Deep Generative Learning

Deep Learning for Computer Vision

Valeriya Strizhkova
15.03.23

About myself

Valeriya Strizhkova

2nd year PhD student @ Inria & 3iA & UCA

valeriya.strizhkova@inria.fr

Research topics:

- emotion detection
- computer vision
- deep learning

Outline

“

{ There are many interesting recent development in deep learning...The most important one, in my opinion, is adversarial training (also called GAN for Generative Adversarial Networks). This, and the variations that are now being proposed, is the most interesting idea in the last 10 years in ML. }

Yann LeCun

The Landscape of Deep Generative Learning

Autoregressive Models
Normalizing Flows
Variational Autoencoders
Generative Adversarial Networks
Energy-based Models
Denoising Diffusion Models

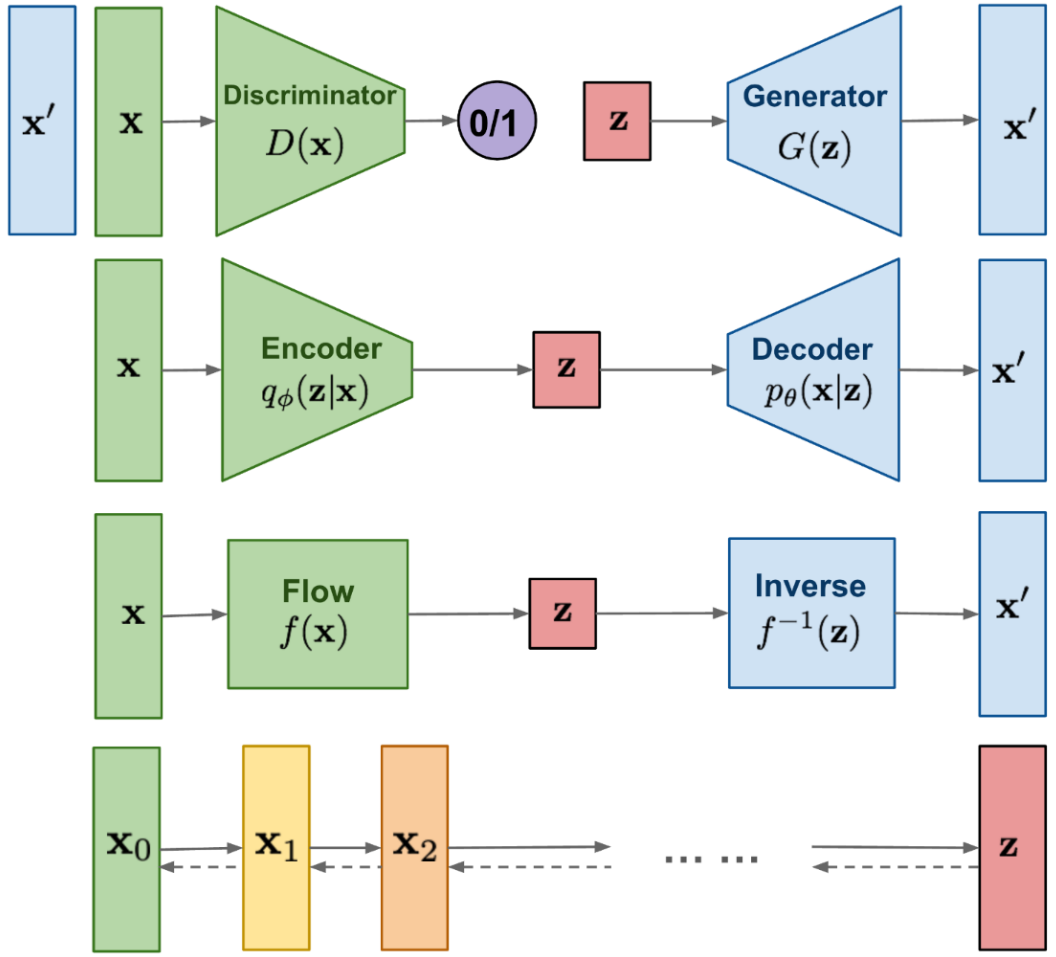
source: https://cvpr2022-tutorial-diffusion-models.github.io

GAN: Adversarial training

VAE: maximize variational lower bound

Flow-based models: Invertible transform of distributions

Diffusion models: Gradually add Gaussian noise and then reverse



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

A question to ChatGPT

Q: When is it better to use **GAN** and when **diffusion models**?

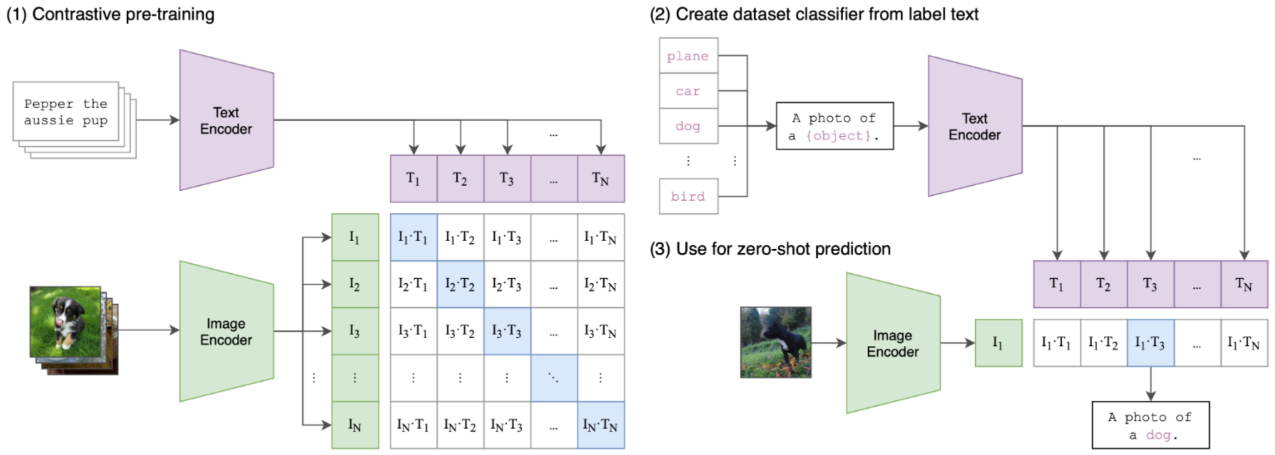
A: The choice between using GANs and diffusion models depends on the specific characteristics of the data being generated and the requirements of the generative modeling task at hand.

In general, GANs are often preferred for tasks that require generating high-quality, visually realistic images, as they have been shown to produce photorealistic images with fine details and textures. GANs are also well-suited for image-to-image translation tasks, such as style transfer or super-resolution, where the goal is to modify an existing image in a particular way while preserving its underlying structure and content.

On the other hand, diffusion models are often preferred for tasks that involve modeling complex distributions or generating samples that exhibit long-range dependencies. Diffusion models have been shown to be effective at modeling complex image and video datasets, and can generate high-quality samples that capture the underlying structure and dynamics of the data. Diffusion models can also be used for tasks such as image denoising or inpainting, where the goal is to remove noise or missing information from an image while preserving its structure and content.

OpenAI’s CLIP

- encodes image, and text to similar embeddings
- is trained with contrastive learning, maximizing cosine similarity of corresponding image and text
- CLIP’s output image embeddings contain both style and semantics
- is used in DALL-E 1, DALL-E 2, image-text classification


(1) Contrastive pre-training
Pepper the aussie pup
Text Encoder
Image Encoder
T1 T2 T3 ... TN
I1 I2 I3 ... IN
I1·T1 I1·T2 I1·T3 ... I1·TN
I2·T1 I2·T2 I2·T3 ... I2·TN
I3·T1 I3·T2 I3·T3 ... I3·TN
IN·T1 IN·T2 IN·T3 ... IN·TN
(2) Create dataset classifier from label text
plane
car
dog
bird
A photo of a {object}.
Text Encoder
(3) Use for zero-shot prediction
Image Encoder
I1
I1·T1 I1·T2 I1·T3 ... I1·TN
A photo of a dog.

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. ICML 2021 https://arxiv.org/abs/2103.00020 https://openai.com/research/clip

CLIP Architecture

- text and image have separate transformer encoders
- visual encoder is ViT (vision transformer)
- text encoder is GPT-2 transformer
- the fixed-length text embedding is extracted from [EOS] token position
- text token embeddings and image patch embeddings also available
- trained on 256 GPUs for 2 weeks

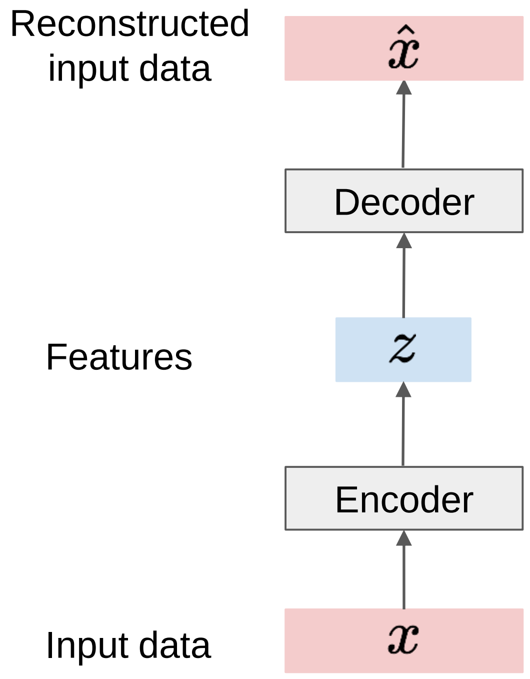
Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, Ilya Sutskever. Learning Transferable Visual Models From Natural Language Supervision. ICML 2021 https://arxiv.org/abs/2103.00020 https://openai.com/research/clip

Outline

Autoencoders (non-variational)

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

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

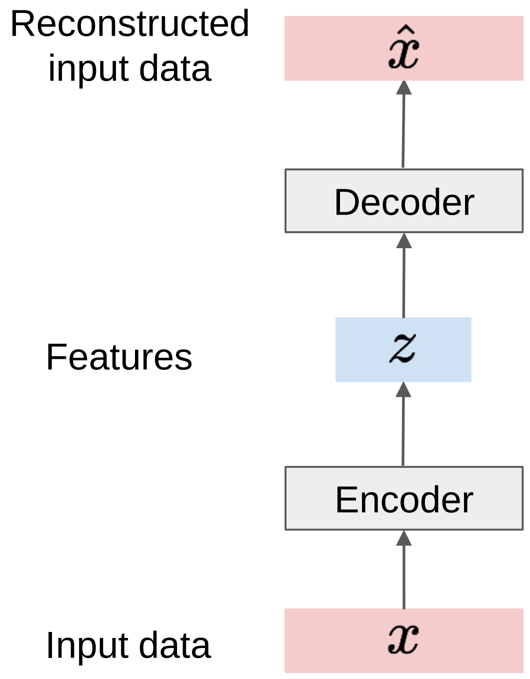


Autoencoders (non-variational)

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

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

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



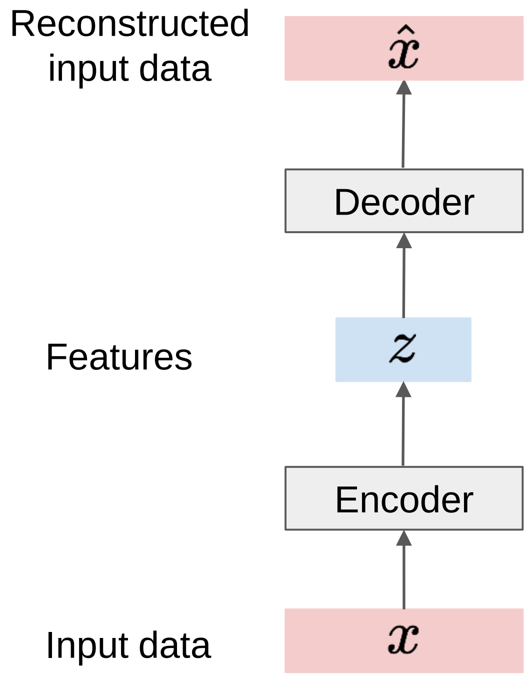
Autoencoders (non-variational)

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

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



Variational Autoencoders (VAE)

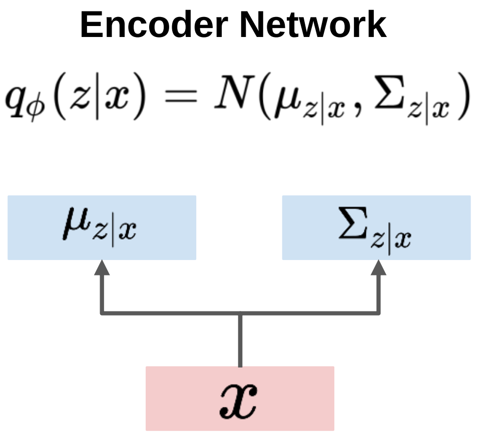
- VAE is an autoencoder that learns **latent features** from data and enables **generative process**.

Variational Autoencoders (VAE)

- 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.

Variational Autoencoders (VAE)

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



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

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

$$\begin{tikzcd} \mu_{z|x} & & \Sigma_{z|x} \\ & x \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Decoder Network

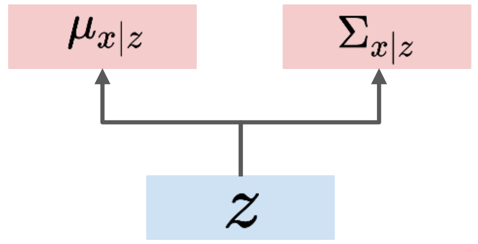
$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$

$$\begin{tikzcd} \mu_{x|z} & & \Sigma_{x|z} \\ & z \arrow[ul] \arrow[ur] & \end{tikzcd}$$

Variational Autoencoders (VAE)

Decoder neural network represent p(x|z) where x is an image, z is latent features to generate x.

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



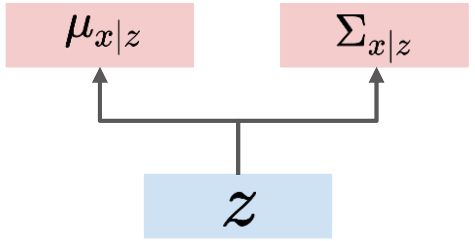
Diederik P Kingma, Max Welling. Auto-Encoding Variational Bayes. ICLR 2014 https://arxiv.org/abs/1312.6114

Variational Autoencoders (VAE)

Decoder neural network represent p(x|z) where x is an image, z is latent features to generate x.

Assume prior p(z) is standard unit diagonal Gaussian.

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



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

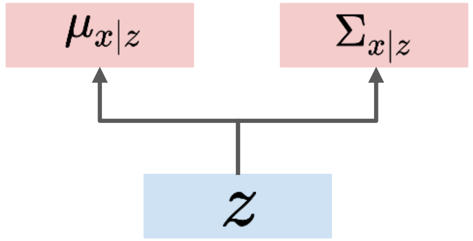
Variational Autoencoders (VAE)

Decoder neural network represent p(x|z) where x is an image, z is latent features to generate x.

Assume prior p(z) is standard unit diagonal Gaussian.

How to train this model?

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



Variational Autoencoders (VAE)

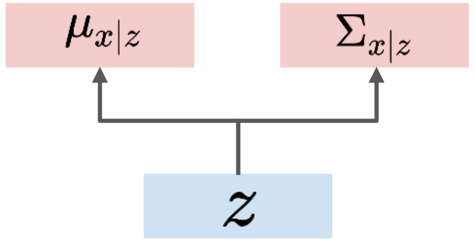
Decoder neural network represent p(x|z) where x is an image, z is latent features to generate x.

Assume prior p(z) is standard unit diagonal Gaussian.

How to train this model?

Maximize likelihood of data

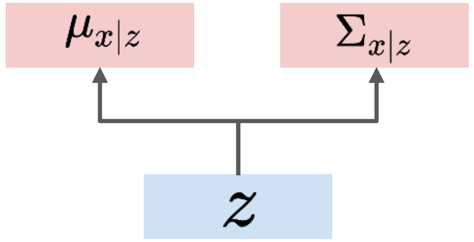
$$p_\theta(x|z) = N(\mu_{x|z}, \Sigma_{x|z})$$



Variational Autoencoders (VAE)

Maximize likelihood of data

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

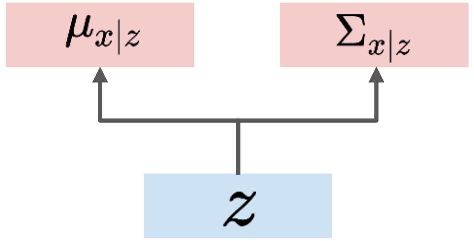


Variational Autoencoders (VAE)

Maximize likelihood of data

$$p_\theta(x) = \int p_\theta(x, z)dz = \int p_\theta(x|z)p_\theta(z)dz$$

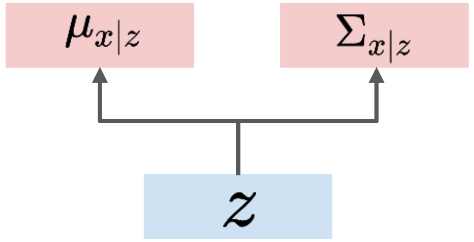
Problem: impossible to integrate over all z



Variational Autoencoders (VAE)

Maximize likelihood of data

Bayes' Rule: $p_\theta(x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}$

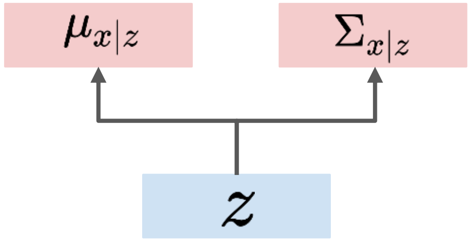


Variational Autoencoders (VAE)

Maximize likelihood of data

Bayes' Rule: $p_\theta(x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}$

Problem: no way to compute $p_\theta(z|x)$



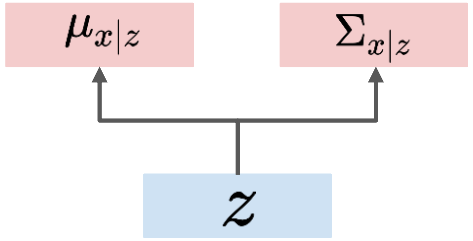
Variational Autoencoders (VAE)

Maximize likelihood of data

Bayes' Rule: $p_\theta(x) = \frac{p_\theta(x|z)p_\theta(z)}{p_\theta(z|x)}$

Problem: no way to compute $p_\theta(z|x)$

Solution: train another network (encoder) that learns $q_\phi(z|x) \approx p_\theta(z|x)$



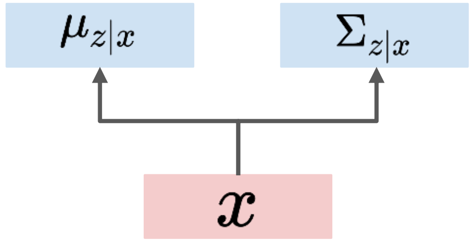
Variational Autoencoders (VAE)

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

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

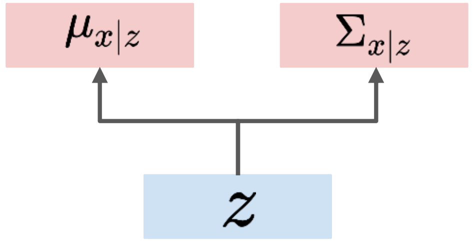
Encoder Network

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



Decoder Network

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

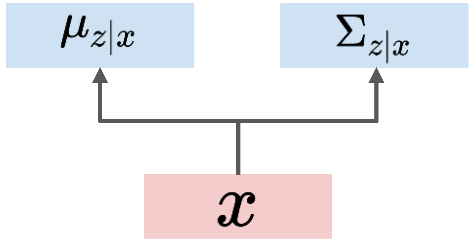


Variational Autoencoders (VAE)

Jointly train **encoder** q and **decoder** p

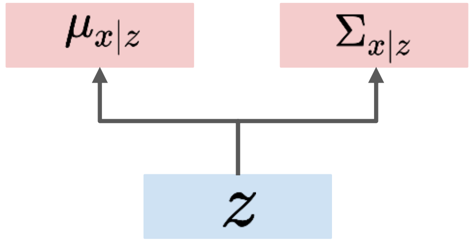
Encoder Network

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



Decoder Network

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



Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)}$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= \log p_\theta(x|z) - \log \frac{q_\phi(z|x)}{p(z)} + \log \frac{q_\phi(z|x)}{p_\theta(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

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

Kullback-Leibler Divergence: $KL(p,q) = E_{x\sim p}[log\frac{p(x)}{q(x)}]$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

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

$KL \geq 0$ => dropping the last term gives a lower bound on the data likelihood

Kullback-Leibler Divergence: $KL(p,q) = E_{x\sim p}[log\frac{p(x)}{q(x)}]$

Variational Autoencoders (VAE)

$$\log p_\theta(x) = \log \frac{p_\theta(x|z)p(z)}{p_\theta(z|x)} = \log \frac{p_\theta(x|z)p(z)q_\phi(z|x)}{p_\theta(z|x)q_\phi(z|x)}$$

$$= E_z[\log p_\theta(x|z)] - E_z[\log \frac{q_\phi(z|x)}{p(z)}] + E_z[\log \frac{q_\phi(z|x)}{p_\theta(z|x)}]$$

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

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

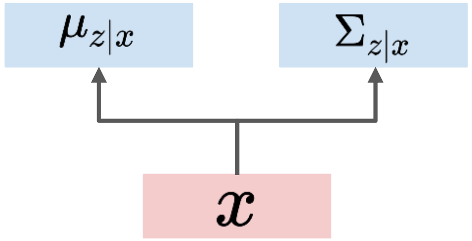
Variational Autoencoders (VAE)

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

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

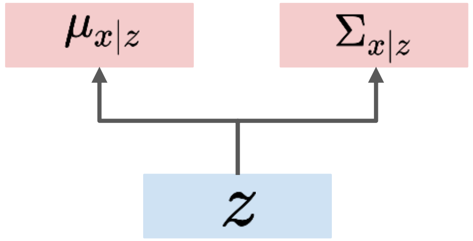
Encoder Network

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



Decoder Network

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



Variational Autoencoders (VAE)

