

Flow Models

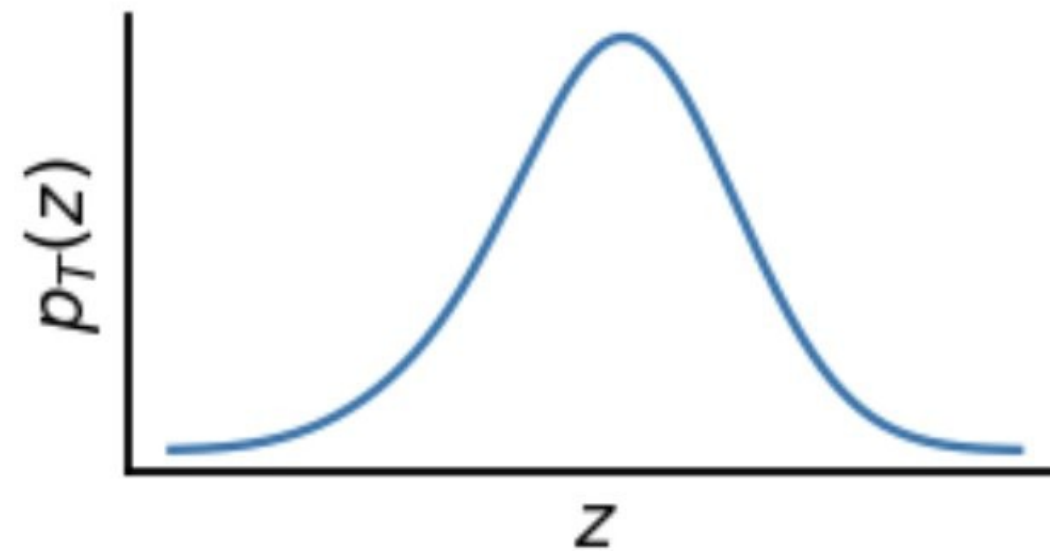
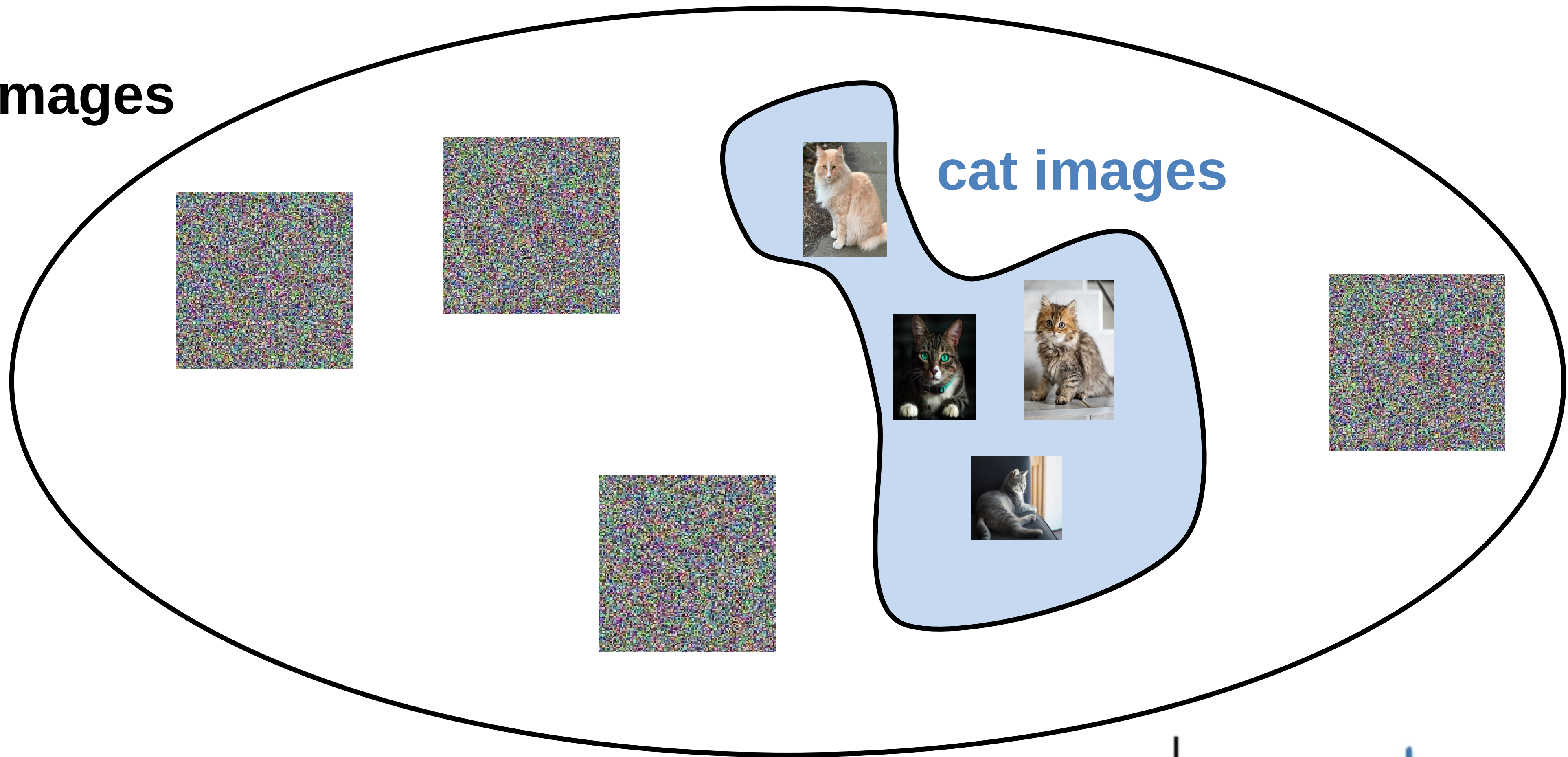


CS180: Intro to Comp. Vision and Comp. Photo
Efros & Ng, UC Berkeley, Fall 2026

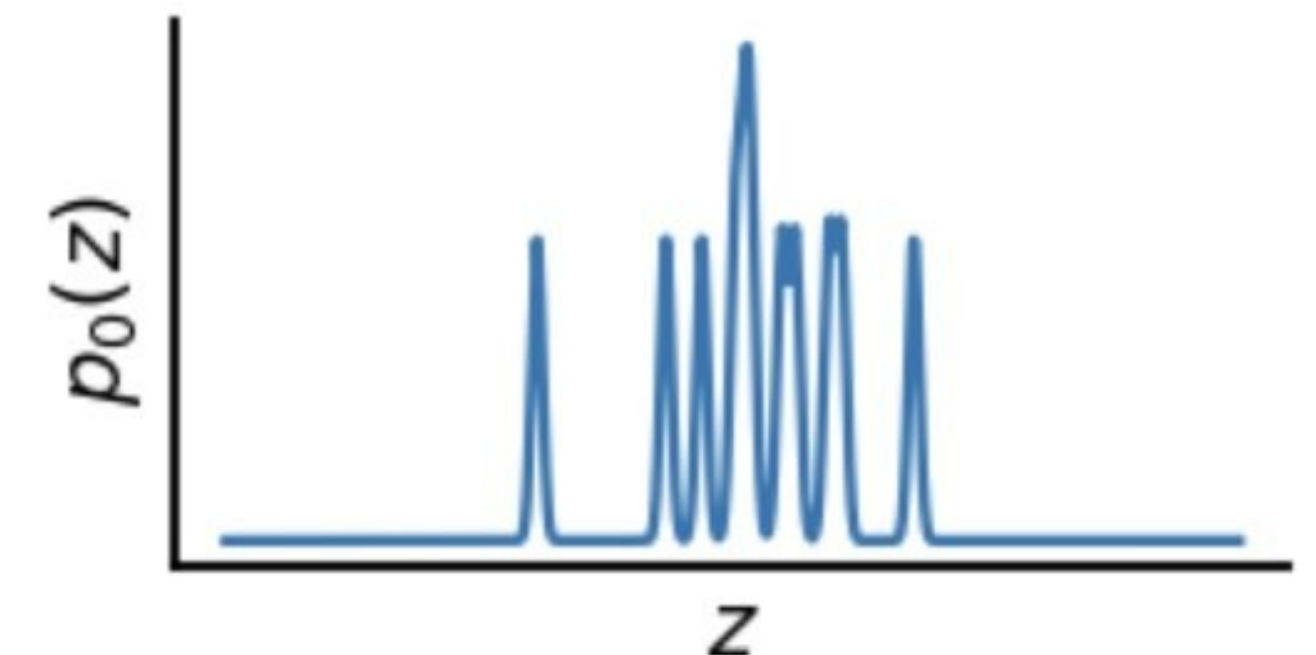
with slides by Angjoo Kanazawa

Generating Images from Noise

Random images

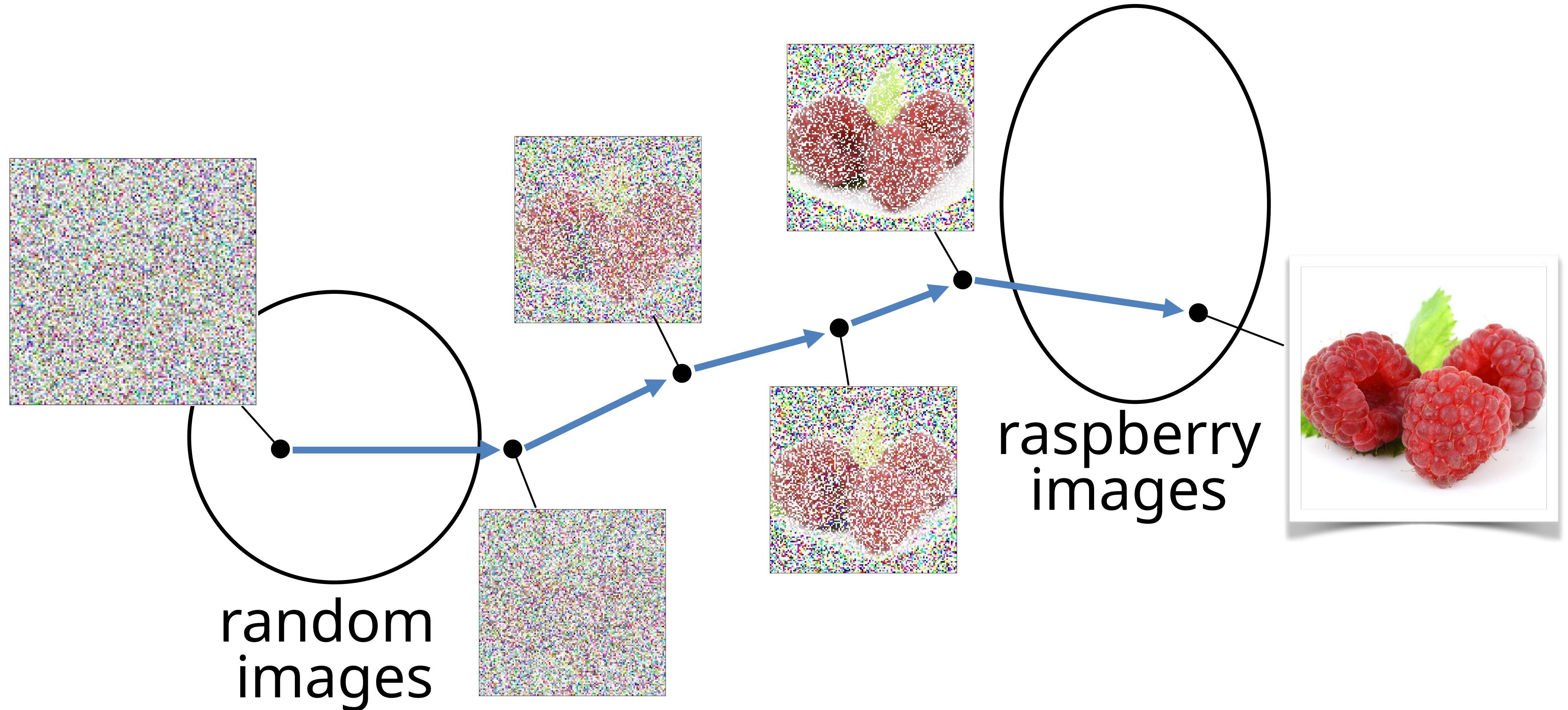


Noise to Images



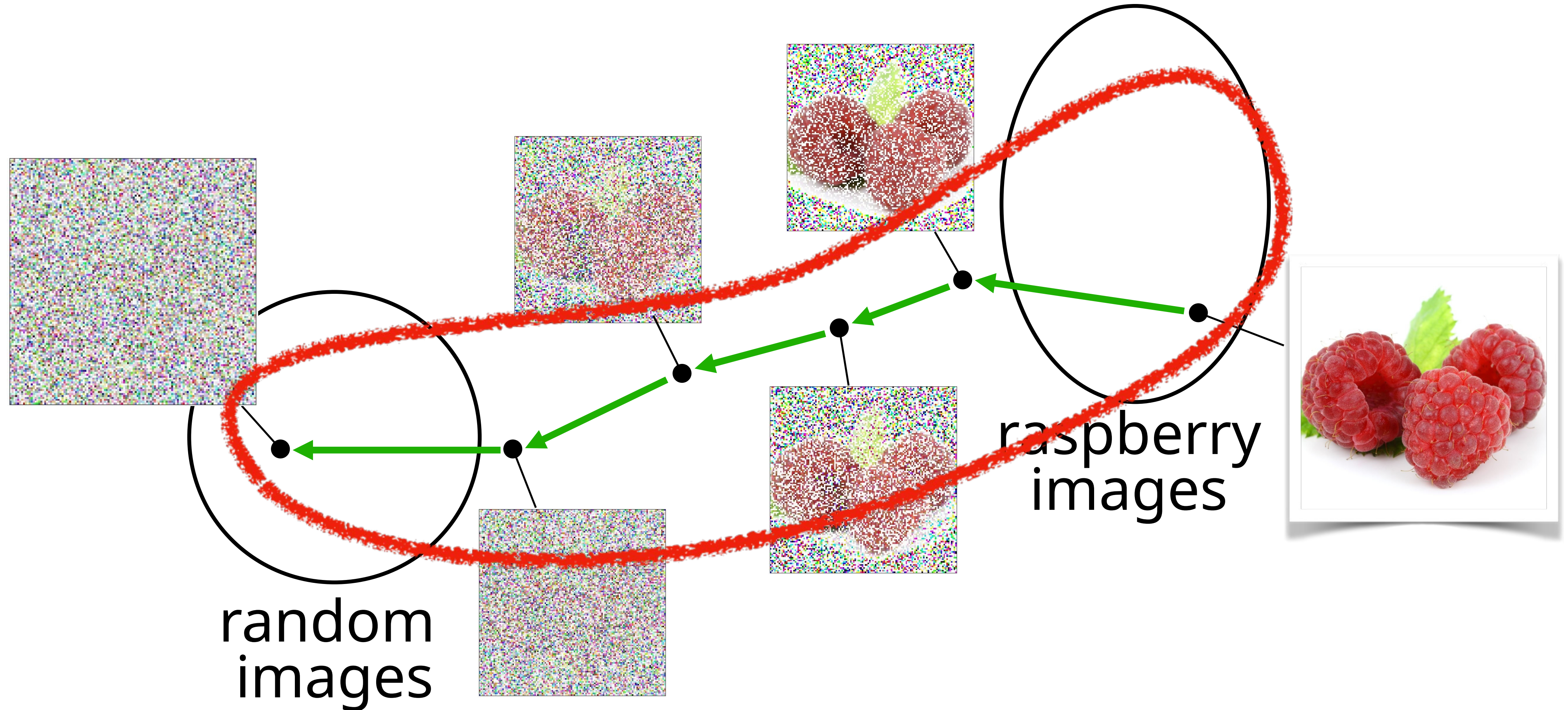
Recap

1. Take real data, corrupt it to left distribution somehow
2. Learn the velocity that undoes the process! (Denoise / Reverse)



\$\$\$ question, how to pick the intermediate path?

How to generate this Green path?



all the obvious estimators of $\ln p_\theta(x)$ are biased downwards, because no matter how many samples of $z_i \sim q_\phi(\cdot|x)$ we take, we have by Jensen's inequality:

$$\mathbb{E}_{z_i \sim q_\phi(\cdot|x)} \left[\ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \geq \mathbb{E}_{z \sim q_\phi(\cdot|x)} \left[\ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]$$

Subtracting the right side, we see that the problem comes down to a biased estimator of zero:

$$\mathbb{E}_{z_i \sim q_\phi(\cdot|x)} \left[\ln \left(\frac{1}{N} \sum_i \frac{p_\theta(x, z_i)}{q_\phi(z_i|x)} \right) \right] \leq \ln \mathbb{E}_{z_i \sim q_\phi(\cdot|x)} \left[\frac{1}{N} \sum_i \frac{p_\theta(x, z_i)}{q_\phi(z_i|x)} \right] = \ln p_\theta(x)$$

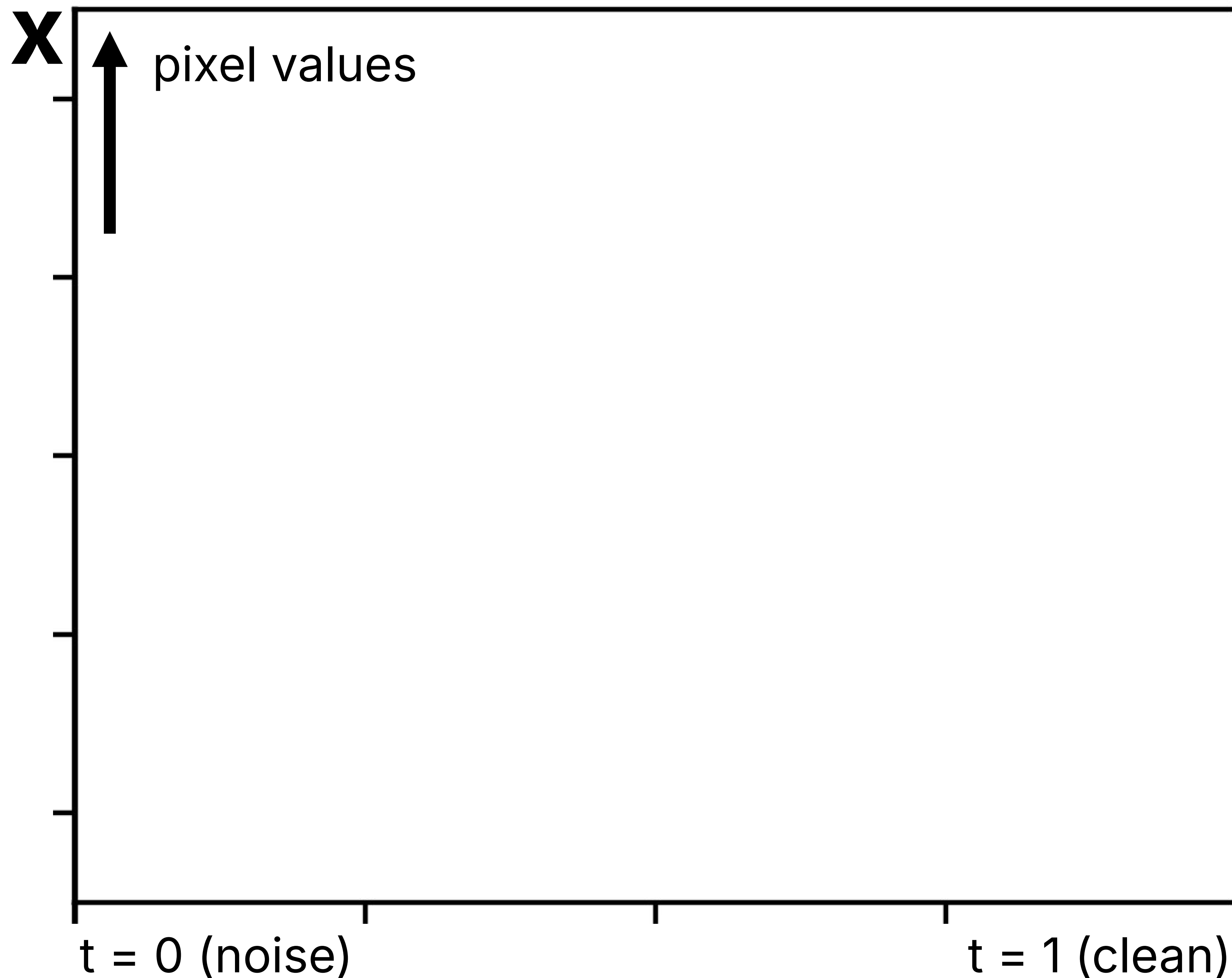
At this point, we could branch off towards the development of an importance-weighted autoencoder^{[\[note 2\]](#)}, but we will instead continue with the simplest case with $N = 1$:

$$\ln p_\theta(x) = \ln \mathbb{E}_{z \sim q_\phi(\cdot|x)} \left[\frac{p_\theta(x, z)}{q_\phi(z|x)} \right] \geq \mathbb{E}_{z \sim q_\phi(\cdot|x)} \left[\ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right]$$

The tightness of the inequality has a closed form:

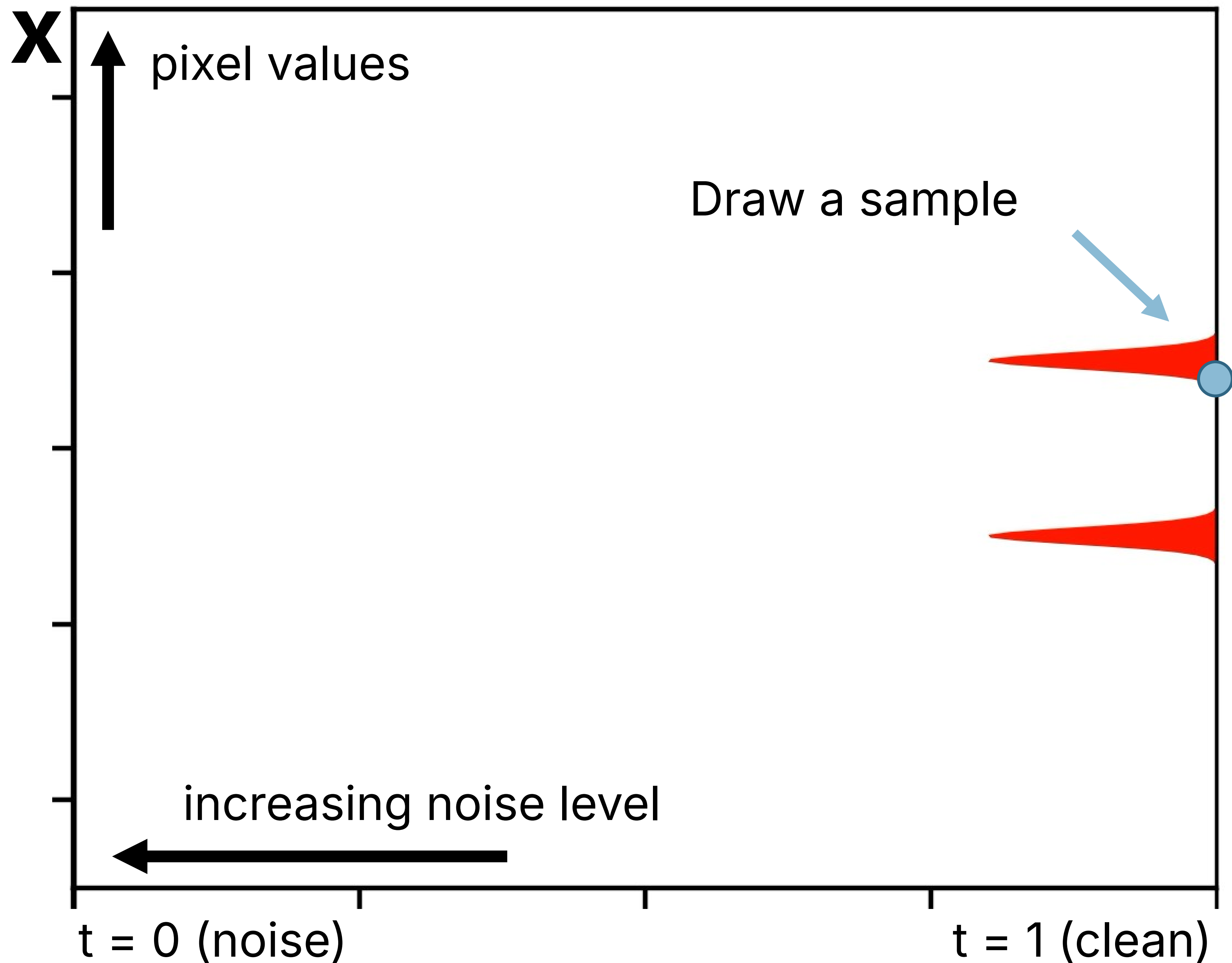
$$\ln p_\theta(x) - \mathbb{E}_{z \sim q_\phi(\cdot|x)} \left[\ln \frac{p_\theta(x, z)}{q_\phi(z|x)} \right] = D_{KL}(q_\phi(\cdot|x) \parallel p_\theta(\cdot|x))$$

We have thus obtained the ELBO function:



