DeepIntoDeep

Image Generation

발표자: 김승현

Image Generation

김승현

Artificial Intelligence in Korea University(AIKU)

Department of Computer Science and Engineering, Korea University

Image Generation

Introduction

Richard Feynman

“What I cannot create, I do not understand”

Generative Models

“What I understand, I can create”

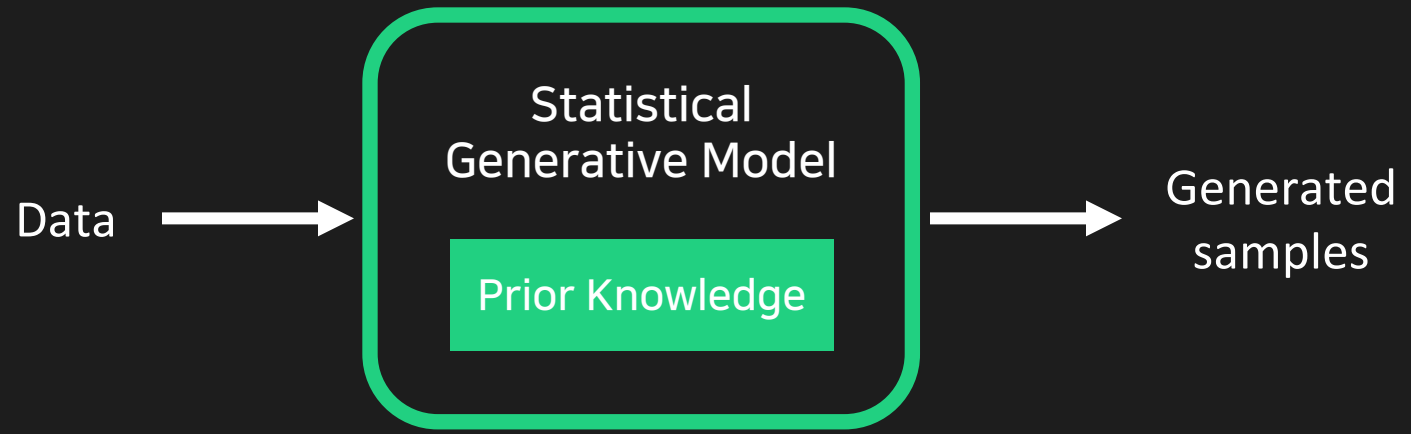
Statistical Generative Models

A statistical generative model requires data and prior knowledge

- Data – a set of samples (e.g., images of bedrooms)
- Prior knowledge – parameteric form (e.g., Gaussian), loss function (e.g., MLE), etc.

It learns from data, based on the prior knowledge

Once estimated, it can generate new data samples (similar to the original dataset)

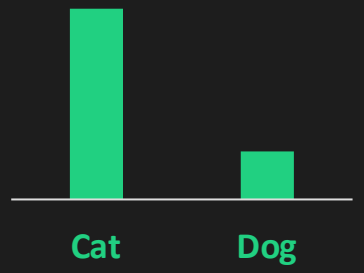


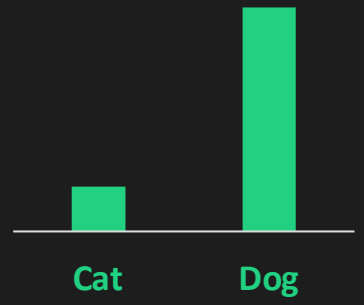
Discriminative vs. Generative

Discriminative Models

Discriminate between different kinds of data samples

Learn a probability distribution $p(y|x)$





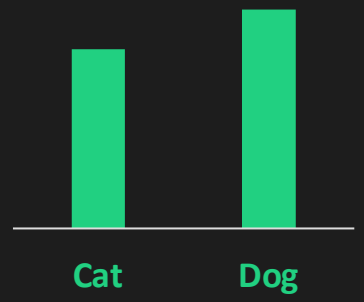
Discriminative vs. Generative

Discriminative Models

Discriminate between different kinds of data samples

Learn a probability distribution $p(y|x)$

Unable to handle unreasonable inputs



Discriminative vs. Generative

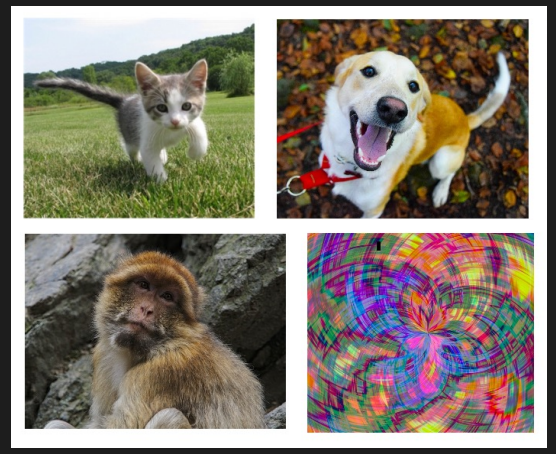
Generative Models

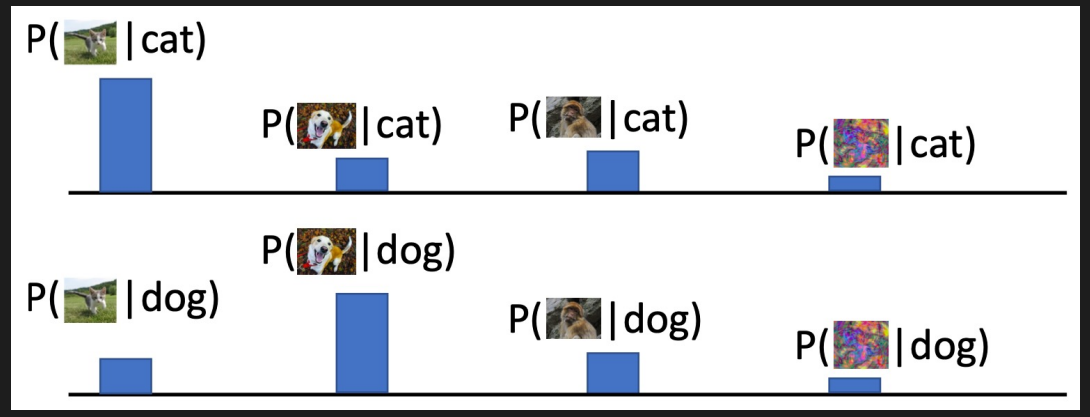
Generate new data samples

Learn a probability distribution $p(x|y)$ ($p(x)$ if there are no labels)

All possible images compete with each other for probability mass

Unreasonable inputs will take small values



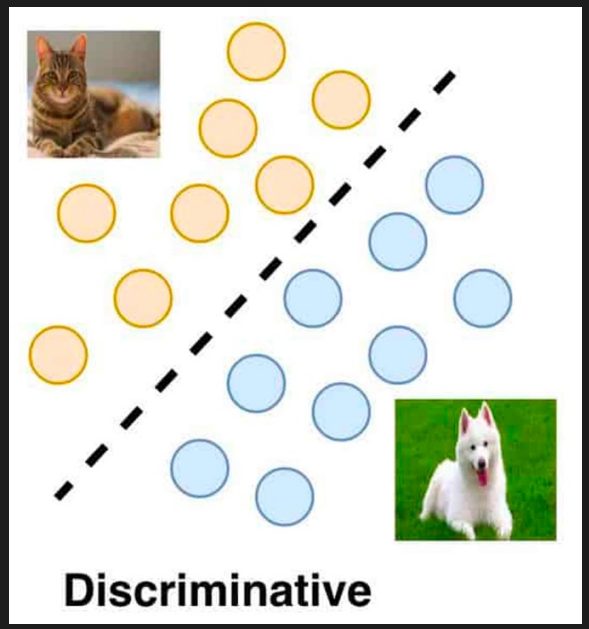


Generative Models are Hard

A discriminative model might focus on a few pattern

e.g., the difference between “sailboat” and “not sailboat”

It just tries to draw boundaries in the data space



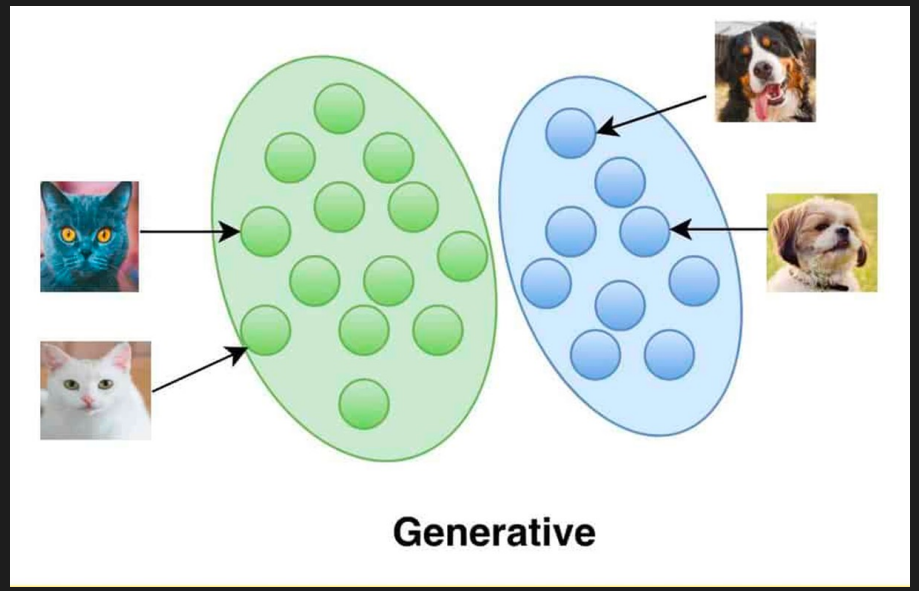
Generative Models are Hard

Generative models tackle a more difficult task than discriminative models

A generative model might capture abstract, high-level correlations

e.g., “things that look like boats are probably going to appear near things that look like water”

It tries to model how data is placed throughout the space



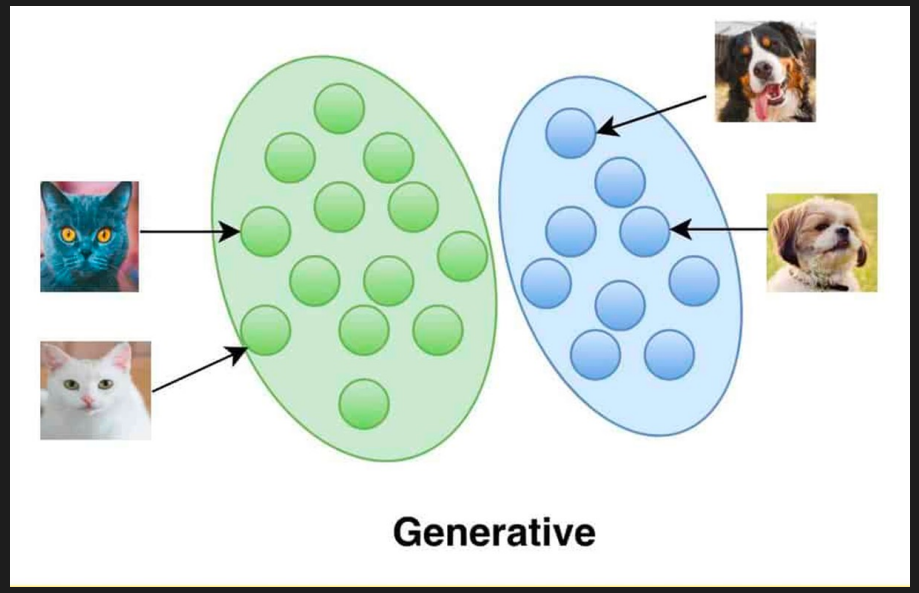
Generative Models are Hard

Generative models tackle a more difficult task than discriminative models

A generative model might capture abstract, high-level correlations

e.g., “things that look like boats are probably going to appear near things that look like water”

It tries to model how data is placed throughout the space

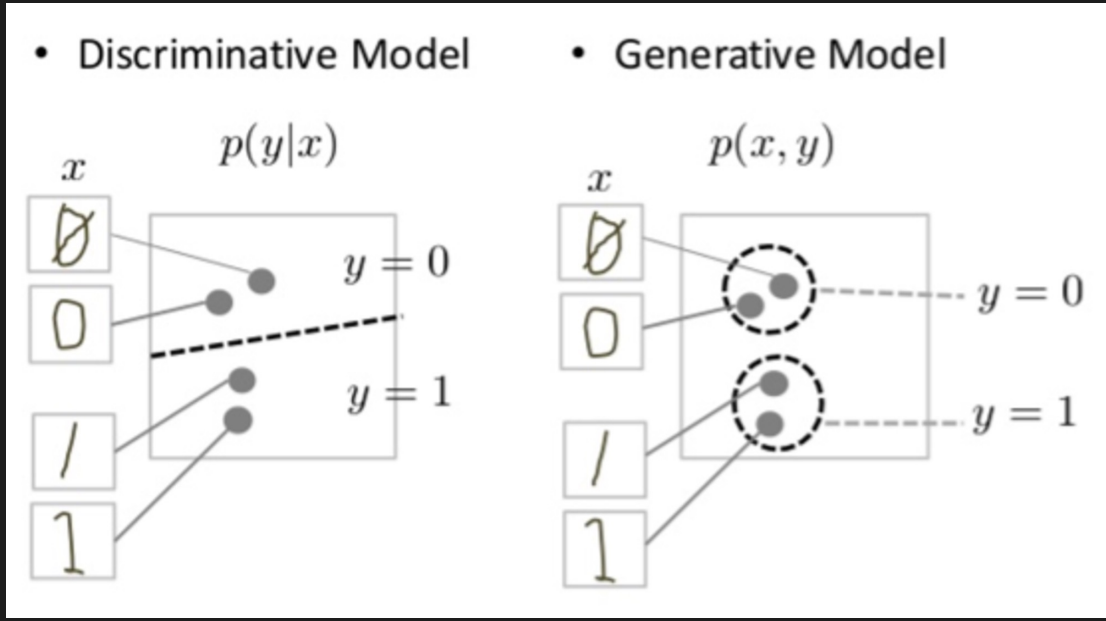


Generative Models are Hard: Example

A discriminate model draws a line in the data space

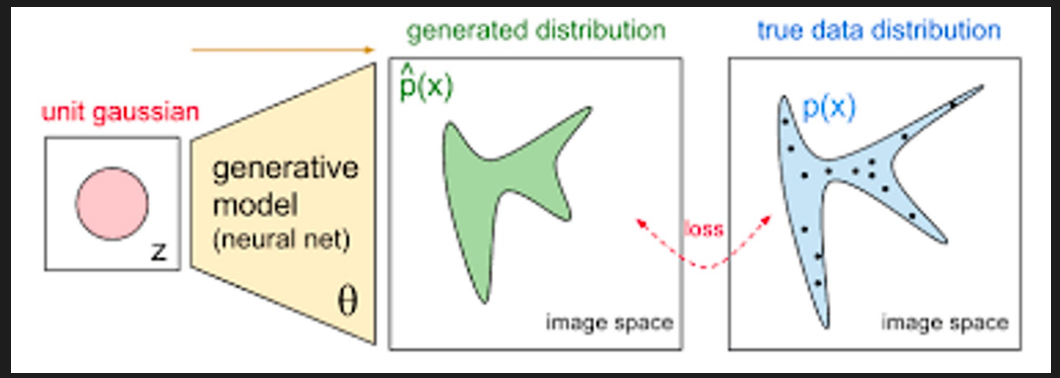
A generative model produces 1's and 0's images

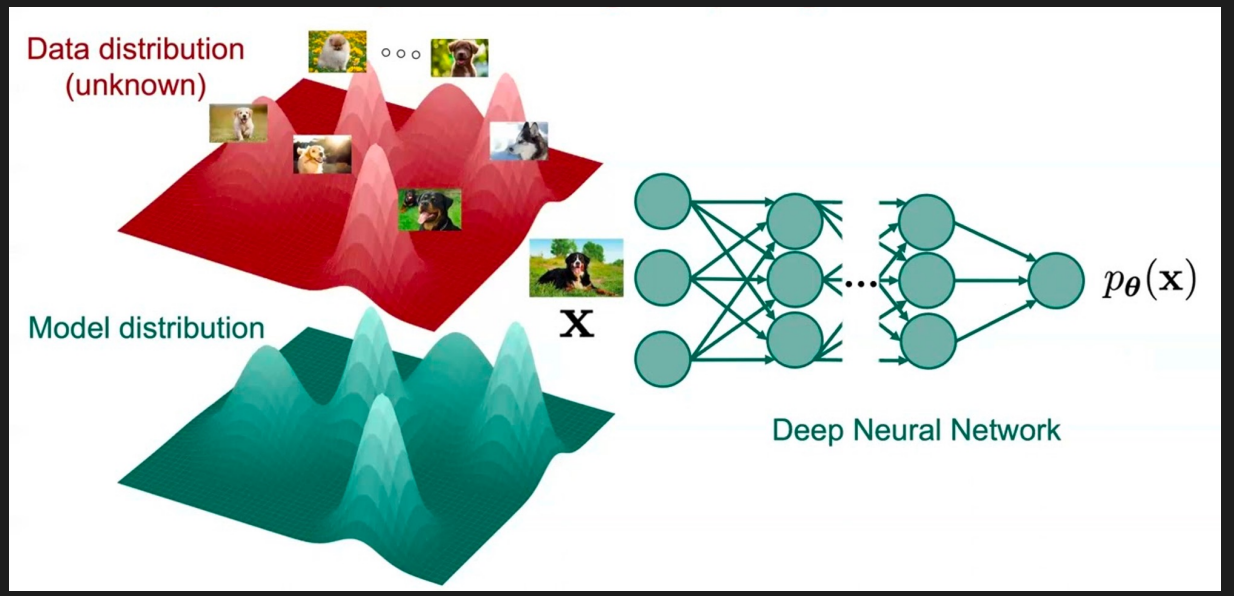
- Those images falls close to their counterparts in the data space
- It has to model the distribution throughout the data space



