Generative Adversarial Networks

EE807: Recent Advances in Deep Learning

Lecture 9

Slide made by

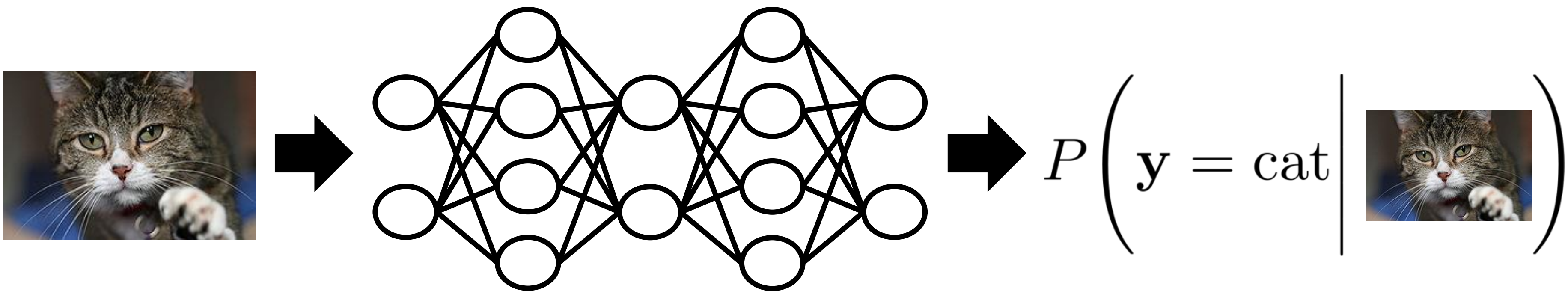
Yunhun Jang, Hyungwon Choi, Sangwoo Mo and Sungsoo Ahn

KAIST EE

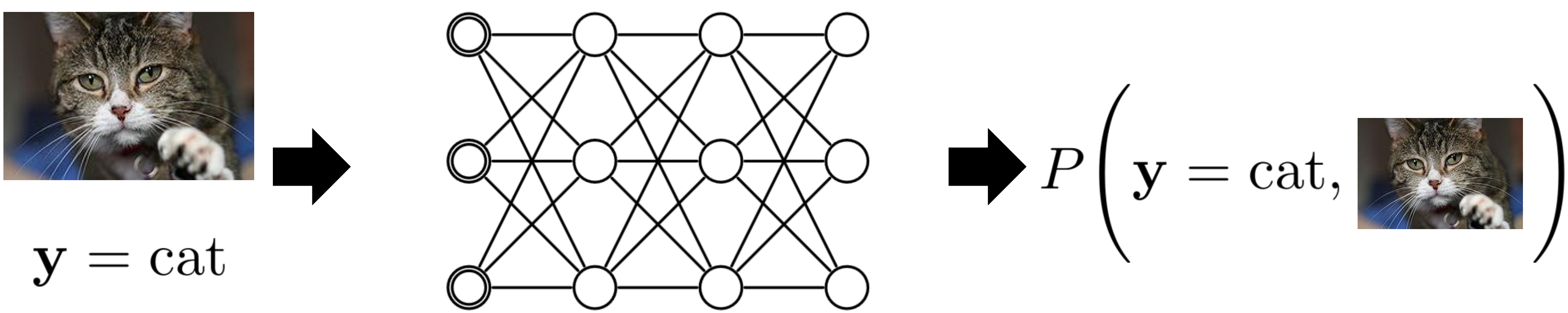
Table of Contents

Recap: Generative Model and Discriminative Model

- Given an observed variable $\mathbf{x}$ and a target variable $\mathbf{y}$
- **Discriminative model** is a model of a conditional distribution $P(\mathbf{y}|\mathbf{x})$
  - e.g., neural networks



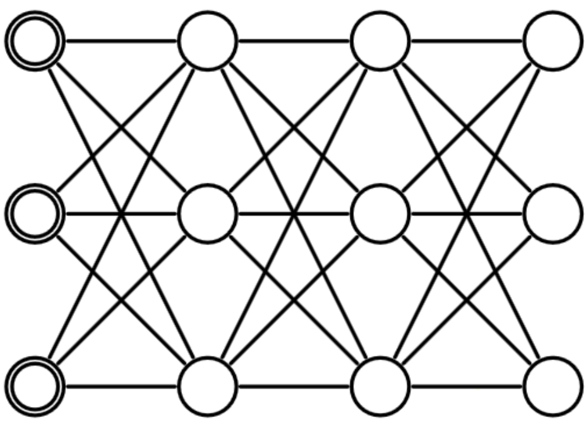
- **Generative model** is a model of a joint distribution $P(\mathbf{x}, \mathbf{y})$ (or $P(\mathbf{x})$)
  - e.g., Boltzmann machines, sum-product networks



* source : https://en.wikipedia.org/wiki/Cat#/media/File:Kittyply_edit1.jpg

- Generative models model a full probability distribution of given data
- $P(\mathbf{x})$ enables us to generate new data similar to existing (training) data
  - This is impossible under discriminative models
- **Sampling methods** are required for generation

$P(\mathbf{x})$



$\sim P(\mathbf{x})$

$\sim P(\mathbf{x})$

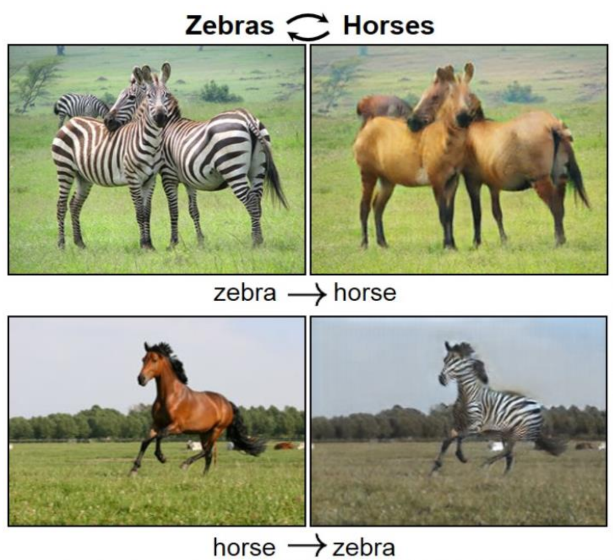
$\sim P(\mathbf{x})$

$\sim P(\mathbf{x})$

* source : https://en.wikipedia.org/wiki/File:Cat_poster_1.jpg

- Generate new samples from the same distribution with training data
- Many real-world applications are related with generating data
- Common applications
  - Vision: super-resolution, style transfer, and image inpainting, etc.
  - Audio: synthesizing audio, speech generation, voice conversion, etc.
  - And many more..

