Generative Models

CAS CS 585 Image and Video Computing

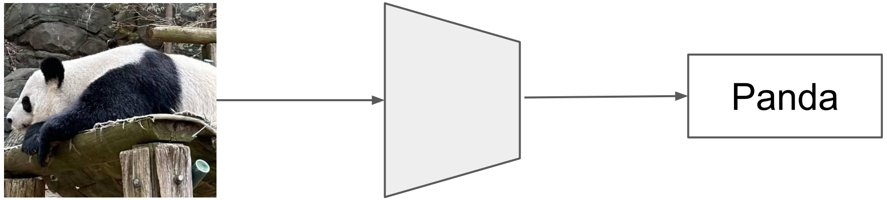
Hao Yu

April 2, 2024

BOSTON UNIVERSITY
Department of Computer Science

What are Generative Models?

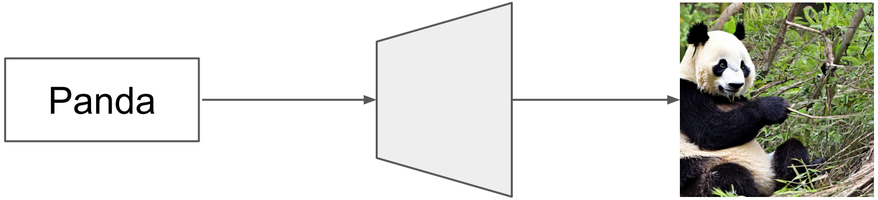
- Discriminative Models:



Input: image x

Output: label y

- Generative Models:



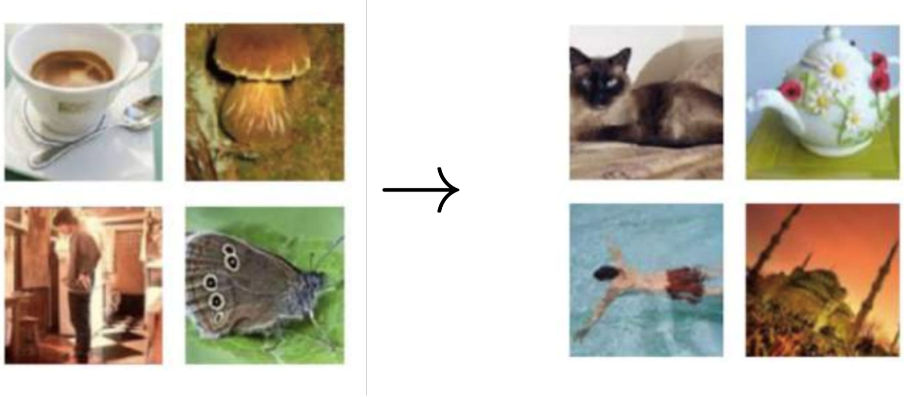
Input: label y

Output: image x

Image generated by Stable Diffusion

What are Generative Models?

Given training data from some distribution, learn a model that represents that distribution and can generate new samples from the same distribution.



Training data ~ $P_{data}(x)$ Generated ~ $P_{model}(x)$

Explicit v.s. Implicit Generative Models

- Explicit: explicitly define $P_{model}(x)$
  - VAE
- Implicit: learn a model that can sample from $P_{model}(x)$ without explicitly defining it.
  - GANs

Why Generative Models?

Realistic, high-quality samples, super-resolution, and image inpainting, etc.

Why Generative Models?

Text-to-Video

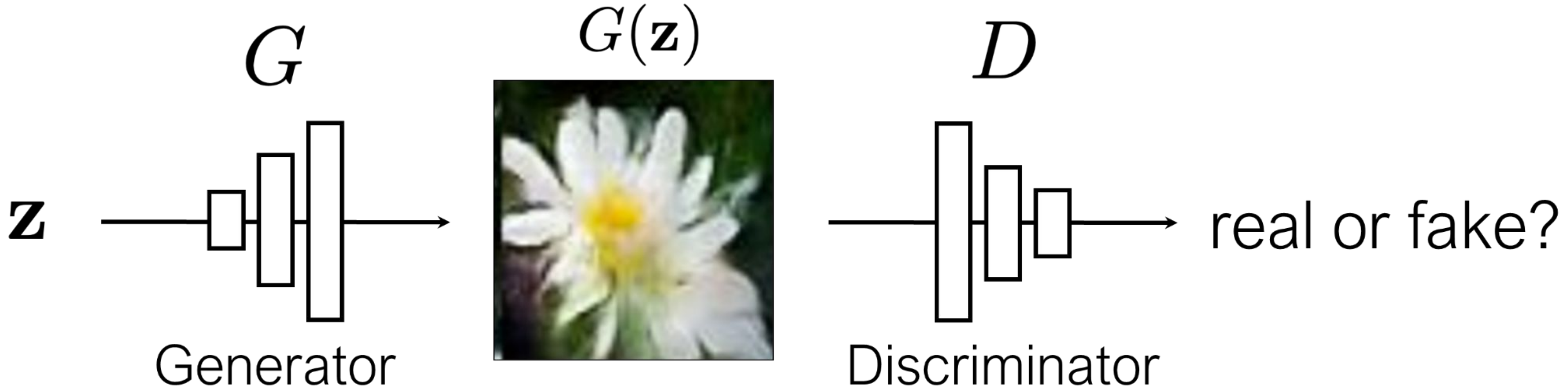
Prompt: A cartoon kangaroo disco dances.

Sora, 2024

Generative Models

- Generative adversarial networks (GANs)
- Denoising diffusion models

Generative Adversarial Networks (GANs)

