AutoEncoder, GAN

목차

- AutoEncoder
- Generative Adversarial Network

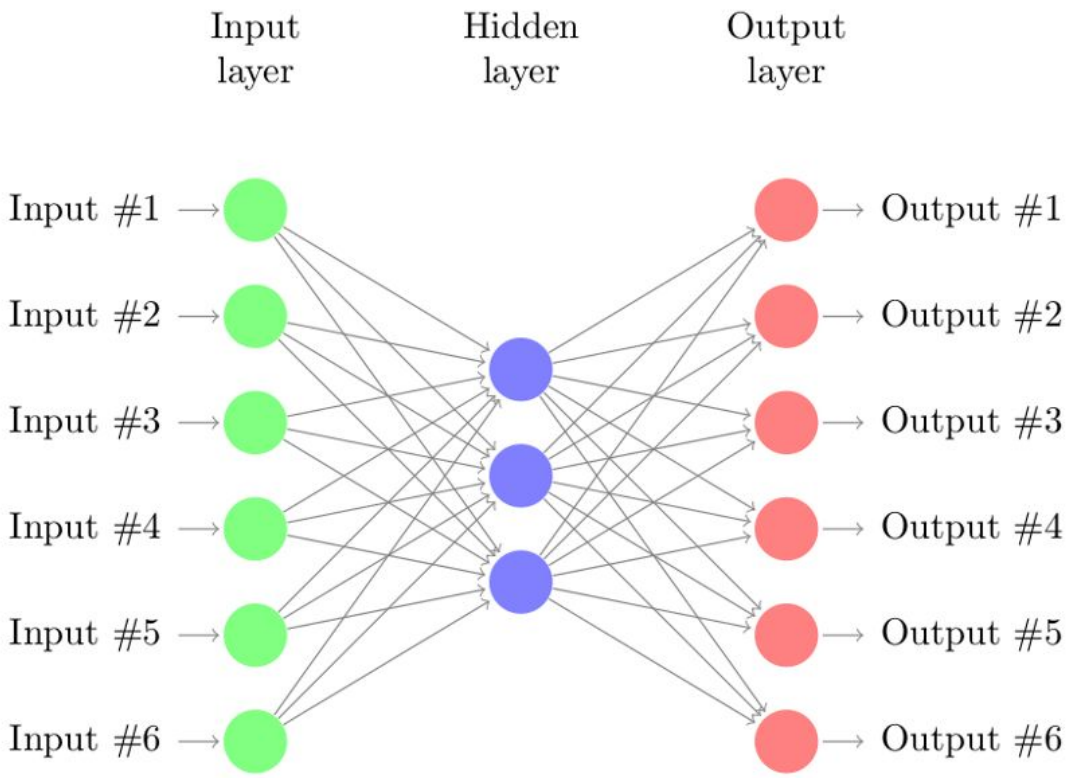
AutoEncoder

AutoEncoder

- What is AutoEncoder?
  - An autoencoder is an artificial neural network used for **unsupervised learning**.
  - An autoencoder **learn a representation for a set of data**, typically for the purpose of **dimensionality reduction**.
  - The autoencoder concept has become more widely used for learning **generative models** of data(VAE)

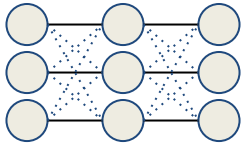
AutoEncoder

- Architecture

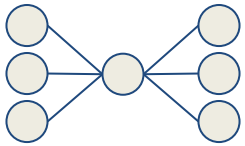


AutoEncoder

- Architecture
  - An autoencoder is learned to produced the same output as the input(X'=X).
  - The easiest answer is the identity function(x = f(x)). But it is meaningless to approximate it with neural networks.

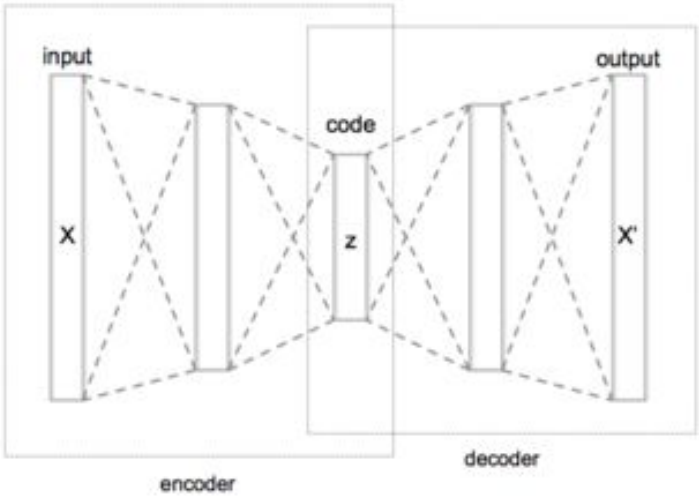


  - Autoencoder uses hidden nodes that are less than the input / output nodes,(X-> z(low dimensionality) -> X'(=X)) so they get interesting functionality.



AutoEncoder

- Architecture



- The process of mapping (or compressing) data to a sub-dimension and then restoring it back to its original dimensions is the same as the encoding-decoding process, which compresses the data and restores the original without loss.
- Therefore, an artificial neural network with this structure is called 'Auto Encoder' and can be divided into an encoder that compresses data and a decoder that restores compressed data.

Dimensionality reduction

- The auto encoder can reduce the dimensions while preserving the characteristics of the original data.

- If linear activations are used, then the optimal solution to an autoencoder is strongly related to PCA.

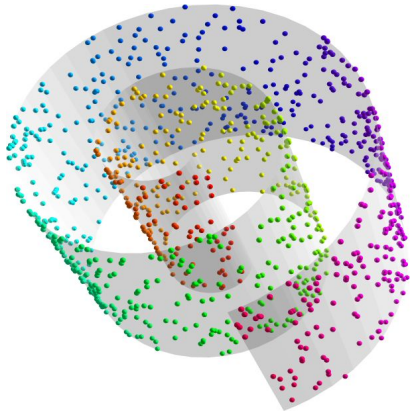
- If non-linear activations are used, then an autoencoder exhibit a interesting non-linear characteristic.

Manifold Learning

- Manifold Learning 관점에서 살펴본다면?