Super-resolution [Ledig, et. al., 2017]



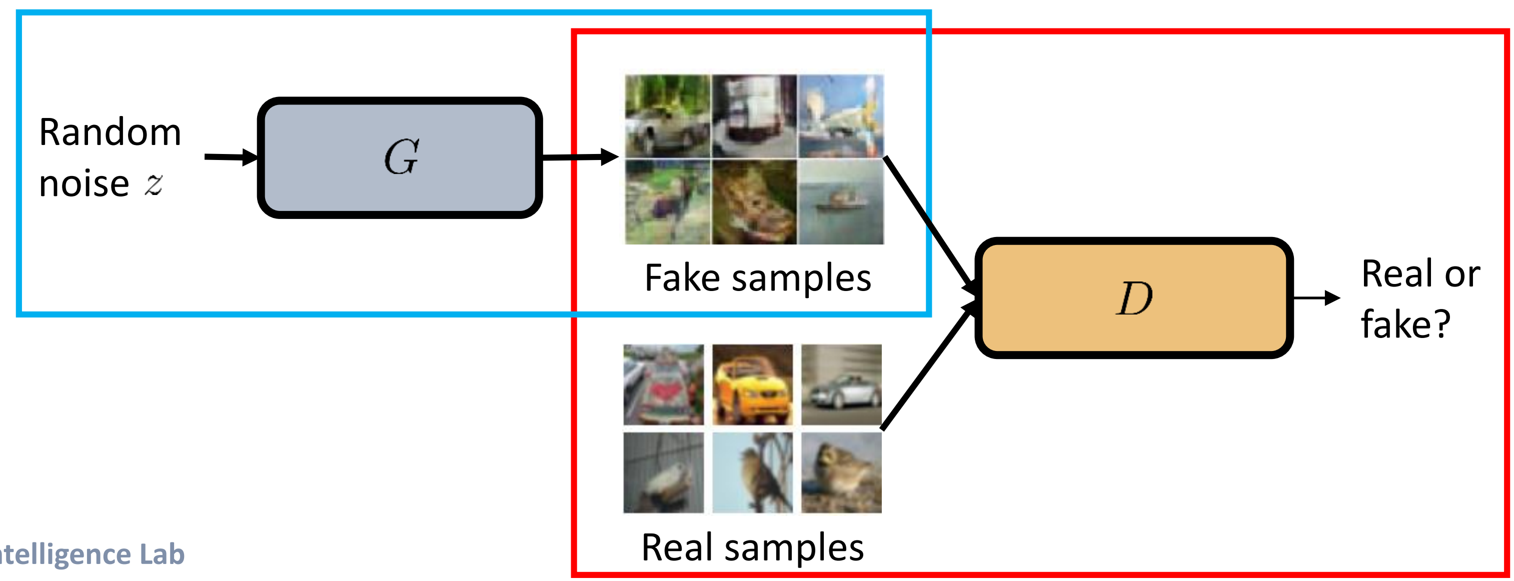
Style transfer [Zhu, et. al., 2017]

High-res image generation [Karras, et. al., 2018]

- Modeling a joint distribution of $\mathbf{x}$ with an explicit probability density function
  - Multivariate Gaussian distributions
    - $P(\mathbf{x}) \propto \exp\left(-\frac{1}{2}(\mathbf{x}-\mu)\Sigma^{-1}(\mathbf{x}-\mu)\right)$
    - Tractable inference, low expressive power
  - Graphical models (e.g., RBM, DBM, etc.)
    - $P(\mathbf{x}) \propto \exp\left(\sum_i b_i x_i + \sum_{i,j} w_{ij} x_i x_j\right)$
    - Intractable inference, high expressive power with compact representations
- Modeling a joint distribution of $\mathbf{x}$ with an implicit density function
  - **Generative adversarial networks (GAN)**
    - Use function approximation capacity of neural networks
    - Modeling the data distribution with implicit density function using neural networks
    - Sampling: simple forward propagation of a generator neural network

Table of Contents

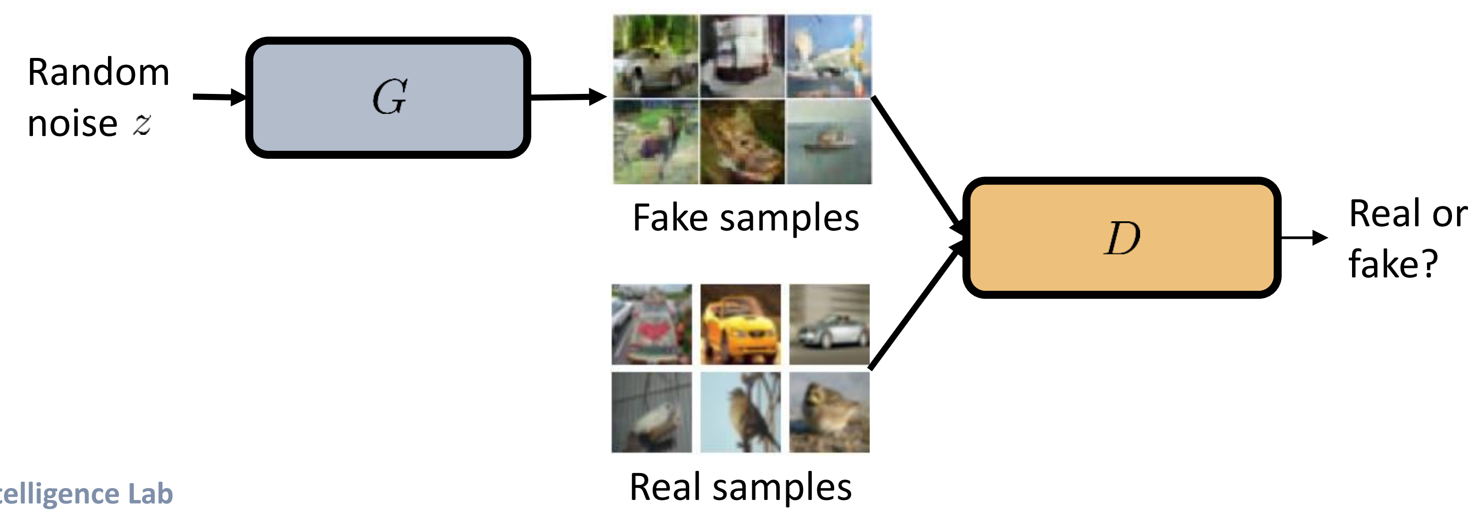
- Many previous approaches (explicit generative models) have difficulties in
  - Sampling from **high-dimensional** and **complex** distributions
  - And make it **realistic**
- Basic idea of GAN [Goodfellow, et. al., 2014]
  - Do not use any explicit density function $p_{\text{model}}(\mathbf{x})$
  - **Two player game** between discriminator network D and generator network G
  - D **tries to distinguish** real data and samples generated by G (fake samples)
  - G **tries to fool** the D by generating real-looking images
  - Utilizes **large capacity of neural nets** to model the sampling function



- D **tries to distinguish** real data and samples generated by G (fake samples)
- G **tries to fool** the D by generating real-looking images
- Objective function:

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

  - For D, maximize objective by making $D(x)$ is close to 1 and $D(G(z))$ is close to 0
  - For G, minimize objective by making $D(G(z))$ is close to 1



- Objective function [Goodfellow, et. al., 2014]:

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

- Alternative training between D and G
  - For D

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

  - For G

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

  - In practice, optimizing generator objective does not work well (details in later slides)

- Discriminator
  - For fixed G, the D optimizes:

$$V(\theta_d, \theta_g) = \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))$$
$$= \int_x p_{\text{data}}(x) \log(D_{\theta_d}(x))dx + \int_z p_z(z) \log(1 - D_{\theta_d}(G_{\theta_g}(z))dz$$
$$= \int_x p_{\text{data}}(x) \log(D_{\theta_d}(x)) + p_g(x) \log(1 - D_{\theta_d}(x))dx$$

  - Optimal discriminator is

$$D_{\theta_d^*}(\mathbf{x}) = \frac{p_{\text{data}}(\mathbf{x})}{p_{\text{data}}(\mathbf{x}) + p_g(\mathbf{x})}$$

  - If $p_{\text{data}} = p_g$, optimal discriminator $D_{\theta_d^*}(\mathbf{x}) = \frac{1}{2}$

What is Optimized in GAN Objective?

