GAN

Generative Adversarial Networks

Why?

➢ 위키피디아에 GAN을 검색해보면?

- 2014년 6월에 Ian Goodfellow와 그의 동료들이 고안한 기계학습 프레임워크

Generative adversarial network

From Wikipedia, the free encyclopedia

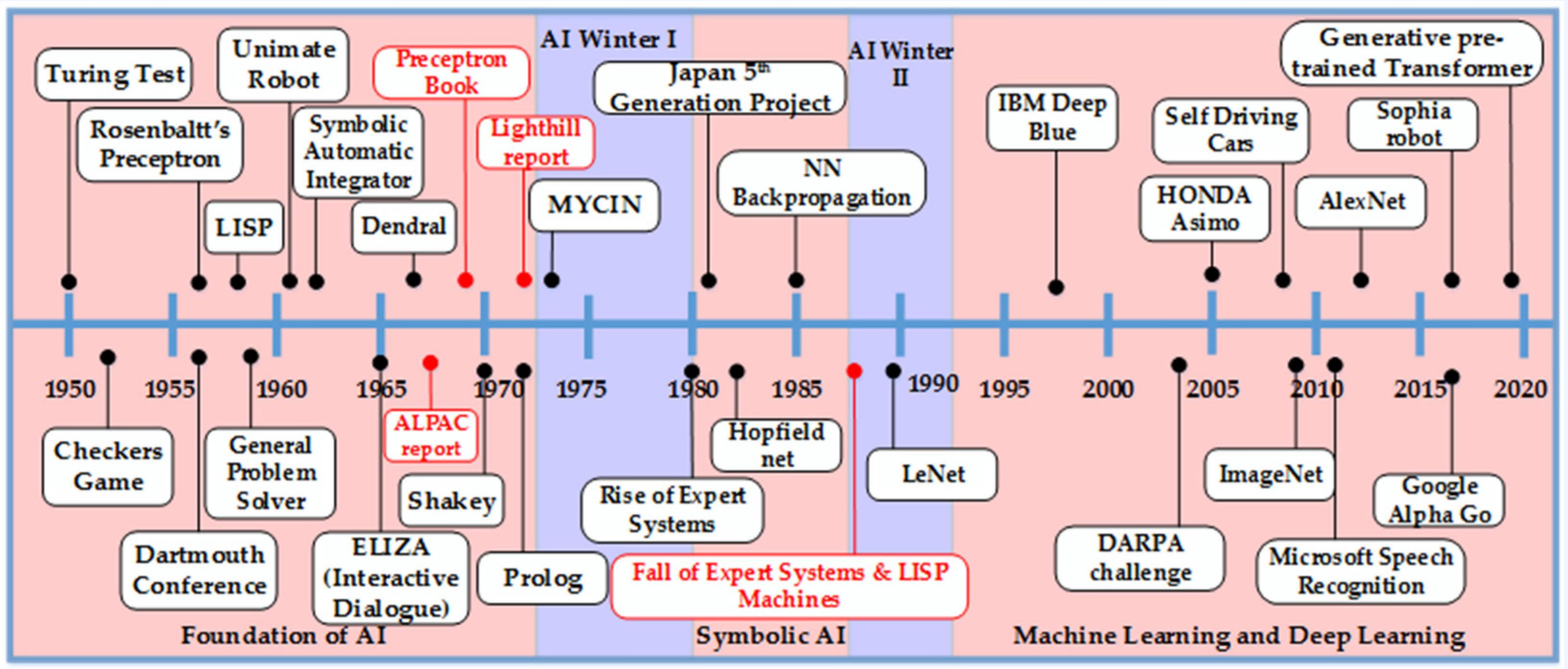
Not to be confused with Adversarial machine learning.

A **generative adversarial network** (**GAN**) is a class of machine learning frameworks designed by Ian Goodfellow and his colleagues in June 2014.[1] Two neural networks contest with each other in the form of a zero-sum game, where one agent's gain is another agent's loss.

Why?

➢ 2014년의 AI는?

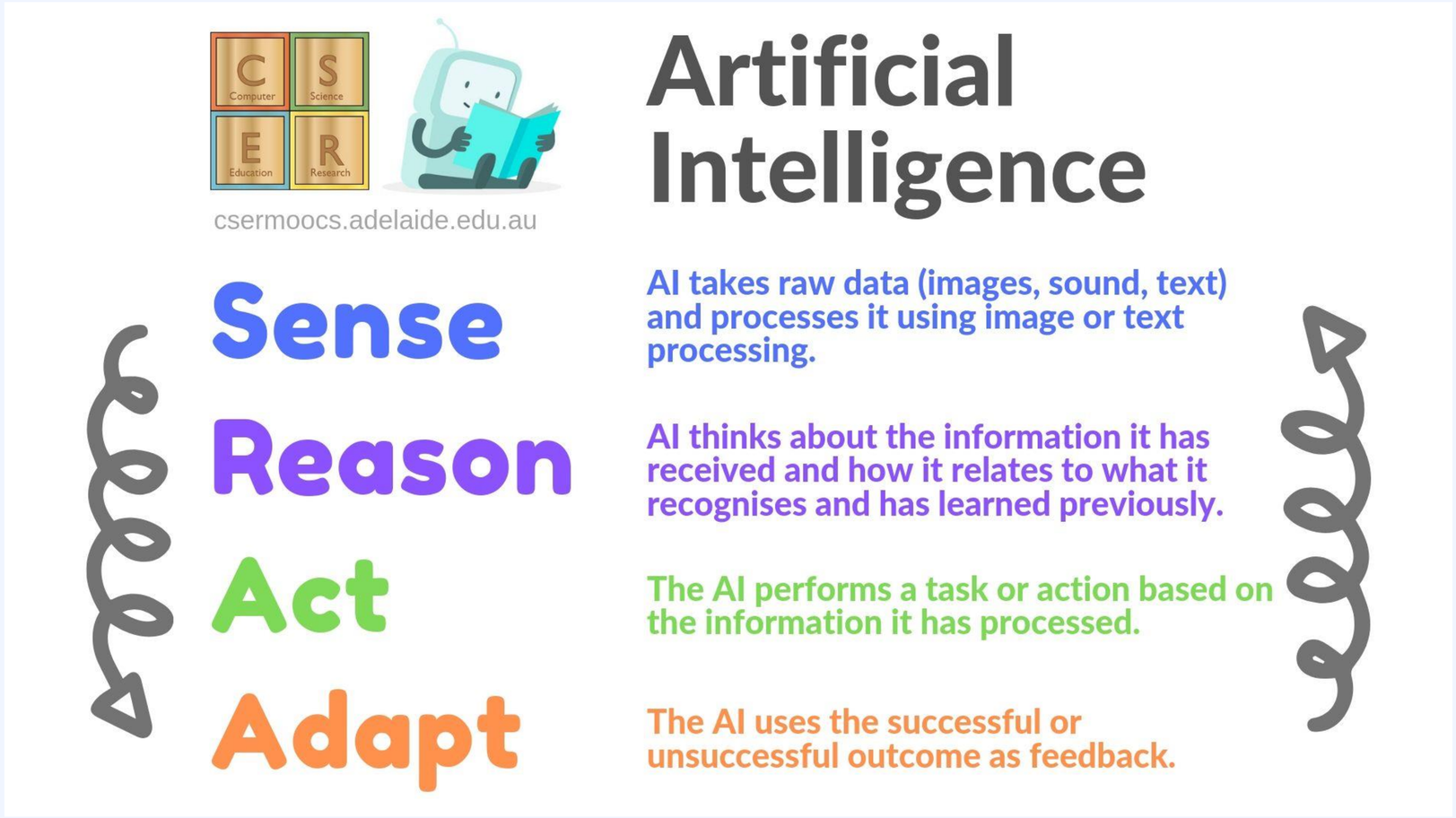
- 2012: AlexNet
- 2014: VGGNet, GoogLeNet
- 2016: AlphaGo



[출처] https://www.mdpi.com/2072-666X/12/6/665/htm

Why?

