Introduction to Computer Vision: Generative models

Natalia Neverova

Last lecture's recap

- Semantic segmentation
- Extending R-CNN to other tasks
- Instance segmentation
- Panoptic segmentation
- Human pose estimation
- .. and others

Today's focus

- Supervised vs unsupervised learning
- Discriminative vs generative models
- Diffusion models
- Diffusion models for images

Supervised vs Unsupervised learning

Supervised learning

Data: (x,y)
x is data, y is label

Goal: learn a function to map x->y

Examples: classification, regression,
object detection, semantic segmentation, etc.

classification

cat

Supervised vs Unsupervised learning

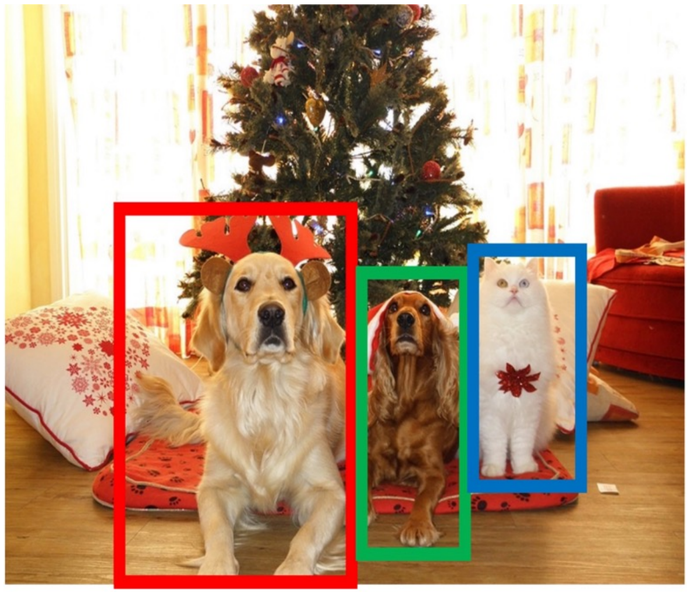
Supervised learning

Data: (x,y)
x is data, y is label

Goal: learn a function to map x->y

Examples: classification, regression,
object detection, semantic segmentation, etc.

object detection



dog, dog, cat

Supervised vs Unsupervised learning

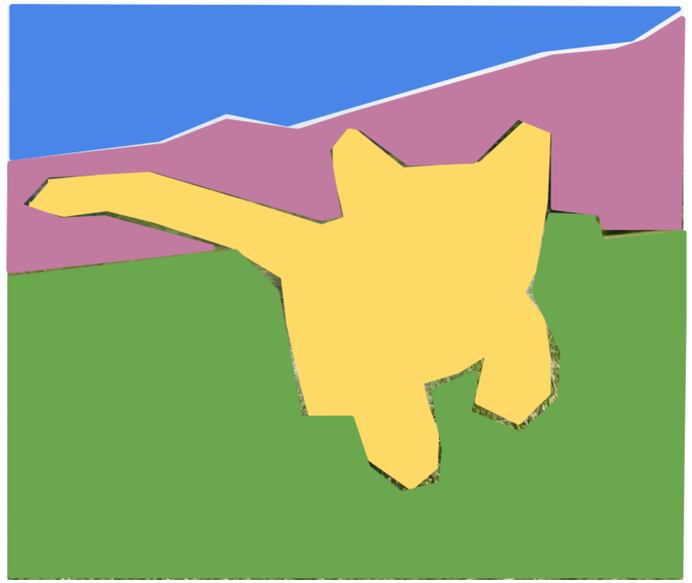
Supervised learning

Data: (x,y)
x is data, y is label

Goal: learn a function to map x->y

Examples: classification, regression,
object detection, semantic segmentation, etc.

semantic segmentation



GRASS, CAT, TREE, SKY

Supervised vs Unsupervised learning

Supervised learning

Data: (x,y)
x is data, y is label

Goal: learn a function to map x->y

Examples: classification, regression, object detection, semantic segmentation, etc.

Unsupervised learning

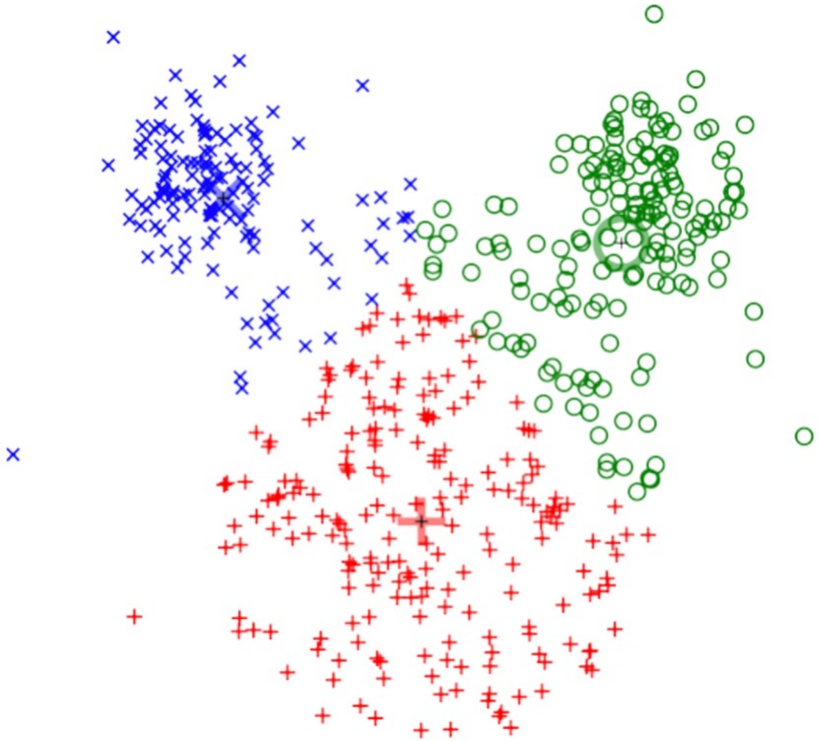
Data: x
Just data, no labels!

Goal: learn some *underlying* hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised learning

clustering (e.g. K-Means)



Unsupervised learning

Data: x

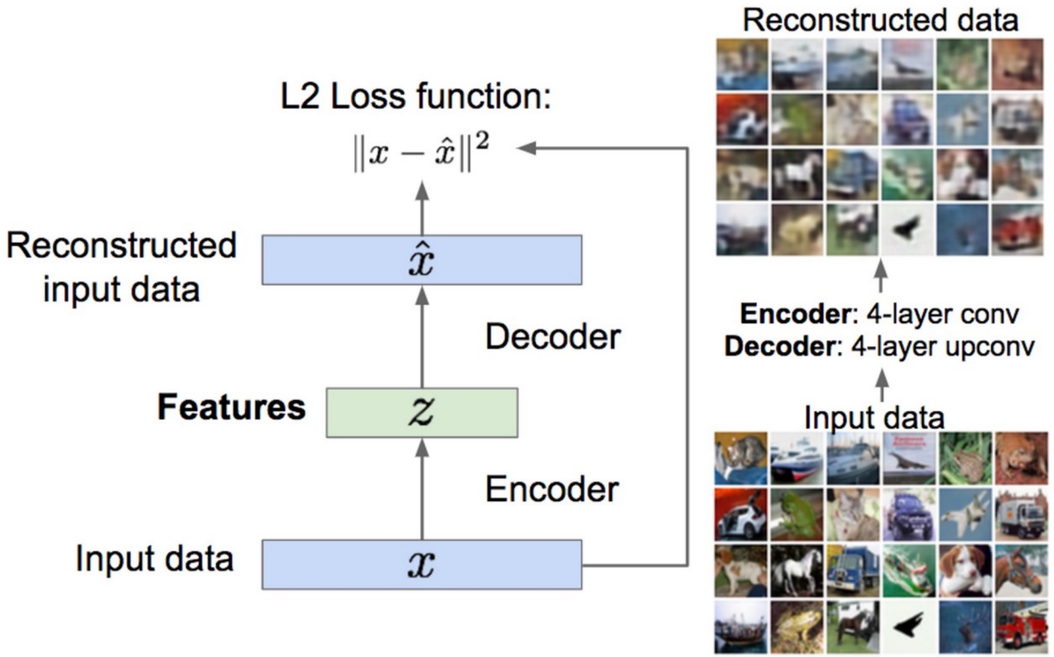
Just data, no labels!

Goal: learn some *underlying* hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Supervised vs Unsupervised learning

feature learning
(e.g. autoencoders)



Unsupervised learning

Data: x

Just data, no labels!

Goal: learn some *underlying* hidden structure of the data

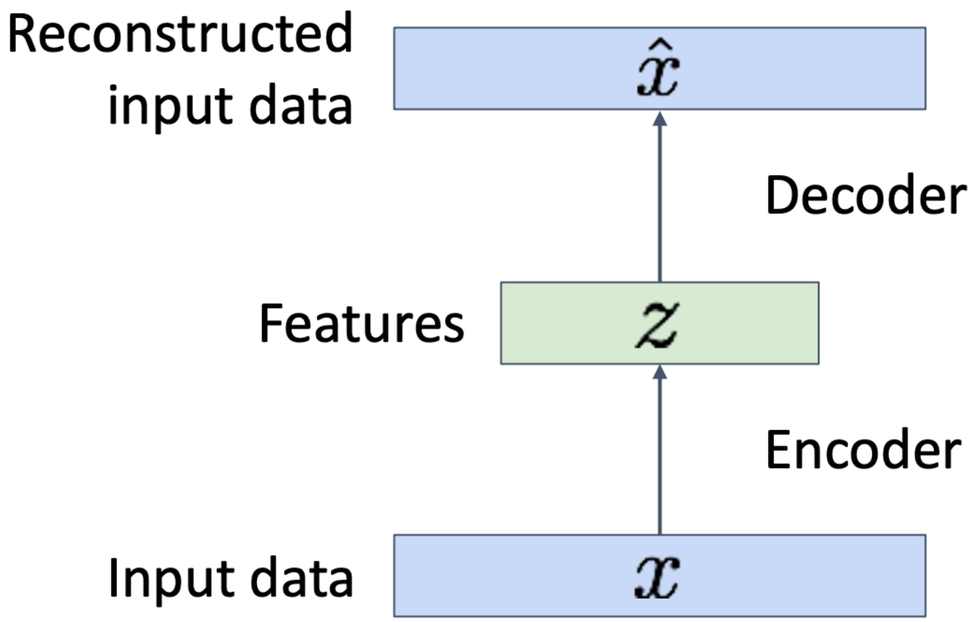
Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Autoencoders

Unsupervised method for learning feature vectors from raw data x, without any labels. Features should extract useful information (maybe object identities, properties, scene type, etc.) that we can use for downstream tasks.

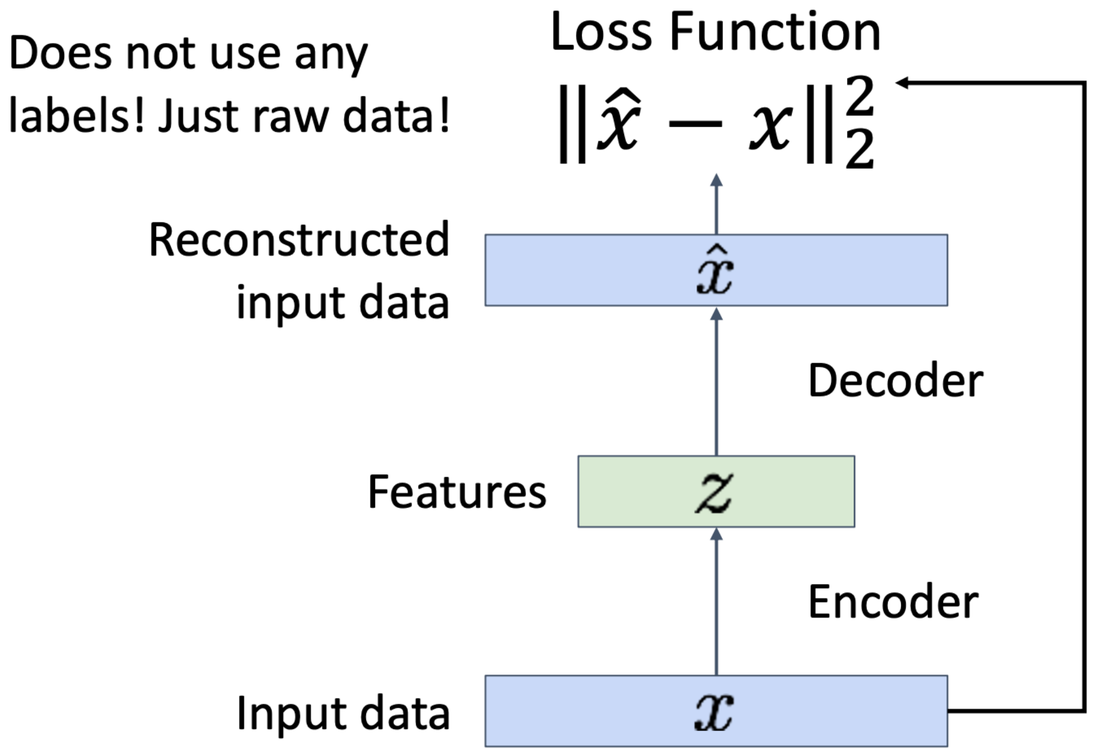
Autoencoders

Idea: use the features to reconstruct the input data with a decoder. “Autoencoding” – encoding itself.