- Generator
  - For fixed $D_{\theta_d^*}$, the G optimizes:

$$
\begin{aligned}
V(\theta_d^*, \theta_g) &= \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d^*}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d^*}(G(z))) \\
&= \mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d^*}(x) + \mathbb{E}_{x\sim p_g} \log(1 - D_{\theta_d^*}(x)) \\
&= \mathbb{E}_{x\sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_g(\mathbf{x})}\right] + \mathbb{E}_{x\sim p_g} \left[\log \frac{p_g(x)}{p_{\text{data}}(x) + p_g(x)}\right] \\
&= -\log 4 + KL\left(p_{\text{data}} \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \,\middle\|\, \frac{p_{\text{data}} + p_g}{2}\right) \\
&= -\log 4 + 2 \cdot \boxed{JS(p_{\text{data}} \,\|\, p_g)}
\end{aligned}
$$

- When discriminator is optimal
  - **Generator objective** becomes **minimizing the Jensen-Shannon (JS) divergence**
  - Many previous generative models use KL divergence (maximum likelihood)
  - Unlike KL divergence, JS divergence helps to
    - Generate sharp, clear images but causes a missing mode problem

GAN Training Algorithm: in Practice

- Alternative training of discriminator and generator
  - Recall: G optimizes JS divergence when D is optimal
  - But D is not optimal generally
  - By updating discriminator k-steps per each iteration of generator, this problem could be reduced

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 $\boxed{k \text{ steps}}$ **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.

* source : Goodfellow, et. al., Generative adversarial nets, NIPS 2014

GAN Training Algorithm: in Practice

- Alternative training between D and G
  - For D

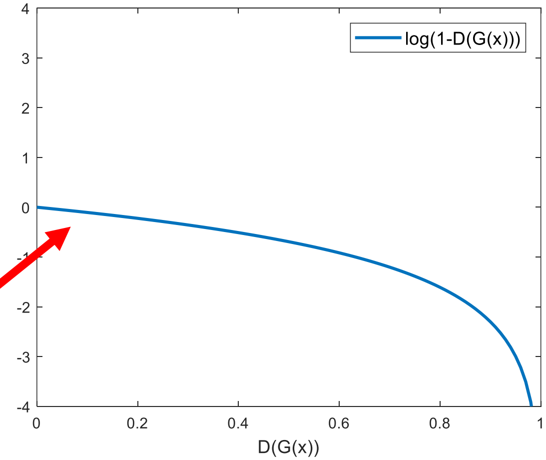
$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

  - For G

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

  - In practice, optimizing generator objective does not work well
  - When generated sample looks **bad** (at the beginning of training) **gradient is relatively flat**
    - Learning by back-prop becomes difficult

Flat gradients when a sample is really bad



* source : http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture12.pdf

- Alternative training between D and G
  - For D

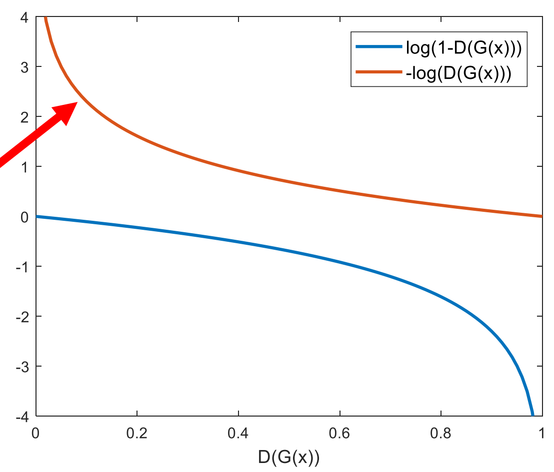
$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{\text{data}}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

  - In practice, G is optimized by

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

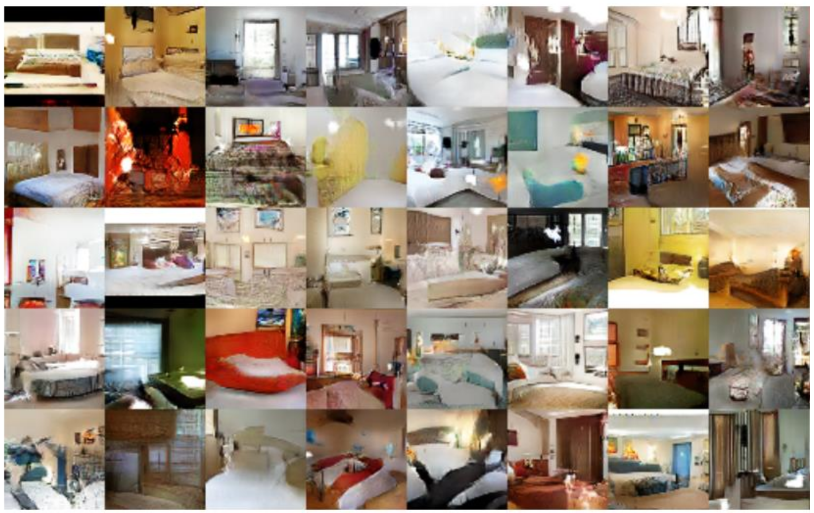
  - $-\log(D_{\theta_d}(G_{\theta_g}(z)))$ gives **stronger gradients** early in learning

Stronger gradients when a sample is really bad



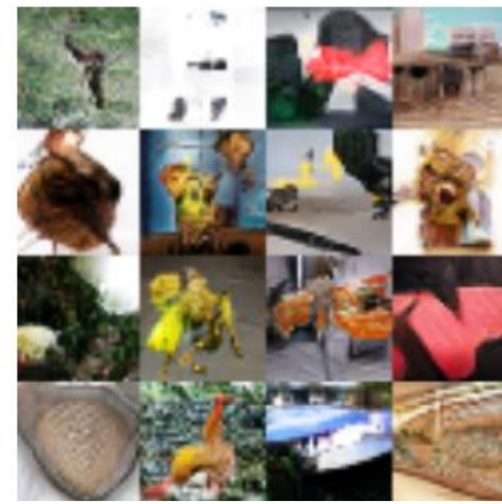
* source : http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture12.pdf

- GAN generates sharp, clear images compared to previous generative models
  - Most previous works are suffered by blurred unrealistic generated samples



Bedroom images

Faces images



ImageNet

- Then, **what makes GAN be able to generate realistic samples**?
  - GAN utilizes the function approximation power of neural networks
    - But it is also the cases for other models (e.g., Variational auto encoder; VAE)
  - What else?

