

# Generative Models

2021. 8

Yongjin Jeong, KwangWoon University

[참고] 본 자료에는 인터넷에서 다운받아 사용한 그림이나 수식들이 일부  
있으니 다른 용도로 사용하거나 외부로 유출을 금해 주시기 바랍니다.

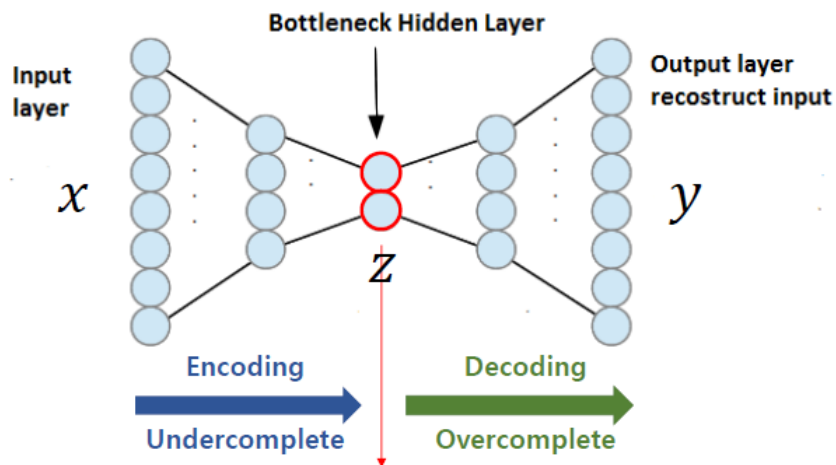
# Contents

- Autoencoders
- Variational Autoencoder (VAE)
- GANs

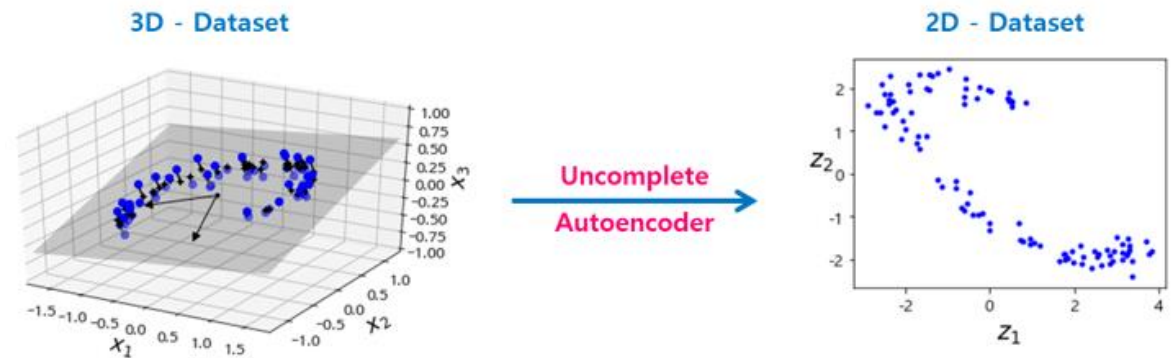
# Autoencoders

- **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.



- Code
- Latent Variable
- Feature
- Hidden representation

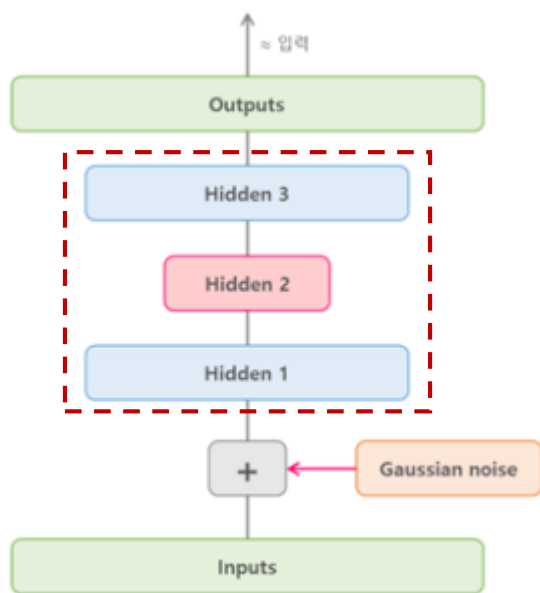


# Autoencoders

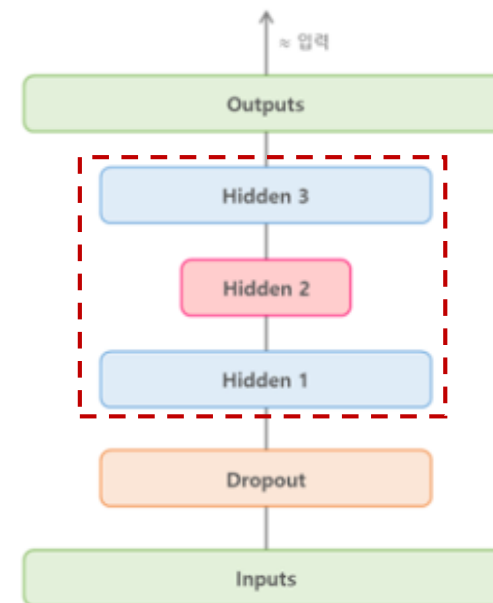
- **Applications of Autoencoders**
  - Dimensionality reduction
  - Anomaly detection
  - Image denoising
  - Image compression
  - Image generation

