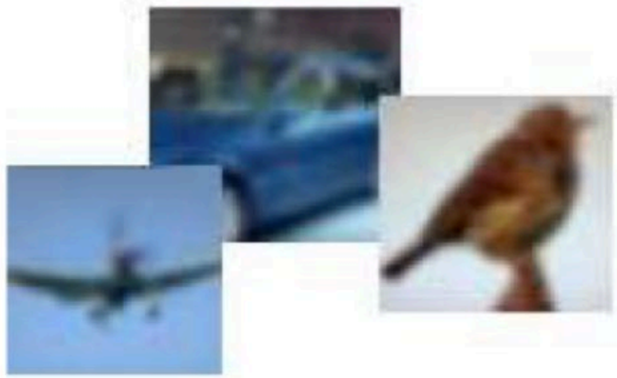
# GAN

POSTECH MIV Lab.

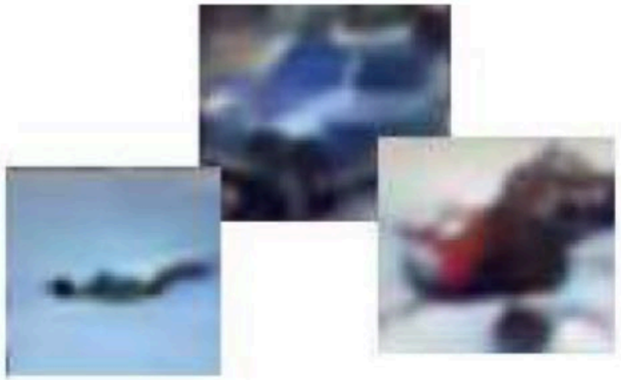
TA: Joonhyuk Park, Hyuna Cho, Minjae Jeong

# Generative models

- 생성모델 (Generative model)은 학습 데이터를 사용하여 학습 데이터의 분포를 따르는 유사한 데이터를 생성하는 모델.



