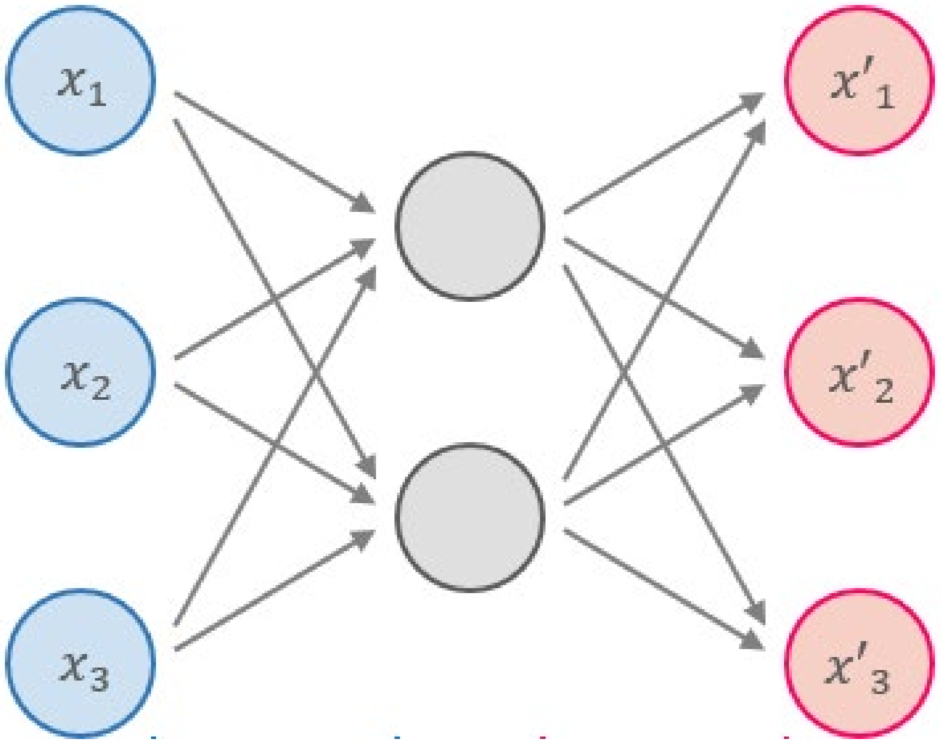
Auto Encoder
&
VAE

5기 남의서



00. 딥러닝이란?

머신러닝의 분야, 인공신경망의 층을
연속적으로 깊게 쌓아올려 데이터를 학습하는 알고리즘

01. Collect training data

$$x = \{x_1, x_2, \ldots, x_N\}$$
$$y = \{y_1, y_2, \ldots, y_N\}$$
$$\mathcal{D} = \{(x_1, y_1), (x_2, y_2) \ldots (x_N, y_N)\}$$

02. Define functions

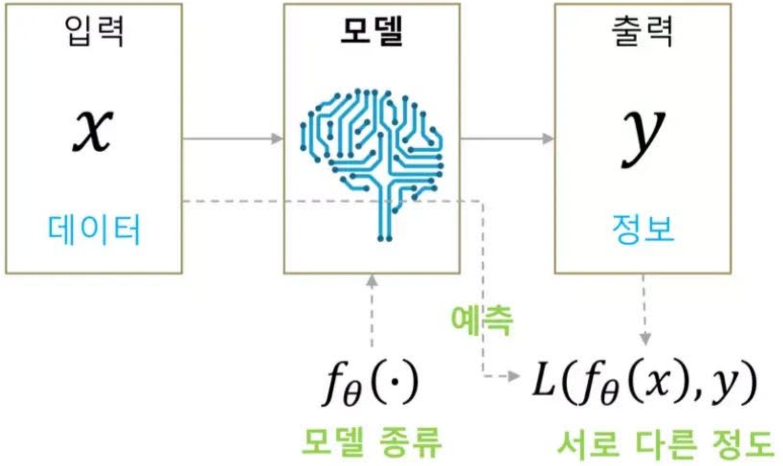
- Output : $f_\theta(x)$
- Loss : $L(f_\theta(x), y)$

03. Learning/Training

Find the optimal parameter

04. Predicting/Testing

Compute optimal function output



$$\theta^* = \underset{\theta}{\operatorname{argmin}} L(f_\theta(x), y)$$

주어진 데이터를 제일 잘
설명하는 모델 찾기

$$y_{new} = f_{\theta^*}(x_{new})$$

고정 입력, 고정 출력

00. 딥러닝이란?

머신러닝의 분야, 인공신경망의 층을
연속적으로 깊게 쌓아올려 데이터를 학습하는 알고리즘

ML PROBLEM | Deep Neural Networks

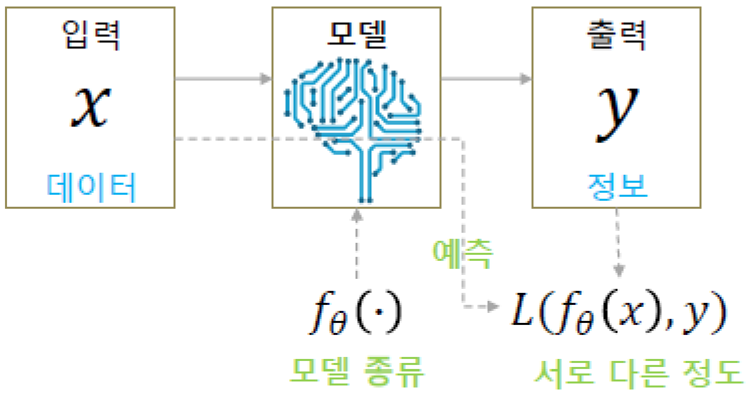
REVISIT DNN
2 / 17

01. Collect training data

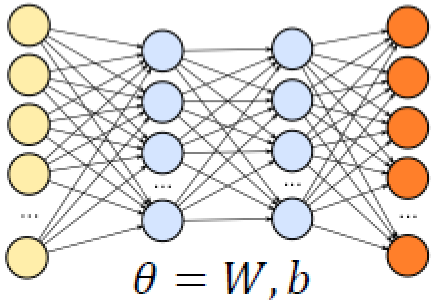
02. Define functions

03. Learning/Training

04. Predicting/Testing



$f_\theta(\cdot)$ Deep Neural Network



파라미터는 웨이트와 바이어스

$L(f_\theta(x), y)$ $L(f_\theta(x), y) = \sum_i L(f_\theta(x_i), y_i)$

Assumption 1.

Total loss of DNN over training samples is the sum of loss for each training sample

Assumption 2.

Loss for each training example is a function of final output of DNN

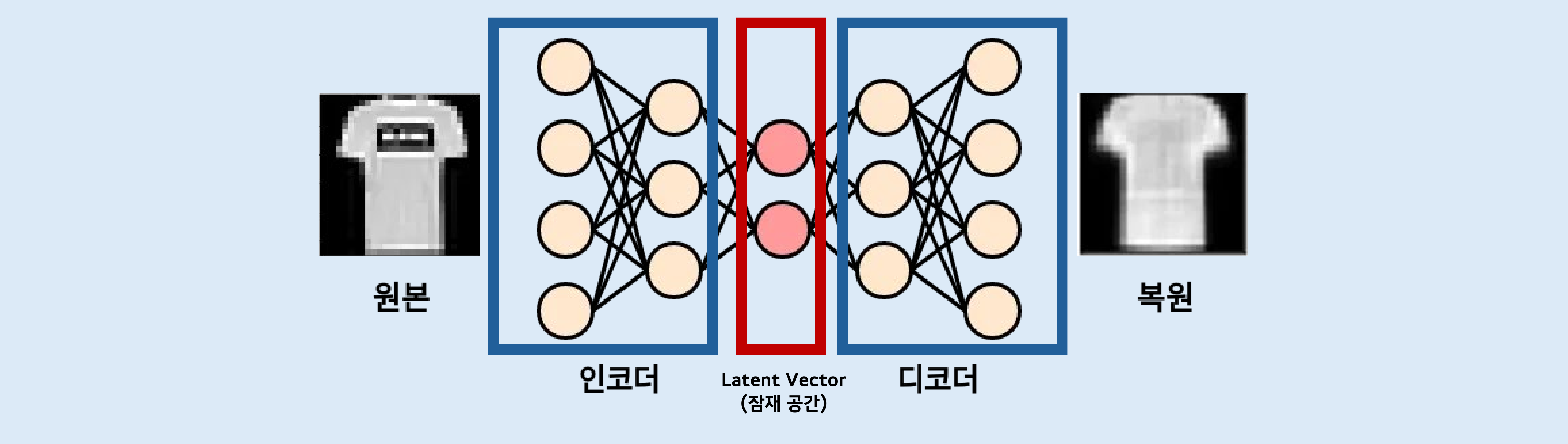
Backpropagation을 통해 DNN학습을 학습 시키기 위한 조건들

01. AutoEncoder

AutoEncoder는 인공 신경망(artificial neural network)으로서
efficient coding의 비지도 학습을 위해 사용됩니다.

AutoEncoder의 주요 목표는 일반적으로 차원 축소를 목적으로
데이터 집합에 대한 표현(encoding)을 학습하는 것입니다.
최근에는, AutoEncoder의 개념이 데이터의 생성모델을 학습하는데
광범위하게 사용되고 있습니다. (위키피디아)

01. AutoEncoder



02. Dimensionality Reduction

Dimensionality Reduction

Linear

- Principal Component Analysis (PCA)
- Linear Discriminant Analysis (LDA)
- etc..

Non-Linear

- Autoencoders (AE)
- t-distributed stochastic neighbor embedding (t-SNE)
- Isomap
- Locally-linear embedding (LLE)
- etc..