# 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.



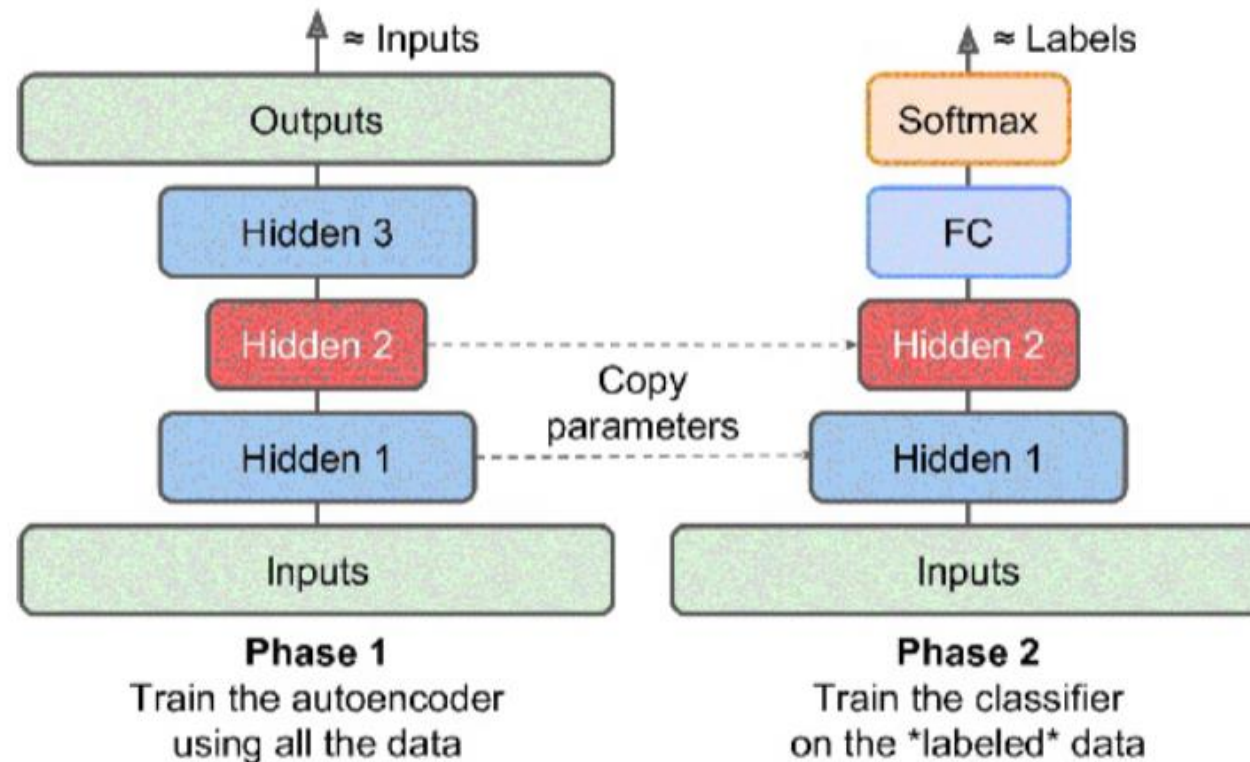
Denoising Autoencoder  
With Gaussian noise



Denoising Autoencoder  
With Dropout

# Stacked (or deep) Autoencoder

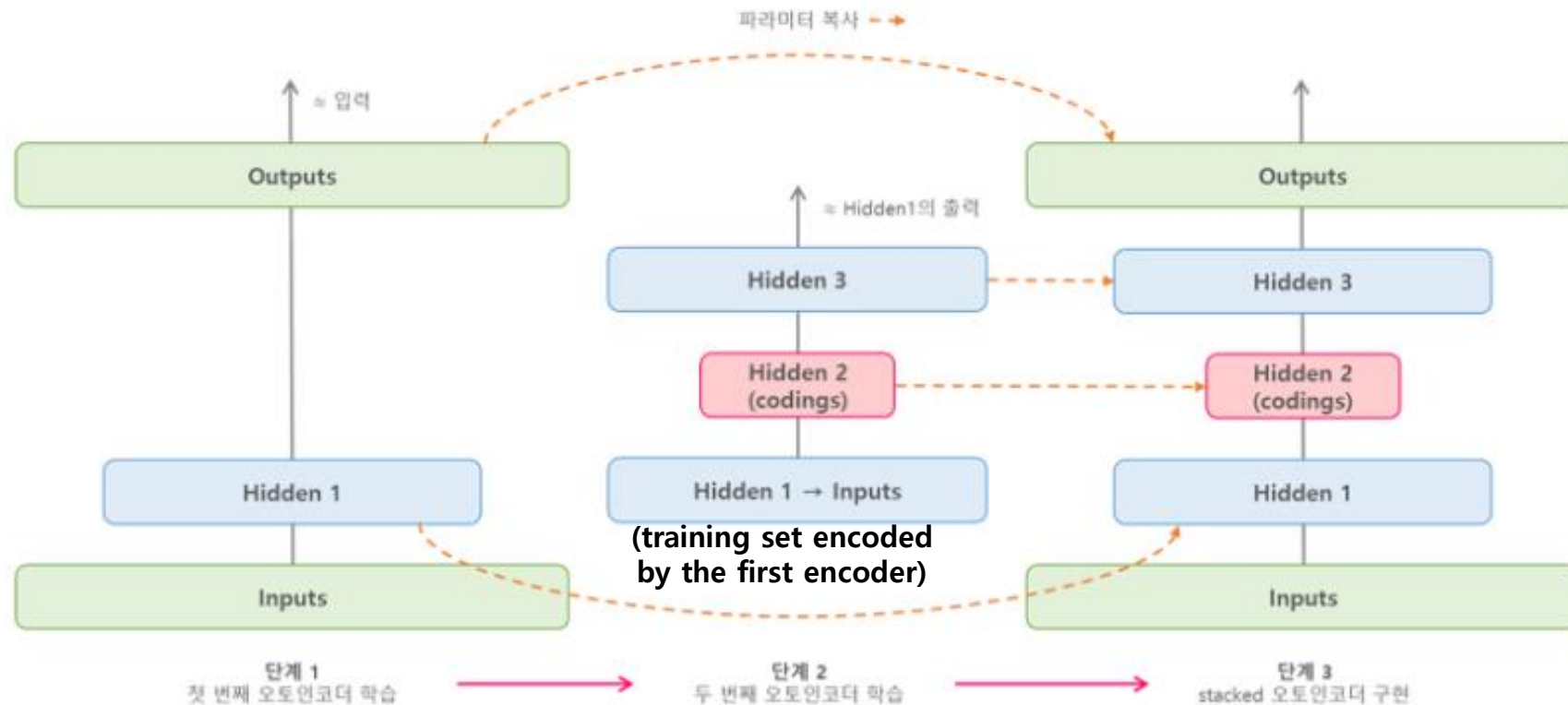
- Unsupervised **Pre-training** using Stacked Autoencoders
  - When you have a complex supervised task but you do not have enough labeled training data (labeling is time-consuming and costly)



