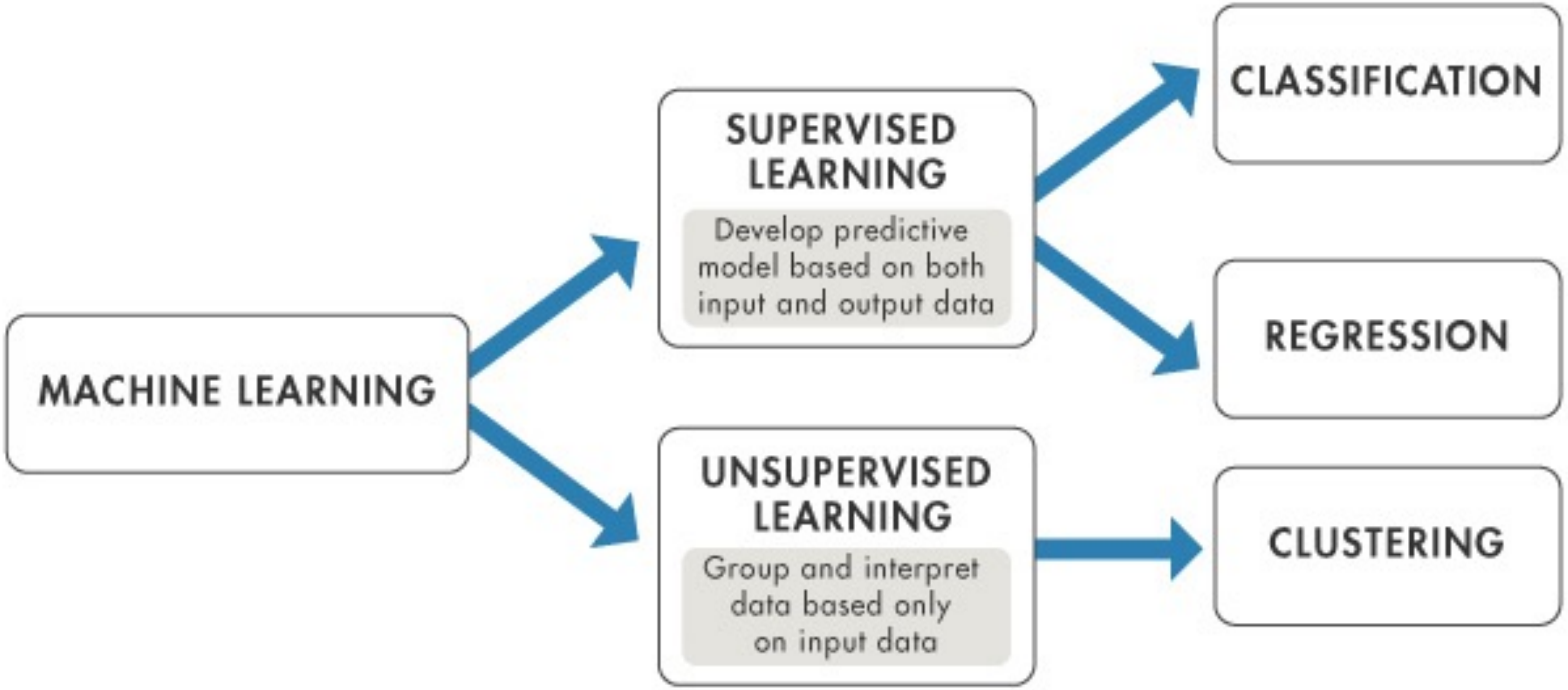
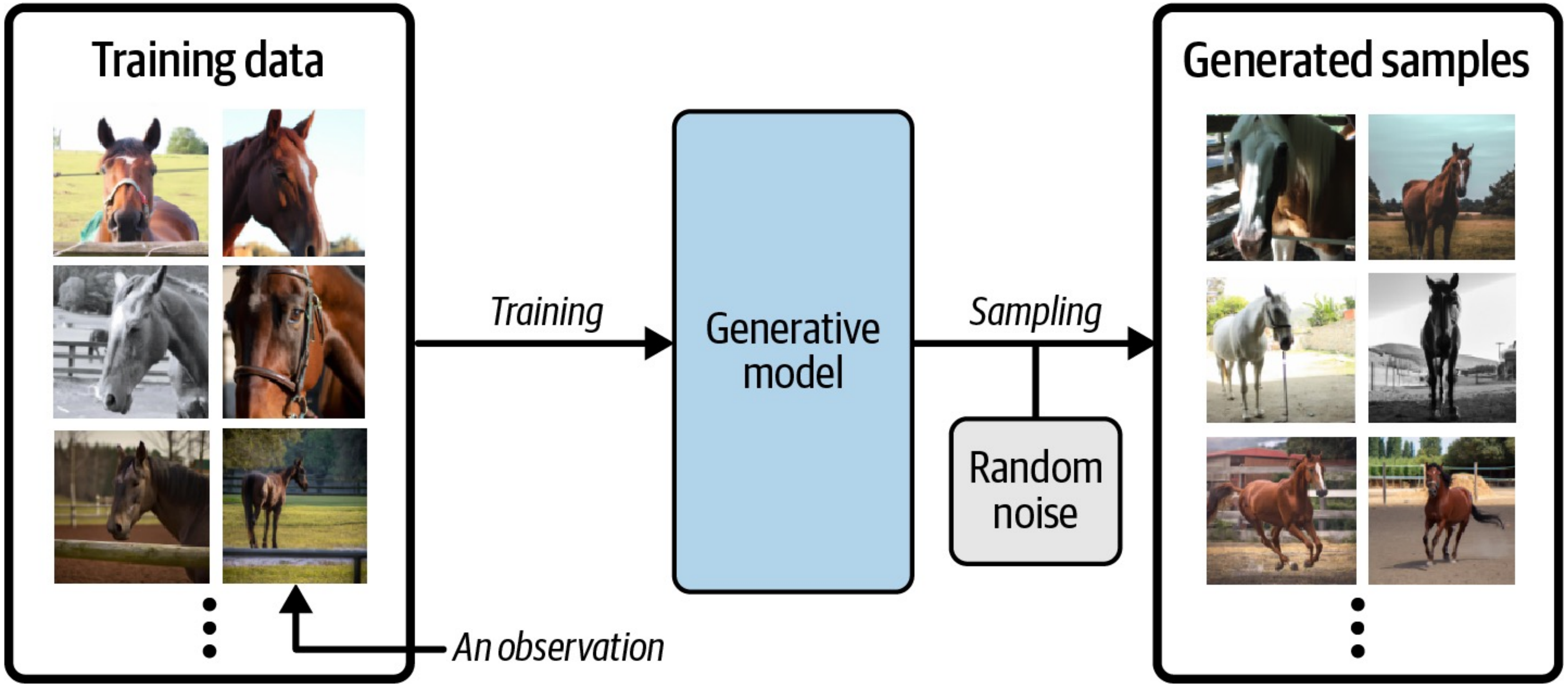
Variational Autoencoder

