BRANCHED OPTIMAL TRANSPORT AMORTIZATION

Anonymous authors
Paper under double-blind review

ABSTRACT

Methods of Branched Optimal Transport (BOT) mimic the economy and efficiency of natural tree-like structures, such as those found in rivers and biological systems. These methods are widely applicable for designing efficient networks in society, from river basins and blood vessels to mail and gas distribution systems. However, they remain understudied in the context of designing deep generative models, particularly at a large scale. Standard continuous-time generative models, such as the flow matching approach, fail to capture the inherent hierarchical and branching patterns present in real-world data. Current models provide no mechanism for flows to merge or share pathways to minimize total transport cost. Inspired by the "economy of scale" principle in BOT, we introduce a novel, scalable branched flow-matching algorithm designed to solve the branched optimal transport problem in high dimensions. Our method adapts the Benamou-Brenier continuous-time optimal transport formulation to learn branched generative flows. These flows allow probability mass to aggregate along common pathways before branching out to diverse targets. Parametrized by neural networks, our method effectively learns complex branched generative processes. We demonstrate its effectiveness on challenging high-dimensional tasks in biology and image generation.

1 INTRODUCTION

Deep generative modeling has advanced rapidly, with diffusion models and simulation-free flow–based methods (e.g., flow matching and rectified flows) now supporting state-of-the-art synthesis, fast sampling, and scalable training. These approaches learn vector fields (or stochastic dynamics) that transport a simple base distribution to complex data distributions, and many of them explicitly encourage *straight* transport trajectories between coupled samples—mirroring the *displacement interpolation* paradigm in classical optimal transport (OT) (Ho et al., 2020; Lipman et al., 2022; Tong et al., 2023; Liu et al., 2022).

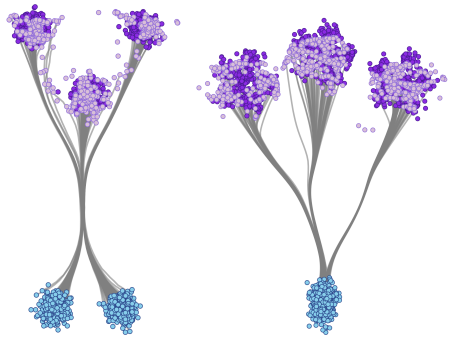


Figure 1: BOTA Gaussian results for $\alpha = 0.5$.

However, straight-line, independent motions of mass are often a poor fit for data with hierarchical or multimodal organization. ImageNet, for example, is built on the WordNet taxonomy: “golden retriever” and “labrador” are siblings under “dog,” which is nested under broader synsets such as “canine” and “mammal.” It is natural to expect that the generative process should *share substantial computation* along the path common to “dogs” before specializing to particular breeds. Such structures arise even more naturally and meaningfully in biological processes, which can also be observed as empirical distributions. Modeling such shared structure is difficult.

Branched optimal transport generalizes classical OT by introducing economies of scale: moving two units of mass together along the same route is cheaper than moving them separately. Formally, BOT minimizes energies in which the cost along an edge is *subadditive* in the mass (often proportional to m^α with $0 < \alpha < 1$), which causes optimal solutions to *coalesce* into **tree- or network-like structures** rather than independent straight paths. This mechanism closely matches how efficient networks form in nature and society, from river basins to mail and gas distribution systems (Buttazzo & Stepanov, 2003; Xia, 2003; Bernot et al., 2005).

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.

The BOT perspective is powerful for generative modeling because it encodes **shared prefixes of computation**: samples that belong to nearby leaves in a taxonomy naturally travel together for most of the journey, branching only when necessary. In contrast with Euclidean quadratic-cost OT—where displacement interpolation moves particles independently along straight segments defined by a Brenier map—BOT explicitly rewards branching, aligning the transport with hierarchical data.

Mathematically, BOT has a mature foundation: it has been studied via discrete "irrigation" models, continuum formulations, and relaxations (Bernot et al., 2009). Beyond the modeling appeal, BOT solutions satisfy rich structural properties (e.g., optimal junctions have bounded degree), reflecting the network nature of the minimizers (Santambrogio, 2015; Bernot et al., 2009; Oudet & Santambrogio, 2011). Despite its appeal, **BOT is algorithmically challenging**. Even in planar settings with finitely many sources/sinks, optimizing BOT networks is *NP-hard*, and require solving large PDE systems, which becomes prohibitive at modern data scales. This has kept BOT largely confined to small or low-dimensional instances.

At the same time, flow matching and rectified-flow models are trained at web scale and benefit from transport formulations that are differentiable end-to-end. A **scalable BOT solver** would let us *replace independent straight-path assumptions* by *economy-of-scale transport* that respects data hierarchies within practical training pipelines.

Neural parameterizations are a natural fit. In standard OT, neural solvers already amortize transport maps or potentials and integrate cleanly with deep learning. Extending this spirit to BOT promises (i) *amortized inference* of branched routes that can be reused across batches and tasks, (ii) *scalability* to high-dimensional data without pairwise couplings, and (iii) *generalization* to unseen samples through a single learned vector field.

Contribution: We introduce **Branched Optimal Transport Amortization (BOTA)**, a continuous and scalable solver for branched optimal transport. Our core contribution is a *novel optimization problem* that we adapt into a generative algorithm, which is efficiently parameterized by neural networks using a flow-matching objective. We demonstrate the effectiveness of our method on challenging biological and image generation tasks, showing that it successfully learns meaningful hierarchical generative processes that reflect the underlying structure of the data.

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., $d\mu(x) = d\rho(x)dx$). Vectors are column vectors, $\|\cdot\|$ is the standard Euclidean norm, and $\nabla\cdot$ is the divergence operator and $\|\cdot\|_F$ is the Frobenius norm.

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):

$$\frac{d}{dt}x_t = v_\theta(x_t, t), \qquad x_{t=0} \sim \mu_0. \tag{1}$$

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 the *continuity equation* (2) with the parameterized velocity field v_θ. Where continuity equation encodes mass conservation:

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

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

$$\frac{d}{dt} \log \rho_t(x_t) = -\nabla \cdot v_\theta(x_t, t). \tag{2}$$

This allows for a likelihood calculation by integrating this quantity over time. Training can be done by directly maximizing likelihood (integrating (2)).

