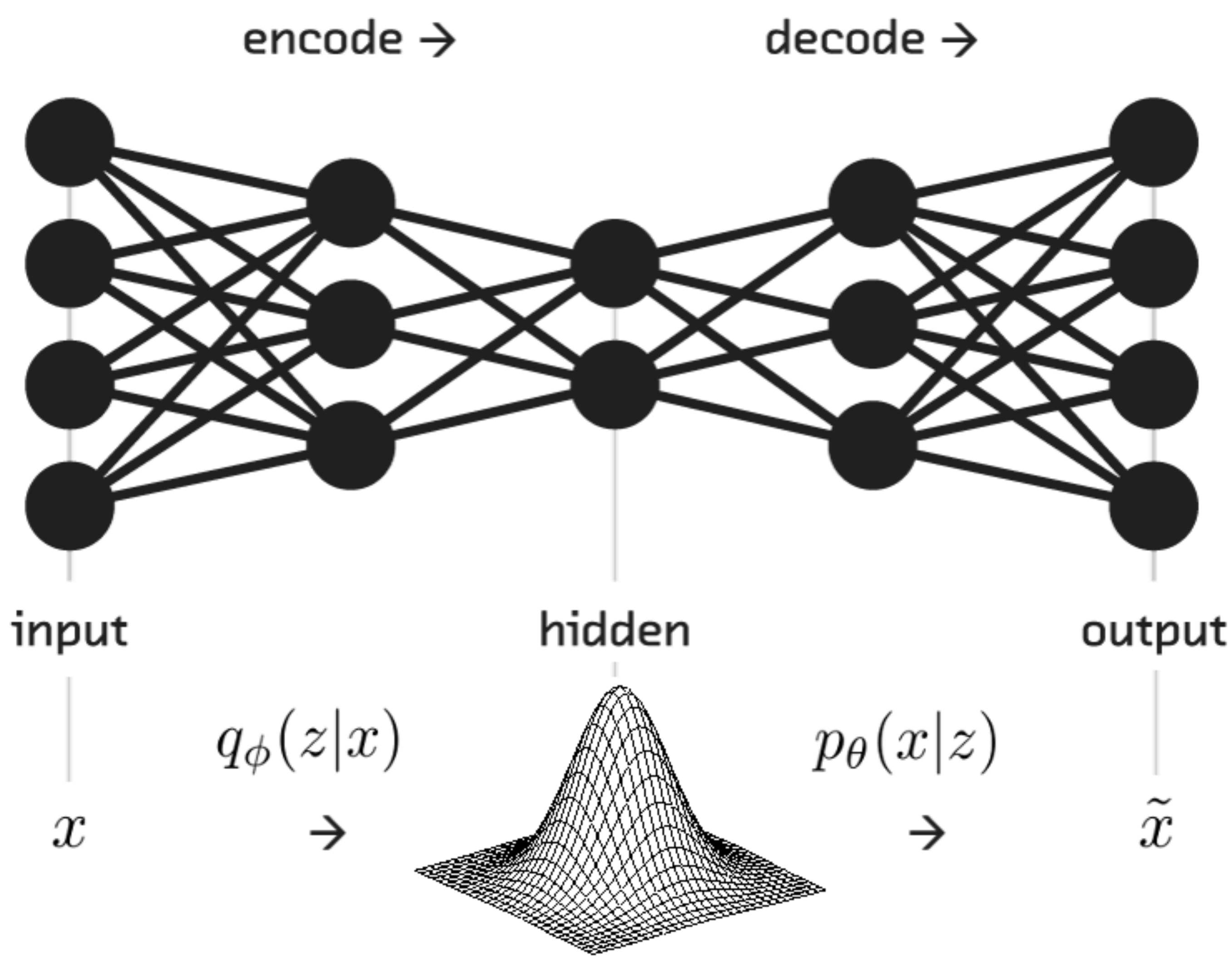
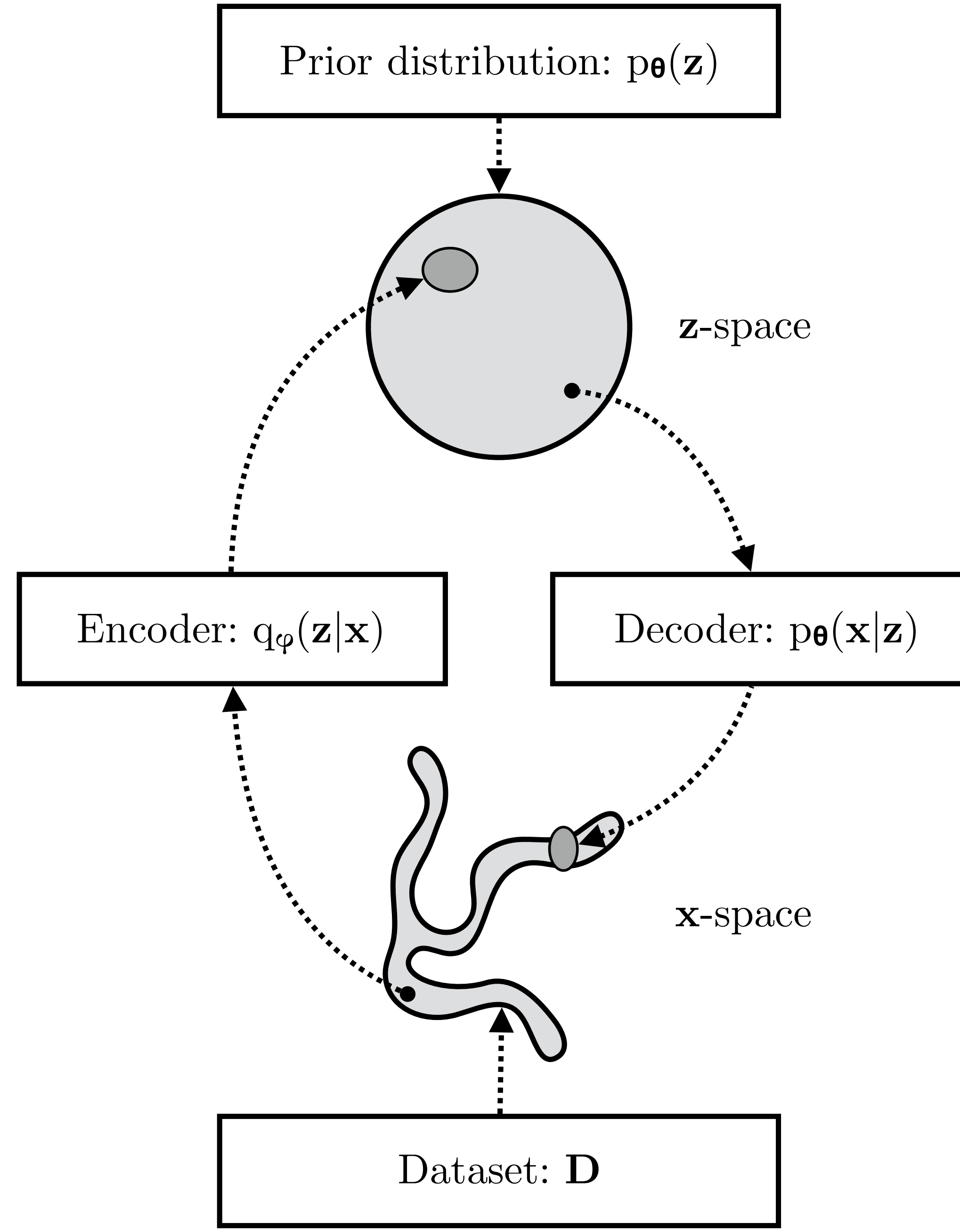
ENM5310: Data-driven modeling and probabilistic scientific computing