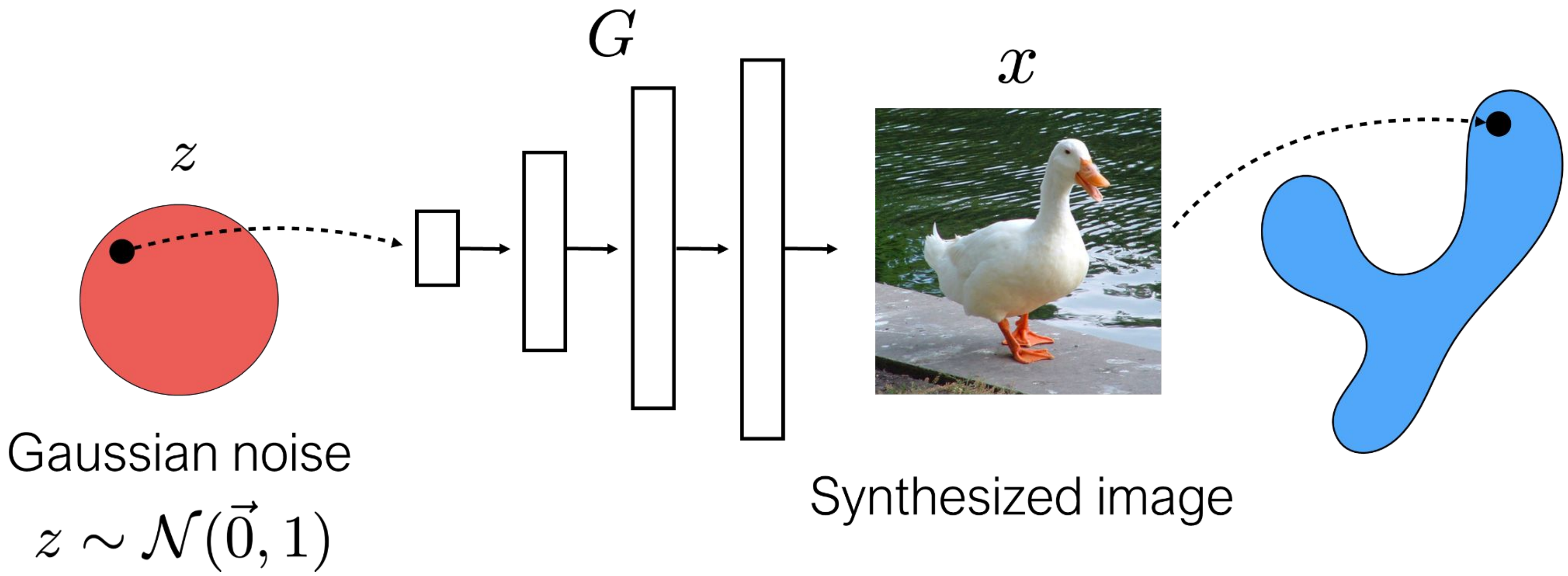


G tries to synthesize fake images that fool **D**

D tries to identify the fakes

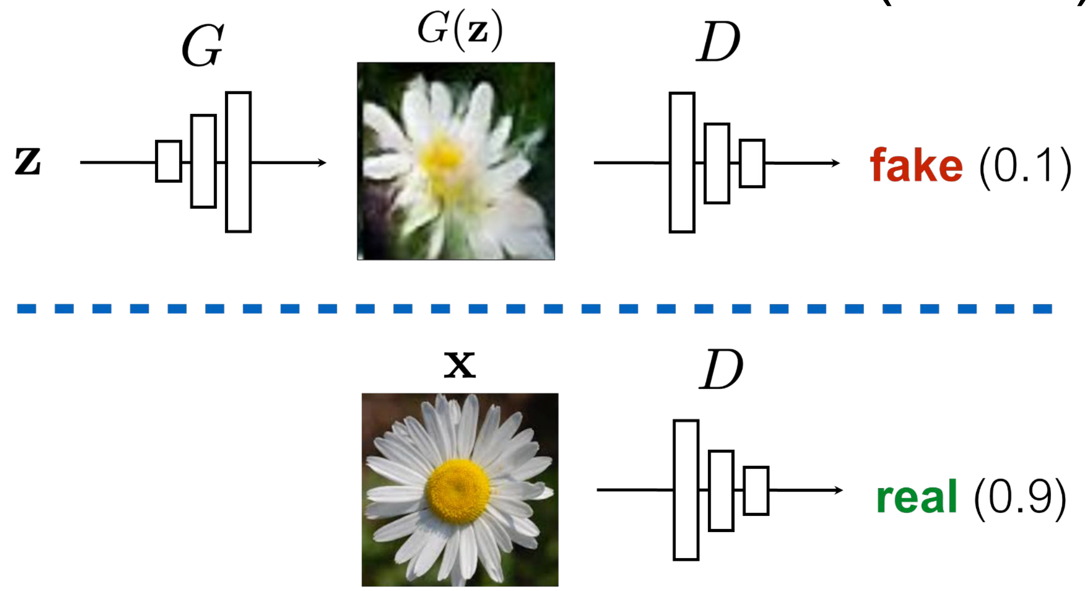
[Goodfellow et al., 2014]

Generative Adversarial Networks (GANs)



Slides Credit: MIT 6.869/6.189 Advances in Computer Vision, Bill Freeman and Phillip Isola

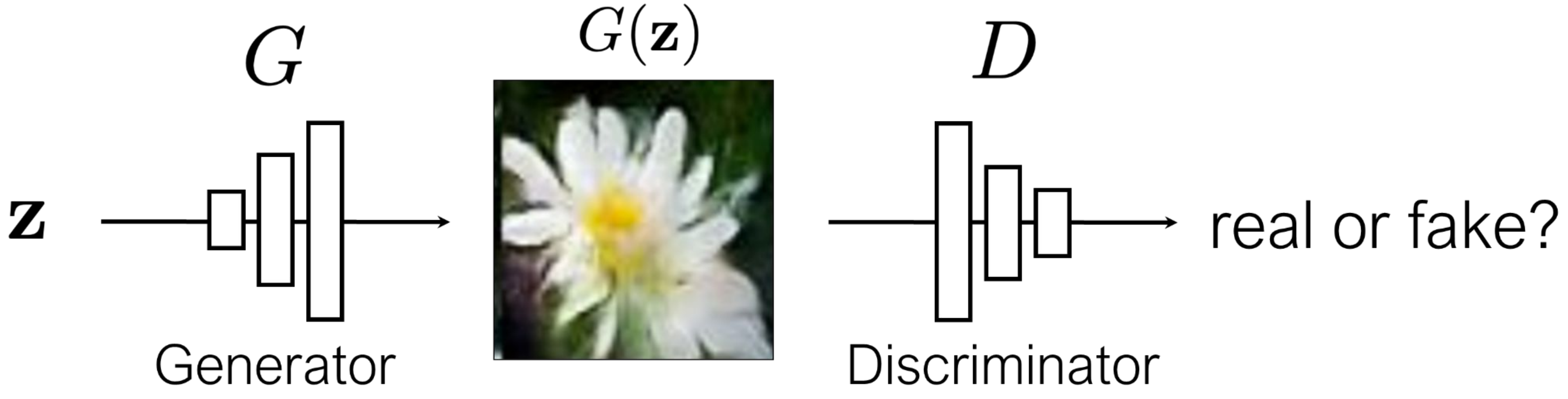
Generative Adversarial Networks (GANs)



$$\arg\max_{D} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

[Goodfellow et al., 2014]

Generative Adversarial Networks (GANs)

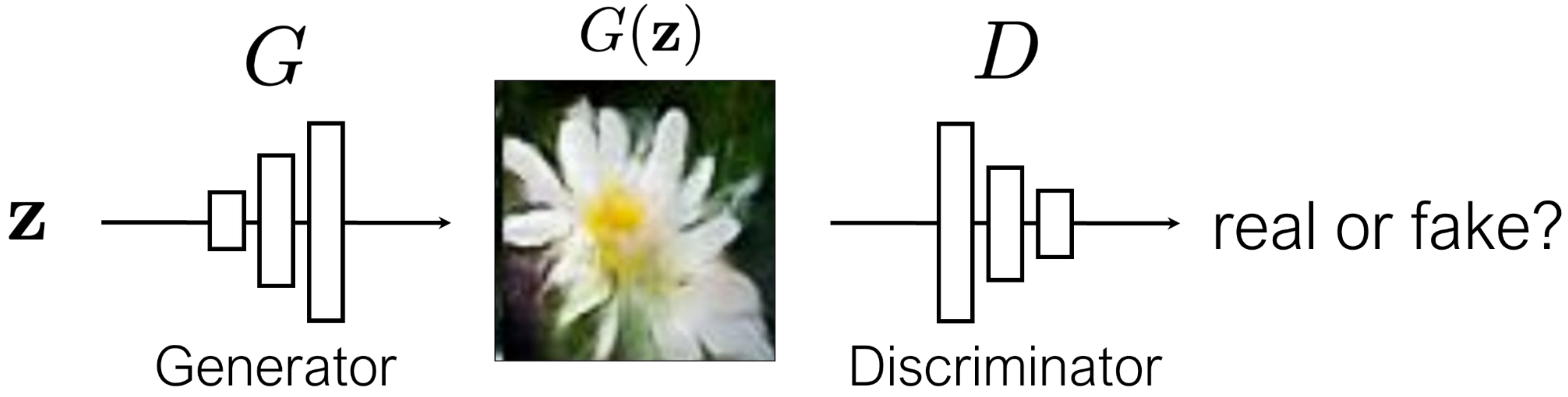


G tries to synthesize fake images that ***fool*** **D**:

$$\arg\min_{G} \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

[Goodfellow et al., 2014]