# Stacked Autoencoder

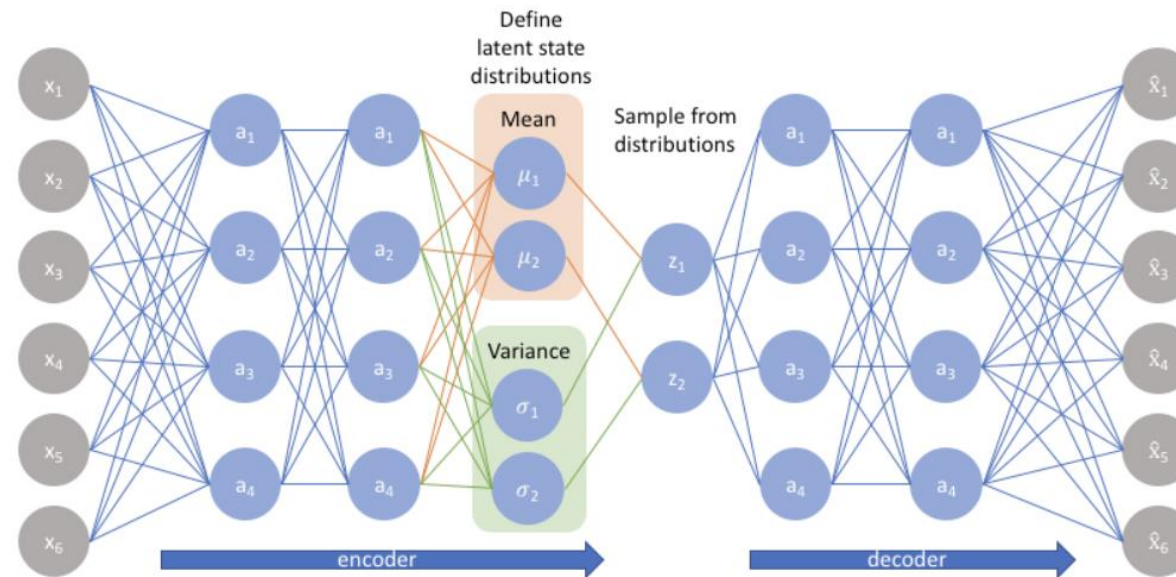
- **Training One Auto-encoder at a time**

- Rather than training the whole stacked auto-encoder in one go, it is much faster to train one shallow auto-encoder at a time, then stack all of them into a single. (not used as much these days)