= Feature Extraction
= Manifold Learning
→ AutoEncoder의 encoder는
차원 축소의 역할을 수행

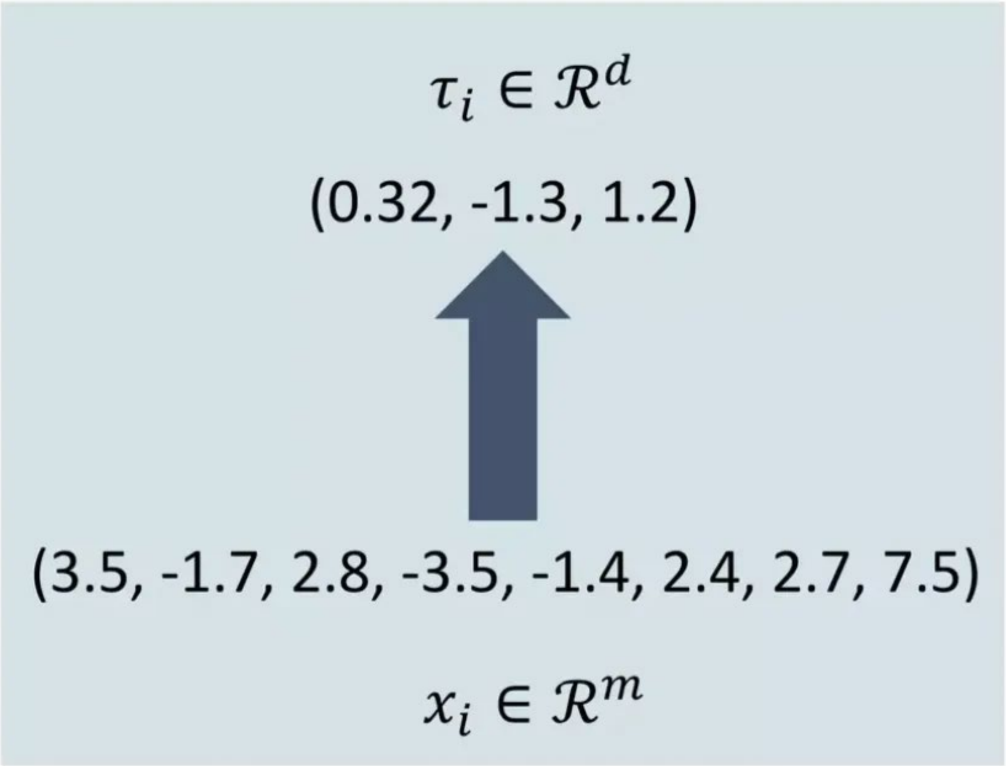
02. Dimensionality Reduction - Manifold

1. 데이터 압축

2. 데이터 시각화

3. 차원의 저주

4. 유용한 특징 추출하기



Dimensionality Reduction is an Unsupervised Learning Task!

