Vanilla
Generative Adversarial Network

Hao Dong

Peking University

Vanilla
Generative Adversarial Network

Hao Dong

Peking University

Where we are?

- Product Rule

  $p(x_1, x_2, \dots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots p(x_n|x_1,\dots, x_{n-1})$

- Autoregressive Models

  $p(x_1, x_2, \dots, x_n) = p(x_1)p(x_2|x_1)p(x_3|x_1, x_2) \dots p(x_n|x_1,\dots, x_{n-1})$

- Variational Autoencoders

  $p(X) = \sum_Z p(X|Z)p(Z)$

- Normalising Flow Models

  $$p(x_1, x_2, \dots, x_n) = \pi(\mathbf{z})\left|det\left(J_{f^{1^{-}}}\right)\right| = \pi(\mathbf{z})\left|\frac{1}{det(J_f)}\right|$$

- All the above methods are based on Maximising Likelihoods

- Is the likelihood a good way to measure the distance between TWO distributions?

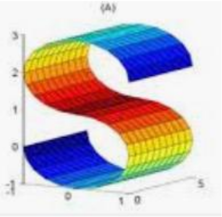
Vanilla Generative Adversarial Network

- Background
    - Distribution, Manifold, Measure
    - Divergences
    - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

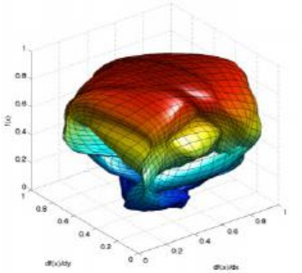
- Background
  - Distribution, Manifold, Measure
  - Divergences
  - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

Distribution & Manifolds

- Manifold: In mathematics, a manifold is a topological space that locally resembles Euclidean space near each point.

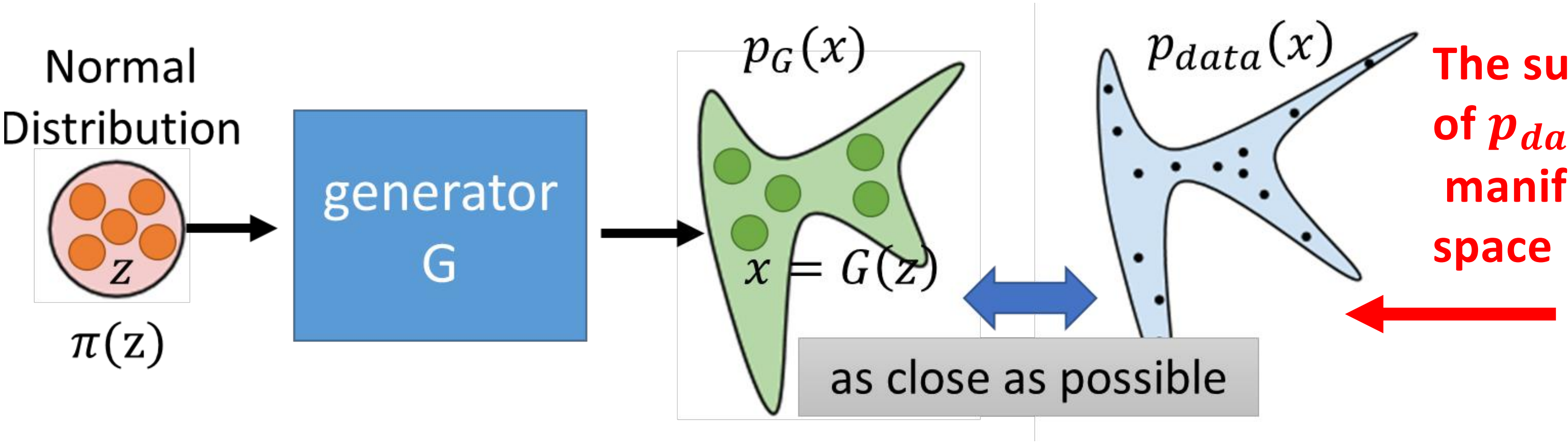


2-D manifold in 3-D space （流形）



3-D manifold in 3-D space （流形）

- Recap: In lecture of flow-based model:



Normal Distribution

z

$\pi(z)$

generator G

$p_G(x)$

$x = G(z)$

$p_{data}(x)$

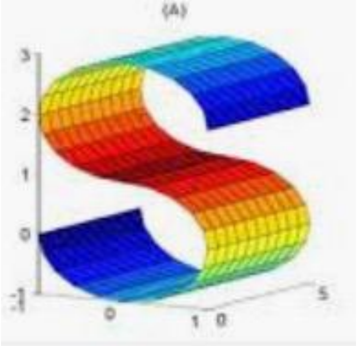
as close as possible

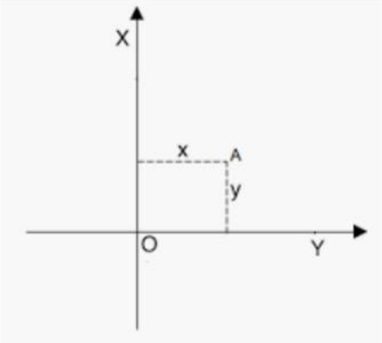
The support set （支撑集） of $p_{data}(x_1, x_2, \dots x_n)$ is a manifold （流形） in n-D space

Measure

- **Measure （测度）** is **an extension of the concept of length, area and volume in high dimensional space**, which can be understood as "Supervolume“

- Examples:
  - 2-d manifold in 3-D space, measure = 0 <-> 2-d surface in 2-d space, measure is its area





  - $S = \{x | x\ is\ rational\ number\ \wedge x\ \in [0,1]\}$, then measure of S in $\mathcal{R}^1$ is 0

- Background
  - Distribution, Manifold, Measure
  - Divergences
  - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

Some Other Divergences

- There're many other estimations that measure "distance" between two distributions.
  - $KL(P||Q) = E_P[\log(\frac{P}{Q})]$
  - $KL_\alpha(P||Q) = KL(P||(\alpha P + (1-\alpha)Q))$
  - $JS(P||Q) = \frac{1}{2}KL(P||\frac{P+Q}{2}) + \frac{1}{2}KL(Q||\frac{P+Q}{2}) = \frac{1}{2}KL_{0.5}(P||Q) + \frac{1}{2}KL_{0.5}(Q||P)$ 0 ~ log2
  - $W(P||Q) = \inf_{\gamma \sim \prod(P,Q)} E_{(x,y)\sim\gamma}[|x-y|]$
  - $D_f(P||Q) = E_P[f(\frac{P}{Q})]$, if $f(t) = -logt, D_f = KL$
- Can we take other divergence as objective?

