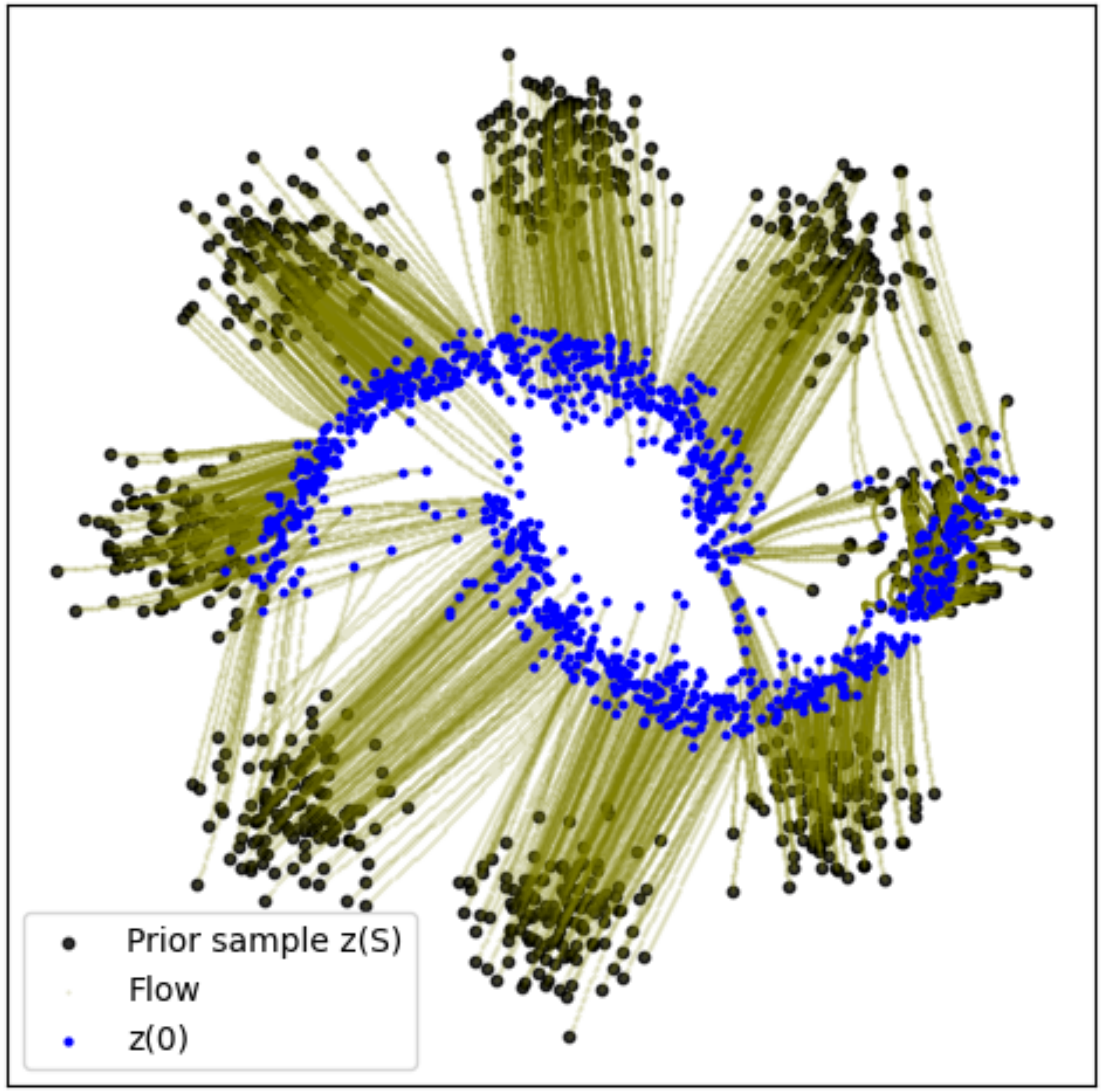
Flow Matching

An alternative approach to generative modelling

Chris Nemeth - 27/11/24

Motivation



How do we transport the mixture of Gaussians distribution to the two the two moons distribution?

Introduction

- This talk is largely based on this great blog post: https://mlg.eng.cam.ac.uk/blog/2024/01/20/flow-matching.html

- Flow matching in a sentence:

  *"Flow matching (FM) combines aspects from **Continuous Normalising Flows** (CNFs) and **Diffusion Models** (DMs), alleviating key issues both methods have."*

- In this talk I'll cover the following:
  - Normalising flow and their continuous-time variants
  - Flow matching and conditional flow matching
  - Some examples of how FM works.

Normalising Flows

- Normalising Flows transform a simple base distribution $q_0(x)$ into a complex distribution $p_1(y)$ using an invertible and continuously differentiable mapping ϕ.

- Using the **change-of-variable rule**, the density $p_1(\cdot)$ is given by:

$$p_1(y) = q_0(\phi^{-1}(y)) \left| \det \frac{\partial \phi^{-1}(y)}{\partial y} \right| .$$

$$= \frac{q_0(x)}{\left| \det \left[\frac{\partial \phi}{\partial x}(x) \right] \right|} \quad \text{with } x = \phi^{-1}(y)$$

- Note that $\phi \circ \phi^{-1} = \mathrm{Id}$.

Normalising Flows: Gaussian Example

- Let's assume that we have a Gaussian $q_0(x) = \mathcal{N}(\mu, \sigma^2)$
- Let's use a linear mapping function $\phi(x) = ax + b$
- Using the change-of-variables formula (or simpler the linear property of Gaussians):

$$p_1(y) = \mathcal{N}(y; a\mu + b, a^2\sigma^2)$$

Normalising Flows

- Naturally, if we want to use normalising flows for generative modelling, then the mapping ϕ **has to be sufficiently complex**, i.e. a neural network, which we parameterise by θ, where now ϕ_θ.

