Generative Models

2024. 8

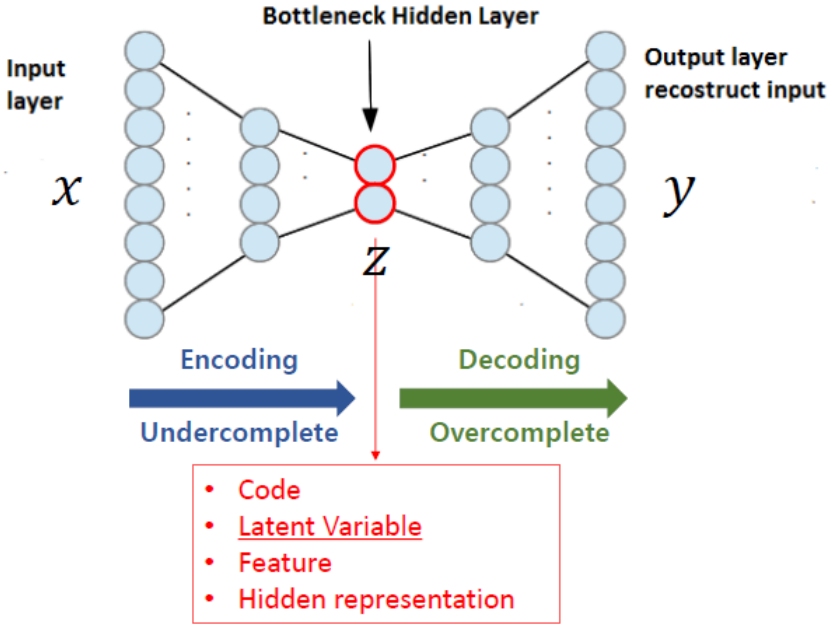
Yongjin Jeong, KwangWoon University

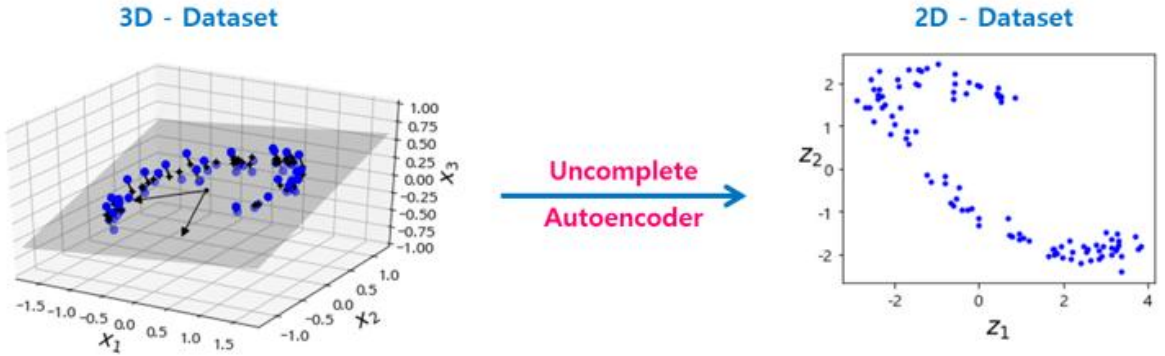
Contents

- Autoencoders
- Variational Autoencoder (VAE)
- GANs

Autoencoders

- **Autoencoder**
  - To learn a representation (encoding) for training data set (dimensionality reduction) by training the network to ignore signal "noise", in other words, feature extraction.
  - Focus on the encoder.





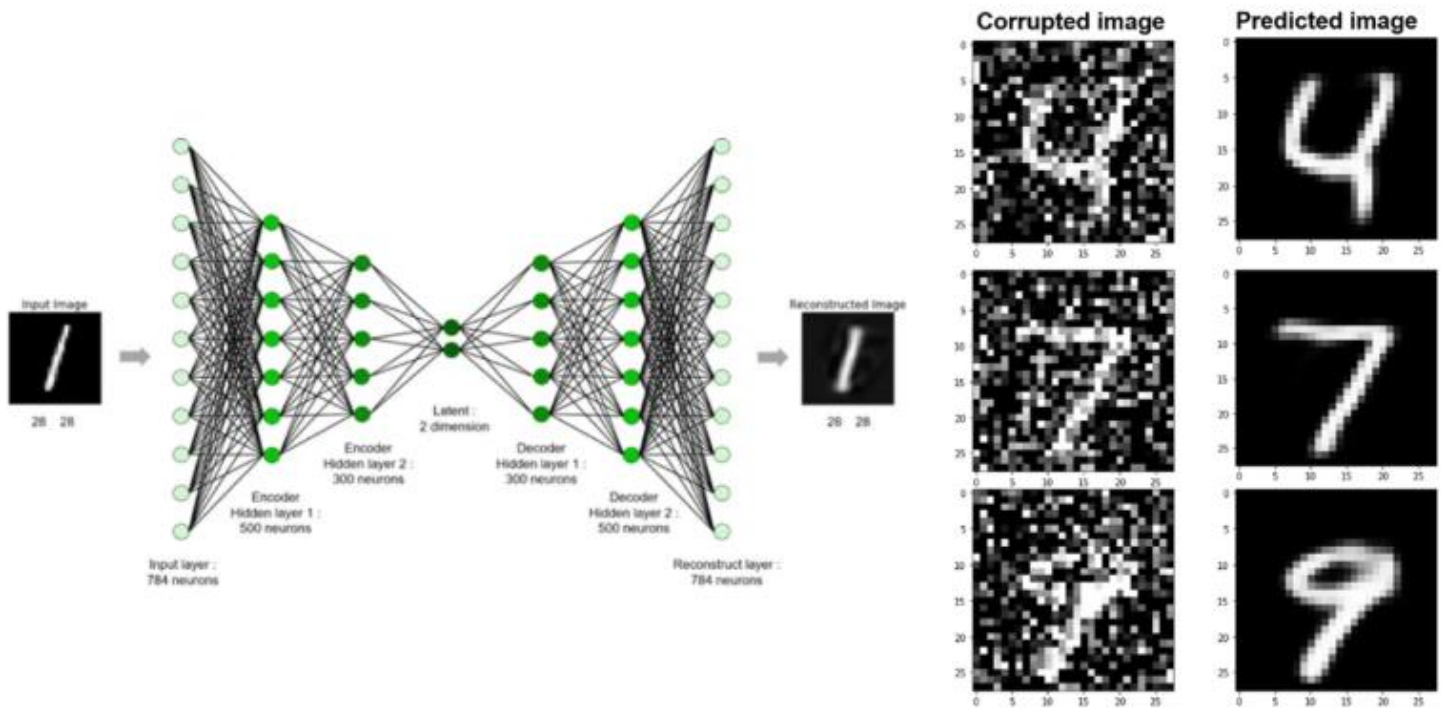
http://videolectures.net/deeplearning2015_vincent_autoencoders/?q=vincent%20autoencoder