- Background
  - Distribution, Manifold, Measure
  - Divergences
  - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

Rethinking MLE-based Methods

- MLE-based methods essentially minimise the KL-divergence of P_r and P_g
  - Let P_r and P_g denote real data distribution and generated distribution respectively

    $$\min KL(P_r||P_g) = \min E_{P_r}\left[\log\left(\frac{P_r}{P_g}\right)\right] = \min -E_{P_r}[logP_g] = \max E_{P_r}[logP_g]$$
  - Recall that $KL(P_r||P_g)$ reaches its minimum when $P_r = P_g$, which implies that MLE-based methods essentially minimise the “distance” between P_r and P_g.
  - Autoregressive models and Flow-based models directly optimise the MLE
  - VAE optimises a lower bound of MLE
- Any other “distances”?

- Background
  - Distribution, Manifold, Measure
  - Divergences
  - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

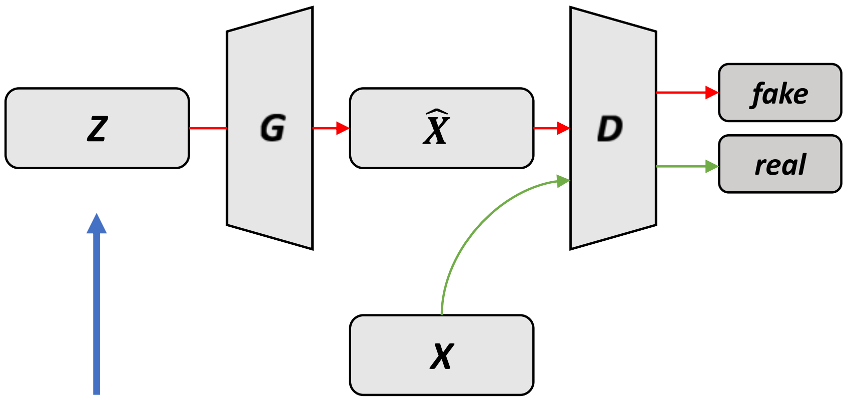
Vanilla GAN - Intro

- Starts from the intuition of min-max game:

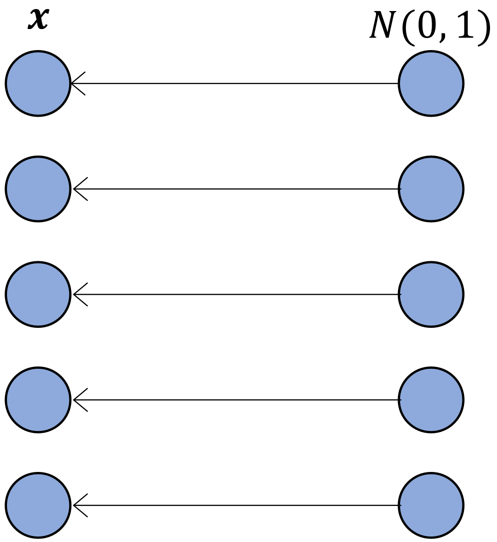
$p(X)$

p_{data}

Vanilla GAN - Intro



Normal/~~uniform~~ distribution



Unidirectional Mapping

Adversarial Learning: map a distribution to another distribution

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

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

Vanilla GAN - Intro

PEKING UNIVERSITY

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

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

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

Why optimising this objective can work?

Vanilla GAN - Theoretical Results

- 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 $D^* = \frac{p_{data}}{p_g + p_{data}} = 0.5 \quad (p_g = \ p_{data})$
  - Then for D^* fixed, $V(G, D) = -log4 + JS(p_{data}||p_g) = -log4 = -2log2 \quad (p_g = \ p_{data})$
  - Thus, when D and G are optimal, $p_{data} = p_g$

Vanilla GAN - Theoretical Results

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

Vanilla GAN - Theoretical Results

- First, we claim that for G fixed

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

$$\begin{aligned} V(G,D) &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x}))dx + \int_z p_{\boldsymbol{z}}(\boldsymbol{z}) \log(1 - D(g(\boldsymbol{z})))dz \\ &= \int_{\boldsymbol{x}} p_{\text{data}}(\boldsymbol{x}) \log(D(\boldsymbol{x})) + p_g(\boldsymbol{x}) \log(1 - D(\boldsymbol{x}))dx \end{aligned}$$

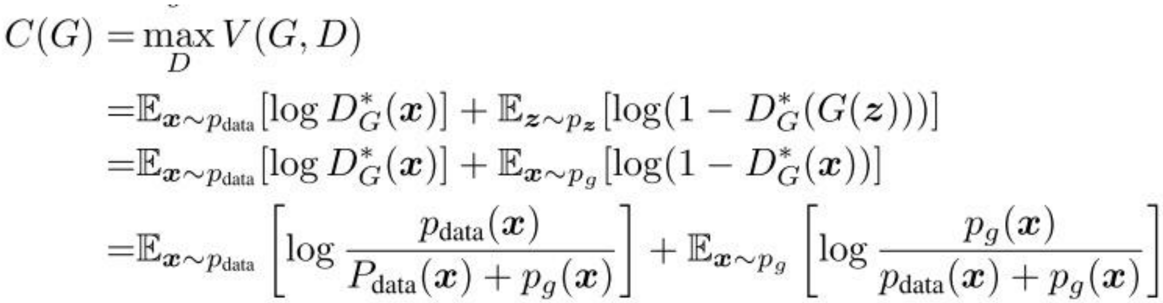
(logA)’ = $\frac{1}{A}$

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

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

Vanilla GAN - Theoretical Results

- Then for D^* fixed



$$
\begin{aligned}
C(G) &= \max_D V(G,D) \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{z}\sim p_{\boldsymbol{z}}}[\log(1 - D_G^*(G(\boldsymbol{z})))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}[\log D_G^*(\boldsymbol{x})] + \mathbb{E}_{\boldsymbol{x}\sim p_g}[\log(1 - D_G^*(\boldsymbol{x}))] \\
&= \mathbb{E}_{\boldsymbol{x}\sim p_{\text{data}}}\left[\log \frac{p_{\text{data}}(\boldsymbol{x})}{P_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right] + \mathbb{E}_{\boldsymbol{x}\sim p_g}\left[\log \frac{p_g(\boldsymbol{x})}{p_{\text{data}}(\boldsymbol{x}) + p_g(\boldsymbol{x})}\right]
\end{aligned}
$$

- Recap $JS(P||Q) = \frac{1}{2}KL(P||\frac{P+Q}{2}) + \frac{1}{2}KL(Q||\frac{P+Q}{2})$

