Autoencoders

A way for Unsupervised Learning of Nonlinear Manifold

이활석

한글로 작성된 부분은 개인적인 견해인 경우가 많으니 참조하시기 바랍니다.
각 슬라이드 아래에는 해당 슬라이드에서 사용된 자료 출처가 제시되어 있습니다.

Autoencoder

From Wikipedia, the free encyclopedia

An **autoencoder**, **autoassociator** or **Diabolo network**[1]:19 is an artificial neural network used for unsupervised learning of efficient codings.[2][3] The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for the purpose of dimensionality reduction. Recently, the autoencoder concept has become more widely used for learning generative models of data.[4][5]

Contents [hide]

[KEYWORDS]

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

Nonlinear dimensionality reduction

From Wikipedia, the free encyclopedia

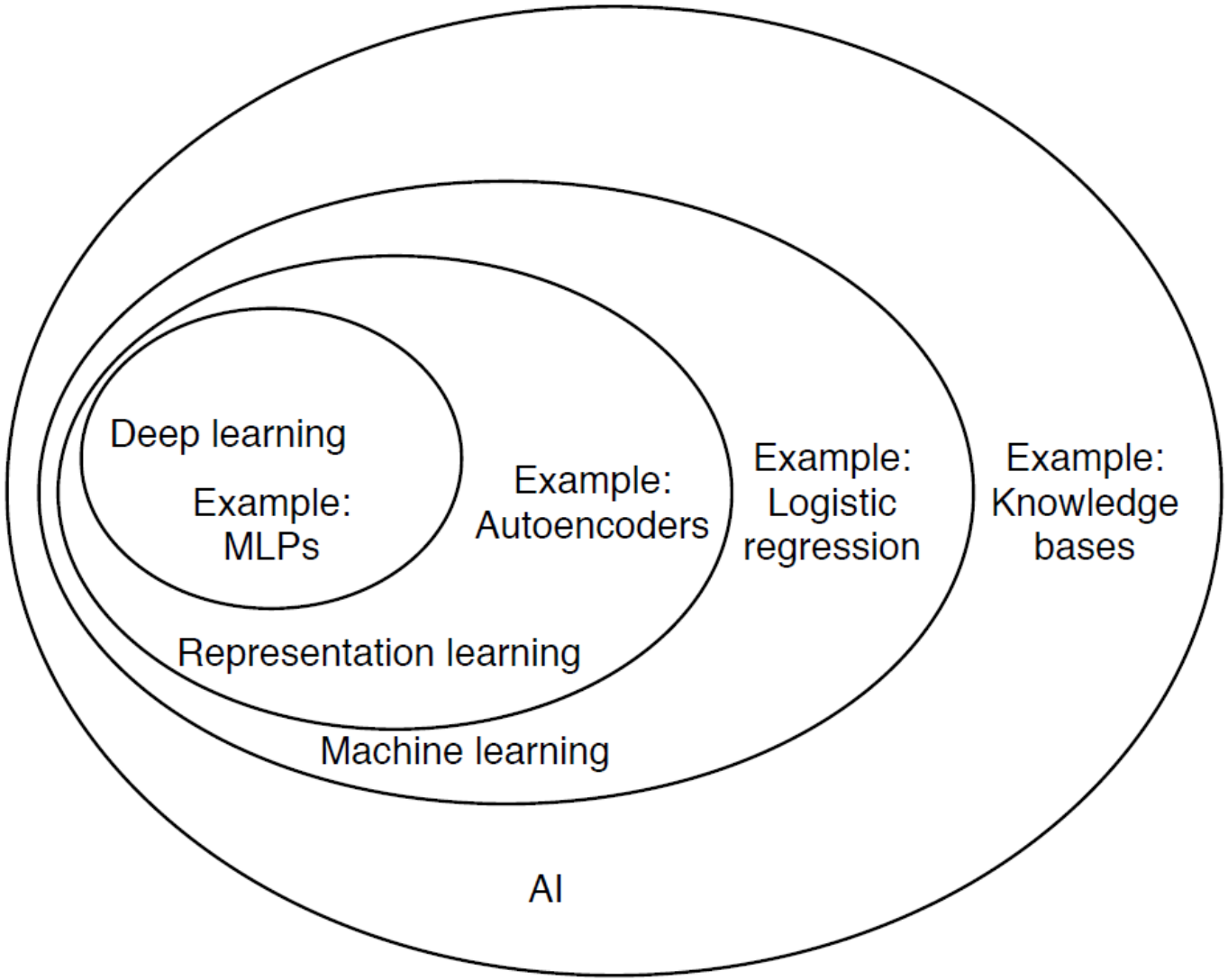
Below is a summary of some of the important algorithms from the history of **manifold learning** and **nonlinear dimensionality reduction** (NLDR).[1][2] Many of these non-linear dimensionality reduction methods are related to the linear methods listed below. Non-linear methods can be broadly classified into two groups: those that provide a mapping (either from the high-dimensional space to the low-dimensional embedding or vice versa), and those that just give a visualisation. In the context of machine learning, mapping methods may be viewed as a preliminary feature extraction step, after which pattern recognition algorithms are applied. Typically those that just give a visualisation are based on proximity data – that is, distance measurements.

Contents [hide]

[KEYWORDS]

- Unsupervised learning
- Nonlinear Dimensionality reduction
  - = Representation learning
  - = Efficient coding learning
  - = Feature extraction
  - = Manifold learning
- Generative model learning

FOUR KEYWORDS | Representation learning



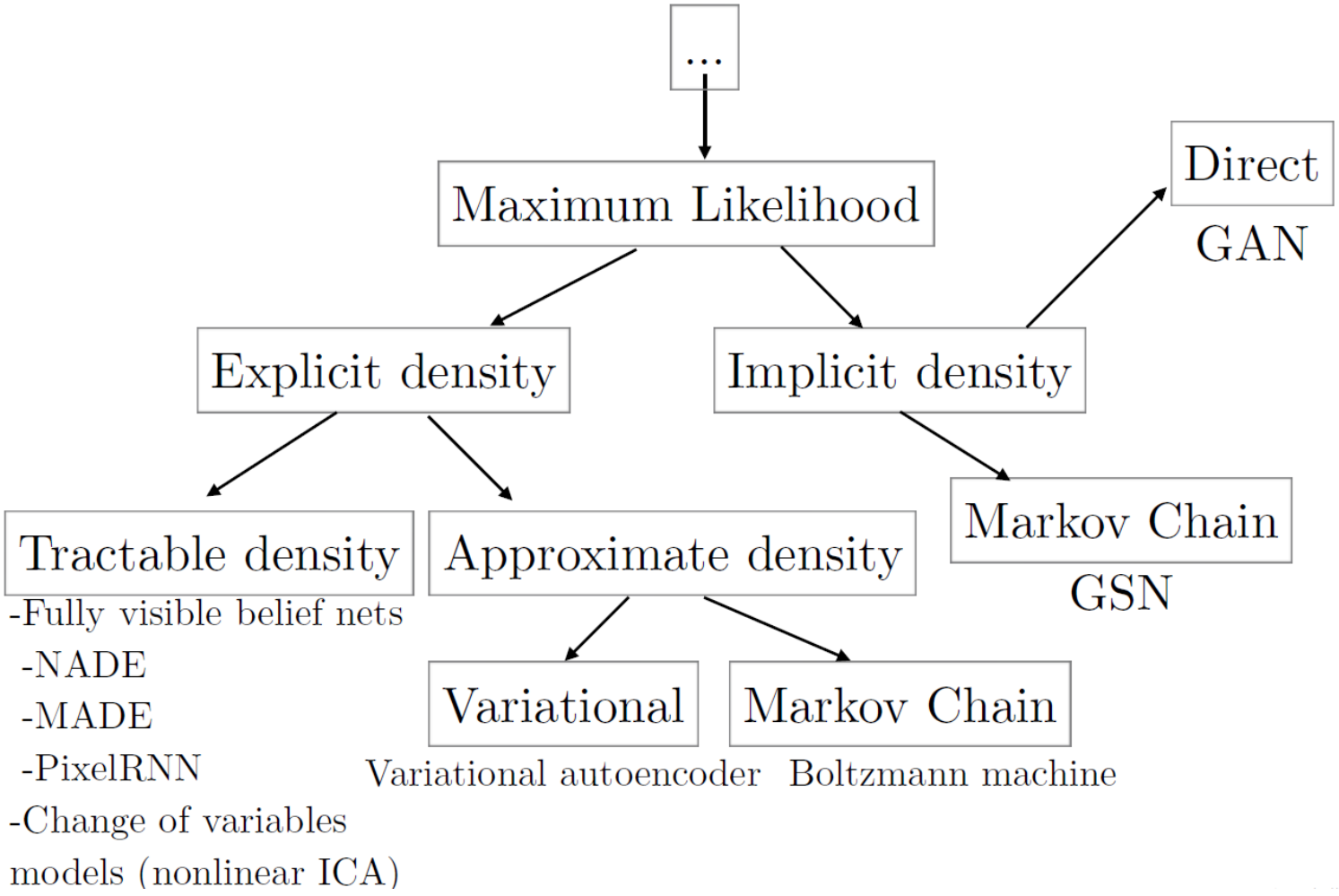
[KEYWORDS]

- Unsupervised learning
- Nonlinear Dimensionality reduction
  - = Representation learning
  - = Efficient coding learning
  - = Feature extraction
  - = Manifold learning
- Generative model learning

http://videolectures.net/kdd2014_bengio_deep_learning/

FOUR KEYWORDS

ML density estimation

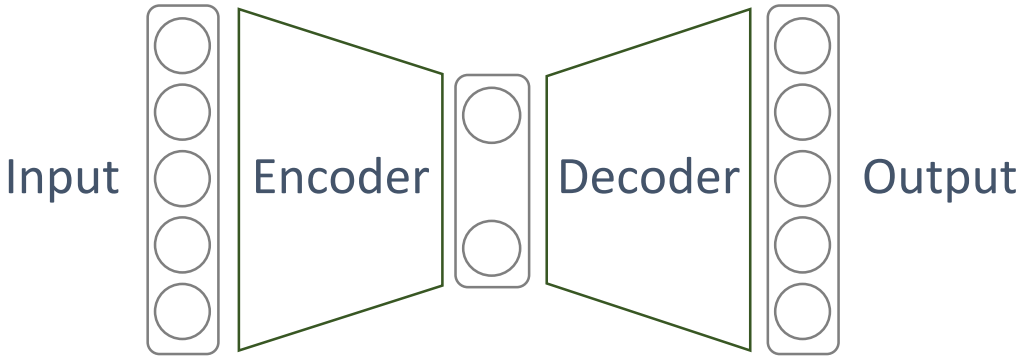


[KEYWORDS]

- Unsupervised learning
- Nonlinear Dimensionality reduction
  - = Representation learning
  - = Efficient coding learning
  - = Feature extraction
  - = Manifold learning
- Generative model learning
- ML density estimation

http://www.iangoodfellow.com/slides/2016-12-04-NIPS.pdf

FOUR KEYWORDS | Summary



[4 MAIN KEYWORDS]

1. Unsupervised learning
2. Manifold learning
3. Generative model learning
4. ML density estimation

오토인코더를 학습할 때:

학습 방법은 비교사 학습 방법을 따르며, ⇢ Unsupervised learning

Loss는 negative ML로 해석된다. ⇢ ML density estimation

학습된 오토인코더에서:

인코더는 차원 축소 역할을 수행하며, ⇢ Manifold learning

디코더는 생성 모델의 역할을 수행한다. ⇢ Generative model learning

CONTENTS

01. Revisit Deep Neural Networks

- Machine learning problem
- Loss function viewpoints I : Back-propagation
- Loss function viewpoints II : Maximum likelihood
- Maximum likelihood for autoencoders

02. Manifold Learning

03. Autoencoders

04. Variational Autoencoders

05. Applications

KEYWORD : ML density estimation

딥뉴럴넷을 학습할 때 사용되는 로스함수는 다양한 각도에서 해석할 수 있다.

그 중 하나는 back-propagation 알고리즘이 좀 더 잘 동작할 수 있는지에 대한 해석이다.
(gradient-vanishing problem이 덜 발생할 수 있다는 해석)

다른 하나는 negative maximum likelihood로 보고 특정 형태의 loss는 특정 형태의 확률분포를 가정한다는 해석이다.

Autoencoder를 학습하는 것 또한 maximum likehihood 관점에서의 최적화로 볼 수 있다.

