations with $\alpha = 1$ perform well immediately and remain nearly constant as the number of steps increases. Flow Matching, in contrast, needs many more steps to reach its best value (Table 4).

Table 7: FFHQ latents: best FD over the inference-step sweep for different α (rows) and cohesion weights λ_{coh} (columns). For reference, FM obtains 4.56 ± 0.41 and MF 10.66 ± 2.05.

$\alpha \setminus \lambda_{\text{coh}}$	10^{-4}	10^{-3}	10^{-2}	10^{-1}
0.3	5.30	5.36	5.69	4.94
0.5	5.04	5.30	5.31	4.74
0.7	4.70	5.11	5.11	4.64
0.9	5.16	4.52	4.89	4.41
1.0	4.31	4.41	4.53	4.53

Table 8: FFHQ latents: FD versus number of inference steps for representative configurations.

Configuration	1 step	2 steps	4 steps	8 steps	Best
$\alpha = 0.7, \lambda = 0$	1607	201.7	10.31	**4.70**	4.70
$\alpha = 0.5, \lambda = 0$	1131	111.4	6.19	**5.04**	5.04
$\alpha = 1, \lambda = 0.01$	7.63	**4.31**	4.31	4.34	4.31
$\alpha = 1, \lambda = 0.1$	4.39	4.24	**4.23**	4.23	4.23

Well-posedness remedies. Appendix G lists standard remedies against the potential "infinite-speed" degeneracy of the sublinear cost. Table 9 evaluates them on FFHQ; at inference we measure the velocity distribution over all particles and all 16 integration steps. The unconstrained model does not exhibit an infinite-speed collapse in practice; hard speed bounds substantially reduce extreme velocities and provide an effective safeguard when stronger control is required.

Table 9: FFHQ well-posedness ablation ($\alpha = 0.7$, $\lambda_{\text{coh}} = 0.01$, 60,000 iterations; early-stopping threshold $\delta = 10^{-3}$ in all rows). η is the smoothing parameter of the surrogate $(\|v\|^2 + \eta^2)^{\alpha/2}$.

Remedy	Best FD	v_{mean}	$v_{\max}/v_{\text{mean}}$
None	5.11	12.9	12.3
Smooth surrogate ($\eta = 0.1$)	**4.81**	12.8	14.8
Hard bound ($v_{\max} = 20$)	5.44	13.6	4.0
Hard bound ($v_{\max} = 5$)	6.25	9.5	1.6

Jacobian estimator. The cohesion term requires $\|\nabla_x v_\theta\|_F^2$. Table 10 compares the exact Jacobian (512 vector–Jacobian products in the 512-D latent space) with Hutchinson estimators. A single Hutchinson probe is unbiased, has a measured mean relative error of only 0.44%, and gives an FD essentially identical to four probes; we therefore use one probe in practice.

Table 10: Cost and accuracy of Jacobian estimators for the cohesion term (FFHQ, $K = 10$ integration steps).

Estimator	Time per call	Per iteration ($K = 10$)	Relative error	FD
Exact (512 VJPs)	584.7 ms	5.8 s	reference	infeasible
Hutchinson, 1 probe	**2.0 ms**	**20 ms**	**0.44%**	5.02
Hutchinson, 4 probes	5.2 ms	52 ms	—	5.05
No cohesion term	0	0	—	4.98

I.3 A DIRECT BRANCHING METRIC

To quantify branching directly we define, for K integration steps, the *front-load*

$$\text{front-load} = K \times \frac{\text{path length during the first step}}{\text{total path length}}, \tag{38}$$

so that a constant-speed flow has front-load 1. Table 11 reports it over 4,000 FFHQ trajectories with $K = 16$: Y-Flows have approximately $4.1\times$ the front-load of FM and a monotonically decreasing per-step path-length profile, whereas FM and MF are flat.

Table 11: Front-load metric on FFHQ latents ($K = 16$, 4,000 trajectories).

Method	Front-load	First 8 per-step path lengths	Profile
FM	0.91 ± 0.01	0.057, 0.060, 0.064, 0.065, 0.065, 0.064, 0.064, 0.063	flat
MF	1.02 ± 0.02	0.063, 0.061, 0.059, 0.059, 0.058, 0.058, 0.058, 0.059	flat
YF	$\mathbf{3.77 \pm 0.06}$	**0.231, 0.152, 0.108, 0.082, 0.066, 0.055, 0.047, 0.041**	monotonic decay

I.4 MNIST in Raw Pixel Space

We generate MNIST directly in the 784-dimensional pixel space without a pretrained autoencoder. All methods use the same 1.67M-parameter U-Net, EMA decay 0.999, a matched wall-clock training budget, raw-pixel FD, and the same inference stopping criterion; YF-KD uses the setup of Appendix F. Table 12 shows that Y-Flows reach near-converged quality with substantially fewer inference steps: at four steps the FD is 4.60 versus 18.65 for FM, and about 10 FM steps are needed to match the FD that Y-Flows reach in four. Y-Flows perform best between four and eight steps, while Flow Matching reaches its lowest FD at 16 steps.