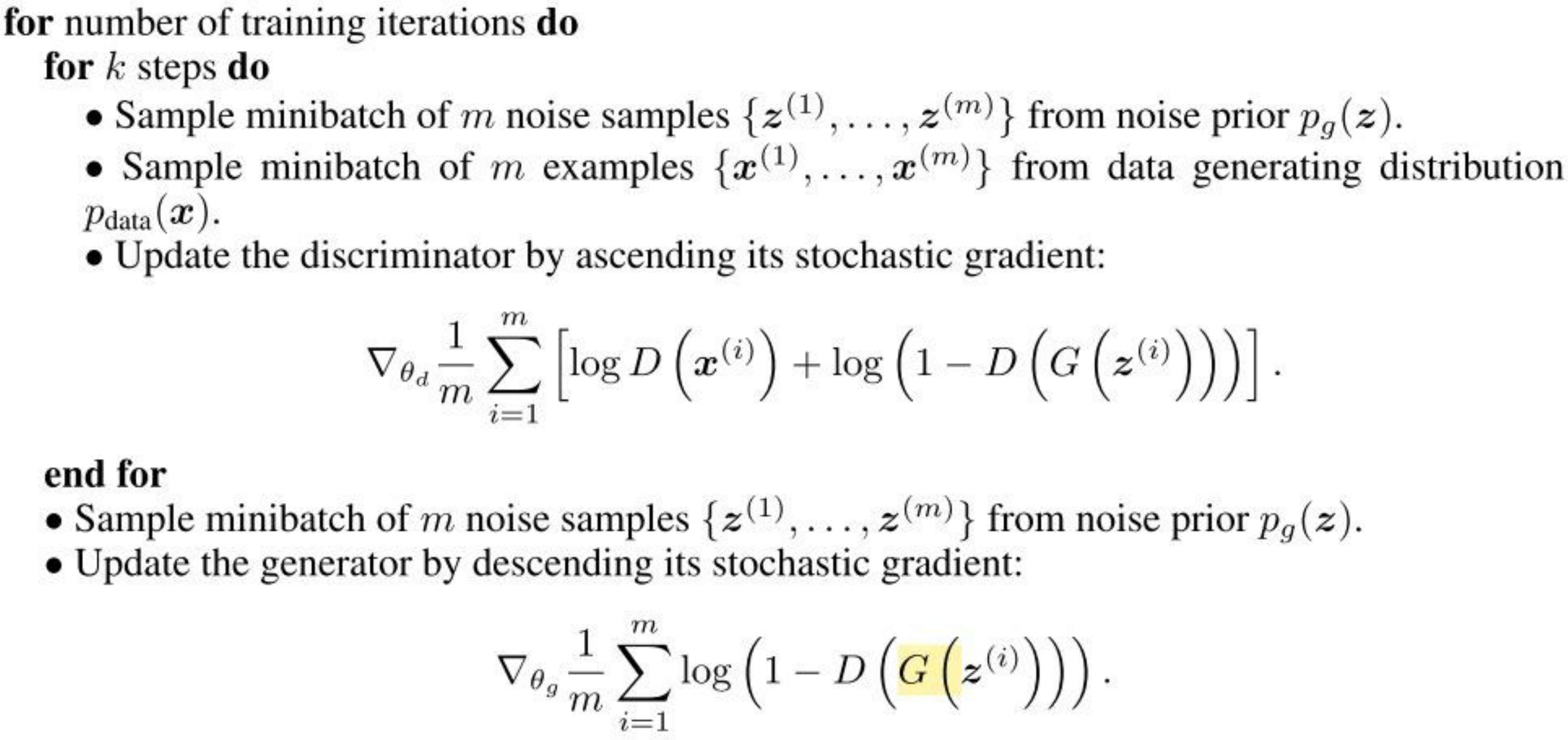
- Then $\boxed{V(G,D) = -log4 + 2 * JS(p_{data}||p_g)}$

0 ~ log2

$$V(G,D) = -2log2 + 2log2 = 0$$

Vanilla GAN - Training Algorithm

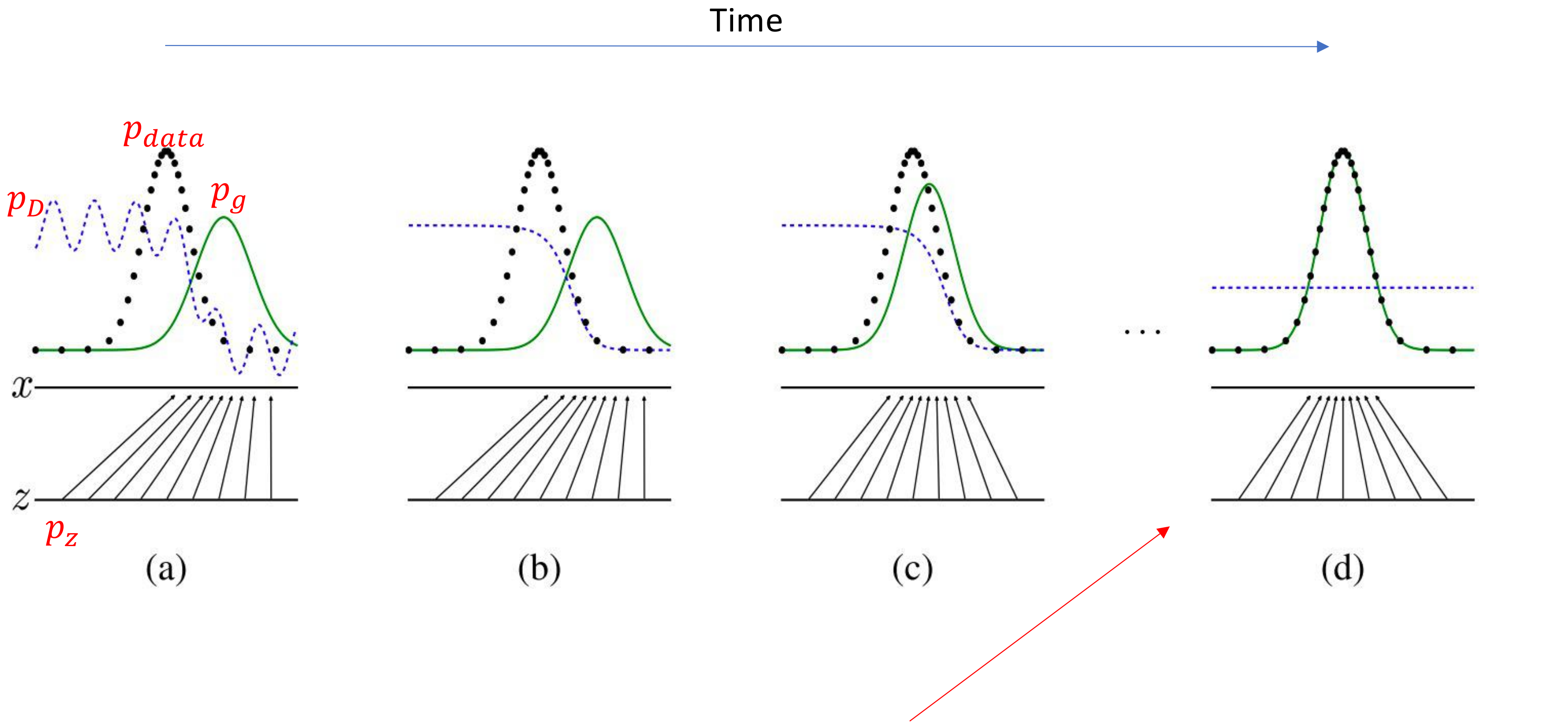
- Loop for: Sample a batch$\{z_i\}\{x_i\}$ -> updating D -> updating G



```
for number of training iterations do
    for $k$ steps do
        • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
        • Sample minibatch of $m$ examples $\{x^{(1)}, \ldots, x^{(m)}\}$ from data generating distribution $p_{\text{data}}(x)$.
        • Update the discriminator by ascending its stochastic gradient:
              $\nabla_{\theta_d} \frac{1}{m} \sum_{i=1}^{m} \left[\log D\left(x^{(i)}\right) + \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right)\right].$
    end for
    • Sample minibatch of $m$ noise samples $\{z^{(1)}, \ldots, z^{(m)}\}$ from noise prior $p_g(z)$.
    • Update the generator by descending its stochastic gradient:
          $\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right).$
```