Lecture #24: Flow Matching

University *of* Pennsylvania

Variational auto-encoders





Kingma, D. P. (2017). Variational inference & deep learning: A new synthesis. (PhD Thesis)

Can we go beyond the mean field approximation for $q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})$?

Generative Modeling

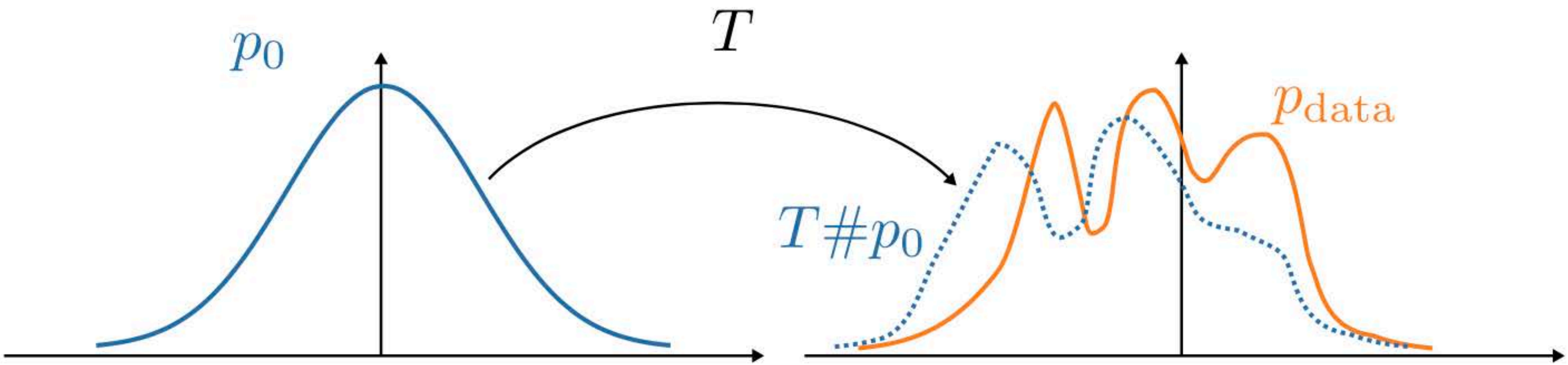


Figure 1. Modern generative modelling principle: trying to find a map $\boldsymbol{T}$ that sends the base distribution $\boldsymbol{p}_0$ as close as possible to the data distribution $\boldsymbol{p}_{\text{data}}$.

$$\boldsymbol{\theta}^* = \underset{\theta}{\operatorname{argmax}} \sum_{i=1}^{n} \log \left((\boldsymbol{T}_\theta \# \boldsymbol{p}_0)(\boldsymbol{x}^{(i)}) \right)$$

Normalizing Flows

In order to compute the log likelihood objective function in (1), if T_θ is a diffeomorphism (and thus has a differentiable inverse T_θ^{-1}), one can rely on the so-called *change-of-variable formula*

$$\log p_1(x) = \log p_0(T_\theta^{-1}(x)) + \log|\det J_{T_\theta^{-1}}(x)| \tag{2}$$

where $J_{T_\theta^{-1}} \in \mathbb{R}^{d\times d}$ is the Jacobian of T_θ^{-1}. Relying on this formula to evaluate the likelihood imposes two constraints on the network:

- T_θ must be invertible; in addition T_θ^{-1} should be easy to compute in order to evaluate the first right-hand side term in (2)
- T_θ^{-1} must be differentiable, and the (log) determinant of the Jacobian of T_θ^{-1} must not be too costly to compute in order to evaluate the second right-hand side term in (2) [5].

Normalizing Flows

The philosophy of Normalizing Flows (NFs) [2, 3, 4] is to design special neural architectures satisfying these two requirements. Normalizing flows are maps $T_\theta = \phi_1 \circ \dots \phi_K$, where each ϕ_k is a simple transformation satisfying the two constraints – and hence so does T_θ. Defining recursively $x_0 = x$ and $x_k = \phi_k(x_{k-1})$ for $k \in \{1, \dots, K\}$, through the chain rule, the likelihood is given by

$$
\begin{aligned}
\log p_1(x) &= \log p_0(\phi_1^{-1} \circ \dots \circ \phi_K^{-1}(x)) + \log|\det J_{\phi_1^{-1} \circ \dots \circ \phi_K^{-1}}(x)| \\
&= \log p_0(\phi_1^{-1} \circ \dots \circ \phi_K^{-1}(x)) + \sum_{k=1}^{K} \log|\det J_{\phi_k^{-1}}(x_k)|
\end{aligned}
$$

which is still easy to evaluate provided each ϕ_k satisfies the two constraints.

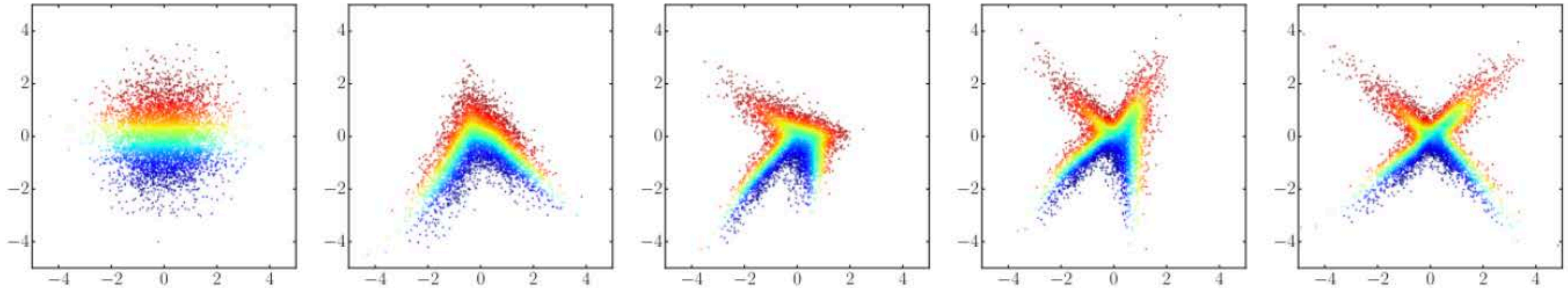


Figure 2. Normalizing flow with $K = 4$, transforming an isotropic Gaussian (leftmost) to a cross shape target distribution (rightmost). Picture from [4]

RealNVP

A more complex example of NF, that satisfies both constraints, is Real NVP [5].

