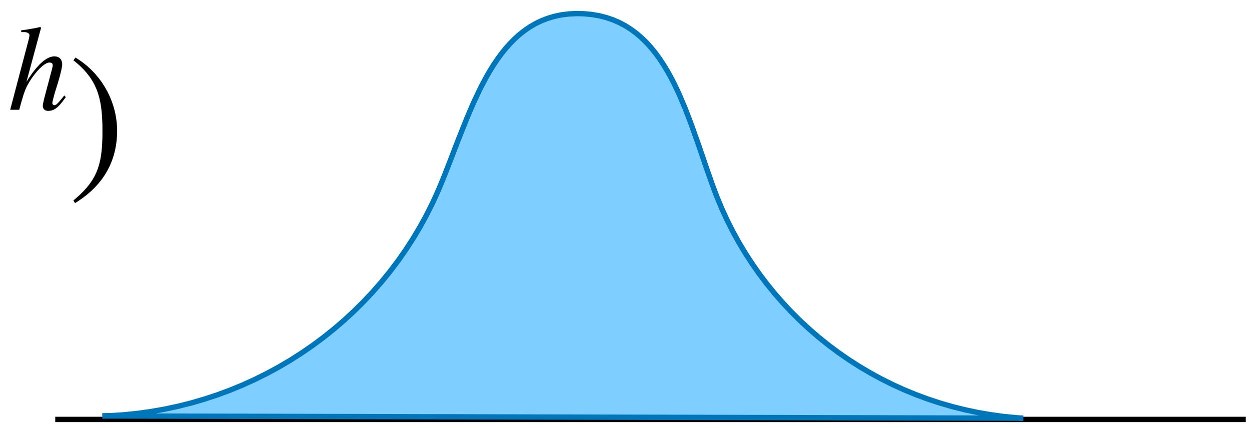
Diffusion Models and Imitation Learning

Sanjiban Choudhury

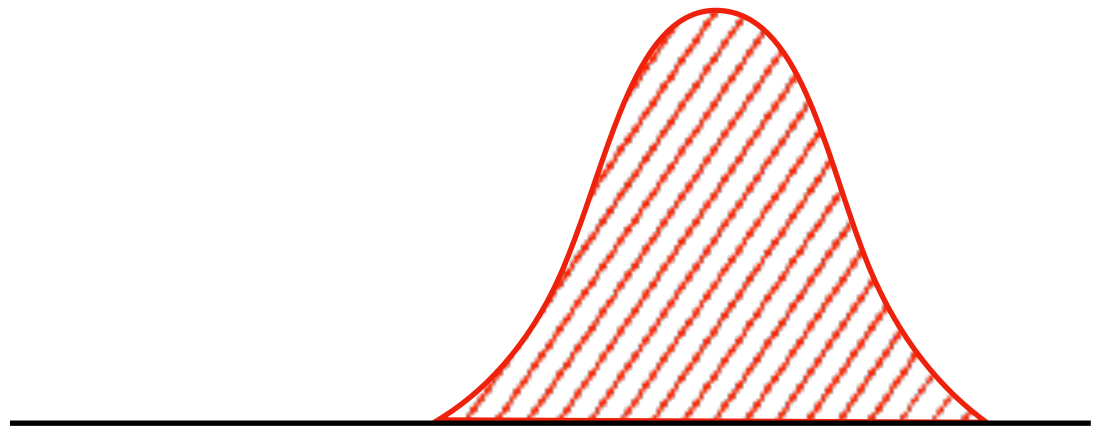
Cornell Bowers CIS
Computer Science

The Distribution Matching Problem

$P_{expert}(\xi^h)$



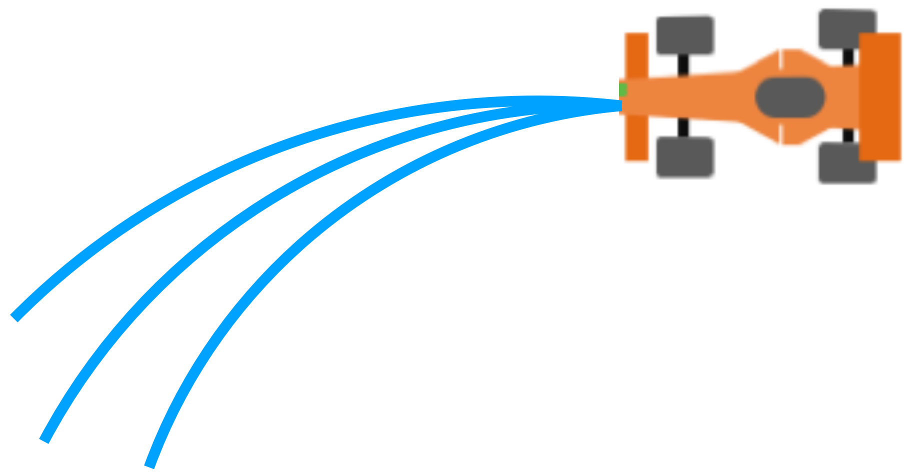
(Unknown) expert distribution



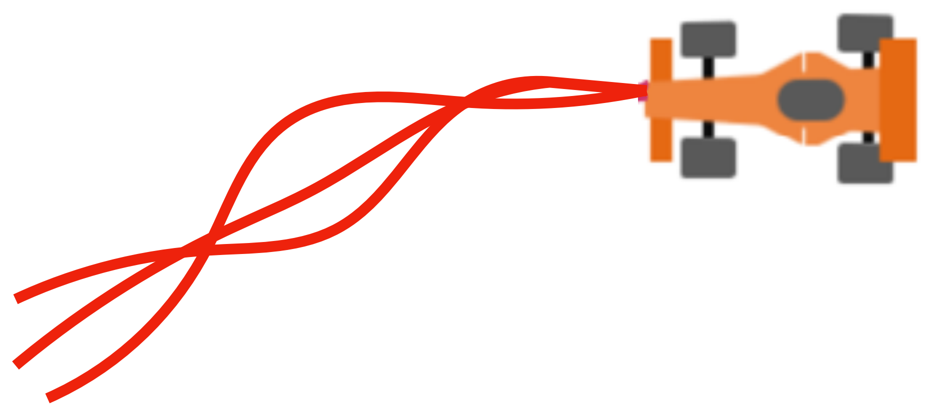
$P_\theta(\xi)$

Learn distribution over trajectories

All we see are expert samples

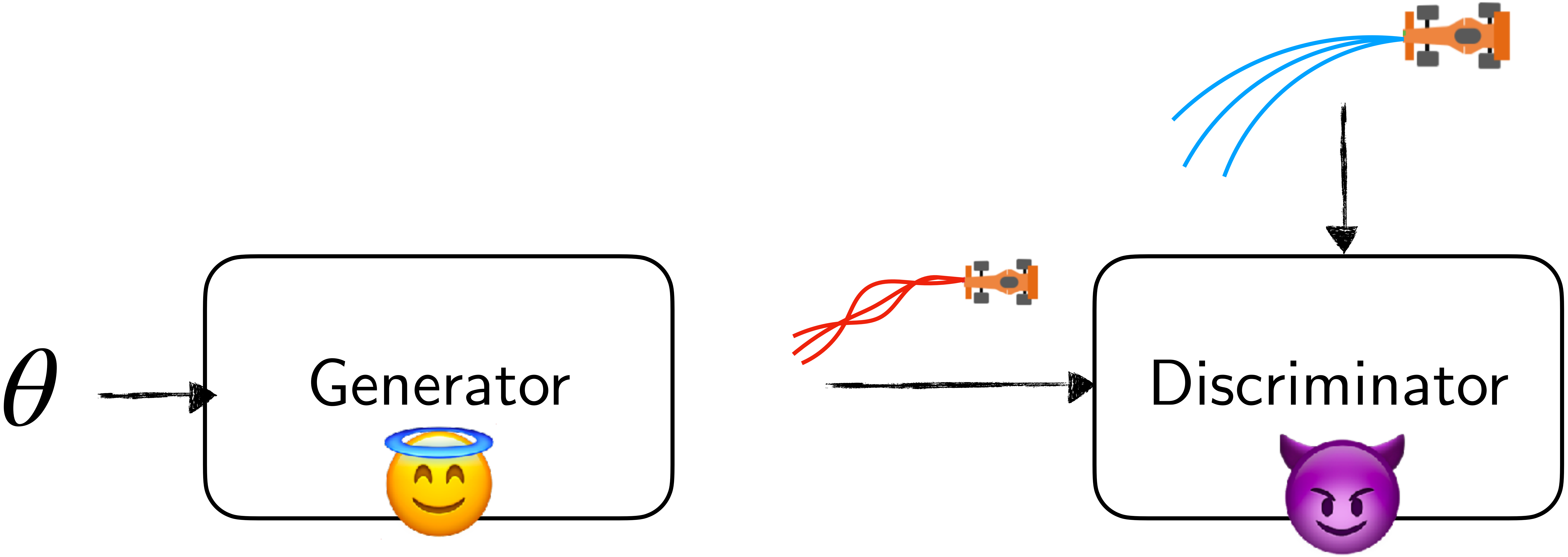


Learner can also generate samples



What loss should we use?

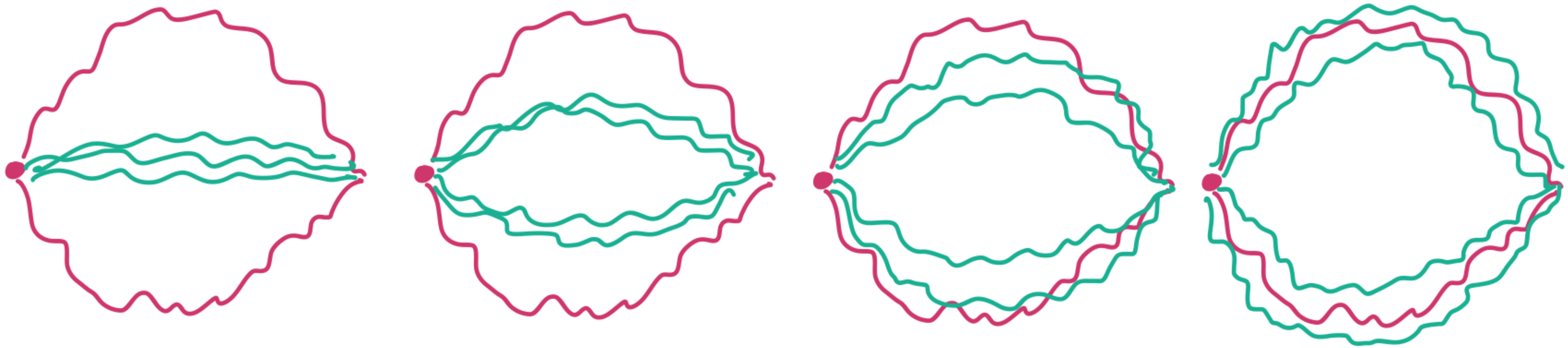
Use GANs to estimate divergence!



Minimize discriminator loss!

Maximally discriminate between learner and generator samples

One such instance: MaxEnt IRL!



for $i = 1, \ldots, N$ # Loop over datapoints

$$\xi_i \sim \frac{1}{Z} \exp\left(-C_\theta(\xi, \phi_i)\right)$$ # Call planner!

$$\theta^+ = \theta - \eta[\nabla_\theta C_\theta(\xi_i^h, \phi_i) - \nabla_\theta C_\theta(\xi_i, \phi_i)]$$ # Update cost

(Push down human cost) *(Push up planner cost)*

The Problem

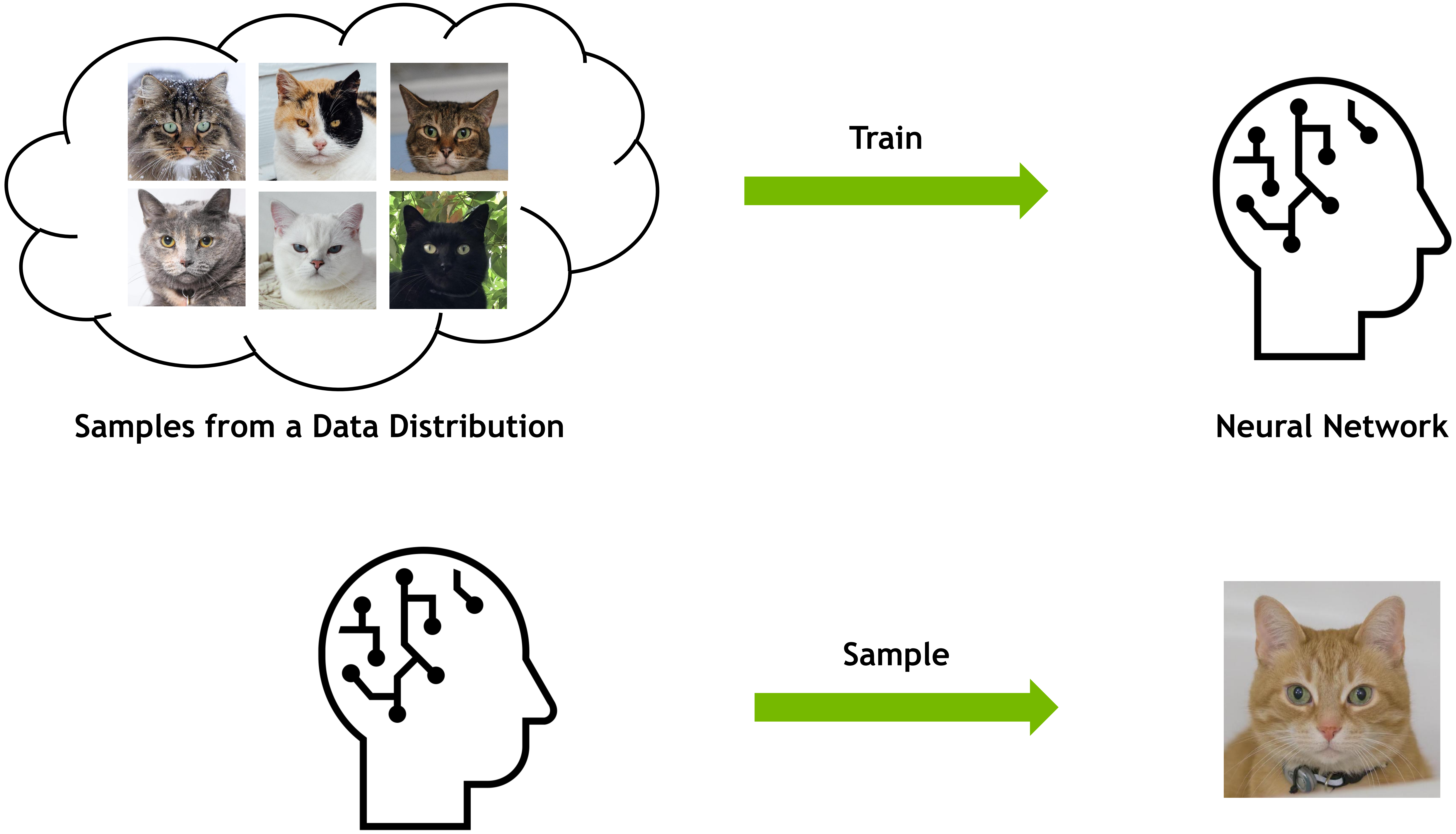
Adversarial games are challenging to solve!

Typically require tricks to make sure discriminator does not get too powerful (gradient penalty, early stopping etc)

Are there simpler ways to learn $p_\theta(\xi) \approx p^*(\xi)$?

Steal from computer vision!

Diffusion Models: Latest Generative Model!



Xander Steenbrugge
@xsteenbrugge

"Voyage through Time"
is my first artpiece using
blown away with the pos

What you can do with this today

```
seed_and_prompt_sequence = [
        3764, 'in the beginning there was nothing, just darkness',
        1537, 'special effects render of the big bang',
        6573, 'HD photo of a large amount of spiral galaxies',
        1791, 'early planet formation in the solar system',
        9973, 'the Hadean earth was bombarded with asteroids and massive volcanic eruptions',
        736,  'panoramic view of earth with ocean surrounding newly formed land and volcanos',
        3639, 'hydrothermal vents at the bottom of the ocean',
        3559, 'bacteria under a microscope',
        4724, 'bacteria under a microscope',
        3359, 'ammonites floating in the ocean',
        6344, 'the first reptile to leave the ocean and crawl onto the land',
        6344, 'the first reptile to leave the ocean and crawl onto the land',
        6813, 'massive brachiosaurus walking amidst a green mountain range',
        6678, 'the exctinction of the dinosaurs be a huge meteorite',
        7450, 'small mammals thriving in a cave',
        9766, 'small, prehistoric mammals living in the jungle',
        5009, 'group of monkeys in the forest',
        7287, 'HD photograph of neanderthal, the first man',
        6008, 'cave painting',
        208,  'cavemen tribe gathered around a fire at night looking at the stars',
        2222, 'maasai tribe hunting on the savanna with spears',
        571,  'homo sapiens using stone tools',
        632,  'a small, tribal village with huts',
        1332, 'at the dawn of civilization, small villages emerged',
        2496, 'ancient egypt, the first massive civiliation',
        1869, 'the height of the roman empire, incredible architecture, by Greg Rutkowski',
        7559, 'medieval town square',
        1265, 'medieval city',
        6628, 'the skyline of New York city',
```

What you can do with this today

Sanjiban + AI art (created with MidJourney)

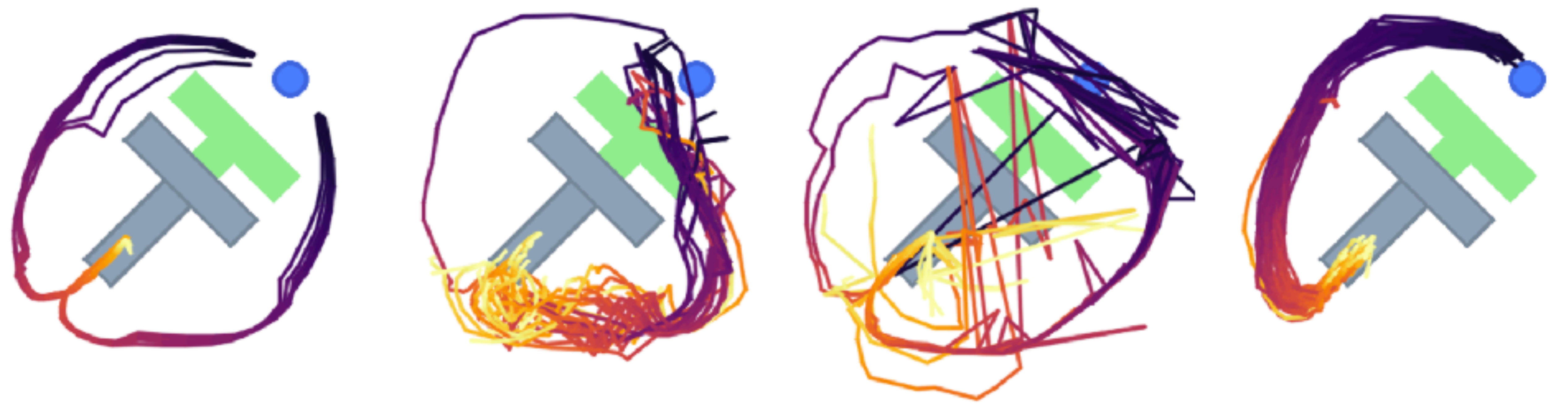
Art is cute ...

but what about robots??