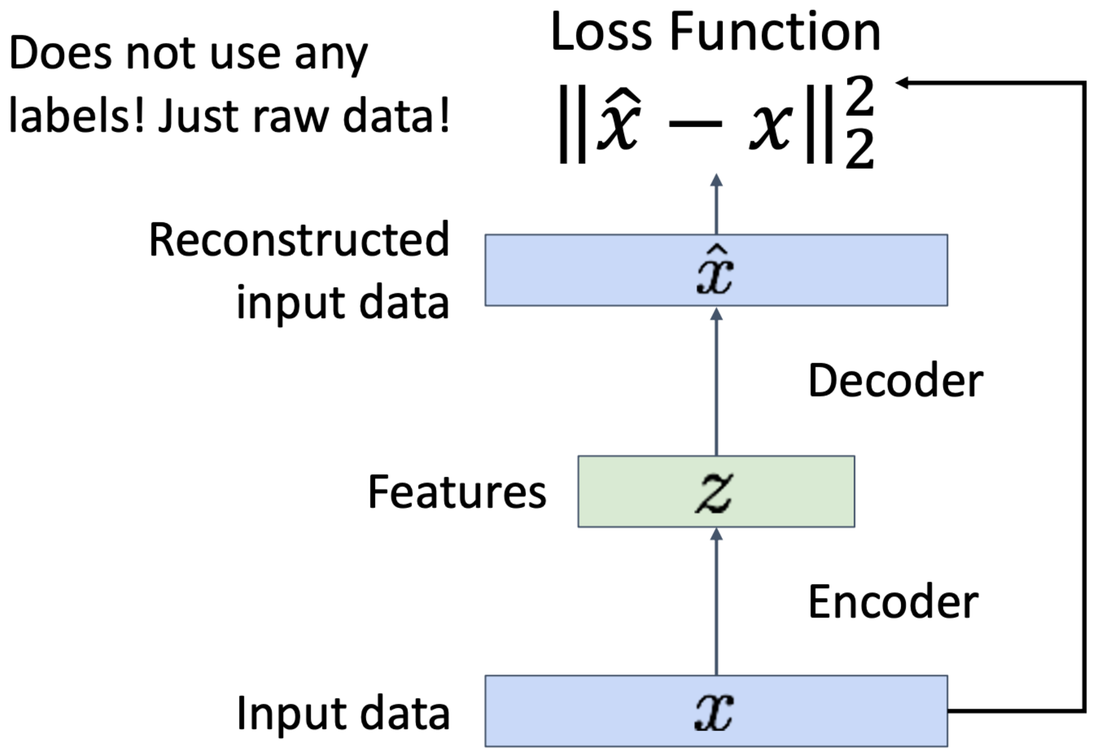
Autoencoders

Loss: L2 distance between input and reconstructed data. Features need to be lower dimensional than data.



Autoencoders

Loss: L2 distance between input and reconstructed data. Features need to be lower dimensional than data.



After training, throw away decoder and use encoder for a downstream task.

Supervised vs Unsupervised learning

Supervised learning

Data: (x,y)
x is data, y is label

Goal: learn a function to map x->y

Examples: classification, regression, object detection, semantic segmentation, etc.

Unsupervised learning

Data: x
Just data, no labels!

Goal: learn some *underlying* hidden structure of the data

Examples: clustering, dimensionality reduction, feature learning, density estimation, etc.

Discriminative vs Generative Models

Discriminative Model:

learn a probability distribution p(y|x)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)

Discriminative vs Generative Models

Discriminative Model:

learn a probability distribution p(y|x)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)

Data: x

Label: y

cat

Discriminative vs Generative Models

Probability recap:

Density Function

p(x) assigns a positive number to each possible x; higher numbers mean x is more likely

Density functions are normalized:

$$\int_X p(x)dx = 1$$

Different values of x **compete** for density

Data: x

Label: y

cat

Discriminative vs Generative Models

Discriminative Model:

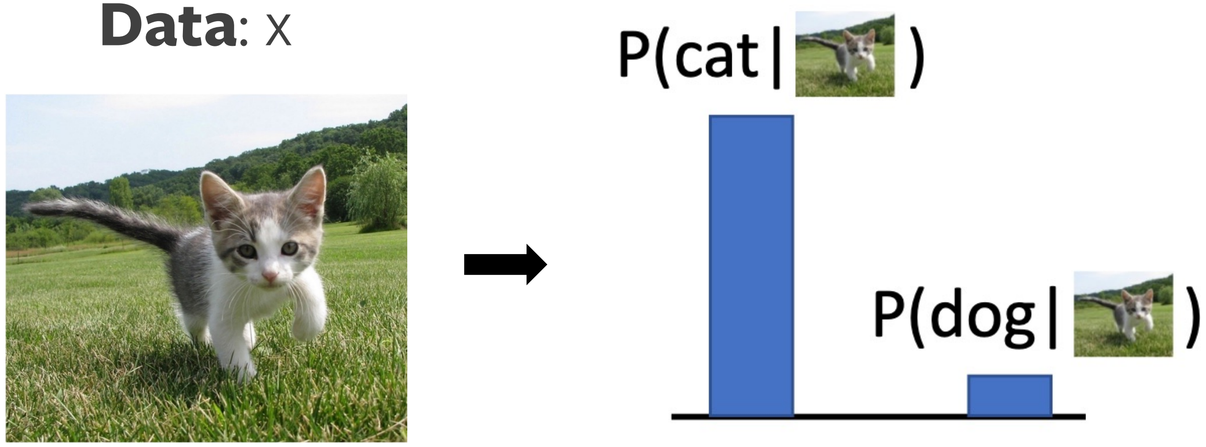
learn a probability distribution p(y|x)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)



Discriminative vs Generative Models

Discriminative Model:

learn a probability distribution p(y|x)

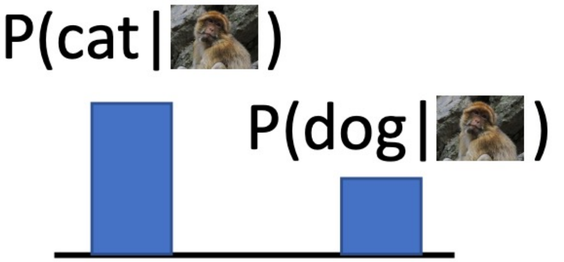
Generative Model:

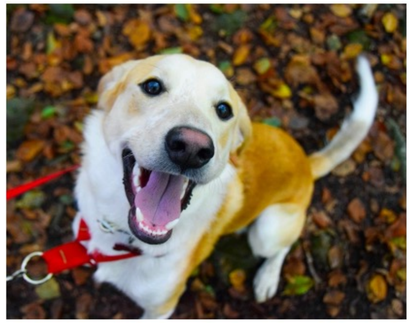
learn a probability distribution p(x)

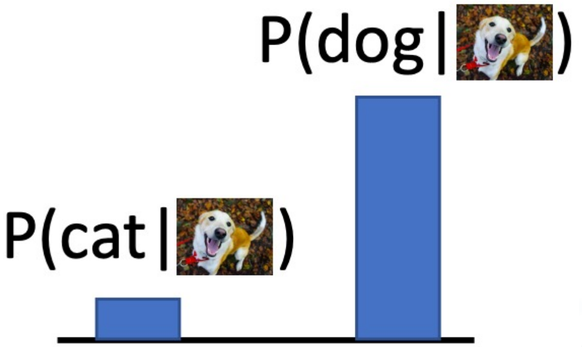
Conditional Generative Model:

learn p(x|y)

Data: x







Discriminative model: no way for the model to handle unreasonable inputs; it must give label distributions for all images

Discriminative vs Generative Models

Discriminative Model:

learn a probability distribution p(y|x)

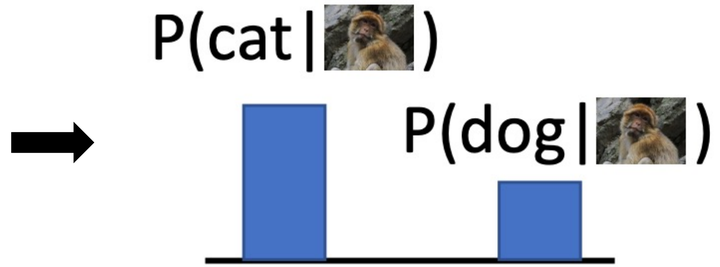
Generative Model:

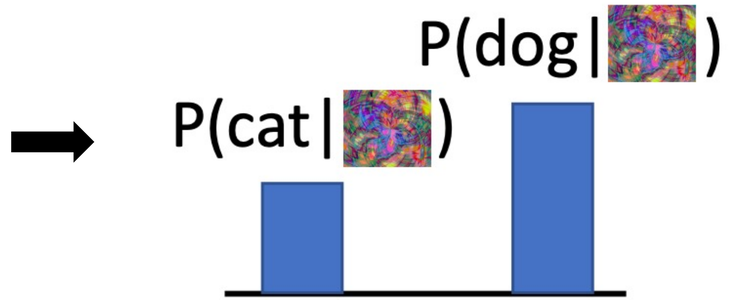
learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)

Data: x





Discriminative model: no way for the model to handle unreasonable inputs; it must give label distributions for all images

Discriminative vs Generative Models

Discriminative Model:

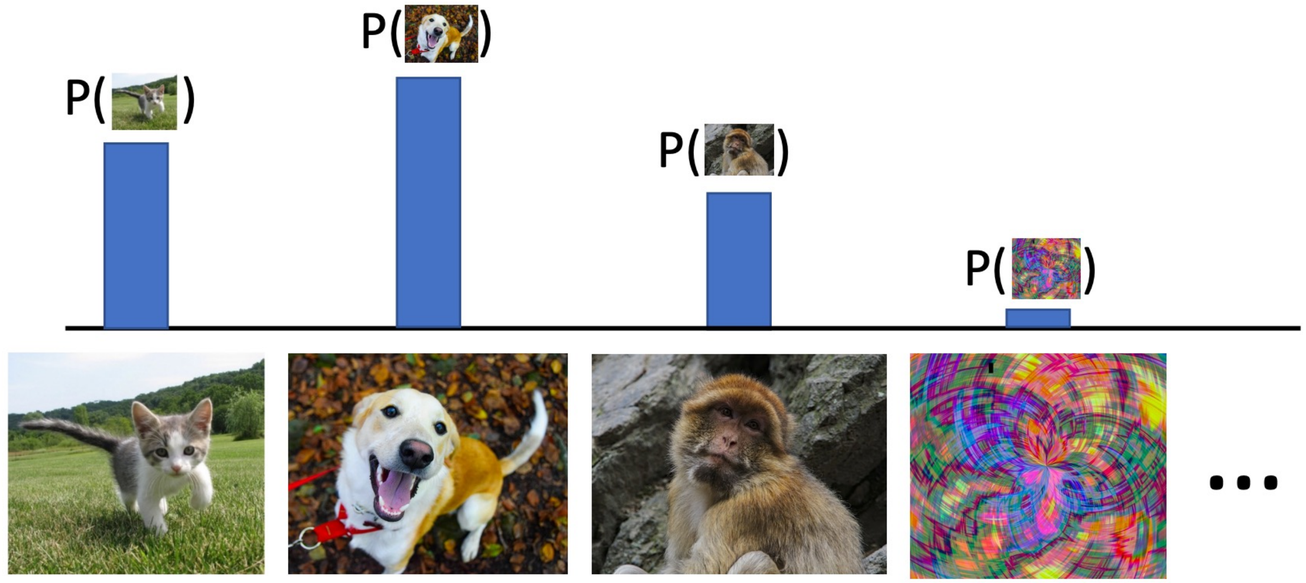
learn a probability distribution p(y|x)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)



Generative model: all possible images compete with each other for probability mass

Discriminative vs Generative Models

Discriminative Model:

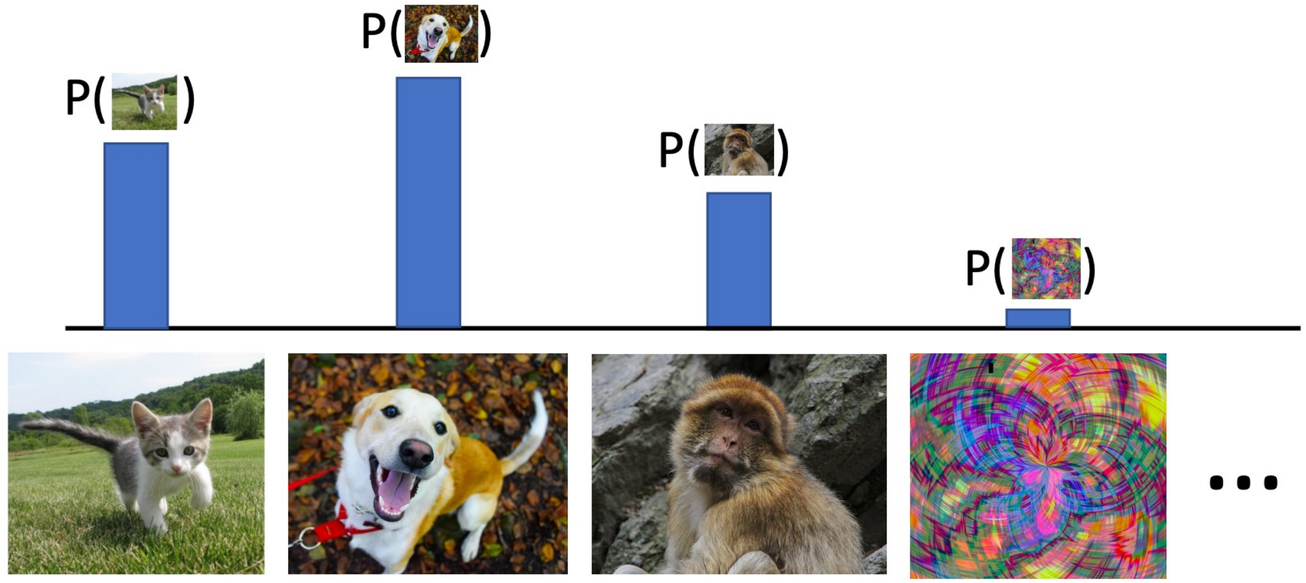
learn a probability distribution p(y|x)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)



Generative model: all possible images compete with each other for probability mass

Requires deep image understanding! Is a dog more likely to sit or stand? How about 3-legged dog vs 3-armed monkey?