Generative Adversarial Networks (GANs)

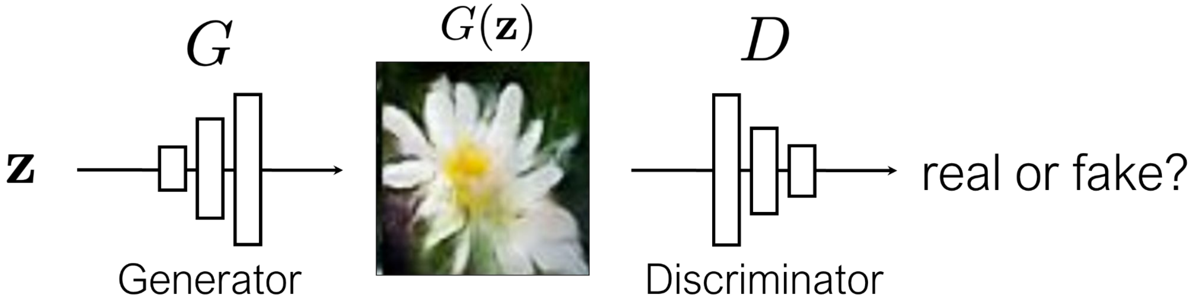


G tries to synthesize fake images that ***fool*** the ***best*** **D**:

$$\arg\min_{G}\max_{D}\mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})]+\mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1-D(G(\boldsymbol{z})))]$$

[Goodfellow et al., 2014]

Training



G tries to synthesize fake images that fool **D**

D tries to identify the fakes

Training: alternate between training D and G with backprop.

[Goodfellow et al., 2014]

Common Issues

- Mode collapse
  - A situation where the generator produces limited or repetitive outputs, failing to capture the full diversity of the training data distribution.
- Adversarial training is unstable
- Saturation problem (weak gradients)

$$\arg \min_G \max_D \mathbb{E}_{\boldsymbol{x} \sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))]$$

  - If G is poor, D can easily distinguish between real and generated samples. The prediction of D is close to 0, and the generator's cost is close to 0.
  - A better cost function:

$$\arg \max_G \mathbb{E}_{\boldsymbol{z} \sim p_{\boldsymbol{z}}(\boldsymbol{z})} \log D(G(\mathbf{z}))$$

[Goodfellow et al., 2014]

GANs are implicit generative models

$p(\mathbf{x})$ ⟵ "generative model" of the data $\mathbf{x}$

Noise distribution $\quad G(\mathbf{z}) \quad$ Data distribution

$\mathbf{z} \sim \mathcal{N}(0,1)$ ⟶ GAN ⟶ $\mathbf{x} \sim p(\mathbf{x})$

$G(\mathbf{z}) \sim p(\mathbf{x})$ ⟵ Samples from a perfectly optimized, sufficiently expressive GAN are samples from the data distribution

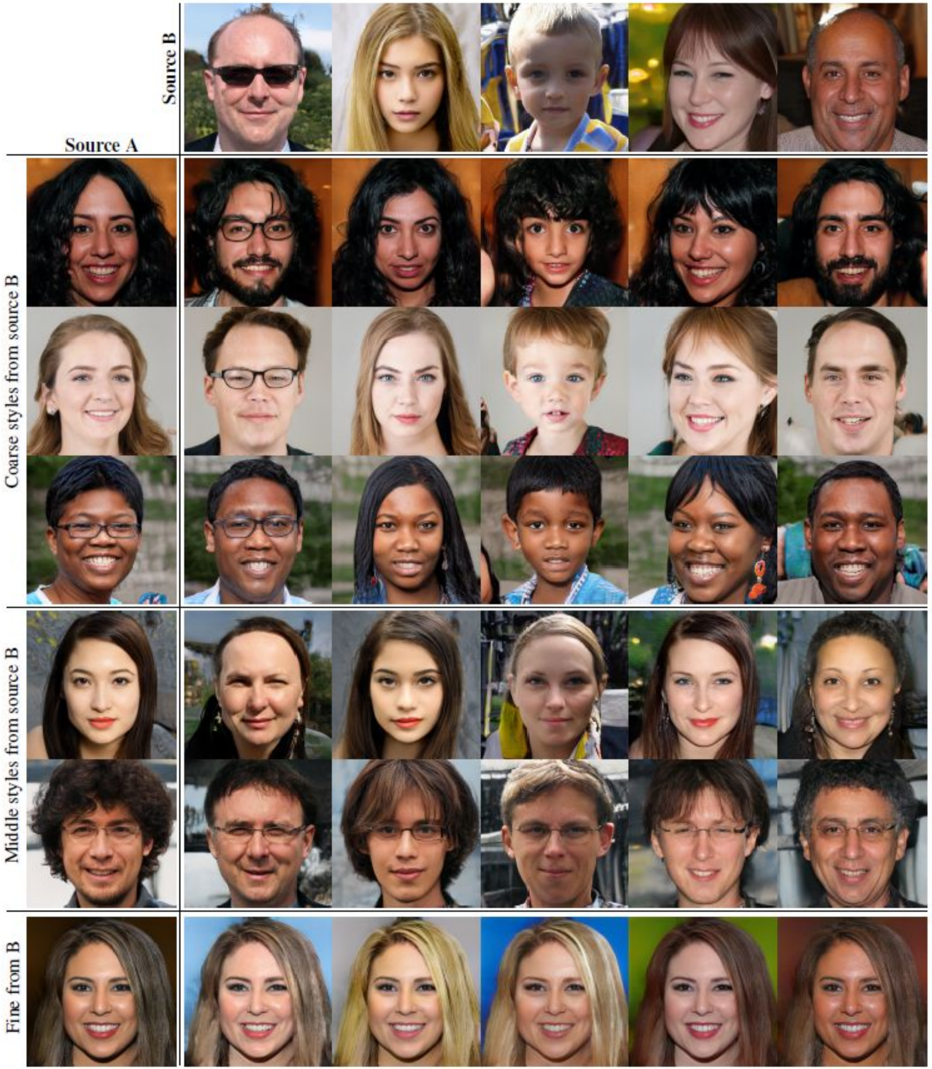
Generative Adversarial Networks (GANs)

Kerras, et al., 2018

Generative Adversarial Networks (GANs)

Kerras, et al., 2018

StyleGAN

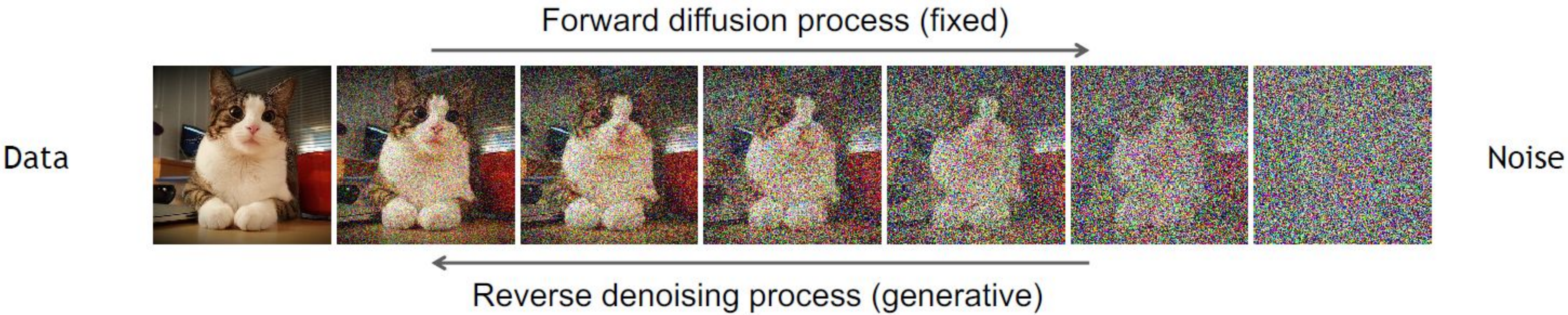


[Kerras, et al., 2019](#)

