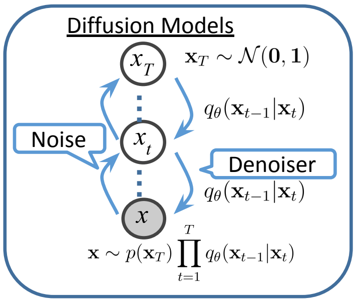
Outline

- Generative Models Basics
- Autoregressive Models
- Autoencoder and Variational Autoencoder
- Generative Adversarial Network
- **Diffusion Models**

Denoising Diffusion Models



Forward diffusion process (fixed)

Data  Noise

Reverse denoising process (generative)

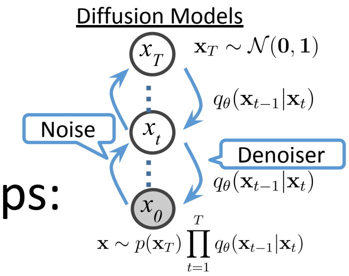
Denoising Diffusion Models

Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Slide credit: Kreis, Gao, Vahdat, CVPR 2022 Tutorial

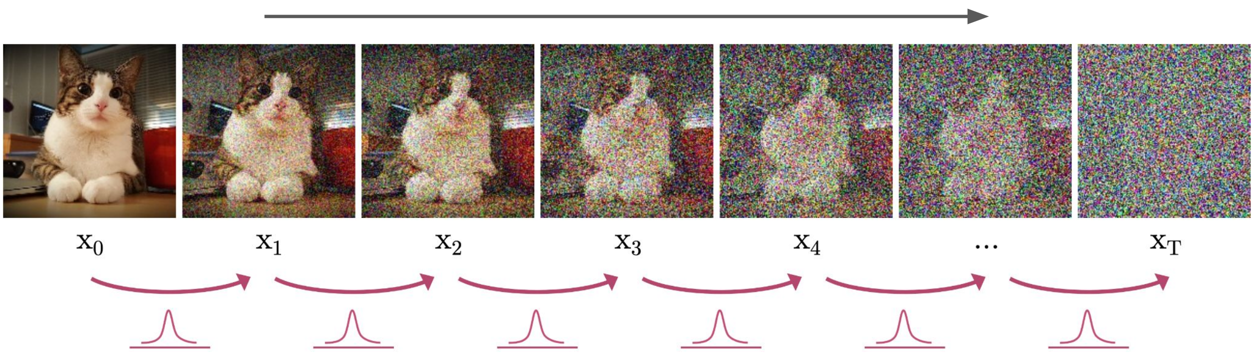
Data

Forward Diffusion Process



The formal definition of the forward process in T steps:

Forward diffusion process (fixed)



Noise

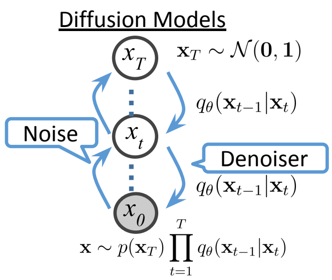
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) = \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I}) \;\rightarrow\; q(\mathbf{x}_{1:T}|\mathbf{x}_0) = \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1}) \qquad \text{(joint)}$$

β_s controls the *noise schedule* such that $\mathbf{x}_T$ is complete noise.