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 toward 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:

$$u_t(x) = \mathbb{E}_{z\sim q}\left[\frac{\rho_t(x \mid z)}{\rho_t(x)}\, u_t(x \mid z)\right], \tag{3}$$

where $q(z \mid x)$ is the posterior. The key theorem (Lipman et al., 2022) is to minimize the *Conditional Flow Matching*:

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t,\, z\sim q,\, x\sim\rho_t(\cdot|z)}\left[\|v_\theta(x,t) - u_t(x \mid z)\|^2\right] \tag{4}$$

with $t \sim \mathcal{U}[0,1]$ yields the same gradient for θ as minimizing the intractable $\mathcal{L}_{\text{FM}}(\theta)$ in (4). 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 $u_t(x \mid z) = x_1 - x_0$.

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) $\gamma \in \Gamma(\mu_0, \mu_1)$ with marginals μ_0 and μ_1:

$$\inf_{\gamma\in\Gamma(\mu_0,\mu_1)} \int_{\Omega\times\Omega} c(x,y)\, d\gamma(x,y). \tag{5}$$

For cost $c(x, y) = \|x - y\|^2$, the square root of the solution is the Wasserstein-2 distance. The flow matching approach discussed in the section above can be connected with the flow matching to create 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 γ between μ_0 and μ_1 5. In practice, the OT plan γ is efficiently approximated using mini-batch OT by Sinkhorn algorithm (Cuturi, 2013).

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 dynamics that satisfies *continuity equation* 2 with the velocity field v. The Benamou–Brenier theorem states that the squared Wasserstein distance equals the minimal *kinetic energy* of such a flow:

$$\inf_{\rho_t,\, v_t} \left\{ \int_0^1 \int_\Omega \|v_t(x)\|^2 \rho_t(x)\, dx\, dt \quad \text{subject to (2)} \right\}. \tag{6}$$

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, $v_t(x_t) = T(x_0) - x_0$, and the particle dynamics is linear: $x_t = (1-t)x_0 + tT(x_0)$.

3 Branched Benamou–Brenier OT

Branched Benamou–Brenier formulation studied by Brasco et al. (2011), which is also equivalent to the formulation of Xia (2003). The idea is to consider the classical Benamou–Brenier optimal transport problem 6, but with a *concave* cost function. The boundary conditions remain the same as in standard optimal transport: $\rho(0, \cdot) = \mu_0$ and $\rho(1, \cdot) = \mu_1$. With $\alpha \in (0, 1)$ as the branching parameter, we define the cost that measures the *work* of moving a mass m over a distance ℓ, simply

written as $m^\alpha \ell$. If $\alpha = 1$, the cost becomes linear in the mass ($m^1 \ell$), recovering the classical OT setting 2. By identifying the length ℓ with the velocity v and the mass m with the time-dependent probability density ρ_t, the dynamic branched optimal transport can be formulated as:

$$B_\alpha(\mu_0, \mu_1) \;=\; \inf_{\rho(t,\cdot),\, v(t,\cdot)} \int_0^1 \sum_{i \in I_t} |v_{t,i}| \, (\rho_{t,i})^\alpha \, dt. \tag{7}$$

For the cost (7) to be finite, the measure $\rho(t, \cdot)$ must be *purely atomic* for almost every time t, i.e., the mass is concentrated at points $\{x_{t,i}\}_{i \in I_t}$:

$$\rho(t, \cdot) = \sum_{i \in I_t} \rho_{t,i} \, \delta_{x_{t,i}}, \quad \text{where } \rho_{t,i} = \rho(t, \{x_{t,i}\}).$$

The associated momentum is then $q = \rho v = \sum_{i \in I_t} v_{t,i} \, \rho_{t,i} \, \delta_{x_{t,i}}$, where $v_{t,i} = v(t, x_{t,i})$ is the velocity of the atom $x_{t,i}$. This formulation is computationally intensive, which motivates our development of a more efficient relaxation.

4 Branched Benamou Brenier Neural OT

To build a Benamou Brenier style Branced OT flows we need to develop a flow algorithms that meet a few constraints:

- The boundary constraint: we need a flow whose starting and end points are equal to the given distributions: $\rho(0, \cdot) = \mu_0$ $\rho(1, \cdot) = \mu_1$.
- Mass-dependent cost: We need to have a time-dependent density function that we are using as part of the cost function.
- Atomic mass constraint: Our density function must be atomic almost every time t, so the mass needs to be concentrated in a countable set of points.

To meet these constraints, it is convenient to draw an analogy between branched optimal transport and continuous normalizing flows. Having a continued vector field, we can derive a time continuous probability density function that plays a central role in the formulation of branched optimal transport. Also, CNFs by design meet the the boundary constraint 2. So, first, let us parameterize the velocity field by a continuous normalizing flow.

$$\dot{x}_t = v_\theta\big(x_t, t\big), \qquad t \in [t_0, t_1],$$

whose density $p_\theta(x, t)$ evolves under the instantaneous change–of–variables, CNF formula:

$$\log p_\theta(x_{t_1}, t_1) = \log p_\theta(x_{t_0}, t_0) - \int_{t_0}^{t_1} \operatorname{tr}\big(\partial_x v_\theta(x_t, t)\big) \, dt. \tag{8}$$

This enforces the continuity equation by construction, making CNFs natural for dynamic OT objectives. To capture branched transport, we concavify the kinetic action by the density, mirroring the BOT cost $m^\alpha \ell$ with $\alpha \in (0, 1)$. Our learning objective over terminal samples is

$$\mathcal{L}^\alpha(\theta) = \int_{t_0}^{t_1} \int_\Omega \underbrace{\big\|v_\theta(x,t)\big\| p_\theta(x,t)^\alpha \, dx \, dt}_{\text{branched cost}} - \lambda \underbrace{\mathbb{E}_{x_{t_1} \sim \mu_1}\big[\log p_\theta(x_{t_1}, t_1)\big]}_{\text{match data at } t_1}. \tag{9}$$

The second term is the CNF maximum-likelihood fit to μ_1 at t_1. The first term is a BOT-style dynamic action: it penalizes kinetic effort $\|v_\theta\|$ but discounts it by p_θ^α with $\alpha < 1$. Our method is a concave reweighting that implements the BOT principle *move together, then split*, while keeping the CNF machinery and training recipe.

A challenge in implementing the BOT objective (9) within a CNF framework is the *atomic mass constraint*. The theoretical BOT formulation requires the evolving measure $\rho(t, \cdot)$ to be a sum of discrete point masses for the cost to be finite. However, CNFs naturally define a *continuous* density field $p_\theta(x, t)$, making these two concepts fundamentally incompatible for optimization.

