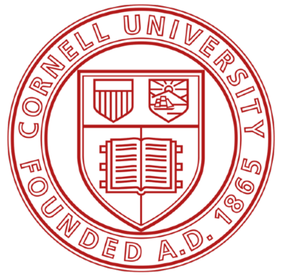
Lecture 14: Diffusion models - Part 1

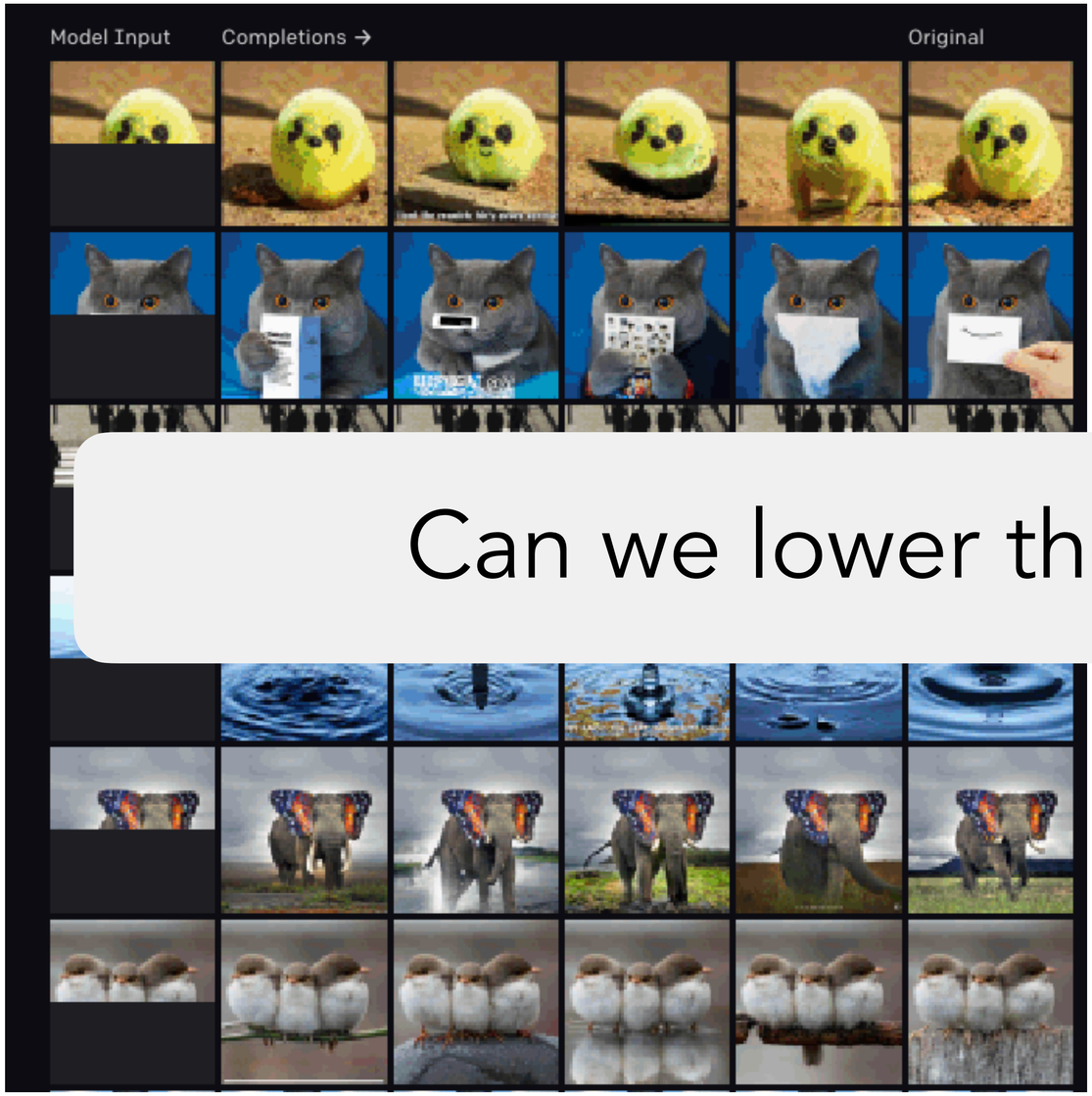
CS 5670: Introduction to Computer Vision

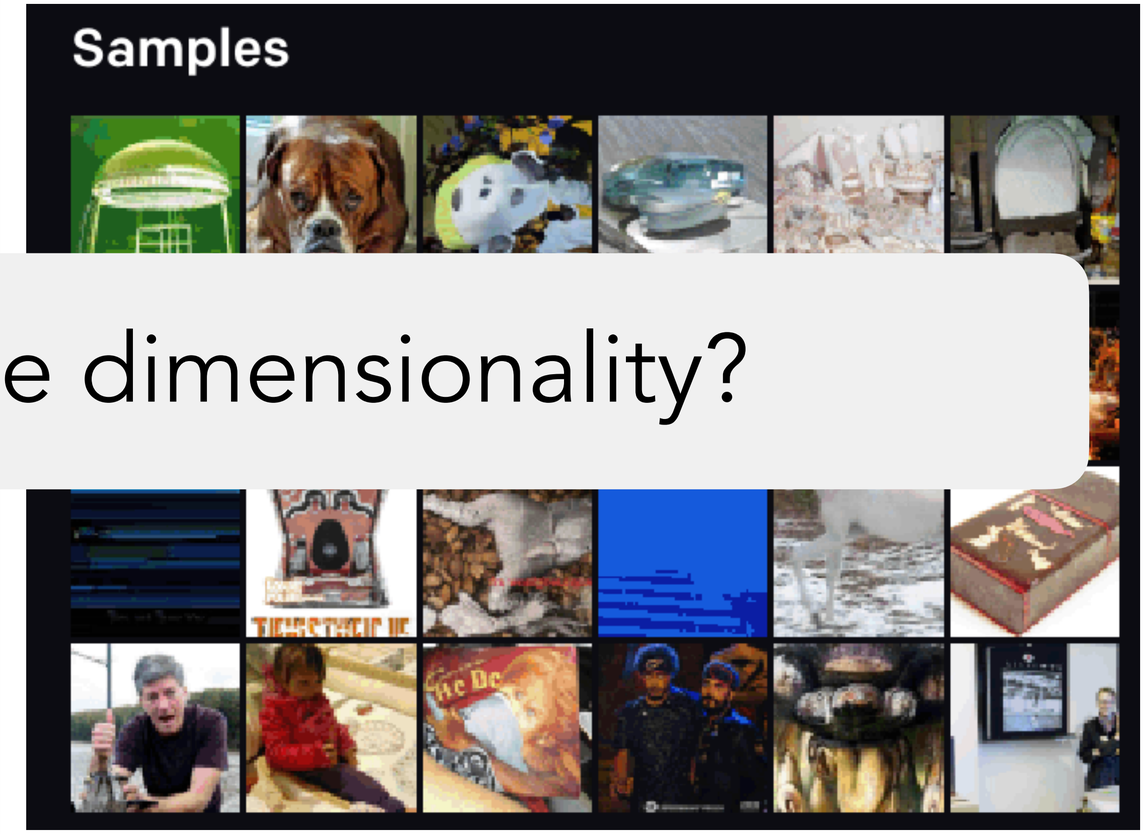


Announcements

- Midterm on Oct. 20

Last class: pixel autoregressive models





Can we lower the dimensionality?

https://openai.com/blog/image-gpt/

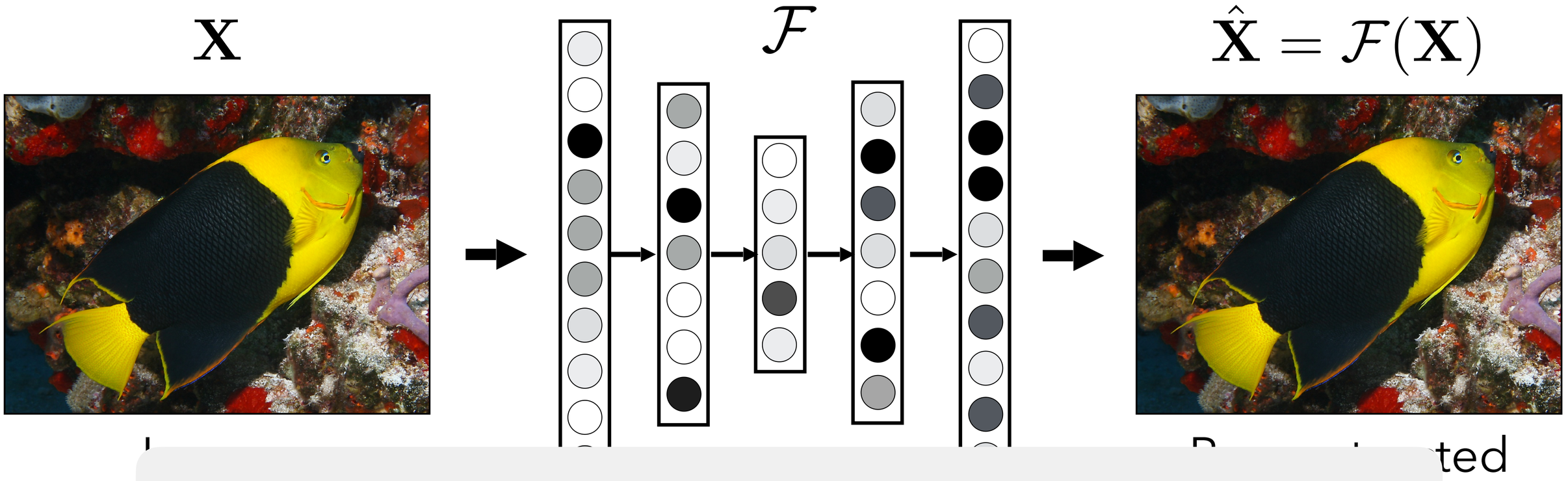
M. Chen et al., Generative pretraining from pixels, ICML 2020

Source: S. Lazebnik

Today

- Generative models in latent space (continuing from last class)
- Diffusion models

Lower dimensionality using an autoencoder

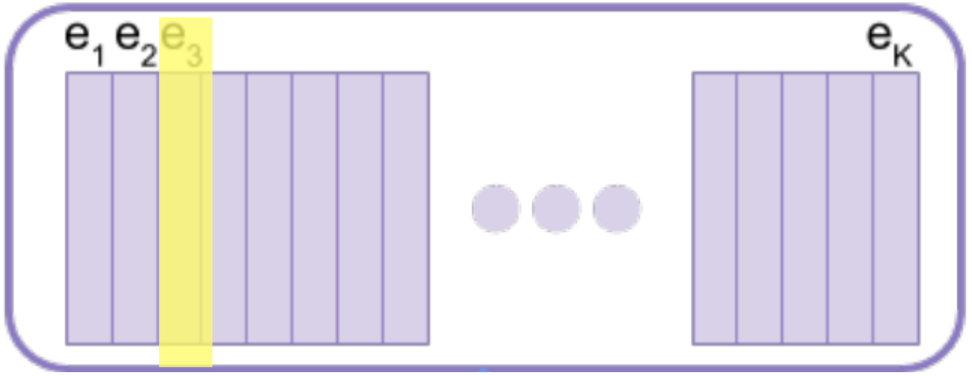


How can we make the output discrete?

Figure source: Torralba, Freeman, Isola

Vector quantization

Predict a real-valued vector, then "snap" it to a nearest neighbor from a *codebook*.



Codebook

Patch **x**

E(**x**) → **z** → Quantization → $\mathbf{e}_k$

$$\text{Quantize}(E(\mathbf{x})) = \mathbf{e}_k \quad \text{where } k = \arg\min_j ||E(\mathbf{x}) - \mathbf{e}_j||$$

Figure adapted from [van den Oord et al., "VQ-VAE", 2017

Vector quantized (variational) autoencoder

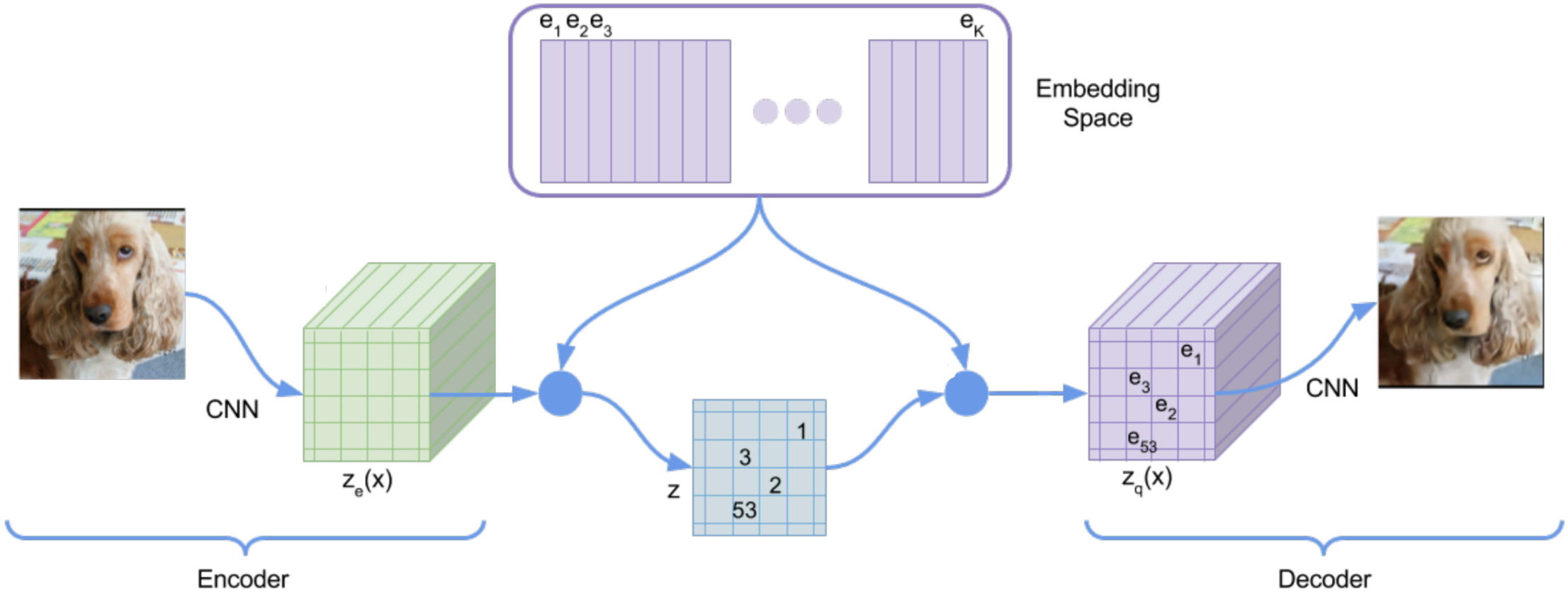
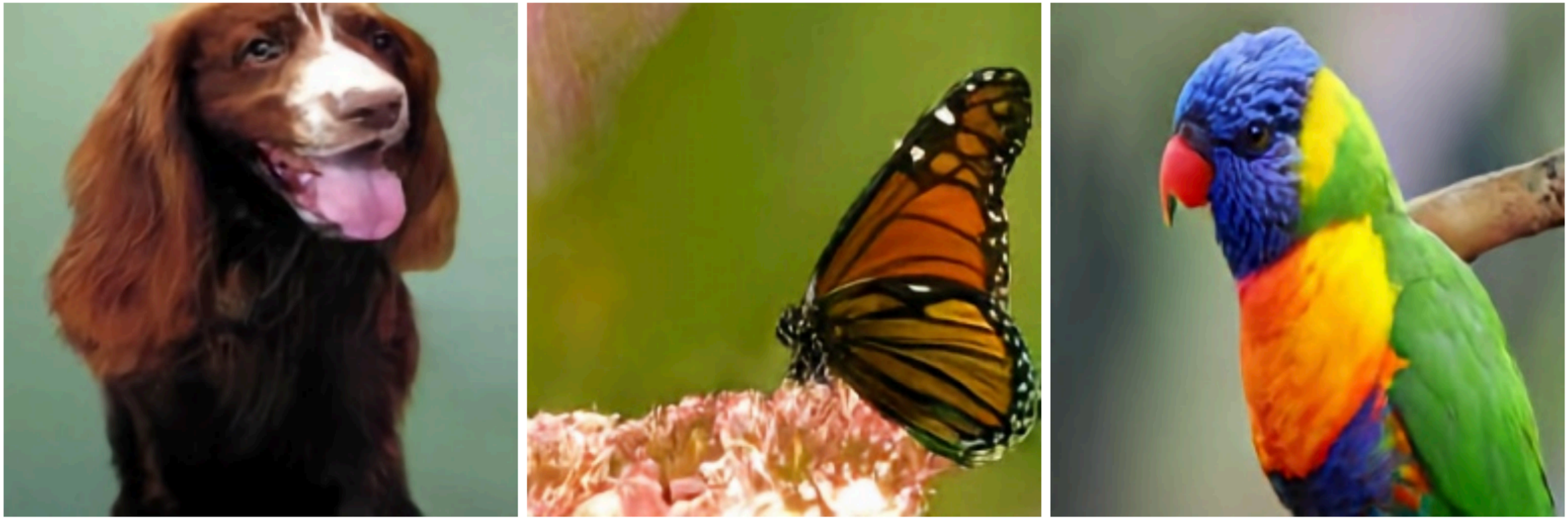


Figure source: [van den Oord et al., “VQ-VAE”, 2017]

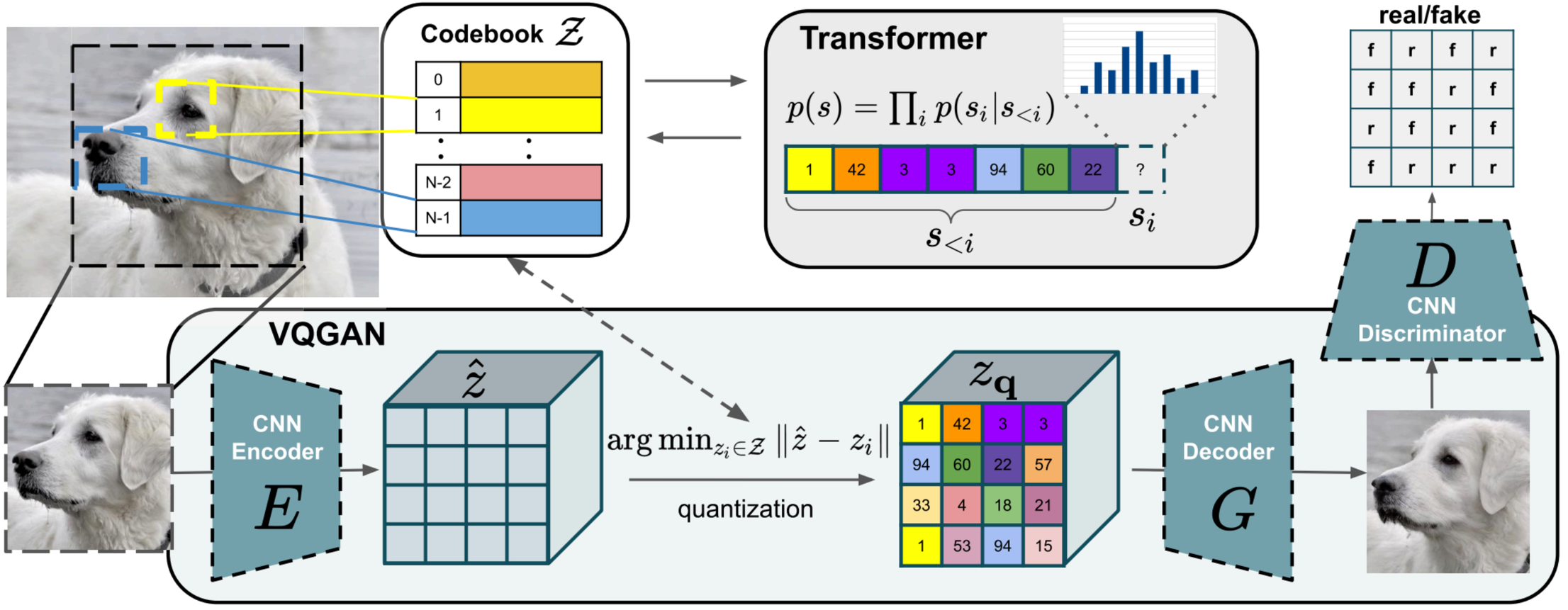
VQ-VAE image generation results



Generated images (256 × 256)