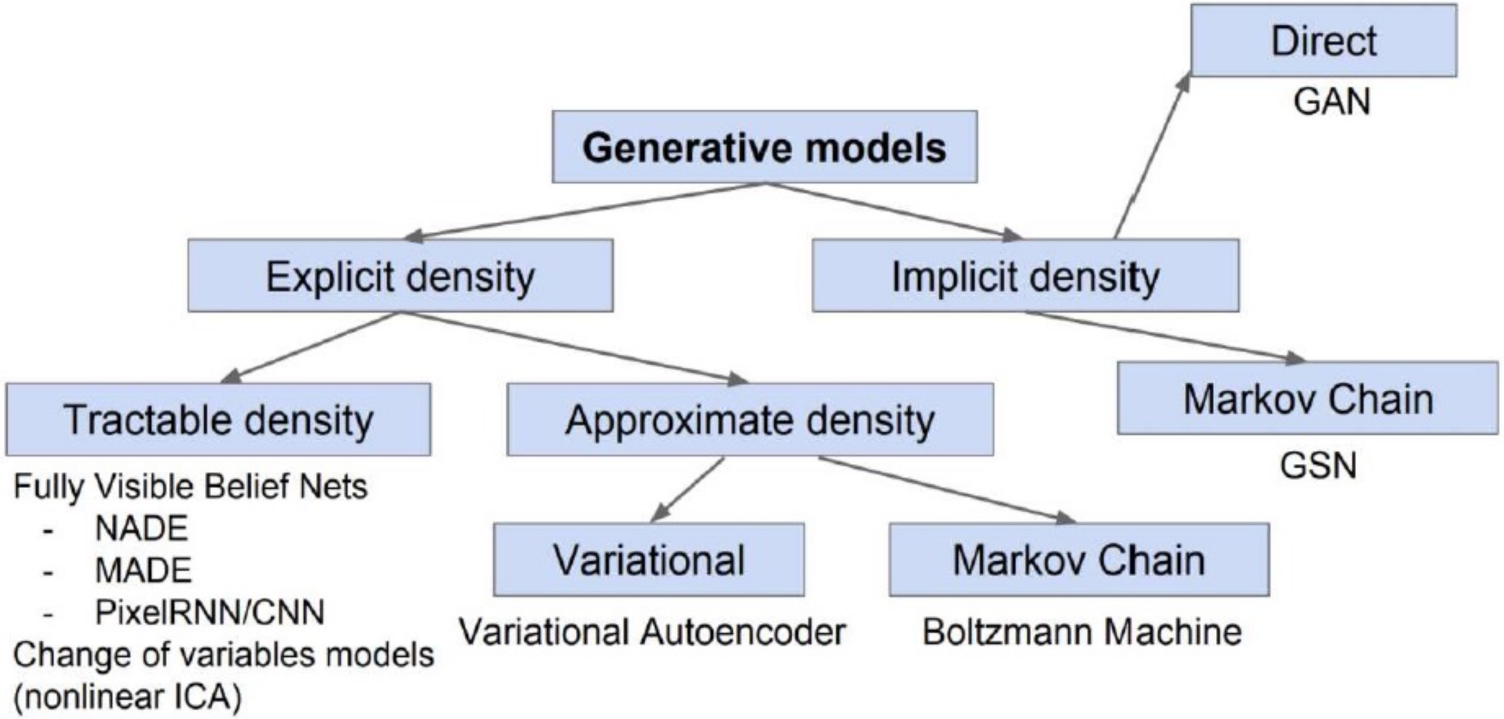
Unsupervised Learning



Generative Model

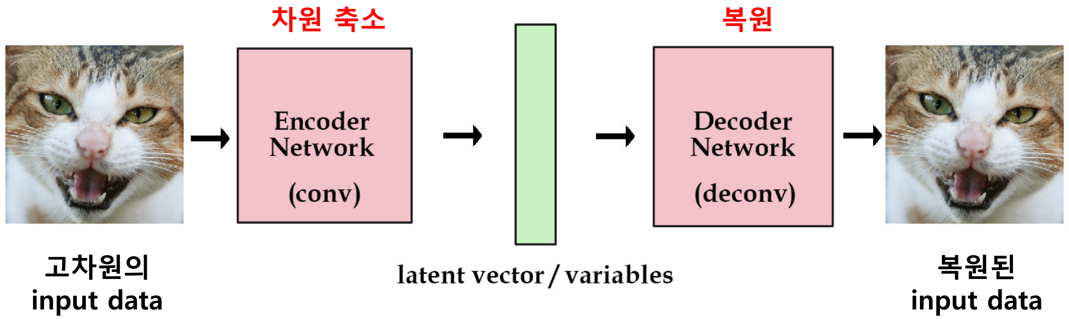


Generative Model



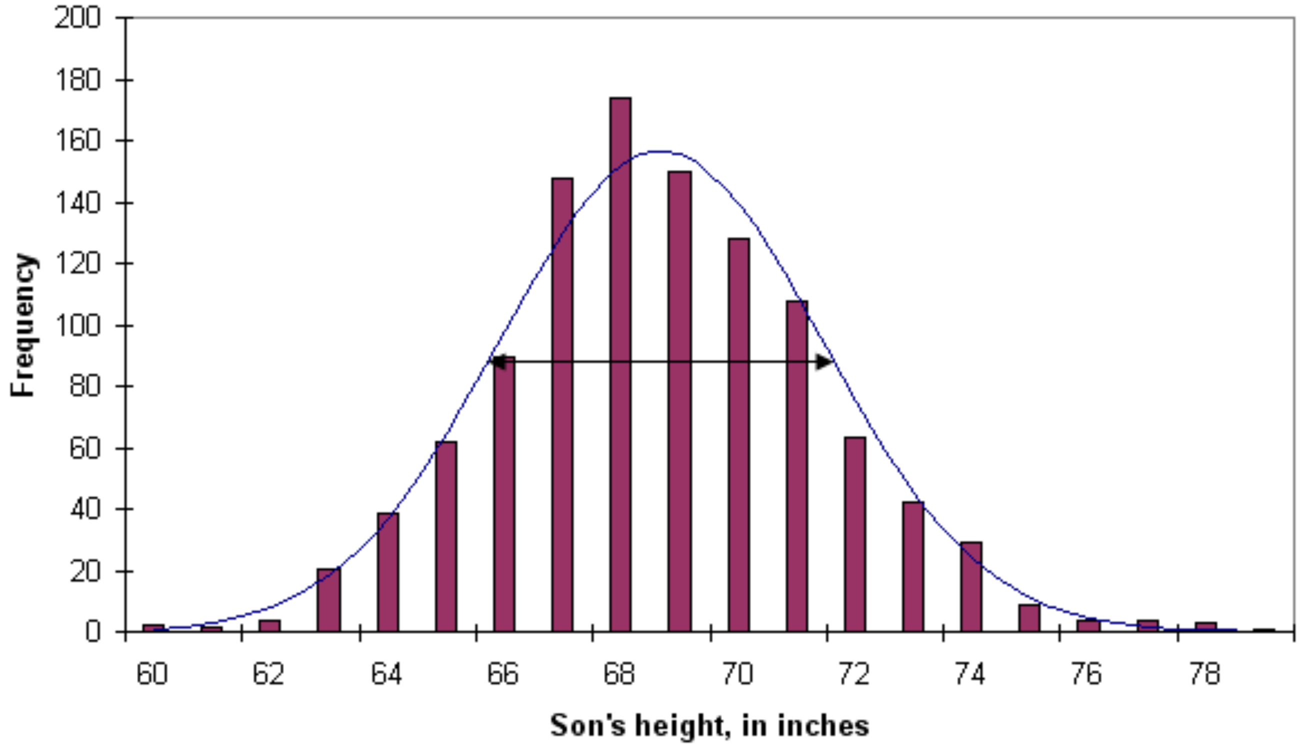
AE vs VAE

Auto-Encoder



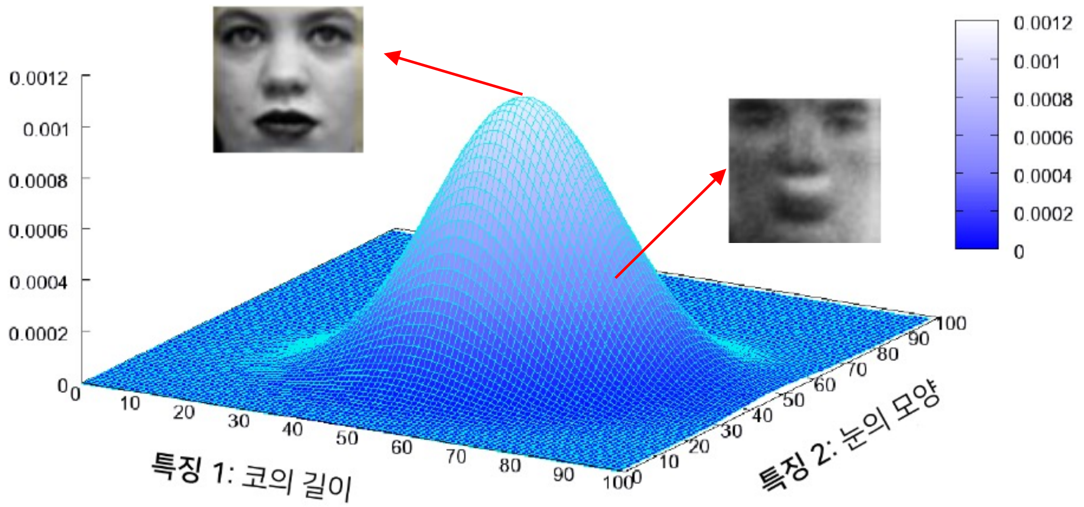
input x 자신을 언제든지 reconstruct할 수 있는 latent vector z를 만드는 것이 목적

Likelihood

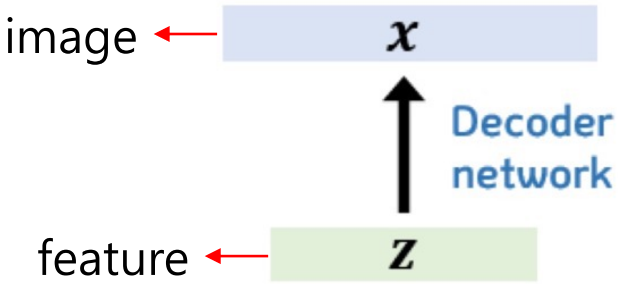


200
180
160
140
120
100
80
60
40
20
0
Frequency
60
62
64
66
68
70
72
74
76
78
Son's height, in inches

Multivariate Probability Distribution



Decoder



z latent variable의 확률분포

$$p_\theta(z)$$

z가 given일 때 x의 확률분포

$$p_\theta\left(x|z^{(i)}\right)$$

어떻게 학습?

