TrajectoryNet: A Dynamic Optimal Transport Network for Modeling Cellular Dynamics

Alexander Tong [1] **Jessie Huang** [1] **Guy Wolf** [*] [2] [3] **David van Dijk** [*] [1] [4] **Smita Krishnaswamy** [*] [1] [5]

Abstract

It is increasingly common to encounter data from dynamic processes captured by static cross-sectional measurements over time, particularly in biomedical settings. Recent attempts to model individual trajectories from this data use optimal transport to create pairwise matchings between time points. However, these methods cannot model continuous dynamics and non-linear paths that entities can take in these systems. To address this issue, we establish a link between continuous normalizing flows and dynamic optimal transport, that allows us to model the expected paths of points over time. Continuous normalizing flows are generally under constrained, as they are allowed to take an arbitrary path from the source to the target distribution. We present *TrajectoryNet*, which controls the continuous paths taken between distributions to produce dynamic optimal transport. We show how this is particularly applicable for studying cellular dynamics in data from single-cell RNA sequencing (scRNA-seq) technologies, and that TrajectoryNet improves upon recently proposed static optimal transport-based models that can be used for interpolating cellular distributions.

1. Introduction

In data science we are often confronted with cross-sectional samples of time-varying phenomena, especially in biomedical data. Examples include health measurements of different age cohorts (Oeppen & Vaupel, 2002), or disease measurements at different stages of disease progression (Waddington, 1942). In these measurements we consider data that is sampled at multiple timepoints, but at each timepoint we have access only to a distribution (cross-section) of the population at that time. Extracting the longitudinal dynamics of development or disease from static snapshot measurements can be challenging as there are few methods of interpolation between distributions. Further exacerbating this problem is the fact that the same entities are often not measured at each time, resulting in a lack of point-to-point correspondences. Here, we propose to formulate this problem as one of unbalanced dynamic transport, where the goal is to transport entities from one cross sectional measurement to the next using efficient and smooth paths. Our main contribution is to establish a link between *continuous normalizing flows* (CNF) (Grathwohl et al., 2019) and dynamic optimal transport (Benamou & Brenier, 2000), allowing us to efficiently solve the transport problem using a Neural ODE framework (Chen et al., 2018a). To our knowledge, TrajectoryNet[1] is the first method to consider the specific paths taken by a CNF between distributions.

The continuous normalizing flow formulation allows us to generalize optimal transport to a series of distributions as in recent work (Chen et al., 2018b; Benamou et al., 2019). These works focus on the theoretical aspects of the problem, here focus on the computational aspects. This link allows us to smooth flows over multiple and possibly unevenly spaced distributions in high dimensions. This matches the setting of time series data from single-cell RNA sequencing.

Single-cell RNA sequencing (Macosko et al., 2015) is a relatively new technology that has made it possible for scientists to randomly sample the entire transcriptome, i.e., 20-30 thousand species of mRNA molecules representing transcribed genes of the cell. This technology can reveal detailed information about the identity of individual cells based on transcription factors, surface marker expression, cell cycle and many other facets of cellular behavior. In particular, this technology can be used to learn how cells differentiate from one state to another: for example, from embryonic stem cells to specified lineages such as neuronal or

[*]Equal contribution [1]Department of Computer Science, Yale University, New Haven, CT, USA [2]Department of Mathematics & Statistics, Université de Montréal, Montréal, QC, Canada [3]Mila – Quebec AI Institute, Montréal, QC, Canada [4]Internal Medicine, Cardiology, Yale University, New Haven, CT, USA [5]Department of Genetics, Yale University, New Haven, CT, USA. Correspondence to: Smita Krishnaswamy <smita.krishnaswamy@yale.edu>.

Proceedings of the 37^{th} International Conference on Machine Learning, Online, PMLR 119, 2020. Copyright 2020 by the author(s).

[1]Code is available here: https://github.com/KrishnaswamyLab/TrajectoryNet

cardiac. However, hampering this understanding is the fact that scRNA-seq only offers static snapshots of data, since all cells are destroyed upon measurement. Thus it is impossible to monitor how an individual cell changes over time. Moreover, due to the expensive nature of this technology, generally only a handful of discrete timepoints are collected in measuring any transition process. TrajectoryNet is especially well suited to this data modality. Existing methods attempt to infer a trajectory within one timepoint (Haghverdi et al., 2016; Saelens et al., 2019; La Manno et al., 2018), or interpolate linearly between two timepoints (Yang & Uhler, 2019; Schiebinger et al., 2019), but TrajectoryNet can interpolate non-linearly using information from more than two timepoints. TrajectoryNet has advantages over existing methods in that it:

1. can interpolate by following the manifold of observed entities between measured timepoints, thereby solving the static-snapshot problem,
2. can create continuous-time trajectories of individual entities, giving researchers the ability to follow an entity in time,
3. forms a deep representational model of system dynamics, which can then be used to understand drivers of dynamics (gene logic in the cellular context), via perturbation of this deep model.

While our experiments apply this work specifically to cellular dynamics, these penalties can be used in many other situations where we would like to model dynamics based on cross-sectional population level data.

2. Background and Related Work

Optimal Transport. Introduced originally by (Monge, 1781) and in modern form by (Kantorovich, 1942), the linear program formulation of static optimal transport (OT) has the relatively high cost of $O(n^3)$ for discrete measures. Recently, there have been a number of fast approximations using entropic regularization. Cuturi (2013) presented a parallel algorithm for the discrete case as an application of Sinkhorn's algorithm (Sinkhorn, 1964). Recent effort approximates OT on subspaces (Muzellec & Cuturi, 2019) or even a single dimension (Kolouri et al., 2019). These efforts emphasis the importance to the field of obtaining fast OT algorithms. Another direction that has recently received increased attention is in unbalanced optimal transport where the goal is to relax the problem to add and remove mass (Benamou, 2003; Chizat et al., 2018; Liero et al., 2018; Schiebinger et al., 2019). While many efficient static optimal transport algorithms exist, and recently for the unbalanced case (Yang & Uhler, 2019), much less attention has focused on dynamic optimal transport, the focus of this work.

Dynamic Optimal Transport. Another formulation of optimal transport is known as *dynamic* optimal transport. Benamou & Brenier (2000) showed how the addition of a natural time interpolation variable gives an alternative interpretation with links to fluid dynamics that surprisingly leads to a convex optimization problem. However, while solvers for the discretized dynamic OT problem are effective in low dimensions and for small problems they require a discretization of space into grids giving cost exponential in the dimension (See Peyré & Cuturi (2019) Chap. 7 for a good overview of this problem). One of our main contributions is to provide an approximate solver for high dimensional smooth problems using a neural network.

Single-cell Trajectories from a Static Snapshot. Temporal interpolation in single-cell data started with solutions that attempt to infer an axis within one single time point of data cell "pseudotime" – used as a proxy for developmental progression – using known markers of development and the asynchronous nature of cell development (Trapnell et al., 2014; Bendall et al., 2014). An extensive comparison of 45 methods for this type of analysis gives method recommendations based on prior assumptions on the general structure of the data (Saelens et al., 2019). However, these methods can be biased and fail in a number of circumstances (Weinreb et al., 2018; Lederer & La Manno, 2020) and do not take into account experimental time.

Matching Populations from Multiple Time Points. Recent methods get around some of these challenges using multiple timepoints (Hashimoto et al., 2016; Schiebinger et al., 2019; Yang & Uhler, 2019). However, these methods generally resort to matching populations between coarse-grained timepoints, but do not give much insight into how they move between measured timepoints. Often paths are assumed to minimize total Euclidean cost, which is not realistic in this setting. In contrast, the methods that estimate dynamics from single timepoints (La Manno et al., 2018; Bergen et al., 2019; Erhard et al., 2019; Hendriks et al., 2019) have the potential to give relatively accurate estimation of local direction, but cannot give accurate global estimation of distributional shift. A recent line of work on generalizing splines to distributions (Chen et al., 2018b; Benamou et al., 2019) investigates this problem from a theoretical perspective, but provides no efficient implementation.

With TrajectoryNet, we aim to unite these approaches into a single model combining in inferring continuous time trajectories from multiple timepoints, globally, while respecting local dynamics within a single timepoint.

