Introduction to Machine Learning CS182

Lu Sun

School of Information Science and Technology

ShanghaiTech University

December 28, 2023

Today:

- Deep Generative Networks (DGN)

Readings:

- Deep Learning (DL), Chapters 14&20

Today's Agenda

Deep Generative Models (DGM)

- Overview
- Representation Learning with Autoencoder
- Generative Adversarial Network (GAN)
- Applications of GANs

Acknowledgement: Hugo Larochelle's, MehryarMohri@NYU's,YingyuLiang@Princeton's, BhikshaRaj@CMU's&FeifeiLi@Stanford'scourse notes

Deep Generative Networks-DGM

- **Overview**
- Representation Learning with Autoencoder
- Generative Adversarial Network (GAN)
- Applications of GANs

Unsupervised Learning

- Task formulation

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

Unsupervised Learning

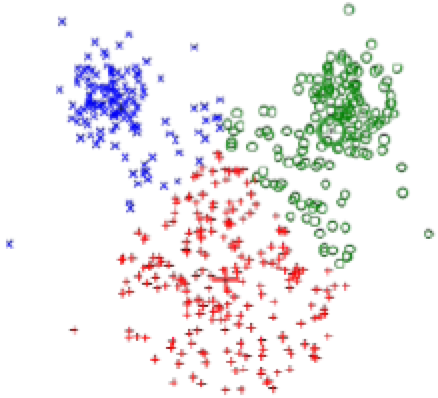
- Task formulation

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



K-means clustering

Unsupervised Learning

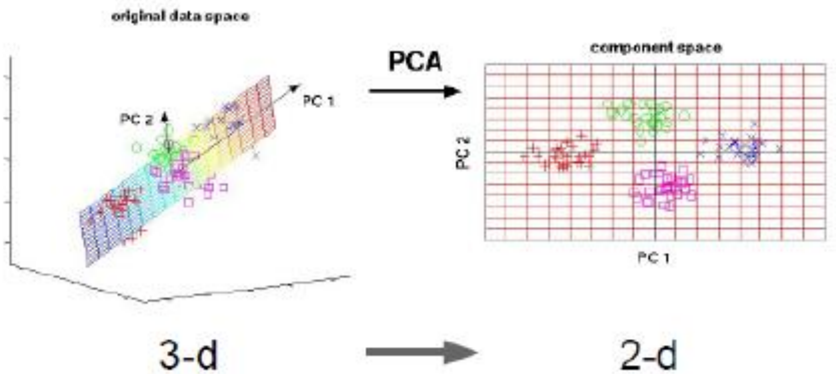
- Task formulation

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



Principal Component Analysis
(Dimensionality reduction)

Unsupervised Learning

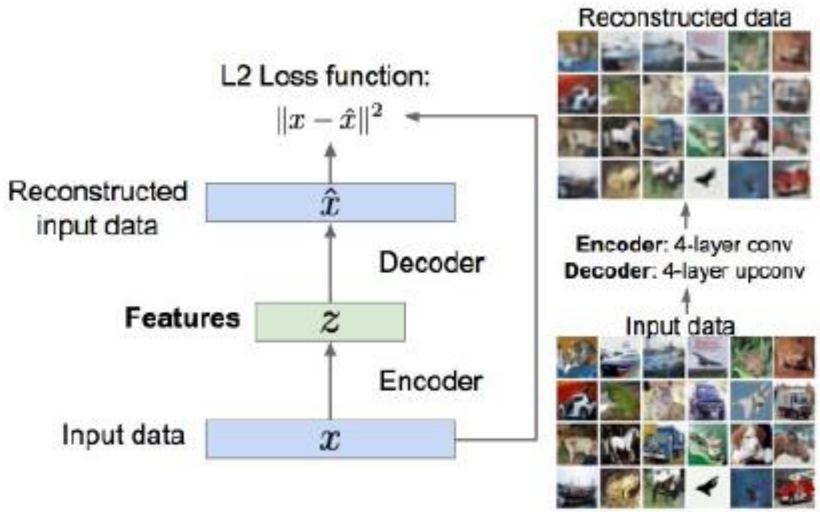
- Task formulation

Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.



Autoencoders
(Feature learning)

Unsupervised Learning

- Task formulation

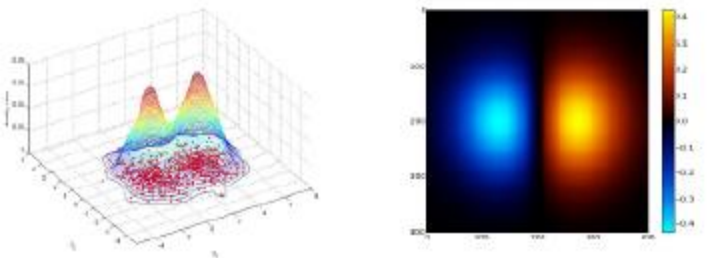
Unsupervised Learning

Data: x
Just data, no labels!

Goal: Learn some underlying hidden *structure* of the data

Examples: Clustering, dimensionality reduction, feature learning, density estimation, etc.

1-d density estimation



2-d density estimation

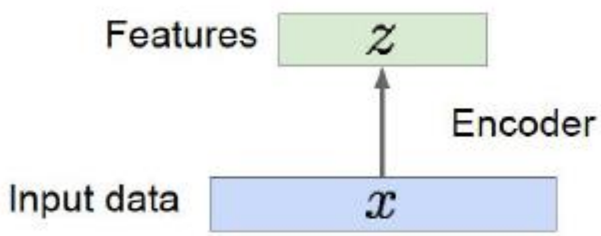
Deep Generative Networks-DGM

- Overview
- **Representation Learning with Autoencoder**
- Generative Adversarial Network (GAN)
- Applications of GANs

Autoencoder

- Feature representation learning

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

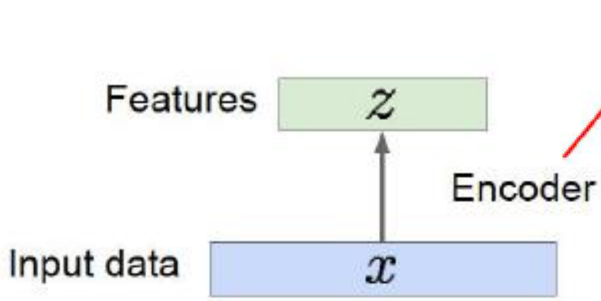


Autoencoder

- Feature representation learning

Unsupervised approach for learning a lower-dimensional feature representation from unlabeled training data

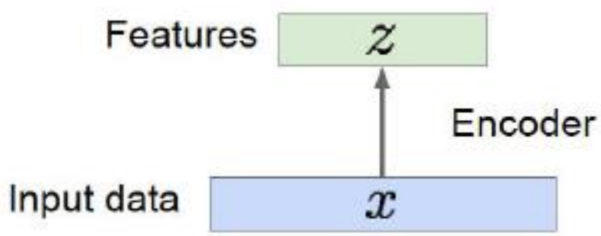
Originally: Linear + nonlinearity (sigmoid)
Later: Deep, fully-connected
Later: ReLU CNN



Autoencoder

- Feature representation learning

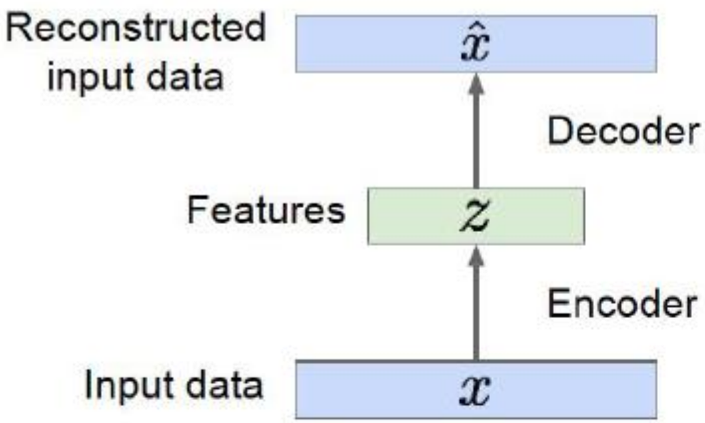
How to learn this feature representation?



Autoencoder

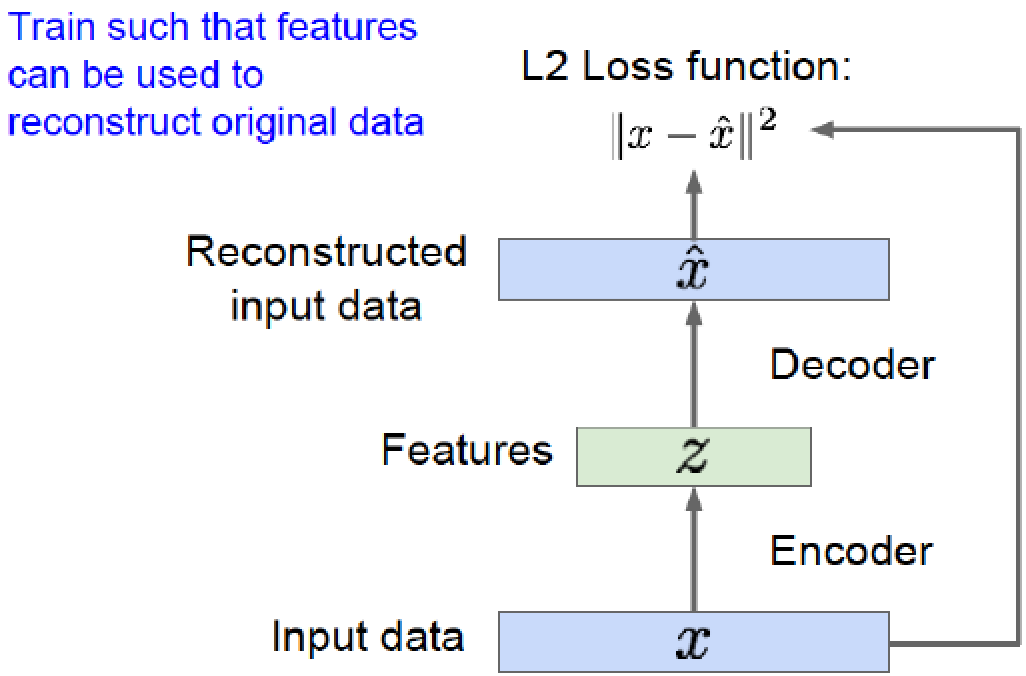
- Feature representation learning

How to learn this feature representation?
Train such that features can be used to reconstruct original data
"Autoencoding" - encoding itself