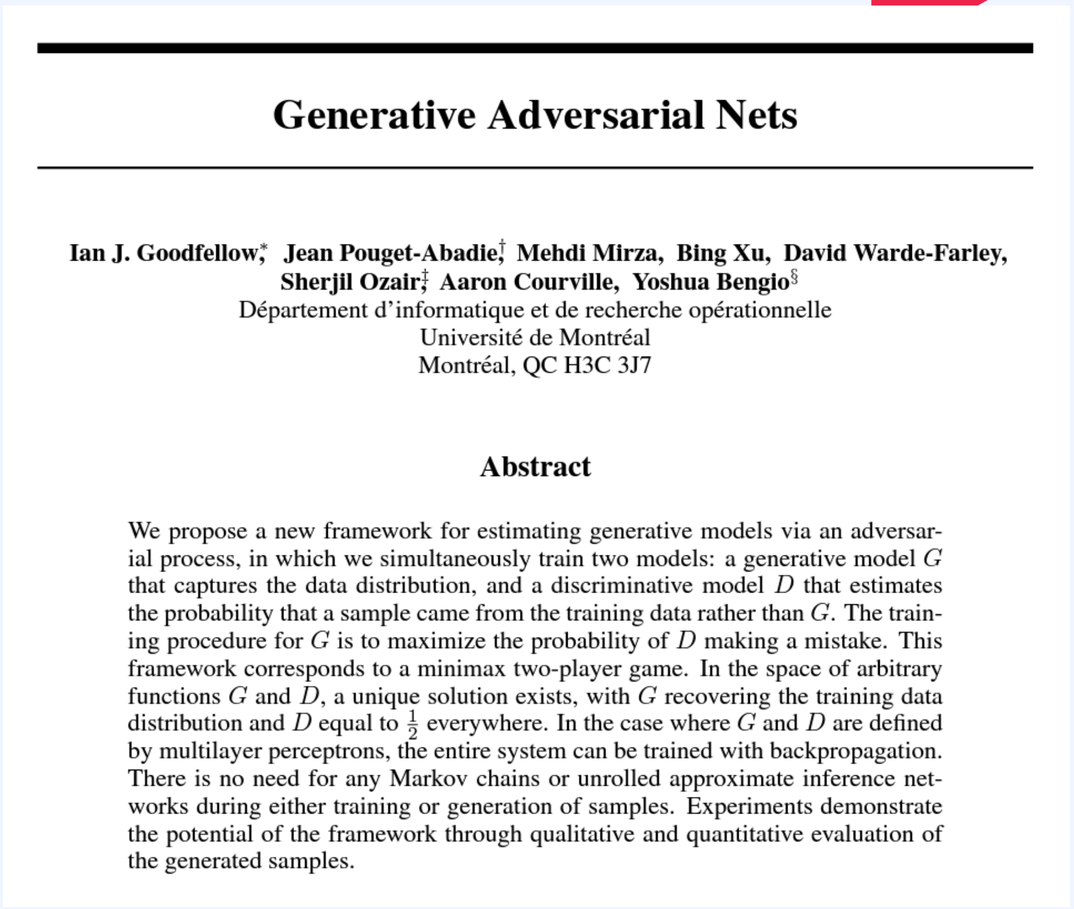
➢ What AI can do?



[출처] https://csermoocs.appspot.com/ai_primary/assets/img/ai_sense_adapt.jpg

What?

- Generative Adversarial Nets(Paper)
  - 생성가능한 적대적 신경망
  - Generator(생성자)와 Discriminator(판별자)로 구성
- Generator(생성자)
  - 입력: 노이즈
  - 출력: 가짜 데이터
- Discriminator(판별자)
  - 입력: 가짜 데이터 또는 진짜 데이터
  - 출력: 진위여부



Generative Adversarial Nets

Ian J. Goodfellow[*], **Jean Pouget-Abadie**[†], **Mehdi Mirza**, **Bing Xu**, **David Warde-Farley**, **Sherjil Ozair**[‡], **Aaron Courville**, **Yoshua Bengio**[§]
Département d'informatique et de recherche opérationnelle
Université de Montréal
Montréal, QC H3C 3J7

Abstract

We propose a new framework for estimating generative models via an adversarial process, in which we simultaneously train two models: a generative model G that captures the data distribution, and a discriminative model D that estimates the probability that a sample came from the training data rather than G. The training procedure for G is to maximize the probability of D making a mistake. This framework corresponds to a minimax two-player game. In the space of arbitrary functions G and D, a unique solution exists, with G recovering the training data distribution and D equal to $\frac{1}{2}$ everywhere. In the case where G and D are defined by multilayer perceptrons, the entire system can be trained with backpropagation. There is no need for any Markov chains or unrolled approximate inference networks during either training or generation of samples. Experiments demonstrate the potential of the framework through qualitative and quantitative evaluation of the generated samples.

What?

➢ Generative Adversarial Nets(Paper)

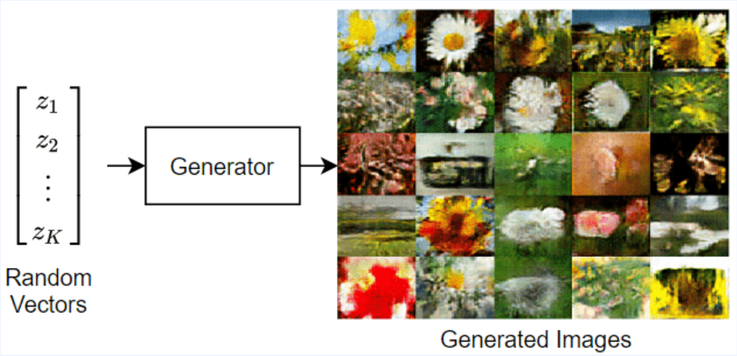
- 생성가능한 적대적 신경망
- Generator(생성자)와 Discriminator(판별자)로 구성

➢ Generator(생성자)

- 입력: 노이즈
- 출력: 가짜 데이터

➢ Discriminator(판별자)

- 입력: 가짜 데이터 또는 진짜 데이터
- 출력: 진위여부



[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

What?

➢ Generative Adversarial Nets(Paper)

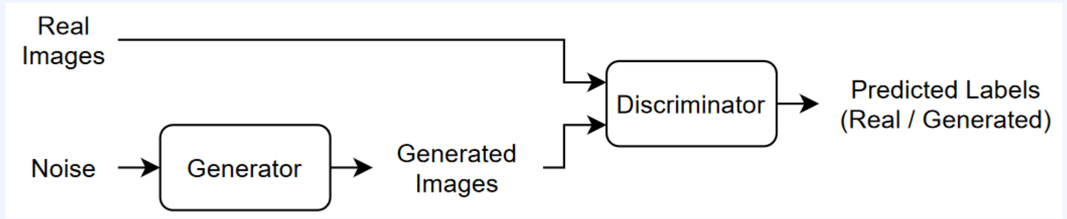
- 생성가능한 적대적 신경망
- Generator(생성자)와 Discriminator(판별자)로 구성

➢ Generator(생성자)

- 입력: 노이즈
- 출력: 가짜 데이터

➢ Discriminator(판별자)

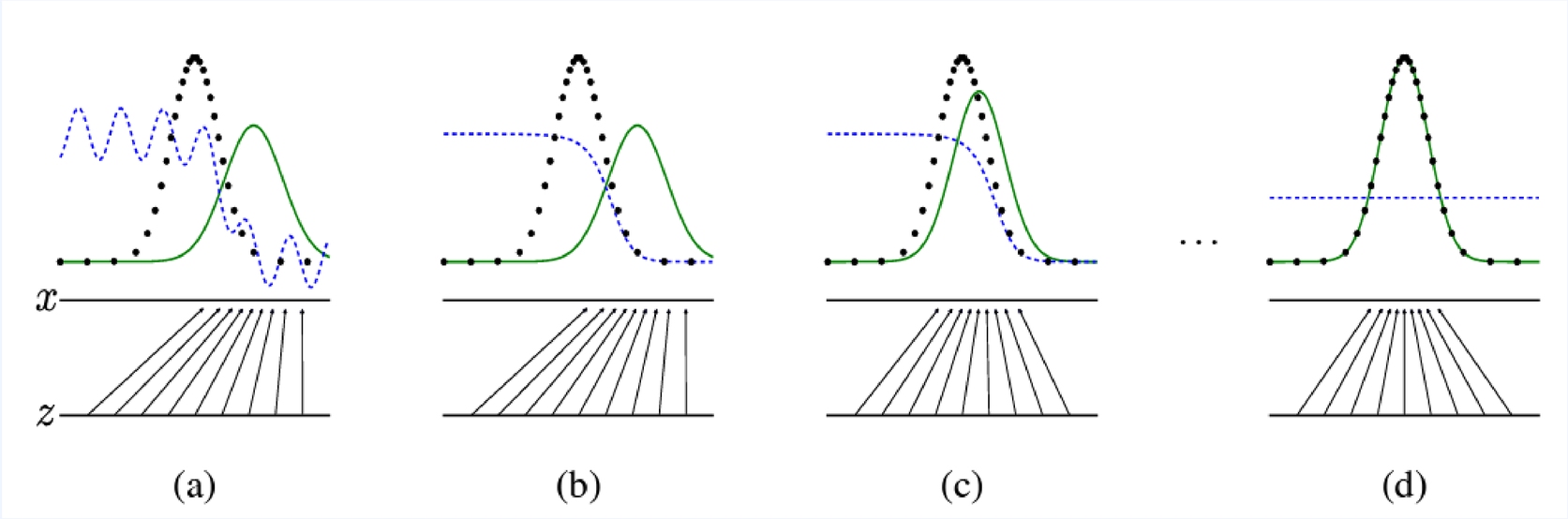
- 입력: 가짜 데이터 또는 진짜 데이터
- 출력: 진위여부



[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

What?