Training samples ~$p_{data}(x)$

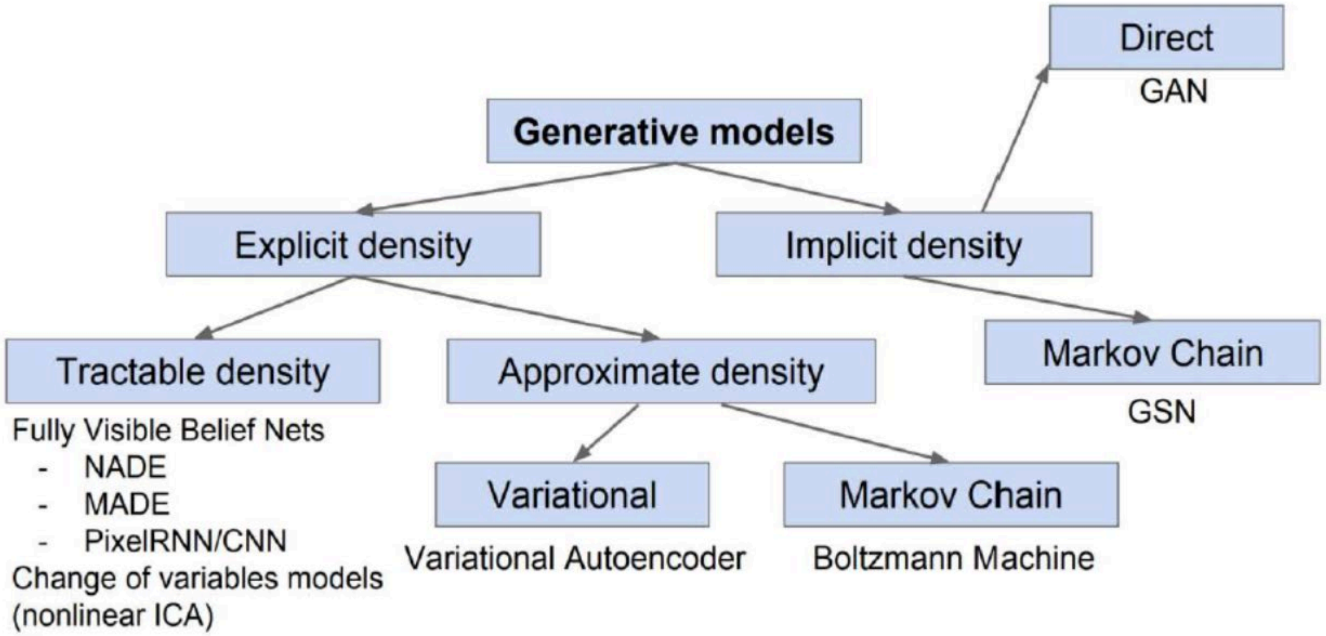


Generated samples ~$p_{model}(x)$

Want to learn $p_{model}(x)$ similar to $p_{data}(x)$

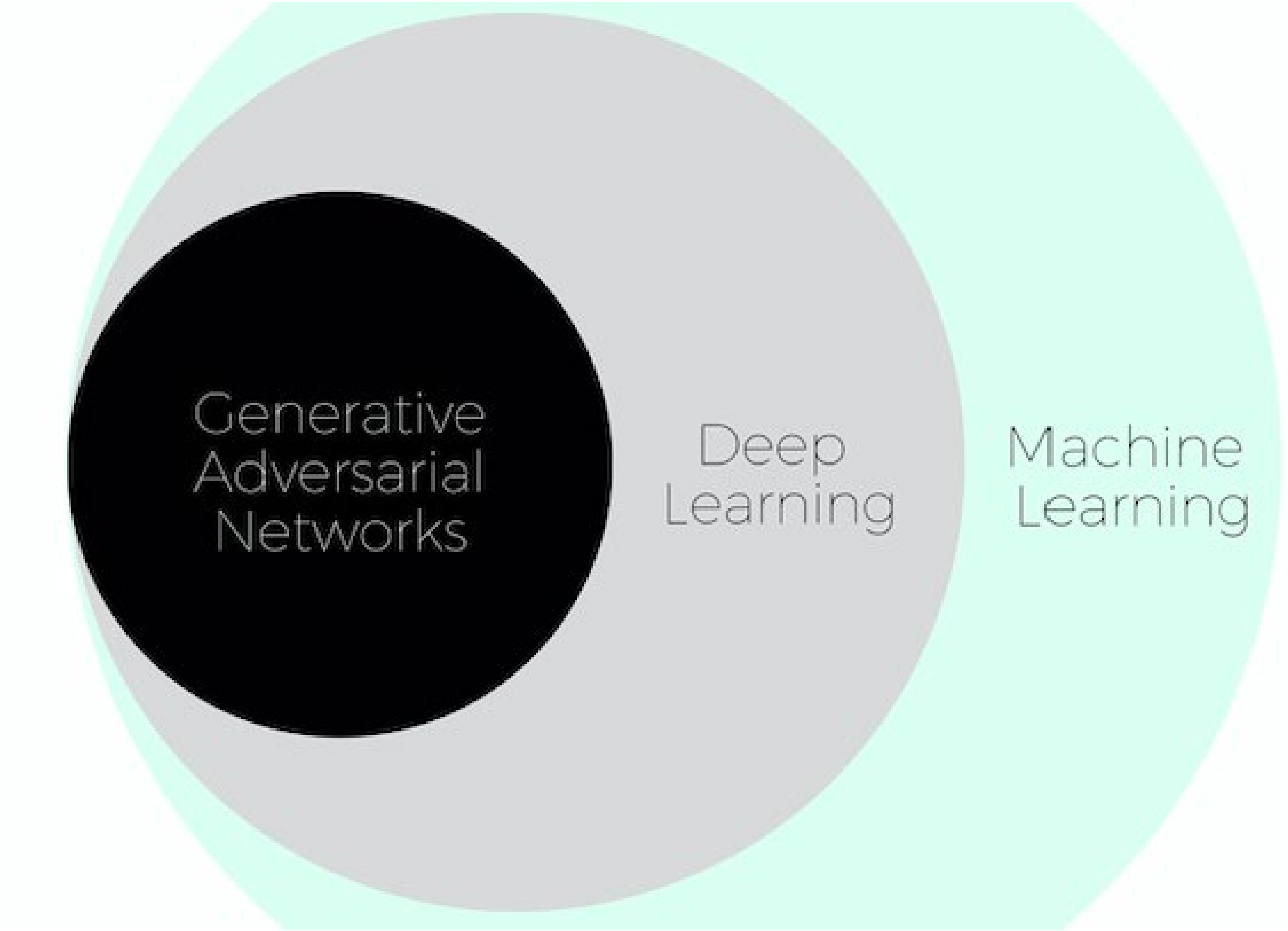
# Generative models

- Explicit vs. Implicit generative model
  - 학습데이터의 분포 P(x)를 바로 모델링 하는 생성모델 explicit generative models
  - 학습데이터의 분포를 몰라도 생성할 수 있는 모델 implicit generative models



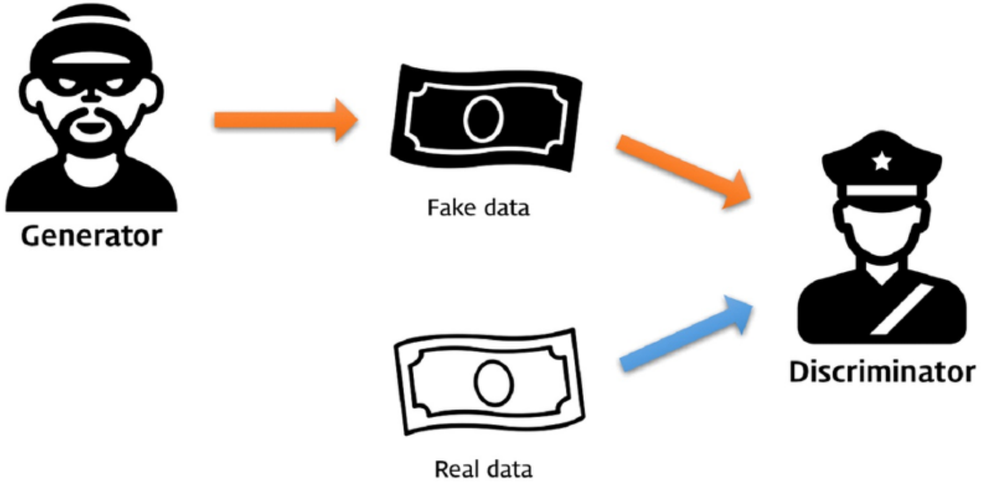
# GAN이란

- G(Generative)
- A(Adversarial)
- N(Network)



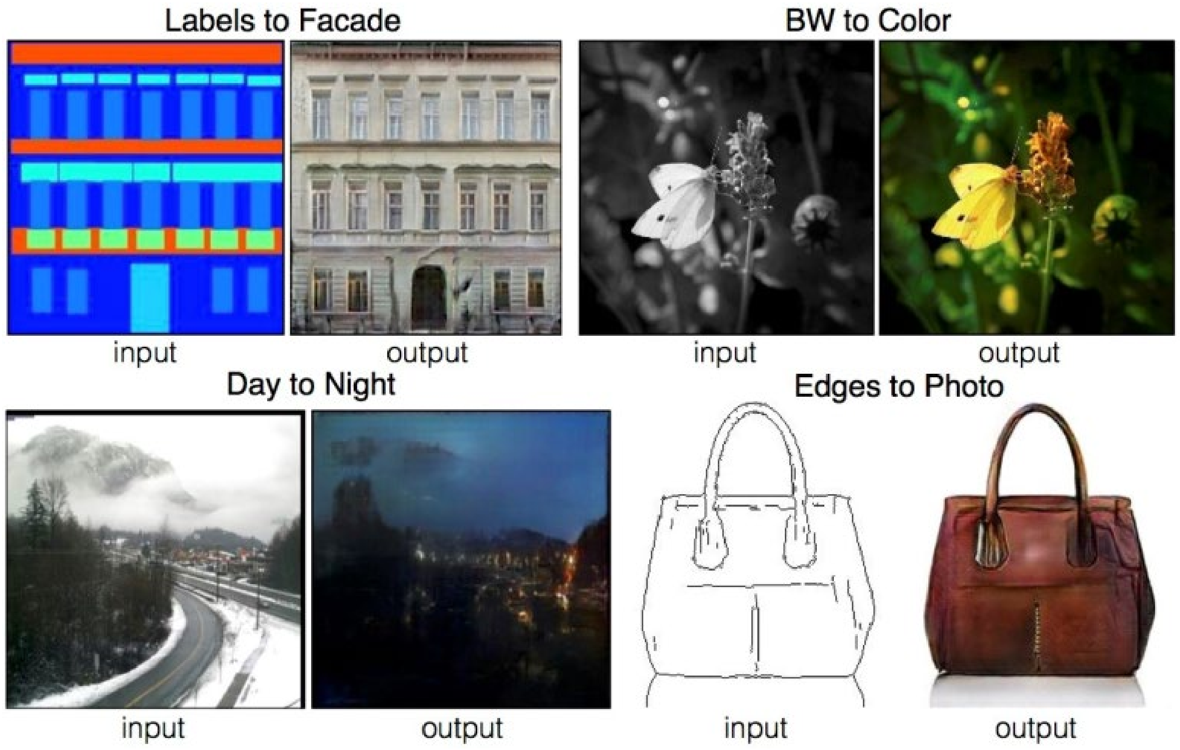
# GAN이란

- 게임이론적인 접근방식
- 지폐위조범(Generator)은 경찰을 최대한 열심히 속이려고 한다. 경찰(Discriminator)은 이렇게 위조된 지폐를 진짜와 감별하려고(Classify) 노력한다.
- 이러한 경쟁 속에서 두 그룹 모두 속이고 구별하는 능력이 발전하고, 결과적으로는 진짜 지폐와 위조 지폐를 구별할 수 없을 정도(구별할 확률 pd=0.5)에 이르게 된다.