Figure source: [van den Oord et al., “VQ-VAE 2”, 2017

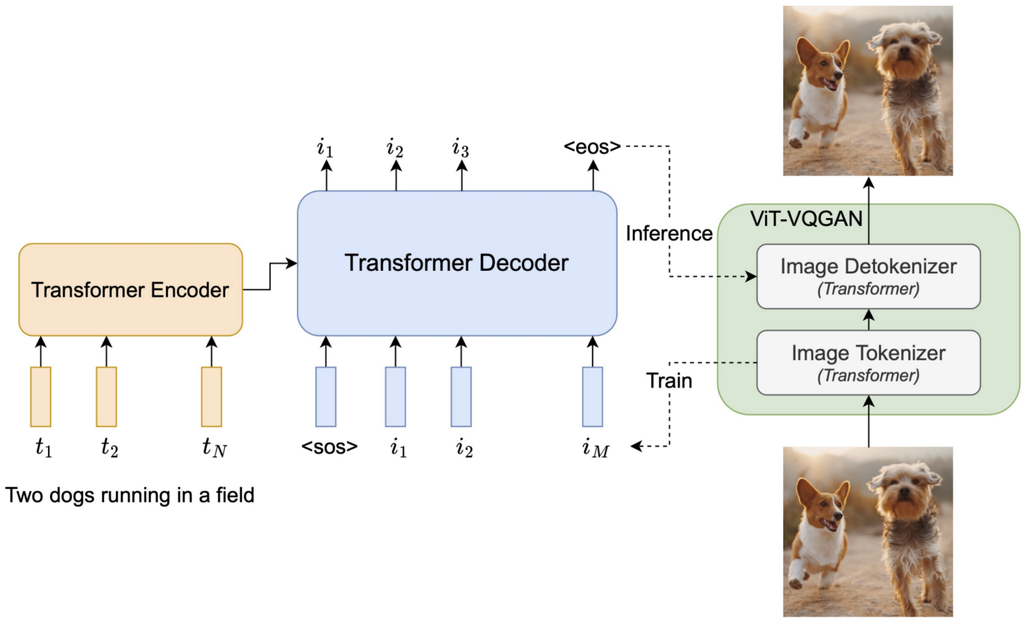
Vector quantized GAN (VQ-GAN)



Add a GAN loss to get crisper images

[Esser et al., "Taming Transformers for High-Resolution Image Synthesis", 2021

Autoregressive text-to-image generation



A. *A photo of a frog reading the newspaper named "Toaday" written on it. There is a frog printed on the newspaper too.*

B. *A portrait of a statue of the Egyptian god Anubis wearing aviator goggles, white t-shirt and leather jacket. The city of Los Angeles is in the background. Hi-res DSLR photograph.*

[Yu et al., "Parti", 2022]

Today

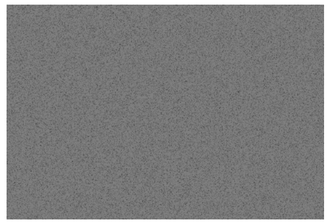
- Generative models in latent space
- **Diffusion models**

Recall: the denoising problem

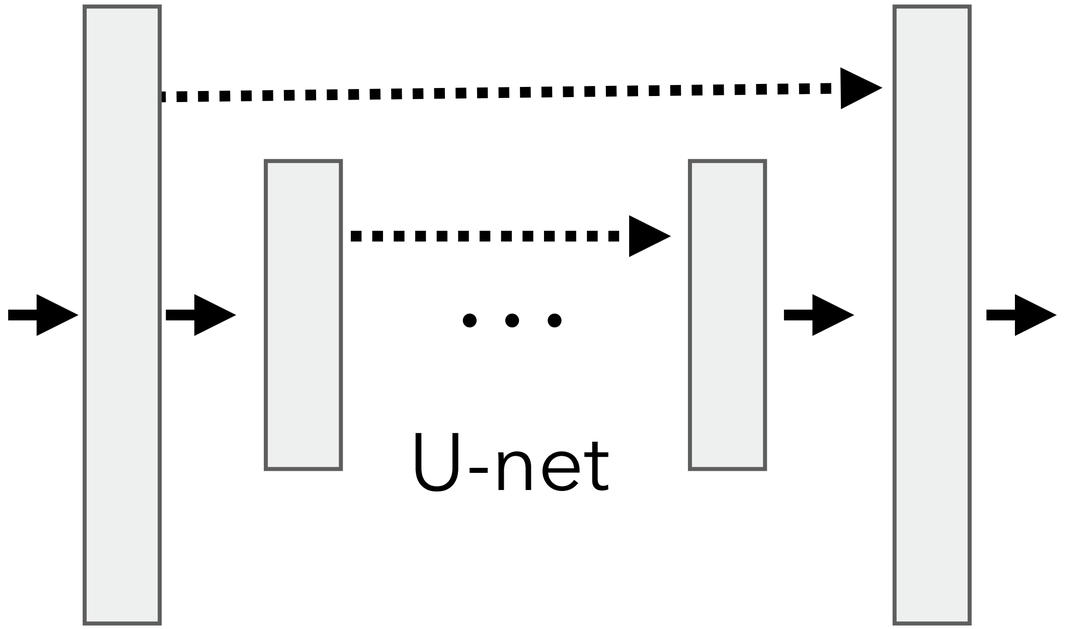
=

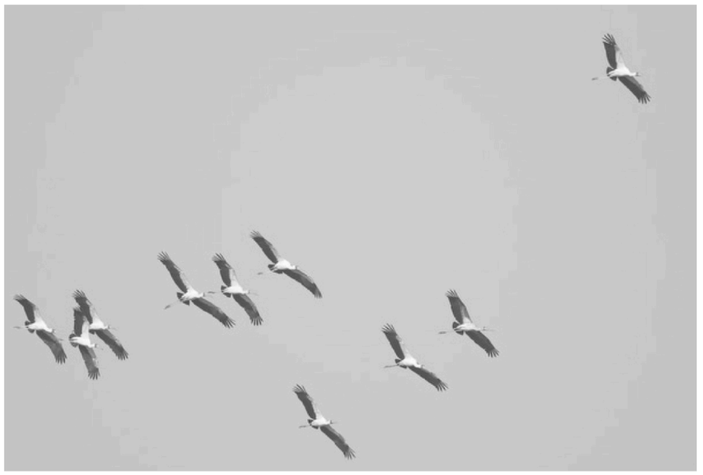
$\mathbf{x}_{\text{clean}}$

+



random noise

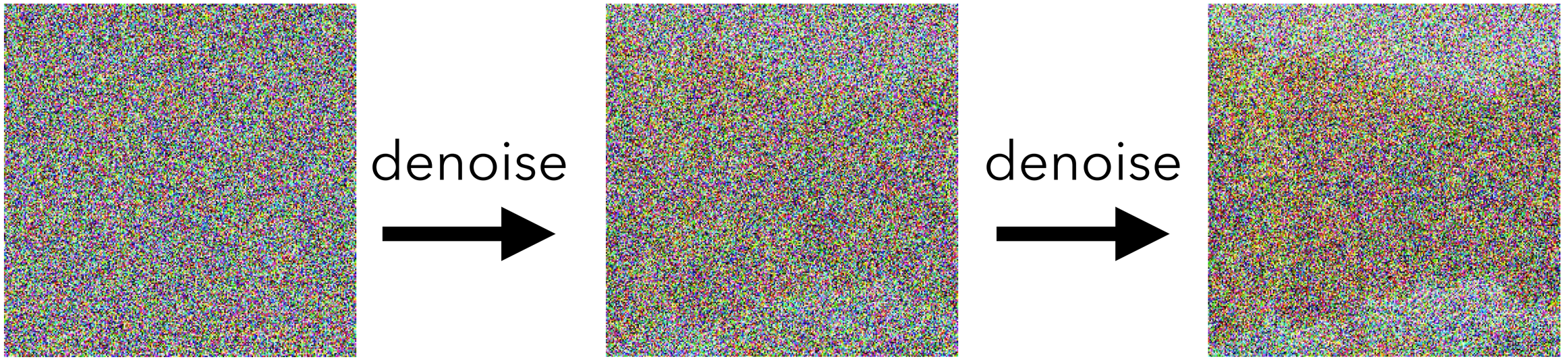




$\hat{\mathbf{x}}$

Loss: $\|\mathbf{x}_{\text{clean}} - \hat{\mathbf{x}}\|^2$

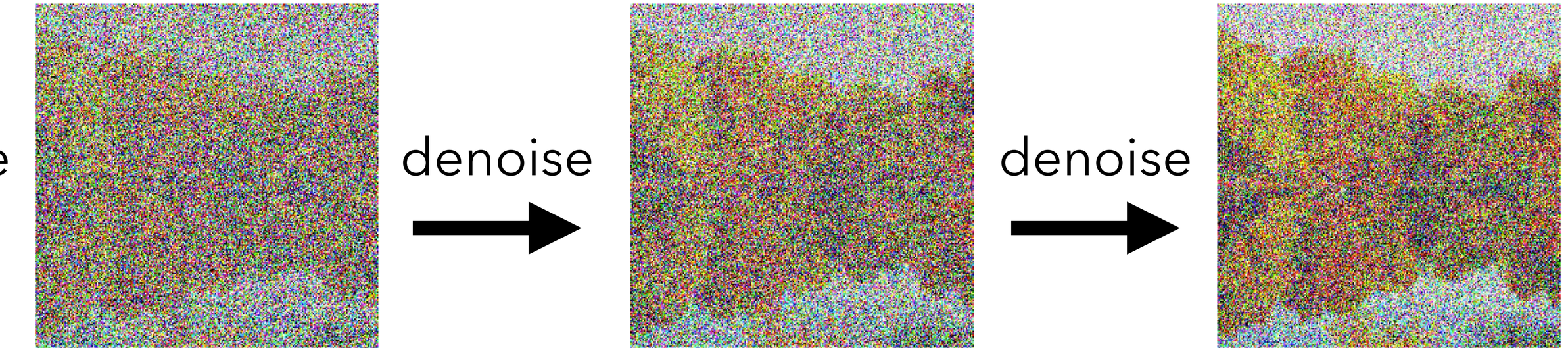
From noise to an image



Random noise

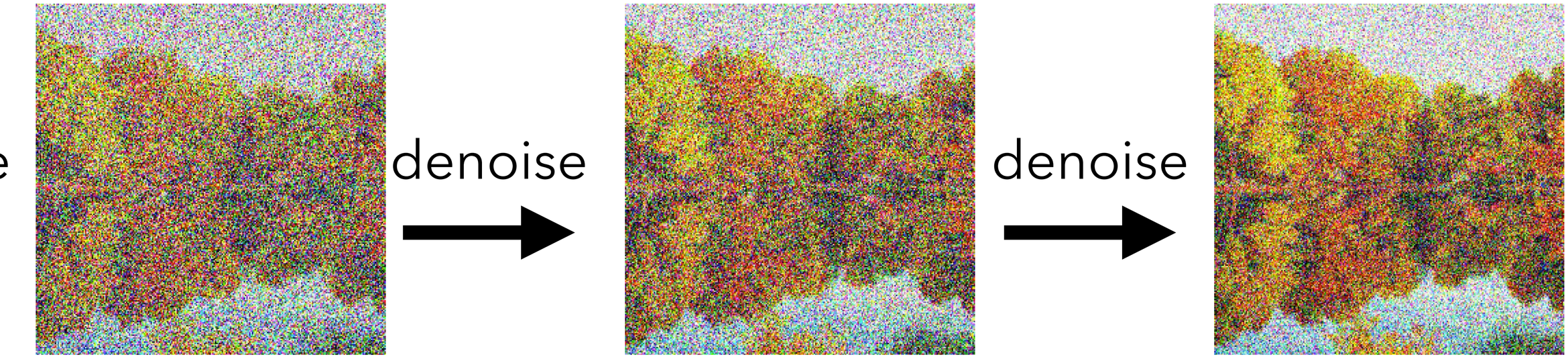
Example source: Aditya Ramesh

From noise to an image



Example source: Aditya Ramesh

From noise to an image



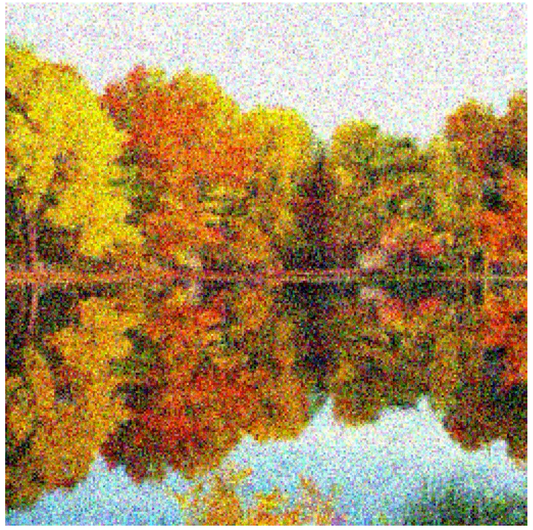
Example source: Aditya Ramesh

From noise to an image

Example source: Aditya Ramesh

From noise to an image

se



denoise

denoise

Example source: Aditya Ramesh

From noise to an image

noise

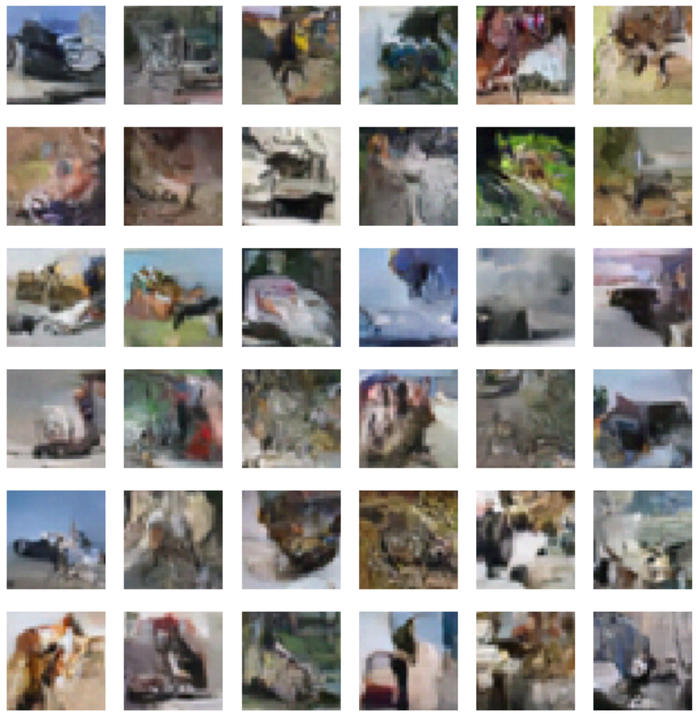
denoise

denoise

Generated image!

Example source: Aditya Ramesh

Diffusion models

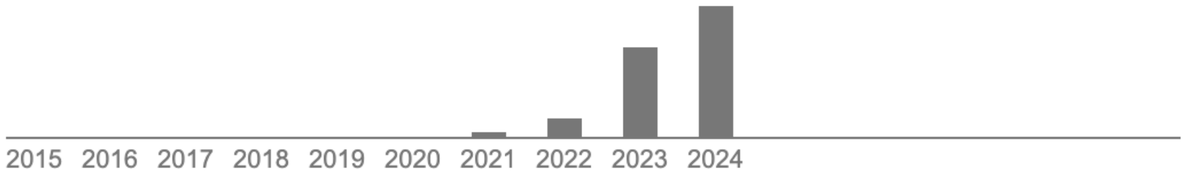


[Sohl-Dickstein et al., 2015]

