2020.07 하계방학

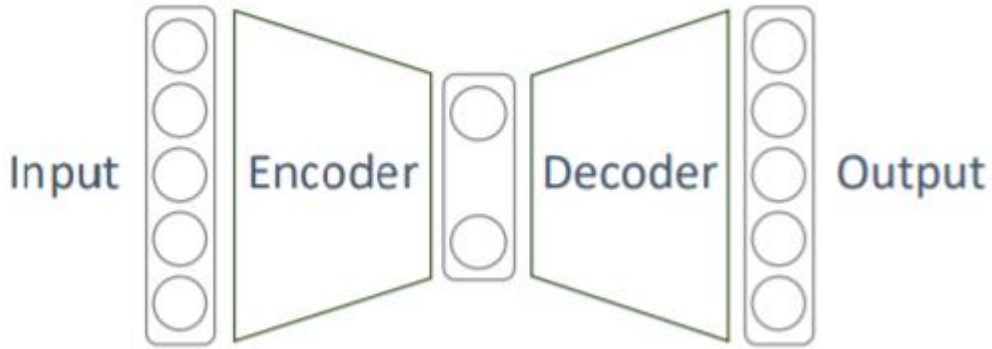
VAE

Variational Auto Encoder

20151739 공대현

- **Prior Knowledge**

  -Autoencoder

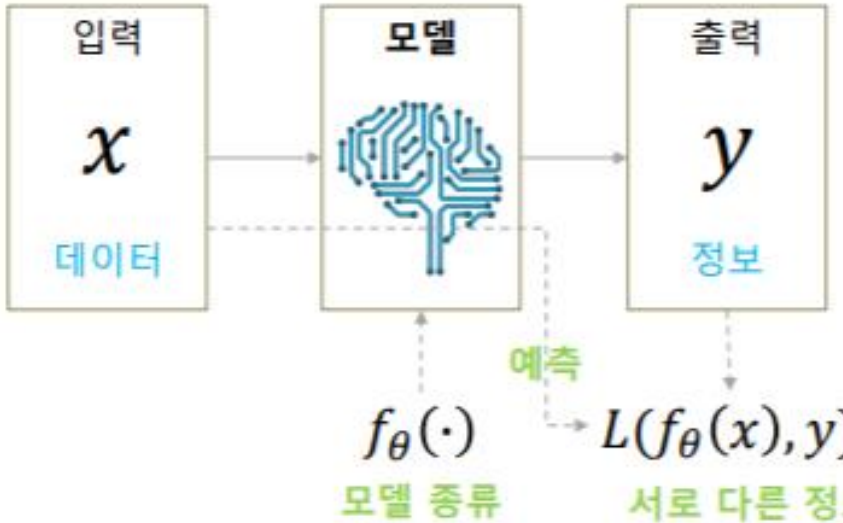


[Keyword]

- ❖ Unsupervised learning
- ❖ Representation learning = Efficient coding learning
- ❖ Dimensionality reduction
- ❖ Generative model learning

- # Prior Knowledge

-Update parameter



θ : $parameter\ set\ of\ Weight\ and\ Bias$

θ^*: $optimized\ parmeter\ set\ \ for\ loss\ fuction$

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

주어진 데이터를 제일 잘 설명하는 모델 찾기

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

고정 입력, 고정 출력

• Prior Knowledge

-Gradient Descent

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

Questions	*Strategies*
How to update $\theta \rightarrow \theta + \Delta\theta$	Only if $L(\theta + \Delta\theta) < L(\theta)$
When we stop to search??	If $L(\theta + \Delta\theta) == L(\theta)$
How to find $\Delta\theta$ so that $L(\theta + \Delta\theta) < L(\theta)$?	$\Delta\theta = -\eta\nabla L$, where $\eta > 0$

Taylor Expansion → $L(\theta + \Delta\theta) = L(\theta) + \nabla L \cdot \Delta\theta + second\ derivative + third\ derivative + \cdots$

Approximation → $L(\theta + \Delta\theta) \approx L(\theta) + \nabla L \cdot \Delta\theta$ 더 많은 차수를 사용할 수록 더 넓은 지역을 작은 오차로 표현 가능

If $\Delta\theta = -\eta\Delta L$, then $\Delta L = -\eta\|\nabla L\|^2 < 0$, where $\eta > 0$ and called learning rate

∇L is gradient of L and indicates the steepest increasing direction of L

• Prior Knowledge

-Gradient Descent

Redefinition → $\nabla L(\theta_k, \mathcal{D}) \triangleq \sum_i \nabla L(\theta_k, \mathcal{D}_i) / N$

Stochastic Gradient Descent → $\nabla L(\theta_k, \mathcal{D}) \approx \sum_j \nabla L(\theta_k, \mathcal{D}_i) / M$, where $M < N$

M : batch size

$$\theta_{k+1} = \theta_k - \eta \nabla L(\theta_k, \mathcal{D})$$

$$w_{k+1}^l = w_k^l - \eta \nabla_{w_k^l} L(\theta_k, \mathcal{D})$$

$$b_{k+1}^l = b_k^l - \eta \nabla_{b_k^l} L(\theta_k, \mathcal{D})$$

[Backpropagation Algorithm]

1. Error at the output layer

$$\delta^L = \nabla_a C \odot \sigma'(z^L)$$

- C : Cost (Loss)
- a : final output of DNN
- $\sigma(\cdot)$: activation function

2. Error relationship between two adjacent layers

$$\delta^l = \sigma'(z^l) \odot \left((w^{l+1})^T \delta^{l+1} \right)$$

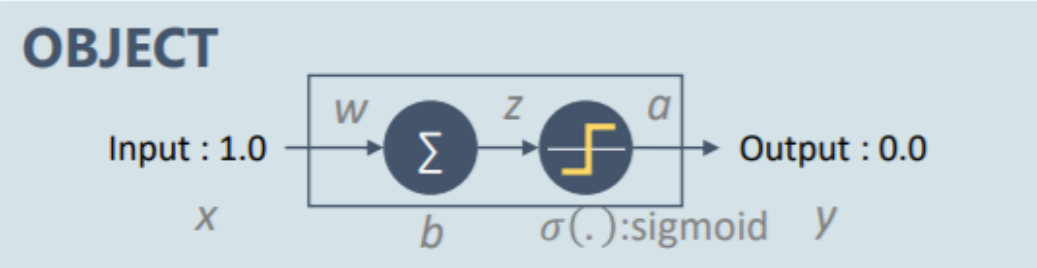
3. Gradient of C in terms of bias $\quad \nabla_{b^l} C = \delta^l$

4. Gradient of C in terms of weight $\quad \nabla_{W^l} C = \delta^l (a^{l-1})^T$

• Prior Knowledge

-Loss Function

Type 1 : Mean Square Error / Quadratic loss



$$C = (a-y)^2/2 = a^2/2$$

$$\nabla_a C = (a-y)$$

$$\delta = \nabla_a C \odot \sigma'(z) = (a-y)\sigma'(z)$$

$$\frac{\partial C}{\partial w} = x\delta = \delta \qquad \frac{\partial C}{\partial b} = \delta$$

$$w = w - \eta\delta \qquad b = b - \eta\delta$$

$z = Wx+b$

output $= a = \sigma(z)$

Cost $=$ Loss $= C = (a-y)^2/N$ $\quad (N = \text{len}(x))$

$= (\sigma(z)-y)^2/2$

$$\frac{\partial C}{\partial w} = (\sigma(z)-y)\cdot\sigma'(z)\frac{\partial z}{\partial w} = (\sigma(Wx+b)-y)\cdot\sigma'(Wx+b)\cdot x = \delta_w$$

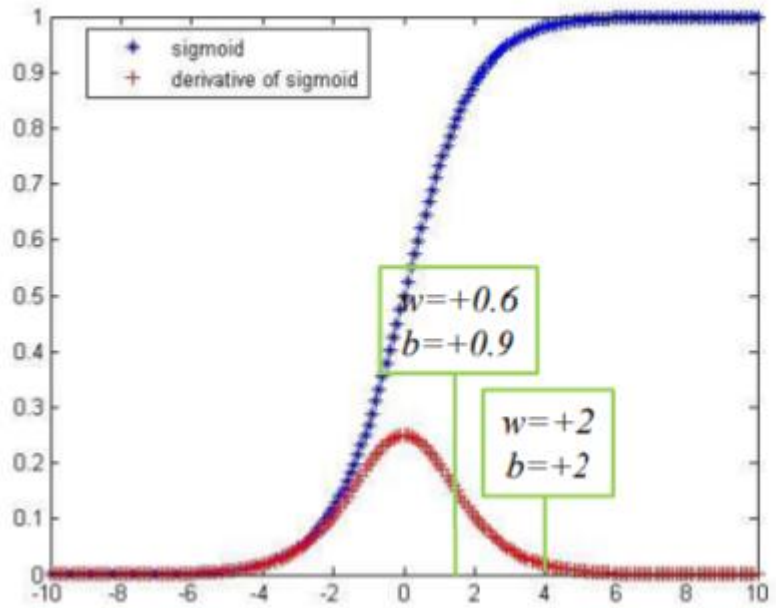
$$\frac{\partial C}{\partial b} = (\sigma(z)-y)\cdot\sigma'(z)\cdot\frac{\partial z}{\partial b} = (\sigma(Wx+b)-y)\cdot\sigma'(Wx+b) = \delta_b$$

$$\begin{cases} W = W - \eta\delta_w \\ b = b - \eta\delta_b \end{cases}$$

- # Prior Knowledge

-Loss Function

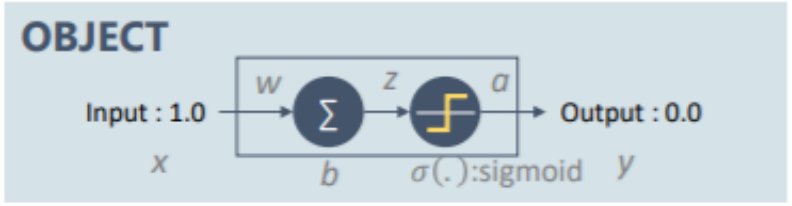
MSE의 단점: sigmoid의 미분값이 곱해져 gradient vanishing 현상이 일어난다.



• Prior Knowledge

-Loss Function

Type 2 : Cross Entropy



MSE와는 달리 CE는 출력 레이어에서의
에러값에 activation function의 미분값이
곱해지지 않아 gradient vanishing problem에서
좀 더 자유롭다.
(학습이 좀 더 빨리 된다)

그러나 레이어가 여러 개가 사용될 경우에는
결국 activation function의 미분값이 계속해서
곱해지므로 gradient vanishing problem에서
완전 자유로울 수 없다.

ReLU는 미분값이 1 혹은 0이므로 이러한
관점에서 훌륭한 activation function이다.

$$C = -[y \ln a + (1-y)\ln(1-a)]$$

$$\nabla_a C = -\frac{y}{a} - (1-y)\frac{-1}{1-a} = \frac{y-a}{(1-a)a}$$

$$= -\frac{y}{a} + \frac{(1-y)}{1-a}$$

$$= \frac{-(1-a)y}{(1-a)a} + \frac{(1-y)a}{(1-a)a}$$

$$= \frac{-y + ay + a - ay}{(1-a)a}$$

$$= \frac{a-y}{(1-a)a}$$

Sigmoid 미분 -> 자기자신(1-자기자신)

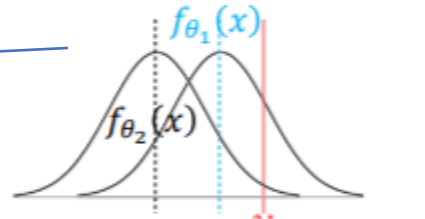
$$\sigma'(z) = \frac{\partial a}{\partial z} = \sigma'(z) = (1-\sigma(z))\sigma(z) = (1-a)a$$

$$\delta_{MSE} = (a-y)\sigma'(z) \dashrightarrow \delta_{CE} = \nabla_a C \odot \sigma'(z^L) = \frac{a-y}{(1-a)a}(1-a)a = a-y$$

• Prior Knowledge

-Maximum likelihood ($P(A|B)$)

f_θ의 확률분포의 평균이 y에 가깝게 만듦



$p(y|f_{\theta_2}(x)) < p(y|f_{\theta_1}(x))$

예측

$f_\theta(\cdot)$ → $p(y|f_\theta(x))$

모델 종류

정해진 확률분포에서 출력이 나올 확률

$$\theta^* = \underset{\theta}{\operatorname{argmin}}[-\log(p(y|f_\theta(x)))]$$

주어진 데이터를 제일 잘 설명하는 모델 찾기

$$-\log(p(y|f_\theta(x))) = -\sum_i \log(p(y_i|f_\theta(x_i)))$$

$$y_{new} \sim p(y|f_{\theta^*}(x_{new}))$$

고정 입력, 고정/다른 출력

• Prior Knowledge

- Setting distribution

Univariate cases

$$-\log(p(y_i|f_\theta(x_i)))$$

연속

Gaussian distribution

$$f_\theta(x_i) = \mu_i, \sigma_i = 1$$

$$p(y_i|\mu_i,\sigma_i) = \frac{1}{\sqrt{2\pi}\sigma_i}\exp\left(-\frac{(y_i-\mu_i)^2}{2\sigma_i^2}\right)$$

$$\log(p(y_i|\mu_i,\sigma_i)) = \log\frac{1}{\sqrt{2\pi}\sigma_i} - \frac{(y_i-\mu_i)^2}{2\sigma_i^2}$$

$$-\log(p(y_i|\mu_i)) = -\log\frac{1}{\sqrt{2\pi}} + \frac{(y_i-\mu_i)^2}{2}$$

$$-\log(p(y_i|\mu_i)) \propto \frac{(y_i-\mu_i)^2}{2} = \frac{(y_i-f_\theta(x_i))^2}{2}$$

Mean Squared Error

이산

Bernoulli distribution

$$f_\theta(x_i) = p_i$$

$$p(y_i|p_i) = p_i^{y_i}(1-p_i)^{1-y_i}$$

$$\log(p(y_i|p_i)) = y_i\log p_i + (1-y_i)\log(1-p_i)$$

$$-\log(p(y_i|p_i)) = -[y_i\log p_i + (1-y_i)\log(1-p_i)]$$

Cross-entropy

• Prior Knowledge

- Setting distribution

Multivariate cases

$$-\log(p(y_i|f_\theta(x_i)))$$

Gaussian distribution

$$f_\theta(x_i) = \mu_i, \Sigma_i = I$$

$$p(y_i|\mu_i,\Sigma_i) = \frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}}\exp\left(-\frac{(y_i-\mu_i)^T\Sigma_i^{-1}(y_i-\mu_i)}{2}\right)$$

$$\log(p(y_i|\mu_i,\Sigma_i)) = \log\frac{1}{(2\pi)^{n/2}|\Sigma_i|^{1/2}} - \frac{(y_i-\mu_i)^T\Sigma_i^{-1}(y_i-\mu_i)}{2}$$

$$-\log(p(y_i|\mu_i)) = -\log\frac{1}{(2\pi)^{n/2}} + \frac{\|y_i-\mu_i\|_2^2}{2}$$

$$-\log(p(y_i|\mu_i)) \propto \frac{\|y_i-\mu_i\|_2^2}{2} = \frac{\|y_i-f_\theta(x_i)\|_2^2}{2}$$

Mean Squared Error

Categorical distribution

$$f_\theta(x_i) = p_i$$

$$p(y_i|p_i) = \prod_{j=1}^{n} p_{i,j}^{y_{i,j}}(1-p_{i,j})^{1-y_{i,j}}$$

$$\log(p(y_i|p_i)) = \sum_{j=1}^{n}(y_{i,j}\log p_{i,j} + (1-y_{i,j})\log(1-p_{i,j}))$$

$$-\log(p(y_i|p_i)) = -\sum_{j=1}^{n}(y_{i,j}\log p_{i,j} + (1-y_{i,j})\log(1-p_{i,j}))$$

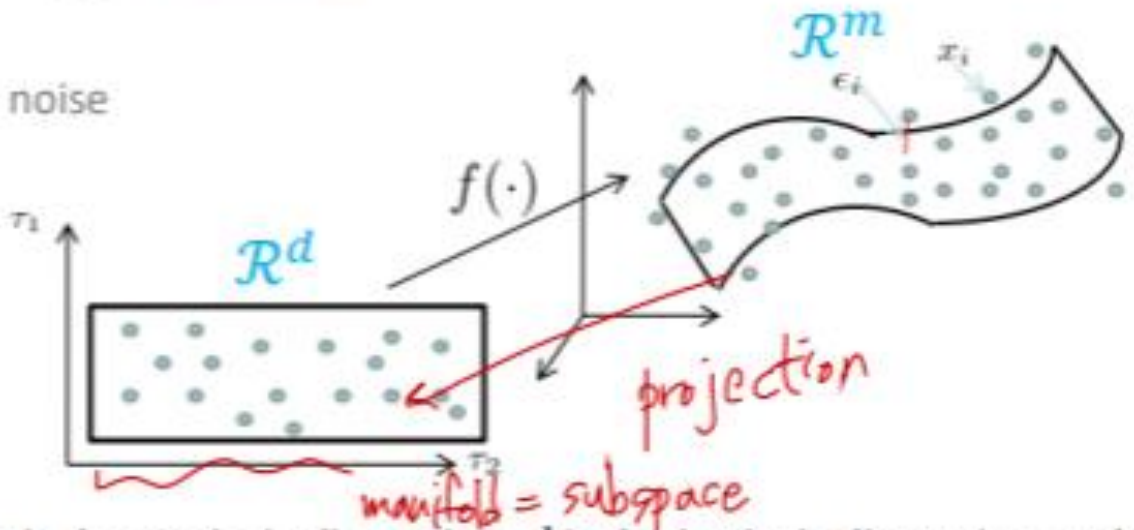
Cross-entropy

• Manifold learning

- A d dimensional manifold $\mathcal{M}$ is embedded in an m dimensional space, and there is an explicit mapping $f: \mathcal{R}^d \rightarrow \mathcal{R}^m$ *where* $d \leq m$

- We are given samples $x_i \in \mathcal{R}^m$ with noise

$$x_i = f(\tau_i) + \epsilon_i$$



- $f(\cdot)$ is called embedding function, m is the extrinsic dimension, d is the intrinsic dimension or the dimension of the latent space

