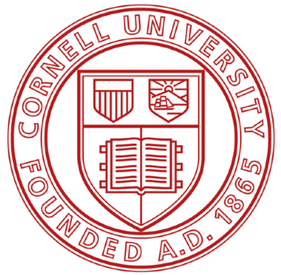
Lecture 10: Diffusion models

CS 5788: Introduction to Generative Models



Announcements

- Project proposal information out tonight

CS5788 Project Proposal

Your Project Title Here

Student Name 1 Student Name 2 Student Name 3
{email1, email2, email3}@cornell.edu

1. Problem Statement

Instructions: Clearly define the problem your project aims to address. Explain why this problem is significant or interesting. What are the specific challenges you are trying to overcome?

Replace this text with your response.

2. Related Work

Instructions: Review at least three relevant prior works related to your topic. Summarize their main ideas, contributions, and limitations. Highlight how your proposed approach differs from or builds upon these existing methods.

Replace this text with your response.

3. Methodology

Instructions: Detail how you intend to solve this problem. Be sure to discuss the following aspects:

- **Dataset:** What data will you use, and how will it be preprocessed?
- **Architecture/Algorithms:** What deep learning models, architectures, or algorithms will you design or implement?
- **Evaluation:** What metrics will you use to evaluate your model's performance?

Replace this text with your response.

4. Deliverables and Expected Results

Instructions: Outline your project milestones and primary deliverables. What concrete artifacts (e.g., code, trained models, a novel dataset) will you produce by the end of the course? Furthermore, what are your expected baseline results or outcomes?

Replace this text with your response.

- **Dataset:** What data will you use, and how will it be preprocessed?
- **Architecture/Algorithms:** What deep learning models, architectures, or algorithms will you design or implement?
- **Evaluation:** What metrics will you use to evaluate your model's performance?

Replace this text with your response.

4. Deliverables and Expected Results

Instructions: Outline your project milestones and primary deliverables. What concrete artifacts (e.g., code, trained models, a novel dataset) will you produce by the end of the course? Furthermore, what are your expected baseline results or outcomes?

Replace this text with your response.

5. Feasibility

Instructions: Discuss the feasibility of your project. Do you have access to the necessary computational resources (e.g., GPUs) to train your proposed models? Is your dataset already available, or will data collection take up significant time? Justify why this project can be completed within the course timeframe.

Replace this text with your response.

6. Workload Distribution

Instructions: If you are working as a team, detail the specific responsibilities and expected contributions of each team member. How will the tasks be divided to ensure equal contribution?

Student 1: Responsibilities...
Student 2: Responsibilities...
Student 3: Responsibilities...

References

Today

- Finish discussion of EBMs
- Introduction to diffusion models
- Next class: more on why diffusion works

Recall: Energy-based model

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 if we directly estimate the score?

How does true distribution change when you change $\mathbf{x}$?

Score function for our learned distribution (implemented as $s_\theta(\mathbf{x})$.

Want to minimize *Fisher divergence*:

$$D_F(p_D(\mathbf{x}) \parallel p_\theta(\mathbf{x})) = \mathbb{E}_{\mathbf{x}\sim p_D}\left[\frac{1}{2} \left\| \nabla_{\mathbf{x}} \log p_D(\mathbf{x}) - \nabla_{\mathbf{x}} \log p_\theta(\mathbf{x}) \right\|^2\right]$$

where p_D is the true distribution. However, we don't know $\nabla_{\mathbf{x}} \log(p_D(\mathbf{x}))$!

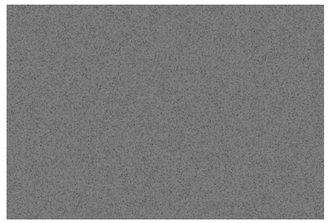
Many approaches to address this (see textbook for details). We'll do a very brief overview of a few approaches, without going into much technical detail.

Recall: the denoising problem

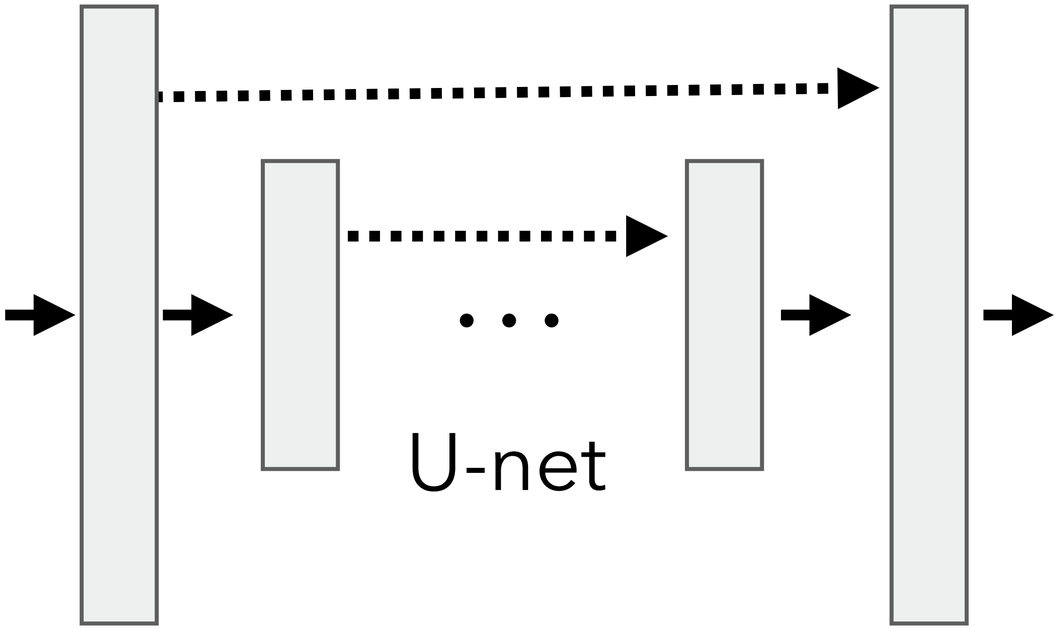
=

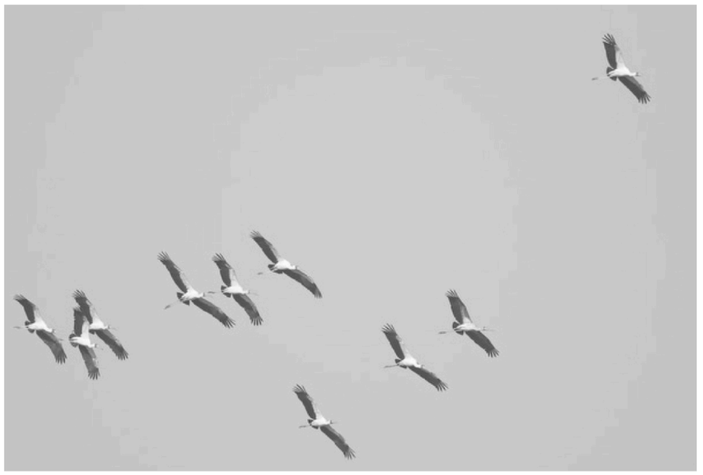
$\mathbf{x}_{\text{clean}}$

+



random noise





$\hat{\mathbf{x}}$

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

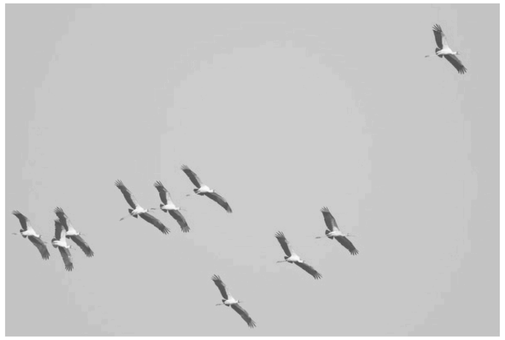
Denoising score matching

It's easier to use the score function for *noisy* inputs.

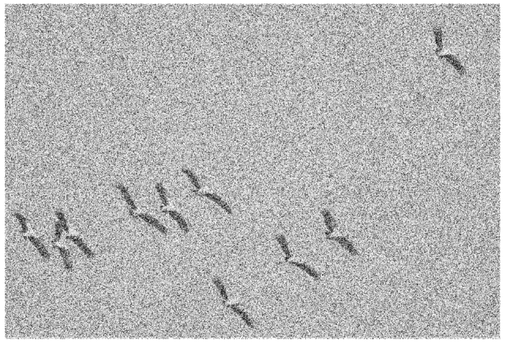
Suppose $\tilde{\mathbf{x}} = \mathbf{x} + \epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ Then $q(\tilde{\mathbf{x}}) = \int_{\mathbf{x}} \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2 I) p_D(\mathbf{x}) d\mathbf{x}$

When conditioned on an example $\mathbf{x}$, the score function has a simple form:

$$\nabla_{\tilde{\mathbf{x}}} \log q(\tilde{\mathbf{x}} \mid \mathbf{x}) = \nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 I) = -\frac{(\tilde{\mathbf{x}} - \mathbf{x})}{\sigma^2}$$



$\mathbf{x} \sim p_D(\mathbf{x})$



$\tilde{\mathbf{x}} \sim q(\tilde{\mathbf{x}} \mid \mathbf{x})$

Denoising score matching

Idea: It's easier to use the score function for *noisy* inputs.

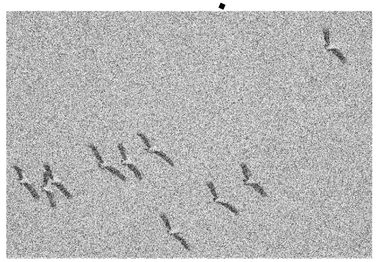
$$\nabla_{\tilde{\mathbf{x}}} \log q(\tilde{\mathbf{x}} \mid \mathbf{x}) = \nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 I) = -\frac{(\tilde{\mathbf{x}} - \mathbf{x})}{\sigma^2}$$

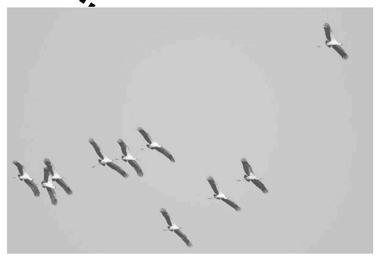
We can take advantage of this to define a simpler loss function (sketch):

$$D_F(q(\tilde{\mathbf{x}}) \parallel p_\theta(\tilde{\mathbf{x}})) = \mathbb{E}_{q(\tilde{\mathbf{x}})}\left[\frac{1}{2} \parallel \nabla_{\tilde{\mathbf{x}}} \log p_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} q(\tilde{\mathbf{x}}) \parallel^2\right]$$

$$= \mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})p_D(\mathbf{x})}\left[\frac{1}{2} \parallel \nabla_{\tilde{\mathbf{x}}} \log p_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} q(\tilde{\mathbf{x}} \mid \mathbf{x}) \parallel^2\right] + \text{constant}$$

add noise to each example





Denoising score matching

Idea: It's easier to use the score function for *noisy* inputs.

$$\nabla_{\tilde{\mathbf{x}}} \log q(\tilde{\mathbf{x}} \mid \mathbf{x}) = \nabla_{\tilde{\mathbf{x}}} \log \mathcal{N}(\tilde{\mathbf{x}} \mid \mathbf{x}, \sigma^2 I) = -\frac{(\tilde{\mathbf{x}} - \mathbf{x})}{\sigma^2}$$

We can take advantage of this to define a simpler loss function (sketch):

$$D_F(q(\tilde{\mathbf{x}}) \parallel p_\theta(\tilde{\mathbf{x}})) = \mathbb{E}_{q(\tilde{\mathbf{x}})}\left[\frac{1}{2} \parallel \nabla_{\tilde{\mathbf{x}}} \log p_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} q(\tilde{\mathbf{x}}) \parallel^2\right]$$

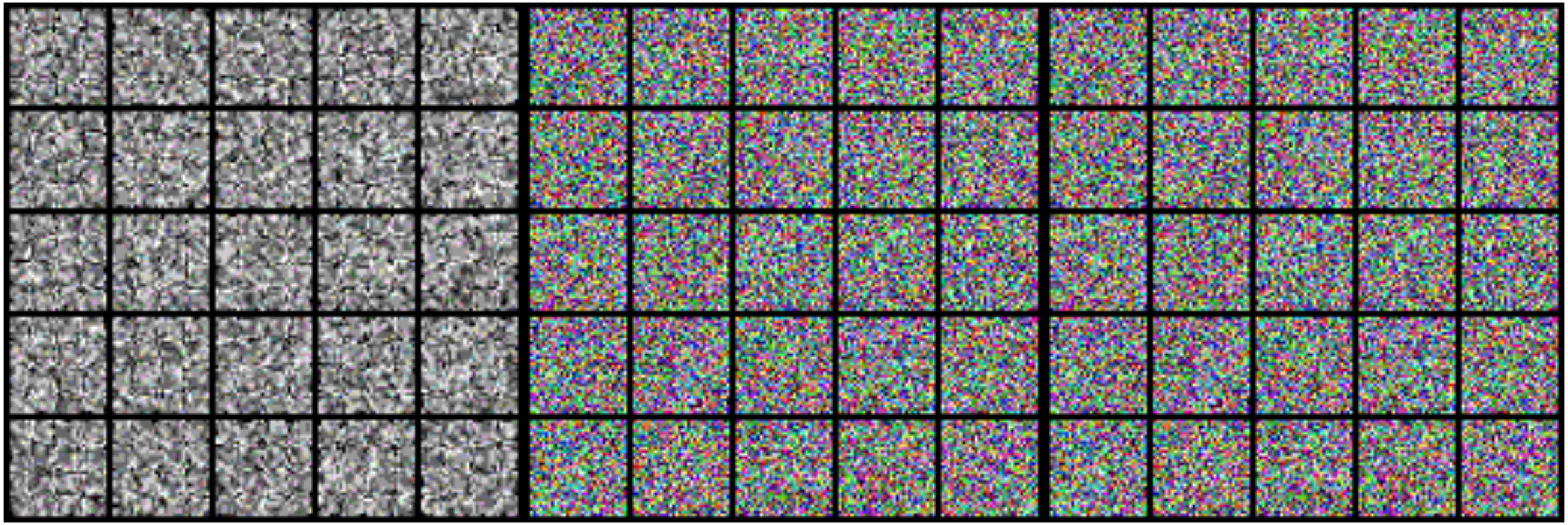
$$= \mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})p_D(\mathbf{x})}\left[\frac{1}{2} \parallel \nabla_{\tilde{\mathbf{x}}} \log p_\theta(\tilde{\mathbf{x}}) - \nabla_{\tilde{\mathbf{x}}} q(\tilde{\mathbf{x}} \mid \mathbf{x}) \parallel^2\right] + \text{constant}$$

add noise to each example

$$= \mathbb{E}_{q(\tilde{\mathbf{x}}|\mathbf{x})p_D(\mathbf{x})}\left[\frac{1}{2} \left\| s_\theta(\tilde{\mathbf{x}}) - \frac{(\mathbf{x} - \tilde{\mathbf{x}})}{\sigma^2} \right\|^2\right] + \text{constant}$$

plug in formula for score of noisy data

Sampling from a trained score matching model



[Song & Ermon, “Generative Modeling by Estimating Gradients of the Data Distribution”, 2019]

