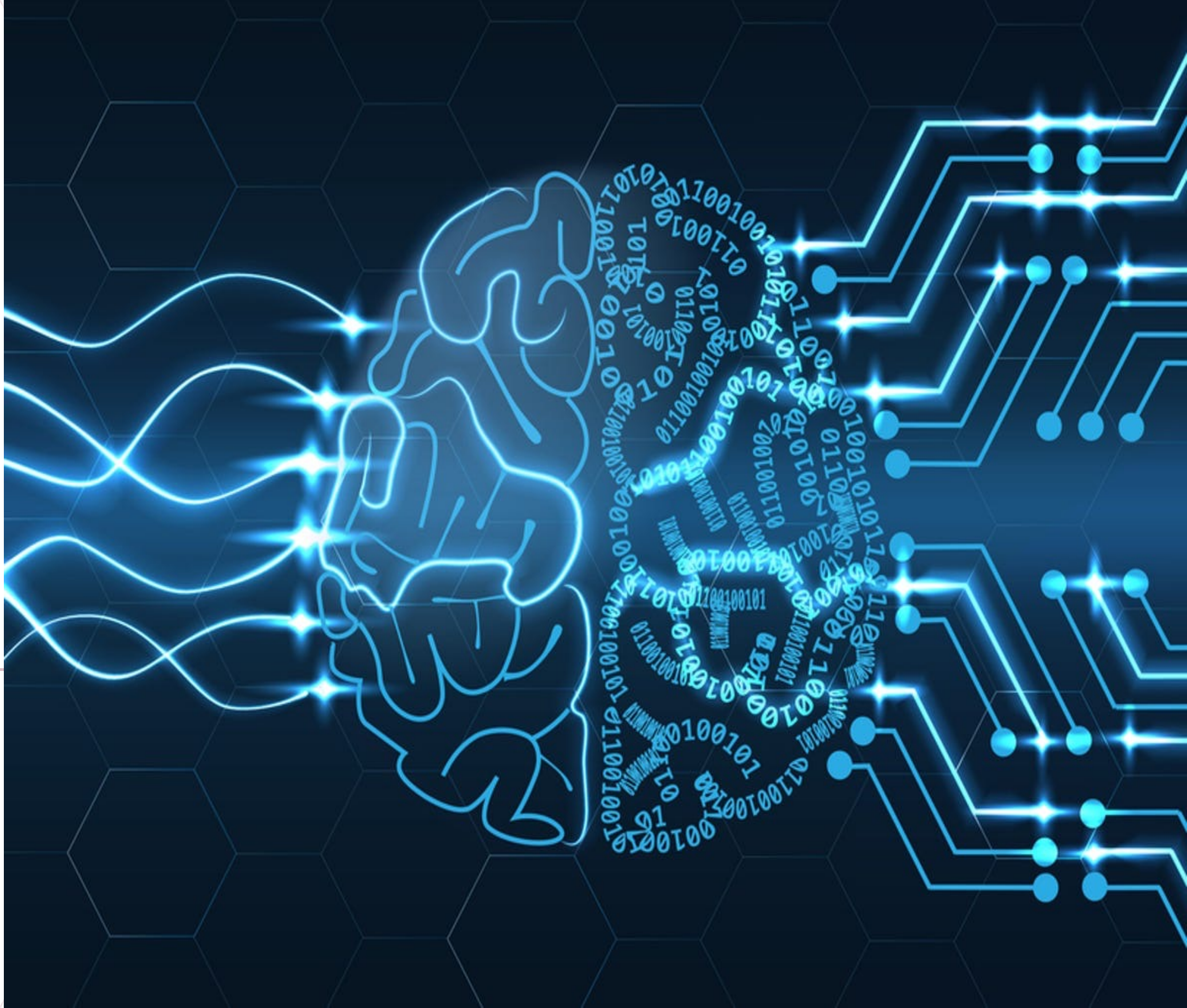
VAE

SNU SCSC

Oct 11 , 2022.