Closed form solution when q_ϕ is diagonal Gaussian and p is unit Gaussian:

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

$$-KL(q_\phi(z|x), p(z)) = \int_Z q_\phi(z|x) \log \frac{p(z)}{q_\phi(z|x)} dz$$

$$= \int_Z N(z, \mu_{z|x}, \Sigma_{z|x}) \log \frac{N(z,0,I)}{N(z,\mu_{z|x},\Sigma_{z|x})} dz$$

$$= \sum_{j=1}^{J} (1 + \log((\sum_{z|x})_j^2) - (\mu_{z|x})_j^2 - (\Sigma_{z|x})_j^2)$$

Variational Autoencoders (VAE)

Closed form solution when q_ϕ is diagonal Gaussian and p is unit Gaussian:

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

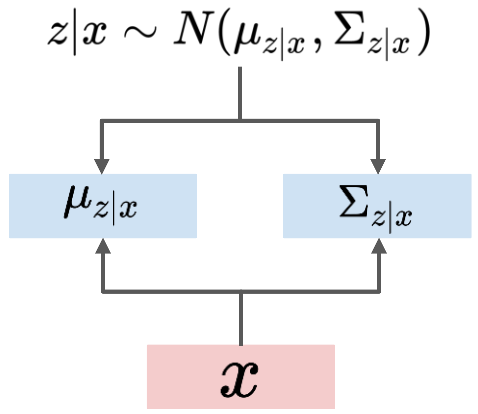
$E_{z\sim q_\phi(z|x)}[\log p_\theta(x|z)]$ is data reconstruction term

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

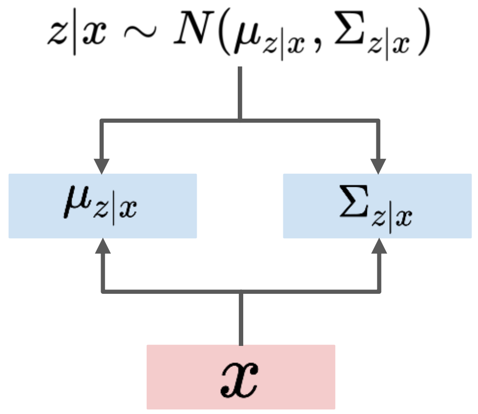


Variational Autoencoders (VAE)

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)**

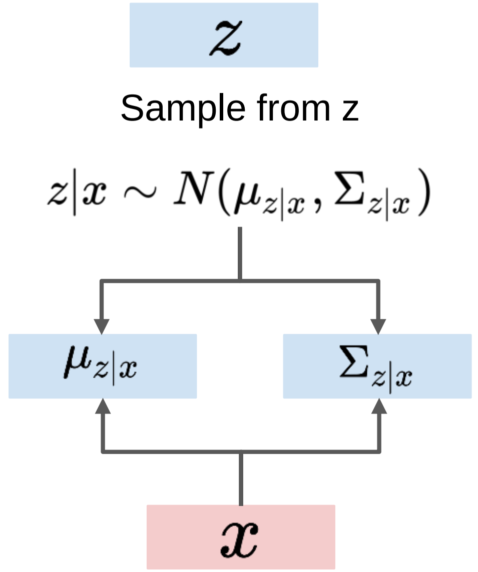


Variational Autoencoders (VAE)

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

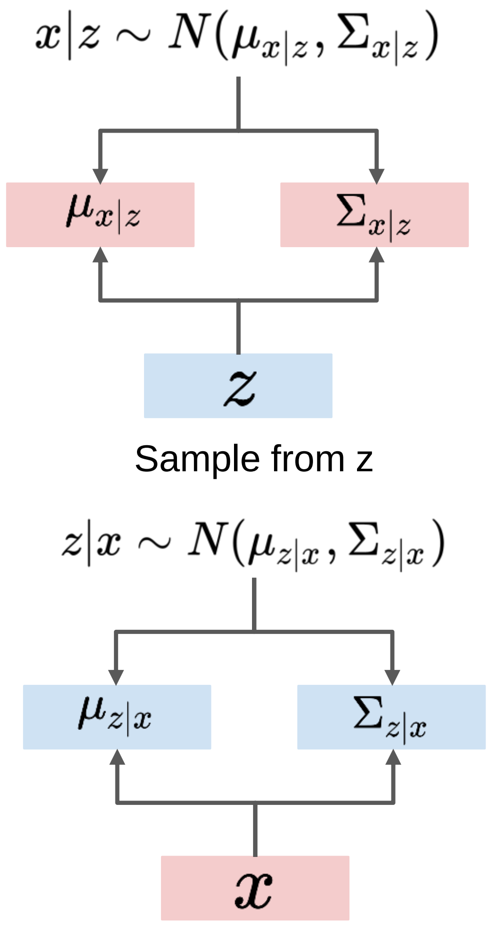


Variational Autoencoders (VAE)

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**



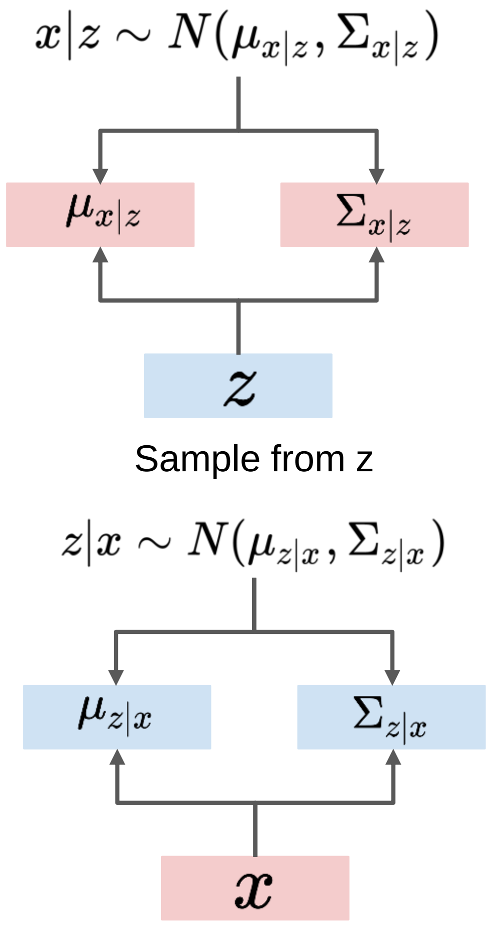
$x|z \sim N(\mu_{x|z}, \Sigma_{x|z})$
$\mu_{x|z}$
$\Sigma_{x|z}$
z
Sample from z
$z|x \sim N(\mu_{z|x}, \Sigma_{z|x})$
$\mu_{z|x}$
$\Sigma_{z|x}$
x

Variational Autoencoders (VAE)

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

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

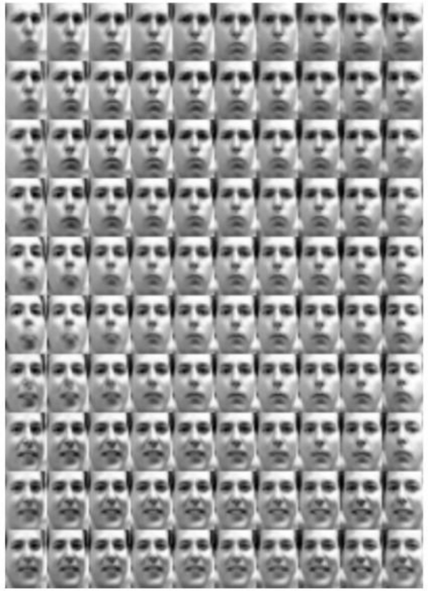
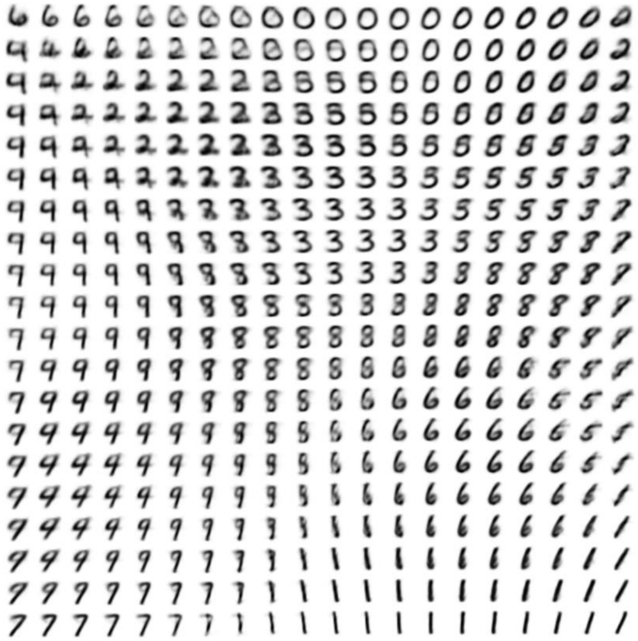
1. The input is **encoded** as distribution over the latent space
2. **Encoder output should match prior p(z)**
3. A point from the latent space is sampled from that distribution
4. The sampled point is **decoded**
5. **The reconstruction error is computed**



$x|z \sim N(\mu_{x|z}, \Sigma_{x|z})$
$\mu_{x|z}$
$\Sigma_{x|z}$
z
Sample from z
$z|x \sim N(\mu_{z|x}, \Sigma_{z|x})$
$\mu_{z|x}$
$\Sigma_{z|x}$
x

Variational Autoencoders (VAE)

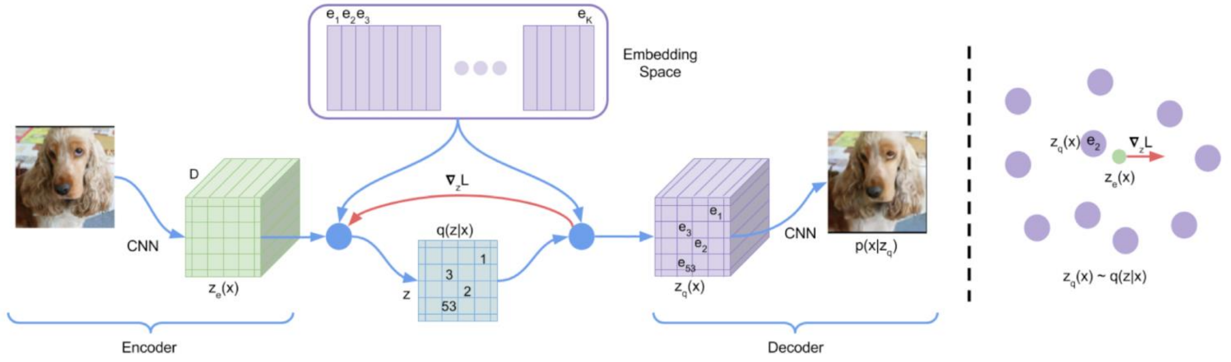
Learned data manifold for generative models with two-dimensional latent space:

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

Discrete Variational Auto-Encoder (dVAE)

- introduced in VQ-VAE 1 [1] and VQ-VAE-2 [2]
- image encoder maps to latent 32x32 grid of embeddings
- vector quantization maps to 8k code words
- decoder maps from quantized grid to the image
- copy gradients from decoder input z to the encoder output

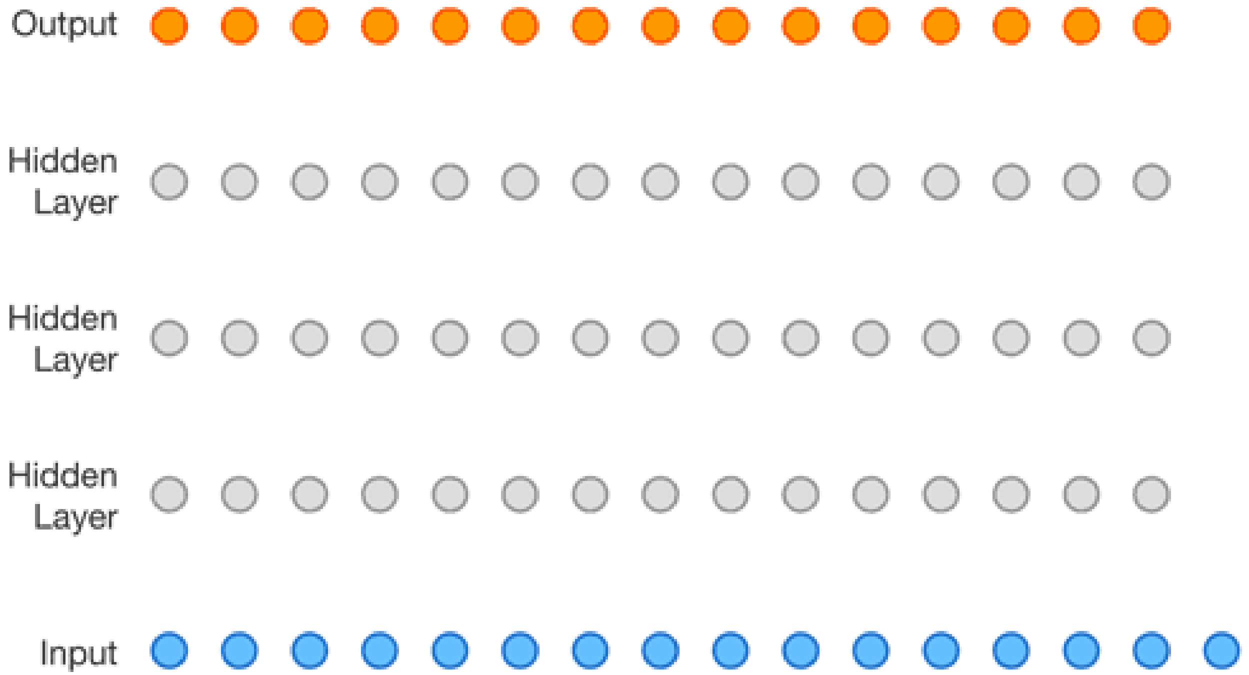


[1] Aaron van den Oord, Oriol Vinyals, Koray Kavukcuoglu. Neural Discrete Representation Learning. NeurIPS 2017 https://arxiv.org/abs/1711.00937

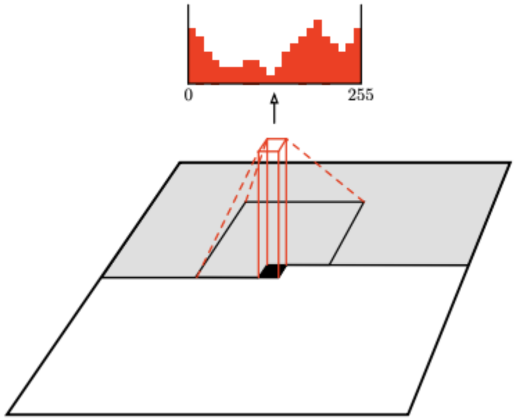
[2] Ali Razavi, Aaron van den Oord, Oriol Vinyals. Generating Diverse High-Fidelity Images with VQ-VAE-2. NeurIPS 2019 https://arxiv.org/abs/1906.00446

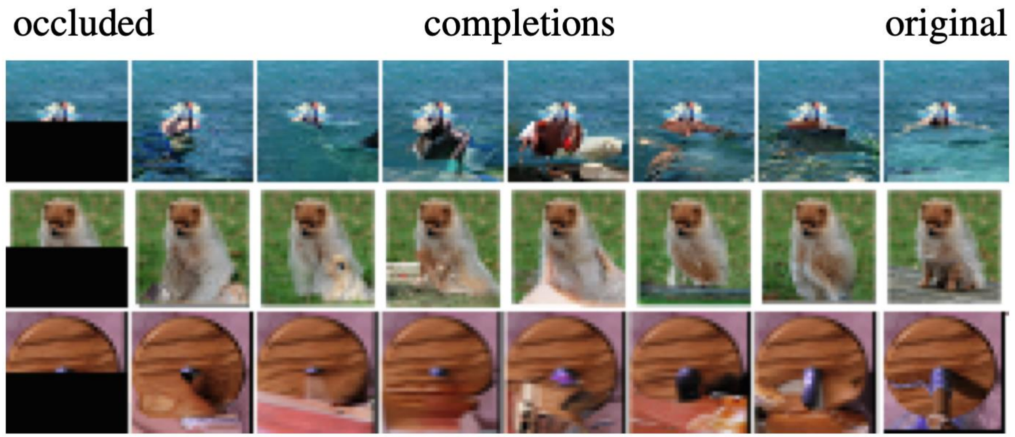
Outline

Autoregressive Models



Pixel Recurrent Neural Networks





Aaron van den Oord, Nal Kalchbrenner, Koray Kavukcuoglu. Pixel Recurrent Neural Networks. ICML 2016

DALL-E 1

- is introduced by OpenAI
- generates 256×256 images from text via dVAE
- autoregressively generates image tokens from textual tokens on a discrete latent space

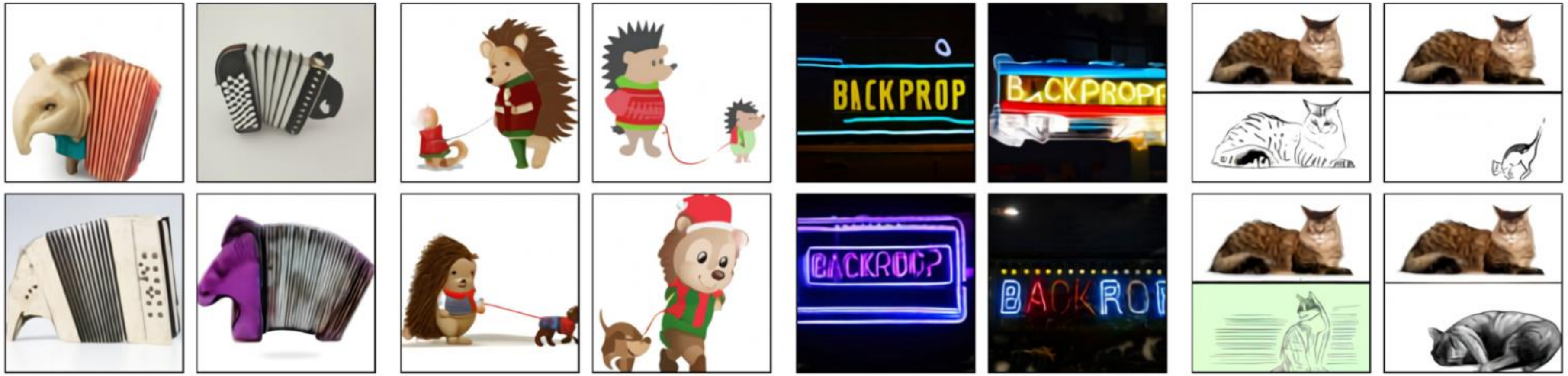
Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever. Zero-Shot Text-to-Image Generation. PMLR 2021
https://arxiv.org/abs/2102.12092

DALL-E 1 Training

1. train encoder and decoder image of image into 32x32 grid of 8k possible code word tokens (dVAE)
2. concatenate encoded text tokens with image tokens into single array
3. train to predict next image token from the preceding tokens (autoregressive transformer)
4. discard the image encoder, keep only image decoder and next token predictor

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever. Zero-Shot Text-to-Image Generation. PMLR 2021
https://arxiv.org/abs/2102.12092

DALL-E 1 Results



(a) a tapir made of accordion. a tapir with the texture of an accordion.