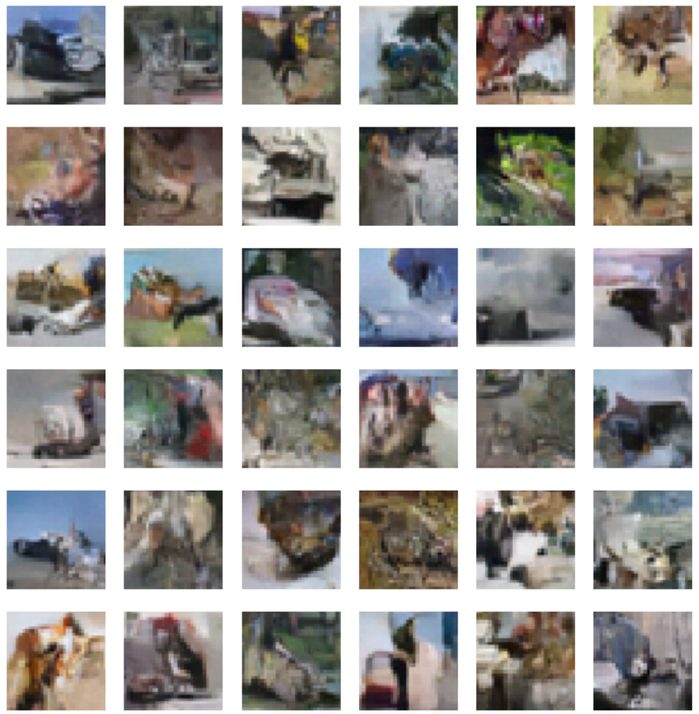
Diffusion models

Deep unsupervised learning using nonequilibrium thermodynamics

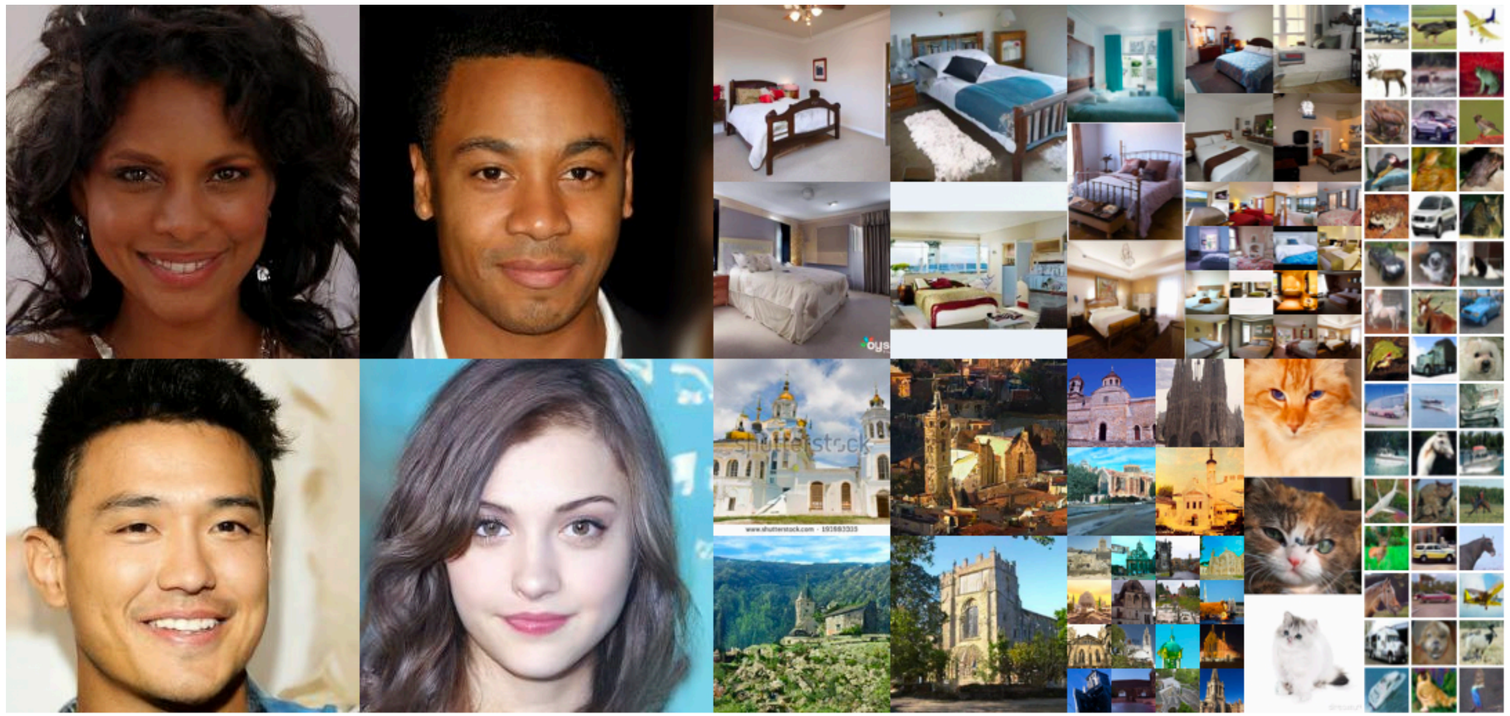
Authors	Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, Surya Ganguli
Publication date	2015/3/12
Journal	International Conference on Machine Learning
Description	A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions in which learning, sampling, inference, and evaluation are still analytically or computationally tractable. Here, we develop an approach that simultaneously achieves both flexibility and tractability. The essential idea, inspired by non-equilibrium statistical physics, is to systematically and slowly destroy structure in a data distribution through an iterative forward diffusion process. We then learn a reverse diffusion process that restores structure in data, yielding a highly flexible and tractable generative model of the data. This approach allows us to rapidly learn, sample from, and evaluate probabilities in deep generative models with thousands of layers or time steps, as well as to compute conditional and posterior probabilities under the learned model. We additionally release an open source reference implementation of the algorithm.
Total citations	Cited by 5829



Diffusion models



[Sohl-Dickstein et al., 2015]



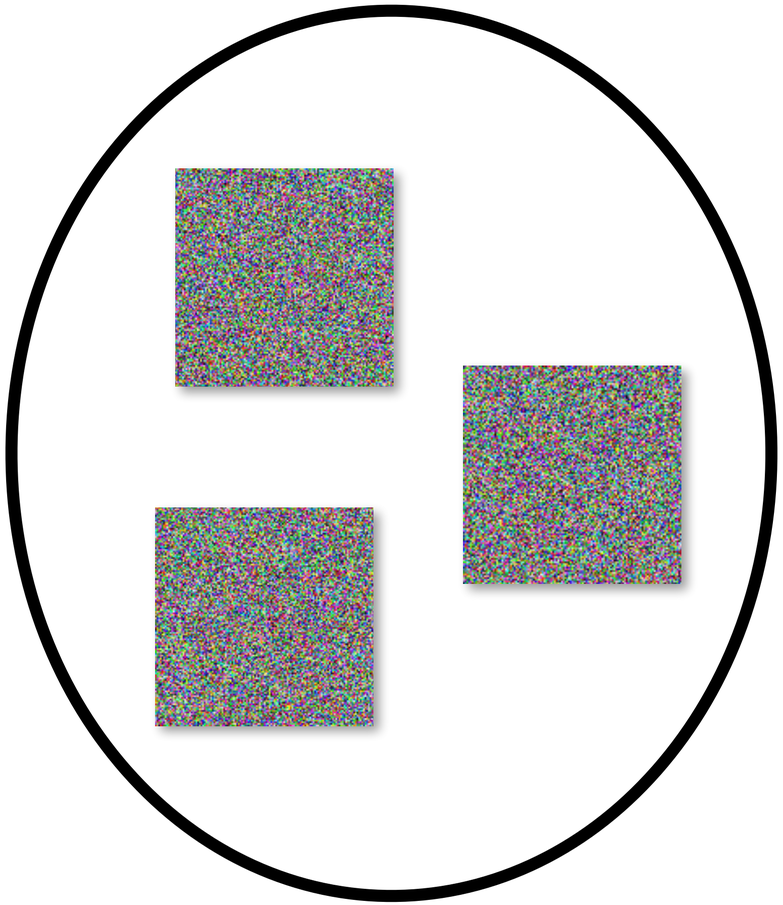
[Ho, et al., “Denoising Diffusion”, 2020]

“Sprouts in the shape of text 'Imagen' coming out of a fairytale book.”

“A dragon fruit wearing karate belt in the snow.”

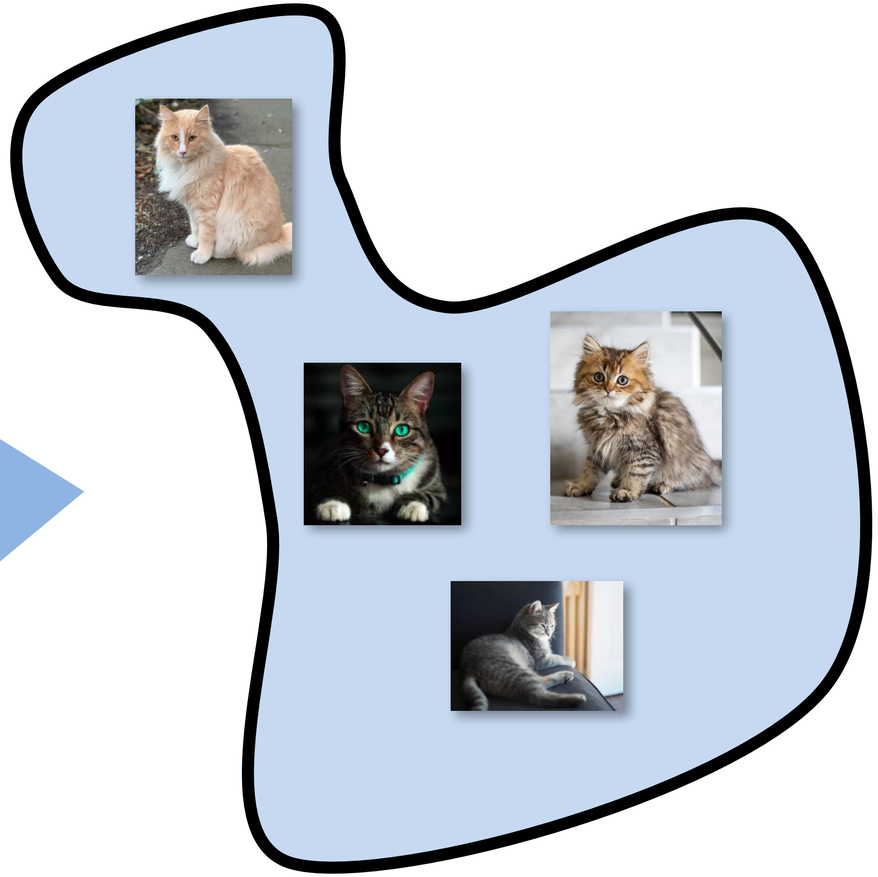
Source: L. Shen, C. Saharia et al. Photorealistic Text-to-Image Diffusion Models with Deep Language Understanding. NeurIPS 2022


Today: Google Veo 3, 2025



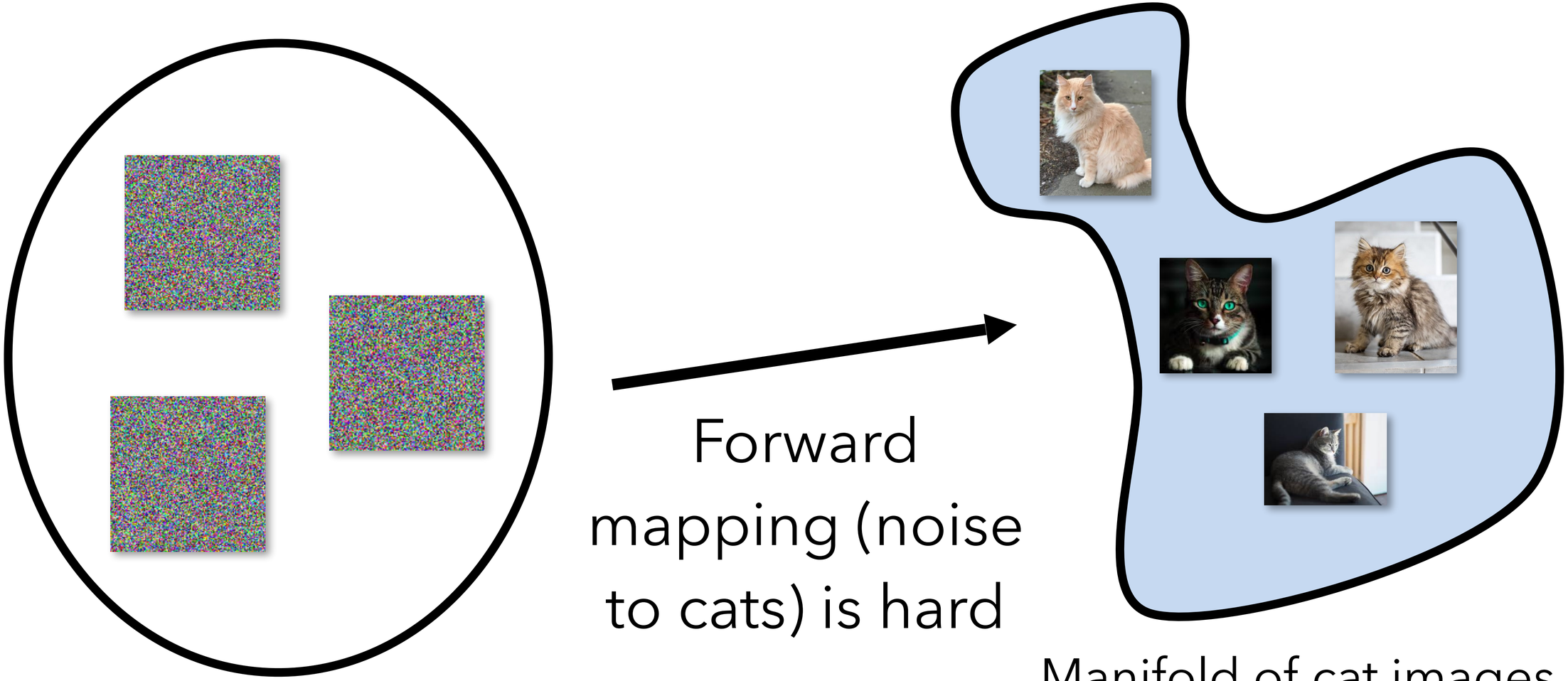
Random images

Diffusion

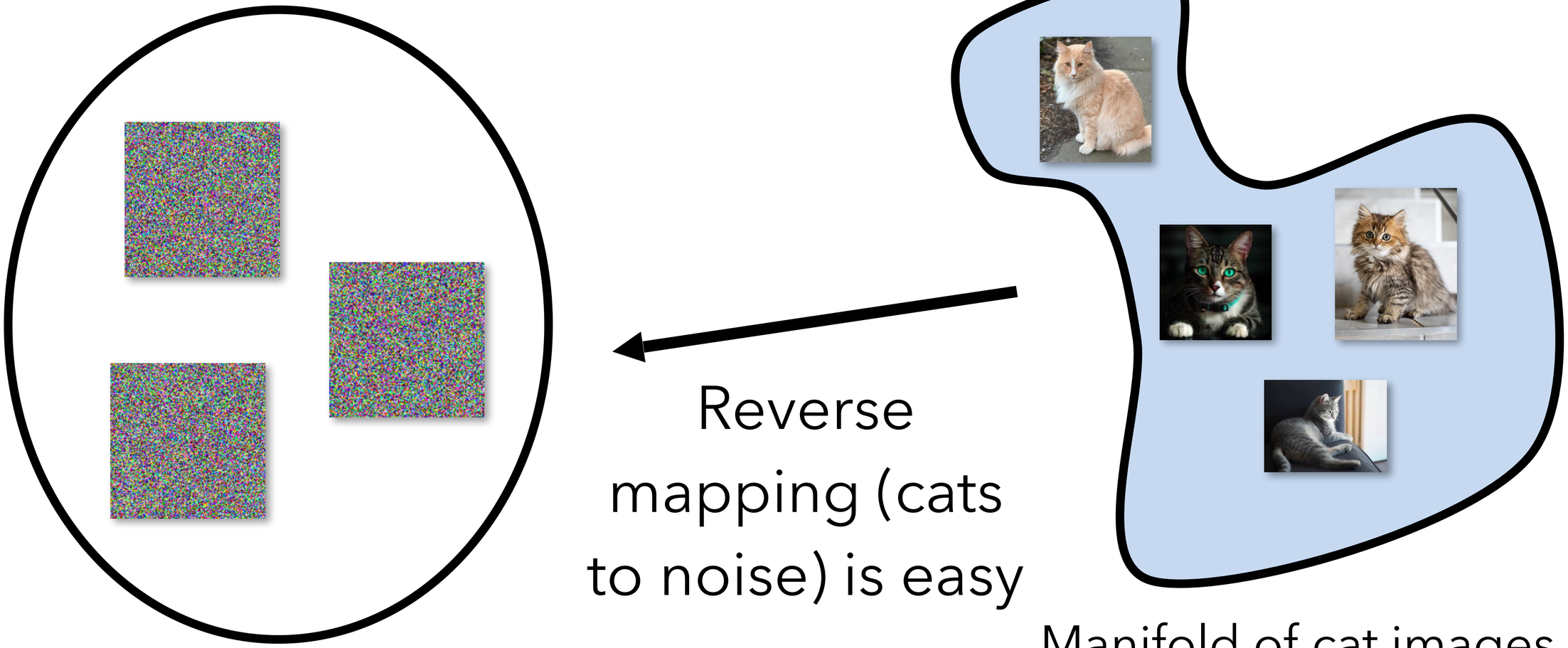


Manifold of cat images

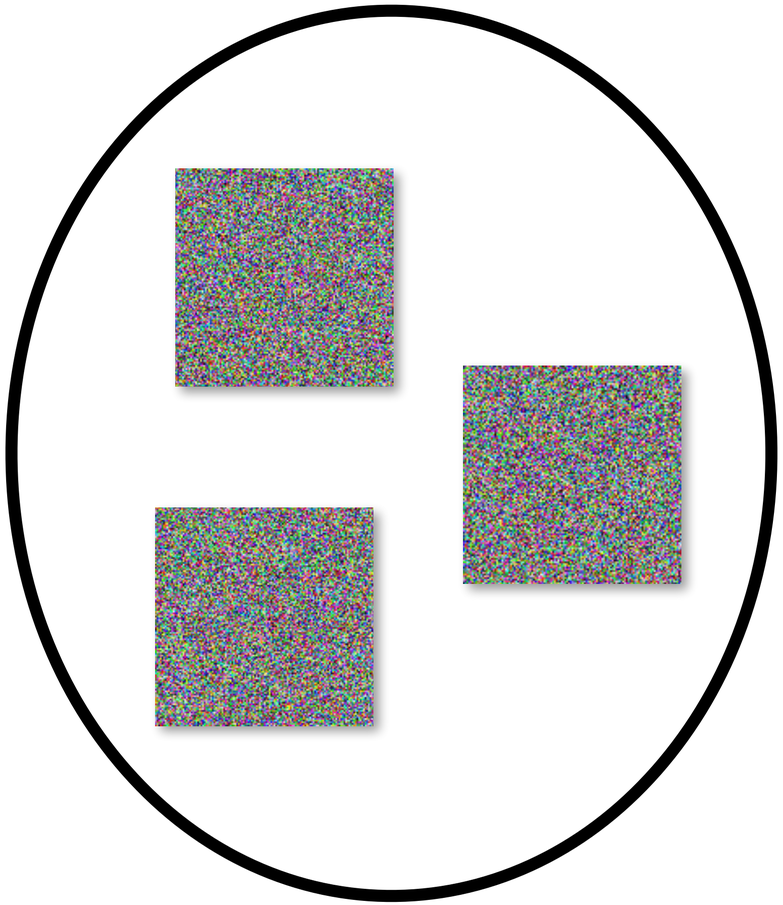
Slide source: N. Snavely & S. Seitz



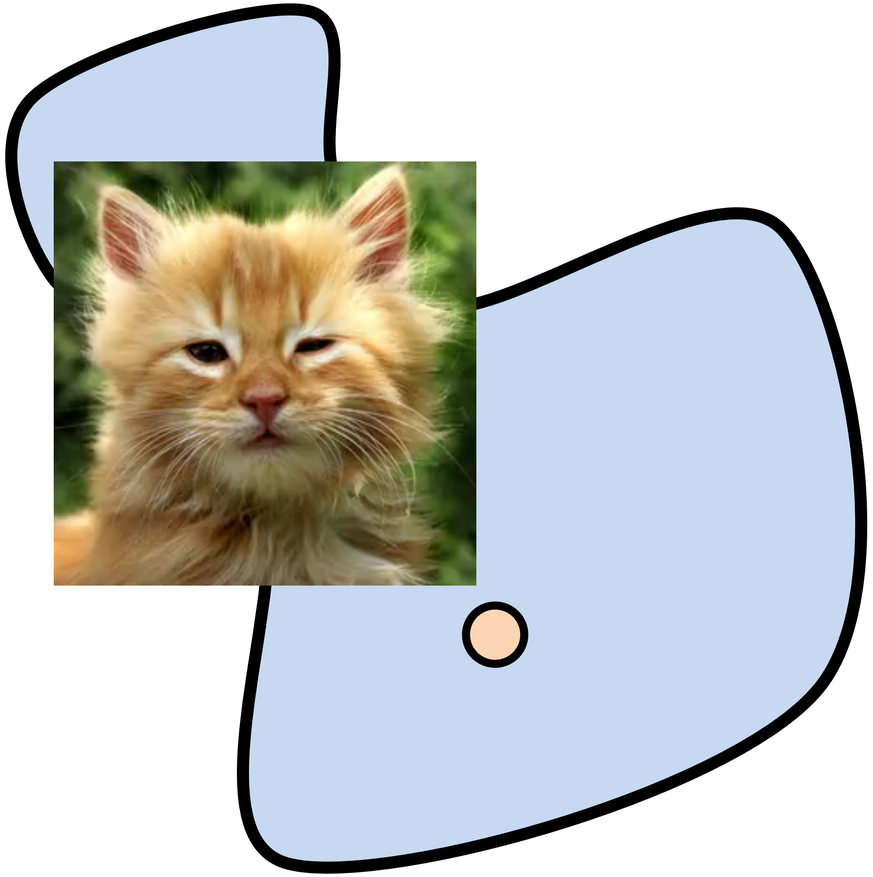
Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz

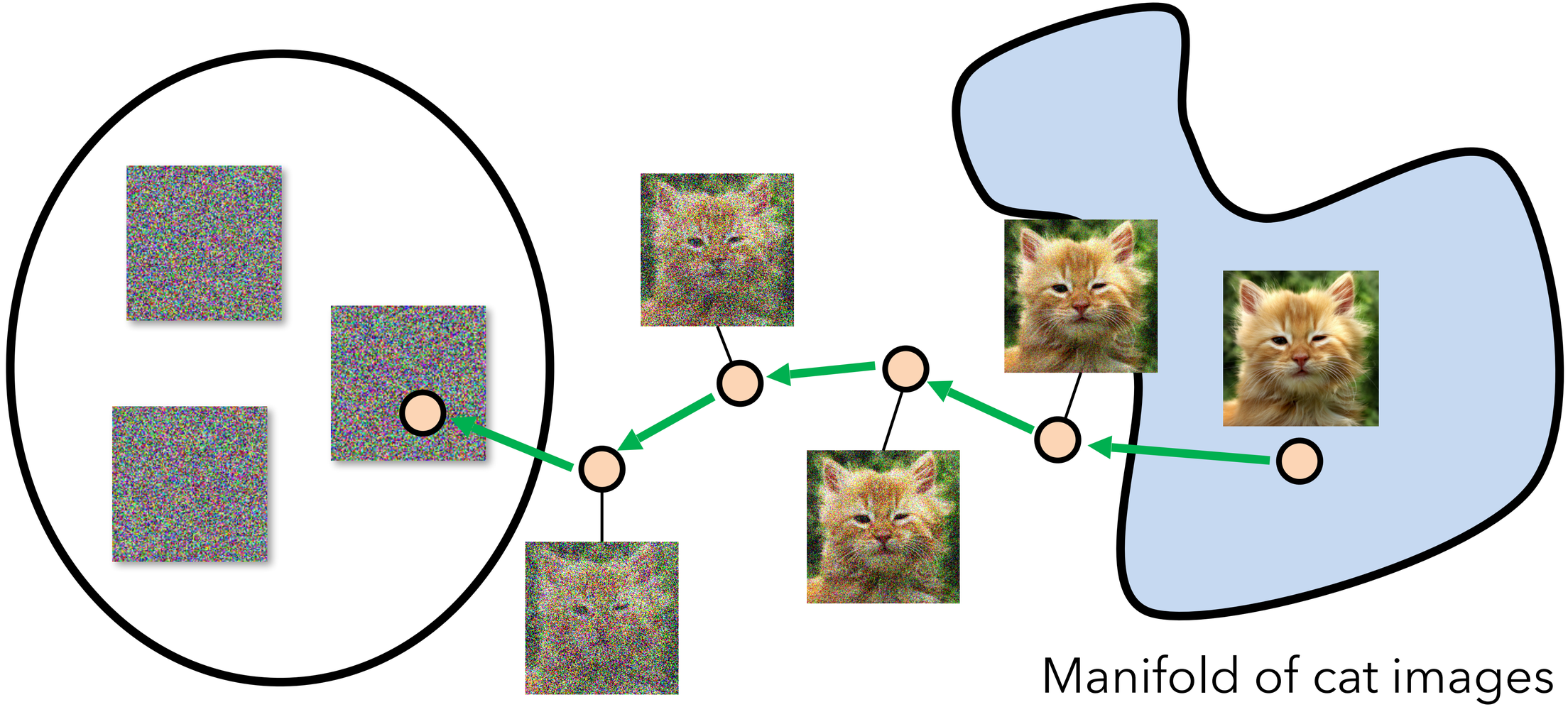


Random images

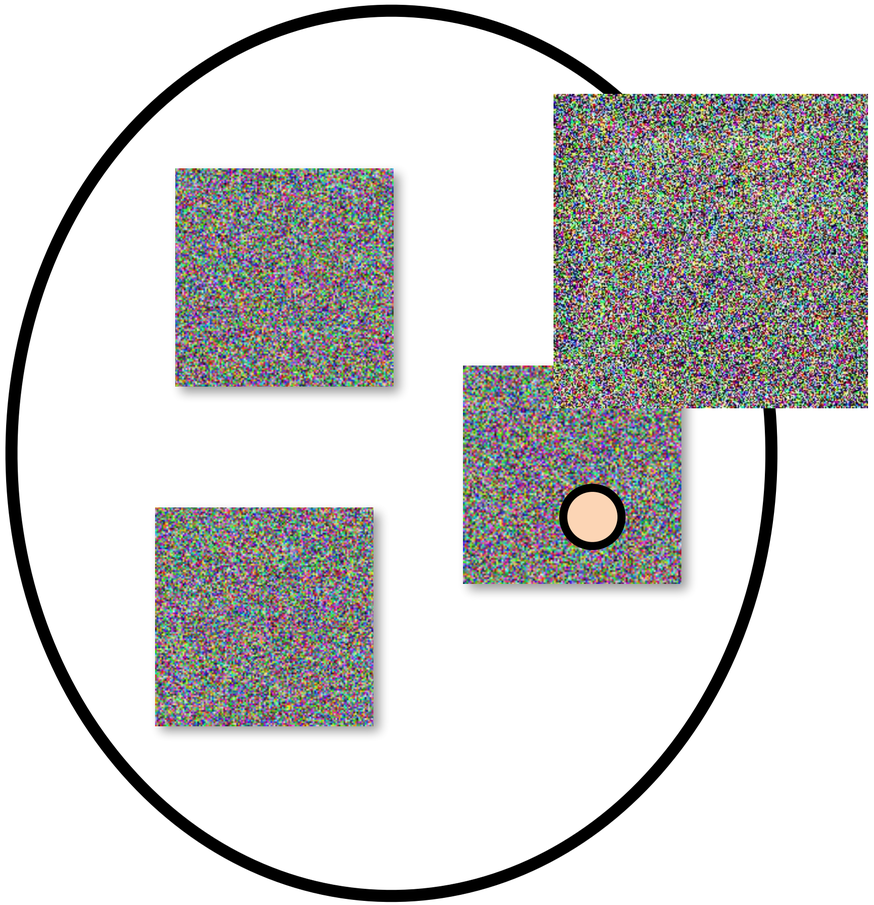


Manifold of cat images

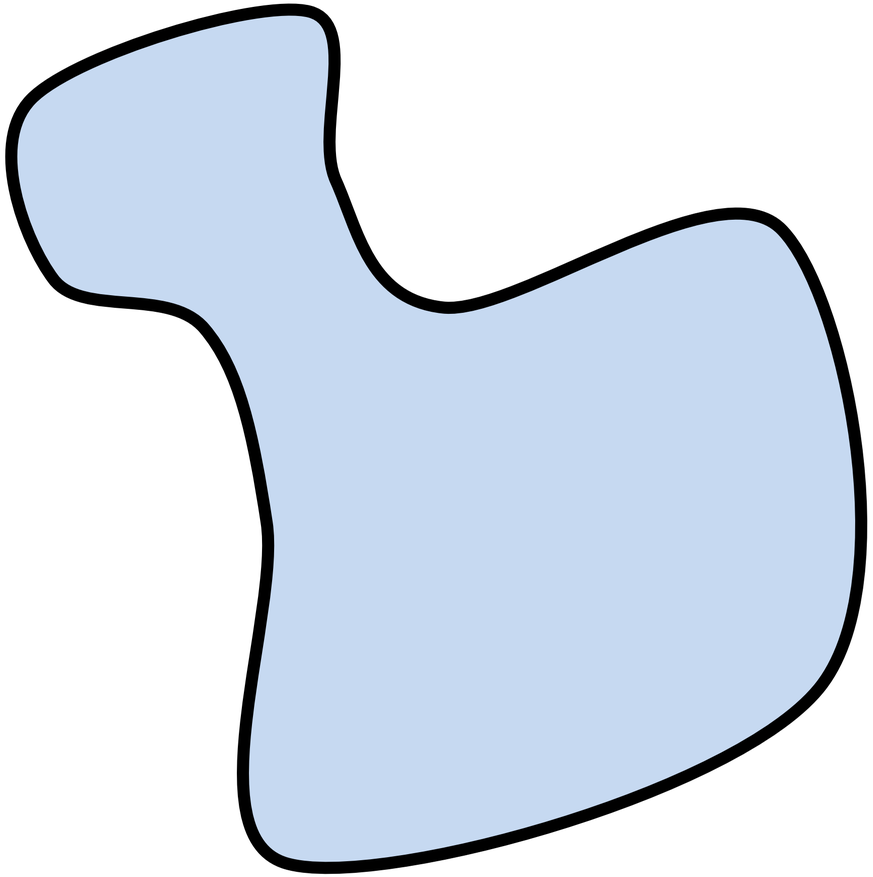
Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz

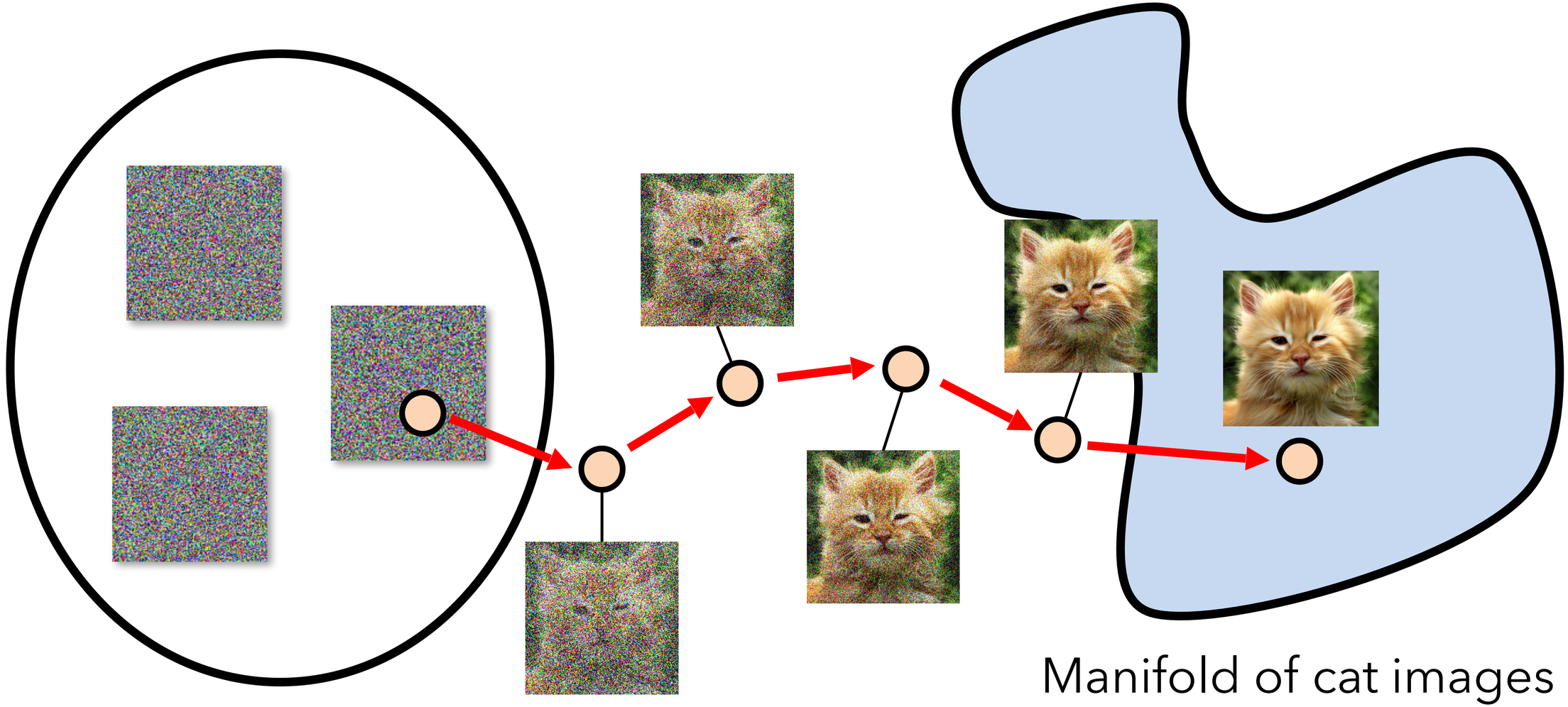


Random images

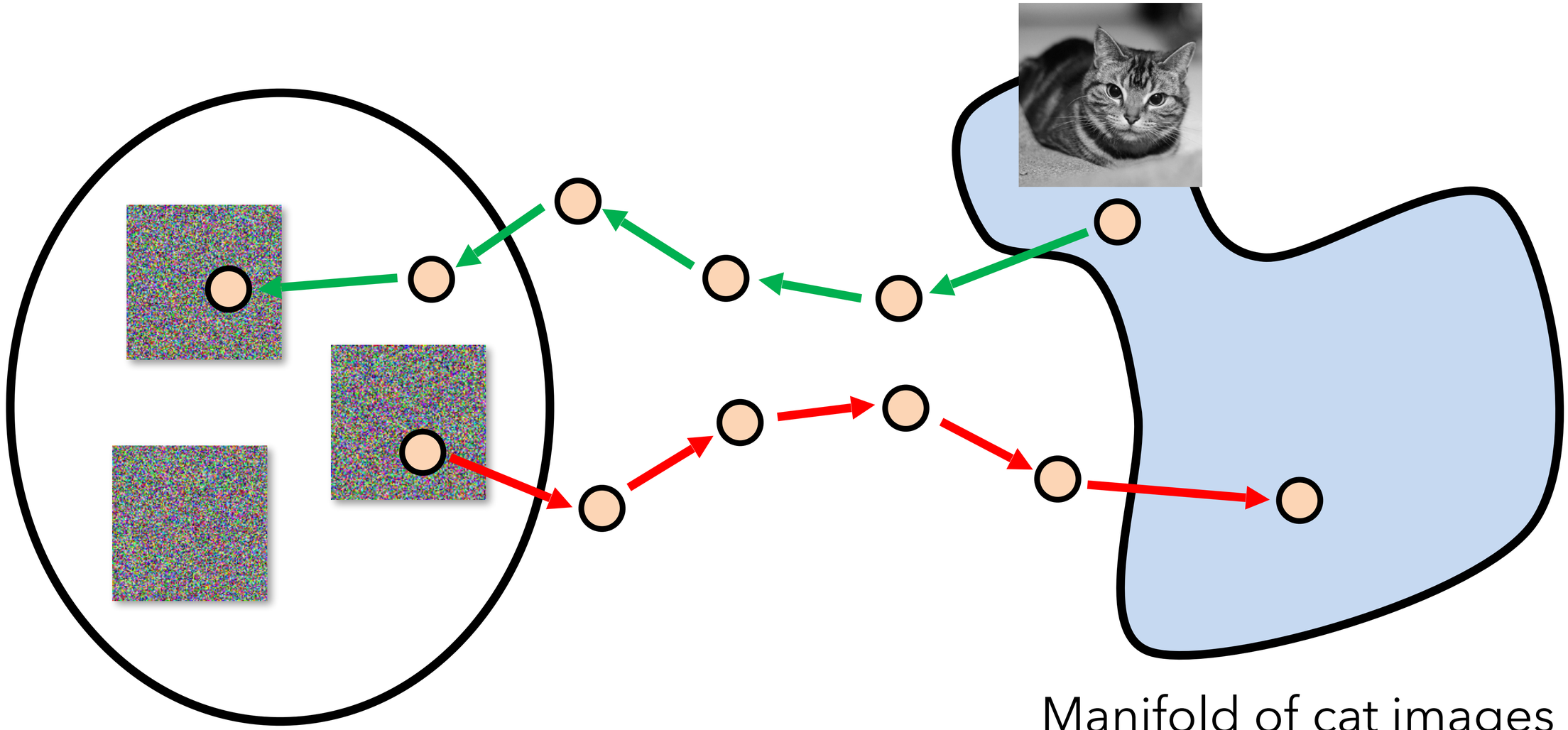


Manifold of cat images

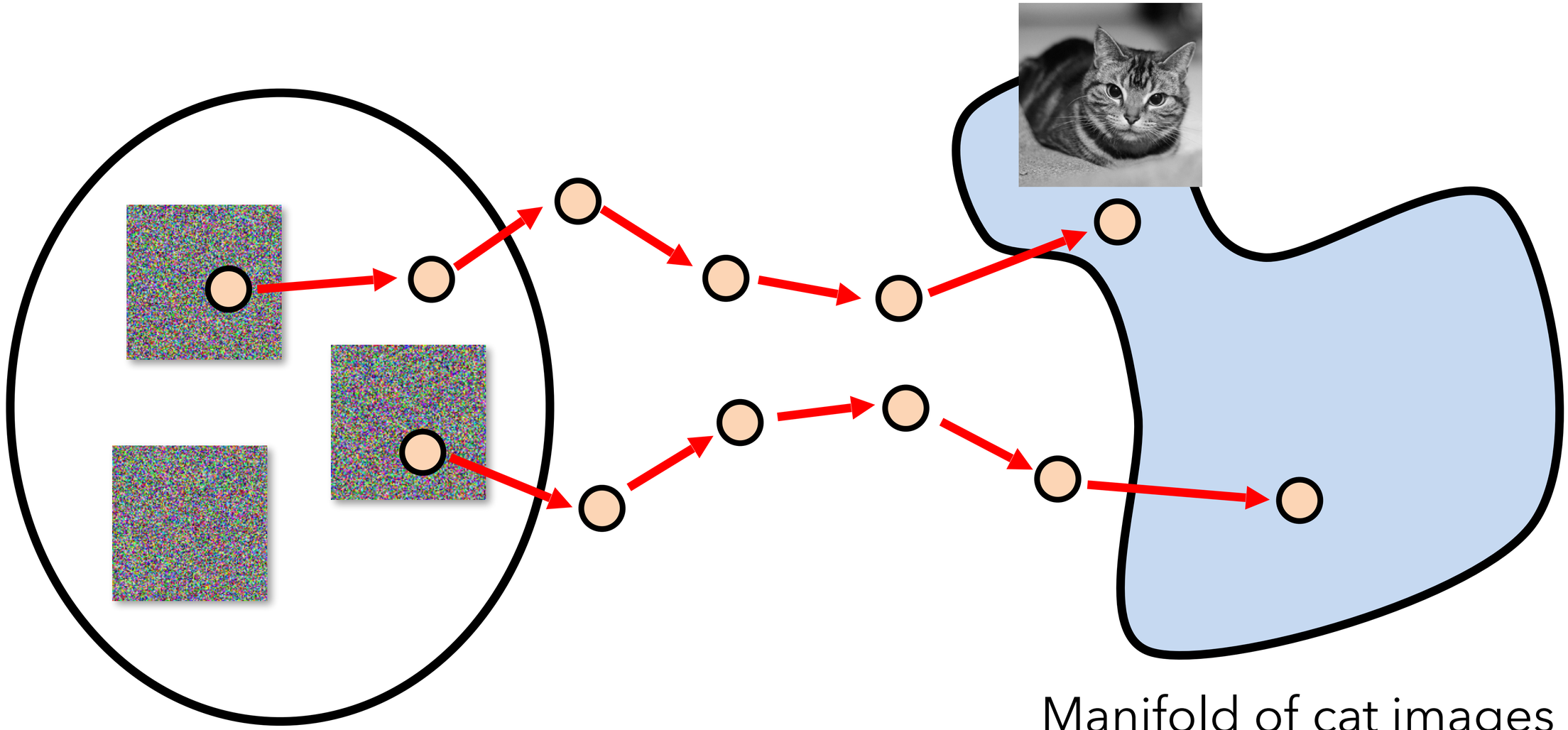
Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz