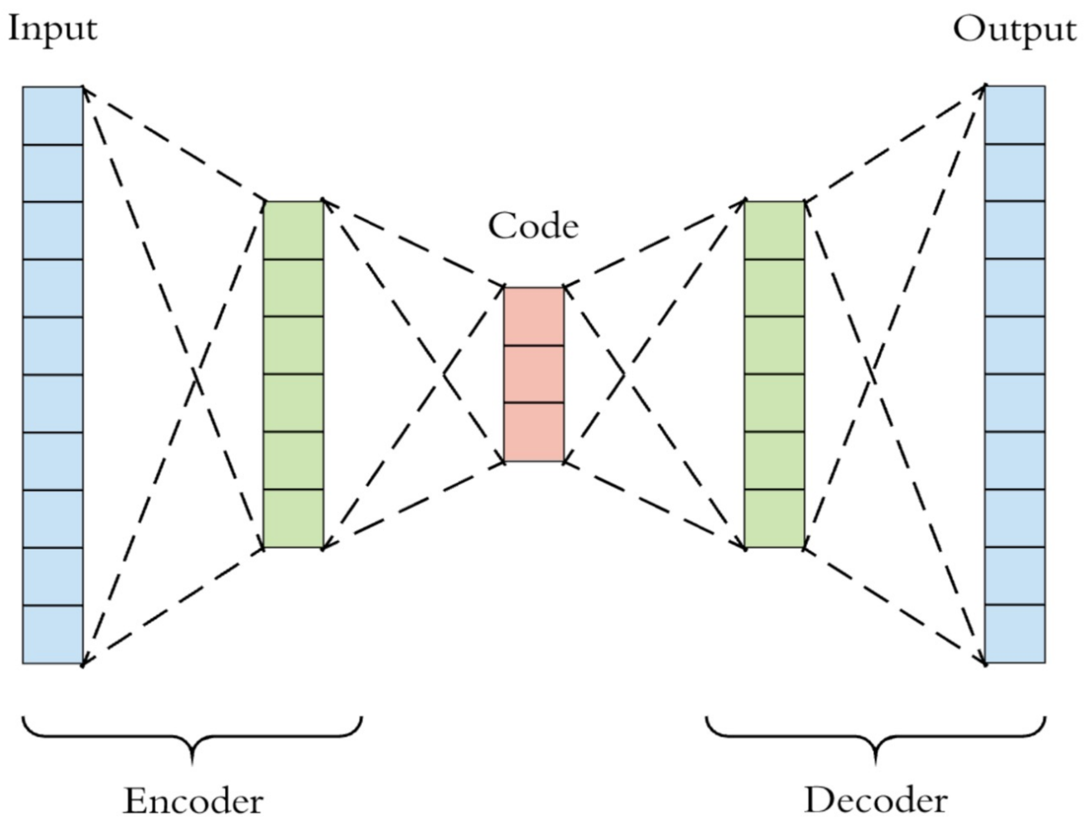
Seoyoung Lim



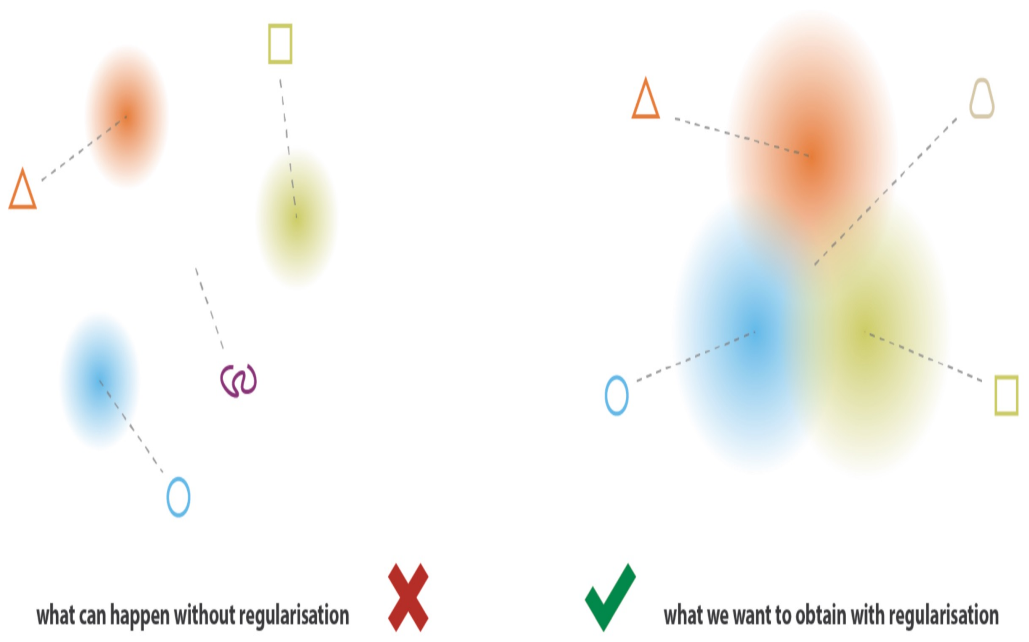
Autoencoder – Traditional Autoencoder

- Encoder : Compresses the input data a latent – space representation.

- Decoder : Decoded the encoded data back to original dimension.



Models – Why Variational AutoEncoder



- Traditional AutoEncoder Problem : the latent vectors (encoding) tends to be irregular, unorganized, uniterpretable

Models – VAE(s)