- We can estimate θ using maximum likelihood estimation,

$$\text{argmax}_\theta \ \mathbb{E}_{x \sim D}[\log p_1(x)]$$

- If ϕ_θ is a neural net, then for normalising flows, how do we ensure that ϕ_θ is **invertible, computable** and its **Jacobian is computable**?

Normalising Flows

- **Residual flow:**

$$\phi_k(x) = x + \delta u_k(x)$$

where $u_k(x)$ is a neural network (which apparently has a similar structure to a *residual network).* This is one choice of transformation which people seem to think balances between expressivity and computability.

- We can then stack these transformations to create a *flow:*

$$\phi = \phi_k \circ \ldots \circ \phi_2 \circ \phi_1$$

- **Model log-likelihood:**

$$\log q(y) = \log p(\phi^{-1}(y)) + \sum_{k=1}^{K} \log \det \left[\frac{\partial \phi_k^{-1}}{\partial x_{k+1}}(x_{k+1}) \right], \text{with } x_k = \phi_K^{-1} \circ \ldots, \circ \phi_k^{-1}(y)$$

Continuous-Time Normalising Flows

- If our flow is defined as $\phi(x) = x + \delta u(x)$ for some $\delta > 0$, then we can rearrange to get

$$\frac{\phi(x) - x}{\delta} = u(x)$$

- If we set $\delta = 1/K$ and let $K \to \infty$, then the flow $\phi_K \circ \ldots, \circ \phi_2 \circ \phi_1$ is given by the ODE:

$$\frac{\mathrm{d}x_t}{\mathrm{d}t} = \lim_{\delta \to 0} \frac{x_{t+\delta} - x_t}{\delta} = \frac{\phi_t(x_t) - x_t}{\delta} = u_t(x_t).$$

Continuous-Time Normalising Flows

- The *flow ODE* $\phi_t : [0,1] \times \mathbb{R}^d \to \mathbb{R}^d$ is defined s.t.

$$\frac{\mathrm{d}\phi_t}{\mathrm{d}t} = u_t(\phi_t(x_0)).$$

- In other words, ϕ_t maps the initial condition x_0 to the ODE solution at time t

$$x_t := \phi_t(x_0) = x_0 + \int_0^t u_s(x_s)\mathrm{d}s.$$

- So, we have the mapping $\phi_t(\,\cdot\,)$, but for a normalising flow we also need $\log\det$ of the Jacobian!

Continuous-Time Normalising Flows

- We need to find the density p_t for ϕ_t (or equivalently u_t), which we can get via the **Transport Equation**

$$\frac{\partial}{\partial_t} p_t(x_t) = -\Big(\nabla \cdot u_t p_t\Big)(x_t)$$

- The *total derivative* in log-space (after some manipulation of the above)

$$\frac{\mathrm{d}}{\mathrm{d}t} = \log p_t(x_t) = -(\nabla \cdot u_t)(x_t)$$

which leads to the log-density

$$\log p_t = \log p_0(x_0) - \int_0^t (\nabla \cdot u_s)(x_s)\mathrm{d}s$$

Continuous-Time Normalising Flows

- Recall, the log-density is

$$\log p_t = \log p_0(x_0) - \int_0^t (\nabla \cdot u_s)(x_s)\mathrm{d}s$$

however we don't have access to the *vector field* u_t, but we can approximate it with a *neural network* $u_\theta : \mathbb{R}_+ \times \mathbb{R}^d \to \mathbb{R}^d$, to produce a parametric model

$$\log p_\theta(x) := \log p_1(x) = \log p_0(x_0) - \int_0^1 (\nabla \cdot u_\theta)(x_t)\mathrm{d}t.$$

- We can now use an *ODE solver* to estimate x_t and $\log p_t$ by solving

$$\frac{\mathrm{d}}{\mathrm{d}t} = \begin{pmatrix} x_t \\ \log p_t \end{pmatrix} = \begin{pmatrix} u_\theta(t, x_t) \\ -\mathrm{div}\ u_\theta(t, x_t) \end{pmatrix}$$

Discrete Vs. Continuous Time

- Why should we use a continuous normalising flow?

    1. Continuous normalising flows (CNFs) automatically determine the number of residual flow steps K via an adaptive solver, using an error threshold ϵ to set the discretisation step size δ, where $K = 1/\delta$. Unlike residual flows, where distinct parameters θ_k are used for each layer, CNFs share parameters over time t.

    2. In residual flows, training requires ensuring u_θ is $1/\delta$-Lipschitz to maintain invertibility, posing strict constraints. CNFs only require $u_\theta(t, x)$ to be Lipschitz in x, with no specific constant, making this condition easier to enforce in neural architectures.

Training CNFs

- We can learn the parameters of the CNF using maximum likelihood estimation.

$$\mathscr{L}(\theta) = \mathbb{E}_{x \sim q_1} \left[\log p_1(x) \right],$$

where the expectation is taken over the data distribution and p_1 is the parametric model.

- Calculating the log-likelihood requires integrating the time-evolution for samples x_t and log-likelihood $\log p_t$, which both depend on u_θ. This is requires expensive numerical ODEs!

- Can we train the CNF without the ODE integration?