# Deep Learning

- **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**

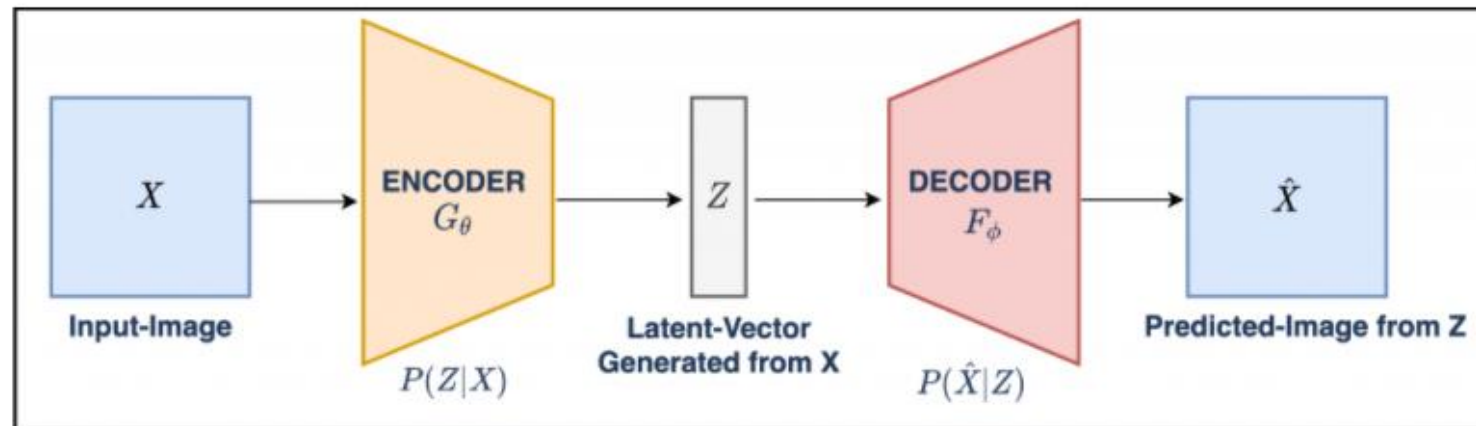




# Autoencoders

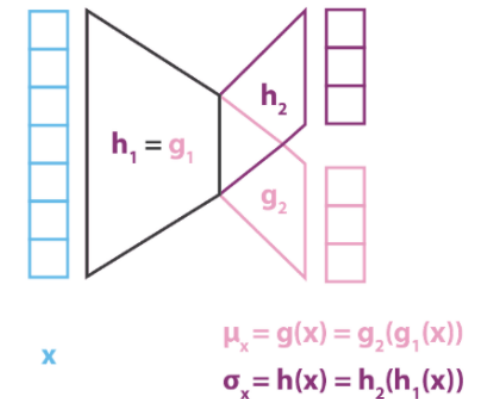
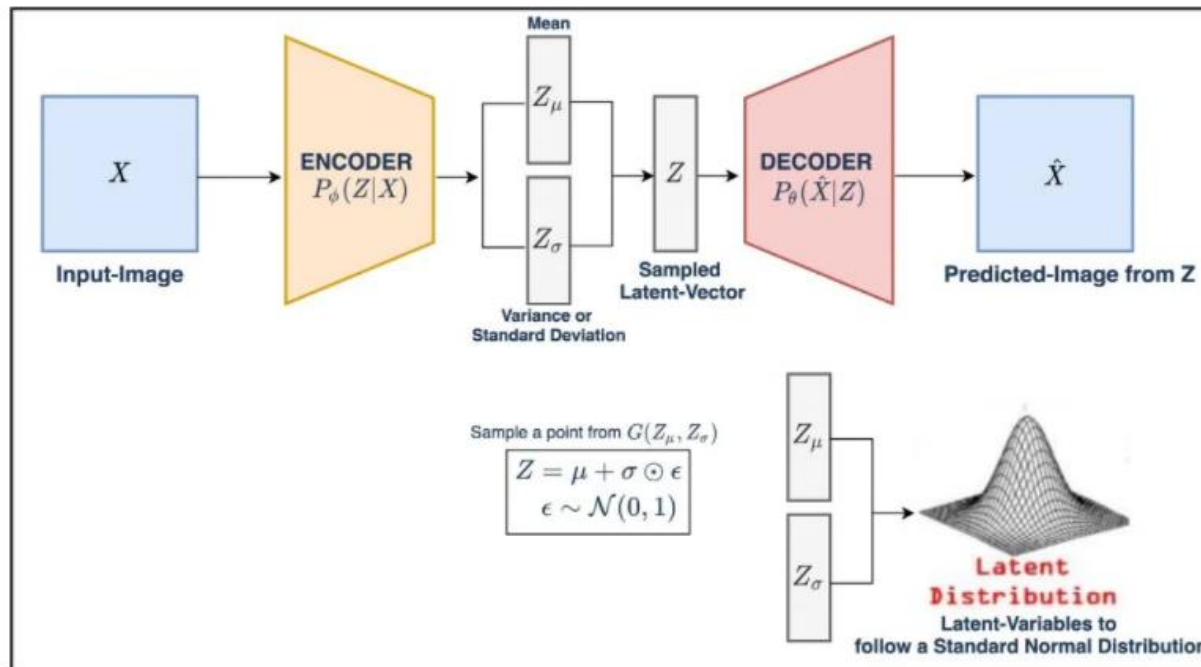
- **Auto-encoder**

- **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



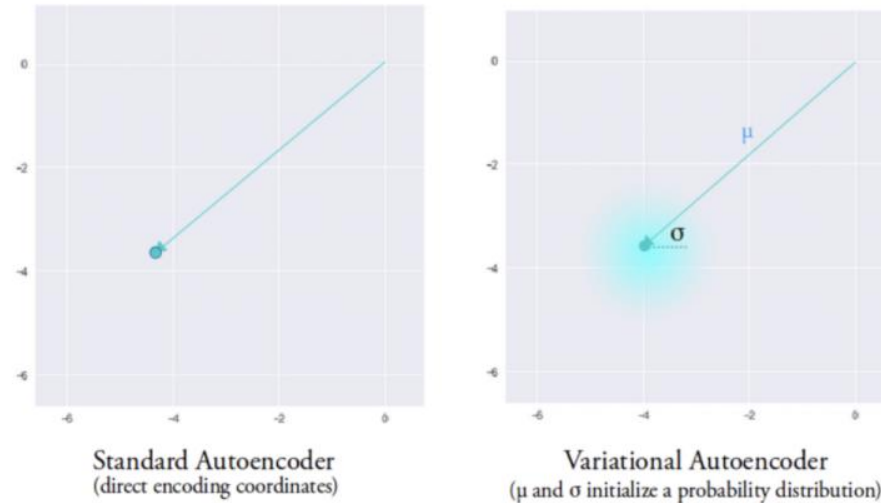
# Variational Autoencoder (VAE)

- 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  $Z_\mu$  and  $Z_\sigma$  (also called a **sampling-layer**)
- We want the latent vector  $Z$  to follow a standard normal distribution. ( $Z_\mu$  and  $Z_\sigma$  should be trained such that they are close to 0 and 1 respectively)



# Variational Autoencoder (VAE)

- **Discrete latent vector (Standard auto-encoder) and a standard normal distribution (VAE)**
  - Instead of a single point, the VAE covers a certain area: a sample from anywhere in the area will be very similar to the original input.



