Deep Generative Learning

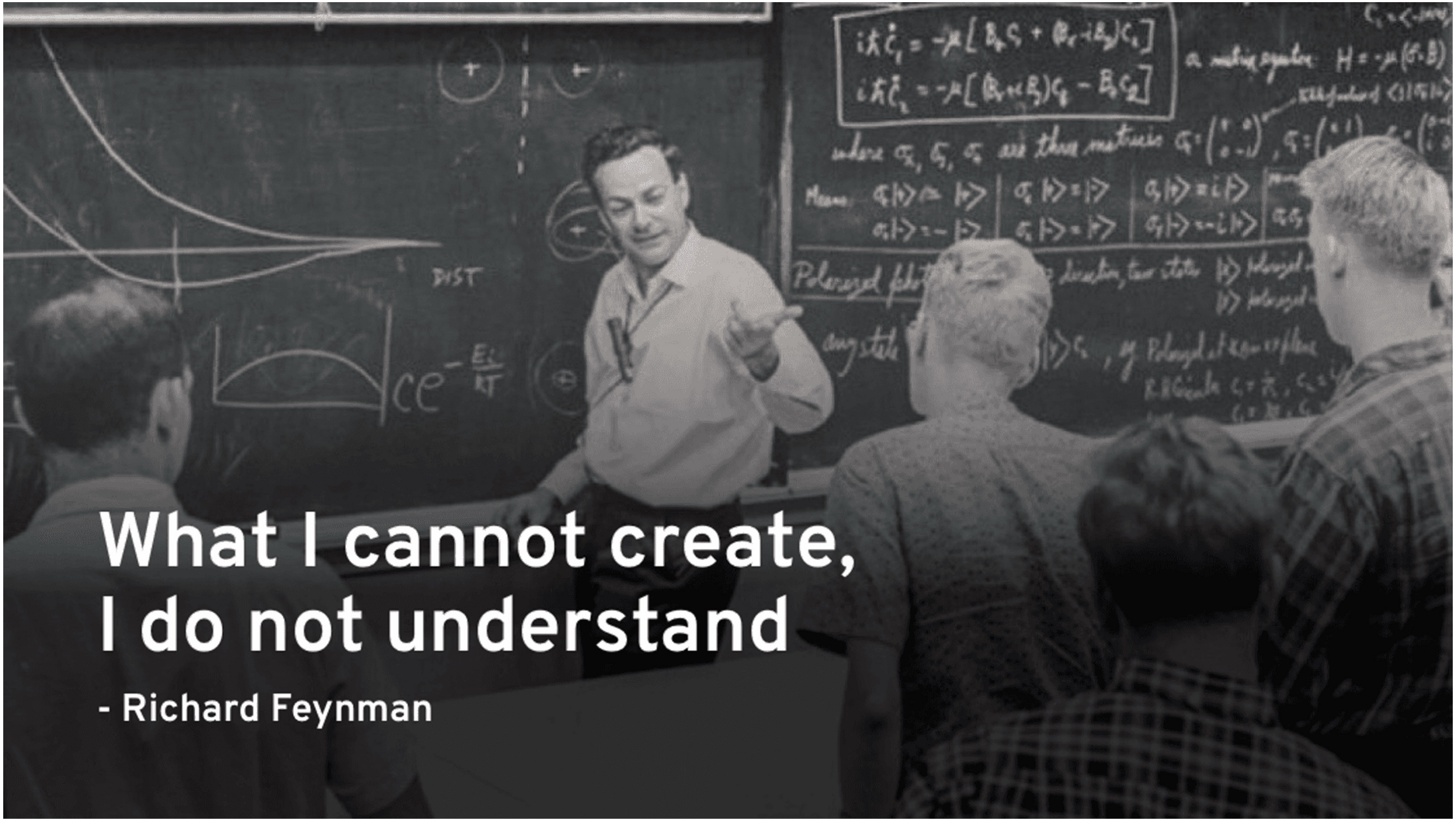
Seongro Yoon

03.03.25

Outline

- Introduction
  - What is a generative model?
  - Types of deep generative models
- Image Generation
  - Autoregressive models
  - Generative Adversarial Networks
  - Variational Autoencoders
  - Diffusion Models
- Evaluation
  - Inception Score
  - Frechet Inception Distance

Introduction


What I cannot create,
I do not understand
- Richard Feynman

Generative AI in Image Applications

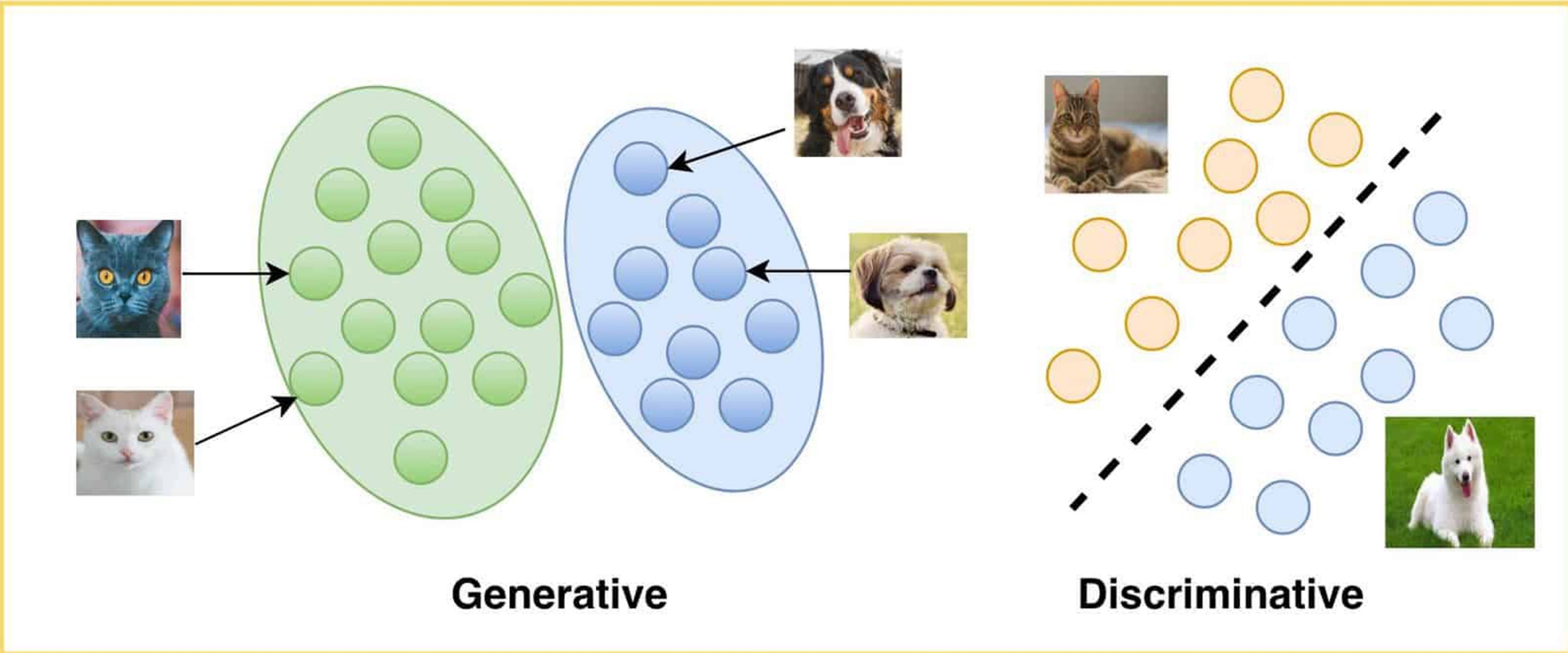
Art & Design

Content Generation

Representation Learning

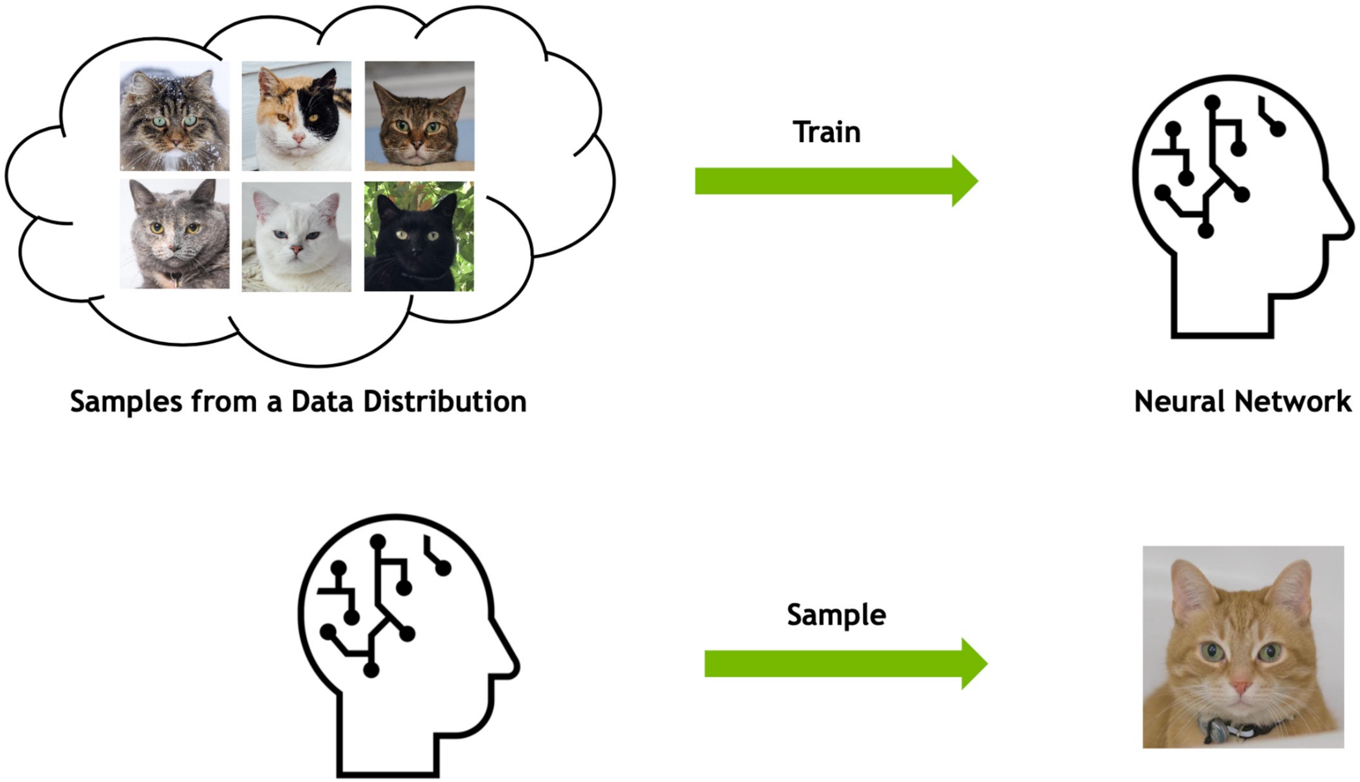
Entertainment

What is a Generative Model?

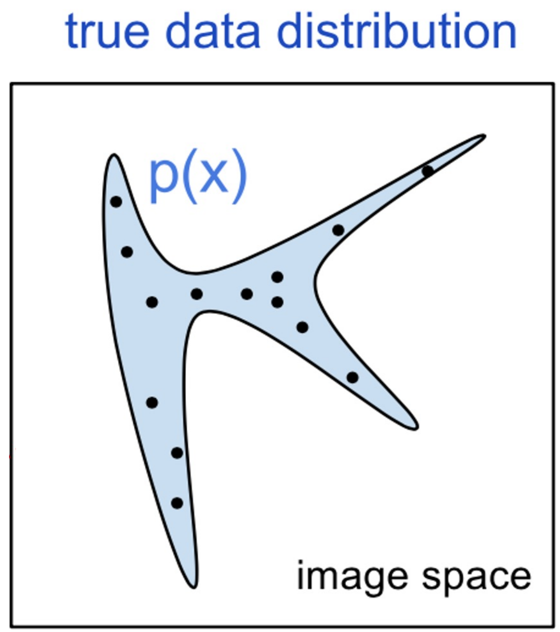


source: https://learnopencv.com/generative-and-discriminative-models/

What is a Generative Model?

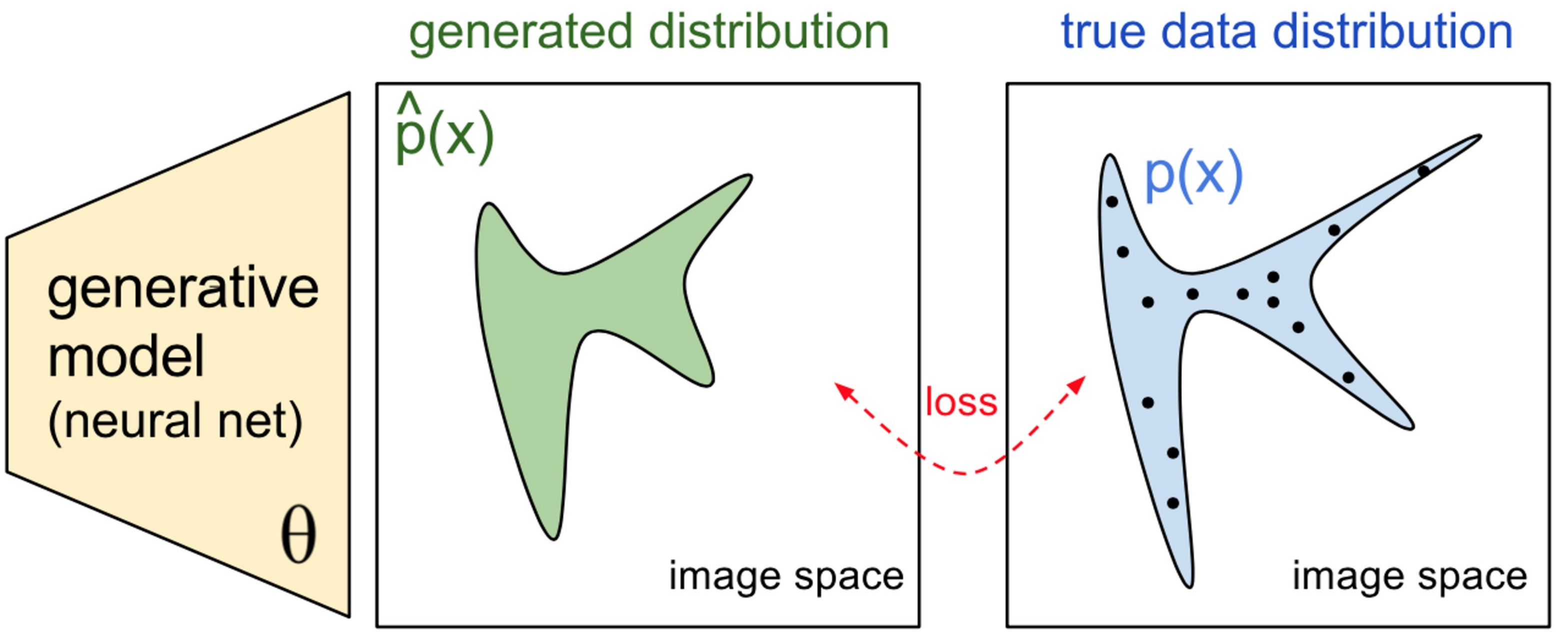


What is a Generative Model?