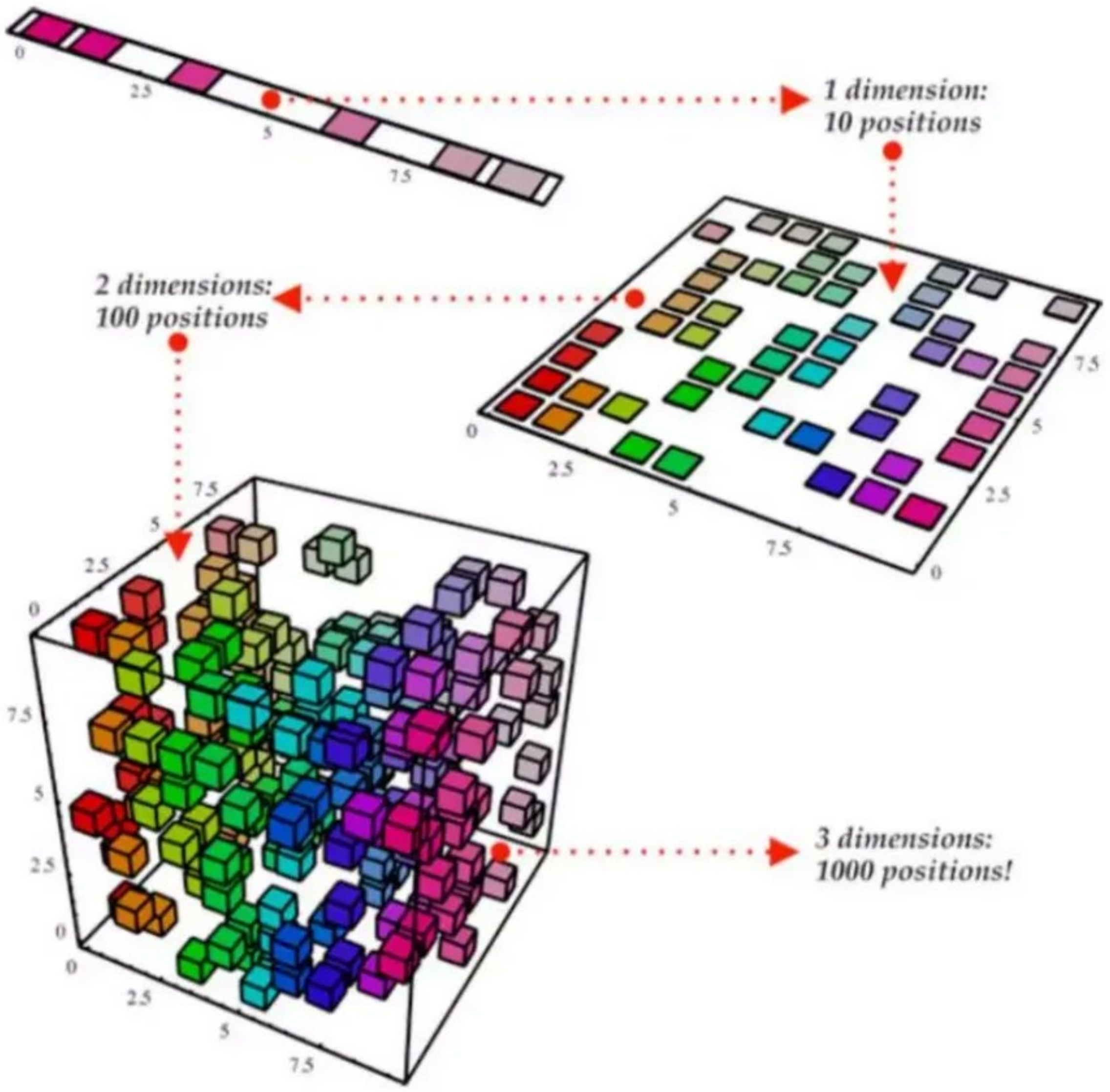
02. Dimensionality Reduction - Manifold

1. 데이터 압축

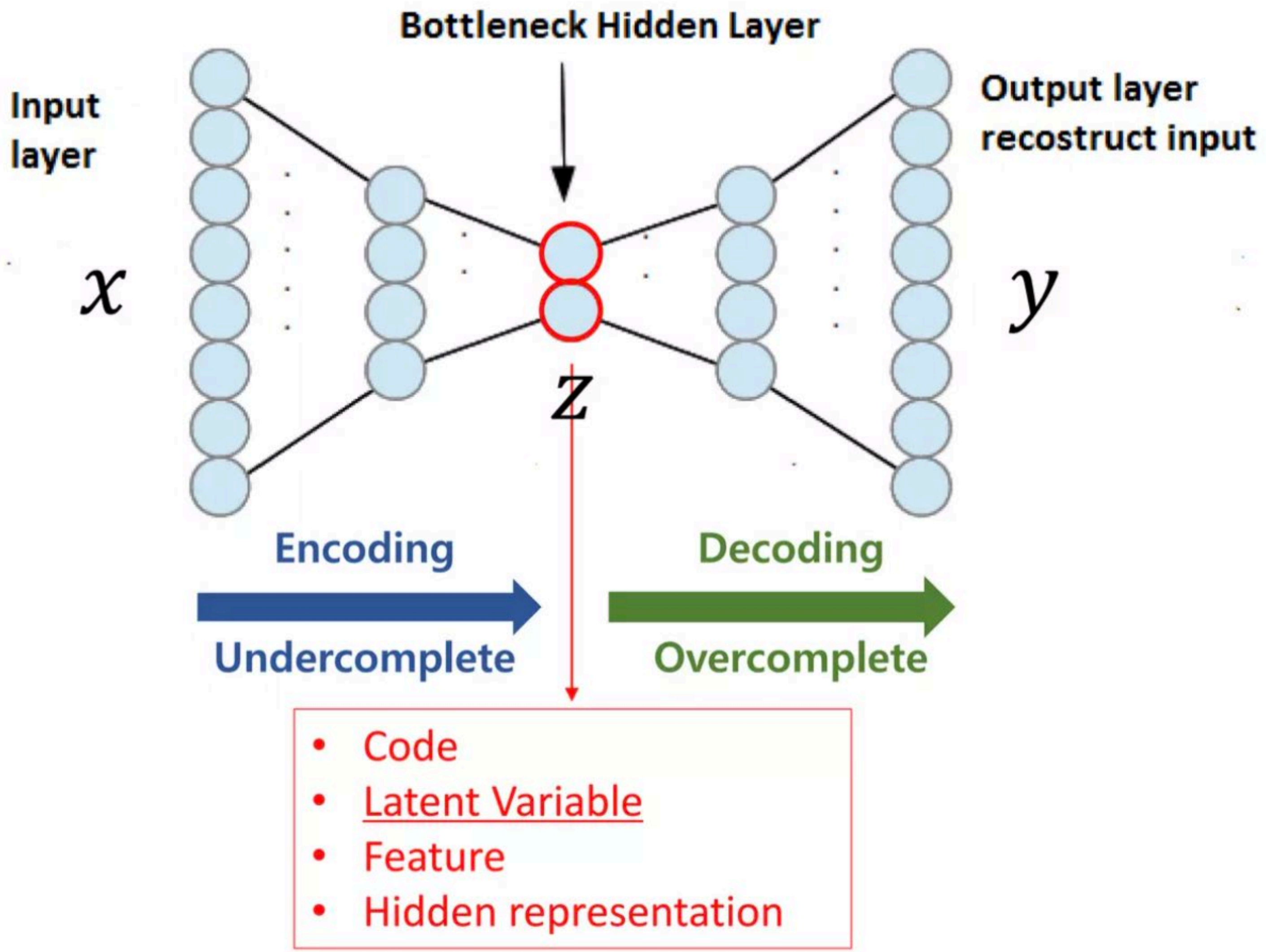
2. 데이터 시각화

3. 차원의 저주

4. 유용한 특징 추출하기



03. Function

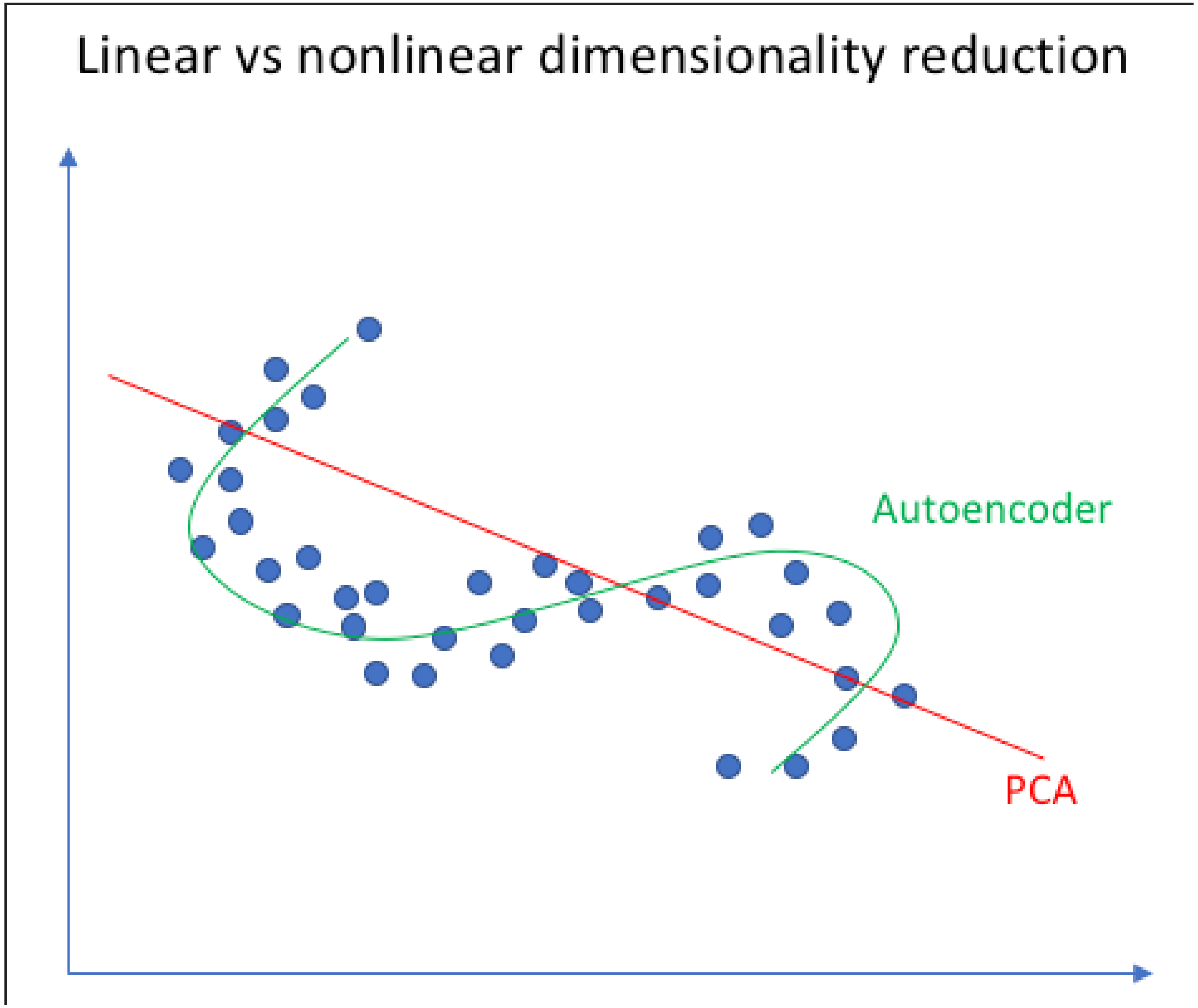


$$z = h(x) \in \mathbb{R}^{d_z}$$

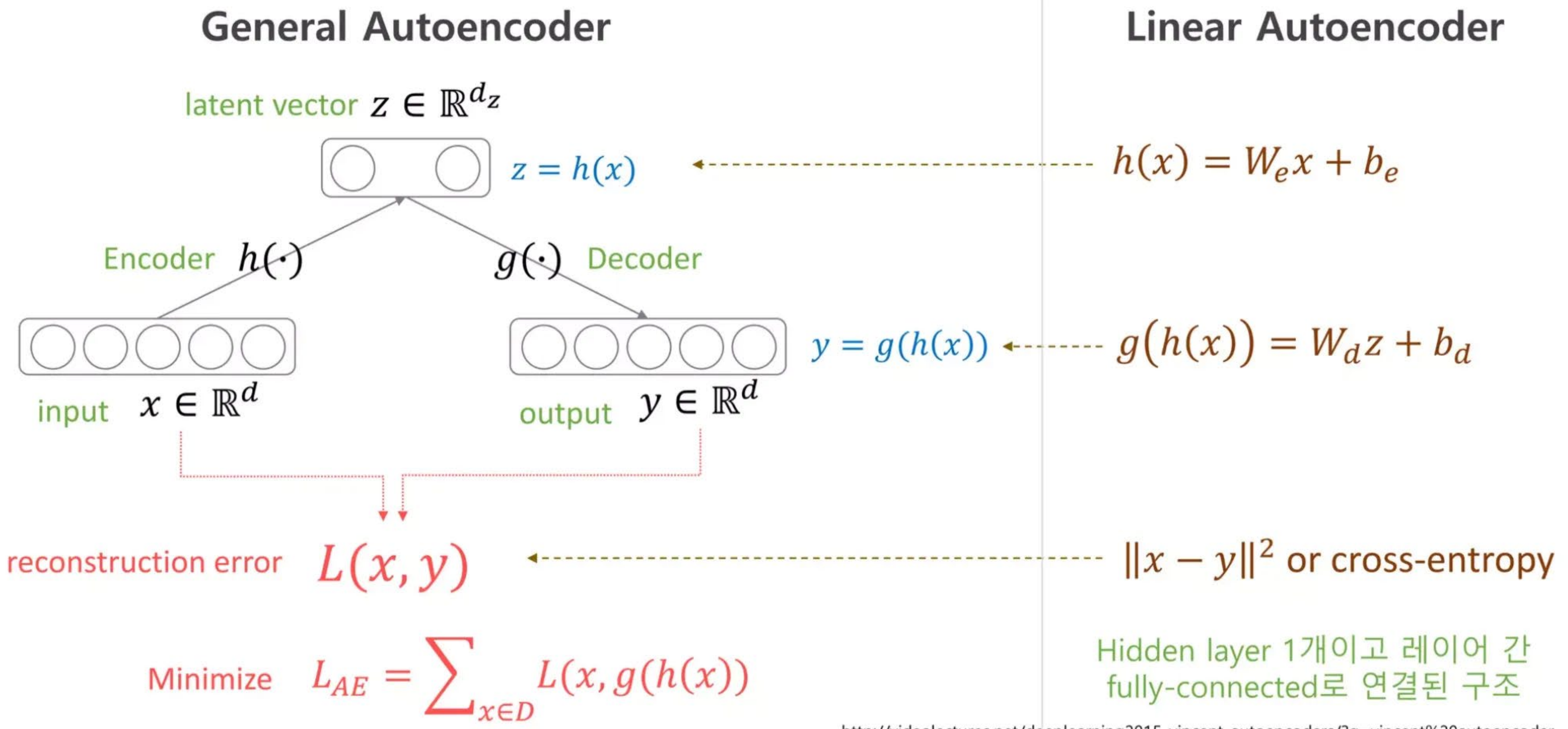
$$y = g(z) = g(h(x))$$

$$L_{AE} = \sum_{x \in D} L(x, y)$$

04. Linear Auto Encoder

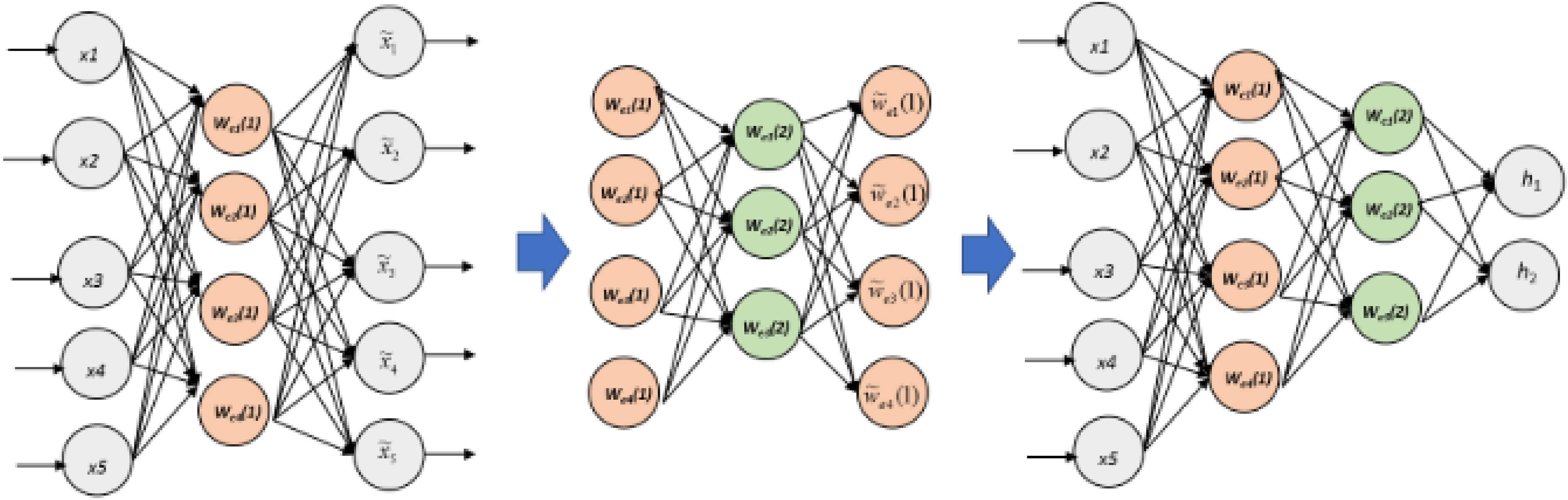


04. Linear Auto Encoder