Denoising Diffusion Models

Denoising diffusion models consist of two processes:

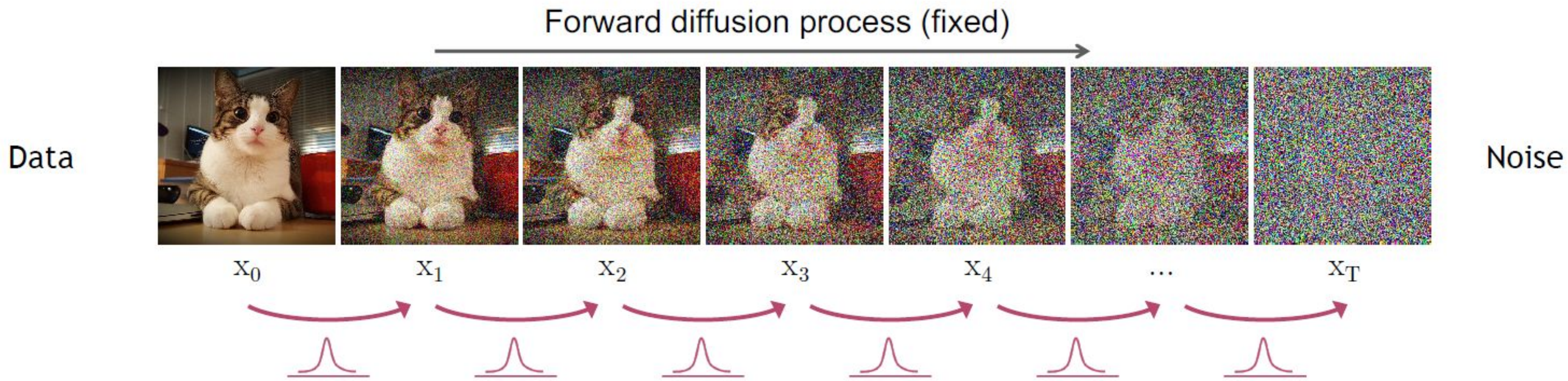
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising



Ho, et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021
Slides Credit: Arash Vahdat, Karsten Kreis, and Ruiqi Gao, Denoising Diffusion-based Generative Modeling: Foundations and Applications, CVPR 2022 Tutorial

Forward Diffusion Process

The formal definition of the forward process in T steps:



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

Slides Credit: Arash Vahdat, Karsten Kreis, and Ruiqi Gao, Denoising Diffusion-based Generative Modeling: Foundations and Applications, CVPR 2022 Tutorial

Reparameterization Trick

Define

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

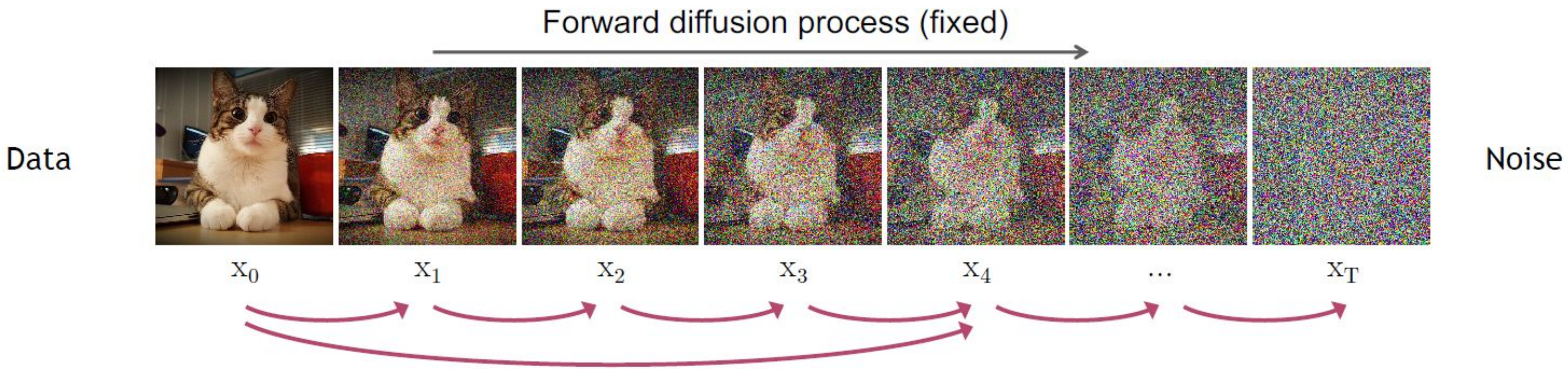
Then

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

$$\begin{aligned}
\mathbf{x}_t &= \sqrt{1-\beta_t}\mathbf{x}_{t-1} + \sqrt{\beta_t}\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}) \\
&= \sqrt{\alpha_t}\mathbf{x}_{t-1} + \sqrt{1-\alpha_t}\epsilon \\
&= \sqrt{\alpha_t \alpha_{t-1}}\mathbf{x}_{t-2} + \sqrt{1-\alpha_t \alpha_{t-1}}\epsilon \\
&= \dots \\
&= \sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon
\end{aligned}$$

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

Forward Diffusion Process



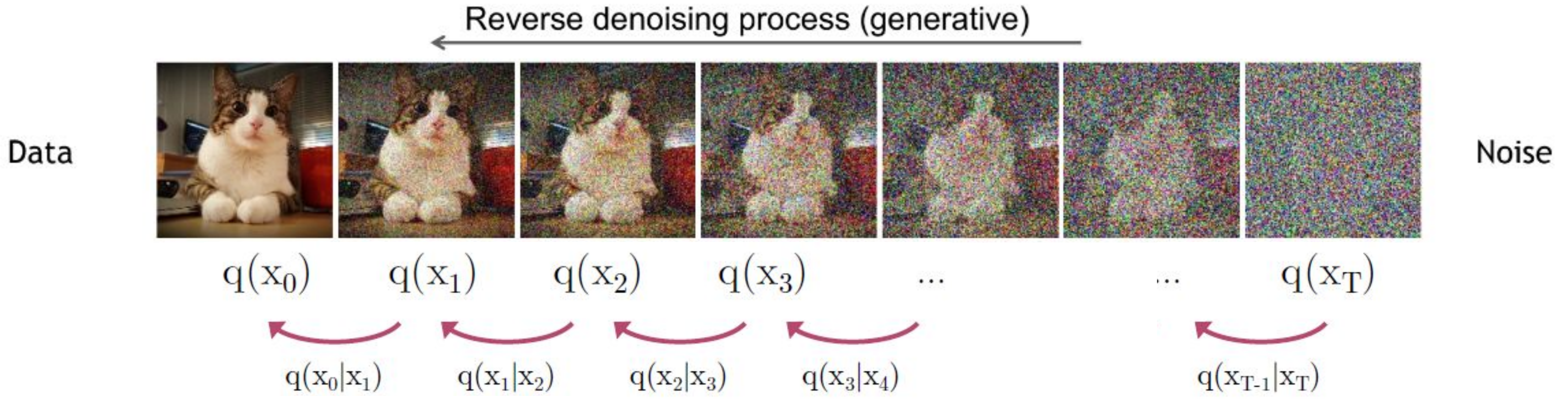
Define $\bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$ ➡ $q(\mathbf{x}_t|\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_t; \sqrt{\bar{\alpha}_t}\mathbf{x}_0, (1-\bar{\alpha}_t)\mathbf{I}))$ (Diffusion Kernel)