네트워크의 출력값이 있을 때
우리가 원하는 정답 x가 나올 확률이 높길바람

= x의 likelihood를 최대화하는 확률분포 찾자

Maximize

$$p_\theta(x) = \int p_\theta(z)p_\theta(x|z)dz$$

Variational Inference

$$p_\theta(x) = \int p_\theta(z) p_\theta(x|z)\, \boxed{dz}$$

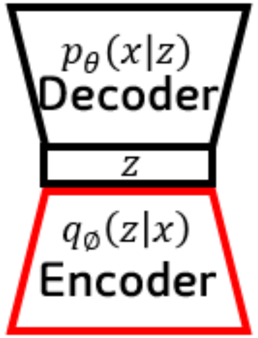
Intractable to compute p(x|z) for every z

Simple gauusian prior   Decoder neural network

Intractable

$$p_\theta(z|x) = p_\theta(x|z) p_\theta(z) / \boxed{p_\theta(x)}$$

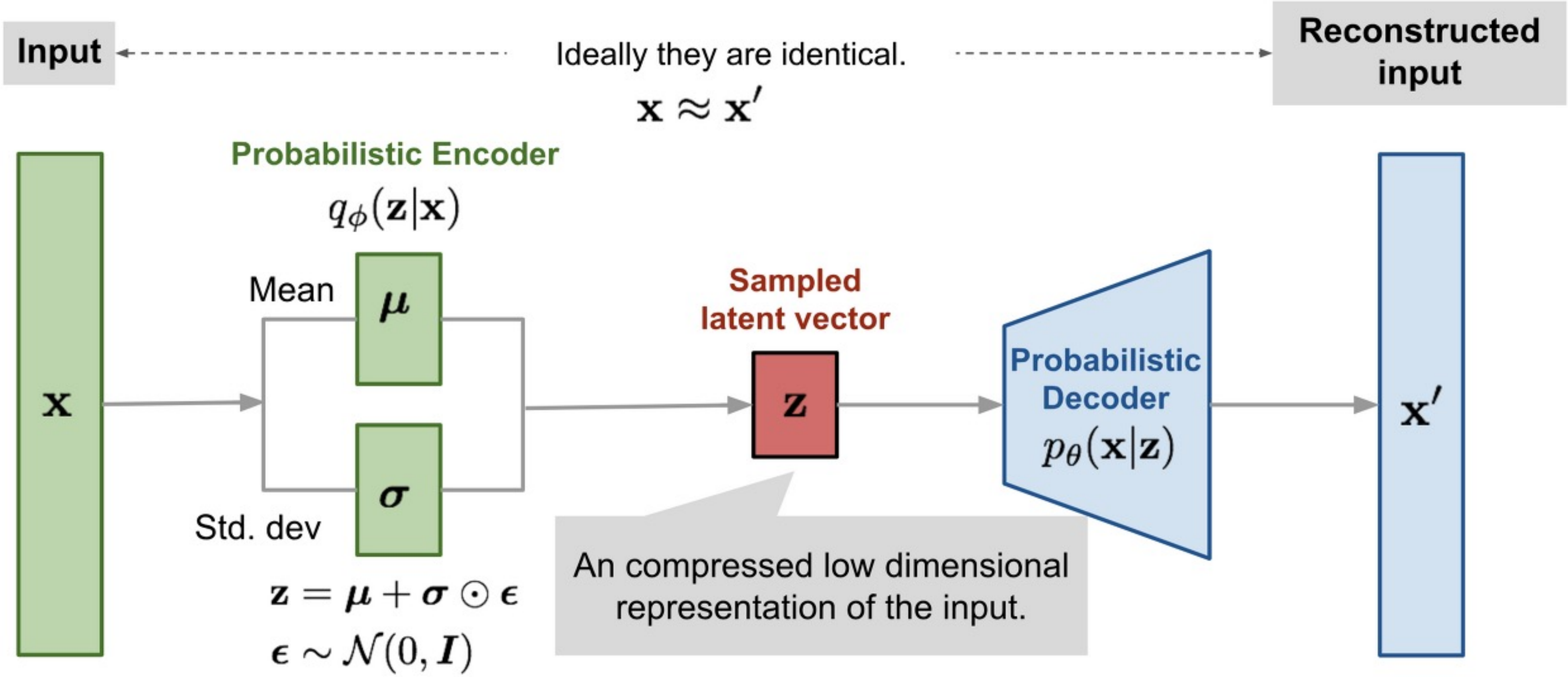
Variational Inference



Encoder 등장

-> 실제 $p_\theta(z|x)$를 가장 근사화하는 네트워크
(p는 true값, q는 추정값)

Variational Inference



ELBO(Evidence Lower bound)

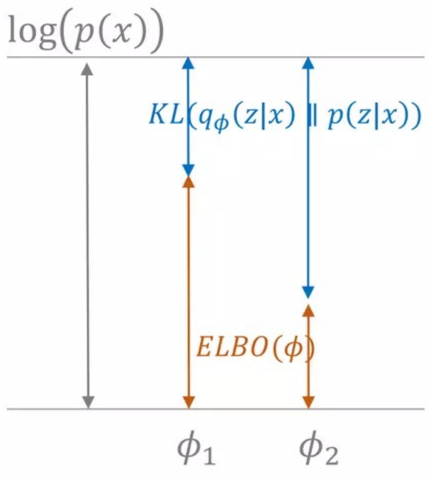
$$\log(p(x)) = \int \log(p(x))q_\phi(z|x)dz \qquad \leftarrow \int q_\phi(z|x)dz = 1$$

$$= \int \log\left(\frac{p(x,z)}{p(z|x)}\right) q_\phi(z|x)dz \quad \leftarrow p(x) = \frac{p(x,z)}{p(z|x)}$$

$$= \int \log\left(\frac{p(x,z)}{q_\phi(z|x)} \cdot \frac{q_\phi(z|x)}{p(z|x)}\right) q_\phi(z|x)dz$$

$$= \underbrace{\int \log\left(\frac{p(x,z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz}_{ELBO(\phi)} + \underbrace{\int \log\left(\frac{q_\phi(z|x)}{p(z|x)}\right) q_\phi(z|x)dz}_{KL\left(q_\phi(z|x) \parallel p(z|x)\right)}$$