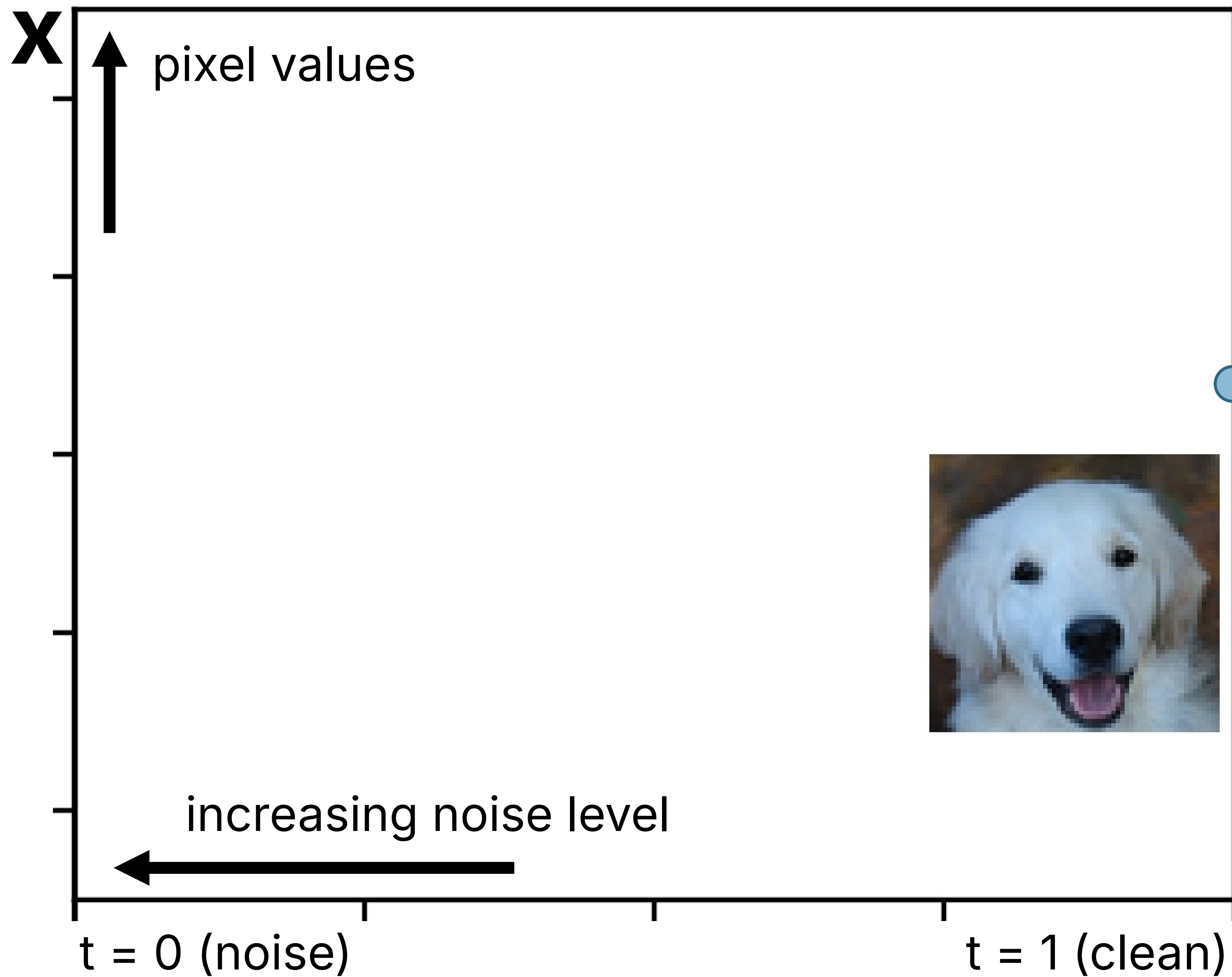
Data distribution

- Data lives in high dimensional space of pixel values
- We'll visualize in 1D



Data distribution

- Data lives in high dimensional space of pixel values
- Let's visualize in 1D



**Data
distribution**

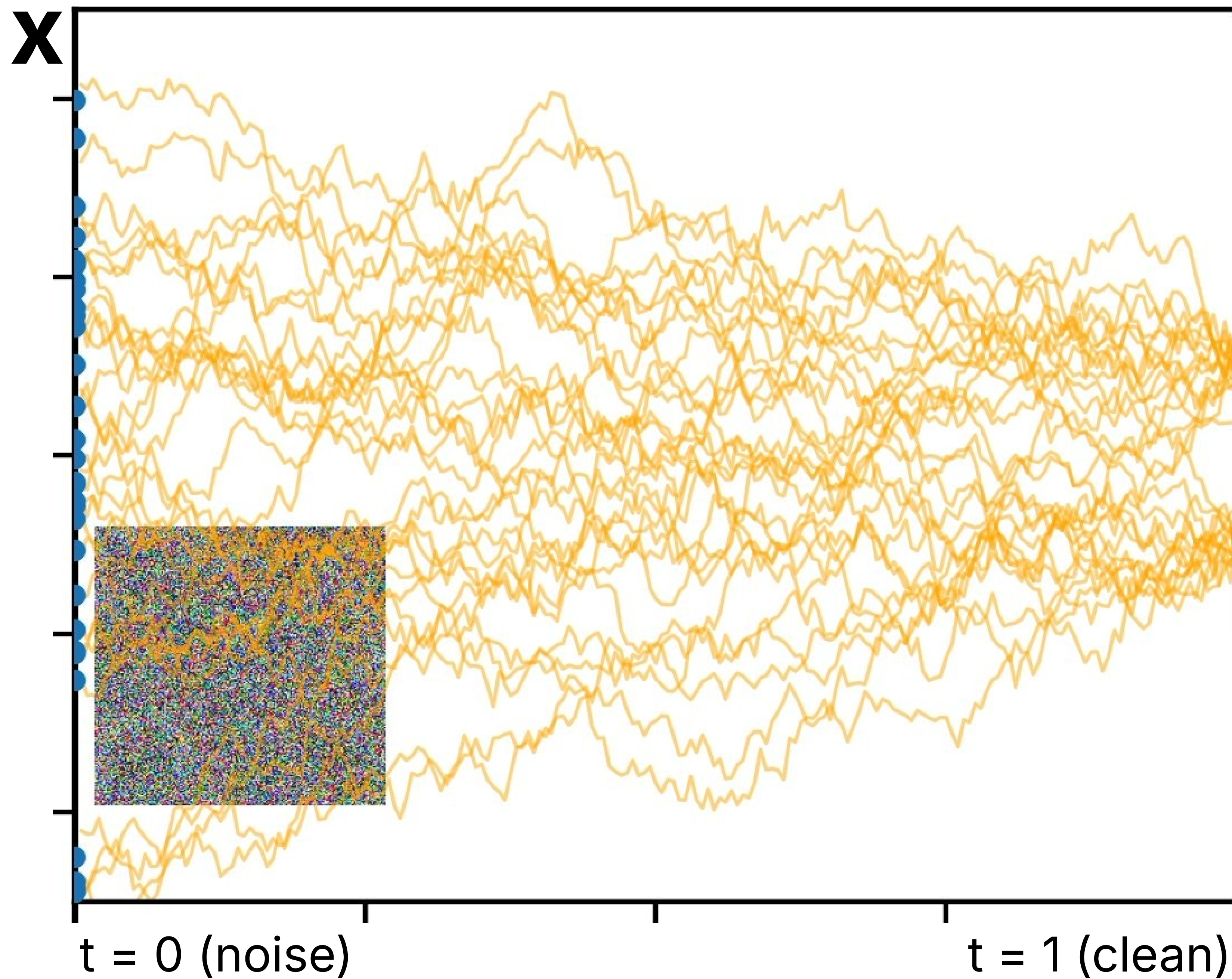
X

**Gradually add
noise to
destroy the
image**

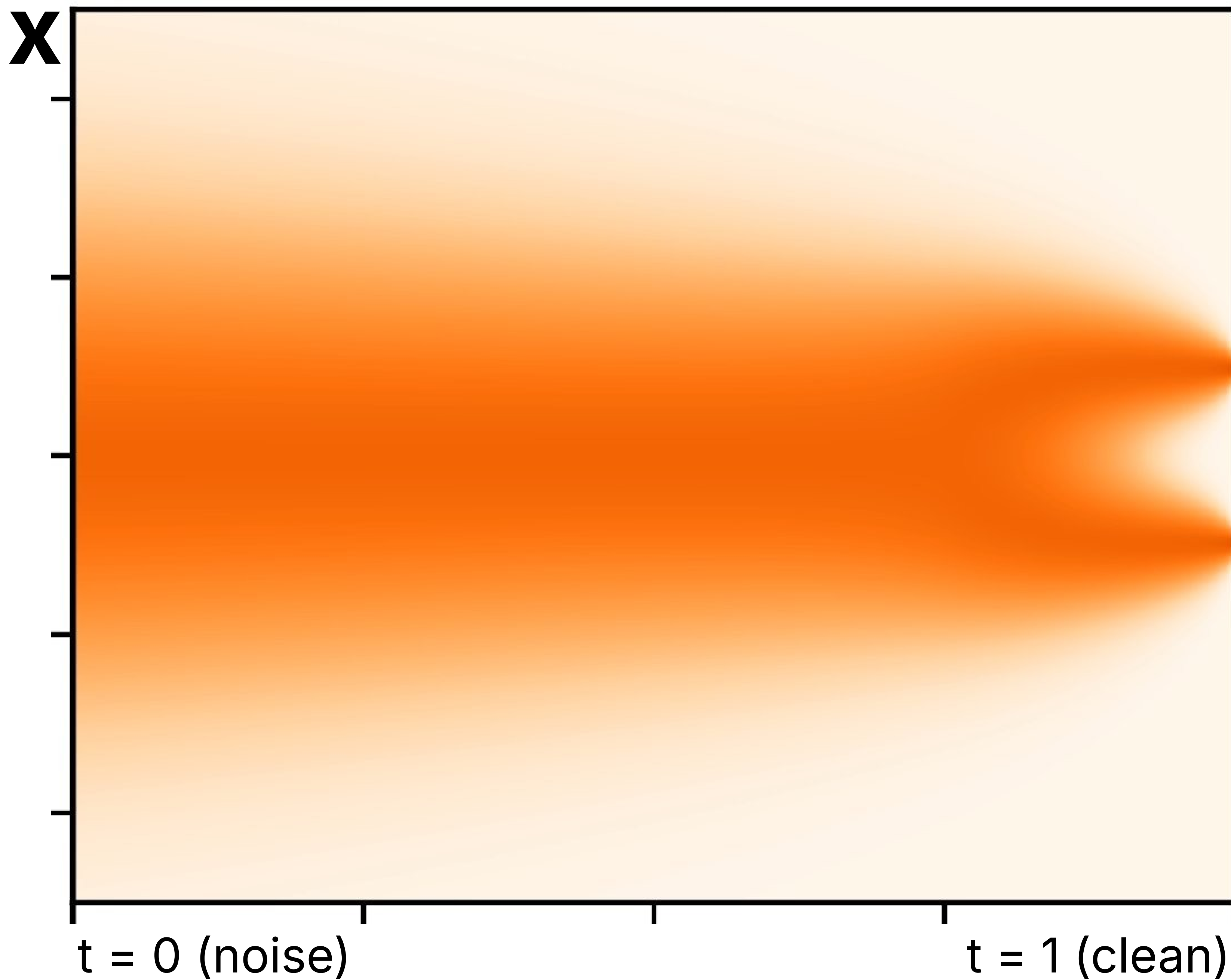


t = 0 (noise)

t = 1 (clean)

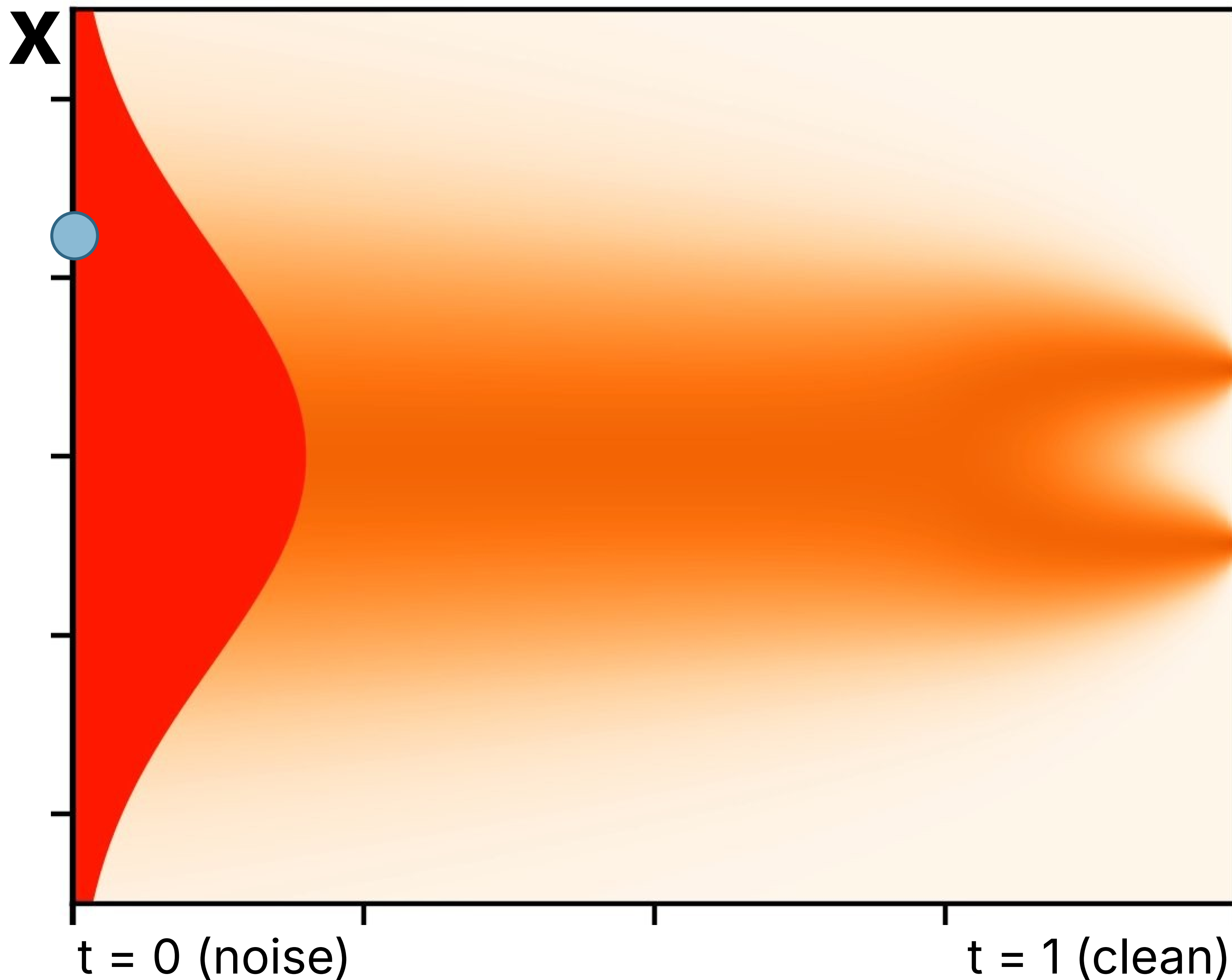


**Data density is
diffused as t
decreases**



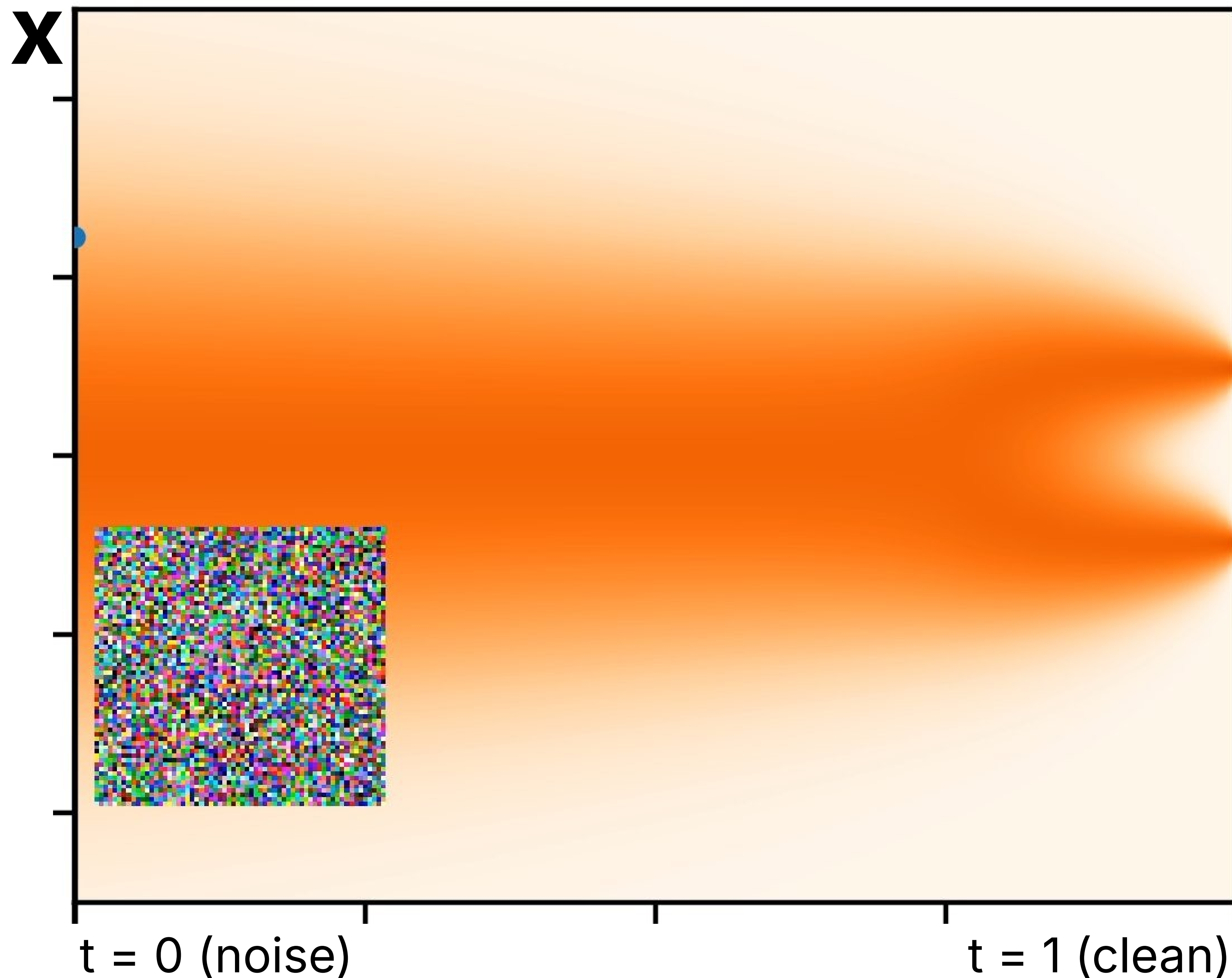
**Data density is
diffused as t
decreases**

- All data ends up normally distributed



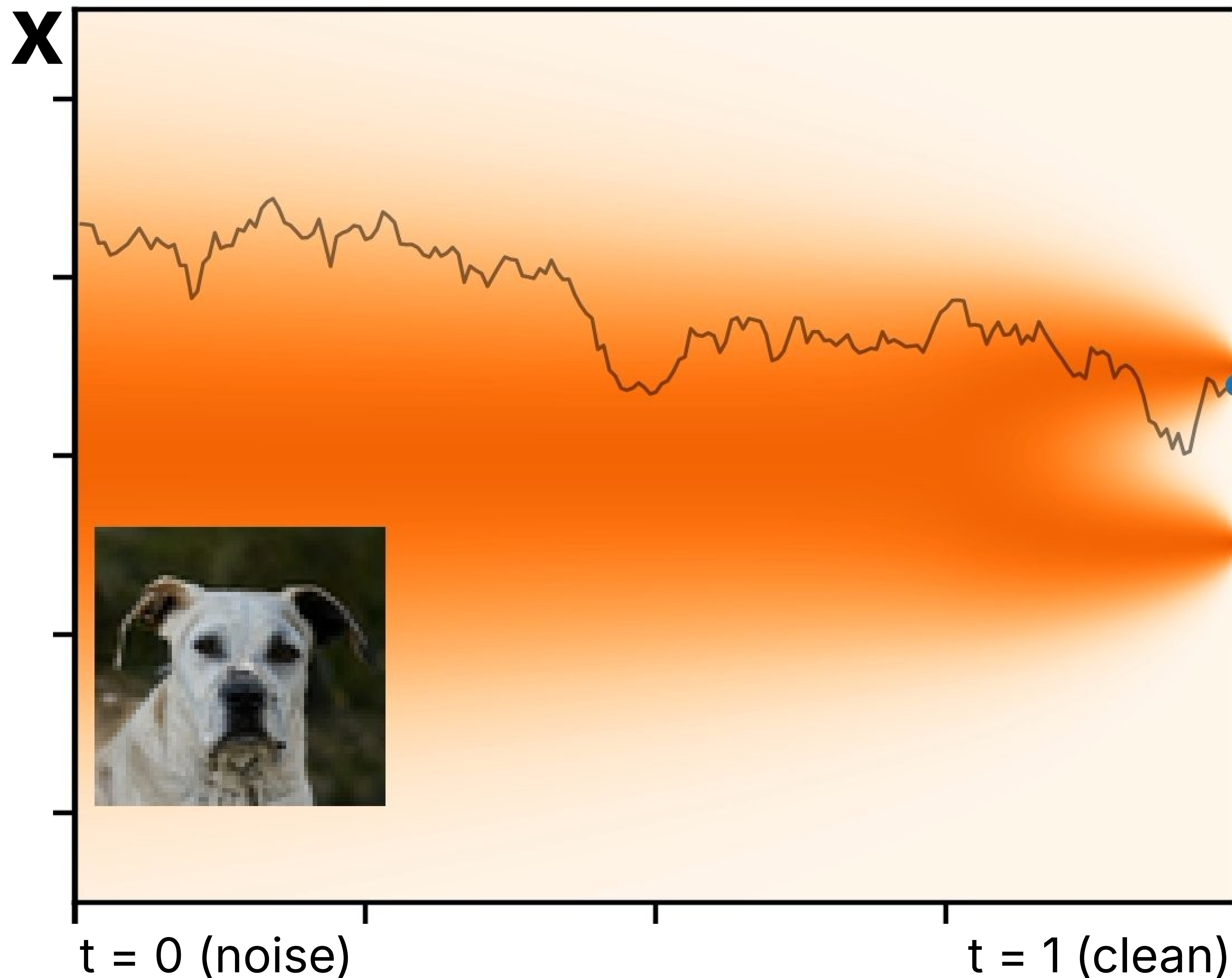
**Data density is
diffused as t
decreases**

- All data ends up normally distributed
- Let's draw a sample...



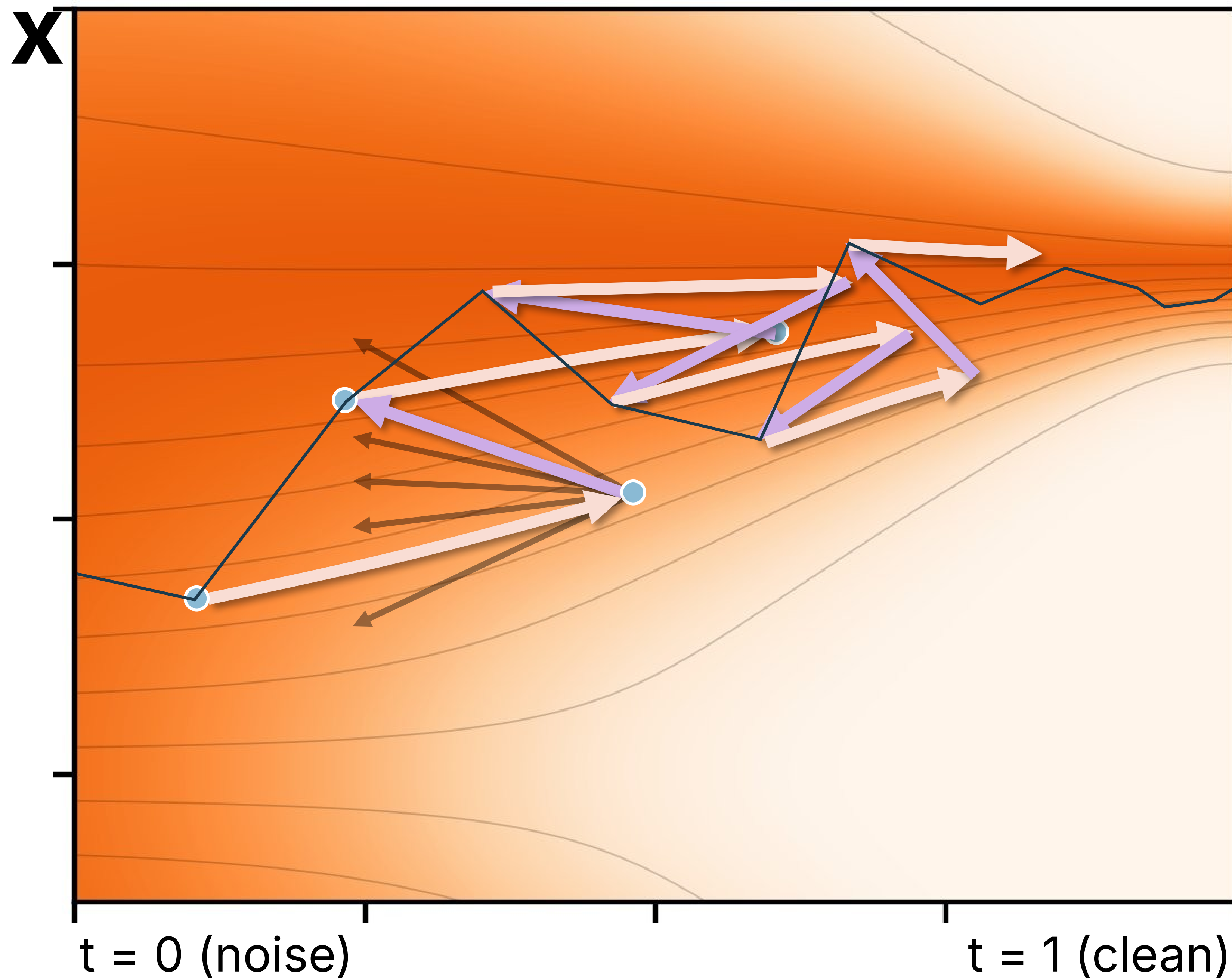
We can *reverse* the trajectory by gradual denoising

- Endpoint is a random sample from data distribution!



We can *reverse* the trajectory by gradual denoising

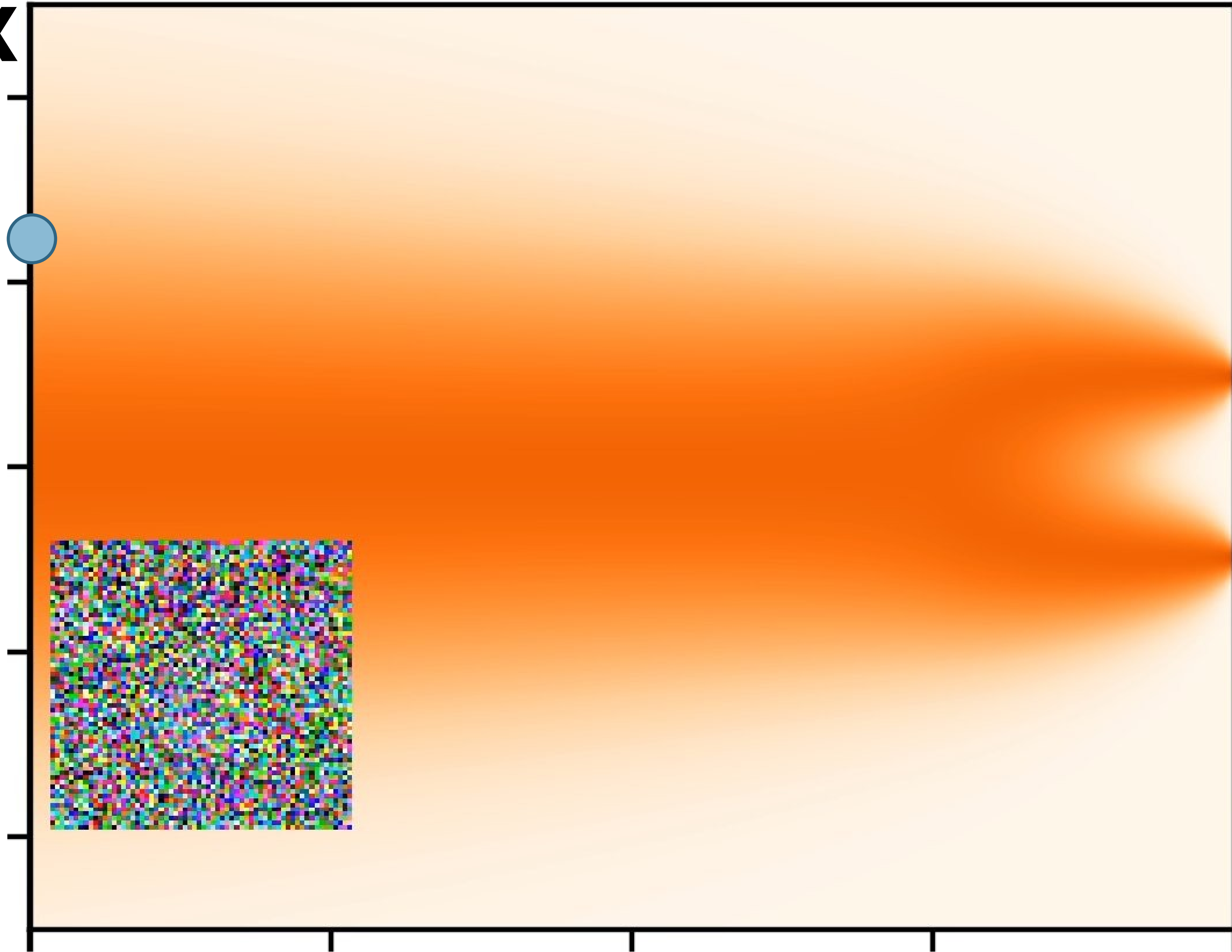
- Endpoint is a random sample from data distribution!