▼ Click here for details about Real NVP

$$\begin{aligned} \phi(x)_{1:d'} &= x_{1:d'} \\ \phi(x)_{d':d} &= x_{d':d} \odot \exp(s(x_{1:d'})) + t(x_{1:d'}) \end{aligned} \tag{3}$$

where $d' \leq d$ and the so-called *scale* s and *translation* t are functions from $\mathbb{R}^{d'}$ to $\mathbb{R}^{d-d'}$, parametrized by neural networks. This transformation is invertible in closed-form, and the determinant of its Jacobian is cheap to compute.

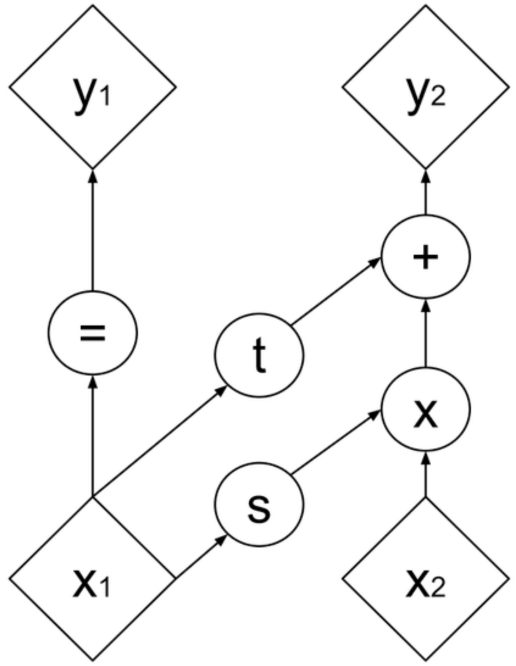
The Jacobian of ϕ defined in (3) is lower triangular:

$$J_\phi(x) = \begin{pmatrix} \mathrm{Id}_{d'} & 0_{d',d-d'} \\ \dots & \mathrm{diag}(\exp(s(x_{1:d}))) \end{pmatrix}$$

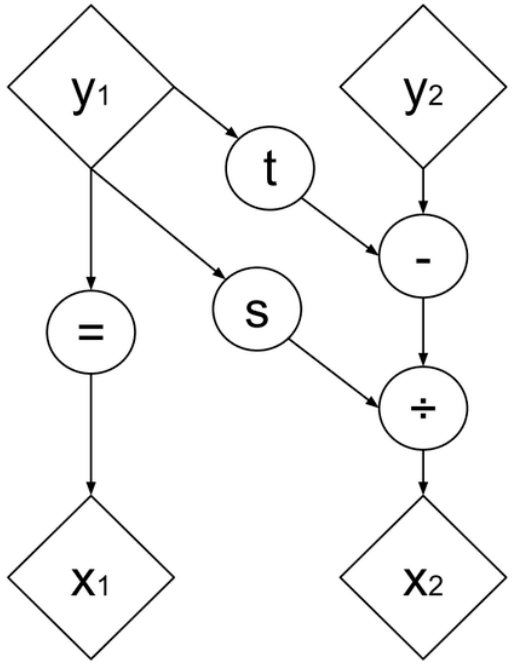
hence its determinant can be computed at a low cost, and in particular without computing the Jacobians of s and t. In addition, ϕ is easily invertible:

$$\begin{aligned} \phi^{-1}(y)_{1:d'} &= y_{1:d'} \\ \phi^{-1}(y)_{d':d} &= (y_{d':d} - t(y_{1:d'})) \odot \exp(-s(y_{1:d'})) \end{aligned}$$

It has met with a huge success in practice and a variety of alternative NFs have been proposed [6, 7, 8]. Unfortunately, the architectural constraints on Normalizing Flows tends to hinder their expressivity [7].



(a) Forward propagation



(b) Inverse propagation

Continuous Normalizing Flows

A successful solution to this expressivity problem is based on an idea similar to that of ResNets, named Continuous Normalizing Flows (CNF) [11]. If we return to the planar normalizing flow, by letting $u_{k-1}(\cdot) \stackrel{\text{def}}{=} K\sigma(b_k^\top \cdot + c)a_k$, we can rewrite the relationship between x_k and x_{k-1} as:

$$\begin{aligned} x_k &= \phi_k(x_{k-1}) \\ &= x_{k-1} + \sigma(b_k^\top x_{k-1} + c)a_k \\ &= x_{k-1} + \frac{1}{K} u_{k-1}(x_{k-1}) \end{aligned}$$

which can be interpreted as a forward Euler discretization, with step $1/K$, of the ODE

$$\begin{cases} x(0) = x_0 \\ \partial_t x(t) = u(x(t), t) \quad \forall t \in [0, 1] \end{cases} \tag{4}$$

Note that the mapping defined by the ODE, $T(x_0) := x(1)$ is inherently invertible because one can solve the *reverse-time* ODE (from $t = 1$ to 0) with the initial condition $x(1) = T(x_0)$.

This ODE is called an *initial value problem*, controlled by the **velocity field** $u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$. In addition to u, it is related to two other fundamental objects:

- the **flow** $f^u : \mathbb{R}^d \times [0, 1] \to \mathbb{R}^d$, with $f^u(x, t)$ defined as the solution at time t to the initial value problem driven by u with initial condition $x(0) = x$.
- the **probability path** $(p_t)_{t \in [0,1]}$, defined by $p_t = f^u(\cdot, t) \# p_0$, i.e., p_t is the distribution of $f^u(x, t)$ when $x \sim p_0$.

The Continuity Equation

A fundamental equation linking $\boldsymbol{p_t}$ and $\boldsymbol{u}$ is the *continuity equation*, also called transport equation:

$$\partial_t p_t + \nabla \cdot u_t p_t = 0 \tag{5}$$

Under technical conditions and up to divergence-free vector fields, for a given $\boldsymbol{p_t}$ (resp. $\boldsymbol{u}$) there exists a $\boldsymbol{u}$ (resp. $\boldsymbol{p_t}$) such that the pair $(\boldsymbol{p_t}, \boldsymbol{u})$ solves the continuity equation [8].