To bridge this gap, we introduce a differentiable relaxation of the atomic constraint, which we term the *soft-atomic cost*. In a computational setting, we cannot handle continuous measures or an infinite number of atoms. We make two approximations. Instead of forcing the continuous density to collapse into discrete points, this objective encourages the density to *cluster* around a predefined set of K proxy atoms. This is achieved by softly assigning the mass of particles from the flow to these centers based on proximity.

At any given time t, for a batch of particles x_t sampled from the flow density $p_\theta(\cdot, t)$, we define the soft mass $m_k(t)$ and the average velocity $v_k(t)$ associated with center c_k and where $w_k(x)$ is a soft-assignment weight, typically a Gaussian kernel that measures the influence of center c_k on a particle at position x:

$$w_k(x) = \frac{\exp(-\frac{\|x-c_k\|^2}{2\sigma^2})}{\sum_{j=1}^{K} \exp(-\frac{\|x-c_j\|^2}{2\sigma^2})}, \quad m_k(t) = \mathbb{E}_{x_t \sim p_\theta(\cdot,t)} \left[w_k(x_t) \right], \tag{10}$$

$$v_k(t) = \frac{\mathbb{E}_{x_t \sim p_\theta(\cdot,t)} \left[v_\theta(x_t, t) w_k(x_t) \right]}{m_k(t)}. \tag{11}$$

The instantaneous soft-atomic cost is then formulated as a differentiable proxy to the theoretical BOT cost. By integrating soft cost over time, we reformulate the learning objective as:

$$\mathcal{L}^\alpha(\theta) = \int_{t_0}^{t_1} \sum_{k=1}^{K} \underbrace{v_k(t) m_k(t)^\alpha dt}_{\text{soft branched cost}} - \lambda \underbrace{\mathbb{E}_{x_{t_1} \sim \mu_1} \big[\log p_\theta(x_{t_1}, t_1) \big]}_{\text{match data at } t_1}. \tag{12}$$

Lemma 1 (Differentiable Approximation of the Instantaneous Branched Transport Cost) *Let $\hat{\rho}_t = \frac{1}{B} \sum_{j=1}^{B} \delta_{x_j(t)}$ be an empirical probability measure representing a distribution at time t with particles at positions $\{x_j\}_{j=1}^{B}$ moving with velocities $\{v_j\}_{j=1}^{B}$. Let $\{c_k\}_{k=1}^{K}$ be a set of fixed centers in $\mathbb{R}^d$. The function soft atomic cost, defined as:*

$$C_{soft}(\{x_j\}, \{v_j\}) = \sum_{k=1}^{K} |v_k| m_k^\alpha, \tag{13}$$

where for each center c_k:

$$m_k = \frac{1}{B} \sum_{j=1}^{B} w_{jk} \quad (\textit{Soft Mass}) \qquad |v_k| = \frac{\sum_{j=1}^{B} w_{jk} |v_j|}{\sum_{j=1}^{B} w_{jk}} \quad (\textit{Soft Average Speed})$$

with weights $w_{jk} = softmax(-||x_j - c_k||^2/2\sigma^2)_k$, is a differentiable approximation of the true instantaneous branched transport cost functional for the discretized measure. In the limit as the kernel width $\sigma \to 0$, this cost converges to the cost of a hard assignment of particles to their nearest centers. Please see proofs in Appendix.

5 Branched Flow Matching via BOT Amortization

Directly optimizing the branched objective (9) over continuous-time neural velocity fields is hard: the objective is non-convex and the atomic constraint is inherently discrete. We therefore introduce a practical two-stage method, Branched Optimal Transport Amortization (BOTA), also called **Branched Flow Matching**: it first solves a discrete-time version of the soft-atomic relaxation for a fixed batch, then amortizes this solution into a continuous-time neural network with a flow-matching objective. This mirrors OT-CFM (4), with the straight-line optimal transport plan of Sec. 2 replaced by a branched one.

5.1 Discrete Branched Transport Solver

First, we compute an exemplar branched flow for a single, fixed batch of N source samples $\{x_0^i\}_{i=1}^{N} \sim \mu_0$ and N target samples $\{x_1^i\}_{i=1}^{N} \sim \mu_1$. We discretize the time interval $[0, 1]$ into T steps of size $\Delta t = 1/T$. Our goal is to find an optimal discrete velocity field $V \in \mathbb{R}^{T \times N \times D}$ that transports the

batch $\{x_0^i\}$ to $\{x_1^i\}$ while minimizing a discrete analogue of the branched transport cost. The particle trajectories $\{X_k\}_{k=0}^T$, where $X_k = \{x_k^i\}_{i=1}^N$, are computed via standard Euler integration:

$$x_{k+1}^i = x_k^i + V_{k,i} \cdot \Delta t, \quad \text{with } x_0^i \text{ given.}$$

Here, $V_{k,i} \in \mathbb{R}^D$ is the velocity of particle i at discrete time step k.

We optimize V by minimizing a loss function that combines a terminal matching cost with the time-integrated soft-atomic energy:

$$\mathcal{L}_{\text{discrete}}(V) = \sum_{k=0}^{T-1} C_{\text{soft}}(X_k, V_k)\Delta t + \lambda \frac{1}{N} \sum_{i=1}^{N} \|x_T^i - x_1^i\|^2.$$

The second term is a standard mean-squared error that forces the final positions x_T^i to match the target positions x_1^i. The first term is the discrete integral of the instantaneous branched transport cost, C_{soft}, which is defined precisely in Lemma 1, Eq. (13), as:

$$C_{\text{soft}}(X_k, V_k) = \sum_{j=1}^{K} |v_j|_k (m_j)_k^\alpha,$$

where $(m_j)_k$ and $(|v_j|)_k$ are the soft mass and soft average speed, respectively, associated with the j-th proxy atom c_j at time step k.

Source–target pairing. The terminal term is a plain MSE between the i-th particle and the i-th target: the source and target batches are drawn independently and paired by index, with no precomputed OT coupling (which would cost $O(N^2 D)$ before the solver starts). The pairing does not shape the branched geometry. C_{soft} depends on the particles only through sums over the particle index, so it is invariant under any relabelling of particles; the pairing enters through the terminal term alone and decides which leaf each particle reaches, not where trunks form or when mass splits. Because the terminal term is soft ($\lambda = 20$ against unit energy weight in all experiments), the solver can trade terminal error for cheaper routing, which is what lets trunks form. Gradient descent on V yields $V^* \in \mathbb{R}^{T \times N \times D}$ and its trajectory $X^* \in \mathbb{R}^{(T+1) \times N \times D}$, a single discrete branched transport path for the batch. Proxy-atom placement, K, σ, and step counts are given in Appendix A.4.