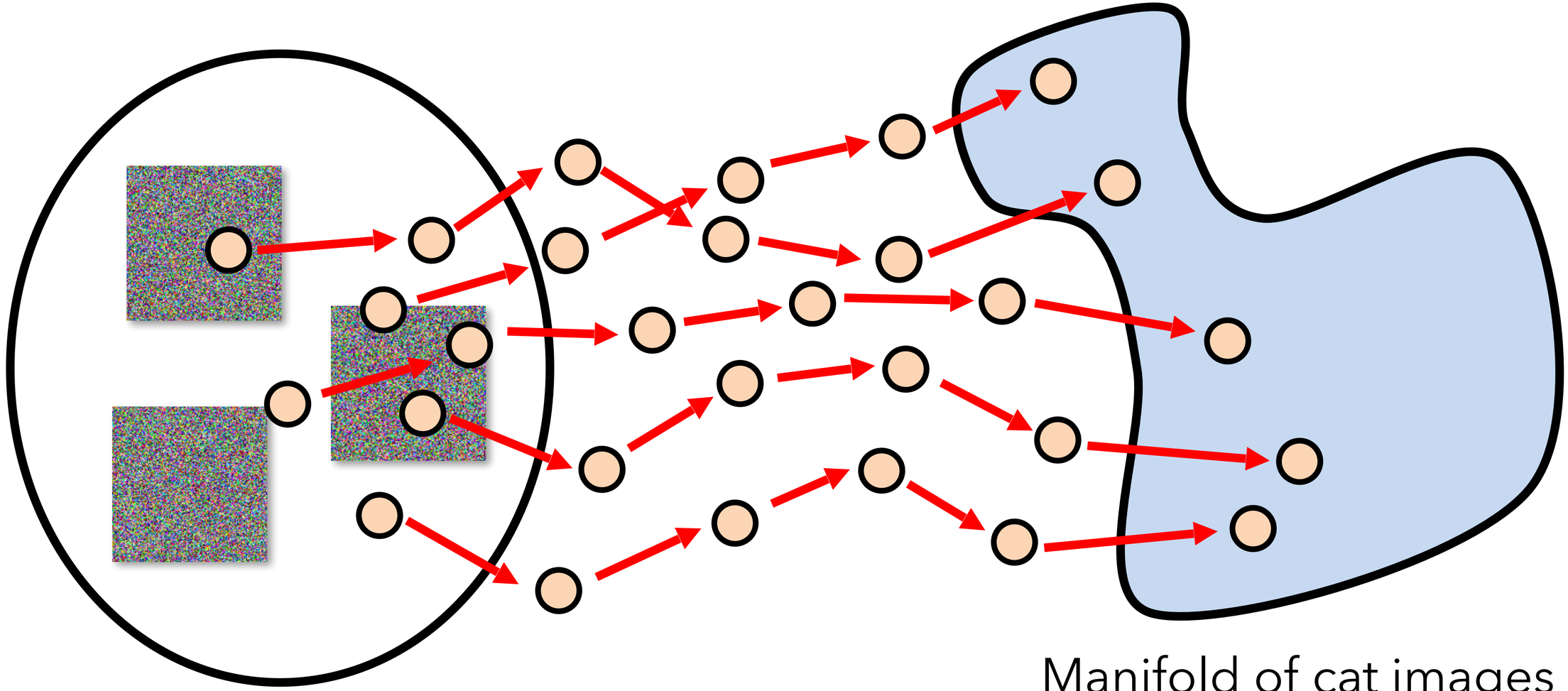
Slide source: N. Snavely & S. Seitz



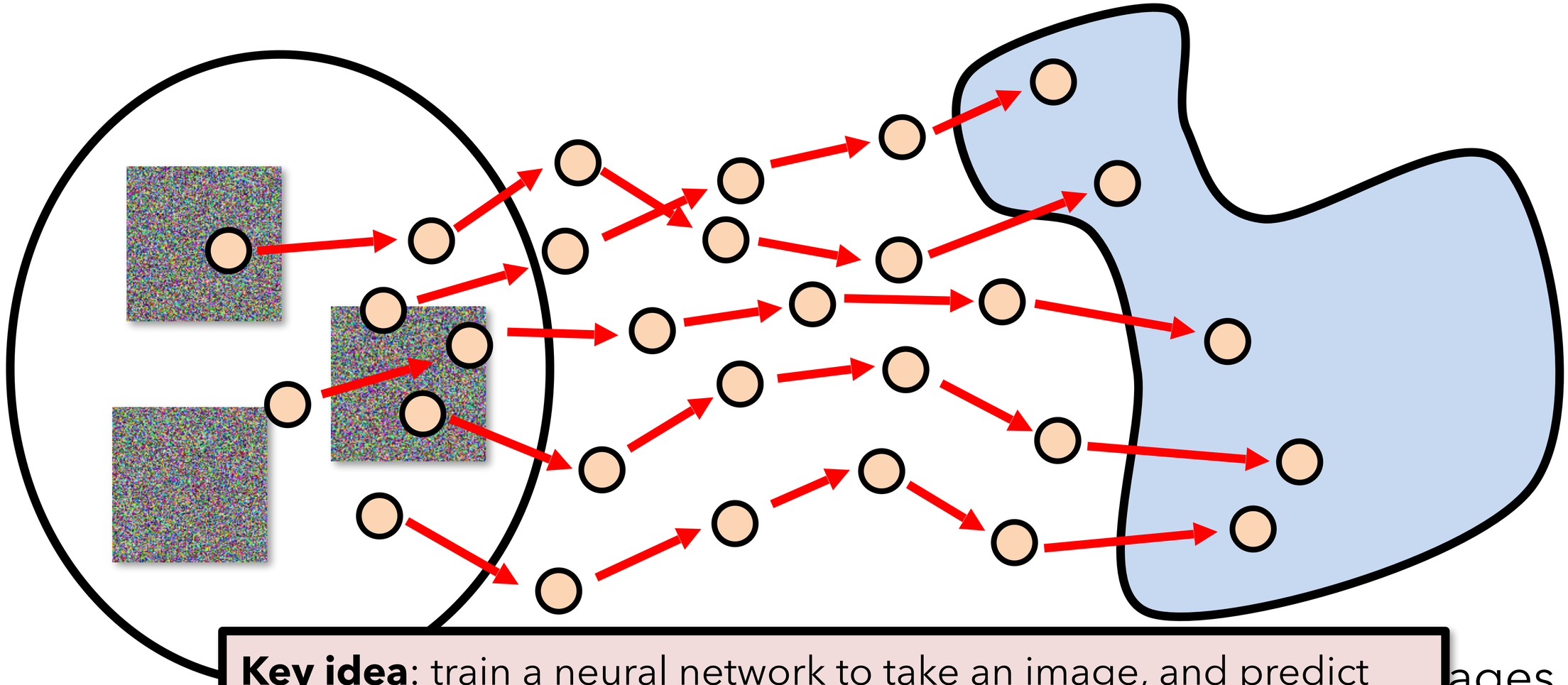
Random images

Manifold of cat images

Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz



Key idea: train a neural network to take an image, and predict the arrows above; that is, predict to convert a noisy image to a slightly less noisy image that is closer to the desired image manifold, using the example above to train.

Slide source: N. Snavely & S. Seitz

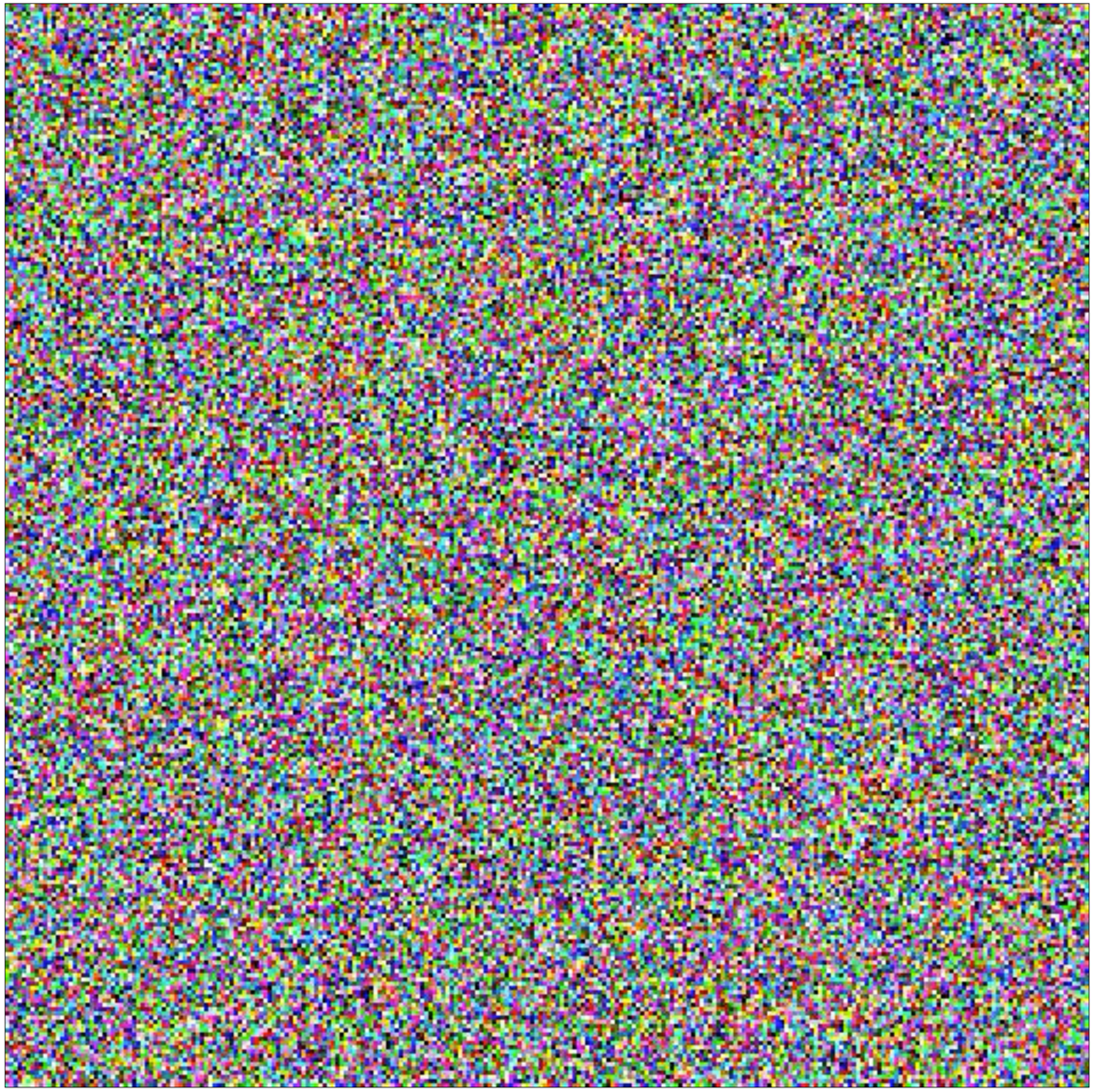
Denoising diffusion neural network

Diffusion neural network

This network can be a U-Net or other suitable image-to-image network

Slide source: N. Snavely & S. Seitz

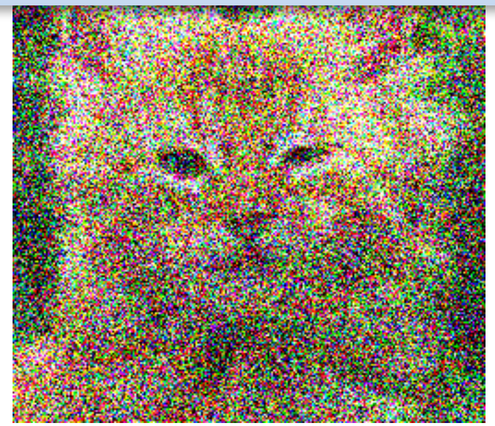
Running a diffusion model for multiple steps



Example source: Aditya Ramesh

Denoising diffusion neural network

Timestep *t* (from say 0 to 999) is an additional input to the network (positionally encoded)



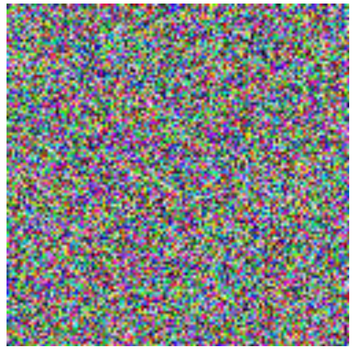
t

Diffusion neural network

A basic diffusion approach will run this denoising for many timesteps (e.g., $T = 1000$ steps)

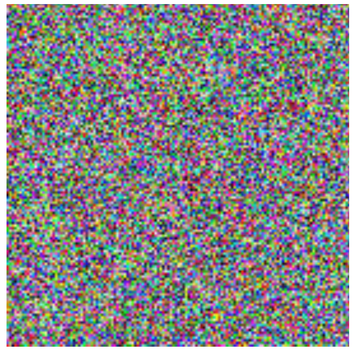
Slide source: N. Snavely & S. Seitz

Sampling many outputs



Noise image 1

Output image 1

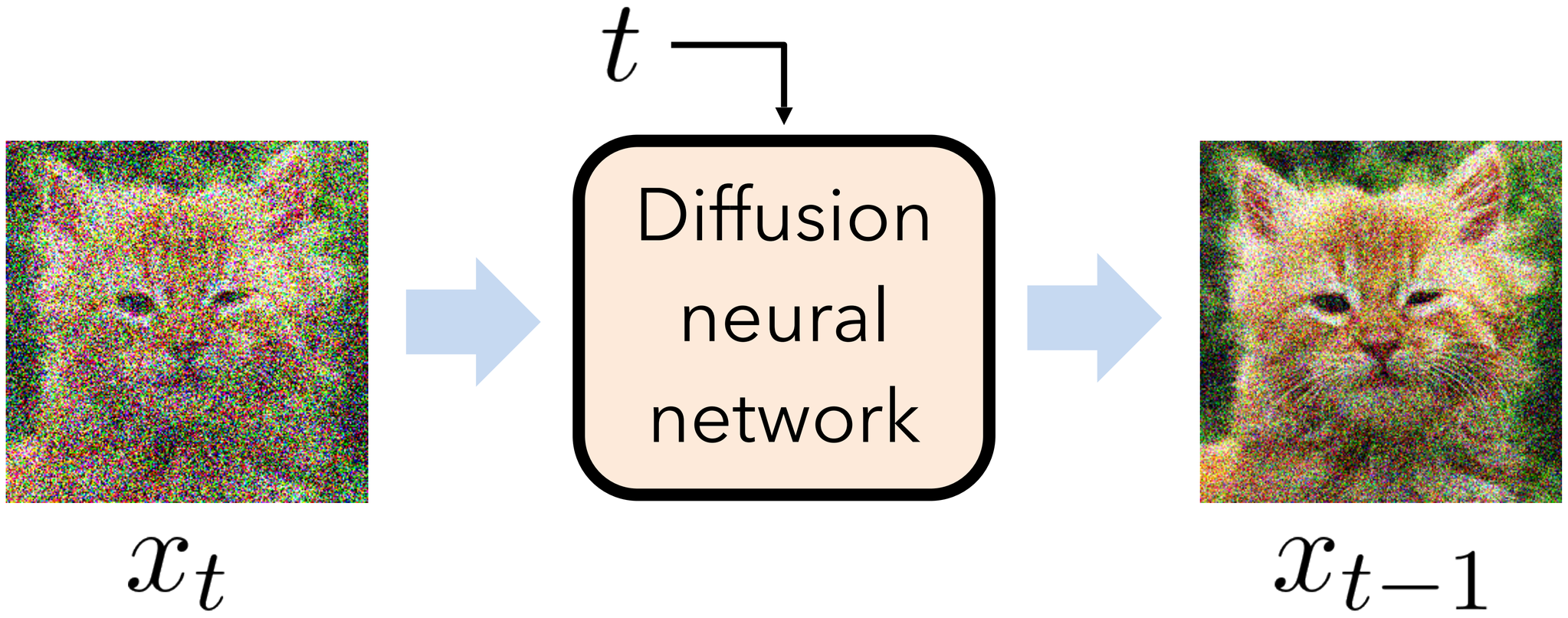


Noise image 2

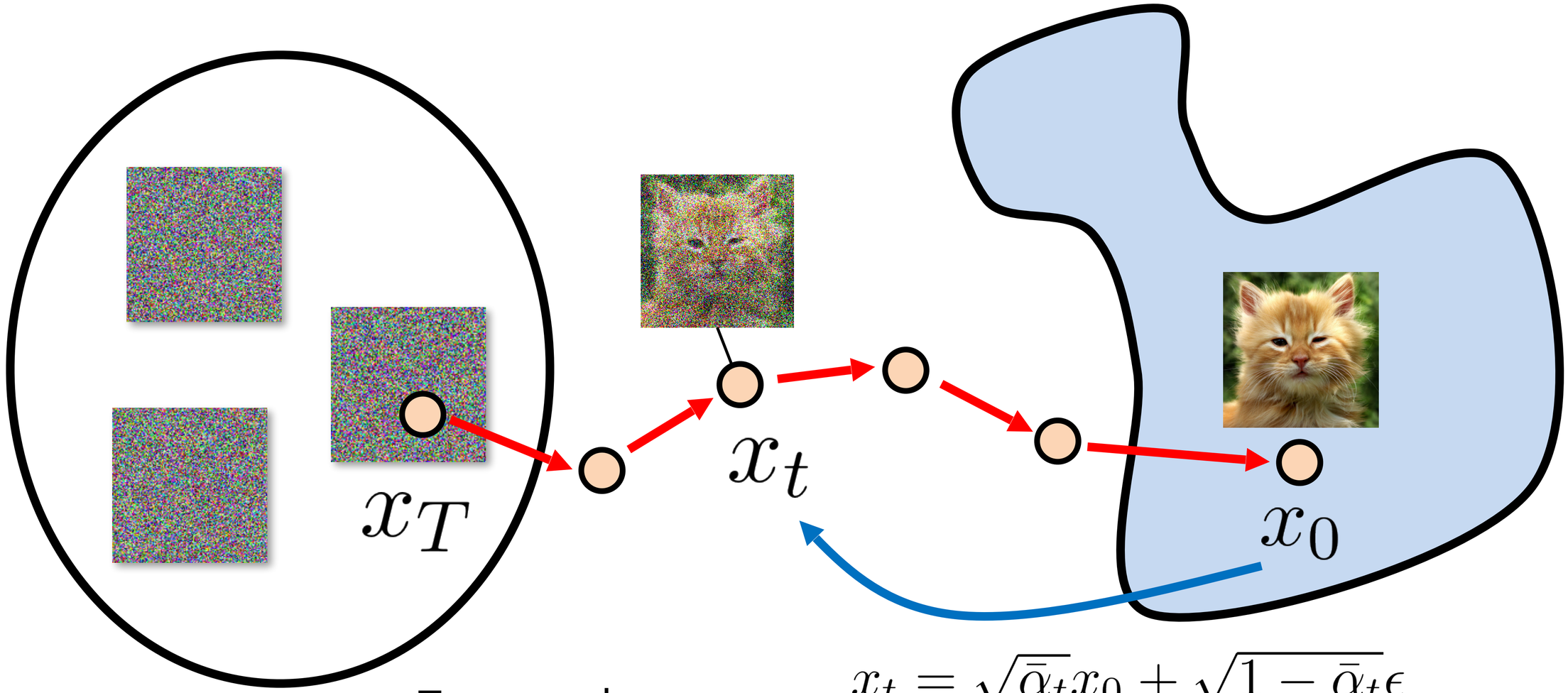
Output image 2

Slide source: N. Snavely

Let’s make this idea more concrete



Adapted from: N. Snavely



Forward process: $x_t = \sqrt{\bar{\alpha}_t} x_0 + \sqrt{1 - \bar{\alpha}_t}\epsilon$