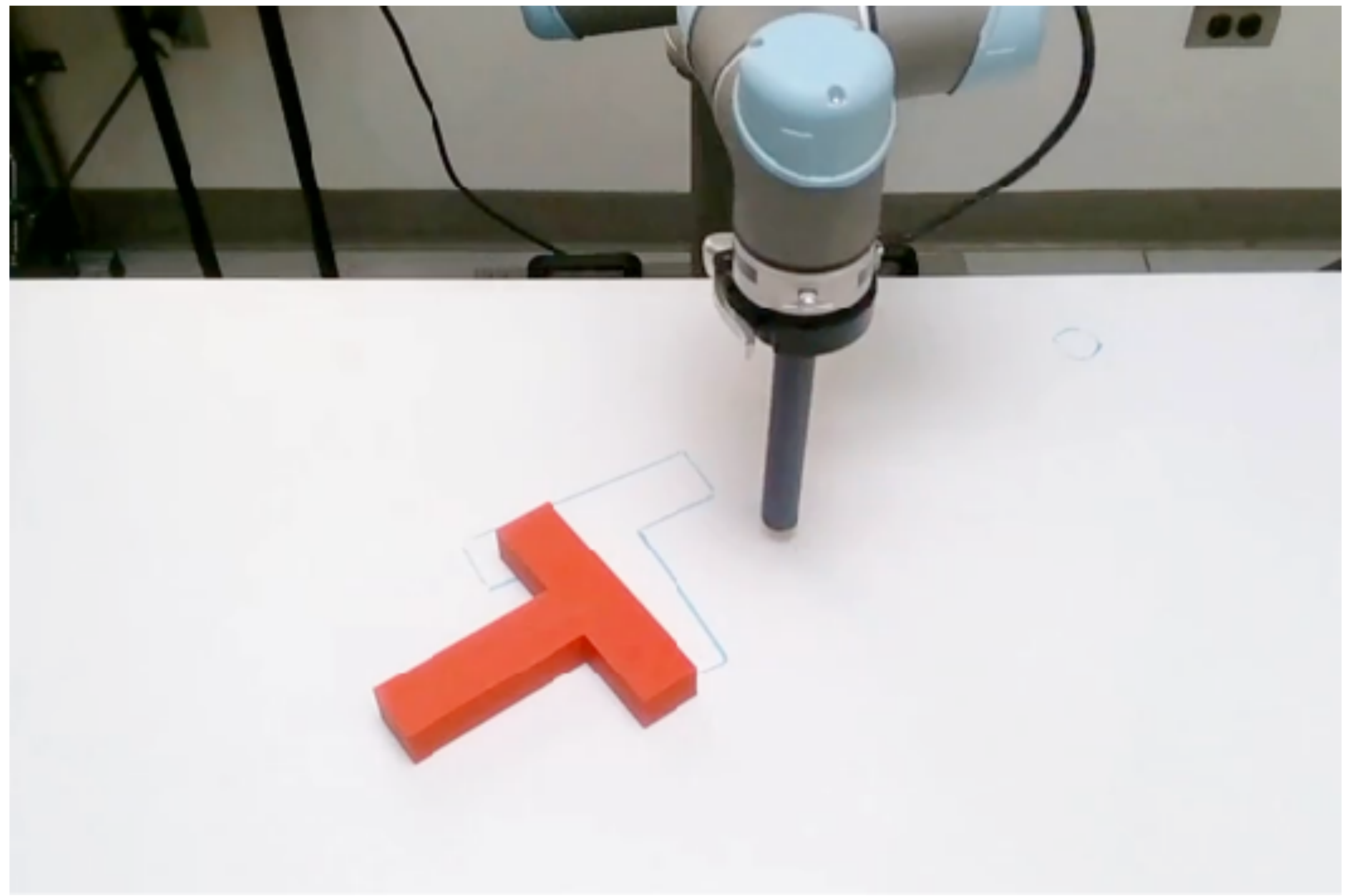
Diffusion Policies!

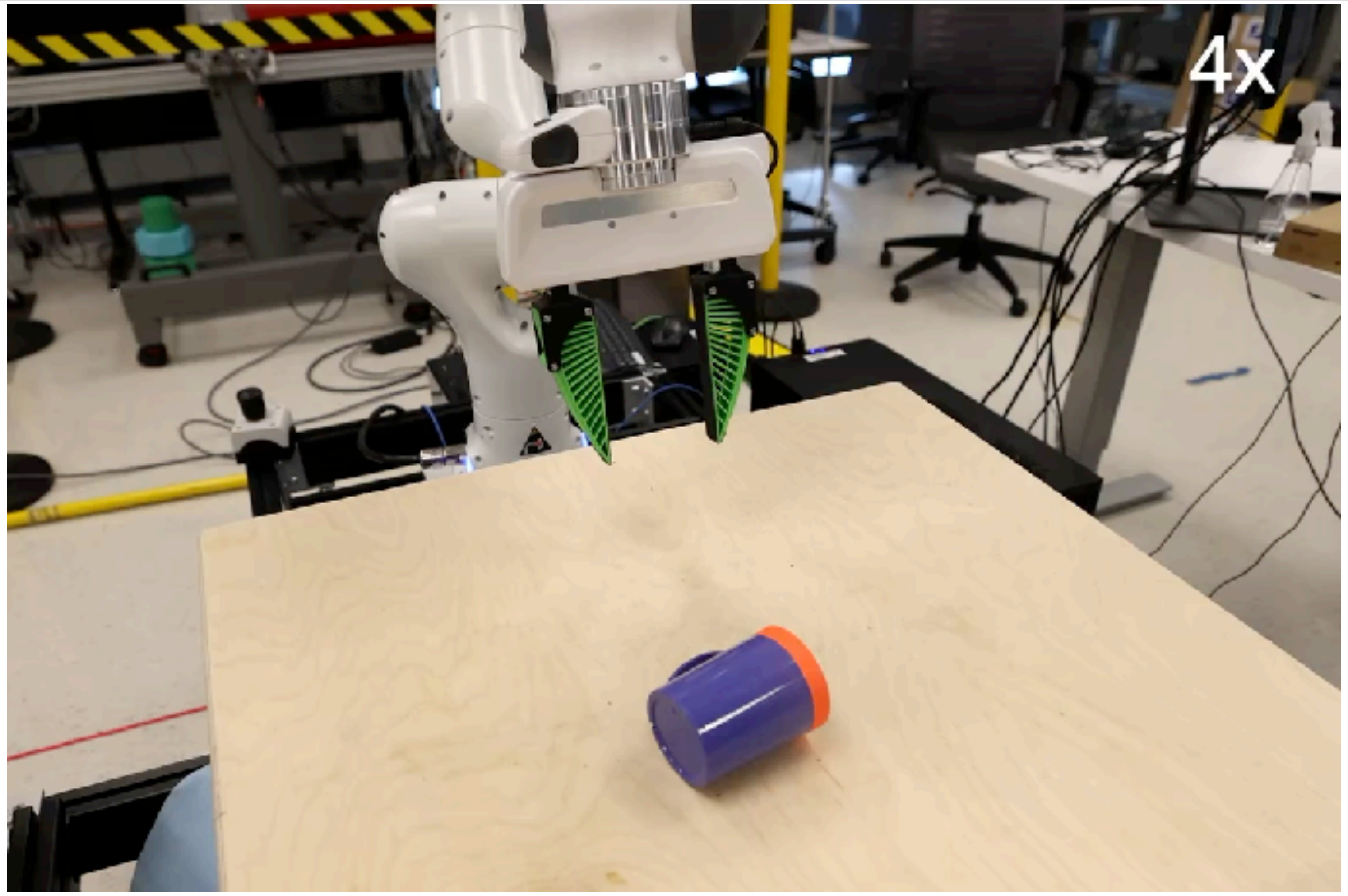
Diffusion Policy:
Visuomotor Policy Learning via Action Diffusion

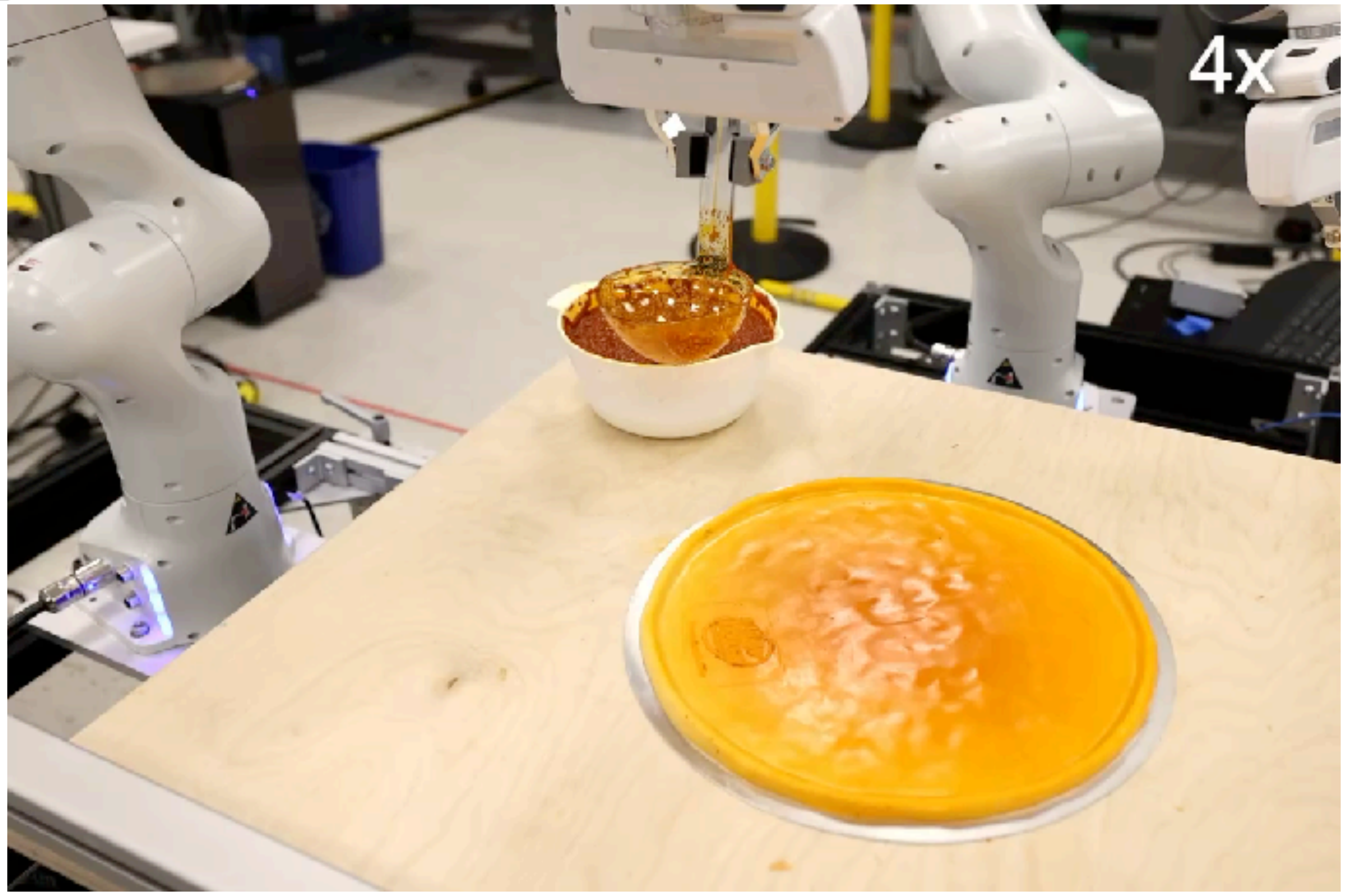
Cheng Chi[1], Siyuan Feng[2], Yilun Du[3], Zhenjia Xu[1], Eric Cousineau[2], Benjamin Burchfiel[2], Shuran Song[1]
[1] Columbia University [2] Toyota Research Institute [3] MIT
https://diffusion-policy.cs.columbia.edu



Diffusion Policy LSTM-GMM BET IBC





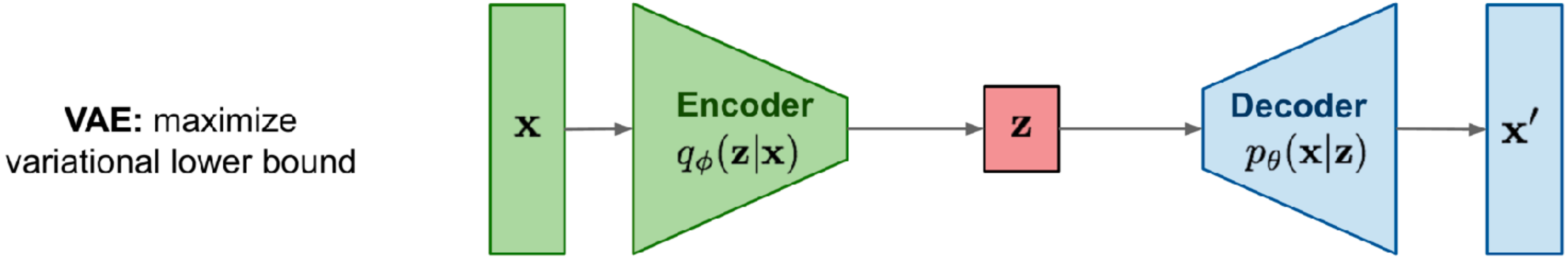


Imitation Learning

Diffusion Models

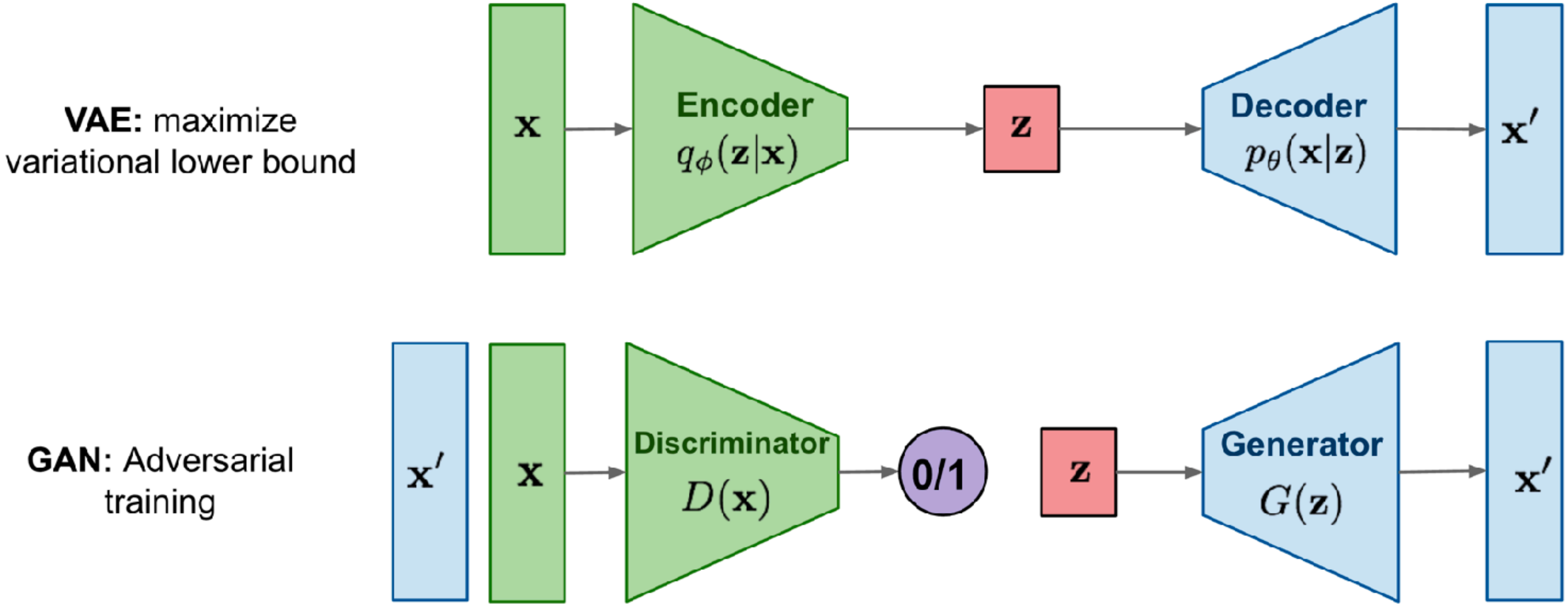
How have researchers in computer vision approached the problem of modeling distributions $p_\theta(x)$?

Option 1: Variational Autoencoder



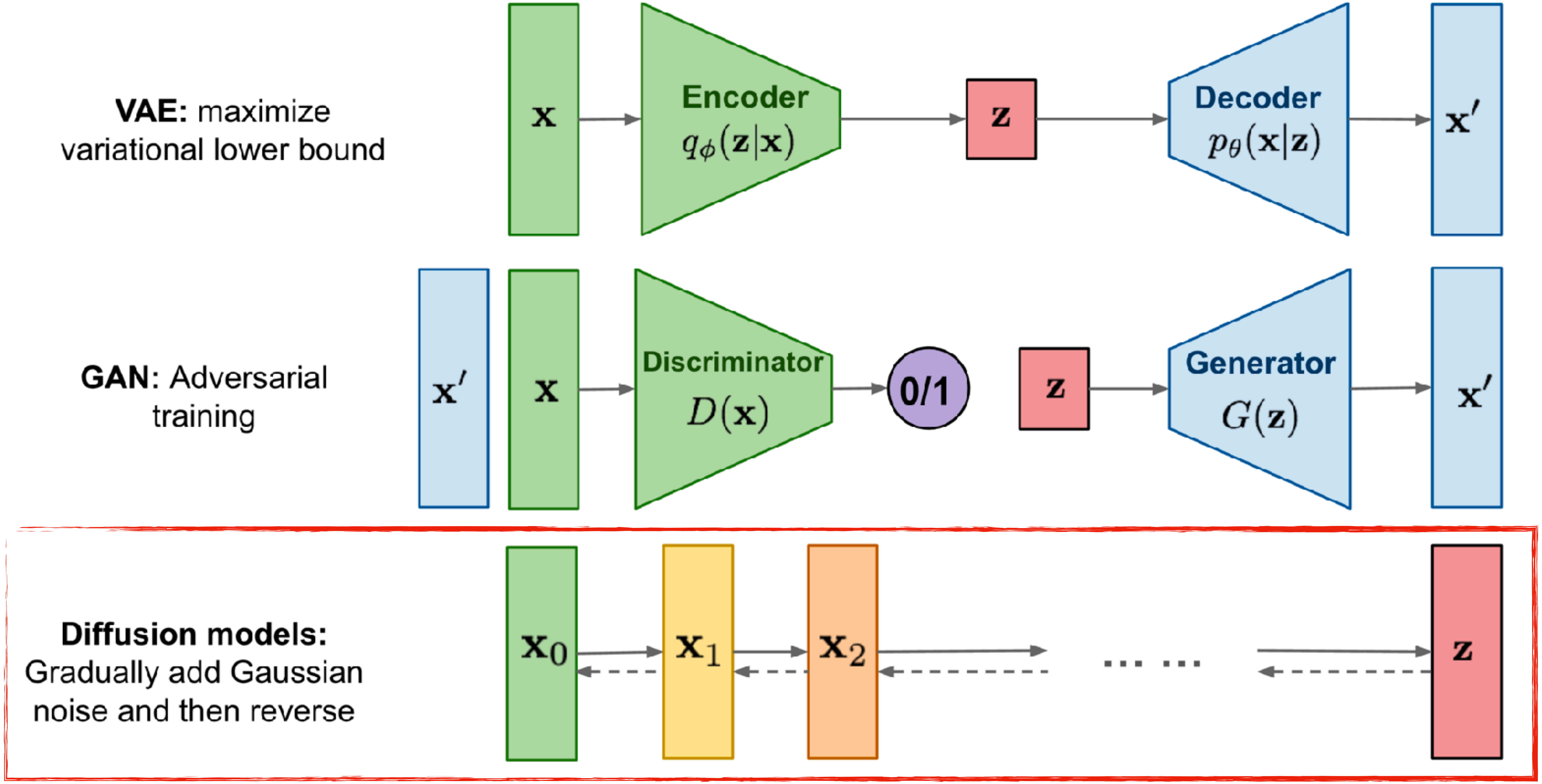
Problem: Images were not very high quality!