5.2 Amortization via Flow Matching

The discrete solution (X^*, V^*) is only valid for the batch it was computed on. To obtain a generative model we "distill" it into a continuous-time neural velocity field $v_\theta(x, t)$ with a flow-matching objective, which turns the problem into standard supervised regression with (X^*, V^*) as ground truth. We construct a continuous-time target field $u_t(x)$ by linear interpolation: for $t \in [0, 1]$, let $k = \lfloor t/\Delta t \rfloor$ (clamped to $[0, T-1]$) and $\beta = (t - k\Delta t)/\Delta t \in [0, 1]$; for each particle i, the target position and velocity are

$$x_t^i = (1-\beta)x_k^{*,i} + \beta x_{k+1}^{*,i}, \qquad u_t(x_t^i) = (1-\beta)V_{k,i}^* + \beta V_{k',i}^*,$$

where $k' = \min(k+1, T-1)$ is the next velocity index.

The Branched Flow Matching (BFM) objective is then to regress the neural network v_θ onto this continuous, interpolated target field u_t:

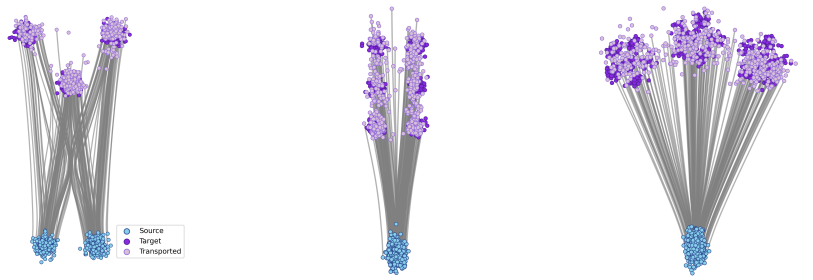
$$\mathcal{L}_{\text{BFM}}(\theta) = \mathbb{E}_{i \sim \mathcal{U}[1,N], t \sim \mathcal{U}[0,1]} \left[\left\| v_\theta(x_t^i, t) - u_t(x_t^i) \right\|^2 \right].$$

By minimizing this objective, the network v_θ learns to approximate the complex, branched velocity field derived from the discrete BOT solver. Once trained, v_θ can be used as a generative model to transport new samples from μ_0 to μ_1 by solving the continuous-time ODE $\dot{x} = v_\theta(x, t)$, effectively amortizing the expensive discrete optimization.

6 Experiments

6.1 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 M clusters ("branches") arranged at the top; the source



(a) 3 (Flow Matching) (b) 6 (Flow Matching) (c) 18 (Flow Matching)

(d) 3 (BOTA) (e) 6 (BOTA) (f) 18 (BOTA)

Figure 2: BOTA Gaussian mixture results for increasing number of Gaussians. Top row: Flow Matching. Bottom row: BOTA ($\alpha = 0.5$).

Table 1: Comparison of methods on the Tedsim dataset (50D PCA); mean $\pm$ std over 5 seeds.

Metric	BSBM	CNF	CFM	FM	BOTA
W_1	17.72 ± 0.12	13.76 ± 0.10	12.01 ± 0.05	12.03 ± 0.13	$\mathbf{11.86 \pm 0.07}$
W_2	17.96 ± 0.15	13.81 ± 0.11	12.12 ± 0.04	12.15 ± 0.16	$\mathbf{12.02 \pm 0.09}$
RBF-MMD	0.63 ± 0.007	0.51 ± 0.005	0.11 ± 0.002	0.15 ± 0.007	$\mathbf{0.11 \pm 0.003}$

is a cluster (or mix of two clusters) at the bottom. We train the same-size models with (i) standard FM and (ii) BOTA. Each curve shows a sample's trajectory from source (blue) to its destination cluster (purple). As M grows ($3 \to 6 \to 18$), FM learns many independent, straight fan-out paths that crowd and cross, offering little shared routing (Figure 2). BOTA instead discovers a shared "trunk" that later splits into branches, yielding short, structured, tree-like transport; the qualitative gap widens with more branches. Both models use the same time-conditioned MLP $v_\theta(x, t)$: 64-d time embedding $\to$ 3×256 SiLU layers $\to$ 2-D output; integrated with 10 fixed Euler steps over $t \in [0, 1]$ (TedSim below uses 50, identically for every method). 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. These panels are deliberately illustrative: they make the effect of the branched cost visible on a target whose geometry is known by construction, and the α sweep in Appendix Fig. 6 shows the trunk forming and disappearing with the exponent alone. Quantitative evaluation is carried by TedSim (Sec. 6.2).

6.2 Biological Data

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 simple, branching differentiation process.

We apply our method to a specific Tedsim scenario, modeling the transport from a single progenitor cell to two distinct terminal states, so that methods are compared on a dataset whose lineage is trunk-then-split by construction. Please see Figure 3 and Table 1. We use the same architecture for all models for a fair comparison.

Comparison baselines: Flow Matching (**FM**) (Lipman et al., 2022), Conditional Flow Matching (**CFM**) (Tong et al., 2023), Branched Schrödinger Bridge Matching (**BSBM**) (Tang et al., 2025) and **CNF** (Chen et al., 2018) are used as baselines in our work. All methods share the architecture, 100k iterations, learning rate 1e-4, Adam, the same integration scheme and the same 5 seeds; Table 1 reports mean $\pm$ std over seeds for every method and metric. BOTA is not proposed as a better density estimator; it solves a different objective, so the primary metric is the branched cost of Eq. (13) evaluated on each method's own trajectories (Appendix Fig. 8b): BOTA attains 174.81 ± 5.02 against 405.08 ± 1.00 for FM and 408.88 ± 1.00 for CFM, a $2.3\times$ reduction far outside the seed spread. FM and CFM differ only in how source and target samples are paired, yet land within 1% of each other, so the gap comes from the objective rather than from a better coupling. On W_1, W_2 and RBF-MMD, BOTA matches or exceeds the best baseline, so the branched structure costs nothing in terminal fit. Training takes $\approx 2\times$ the time of FM, paid once offline, with identical sampling cost (Appendix Table 3).