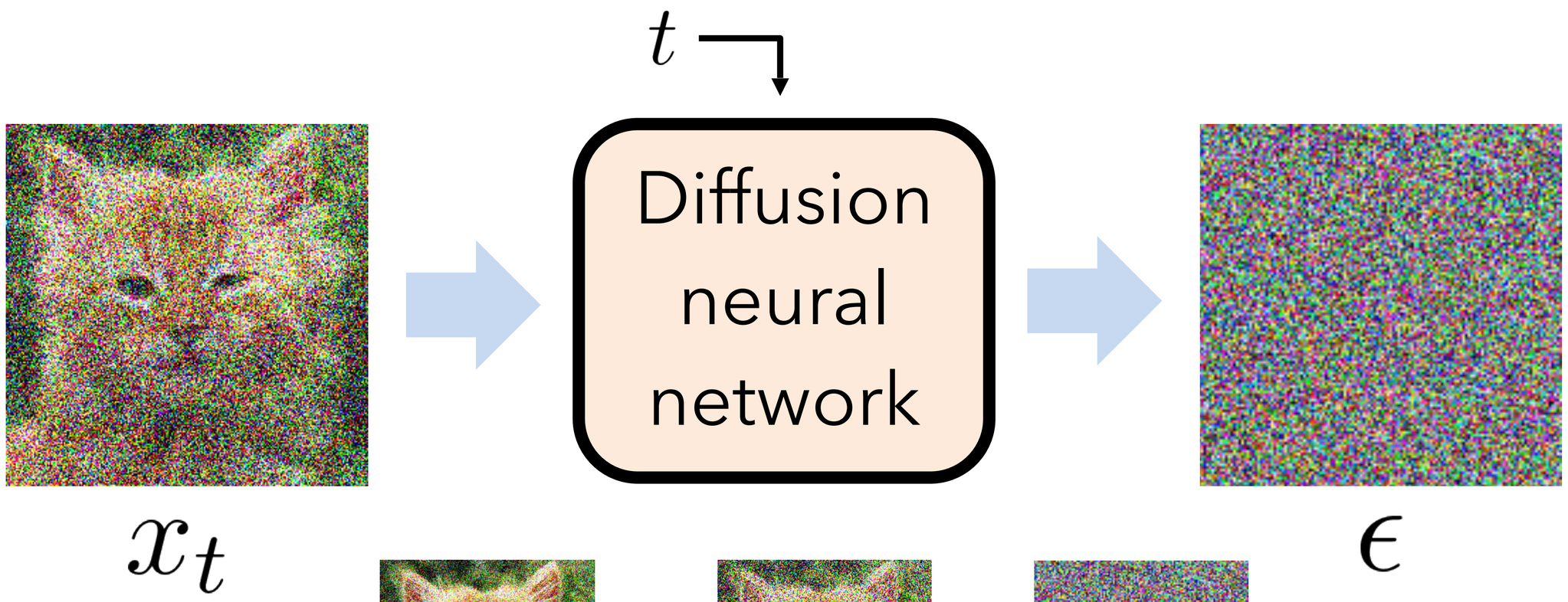
where $\epsilon \sim \mathcal{N}(0, 1)$

$\bar{\alpha}_t$: noise coefficients per time step (magic numbers defined by the designers)

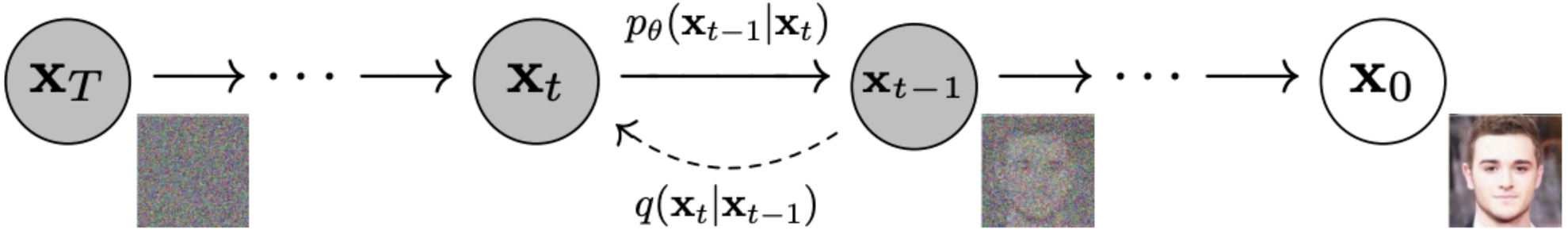
(Gaussian noise image)

Adapted from N. Snavely

In practice, we predict the *noise*



Training a diffusion model



Algorithm 1 Training

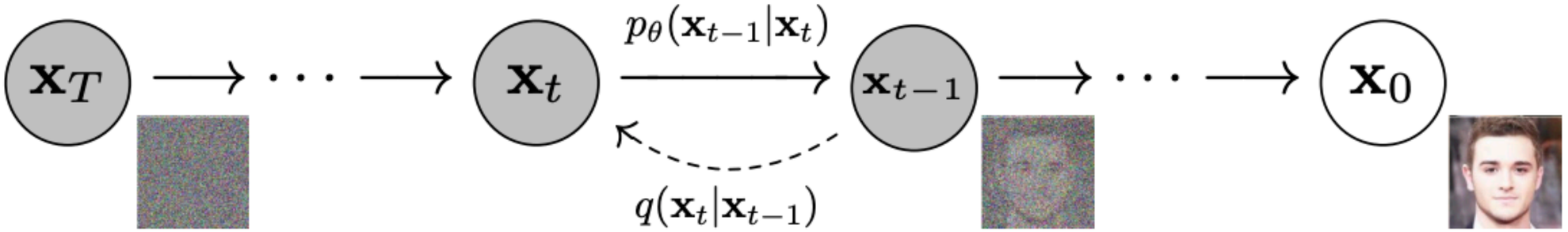
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

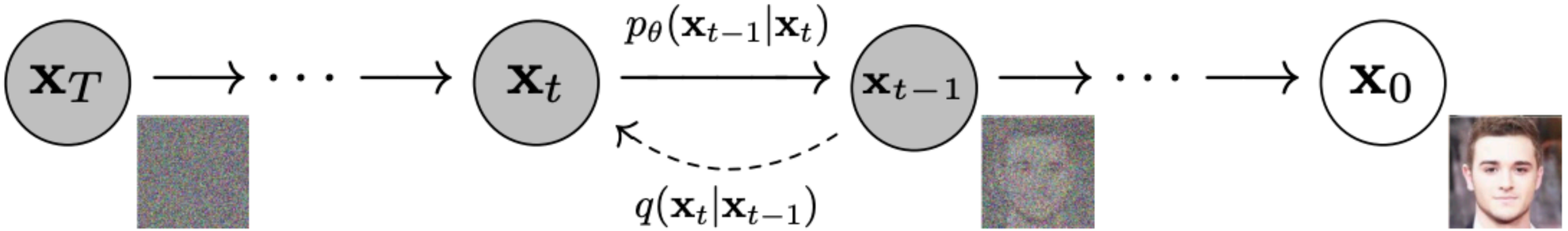
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Sample x_t by adding noise to a clean image x_0

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

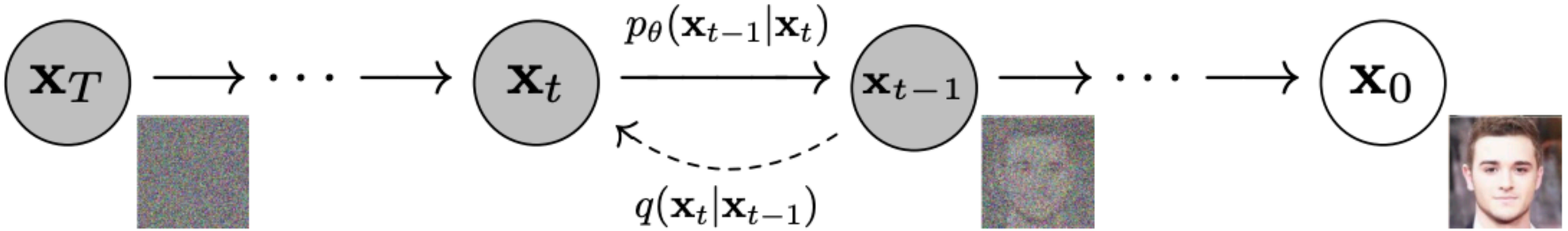
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Random noise

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

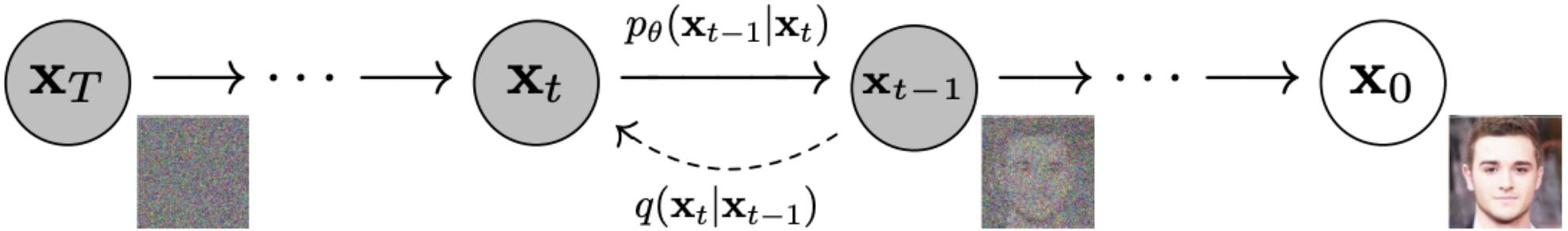
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

6: **until** converged

Noisy image

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

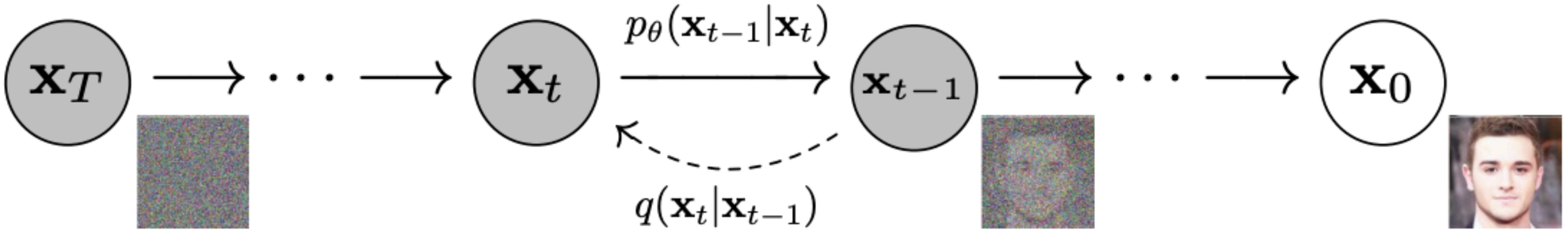
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2$$

6: **until** converged

Diffusion model ϵ_θ predicts the noise ϵ

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Training a diffusion model



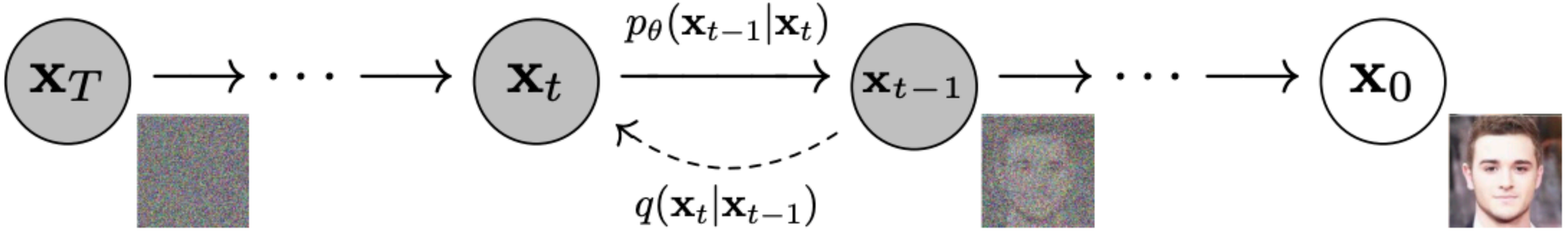
Algorithm 1 Training

1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
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

Make the diffusion model better at denoising this image!

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Sampling from a diffusion model



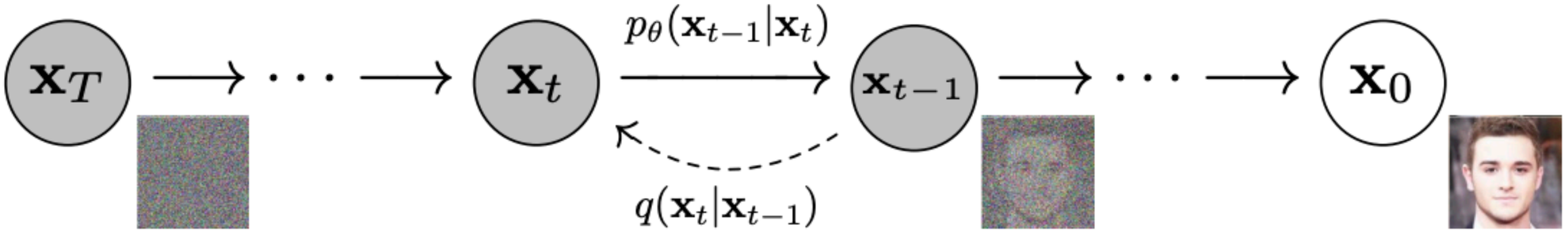
Algorithm 2 Sampling

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** $\mathbf{x}_0$

Remove a little bit of the noise

Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

Sampling from a diffusion model



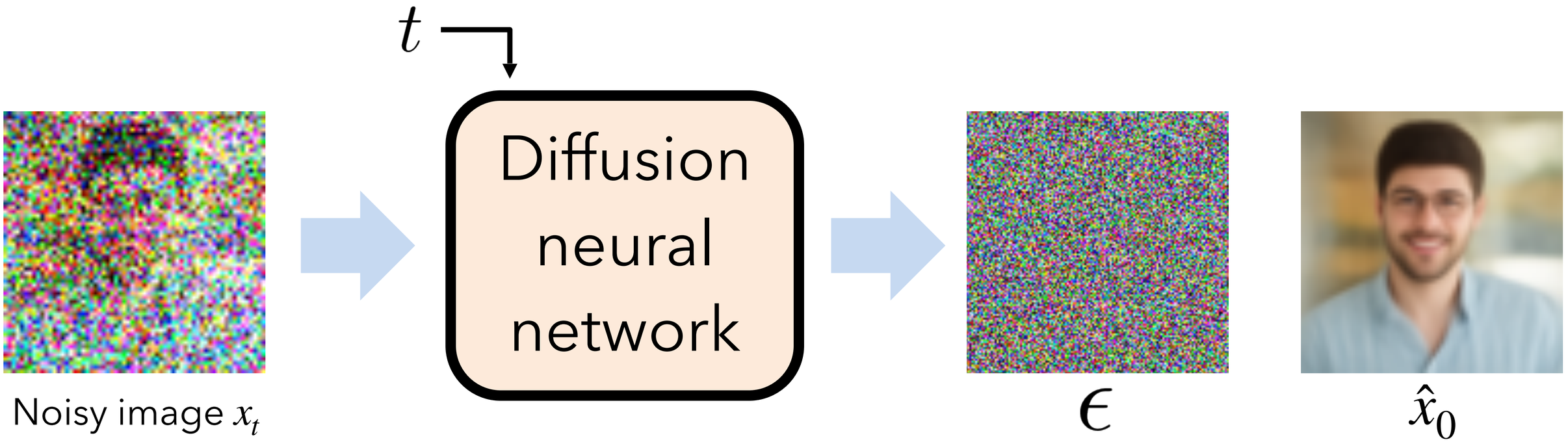
Algorithm 2 Sampling

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** $\mathbf{x}_0$

Add back a bit of noise, too.
(some diffusion variants don't do this)

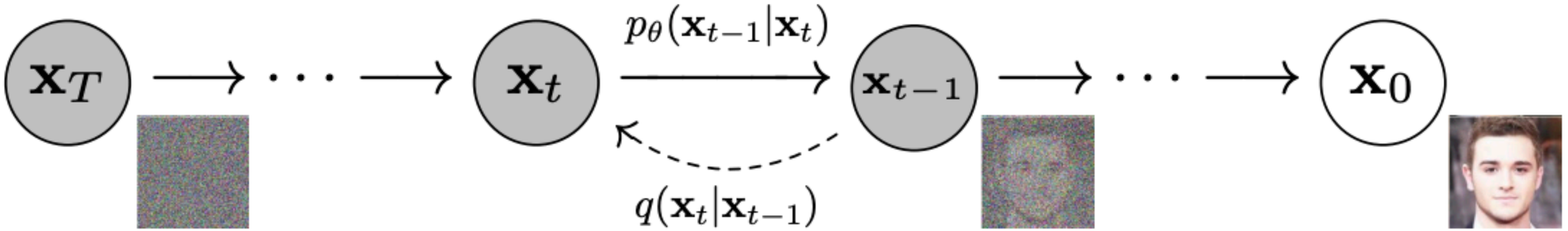
Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

One-step denoising



where $\hat{x}_0 = \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{(1-\alpha_t)}\epsilon_\theta(x_t, t)) = \mathbb{E}[x_0 \mid x_t]$ is "one step" prediction.