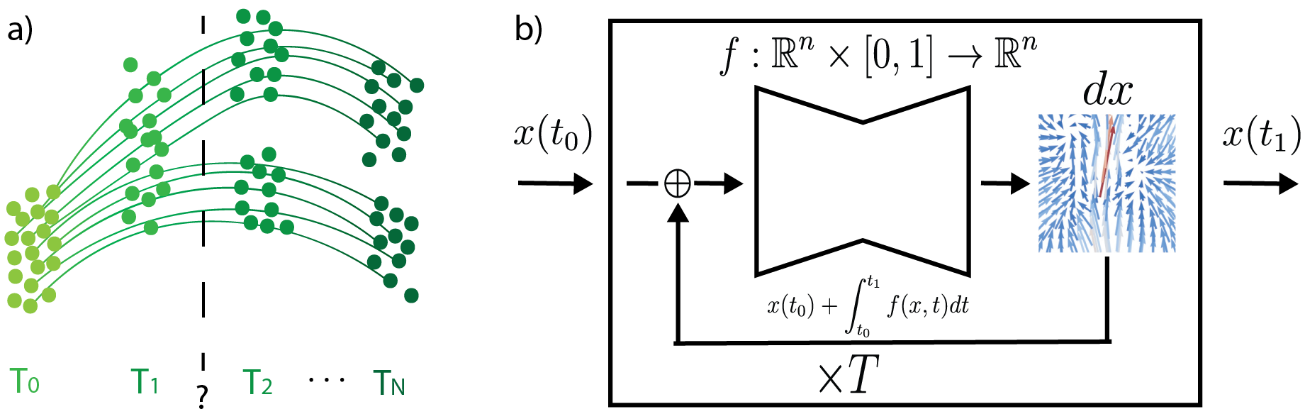
a)
T0
T1
?
T2
· · ·
TN
b)
$f : \mathbb{R}^n \times [0,1] \to \mathbb{R}^n$
$x(t_0)$
dx
$x(t_1)$
$x(t_0) + \int_{t_0}^{t_1} f(x,t)dt$
$\times T$

Figure 1. TrajectoryNet learns trajectories of particles from distributions sampled over time. We use a Neural ODE to learn the derivative of the dynamics function. To find the output at time t_1 for a given input at time t_0 we integrate T times letting the ODE solver choose the integration timepoints.

3. Preliminaries

We provide an overview of static optimal transport, dynamic optimal transport (Benamou & Brenier, 2000), and continuous normalizing flows.

3.1. The Monge-Kantorovich Problem

We adopt notation from the standard text (Villani, 2008). For two probability measures μ, ν defined on $\mathcal{X} \subset \mathbb{R}^n$, let $\Pi(\mu, \nu)$ denote the set of all joint probability measures on $\mathcal{X} \times \mathcal{X}$ whose marginals are μ and ν. Then the p-Kantorovich distance (or Wasserstein distance of order p)between μ and ν is

$$W(\mu,\nu)_p := \left(\inf_{\pi \in \Pi(\mu,\nu)} \int_{\mathcal{X}\times\mathcal{Y}} d(x,y)^p d\pi(x,y) \right)^{1/p}, \tag{1}$$

where $p \in [1, \infty)$. This formulation has led to many useful interpretations both in GANs and biological networks. For the entropy regularized problem, the Sinkhorn algorithm (Sinkhorn, 1964) provides a fast and parallelizable numerical solution in the discrete case. Recent work tackles computationally efficient solutions to the exact problem (Jambulapati et al., 2019) for the discrete case. However, for the continuous case solutions to the discrete problem in high dimensional spaces do not scale well. As the rate of convergence of the empirical Wasserstein metric between empirical measures $\hat{\mu}$ and $\hat{\nu}$ with bounded support is shown in (Dudley, 1969) to be

$$\mathbb{E}[|W_p(\hat{\mu}_n, \hat{\nu}_n) - W_p(\mu,\nu)|] = O(n^{-\frac{1}{d}}) \tag{2}$$

where d is the ambient dimension. However, recent work shows that in high dimensions a more careful treatment that the rate depends on the intrinsic dimension not the ambient dimension (Weed & Bach, 2019). As long as data lies in a low dimensional manifold in ambient space, then we can reasonably approximate the Wasserstein distance. In this work we approximate the support of this manifold using a neural network.

3.2. Dynamic Optimal Transport

Benamou & Brenier (2000) defined and explored a dynamic version of Kantorovich distance. Their work linked optimal transport distances with dynamics and partial differential equations (PDEs). For a fixed time interval $[t_0, t_1]$ with smooth enough, time dependent density and velocity fields, $P(x,t) \geq 0, f(x,t) \in \mathbb{R}^d$, subject to the continuity equation

$$\partial_t P + \nabla \cdot (Pf) = 0 \tag{3}$$

for $t_0 < t < t_1$ and $x \in \mathbb{R}^d$, and the conditions

$$P(\cdot, t_0) = \mu, \quad P(\cdot, t_1) = \nu \tag{4}$$

we can relate the squared L^2 Wasserstein distance to (P, f) in the following way

$$W(\mu,\nu)_2^2 = \inf_{(P,f)} (t_1 - t_0) \int_{\mathbb{R}^d} \int_{t_0}^{t_1} P(x,t)|f(x,t)|^2 dt dx \tag{5}$$

In other words, a velocity field $f(x,t)$ with minimum L^2 norm that transports mass at μ to mass at ν when integrated over the time interval is the optimal plan for an L^2 Wasserstein distance. The continuity equation is applied over all points of the field and asserts that no point is a source or sink for mass. The solution to this flow can be shown to follow a pressureless flow on a time-dependent potential function. Mass moves with constant velocity that linearly interpolates between the initial and final measures. For problems where interpolation of the density between two known densities is of interest, this formulation is very attractive. Existing computational methods for solving the dynamic formulation for continuous measures approximate the flow using a discretization of space-time (Papadakis et al., 2014). This works well in low dimensions, but scales poorly to high dimensions as the complexity is exponential in the input dimension d. We next give background on continuous normalizing flows, which we show can provide a solution with computational complexity polynomial in d.

3.3. Continuous Normalizing Flows

A normalizing flow (Rezende & Mohamed, 2015) transforms a parametric (usually simple) distribution to a more complicated one. Using an invertible transformation f applied to an initial latent random variable y with density P_y, We define $x = f(y)$ as the output of the flow. Then by the change of variables formula, we can compute the density of the output x:

$$\log P_x(\cdot) = \log P_y(\cdot) - \log \ \det \frac{\partial f}{\partial y} \tag{6}$$

A large effort has gone into creating architectures where the log determinant of the Jacobian is efficient to compute (Rezende & Mohamed, 2015; Kingma et al., 2016; Papamakarios et al., 2017).

Now consider a continuous-time transformation, where the derivative of the transformation is parameterized by θ, thus at any timepoint t, $\frac{\partial x(t)}{\partial t} = f_\theta(x(t), t)$. At the initial time t_0, $x(t_0)$ is drawn from a distribution $P(x, t_0)$ which we also denote $P_{t_0}(x)$ for clarity, and it's continuously transformed to $x(t_1)$ by following the differential equation $f_\theta(x(t), t)$:

$$\begin{aligned} &x(t_1) = x(t_0) + \int_{t_0}^{t_1} f_\theta(x(t), t) dt, \quad x(t_0) \sim P_{t_0}(x), \\ &\log P_{t_1}(x(t_1)) = \\ &\qquad \log P_{t_0}(x(t_0)) - \int_{t_0}^{t_1} Tr\left(\frac{\partial f_\theta(x(t), t)}{\partial x(t)}\right) dt, \end{aligned} \tag{7}$$

where at any time t associated with every x through the flow can be found by following the inverse flow. This model is referred as continuous normalizing flows (CNFs) (Chen et al., 2018a). It can be likened to the dynamic version of optimal transport, where we model the measure over time rather than the mapping from P_{t_0} to P_{t_1}

Unsurprisingly, there is a deep connection between CNFs and dynamic optimal transport. In the next section we exploit this connection and show how CNFs can be used to provide a high dimensional solution to the dynamic optimal transport problem with TrajectoryNet.

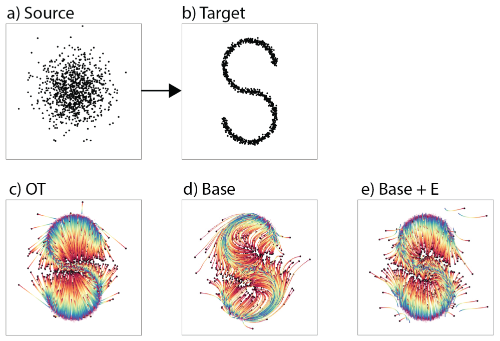

Figure 2. Transporting a Gaussian (a) to an S-curve (b) via (c) static optimal transport, (d) Base TrajectoryNet without regularization follows density (e) TrajectoryNet with energy regularization demonstrates more straight paths similar to OT.

4. TrajectoryNet: Efficient Dynamic Optimal Transport

In this section, we first describe how to adapt continuous normalizing flows to approximate dynamic optimal transport in (Section 4.1). We then describe further adaptations for analysis of single-cell data in (Section 4.2) and finally provide training details in (Section 4.3).

4.1. Dynamic OT Approximation via Regularized CNF

Continuous normalizing flows use a maximum likelihood objective which can be equivalently expressed as a KL divergence. In TrajectoryNet we add an energy regularization to approximate dynamic OT. Dynamic OT is expressed with an optimization over flows with constraints at t_0 and t_1 (see eq. (4)). By relaxing this constraint to minimizing a divergence at t_1 CNFs can approximate dynamic OT.

For sufficiently large λ under constraint (3) this converges to the optimal solution in (5). This is encapsulated in the following theorem. See Appendix A.1 for proof.

