# Score and Flow-Matching

Generative and Deep Learning (GDL)

Davide Bacciu (davide.bacciu@unipi.it)


UNIVERSITÀ DI PISA

# Objectives

A unifying perspective over paradigms

- Score matching
- Continuous-time diffusion
- Flow matching
- Practical aspects
- Generative module wrap-up

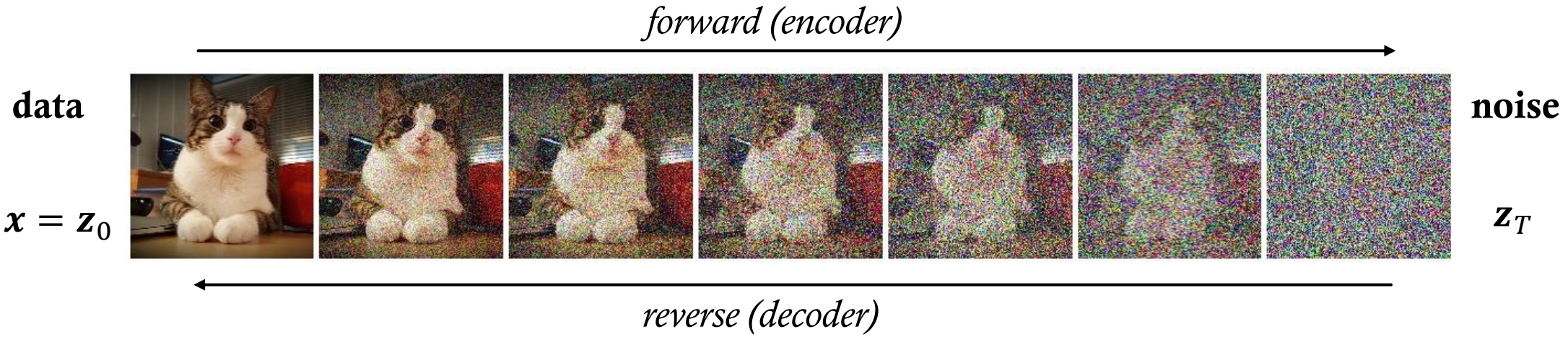
**Big picture:** diffusion ⇒ scores ⇒ probability flow ODE ⇒ flow matching

# Preliminary: a note on time

- Within the lecture we will use both $t$ and $\tau$ to denote time
  - $t \rightarrow$ time in the diffusion process $\mathbf{z}_t$ ($t = 0$ means **data**, $t = 1$ means **noise**)
  - $\tau \rightarrow$ time in flow matching ($\tau = 0$ means **source / prior**, $\tau = 1$ means **target / data)**
- By keeping the two separated we don't have to reverse the direction of time used in diffusion when we talk about flow matching

# From Diffusion to Scores

# Recall: Diffusion



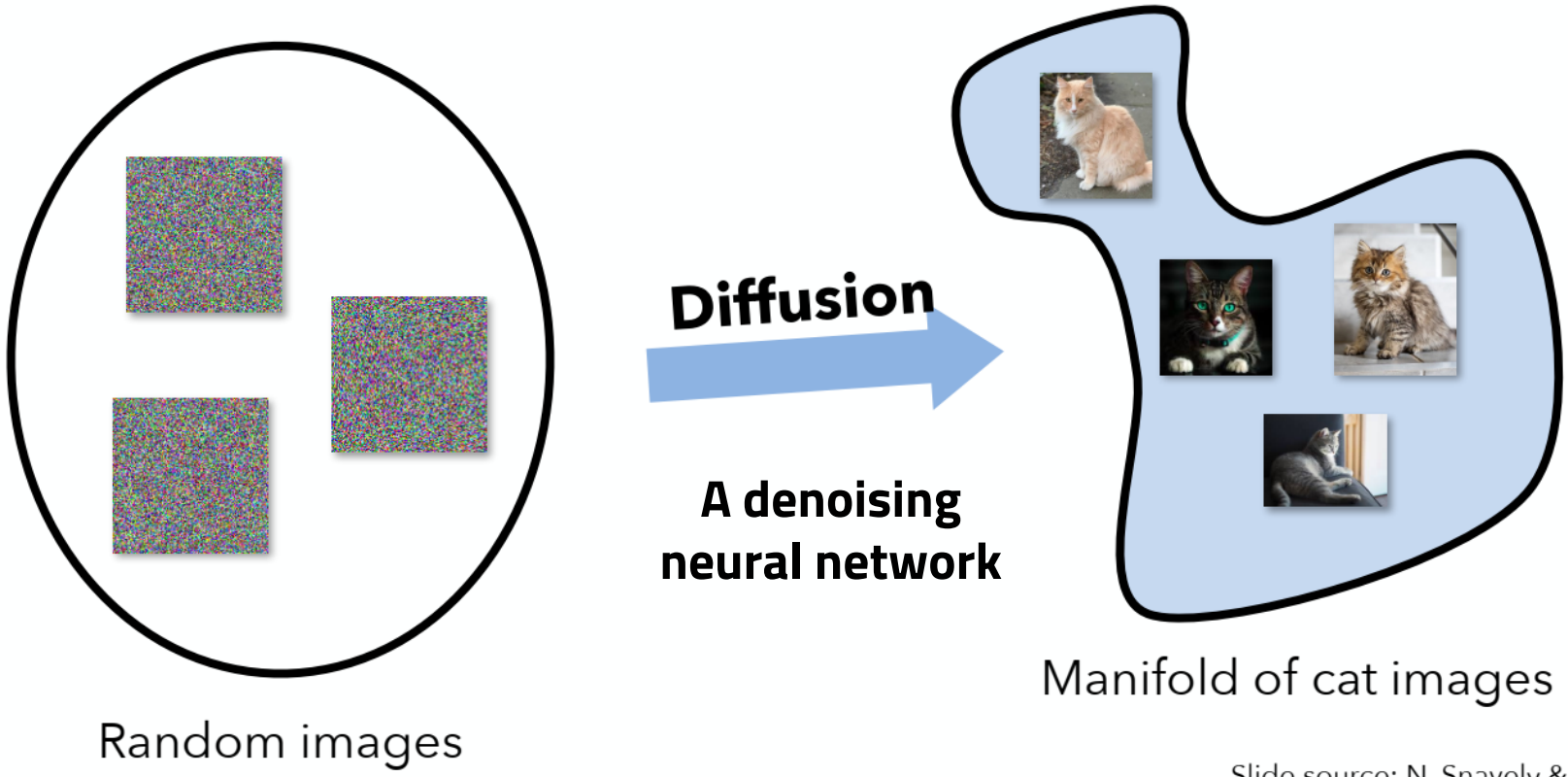

- Closed-form (kernelized) forward perturbation of a clean sample $\boldsymbol{z}_t = \sqrt{\alpha_t}\boldsymbol{z_0} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t$ with $\boldsymbol{\epsilon}_t \sim N(0, I)$ and noising distribution

$$q(\boldsymbol{z}_t|\boldsymbol{z}_0) = \mathcal{N}(\sqrt{\alpha_t}\boldsymbol{z_0}, (1-\alpha_t)\boldsymbol{I})$$

- Reverse denoising solved by a neural network $\boldsymbol{\epsilon}_\theta(\cdot)$ trained to minimize