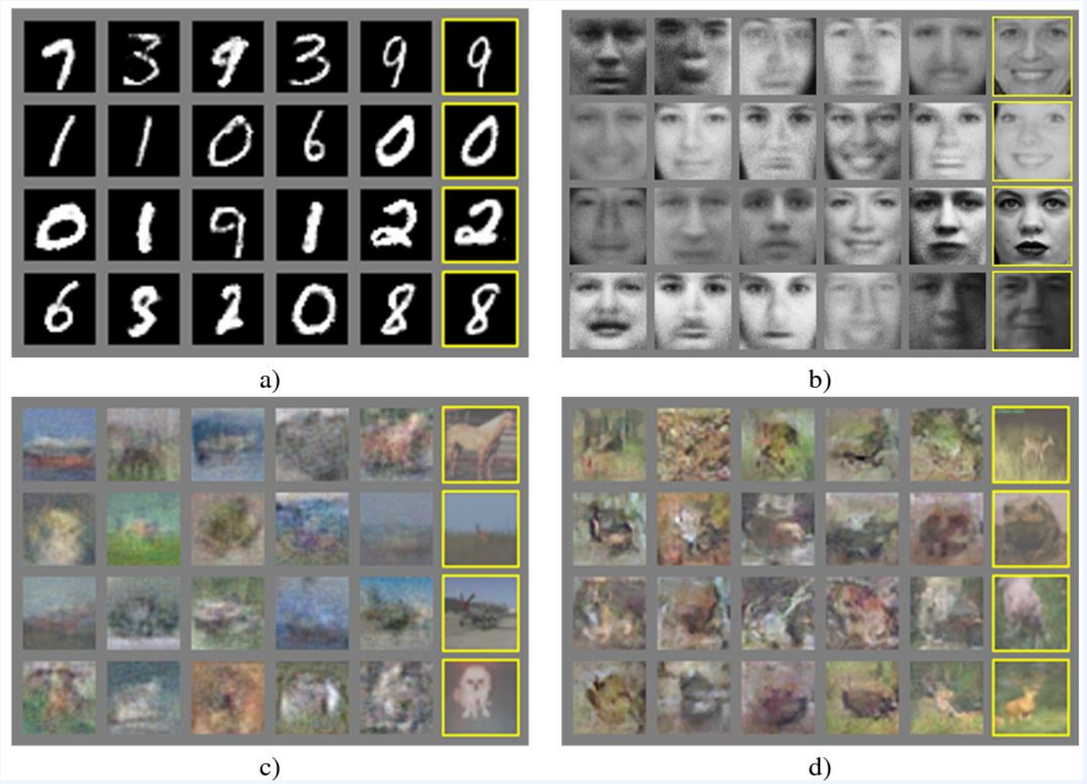
- 검정색 : 실제 데이터의 분포
- 초록색: 생성자가 생성한 데이터의 분포
- 파란색: 판별자가 판단하는 값



What?

➢ GAN이 생성한 이미지

- a : MNIST
- b: TFT(Toronto Face Database)
- c, d: CIFAR10



How?

- D and G play the following two-player minimax game with value function V (G,D)

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

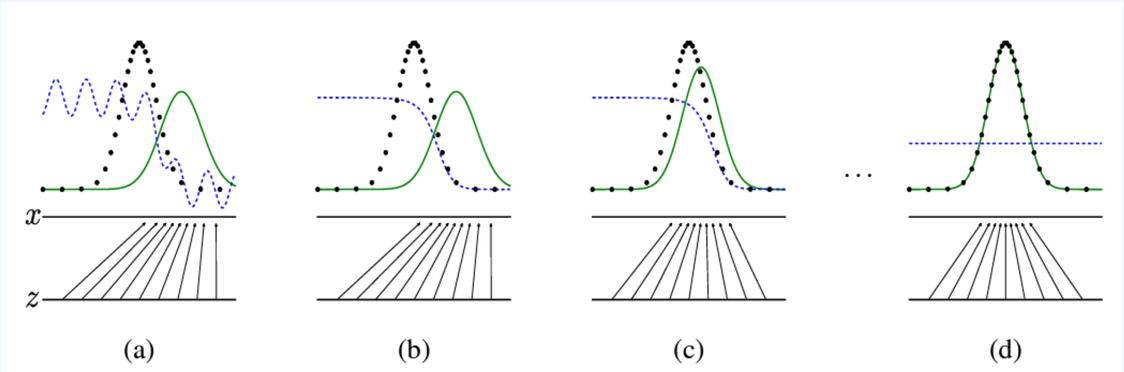
How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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

실제 데이터에 대한
확률분포에서 샘플링한
데이터

임의의 노이즈*에서
샘플링한 데이터
* : 일반적으로 가우시안 노이즈 사용



How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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

실제 데이터가
진짜라고 판단할 경우 1,
가짜라고 판단할 경우 0을
출력

가짜 데이터가
진짜라고 판단할 경우 1,
가짜라고 판단할 경우 0을
출력

How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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

max

$D(x) = 1$

실제 데이터를 진짜라고 판단

max

$D(G(z)) = 0$

가짜 데이터를 가짜라고 판단

How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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

min

$D(x) = 0$

실제 데이터를 가짜라고 판단

min

$D(G(z)) = 1$

가짜 데이터를 진짜라고 판단

How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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.

How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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.

How?

➢ D and G play the following two-player minimax game with value function V (G,D)

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.

Conclusion

➢ GAN의 한계

- 학습시키는 것이 어렵다
  - 생성자와 판별자의 실력이 비슷한 경우에 두 모델이 균형있게 학습이 진행되는데, 그렇지 않은 경우 학습이 잘 진행되지 않음
- 사용된 생성자의 결과물 형태가 어떠한 과정을 통해 나왔는지 알 수 없다
- 새롭게 만들어진 데이터가 얼마나 정확한지 객관적으로 판단하기 어렵다
  - 주관적인 판단 필요

Summary

- GAN은 _____와 _____가 서로 적대적인 관계를 갖으며 학습을 하는 신경망이다.
- GAN의 _____ 은 _____를 입력하여 가짜 데이터를 생성한다.
- GAN의 _____ 은 _____ 를 입력하여 데이터의 진위여부를 판별한다.
- GAN에서 동일한 value function에 대해 _____은 값을 최대화하는 방향으로 학습하고, _____은 값을 최소화하는 방향으로 학습한다.

Summary - Answer

- GAN은 generator 와 discriminator 가 서로 적대적인 관계를 갖으며 학습을 하는 신경망이다.
- GAN의 generator 은 noise 를 입력하여 가짜 데이터를 생성한다.
- GAN의 discriminator 은 real or fake data 를 입력하여 데이터의 진위여부를 판별한다.
- GAN에서 동일한 value function에 대해 discriminator 은 값을 최대화하는 방향으로 학습하고, generator 은 값을 최소화하는 방향으로 학습한다.

GAN

DCGAN

Generate…?

DCGAN

➢ Paper: Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks,

Alec Radford, Luke Metz, Soumith Chintala, ICLR 2016

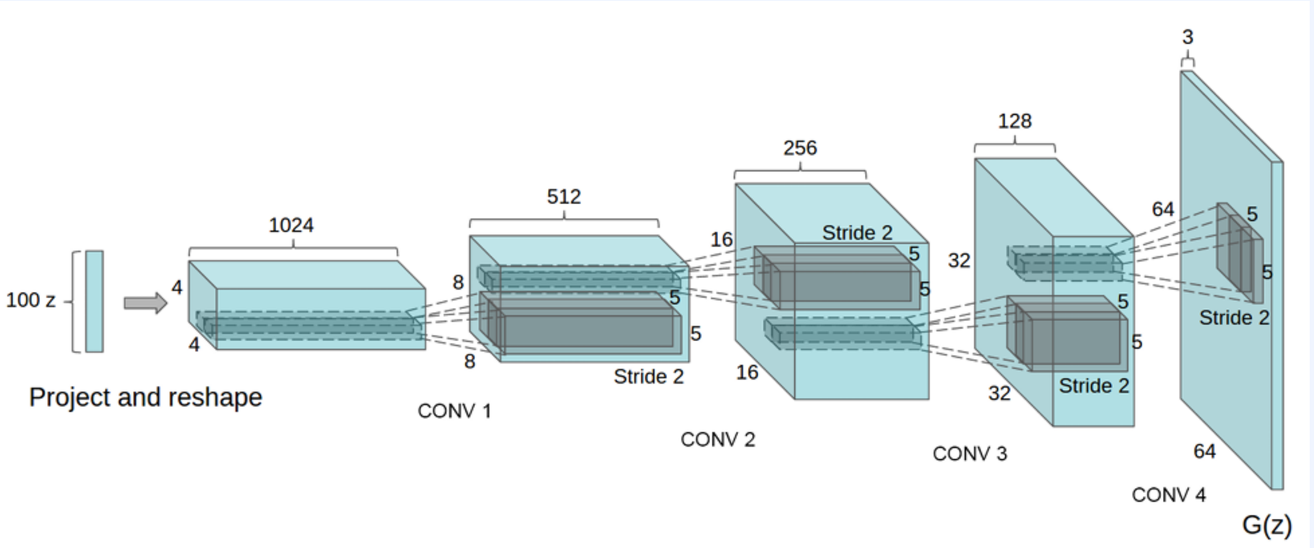
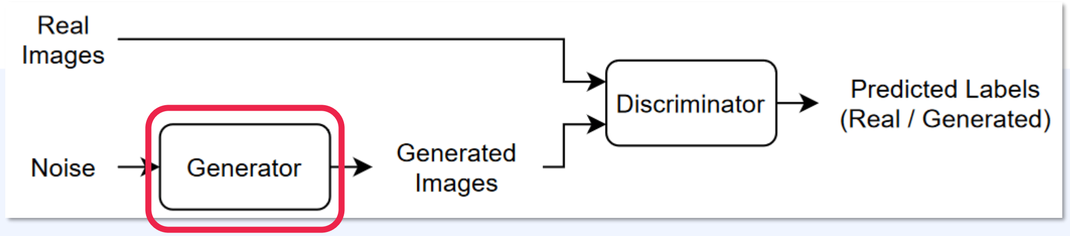
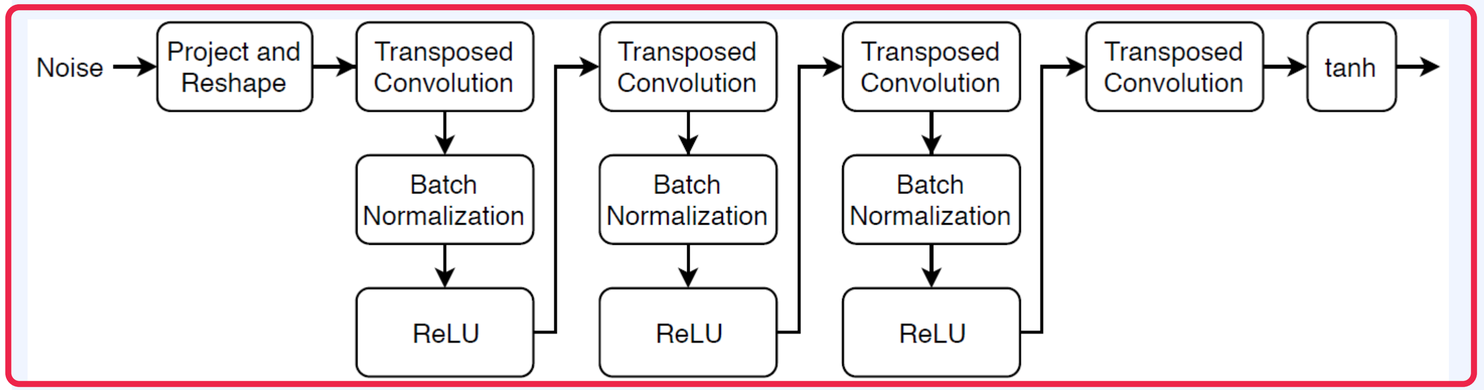