Code

```
def _build_net(self):
    self.x_hat_tfph = tf.placeholder(tf.float32, shape=[None, *self.image_size], name='input_img')
    self.x_tfph = tf.placeholder(tf.float32, shape=[None, *self.image_size], name='target_img')
    self.keep_prob_tfph = tf.placeholder(tf.float32, name='keep_prob')
    self.z_in_tfph = tf.placeholder(tf.float32, shape=[None, self.flags.z_dim], name='latent_variable')

    # encoding
    mu, sigma = self.encoder(self.x_hat_tfph)
    # sampling by re-parameterization technique
    self.z = mu + sigma * tf.random_normal(tf.shape(mu), mean=0., stddev=1., dtype=tf.float32)

    # decoding
    y = self.decoder(self.z)
    self.y = tf.clip_by_value(y, 1e-8, 1 - 1e-8)
    sample_y = self.decoder(self.z_in_tfph, is_reuse=True)
    self.sample_y = tf.clip_by_value(sample_y, 1e-8, 1 - 1e-8)

    # loss
    marginal_likelihood = tf.reduce_sum(self.x_tfph * tf.log(self.y) + (1 - self.x_tfph) * tf.log(1 - self.y),
                                        [1, 2, 3])
    KL_divergence = 0.5 * tf.reduce_sum(tf.square(mu) + tf.square(sigma) - tf.log(1e-8 + tf.square(sigma)) - 1, 1)

    self.marginal_likelihood = tf.reduce_mean(marginal_likelihood)
    self.KL_divergence = tf.reduce_mean(KL_divergence)
    self.ELBO = self.marginal_likelihood - self.KL_divergence
    self.loss = - self.ELBO

    self.train_op = tf.train.AdamOptimizer(self.flags.learning_rate).minimize(self.loss)
```

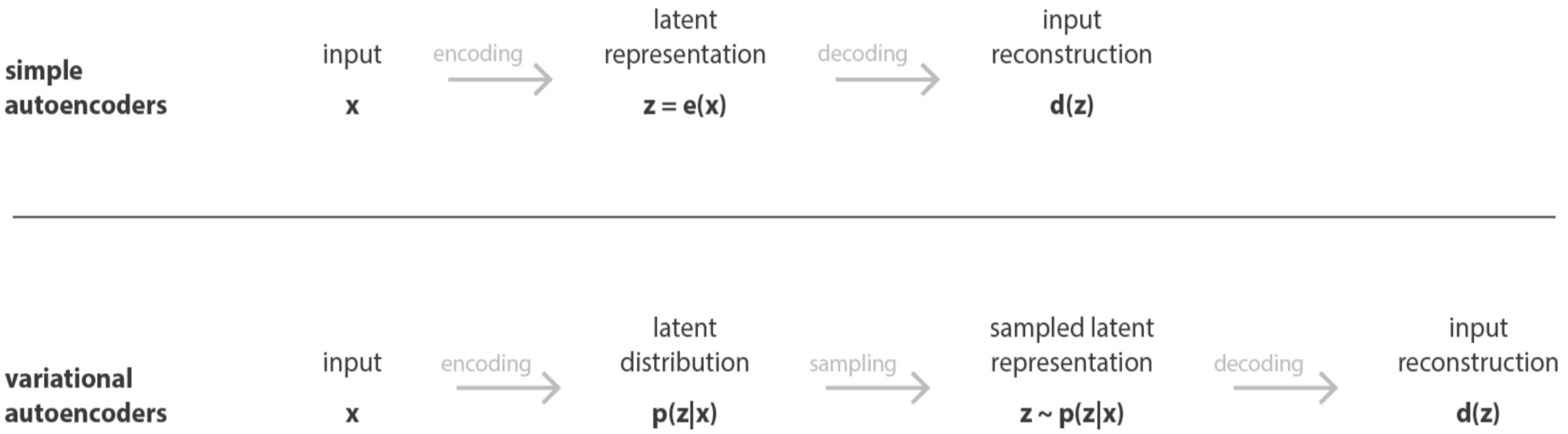
Models – Loss

MSE vs Cross Entropy

$$MSE = \frac{1}{n}\Sigma\underbrace{\left(y - \widehat{y}\right)^2}_{\text{The square of the difference between actual and predicted}}$$

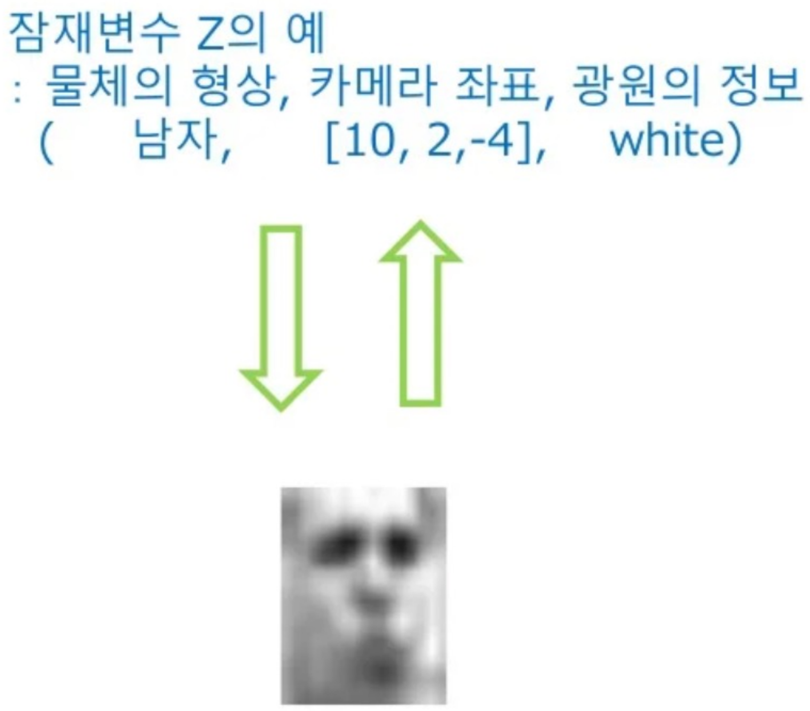
$$E(\boldsymbol{w}) \equiv \frac{1}{2}\frac{1}{|D|}\sum_{d\in D}(y_d - \hat{y}_d)^2$$