- **Why normal distribution?**
  - We want to push the auto-encoder to have codings that look as though they were sampled from a simple Gaussian distribution. (assume that our dataset would inherently follow a distribution similar to the normal distribution.)
  - Enforcing the latent variables to follow a normal distribution in VAE is very common and works the best.

# 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

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

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

```
kl_loss = -0.5 * numpy.sum(1 + numpy.log(Z_sigma ** 2) - numpy.square(Z_mean)
- numpy.exp(np.log(Z_sigma ** 2), axis = 1)

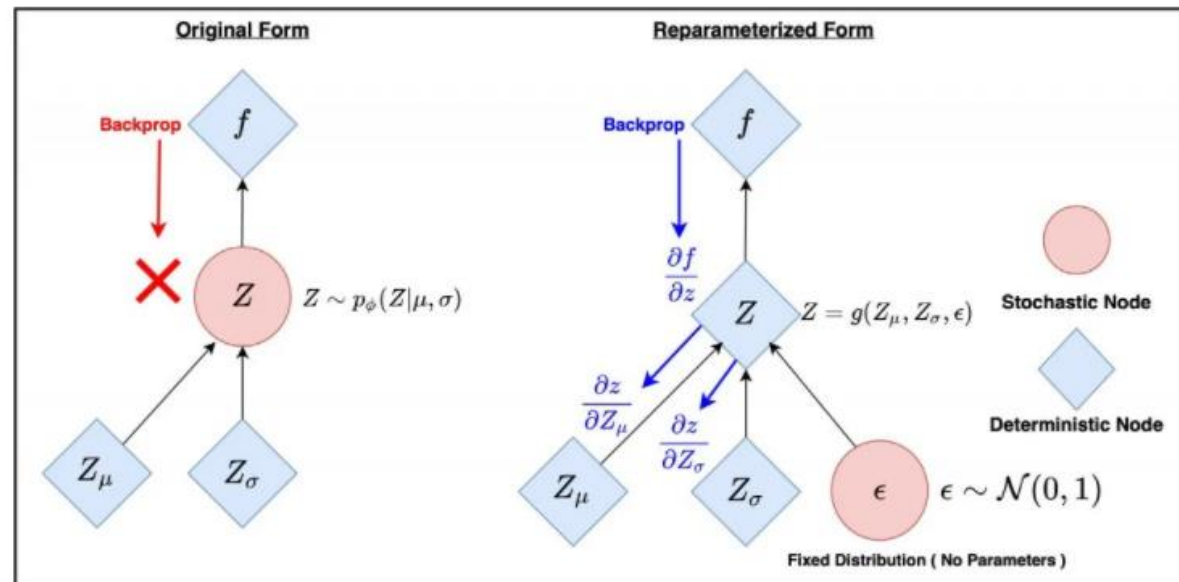
kl_loss = -0.5 * numpy.sum(1 + numpy.log(Z_sigma ** 2) - numpy.square(Z_mean)
- Z_sigma ** 2, axis = 1)
```

# Variational Autoencoder (VAE)

- **Re-parameterization Trick**

- Sampling prevents backpropagation and their training.
- **Trick:** convert the random node  $Z$  to a deterministic node ( $Z_\mu$  and  $Z_\sigma$  remain as the learnable parameters while still maintaining the stochasticity of the entire system via  $\epsilon$ .)

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



# Variational Autoencoder (VAE)

- KL Divergence

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

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

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

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

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

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

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

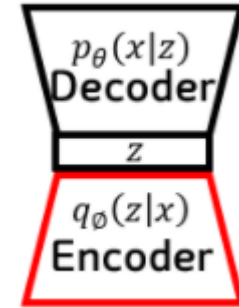


$$\begin{aligned} -D_{KL}(q_{\theta}(z|x_i)||p(z)) &= \log(\sigma_q) - \frac{\sigma_q^2 + \mu_q^2}{2} + \frac{1}{2} \\ &= \frac{1}{2} \log(\sigma_q^2) - \frac{\sigma_q^2 + \mu_q^2}{2} + \frac{1}{2} \\ &= \frac{1}{2} \left[ 1 + \log(\sigma_q^2) - \sigma_q^2 - \mu_q^2 \right] \end{aligned}$$

# Variational Autoencoder (VAE)

- VAE Loss function

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



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

$$(\theta^*, \phi^*) = \operatorname{argmin}_{(\theta, \phi)} \mathcal{L}(\theta, \phi)$$

Cross entropy

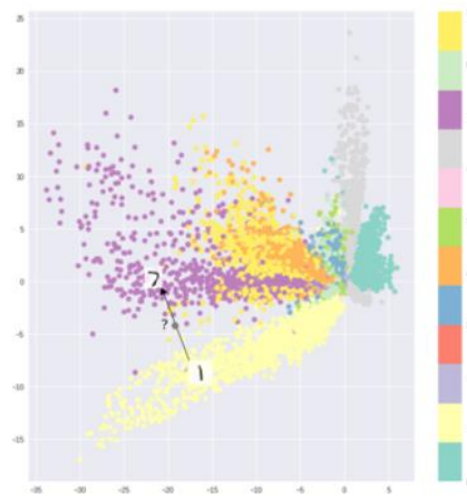
(note that  $\theta$  and  $\phi$  are used in reverse.)