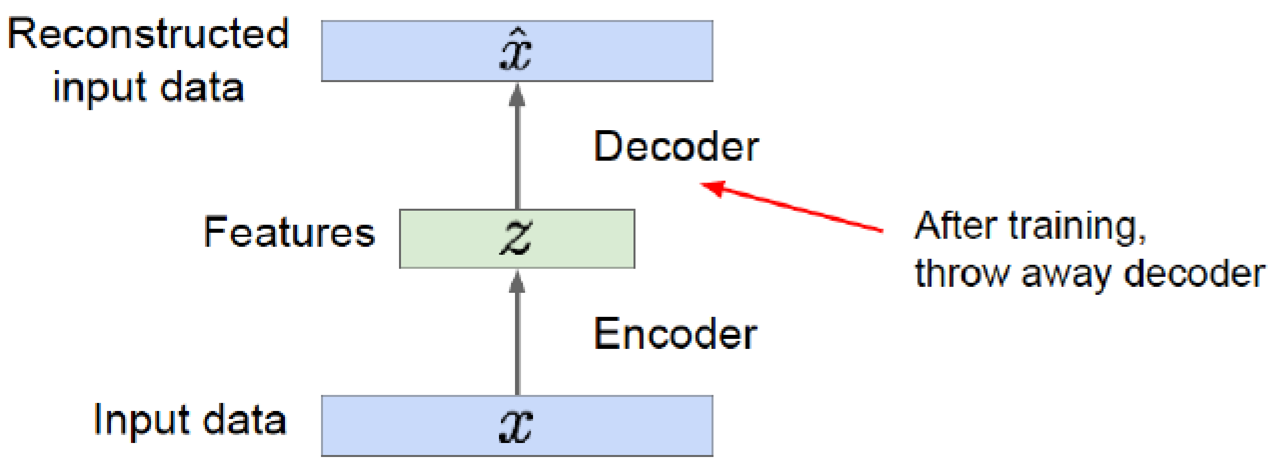
Autoencoder

- Feature representation learning



Autoencoder

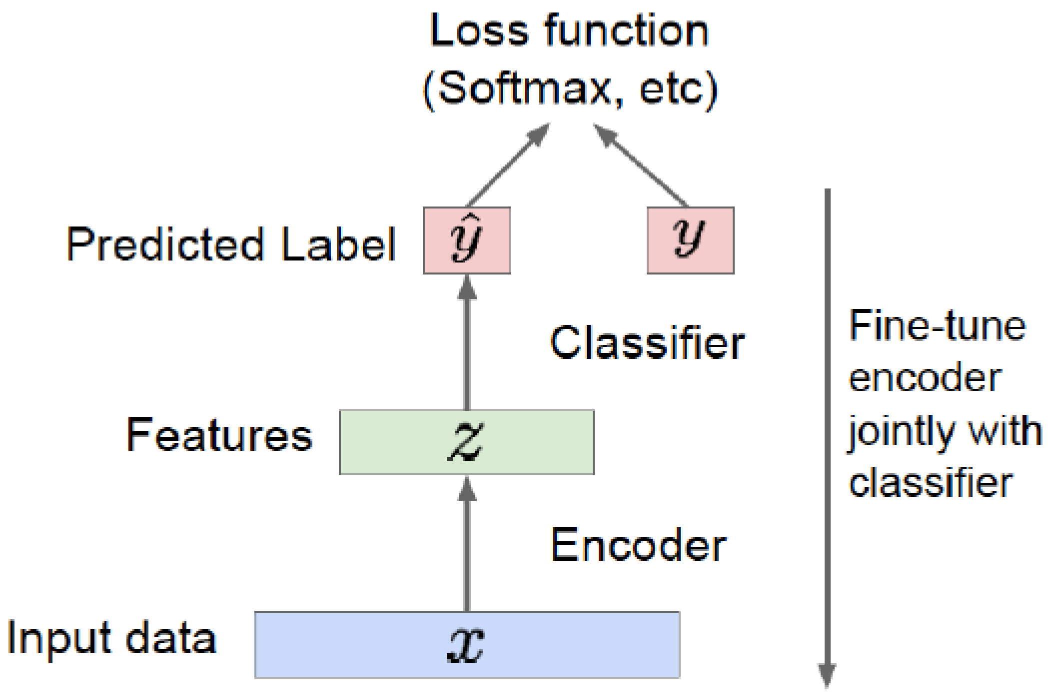
- Feature representation learning



Autoencoder

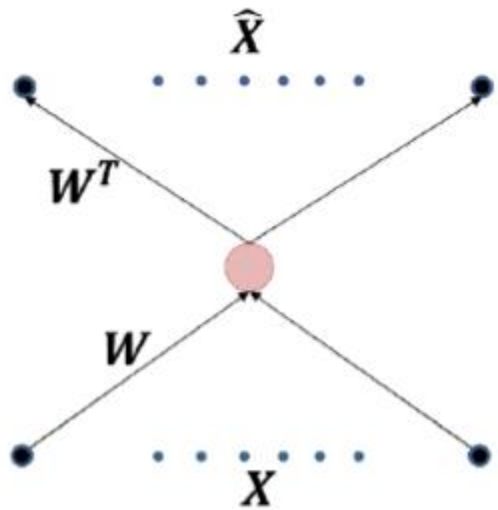
- Feature representation learning

Encoder can be used to initialize a **supervised** model



Autoencoder

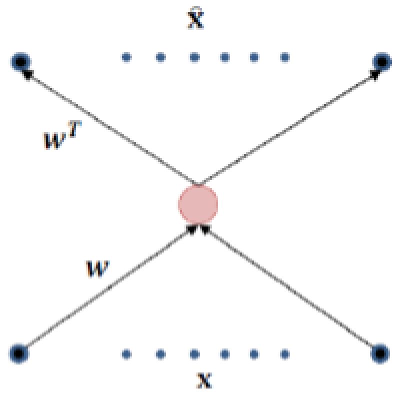
- Linear hidden layer example



- A single hidden unit
- Hidden unit has *linear* activation
- What will this learn?

Autoencoder

- Linear hidden layer example



Training: Learning W by minimizing L2 divergence

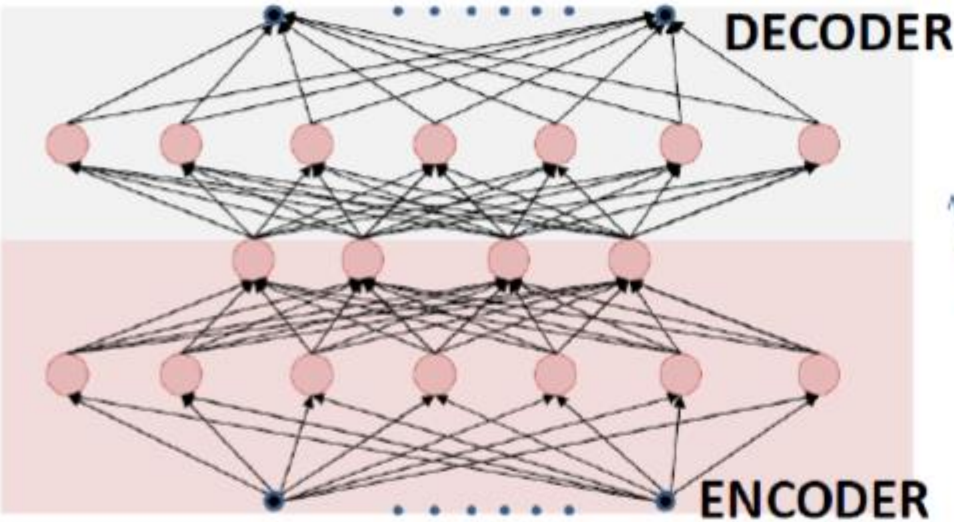
$$\hat{\mathrm{x}} = w^T w\mathrm{x}$$
$$div(\hat{\mathrm{x}}, \mathrm{x}) = \|\mathrm{x} - \hat{\mathrm{x}}\|^2 = \|\mathrm{x} - \mathrm{w}^T w\mathrm{x}\|^2$$

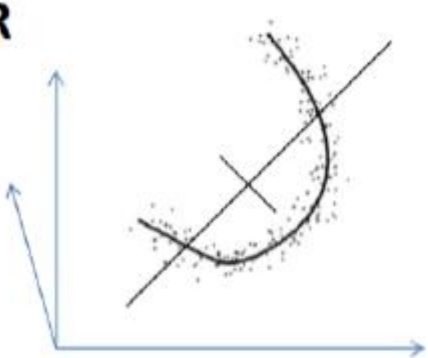
$$\widehat{W} = \underset{W}{\operatorname{argmin}}\, E[div(\hat{\mathrm{x}}, \mathrm{x})]$$
$$\widehat{W} = \underset{W}{\operatorname{argmin}}\, E[\|\mathrm{x} - \mathrm{w}^T w\mathrm{x}\|^2]$$

- This is just PCA!

Autoencoder

- Nonlinear hidden layer



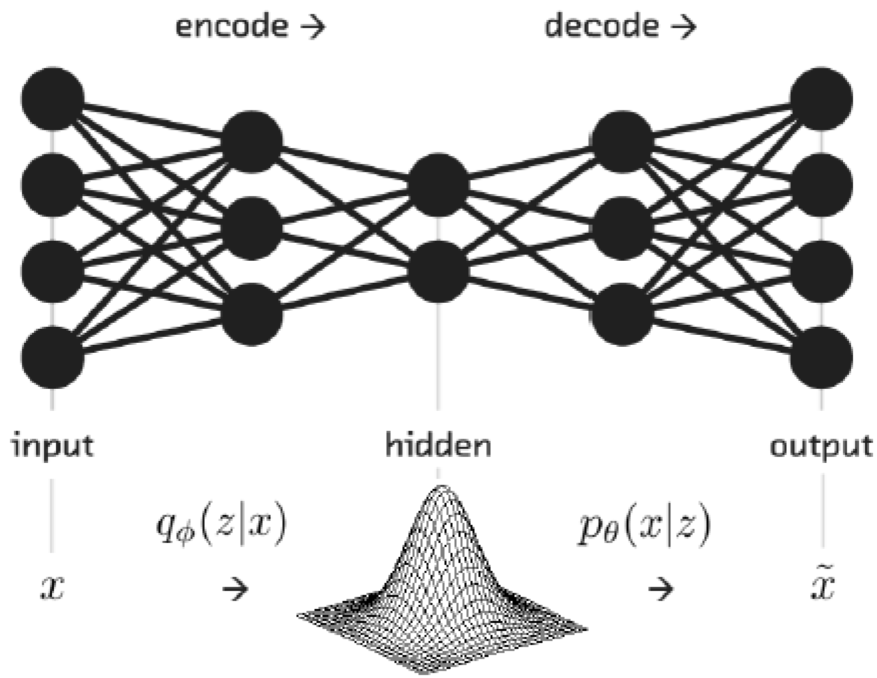


- With non-linearity
  - "Non linear" PCA
  - Deeper networks can capture more complicated manifolds
    - "Deep" autoencoders

Variational Autoencoder (VAE)

- Objective $\mathcal{L}(x,\phi,\theta) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)]$

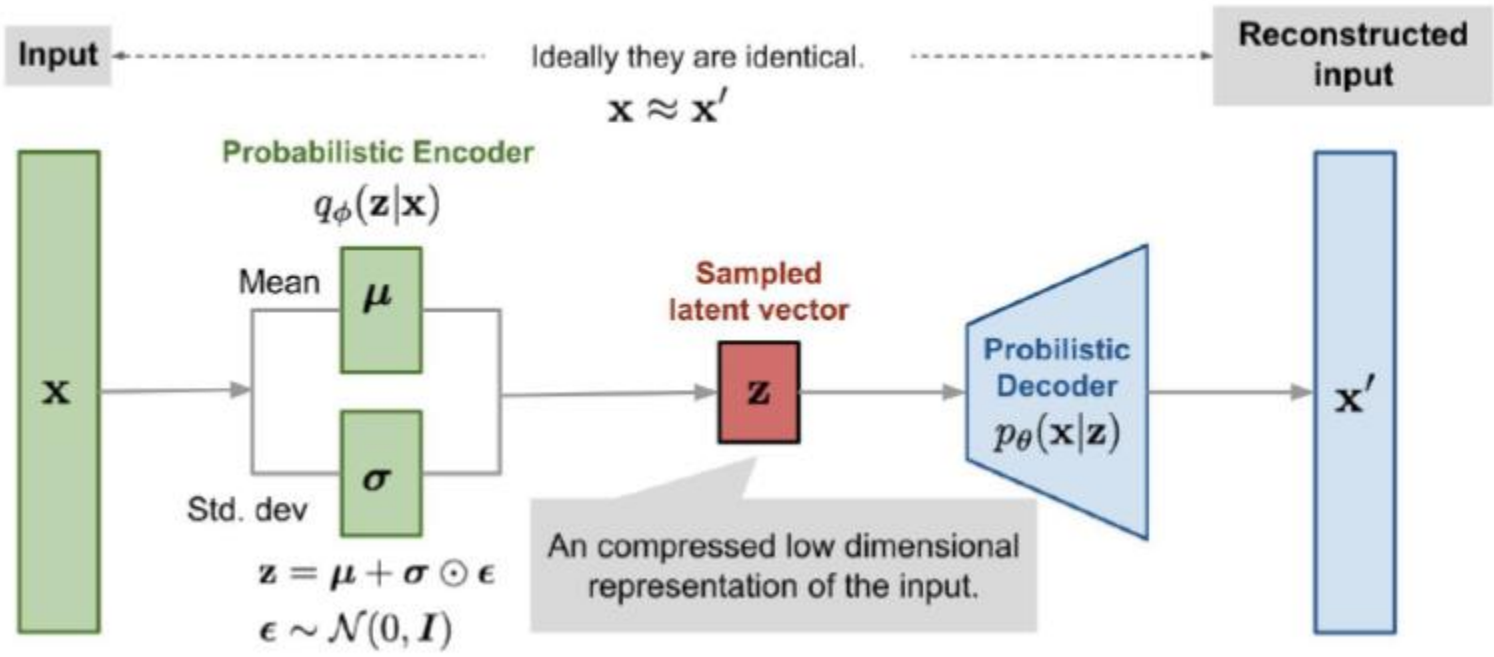
Regularization term Reconstruction term



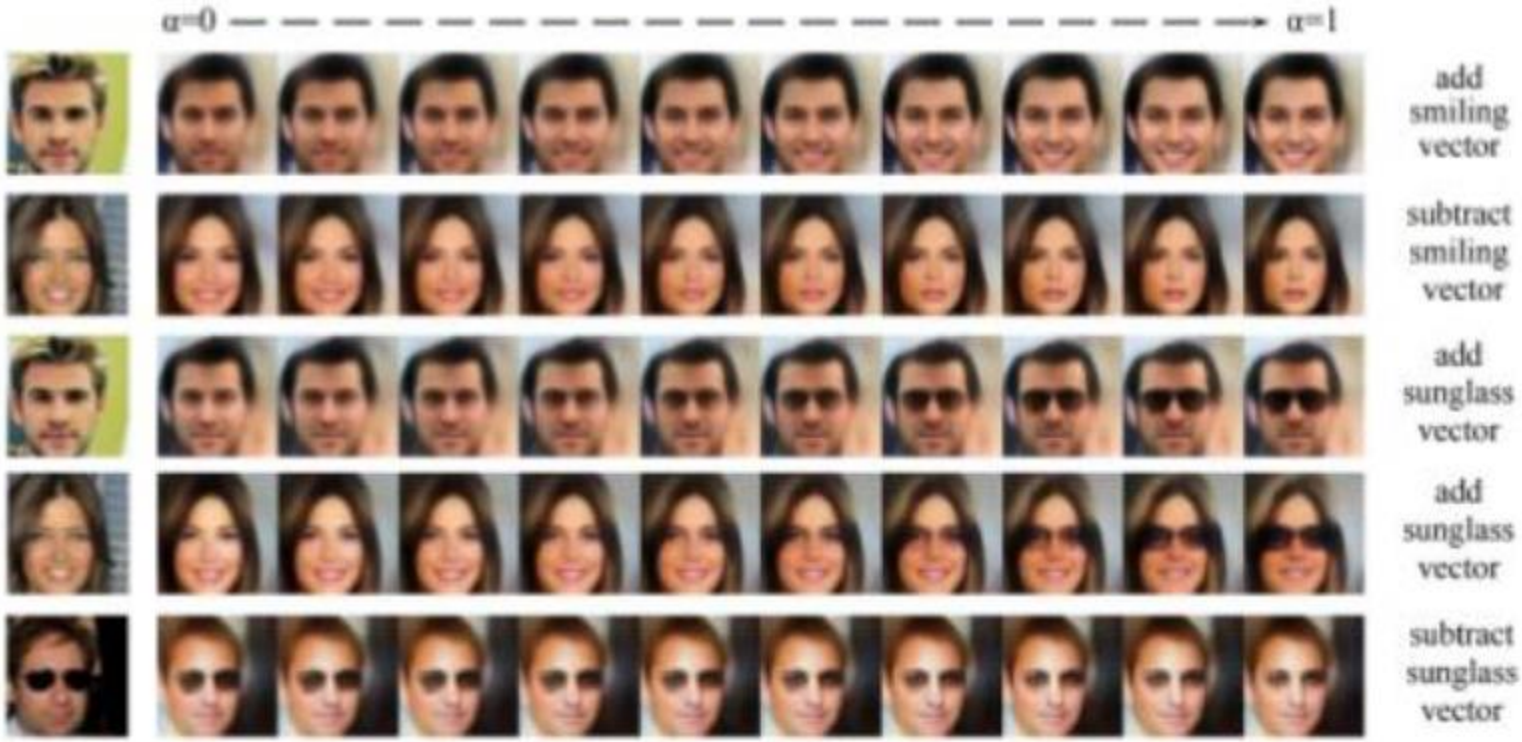
Variational Autoencoder (VAE)