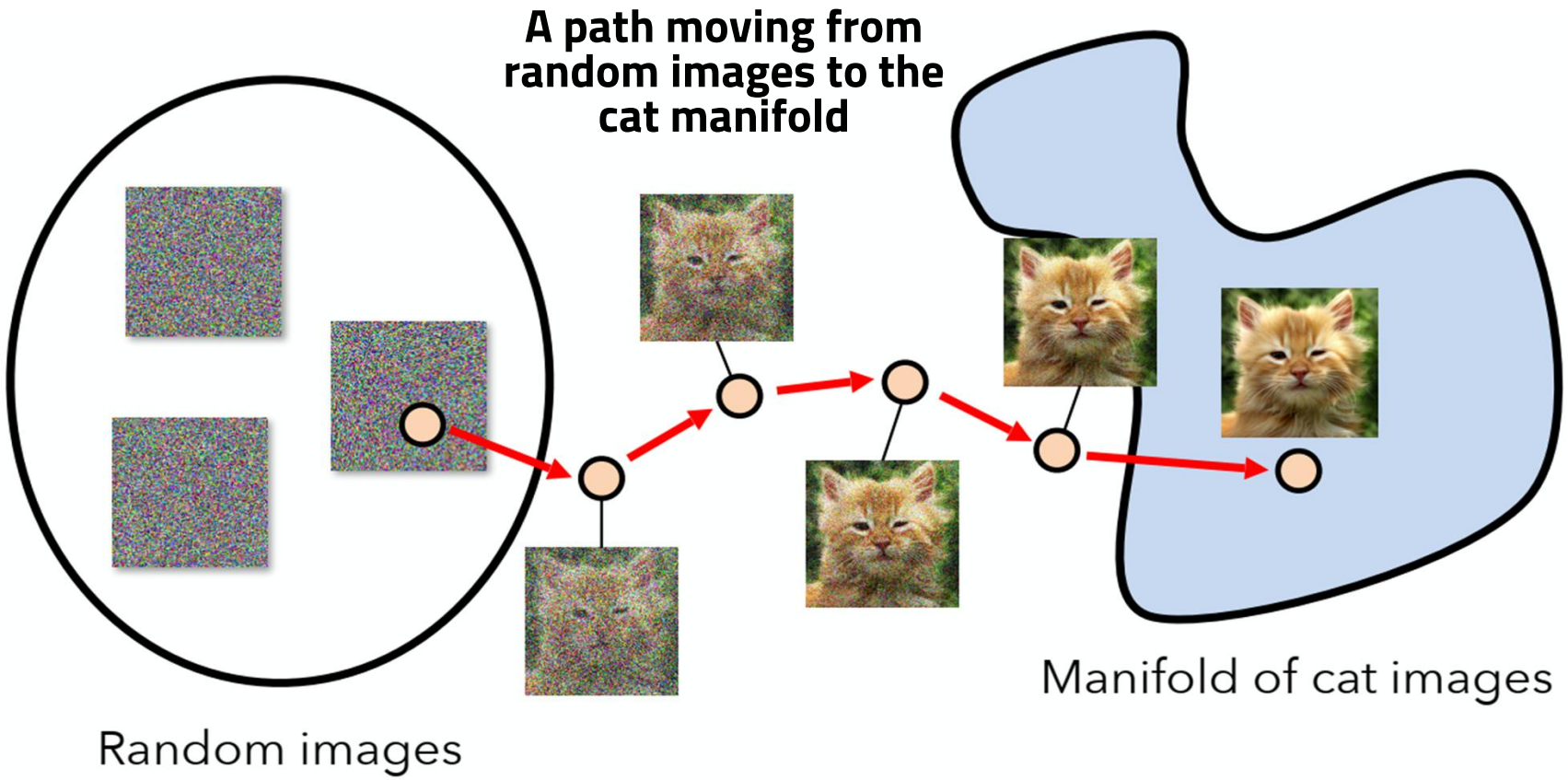
$$\mathbb{E}[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\boldsymbol{z}_t, t)\|^2]$$

# A slightly different view




Slide source: N. Snavely & S. Seitz

# A slightly different view




**Rings any bell?**

# Creating a bridge

- Our friend Tweedie's formula gives us a way to estimate the true posteriori mean $\boldsymbol{z_0}$ of some $\boldsymbol{z_t}$ distributed as $\mathcal{N}(\boldsymbol{z_0}, \boldsymbol{\Sigma})$ (which is exactly the case for $q(\boldsymbol{z_t}|\boldsymbol{z_0})$!)

$$\mathbb{E}\left[\sqrt{\alpha_t}\boldsymbol{z_0}\middle|\boldsymbol{z_t}\right] = \boldsymbol{z_t} + (1-\alpha_t)\,\nabla_{\boldsymbol{z_t}} \log p\,(\boldsymbol{z_t})$$

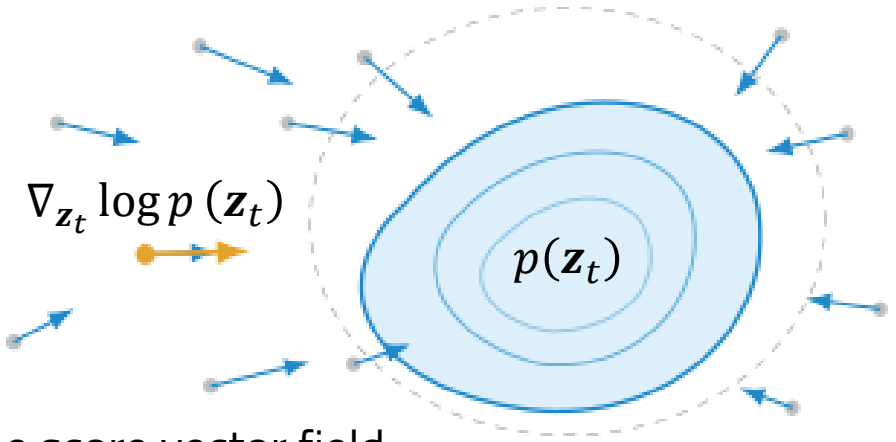
The denoised image — The noisy sample — The variance of our q() — ???

**The score function**

$$\boldsymbol{s}_t(\boldsymbol{z}) = \frac{\partial \log p(\boldsymbol{z_t})}{\partial \boldsymbol{z_t}} = \nabla_{\boldsymbol{z_t}} \log p\,(\boldsymbol{z_t})$$

- We can derive the best estimate for $\mathbb{E}\left[\sqrt{\alpha_t}\boldsymbol{z_0}\middle|\boldsymbol{z_t}\right] = \sqrt{\alpha_t}\boldsymbol{z}_0$ that plugged in Tweedie's formula yields

$$\boldsymbol{z}_0 = \frac{\boldsymbol{z_t} + (1-\alpha_t)\nabla_{\boldsymbol{z_t}} \log p(\boldsymbol{z_t})}{\sqrt{\alpha_t}}$$




The score vector field

# Putting things together

- From Tweedie's formula we obtain that

$$\boldsymbol{z}_0 = \frac{\boldsymbol{z_t} + (1-\alpha_t)\nabla_{\boldsymbol{z}_t} \log p\left(\boldsymbol{z}_t\right)}{\sqrt{\alpha_t}}$$

Notice any similarity?