Models – VAE(s)


simple autoencoders
input
x
encoding
latent representation
z = e(x)
decoding
input reconstruction
d(z)
variational autoencoders
input
x
encoding
latent distribution
p(z|x)
sampling
sampled latent representation
z ~ p(z|x)
decoding
input reconstruction
d(z)

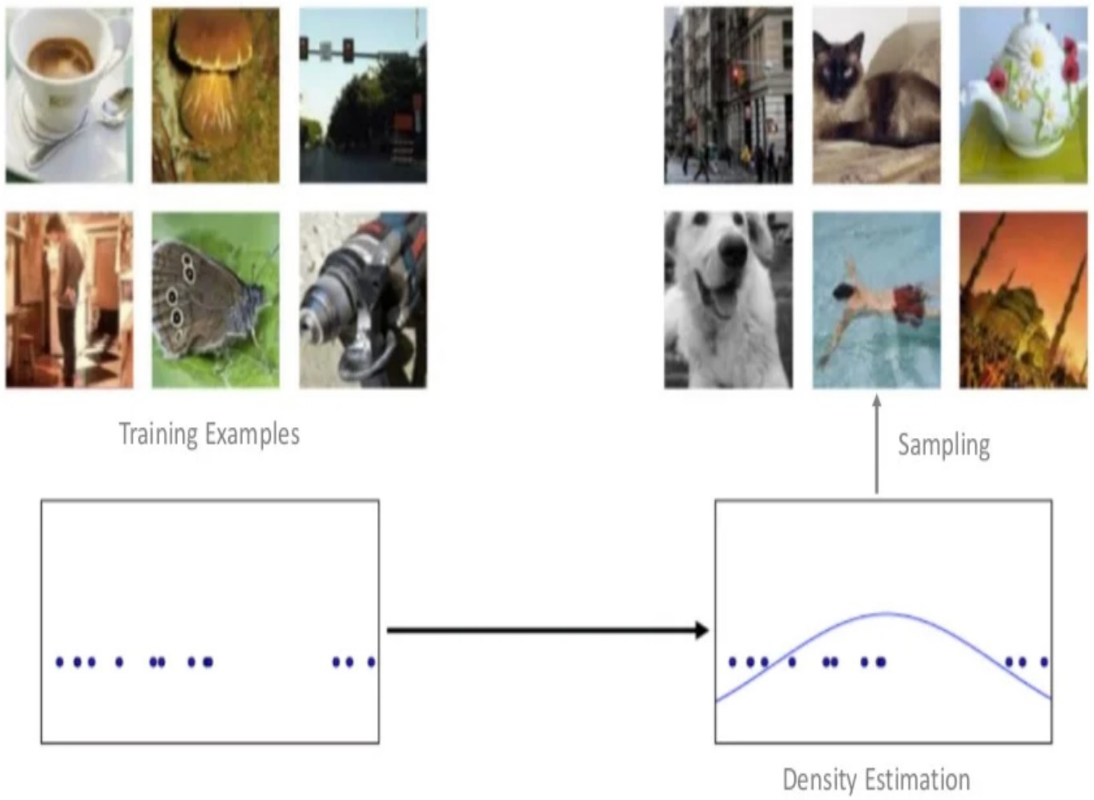
Models – Latent Space

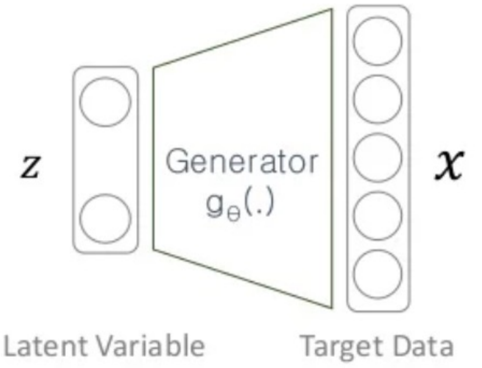
X : 이미지 -> Z : 잠재변수



이미지 X

Models – Generative Model





$z \sim p(z)$ Random variable

$g_\theta(\cdot)$ Deterministic function parameterized by θ

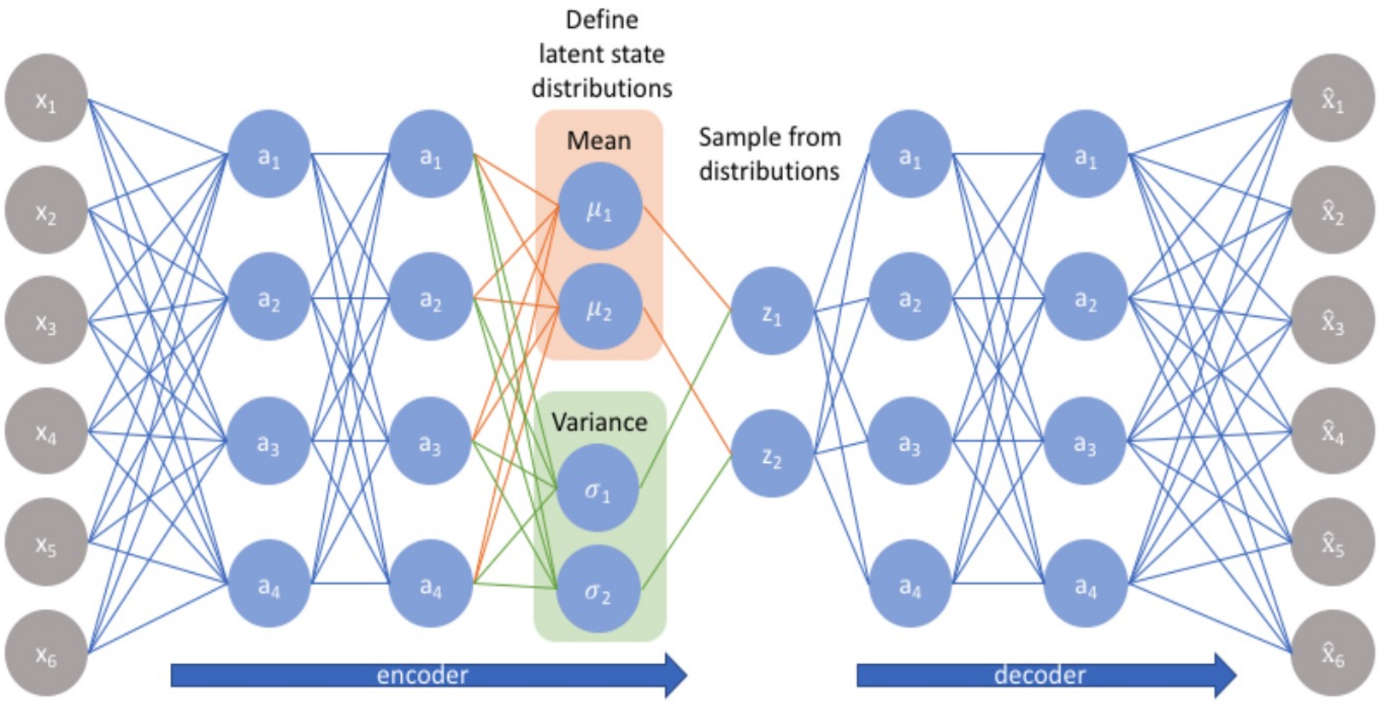
$x = g_\theta(z)$ Random variable

$$p(x|g_\theta(z)) = p_\theta(x|z)$$

우리가 최종적으로 알고자 하는 것은 P(x) 라는 최종 결과값