Discriminative vs Generative Models

Discriminative Model:

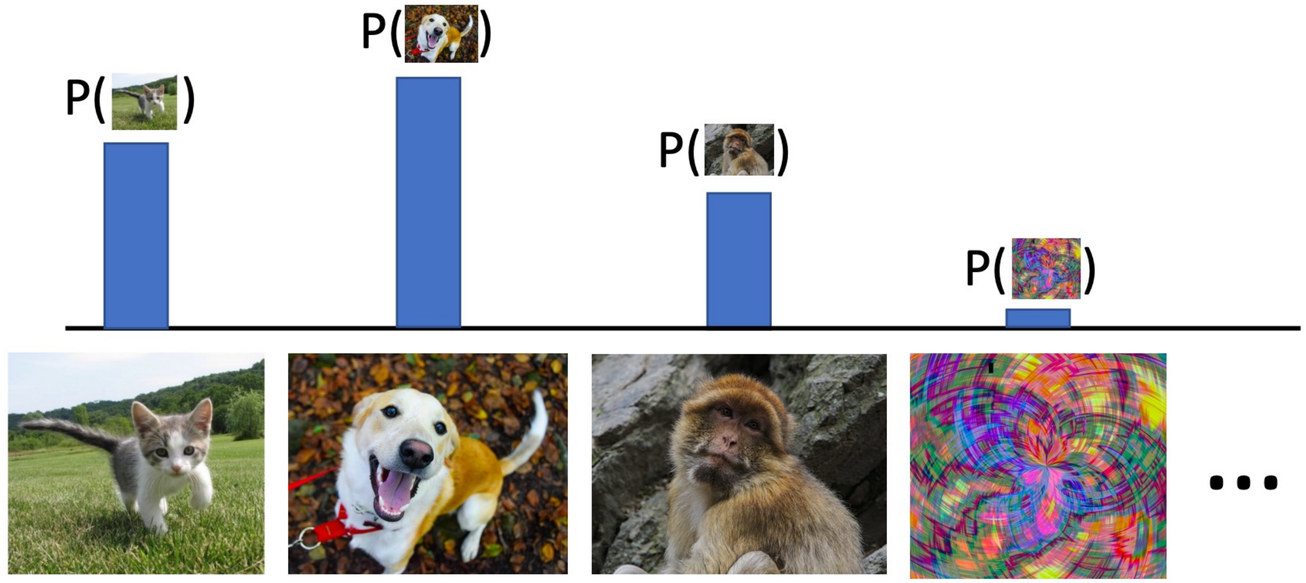
learn a probability distribution p(y|x)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)



Generative model: all possible images compete with each other for probability mass

Model can "reject" unreasonable inputs by assigning them small values

Discriminative vs Generative Models

Discriminative Model:

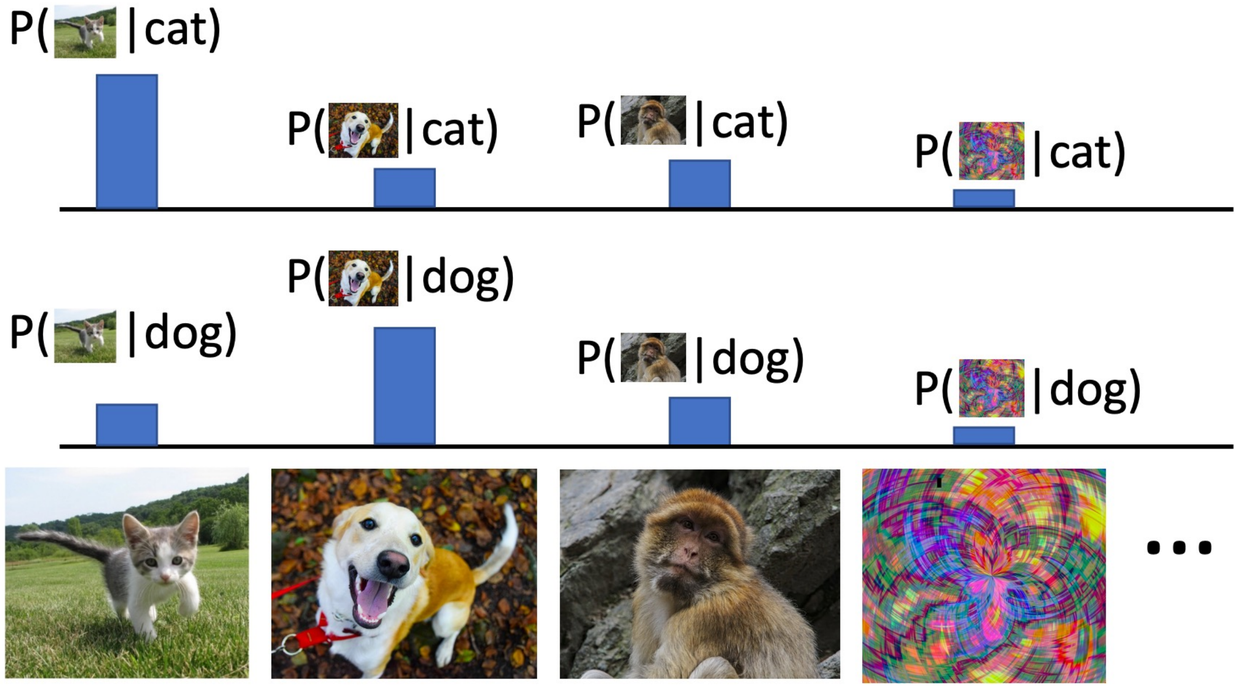
learn a probability distribution p(y|x)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)



Conditional generative model: each possible label induces a competition among all images

Discriminative vs Generative Models

Discriminative Model:

learn a probability distribution p(y|x)

➡

Assign labels to data

Feature learning (with labels)

Generative Model:

learn a probability distribution p(x)

Conditional Generative Model:

learn p(x|y)

Discriminative vs Generative Models

Discriminative Model:

learn a probability distribution p(y|x)

➡ Assign labels to data

Feature learning (with labels)

Generative Model:

learn a probability distribution p(x)

➡ Detect outliers

Feature learning (without labels)

Sample to **generate** new data

Conditional Generative Model:

learn p(x|y)

Discriminative vs Generative Models

Discriminative Model:

learn a probability distribution p(y|x)

→ Assign labels to data

→ Feature learning (with labels)

Generative Model:

learn a probability distribution p(x)

→ Detect outliers

→ Feature learning (without labels)

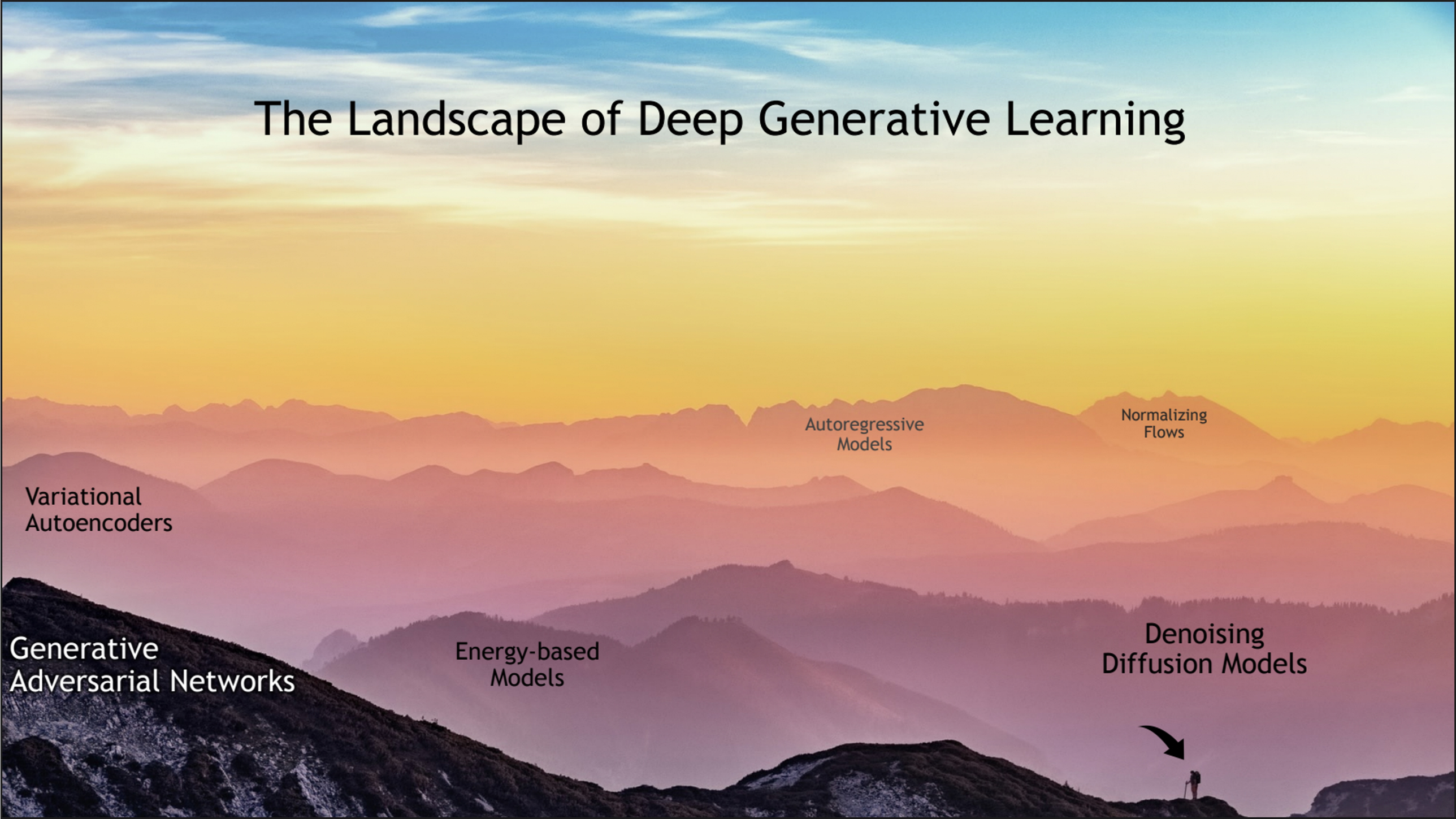
→ Sample to **generate** new data

Conditional Generative Model:

learn p(x|y)

→ Assign labels, while rejecting outliers!

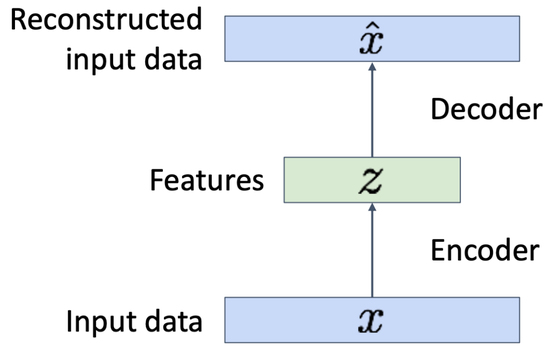
→ Generate new data conditioned on labels


The Landscape of Deep Generative Learning
Autoregressive Models
Normalizing Flows
Variational Autoencoders
Generative Adversarial Networks
Energy-based Models
Denoising Diffusion Models

Generative Adversarial Networks (GANs)

Setup: Assume we have data x_i drawn from a distribution $p_{data}(x)$. Want to sample from p_{data}.

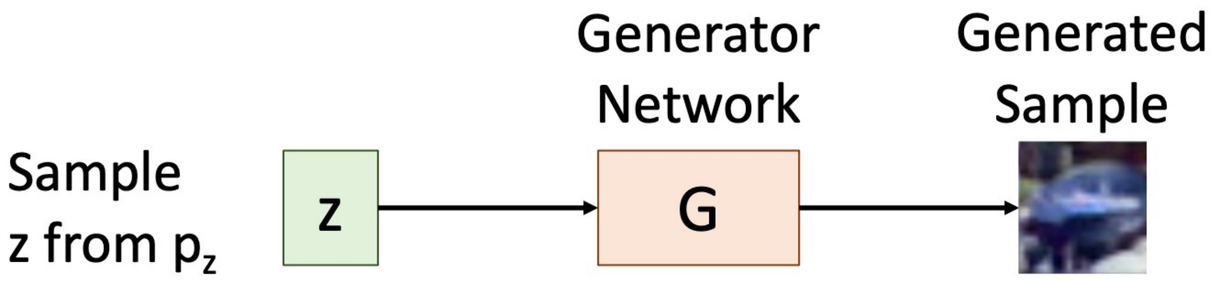
Idea: introduce a latent variable z with simple prior p(z). Sample z from p(z) and pass to a Generator Network x = G(z). Then x is a sample from the Generator distribution p_G. Want $p_G = p_{data}$.



Generative Adversarial Networks (GANs)

Setup: Assume we have data x_i drawn from a distribution $p_{data}(x)$. Want to sample from p_{data}.

Idea: introduce a latent variable z with simple prior p(z). Sample z from p(z) and pass to a Generator Network x = G(z). Then x is a sample from the Generator distribution p_G. Want $p_G = p_{data}$.