Autoencoders

- **Applications of Autoencoders**
  - Dimensionality reduction
  - Anomaly detection
  - Image denoising
  - Image compression
  - Image generation

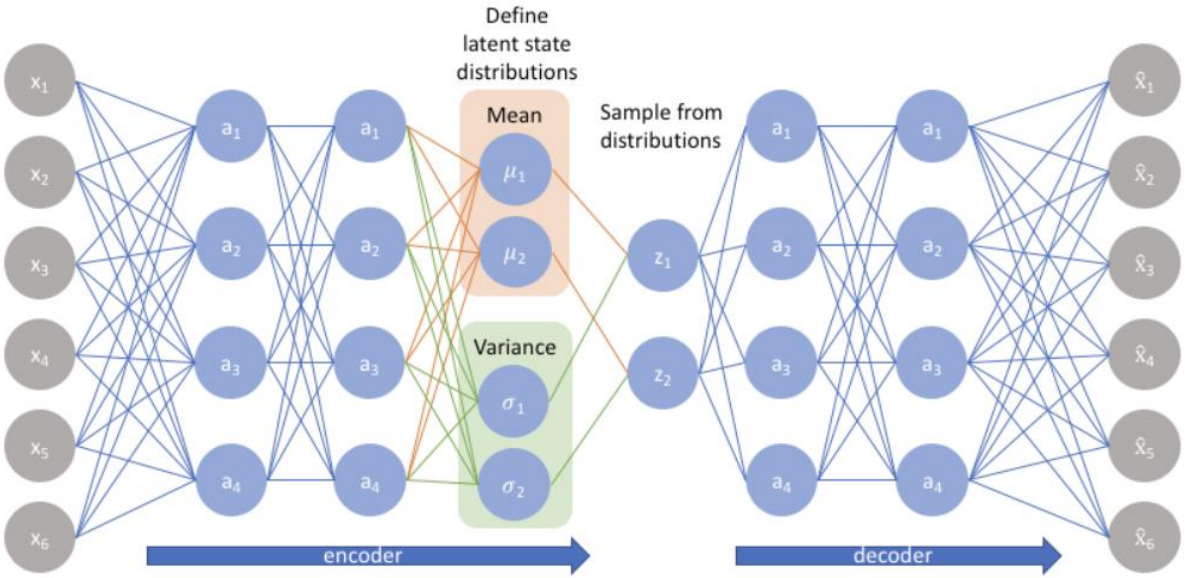
Denoising Autoencoder

- **Denoising Autoencoder**
  - Add noise to the input and learn to reconstruct the original input without noise
  - Noise can be generated in random input units using Gaussian Noise or Dropout.



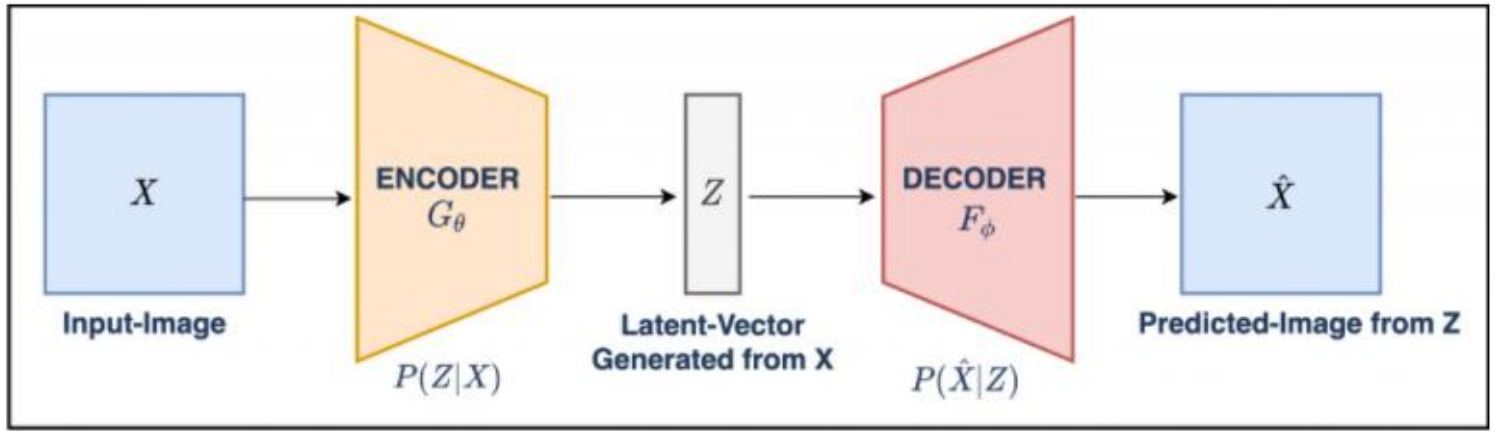
Variational Autoencoder

- **Variational Autoencoder (VAE)**
  - Provides a probabilistic manner for describing an observation in latent space
  - Encoder is formulated to describe a probability distribution for each latent attribute
  - Encoder is referred to as recognition model, and decoder is referred to as the generative model