Modeling Data Space

Generative models model the distribution throught the data space

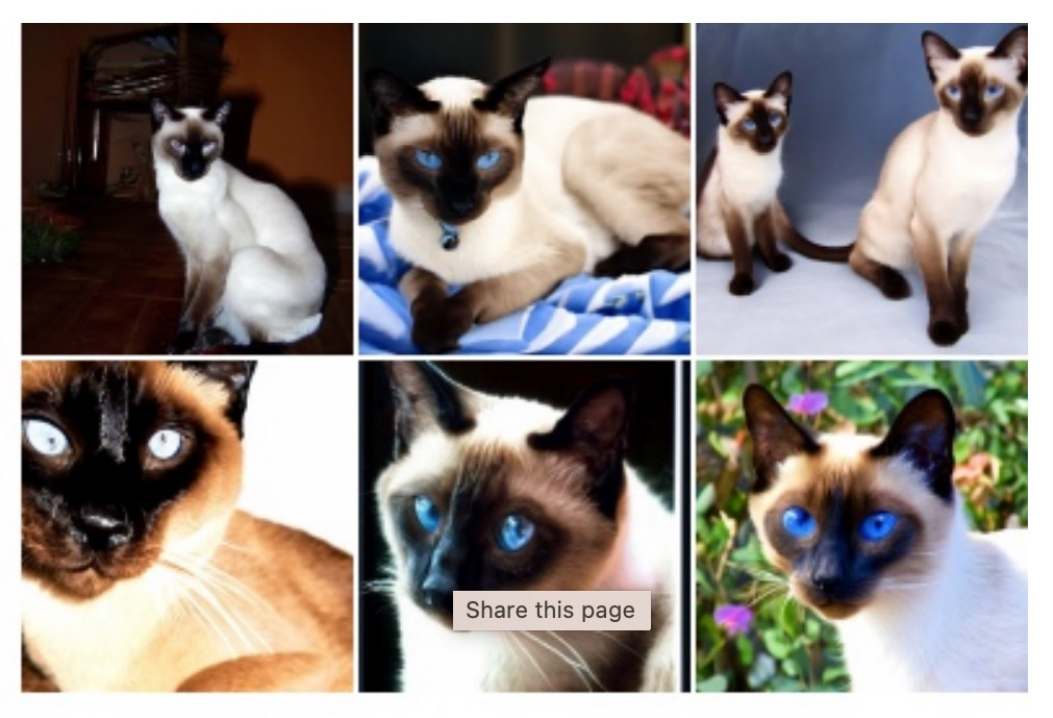


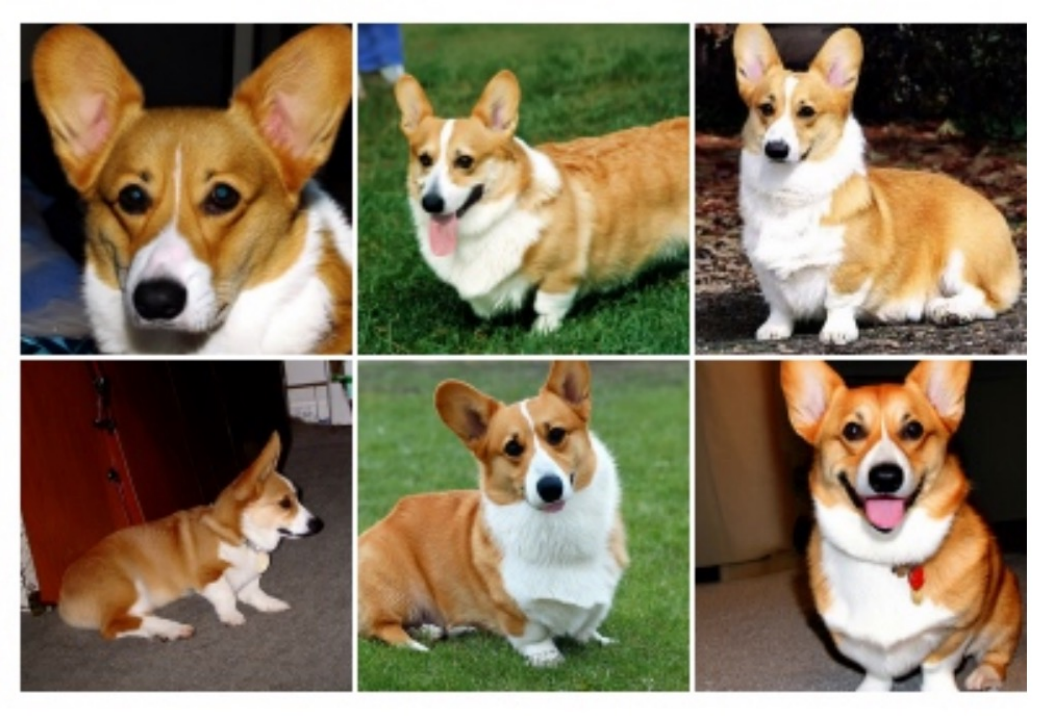


Applications

Applications

Conditional Image Generation (class)





https://arxiv.org/abs/2207.12598

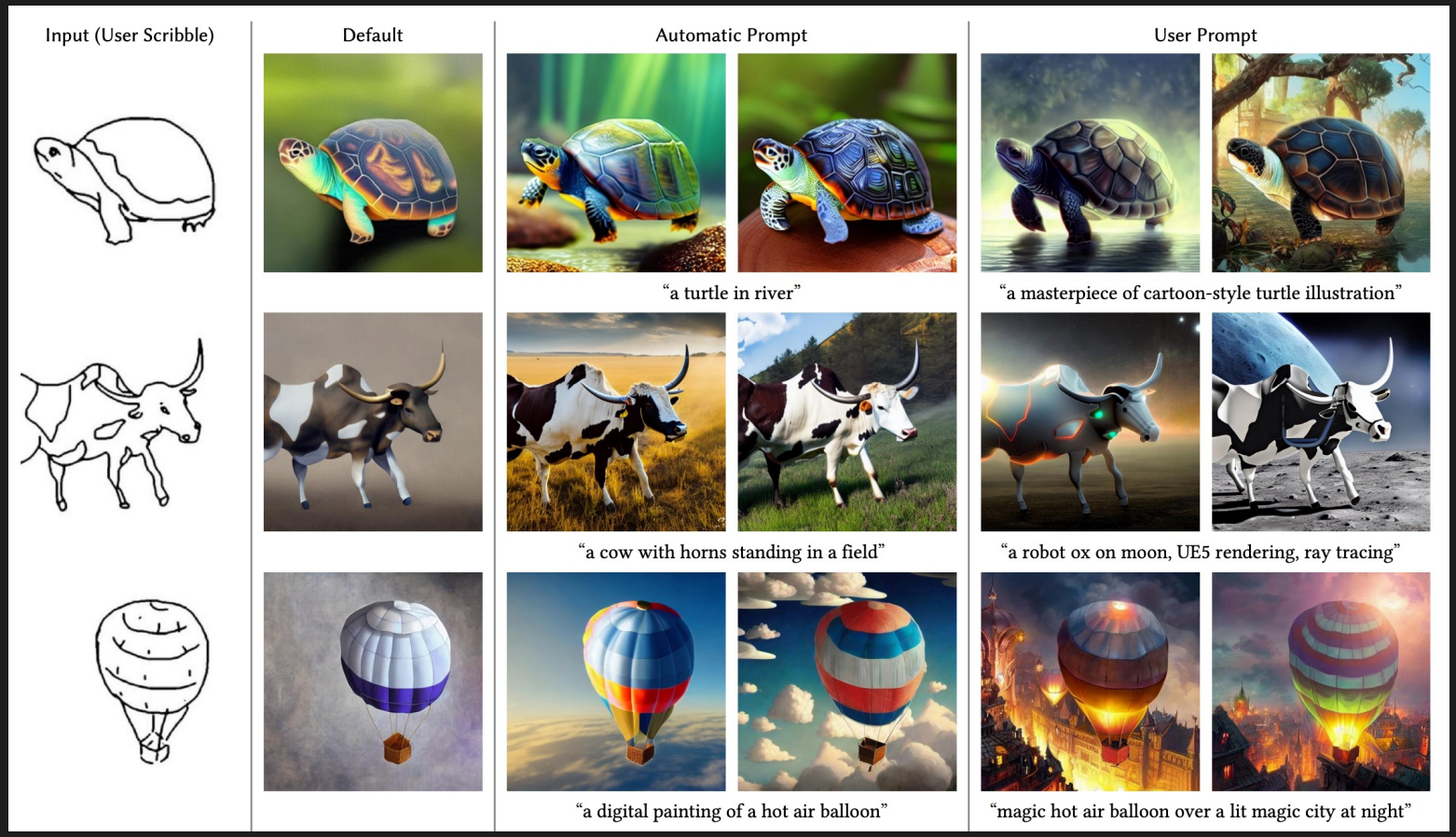
Applications

Conditional Image Generation (text)

https://arxiv.org/abs/2303.05511

Applications

Conditional Image Generation (image)



https://arxiv.org/abs/2302.05543

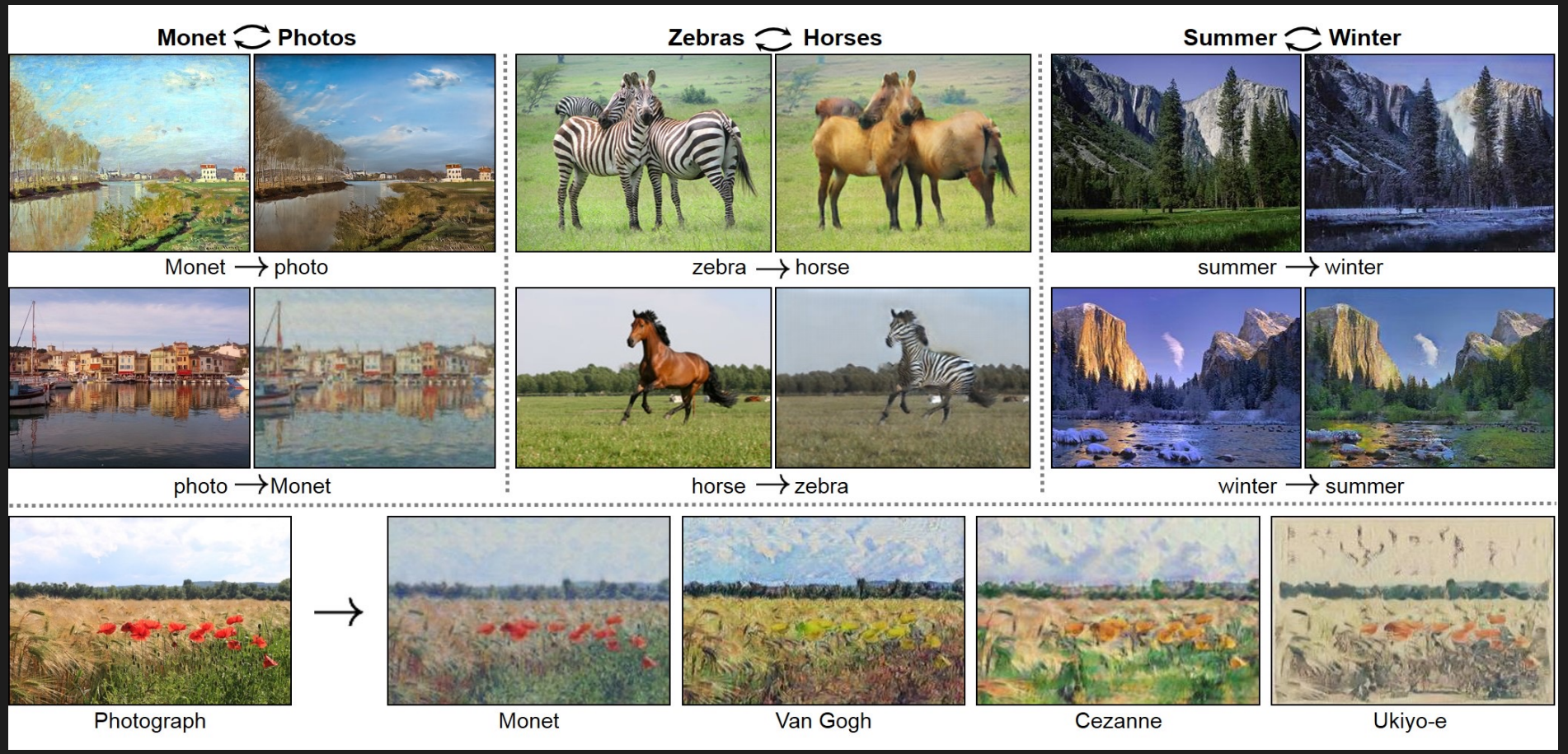
Applications

Generated Image Editing

https://arxiv.org/abs/2208.01626

Applications

Image Translation



https://arxiv.org/abs/1703.10593

Applications

Style-based Generation

https://arxiv.org/abs/1812.04948

Applications

Super Resolution

https://arxiv.org/abs/2303.05511

Evaluation

Inception Score

A measure of how realistic the generated output is

IS measures two things simultaneously:

1. The images have variety
   (e.g., each image is a different breed of dog)
2. Each image distinctly looks like something
   (e.g., one image is clearly a Poodle, the next is a French Bulldog)

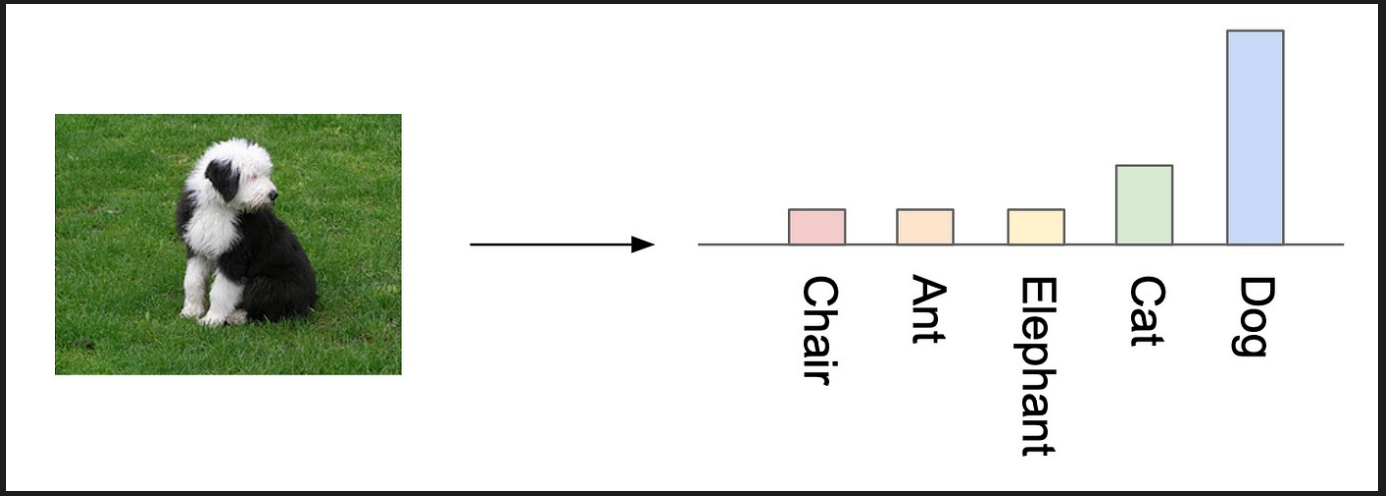
Evaluation

Inception Score

An Inception classifier takes images and returns probability distributions of labels

If the image contains one well-formed thing, the output is a narrow distribution

Otherwise the output will be closer to the uniform distribution



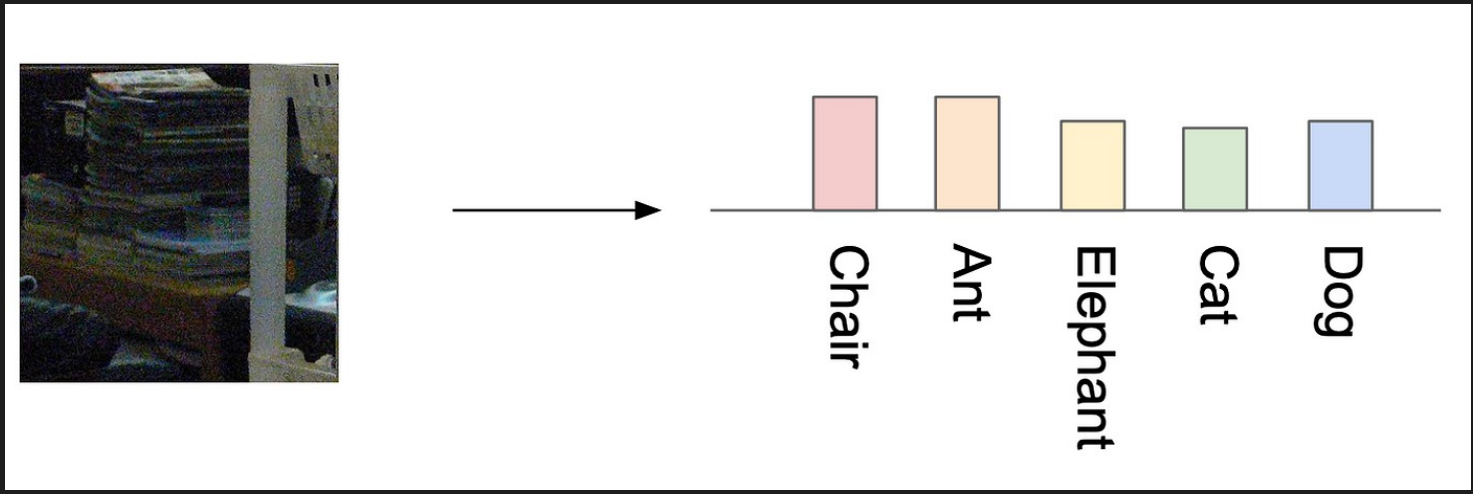
Evaluation

Inception Score

An Inception classifier takes images and returns probability distributions of labels

If the image contains one well-formed thing, the output is a narrow distribution

Otherwise the output will be closer to the uniform distribution

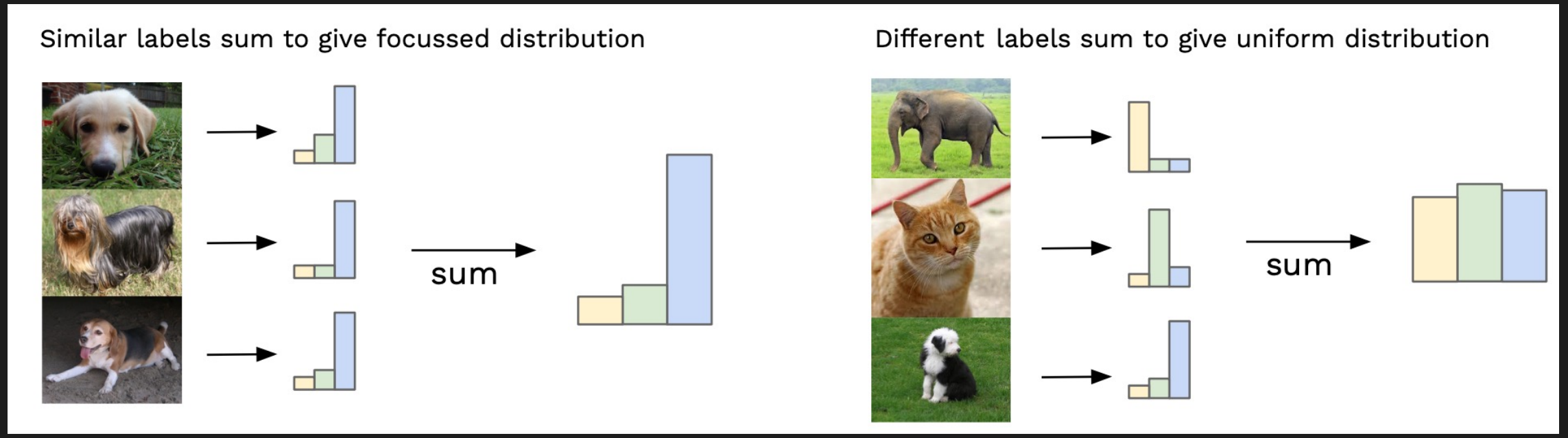


Evaluation

Inception Score

Combine the classifier outputs on the images

We want each image to each be distinct and to collectively have variety

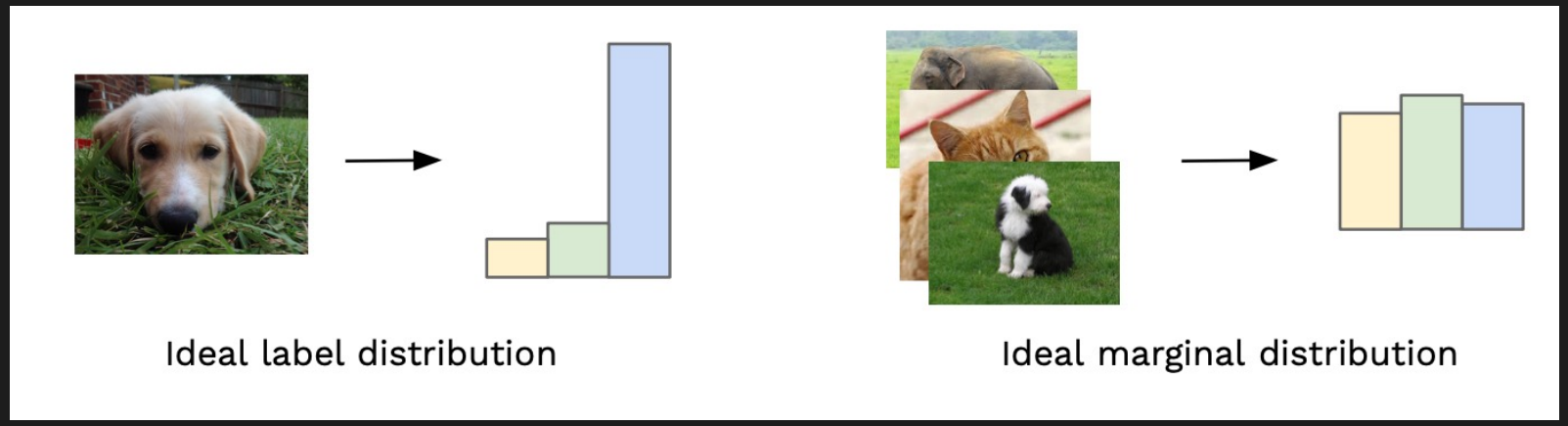


Evaluation

Inception Score

Combine the classifier outputs on the images

We want each image to each be distinct and to collectively have variety



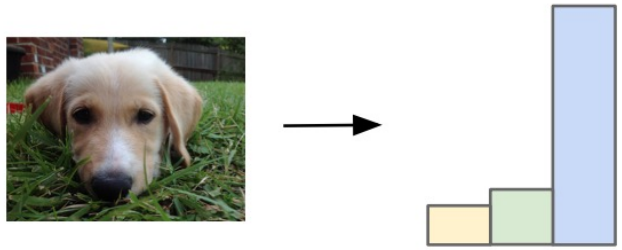
Evaluation

Inception Score

Sharpness (S)

- High sharpness implies classifier is confident its prediction
- That is, the classifier's predictive distribution $c(\mathrm{y}|\mathrm{x})$ has low entropy

$$S = \exp\left(E_{\mathbf{x}\sim p}\left[\int c(y|\mathbf{x}) \log c(y|\mathbf{x}) \mathrm{d}y\right]\right)$$