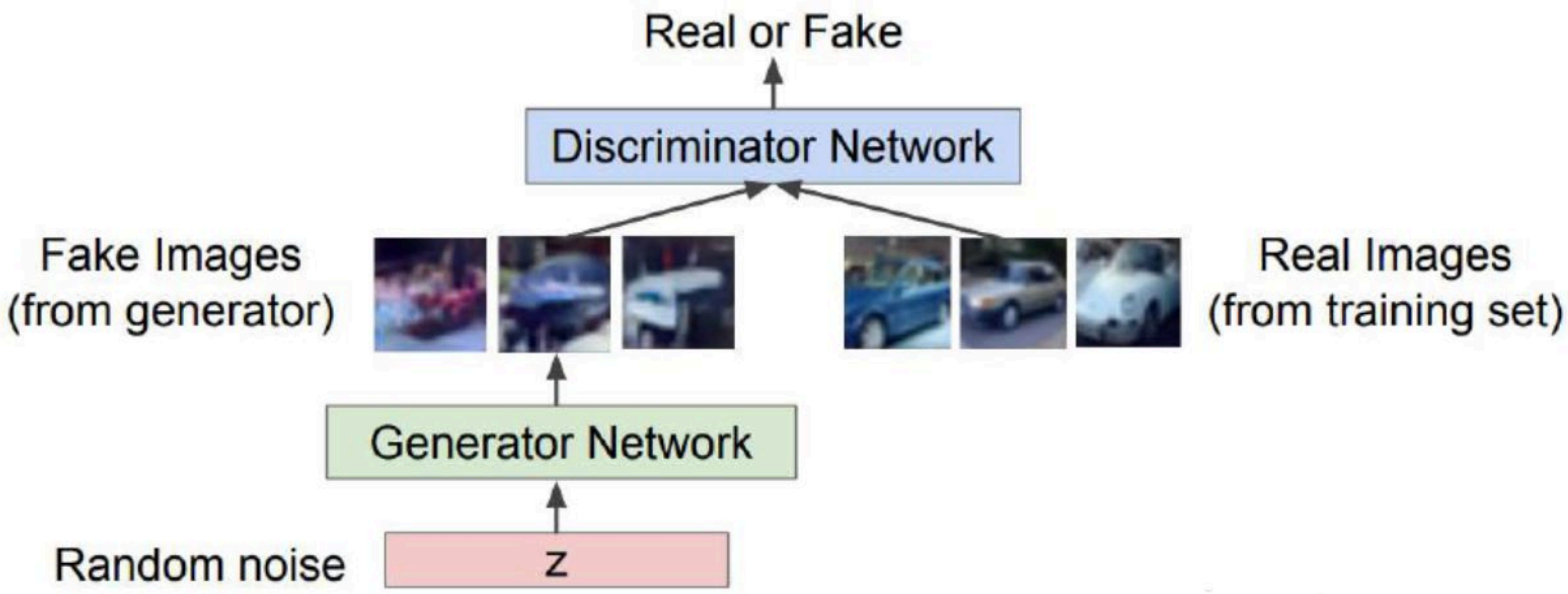
# GAN Applications

- Artwork, super-resolution, colorization, etc.