Score-based models

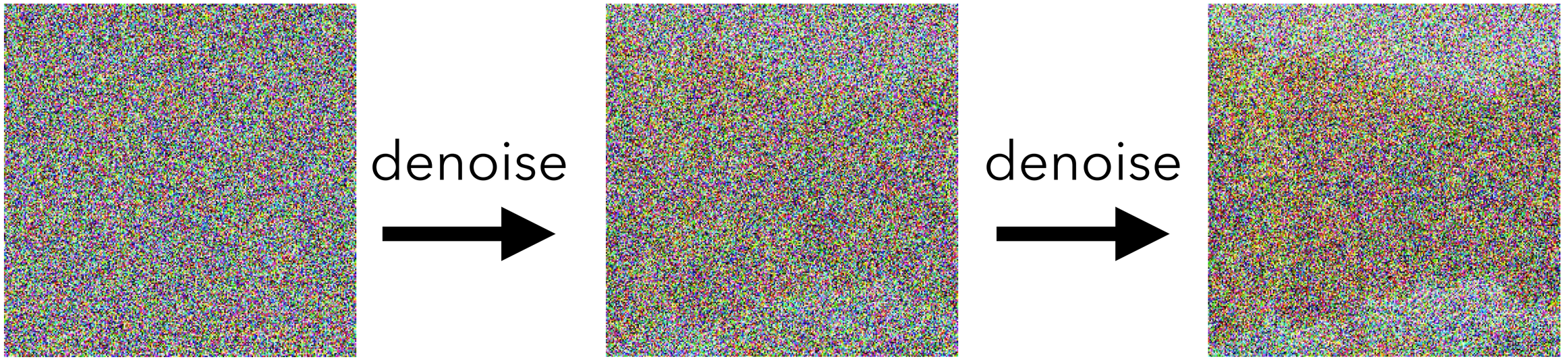
- Model the score function of an EBM $s_\theta(\mathbf{x}) = \nabla_{\mathbf{x}} \log p(\mathbf{x})$
- Designed for easy sampling
- Avoids modeling the normalizing constant $Z(\theta)$
- Doesn't explicitly model $p_\theta(\mathbf{x})$ or $E_\theta(\mathbf{x})$.
- Many ways to train them, including by denoising.

Diffusion models

Today

- Finish discussion of EBMs
- **Introduction to diffusion models**
- Next class: more on why diffusion works

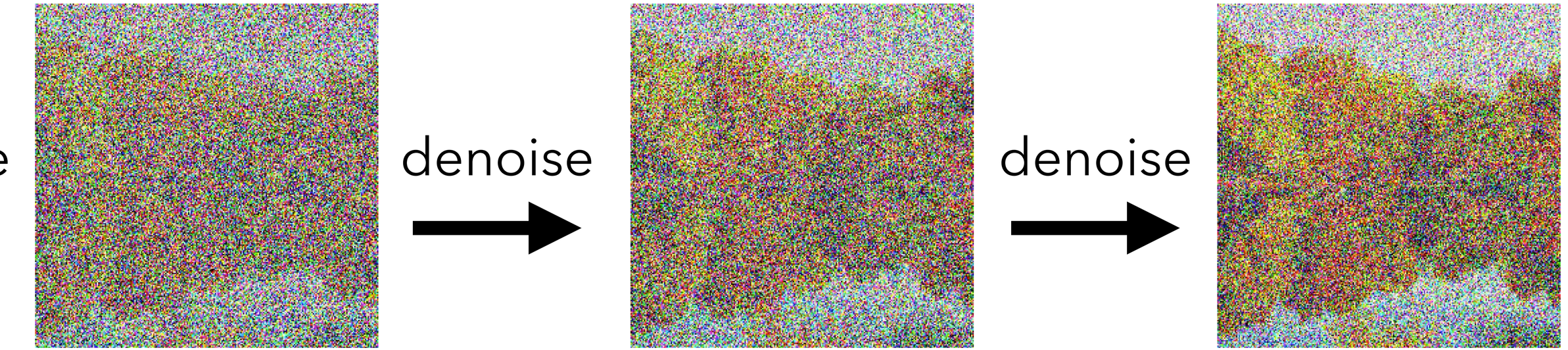
From noise to an image



Random noise

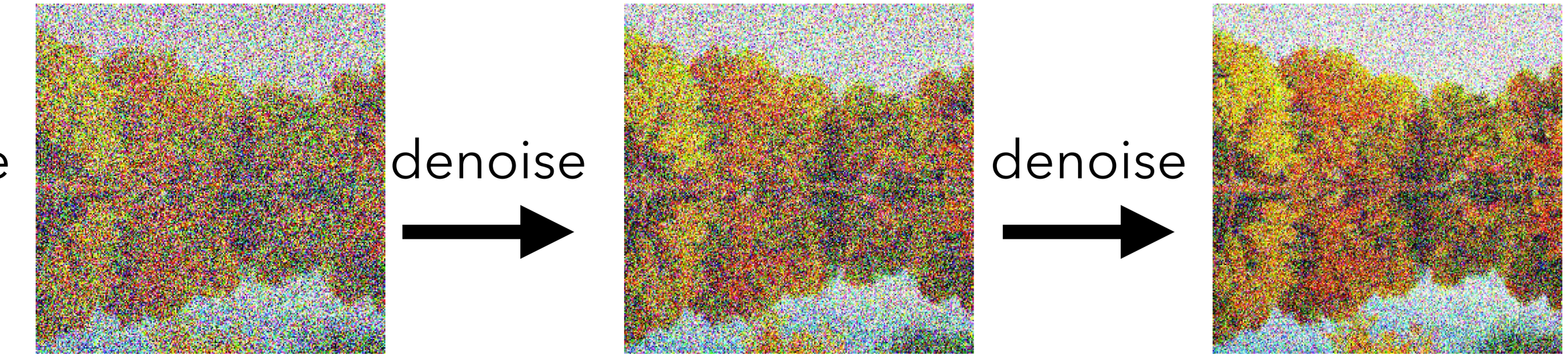
Example source: Aditya Ramesh

From noise to an image



Example source: Aditya Ramesh

From noise to an image



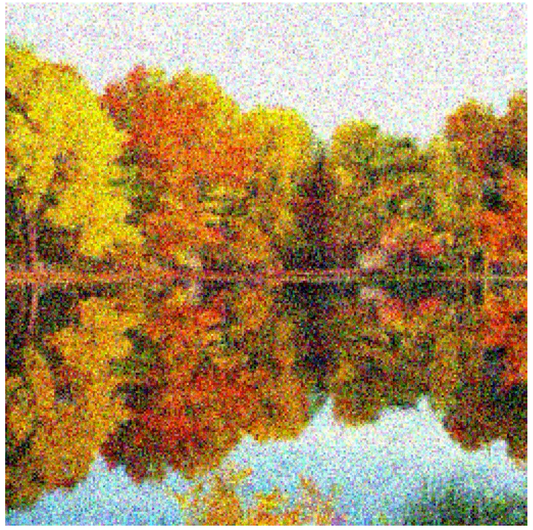
Example source: Aditya Ramesh

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

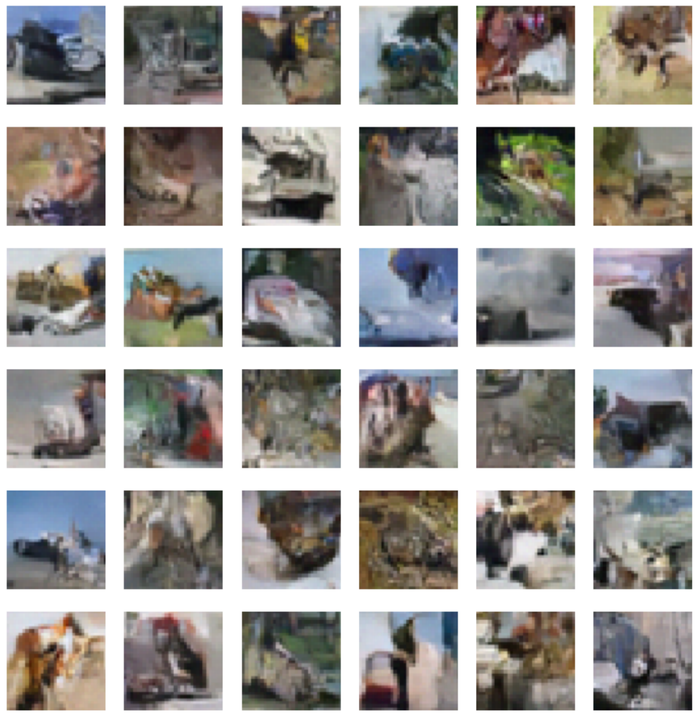
denoise

denoise

Generated image!

Example source: Aditya Ramesh

Diffusion models

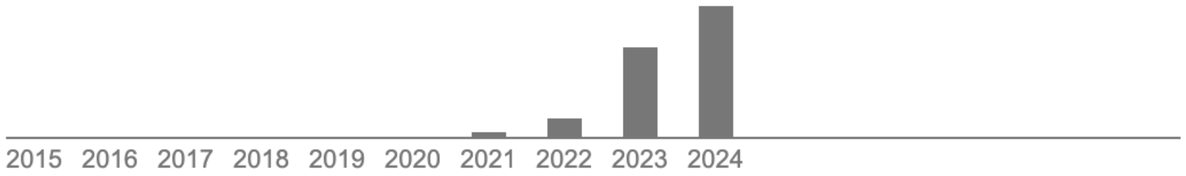


[Sohl-Dickstein et al., 2015]

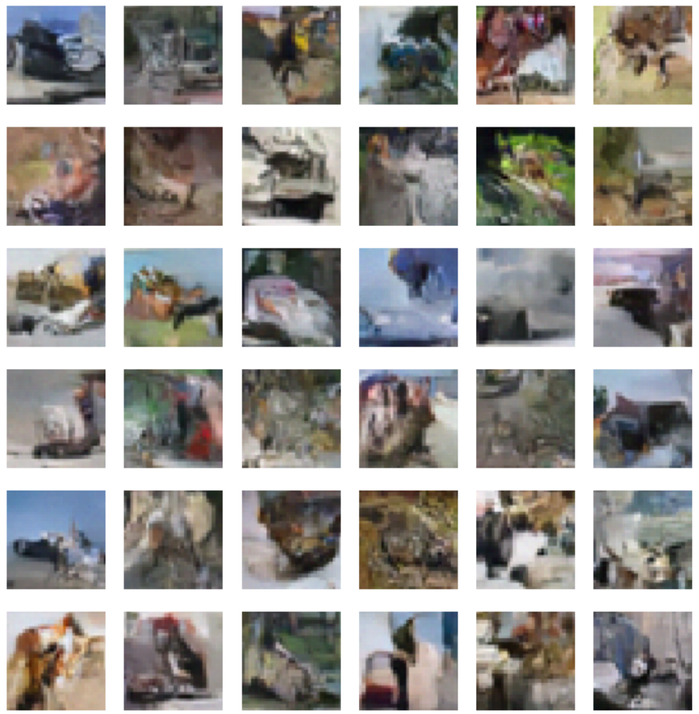
Diffusion models

Deep unsupervised learning using nonequilibrium thermodynamics

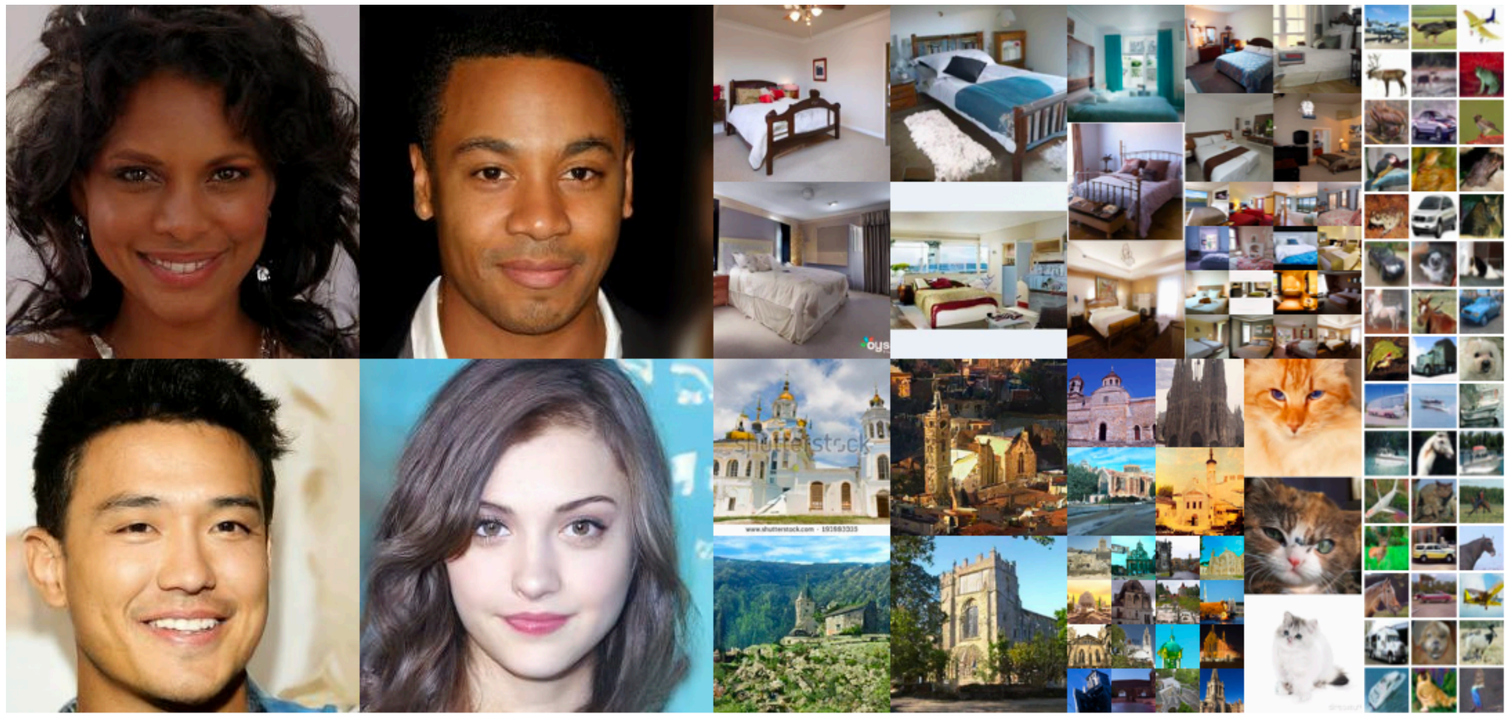
Authors	Jascha Sohl-Dickstein, Eric A Weiss, Niru Maheswaranathan, Surya Ganguli
Publication date	2015/3/12
Journal	International Conference on Machine Learning
Description	A central problem in machine learning involves modeling complex data-sets using highly flexible families of probability distributions 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]