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

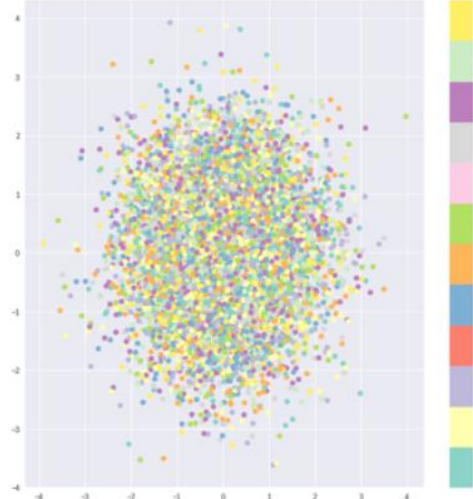
# Variational Autoencoder (VAE)

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

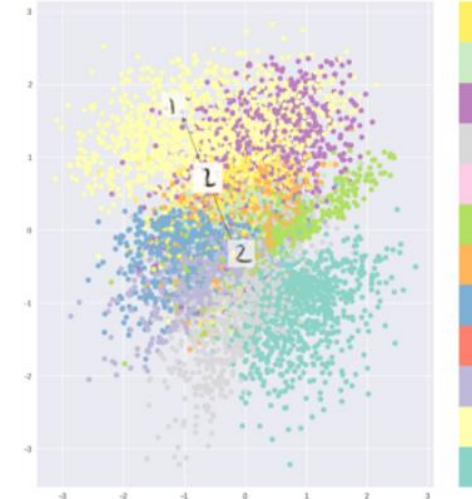
Only reconstruction loss



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

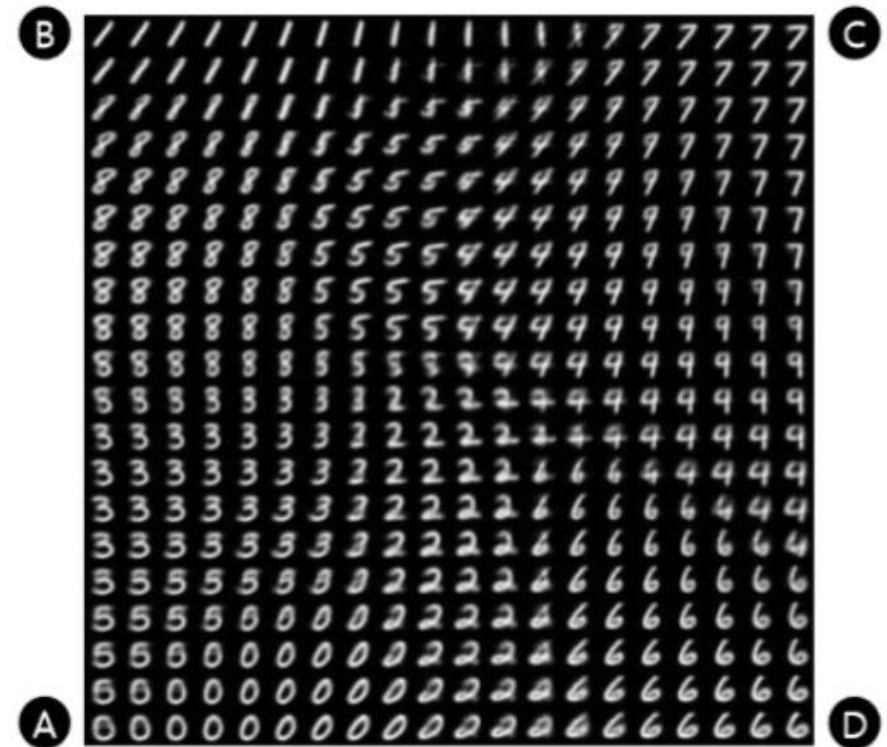
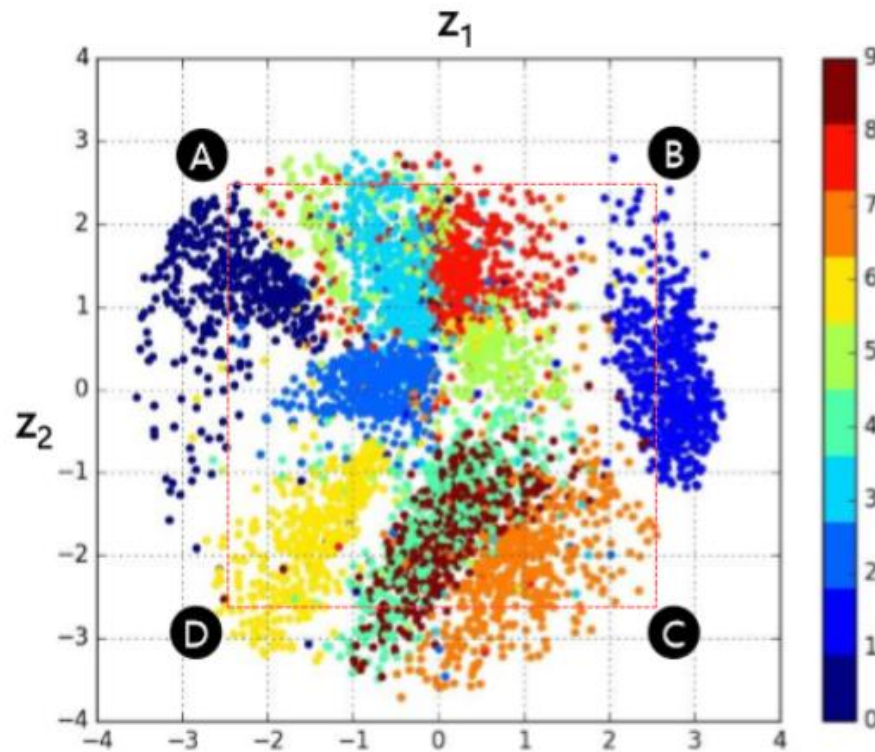
- Focus only on KL divergence loss, and describe every observation using the same Gaussian (e.g. same characteristics)
- failed to describe the original data



