CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

CSDS 600: Deep Generative Models

Generative Adversarial Network (1)

Yu Yin (yu.yin@case.edu)

Case Western Reserve University

https://yin-yu.github.io/

Surveys on Generative AI

- Multimodal Image Synthesis and Editing: The Generative AI Era, TPAMI, 2023
- Diffusion Models in Vision: A Survey, TPAMI, 2023
- A Comprehensive Survey of AI-Generated Content (AIGC): A History of Generative AI from GAN to ChatGPT, 2023
- Deep Generative Models for
  - Graph Generation
  - 3D Representations
  - Medical image
  - Chemical Sciences
  - …

Maximizing Likelihoods

- Autoregressive Models

$$p(\mathbf{x}) = \prod_{i=1}^{n} p(x_i|x_1, x_2, \ldots, x_{i-1}) = \prod_{i=1}^{n} p_\theta(x_i|x_{<i})$$

- Variational Autoencoders

$$p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$$

- Normalizing Flow Models

$$p(\mathbf{x}) = \pi(\mathbf{z})\left|\det\frac{dz}{dx}\right| = \pi(\mathbf{z})\left|\det(J_{f^{-1}})\right|$$

- All the above methods are based on Maximizing Likelihoods
- Is the likelihood a good way to measure the distance between TWO distributions?

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Generative Adversarial Network (GAN)

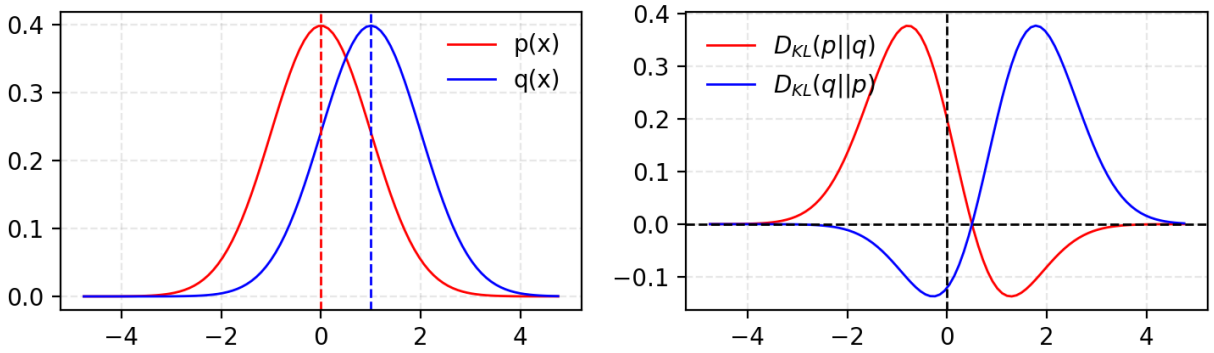
- Background on Divergences
- GAN
  - Introduction
  - Deep convolutional GAN
  - Adversarial Loss vs. MSE
- Challenges of GAN

Divergences

- **KL (Kullback–Leibler) divergence**

$$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

  - D_{KL} achieves the minimum zero when *p(x) == q(x)* everywhere.
  - D_{KL} is asymmetric.



Divergences

- **KL (Kullback–Leibler) divergence**

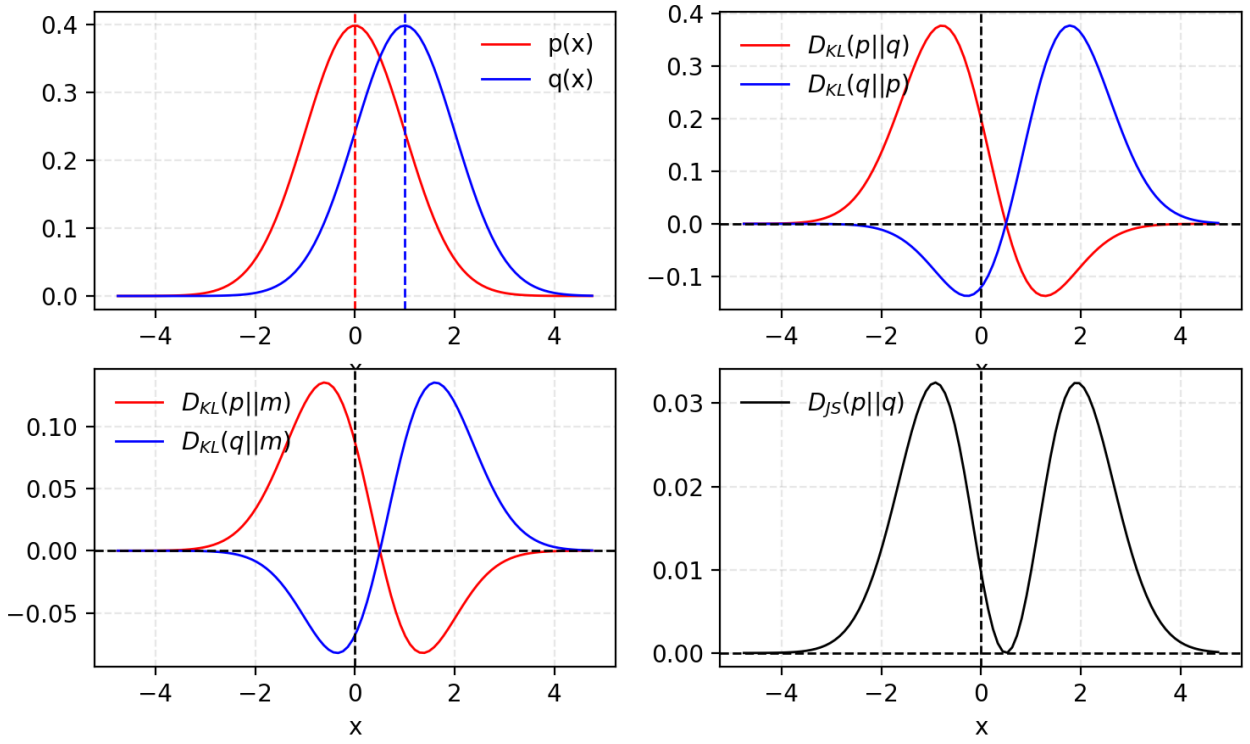
$$D_{KL}(p\|q) = \int_x p(x) \log \frac{p(x)}{q(x)} dx$$

  - D_{KL} achieves the minimum zero when *p(x) == q(x)* everywhere.
  - D_{KL} is asymmetric

- **Jensen–Shannon Divergence**

$$D_{JS}(p\|q) = \frac{1}{2} D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q\|\frac{p+q}{2})$$