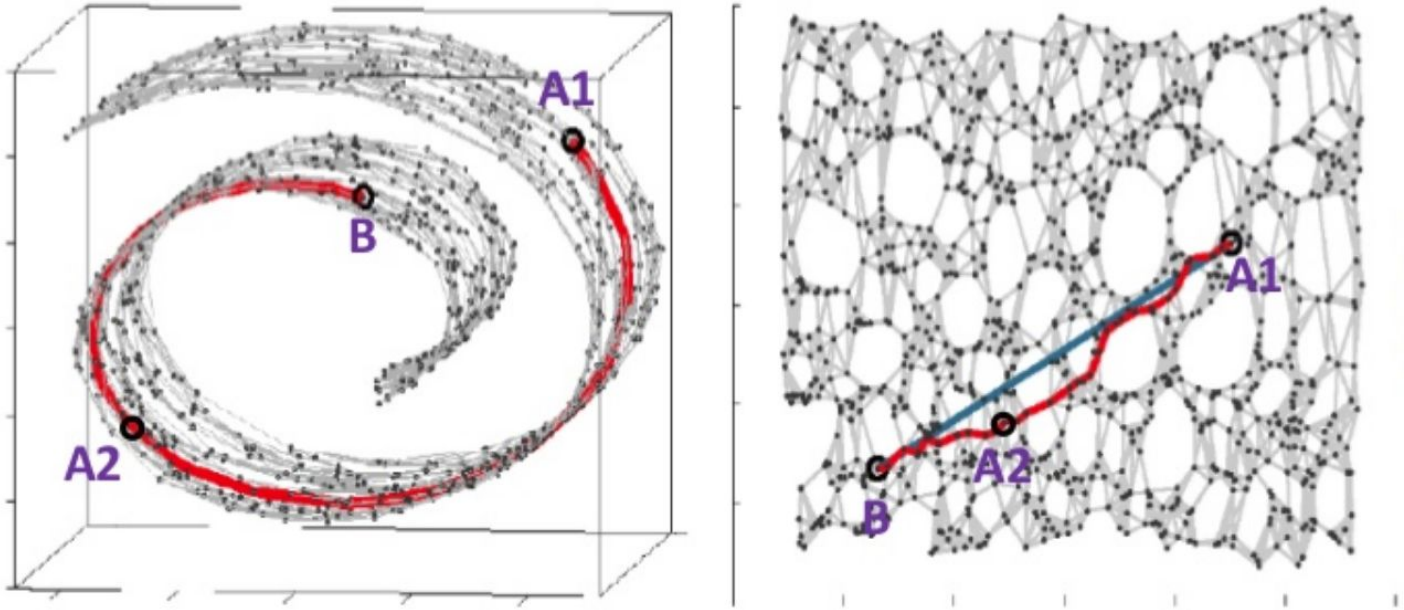
- 고차원의 데이터는 고차원 공간 안에 골고루 분포되어 있지 않고, 어느 특정한 subspace에 분포되어 있으며, 이 subspace를 manifold라 한다. manifold를 벗어나게 되면 데이터의 밀도가 급격하게 낮아진다.

- 실제로 어떤 dataset이 존재하는 manifold를 찾는다면 그 manifold 공간 위에서 좌표들이 변할 때 원 data들도 유의미하게 변화하게 된다.



Manifold Learning

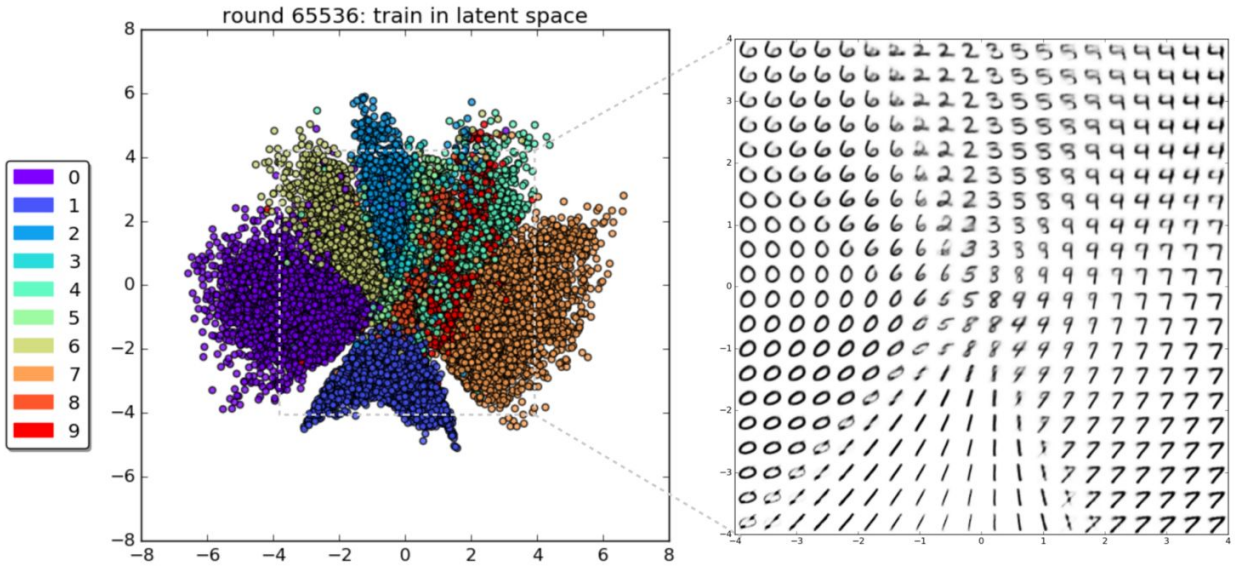
- 실제 데이터 공간에서 의미론적으로 가깝다고 생각되는 두 데이터의 거리가 멀 수 있음으로 manifold를 잘 찾는것은 중요하다.



Manifold Learning(example)

- autoencoder를 통해 manifold를 학습시킬 때, 따로 label을 주지 않아도 매우 dominant한 feature인 label 별로 분리되어 모이는 것을 볼 수 있으며 중요한 feature를 찾았다고 볼 수 있다.

- manifold 위에서 움직일 때 유의미한 변화(기울기, 굵기)등이 변화한다.



Pre-training

- Pretrainin?

- The early structure of Deep nets had a “Local minima” problem caused by the vanishing gradient.

- Vanishing gradient problem?

- Pretraining Deep nets with autoencoder solves the problem.

Visualization

- Visualization?

- Autoencoder can learn data projections that are more interesting than PCA or other basic techniques.