두 확률분포 간의 거리≥0



ELBO(Evidence Lower bound)

$$\log(p(x)) = ELBO(\phi) + KL(q_\phi(z|x)|p(z|x))$$

$$q_{\phi^*}(z|x) = \underset{\phi}{\operatorname{argmax}}\, ELBO(\phi)$$

$$ELBO(\phi) = \int \log\left(\frac{p(x,z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz$$

$$= \int \log\left(\frac{p(x|z)p(z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz$$

$$= \int \log(p(x|z))q_\phi(z|x)dz - \int \log\left(\frac{q_\phi(z|x)}{p(z)}\right) q_\phi(z|x)dz$$

$$= \mathbb{E}_{q_\phi(z|x)}[\log(p(x|z))] - KL(q_\phi(z|x)||p(z))$$ 앞 슬라이드에서의 KL과 인자가 다른 것에 유의

ELBO(Evidence Lower bound)

$$\arg\min_{\phi,\theta} \sum_i \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p(x_i|g_\theta(z))\right)\right] + KL\left(q_\phi(z|x_i)||p(z)\right)}_{L_i(\phi,\theta,x_i)}$$

원 데이터에 대한 likelihood

Variational inference를 위한 approximation class 중 선택

다루기 쉬운 확률 분포 중 선택

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

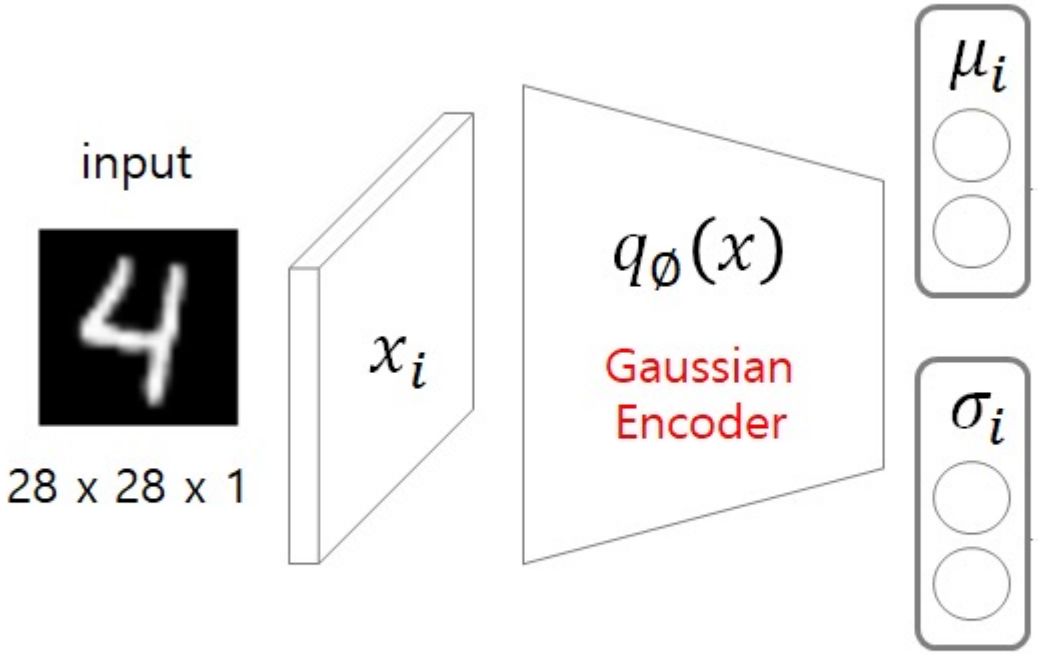
Reconstruction Error

- 현재 샘플링 용 함수에 대한 negative log likelihood
- x_i에 대한 복원 오차 (AutoEncoder 관점)

Regularization

- 현재 샘플링 용 함수에 대한 추가 조건
- 샘플링의 용의성/생성 데이터에 대한 통제성을 위한 조건을 prior에 부여 하고 이와 유사해야 한다는 조건을 부여