Why wiggles?

- Take one step to reduce noise
- Add a bit of fresh noise back

X



t = 0 (noise)

t = 1 (clean)

Flow Models

- Deterministic

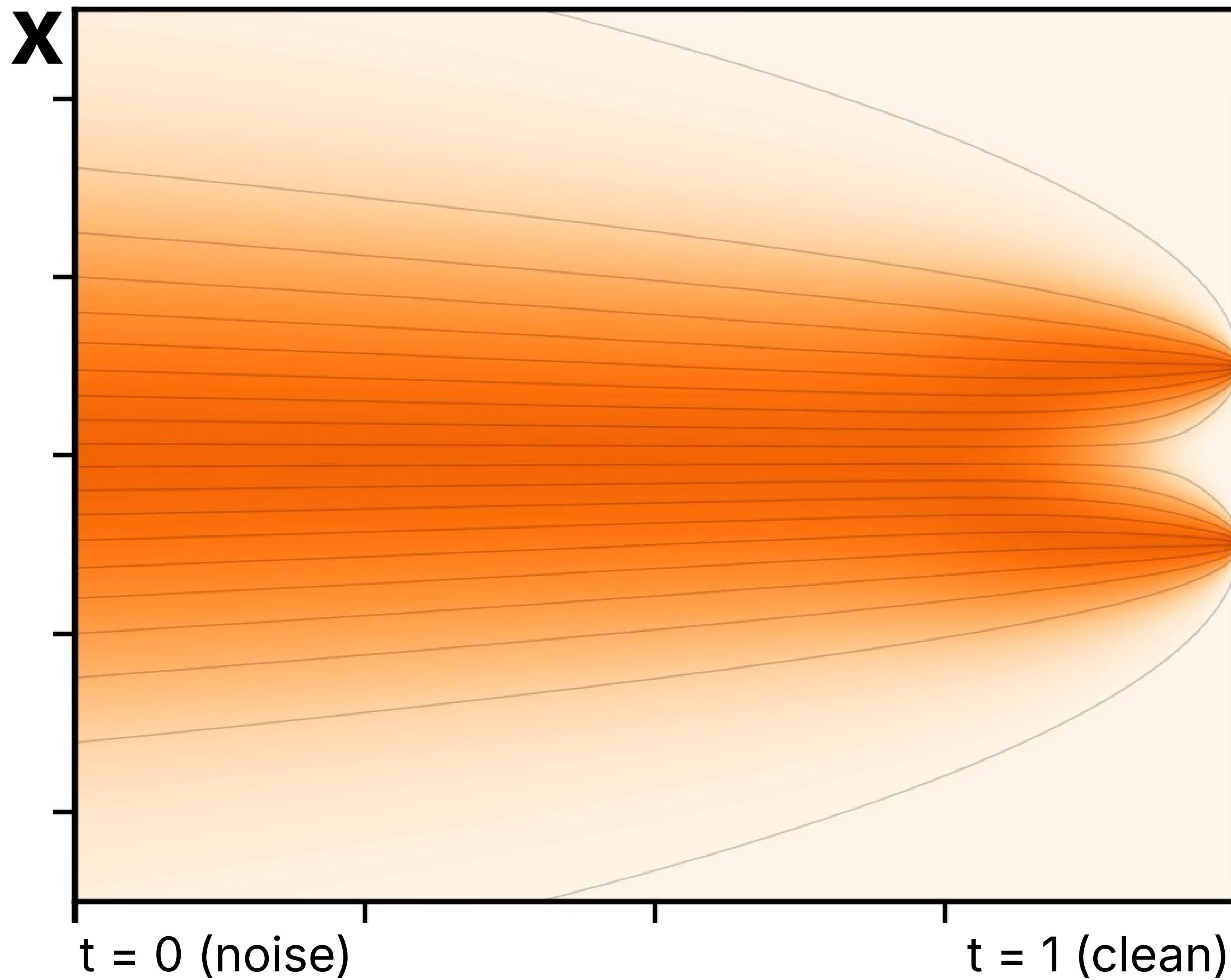
X

**Deterministic
ODE**

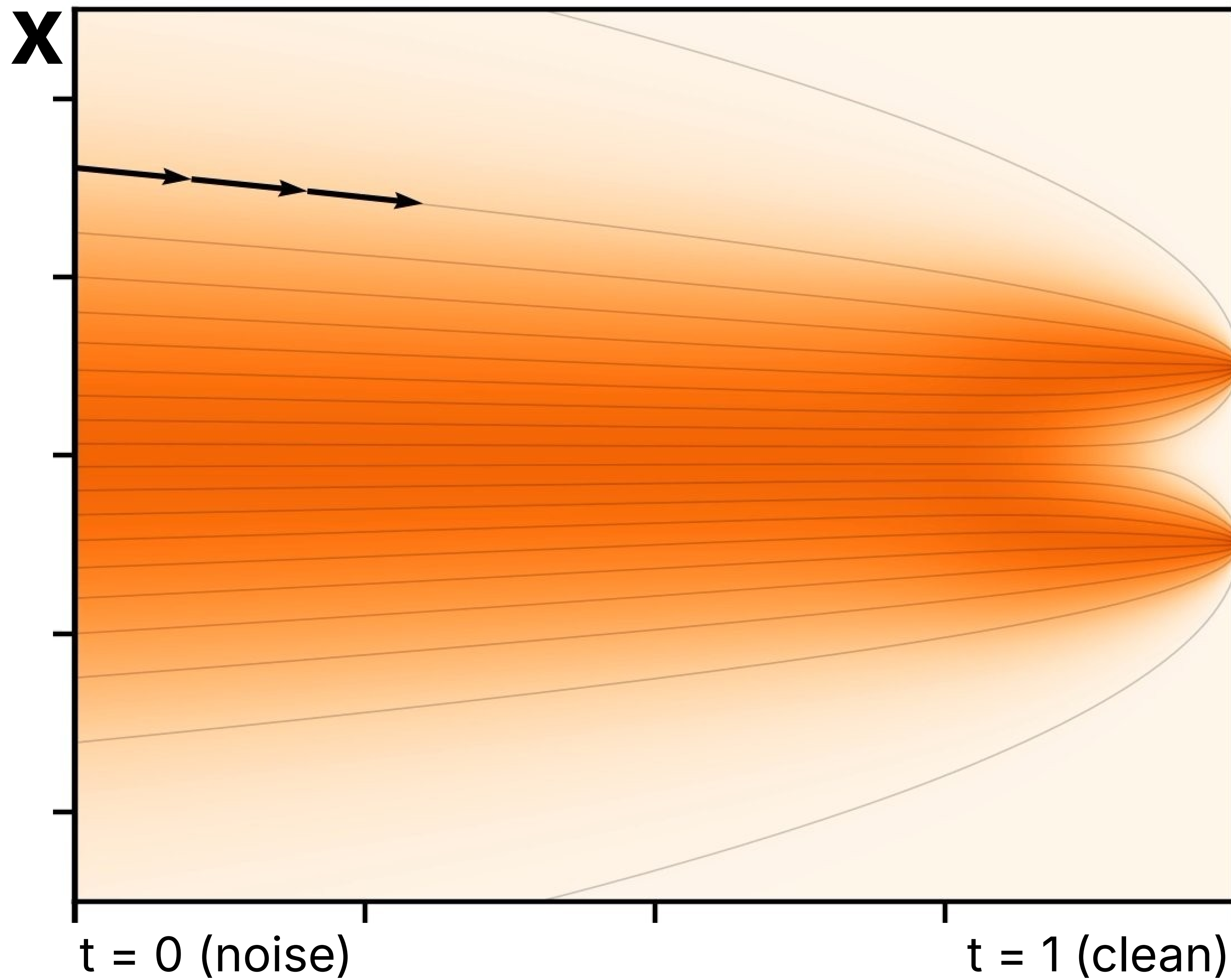


$t = 0$ (noise)

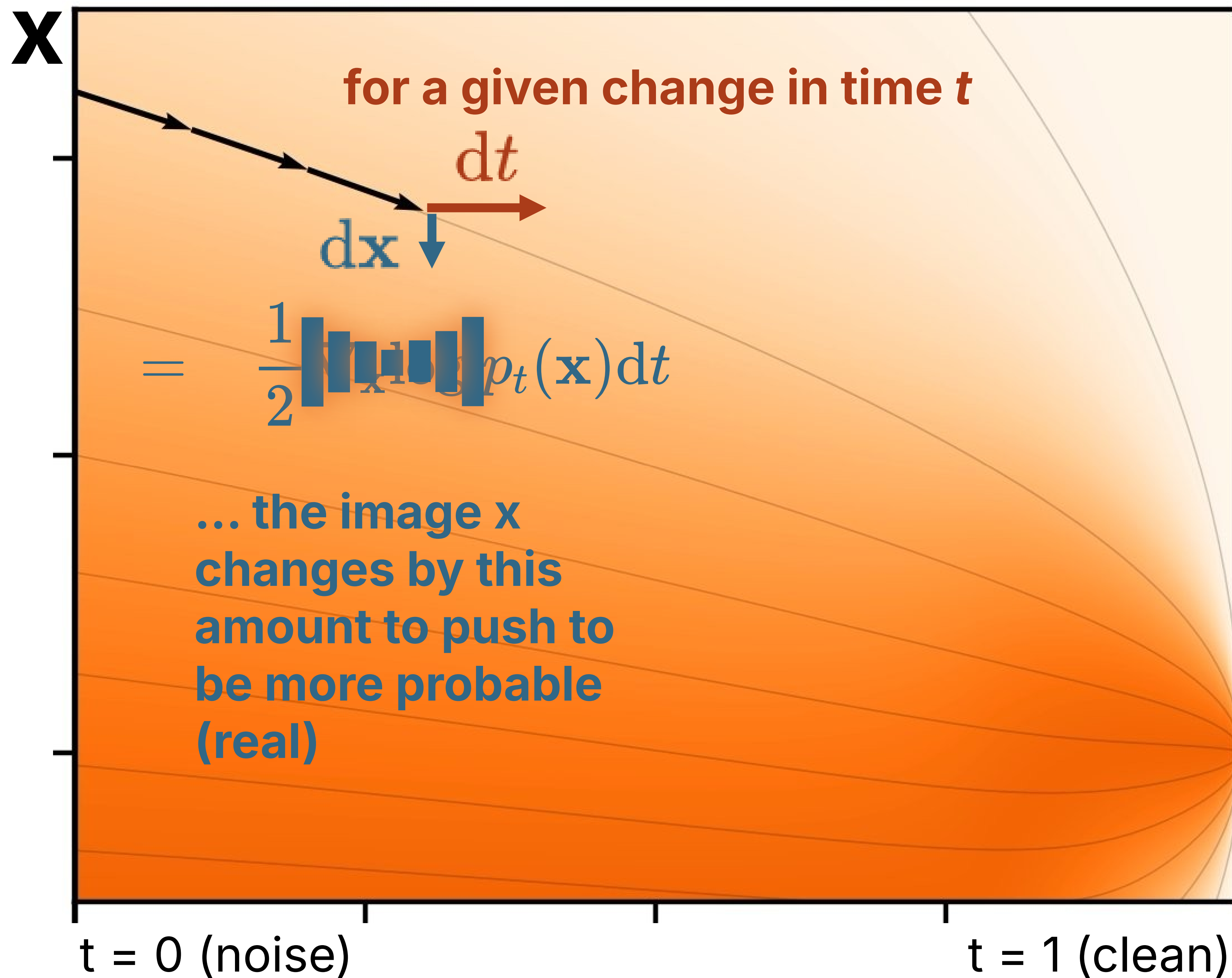
$t = 1$ (clean)



**Solution by
discretization**



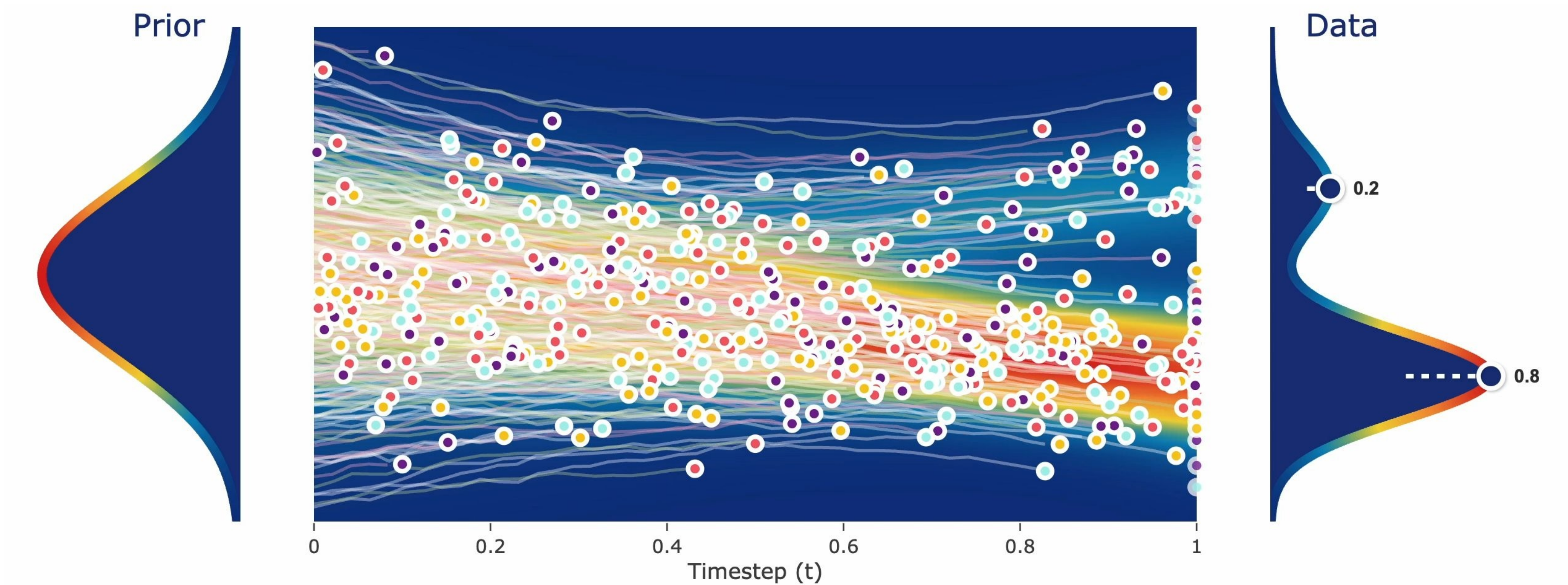
**Solution by
discretization**



**Solution by
discretization**

Continuous Flows

Transport Map from Noise to Data

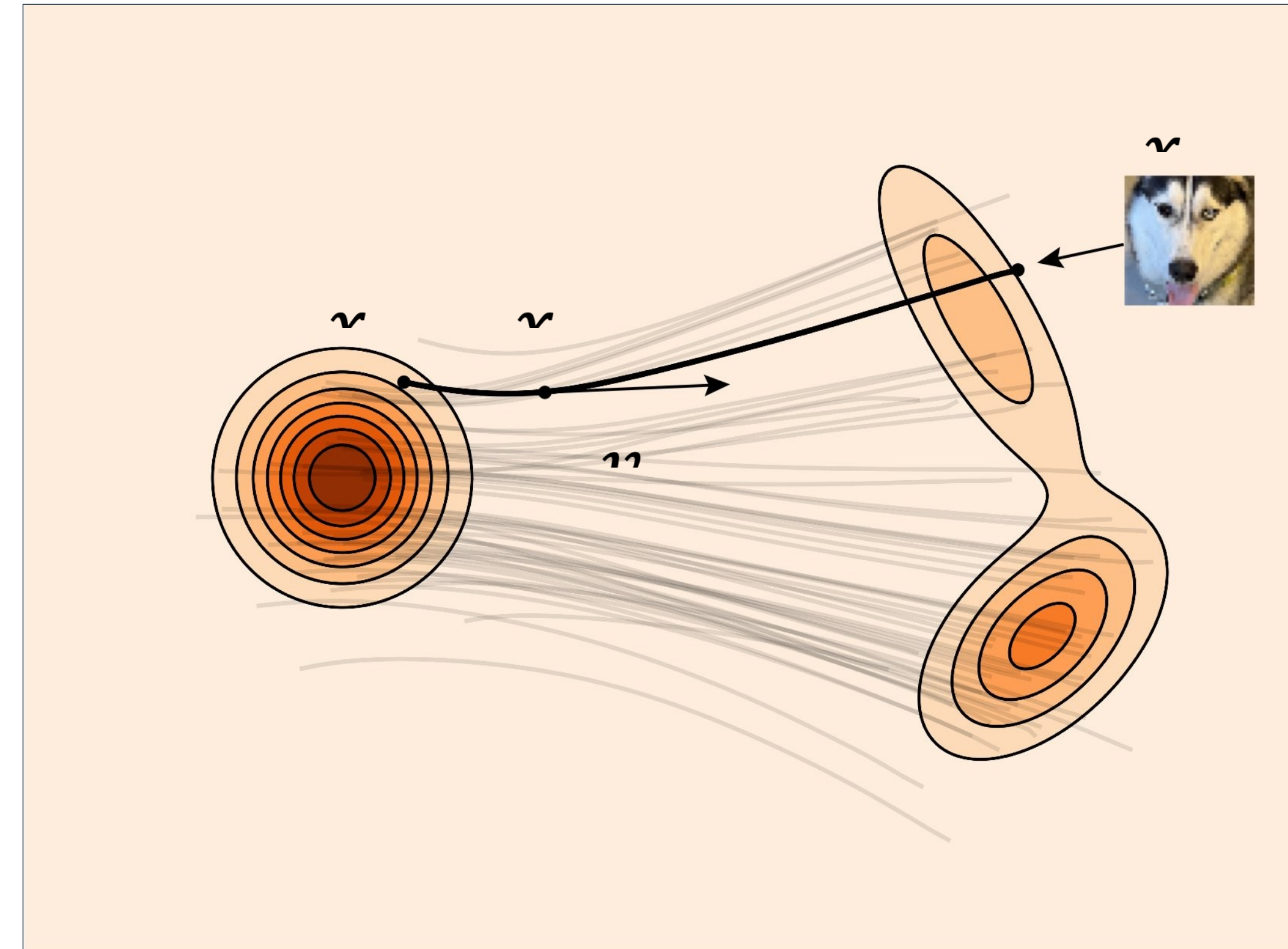


What about...

- DDPM
 - DDIM
 - VP/VE-ODE
 - **Flow matching**
 - EDM
 - ...?
-
- Different choices of timesteps, image scaling, etc.
 - Some better than others, but fundamentally all the same

Flow training

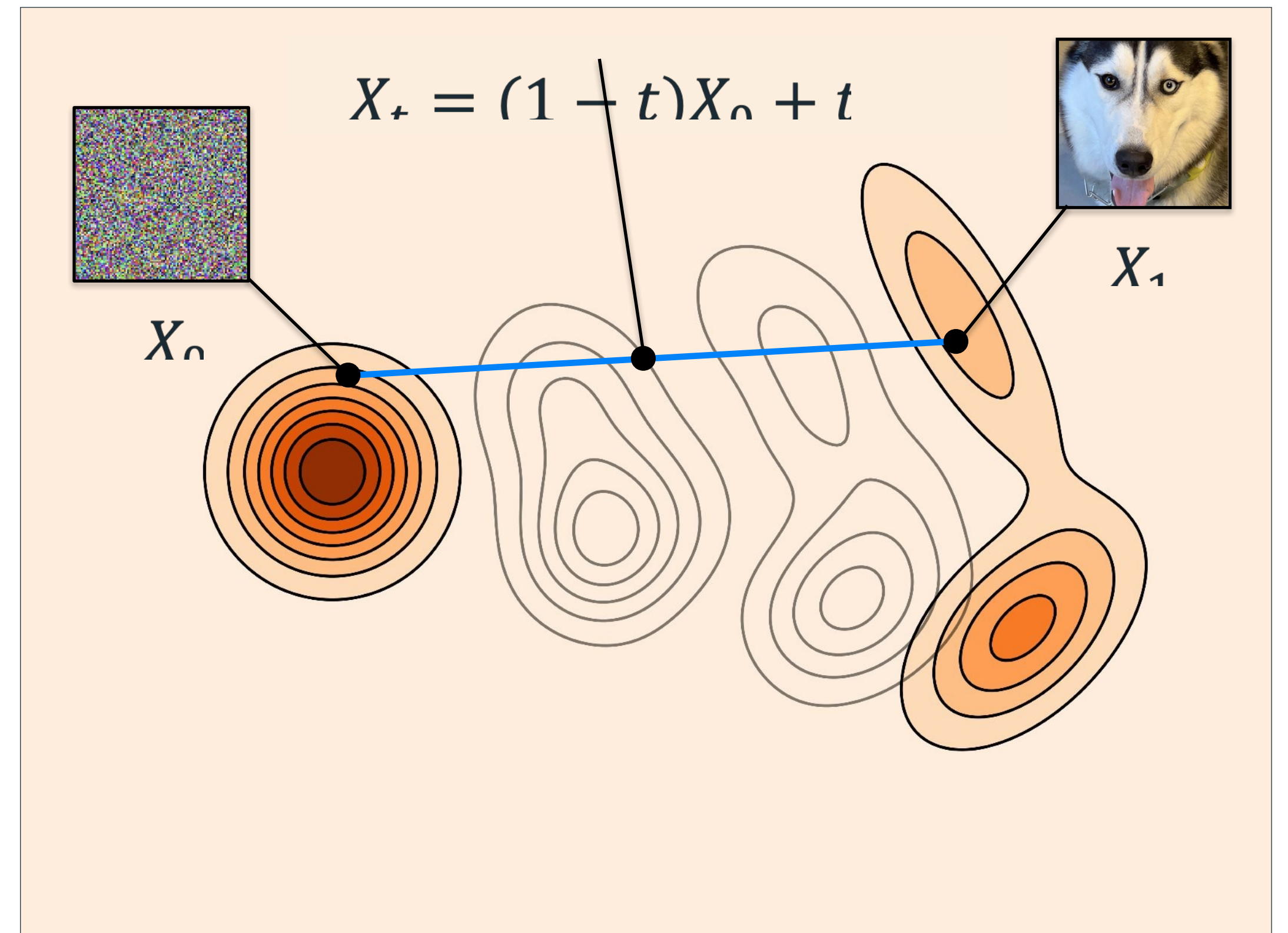
- For each data x_1
 - Sample some noise x_0
- Combine x_1 and x_0 in some way get x_t
- Now learn to predict the velocity at x_t
 - What is the velocity? Depends on how you got x_t



A Very Simple Way

Linear interpolation!

$$x_t = (1 - t)x_0 +$$



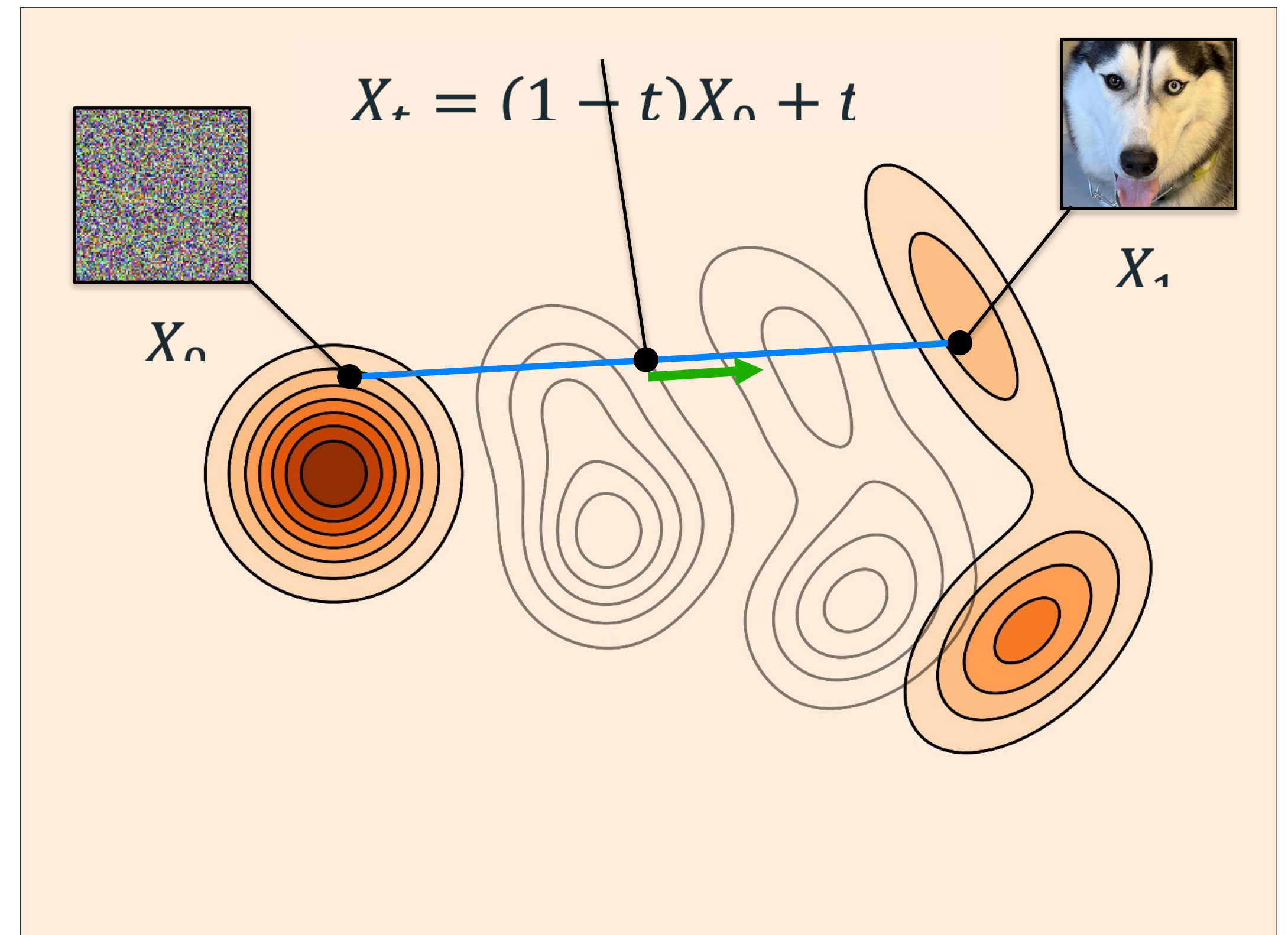
What is the velocity supervision?