Theorem 4.1. *With time varying field $f(x,t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^d$ and density $P(x,t) : \mathbb{R}^d \times \mathbb{R} \to \mathbb{R}^+$ such that $\int P(x,t)dx = 1$ for all $t_0 \leq t \leq t_1$ and subject to the continuity (3). There exists a sufficiently large λ such that*

$$\begin{aligned} W(\mu,\nu)_2^2 = (t_1 - t_0) \inf_{(P,f)} \mathop{\mathbb{E}}_{x_0 \sim \mu} \left[\int_{t_0}^{t_1} \|f(x(t), t)\|^2 dt \right] \\ + \lambda KL(P(\cdot, t_1) \parallel \nu); \quad s.t.\ P(\cdot, t_0) = \mu \end{aligned}$$

Intuitively, a continuous normalizing flow with a correctly scaled penalty on the squared norm of f approximates the W_2 transport between μ and ν. Dynamic optimal transport can be thought of as finding a distribution over paths such that the beginnings of the paths match the source distribution, end of the path matches the target distribution, and the cost of the transport is measured by expected path length. Continuous normalizing flows relax the target distribution match with a KL-divergence penalty. When the KL-divergence is small then the constraint is satisfied. However, CNFs usually do not enforce the path length constraint, which we add using a penalty on the norm of f as defined by the Neural ODE. When we impose this penalty over uniformly sampled data, this is equivalent to penalizing the expected path length.

We can approximate the first part of this continuous time equation using a Riemann sum as $\mathbb{E}_{x_0 \sim \mu}\left[\int_{t_0}^{t_1} \|f_\theta(x,t)\|^2 \, dt\right] = \sum_{x \sim P_{t_0}} \sum_{i=t_0}^{t_1} \Delta t_i \|f_\theta(x,t)\|^2$. This requires a forward integration using a standard ODE solver to compute as shown in Chen et al. (2018a). If we consider the case where the divergence is small, then this can be combined with the standard backwards pass for even less added computation. Instead of penalizing $\|f_\theta(x,t)\|^2$ on a forward pass, we penalize the same quantity on a backwards pass. Using the maximum likelihood and KL divergence equivalence, we obtain a loss

$$L(x) = -\log p(x) + \lambda_e \int_t \|f(x(t),t)\|^2 \tag{8}$$

Where the integral above is computed using an ODE solver. In practice, both a penalty on the Jacobian or additional training noise helped to get straight paths with a lower energy regularization λ_e. We found that a value of λ_e large enough to encourage straight paths, unsurprisingly also shortens the paths undershooting the target distribution. To counteract this, we add a penalty on the norm of the Jacobian of f as used in Vincent et al. (2010); Rifai et al. (2011). Since f represents the derivative of the path, this discourages paths with high local curvature, and can be thought of as penalizing the second derivative (acceleration) of the flow. Our energy loss is then

$$L_{energy}(x) = \lambda_e \int_t \|f(\tilde{x},t)\|^2 + \lambda_j \int_t \|J_f(\tilde{x})\|_F^2, \tag{9}$$

where $\|J_f(x)\|_F^2$ is the Frobenius norm of the Jacobian of f. Comparing Figure 2(e) to (d) we demonstrate the effect of this regularization. Without energy regularization TrajectoryNet paths follow the data. However, with energy regularization we approach the paths of the optimal map. TrajectoryNet solution biases towards undershooting the target distribution. Our energy loss gives control over how much to penalize indirect, high energy paths.

Optimal transport is traditionally performed between a source and target distribution. Extensions to a series of distributions is normally done by performing optimal transport between successive pairs of distributions as in Schiebinger et al. (2019). This creates flows that have discontinuities at the sampled times, which may be undesirable when the underlying system is smooth in time as in biological systems. The dynamic model approximates dynamic OT for two timepoints, but by using a single smooth function to model the whole series the flow becomes the minimal cost *smooth* flow over time.

4.2. Further Adaptation for Single-Cell Trajectories

Up to this point we have shown how to perform dynamic optimal transport in high dimensions with a regularized CNF. We now introduce priors needed to mimic cellular systems that are characterized by growth/death rather than just transport, endowed with a manifold structure, and knowledge of local velocity arrows. Similar priors may also be applicable to other data types. For example in studying the dynamics of a disease, people may be newly infected or cured, we may have knowledge on acceptable transition states, indicating a density penalty, or visits may be clustered such as in a hospital stay so we may have estimates of near term patient trends, indicating the use of velocity priors. To enforce these priors we add corresponding regularizations listed below:

1. A *growth rate regularization* that accommodates unbalanced transport, described in Section 4.2.1.
2. A *density-based penalty* which encourages interpolations that lie on dense regions of the data. Often data lies on a low-dimensional manifold, and it is desirable for paths to follow this manifold at the cost of higher energy (See Section 4.2.2 for details).
3. A *velocity regularization* where we enforce local estimates of velocity at measured datapoints to match the first time derivative of cell state change. (See Section 4.2.3 for details).

These regularizations are summarized in a single loss function defined as

$$\begin{aligned} L_T = & \underbrace{\overbrace{\sum_{i=1}^{k} -\log P_{t_i}(x_{t_i})}^{\text{Normalizing Flow}} + L_{energy}}_{\text{Dynamic OT}} \\ & + \underbrace{L_{density} + L_{velocity} + L_{growth}}_{\text{Biological priors}} \end{aligned} \tag{10}$$

4.2.1. Allowing Unbalanced Optimal Transport

We use a simple and computationally efficient method that adapts discrete static unbalanced optimal transport to our framework in the continuous setting. This is a necessary extension but is by no means a focus of our work. While we could also apply an adversarial framework, we choose to avoid the instabilities of adversarial training and use a simple network trained from the solution to the discrete problem. We train a network $G(x, t) : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^+$, which takes as input a cell state and time pair and produces a growth rate of a cell at that time. This is trained to match the result from discrete optimal transport. For further specification see Appendix B. We then fix weights of this network and modify the way we integrate mass over time to

$$\log M_{t_i}(x) = \log M_{t_{i-1}}(x) - \int_{t_{i-1}}^{t_i} Tr\left(\frac{\partial f_\theta(x(t), t)}{\partial x(t)}\right) dt + \log G(x_{t_{i-1}}, t_{i-1}) \tag{11}$$

We note that adding growth rate regularization in this way does not guarantee conservation of mass. We could normalize $M(x)$ to be a probability distribution during training, e.g., as $P(x) = M(x)/\sum_{x\in\mathbb{R}^d} M(x)$. However, this now requires an integration over $\mathbb{R}^d$, which is too computationally costly. Instead, we use the equivalence of the maximum likelihood formulation over a fixed growth function g and normalize it after the network is trained.

4.2.2. Enforcing Transport on a Manifold

Methods that display or perform computations on the cellular manifold often include an implicit or explicit way of normalizing for density and model data geometry. PHATE (Moon et al., 2019) uses scatter plots and adaptive kernels to display the geometry. SUGAR (Lindenbaum et al., 2018) explicitly models the data geometry. We would like to constrain our flows to the manifold geometry but not to its density. We penalize the flow such that it is always close to at least a few measured points across all timepoints.

$$L_{density}(x, t_d) = \sum_k \max(0, \text{min-k}\big(\{\|x(t_d) - z\| : z \in \mathcal{X}\}\big) - h) \tag{12}$$

This can be thought of as a loss that penalizes points until they are within h Euclidean distance of their k nearest neighbors. We use $h = 0.1$ and $k = 5$ in all of our experiments. We evaluate $L_{density}$ on an interpolated time $t_d \in (t_0, t_k)$ every batch.

4.2.3. Conforming to known Velocity

Often it is the case where it is easy to measure direction of change in a short time horizon, but not have good predictive power at the scale of measured timesteps. In health data, we can often collect data from a few visits over a short time horizon estimating the direction of a single patient in the near future. In single-cell data, RNA-velocity (La Manno et al., 2018; Bergen et al., 2019) provides an estimates $\widehat{dx/dt}$ at every measured cell. We use these measurements to regularize the direction of flow at every measured point. Our regularization requires evaluating $f(x, t)$ periodically at every measured cell adding the regularization:

$$\begin{aligned} L_{velocity}(x, t, \widehat{dx/dt}) &= \text{cosine-similarity}(f(x, t), \widehat{dx/dt}) \\ &= \frac{f(x, t) \cdot \widehat{dx/dt}}{\|f(x, t)\| \ \ \widehat{dx/dt}} \end{aligned} \tag{13}$$

This encourages the direction of the flow at a measured point to be similar to the direction of local velocity. This ignores the magnitude of the estimate, and only heeds the direction. While RNA-velocity provides some estimate of relative speed, the vector length is considered not as informative, as it is unclear how to normalize these vectors in a system specific way (La Manno et al., 2018; Bergen et al., 2019). We note that while current estimates of velocity can only estimate direction, this does not preclude future methods that can give accurate magnitude estimates. $L_{velocity}$ can easily be adapted to take magnitudes into account by considering L_2 similarity for instance.