- Finding $f(\cdot)$ or τ_i from the given x_i is called manifold learning

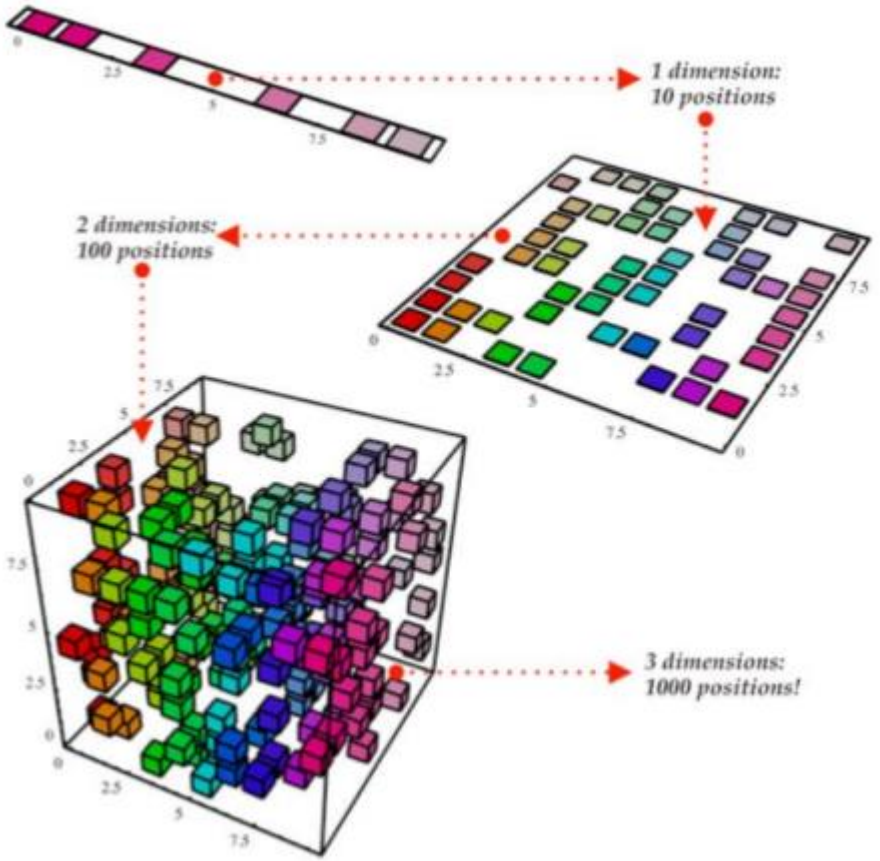
- We assume $p(\tau)$ is smooth, is distributed uniformly, and noise is small → Manifold Hypothesis

- # Manifold learning

-차원의 저주

데이터의 차원이 증가할수록 해당 공간의 크기(부피)가 기하급수적으로 증가
→ 동일한 개수의 데이터의 밀도는 차원이 증가할수록 급속도로 희박해짐

∴ 차원이 증가할수록 데이터의 분포 분석 또는 모델추정에 필요한 샘플 데이터의 개수가 기하급수적으로 증가

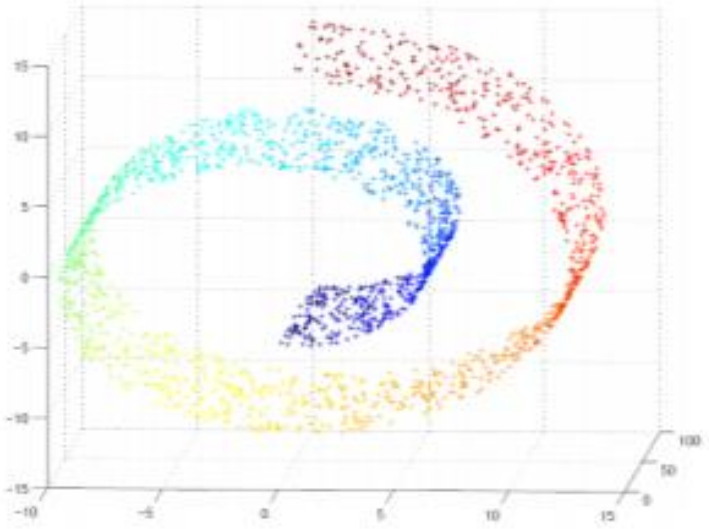


- # Manifold learning

-Manifold hypothesis (assumption)

Natural data in high dimensional spaces concentrates close to lower dimensional manifolds.
고차원의 데이터의 밀도는 낮지만, 이들의 집합을 포함하는 저차원의 매니폴드가 있다.

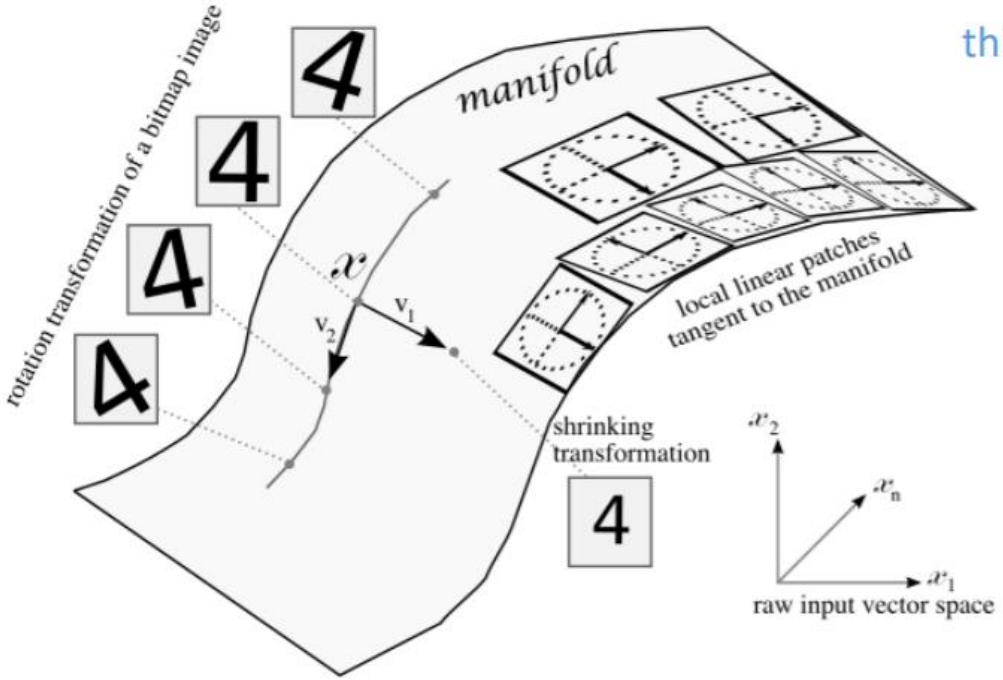
Probability density decreases very rapidly when moving away from the supporting manifold.
이 저차원의 매니폴드를 벗어나는 순간 급격히 밀도는 낮아진다.

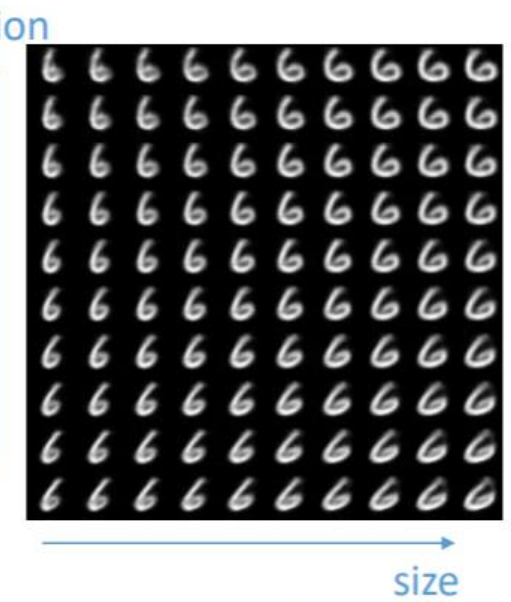


• Manifold learning

Manifold follows naturally from continuous underlying factors (≈intrinsic manifold coordinates)
Such continuous factors are part of a meaningful representation!

매니폴드 학습 결과 평가를 위해 매니폴드 좌표들이 조금씩 변할 때 원 데이터도 유의미하게 조금씩 변함을 보인다.

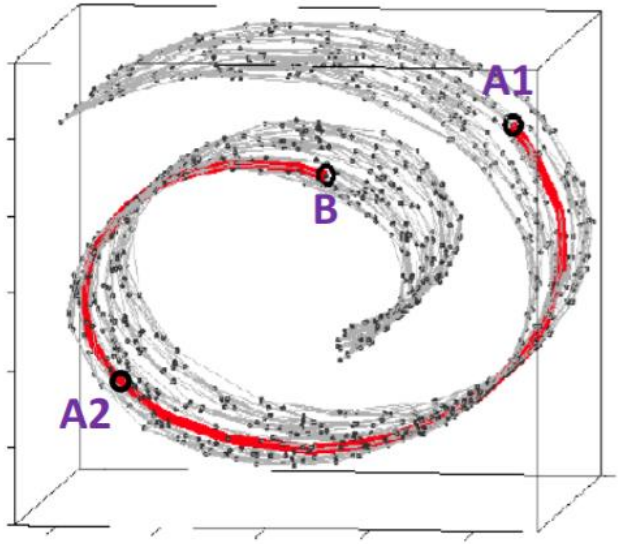




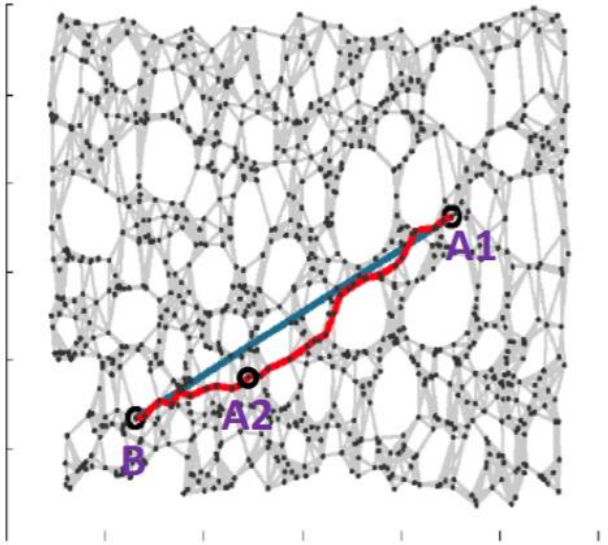
- # Manifold learning

-Reasonable distance metric

의미적으로 가깝다고 생각되는 고차원 공간에서의 두 샘플들 간의 거리는 먼 경우가 많다.
고차원 공간에서 가까운 두 샘플들은 의미적으로는 굉장히 다를 수 있다.
차원의 저주로 인해 고차원에서의 유의미한 거리 측정 방식을 찾기 어렵다.



Distance in high dimension

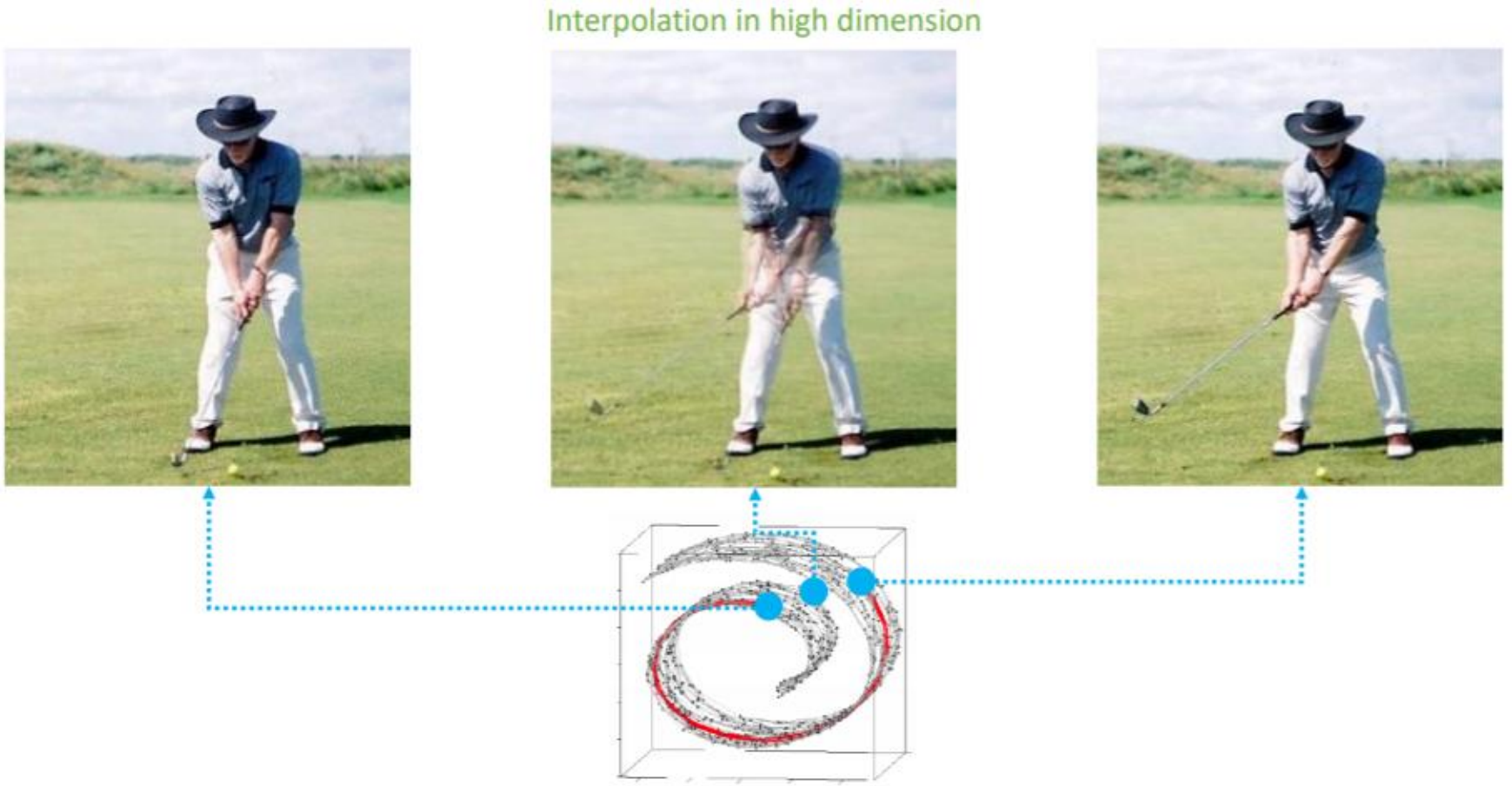


Distance in manifold

중요한 특징들을
찾았다면 이 특징을
공유하는 샘플들도
찾을 수 있어야 한다.