- Objective $\mathcal{L}(x, \phi, \theta) = -D_{KL}(q_\phi(z|x)||p_\theta(z)) + E_{q_\phi(z|x)}[\log p_\theta(x|z)]$

Regularization term Reconstruction term



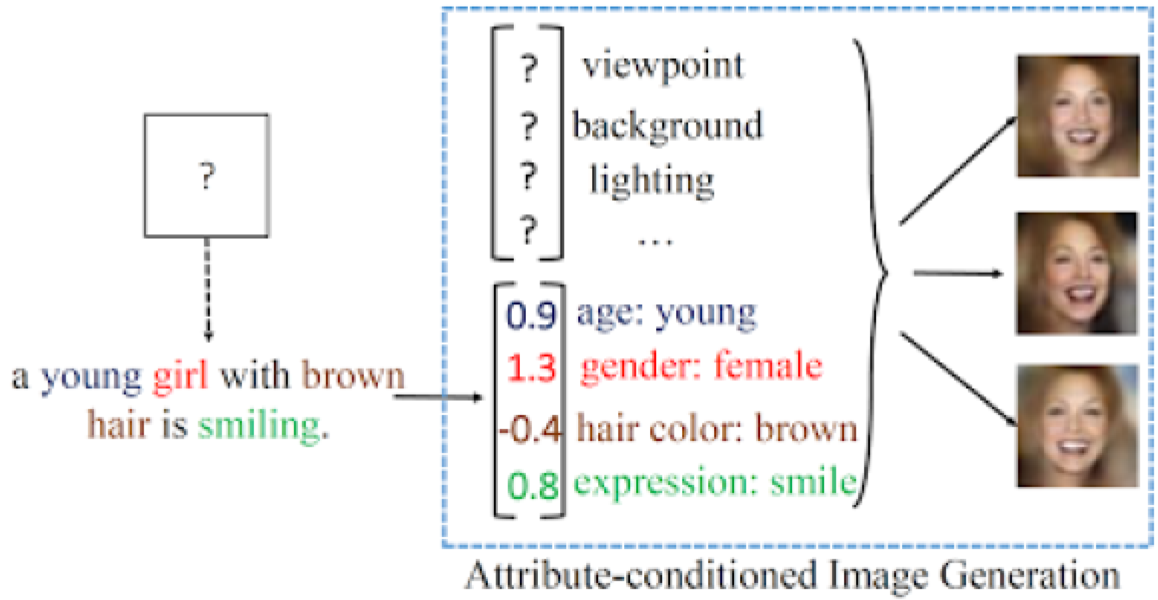
Interpreting the Latent Space

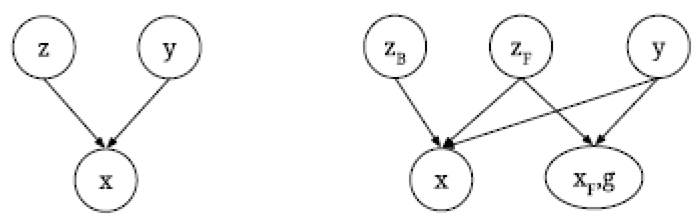


https://arxiv.org/pdf/1610.00291.pdf

Example: Attribute2Image

- Main ideas

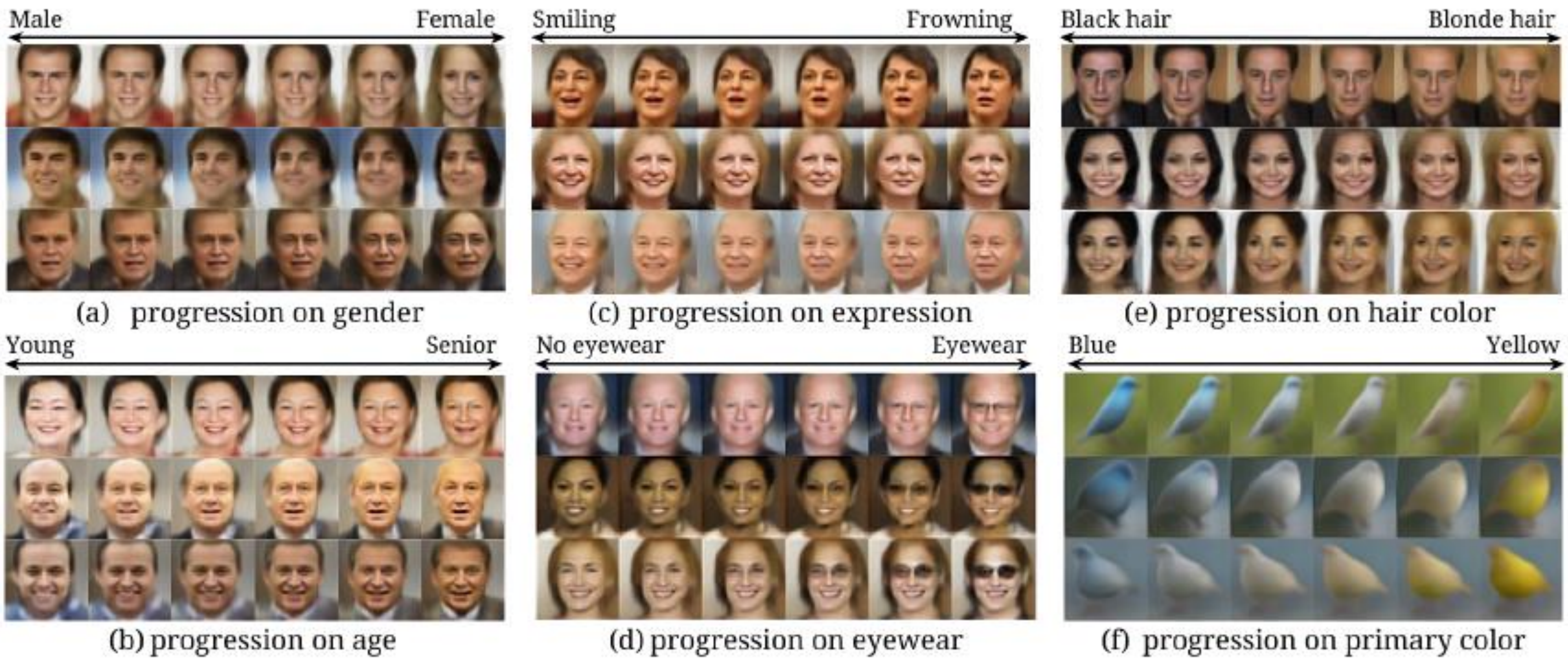




(a) CVAE: $p_\theta(x|y, z)$ (b) disCVAE: $p_\theta(x, x_F, g|y, z_F, z_B)$

Example: Attribute2Image

- Results



Problem of VAE

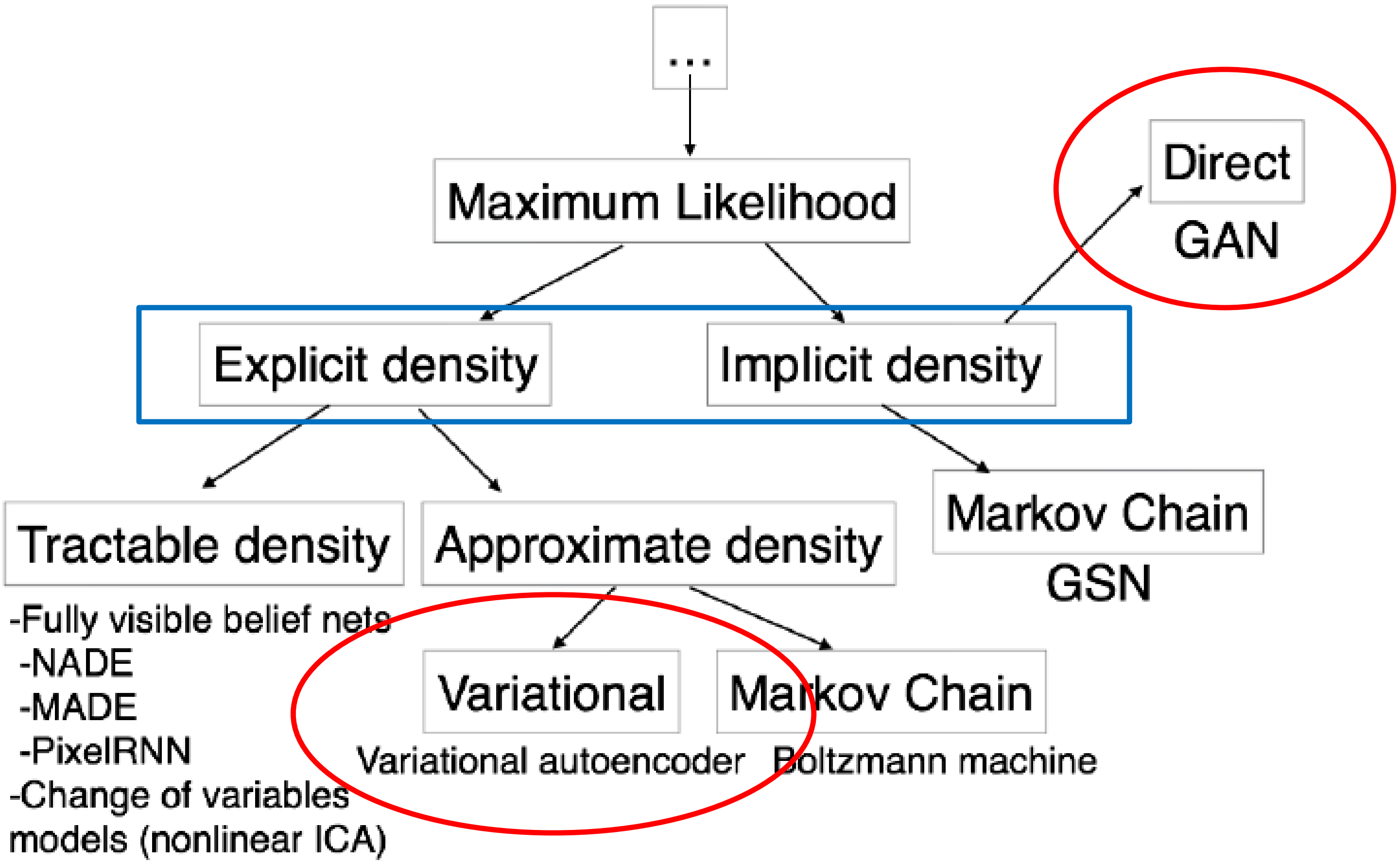
- Blurry images

https://blog.openai.com/generative-models/

Deep Generative Networks-DGM

- Overview
- Representation Learning with Autoencoder
- **Generative Adversarial Network (GAN)**
- Applications of GANs

Taxonomy of Generative Models



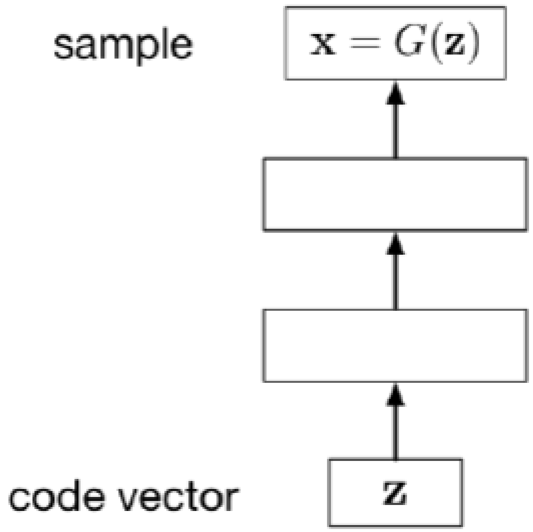
Implicit Generative Models

- Working with explicit model p(x) could be expensive
  - Variational Autoencoder (variational inference)
  - Boltzmann Machines (MCMC)

- Representation learning may not require p(x)
  - Sometimes we are more interested in taking samples from p(x) instead of p itself