For sampling: $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon$ where $\epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \rightarrow 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Slides Credit: [Arash Vahdat, Karsten Kreis, and Ruiqi Gao, Denoising Diffusion-based Generative Modeling: Foundations and Applications, CVPR 2022 Tutorial]

Reverse Denoising Process



Sample $\mathbf{x}_T \sim \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$

Iteratively sample $\mathbf{x}_{t-1} \sim \underbrace{q(\mathbf{x}_{t-1}|\mathbf{x}_t)}_{\text{True Denoising Dist.}}$

$$\underbrace{q(\mathbf{x}_t)}_{\substack{\text{Diffused} \\ \text{data dist.}}} = \int \underbrace{q(\mathbf{x}_0, \mathbf{x}_t)}_{\substack{\text{Joint} \\ \text{dist.}}} \, d\mathbf{x}_0 = \int \underbrace{q(\mathbf{x}_0)}_{\substack{\text{Input} \\ \text{data dist.}}} \, \underbrace{q(\mathbf{x}_t|\mathbf{x}_0)}_{\substack{\text{Diffusion} \\ \text{kernel}}} \, d\mathbf{x}_0$$

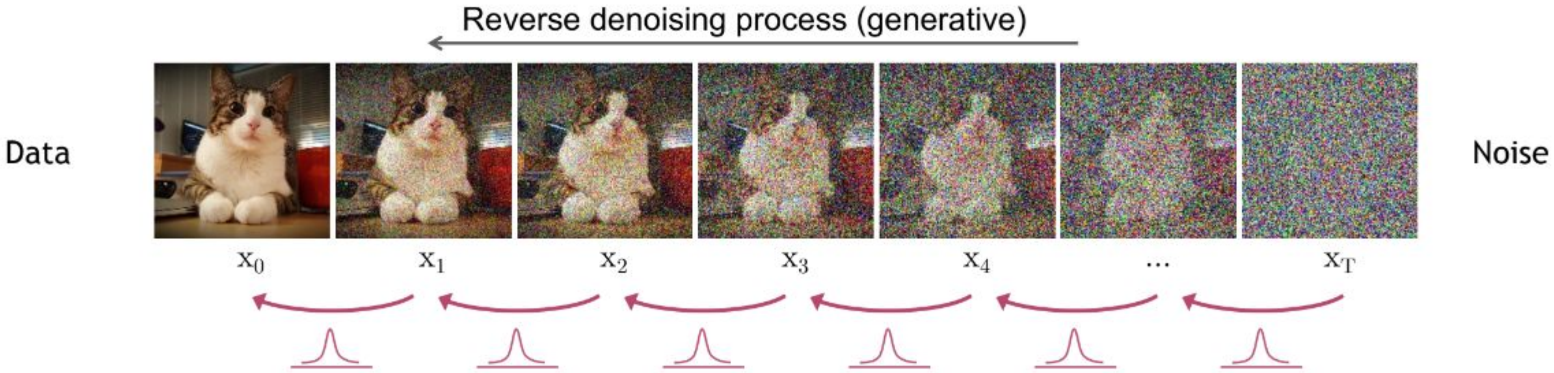
In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

Can we approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$? Yes, we can use a Normal distribution if β_t is small in each forward diffusion step.

Slides Credit: [Arash Vahdat, Karsten Kreis, and Ruiqi Gao, Denoising Diffusion-based Generative Modeling: Foundations and Applications, CVPR 2022 Tutorial]

Reverse Denoising Process

Formal definition of reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \boldsymbol{\mu}_\theta(\mathbf{x}_t, t), \boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t))$$

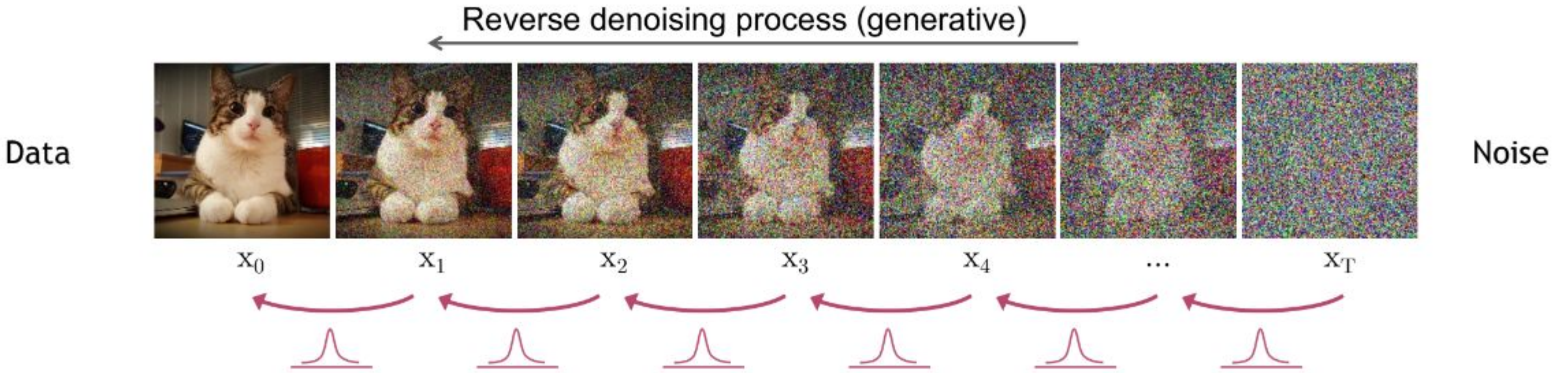
$$\Rightarrow \quad p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Trainable network
(U-net, Denoising Autoencoder)

$$\boldsymbol{\Sigma}_\theta(\mathbf{x}_t, t) = \sigma_t^2 \mathbf{I}$$

Reverse Denoising Process

Formal definition of reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}, \sigma_t^2\mathbf{I})$$

Trainable network
(U-net, Denoising Autoencoder)

➡ $$p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

23

Slides Credit: Arash Vahdat, Karsten Kreis, and Ruiqi Gao, Denoising Diffusion-based Generative Modeling: Foundations and Applications, CVPR 2022 Tutorial

Training Objective

For training, we use a variational upper bound on negative log likelihood $\mathbb{E}\left[-\log p_\theta(\mathbf{x}_0)\right]$

We represent the mean of the denoising model using a noise-prediction network:

$$\mu_\theta(\mathbf{x}_t, t) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon_\theta(\mathbf{x}_t, t)\right)$$

With this parameterization and further simplification, the final objective is:

$$L_{\text{simple}} = \mathbb{E}_{\mathbf{x}_0 \sim q(\mathbf{x}_0), \epsilon \sim \mathcal{N}(\mathbf{0}, \mathbf{I}), t \sim \mathcal{U}(1,T)}\left[||\epsilon - \epsilon_\theta(\underbrace{\sqrt{\bar{\alpha}_t}\,\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\,\epsilon}_{\mathbf{x}_t}, t)||^2\right]$$