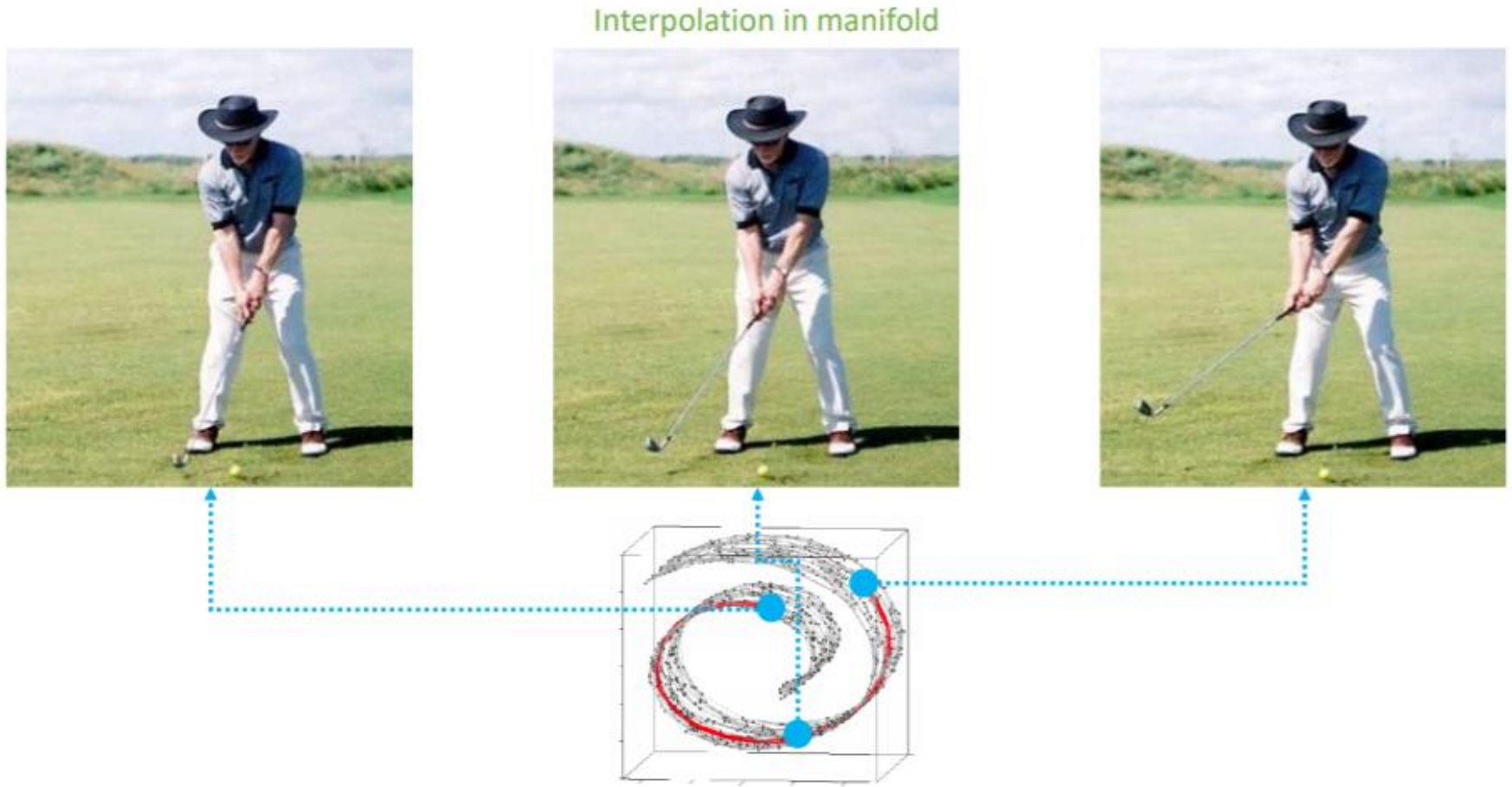
- **Manifold learning**

  -Reasonable distance metric



- **Manifold learning**

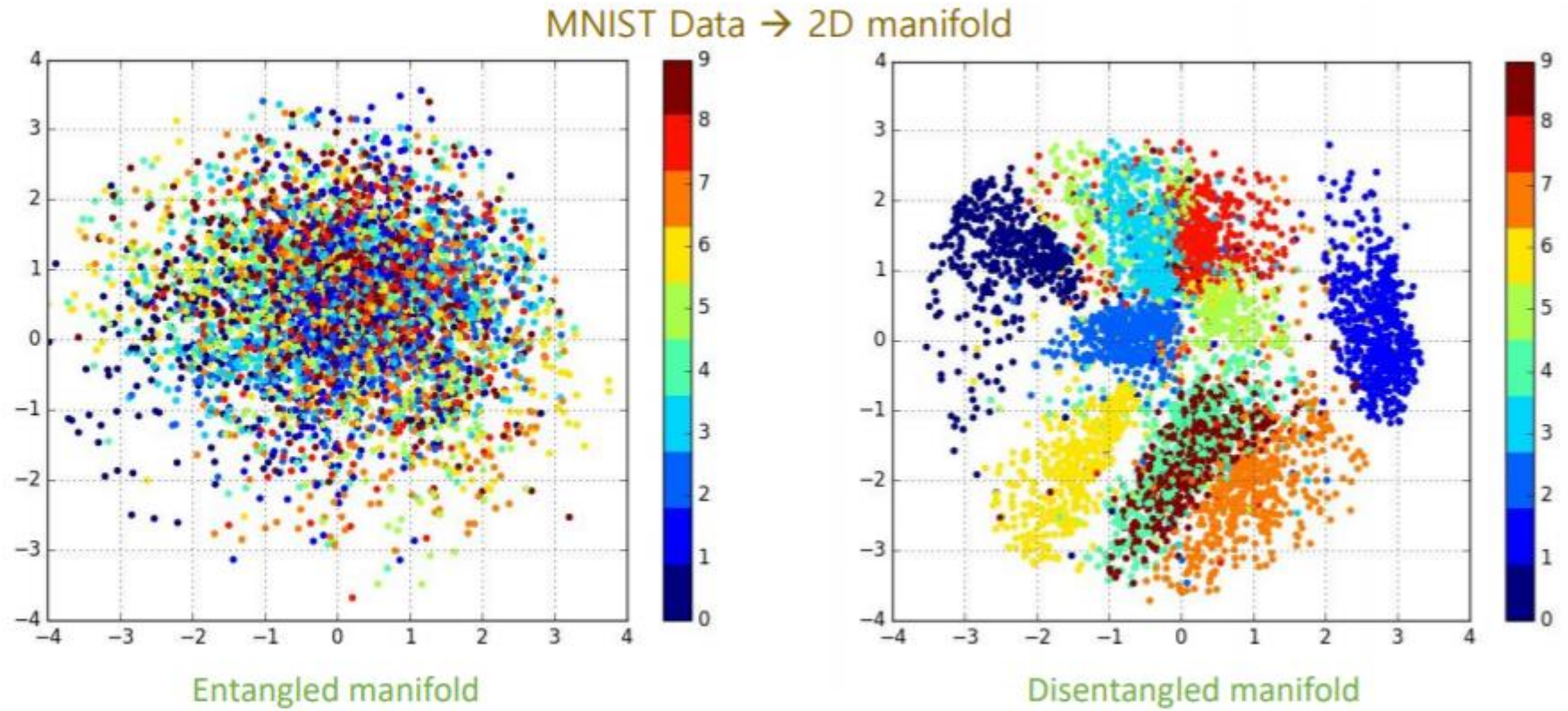
  -Reasonable distance metric



- # Manifold learning

-Disentangled manifold의 필요성

When a manifold is disentangled, it would be more interpretable and easier to apply to tasks



• Manifold learning

-Ways of Reducting Dimension

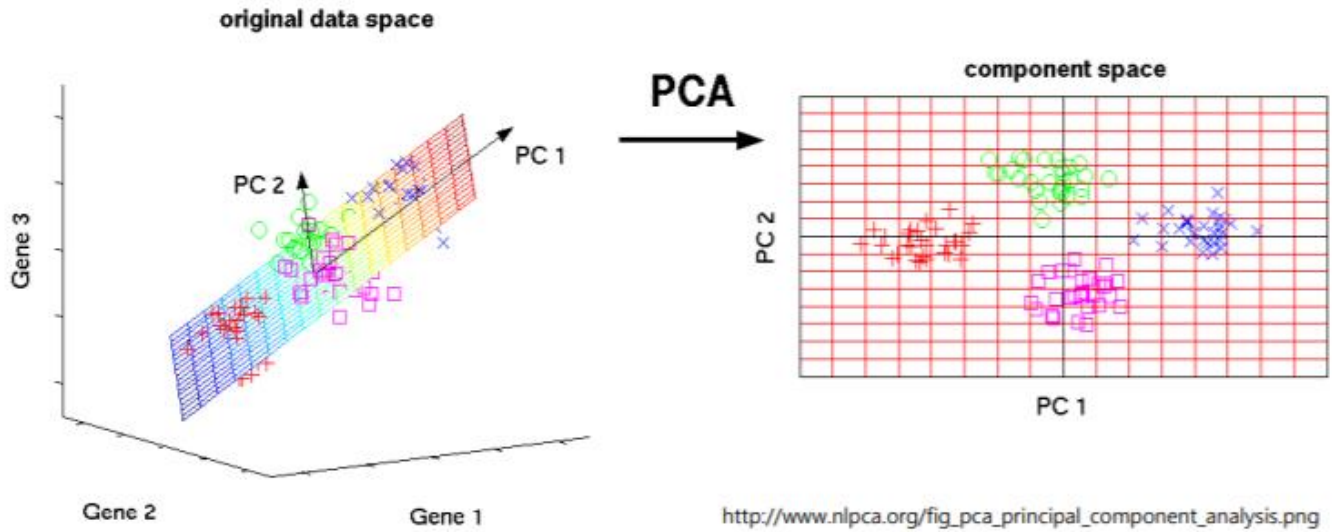
Dimensionality Reduction

- 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..

• Manifold learning

- PCA(Principal Component Analysis)

분산이 최대가 되는 manifold를 찾는다.



http://www.nlpca.org/fig_pca_principal_component_analysis.png

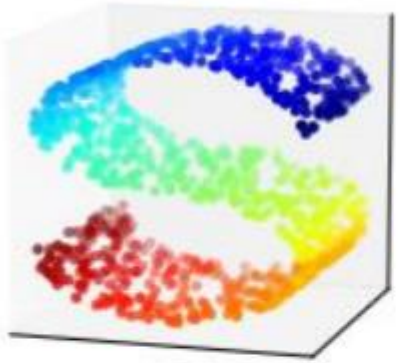
Wx + b 와 유사

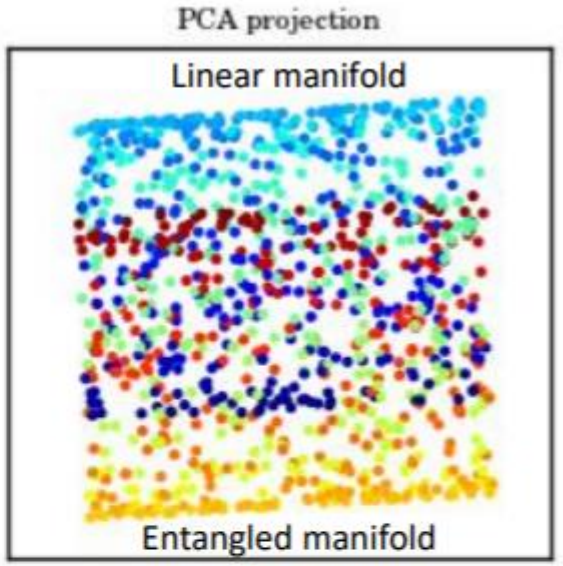
- Finds k directions in which data has highest variance
  - Principal directions (eigenvectors) W
- Projecting inputs x on these vectors yields reduced dimension representation (&decorrelated)
  - Principal components
  - $h = f_\theta(x) = W(x - \mu)\ with\ \theta = \{W, \mu\}$

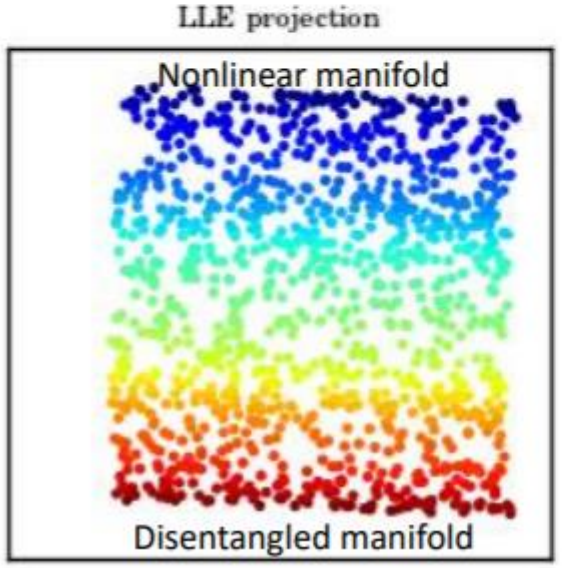
- Why mention PCA?
  - Prototypical unsupervised representation learning algorithm
  - Related to autoencoders
  - Prototypical manifold modeling algorithm

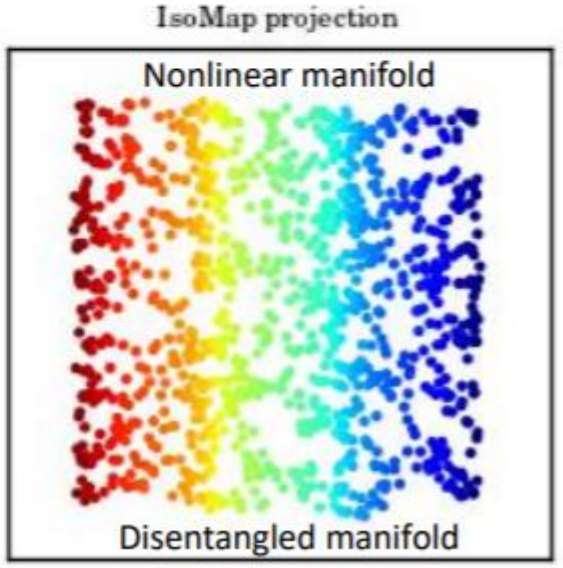
- # Manifold learning

-Manifold 비교









• Manifold learning

-고차원데이터에서의 Euclidian distance

- They explicitly use distance based neighborhoods.
- Training with k-nearest neighbors, or pairs of points.
- Typically Euclidean neighbors
- But in high d, your nearest Euclidean neighbor can be very different from you

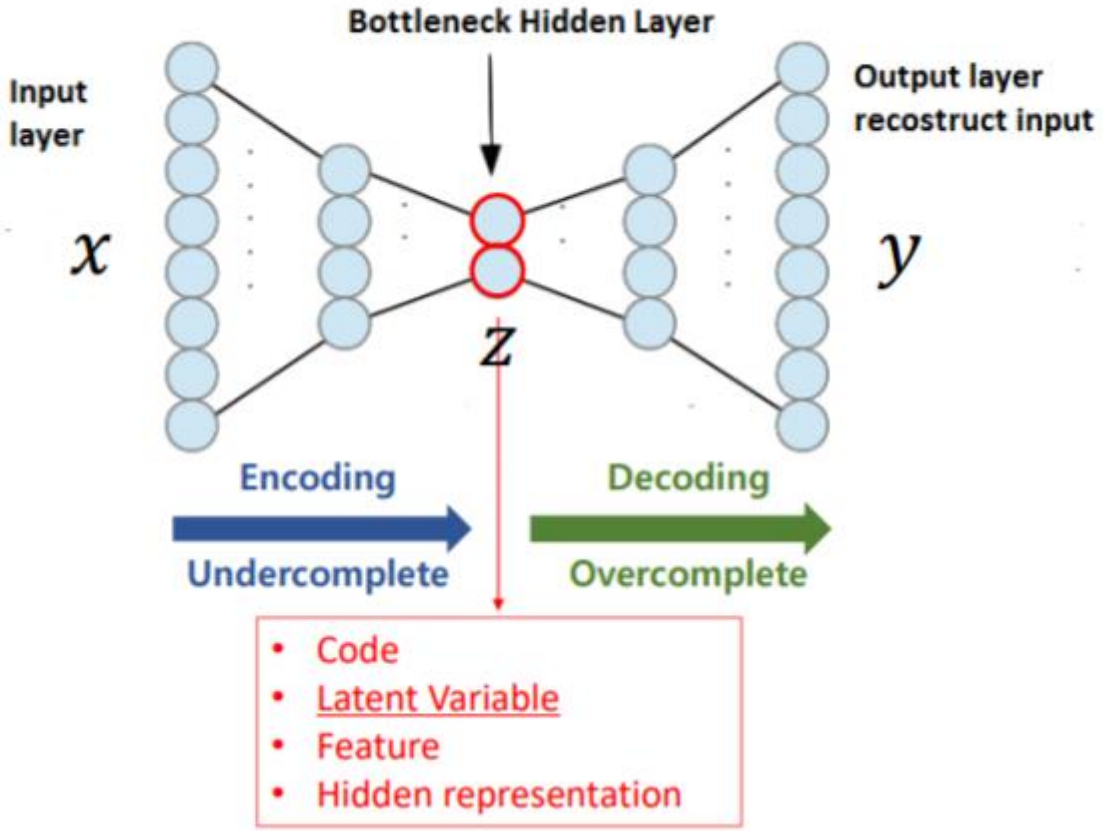
고차원 데이터 간의 유클리디안 거리는 유의미한 거리 개념이 아닐 가능성이 높다.

- **Auto Encoder**

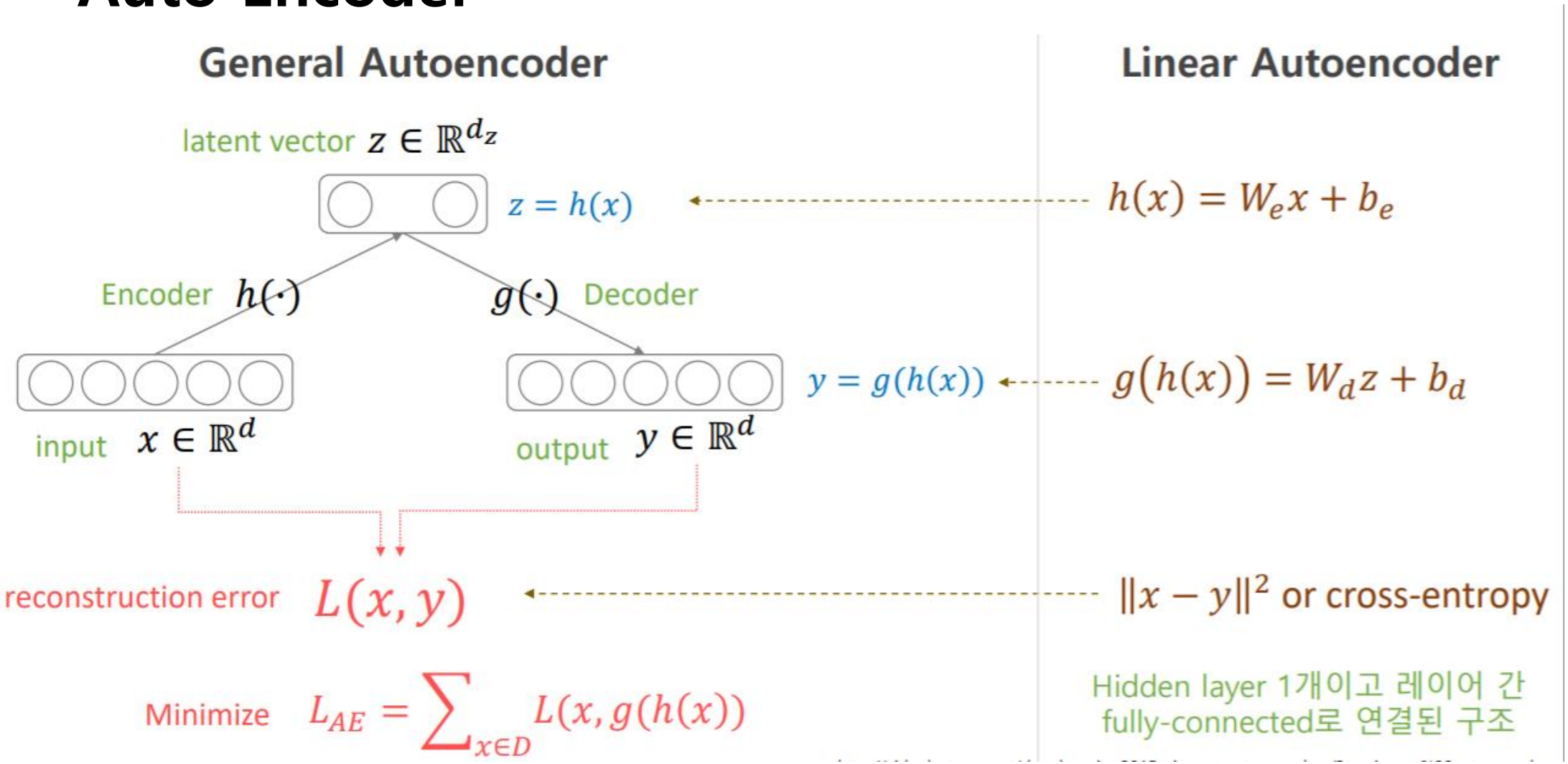
Autoencoders

= Auto-associators
= Diabolo networks
= Sandglass-shaped net

Diabolo

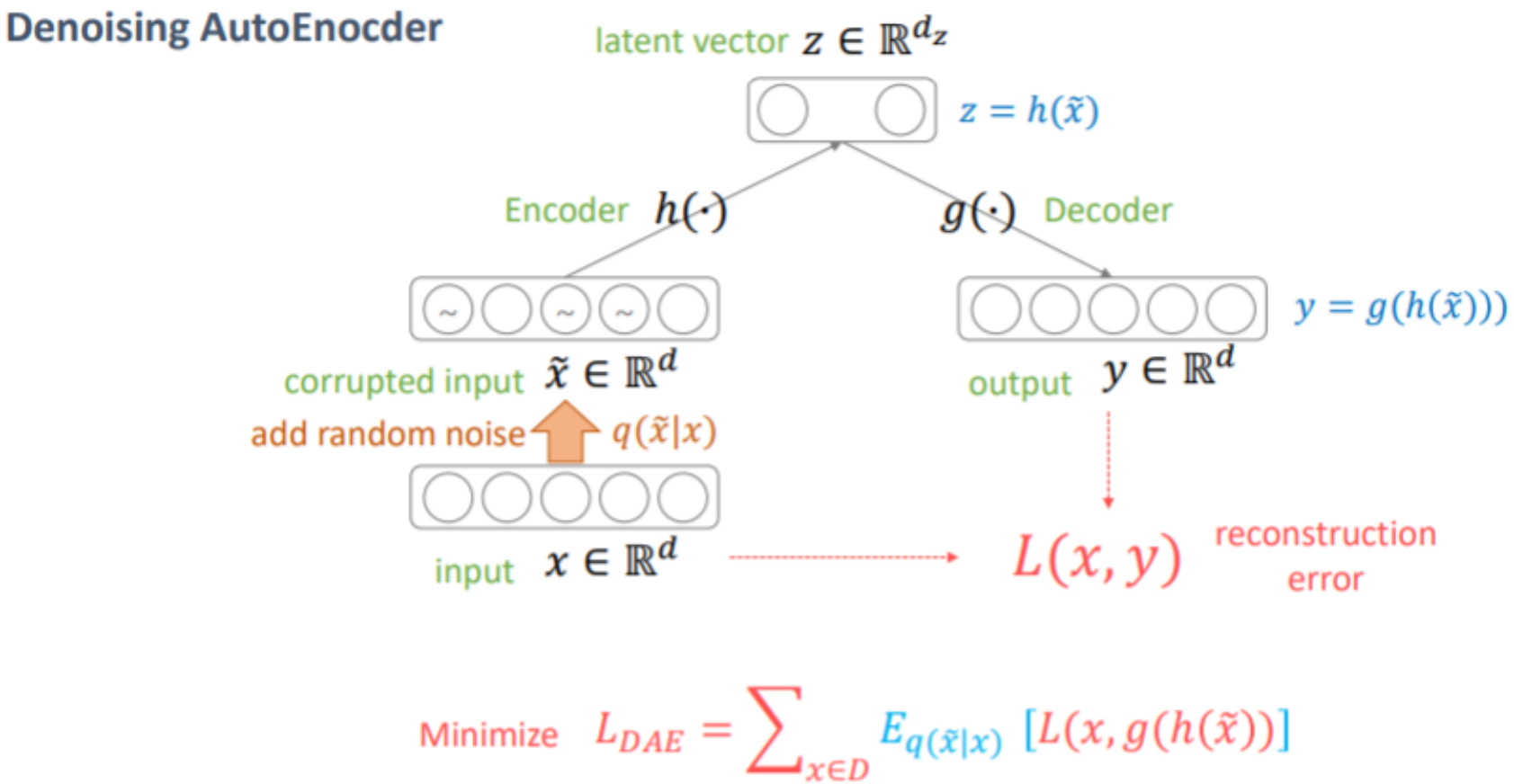


- **Auto Encoder**



- **Auto Encoder**

-Denoising AutoEncoder



- # Auto Encoder

-Denoising AutoEncoder

Denoising corrupted input

- will encourage representation that is robust to small perturbations of the input
- Yield similar or better classification performance as deep neural net pre-training