* source : Radford, et. al., Unsupervised representation learning with deep convolutional generative adversarial networks.

Difference with Previous Generative Models

- Maximum likelihood methods (= KL divergence minimization)

$$KL(p_{\text{data}} \parallel p_g) = \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_g(x)} dx$$

  - $p_{\text{data}}(x) > p_g(x)$
    - When $p_{\text{data}}(x) > 0, p_g(x) \to 0$, the integrand grows quickly to infinity
    - **High penalty** when generator's distribution **does not cover parts of the train data**
  - $p_{\text{data}}(x) < p_g(x)$
    - When $p_{\text{data}}(x) \to 0, p_g(x) > 0$, the integrand goes to 0
    - **Low penalty** for generating **fake looking samples**
  - KL divergence solution tends to **cover all the modes**

- Inverse KL divergence $KL(p_g \parallel p_{\text{data}})$ tends to **fit single mode**

Difference with Previous Generative Models

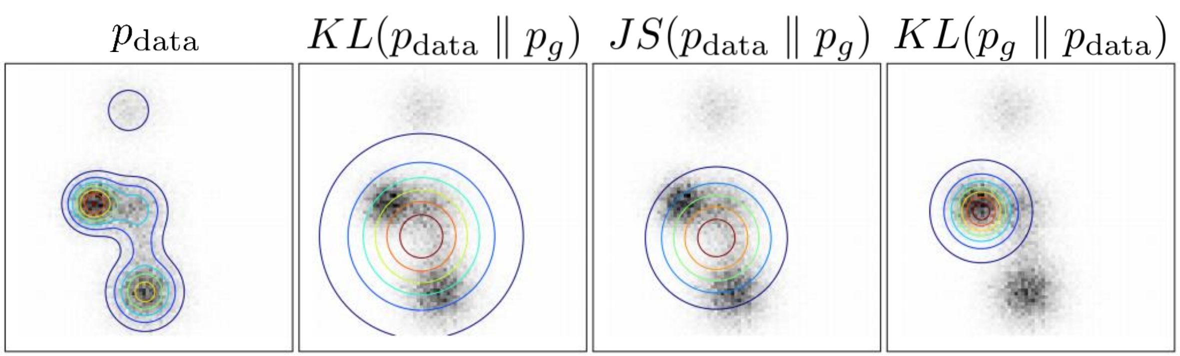
- Maximum likelihood methods (= KL divergence minimization)

$$KL(p_{\text{data}} \parallel p_g) = \int_x p_{\text{data}}(x) \log \frac{p_{\text{data}}(x)}{p_g(x)} dx$$

  - KL divergence solution tends to **cover all the modes**
- Inverse KL divergence $KL(p_g \parallel p_{\text{data}})$ tends to **fit single mode**
- Jensen-Shannon divergence

$$JS(p_{\text{data}} \parallel p_g) = KL\left(p_{\text{data}} \middle\| \frac{p_{\text{data}} + p_g}{2}\right) + KL\left(p_g \middle\| \frac{p_{\text{data}} + p_g}{2}\right)$$

  - (A bit like a) combination of the two divergences
  - Using **JS divergence instead of KL divergence** helps to generate realistic images [Huszar 2015]



 * source : https://www.inference.vc/how-to-train-your-generative-models-why-generative-adversarial-networks-work-so-well-2/

- Hard to achieve Nash equilibrium to a two-player non-cooperative game [Salimans, et. al., 2016]
  - Each model updates its own objective function
  - Modification of θ_d that reduces D's objective can increase G's, and vice versa

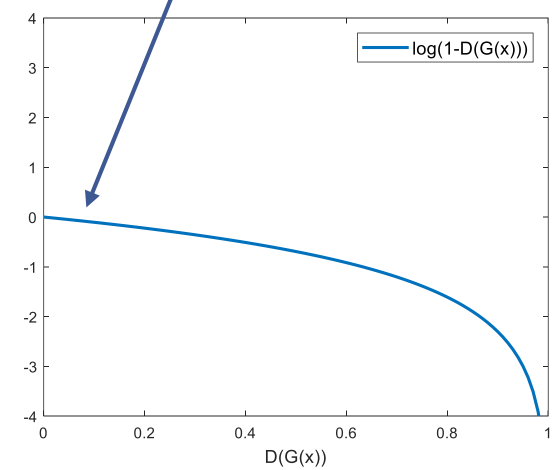
- Mode collapse
  - Generator collapse to parameters that produces the same outputs
  - Generator can fool if it is really good at making only **a** good looking sample
  - JS divergence does not penalize missing mode as hard as KL divergence

Examples of mode collapse in GAN.

* source : Arjovsky, et. al., Wasserstein GAN, ICML 2017

- Vanishing gradients [Arjovsdky and Bottou, 2017]
  - To get accurate feedback from D and to approximate objective of G as JS divergence, D should be trained well
  - However, well-trained discriminator makes gradient of generator **vanished**

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



  - To alleviate vanishing gradients, practical objective is used

$$\nabla_{\theta_g} \mathbb{E}_{z\sim p_z} \left[-\log(D_{\theta_d}(G_{\theta_g}(z))) \right]$$

  - However, it leads objective of generative model into

$$\nabla_{\theta_g} \mathbb{E}_{z\sim p_z} \left[-\log(D_{\theta_d^*}(G_{\theta_g}(z))) \right]$$
$$= \nabla_{\theta_g} \left[KL(p_g \parallel p_{\text{data}}) - 2JS(p_{\text{data}} \parallel p_g) \right]$$

    - **JS divergence has negative sign**: make distribution to be different
    - **Inverse KL term** gives extremely **high cost to generating fake looking samples**, while extremely **low cost on mode dropping**

