Diffusion models, Flow Matching and their applications for Image Inverse Problems

Samuel Hurault

June,19 2026

Generative models

From a dataset, generate novel data

Learned representation

Generation

Generative models

From a dataset, generate novel data

Learned representation

« A blond child with curly hair»

Condition

Generation

Generative models

From a dataset, generate novel data

Learned representation

Measurement

Generation

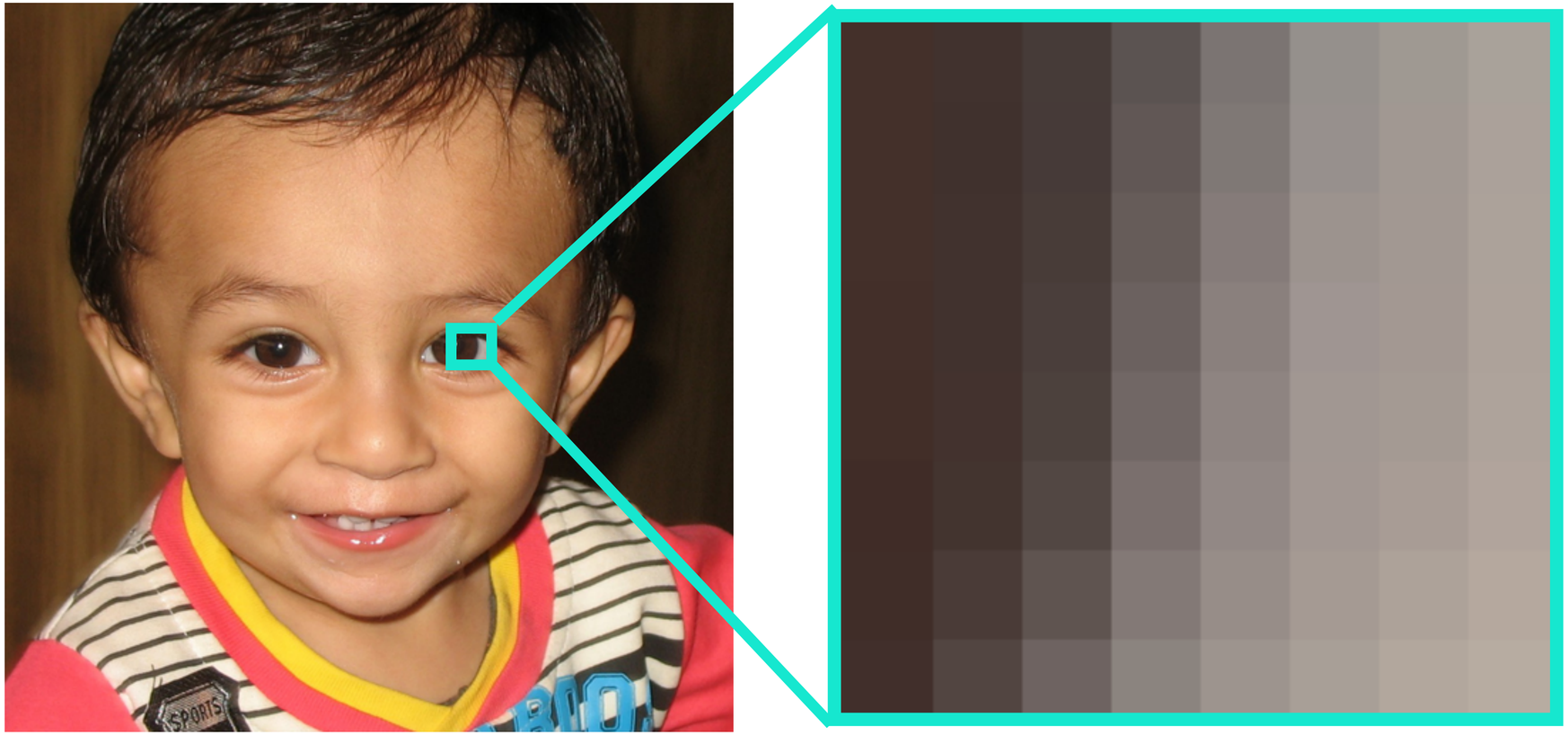
Generative models

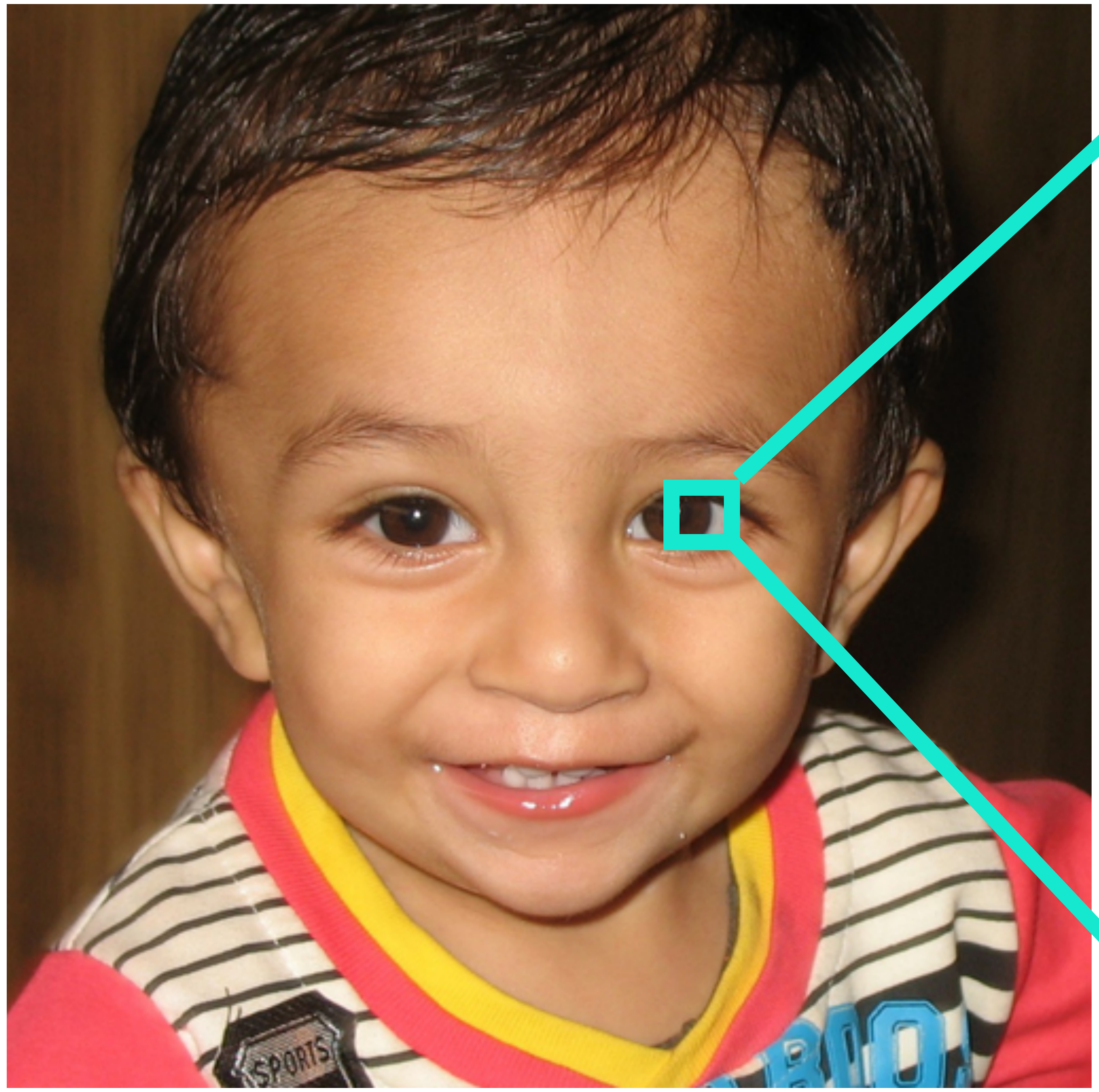
From a dataset, generate novel data

Learned representation

Measurement

Generation

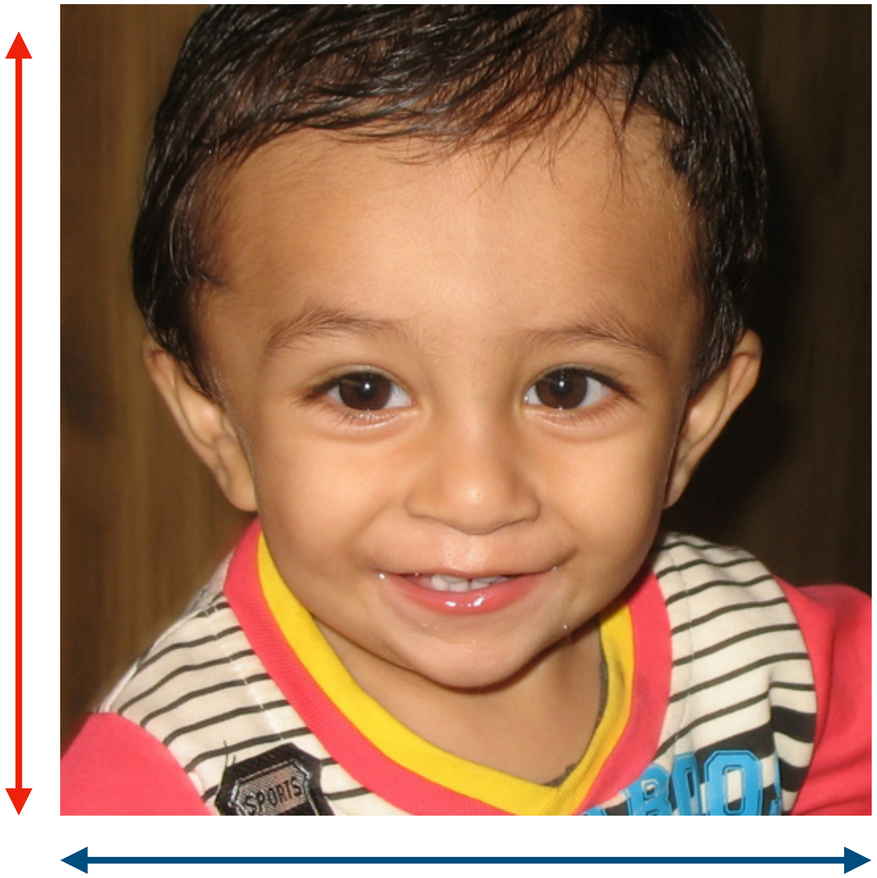




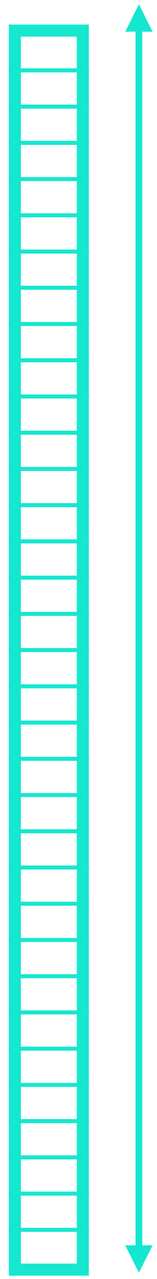
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34
0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34	0.85 0.12 0.34

