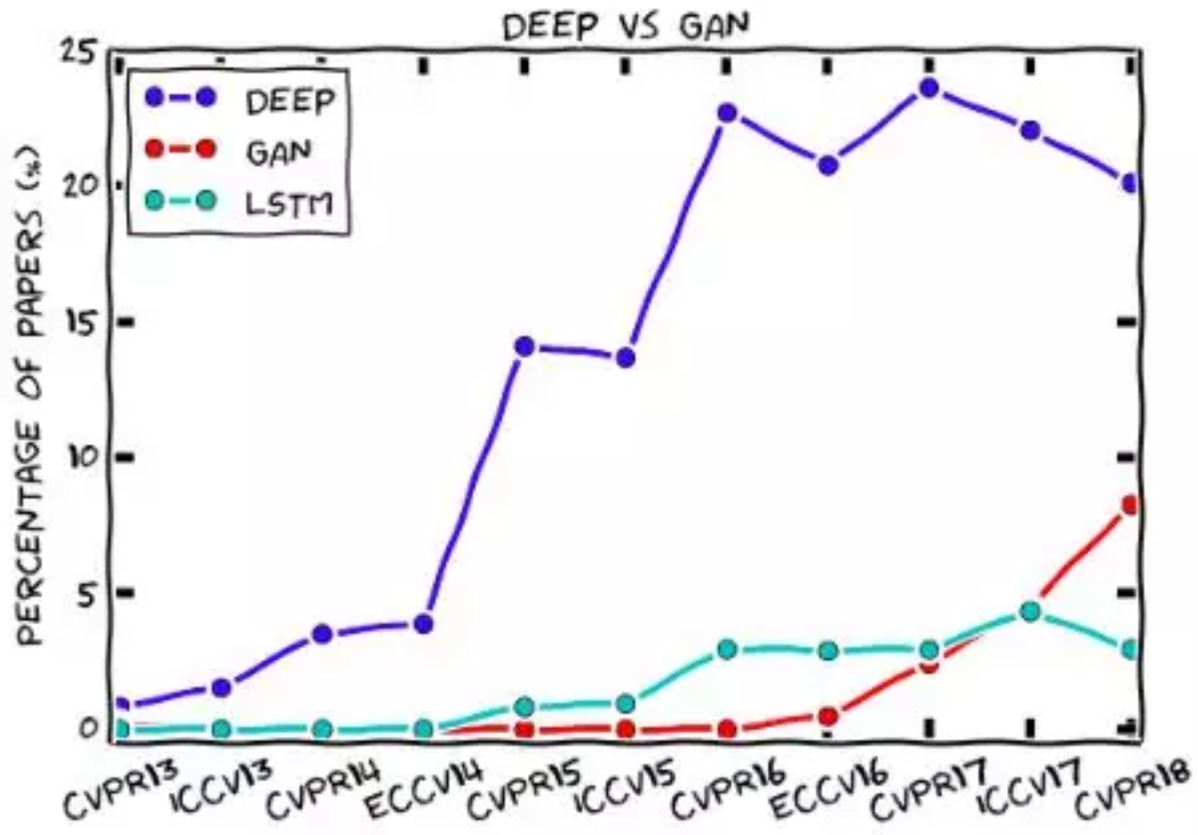
Generative Adversarial Network

Liangjie

Outline

- Basic theory (GAN, DCGAN)
- Recent improvement (WGAN, AAE, WAE)
- Applications (CGAN, Text2Img, DeblurGAN)

A hot topic

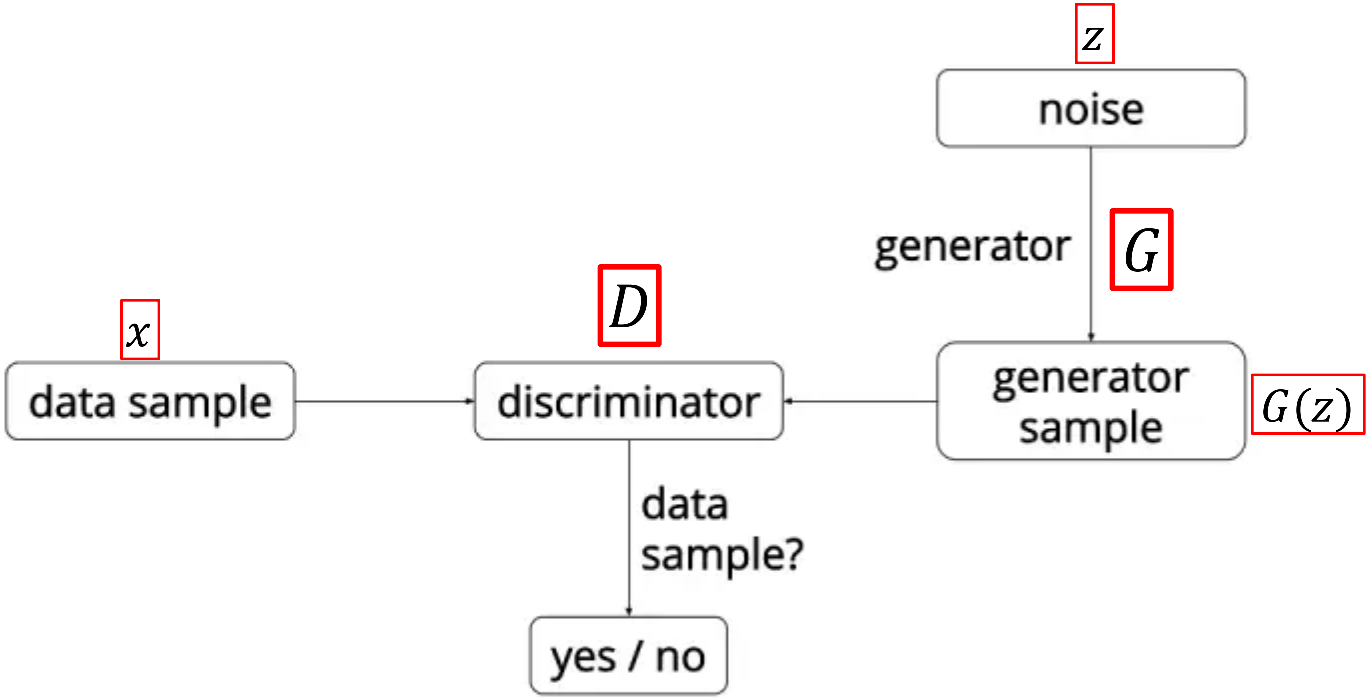


GAN related papers per community in CVPR 2018:

- style transfer/cycle GAN/domain adaption: **13**
- Photo enhancement/deblur/high-resolution reconstruction/..: **7**
- Optimizing GAN theory: **6**
- Image synthesis: **10**
- human face related: **7**
- human pose related: **4**
- Person Re-ID: **3**
- Others …

- **TOTAL: 75+, around 8% of 979 papers.**

The idea of GAN



Competition in this game drives both teams (**G** and **D**) to improve their capacity until the counterfeits are indistinguishable from the genuine articles.