  - Range [0,1]
  - D_{JS} is symmetric



m = (p+q)/2

GAN -- Introduction

Starts from the intuition of min-max game:

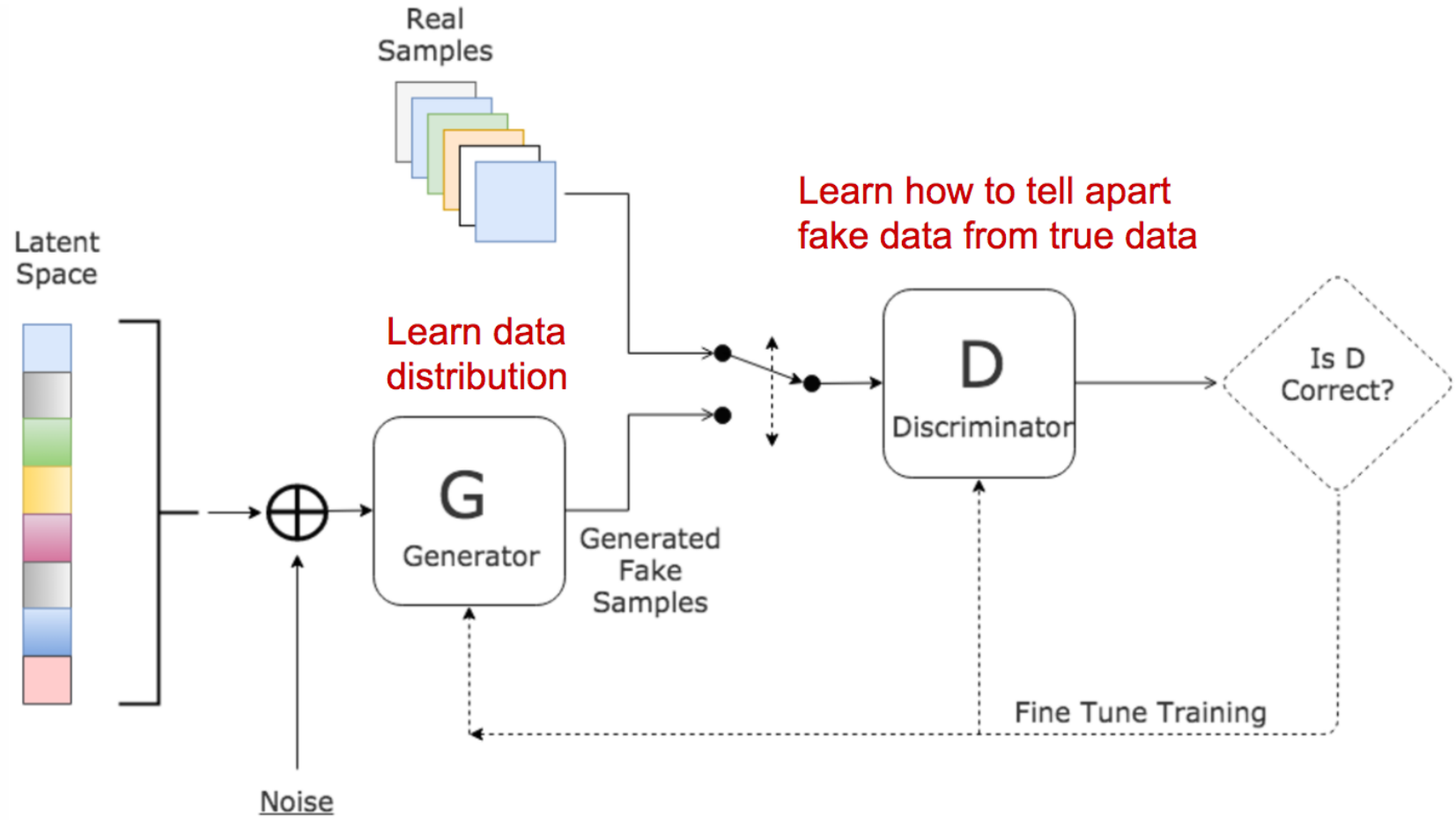
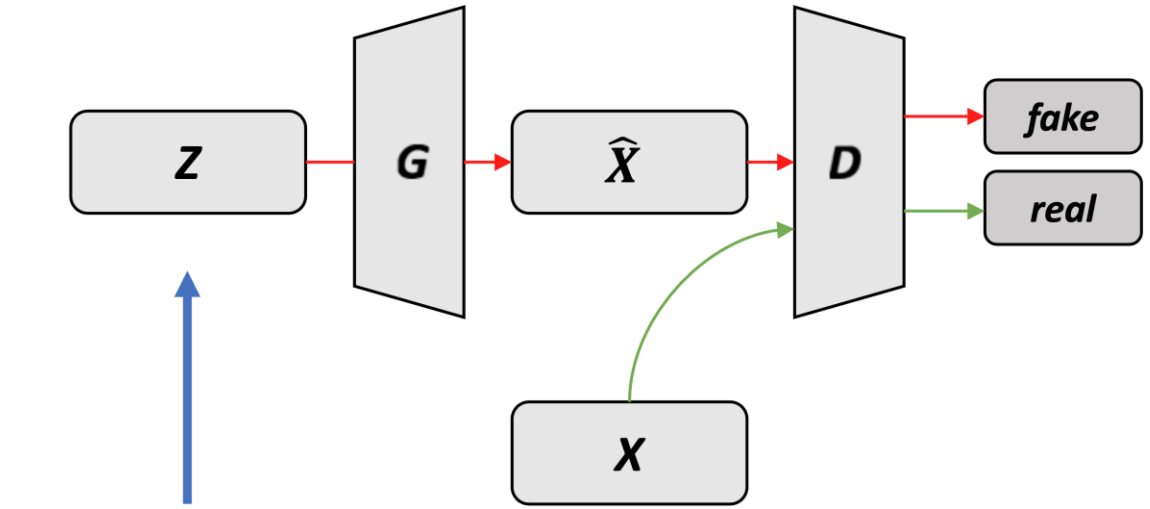


Image: www.kdnuggets.com/2017/01/generative-...-learning.html

GAN



$$\mathcal{L}_D = -\mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(\boldsymbol{x})] - \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D(G(\boldsymbol{z})))]$$

$$\mathcal{L}_G = \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D(G(\boldsymbol{z})))]$$

Goodfellow, I., Generative adversarial nets. NeuIPS'14.

