EECE 571F: Advanced Topics in Deep Learning

Lecture 11: Flow Matching

Renjie Liao, Qi Yan, Yuanpei Gao

University of British Columbia

Winter, Term 1, 2024

Outline

- **Normalizing Flows and Continuous Normalizing Flows**
    - The Continuity Equation
- The Fokker Plank Equation
- Flow matching
- Variants:
    - Batch Optimal Transport Flow Matching

Normalizing Flows

- Our goal with this setup is to learn the transformation from $p(\cdot)$ to the complex data distribution $q(\cdot)$.

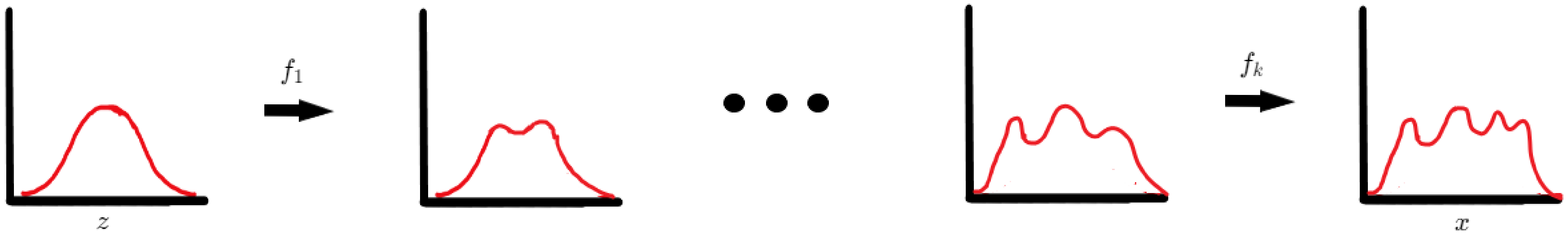


Image credit: https://dsl-lab.github.io/blog/2024/flows/

Normalizing Flows

- Our goal with this setup is to learn the transformation from $p(\cdot)$ to the complex data distribution $q(\cdot)$.
- We can do this by learning the invertible transformation f_θ using neural networks.

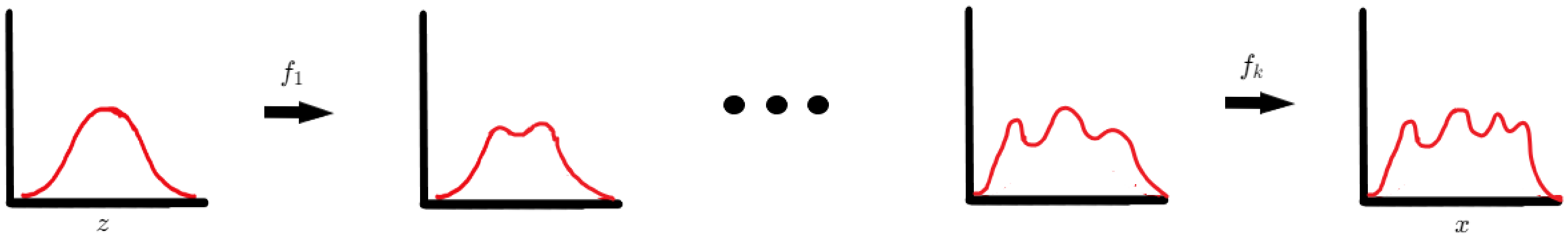


Image credit: https://dsl-lab.github.io/blog/2024/flows/

Normalizing Flows

- Our goal with this setup is to learn the transformation from $p(\cdot)$ to the complex data distribution $q(\cdot)$.
- We can do this by learning the invertible transformation f_θ using neural networks.
- f_θ can contain multiple transformations. Each transformation transforms an input distribution into a slightly more complex distribution.

$$f_\theta = f_k \circ f_{k-1} \cdots f_2 \circ f_1.$$

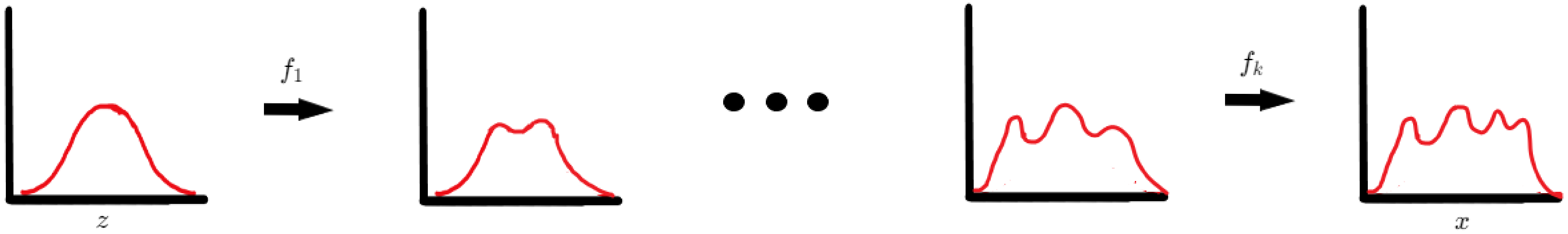


Image credit: https://dsl-lab.github.io/blog/2024/flows/

Normalizing Flows

- Starting with known distribution $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\cdot)$

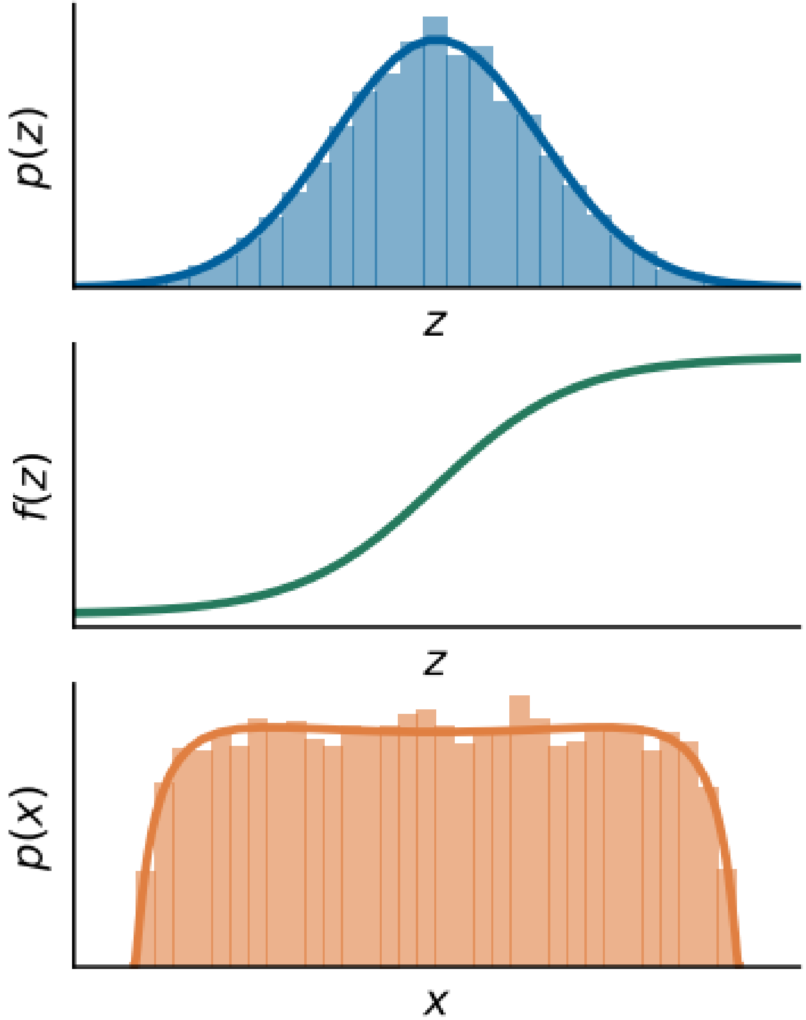


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Normalizing Flows

- Starting with known distribution $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\cdot)$
- Let f_θ be an invertible and differentiable function, apply the transformation to $\boldsymbol{z}$:

$$p_\theta(\boldsymbol{x}) = p_{\boldsymbol{z}}(f_\theta^{-1}(\boldsymbol{x})) \cdot \left|\det \frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|$$

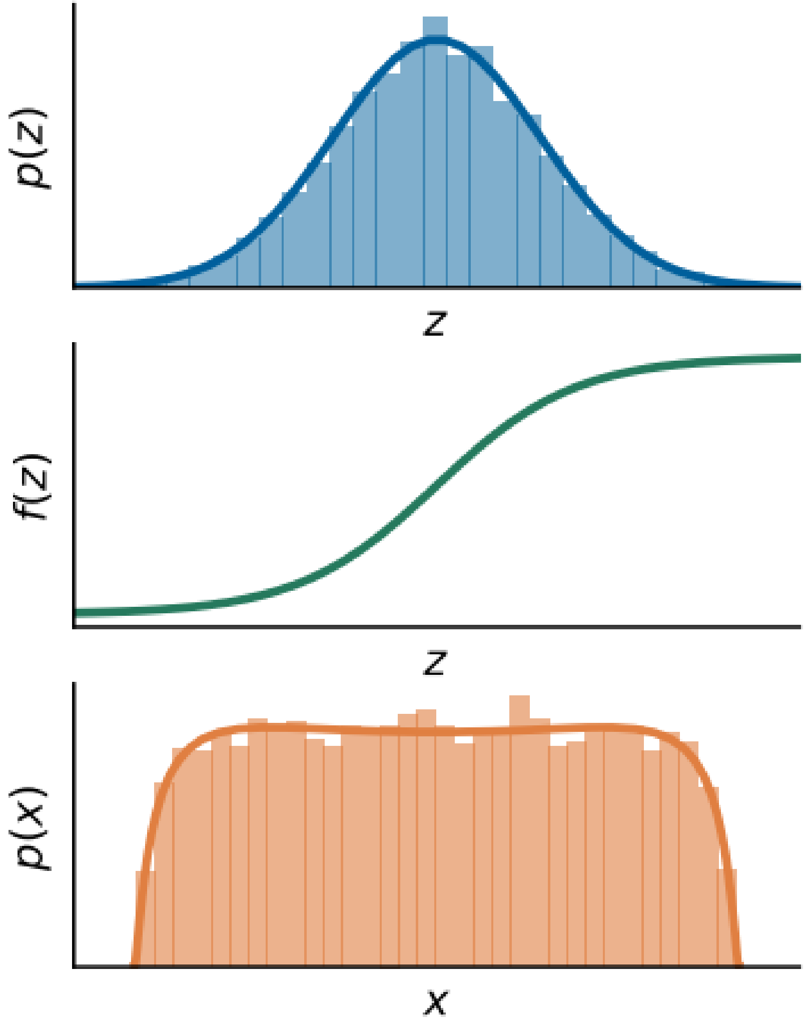


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Normalizing Flows – Multivariate Change of Variable

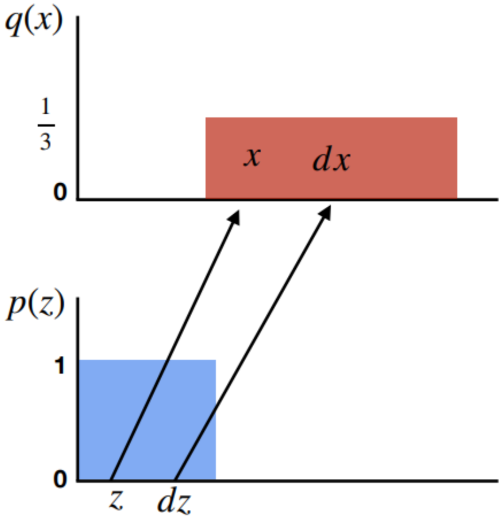
- Consider we have $\boldsymbol{x} = f(\boldsymbol{z})$, when transforming coordinates from $\boldsymbol{z}$-space to $\boldsymbol{x}$-space, we are interested in understanding how infinitesimal regions around a point in the original space change under the transformation.
- The function $f(\boldsymbol{z})$ can be approximated using first-order Taylor expansion:

$$\boldsymbol{x} \simeq f(\boldsymbol{z_0}) + J(\boldsymbol{z_0})(\boldsymbol{z} - \boldsymbol{z_0})$$

Normalizing Flows – Multivariate Change of Variable

- Based on the probability density preservation under transformation, we can have:

$$p_{\boldsymbol{x}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = p_{\boldsymbol{z}}(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$$

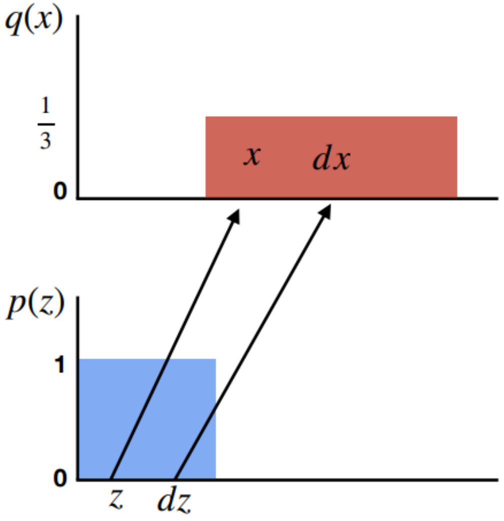


Normalizing Flows – Multivariate Change of Variable

- Based on the probability density preservation under transformation, we can have:

$$p_{\boldsymbol{x}}(\boldsymbol{x})\mathrm{d}\boldsymbol{x} = p_{\boldsymbol{z}}(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$$

- The infinitesimal volume transform is (only the linear term matters):



$$\mathrm{d}\boldsymbol{x} = |\det(J(\boldsymbol{z}))|\mathrm{d}\boldsymbol{z} \qquad J(\boldsymbol{z}) = \begin{pmatrix} \frac{\partial z_1}{\partial x_1} & \frac{\partial z_2}{\partial x_1} & \cdots & \frac{\partial z_n}{\partial x_1} \\ \frac{\partial z_1}{\partial x_2} & \frac{\partial z_2}{\partial x_2} & \cdots & \frac{\partial z_n}{\partial x_2} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_1}{\partial x_m} & \frac{\partial z_2}{\partial x_m} & \cdots & \frac{\partial z_n}{\partial x_m} \end{pmatrix}$$

$$p_{\boldsymbol{x}}(\boldsymbol{x})|\det(J(\boldsymbol{z}))|\mathrm{d}\boldsymbol{z} = p_{\boldsymbol{z}}(\boldsymbol{z})\mathrm{d}\boldsymbol{z}$$

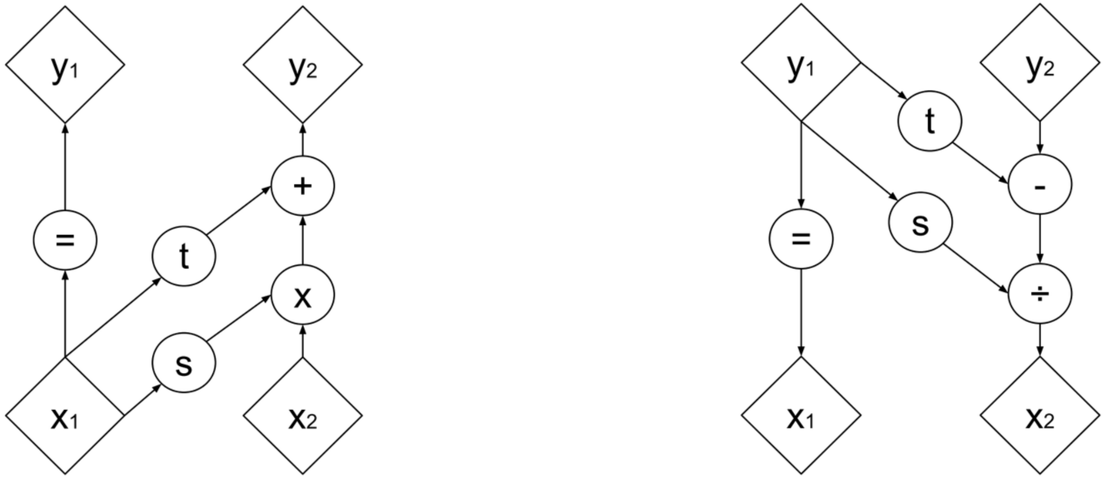
Normalizing Flows – Multivariate Change of Variable

- Rearrange the equations and we will have:

$$p_{\boldsymbol{x}}(\boldsymbol{x}) = p_{\boldsymbol{z}}(\boldsymbol{z})\left|\det(J(\boldsymbol{z}))\right|^{-1} = p_{\boldsymbol{z}}(f^{-1}(\boldsymbol{x}))\left|\det(J(f^{-1}(\boldsymbol{x})))\right|^{-1}$$