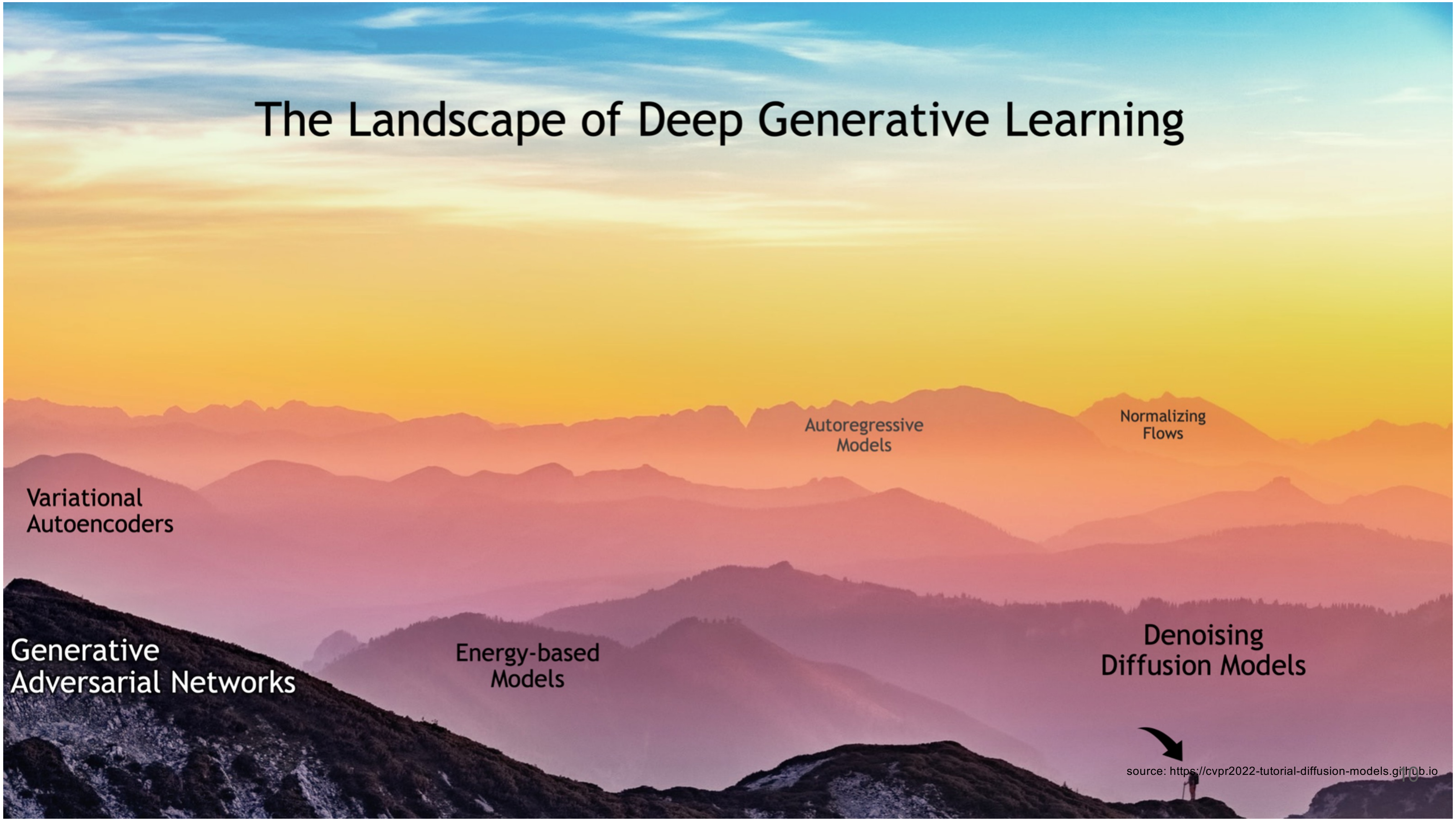


source: https://openai.com/research/generative-models

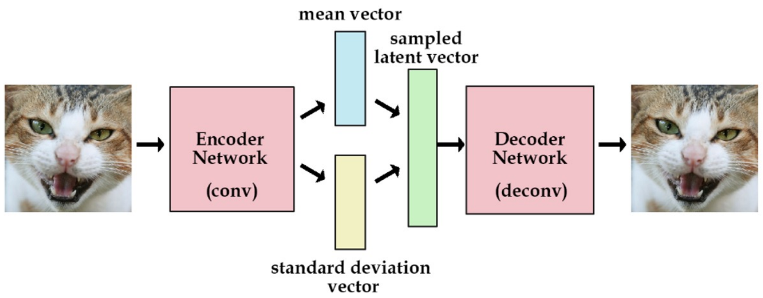
What is a Generative Model?



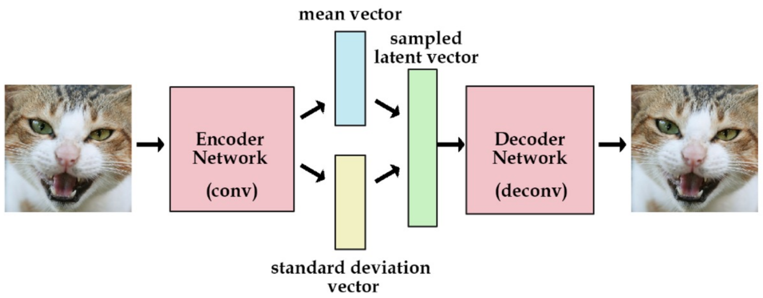
source: https://openai.com/research/generative-models

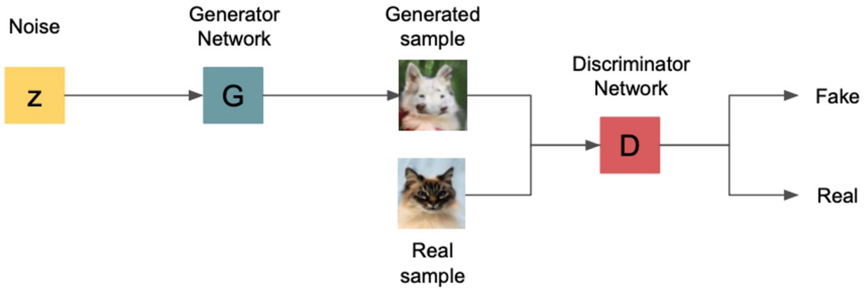

The Landscape of Deep Generative Learning
Autoregressive Models
Normalizing Flows
Variational Autoencoders
Generative Adversarial Networks
Energy-based Models
Denoising Diffusion Models
source: https://cvpr2022-tutorial-diffusion-models.github.io
10