Ideal label distribution

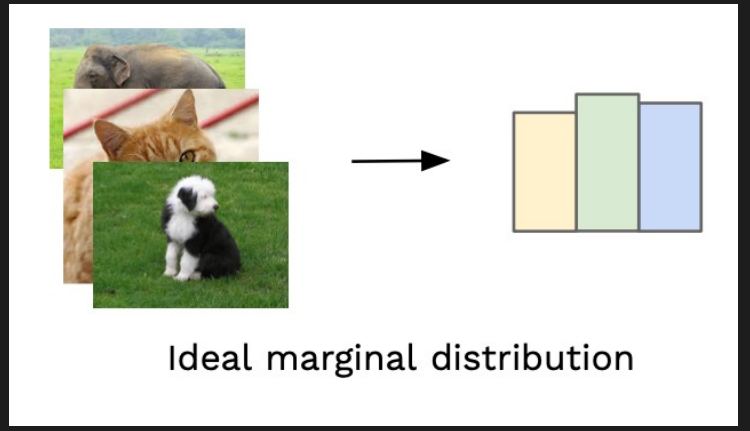
Evaluation

Inception Score

Diversity (D)

- High diversity implies the classifier's marginal distribution $c(\mathrm{y})$ has high entropy

$$D = \exp\left(-E_{\mathbf{x}\sim p}\left[\int c(y|\mathbf{x}) \log c(y) \mathrm{d}y\right]\right)$$



Evaluation

Inception Score

Inception score combine the two criteria: IS = D X S

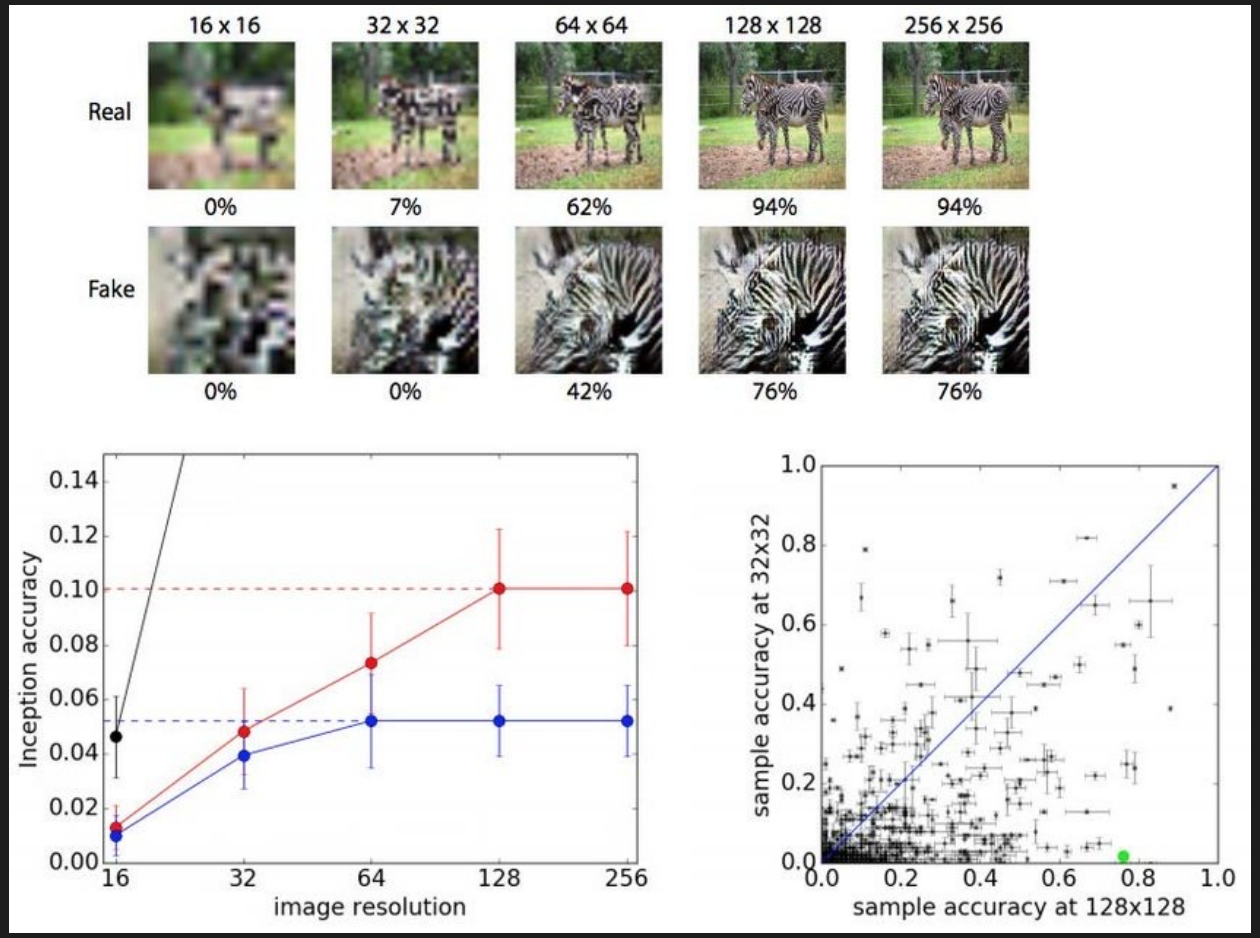
High IS corresponds to better quality

If a classifier is unavailable, use a classifier trained on a large dataset

e.g., Inception network trained on the ImageNet

Evaluation

Inception Score

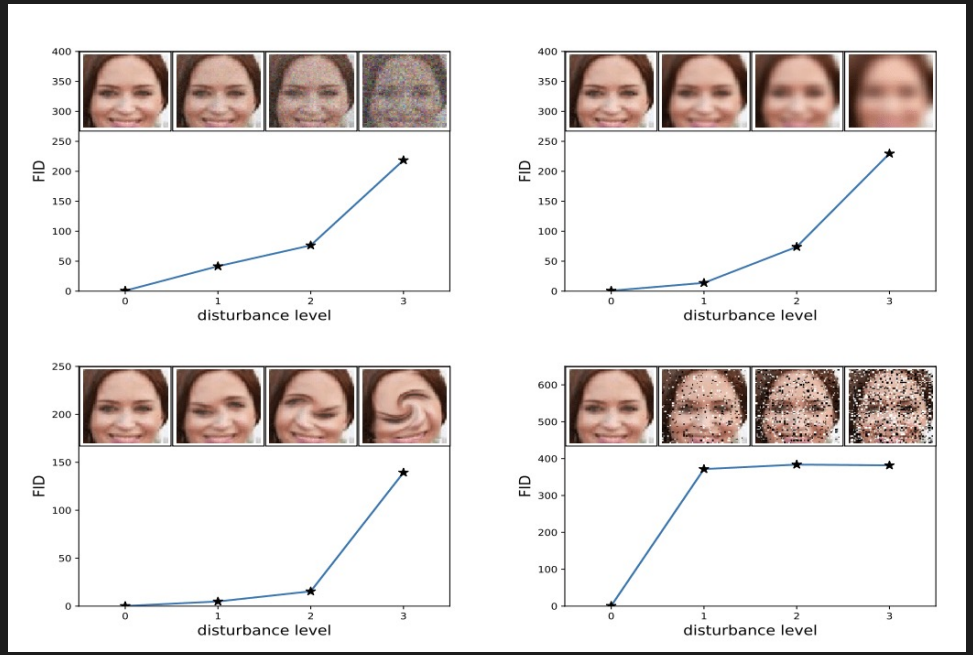


Evaluation

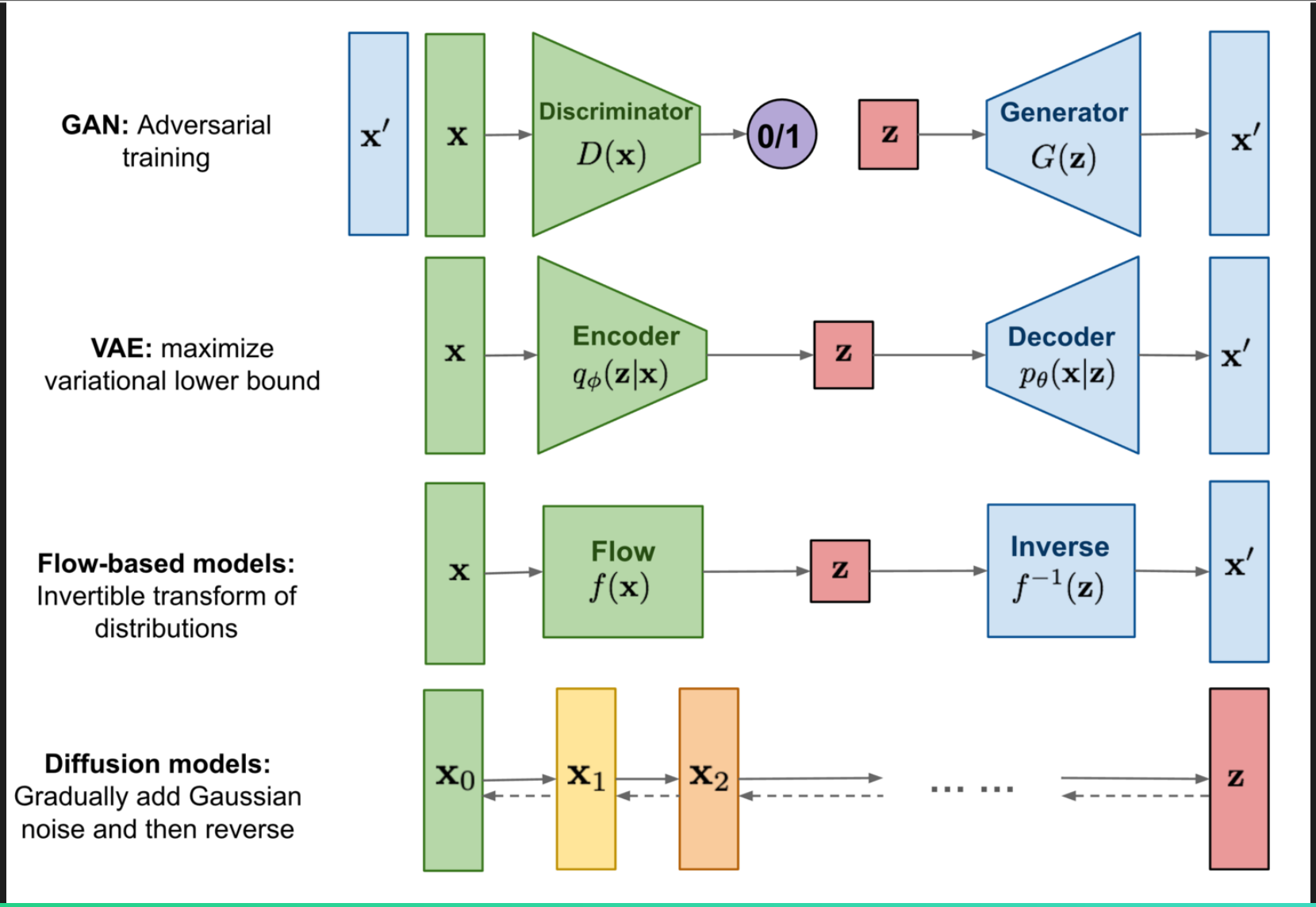
Frechet Inception Distance

IS does not take into account the desired data distribution

FID measures similarities in the feature representations for generated samples and those in the test dataset



Models for Image Generation



Autoregressive Models

Autoregressive Models

Given a data sample $x = (x_1, x_2, x_3, \cdots, x_T)$

Our model will be like

$$p(x) = p(x_1, x_2, x_3, \cdots, x_T)$$

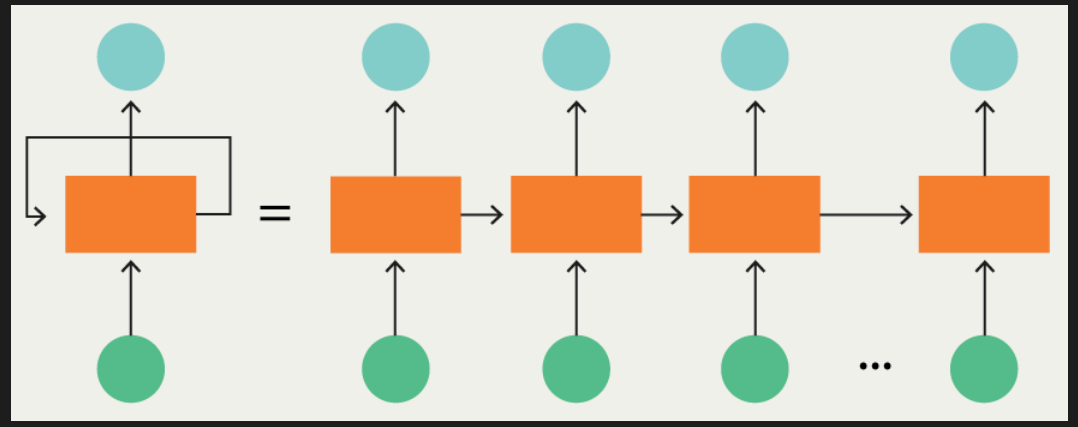
Using the chain rule $f(x, y) = f(x)f(y|x)$

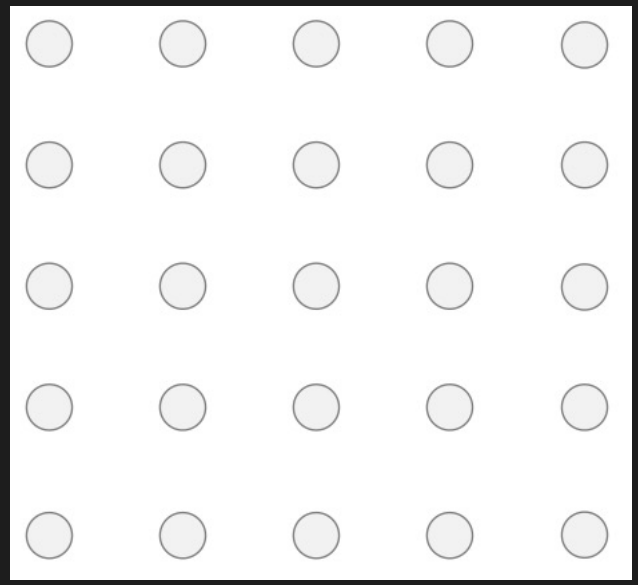
$$p(x_1, x_2, x_3, \cdots, x_T)$$
$$= p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \cdots p(x_T|x_1, x_2, x_3, \cdots, x_{T-1})$$
$$= \prod_{t=1}^{T} p(x_t|x_1, x_2, x_3, \cdots, x_{t-1})$$

PixelRNN

Starting at the upper left corner, generate one pixel at a time

Compute a hidden state for each pixel depending on hidden states and pixel values from the left and the above

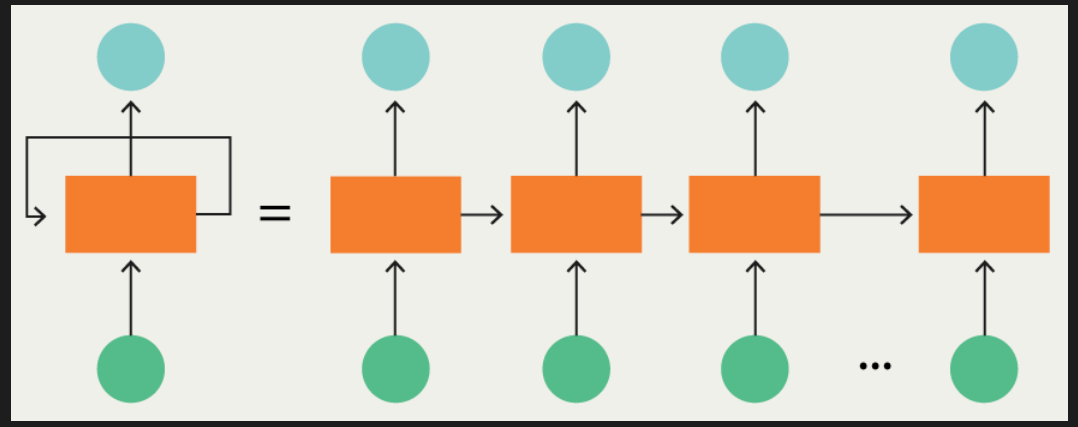


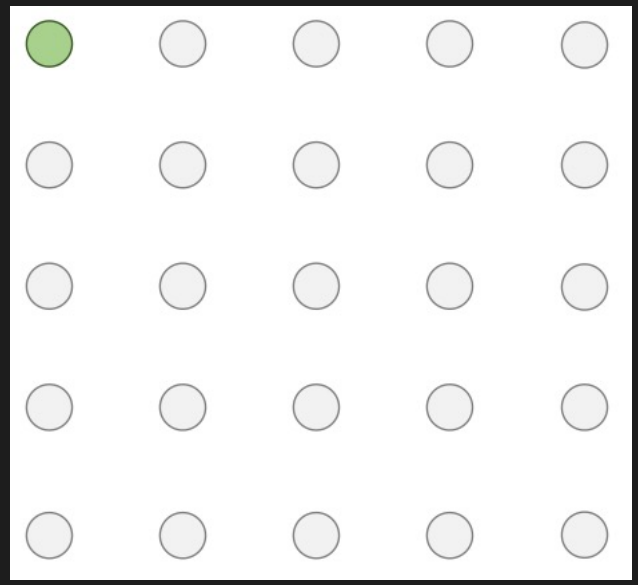


PixelRNN

Starting at the upper left corner, generate one pixel at a time

Compute a hidden state for each pixel depending on hidden states and pixel values from the left and the above

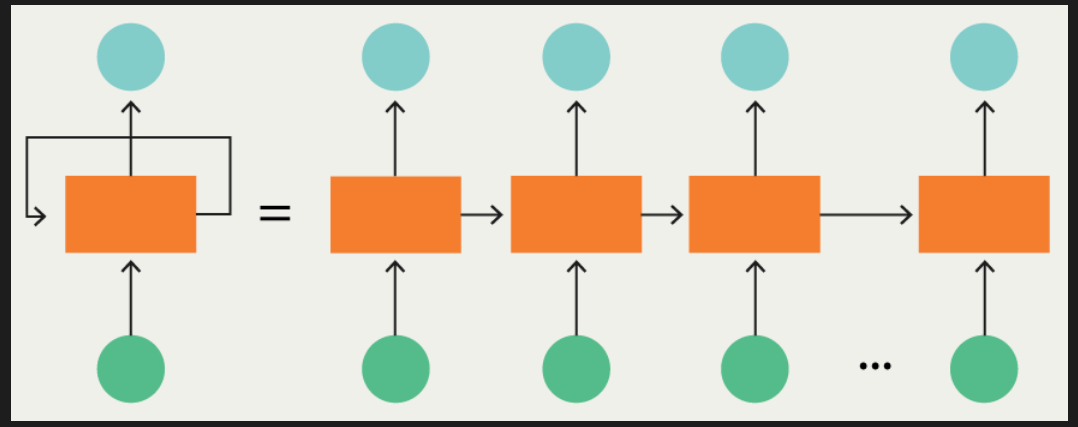


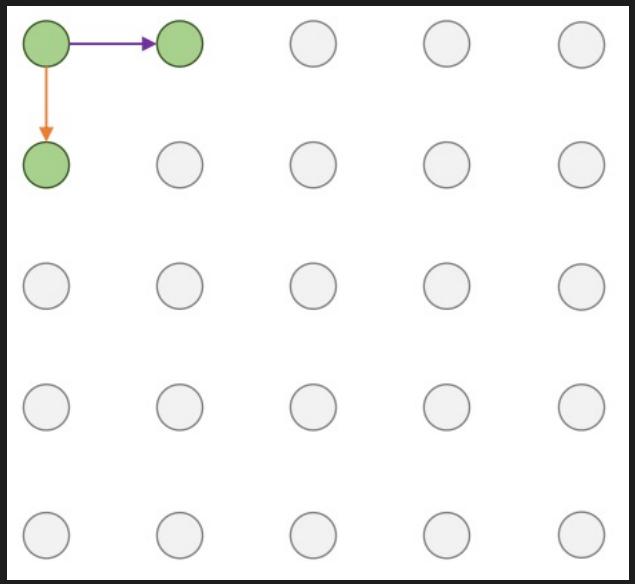


PixelRNN

Starting at the upper left corner, generate one pixel at a time

Compute a hidden state for each pixel depending on hidden states and pixel values from the left and the above