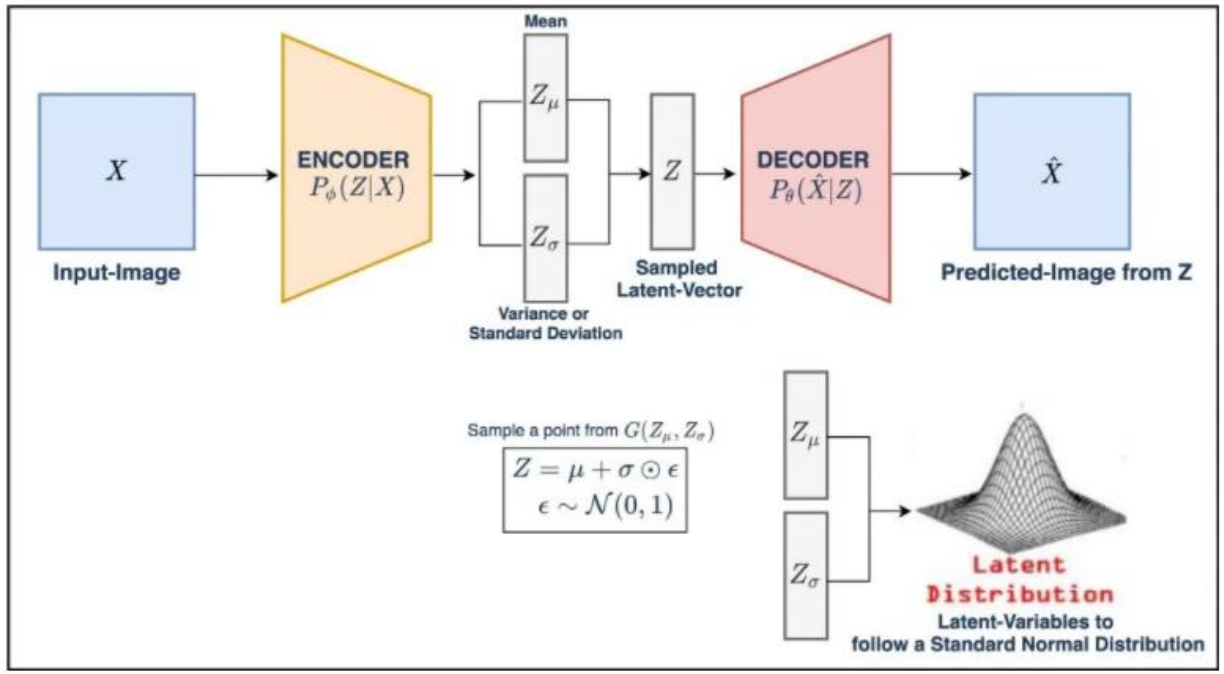
Autoencoders

- **Auto-Encoder**
  - **Reconstruction loss (MSE)** = $\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2$ (N is number of images in a batch)
  - It is not good at generating new images because of its latent space structure:
    - The latent space was not continuous and did not allow easy interpolation.
    - Encoded vectors are grouped in clusters corresponding to different data classes, and there are huge gaps between the clusters.
    - While generating a new sample, the decoder often produces a gibberish output if the chosen point in the latent space did not contain any data

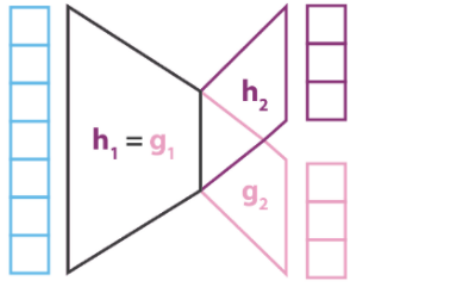


Variational Autoencoder (VAE)

- A probabilistic and generative model (focus on the decoder)
- Instead of mapping the image on a point in space, the encoder of VAE maps the image onto a normal distribution.
- Sample a latent vector Z from two latent variables Z_μ and Z_σ (also called a sampling-layer)
- We want the latent vector Z to follow a standard normal distribution. (Z_μ and Z_σ should be trained such that they are close to 0 and 1 respectively)

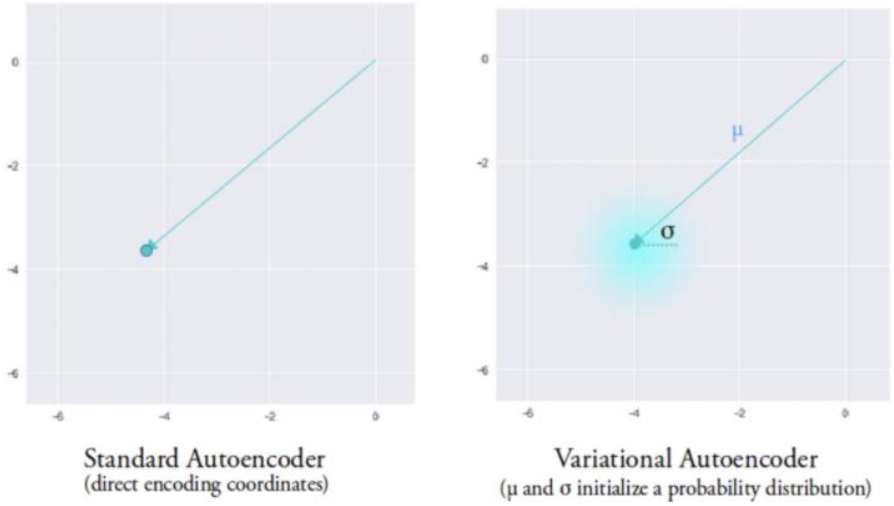


https://learnopencv.com/variational-autoencoder-in-tensorflow/



Variational Autoencoder (VAE)

- AE and VAE: Discrete latent vector and a standard normal distribution
  - AE: 연속성 보장이 안됨 – 생성 모델로 부적합
    - 단순히 입력을 압축한 벡터라서, 잠재 공간이 불연속적일 수 있다.
    - 특정 데이터 샘플 주변에만 의미 있는 벡터가 존재하고, 그 사이 공간은 의미 없는 값일 수 있다.
  - VAE: 연속성 보장
    - 잠재 공간을 정규분포에 맞추도록 KL 발산 손실을 추가.
    - 덕분에 잠재 공간 전체가 매끄럽게 연결되어 있어, 임의의 점을 샘플링해도 의미 있는 데이터가 생성.
- Why normal distribution?
  - 수학적 단순성
  - 재매개변수화 트릭(Reparameterization Trick)
  - 잠재 공간의 연속성
  - 샘플링 용이성



Variational Autoencoder (VAE)

- **Objective Function of VAE**
  - Unit Gaussian Distribution $\mathcal{N}(0,1)$, and
  - Minimize the reconstruction error $\frac{1}{N}\sum_{i=1}^{N}(X_i - \hat{X}_i)^2$.