Flow Matching

- Flow matching is a way of training a CNF by formulating the problem as a regression objective, wrt a parametric vector field u_θ.

$$\mathscr{L}(\theta) = \mathbb{E}_{t \sim \mathscr{U}[0,1]} \mathbb{E}_{x \sim p_t} \left[\| u_\theta(t, x) - u(t, x) \|^2 \right],$$

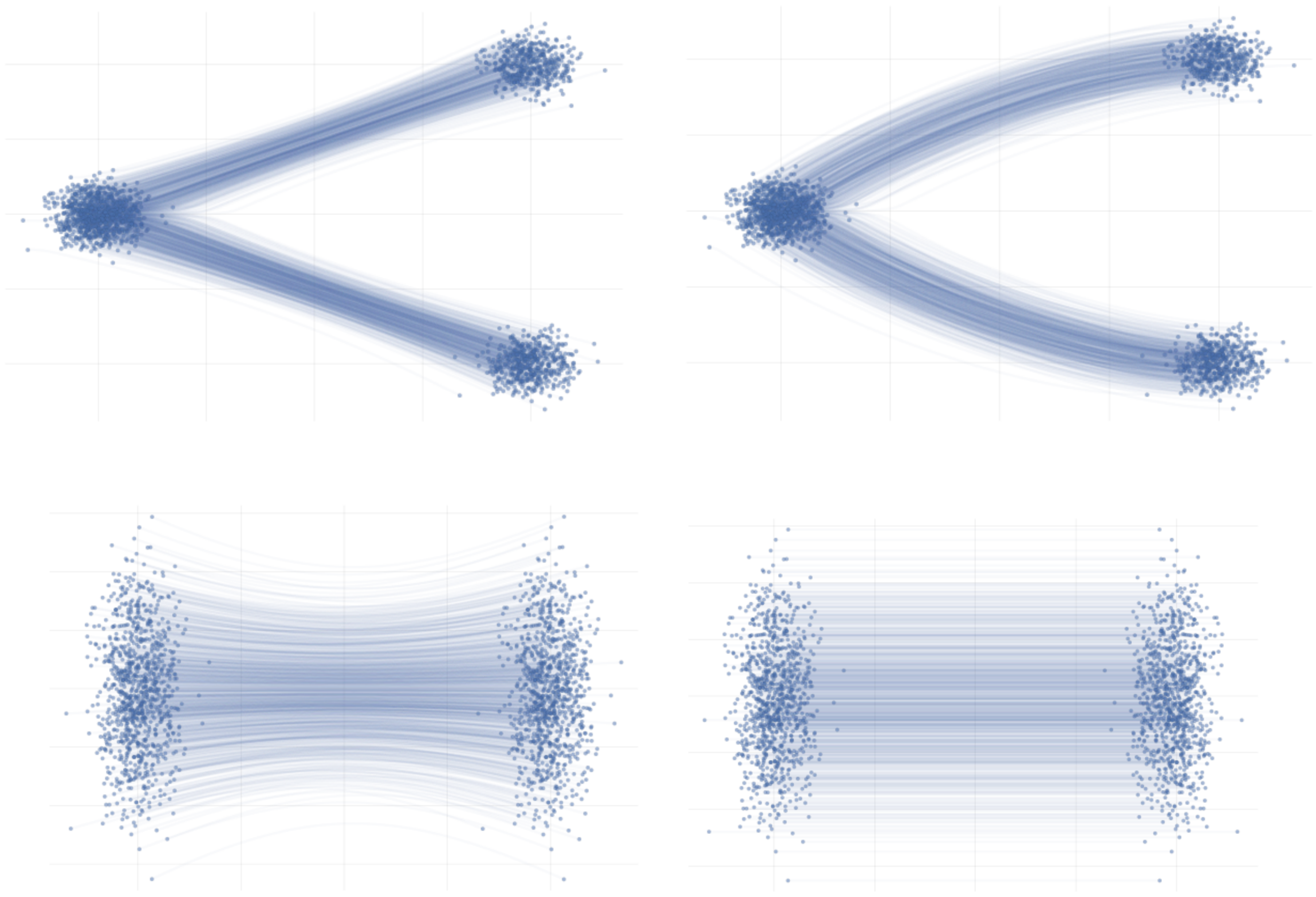
where $u(t, x)$ is the vector field which creates the probability path p_t that interpolates between p_0 and p_1, i.e.

$$\log(p_1) = \log(p_0) - \int_0^1 (\nabla \cdot u_t)(x) \mathrm{d}t$$

- Of course solving the above regression requires access to $u(t, x)$. So how do we estimate u_θ?

Flow Matching

- Obviously, we require a *"valid"* $u(t, x)$, but there is no *unique* vector field for mapping p_0 to p_1.



Conditional Flows

- Recall that the transport equation relates the vector field u_t to the *probability path* p_t

$$\frac{\partial p_t(x)}{\partial t} = -\nabla \cdot \big(u_t(x)p_t(x)\big)$$

which means that finding u_t or p_t is equivalent.

- We can express the probability path p_t as the marginal over a latent variable z

$$p_t(x) = \int p(z)p_{t|z}(x_t \mid z)\mathrm{d}z$$

where $p_{t|z}(x \mid z)$ is called the **conditional probability path**, which satisfies some boundary conditions $t = 0$ and $t = 1$ so that p_t is valid interpolation between q_0 and q_1