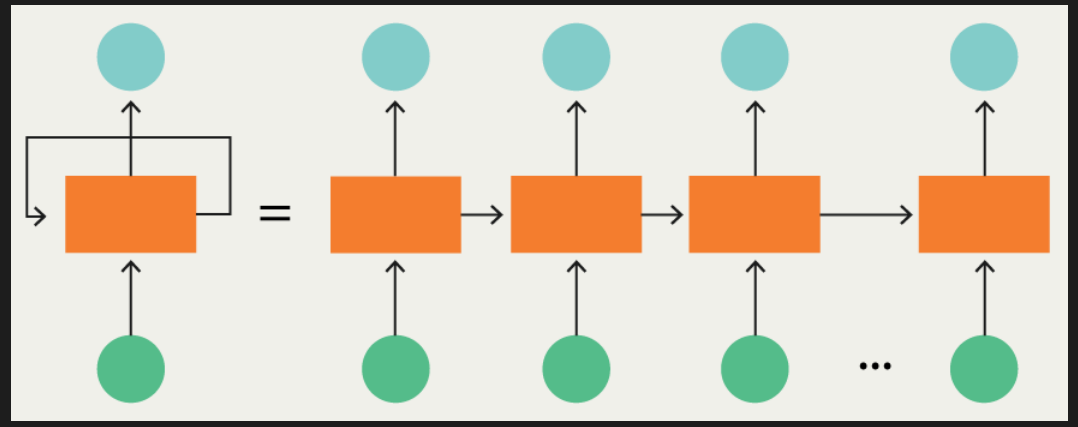


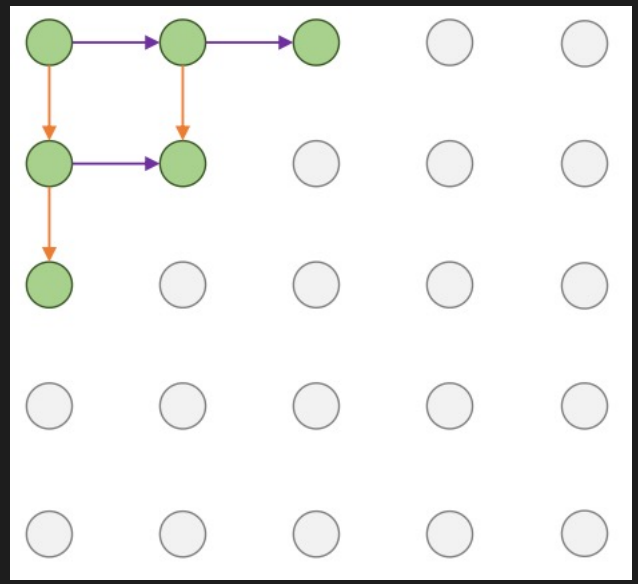


PixelRNN

Starting at the upper left corner, generate one pixel at a time

Compute a hidden state for each pixel depending on hidden states and pixel values from the left and the above

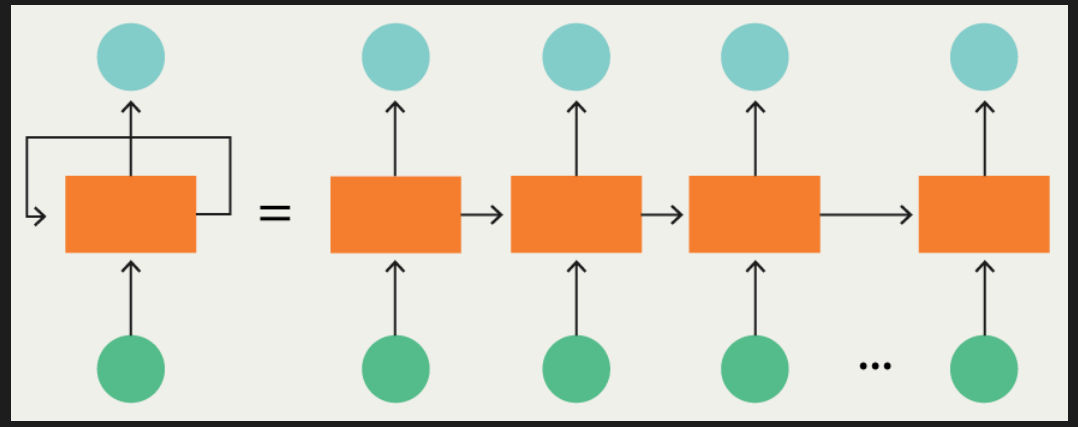


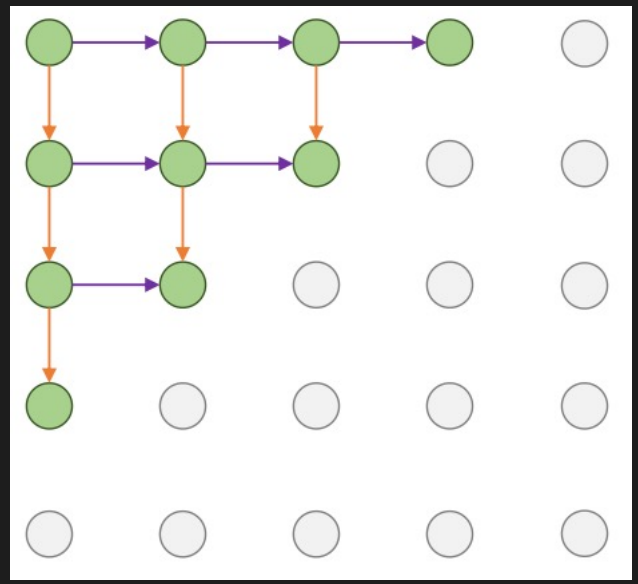


PixelRNN

Starting at the upper left corner, generate one pixel at a time

Compute a hidden state for each pixel depending on hidden states and pixel values from the left and the above

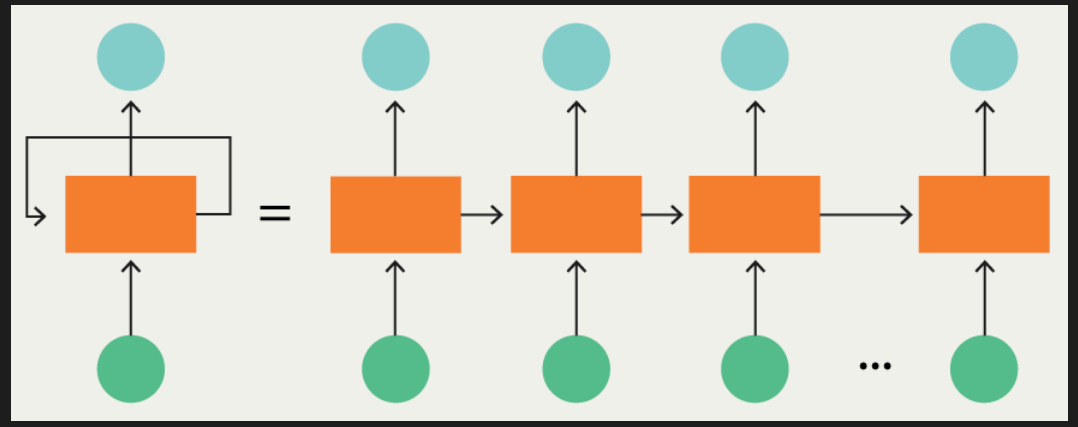


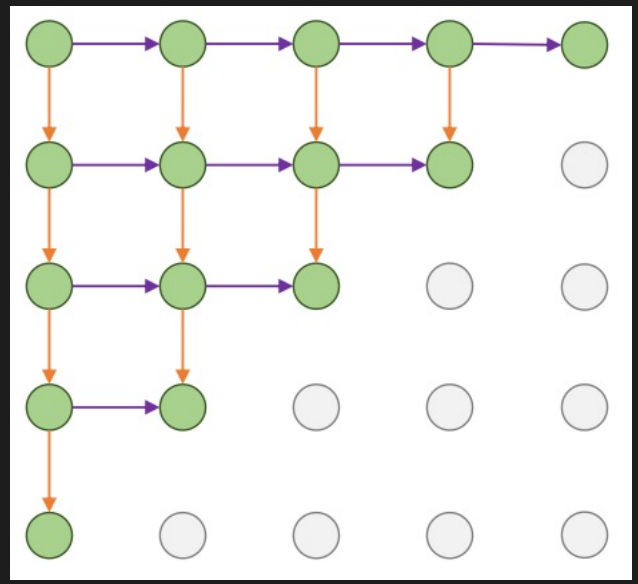


PixelRNN

Starting at the upper left corner, generate one pixel at a time

Compute a hidden state for each pixel depending on hidden states and pixel values from the left and the above

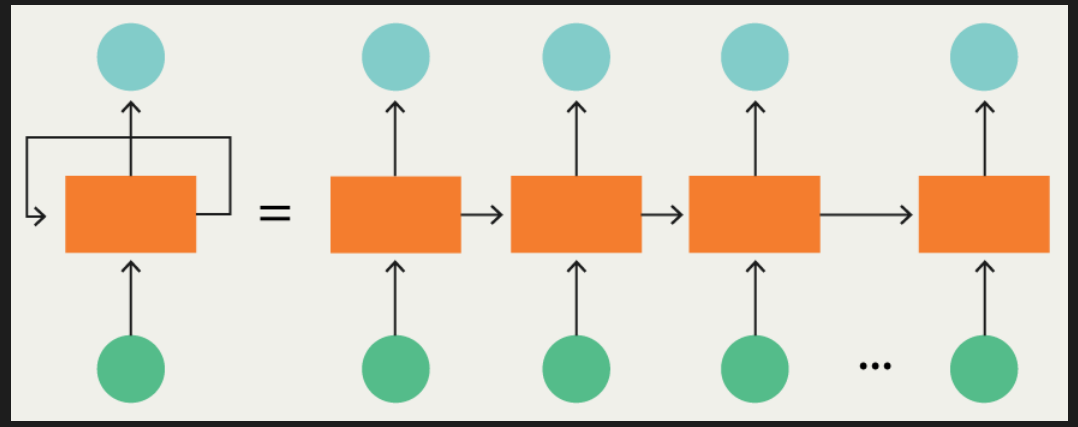


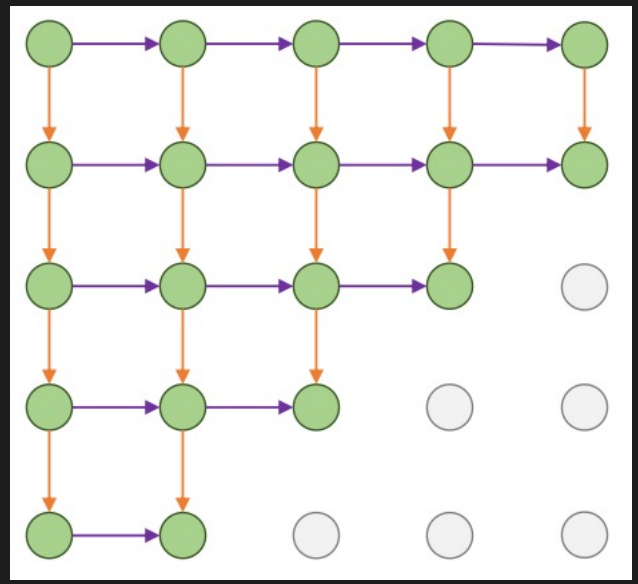


PixelRNN

Starting at the upper left corner, generate one pixel at a time

Compute a hidden state for each pixel depending on hidden states and pixel values from the left and the above