Figure 1: DCGAN generator used for LSUN scene modeling. A 100 dimensional uniform distribution Z is projected to a small spatial extent convolutional representation with many feature maps. A series of four fractionally-strided convolutions (in some recent papers, these are wrongly called deconvolutions) then convert this high level representation into a 64×64 pixel image. Notably, no fully connected or pooling layers are used.

DCGAN

➢ Architecture

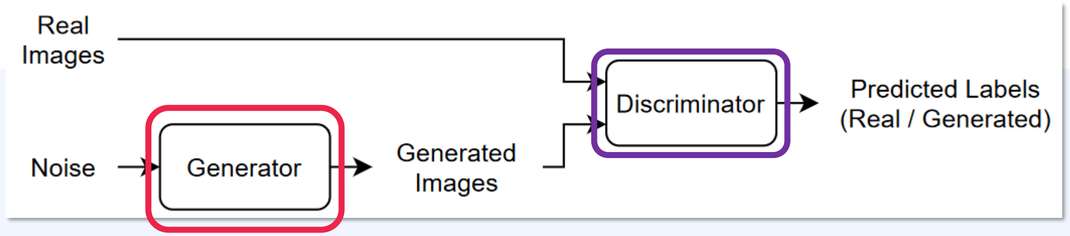


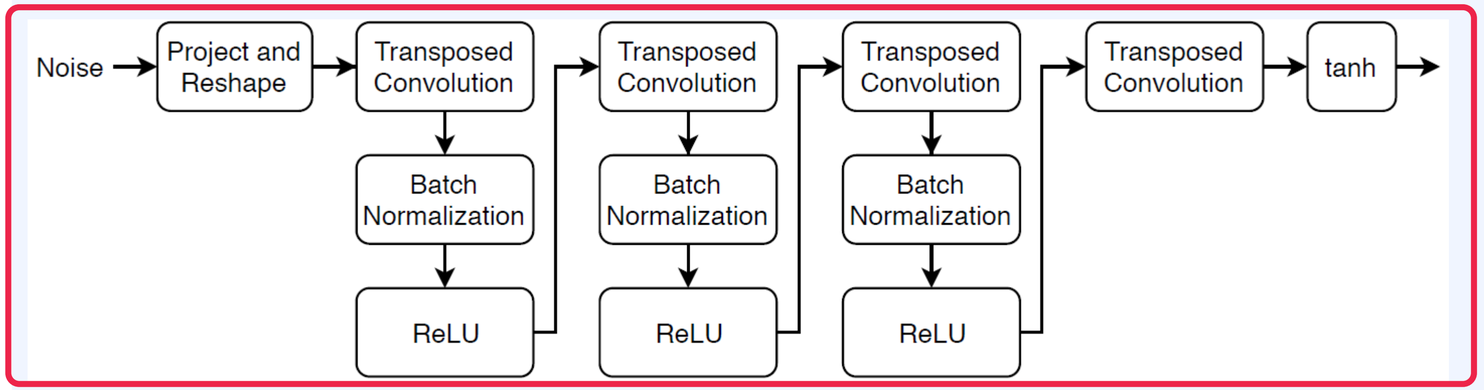


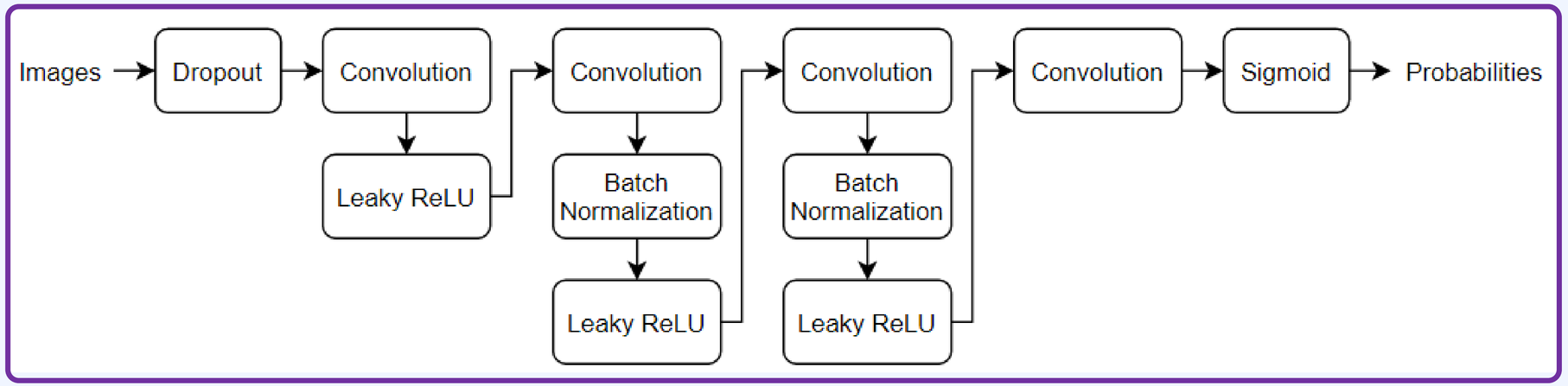
[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

DCGAN

➢ Architecture







[출처] https://www.mathworks.com/help/deeplearning/ug/train-generative-adversarial-network.html

DCGAN

➢ Architecture

Architecture guidelines for stable Deep Convolutional GANs

- Replace any pooling layers with strided convolutions (discriminator) and fractional-strided convolutions (generator).
- Use batchnorm in both the generator and the discriminator.
- Remove fully connected hidden layers for deeper architectures.
- Use ReLU activation in generator for all layers except for the output, which uses Tanh.
- Use LeakyReLU activation in the discriminator for all layers.

DCGAN

Figure 2: Generated bedrooms after one training pass through the dataset. Theoretically, the model could learn to memorize training examples, but this is experimentally unlikely as we train with a small learning rate and minibatch SGD. We are aware of no prior empirical evidence demonstrating memorization with SGD and a small learning rate.

1 epoch

Figure 3: Generated bedrooms after five epochs of training. There appears to be evidence of visual under-fitting via repeated noise textures across multiple samples such as the base boards of some of the beds.