Normalizing Flows – Example: Real NVP

- The design of Real NVP model: affine coupling layer



(a) Forward propagation (b) Inverse propagation

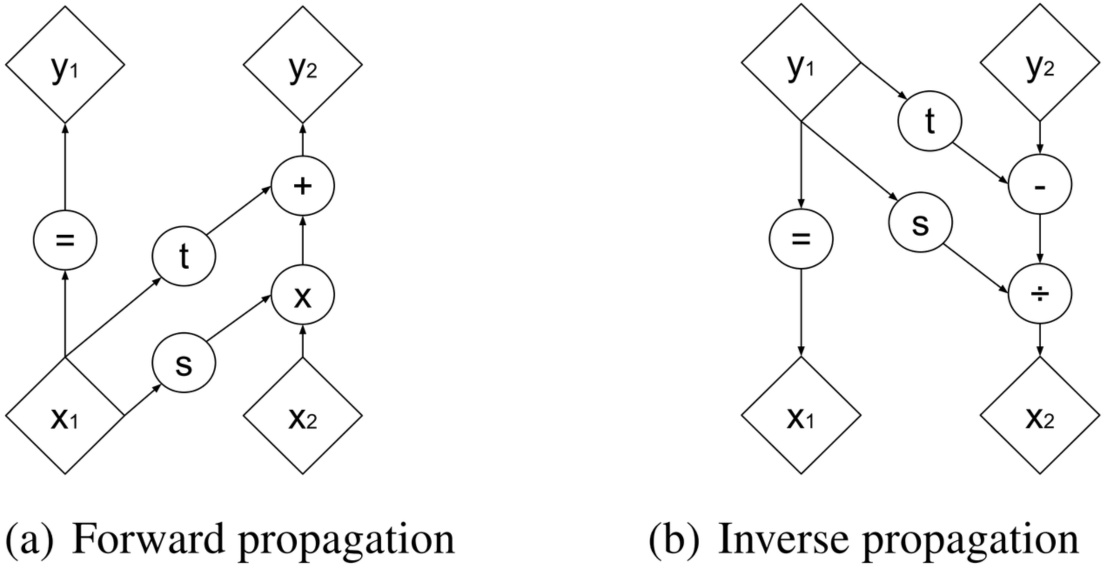
Image credit: https://arxiv.org/pdf/1605.08803

Normalizing Flows – Example: Real NVP

- The design of Real NVP model: affine coupling layer
- For the forward mapping:

$$y_{1:d} = x_{1:d}$$

$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$



(a) Forward propagation (b) Inverse propagation

Image credit: https://arxiv.org/pdf/1605.08803

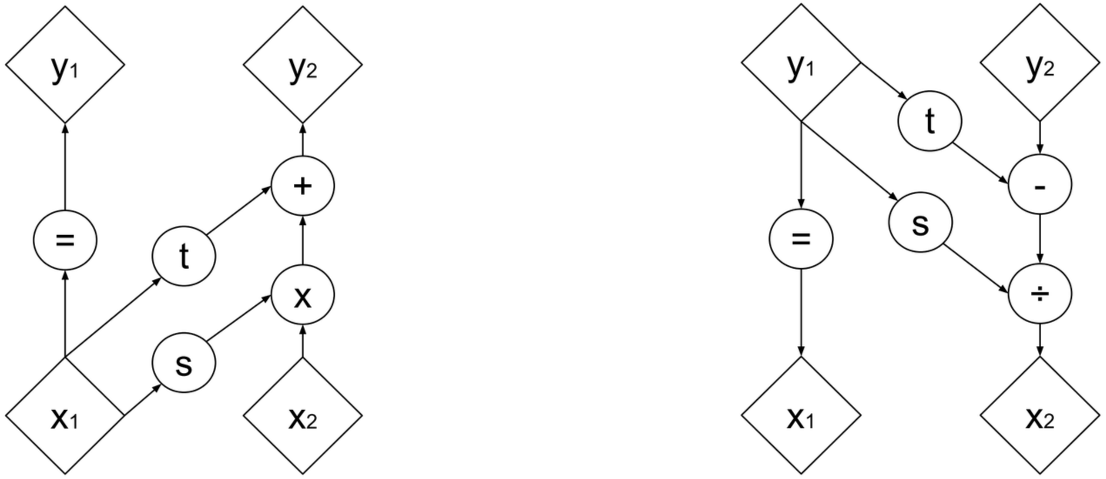
Normalizing Flows – Example: Real NVP

- The design of Real NVP model: affine coupling layer
- For the forward mapping:

$$y_{1:d} = x_{1:d}$$
$$y_{d+1:D} = x_{d+1:D} \odot \exp(s(x_{1:d})) + t(x_{1:d})$$

- For the inverse mapping:

$$x_{1:d} = y_{1:d}$$
$$x_{d+1:D} = (y_{d+1:D} - t(y_{1:d})) \odot \exp(-s(y_{1:d}))$$



(a) Forward propagation (b) Inverse propagation

Image credit: https://arxiv.org/pdf/1605.08803

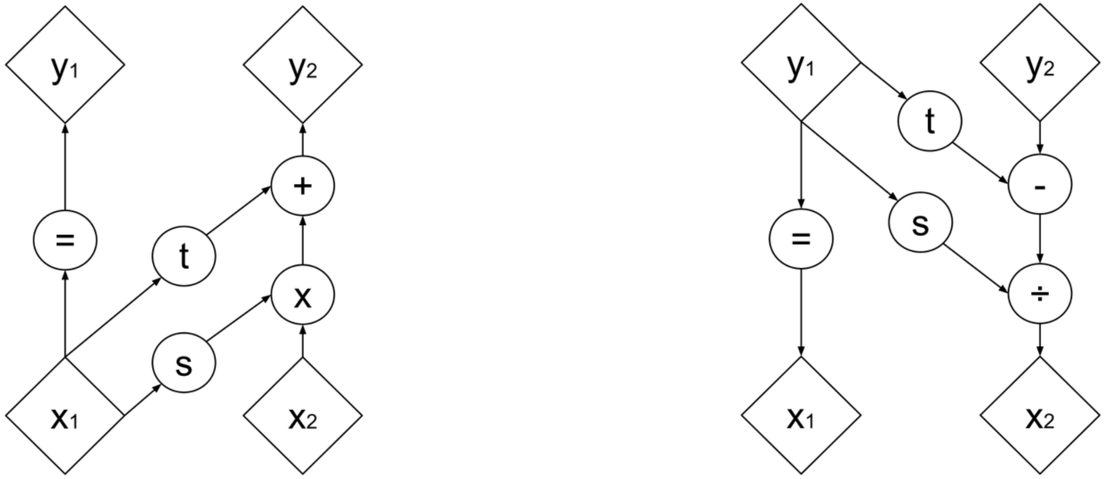
Normalizing Flows – Example: Real NVP

- From the coupling layer, we can easily derive the Jacobian:

$$\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}^T} = \begin{bmatrix} I_d & 0 \\ \frac{\partial y_{d+1:D}}{\partial x_{1:d}^T} & \operatorname{diag}(\exp[s(x_{1:D})]) \end{bmatrix}$$

- It is triangular, which means the determinant is the product of the diagonals. The log of Jacobian determinants can be simplified as:

$$\log\left(\left|\det\left(\frac{\partial \boldsymbol{y}}{\partial \boldsymbol{x}^T}\right)\right|\right) = \sum_j s_j(x_{1:d})$$



(a) Forward propagation

(b) Inverse propagation

Image credit: https://arxiv.org/pdf/1605.08803

Normalizing Flows

- Starting with known distribution $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\cdot)$
- Let f_θ be an invertible and differentiable function, apply the transformation to $\boldsymbol{z}$:

$$p_\theta(\boldsymbol{x}) = p_{\boldsymbol{z}}(f_\theta^{-1}(\boldsymbol{x})) \cdot \left| \det \frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}} \right|$$

- Maximize likelihood of data:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x_i)$$

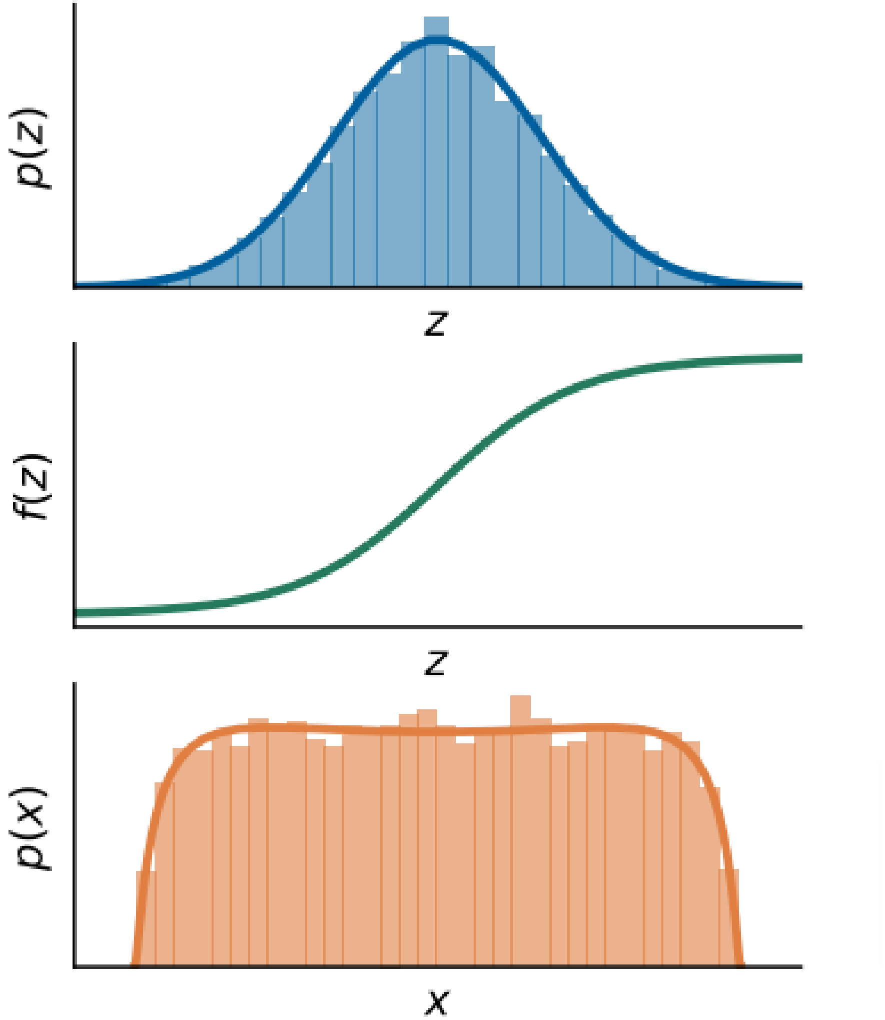


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Normalizing Flows

- Starting with known distribution $\boldsymbol{z} \sim p_{\boldsymbol{z}}(\cdot)$
- Let f_θ be an invertible and differentiable function, apply the transformation to $\boldsymbol{z}$:

$$p_\theta(\boldsymbol{x}) = p_{\boldsymbol{z}}(f_\theta^{-1}(\boldsymbol{x})) \cdot \left|\det \frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}\right|$$

- Maximize likelihood of data:

$$\theta^* = \arg\max_\theta \sum_{i=1}^{N} \log p_\theta(x_i)$$

- $\frac{\partial f_\theta^{-1}(\boldsymbol{x})}{\partial \boldsymbol{x}}$ is the Jacobian of the transformation $f_\theta^{-1}(\boldsymbol{x})$

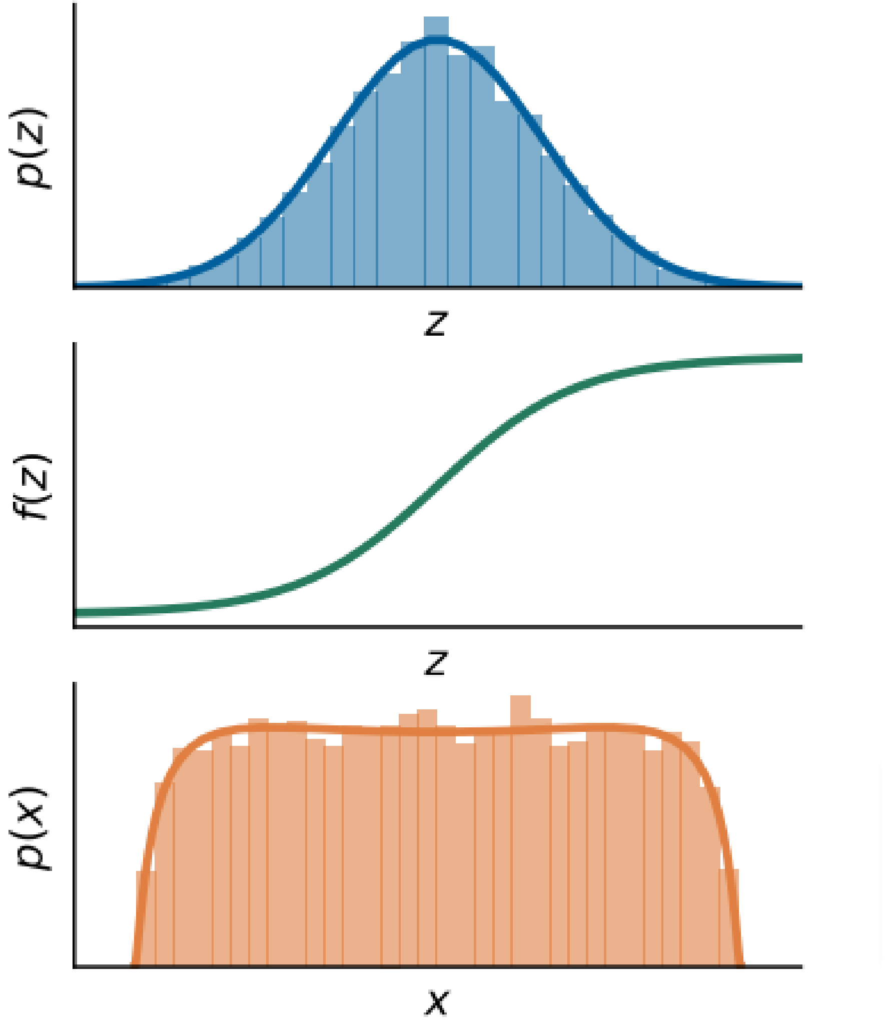


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows

- Continuously normalizing flows are a **generalization of normalizing flows** where the transformations are parameterized by continuous dynamics governed by an ordinary differential equation (ODE).

