17. Generative Model: GAN

Dongwoo Kim

dongwoo.kim@postech.ac.kr

CSED515 - 2023 Spring

Table of Contents

1 Generative Adversarial Network (GAN)

Known Issues with GAN

2 Wasserstein GAN

Wasserstein Distance

Wasserstein GAN

3 Improved Training of WGAN

Gradient Penalty

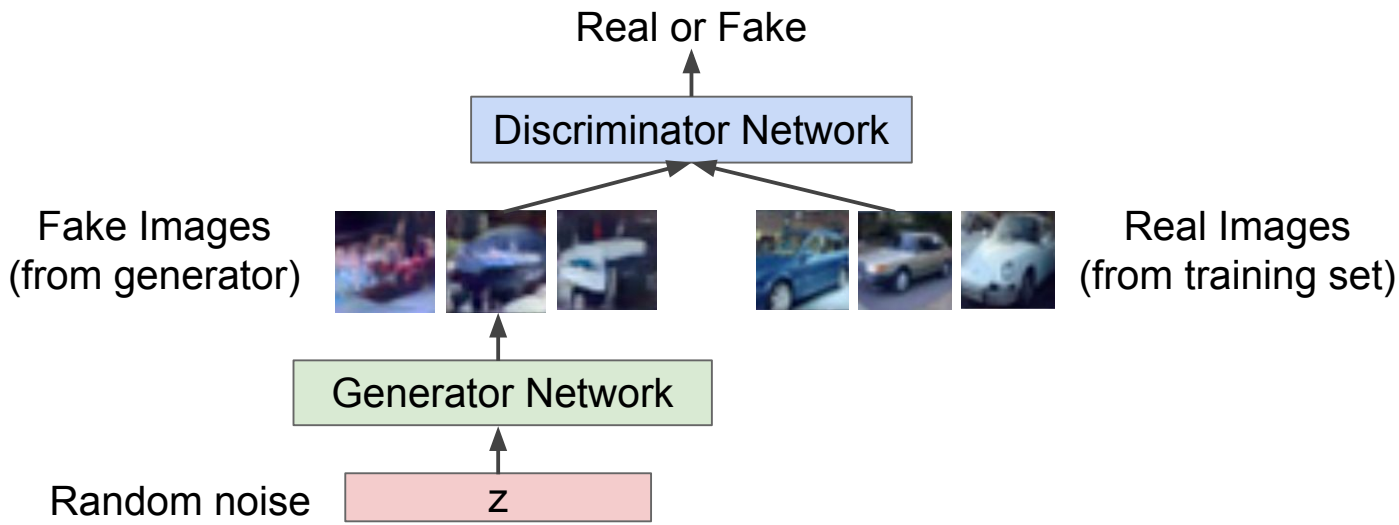
Spectral Normalization

Focus on Generation

- Training VAE requires a summation over large probability space for likelihood $p(x)$
- How can we avoid such non-trivial summation, and just learn how to generate samples? Generative Adversarial Network (GAN)
  - Specifically, GAN aims at learning a transformation from a simple distribution (e.g., Gaussian noise) to training sample distribution

Training GAN: Two-player Game

- **Generator network** (G): try to fool the discriminator by generating real-looking images
- **Discriminator network** (D): try to distinguish between real and fake images



from [Emily et al. 15]

Training GAN: Two-player Game

- **Generator network** (G): try to fool the discriminator by generating real-looking images
- **Discriminator network** (D): try to distinguish between real (1) and fake (0) images, i.e., a binary classifier

Minimax game:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}} \log D_{\phi}(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_{\phi}(G_{\theta}(z)))$$

- Discriminator ϕ aims at maximizing objective:
  - $D(x)$ to be close to 1 for real/given x
  - $D(G(z))$ to be close to 0 for fake/generated $G(z)$
- Generator θ aims at minimizing objective:
  - $D(G(z))$ to be close to 1, i.e., discriminator is fooled

Training GAN

Minimax game:

$$\min_{\theta} \max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}} \log D_\phi(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\phi(G_\theta(z)))$$

Alternate between:

- Gradient ascent on discriminator

$$\max_{\phi} \mathbb{E}_{x \sim p_{\text{data}}} \log D_\phi(x) + \mathbb{E}_{z \sim p(z)} \log(1 - D_\phi(G_\theta(z)))$$