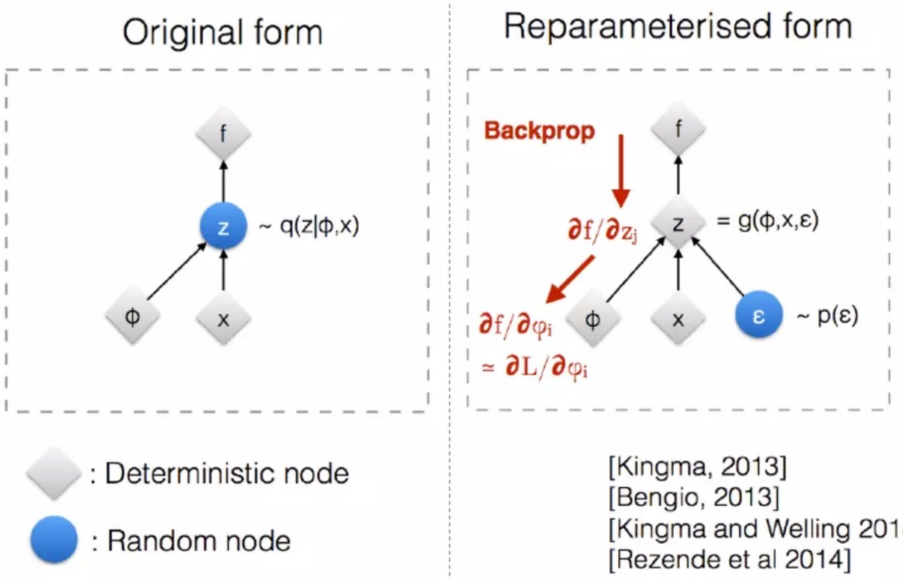
Encoder




```python
img_shape = (28,28,1)
batch_size = 16
latent_dim = 2

input_img = keras.Input(shape = img_shape)
x = layers.Conv2D(32,3,padding='same',activation='relu')(input_img)
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)

shape_before_flattening = K.int_shape(x) # return tuple of integers of shape of x

x = layers.Flatten()(x)
x = layers.Dense(32,activation='relu')(x)

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

Reparameterization Trick

Reparameterization Trick



[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

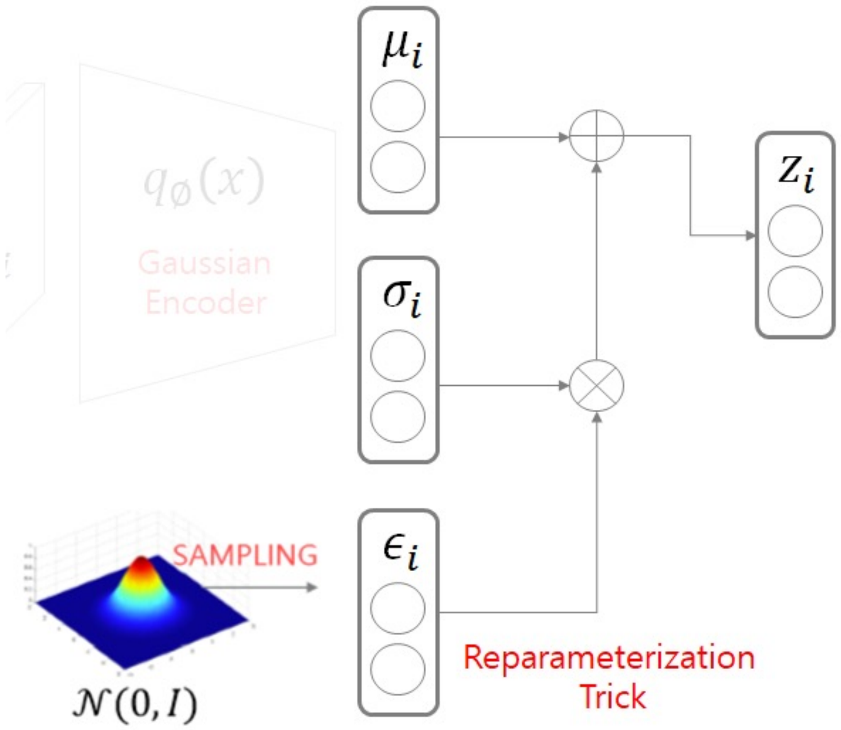
Sampling Process

$$z^{i,l} \sim N(\mu_i, \sigma_i^2 I)$$

$$z^{i,l} = \mu_i + \sigma_i^2 \odot \epsilon$$
$$\epsilon \sim N(0, I)$$

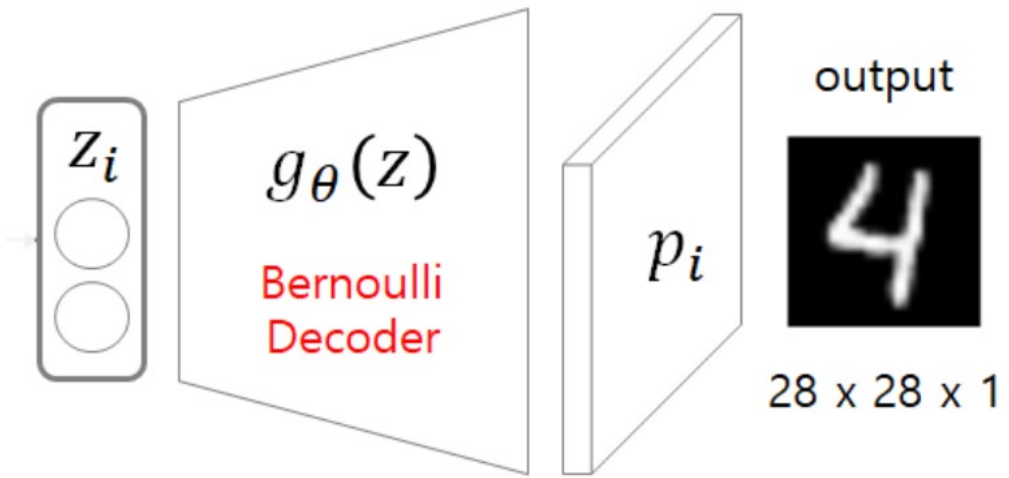
Same distribution!
But it makes backpropagation possible!!

Reparameterization Trick

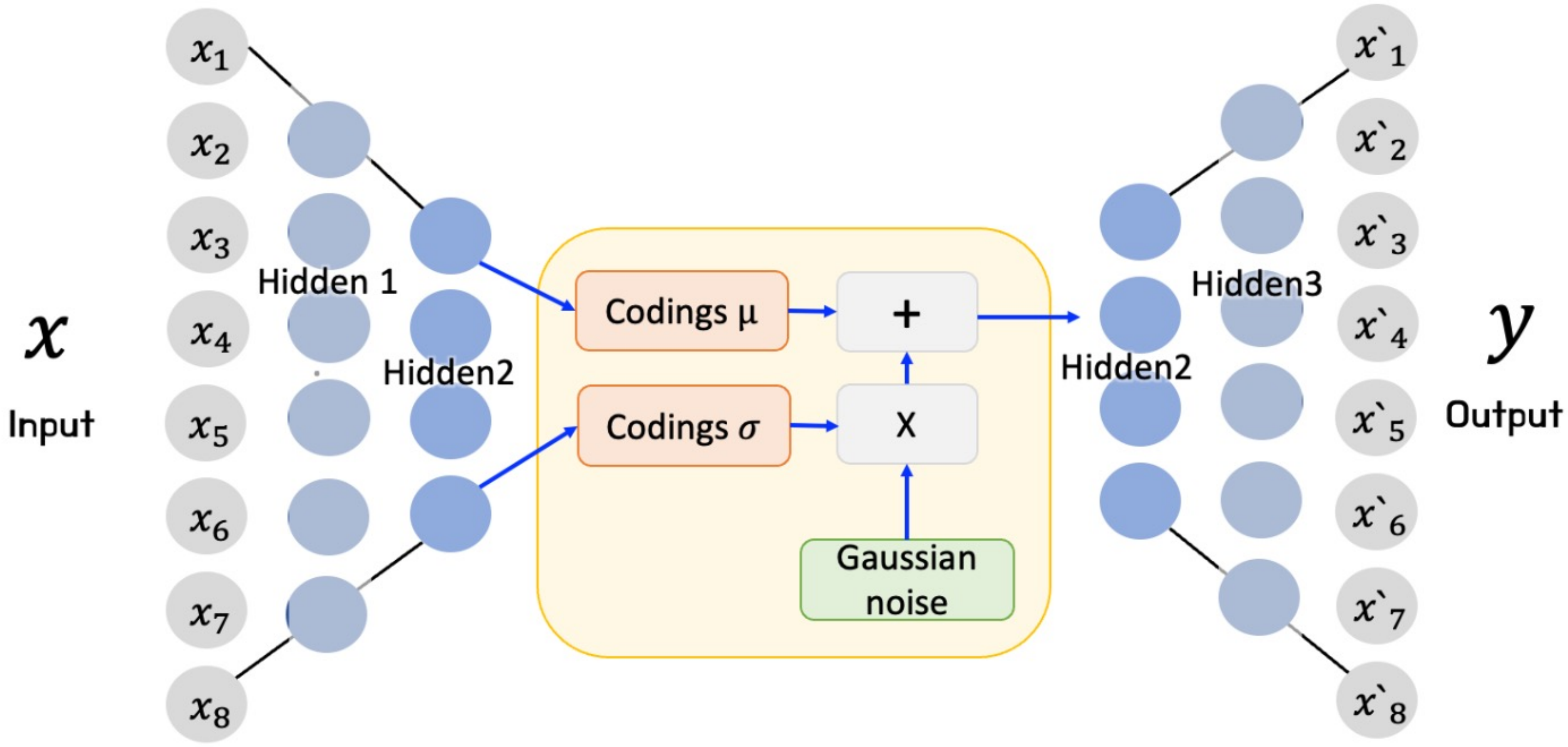