An image is a vector in $\mathbb{R}^d$ with

$d =$ nb pixel width x nb pixel height x nb channels



Stacking



$\in \mathbb{R}^d$

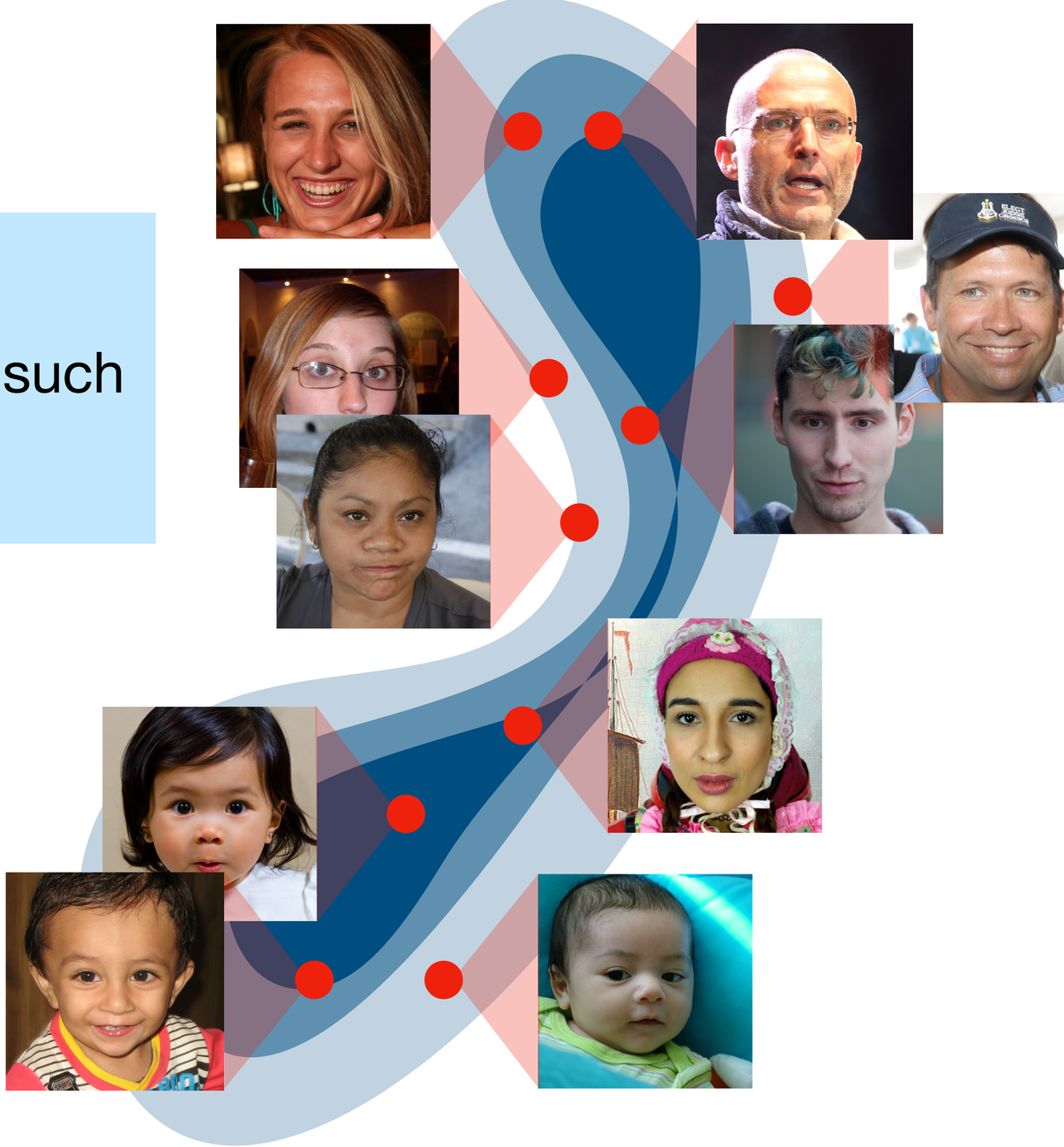
Each image i is a point in $x_i \in \mathbb{R}^d$

Assumption:

There exists a probability distribution p_{data} such that each image $x_i \sim p_{\text{data}}$

Objective:

Approximate p_{data} with a parametric distribution p_θ « easy to sample »



Trade-off:

approximation quality $p_\theta \approx p_{\text{data}}$ ⟷ sampling efficiency

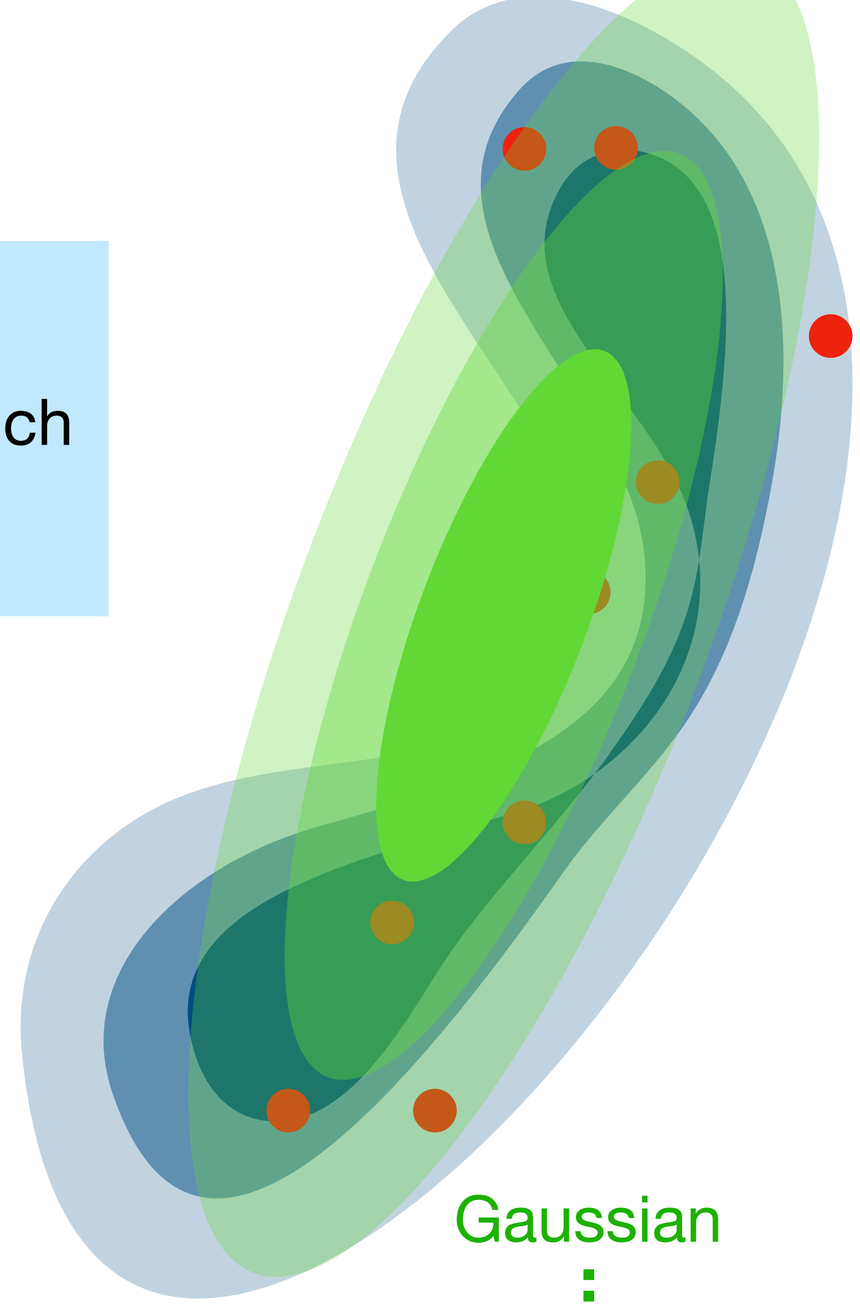
Each image i is a point in $x_i \in \mathbb{R}^d$

Assumption:

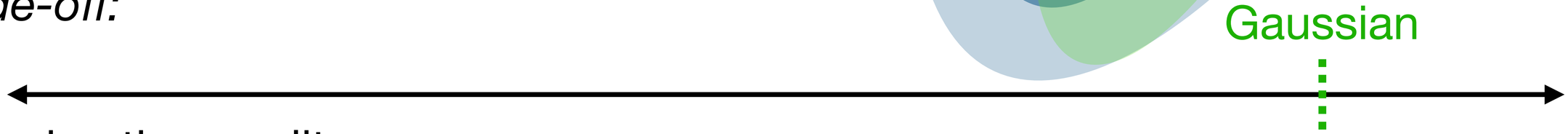
There exists a probability distribution p_{data} such that each image $x_i \sim p_{\text{data}}$

Objective:

Approximate p_{data} with a parametric distribution p_θ « easy to sample »