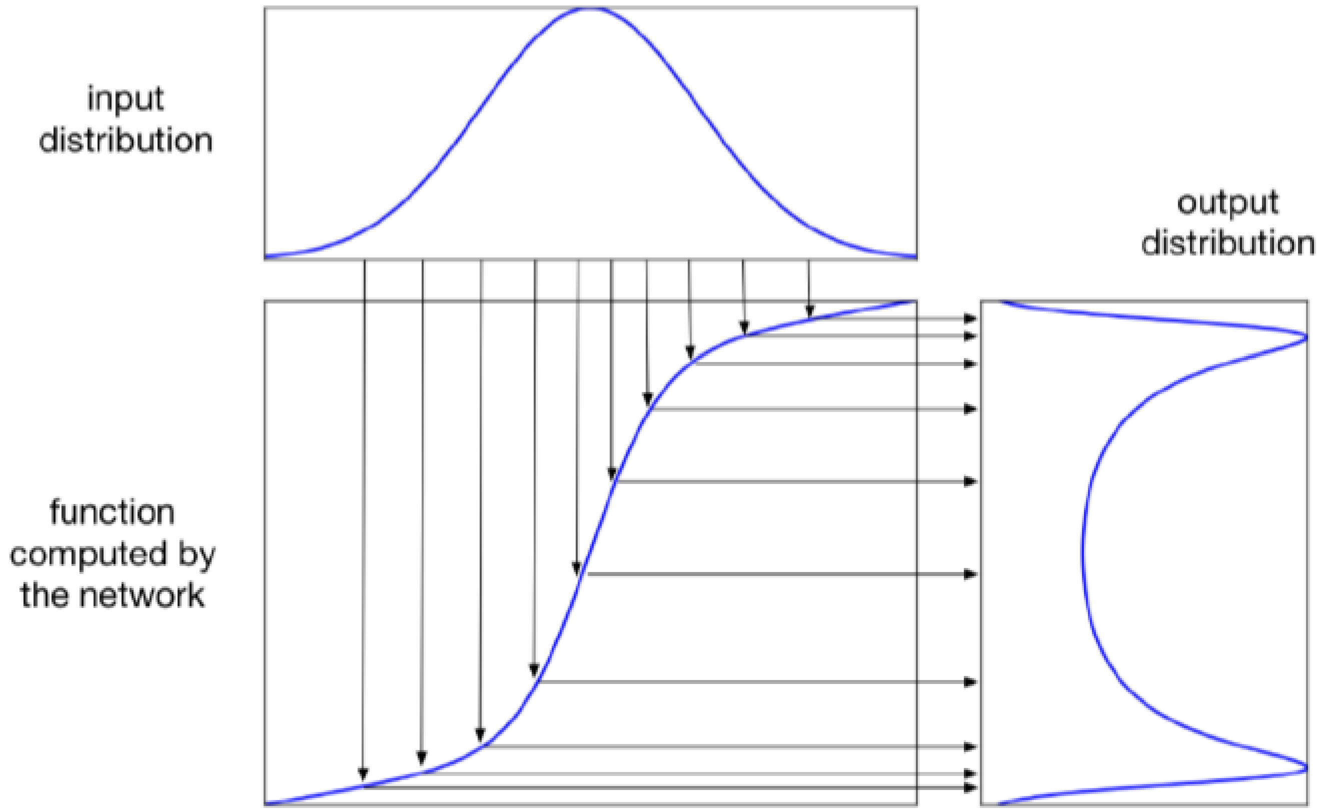
Implicit Generative Models

- Implicitly define a probability distribution
- Start by sampling the code vector z from a fixed, simple distribution
- A generator network computes a differentiable function G mapping z to an x in data space



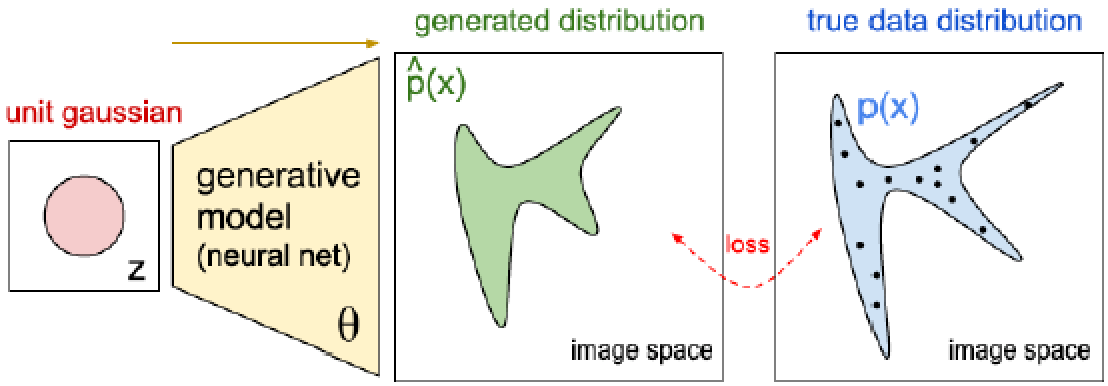
Implicit Generative Models

- Intuition: 1D example



Implicit Generative Models

- Intuition

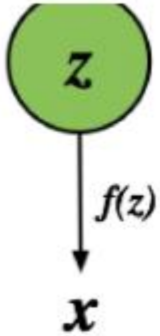


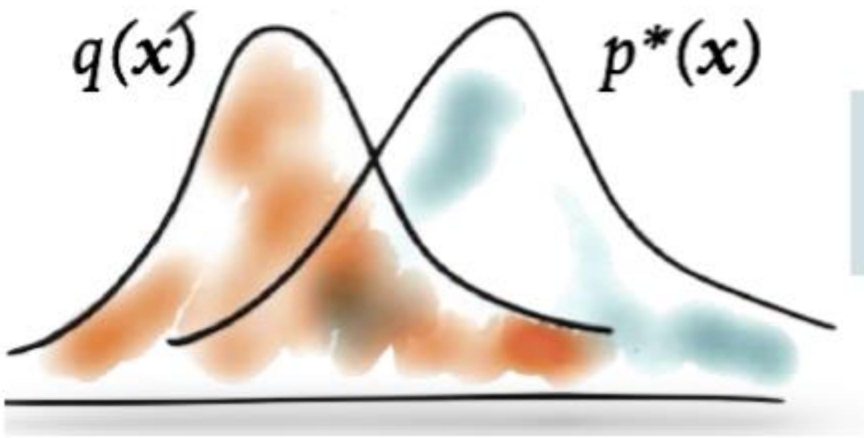
advocate/penalize samples within the blue/white region.

Learning by Comparison

- Basic idea

For some models, we only have access to an unnormalised probability, partial knowledge of the distribution, or a simulator of data.

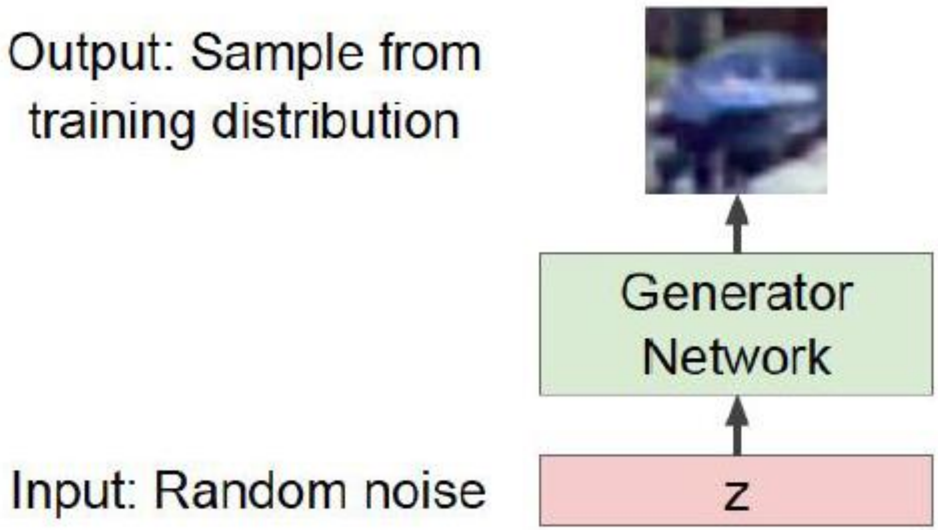




We compare the estimated distribution q(x) to the true distribution p*(x) using samples.

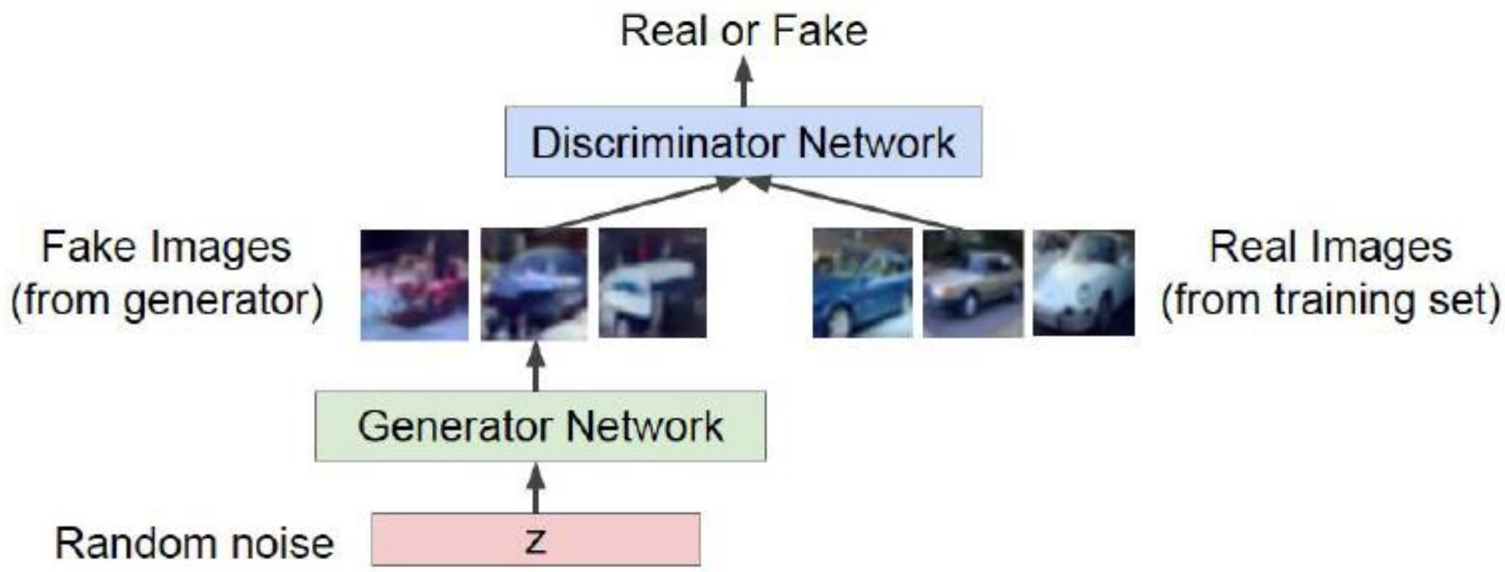
Generative Adversarial Networks (GAN)

- Using a neural network to generate data



Generative Adversarial Networks (GAN)

- Using another neural network to determine if the data is real or not

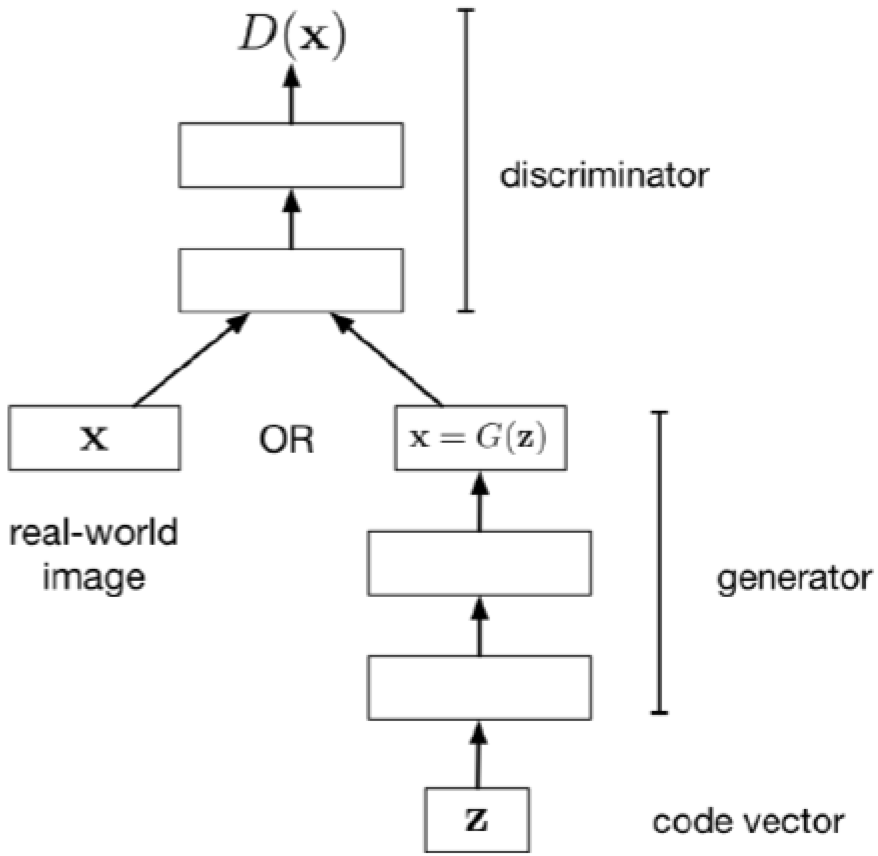


Adversarial Learning

- GAN objective for the generator is some complicated objective function defined by a neural network.
  - This means a new way of thinking about "distance".
  - We are training networks to minimize the "distance" or "divergence" between generated images and real images.
  - Instead of some hand-crafted distance metric like L1 or L2, we can make something completely new.
  - A neural network, with the right architecture, is arguably the definition of perceptual similarity (assuming our visual system is some sort of neural network).