* source : https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html

Table of Contents

- Minibatch Discrimination
  - Discriminator **looks** at **multiple examples in combination** in minibatch $\{x_1, x_2, \ldots, x_n\}$
    - Rather than a single one, to avoid collapse of the generator

$$M_i = f(x_i)T$$

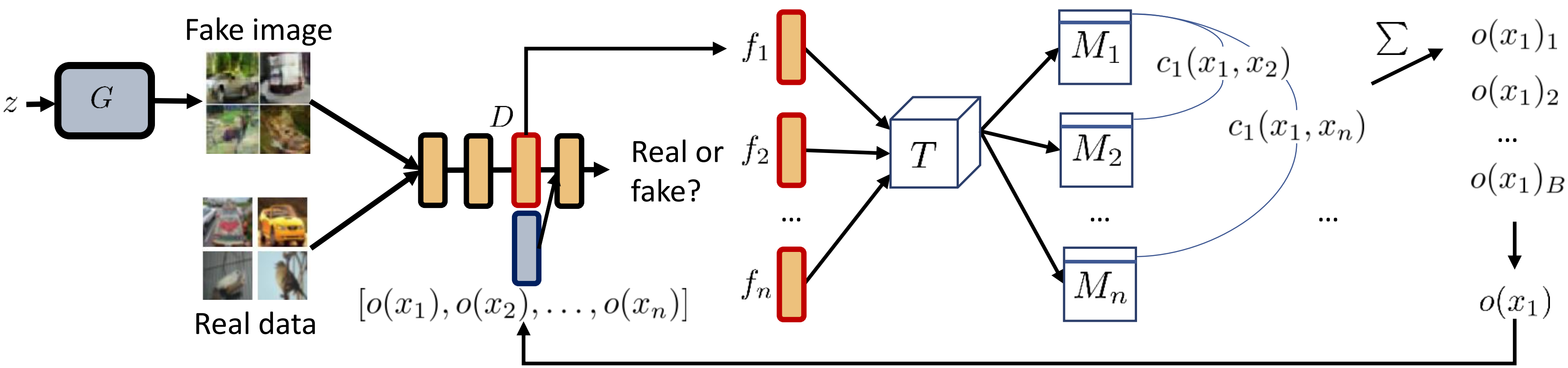
$$c_b(x_i, x_j) = \exp(-\|M_{i,b} - M_{j,b}\|)$$

$$o(x_i)_b = \sum_{j=1}^{n} c_b(x_i, x_j)$$

$$o(x_i) = [o(x_i)_1, o(x_i)_2, \ldots, o(x_i)_B]$$

where $M_{i,b}$ is b-th row of M_i , and $c_b(x_i, x_j)$ measures distance between $f(x_i), f(x_j) \in \mathbb{R}^A$ on some transformation $T \in \mathbb{R}^{A\times B\times C}$

- Concatenate $o(x_i)$ with $f(x_i)$ and use it as an input to the next layer of D
- G should generate samples that has similar statistics with train data samples



* source : Salimans, et. al., Improved techniques for training GANS, NIPS 2016

Improved Techniques to Alleviate Problems in GAN [Salimans, et. al., 2016]

- Feature matching
  - Instead of directly maximizing the output of the D, make G to **generate data that matches features of the real data**
  - Loss of generator becomes:

$$\min_{\theta_g} \mathbb{E}_{z\sim p_z} \log(1 - D_{\theta_d}(G_{\theta_g}(z))) \quad \Rightarrow \quad \min_{\theta_g} \|\mathbb{E}_{x\sim p_{\text{data}}} f(x) - \mathbb{E}_{z\sim p_z} f(G(z))\|$$

  where f is activations of an intermediate layer of D
  - D's loss remains the same with original GAN's discriminator loss
- Historical averaging
  - Add additional loss term $\left\|\theta - \frac{1}{t}\sum_{i=1}^{t}\theta_i\right\|^2$ to **penalize changing θ too fast**
- One-sided label smoothing
  - Instead of providing 0, 1 labels, use soften values (e.g., 0.9, 0.1)
  - Reduce the networks' vulnerability
- Virtual batch-normalization
  - Using fixed batch of data for batch-normalization
  - Reduce high dependency between samples in a minibatch

Wasserstein Distance

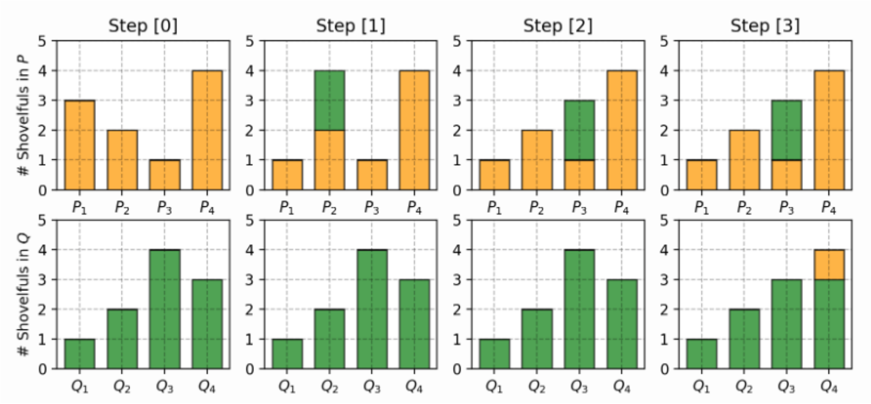
- Some heuristics can alleviate the issue for training GAN
  - But, they are not fundamental solutions and are not clear to work in general

- Wasserstein distance: **measure of the distance between two probability distributions** (also called Earth Mover's distance)

$$W(p_{\text{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|$$

  - Intuitively, minimal total amount of work to transform one *heap of dirt* into the other
  - Work is defined as the amount of *dirt* in a chunk times the distance it was moved
  - Example
    - $W(P,Q)$: the minimum amount of work from distribution P to Q

$$P_1 = 3, P_2 = 2, P_3 = 1, P_4 = 4$$
$$Q_1 = 1, Q_2 = 2, Q_3 = 4, Q_4 = 3$$
$$W(P,Q) = 5$$

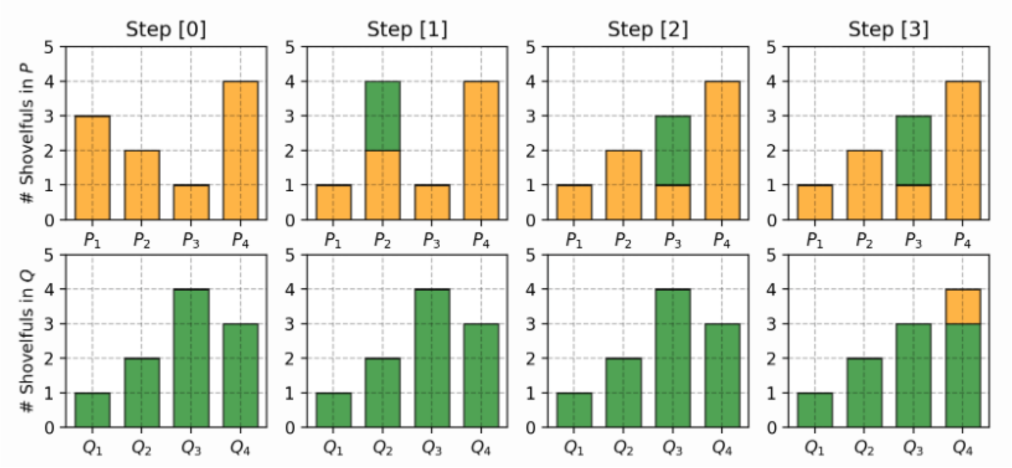


* source : https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#wasserstein-gan-wgan

- Wasserstein distance: **measure of the distance between two probability distributions** (also called Earth Mover's distance)

$$W(p_{\text{data}}, p_g) = \inf_{\gamma \in \Pi(p_{\text{data}}, p_g)} \mathbb{E}_{(x,y)\sim\gamma} \|x - y\|$$