Why not updating D to its optimum first?

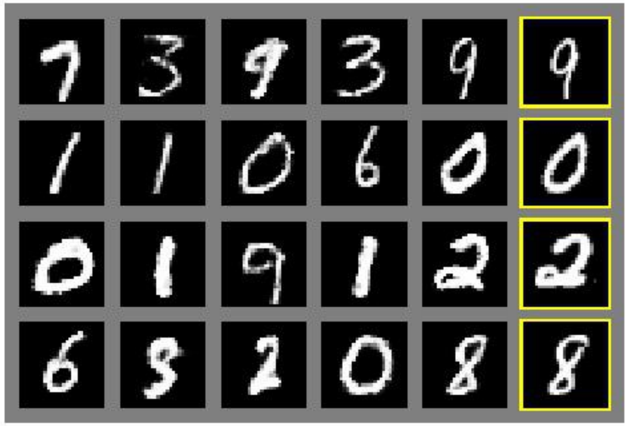
Vanilla GAN - Theoretical Results

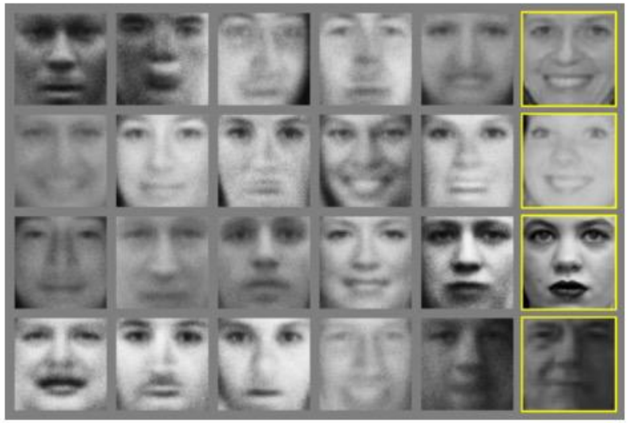


The discriminator is unable to differentiate between the two distributions, i.e. D(x) = 0.5