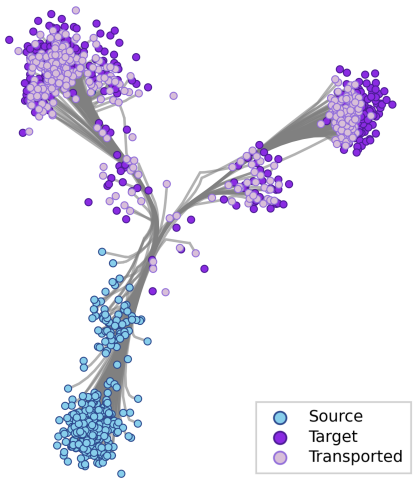


Figure 3: BOTA results on complicated biology Tedsim dataset with $\alpha = 0.5$.

6.3 Image Data

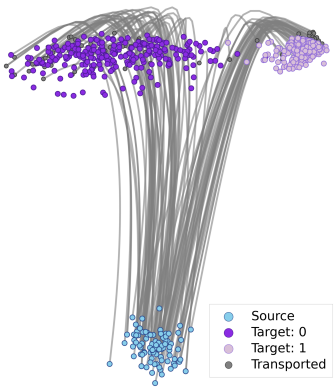


Figure 4: PCA of the learned flow for generating classes 0 and 1.

MNIST. As a proof of concept on image data we use MNIST (Deng, 2012) in two settings: generating the two classes 0 and 1 to examine branching, and generating the full dataset from a Gaussian. Figure 4 shows that our method reveals branching structure here as well. Hyperparameters are those of the biology experiment.

FFHQ. All experiments are conducted in the latent space of a pretrained, frozen ALAE model on FFHQ 1024×1024, used *only* to decode latents to images for evaluation. We load latent vectors paired with gender labels and split them into 60,000 training and 10,000 test samples. The goal is to show that our method produces a smooth and hierarchical interpolation between two classes. We learn a BOTA from source-class latents to target-class latents (e.g., female $\rightarrow$ male); mini-batches for x_0 and x_1 are sampled from the respective class-specific training subsets. We amortize a time-dependent vector field $v_\theta(x,t)$ that defines the ODE $\frac{d}{dt}x(t) = v_\theta\big(x(t),t\big)$ integrated from $t=0$ to $t=1$.

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; the output is D-dimensional. For the results please see Figure 5.

7 Related Work

Branched optimal transport has been formalized as traffic plans, irrigation flows, and branched networks. For a single source μ, Maddalena et al. (2003) and Bernot et al. (2005) model the network as a set of "fibers" $\chi(p,\cdot)$, where $\chi(p,t)$ is the location at time t of a particle p in an abstract probability space Ω. Such approaches explain how mass clusters around "branch nodes," but remain computationally intractable for high-dimensional data.

In parallel, the deep generative modeling community (Asadulaev et al., 2025; Tang et al., 2025) proposed Y-shaped and branched Generative Flows, a continuous-time flow framework that encourages branching of paths. However, these models do not explicitly solve the BOT optimization problem; they embed a BOT-inspired inductive bias in a single-stage network training, without a mass-dependent cost or a branching exponent, and do not guarantee a globally optimal branched transport plan for the given data.



Figure 5: Branched flow interpolation on FFHQ dataset From left to right over the timestamps. Female → Male task.

8 Limitations and Broader Impact

BOTA is designed for multi-modal targets reached from a common origin, where trajectory structure matters. We do not expect it to help for unimodal targets with no shared prefix, when source and target are already close, or when only endpoint quality is evaluated: on FFHQ its FID is comparable to FM, not better (Appendix Table 2). The soft-atomic relaxation depends on the proxy atoms, K and σ; too large a σ makes every weight $1/K$ and removes the branching signal, too small makes the weights one-hot and the gradients vanish (Lemma 1). Stage 1 is non-convex and upper-bounds what stage 2 can express: its local minima and relaxation error are inherited by the amortized field, not corrected. Finally, the branched cost of Eq. (13) is the criterion the model optimizes, not a certificate of lineage recovery; Table 1 measures terminal fit, and a direct test against TedSim's ancestor–descendant coupling is left to future work. This work may support applications involving structured branching processes such as cellular differentiation, but inherits standard generative-modeling risks, including misuse for synthetic data generation and amplification of dataset biases.

9 Conclusion

We introduce Branched Optimal Transport Amortization (BOTA), a scalable generative algorithm designed to overcome the inability of standard continuous-time models to capture hierarchical, tree-like data structures. By adapting the Benamou-Brenier formulation with a concave "economy of scale" cost function, BOTA encourages probability mass to aggregate along shared "trunks" before branching out to specific targets. To make this computationally feasible, the method utilizes a novel differentiable "soft-atomic" relaxation and amortizes the solution into a continuous neural vector field via flow matching. On TedSim, BOTA attains a $2.3\times$ lower branched cost than FM and CFM while matching the best baseline on terminal fit, at $\approx 2\times$ FM training cost and identical sampling cost; on FFHQ latents it yields hierarchical interpolations at no cost in FID.

AI use statement

We used generative AI tools only to edit the manuscript for grammar and readability, with all edits reviewed by the authors. They were not used for research ideas, hypotheses, theory or proofs, experimental design, code, data processing, or interpretation of results. We take responsibility for the final content of this work.

Reproducibility statement

Code for the solver, the soft-atomic cost, the amortization stage and all experiments is in the supplementary material; hyperparameters are in Appendix A.4 and the proof of Lemma 1 in Appendix A.

REFERENCES

Arip Asadulaev, Semyon Semenov, Abduragim Shtanchaev, Eric Moulines, Fakhri Karray, and Martin Takac. Y-shaped generative flows. *arXiv preprint arXiv:2510.11955*, 2025.

Marc Bernot, Vicent Caselles, and Jean-Michel Morel. Traffic plans. *Publicacions Matemàtiques*, 49 (2):417–451, 2005.

Marc Bernot, Vicent Caselles, and Jean-Michel Morel. *Optimal transportation networks: models and theory*. Springer, 2009.

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.

Li Deng. The mnist database of handwritten digit images for machine learning research [best of the web]. *IEEE signal processing magazine*, 29(6):141–142, 2012.

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

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

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

Francesco Maddalena, Giovanni Taglialatela, and Jean-Michel Morel. A variational model of irrigation patterns. *Interfaces and Free Boundaries*, 5(4):391–416, 2003.

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.