I. Goodfellow, et.al. Generative Adversarial Nets, NIPS2014.

The Algorithm

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

The probability that x is a real sample.

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:

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

  end for

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

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

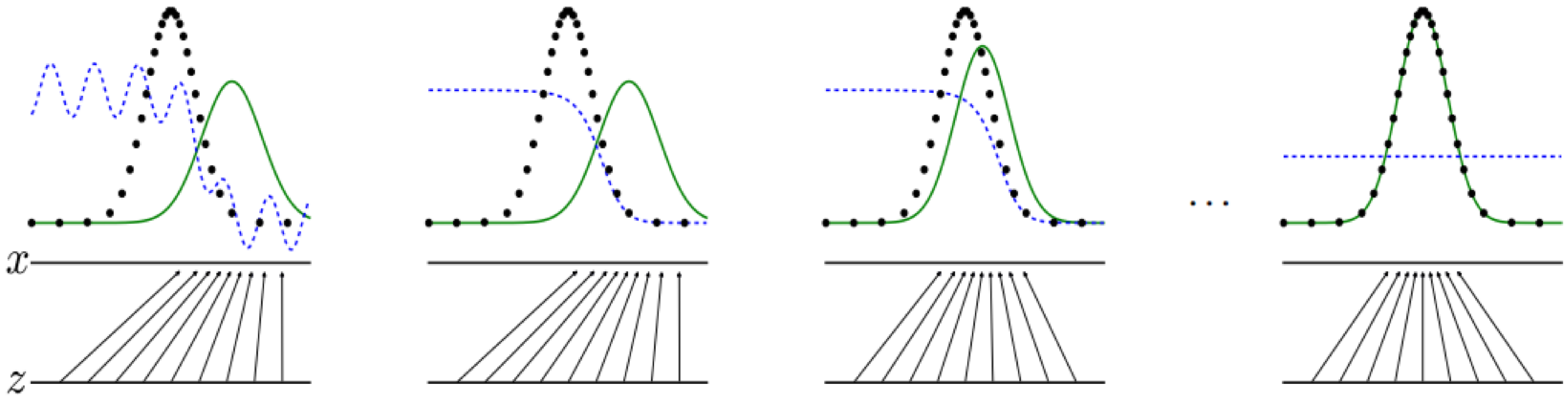
end for

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

First train **D** for several times,
Then train **G**.

k is hard to control!

The training process of the original GAN



(a) (b) (c) (d)

The original state. Both G and D are not powerful.

After training D. D can tell which sample is real.

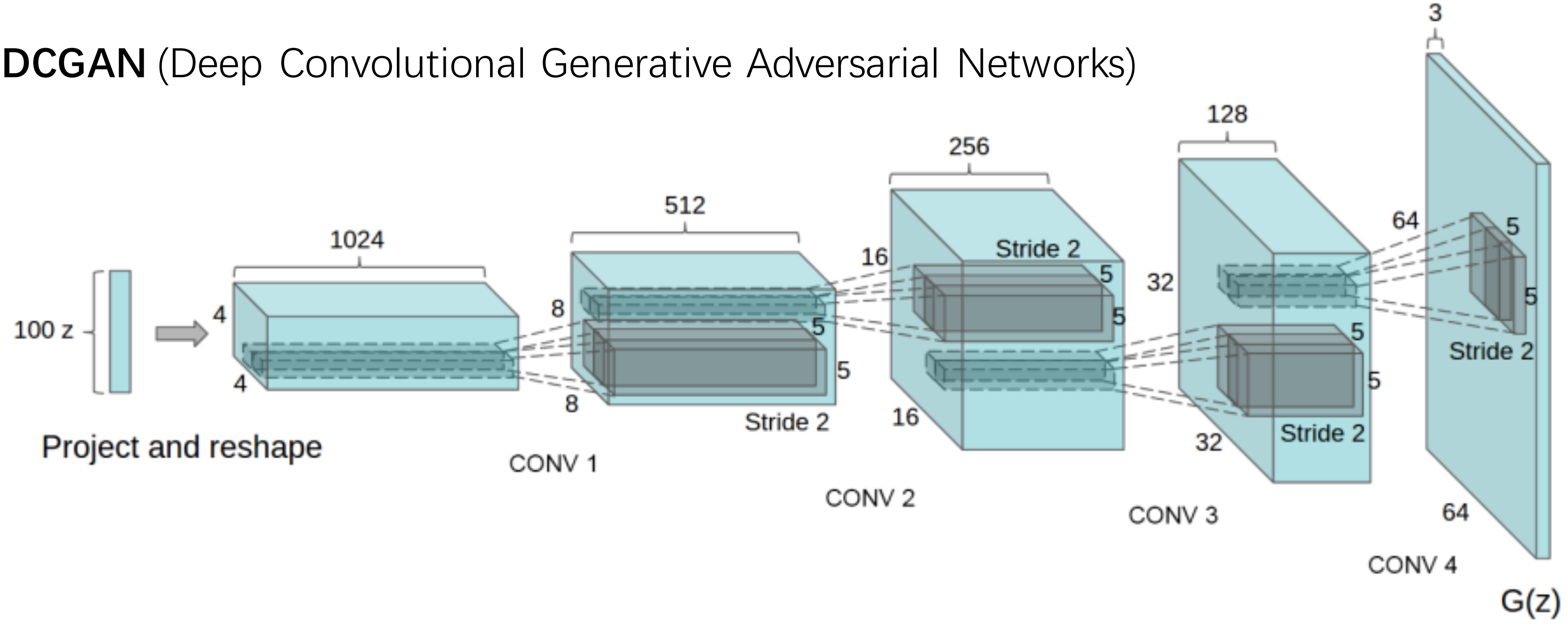
After training G. G can generate 'real' samples.