Sampling from a diffusion model



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** $\mathbf{x}_0$

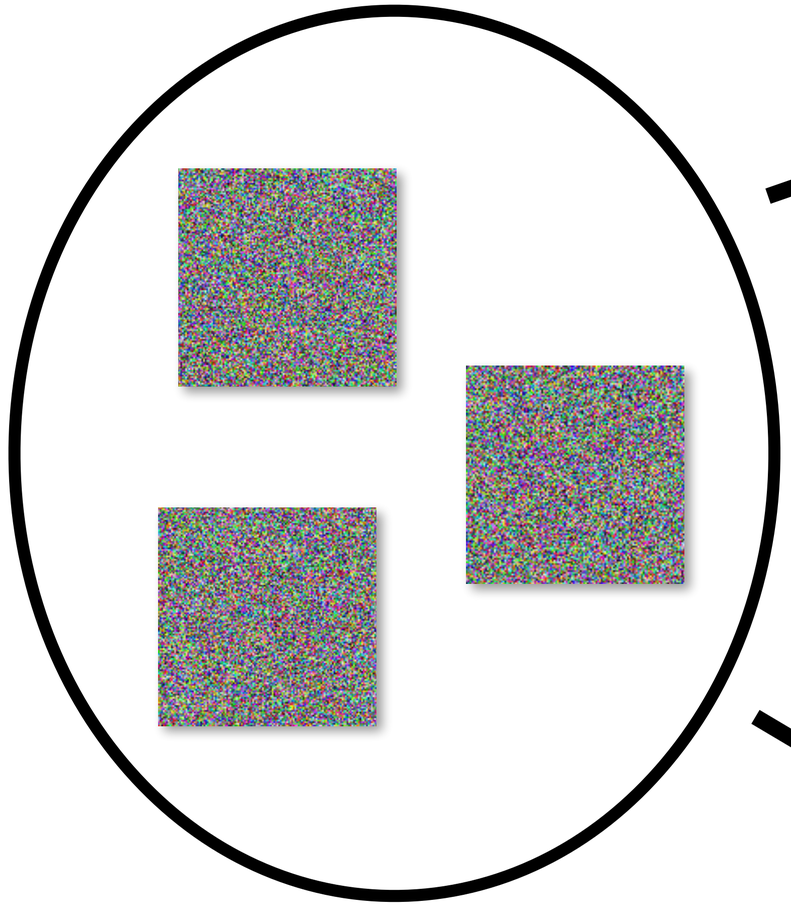
Can write update as:

$$x_{t-1} = w_0 \hat{x}_0 + w_1 x_t + \sigma_t z$$

for constants w_0, w_1 where $\hat{x}_0$ is "one step" estimate of clean image:

$$\hat{x}_0 = \frac{1}{\sqrt{\alpha_t}}(x_t - \sqrt{(1-\alpha_t)}\epsilon_\theta(x_t, t))$$

Figure from [J. Ho et al, "DDPM", 2020]



Random images

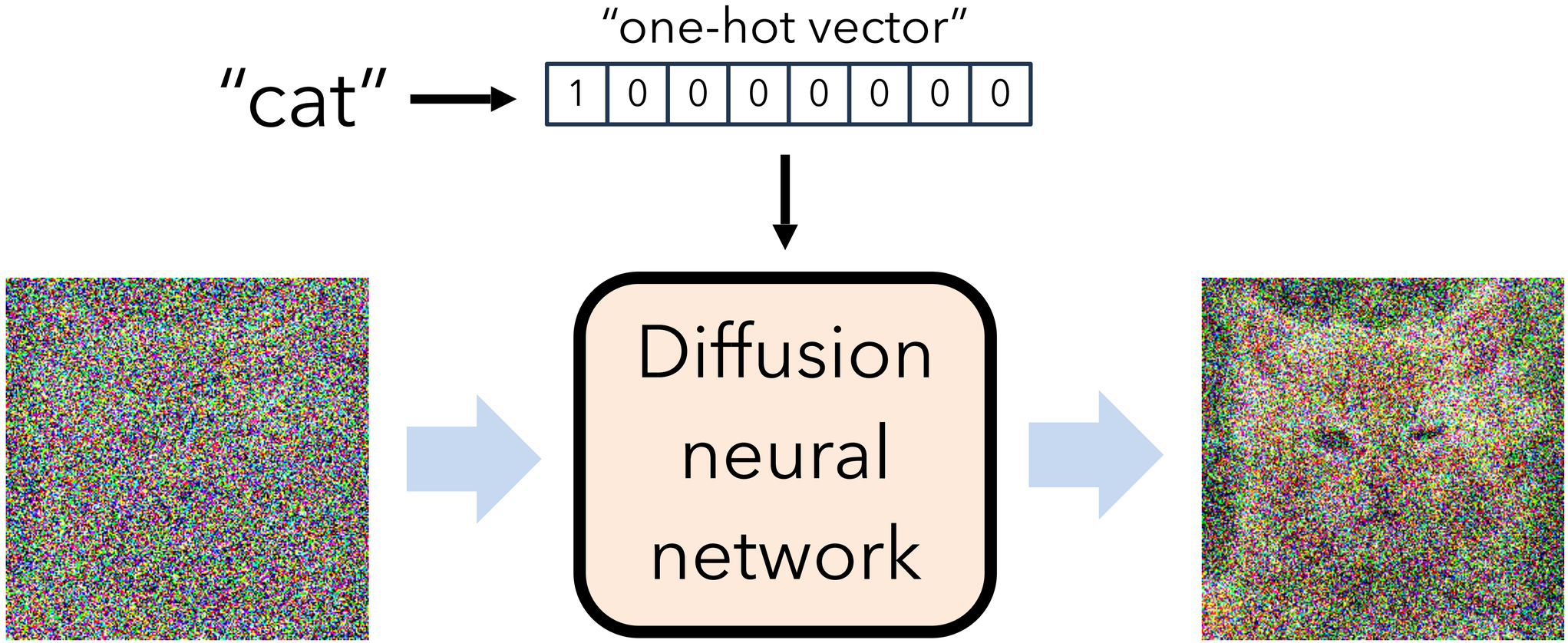
cat images

llama images

How can we avoid training a separate diffusion network for each concept?

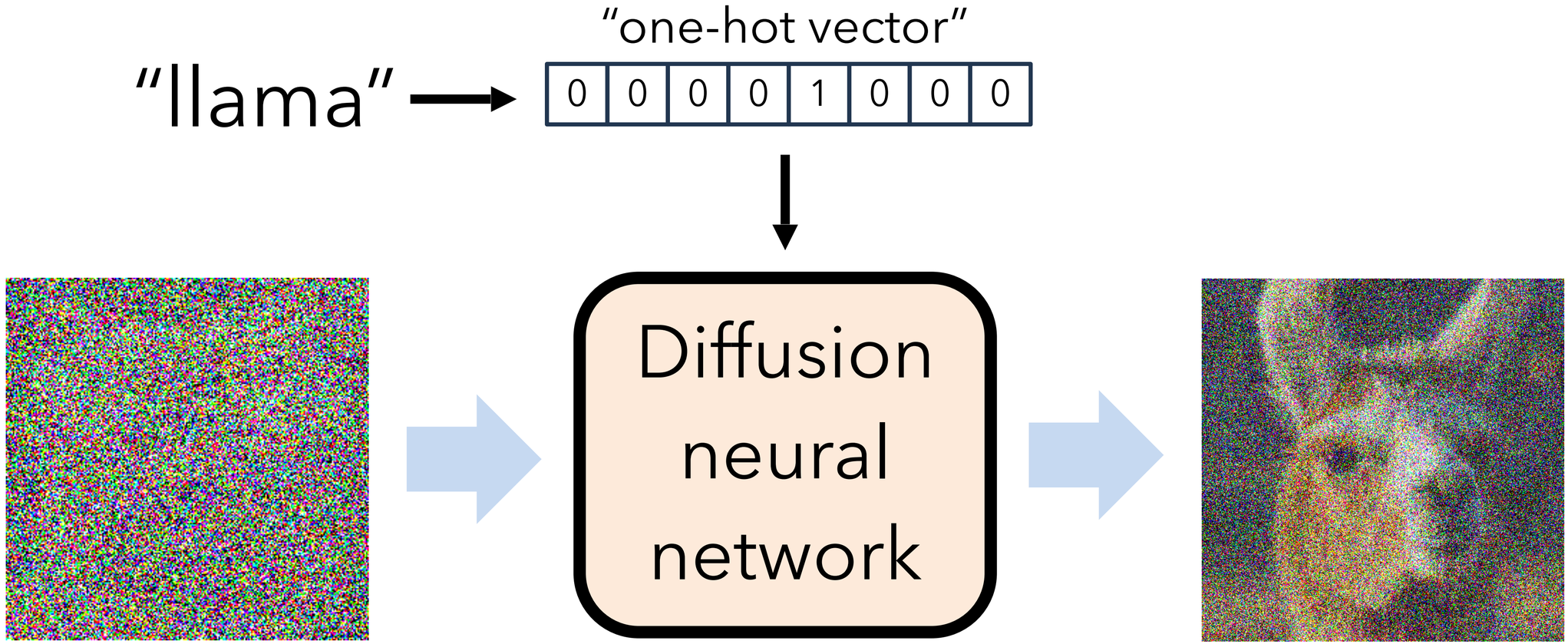
Slide concept: Steve Seitz

Idea 1: add a class label as conditioning



Source: N. Snavely

Idea 1: add a class label as conditioning



"Class conditioning" – only allows us to generate a fixed set of classes

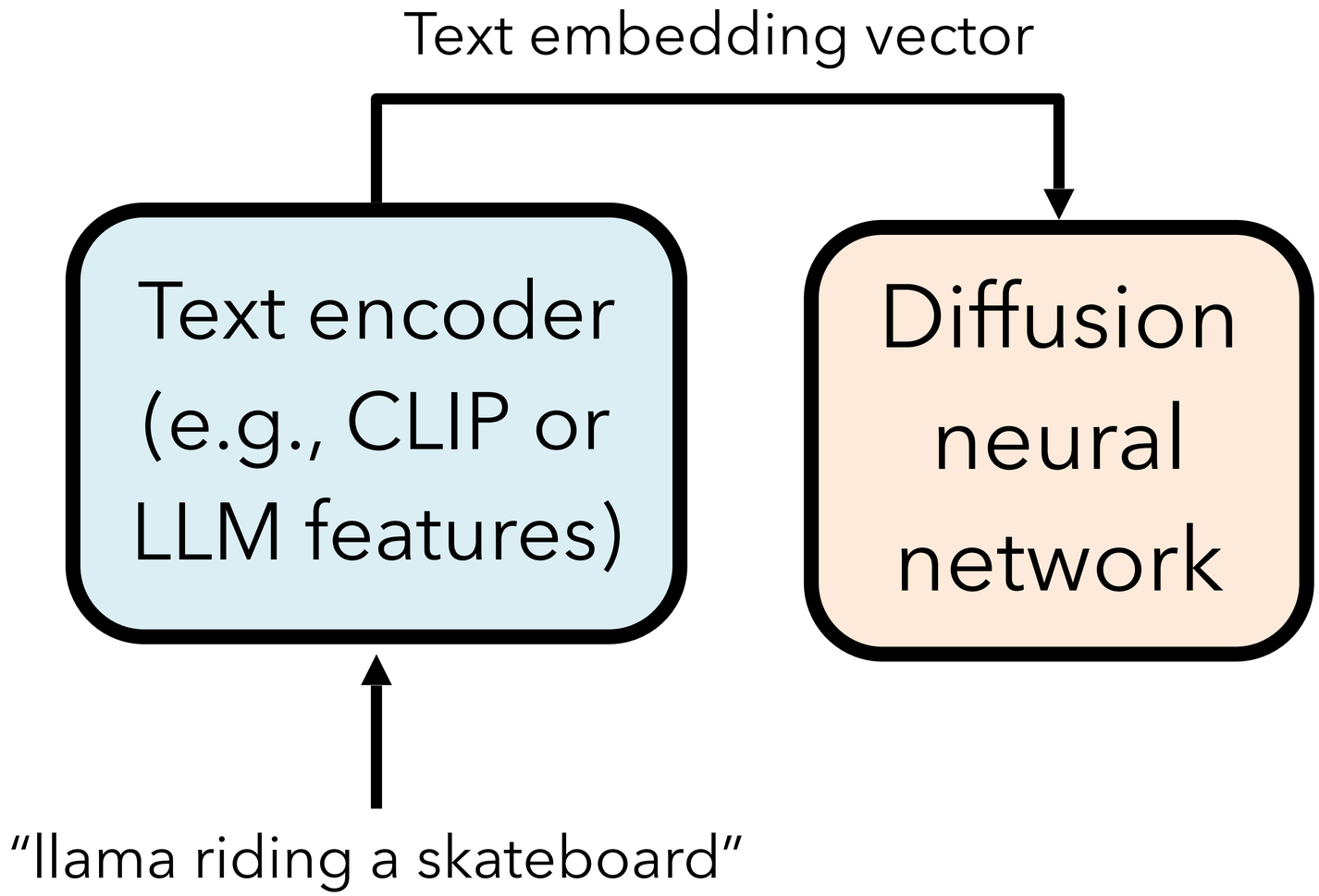
Source: N. Snavely

Works but doesn’t always produce high quality images

“Malamute” class

Figure source: [Ho and Salimans, 2021]

Idea 2: add a text prompt as conditioning

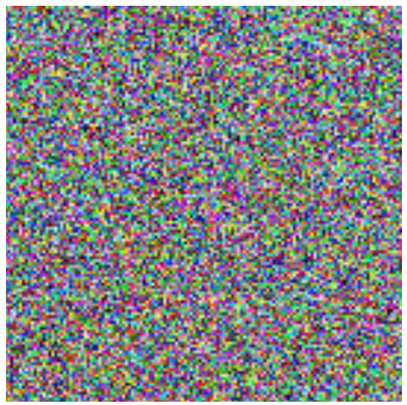


Adapted from N. Snavely

Idea 3: use an image classifier to guide the denoising process (“classifier guidance”)

$$= \frac{1}{\sqrt{\alpha_t}} x_t - \frac{1-\alpha_t}{\sqrt{\alpha_t}\sqrt{1-\bar{\alpha}_t}} \epsilon$$

x_{t-1}



+ Update image that makes the input look more like a cat

Source: N. Snavely

Idea 3: use an image classifier to guide the denoising process ("classifier guidance")

Update image that makes the input look more like a cat

$$= \nabla F_\theta(x_t | y = \text{“}cat\text{”})$$

F_θ : Image classifier, e.g., AlexNet trained on ImageNet, but fine-tuned for noisy images

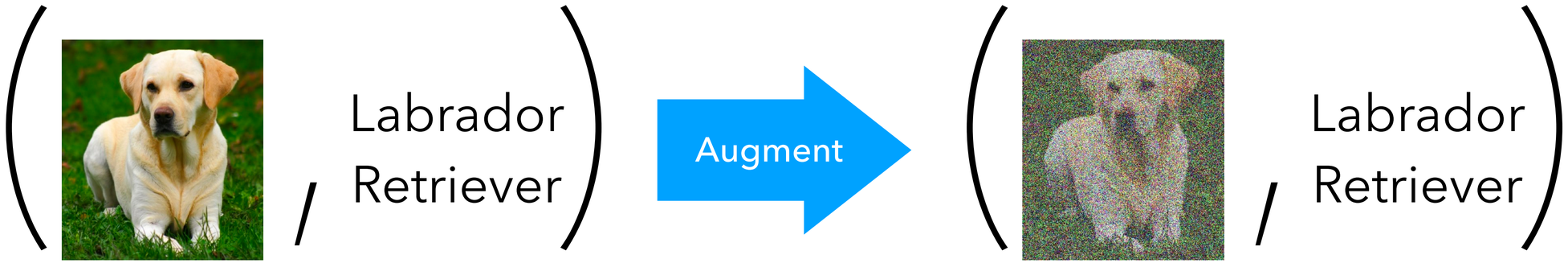
$$\tilde{\epsilon}_t(x_t, t, y) = \epsilon_\theta(x_t, t) - \gamma \nabla_{x_t} \log p(x_t | y)$$

Adapted from: N. Snavely

Classifier Guidance

Problem: Classifier isn't trained on noisy images!

Solution: Finetune the classifier on noisy images



Slide credit: Daniel Geng

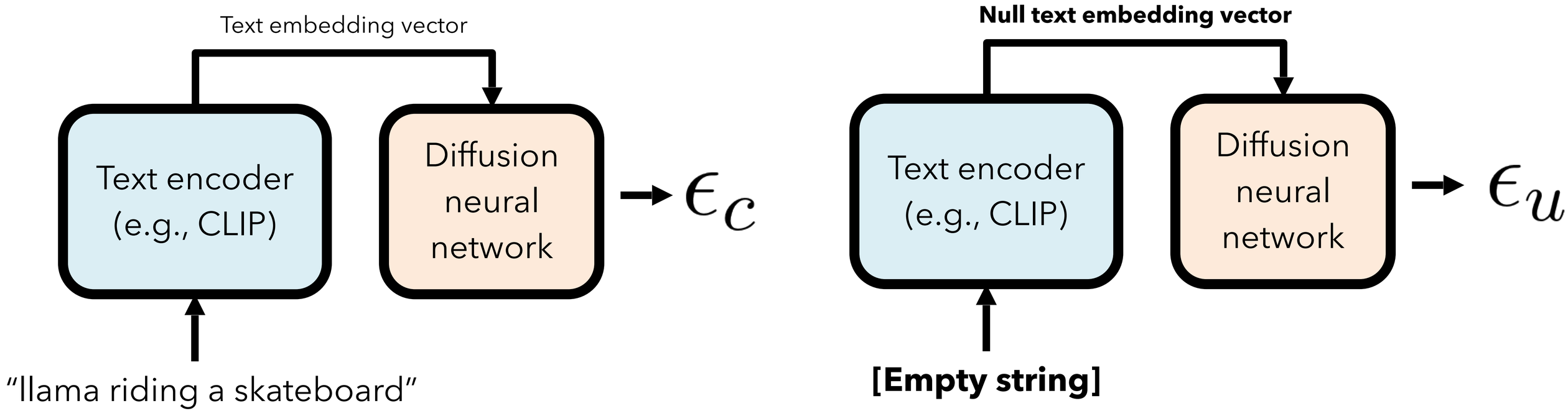
Classifier Guidance

Guidance Weight 1.0

Guidance Weight 10.0

Slide: Daniel Geng, Source: Dhariwal and Nichol, “Diffusion Models Beat GANs on Image Synthesis”

Idea 4: Classifier-free guidance (CFG)



Mix conditional and unconditional noise:

$$\epsilon = \epsilon_u + \gamma(\epsilon_c - \epsilon_u) \quad \text{where } \gamma > 1$$

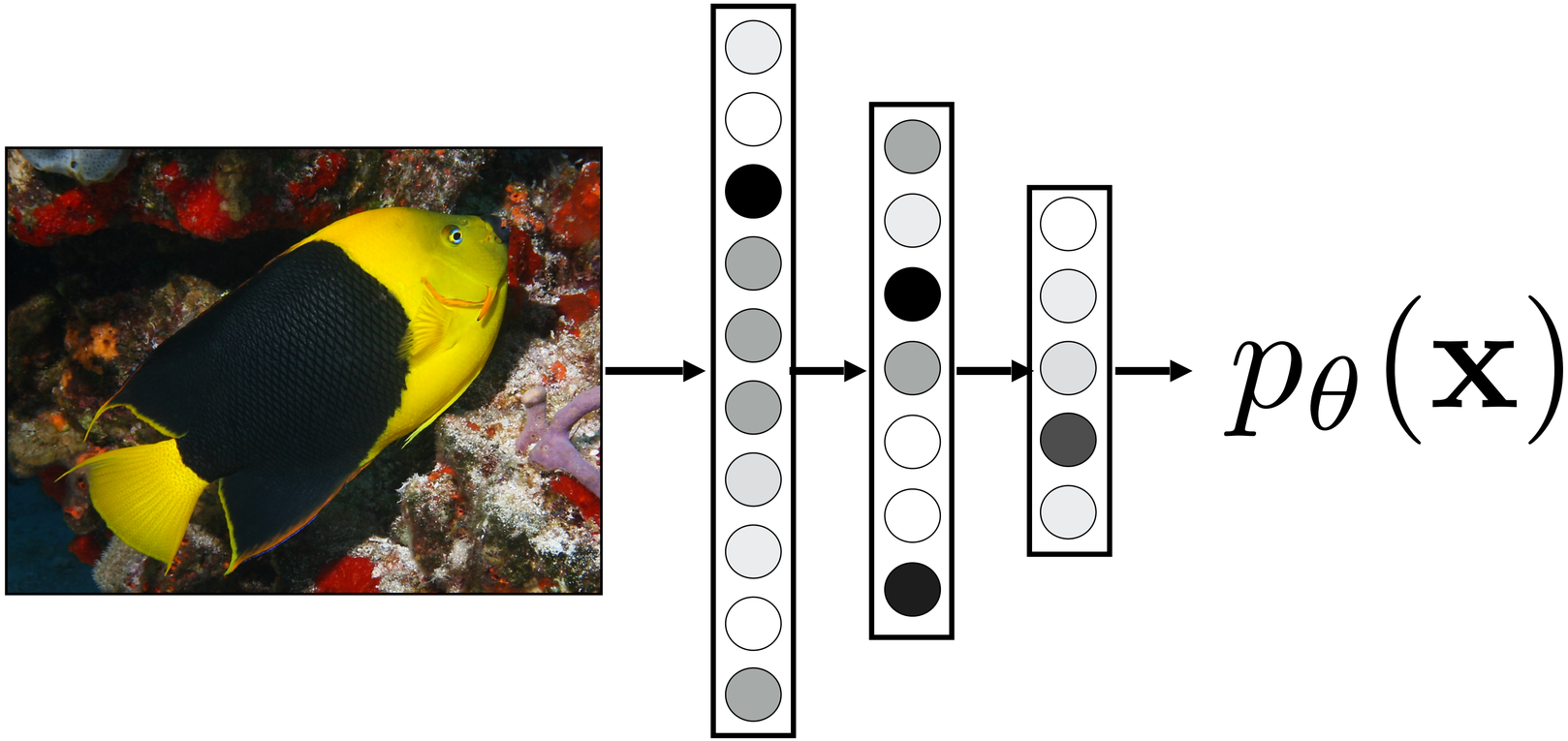
Classifier-free Guidance

Low γ

High γ

Figure source: [Ho and Salimans, 2021]

Another perspective on diffusion models



How would such a model work?