Trade-off:



approximation quality $p_\theta \approx p_{\text{data}}$

sampling efficiency

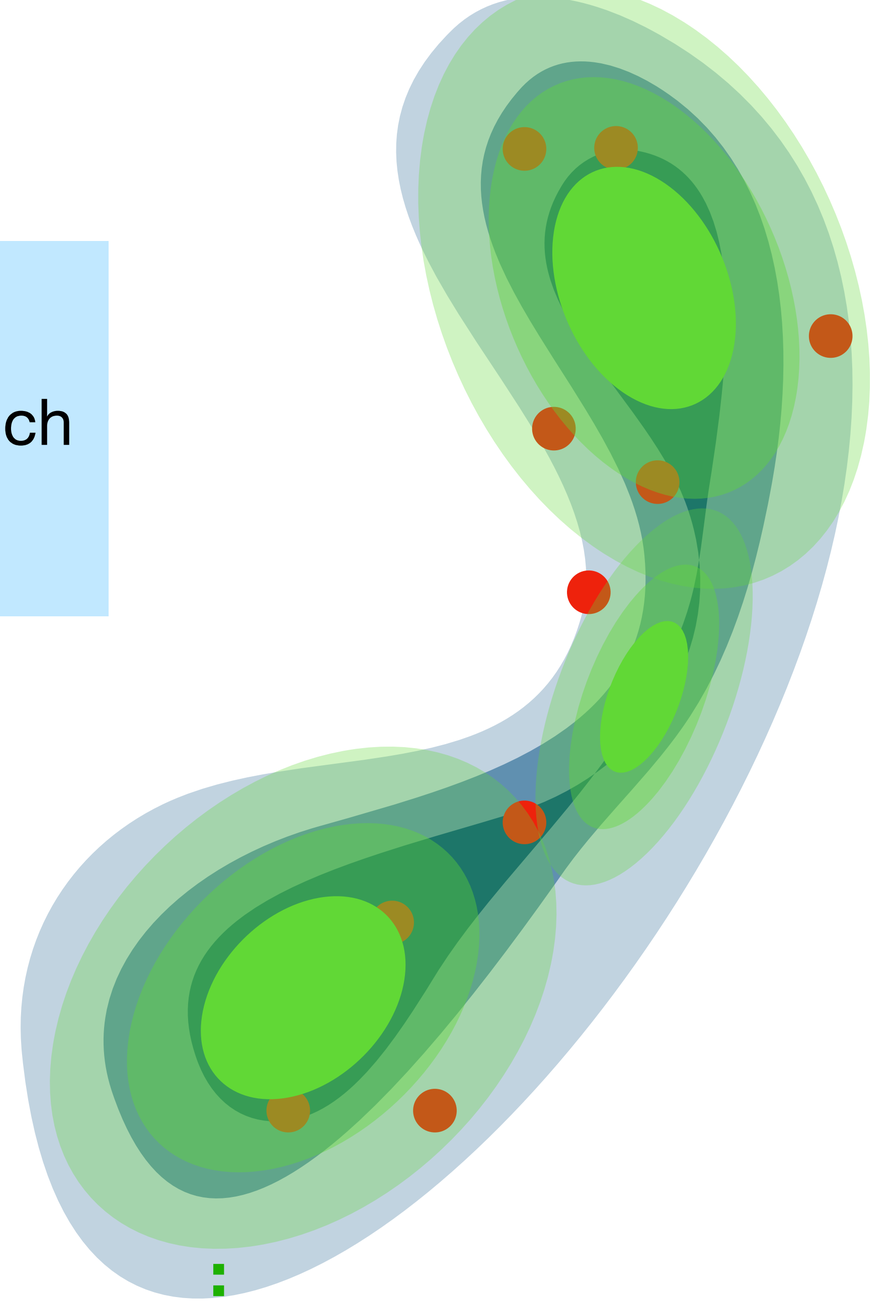
Each image i is a point in $x_i \in \mathbb{R}^d$

Assumption:

There exists a probability distribution p_{data} such that each image $x_i \sim p_{\text{data}}$

Objective:

Approximate p_{data} with a parametric distribution p_θ « easy to sample »



Trade-off:

approximation quality $p_\theta \approx p_{\text{data}}$ — Gaussian Mixture — sampling efficiency

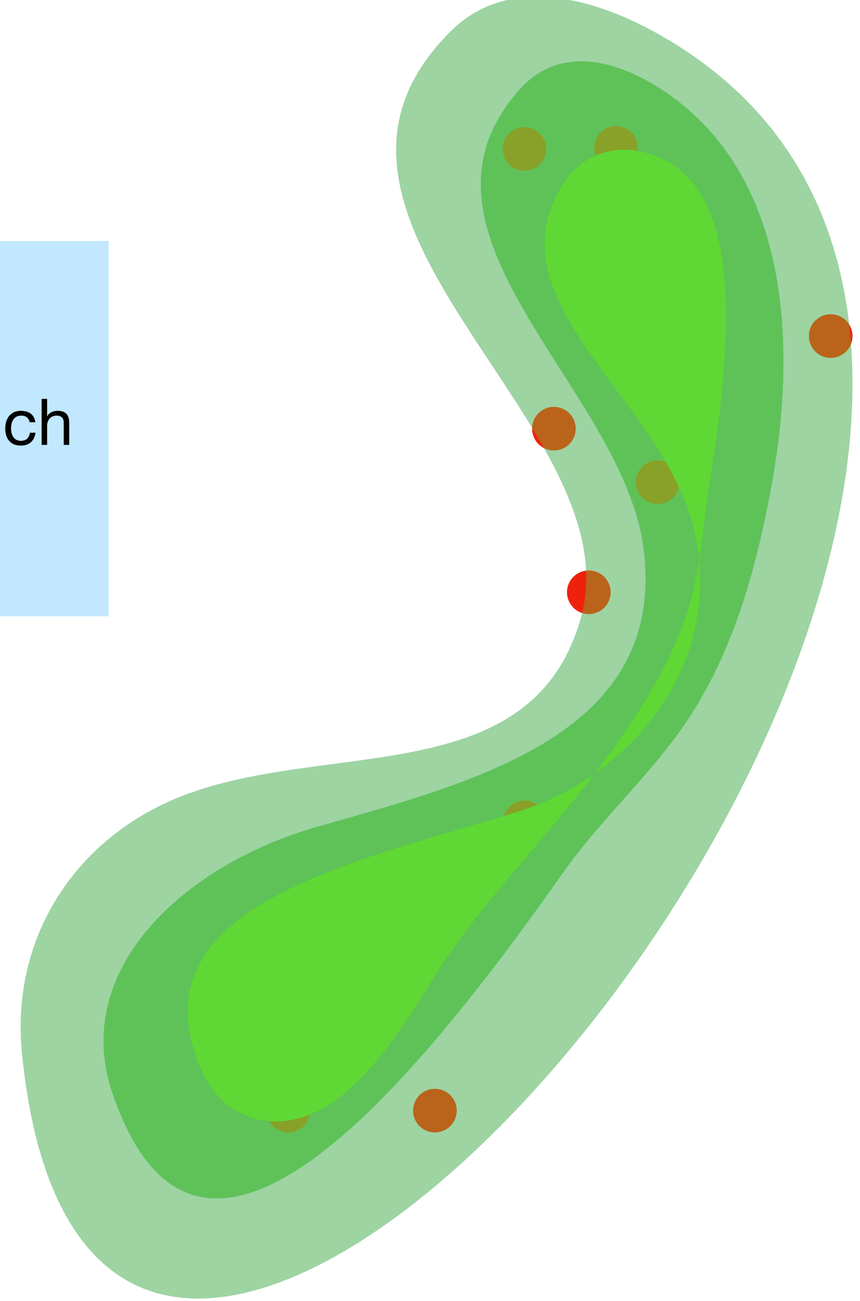
Each image i is a point in $x_i \in \mathbb{R}^d$

Assumption:

There exists a probability distribution p_{data} such that each image $x_i \sim p_{\text{data}}$

Objective:

Approximate p_{data} with a parametric distribution p_θ « easy to sample »



Trade-off:

Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

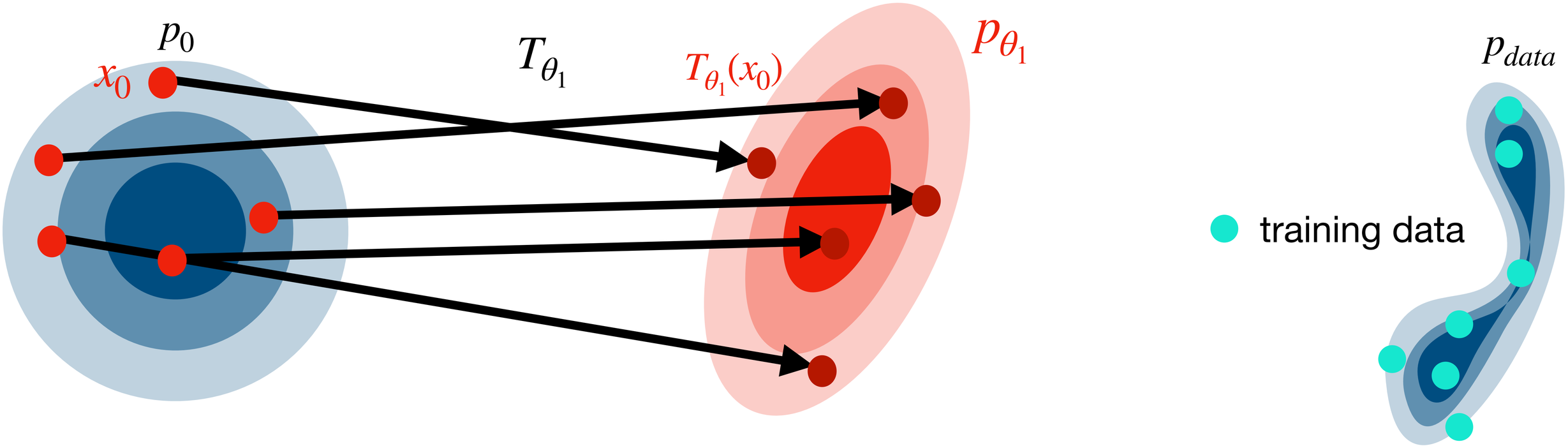
- Choose p_0 easy to sample
- Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data}$$

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize

$$\min_\theta D(p_{\text{data}}, p_\theta)$$



Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

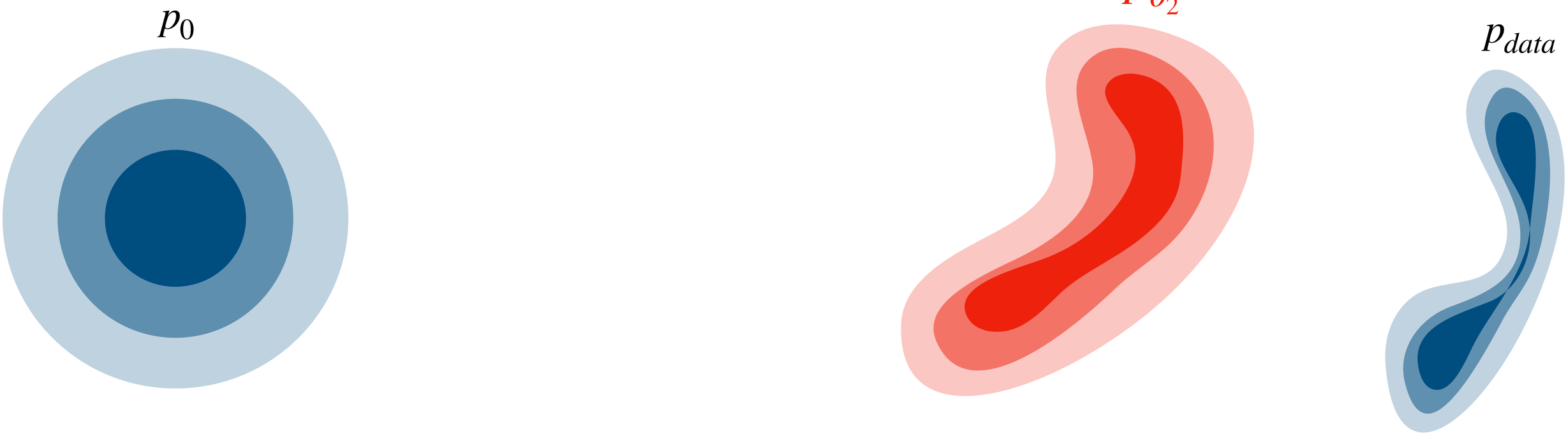
- ➡ Choose p_0 easy to sample
- ➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data}$$