5 epochs

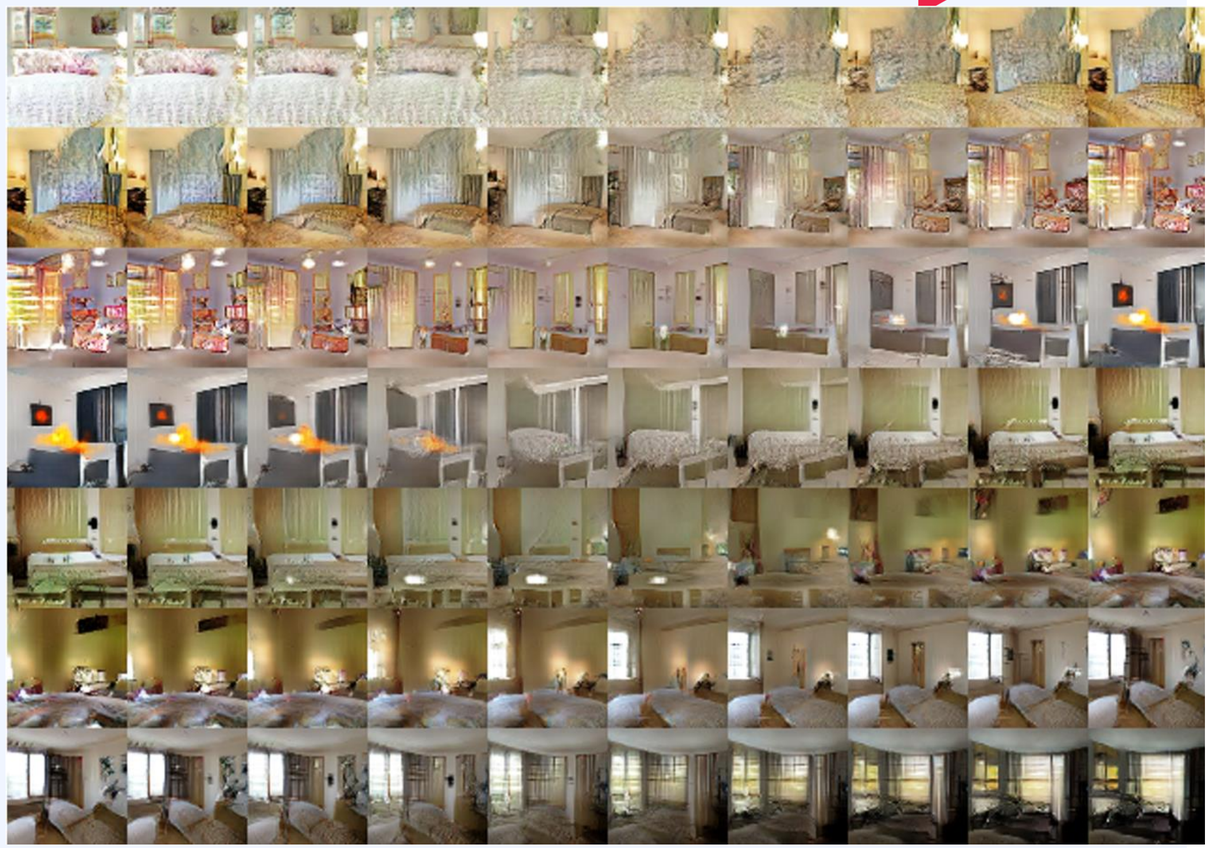
DCGAN

- INVESTIGATING AND VISUALIZING THE INTERNALS OF THE NETWORKS
  - WALKING IN THE LATENT SPACE
    - understand the landscape of the latent space
  - VISUALIZING THE DISCRIMINATOR FEATURES
  - MANIPULATING THE GENERATOR REPRESENTATION
    - FORGETTING TO DRAW CERTAIN OBJECTS
    - VECTOR ARITHMETIC ON FACE SAMPLES

DCGAN

➢ INVESTIGATING AND VISUALIZING THE INTERNALS OF THE NETWORKS

- **WALKING IN THE LATENT SPACE**
  - understand the landscape of the latent space
- VISUALIZING THE DISCRIMINATOR FEATURES
- MANIPULATING THE GENERATOR REPRESENTATION
  - FORGETTING TO DRAW CERTAIN OBJECTS
  - VECTOR ARITHMETIC ON FACE SAMPLES



DCGAN

➢ INVESTIGATING AND VISUALIZING THE INTERNALS OF THE NETWORKS

- WALKING IN THE LATENT SPACE
  - understand the landscape of the latent space
- **VISUALIZING THE DISCRIMINATOR FEATURES**
- MANIPULATING THE GENERATOR REPRESENTATION
  - FORGETTING TO DRAW CERTAIN OBJECTS
  - VECTOR ARITHMETIC ON FACE SAMPLES

Random filters Trained filters

DCGAN

➢ INVESTIGATING AND VISUALIZING THE INTERNALS OF THE NETWORKS

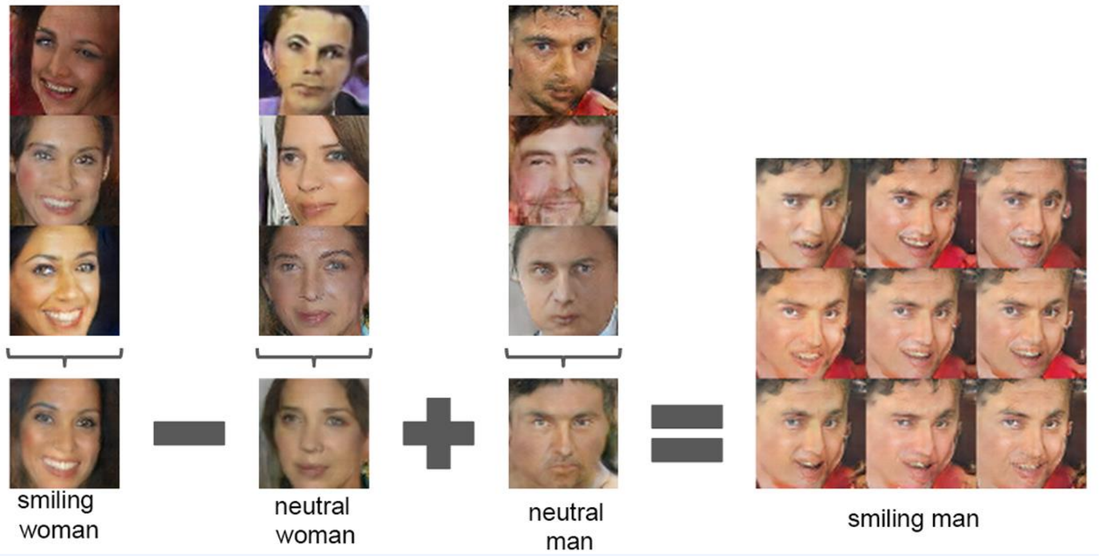
- WALKING IN THE LATENT SPACE
  - understand the landscape of the latent space
- VISUALIZING THE DISCRIMINATOR FEATURES
- **MANIPULATING THE GENERATOR REPRESENTATION**
  - FORGETTING TO DRAW CERTAIN OBJECTS
  - VECTOR ARITHMETIC ON FACE SAMPLES

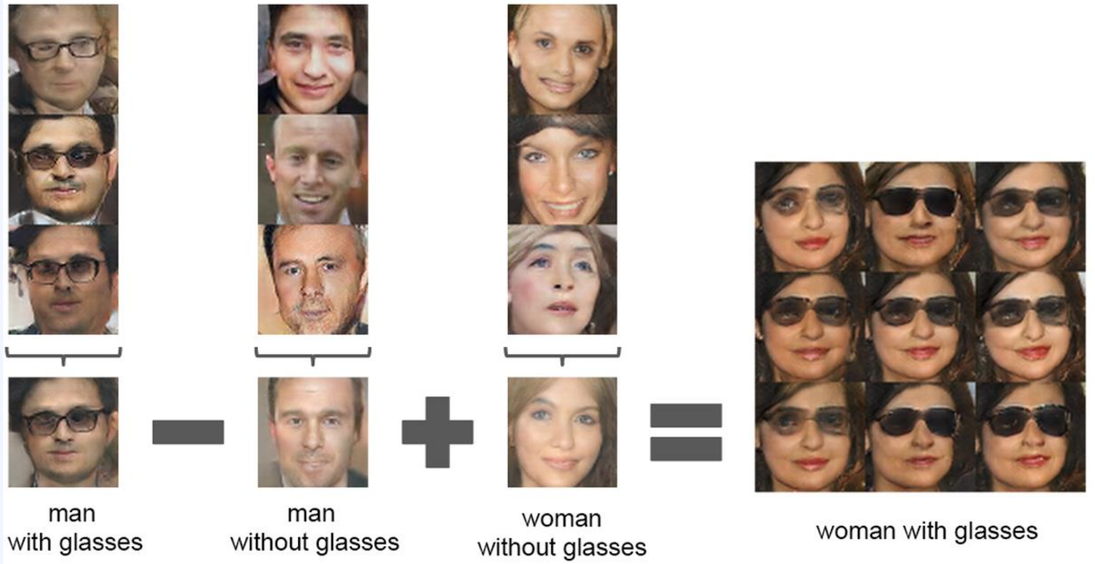
Figure 6: Top row: un-modified samples from model. Bottom row: the same samples generated with dropping out "window" filters. Some windows are removed, others are transformed into objects with similar visual appearance such as doors and mirrors. Although visual quality decreased, overall scene composition stayed similar, suggesting the generator has done a good job disentangling scene representation from object representation. Extended experiments could be done to remove other objects from the image and modify the objects the generator draws.