4.3. Training

For simplicity, the neural network architecture of TrajectoryNet consists of three fully connected layers of 64 nodes with leaky ReLU activations. It takes as input a cell state and time and outputs the derivative of state with respect to time at that point. To train a continuous normalizing flow we need access to the density function of the source distribution. Since this is not accessible for an empirical distribution we use an additional Gaussian at t_0, defining $P_{t_0}(\cdot) = \mathcal{N}(0, 1)$, the standard Gaussian distribution, where $P_t(x)$ is the density function at time t.

For a training step we draw samples $x_{t_i} \sim \mathcal{X}_{t_i}$ for $i \in \{1, \ldots, k\}$ and calculate the loss with a single backwards integration of the ODE. In the following sections we will explain how adding the individual penalty terms achieve regularized trajectories. While there are a number of ways to computationally approximate these quantities, we use a parallel method to iteratively calculate the $\log P_{t_i}$ based on $\log P_{t_{i-1}}$. To make a backward pass through all timepoints we start at the final timepoint, integrate the batch to the second to last timepoint, concatenate these points to the samples from the second to last timepoint, and continue till t_0, where the density is known for each sample. We note that this can compound the error especially for later timepoints if k is large or if the learned system is stiff, but gives significant speedup during training.

To sample from P_{t_i} we first sample $\hat{x}_{t_0} \sim P_{t_0}$ then use the adjoint method to perform the integration $\hat{x}_{t_i} = \hat{x}_{t_0} + \int_{t_0}^{t_i} f_\theta(x(t), t)dt; \quad x(0) = \hat{x}_{t_0}$.

5. Experiments

All experiments were performed with the TrajectoryNet framework with a network of consisting of three layers with LeakyReLU activations. Optimization was performed on 10,000 iterations of batches of size 1,000 using the dopri5 solver (Dormand & Prince, 1980) with both absolute and relative tolerances set to 1×10^{-5} and the ADAM optimizer (Kingma & Ba, 2015) with learning rate 0.001, and weight decay 5×10^{-5} as in (Grathwohl et al., 2019). We evaluate using three TrajectoryNet models with different regularization terms. The Base model refers to a standard normalizing flow. +E adds L_{energy}, +D adds $L_{density}$,+V adds $L_{velocity}$, and +G adds L_{growth}.

Comparison to Existing Methods. Since there are no ground truth methods to calculate the trajectory of a single cell we evaluate our model using interpolation of held-out timepoints. We leave out an intermediary timepoint and measure the Kantorovich-distance also known as the earth mover's distance (EMD) between the predicted and held-out distributions. For EMD lower is more accurate. We compare the distribution interpolated by TrajectoryNet with four other distributions. The previous timepoint, the next timepoint, a random timepoint and the McCann interpolant in the discrete OT solution as used in (Schiebinger et al., 2019).

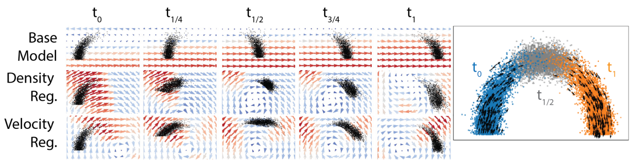

Figure 3. Density regularization or velocity regularization can be used to follow a 1D manifold in 2D.

	EMD Arch	Cycle	Tree	MSE Arch	Cycle	Tree
Base	0.691	0.037	0.490	0.300	0.190	0.218
Base + D	0.607	0.049	0.373	0.236	0.191	0.145
Base + V	**0.243**	0.033	**0.143**	**0.107**	**0.068**	**0.098**
Base + D + V	0.415	0.034	0.252	0.156	0.081	0.132
OT	0.644	**0.032**	0.492	0.252	0.192	0.196
prev	1.086	0.035	1.092	0.652	0.192	0.666
next	1.090	0.035	1.068	0.659	0.192	0.689
rand	0.622	0.406	0.420	0.243	0.346	0.161

Table 1. Shows the Wasserstein distance EMD and MSE for artificial datasets between the left out timepoint and the predicted points for our two generated datasets. Mean over 3 seeds.

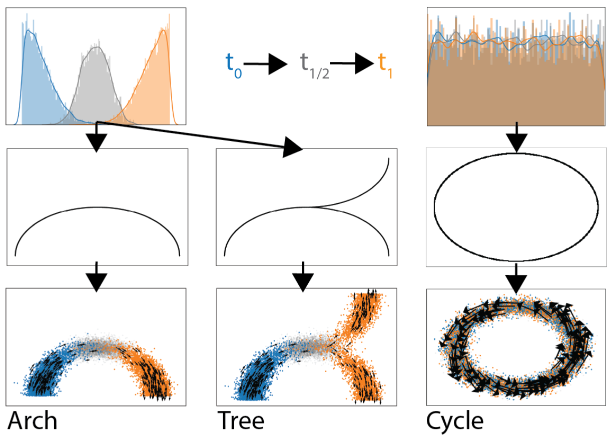

Figure 4. A 1D distribution of data over time embedded in two dimensions along a smooth manifold. On a single branch (left), with a tree structure (center), and circle (right).

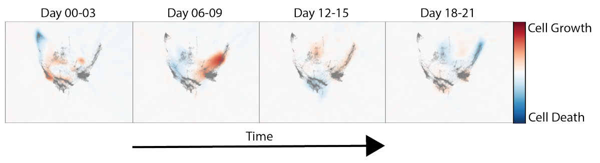

Figure 5. Cell growth model learned on Embryoid Body Data (Moon et al., 2019)

5.1. Artificial Data

For artificial data where we have known paths, we can measure the mean squared error (MSE) predicted by the model based on the first timepoint. Here we leave out the middle timepoint $t_{1/2}$ for training then calculate the MSE between the predicted point at time $t_{1/2}$ and the true point at $t_{1/2}$ for 5000 sampled trajectories. This gives a measure of how accurately we can model simple dynamical systems.

We first test TrajectoryNet on two datasets where points lie on a 1D manifold in 2D with Gaussian noise (See Figure 4). First two half Gaussians are sampled with means zero and one in one dimension. These progressions are then lifted onto curved manifolds in two dimensions either an arch or a tree mimicking a differentiating system where we have two sampled timepoints that have some overlap. Table 1 shows the Wasserstein distance (EMD) and the mean squared error for different interpolation methods between the interpolated distribution at $t_{1/2}$ and the true interpolated distribution at $t_{1/2}$. Because optimal transport considers the shortest Euclidean distance, the base model and OT methods follow the lowest energy path, which is straight across. With density regularization or velocity regularization TrajectoryNet learns paths that follow the density manifold. Figure 3 and Figure S2 demonstrate how TrajectoryNet with density or velocity regularization learns to follow the manifold.

A third artificial dataset shows the necessity of using velocity estimates for some data. Here we have an unchanging distribution of points distributed uniformly over the unit circle, but are traveling counterclockwise at $\pi/5$ radians per unit time. This is similar to the cell-cycle process in adult systems. Without velocity estimates it is impossible to pick up this type of dynamical system. This is illustrated by the MSE of the cycle dataset using velocity regularization in Table 1.

5.2. Single-Cell Data

We run our model on 5D PCA due to computational constraints, but note that computation time scales roughly linearly with dimension for our test cases (See Appendix C), which is consistent to what was found in Grathwohl et al. (2019). Since there are no ground truth trajectories in real data, we can only evaluate using distributional distances. We do leave-one-out validation, training the model on all but one of the intermediate timepoints then evaluating the EMD between the validation data and the model's predicted distribution. We evaluate and compare our method on two single-cell RNA sequencing datasets.

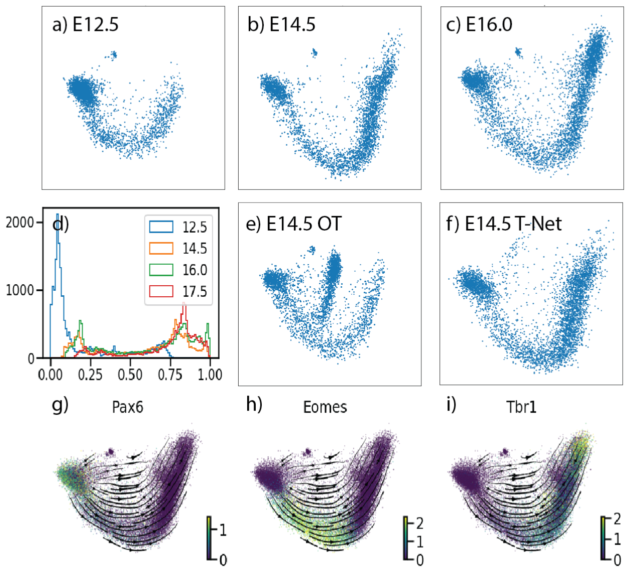

Figure 6. Shows the first 2 PCs of the mouse cortex dataset. (a-c) show the distributions for the first three timepoints. (d) shows the distribution of cells over PC1. the interpolated points for E14.5 using (e) static OT, and (f) TrajectoryNet with density regularization. (g-i) shows expression of three markers of early (Pax6) mid (Eomes) and late (Tbr1) stage neurons.