ML PROBLEM | Classic Machine Learning

01. Collect training data

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

02. Define functions

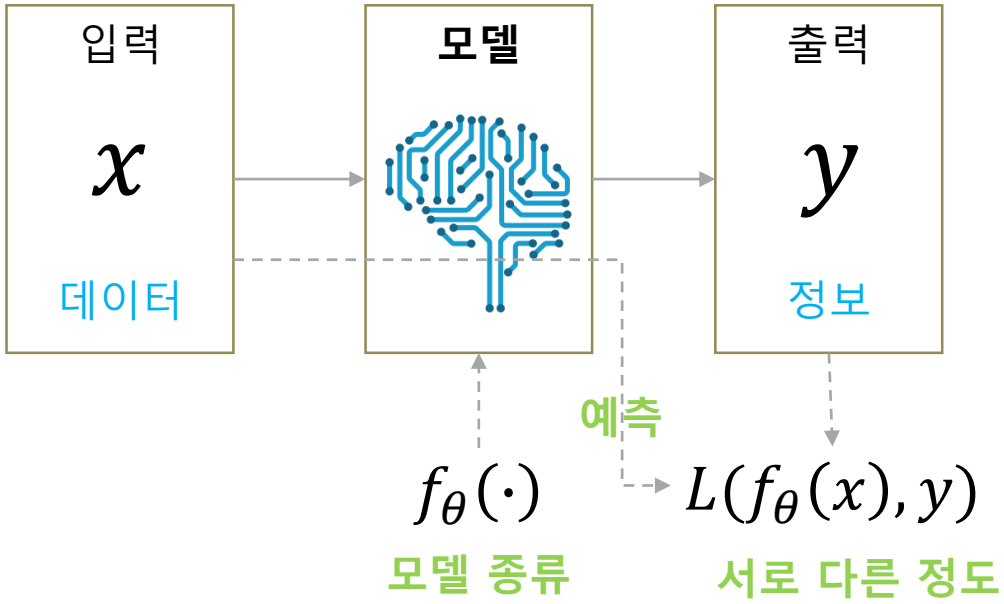
- Output : $f_\theta(x)$
- Loss : $L(f_\theta(x), y)$

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

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

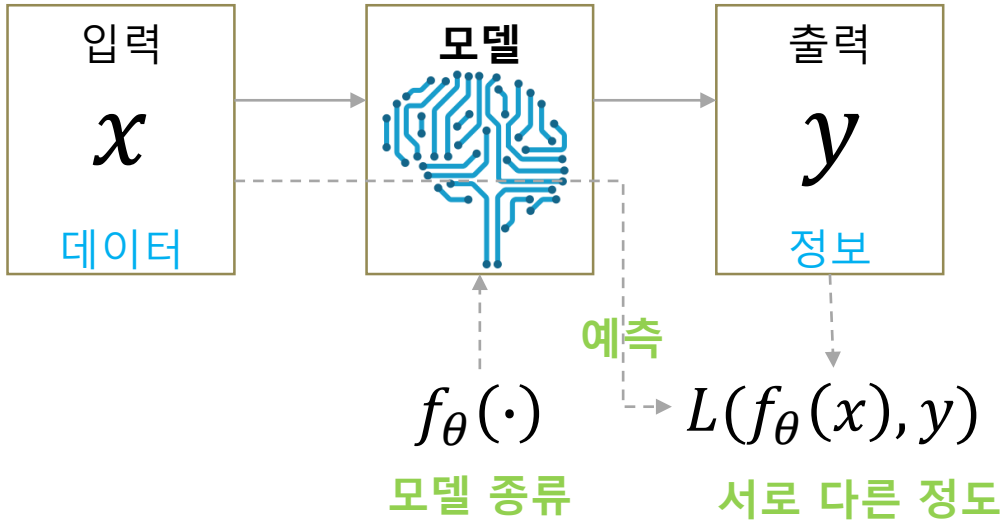
고정 입력, 고정 출력

01. Collect training data

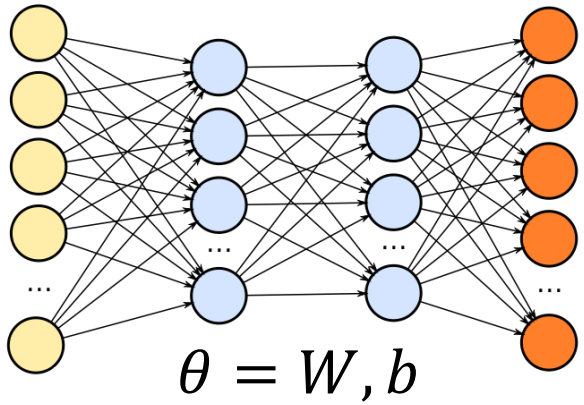
02. Define functions

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. Collect training data

02. Define functions

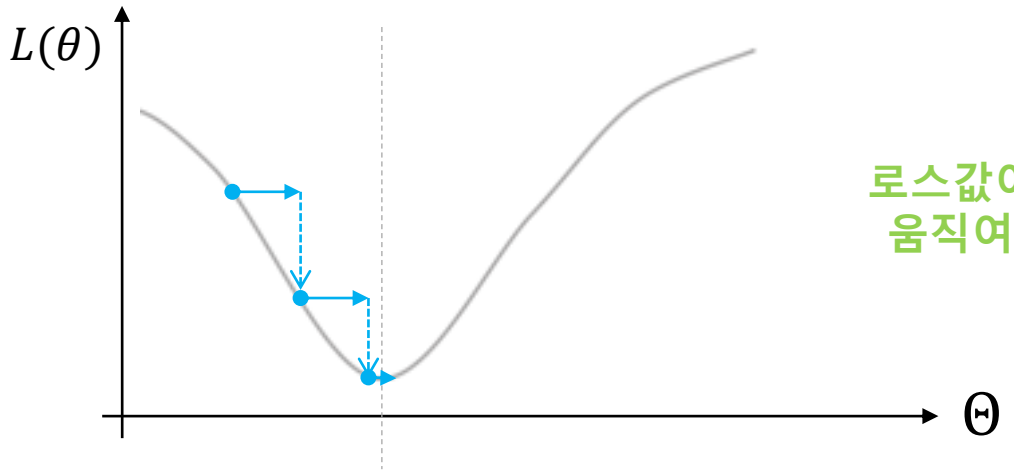
03. Learning/Training

04. Predicting/Testing

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

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

Questions	*Strategies*
How to update $\theta \rightarrow \theta + \Delta\theta$	Only if $L(\theta + \Delta\theta) < L(\theta)$
When we stop to search??	If $L(\theta + \Delta\theta) == L(\theta)$



로스값이 줄어드는 방향으로 계속 이동하고,
움직여도 로스값이 변함 없을 경우 멈춘다

01. Collect training data

02. Define functions

03. Learning/Training

04. Predicting/Testing

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

$L(\theta + \Delta\theta) - L(\theta) = \Delta L = \nabla L \cdot \Delta\theta$

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

Learning rate를 사용하여 조금씩 파라미터 값을 바꾸는 것은 로스 함수의 1차 미분항까지만 사용했기에 아주 좁은 영역에서만 감소 방향이 정확하기 때문이다.

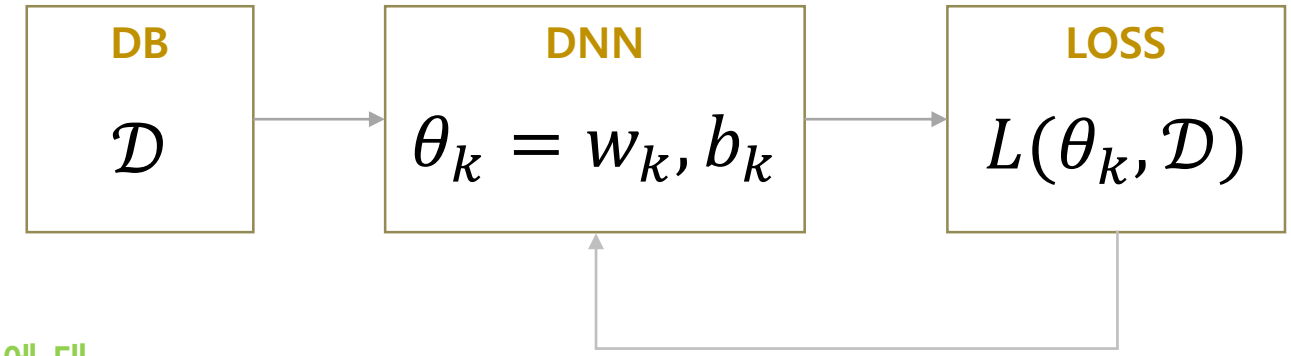
01. Collect training data

02. Define functions

03. Learning/Training

04. Predicting/Testing

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



전체 데이터에 대한 로스 함수가 각 데이터 샘플에 대한 로스의 합으로 구성되어 있기에 미분 계산을 효율적으로 할 수 있다.

만약 곱으로 구성되어 있으면 미분을 위해 모든 샘플의 결과를 메모리에 저장해야 한다.

$$L(\theta_k, \mathcal{D}) = \sum_i L(\theta_k, \mathcal{D}_i)$$

$$\nabla L(\theta_k, \mathcal{D}) = \sum_i \nabla L(\theta_k, \mathcal{D}_i)$$

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

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

M : batch size

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

원래는 모든 데이터에 대한 로스 미분값의 합을 구한 후 파라미터를 갱신해야 하지만, 배치 크기만큼만 로스 미분값의 합을 구한 후 파라미터를 갱신한다.

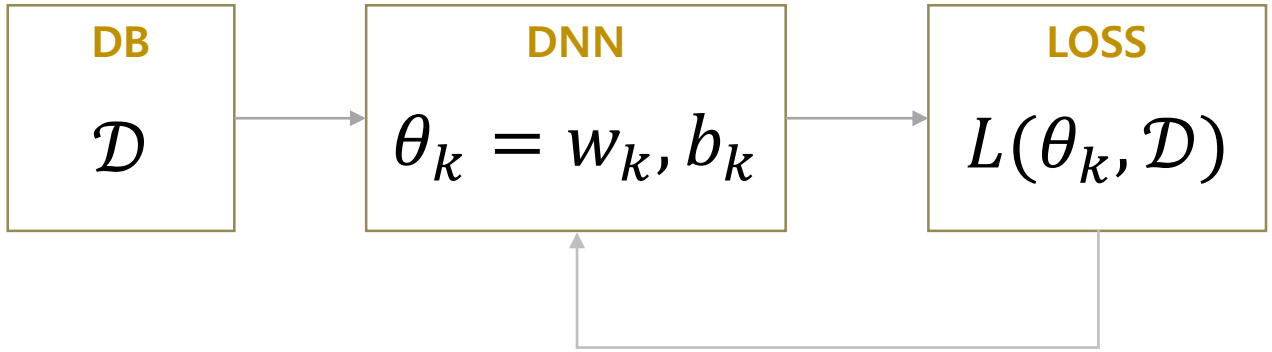
01. Collect training data

02. Define functions

03. Learning/Training

04. Predicting/Testing

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



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

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

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

특정 레이어에서 파라미터 갱신식

[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

$$\nabla_{b^l} C = \delta^l$$

4. Gradient of C in terms of weight

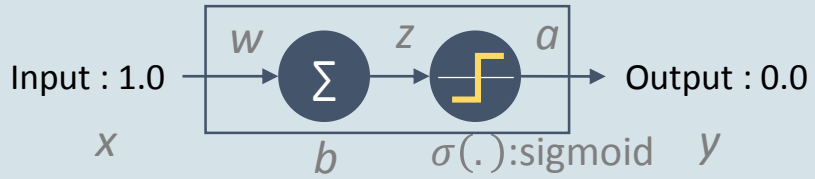
$$\nabla_{W^l} C = \delta^l (a^{l-1})^T$$

로스함수의 미분값이 딥뉴럴넷 학습에서 제일 중요!!

http://neuralnetworksanddeeplearning.com/chap2.html

Type 1 : Mean Square Error / Quadratic loss

OBJECT



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

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

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

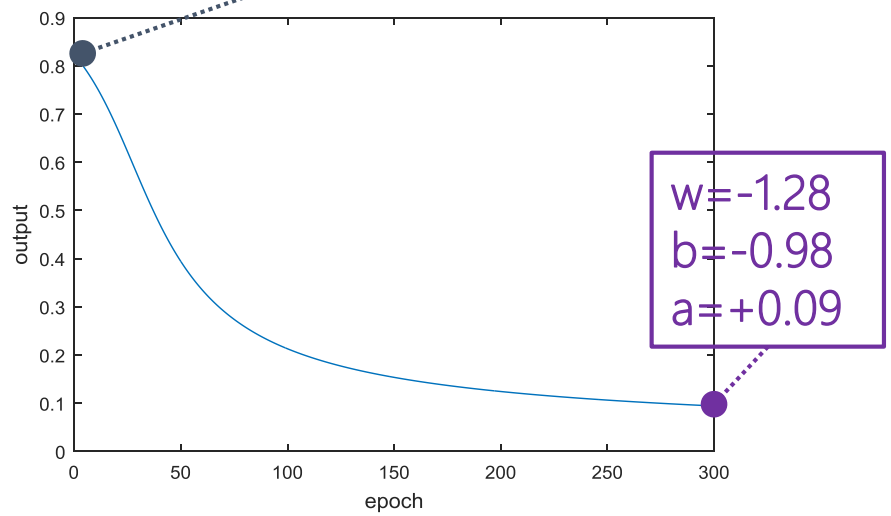
$$\frac{\partial C}{\partial w} = x\delta = \delta$$

$$w = w - \eta\delta$$

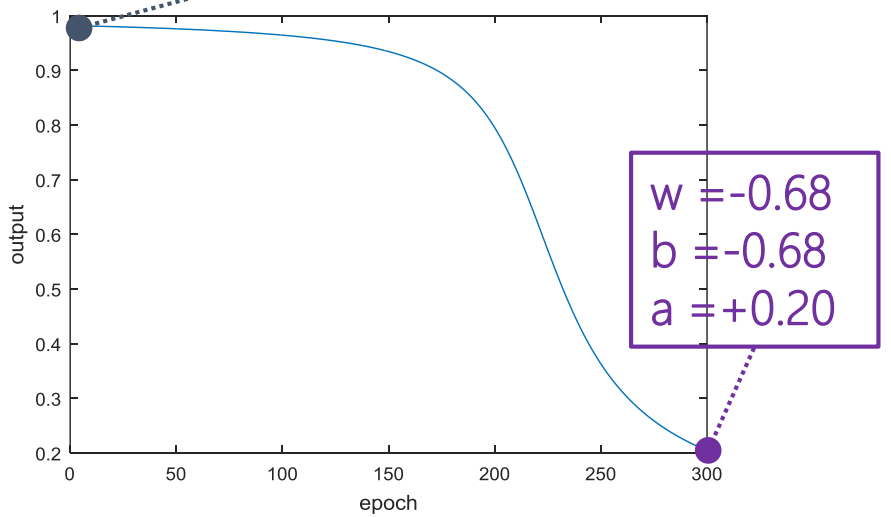
$$\frac{\partial C}{\partial b} = \delta$$