GAN

$$G^* = \min_G \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D(G(\boldsymbol{z})))]$$

$$D^* = \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D(G(\boldsymbol{z})))]$$

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

GAN

We claim that this min-max game has a global optimum for $p_g = p_{data}$

- First, we claim that for G fixed, the optimum $D^* = p_{data} / (p_g + p_{data}) = 0.5$ $(p_g = p_{data})$
- Then for D^* fixed, $V(G, D) = -log4 + JS(p_{data} \,||\, p_g) = -log4 = -2log2$ $(p_g = p_{data})$
- Thus, when D and G are optimal, $p_{data} = p_g$

GAN

Dive into the Objective

- $\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_z}[\log(1 - D(G(\boldsymbol{z})))]$
- When $z\sim p(z)$, let p_g denote distribution of $G(z)$
- $\min_G \max_D V(D,G) = \min_G \max_D \mathbb{E}_{\boldsymbol{x}\sim p_{data}}[\log D(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D(\boldsymbol{x})]$

GAN

- First, we claim that for G fixed

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

$$V(G,D) = \int_x p_{data}(\boldsymbol{x}) \log(D(x)) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))\, dx$$

$$V(G,D)' = \frac{p_{data}}{D(x)} - \frac{p_g}{1 - D(x)} = 0$$

$$(\log(A))' = \frac{1}{A}$$

- The optimum $D^* = \frac{p_{data}}{p_{data}+p_g} = 0.5$

GAN

- Then for D^* fixed

$$V(G, D^*) = \mathbb{E}_{x \sim p_{data}}[\log D^*(\boldsymbol{x})] + \mathbb{E}_{x \sim p_g}[\log(1 - D^*(\boldsymbol{x}))]$$

$$= \mathbb{E}_{x \sim p_{data}}\left[\log \frac{p_{data}}{p_{data} + p_g}\right] + \mathbb{E}_{x \sim p_g}\left[\log \frac{p_g}{p_{data} + p_g}\right]$$

- Recap $D_{JS}(p\|q) = \frac{1}{2} D_{KL}(p\|\frac{p+q}{2}) + \frac{1}{2} D_{KL}(q\|\frac{p+q}{2})$

- Then $V(G, D^*) = -log4 + 2 * D_{JS}(p_{data}\|p_g) = -2log2$

$$V(G^*, D^*) = -2log2$$

Training loop:

Algorithm 1 Minibatch stochastic gradient descent training of generative adversarial nets. The number of steps to apply to the discriminator, k, is a hyperparameter. We used $k = 1$, the least expensive option, in our experiments.

for number of training iterations **do**

  for k steps **do**

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Sample minibatch of m examples $\{\boldsymbol{x}^{(1)}, \ldots, \boldsymbol{x}^{(m)}\}$ from data generating distribution $p_{\text{data}}(\boldsymbol{x})$.
  - Update the discriminator by ascending its stochastic gradient:

$$\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(\boldsymbol{x}^{(i)}\right) + \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right) \right].$$

  end for

  - Sample minibatch of m noise samples $\{\boldsymbol{z}^{(1)}, \ldots, \boldsymbol{z}^{(m)}\}$ from noise prior $p_g(\boldsymbol{z})$.
  - Update the generator by descending its stochastic gradient:

$$\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).$$

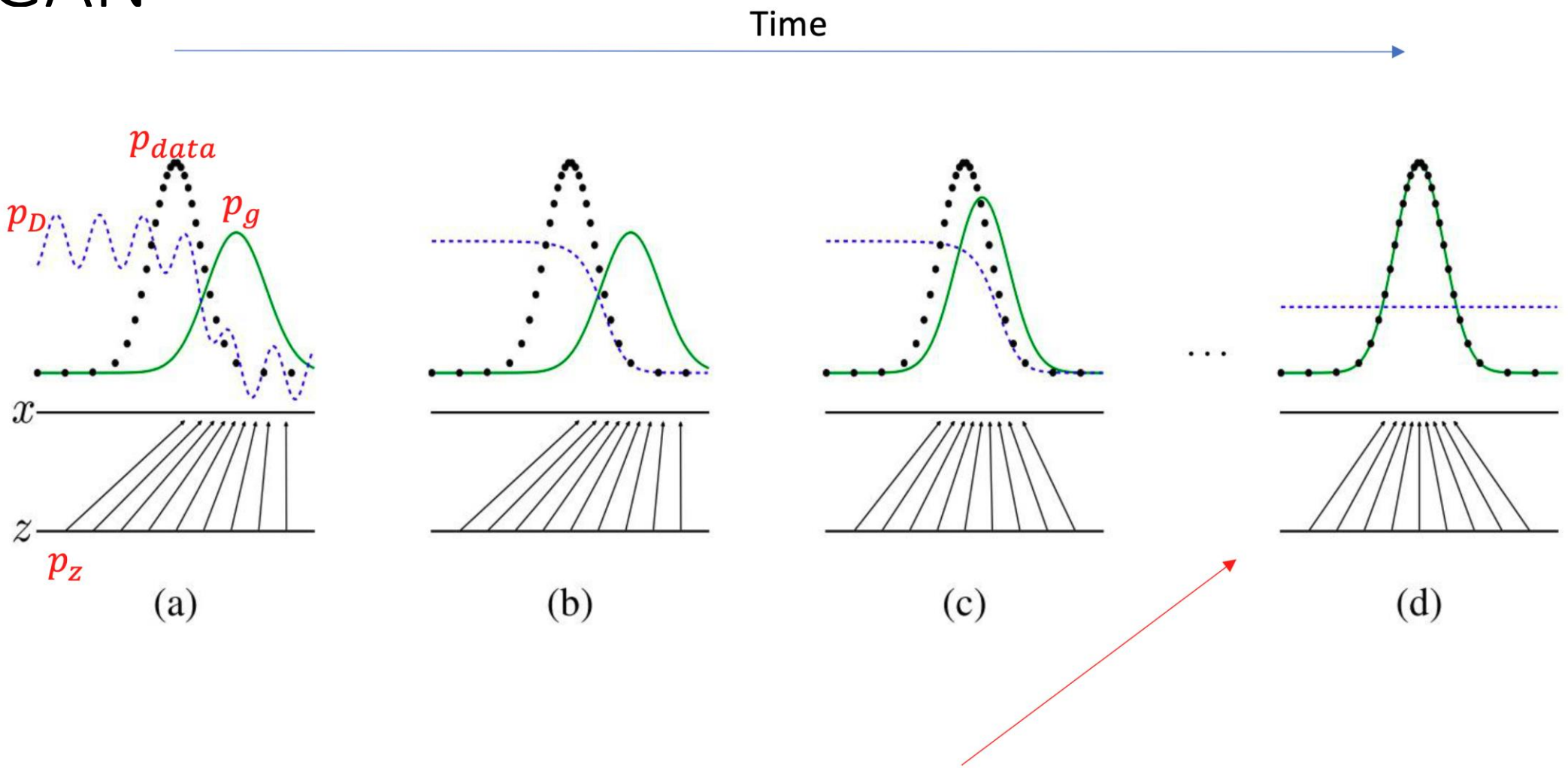
end for

The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

- Sample a batch {z^i, x^i}
- updating D
- updating G

Why not updating D to its optimum first?