Conditional Flows

- We have access to data samples $x_1 \sim q_1$ so let's just set $z = x_1$, which gives

$$p_t(x) = \int q(x_1) p_{t|1}(x_t \,|\, x_1) \mathrm{d}x_1$$

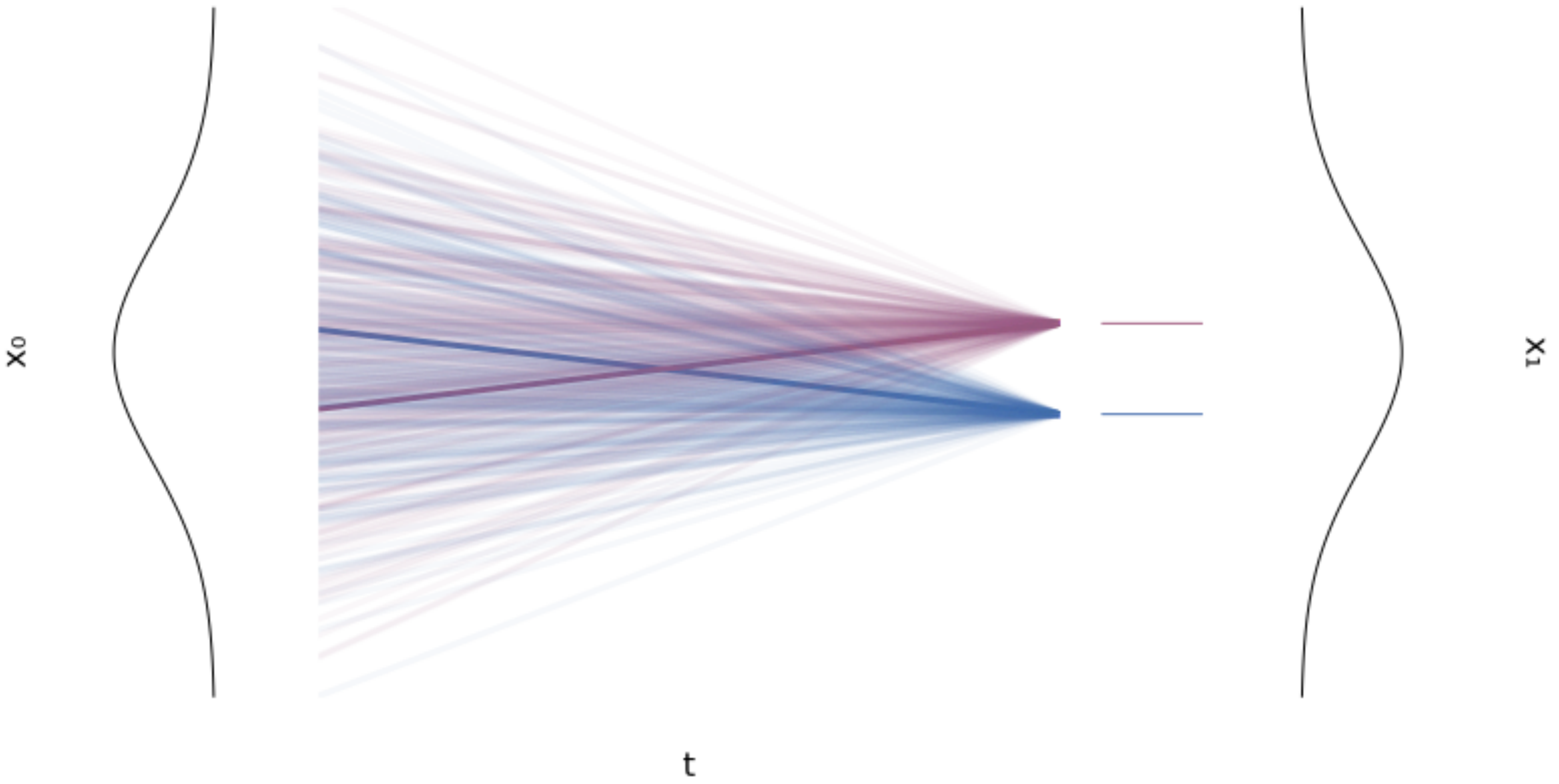
- In this set-up, the conditional probability path $p_{t|1}$ has to satisfy the boundary conditions

$$p_0(x \,|\, x_1) = p_0 \quad \text{and}$$

$$p_1(x \,|\, x_1) = \mathcal{N}(x \,|\, x_1, \sigma_{\min}^2 \mathrm{I}) \to \delta_{x_1}(x) \text{ as } \sigma_{\min} \to 0$$

- Typically, we would choose $p_0(x) = \mathcal{N}(x; 0, \mathrm{I})$.

Continuous Flows: Gaussian Example



Conditional Flows

- The **conditional probability path** also satisfies a *transport equation*

$$\frac{\partial p_t(x \mid x_1)}{\partial t} = -\nabla\left(u_t(x \mid x_1)p_t(x \mid x_1)\right)$$

*w*here $u_t(x \mid x_1)$ is the **conditional vector field.**

- But we really want the marginal vector field $u_t(x)$, which we now have

$$\begin{aligned} u_t(x) &= \mathbb{E}_{x_1 \sim p_{1|t}}[u_t(x \mid x_1)] \\ &= \int u_t(x \mid x_1)\frac{p_t(x \mid x_1)q_1(x_1)}{p_t(x)}\mathrm{d}x_1 \end{aligned}$$

Flow Matching and Conditional Flow Matching

- Recall the flow matching objective:

$$\mathscr{L}_{\mathrm{FM}}(\theta) = \mathbb{E}_{t \sim \mathscr{U}[0,1], x \sim p_t}[\|u_\theta(t,x) - u(t,x)\|^2]$$

- We can use the *conditional vector field* $u_t(x \mid x_1)$ and marginalise x_1

$$\mathscr{L}_{\mathrm{CFM}}(\theta) = \mathbb{E}_{t \sim \mathscr{U}[0,1], x_1 \sim q, x_t \sim p_t(x|x_1)}[\|u_\theta(t,x) - u_t(x \mid x_1)\|^2]$$

- These loss functions are equivalent in that $\nabla_\theta \mathscr{L}_{\mathrm{FM}}(\theta) = \nabla_\theta \mathscr{L}_{\mathrm{CFM}}(\theta)$

- Note that in CNFs there is no enforced preference over the vector field u_t, whereas in CFM the vector field is dependent on on the conditional vector field.