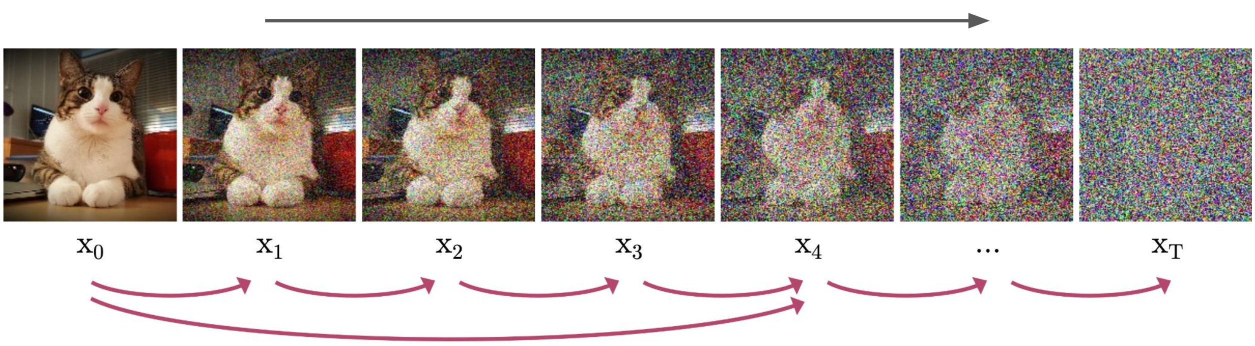
Slide credit: Kreis, Gao, Vahdat, CVPR 2022 Tutorial

Data

Diffusion Kernel



Forward diffusion process (fixed)



Noise

"Skip" *t* steps **(Diffusion Kernel)**

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

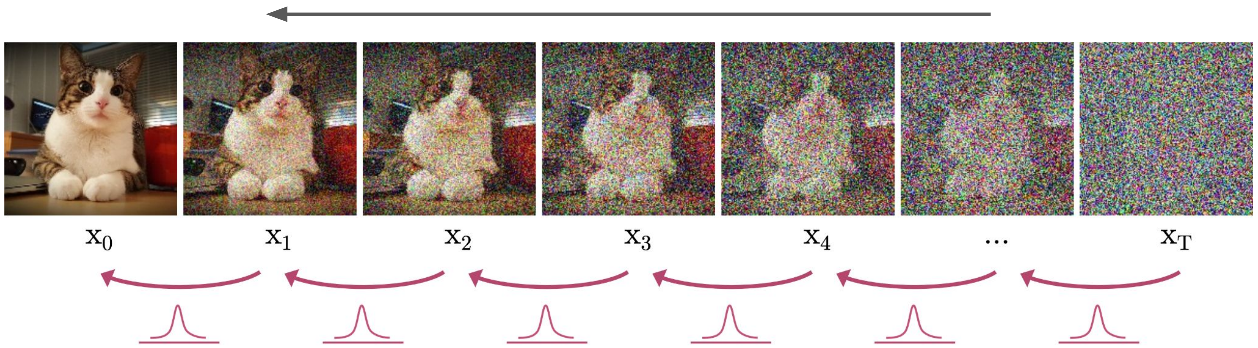
$$\text{Define } \bar{\alpha}_t = \prod_{s=1}^{t}(1-\beta_s)$$

Slide credit: Kreis, Gao, Vahdat, CVPR 2022 Tutorial

Data

Reverse Denoising Process

Reverse denoising process (generative)



$$q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$$

Intractable… 😟

Idea: Approximate $q(\mathbf{x}_{t-1}|\mathbf{x}_t)$. Use a Normal distribution if β_t is small in each forward diffusion step.

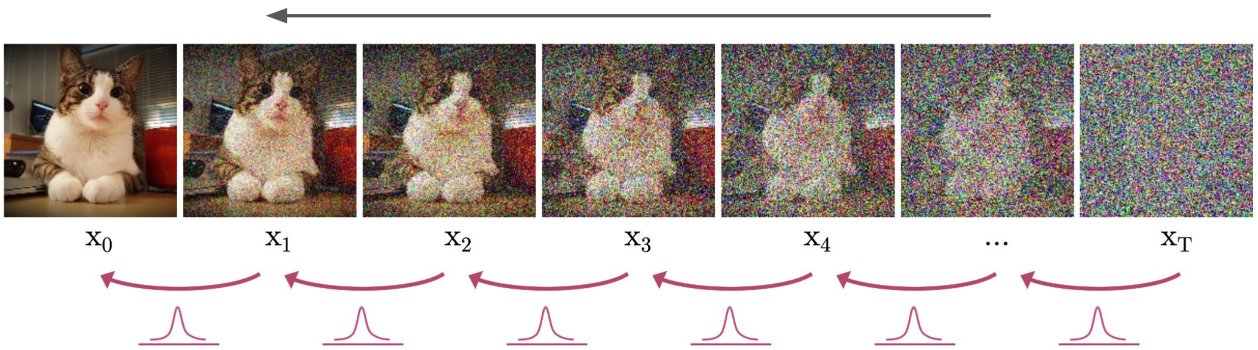
Slide credit: Kreis, Gao, Vahdat, CVPR 2022 Tutorial