Mouse Cortex Data.[2] The first dataset has structure similar to the Arch toy dataset. It consists of cells collected from mouse embryos at days E12.5, E14.5, E16, and E17.5. In Figure 6(d) we can see at this time in development of the mouse cortex the distribution of cells moves from a mostly neural stem cell population at E12.5 to a fairly developed and differentiated neuronal population at E17.5 (Cotney et al., 2015; Katayama et al., 2016). The major axis of variation is neuron development. Over the 4 timepoints we have 2 biological replicates that we can use to evaluate variation between animals. In Table 2, we can see that TrajectoryNet outperforms baseline models, especially when adding density and velocity information. The curved manifold structure of this data, and gene expression data in general means that methods that interpolate with straight paths cannot fully capture the structure of the data. Since TrajectoryNet models full paths between timepoints, adding density and velocity information can bend the cell paths to follow the manifold utilizing all available data rather than two timepoints as in standard optimal transport.

[2]For videos of the dynamics learned by TrajectoryNet see `http://github.com/krishnaswamylab/TrajectoryNet`

	rep1	rep2	mean
Base	0.888 $\pm$ 0.07	0.905 $\pm$ 0.06	0.897 $\pm$ 0.06
Base + D	0.882 $\pm$ 0.03	0.895 $\pm$ 0.03	0.888 $\pm$ 0.03
Base + V	0.900 $\pm$ 0.09	0.898 $\pm$ 0.10	0.899 $\pm$ 0.10
Base + D + V	**0.851** $\pm$ 0.08	**0.866** $\pm$ 0.07	**0.859** $\pm$ 0.07
OT	1.098	1.095	1.096
prev	1.628	1.573	1.600
next	1.324	1.391	1.357
rand	1.333	1.288	1.311

Table 2. Shows the Wasserstein distance between the left out timepoint and the predicted distribution for various methods on a 4 timepoint mouse embryo cortex dataset. Mean and standard deviation over 3 seeds.

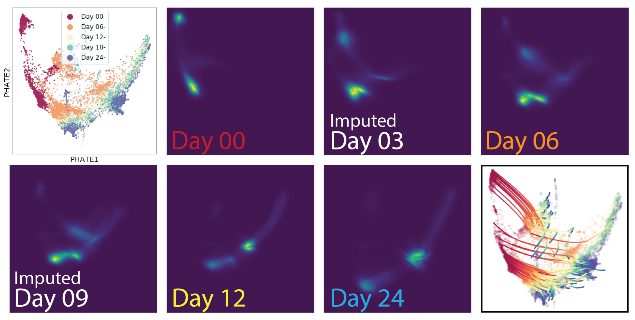

Figure 7. Shows the Embryoid body dataset projected into 2D with PHATE (Moon et al., 2019) with paths and densities imputed using TrajectoryNet.

Embryoid body Data. Next, we evaluate on a differentiating Embryoid body scRNA-seq time course. Figure 7 shows this data projected into two dimensions using a non-linear dimensionality reduction method called PHATE (Moon et al., 2019). This data consists of 5 timepoints of single cell data collected in a developing human embryo system (Day 0-Day 24). See Figure 5 for a depiction of the growth rate. Initially, cells start as a single stem cell population, but differentiate into roughly 4 cell precursor types. This gives a branching structure similar to our artificial tree dataset. In Table 3 we show results when each of the three intermediate timepoints are left out. In this case velocity regularization does not seem to help, we hypothesis this has to do with the low unspliced RNA counts present in the data (See Figure S3). We find that energy regularization and growth rate regularization help only on the first timepoint, and that density regularization helps the most overall.

We can also project trajectories back to gene space. This gives insights into when populations might be distinguishable. In Figure 8, we demonstrate how TrajectoryNet can be projected back to the gene space. We sample cells from the end of the four main branches, then integrate TrajectoryNet backwards to get their paths through gene space. This recapitulates known biology in Moon et al. (2019). See appendix D for a more in-depth treatment.

	t=1	t=2	t=3	mean
Base	0.764	0.811	0.863	0.813
Base + D	0.759	**0.783**	0.811	**0.784**
Base + V	0.816	0.839	0.865	0.840
Base + D + V	0.930	0.806	**0.810**	0.848
Base + E	. 0.737	0.896	0.842	0.825
Base + G	**0.700**	0.913	0.829	0.814
OT	0.791	0.831	0.841	0.821
prev	1.715	1.400	0.814	1.309
next	1.400	0.814	1.694	1.302
rand	0.872	1.036	0.998	0.969

Table 3. Shows the Wasserstein distance (EMD) between the left out timepoint and the predicted distribution for various methods on the 5 timepoint Embryoid body dataset.

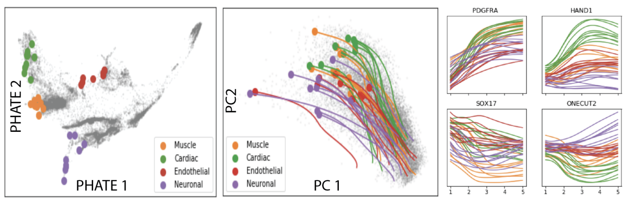

Figure 8. For curated endpoints, shows location on PHATE dimensions, TrajectoryNet paths projected into PCA space, and trajectories for 4 genes.

6. Conclusion

TrajectoryNet computes dynamic optimal transport between distributions of samples at discrete times to model realistic paths of samples continuously in time. In the single-cell case, TrajectoryNet ”reanimates,” cells which are destroyed by measurement to recreate a continuous-time trajectory. This is also relevant when modeling any underlying system that is high-dimensional, dynamic, and non-linear. In this case, existing static OT methods are under-powered and do not interpolate well to intermediate timepoints between measured ones. Existing dynamic OT methods (non-neural network based) are computationally infeasible for this task.

In this work we integrate multiple priors and assumptions into one model to bias TrajectoryNet towards more realistic dynamic optimal transport solutions. We demonstrated how this gives more power to discover hidden and time specific relationships between features. In future work, we would like to consider stochastic dynamics (Li et al., 2020) and learning the growth term together with the dynamics.

Acknowledgements

This research was partially funded by IVADO (l’institut de valorisation des données) [*G.W.*]; Chan-Zuckerberg Initiative grants 182702 & CZF2019-002440 [*S.K.*]; and NIH grants R01GM135929 & R01GM130847 [*G.W., S.K.*]. The content provided here is solely the responsibility of the authors and does not necessarily represent the official views of the funding agencies.

References

Benamou, J.-D. Numerical resolution of an “unbalanced” mass transport problem. *ESAIM: Mathematical Modelling and Numerical Analysis*, 37(5):851–868, 2003.

Benamou, J.-D. and Brenier, Y. A computational fluid mechanics solution to the Monge-Kantorovich mass transfer problem. *Numerische Mathematik*, 84(3):375–393, 2000.

Benamou, J.-D., Gallouët, T. O., and Vialard, F.-X. Second-Order Models for Optimal Transport and Cubic Splines on the Wasserstein Space. *Foundations of Computational Mathematics*, 19(5):1113–1143, 2019.

Bendall, S. C., Davis, K. L., Amir, E.-a. D., Tadmor, M. D., Simonds, E. F., Chen, T. J., Shenfeld, D. K., Nolan, G. P., and Pe’er, D. Single-Cell Trajectory Detection Uncovers Progression and Regulatory Coordination in Human B Cell Development. *Cell*, 157(3):714–725, 2014.

Bergen, V., Lange, M., Peidli, S., Wolf, F. A., and Theis, F. J. Generalizing RNA velocity to transient cell states through dynamical modeling. *BioRxiv 820936*, 2019.

Chen, R. T. Q., Rubanova, Y., Bettencourt, J., and Duvenaud, D. Neural Ordinary Differential Equations. In *Advances in Neural Information Processing Systems 31*, 2018a.

Chen, Y., Conforti, G., and Georgiou, T. T. Measure-Valued Spline Curves: An Optimal Transport Viewpoint. *SIAM Journal on Mathematical Analysis*, 50(6):5947–5968, 2018b.

Chizat, L., Peyré, G., Schmitzer, B., and Vialard, F.-X. Unbalanced optimal transport: Dynamic and Kantorovich formulations. *Journal of Functional Analysis*, 274(11): 3090–3123, 2018.

Cotney, J., Muhle, R. A., Sanders, S. J., Liu, L., Willsey, A. J., Niu, W., Liu, W., Klei, L., Lei, J., Yin, J., Reilly, S. K., Tebbenkamp, A. T., Bichsel, C., Pletikos, M., Sestan, N., Roeder, K., State, M. W., Devlin, B., and Noonan, J. P. The autism-associated chromatin modifier CHD8 regulates other autism risk genes during human neurodevelopment. *Nature Communications*, 6(1):6404, 2015.

Cuturi, M. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems 26*, pp. 2292–2300, 2013.

Dormand, J. and Prince, P. A family of embedded Runge-Kutta formulae. *Journal of Computational and Applied Mathematics*, 6(1):19–26, 1980.