- But from the kernelized denoising equation of diffusion models we know that

$$\boldsymbol{z}_0 = \frac{\boldsymbol{z_t} - \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t}{\sqrt{\alpha_t}}$$

Diffusion models do not merely subtract noise: they learn a time-dependent vector field (the score) over the data space

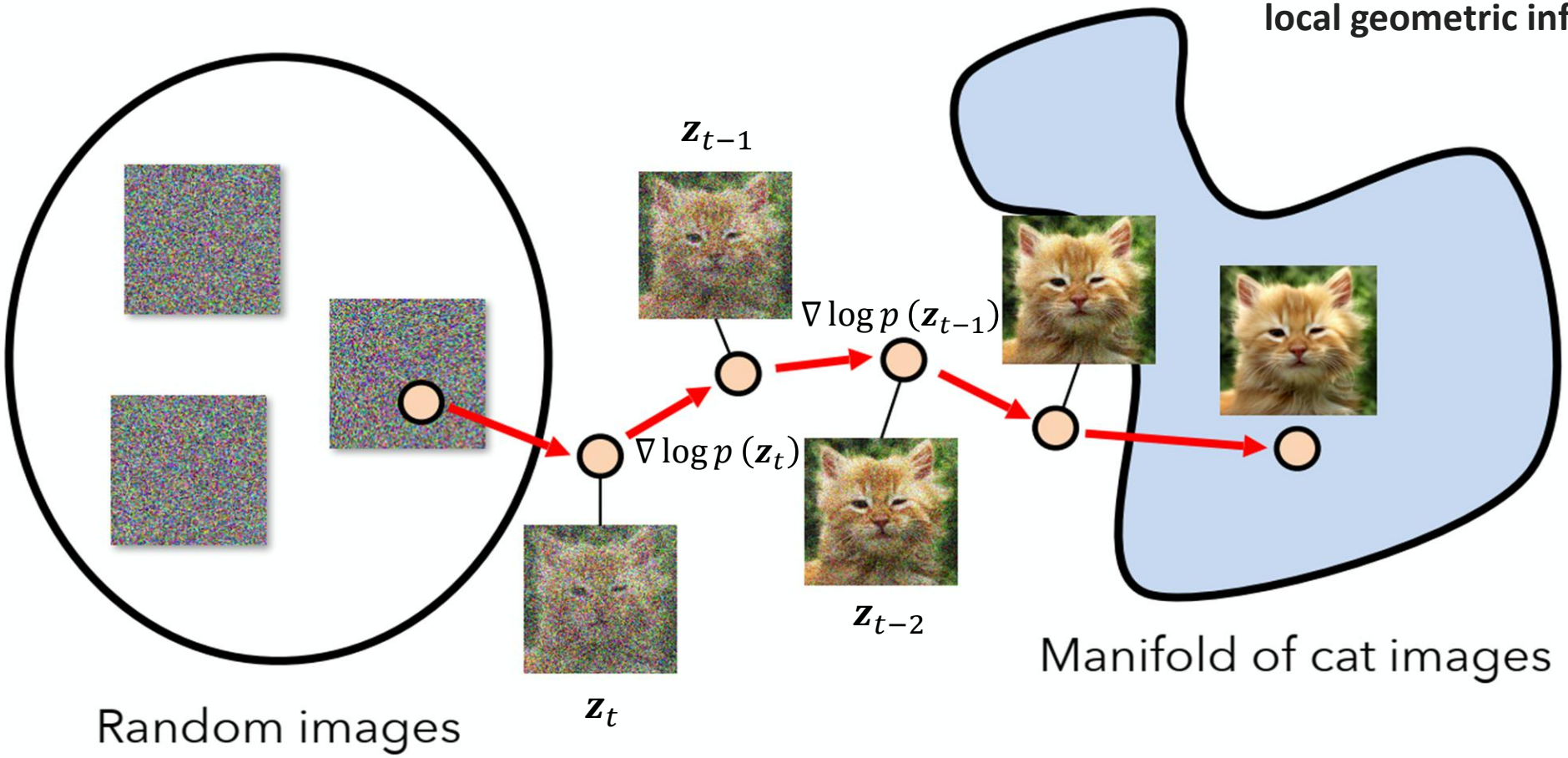
- By few algebraic manipulations yields

$$\nabla_{\boldsymbol{z}_t} \log p\left(\boldsymbol{z}_t\right) = -\frac{\boldsymbol{\epsilon_t}}{\sqrt{1-\alpha_t}}$$

- Moving in the opposite direction of the added noise goes in the direction of maximising the log likelihood

# Let us put it graphically

The reverse diffusion only needs **local geometric information**!



# Why the score is so nice?

- Given a parameterized probability distribution of a general exponential form

$$\mathrm{p}(\boldsymbol{z}) = \frac{1}{Z_\theta}\exp(-f_\theta(\boldsymbol{z}))$$

with $f_\theta$ parameterized energy function and $Z_\theta$ the (infamous) partition function

- Then taking derivative of the log of both sides

$$\nabla_{\boldsymbol{z}} \log p\,(\boldsymbol{z}) = \nabla_{\boldsymbol{z}} \log \frac{1}{Z_\theta}\exp(-f_\theta(\boldsymbol{z})) = \nabla_{\boldsymbol{z}} \log \frac{1}{Z_\theta} + \nabla_{\boldsymbol{z}} \log \exp(-f_\theta(\boldsymbol{z})) \;\; = -\nabla_{\boldsymbol{z}} f_\theta(\boldsymbol{z})$$