Types of generative models

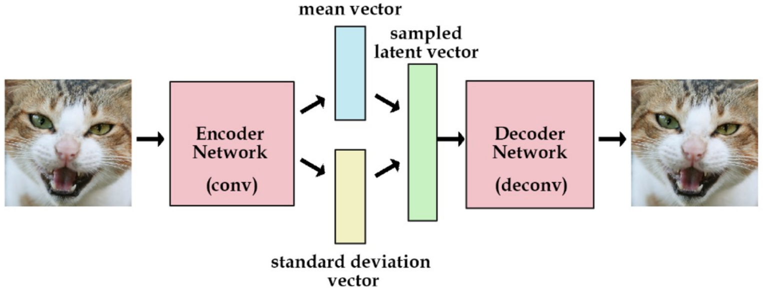


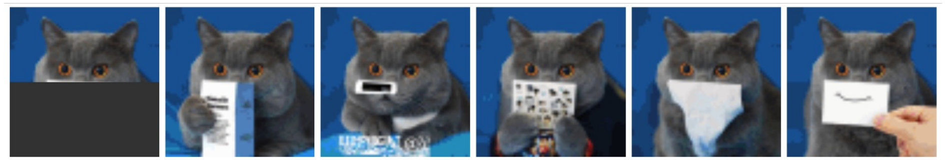
Types of generative models

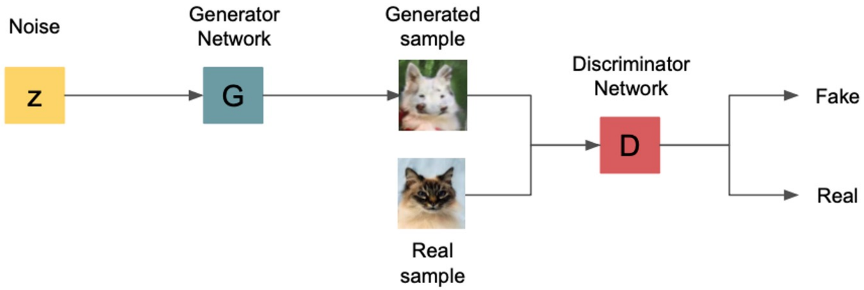




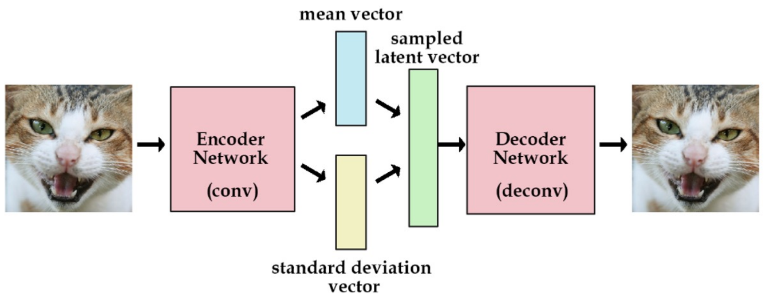
Types of generative models

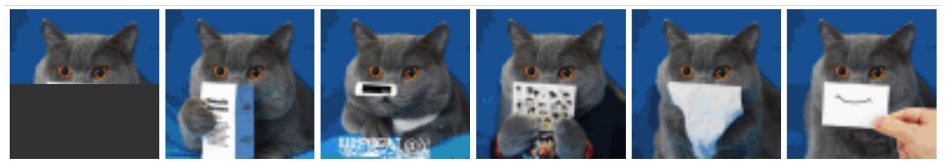


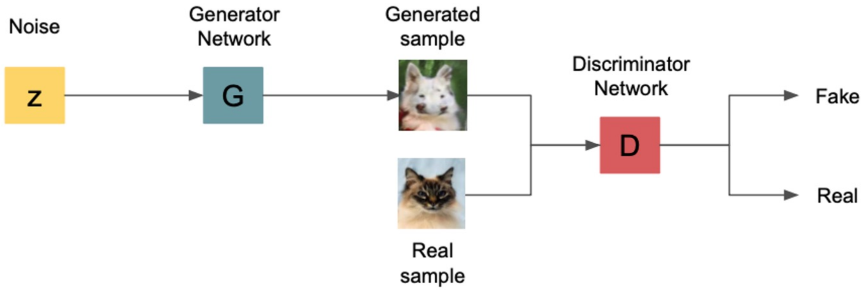




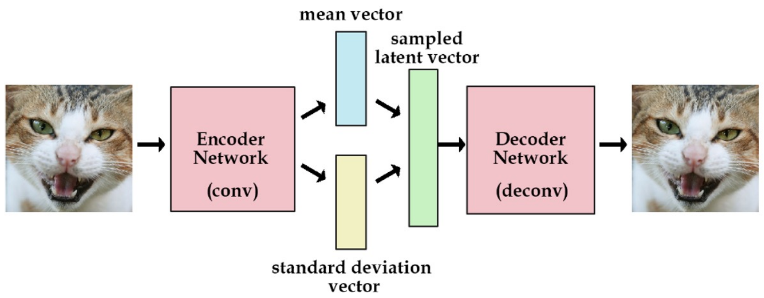
Types of generative models

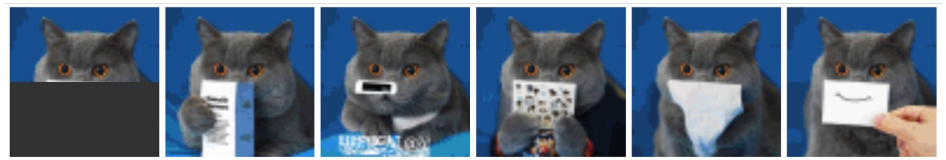


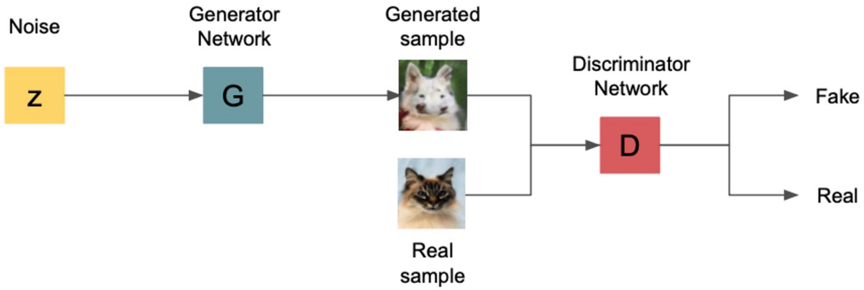




Types of generative models

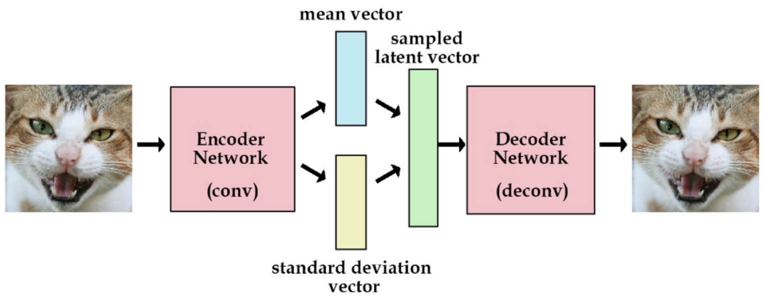


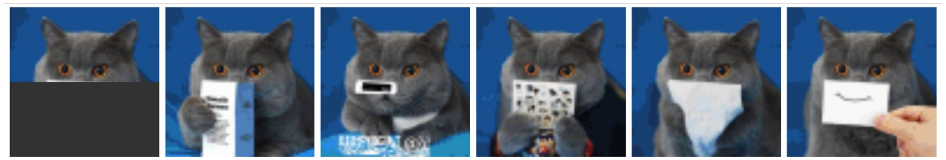


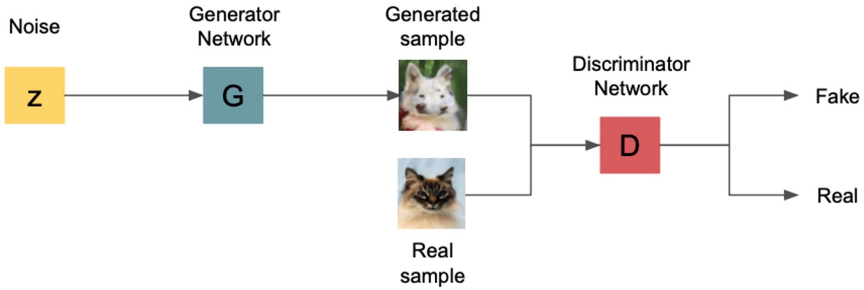


Types of generative models

Variational Autoencoder:

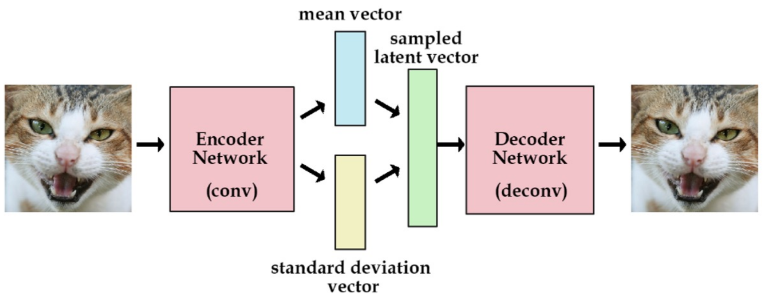


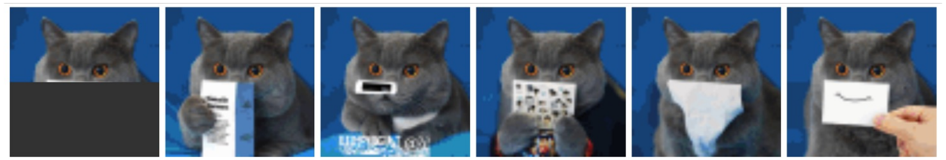


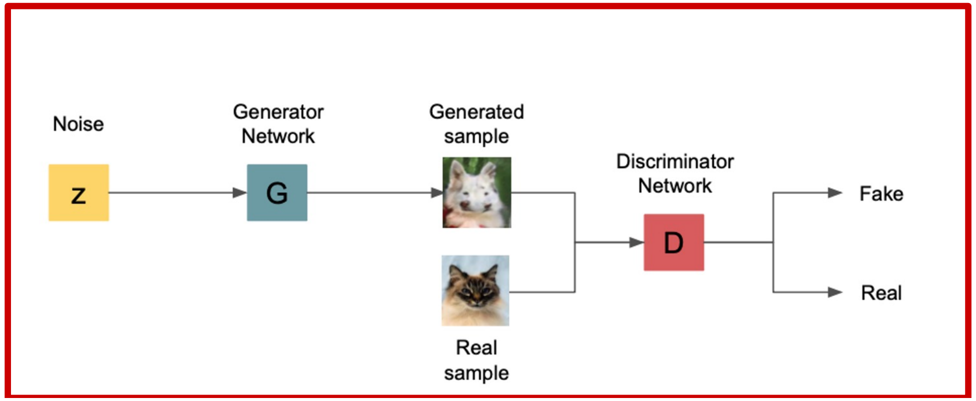


Types of generative models

Variational Autoencoder:

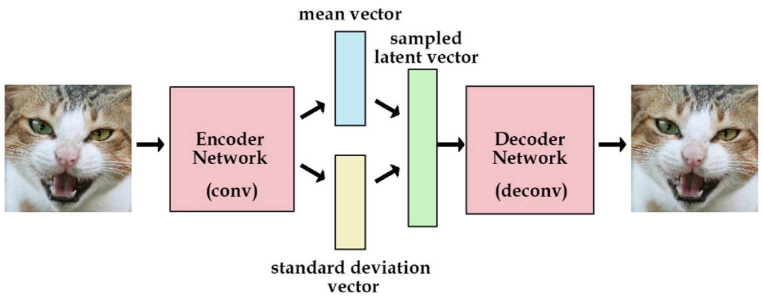


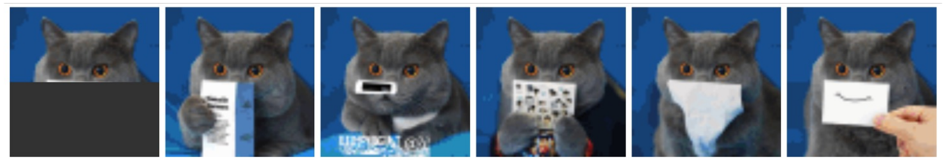




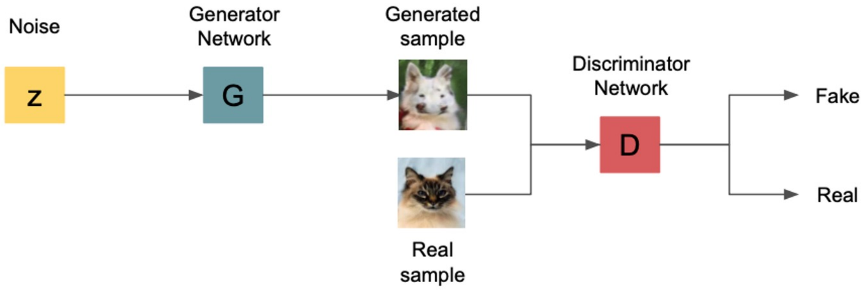
Types of generative models

Variational Autoencoder:



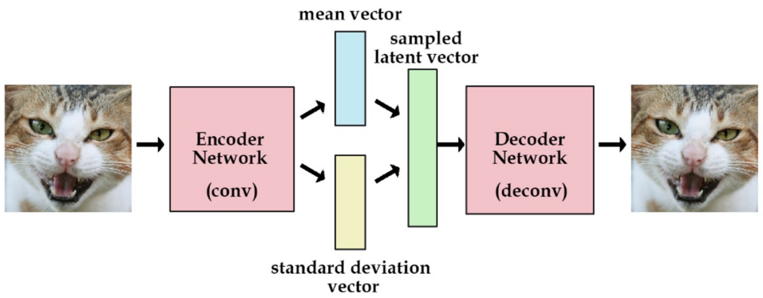


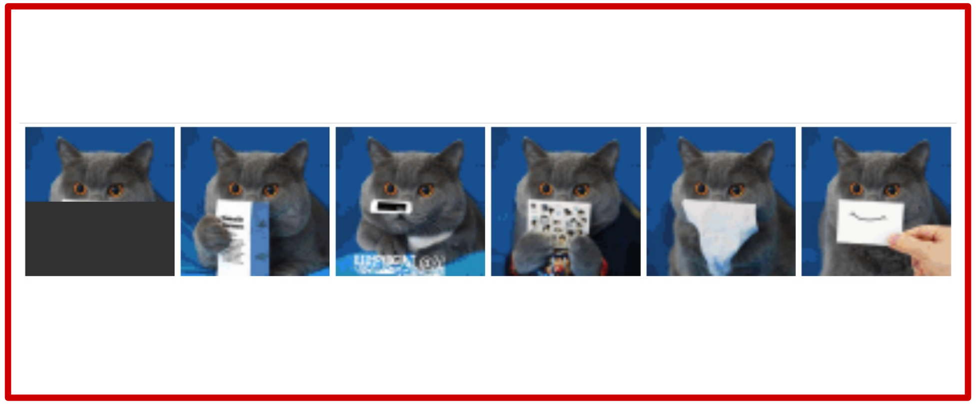
Generative Adversarial Network:



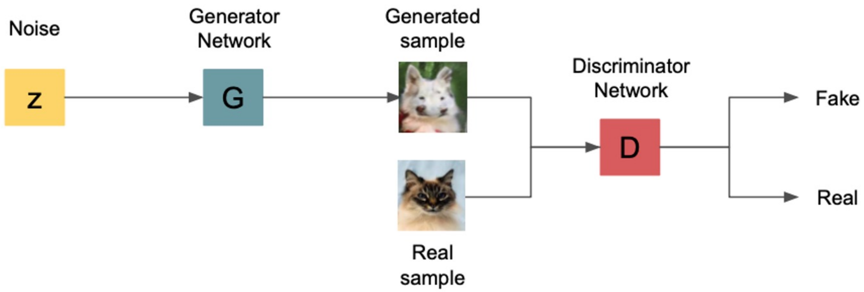
Types of generative models

Variational Autoencoder:



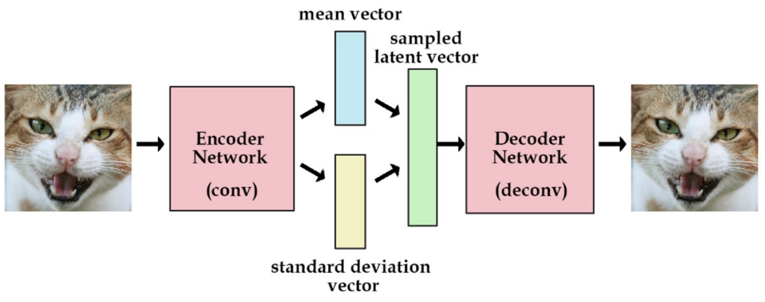


Generative Adversarial Network:

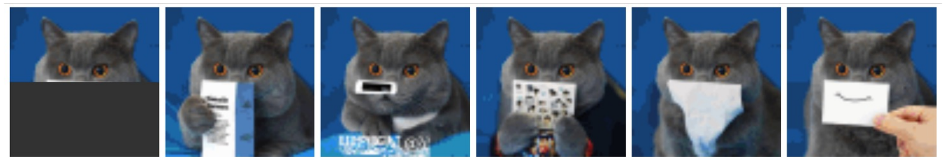


Types of generative models

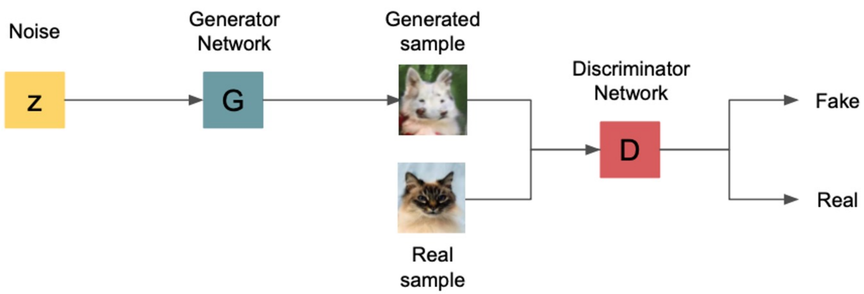
Variational Autoencoder:



Autoregressive model:

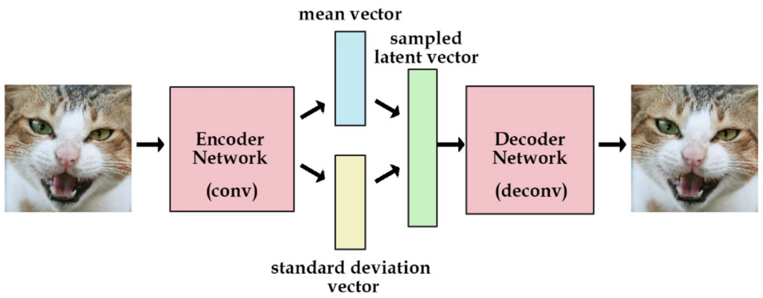


Generative Adversarial Network:

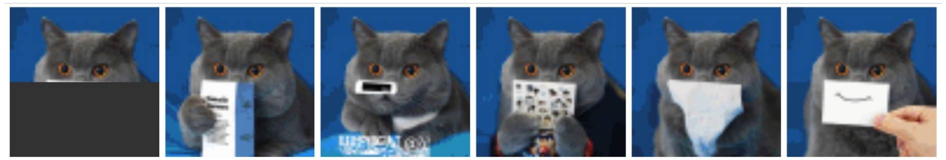


Types of generative models

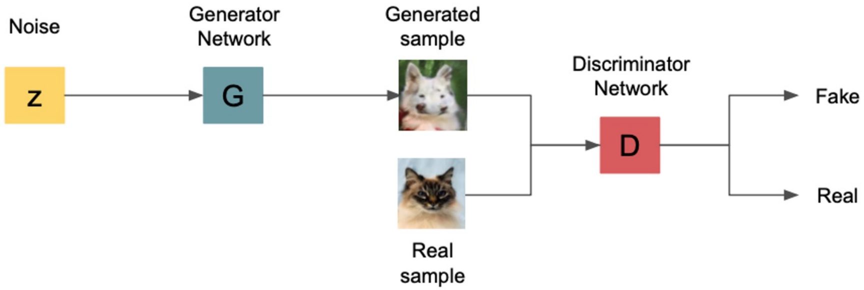
Variational Autoencoder:



Autoregressive model:

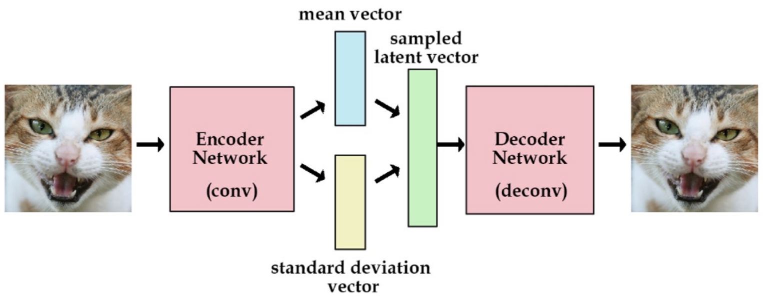


Generative Adversarial Network:

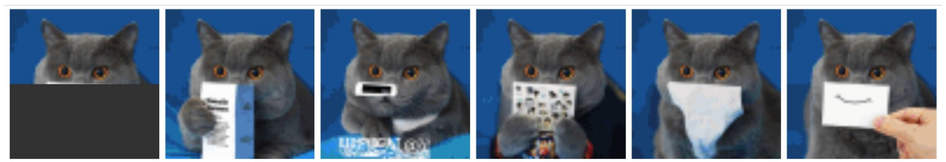


Types of generative models

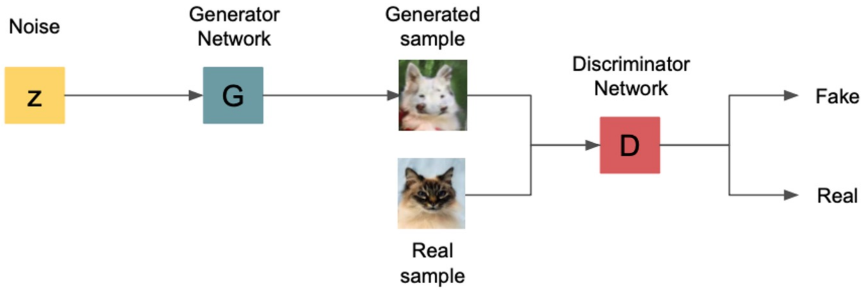
Variational Autoencoder:



Autoregressive model:



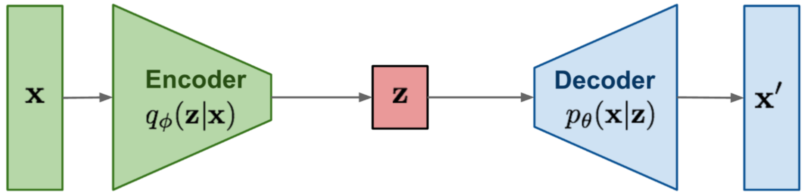
Generative Adversarial Network:



Diffusion model:

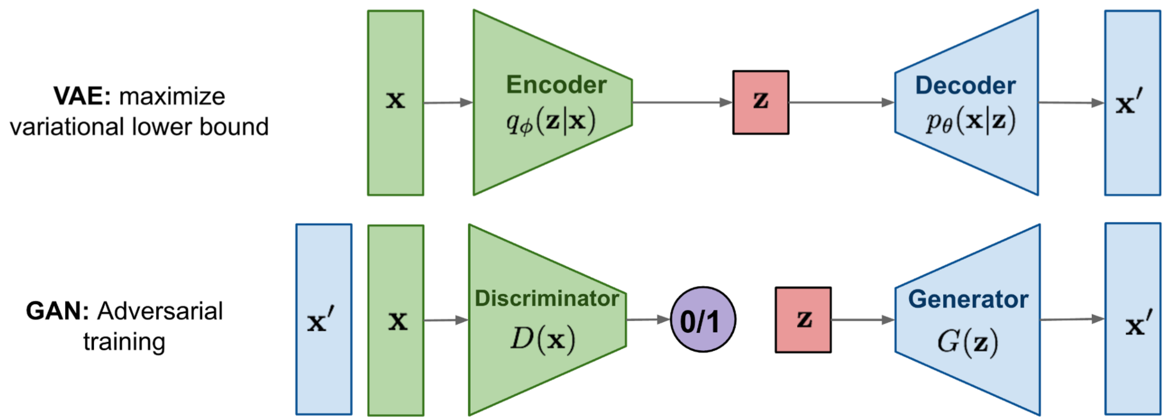
Types of generative models

VAE: maximize variational lower bound



source: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Types of generative models



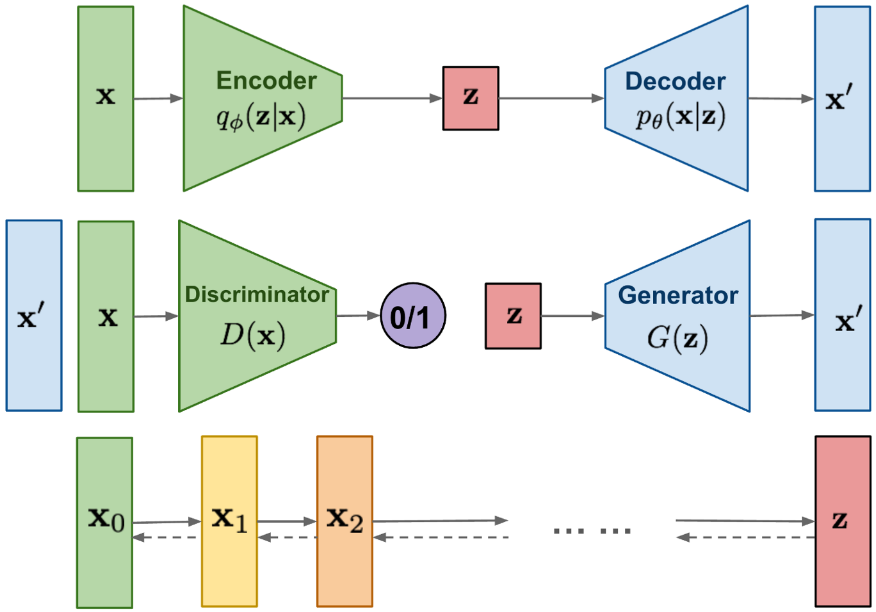
source: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Types of generative models

VAE: maximize variational lower bound

GAN: Adversarial training

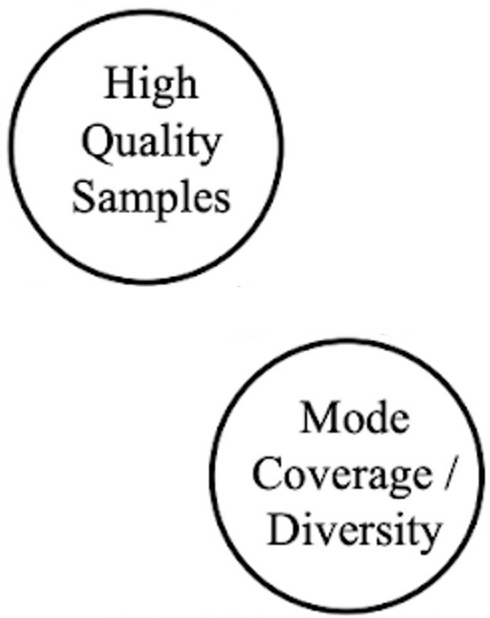
Diffusion models: Gradually add Gaussian noise and then reverse



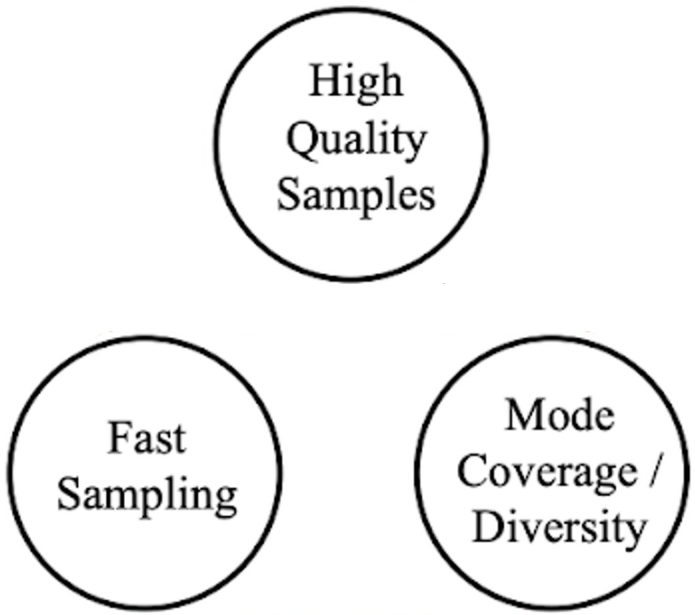
source: https://lilianweng.github.io/posts/2021-07-11-diffusion-models/

Generative learning trilemma

Generative learning trilemma



Generative learning trilemma



Generative learning trilemma

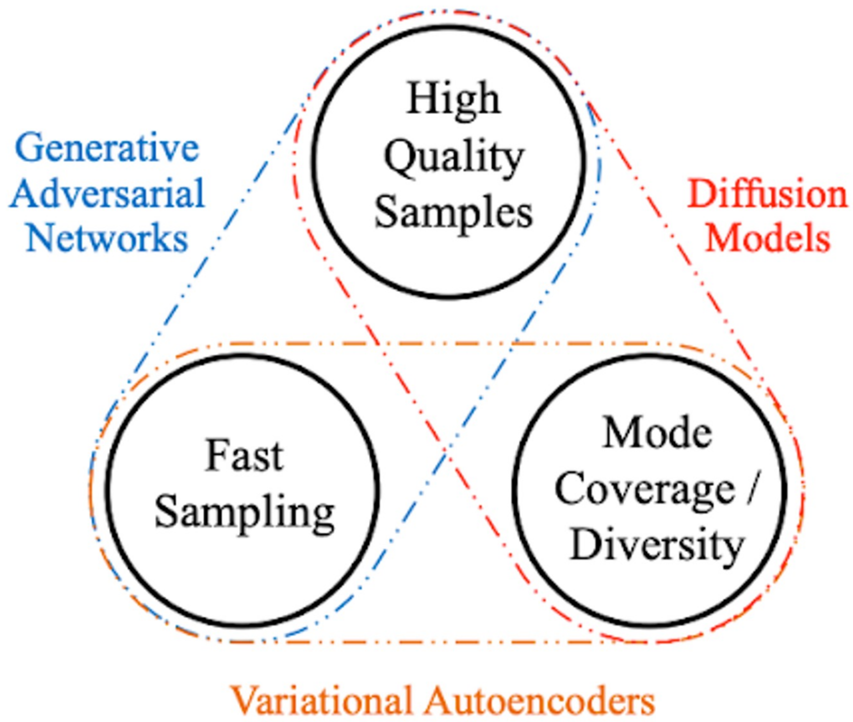
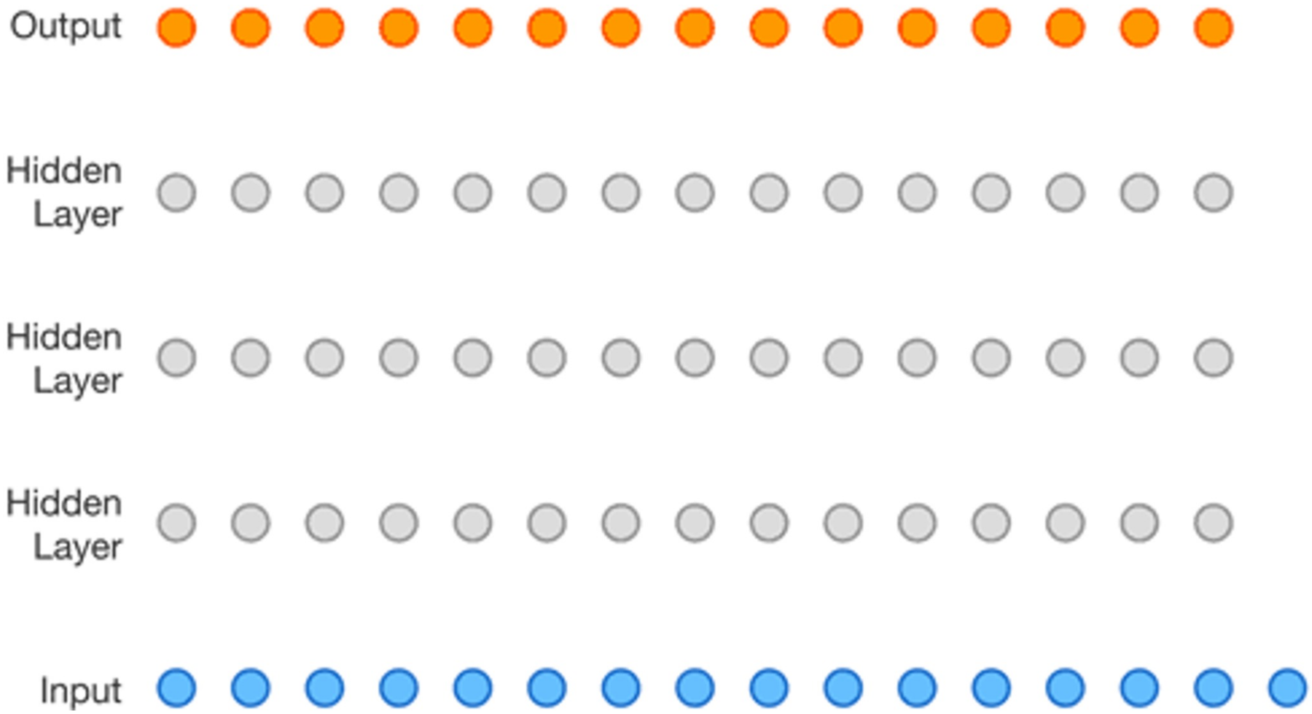
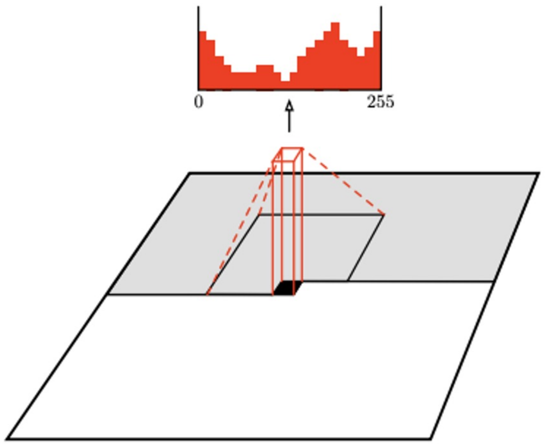