Data

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:

Reverse denoising process (generative)



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

$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\frac{1}{\sqrt{\alpha_t}}\left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \underbrace{\boldsymbol{\epsilon}(\mathbf{x}_t, t)}_{\text{Trainable network}}\right)}, \sigma_t^2 \mathbf{I}) \rightarrow p_\theta(\mathbf{x}_{0:T}) = p(\mathbf{x}_T)\prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$$

Apply $\mu_\theta(\cdot)$ T times!

Slide credit: Kreis, Gao, Vahdat, CVPR 2022 Tutorial

U-Net: Image-to-Image CNN network

The trainable noise prediction network is usually a U-Net

Contracting path

Expanding path

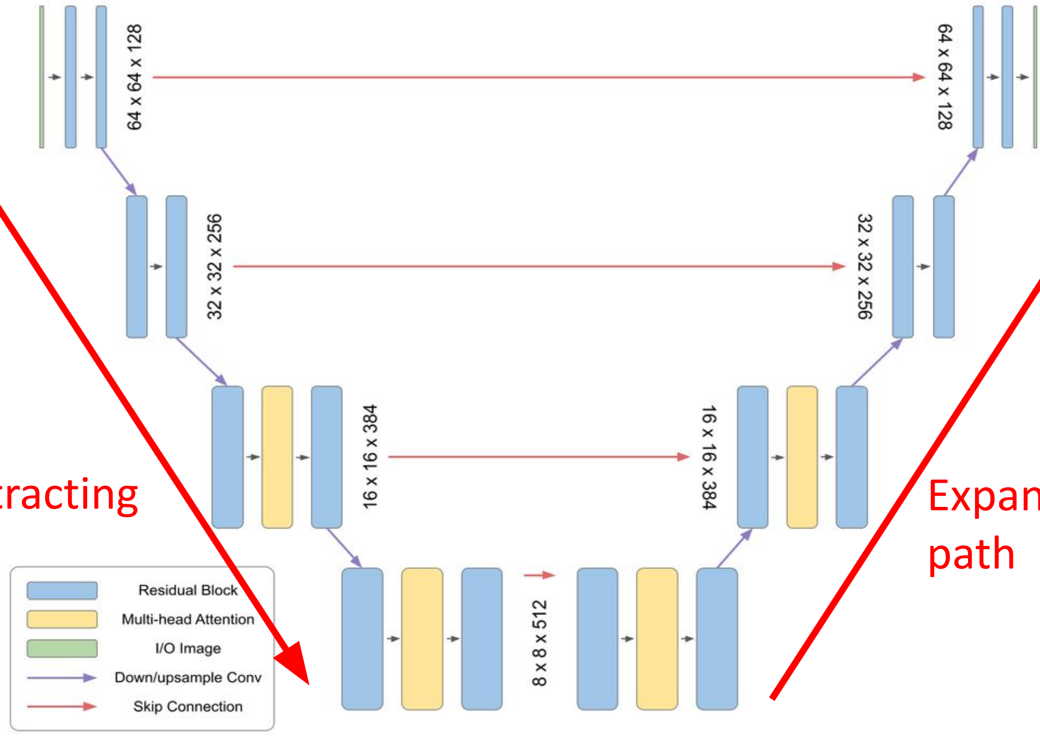
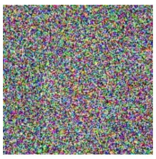