- For 2D or 3D visualization specifically t-SNE is probably the best algorithm.

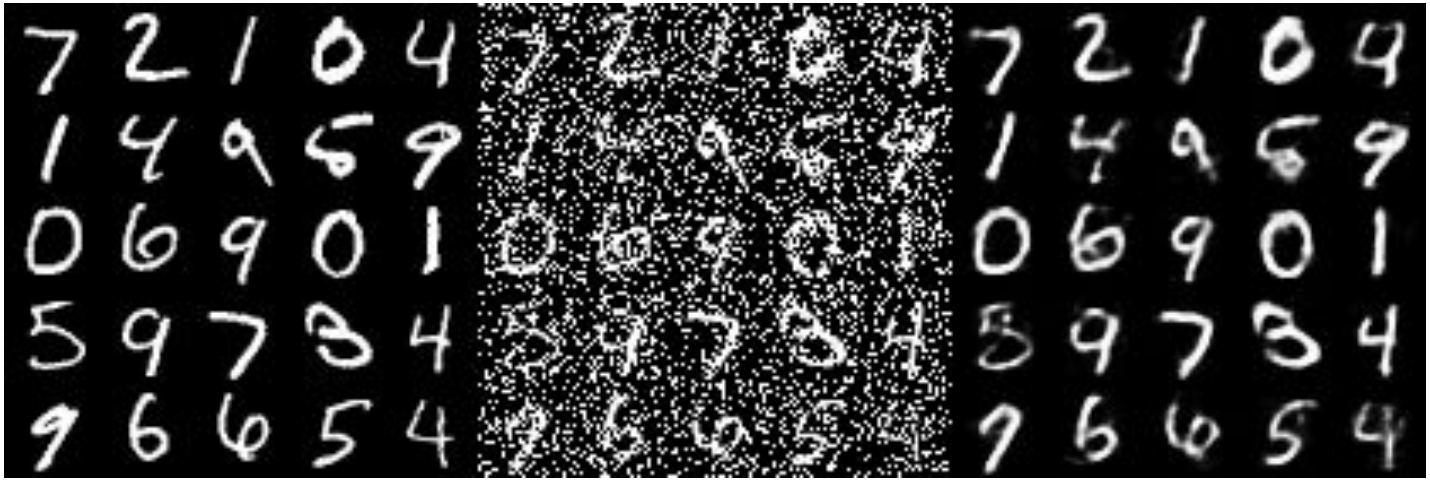
Denoising

- Denoising autoencoders take a partially corrupted input whilst training to recover the original undistorted input.

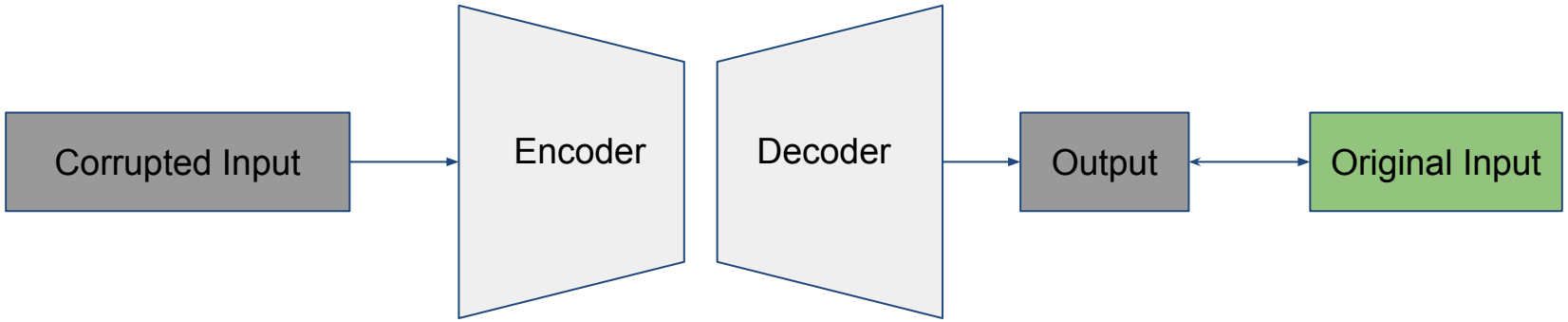
- This technique has been introduced with a specific approach to good representation.

- A good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input.

Denoising Autoencoder

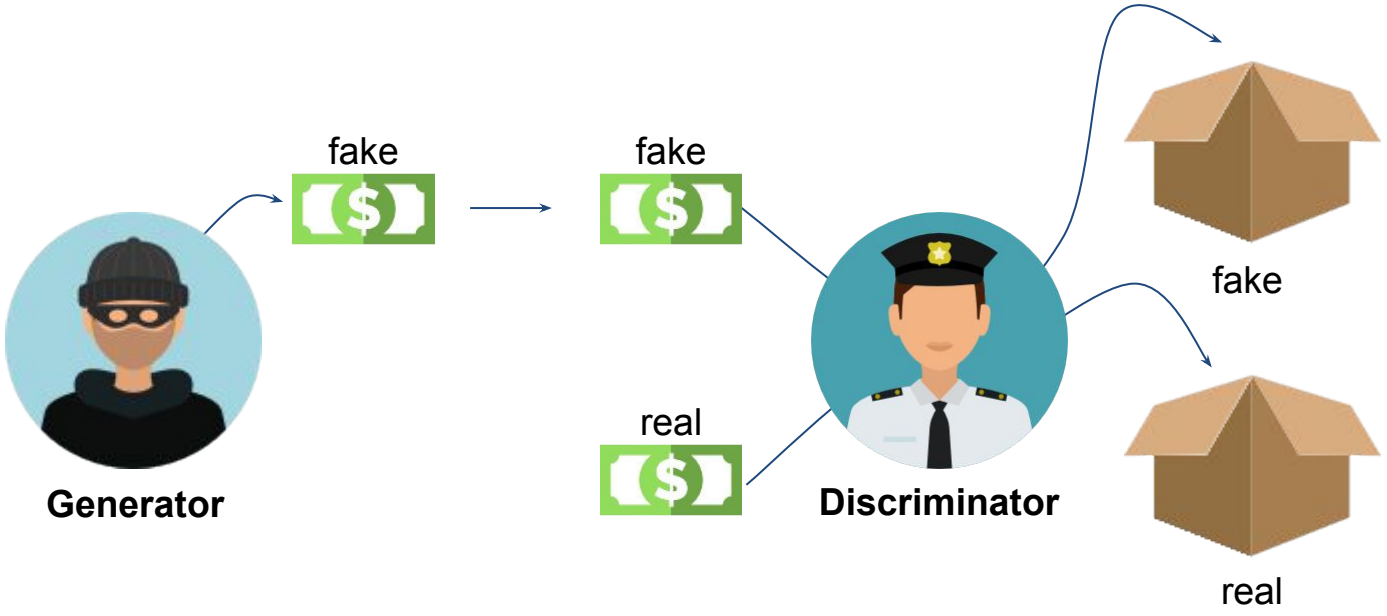


Original Input | Corrupted Input | Output



Generative Adversarial Network

What is GAN?