Image Generation

Autoregressive Models



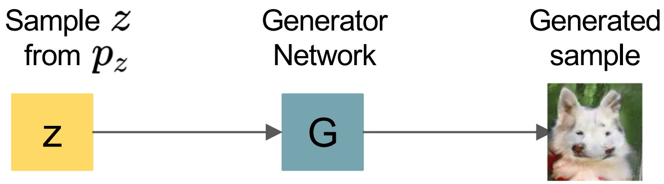
PixelCNN



van den Oord et al. Pixel Recurrent Neural Networks. ICML 2016

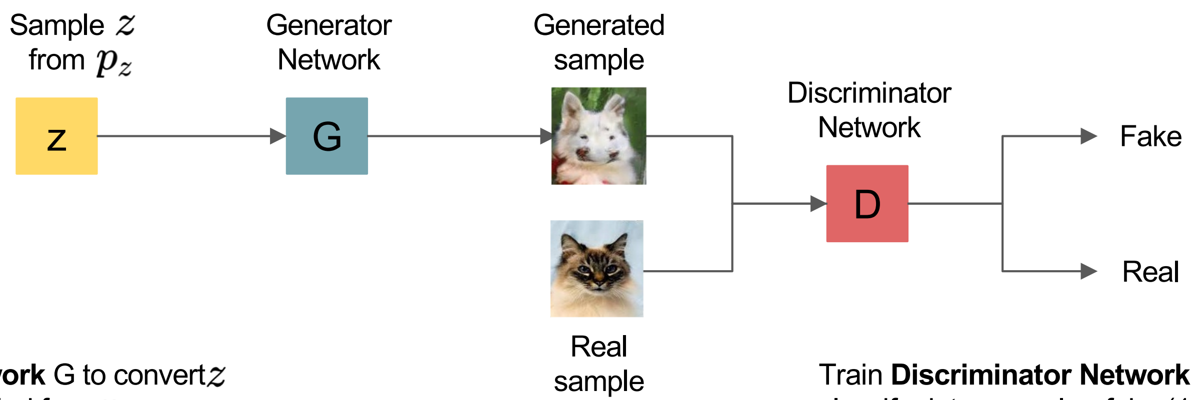
Generative Adversarial Networks

- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.
- Sample $z \sim p(z)$ 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

- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.
- Sample $z \sim p(z)$ 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

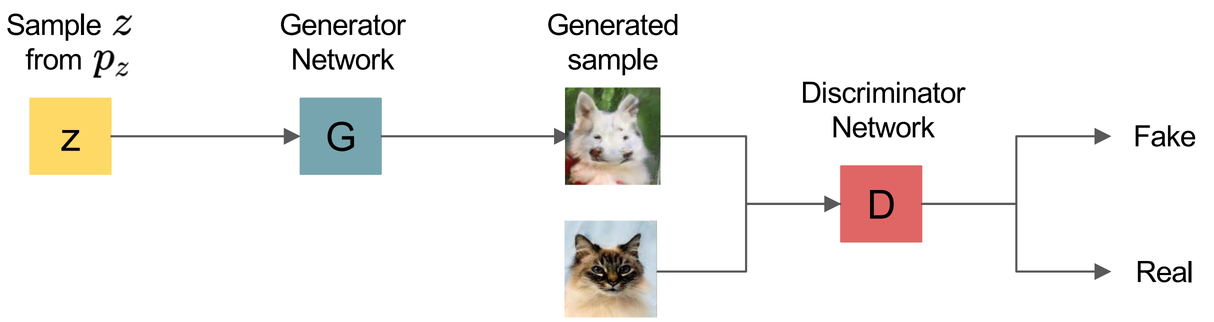
Train **Discriminator Network** D to classify data as real or fake (1/0)

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014

Generative Adversarial Networks: Training Objective

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

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



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)

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014

Generative Adversarial Networks: Training Objective

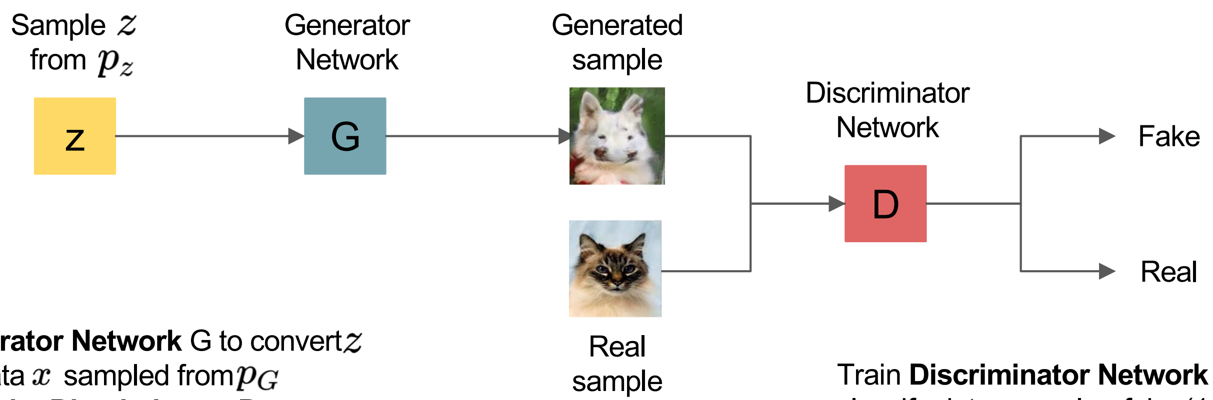
Jointly train generator G and discriminator D with a **minimax game**

Discriminator wants D(x)=1 for real data

Discriminator wants D(x)=0 for fake data

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

Generator wants D(x)=1 for fake 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)

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014

Generative Adversarial Networks: Training Objective

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

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

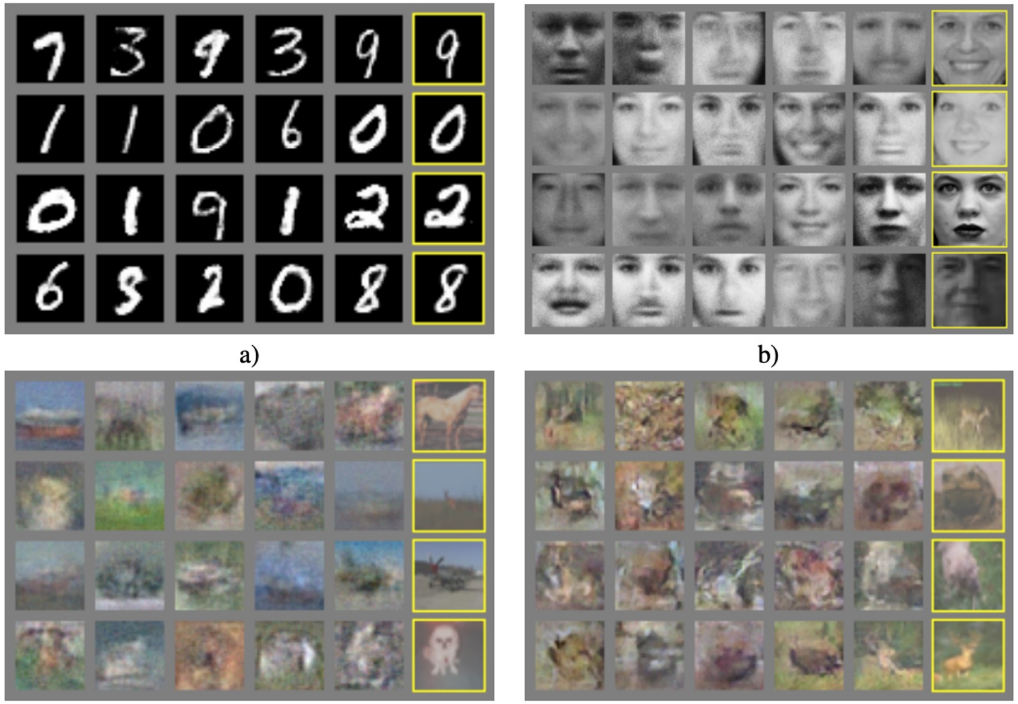
$$= \min_{\mathbf{G}} \max_{\mathbf{D}} \mathbf{V}(\mathbf{G}, \mathbf{D})$$

Train G and D using alternating gradient updates:

1. Update $\mathbf{D} = \mathbf{D} + \alpha_{\mathbf{D}} \frac{\delta \mathbf{V}}{\delta \mathbf{D}}$
2. Update $\mathbf{G} = \mathbf{G} - \alpha_{\mathbf{G}} \frac{\delta \mathbf{V}}{\delta \mathbf{G}}$

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014

Generative Adversarial Networks: first results



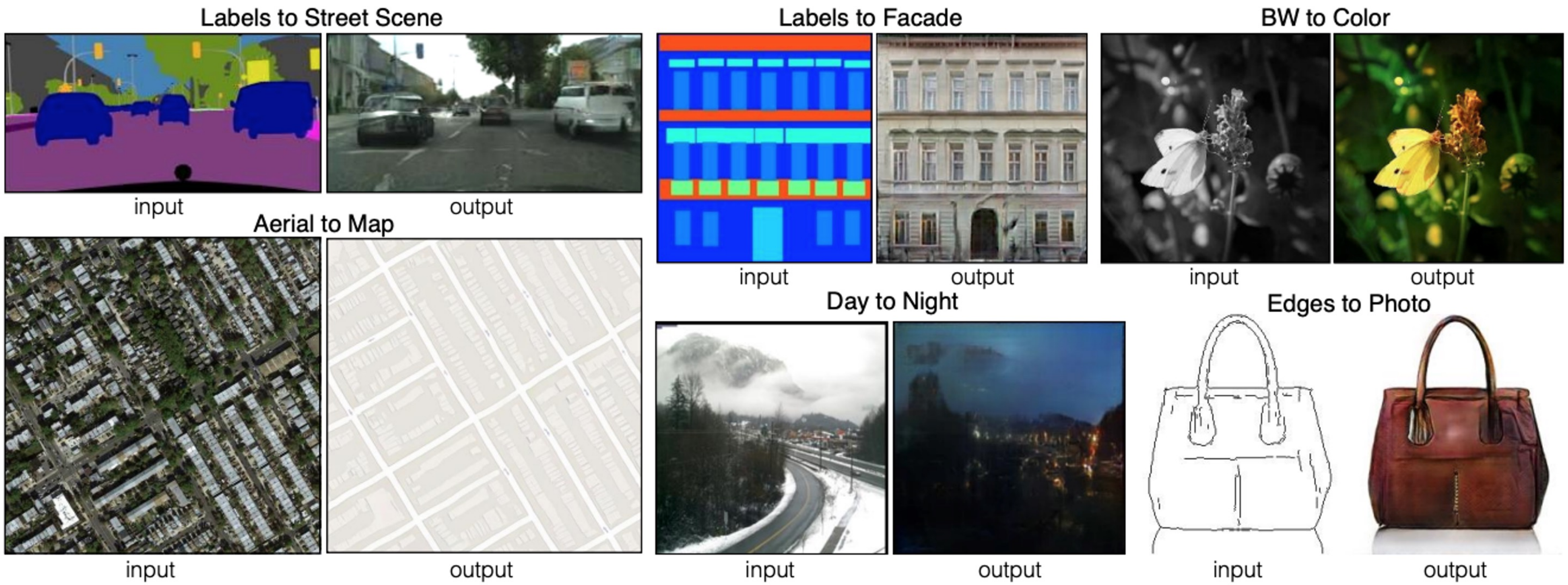
a) b) c) d)