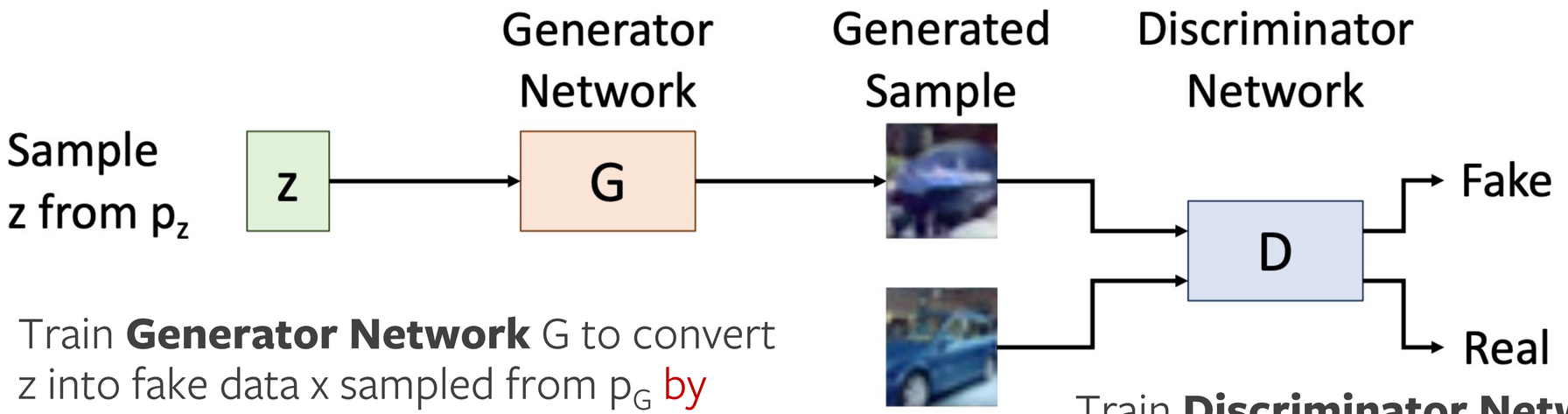


Train **Generator Network** G to convert z into fake data x sampled from p_G

Generative Adversarial Networks (GANs)

Setup: Assume we have data x_i drawn from a distribution $p_{data}(x)$. Want to sample from p_{data}.

Idea: introduce a latent variable z with simple prior p(z). Sample z from p(z) and pass to a Generator Network x = G(z). Then x is a sample from the Generator distribution p_G. Want $p_G = p_{data}$.



Train **Generator Network** G to convert z into fake data x sampled from p_G by "fooling" the discriminator D

Train **Discriminator Network** D to classify data as real or fake (1/0)

Generative Adversarial Networks (GANs)

Training: Jointly train generator G and discriminator D with a **minimax game**

$$\min_G \max_D \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D\left(G(z)\right) \right) \right] \right)$$

Generative Adversarial Networks (GANs)

Training: Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_{G} \max_{D} \left(E_{x \sim p_{data}} [\log D(x)] + E_{z \sim p(z)} \left[\log \left(1 - D(G(z)) \right) \right] \right)$$

Generative Adversarial Networks (GANs)

Training: Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants D(x) = 1 for real data

Discriminator wants D(x) = 0 for fake data

$$\min_{G}\max_{D}\left(E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}\left[\log\left(1 - D(G(z))\right)\right]\right)$$

Generator wants D(x) = 1 for fake data

Train G and D using alternating gradient updates.

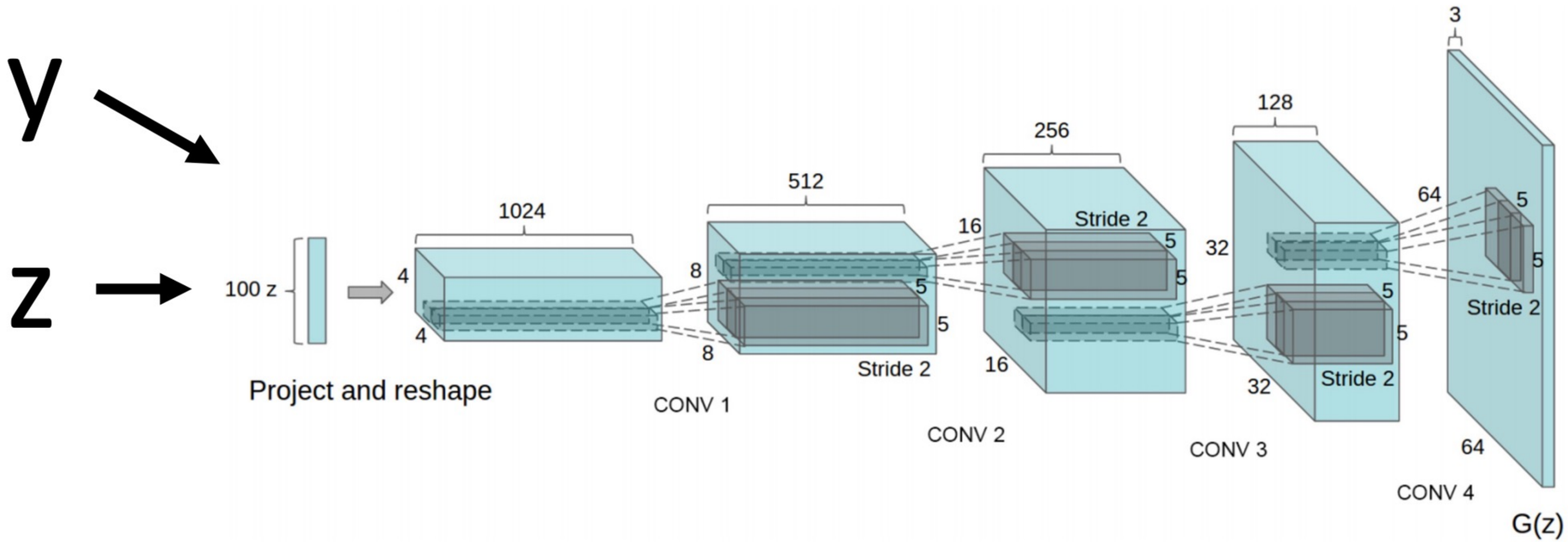
This minimax game achieves its global minimum when $p_G = p_{data}$!

Generative Adversarial Networks (GANs)

Karras et al, “Analyzing and Improving the Image Quality of StyleGAN”, CVPR 2020

Conditional GANs

Recall: Conditional generative models learn p(x|y) instead of p(x). Make generator and discriminator both take label y as an additional input.



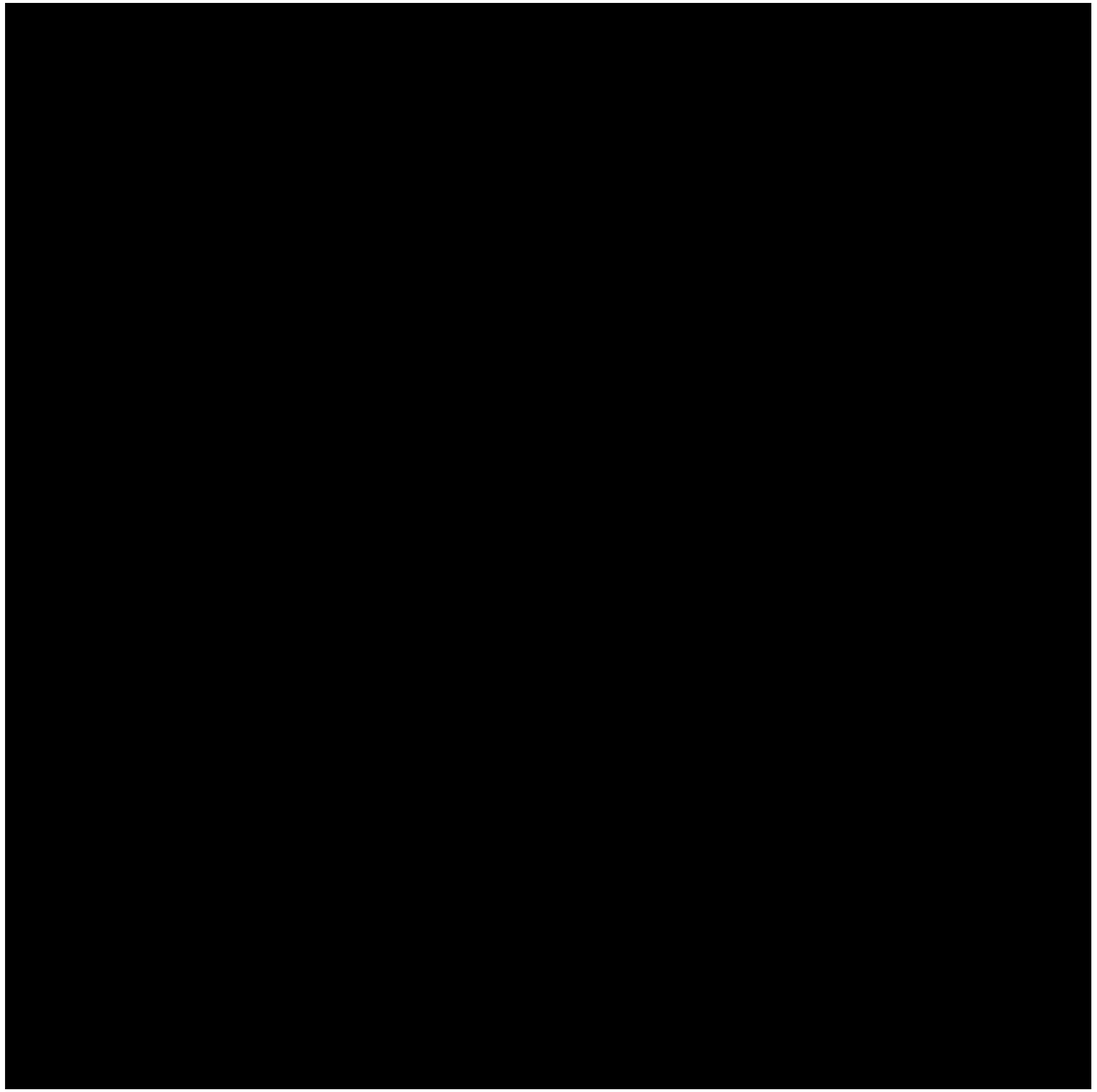
Karras et al, "Analyzing and Improving the Image Quality of StyleGAN", CVPR 2020

Conditional GANs: BigGAN

512x512 images on ImageNet

Brock et al. ICLR 2019

Latest model: StyleGan-T



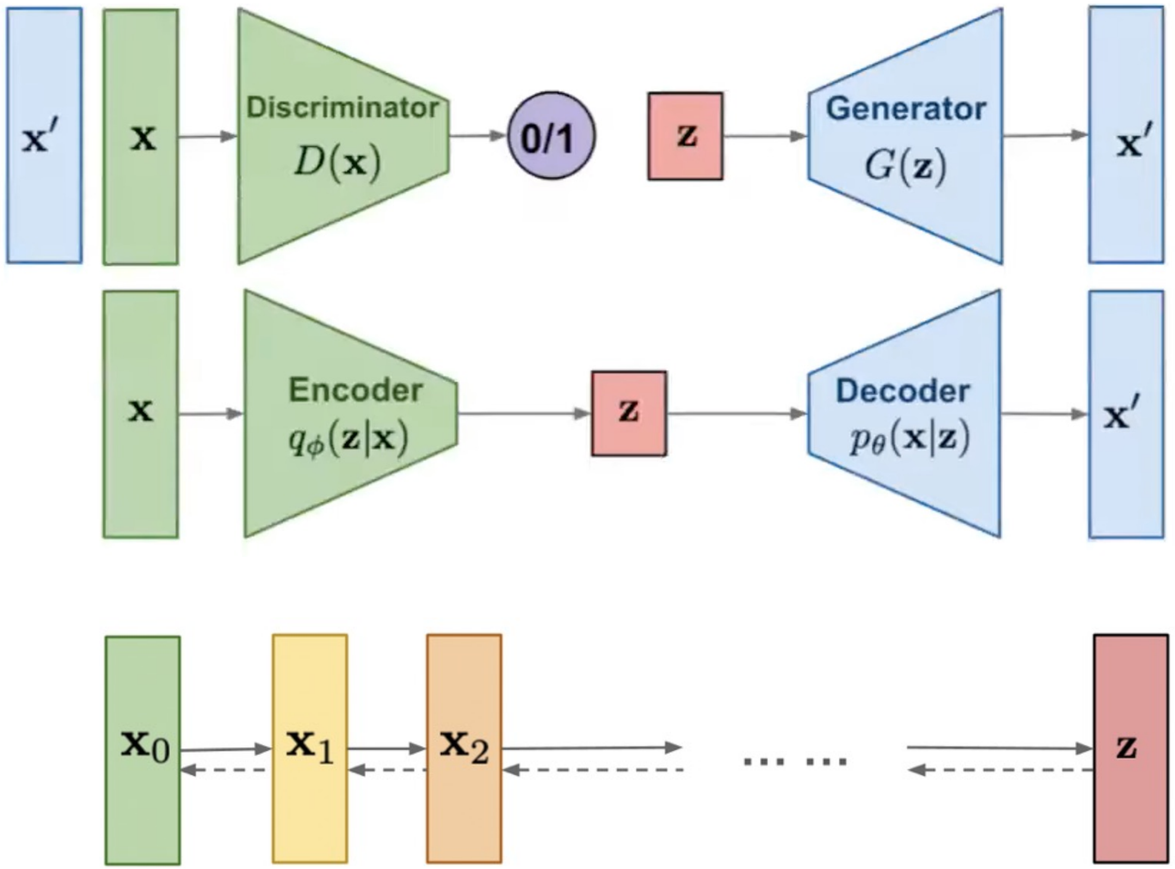
Sauer et al. ICML 2023

Diffusion models

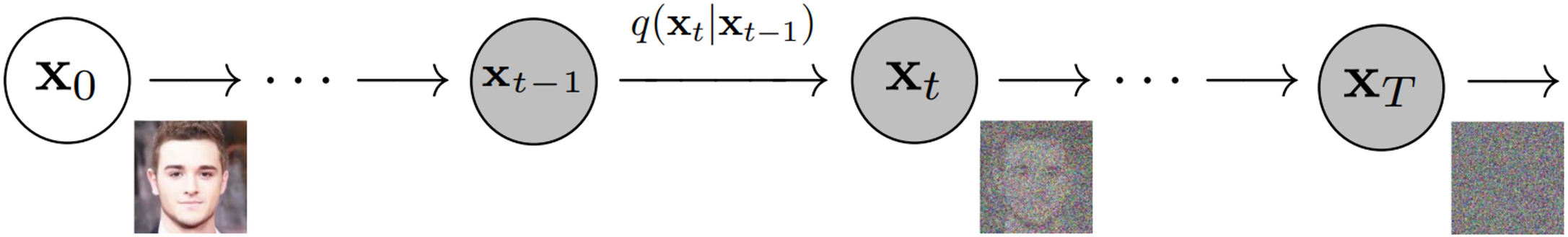
Diffusion models are inspired by non-equilibrium thermodynamics.