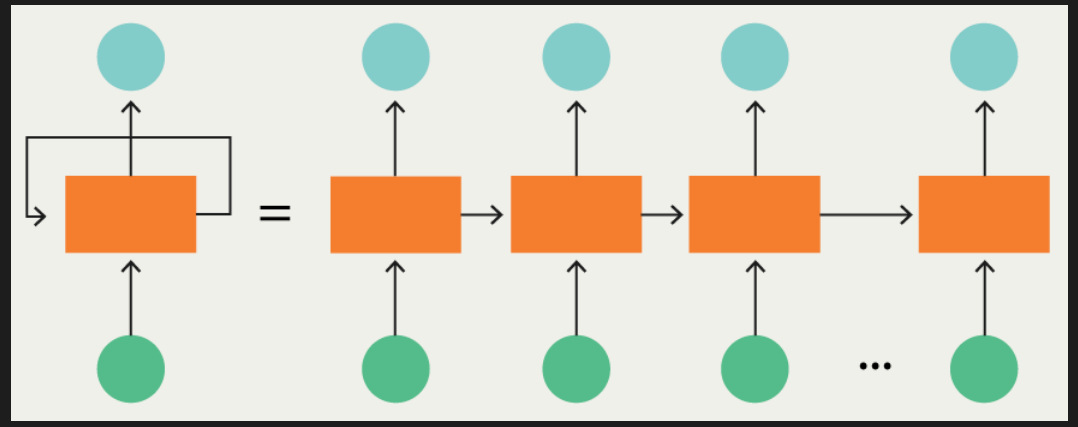


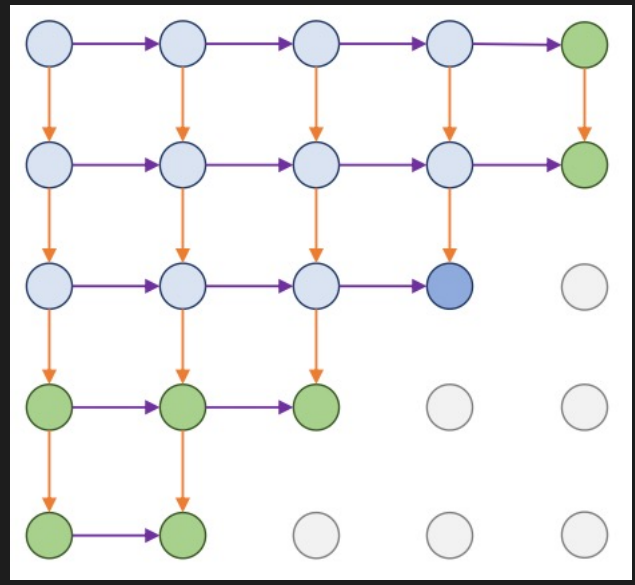
PixelRNN

Each pixel depends implicitly on all pixels above and to the left

But very slow in both training and testing

2N-1 sequential steps are required to generate a N x N image



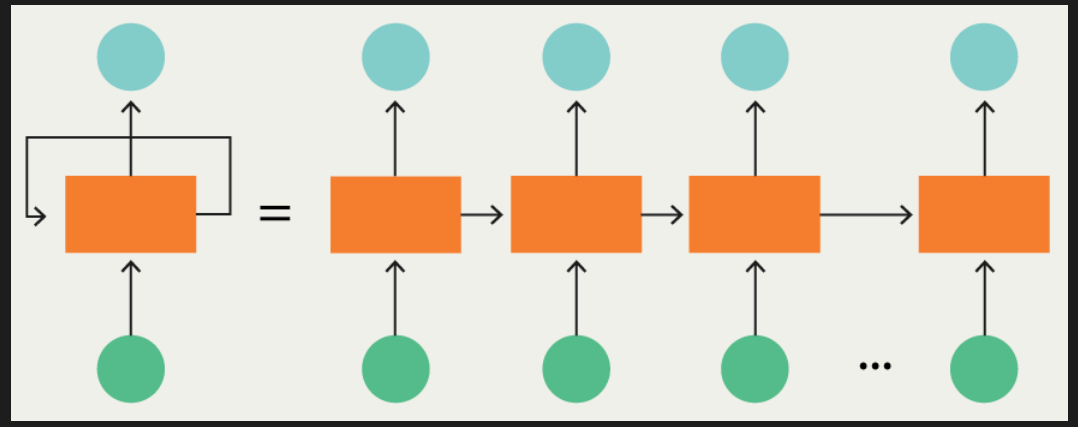


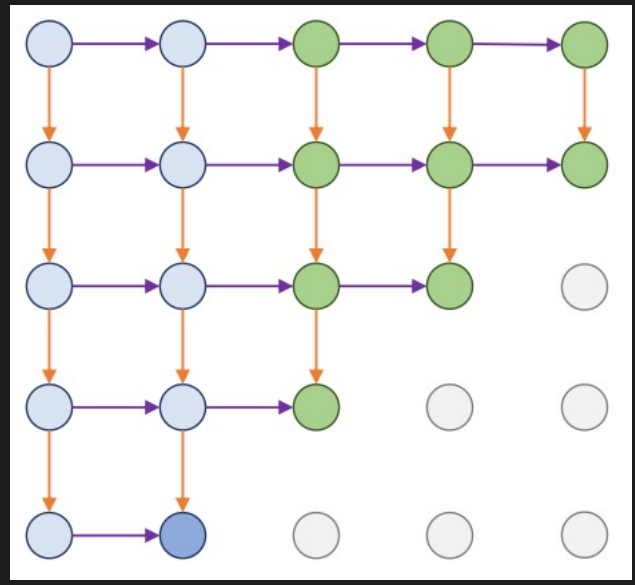
PixelRNN

Each pixel depends implicitly on all pixels above and to the left

But very slow in both training and testing

2N-1 sequential steps are required to generate a N x N image

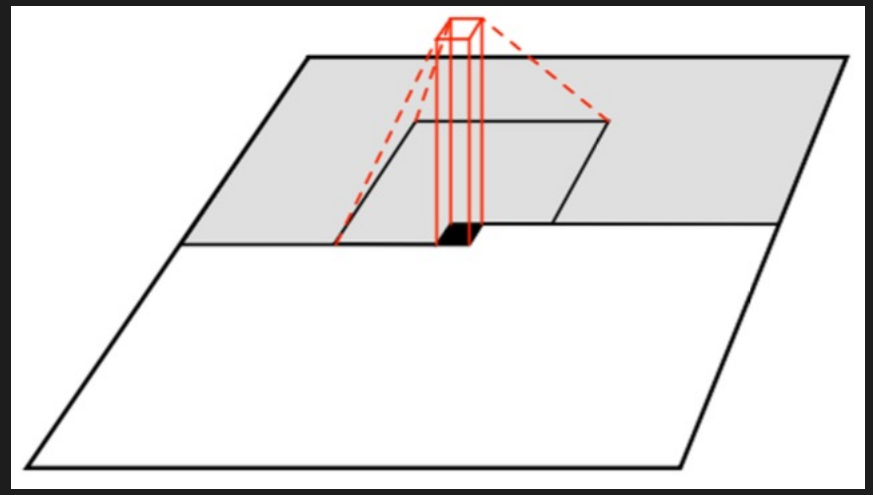




PixelCNN

Still generate one pixel starting from the corner

Model the dependency on previous pixels using CNN over a context region



PixelCNN

In training the convolutions can be parallelized; become faster

But generation must be done sequentially; still slow

32x32 CIFAR-10

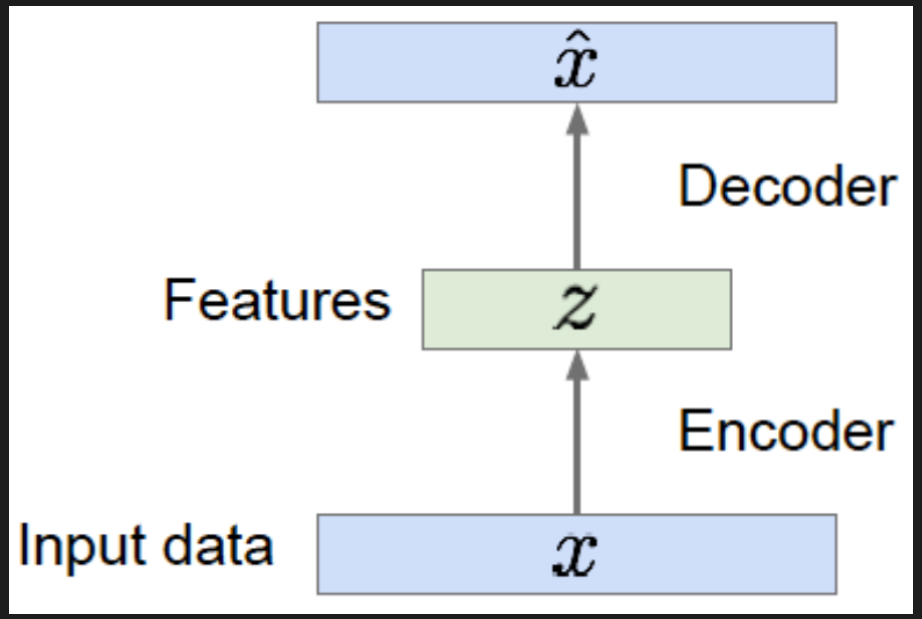
32x32 ImageNet

Variational Autoencoders

Autoencoders

Unsupervised method to learn features from raw data without any labels

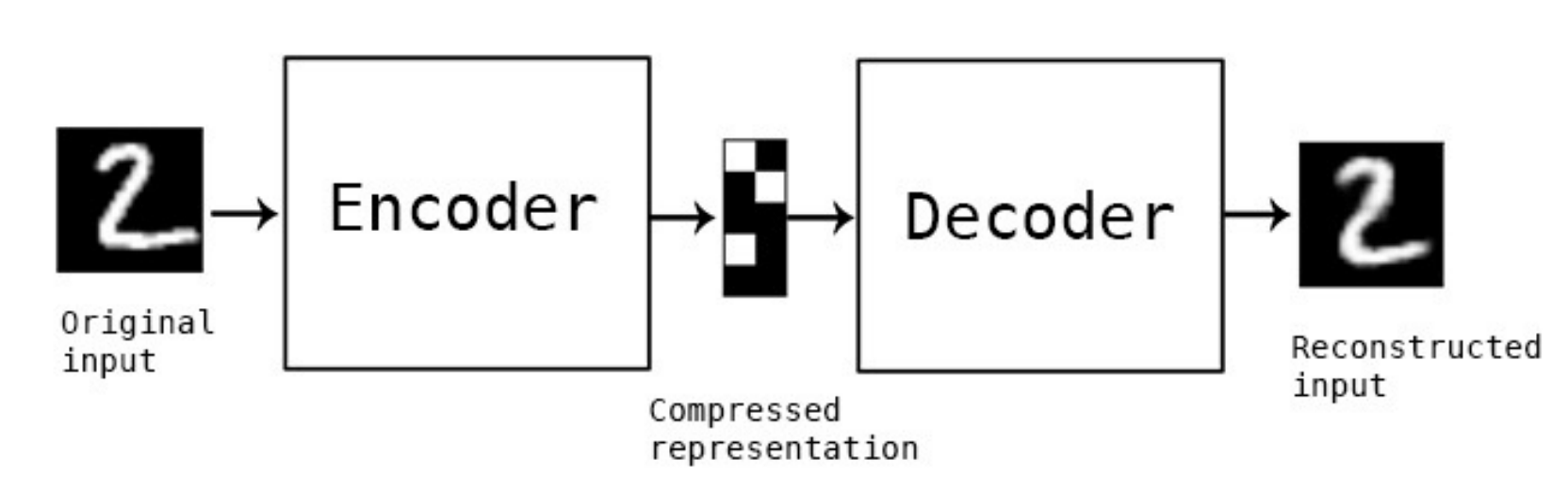
Autoencoder = Encoding itself



Autoencoders

Encode an image into a low dimensional space

And then decode the features to reconstruct the original image



Autoencoders

Minimize L2 distance between the original input and the reconstructed image

There is no need for labels

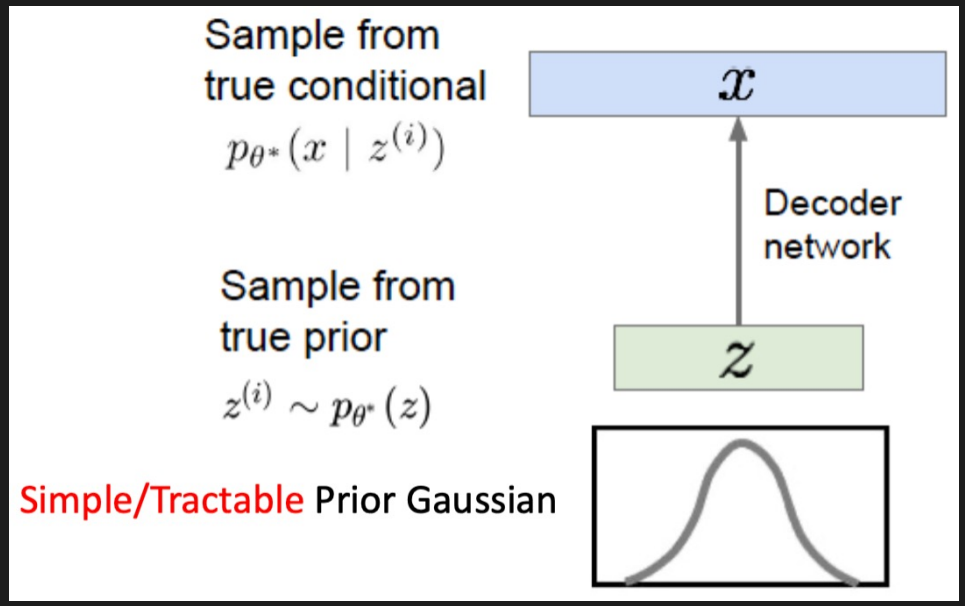
But we cannot generate new data samples

Variational Autoencoders

Assume the prior distribution is a simple, tractable distribution

e.g., Gaussian distribution

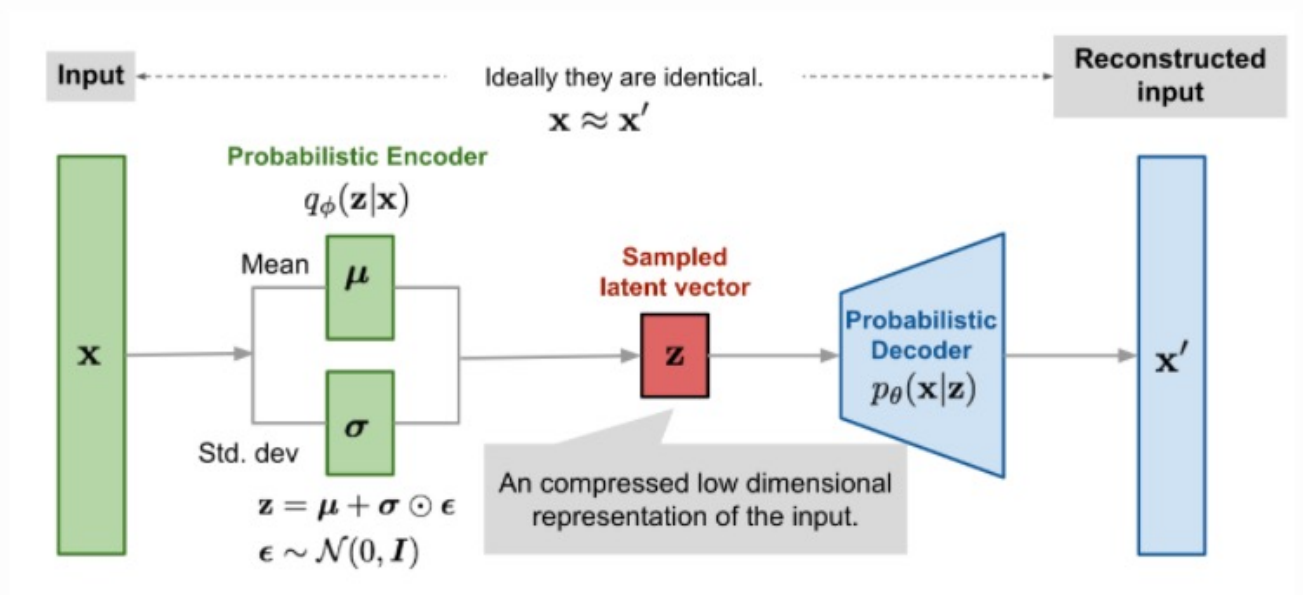
Then we can generate a new sample by (1) sampling z from the prior distribution, and (2) decoding using the conditional distribution (i.e., a decoder)



Variational Autoencoders

Training

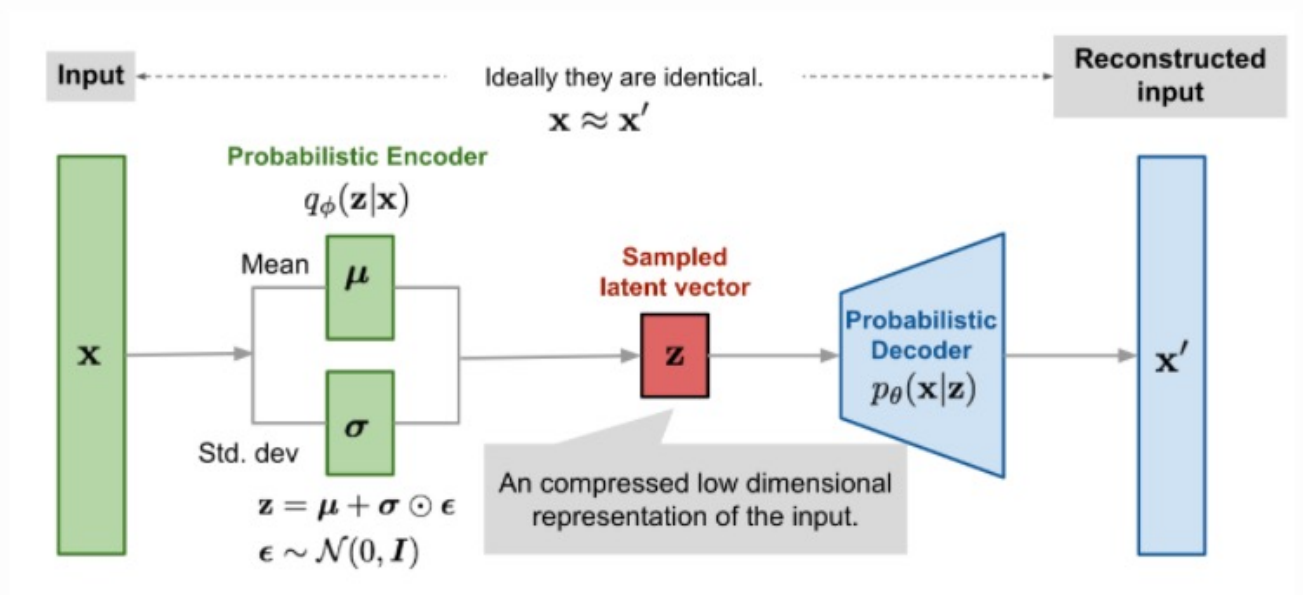
1. Obtain a latent distribution from the input using the encoder
2. Sample z and decode it to reconstruct the original input



Variational Autoencoders

Sampling

1. Sample z from the prior distribution
2. Decode z to make an image from it using the decoder



Variational Autoencoders

Evidence Lower Bound

We want to find a variational distribution close to the posterior distribution

= Minimizing the KL divergence term

$$
\begin{aligned}
p_\theta(D) &= \mathbb{E}_{q_\phi(z|x)}\left[\ln p_\theta(x)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{p_\theta(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)q_\theta(z|x)}{q_\theta(z|x)p_\theta(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{q_\theta(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{q_\theta(z|x)}{p_\theta(z|x)}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{q_\theta(z|x)}\right]}_{\text{ELBO}\uparrow} + \underbrace{D_{KL}(q_\theta(z|x)\|p_\theta(z|x))}_{\text{Objective}\downarrow}
\end{aligned}
$$

Variational Autoencoders

Evidence Lower Bound

It is not tractable since we do not know the posterior distribution

Instead, maximize the remaining term

$$
\begin{aligned}
p_\theta(D) &= \mathbb{E}_{q_\phi(z|x)}\left[\ln p_\theta(x)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{p_\theta(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)q_\theta(z|x)}{q_\theta(z|x)p_\theta(z|x)}\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{q_\theta(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{q_\theta(z|x)}{p_\theta(z|x)}\right] \\
&= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{q_\theta(z|x)}\right]}_{\text{ELBO}\uparrow} + \underbrace{D_{KL}(q_\theta(z|x)\|p_\theta(z|x))}_{\text{Objective}\downarrow}
\end{aligned}
$$

Variational Autoencoders

Evidence Lower Bound

The ELBO term consists of

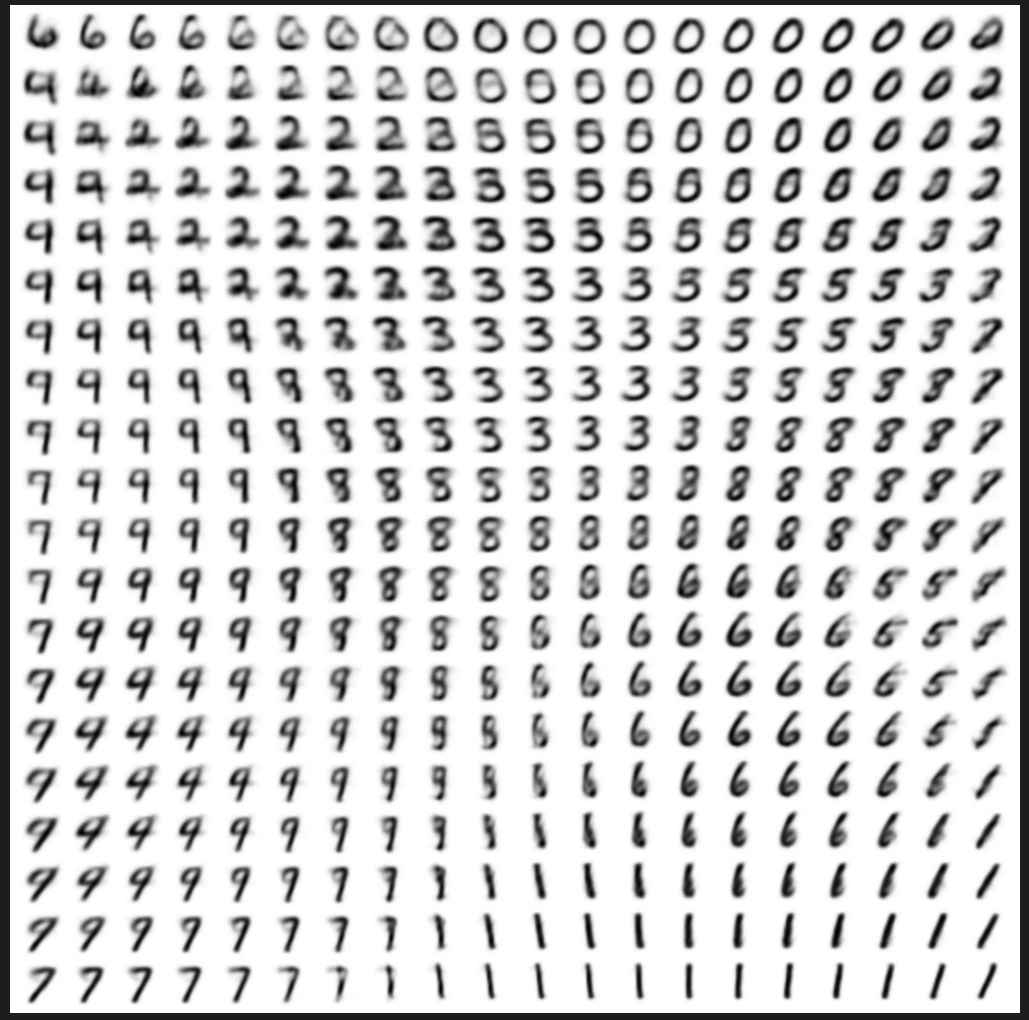
1. Reconstruction term - reconstruction loss of the autoencoder
2. Prior fitting term – make a latent distribution close to the prior distribution

$$\mathbb{E}_{q_\phi(z|x)}\left[\ln \frac{p_\theta(x,z)}{q_\theta(z|x)}\right] = \int \ln \frac{p_\theta(x|z)p(z)}{q_\phi(z|x)} q_\phi(z|x)dz$$
$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}\left[p_\theta(x|z)\right]}_{\text{Reconstruction term}} - \underbrace{D_{KL}(q_\phi(z|x)\|p(z))}_{\text{Prior fitting term}}$$

Variational Autoencoders

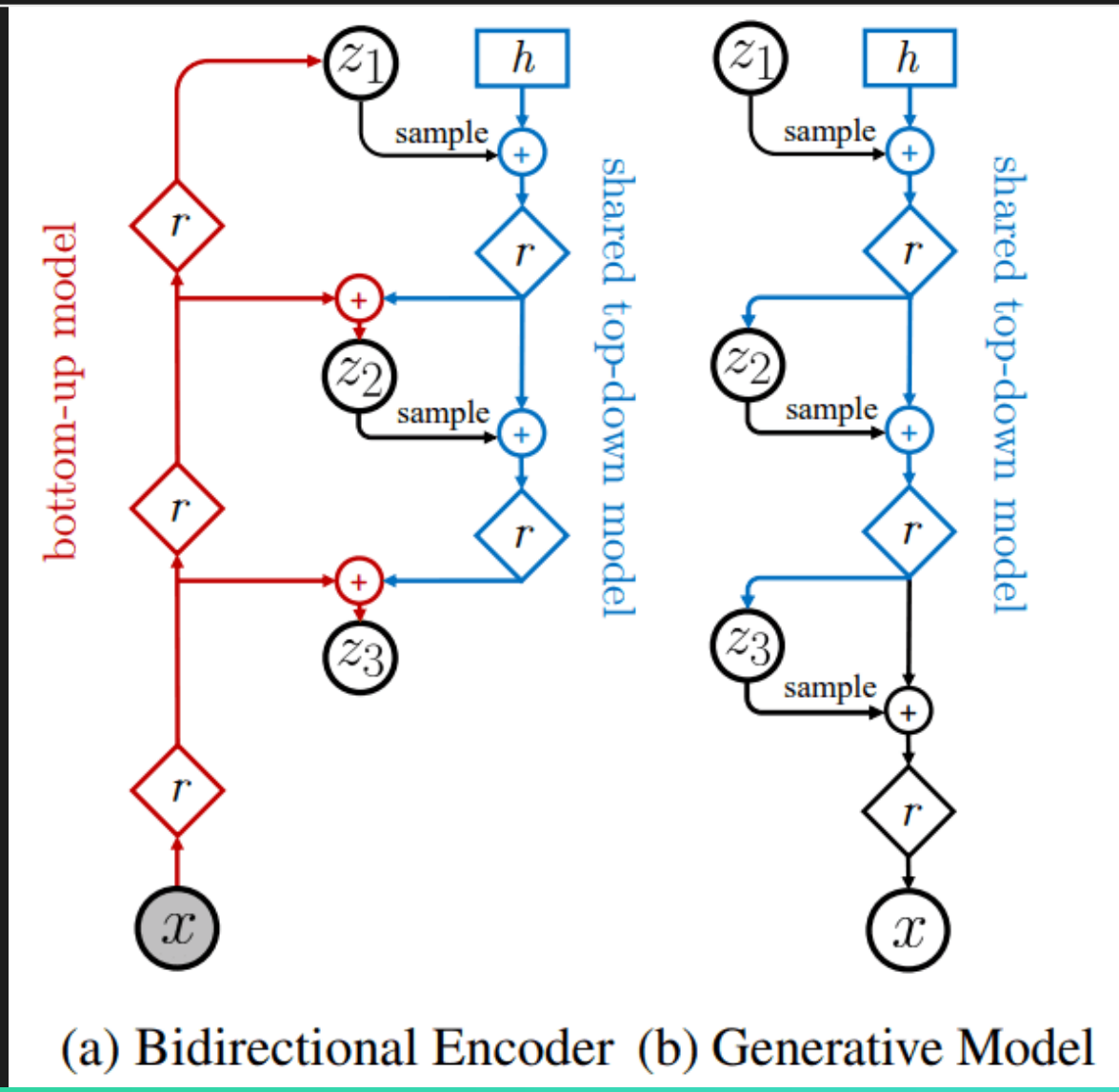
32x32 CIFAR-10

Variational Autoencoders



NVAE: Hierarchical VAE

A deep hierarchical VAE using depth-wise convolutions and batch normalization



(a) Bidirectional Encoder (b) Generative Model

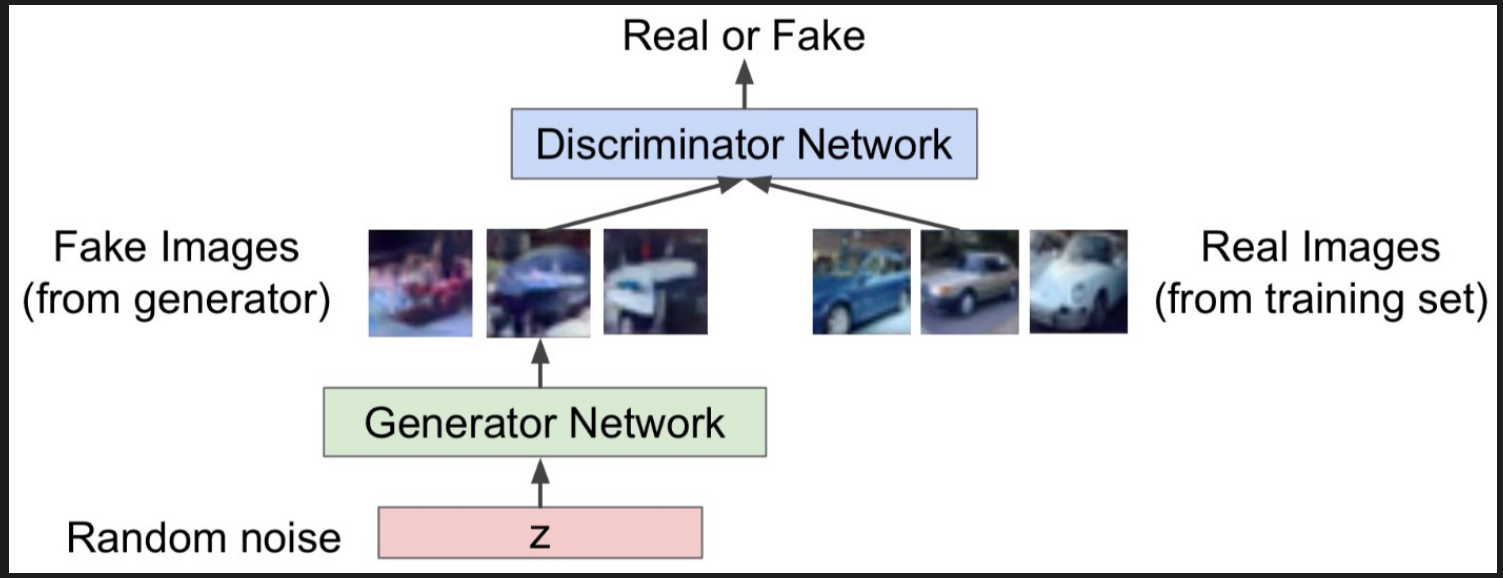
NVAE: Hierarchical VAE

Generative Adversarial Networks

Generative Adversarial Networks

How can we model complex, high-dimensional data distribution?