GAN



The discriminator is unable to differentiate between the two distributions, i.e. D(x) = 0.5

GAN

- Experiments of Vanilla GAN on MNIST and TFD
  - Random sample

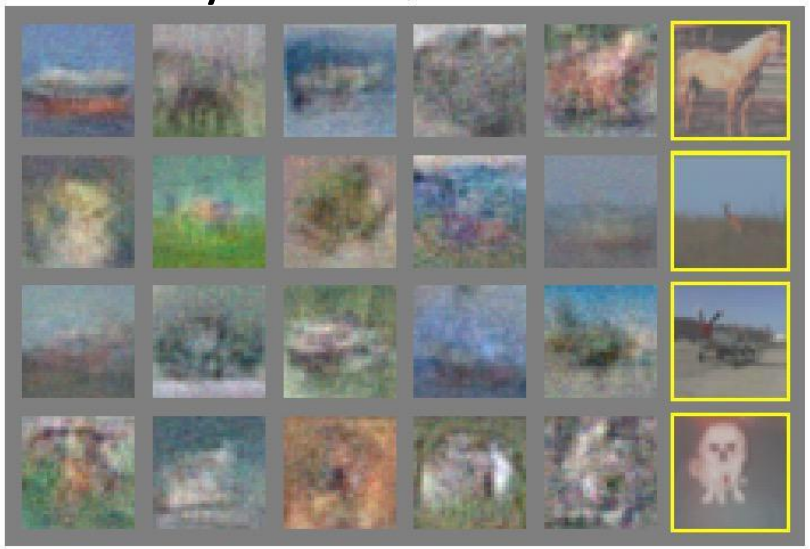
  - Interpolation on Latent Codes

Goodfellow, I., Generative adversarial nets. NeuIPS'14.

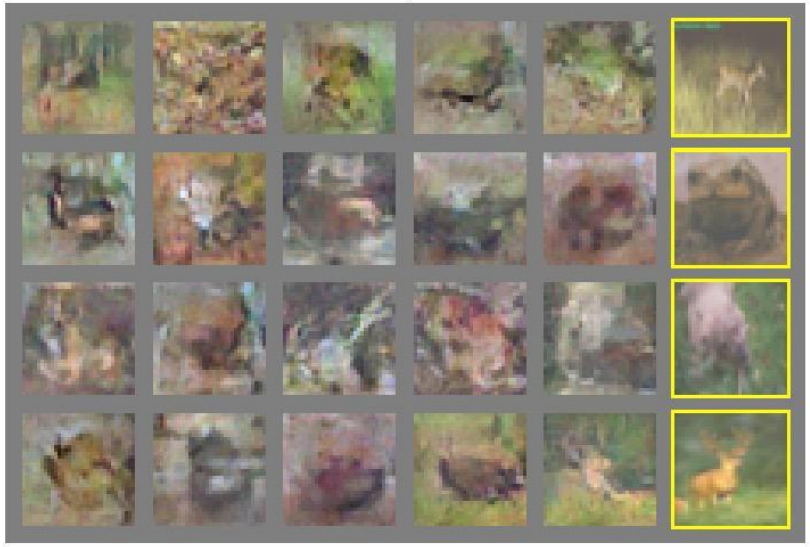
GAN

- Experiments of Vanilla GAN on CIFAR10

Fully connected model



Convolutional model



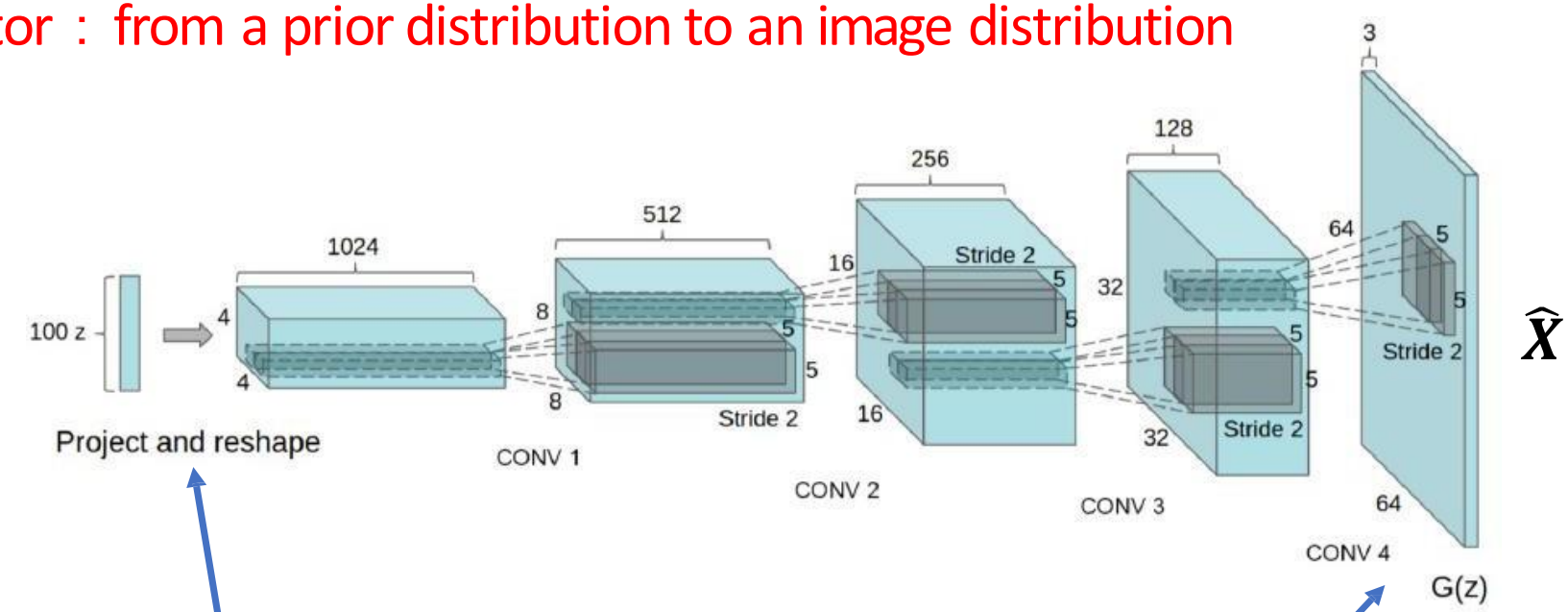
Can we get better performance?