Continuous Flows: Gaussian Example

- We need to choose a probability path $p_t(x \mid x_1)$ and a conditional vector field $u_t(x \mid x_1)$.

- To make things simple, we can choose

$$p_t(x \mid x_1) = \mathcal{N}(\mu_t(x_1), \sigma_t(x_1)^2 \mathrm{I})$$

- We set $\mu_0(x_1) = 0$ and $\sigma_0(x_1)^2 = 1$, so all probability paths converge to $p(x) = \mathcal{N}(x \mid 0, \mathrm{I})$ at $t = 0$. We also set, $\mu_1(x_1) = x_1$ and $\sigma_1(x_1) = \sigma_{\min}$ so that $p_1(x \mid x_1)$ is a Gaussian concentrated at x_1.

- A simple choice for the mean μ_t and std σ_t is a **linear interpolation**

Continuous Flows: Gaussian Example

- A simple choice for the mean μ_t and std σ_t is a **linear interpolation**

$$\mu_t(x_1) = t x_1, \quad \sigma_t(x_1) = (1 - t) + t\sigma_{\min}$$
$$\dot{\mu}_t(x_1) = x_1, \quad \dot{\sigma}_t(x_1) = -1 + \sigma_{\min}$$

- Thm. 3 in the paper shows that for this set-up, the **conditional vector field** is

$$u_t(x \mid x_1) = \frac{\dot{\sigma}_t(x_1)}{\sigma_t(x_1)}(x - \mu_t(x_1)) + \dot{\mu}_t(x_1)$$
$$= \frac{x_1 - (1 - \sigma_{\min})x}{1 - (1 - \sigma_{\min})t}$$

Issues with Flow Matching

- The main issue is the *crossing path* phenomenon, which leads to:
    1. Non-straight marginal paths -> ODE is harder to integrate
    2. Many possible x_1 for x_t -> high CFM loss variance

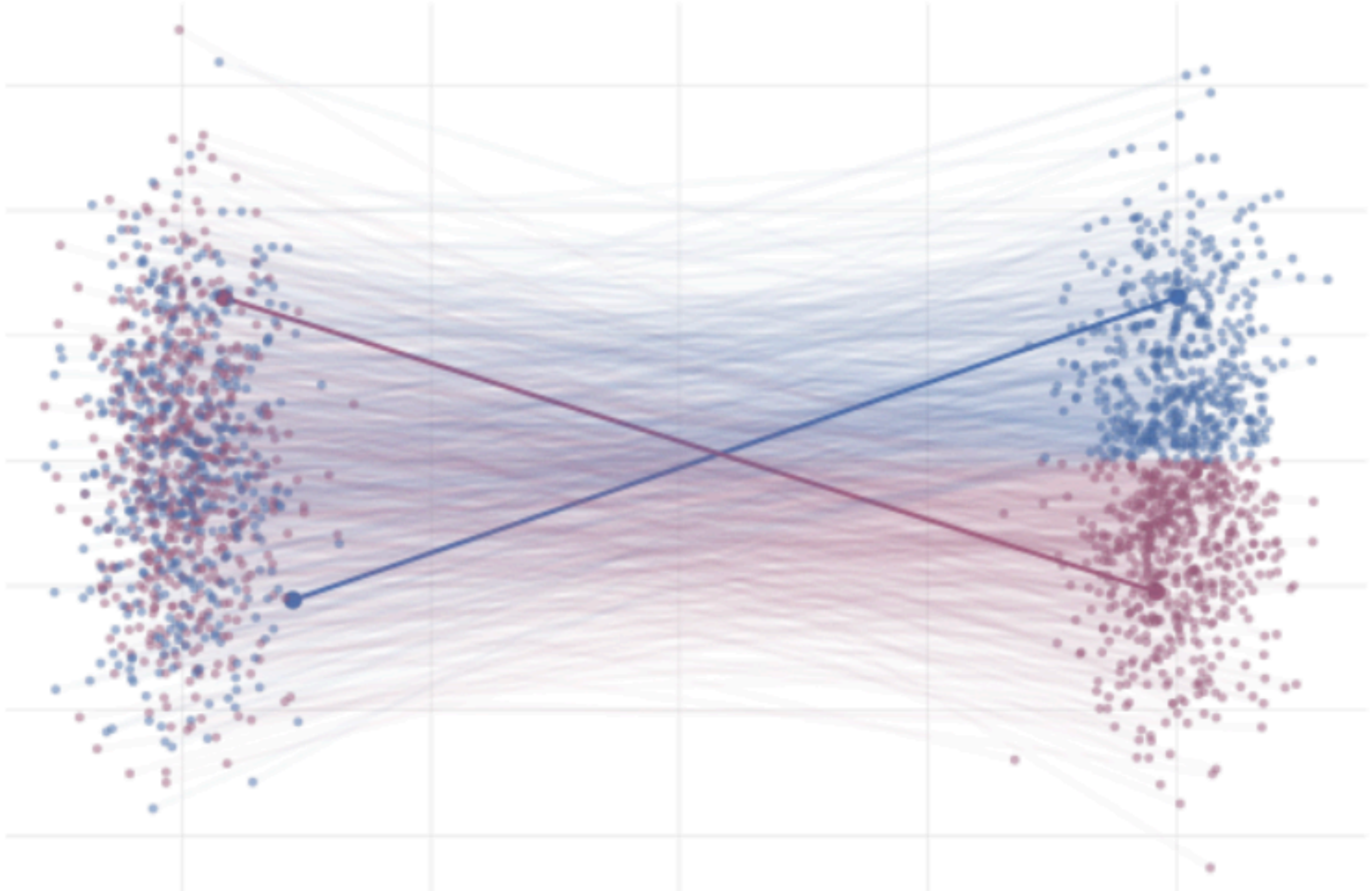


Figure 16: Realizations of conditional paths from $p_0 = p_1 = \mathcal{N}(0, 1)$ for two different $x_1^{(i)}, x_1^{(2)} \sim q$ with conditional vector field given by $u_t(x \mid x_1) = (1-t)x + tx_1$.