Any arbitrary scale factor applied to the probability density disappears

Which is a much more tractable object as it removes the partition function

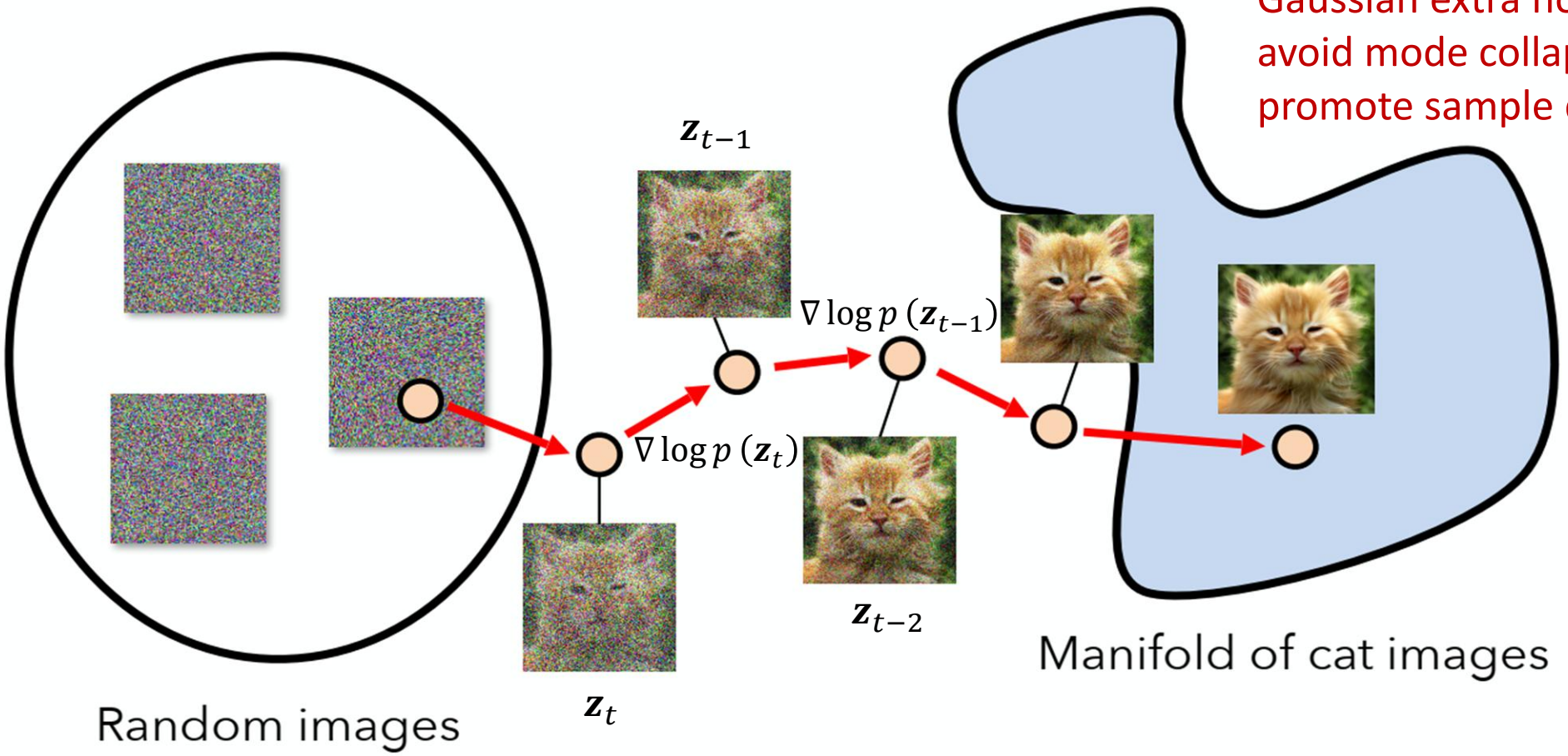
- We can train a neural network $s_\theta(\boldsymbol{z})$ to learn the score $\nabla_{\boldsymbol{z}} \log p\,(\boldsymbol{z})$ which is a standard regression task
- Given the score we can generate new data by starting at any point in space and iteratively following the score until we reach stability (a mode)

# Again Graphically

**Langevin dynamics sampling**

$$\boldsymbol{z}_{t-1} = \boldsymbol{z}_t + c_1 \nabla \log p\left(\boldsymbol{z}_t\right) + c_2 \boldsymbol{\epsilon}$$

Gaussian extra noise to avoid mode collapse and promote sample diversity



# Score matching

- In order to train our network $s_\theta(\boldsymbol{z})$ via the score matching loss

$$L_{SM} = \|s_\theta(\boldsymbol{z}) - \nabla_{\boldsymbol{z}} \log p(\boldsymbol{z})\|^2$$

we cannot assume (in the general case) to have access to the ground truth $\nabla_{\boldsymbol{z}} \log p(\boldsymbol{z})$ or $-\nabla_{\boldsymbol{z}} f_\theta(\boldsymbol{z})$

- We use some tricks to define losses which can be computed, the most popular being the Denoising Score Matching (DSM)

$$L_{DSM} = \|s_\theta(\tilde{\boldsymbol{z}}) - \nabla_{\boldsymbol{x}} \log p(\tilde{\boldsymbol{z}}|\boldsymbol{z})\|^2$$

where $\mathrm{p}(\tilde{\boldsymbol{z}}|\boldsymbol{z}) = \mathcal{N}(\boldsymbol{z}, \sigma^2)$ because we obtain $\tilde{\boldsymbol{z}}$ by adding some gaussian noise $\boldsymbol{\epsilon}$ with variance $\sigma^2$ to $\boldsymbol{z}$

- Luckily enough under these conditions $\nabla_{\boldsymbol{z}} \log p(\tilde{\boldsymbol{z}}|\boldsymbol{z}) = -\frac{1}{\sigma}\boldsymbol{\epsilon}$ and

$$L_{DSM} = \frac{1}{\sigma}\|s_\theta(\tilde{\boldsymbol{z}}) + \epsilon\|^2 = \frac{1}{\sigma}\|\epsilon - s_\theta(\tilde{\boldsymbol{z}})\|^2$$

Again: the denoising target (almost) of a diffusion model

# Score matching - Training

**Training for SM**

1. Pick a datapoint $\mathbf{x}$.
2. Sample $\epsilon$ from $\mathcal{N}(\epsilon|0, \mathbf{I})$.
3. Calculate the noisy version of data, $\tilde{\mathbf{x}} = \mathbf{x} + \sigma \cdot \epsilon$.
4. Calculate the score, $\tilde{s}_\theta(\tilde{\mathbf{x}})$.
5. Calculate gradient with respect to $\theta$ of the score matching objective, i.e., $\Delta\theta = \nabla_\theta \frac{1}{2\sigma} \|\epsilon - \tilde{s}_\theta(\tilde{\mathbf{x}})\|^2$. We accomplish that by applying automatic differentiation.
6. Update the score model: $\theta := \theta - \Delta\theta$. Go to 1 until STOP.

# Score matching - Generation

**Sampling with Langevin Dynamics for SM**

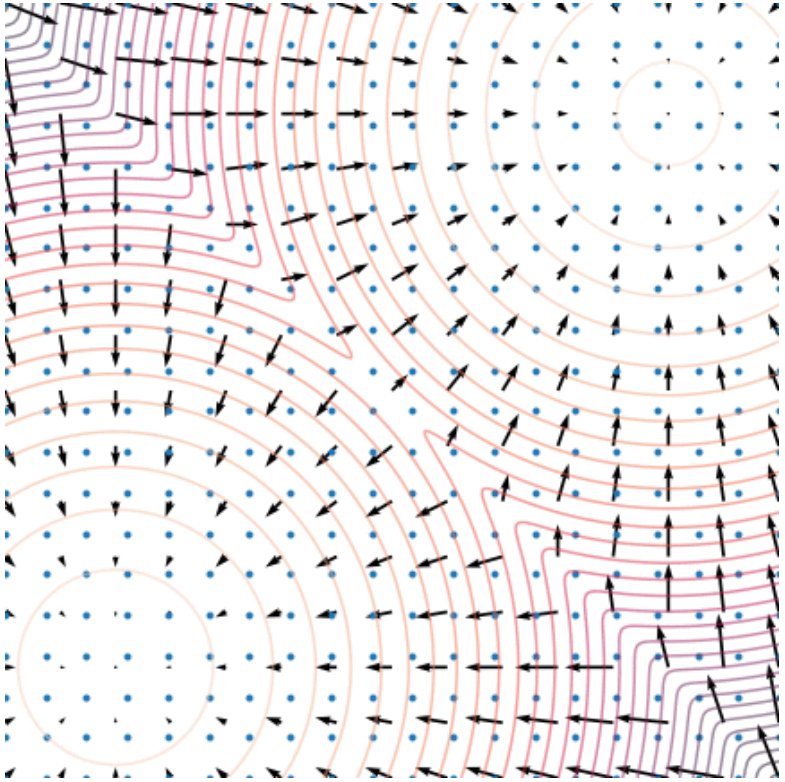
1. Sample a point $\mathbf{x}_0$ at random (e.g., using a uniform distribution).
2. Run Langevin dynamics for $T$ steps, where $\Delta = \frac{1}{T}$:

$$\mathbf{x}_{t+\Delta} = \mathbf{x}_t + \alpha\, \tilde{s}_\theta(\mathbf{x}_t) + \eta \cdot \epsilon, \tag{9.14}$$