Adversarial Learning

- Adversarial loss



Adversarial Learning

- Let D denote the discriminator's predicted probability of being real data
- Discriminator's cost function: cross-entropy loss for task of classifying real vs. fake images

$$\mathcal{J}_D = \mathbb{E}_{\mathbf{x}\sim\mathcal{D}}[-\log D(\mathbf{x})] + \mathbb{E}_{\mathbf{z}}[-\log(1 - D(G(\mathbf{z})))]$$

- One possible cost function for the generator: the opposite of the discriminator's

$$\begin{aligned}\mathcal{J}_G &= -\mathcal{J}_D \\ &= \text{const} + \mathbb{E}_{\mathbf{z}}[\log(1 - D(G(\mathbf{z})))]\end{aligned}$$

Two-Player Game

- Minimax formulation
  - The generator and discriminator are playing a zero-sum game against each other

$$\max_{G} \min_{D} \mathcal{J}_D$$

  - Using parametric models

Minimax objective function:

$$\min_{\theta_g} \max_{\theta_d} \left[\mathbb{E}_{x \sim p_{data}} \log \underbrace{D_{\theta_d}(x)}_{\text{Discriminator output for real data x}} + \mathbb{E}_{z \sim p(z)} \log(1 - \underbrace{D_{\theta_d}(G_{\theta_g}(z))}_{\text{Discriminator output for generated fake data G(z)}}) \right]$$

Discriminator outputs likelihood in (0,1) of real image

Learning Procedure

- Minimax objective function

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

Alternate between:

1. **Gradient ascent** on discriminator

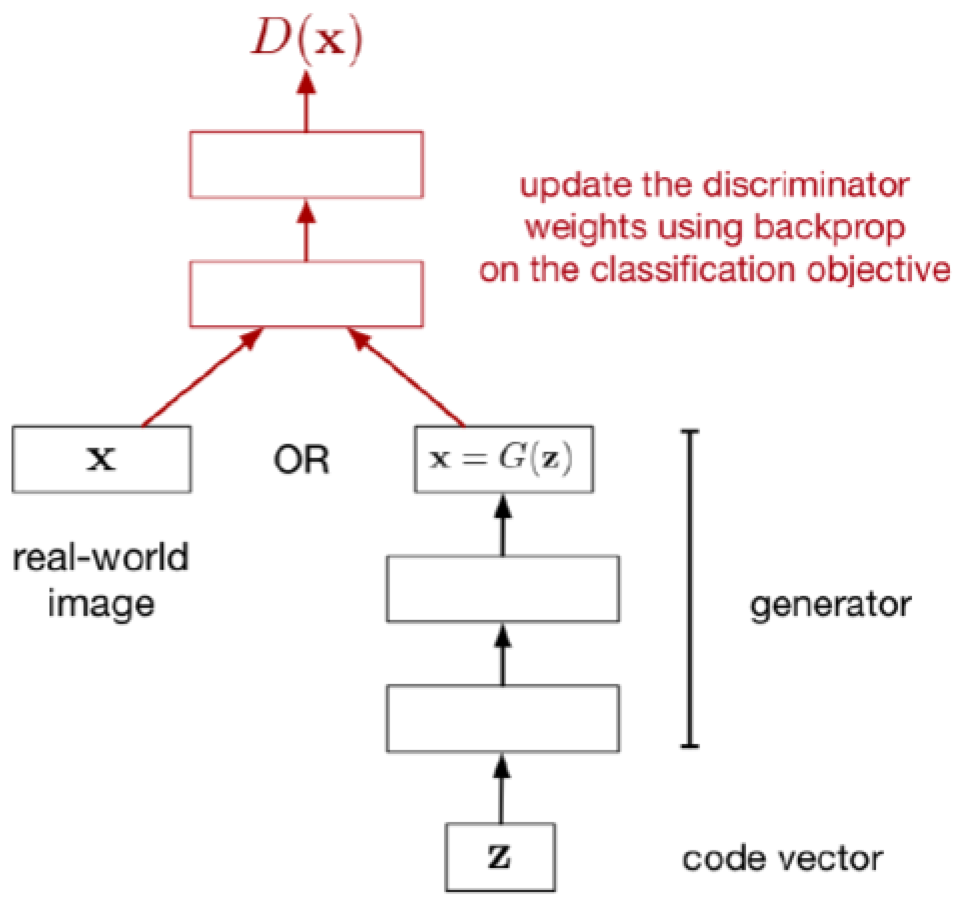
$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \right]$$