→ Sample latent variables from a simple distribution and learn transformation



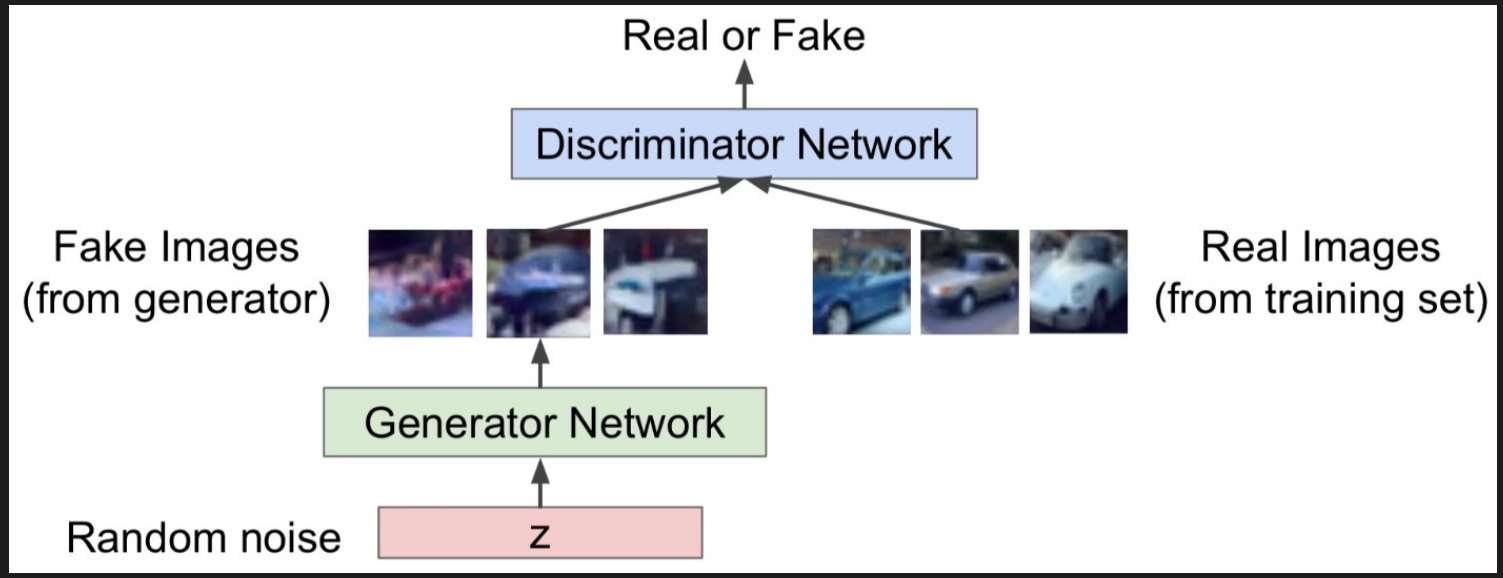
Generative Adversarial Networks

Discriminator

Try to distinguish between the real and fake (generated) images

Generator

Try to fool the discriminator by generating real-looking images



Generative Adversarial Networks

Training

Train a discriminator and a generator jointly in a minimax game

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Generative Adversarial Networks

Training

A discriminator wants to maximize the objective

D(x) becomes close to 1 (real) and D(G(z)) close to 0 (fake)

A generator wants to minimize the objective

D(G(z)) becomes close to 1 (real), that is, fool the discriminator

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

Generative Adversarial Networks

Training

Alternate between

1. Gradient ascent on the discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. Gradient descent on the generator

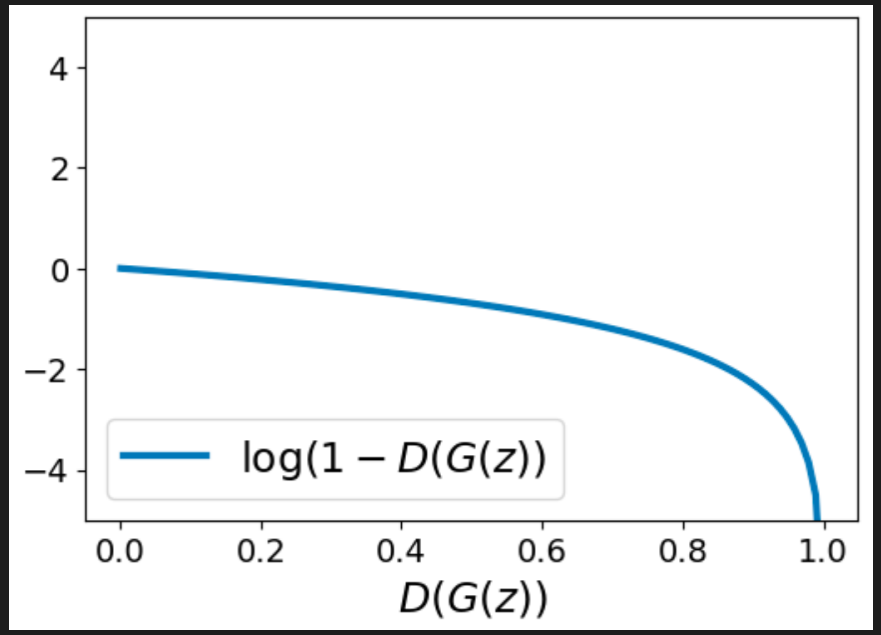
$$\min_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$

Generative Adversarial Networks

Training

Initialliy the generator is bad and the discriminator can easily distinguish real/fake

D(G(z) is close to 0 → Vanishing gradient for the generator

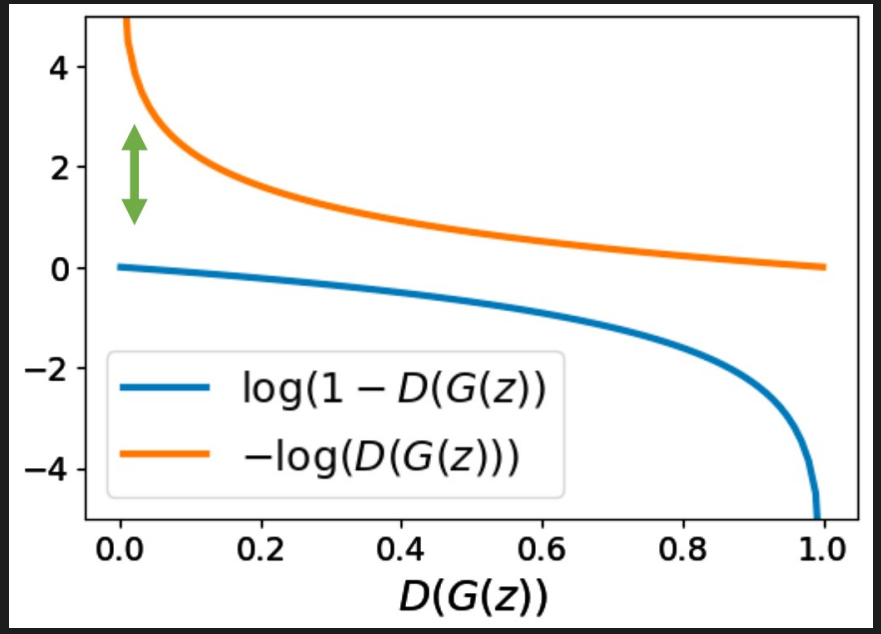


Generative Adversarial Networks

Training

Instead train the generator to miminize –log(D(G(z)))

The generator now has strong gradient signal



Generative Adversarial Networks

Training

Alternate between

1. Gradient ascent on the discriminator

$$\max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

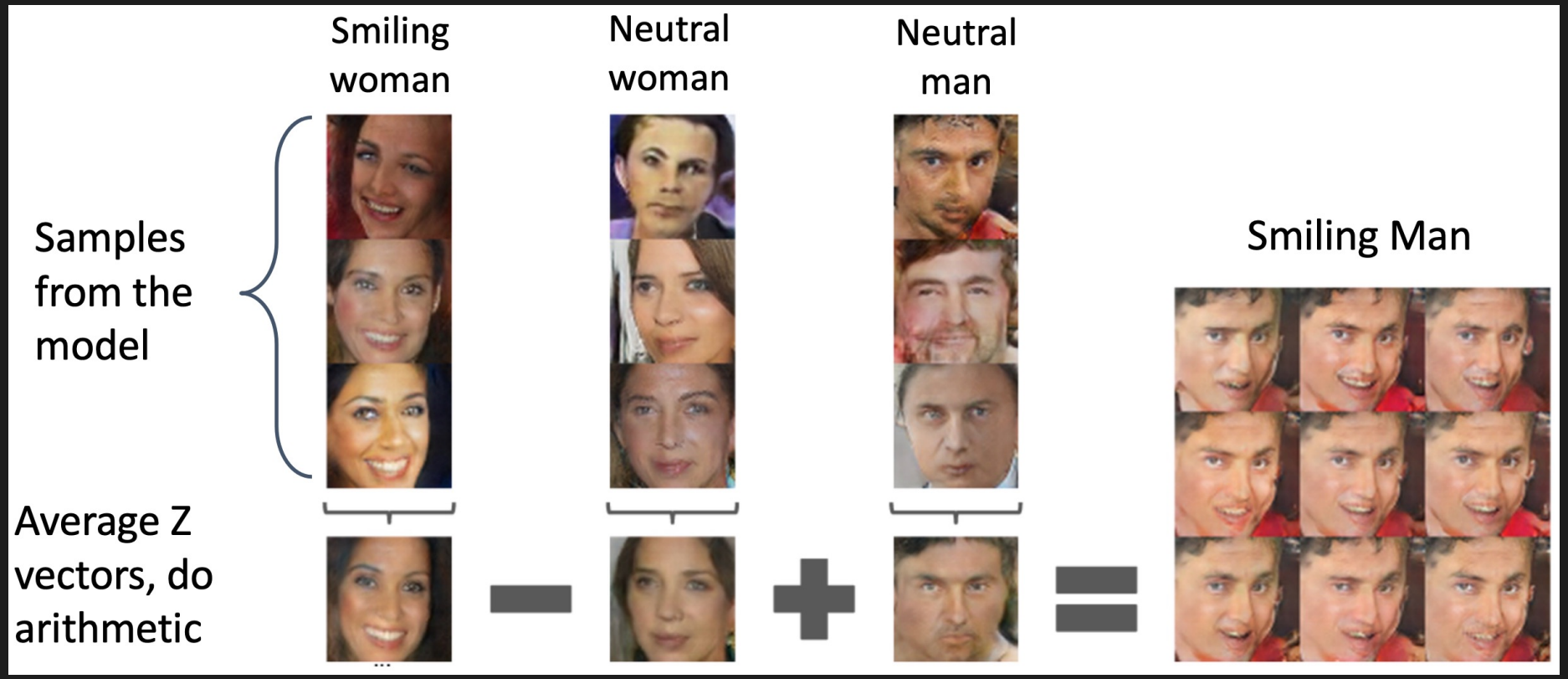
2. Gradient ascent on the generator with another objective

$$\max_{\theta_g} \mathbb{E}_{z \sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$

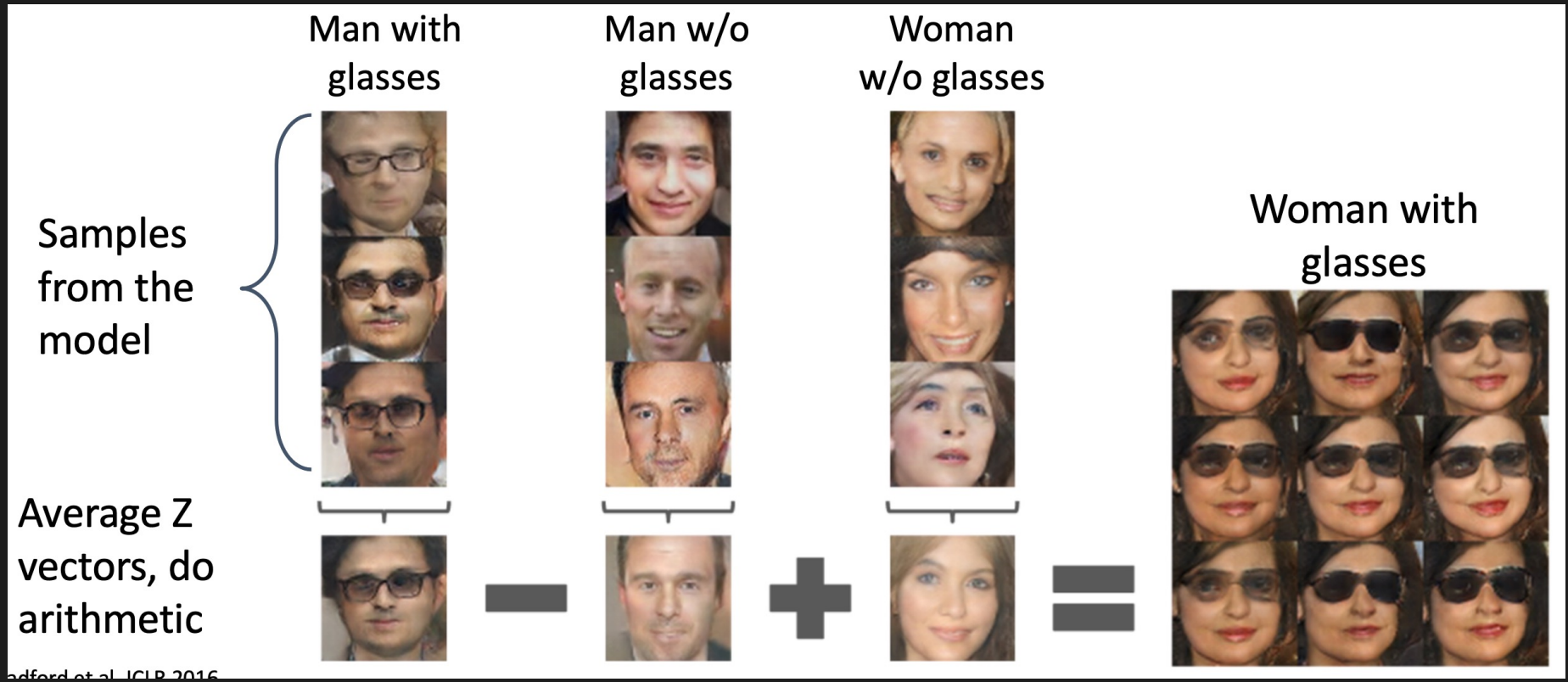
Generative Adversarial Networks: Example

Generative Adversarial Networks: Example

Generative Adversarial Networks: Example



Generative Adversarial Networks: Example



adford et al, ICLR 2016

Generative Adversarial Networks: Example

Improved Loss Functions

Wasserstein GAN (WGAN)

Arjovsky, Chintala, and Bouttou, “Wasserstein GAN”, 2017

WGAN with Gradient Penalty (WGAN-GP)