Goodfellow, I., Generative adversarial nets. NeuIPS'14.

Deep convolutional GAN (DCGAN)

- Using the Power of Convolutional Nets

Generator ： from a prior distribution to an image distribution

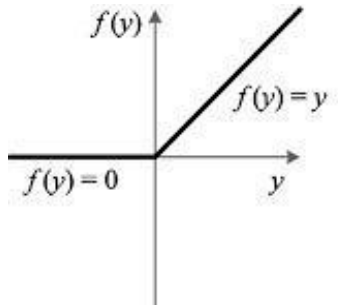


Normal distribution as the latent distribution
z = 100 values

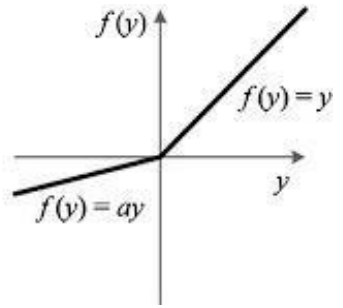
Data distribution x
= 64x64x3 values

Deep convolutional GAN

- GAN is difficult to train: 1. small G gradient if D is good. 2. oscillation
- DCGAN tricks
  - Batch normalization on all layers except the final layer of G and input layer of D, with a decay of 0.9 (default was 0.99)
  - Adam optimizer with a 1st order momentum (beta1) of 0.5 (default was 0.9)
  - Leaky ReLU with an alpha of 0.2 (default was ReLU)
  - Strided convolutions (default was Maxpooling)
  - Learning rate of 0.0002 (default was 0.0001)
  - More: https://github.com/soumith/ganhacks#authors