- **VAE's total loss**
  - Loss = Re-construction_loss + KL_divergence_loss

$$L_{MSE}(\theta,\phi) = \frac{1}{N}\sum_{i=1}^{N}\left(x_i - f_\theta(g_\phi(x_i))\right)^2$$ or CE (Cross Entropy)

$$L_{KL}[G(Z_\mu, Z_\sigma)\|\mathcal{N}(0,1)] = -0.5 * \sum_{i=1}^{N} 1 + log(Z_{\sigma_i}^2) - Z_{\mu_i}^2 - Z_{\sigma_i}^2$$

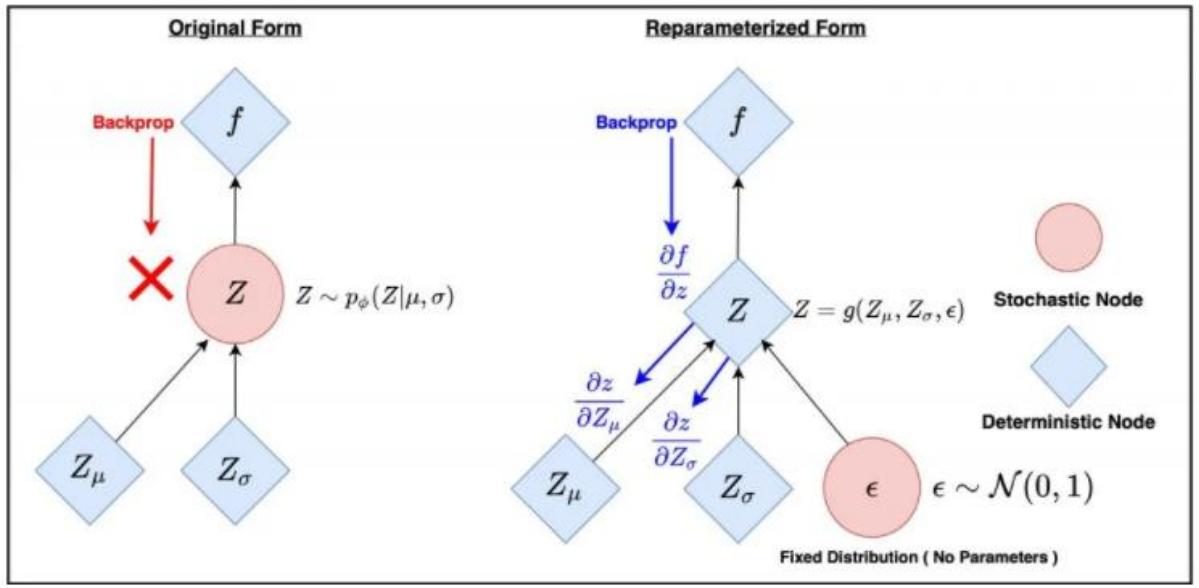
- MSE (continuous values): 입력 데이터가 **연속적인 값**을 가지는 경우 (예: 자연 이미지, 음성 신호)
- CE (discrete values): 입력 데이터가 **0~1 범위의 픽셀 값**을 가지는 경우 (예: MNIST 손글씨 이미지)
- 주로 Binary CE 를 사용함. (이 경우 yi 값은 0,1 이 아닌 확률 기대값으로 해석) (예: yi=0.7: 해당 픽셀이 " 1 " 일 확률이 70%)

$$\mathcal{L}_{\text{BCE}} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right]$$

Variational Autoencoder (VAE)

- **Re-parameterization Trick**
  - Sampling prevents backpropagation and their training. (샘플링 $z \sim N(\mu, \sigma^2)$ 은 미분 불가능 연산)
  - **Trick**: convert the random node $\boldsymbol{Z}$ to a deterministic node ($\boldsymbol{Z}_\mu$ and $\boldsymbol{Z}_\sigma$ remain as the learnable parameters while still maintaining the stochasticity of the entire system via ε.)

$$Z = Z_\mu + Z_\sigma^2 \odot \varepsilon \qquad (\text{Here, } \varepsilon \sim \mathcal{N}(0,1) \text{ and } \odot \text{ is element-wise multiplication.})$$



Variational Autoencoder (VAE)

- **KL Divergence**
  - 두 확률 분포의 다른 정도 (= Relative Entropy)
  - 두 분포가 Gaussian 일 경우 간략하게 표현 가능

$$q_\theta(z|x_i) \rightarrow \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right)$$

$$p(z) \rightarrow \frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(-\frac{(x-\mu_p)^2}{2\sigma_p^2}\right)$$

$$\sigma_p = 1 \text{ and } \mu_p = 0.$$

$$-D_{KL}(q_\theta(z|x_i)||p(z)) =$$

$$\int \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right) \log\left(\frac{\frac{1}{\sqrt{2\pi\sigma_p^2}} \exp\left(-\frac{(x-\mu_p)^2}{2\sigma_p^2}\right)}{\frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right)}\right) dz$$

$$= \int \frac{1}{\sqrt{2\pi\sigma_q^2}} \exp\left(-\frac{(x-\mu_q)^2}{2\sigma_q^2}\right) \times$$

$$\left\{-\frac{1}{2}\log(2\pi) - \log(\sigma_p) - \frac{(x-\mu_p)^2}{2\sigma_p^2} + \frac{1}{2}\log(2\pi) + \log(\sigma_q) + \frac{(x-\mu_q)^2}{2\sigma_q^2}\right\} dz$$

$$-D_{KL}(q_\theta(z|x_i)||p(z)) = \log(\sigma_q) - \frac{\sigma_q^2 + \mu_q^2}{2} + \frac{1}{2}$$