Goodfellow et al. Generative Adversarial Nets. NeurIPS 2014

StyleGAN

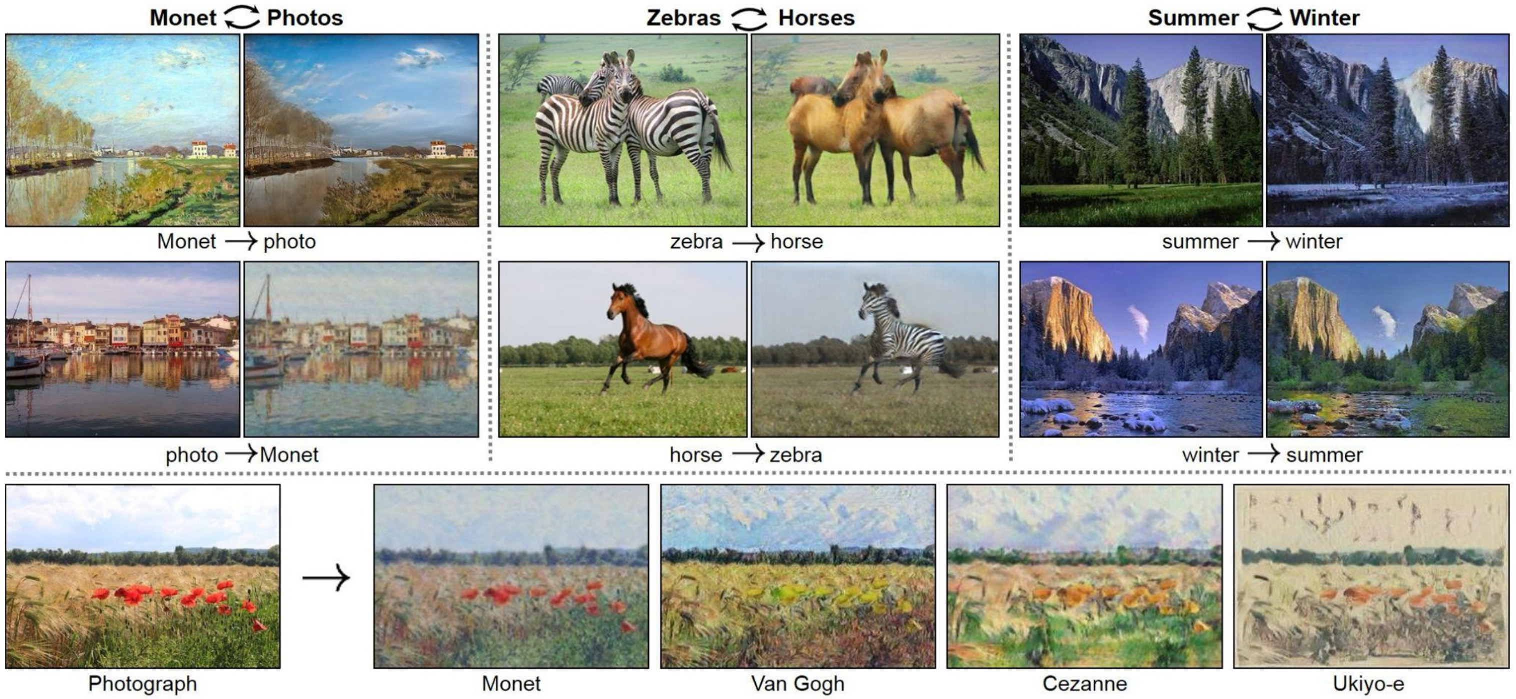
Karras et al. A Style-Based Generator Architecture for Generative Adversarial Networks. CVPR 2019

Image-to-Image Translation: Pix2Pix



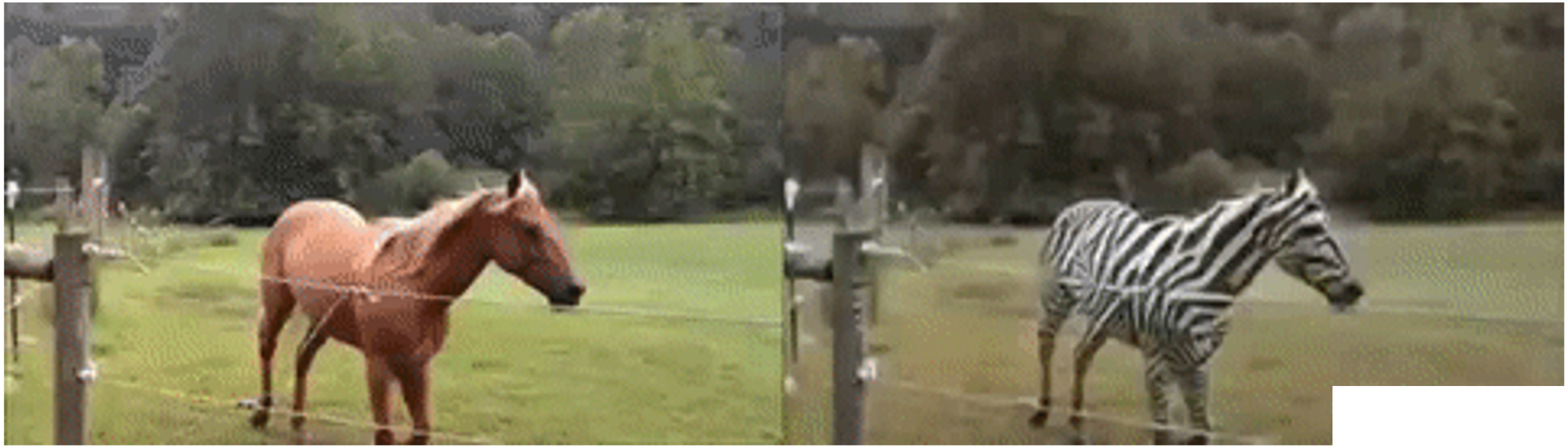
Isola et al, Image-to-Image Translation with Conditional Adversarial Networks, CVPR 2017

Unpaired Image-to-Image Translation: CycleGAN



[Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017]

Unpaired Image-to-Image Translation: CycleGAN

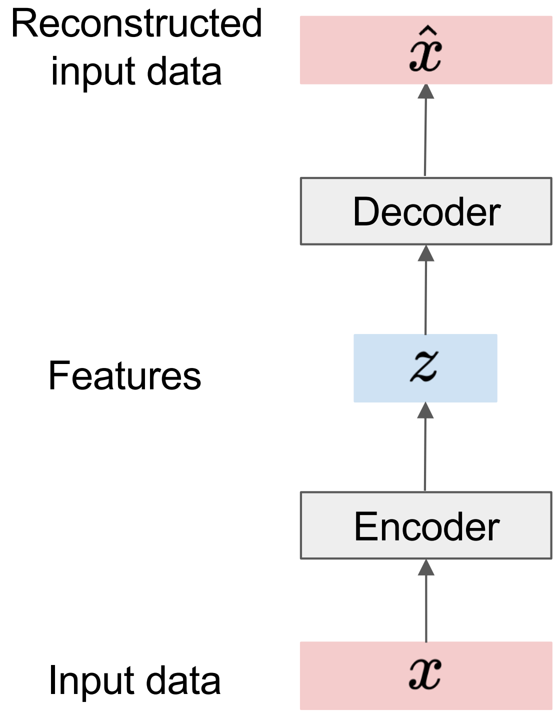


[Zhu et al. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017](#)

Autoencoders (non-variational)

Unsupervised method for learning latent features from data without any labels.

$$L = ||\hat{x} - x||_2^2$$

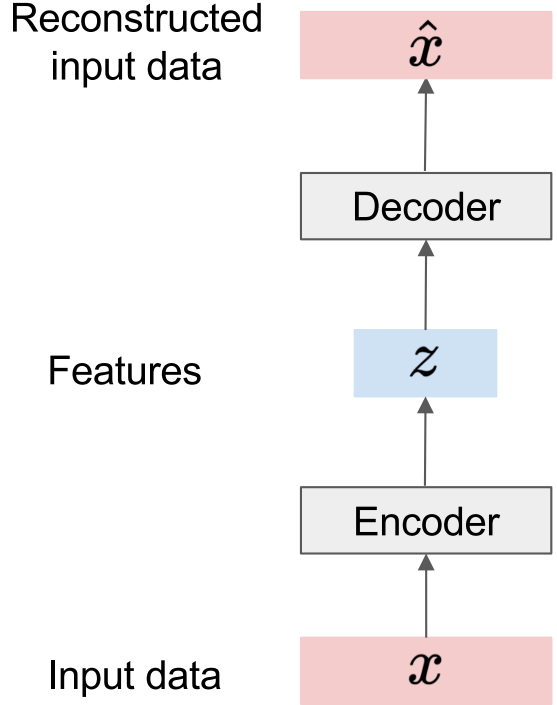


Autoencoders (non-variational)

Unsupervised method for learning latent features from data without any labels.

Features need to be **lower dimensional** than the data.

$$L = ||\hat{x} - x||_2^2$$



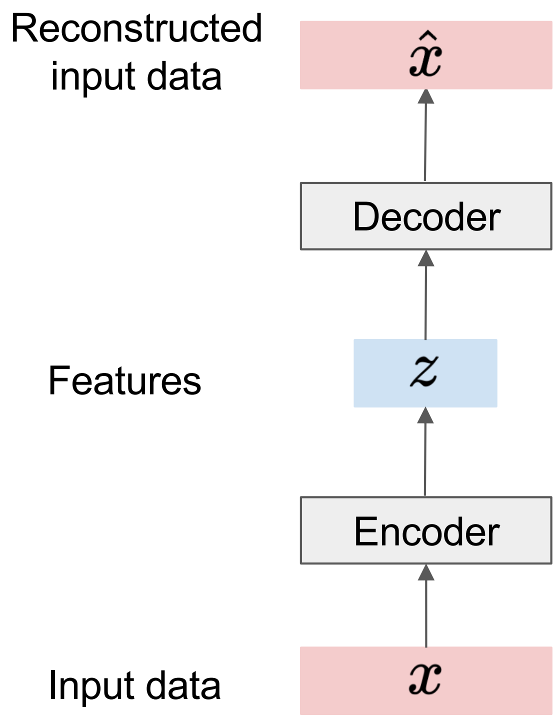
Autoencoders (non-variational)

$$L = ||\hat{x} - x||_2^2$$

Unsupervised method for learning latent features from data without any labels.

Features need to be **lower dimensional** than the data.

Limitation: no way to produce any new content



Variational Autoencoders (VAE)

Add a probabilistic constraint between the encoder and decoder

Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Add a probabilistic constraint between the encoder and decoder

VAE is an autoencoder that learns **latent features** from data and enables **generative process**.

Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Add a probabilistic constraint between the encoder and decoder

VAE is an autoencoder that learns **latent features** from data and enables **generative process**.

Instead of encoding an input as a single point, VAE encodes it as a distribution over the latent space.

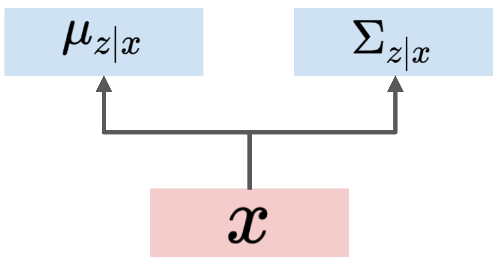
Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Encoder network inputs data x and outputs distribution over latent codes z

Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



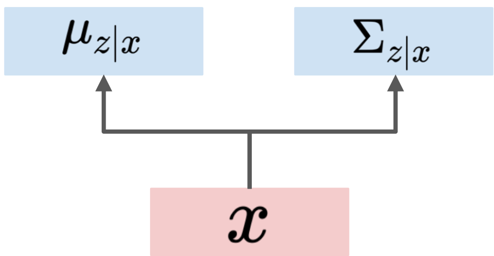
Variational Autoencoders (VAE)

Encoder network inputs data x and outputs distribution over latent codes z

Decoder network inputs latent code z and outputs distribution over data x

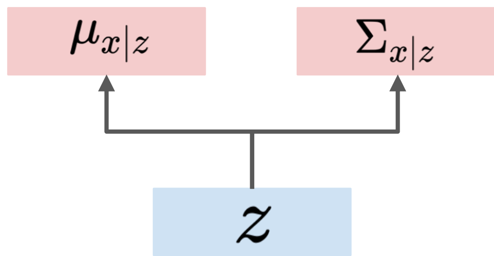
Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



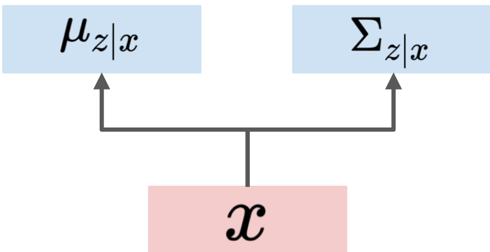
Variational Autoencoders (VAE)

Jointly train **encoder** q and **decoder** p to maximize the **variational lower bound** on the data likelihood

$$\log p_\theta(x) \geq E_{z \sim q_\phi(z|x)}[\log p_\theta(x|z)] - KL(q_\phi(z|x), p(z))$$

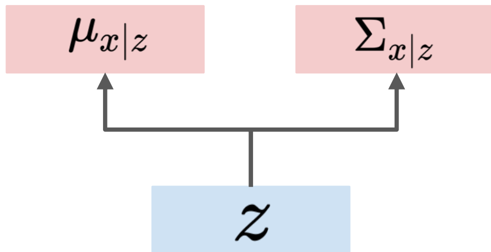
Encoder Network

$$q_\phi(z|x) = N(\mu_{z|x}, \Sigma_{z|x})$$



Decoder Network

$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$

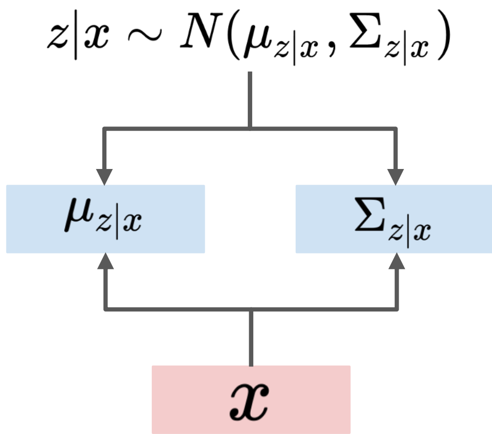


Variational Autoencoders (VAE)

Train by maximize the **variational lower bound**.

$$\boxed{E_{z \sim q_\phi(z|x)}[\log p_\Theta(x|z)]} - \boxed{KL(q_\phi(z|x), p(z))}$$

1. The input is **encoded** as distribution over the latent space

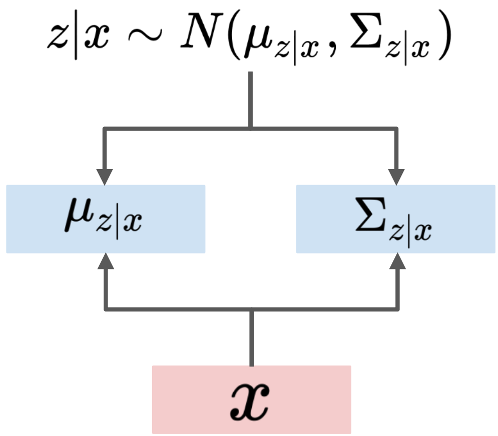


Variational Autoencoders (VAE)

Train by maximize the **variational lower bound**.

$$\boxed{E_{z\sim q_\phi(z|x)}[\log p_\Theta(x|z)]} - \boxed{KL(q_\phi(z|x), p(z))}$$