Table 12: MNIST generation in raw pixel space (FD, ↓). K_{train} is the number of Euler steps used during training.

Model	K_{train}	FD@1	FD@2	FD@4	FD@8	FD@16
FM (identical backbone)	—	195.9	70.4	18.65	5.78	**3.23**
YF-KD	16	784.2	164.9	**4.60**	**4.30**	4.65

Ablation of α and λ. Table 13 varies the transport exponent and cohesion weight (same U-Net, exactly 12k velocity updates, FD on 5k generated vs. 5k held-out images; entries report the best FD and the number of steps at which it is attained). The best configuration ($\alpha = 0.5$) reaches FD 4.46 at four steps, versus approximately 16.5 for FM at the same budget under this fixed-update protocol (FM obtains 18.65 at four steps under the matched wall-clock protocol of Table 12). The sublinear exponent is essential: the linear control $\alpha = 1$ degrades FD from 4.61 ($\alpha = 0.7$) to 9.59 and removes the concentrated trajectory structure (Table 14): for the best configuration a single step accounts for 78% of the total path length and two steps cover 90% (k-90).

Table 13: MNIST ablation: best FD @ number of inference steps for different α (rows) and λ (columns).

$\alpha \setminus \lambda$	10^{-4}	10^{-3}	10^{-2}
0.3	4.76 @ 4	5.24 @ 4	31.95 @ 2
0.5	**4.46 @ 4**	5.10 @ 4	27.24 @ 1
0.7	4.61 @ 4	5.18 @ 4	21.46 @ 1
0.9	4.98 @ 4	6.92 @ 4	47.86 @ 1

Table 14: MNIST trajectory structure. Peak path share: fraction of the total path length covered in the single largest step; k-90: number of steps covering 90% of the path length; NFE/32: average number of function evaluations under the early-stopping criterion with a budget of 32.

Configuration	Best FD	Peak path share	k-90	NFE/32
Y-Flows, $\alpha = 0.5$	**4.46**	**0.780**	**2**	**5.3**
Y-Flows, $\alpha = 0.7$	4.61	0.591	2	7.0
Y-Flows, $\alpha = 1$	9.59	0.157	7	32.0
Flow Matching	≈ 3.4	≈ 0.13	8	32.0

I.5 Scalable Few-Step Generation on CelebA-64

Finally, we evaluate Y-Flows on CelebA at 64×64 resolution directly in RGB pixel space (12,288 dimensions), without an autoencoder or latent representation. The source is a deliberately displaced

Gaussian ($s = 414$, $\sigma = 0.534$), i.e., a compact initial distribution far from the image manifold. The backbone is a DDPM-style U-Net with channel multipliers $(1, 2, 2, 4)$, two residual blocks per resolution, self-attention at 16×16 and 8×8, and a 512-dimensional sinusoidal time embedding (21.4M parameters). The objective combines the sublinear transport cost with $\alpha = 0.5$ and the Kantorovich-dual endpoint constraint of Appendix F with a ResNet-50-style critic (10.77M parameters); the critic is unnecessary at inference. Training used four A100 GPUs, batch size 128, AdamW, EMA weights, generator/critic learning rates of $5 \times 10^{-5}/10^{-4}$, gradient clipping at 10, and Euler rollouts with $K = 8$ or $K = 10$ training steps. We report FID from 5,000 generated and 5,000 real images against a 45.1M-parameter Flow Matching reference; the Y-Flows generator is deliberately about $2\times$ smaller to compensate for the critic.

Table 15 shows that Y-Flows improve FID at every evaluated budget: FID decreases from 179.7 to 75.2 at two steps and from 86.3 to 37.9 at four steps. As the number of steps increases the gap narrows, indicating that the method primarily accelerates convergence rather than altering the final attainable quality. The $K = 8$ configuration reaches FID 12.3 in only eight network evaluations, slightly better than the reference at 32 evaluations, i.e., comparable or better quality with approximately four times fewer inference steps and a smaller network; this configuration is, however, less stable (FID 22.3 at seven steps, 12.3 at eight, 27.0 at nine), revealing specialization to the training discretization. Table 16 reports the number of steps needed to reach a target FID: the smoother $K = 10$ model needs approximately 1.3–$2.4\times$ fewer evaluations across practical quality levels. Figure 9 shows samples of the $K = 8$ model at increasing inference budgets from identical noise; the faces are already recognizable after four steps and stabilize after eight. Figure 8 visualizes the learned transport: trajectories leave the compact source as a single bundle, travel together along a shared trunk, and fan out only close to the image manifold, i.e., the Y-shaped geometry of Definition 1 persists in 12,288 dimensions.