Option 2: Generative Adversarial Networks



Problem: Failed to capture multiple modes / diversity

Option 3: Diffusion Models



Problem: Currently too slow (but can be faster!)

History of diffusion models

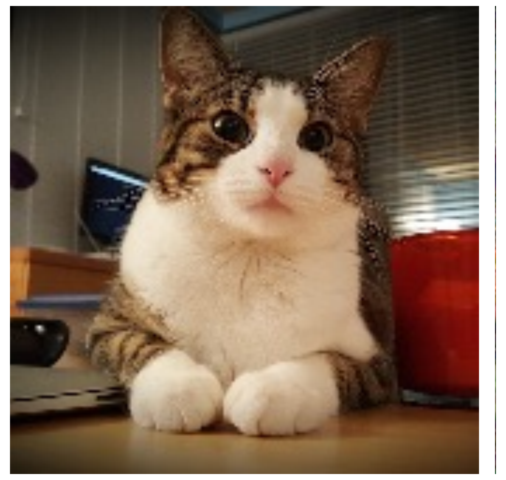
Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015

Song and Ermon, “Generative Modeling by Estimating Gradients of the Data Distribution”, NeurIPS, 2019

Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020

Denoising Diffusion Models

Data

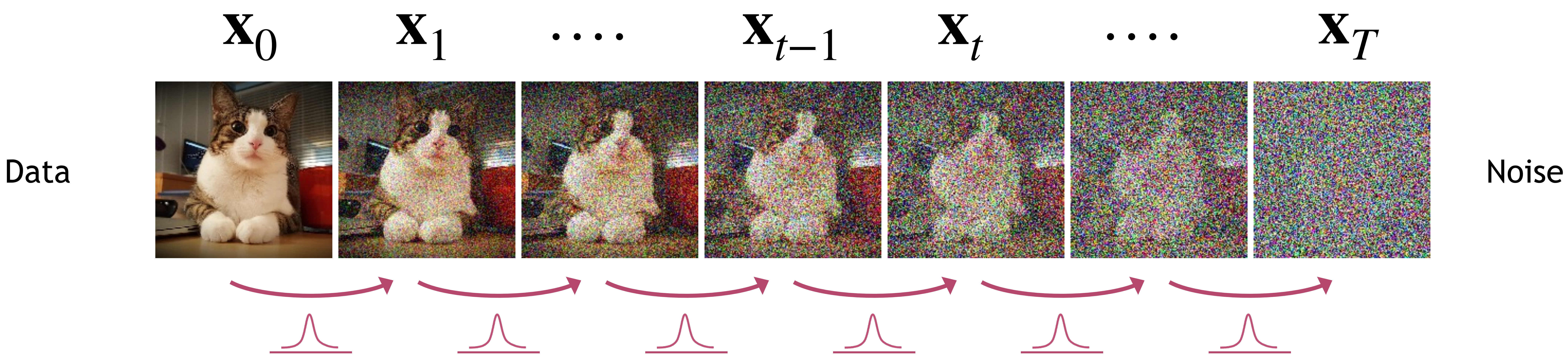


Two main process

Forward diffusion process gradually adds more noise to the input

Reverse diffusion process that learns to generate data by denoising

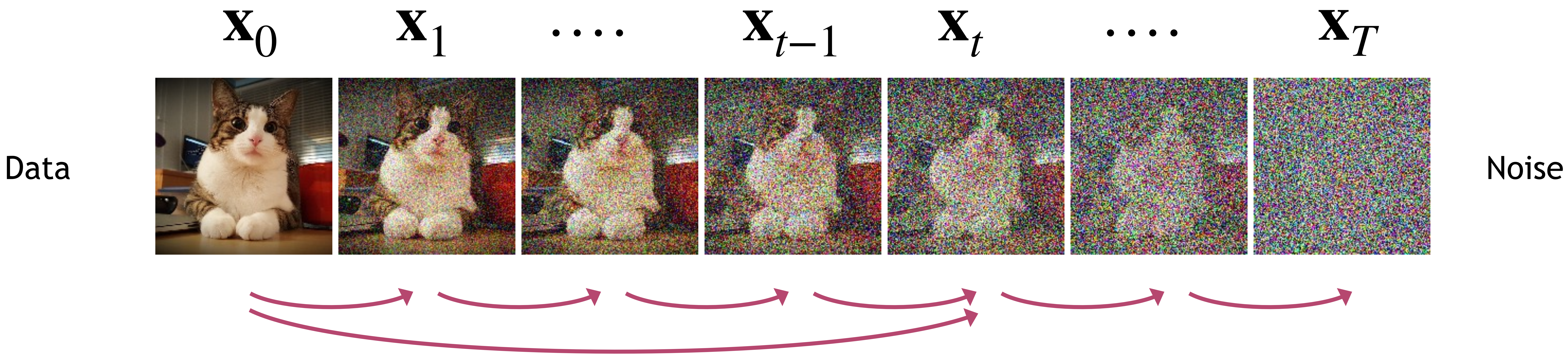
Forward: Add a bit of Gaussian noise



$$q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t \mathbf{I}) \qquad q(\mathbf{x}_{1:T} \,|\, \mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t \,|\, \mathbf{x}_{t-1})$$

Variance schedule $\beta_1 < \beta_2 < \ldots < \beta_T$

Forward: Add a bit of Gaussian noise



Stacking Gaussians gives you a Gaussian

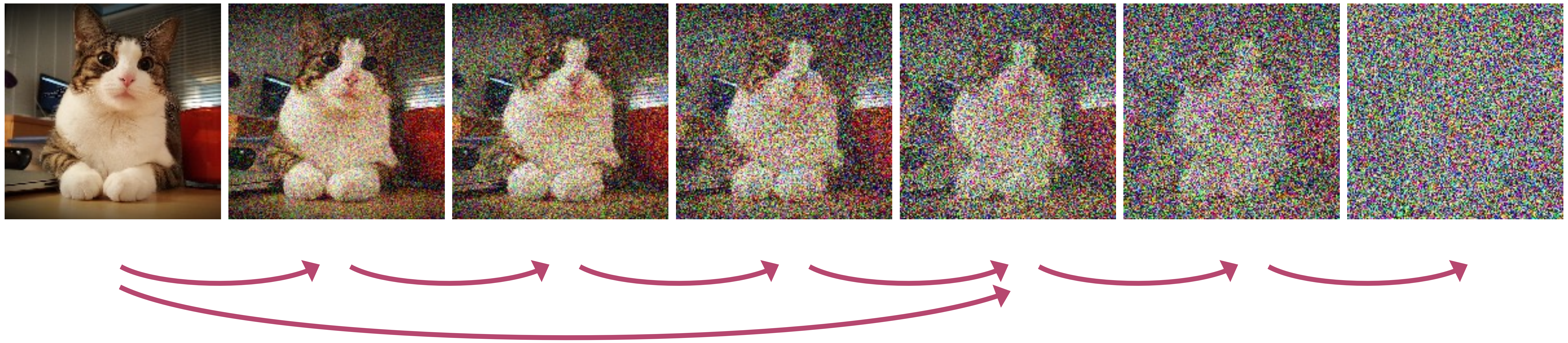
$$\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}$$

where $\alpha_t = 1 - \beta_t \qquad \bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i \qquad \epsilon \sim \mathcal{N}(0,I)$

Forward: Add a bit of Gaussian noise

$\mathbf{x}_0 \quad \mathbf{x}_1 \quad \cdots\cdot \quad \mathbf{x}_{t-1} \quad \mathbf{x}_t \quad \cdots\cdot \quad \mathbf{x}_T$

Data



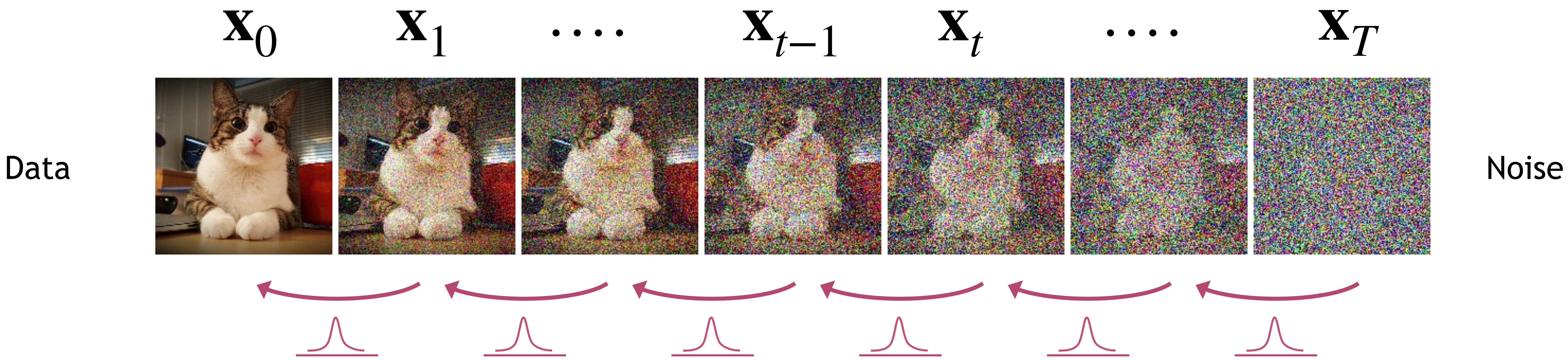
Noise

Directly sample from this!

$$q(\mathbf{x}_t \mid \mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I})$$

Multipliers $\bar{\alpha}_1 > \ldots > \bar{\alpha}_T \approx 0$

Reverse: Learn to denoise

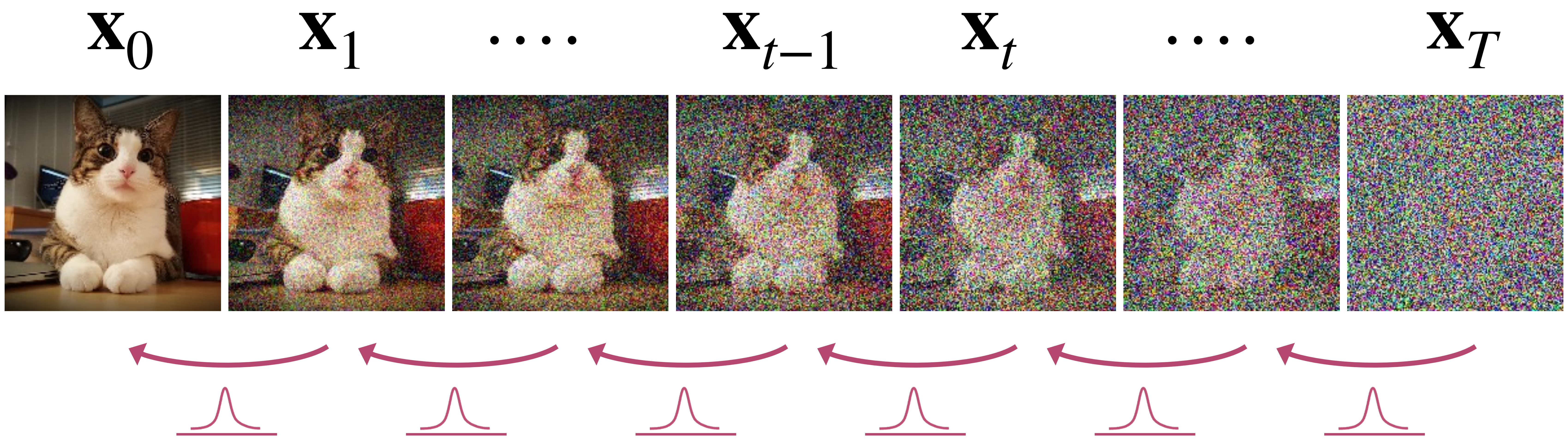


Start from $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

Learn a distribution $p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t)$

$$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) \quad p_\theta(\mathbf{x}_{t-1} | \mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

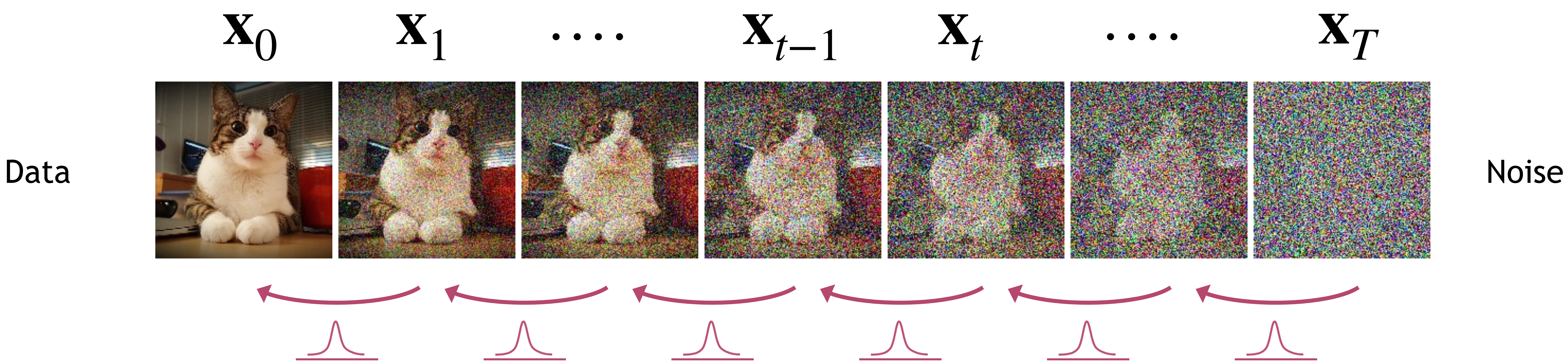
How do we train our model?



Write out the variational upper bound and minimize that

$$L_{\mathrm{VUB}} = \mathbb{E}_{q(\mathbf{x}_{0:T})}\left[\log \frac{q(\mathbf{x}_{1:T} \mid \mathbf{x}_0)}{p_\theta(\mathbf{x}_{0:T})}\right]$$

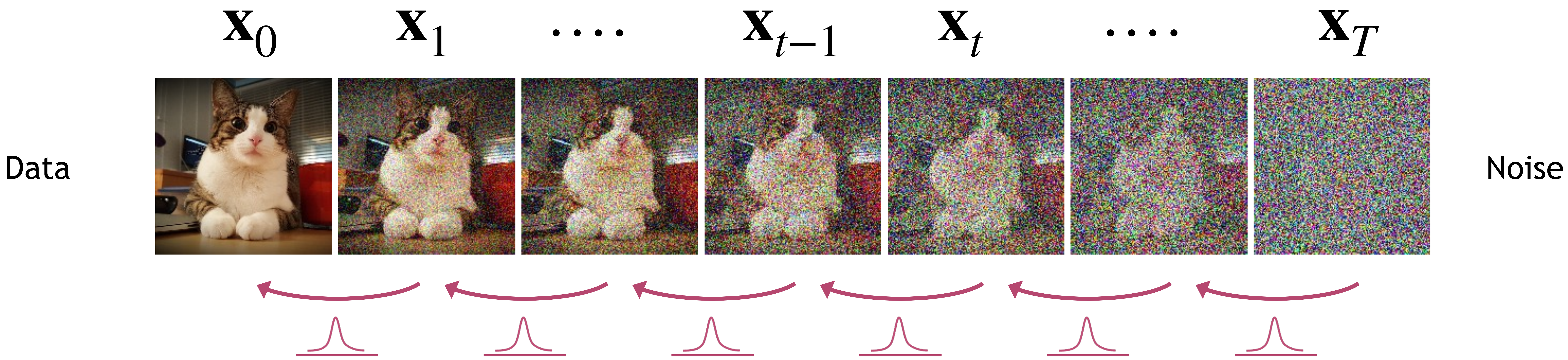
How do we train our model?



Simplifying trick: Don't predict the images, predict the noise ϵ_θ

$$\boldsymbol{\mu}_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\right)$$

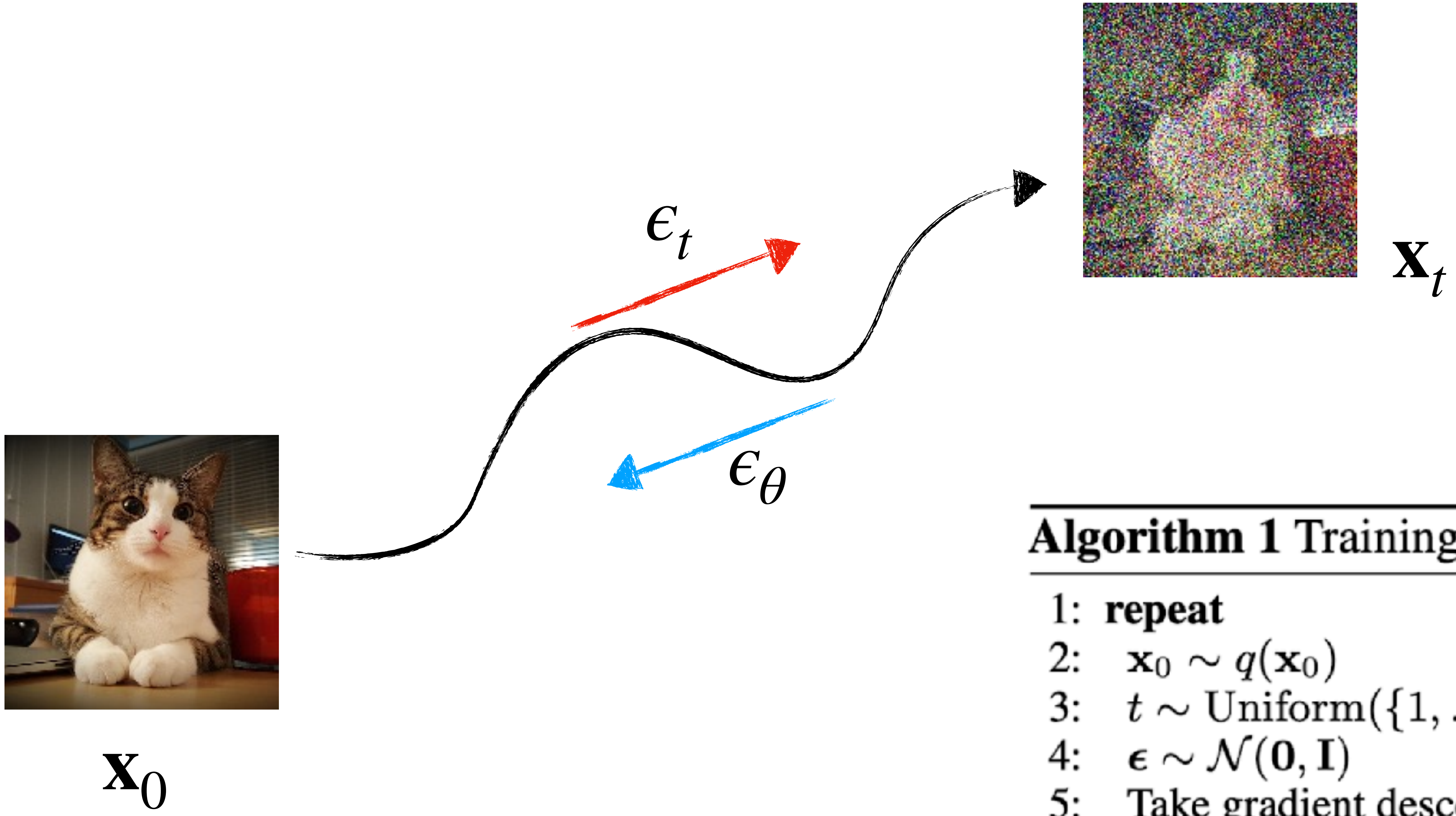
How do we train our model?