What is GAN?

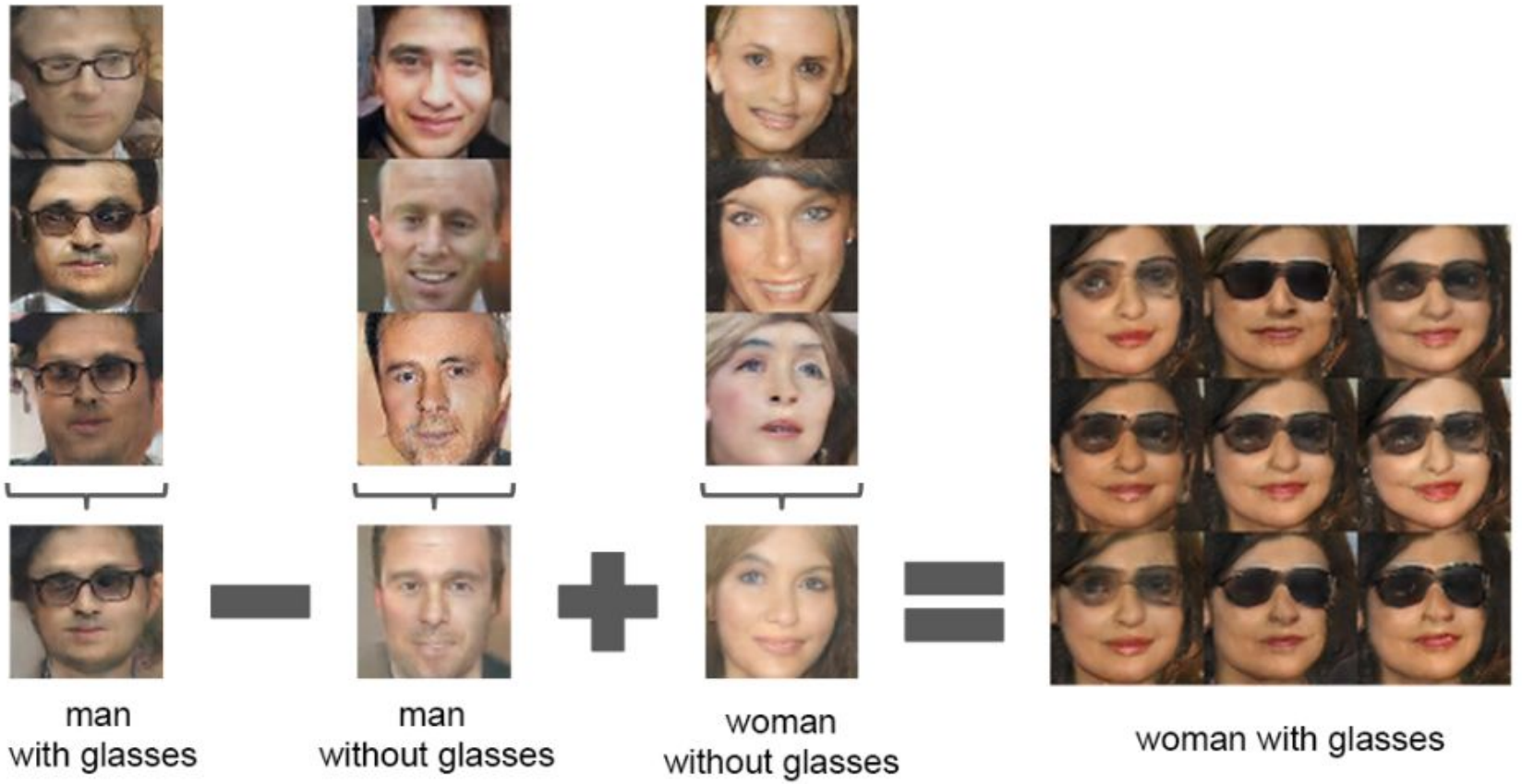
- Generative

- Adversarial

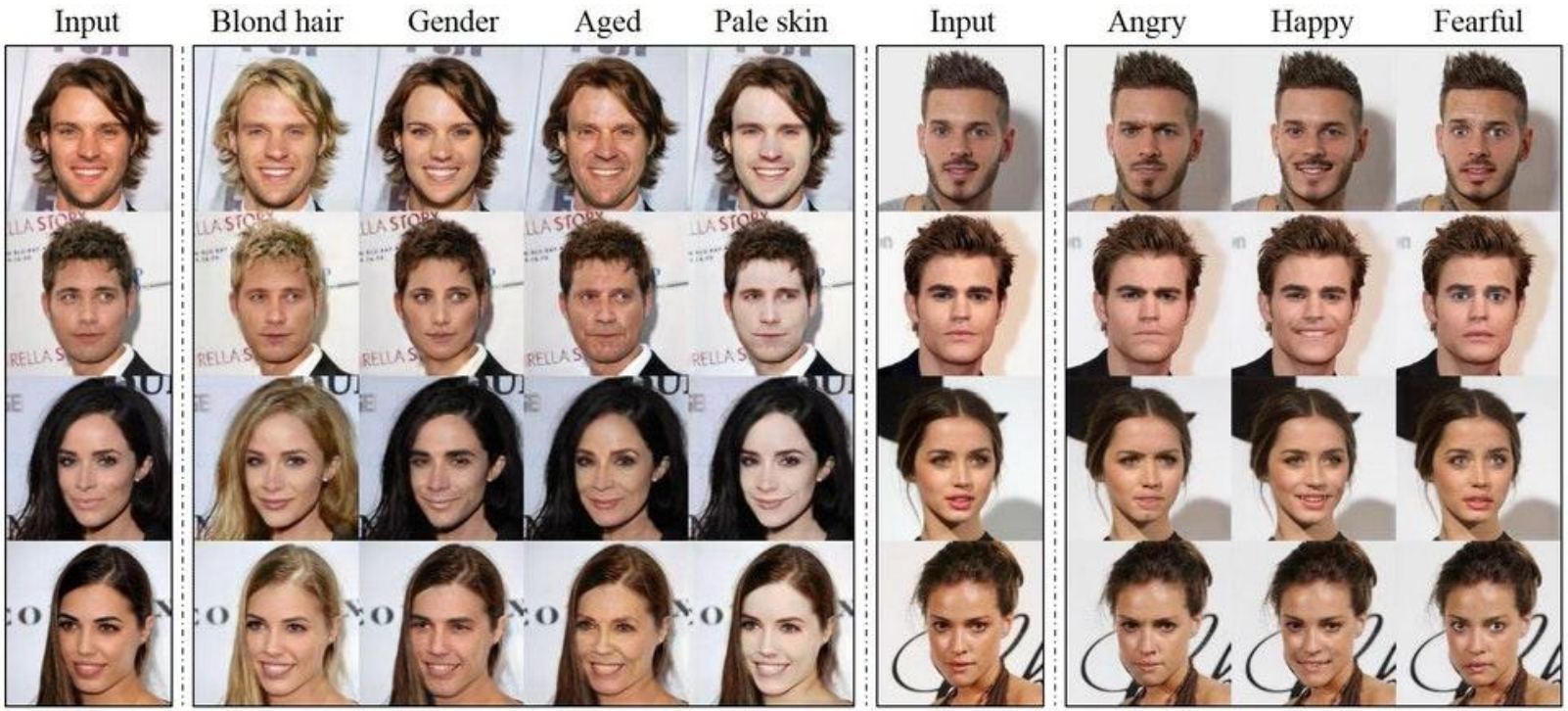
- Network

What is GAN?

What is GAN?



What is GAN?



What is GAN?