ReLU



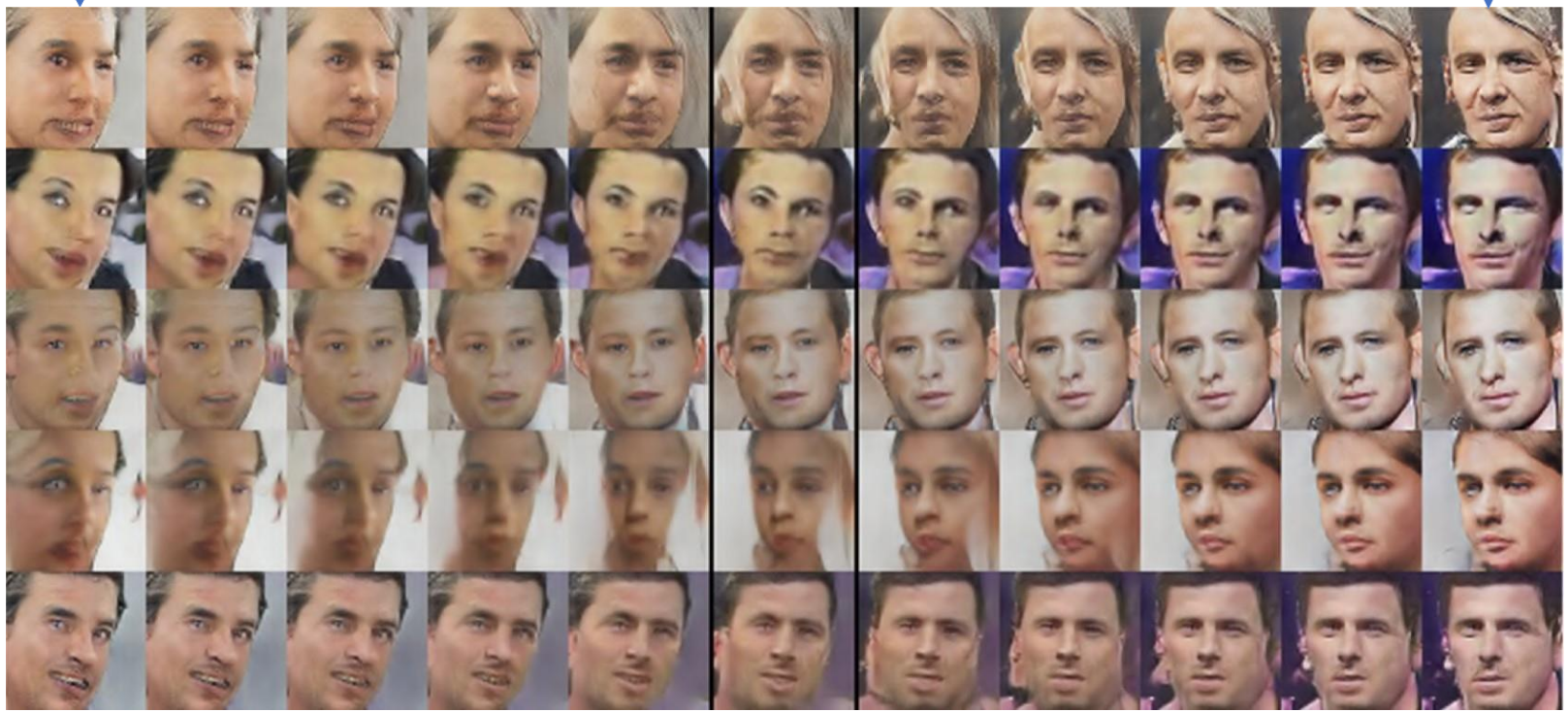
Leaky ReLU

Deep convolutional GAN

- Latent Representation

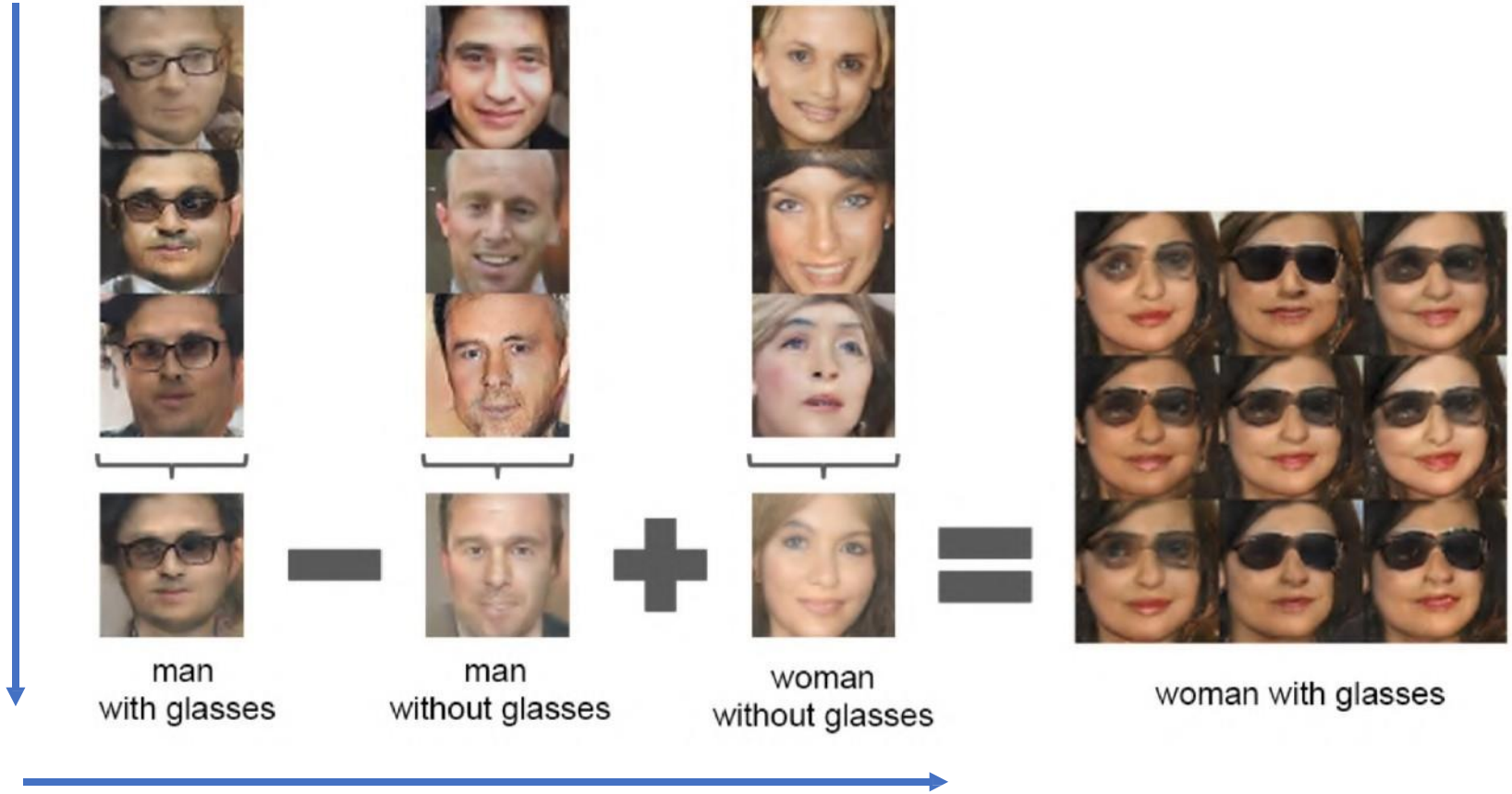
Noise vector $\boldsymbol{z_1}$ — Interpolation: $(\mathbf{1} - \alpha)\ \boldsymbol{z_1} + \alpha \boldsymbol{z_2}$ — Noise vector $\boldsymbol{z_2}$