Crunch through a ton of math, simplify terms and you get

$$
\begin{aligned}
L_t^{\text{simple}} &= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)\|^2\Big] \\
&= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\boldsymbol{\epsilon}_t}\Big[\|\boldsymbol{\epsilon}_t - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}_t, t)\|^2\Big]
\end{aligned}
$$

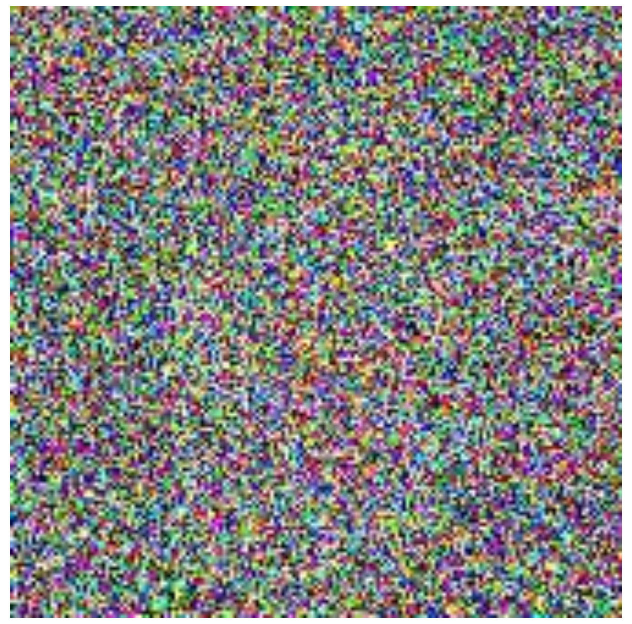
Super simple training loop!



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$
6: **until** converged

Super simple inference step!



$\mathbf{x}_T$

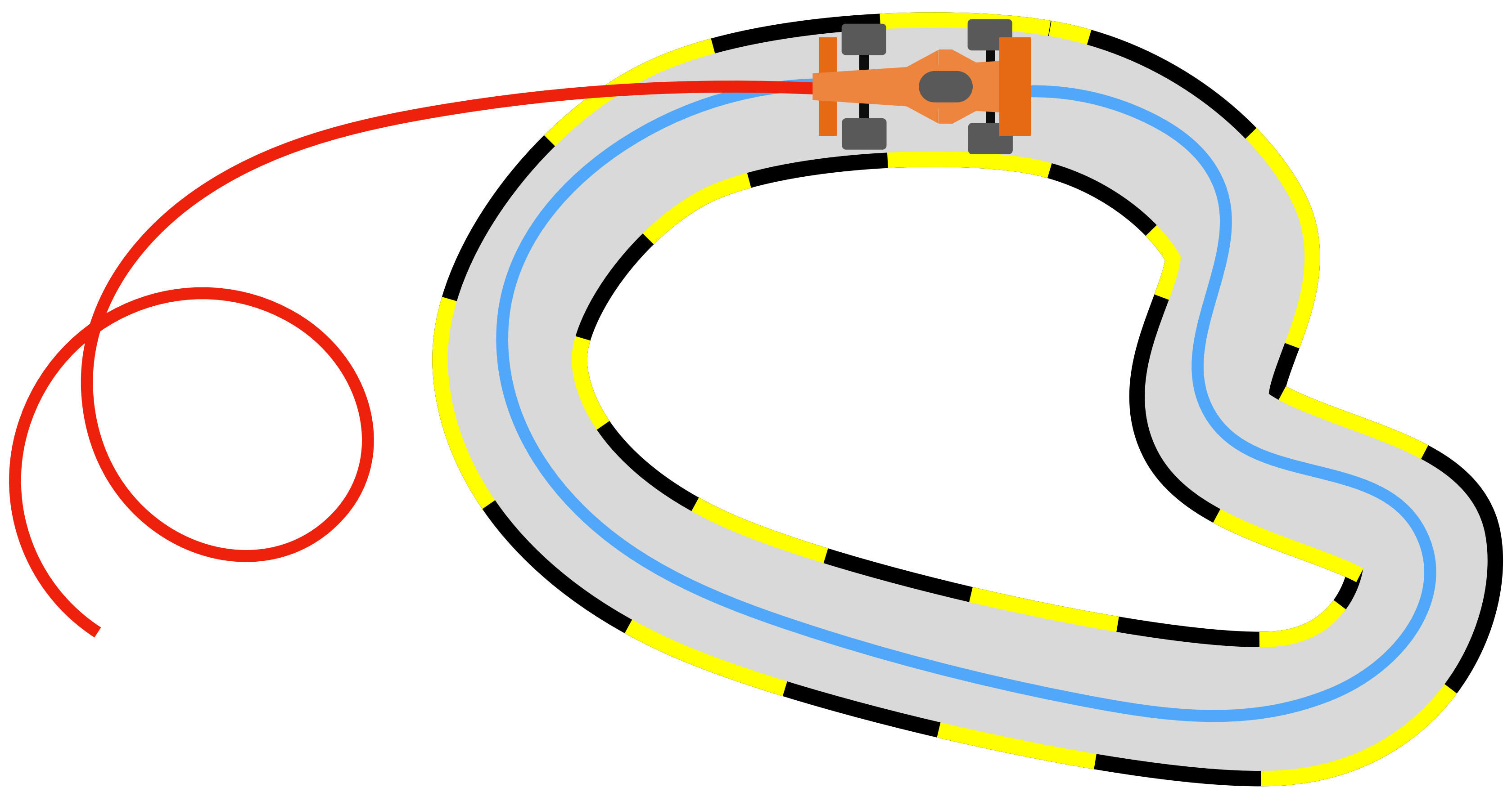
$\mathbf{x}_{T-1}$

$\mathbf{x}_0$

Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \sigma_t \mathbf{z}$
5: **end for**
6: **return** $\mathbf{x}_0$

Wait just a minute ...
Isn't there a distribution
shift issue here?



Expert distribution

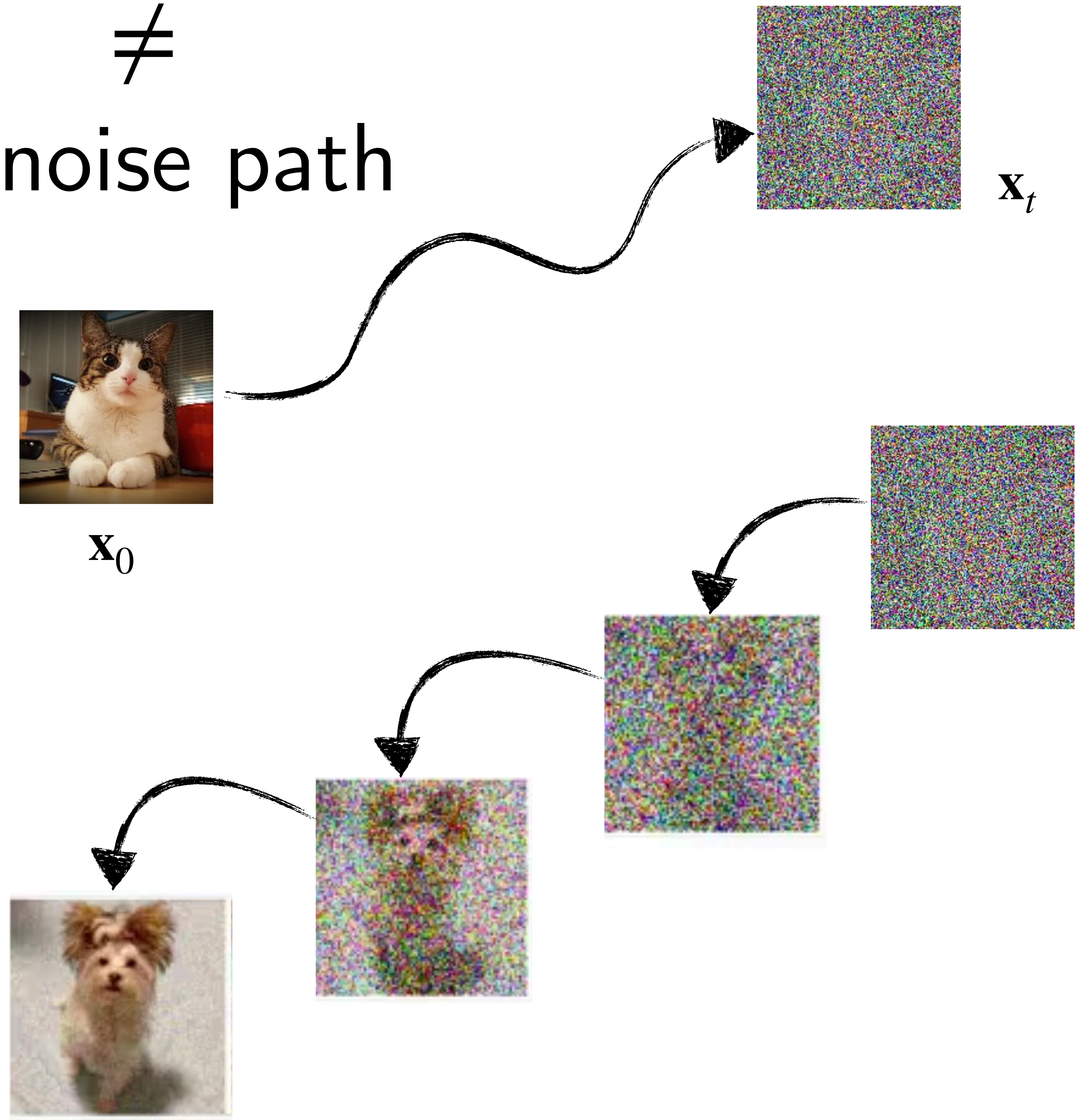
$\neq$

Learner distribution

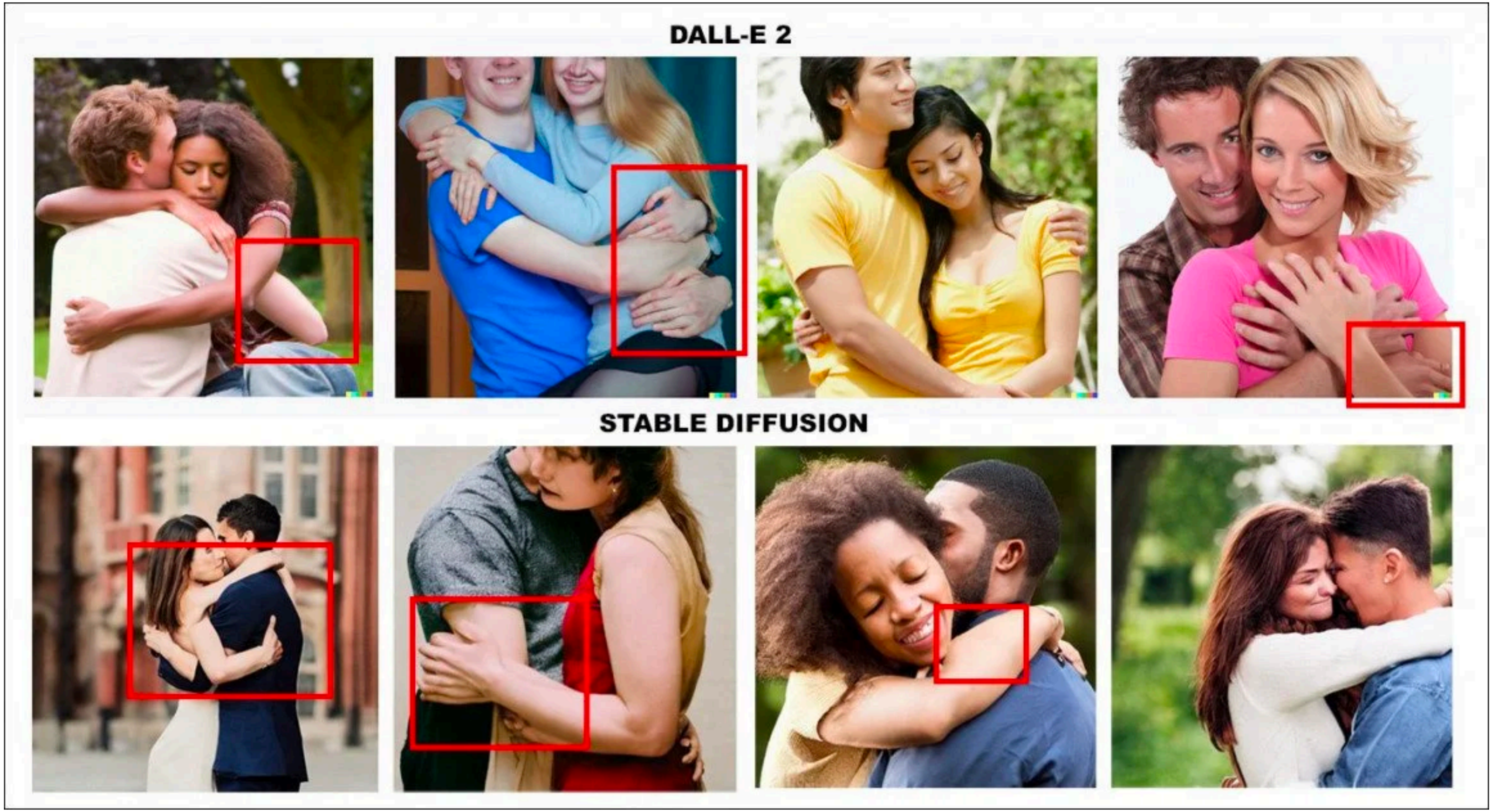
Noise path

$\neq$

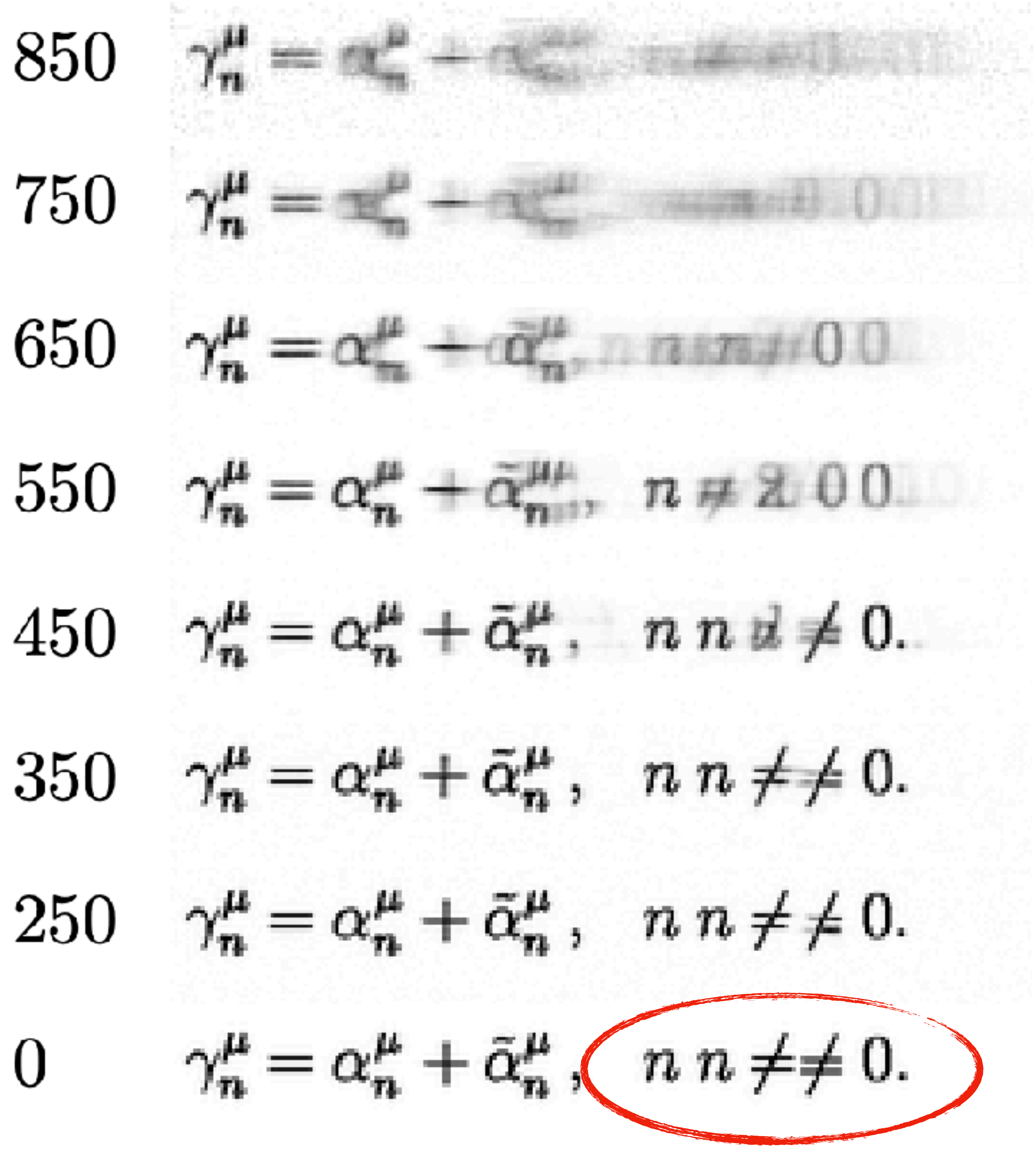
Denoise path



Unrealistic images produced by distribution shift!



Dire consequences when mistakes matter!



What do we do about this?