Energy-based model (EBM)

Energy function implemented by neural net, $E_\theta(\mathbf{x}) : \mathbb{R}^n \to \mathbb{R}$

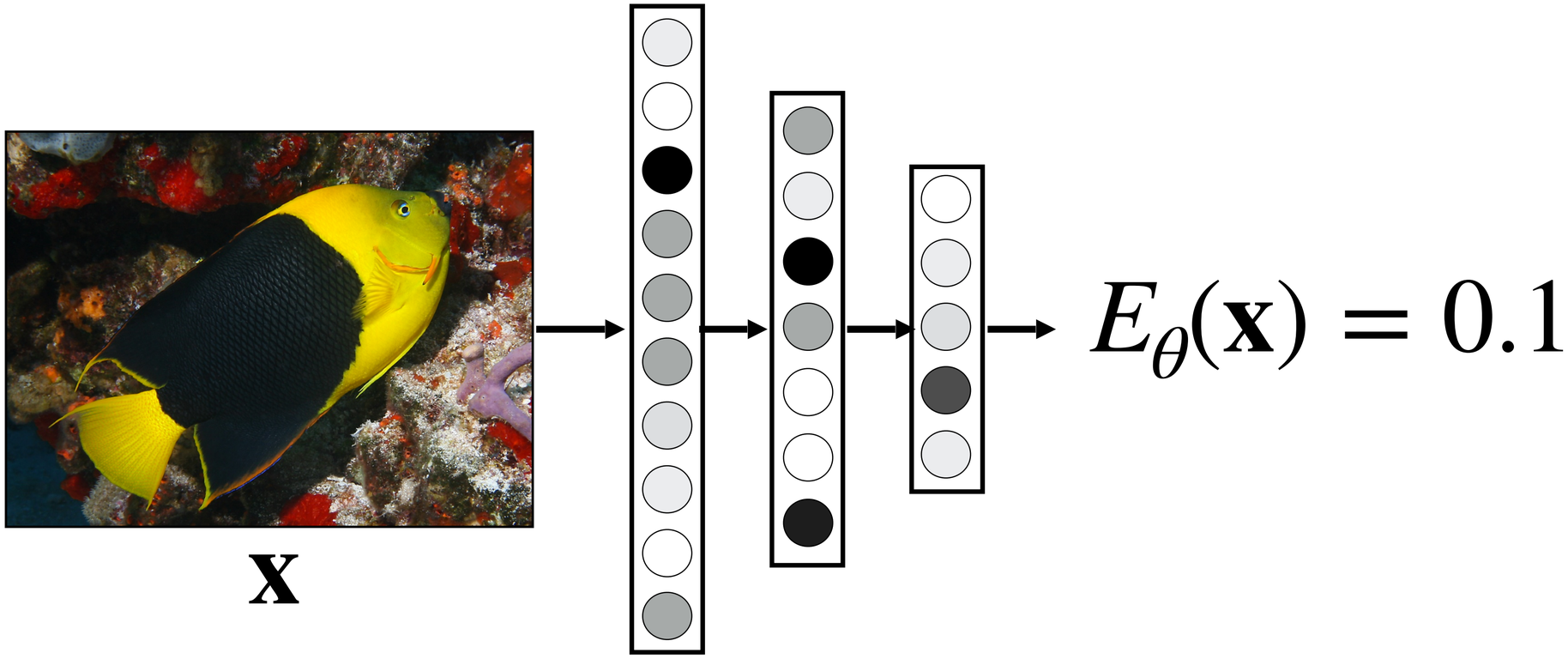
$$p_\theta(\mathbf{x}) = \frac{\exp(-E_\theta(\mathbf{x}))}{Z(\theta)}$$

Normalization constant a.k.a. **partition function**

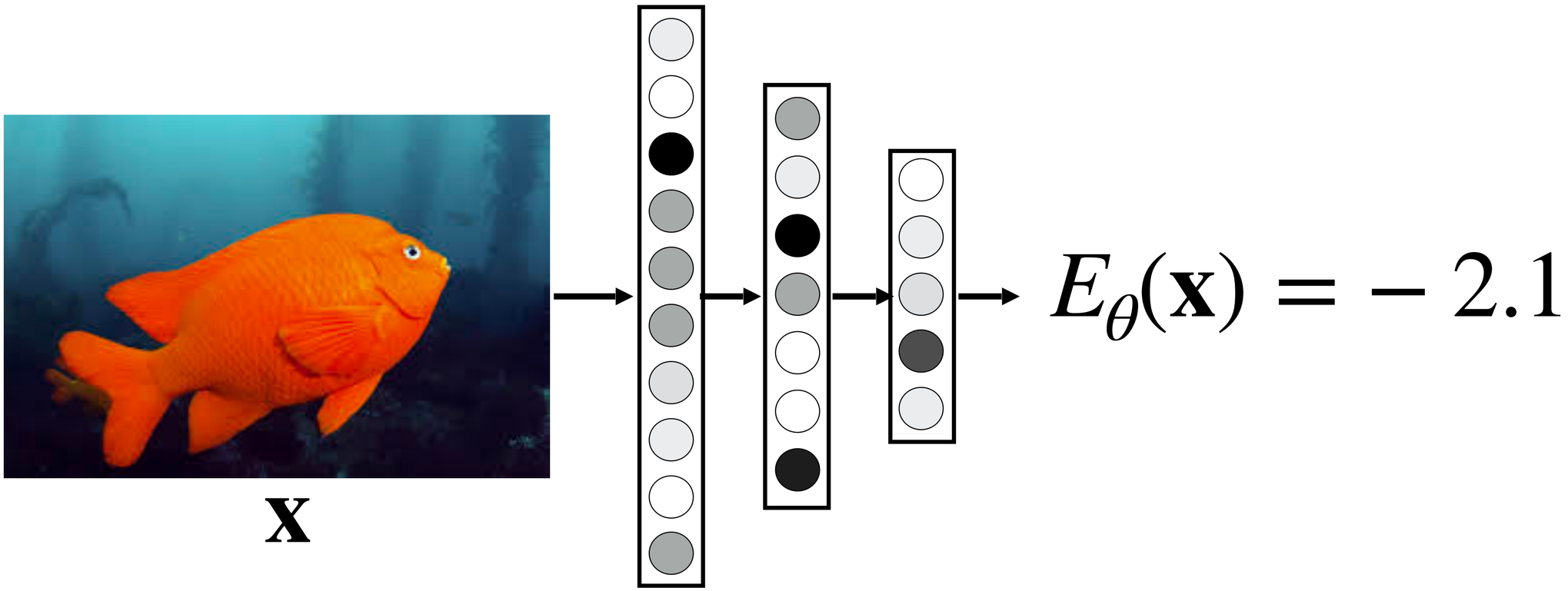
where $Z(\theta) = \int_{\mathbf{x}} \exp(-E_\theta(\mathbf{x}))d\mathbf{x}$

What makes this idea challenging? Needs to sum to 1!

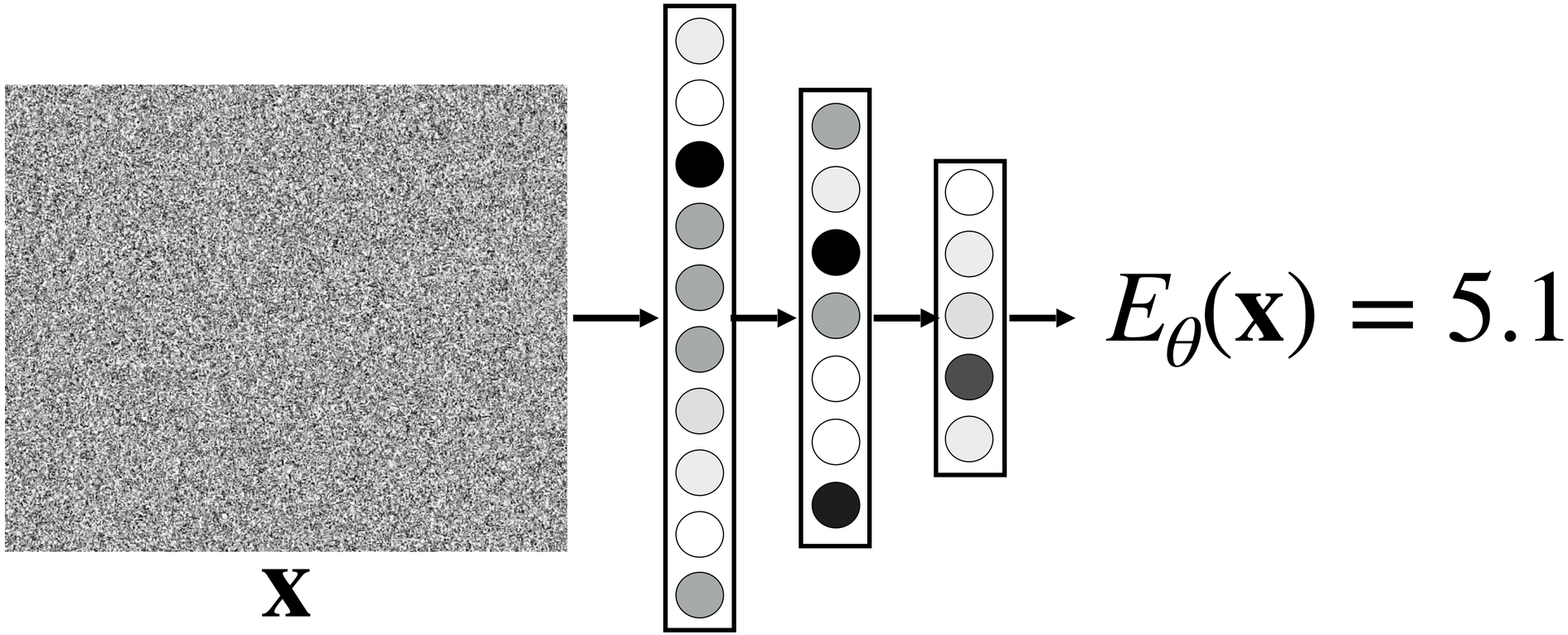
Energy function



Energy function



Energy function



When computing $Z(\theta)$, we need to integrate over the full input space, which makes it challenging.

Denoisers as score functions

- The **score function** $s_\theta(\mathbf{x}) = \nabla_\mathbf{x} \log(p_\theta(\mathbf{x}))$ generally much easier to work with.

  For example, in EBMs:

$$\nabla_\mathbf{x} \log(p_\theta(\mathbf{x})) = \nabla_\mathbf{x} \log \left(\frac{\exp(-E_\theta(\mathbf{x}))}{Z(\theta)} \right) = \nabla_\mathbf{x} \exp(-E_\theta(\mathbf{x}))$$

- So, if we could learn the score function, that would that would circumvent the need for computing the normalization constant.

- You can show that the denoiser learns the score function of the noised data distribution by Tweedie's formula [Robbins, 1956]:

$$\epsilon_\theta(\mathbf{x}_t) \approx -\sigma_t \nabla_{\mathbf{x}_t} \log p(\mathbf{x}_t)$$

  where $\mathbf{x}_t$ is the data noised at step t of the diffusion model with Gaussian noise σ_t.

- So, when we remove noise, we are (roughly speaking) following the gradient of log prob. and moving to higher density regions.

Score function perspective on classifier-free guidance

Classifier-free guidance:

$$\epsilon_\theta(\mathbf{x}_t) + \gamma(\epsilon_\theta(\mathbf{x}_t, c) - \epsilon_\theta(\mathbf{x}_t))$$

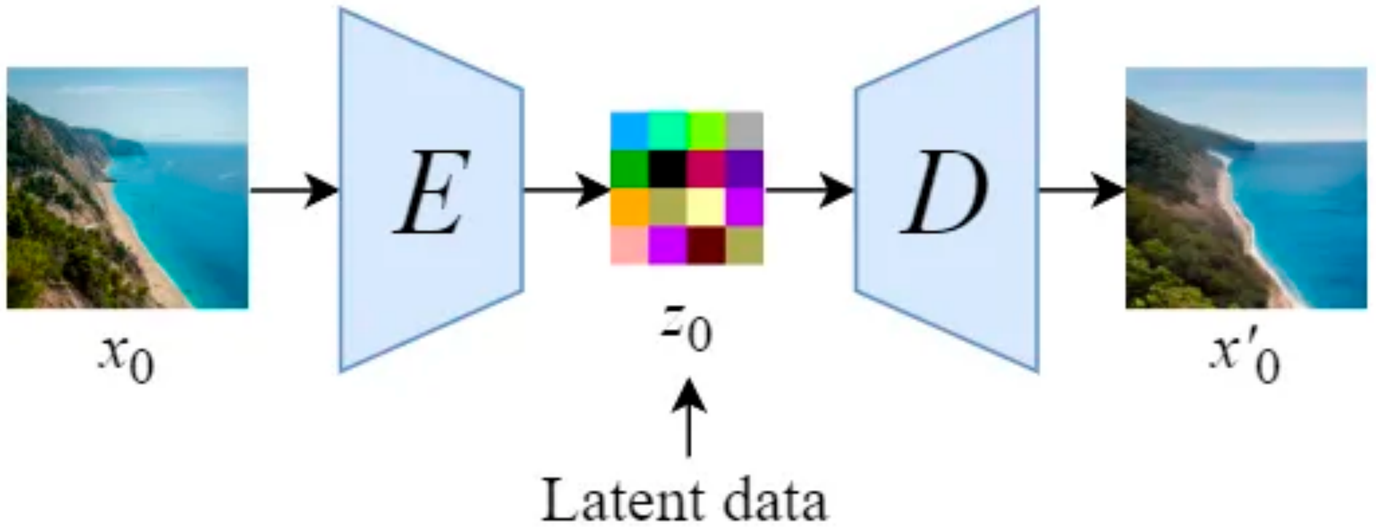
From a score function perspective this is (approximately):

$$\approx -\nabla_{\mathbf{x}_t}[\log p_\theta(\mathbf{x}_t) + \gamma\,(\log p_\theta(\mathbf{x}_t \mid y) - \log p_\theta(\mathbf{x}_t))]$$

$$= -\nabla_{\mathbf{x}_t}\,\log\big(p_\theta(\mathbf{x}_t)^{1-\gamma}\,p_\theta(\mathbf{x}_t \mid y)^\gamma\big)$$

Latent diffusion models (LDMs)

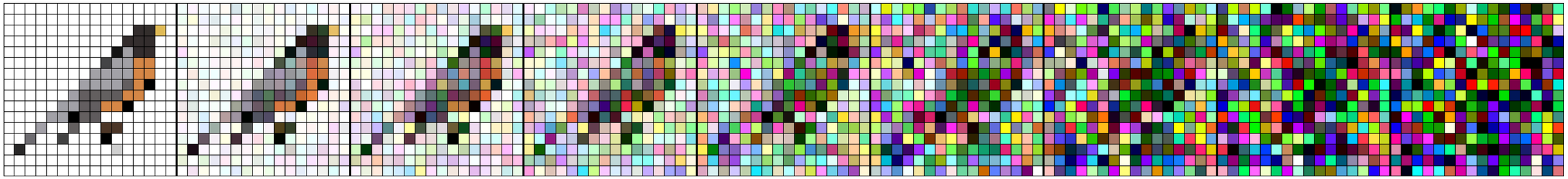
- Do diffusion in latent space for speed and potentially for better perceptual quality (just like the VQ-GAN).



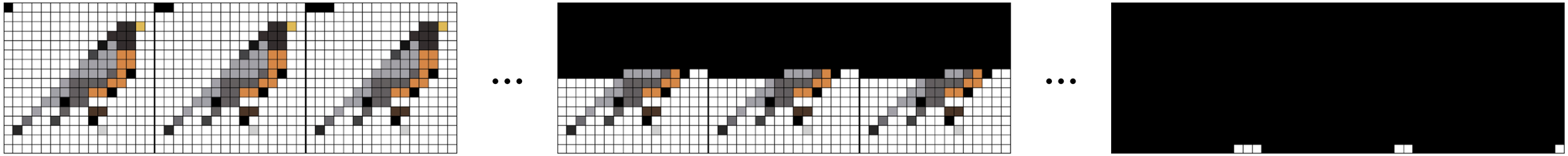
R. Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022
Slide source: Viraj Shah, L. Shen

Autoregressive models vs. diffusion models

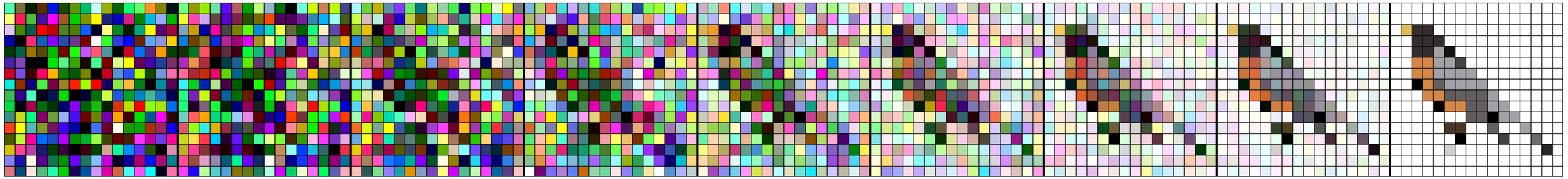
Forward diffusion process



Reverse autoregressive sequence



Diffusion model



Autoregressive model