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

05. SAE(Stacked Auto Encoder)

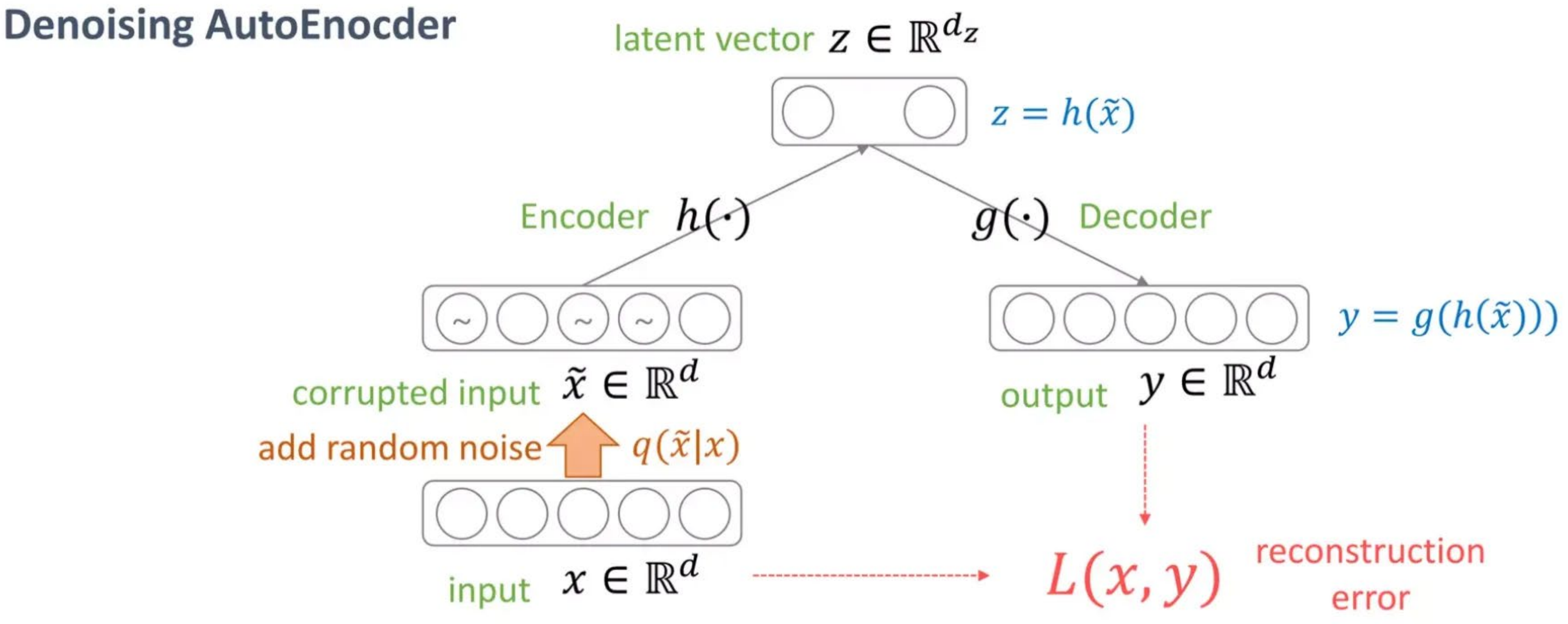


Step 1: Pre-training layer 1

Step 2: Pre-training layer 2

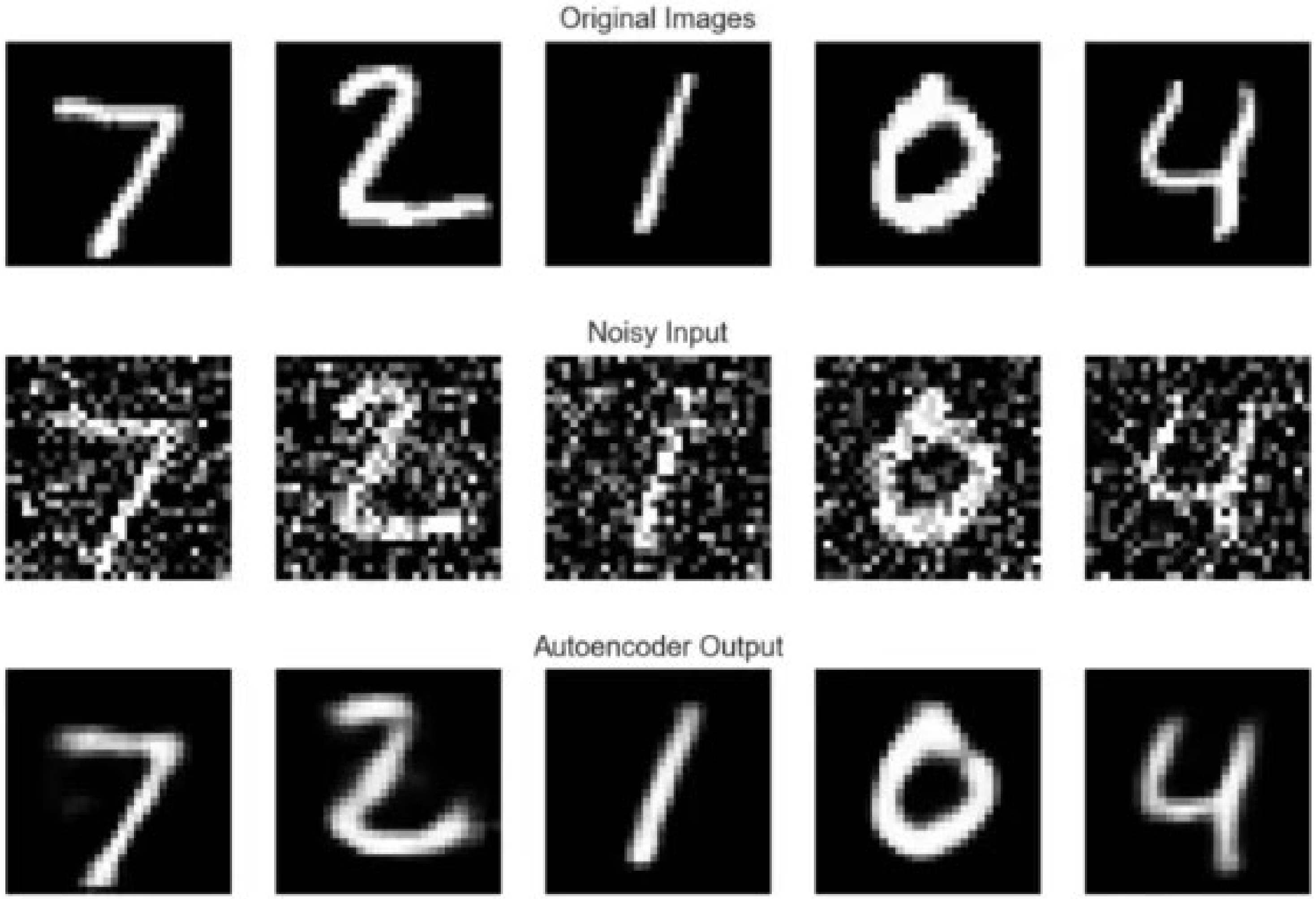
Step 3: Model fine tuning

06. DAE (Denoising Auto Encoder)

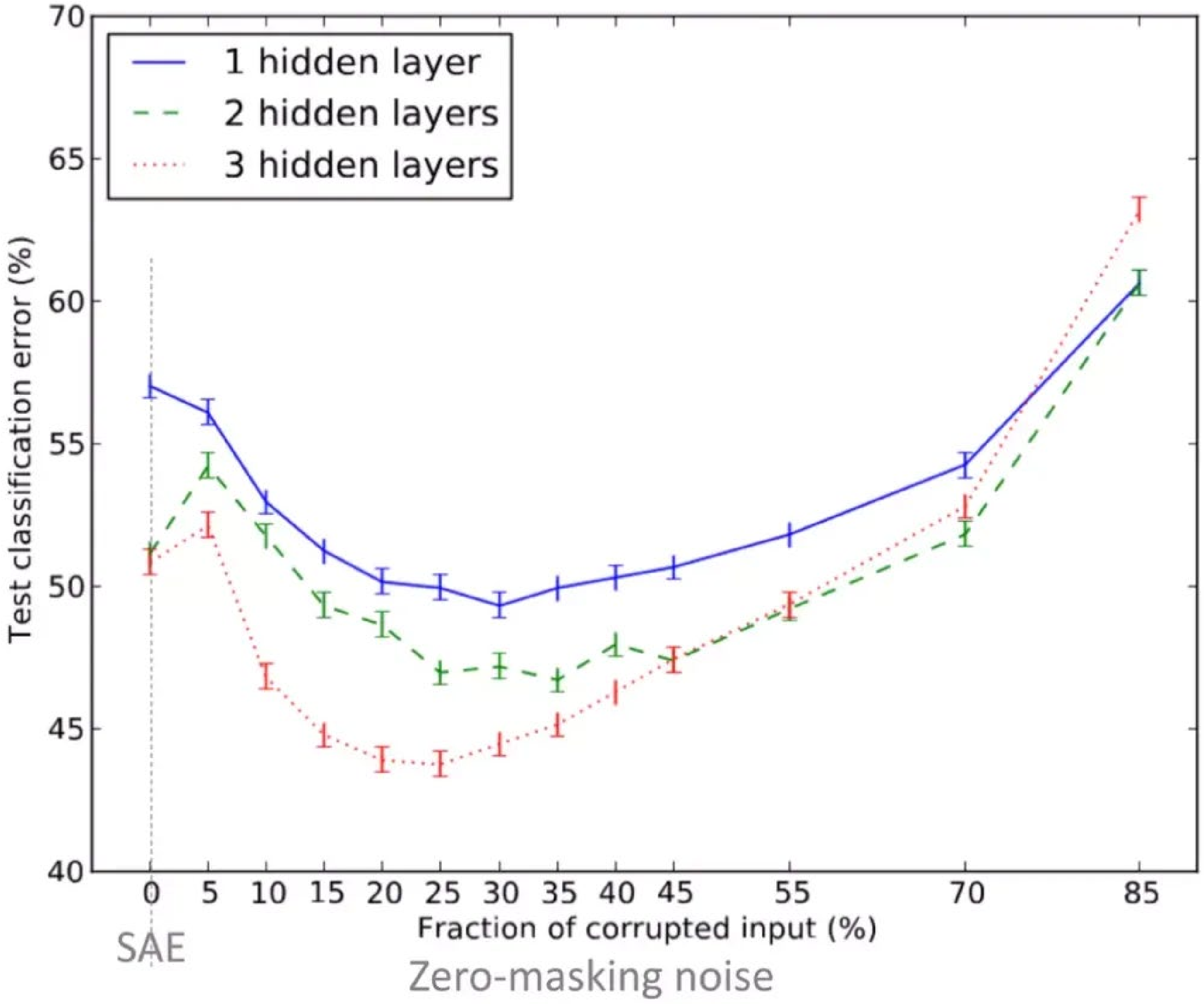


Minimize $L_{DAE} = \sum_{x \in D} E_{q(\tilde{x}|x)} \left[L(x, g(h(\tilde{x}))) \right]$

06. DAE (Denoising Auto Encoder)

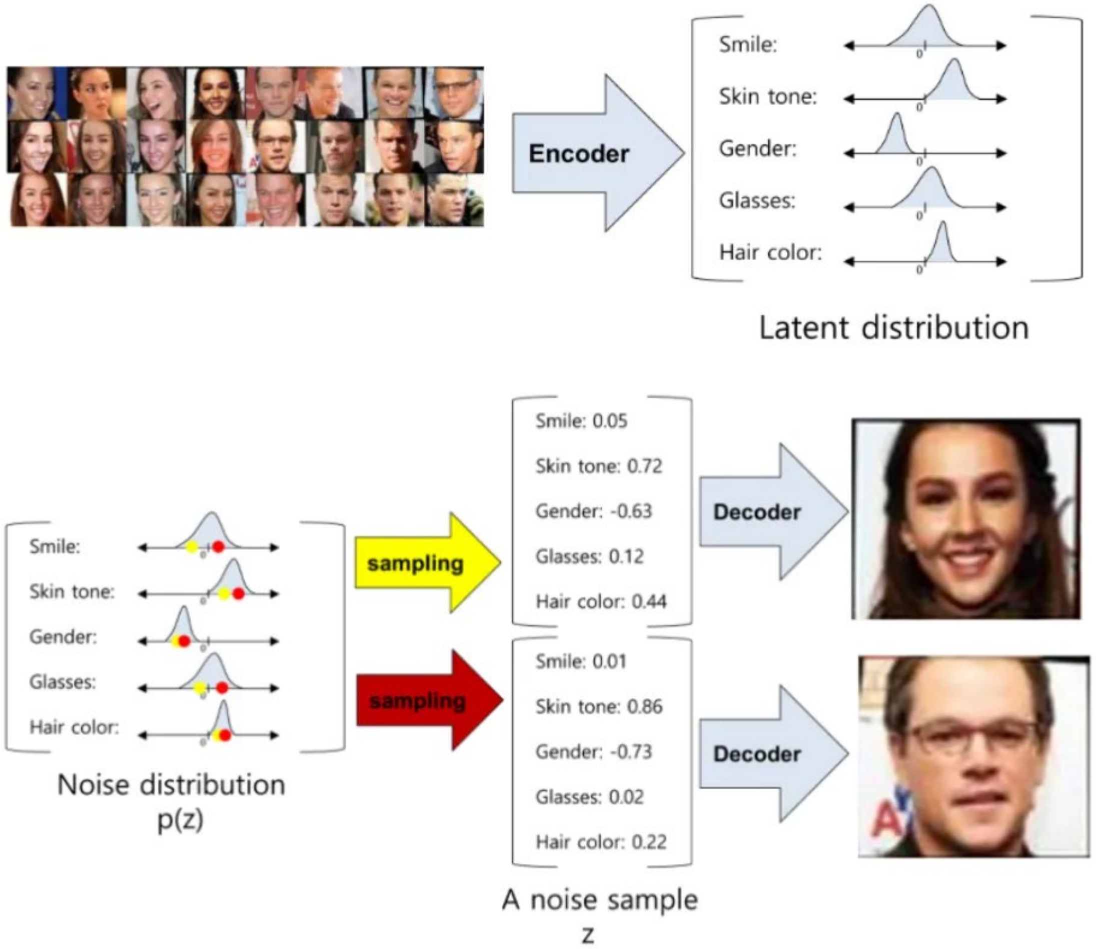