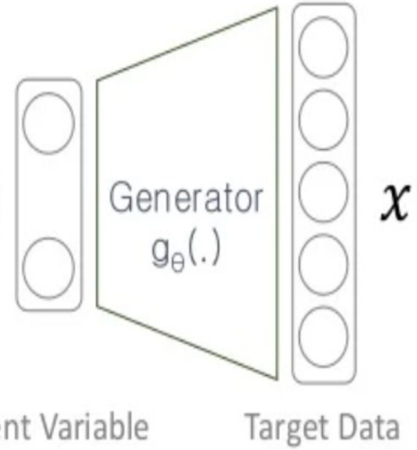
$$\int p(x|g_\theta(z))p(z)dz = p(x)$$

Models – VAE(s)



[Source: Jeremy Jordan]

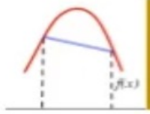
$p(z|x) \approx q_\phi(z|x) \sim z$



Models – VAE(s)

Relationship among $p(x), p(z|x), q_\phi(z|x)$: Derivation 1

$$\log(p(x)) = \log\left(\int p(x|z)p(z)dz\right) \geq \int \log(p(x|z))p(z)dz \quad \leftarrow$$



[Jensen's Inequality]

For concave functions f(.)
f(E[x])≥E[f(x)]

f(.) = log(.) is concave

$$\log(p(x)) = \log\left(\int p(x|z)\frac{p(z)}{q_\phi(z|x)}q_\phi(z|x)dz\right) \geq \int \log\left(p(x|z)\frac{p(z)}{q_\phi(z|x)}\right)q_\phi(z|x)dz \leftarrow \text{Variational inference}$$