Possible corruptions

- Zeroing pixels at random (now called dropout noise)
- Additive Gaussian noise
- Salt-and-pepper noise
- Etc

Cannot compute expectation exactly

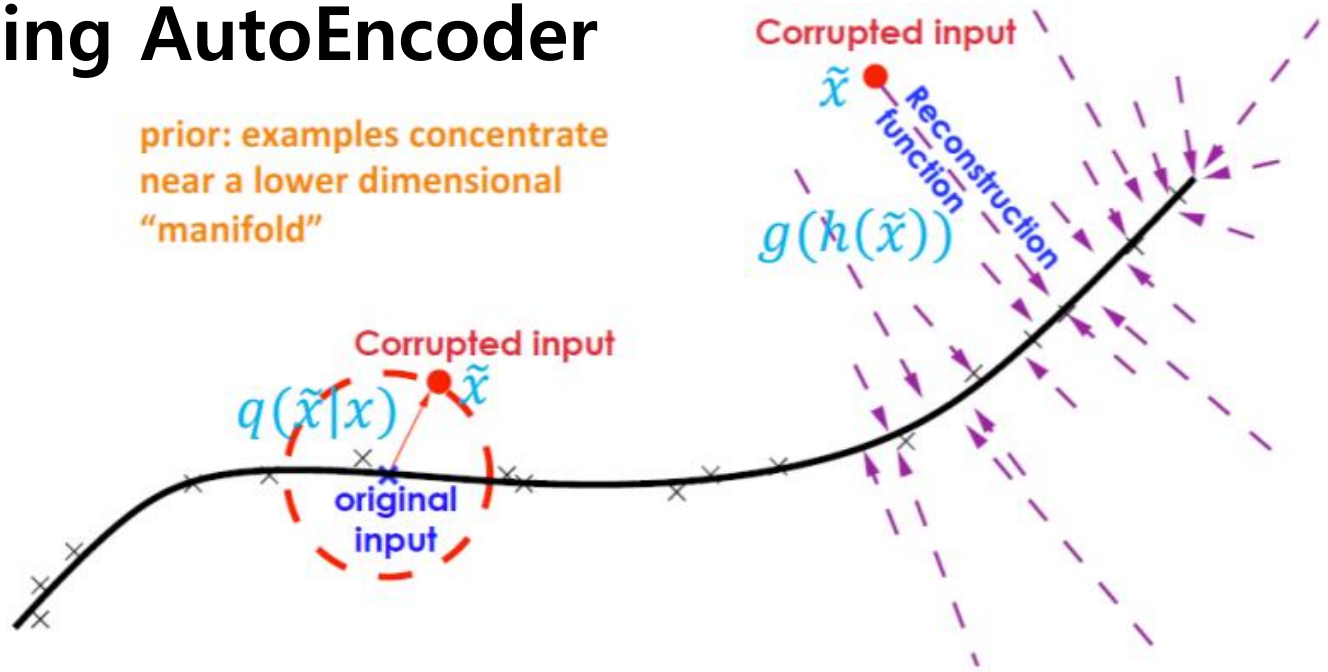
- Use sampling corrupted inputs

$$L_{DAE} = \sum_{x \in D} E_{q(\tilde{x}|x)} [L(x, g(h(\tilde{x}))] \approx \sum_{x \in D} \frac{1}{L} \sum_{i=1}^{L} L(x, g(h(\tilde{x}_i))$$

L개 샘플에 대한 평균으로 대체

- **Auto Encoder**

-Denoising AutoEncoder



- Suppose training data (x) concentrate near a low dimensional manifold.
- Corrupted examples (●) obtained by applying corruption process.
- $q(\tilde{x}|x)$ will generally lie farther from the manifold.
- The model learns with $p(x|\tilde{x})$ to "project them back" (via autoencoder $g(h(\tilde{x}))$) onto the manifold.
- Intermediate representation $z = h(x)$ may be interpreted as a coordinate system for points x on the manifold.

- # Auto Encoder

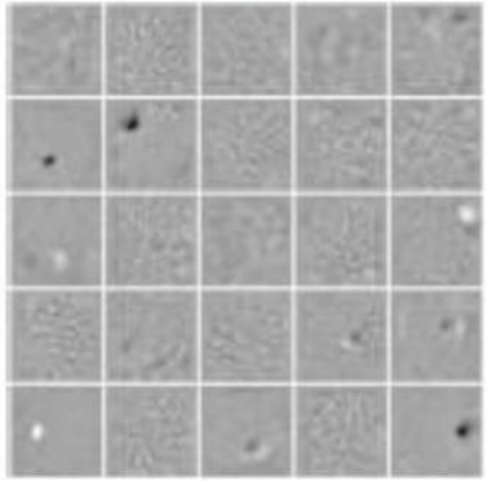
-Denoising AutoEncoder

MNIST digits (64x64 pixels)

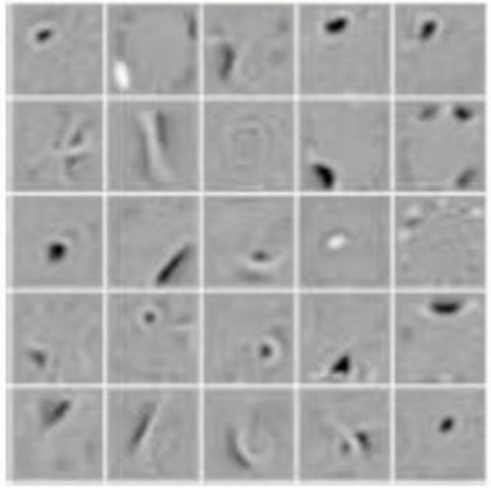
- Cross Entropy
- 100 hidden units
- Zero-masking noise

- Noise를 섞을수록 결과가 더 좋게 나옴

AE

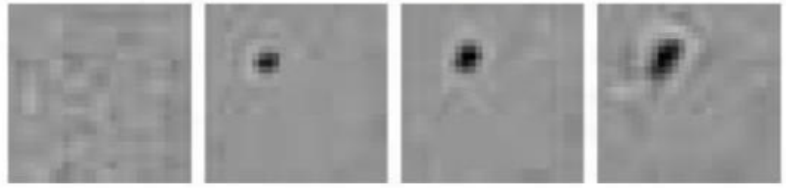


DAE



25% corruption

Increasing noise



(d) Neuron A (0%, 10%, 20%, 50% corruption)

(e) Neuron B (0%, 10%, 20%, 50% corruption)

- # Auto Encoder

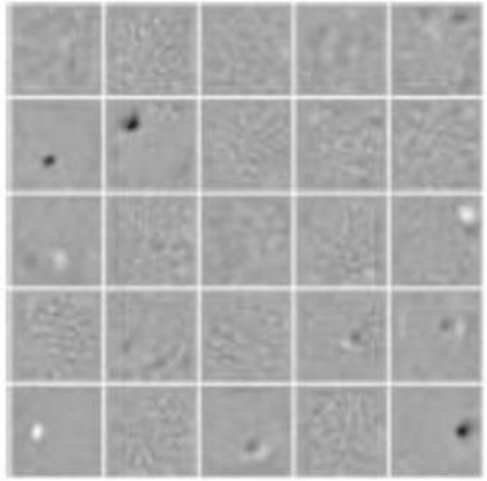
-Denoising AutoEncoder

MNIST digits (64x64 pixels)

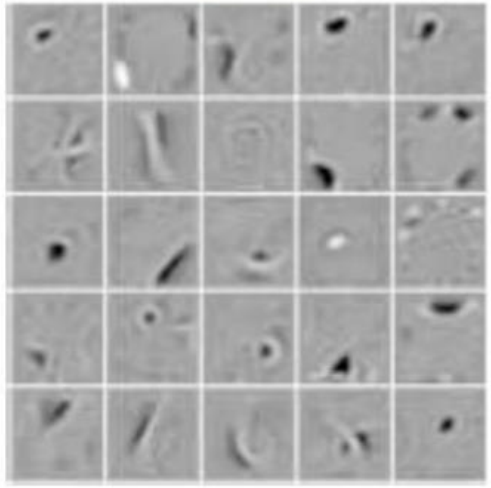
- Cross Entropy
- 100 hidden units
- Zero-masking noise

- Noise를 섞을수록 결과가 더 좋게 나옴

AE

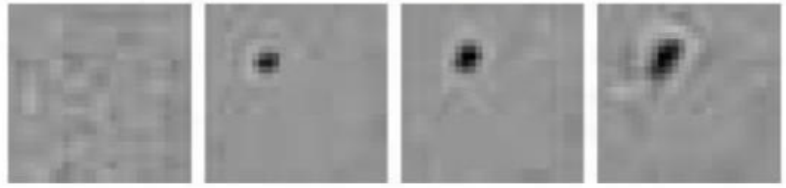


DAE



25% corruption

Increasing noise



(d) Neuron A (0%, 10%, 20%, 50% corruption)

(e) Neuron B (0%, 10%, 20%, 50% corruption)

- # Auto Encoder

-Contractive AutoEncoder

SCAE stochastic regularization term : $E_{q(\tilde{x}|x)}\ [\|h(x) - h(\tilde{x})\|^2]$

For small additive noise, $\tilde{x}|x = x + \epsilon, \epsilon \sim \mathcal{N}(0, \sigma^2 I)$

Taylor series expansion yields, $h(\tilde{x}) = h(x + \epsilon) = h(x) + \frac{\partial h}{\partial x}\epsilon + \cdots$

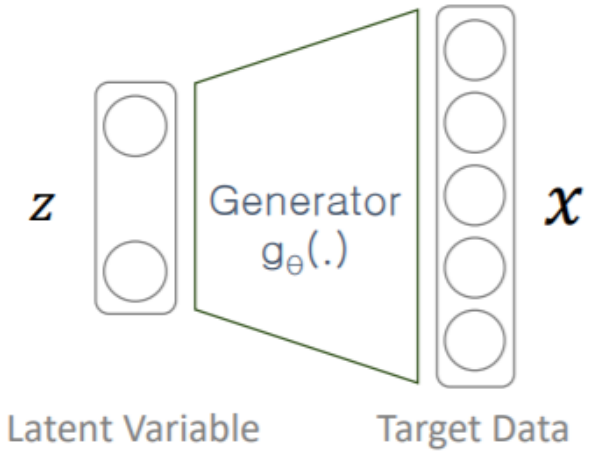
It can be showed that

$$\underbrace{E_{q(\tilde{x}|x)}\ [\|h(x) - h(\tilde{x})\|^2]}_{\text{Stochastic Regularization (SCAE)}} \approx \underbrace{\left\|\frac{\partial h}{\partial x}(x)\right\|_F^2}_{\text{Analytic Regularization (CAE)}}$$

Frobenius Norm

$$\|A\|_F^2 = \sum_{i=1}^{m}\sum_{j=1}^{n}|a_{ij}|^2$$

• Variational Auto Encoder



$z \sim p(z)$ Random variable

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

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

Latent variable can be seen as a set of control parameters for target data (generated data)

For MNIST example, our model can be trained to generate image which match a digit value z randomly sampled from the set [0, ..., 9].

그래서, p(z)는 샘플링 하기 용이해야 편하다.

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

We are aiming maximize the probability of each x in the training set, under the entire generative process, according to:

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

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

• Variational Auto Encoder

- 그냥 z에서 sampling하면 안되는 이유

$$p(x) \approx \sum_i p(x|g_\theta(z_i))p(z_i)$$

If $p(x|g_\theta(z)) = \mathcal{N}(x|g_\theta(z), \sigma^2 * I)$, the negative log probability of X is proportional squared Euclidean distance between $g_\theta(z)$ and x.

x : Figure 3(a)
$z_{bad} \rightarrow g_\theta(z_{bad})$: Figure 3(b)
$z_{bad} \rightarrow g_\theta(z_{good})$: Figure 3(c) (identical to x but shifted down and to the right by half a pixel)

$$\|x - z_{bad}\|^2 < \|x - z_{good}\|^2 \rightarrow p(x|g_\theta(z_{bad})) > p(x|g_\theta(z_{good}))$$

Solution 1: we should set the σ hyperparameter of our Gaussian distribution such that this kind of erroroneous digit does not contribute to p(X) → hard..
Solution 2: we would likely need to sample many thousands of digits from z_{good} → hard..

(a) (b) (c)

Figure 3: It's hard to measure the likelihood of images under a model using only sampling. Given an image X (a), the middle sample (b) is much closer in Euclidean distance than the one on the right (c). Because pixel distance is so different from perceptual distance, a sample needs to be extremely close in pixel distance to a datapoint X before it can be considered evidence that X is likely under the model.

생성기에 대한 확률모델을 가우시안으로 할 경우, MSE관점에서 가까운 것이 더 p(x)에 기여하는 바가 크다.
MSE가 더 작은 이미지가 의미적으로도 더 가까운 경우가 아닌 이미지들이 많기 때문에 현실적으로 올바른 확률값을 구하기가 어렵다.

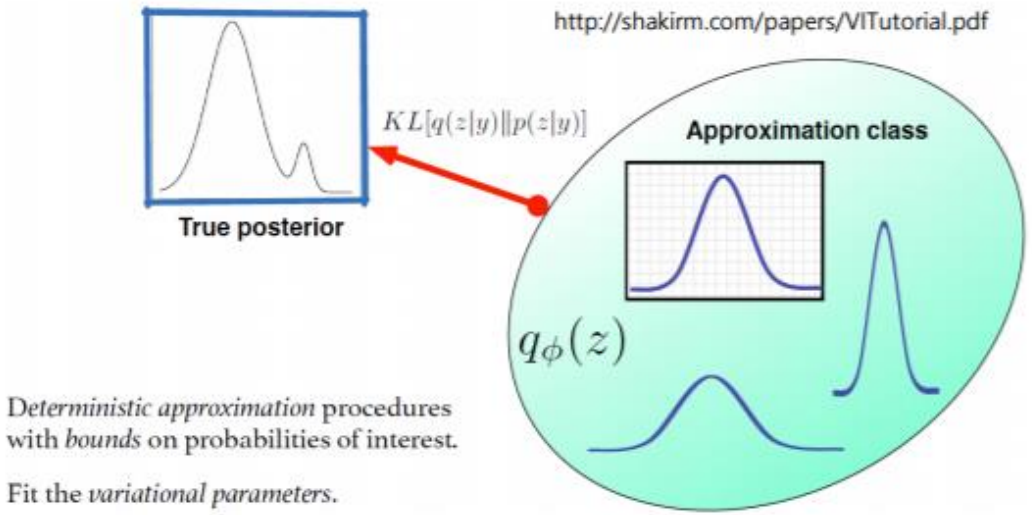
Tutorial on Variational Autoencoders : https://arxiv.org/pdf/1606.05908

- # Variational Auto Encoder

- 해결책: sampling $z\ from\ p(z|x)$ by variance inference

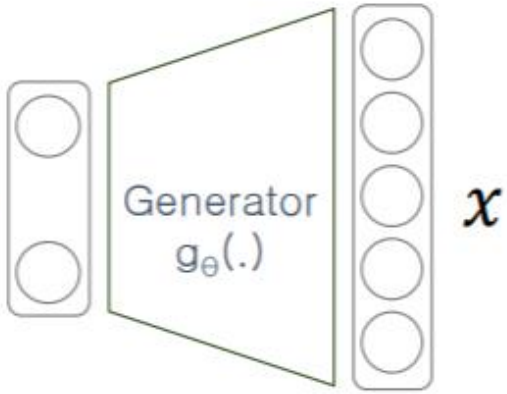
One possible solution : sampling z from $p(z|x)$

앞 슬라이드에서 Solution2가 가능하게 하는 방법 중 하나는 z를 정규분포에서 샘플링하는 것보다 x와 유의미하게 유사한 샘플이 나올 수 있는 확률분포 $p(z|x)$로 부터 샘플링하면 된다. 그러나 $p(z|x)$ 가 무엇인지 알지 못하므로, 우리가 알고 있는 확률분포 중 하나를 택해서 $(q_\phi(z|x))$ 그것의 파라미터값을 (λ) 조정하여 $p(z|x)$ 와 유사하게 만들어 본다. (Variational Inference)



[Variational Inference]

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

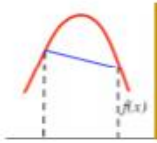


- # Variational Auto Encoder

-Derivation

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)

ELBO를 최대화하는 ϕ^*값을 찾으면 $\log(p(x)) = \mathbb{E}_{q_{\phi^*}(z|x)}[\log(p(x|z))] - KL(q_{\phi^*}(z|x)||p(z))$이다.