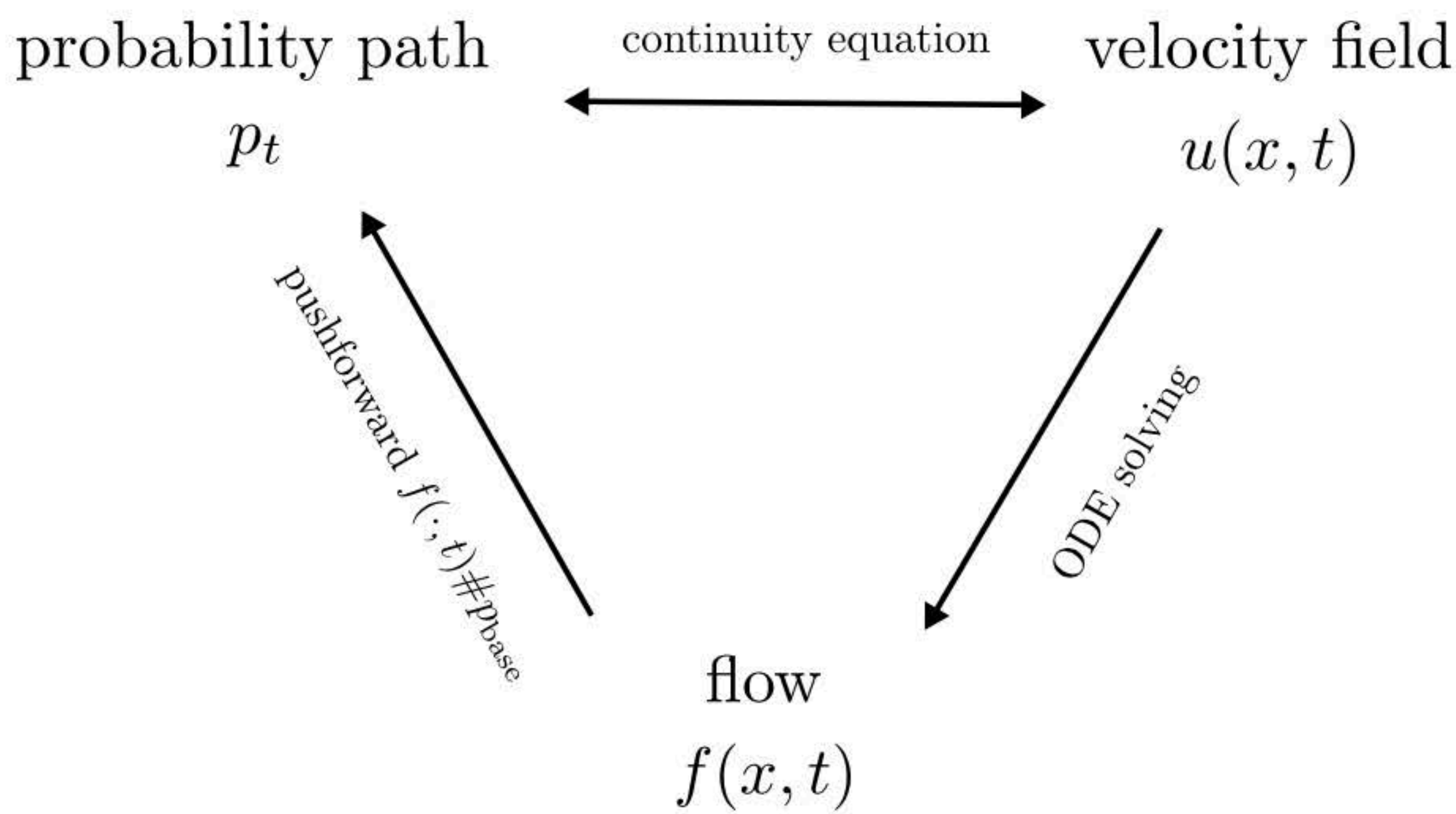


Figure 4. Link between the probability path, the velocity field and the flow.

CNF Pros and Cons

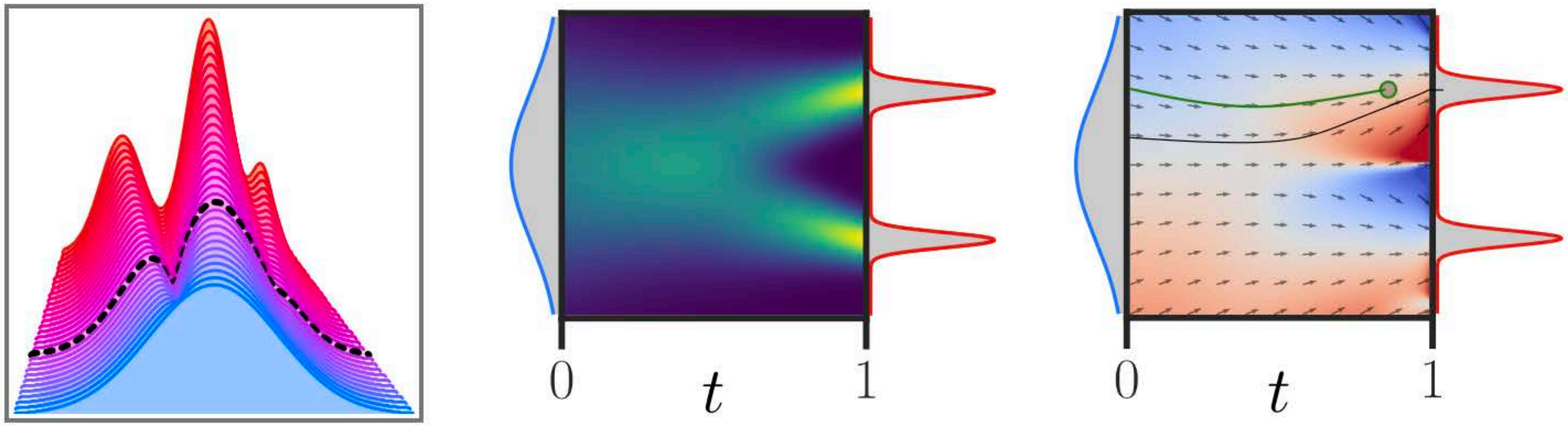
Pros

- The constraints one needs to impose on are much less stringent than in the discrete case: for the solution of the ODE to be unique, one only needs to be Lipschitz continuous in x and continuous in t
- Inverting the flow can be achieved by simply solving the ODE in reverse
- Computing the likelihood does not require inverting the flow, nor to compute a log determinant; only the trace of the Jacobian is required, that can be approximated using the Hutchinson trick.
- Adaptive time-step ODE solvers can automatically choose the number of steps and control the trade-off between sampling speed and approximation error.

Cons

- Training a neural ODE with log-likelihood does not scale well to high-dimensional spaces, and the process tends to be expensive and unstable, likely due to numerical approximations and to the (infinite) number of possible probability paths.

Flow Matching



- **(left)** A flow that maps a simple distribution p_0 in blue (typically $\mathcal{N}(0,1)$) into the data distribution to be modelled p_{data} in red. The probability path $p(x|t)$ associates to each time t, a distribution (dashed).