```python
def sampling(args):
    z_mean, z_log_var = args
    epsilon = K.random_normal(shape=(K.shape(z_mean)[0],latent_dim),mean=0., stddev=1.)
    return z_mean + K.exp(z_log_var) * epsilon

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

Decoder

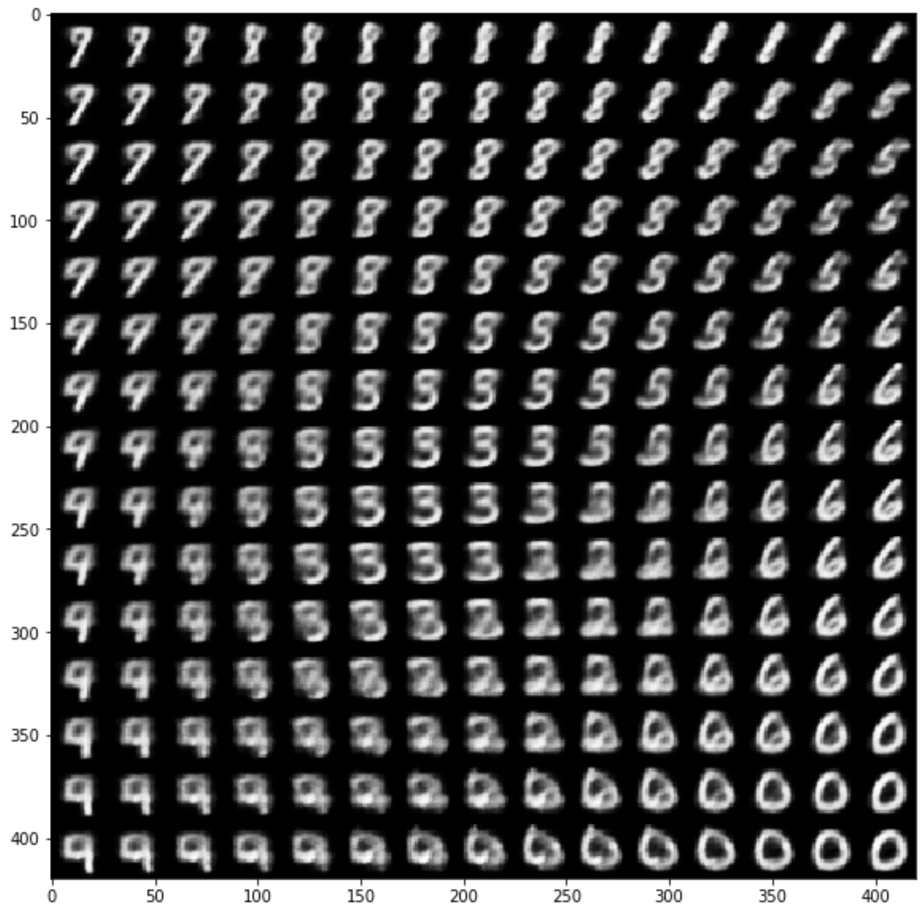



```
decoder_input = layers.Input(K.int_shape(z)[1:])
x = layers.Dense(np.prod(shape_before_flattening[1:]),activation='relu')(decoder_input)
x = layers.Reshape(shape_before_flattening[1:])(x)
x = layers.Conv2DTranspose(32,3,padding='same',activation='relu',strides=(2,2))(x)
x = layers.Conv2D(1,3,padding='same',activation='sigmoid')(x)

decoder = Model(decoder_input, x)
z_decoded = decoder(z)
```



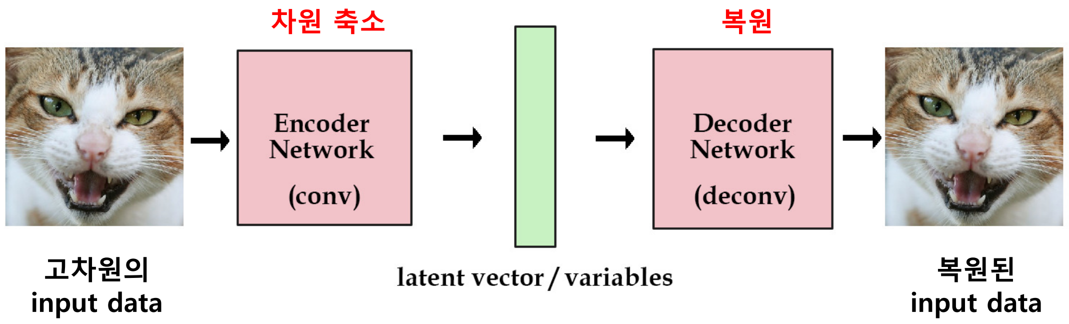
x_1
x_2
x_3
x_4
x_5
x_6
x_7
x_8
x
Input
Hidden 1
Hidden2
Codings μ
Codings σ
+
X
Gaussian noise
Hidden2
Hidden3
$x`_1$
$x`_2$
$x`_3$
$x`_4$
$x`_5$
$x`_6$
$x`_7$
$x`_8$
y
Output