Image Source: https://www.assemblyai.com/blog/minimagen-build-your-own-imagen-text-to-image-model/

Training

Repeat:

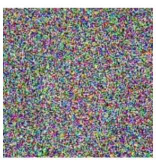
Sample:  $\mathbf{x}_0$  $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$

$$t \sim \text{Uniform}(\{1, \ldots, \text{T}\})$$

Diffusion kernel:

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

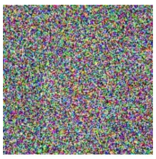
Add noise to clean image (with variance equivalent to t steps of noise)

Gradient update: $\nabla_\theta \Bigg\|$  $- \boldsymbol{\epsilon}_\theta\Bigg($  $, t\Bigg)\Bigg\|^2$

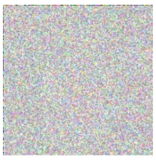
$$\nabla_\theta \left\| \boldsymbol{\epsilon} - \boldsymbol{\epsilon}_\theta \left(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1 - \bar{\alpha}_t}\boldsymbol{\epsilon}, t \right) \right\|^2$$

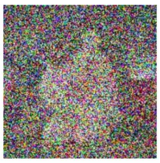
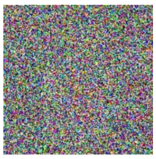
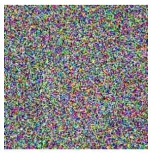
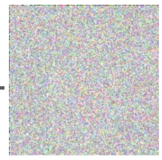
Give model noisy image, ask it to predict the added noise. Train with L2 loss between predicted noise and original noise

Sampling

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

for t = 1 … T **do**:

Sample noise:  $\mathbf{z}_t \sim \mathcal{N}(\mathbf{0}, \sigma_t \mathbf{I})$

Predict next step:  $= \frac{1}{\sqrt{\alpha_t}} \Big($  $- \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta \Big($  $, t \Big)\Big) +$ 

$$x_{t-1} = \frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t) \right) + \mathbf{z}_t$$

Summary: Training and Sample Generation

Algorithm 1 Training

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

Algorithm 2 Sampling

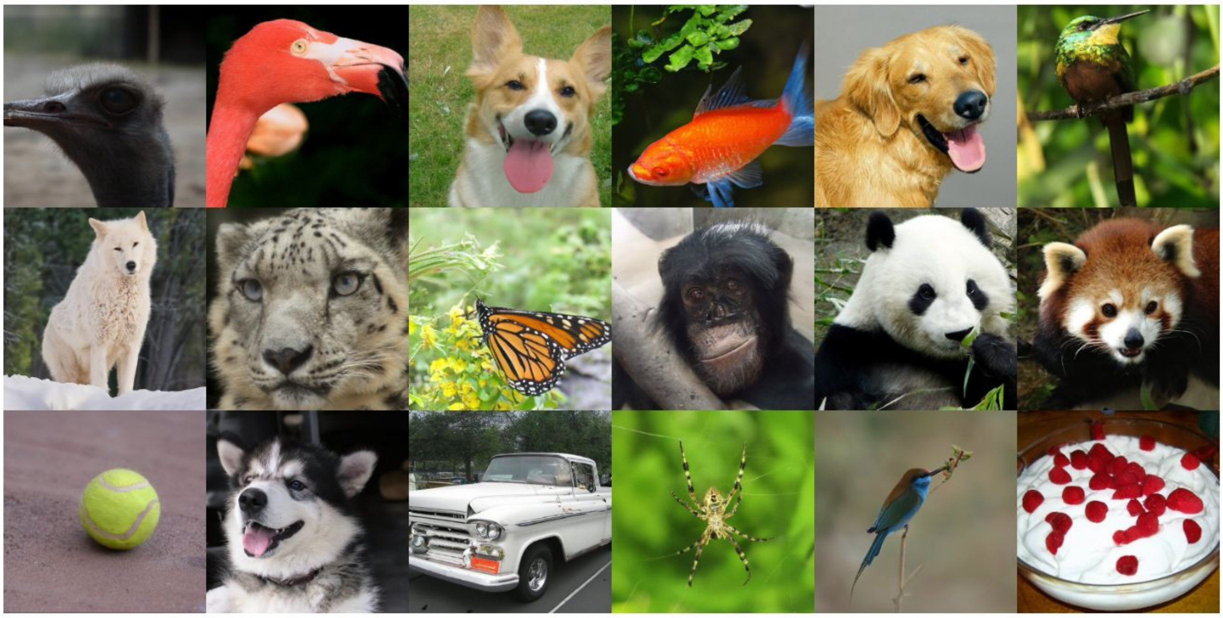
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: **for** $t = T, \dots, 1$ **do**
3: $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
4: $\mathbf{x}_{t-1} = \boxed{\frac{1}{\sqrt{\alpha_t}} \left(\mathbf{x}_t - \frac{1-\alpha_t}{\sqrt{1-\bar{\alpha}_t}} \boldsymbol{\epsilon}_\theta(\mathbf{x}_t,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