3. Return the final point $\mathbf{x}_1$.

# Wrapping up score matching

- Diffusion models can be related to specific score matching objectives
  - Depending on the choice of the noising distribution that we assume
- Score matching gives a vector field interpretation over diffusion models
  - Composition is easy in vector fields (summation)
  - Generation becomes a matter of taking paths in data space following the vector field
- However, noise processes need to have a specific form to make the math of denoising/score work



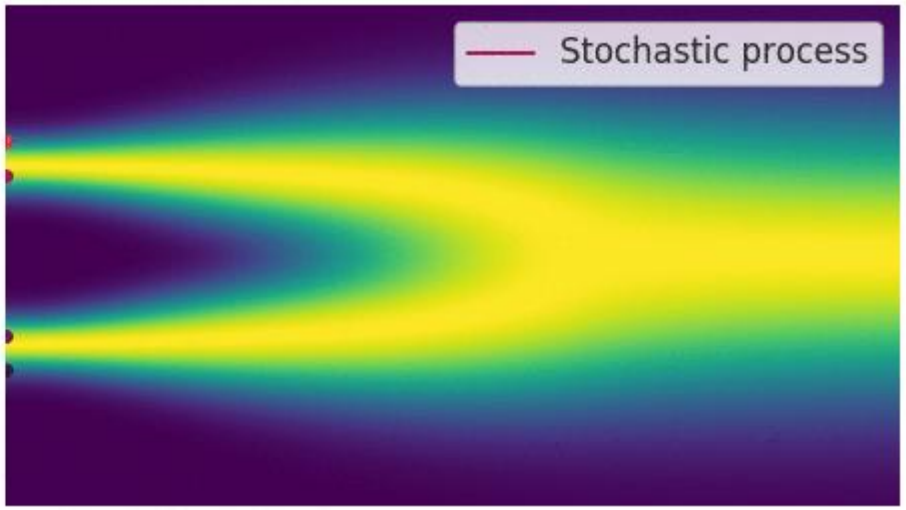
Score vector field of 2 Gaussians and their sampling via Langevin

# From Scores to Flows

# Going with the flow

- One critical aspect in score is that the choice of a single noise scaling $\sigma$ is not at all trivial
- The solution that was found was to use multiple noise processes at the same time with different $\sigma_1, \dots, \sigma_L$ and then…

Make $L \to \infty$ to perturb the data distribution with continuously growing levels of noise




The noise perturbation procedure is a continuous-time stochastic process

# Noise Diffusion as a Stochastic Process

- Diffusion processes (and hence score matching) are solutions of stochastic differential equations (SDEs)

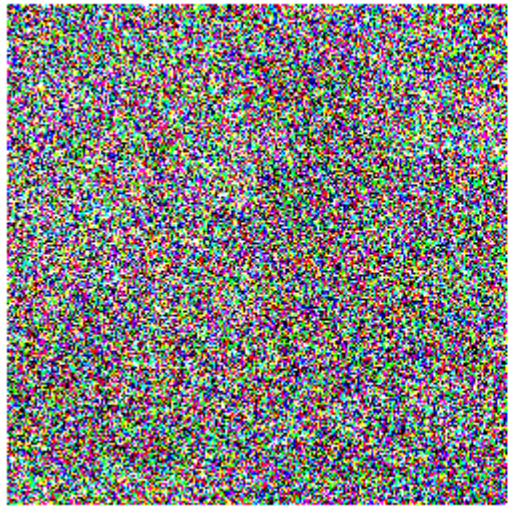
$$d\boldsymbol{z} = \boldsymbol{f}(\boldsymbol{z}, t)dt + g(t)d\boldsymbol{w}$$

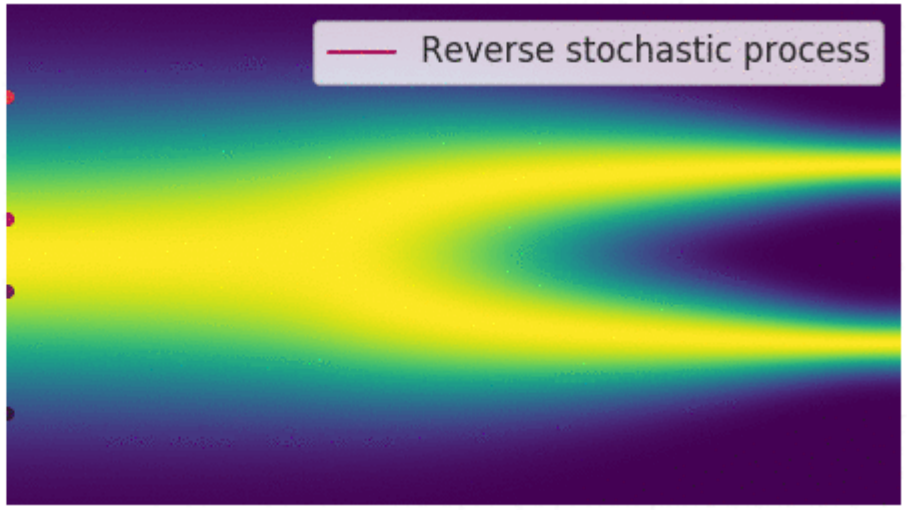
- $\boldsymbol{z}$: Instantaneously perturbed sample
- $\boldsymbol{f}(\boldsymbol{z}, t)$: Drift coefficient
- $g(t)$: Diffusion coefficient
- $d\boldsymbol{w}$: Infinitesimal white noise (Brownian motion)

- The evolution of $\boldsymbol{z}(t)$ for $t \in [0,1]$ according to the SDE produces stochastic trajectories
- At any point in time $t$ we have the marginal distribution $\mathrm{p}(\boldsymbol{z}(t)) = \mathrm{p}_t(\mathbf{z})$ for the process (the analogous of $p_{\sigma_i}(\boldsymbol{z})$ for infinite noise scales)

# Reverse SDE

Once we have defined the diffusion (noising) direction, we can derive the equivalent reverse direction (denoising)






The trajectories can be obtained in closed form

$$d\boldsymbol{z} = [\boldsymbol{f}(\boldsymbol{z}, t) - g^2(t)\nabla \log \mathrm{p}_t(\mathbf{z})]dt + g(t)d\boldsymbol{w}$$