- Gradient descent on generator

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

of which training is not going well in practice

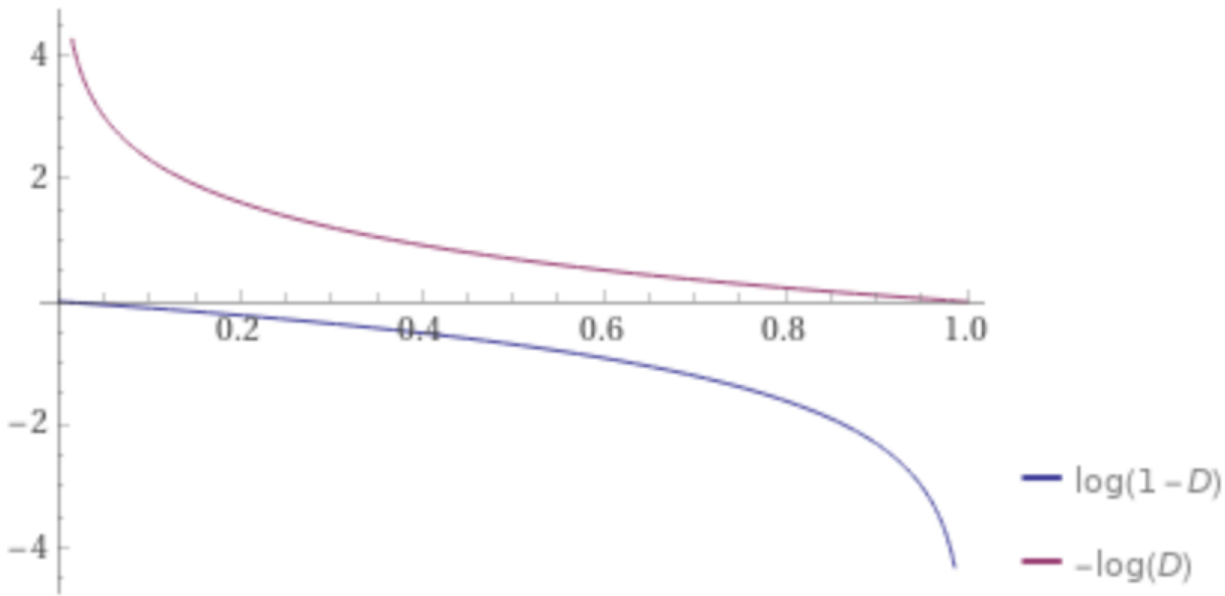
A Heuristic to Improve Convergence

Gradient descent on generator

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

- If G is very bad compared to D, then we have almost zero gradient
- Hence, instead of $\log(1 - D_\phi(G_\theta(z)))$, use

$$-\log(D_\phi(G_\theta(z)))$$



Putting it together

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:

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

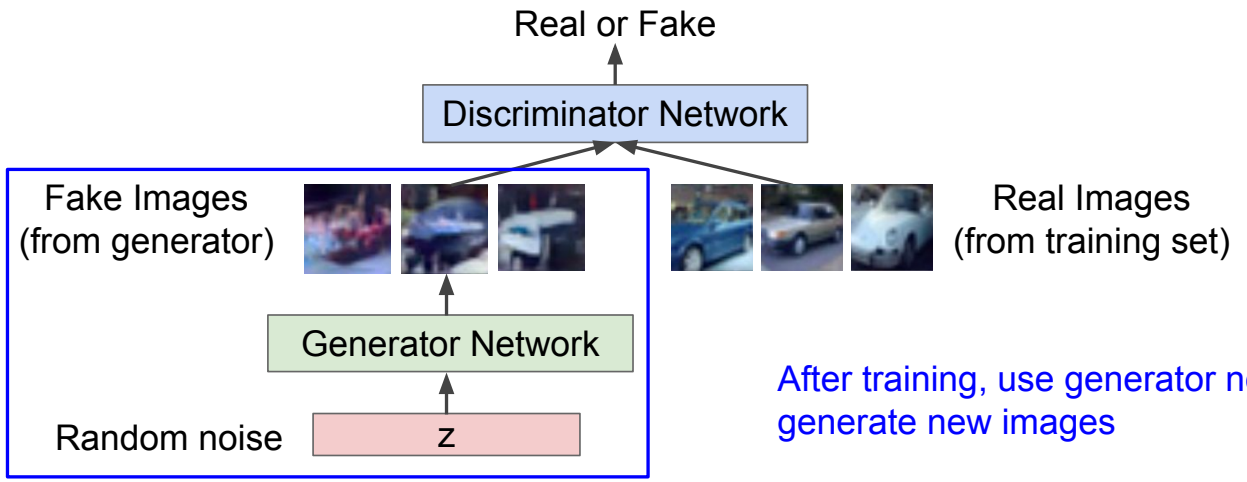
end for

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

from [Goodfellow, et al., 14]

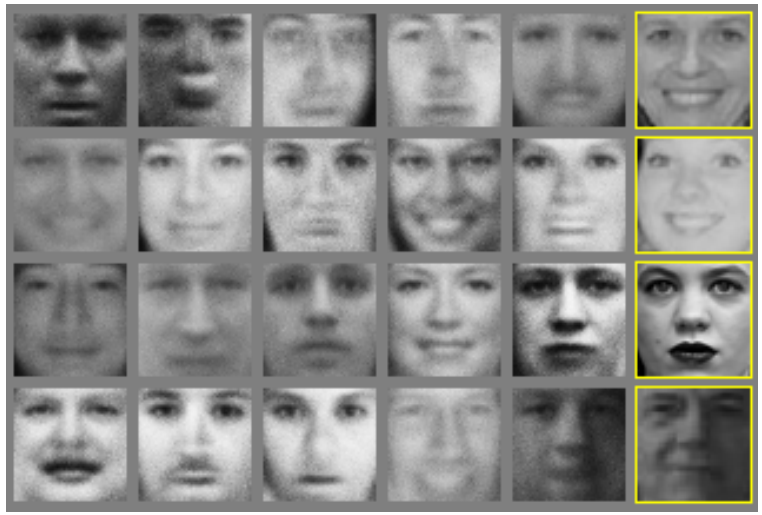
Generating Data from GAN

After training GAN, use the generator part to generate data



from [Emily et al. 15]

[Goodfellow et al. 14: GAN]



[Radford et al. 16: DCGAN]

[Zhang et al. 16: StackGAN]