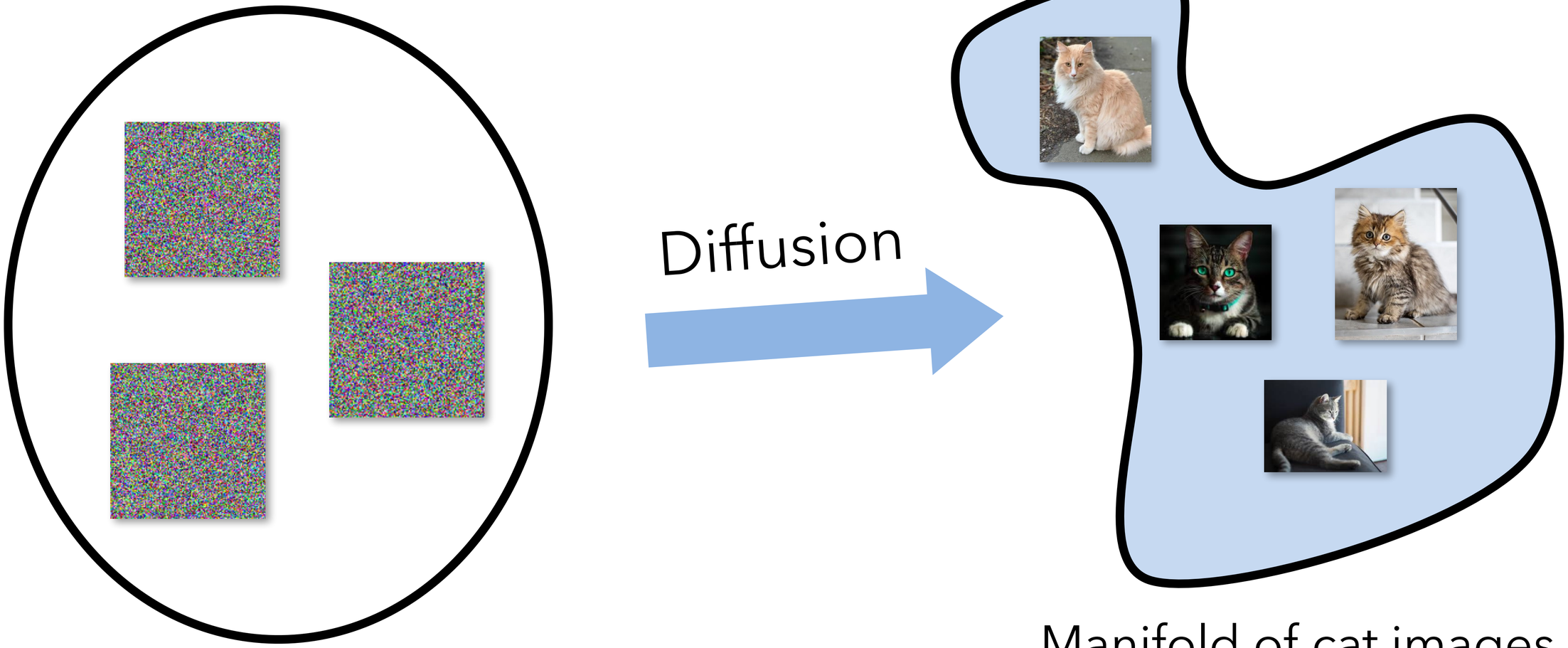
[Ho, et al., “Denoising Diffusion”, 2020]

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

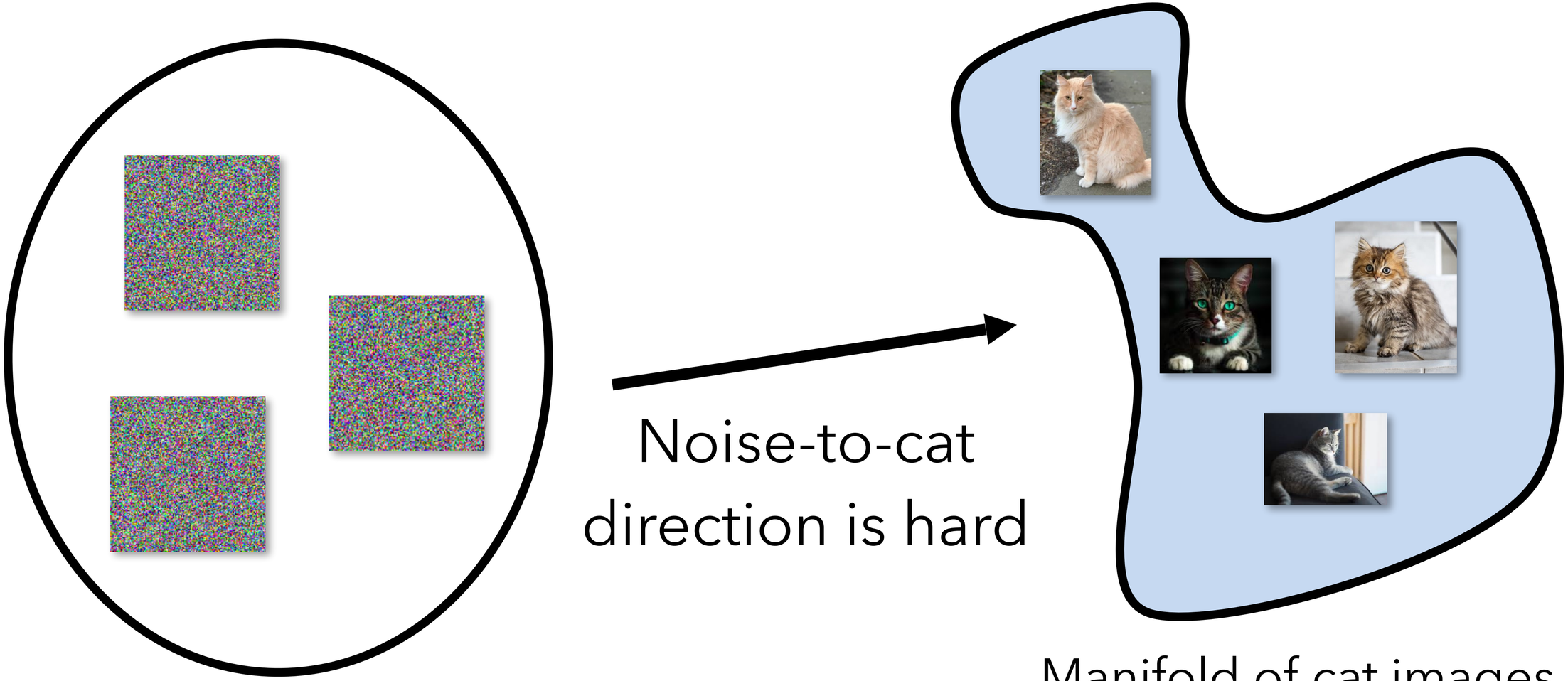
“A dragon fruit wearing karate belt in the snow.”