DCGAN

➢ INVESTIGATING AND VISUALIZING THE INTERNALS OF THE

- WALKING IN THE LATENT SPACE
  - understand the landscape of the latent space
- VISUALIZING THE DISCRIMINATOR FEATURES
- **MANIPULATING THE GENERATOR REPRESENTATION**
  - FORGETTING TO DRAW CERTAIN OBJECTS
  - VECTOR ARITHMETIC ON FACE SAMPLES





DCGAN

➢ INVESTIGATING AND VISUALIZING THE INTERNALS OF THE NETWORKS

- WALKING IN THE LATENT SPACE
  - understand the landscape of the latent space
- VISUALIZING THE DISCRIMINATOR FEATURES
- **MANIPULATING THE GENERATOR REPRESENTATION**
  - FORGETTING TO DRAW CERTAIN OBJECTS
  - VECTOR ARITHMETIC ON FACE SAMPLES

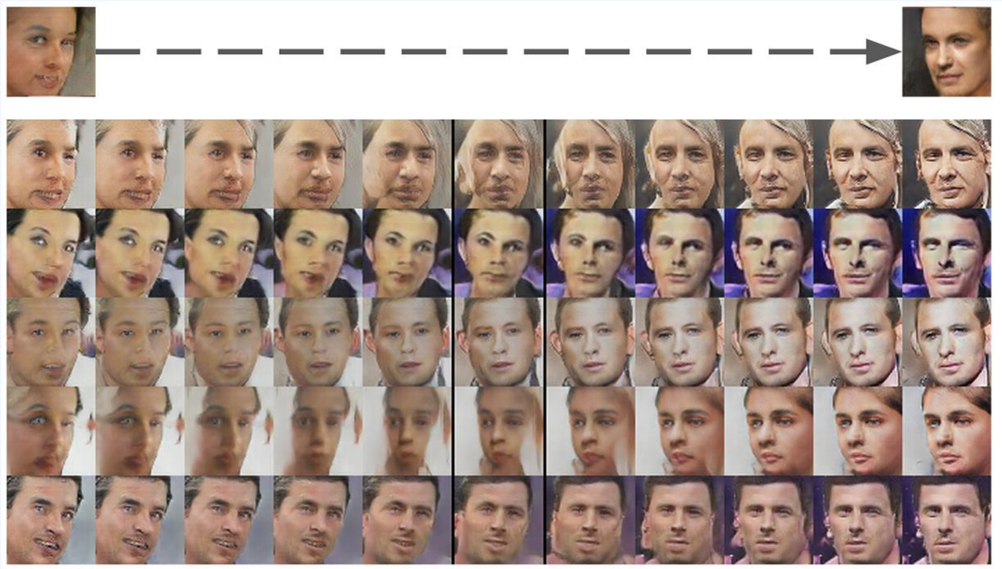


Figure 8: A ”turn” vector was created from four averaged samples of faces looking left vs looking right. By adding interpolations along this axis to random samples we were able to reliably transform their pose.

Summary

➢ DCGAN

- Generator는 training set을 단순히 암기하거나 copy하지 않는다.
  - Generator의 input z를 조금씩 변경하여 생성한 이미지도 완전히 다른 이미지가 나오는 것이 아니라 조금씩 변경되는 형태로 생성되는 것으로 보아, latent space의 어떤 점과 생성된 값이 1:1로 매칭되는 형태가 아님을 알 수 있었다.
- Generator에서 생성된 filter는 특정 object를 생성하는 역할을 수행한다.
  - 예를 들어서 어떤 filter는 창문을 생성하는 filter라면 그 filter를 제외하고 생성한 이미지에는 창문이 존재하지 않는다.
- DCGAN은 벡터 산술 연산이 가능하고, 그 결과 생성되는 이미지를 조절할 수 있다.
  - 웃는 여성 – 무표정 여성 + 무표정 남성 = 웃는 남성의 연산이 가능하다.

GAN

GAN의 활용

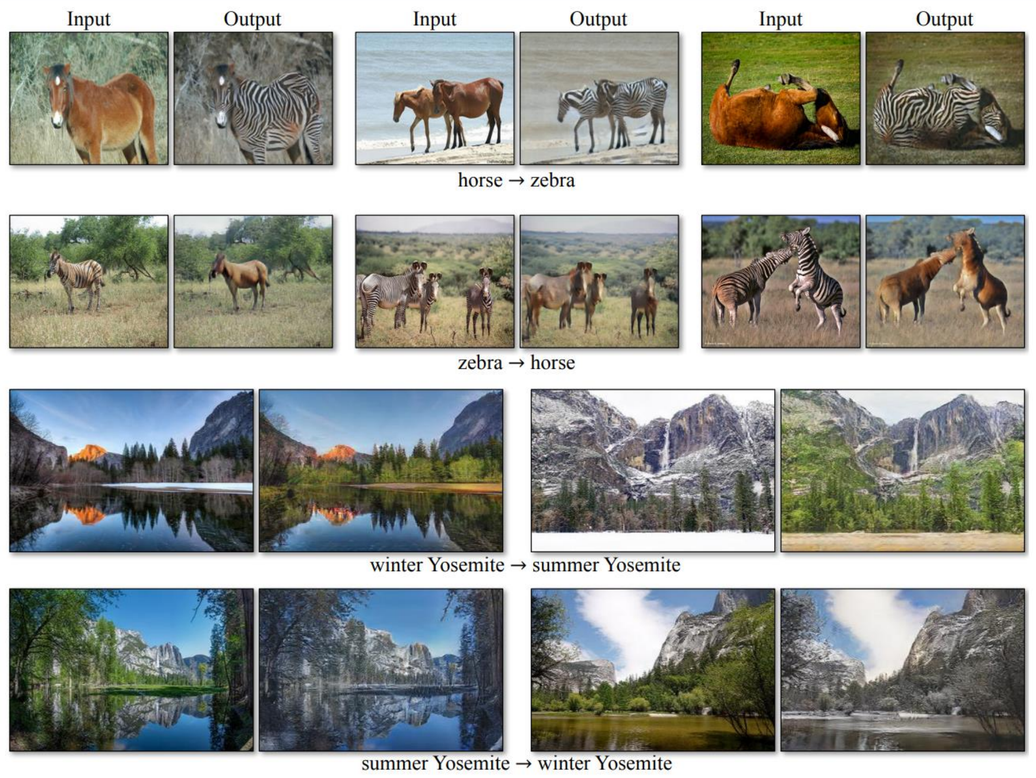
Pix2Pix

➢ Image-to-Image Translation with Conditional Adversarial Networks

https://arxiv.org/pdf/1611.07004.pdf

CycleGAN

➢ Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks



https://arxiv.org/pdf/1703.10593.pdf

CycleGAN

➢ Unpaired Image-to-Image Translation using Cycle-Consistent Adversarial Networks

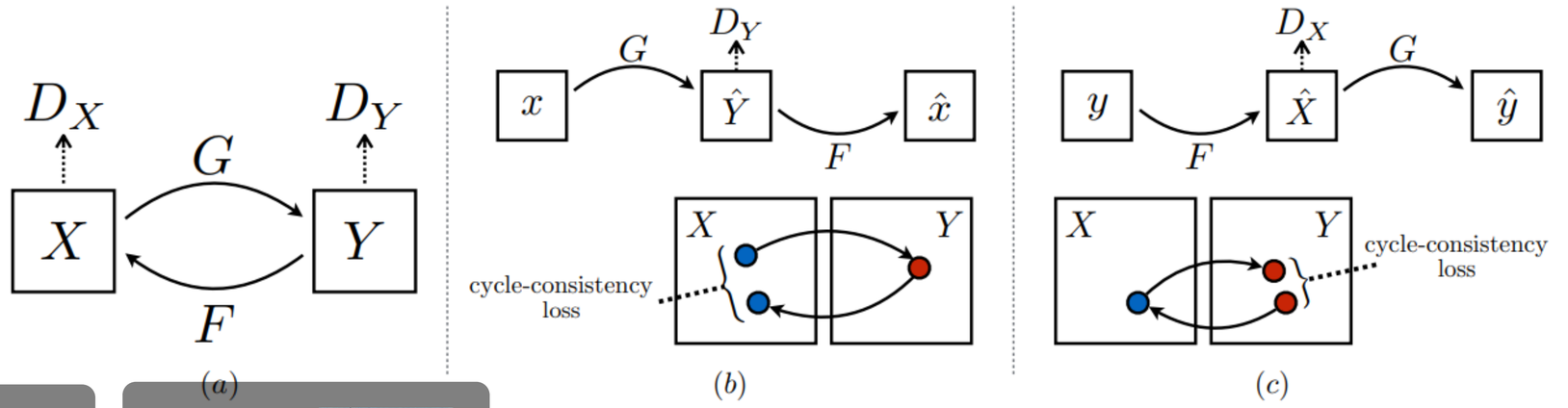
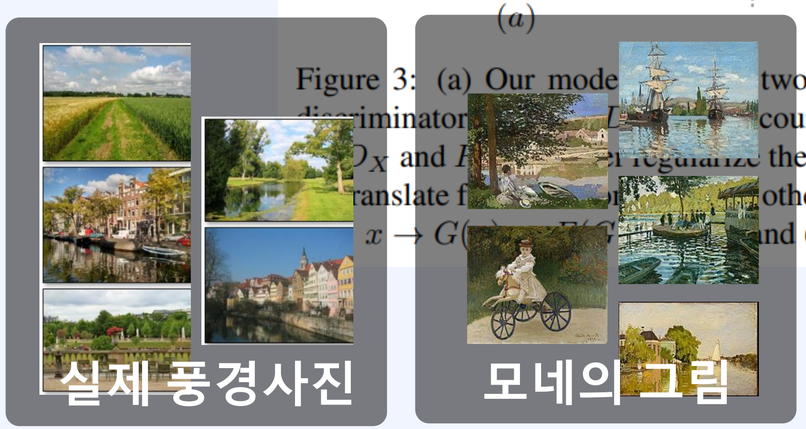