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize

$$\min_\theta D(p_{\text{data}}, p_\theta)$$



Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

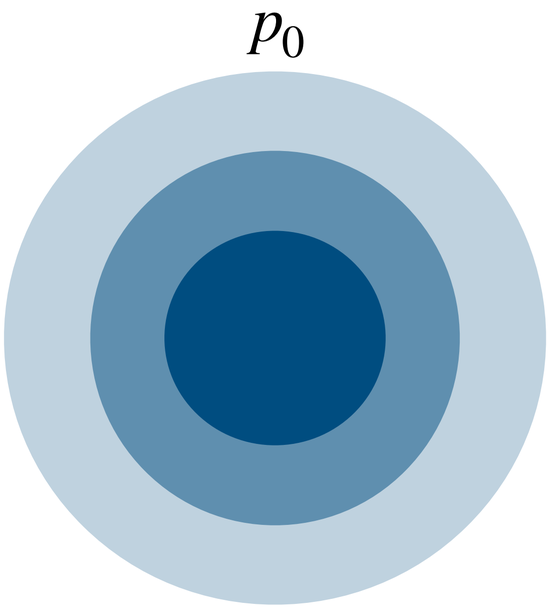
- ➡ Choose p_0 easy to sample
- ➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

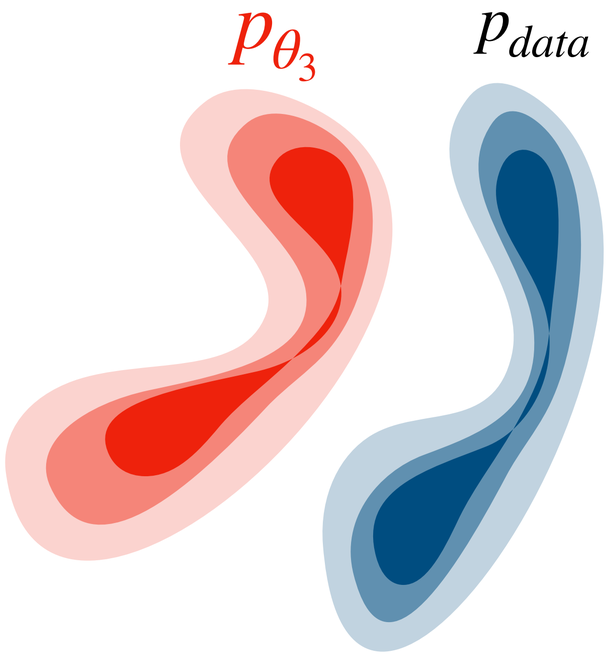
$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data}$$

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize

$$\min_\theta D(p_{\text{data}}, p_\theta)$$





Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

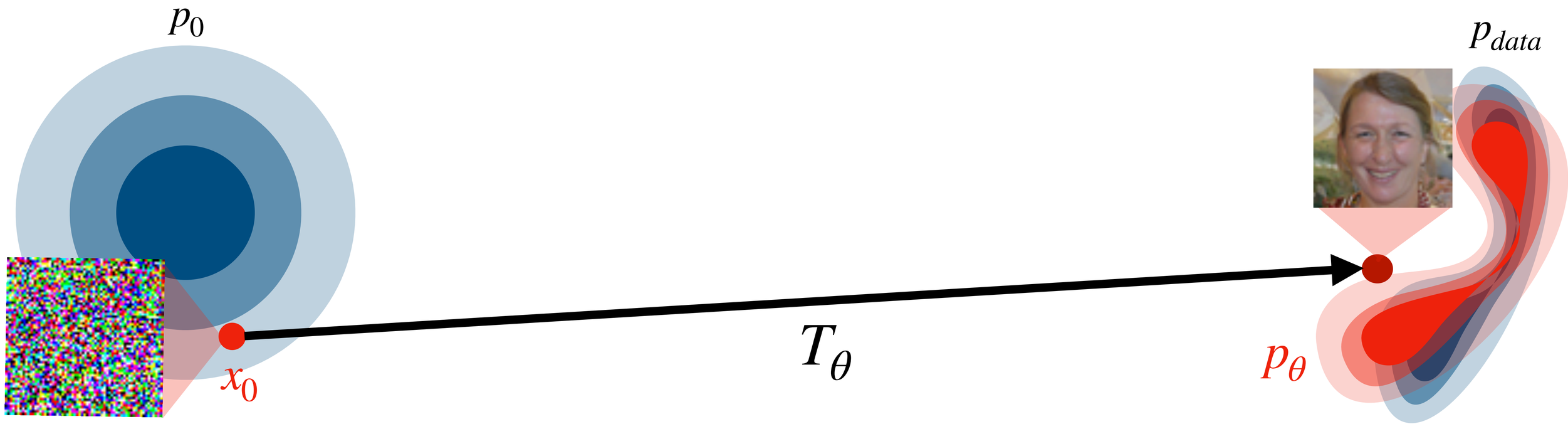
- Choose p_0 easy to sample
- Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data}$$

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize

$$\min_\theta D(p_{\text{data}}, p_\theta)$$



Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

- Choose p_0 easy to sample
- Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data}$$

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize

$$\min_\theta D(p_{\text{data}}, p_\theta)$$

Challenges :

- Very high dimension: $d \approx 10^6$ for images
- Sample quality vs diversity
- Training efficiency
- Sampling efficiency

Generative models given samples of p_{data}, generate new samples of p_{data}

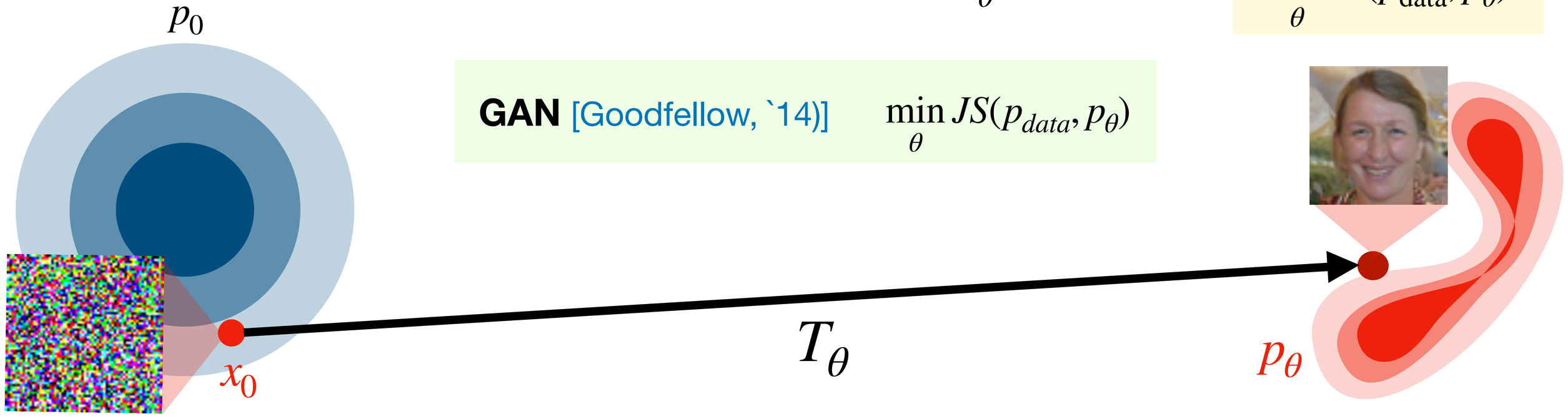
Flow-based methods:

- ➡ Choose p_0 easy to sample
- ➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data} \qquad p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$

GAN [Goodfellow, `14)] $\min_\theta JS(p_{data}, p_\theta)$



Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

- ➡ Choose p_0 easy to sample
- ➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

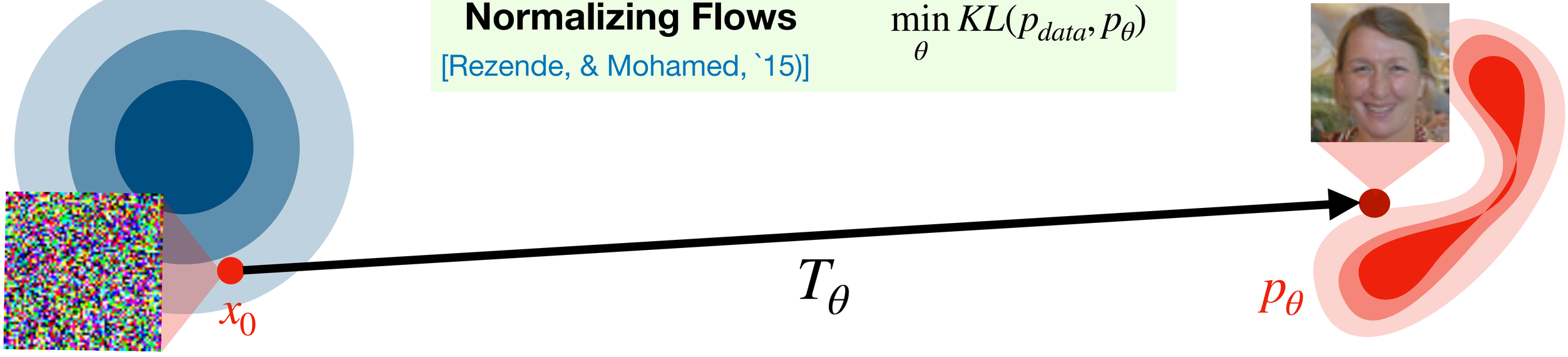
$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data}$$

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$

Normalizing Flows $\min_\theta KL(p_{data}, p_\theta)$

[Rezende, & Mohamed, `15)]



Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

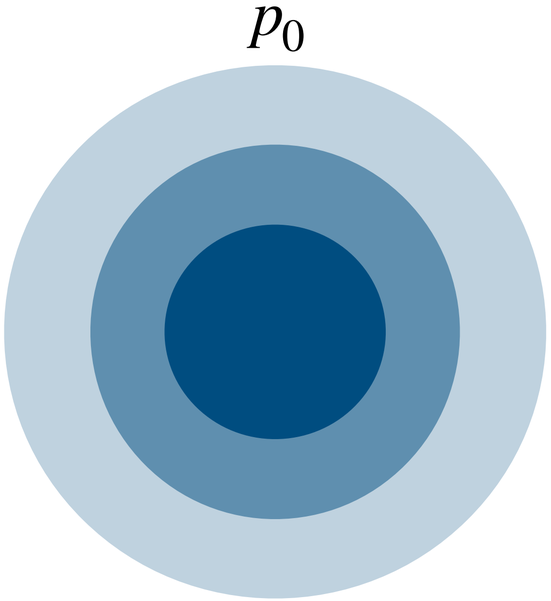
➡ Choose p_0 easy to sample

➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$x_0 \sim p_0,\ x = T_\theta(x_0)$ samples p_{data}