06. DAE (Denoising Auto Encoder)



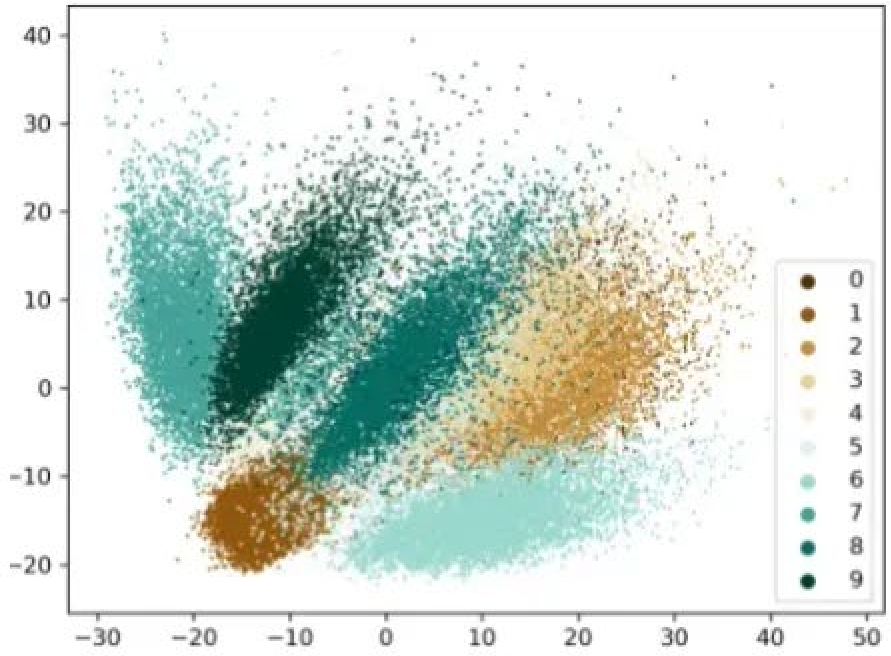
07. VAE (Variantional Auto Encoder)

input image x를 잘 설명하는 특징을 추출하여 latent vector z에 담고
이 latent vector z를 통해 x와 유사하지만 완전히 새로운 데이터를 생성해내는 것을 목표로 함
⇒ 생성 모델(generative model)

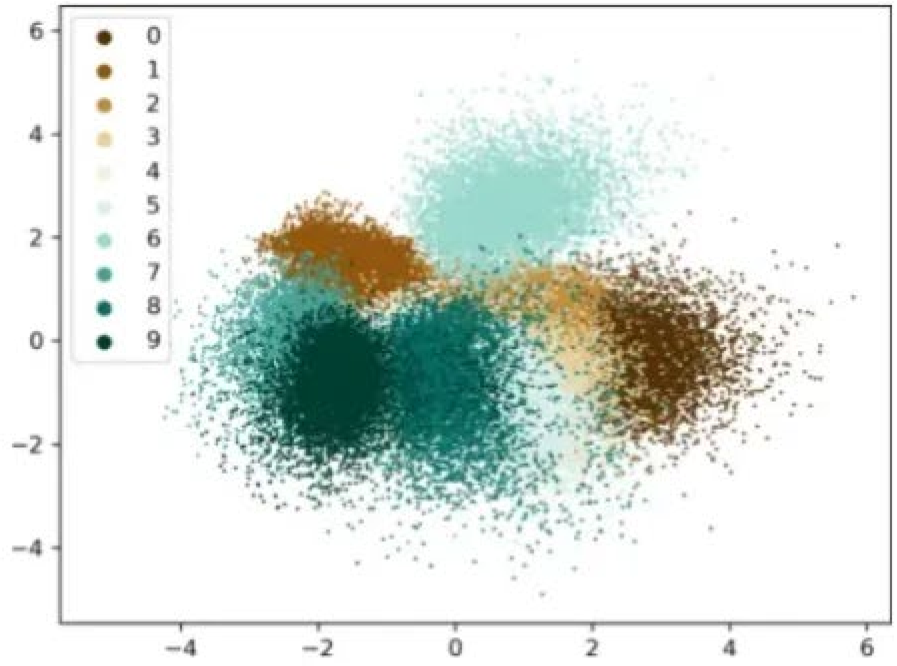


07. VAE (Variantional Auto Encoder)

input image x를 잘 설명하는 특징을 추출하여 latent vector z에 담고
이 latent vector z를 통해 x와 유사하지만 완전히 새로운 데이터를 생성해내는 것을 목표로 함
⇒ 생성 모델(generative model)