- **(center)** The two distributions (in gray) together with a probability path $p(x|t)$ shown as a heatmap. Such a sufficiently regular probability path is governed by a velocity field $u(x,t)$.

- **(right)** The velocity field $u(x,t)$ (shown with arrows and colors) corresponding to the previous probability path. The animation shows samples from p_0 that follow the velocity field. The distribution of these samples corresponds to $p(x|t)$.

Flow Matching

Approach: Directly formulate a regression objective to learn the velocity field u(x,t).

Given a target probability density path $p_t(x)$ and a corresponding vector field $u_t(x)$, which generates $p_t(x)$, we define the Flow Matching (FM) objective as

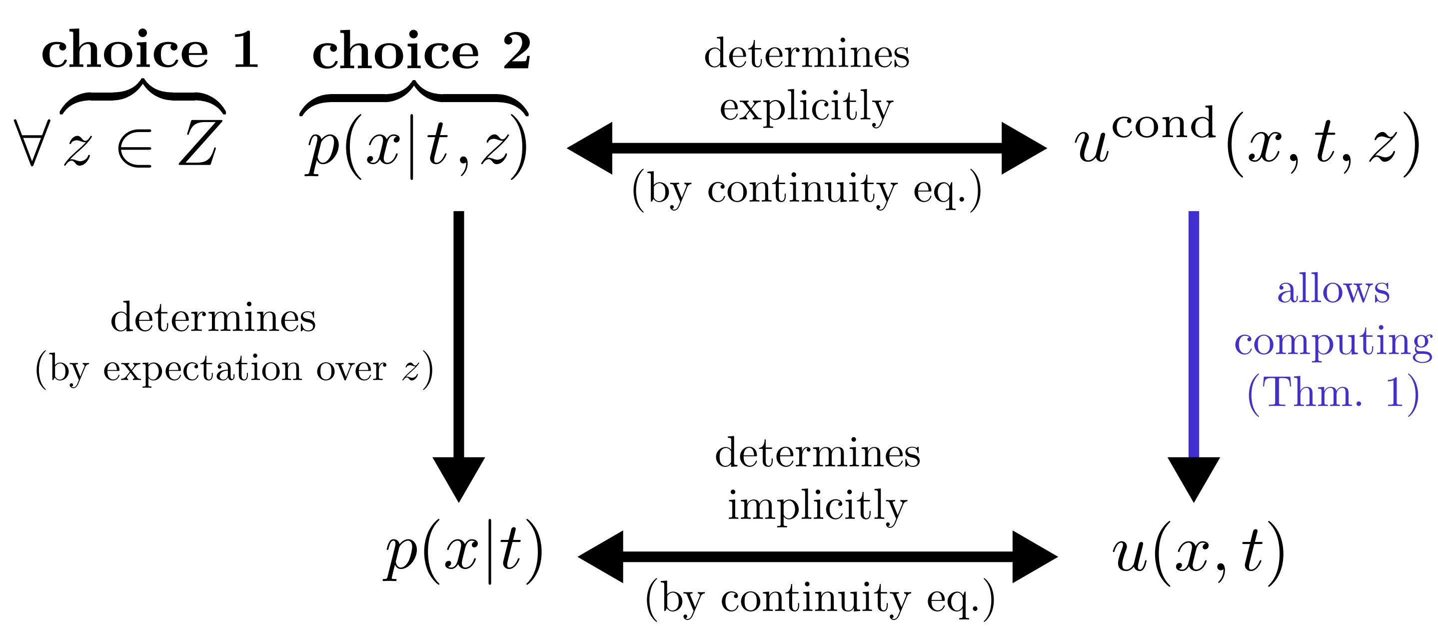
$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t,p_t(x)} \|v_t(x) - u_t(x)\|^2, \tag{5}$$

where θ denotes the learnable parameters of the CNF vector field v_t (as defined in Section 2), $t \sim \mathcal{U}[0, 1]$ (uniform distribution), and $x \sim p_t(x)$. Simply put, the FM loss regresses the vector field u_t with a neural network v_t. Upon reaching zero loss, the learned CNF model will generate $p_t(x)$.

But…

- The true p_t(x) & u(x,t) are unknown/intractable/expensive to compute.
- We need to contract a tractable target for training u(x,t).

Idea:



Modeling Choices

How to fully specify a probability path p_t? For unknown target data distribution p_{data} it is hard to choose a priori a probability path or velocity field. CFM core idea is to choose a conditioning variable z and a conditional probability path $p(x|t,z)$ (examples below) such that (1) the induced global probability path $p(x|t)$ transforms p_0 into p_{data}, (2) the associated velocity field u^{cond} has an analytic form.

General construction of conditional probability paths

> To build a conditional probability path, the user must make two modelling choices:
>
> - first, a **conditioning variable** z (independent of t)
> - then, **conditional probability paths** [13] $p(x|t,z)$ that must satisfy the following constraint: marginalizing $p(x|z,t=0)$ (resp. $p(x|z,t=1)$) over z, yields p_0 (resp. p_{data}). In other words, $p(x|t,z)$ must satisfy
>
> $$\forall x \;\; \mathbb{E}_z\left[p(x|z,t=0)\right] = p_0(x) \;\; ,$$
> $$\forall x \;\; \mathbb{E}_z\left[p(x|z,t=1)\right] = p_{\text{data}}(x) \;\; .$$

Linear Interpolation

Example 1: Linear interpolation [14, 15]

> A first choice is to condition on the base points and the target points, i.e., z is a random variable defined as:
>
> $$z \overset{\text{choice}}{=} (x_0, x_1) \sim p_0 \times p_{\text{data}} \,.$$
>
> Among all the possible probability paths, one can choose to use very concentrated Gaussian distributions and simply interpolate between x_0 and x_1 in straight line: for some fixed standard deviation σ, it writes as
>
> $$p\big(x|t, z=(x_0, x_1)\big) \overset{\text{choice}}{=} \mathcal{N}((1-t)\cdot x_0 + t \cdot x_1, \sigma^2 \mathrm{Id}) \,.$$
>
> To recover the correct distributions p_0 at $t=0$ (resp. p_{target} at $t=1$), one must enforce $\sigma = 0$, finally leading to
>
> $$p\big(x|t, z=(x_0, x_1)\big) \overset{\text{choice}}{=} \delta_{(1-t)\cdot x_0 + t \cdot x_1}(x) \,,$$
>
> where δ denotes the Dirac delta distribution.