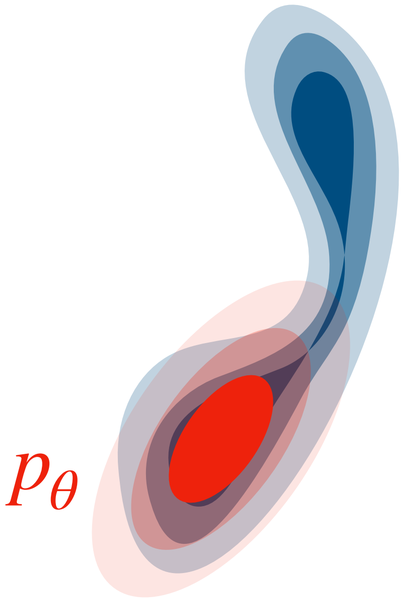
$p_\theta := T_\theta \# p_0 \approx p_{data}$

Train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$



Normalizing Flows $\min_\theta KL(p_{data}, p_\theta)$

[Rezende, & Mohamed, `15)]



✖ Hard to train a single-step model: unstable training, mode collapse…

Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

➡ Choose p_0 easy to sample

➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$x_0 \sim p_0,\ x = T_\theta(x_0)$ samples p_{data}

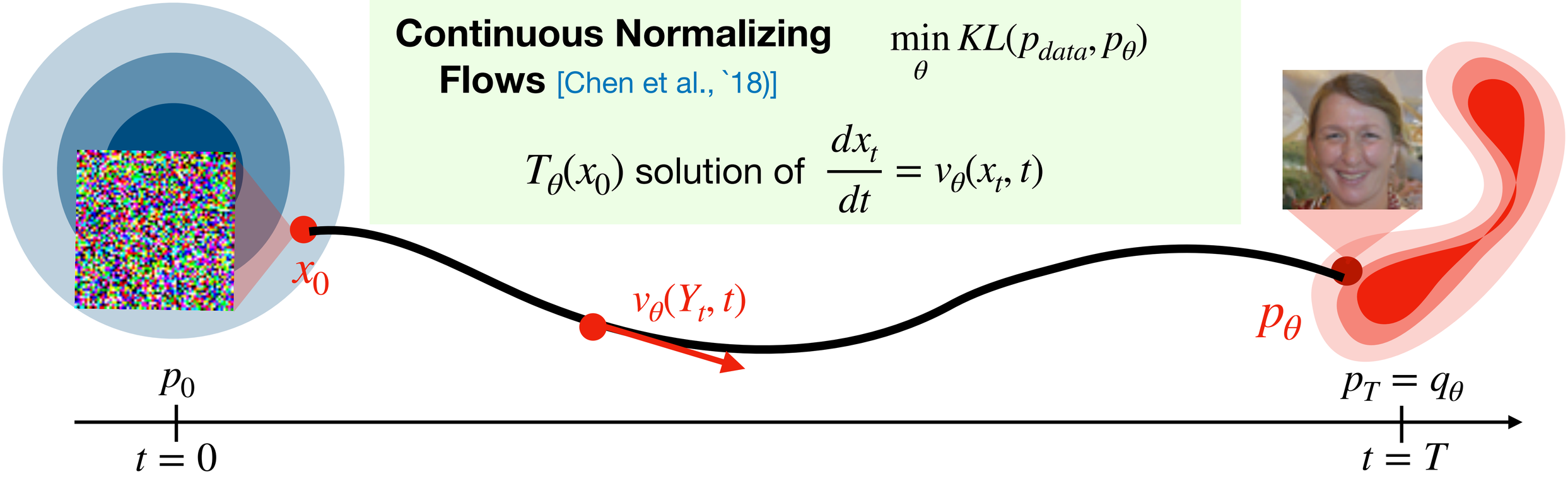
$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$

Continuous Normalizing Flows [Chen et al., `18)]

$\min_\theta KL(p_{data}, p_\theta)$

$T_\theta(x_0)$ solution of $\frac{dx_t}{dt} = v_\theta(x_t, t)$



Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

➡ Choose p_0 easy to sample

➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$x_0 \sim p_0,\ x = T_\theta(x_0)$ samples p_{data}

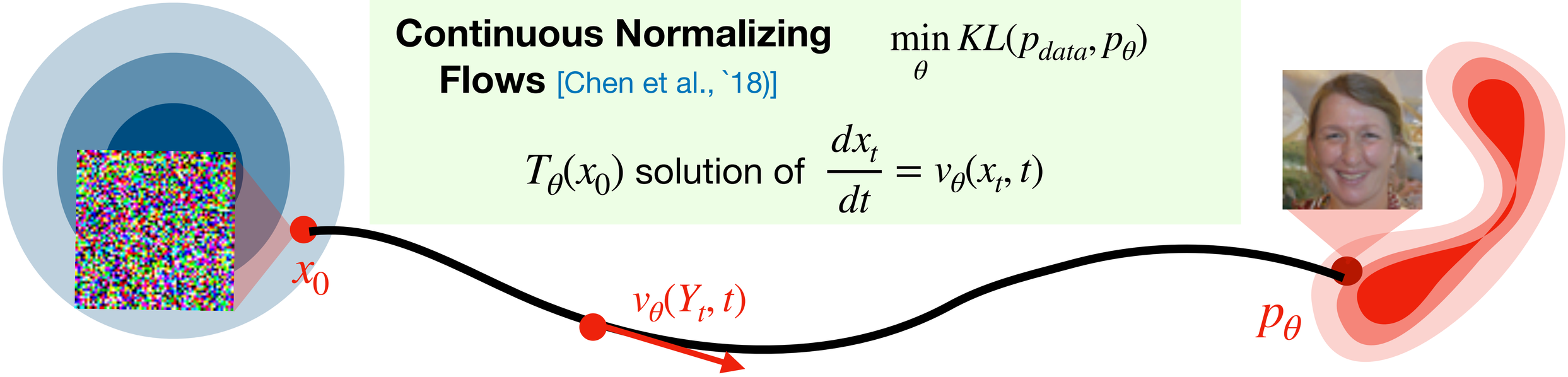
$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$

Continuous Normalizing Flows [Chen et al., `18)]

$\min_\theta KL(p_{data}, p_\theta)$

$T_\theta(x_0)$ solution of $\frac{dx_t}{dt} = v_\theta(x_t, t)$



✓ Learns a continuous path
Gaussian → data

✗ Hard training: supervision only **at final time**

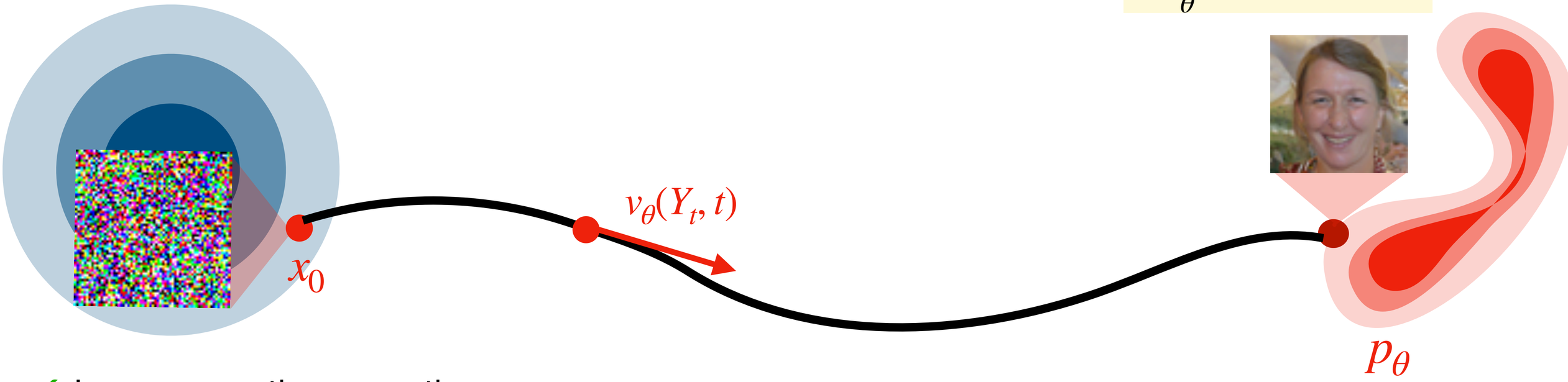
Generative models given samples of p_{data}, generate new samples of p_{data}

Diffusion Models [Song et al., `21] **/ Flow Matching** [Lipman et al., `23]

$T_\theta(x_0)$ solution of $\frac{dx_t}{dt} = v_\theta(x_t, t)$

v_θ trained to approximate a predefined velocity v^*

$$\min_\theta D(v^*, v^\theta)$$



✓ Learns a continuous path
Gaussian → data

✓ Easy training: supervision **at all times**

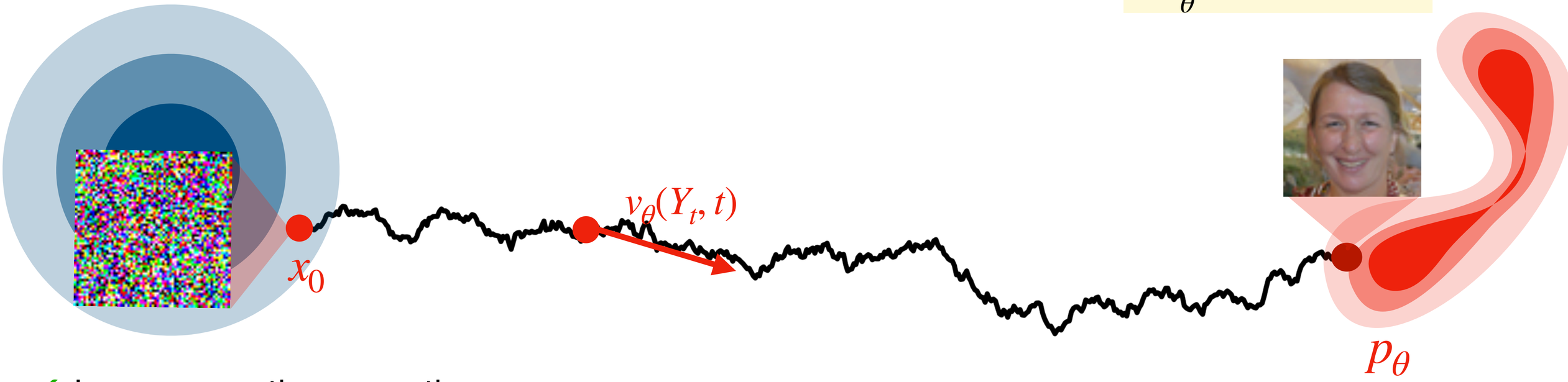
Generative models given samples of p_{data}, generate new samples of p_{data}