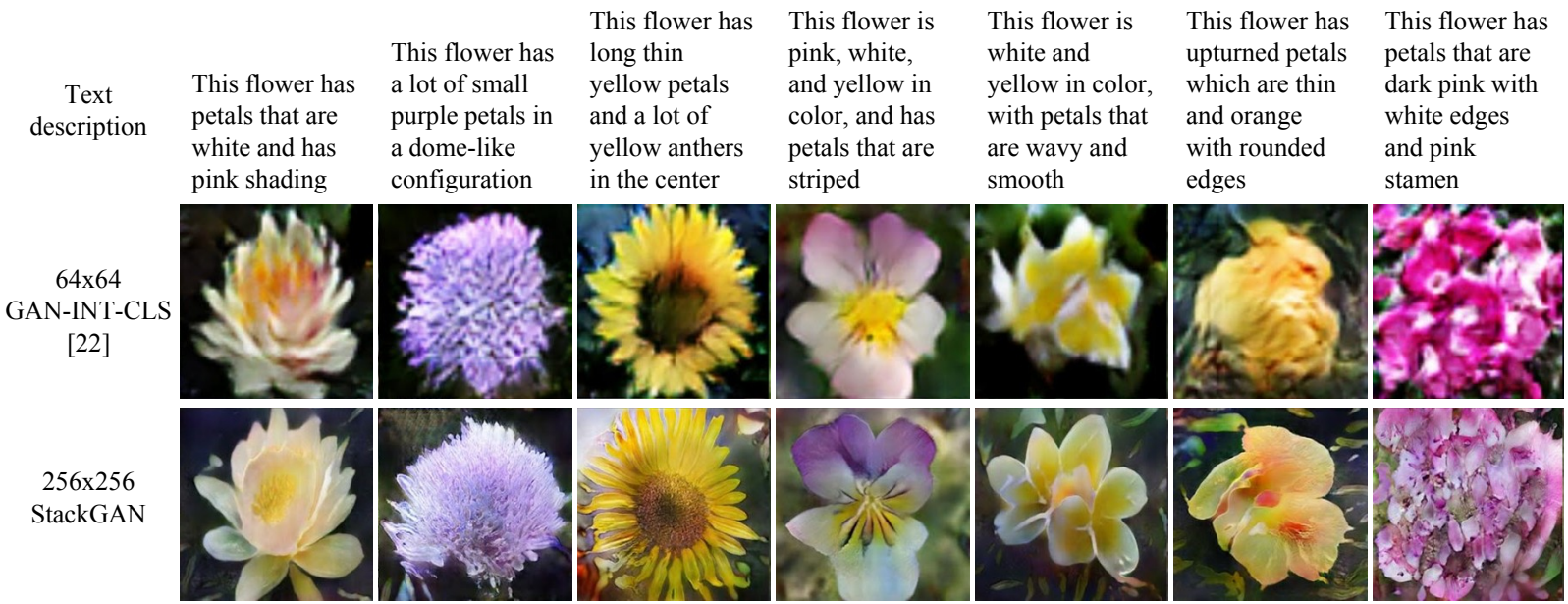


Figure 4. Example results by our proposed StackGAN and GAN-INT-CLS [22] conditioned on text descriptions from Oxford-102 test set.

Known Issues with GAN

- Non-convergence
- Mode-collapse

Non-convergence

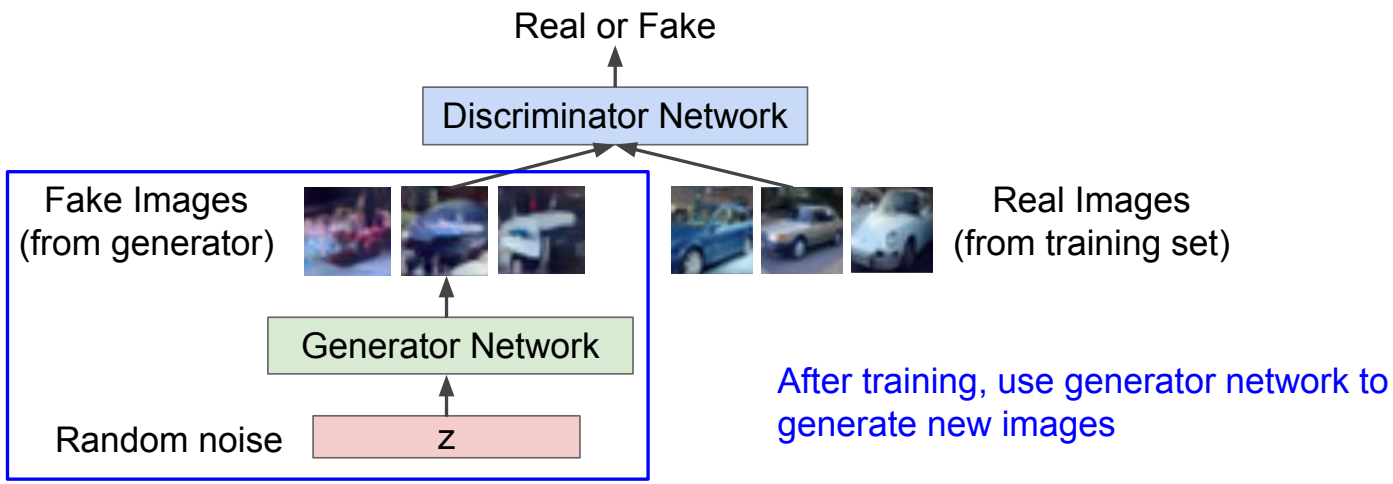
Assume our minimax objective looks like

$$\min_x \max_y V(x, y) = xy$$

- Step 0y: $x > 0, y > 0, V > 0$ $\Rightarrow$ increase y
- Step 0x: $x < 0, y > 0, V < 0$ $\Rightarrow$ decrease x
- Step 1y: $x < 0, y < 0, V > 0$ $\Rightarrow$ decrease y
- Step 1x: $x > 0, y < 0, V < 0$ $\Rightarrow$ increase x
- Step 2y: $x > 0, y > 0, V > 0$ $\Rightarrow$ same as the step 0 (repeat)

Even with a small learning rate, it will not converge!