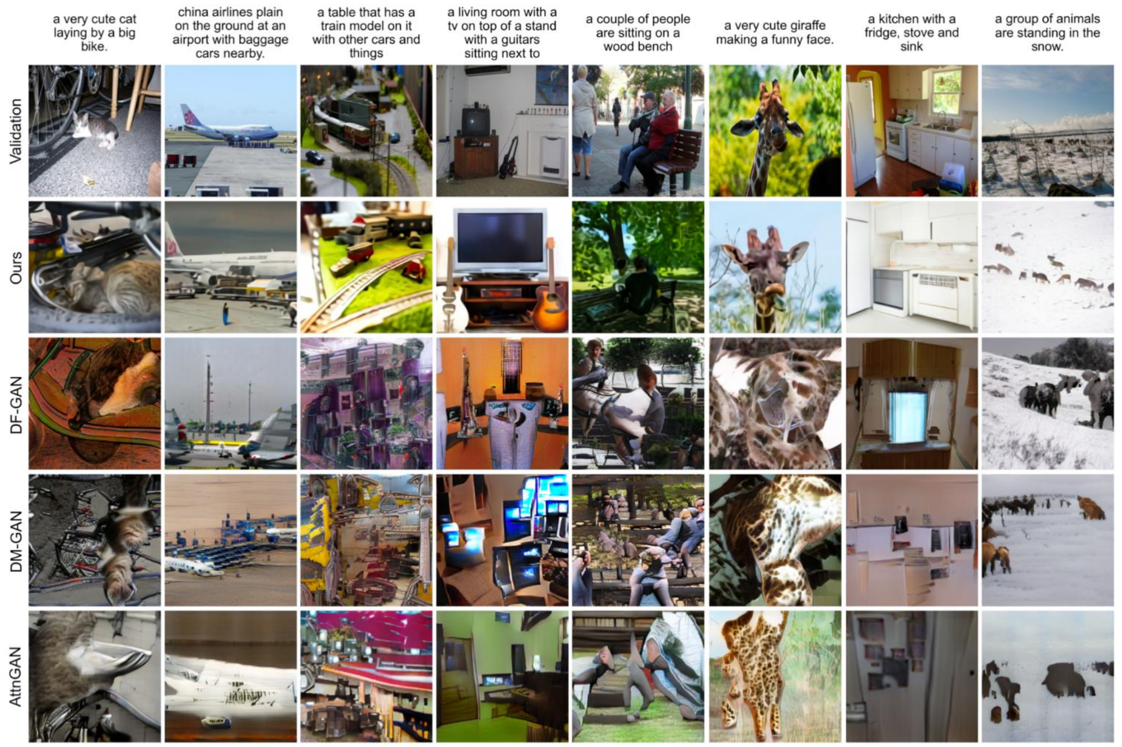
(b) an illustration of a baby hedgehog in a christmas sweater walking a dog

(c) a neon sign that reads "backprop". a neon sign that reads "backprop". backprop neon sign

(d) the exact same cat on the top as a sketch on the bottom

Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever. Zero-Shot Text-to-Image Generation. PMLR 2021
https://arxiv.org/abs/2102.12092

DALL-E 1 Results


a very cute cat laying by a big bike.
china airlines plain on the ground at an airport with baggage cars nearby.
a table that has a train model on it with other cars and things
a living room with a tv on top of a stand with a guitars sitting next to
a couple of people are sitting on a wood bench
a very cute giraffe making a funny face.
a kitchen with a fridge, stove and sink
a group of animals are standing in the snow.
Validation
Ours
DF-GAN
DM-GAN
AttnGAN

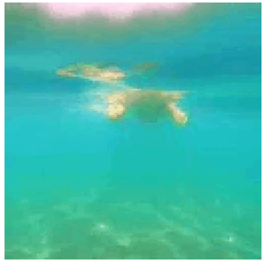
Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, Ilya Sutskever. Zero-Shot Text-to-Image Generation. PMLR 2021
https://arxiv.org/abs/2102.12092

Phenaki

The water is magical

Prompts used:
A photorealistic teddy bear is swimming in the ocean at San Francisco
The teddy bear goes under water
The teddy bear keeps swimming under the water with colorful fishes
A panda bear is swimming under water

Chilling on the beach



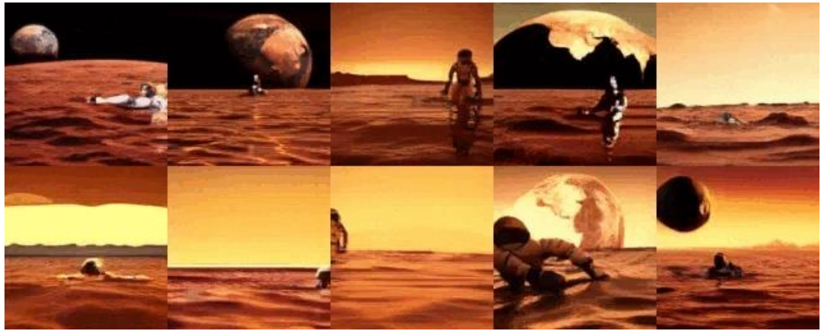
Prompts used:
A teddy bear diving in the ocean
A teddy bear emerges from the water
A teddy bear walks on the beach
Camera zooms out to the teddy bear in the campfire by the beach

Fireworks on the spacewalk

Prompts used:
Side view of an astronaut is walking through a puddle on mars
The astronaut is dancing on mars
The astronaut walks his dog on mars
The astronaut and his dog watch fireworks

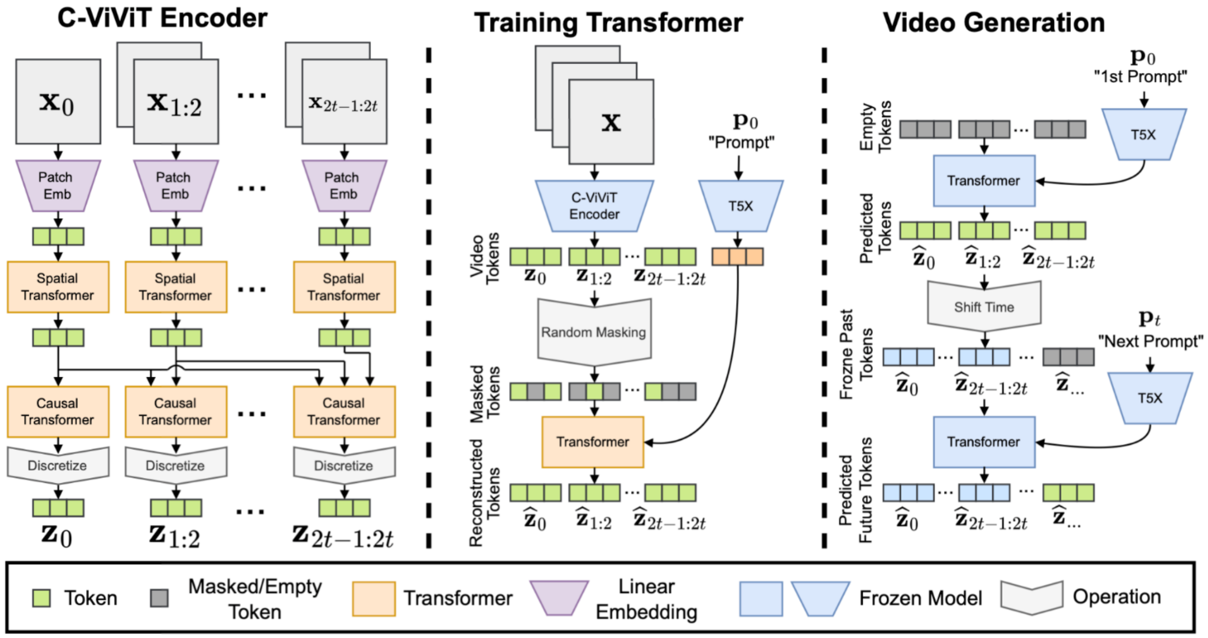
Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, Dumitru Erhan. Phenaki: Variable Length Video Generation From Open Domain Textual Description. 2022 https://arxiv.org/abs/2210.02399
https://phenaki.video

Phenaki



Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, Dumitru Erhan. Phenaki: Variable Length Video Generation From Open Domain Textual Description. 2022 https://arxiv.org/abs/2210.02399
https://phenaki.video

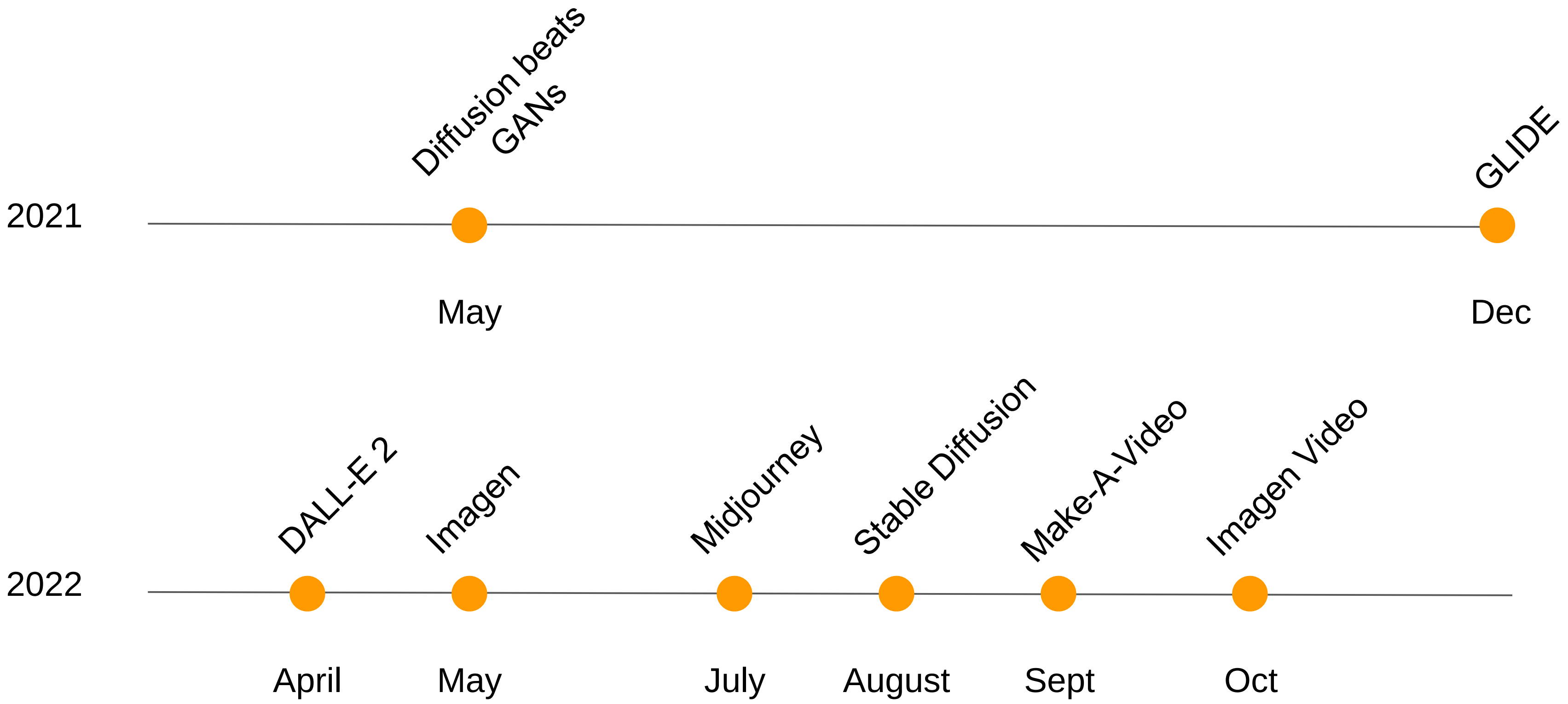
Phenaki



Ruben Villegas, Mohammad Babaeizadeh, Pieter-Jan Kindermans, Hernan Moraldo, Han Zhang, Mohammad Taghi Saffar, Santiago Castro, Julius Kunze, Dumitru Erhan. Phenaki: Variable Length Video Generation From Open Domain Textual Description. 2022 https://arxiv.org/abs/2210.02399
https://phenaki.video

Outline

Diffusion Models Timeline



Diffusion Models

- Diffusion models are inspired by non-equilibrium thermodynamics.
- They define a Markov chain of diffusion steps to slowly add random noise to data and then learn to reverse the diffusion process to construct desired data samples from the noise.
- Unlike VAE or flow models, diffusion models are learned with a fixed procedure and the latent variable has high dimensionality (same as the original data).

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

Slide Credits

CVPR 2022 Tutorial

Denoising Diffusion-based Generative Modeling

by Karsten Kreis, Ruiqi Gao and Arash Vahdat

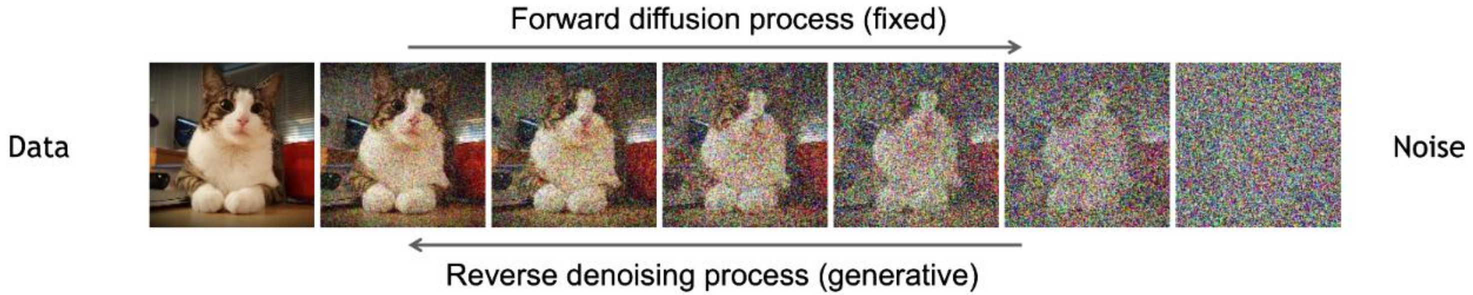
https://cvpr2022-tutorial-diffusion-models.github.io

Denoising Diffusion Models

Learning to generate by denoising

Denoising diffusion models consist of two processes:

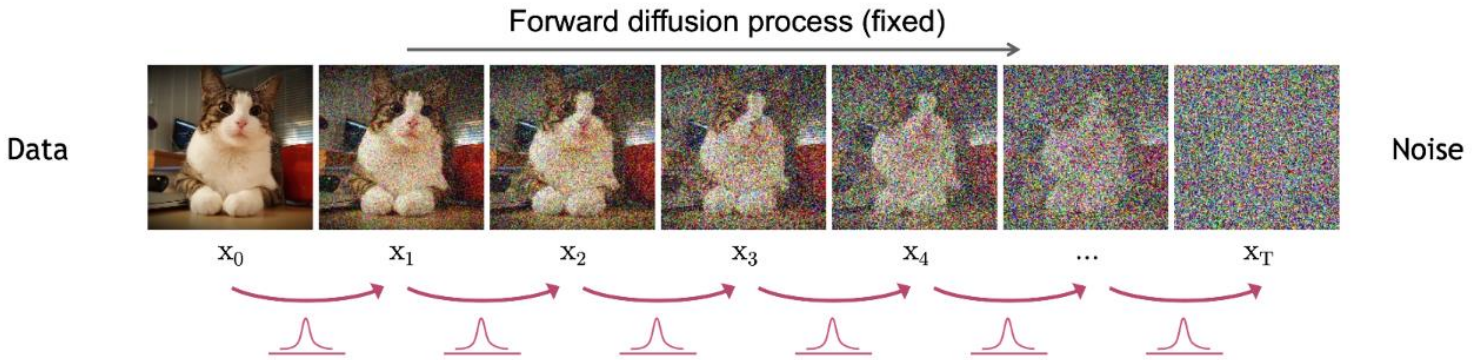
- Forward diffusion process that gradually adds noise to input
- Reverse denoising process that learns to generate data by denoising


Forward diffusion process (fixed)
Data
Noise
Reverse denoising process (generative)

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

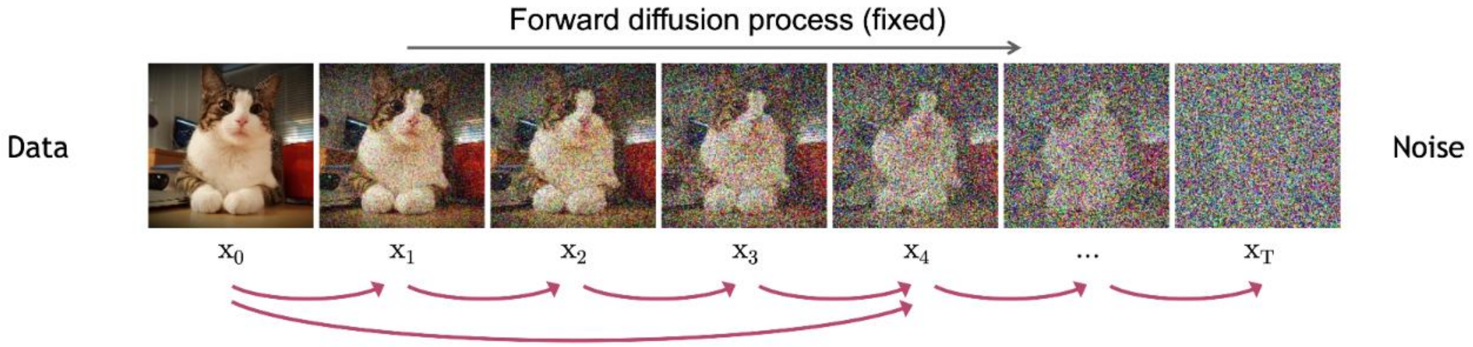
Forward Diffusion Process

The formal definition of the forward process in T steps:



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

Diffusion Kernel



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

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

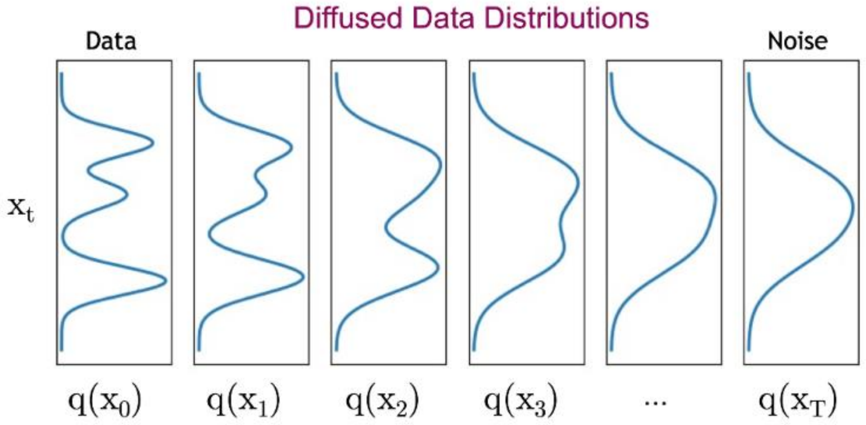
β_t values schedule (i.e., the noise schedule) is designed such that $\bar{\alpha}_T \rightarrow 0$ and $q(\mathbf{x}_T|\mathbf{x}_0) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

What happens to a distribution in the forward diffusion?

So far, we discussed the diffusion kernel $q(\mathbf{x}_t|\mathbf{x}_0)$ but what about $q(\mathbf{x}_t)$?

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

The diffusion kernel is Gaussian convolution.



We can sample $\mathbf{x}_t \sim q(\mathbf{x}_t)$ by first sampling $\mathbf{x}_0 \sim q(\mathbf{x}_0)$ and then sampling $\mathbf{x}_t \sim q(\mathbf{x}_t|\mathbf{x}_0)$ (i.e., ancestral sampling).

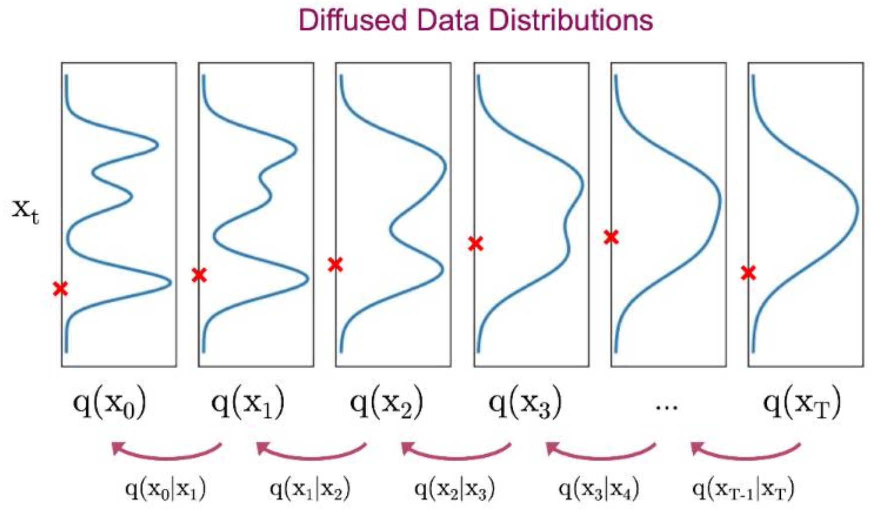
Generative Learning by Denoising

Recall, that the diffusion parameters are designed such that $q(\mathbf{x}_T) \approx \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I}))$

Generation:

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

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

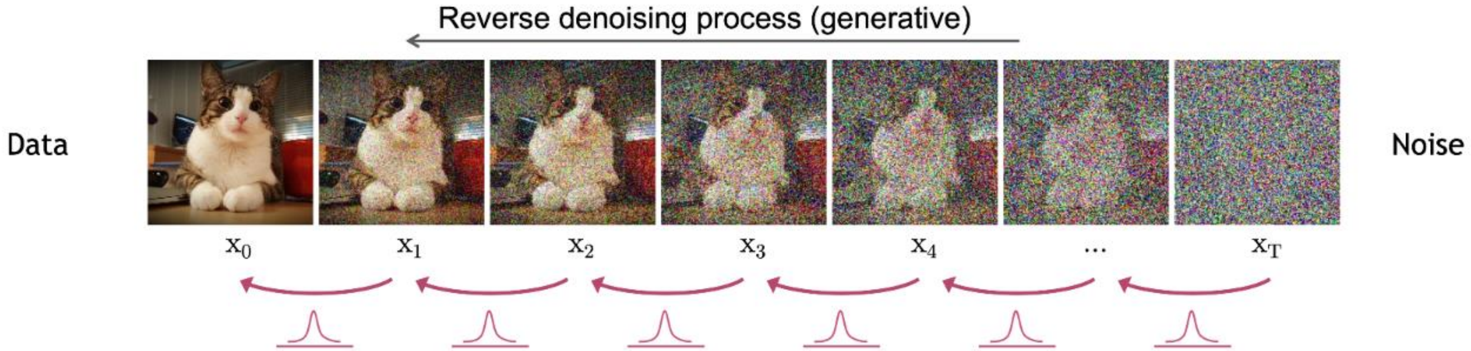


In general, $q(\mathbf{x}_{t-1}|\mathbf{x}_t) \propto q(\mathbf{x}_{t-1})q(\mathbf{x}_t|\mathbf{x}_{t-1})$ is intractable.

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.

Reverse Denoising Process

Formal definition of forward and reverse processes in T steps:



$$p(\mathbf{x}_T) = \mathcal{N}(\mathbf{x}_T; \mathbf{0}, \mathbf{I})$$
$$p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t) = \mathcal{N}(\mathbf{x}_{t-1}; \underbrace{\mu_\theta(\mathbf{x}_t, t)}_{\substack{\text{Trainable network} \\ \text{(U-net, Denoising Autoencoder)}}}, \sigma_t^2\mathbf{I})$$

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

Learning Denoising Model

Variational upper bound

For training, we can form variational upper bound that is commonly used for training variational autoencoders:

$$\mathbb{E}_{q(\mathbf{x}_0)}\left[-\log p_\theta(\mathbf{x}_0)\right] \leq \mathbb{E}_{q(\mathbf{x}_0)q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\left[-\log \frac{p_\theta(\mathbf{x}_{0:T})}{q(\mathbf{x}_{1:T}|\mathbf{x}_0)}\right] =: L$$

Sohl-Dickstein et al. ICML 2015 and Ho et al. NeurIPS 2020 show that:

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

where $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ is the tractable posterior distribution:

$$q(\mathbf{x}_{t-1}|\mathbf{x}_t,\mathbf{x}_0) = \mathcal{N}(\mathbf{x}_{t-1}; \tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0), \tilde{\beta}_t\mathbf{I}),$$

$$\text{where } \tilde{\mu}_t(\mathbf{x}_t,\mathbf{x}_0) := \frac{\sqrt{\bar{\alpha}_{t-1}}\beta_t}{1-\bar{\alpha}_t}\mathbf{x}_0 + \frac{\sqrt{1-\beta_t}(1-\bar{\alpha}_{t-1})}{1-\bar{\alpha}_t}\mathbf{x}_t \text{ and } \tilde{\beta}_t := \frac{1-\bar{\alpha}_{t-1}}{1-\bar{\alpha}_t}\beta_t$$

Parameterizing the Denoising Model

Since both $q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)$ and $p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)$ are Normal distributions, the KL divergence has a simple form:

$$L_{t-1} = D_{\mathrm{KL}}(q(\mathbf{x}_{t-1}|\mathbf{x}_t, \mathbf{x}_0)||p_\theta(\mathbf{x}_{t-1}|\mathbf{x}_t)) = \mathbb{E}_q \left[\frac{1}{2\sigma_t^2}||\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) - \mu_\theta(\mathbf{x}_t, t)||^2\right] + C$$

Recall that $\mathbf{x}_t = \sqrt{\bar{\alpha}_t}\ \mathbf{x}_0 + \sqrt{(1-\bar{\alpha}_t)}\ \epsilon$. Ho et al. NeurIPS 2020 observe that:

$$\tilde{\mu}_t(\mathbf{x}_t, \mathbf{x}_0) = \frac{1}{\sqrt{1-\beta_t}}\left(\mathbf{x}_t - \frac{\beta_t}{\sqrt{1-\bar{\alpha}_t}}\epsilon\right)$$

They propose to 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

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

Training Objective Weighting

Trading likelihood for perceptual quality

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

The time dependent λ_t ensures that the training objective is weighted properly for the maximum data likelihood training.