Dudley, R. M. The Speed of Mean Glivenko-Cantelli Convergence. *The Annals of Mathematical Statistics*, 40(1): 40–50, 1969.

Erhard, F., Baptista, M. A. P., Krammer, T., Hennig, T., Lange, M., Arampatzi, P., Jürges, C. S., Theis, F. J., Saliba, A.-E., and Dölken, L. scSLAM-seq reveals core features of transcription dynamics in single cells. *Nature*, 571(7765):419–423, 2019.

Grathwohl, W., Chen, R. T. Q., Bettencourt, J., Sutskever, I., and Duvenaud, D. FFJORD: Free-form Continuous Dynamics for Scalable Reversible Generative Models. In *7th International Conference on Learning Representations*, 2019.

Haghverdi, L., Büttner, M., Wolf, F. A., Buettner, F., and Theis, F. J. Diffusion pseudotime robustly reconstructs lineage branching. *Nature Methods*, 13(10):845–848, 2016.

Hashimoto, T. B., Gifford, D. K., and Jaakkola, T. S. Learning Population-Level Diffusions with Generative Recurrent Networks. In *Proceedings of the 33rd International Conference on Machine Learning*, pp. 2417–2426, 2016.

Hendriks, G.-J., Jung, L. A., Larsson, A. J. M., Lidschreiber, M., Andersson Forsman, O., Lidschreiber, K., Cramer, P., and Sandberg, R. NASC-seq monitors RNA synthesis in single cells. *Nature Communications*, 10(1):3138, 2019. ISSN 2041-1723. doi: 10.1038/s41467-019-11028-9.

Jambulapati, A., Sidford, A., and Tian, K. A Direct $\tilde{O}(1/\epsilon)$ Iteration Parallel Algorithm for Optimal Transport. In *Advances in Neural Information Processing Systems 32*, pp. 11359–11370, 2019.

Kanton, S., Boyle, M. J., He, Z., Santel, M., Weigert, A., Sanchís-Calleja, F., Guijarro, P., Sidow, L., Fleck, J. S., Han, D., Qian, Z., Heide, M., Huttner, W. B., Khaitovich, P., Pääbo, S., Treutlein, B., and Camp, J. G. Organoid single-cell genomic atlas uncovers human-specific features of brain development. *Nature*, 574(7778):418–422, 2019.

Kantorovich, L. V. On the Translocation of Masses. *Doklady Akademii Nauk*, 1942.

Katayama, Y., Nishiyama, M., Shoji, H., Ohkawa, Y., Kawamura, A., Sato, T., Suyama, M., Takumi, T., Miyakawa, T., and Nakayama, K. I. CHD8 haploinsufficiency results in autistic-like phenotypes in mice. *Nature*, 537(7622): 675–679, 2016.

Kingma, D. P. and Ba, J. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations*, 2015.

Kingma, D. P., Salimans, T., Jozefowicz, R., Chen, X., Sutskever, I., and Welling, M. Improving Variational Inference with Inverse Autoregressive Flow. In *Advances in Neural Information Processing Systems 29*, 2016.

Kolouri, S., Nadjahi, K., Simsekli, U., Badeau, R., and Rohde, G. Generalized Sliced Wasserstein Distances. In *Advances in Neural Information Processing Systems 32*, pp. 261–272, 2019.

La Manno, G., Soldatov, R., Zeisel, A., Braun, E., Hochgerner, H., Petukhov, V., Lidschreiber, K., Kastriti, M. E., Lönnerberg, P., Furlan, A., Fan, J., Borm, L. E., Liu, Z., van Bruggen, D., Guo, J., He, X., Barker, R., Sundström, E., Castelo-Branco, G., Cramer, P., Adameyko, I., Linnarsson, S., and Kharchenko, P. V. RNA velocity of single cells. *Nature*, 560(7719):494–498, 2018.

Lederer, A. R. and La Manno, G. The emergence and promise of single-cell temporal-omics approaches. *Current Opinion in Biotechnology*, 63:70–78, 2020.

Li, X., Wong, T.-K. L., Chen, R. T. Q., and Duvenaud, D. Scalable Gradients for Stochastic Differential Equations. *Artificial Intelligence and Statistics*, 2020.

Liero, M., Mielke, A., and Savaré, G. Optimal Entropy-Transport problems and a new Hellinger–Kantorovich distance between positive measures. *Inventiones mathematicae*, 211(3):969–1117, 2018.

Lindenbaum, O., Stanley, J., Wolf, G., and Krishnaswamy, S. Geometry Based Data Generation. In *Advances in Neural Infromation Processing Systems 31*, pp. 1400–1411, 2018.

Macosko, E. Z., Basu, A., Satija, R., Nemesh, J., Shekhar, K., Goldman, M., Tirosh, I., Bialas, A. R., Kamitaki, N., Martersteck, E. M., Trombetta, J. J., Weitz, D. A., Sanes, J. R., Shalek, A. K., Regev, A., and McCarroll, S. A. Highly Parallel Genome-wide Expression Profiling of Individual Cells Using Nanoliter Droplets. *Cell*, 161 (5):1202–1214, 2015.

Monge, G. Mémoire sur la théorie des déblais et des remblais. *Histoire de l'Académie Royale des Science*, 1781.

Moon, K. R., van Dijk, D., Wang, Z., Gigante, S., Burkhardt, D. B., Chen, W. S., Yim, K., van den Elzen, A., Hirn, M. J., Coifman, R. R., Ivanova, N. B., Wolf, G., and Krishnaswamy, S. Visualizing structure and transitions in high-dimensional biological data. *Nature Biotechnology*, 37(12):1482–1492, 2019.

Muzellec, B. and Cuturi, M. Subspace Detours: Building Transport Plans that are Optimal on Subspace Projections. In *Advances in Neural Information Processing Systems 32*, pp. 6917–6928, 2019.

Oeppen, J. and Vaupel, J. W. Broken Limits to Life Expectancy. *Science*, 296(5570):1029–1031, 2002.

Papadakis, N., Peyré, G., and Oudet, E. Optimal Transport with Proximal Splitting. *SIAM Journal on Imaging Sciences*, 7(1):212–238, 2014.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked Autoregressive Flow for Density Estimation. In *Advances in Neural Information Processing Systems 30*, pp. 2338–2347, 2017.

Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Kopf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., and Chintala, S. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pp. 8026–8037, 2019.

Peyré, G. and Cuturi, M. *Computational Optimal Transport*. arXiv:1803.00567, 2019.

Rezende, D. J. and Mohamed, S. Variational Inference with Normalizing Flows. In *Proceedings of the 32nd International Conference on Machine Learning*, volume 37, pp. 1530–1538, 2015.

Rifai, S., Vincent, P., Muller, X., Glorot, X., and Bengio, Y. Contractive Auto-Encoders: Explicit Invariance During Feature Extraction. In *Proceedings of the 29th International Conference on Machine Learning*, pp. 833–840, 2011.

Saelens, W., Cannoodt, R., Todorov, H., and Saeys, Y. A comparison of single-cell trajectory inference methods. *Nature Biotechnology*, 37(5):547–554, 2019.

Schiebinger, G., Shu, J., Tabaka, M., Cleary, B., Subramanian, V., Solomon, A., Gould, J., Liu, S., Lin, S., Berube, P., Lee, L., Chen, J., Brumbaugh, J., Rigollet, P., Hochedlinger, K., Jaenisch, R., Regev, A., and Lander, E. S. Optimal-Transport Analysis of Single-Cell Gene Expression Identifies Developmental Trajectories in Reprogramming. *Cell*, 176(4):928–943.e22, 2019.

Sinkhorn, R. A relationship between arbitrary positive matrices and doubly stochastic matrices, 1964.

Trapnell, C., Cacchiarelli, D., Grimsby, J., Pokharel, P., Li, S., Morse, M., Lennon, N. J., Livak, K. J., Mikkelsen, T. S., and Rinn, J. L. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature Biotechnology*, 32(4):381–386, 2014.

Villani, C. *Optimal Transport, Old and New*. Springer, June 2008. ISBN 978-3-540-71050-9.

Vincent, P., Larochelle, H., Lajoie, I., Bengio, Y., and Manzagol, P.-A. Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion. *Journal of Machine Learning Research*, pp. 3371–3408, 2010.

Waddington, C. H. The epigenotype. *Endeavour*, 1:18–20, 1942.

Weed, J. and Bach, F. Sharp asymptotic and finite-sample rates of convergence of empirical measures in Wasserstein distance. *Bernoulli*, 25(4A):2620–2648, 2019.

Weinreb, C., Wolock, S., Tusi, B. K., Socolovsky, M., and Klein, A. M. Fundamental limits on dynamic inference from single-cell snapshots. *Proceedings of the National Academy of Sciences*, 115(10):E2467–E2476, 2018.

Yang, K. D. and Uhler, C. Scalable Unbalanced Optimal Transport Using Generative Adversarial Networks. In *7th International Conference on Learning Representations*, pp. 20, 2019.