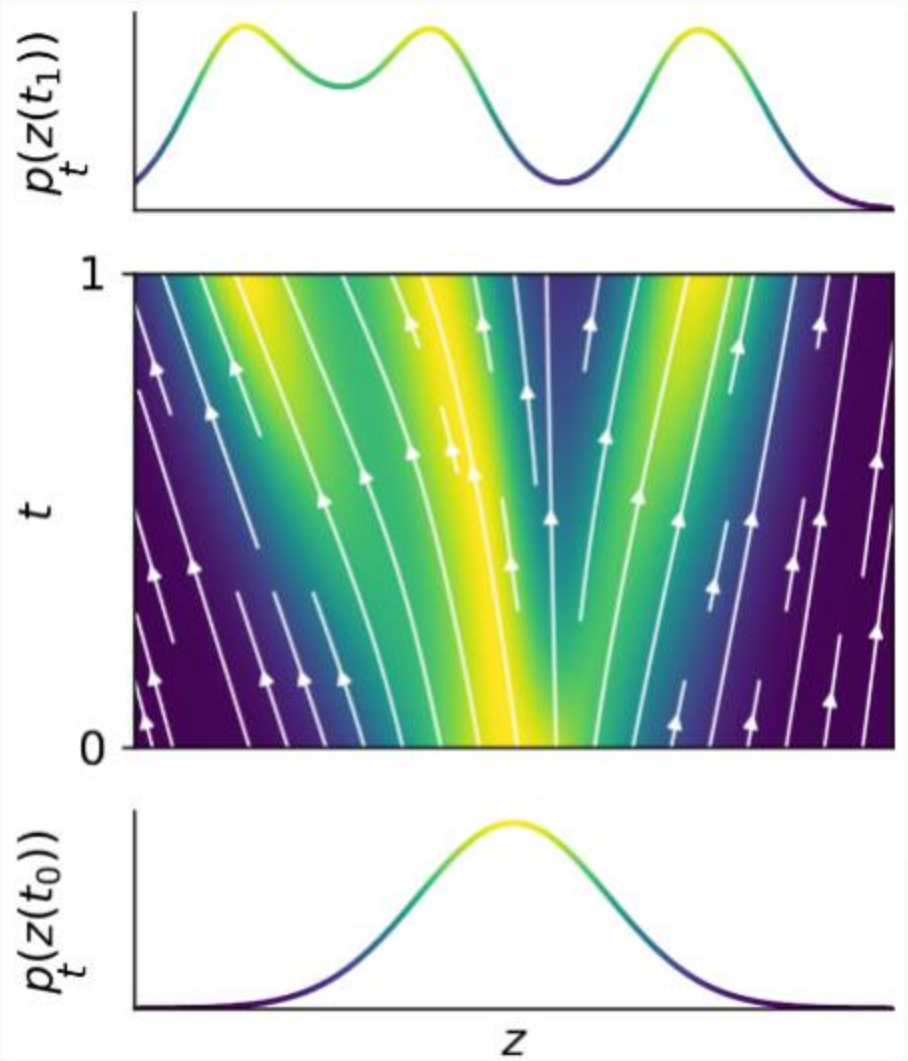


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows

- Define the transformation as an ODE

$$\boldsymbol{x} = \boldsymbol{z}(t_1) = \int_{t_0}^{t_1} \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t$$

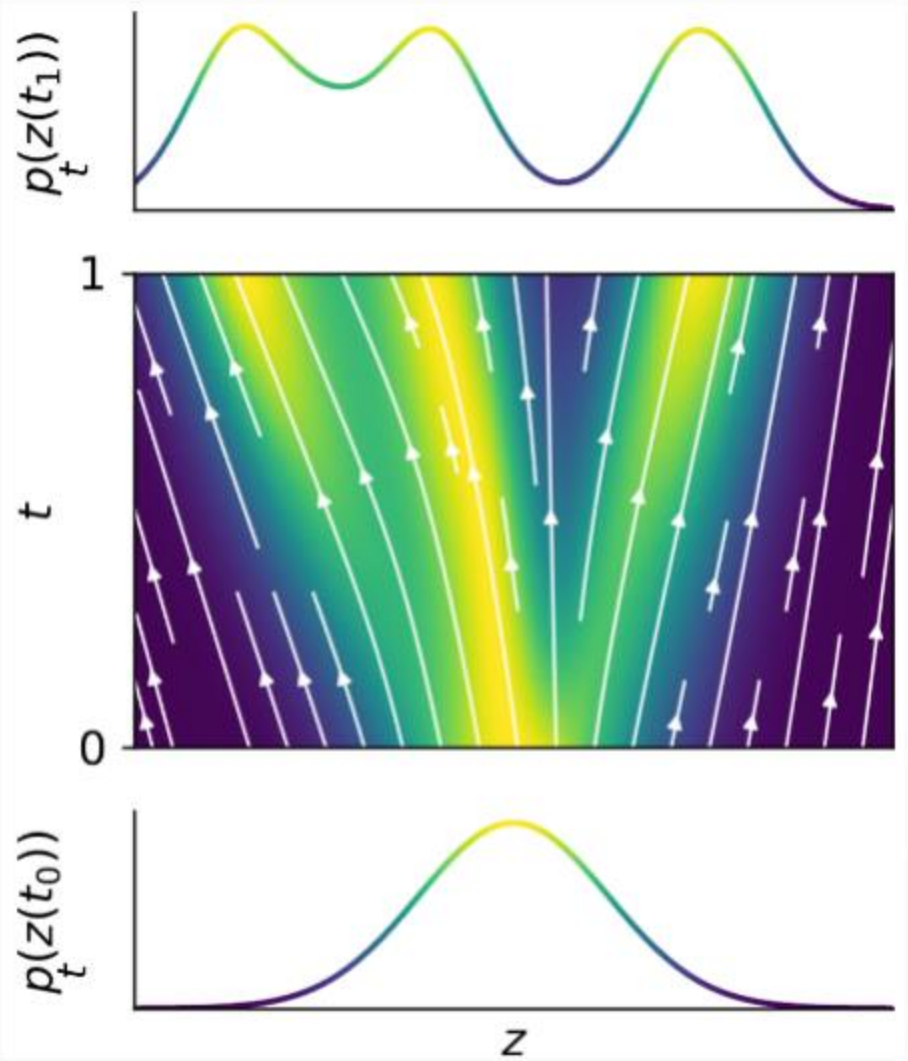


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows

- Define the transformation as an ODE

$$\boldsymbol{x} = \boldsymbol{z}(t_1) = \int_{t_0}^{t_1} \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t$$

- Here $\boldsymbol{v}_\theta(\boldsymbol{z}(t), t)$ represents the velocity field of the latent variable $\boldsymbol{z}$ as it evolves under a continuous transformation

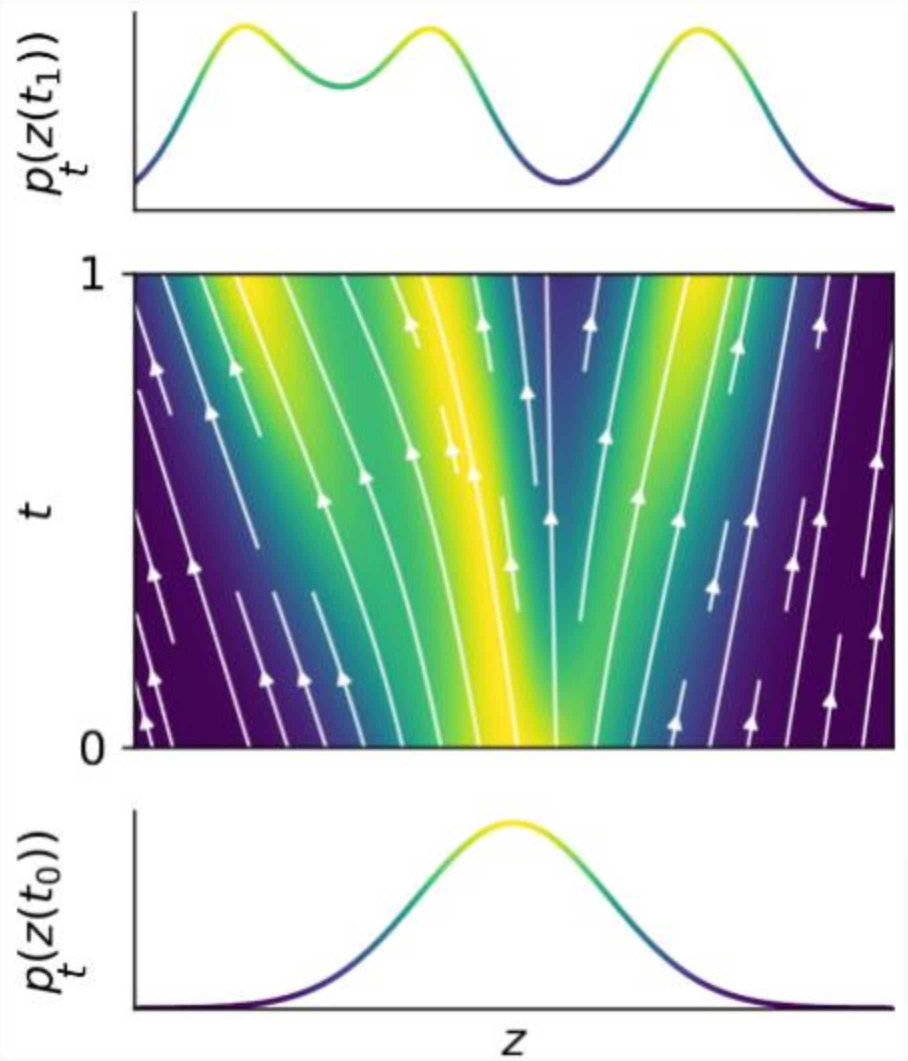


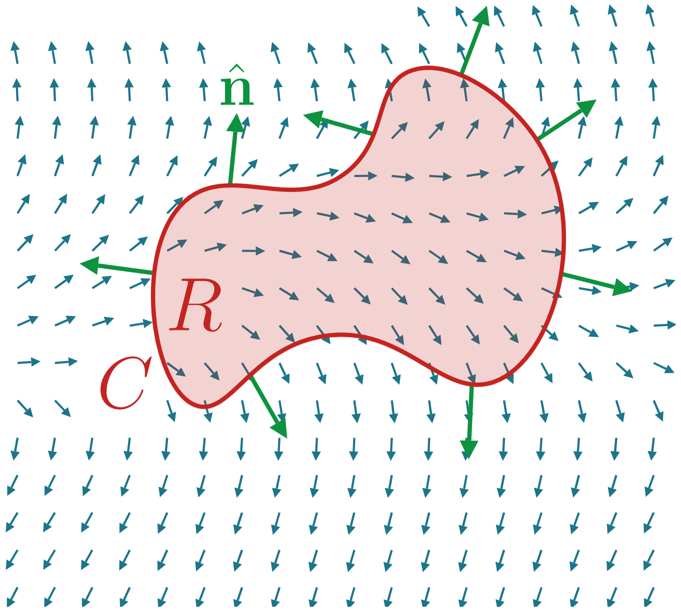
Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Outline

- Normalizing Flows and Continuous Normalizing Flows
    - **The Continuity Equation**
- The Fokker Plank Equation
- Flow matching
- Variants:
    - Batch Optimal Transport Flow Matching

Continuous Normalizing Flows

- Gauss’s Divergence Theorem: the flux of a vector field through a closed surface equals the volume integral of its divergence over the enclosed region.

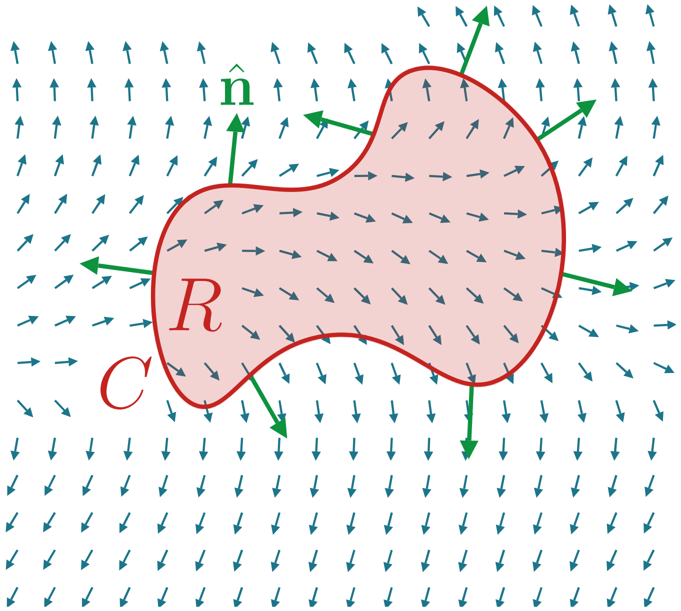


$$\underbrace{\oint (p(z(t))v_\theta(z(t),t)) \cdot \mathbf{n}\,\mathrm{d}C}_{\text{Flux integral through the boundary } C} = \underbrace{\iint_R \nabla \cdot (p(z(t))v_\theta(z(t),t))\,\mathrm{d}R}_{\text{Divergence integral over the region } R}$$

Image credit: https://www.khanacademy.org/math/multivariable-calculus/greens-theorem-and-stokes-theorem/divergence-theorem-articles/a/2d-divergence-theorem

Continuous Normalizing Flows

- Gauss’s Divergence Theorem : the flux of a vector field through a closed surface equals the volume integral of its divergence over the enclosed region.



$$\underbrace{\oint (p(z(t))v_\theta(z(t),t)) \cdot \mathbf{n}\,\mathrm{d}C}_{\text{Flux integral through the boundary } C} = \underbrace{\iint_R \nabla \cdot (p(z(t))v_\theta(z(t),t))\,\mathrm{d}R}_{\text{Divergence integral over the region } R}$$

$p(z(t))$ is the density at position $z(t)$

$v_\theta(z(t),t)$ describes the relevant flow

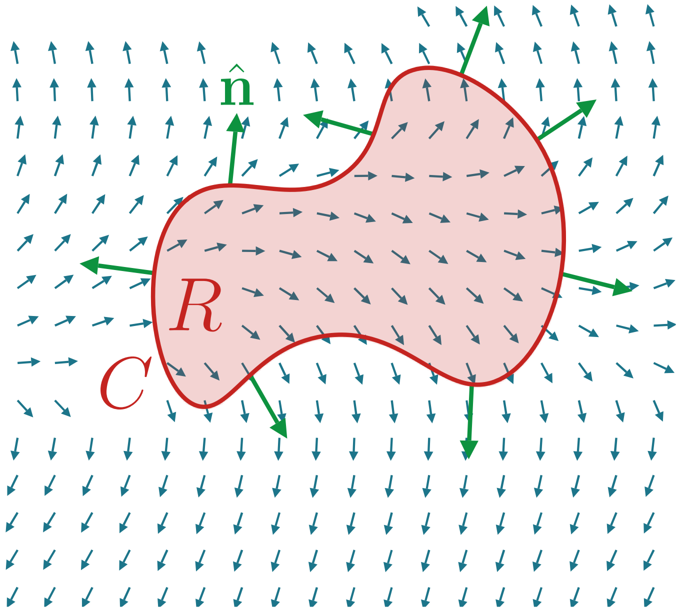
$p(z(t))v_\theta(z(t),t)$ describes how much density flows per unit time in a unit area.

Physical analogy: Think of the flow of fluid mass!

Image credit: https://www.khanacademy.org/math/multivariable-calculus/greens-theorem-and-stokes-theorem/divergence-theorem-articles/a/2d-divergence-theorem

Continuous Normalizing Flows

- Consider the law of conservation (the continuity equation):

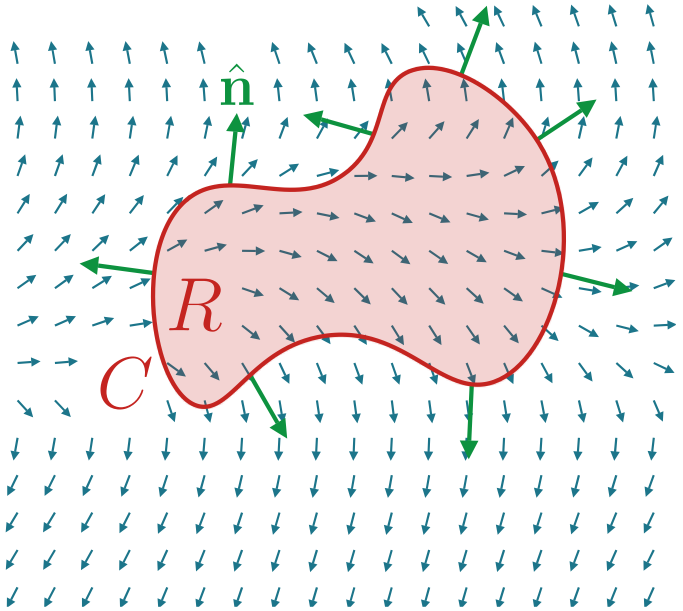


$$\underbrace{\iint_R \frac{\partial p(z(t))}{\partial t}\,\mathrm{d}R}_{\text{Flux in over the region } R} + \underbrace{\oint \left(p(z(t))v_\theta(z(t),t)\right)\cdot \mathbf{n}\,\mathrm{d}C}_{\text{Flux out through the boundary } C} = 0$$

Image credit: https://www.khanacademy.org/math/multivariable-calculus/greens-theorem-and-stokes-theorem/divergence-theorem-articles/a/2d-divergence-theorem

Continuous Normalizing Flows

- Consider the law of conservation (the continuity equation):



$$\iint_R \frac{\partial p(z(t))}{\partial t} \mathrm{d}R + \oint (p(z(t)) v_\theta(z(t), t)) \cdot n \mathrm{d}C = 0$$

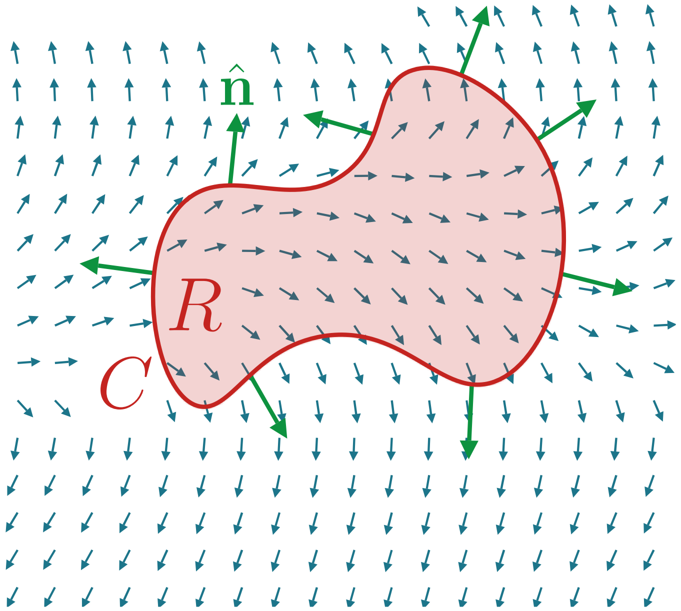
Apply Gauss's Divergence Theorem

$$\underbrace{\iint_R \frac{\partial p(z(t))}{\partial t} \mathrm{d}R}_{\text{Flux in over the region } R} + \underbrace{\iint_R \nabla \cdot (p(z(t)) v_\theta(z(t), t)) \mathrm{d}R}_{\text{Divergence integral over the region } R} = 0$$

Image credit: https://www.khanacademy.org/math/multivariable-calculus/greens-theorem-and-stokes-theorem/divergence-theorem-articles/a/2d-divergence-theorem

Continuous Normalizing Flows

- Consider the law of conservation (the continuity equation):



$$\iint_R \frac{\partial p(z(t))}{\partial t} \mathrm{d}R + \oint (p(z(t)) v_\theta(z(t), t)) \cdot \mathbf{n} \mathrm{d}C = 0$$

$$\iint_R \frac{\partial p(z(t))}{\partial t} \mathrm{d}R + \iint_R \nabla \cdot (p(z(t)) v_\theta(z(t), t)) \mathrm{d}R = 0$$

$$\boxed{\frac{\partial p(z(t))}{\partial t} + \nabla \cdot (p(z(t)) v_\theta(z(t), t)) = 0}$$

Continuity equation (differential form)

This is due to the fact that the conservation law holds for all kinds of regions, densities, and velocity fields!