2. **Gradient descent** on generator

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

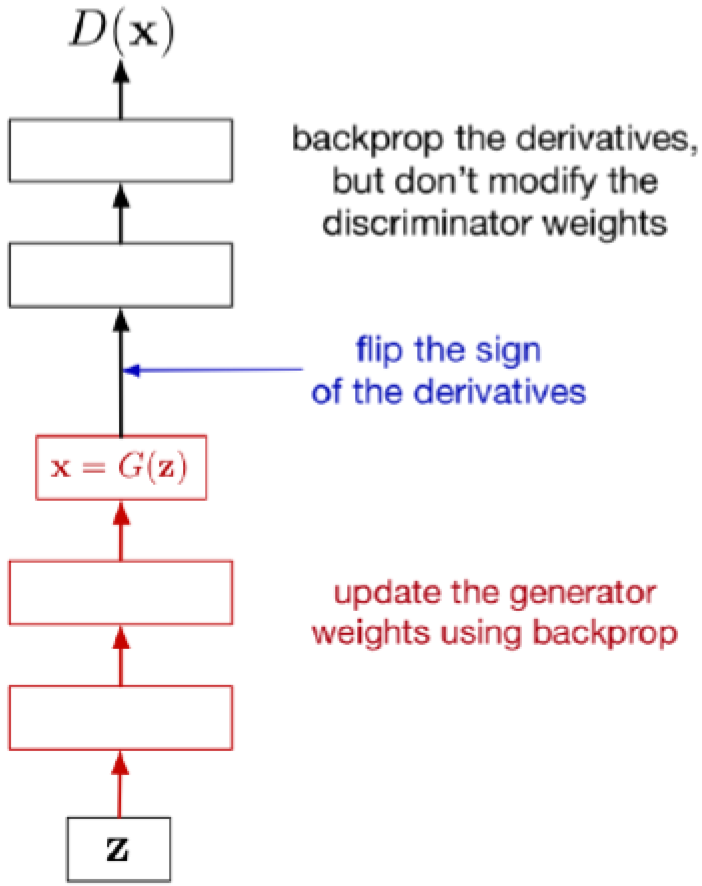
Learning Procedure

- Updating the discriminator

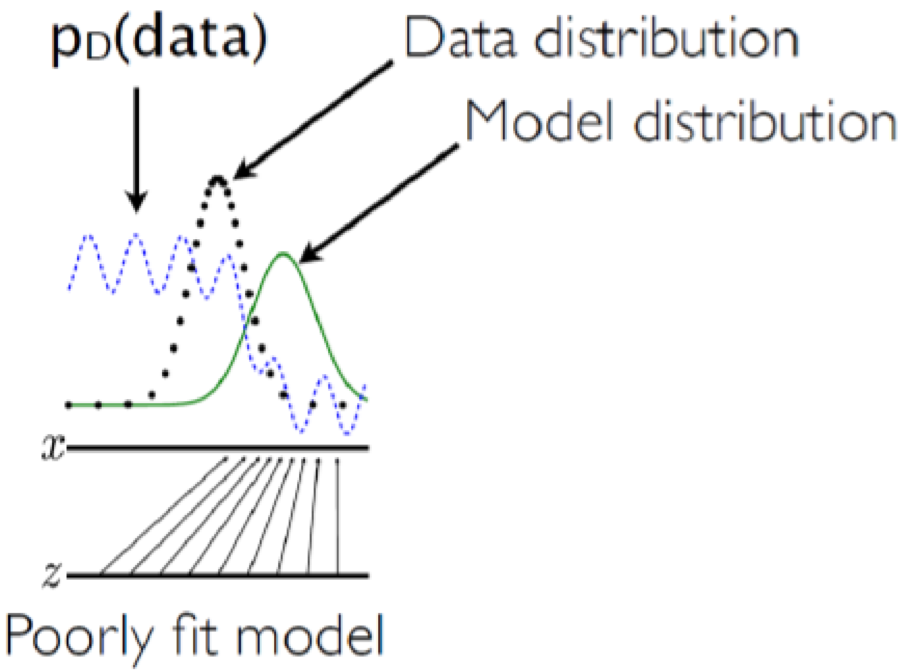


Learning Procedure

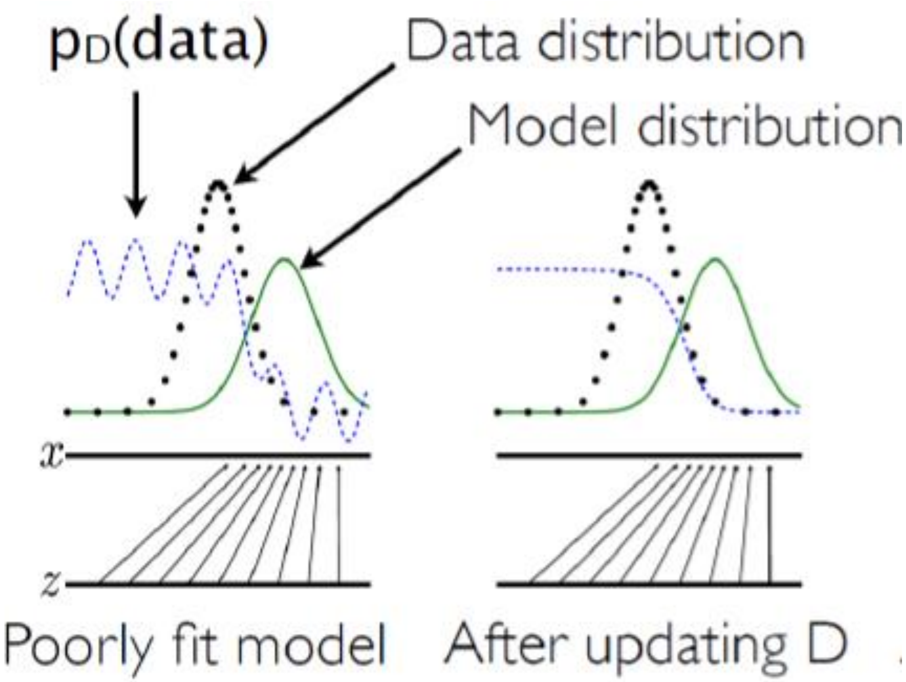
- Updating the generator



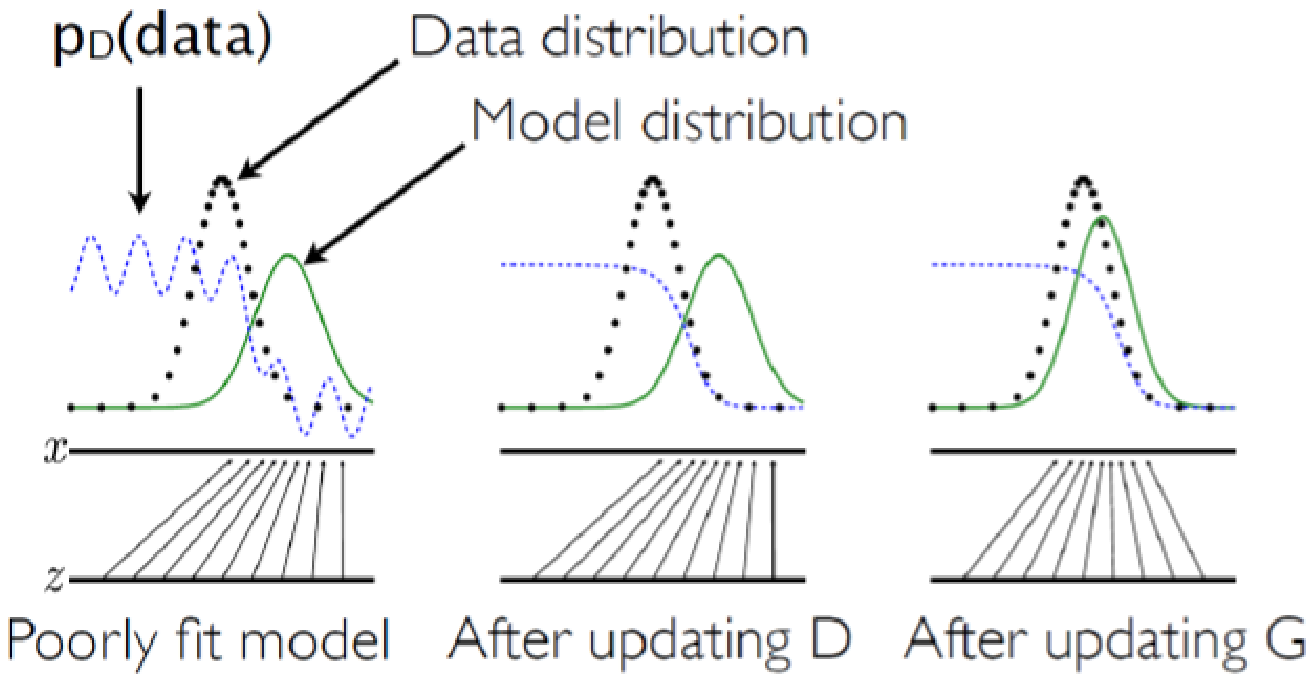
Training GANs



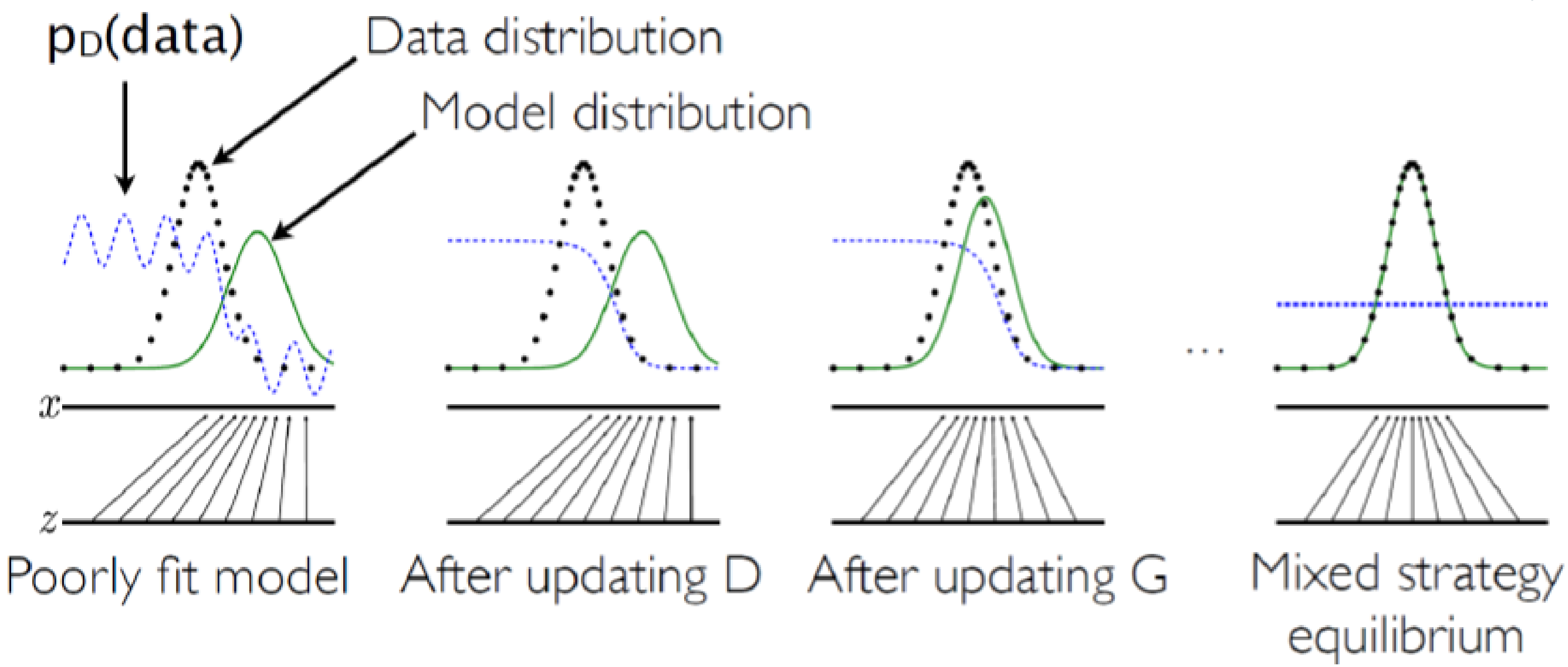
Training GANs



Training GANs

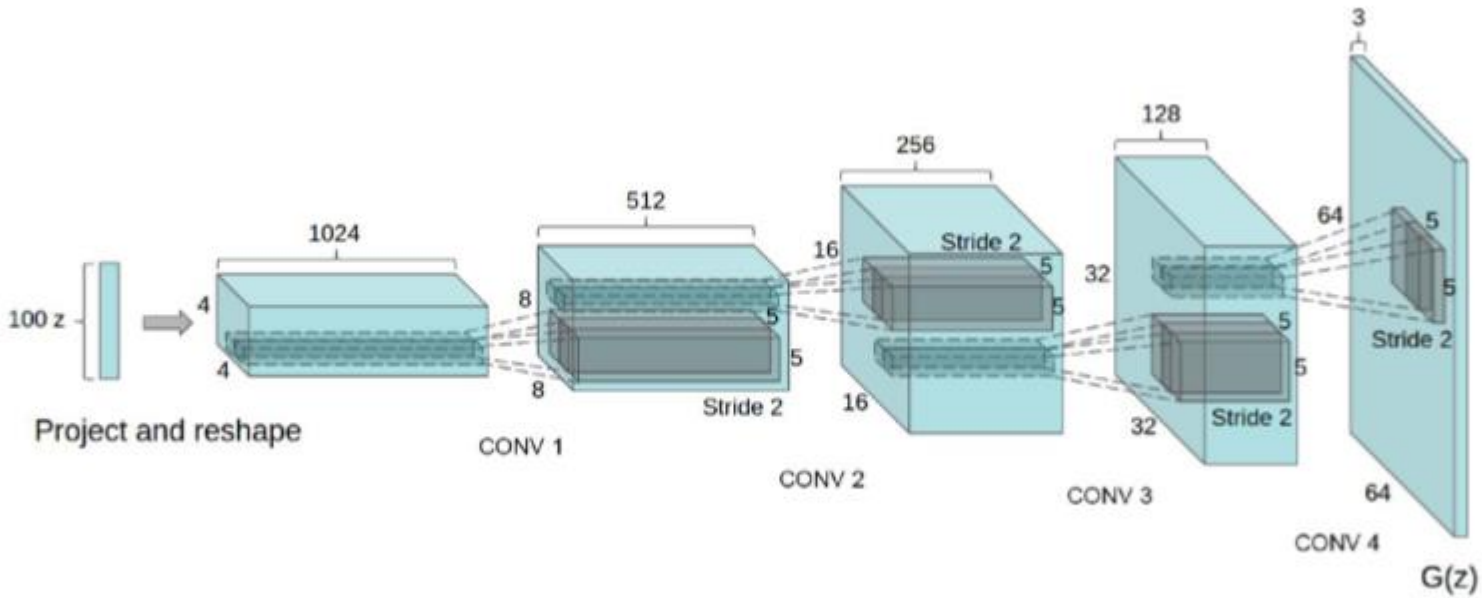


Training GANs



Typical Generator Architecture

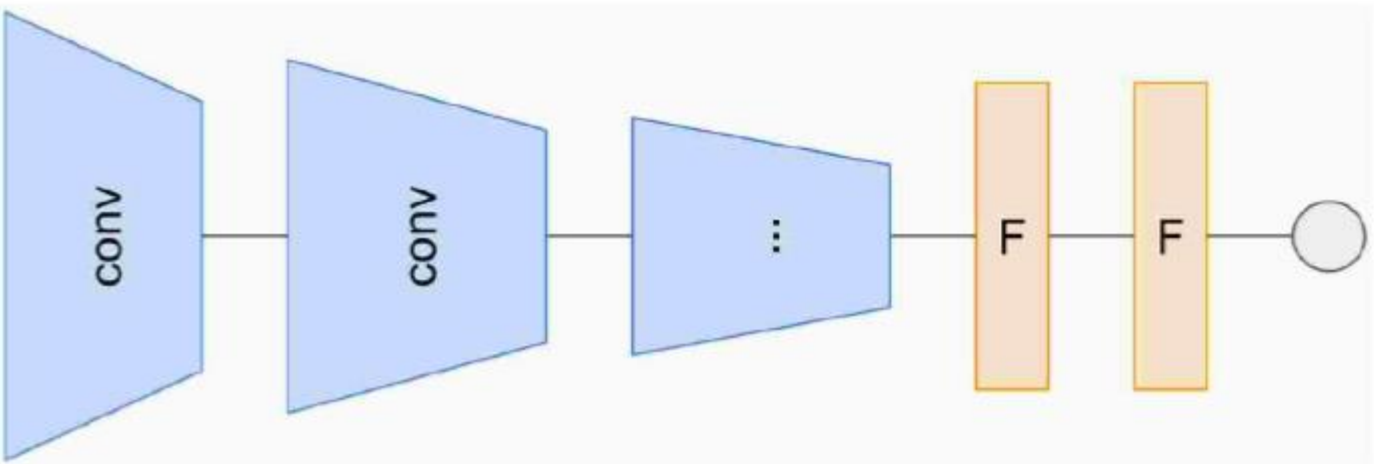
- For images



- Unit Gaussian distribution on z, typically 10-100 dim.
- Up-convolutional deep network (reverse recognition CNN)

Typical Discriminator Architecture

- For images



  - Recognition CNN model
  - Binary classification output: real / synthetic

Training GANs

- Since GANs were introduced in 2014, there have been hundreds of papers introducing various architectures and training methods
- GAN Zoo: https://github.com/hindupuravinash/the-gan-zoo
- In general, training a GAN is tricky and unstable
- Many tricks:
  - S. Chintala, How to train a GAN, ICCV 2017 tutorial
  - https://github.com/soumith/talks/blob/master/2017-ICCV_Venice/How_To_Train_a_GAN.pdf

Generative Samples

Celebrities:

Karras et al., 2017. Progressive growing of GANs for improved quality, stability, and variation

Generative Samples

Objects:

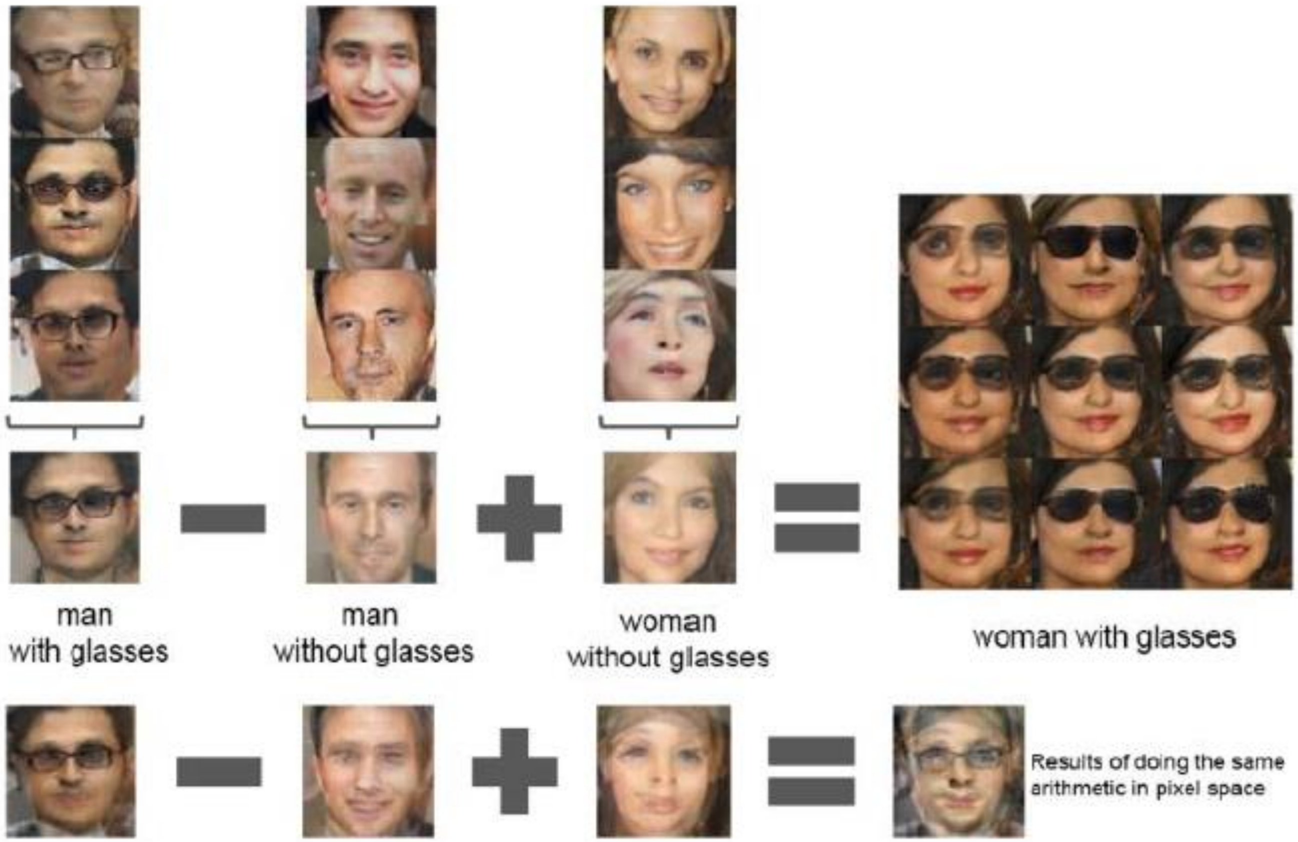
Walk Around Data Manifold

Interpolating between random points in laten space

Radford et al, ICLR 2016

Walk Around Data Manifold

- Vector Arithmetic



Deep Generative Networks-DGM

- Overview
- Representation Learning with Autoencoder
- Generative Adversarial Network (GAN)
- **Applications of GANs**

Conditional GANs

- Conditional GANs include a label and learn P(X|Y)
  - Add conditional variable y into G and D
  - Objective function