• Variational Auto Encoder

-Derivation

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

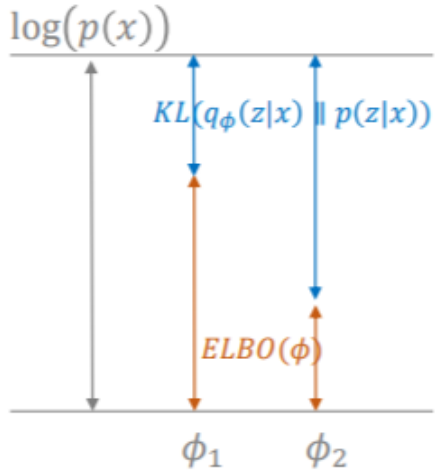
$$\log(p(x)) = \int \log(p(x)) q_\phi(z|x) dz \quad \leftarrow \int q_\phi(z|x)dz = 1$$

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

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

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

두 확률분포 간의 거리≥0

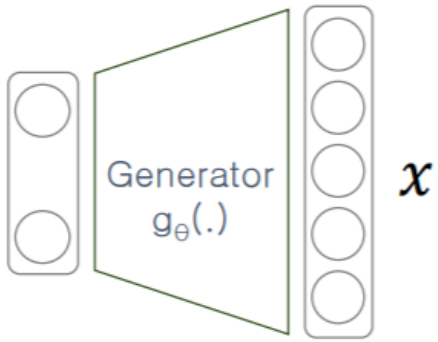


KL을 최소화하는 $q_\phi(z|x)$의 ϕ값을 찾으면 되는데 $p(z|x)$를 모르기 때문에,
KL최소화 대신에 ELBO를 최대화하는ϕ값을 찾는다.

- # Variational Auto Encoder

-Derivation

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



Optimization Problem 1 on ϕ: Variational Inference

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

Optimization Problem 2 on θ: Maximum likelihood

$$-\sum_i \log(p(x_i)) \leq -\sum_i \left\{\mathbb{E}_{q_\phi(z|x_i)}[\log(p(x_i|g_\theta(z)))] - KL(q_\phi(z|x_i)||p(z))\right\}$$

Final Optimization Problem

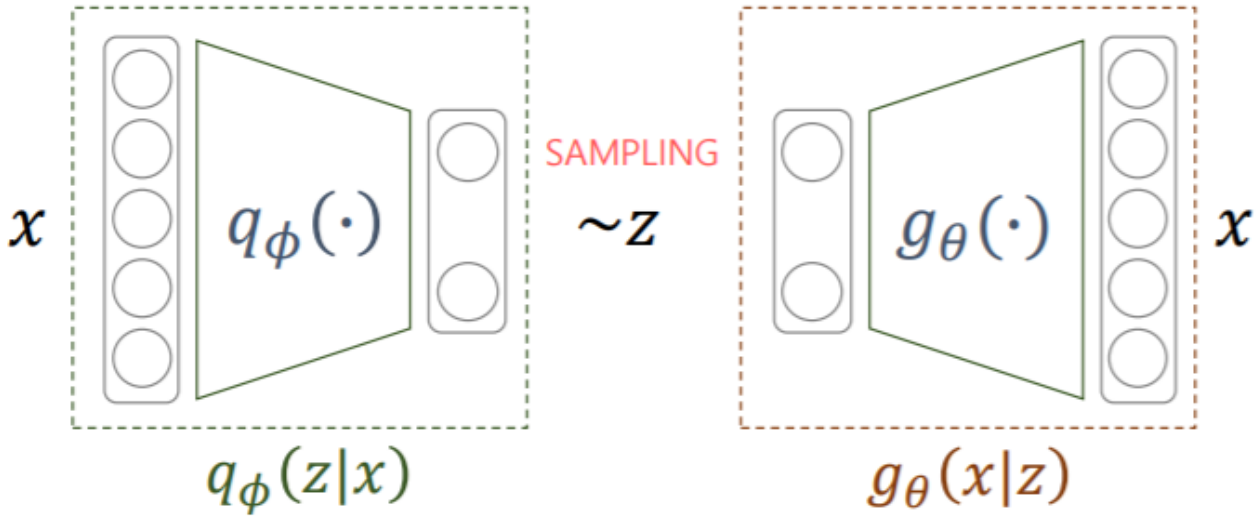
$$\arg\min_{\phi,\theta} \sum_i -\mathbb{E}_{q_\phi(z|x_i)}[\log(p(x_i|g_\theta(z)))] + KL(q_\phi(z|x_i)||p(z))$$

- # Variational Auto Encoder

-Derivation

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

Generation Network를 학습시키고 싶어서 Inference Network를 만들었더니 AutoEncoder와 모양이 비슷해진 꼴



The mathematical basis of VAEs actually has relatively little to do with classical autoencoders

- # Variational Auto Encoder

-Loss Function

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

원 데이터에 대한 likelihood

Variational inference를 위한 approximation class 중 선택

다루기 쉬운 확률 분포 중 선택

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

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

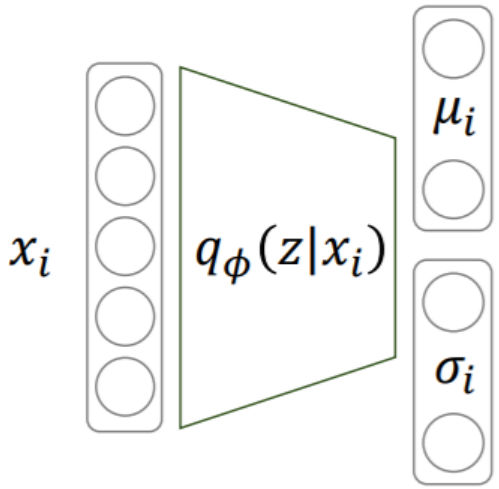
- 현재 샘플링 용 함수에 대한 추가 조건
- 샘플링의 용의성/생성 데이터에 대한 통제성을 위한 조건을 prior에 부여 하고 이와 유사해야 한다는 조건을 부여

• Variational Auto Encoder

-Regularization of Loss Function

Assumptions

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



Assumption 1

[Encoder : approximation class]
multivariate gaussian distribution with a diagonal covariance

$$q_\phi(z|x_i) \sim N(\mu_i, \sigma_i^2 I)$$

Assumption 2

[prior] multivariate normal distribution

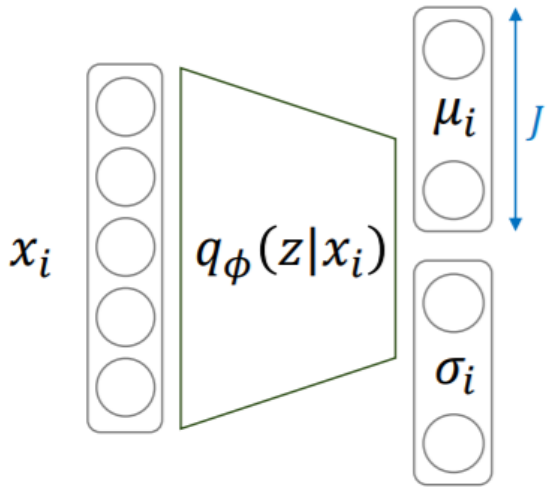
$$p(z) \sim N(0, I)$$

• Variational Auto Encoder

-Regularization of Loss Function

KL divergence

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



$$KL\big(q_\phi(z|x_i)||p(z)\big) = \frac{1}{2}\left\{tr(\sigma_i^2 I) + \mu_i^T\mu_i - J + \ln\frac{1}{\prod_{j=1}^{J}\sigma_{i,j}^2}\right\}$$

$$= \frac{1}{2}\left\{\sum_{j=1}^{J}\sigma_{i,j}^2 + \sum_{j=1}^{J}\mu_{i,j}^2 - J - \sum_{j=1}^{J}\ln(\sigma_{i,j}^2)\right\}$$

$$= \frac{1}{2}\sum_{j=1}^{J}(\mu_{i,j}^2 + \sigma_{i,j}^2 - \ln(\sigma_{i,j}^2) - 1)$$ Easy to compute!!

Kullback–Leibler divergence [edit]

The Kullback–Leibler divergence from $\mathcal{N}_0(\boldsymbol{\mu}_0, \boldsymbol{\Sigma}_0)$ to $\mathcal{N}_1(\boldsymbol{\mu}_1, \boldsymbol{\Sigma}_1)$, for non-singular matrices Σ_0 and Σ_1, is:[8]

$$D_{\mathrm{KL}}(\mathcal{N}_0\|\mathcal{N}_1) = \frac{1}{2}\left\{\operatorname{tr}\left(\boldsymbol{\Sigma}_1^{-1}\boldsymbol{\Sigma}_0\right) + (\boldsymbol{\mu}_1-\boldsymbol{\mu}_0)^{\mathrm{T}}\boldsymbol{\Sigma}_1^{-1}(\boldsymbol{\mu}_1-\boldsymbol{\mu}_0) - k + \ln\frac{|\boldsymbol{\Sigma}_1|}{|\boldsymbol{\Sigma}_0|}\right\},$$

(posterior: $\mathcal{N}_0$, prior: $\mathcal{N}_1$)

where k is the dimension of the vector space.

• Variational Auto Encoder

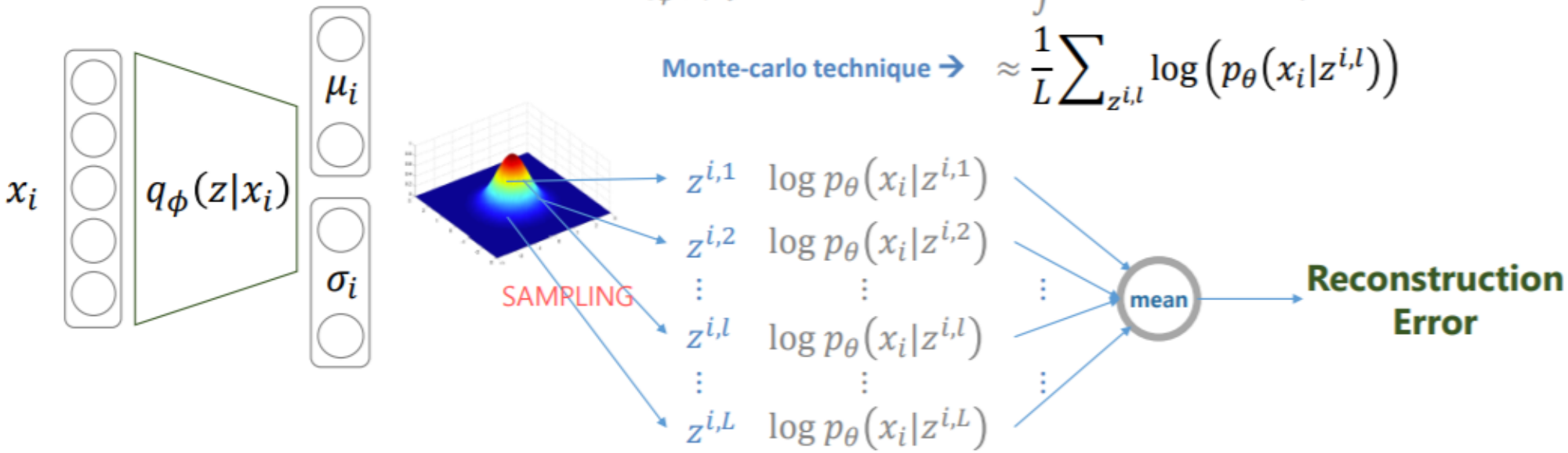
-Reconstruction of Loss Function

Sampling

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

$$\mathbb{E}_{q_\phi(z|x_i)}[\log(p_\theta(x_i|z))] = \int \log(p_\theta(x_i|z))q_\phi(z|x_i)dz$$

Monte-carlo technique → $$\approx \frac{1}{L}\sum_{z^{i,l}} \log\left(p_\theta(x_i|z^{i,l})\right)$$



- L is the number of samples for latent vector
- Usually L is set to 1 for convenience

https://home.zhaw.ch/~dueo/bbs/files/vae.pdf

• Variational Auto Encoder

-Reparametrization Trick

$$-E_{q_\phi(z|x)}[\log\{p(x|z)\}] + D_{KL}(q_\phi(z|x) \| p(z))$$

원래 z를 $q_\emptyset$ 로부터 sample 하므로 random해서 미분불가 -> reparametrization tirck 이용

x가 주어졌을 때 z의 확률의 [z가 주어졌을 때 x의 확률의 로그 값] 평균 ▸ Reconstruction Error

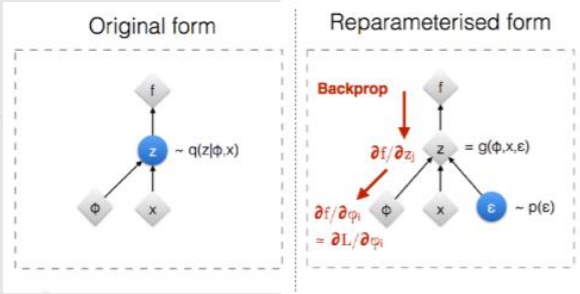
$$-E_{q_\phi(z|x)}[\log\{p(x|z)\}] = -\int q_\phi(z|x)\log\{p(x|z)\} \approx -\frac{1}{L}\sum_l^L \log\{p(x|z_l)\} \text{ where } z_l \sim q_\phi(z|x)$$

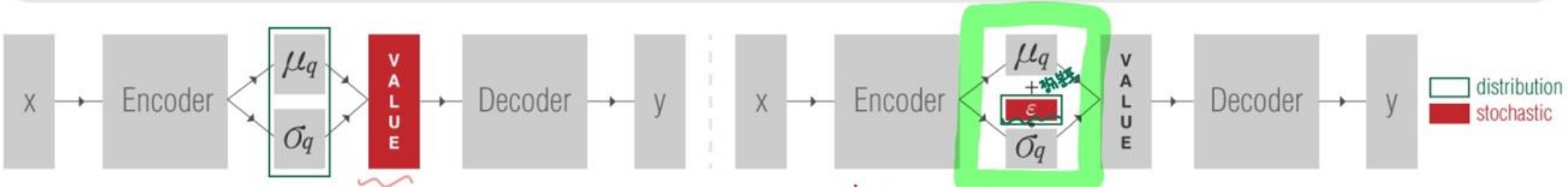
Monte Carlo Estimation

BUT when using the MonteCralo technique, we can't differentiate so that it is stochastic.

For that reason, we use Reparametrization Trick; 'z' is replaced with '$\mu_q + \sigma_q \varepsilon$'

$$-E_{q_\phi(z|x)}[\log\{p(x|z)\}] = -E_{\varepsilon \sim N(0,1)}[\log\{p(x|\mu_q + \sigma_q \varepsilon)\}]$$





미분불가

미분가능

• Variational Auto Encoder

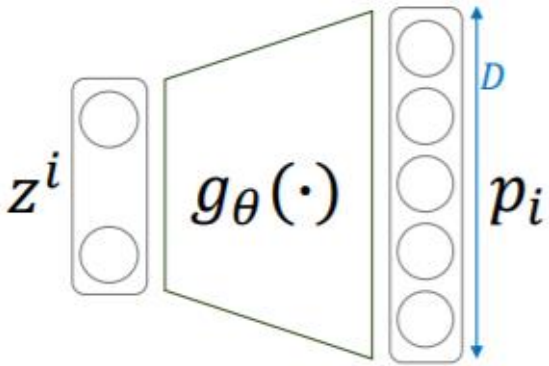
-Reconstruction Error Assumption

Assumption

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

$$\mathbb{E}_{q_\phi(z|x_i)}[\log(p_\theta(x_i|z))] = \int \log(p_\theta(x_i|z))q_\phi(z|x_i)dz \approx \frac{1}{L}\sum_{z^{i,l}} \log\left(p_\theta(x_i|z^{i,l})\right) \approx \log\left(p_\theta(x_i|z^i)\right)$$

Monte-carlo technique (under the $\approx$) — L=1 (under the last $\approx$)



$p_\theta(x_i|z^i) \sim Bernoulli(p_i)$

Assumption 3-1

[Decoder, likelihood]

multivariate bernoulli or gaussain distribution

$$\log\left(p_\theta(x_i|z^i)\right) = \log\prod_{j=1}^{D} p_\theta(x_{i,j}|z^i) = \sum_{j=1}^{D} \log p_\theta(x_{i,j}|z^i)$$

$$= \sum_{j=1}^{D} \log p_{i,j}^{x_{i,j}}(1-p_{i,j})^{1-x_{i,j}} \quad \leftarrow p_{i,j} \triangleq network\ output$$

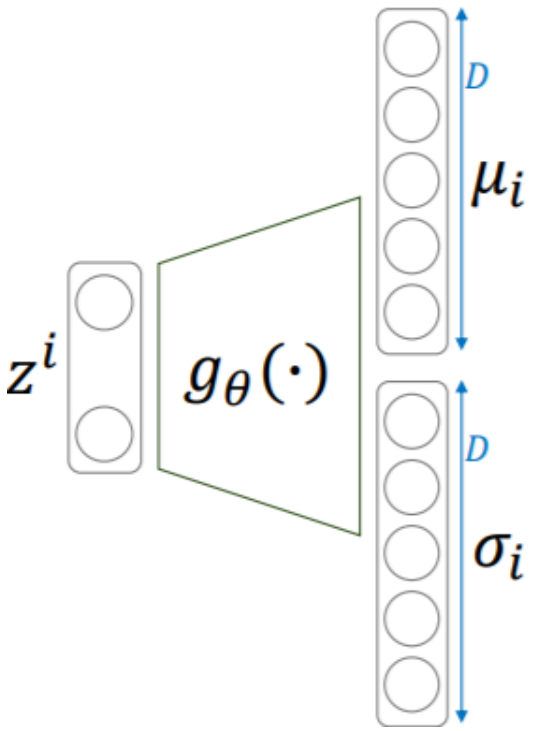
$$= \sum_{j=1}^{D} x_{i,j}\log p_{i,j} + (1-x_{i,j})\log(1-p_{i,j}) \quad \leftarrow \text{Cross entropy}$$

- # Variational Auto Encoder

-Reconstruction Error Assumption

Assumption

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



$$\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p_\theta(x_i|z)\right)\right] \approx \log\left(p_\theta\left(x_i|z^i\right)\right)$$

Assumption 3-2

[Decoder, likelihood]
multivariate bernoulli or gaussain distribution

$$\log\left(p_\theta\left(x_i|z^i\right)\right) = \log\left(N(x_i;\mu_i,\sigma_i^2 I)\right)$$