  - Intuitively, minimal total amount of work to transform one *heap of dirt* into the other
  - Work is defined as the amount of *dirt* in a chunk times the distance it was moved
  - Example
    - $\Pi(p_{\text{data}}, p_g)$ is the set of all possible joint probability distributions between p_{data} and p_g
    - Infimum over joint distribution γ (each γ corresponds to one dirt transport plan like in example in a slide before)



* source : https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#wasserstein-gan-wgan

Comparison between Wasserstein Distance and Other Distance Metrics

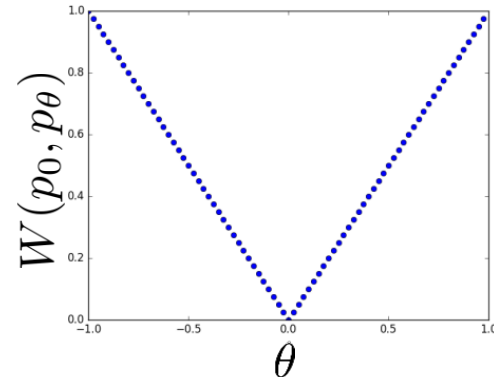
- When two distributions are located without overlaps
  - Still provides meaningful and smooth representation of the distance (and gradients)

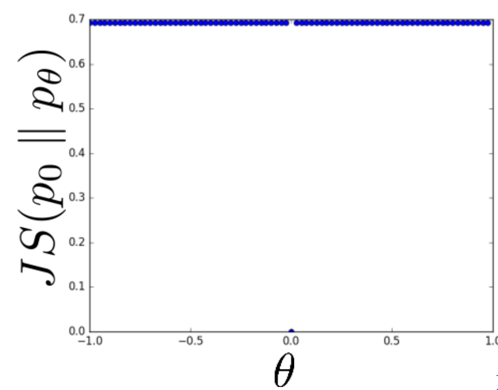
- Example [Arjovsky, et. al., 2017]
  - Let $Z \sim U[0,1]$, p_0 be the distribution of $(0, Z) \in \mathbb{R}^2$
  - $g_\theta(Z) = (\theta, Z)$ with θ, a single real parameter, and p_θ is the distribution of $g_\theta(Z)$
  - Distance between two distributions are:

$$W(p_0, p_\theta) = |\theta|$$

$$JS(p_0 \parallel p_\theta) = \begin{cases} \log 2 & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0 \end{cases}$$

$$KL(p_0 \parallel p_\theta) = KL(p_\theta \parallel p_0) = \begin{cases} \infty & \text{if } \theta \neq 0, \\ 0 & \text{if } \theta = 0 \end{cases}$$

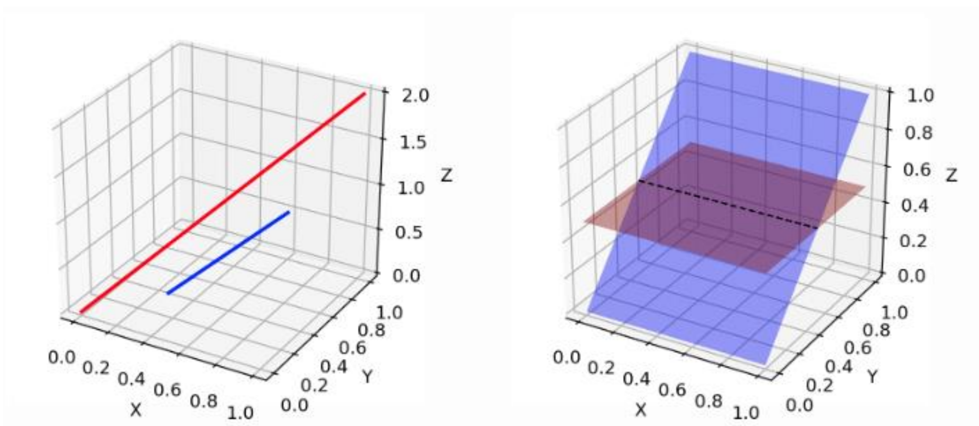




  - Parameter θ can be learned on the Wasserstein distance
  - Parameter θ cannot be learned on JS or KL divergence

- This example shows that there exist distributions that
  - **Don't converge under the JS, KL, or inverse KL**
    - For the **JS, KL, and inverse KL, there are cases where the gradient is always 0**
    - This is especially not good from an optimization perspective
  - **Do converge under the Wasserstein distance**

- Easy to get similar results, if p_{data} and p_g are on low-dimensional manifolds in high dimensional space



Low dimensional manifolds in high dimension space can hardly have overlaps.
(Left) two lines in a 3-d space. (Right) two surfaces in 3-d space

* source: https://lilianweng.github.io/lil-log/2017/08/20/from-GAN-to-WGAN.html#wasserstein-gan-wgan

- Infimum over joint distribution $\gamma \in \Pi(p_{\text{data}}, p_g)$ is computationally **intractable**

- Using Kantorovich-Rubinstein duality [Villani, 2009], Wasserstein distance becomes:

$$W(p_{\text{data}}, p_g) = \sup_{\|f\|_L \leq 1} \mathbb{E}_{x \sim p_{\text{data}}}[f(x)] - \mathbb{E}_{x \sim p_g}[f(x)]$$

  - The Supremum is over all the 1-Lipschitz functions $f : \mathcal{X} \to \mathbb{R}$
  - Let f is parameterized by w, then one could consider solving the problem

$$\max_{w \in \mathcal{W}} \mathbb{E}_{x \sim p_{\text{data}}}[f_w(x)] - \mathbb{E}_{z \sim p_z}\left[f_w(g_{\theta_g}(z))\right]$$

  - To enforce the Lipschitz constraint, **clamp the weights** to a fixed box (e.g., $\mathcal{W} = [-0.01, 0.01]^\ell$, where ℓ is dimension of parameter $w \in \mathcal{W}$)

* source: WGAN and Kantorovich-Rubinstein duality https://vincentherrmann.github.io/blog/wasserstein/

- Comparison of GAN and WGAN
  - Discriminator (outputs probability of real or fake) becomes a continuous function to help compute Wasserstein distance (with weight clamping)

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

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

$$\boxed{\begin{aligned} & g_w \leftarrow \nabla_w \left[\tfrac{1}{m} \textstyle\sum_{i=1}^m f_w(x^{(i)}) - \tfrac{1}{m} \textstyle\sum_{i=1}^m f_w(g_\theta(z^{(i)}))\right] \\ & w \leftarrow w + \alpha \cdot \text{RMSProp}(w, g_w) \\ & w \leftarrow \text{clip}(w, -c, c) \end{aligned}}$$