$$x_t = \alpha_t x_0 + \beta_t$$

$$\mathbb{E}_{t, X_0, X_1} \| u_t^\theta(X_t) - (X_1 - X_0) \|$$

$$x_t = (1 - t)x_0 + t$$

$$\frac{dx_t}{dt} = -x_0 + x_1$$



*Conditioned on a single sample

Inside a Training Loop

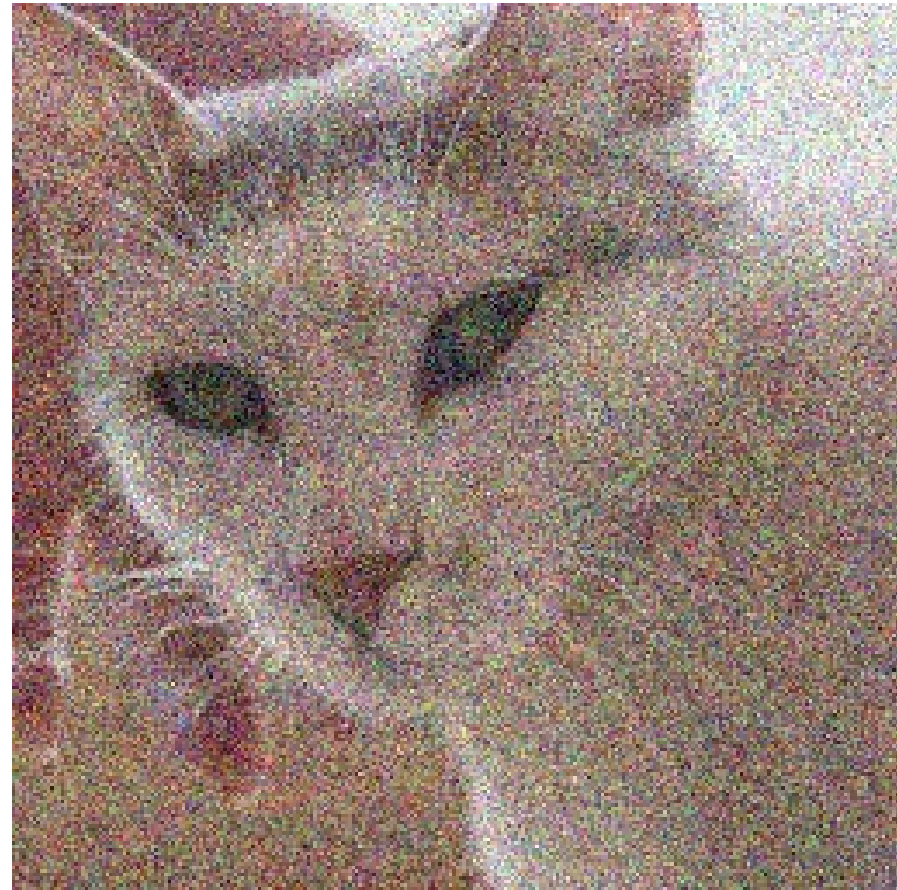
Flow Matching

```
x = next(dataset)
t = torch.rand(1) # Sample timestep (0,1)
noise = torch.randn_like(x) # Sample noise
x_t = (1-t) * noise + (t) * x # Get noisy x_t

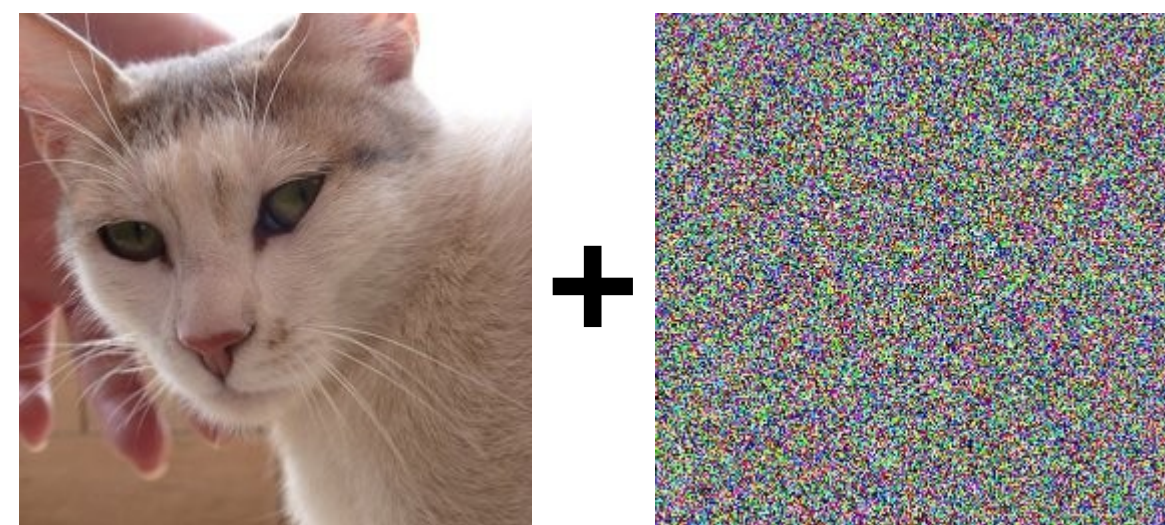
flow_pred = model(x_t, t) # Predict noise in x_t
flow_gt = x - noise # ground truth flow (w/ linear sched)
loss = F.mse_loss(flow_pred, flow_gt) # Update model
loss.backward()
optimizer.step()
```


Training U-Net Denoiser

x_t

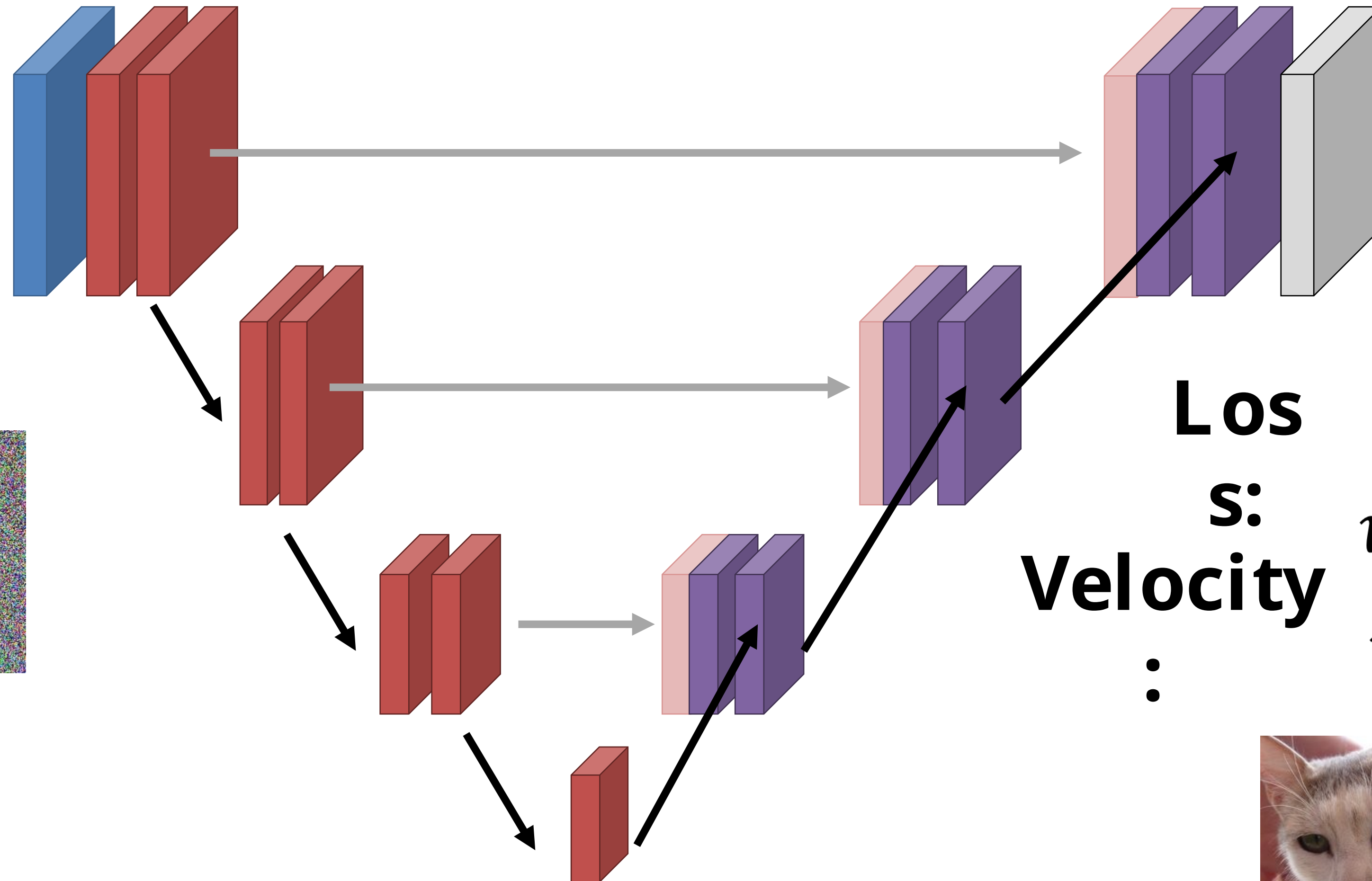


$\hat{v}$



x_{clean}

ϵ

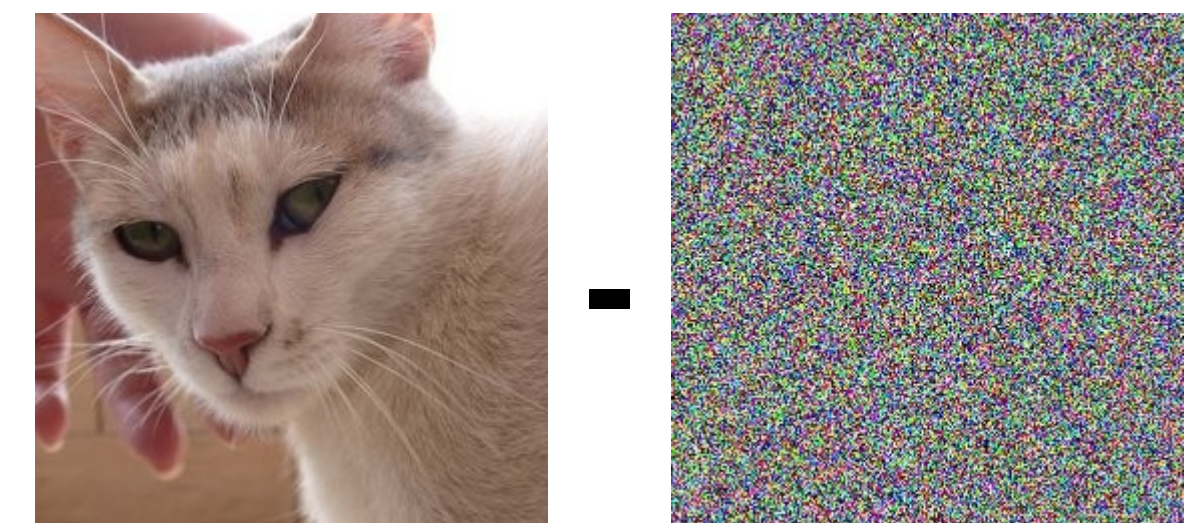


Loss:
S: Velocity
:

$$\|v - \hat{v}\|^2$$

$$v = \frac{x_{clean} - x_t}{(1 - t)}$$

$$v = x_{clean} - \epsilon$$



x_{clean}

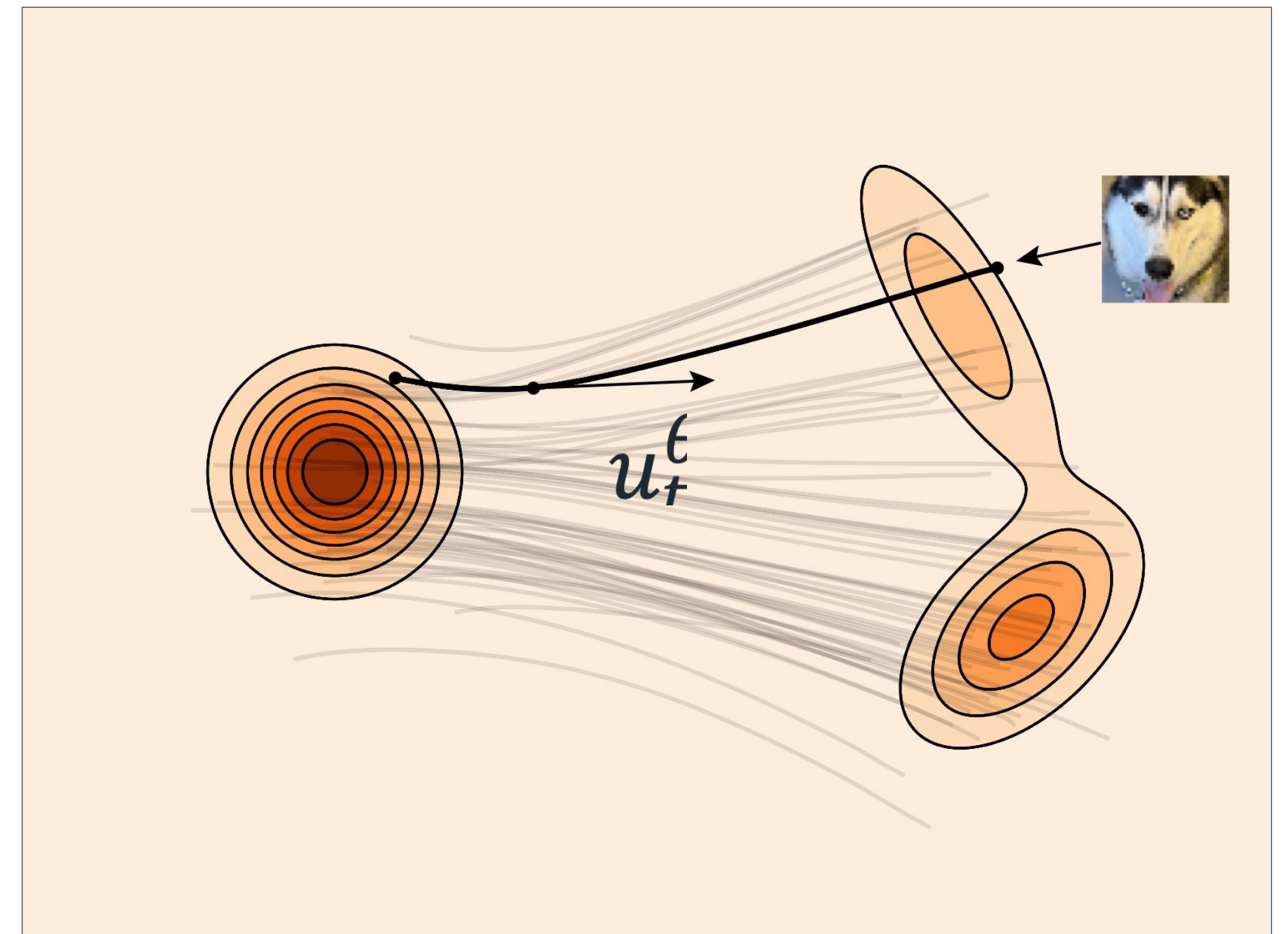
ϵ

Flow Matching

Test-time sampling

- Just take a small step in the velocity
- Use any ODE Solver, i.e. integration you like, like Euler integration:

$$x_{t+\Delta t} = x_t + \Delta t \cdot \left. \frac{dx}{dt} \right|_{x_t, t}$$



Sample
from $X_0 \sim p$

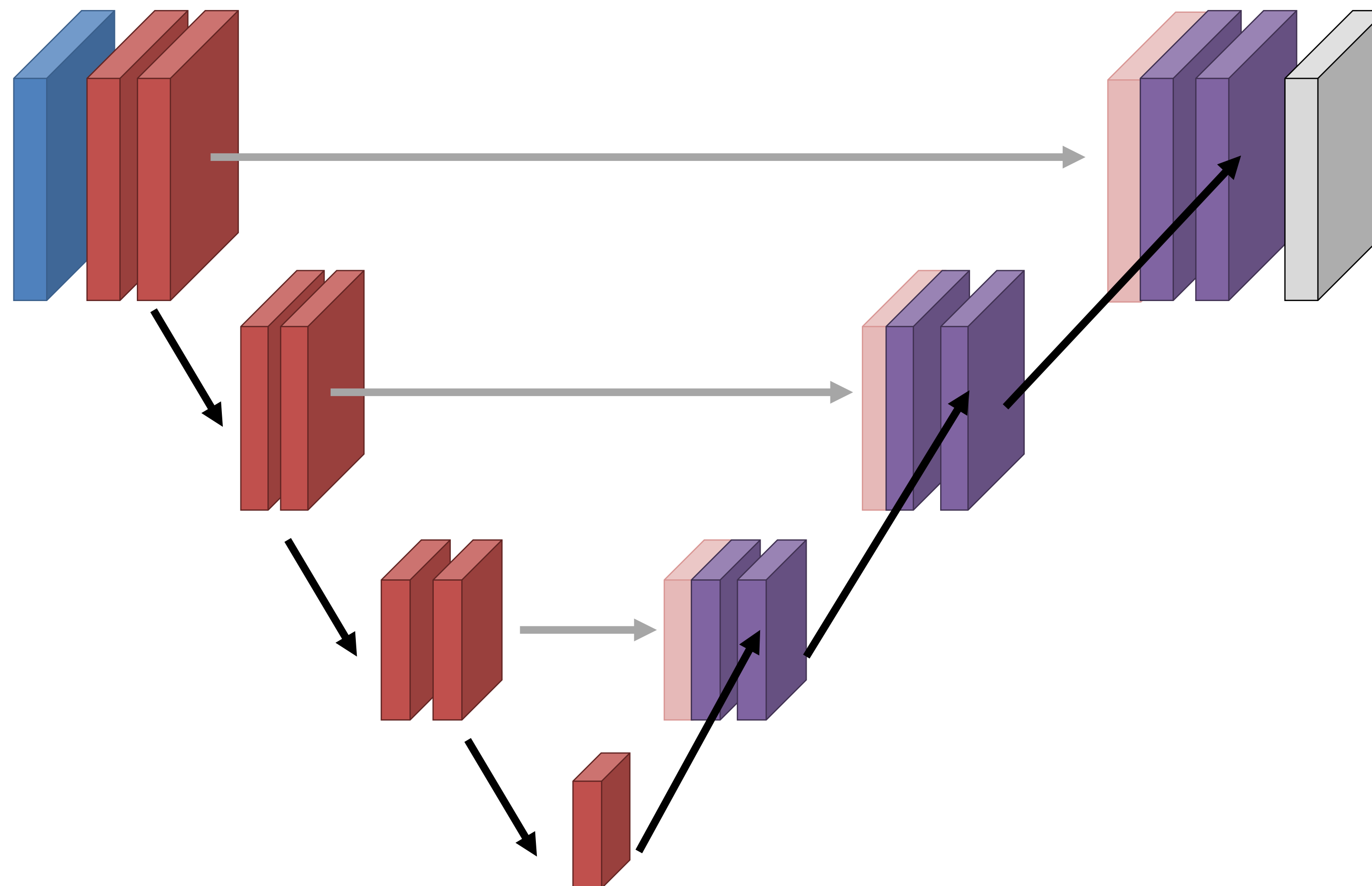
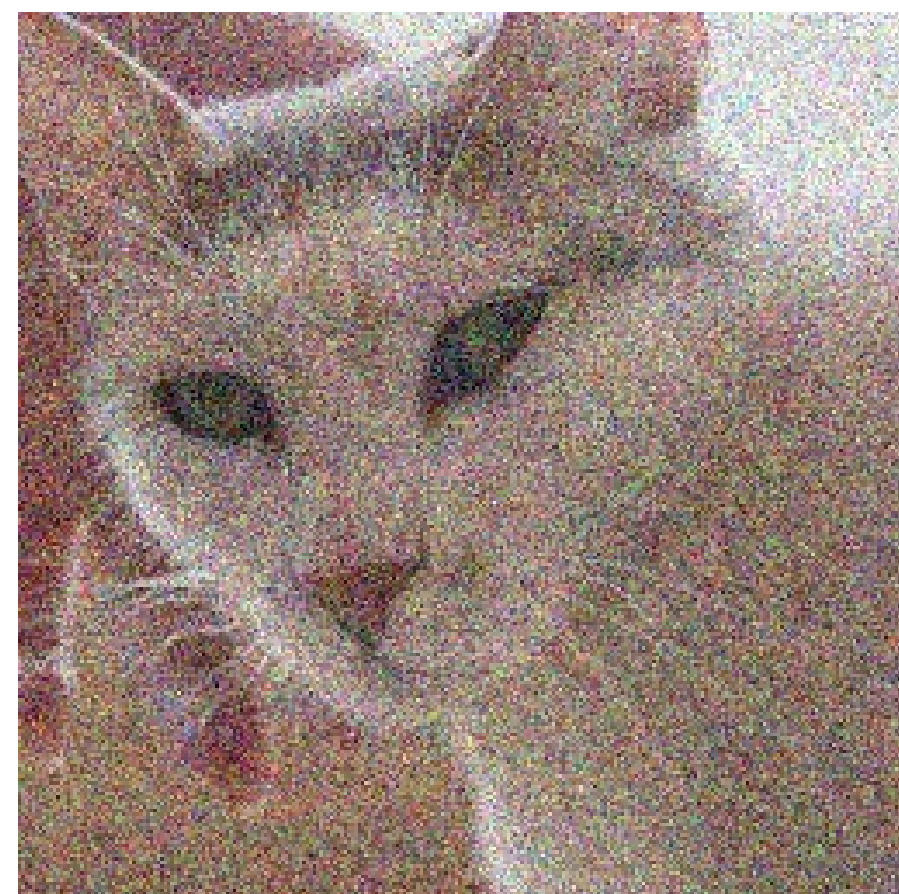
Inside Sampling Loop

```
steps = 100
dt = 1.0 / steps
for i in range(steps):
    velocity = model(x_t, t) # Predict velocity at x_t
    x_t = x_t + dt * velocity # Step in velocity
```

x_t

U-Net Denoiser at inference

$\hat{v}$



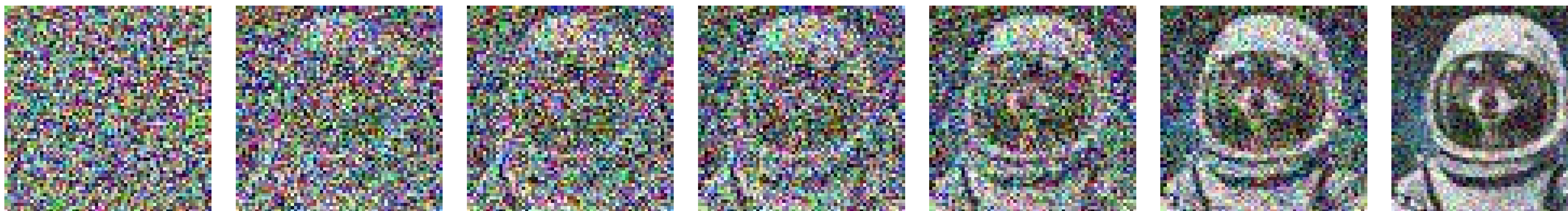
Flow Matching

Training Time:

$$x_t = (1 - t)x_0 + tx_1$$

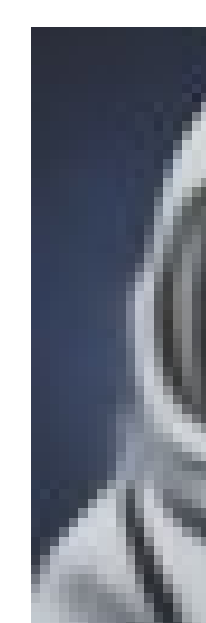
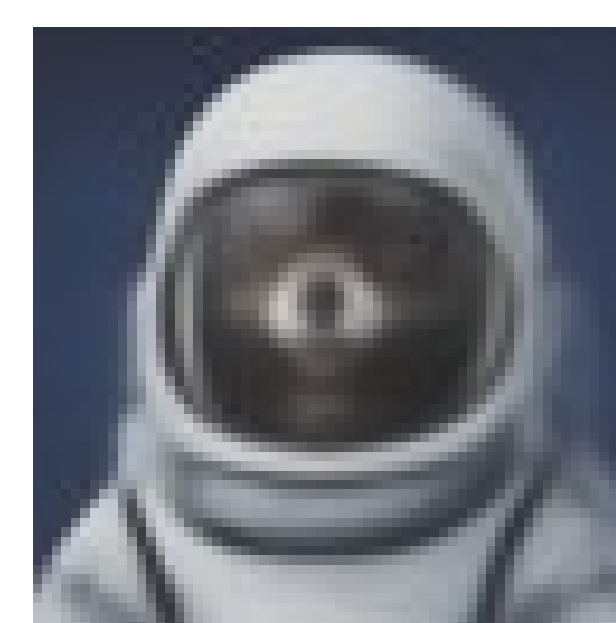
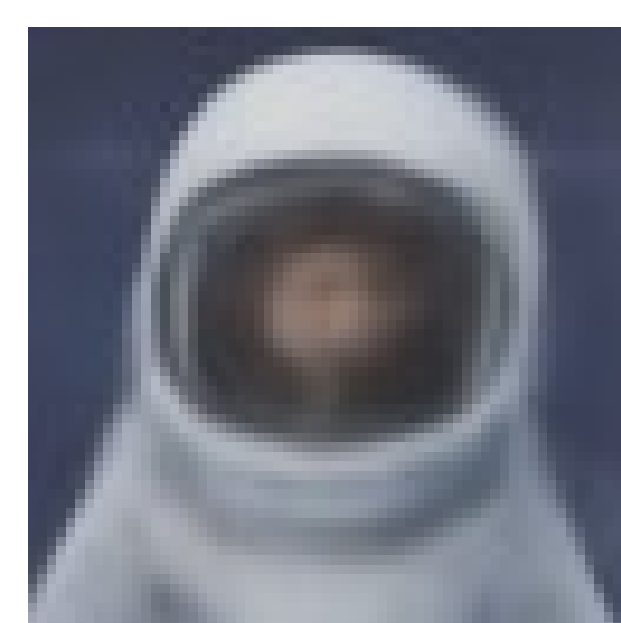
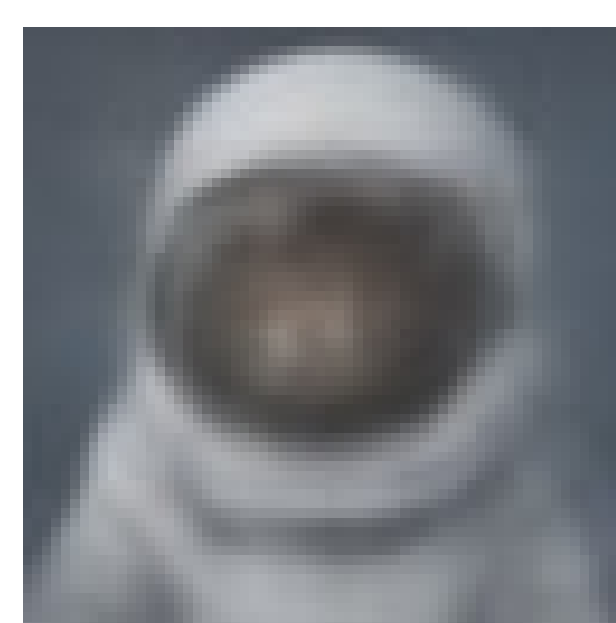
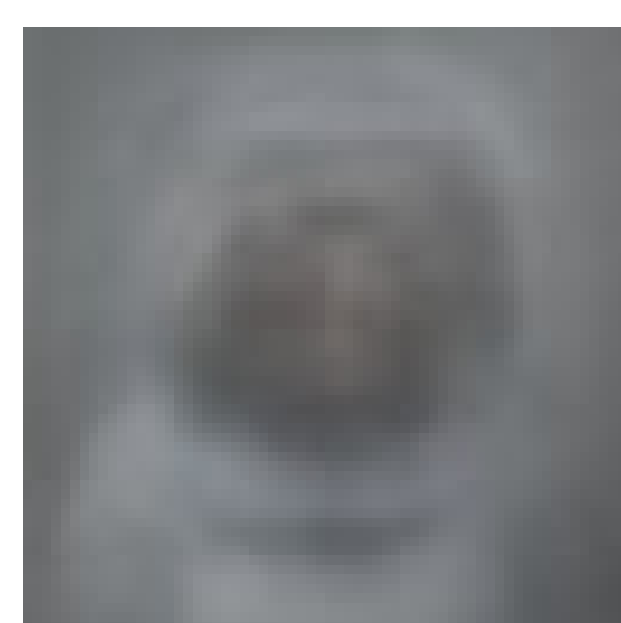
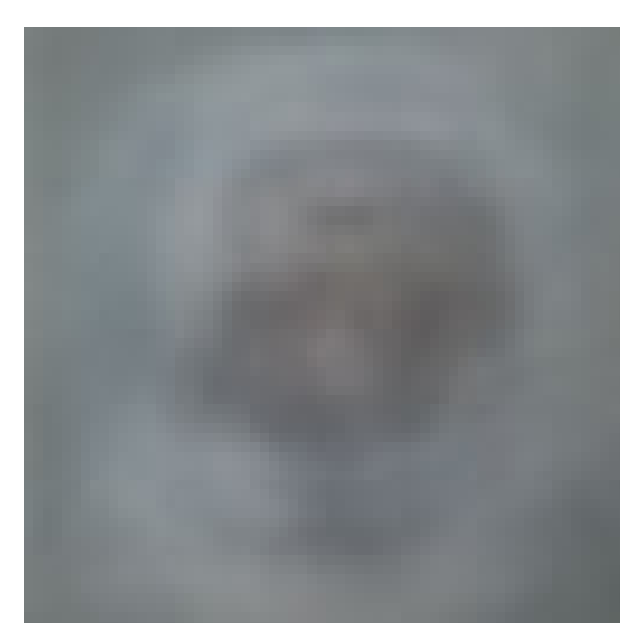
Inference