Time-dependent score learned by our neural network

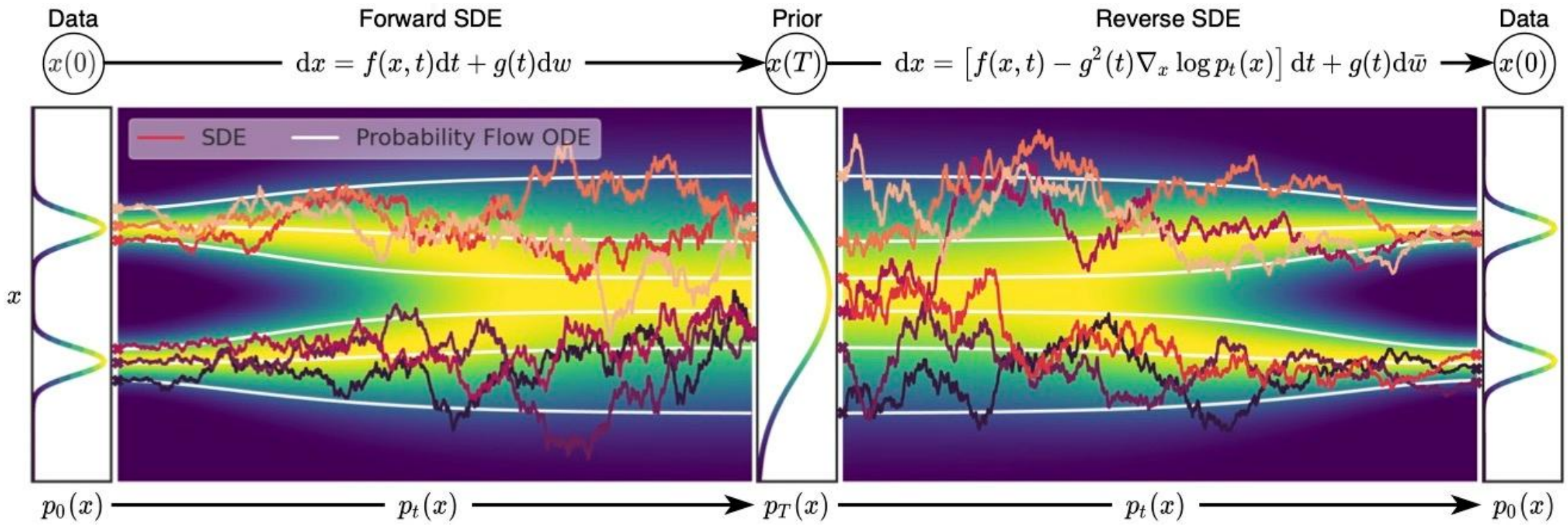
# From SDEs to ODEs

- The reverse SDE describes a stochastic process mapping samples from a simple base distribution $\mathrm{p}_{t=1}(\mathbf{z})$ to samples from the data distribution $\mathrm{p}_{t=0}(\mathbf{z})$
- But it does so through **a very noisy/stochastic process due to the Brownian motion term** $g(t)d\boldsymbol{w}$
- Luckily if we remove the Brownian term, we obtain an Ordinary Differential Equation (ODE) (the Probability flow ODE) that preserves the same marginal distributions as the SDE (after appropriate adjustments)!

$$d\boldsymbol{z} = \left[\boldsymbol{f}(\boldsymbol{z}, t) - \frac{1}{2} g^2(t) \nabla \log \mathrm{p}_t(\mathbf{z})\right] dt$$

The ODE does not reproduce the same individual paths as the SDE: it reproduces the same sequence of distributions $\mathrm{p}_t(\mathbf{z})$

# From SDEs to ODEs




A bijective mapping between samples from the simple base distribution, and samples from the data distribution

# Remarks

- We have a sampling process where all the randomness is contained in the initial base distribution sample
  - Going from the random sample to a data sample (and back) is completely deterministic
- We can
  - Map data points to their corresponding latent representations by simulating the ODE forward
  - Manipulate them
  - Map them back to the data space by simulating the ODE backward

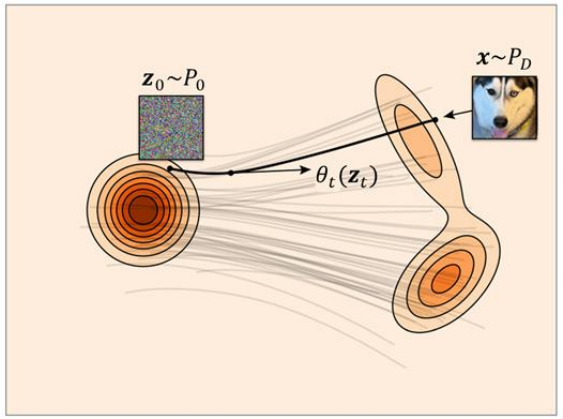
# A final twist

- The Probability flow ODE defines a velocity field

$$\frac{d\mathbf{z}_t}{dt} = [\boldsymbol{f}(\mathbf{z}, t) - g^2(t)\nabla \log \mathrm{p}_t(\mathbf{z})]$$

- What if we make the model directly estimate the transport velocity?

$$\frac{\mathrm{d}\mathbf{z}_t}{dt} = v_\theta(\mathbf{z}_t, t)$$




**A continuous normalizing flow!**

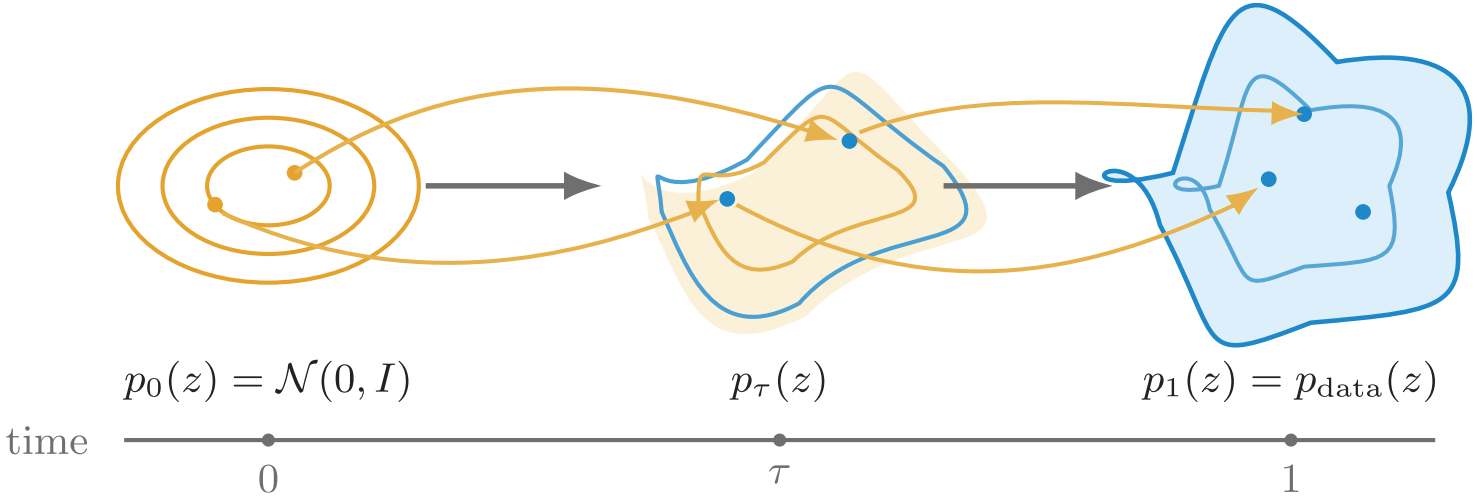
$$\begin{cases} \mathbf{z}_0 \sim N(0,1) \\ \dfrac{\partial \mathbf{z}_t}{\partial t} = \theta_t(\mathbf{z}_t) \end{cases}$$

**Flow Matching**

We learn the transport velocity *directly*, without going through likelihood training or reverse-time SDE theory (no need to solve the ODE during training)