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

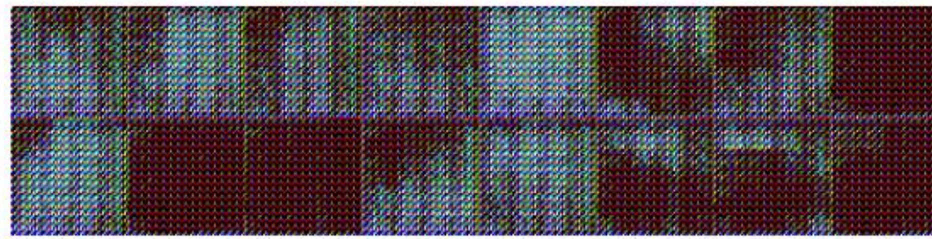
$$\boxed{\nabla_{\theta_g} \frac{1}{m} \sum_{i=1}^{m} \log\left(1 - D\left(G\left(\boldsymbol{z}^{(i)}\right)\right)\right).}$$

$$\boxed{\begin{aligned} & g_\theta \leftarrow -\nabla_\theta \tfrac{1}{m} \textstyle\sum_{i=1}^m f_w(g_\theta(z^{(i)})) \\ & \theta \leftarrow \theta - \alpha \cdot \text{RMSProp}(\theta, g_\theta) \end{aligned}}$$

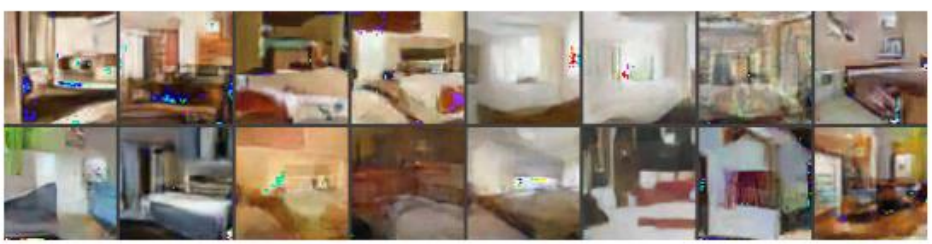
end for
The gradient-based updates can use any standard gradient-based learning rule. We used momentum in our experiments.

WGAN vs GAN


(Left) WGAN vs. (Right) GAN with DCGAN architecture . Both produce high quality samples

(Left) WGAN vs. (Right) GAN with less parameter models and without batch normalization




(Left) WGAN vs. (Right) GAN with MLP generator.
Vanilla GAN does mode collapse, while WGAN still produces good samples

* source : Arjovsky, et. al., Wasserstein GAN, ICML 2017

Enforcing the Lipschitz Constraint of Discriminator

- To maintain Lipschitz constraint WGAN uses weight clamping
  - But it is naïve and no guaranteed method
  - Weight clamping leads to optimization difficulties sometimes

- Recent works try to improve the method for maintaining Lipschitz constraint
  - Improved training of Wasserstein GANs (WGAN-GP) [Gulrajani, et. al., 2017]
    - Use **gradient penalty** to maintain Lipschitz constraint

$$\mathbb{E}_{\hat{x}\sim p_{\hat{x}}}\left[(\|\nabla_{\hat{x}}D(\hat{x})\|_2 - 1)^2\right]$$

    where $\hat{x} = \varepsilon x + (1-\varepsilon)G(z)$
  - Spectral normalization for generative adversarial networks [Miyato, et. al., 2018]
    - Control the Lipschitz constant of D by **constraining the spectral norm of each layer**

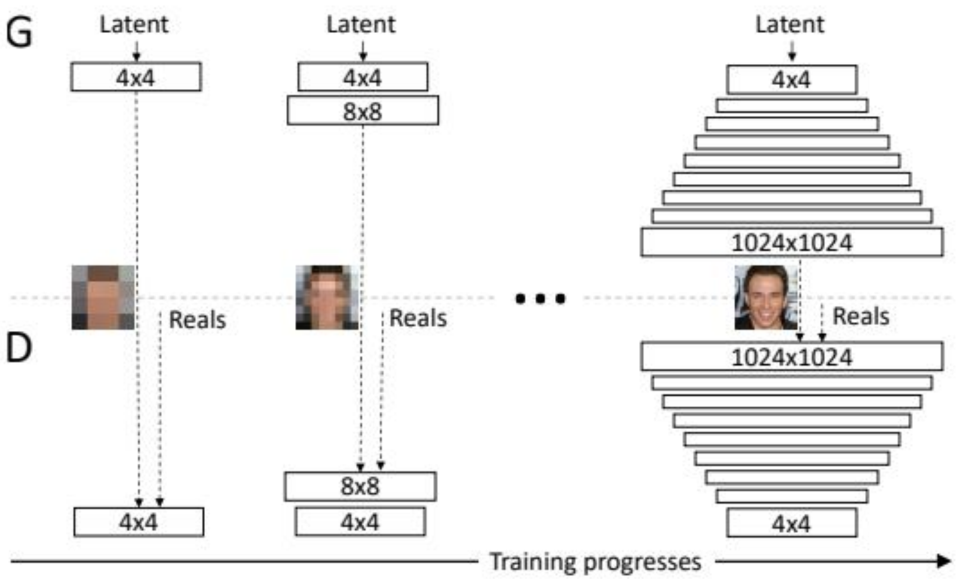
$$\bar{W}_{SN}(W) = W/\sigma(W)$$

    where $\sigma(W)$ is the spectral norm of W

- Nevertheless, stabilizing training GAN is still a on-going research topic!

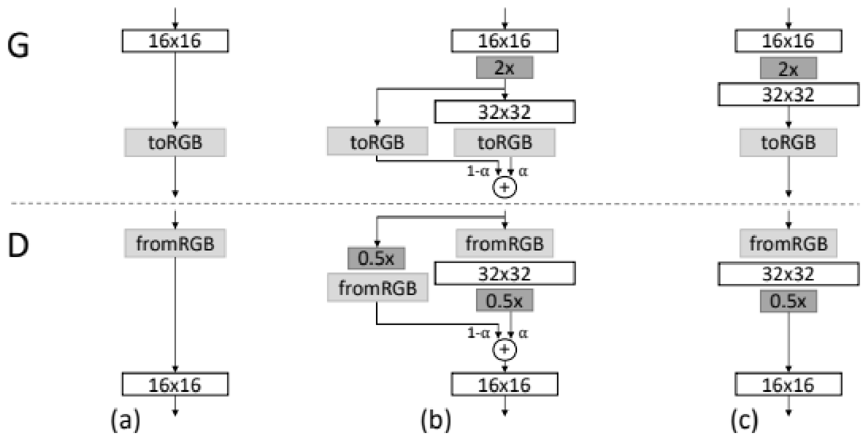
Table of Contents

- GANs produce sharp images
  - But only in fairly small resolutions and with somewhat limited variation
- Training continues to be unstable despite recent progress
- Generating **high resolution image is difficult**
  - It is **easier to tell the generated images** from training images in high-res images [Karras, et. al., 2018]
  - Grow both generator and discriminator **progressively**
  - **Start learning from easier** low-resolution images
  - Add new layers that introduce higher-resolution details as the training progress



* source : Karras, et. al., Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018

- Fade in the new layers smoothly
  - Prevent sudden *shocks* to the already well-trained, smaller-resolution layers