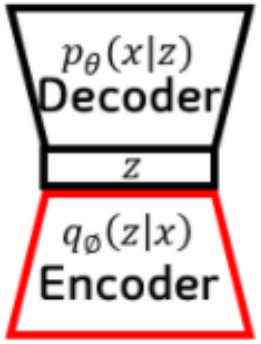
$$= \frac{1}{2}\log(\sigma_q^2) - \frac{\sigma_q^2 + \mu_q^2}{2} + \frac{1}{2}$$

$$= \frac{1}{2}\left[1 + \log(\sigma_q^2) - \sigma_q^2 - \mu_q^2\right]$$

Variational Autoencoder (VAE)

- **VAE Loss function**

$$L_i(\phi, \theta, x_i) = \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}[\log(p_\theta(x_i|z))]}_{\text{Reconstruction Error}} + \underbrace{KL(q_\phi(z|x_i)||p(z))}_{\text{Regularization}}$$



$$\mathcal{L} = -\sum_{j=1}^{J} \frac{1}{2}\left[1 + \log\left(\sigma_i^2\right) - \sigma_i^2 - \mu_i^2\right] - \frac{1}{L}\sum_{l} E_{\sim q_\emptyset(z|x_i)}\left[\log p(x_i|z^{(i,l)})\right]$$

$$(\theta^*, \phi^*) = argmin_{(\theta,\phi)}\mathcal{L}(\theta, \phi)$$

Cross entropy

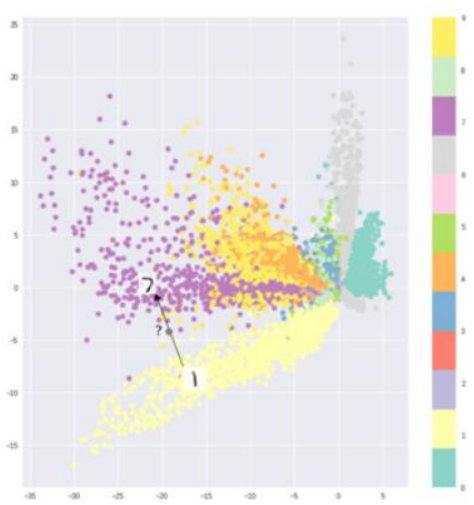
L: number of samples
J: dim of latent vector z

Ref: https://arxiv.org/pdf/1907.08956.pdf

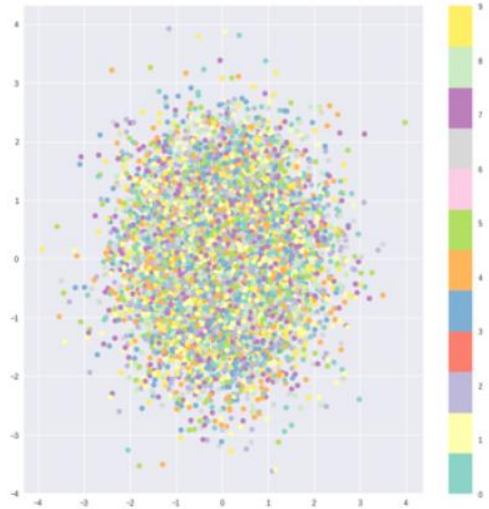
Variational Autoencoder (VAE)

$$\mathcal{L}(x, \hat{x}) + \beta \sum_j KL(q_j(z|x) || N(0,1))$$

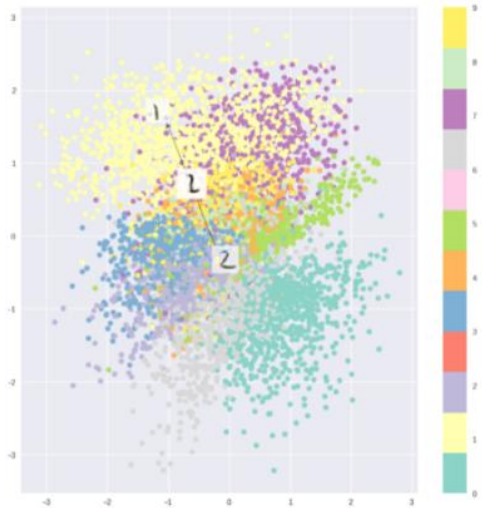
Only reconstruction loss



Only KL divergence



Combination



- Focus only on reconstruction loss, and allows the decoder be able to reproduce the original handwritten digits.
- there are areas in latent space which don't represent *any* of our observed data