$$b = b - \eta\delta$$

w_0=+0.6, b_0=+0.9, a_0=+0.82



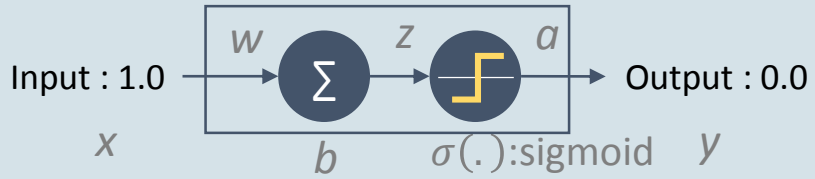
w_0=+2.0, b_0=+2.0, a_0=+0.98



http://neuralnetworksanddeeplearning.com/chap2.html

Type 1 : Mean Square Error / Quadratic loss

OBJECT



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

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

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

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

$$w = w - \eta\delta$$

$$\frac{\partial C}{\partial b} = \delta$$

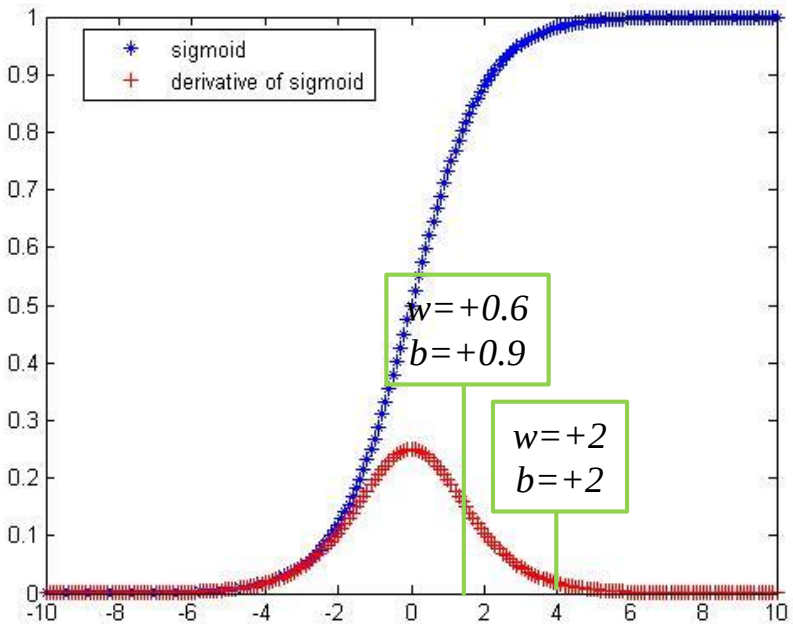
$$b = b - \eta\delta$$

Learning slow means are $\partial C / \partial w, \partial C / \partial b$ small !!

Why they are small???

$$\frac{\partial C}{\partial w} = x\delta = xa\sigma'(z) = a\sigma'(z)$$

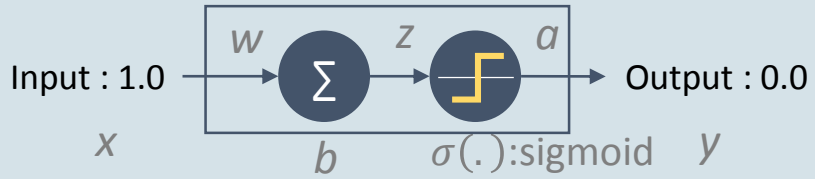
$$\frac{\partial C}{\partial b} = \delta = a\sigma'(z)$$



http://neuralnetworksanddeeplearning.com/chap2.html

Type 2 : Cross Entropy

OBJECT



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.

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

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

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

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

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

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

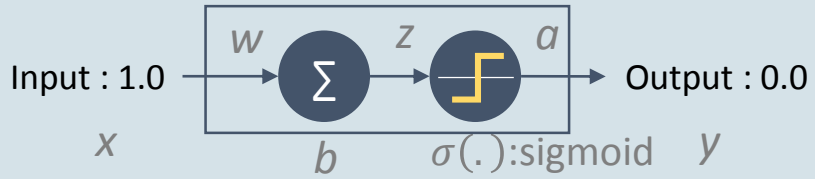
$$\sigma'(z) = \frac{\partial a}{\partial z} = \sigma'(z) = \big(1-\sigma(z)\big)\sigma(z) = (1-a)a$$

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

http://neuralnetworksanddeeplearning.com/chap2.html

Type 2 : Cross Entropy

OBJECT



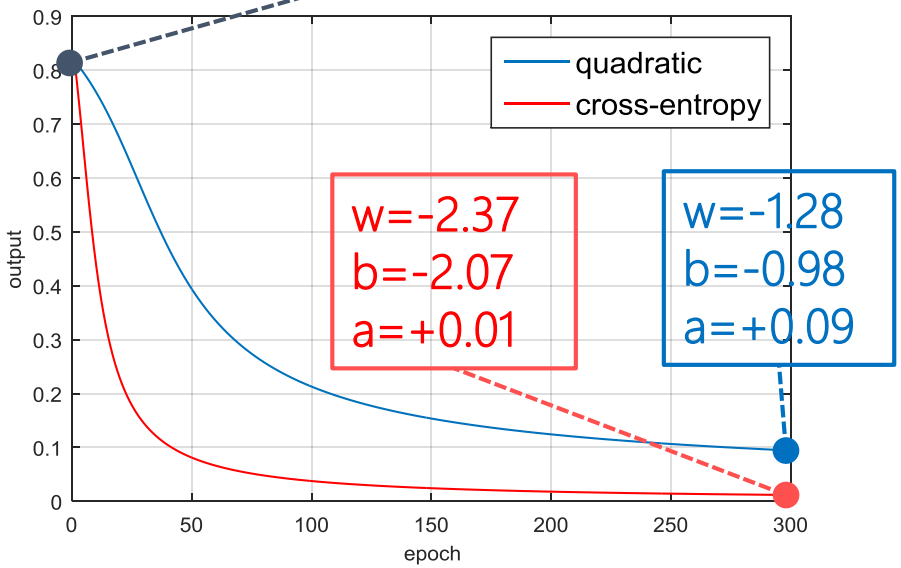
$$C = -[y\,ln\,a + (1-y)ln(1-a)]$$

$$\delta = a - y$$

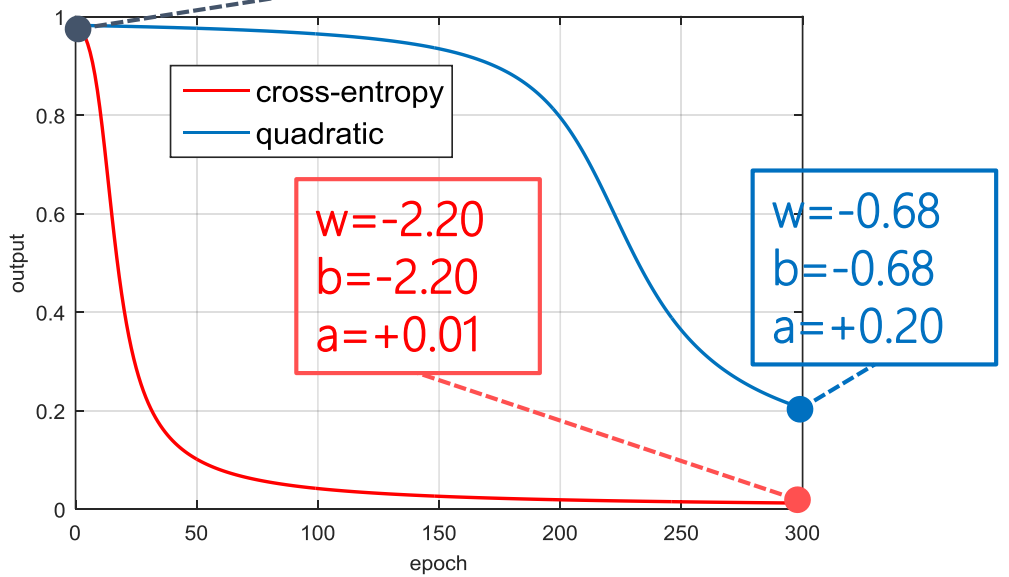
$$\frac{\partial C}{\partial w} = x\delta = \delta \qquad w = w - \eta\delta$$

$$\frac{\partial C}{\partial b} = \delta \qquad b = b - \eta\delta$$

w_0=+0.6, b_0=+0.9, a_0=+0.82



w_0=+2.0, b_0=+2.0, a_0=+0.98



http://neuralnetworksanddeeplearning.com/chap2.html

Back to Machine Learning Problem

01. Collect training data

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

02. Define functions

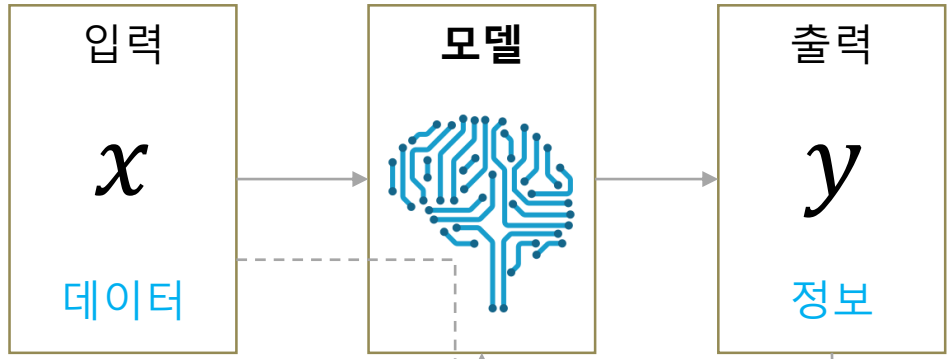
- Output : $f_\theta(x)$
- Loss : $-\log(p(y|f_\theta(x)))$

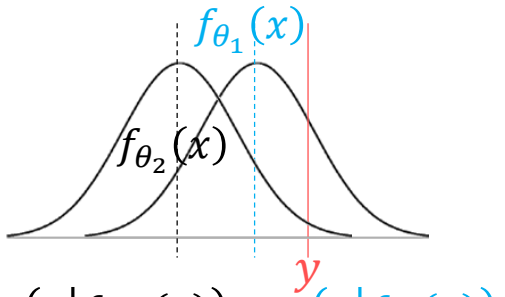
03. Learning/Training

Find the optimal parameter

04. Predicting/Testing

Compute optimal function output





$p(y|f_{\theta_{12}}(x)) < p(y|f_{\theta_1}(x))$

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

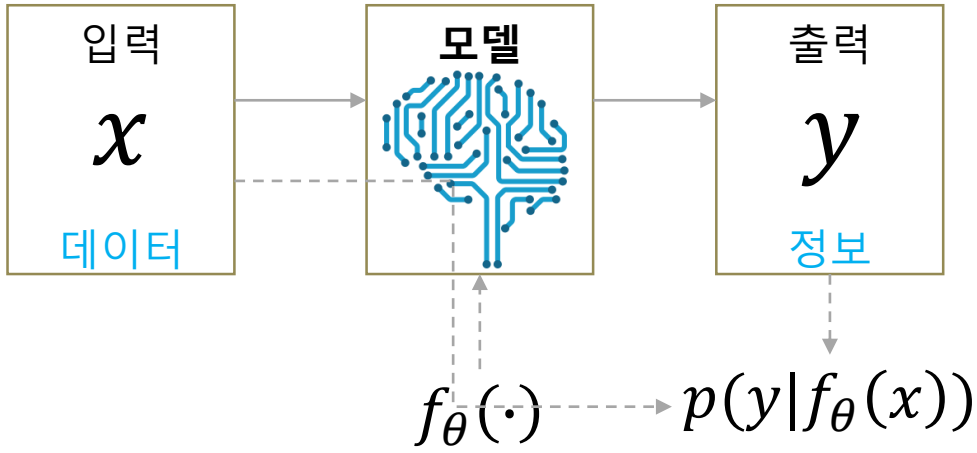
$$y_{new} \sim p(y|f_{\theta^*}(x_{new}))$$

고정 입력, 고정/다른 출력

LOSS FUNCTION | View-Point II : Maximum Likelihood

Back to Machine Learning Problem

01. Collect training data
02. Define functions
03. Learning/Training
04. Predicting/Testing



i.i.d Condition on $p(y|f_\theta(x))$

Assumption 1 : Independence

All of our data is independent of each other

$$p(y|f_\theta(x)) = \prod_i p_{D_i}(y|f_\theta(x_i))$$

Assumption 2: Identical Distribution

Our data is identically distributed

$$p(y|f_\theta(x)) = \prod_i p(y|f_\theta(x_i))$$

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

Assumption 1. OK
Total loss of DNN over training samples is the sum of loss for each training sample

Assumption 2. OK
Loss for each training example is a function of final output of DNN

Univariate cases

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

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

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

Multivariate cases

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

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

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

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

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

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

Cross-entropy

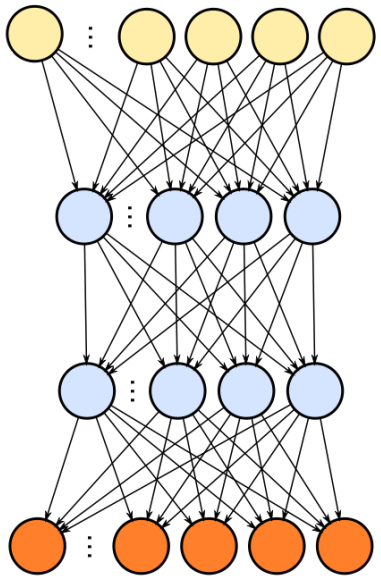
Also called Generalized Bernoulli or Multinoulli distribution

Multivariate cases

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

Distribution estimation

$p(y_i|x_i)$

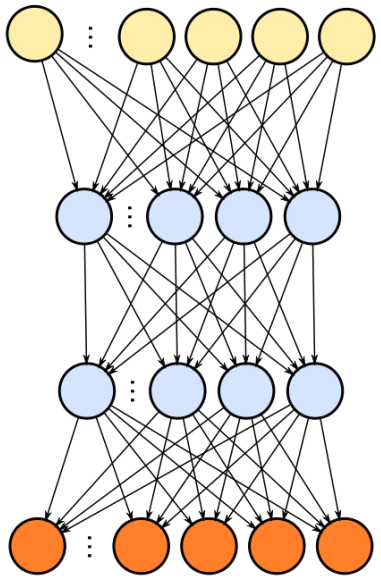


x_i

Likelihood값을 예측하는 것이 아니라,
Likelihood의 파라미터값을 예측하는 것이다.