More details in Ho, et al., 2020

Denoising Diffusion Models

Algorithm 1 Training

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

Algorithm 2 Sampling

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

Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$

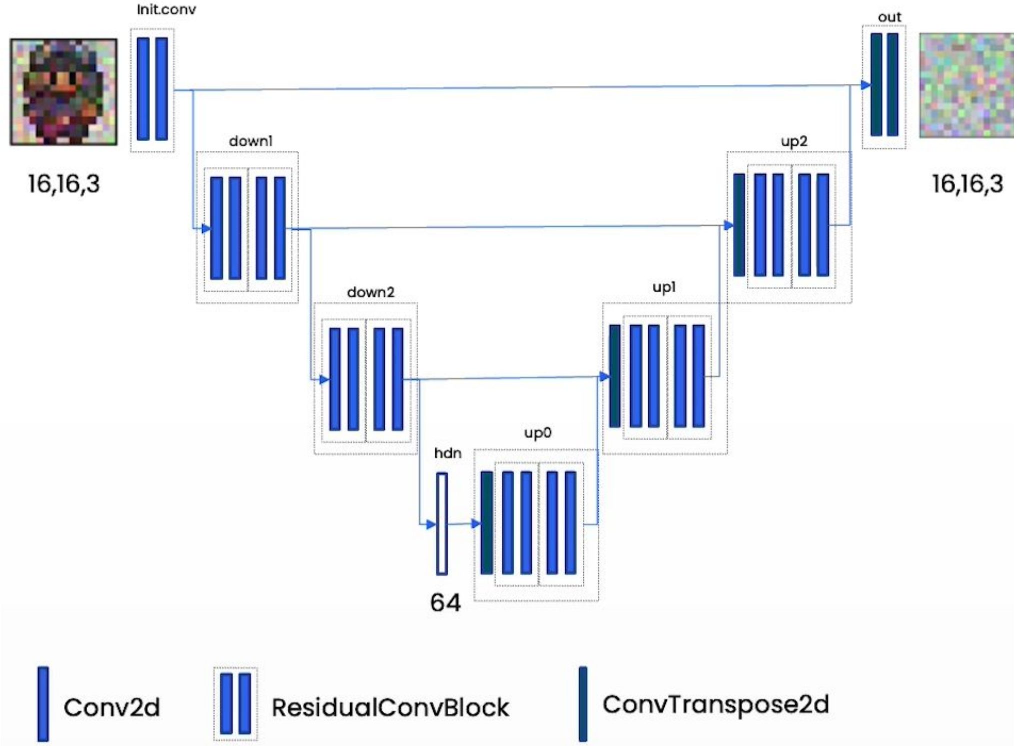
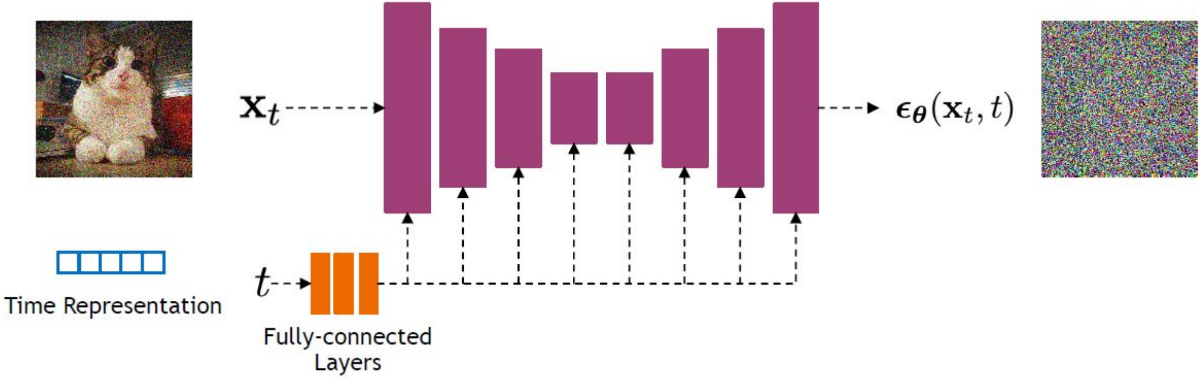


Image Credit: DeepLearning.AI

Network Architectures

Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\epsilon_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

Slides Credit: Arash Vahdat, Karsten Kreis, and Ruiqi Gao, Denoising Diffusion-based Generative Modeling: Foundations and Applications, CVPR 2022 Tutorial

Network Architectures

U-Net can take in more information in the form of embeddings.

Context embedding: relating to controlling the generation, e.g., text description.