Supplement

A. Technical details

A.1. Proof of Theorem 4.1

First, we apply the Lagrange multiplier method by introducing the variable λ to the minimization problem of (5) subject to constraints (4). As we always begin with the base distribution and at any time t, x is defined by $f(x,t)$ and the initial value $x(t_0)=x_0$, which has $\mathrm{KL}(P(t_0,\cdot)\parallel\mu)=0$.

$$\inf_{(P,f)}\sup_{\lambda}(t_1-t_0)\mathop{\mathbb{E}}_{x_0\sim\mu}\int_{t_0}^{t_1}P(x,t)\left|f(x,t)\right|^2 dt + \lambda\mathrm{KL}(P(t_1,\cdot)\parallel\nu) \tag{14}$$

The expectation part is equivalent to $\int_{\mathbb{R}^d}\int_{t_0}^{t_1}P(x,t)\left|f(x,t)\right|^2 dtdx$, and we use interchangeably in the the remaining of the proof. Since the KL divergences are non-negative, $\lambda\geq 0$. The optimal solution of the min-max problem is the optimal solution to the original problem. Consider the true minimal loss given by the optimal solution to be c, we know that $L(\lambda)\leq c$, where

$$L(\lambda)=\sup_{\lambda\geq 0}\inf_{(P,f)}(t_1-t_0)\mathop{\mathbb{E}}_{x_0\sim\mu}\int_{t_0}^{t_1}P(x,t)\left|f(x,t)\right|^2 dt + \lambda\mathrm{KL}(P(t_1,\cdot)\parallel\nu) \tag{15}$$

In order to show that the solution of the max-min problem converges to c, we first show that it is monotonic in λ. For easier reading, set $E=\int_{\mathbb{R}^d}\int_{t_0}^{t_1}P(x,t)\left|f(x,t)\right|^2 dtdx$, $M=\mathrm{KL}(P(t_1,\cdot)\parallel\nu)$. Both E and M are functions of f. For any pair of E,M values, if $\lambda_1>\lambda_2$, $E+\lambda_1 M> E+\lambda_2 M$. Thus the maximum and minimum of the function $L=E+\lambda M$ is also monotonic in λ, and it will converge to the supremum.

Next, we show that the divergence term $M(f)$ decreases monotonically as λ increases, and it converges to 0 as λ goes to infinity. For a given λ, let $f^*_\lambda=\arg\inf E(f)+\lambda M(f)$. By definition, $E(f^*_{\lambda_1})+\lambda_1 M(f^*_{\lambda_1})\leq E(f^*_{\lambda_2})+\lambda_1 M(f^*_{\lambda_2})$, and $E(f^*_{\lambda_1})+\lambda_2 M(f^*_{\lambda_1})\geq E(f^*_{\lambda_2})+\lambda_2 M(f^*_{\lambda_2})$. Thus $(\lambda_1-\lambda_2)(M(f^*_{\lambda_1})-M(f^*_{\lambda_2}))\leq 0$. If $\lambda_1>\lambda_2$,then $M(f^*_{\lambda_1})-M(f^*_{\lambda_2})\leq 0$. The sequence $M(f)$ decreases monotonically as λ increases. Because $L(\lambda)$ is upper bounded by c and $M(f)\geq 0$, $M(f)$ converges to zero as λ goes to infinity.

Now we have shown that $L(\lambda)$ is a monotone sequence and is upper bounded by c, and that the divergence term M converges to zero, we next show that L converges to c, and that the optimal solution of the max-min problem in (15) is the optimal solution of the original problem. Since the divergence term is non-negative, we have a lower bound for $L(\lambda)$ as

$$\forall\lambda,\quad L(\lambda)\geq\inf E(f) \quad \text{s.t.}\quad M(f)\leq M(f^*_\lambda) \tag{16}$$

Because $L(\lambda)$ is monotonically increasing, and $M(f)$ is monotonically decreasing, $H(\lambda)=\inf E(f)$ increases as λ increases.

Lemma A.1.

$$\forall\epsilon>0,\quad\forall f,\quad s.t.\quad M(f)\leq\epsilon,$$
$$\exists\hat{f},\quad s.t.\quad M(\hat{f})=0,\quad and$$
$$E(\hat{f})-E(f)\leq\frac{D^2}{\sqrt{2T}}\sqrt[4]{\epsilon}(1+\sqrt[4]{\epsilon})$$

where D is the diameter of the probability space and T is the transformation completion time.

Proof. For a certain λ, starting from the base distribution μ, at time T the distribution is transformed, by following f, to ν', and $\mathrm{KL}(\nu'\parallel\nu)=\epsilon$. Now consider a different transformation $\hat{f}$, which is composed of two part: the first part is an accelerated f, so that ν' is achieved by time $T/(1+\xi)$, and the second part is transforming ν' to ν in the remaining time of $\frac{\xi T}{(1+\xi)}$. Thus at time T, by following $\hat{f}$, we achieve zero divergence, $M(\hat{f})=0$. The new transformation $\hat{f}$ has an increased E, from the acceleration and the additional transformation.

$$E(\hat{f})=\frac{T}{1+\xi}\int_{\mathbb{R}^d}\int_0^{\frac{T}{1+\xi}}P\left(x,(1+\xi)\,t\right)|(1+\xi)f(x,(1+\xi)t)|^2dtdx + \int_Z\int_Y\int_{\frac{T}{1+\xi}}^{T}\mathcal{M}(y,z)\frac{|z-y|^2}{(\frac{\xi}{1+\xi}T)^2}dtdydz \tag{17}$$

where Z has distribution ν, Y has distribution ν', and $\mathcal{M}(y,z)$ is a mapping from Y to Z. The first part of $E(\hat{f})$ is just $E(f)$. The second part is upper bounded by $\frac{TV(Y,Z)D^2}{\frac{\xi}{1+\xi}T}$, where $TV(Y,Z)$ is the total variation between Y and Z, which is in turn upper bounded by $\sqrt{\epsilon/2}$. We choose $\xi=\sqrt[4]{\epsilon}$. □

By definition, $E(\hat{f})\geq c$, as c is the infimum at zero divergence. Assuming $L(\lambda)$ converges to $c-\alpha$, and

$\alpha > 0$, $c - \alpha \geq E(f)$. Then by Lemma A.1, $\forall \epsilon$, $\alpha < \frac{D^2}{\sqrt{2T}} \sqrt[4]{\epsilon}(1 + \sqrt[4]{\epsilon})$, and we have a contradiction. Now we complete the proof that $L(\lambda)$ converges to c and that for a large enough λ, the solution of the max-min problem (15) is the solution of (5) when subject to conditions (4).

B. Growth Rate Model Training

Our growth network $G(x, t)$ is trained to match the discrete unbalanced optimal transport problem with entropic regularization:

$$\begin{aligned} \gamma = \operatorname{argmin}_\gamma < \gamma, M >_F &+ \lambda\big(\sum_{i,j} \gamma_{i,j} \log(\gamma_{i,j})\big) \\ &+ \alpha KL(\gamma 1, \mu) + \beta KL(\gamma^T 1, \nu) \\ \text{s.t. } \gamma \geq 0 & \end{aligned} \tag{18}$$

Where $< \cdot, \cdot >_F$ is the Frobenius norm of elementwise matrix multiplication of the transportation matrix γ and the cost matrix M, and where λ, α, β are regularization constants > 0 on the source and target unbalanced distributions μ, ν. Then the growth rate of each cell i in μ to ν is then

$$g_i = \gamma_{(i,\cdot)} 1 \tag{19}$$

In our experiments we set $\lambda = 0.1, \beta = 10000$ and tune α for reasonable growth rates. This gives a growth rate at every observed cell; however, our model needs a growth rate defined continuously at every measured timepoint. For this, we learn a neural network that is trained to match the growth rate at measured cells and equal to one at negative sampled points. We use a simple form of negative sampling, for each batch of real points we sample an equal sized batch of points from a uniform distribution over the [-1,1] hypercube, where these negative points are given a growth rate value of 1. The network is trained with mean squared error loss to match these growth rates at all measured times.

C. Scaling with Dimension

Runtime Considerations. Existing numerical methods for solving dynamic OT rely on proximal splitting methods over a discretized staggered grid (Benamou & Brenier, 2000; Papadakis et al., 2014; Peyré & Cuturi, 2019). This results in a non-smooth but convex optimization problem over these grid points. However, the number of grid points scales exponentially with the dimension, so these methods are only applicable in low dimensions. TrajectoryNet scales polynomially with the dimension. See Figure S1 for empirical measurements.

To test the computation time with dimension we run TrajectoryNet for 100 batches of 1000 points on the mouse cortex dataset over different dimensionalities. For hardware we use a single machine with An AMD Ryzen Threadripper 2990WX 32-core Processor, 128GB of memory, and three Nvidia TITAN RTX GPUs. Our model is coded in the Pytorch framework (Paszke et al., 2019). We count the total number of function evaluations (both forward and backward) divide the total time by this. In Figure S1, you can see the seconds per evaluation is roughly linear with the dimensionality of the data. This does not imply convergence of the model is linear in dimension, only that computation per iteration is linear. As suggested in Grathwohl et al. (2019), number of iterations until convergence is a function of how complicated the distributions are, and less dependent on the ambient dimension itself. By learning flows along a manifold with $L_{density}$, our method may scale closer to the intrinsic dimensionality of the data rather than the ambient.