Image credit: https://www.khanacademy.org/math/multivariable-calculus/greens-theorem-and-stokes-theorem/divergence-theorem-articles/a/2d-divergence-theorem

Continuous Normalizing Flows

- The continuity equation:

$$\underbrace{\frac{\partial p(\boldsymbol{z}(t))}{\partial t}}_{\text{Flux in}} + \underbrace{\nabla \cdot (p(\boldsymbol{z}(t))\boldsymbol{v}_\theta(\boldsymbol{z}(t), t))}_{\text{Flux out}} = 0$$

- The continuity equation is a **principle of conservation** in fluid dynamics and other physical systems. It states that the change in density over time is balanced by the flux of density due to the velocity field.

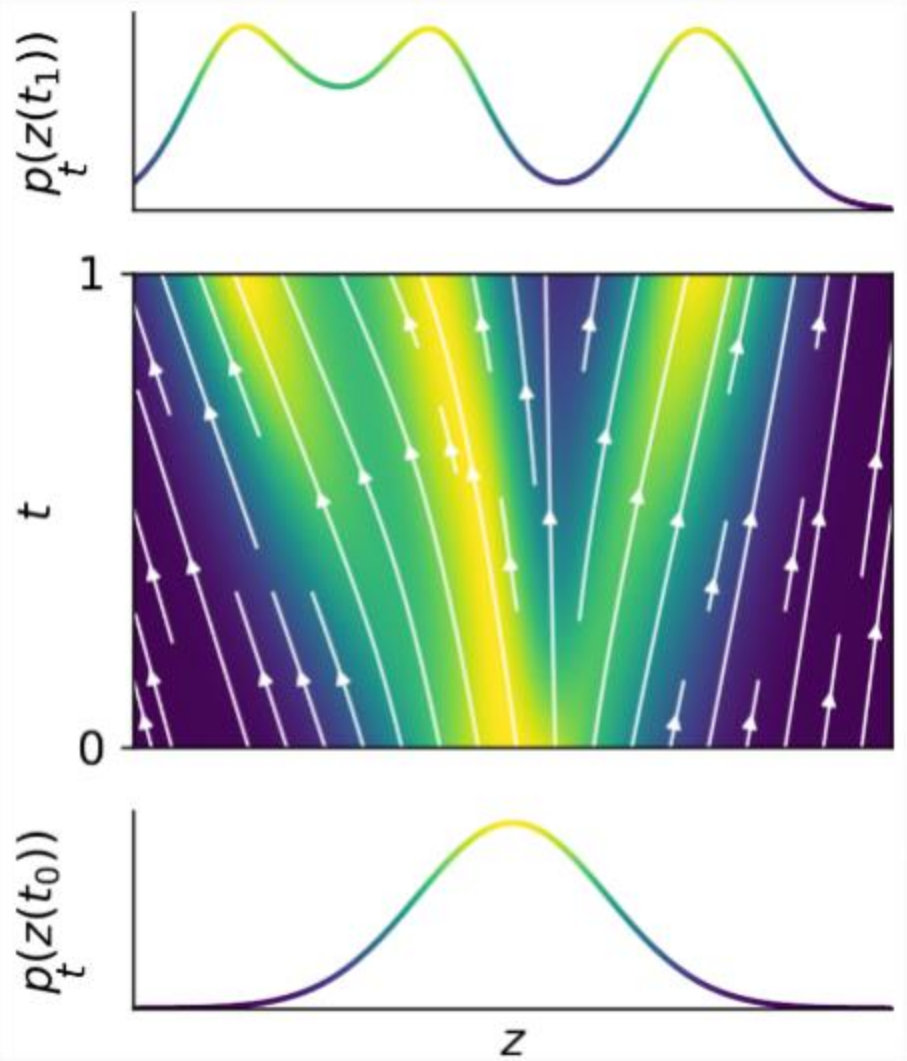


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows

- The continuity equation:

$$\underbrace{\frac{\partial p(\boldsymbol{z}(t))}{\partial t}}_{\text{Flux in}} + \underbrace{\nabla \cdot (p(\boldsymbol{z}(t))\boldsymbol{v}_\theta(\boldsymbol{z}(t), t))}_{\text{Flux out}} = 0$$

- The divergence symbol $\nabla\cdot$ measures the "net flow" of a vector field out of a point in space.

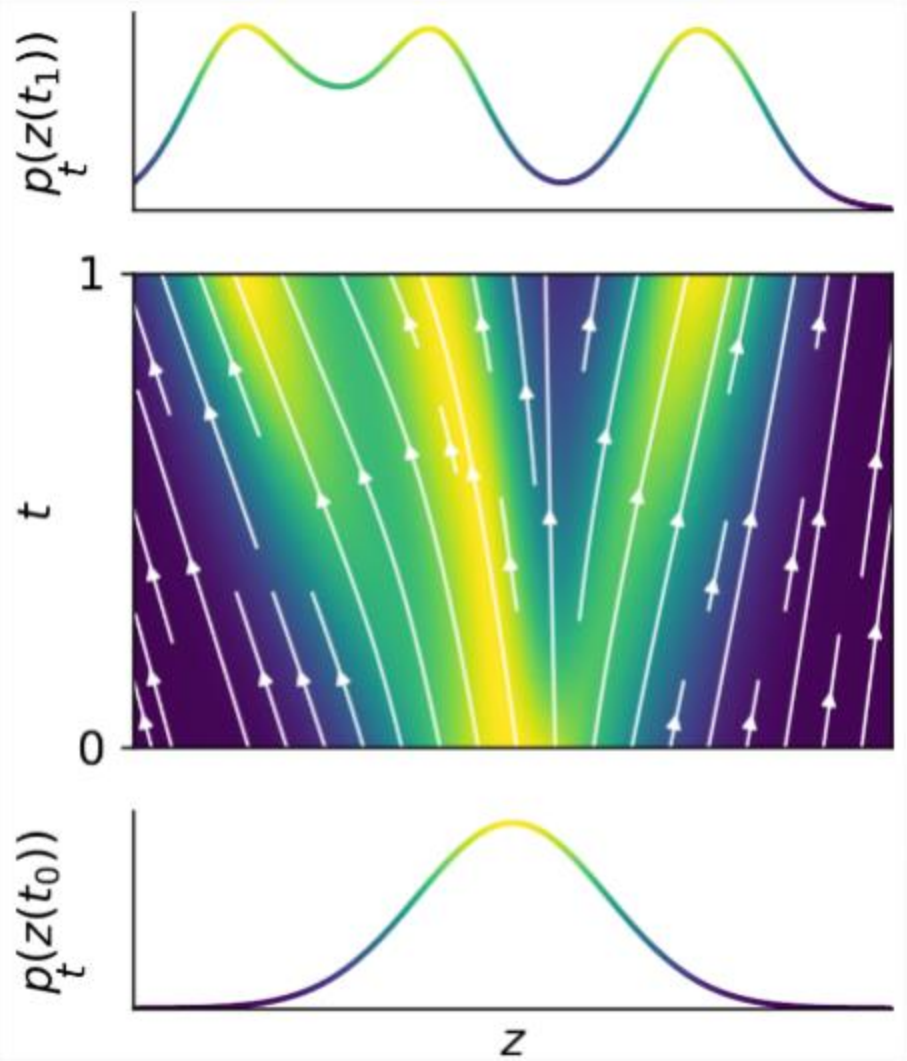


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows

- The continuity equation:

$$\underbrace{\frac{\partial p(\boldsymbol{z}(t))}{\partial t}}_{\text{Flux in}} + \underbrace{\nabla \cdot (p(\boldsymbol{z}(t))\boldsymbol{v}_\theta(\boldsymbol{z}(t), t))}_{\text{Flux out}} = 0$$

- The divergence symbol $\nabla\cdot$ measures the "net flow" of a vector field out of a point in space.
- For $\boldsymbol{v}(\boldsymbol{z}) = [v_1(\boldsymbol{z}), v_2(\boldsymbol{z}), \cdots, v_n(\boldsymbol{z})]$

$$\nabla \cdot \boldsymbol{z} = \frac{\partial v_1}{\partial z_1} + \frac{\partial v_2}{\partial z_2} + \cdots + \frac{\partial v_n}{\partial z_n}$$

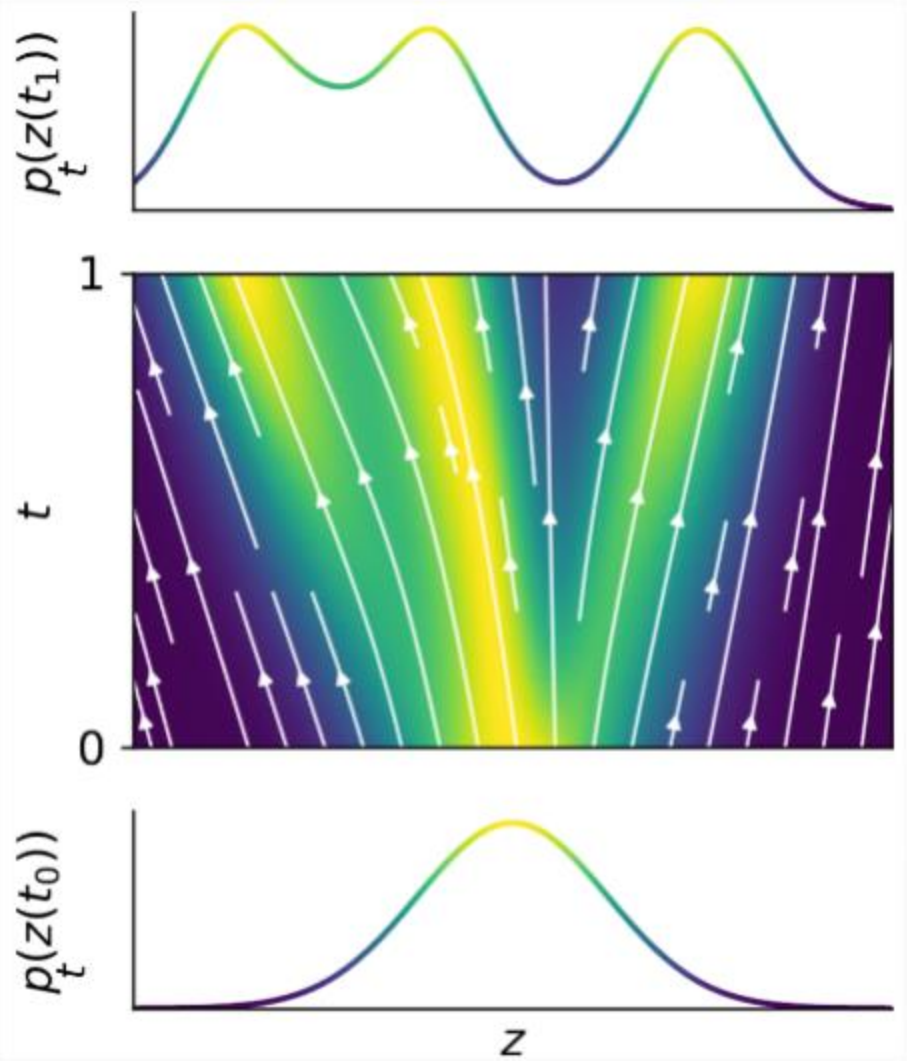


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows

- The continuity equation:

$$\underbrace{\frac{\partial p(\boldsymbol{z}(t))}{\partial t}}_{\text{Flux in}} + \underbrace{\nabla \cdot (p(\boldsymbol{z}(t))\boldsymbol{v}_\theta(\boldsymbol{z}(t), t))}_{\text{Flux out}} = 0$$

- One property of the divergence operator is the product rule:

$$\nabla \cdot (p_t(\boldsymbol{x})u_t(\boldsymbol{x})) = p_t(\boldsymbol{x})\nabla \cdot u_t(\boldsymbol{x}) + u_t(\boldsymbol{x})^T \nabla_{\boldsymbol{x}} p_t(\boldsymbol{x})$$

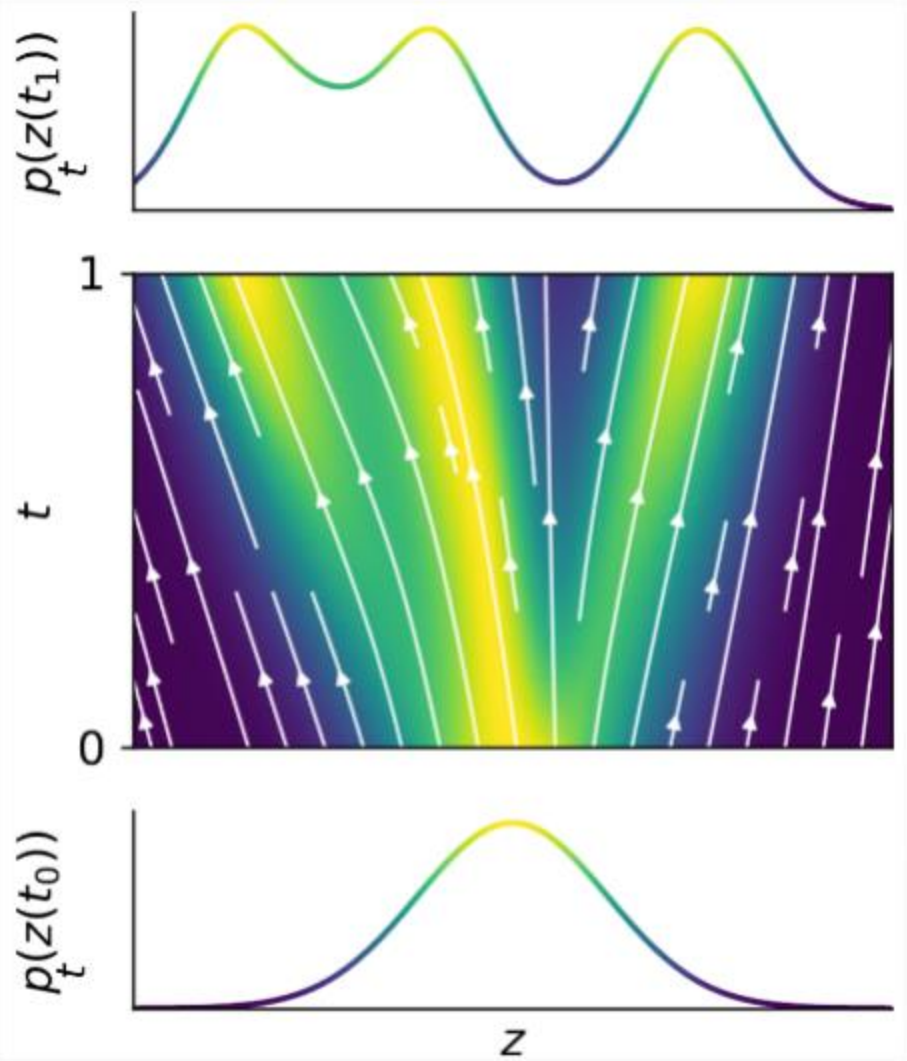


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows - Instantaneous change of density