Transition from 16 × 16 images **(a)** to 32 × 32 images **(c)**. During the transition **(b)** we treat the layers that operate on the higher resolution like a residual block, whose weight α increases linearly from 0 to 1

- **Simplified** minibatch discrimination [Salimans, et. al., 2016]
  - **Compute standard deviation** for each feature in each spatial location and average it
  - Use it as an additional feature map for the input of the next layer

* source : Karras, et. al., Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018

Progressive GAN: Results

1024x1024 images generated using the CELEBA-HQ dataset
https://www.youtube.com/watch?v=G06dEcZ-QTg&feature=youtu.be

Mao et al. (2016b) (128 × 128) Gulrajani et al. (2017) (128 × 128) Our (256 × 256)

Visual quality comparison: LSUN bedroom

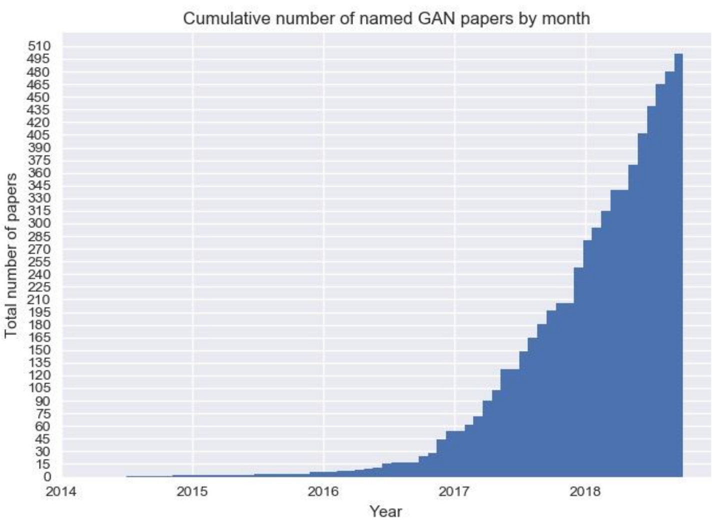
LSUN other categories generated image (256x256)

* source : Karras, et. al., Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018

The GAN-Zoo

- Lots of *GAN papers* are published since 2014
- Hundreds of papers about theories and applications
  - About better training and various applications to many types of dataset/tasks
  - If you are interested for more, see the-gan-zoo (https://github.com/hindupuravinash/the-gan-zoo)

- 3D-ED-GAN - Shape Inpainting using 3D Generative Adversarial Network and Recurrent Convolutional Networks
- 3D-GAN - Learning a Probabilistic Latent Space of Object Shapes via 3D Generative-Adversarial Modeling (github)
- 3D-IWGAN - Improved Adversarial Systems for 3D Object Generation and Reconstruction (github)
- 3D-PhysNet - 3D-PhysNet: Learning the Intuitive Physics of Non-Rigid Object Deformations
- 3D-RecGAN - 3D Object Reconstruction from a Single Depth View with Adversarial Learning (github)
- ABC-GAN - ABC-GAN: Adaptive Blur and Control for improved training stability of Generative Adversarial Networks (github)
- ABC-GAN - GANs for LIFE: Generative Adversarial Networks for Likelihood Free Inference
- AC-GAN - Conditional Image Synthesis With Auxiliary Classifier GANs
- acGAN - Face Aging With Conditional Generative Adversarial Networks
- ACGAN - Coverless Information Hiding Based on Generative adversarial networks
- acGAN - On-line Adaptative Curriculum Learning for GANs
- ACtuAL - ACtuAL: Actor-Critic Under Adversarial Learning
- AdaGAN - AdaGAN: Boosting Generative Models
- Adaptive GAN - Customizing an Adversarial Example Generator with Class-Conditional GANs
- AdvEntuRe - AdvEntuRe: Adversarial Training for Textual Entailment with Knowledge-Guided Examples
- AdvGAN - Generating adversarial examples with adversarial networks
- AE-GAN - AE-GAN: adversarial eliminating with GAN
- AE-OT - Latent Space Optimal Transport for Generative Models
- AEGAN - Learning Inverse Mapping by Autoencoder based Generative Adversarial Nets
- AF-DCGAN - AF-DCGAN: Amplitude Feature Deep Convolutional GAN for Fingerprint Construction in Indoor Localization System
- AffGAN - Amortised MAP Inference for Image Super-resolution
- AIM - Generating Informative and Diverse Conversational Responses via Adversarial Information Maximization



References

[Goodfellow, et. al., 2014] Generative adversarial nets, NIPS 2014
link: http://papers.nips.cc/paper/5423-generative-adversarial-nets.pdf

[Theis, et. al., 2016] A note on the evaluation of generative models, ICLR 2016
link: http://bethgelab.org/media/publications/1511.01844v1.pdf

[Radford, et. al., 2015] Unsupervised representation learning with deep convolutional generative adversarial networks.
link: https://arxiv.org/pdf/1511.06434.pdf

[Ledig, et. al., 2017] Photo-realistic single image super-resolution using a generative adversarial networks, CVPR 2017
link: http://openaccess.thecvf.com/content_cvpr_2017/papers/Ledig_Photo-Realistic_Single_Image_CVPR_2017_paper.pdf

[Zhu, et. al., 2017] Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks, ICCV 2017
link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=8237506

[Karras, et. al., 2018] Progressive growing of GANs for improved quality, stability, and variation, ICLR 2018
link: https://arxiv.org/abs/1710.10196

[Salimans, et. al., 2016] Improved techniques for training GANS, NIPS 2016
link: https://arxiv.org/abs/1606.03498

[Huszar 2015] How (not) to train your generative model: scheduled sampling, likelihood, adversary?
link: https://arxiv.org/pdf/1511.05101.pdf

[Arjovsky, et. al., 2017] Wasserstein GAN, ICML 2017
link: https://arxiv.org/pdf/1701.07875.pdf

References

[Arjovsdky and Bottou, 2017] Towards principled methods for training generative adversarial networks, ICLR 2017
link: https://arxiv.org/pdf/1701.04862.pdf

[Villani, 2009] Optimal transport: old and new, Grundlehren der mathematischen wissenschaften 2009
link: http://cedricvillani.org/wp-content/uploads/2012/08/preprint-1.pdf

[Reed, et. al., 2016] Generative adversarial text to image synthesis, ICML 2016
link: https://arxiv.org/pdf/1605.05396.pdf

[Wang, et. al., 2004] Image quality assessment: from error visibility to structural similarity, IEEE transactions on image processing 2004
link: https://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=1284395

[Radford, et. al., 2015] Unsupervised representation learning with deep convolutional generative adversarial networks, 2015
link: https://arxiv.org/pdf/1511.06434.pdf

[Gulrajani, et. al., 2017] Improved training of Wasserstein GANs, NIPS 2017
link: https://arxiv.org/pdf/1704.00028.pdf

[Miyato, et. al., 2018] Spectral normalization for generative adversarial networks, ICLR 2018
link: https://arxiv.org/pdf/1802.05957.pdf