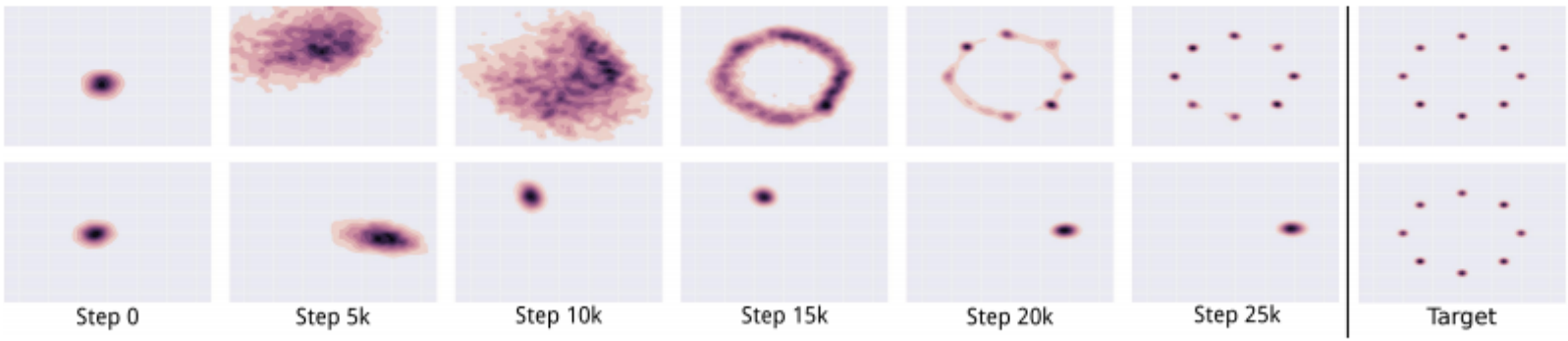
Mode-collapse



What if the generator keeps generating a single realistic image?

$\Rightarrow$ The discriminator will be always fooled by the single sample.

Mode-collapse: Example 1

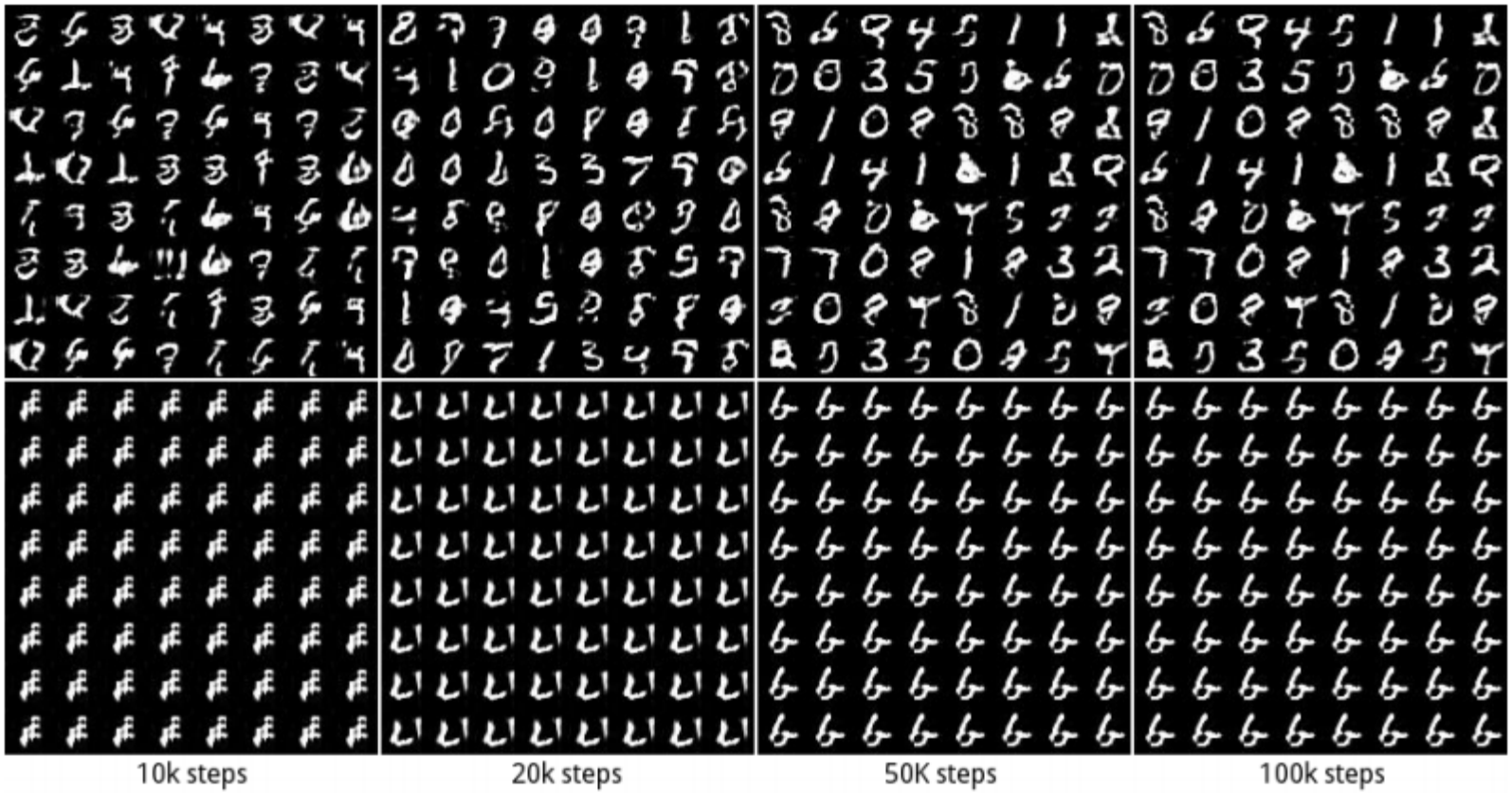


from [Metz et al. 17]

Above: well trained generator.