Linear Interpolation

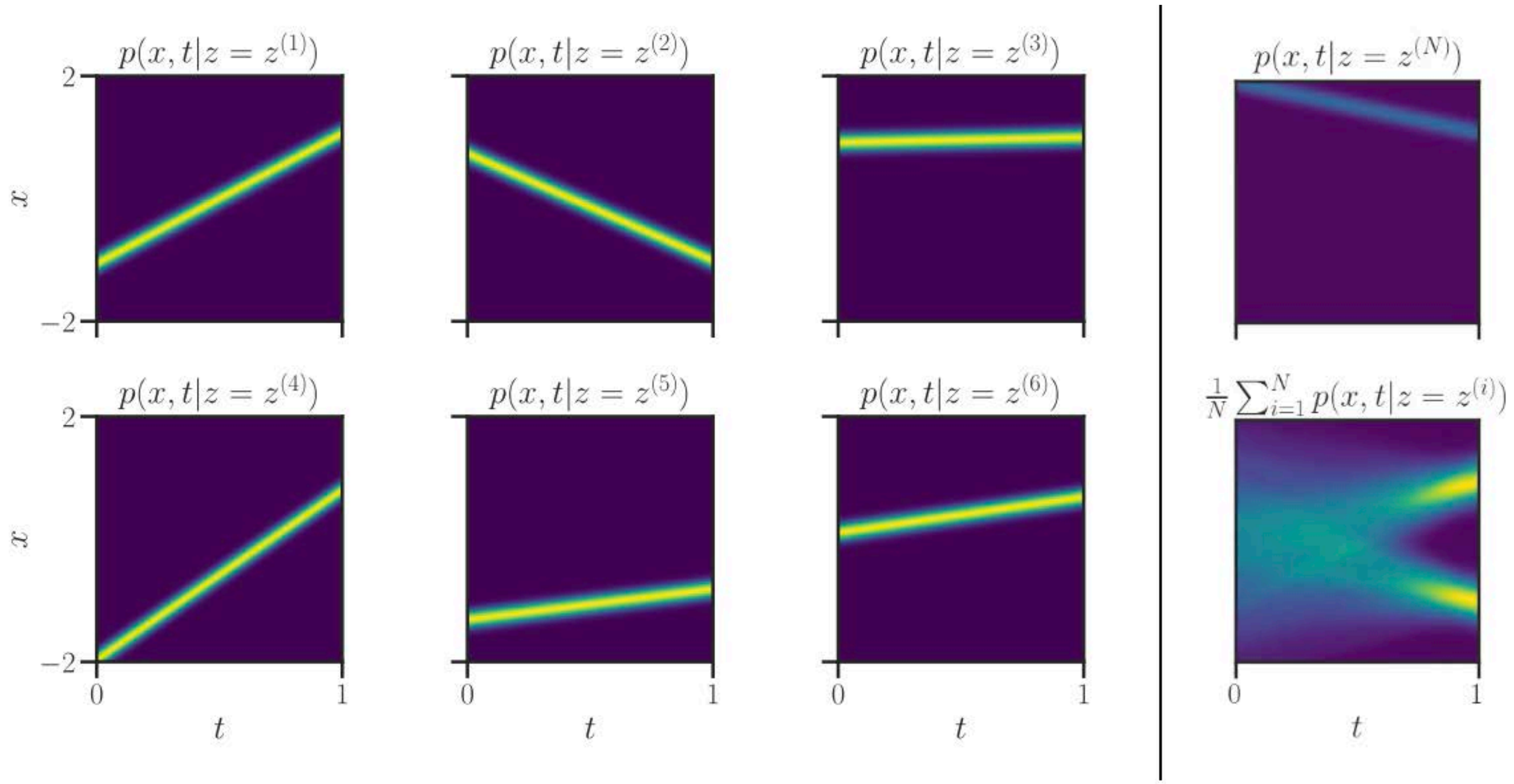


Figure 8. Conditional probability paths as linear interpolation.
(left) $p(x|t, z = z^{(i)})$ for six samples $z^{(i)}$ (each being a pair (x_0, x_1)). **(right)** Visualizing the convergence of the empirical average towards $p(x|t) = \mathbb{E}_z\left[p(x|t,z)\right] \approx \frac{1}{N}\sum_{i=1}^{N} p(x|t, z = z^{(i)})$.

Then, one can show that setting

$$u^{\text{cond}}(x, t, z = (x_0, x_1)) = x_1 - x_0$$

satisfies the continuity equation with $p(x|t,z)$ [12]. Hence, the two choices made – z and $p(x|t,z)$ – result in a very easy-to-compute conditional velocity field $u^{\text{cond}}(x, t, z = (x_0, x_1))$ which will be later used as a supervision signal to learn $u_\theta(x,t)$.

Conical Gaussian Paths

Example 2: Conical Gaussian paths [16]

> One can make other choices for the conditioning variable, for instance
>
> $$z \stackrel{\text{choice}}{=} x_1 \sim p_{\text{data}}\,,$$
>
> and the following choice for the conditional probability path: simply translate and progressively scale down the base normal distribution towards a Dirac delta in z:
>
> $$p(x|t, z = x_1) \stackrel{\text{choice}}{=} \mathcal{N}(tx_1, (1-t)^2\text{Id})\,.$$