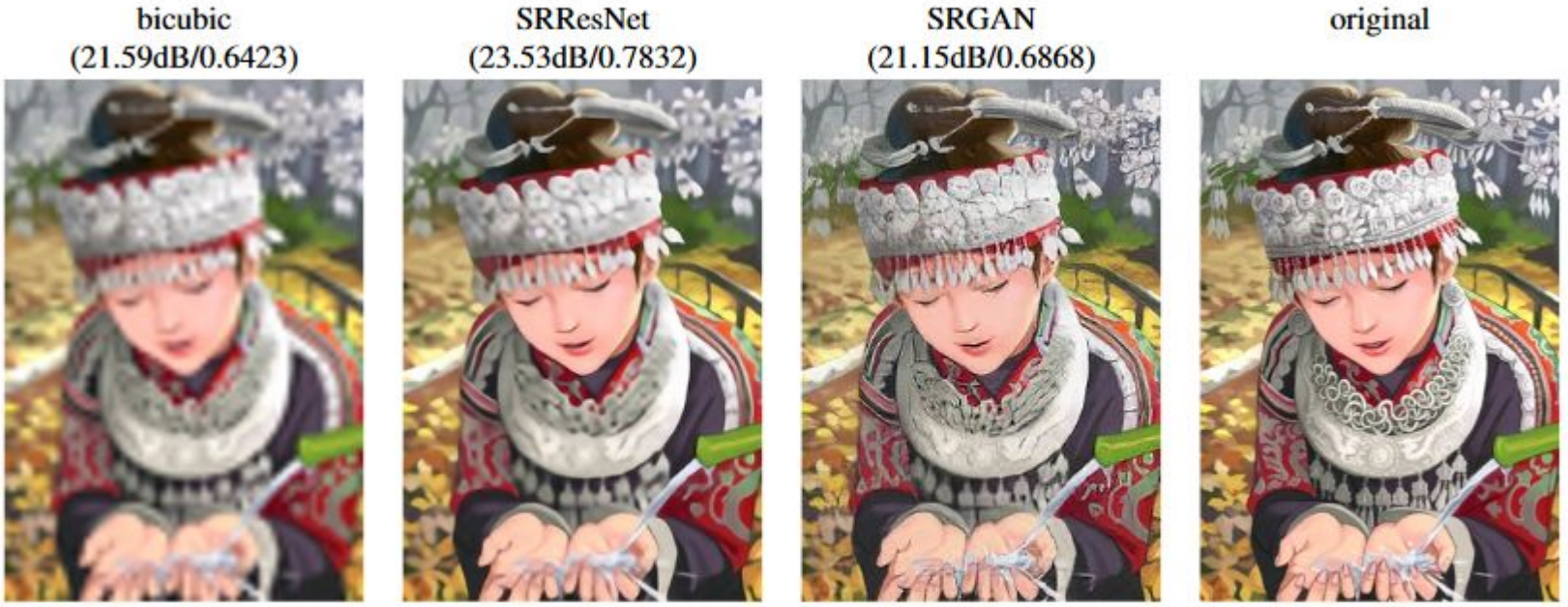
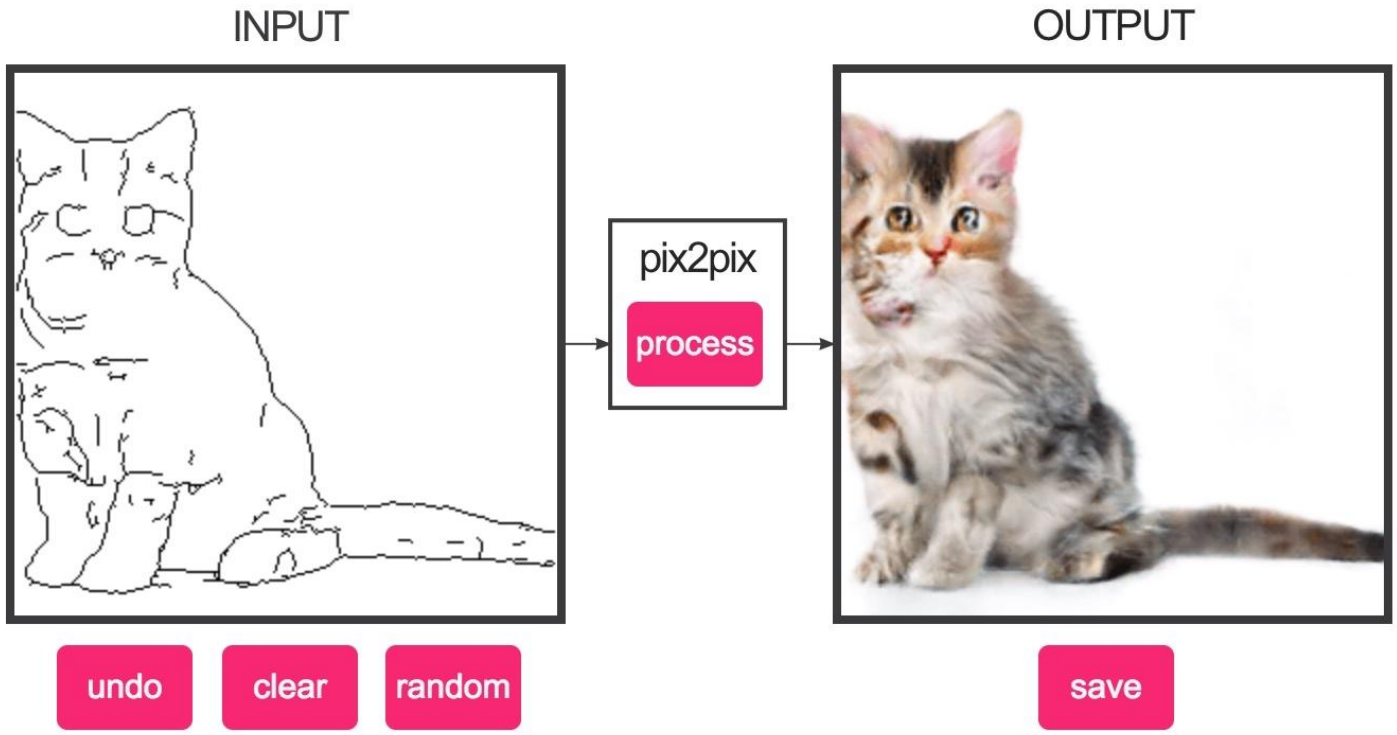
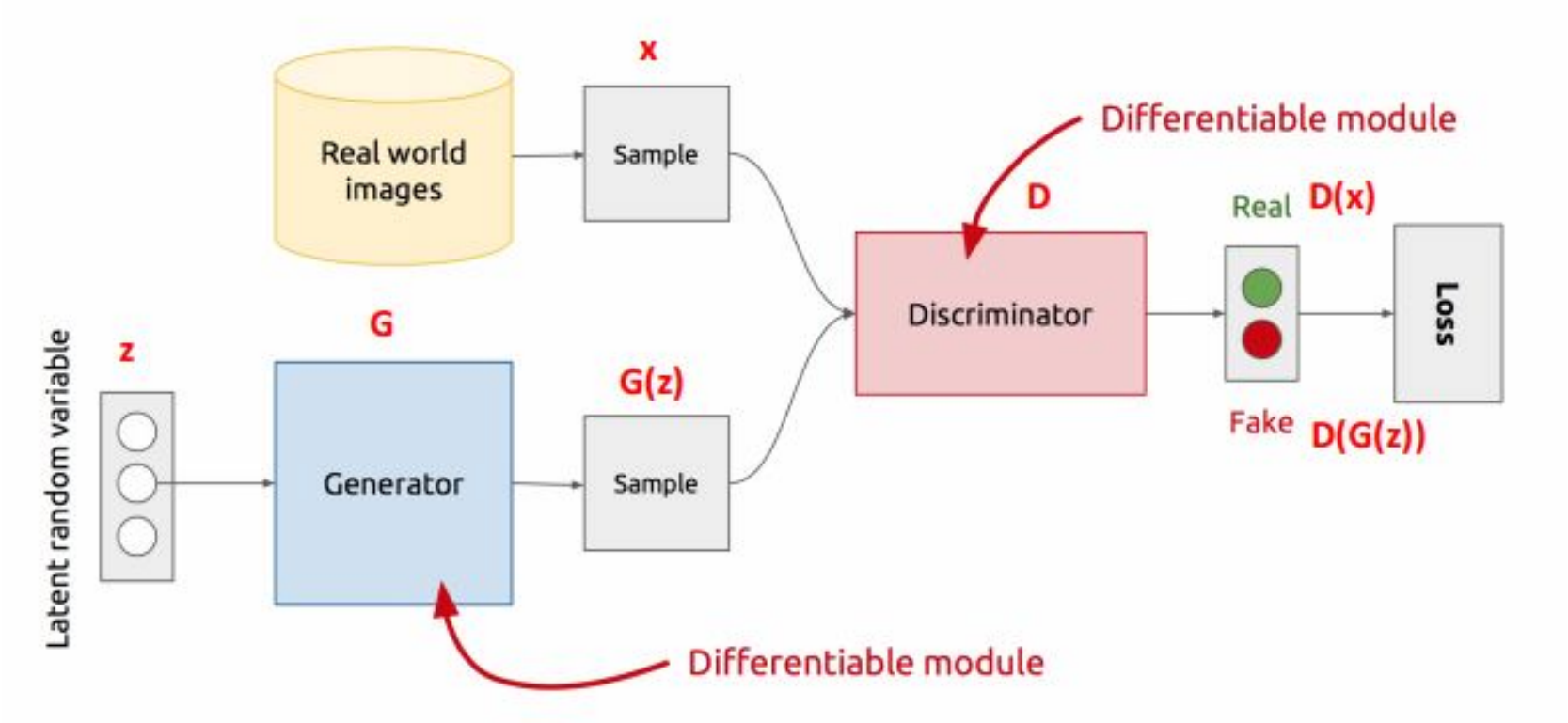


Figure 2: From left to right: bicubic interpolation, deep residual network optimized for MSE, deep residual generative adversarial network optimized for a loss more sensitive to human perception, original HR image. Corresponding PSNR and SSIM are shown in brackets. [4× upscaling]