Gaussian distribution

$f_\theta(x_i) = \mu_i$

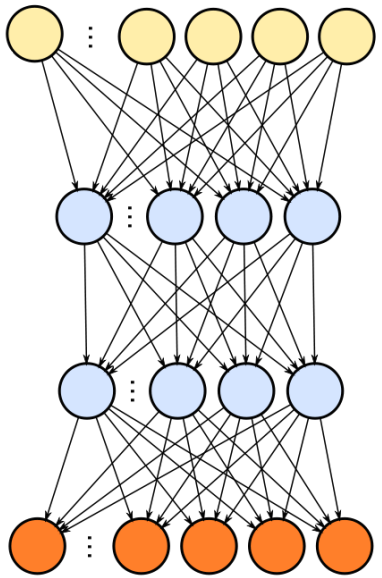


x_i

Mean Squared Error

Categorical distribution

$f_\theta(x_i) = p_i$



x_i

Cross-entropy

Let's see Yoshua Bengio's slide

Log-Likelihood for Neural Nets

- Estimating a conditional probability $P(Y|X)$
- Parametrize it by $P(Y|X) = P(Y|\omega = f_\theta(X))$
- Loss = $-\log P(Y|X)$
- E.g. Gaussian *Y*, $\omega = (\mu, \sigma)$

  typically only μ is the network output, depends on *X*

  Equivalent to MSE criterion:

  Loss = $-\log P(Y|X) = \log \sigma + ||f_\theta(X) - Y||^2/\sigma^2$
- E.g. Multinoulli *Y* for classification,

  $\omega_i = P(Y = i|x) = f_{\theta,i}(X) = \text{softmax}_i(a(X))$

  Loss = $-\log \omega_Y = -\log f_{\theta,Y}(X)$

http://videolectures.net/kdd2014_bengio_deep_learning/

Connection to Autoencoders

	Autoencoder	Variational Autoencoder
Probability distribution	$p(x\|x)$	$p(x)$
Gaussian distribution	Mean Squared Error Loss	Mean Squared Error Loss
Categorical distribution	Cross-Entropy Loss	Cross-Entropy Loss

01. Revisit Deep Neural Networks

02. Manifold Learning

- Four objectives
- Dimension reduction
- Density estimation

03. Autoencoders

04. Variational Autoencoders

05. Applications

KEYWORDS : Manifold learning, Unsupervised learning

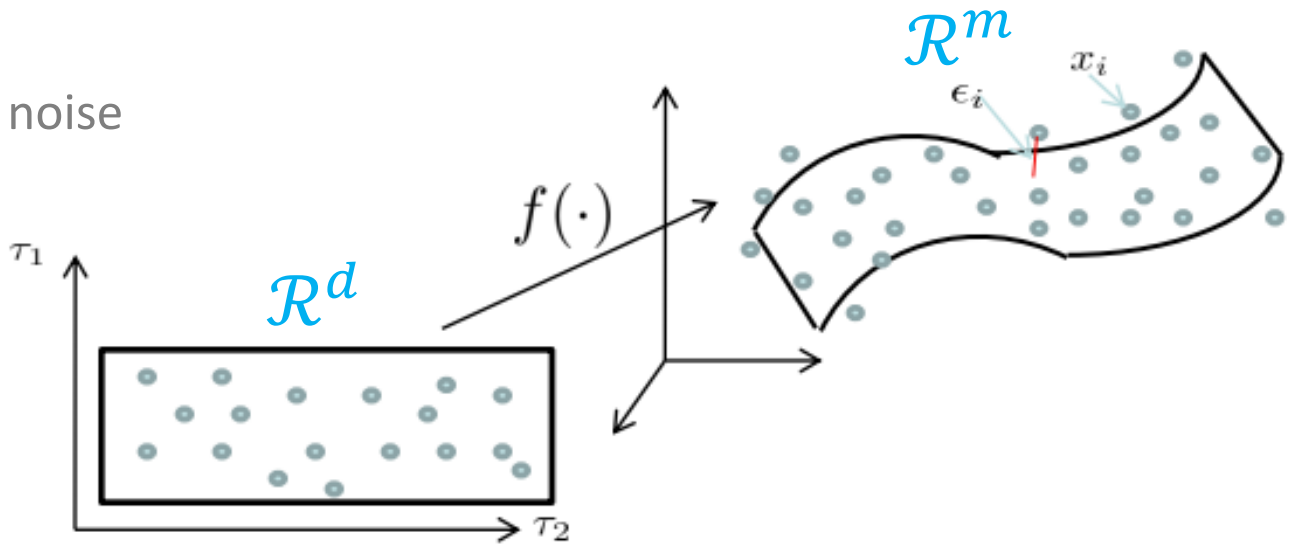
Autoencoder의 가장 중요한 기능 중 하나는 매니폴드를 학습한다는 것이다.

매니폴드 학습의 목적 4가지인 데이터 압축, 데이터 시각화, 차원의 저주 피하기, 유용한 특징 추출하기에 대해서 설명할 것이다.

Autoencoder의 주요 기능인 차원 축소, 확률 분포 예측과 관련된 기존의 방법들을 살펴보고, 그 한계점에 대해서 짚어볼 것이다.

- 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

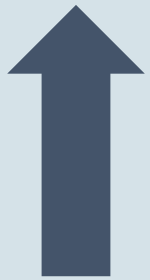
https://math.stackexchange.com/questions/1203714/manifold-learning-how-should-this-method-be-interpreted

INTRODUCTION

What is it useful for?

$\tau_i \in \mathcal{R}^d$

(0.32, -1.3, 1.2)



(3.5, -1.7, 2.8, -3.5, -1.4, 2.4, 2.7, 7.5)

$x_i \in \mathcal{R}^m$

Dimensionality Reduction is an Unsupervised Learning Task!

01. Data compression

02. Data visualization

03. Curse of dimensionality

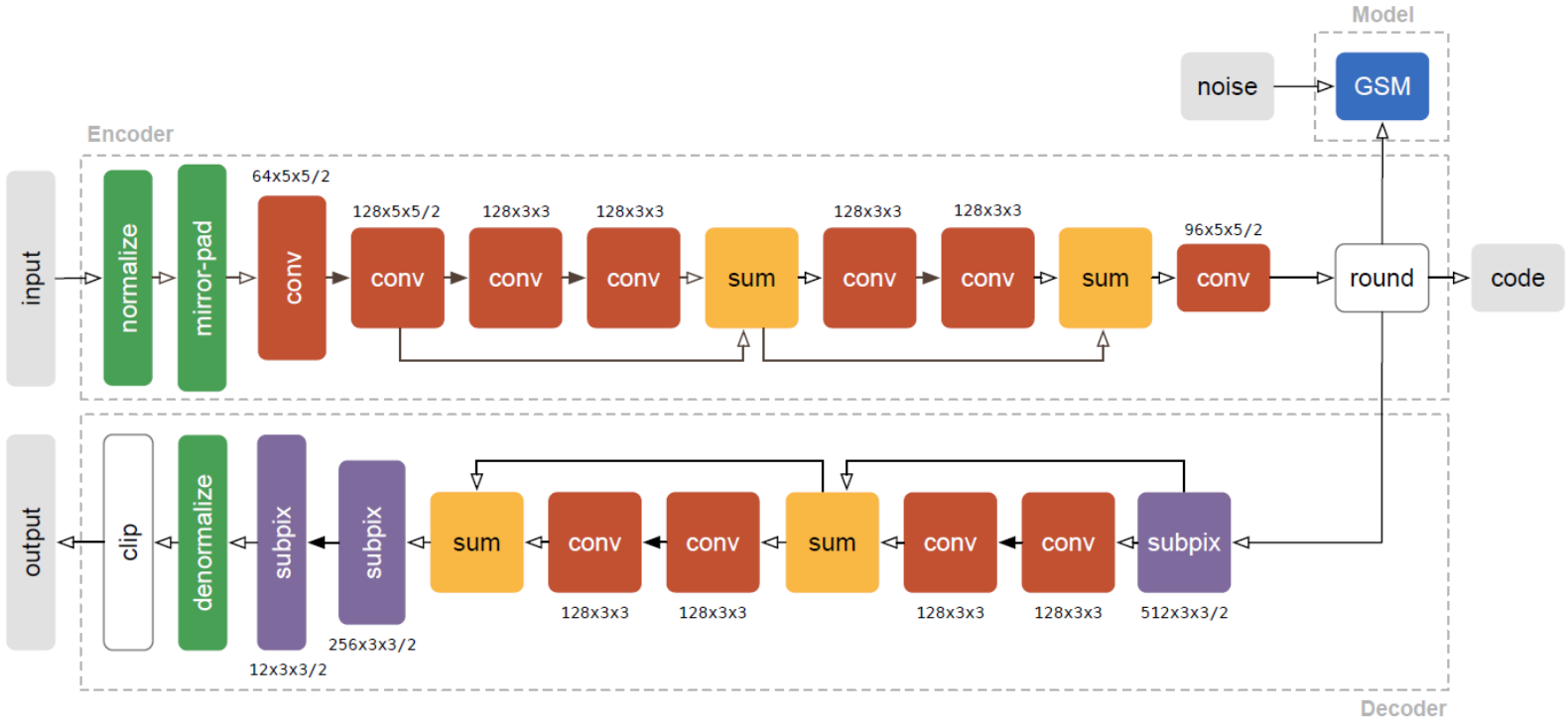
Manifold Hypothesis

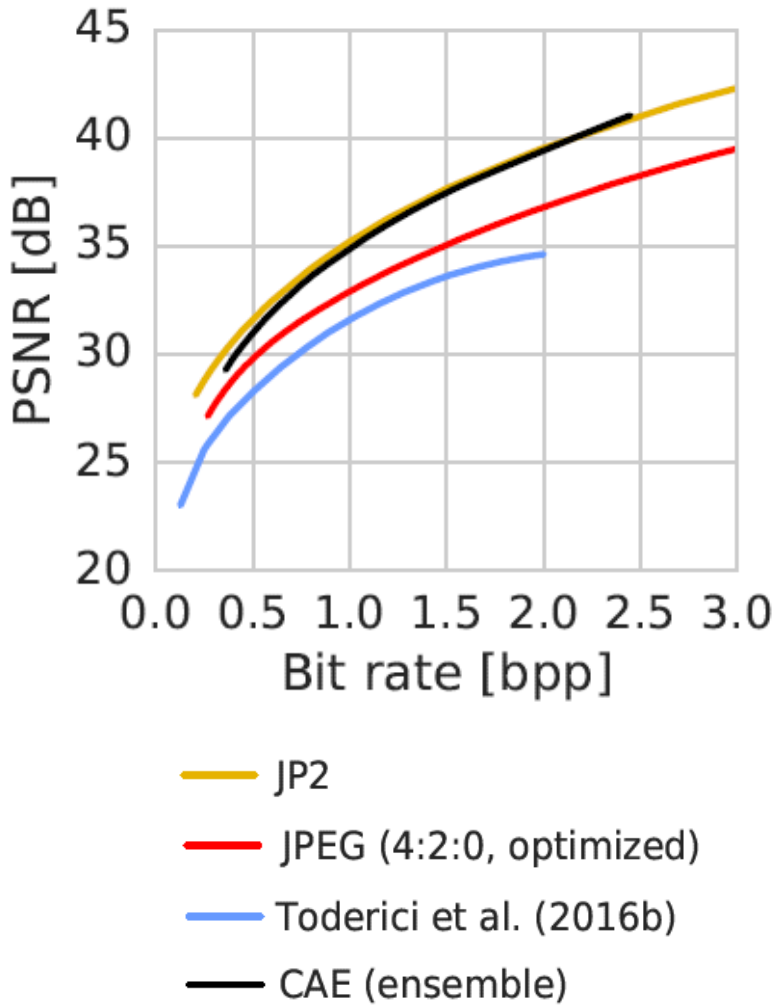
04. Discovering most important features

Reasonable distance metric
Needs disentagling the underlying explanatory factors (making sense of the data)