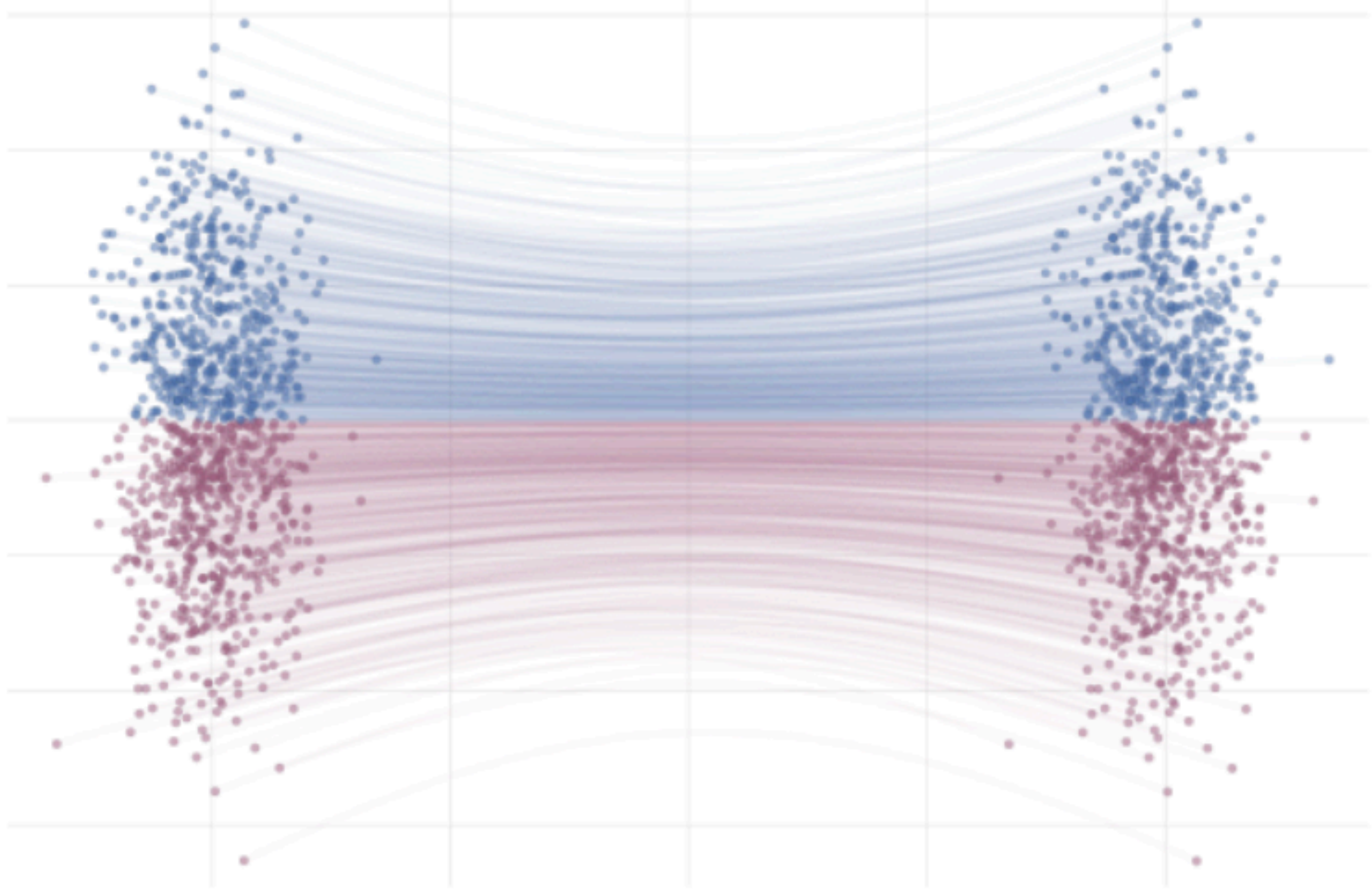


Figure 17: Paths from p_0 to p_1 following the true marginal vector field $u_t(x)$. Paths are highlighted by the sign of the 2nd vector component.