Gulrajani et al, “Improved Training of Wasserstein GANs”, NeurIPS 2017

Generative Adversarial Networks: Example

High Resolution

512 x 384 cars

1024 x 1024 faces

Generative Adversarial Networks: Example

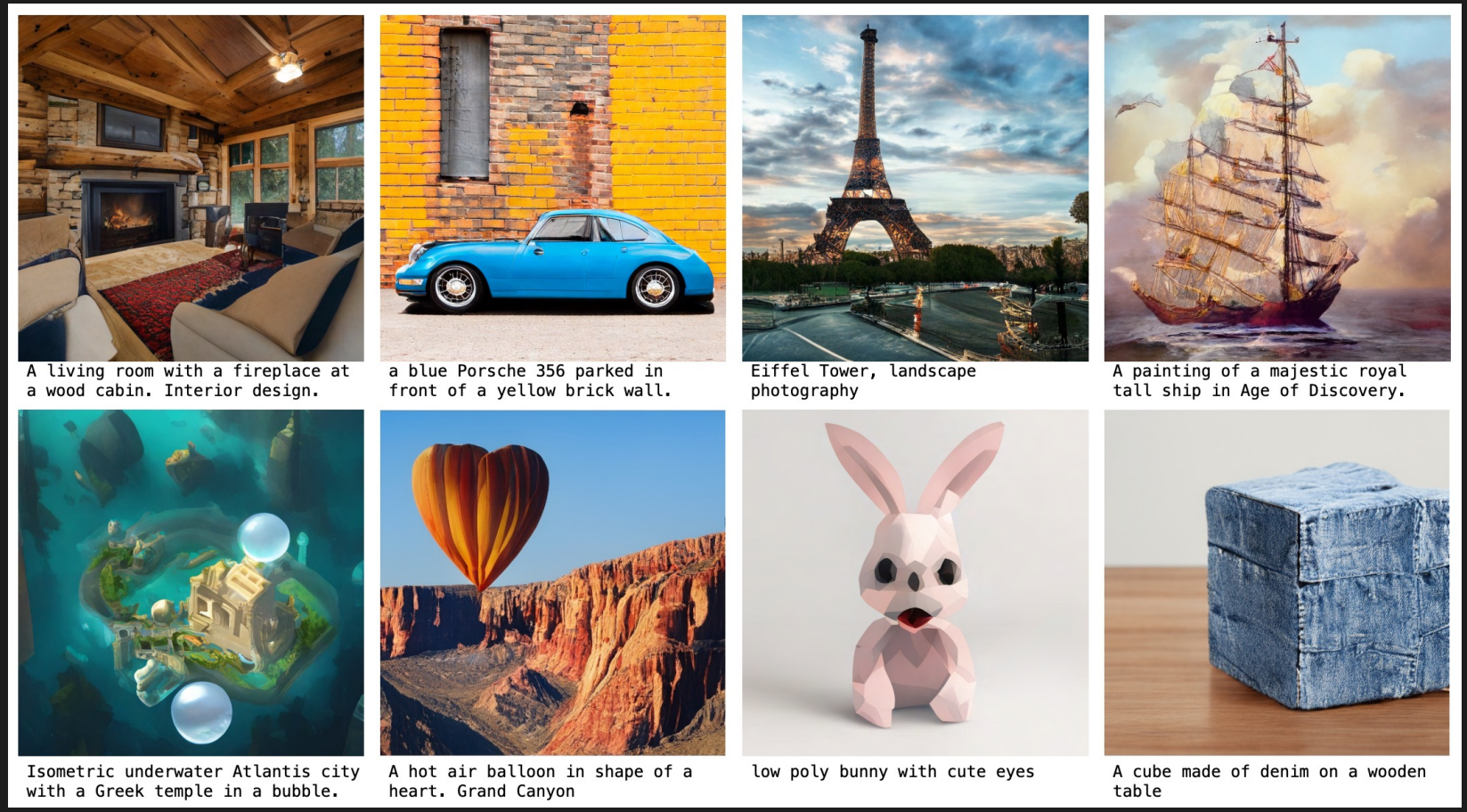
BigGAN

Generative Adversarial Networks: Example

BigGAN

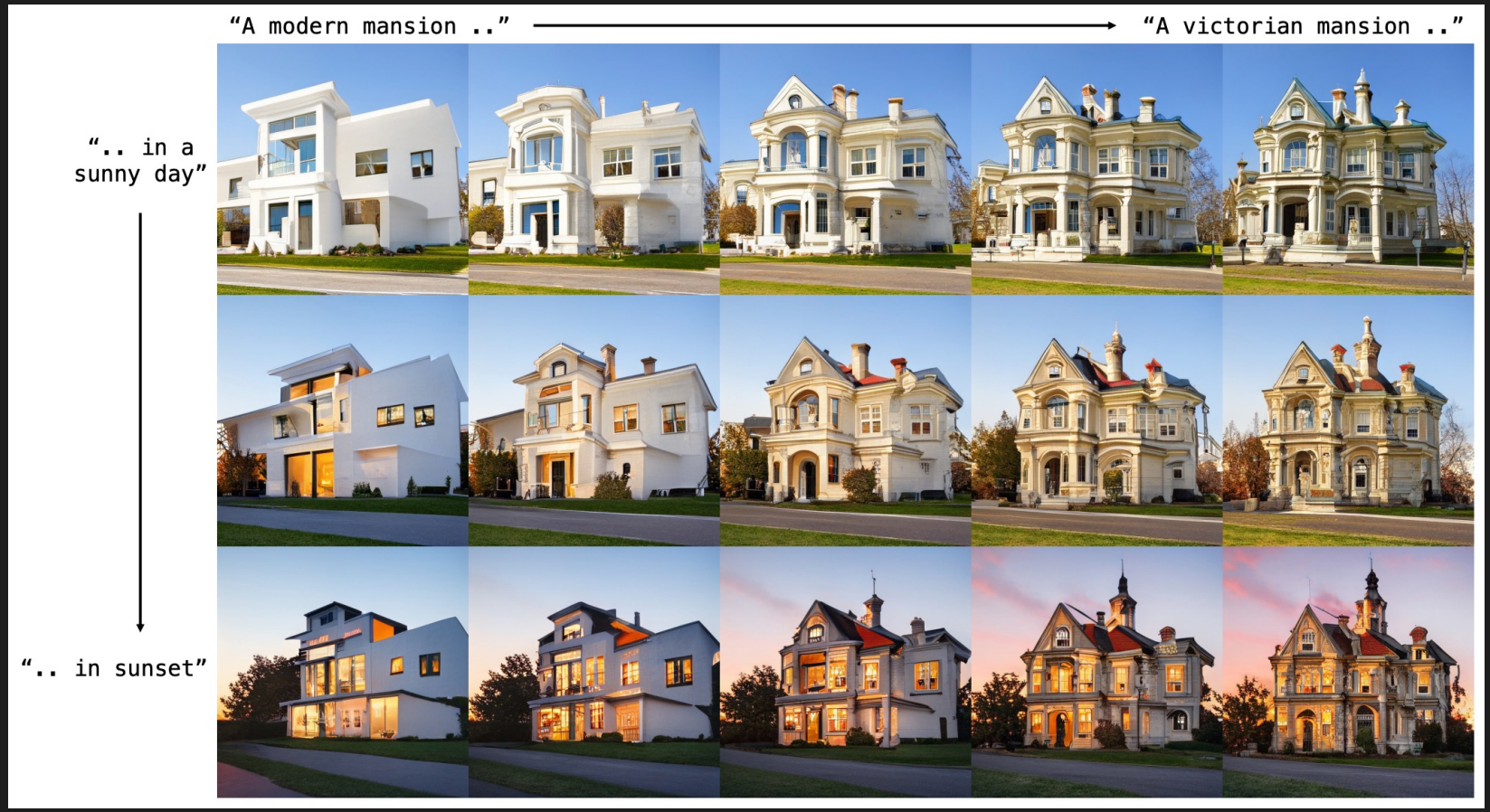
Generative Adversarial Networks: Example

GigaGAN



Generative Adversarial Networks: Example

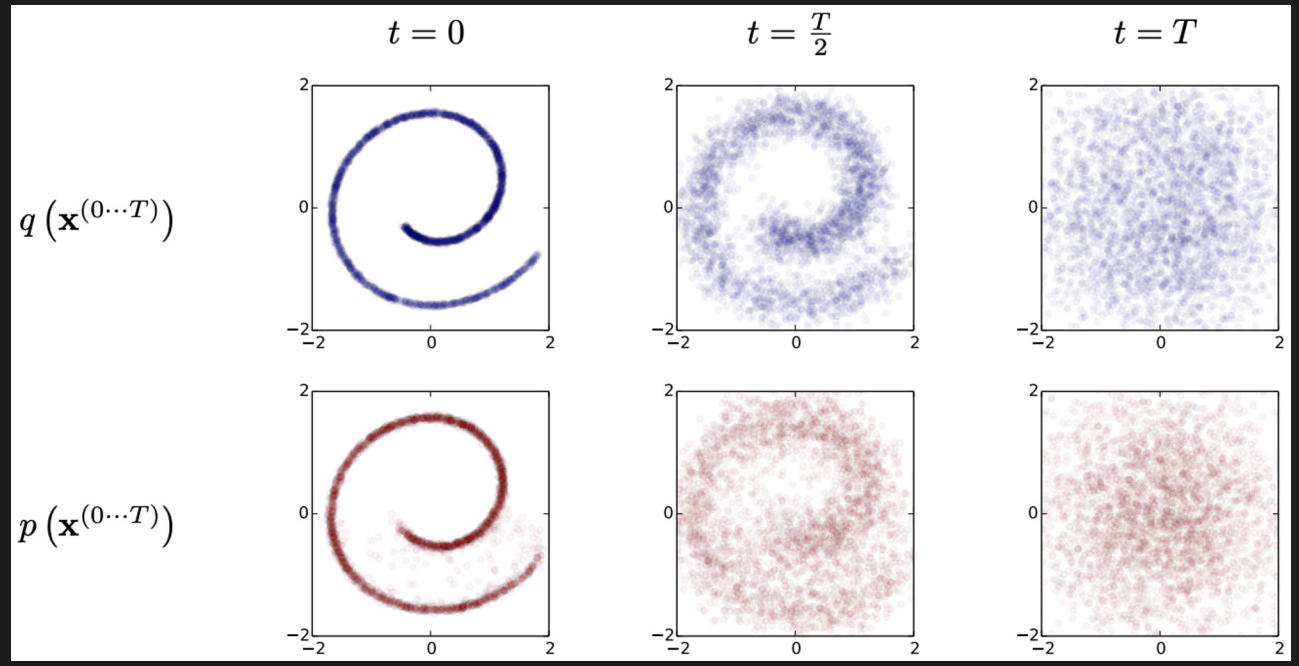
GigaGAN



Diffusion Models

Diffusion Models

A parameterized Markov chain whose transitions are learned to reverse a diffusion process

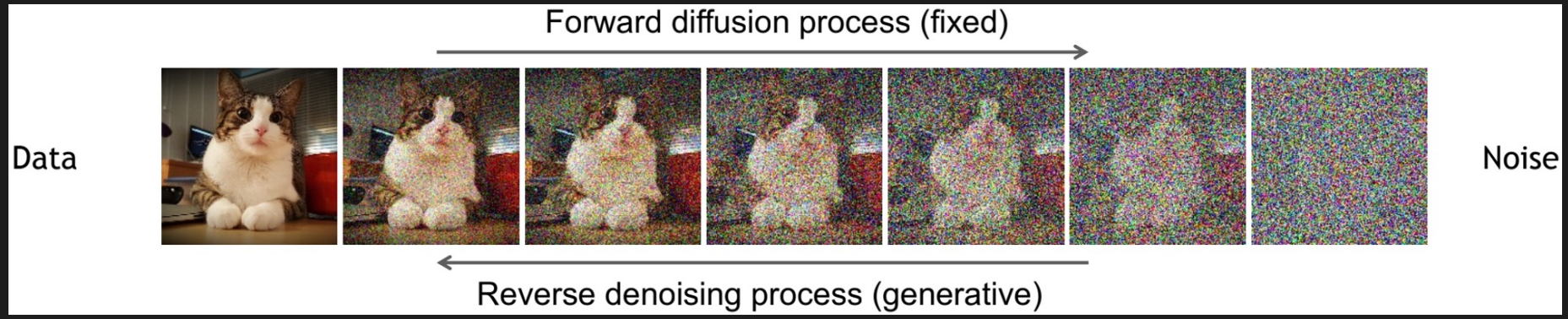


Diffusion Models

Diffusion models consist of two processes

1. Forward diffusion process gradually adds noise to input
2. Reverse denoising process that learns to generate data by denoising

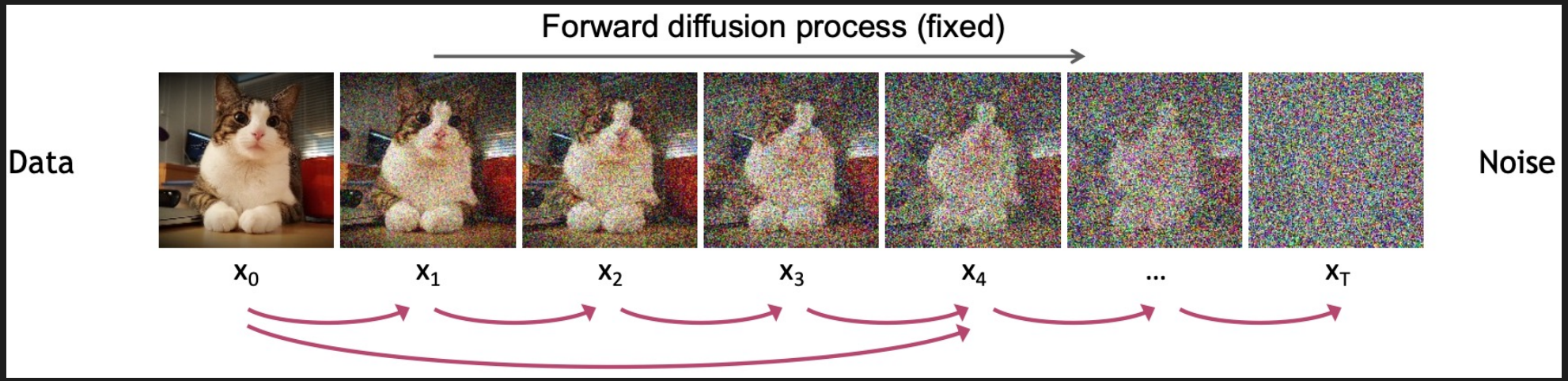
For many forward diffusion processes, the reverse diffusion processes can be described using the same functional form



Diffusion Models

Forward Diffusion Process

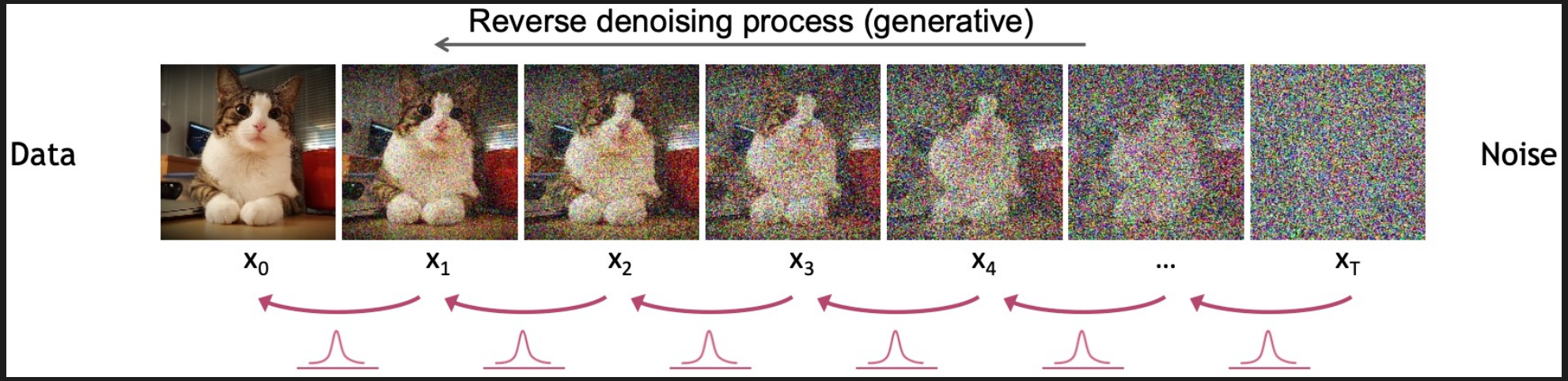
$$q(\mathbf{x}_{1:T}|\mathbf{x}_0) := \prod_{t=1}^{T} q(\mathbf{x}_t|\mathbf{x}_{t-1})$$
$$q(\mathbf{x}_t|\mathbf{x}_{t-1}) := \mathcal{N}(\mathbf{x}_t; \sqrt{1-\beta_t}\mathbf{x}_{t-1}, \beta_t\mathbf{I})$$



Diffusion Models

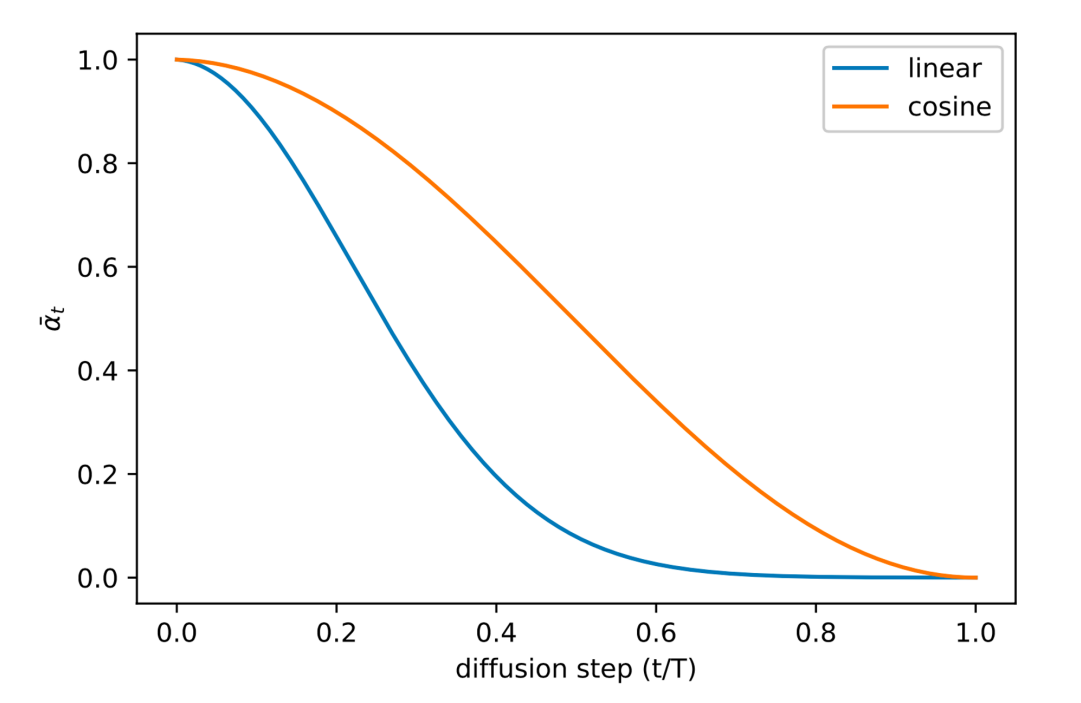
Reverse Denoising Process

$$
\begin{aligned}
&p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}) \\
&p_\theta(\mathbf{x}_{0:T}) := p(\mathbf{x}_T) \prod_{t=1}^{T} p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) \\
&p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) := \mathcal{N}(\mathbf{x}_{t-1}; \mu_\theta(\mathbf{x}_t, t), \mathbf{\Sigma}_\theta(\mathbf{x}_t, t))
\end{aligned}
$$



Diffusion Models

Variance Schedule



Diffusion Models

Simplified Loss by Noise Prediction

Given the perturbed image, predict its denoised version

We can instead predict the noise used for perturbation

$$L = \mathbb{E}_q \Big[\underbrace{D_{KL}(q(\mathbf{x}_T|\mathbf{x}_0) \| p(\mathbf{x}_T))}_{L_T} + \underbrace{\prod_{t>1} D_{KL}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0) \| p(\mathbf{x}_{t-1}|\mathbf{x}_t))}_{L_{t-1}} - \underbrace{\log p_\theta(\mathbf{x}_0|\mathbf{x}_1)}_{L_0} \Big]$$

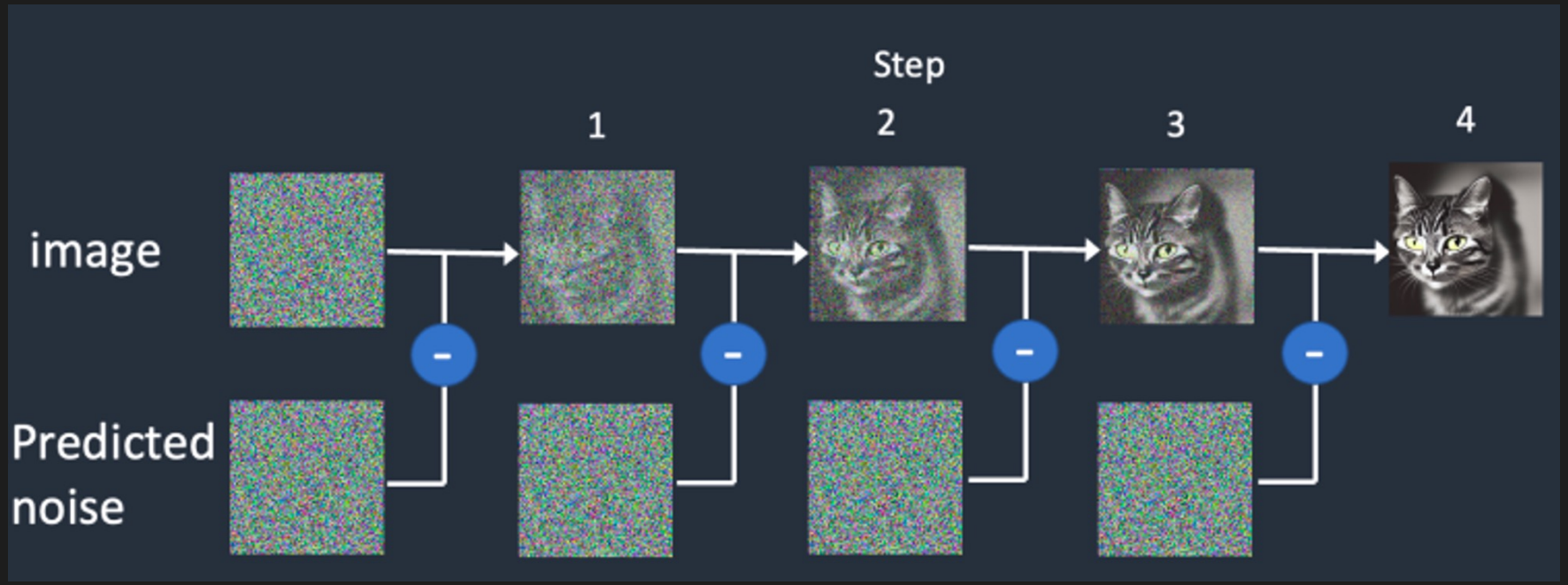
$$L_{simple}(\theta) := \mathbb{E}_{t, \mathbf{x}_0, \epsilon} \left[\| \epsilon - \epsilon_\theta(\sqrt{\bar{\alpha}_t}\mathbf{x}_0 + \sqrt{1-\bar{\alpha}_t}\epsilon, t) \|^2 \right]$$

Diffusion Models

Simplified Loss by Noise Prediction

Given the perturbed image, predict its denoised version

We can instead predict the noise used for perturbation



Diffusion Models

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

Algorithm 2 Sampling

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

Diffusion Models

Denoising Diffusion Implicit Model

Generalize DDPMs via a class of non-Markovian diffusion processes

These non-Markovian processes can correspond to deterministic, faster generative processes