Conical Gaussian Paths

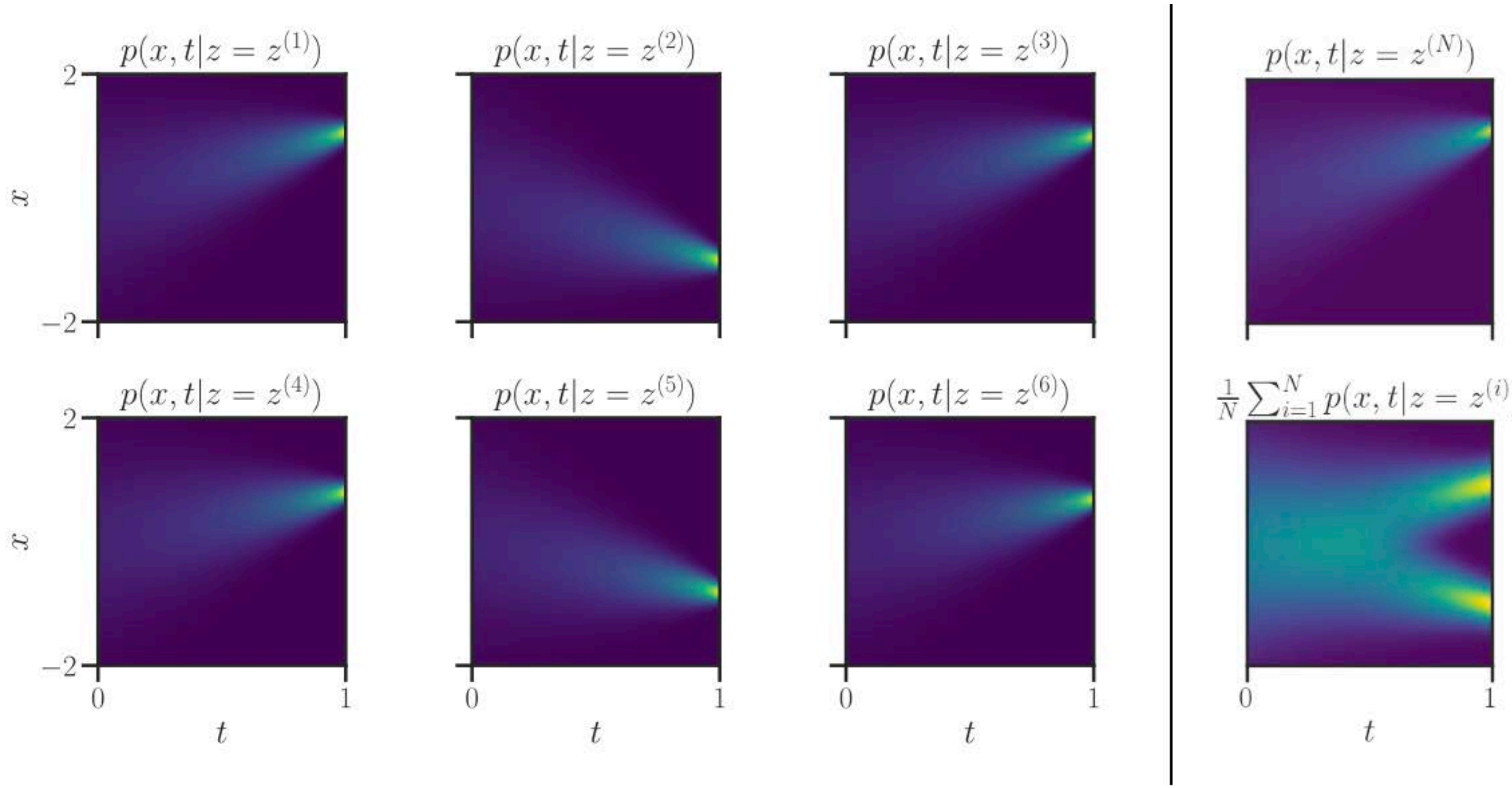


Figure 9. Conditional probability paths as shrinking conical Gaussians.
(left) $p(x|t, z = z^{(i)})$ for six samples $z^{(i)}$ (each being a value x_1). **(right)** Visualizing the convergence of the empirical average towards $p(x|t) = \mathbb{E}_z\left[p(x|t,z)\right] \approx \frac{1}{N}\sum_{i=1}^{N} p(x|t, z = z^{(i)})$.

Then, one can show that setting $u^{\mathrm{cond}}(x, t, z = x_1) = \frac{x - x_1}{1-t}$ leads to a couple $(u^{\mathrm{cond}}(x,t,z), p(x|t,z))$ satisfying the continuity equation.

From Conditional to Unconditional Velocity

> The previous section provided examples on how to choose a conditioning variable z and a simple conditional probability path $p(x|t, z)$. The marginalization of $p(x|t, z)$ directly yields a (intractable) closed-form formula for the probability path: $p(x|t) = \mathbb{E}_z\left[p(x|t, z)\right]$.

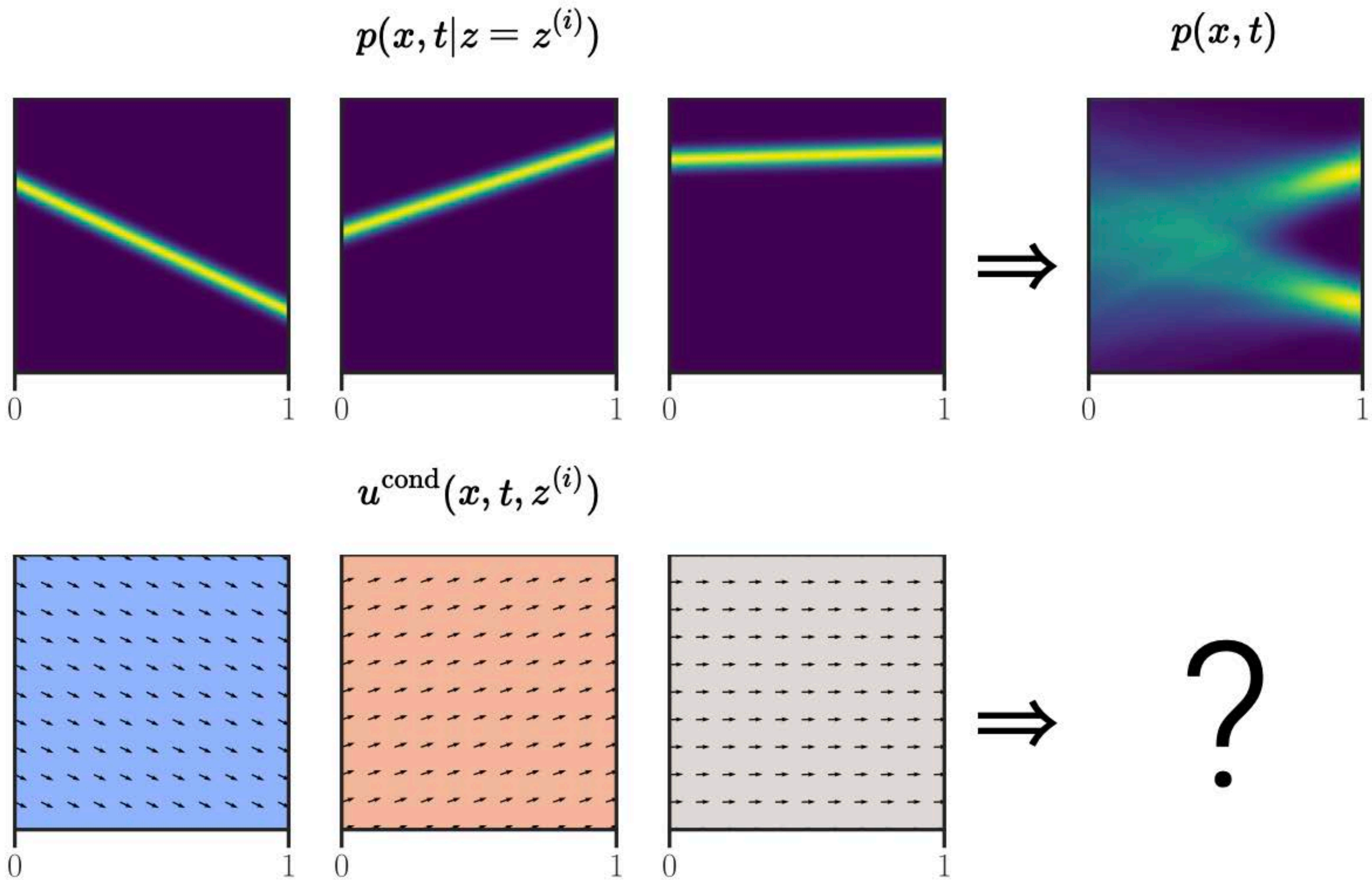


Figure 10. **(top)** Illustration of conditional probability paths. $p(x, t|z^{(i)}) = \mathcal{N}((1-t) \cdot x_0 + t \cdot x_1, \sigma^2)$. **(bottom)** Illustration of the associated conditional velocity fields $u^{\text{cond}}(x, t, z) = x_1 - x_0$ for three different values of $z = (x_0, x_1)$. **(top right)** By marginalization over z, the conditional probability paths directy yield an expression for the probability path. **(bottom right)** Expressing the velocity field $u(x, t)$ as a function of the conditional velocity field $u^{\text{cond}}(x, t, z)$ is not trivial.