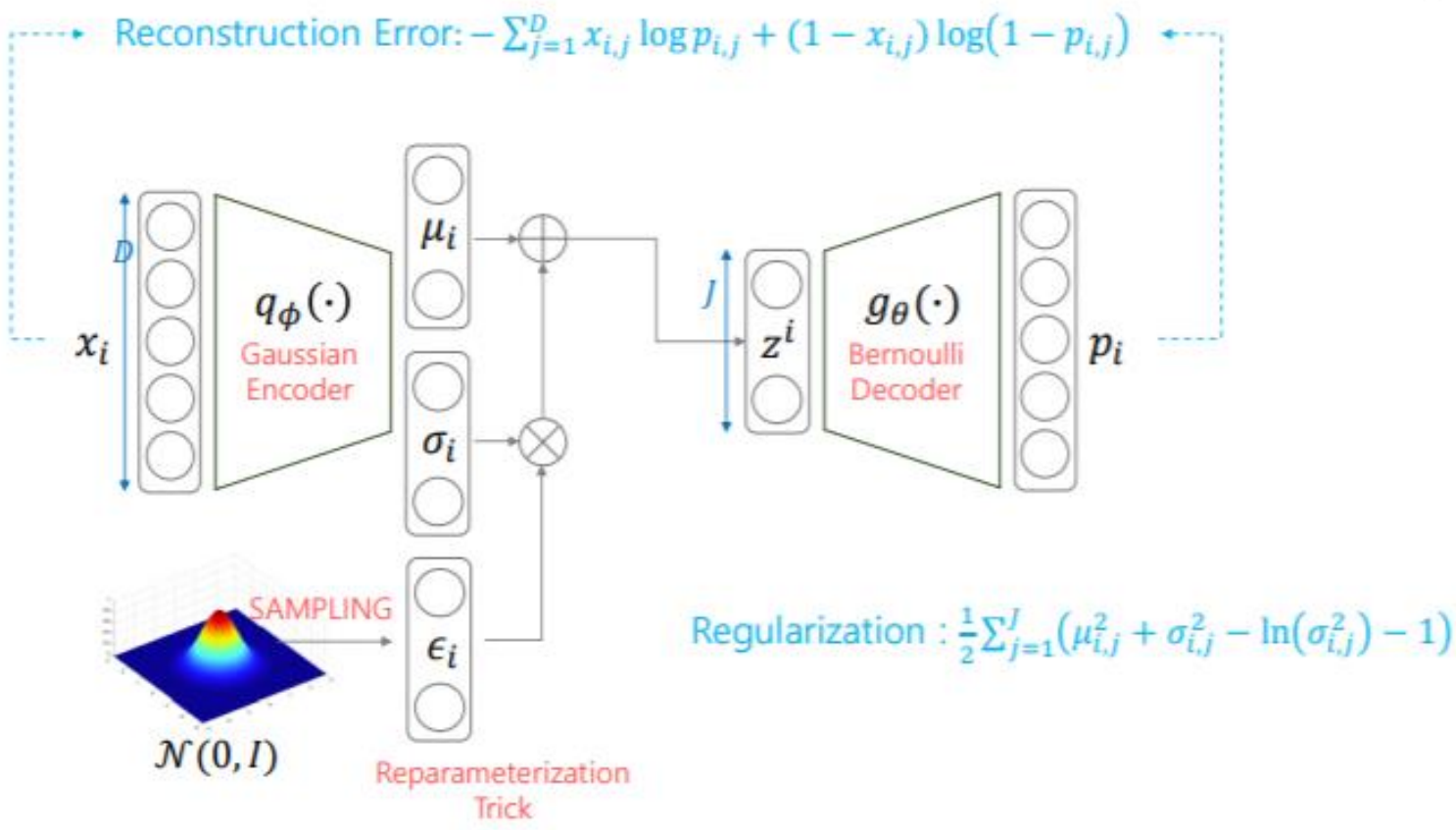
$$= -\sum_{j=1}^{D}\frac{1}{2}\log(\sigma_{i,j}^2) + \frac{(x_{i,j}-\mu_{i,j})^2}{2\sigma_{i,j}^2}$$

For gaussain distribution with identity covariance

$$\log\left(p_\theta\left(x_i|z^i\right)\right) \propto -\sum_{j=1}^{D}(x_{i,j}-\mu_{i,j})^2 \quad \leftarrow \text{Squared Error}$$

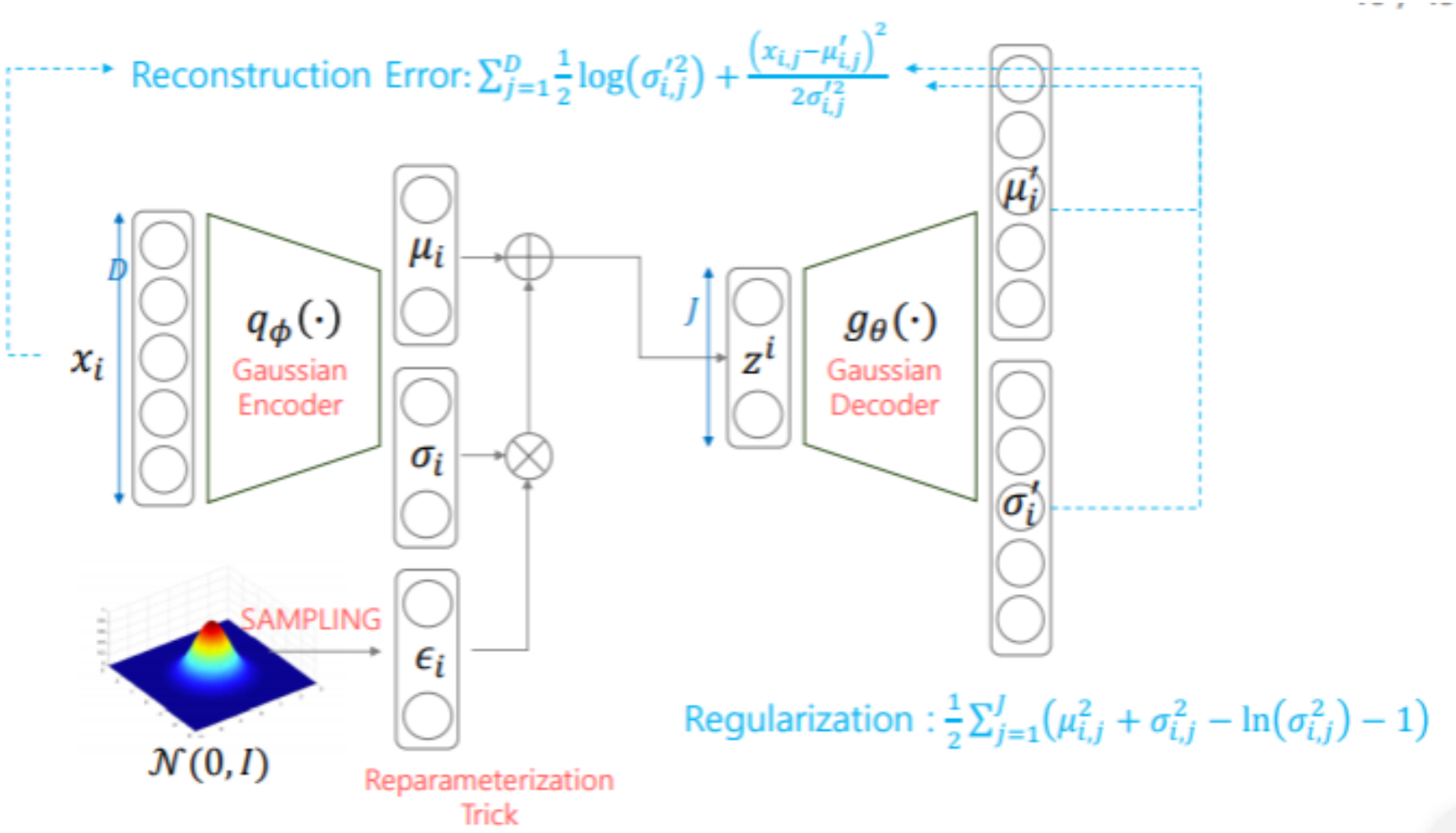
- # Variational Auto Encoder

-Structure (Gaussian Encoder + Bernoulli Decoder)



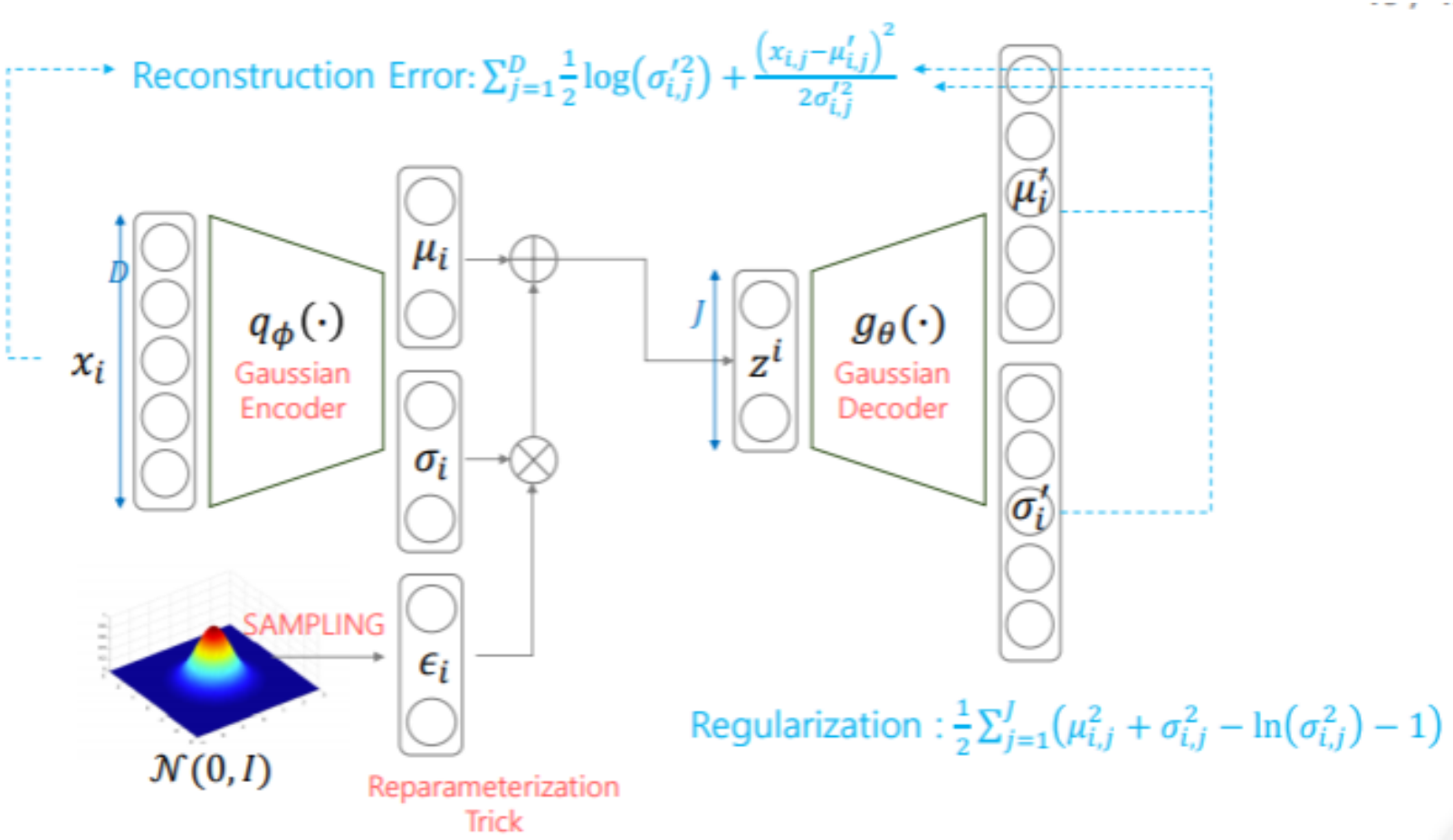
- # Variational Auto Encoder

-Structure (Gaussian Encoder + Gaussian Decoder)



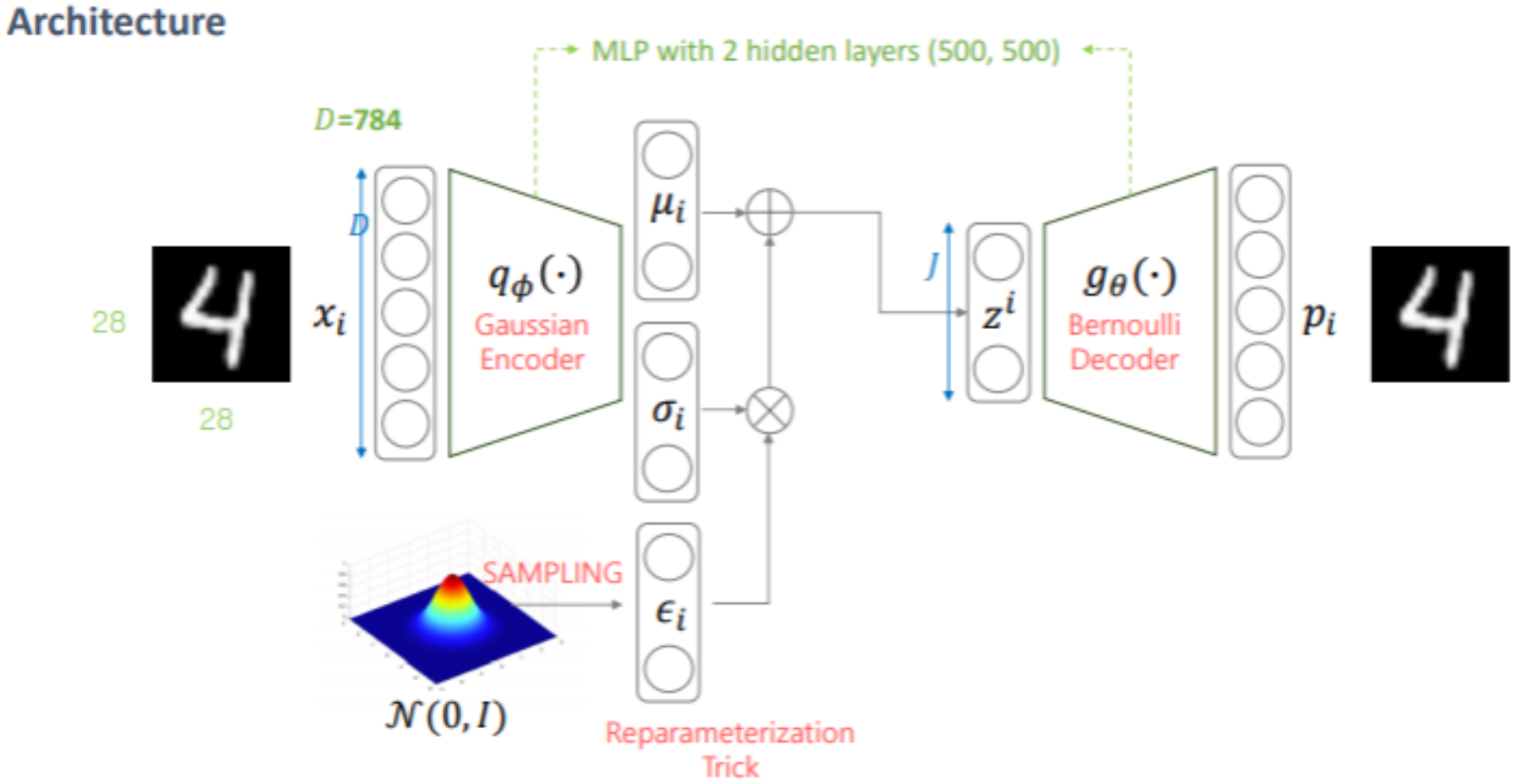
- # Variational Auto Encoder

-Structure (Gaussian Encoder + Gaussian Decoder with Identity Covariance)



- # Variational Auto Encoder

-Result (MNIST)



- # Variational Auto Encoder

-Reproduce (MNIST)