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

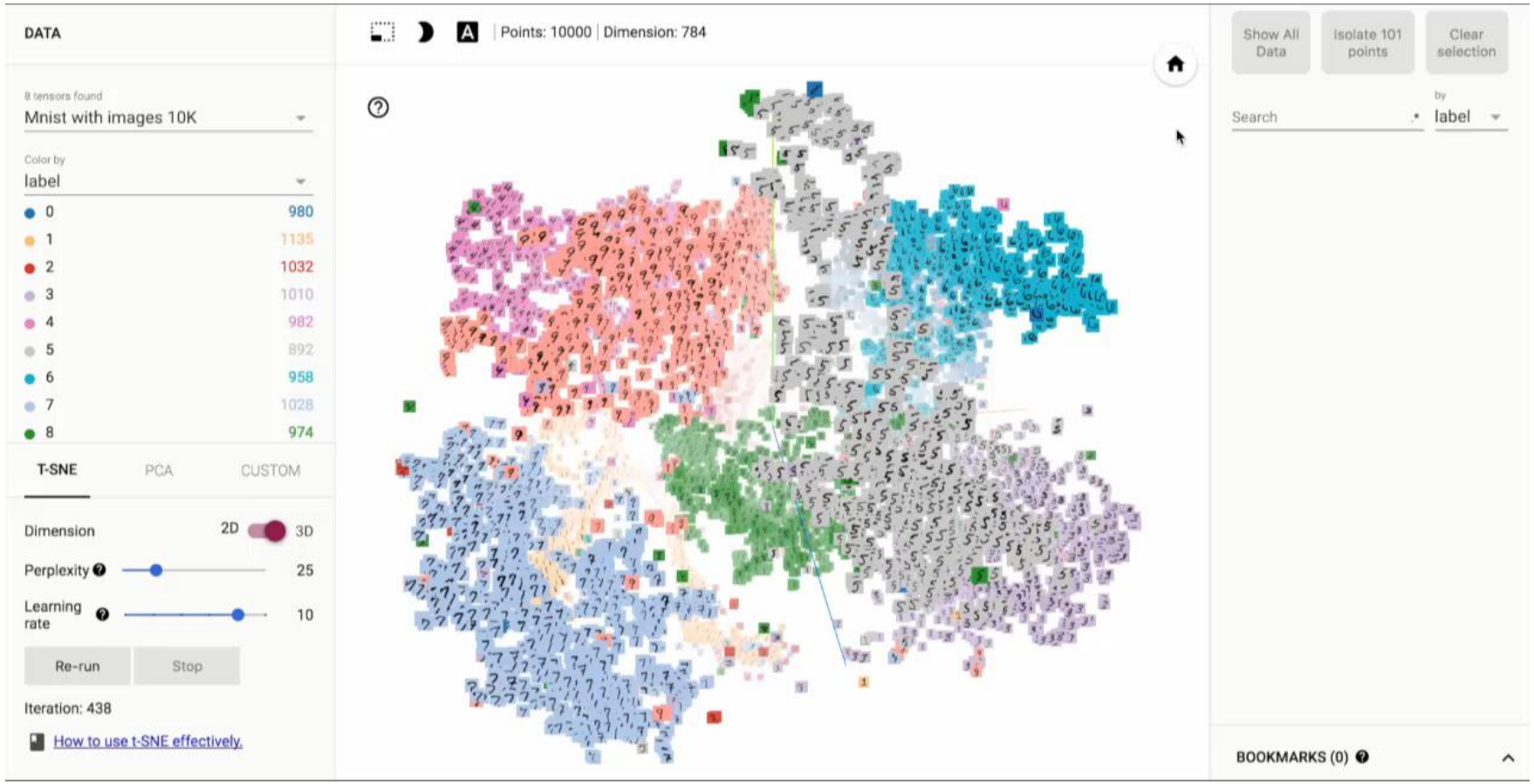
Example : Lossy Image Compression with compressive Autoencoders, ‘17.03.01





http://theis.io/media/publications/paper.pdf

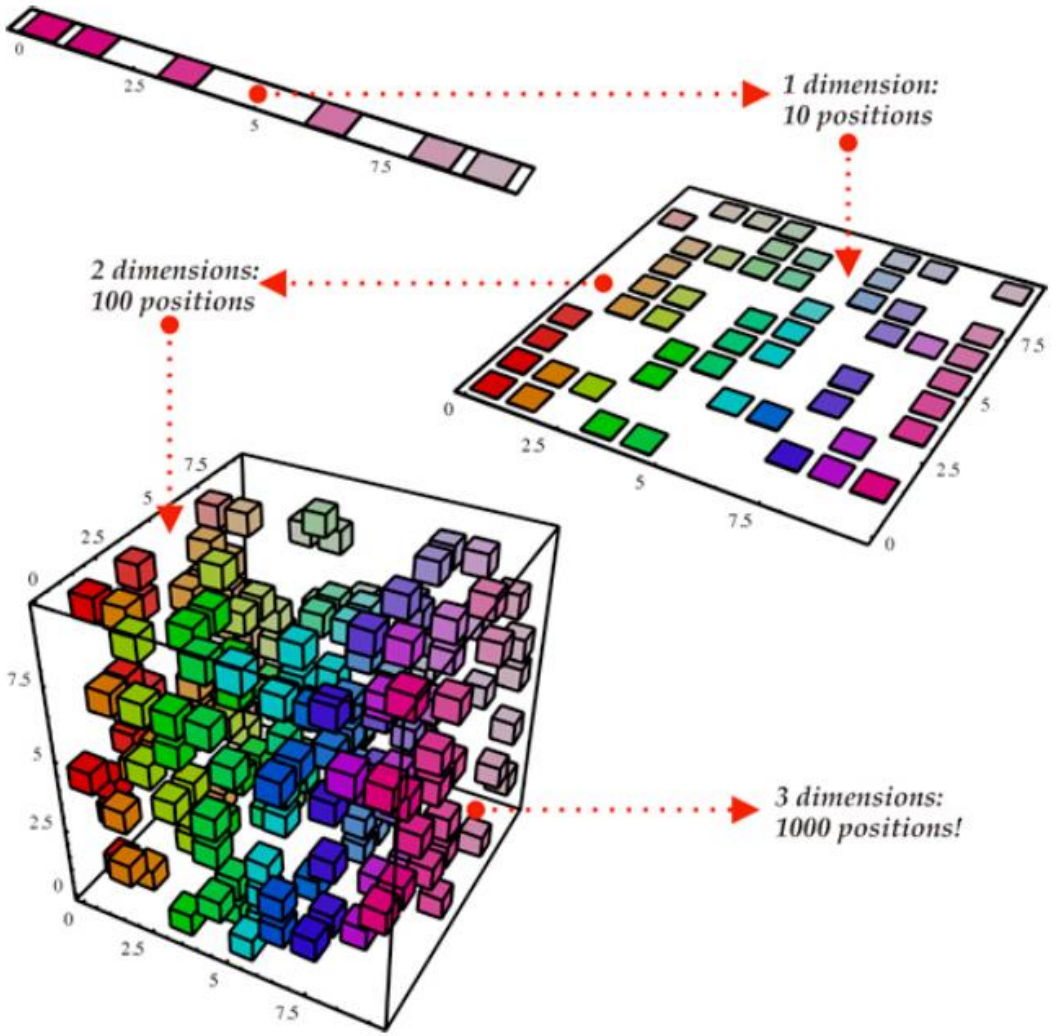
t-distributed stochastic neighbor embedding (t-SNE)



https://www.tensorflow.org/get_started/embedding_viz
http://vision-explorer.reactive.ai/#/?_k=aodf68
http://fontjoy.com/projector/

데이터의 차원이 증가할수록 해당 공간의 크기(부피)가 기하급수적으로 증가하기 때문에 동일한 개수의 데이터의 밀도는 차원이 증가할수록 급속도로 희박해진다.

따라서, 차원이 증가할수록 데이터의 분포 분석 또는 모델추정에 필요한 샘플 데이터의 개수가 기하급수적으로 증가하게 된다.



http://videolectures.net/kdd2014_bengio_deep_learning/

http://darkpgmr.tistory.com/145

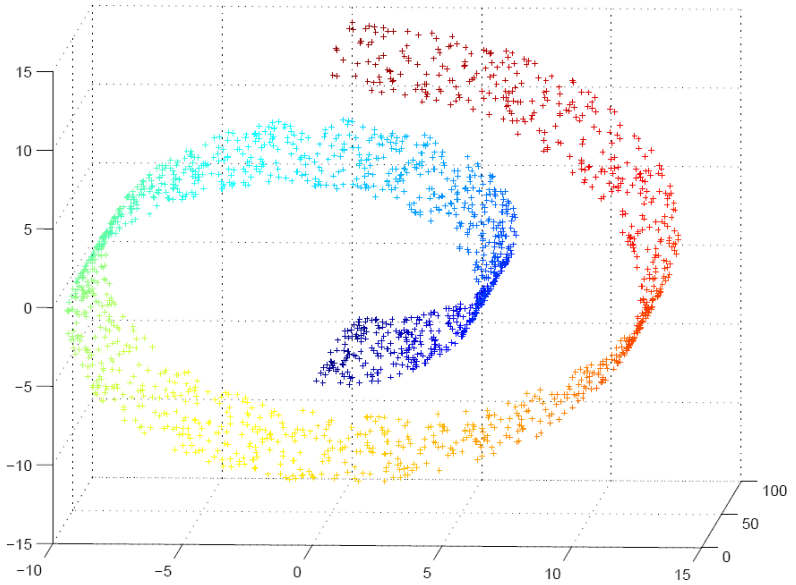
Manifold Hypothesis (assumption)

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.

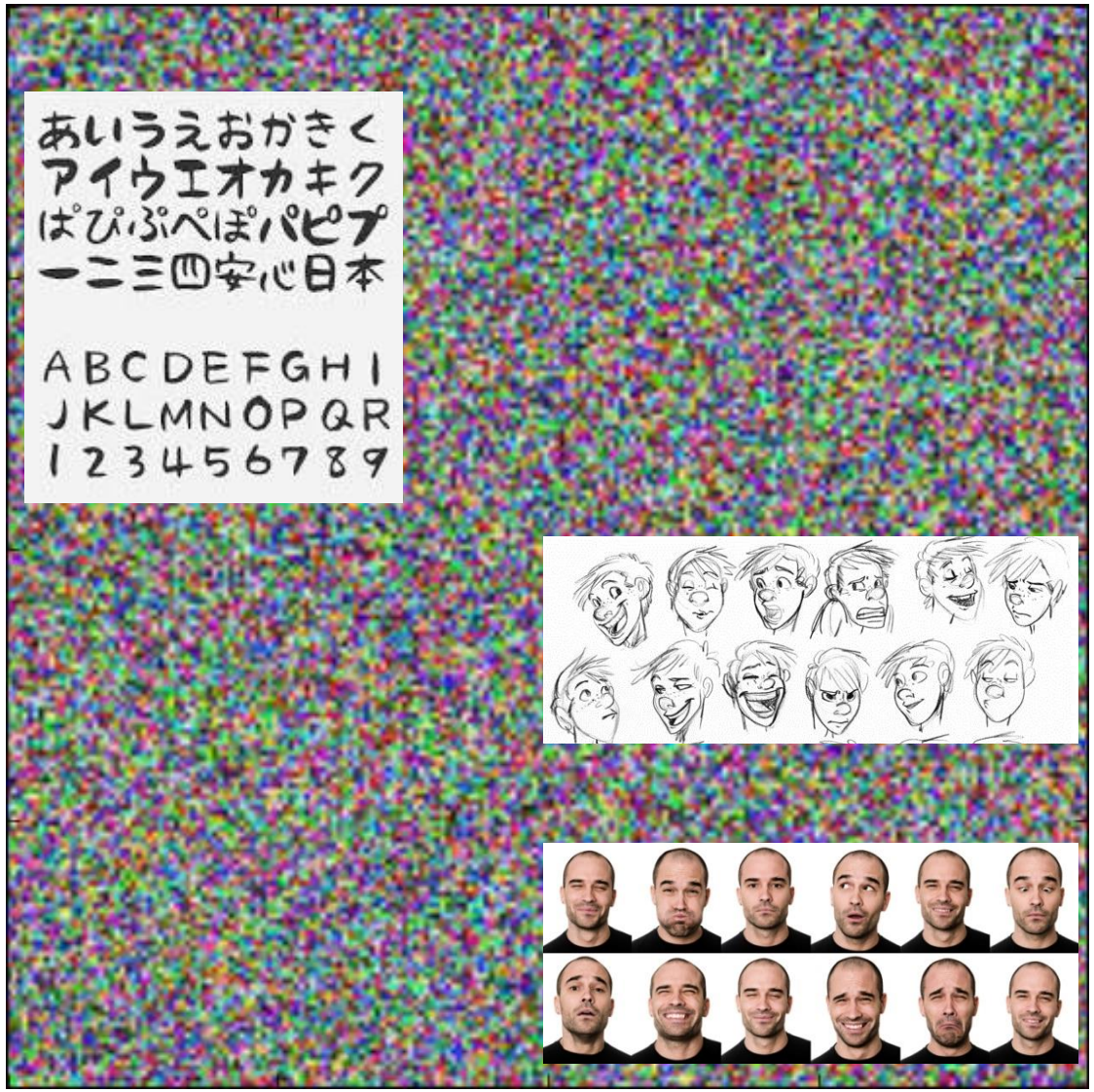


http://videolectures.net/deeplearning2015_vincent_autoencoders/?q=vincent%20autoencoder

Manifold Hypothesis (assumption)

- 200x200 RGB image has 10^96329 possible states.
  - Random image is just noisy.
- Natural images occupy a tiny fraction of that space
- suggests peaked density
- Realistic smooth transformations from one image to another continuous path along manifold
- Data density concentrates near a lower dimensional manifold
  - It can shift the curse from high d to d << m

http://www.freejapanesefont.com/category/calligraphy-2/
http://baanimotion.blogspot.kr/2014/05/animated-acting.html
https://medicalxpress.com/news/2013-03-people-facial.html