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

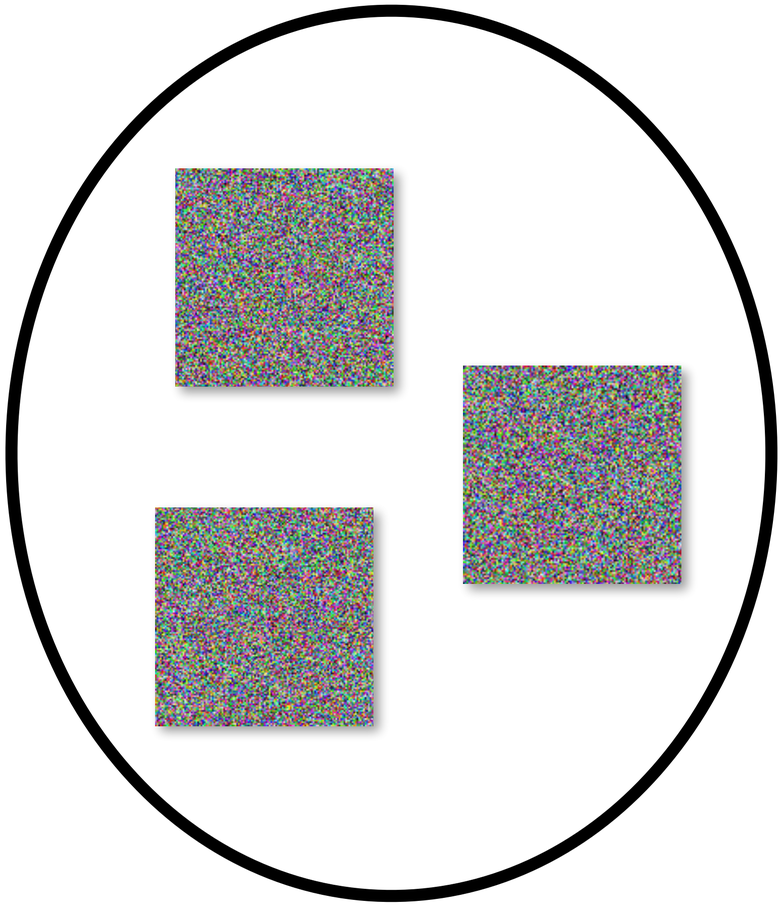

Today: Google Veo 3, 2025



Slide source: N. Snavely & S. Seitz

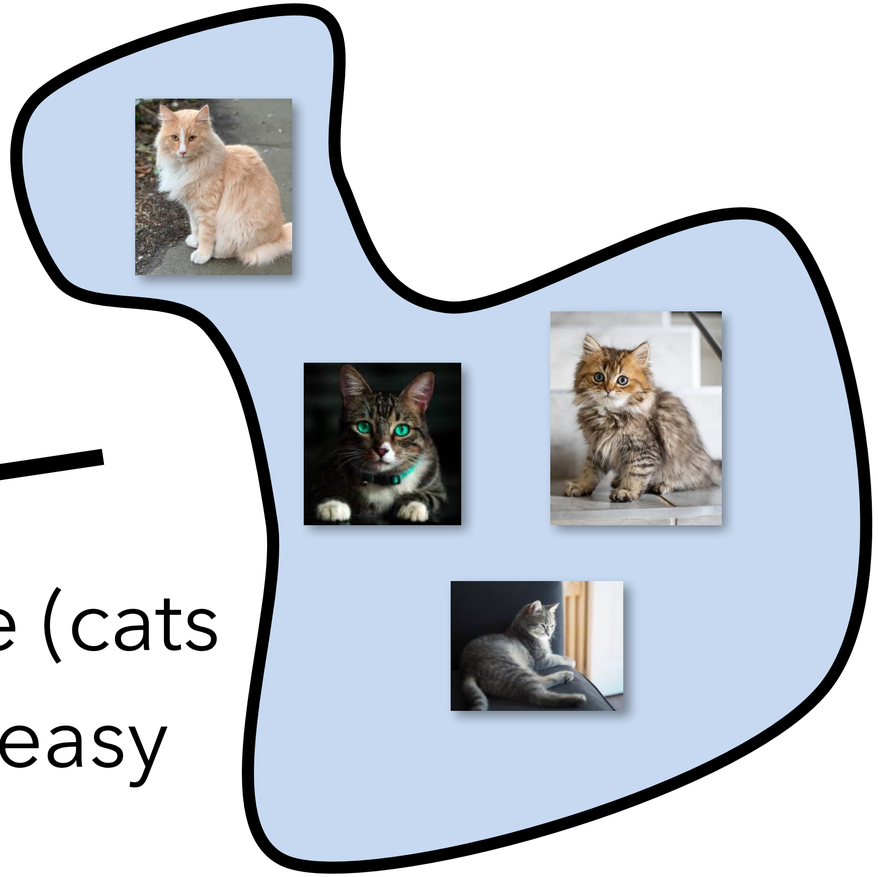


Slide source: N. Snavely & S. Seitz



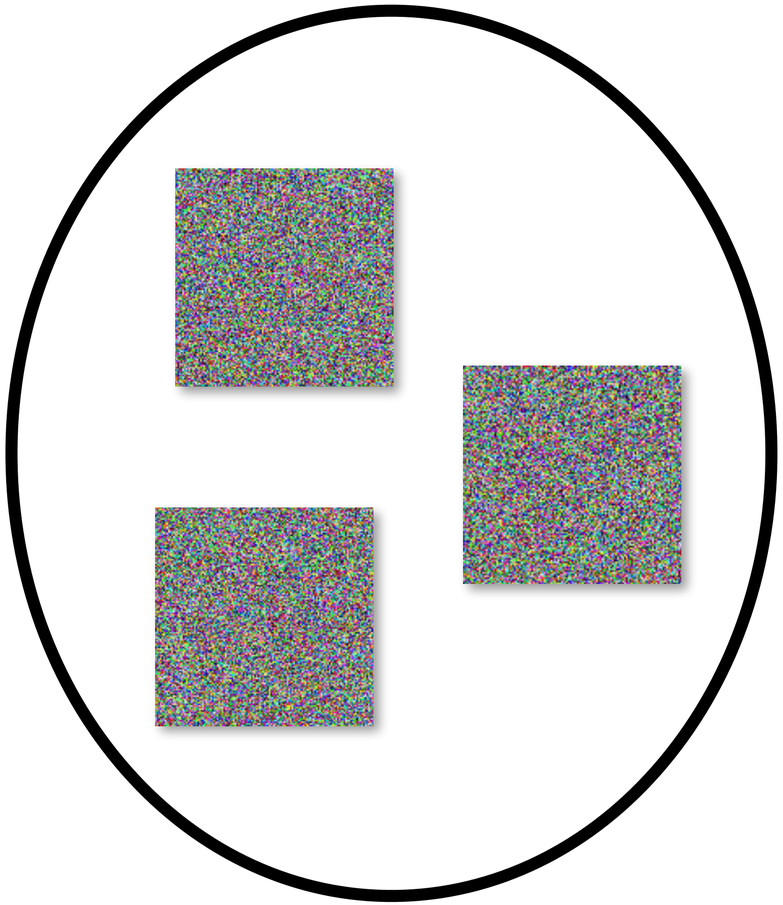
Random images

The opposite (cats to noise) is easy

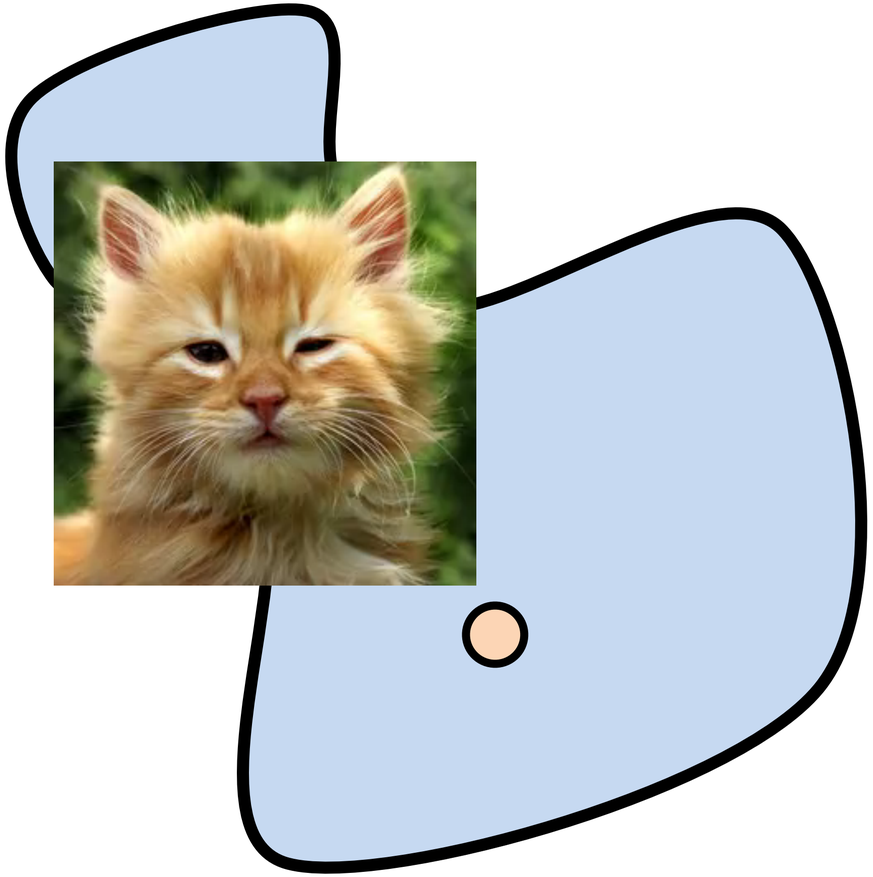


Manifold of cat images

Slide source: N. Snavely & S. Seitz

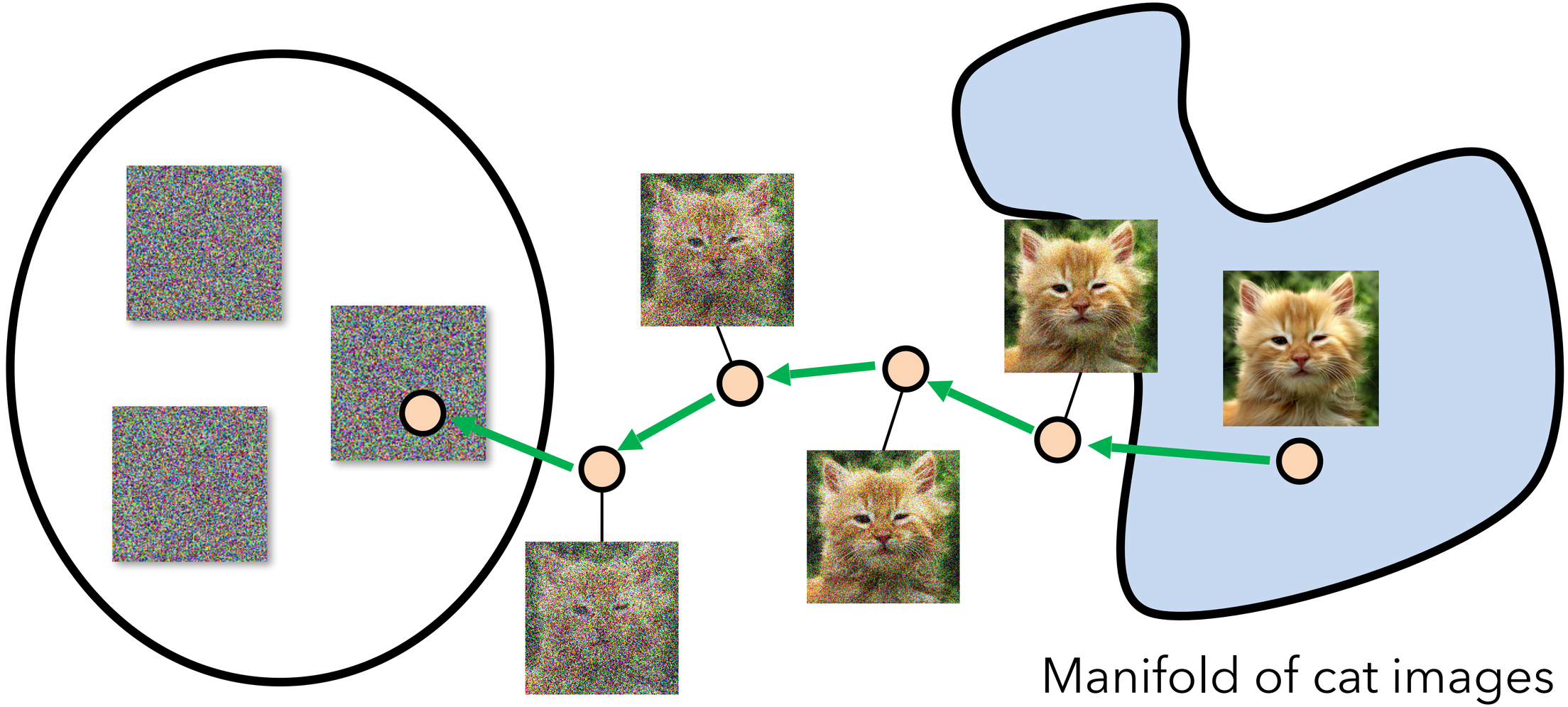


Random images

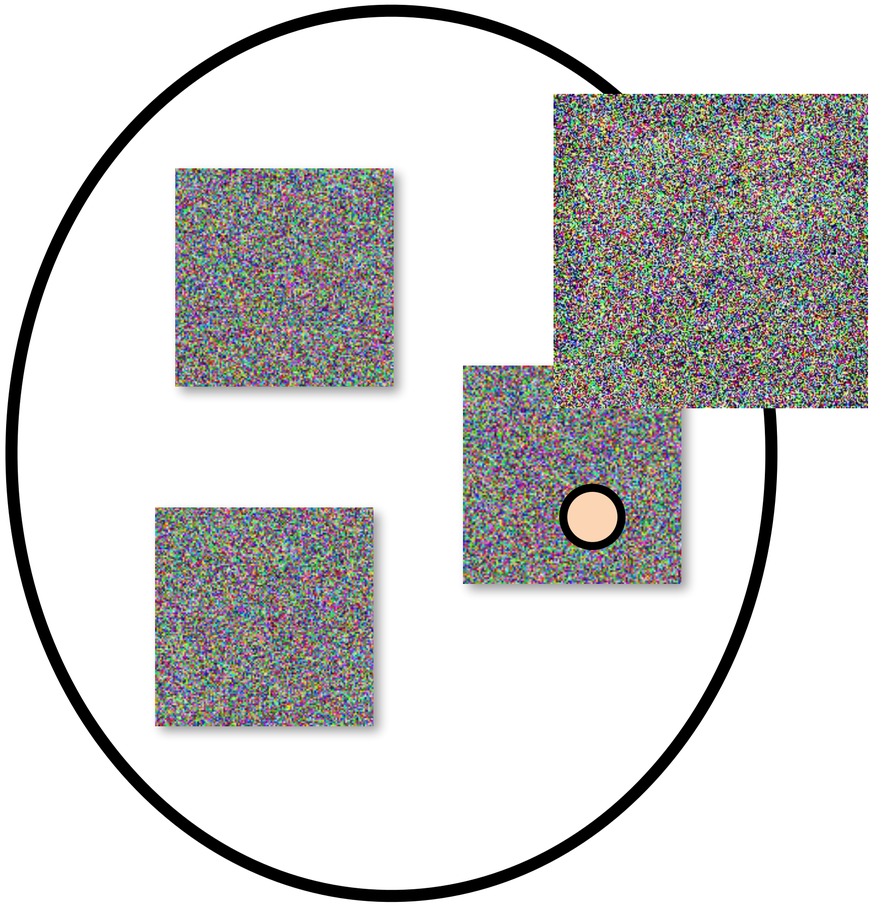


Manifold of cat images

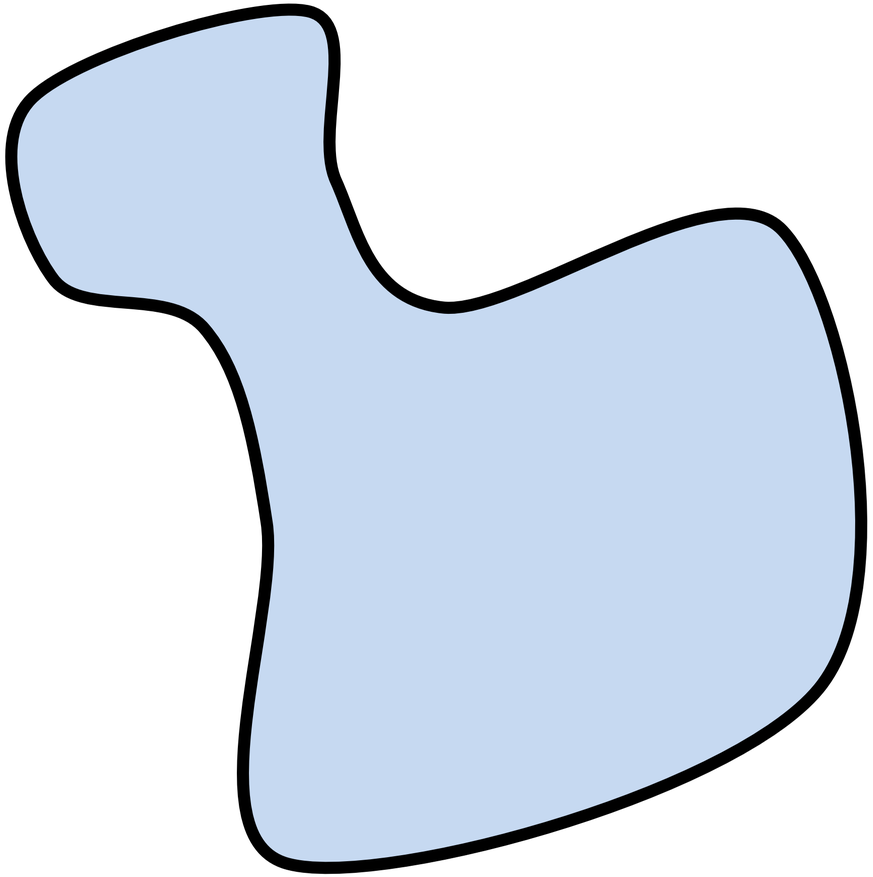
Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz

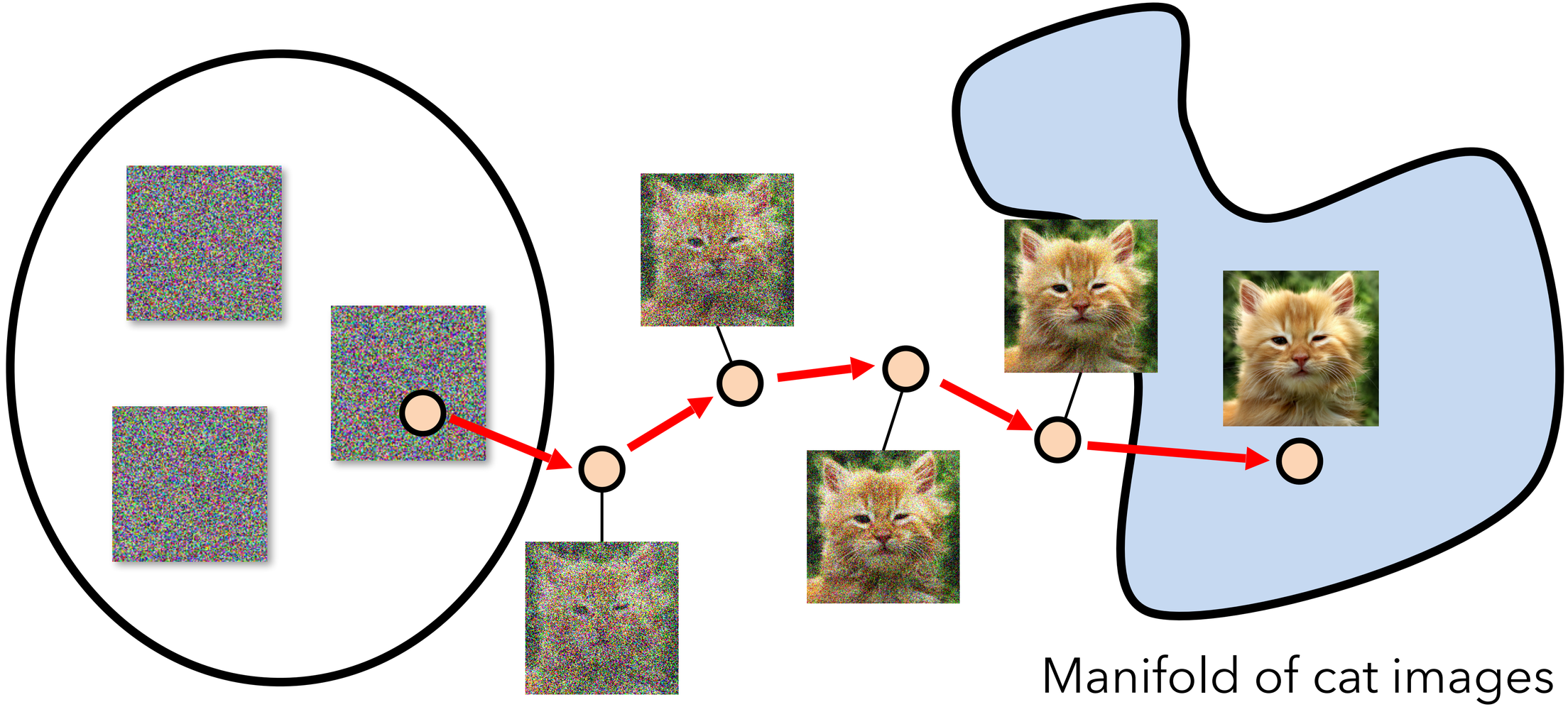


Random images

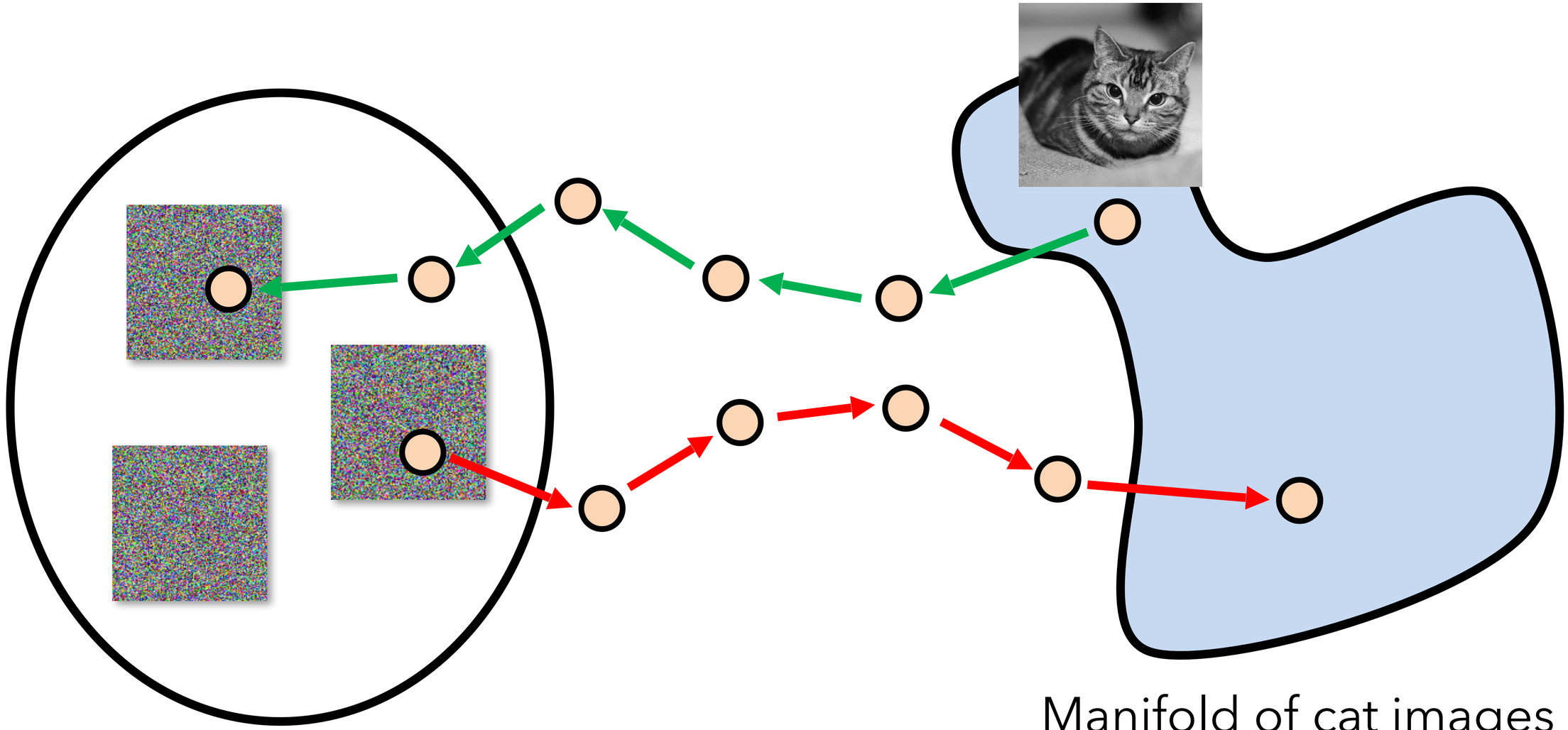


Manifold of cat images

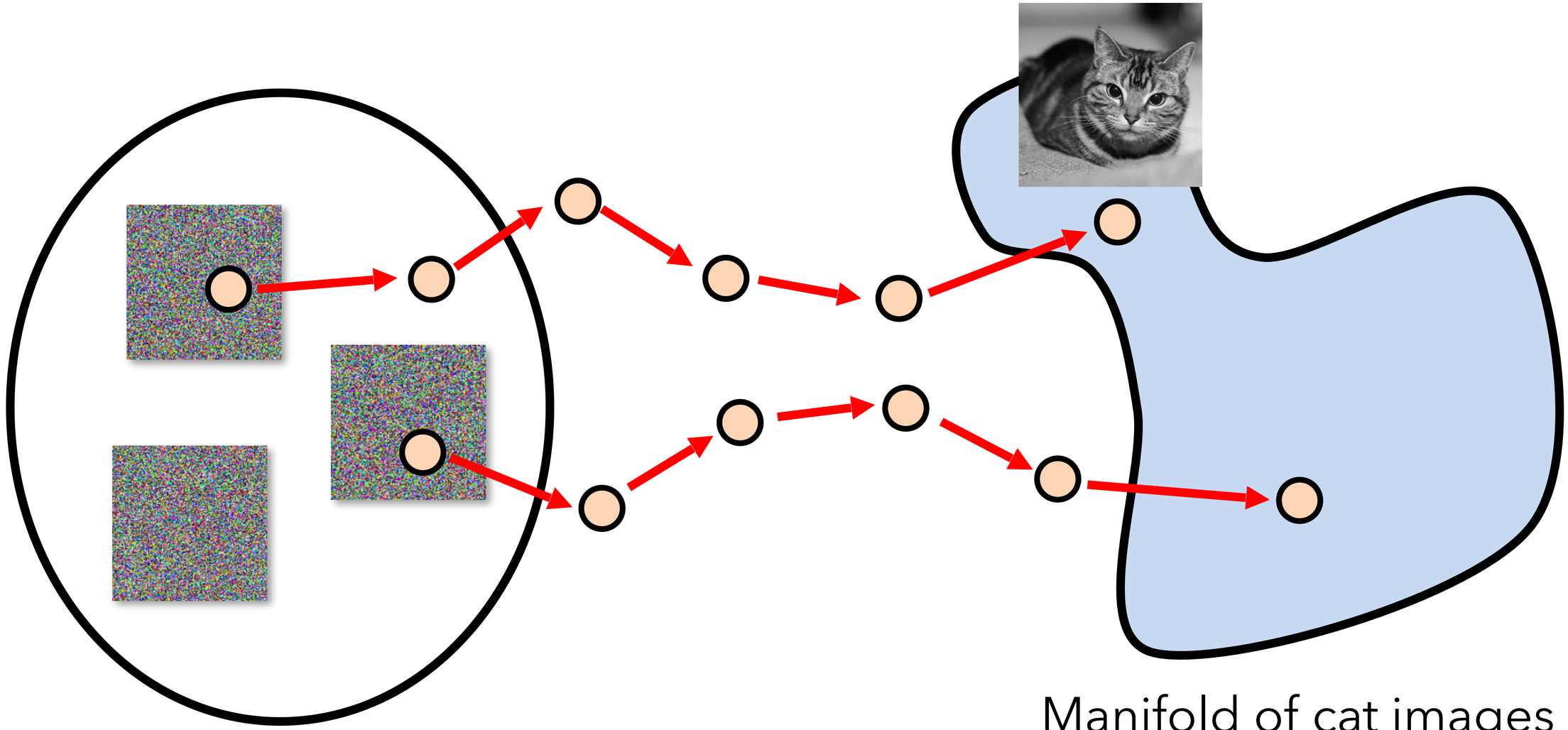
Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz



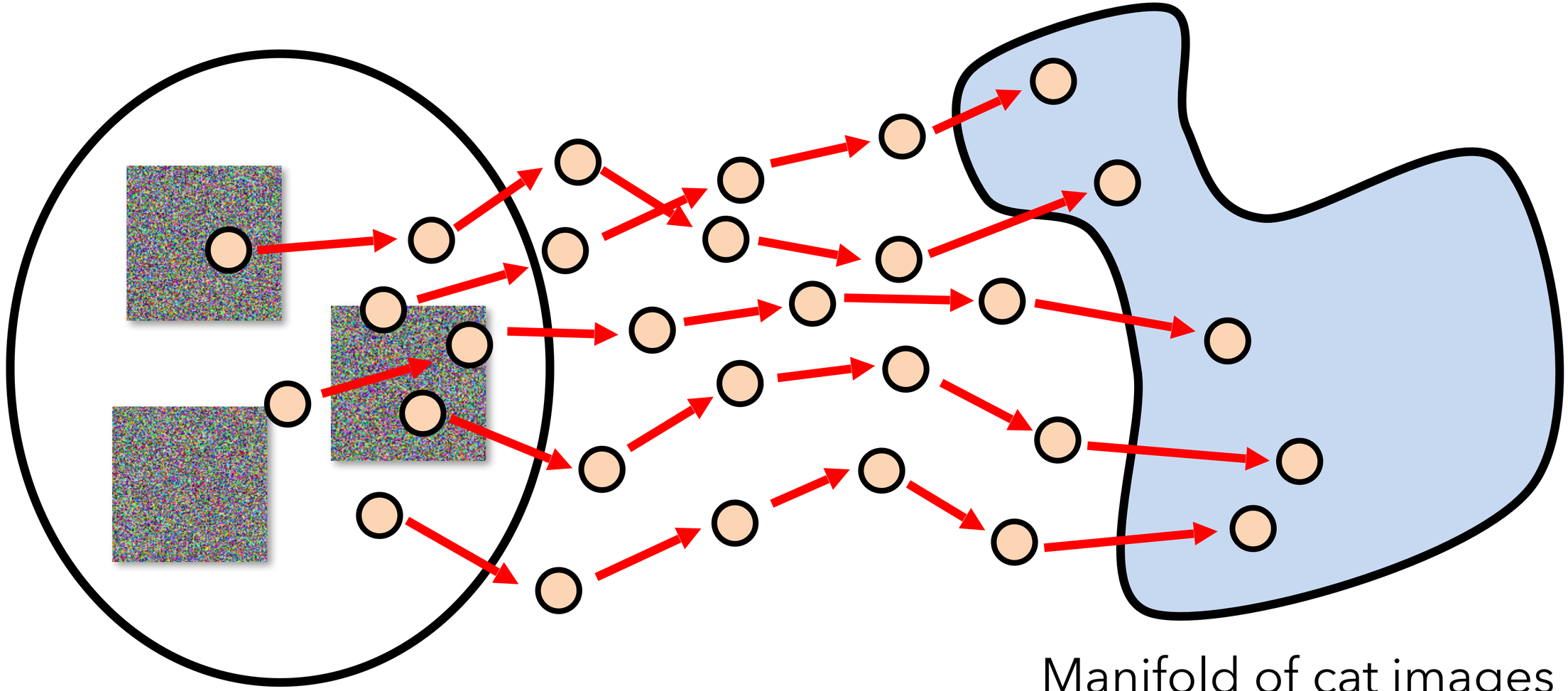
Slide source: N. Snavely & S. Seitz



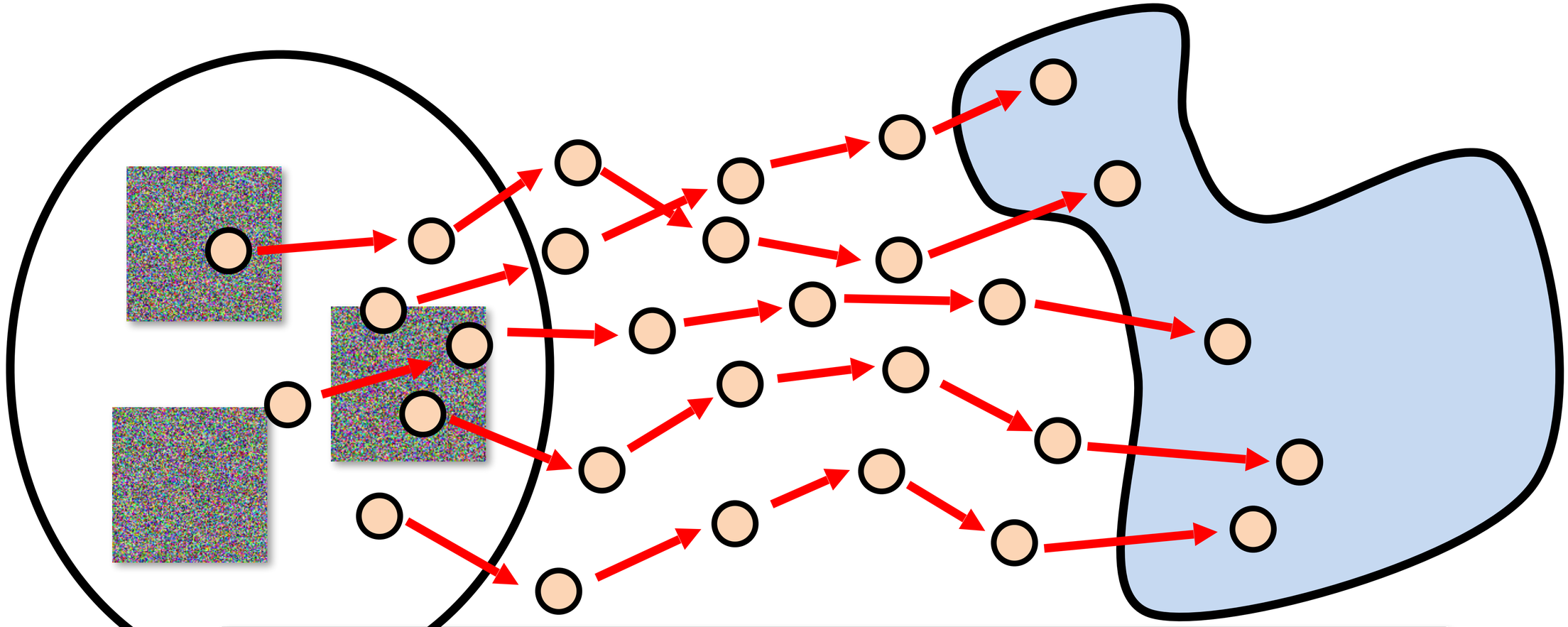
Random images

Manifold of cat images

Slide source: N. Snavely & S. Seitz



Slide source: N. Snavely & S. Seitz



ages

Rand

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

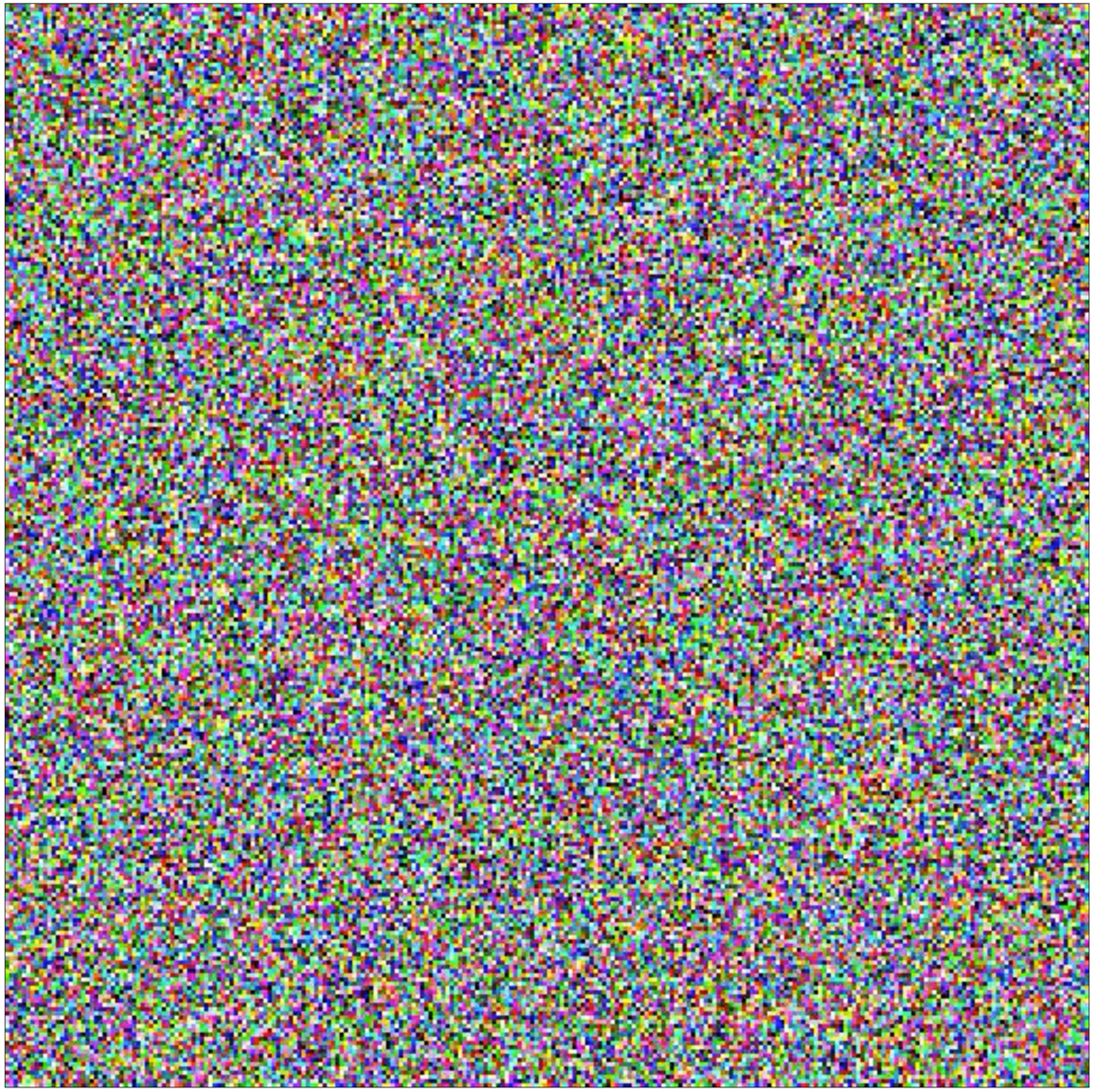
Denoising diffusion neural network

Diffusion neural network

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

Slide source: N. Snavely & S. Seitz

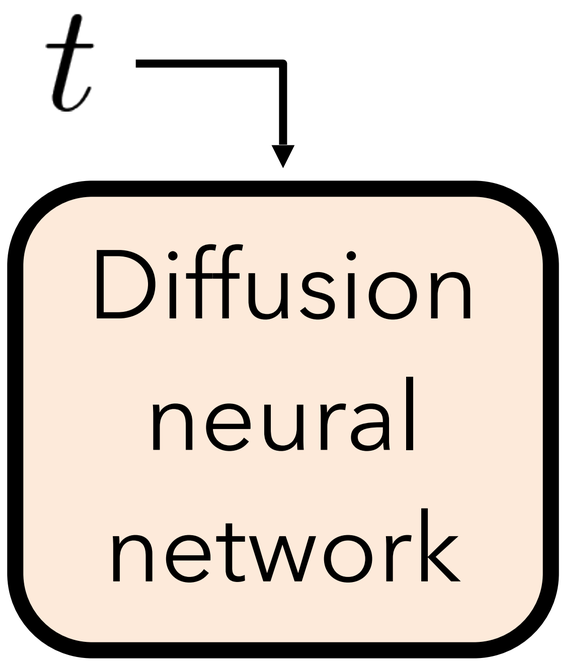
Running a diffusion model for multiple steps



Example source: Aditya Ramesh

Denoising diffusion neural network

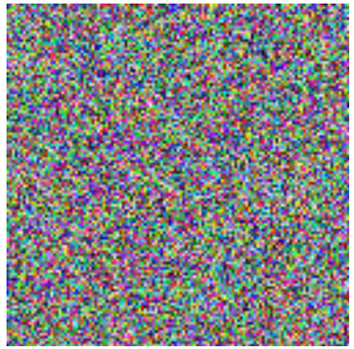
Timestep *t* (from say 0 to 999) is an additional input to the network (positionally encoded)



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

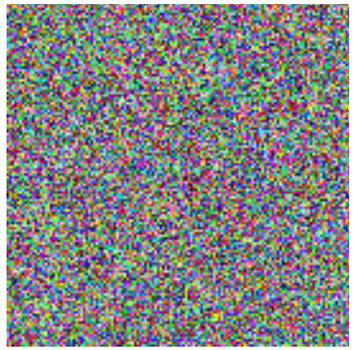
Slide source: N. Snavely & S. Seitz

Sampling many outputs



Noise image 1

Output image 1

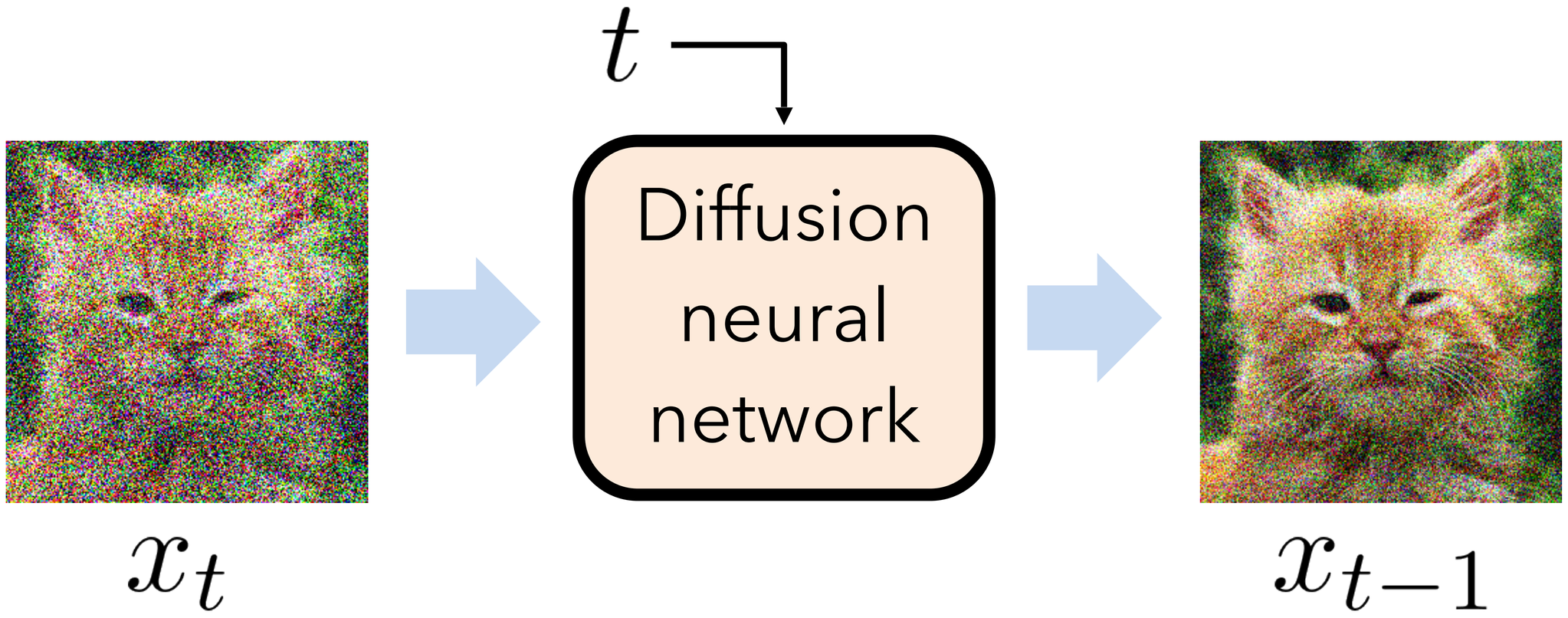


Noise image 2

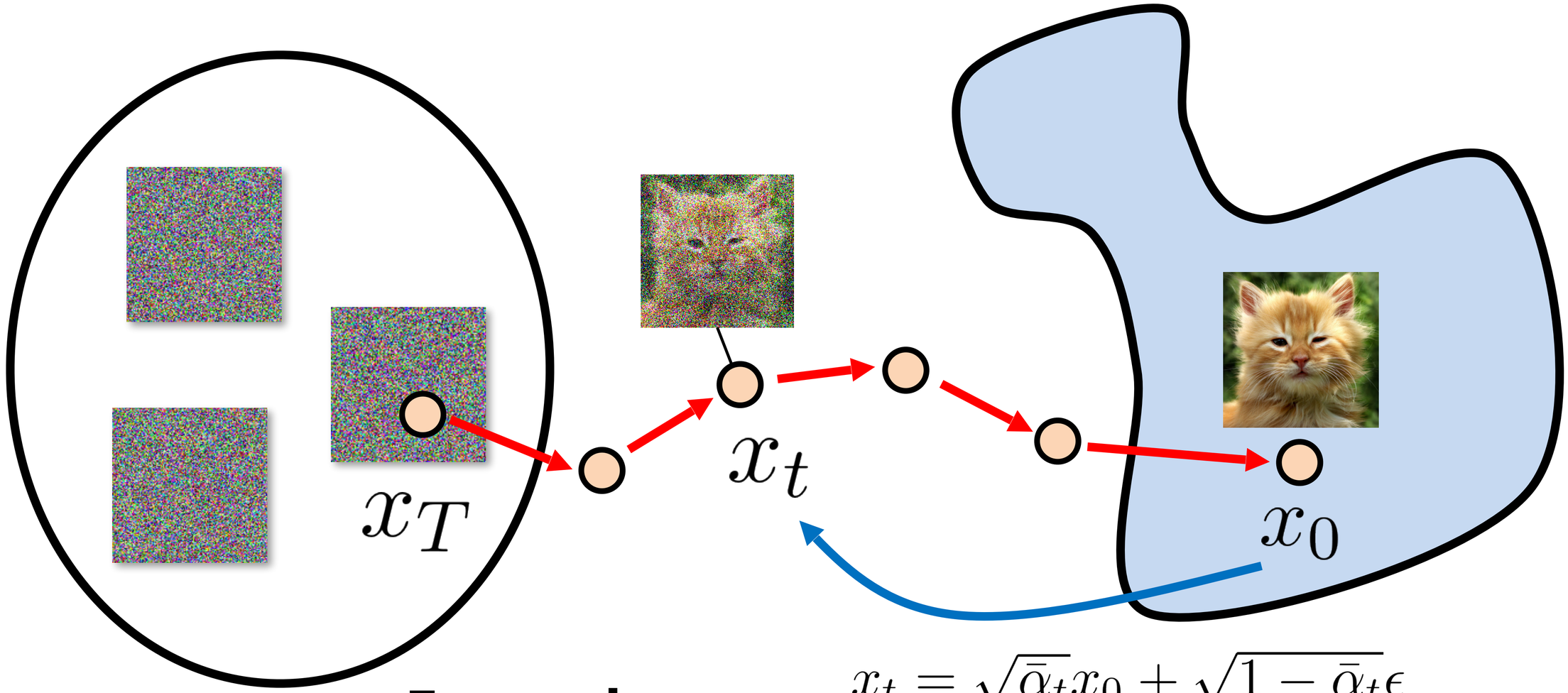
Output image 2

Slide source: N. Snavely

Let’s make this idea more concrete



Adapted from: N. Snavely



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

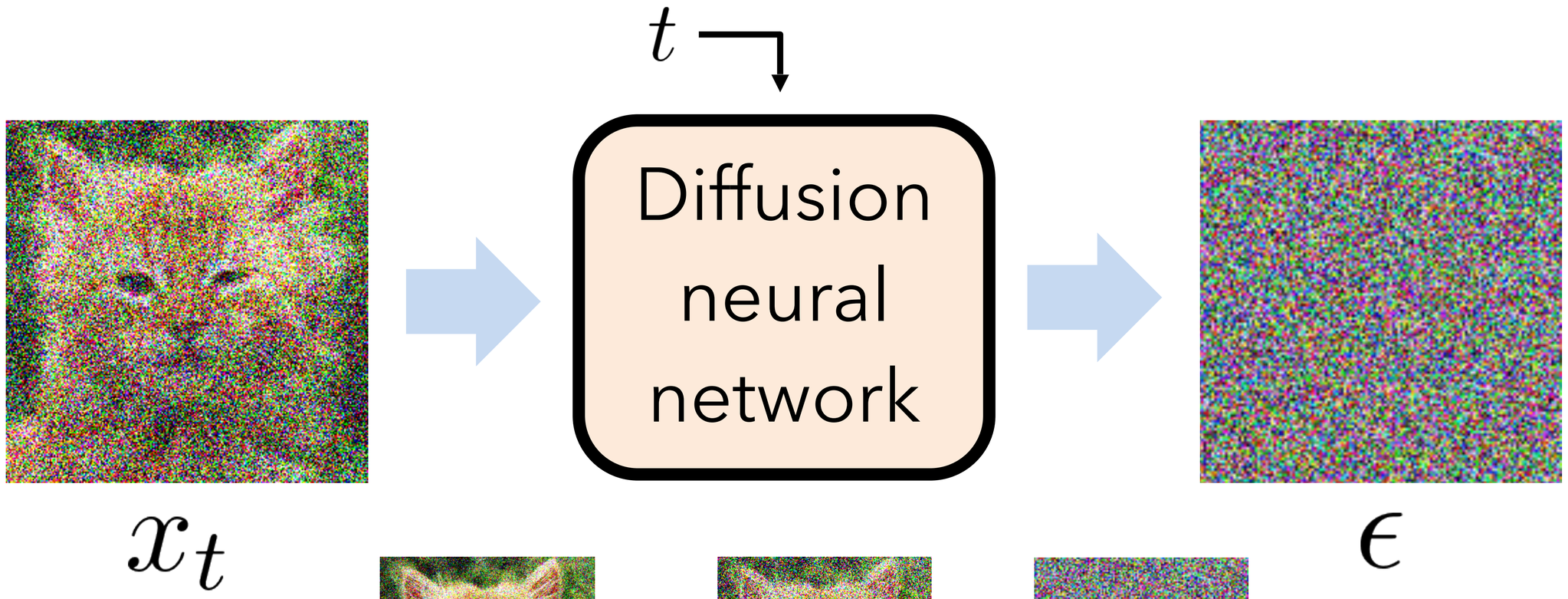
where $\epsilon \sim \mathcal{N}(0, 1)$

(Gaussian noise image)

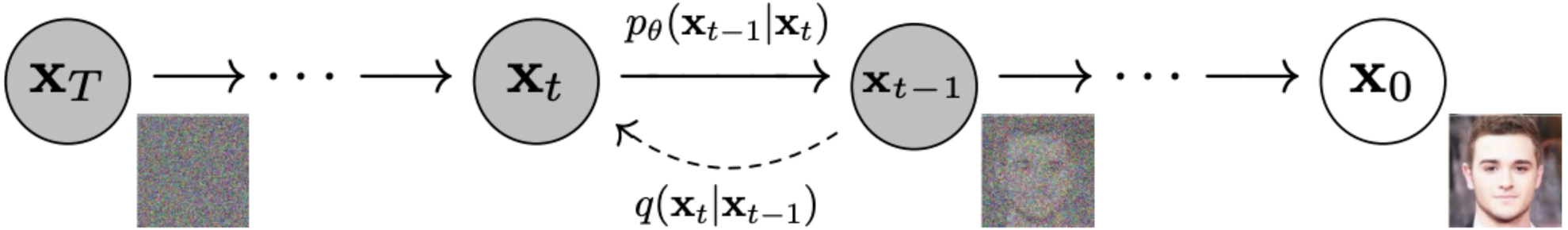
$\bar{\alpha}_t$: noise coefficients per time step (magic numbers defined by the designers)