# Flow Matching

- Let us use now our inverted time $\tau = 1 - t$
- Choose a **probability path** $(p_\tau)_{\tau \in [0,1]}$ connecting a base distribution $p_0$ to the data distribution $p_1$
- Regress a velocity field $v_\theta(\boldsymbol{z}_\tau, \tau)$ on the ground truth one $u_\tau(\boldsymbol{z}_\tau)$ that generates this path



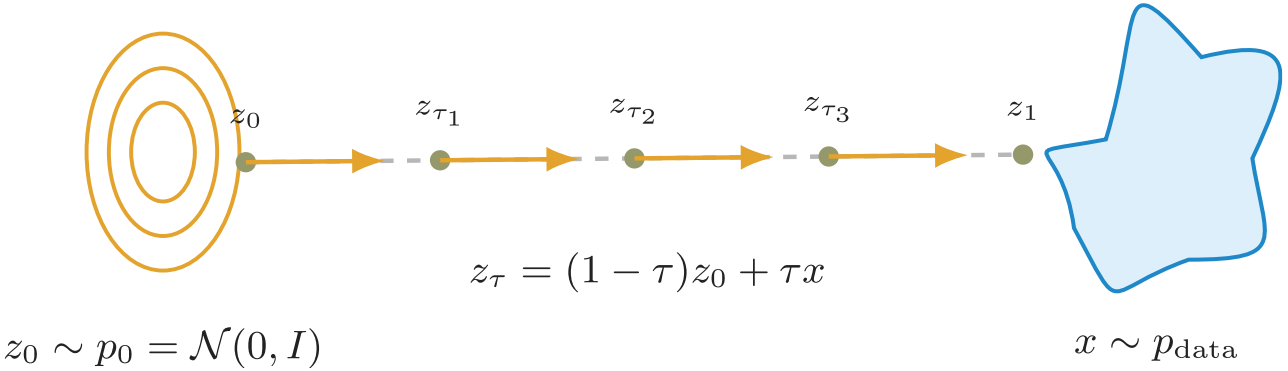

Diffusion: the path is usually dictated by a carefully designed noise process
Flow matching: choose the path (more) freely

**Different methods depending on how we choose the path**

# Simplest Path: Rectified Flow

- Given a source sample $\boldsymbol{z_0} \sim p_0$
- Given a data sample $\boldsymbol{x} \sim p_{data}$ $(p_1)$
- Define

$$\boldsymbol{z_\tau} = (1-\tau)\boldsymbol{z_0} + \tau \boldsymbol{x}$$




- A linear interpolation: straight trajectories
- Constant velocity along each segment
- Differentiating yields a very simple supervision target for $v_\theta(\boldsymbol{z_\tau}, \tau)$, that is

$$u_\tau(\boldsymbol{z_\tau}|\boldsymbol{x}, \boldsymbol{z_0}) = \frac{\mathrm{d}\mathbf{z}_\tau}{d\tau} = \boldsymbol{x} - \boldsymbol{z_0}$$

**Conditional velocity field**

# Gaussian Probability Paths

- A more general family of (simple) flows is

$$\boldsymbol{z}_\tau = \alpha_\tau \boldsymbol{x} + \sigma_\tau \boldsymbol{\epsilon} \text{ with } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

- With boundary conditions

$$\underbrace{\alpha_0 = 0,\ \sigma_0 = 1}_{\boldsymbol{z}_0 = \epsilon} \text{ and } \underbrace{\alpha_1 = 1,\ \sigma_1 = 0}_{\boldsymbol{z}_1 = x}$$

$\alpha_\tau$: How much data we have mixed in at time $\tau$

$\sigma_\tau$: How much noise remains at time $\tau$

# Gaussian Probability Paths

- A more general family of (simple) flows is

$$\boldsymbol{z}_\tau = \alpha_\tau \boldsymbol{x} + \sigma_\tau \boldsymbol{\epsilon} \text{ with } \boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \boldsymbol{I})$$

- With boundary conditions

$$\alpha_0 = 0,\ \sigma_0 = 1 \text{ and } \alpha_1 = 1,\ \sigma_1 = 0$$

- Differentiating to get the velocities yields

$$u_\tau(\boldsymbol{z}_\tau | \boldsymbol{x}, \boldsymbol{\epsilon}) = \frac{\mathrm{d}\mathbf{z}_\tau}{d\tau} = \dot{\alpha}_\tau \boldsymbol{x} + \dot{\sigma}_\tau \boldsymbol{\epsilon}$$

**This is diffusion style noising!!**

which is a supervision target that can be computed knowing the training pair $(\boldsymbol{x}, \boldsymbol{\epsilon})$ and the schedules of $\alpha_\tau$ and $\sigma_\tau$

Gaussian probability paths subsume diffusion paths as special cases

⇒ Gaussian flow matching and diffusion can describe the same underlying generative model under different parameterizations and loss weightings

# Conditional Flow Matching

- Ideally, we would regress the marginal velocity $u_\tau(\boldsymbol{z}_\tau)$, but this is not directly observable
- Instead, choose an easy conditional path and regress its conditional velocity

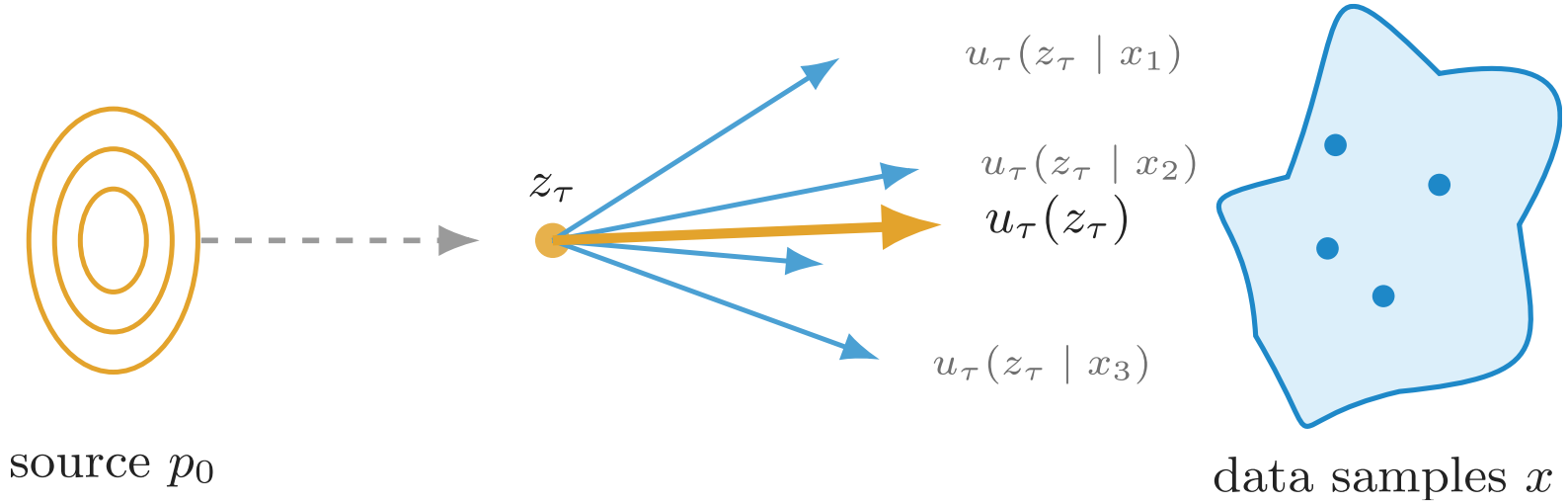
$$\mathcal{L}_{CFM}(\theta) = \mathbb{E}_{\tau,\boldsymbol{x},\boldsymbol{z} \sim q_\tau(\cdot \mid x)} \left[\| v_\theta(\boldsymbol{z}, \tau) - u_\tau(\boldsymbol{z} \mid \boldsymbol{x}, \boldsymbol{\epsilon})\|^2\right]$$