However, this weight is often very large for small t's.

Ho et al. NeurIPS 2020 observe that simply setting $\lambda_t = 1$ improves sample quality. So, they propose to use:

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

For more advanced weighting see Choi et al., Perception Prioritized Training of Diffusion Models, CVPR 2022.

Summary

Training and Sample Generation

Algorithm 1 Training

```
1: repeat
2:   $\mathbf{x}_0 \sim q(\mathbf{x}_0)$
3:   $t \sim \mathrm{Uniform}(\{1, \ldots, T\})$
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
```

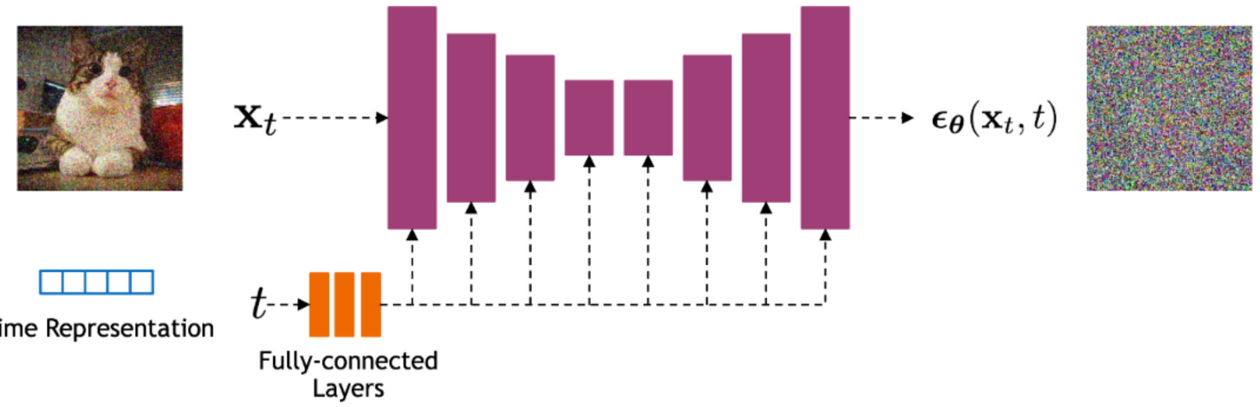
Algorithm 2 Sampling

```
1: $\mathbf{x}_T \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$
2: for $t = T, \ldots, 1$ 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 $\mathbf{x}_0$
```

Implementation Considerations

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)

Summary

Denoising Diffusion Probabilistic Models

In this part, we reviewed denoising diffusion probabilistic models.

The model is trained by sampling from the forward diffusion process and training a denoising model to predict the noise.

We discussed how the forward process perturbs the data distribution or data samples.

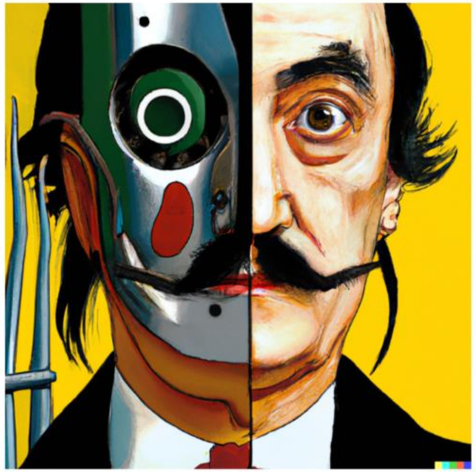
The devil is in the details:

- Network architectures
- Objective weighting
- Diffusion parameters (i.e., noise schedule)

See “Elucidating the Design Space of Diffusion-Based Generative Models” by Karras et al. for important design decisions.

DALL-E 2

- is introduced by OpenAI
- generates 1024 x 1024 images from text using diffusion models.


vibrant portrait painting of Salvador Dalí with a robotic half face

a shiba inu wearing a beret and black turtleneck

a close up of a handpalm with leaves growing from it

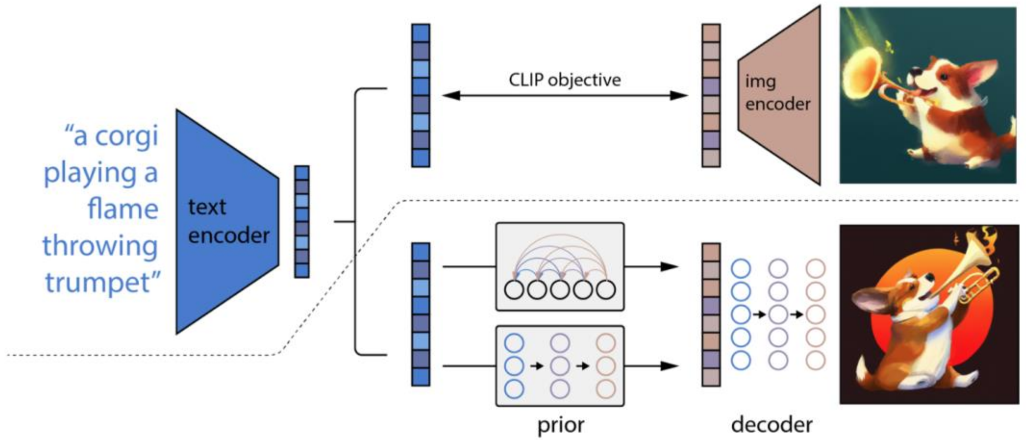
Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen.

Hierarchical Text-Conditional Image Generation with CLIP Latents. 2022 https://arxiv.org/abs/2204.06125

https://openai.com/product/dall-e-2

DALL-E 2

1. generates a CLIP text embedding for text caption
2. “prior” network generates CLIP image embedding from text embedding
3. diffusion decoder generates image from the image embedding



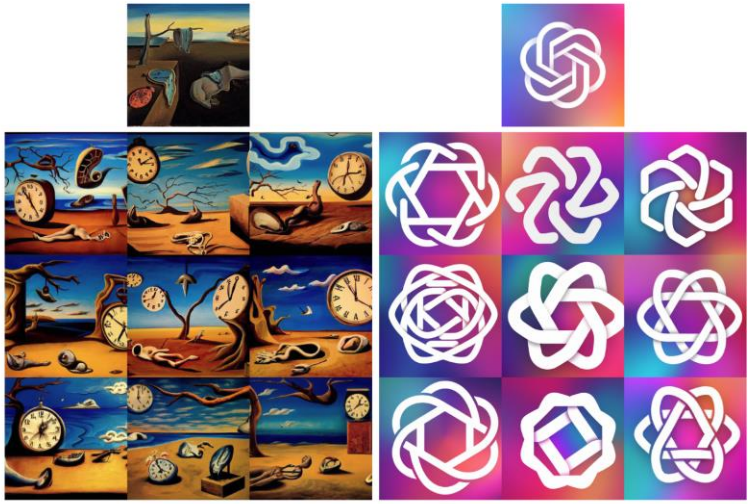
Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen.

Hierarchical Text-Conditional Image Generation with CLIP Latents. 2022 https://arxiv.org/abs/2204.06125

https://openai.com/product/dall-e-2

DALL-E 2 training

Can vary images while preserving style and semantics in the embeddings



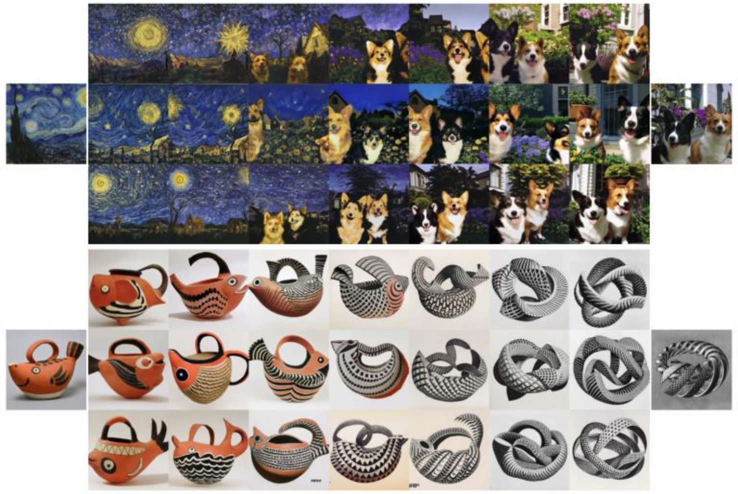


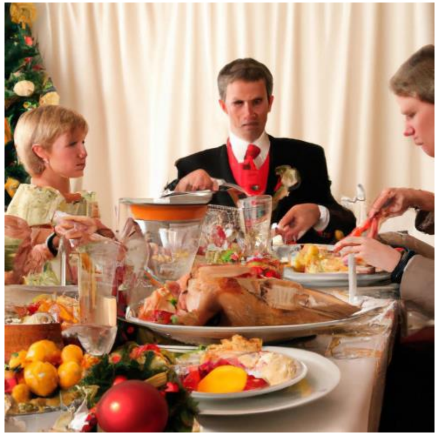
Figure 4: Variations between two images by interpolating their CLIP image embedding and then decoding with a diffusion model. We fix the decoder seed across each row. The intermediate variations naturally blend the content and style from both input images.

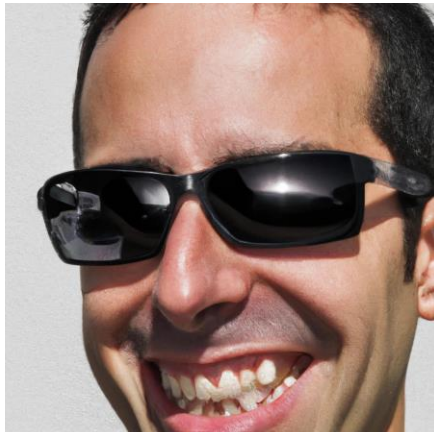
Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen.

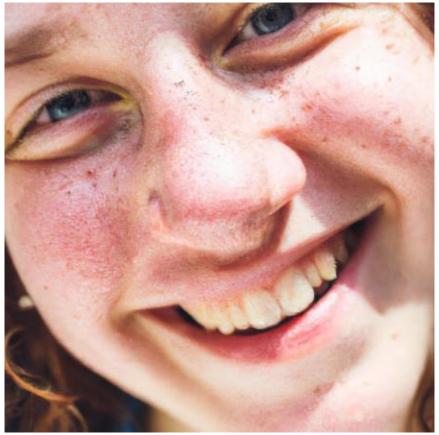
Hierarchical Text-Conditional Image Generation with CLIP Latents. 2022 https://arxiv.org/abs/2204.06125

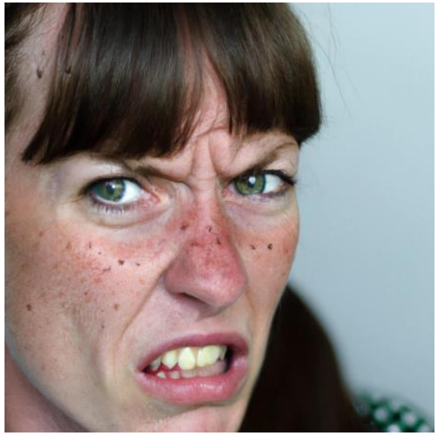
https://openai.com/product/dall-e-2

DALL-E 2 Limitations

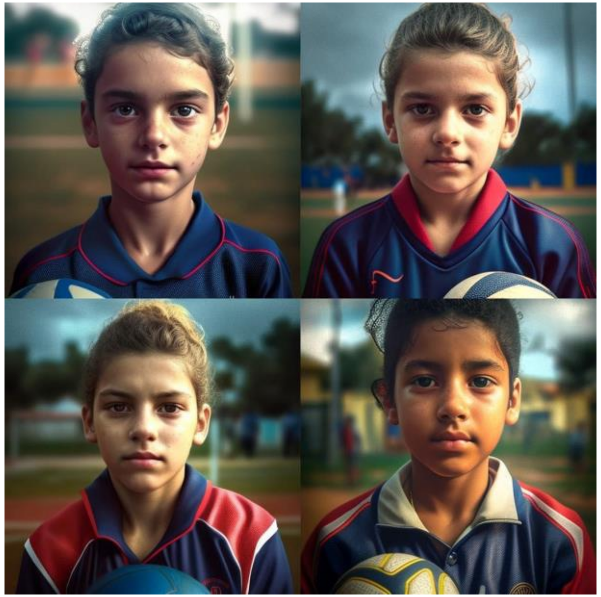

A family dining for Christmas

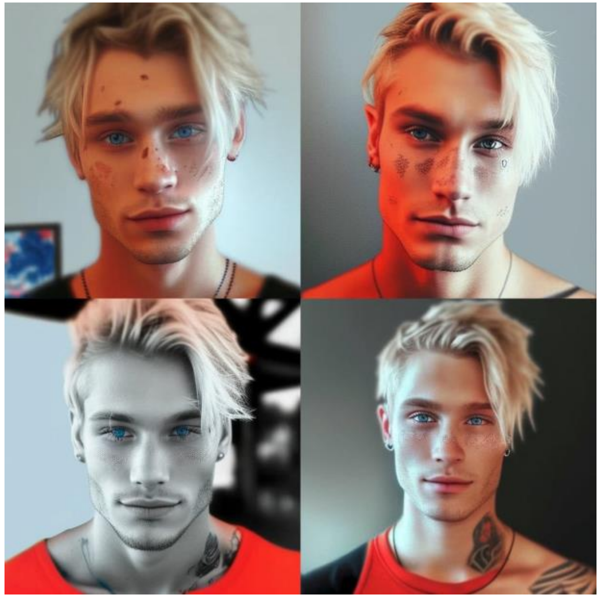

A hyperrealistic man's face smiling with a pair of sunglasses

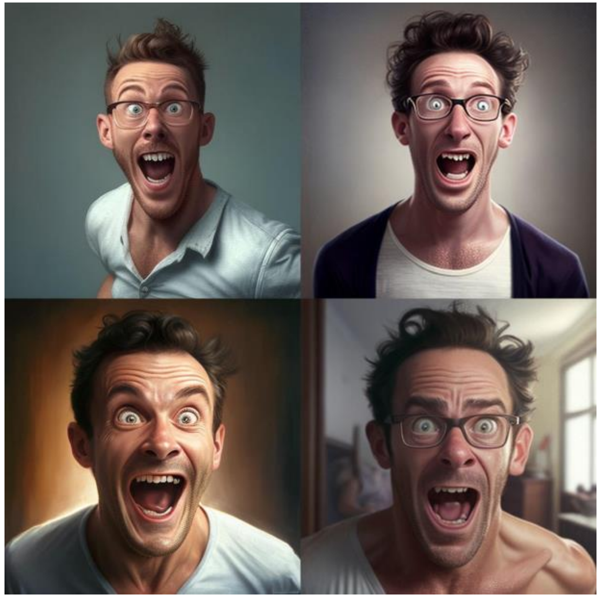

A green-eyed woman with freckles showing happines


A green-eyed woman with freckles showing disgust

Midjourney


Capture the essence of a young footballer wearing the Paris Saint-Germain cinematic uniform through an extreme close-up portrait, focusing on the intricate details of her face and expressions, very detailed, Octane Render, cinematic, digital art


animated blonde blue eyes love tattoo soft male guy enhanced by ai smile defined --q 2 --s 750

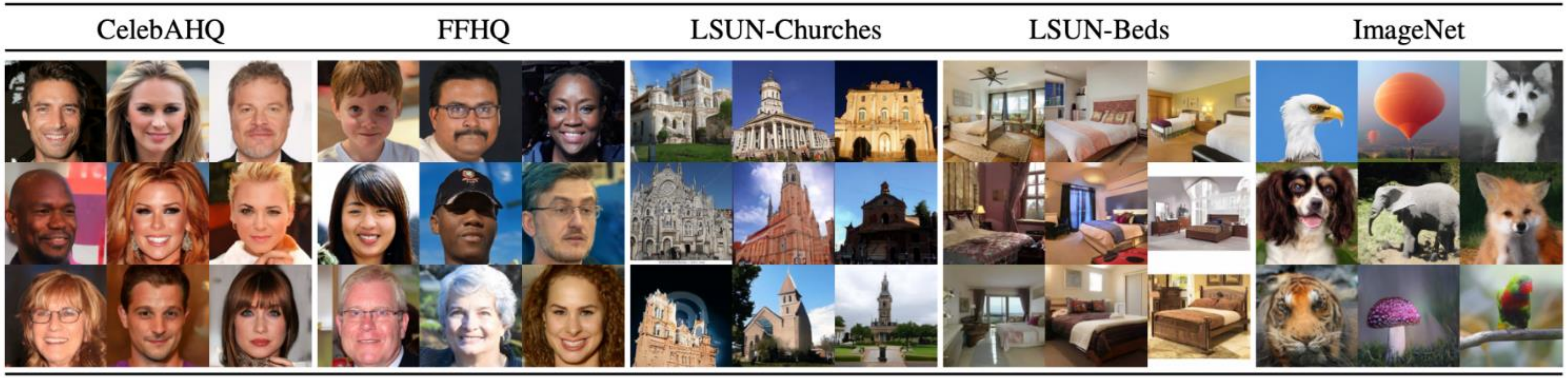

surprised and happy man realistic

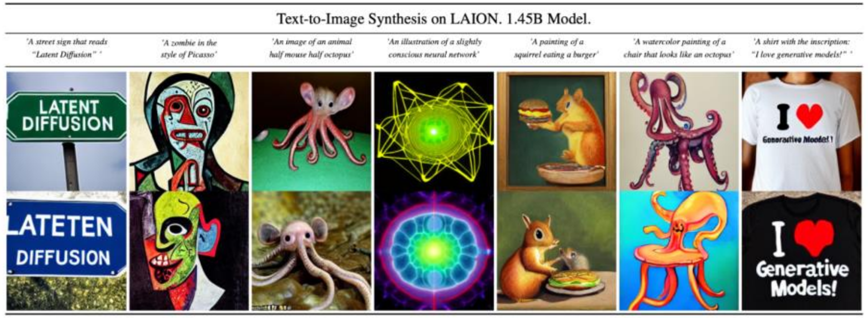
Midjourney

eiffel tower in space with an alien on top

model and robot in fashion show, public, cyberpunk,small robot ,red road , cyberpunk,8k rendering

Latent Diffusion Models

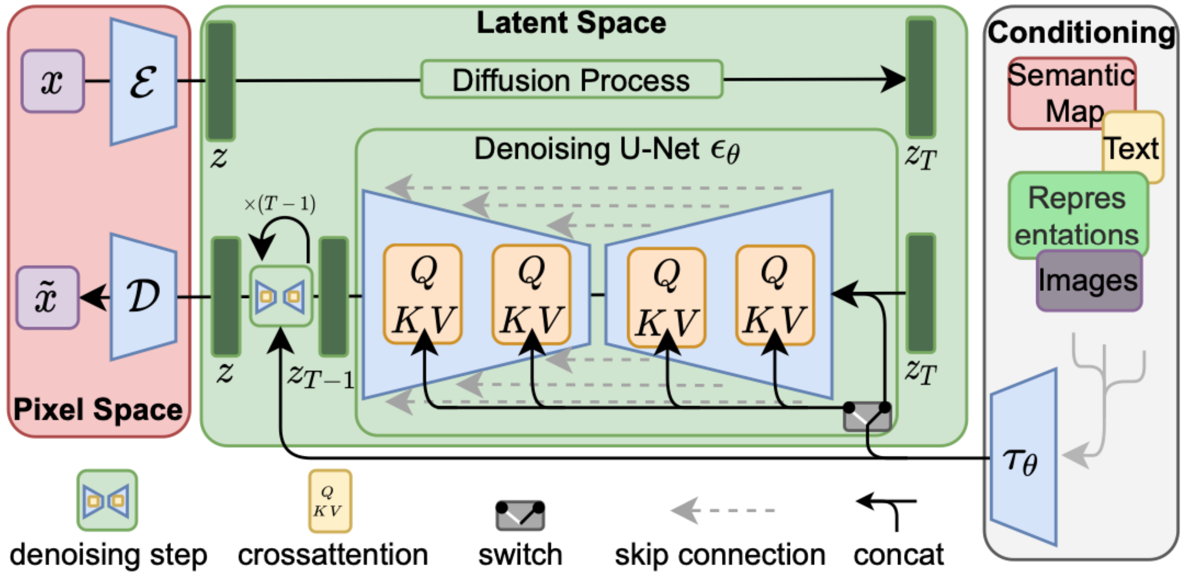



Text-to-Image Synthesis on LAION. 1.45B Model.
'A street sign that reads "Latent Diffusion" '
'A zombie in the style of Picasso'
'An image of an animal half mouse half octopus'
'An illustration of a slightly conscious neural network'
'A painting of a squirrel eating a burger'
'A watercolor painting of a chair that looks like an octopus'
'A shirt with the inscription: "I love generative models!" '
LATENT DIFFUSION
LATETEN DIFFUSION
I Generative Models!

Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer.

High-Resolution Image Synthesis with Latent Diffusion Models. 2022 https://arxiv.org/abs/2112.10752

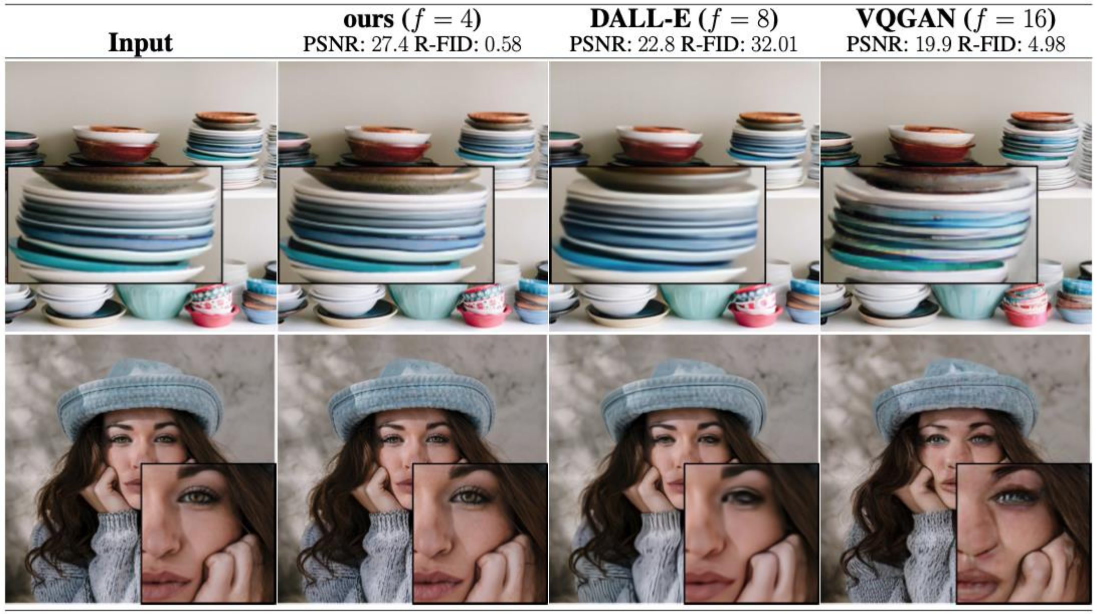
Latent Diffusion Models



Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer.

High-Resolution Image Synthesis with Latent Diffusion Models. 2022 https://arxiv.org/abs/2112.10752

Latent Diffusion Models



Robin Rombach, Andreas Blattmann, Dominik Lorenz, Patrick Esser, Björn Ommer.

High-Resolution Image Synthesis with Latent Diffusion Models. 2022 https://arxiv.org/abs/2112.10752

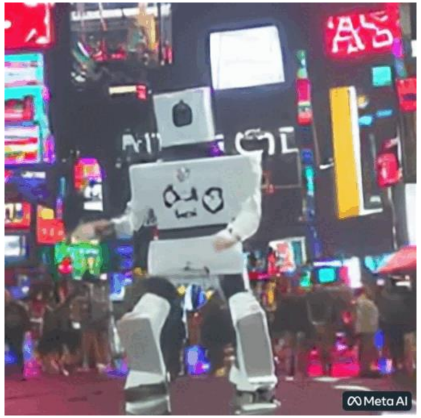
Stable Diffusion

Prompt: realistic smiling woman

https://stablediffusionweb.com

Make-A-Video

A teddy bear painting a portrait



Robot dancing in times square

Cat watching TV with a remote in hand

A fluffy baby sloth with an orange knitted hat trying to figure out a laptop close up highly detailed studio lighting screen reflecting in its eye

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, Yaniv Taigman.

Make-A-Video: Text-to-Video Generation without Text-Video Data. 2022 https://arxiv.org/abs/2209.14792

https://makeavideo.studio

Make-A-Video architecture

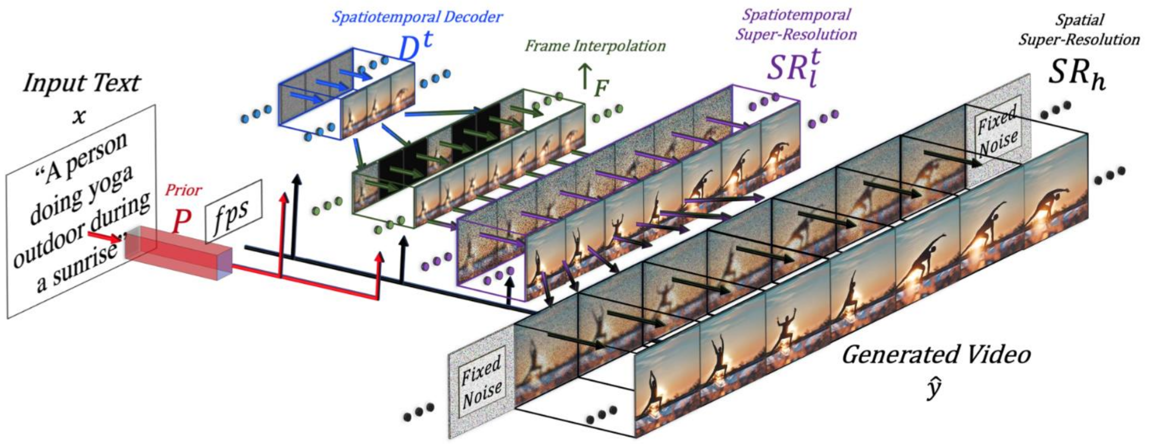


Figure 2: **Make-A-Video high-level architecture.** Given input text x translated by the prior P into an image embedding, and a desired frame rate fps, the decoder D^t generates 16 64×64 frames, which are then interpolated to a higher frame rate by $\uparrow_F$, and increased in resolution to 256×256 by SR_l^t and 768×768 by SR_h, resulting in a high-spatiotemporal-resolution generated video $\hat{y}$.

Uriel Singer, Adam Polyak, Thomas Hayes, Xi Yin, Jie An, Songyang Zhang, Qiyuan Hu, Harry Yang, Oron Ashual, Oran Gafni, Devi Parikh, Sonal Gupta, Yaniv Taigman.

Make-A-Video: Text-to-Video Generation without Text-Video Data. 2022 https://arxiv.org/abs/2209.14792

https://makeavideo.studio

Outline

- Introduction
- Variational Autoencoders
- Autoregressive Models
- Diffusion Models
- Generative Adversarial Networks
    - Architecture and training objective
    - Image generation
    - Video generation
    - Problems of GANs
- Evaluation

Generative Adversarial Networks

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

Generative Adversarial Networks

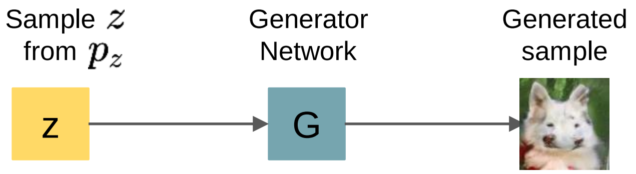
- Setup: Assume we have data x_i drawn from distribution $p_{data}(x)$. Want to sample from p_{data}.
- Idea: Introduce a latent variable z with simple prior $p(z)$.
- Sample $z \sim p(z)$ and pass to a Generator Network $x = G(z)$

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 $p_G = p_{data}$

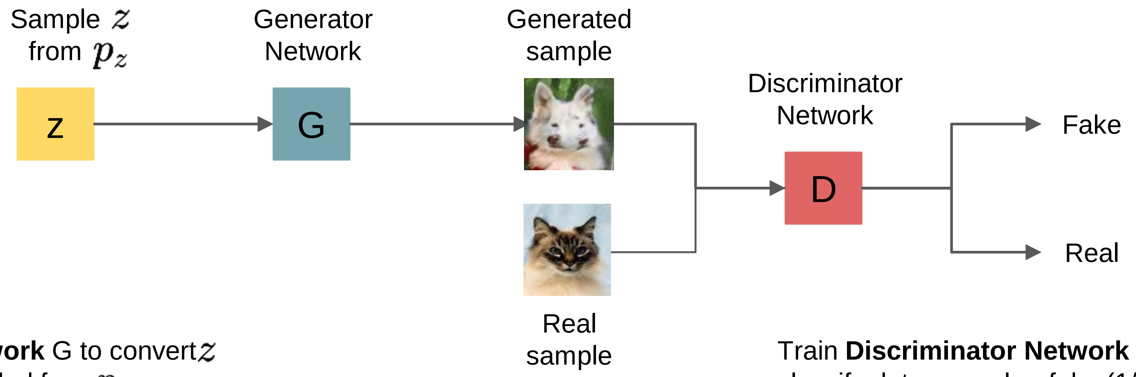
Generative Adversarial Networks

- 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 $p_G = p_{data}$



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

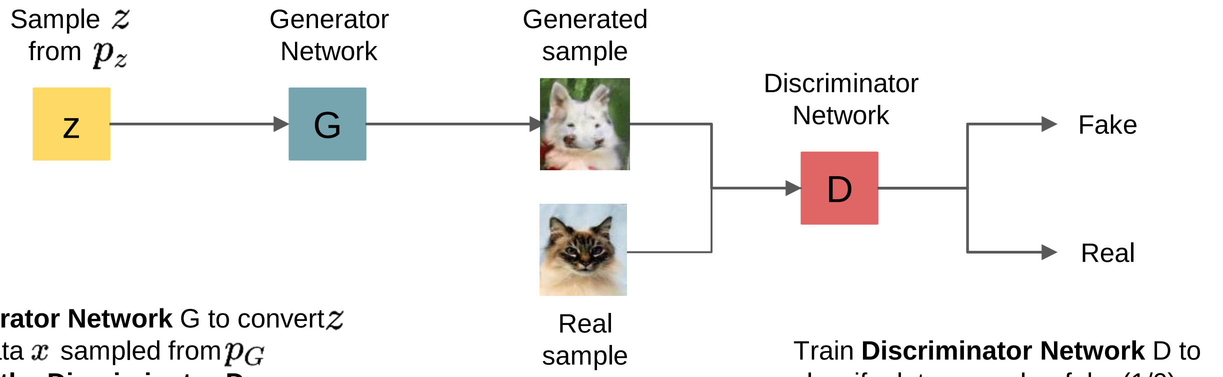
Train **Discriminator Network** D to classify data as real or fake (1/0)

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (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)

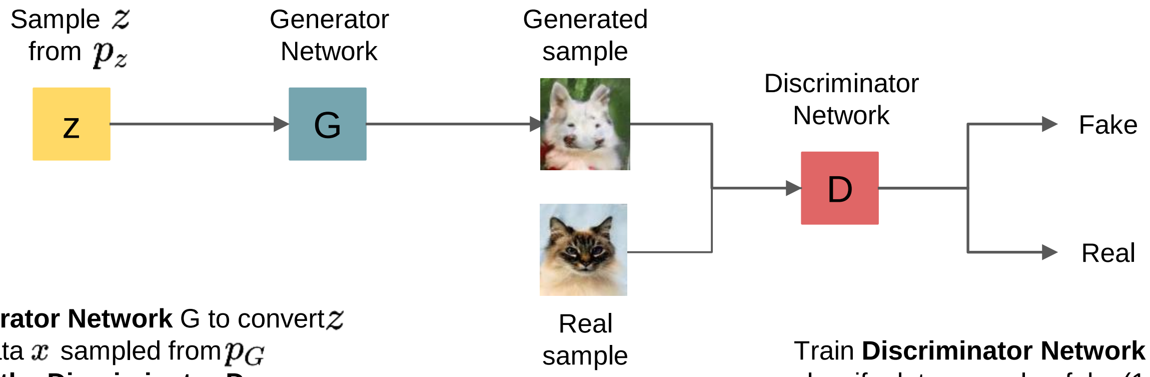
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Training Objective

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

Discriminator wants D(x)=1 for real data

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



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)

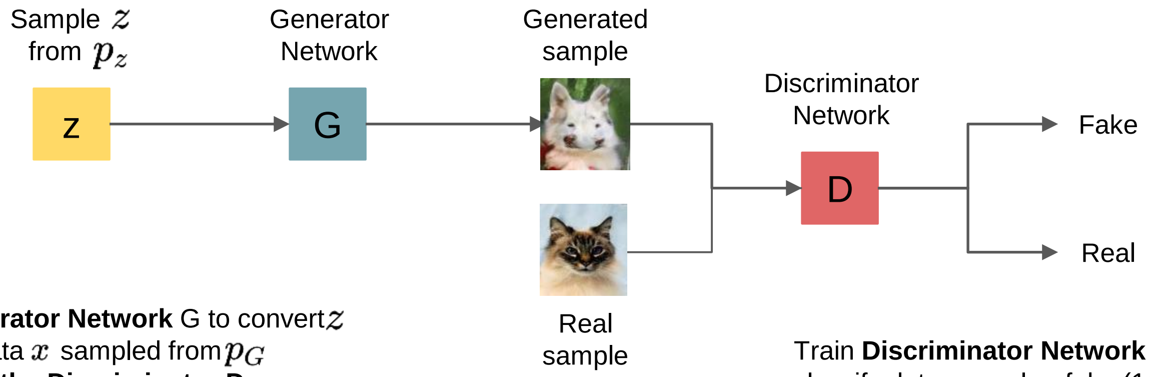
Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Training Objective

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

Discriminator wants D(x)=0 for fake data

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



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

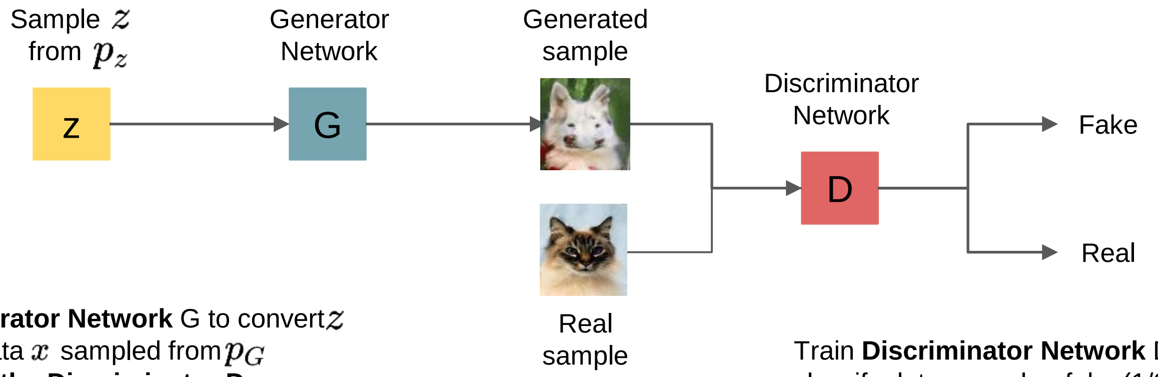
Train **Discriminator Network** D to classify data as real or fake (1/0)

Generative Adversarial Networks: Training Objective

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

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

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)

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (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 $\mathbf{G} = \mathbf{G} - \alpha_{\mathbf{G}} \frac{\delta \mathbf{V}}{\delta \mathbf{G}}$

Generative Adversarial Networks: Optimality

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

Generative Adversarial Networks: Optimality

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

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

Generative Adversarial Networks: Optimality

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

$$= \min_{G}\max_{D}(E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_{G}\max_{D}\int_X (p_{data}(x)\log D(x) + p_G(x)\log(1 - D(x)))dx$$

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x \sim p_{data}} [\log D(x)] + E_{x \sim p_G} [\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$f(y) = a \log y + b \log(1 - y)$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$f(y) = a \log y + b \log(1 - y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$f(y) = a \log y + b \log(1 - y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}$$

$$f'(y) = 0 \Leftrightarrow y = \frac{a}{a+b}$$

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \max_D \int_X (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$= \min_G \int_X \max_D (p_{data}(x) \log D(x) + p_G(x) \log(1 - D(x)))dx$$

$$f(y) = a \log y + b \log(1 - y)$$

$$f'(y) = \frac{a}{y} - \frac{b}{1-y}$$

$$f'(y) = 0 \Leftrightarrow y = \frac{a}{a+b}$$

Optimal Discriminator:

$$D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$$

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \int_X (p_{data}(x) \log D^*_G(x) + p_G(x) \log(1 - D^*_G(x)))dx$$

Optimal Discriminator: $D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{x\sim p_G}[\log(1 - D(x))])$$

$$= \min_G \int_X (p_{data}(x) \log D_G^*(x) + p_G(x) \log(1 - D_G^*(x)))dx$$

$$= \min_G \int_X (p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x)+p_G(x)})dx$$

Optimal Discriminator: $D_G^*(x) = \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

$$\min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))])$$
$$= \min_G \int_X (p_{data}(x) \log D_G^*(x) + p_G(x) \log(1 - D_G^*(x)))dx$$
$$= \min_G \int_X (p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x)+p_G(x)})dx$$
$$= \min_G (E_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{p_G(x)}{p_{data}(x)+p_G(x)}])$$

Generative Adversarial Networks: Optimality

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

$$= \min_G \int_X (p_{data}(x) \log D_G^*(x) + p_G(x) \log(1 - D_G^*(x)))dx$$

$$= \min_G \int_X (p_{data}(x) \log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)} + p_G(x) \log \frac{p_G(x)}{p_{data}(x)+p_G(x)})dx$$

$$= \min_G (E_{x\sim p_{data}}[\log \frac{p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{p_G(x)}{p_{data}(x)+p_G(x)}])$$

$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: Optimality

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

$$= \min_G (E_{x\sim p_{data}}[\log \tfrac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \tfrac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

Generative Adversarial Networks: Optimality

$$\min_G \max_D (E_{x \sim p_{data}}[\log D(x)] + E_{z \sim p(z)}[\log(1 - D(G(z)))])$$
$$= \min_G (E_{x \sim p_{data}}[\log \frac{2 \times p_{data}(x)}{p_{data}(x) + p_G(x)}] + E_{x \sim p_G}[\log \frac{2 \times p_G(x)}{p_{data}(x) + p_G(x)}] - \log 4)$$

Kullback-Leibler Divergence: $KL(p, q) = E_{x \sim p}[log \frac{p(x)}{q(x)}]$

Generative Adversarial Networks: Optimality

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

$$= \min_G (E_{x\sim p_{data}}[\log \tfrac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \tfrac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

$$= \min_G (KL(p_{data}, \tfrac{p_{data}+p_G}{2}) + KL(p_G, \tfrac{p_{data}+p_G}{2}) - \log 4)$$

Kullback-Leibler Divergence: $KL(p, q) = E_{x\sim p}[log\frac{p(x)}{q(x)}]$

Generative Adversarial Networks: Optimality

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

$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$

$$= \min_G (KL(p_{data}, \frac{p_{data}+p_G}{2}) + KL(p_G, \frac{p_{data}+p_G}{2}) - \log 4)$$

Jensen-Shannon Divergence: $JSD(p,q) = \frac{1}{2}KL(p, \frac{p+q}{2}) + \frac{1}{2}KL(q, \frac{p+q}{2})$

Generative Adversarial Networks: Optimality

$$\begin{aligned}
&\min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))]) \\
&= \min_G (E_{x\sim p_{data}}[\log \tfrac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \tfrac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4) \\
&= \min_G (KL(p_{data}, \tfrac{p_{data}+p_G}{2}) + KL(p_G, \tfrac{p_{data}+p_G}{2}) - \log 4) \\
&= \min_G (2 \times JSD(p_{data}, p_G) - \log 4)
\end{aligned}$$

Jensen-Shannon Divergence: $JSD(p,q) = \frac{1}{2}KL(p, \frac{p+q}{2}) + \frac{1}{2}KL(q, \frac{p+q}{2})$

Generative Adversarial Networks: Optimality

$$\min_G \max_D (E_{x\sim p_{data}}[\log D(x)] + E_{z\sim p(z)}[\log(1 - D(G(z)))])$$
$$= \min_G (E_{x\sim p_{data}}[\log \frac{2\times p_{data}(x)}{p_{data}(x)+p_G(x)}] + E_{x\sim p_G}[\log \frac{2\times p_G(x)}{p_{data}(x)+p_G(x)}] - \log 4)$$
$$= \min_G (KL(p_{data}, \frac{p_{data}+p_G}{2}) + KL(p_G, \frac{p_{data}+p_G}{2}) - \log 4)$$
$$= \min_G (2 \times JSD(p_{data}, p_G) - \log 4)$$

JSD is always nonnegative and zero when the two distributions are equal

=> the global minimum is $p_{data} = p_G$

Jensen-Shannon Divergence: $JSD(p,q) = \frac{1}{2}KL(p, \frac{p+q}{2}) + \frac{1}{2}KL(q, \frac{p+q}{2})$

Generative Adversarial Networks: Optimality

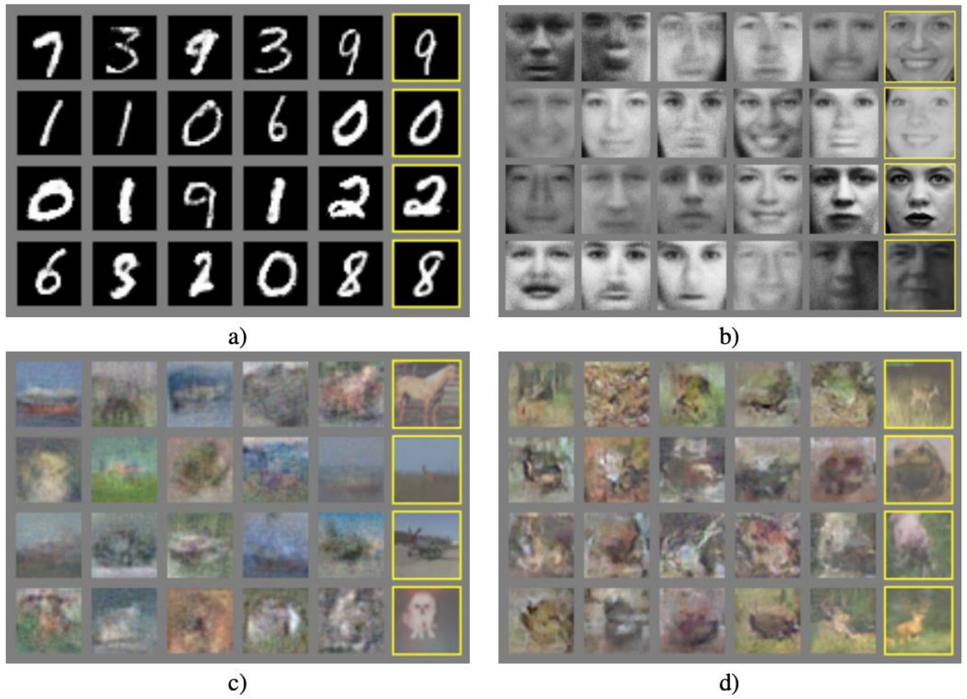
$$\min_G \max_D \left(E_{x \sim p_{data}}[\log D(x)] + E_{z \sim p(z)}[\log(1 - D(G(z)))]\right)$$

$$= \min_G \left(2 * JSD(p_{data}, p_G) - \log 4\right)$$

Summary: The global minimum of the minimax game happens when:

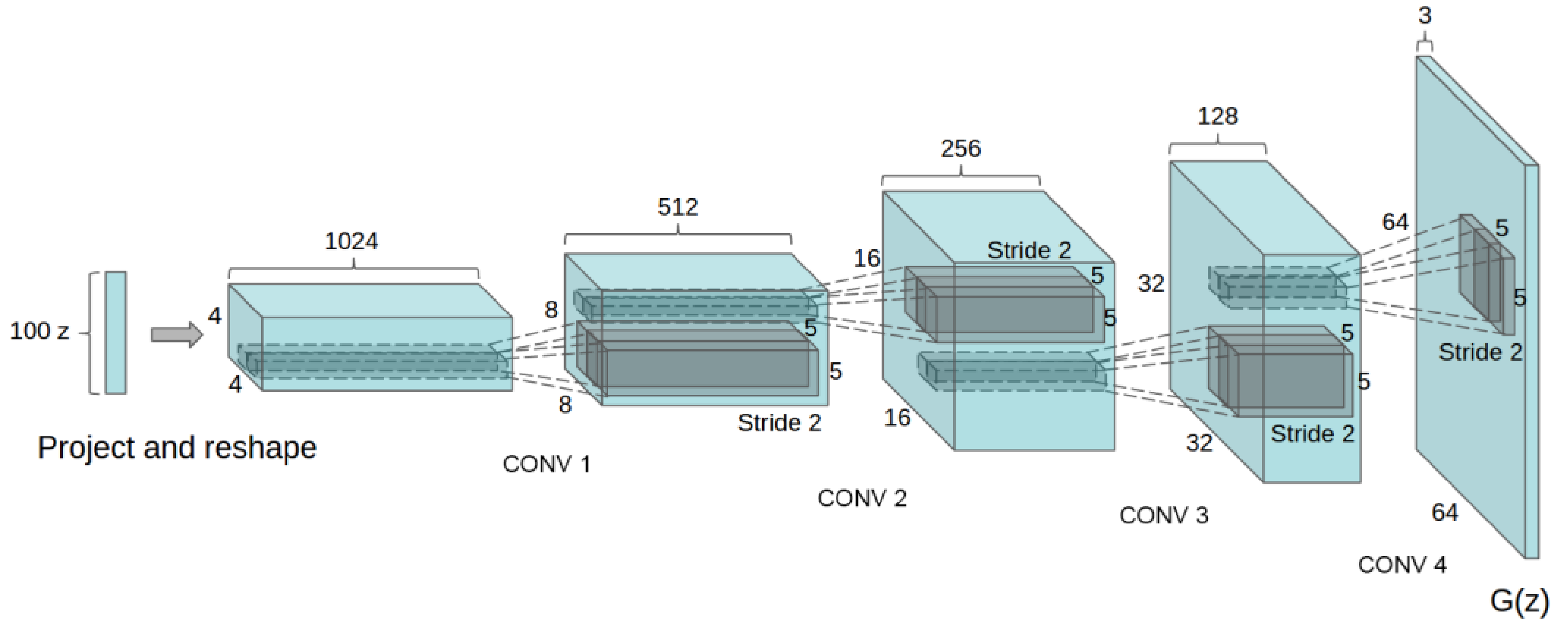
1. $D^*_G(x) = \frac{p_{data}(x)}{p_{data}(x) + p_G(x)}$ (Optimal discriminator for any G)
2. $p_G(x) = p_{data}(x)$ (Optimal generator for optimal D)

Generative Adversarial Networks: results

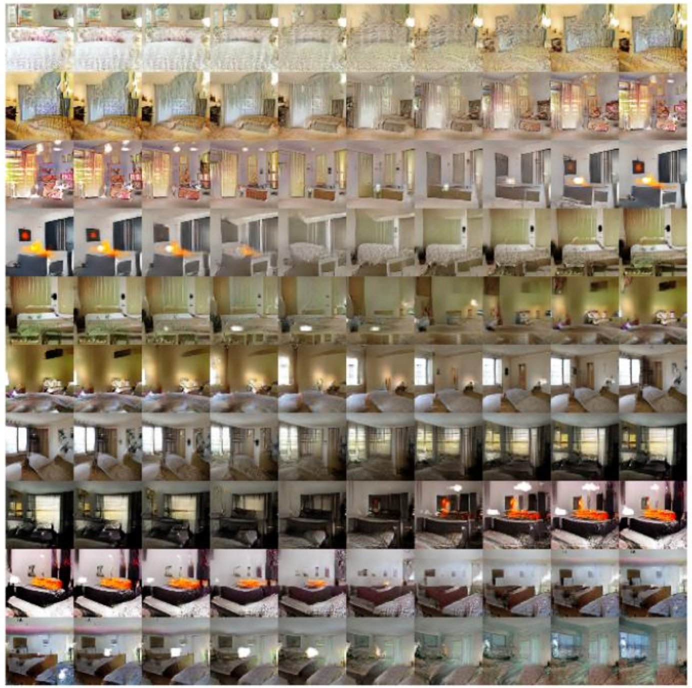


Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, Yoshua Bengio. Generative Adversarial Nets. Advances in Neural Information Processing Systems (NeurIPS) 2014.

Generative Adversarial Networks: DC-GAN

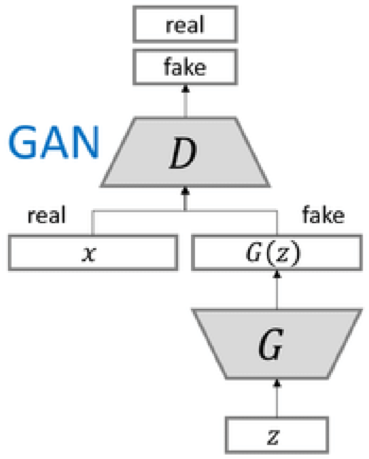


Generative Adversarial Networks: Interpolation

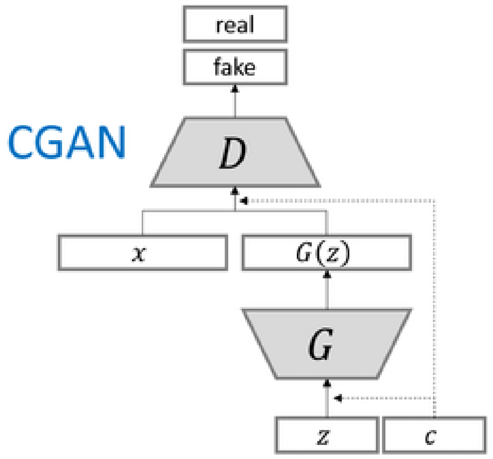


Alec Radford, Luke Metz, Soumith Chintala. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. ICLR 2016

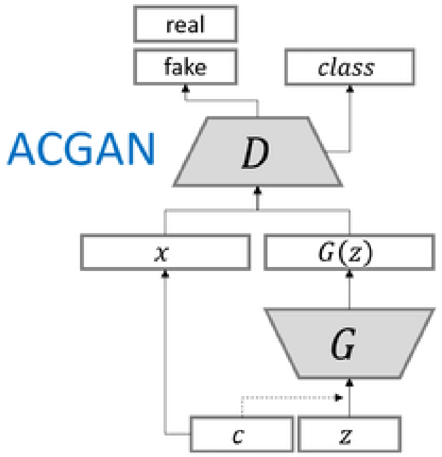
Conditional GANs



(a)



(b)

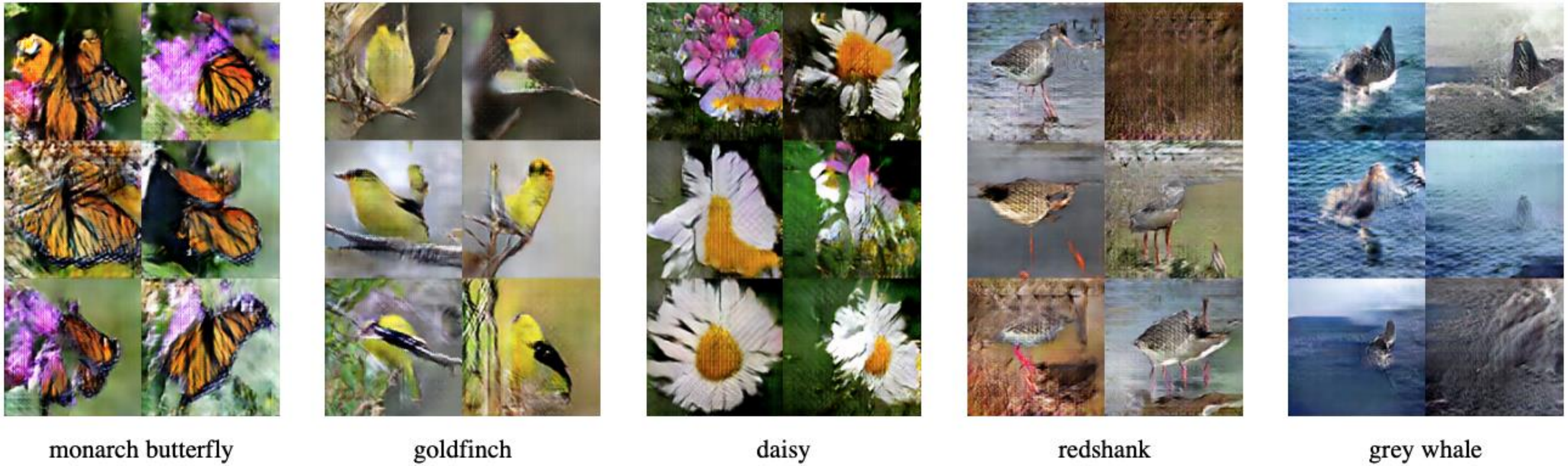


(c)

[b] Mehdi Mirza, Simon Osindero. Conditional Generative Adversarial Nets. 2014

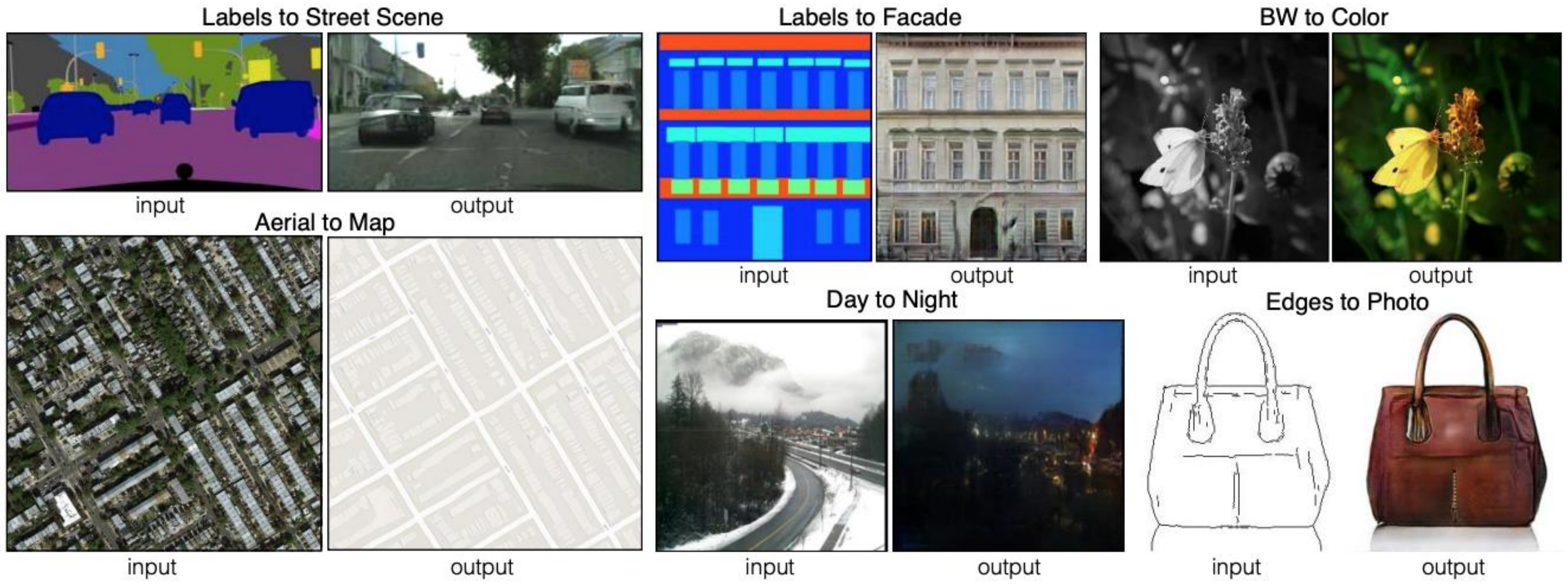
[c] Augustus Odena, Christopher Olah, Jonathon Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. ICML 2016

Conditional GANs



Augustus Odena, Christopher Olah, Jonathon Shlens. Conditional Image Synthesis With Auxiliary Classifier GANs. ICML 2016

Image-to-Image Translation: Pix2Pix



Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, Alexei A. Efros. Image-to-Image Translation with Conditional Adversarial Networks. CVPR 2017 https://arxiv.org/abs/1611.07004

Image-to-Image Translation: Pix2Pix

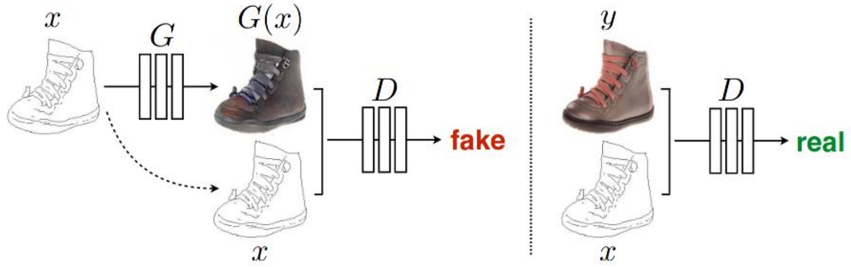
Objective:

$$G^* = \arg\min_G \max_D \mathcal{L}_{cGAN}(G, D) + \lambda \mathcal{L}_{L1}(G)$$

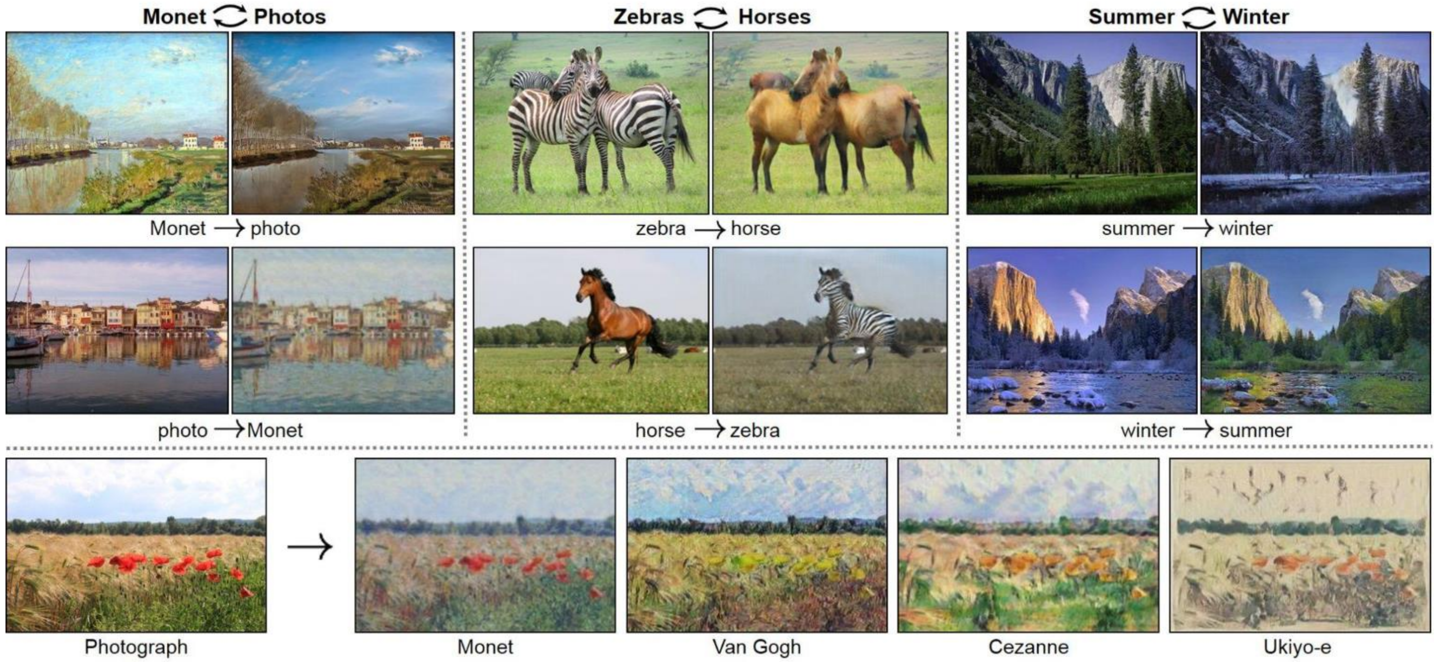
where

$$\begin{aligned}\mathcal{L}_{cGAN}(G, D) =& \mathbb{E}_{x,y}[\log D(x, y)] + \\ & \mathbb{E}_{x,z}[\log(1 - D(x, G(x, z))]\end{aligned}$$

$$\mathcal{L}_{L1}(G) = \mathbb{E}_{x,y,z}[\|y - G(x, z)\|_1]$$

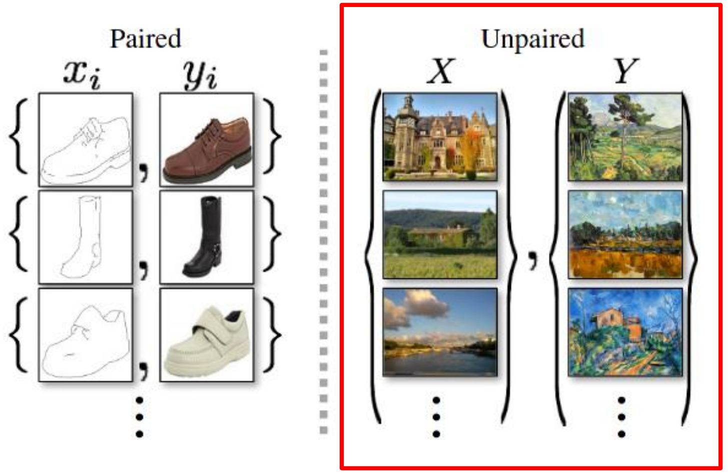


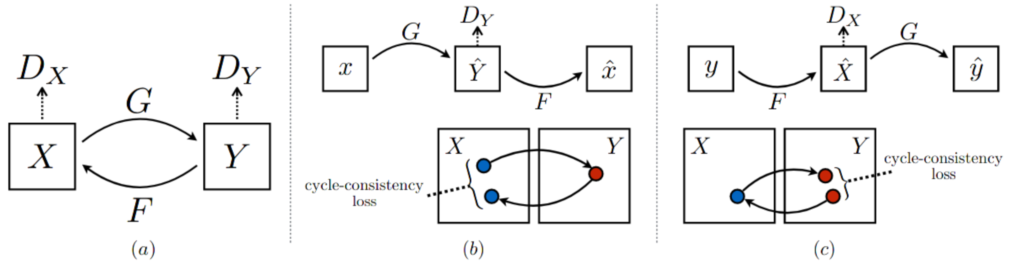
Unpaired Image-to-Image Translation: CycleGAN



Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Unpaired Image-to-Image Translation: CycleGAN





Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Unpaired Image-to-Image Translation: CycleGAN