The final state. Both G and D are powerful.

The weakness

1) The structure is too simple thus lacks ability.
2) The generative model and the discriminative model should utilize the deep convolutional network.

DCGAN (Deep Convolutional Generative Adversarial Networks)



A. Radford, et.al. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks, ICLR2016.

Still have weakness

1. The training process is unstable.
2. Hardly to control the ability of the **G** and **D**.
3. The gradient always disappears.
4. Do not have an index to show the performance of the model.
5. The obtained samples lack of diversity.

→ **WGAN** (Wasserstein GAN)

Martin. Arjovsky, et.al. Wasserstein GAN. arxiv.org/abs/1701.07875

WGAN (Wasserstein GAN)

Algorithm 1 WGAN, our proposed algorithm. All experiments in the paper used the default values $\alpha = 0.00005$, $c = 0.01$, $m = 64$, $n_{\text{critic}} = 5$.

Require: : α, the learning rate. c, the clipping parameter. m, the batch size. n_{critic}, the number of iterations of the critic per generator iteration.

Require: : w_0, initial critic parameters. θ_0, initial generator's parameters.

```
while θ has not converged do
    for t = 0, ..., n_critic do
        Sample {x^(i)}_{i=1}^m ~ P_r a batch from the real data.
        Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
        g_w ← ∇_w [ (1/m) Σ_{i=1}^m f_w(x^(i)) − (1/m) Σ_{i=1}^m f_w(g_θ(z^(i))) ]
        w ← w + α · RMSProp(w, g_w)
        w ← clip(w, −c, c)
    end for
    Sample {z^(i)}_{i=1}^m ~ p(z) a batch of prior samples.
    g_θ ← −∇_θ (1/m) Σ_{i=1}^m f_w(g_θ(z^(i)))
    θ ← θ − α · RMSProp(θ, g_θ)
end while
```

The modifications:

1. No sigmoid in last layer of **D**.
2. No $\log(a)$ in the loss of the **G** and **D**.
3. For **D**, clip the updated parameter to [-c , c].

Martin Arjovsky, et.al. Wasserstein GAN. arxiv.org/abs/1701.07875

The problem in GAN

1. If the **D** has strong ability to make decision, the gradient of **G** will disappear

When training the **D**: we maximize the equation:

$$L(D, g_\theta) = \mathbb{E}_{x\sim P_r} \log D(x) + \mathbb{E}_{x\sim P_g} \log[1 - D(x)]$$

$$= P_r(x) \log D(x) + P_g(x) \log[1 - D(x)]$$

The derivative of $D(x)$ equals 0:

$$\frac{P_r(x)}{D(x)} - \frac{P_g(x)}{1 - D(x)} = 0.$$

The optimal discriminator :

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

M. Arjovsky, et.al. Wasserstein GAN. arxiv.org/abs/1701.07875

The problem in GAN

1. If the **D** has strong ability to make decision, the gradient of **G** will disappear

When training the **G**: we minimize the equation:

$$D^*(x) = \frac{P_r(x)}{P_r(x) + P_g(x)}$$

$$\mathbb{E}_{x\sim P_r} \log D(x) + \mathbb{E}_{x\sim P_g} \log[1 - D(x)]$$

$$\Rightarrow \mathbb{E}_{x\sim P_r} \log \frac{P_r(x)}{\frac{1}{2}(P_r(x)+P_g(x))} + \mathbb{E}_{x\sim P_g} \log[1 - \frac{P_r(x)}{\frac{1}{2}(P_r(x)+P_g(x))}] - 2\log 2$$

M. Arjovsky, et.al. Towards Principled Methods for Training Generative Adversarial Networks. ICLR 2017

The problem in GAN

1. If the **D** has strong ability to make decision, the gradient of **G** will disappear

KL divergence :

$$KL(P_1||P_2) = \mathbb{E}_{x\sim P_r}\log\frac{P_1}{P_2}$$

JS divergence :

$$JS(P_1||P_2) = \frac{1}{2}KL(P_1||(P_1+P_2)/2) + \frac{1}{2}KL(P_2||(P_1+P_2)/2)$$

$$\mathbb{E}_{x\sim P_r}\log\frac{P_r(x)}{\frac{1}{2}(P_r(x)+P_g(x))} + \mathbb{E}_{x\sim P_g}\log[1-\frac{P_r(x)}{\frac{1}{2}(P_r(x)+P_g(x))}] - 2\log2$$

⇨ $= 2JS(P_r||P_g) - 2log2$ ⇨ When training the **G**, we minimize the JS divergence between real distribution and generative distribution.

M. Arjovsky, et.al. Towards Principled Methods for Training Generative Adversarial Networks. ICLR 2017

The problem in GAN

1. If the **D** has strong ability to make decision, the gradient of **G** will disappear

$$JS(P_r||P_g) = \begin{cases} log2 & P_r = 0, P_g \neq 0; or\ P_r \neq 0, P_g = 0 \\ 0 & P_r = 0, P_g = 0; or\ \boxed{P_r \neq 0, P_g \neq 0} \end{cases}$$

The two distribution can hardly have combinations since z is obtained randomly and the intrinsic dimension is much lower than the real space.

⇨ The gradient of **G** disappears.