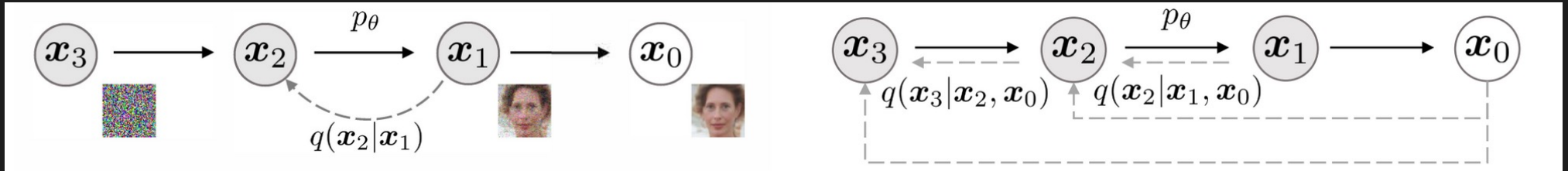
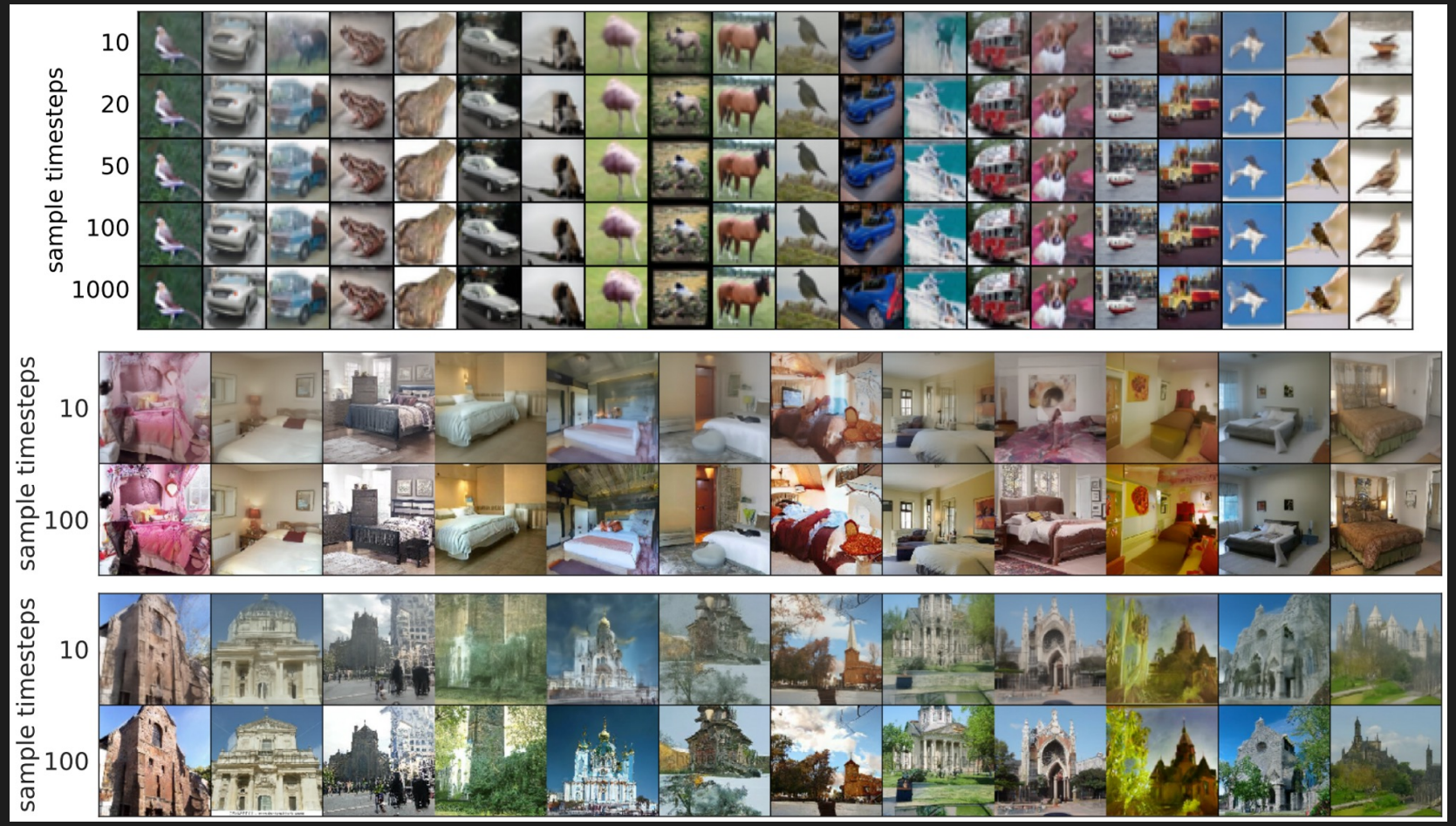


Figure 1: Graphical models for diffusion (left) and non-Markovian (right) inference models.

Diffusion Models

Denoising Diffusion Implicit Model



Diffusion Models

Classifier Guidance

Train a classifier on noisy image

Use its gradient to guide the diffusion sampling process toward the condition

Algorithm 1 Classifier guided diffusion sampling, given a diffusion model $(\mu_\theta(x_t), \Sigma_\theta(x_t))$, classifier $f_\phi(y|x_t)$, and gradient scale s.

Input: class label y, gradient scale s
$x_T \leftarrow$ sample from $\mathcal{N}(0, \mathbf{I})$
for all t from T to 1 **do**
  $\mu, \Sigma \leftarrow \mu_\theta(x_t), \Sigma_\theta(x_t)$
  $x_{t-1} \leftarrow$ sample from $\mathcal{N}(\mu + s\Sigma \nabla_{x_t} \log f_\phi(y|x_t), \Sigma)$
end for
return x_0

Diffusion Models

Classifier Guidance

Train a classifier on noisy image

Use its gradient to guide the diffusion sampling process toward the condition

Figure 3: Samples from an unconditional diffusion model with classifier guidance to condition on the class "Pembroke Welsh corgi". Using classifier scale 1.0 (left; FID: 33.0) does not produce convincing samples in this class, whereas classifier scale 10.0 (right; FID: 12.0) produces much more class-consistent images.

Diffusion Models

Classifier-Free Guidance

Classifier guidance requires training a classifier

Guide the diffusion process using unconditional and conditional models

Algorithm 1 Joint training a diffusion model with classifier-free guidance

Require: p_{uncond}: probability of unconditional training

1: **repeat**
2: $(\mathbf{x}, \mathbf{c}) \sim p(\mathbf{x}, \mathbf{c})$ ▷ Sample data with conditioning from the dataset
3: $\mathbf{c} \leftarrow \varnothing$ with probability p_{uncond} ▷ Randomly discard conditioning to train unconditionally
4: $\lambda \sim p(\lambda)$ ▷ Sample log SNR value
5: $\boldsymbol{\epsilon} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
6: $\mathbf{z}_\lambda = \alpha_\lambda \mathbf{x} + \sigma_\lambda \boldsymbol{\epsilon}$ ▷ Corrupt data to the sampled log SNR value
7: Take gradient step on $\nabla_\theta \left\| \boldsymbol{\epsilon}_\theta(\mathbf{z}_\lambda, \mathbf{c}) - \boldsymbol{\epsilon} \right\|^2$ ▷ Optimization of denoising model
8: **until** converged

Diffusion Models

Classifier-Free Guidance

Classifier guidance requires training a classifier

Guide the diffusion process using unconditional and conditional models

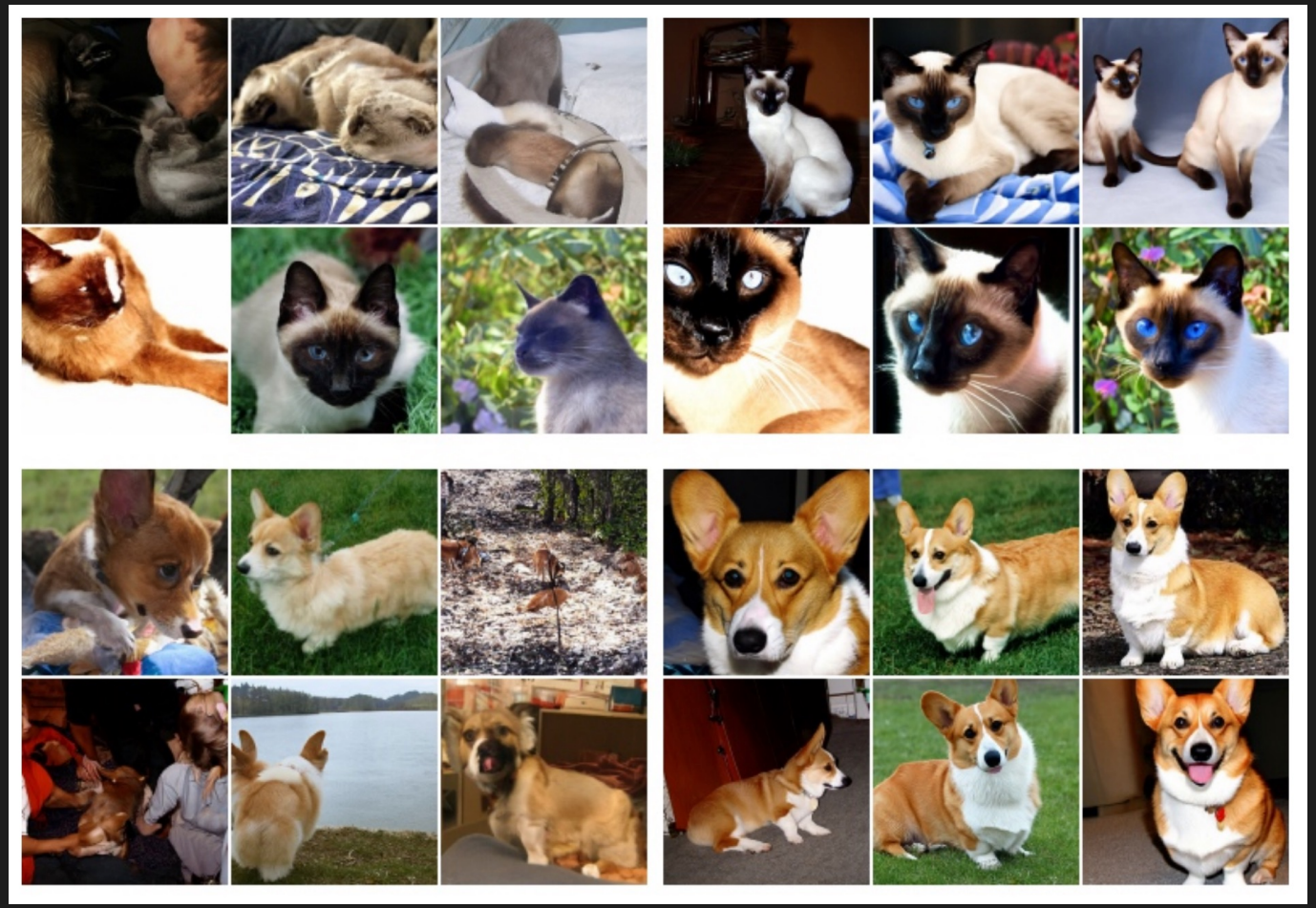
Algorithm 2 Conditional sampling with classifier-free guidance

Require: w: guidance strength
Require: $\mathbf{c}$: conditioning information for conditional sampling
Require: $\lambda_1, \ldots, \lambda_T$: increasing log SNR sequence with $\lambda_1 = \lambda_{\min}, \lambda_T = \lambda_{\max}$

$\mathbf{z}_1 \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
for $t = 1, \ldots, T$ **do**
  ▷ Form the classifier-free guided score at log SNR λ_t
  $\tilde{\boldsymbol{\epsilon}}_t = (1 + w)\boldsymbol{\epsilon}_\theta(\mathbf{z}_t, \mathbf{c}) - w\boldsymbol{\epsilon}_\theta(\mathbf{z}_t)$
  ▷ Sampling step (could be replaced by another sampler, e.g. DDIM)
  $\tilde{\mathbf{x}}_t = (\mathbf{z}_t - \sigma_{\lambda_t}\tilde{\boldsymbol{\epsilon}}_t)/\alpha_{\lambda_t}$
  $\mathbf{z}_{t+1} \sim \mathcal{N}(\tilde{\boldsymbol{\mu}}_{\lambda_{t+1}|\lambda_t}(\mathbf{z}_t, \tilde{\mathbf{x}}_t), (\tilde{\sigma}^2_{\lambda_{t+1}|\lambda_t})^{1-v}(\sigma^2_{\lambda_t|\lambda_{t+1}})^v)$ if $t < T$ else $\mathbf{z}_{t+1} = \tilde{\mathbf{x}}_t$
end for
return $\mathbf{z}_{T+1}$

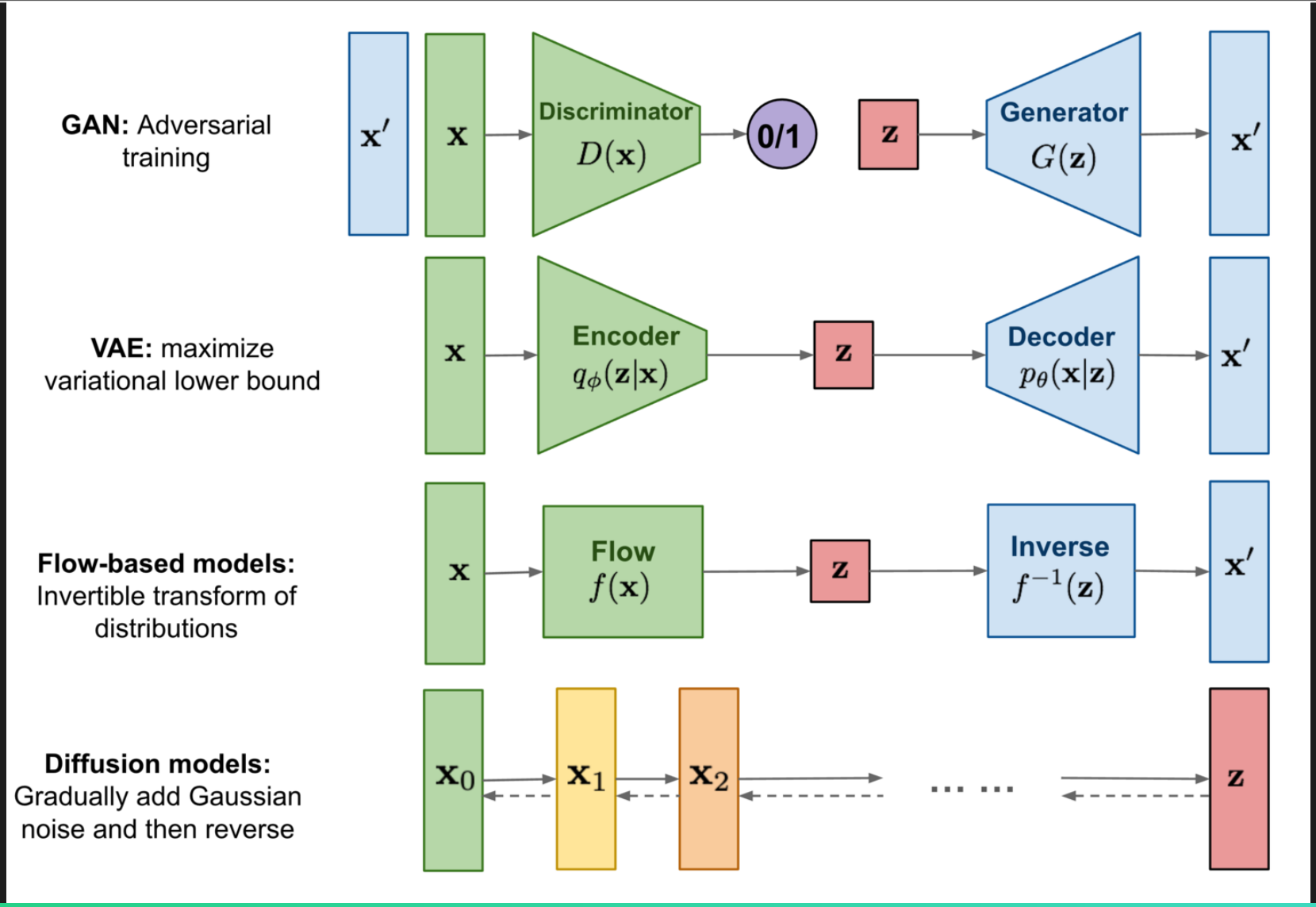
Diffusion Models

Classifier-Free Guidance



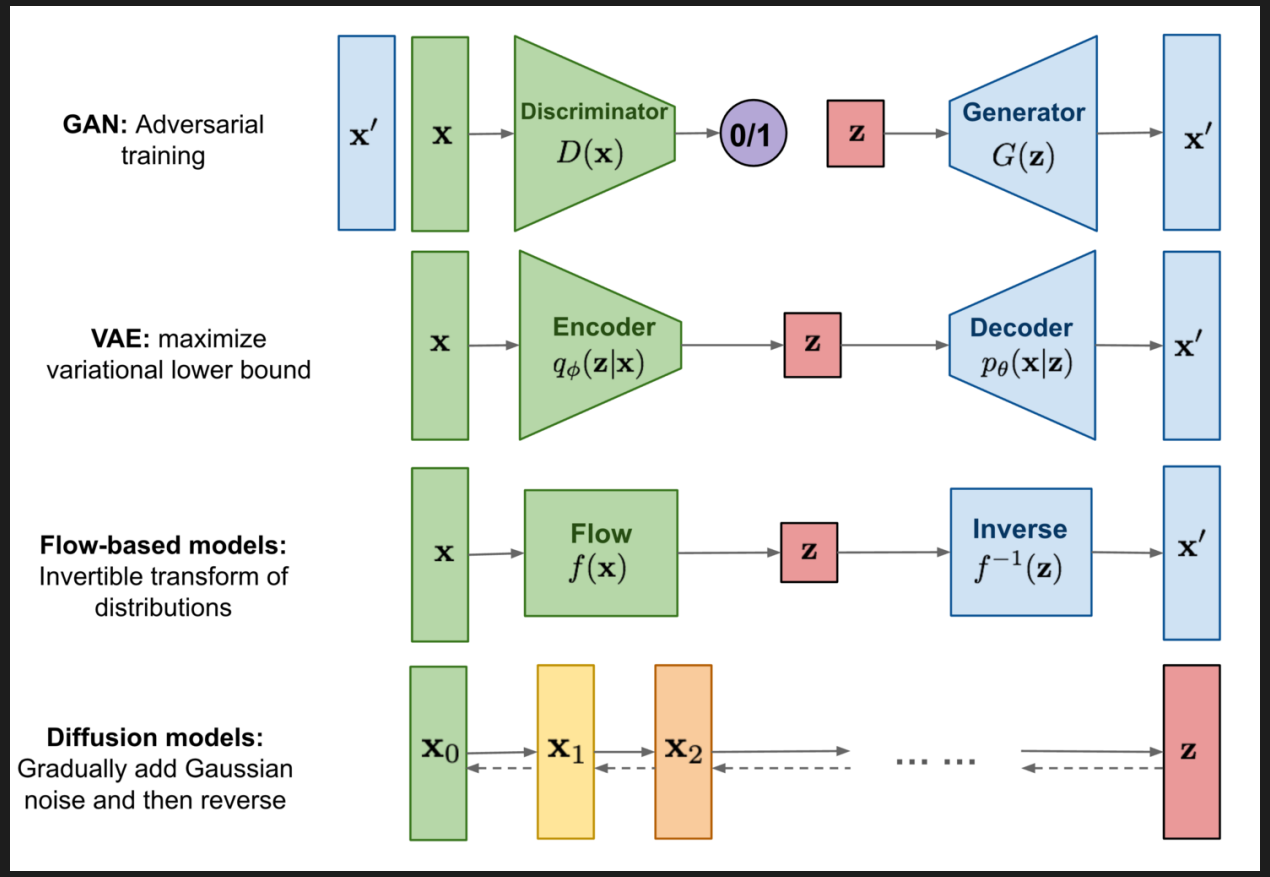
Summary

Models for Image Generation



Discussion Assignment

1. 다음 그림을 참고하여 VAE, GAN, Diffusion model을 각 한 문장으로 요약하세요.



Discussion Assignment

2. 이미지 생성 모델 데모 사이트에서 프롬프트를 입력하여 이미지를 생성해보세요. 디스커션 조 내에서 생성한 이미지를 공유하여 자유롭게 토론하세요.

- Stable Diffusion Online https://stablediffusionweb.com/
- DALL·E 2 https://labs.openai.com/

참고자료

자료 출처

- University of Michigan EECS 498.008 / 598.008 Deep Learning for Computer Vision
- Stanford University CS231n Deep Learning for Computer Vision
- Background: What is a Generative Model?
- A simple explanation of the Inception Score

추가 자료

- Scaling up GANs for Text-to-Image Synthesis
- Prompt-to-Prompt Image Editing with Cross-Attention Control

감사합니다.

AIKU, 디자인 가이드

제목: Spoqa Han Sans Neo Bold 32pt #FFFFFF

- 이 글은 예시입니다.
  - 두번째 들여쓰기 입니다.
    - 세번째 들여쓰기입니다.
      - 네번째 들여쓰기입니다.

본문: Spoqa Han Sans Neo Medium 24pt ~ 16pt #FFFFFF

Column1	Column2
Index1	Value1
Index2	Value2
Index3	Value3

표: Spoqa Han Sans Neo Medium 12pt #FFFFFF