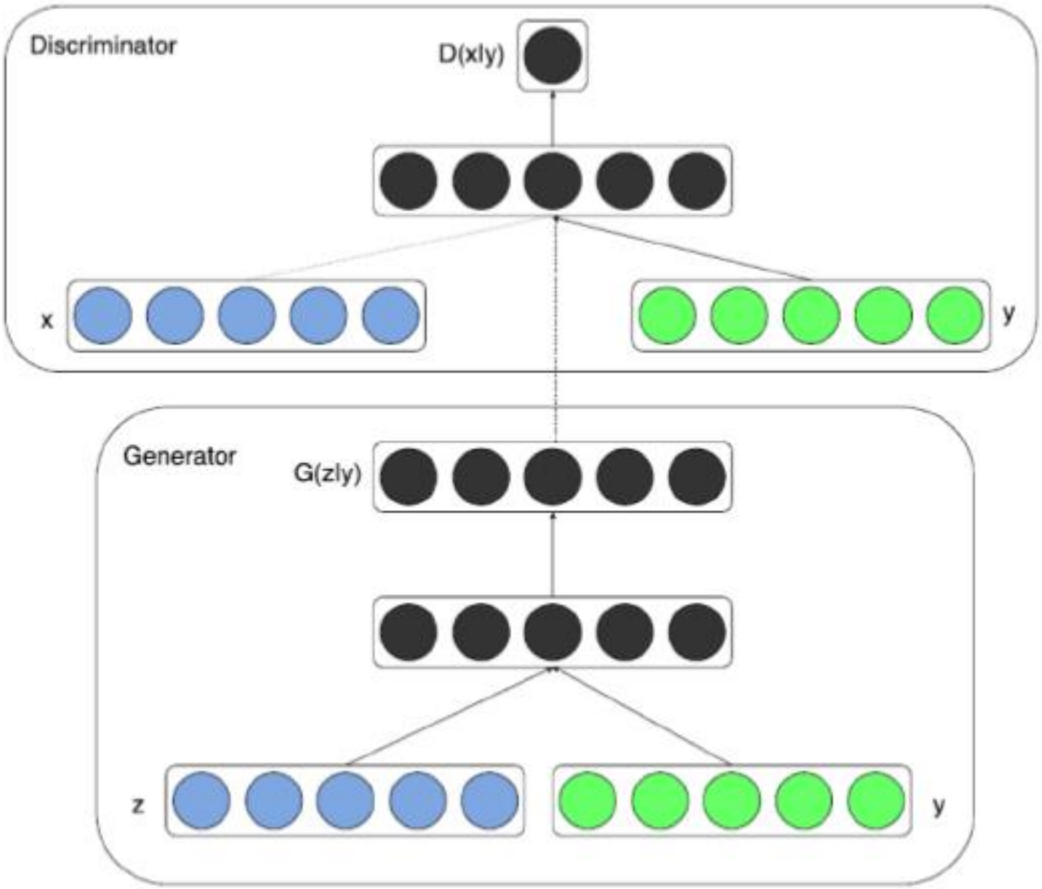
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z})))].$$

$$\Downarrow$$

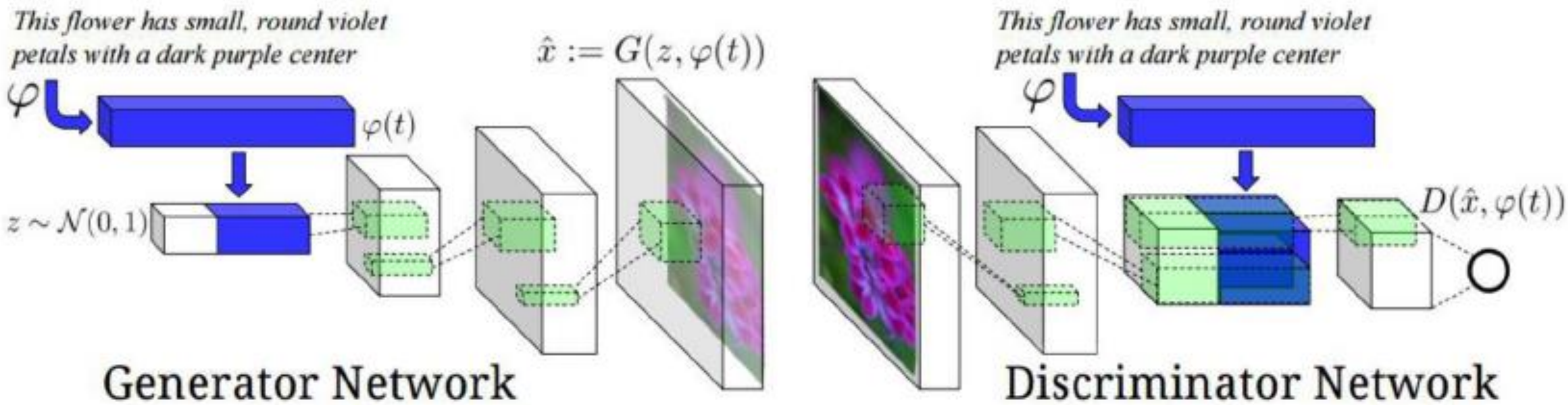
$$\min_G \max_D V(D,G) = \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}(\boldsymbol{x})}[\log D(\boldsymbol{x}|\boldsymbol{y})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}(\boldsymbol{z})}[\log(1 - D(G(\boldsymbol{z}|\boldsymbol{y})))].$$

Conditional GANs

- Model architecture



Conditional GANs



Reed et al 2015

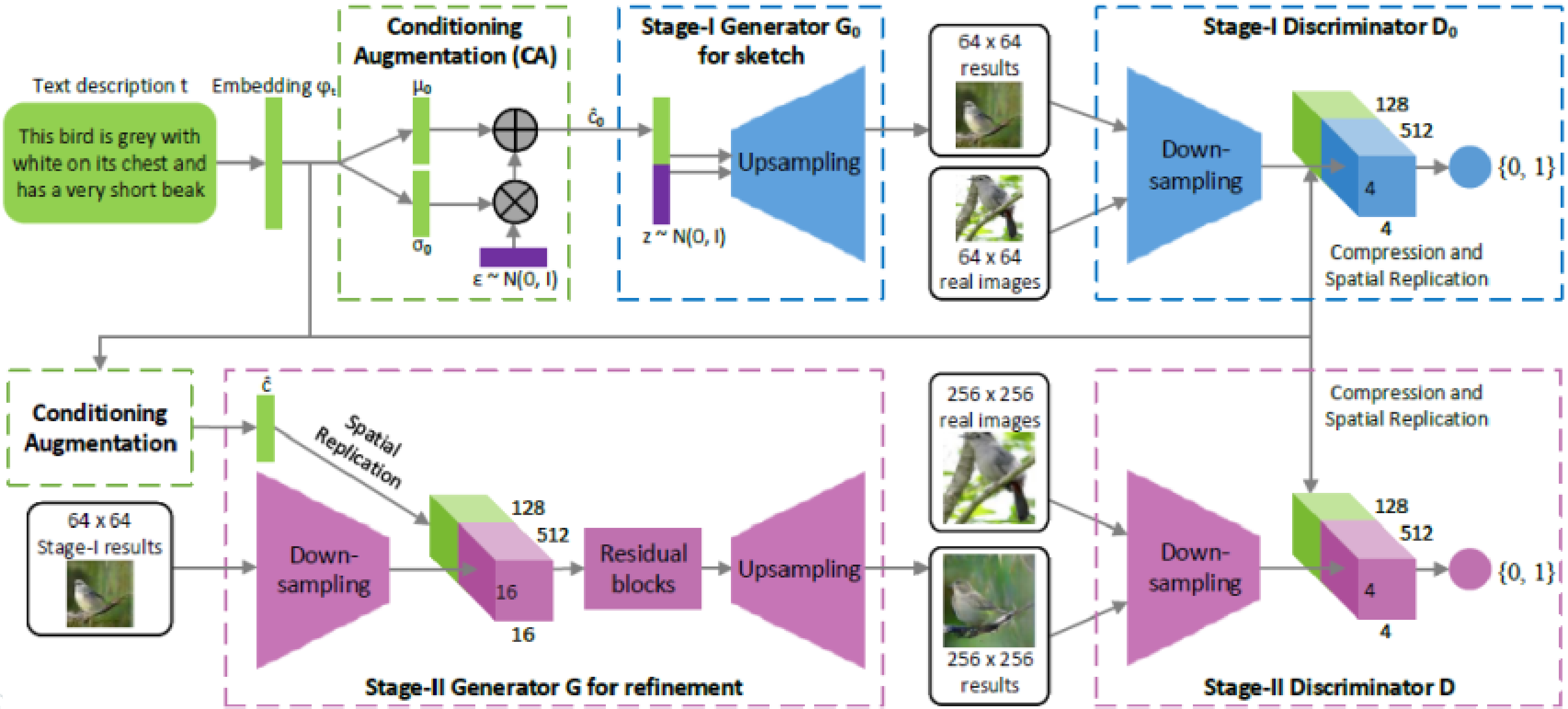
StackGAN

- A coarse-to-fine manner

Zhang et al. 2016

StackGAN

- Use stacked GAN structure



More StackGAN results

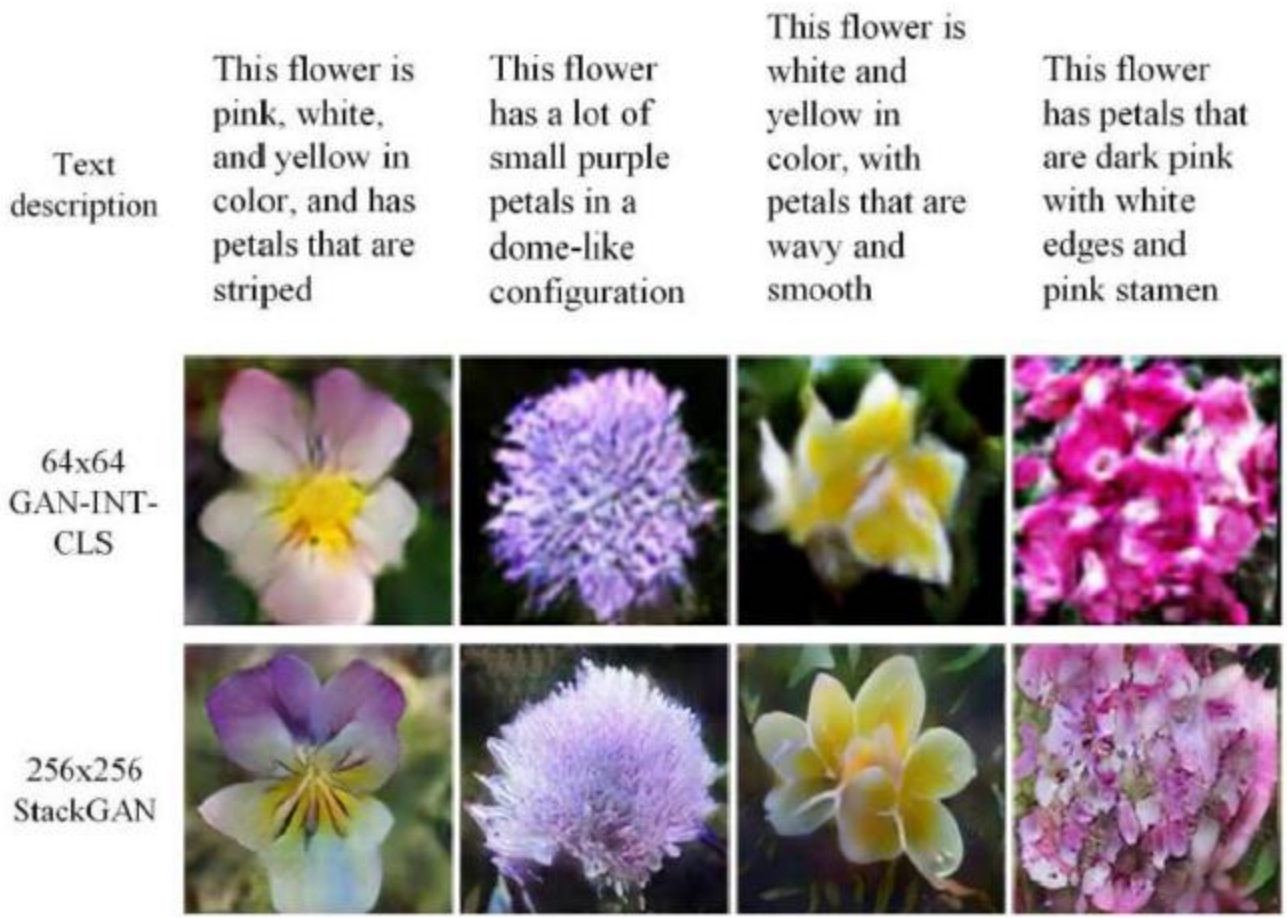


Image-to-Image Translation

- One-to-many or many-to-one mapping [Isola et al., 2016]