Result



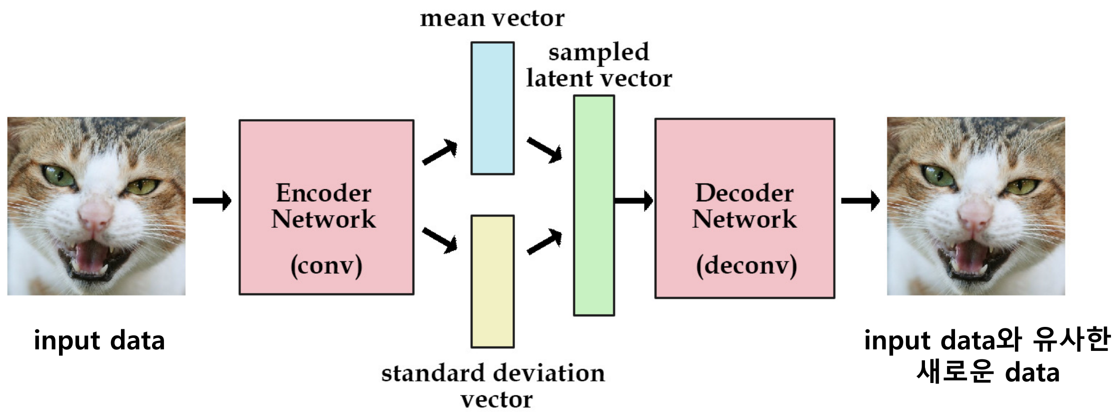
AE vs VAE

Auto-Encoder



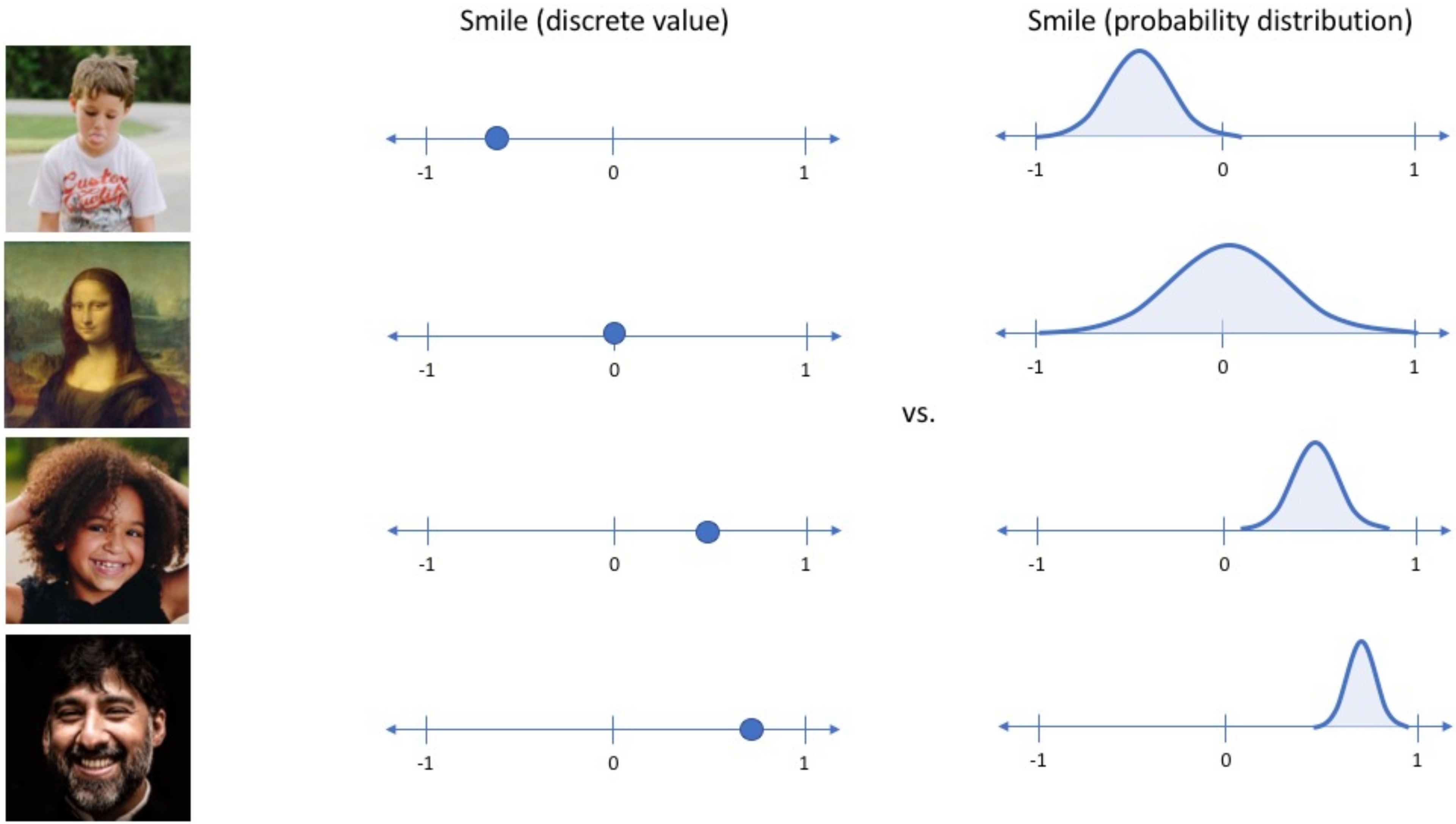
input x 자신을 언제든지 reconstruct할 수 있는 latent vector z를 만드는 것이 목적

Variational Auto-Encoder



input x와 유사한 data를 만들 수 있는 latent vector z의 확률 분포 함수를 찾는 것이 목적

AE vs VAE


Smile (discrete value)
Smile (probability distribution)
-1
0
1
vs.