Dimension 클수록 효과좋다

Input image

J = |z| =2

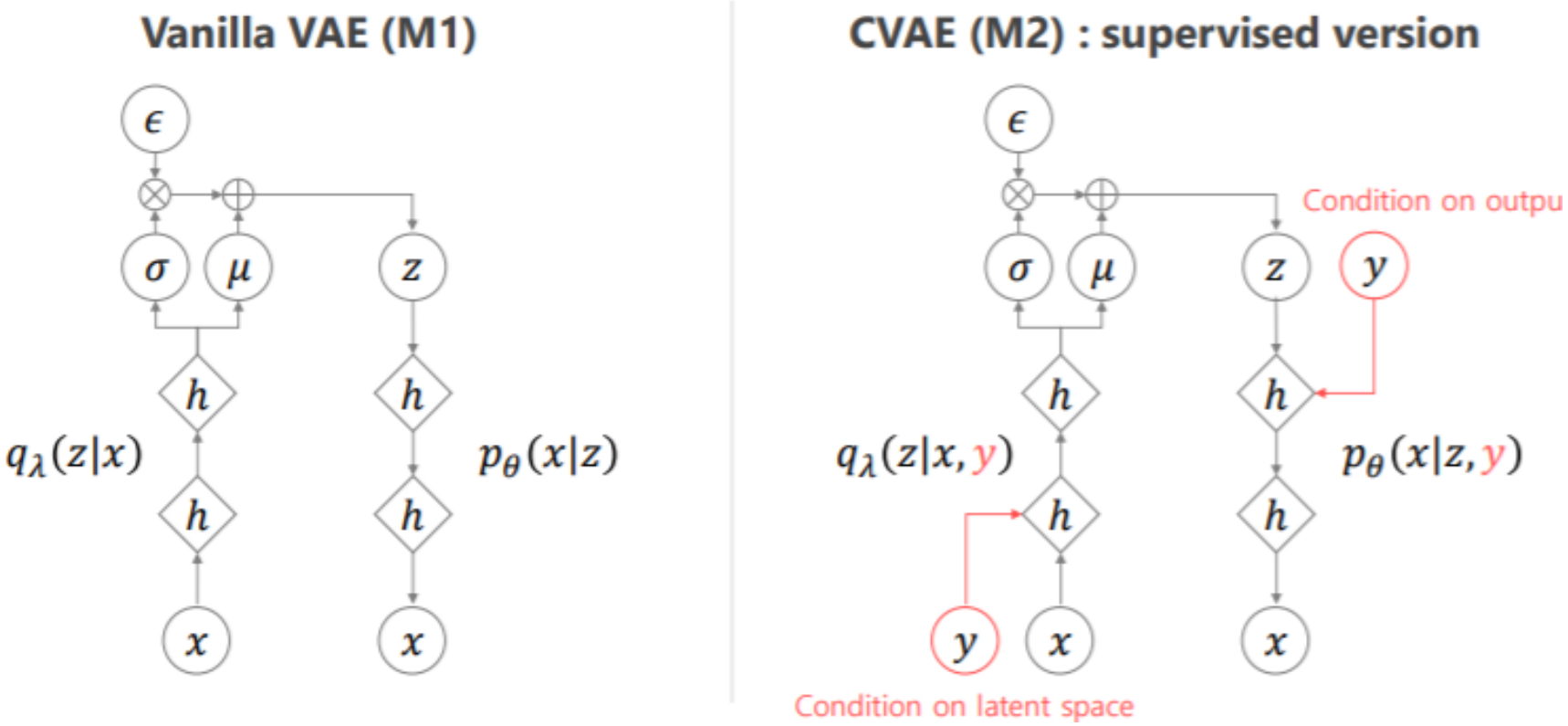
J = |z| =5

J = |z| =20

- # Variational Auto Encoder

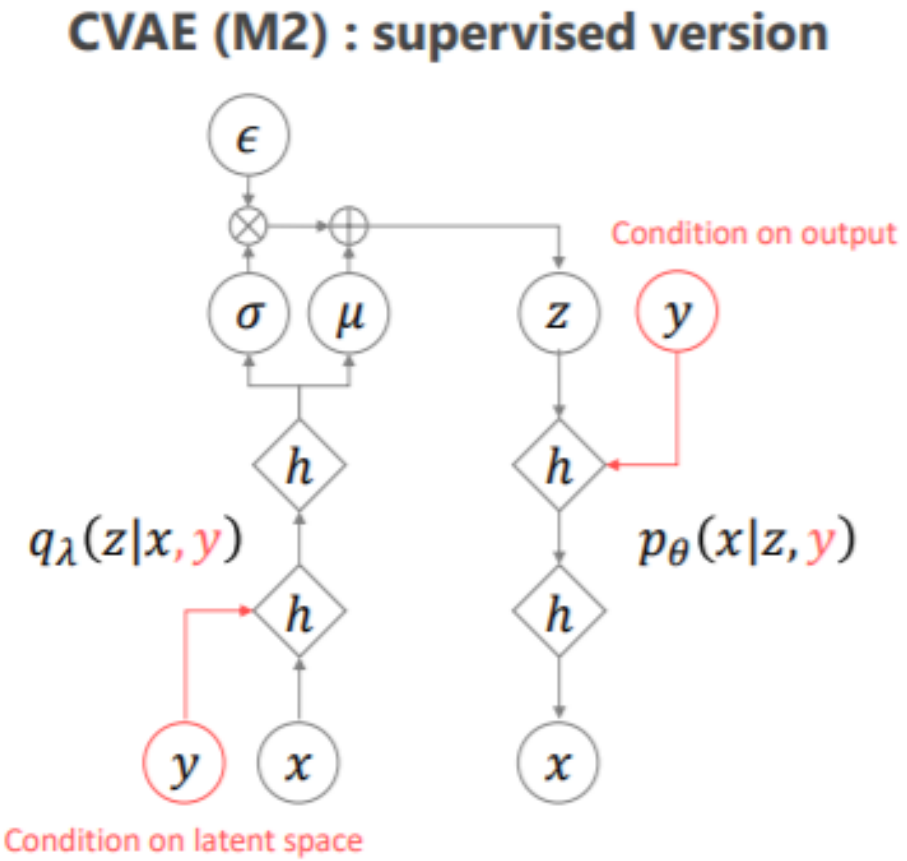
- CVAE (Conditional VAE) (supervised)

Conditional VAE



- # Variational Auto Encoder

- CVAE (Conditional VAE) (supervised)



$$\log(p_\theta(x,y)) = \log \int p_\theta(x,y|z)\frac{p(z)}{q_\phi(z|x,y)}q_\phi(z|x,y)dz$$

$$\geq \int \log\left(p_\theta(x,y|z)\frac{p(z)}{q_\phi(z|x,y)}\right)q_\phi(z|x,y)dz$$

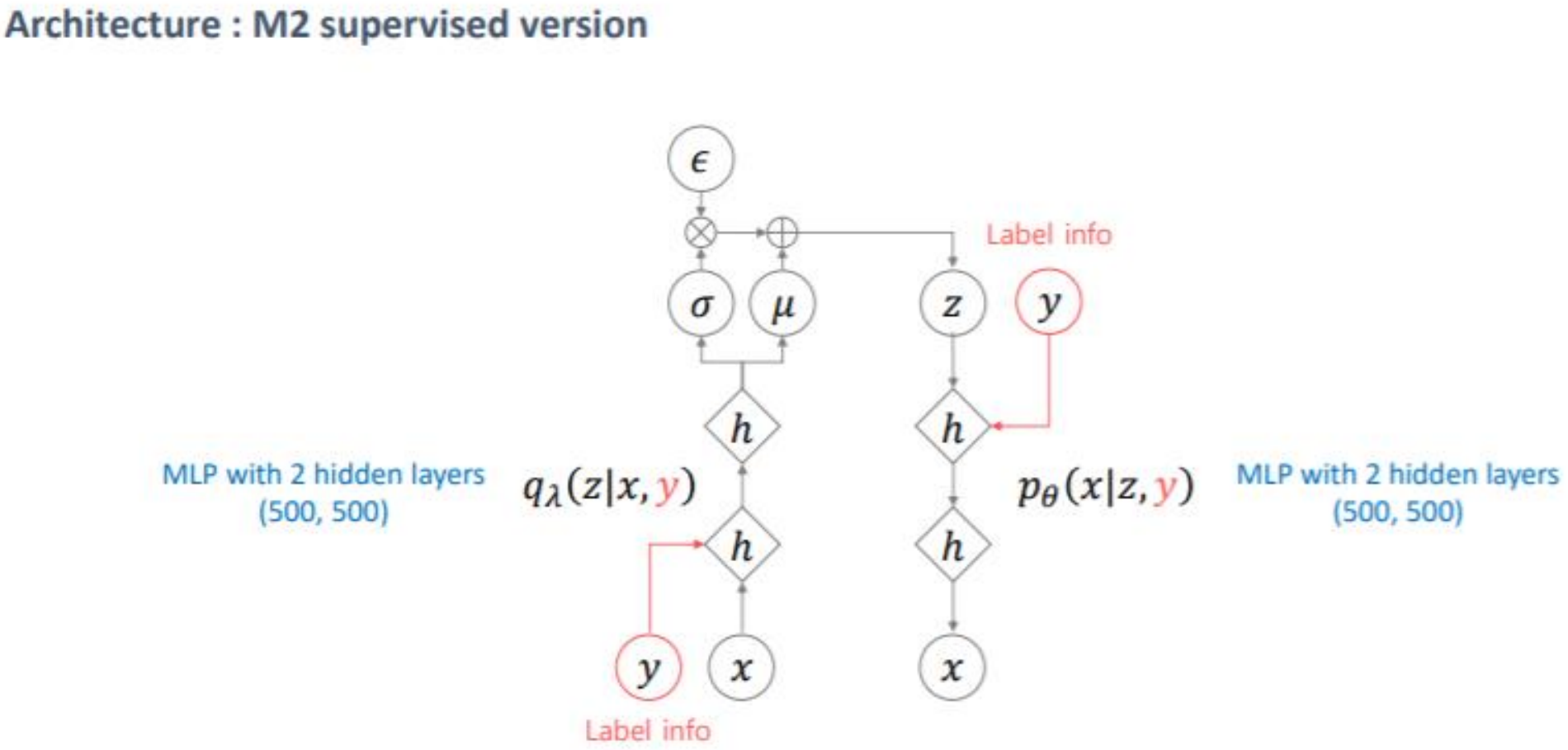
$$= \int \log\left(p_\theta(x|y,z)\frac{p(y)p(z)}{q_\phi(z|x,y)}\right)q_\phi(z|x,y)dz$$

$$= \mathbb{E}_{q_\phi(z|x,y)}\left[\log(p_\theta(x|y,z)) + \log(p(y))\right.$$

$$= -\mathcal{L}(x,y)$$ ELBO!!

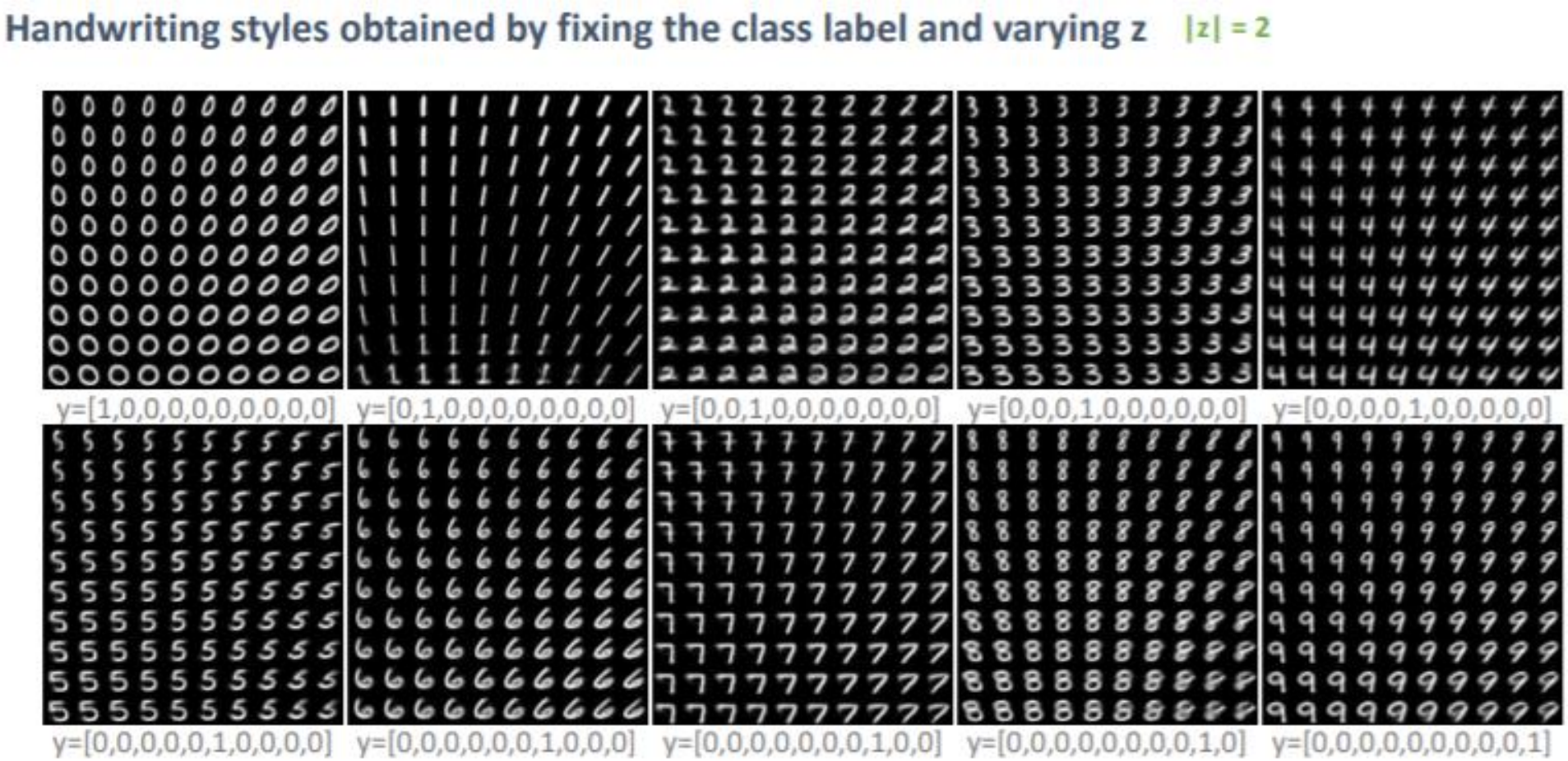
- # Variational Auto Encoder

- CVAE (Conditional VAE) (supervised)



- # Variational Auto Encoder

- CVAE (Conditional VAE) (supervised)



- # Variational Auto Encoder

- CVAE (Conditional VAE) (supervised)

Analogie Classification : Result in paper

Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 (± 0.95)	11.82 (± 0.25)	11.97 (± 1.71)	**3.33** (± 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 (± 0.049)	4.94 (± 0.13)	**2.59** (± 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 (± 0.12)	4.24 (± 0.07)	3.60 (± 0.56)	**2.40** (± 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 (± 0.04)	3.92 (± 0.63)	**2.18** (± 0.04)

Label 100개만 썼을때 (49900개는 label 없이 image만)

N = 100일 때 직접 돌려보니, 0.9514 → 4.86

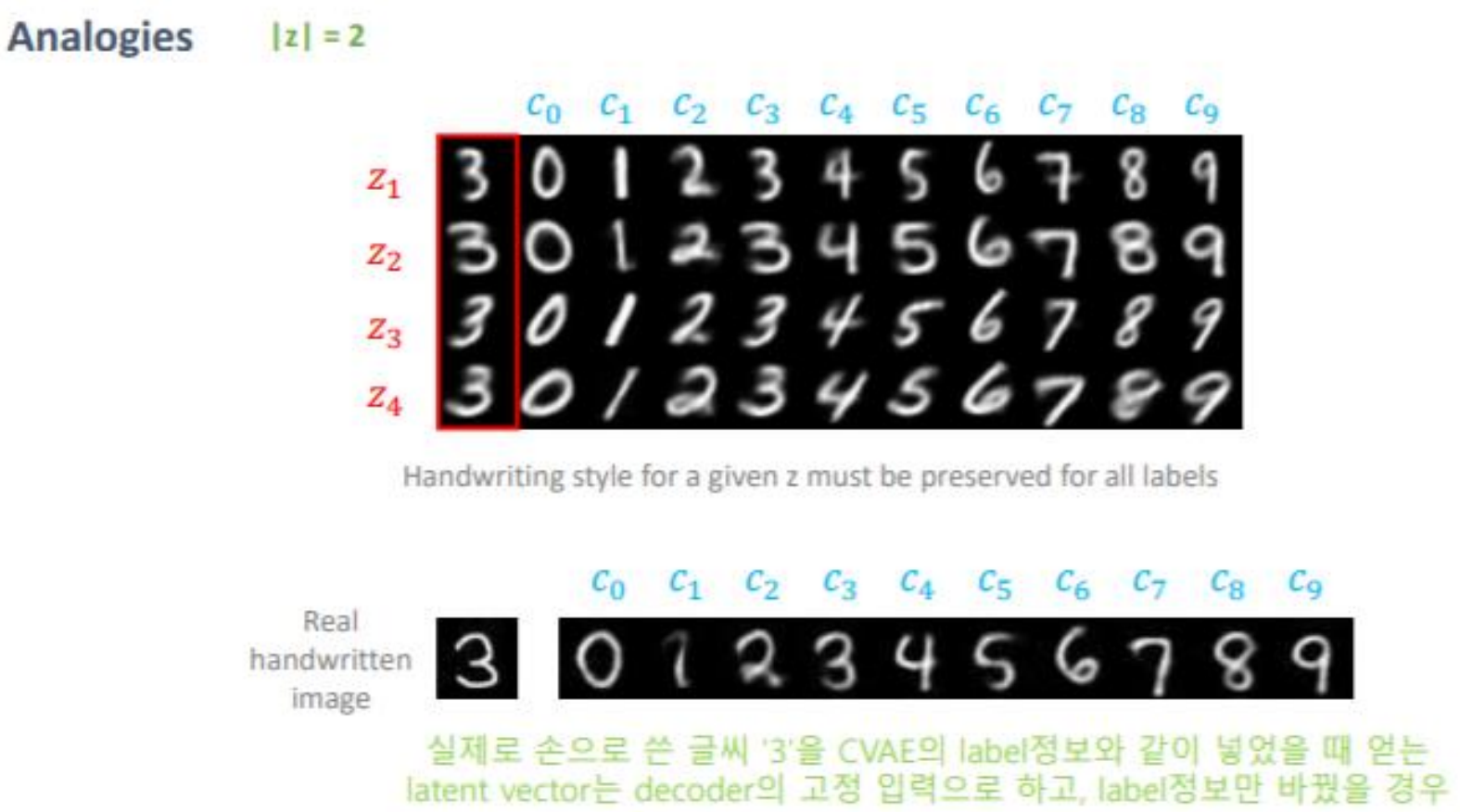
(총 50000개 중, 100개만 레이블 사용, 49900개는 미사용)

https://github.com/saemundsson/semisupervised_vae

Z-sampling

- # Variational Auto Encoder

- CVAE (Conditional VAE) (supervised)



- # Variational Auto Encoder

- CVAE (Conditional VAE) (supervised)

Classification : Result in paper

Table 1: Benchmark results of semi-supervised classification on MNIST with few labels.