They define a chain of diffusion steps to slowly add random noise to data (forward process) and then learn to reverse the diffusion process to reconstruct desired data samples from the noise (reverse process).

Diffusion models



Diffusion models: forward process

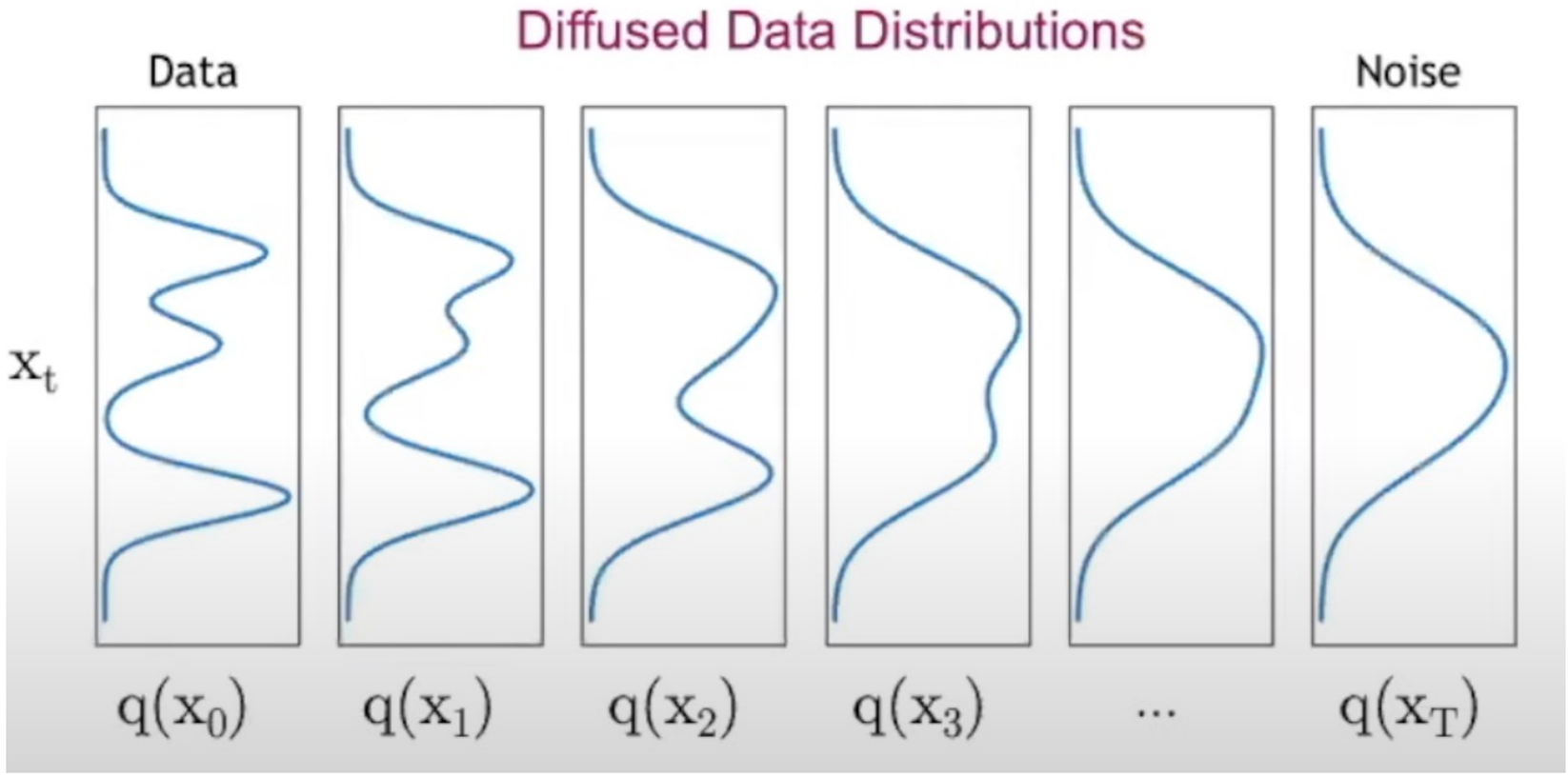


(1) Take data x_0 and keep gradually adding very small amounts of Gaussian noise to it

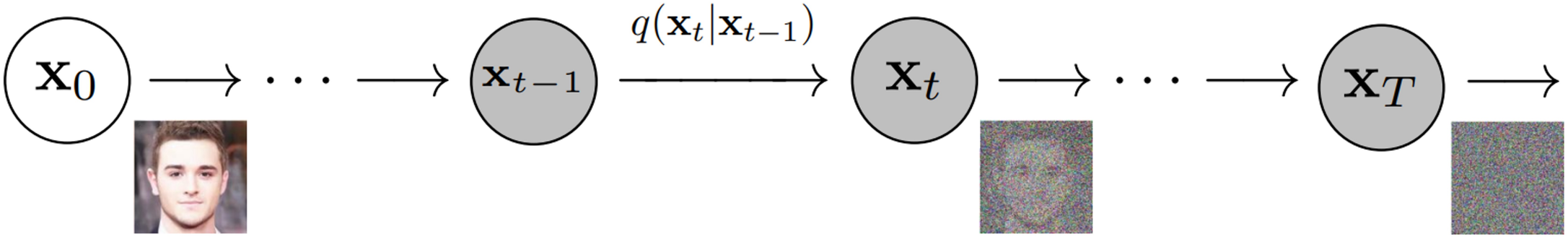
(2) Repeat this process for T steps – as the timesteps increase, the more features of the original input are destroyed

You can prove that as T approaches infinity, you eventually end with Isotropic Gaussian (i.e. pure random noise)

Diffusion models: forward process



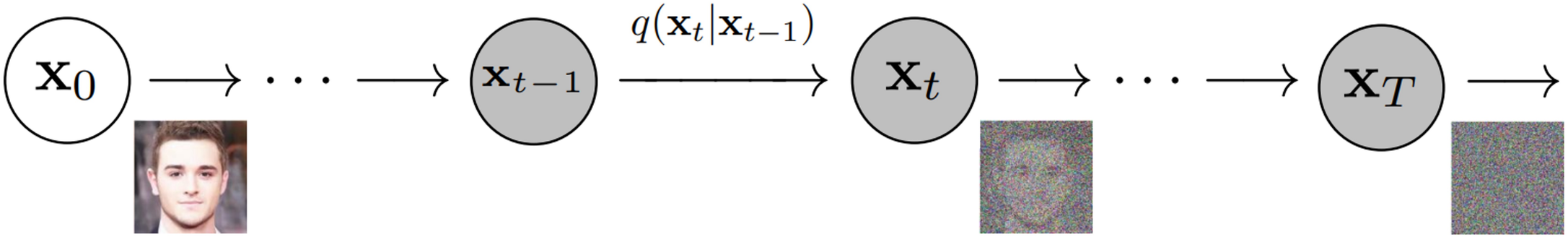
Diffusion models: forward process



$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

distribution of x_t given x_{t-1}

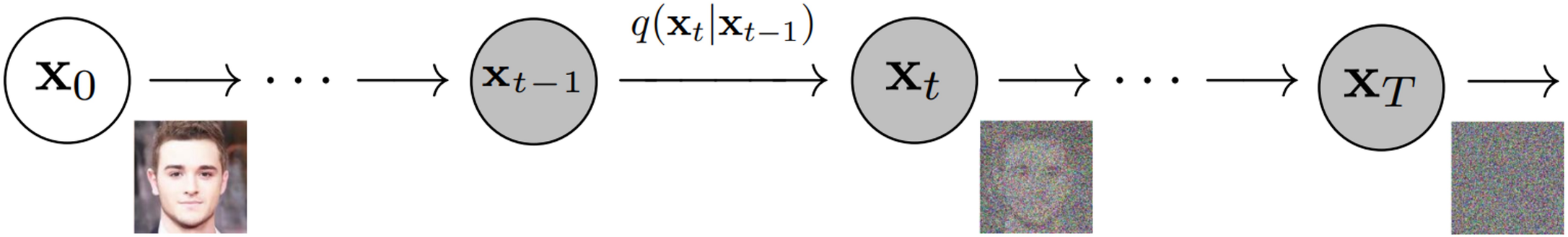
Diffusion models: forward process



controls schedule $\{\beta_t \in (0,1)\}_{t=1}^{T}$

$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_{t-1}, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I})$$

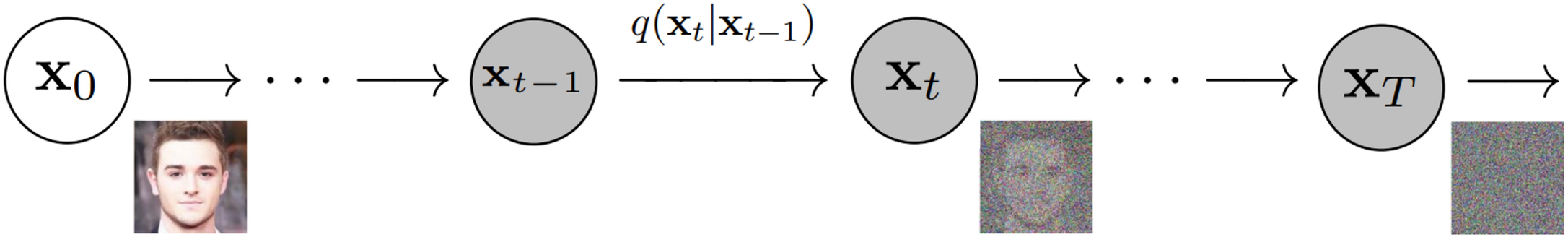
Diffusion models: forward process



$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_{t-1}, \quad \epsilon_{t-1}, \epsilon_{t-2}, \ldots \sim \mathcal{N}(0, \mathbf{I})$$

let's change variables: $\alpha_t = 1 - \beta_t$

Diffusion models: forward process



$$\mathbf{x}_t = \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon_{t-1} =$$

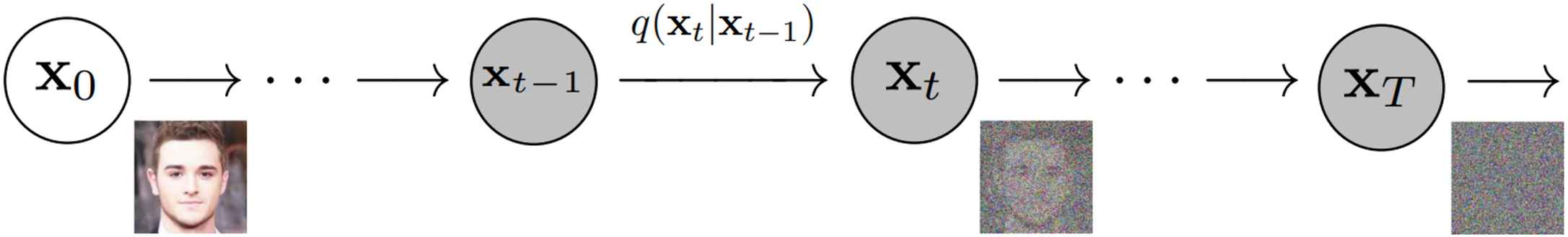
$$= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon_{t-1}$$

$$= \sqrt{\alpha_t\alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\hat{\epsilon}_{t-2}$$

merges two Gaussians

Recall that when we merge two Gaussians with different variance, $\mathcal{N}(0, \sigma_1^2, \mathbf{I})$ and $\mathcal{N}(0, \sigma_2^2\mathbf{I})$, the new distribution is $\mathcal{N}(0, (\sigma_1^2 + \sigma_2^2)\mathbf{I})$. Here the merged standard deviation is $\sqrt{(1-\alpha_t) + \alpha_t(1-\alpha_{t-1})} = \sqrt{1-\alpha_t\alpha_{t-1}}$.