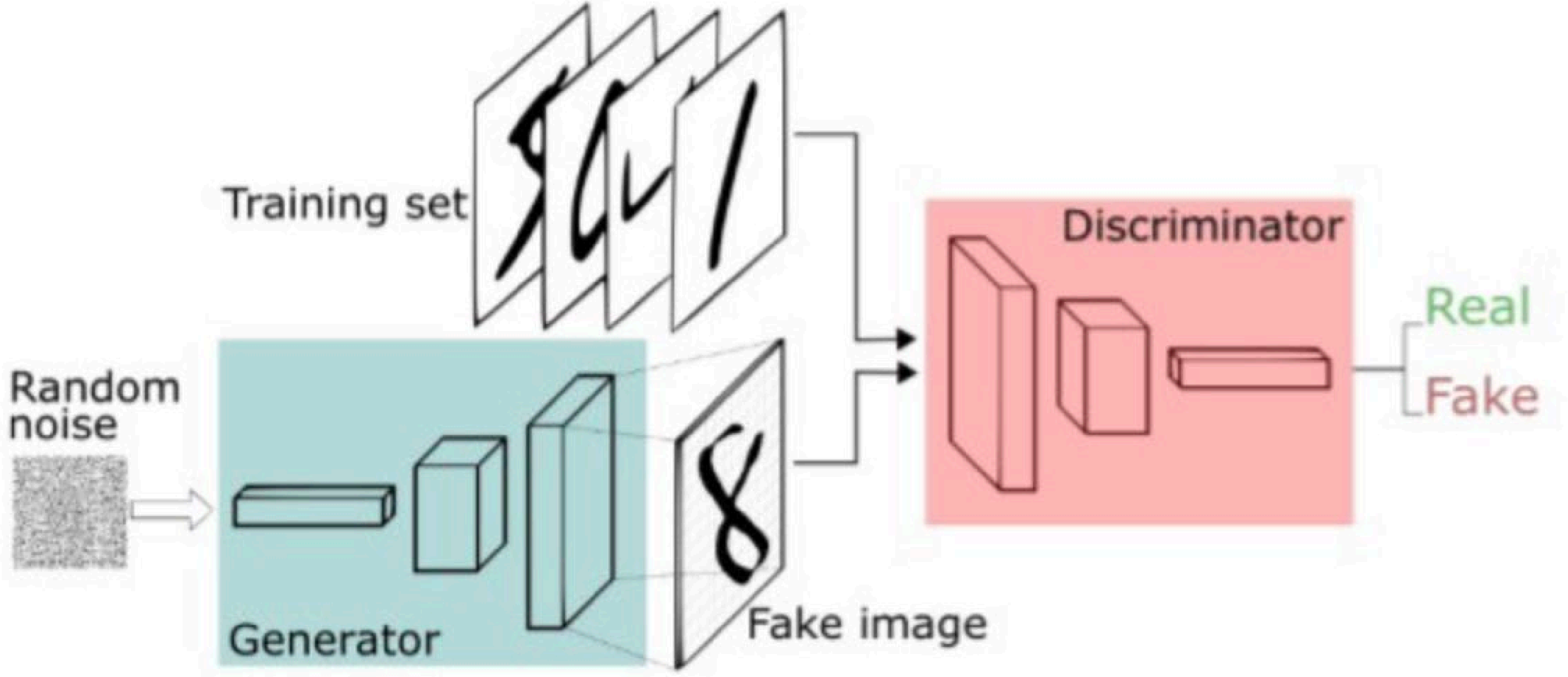



※ 자료 : Ian Goodfellow, NIPS 2016 Tutorial: Generative Adversarial Networks

# GAN Architecture



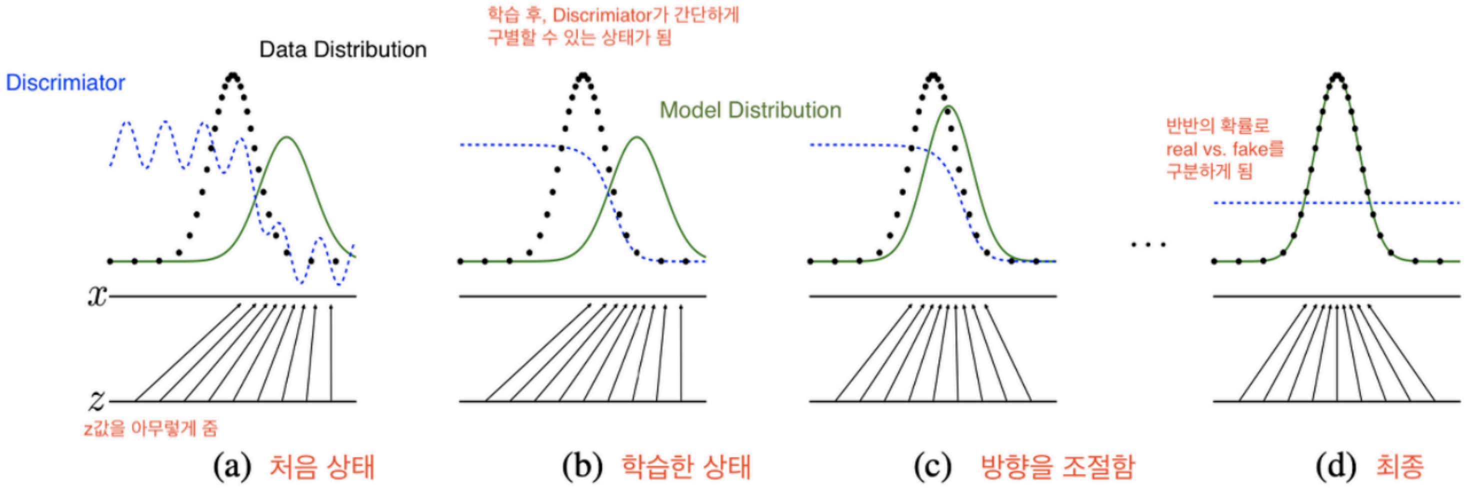
# GAN Architecture




Discriminator and Generator (source:” Pathmind “)

# GAN이란

- Q_model(x|z) : 정의하고자 하는 z값을 줬을 때 x 이미지를 내보내는 모델
- P_data(x) : x라는 data distribution은 있지만 어떻게 생긴지는 모르므로, P 모델을 Q 모델에 가깝게 가도록 함
- **파란 점선 ---** : discriminator distribution (분류 분포) > 학습을 반복하다보면 가장 구분하기 어려운 구별 확률인 1/2 상태가 됨
- **녹색 선 ‒** : generative distribution (가짜 데이터 분포)
- **검은색 점선 ---** : data generating distribution (실제 데이터 분포)




출처 : https://wegonnamakeit.tistory.com/54

# GAN Objective Function

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

The GAN Objective Function

| | |
|---|---|
| D should maximize V(D, G) : D 입장에서는 V가 최댓값 | 1. D가 구분을 잘하는 경우, 만약 Real data가 들어오면 D(X)=1, D(G(Z))=0 : 진짜면 1, 가짜면 0을 내뱉음. (G(Z)에 가짜가 들어온 경우, 가짜를 잘 구분한 것임) |
| D should minimize V(D,G) : G 입장에서 V가 최솟값 | 1. D가 구분을 못하는 경우, 만약 Real Data가 들어오면 D(G(Z))=1: 진짜를 0, 가짜를 1로 내뱉음 (진짜를 구분하지 못하고 가짜를 진짜로 착각함)<br>2. Log안의 D(x)값이 0이 되어, V값이 음의 무한대로 됨.<br>3. Minimize를 위해 음의 무한대로 보내는게 G 입장에서는 가장 좋음 |
| - X: real 이미지, Z: latent code, G(Z) : fake 이미지, D(X) : real 이미지라고 분류한 확률, D(G(Z)) : D가 fake라고 분류한 확률<br>- G(z)는 D(G(z))가 1로 판단하도록 학습하고, D(G(Z))는 0으로 판단하도록 학습함. | |

출처 : https://wegonnamakeit.tistory.com/54

# GAN Objective Function

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

The GAN Objective Function

Alternate between:

1. **Gradient ascent** on discriminator

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

2. **Gradient descent** on generator

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

출처 : https://wegonnamakeit.tistory.com/54

# GAN Objective Function

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

The GAN Objective Function

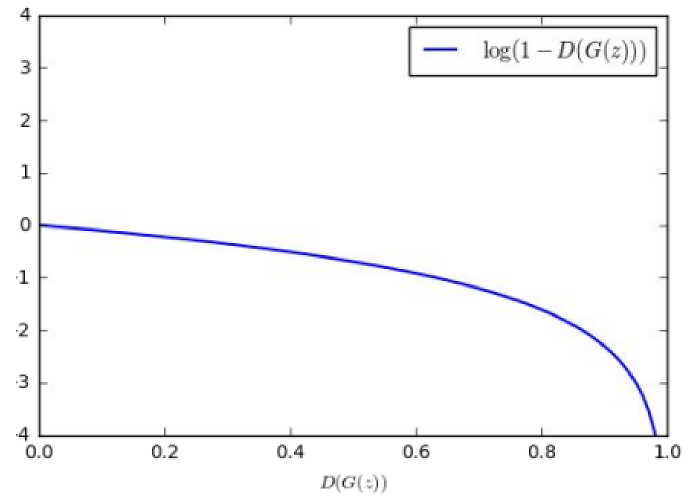
Alternate between:

1. **Gradient ascent** on discriminator

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

2. **Gradient descent** on generator

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




출처 : https://wegonnamakeit.tistory.com/54

# GAN Objective Function

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

The GAN Objective Function

Alternate between:

1. **Gradient ascent** on discriminator

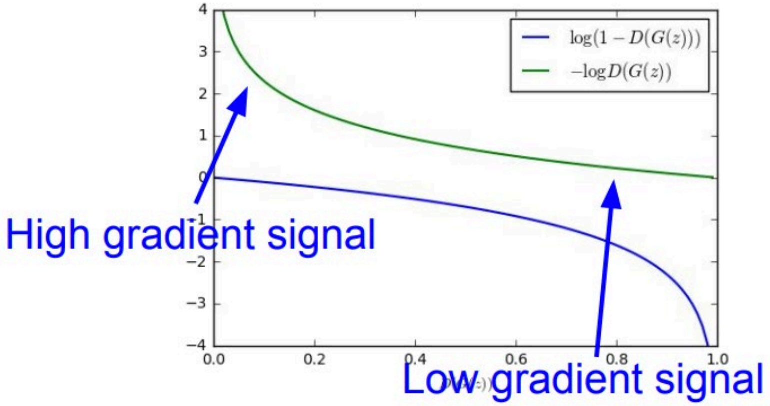
$$\max_{\theta_d} \left[\mathbb{E}_{x\sim p_{data}} \log D_{\theta_d}(x) + \mathbb{E}_{z\sim p(z)} \log(1 - D_{\theta_d}(G_{\theta_g}(z)))\right]$$