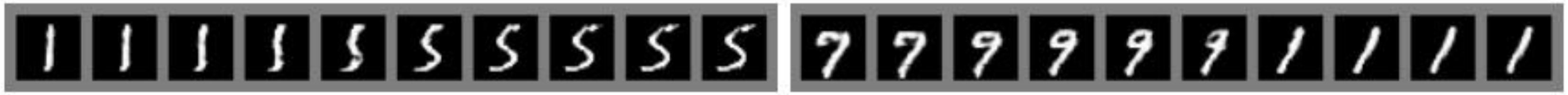
Vanilla GAN - Experiments

- MNIST and TFD
  - Random sample





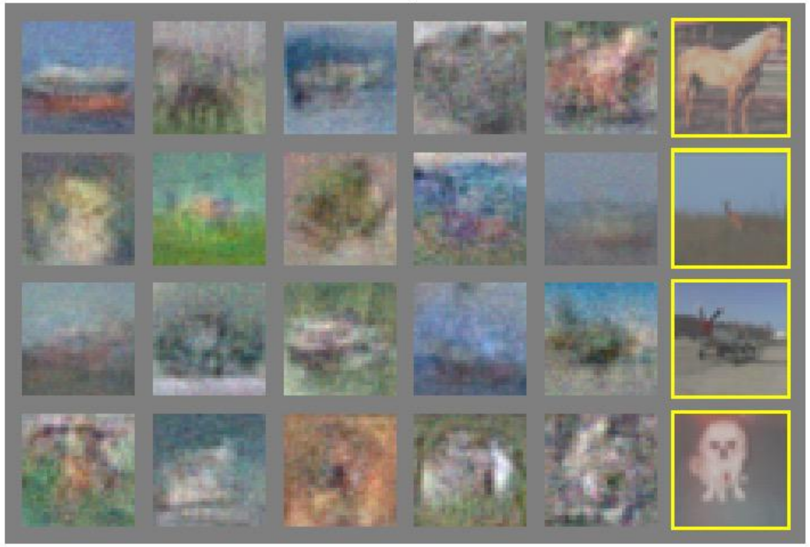
  - Interpolation on Latent Codes



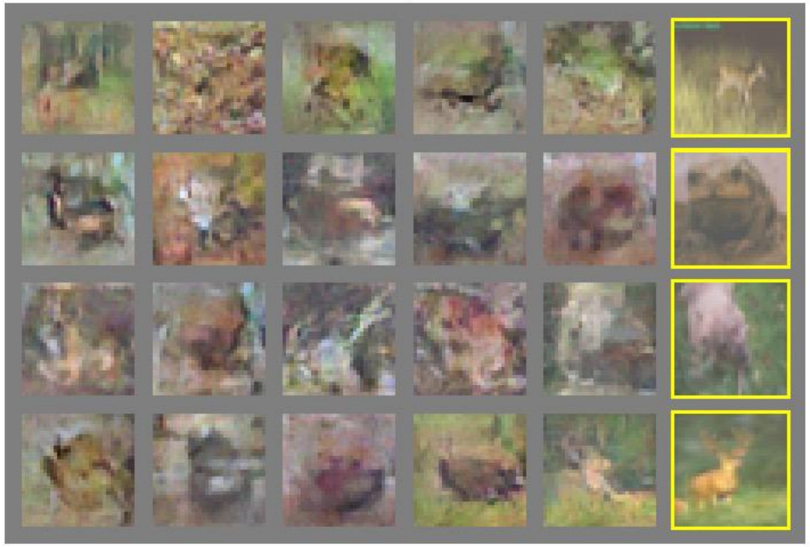
Vanilla GAN - Experiments

- CIFAR10

Fully connected model



Convolutional model



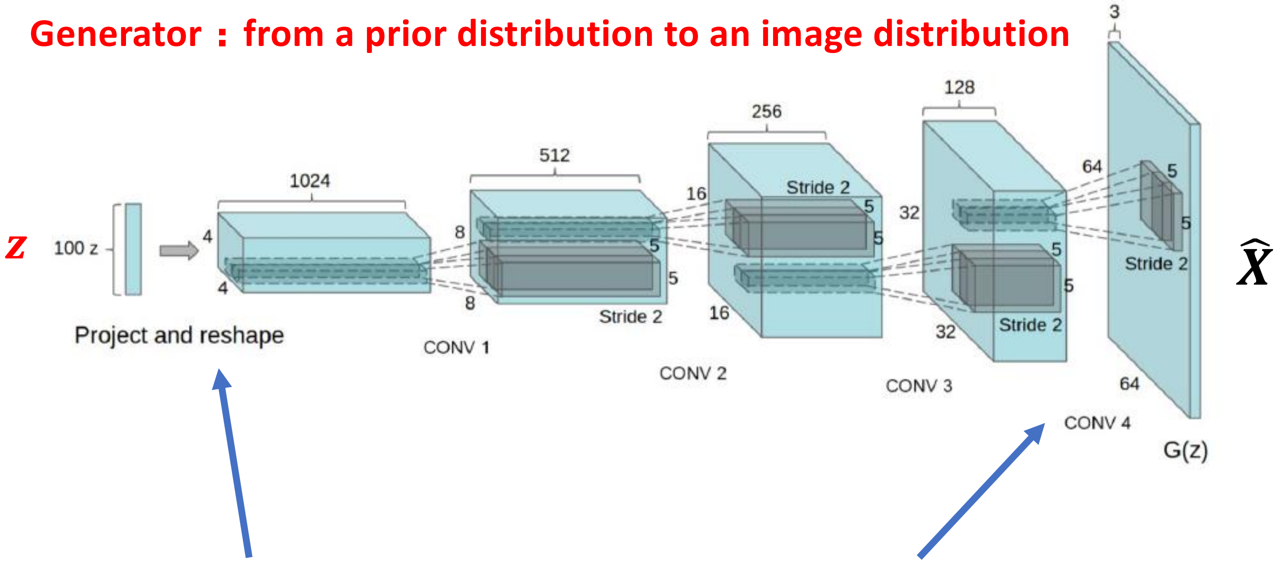
Can we get better performance?

- Background
  - Distribution, Manifold, Measure
  - Divergences
  - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

Deep Convolutional GAN (DCGAN)

- Using the Power of Convolutional Nets

Generator：from a prior distribution to an image distribution



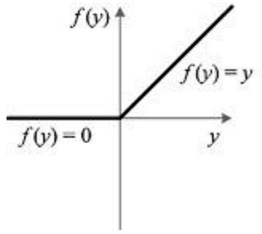
Normal distribution as
the latent distribution
z = 100 values

Data distribution
x = 64x64x3 values

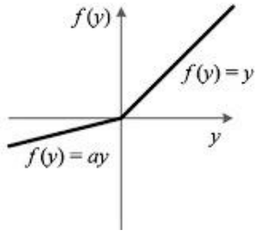
Deep Convolutional GAN (DCGAN)

- GAN is difficult to train: 1. small G gradient if D is good. 2. oscillation

- DCGAN tricks
  - 1. Batch normalisation on all layers except the final layer of G and input layer of D, with a decay of 0.9 (default was 0.99)
  - 2. Adam optimiser with a 1st order momentum (beta1) of 0.5 (default was 0.9)
  - 3. Leaky ReLU with an alpha of 0.2 (default was ReLU)
  - 4. Strided convolutions (default was Maxpooling)
  - 5. Learning rate of 0.0002 (default was 0.0001)



ReLU



Leaky ReLU

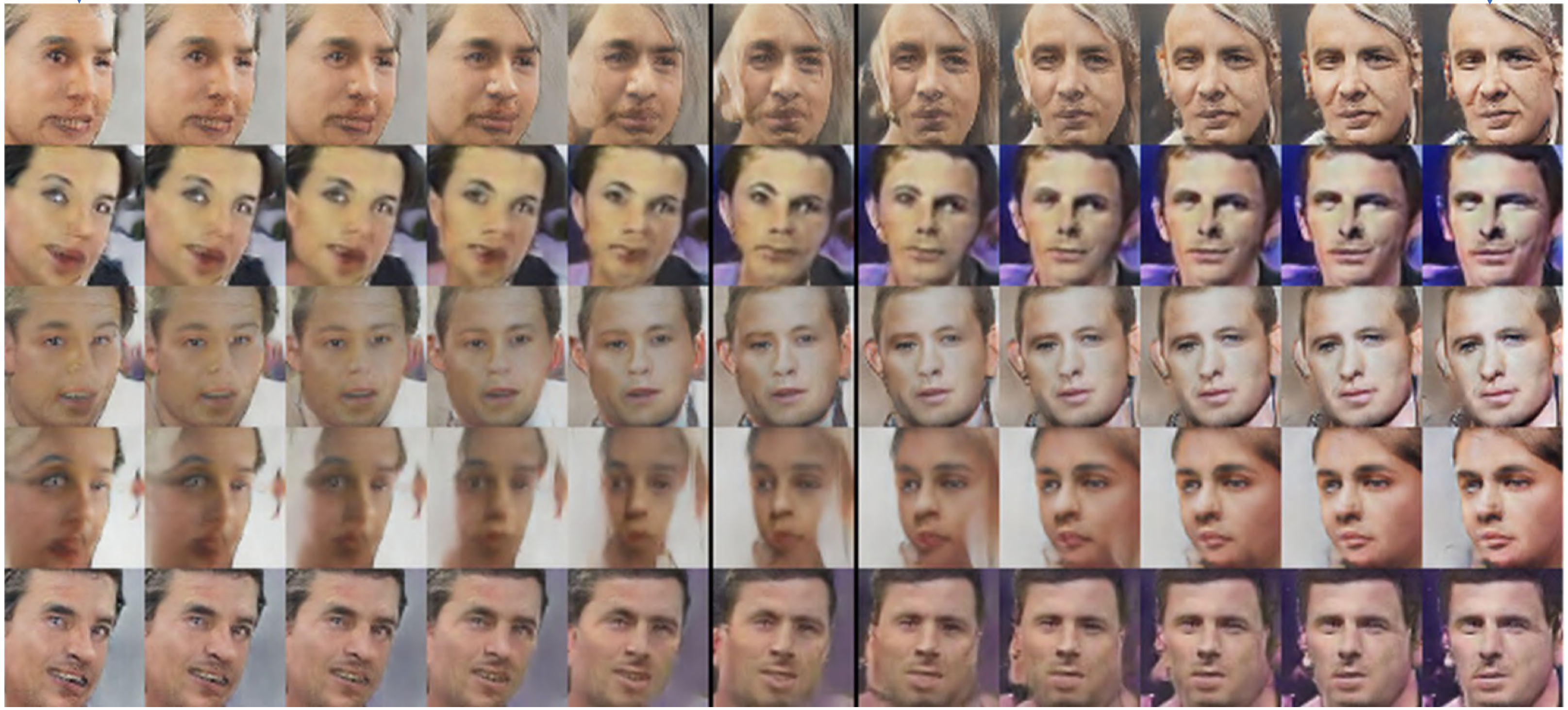
Deep Convolutional GAN (DCGAN)