Below: mode-collapse

Mode-collapse: Example 2



from [Metz et al. 17]

Above: well trained generator.

Below: mode-collapse

A simple trick to avoid mode-collapse: Mini-batch GANs

- Extract features that capture diversity in the mini-batch
  - For e.g. L2 norm of the difference between all pairs from the batch
- Feed those features to the discriminator along with the image
- Feature values will differ b/w diverse and non-diverse batches
  - Thus, Discriminator will rely on those features for classification
- This in turn,
  - Will force the Generator to match those feature values with the real data
  - Will generate diverse batches
- Many more advanced techniques are also proposed.

Table of Contents

1 Generative Adversarial Network (GAN)

Known Issues with GAN

2 Wasserstein GAN

Wasserstein Distance

Wasserstein GAN

3 Improved Training of WGAN

Gradient Penalty

Spectral Normalization

Revisit GAN Objective

Okay, we know that GAN can generate realistic samples via minimax.

However, the approach (let two networks fight against each other) seems somewhat arbitrary.

How can we understand the underlying principle of the GAN?

Revisit GAN Objective

Training objective for generator is

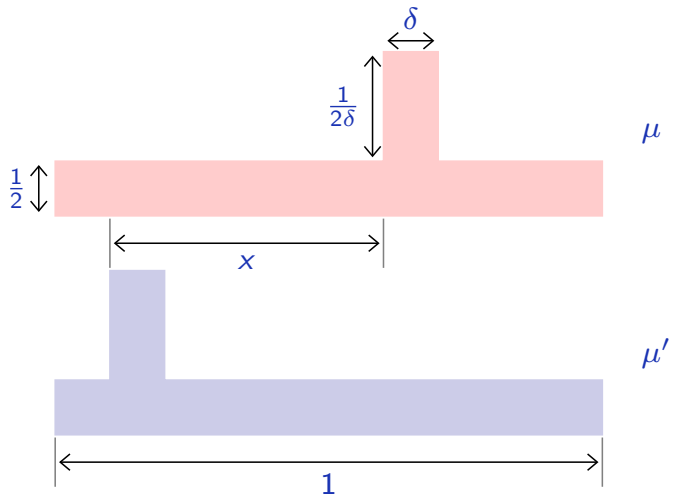
$$\max_{\phi} \mathbb{E}_{x\sim p_{\text{data}}} [\log D_\phi(x)] + \mathbb{E}_{x\sim p_G(x)} [\log(1 - D_\phi(x))]$$

For the optimal discriminator $D_\phi(x)$, we have

$$\begin{aligned}
&\mathbb{E}_{x\sim p_{\text{data}}} [\log D_\phi(x)] + \mathbb{E}_{x\sim p_G(x)} [\log(1 - D_\phi(x))] \\
&= \mathbb{E}_{x\sim p_{\text{data}}} \left[\log \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right] + \mathbb{E}_{x\sim p(z)} \left[\log \left(1 - \frac{p_{\text{data}}(x)}{p_{\text{data}}(x) + p_G(x)}\right)\right] \\
&= \underbrace{KL\left(p_{\text{data}} \| \frac{p_{\text{data}} + p_G}{2}\right) + KL\left(p_G \| \frac{p_{\text{data}} + p_G}{2}\right)}_{2\times \text{JS Divergence}(p_{\text{data}}, p_G)} - 2\log 2
\end{aligned}$$

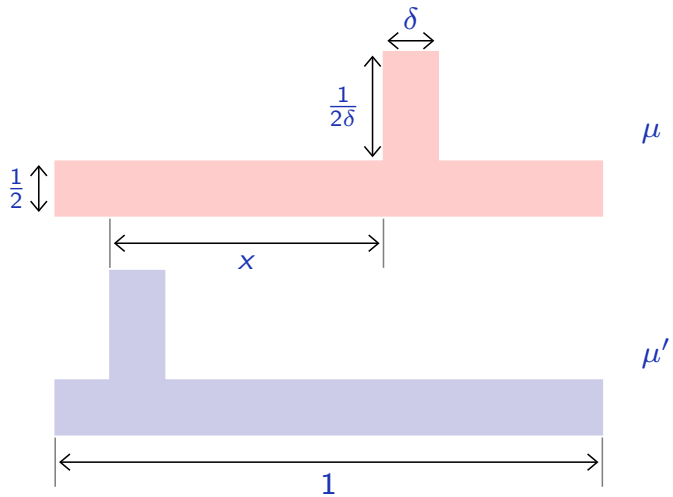
JS Divergence

Arjovsky et al. (2017) pointed out that JSD does not account [much] for the metric structure of the space. E.g. in the following example:



JS Divergence

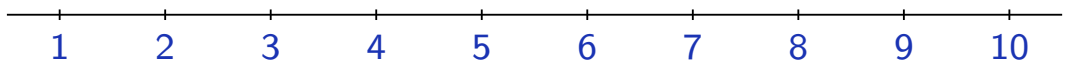
Arjovsky et al. (2017) pointed out that JSD does not account [much] for the metric structure of the space. E.g. in the following example:



We can show that $\mathbb{D}_{\mathrm{JS}}(\mu, \mu') \propto \min(\delta, |x|)$, hence all $x \notin [-\delta, \delta]$ are "as good".