2. **Gradient descent** on generator

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

**Instead: Gradient ascent on generator, different objective**

$$\max_{\theta_g} \mathbb{E}_{z\sim p(z)} \log(D_{\theta_d}(G_{\theta_g}(z)))$$




출처 : https://wegonnamakeit.tistory.com/54

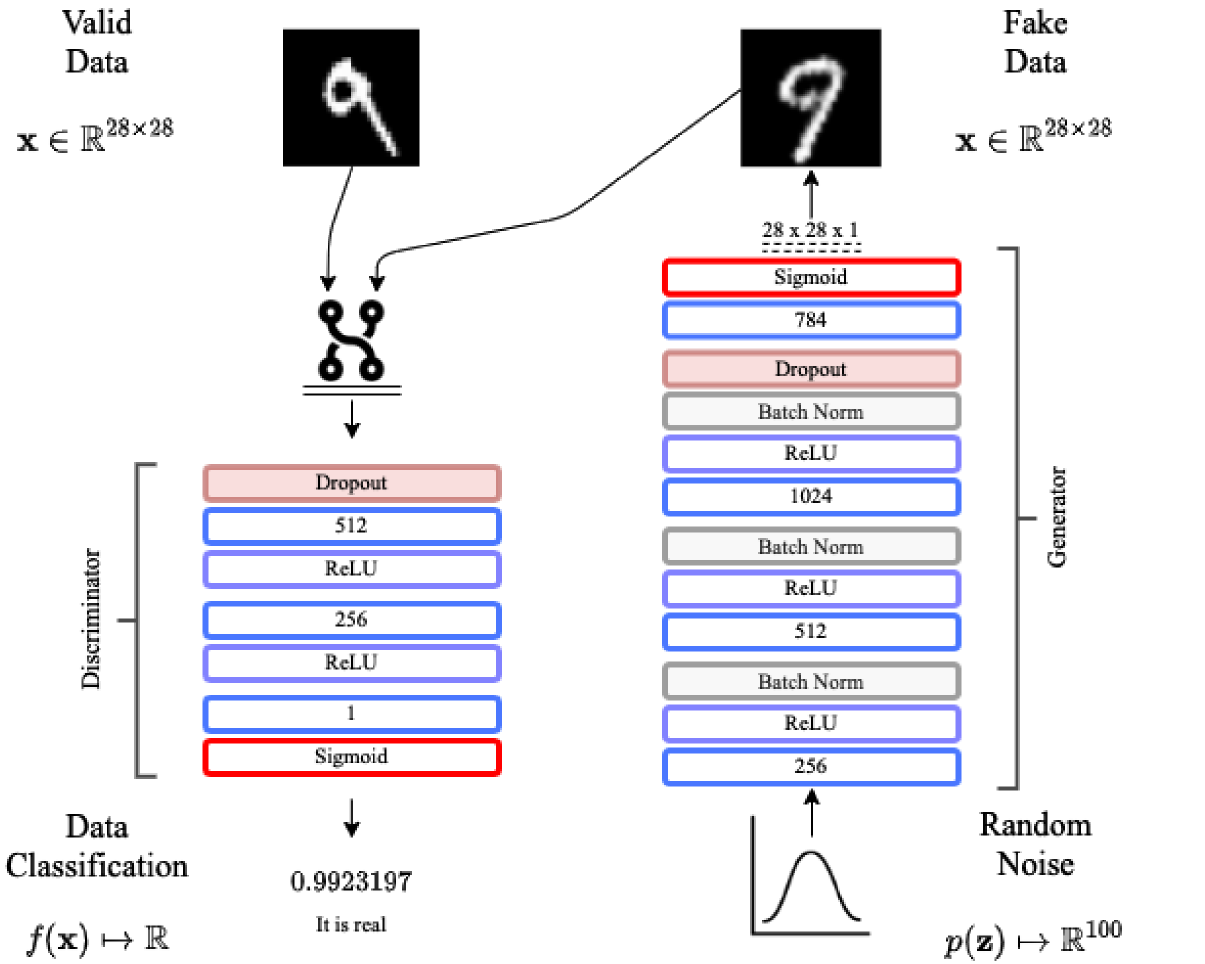
# BCE loss

$$Loss = L(y, \hat{p}) = -y \log(\hat{p}) - (1 - y) \log(1 - \hat{p})$$

where $y \Rightarrow$ label,

$\hat{p} \Rightarrow$ estimated probability of belonging to the positive class

# Vanilla GAN



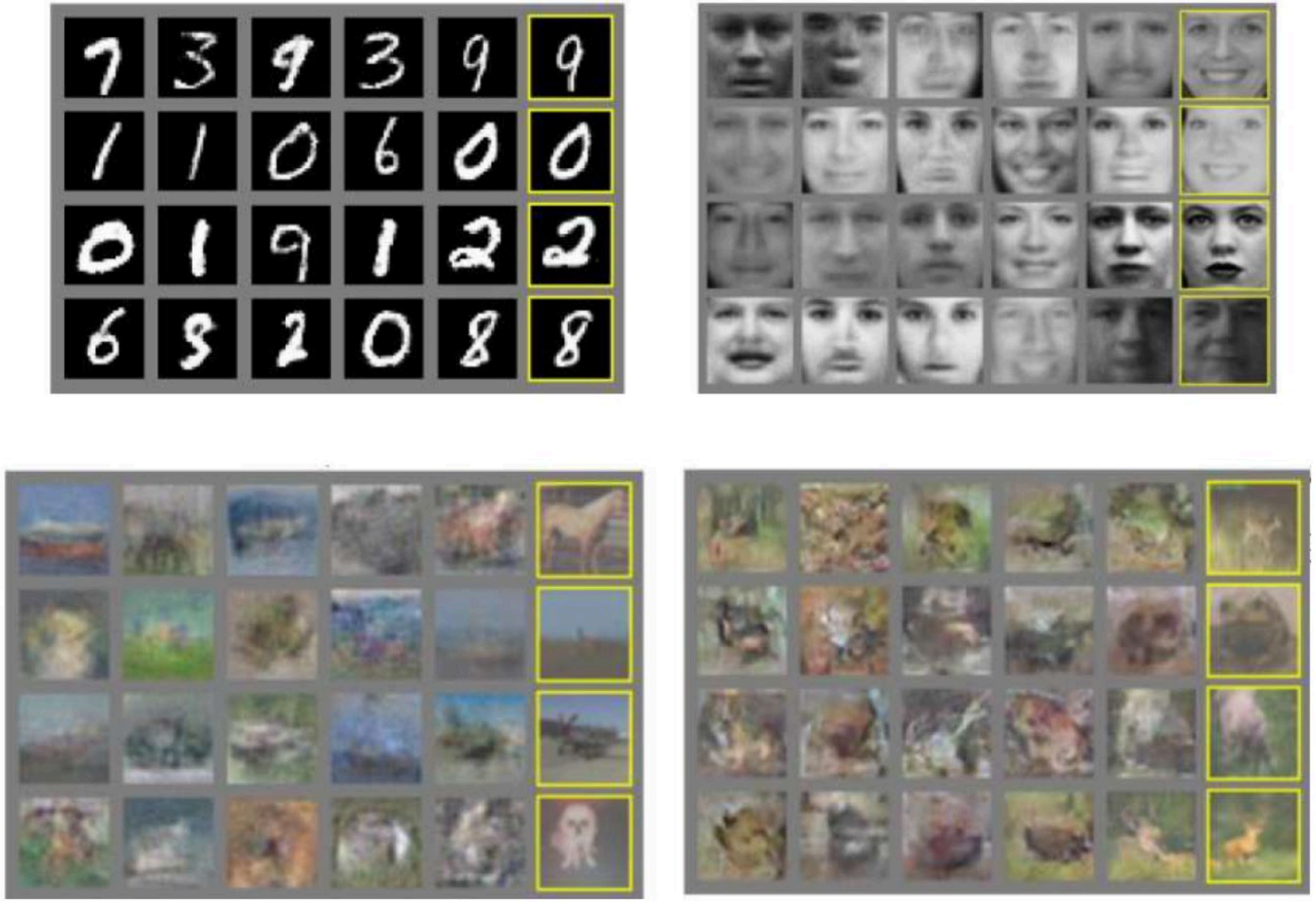

**Generator Parameters**

This network has a total of 1,493,520 parameters.

**Discriminator Parameters**

This network has a total of 533,505 parameters.

# Vanilla GAN



# DCGAN (Deep Convolutional GAN)

- 앞의 vanilla GAN 같은 경우 쉬운 dataset (e.g. MNIST)는 잘 생성하지만 복잡한 이미지(개, 고양이)를 생성하는 것은 한계.
- DCGAN은 MLP대신에 CNN을 적용한 모델
- Details

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.

# Practice 1: Vanilla_GAN

# Details (Generator, Discriminator)

| Layer (type) | Output Shape | Param # |
|---|---|---|
| Linear-1 | [-1, 256] | 25,856 |