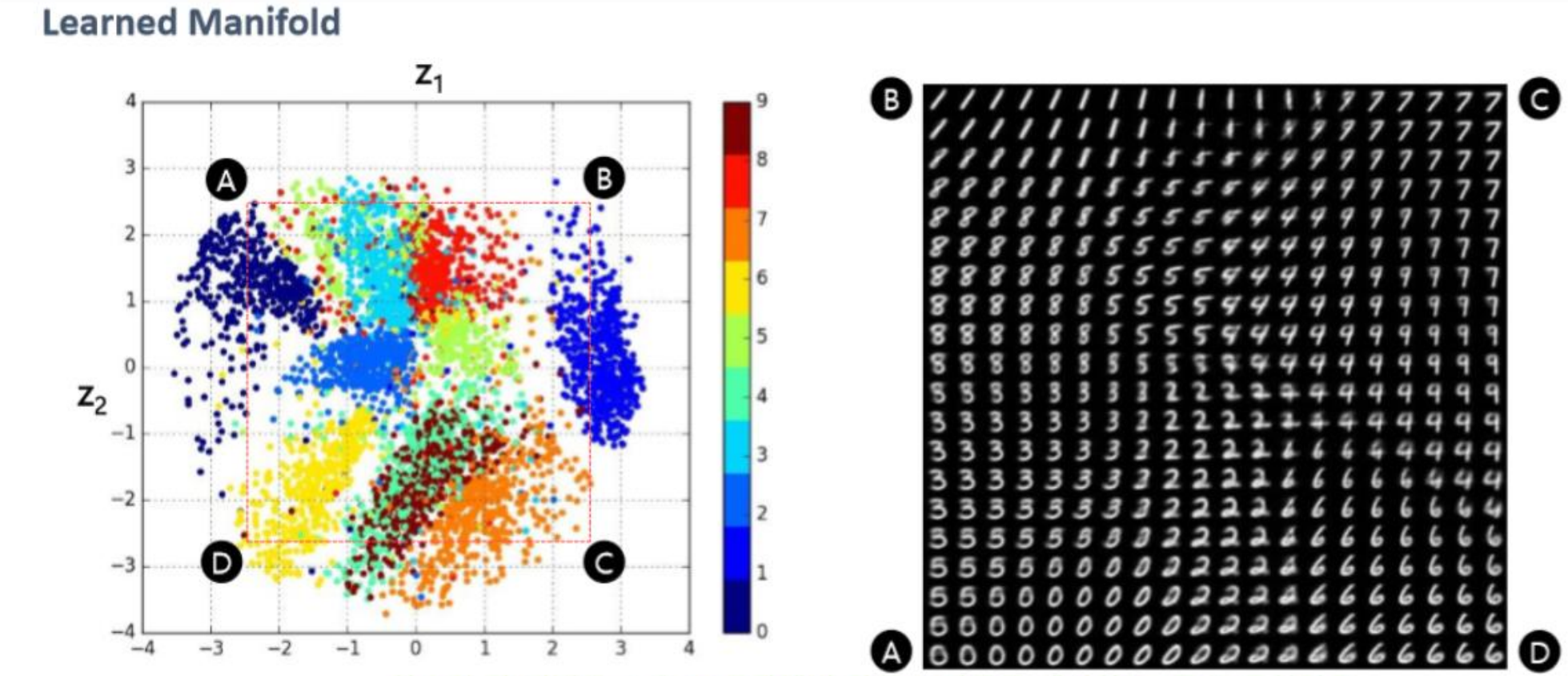
- Focus only on KL divergence loss, and describe every observation using the same Gaussian (e.g. same characteristics)
- failed to describe the original data

https://www.jeremyjordan.me/variational-autoencoders/

Variational Autoencoder (VAE)

- **An example of well-trained VAE with MNIST dataset**

$$z = \mu + \sigma \cdot \epsilon, \quad \epsilon \sim \mathcal{N}(0, 1)$$



학습이 잘 되었을 수록 2D공간에서 같은 숫자들을 생성하는 z들은 뭉쳐있고,
다른 숫자들은 생성하는 z들은 떨어져 있어야 한다.

https://github.com/hwalsuklee/tensorflow-mnist-VAE

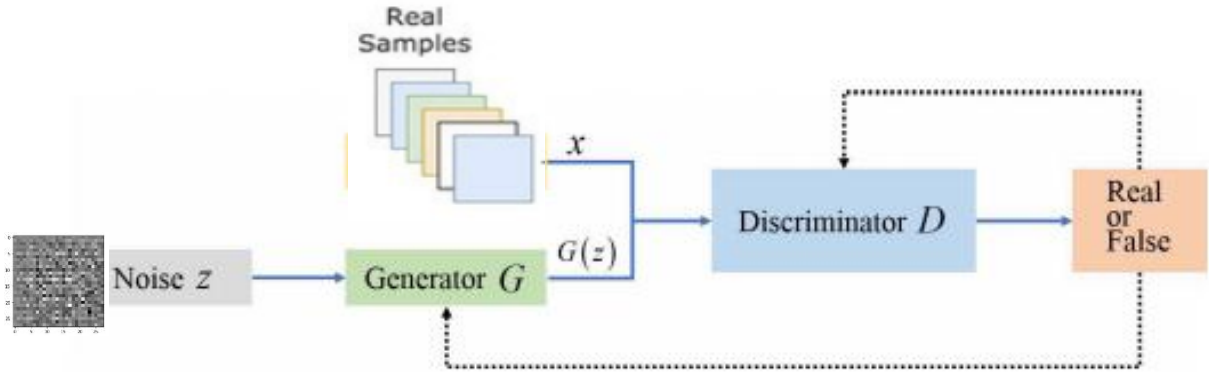
Variational Autoencoder (VAE)

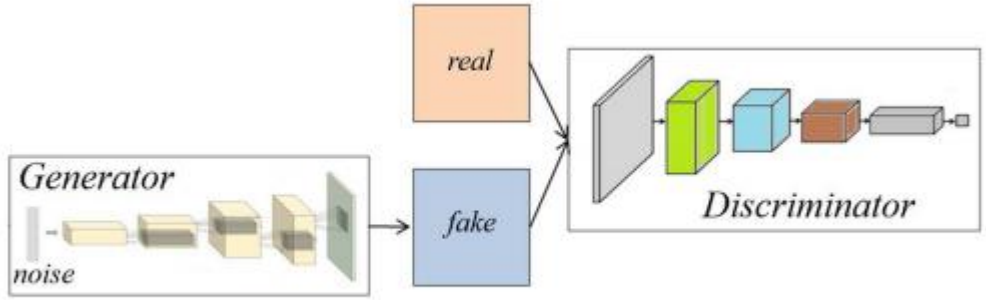
- **Another example**
  - Gradually changing hair color and sunglasses to a face
  - https://github.com/compthree/variational-autoencoder

- ε를 고정하면: 같은 입력 x에 대해 항상 동일한 z가 나오므로, 결과 이미지가 안정적으로 재현.
- ε를 바꾸면: 같은 입력이라도 다른 잠재 벡터가 생성되어, 다양한 이미지 변형을 얻을 수 있다.
- ε의 크기를 조절하면:
  - 작은 ε → μ 근처의 안정적인 샘플 → 원본과 유사한 이미지
  - 큰 ε → 잠재 공간의 더 먼 영역 → 원본과 다른, 창의적인 이미지