Diffusion models: forward process

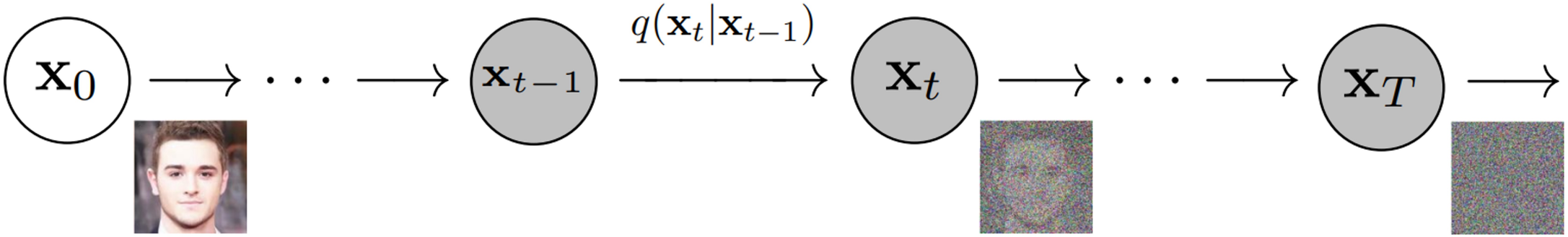


$$\mathbf{x}_t = \sqrt{\alpha_t \alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t\alpha_{t-1}}\hat{\epsilon}_{t-2} =$$

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

$$\bar{\alpha}_t = \prod_{i=1}^{t} \alpha_i$$

Diffusion models: forward process



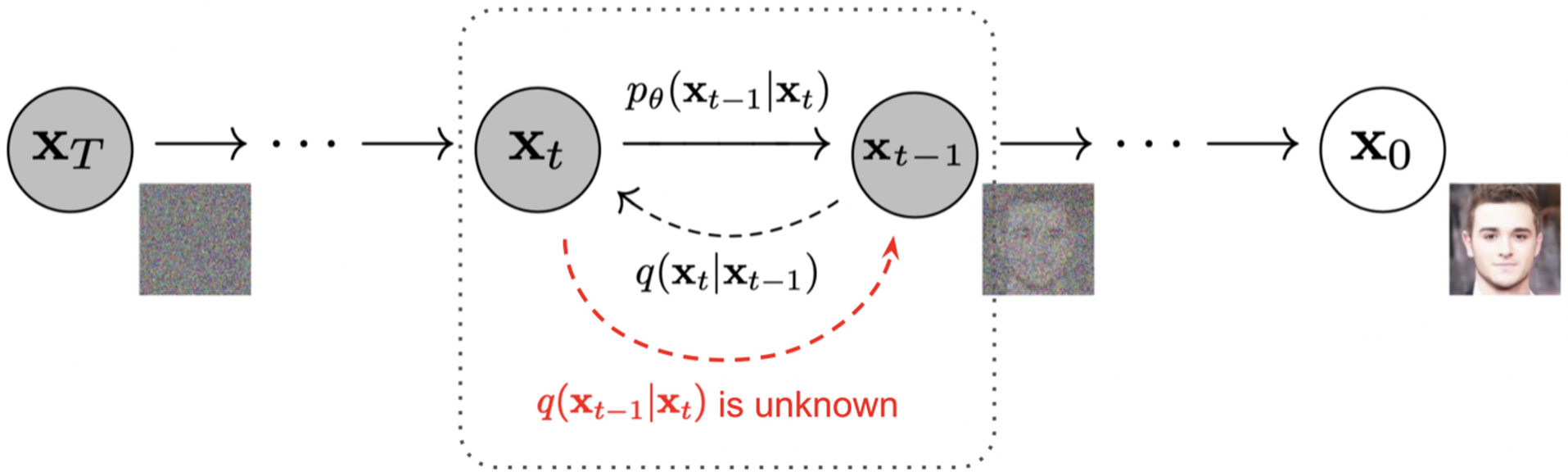
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$

distribution of x_t given x_{t-1}

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

distribution of x_t given x_0

Diffusion models: reverse process



The goal of a diffusion model is to **learn the reverse denoising process** to iteratively undo the forward process

In this way, the reverse process appears as if it is generating new data from random noise!

Diffusion models: reverse process

$$L_t = \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[||\epsilon_t - \epsilon_\theta(\mathbf{x}_t, t)||^2\right]$$

$$= \mathbb{E}_{t\sim[1,T],\mathbf{x}_0,\epsilon_t}\left[||\epsilon_t - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon_t, t)||^2\right]$$

Learning objective: **predict the noise at time step t from the input $\mathbf{x_t}$**

Diffusion models: training

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

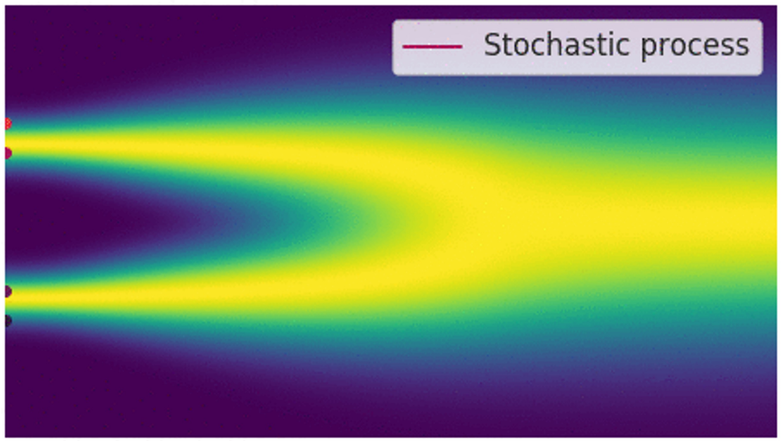
Diffusion models: sampling

Algorithm 2 Sampling

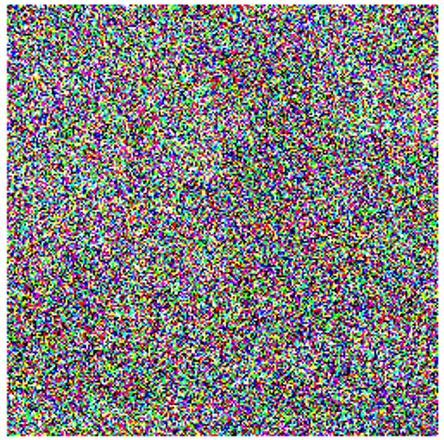
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \ldots, 1$ **do**
3: $\quad \mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ if $t > 1$, else $\mathbf{z} = \mathbf{0}$
4: $\quad \mathbf{x}_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}}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

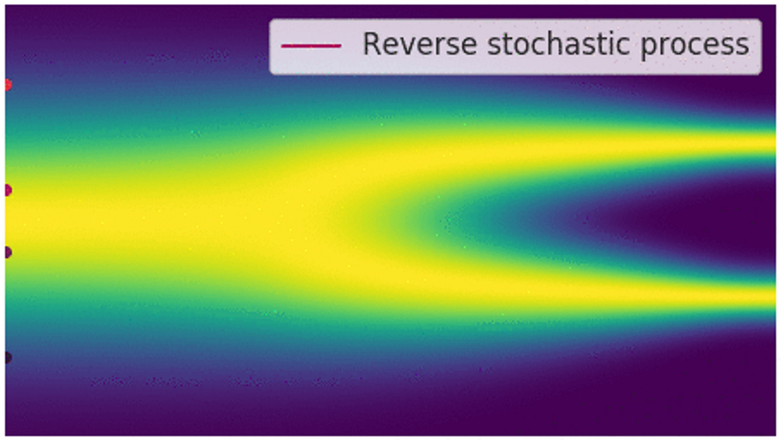
$$\sigma_t^2 = \tilde{\beta}_t = \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t} \beta_t$$

Diffusion for image generation



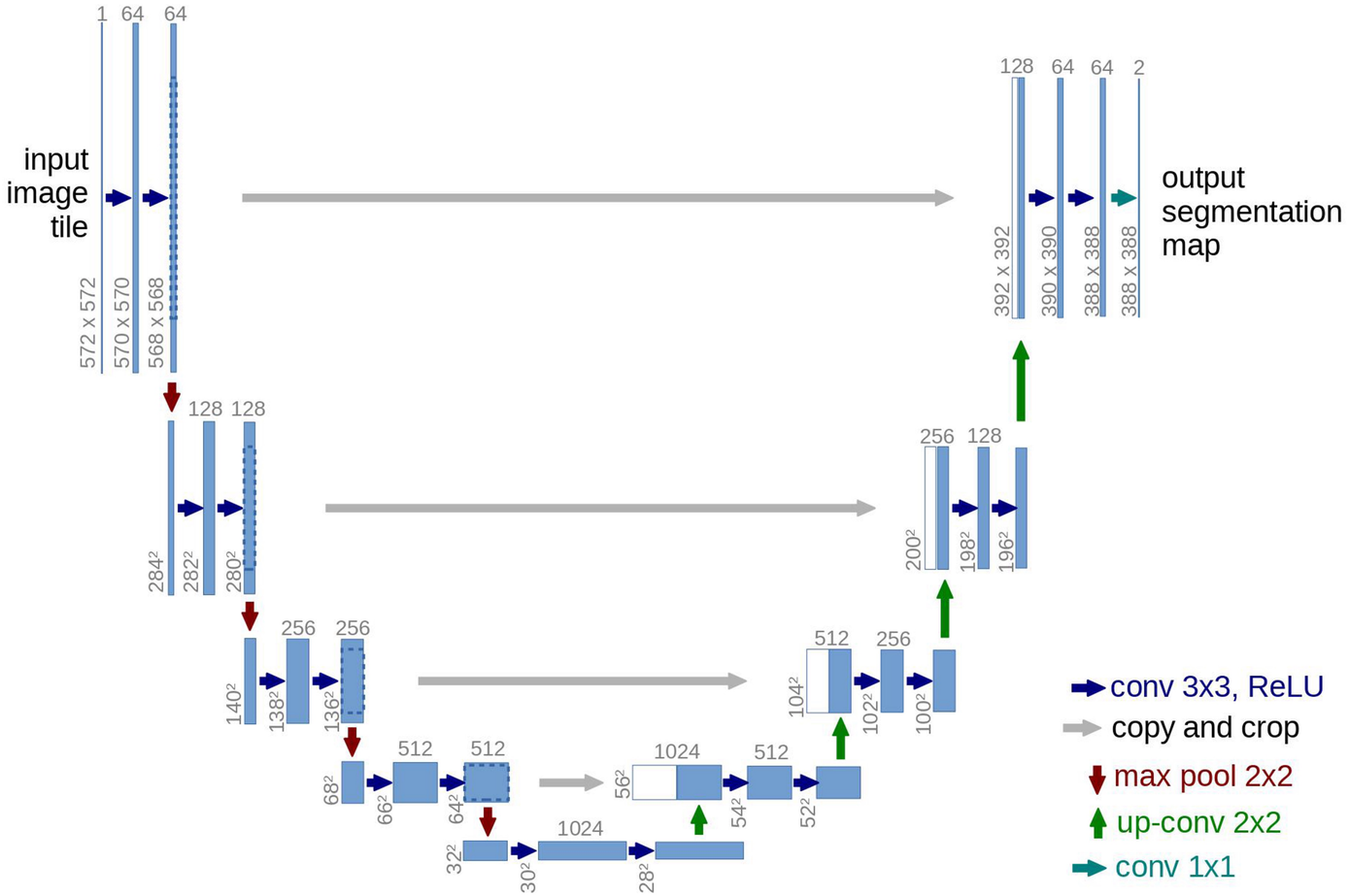
Forward process: converting the image distribution to pure noise





Reverse process: sampling from the image distribution, starting with pure noise

Back to UNet with tricks



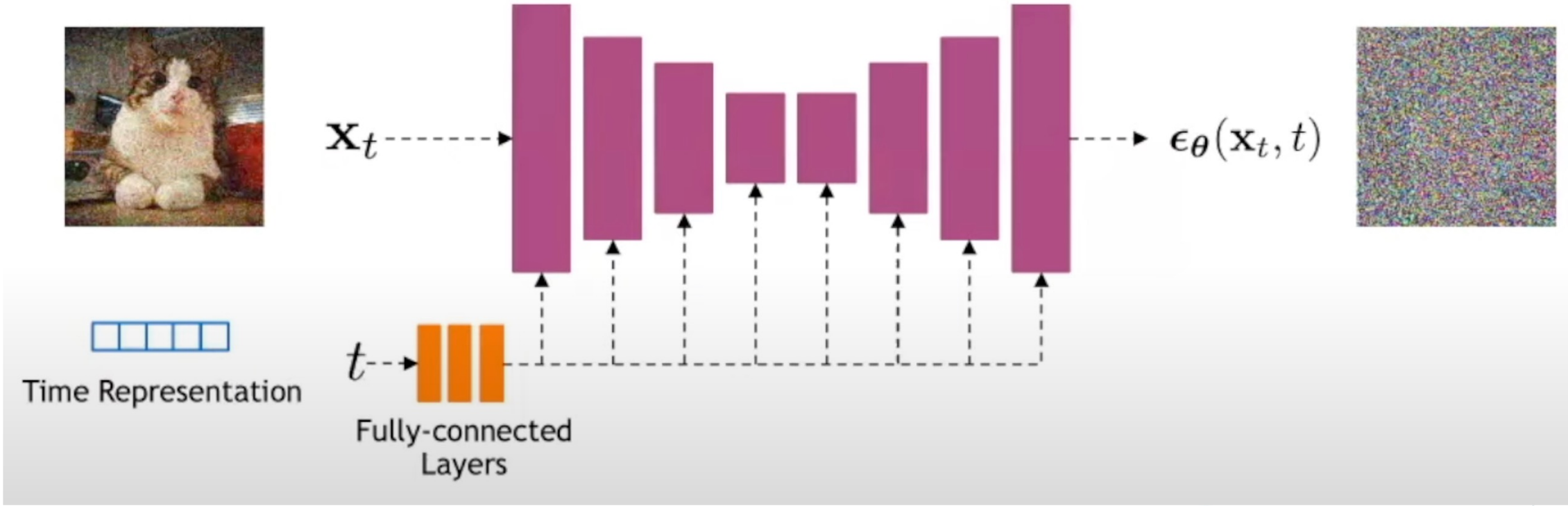
Need an architecture with identical output and input spatial dimensions – UNet is a natural choice!

Add:

- positional embeddings
- ResNet blocks
- ConvNet blocks
- attention modules
- group normalization
- different activations

for better performance.

Back to UNet with tricks



Linear vs Cosine Schedule

A linear noise schedule converts initial data to noise really quickly, making the reverse process harder for the model to learn.

A cosine-like function that is changing relatively gradually neat the endpoints works better (but it is arbitrary!)