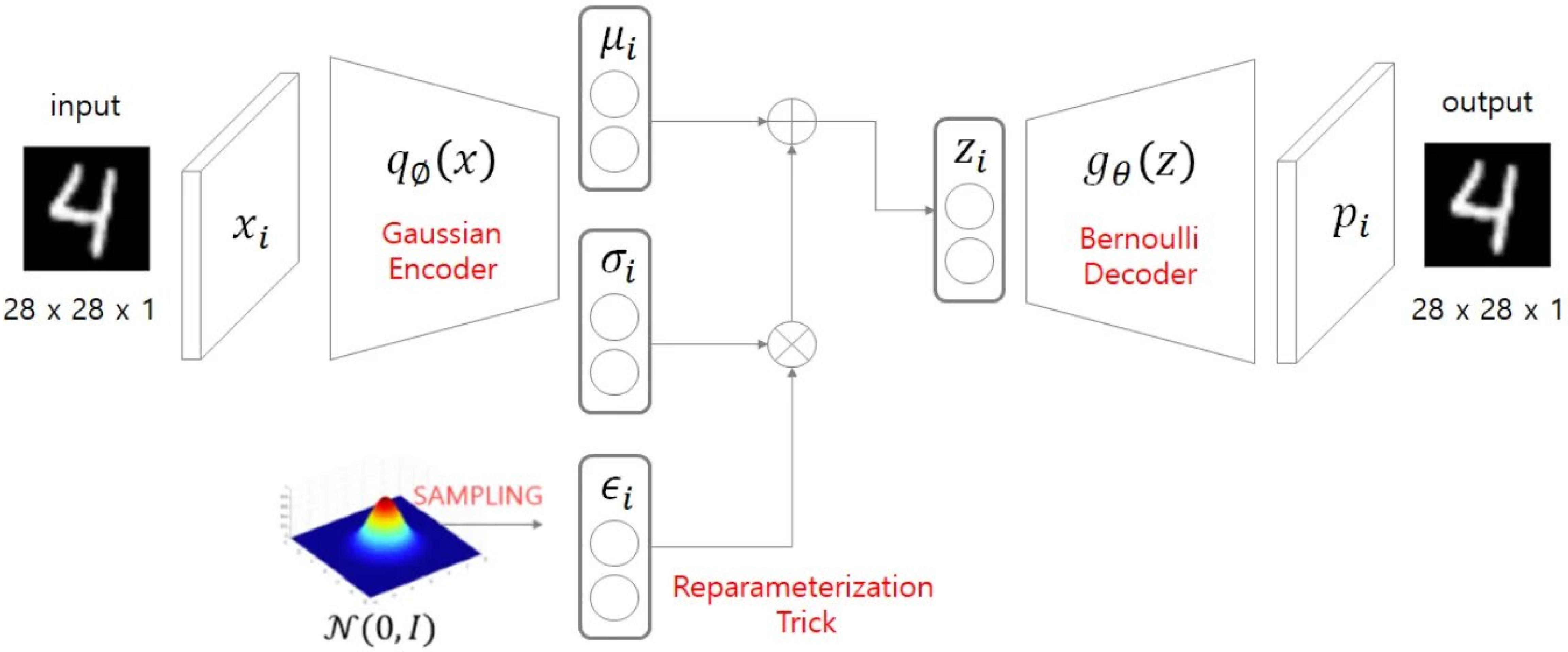


(a) *Latent Distribution by Label for AE*

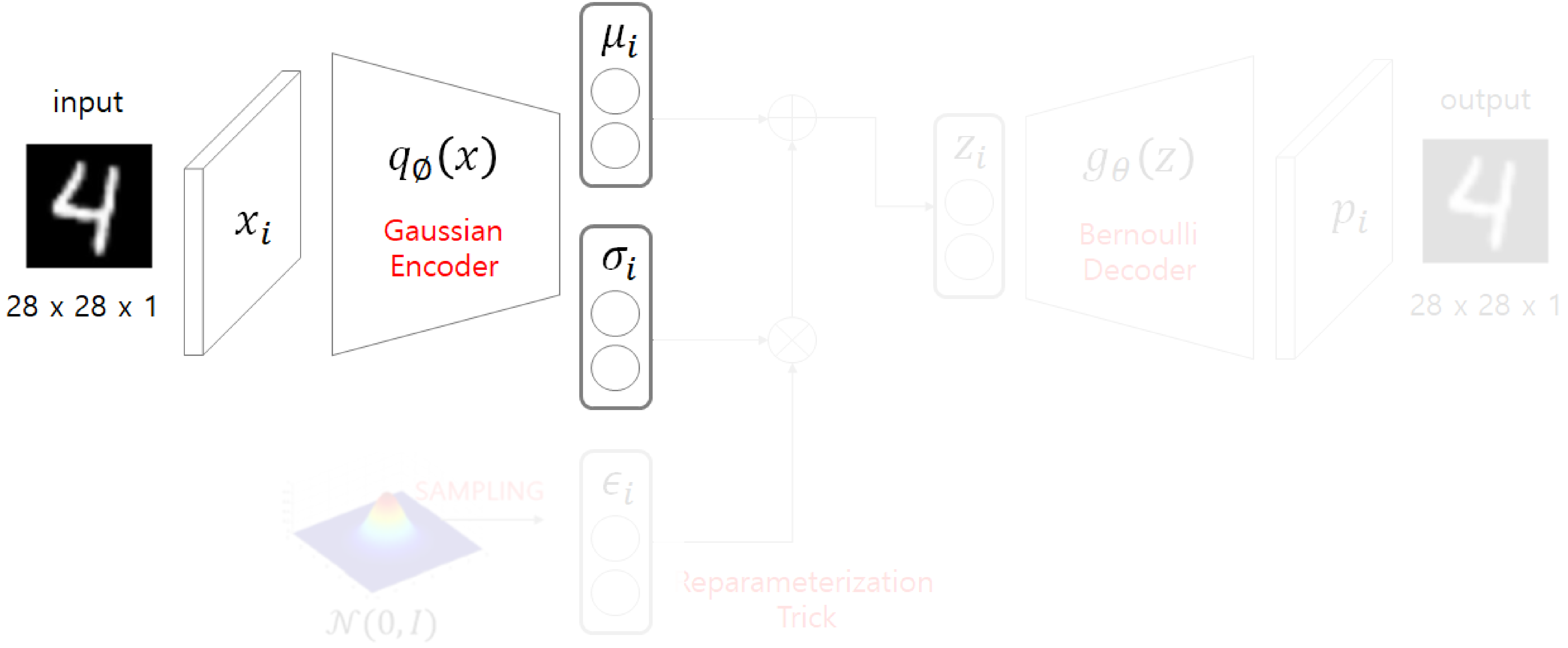


(b) *Latent Distribution by Label for VAE*

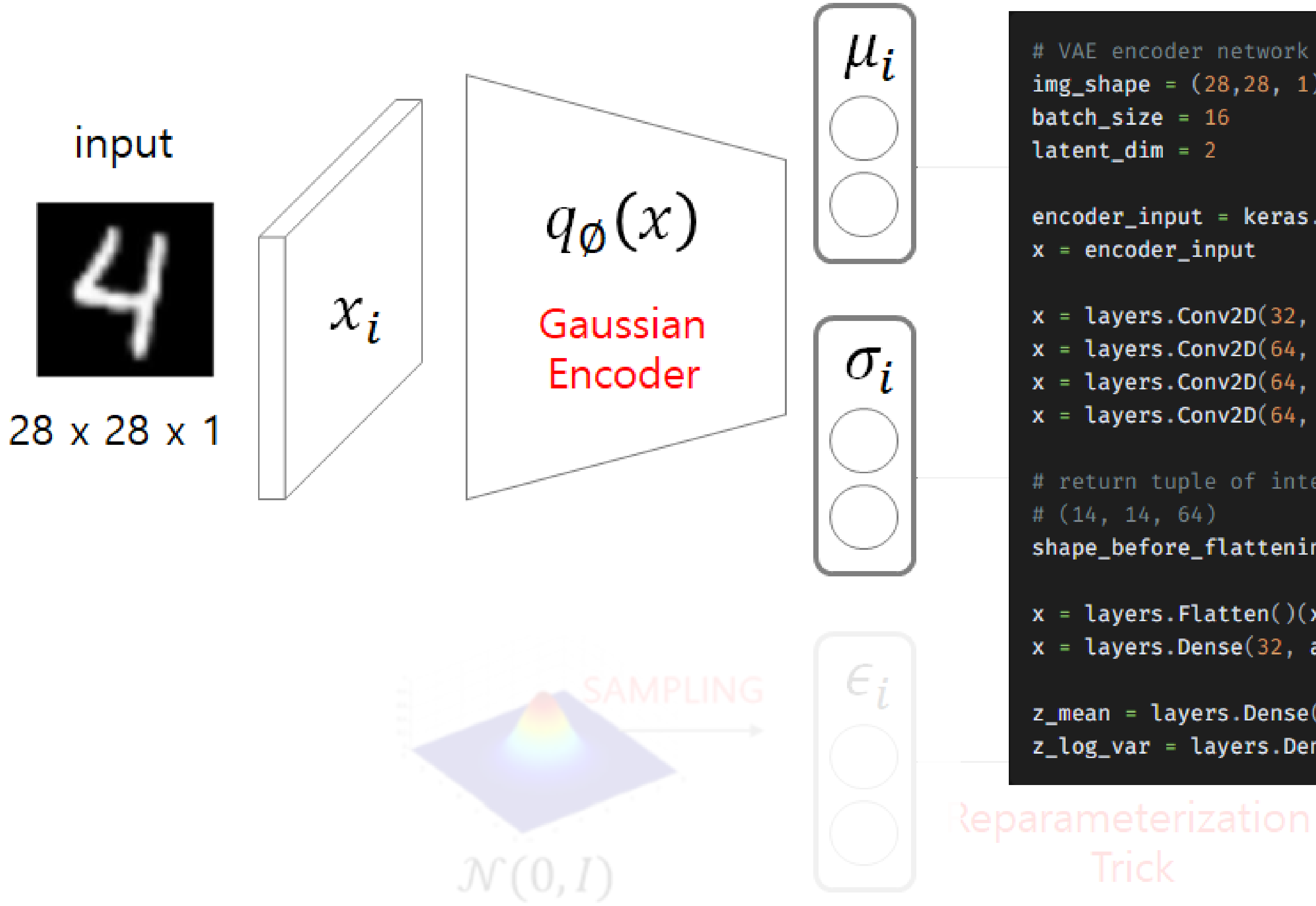
07. VAE (Variantional Auto Encoder)