Diffusion Models [Song et al., `21] **/ Flow Matching** [Lipman et al., `23]

$T_\theta(x_0)$ solution of $\frac{dx_t}{dt} = v_\theta(x_t, t) + \sqrt{2a_t}\, dB_t$

v_θ trained to approximate a predefined velocity v^*

$$\min_\theta D(v^*, v^\theta)$$

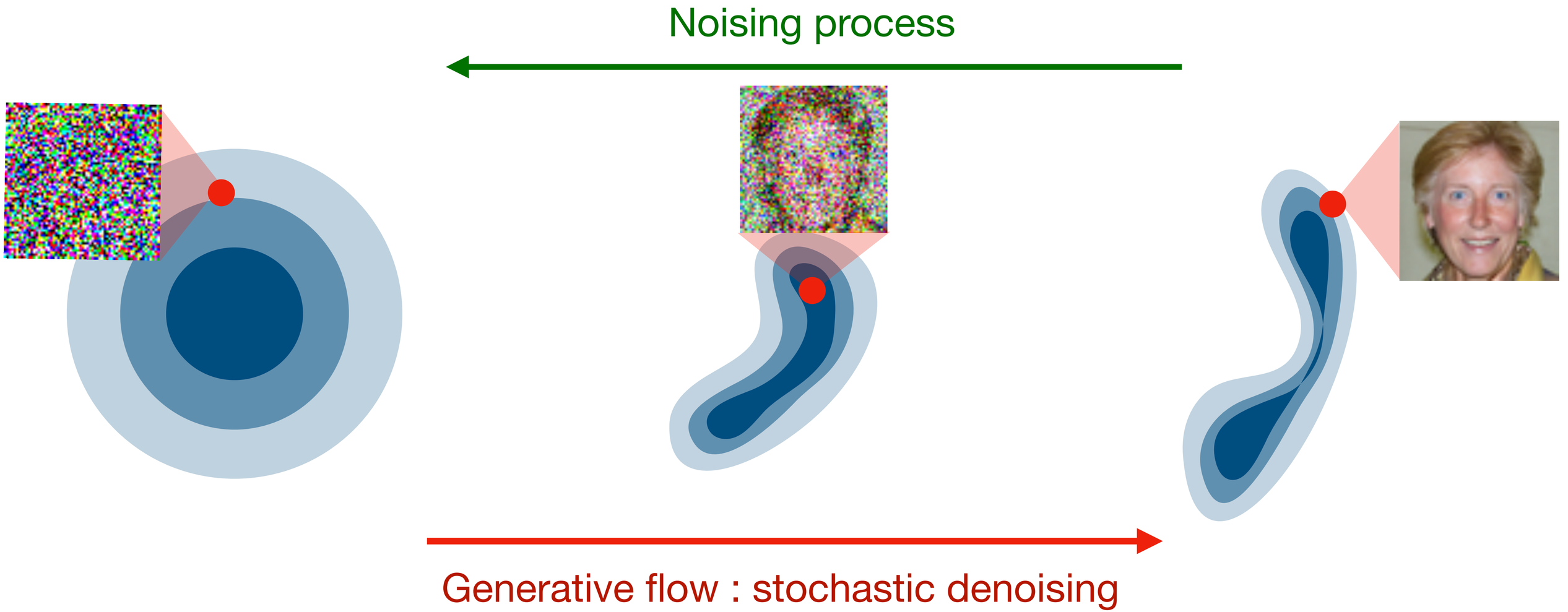


✓ Learns a continuous path
Gaussian → data

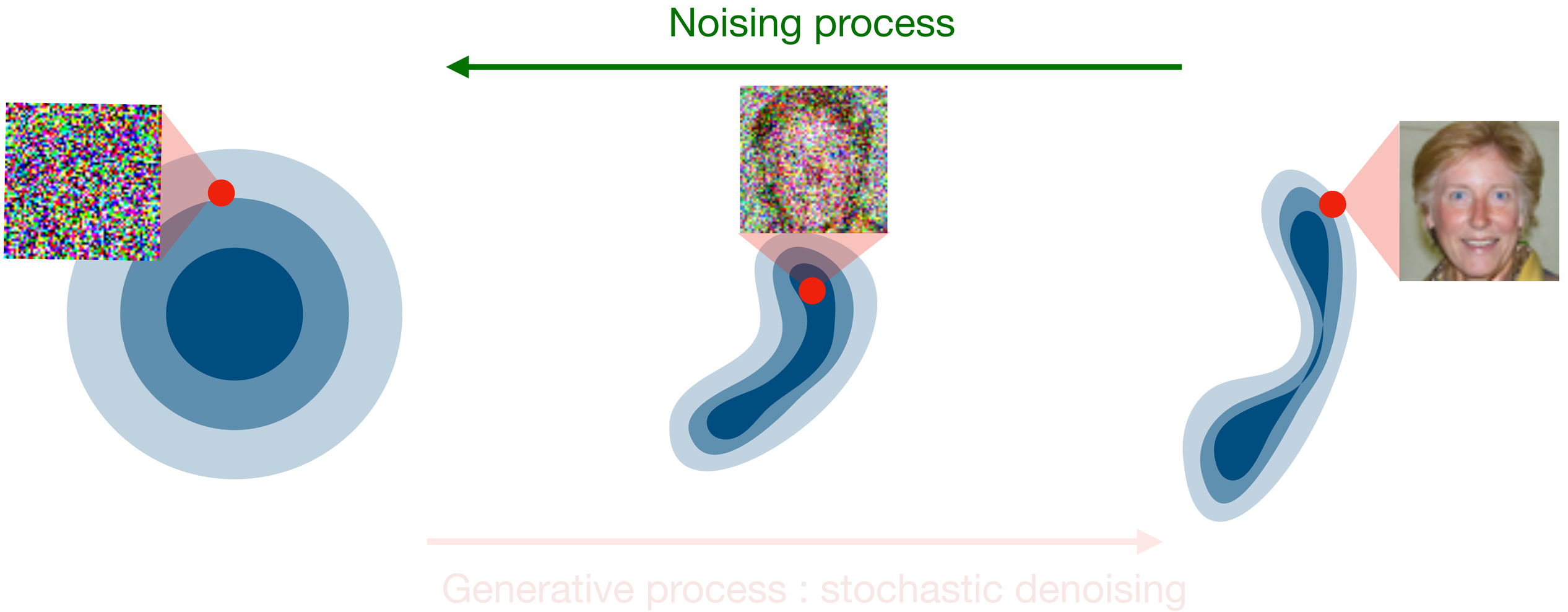
✓ Easy training: supervision **at all times**

Denoising: the building bock of diffusion models

Diffusion models



Diffusion models

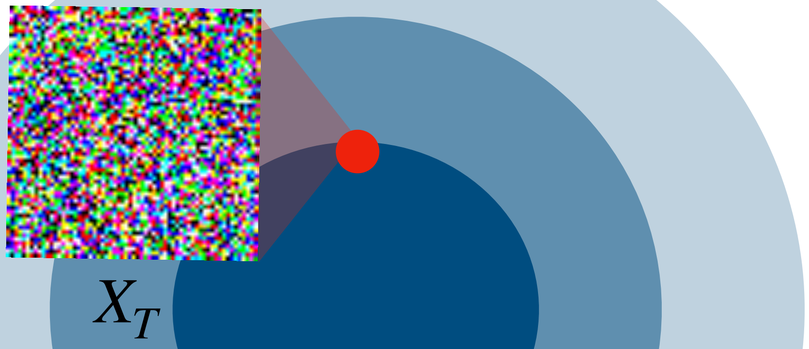


Diffusion models **Noising process** (from data to noise)

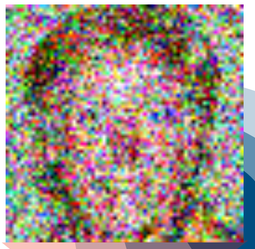
Variance Exploding (VE)

$$X_0 \sim p_{\text{data}}, W_t \sim \mathcal{N}(0, \text{Id}), \quad X_t \overset{d}{=} X_0 + \sigma_t W_t$$

$\sigma_t > 0, \quad \sigma_t \to \infty$



X_T



X_t

X_0

$q_T \approx \mathcal{N}(0, \sigma_T^2 \text{Id})$

$q_t = \text{Law}(X_0 + \sigma_t W_t)$

$q_0 = p_{data}$

$t = T \gg 1$

t

$t = 0$

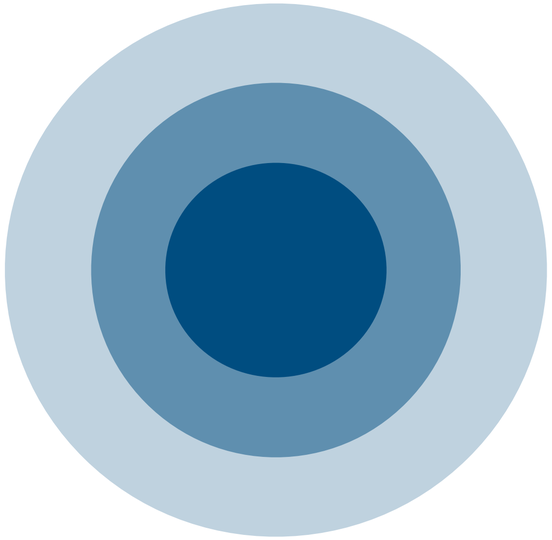
$$\text{Cov}(X_t) = \text{Cov}(X_0) + \sigma_t^2 \text{Id}$$

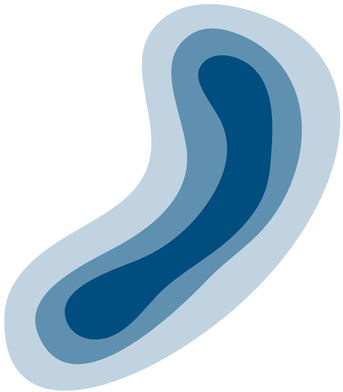
Diffusion models **Noising process** (from data to noise)

Variance Preserving (VP)

$$X_0 \sim p_{\text{data}}, W_t \sim \mathcal{N}(0, \text{Id}), \; X_t \overset{d}{=} \eta_t(X_0 + \sigma_t W_t)$$

$$\sigma_t > 0, \quad \eta_t = \frac{1}{\sqrt{\sigma_t^2 + 1}}$$





$q_T \approx \mathcal{N}(0, \text{Id})$ $\quad$ $q_t = \text{Law}(\eta_t(X_0 + \sigma_t w_t))$ $\quad$ $q_0 = p_{data}$