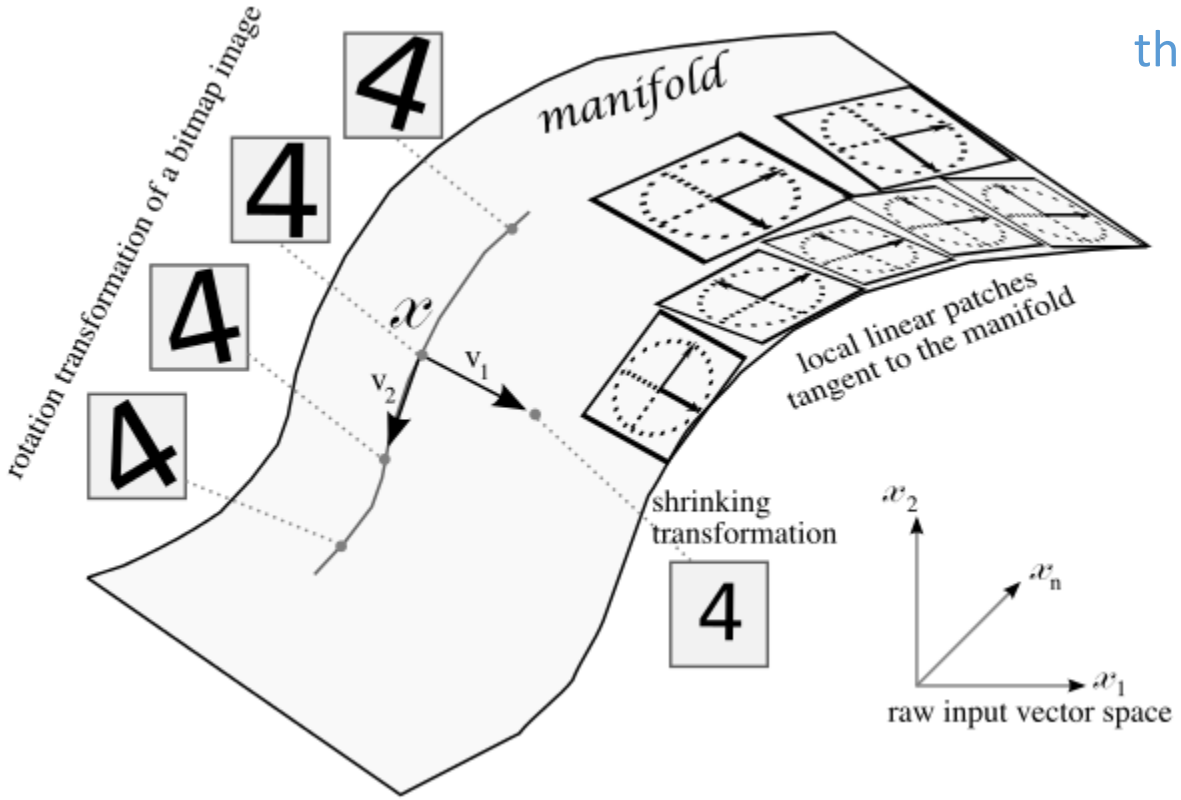
http://videolectures.net/deeplearning2015_vincent_autoencoders/?q=vincent%20autoencoder

OBJECTIVES

Discovering most important features

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.



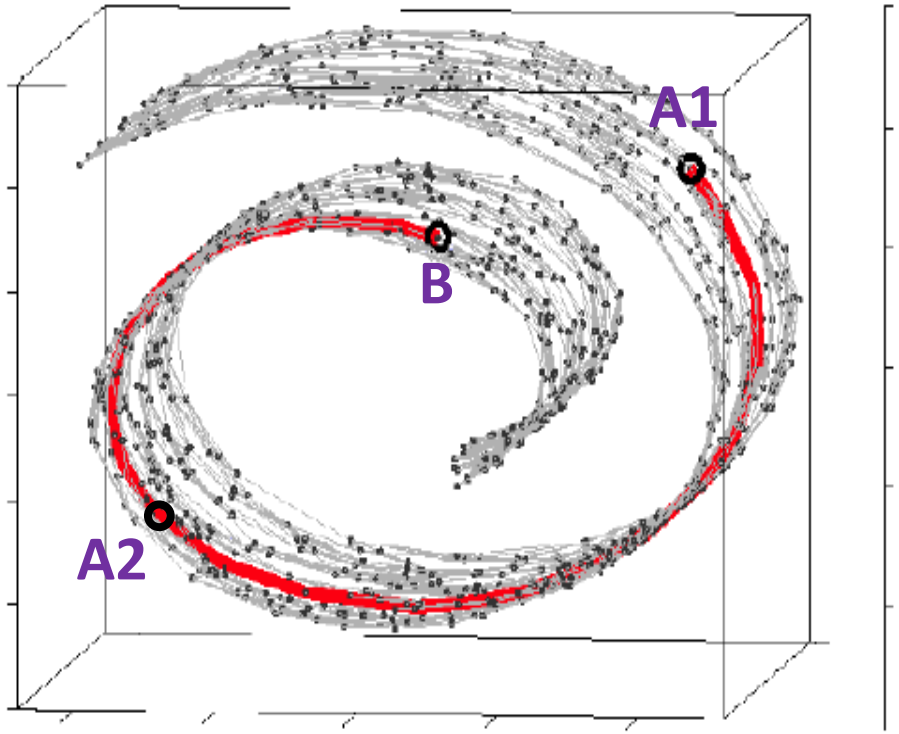
From InfoGAN

From VAE

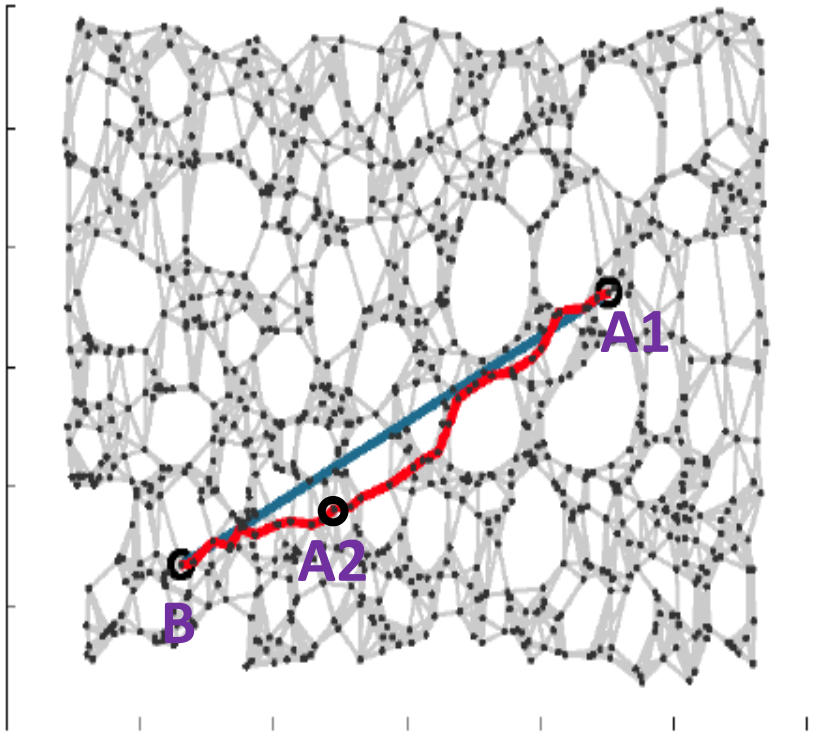
https://dmm613.wordpress.com/tag/machine-learning/

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

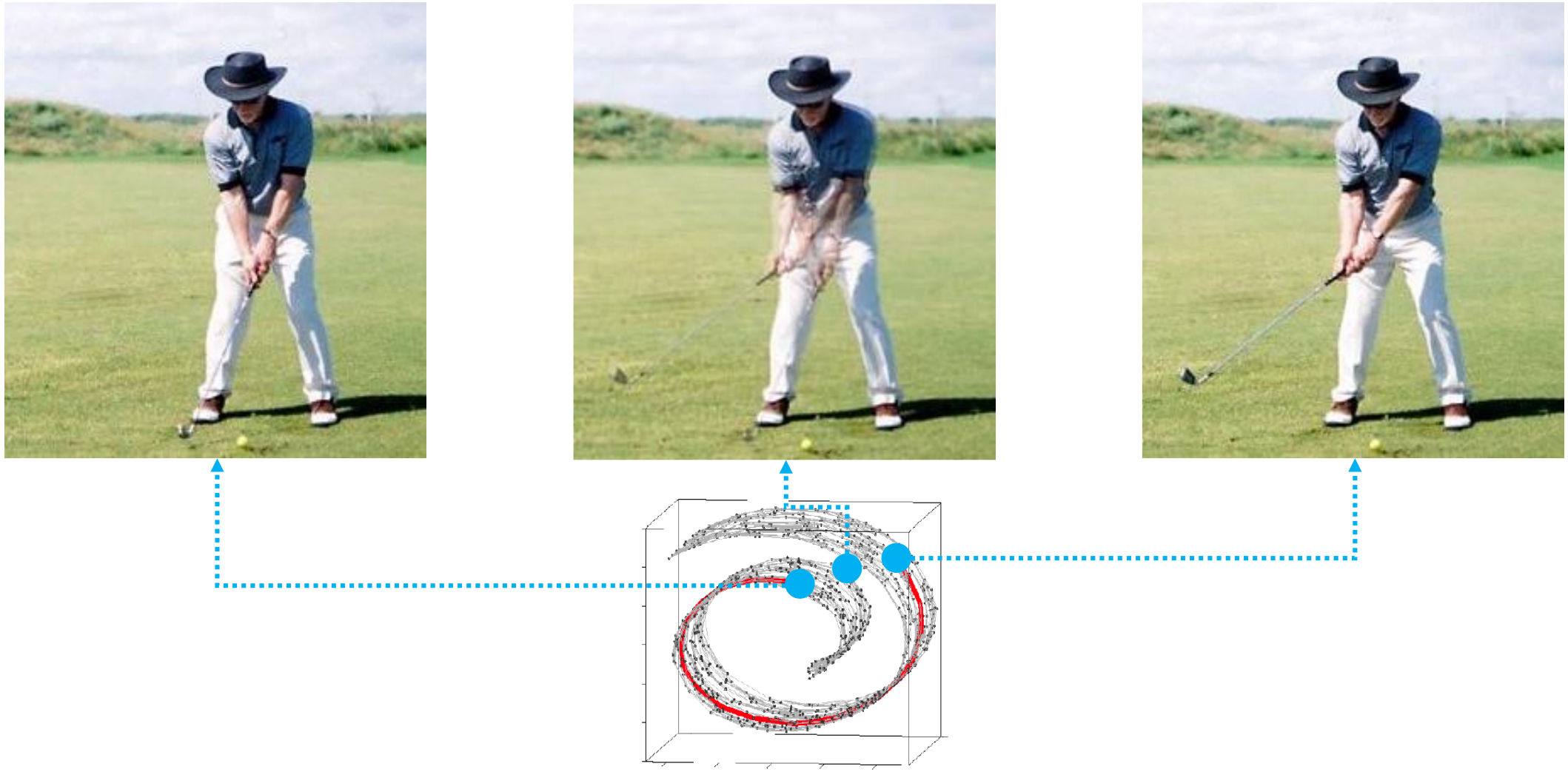


Distance in manifold

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

Reasonable distance metric

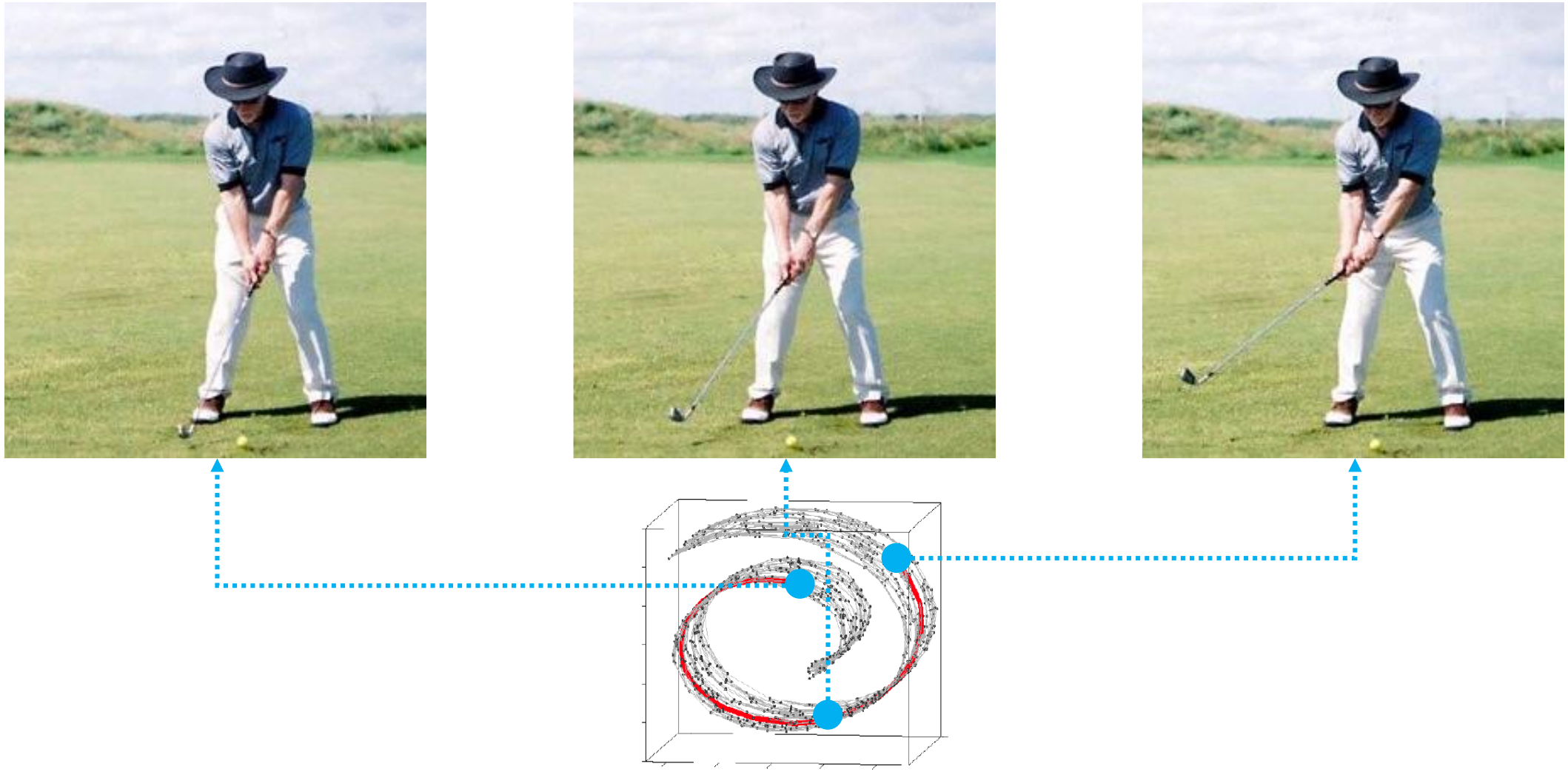
Interpolation in high dimension



https://www.cs.cmu.edu/~efros/courses/AP06/presentations/ThompsonDimensionalityReduction.pdf