- Latent Representation

Interpolation: $(\mathbf{1}-\alpha)\boldsymbol{z}_{\mathbf{1}}+\alpha\boldsymbol{z}_{\mathbf{2}}$

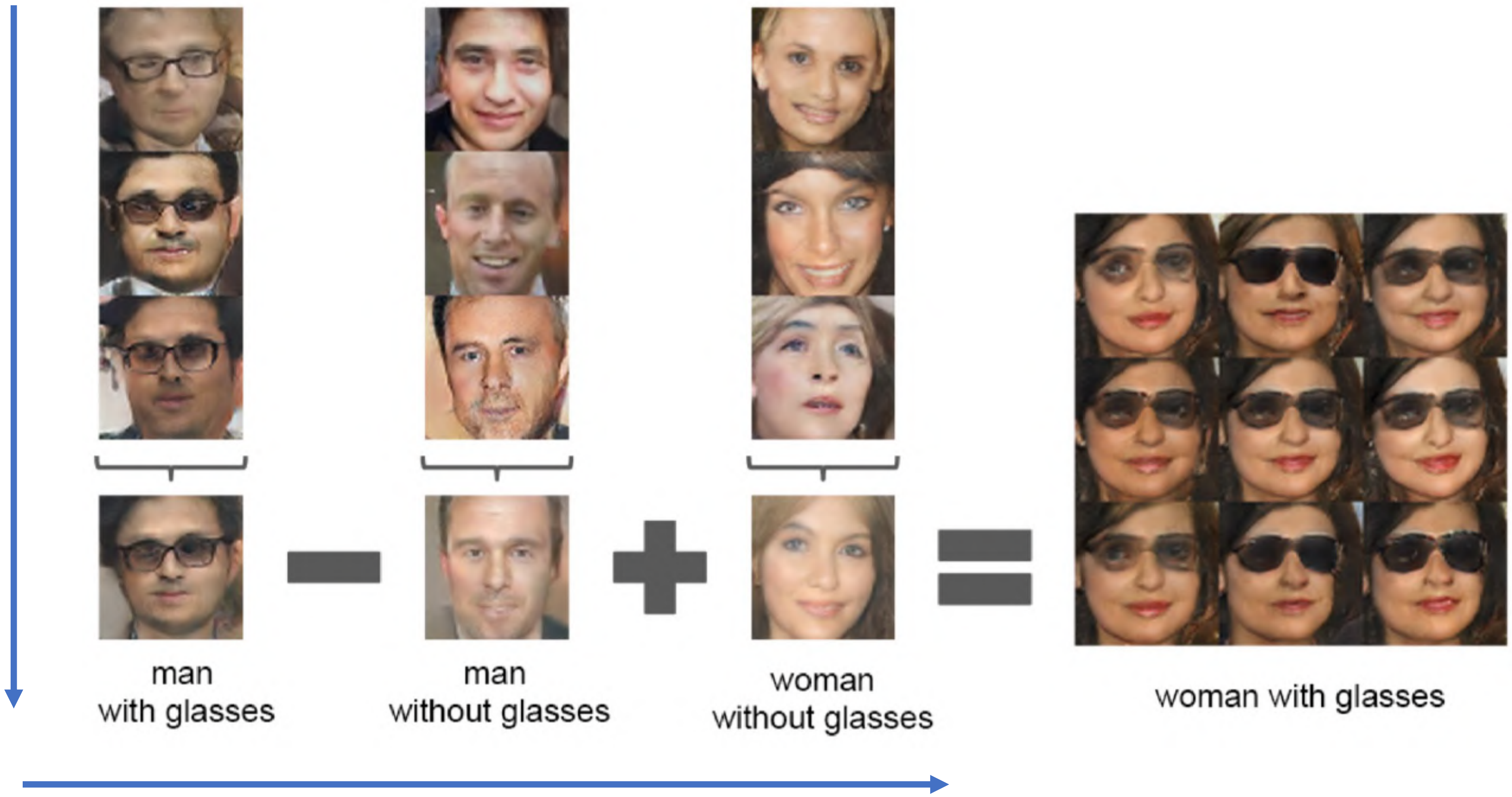
Noise vector $\boldsymbol{z}_{\mathbf{1}}$ → **Noise vector** $\boldsymbol{z}_{\mathbf{2}}$

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



Deep Convolutional GAN (DCGAN)