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

$$= \underbrace{\mathbb{E}_{q_\phi(z|x)}[\log(p(x|z))] - KL(q_\phi(z|x)||p(z))}_{ELBO(\phi)}$$

Variational lower bound
Evidence lower bound (ELBO)

Models – VAE(s)

Relationship among $p(x), p(z|x), q_\phi(z|x)$: Derivation 2

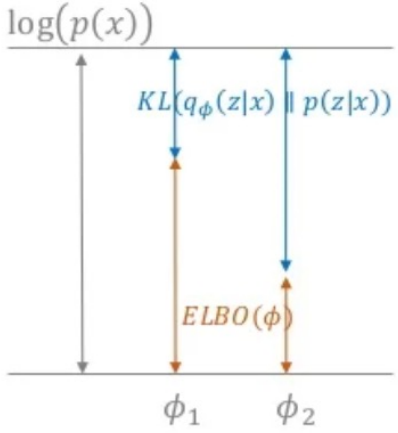
$$\log(p(x)) = \int \log(p(x))\,q_\phi(z|x)dz \qquad \leftarrow \int q_\phi(z|x)dz = 1$$

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

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

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

두 확률분포 간의 거리≥0



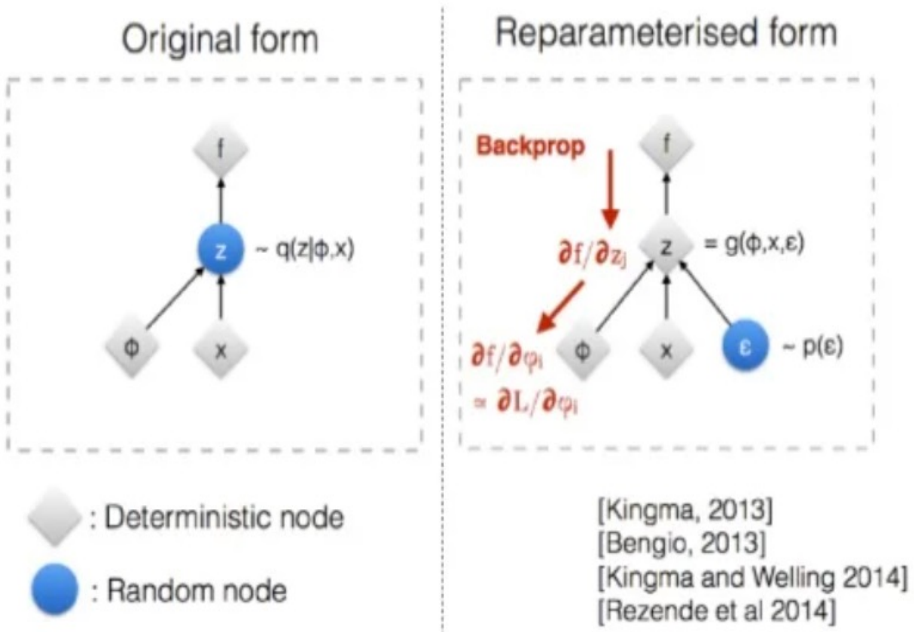
Models – VAE(s)

Code

```
def _build_net(self):
    self.x_hat_tfph = tf.placeholder(tf.float32, shape=[None, *self.image_size], name='input_img')
    self.x_tfph = tf.placeholder(tf.float32, shape=[None, *self.image_size], name='target_img')
    self.keep_prob_tfph = tf.placeholder(tf.float32, name='keep_prob')
    self.z_in_tfph = tf.placeholder(tf.float32, shape=[None, self.flags.z_dim], name='latent_variable')

    # encoding
    mu, sigma = self.encoder(self.x_hat_tfph)
    # sampling by re-parameterization technique
    self.z = mu + sigma * tf.random_normal(tf.shape(mu), mean=0., stddev=1., dtype=tf.float32)

    # decoding
    y = self.decoder(self.z)
    self.y = tf.clip_by_value(y, 1e-8, 1 - 1e-8)
    sample_y = self.decoder(self.z_in_tfph, is_reuse=True)
    self.sample_y = tf.clip_by_value(sample_y, 1e-8, 1 - 1e-8)

    # loss
    marginal_likelihood = tf.reduce_sum(self.x_tfph * tf.log(self.y) + (1 - self.x_tfph) * tf.log(1 - self.y),
                                        [1, 2, 3])
    KL_divergence = 0.5 * tf.reduce_sum(tf.square(mu) + tf.square(sigma) - tf.log(1e-8 + tf.square(sigma)) - 1, 1)

    self.marginal_likelihood = tf.reduce_mean(marginal_likelihood)
    self.KL_divergence = tf.reduce_mean(KL_divergence)
    self.ELBO = self.marginal_likelihood - self.KL_divergence
    self.loss = - self.ELBO

    self.train_op = tf.train.AdamOptimizer(self.flags.learning_rate).minimize(self.loss)
```