1. The input is **encoded** as distribution over the latent space
2. **Encoder output should match prior p(z)**

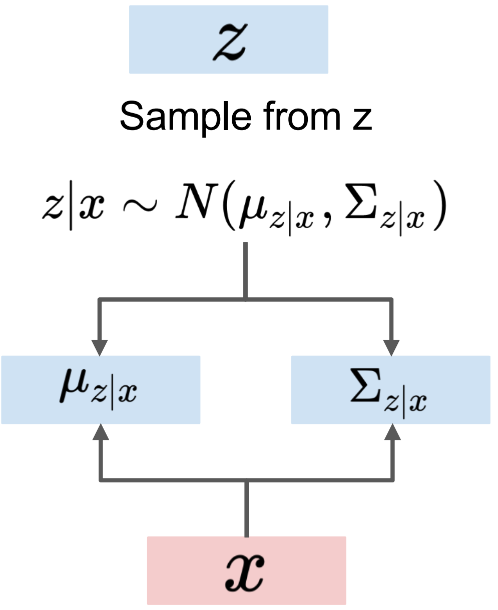


Variational Autoencoders (VAE)

Train by maximize the **variational lower bound.**

$$\boxed{E_{z\sim q_\phi(z|x)}[\log p_\Theta(x|z)]} - \boxed{KL(q_\phi(z|x), p(z))}$$

1. The input is **encoded** as distribution over the latent space
2. **Encoder output should match prior p(z)**
3. A point from the latent space is sampled from that distribution

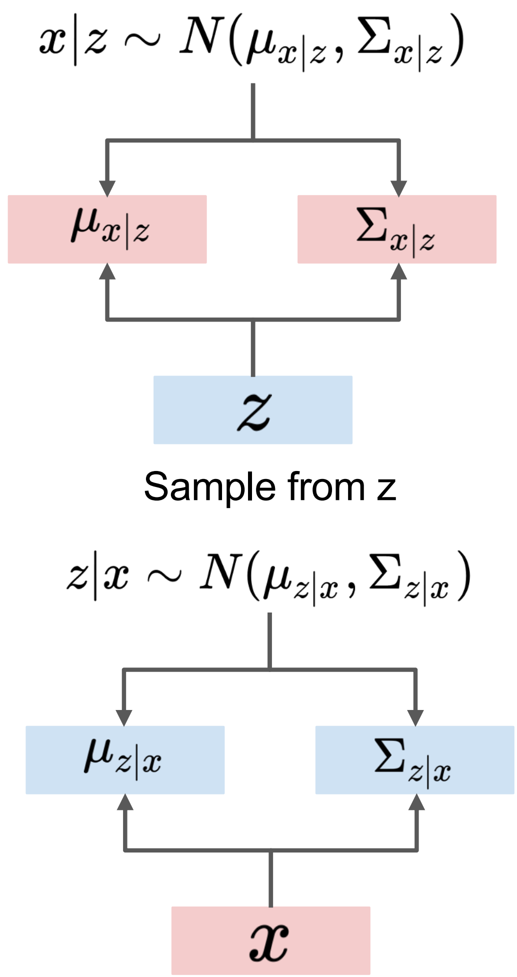


Variational Autoencoders (VAE)

Train by maximize the **variational lower bound.**

$$\boxed{E_{z\sim q_\phi(z|x)}[\log p_\Theta(x|z)]} - \boxed{KL(q_\phi(z|x), p(z))}$$

1. The input is **encoded** as distribution over the latent space
2. **Encoder output should match prior p(z)**
3. A point from the latent space is sampled from that distribution
4. The sampled point is **decoded**

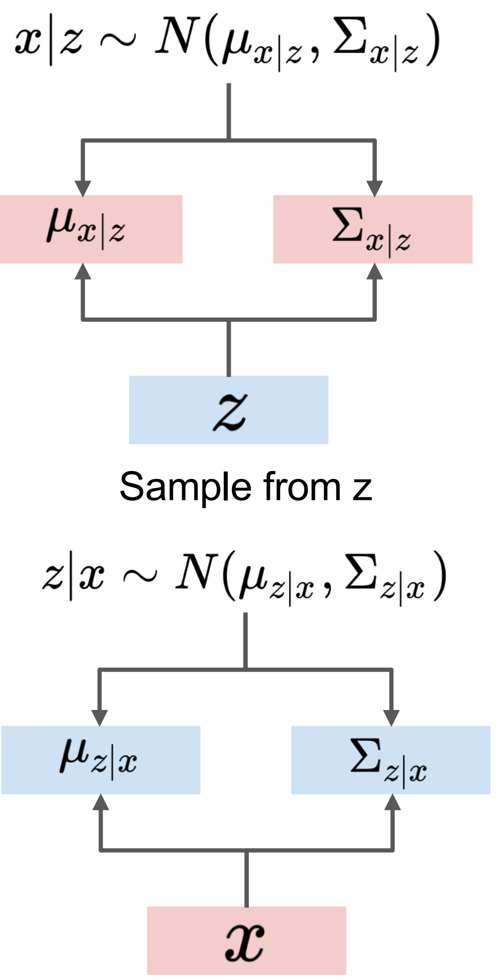


Variational Autoencoders (VAE)

Train by maximize the **variational lower bound.**

$$\boxed{E_{z\sim q_\phi(z|x)}[\log p_\Theta(x|z)]} - \boxed{KL(q_\phi(z|x), p(z))}$$

1. The input is **encoded** as distribution over the latent space
2. **Encoder output should match prior p(z)**
3. A point from the latent space is sampled from that distribution
4. The sampled point is **decoded**
5. **The reconstruction error is computed**



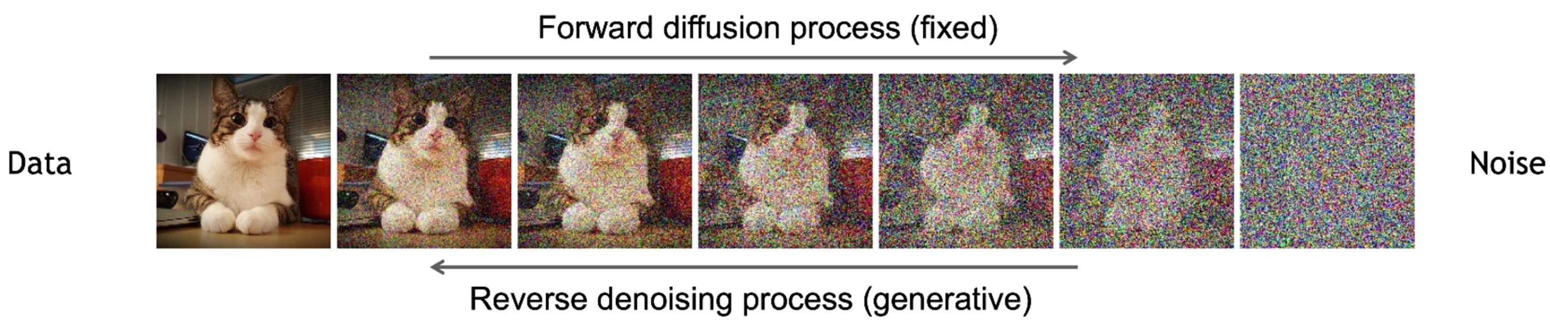
VAE results

Denoising Diffusion Models

Learning to generate by denoising

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

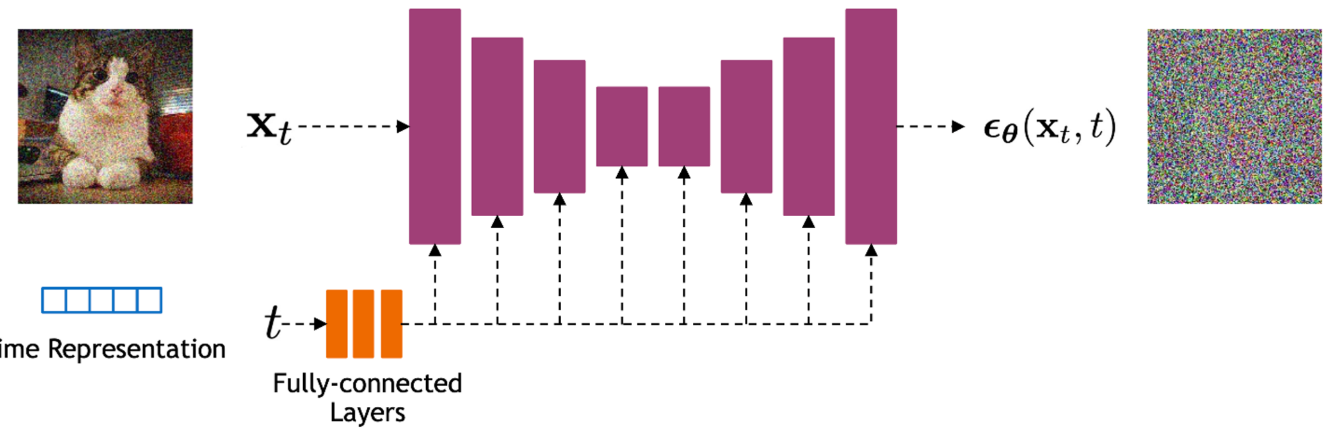


Sohl-Dickstein et al., Deep Unsupervised Learning using Nonequilibrium Thermodynamics, ICML 2015
Ho et al., Denoising Diffusion Probabilistic Models, NeurIPS 2020
Song et al., Score-Based Generative Modeling through Stochastic Differential Equations, ICLR 2021

Implementation Considerations

Network Architectures

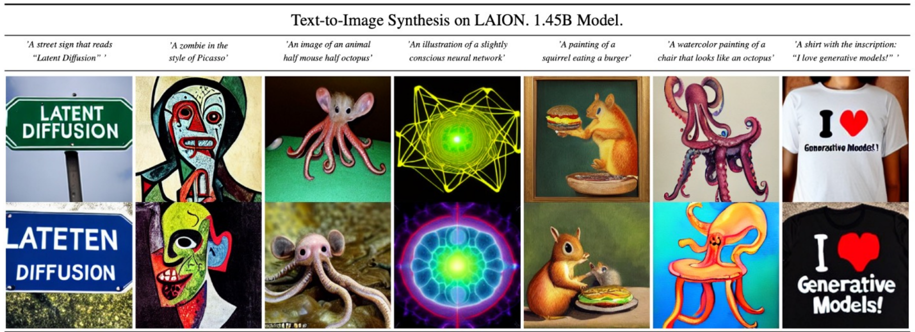
Diffusion models often use U-Net architectures with ResNet blocks and self-attention layers to represent $\boldsymbol{\epsilon}_\theta(\mathbf{x}_t, t)$



Time representation: sinusoidal positional embeddings or random Fourier features.

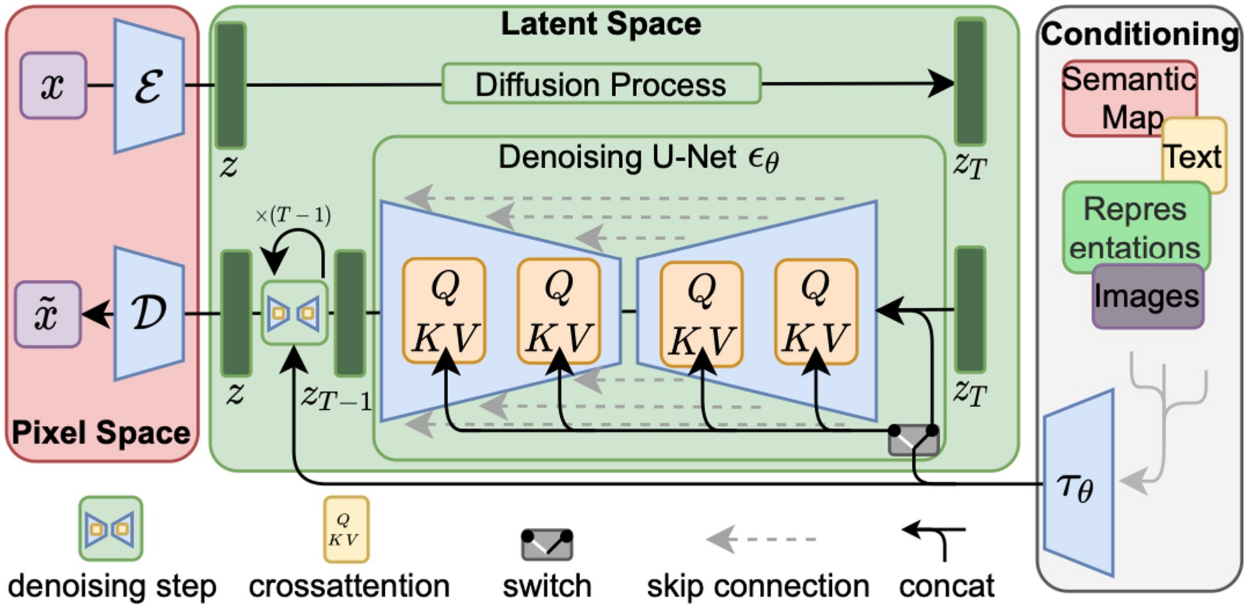
Time features are fed to the residual blocks using either simple spatial addition or using adaptive group normalization layers. (see Dharivwal and Nichol NeurIPS 2021)

Latent Diffusion Models



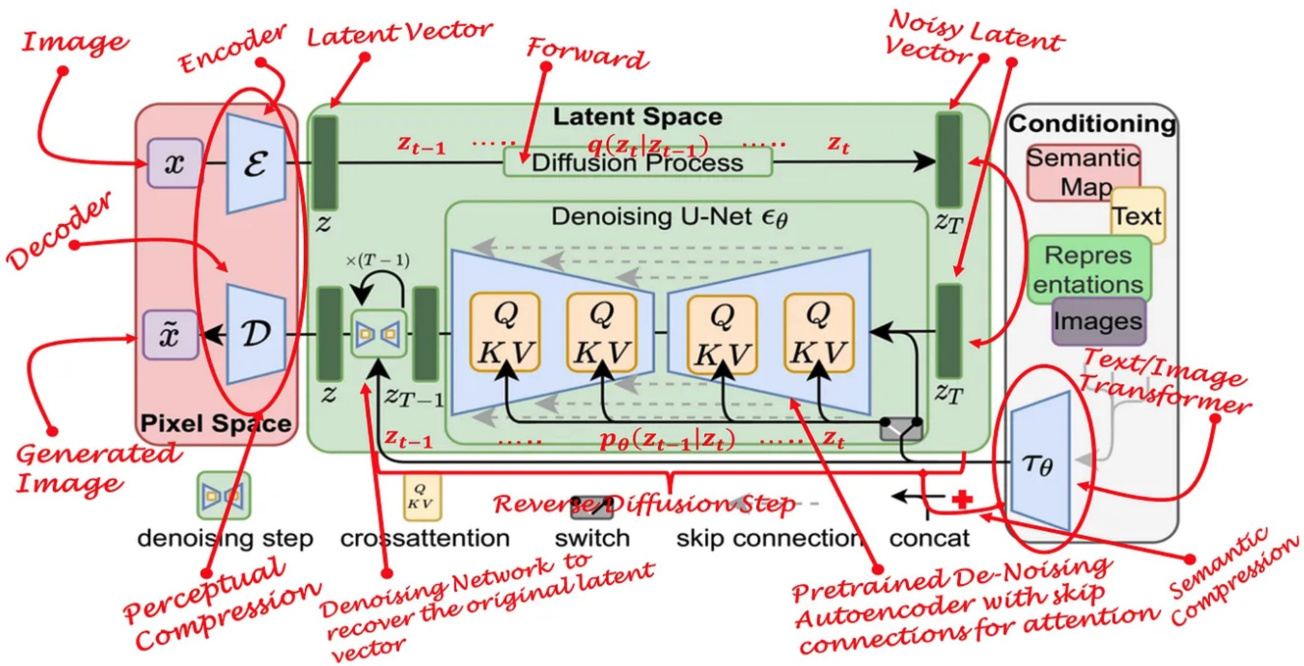
Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022

Latent Diffusion Models



Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022

Latent Diffusion Models



[Rombach et al. High-Resolution Image Synthesis with Latent Diffusion Models. CVPR 2022]

Generative models evaluation

Generative Models Evaluation

Which of these images looks better?