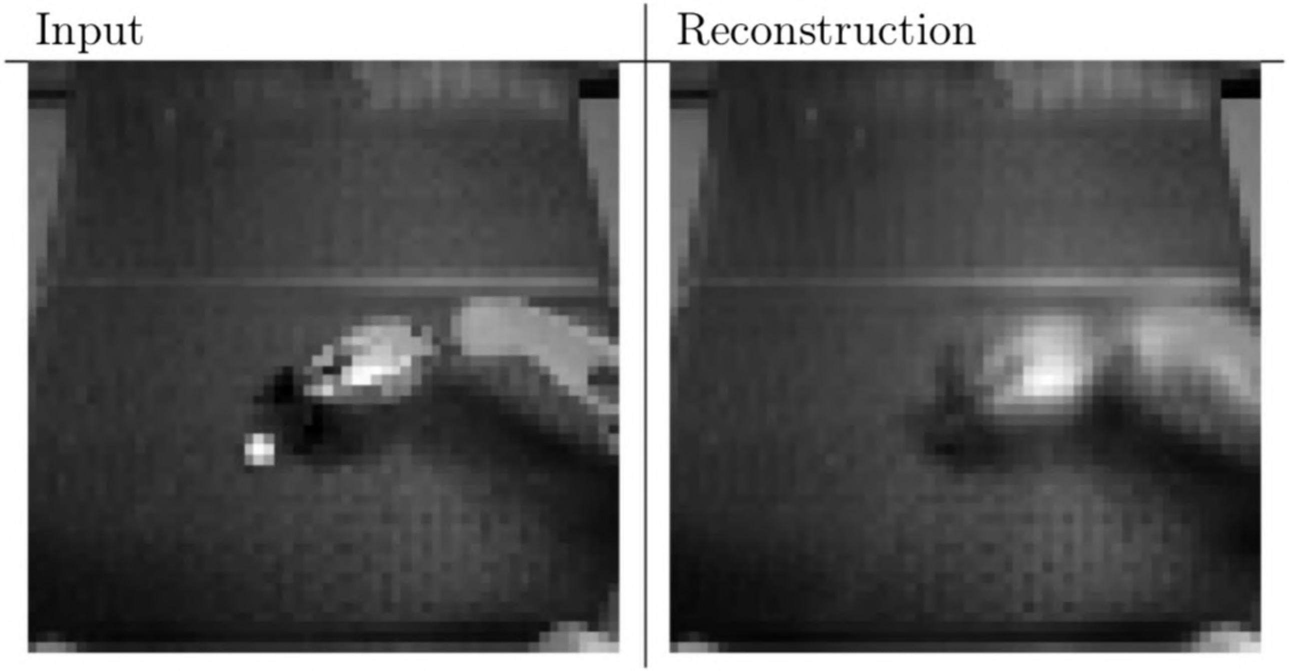
- Latent Representation



- Background
  - Distribution, Manifold, Measure
  - Divergences
  - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

Adversarial Loss vs. MSE

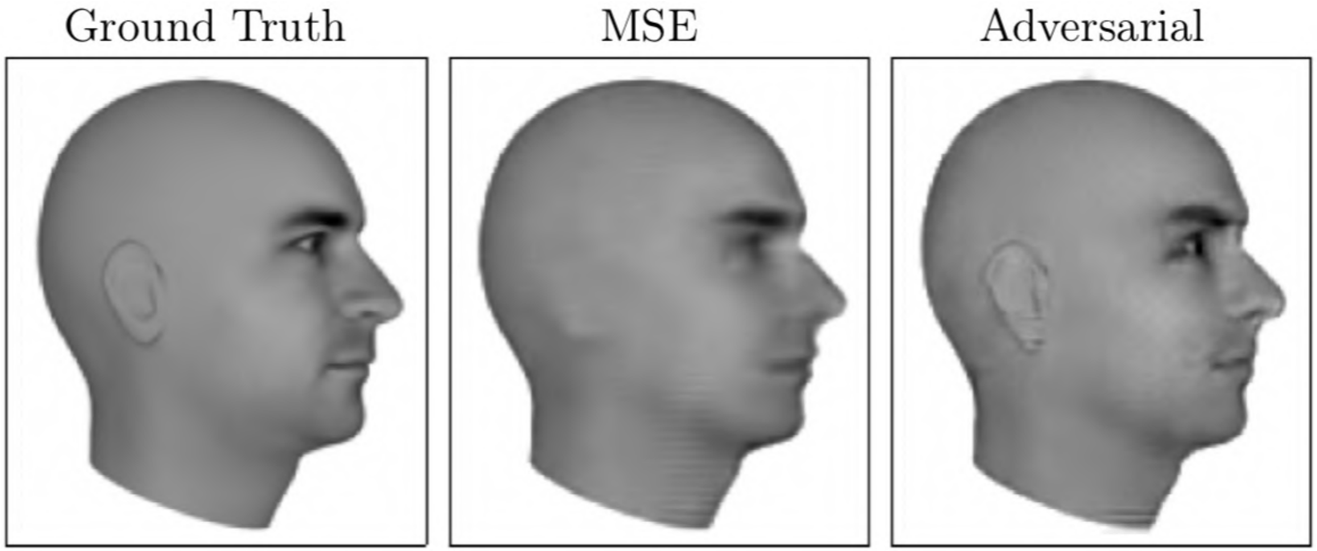
- Limitation of mean squared error



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

example from “deep learning” Ian Goodfellow …

Adversarial Loss vs. MSE



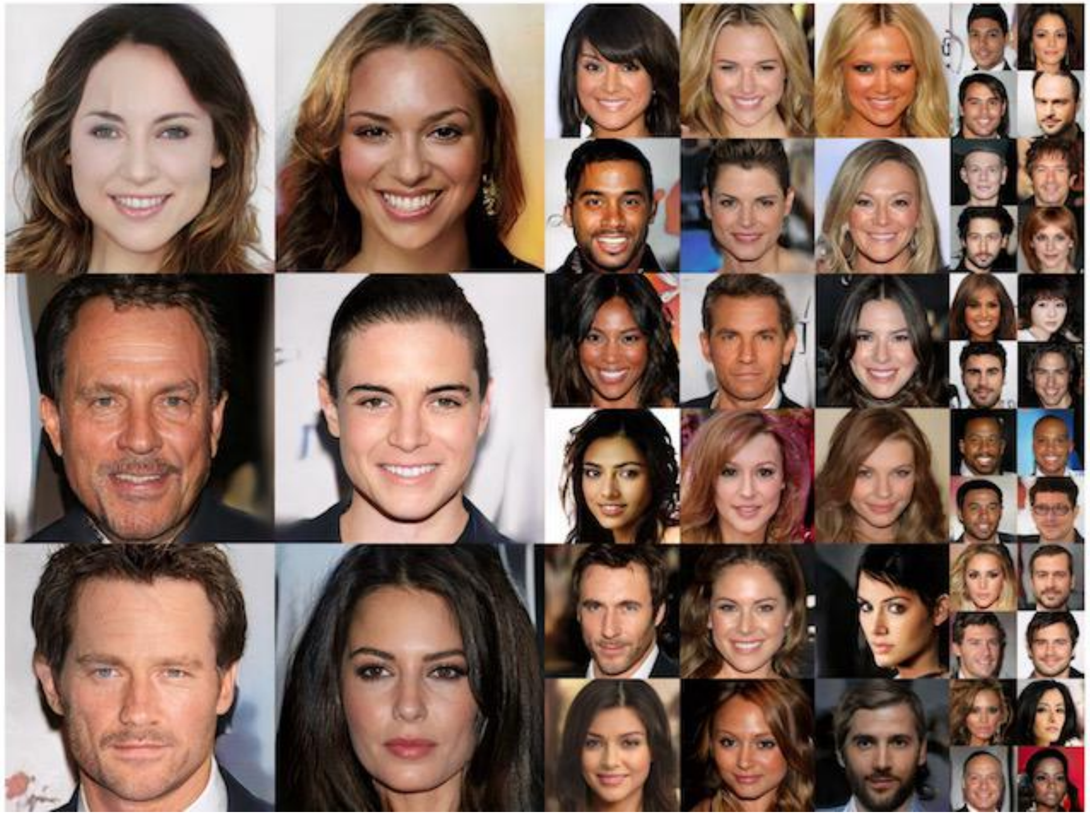
(Center) Image produced by a predictive generative network trained with mean squared error alone. Because the ears do not cause an extreme difference in brightness compared to the neighboring skin, they were not sufficiently salient for the model to learn to represent them.