Figure 3: (a) Our mode… two mapping functions $G : X \rightarrow Y$ and $F : Y \rightarrow X$, and associated adversarial discriminator… courages G to translate X into outputs indistinguishable from domain Y, and vice versa …D_X and F… regularize the mappings, we introduce two *cycle consistency losses* that capture the intuition that if …translate f… other and back again we should arrive at where we started: (b) forward cycle-consistency $x \rightarrow G(\ldots)$… and (c) backward cycle-consistency loss: $y \rightarrow F(y) \rightarrow G(F(y)) \approx y$



https://arxiv.org/pdf/1703.10593.pdf

DiscoGAN

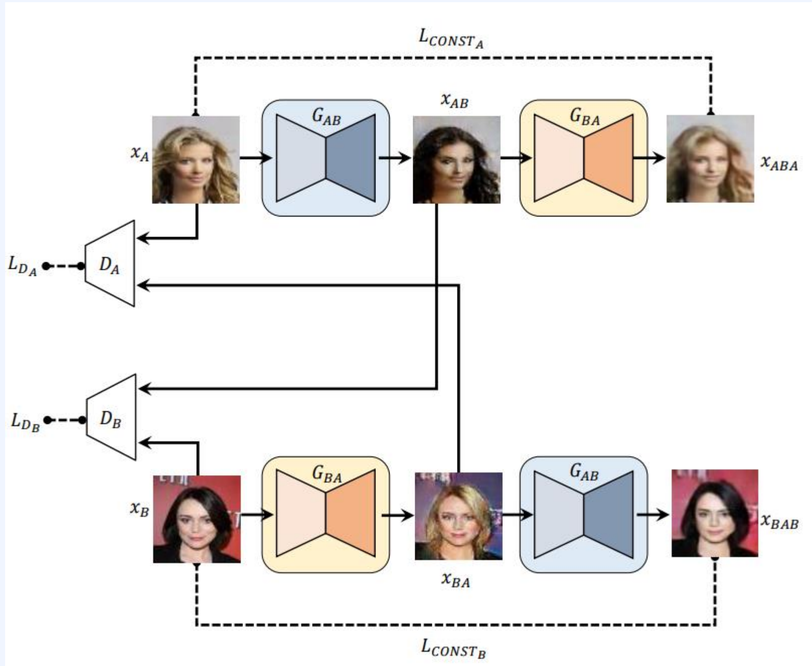
➢ Learning to Discover Cross-Domain Relations with Generative Adversarial Networks

(a) Learning cross-domain relations **without any extra label**

(b) Handbag images (input) & **Generated** shoe images (output)

(c) Shoe images (input) & **Generated** handbag images (output)

https://github.com/SKTBrain/DiscoGAN



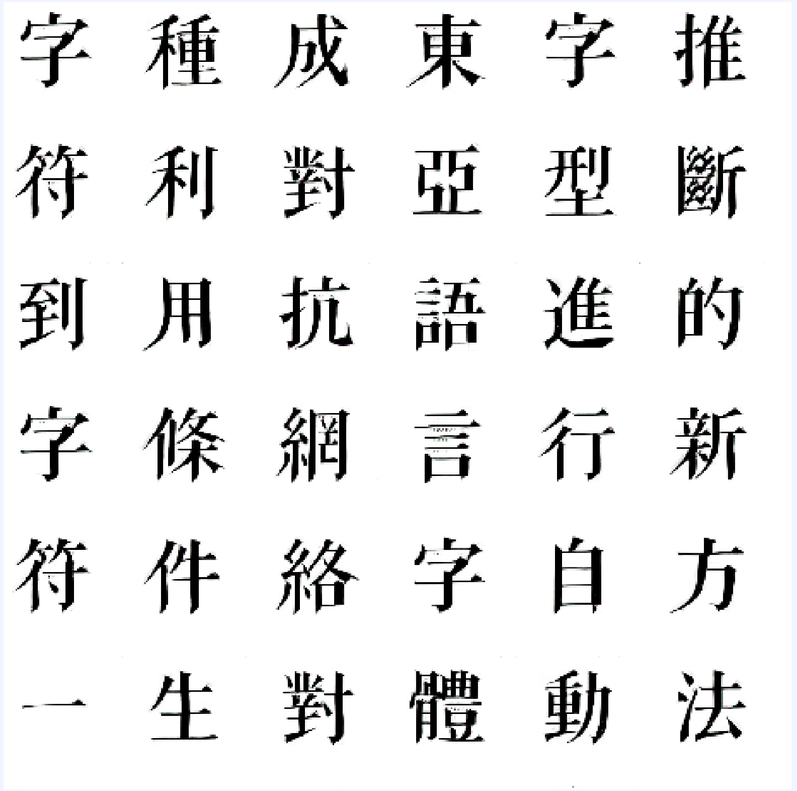
https://arxiv.org/pdf/1703.05192.pdf

StyleGAN

https://github.com/NVlabs/stylegan

Font Generation

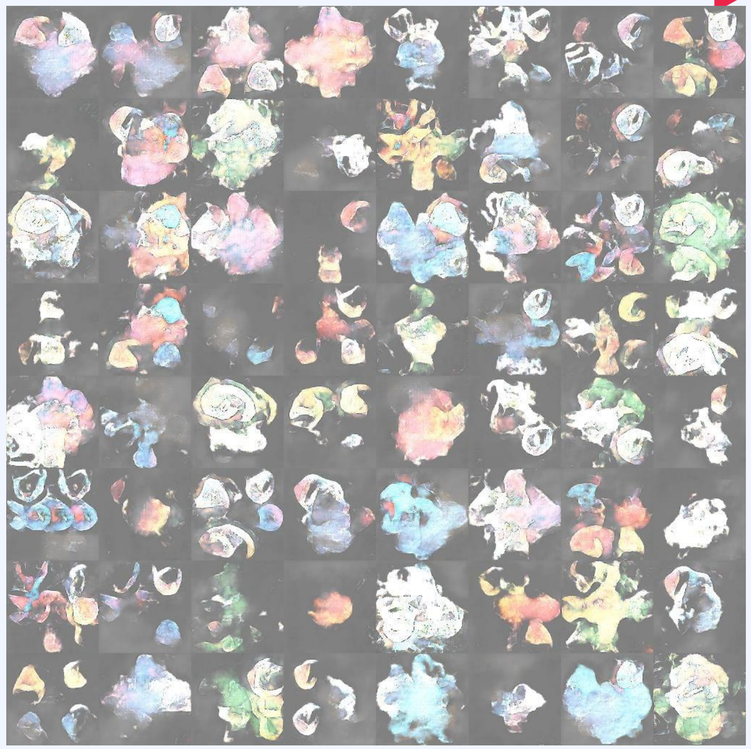
➢ zi2zi: Master Chinese Calligraphy with Conditional Adversarial Networks



https://kaonashi-tyc.github.io/2017/04/06/zi2zi.html

Cartoon Generation

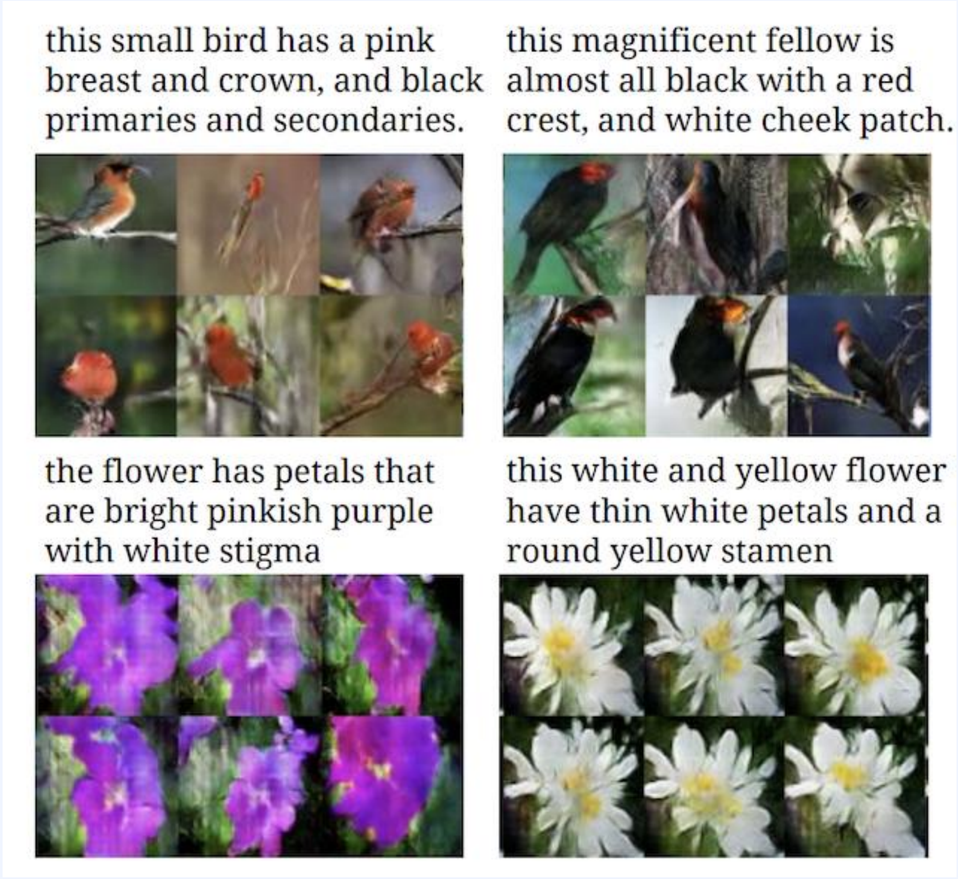
Towards the Automatic Anime Characters Creation with Generative Adversarial Networks