M. Arjovsky, et.al. Towards Principled Methods for Training Generative Adversarial Networks. ICLR 2017

The problem in GAN

2. The training process is unstable.

For **G**, it minimizes the object function:

$$\mathbb{E}_{x\sim P_g}[-\log D^*(x)] = KL(P_r||P_g) - \mathbb{E}_{x\sim P_g}\log[1-D^*(x)]$$

$$= KL(P_r||P_g) - 2JS(P_r||P_g) + 2\log 2 + \mathbb{E}_{x\sim P_r}\log D^*(x)$$

$$\Rightarrow KL(P_r||P_g) - 2JS(P_r||P_g)$$

One to close while another to far! ⇒ unstable

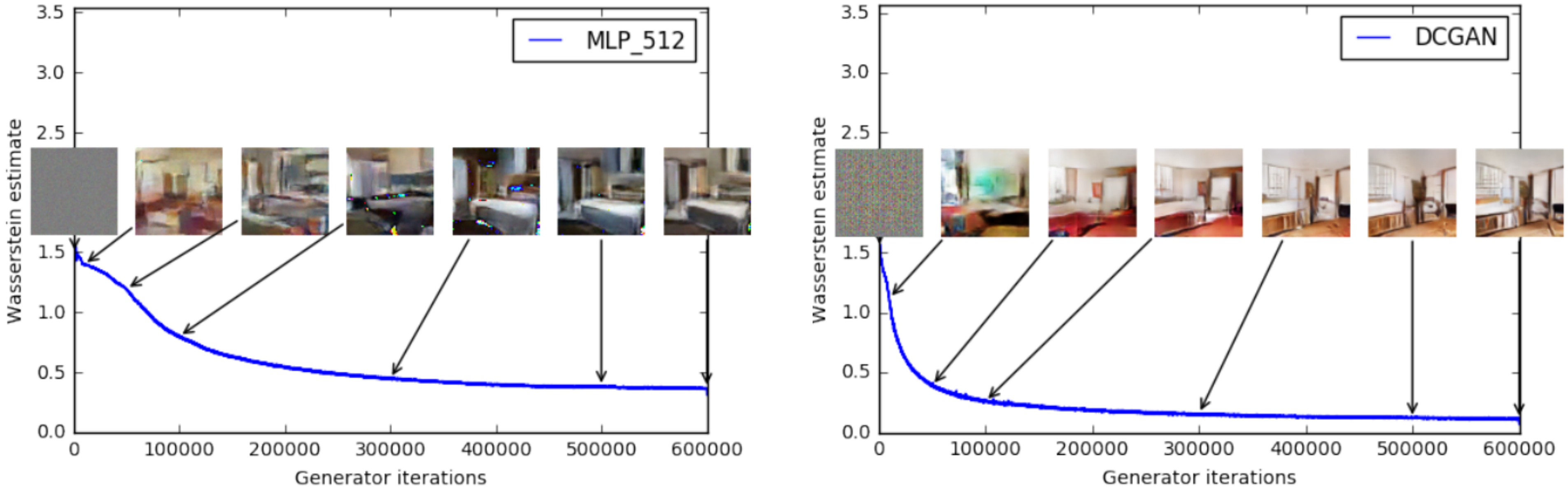
M. Arjovsky, et.al. Towards Principled Methods for Training Generative Adversarial Networks. ICLR 2017

The problem in GAN

3. Lack an index to show the performance of the model

Wasserstein Distance (Earth-Mover distance):

$$W(P_r, P_g) = inf_{\gamma \sim \Pi(P_r, P_g)} \mathbb{E}_{(x,y)\sim\gamma}[||x - y||]$$



M. Arjovsky, et.al. Wasserstein GAN. arxiv.org/abs/1701.07875

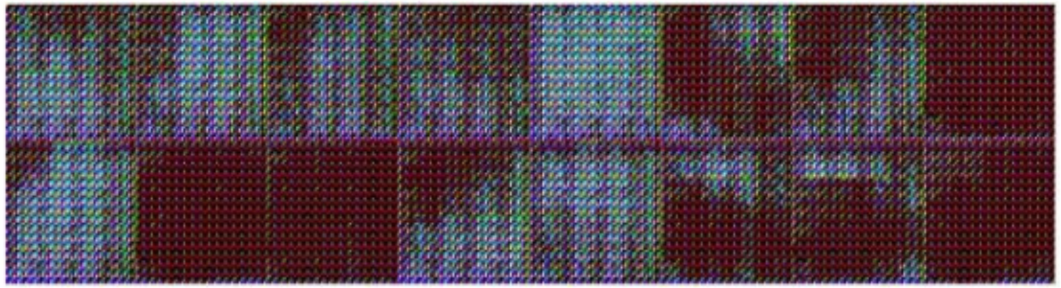
Wasserstein GAN

Advantages:

- does not need to carefully control the capacity of G and D
- alleviates the mode collapse problem
- provides a metric of model capacity during training
- does not require well-designed network architectures