Slide credit: Kreis, Gao, Vahdat, CVPR 2022 Tutorial

Improvements: Conditional Generation

Guided Diffusion

512 x 512 ImageNet Conditioned Samples

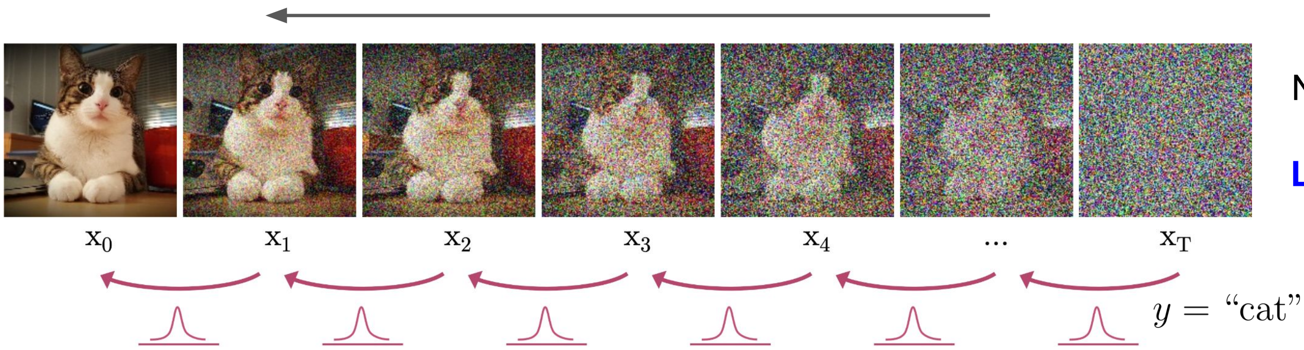


Dhariwal and Nichol “Diffusion Models Beat GANs on Image Synthesis” NeurIPS 2021

Data

Improvements: Conditional Generation

Conditional Reverse denoising process (generative)



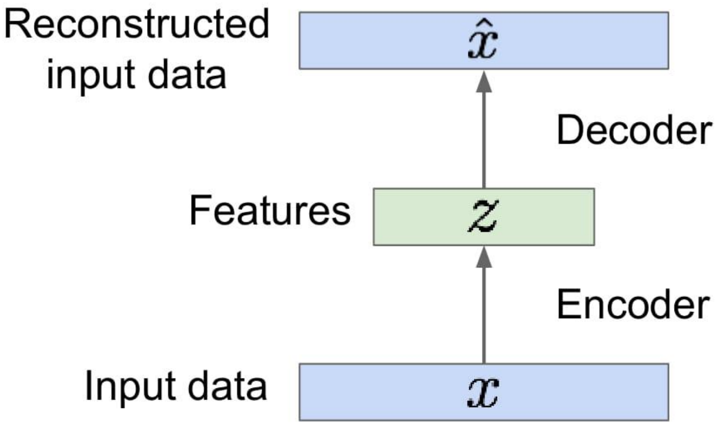
Noise and **Label**

We pass the condition (y=cat) to the noise-predicting model in addition to the noisy image and step number to “control” the generation (i.e. force the generated image to be a cat instead of a dog)

Text-to-image generation: train the diffusion model to condition on text!