Adapted from N. Snavely

In practice, we often predict the noise



Training a diffusion model



Algorithm 1 Training

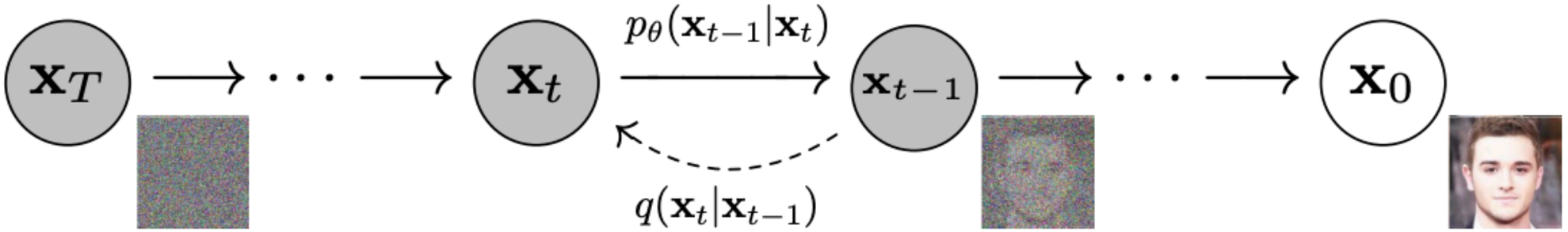
1: **repeat**
2: $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3: $t \sim \text{Uniform}(\{1, \dots, T\})$
4: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
5: Take gradient descent step on

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

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

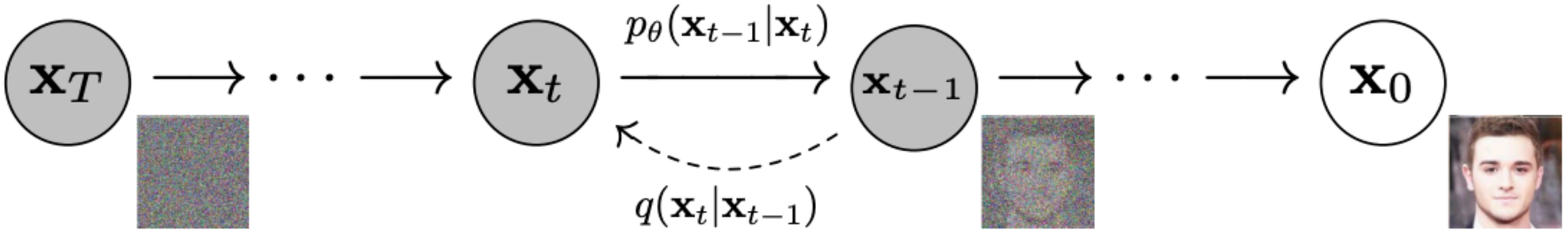
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

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

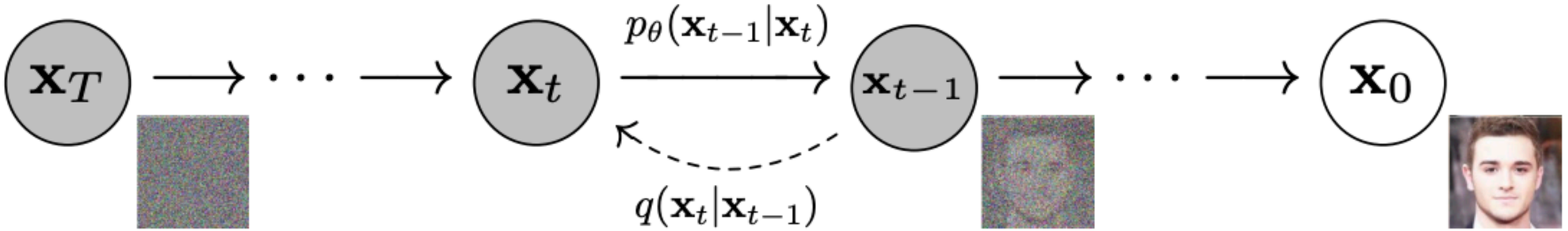
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

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

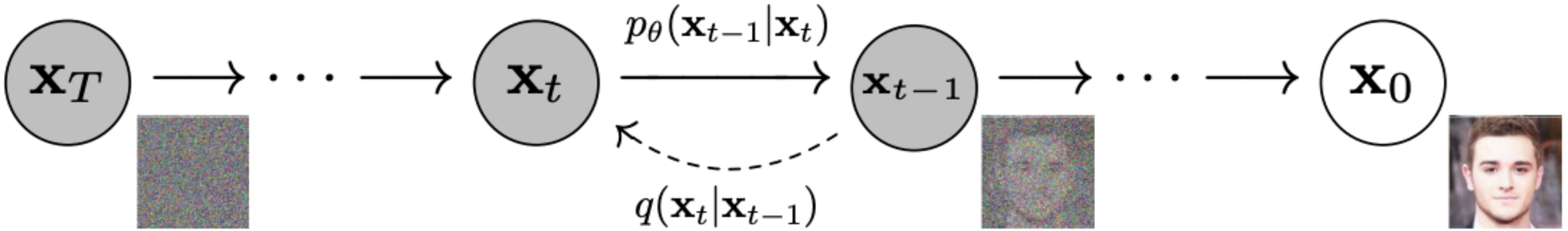
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t) \right\|^2$$

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

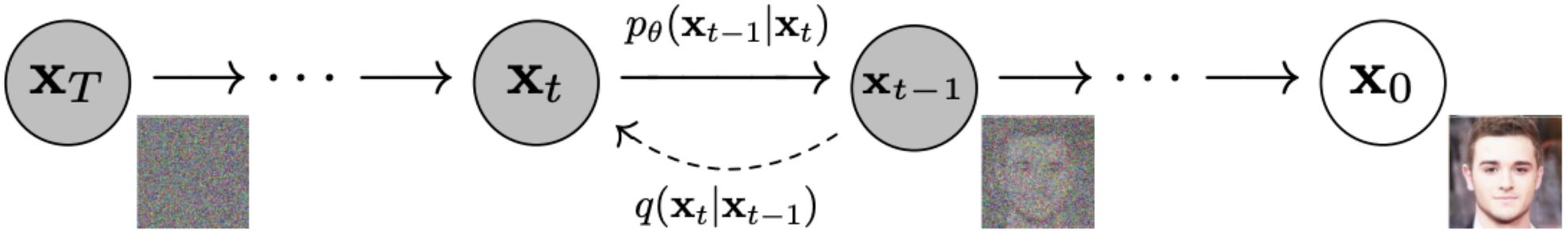
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta\left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\boldsymbol{\epsilon}, t\right)\right\|^2$$

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



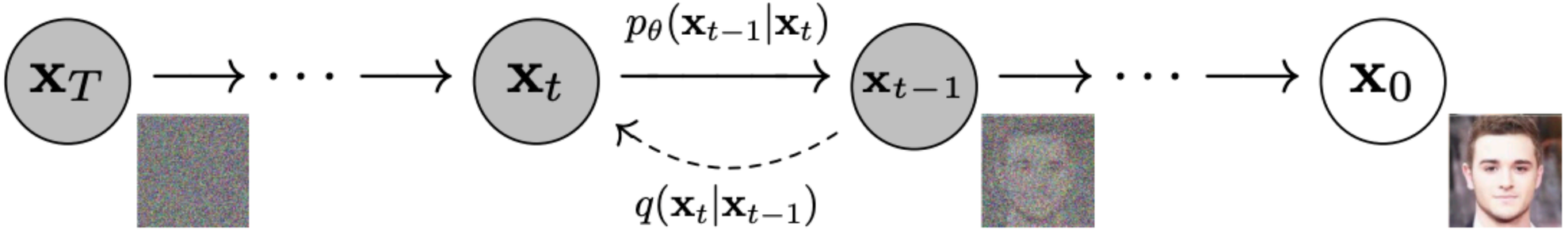
Algorithm 1 Training

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



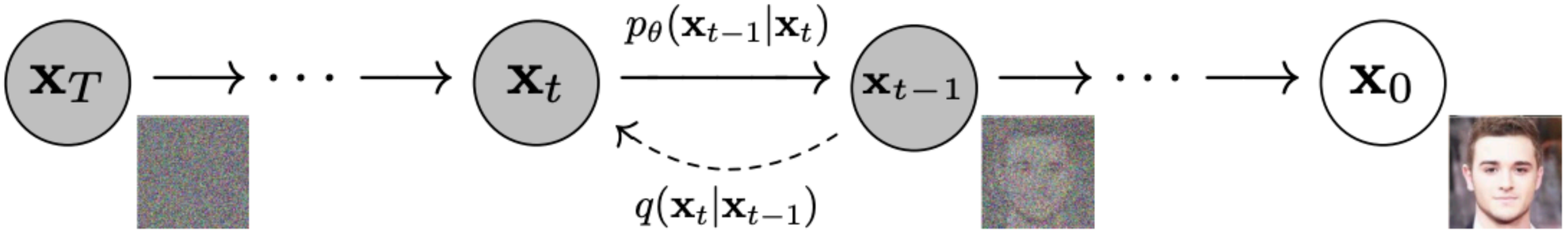
Algorithm 2 Sampling

1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **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



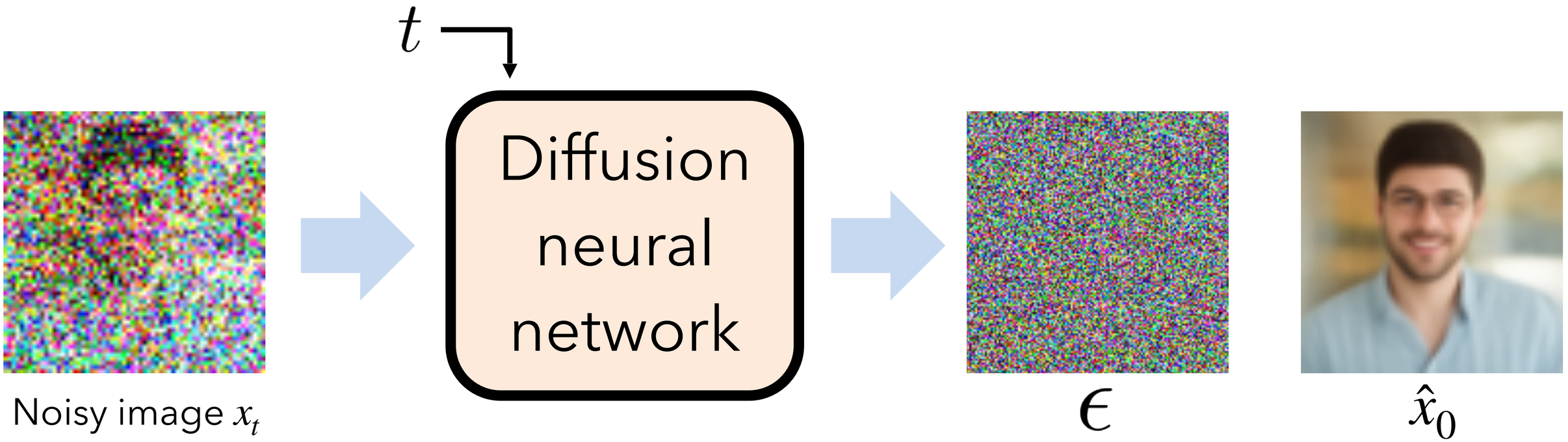
Algorithm 2 Sampling

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)

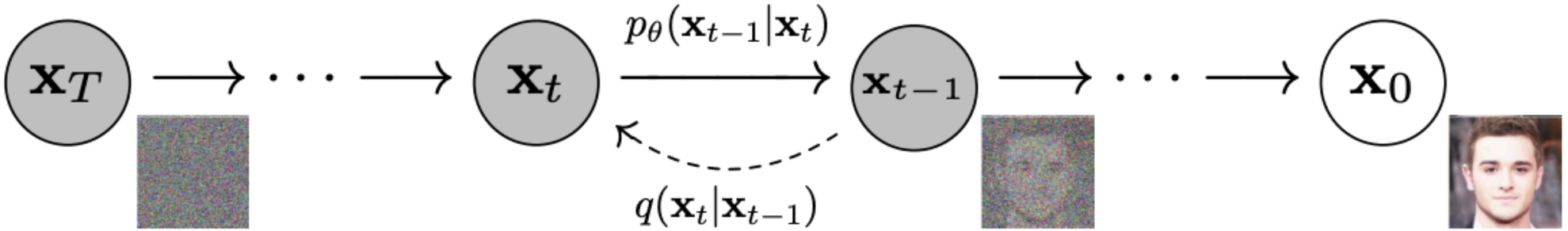
Figure from [J. Ho et al, "DDPM", 2020]
Adapted from Liyue Shen

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** $\mathbf{x}_0$

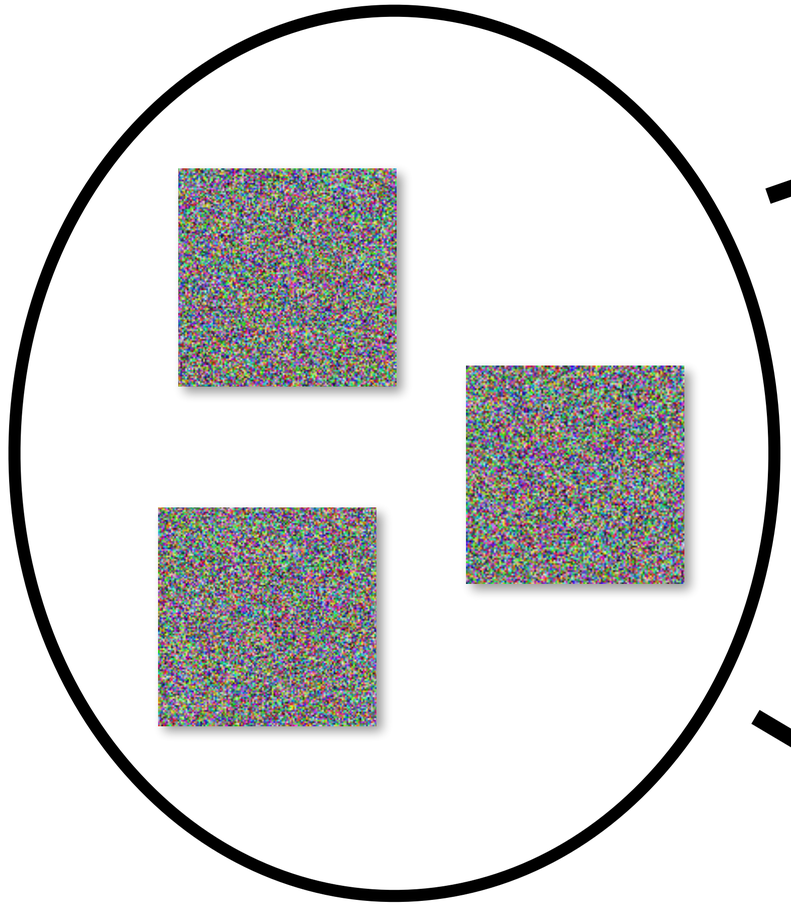
Can write update as:

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

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

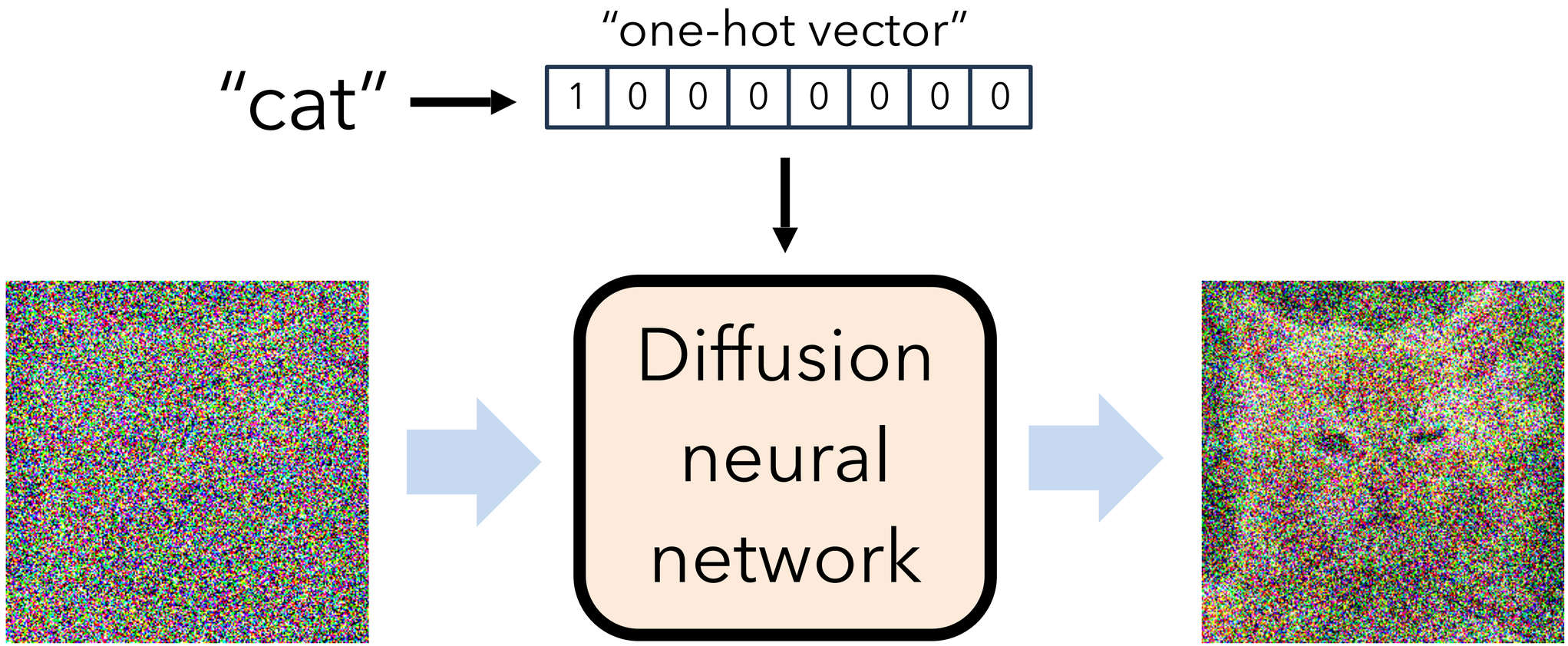
cat images

llama images

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

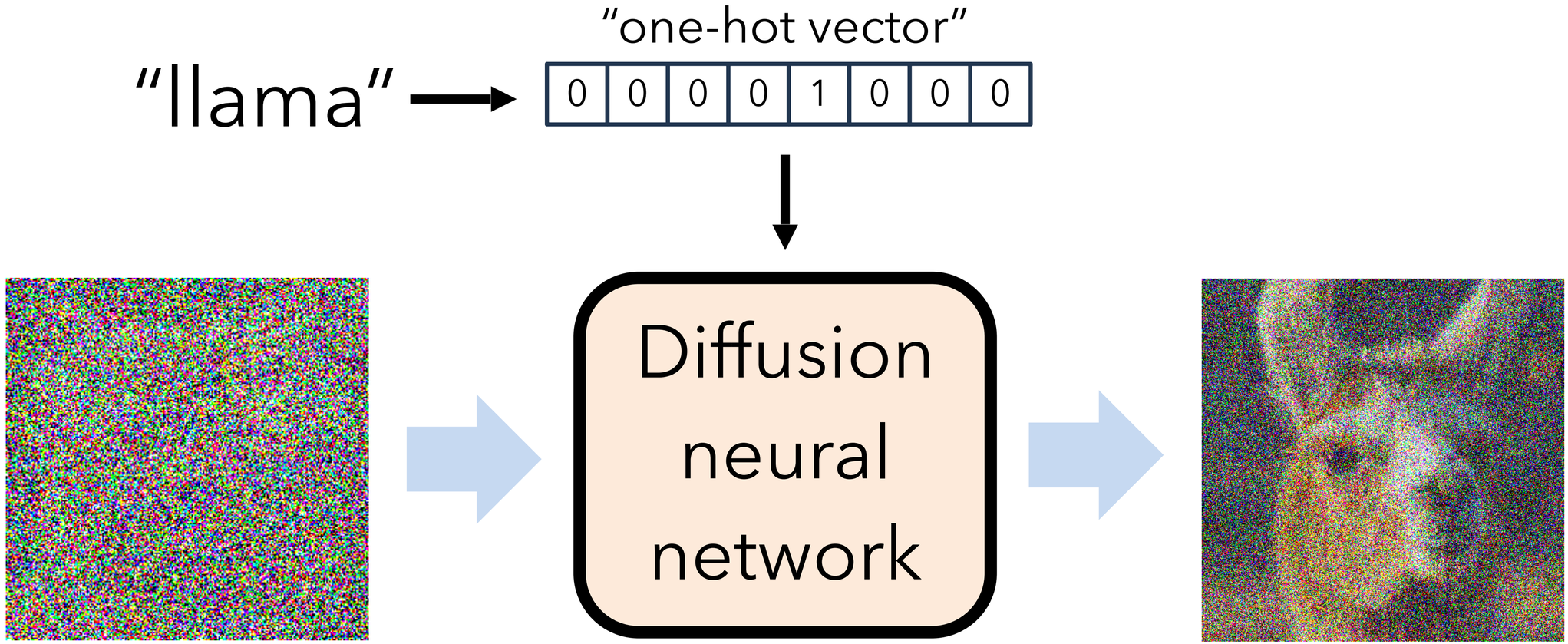
Slide concept: Steve Seitz

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

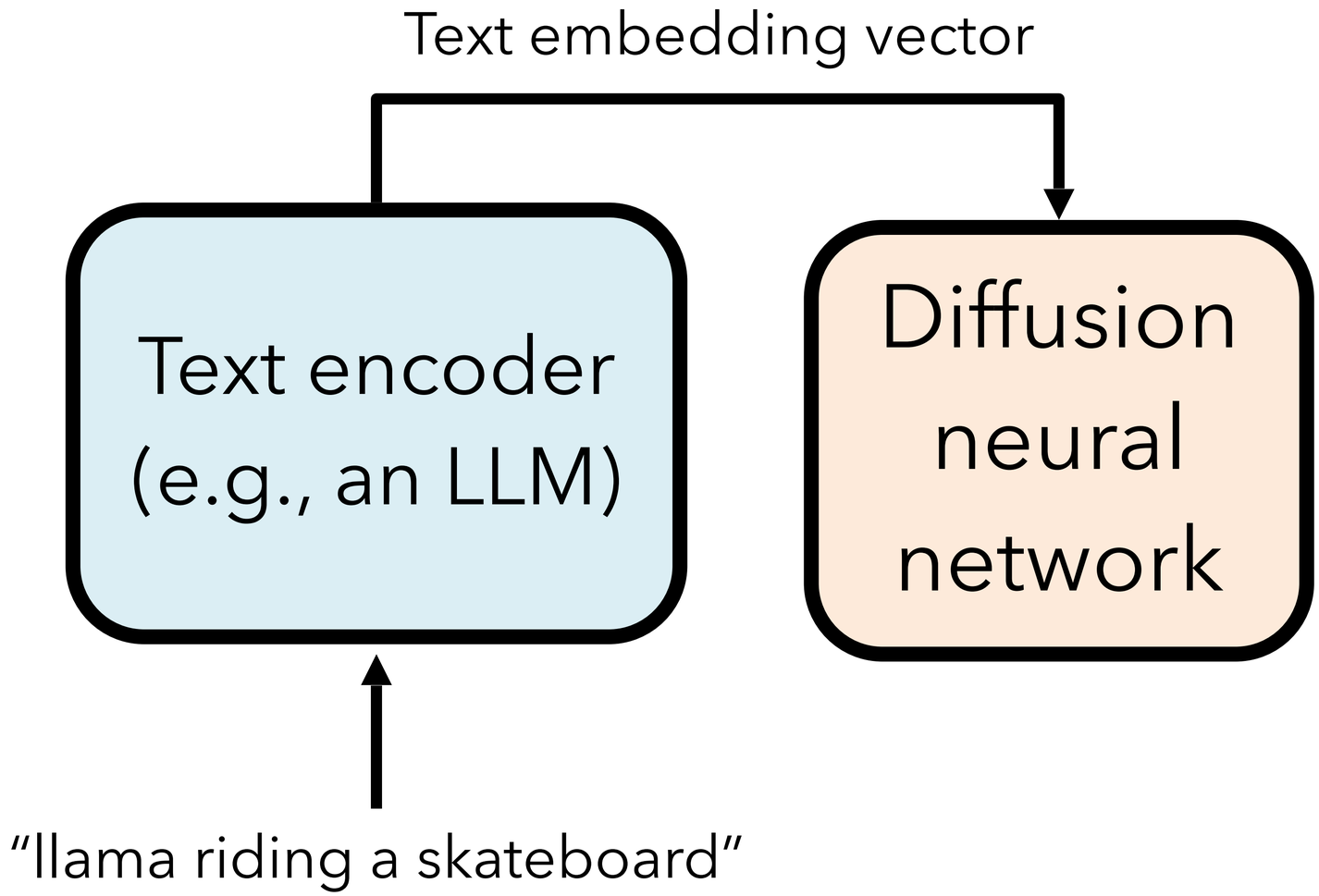
Source: N. Snavely

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”)

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

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

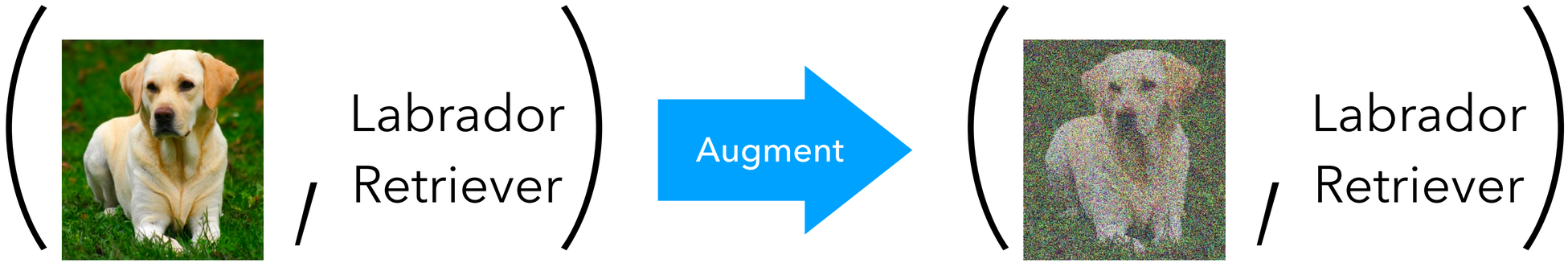
$$\tilde{\epsilon}_t(x_t, t, y) = \epsilon_\theta(x_t, t) - \gamma \nabla_{x_t} \log p(x_t | y)$$

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

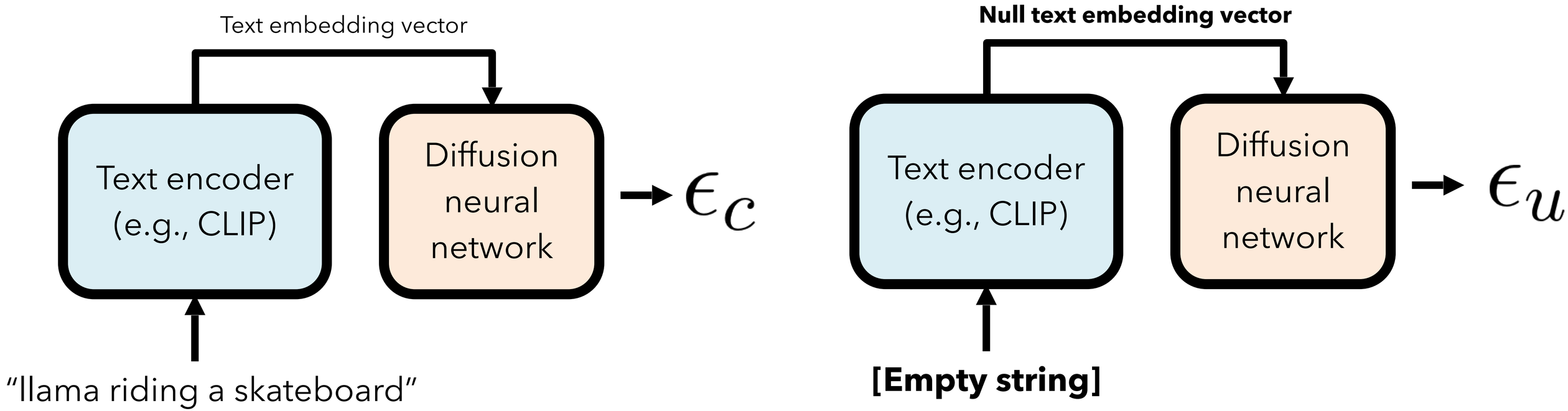
Classifier Guidance

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]

Denoisers as score functions

Consider the distribution of noisy images:

Suppose $\tilde{\mathbf{x}} = \mathbf{x} + \sigma\epsilon$, where $\epsilon \sim \mathcal{N}(0, I)$ Then $q_\sigma(\tilde{\mathbf{x}}) = \int_{\mathbf{x}} \mathcal{N}(\tilde{\mathbf{x}}; \mathbf{x}, \sigma^2 I) p_D(\mathbf{x}) d\mathbf{x}$

Tweedie's formula:

$$\mathbb{E}_{\mathbf{x} \sim p(\mathbf{x}|\tilde{\mathbf{x}})}[\mathbf{x}] = \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$$

If we want to denoise an input, (perhaps surprisingly), you can just take a gradient step using the score function of the *noisy* distribution.

Denoisers as score functions

Tweedie's formula:

$$\mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\tilde{\mathbf{x}})}[\mathbf{x}] = \tilde{\mathbf{x}} + \sigma^2 \nabla_{\tilde{\mathbf{x}}} \log q_\sigma(\tilde{\mathbf{x}})$$

The noise is: $\epsilon = \dfrac{\tilde{\mathbf{x}} - \mathbf{x}}{\sigma} \approx \dfrac{\tilde{\mathbf{x}} - \mathbb{E}_{\mathbf{x}\sim p(\mathbf{x}|\tilde{\mathbf{x}})}[\mathbf{x}]}{\sigma}$

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

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):

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

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

Next class: more diffusion