Time: $x_{t+h} = x_t + hv_\theta(x_t, t, c)$



denoise

blend



$$\hat{x}_1 = x_t + (1 - t)v_\theta(x_t, t, c)$$

Curious Property #1

- In ML, you want to always train the same data you test on
- But in Diffusion/Flow Models, they are different:

Training Time:

$$x_t = (1 - t)x_0 + tx_1$$

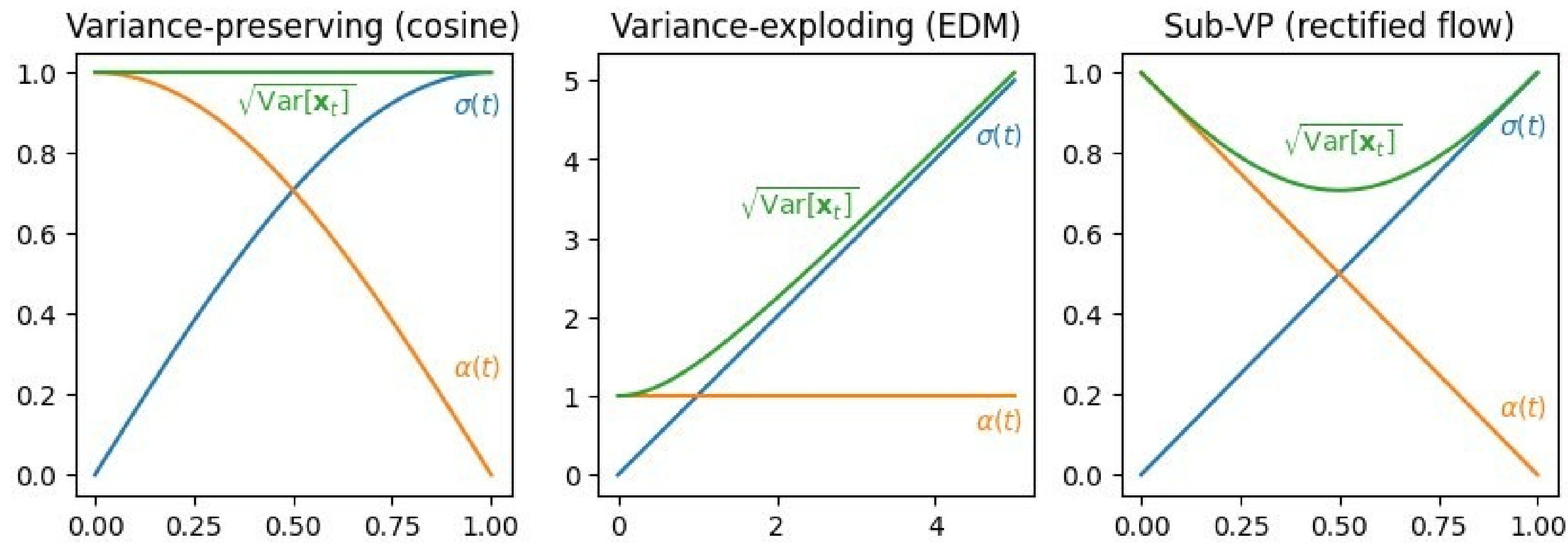
Inference

Time: $x_{t+h} = x_t + h v_\theta(x_t, t, c)$

- Not great, but seems to still work, somewhat

Other options lead to prior works

- Other choices: $\mathbf{x}_t = \alpha_t \mathbf{x}_0 + \beta_t \epsilon_t$
- Preserve variance (VP-ODE) - DDPM
- Exploding variance (VE-ODE) - Score Matching/DDIM
- Linear interpolation (Flow Matching, Rectified Flow)



Why does Flow Matching work?

FM $\rightarrow$ predict the velocity conditioned on a single sample

Marginal vs. Conditional Flow

- What we want is the flow (velocity field) that takes samples from p_0 to p_1 when integrated (called Marginal Flow)
 - e.g. Find the **best** noise that would flow to each training image
- But what we did was to train flow for each sample (Conditional Flow)

Turns out Gradient is the same!

- **Flow Matching loss:**

$$\mathcal{L}_{\text{FM}}(\theta) = \mathbb{E}_{t, X_t} \| u_t^\theta(X_t) - u_t \|$$

We can't do this bc we don't know what ground truth flow is

- **Conditional Flow Matching loss:**

$$\mathcal{L}_{\text{CFM}}(\theta) = \mathbb{E}_{t, X_1, X_t} \| u_t^\theta(X_t) - u_t(X_t) \|$$

Theorem: Gradient of the losses are equivalent,

$$\nabla_{\theta} \mathcal{L}_{\text{FM}}(\theta) = \nabla_{\theta} \mathcal{L}_{\text{CFM}}(\theta)$$

Why does this work?

- High level: You can average conditional flow (marginalization)

$$\begin{aligned} u_t(x) &= \mathbb{E}[u_t(X_t|X_1)|X_t = x] \\ &= \int u_t(x|x_1)p(x_1|X_t = x) \end{aligned}$$

Marginal flow
(what we want)

Conditional flow
(easy to obtain from each sample)

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

Bayes rule with notation:

$$p(x_1) = a(x_1)$$

It's all just weighted average

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

$$u_t(x_t) = \sum \underbrace{u_t(x_t|x_1)}_{\text{Path from } x_1 \text{ to } x_t} \underbrace{\frac{p_t(x_t|x_1)}{p_t(x)}}_{\text{Path weight (how relevant } x_1 \text{ is to } x_t)} q$$

Weight of sample $x_0 = 1$

Marginal flow

Just a weighted average of the flow to each data sample!!!!

You can actually do this non-parametrically.

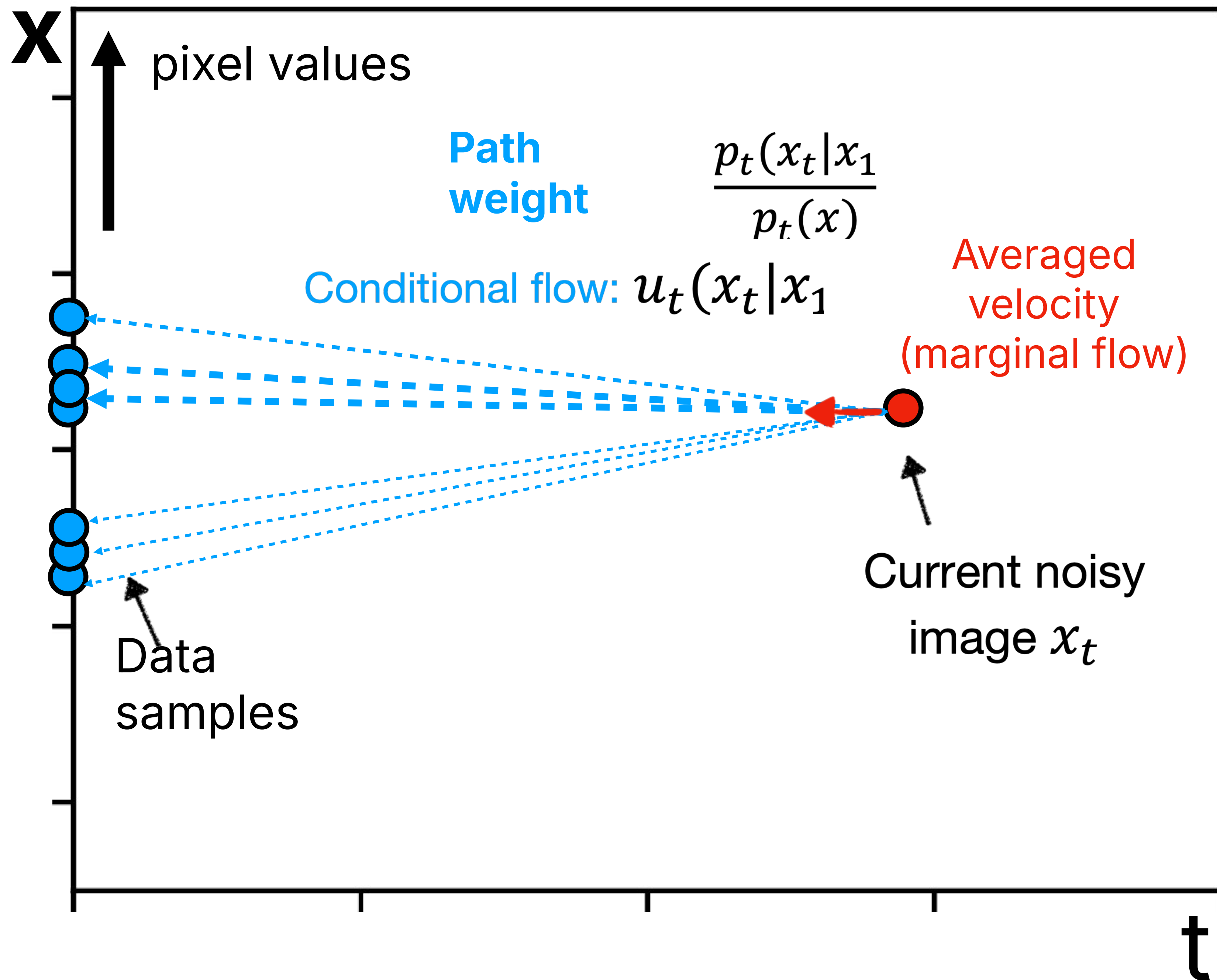


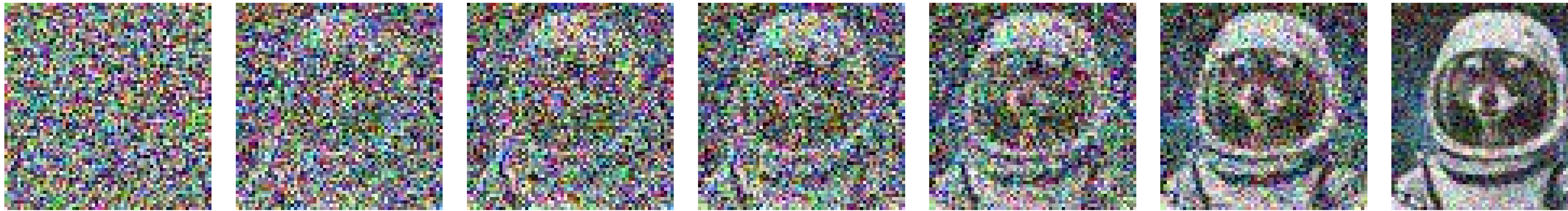
Illustration of the analytic solution

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

$$u_t(x_t) = \sum \underbrace{u_t(x_t | x_1)}_{\text{Path from } x_1 \text{ to } x_t} \underbrace{\frac{p_t(x_t | x_1)}{p_t(x)}}_{\text{Path weight (how relevant } x_1 \text{ is to } x_t)} q(x_1)$$

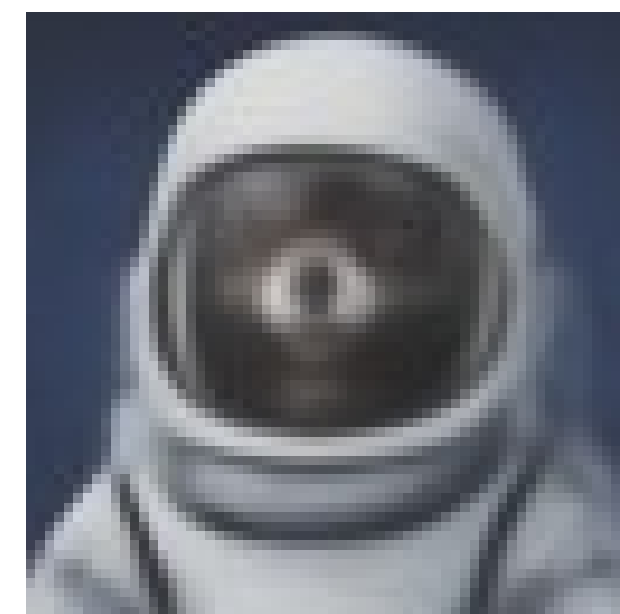
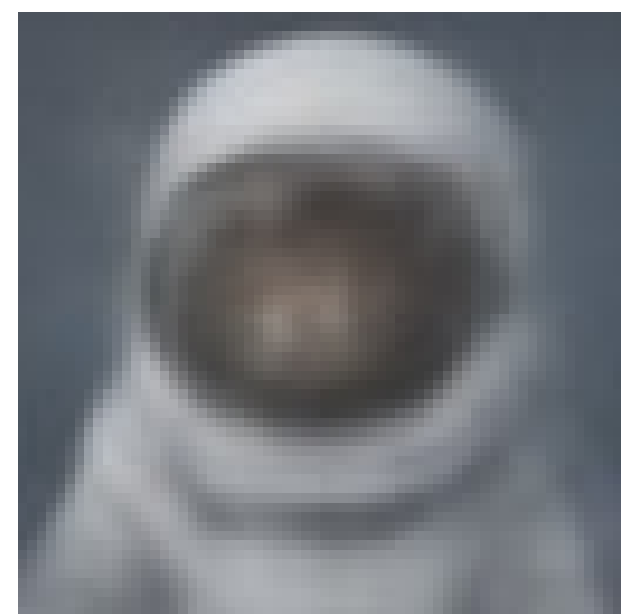
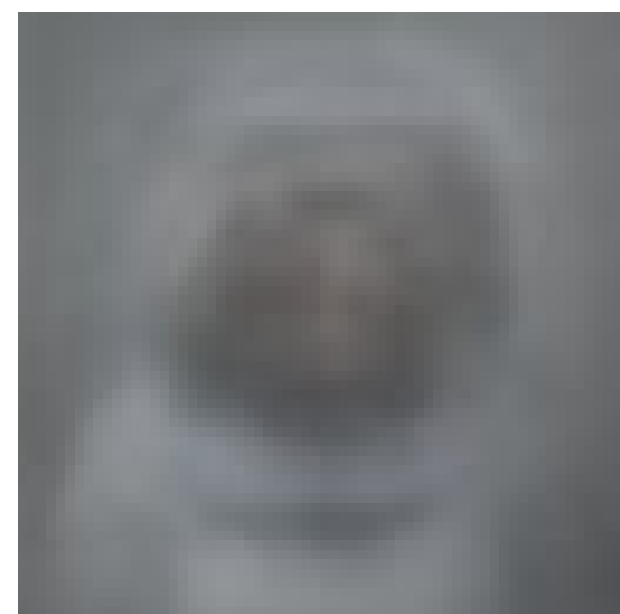
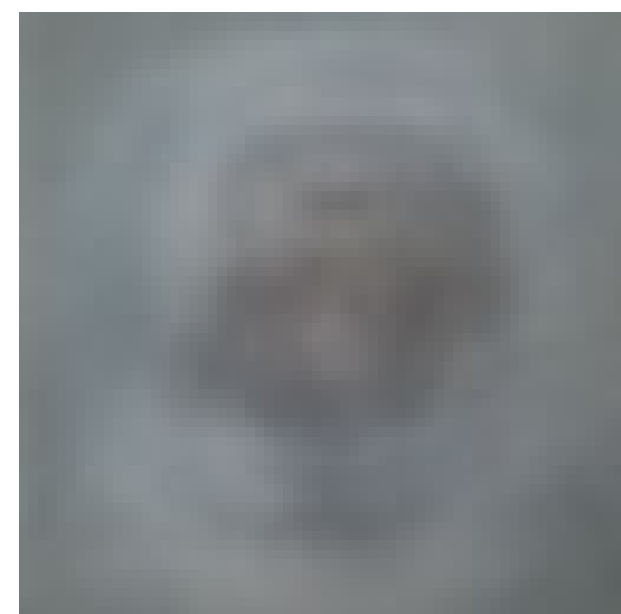
Marginal flow

Now we can explain why x_t start out so
flurry?



denoise

blend

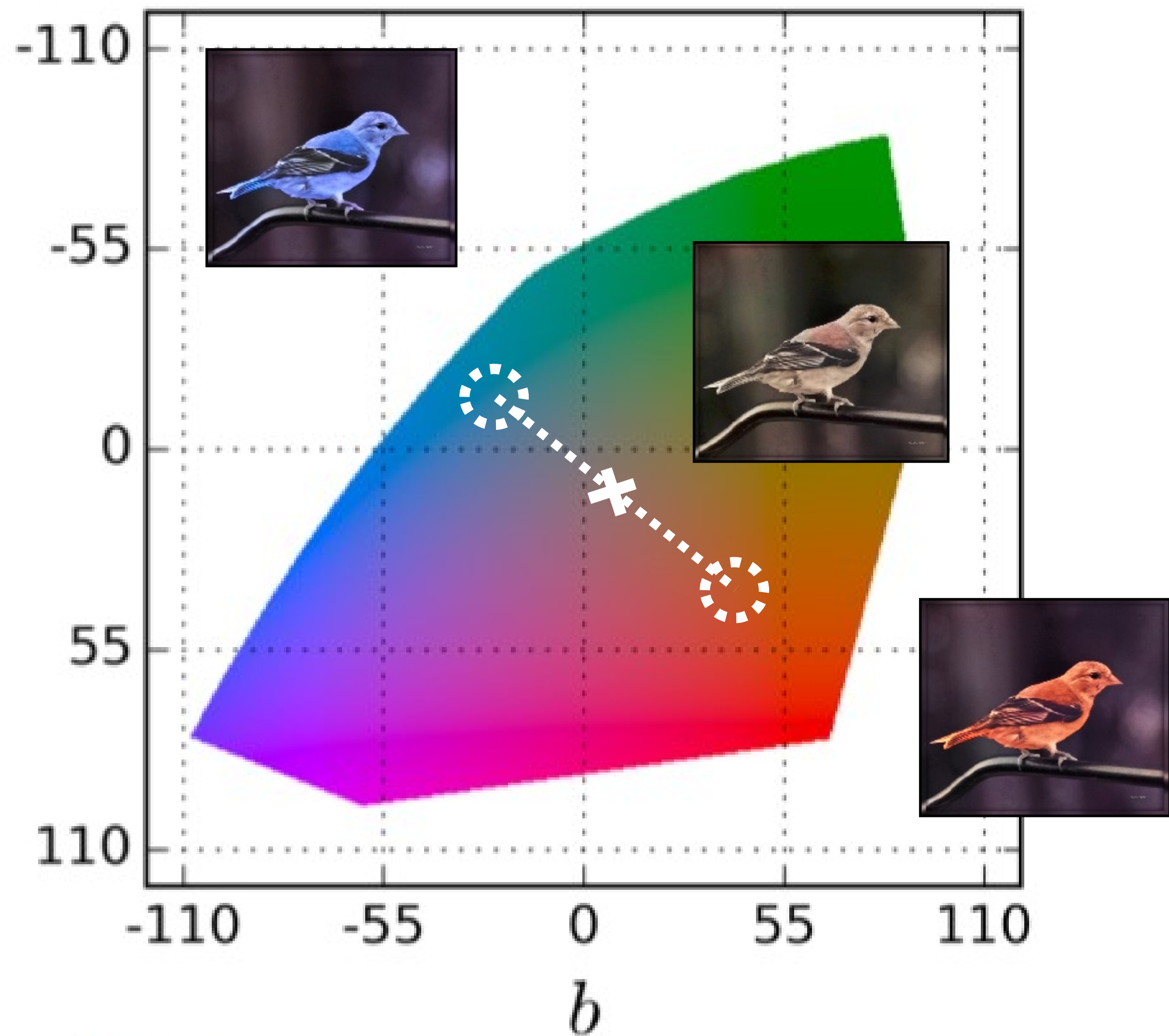


$$\hat{x}_1 = x_t + (1 - t)v_\theta(x_t, t, c)$$



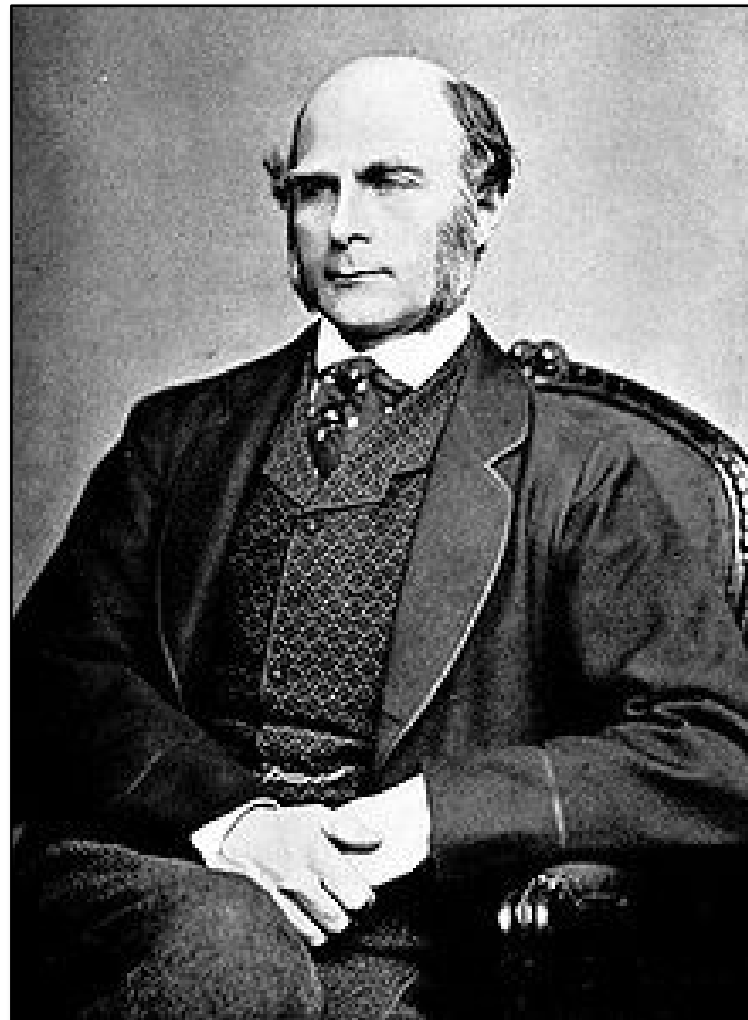
Our old friend L2 is the problem!

a



$$L_2(\hat{\mathbf{Y}}, \mathbf{Y}) = \frac{1}{2} \sum_{h,w} \|\mathbf{Y}_{h,w} - \hat{\mathbf{Y}}_{h,w}\|_2^2$$

Image Composites



Sir Francis
Galton
1822-1911



Multiple Individuals



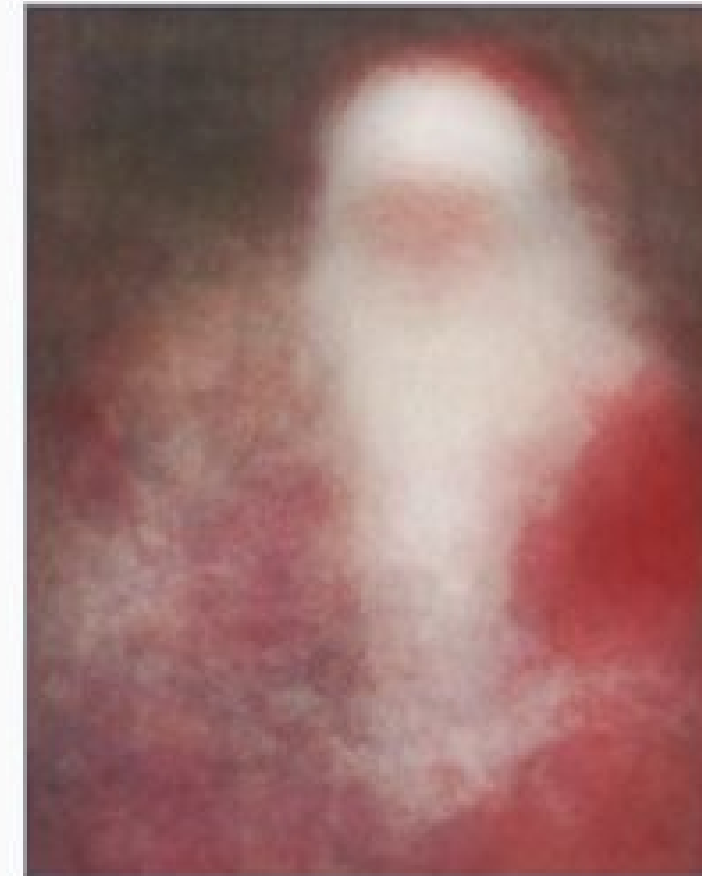
Composite

[Galton, "Composite Portraits", Nature, 1878]

“100 Special Moments” by Jason Salavon



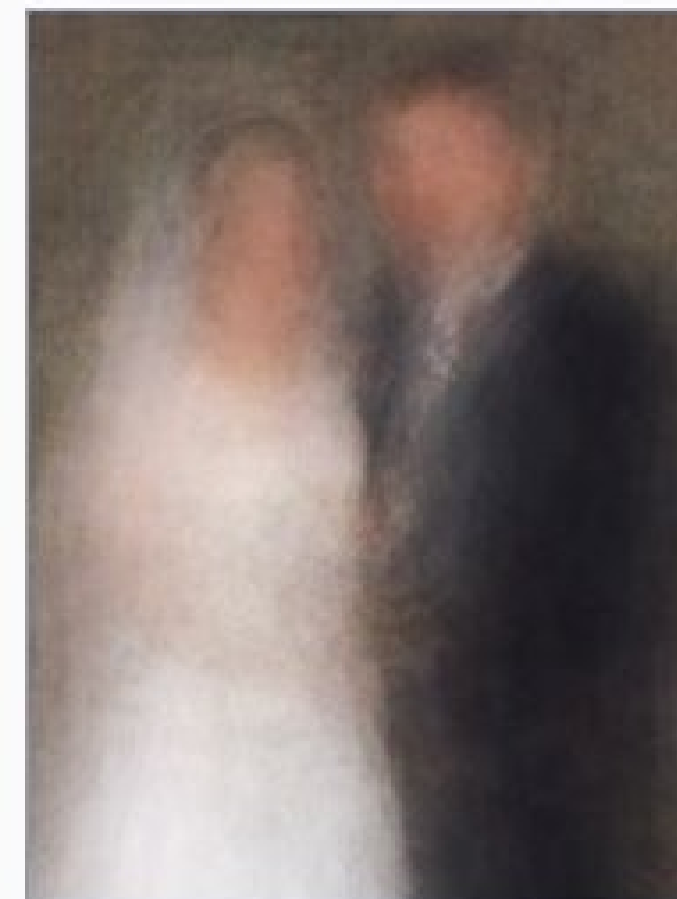
Little Leaguer



Kids with Santa



The Graduate



Newlyweds

Why
blurry?