Some Results

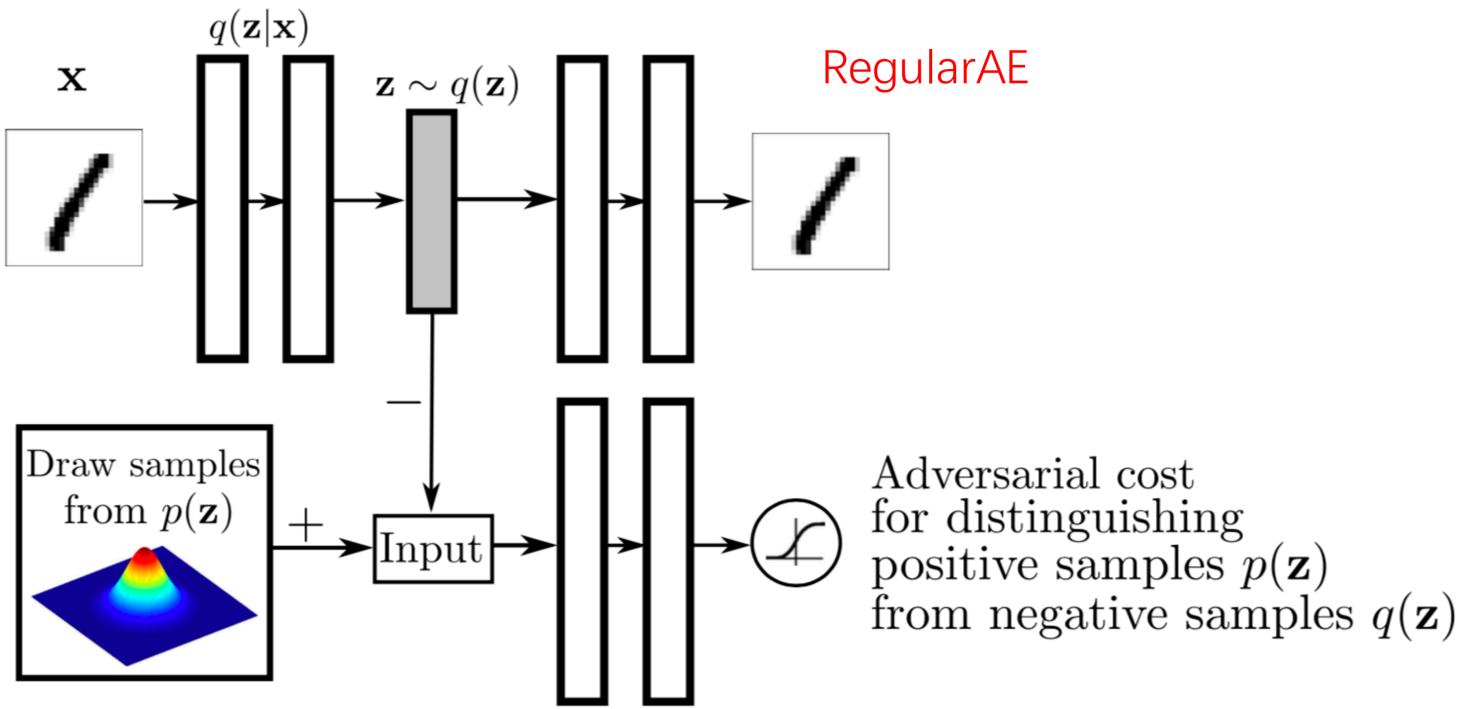
WGANs vs DCGANs (without BN)



WGANs vs GANs (without CNN)

M. Arjovsky, et.al. Wasserstein GAN. arxiv.org/abs/1701.07875

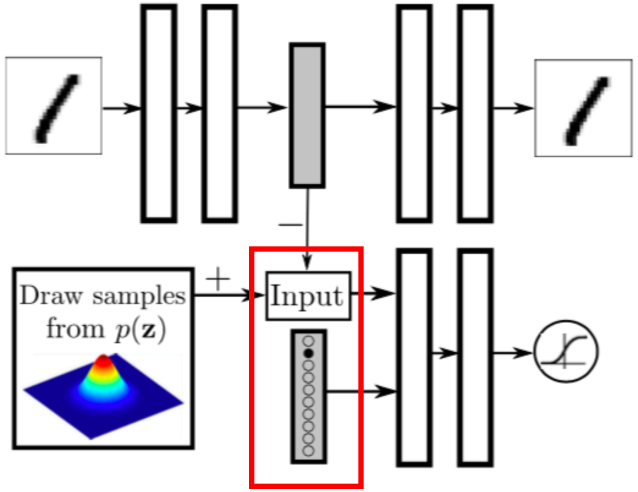
Adversarial Auto-Encoders

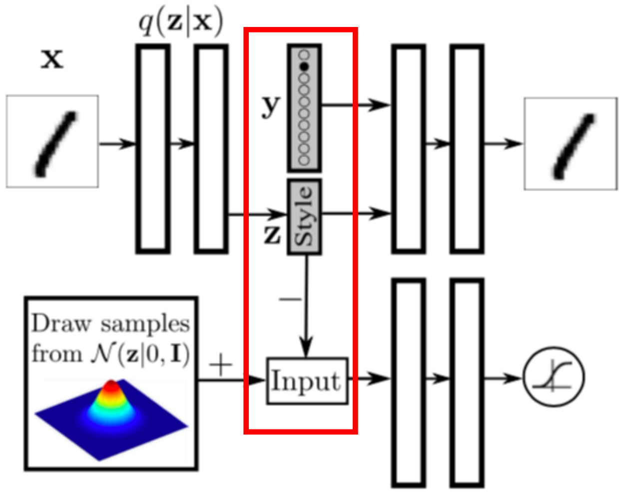


RegularAE

D：z is derived from the encoder of the random sampled from prior distribution

Finally: q_z has same distribution with p_z, we can feed the random sampled latent code to the decoder, to generate new samples.





A Makhzani, et.al. Adversarial Auto-encoders. arXiv preprint arXiv:1511.05644

Adversarial Auto-Encoders

(a) MNIST

(b) SVHN

A Makhzani, et.al. Adversarial Auto-encoders. arXiv preprint arXiv:1511.05644

Wasserstein Auto-Encoders

Encoder:

- Based on a specified divergence, matches the encoded distribution Q_Z of training samples to the prior P_Z.
- Ensures that the latent codes fed to the decoders are informative to reconstruct the training sample.

Decoder:

- According to the cost function, reconstruct the encoded training sample.

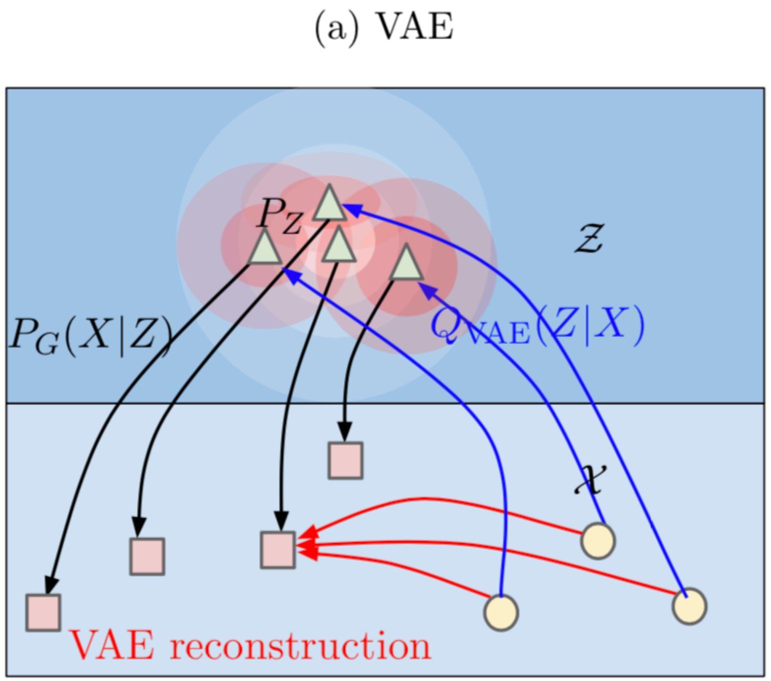
Loss Function: Optimal transport cost $W_c(P_X, P_G)$, a family of Wasserstein distance