- In CNFs, we transform a simple distribution to a more complex target distribution, and the challenge is understanding how the probability density changes during the transformation. And this change is governed by **the instantaneous change of density**.
- Here we use $\phi_t(\boldsymbol{x})$ to denote the flow trajectory.
- Consider the total derivative of $\log p_t(\phi_t(\boldsymbol{x}))$

$$
\begin{aligned}
\frac{\mathrm{d}\log p_t(\phi_t(\boldsymbol{x}))}{\mathrm{d}t} &= \frac{\partial \log p_t(\phi_t(\boldsymbol{x}))}{\partial t} \cdot \frac{\partial t}{\partial t} + \nabla_{\boldsymbol{x}} \log p_t(\phi_t(\boldsymbol{x})) \cdot \frac{\mathrm{d}\phi_t(\boldsymbol{x})}{\mathrm{d}t} \\
&= \frac{\partial \log p_t(\phi_t(\boldsymbol{x}))}{\partial t} + \nabla_{\boldsymbol{x}} \log p_t(\phi_t(\boldsymbol{x})) \cdot \frac{\mathrm{d}\phi_t(\boldsymbol{x})}{\mathrm{d}t} \\
&= \frac{\partial \log p_t(\phi_t(\boldsymbol{x}))}{\partial t} + \nabla_{\boldsymbol{x}} \log p_t(\phi_t(\boldsymbol{x})) \cdot u_t(\phi_t(x))
\end{aligned}
$$

Continuous Normalizing Flows - Instantaneous change of density

- The continuity equation with the product rule of divergence:

$$\frac{\partial}{\partial t}p_t(\phi_t(\boldsymbol{x})) + p_t(\phi_t(\boldsymbol{x}))\nabla \cdot u_t(\phi_t(\boldsymbol{x})) + u_t(\phi_t(\boldsymbol{x}))^T \nabla_{\boldsymbol{x}} p_t(\phi_t(\boldsymbol{x})) = 0$$

$$\frac{1}{p_t(\phi_t(\boldsymbol{x}))}(\frac{\partial}{\partial t}p_t(\phi_t(\boldsymbol{x})) + p_t(\phi_t(\boldsymbol{x}))\nabla \cdot u_t(\phi_t(\boldsymbol{x})) + u_t(\phi_t(\boldsymbol{x}))^T \nabla_{\boldsymbol{x}} p_t(\phi_t(\boldsymbol{x}))) = 0$$

$$\frac{\partial}{\partial t}\log p_t(\phi_t(\boldsymbol{x})) = -\nabla \cdot u_t(\phi_t(\boldsymbol{x})) - u_t(\phi_t(\boldsymbol{x}))^T \nabla_{\boldsymbol{x}} \log p_t(\phi_t(\boldsymbol{x}))$$

Continuous Normalizing Flows - Instantaneous change of density

- Consider the total derivative of $\log p_t(\phi_t(\mathbf{x}))$

$$\frac{\mathrm{d} \log p_t(\phi_t(\boldsymbol{x}))}{\mathrm{d}t} = \frac{\partial \log p_t(\phi_t(\boldsymbol{x}))}{\partial t} + \nabla_{\boldsymbol{x}} \log p_t(\phi_t(\boldsymbol{x})) \cdot u_t(\phi_t(x))$$

- The continuity equation with the product rule of divergence:

$$\frac{\partial}{\partial t} \log p_t(\phi_t(\boldsymbol{x})) = -\nabla \cdot u_t(\phi_t(\boldsymbol{x})) - u_t(\phi_t(\boldsymbol{x})) \cdot \nabla_{\boldsymbol{x}} \log p_t(\phi_t(\boldsymbol{x}))$$

- Now, replace the first term with continuity equation, we will have:

$$\log p_1(\phi_1(\boldsymbol{x})) = \log p_0(\phi_0(\boldsymbol{x})) - \int_0^1 \nabla \cdot u_t(\phi_t(\boldsymbol{x}))\mathrm{d}t$$

Continuous Normalizing Flows

- Define the transformation as an ODE

$$\boldsymbol{x} = \boldsymbol{z}(t_1) = \int_{t_0}^{t_1} \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t$$

- Instantaneous change of density

$$\frac{\partial \log p_t(\boldsymbol{z}(t))}{\partial t} = -\nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t))$$

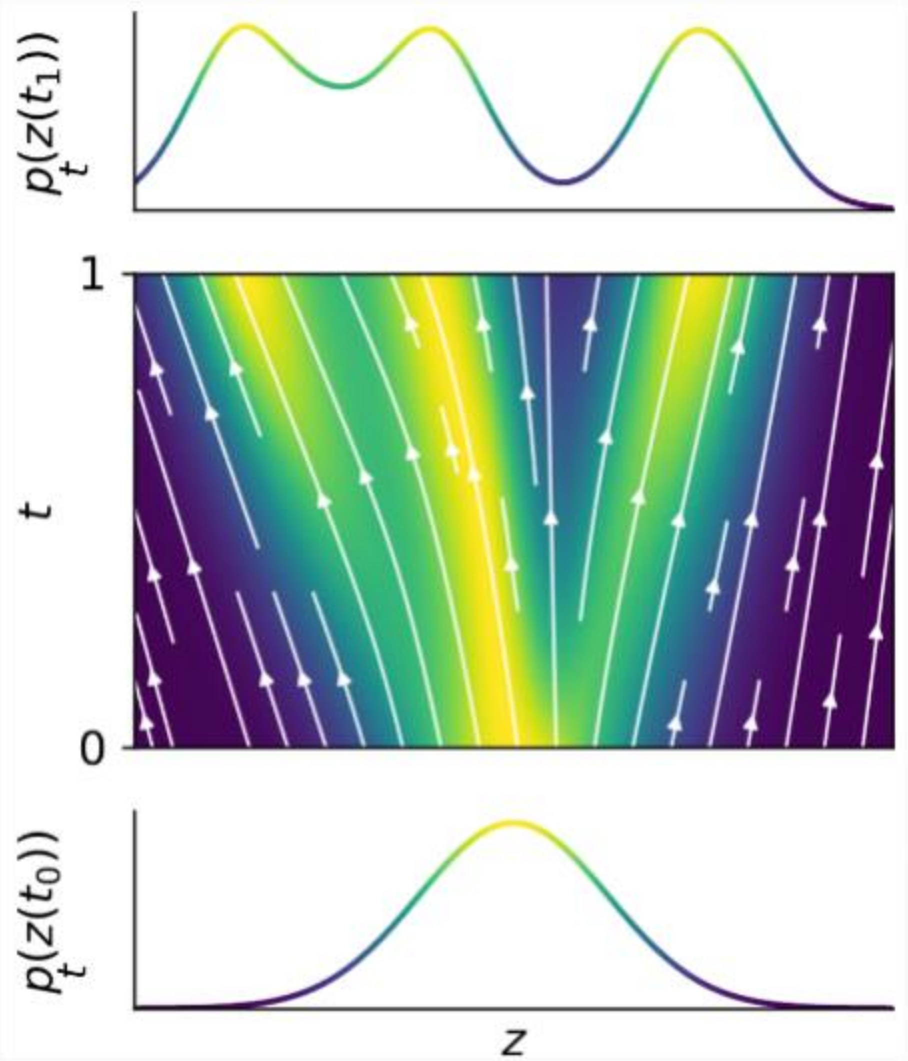


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Continuous Normalizing Flows

- Define the transformation as an ODE

$$\boldsymbol{x} = \boldsymbol{z}(t_1) = \int_{t_0}^{t_1} \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t$$

- Instantaneous change of density

$$\frac{\partial \log p_t(\boldsymbol{z}(t))}{\partial t} = -\nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t))$$

- Solve the ODE for $\log p_{t_1}(\boldsymbol{z}(t_1))$

$$\log p_{t_0}(\boldsymbol{z}(t_0)) - \int_{t_0}^{t_1} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t$$

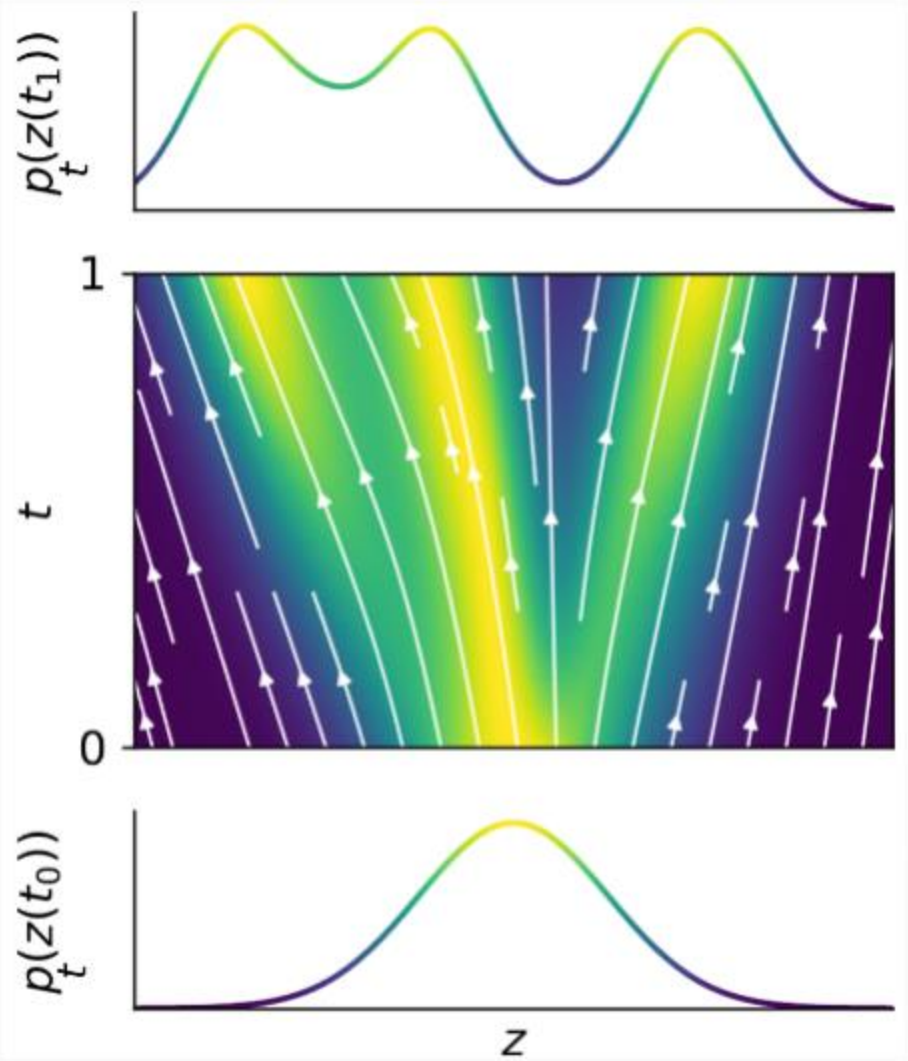


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Training of the Neural ODEs

- The ODEs parameterized by neural networks are called Neural ODEs.
- We still adopt maximum likelihood training objective.

$$\theta^* = \arg\max_{\theta} \sum_{i=1}^{N} \boxed{\log p_\theta(\boldsymbol{x}_i)}$$

Training of the Neural ODEs

- Training requires simulation (solving ODE) to obtain exact likelihood

$$\begin{aligned}\boxed{\log p_\theta(\boldsymbol{x})} &= \log p_{t_0}(\boldsymbol{z}(t_0)) - \int_{t_0}^{t_1} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t \\ &= \log p_{t_0}(\boldsymbol{z}(t_0)) + \int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t\end{aligned}$$

Training of the Neural ODEs

- Training requires simulation (solving ODE) to obtain exact likelihood

$$\begin{aligned}\log p_\theta(\boldsymbol{x}) &= \log p_{t_0}(\boldsymbol{z}(t_0)) - \int_{t_0}^{t_1} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t \\ &= \log p_{t_0}\boxed{(\boldsymbol{z}(t_0))} + \boxed{\int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t}\end{aligned}$$

Both need to be numerically solved through ODEs

Training of the Neural ODEs

- Training requires simulation (solving ODE) to obtain exact likelihood

$$\log p_\theta(\boldsymbol{x}) = \log p_{t_0}\boxed{(\boldsymbol{z}(t_0))} + \boxed{\int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t}$$

$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}}_{\text{inversed } t_1 \to t_0} \begin{bmatrix} \boldsymbol{z}_t \\ \int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t \end{bmatrix} = \begin{bmatrix} -\boldsymbol{v}_\theta(\boldsymbol{z}(t), t) \\ \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t) \end{bmatrix}$$

Training of the Neural ODEs

- Training requires simulation (solving ODE) to obtain exact likelihood

$$\log p_\theta(\boldsymbol{x}) = \log p_{t_0}\boxed{(\boldsymbol{z}(t_0))} + \boxed{\int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t}$$

$$\underbrace{\frac{\mathrm{d}}{\mathrm{d}\tilde{t}}}_{\text{inversed } t_1 \to t_0} \begin{bmatrix} \boldsymbol{z}_t \\ \int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t \end{bmatrix} = \begin{bmatrix} -\boldsymbol{v}_\theta(\boldsymbol{z}(t), t) \\ \boxed{\nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)} \end{bmatrix}$$

Trace of Jacobian.
We can use Hutchinson’s trace estimator.

Training of the Neural ODEs

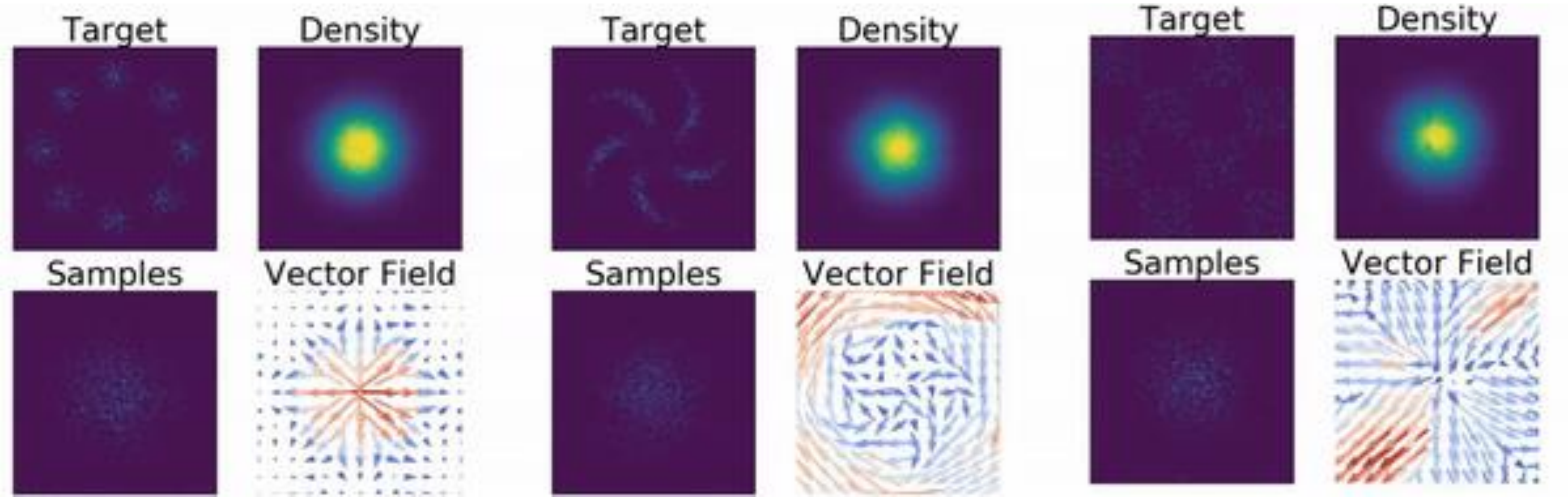
- Training requires simulation (solving ODE) to obtain exact likelihood

$$\log p_\theta(\boldsymbol{x}) = \log p_{t_0}(\boldsymbol{z}(t_0)) + \int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t$$

$$\frac{\mathrm{d}}{\mathrm{d}\tilde{t}} \begin{bmatrix} \boldsymbol{z}_t \\ \int_{t_1}^{t_0} \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t)\mathrm{d}t \end{bmatrix} = \begin{bmatrix} -\boldsymbol{v}_\theta(\boldsymbol{z}(t), t) \\ \nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t) \end{bmatrix}$$

- Solving ODEs numerically at each training iteration is slow!
- Gradient computation for backpropagation requires careful handling (adjoint method).

Continuous Normalizing Flows



Outline

- Normalizing Flows and Continuous Normalizing Flows
    - The Continuity Equation
- **The Fokker Plank Equation**
- Flow matching
- Variants:
    - Batch Optimal Transport Flow Matching

The Fokker Plank Equation

- What happens to the continuity equation if there is stochastic noise?