$\alpha = 0$ $\alpha = 0.5$ $\alpha = 1$



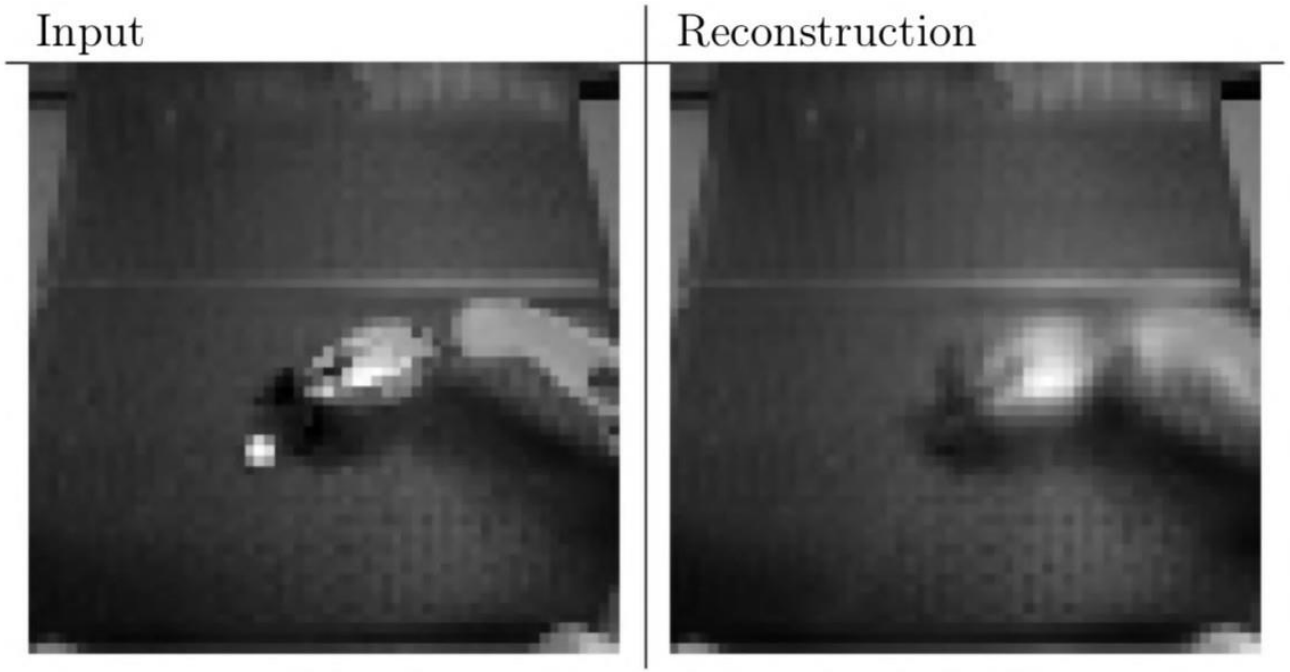
Deep convolutional GAN

- Latent Representation



Adversarial Loss vs. MSE

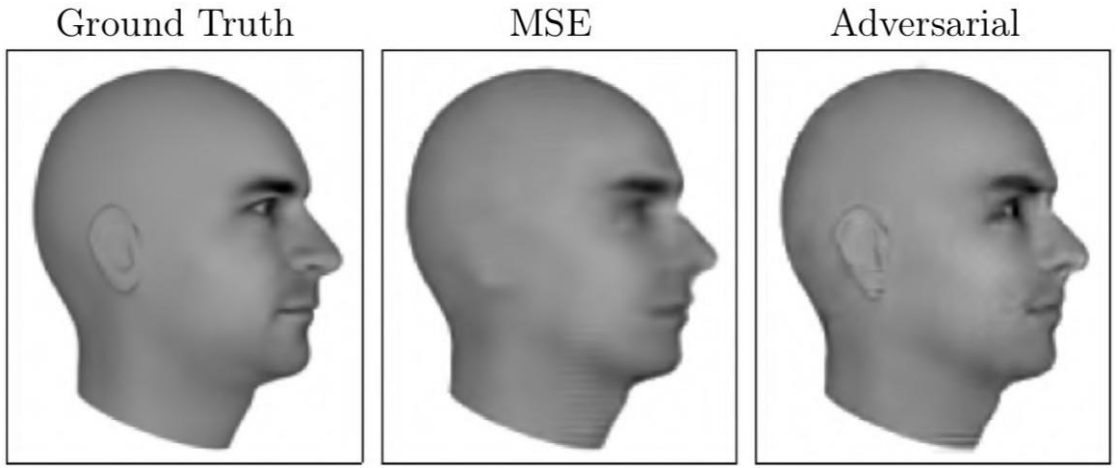
- Limitation of mean squared error



An autoencoder trained with mean squared error for a robotics task has failed to reconstruct a ping pong ball.

Ian Goodfellow, "deep learning"

Adversarial Loss vs. MSE



- (Center) Image produced by a predictive generative network trained with mean squared error alone. Because the ears do not cause an extreme difference in brightness compared to the neighboring skin, they were not sufficiently salient for the model to learn to represent them.
- (Right) Image produced by a model trained with a combination of mean squared error and adversarial loss. Using this learned cost function, the ears are salient because they follow a predictable pattern.

Ian Goodfellow, "deep learning"

Adversarial Loss vs. MSE

- Some impressive samples of GAN, compared with other generative models:

Style-GAN 2019

DFC-VAE

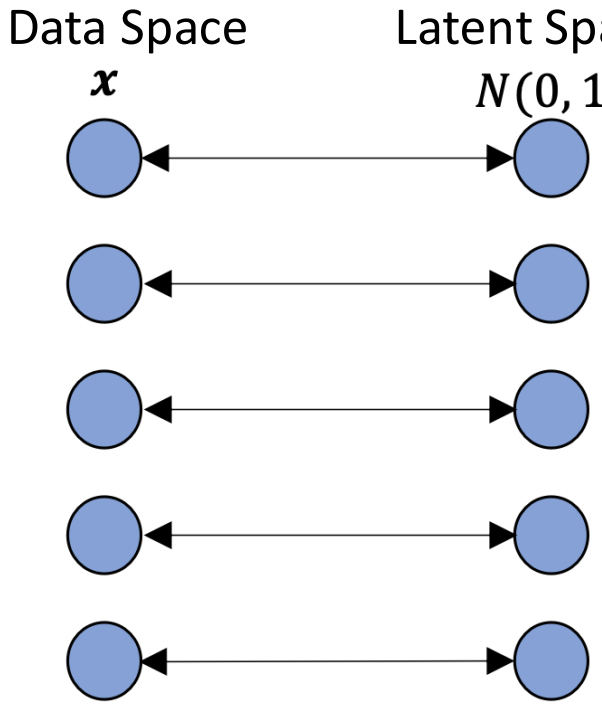
Challenges of GAN

- Density Estimation?
  - Autoregressive Models: $p(\mathbf{x}) = \prod_{i=1}^{n} p_\theta(x_i|x_{<i})$
  - Variational Autoencoders: $p(\mathbf{x}) = \int p(\mathbf{x}|\mathbf{z})p(\mathbf{z})d\mathbf{z}$
  - Normalizing Flow Models: $p(\mathbf{x}) = \pi(\mathbf{z})\left|\det\frac{dz}{dx}\right| = \pi(\mathbf{z})\left|\det(J_{f^{-1}})\right|$

  ➢Then we can estimate the density function of $p(\mathbf{x})$ which can be used for anomaly detection.
- GAN cannot estimate the density function of $p\ x$, so GAN is called implicit generative model. AR, VAE, Flow are called explicit generative models

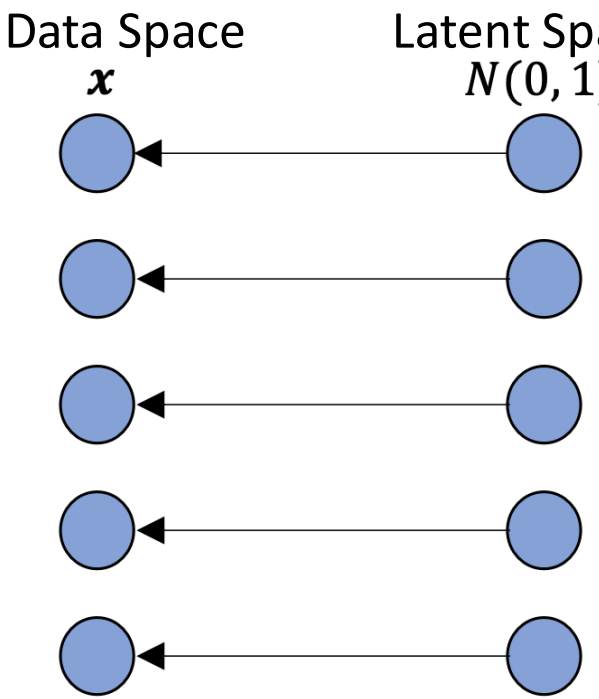
Challenges of GAN

- Inferring Latent Representation?