Models – VAE(s) : Backpropagation

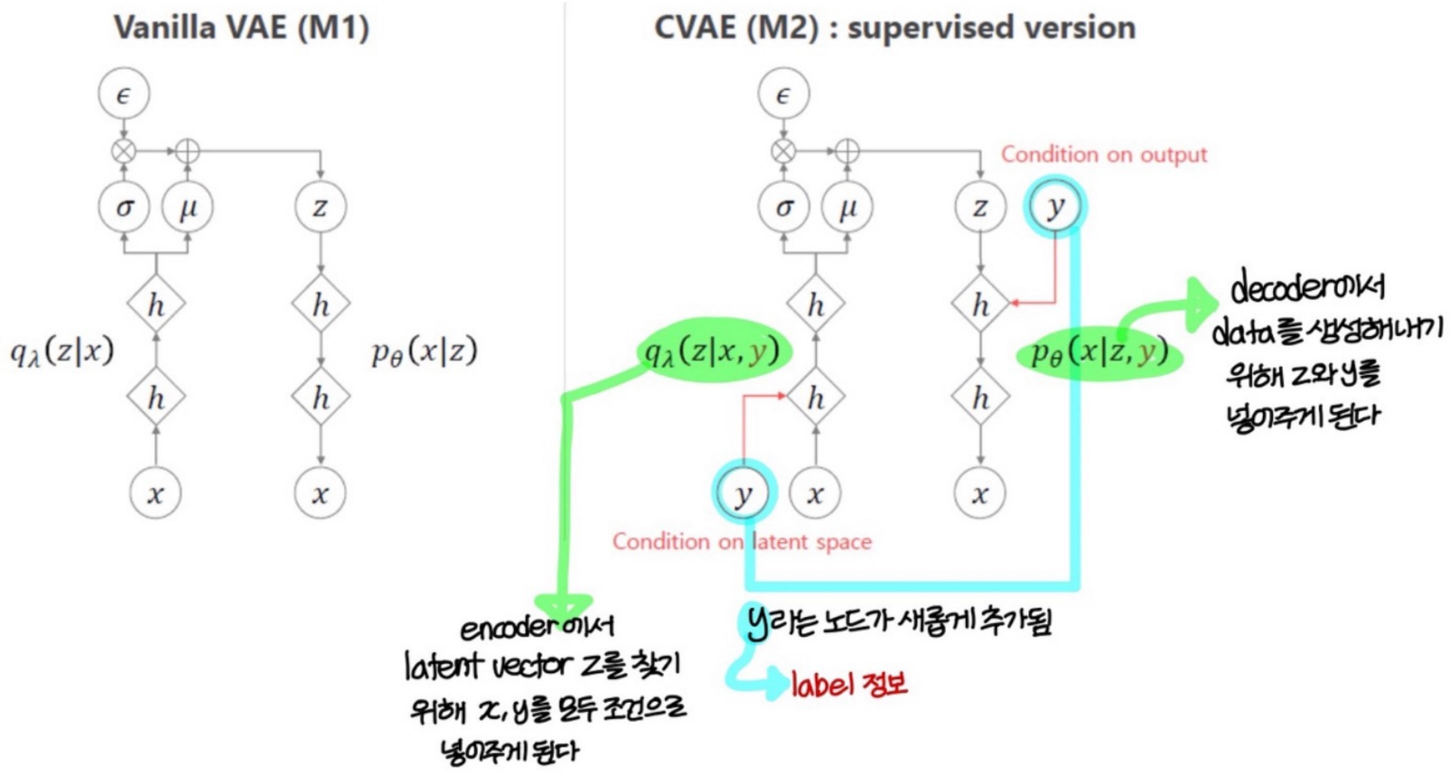
Reparameterization Trick



Sampling Process

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

Models – Contrastive Autoencoder



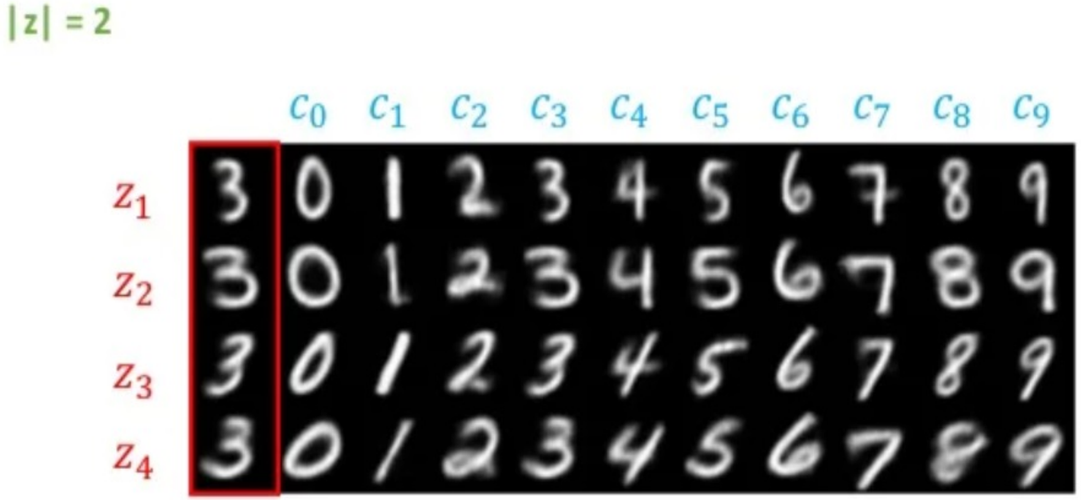
Ex) Encoder, Decoder 에 레이블의 정보를 입력해주는것

Models – CVAE (CODE)

```
# Gaussian MLP as conditional encoder
def gaussian_MLP_conditional_encoder(x, y, n_hidden, n_output, keep_prob):
    with tf.variable_scope("gaussian_MLP_encoder"):
        # concatenate condition and image
        dim_y = int(y.get_shape()[1])
        input = tf.concat(axis=1, values=[x, y])

        # initializers
        w_init = tf.contrib.layers.variance_scaling_initializer()
        b_init = tf.constant_initializer(0.)

        # 1st hidden layer
        w0 = tf.get_variable('w0', [input.get_shape()[1], n_hidden+dim_y], initializer=w_init)