Reasonable distance metric

Interpolation in manifold

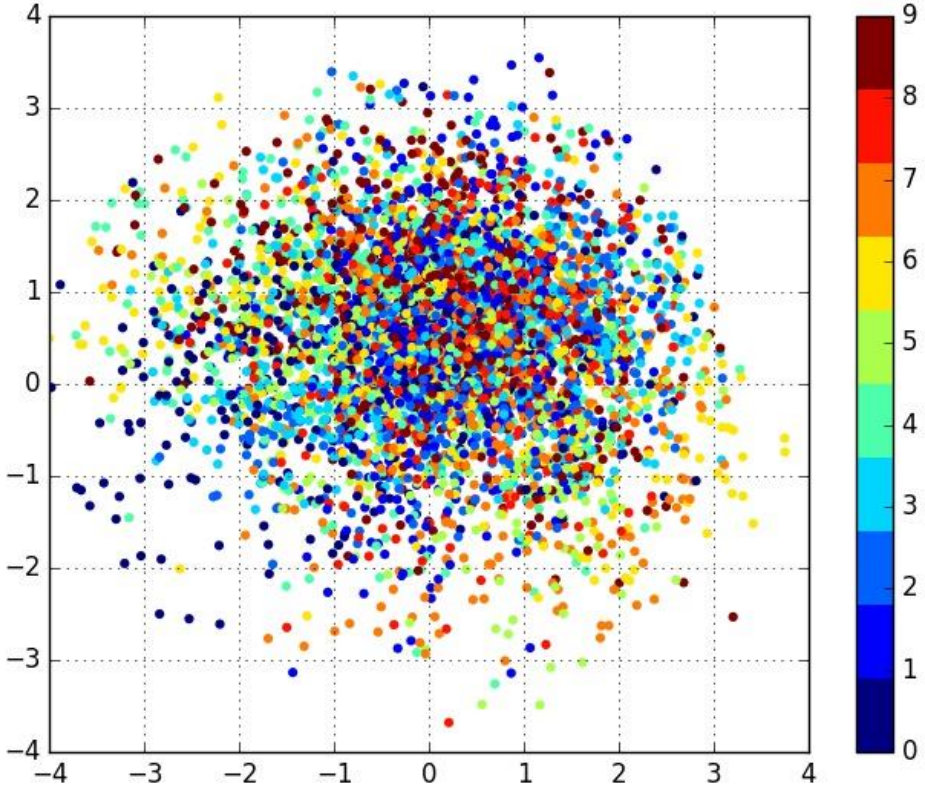


https://www.cs.cmu.edu/~efros/courses/AP06/presentations/ThompsonDimensionalityReduction.pdf

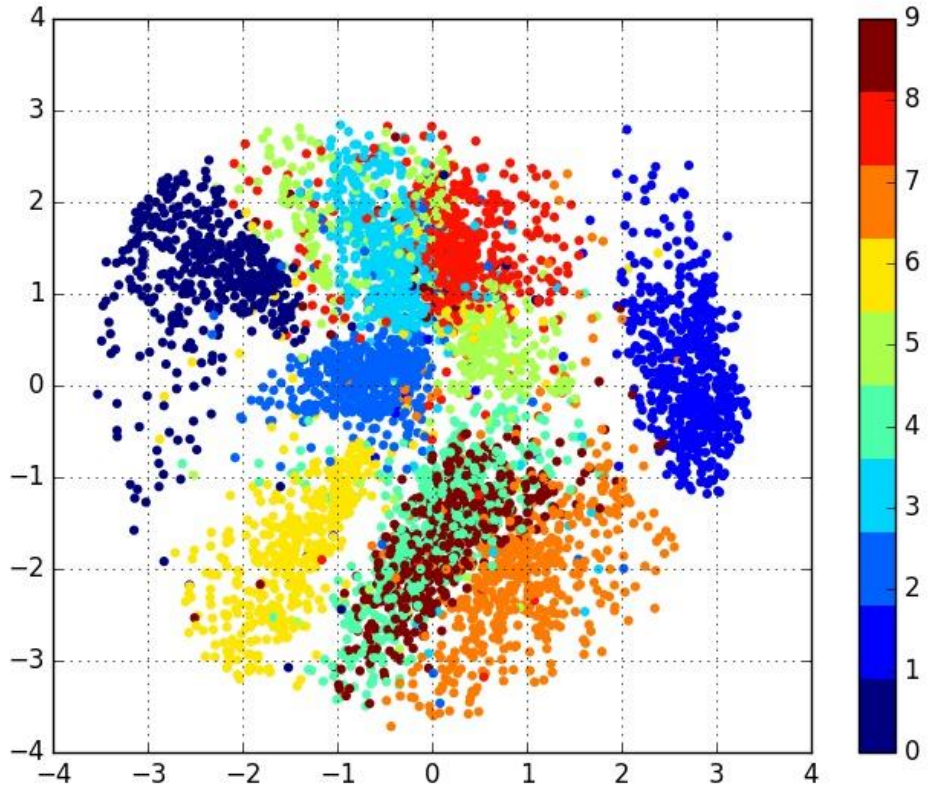
Needs disentagling the underlying explanatory factors

In general, learned manifold is entangled, i.e. encoded in a data space in a complicated manner.
When a manifold is disentangled, it would be more interpretable and easier to apply to tasks

MNIST Data → 2D manifold



Entangled manifold



Disentangled manifold

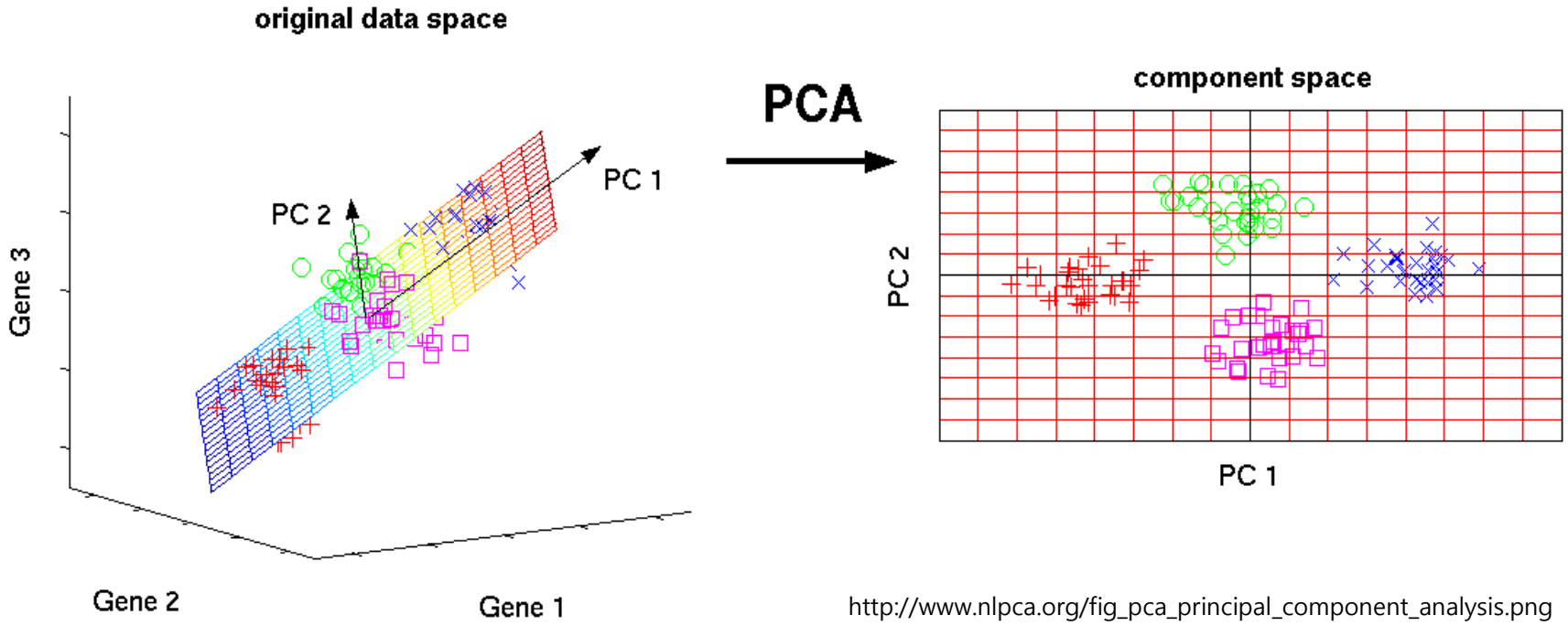
Dimensionality Reduction

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..

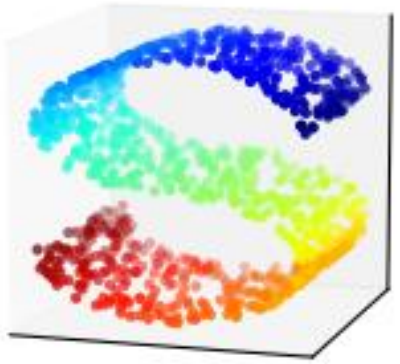


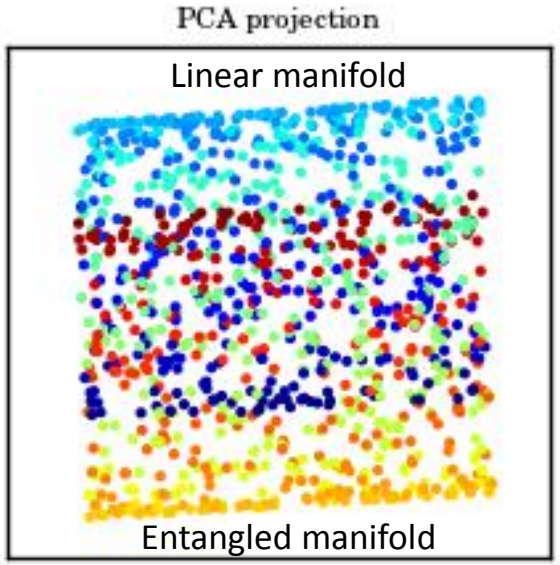
http://www.nlpca.org/fig_pca_principal_component_analysis.png

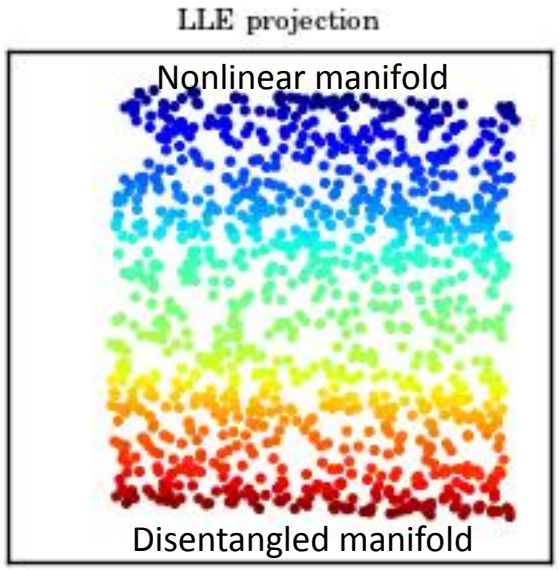
- Finds k directions in which data has highest variance
  - Principal directions (eigenvectors) W
- Projecting inputs x on these vectors yields reduced dimension representation (&decorrelated)
  - Principal components
  - $h = f_\theta(x) = W(x - \mu)\ with\ \theta = \{W, \mu\}$