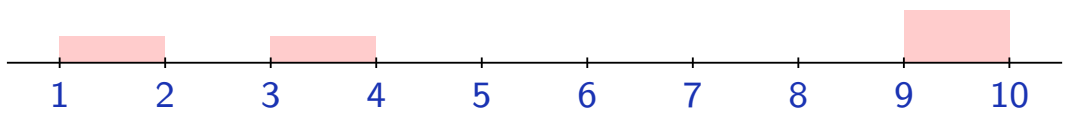
Wasserstein Distance

An alternative choice is the "earth moving distance", or Wasserstein distance, which intuitively is the minimum mass displacement to transform one distribution into the other.



Wasserstein Distance

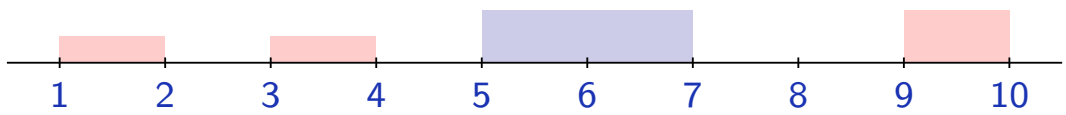
An alternative choice is the "earth moving distance", or Wasserstein distance, which intuitively is the minimum mass displacement to transform one distribution into the other.



$$\mu = \frac{1}{4}\mathbf{1}_{[1,2]} + \frac{1}{4}\mathbf{1}_{[3,4]} + \frac{1}{2}\mathbf{1}_{[9,10]}$$

Wasserstein Distance

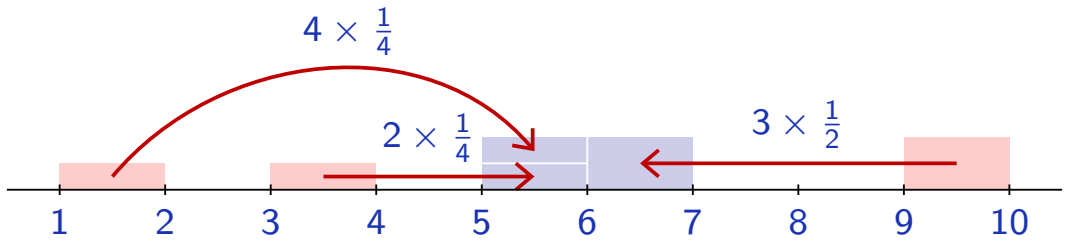
An alternative choice is the "earth moving distance", or Wasserstein distance, which intuitively is the minimum mass displacement to transform one distribution into the other.



$$\mu = \frac{1}{4}\mathbf{1}_{[1,2]} + \frac{1}{4}\mathbf{1}_{[3,4]} + \frac{1}{2}\mathbf{1}_{[9,10]} \qquad \mu' = \frac{1}{2}\mathbf{1}_{[5,7]}$$

Wasserstein Distance

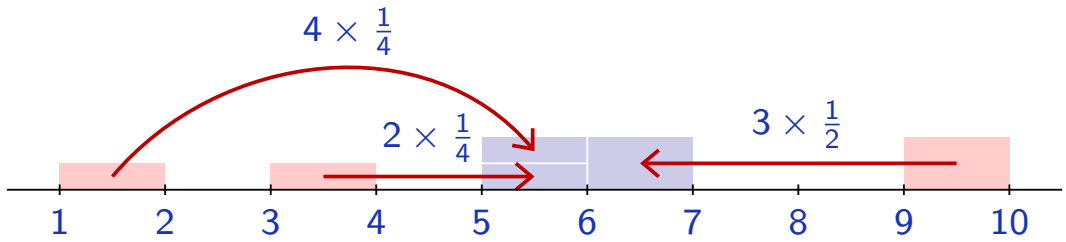
An alternative choice is the "earth moving distance", or Wasserstein distance, which intuitively is the minimum mass displacement to transform one distribution into the other.



$$\mu = \frac{1}{4}\mathbf{1}_{[1,2]} + \frac{1}{4}\mathbf{1}_{[3,4]} + \frac{1}{2}\mathbf{1}_{[9,10]} \qquad \mu' = \frac{1}{2}\mathbf{1}_{[5,7]}$$

Wasserstein Distance

An alternative choice is the "earth moving distance", or Wasserstein distance, which intuitively is the minimum mass displacement to transform one distribution into the other.