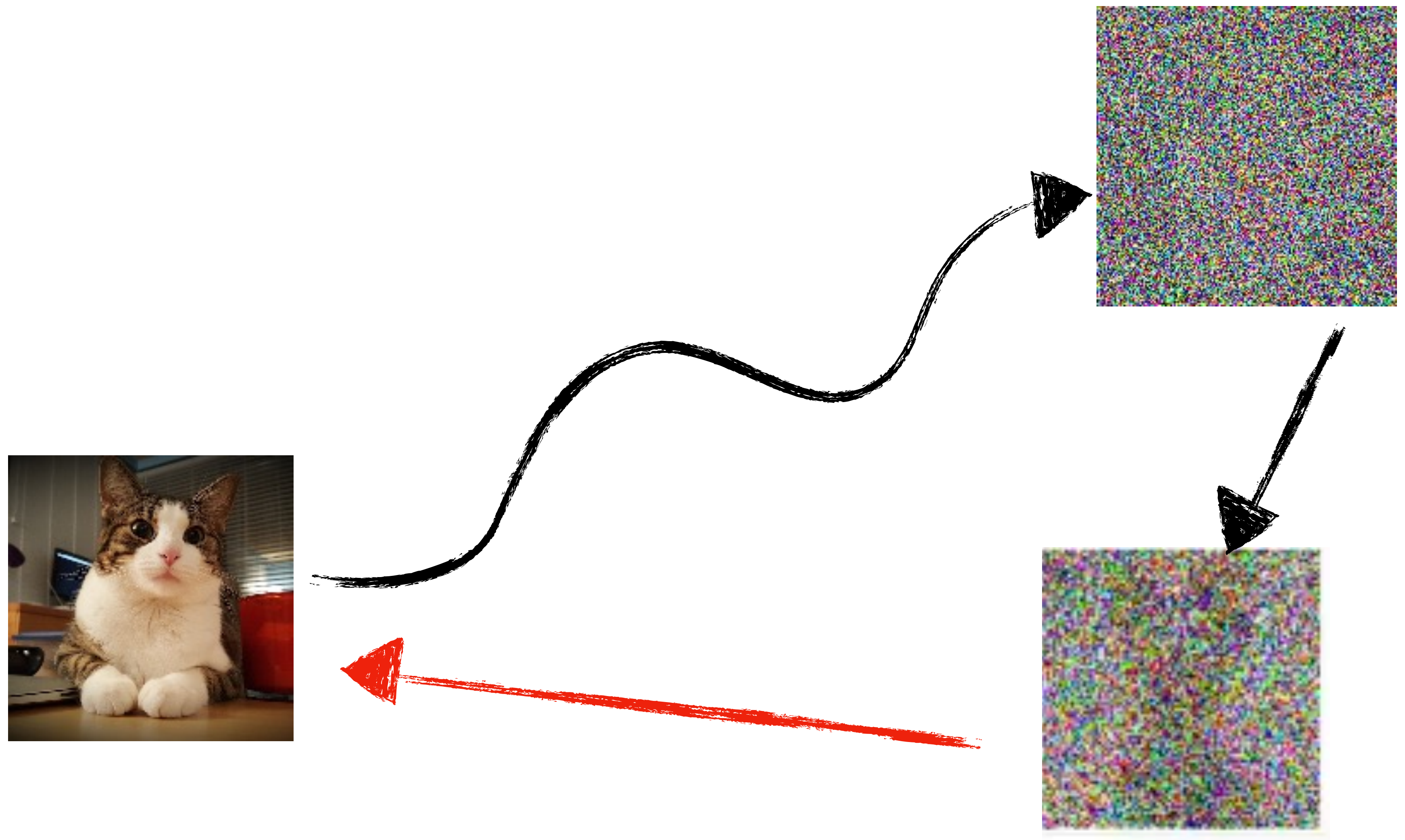
Clean the data to remove any unrealistic images that may confuse it?

Take small steps so as to not leave the realistic manifold?

There must be a better way!

Can DAgger once again save the day?

DAgger for Diffusion



For i = 0 … N

Noise the input image

Denoise m steps using current learner

Add original image as target

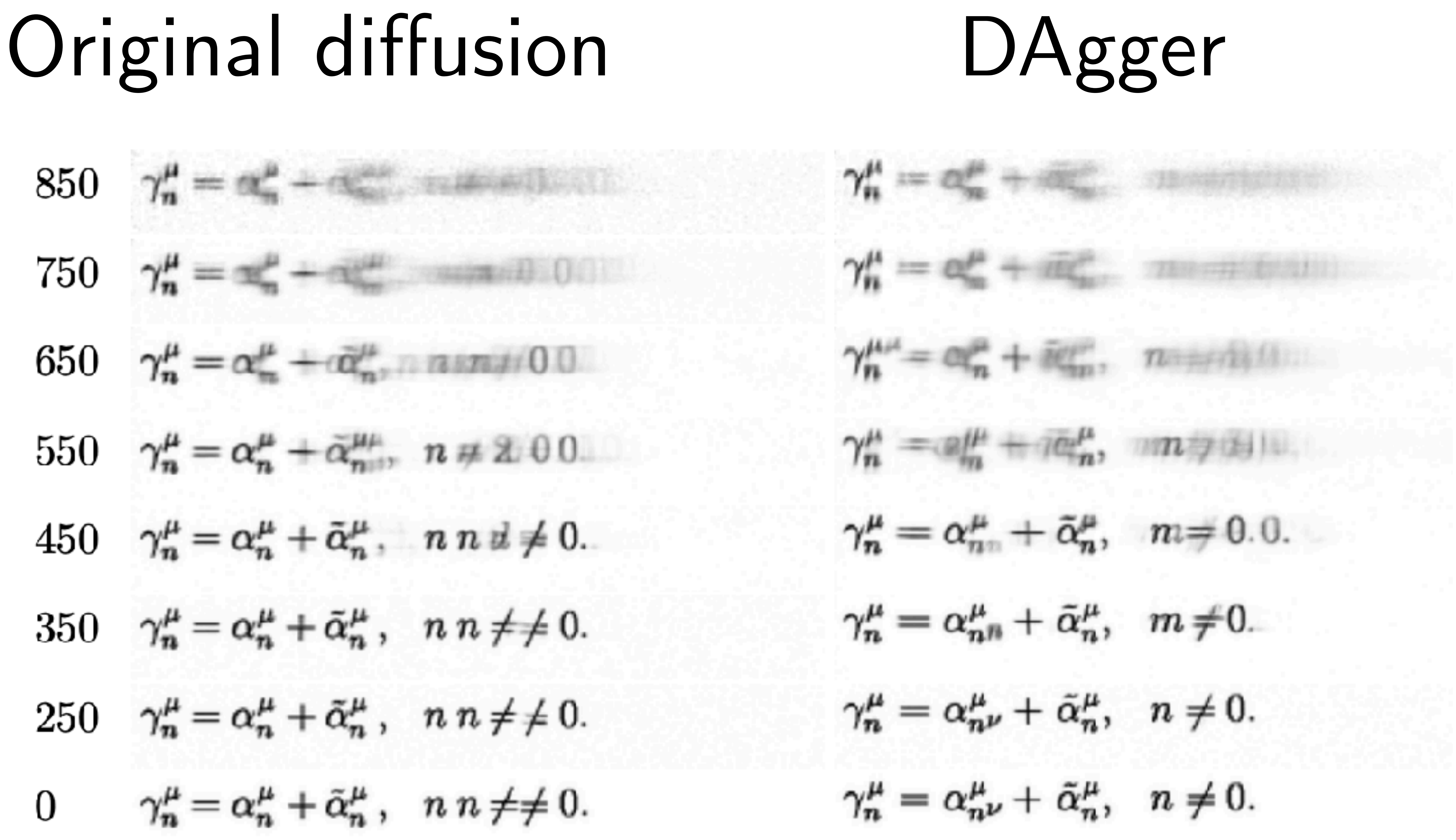
Aggregate data!

MARKUP-TO-IMAGE DIFFUSION MODELS WITH SCHEDULED SAMPLING

Yuntian Deng[1], Noriyuki Kojima[2], Alexander M. Rush[2]

[1] Harvard University dengyuntian@seas.harvard.edu
[2] Cornell University {nk654,arush}@cornell.edu

Does it work?



Is there another way?

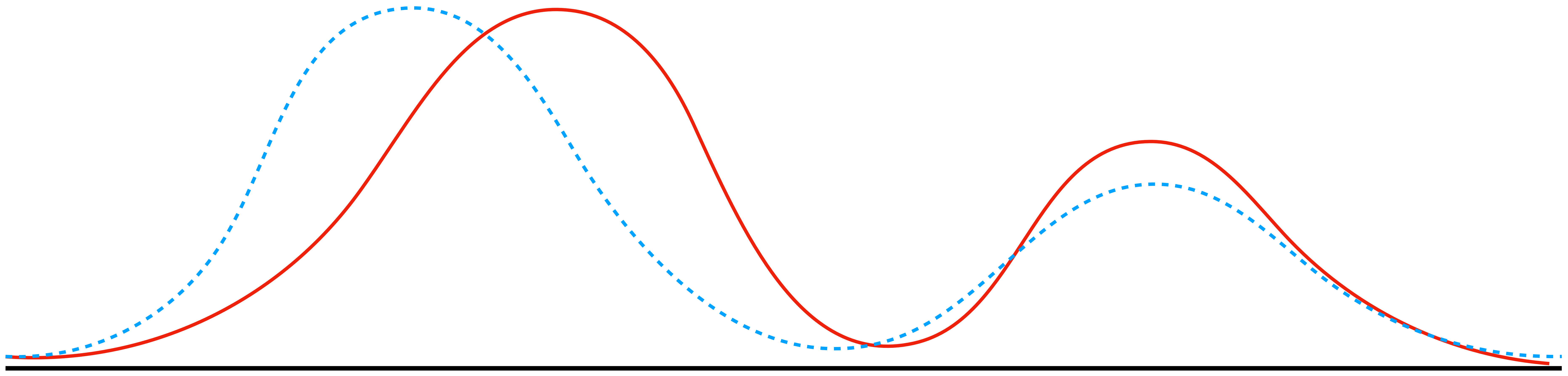
DAgger feels like overkill!

We control both the noise and the denoise process

Can't we just noise / train more intelligently?

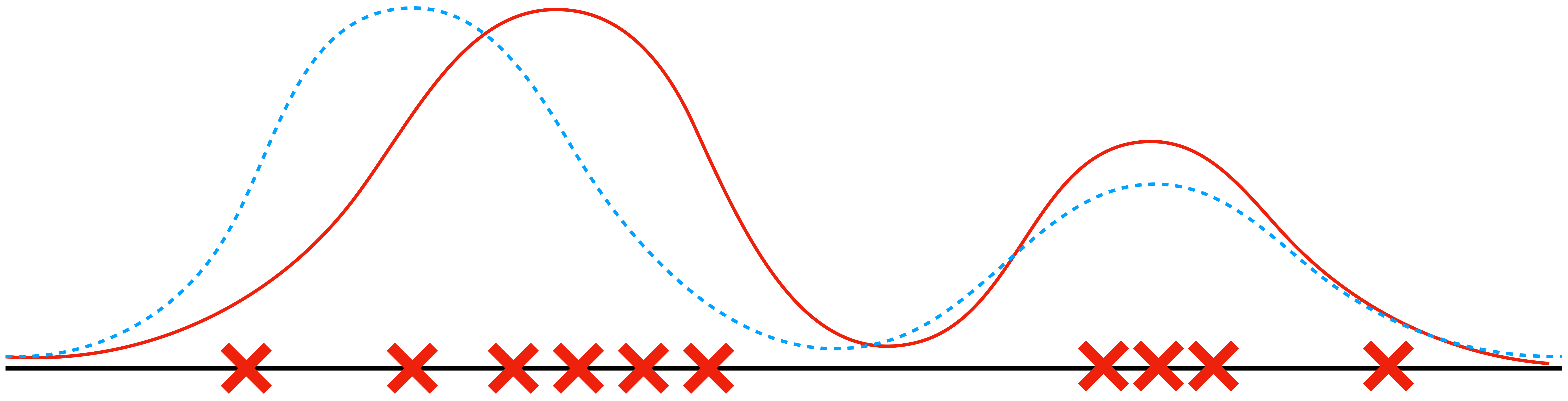
What is our goal?

We want to model a distribution $p_\theta(x) \approx p^*(x)$



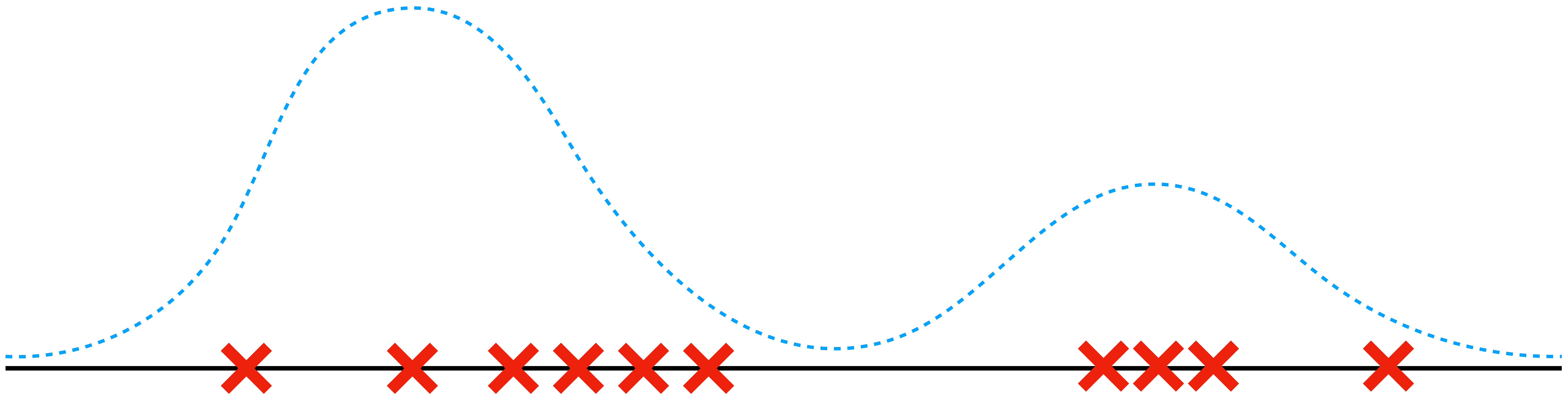
What is our goal?

We only see samples from $x_1, x_2, \ldots x_N \sim p^*(x)$



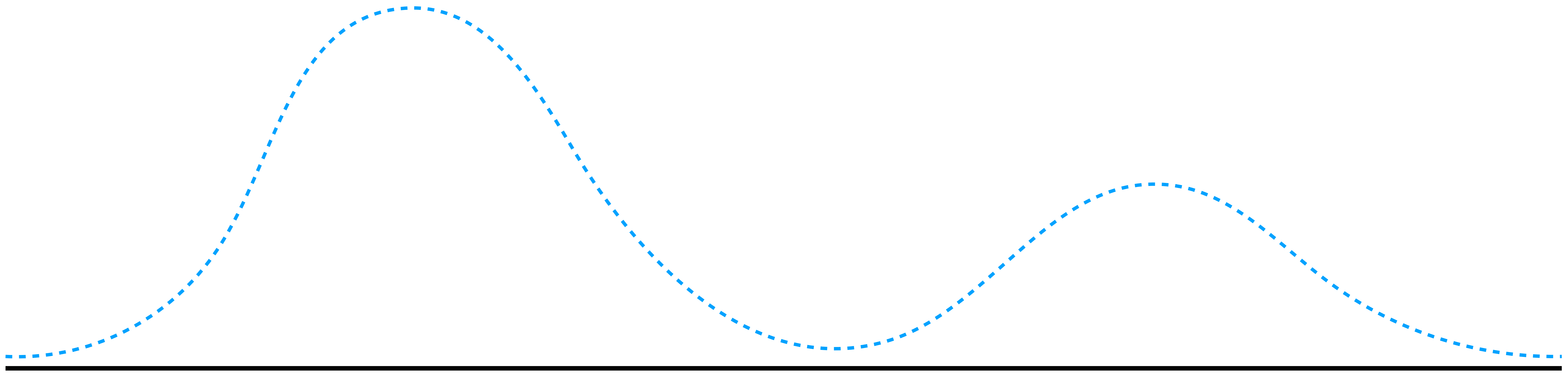
How do we do this?

Typically choose $p_\theta(x) = \dfrac{\exp(-f_\theta(x))}{Z(\theta)}$ (Recall MaxEnt!)



How do we do this?

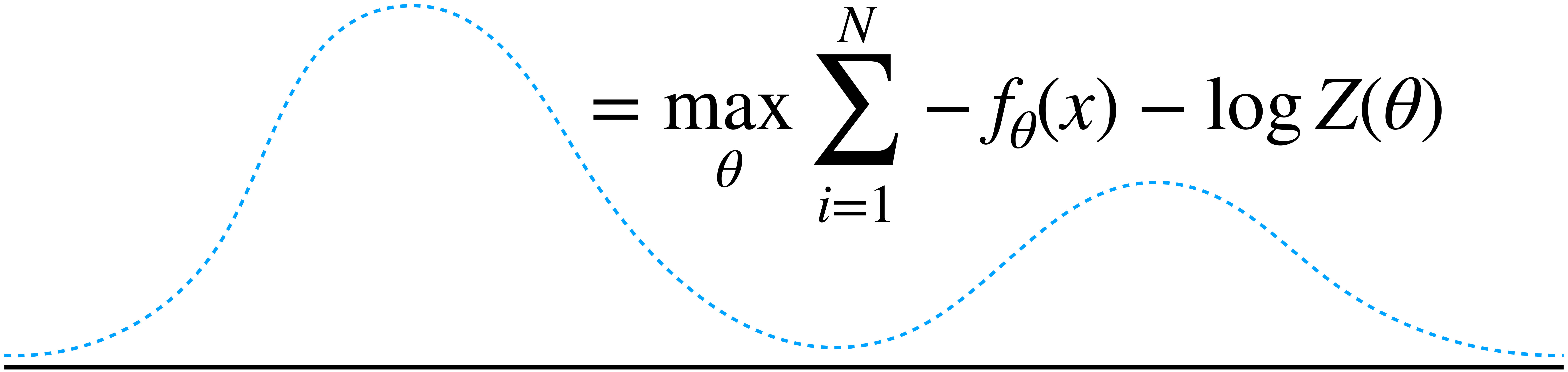
$$\max_{\theta} \sum_{i=1}^{N} \log p_{\theta}(x_i)$$



How do we do this?

$$\max_{\theta} \sum_{i=1}^{N} \log p_{\theta}(x_i) = \max_{\theta} \sum_{i=1}^{N} \log \frac{\exp(-f_{\theta}(x))}{Z(\theta)}$$

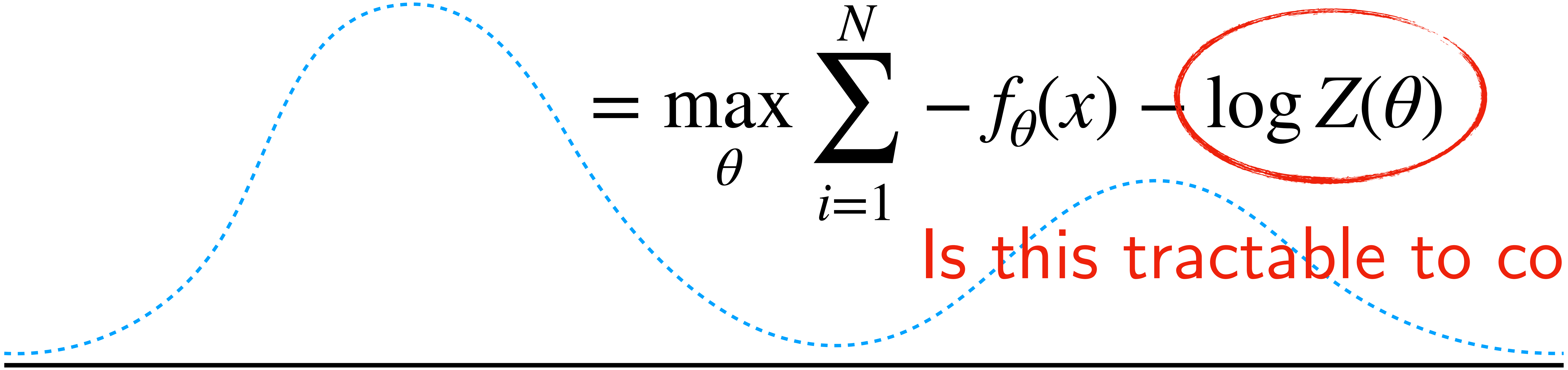
$$= \max_{\theta} \sum_{i=1}^{N} -f_{\theta}(x) - \log Z(\theta)$$



How do we do this?

$$\max_\theta \sum_{i=1}^{N} \log p_\theta(x_i) \qquad = \max_\theta \sum_{i=1}^{N} \log \frac{\exp(-f_\theta(x))}{Z(\theta)}$$

$$= \max_\theta \sum_{i=1}^{N} -f_\theta(x) - \log Z(\theta)$$



Is this tractable to compute?