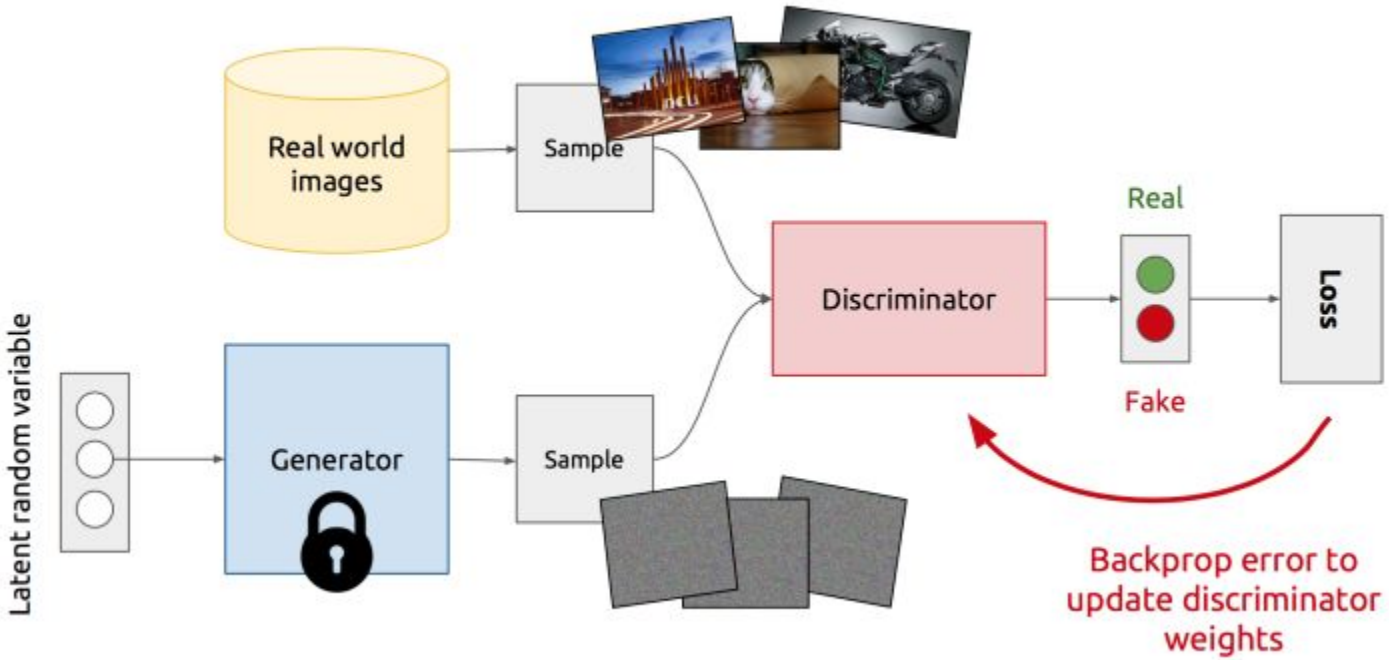
What is GAN?



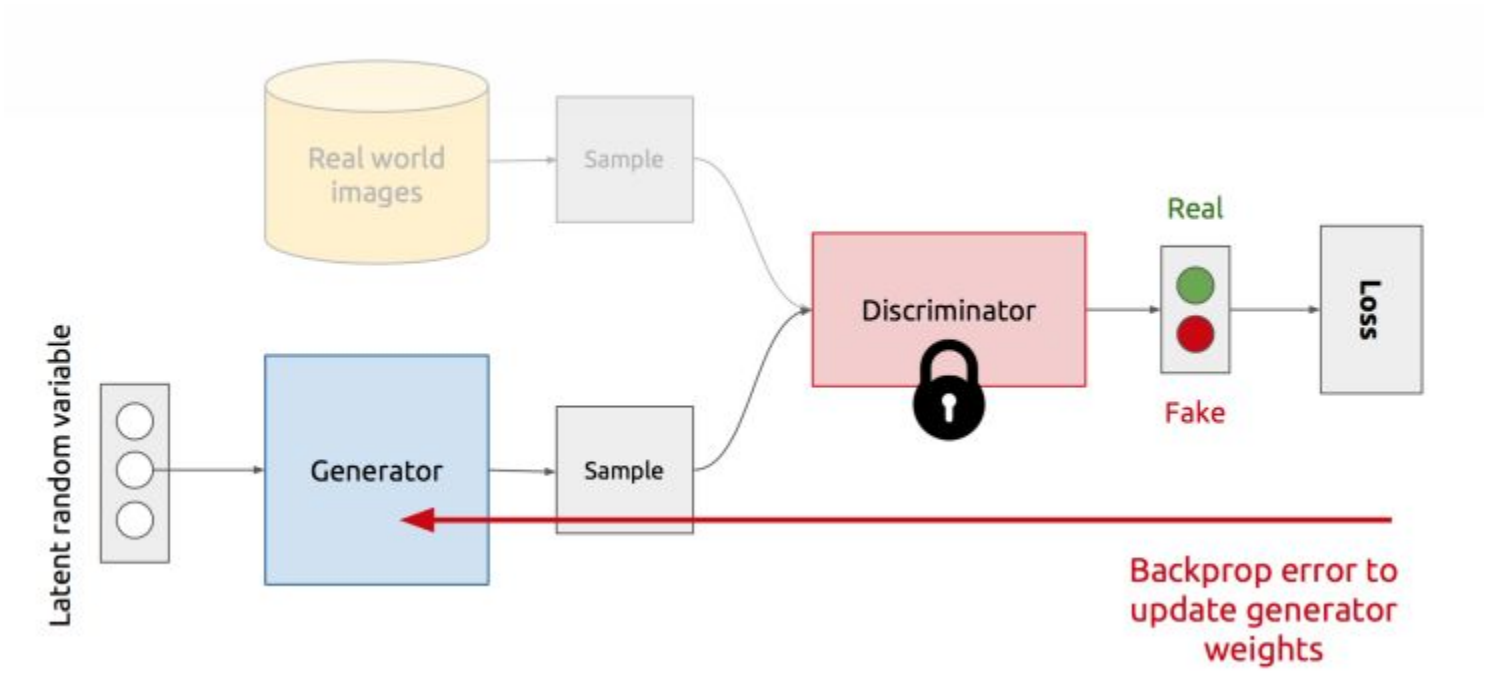
Architecture



Training 1 Discriminator



Training 2 Generator



Formulation

- **minimax game** :

$$\min_G \max_D V(D,G)$$

$$V(D,G) = \mathbb{E}_{x\sim p(x)}[\log D(x)] + \mathbb{E}_{z\sim q(z)}[\log(1 - D(G(z)))]$$

- **Nash equilibrium**
  게임이론에서 player가 상대가 전략을 바꾸지 않을 것이라는 전제 하에 전략을 바꿀 유인이 없는 상태. 가위 바위 보에서 모두가 ⅓ 확률로 랜덤하게 하나를 고르는 전략을 선택하는 상태가 대표적인 Nash equilibrium이다.