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{u}(\boldsymbol{x}, t)\mathrm{d}t + \sigma(\boldsymbol{x}, t)\mathrm{d}\boldsymbol{W}_t$$

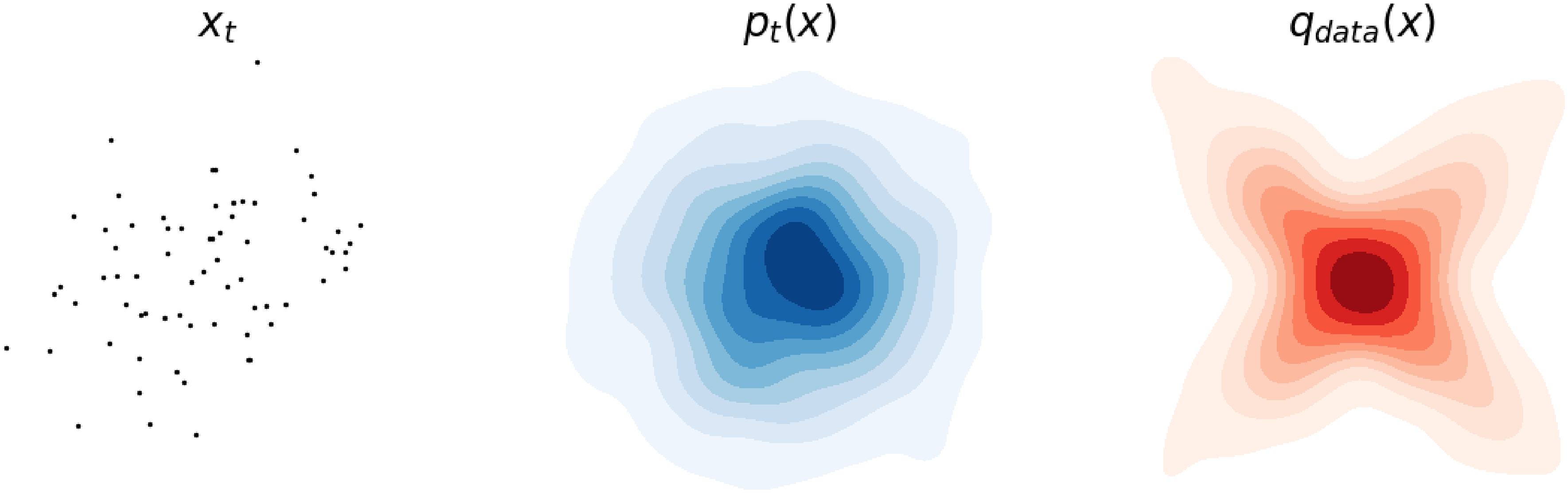


Image credit: Das, "Building Diffusion Model's theory from ground up", ICLR Blogposts, 2024.

The Fokker Plank Equation

- What happens to the continuity equation if there is stochastic noise?

$$\mathrm{d}\boldsymbol{x} = \underbrace{\boldsymbol{u}(\boldsymbol{x},t)\mathrm{d}t}_{\substack{\text{Drift term}\\ \text{(deterministic)}}} + \underbrace{\sigma(\boldsymbol{x},t)\underbrace{\boxed{\mathrm{d}\boldsymbol{W}_t}}_{\text{Wiener process}}}_{\substack{\text{Diffusion term}\\ \text{(stochastic)}}}$$

- The ODE now becomes a *stochastic differential equation* (SDEs).

The Fokker Plank Equation

- What defines the Wiener process (aka Brownian motion)?

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{u}(\boldsymbol{x}, t)\mathrm{d}t + \sigma(\boldsymbol{x}, t)\boxed{\mathrm{d}\boldsymbol{W}_t}$$

- Its increments are independent Gaussians.

$$\forall t, u > 0, s < t$$
$$(\boldsymbol{W}_{t+u} - \boldsymbol{W}_t) \sim \mathcal{N}(\boldsymbol{0}, u\boldsymbol{I})$$
$$(\boldsymbol{W}_{t+u} - \boldsymbol{W}_t) \perp \boldsymbol{W}_s$$
$$\boldsymbol{W}_0 = \boldsymbol{0}$$

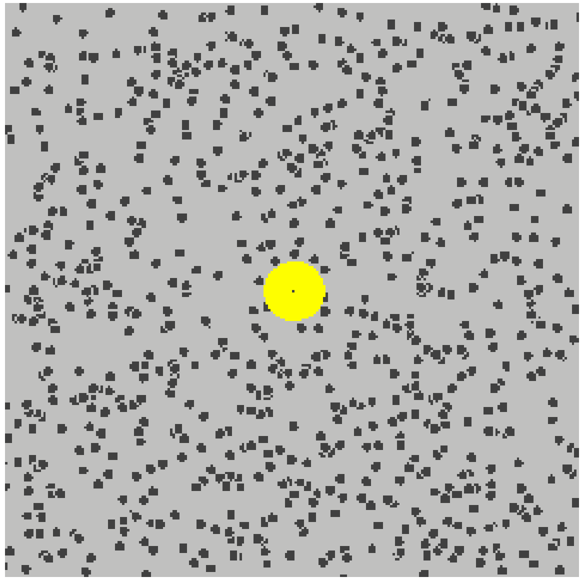


Image credit: https://en.wikipedia.org/wiki/Brownian_motion

The Fokker Plank Equation

- What defines the Wiener process (aka Brownian motion)?

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{u}(\boldsymbol{x}, t)\mathrm{d}t + \sigma(\boldsymbol{x}, t)\boxed{\mathrm{d}\boldsymbol{W}_t}$$

- Increment in infinitesimal time interval is Gaussian.

$$\mathrm{d}\boldsymbol{W}_t \sim \mathcal{N}(0, \mathrm{d}t\boldsymbol{I})$$

$$\boldsymbol{W}_{t+\Delta t} - \boldsymbol{W}_t \approx \sqrt{\Delta t}\boldsymbol{\epsilon}$$

$$\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$$

The Fokker Plank Equation

- How does $p(\boldsymbol{x}, t)$ change w.r.t. time if $\boldsymbol{x}$ is governed by the SDE?

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{u}(\boldsymbol{x}, t)\mathrm{d}t + \sigma(\boldsymbol{x}, t)\mathrm{d}\boldsymbol{W}_t$$

- This is given by the famous Fokker-Plank equation:

$$\frac{\partial}{\partial t} p(\boldsymbol{x}, t) = -\nabla \cdot [\boldsymbol{u}(\boldsymbol{x}, t) p(\boldsymbol{x}, t)] + \nabla^2 \cdot \left[\frac{\sigma^2(\boldsymbol{x}, t)}{2} p(\boldsymbol{x}, t)\right]$$

  - Also known as the Kolmogorov forward equation.
  - The initial distribution at $t = 0$ must be known.

The Fokker Plank Equation

- SDEs:

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{u}(\boldsymbol{x}, t)\mathrm{d}t + \sigma(\boldsymbol{x}, t)\mathrm{d}\boldsymbol{W}_t$$

$$\frac{\partial}{\partial t} p(\boldsymbol{x}, t) = -\nabla \cdot [\boldsymbol{u}(\boldsymbol{x}, t) p(\boldsymbol{x}, t)] + \nabla^2 \cdot \left[\frac{\sigma^2(\boldsymbol{x}, t)}{2} p(\boldsymbol{x}, t)\right]$$

- ODEs:

$$\mathrm{d}\boldsymbol{x} = \boldsymbol{u}(\boldsymbol{x}, t)\mathrm{d}t$$

$$\frac{\partial}{\partial t} p(\boldsymbol{x}, t) = -\nabla \cdot [\boldsymbol{u}(\boldsymbol{x}, t) p(\boldsymbol{x}, t)]$$

Outline

- Normalizing Flows and Continuous Normalizing Flows
  - The Continuity Equation
- The Fokker Plank Equation
- **Flow matching**
- Variants:
  - Batch Optimal Transport Flow Matching

Flow Matching Model

- Motivation: Recall the continuity equation, which shows how the probability and the velocity field is coupled, since probability density is conserved as it flows through the space.

$$\frac{\partial p(\boldsymbol{z}(t))}{\partial t} + \nabla \cdot (p(\boldsymbol{z}(t))\boldsymbol{v}_\theta(\boldsymbol{z}(t), t)) = 0$$

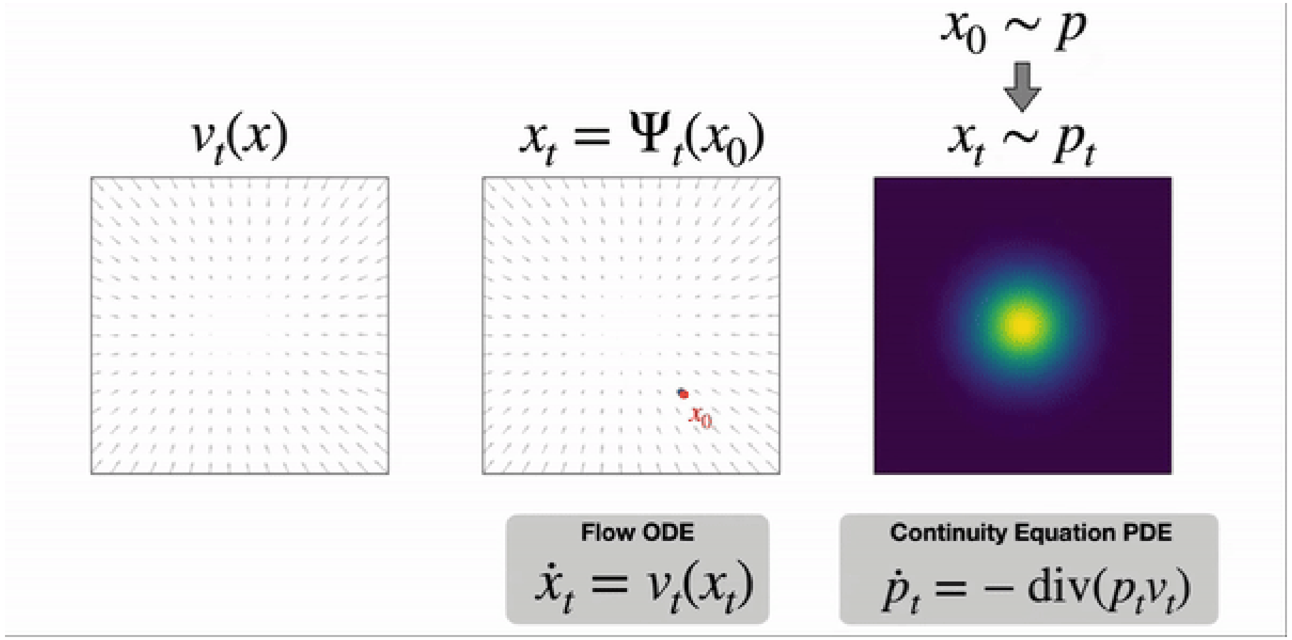


Image credit: Lecture slides by Yaron Lipman, https://drive.google.com/file/d/1Dkl_NEo1YpoDByxJLuNbqqA493-4NdCY/view

Flow Matching Model

- We want to have a loss for this generative model that is differentiable and tractable. Here we can try to minimize the distance between the target distribution p_1 and the data distribution q by minimizing the KL Divergence:

$$D_{KL}(q||p_1) = -\mathbb{E}_{\boldsymbol{x}\sim q} \log p_1(\boldsymbol{x}) + c$$

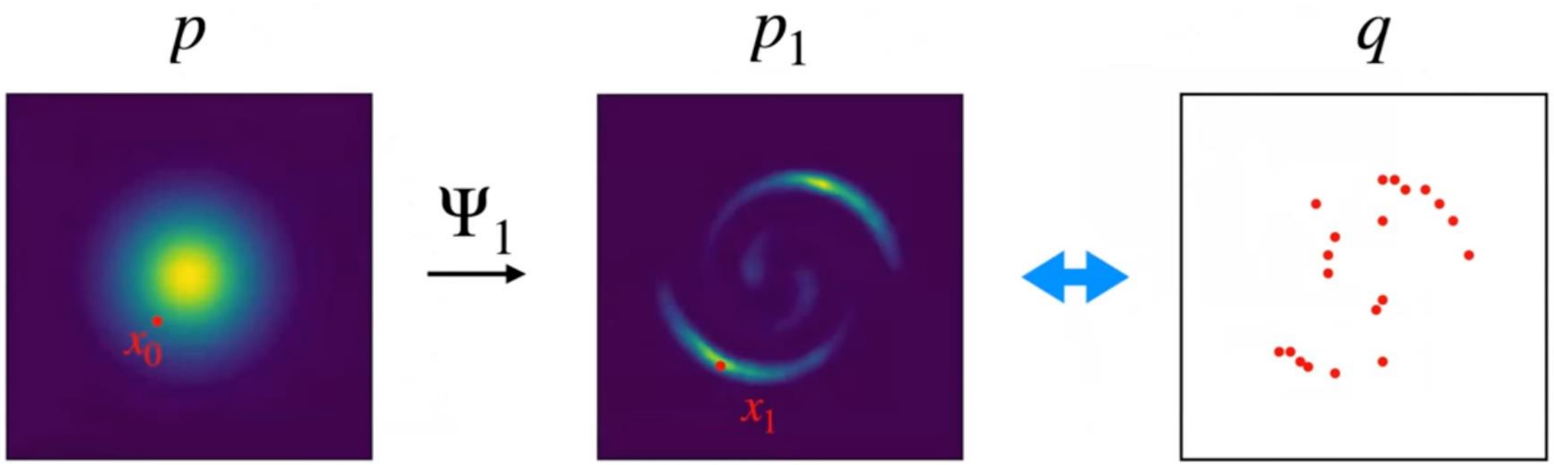


Image credit: Lecture slides by Yaron Lipman, https://drive.google.com/file/d/1Dkl_NEo1YpoDByxJLuNbqqA493-4NdCY/view

Flow Matching Model

- We want to have a loss for this generative model that is differentiable and tractable. Here we can try to minimize the distance between the target distribution p_1 and the data distribution q by minimizing the KL Divergence:

$$D_{KL}(q||p_1) = -\mathbb{E}_{\boldsymbol{x}\sim q}\boxed{\log p_1(\boldsymbol{x})} + c$$

$$\frac{\partial \log p_t(\boldsymbol{z}(t))}{\partial t} = -\nabla \cdot \boldsymbol{v}_\theta(\boldsymbol{z}(t), t))$$

Need simulation if use instantaneous change of variable

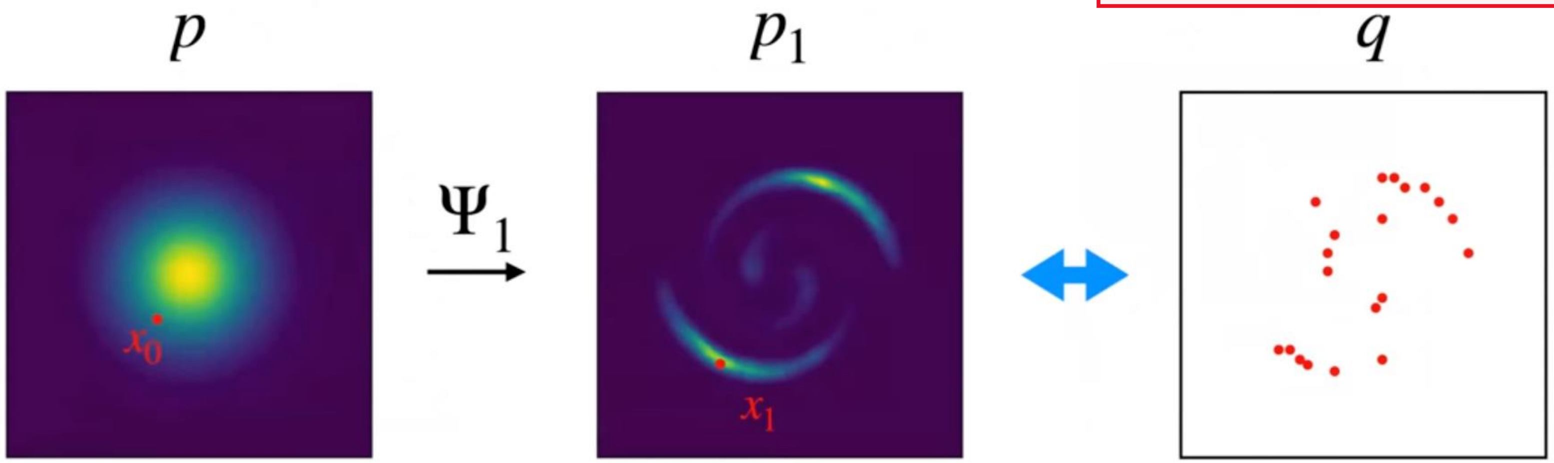


Image credit: Lecture slides by Yaron Lipman, https://drive.google.com/file/d/1Dkl_NEo1YpoDByxJLuNbqqA493-4NdCY/view

Flow Matching Model

- So, revisit the continuity equation, we can observe that based on a known vector field, we could know how the density evolves.