Where $q_\tau(\cdot \mid \boldsymbol{x})$ is the chosen conditional path distribution (i.e. the Gaussian one)

Concretely

- Sample $\boldsymbol{x}$ from the dataset
- Sample a source variable ($\boldsymbol{z}_0$ or $\boldsymbol{\epsilon}$)
- Construct $\boldsymbol{z}_\tau$ at a random time $\tau$
- Regress $v_\theta(\boldsymbol{z}_\tau, \tau)$ on the known conditional velocity target $u_\tau(\boldsymbol{z}_\tau \mid \boldsymbol{x}, \boldsymbol{\epsilon})$ (or $u_\tau(\boldsymbol{z}_\tau \mid \boldsymbol{x}, \boldsymbol{z_0})$)

# Why Conditional Flow Matching Works




- The (ideal) target marginal velocity is the conditional expectation of the easy conditional targets

$$u_\tau(\boldsymbol{z}) = \mathbb{E}\left[u_\tau(\boldsymbol{z}_\tau \mid \boldsymbol{x}, \boldsymbol{\epsilon}) \mid \boldsymbol{z}_\tau = \boldsymbol{z}\right]$$

- The conditional targets average to the correct velocity that transports the marginal path
- The $\mathcal{L}_{CFM}(\theta)$ loss gives the correct gradient for the marginal flow objective

# Training flow matching Vs diffusion

Target: move along the transport direction

**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
- $\boldsymbol{x} \sim p_1(\boldsymbol{x})$ ▷ sample data
- $\boldsymbol{z}_0 \sim p_0(\boldsymbol{z}_0)$ ▷ sample noise
- $\dot{\boldsymbol{z}}_\tau = u_\tau(\boldsymbol{z}_\tau | \boldsymbol{x}, \boldsymbol{z}_0)$ ▷ conditional flow
- Gradient step with $\nabla_\theta \| v_\theta(\boldsymbol{z}_\tau, \tau) - \dot{\boldsymbol{z}}_\tau \|^2$

**Output:** $v^\theta$

predict velocity

$\boldsymbol{z}_\tau = (1-\tau)\boldsymbol{z}_0 + \tau \boldsymbol{x}$

construct interpolated point

Target: remove the perturbing noise

**Algorithm 2:** Diffusion training.

**Input** : dataset $q$, noise $p$

Initialize $s^\theta$

**while** *not converged* **do**

- $t \sim \mathcal{U}([0,1])$ ▷ sample time
- $\boldsymbol{x} \sim p_1(\boldsymbol{x})$ ▷ sample data
- $\boldsymbol{z}_t \sim p(\boldsymbol{z}_t | \boldsymbol{x})$ ▷ sample conditional prob
- Gradient step with $\nabla_\theta \| s_t^\theta(x_t) - \nabla_{x_t} \log\ p(\boldsymbol{z}_t | \boldsymbol{x}) \|^2$

**Output:** $s^\theta$

predict score/noise

$\boldsymbol{z}_t = \sqrt{\alpha_t}\boldsymbol{z_0} + \sqrt{1-\alpha_t}\boldsymbol{\epsilon}_t$

construct noisy point

# Sampling - Integrate the learned ODE

- After training, generate by solving the Cauchy problem

$$\frac{\mathrm{d}\mathbf{z}_\tau}{d\tau} = v_\theta(\boldsymbol{z}_\tau, \tau) \text{ with } \boldsymbol{z}_0 \sim p_0$$

- Can be solved by any available numerical ODE solver
  - Euler, Midpoint, Runge-Kutta, …
- A first-order Euler is typically quite friendly

$$z_{\tau+\Delta\tau} = z_\tau + \Delta\tau\, v_\theta\,(\boldsymbol{z}_\tau\,, \tau)$$

- Sampling is entirely deterministic
- Number of steps may be much smaller than in classical diffusion, if the chosen path yields a smooth velocity field (e.g. a Gaussian path)

# Wrap-up

# Flow Matching Networks - Practically

**NN backbone as in diffusion models**

- U-Net for image-space generation
- Transformer variants in latent or patch space
- Sinusoidal or learned time embeddings
- Cross-attention / conditioning exactly as in conditional diffusion

**What changes is the head and target**

- predict noise $\epsilon$ (diffusion)
- predict score s (score matching)
- predict velocity v (flow matching)

*In modern practice, the architectural gap between diffusion and flow matching is much smaller than the conceptual gap in the textbooks*

# Take home messages

- Diffusion models already learn scores: the usual noise-prediction objective is a reparameterized denoising score-matching loss
- Score-based models make diffusion continuous in time through forward and reverse SDEs
- Probability flow ODEs show that the same marginals can be generated deterministically, connecting diffusion to CNFs.
- Flow matching skips likelihood training and directly regresses the transport velocity along a chosen probability path
- The practical recipe remains familiar: sample time, construct a perturbed/interpolated point, regress a target vector field, integrate at test time

**Scores explain diffusion. Flows explain transport. Flow matching makes the transport directly learnable**

# END OF MODULE SUMMARY

# Generative DL Summary (I)

## Generative adversarial networks

- Adversarial learning as a general and effective principle
- Effective and efficient in generating high quality samples
- Do not learn sample likelihood
- GANs generally more unstable than other deep generative models

## Variational Autoencoders

- ELBO trained and imposing standard normal structure
- Encoder-Decoder scheme with latent variables of any dimension
- Can be integrated with adversarial, diffusion and flow approaches
- Useful to study representation learning aspects but bad at sampling

## Diffusion models

- Generate data from noise through a learned incremental denoising with fixed steps
- A hierarchical VAE with fixed encoder and no explicit density
- Easy to train, scalable to parallel hardware and generate high quality samples (though can be slow)

# Generative DL Summary (II)

## Normalizing flows

- Can learn arbitrary distributions for high-dimensional data in a tractable way using change of variable
- Can handle efficient sampling and interpolation
- Generalize and make tractable autoregressive modelling
- Require bijective transformations and “well-behaved” Jacobians

## Score matching models

- Very principled stochastic framework: allowed to connect the dots between generative models
- Score is additive and naturally composes probabilities
- Strong machinery for conditioning and solving inverse flow
- Stochasticity can exert its toll (in training/SDE)

## Flow matching models

- Simple regression objective
- Direct deterministic sampling through an ODE (often with few integration steps)
- Flexible path design
- Creates connections with optimal transport

# Coming-up next

- Deep learning for graphs (2 lectures)
  - Processing graph structured data in neural network
  - Learning tasks on graphs
  - Foundational neural models for graphs
  - Information propagation on graphs
  - Generative learning and graphs