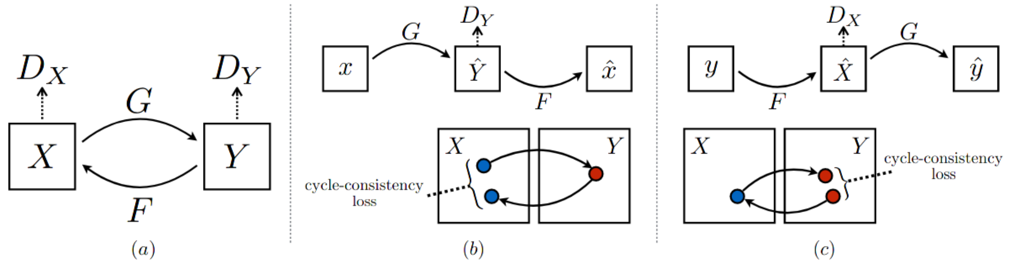
Objective:

$$\begin{aligned}\mathcal{L}(G, F, D_X, D_Y) =& \mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) \\ &+ \mathcal{L}_{\text{GAN}}(F, D_X, Y, X) \\ &+ \lambda\mathcal{L}_{\text{cyc}}(G, F),\end{aligned}$$

where

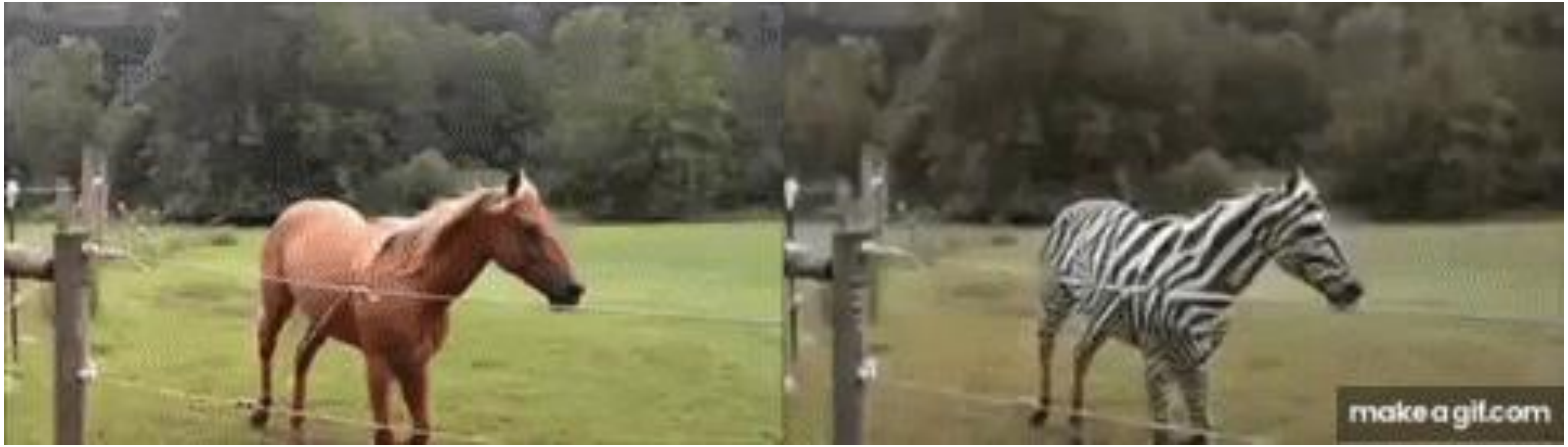
$$\begin{aligned}\mathcal{L}_{\text{GAN}}(G, D_Y, X, Y) =& \mathbb{E}_{y\sim p_{\text{data}}(y)}[\log D_Y(y)] \\ &+ \mathbb{E}_{x\sim p_{\text{data}}(x)}[\log(1 - D_Y(G(x))],\end{aligned}$$

$$\begin{aligned}\mathcal{L}_{\text{cyc}}(G, F) =& \mathbb{E}_{x\sim p_{\text{data}}(x)}[\|F(G(x)) - x\|_1] \\ &+ \mathbb{E}_{y\sim p_{\text{data}}(y)}[\|G(F(y)) - y\|_1].\end{aligned}$$



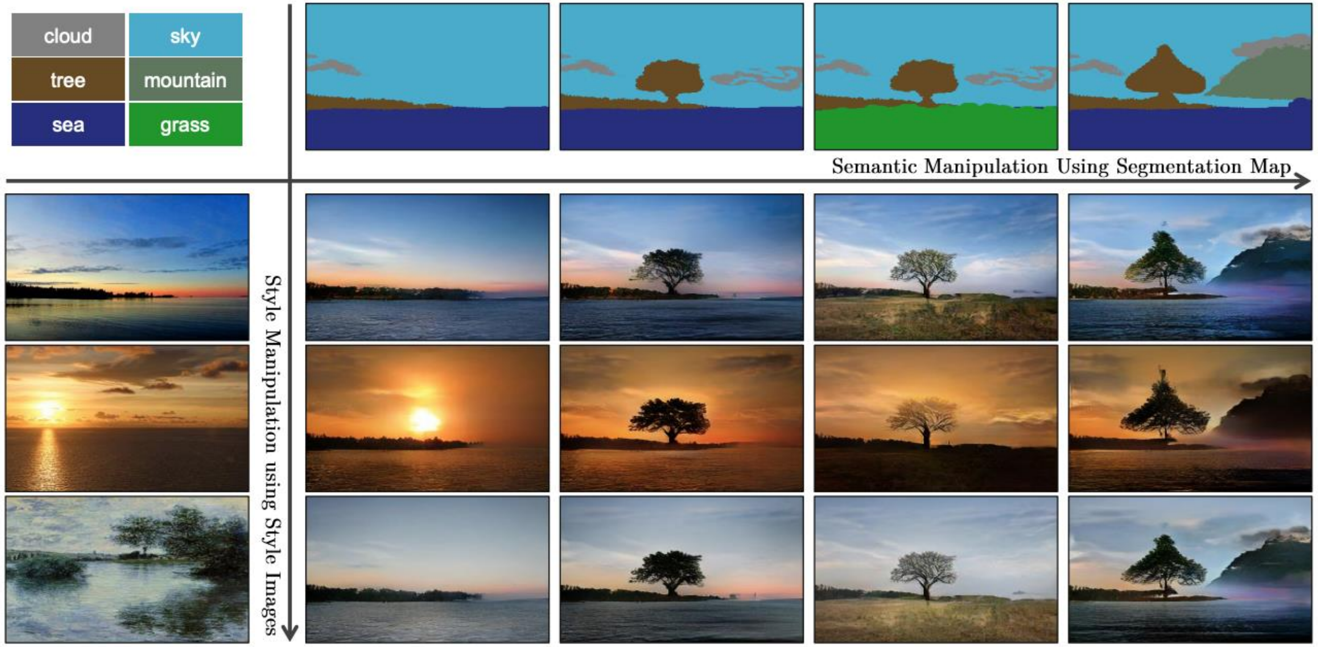
Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Unpaired Image-to-Image Translation: CycleGAN



Jun-Yan Zhu, Taesung Park, Phillip Isola, Alexei A. Efros. Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks. ICCV 2017
https://arxiv.org/abs/1703.10593

Label Map to Image



cloud
sky
tree
mountain
sea
grass
Semantic Manipulation Using Segmentation Map
Style Manipulation using Style Images

Taesung Park, Ming-Yu Liu, Ting-Chun Wang, Jun-Yan Zhu. Semantic Image Synthesis with Spatially-Adaptive Normalization. CVPR 2019 https://arxiv.org/abs/1903.07291

Label Map to Image

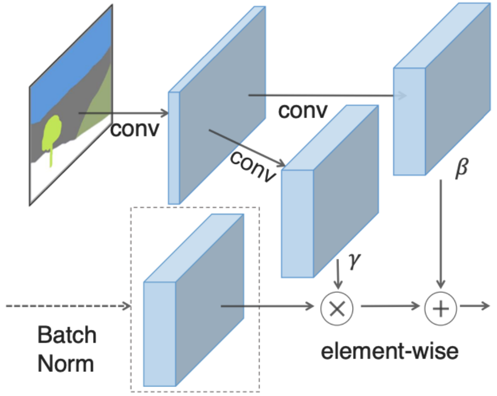


Figure 2: In the SPADE, the mask is first projected onto an embedding space and then convolved to produce the modulation parameters γ and β. Unlike prior conditional normalization methods, γ and β are not vectors, but tensors with spatial dimensions. The produced γ and β are multiplied and added to the normalized activation element-wise.

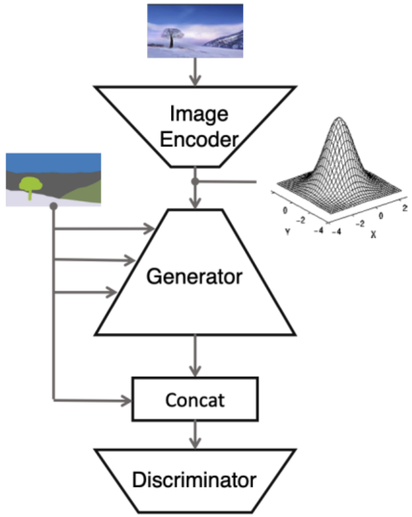
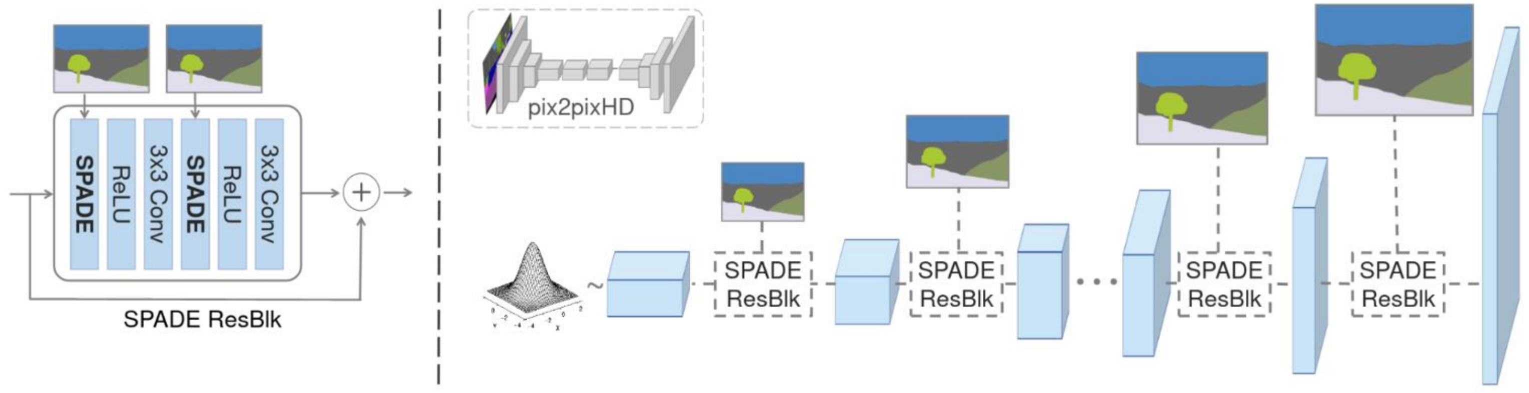
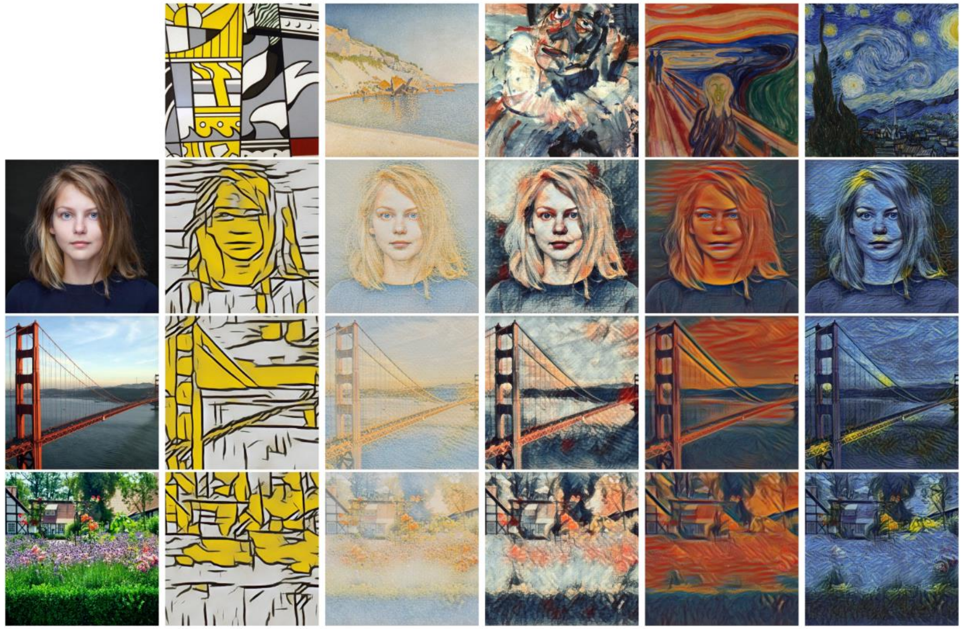


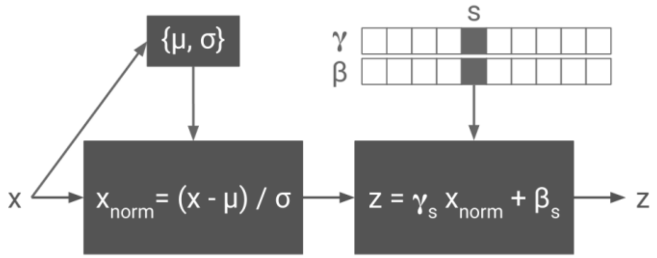
Figure 15: The image encoder encodes a real image to a latent representation for generating a mean vector and a variance vector. They are used to compute the noise input to the generator via the reparameterization trick [28]. The generator also takes the segmentation mask of the input image as input via the proposed SPADE ResBlks. The discriminator takes concatenation of the segmentation mask and the output image from the generator as input and aims to classify that as fake.

Label Map to Image



Conditional GANs: Conditional Instance Normalization



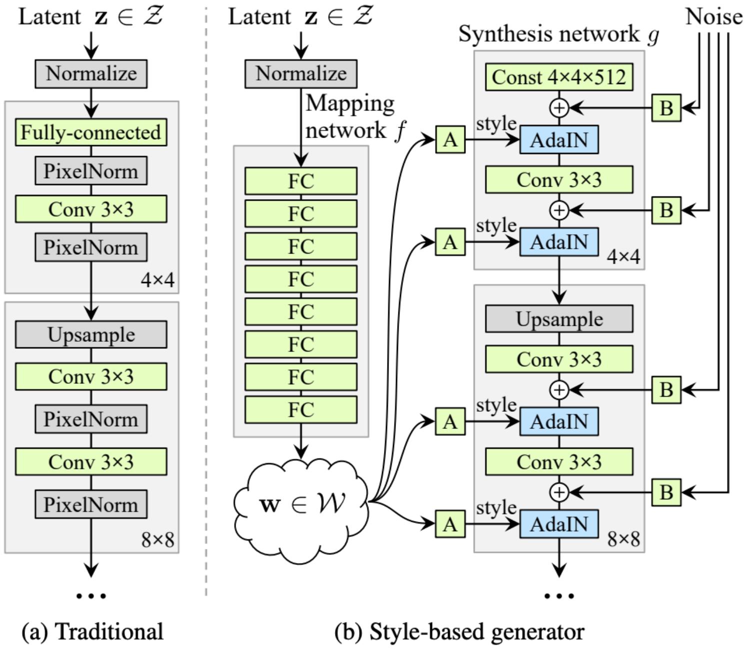


s
γ
β
{μ, σ}
x
$x_{norm} = (x - \mu) / \sigma$
$z = \gamma_s x_{norm} + \beta_s$
z

The input activation x is normalized across spatial dimensions and scaled and shifted using style-dependent parameter vectors γ_s, β_s where s indexes the style label.

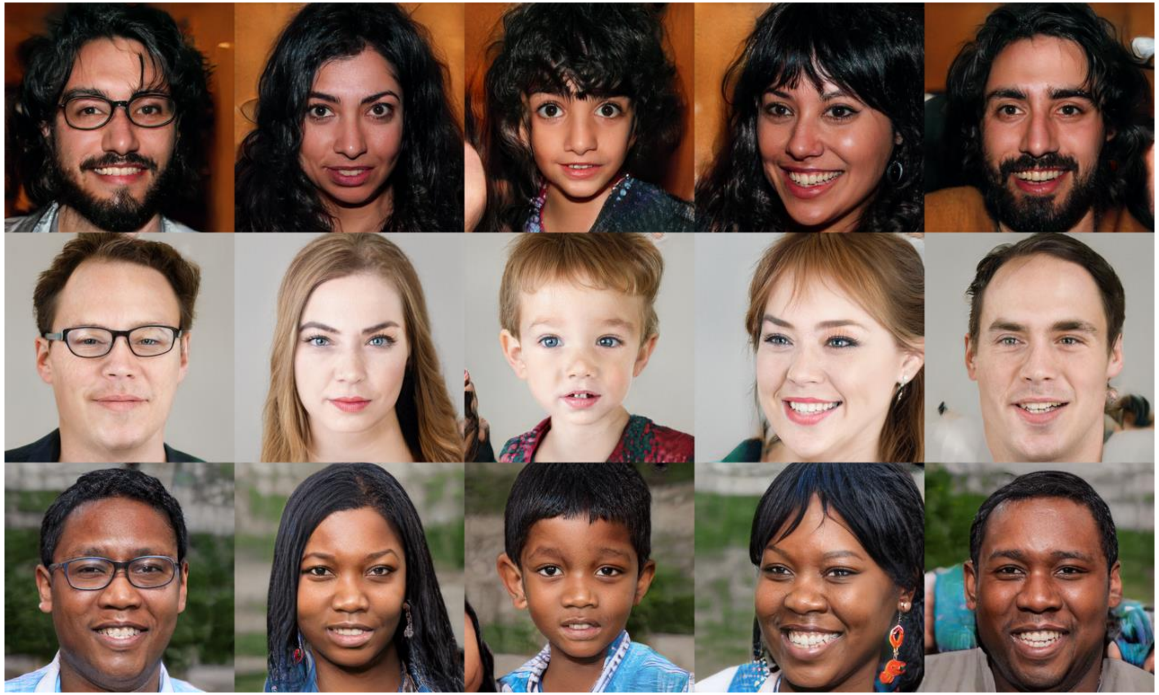
Vincent Dumoulin, Jonathon Shlens, Manjunath Kudlur, A Learned Representation For Artistic Style. ICLR 2017 https://arxiv.org/abs/1610.07629

StyleGAN



Tero Karras, Samuli Laine, Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. CVPR 2019

StyleGAN

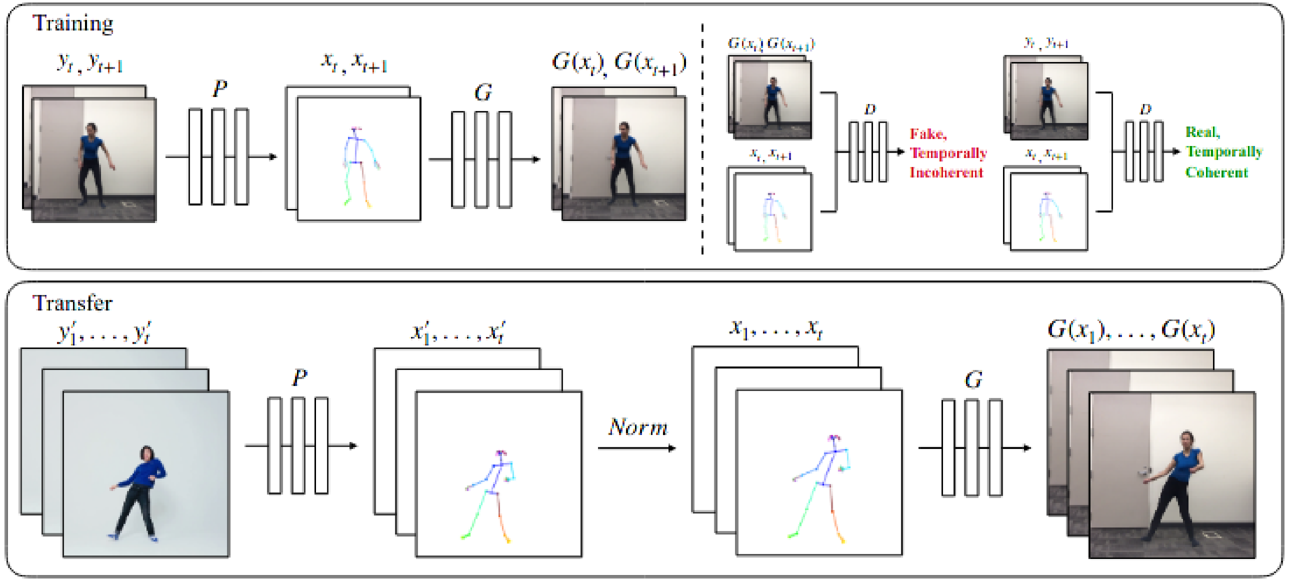


Tero Karras, Samuli Laine, Timo Aila. A Style-Based Generator Architecture for Generative Adversarial Networks. CVPR 2019

Video Generation. Everybody Dance Now

Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros. Everybody Dance Now. ICCV 2019

Video Generation. Everybody Dance Now



Caroline Chan, Shiry Ginosar, Tinghui Zhou, Alexei A. Efros. Everybody Dance Now. ICCV 2019

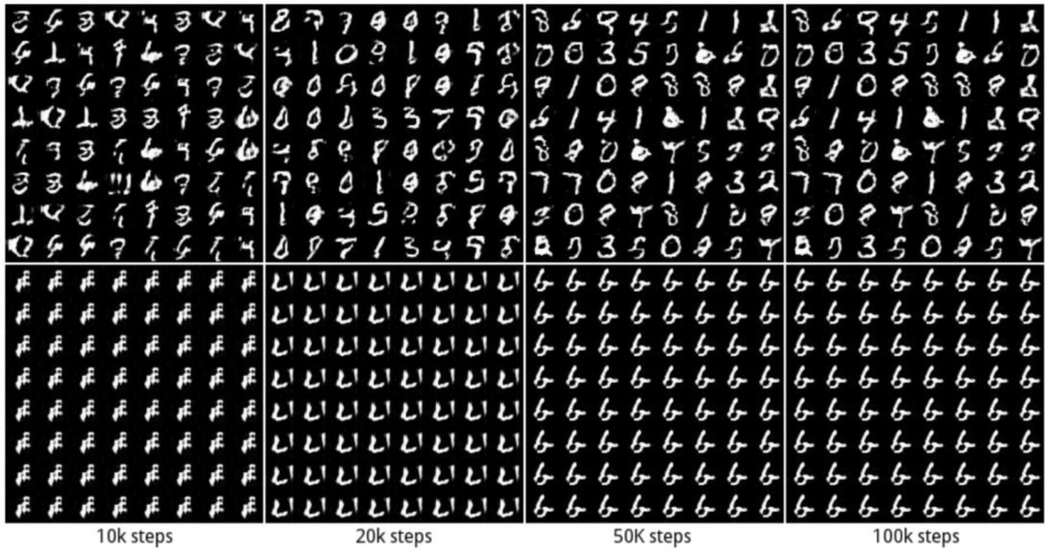
Problems of GANs

- Mode collapse:
  - G collapses providing limited sample variety
- Non-convergence:
  - model parameters oscillate, destabilize and never converge
- Diminished gradient:
  - D is so successful that the G gradient vanishes and learns nothing

Mode collapse

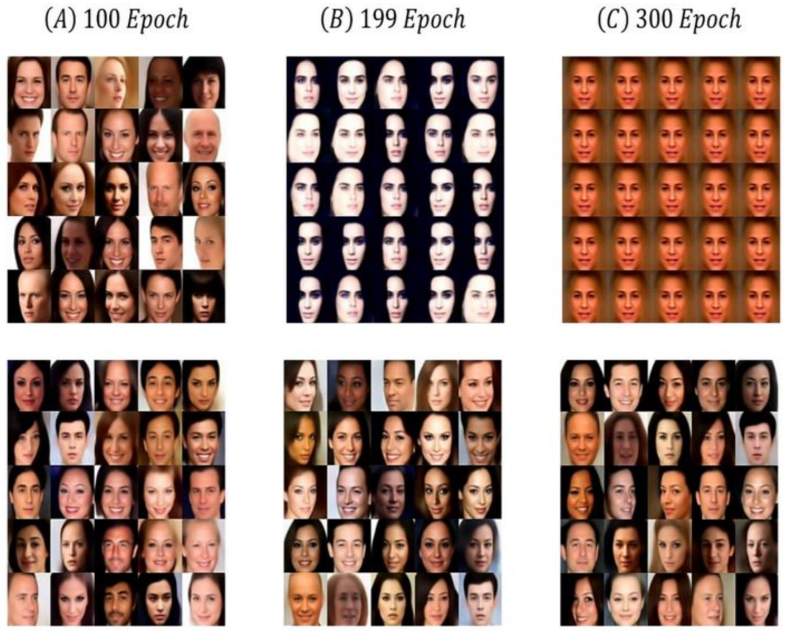
Real-life data is multimodal (10 in MNIST)

Mode collapse: when few modes generated



Partial mode collapse

The generator produces realistic and diverse samples, but much less diverse than the real-world data distribution.