Sum-population means



AUSTRIA



AFGHANISTAN



ARGENTINA



BURMA (MYANMAR)



GERMANY



GREECE



CAMBODIA



ENGLAND



ETHIOPIA



FRANCE



IRAQ



IRELAND



MONGOLIA



PERU



POLAND



PUERTO RICO

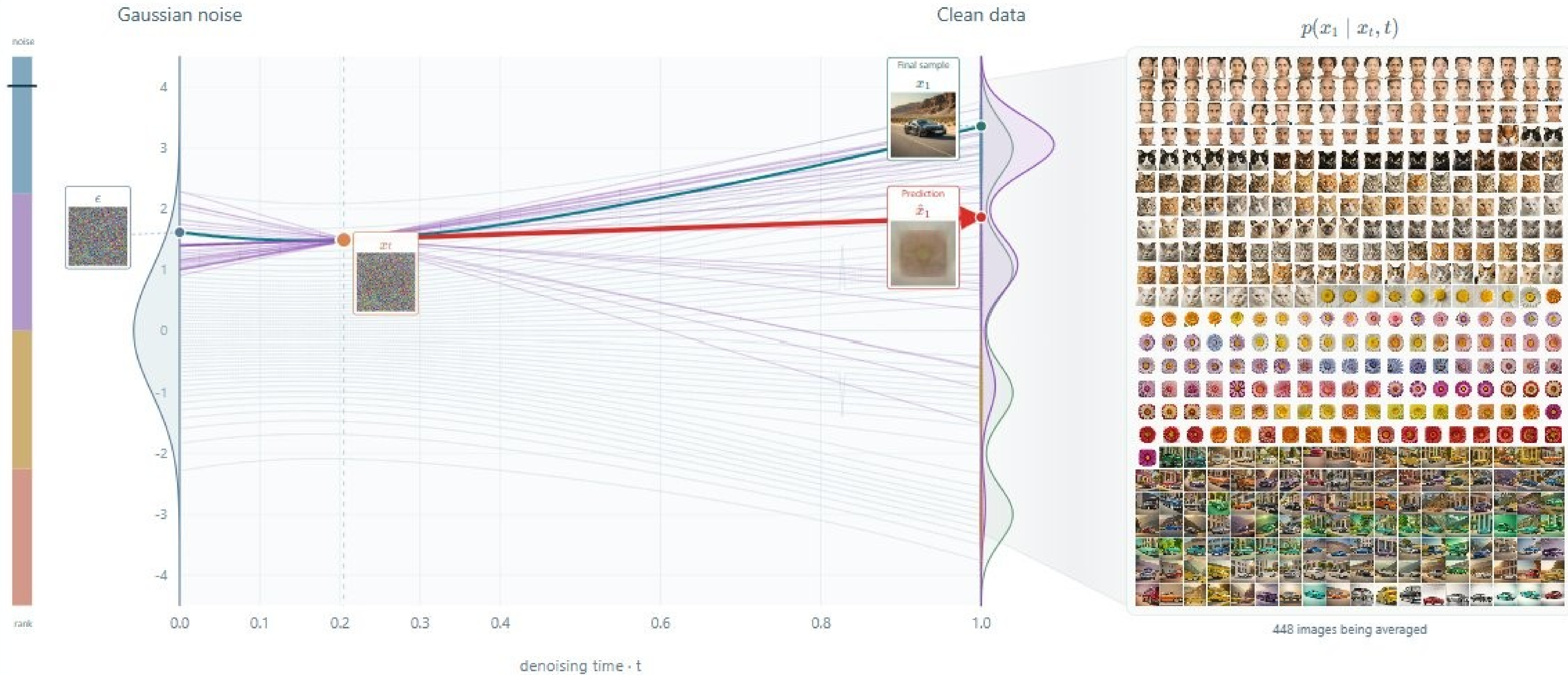


UZBEKISTAN



AFRICAN AMERICAN

The full intuition!



Donghoon Ahn's visualization: <https://sunovivid.github.io/blog/2026/flow-mat-ry/>

Important Caveat

- **One can minimize the diffusion objective (marginal flow) non-parametrically and perfectly minimize the loss.**
- But there is no learning! No ability to generate new images!
- i.e. Exactly minimizing this objective does not guarantee interpolation/compositionally, learning of the image manifold!
- Parametrizing it with neural networks results in magic smoothing to generate new images and interpolate between them. Exactly what makes this possible still active area of research

Diffusion/Flow Models tend to
model important correlations





“Person holding a laptop”



“Person holding birthday cake”



“Person offering tea”



“Person holding a watermelon”



“Person holding a self-portrait”



“Person holding a painting of himself”

Practical Inference Tricks

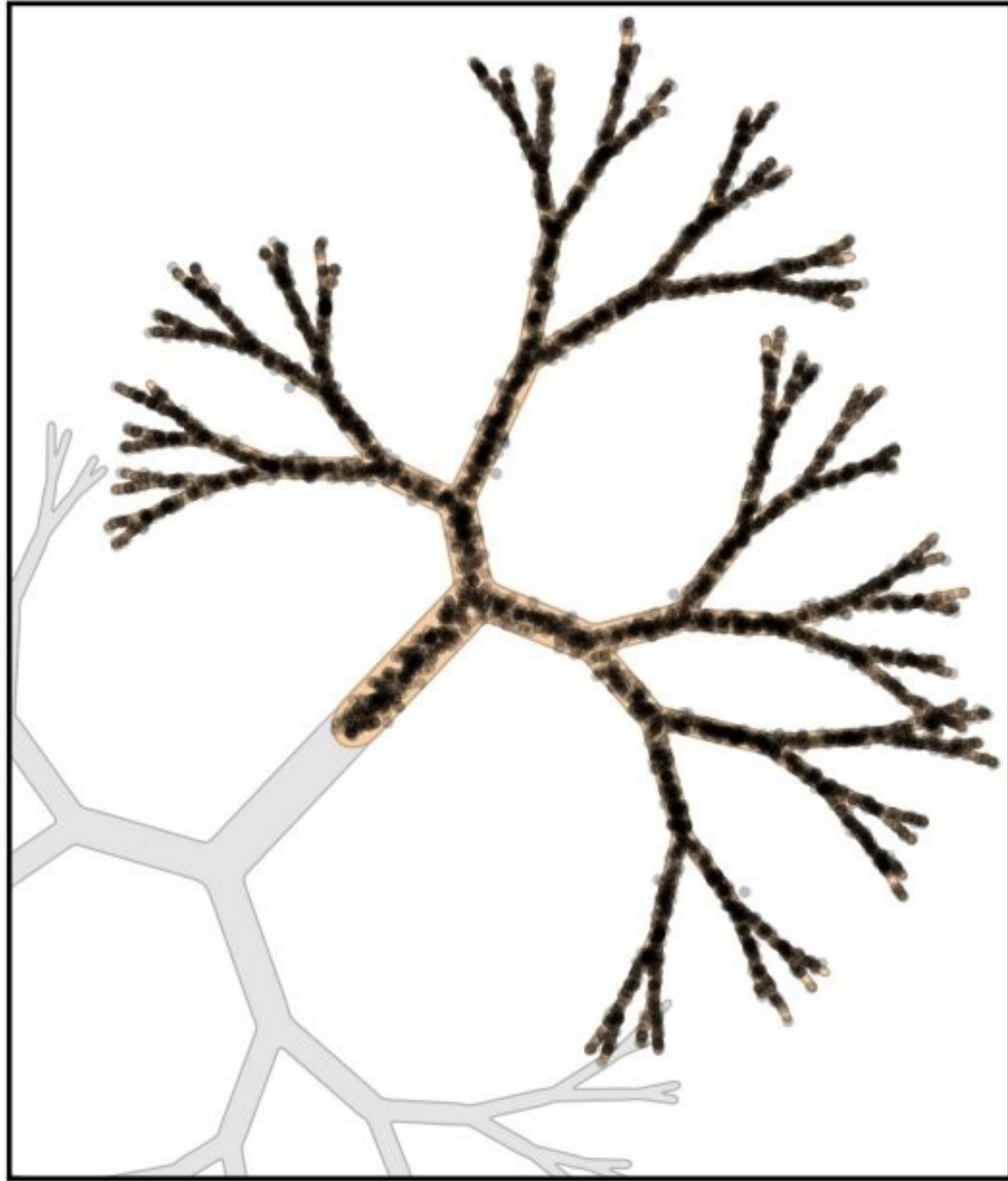
Great! Let's collect a billion images, train,
and generate...



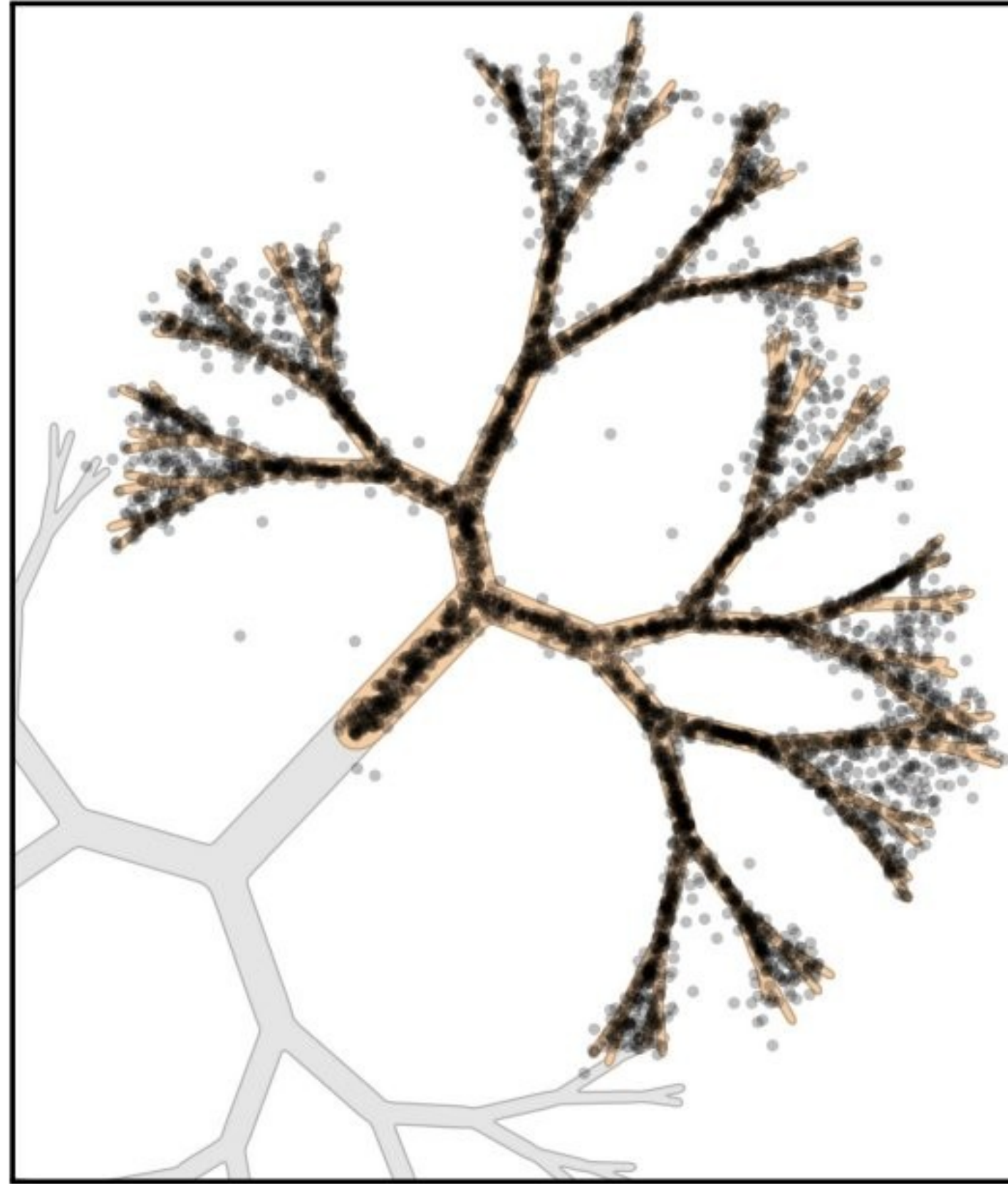
An old man in a colorful smock splattering paint alongside a young man in an oversized t-shirt and goggles.

... oh.

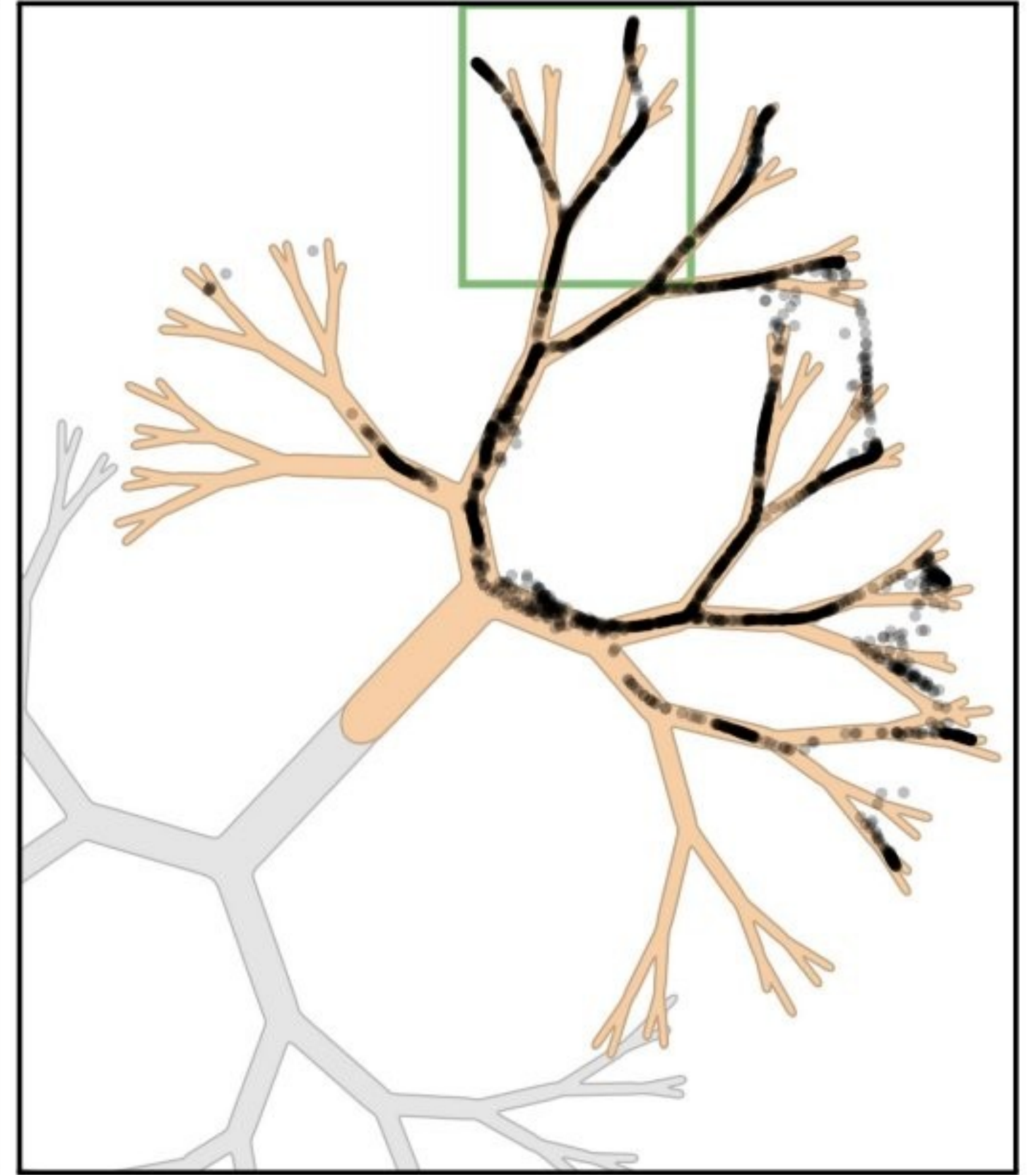
What went wrong?



(a) Ground truth



(b) No guidance



(c) Classifier-free
guidance

A photo of a raccoon wearing an astronaut helmet

guidance off



guidance on



Two Approaches

Original

Classifier
Guidance

Guide with a
pretrained classifier.

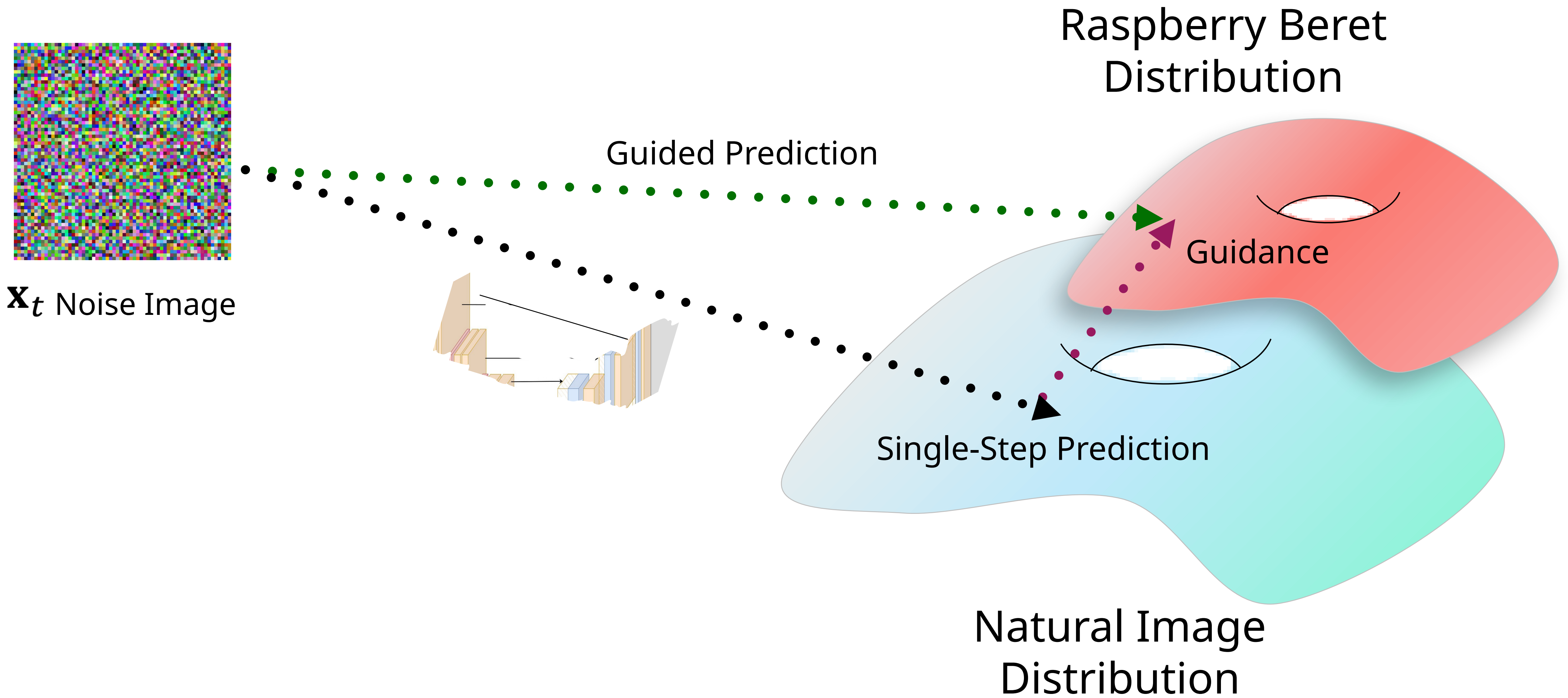
Current

Classifier-Free
Guidance

Guide a diffusion
model with itself.

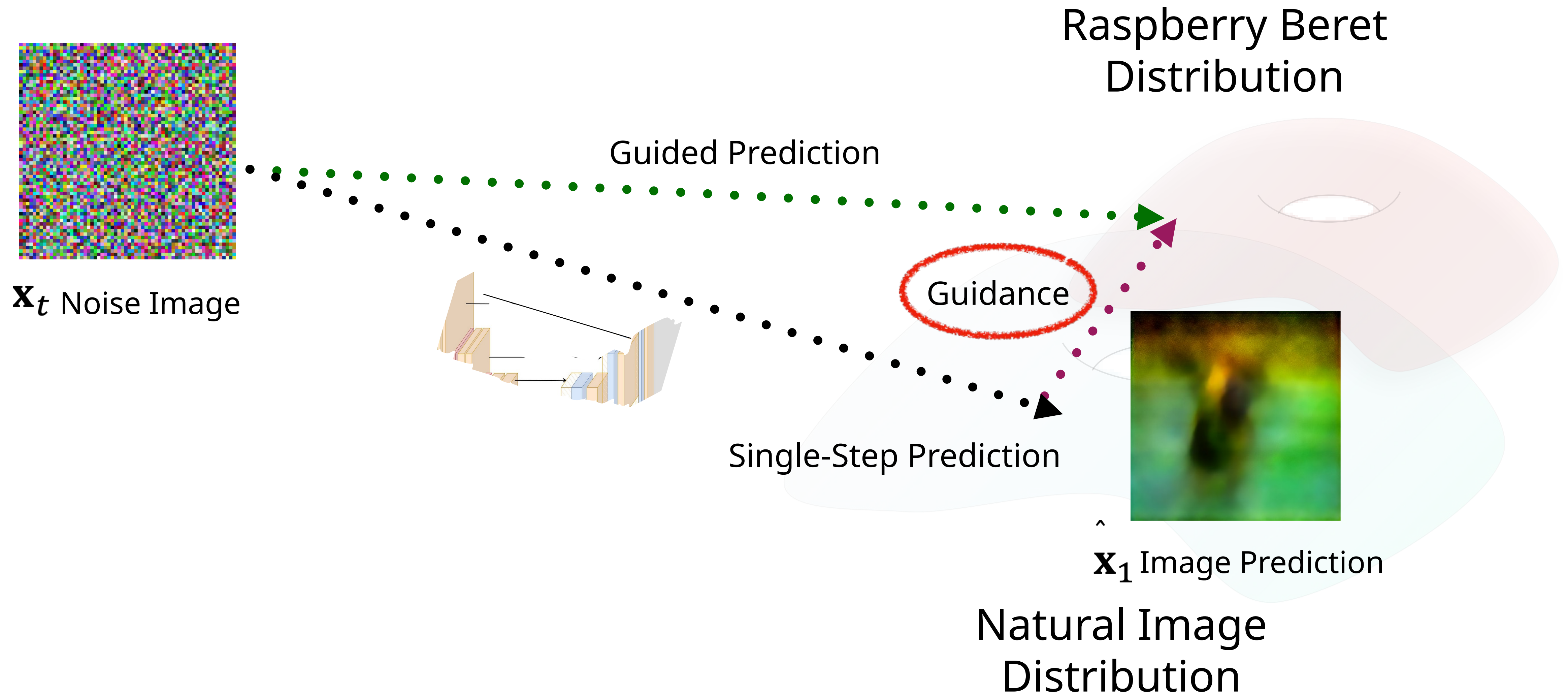
Diffusion Guidance

Push Toward a Conditional Mode



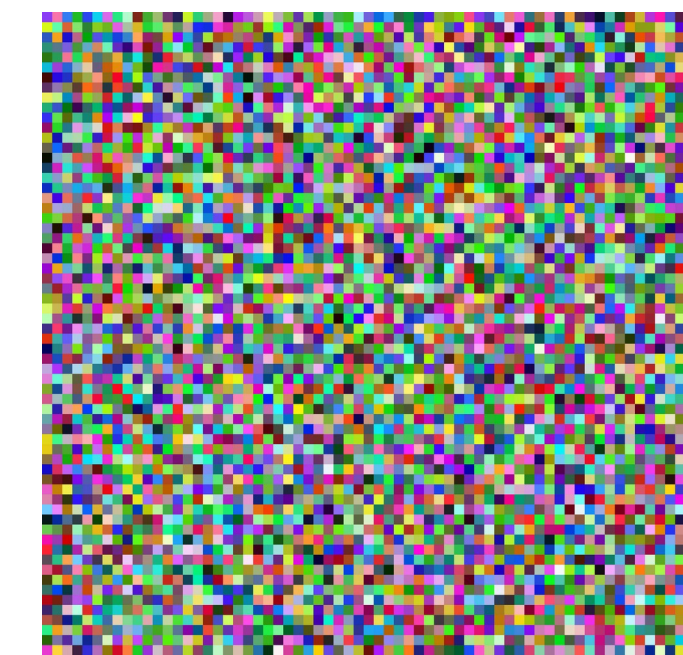
Classifier Guidance

Using a Pretrained Classifier



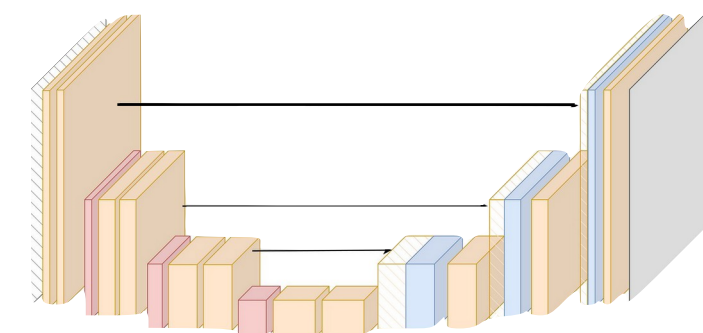
Classifier Guidance

Using a Pretrained Classifier



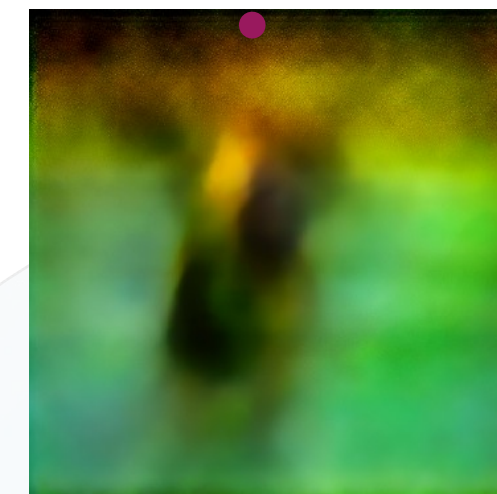
$\mathbf{x}_t$ Noise Image

...