Filippo Santambrogio. *Optimal Transport for Applied Mathematicians*, volume 87 of *Progress in Nonlinear Differential Equations and Their Applications*. Birkhäuser, 2015. doi: 10.1007/978-3-319-20828-2.

Sophia Tang, Yinuo Zhang, Alexander Tong, and Pranam Chatterjee. Branched schr\" odinger 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 Extended Analysis

$$\mathcal{L}^{\alpha}(\theta) = \int_{t_0}^{t_1} \sum_{k=1}^{K} \underbrace{\|\bar{v}_k(t)\|\, m_k(t)^{\alpha}}_{\text{soft branched cost}}\, dt - \lambda\, \underbrace{\mathbb{E}_{x_{t_1}\sim\mu_1}\big[\log p_\theta(x_{t_1}, t_1)\big]}_{\text{match data at } t_1}. \tag{14}$$

Lemma 1 (Differentiable Approximation of the Instantaneous Branched Transport Cost). *Let $\hat{\rho}_t = \sum_{j=1}^{B} \rho_j\, \delta_{x_j(t)}$ be an empirical measure representing a particle approximation of a distribution at time t, with particle positions $\{x_j(t)\}_{j=1}^{B} \subset \mathbb{R}^d$ and velocities $\{v_j(t)\}_{j=1}^{B} \subset \mathbb{R}^d$. Let $\{c_k\}_{k=1}^{K}$ be a set of fixed centers. Define the* `soft_atomic_cost` *as*

$$C_{\textit{soft}}(\{x_j\}, \{v_j\}) = \sum_{k=1}^{K} \|\bar{v}_k\|\, m_k^{\alpha},$$

where for each center c_k,

$$m_k = \sum_{j=1}^{B} w_{jk}\, \rho_j \qquad (\textit{Soft Mass}),$$

$$\bar{v}_k = \frac{\sum_{j=1}^{B} w_{jk}\, v_j}{\sum_{j=1}^{B} w_{jk} + \varepsilon} \qquad (\textit{Soft Averaged Velocity}),$$

with smooth weights

$$w_{jk} = \frac{\exp\!\big(-\|x_j - c_k\|^2/2\sigma^2\big)}{\sum_{\ell=1}^{K} \exp(-\|x_j - c_\ell\|^2/2\sigma^2)}.$$

Then $C_{\textit{soft}}$ is differentiable in $\{x_j, v_j\}$ for any $\sigma > 0$, and as $\sigma \to 0$, it converges to the hard-assignment cost where each particle is assigned to its nearest center.

Proof. The instantaneous branched transport cost for a purely atomic measure $\rho_t = \sum_i \rho_{t,i}\, \delta_{x_i}$ (Brasco et al., 2011) is given by

$$\mathcal{F}(\rho_t) = \sum_i \|v_{t,i}\|\, \rho_{t,i}^{\alpha}.$$

To approximate this functional with a differentiable surrogate, we discretize space into K centers $\{c_k\}$ and replace the discontinuous Voronoi assignment with a soft weighting.

In the hard, non-differentiable case, the partition of space into Voronoi cells $\{V_k\}$ yields

$$m_k^{\text{hard}} = \sum_{j=1}^{B} \rho_j\, \mathbf{1}_{x_j \in V_k},$$

$$\bar{v}_k^{\text{hard}} = \frac{\sum_{j=1}^{B} \rho_j\, v_j\, \mathbf{1}_{x_j \in V_k}}{\sum_{j=1}^{B} \rho_j\, \mathbf{1}_{x_j \in V_k}}.$$

This construction is non-differentiable because the indicator $\mathbf{1}_{x_j \in V_k}$ is discontinuous.

We therefore replace it with smooth, normalized kernel weights w_{jk} as defined above. The Gaussian kernel ensures that w_{jk} is infinitely differentiable in x_j. As $\sigma \to 0$, the kernel becomes sharply peaked, so that

$$w_{jk} \;\to\; \mathbf{1}\Big[k = \arg\min_{\ell} \|x_j - c_\ell\|\Big].$$

Consequently,

$$m_k \;\to\; m_k^{\text{hard}}, \quad \bar{v}_k \;\to\; \bar{v}_k^{\text{hard}}.$$

Substituting these quantities into the definition of C_{soft} gives

$$C_{\text{soft}} = \sum_{k=1}^{K} \|\bar{v}_k\|\, m_k^{\alpha} = \sum_{k=1}^{K} \left\| \frac{\sum_j w_{jk} v_j}{\sum_j w_{jk}} \right\| \left(\sum_j w_{jk} \rho_j \right)^{\alpha}.$$

Each operation involved (sums, norms, powers, and exponentials) is differentiable for $\sigma > 0$ and $\varepsilon > 0$, ensuring that C_{soft} is a smooth, differentiable approximation of the branched transport cost.

In the limit $\sigma \to 0$, the soft assignments converge to hard nearest-center assignments, and hence

$$\lim_{\sigma \to 0} C_{\text{soft}} = \sum_{k=1}^{K} \|\bar{v}_k^{\text{hard}}\| \, (m_k^{\text{hard}})^{\alpha}.$$

If the centers coincide with the atom positions ($c_k = x_k$) and $K = B$, this expression recovers exactly

$$\mathcal{F}(\rho_t) = \sum_{i=1}^{B} \|v_i\| \, \rho_i^{\alpha}.$$

Thus, the proposed C_{soft} provides a differentiable surrogate that converges to the atomic branched transport cost in the small-σ limit. □

This differentiable reformulation replaces the hard, non-differentiable atomic constraint with a smooth objective that still promotes the desired clustering and branching behavior. It enables gradient-based learning of velocity fields that generate transport maps with the geometric structure characteristic of branched optimal transport, thereby providing a practical and effective objective for solving the branched optimal transport problem.

A.1 Gaussian Data

In this section, we further analyze the role of the branching parameter α in shaping transport geometry through a series of Gaussian ablations (Fig. 6). These experiments provide both qualitative and structural insight into how trajectory behavior evolves across regimes. For larger values of α, trajectories become increasingly independent, closely resembling classical optimal transport solutions. In contrast, smaller values of α encourage mass to follow shared pathways, giving rise to pronounced trunk-and-branch structures. This transition highlights how α governs the balance between independent transport and cooperative routing, ultimately controlling the geometric organization of mass flow.

A.2 Image Data

In this supplementary section, we extend the analysis presented in the main paper with additional qualitative and quantitative results on MNIST and FFHQ.