Solution to mode collapse

Wasserstein loss

- Trains the discriminator to optimality without worrying about vanishing gradients.
- If the discriminator doesn't get stuck in local minima, it learns to reject the outputs that the generator stabilizes on.

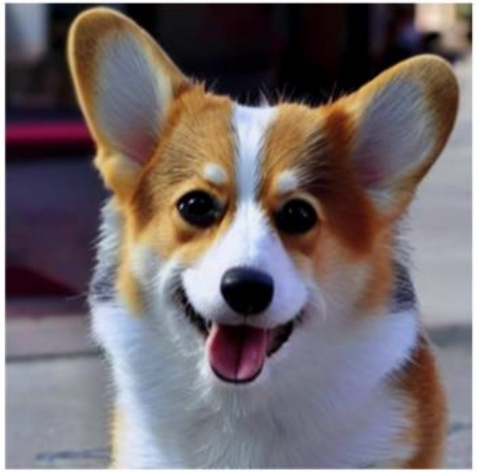
Martin Arjovsky, Soumith Chintala, Léon Bottou. Wasserstein GAN. 2017 https://arxiv.org/abs/1701.07875

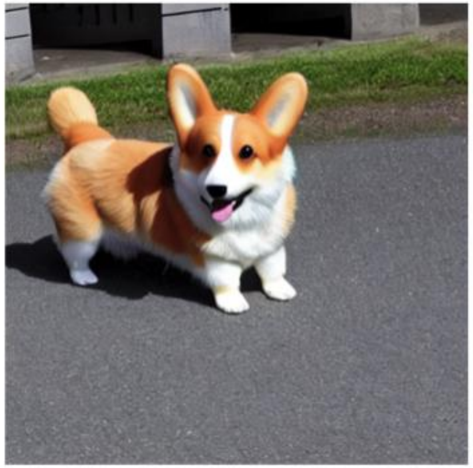
Generative Models Evaluation

Which of these images looks better?

Generative Models Evaluation

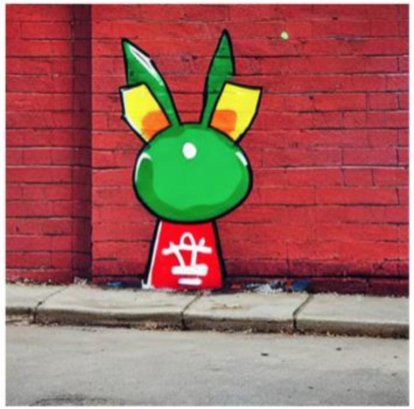
Which of these images looks more realistic?

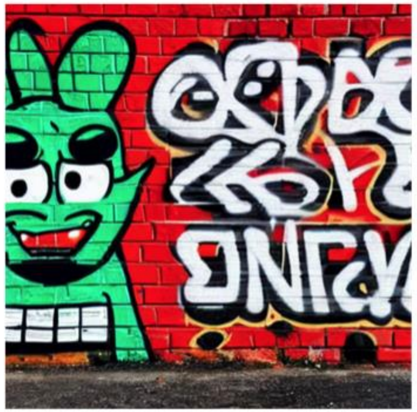




Generative Models Evaluation

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 (IS) [1]
- Frechet Inception Distance (FID) [2]
- Structured Similarity Index Metric (SSIM) [3]
- CLIP score [4]

[1] Tim Salimans, Ian Goodfellow, Wojciech Zaremba, Vicki Cheung, Alec Radford, Xi Chen. Improved Techniques for Training GANs. NeurIPS 2016 https://arxiv.org/abs/1606.03498
[2] 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
[3] Zhou Wang; A.C. Bovik; H.R. Sheikh; E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 2004 https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf
[4] Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, Yejin Choi.CLIPScore: A Reference-free Evaluation Metric for Image Captioning. EMNLP 2021https://arxiv.org/abs/2104.08718

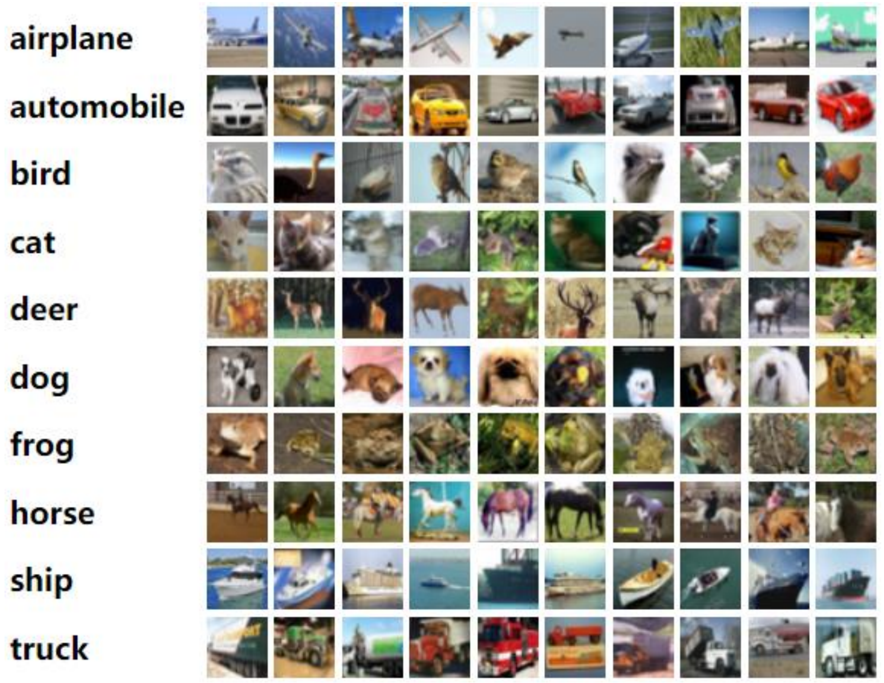
Inception Score (IS)

IS measures:

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

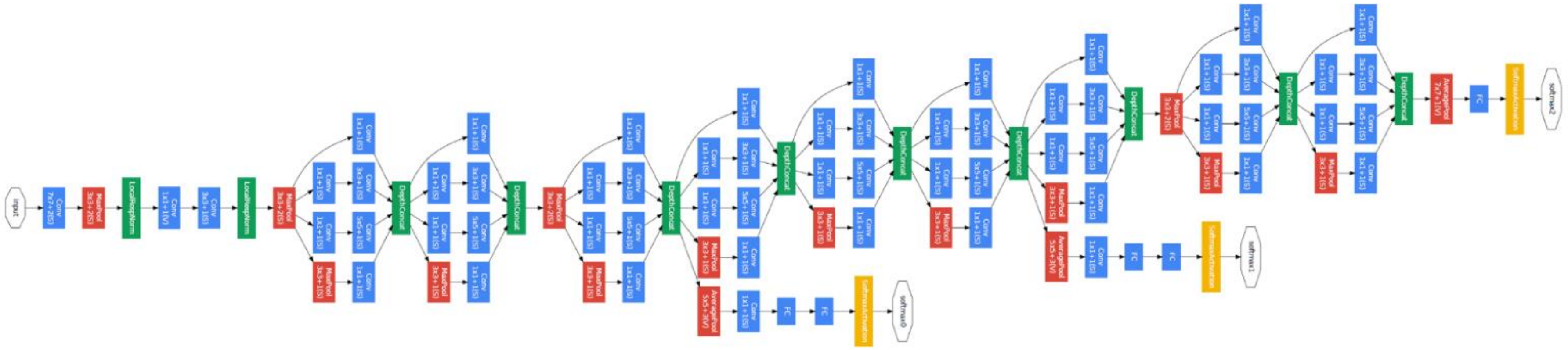
Inception Score (IS)

CIFAR10 dataset:



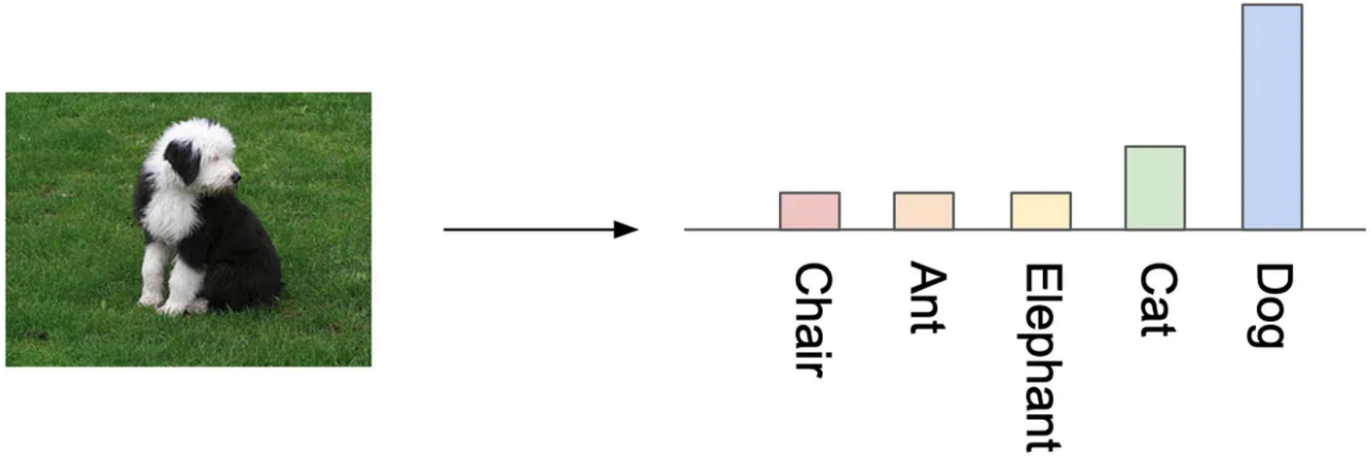
Inception Score (IS)

Generate 50000 images by the **Inception** image classifier pre-trained on CIFAR10



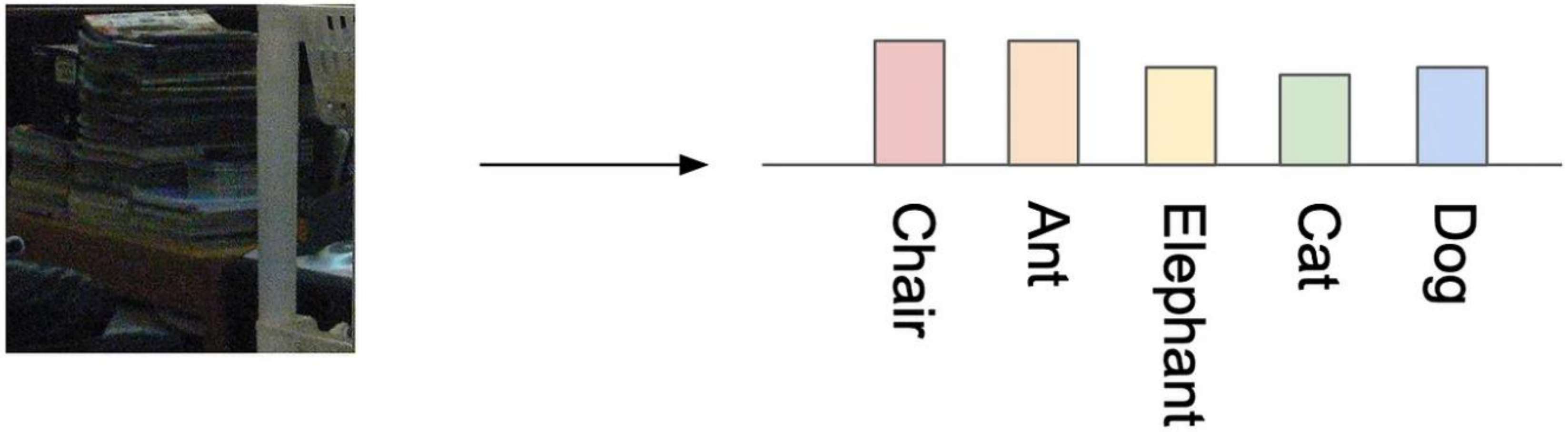
Inception Score (IS)

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



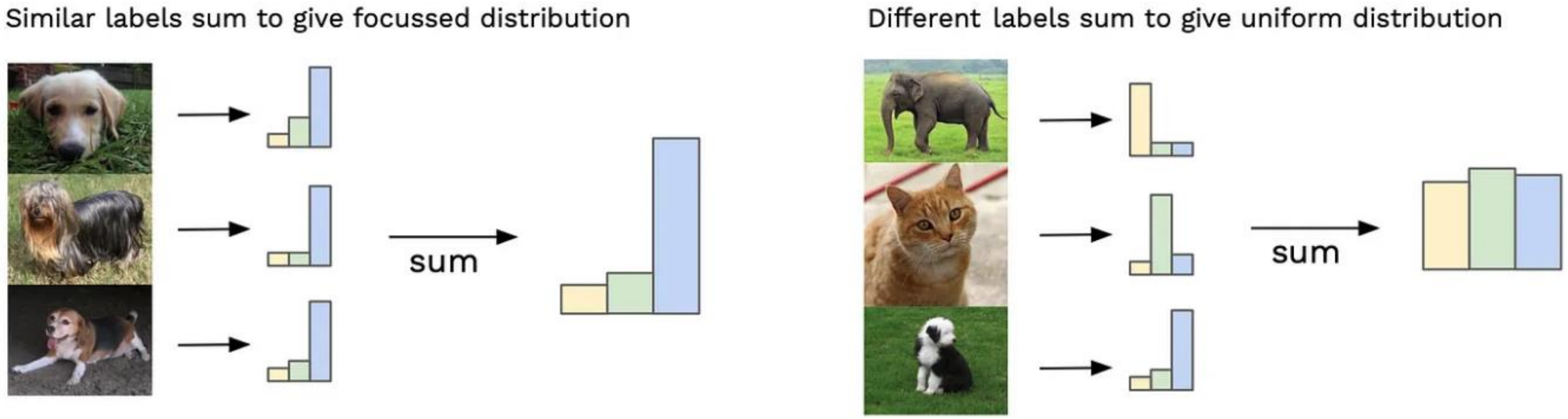
Inception Score (IS)

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



Inception Score (IS)

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


Similar labels sum to give focussed distribution
sum
Different labels sum to give uniform distribution
sum

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:

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

Inception Score (IS) limitations

IS is limited by what the Inception classifier can detect, which is linked to the training data

Frechet Inception Distance (FID)

FID compares the distribution of generated images with the distribution of real images

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)

- Uses 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)

- Uses the Inception network to extract features from an intermediate layer
- Models 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)

- Uses the Inception network to extract features from an intermediate layer
- Models 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 is:

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

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
- 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

Structured Similarity Index Metric (SSIM)

The Structural Similarity Index (SSIM) metric mimicks the human visual perception system which is highly capable of identifying **structural information** from a scene.

SSIM extracts 3 features from an image on a **pixel-wise** basis:

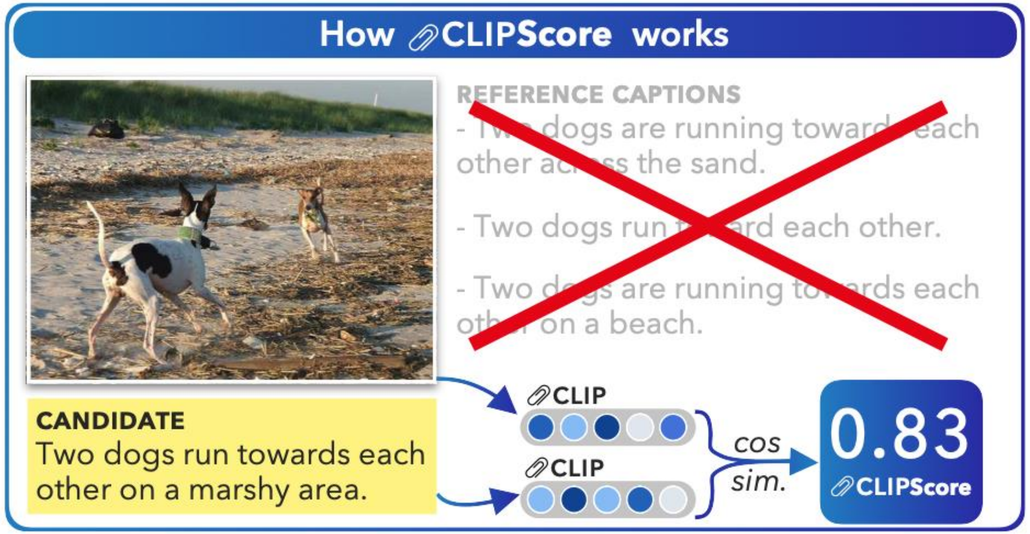
- Luminance
- Contrast
- Structure

The comparison between the two images is performed on the basis of these 3 features.

Zhou Wang; A.C. Bovik; H.R. Sheikh; E.P. Simoncelli. Image quality assessment: from error visibility to structural similarity. IEEE Transactions on Image Processing 2004 https://www.cns.nyu.edu/pub/eero/wang03-reprint.pdf

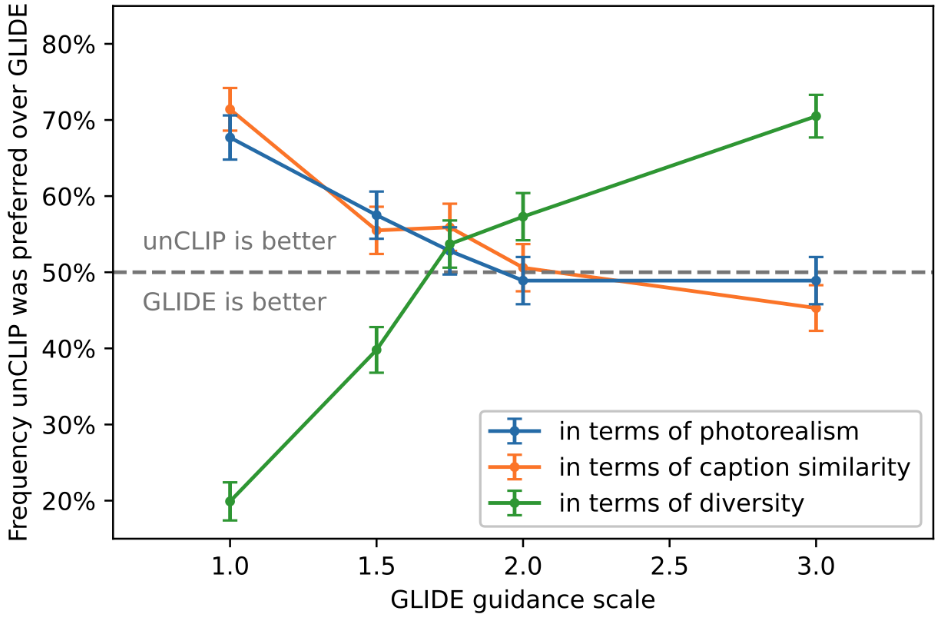
CLIP score

- CLIP score measures the compatibility of image-caption pairs.
- Higher CLIP scores imply higher compatibility


How CLIPScore works
REFERENCE CAPTIONS
- Two dogs are running towards each other across the sand.
- Two dogs run toward each other.
- Two dogs are running towards each other on a beach.
CANDIDATE
Two dogs run towards each other on a marshy area.
CLIP
CLIP
cos sim.
0.83
CLIPScore

Jack Hessel, Ari Holtzman, Maxwell Forbes, Ronan Le Bras, Yejin Choi.CLIPScore: A Reference-free Evaluation Metric for Image Captioning. EMNLP 2021
https://arxiv.org/abs/2104.08718

DALL-E 2 evaluation results



Increasing guidance scale improves sample quality at the cost of diversity

Aditya Ramesh, Prafulla Dhariwal, Alex Nichol, Casey Chu, Mark Chen.

Hierarchical Text-Conditional Image Generation with CLIP Latents. 2022 https://arxiv.org/abs/2204.06125

https://openai.com/product/dall-e-2

TP9

https://colab.research.google.com/drive/1PYx6qESrgTmm38eCYHpkN-1UovsK4UD7?usp=sharing

https://colab.research.google.com/drive/1NWoFAZjLeaCMb3JsMUPJX5bm0PenCRYy?usp=sharing

https://colab.research.google.com/drive/1xOfQOR-XCP1uw0pOqR4nXlTZU6wsXGp2?usp=sharing

Q&A