# Variational Autoencoder (VAE)

- An example of well-trained VAE with MNIST dataset

Learned Manifold



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

# Variational Autoencoder (VAE)

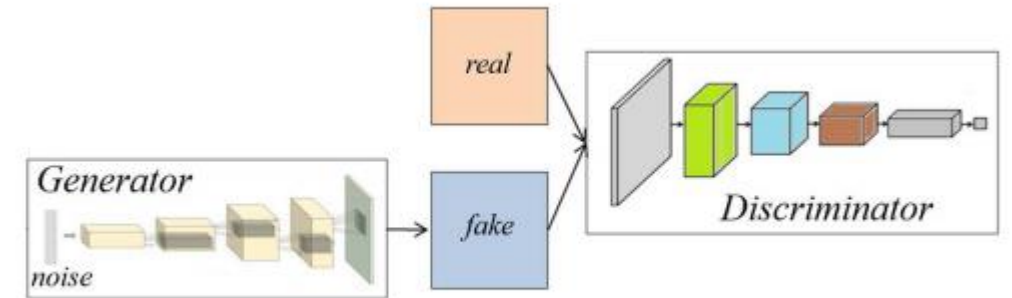
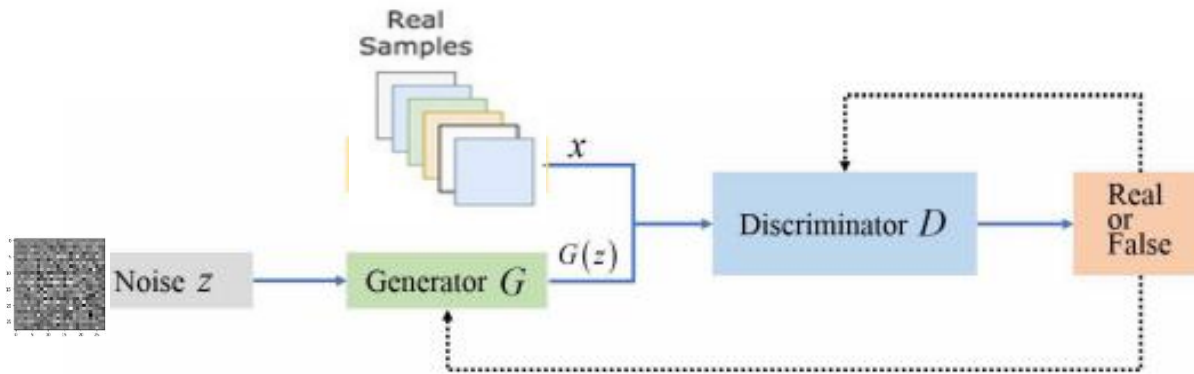
- **Another example**
  - Gradually changing hair color and sunglasses to a face
  - <https://github.com/comptthree/variational-autoencoder>



# Deep Learning

- **Generative Adversarial Network (GAN)**

- Loss function: to make  $\text{real}(D(x))$  close to 1, and  $\text{fake}(D(G(z)))$  close to 0
- Discriminator: binary classifier (1: real, 0: fake)
- Two separated optimizers



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

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

$y_i=1$  의 loss function

$y_i=0$  의 loss function

# GAN ?

- **Generative Adversarial Network (GAN)**

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

```
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)
```

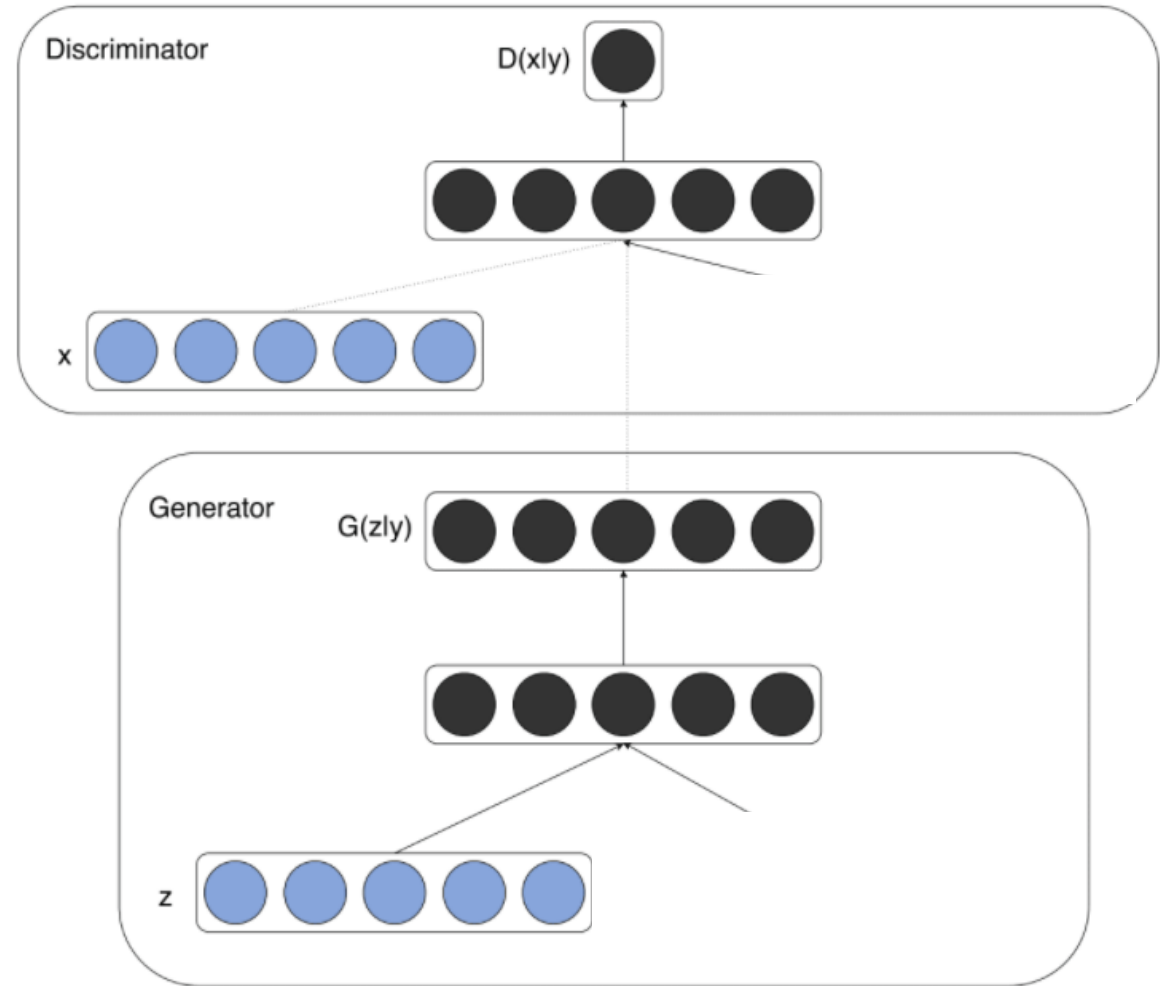
```
generator_optimizer = tf.keras.optimizers.Adam(1e-4)  
discriminator_optimizer = tf.keras.optimizers.Adam(1e-4)
```