$t = T \gg 1$ $\quad$ t $\quad$ $t = 0$

$$\text{Cov}(X_t) = \eta_t^2 \, \text{Cov}(X_0) + \left(1 - \eta_t^2\right) \text{Id}$$

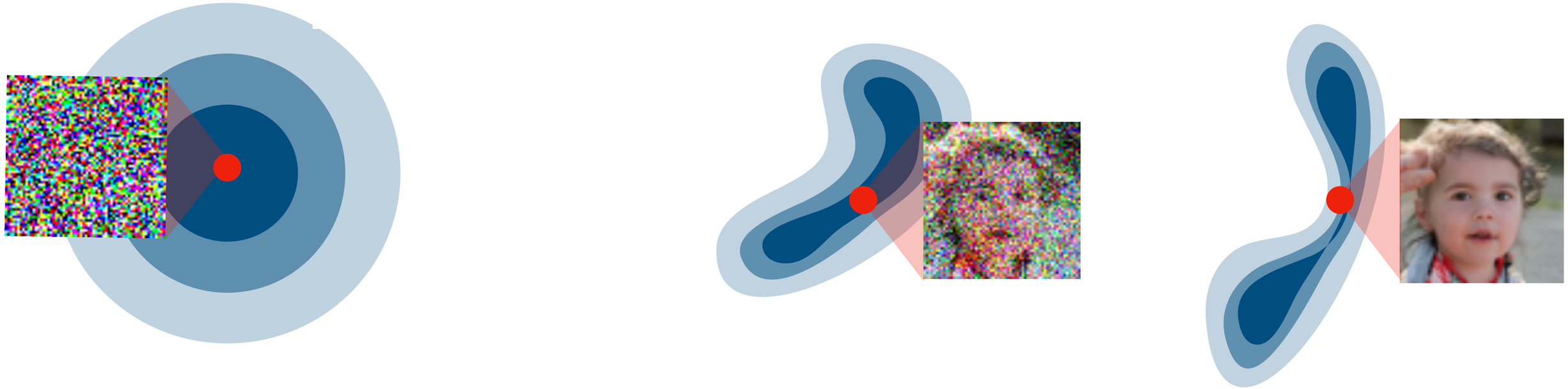
Diffusion models **Noising process** (from data to noise)

General schedule (η_t, σ_t)

$$X_0 \sim p_{\text{data}}, W_t \sim \mathcal{N}(0, \text{Id}), \quad X_t \overset{d}{=} \eta_t(X_0 + \sigma_t W_t)$$

Diffusion models

Noising process



Generative process : stochastic denoising

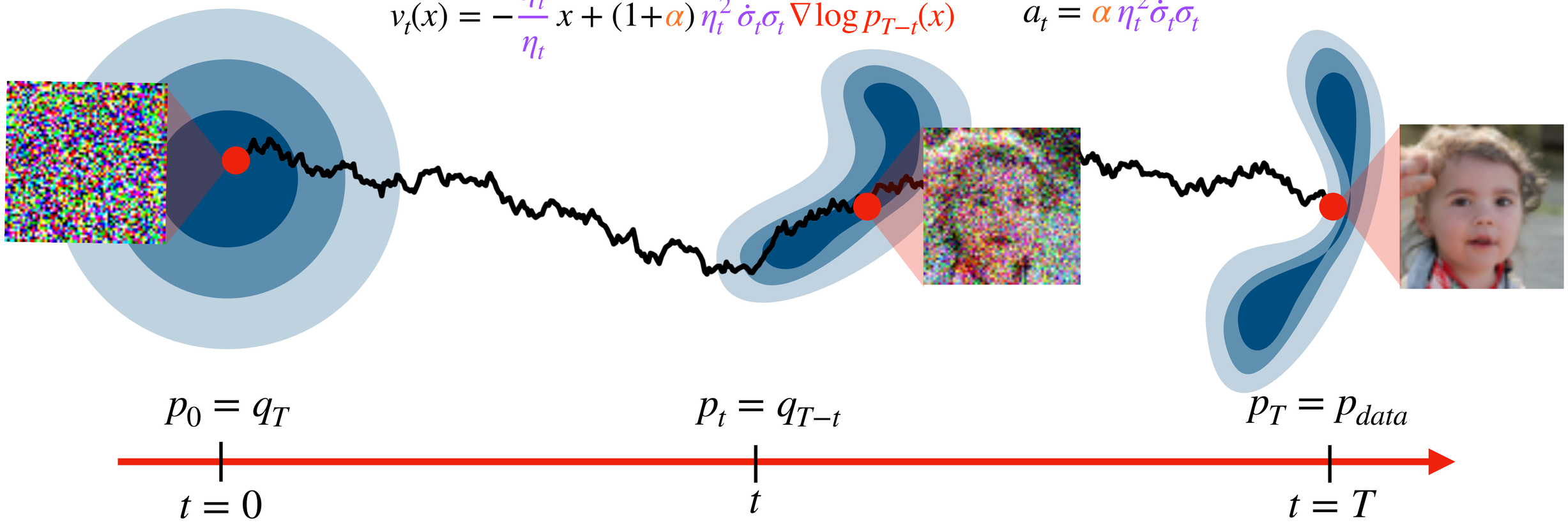
Diffusion models **Generative process** (from noise to data)

$p_t = q_{T-t}$ is sampled from the **SDE**

For $\alpha \geq 0$,

$$Y_0 \sim p_0\,, \quad dY_t = v_t(Y_t)\,dt + \sqrt{2a_t}\,dW_t$$

$$v_t(x) = -\frac{\dot{\eta}_t}{\eta_t}\,x + (1+\alpha)\,\eta_t^2\,\dot{\sigma}_t\sigma_t\,\nabla \log p_{T-t}(x) \qquad a_t = \alpha\,\eta_t^2\dot{\sigma}_t\sigma_t$$



Parameters of the algorithm: diffusion coefficient α, time schedulers η_t, σ_t

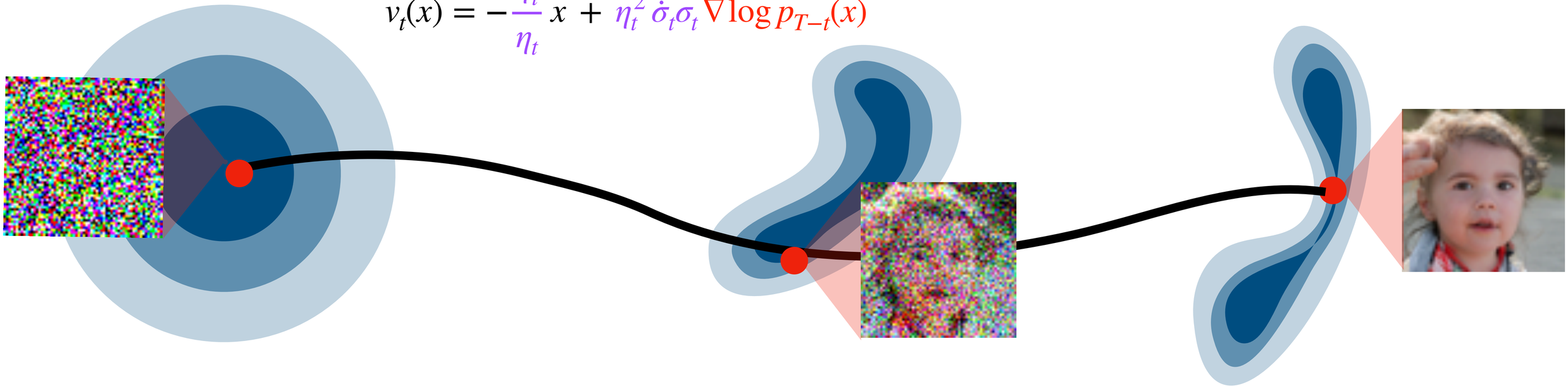
Diffusion models **Generative process** (from noise to data)

$p_t = q_{T-t}$ is sampled from the **ODE**

For $\alpha \geq 0$,

$$Y_0 \sim p_0\,, \quad dY_t = v_t(Y_t)\,dt$$

$$v_t(x) = -\frac{\dot{\eta}_t}{\eta_t}\,x + \eta_t^2\,\dot{\sigma}_t\sigma_t\,\nabla \log p_{T-t}(x)$$



$p_0 = q_T$ $\quad$ $p_t = q_{T-t}$ $\quad$ $p_T = p_{data}$

$t = 0$ $\quad$ t $\quad$ $t = T$

Parameters of the algorithm: diffusion coefficient α, time schedulers η_t, σ_t

Diffusion models errors

$$Y_0 \sim p_0\,, \quad dY_t = v_t(Y_t)\,dt + \sqrt{2a_t}\,dW_t$$

$$v_t(x) = -\frac{\dot{\eta}_{T-t}}{\eta_{T-t}}\,x + (1+\alpha)\,\eta^2_{T-t}\,\dot{\sigma}_{T-t}\sigma_{T-t}\,\nabla \log p_{T-t}(x)$$

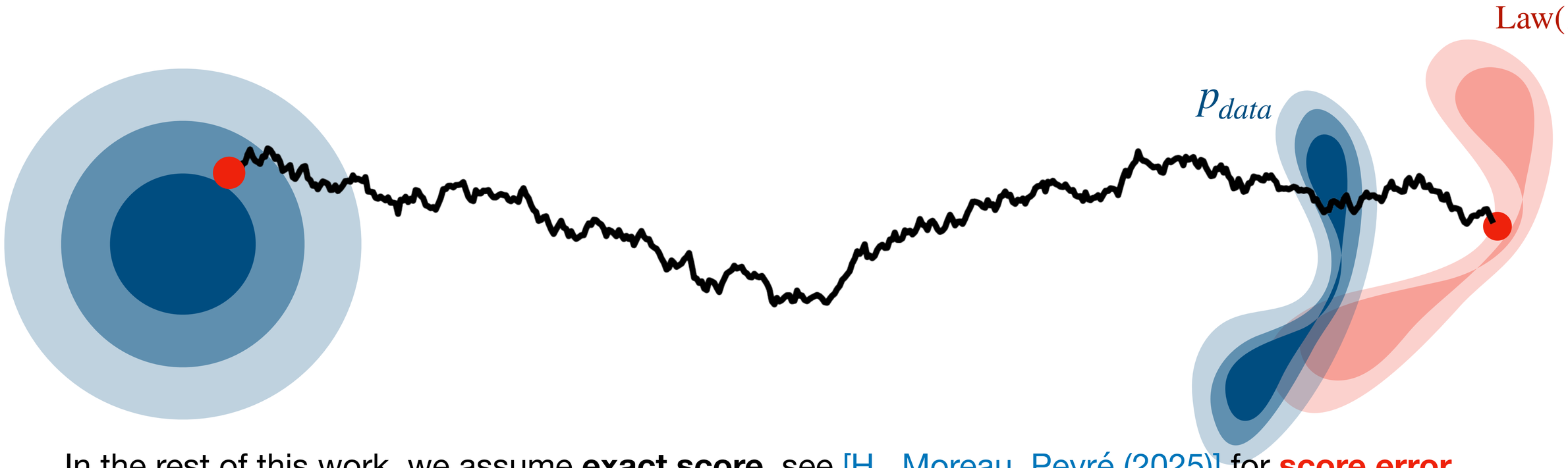
Diffusion models errors: score training error

$$Y_0 \sim p_0\,, \quad dY_t = v_t(Y_t)\,dt + \sqrt{2a_t}\,dW_t$$

$$v_t(x) = -\frac{\dot\eta_t}{\eta_t}\,x + (1+\alpha)\,\eta_t^2\,\dot\sigma_t\sigma_t\,\nabla \log p_{T-t}(x)$$

$$s_t^\theta \approx \nabla \log p_t$$

Trained by denoising Score Matching on a finite dataset $\{x_i \sim p\}_{i \le N_{data}}$