N	NN	CNN	TSVM	CAE	MTC	AtlasRBF	M1+TSVM	M2	M1+M2
100	25.81	22.98	16.81	13.47	12.03	8.10 ($\pm$ 0.95)	11.82 ($\pm$ 0.25)	11.97 ($\pm$ 1.71)	**3.33** ($\pm$ 0.14)
600	11.44	7.68	6.16	6.3	5.13	–	5.72 ($\pm$ 0.049)	4.94 ($\pm$ 0.13)	**2.59** ($\pm$ 0.05)
1000	10.7	6.45	5.38	4.77	3.64	3.68 ($\pm$ 0.12)	4.24 ($\pm$ 0.07)	3.60 ($\pm$ 0.56)	**2.40** ($\pm$ 0.02)
3000	6.04	3.35	3.45	3.22	2.57	–	3.49 ($\pm$ 0.04)	3.92 ($\pm$ 0.63)	**2.18** ($\pm$ 0.04)

Label 100개만 썼을때 (49900개는 label 없이 image만)

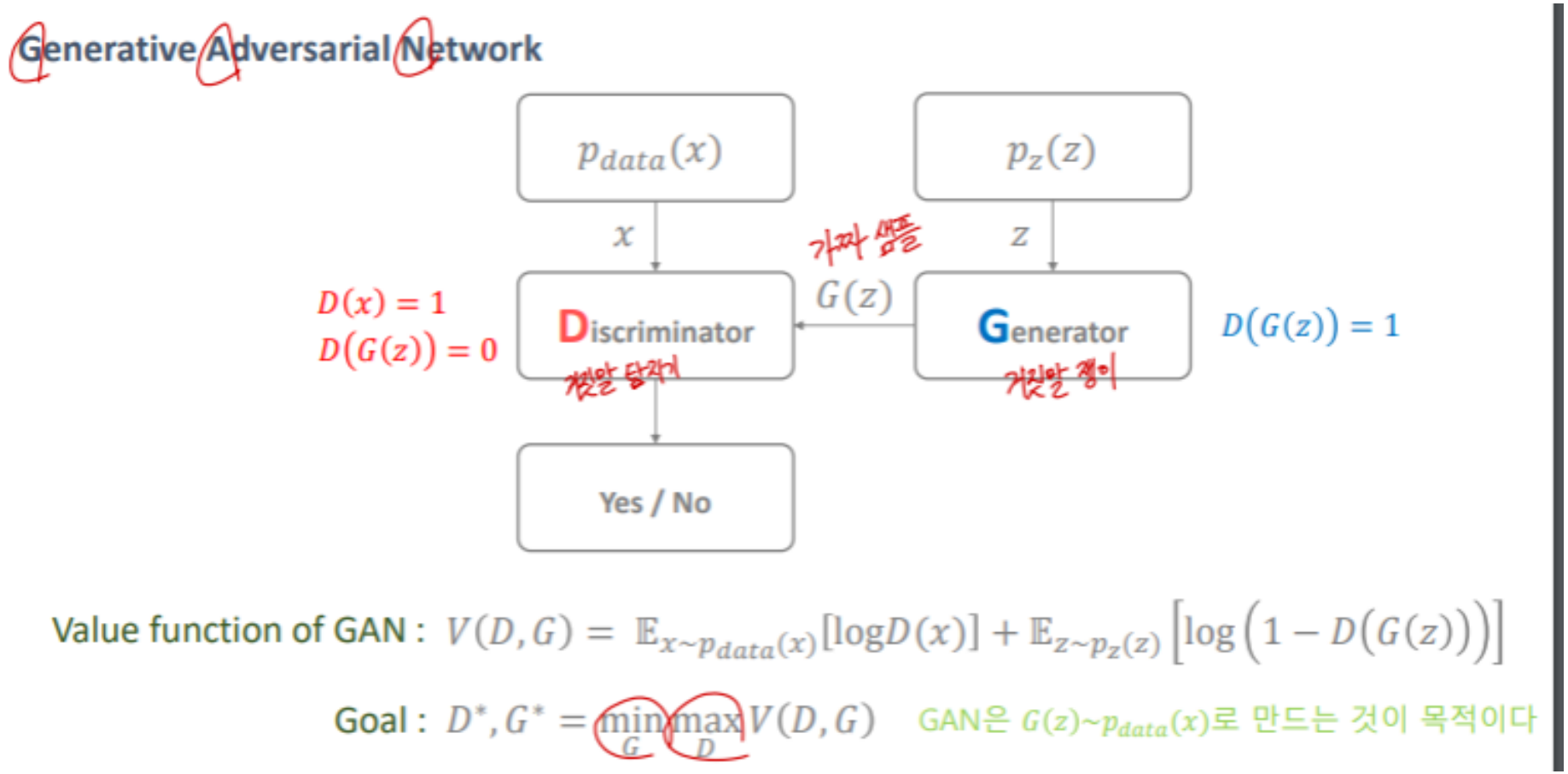
N = 100일 때 직접 돌려보니, 0.9514 → 4.86

(총 50000개 중, 100개만 레이블 사용, 49900개는 미사용)

https://github.com/saemundsson/semisupervised_vae

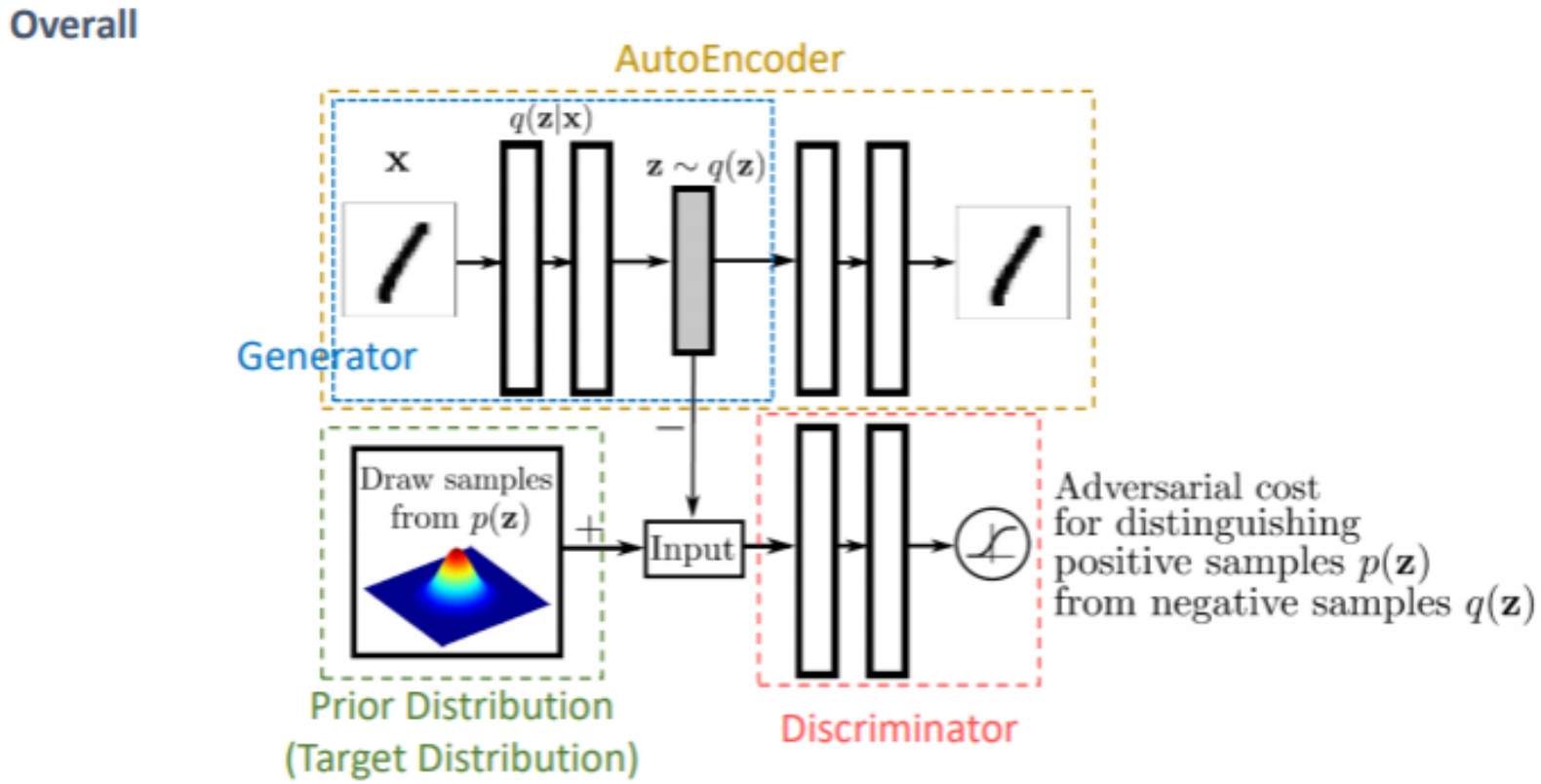
• Variational Auto Encoder

- GAN (Generative Adversarial Network)



- # Variational Auto Encoder

- GAN (Generative Adversarial Network)



- # Variational Auto Encoder

- GAN (Generative Adversarial Network)

Loss Function

KL Loss 대신 사용한다.

GAN loss $V(D,G) = \mathbb{E}_{z\sim p(z)}[\log D(z)] + \mathbb{E}_{x\sim p(x)}\left[\log\left(1 - D(q_\phi(x))\right)\right]$

Let's say G is defined by $q_\phi(\cdot)$ and D is defined by $d_\lambda(\cdot)$

$$V_i(\phi,\lambda,x_i,z_i) = \log d_\lambda(z_i) + \log\left(1 - d_\lambda(q_\phi(x_i))\right)$$

*논문에는 로스 정의가 제시되어 있지 않아 새로 정리한 내용

VAE loss $L_i(\phi,\theta,x_i) = -\mathbb{E}_{q_\phi(z|x_i)}[\log(p_\theta(x_i|z))] + KL(q_\phi(z|x_i)||p(z))$

• Variational Auto Encoder

- GAN Training Procedure

For drawn samples x_i from training data set, z_i from prior distribution $p(z)$

Training Step 1 : Update AE
update ϕ, θ according to reconstruction error

$$L_i(\phi, \theta, x_i) = -\mathbb{E}_{q_\phi(z|x_i)}[\log(p_\theta(x_i|z))]$$

번갈아가면서 학습.

Training Step 2 : Update Discriminator
update λ according to loss for discriminator

$$-V_i(\phi, \lambda, x_i, z_i) = -\log d_\lambda(z_i) - \log\left(1 - d_\lambda(q_\phi(x_i))\right)$$

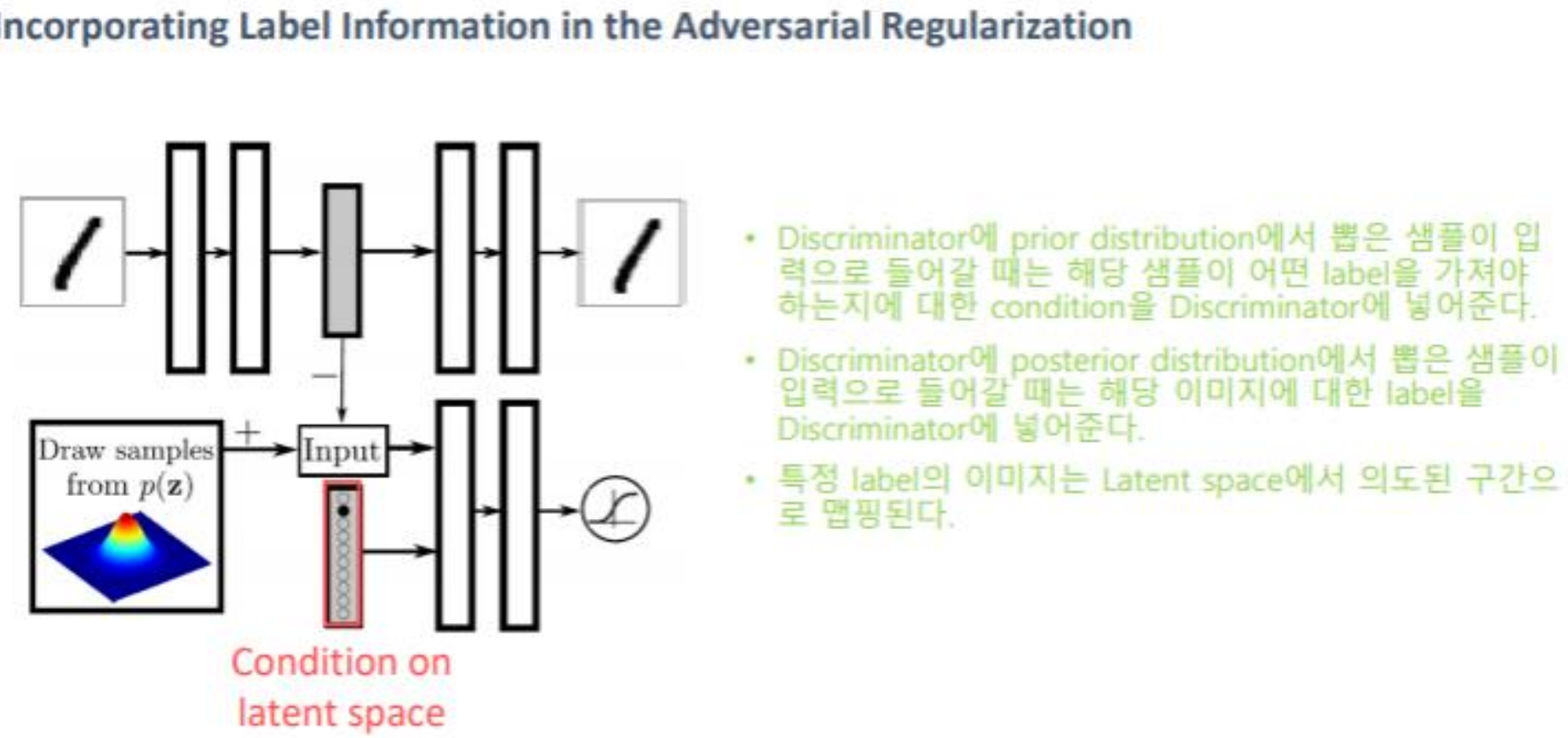
Training Step 3 : Update Generator
update ϕ according to loss for discriminator

$$-V_i(\phi, \lambda, x_i, z_i) = -\log\left(d_\lambda(q_\phi(x_i))\right)$$

*논문에는 학습 절차 정의가 수식으로 제시되어 있지 않아 새로 정리한 내용

• Variational Auto Encoder

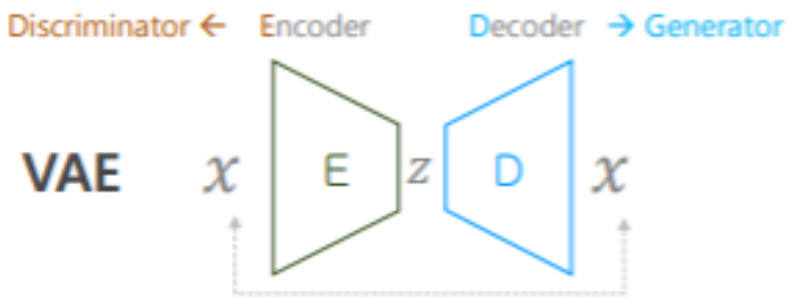
- GAN Incorporating Label Information

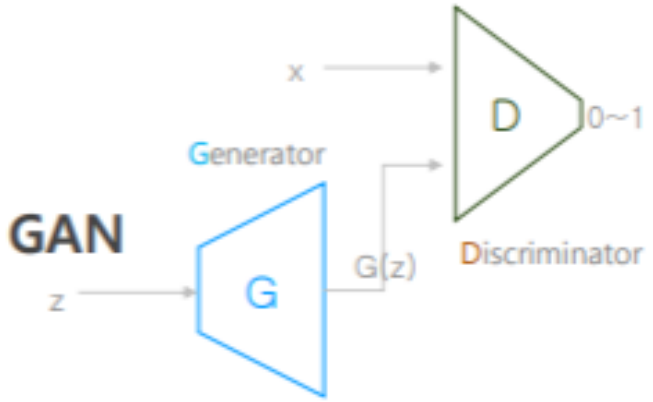


• Variational Auto Encoder

- VAE vs GAN

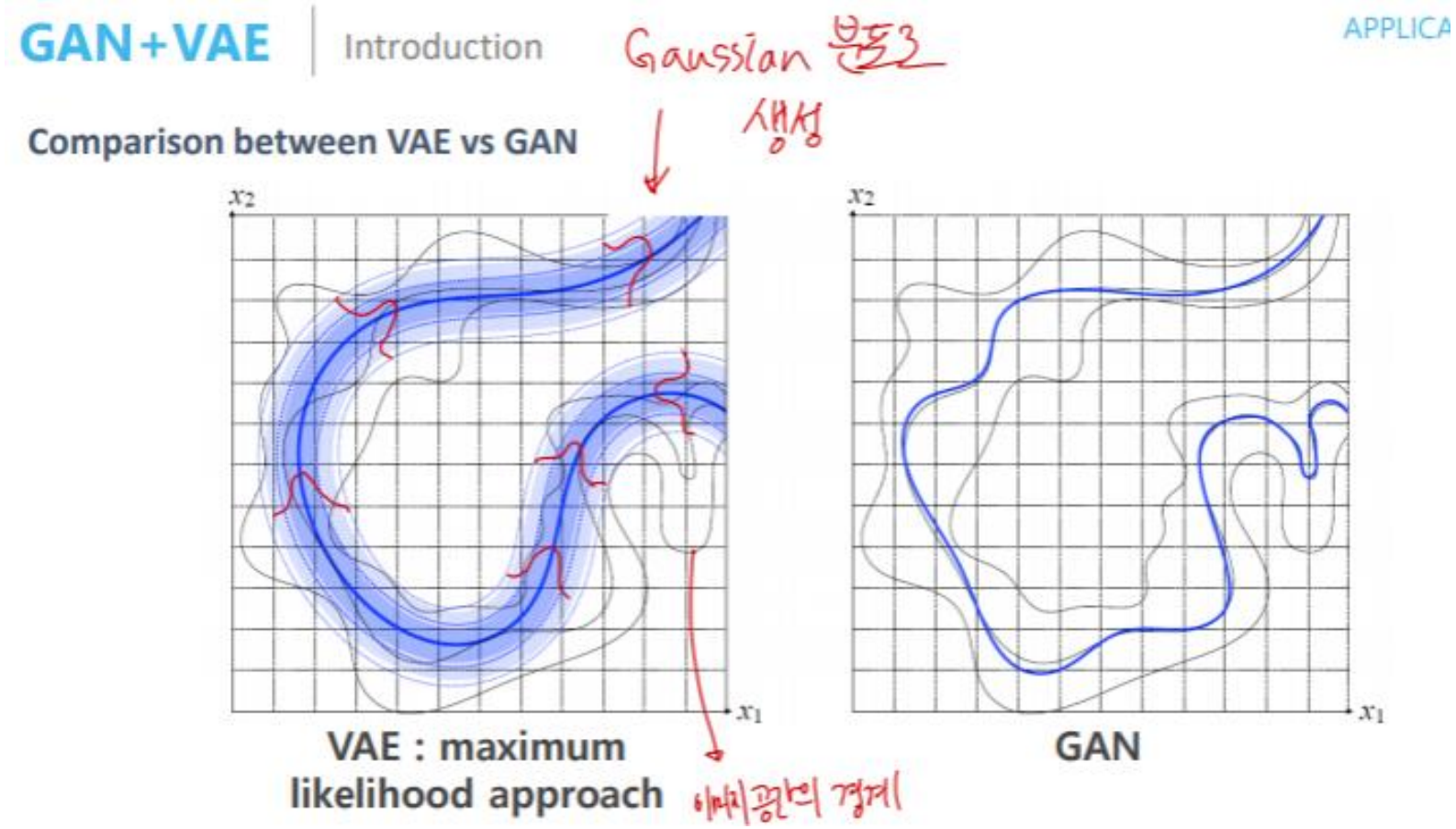
Model	Optimization	Image Quality	Generalization
VAE	• Stochastic gradient descent • Converge to local minimum • Easier	• Smooth • Blurry	• Tend to remember input images
GAN	• Alternating stochastic gradient descent • Converge to saddle points • Harder ❖ Model collapsing ❖ Unstable convergence	• Sharp • Artifact	• Generate new unseen images





- # Variational Auto Encoder

- VAE vs GAN



Reference

https://www.youtube.com/watch?v=o_peo6U7IRM&t=0s