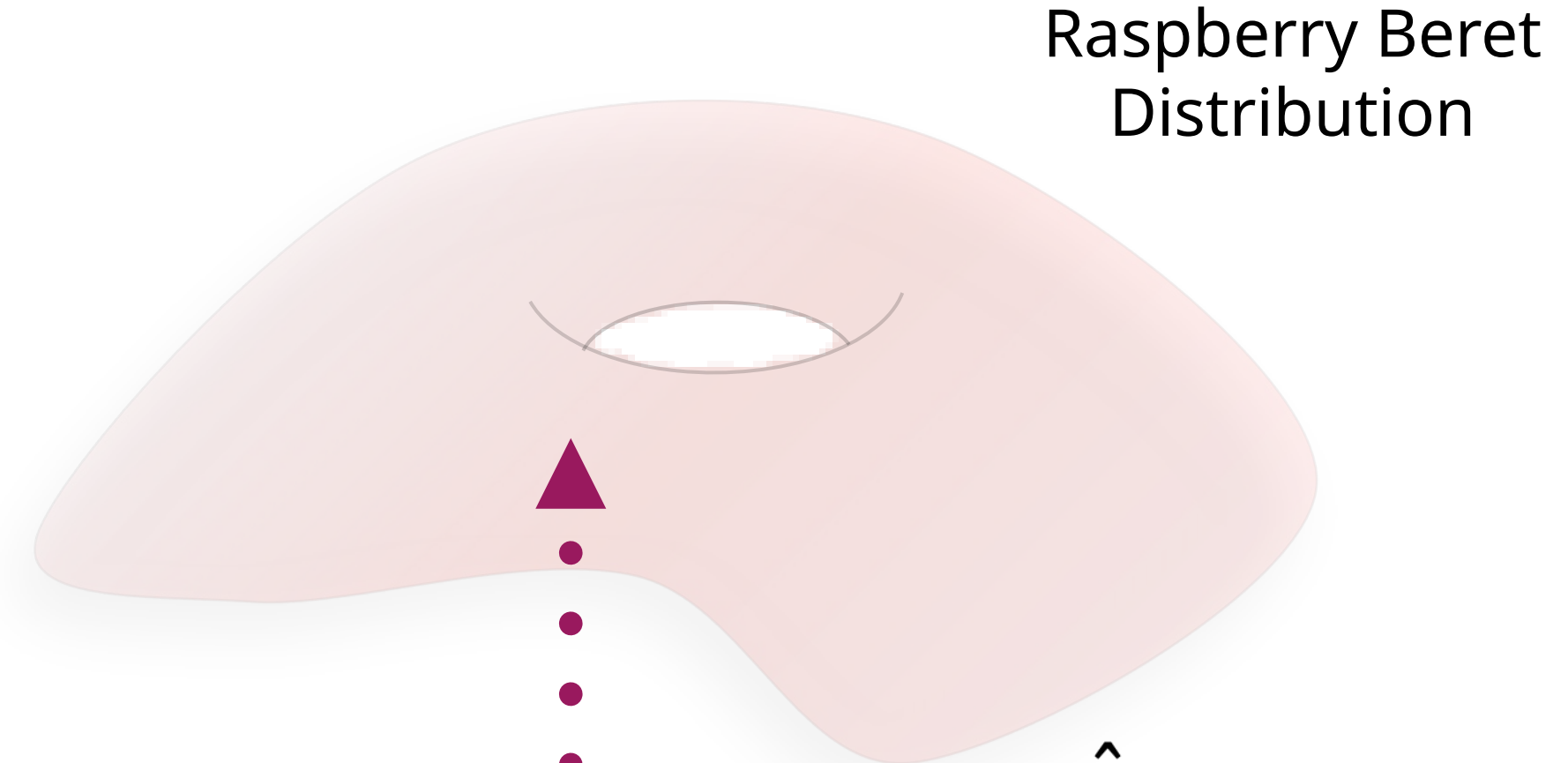


Diffusion Model

...



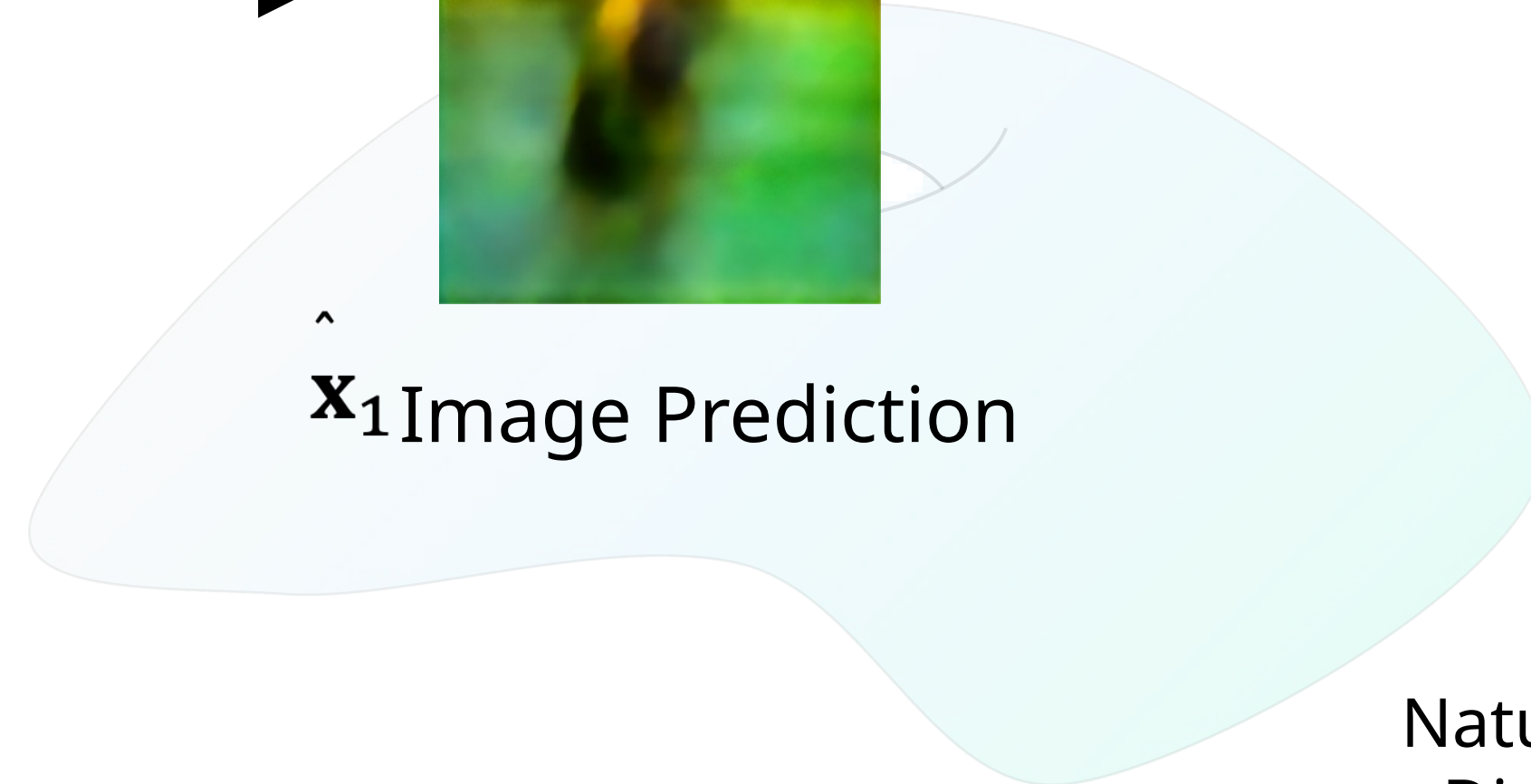
$\hat{\mathbf{x}}_1$ Image Prediction



Raspberry Beret
Distribution

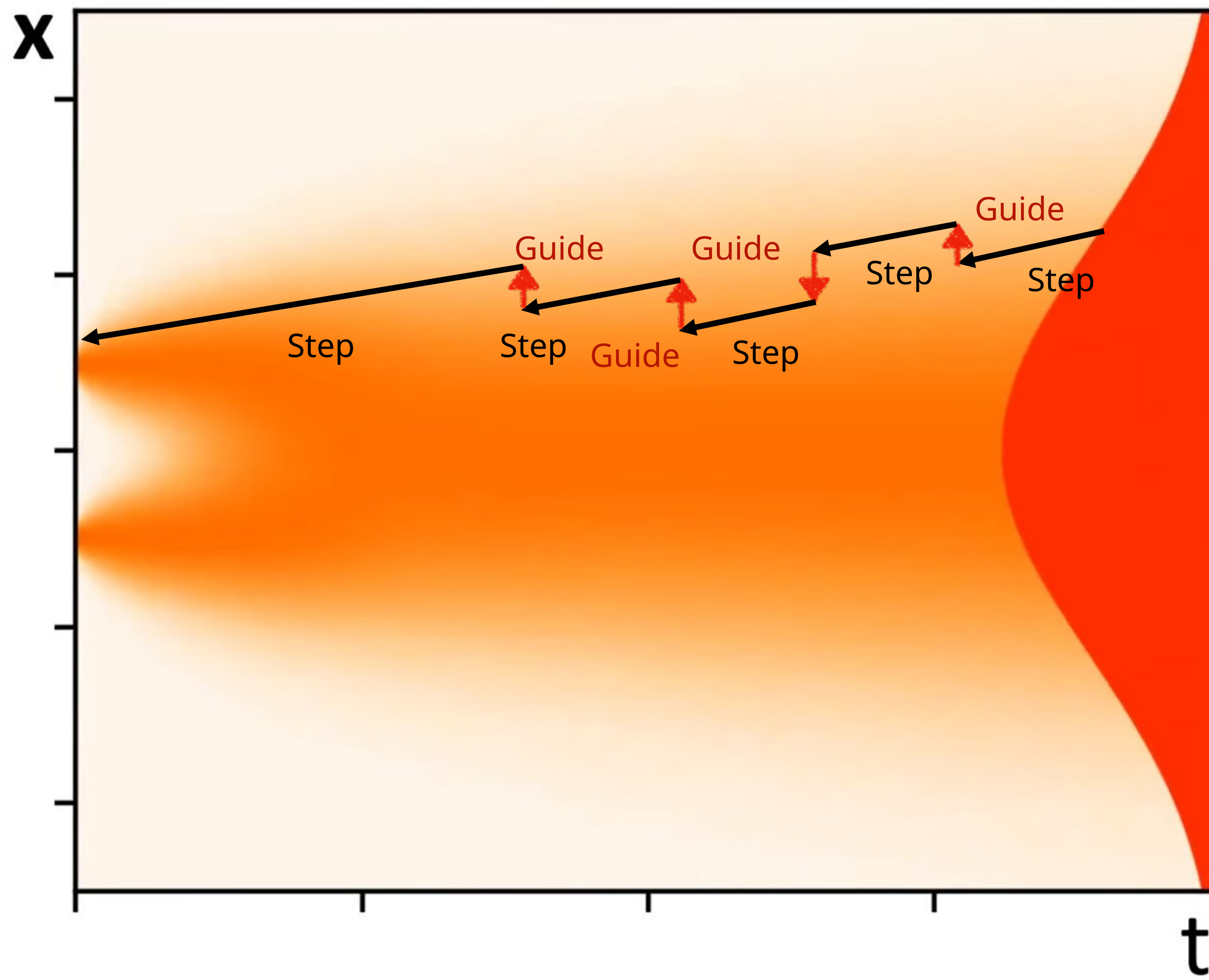


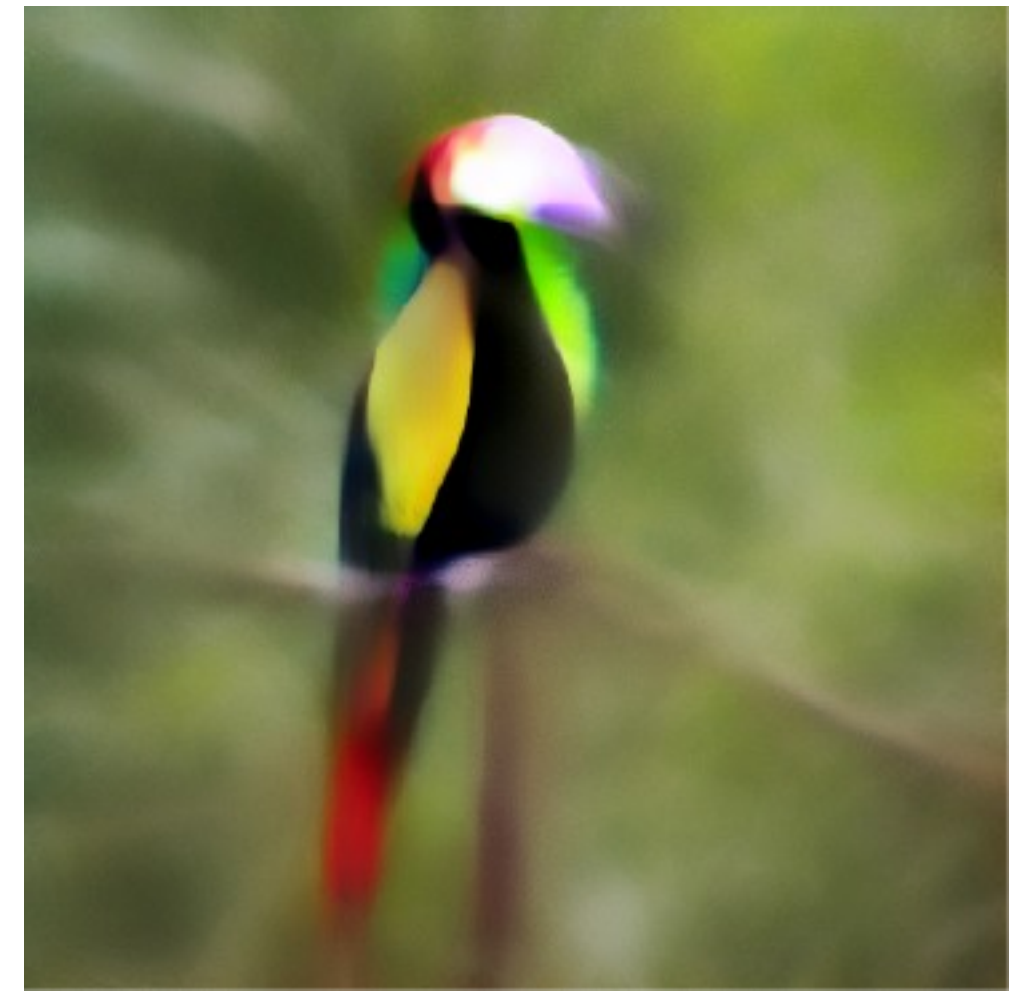
$\nabla \log p(\hat{\mathbf{x}}_0 | c)$ Gradient from
Classifier



Natural Image
Distribution

Sampling ODE View





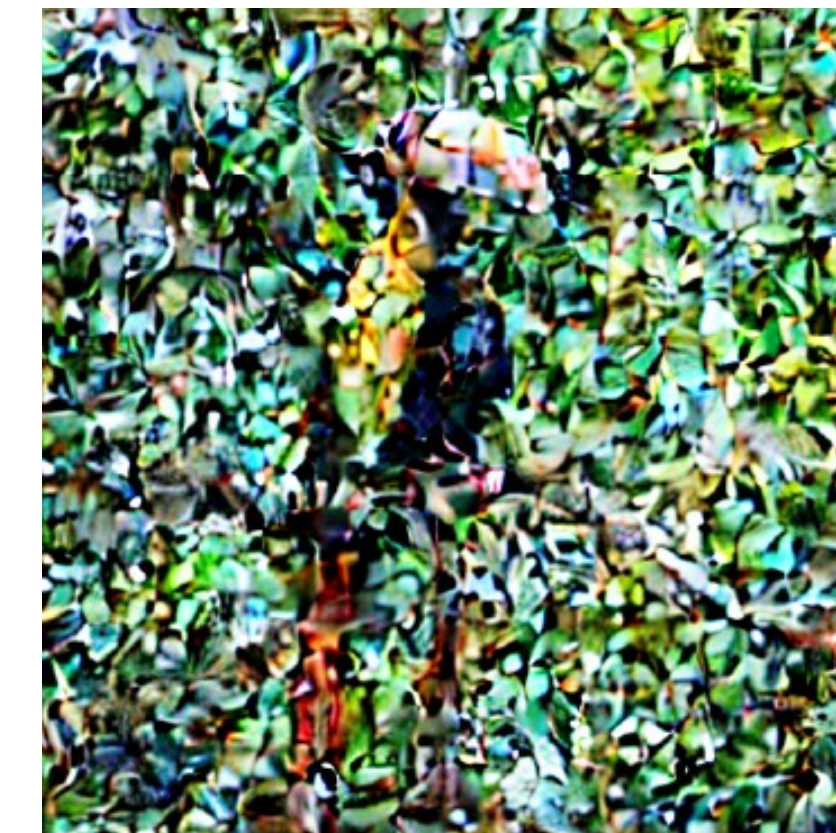
Classify These Images!

Classifier Guidance Pathologies

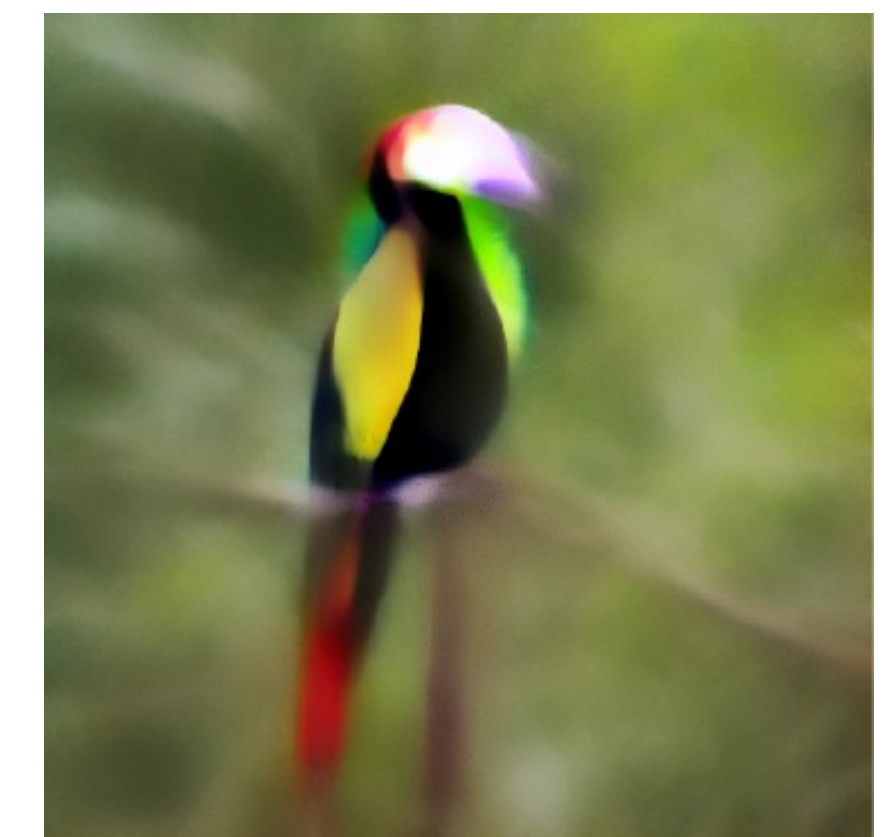
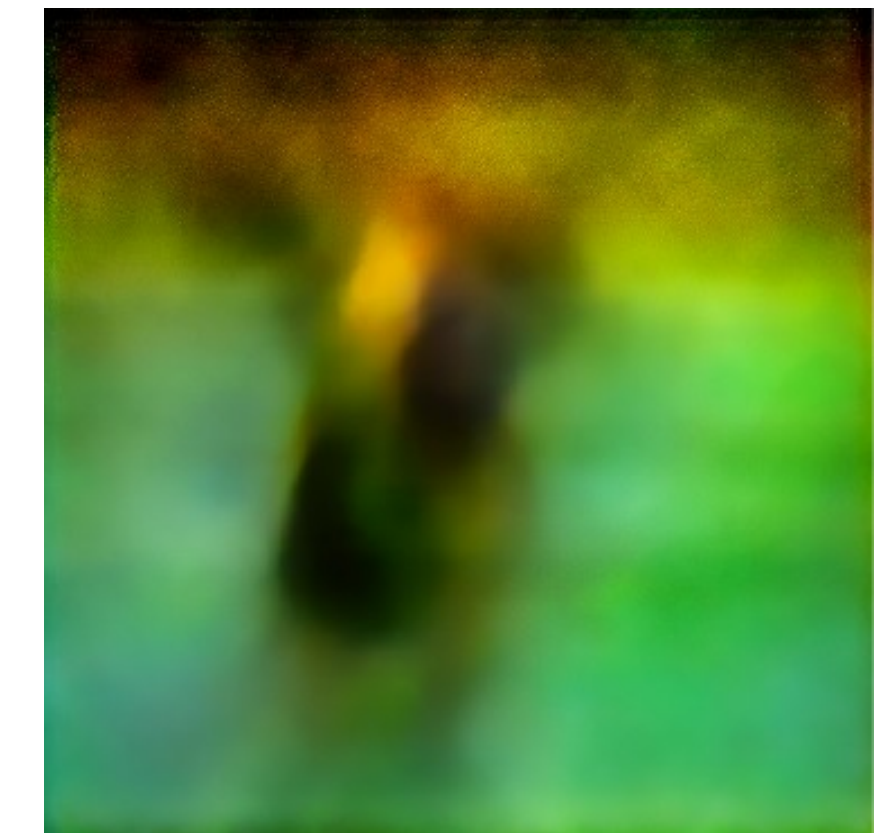
Intermediate Diffusion Steps are O.O.D. for Classifier



ImageNet Training
Data



$\mathbf{x}_t$ Noise Image



$\hat{\mathbf{x}}_1$

Image
Prediction

Two Approaches

Original

Classifier
Guidance

Guide with a
pretrained classifier.

Current

Classifier-Free
Guidance

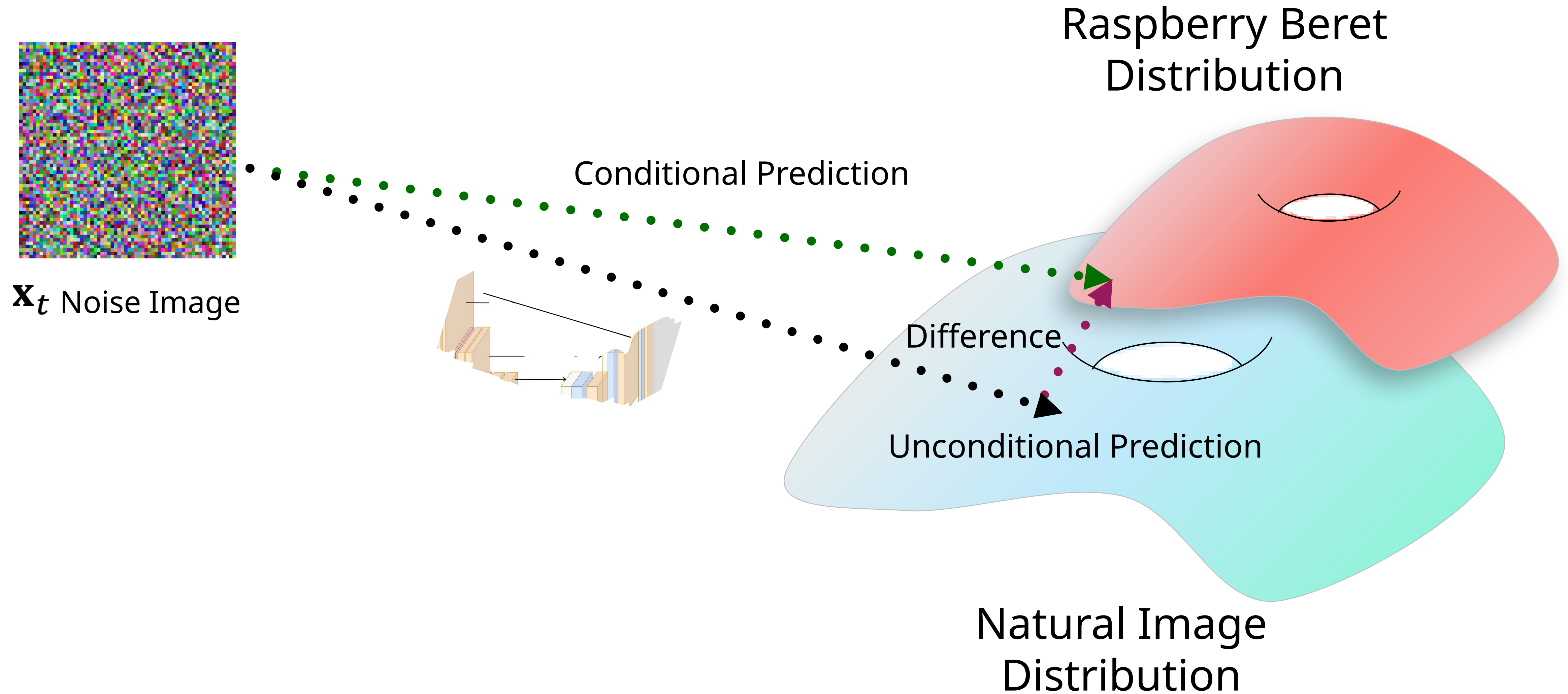
Guide a diffusion
model with itself.