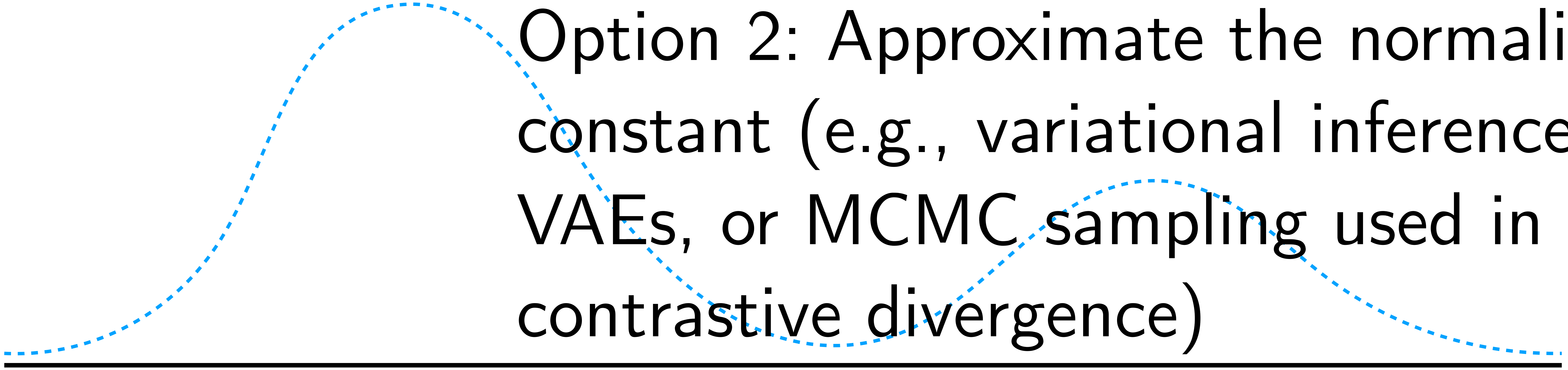
How do we do this?

$$\max_{\theta} \sum_{i=1}^{N} -f_{\theta}(x) - \log Z(\theta)$$

Is this tractable to compute?

Option 1: Restrict the model architecture

Option 2: Approximate the normalizing constant (e.g., variational inference in VAEs, or MCMC sampling used in contrastive divergence)



Is there another way to bypass the normalizing constant?

Yes!

Model the score function

$$\nabla_x \log p_\theta(x)$$

Score-based model

Why?

$$s_\theta(x) = \nabla_x \log p_\theta(x)$$

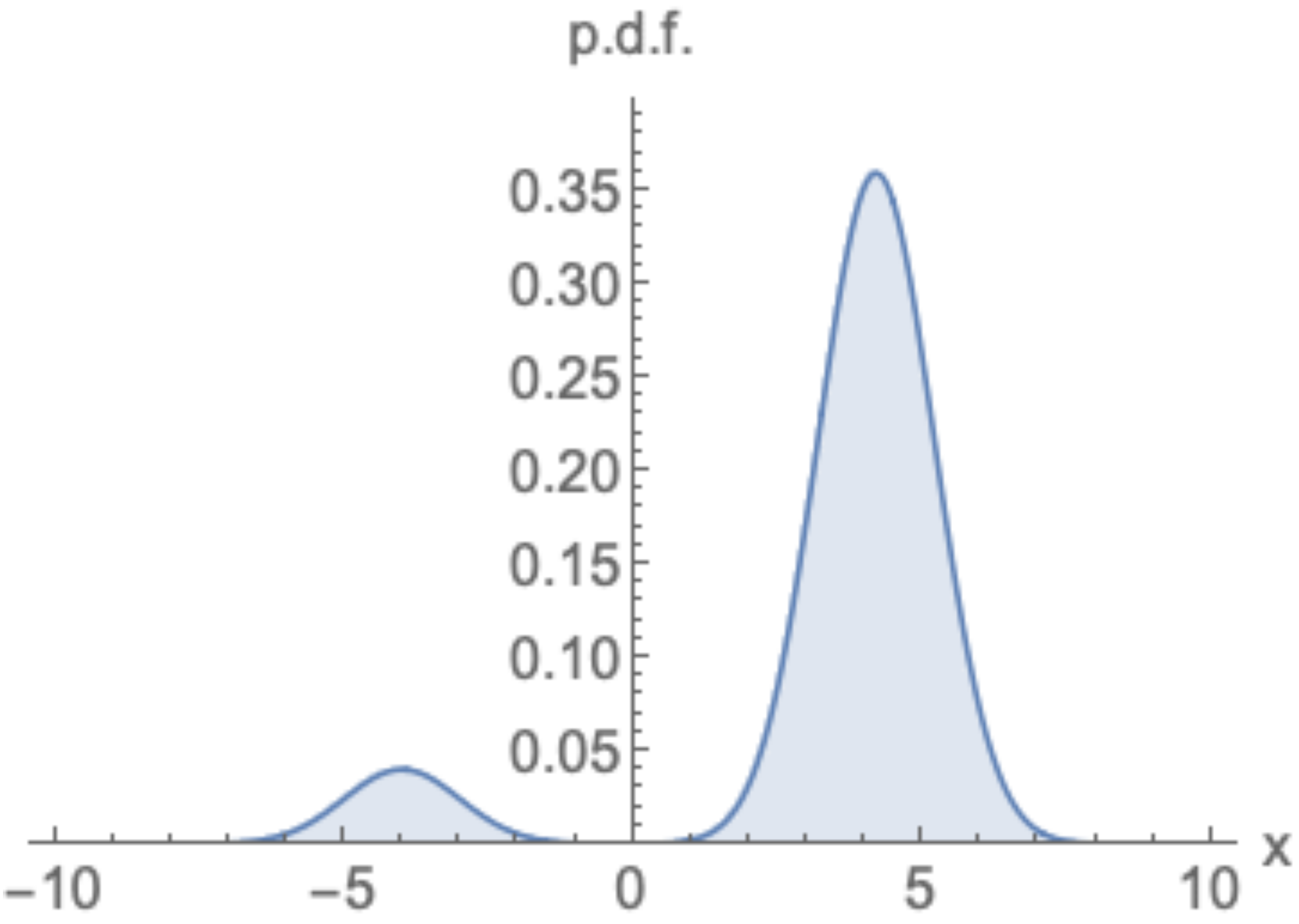
$$= \nabla_x - f_\theta(x) - \nabla_x \log Z(\theta)$$

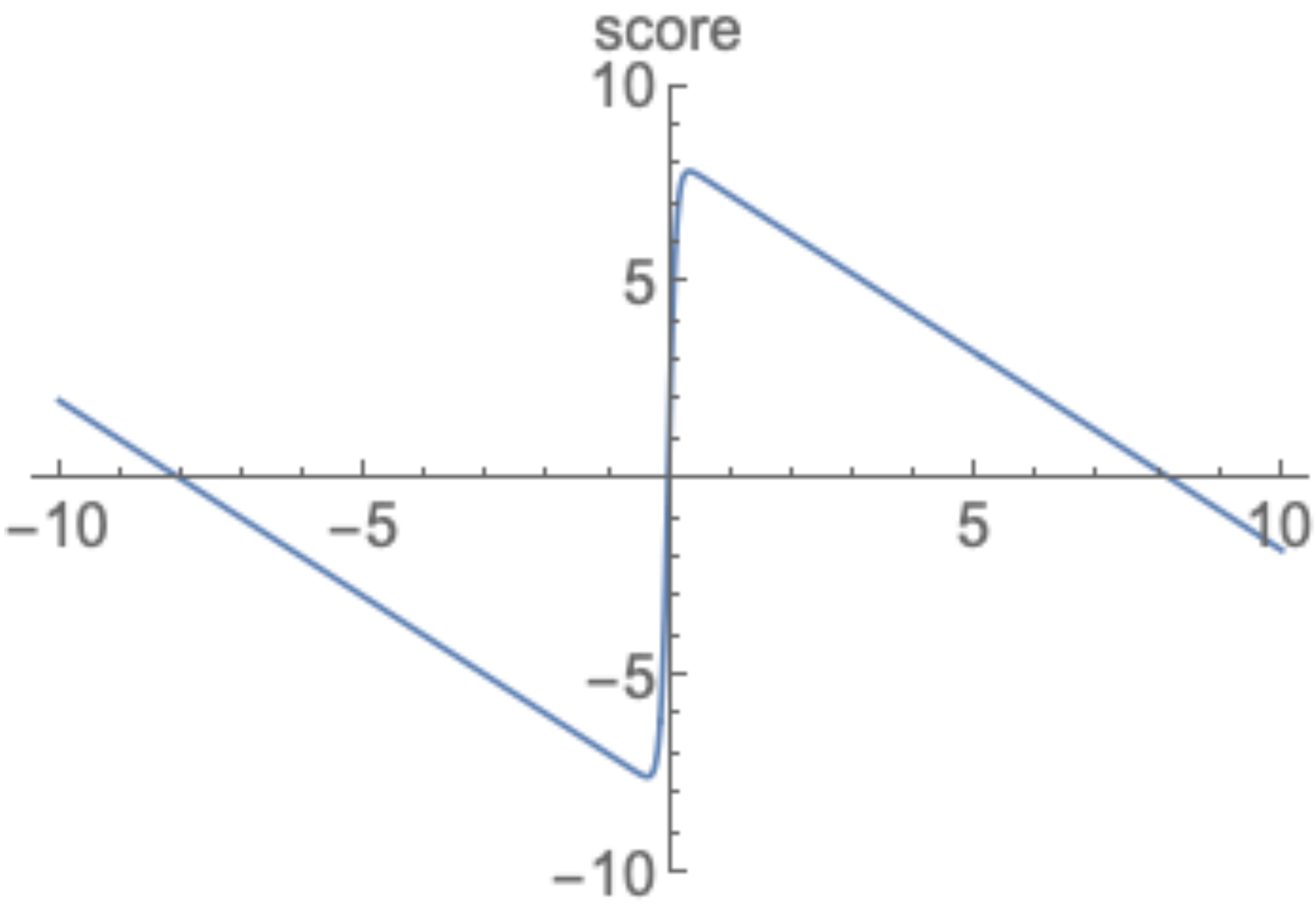
$(= 0!)$

$$= \nabla_x - f_\theta(x)$$

Bye bye normalizing constant!!!

Probability vs Score





Probability functions $p_\theta(x)$ need to be normalized!

Score function $s_\theta(x)$ No normalization!

So … how do we learn the score?

Consider the following optimization problem

$$\frac{1}{2}\mathbb{E}_{p^*}\left[|| \nabla_x \log p^*(x) - \nabla_x \log p_\theta(x) ||_2^2 \right]$$

Consider the following optimization problem

$$\frac{1}{2}\mathbb{E}_{p^*}\left[|| \nabla_x \log p^*(x) - \nabla_x \log p_\theta(x) ||_2^2 \right]$$

The Bad:

We don't know this!
(Can only sample
from p*)

The Good:

No normalizing
term here!

Math to the rescue!

Can prove that $\frac{1}{2}\mathbb{E}_{p^*}\left[||\nabla_x \log p^*(x) - \nabla_x \log p_\theta(x)||\right]$

is the same as

$$\mathbb{E}_{p^*}\left[\mathrm{tr}(\nabla_x^2 \log p_\theta(x)) + \frac{1}{2}||\nabla_x \log p_\theta(x)||_2^2\right] + \text{const}$$

We can compute all these terms!

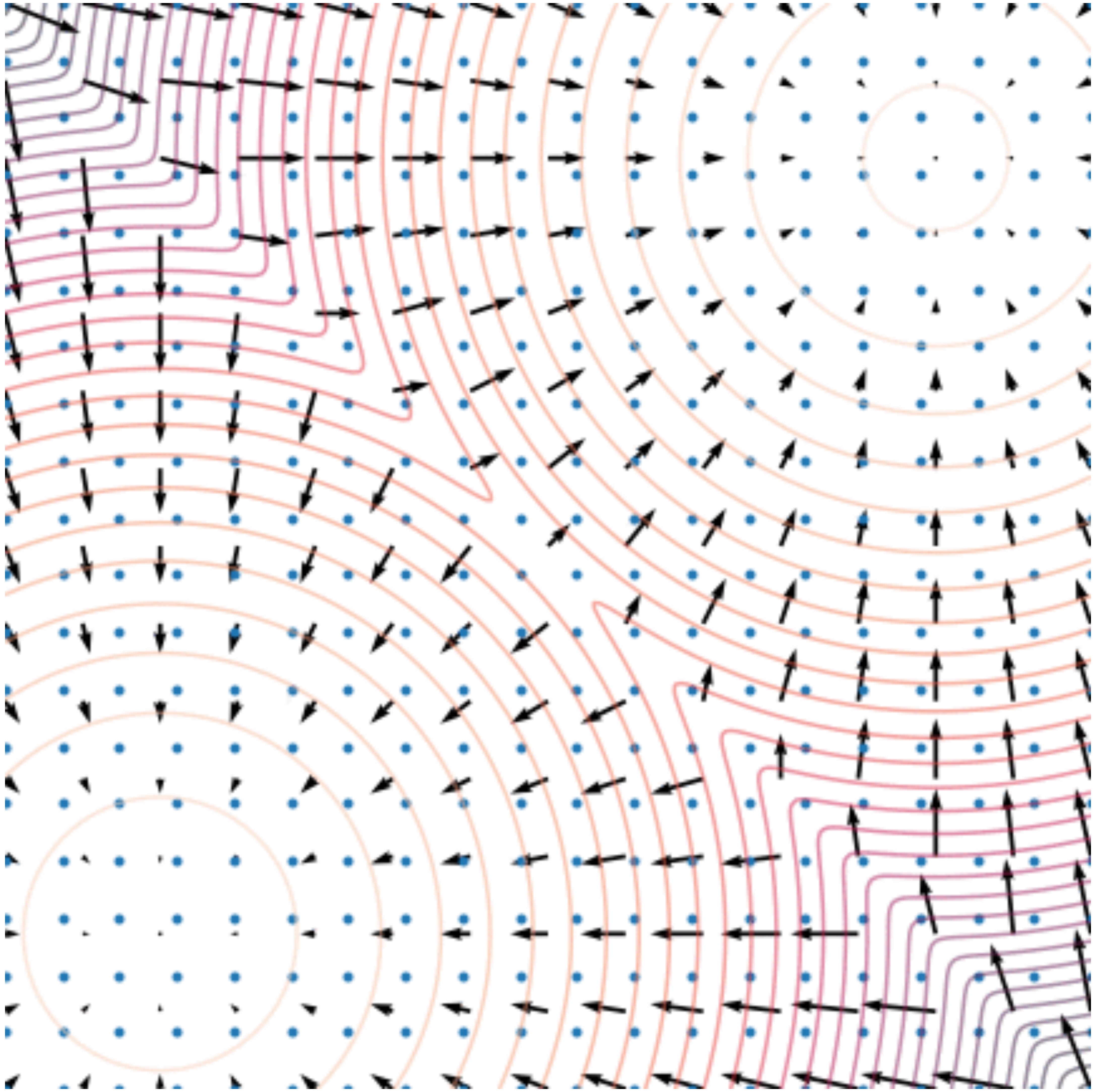
Training score based models

$$\mathscr{L}(\theta) = \mathbb{E}_{p*}\left[\mathrm{tr}(\nabla_x^2 s_\theta(x)) + \frac{1}{2}||\nabla_x s_\theta(x)||_2^2\right]$$

Optimize this loss to get score $s_\theta(x)$

How can I sample from $p_\theta(x)$ using $\nabla_x \log p_\theta(x)$?