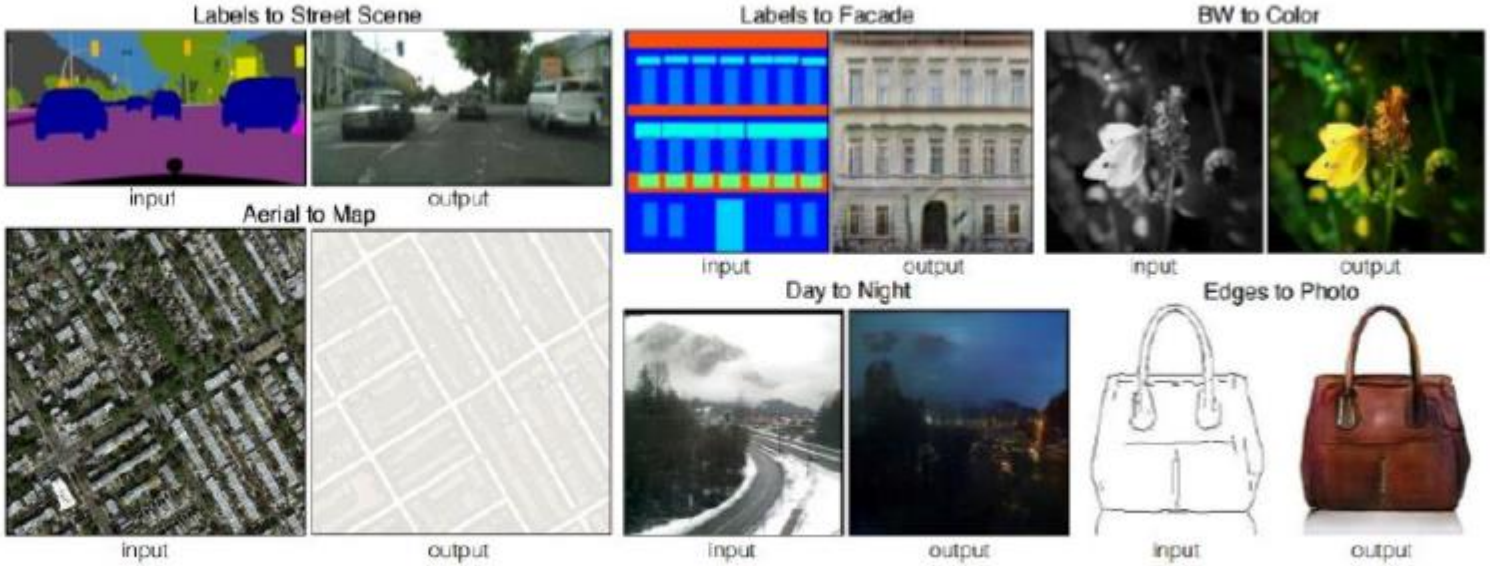
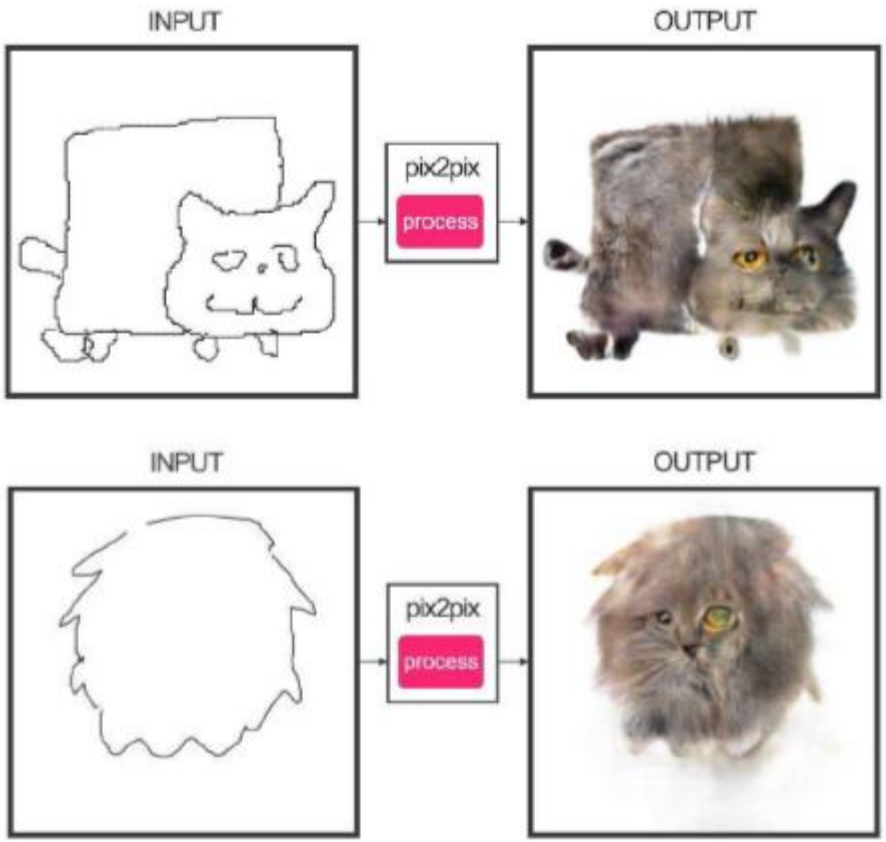


Image-to-Image Translation

- More results

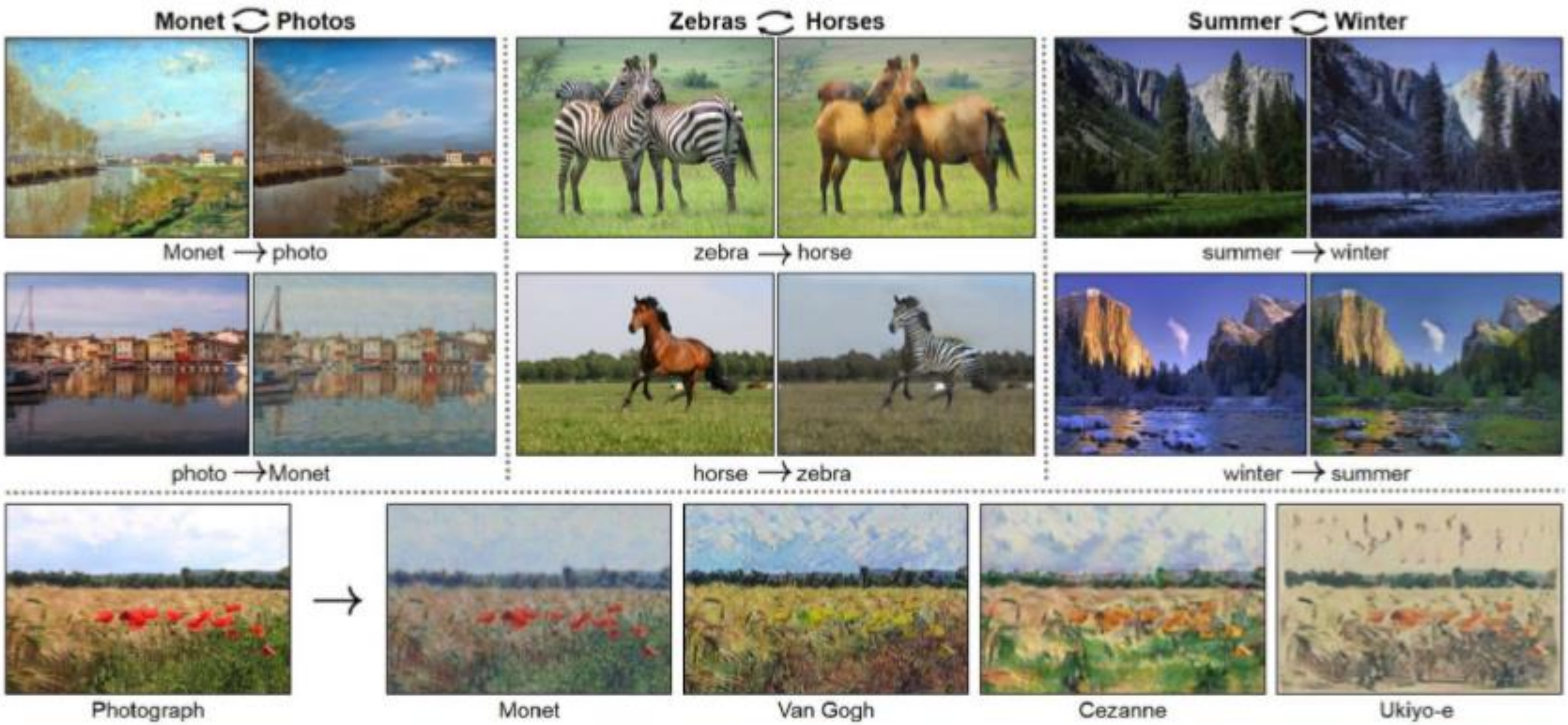
Image-to-Image Translation

- More results



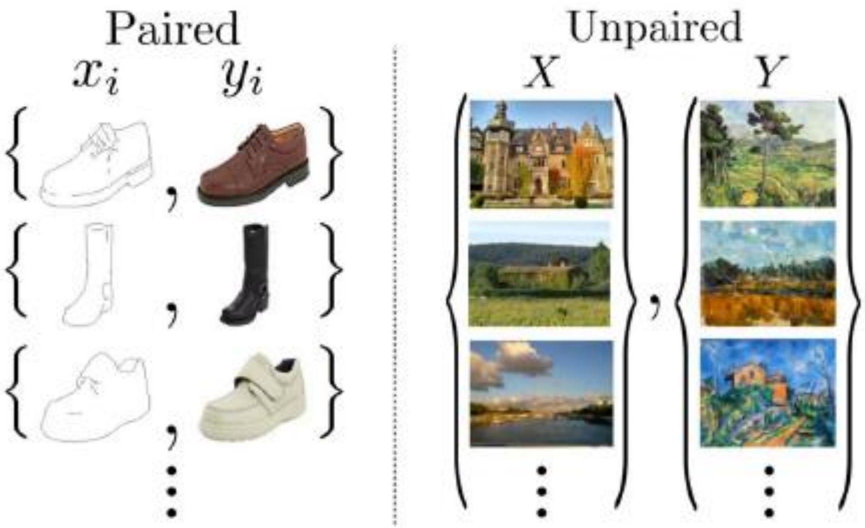
CycleGAN

- Image-to-image translation without paired data



CycleGAN

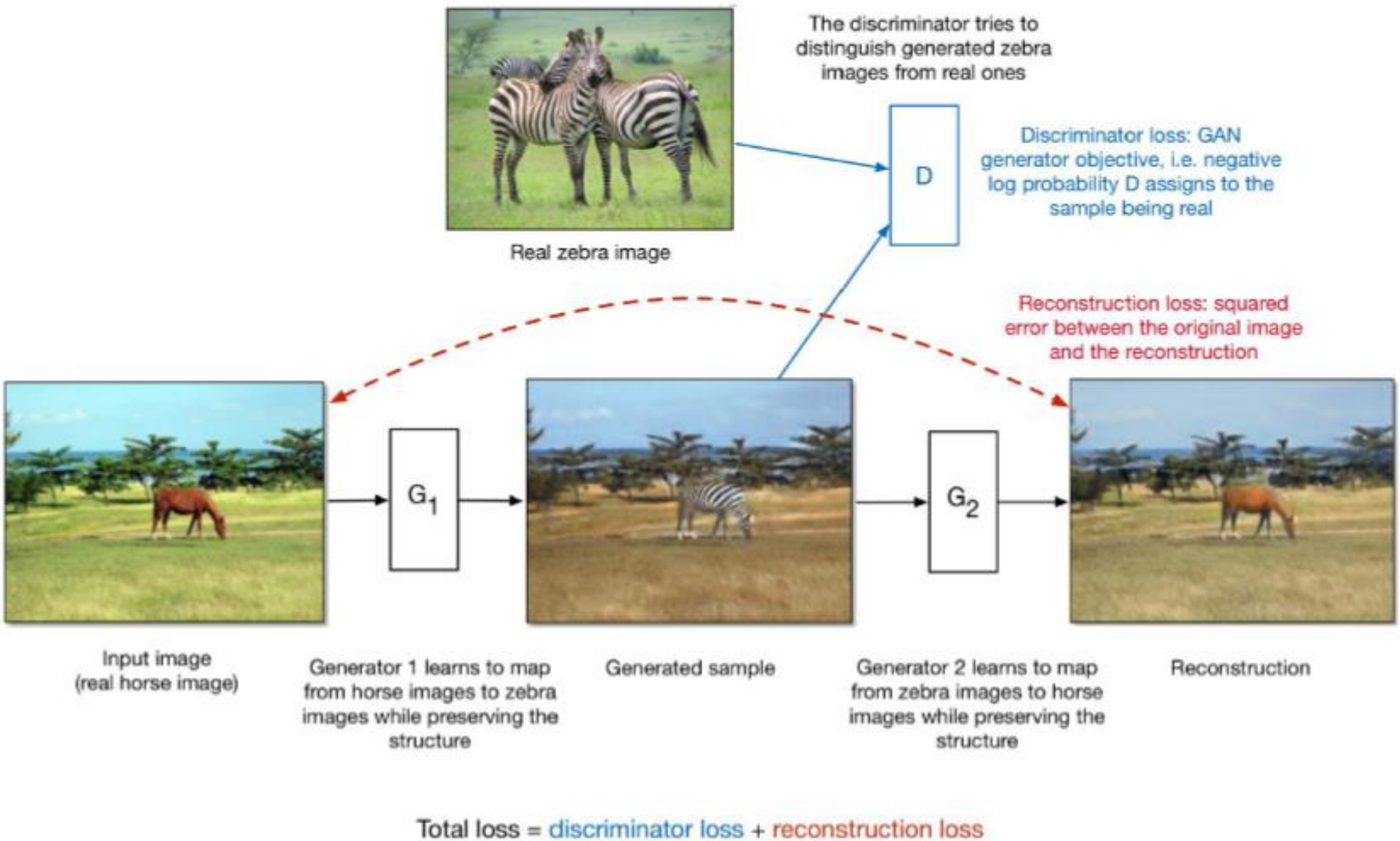
- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.



CycleGAN

- If we had paired data (same content in both styles), this would be a supervised learning problem. But this is hard to find.
- The CycleGAN architecture learns to do it from unpaired data.
  - Train two different generator nets to go from style 1 to style 2, and vice versa.
  - Make sure the generated samples of style 2 are indistinguishable from real images by a discriminator net.
  - Make sure the generators are cycle-consistent: mapping from style 1 to style 2 and back again should give you almost the original image.

CycleGAN



The discriminator tries to distinguish generated zebra images from real ones
D
Discriminator loss: GAN generator objective, i.e. negative log probability D assigns to the sample being real
Real zebra image
Reconstruction loss: squared error between the original image and the reconstruction
G1
G2
Input image (real horse image)
Generator 1 learns to map from horse images to zebra images while preserving the structure
Generated sample
Generator 2 learns to map from zebra images to horse images while preserving the structure
Reconstruction
Total loss = discriminator loss + reconstruction loss

CycleGAN

- Results

- More details

https://hardikbansal.github.io/CycleGANBlog/