Model is trained to output both

- $\epsilon_{\phi}(\mathbf{x}_t; \mathcal{C})$
- $\epsilon_{\phi}(\mathbf{x}_t; \emptyset)$

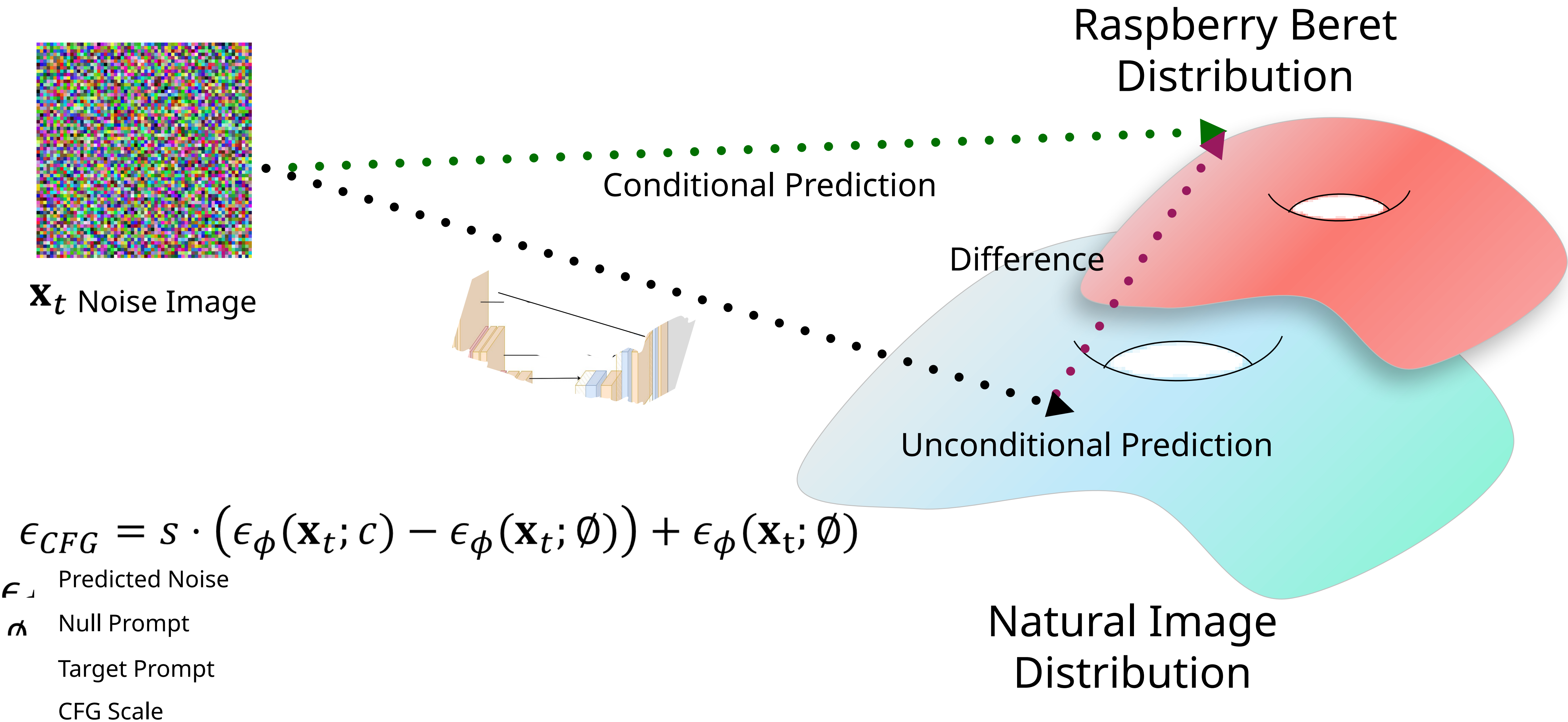
Conditional Diffusion

Model Knows Unconditional, Text-Conditioned Distributions



Classifier-Free Guidance

Model Knows Unconditional, Text-Conditioned Distributions



Classifier-Free Guidance

TLDR: Amplify Delta Between Conditional, Unconditional Predictions

$$\epsilon_{\phi,CFG} = s \cdot \left(\epsilon_{\phi}(\mathbf{x}_t; c) - \epsilon_{\phi}(\mathbf{x}_t; \emptyset) \right) + \epsilon_{\phi}(\mathbf{x}_t; \phi)$$

- ϵ_{ϕ} Predicted Noise
- c Null Prompt
- ϕ Target Prompt
- s CFG Scale

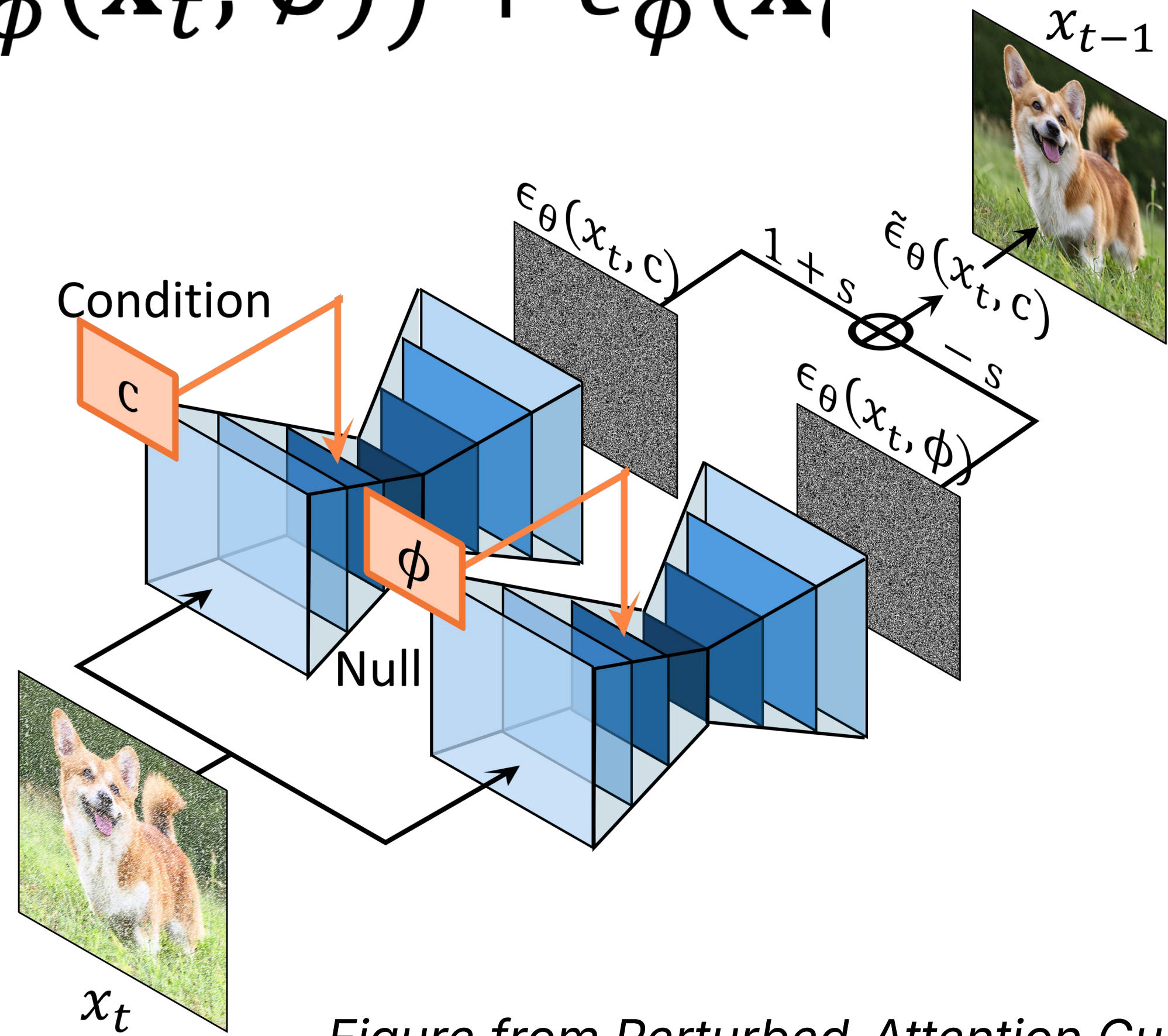


Figure from Perturbed-Attention Guidance (ECCV 2024)

Classifier-Free Guidance

Drives Generations Toward Conditional Mode



$s = 1$ (Conditional
Generation)

$s = 3$ (Conditional
Generation)

Variety of ways to use Flow Models

Visual Anagrams: Generating Multi-View Optical Illusions with Diffusion Models



Daniel Geng



Aaron Park



Andrew Owens

University of Michigan



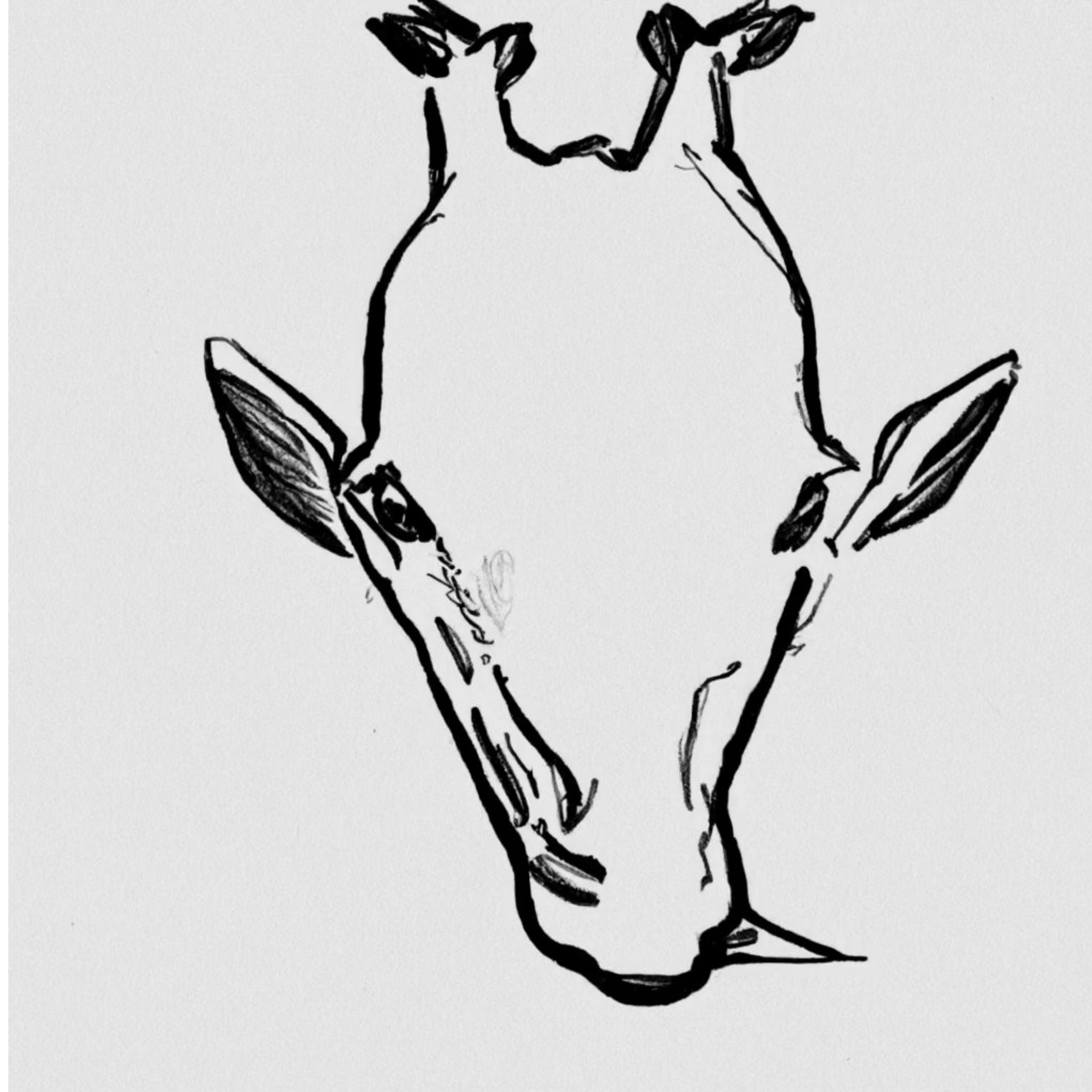
a watercolor
of mount rainier

https://dangeng.github.io/visual_anagrams/

CVPR 2024



an oil painting of people
around a campfire

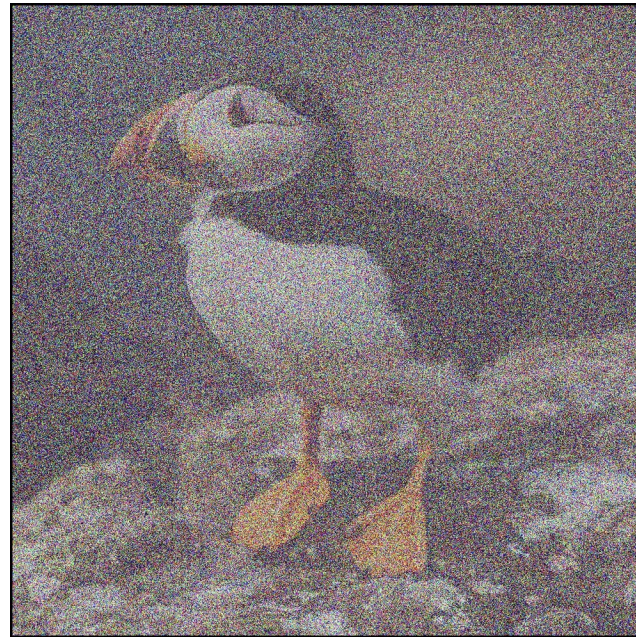
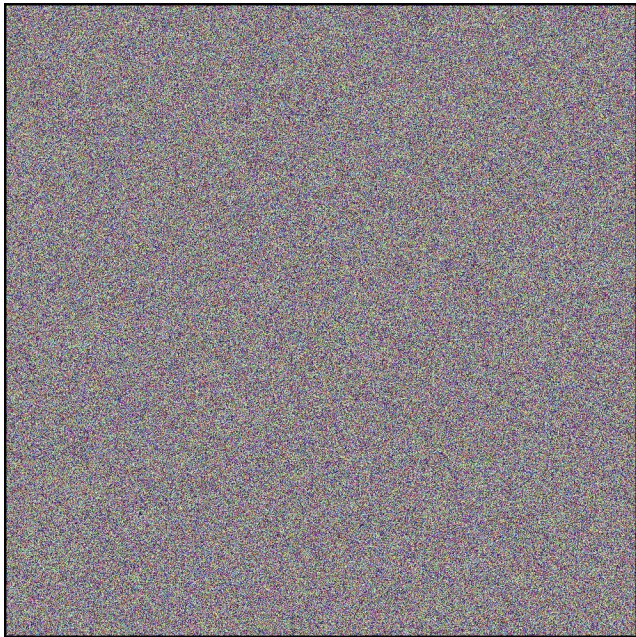


a drawing of a giraffe



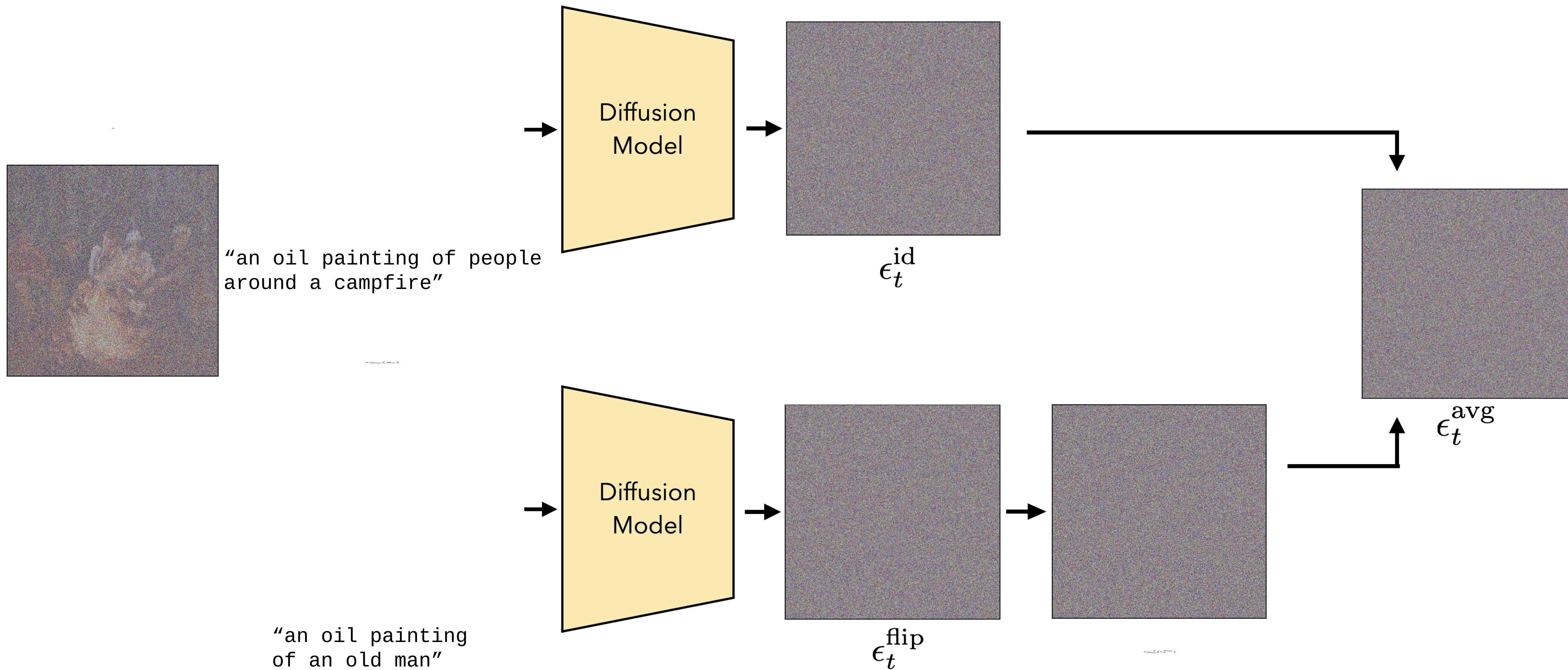
an oil painting of a
snowy mountain village

Diffusion Models

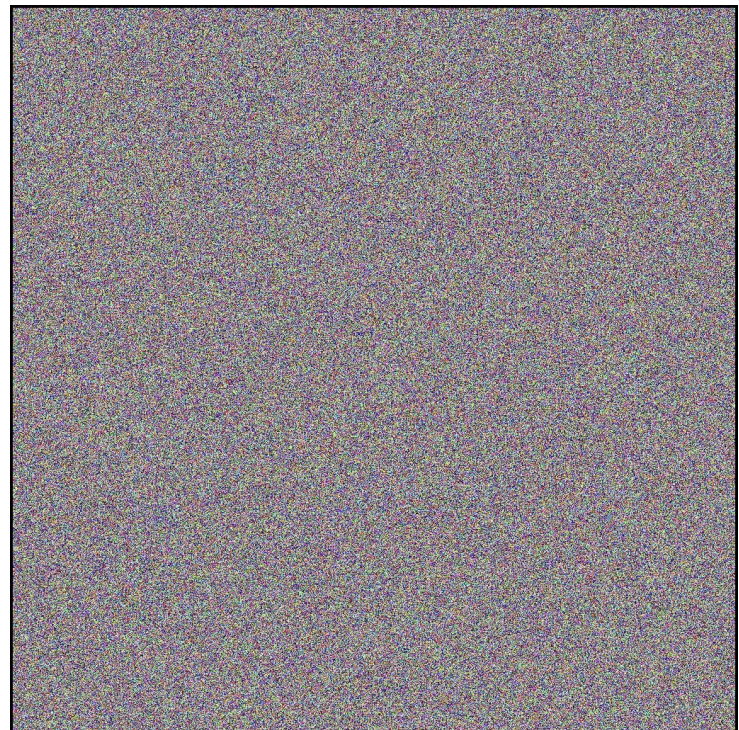


"a photo of
a puffin"

Method



Method



$$p(x|c)$$

Guidance Lets Us Sample a Conditional Distribution

$$p(x|c)$$

Guidance Lets Us Sample a Conditional Distribution

What if we're creative about the conditioning?



Condition on the First frame and Audio, Generate the Video



Condition on Video Pixels, Generate Depth Map

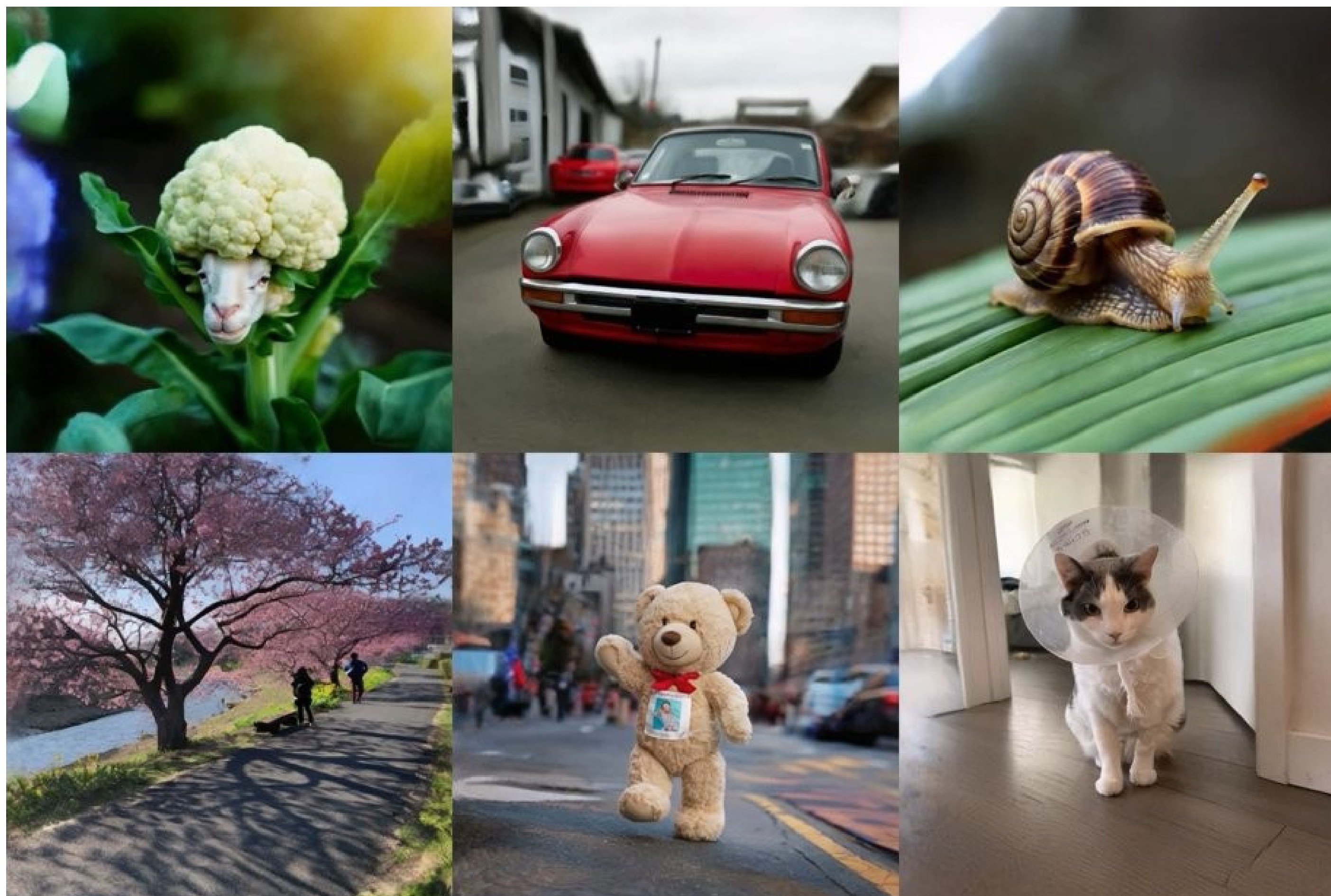
Conditionin
g



Samples



Condition on Start and End Frame, Generate Intermediate Frames



Condition on Image + Camera, Generate Novel Views

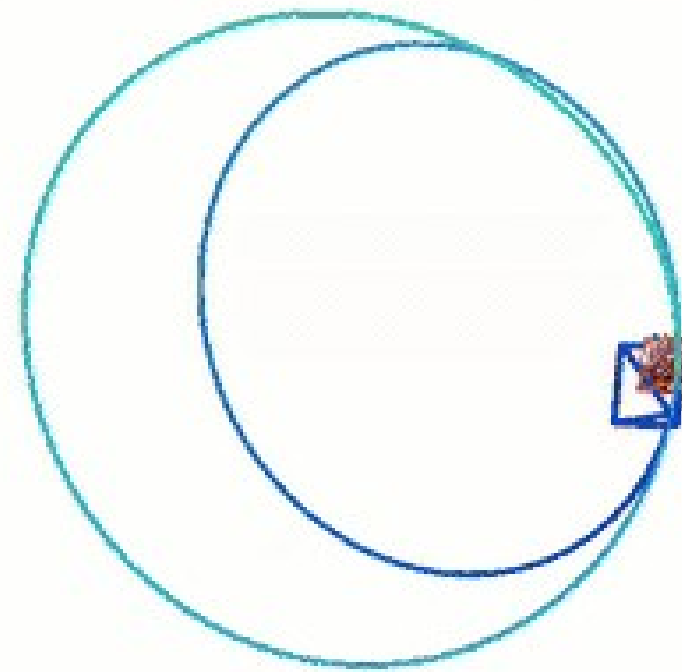
["CAT3D: Create Anything in 3D with Multi-View Diffusion Models" Gao et al. \(2024\)](#)

)

INPUT VIEW



CAMERA CONTROL



Condition on Image + Camera, Generate Novel Views