| LeakyReLU-2 | [-1, 256] | 0 |
| BatchNorm1d-3 | [-1, 256] | 512 |
| Linear-4 | [-1, 512] | 131,584 |
| LeakyReLU-5 | [-1, 512] | 0 |
| BatchNorm1d-6 | [-1, 512] | 1,024 |
| Linear-7 | [-1, 1024] | 525,312 |
| LeakyReLU-8 | [-1, 1024] | 0 |
| BatchNorm1d-9 | [-1, 1024] | 2,048 |
| Dropout-10 | [-1, 1024] | 0 |
| Linear-11 | [-1, 784] | 803,600 |
| Tanh-12 | [-1, 784] | 0 |

Total params: 1,489,936
Trainable params: 1,489,936
Non-trainable params: 0

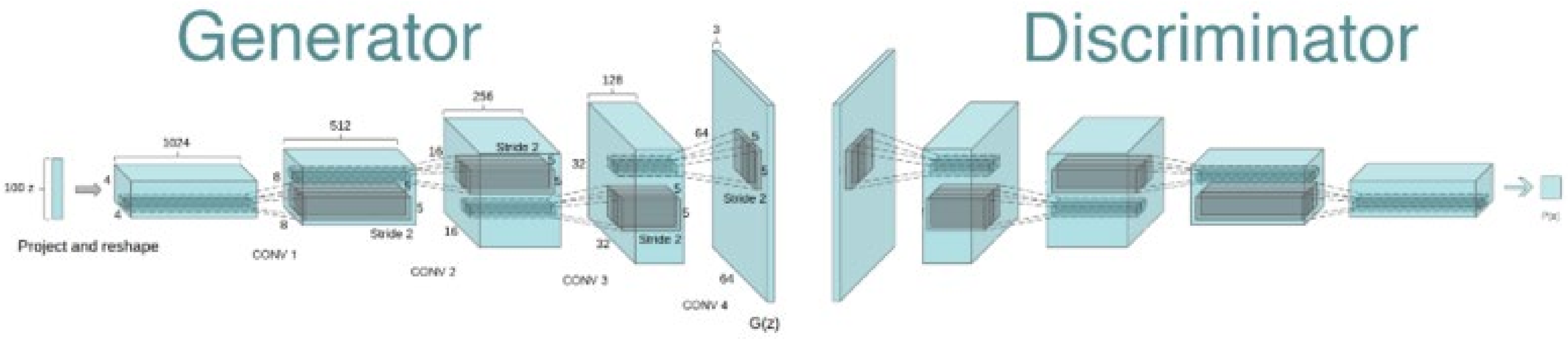
Input size (MB): 0.00
Forward/backward pass size (MB): 0.06
Params size (MB): 5.68
Estimated Total Size (MB): 5.74

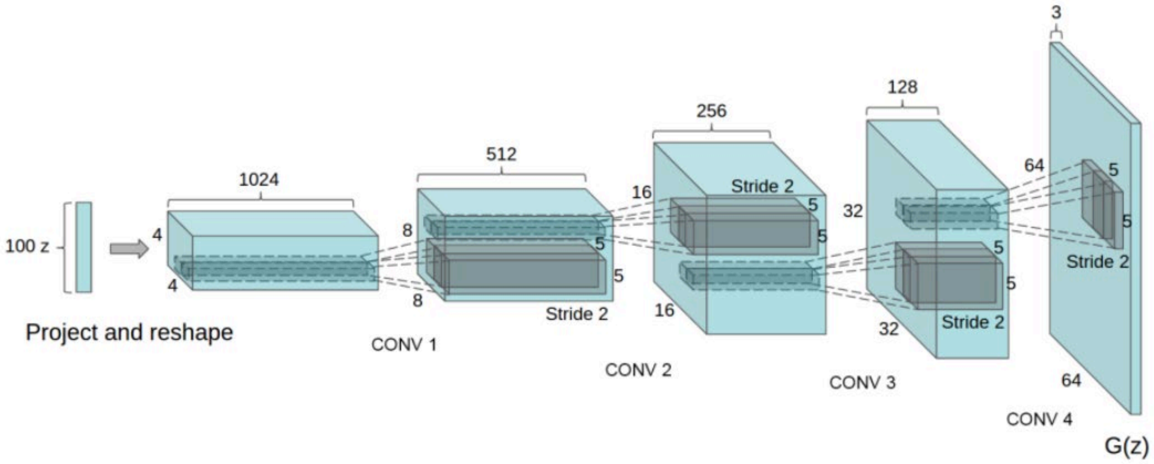
| Layer (type) | Output Shape | Param # |
|---|---|---|
| Dropout-1 | [-1, 784] | 0 |
| Linear-2 | [-1, 512] | 401,920 |
| LeakyReLU-3 | [-1, 512] | 0 |
| Linear-4 | [-1, 256] | 131,328 |
| LeakyReLU-5 | [-1, 256] | 0 |
| Linear-6 | [-1, 1] | 257 |
| Sigmoid-7 | [-1, 1] | 0 |

Total params: 533,505
Trainable params: 533,505
Non-trainable params: 0

Input size (MB): 0.00
Forward/backward pass size (MB): 0.02
Params size (MB): 2.04
Estimated Total Size (MB): 2.06