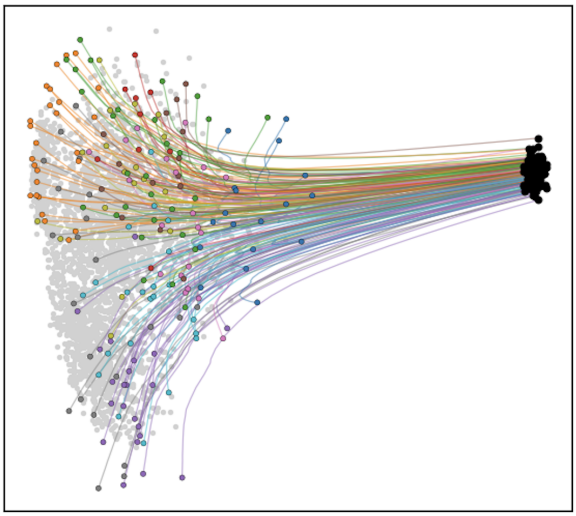


Figure 8: Y-shaped transport on CelebA-64 in raw pixel space ($\alpha = 0.7$, $\lambda = 0.1$), two-dimensional projection. Black: samples from the compact displaced-Gaussian source; gray: CelebA images; colored curves: individual trajectories with their endpoints marked. The trajectories move together along a shared trunk and separate only near the data manifold.

Table 15: CelebA-64 in raw pixel space: FID (↓) at fixed inference budgets.

Model	Params	FID@2	FID@4	FID@8	FID@16	FID@32
Flow Matching	45.1M	179.7	86.3	42.4	21.2	12.8
Y-Flows, $K = 10$	21.4M	**75.2**	**37.9**	22.8	16.1	**12.6**
improvement over FM	—	2.4×	2.3×	1.9×	1.3×	1.0×
Y-Flows, $K = 8$	21.4M	116.0	39.4	**12.3**	**14.8**	12.8
improvement over FM	—	1.5×	2.2×	3.4×	1.4×	1.0×

Table 16: CelebA-64: number of inference steps required to reach a target FID (linear interpolation of the step curves).

Target FID	FM steps	$K = 10$ steps	Reduction	$K = 8$ steps	Reduction
40.0	8.5	3.8	$2.27\times$	4.0	$2.15\times$
30.0	11.2	4.8	$2.32\times$	4.6	$2.44\times$
20.0	17.7	9.6	$1.85\times$	7.2	$2.45\times$
15.0	25.2	19.9	$1.27\times$	7.7	$3.26\times$
13.5	29.8	26.0	$1.15\times$	7.9	$3.79\times$

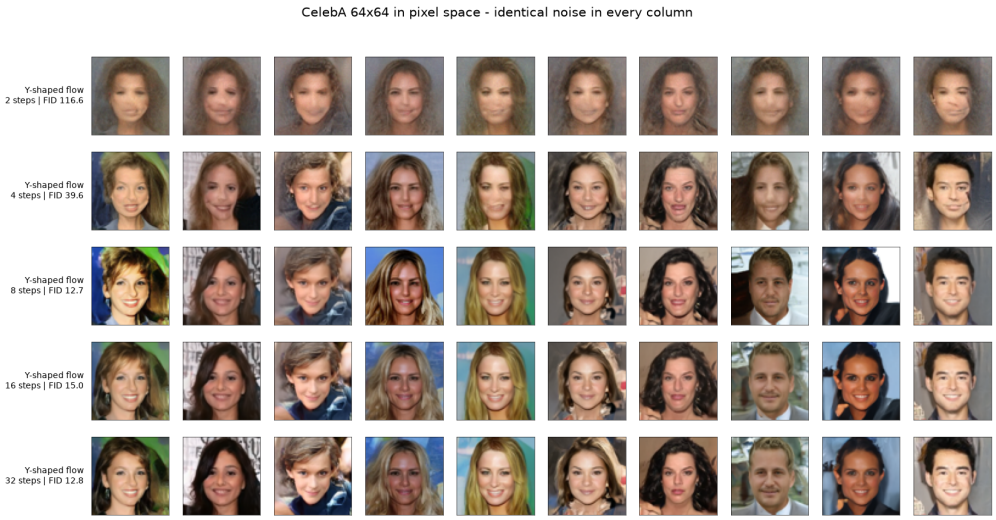


Figure 9: CelebA-64 samples generated in raw pixel space by Y-Flows ($K = 8$) with 2, 4, 8, 16 and 32 inference steps (rows); each column starts from the same noise. FID values for each budget are annotated on the left.