Langevin Dynamics



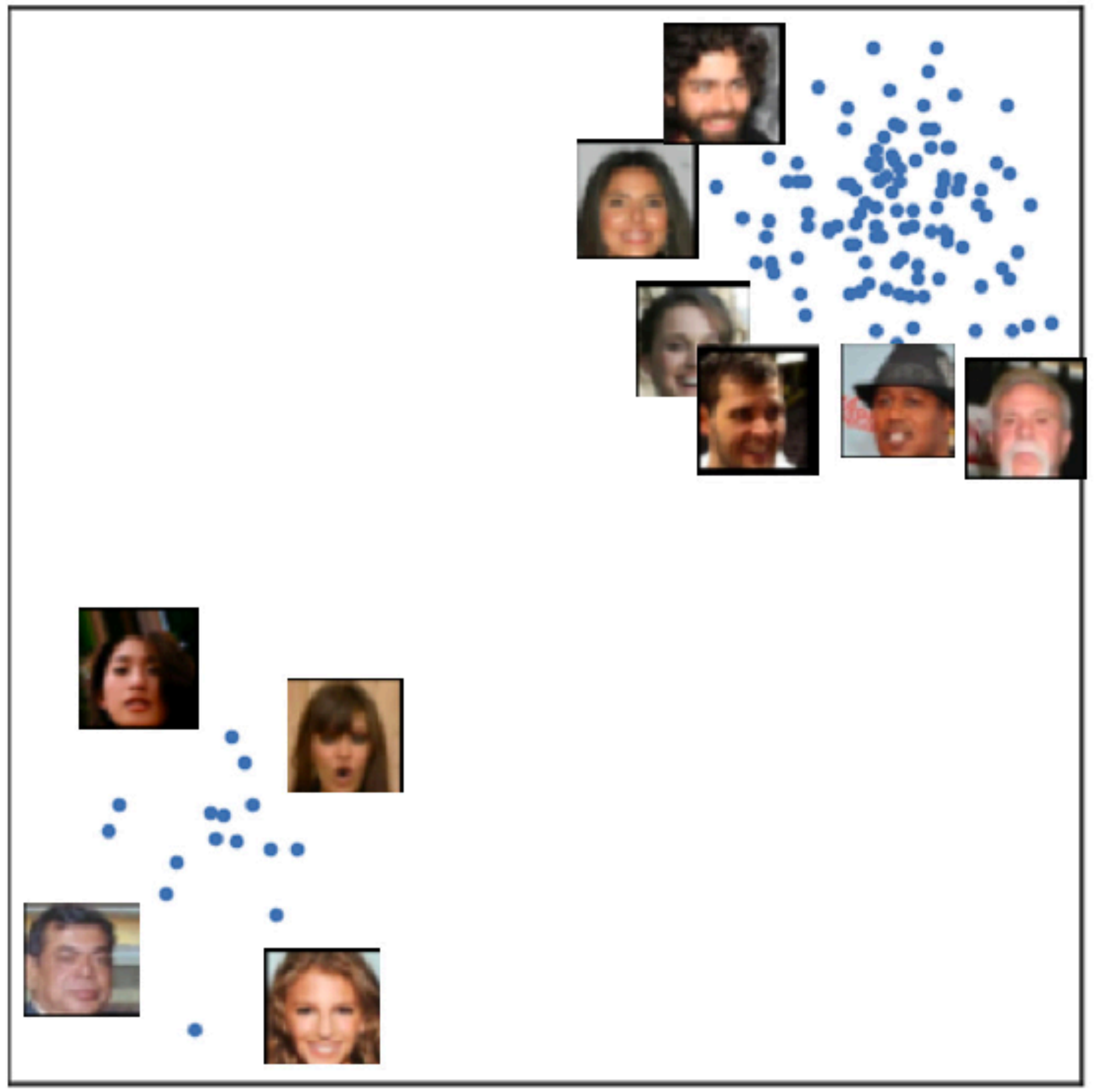
Initialize a random x_0

Update x using noisy gradient steps

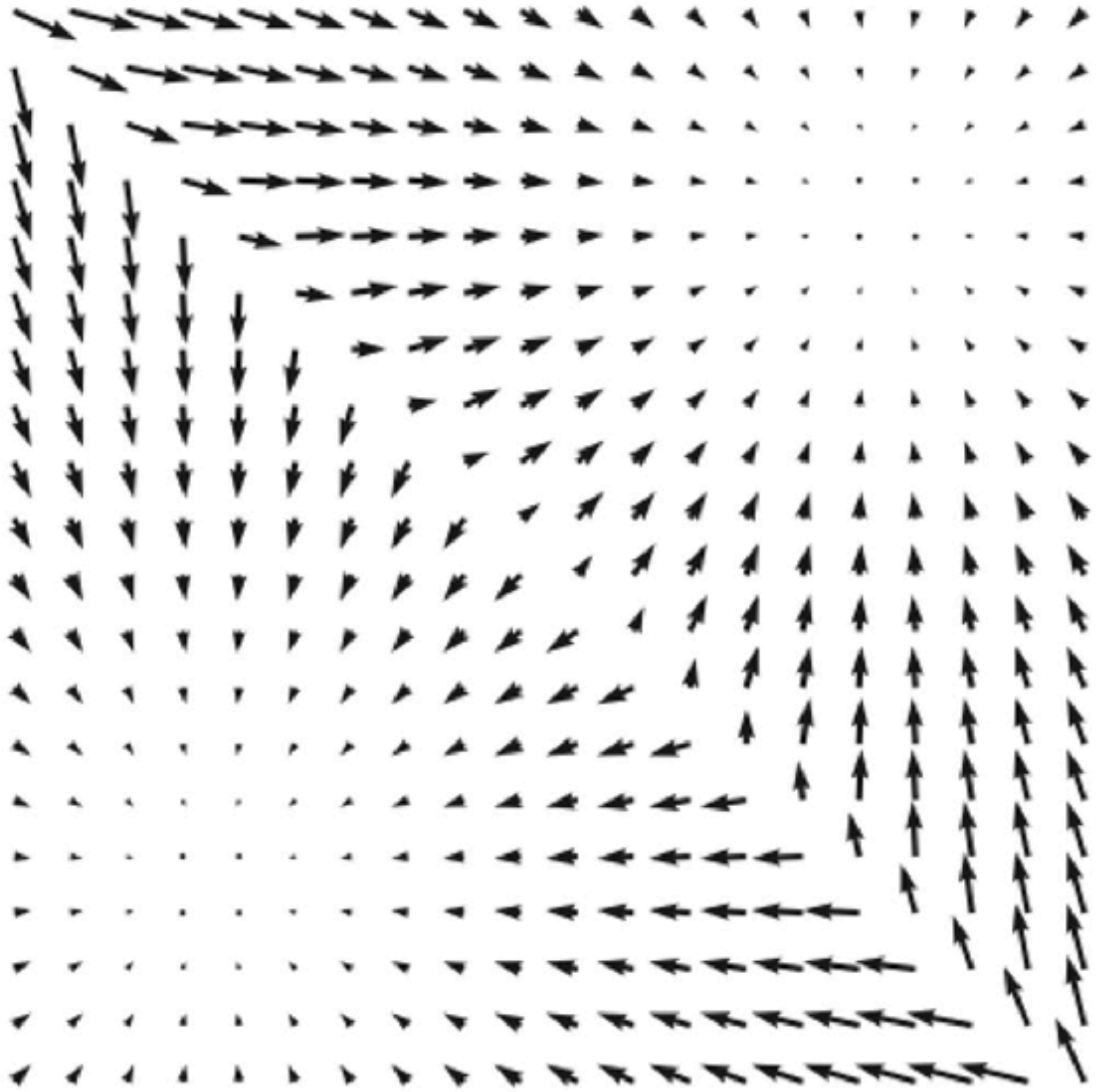
$$x_{i+1} \leftarrow x_i + \epsilon \nabla_x \log p_\theta(x) + \sqrt{2\epsilon} z_i$$

$$z_i \sim \mathcal{N}(0,\mathrm{I})$$

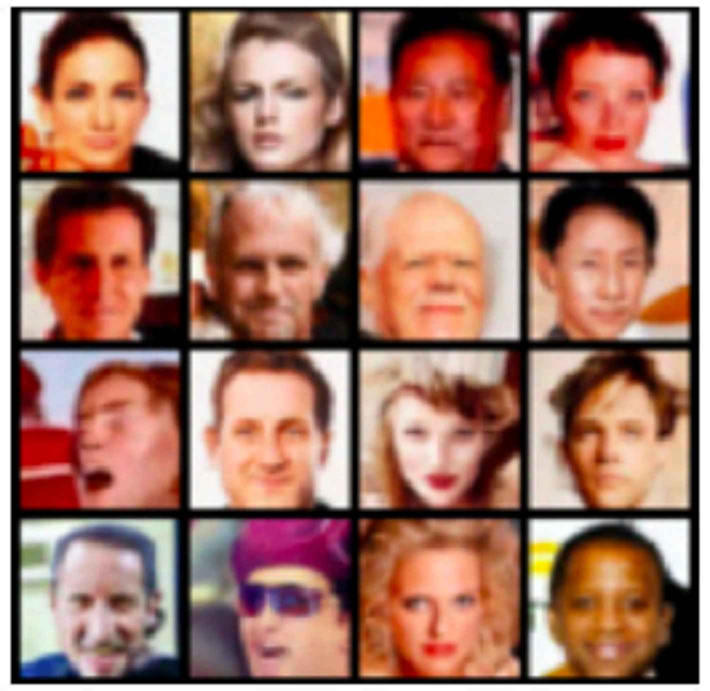
Eventually x_i looks like they come from $p_\theta(x)$



score matching



Langevin dynamics



Data samples

$$\{\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_N\} \overset{\text{i.i.d.}}{\sim} p(\mathbf{x})$$

Scores

$$\mathbf{s}_\theta(\mathbf{x}) \approx \nabla_\mathbf{x} \log p(\mathbf{x})$$

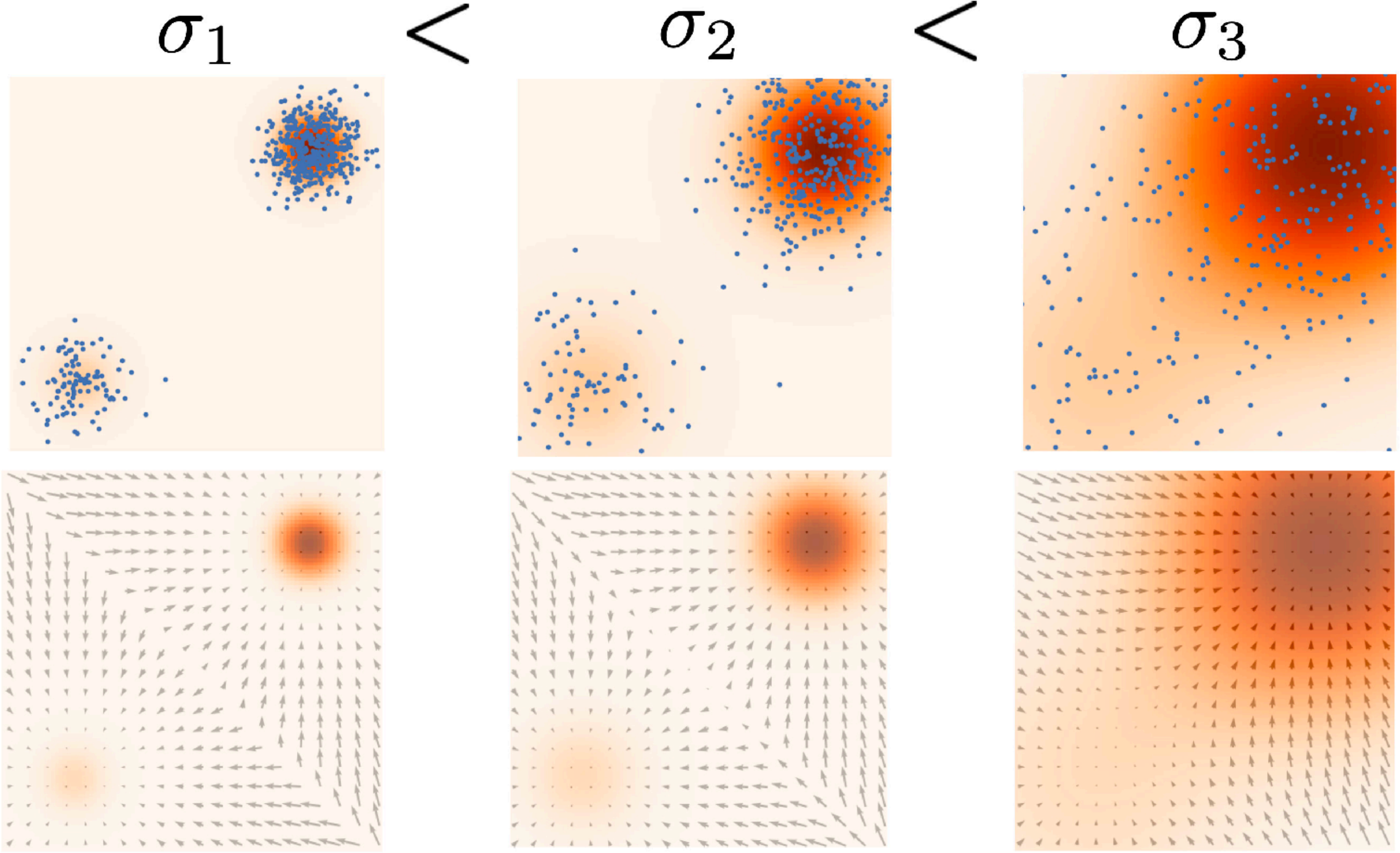
New samples

Problem: Score inaccurate where there is no data

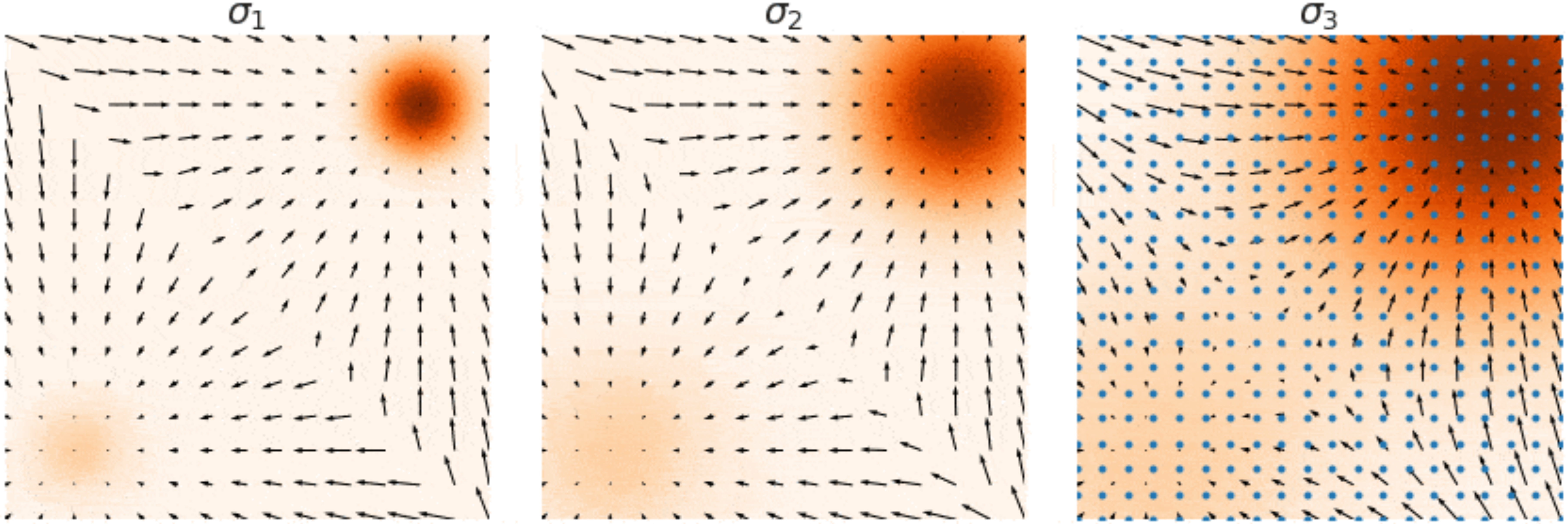
Estimated scores are only accurate in high density regions.

Idea: Add noise to the data!!

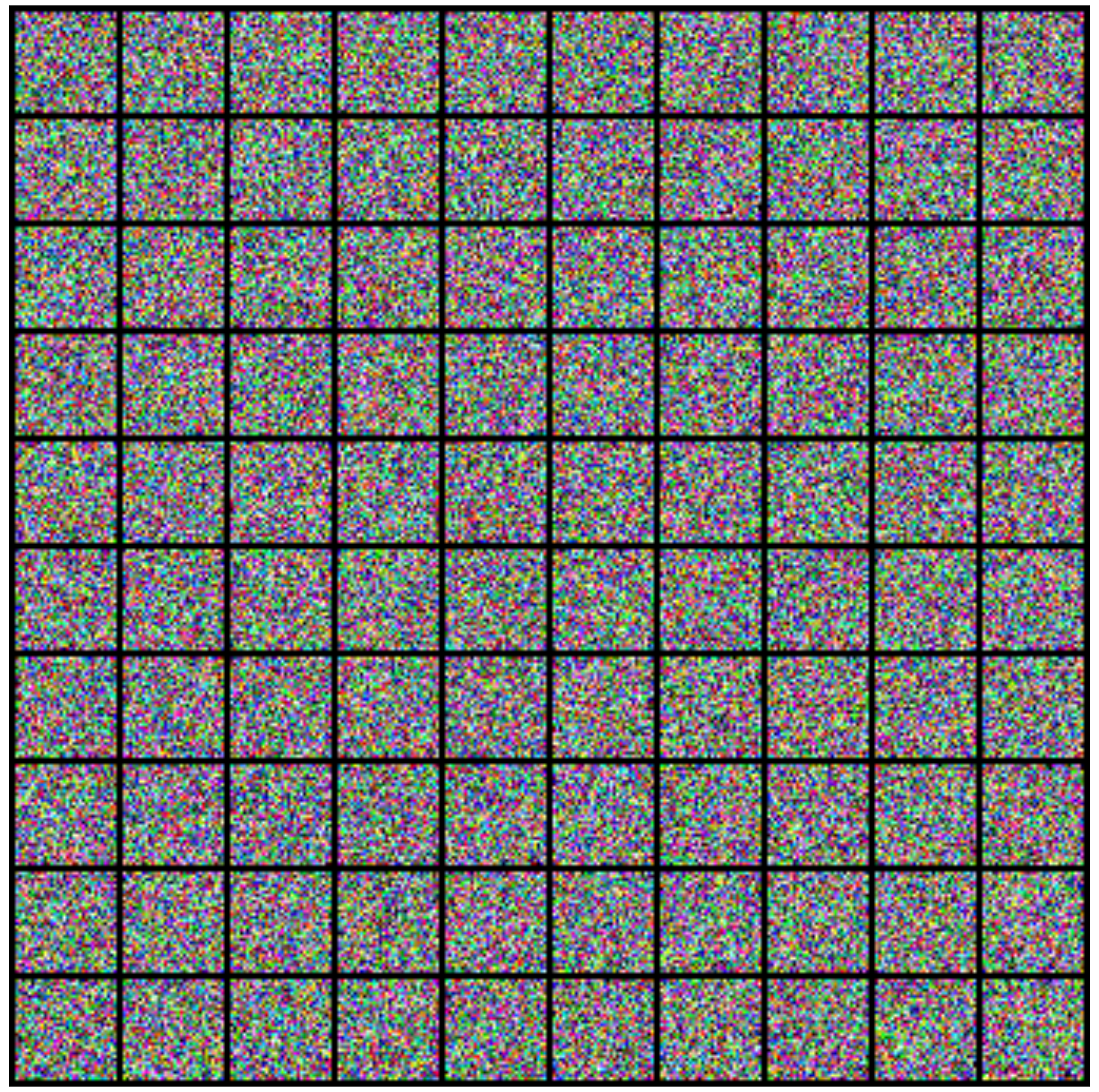
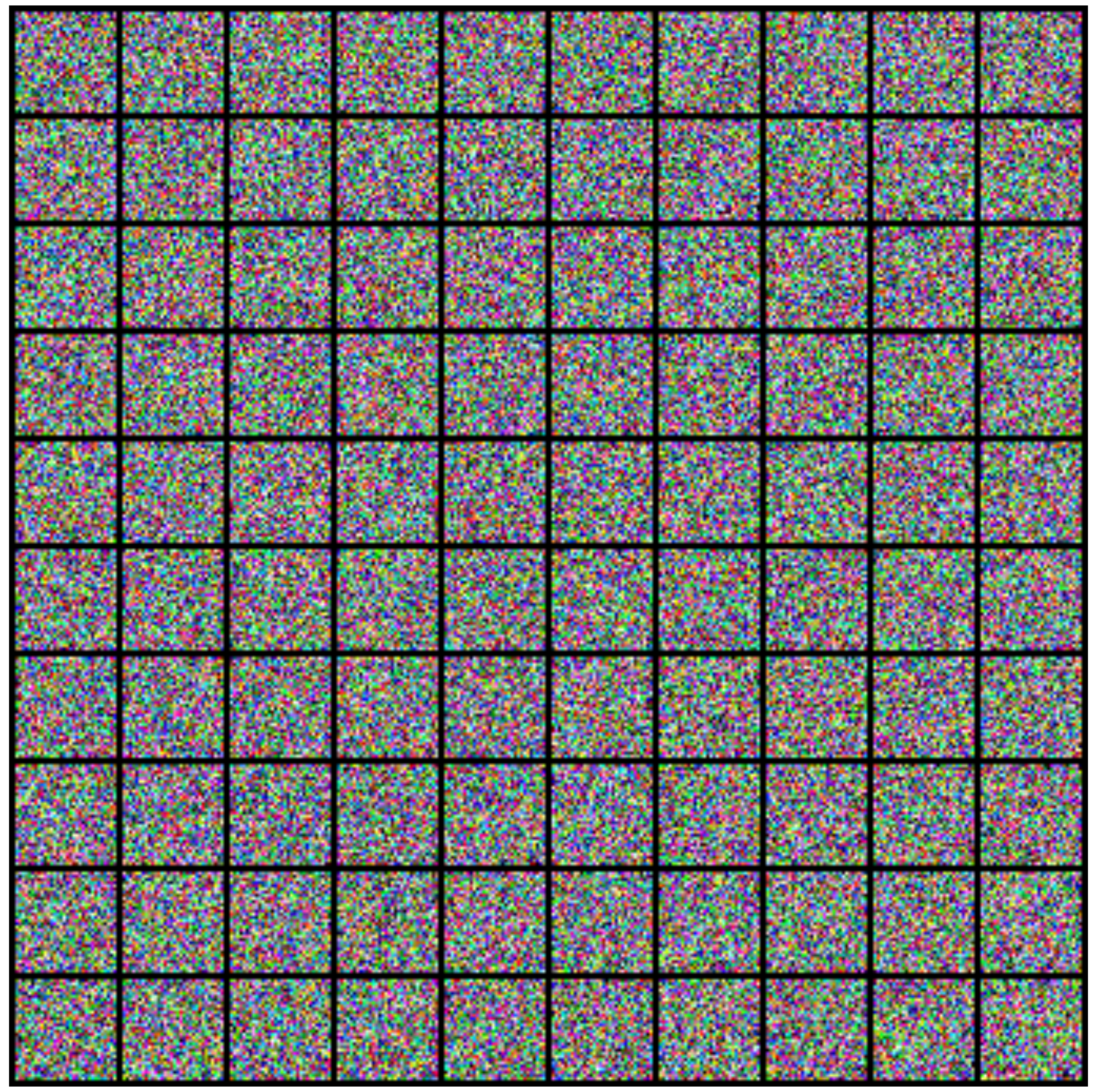
Estimated scores are accurate everywhere for the noise-perturbed data distribution due to reduced low data density regions.