Theorem 1

Let z be any random variable independent of t. Choose conditional probability paths $p(x|t, z)$, and let $u^{\text{cond}}(x, t, z)$ be the velocity field associated to these paths.

Then the velocity field $u(x, t)$ associated to the probability path [14] $p(x, t) = \mathbb{E}_z [p(x, t|z)]$ has a closed-form formula:

$$\forall t, x,\ \ u(x, t) = \mathbb{E}_{z|x,t} \left[u^{\text{cond}}(x, t, z)\right] \ . \tag{9}$$

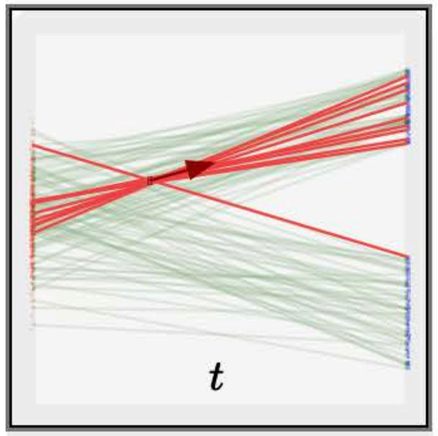


Figure 12. Move your mouse at any location (t, x) to see a sampling of $z|t, x$ (i.e., trajectories between x_0 and x_1 that pass close to (t, x)) and the associated velocity (average of the directions of trajectories)

▼ Click here to unroll the proof

We first prove an intermediate result, which is the form most often found in the literature, but not the most interpretable in our opinion:

$$\forall\, t,\ x,\ u(x, t) = \mathbb{E}_z \left[\frac{u^{\text{cond}}(x, t, z) p(x|t, z)}{p(x|t)} \right] \ . \tag{10}$$

▶ Click here to unroll the proof of (10)

Then, we rewrite this intermediate formulation:

$$\begin{aligned}
\forall\, t,\ x,\ u(x, t) &= \mathbb{E}_z \left[\frac{u^{\text{cond}}(x, t, z) p(x|t, z)}{p(x|t)} \right] \qquad (11) \\
&= \int_z \frac{u^{\text{cond}}(x, t, z) p(x|t, z)}{p(x|t)} p(z) \mathrm{d}z \\
&= \int_z u^{\text{cond}}(x, t, z) \underbrace{p(x|t, z)}_{= \frac{p(z|x,t) \cdot p(x,t)}{p(t,z)}} \cdot p(z) \cdot \underbrace{\frac{1}{p(x|t)}}_{= \frac{p(t)}{p(x,t)}} \mathrm{d}z \\
&= \int_z u^{\text{cond}}(x, t, z) \frac{p(z|x, t) \cdot p(x, t)}{p(t, z)} \cdot p(z) \cdot \frac{p(t)}{p(x, t)} \mathrm{d}z \\
&= \int_z u^{\text{cond}}(x, t, z) p(z|x, t) \underbrace{\frac{p(z) \cdot p(t)}{p(t, z)}}_{=1} \mathrm{d}z \\
&= \int_z u^{\text{cond}}(x, t, z) p(z|x, t) \mathrm{d}z \\
&= \mathbb{E}_{z|x,t} \left[u^{\text{cond}}(x, t, z)\right] \qquad (12)
\end{aligned}$$

Learning with Conditional Velocity Fields.

> We recall that the choices of the conditioning variable z and the probability paths $p(x|t,z)$ **entirely define the (intractable) vector field** $u(x,t)$. Conditional flow matching idea is to learn a vector field $u_\theta^{\mathrm{CFM}}(x,t)$ that estimates/"*matches*" the pre-defined velocity field $u(x,t)$, by regressing against the (cheaper to compute) condition velocity fields $u^{\mathrm{cond}}(x,t,z)$ associated to the conditional probability paths $p(x|t,z)$.

Theorem 2

Regressing against the conditional velocity field $u^{\mathrm{cond}}(x,t,z)$ with the following conditional flow matching loss,

$$\mathcal{L}^{\mathrm{CFM}}(\theta) \overset{\mathrm{def}}{=} \mathbb{E}_{\substack{t\sim\mathcal{U}([0,1])\\ z\sim p_z\\ x\sim p(\cdot|t,z)}} \| u_\theta^{\mathrm{CFM}}(x,t) - \underbrace{u^{\mathrm{cond}}(x,t,z)}_{\substack{\text{chosen to be}\\ \text{explictly defined,}\\ \text{cheap to compute,}\\ \text{e.g., } x_1 - x_0}} \|^2 \ ,$$

is equivalent to directly regressing against the intractable unknown vector field $u(x,t)$

$$\mathcal{L}^{\mathrm{CFM}}(\theta) \underset{\text{(proof below)}}{=} \mathbb{E}_{\substack{t\sim\mathcal{U}([0,1])\\ x\sim p_t}} \| u_\theta^{\mathrm{CFM}}(x,t) - \underbrace{u(x,t)}_{\substack{\text{implicitly defined,}\\ \text{hard/expensive}\\ \text{to compute}}} \|^2 + \underbrace{C}_{\text{indep. of } \theta} \ .$$

SUMMARY: FLOW MATCHING IN PRACTICE

Flow Matching In Practice	Linear Interpolation	Conical Gaussian Paths
1. Define a variable z with some known distribution $p(z)$	$p(z = (x_0, x_1)) = p_0 \times p_{\text{data}}$	$p(z = x_1) = p_{\text{data}}$
2. Define a simple conditional distribution $p(x \mid t, z)$	$\mathcal{N}((1-t) \cdot x_0 + t \cdot x_1, \sigma^2 \cdot \mathrm{Id})$	$\mathcal{N}(t \cdot x_1, (1-t)^2 \cdot \mathrm{Id})$
3. Compute an associated velocity field $u^{\text{cond}}(x, t, z)$	$x_1 - x_0$	$\frac{x_1 - x}{1-t}$
4. Train model using the conditional loss $\mathcal{L}^{\text{CFM}}$ Sample $t \sim \mathcal{U}_{[0,1]}$, $z \sim p_z$, $x \sim p(x \mid t, z)$	Use data points $x^{(1)}, \ldots, x^{(n)}$	
5. Sample from $p_1 \approx p_{\text{data}}$ Sample $x_0 \sim p_0$, Integration scheme on $t \in [0,1]$	Numerical integration, e.g, Euler scheme: $x_{k+1} = x_k + \frac{1}{N} u_\theta(x_k, t)$	