pokeGAN

Text to Image

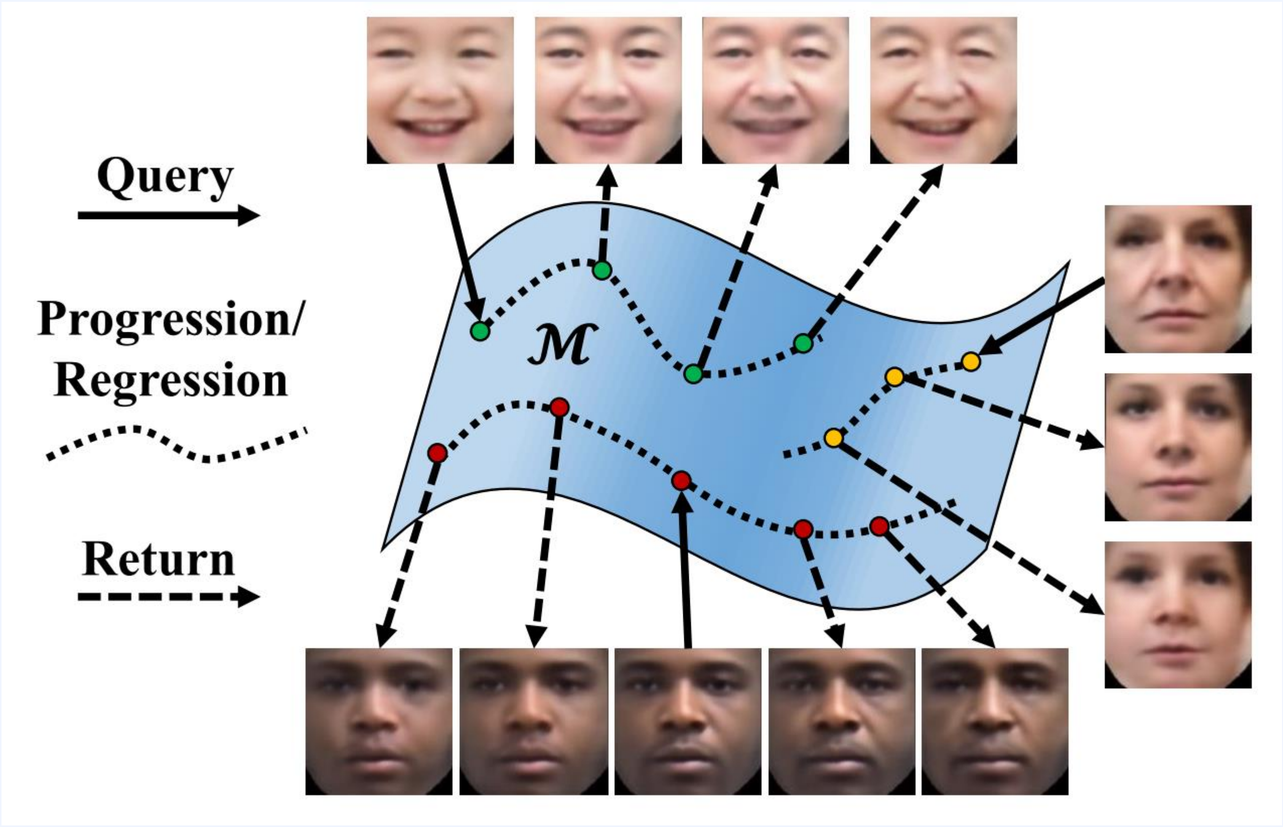
➢ Generative Adversarial Text to Image Synthesis



https://arxiv.org/pdf/1605.05396.pdf

Face Aging

➢ Age Progression/Regression by Conditional Adversarial Autoencoder (CAAE)



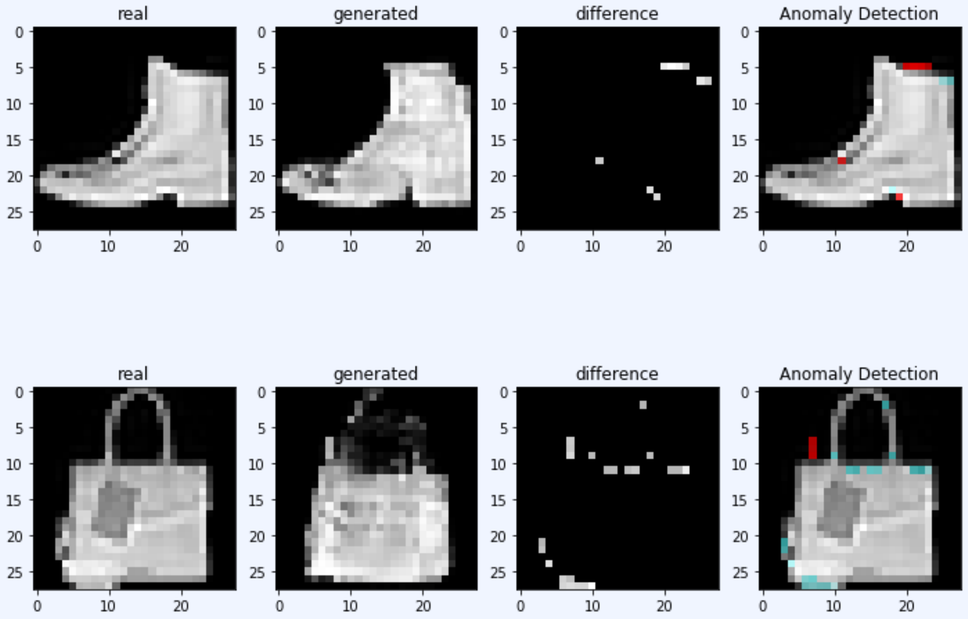
https://github.com/ZZUTK/Face-Aging-CAAE

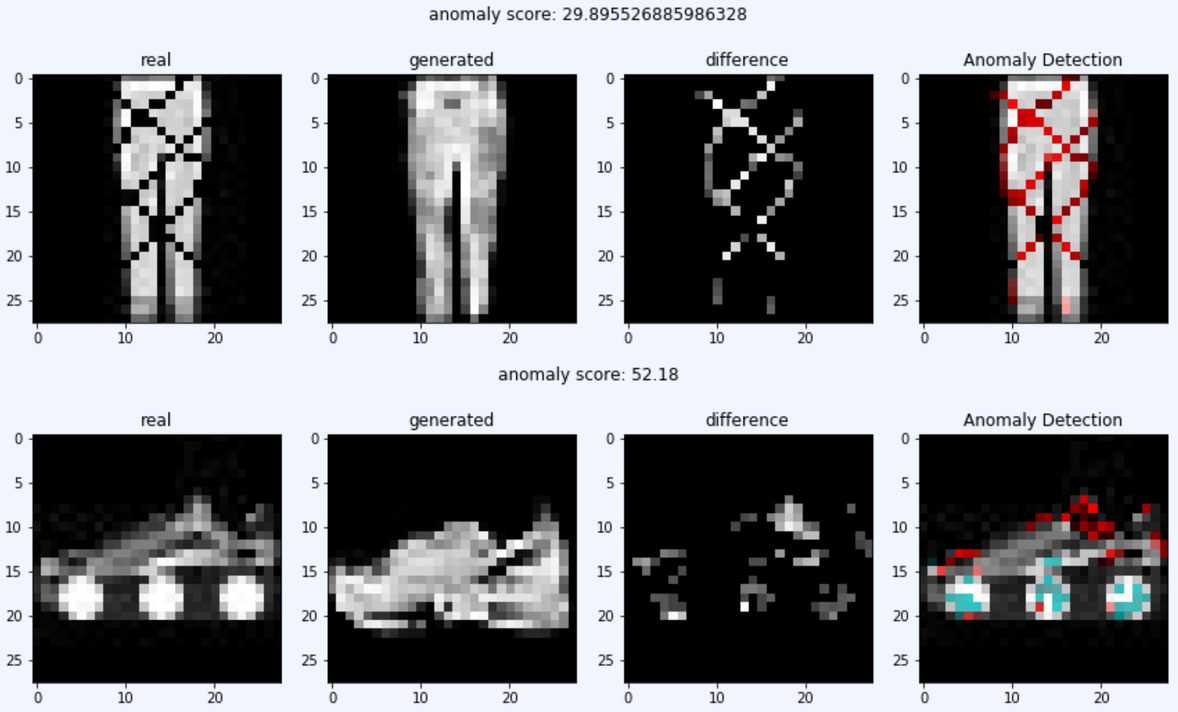
Super resolution

➢ GP-GAN: Towards Realistic High-Resolution Image Blending (ACMMM 2019, oral)

source	destination	mask	composited	blended
				

Anomaly detection





https://github.com/AndersonJo/gan-yhat

GAN

Closing