Apply multiple scales of Gaussian noise to perturb the data distribution (first row), and jointly estimate the score functions (second row).

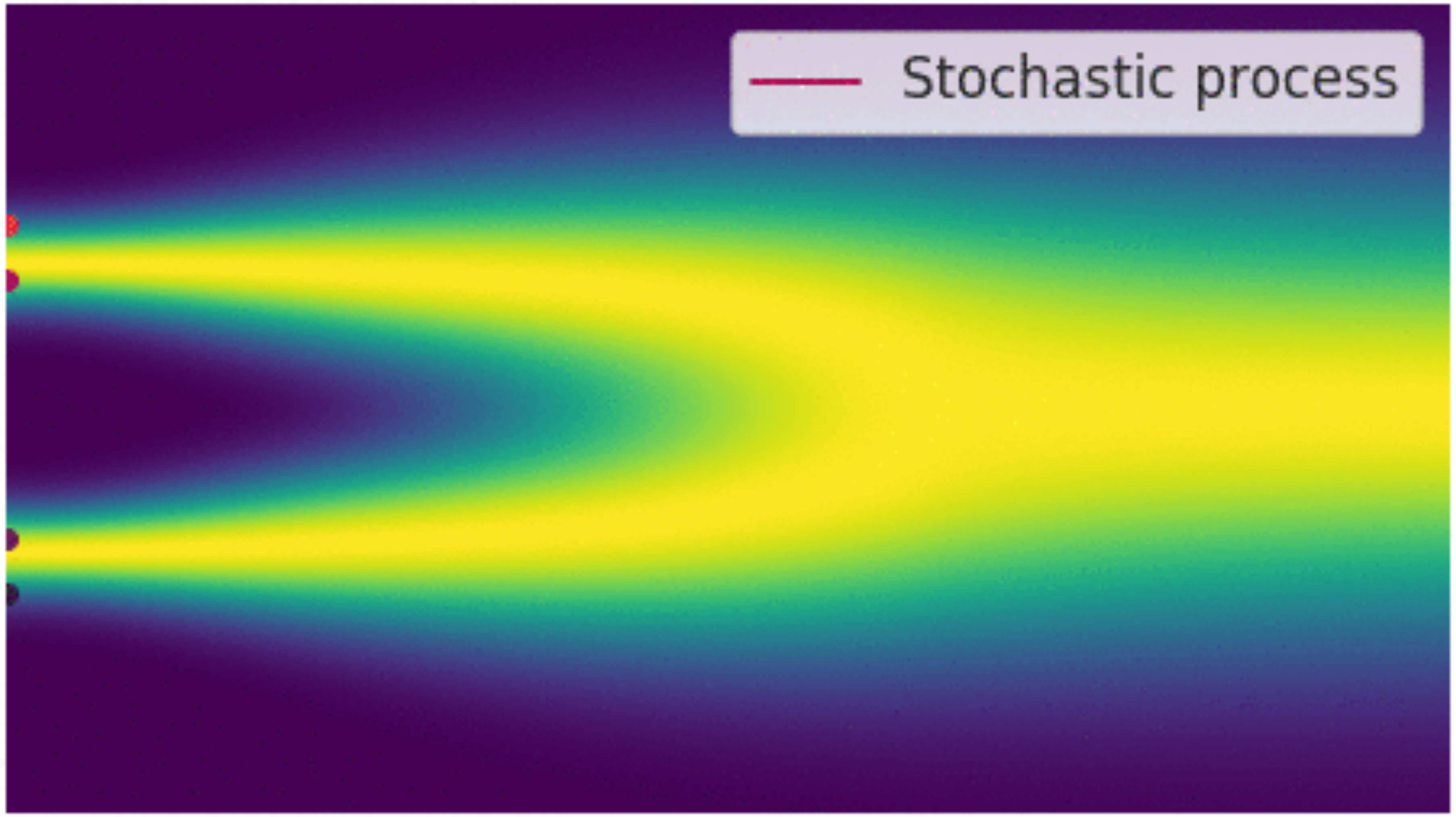


Sequence of Langevin chains with gradually decreasing noise scales.

Celeb-A   CIFAR

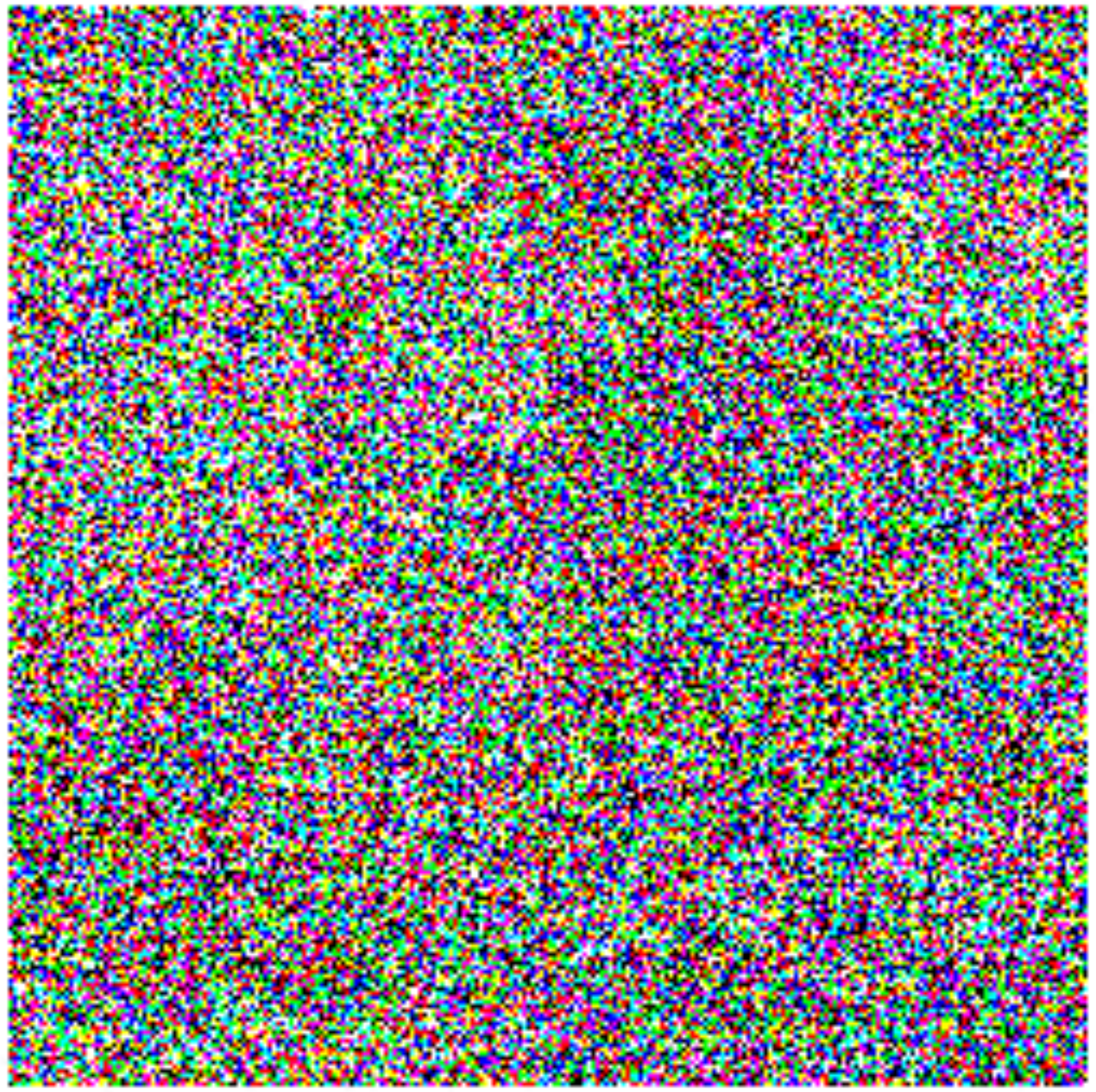
- Choose $\sigma_1 < \sigma_2 < \cdots < \sigma_L$ as a geometric progression, with σ_1 being sufficiently small and σ_L comparable to the maximum pairwise distance between all training data points [19]. L is typically on the order of hundreds or thousands.
- Parameterize the score-based model $\mathbf{s}_\theta(\mathbf{x}, i)$ with U-Net skip connections [18, 20].
- Apply exponential moving average on the weights of the score-based model when used at test time [19, 20].

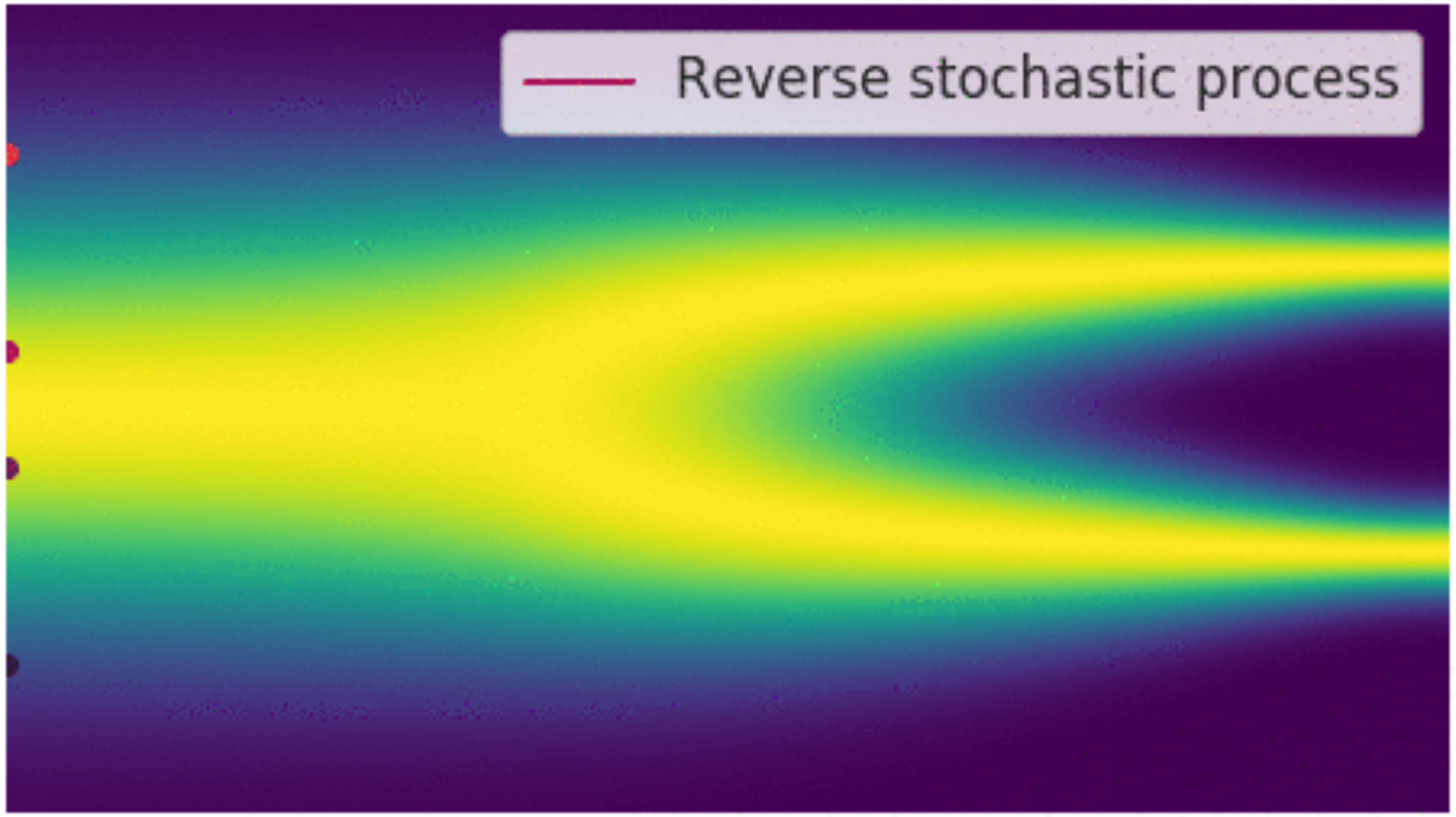
Setting time resolution to zero (continuous time)



$$\mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w},$$

Every SDE has a reverse SDE





$$\mathrm{d}\mathbf{x} = [\mathbf{f}(\mathbf{x}, t) - g^2(t)\nabla_{\mathbf{x}} \log p_t(\mathbf{x})]\mathrm{d}t + g(t)\mathrm{d}\mathbf{w}.$$

Forward SDE (data → noise)

$$\mathbf{x}(0) \longrightarrow \mathrm{d}\mathbf{x} = \mathbf{f}(\mathbf{x}, t)\mathrm{d}t + g(t)\mathrm{d}\mathbf{w} \longrightarrow \mathbf{x}(T)$$

score function

$$\mathbf{x}(0) \longleftarrow \mathrm{d}\mathbf{x} = \left[\mathbf{f}(\mathbf{x}, t) - g^2(t)\boxed{\nabla_{\mathbf{x}} \log p_t(\mathbf{x})}\right] \mathrm{d}t + g(t)\mathrm{d}\bar{\mathbf{w}} \longleftarrow \mathbf{x}(T)$$

Reverse SDE (noise → data)

Why do we care?

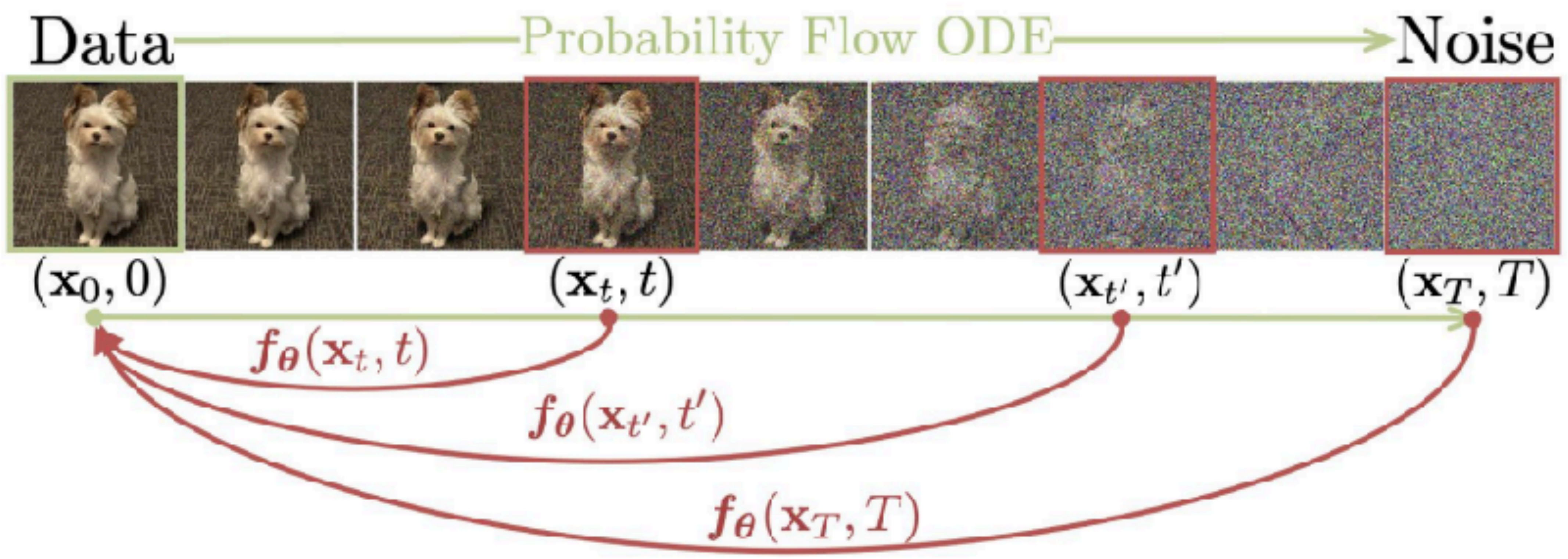
Gives rise to a whole new landscape of ways to estimate score function

No longer restricted to small step size in diffusion models

Consistency models are one such example!

Consistency Models

Yang Song[1] Prafulla Dhariwal[1] Mark Chen[1] Ilya Sutskever[1]



Open questions: Can we pull-back these ideas to decision making?