        b0 = tf.get_variable('b0', [n_hidden+dim_y], initializer=b_init)
        h0 = tf.matmul(input, w0) + b0
        h0 = tf.nn.elu(h0)
        h0 = tf.nn.dropout(h0, keep_prob)

        # 2nd hidden layer
        w1 = tf.get_variable('w1', [h0.get_shape()[1], n_hidden], initializer=w_init)
        b1 = tf.get_variable('b1', [n_hidden], initializer=b_init)
        h1 = tf.matmul(h0, w1) + b1
        h1 = tf.nn.tanh(h1)
        h1 = tf.nn.dropout(h1, keep_prob)
```

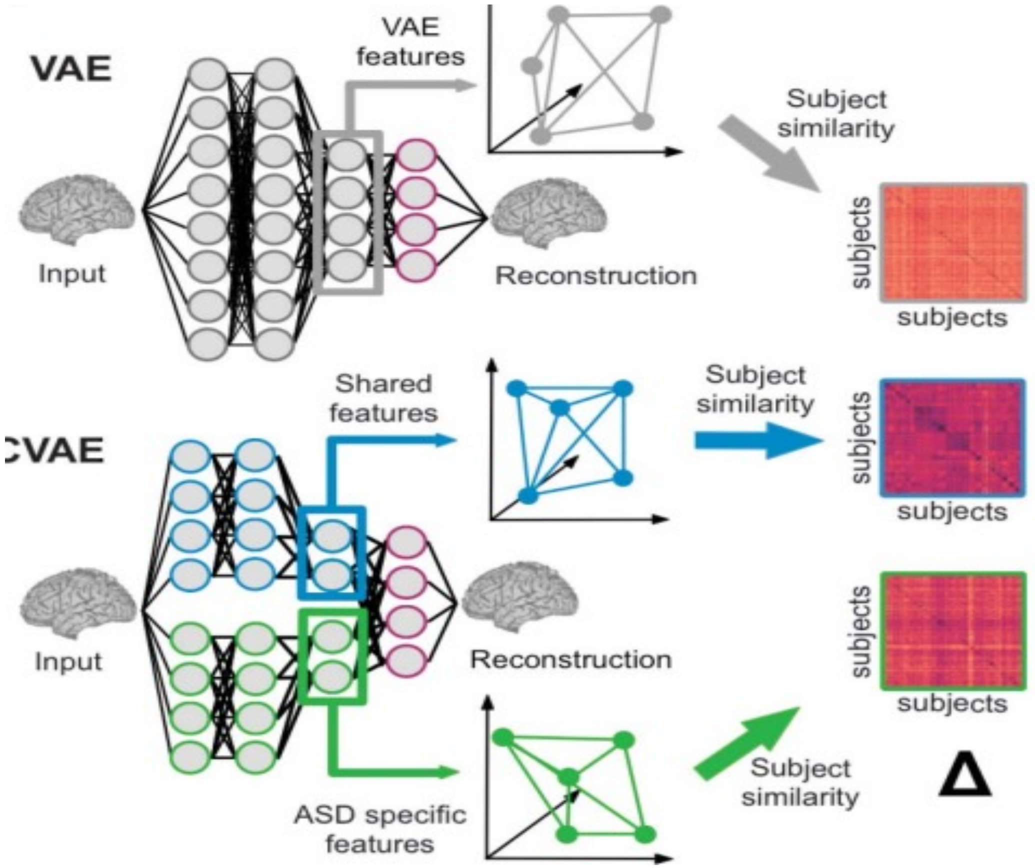
CVAE – Result

Z-sampling



Handwriting style for a given z must be preserved for all labels

Models – Application



VAE

(1) Purpose - to test whether association between neuroanatomy and ASD symptoms can be identified using VAE

(2) Used Representational Similarity Analysis(RSA) - to correlated with individual variation in the ASD patients' characteristics (Scanner type, age etc.)