Improvements: Latent Diffusion Models

Use an **autoencoder** to compress the data!

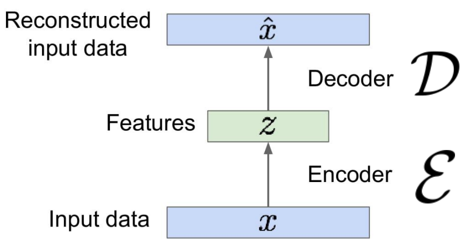


Original Image: 256x256x3

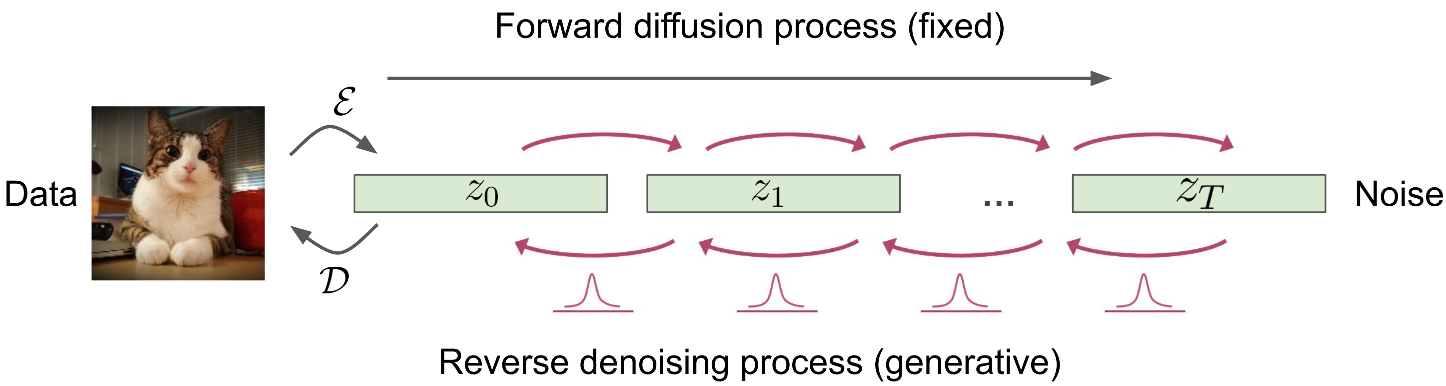
Latent Representation: 32x32x4

Reconstructed Image: 256x256x3

Improvements: Latent Diffusion Models



Use an **autoencoder** to compress the data!



Most popular text-to-image generation models nowadays are Latent Diffusion Models

For example,

- OpenAI DALLE-2 and DALLE-3, and likely all images generated from text within ChatGPT
- Midjourney
- Imagen (by Google)

Diffusion models can also be used for text-to-video generation.

- OpenAI Sora is a latent diffusion model with transformer architecture

Recap

		Pros	Cons
Autoregressive (PixelRNN, PixelCNN)	Explicit density model	Optimizes exact likelihood	Inefficient sequential generation, sample quality not the best
Variational Autoencoders (VAE)	Optimize variational lower bound on likelihood	Useful latent representation, inference queries	Sample quality not the best
Generative Adversarial Networks (GANs)	Game-theoretic approach	Great samples	Tricky and unstable to train, no inference queries
Diffusion Models	Optimize variational lower bound for denoising	Best samples! Easy to train	Inefficient sequential generation

Slide credit: Justin Johnson