GAN

- **Generative Adversarial Network (GAN)**
  - Loss function: to make real(D(x)) close to 1, and fake(D(G(z)) close to 0
  - Discriminator: binary classifier (1: real, 0: fake)
  - Two separated optimizers





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

$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1-y_i)\cdot log(1-p(y_i))$$

yi=1 의 loss function

yi=0 의 loss function

GAN – Loss function

1.1 판별자 (D)

- **역할:** 실제 데이터와 생성된 데이터를 구분.
- **목표:** 실제 데이터 x에 대해 $D(x) = 1$, 생성 데이터 $G(z)$에 대해 $D(G(z)) = 0$.
- **손실 함수:**

$$\mathcal{L}_D = -\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

실제 데이터 샘플 x 에 대해 D(x) 가 1에 가까워지도록 학습

생성 데이터 G(z) 에 대해 D(G(z)) 가 0에 가까워지도록 학습

1.2 생성자 (G)

- **역할:** 실제처럼 보이는 데이터를 생성하여 D를 속임.
- **목표:** 생성된 데이터 $G(z)$가 실제 데이터처럼 보이도록 $D(G(z)) = 1$로 만들기.
- **손실 함수:**

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z}[\log D(G(z))]$$

생성자는 판별자를 속이는 게 목표이므로 D(G(z)) 가 1에 가까워지도록 학습 (클래스 0 은 생성자 입장에서는 고려할 필요가 없음)

GAN – Loss function

$$\min_G \max_D \mathcal{L}(D, G)$$

$$\mathcal{L}(D, G) = \mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] + \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

Discriminator: $\max_D \mathcal{L}(D, G)$

Generator: $\min_G \mathcal{L}(D, G)$

실제 학습 (Minimize)

Discriminator Loss ($\mathcal{L}_D$):

$$\mathcal{L}_D = -\mathbb{E}_{x \sim p_{\text{data}}}[\log D(x)] - \mathbb{E}_{z \sim p_z}[\log(1 - D(G(z)))]$$

Generator Loss ($\mathcal{L}_G$):

$$\mathcal{L}_G = -\mathbb{E}_{z \sim p_z}[\log D(G(z))]$$

생성자 (G) loss:

- **원래 minimax 공식:**

$$L_G = E_{z \sim p_z(z)}[\log(1 - D(G(z)))]$$

- **Non-saturating Loss (더 안정적):**

$$L_G = -E_{z \sim p_z(z)}[\log D(G(z))]$$

GAN

- **Generative Adversarial Network (GAN)**

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

```
def discriminator_loss(real_output, fake_output):
    real_loss = cross_entropy(tf.ones_like(real_output), real_output)
    fake_loss = cross_entropy(tf.zeros_like(fake_output), fake_output)
    total_loss = real_loss + fake_loss
    return total_loss
```

true

pred

```
def generator_loss(fake_output):
    return cross_entropy(tf.ones_like(fake_output), fake_output)
```

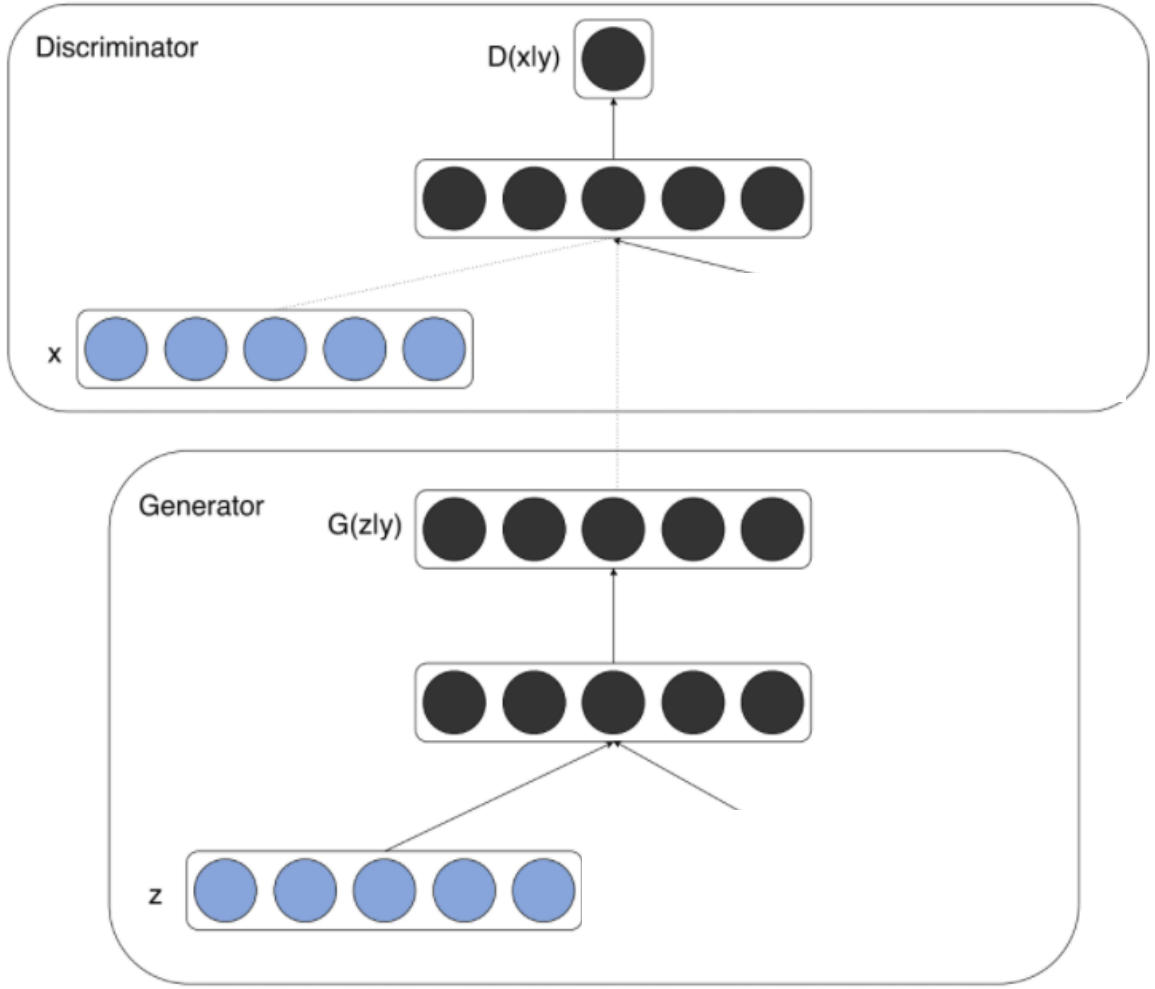
```
generator_optimizer = tf.keras.optimizers.Adam(1e-4)
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
```