Linear (top) vs Cosine (bottom)

Latent Diffusion models

Training models in the pixel space is **excessively computationally expensive** (can easily multiple days on multiple GPUs)

- Even image synthesis is very slow!
- Images are high dimensional → more things to model

Most “bits” of an image contribute to its perceptual characteristics since aggressively compressing it usually maintains its semantic and conceptual composition

- In layman’s terms, there are more bits for describing pixel-level details while less bits for describing “the meaning” within an image
- Generative models should learn the latter

- Can we separate these two components?

Latent Diffusion models

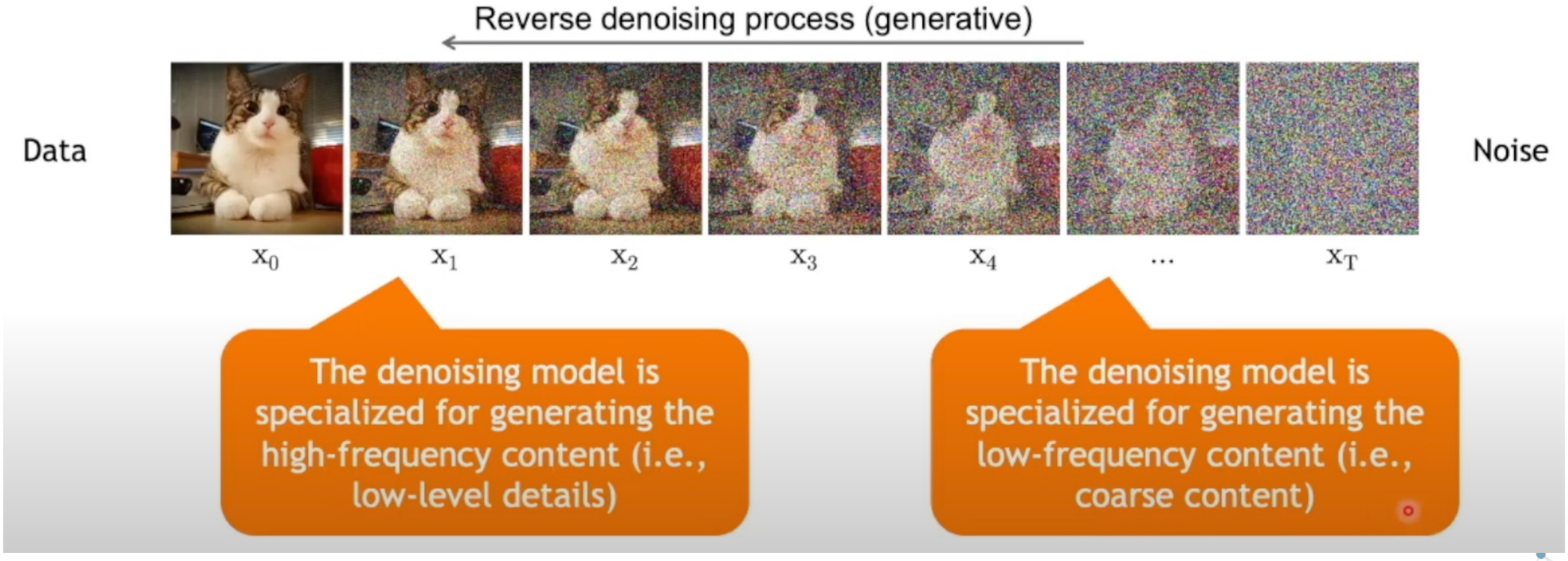
Latent Diffusion Models can be divided into two stages:

(1) Training perceptual compression models that strip away irrelevant high-level details and learn a latent space that is semantically equivalent to the high-level image pixel-space

(2) Performing a diffusion process *in this latent space*. There are several benefits to this:

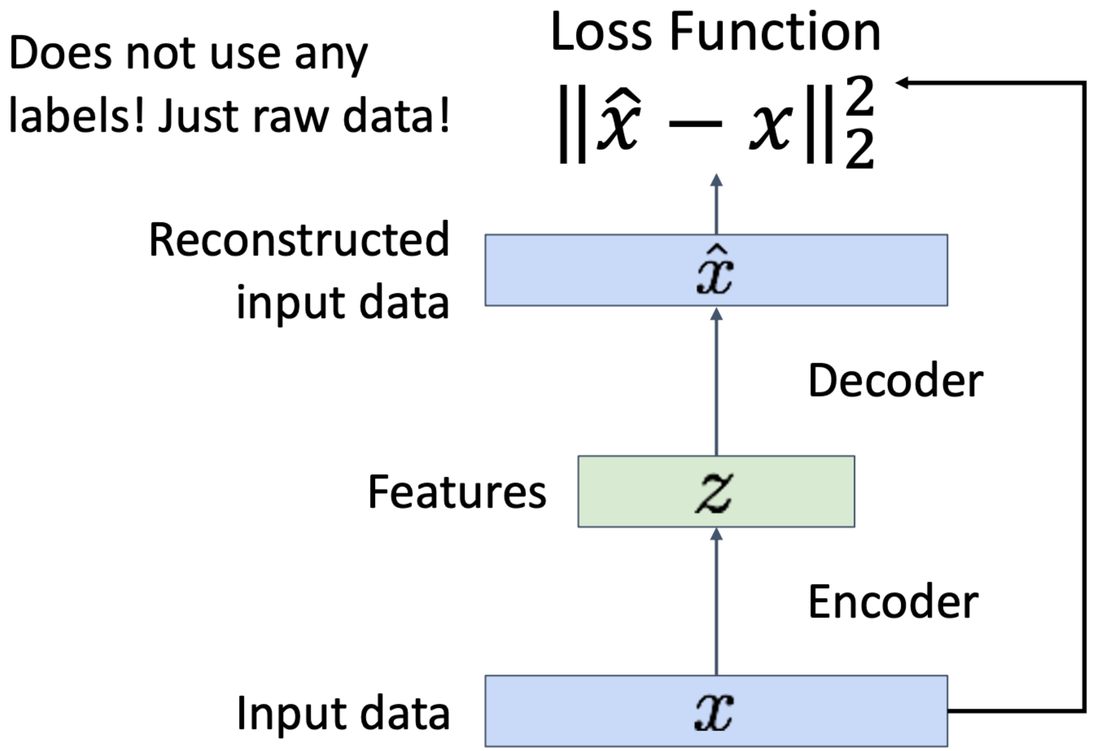
a. The diffusion process is only focusing on the **relevant semantic bits** of the data
b. Performing diffusion in a low dimensional space is **significantly more efficient**

Latent Diffusion models



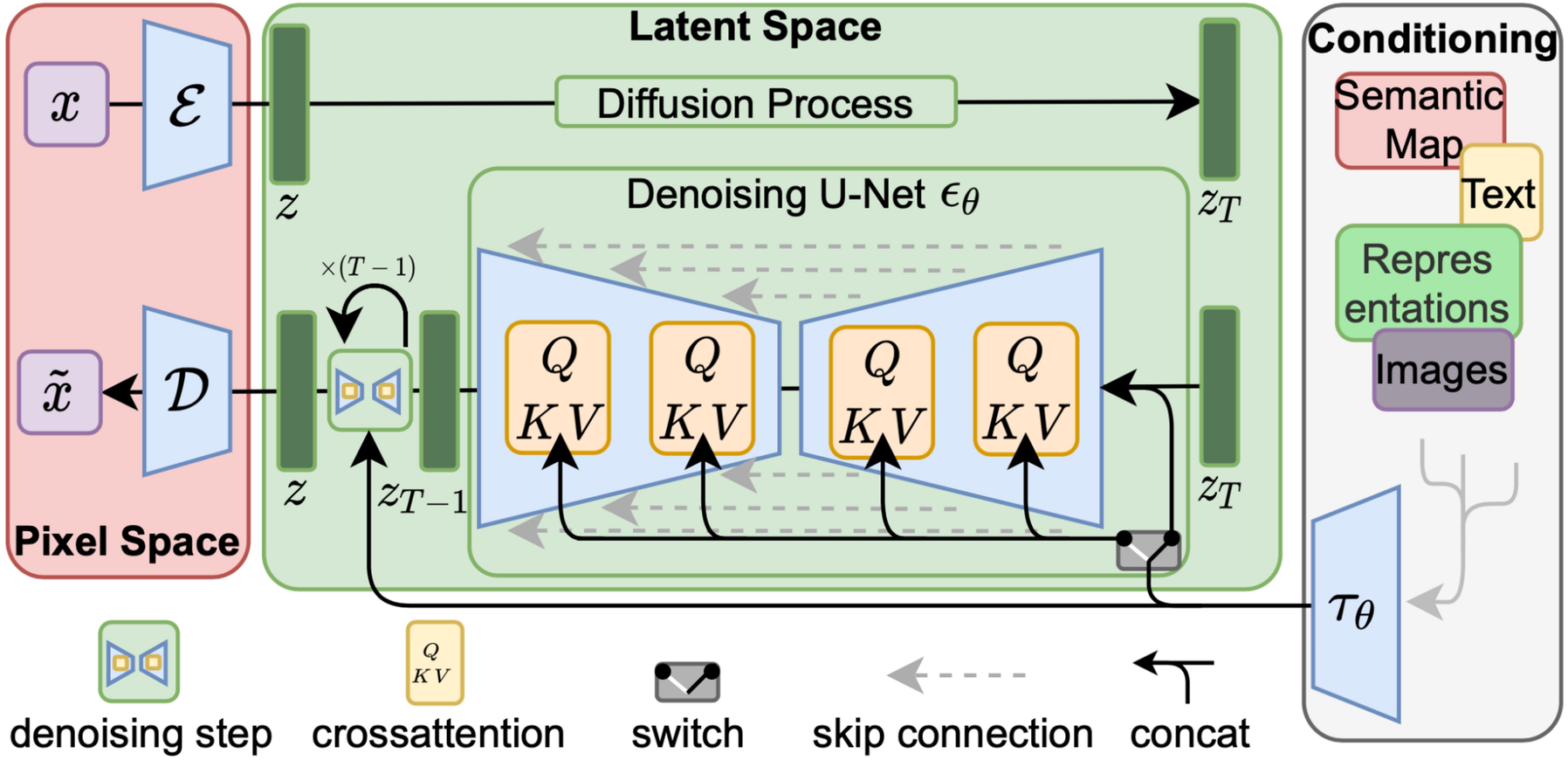
Reminder: Autoencoders

Loss: L2 distance between input and reconstructed data. Features need to be lower dimensional than data.

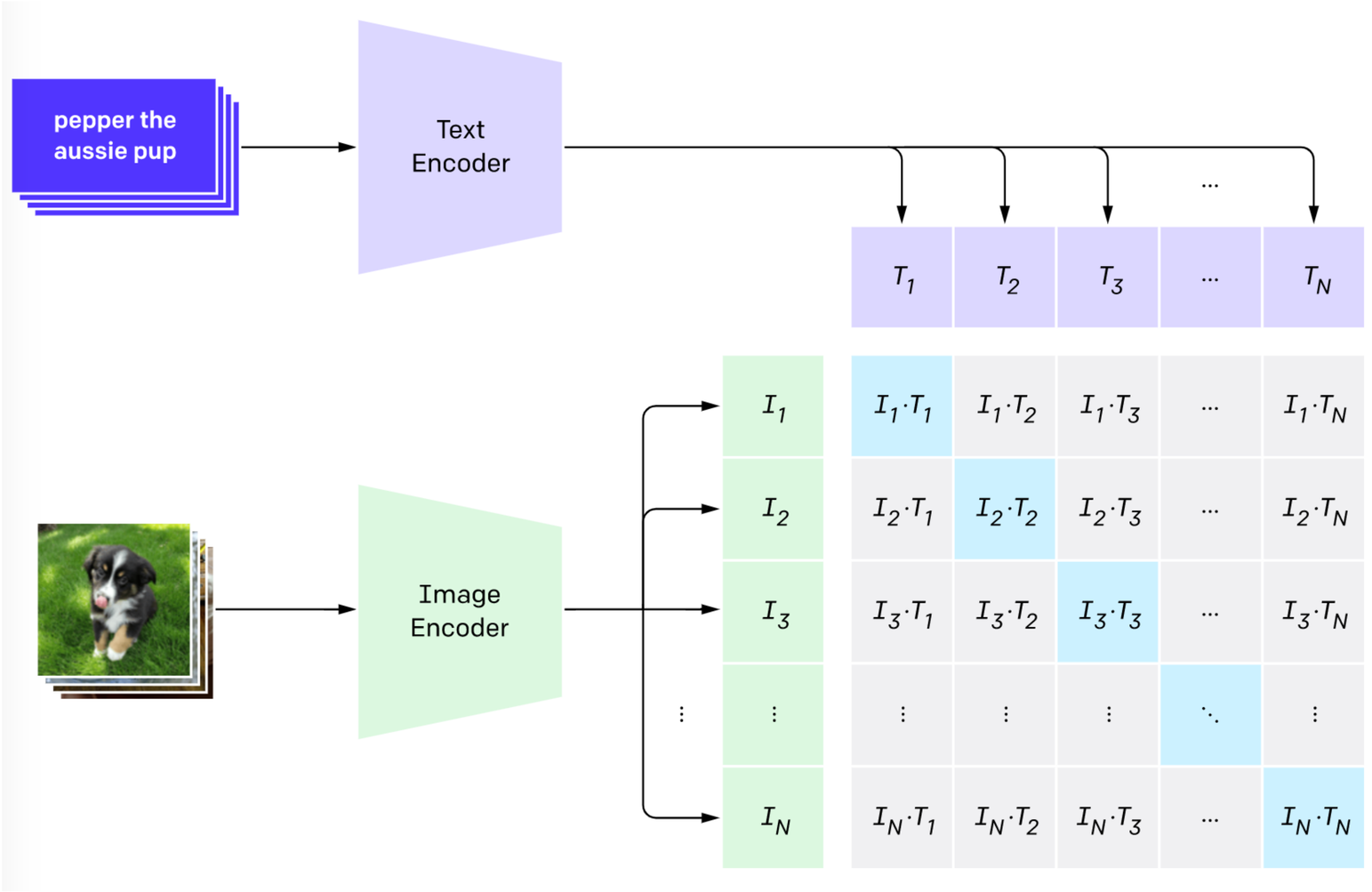


After training, throw away decoder and use encoder for a downstream task.