# DCGAN (Deep Convolutional GAN)




DCGAN Generator 구조

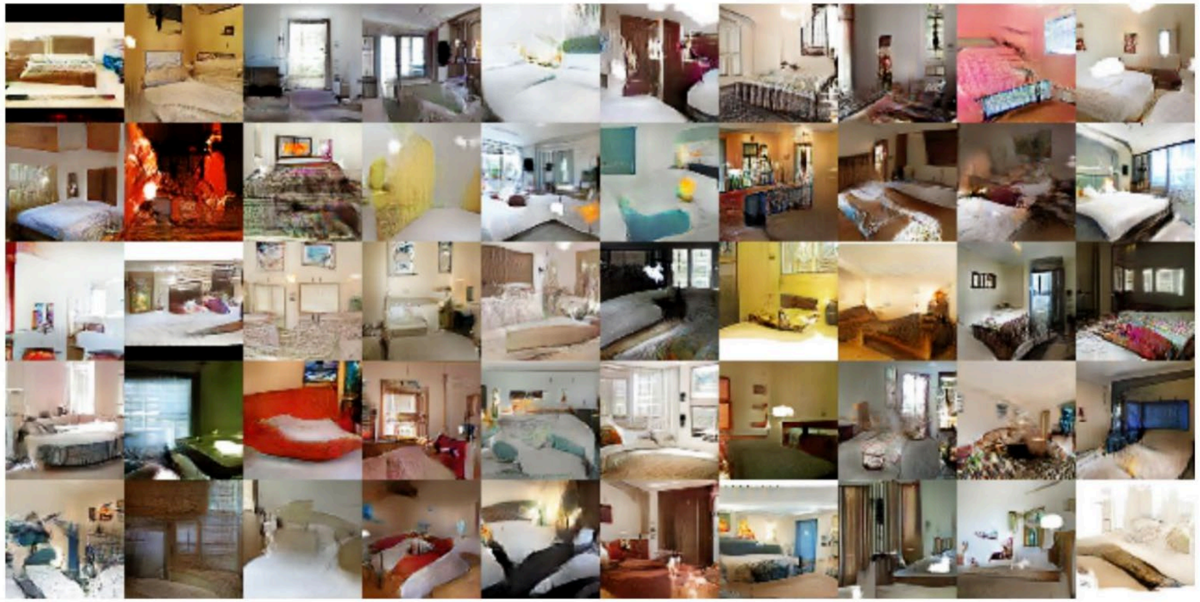
# Practice 2 : DCGAN

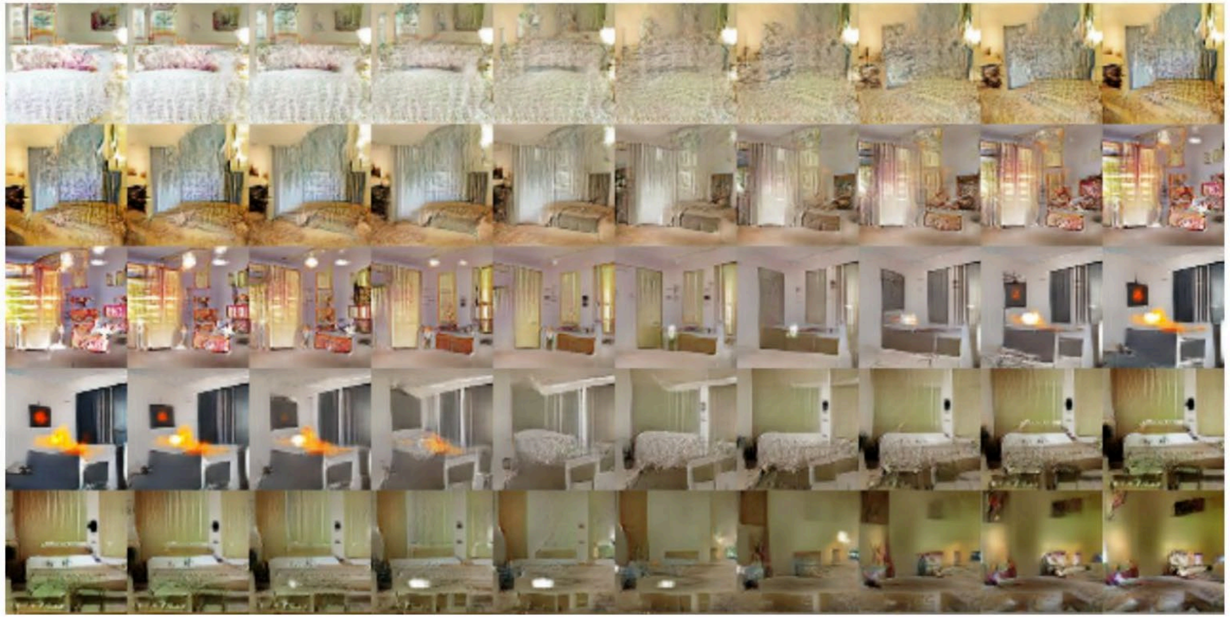
# Details (Generator, Discriminator)