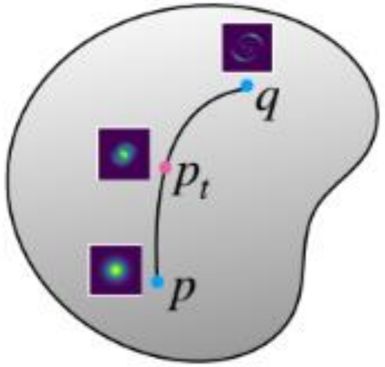


Image credit: Lecture slides by Yaron Lipman, https://drive.google.com/file/d/1Dkl_NEo1YpoDByxJLuNbqqA493-4NdCY/view

Flow Matching Model

- Therefore, instead of directly optimizing the probability density path, we can optimize the vector field.

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t,p_t(\boldsymbol{x})} \left\| \boldsymbol{v}_t(\boldsymbol{x}) - \boldsymbol{u}_t(\boldsymbol{x}) \right\|^2 ,$$

Flow Matching Model

- Therefore, instead of directly optimizing the probability density path, we can optimize the vector field.

$$\mathcal{L}_{FM}(\theta) = \mathbb{E}_{t,p_t(\boldsymbol{x})} \left\| \boldsymbol{v}_t(\boldsymbol{x}) - \boldsymbol{u}_t(\boldsymbol{x}) \right\|^2 ,$$

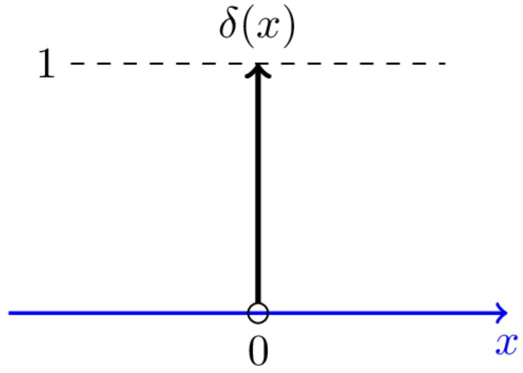
- However, we still cannot compute this loss because we don’t know p_t or u_t.

Flow Matching Model

- We need to find a tractable loss.
- Assume we have samples from data distribution $q(\boldsymbol{x}_1)$, construct conditional probability paths $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$, and marginalize the probability over data distribution:

$$p_t(\boldsymbol{x}) = \int p_t(\boldsymbol{x}|\boldsymbol{x}_1)q(\boldsymbol{x}_1)\mathrm{d}\boldsymbol{x}_1$$

$$\begin{cases} p_0(\boldsymbol{x}|\boldsymbol{x}_1) = p(\boldsymbol{x}_0) & t = 0 \\ p_1(\boldsymbol{x}|\boldsymbol{x}_1) \simeq \delta(\boldsymbol{x}_1) & t = 1 \end{cases}$$

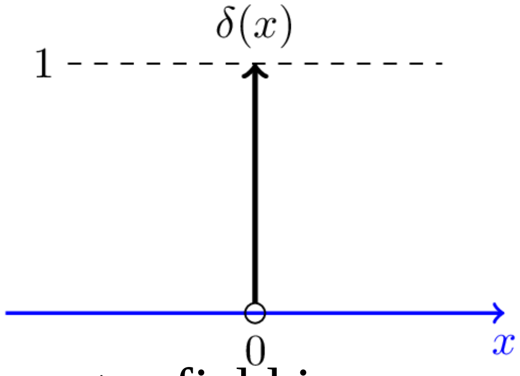


Flow Matching Model

- We need to find a tractable loss.
- Assume we have samples from data distribution $q(\boldsymbol{x}_1)$, construct conditional probability paths $p_t(\boldsymbol{x}|\boldsymbol{x}_1)$, and marginalize the probability over data distribution:

$$p_t(\boldsymbol{x}) = \int p_t(\boldsymbol{x}|\boldsymbol{x}_1)q(\boldsymbol{x}_1)\mathrm{d}\boldsymbol{x}_1$$

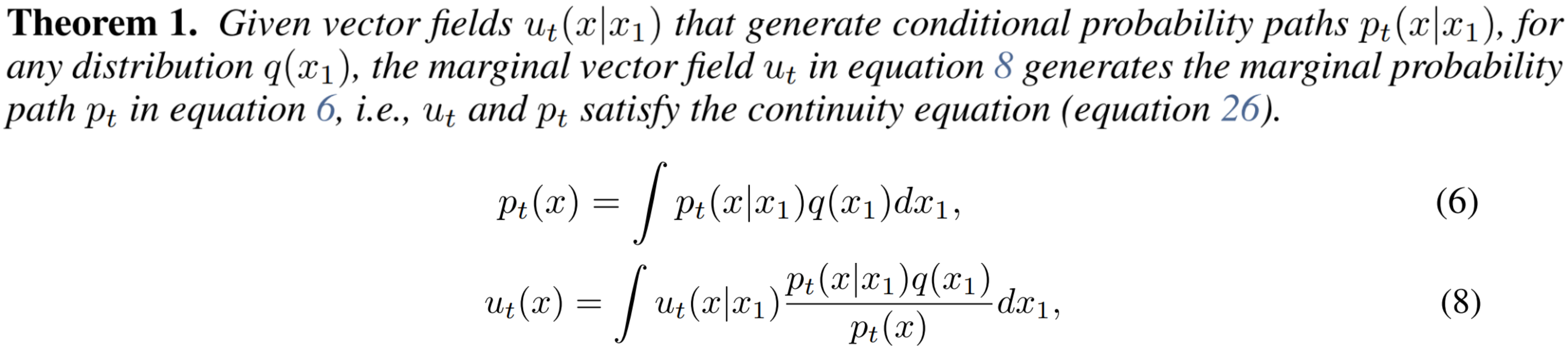
$$\begin{cases} p_0(\boldsymbol{x}|\boldsymbol{x}_1) = p(\boldsymbol{x}_0) & t = 0 \\ p_1(\boldsymbol{x}|\boldsymbol{x}_1) \simeq \delta(\boldsymbol{x}_1) & t = 1 \end{cases}$$



- The conditional vector field is $u_t(\boldsymbol{x}|\boldsymbol{x_1})$ and the marginal vector field is:

$$u_t(\boldsymbol{x}) = \int u_t(\boldsymbol{x}|\boldsymbol{x_1})\frac{p_t(\boldsymbol{x}|\boldsymbol{x_1})q(\boldsymbol{x_1})}{p_t(\boldsymbol{x})}\mathrm{d}\boldsymbol{x_1}$$

Flow Matching Model



Theorem 1. *Given vector fields $u_t(x|x_1)$ that generate conditional probability paths $p_t(x|x_1)$, for any distribution $q(x_1)$, the marginal vector field u_t in equation 8 generates the marginal probability path p_t in equation 6, i.e., u_t and p_t satisfy the continuity equation (equation 26).*

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1, \tag{6}$$

$$u_t(x) = \int u_t(x|x_1)\frac{p_t(x|x_1)q(x_1)}{p_t(x)}dx_1, \tag{8}$$

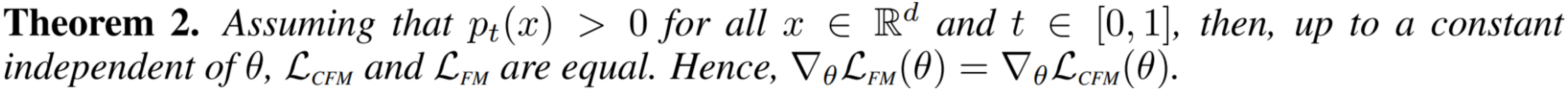
Flow Matching Model

- The conditional flow matching loss:

$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{t,q(\boldsymbol{x_1}),p_t(\boldsymbol{x}|\boldsymbol{x_1})} \left\| v_t(\boldsymbol{x}) - u_t(\boldsymbol{x}|\boldsymbol{x_1}) \right\|^2$$

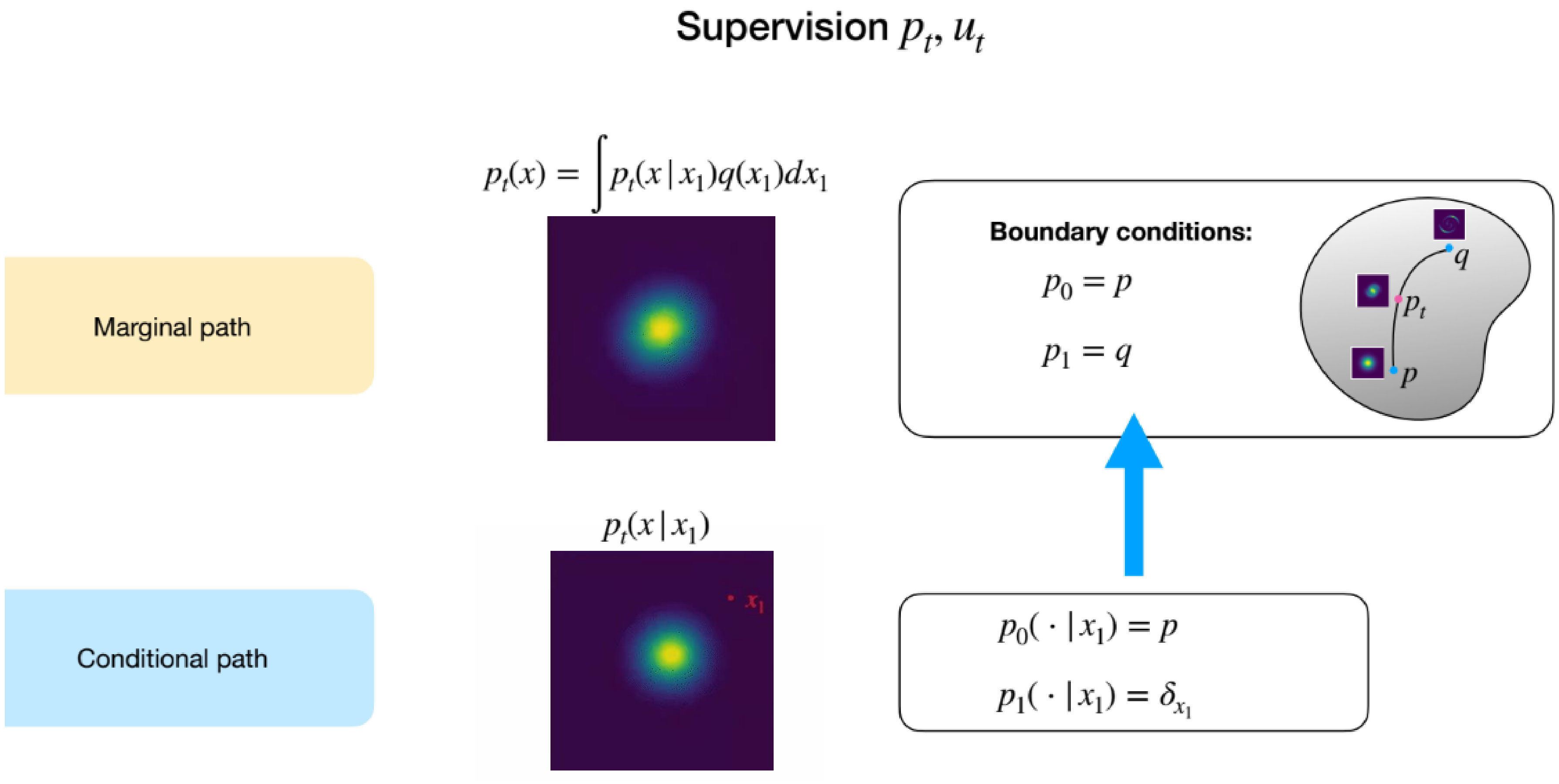
- Performing regression on conditional velocities has the same gradient as the flow matching loss.

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



Theorem 2. *Assuming that $p_t(x) > 0$ for all $x \in \mathbb{R}^d$ and $t \in [0,1]$, then, up to a constant independent of θ, $\mathcal{L}_{\text{CFM}}$ and $\mathcal{L}_{\text{FM}}$ are equal. Hence, $\nabla_\theta \mathcal{L}_{\text{FM}}(\theta) = \nabla_\theta \mathcal{L}_{\text{CFM}}(\theta)$.*

Flow Matching Model



Supervision p_t, u_t

$$p_t(x) = \int p_t(x|x_1)q(x_1)dx_1$$

Marginal path

Boundary conditions:

$p_0 = p$

$p_1 = q$

$p_t(x|x_1)$

Conditional path

$p_0(\cdot|x_1) = p$

$p_1(\cdot|x_1) = \delta_{x_1}$

Image credit: Lecture slides by Yaron Lipman, https://drive.google.com/file/d/1Dkl_NEo1YpoDByxJLuNbqqA493-4NdCY/view

Flow Matching Model

- The conditional flow based on the conditional vector field is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Psi_t(\boldsymbol{x}) = u_t\left(\Psi_t(\boldsymbol{x})|\boldsymbol{x}_1\right)$$

Flow Matching Model

- The conditional flow based on the conditional vector field is:

$$\frac{\mathrm{d}}{\mathrm{d}t}\Psi_t(\boldsymbol{x}) = u_t\left(\Psi_t(\boldsymbol{x})|\boldsymbol{x}_1\right)$$

- Simply, let $p_t(\boldsymbol{x}|\boldsymbol{x}_1) = N(\boldsymbol{x}|\mu_t(\boldsymbol{x}), \sigma_t(\boldsymbol{x})^2 I)$ and $\Psi_t(\boldsymbol{x}) = \sigma_t(\boldsymbol{x}_1)\boldsymbol{x} + u_t(\boldsymbol{x}_1)$

  , where $\sigma_0(\boldsymbol{x}_1) = 1, \sigma_1(\boldsymbol{x}_1) = \sigma_{min}, \mu_0(\boldsymbol{x}_1) = 0, \mu_1(\boldsymbol{x}_1) = \boldsymbol{x}_1$. Here we introduce σ_{min} to mimic $\delta(\boldsymbol{x}_1)$ without losing smoothness when t is close to 1.

$$\Psi_t(\boldsymbol{x}) = \sigma_t(\boldsymbol{x}_1)\boldsymbol{x} + u_t(\boldsymbol{x}_1)$$

Flow Matching Model

- Specifically, the mean and standard deviation change linearly with time:

$$\mu_t(\boldsymbol{x}) = t\boldsymbol{x}_1, \quad \text{and} \quad \sigma_t(\boldsymbol{x}) = 1 - (1 - \sigma_{min})t$$

Flow Matching Model

- Specifically, the mean and standard deviation change linearly with time:

$$\mu_t(\boldsymbol{x}) = t\boldsymbol{x}_1, \quad \text{and} \quad \sigma_t(\boldsymbol{x}) = 1 - (1 - \sigma_{min})t$$

- This gives a straight path:

$$u_t(\boldsymbol{x}|\boldsymbol{x}_1) = \frac{\boldsymbol{x}_1 - (1 - \sigma_{min})\boldsymbol{x}}{1 - (1 - \sigma_{min})t}$$

Flow Matching Model

- The conditional flow with optimal transport is:

$$\Psi_t(x) = [1 - (1 - \sigma_{min})t]x + tx_1$$

Flow Matching Model

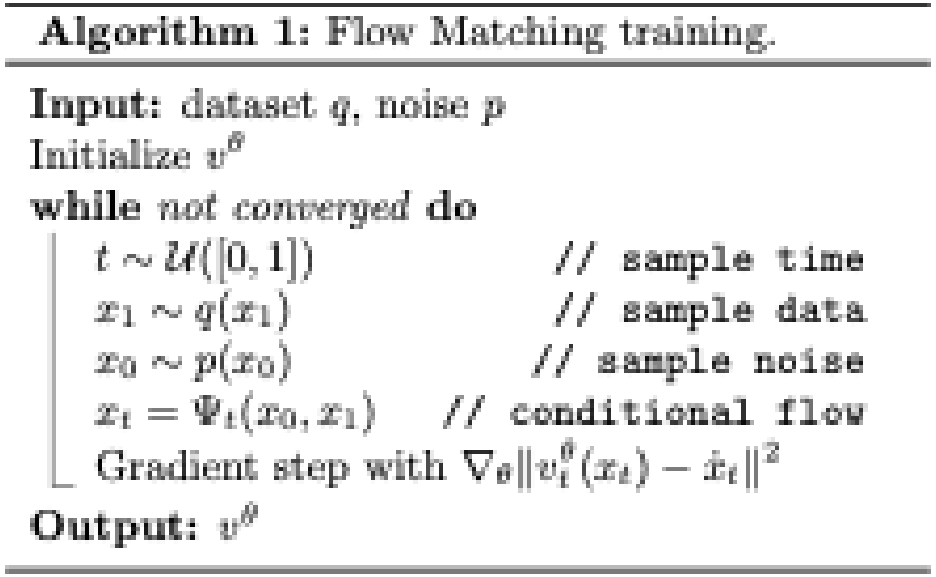
- The conditional flow with optimal transport is:

$$\Psi_t(\boldsymbol{x}) = [1 - (1 - \sigma_{min})t]\boldsymbol{x} + t\boldsymbol{x}_1$$