Issues with Flow Matching

Recall our loss function: $\mathscr{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t\sim\mathscr{U}[0,1],x_1\sim q,x_t\sim p_t(x|x_1)}[\|u_\theta(t,x) - u_t(x\,|\,x_1)\|^2]$

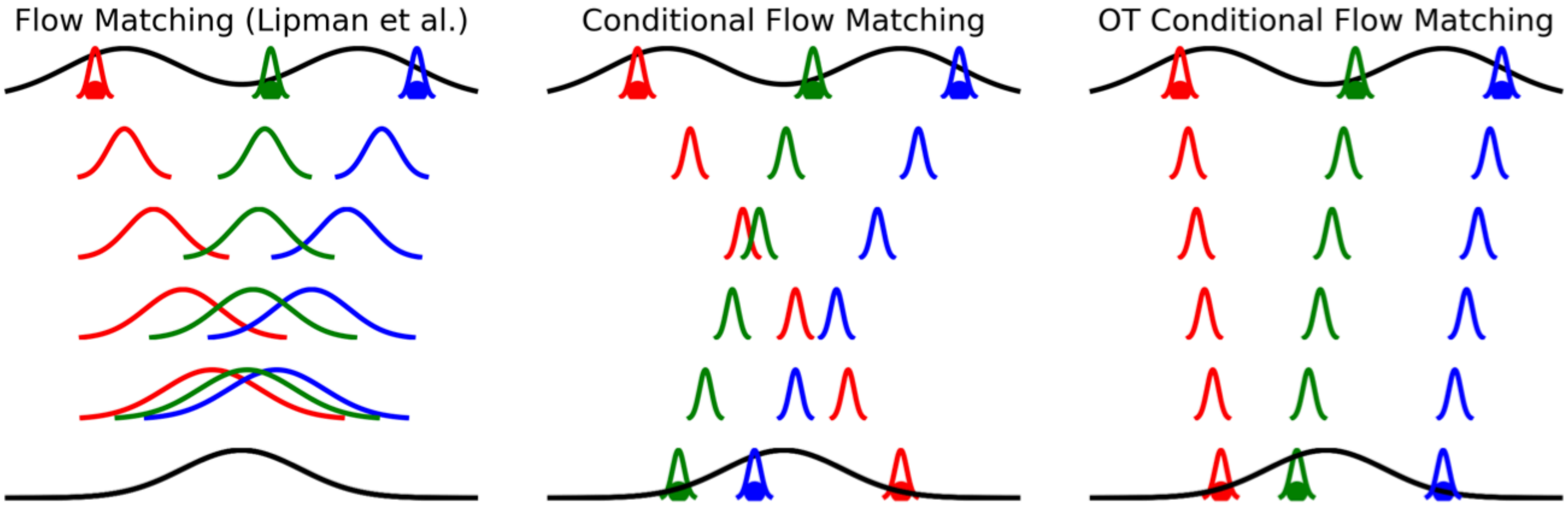
Consider two highlighted paths in the visualization of $u_t(x \mid x_1)$, with data samples $x_1^{(1)}$ and $x_1^{(2)}$. When learning a parameterized vector field $u_\theta(t,x)$ via stochastic gradient descent (SGD), we approximate the CFM loss as:

$$\mathcal{L}_{\text{CFM}}(\theta) \approx \frac{1}{2}\left\|u_\theta(t, x_t^{(1)}) - u(t, x_t^{(1)} \mid x_1^{(1)})\right\| + \frac{1}{2}\left\|u_\theta(t, x_t^{(2)}) - u(t, x_t^{(2)} \mid x_1^{(2)})\right\| \tag{19}$$

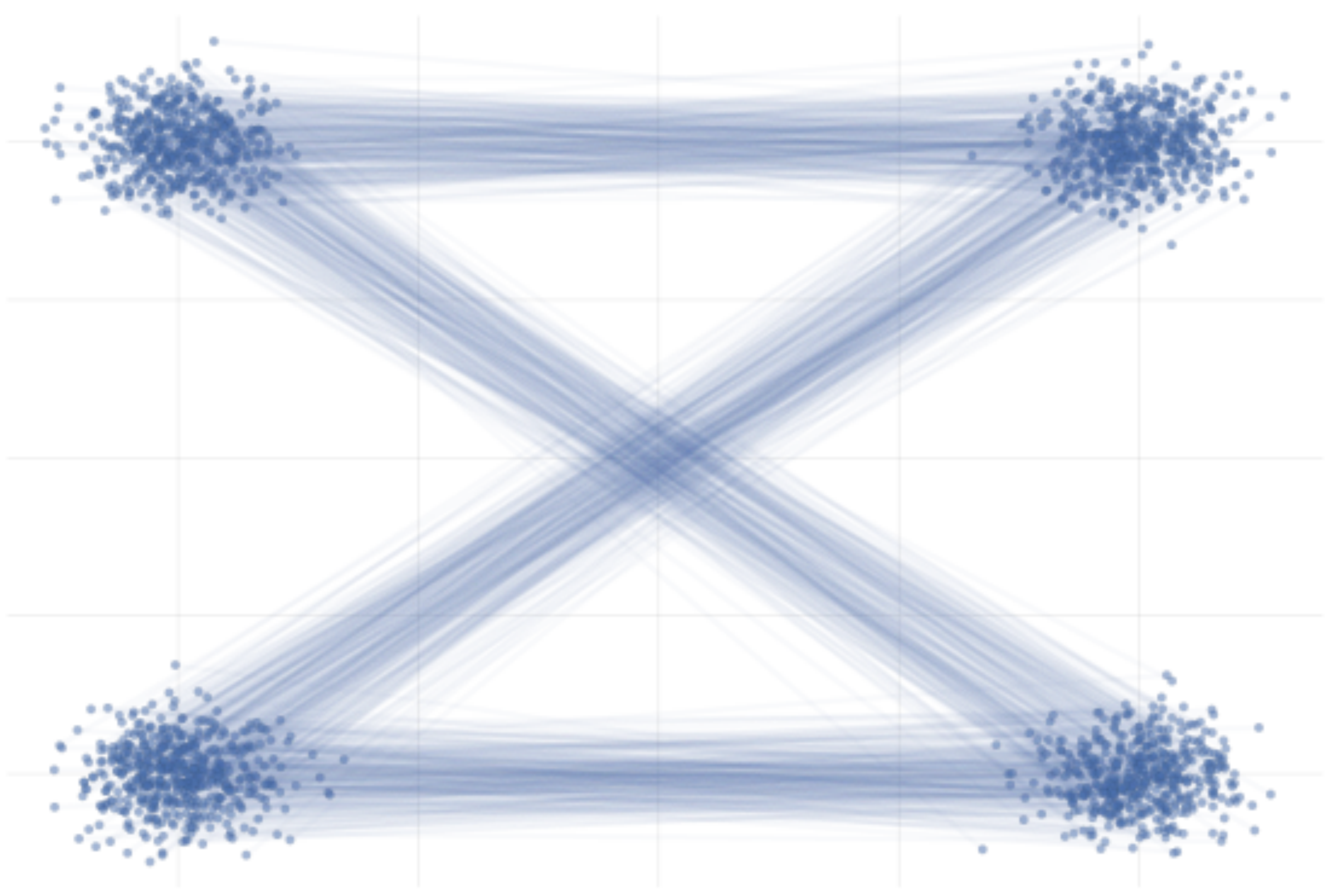
where $t \sim \mathcal{U}[0,1]$, $x_1^{(1)}, x_1^{(2)} \sim q_1$, and $x_t^{(1)} \sim p_t(\cdot \mid x_1^{(1)}), x_t^{(2)} \sim p_t(\cdot \mid x_1^{(2)})$. We compute the gradient with respect to θ for a gradient step.