For MNIST, Figure 7 provides additional samples generated by BOTA. These results illustrate that the model produces clean, well-formed digits across classes, with consistent stroke thickness and minimal visual artifacts. This complements the main paper by showing that BOTA maintains stable trajectories even in discrete, multidimensional datasets, resulting in reliable class-conditional generations.

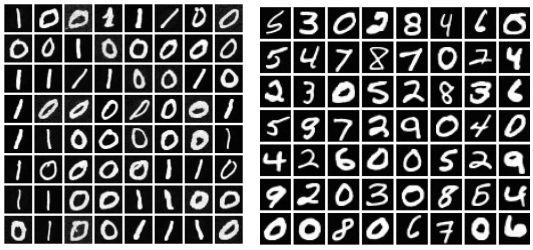
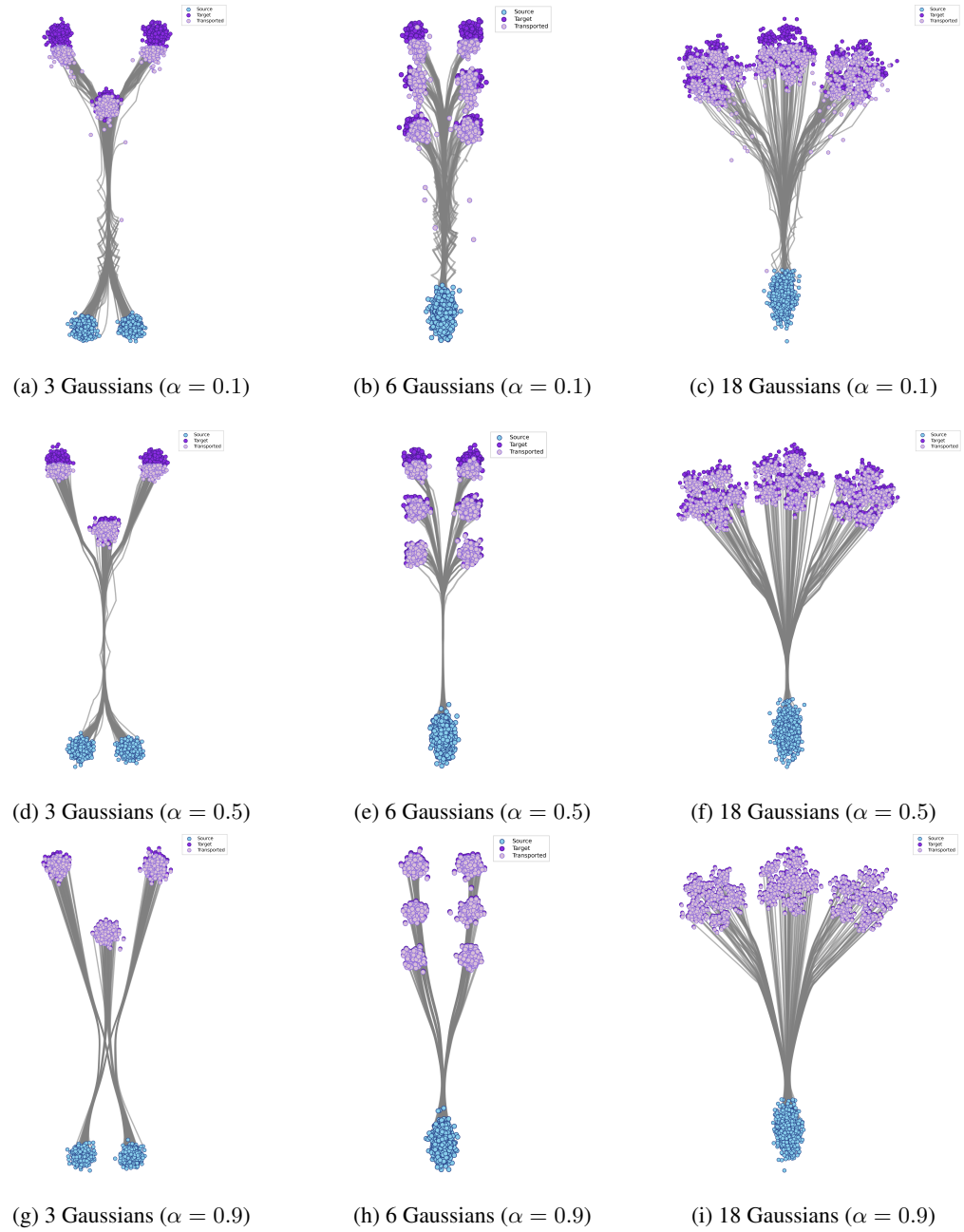


Figure 7: Examples of MNIST digits generated by BOTA.

Table 2: FID scores comparison on the FFHQ dataset.

Method	FID
FM	10.61 ± 1.39
BOTA	$\mathbf{10.46 \pm 1.80}$

For high-resolution natural images, Table 2 reports FID on FFHQ (mean $\pm$ std over seeds).
The two scores are the same within error (10.46 ± 1.80 vs. 10.61 ± 1.39; the gap is roughly one



(a) 3 Gaussians ($\alpha = 0.1$) (b) 6 Gaussians ($\alpha = 0.1$) (c) 18 Gaussians ($\alpha = 0.1$)

(d) 3 Gaussians ($\alpha = 0.5$) (e) 6 Gaussians ($\alpha = 0.5$) (f) 18 Gaussians ($\alpha = 0.5$)

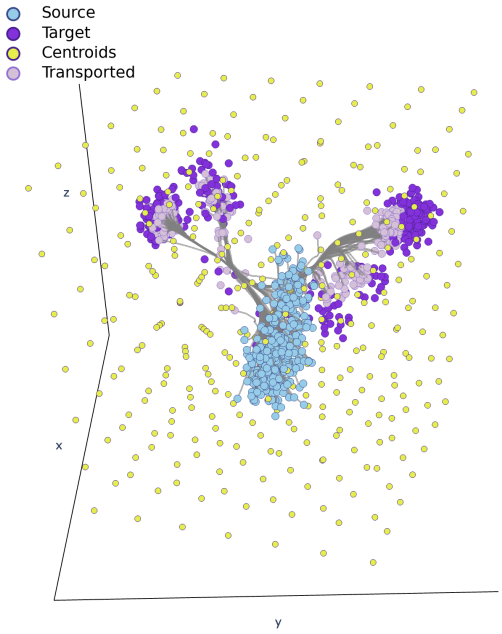
(g) 3 Gaussians ($\alpha = 0.9$) (h) 6 Gaussians ($\alpha = 0.9$) (i) 18 Gaussians ($\alpha = 0.9$)

Figure 6: BOTA Gaussian mixture results for increasing number of Gaussians (3, 6, 18) and different branching parameters α. Top row: $\alpha = 0.1$. Middle row: $\alpha = 0.5$. Bottom row: $\alpha = 0.9$.

tenth of the seed standard deviation), so we do not claim better image generation. The point of the experiment is that the branched, hierarchical interpolation of Fig. 5 is obtained without giving up sample quality.