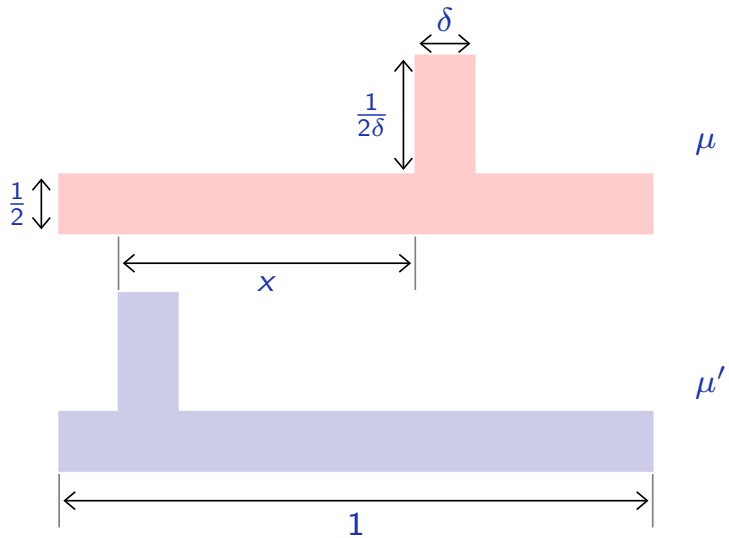


$$\mu = \frac{1}{4}\mathbf{1}_{[1,2]} + \frac{1}{4}\mathbf{1}_{[3,4]} + \frac{1}{2}\mathbf{1}_{[9,10]} \qquad \mu' = \frac{1}{2}\mathbf{1}_{[5,7]}$$

$$\mathbb{W}(\mu, \mu') = 4 \times \frac{1}{4} + 2 \times \frac{1}{4} + 3 \times \frac{1}{2} = 3$$

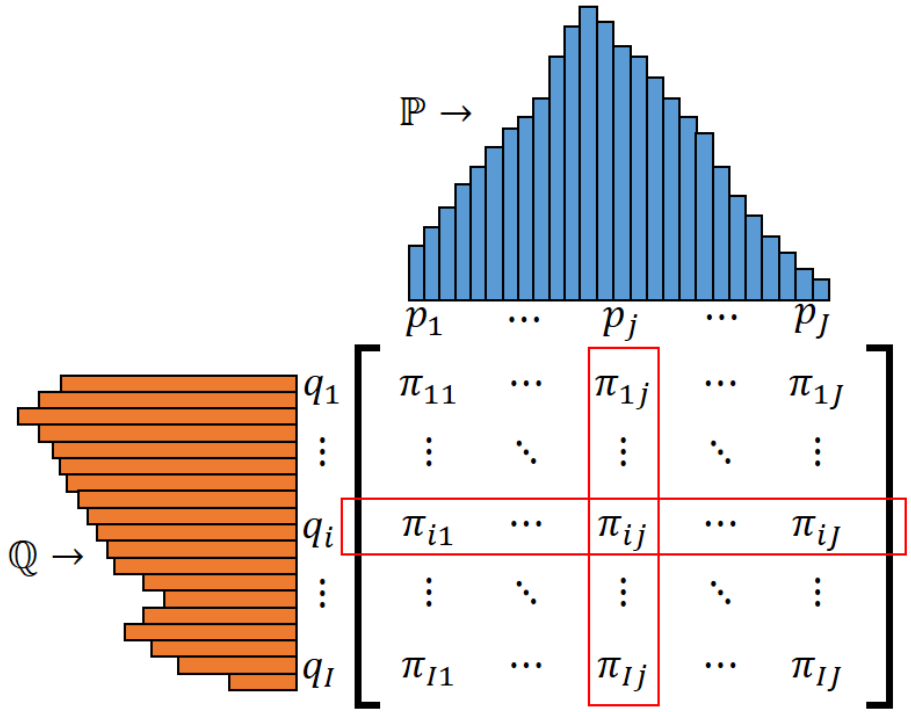
Wasserstein Distance

Intuitively, it increases monotonically with the distance between modes.



$$\mathbb{W}(\mu, \mu') = \frac{1}{2}|x|$$

Wasserstein Distance



$$\theta_W = \arg\min_\theta W(p, p_\theta) = \arg\min_\theta \inf_{\pi \in \Pi(p, p_\theta)} \mathbb{E}_{(x,y)\sim\pi} \left[\|x - y\|_2 \right].$$

Wasserstein GAN

Minimize Wasserstein distance between p and p_θ :

$$\theta_W = \arg\min_\theta W(p, p_\theta) = \arg\min_\theta \inf_{\pi \in \Pi(p, p_\theta)} \mathbb{E}_{(x,y)\sim\pi} [\|x - y\|_2] .$$

- $\Pi(p, q)$ is the set of probability distributions on $\mathcal{X} \times \mathcal{X}$ with marginals p, q, which is highly intractable.

[0] $\|h\|_L$ = Lipschitz seminorm

Wasserstein GAN

Minimize Wasserstein distance between p and p_θ :

$$\theta_W = \arg\min_\theta W(p, p_\theta) = \arg\min_\theta \inf_{\pi \in \Pi(p, p_\theta)} \mathop{\mathbb{E}}_{(x,y)\sim\pi} [\|x - y\|_2] .$$

- $\Pi(p, q)$ is the set of probability distributions on $\mathcal{X} \times \mathcal{X}$ with marginals p, q, which is highly intractable.
- Instead, minimize a dual characterization of the Wasserstein distance:

$$W(p, q) = \inf_{\pi \in \Pi(p,q)} \mathop{\mathbb{E}}_{(x,y)\sim\pi} [\|x - y\|_2] = \sup_{\|h\|_L \leq 1} \left[\mathop{\mathbb{E}}_{x\sim p} [h(x)] - \mathop{\mathbb{E}}_{x\sim q} [h(x)] \right] .$$

[0] $\|h\|_L$ = Lipschitz seminorm

Wasserstein GAN

Minimize a dual characterization of the Wasserstein distance:

$$W(p,q) = \inf_{\pi \in \Pi(p,q)} \mathop{\mathbb{E}}_{(x,y)\sim\pi} [\|x-y\|_2] = \sup_{\|h\|_L \leq 1} \left[\mathop{\mathbb{E}}_{x\sim p}[h(x)] - \mathop{\mathbb{E}}_{x\sim q}[h(x)] \right].$$

From the dual, WGAN objective can be written as

$$\min_{\theta} \max_{w\in\mathcal{W}} \mathbb{E}_{x\sim\mathbb{P}_r} [f_w(x)] - \mathbb{E}_{z\sim p(z)} [f_w(g_\theta(z))],$$

where $\mathcal{W}$ is a set of parameter satisfying 1-Lipschitz condition.