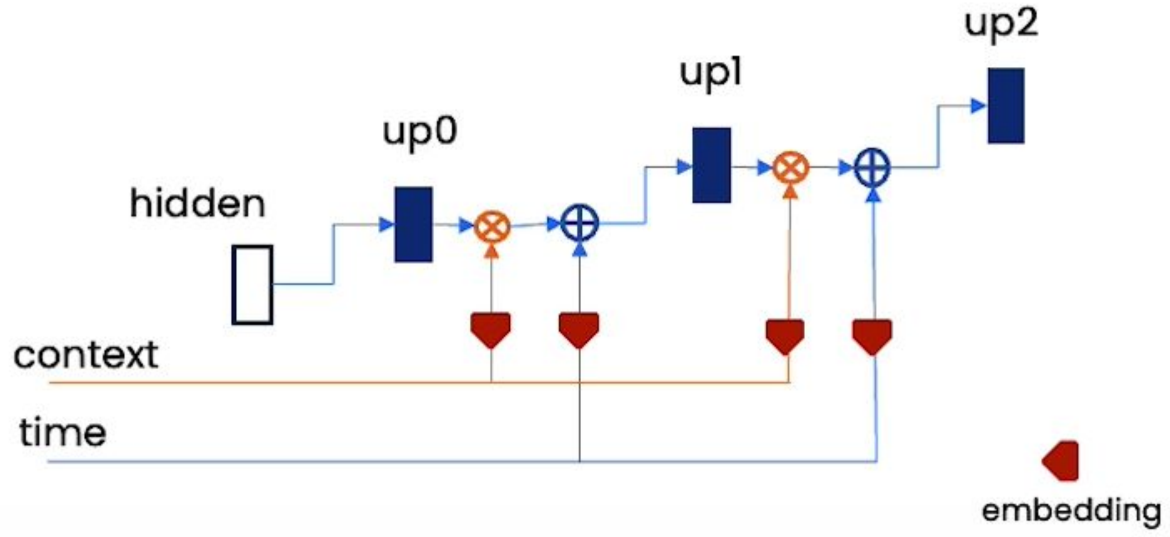
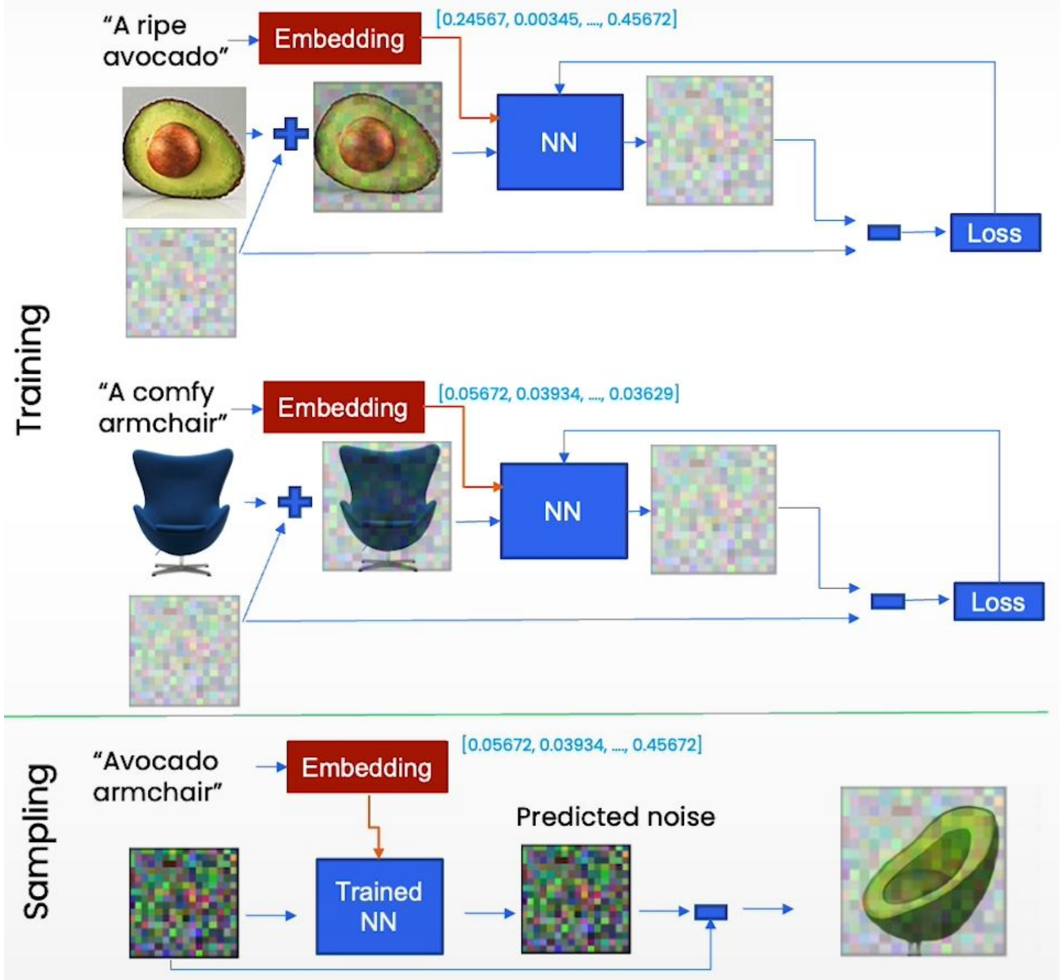
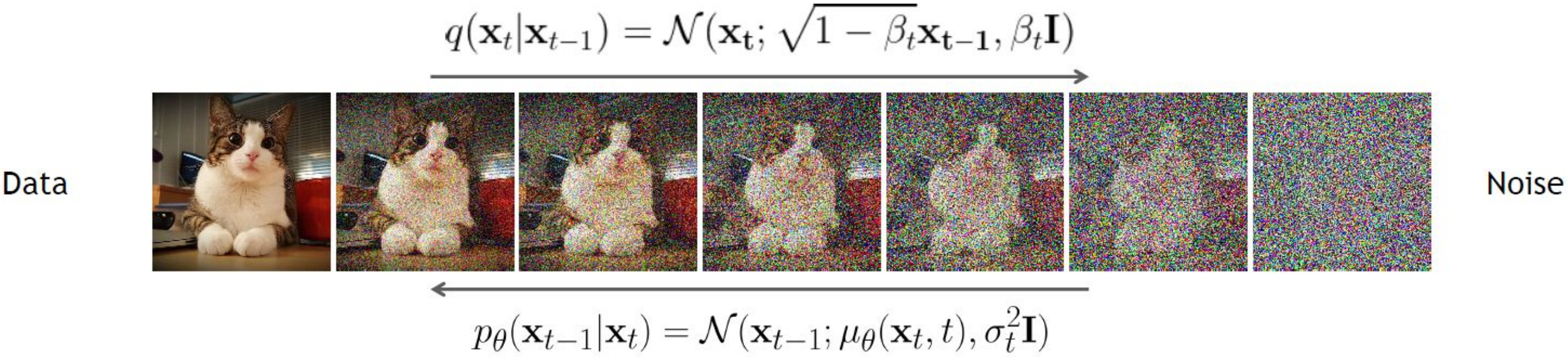


Image Credit: DeepLearning.AI

Adding Context



Noise Schedule



Above, β_t and σ_t^2 control the variance of the forward diffusion and reverse denoising processes respectively.

Often a linear schedule is used for β_t, and σ_t^2 is set equal to β_t.

Kingma et al. NeurIPS 2022 introduce a new parameterization of diffusion models using signal-to-noise ratio (SNR), and show how to learn the noise schedule by minimizing the variance of the training objective.

We can also train while training the diffusion model by minimizing the variational bound (Improved DPM by Nichol and Dhariwal ICML 2021) or after training the diffusion model (Analytic-DPM by Bao et al. ICLR 2022).

Classifier-free Guidance

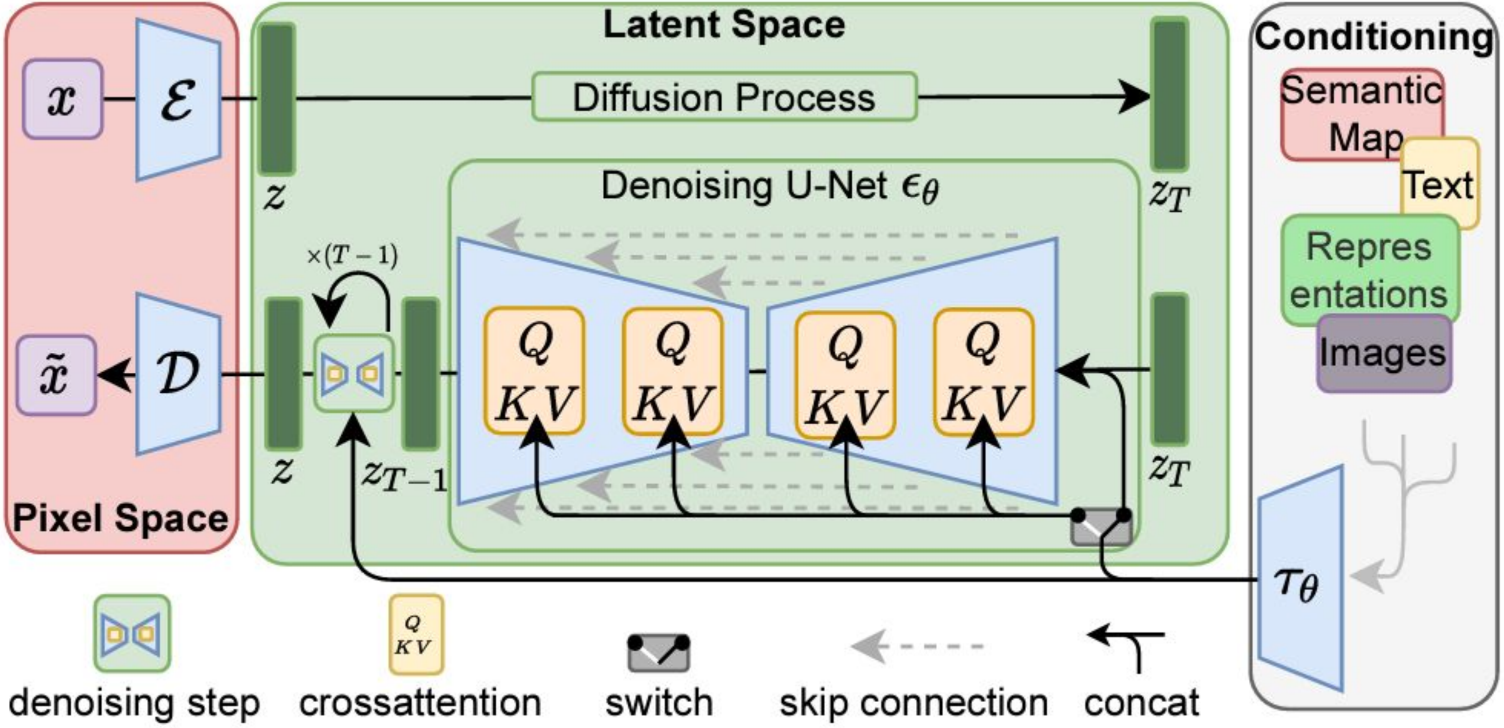
- trade off sample diversity and sample fidelity in conditional diffusion models
- jointly train a conditional and an unconditional diffusion model

$$\tilde{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) = (1+w)\epsilon_\theta(\mathbf{z}_\lambda, \mathbf{c}) - w\epsilon_\theta(\mathbf{z}_\lambda)$$