# (Unconditional) GAN

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

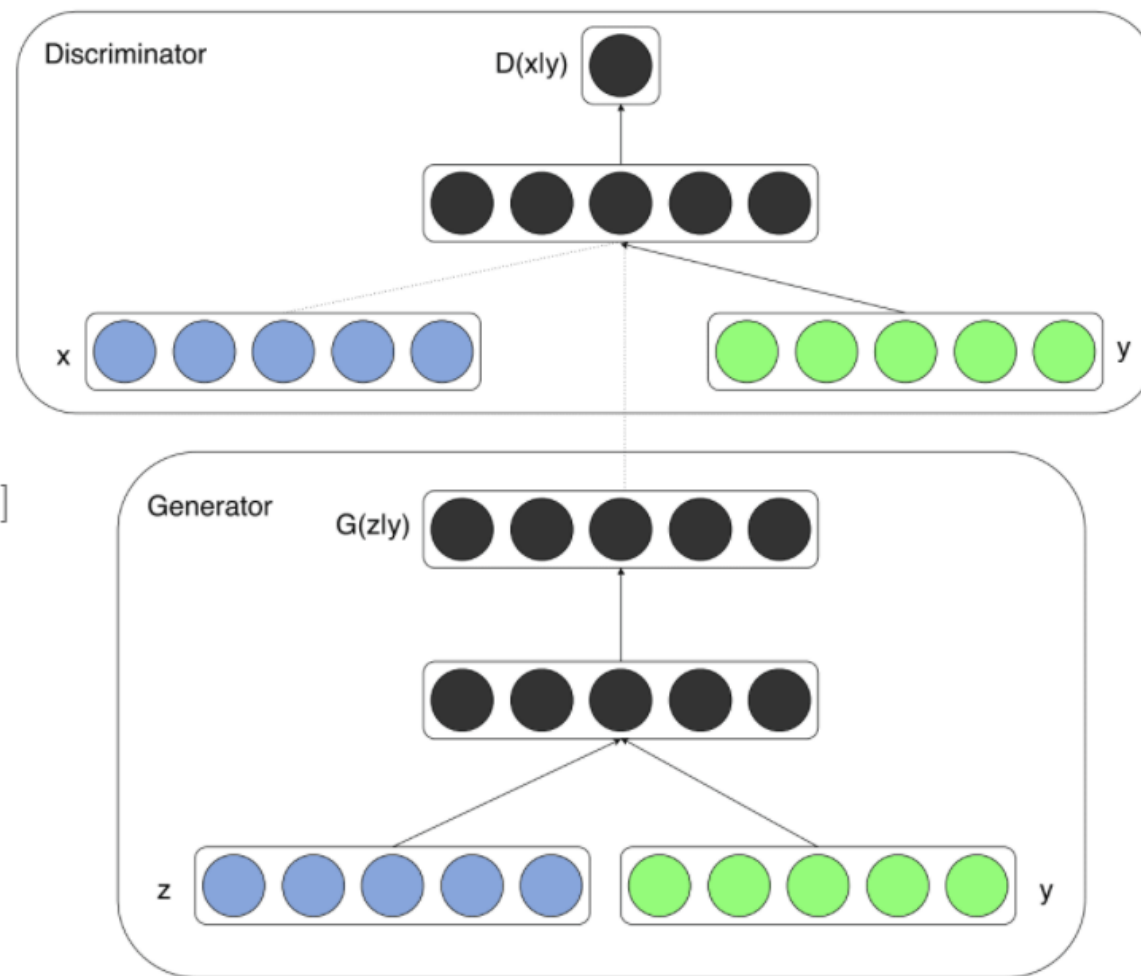
Expectation       $\mathbf{x}$  is sampled from real data      Probability of  $D(\text{real})$        $\mathbf{z}$  is sampled from  $N(0, 1)$       Probability of  $D(\text{fake})$       fake



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)

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



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