07. VAE (Variantional Auto Encoder) - Encoder

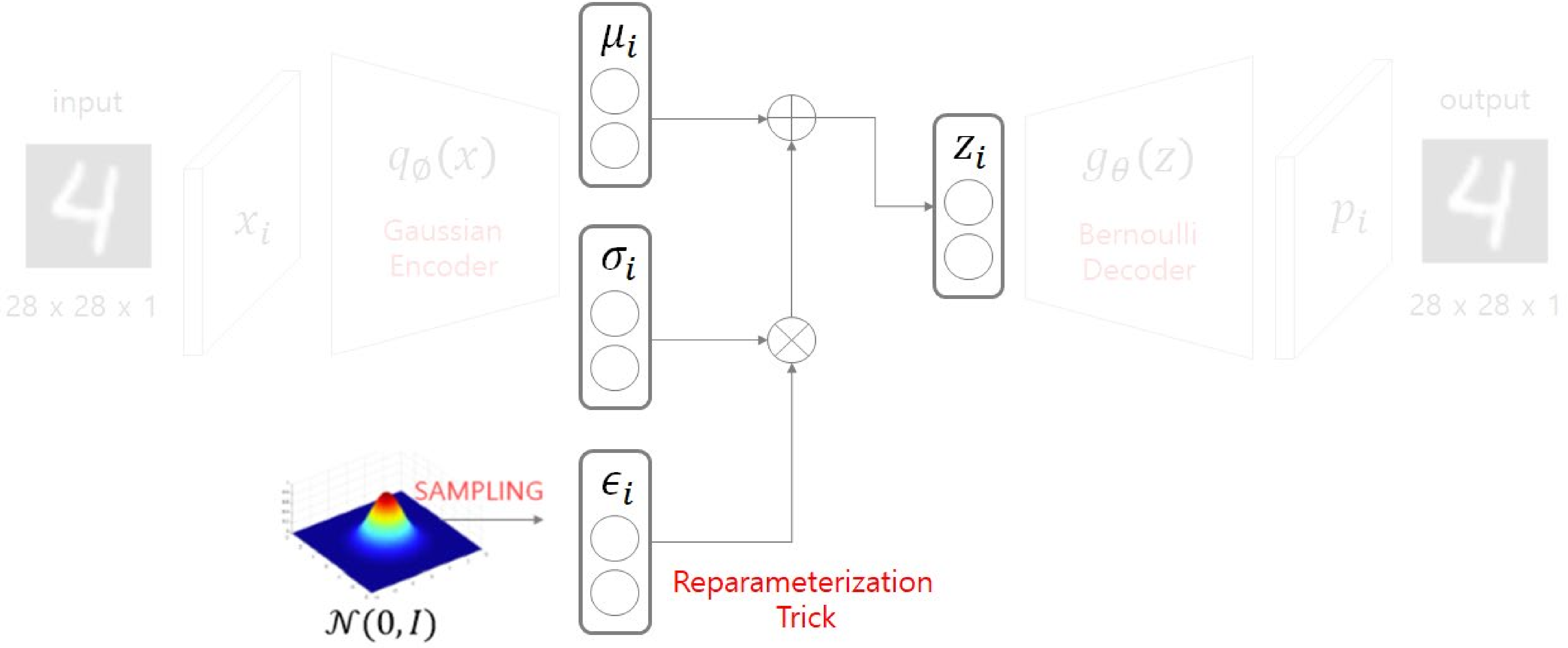


07. VAE (Variantional Auto Encoder) - Encoder

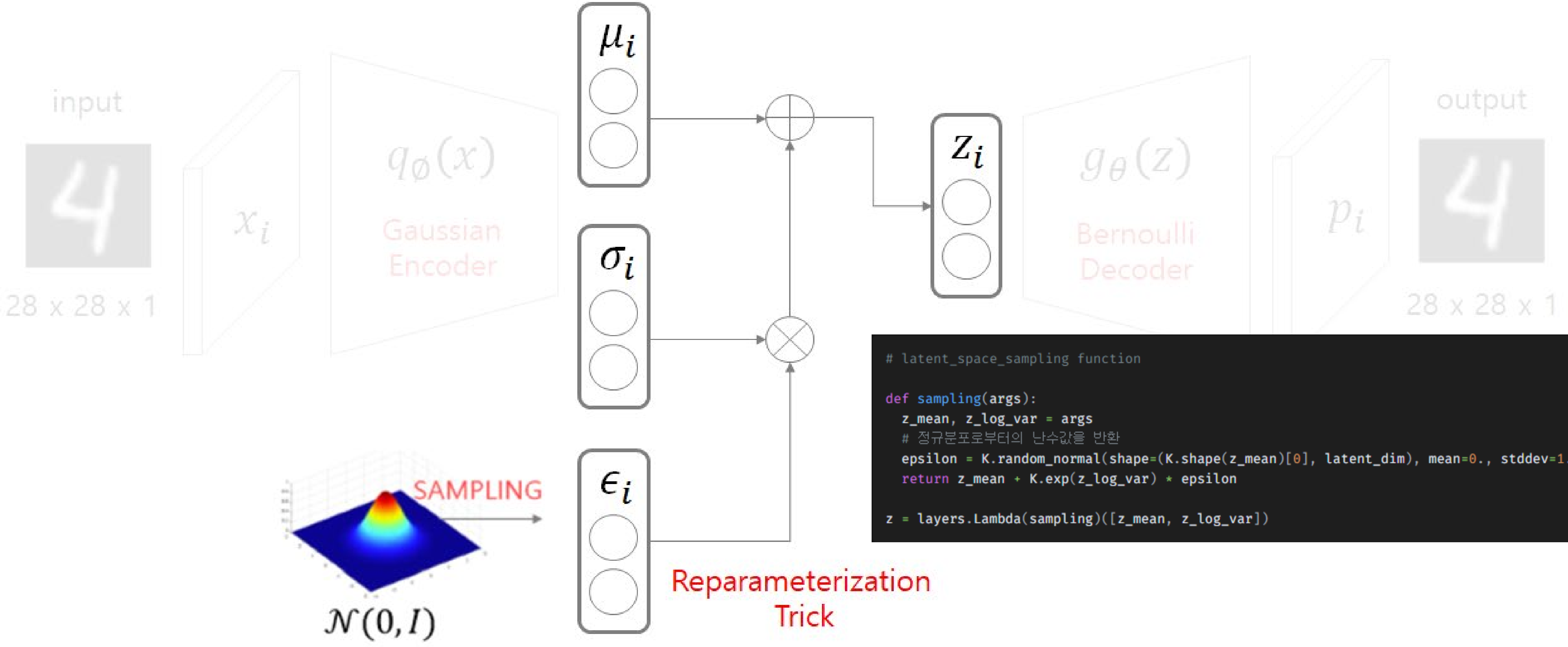



```
# VAE encoder network
img_shape = (28,28, 1)
batch_size = 16
latent_dim = 2

encoder_input = keras.Input(shape = img_shape, name='Encoder_Input')
x = encoder_input

x = layers.Conv2D(32, 3, padding = 'same', activation='relu')(x)
x = layers.Conv2D(64, 3, padding = 'same', activation='relu', strides=(2, 2))(x)
x = layers.Conv2D(64, 3, padding = 'same', activation='relu')(x)
x = layers.Conv2D(64, 3, padding = 'same', activation='relu')(x)

# return tuple of integers of shape of x
# (14, 14, 64)
shape_before_flattening = K.int_shape(x)[1:]

x = layers.Flatten()(x) # (None, 12544)
x = layers.Dense(32, activation='relu')(x) # (None, 32)

z_mean = layers.Dense(latent_dim)(x)
z_log_var = layers.Dense(latent_dim)(x)
```