Generative Models Evaluation

Which of these images looks more realistic?

Generative Models Evaluation

Which of these images appears to be more similar to the text prompt?

Prompt: The saying "BE EXCELLENT TO EACH OTHER" written on a red brick wall with a graffiti image of a green alien wearing a tuxedo. A yellow fire hydrant is on a sidewalk in the foreground.

Generative Models Evaluation

- Human-based ratings and preference judgments
- Inception Score (quality and diversity) [1]
- Frechet Inception Distance [2]

[1] Salimans ey al. Improved Techniques for Training GANs. NeurIPS 2016
[2] Heusel et al. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017

Inception Score (IS)

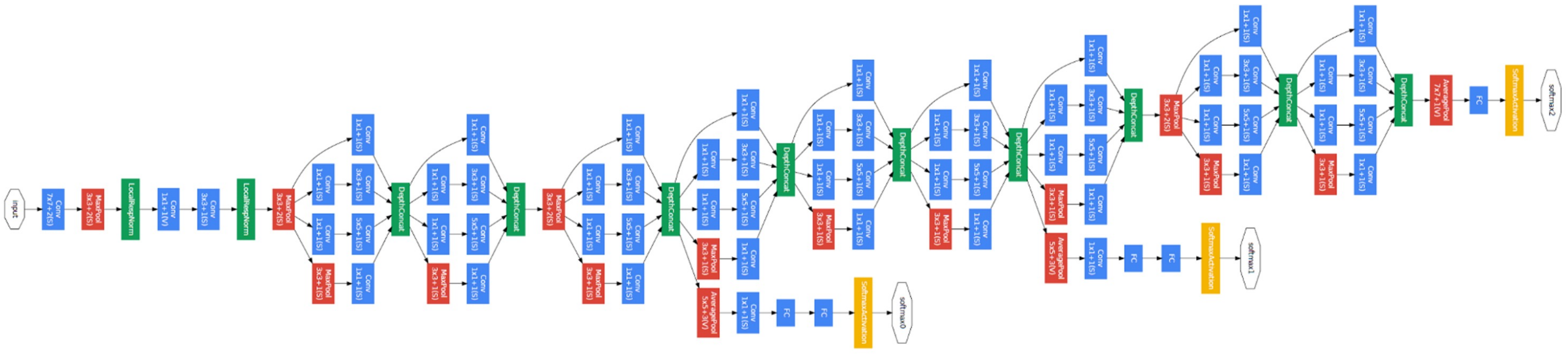
IS measures:

- the **quality** of the generated images
- their **diversity**

Salimans ey al. Improved Techniques for Training GANs. NeurIPS 2016

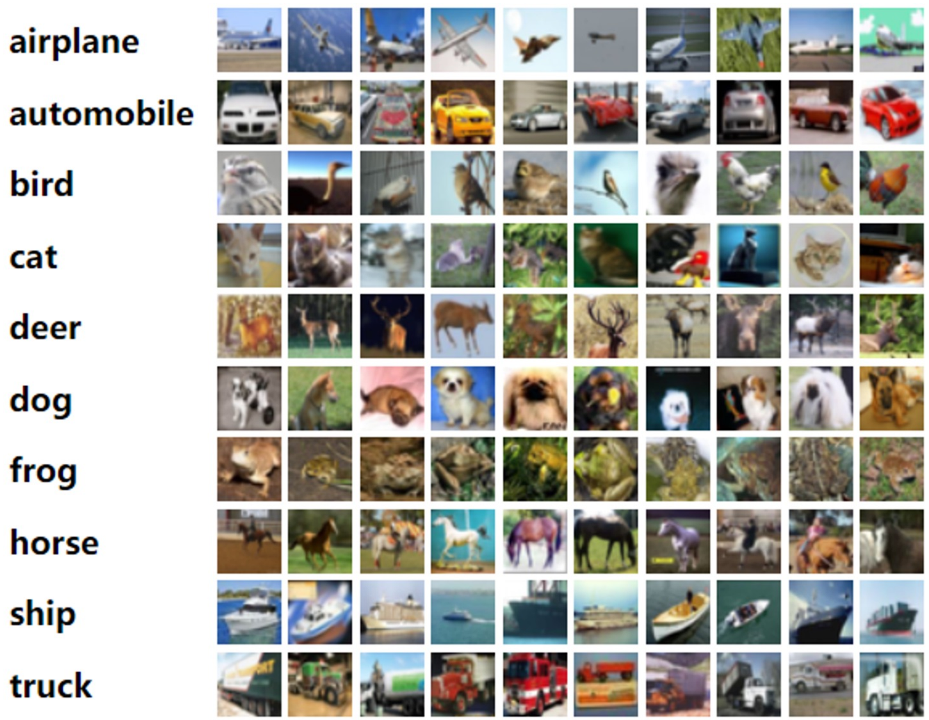
Inception Score (IS)

Inception image classifier pre-trained on CIFAR10



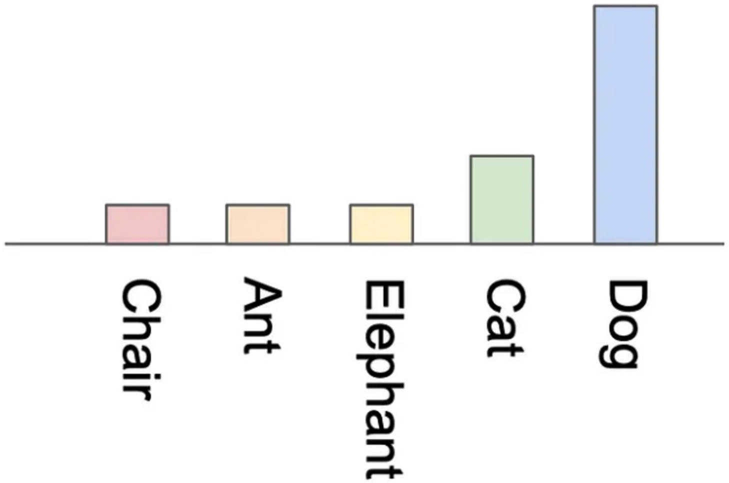
Inception Score (IS)

CIFAR10 dataset:



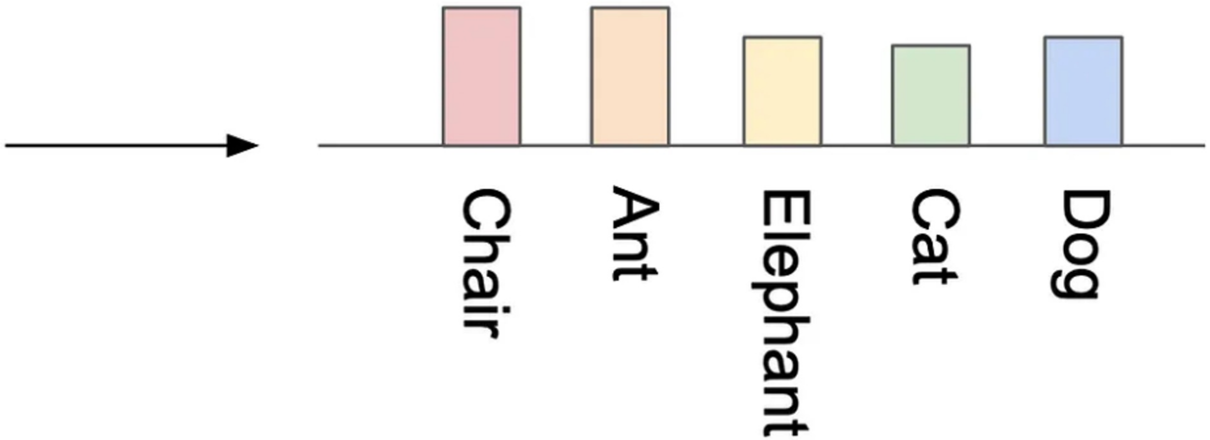
Inception Score (IS)

Generated images are fed into the Inception image classifier network pre-trained on the CIFAR10 dataset predict conditional probability p(y|x) — where y is the label and x is the generated data



Inception Score (IS)

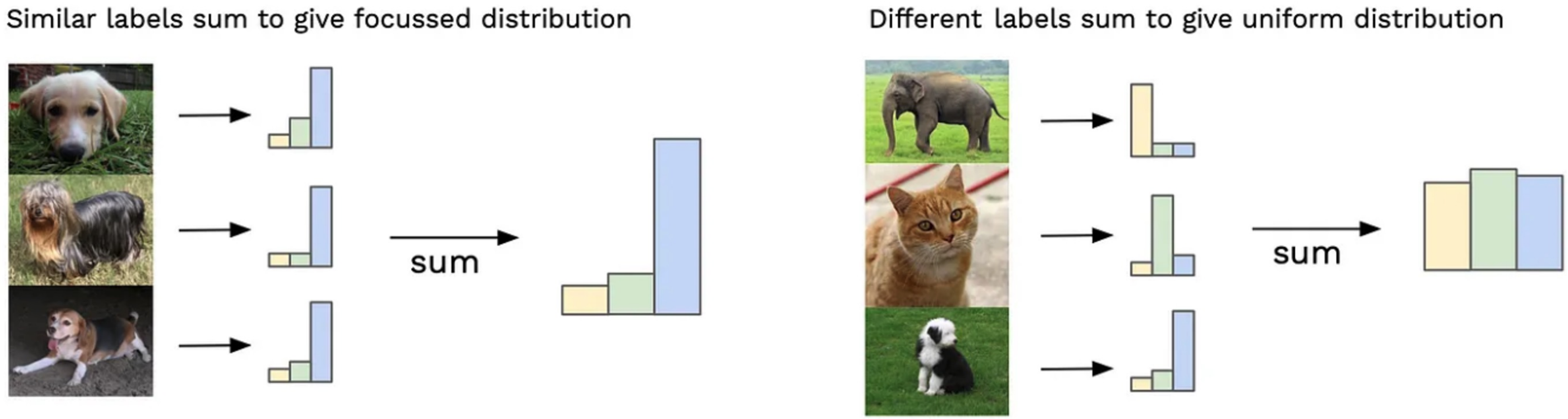
If the probability scores are widely distributed then the generated image is of low quality:



Inception Score (IS)

Calculate marginal probability $p(y) = \int_z p(y|x = G(z))dz$

Marginal distribution tells us how much variety there is in our generator's output.



Inception Score (IS)

- **Quality**: conditional probability **p(y|x)**
- **Diversity**: marginal probability **p(y)**

We want

- the conditional probability **p(y|x)** to be highly predictable (**low entropy**) i.e. given an image, we should know the object type easily.
- the marginal probability **p(y)** to be uniform (**high entropy**).

Inception Score (IS)

Compute their KL-divergence to combine these two criteria:

$$IS(G) = \exp(E_{x \sim p_g} KL(p(y|x) || p(y)))$$

Frechet Inception Distance (FID)

- Use the **Inception network** to extract features from an intermediate layer

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Frechet Inception Distance (FID)

- Use the **Inception network** to extract features from an intermediate layer
- Model data distribution for these features using a multivariate Gaussian distribution with mean μ and covariance Σ

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Frechet Inception Distance (FID)

- Use the **Inception network** to extract features from an intermediate layer
- Model data distribution for these features using a multivariate Gaussian distribution with mean μ and covariance Σ
- The FID between the real images x and generated images g:

$$FID(x, g) = ||\mu_x - \mu_g||_2^2 + Tr(\Sigma_x + \Sigma_g - 2(\Sigma_x \Sigma_g)^{\frac{1}{2}})$$

where *Tr* sums up all the diagonal elements

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Frechet Inception Distance (FID)

- **Lower** FID values mean **better** image quality and diversity
- FID is sensitive to mode collapse, the distance increases when modes are missed
- FID is more robust to noise than IS. If the model only generates one image per class, the distance will be high

Martin Heusel, Hubert Ramsauer, Thomas Unterthiner, Bernhard Nessler, Sepp Hochreiter. GANs Trained by a Two Time-Scale Update Rule Converge to a Local Nash Equilibrium. NeurIPS 2017 https://arxiv.org/abs/1706.08500

Thank you