Latent Diffusion models



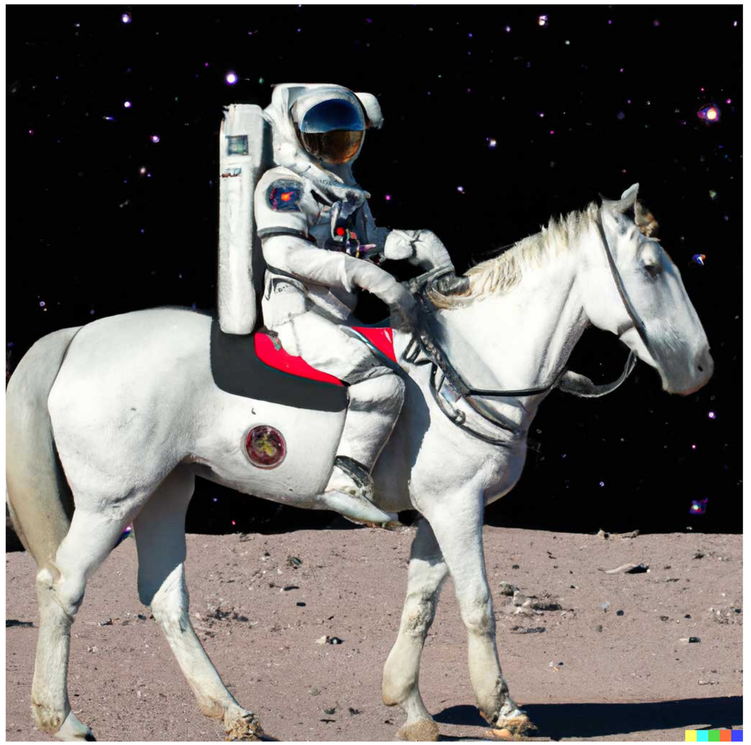
CLIP embeddings



Examples of recent Diffusion Models

DALLE 2 (Text-to-Image)

Teddy bears mixing sparkling chemicals as mad scientists



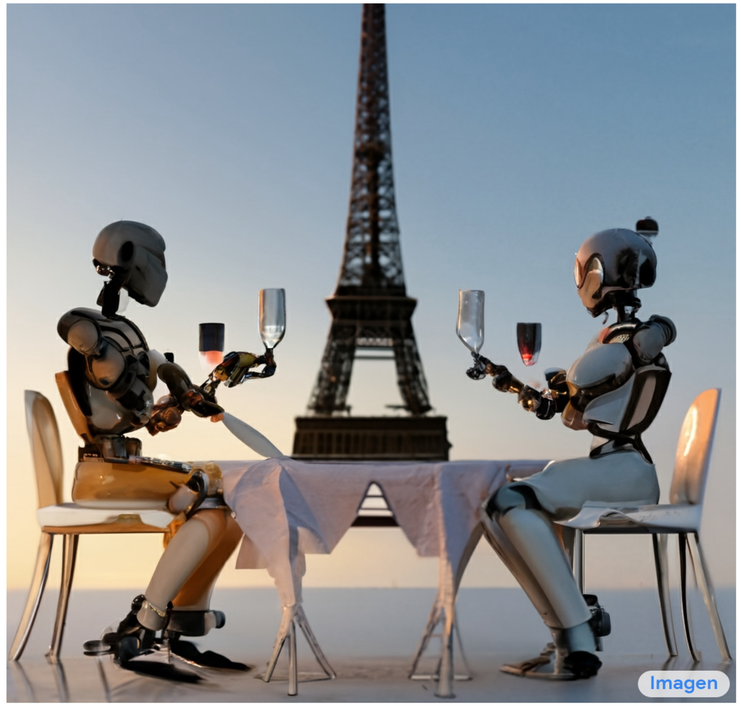
An astronaut riding a horse in a photorealistic style

A bowl of soup as a planet in the universe

Imagen (Text-to-Image)

A cute corgi lives in a house made of sushi

A majestic oil painting of a raccoon Queen wearing red French royal gown.



A robot couple fine-dining with the Eiffel Tower in the background

Make-A-Video (Text-to-Video)

An artist’s brush painting on a canvas close up

A young couple walking in heavy rain

Horse drinking water

Make-A-Video (Text-to-Video)

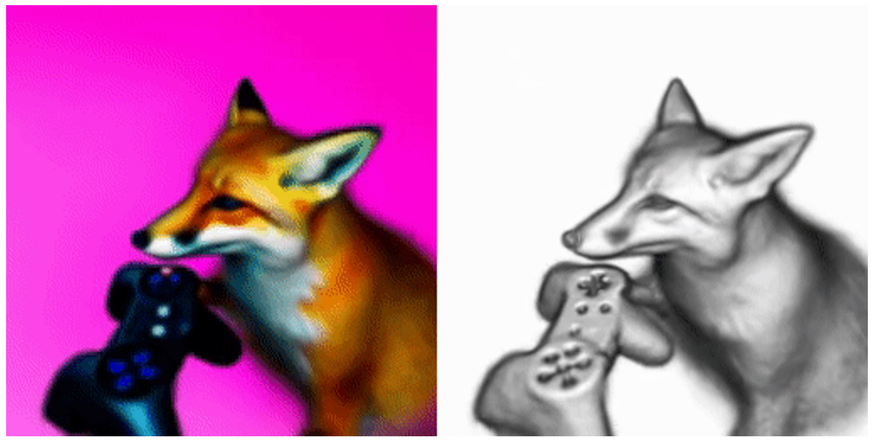
A confused grizzly bear in a calculus class

A golden retriever eating ice cream on a beautiful tropical beach at sunset, high resolution

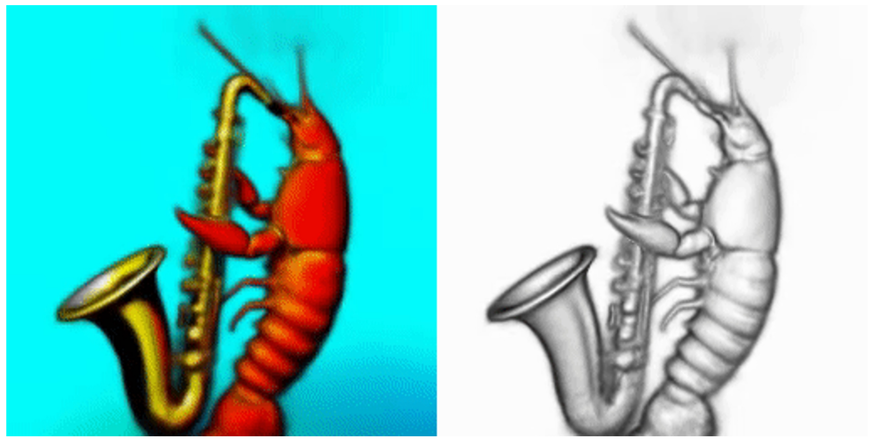
A panda playing on a swing set

Imagen Video (Text-to-Video)

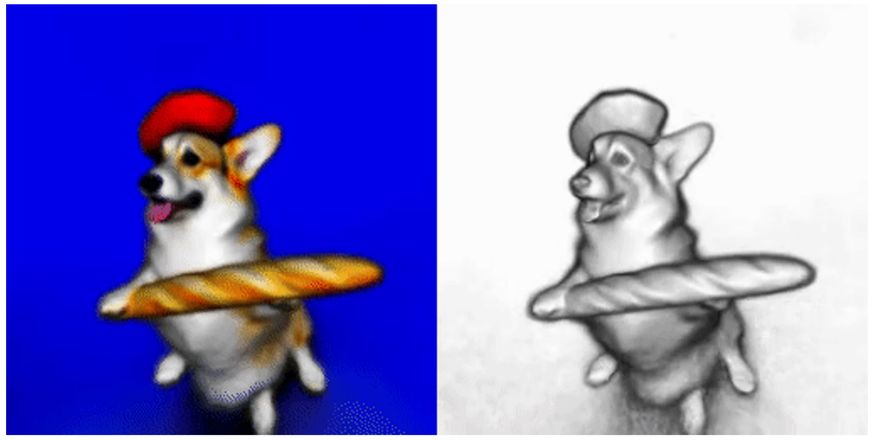
DreamFusion (Text-to-3D)



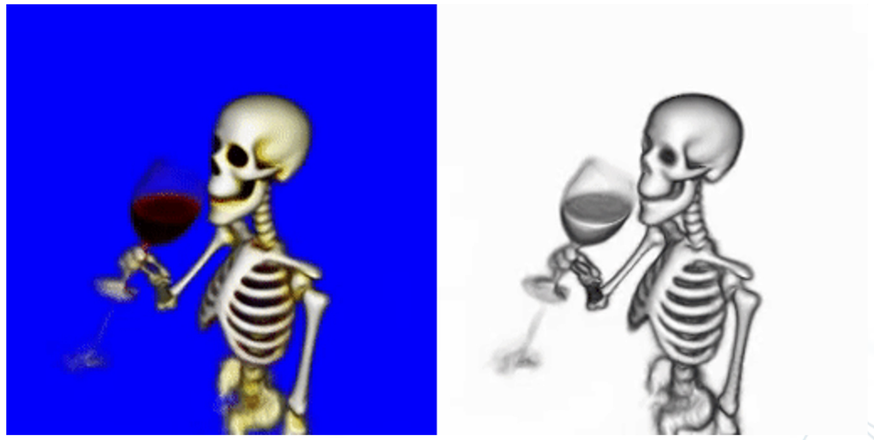
a fox holding a video game controller



a lobster playing the saxophone

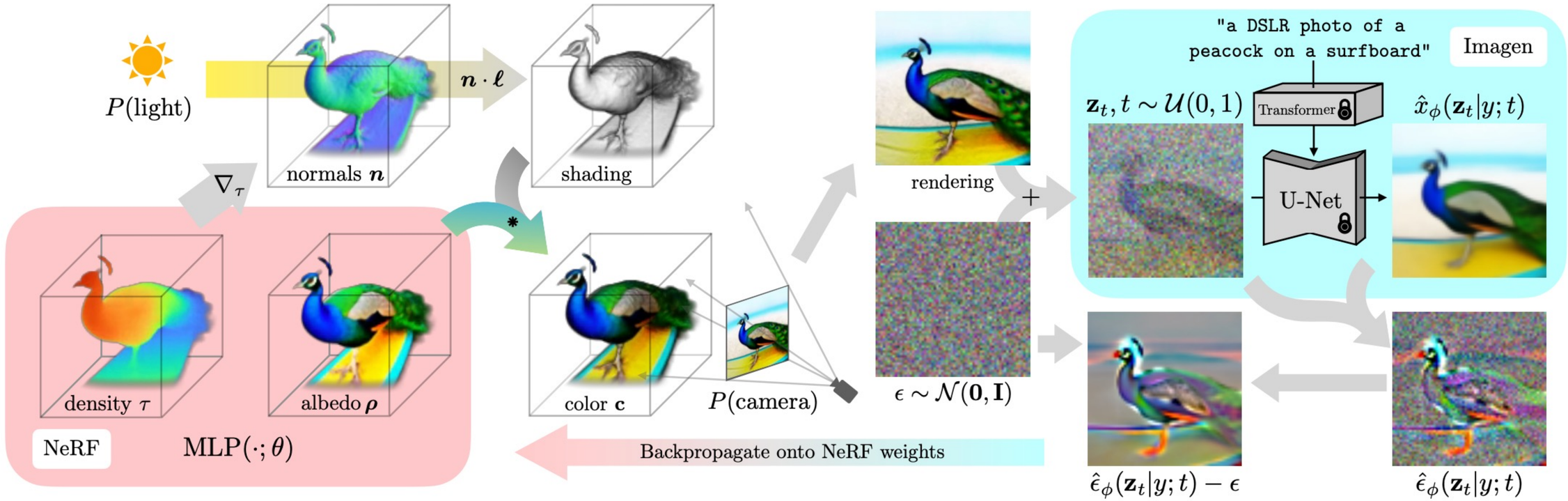


a corgi wearing a beret and holding a baguette, standing up on two hind legs



a human skeleton drinking a glass of red wine

DreamFusion (Text-to-3D)



Make-A-Video3D (Text-to-4D)

Summary

- Supervised vs unsupervised learning
- Discriminative vs generative models
- Diffusion models
- Diffusion models for images

Reading material

Main papers on diffusion (can be theory-heavy):

- Deep Unsupervised Learning using Nonequilibrium Thermodynamics: https://arxiv.org/pdf/1503.03585.pdf
- Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2006.11239.pdf
- Improved Denoising Diffusion Probabilistic Models: https://arxiv.org/pdf/2102.09672.pdf
- Diffusion Models Beat GANs on Image Synthesis: https://arxiv.org/pdf/2105.05233.pdf
- Classifier-free Diffusion Guidance: https://arxiv.org/pdf/2207.12598.pdf
- High Resolution Image Synthesis with Latent Diffusion Models: https://arxiv.org/pdf/2112.10752.pdf

Other resources:

- Lillian Weng's Blog: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/
- The Annotated Diffusion Model: https://huggingface.co/blog/annotated-diffusion
- The Illustrated Stable Diffusion: https://jalammar.github.io/illustrated-stable-diffusion/
- PyTorch implementation of the DDPM Unet: https://nn.labml.ai/diffusion/ddpm/unet.html
- Guidance: a cheat code for diffusion models: https://benanne.github.io/2022/05/26/guidance.html
- Understanding Diffusion Models: A Unified Perspective: https://arxiv.org/pdf/2208.11970.pdf

Questions?