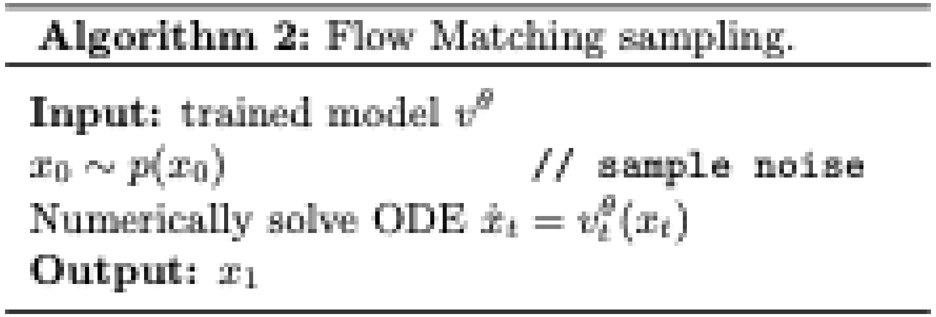
- The reparametrized conditional flow matching loss is:

$$\mathbb{E}_{t,q(\boldsymbol{x}_1),p(\boldsymbol{x}_0)} \left\| v_t(\Psi_t(\boldsymbol{x}_0|\boldsymbol{x}_1)) - (\boldsymbol{x}_1 - (1 - \sigma_{min})\boldsymbol{x}_0) \right\|^2 .$$

Flow Matching Model



```
Algorithm 1: Flow Matching training.
Input: dataset $q$, noise $p$
Initialize $v^\theta$
while not converged do
    $t \sim \mathcal{U}([0,1])$                      // sample time
    $x_1 \sim q(x_1)$                               // sample data
    $x_0 \sim p(x_0)$                               // sample noise
    $x_t = \Psi_t(x_0, x_1)$                        // conditional flow
    Gradient step with $\nabla_\theta \|v_t^\theta(x_t) - \dot{x}_t\|^2$
Output: $v^\theta$
```



```
Algorithm 2: Flow Matching sampling.
Input: trained model $v^\theta$
$x_0 \sim p(x_0)$                                   // sample noise
Numerically solve ODE $\dot{x}_t = v_t^\theta(x_t)$
Output: $x_1$
```

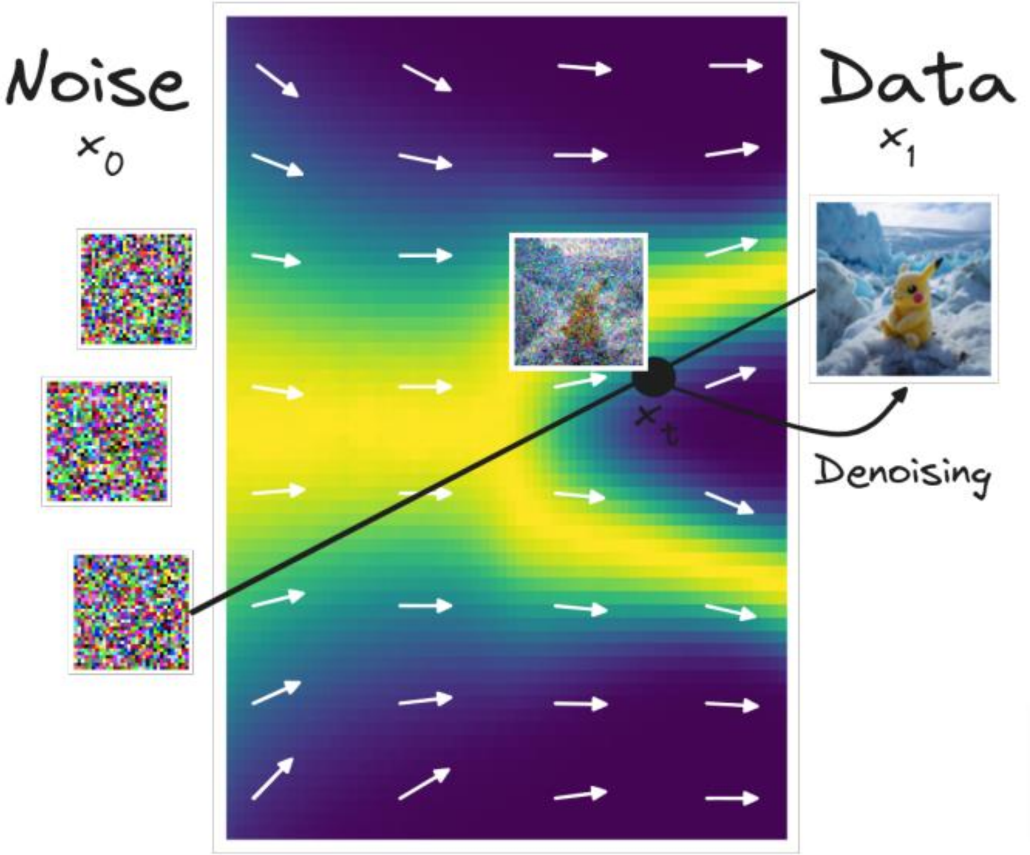
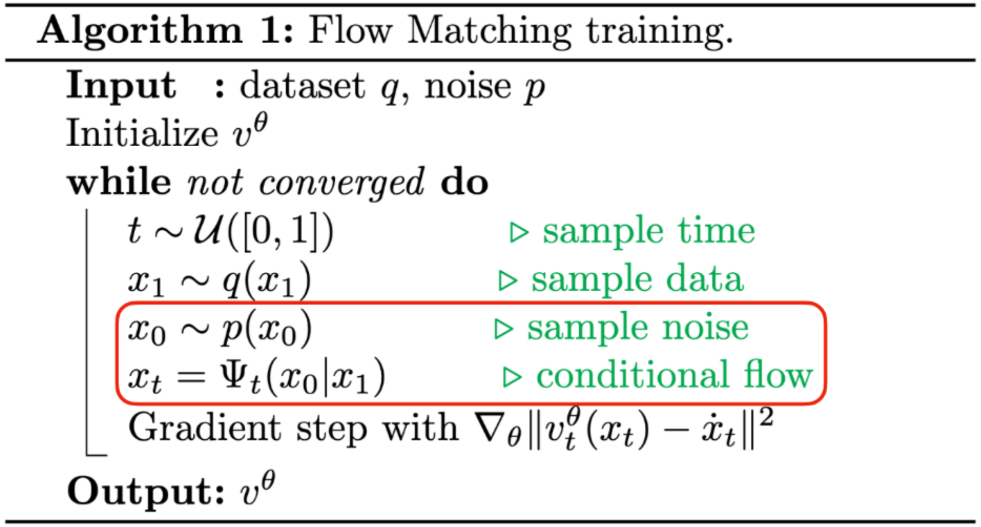
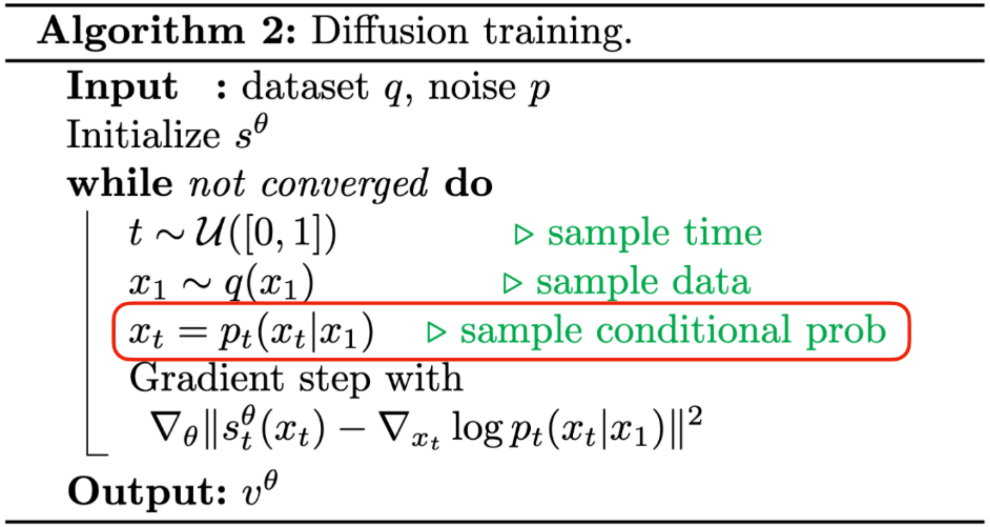


Image credit: Lecture slides by Yaron Lipman, https://drive.google.com/file/d/1Dkl_NEo1YpoDByxJLuNbqqA493-4NdCY/view , https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Flow Matching Model vs. Diffusion Model



```
Algorithm 1: Flow Matching training.
Input   : dataset $q$, noise $p$
Initialize $v^\theta$
while not converged do
    $t \sim \mathcal{U}([0,1])$                    ▷ sample time
    $x_1 \sim q(x_1)$                              ▷ sample data
    $x_0 \sim p(x_0)$                              ▷ sample noise
    $x_t = \Psi_t(x_0|x_1)$                        ▷ conditional flow
    Gradient step with $\nabla_\theta \|v_t^\theta(x_t) - \dot{x}_t\|^2$
Output: $v^\theta$
```



```
Algorithm 2: Diffusion training.
Input   : dataset $q$, noise $p$
Initialize $s^\theta$
while not converged do
    $t \sim \mathcal{U}([0,1])$                    ▷ sample time
    $x_1 \sim q(x_1)$                              ▷ sample data
    $x_t = p_t(x_t|x_1)$                           ▷ sample conditional prob
    Gradient step with
      $\nabla_\theta \|s_t^\theta(x_t) - \nabla_{x_t} \log p_t(x_t|x_1)\|^2$
Output: $v^\theta$
```

$p_t(x_t \,|\, x_1)$ general

$p(x_0)$ is general

$p_t(x_t \,|\, x_1)$ closed-form from of SDE $dx_t = f_t dt + g_t dw$

- *Variance Exploding*: $p_t(x \,|\, x_1) = \mathcal{N}(x \,|\, x_1, \sigma_{1-t}^2 I)$
- *Variance Preserving*: $p_t(x \,|\, x_1) = \mathcal{N}(x \,|\, \alpha_{1-t} x_1, (1 - \alpha_{1-t}^2) I)$

$$\alpha_t = e^{-\frac{1}{2}T(t)}$$

$p(x_0)$ is Gaussian

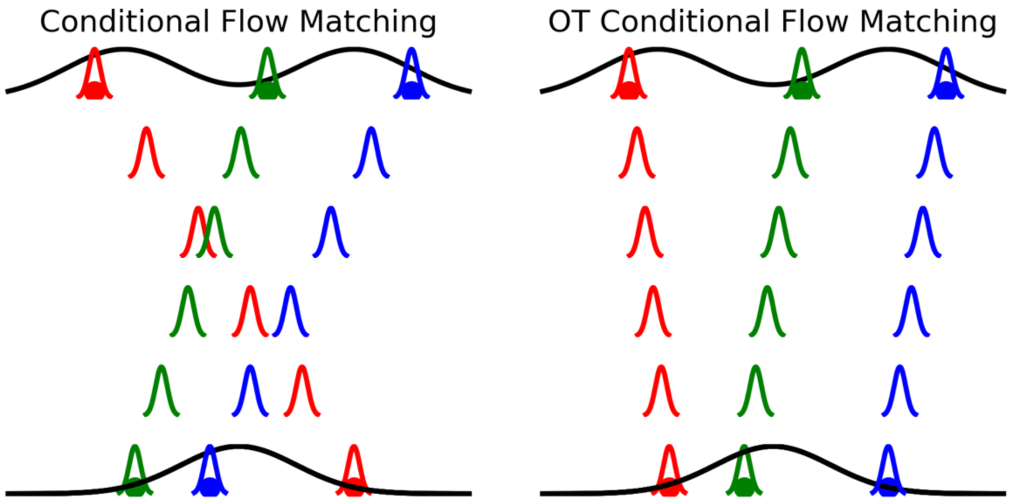
$p_0(\,\cdot\,|\, x_1) \approx p$

Image credit: Lecture slides by Yaron Lipman, https://drive.google.com/file/d/1Dkl_NEo1YpoDByxJLuNbqqA493-4NdCY/view

Outline

- Normalizing Flows and Continuous Normalizing Flows
    - The Continuity Equation
- The Fokker Plank Equation
- Flow matching
- **Variants:**
    - **Batch Optimal Transport Flow Matching**

Mini Batch OT Flow Matching Model



- Paths of various flow matching model design
  - Vanilla Conditional Flow Matching: Each conditional path is straight, but some paths intersect.
  - OT Conditional Flow Matching: Within the mini-batch, all paths are assigned as non-intersecting straight lines.

Image credit: Tong, Alexander, et al. "Improving and generalizing flow-based generative models with minibatch optimal transport." *Transactions on Machine Learning Research.*

Mini Batch OT Flow Matching Model

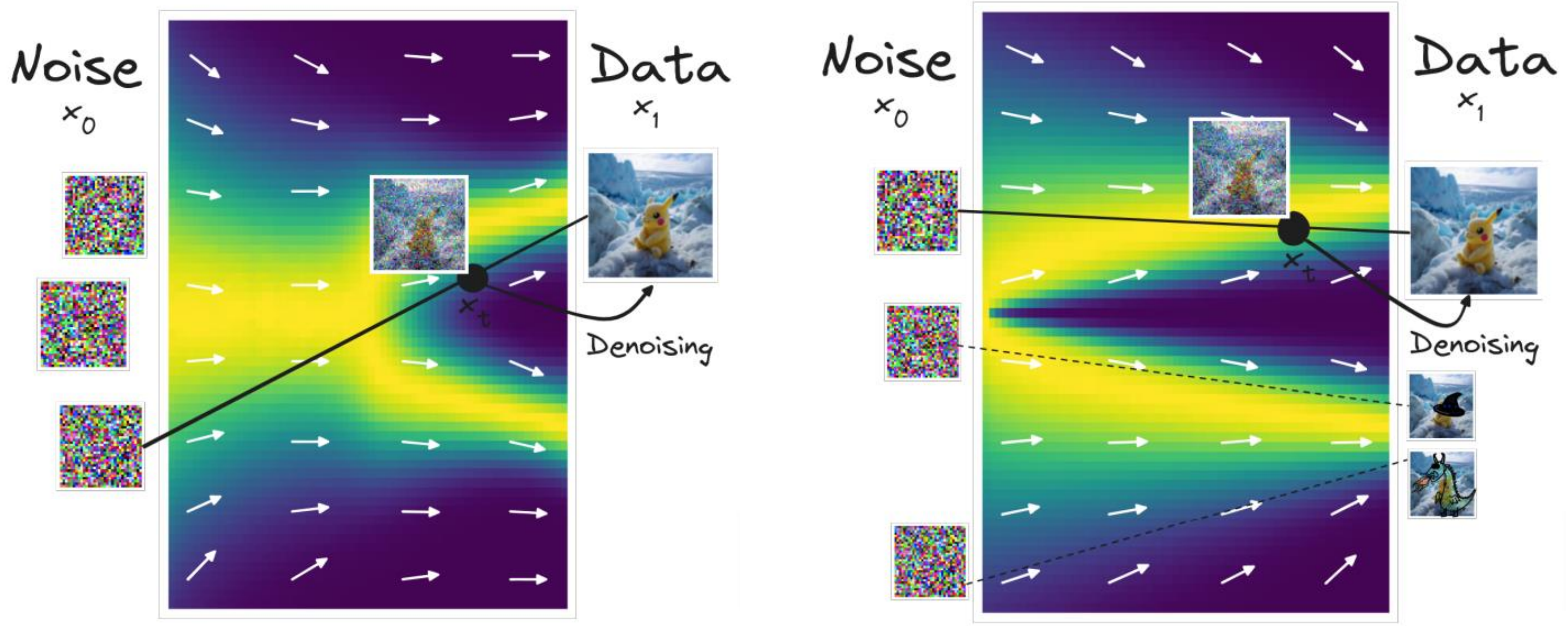


Image credit: https://indico.cern.ch/event/1425234/contributions/5994520/attachments/2872760/5030185/slides.pdf

Questions?