A.3 Biological Data

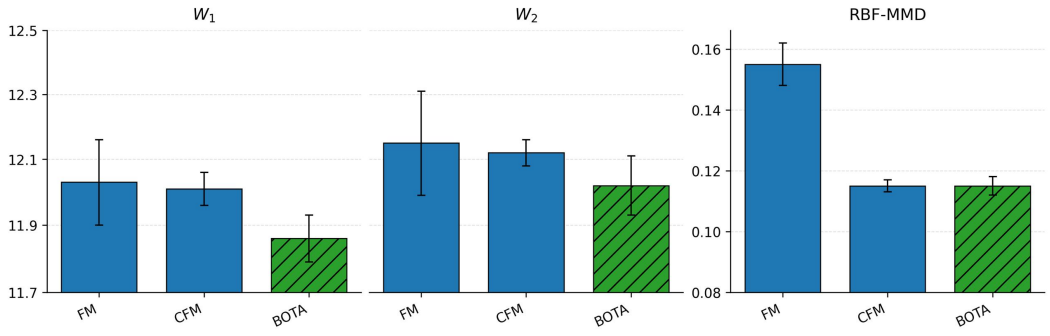
We evaluate BOTA on the Tedsim dataset, a controlled benchmark with known branching differentiation dynamics. Tedsim simulates cellular development from a single progenitor into multiple terminal states, providing both gene expression data and lineage information, making it well-suited for assessing the recovery of branching trajectories.



(a) Learned branched transport on Tedsim dataset.

Method	Cost ↓
FM	405.08 ± 1.00
CFM	408.88 ± 1.00
BOTA	$\mathbf{174.81 \pm 5.02}$

(b) Soft-atomic cost comparison.



(c) Quantitative metrics (W_1, W_2, RBF-MMD) with $\alpha = 0.5$.

Figure 8: BOTA results on the Tedsim dataset combining qualitative transport structure and quantitative evaluation.

As shown in Fig. 8, BOTA produces smooth trajectories with a shared progenitor trunk followed by coherent lineage splits, consistent with the trunk-then-split geometry of the simulated lineage, without mode collapse or spurious branching. Quantitatively, BOTA attains a $2.3\times$ lower soft-atomic cost than FM and CFM (Fig. 8b) with distributional metrics (W_1, W_2, RBF-MMD) at least as good as the best baseline. Both quantities are evaluated on the transport we solve (progenitor state to terminal states); a direct comparison against TedSim's ancestor–descendant coupling between its sampled time points would be the definitive test of path recovery and is left to future work.

A.4 IMPLEMENTATION DETAILS

Proxy atoms, K and σ. The K centers are not fixed in advance: at every time step they are rebuilt around the current particle cloud, inside a box of half-width $2\,\mathrm{std}(x_t)$ centered at the particle mean (a regular grid for $D \leq 3$, a Sobol design for larger D), so their placement follows the mass as it moves and σ only has to be set relative to the scale of the representation. This is what makes the two hyperparameters easy to set: $\sigma = 0.6$ and $K = 50$ are used unchanged in 2-D, in 50-D TedSim PCA space, and in ALAE latents, with no per-dataset tuning. Both failure modes are visible: with σ too large every weight tends to $1/K$, the cost stops depending on position and the branching signal disappears; with σ too small the weights become one-hot and gradients vanish (the non-differentiable limit of Lemma 1). K sets how finely "travelling together" is resolved: since the soft masses sum to one over centers, the cost is $\approx \|\bar{v}\|$ when particles share a center and $\approx \|\bar{v}\|\, K^{1-\alpha}$ when they do not; raising K is the safe direction (cost per iteration grows only linearly in K), and $K = B$ with centers at the particle positions recovers the exact atomic cost of Lemma 1. K therefore follows the number of structures to resolve, not the dimension.

Solver and amortization. Stage 1 optimizes V directly for a batch of $N = 10{,}000$ particles with $T = 50$ Euler steps on TedSim ($T = 10$ for the 2-D Gaussian figures), terminal weight $\lambda = 20$ and unit energy weight. Stage 2 regresses v_θ (an MLP with 4 hidden layers of width 256) onto (x_t, u_t) samples drawn uniformly from the discrete trajectory. Integration at test time uses the same fixed-step Euler scheme and step count as the reference trajectory for every method in Table 1; since stage 2 regresses onto a path that is itself defined by Euler integration on the T-step grid, the same scheme reproduces the reference by construction, and under-resolution would appear as terminal error, which Table 1 does not show.

Runtime and complexity. The branched machinery adds a bounded, one-off training overhead and no sampling overhead. Stage 1 optimizes the discrete velocity tensor $V \in \mathbb{R}^{T\times N\times D}$ directly, with no neural network: each iteration costs $O(T \cdot N \cdot K \cdot D)$, linear in the number of particles N, with no N^2 term (unlike an OT or Sinkhorn pairing, which costs $O(N^2 D)$ before training begins). Stage 2 is unmodified flow matching, so its cost equals that of the FM baseline. Table 3 reports wall-clock training times on Tedsim (50-D PCA) under the identical training protocol of Table 1. In total, BOTA trains in roughly $2\times$ the time of FM, paid once offline, while sampling cost is identical to FM since inference integrates a single velocity field of the same architecture.

Table 3: Training cost on Tedsim (50-D PCA). The discrete solver stage involves no network training; the amortization stage is standard flow matching. Sampling cost is identical for all methods.

	FM	CFM	BOTA
Solver stage	—	—	425 s (no network)
Network training	360 s	480 s	360 s (unmodified FM)
Total training	$1\times$ FM	$1.3\times$ FM	$\approx 2\times$ FM
Per-iteration cost in N	$O(N)$	$O(N^2)$ if OT-paired	$O(T \cdot N \cdot K \cdot D)$, linear in N
Sampling cost	$1\times$	$1\times$	$1\times$