(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.

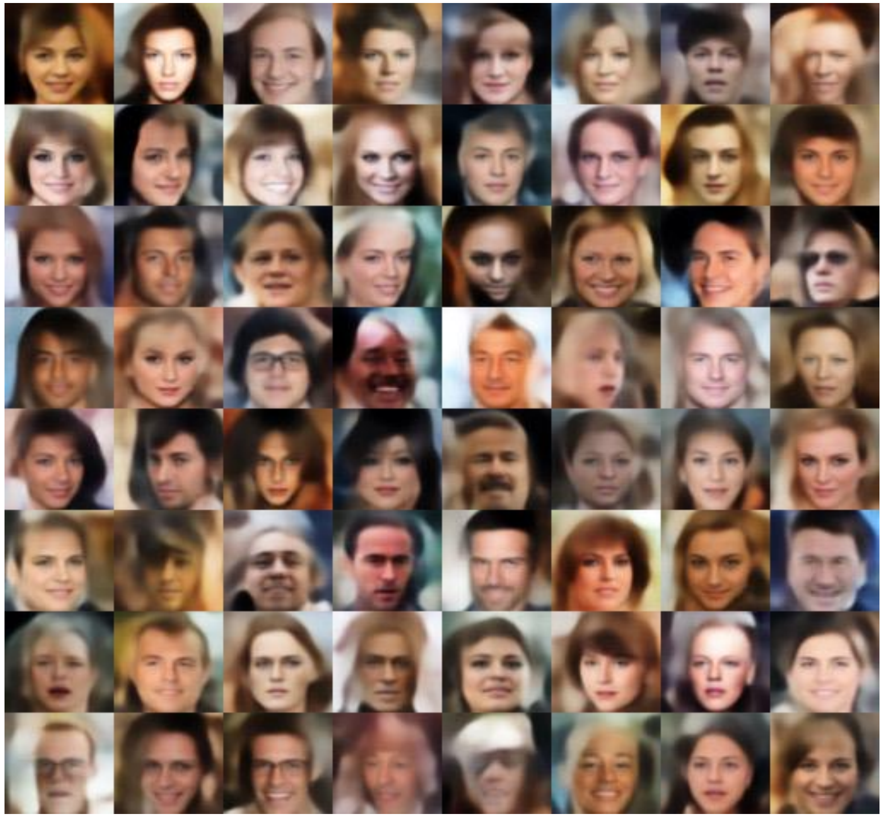
example from “deep learning” Ian Goodfellow …

Adversarial Loss vs. MSE

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



Style-GAN 2019



DFC-VAE

- Background
  - Distribution, Manifold, Measure
  - Divergences
  - Rethinking MLE-based Methods
- GAN: Vanilla GAN
- DCGAN: Deep Convolutional GAN
- Adversarial Loss vs. MSE
- Challenges of GAN

Challenges of GAN

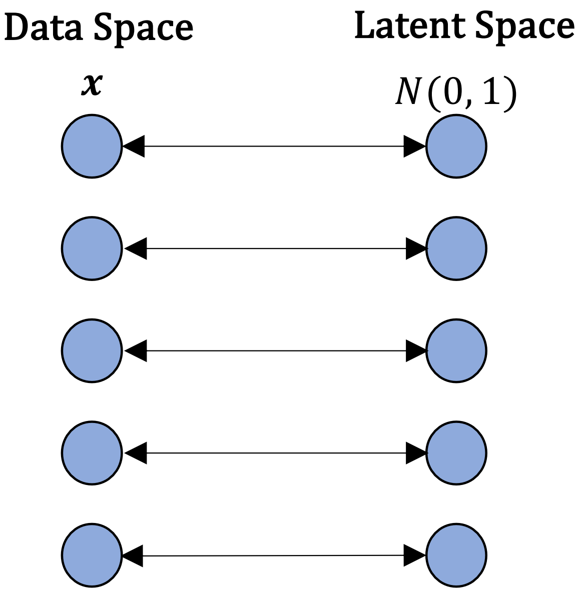
- **Density Estimation**
- Inferring Latent Representation

- Autoregressive Models: $p(x_1, x_2, \ldots, x_n) = p(x_1)p(x_2|x_1) \ldots p(x_n|x_1,\ldots, x_{n-1})$
- VAE: $p(x_1, x_2, \ldots x_n) = E_{z\sim p(z)}[P(x|z)] \approx \frac{1}{K}\sum_{i=1}^{K} P(x|z_i)\,, z_1, z_2 \ldots z_K \sim i.i.d.P(z)$
- Flow:$p(x_1, x_2, \ldots, x_n) = \pi(z)\left|det\left(J_{f^{-1}}\right)\right| = \pi(z)\left|\frac{1}{det(J_f)}\right|$
  - Then we can estimate the density function of $p(x_1, x_2, \ldots x_n)$
    - Application: Detect missing mode
      - If for an instance x', $p(x') \approx 0$, then x' is probably a missing mode

- GAN cannot estimate the density function of $p(x)$, so GAN is called implicit generative model. AR, VAE, Flow are called explicit generative models

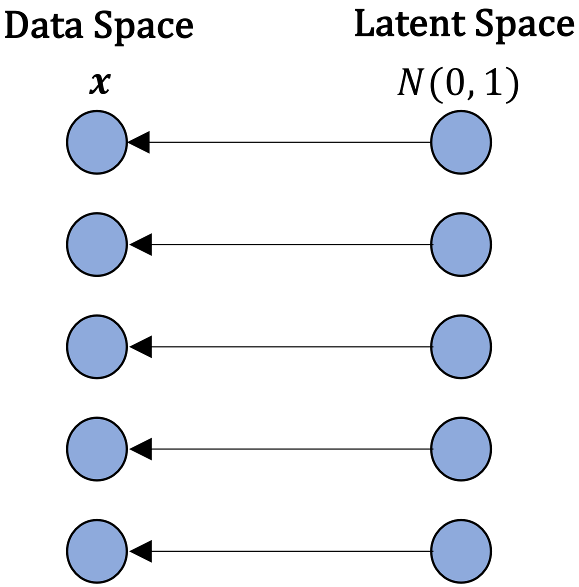
Challenges of GAN

- Density Estimation
- **Inferring Latent Representation**



Bidirectional Mapping

VAE



Unidirectional Mapping

GAN

Summary: Vanilla GAN

- What is support set? What is manifold?
- Why min-max work?
- Why adversarial loss > MSE for image quality?
- Limitations of vanilla GAN?
- Why DCGAN tricks work?
- Compared with Autoregressive Models, VAE and Normalising Flow Models.
- …

Thanks