07. VAE (Variantional Auto Encoder) –Reparameterization Trick(Sampling)

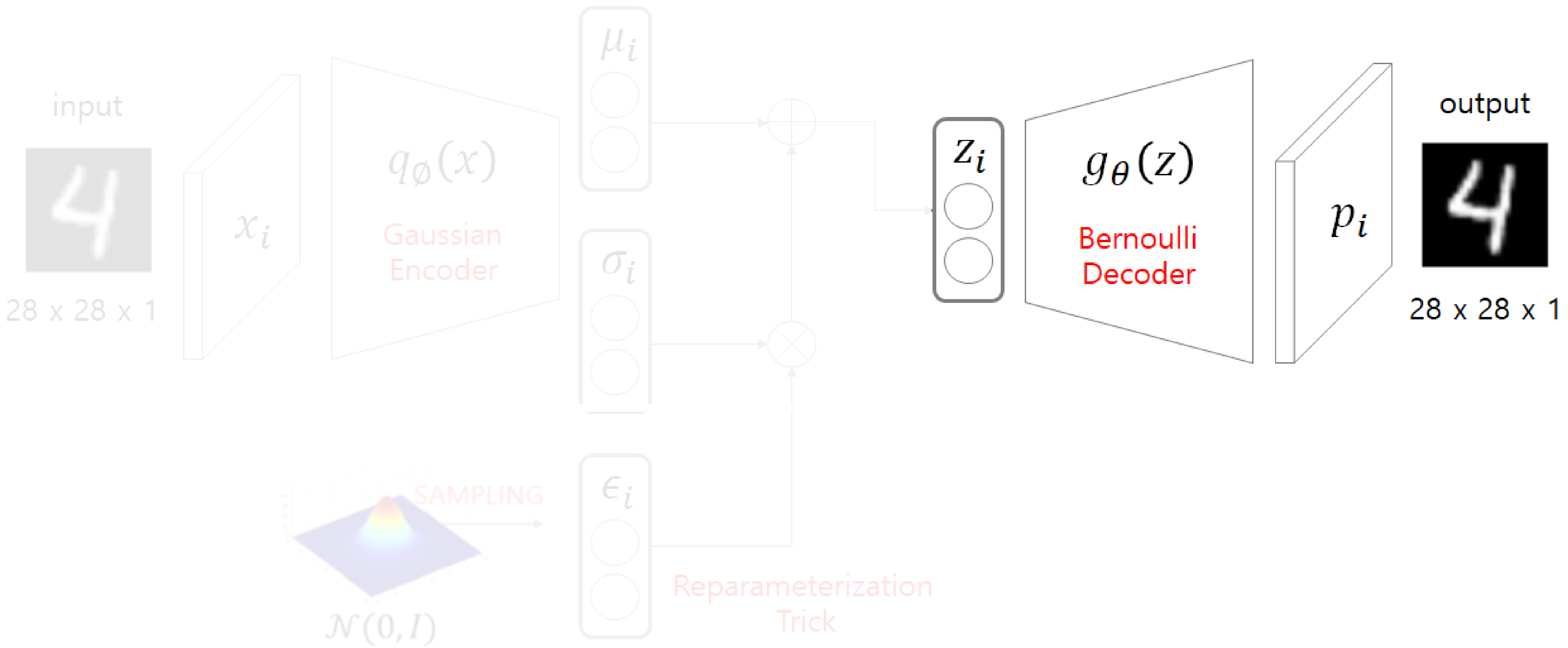


07. VAE (Variantional Auto Encoder) –Reparameterization Trick(Sampling)

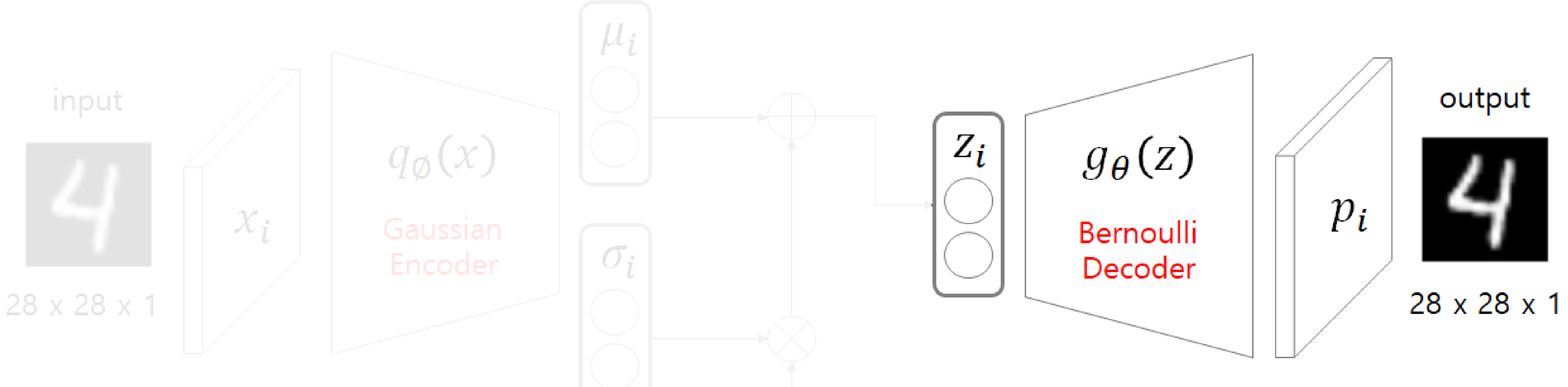



```python
# latent_space_sampling function

def sampling(args):
  z_mean, z_log_var = args
  # 정규분포로부터의 난수값을 반환
  epsilon = K.random_normal(shape=(K.shape(z_mean)[0], latent_dim), mean=0., stddev=1.)
  return z_mean + K.exp(z_log_var) * epsilon

z = layers.Lambda(sampling)([z_mean, z_log_var])
```

07. VAE (Variantional Auto Encoder) – Decoder



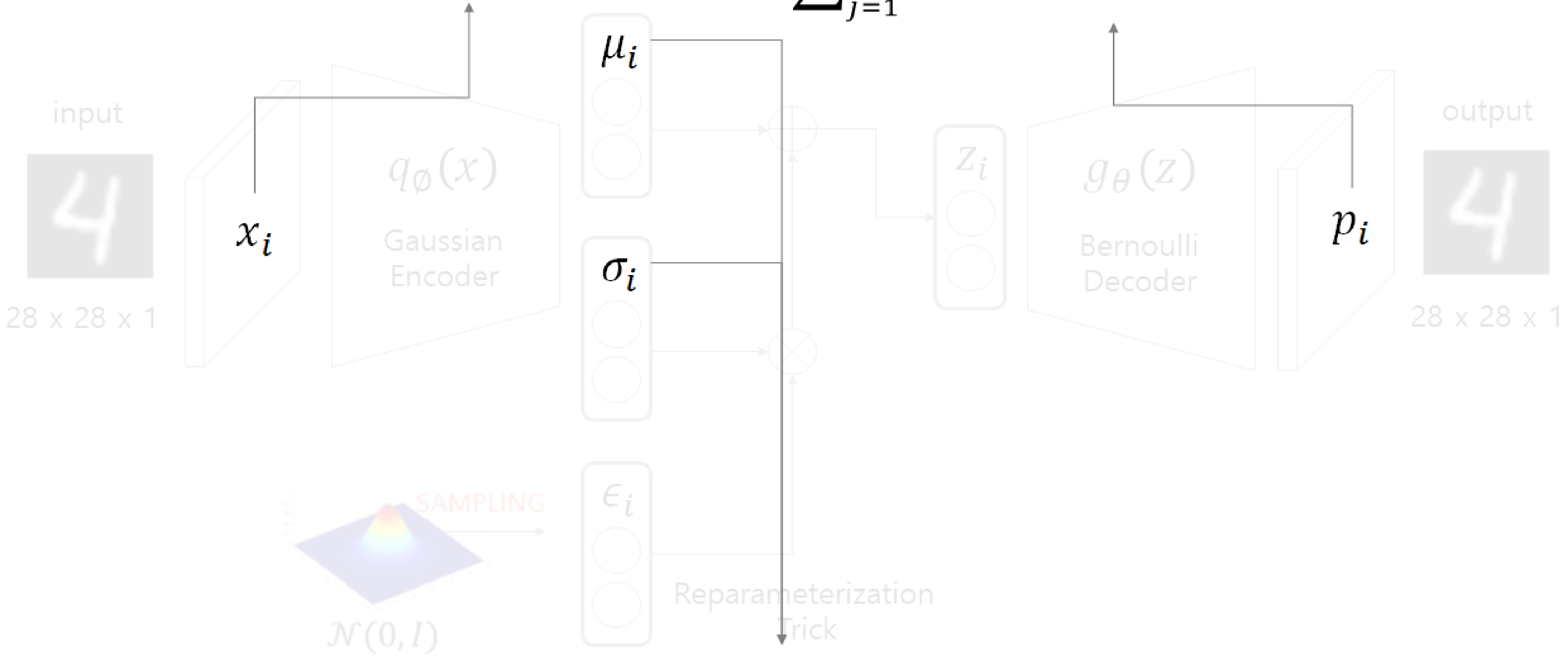
07. VAE (Variantional Auto Encoder) – Decoder




```
decoder_input = layers.Input(K.int_shape(z)[1:]) # 잠재벡터의 차원(latent_dim) 과 같다
x = layers.Dense(np.prod(shape_before_flattening), activation = 'relu')(decoder_input) # r
x = layers.Reshape(shape_before_flattening)(x)
x = layers.Conv2DTranspose(32, 3, padding='same', activation='relu', strides=(2,2))(x)
x = layers.Conv2D(1, 3, padding='same', activation='sigmoid')(x)

decoder_output = x
decoder = Model(inputs = decoder_input, outputs = decoder_output, name="Decoder")
z_decoded = decoder(z)
```

07. VAE (Variantional Auto Encoder) – Loss

Reconstruction Error: $-\sum_{j=1}^{D} x_{i,j} \log p_{i,j} \, (1-x_{i,j}) \log(1-P_{i,j})$



Regularization : $\frac{1}{2} \Sigma_{j=1}^{J} (\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1)$