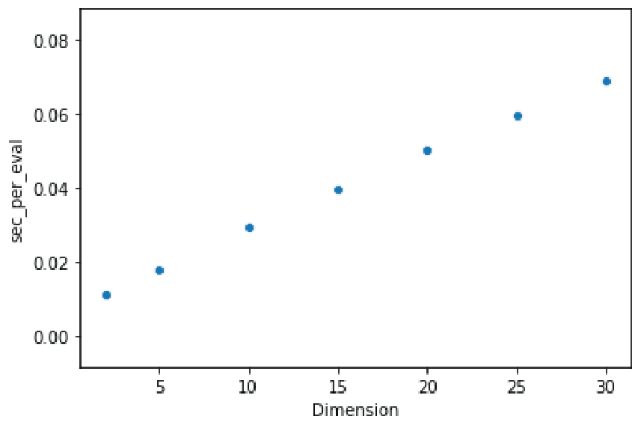

Figure S1. The computation per evaluation is roughly linear in terms of dimension.

D. Biological Considerations

Quality control and normalization is important when estimating RNA-velocity from existing single cell measurements. We suspect that the RNA-velocity measurements from the Embryoid body data may be suspect given the low number of unspliced RNA counts present. In Figure S3 we can see that each timepoint consists of around 10%-20% of unspliced RNA. This is relatively low relative to numbers in other recent works (La Manno et al., 2018; Bergen et al., 2019; Kanton et al., 2019). Low unspliced RNA counts leads to more noise in the estimates of RNA velocity and lower quality.

In Figure 8 we showed how TrajectoryNet can be projected back to the gene space. These projections can be used to infer the differences much earlier in time than they can be identified in the gene space. Here we have four populations that are easily identified by marker genes or clustering at the final timepoint. Since all four populations emerge from a single relatively uniform stem cell population, the question

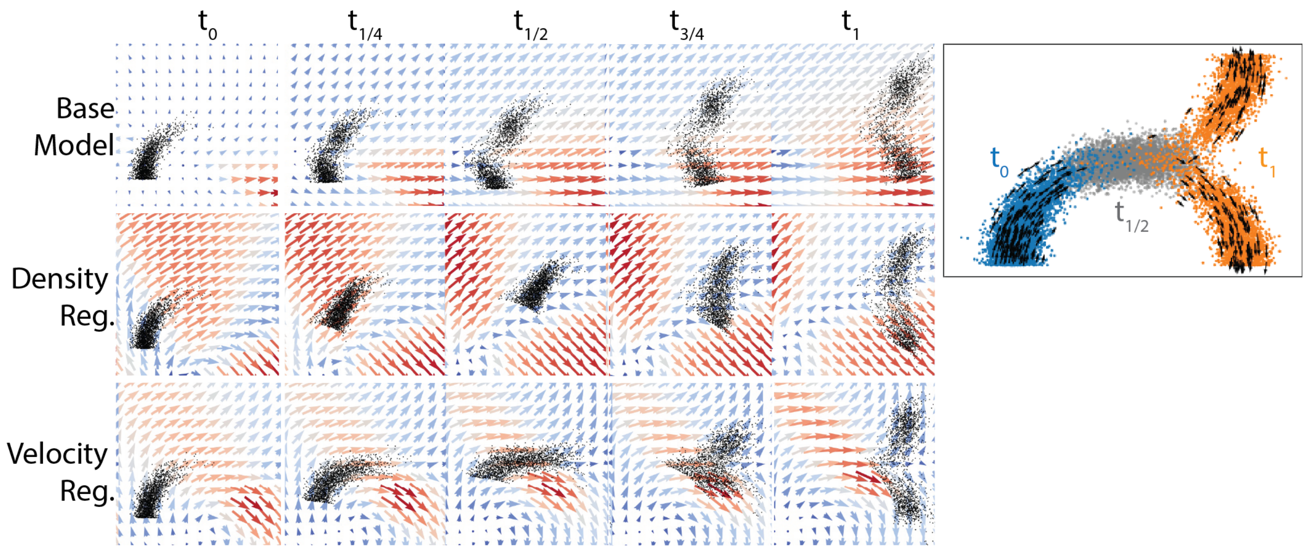

Figure S2. Density regularization or velocity regularization can be used to follow a 1D manifold in 2D.

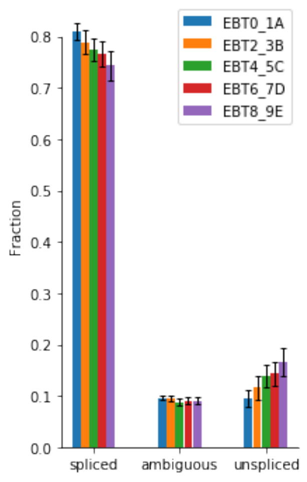

Figure S3. Shows the ratio of spliced, ambiguous, and unspliced RNA counts over the 5 timepoints in the Embryoid body dataset. Mean unspliced here is around 10%-20% of total counts, in other systems this is near 30% (La Manno et al., 2018).

becomes how early can we identify the features of progenitor cells, the cells leading to these differentiated populations. Since TrajectoryNet models cells as probabilities continuously over time, we can find the path for each differentiated cell in earlier timepoints. This allows inferences such as the fact that HAND1, a gene that is generally high in cardiac cells, is high at earlier timepoints, and may even start to distinguish the population as early as day 6. A gene like ONECUT2 is only starts to distinguish neuronal populations at later timepoints. For further information on this particular system see (Moon et al., 2019) Figure 6.

E. Reproducibility

To foster reproducibility, we provide as many details as possible on the experiments in the main paper. Code is available at `http://github.com/krishnaswamylab/TrajectoryNet`.

E.1. 2D Examples

In Figure 2 we transport a Gaussian to an s-curve. The Gaussian consists of 10000 points sampled from a standard normal distribution. The s-curve is generated using the sklearn function `sklearn.datasets.make_s_curve` with noise of 0.05, and 10000 samples. We then take the first and third dimension, and multiply by 1.5 for the proper scaling. To generate the OT subplot we used the Mccann interpolant from 200 points sampled from the Gaussian. To generate panel (d), we used the procedure detailed in the beginning of Section 5 to train TrajectoryNet, then sampled 200 points from a Gaussian distribution and used the adjoint with these points as the initial state at time t_0 to generate points at time t_1. For panel (e) we added an energy regularization with $\lambda_e = 0.1$ and $\lambda_j = 1$. These were found by experimentation, although parameters in the range of $\lambda_e = [0.01, 1]$ and $\lambda_j = [0.1, 1]$ were largely visually similar.

To generate the arch and tree datasets we started with two half Gaussians $\mathcal{N}(\cdot, \frac{1}{2\pi})$ at mean zero and one (as pictured in Figure 4) with 5000 points each, then found the Mccann interpolant at $t_{1/2}$ as the test distribution. We then lift these into 2d by embedding on the half circle of radius 1 and adding noise $\mathcal{N}(0, 0.1)$ to the radius. To generate velocity, we add a velocity tangent to the circle for each point. For the tree dataset we additionally flip (randomly) half of the points with $x > 1$ over the line $y = 1$.

For the Cycle dataset, we start with 5000 uniformly sampled points around the circle, with radius as $\mathcal{N}(1, 0.1)$ We then add an arrow tangent to the circle with magnitude $\pi/5$, Thus in one time unit the points should move $1/10$ of the way around the circle.

E.2. Single Cell Datasets

Both single cell datasets were sequenced using 10X sequencing. The Embryoid body data can be found here[3] and consists of roughly 30,000 cells unfiltered, and 16,000 cells after filtering. The mouse cortex dataset is not currently publicly available, but consists of roughly 20,000 cells after filtering. For both datasets no batch correction was used. Raw sequences were processed with CellRanger. We then used velocyto (La Manno et al., 2018) to produce the unspliced and spliced count matrices. We then used the default parameters in ScVelo (Bergen et al., 2019) to generate velocity arrows on a PCA embedding. These include count normalization across rows, selection of the 3000 most variable genes, filtering of low quality genes, and smoothing counts between cells.

For parameters we did a grid search over $\lambda_{density} \in \{0, 0.1, 0.01\}$, $\lambda_{velocity} \in \{0, 0.001, 0.0001\}$. For Base+E we did a search of $\lambda_{energy} \in \{1.0, 0.1, 0.01\}$, A more extensive search could lead to better results. We intended to show how these regularizations can be used and demonstrate the viability of this approach rather than fully explore parameter space.

E.3. Software Versioning

The following software versions were used. `scvelo==0.1.24`, `torch==1.3.1`, `torchdiffeq==0.0.1`, `velocyto==0.17.17`, `scprep==1.0.3`, `scipy==1.4.1`, `scikit-learn==0.22`, `scanpy==1.4.5`

[3]https://doi.org/10.17632/v6n743h5ng.1