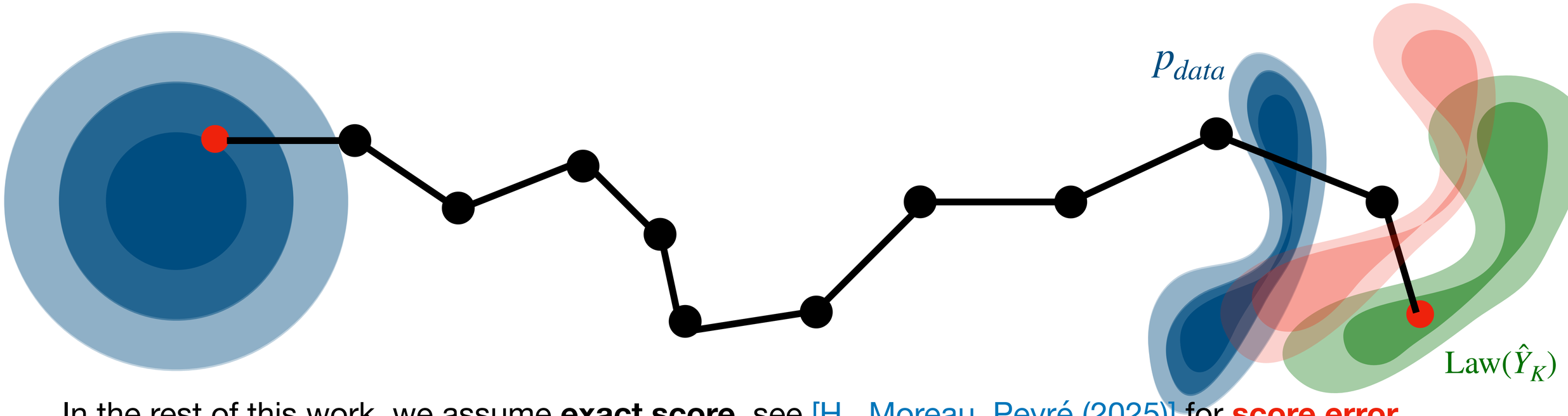


In the rest of this work, we assume **exact score,** see [H., Moreau, Peyré (2025)] for **score error.**

Diffusion models errors: discretization error

Euler discretization: for $\gamma = \frac{T}{K}$,

$$\hat{Y}_0 \sim p_0\,, \quad \hat{Y}_{k+1} = \hat{Y}_k + \gamma\, v_{t_k}(\hat{Y}_k) + \sqrt{2\gamma a_{t_k}}\, W_k$$



In the rest of this work, we assume **exact score,** see [H., Moreau, Peyré (2025)] for **score error.**

Diffusion models for posterior sampling $p(x \mid y = Ax + w)$

Apply the same algorithm with p_{data} replaced by $p(x \mid y)$

For $\alpha \geq 0$,

$$Y_0 \sim p_0\,, \quad dY_t = v_t(Y_t)\,dt + \sqrt{2a_t}\,dW_t$$

$$v_t(x) = -\frac{\dot{\eta}_{T-t}}{\eta_{T-t}}\,x + (1+\alpha)\,\eta_{T-t}^2\,\dot{\sigma}_{T-t}\sigma_{T-t}\,\nabla \log p_{T-t}(x \mid y)$$

Using Baye's formula, for $x_t \sim p_t$

$$\nabla \log p_t(x_t \mid y) = \nabla \log p_t(x_t) + \nabla \log p_t(y \mid x_t)$$

✓ Unconditional score training

$$= \int p(y \mid x_0) p(x_0 \mid x_t) dx_0$$

✗ Untractable

DPS [Chung et. al, `23], PiGDM [Song et. al, `23], Moment Matching [Rozet et. al, `24] :

Use the Gaussian approximation $p(x_0 \mid x_t) \approx \mathcal{N}(\mathbb{E}[x_0 \mid x_t], \mathbb{V}[x_0 \mid x_t])$

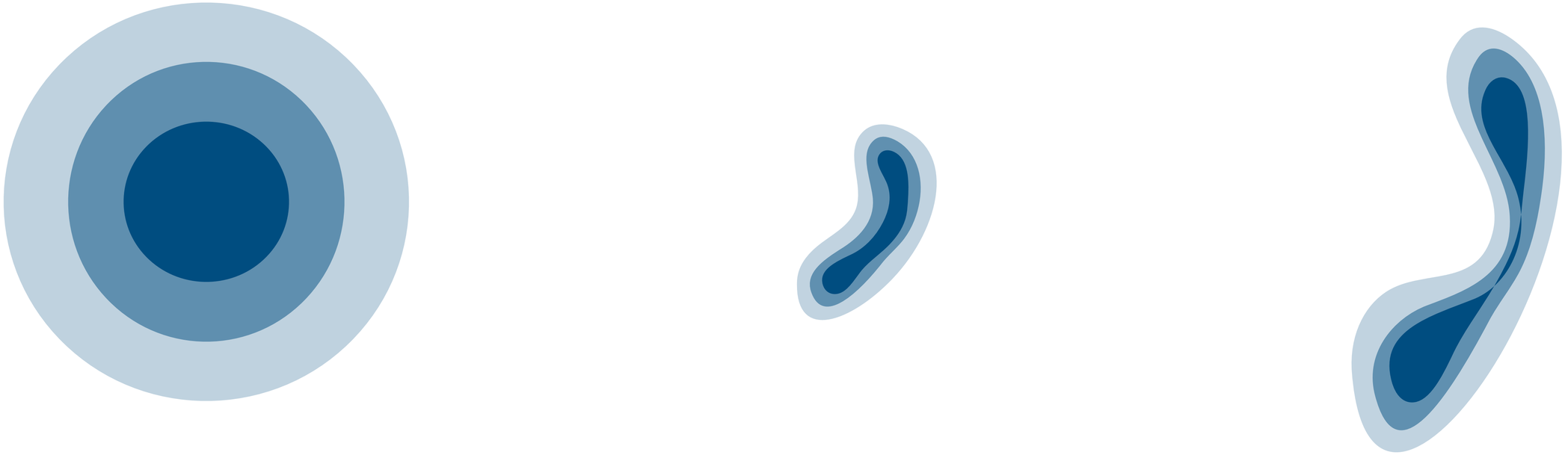
Related to a pertrained denoiser by Tweedie's formulas

Diffusion models **Other Noising Process** (from data to noise)

Flow Matching (FM)

$$X_0 \sim p_{\text{data}}, W_t \sim \mathcal{N}(0, \text{Id}), \quad X_t \overset{d}{=} (1-t)X_0 + tW_t$$

$$\sigma_t = \frac{t}{1-t}, \quad \eta_t = \frac{1}{1+\sigma_t}$$



$q_1 = \mathcal{N}(0, \text{Id})$ $q_t = \text{Law}(\eta_t(X_0 + \sigma_t X_t))$ $q_0 = p_{data}$

$t = T = 1$ t $t = 0$

$$\text{Cov}(X_t) = (1-t)^2 \text{Cov}(X_0) + t^2 \,\text{Id}$$

Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

➡ Choose p_0 easy to sample

➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$x_0 \sim p_0,\ x = T_\theta(x_0)$ samples p_{data}

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

Train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$

GANs [Goodfellow, `14)]

$$\min_\theta JS(p_{data}, p_\theta)$$

Variational form: $$JS(P,Q) = \frac{1}{2} \sup_D \left[\mathbb{E}_{x\sim P} \log D(x) + \mathbb{E}_{x\sim Q} \log(1 - D(x)) \right] + cst$$

Parametrize D by a second neural network (discriminator) D_ϕ and training of parameters (θ, ϕ)

$$\min_\theta \max_\phi \left[\mathbb{E}_{x\sim p_{\text{data}}} \log D_\phi(x) + \mathbb{E}_{x\sim p_0} \log\left(1 - D_\phi(T_\theta(x))\right) \right]$$

Extended to all D with variational forms: f-GAN, WGAN, MMD-GAN ….

✓ First « working » generative models for images

✗ Unstable training (oscillations, mode collapse ….)

Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

➡ Choose p_0 easy to sample

➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$x_0 \sim p_0,\ x = T_\theta(x_0)$ samples p_{data}

$$p_\theta := T_\theta \# p_0 \approx p_{data}$$

First idea: train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$

Normalizing Flows

[Rezende, & Mohamed, `15)]

$$\min_\theta KL(p_{data}, p_\theta) = \min_\theta \mathbb{E}_{x\sim p_{data}}\left[-\log p_\theta(x)\right]$$

Change of variable formula:

$$\log p_\theta(x) = \log p_0(T_\theta^{-1}(x)) + \log\left|\det \frac{dT_\theta^{-1}(x)}{dx}\right|$$

❌ Requires to parametrize T_θ to be invertible

Generative models given samples of p_{data}, generate new samples of p_{data}

Flow-based methods:

- ➡ Choose p_0 easy to sample
- ➡ Find a map $T_\theta : \mathbb{R}^d \to \mathbb{R}^d$ such that

$$x_0 \sim p_0,\ x = T_\theta(x_0) \text{ samples } p_{data} \qquad p_\theta := T_\theta \# p_0 \approx p_{data}$$

First idea: train a neural network T_θ to minimize $\min_\theta D(p_{\text{data}}, p_\theta)$

Continuous Normalizing Flows: $\min_\theta KL(p_{data}, p_\theta)$

[Chen et al., `18)]

$T_\theta(x_0)$: solution of the ODE $Y_0 = x_0, \quad dY_t = v_\theta(Y_t, t)\, dt$