Non-guided   Classifier-free guided

Ho, et al., 2022

Latent Diffusion Model

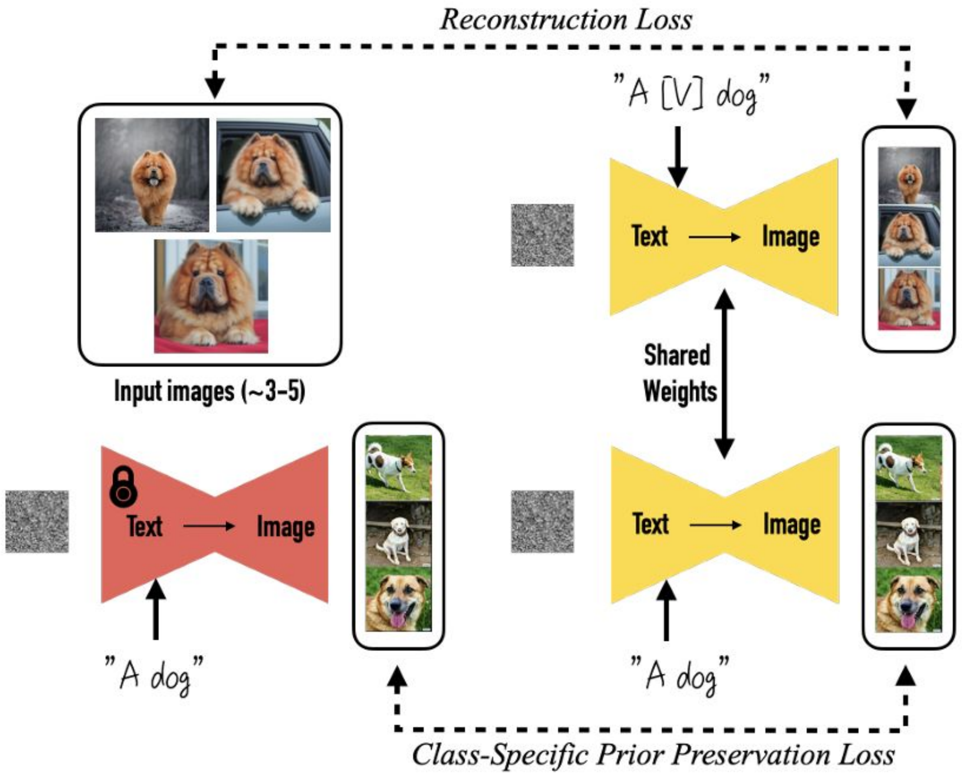


Stable Diffusion

DreamBooth

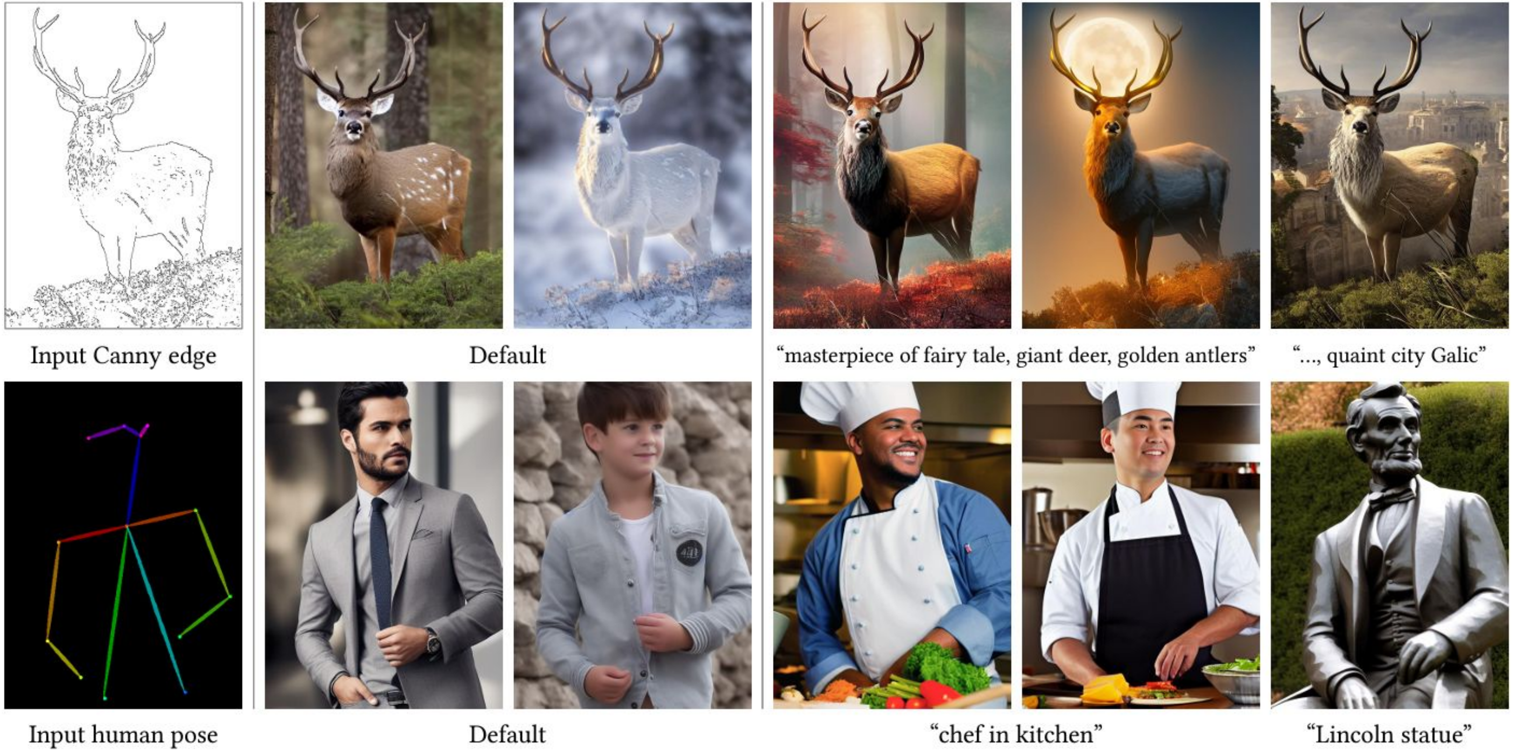
[Ruiz, et al., 2023](#)

DreamBooth



Reconstruction Loss
"A [V] dog"
Text → Image
Input images (~3-5)
Shared Weights
Text → Image
Text → Image
"A dog"
"A dog"
Class-Specific Prior Preservation Loss

ControlNet


Input Canny edge
Default
"masterpiece of fairy tale, giant deer, golden antlers"
"..., quaint city Galic"
Input human pose
Default
"chef in kitchen"
"Lincoln statue"

Learning Objectives

- Generative Models v.s Discriminative Models
- Explicit v.s. Implicit Generative Models
- Formation of GANs
- Common issues in GANs
- Forward and reverse process in diffusion models
- Training and sampling in diffusion models
- UNet in diffusion models
- Conditional diffusion models
- Applications of generative models