Gradient Penalty Optimization

The main issue in this formulation is to optimize the network f under a constraint on its Lipschitz seminorm

$$\|f\|_L \leq 1$$

In other words, f needs to satisfy

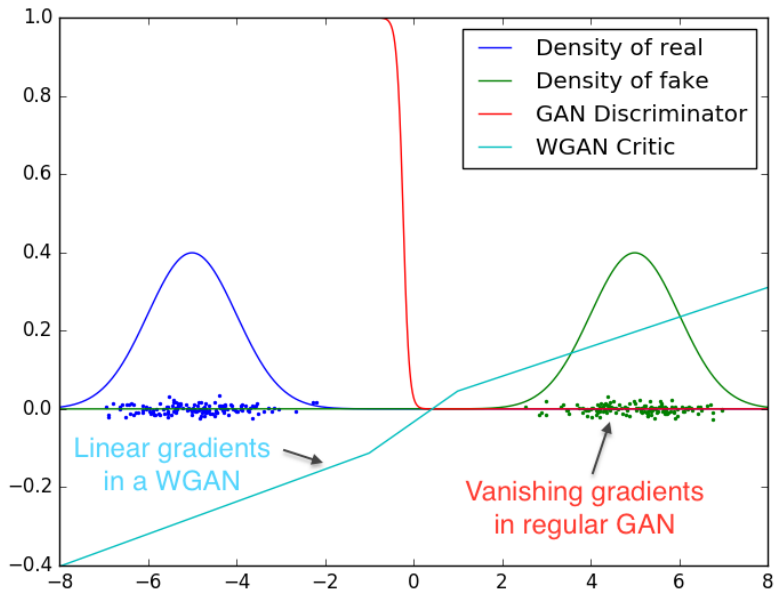
$$|f(x_1) - f(x_2)| \leq 1\,|x_1 - x_2|$$

Arjovsky et al. achieve this by clipping f's weights ($-c \leq w \leq c$).

Benefits of WGAN

The two main observed benefits are

- A greater stability of the learning process, both in principle and in their experiments: they do not witness "mode collapse".
- A greater interpretability of the loss, which is a better indicator of the quality of the samples.



[Arjovsky et al.2017]

Gradient Penalty Optimization

However, as Arjovsky et al. wrote:

> *Weight clipping is a clearly terrible way to enforce a Lipschitz constraint. If the clipping parameter is large, then it can take a long time for any weights to reach their limit, thereby making it harder to train the critic till optimality. If the clipping is small, this can easily lead to vanishing gradients when the number of layers is big, or batch normalization is not used (such as in RNNs).*
>
> [Arjovsky et al., 2017]

In some way, the resulting Wasserstein GAN (WGAN) trades the difficulty to train G for the difficulty to train D.
In practice, this weakness results in extremely long convergence times.

Table of Contents

1 Generative Adversarial Network (GAN)

Known Issues with GAN

2 Wasserstein GAN

Wasserstein Distance

Wasserstein GAN

3 Improved Training of WGAN

Gradient Penalty

Spectral Normalization

Gradient Penalty - Ishaan Gulrajani, et al., 2017

- Idea: enforce $\|f_w\|_L \leq 1$ as a soft constraint using Lagrange multipliers:

$$L(\theta, \varphi, \lambda) = \mathop{\mathbb{E}}_{x \sim p} f_w(x) - \mathop{\mathbb{E}}_{x \sim g_\theta} f_w(x) + \lambda \mathop{\mathbb{E}}_{x \sim ?} \left(\|\nabla_x f_w(x)\| - 1\right)^2.$$

- Technically need Lipschitz condition everywhere; where to enforce it?
- Uniformly along straight lines between points $x \sim p$ and $\tilde{x} \sim g_\theta$.
  - Interpolate x and $\tilde{x}$

Spectral Normalization - Takeru Miyato, et al., 2018

- Since the Lipschitz seminorm of a composition is upper-bounded by the product of the seminorms, if the non-linear layers are also Lipschitz of constant lesser than 1 (e.g. ReLU), this is a sufficient condition.
- Spectral Normalization is a layer normalization that estimates the largest singular value of a weight matrix, and rescale it accordingly.
- While computing the SVD of a matrix is expensive, computing [a good approximation of] the largest SV can be done iteratively for a reasonable cost.

Spectral Normalization

- The largest singular value of a matrix W is also its spectral norm

$$\sigma(W) = \max_{h:\|h\|_2 \leq 1} \|Wh\|_2.$$

- To calculate it, the power iteration method starts with a random vector u_0 and iterates

$$v_{n+1} = \frac{W^\top u_n}{\|W^\top u_n\|_2}$$
$$u_{n+1} = \frac{W v_{n+1}}{\|W v_{n+1}\|_2}$$

- that gives

$$\sigma(W) = \lim_{n\to\infty} u_n^\top W u_n.$$