[ref: https://www.tensorflow.org/tutorials/generative/dcgan?hl=ko]

(Unconditional) GAN

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

- Expectation
- x is sampled from real data
- Probability of D(real)
- z is sampled from N(0, 1)
- Probability of D(fake)
- fake



Example of a Conditional Generator and a Conditional Discriminator in a Conditional Generative Adversarial Network. Taken from Conditional Generative Adversarial Nets, 2014.

Conditional GAN (cGAN)

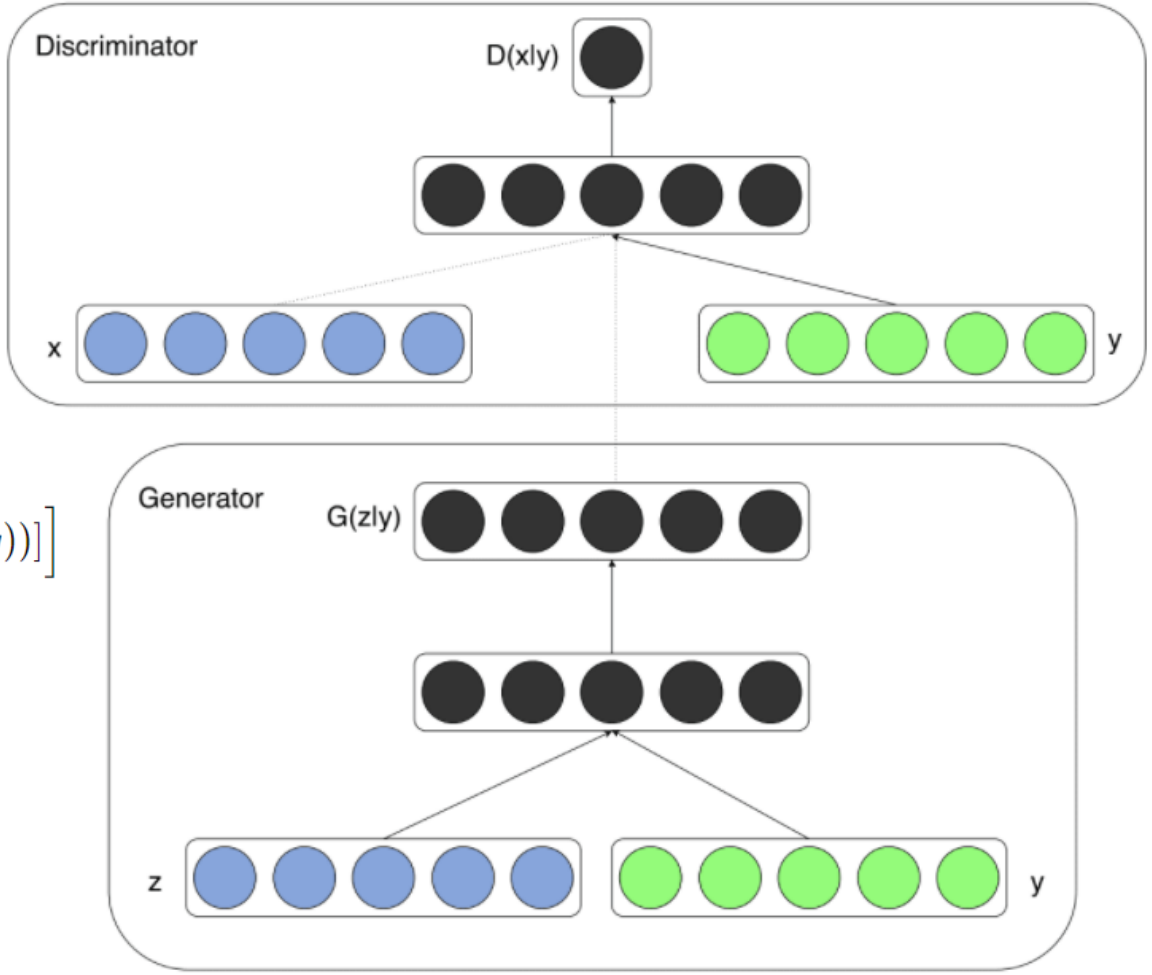
조건부 제어 생성

- y: 생성 과정에서 조건으로 제공되는 정보(예: 클래스 레이블).
- $G(z, y)$: 조건 y에 따라 생성된 데이터 $G(z)$.
- $D(x, y)$: 데이터 x와 조건 y가 일치하는지 판별.

$$\mathcal{L}_{\text{CGAN}} = \min_G \max_D \left[\mathbb{E}_{x,y\sim p_{\text{data}}}[\log D(x,y)] + \mathbb{E}_{z\sim p_z, y\sim p_y}[\log(1 - D(G(z,y),y))] \right]$$

D(x,y): 조건 y에 맞는 진짜/가짜 판별

G(x,y): "노이즈 z와 조건 y를 받아서, 조건에 맞는 이미지 생성



Example of a Conditional Generator and a Conditional Discriminator in a Conditional Generative Adversarial Network.
Taken from Conditional Generative Adversarial Nets, 2014.