In such a scenario, we're attempting to align $u_\theta(t,x)$ with two different vector fields whose corresponding paths are impossible under the marginal vector field $u(t,x)$ that we're trying to learn! This fact can lead to increased variance in the gradient estimate, and thus slower convergence.

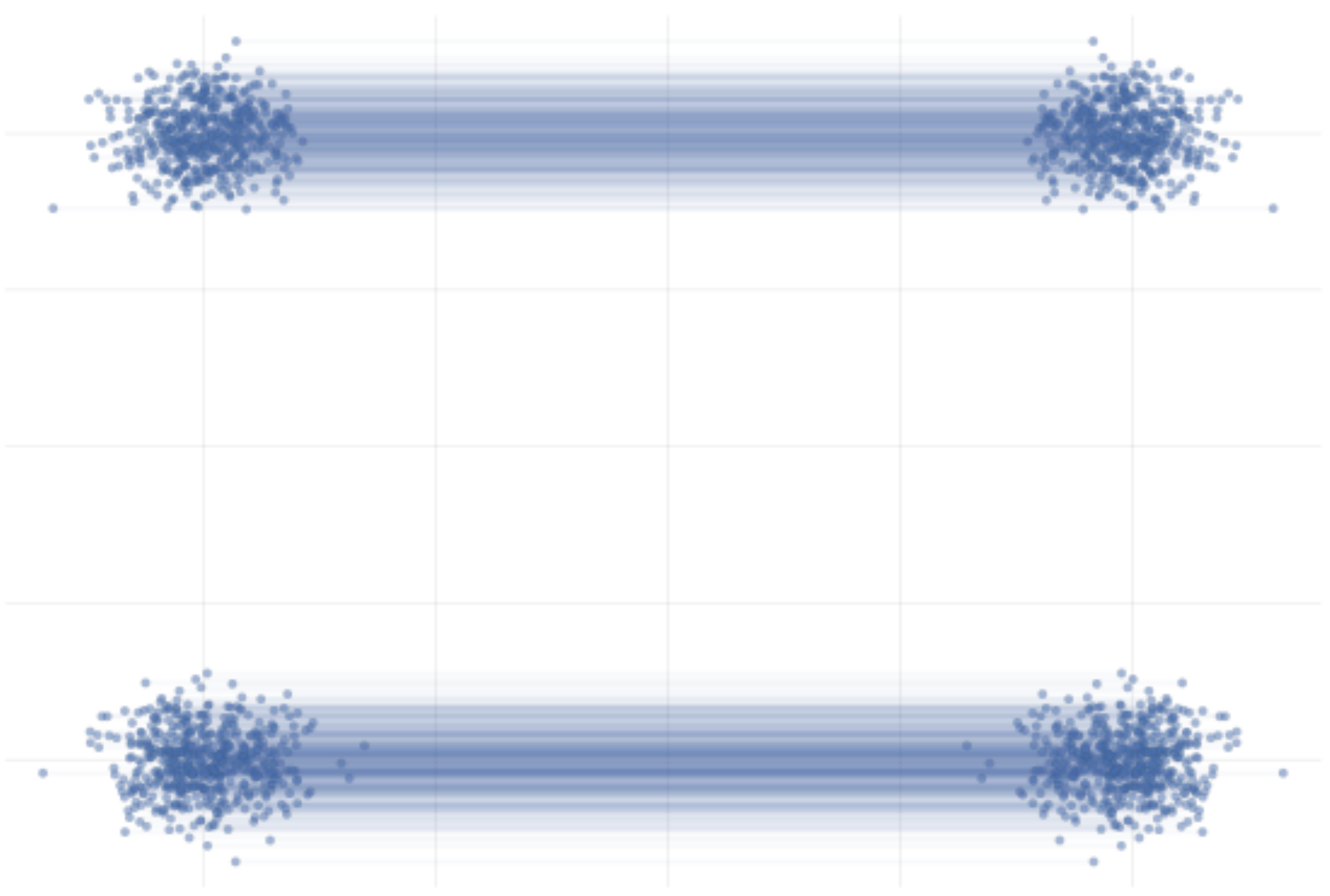
Flow Matching with OT



Flow Matching with OT



CFM - random data sampling



CFM - OT sampling