- GAN의 objective는 2 player game이기 때문에 generator가 실제 data와 아주 유사한 가짜 데이터를 생성하고 discriminator는 1/2확률로 그것이 가짜 혹은 진짜라고 판정하는 상태에서 Nash equilibrium이 된다.

Algorithm

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**

Discriminator updates

- 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

Generator updates

- 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.

Advantage of GANs

- Sampling(or generation) is straightforward
- Training doesn't involving Maximum Likelihood estimation
- Robust to Overfitting since Generator never sees the training data
- Empirically, GANs are good at capturing the modes of the distribution

Problems with GANs

- Not straightforward to compute P(x)
- Training is hard
  - Non-Convergence
  - Mode-Collapse
  - Balance of Networks

two player game

- Discriminator is trying to maximize its reward.
- Generator is trying to minimize Discriminator ’s reward.
- SGD was not designed to find the Nash equilibrium of a game.

Non-Convergence

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

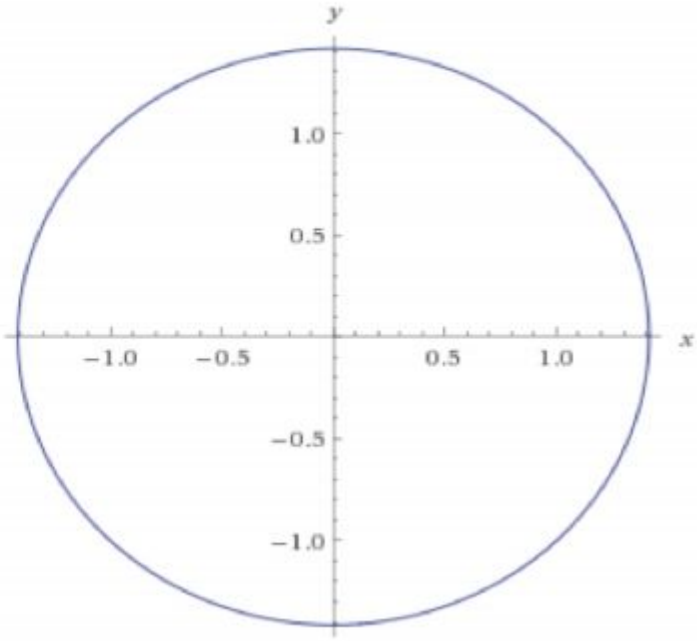
Let $V(x,y) = xy$

• State 1:	x > 0	y > 0	V > 0	Increase y	Decrease x
• State 2:	x < 0	y > 0	V < 0	Decrease y	Decrease x
• State 3:	x < 0	y < 0	V > 0	Decrease y	Increase x
• State 4 :	x > 0	y < 0	V < 0	Increase y	Increase x
• State 5:	x > 0	y > 0	V > 0 == State 1	Increase y	Decrease x

Non-Convergence

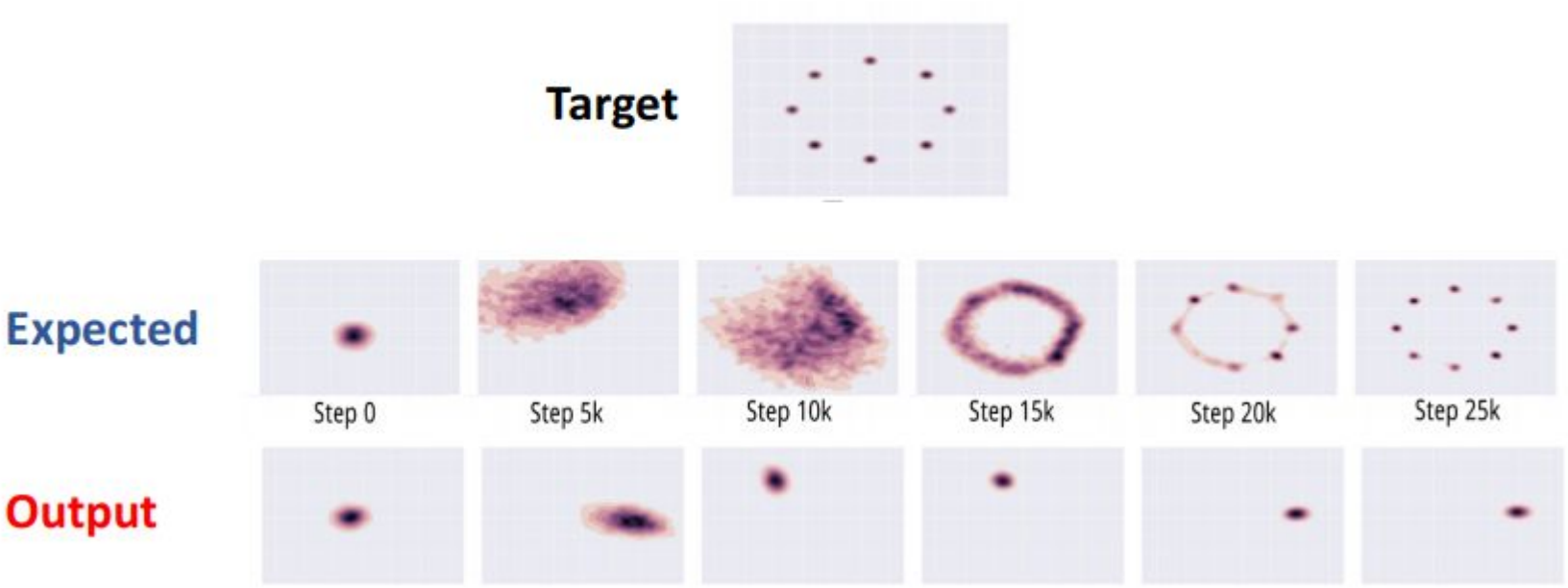
$$\min_x \max_y \; xy$$

- $\frac{\partial}{\partial x} = -y \quad \dots \quad \frac{\partial}{\partial y} = x$
- $\frac{\partial^2}{\partial y^2} = \frac{\partial}{\partial x} = -y$
- Differential equation's solution has sinusoidal terms
- Even with a small learning rate, it will not converge



Mode collapse

- Generator fails to output diverse samples



Mode collapse

Mode collapse