Bidirectional Mapping

VAE



Unidirectional Mapping

GAN

Training Problems in GANs

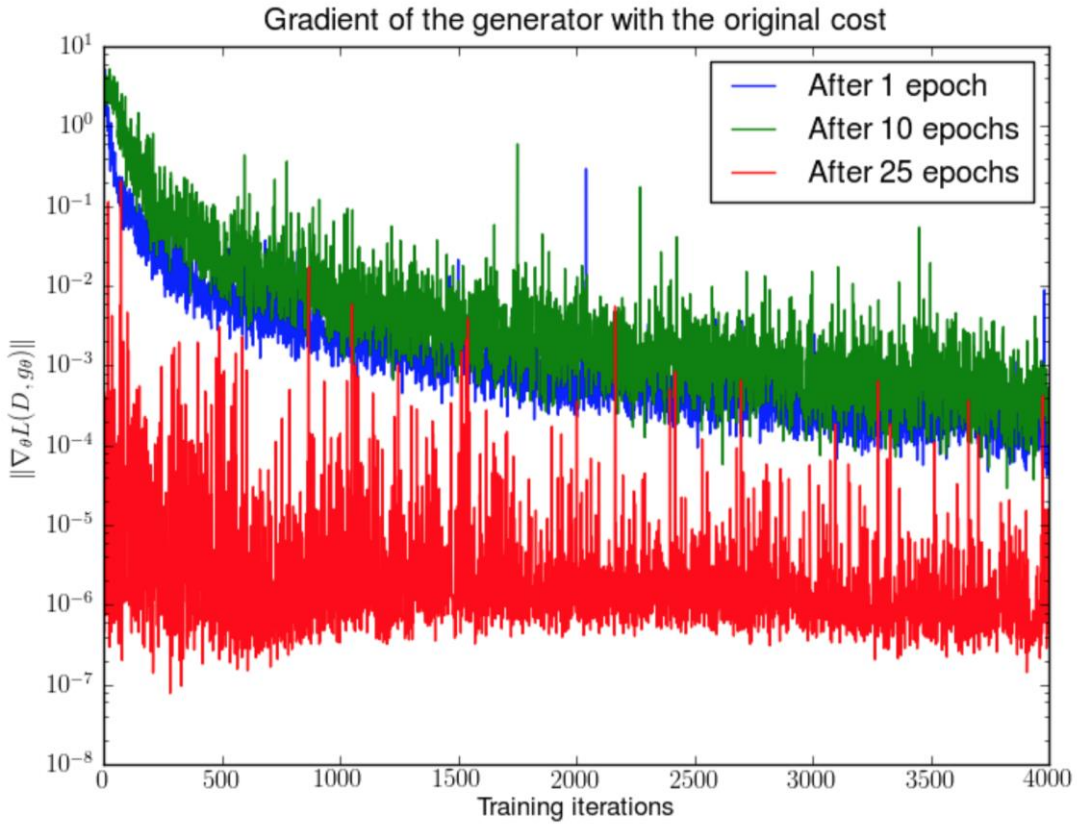
- **Mode collapse**

A DCGAN model is trained with an MLP network with 4 layers, 512 units and ReLU activation function, configured to lack a strong inductive bias for image generation. The results shows a significant degree of mode collapse.

Image source: Arjovsky, Chintala, & Bottou, 2017.)

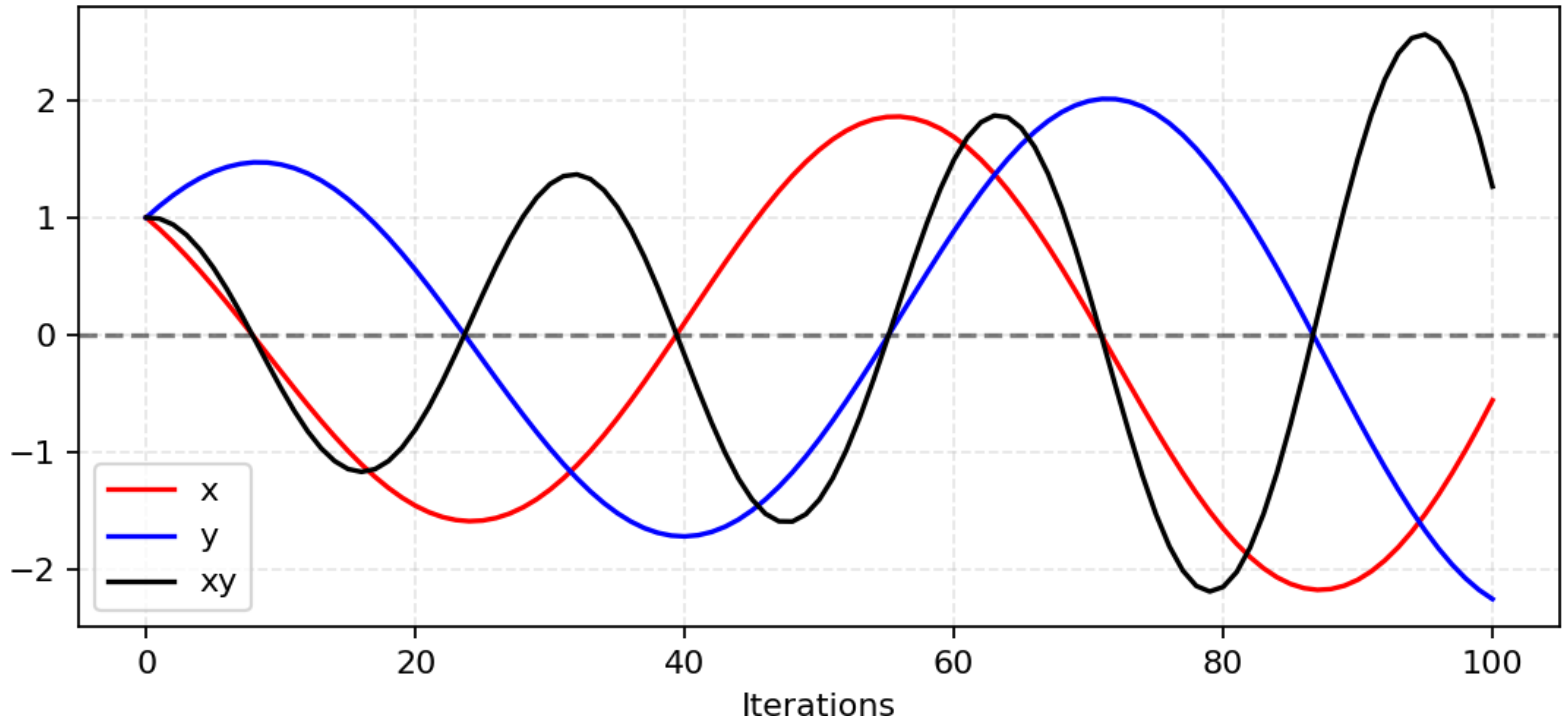
Training Problems in GANs

- **Vanishing gradient**



Training Problems in GANs

- Hard to achieve Nash equilibrium



Next

- Fundamental Problems of GANs
- Wasserstein GAN (WGAN)
- Selected GANs
  - Conditional GAN
  - CycleGAN
  - DualGAN
  - and more

CASE SCHOOL OF ENGINEERING
CASE WESTERN RESERVE UNIVERSITY

Thank You

- Questions?

- Email: yu.yin@case.edu

Reference slides

- Hao Dong. Deep Generative Models
- https://lilianweng.github.io/posts/2018-10-13-flow-models/