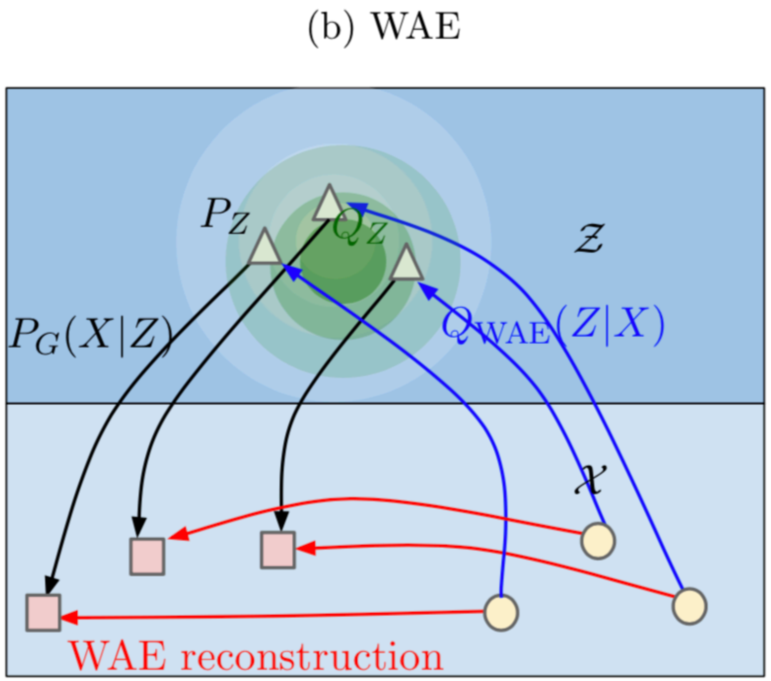
$$D_{\mathrm{WAE}}(P_X, P_G) := \inf_{Q(Z|X)\in\mathcal{Q}} \mathbb{E}_{P_X}\mathbb{E}_{Q(Z|X)}\big[c(X, G(Z))\big] + \lambda \cdot \mathcal{D}_Z(Q_Z, P_Z),$$

Reconstruction loss   Penalty

Ilya Tolstikhin, et.al. Wasserstein Auto-Encoders. ICLR 2018 (8.0)

Wasserstein Auto-Encoders





For each samples in X (yellow circle), forces $Q(Z|X)$ (triangle) to match $P(Z)$ (white shape).

Forces $Q_z = \int Q(Z|X)dP_X$ to match $P(Z)$ (green ball).

Ilya Tolstikhin, et.al. Wasserstein Auto-Encoders. ICLR 2018

Wasserstein Auto-Encoders

Algorithm 1 Wasserstein Auto-Encoder with GAN-based penalty (WAE-GAN).

Require: Regularization coefficient $\lambda > 0$.

Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.

while (ϕ, θ) not converged **do**

Sample $\{x_1, \dots, x_n\}$ from the training set

Sample $\{z_1, \dots, z_n\}$ from the prior P_Z

Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$

Update D_γ by ascending:

$$\frac{\lambda}{n} \sum_{i=1}^{n} \log D_\gamma(z_i) + \log\big(1 - D_\gamma(\tilde{z}_i)\big)$$

Update Q_ϕ and G_θ by descending:

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) - \lambda \cdot \log D_\gamma(\tilde{z}_i)$$

end while

Algorithm 2 Wasserstein Auto-Encoder with MMD-based penalty (WAE-MMD).

Require: Regularization coefficient $\lambda > 0$, characteristic positive-definite kernel k.

Initialize the parameters of the encoder Q_ϕ, decoder G_θ, and latent discriminator D_γ.

while (ϕ, θ) not converged **do**

Sample $\{x_1, \dots, x_n\}$ from the training set

Sample $\{z_1, \dots, z_n\}$ from the prior P_Z

Sample $\tilde{z}_i$ from $Q_\phi(Z|x_i)$ for $i = 1, \dots, n$

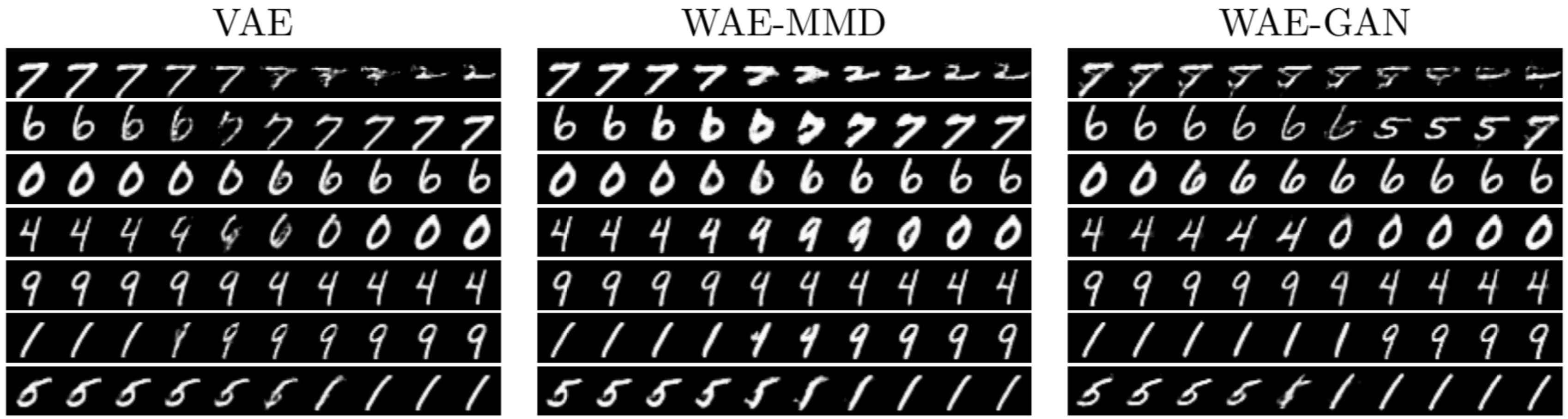
Update Q_ϕ and G_θ by descending:

$$\frac{1}{n} \sum_{i=1}^{n} c\big(x_i, G_\theta(\tilde{z}_i)\big) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(z_\ell, z_j) + \frac{\lambda}{n(n-1)} \sum_{\ell \neq j} k(\tilde{z}_\ell, \tilde{z}_j) - \frac{2\lambda}{n^2} \sum_{\ell, j} k(z_\ell, \tilde{z}_j)$$

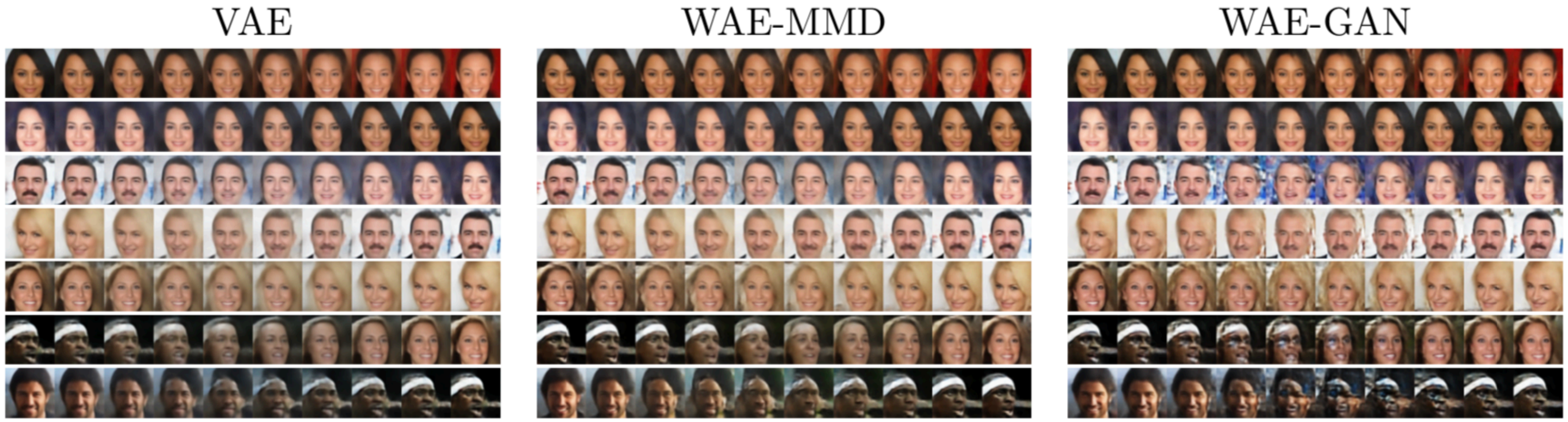
end while

Ilya Tolstikhin, et.al. Wasserstein Auto-Encoders. ICLR 2018

Wasserstein Auto-Encoders



MNIST: 28*28



celebA: 64*64

Ilya Tolstikhin, et.al. Wasserstein Auto-Encoders. ICLR 2018

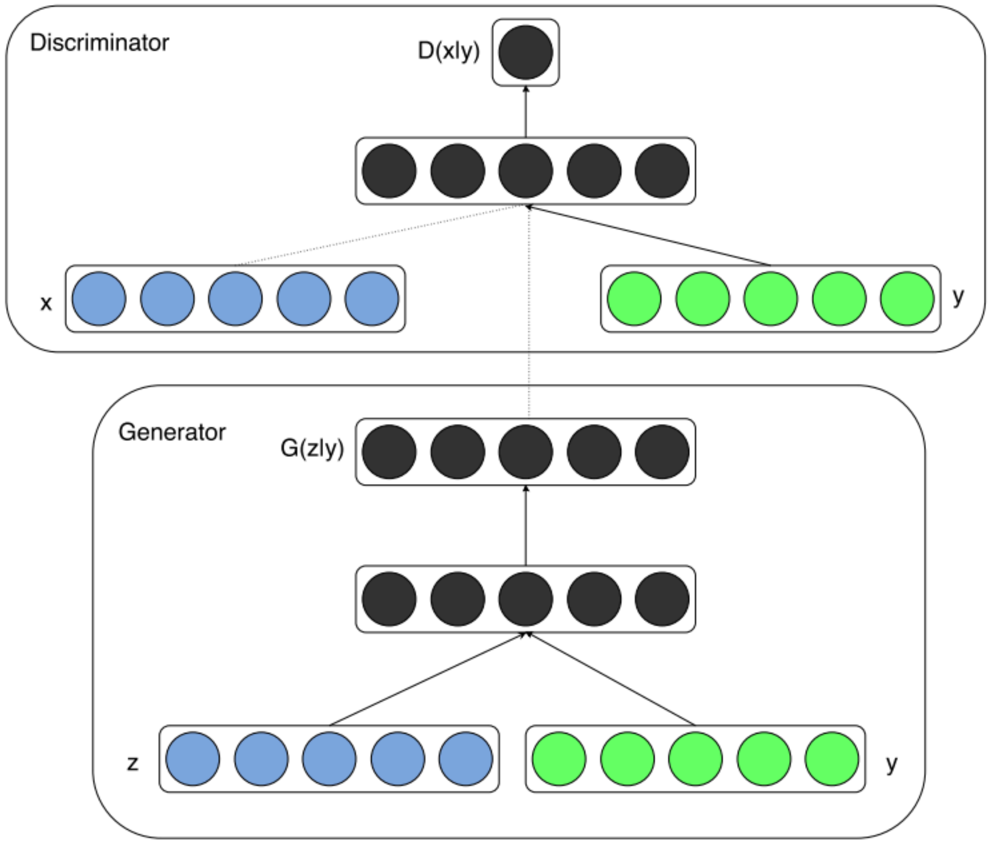
Applications

1. Generate new samples for training, especially those hard to collect.
2. Combine with various low-level vision tasks such as segmentation, etc.
3. Complete the broken image (inpainting).
4. Generate high resolution image from lower one.
5. Generate images from the text descriptions.