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Linear-1                 [-1, 8192]         827,392
       BatchNorm2d-2            [-1, 128, 8, 8]             256
          Upsample-3          [-1, 128, 16, 16]               0
            Conv2d-4          [-1, 128, 16, 16]         147,584
       BatchNorm2d-5          [-1, 128, 16, 16]             256
         LeakyReLU-6          [-1, 128, 16, 16]               0
          Upsample-7          [-1, 128, 32, 32]               0
            Conv2d-8           [-1, 64, 32, 32]          73,792
       BatchNorm2d-9           [-1, 64, 32, 32]             128
        LeakyReLU-10           [-1, 64, 32, 32]               0
           Conv2d-11            [-1, 1, 32, 32]             577
             Tanh-12            [-1, 1, 32, 32]               0
================================================================
Total params: 1,049,985
Trainable params: 1,049,985
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 3.64
Params size (MB): 4.01
Estimated Total Size (MB): 7.65
----------------------------------------------------------------
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 16, 16]             160
         LeakyReLU-2           [-1, 16, 16, 16]               0
         Dropout2d-3           [-1, 16, 16, 16]               0
            Conv2d-4             [-1, 32, 8, 8]           4,640
         LeakyReLU-5             [-1, 32, 8, 8]               0
         Dropout2d-6             [-1, 32, 8, 8]               0
       BatchNorm2d-7             [-1, 32, 8, 8]              64
            Conv2d-8             [-1, 64, 4, 4]          18,496
         LeakyReLU-9             [-1, 64, 4, 4]               0
        Dropout2d-10             [-1, 64, 4, 4]               0
      BatchNorm2d-11             [-1, 64, 4, 4]             128
           Conv2d-12            [-1, 128, 2, 2]          73,856
        LeakyReLU-13            [-1, 128, 2, 2]               0
        Dropout2d-14            [-1, 128, 2, 2]               0
      BatchNorm2d-15            [-1, 128, 2, 2]             256
           Linear-16                    [-1, 1]             513
          Sigmoid-17                    [-1, 1]               0
================================================================
Total params: 98,113
Trainable params: 98,113
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.20
Params size (MB): 0.37
Estimated Total Size (MB): 0.58
----------------------------------------------------------------
```

# DCGAN (Deep Convolutional GAN)



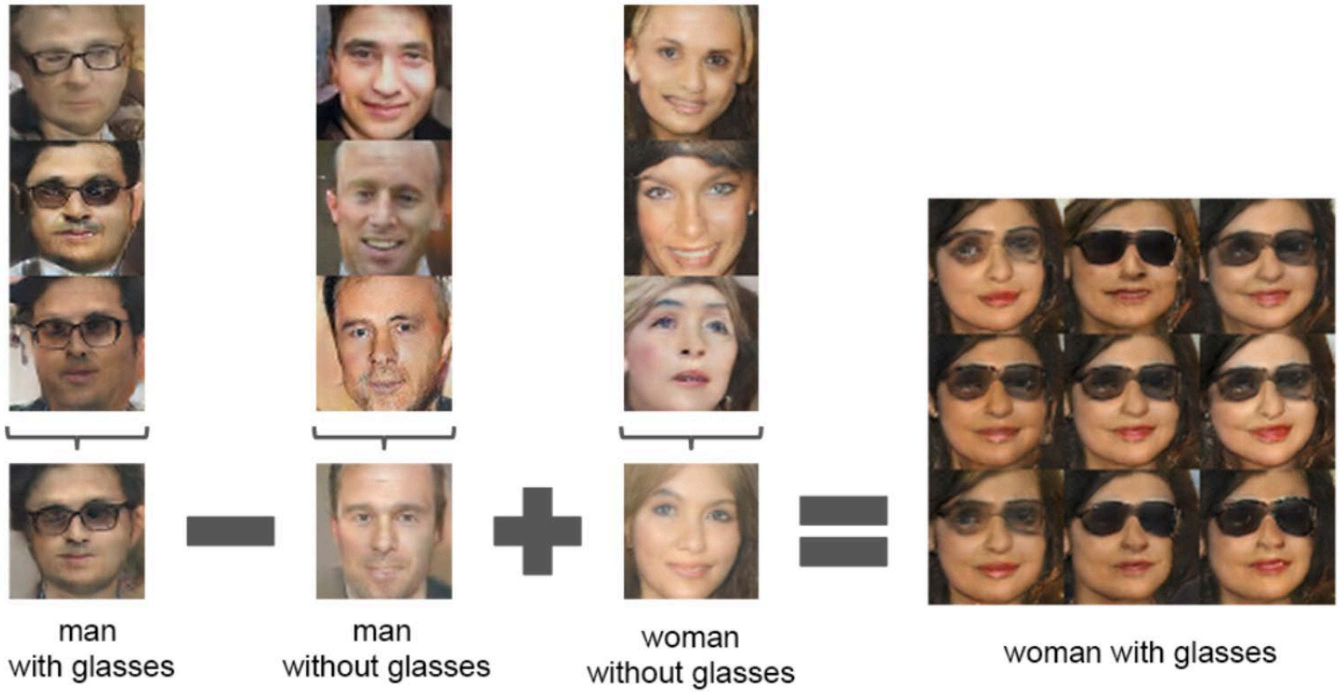


# DCGAN (Deep Convolutional GAN)

- Latent space(Noise) interpolation
  - Generator가 단순히 학습 이미지를 외우거나 베낀게 아니어야 함
  - Walking in the latent space

# DCGAN (Deep Convolutional GAN)

- Vector Arithmetic on face samples



# GAN의 어려운 점

1. Unstable Training (Convergence, Mode Collapse)
2. 생성 모델 평가 지표 (ex. FID, IS, etc.)
3. 학습 데이터의 분포를 따라가기 때문에 어떤 데이터가 생성될지 예측하기 어려움 (cGAN)

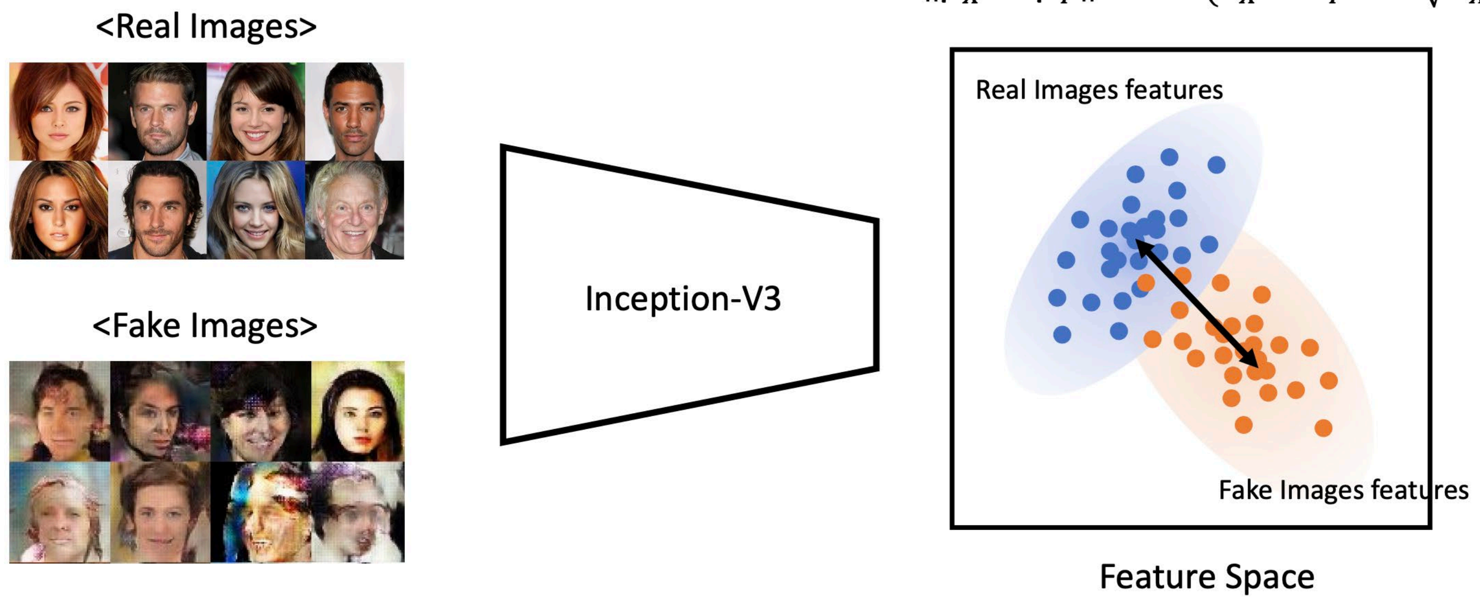
# Evaluation of GAN models

- Hard to evaluate how well the images are generated
  - There is no GT for image generation
  - Solution: Compare the distribution of real and generated images

# Evaluation of GAN models

- FID (Frechet Inception Distance)
  - Use pretrained Inception-V3 model, assuming it can extract essential image features well enough.

$$FID = \|\mu_X - \mu_Y\|^2 + Tr\left(\Sigma_X + \Sigma_Y - 2\sqrt{\Sigma_X \Sigma_Y}\right)$$



# 실습

- Simple GAN (MLP based)
- DCGAN (Deep Convolutional GAN)