...

GAN can be combined with any task that generates some new things, e.g., the mask in segmentation task, the broken part in image inpainting task, or the high-resolution image, etc.

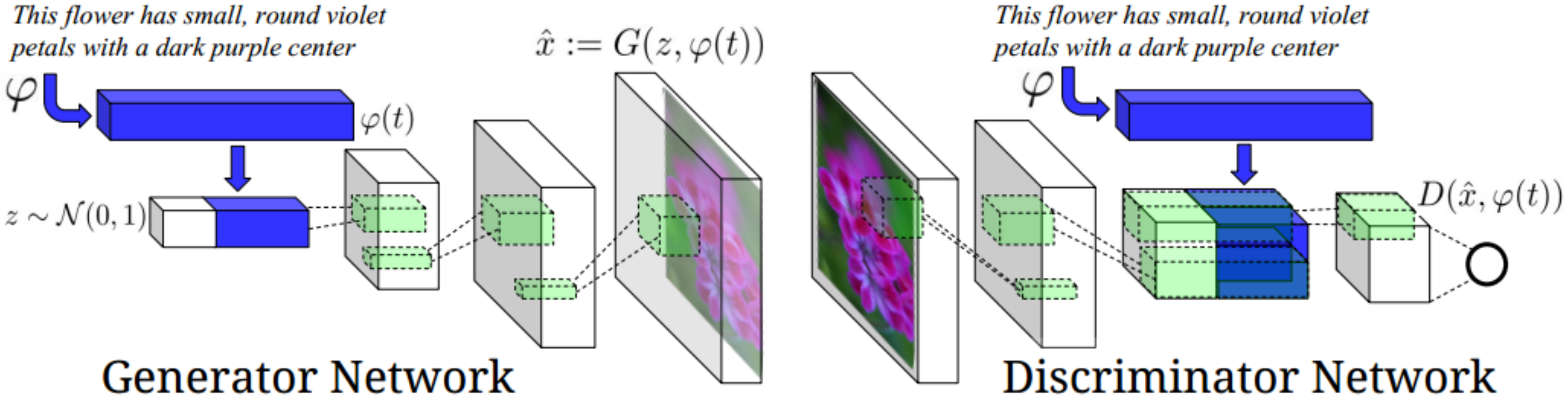
Conditional GAN



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

Mehdi Mirza, et.al. Conditional Generative Adversarial Nets. arXiv preprint arXiv:1411.1784

Text2Img



this small bird has a pink breast and crown, and black primaries and secondaries.

this magnificent fellow is almost all black with a red crest, and white cheek patch.

the flower has petals that are bright pinkish purple with white stigma

this white and yellow flower have thin white petals and a round yellow stamen

S.Reed, et.al. Generative Adversarial Text to Image Synthesis. ICML2016

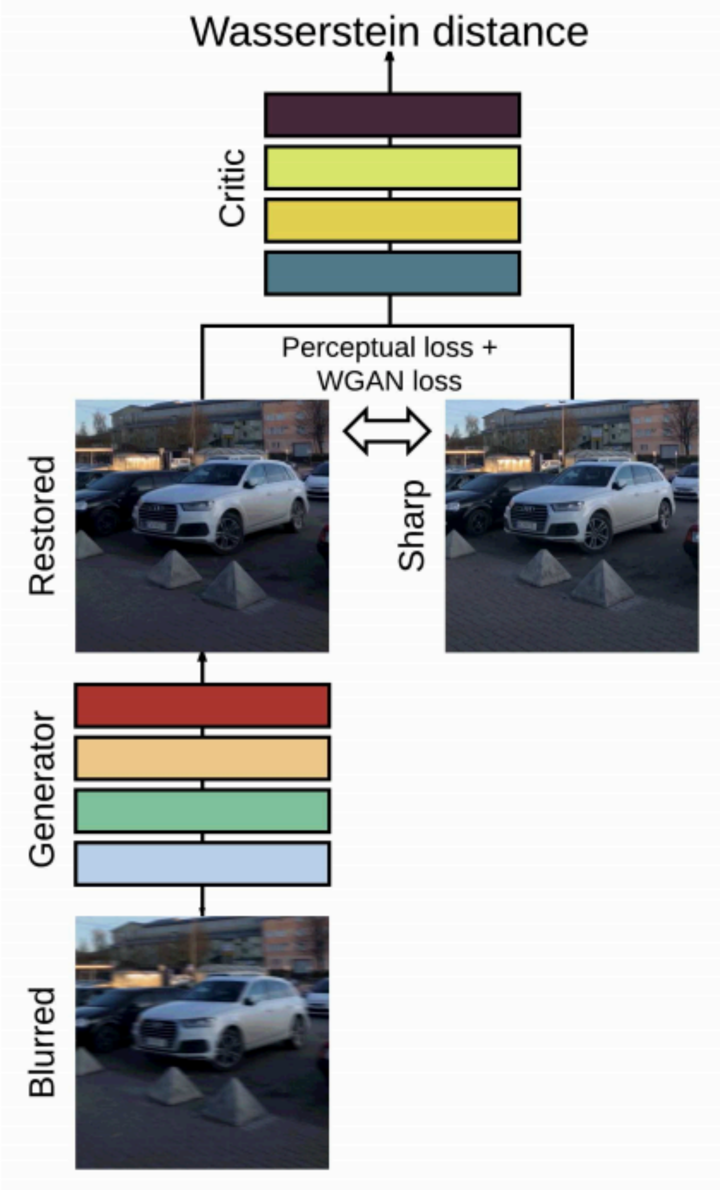
DeblurGAN

WGAN loss:

Wasserstein distance

Perceptual loss:

the **difference** between the VGG-19 conv3.3 feature maps of the sharp and restored images.



Orest Kupyn, et.al. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks. CVPR 2018

Thanks