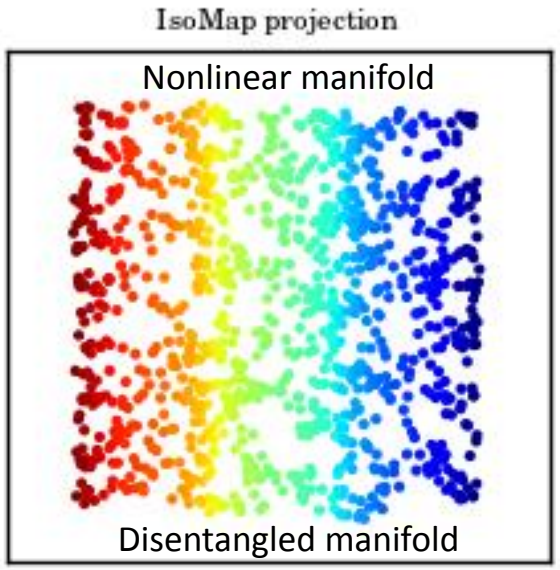
- Why mention PCA?
  - Prototypical unsupervised representation learning algorithm
  - Related to autoencoders
  - Prototypical manifold modeling algorithm

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







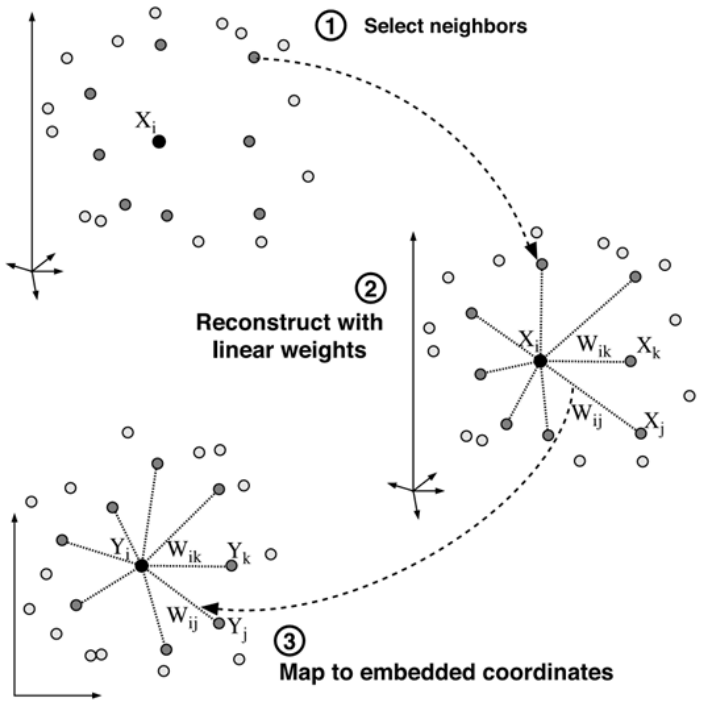


http://www.astroml.org/book_figures/chapter7/fig_S_manifold_PCA.html

Isomap

1. Construct neighborhood graph $d_x(i,j)$ using Euclidean distance
   - ϵ-Isomap: neighbors within a radius ϵ
   - K-Isomap: K nearest neighbors
2. Compute shortest path as the approximation of geodesic distance
   1. $d_G(i,j) = d_x(i,j)$
   2. For $k = 1, 2, \ldots, N$, replace all $d_G(i,j)$ by $\min\{d_G(i,j), d_G(i,k) + d_G(k,j)\}$
3. Construct d-dimensional embedding using MDS

LLE

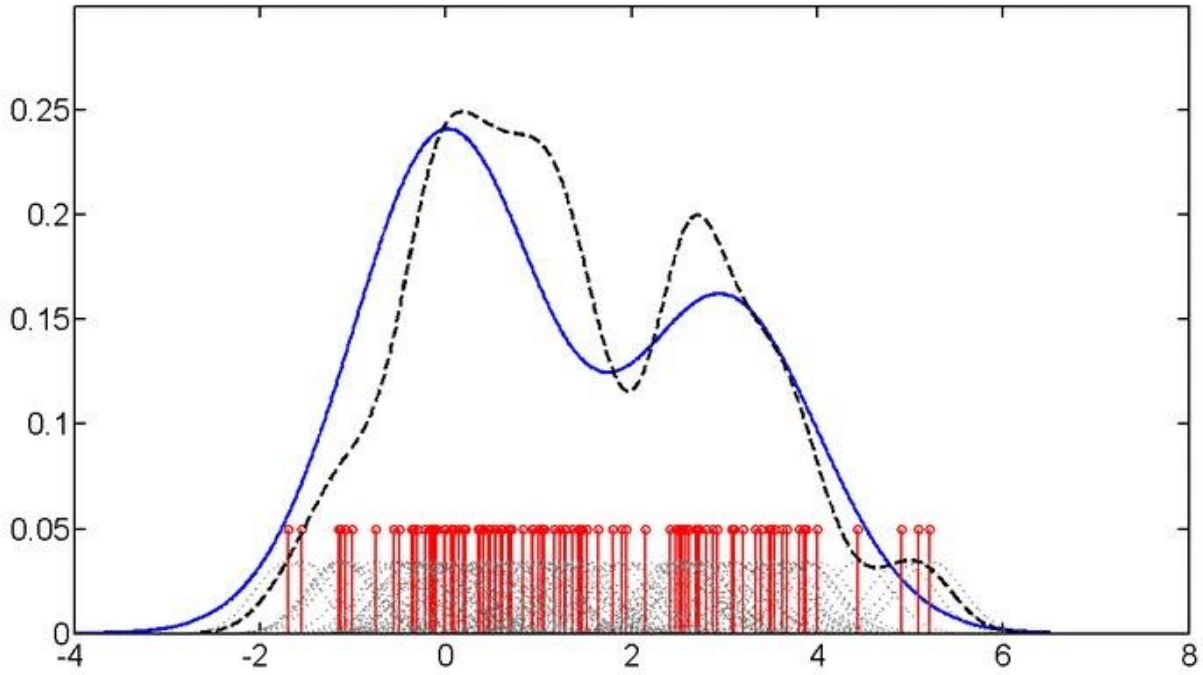


S T Roweis, and L K Saul Science 2000;290:2323-2326

1. Assign neighbours to each data point (k-NN)
2. Reconstruct each point by a weighted linear combination of its neighbors.
3. Map each point to embedded coordinates.

https://www.slideshare.net/plutoyang/manifold-learning-64891420

1D Example



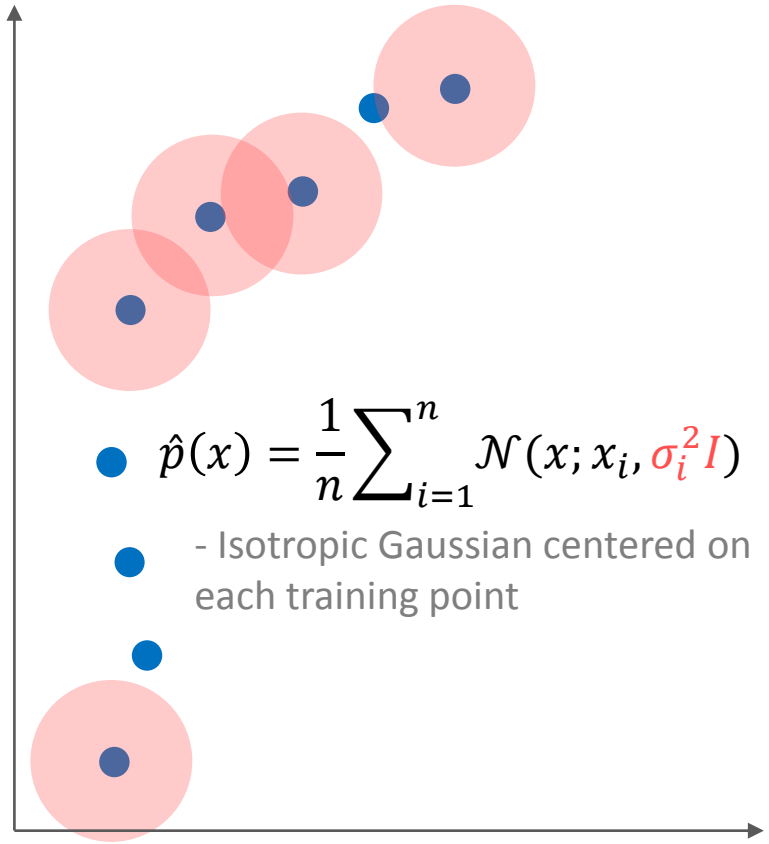
$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{N}(x; x_i, \sigma_i^2)$$

- Demonstration of density estimation using kernel smoothing
- The true density is mixture of two Gaussians centered around 0 and 3, shown with solid blue curve.
- In each frame, 100 samples are generated from the distribution, shown in red.
- Centered on each sample, a Gaussian kernel is drawn in gray.
- Averaging the Gaussians yields the density estimate shown in the dashed black curve.

https://en.wikipedia.org/wiki/Density_estimation#/media/File:KernelDensityGaussianAnimated.gif

2D Example

Classical Parzen Windows

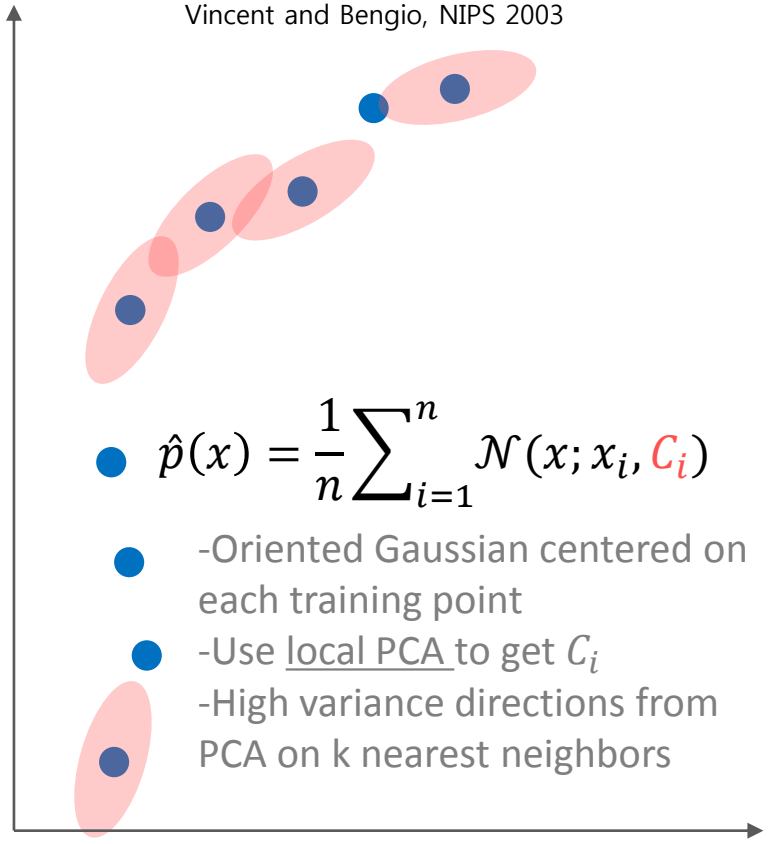


$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{N}(x; x_i, \sigma_i^2 I)$$

- Isotropic Gaussian centered on each training point

Manifold Parzen Windows

Vincent and Bengio, NIPS 2003

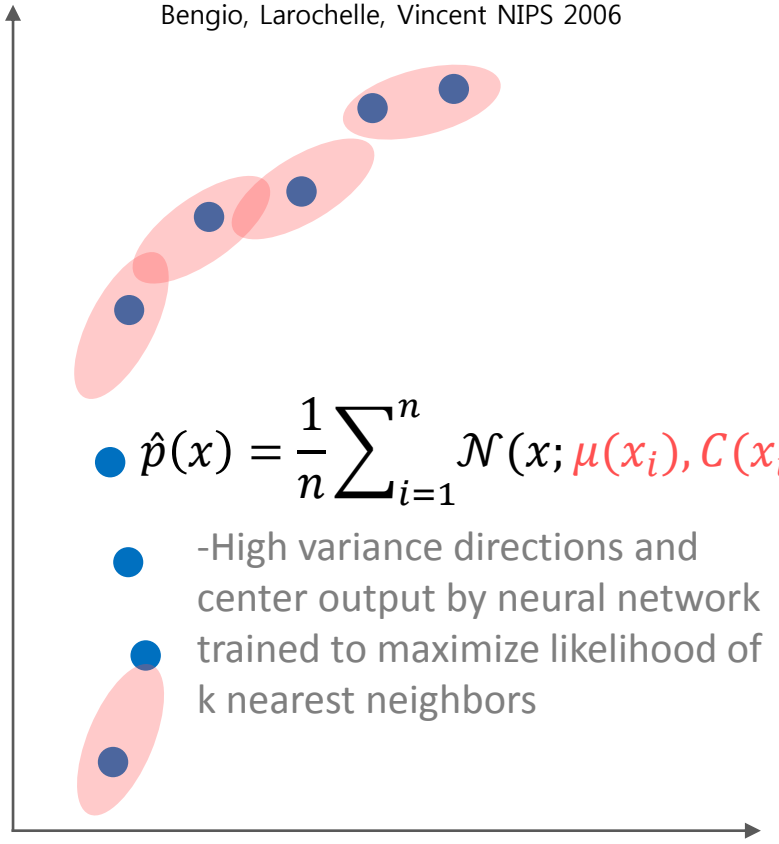


$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{N}(x; x_i, C_i)$$

-Oriented Gaussian centered on each training point
-Use local PCA to get C_i
-High variance directions from PCA on k nearest neighbors

Non-local Manifold Parzen Windows

Bengio, Larochelle, Vincent NIPS 2006



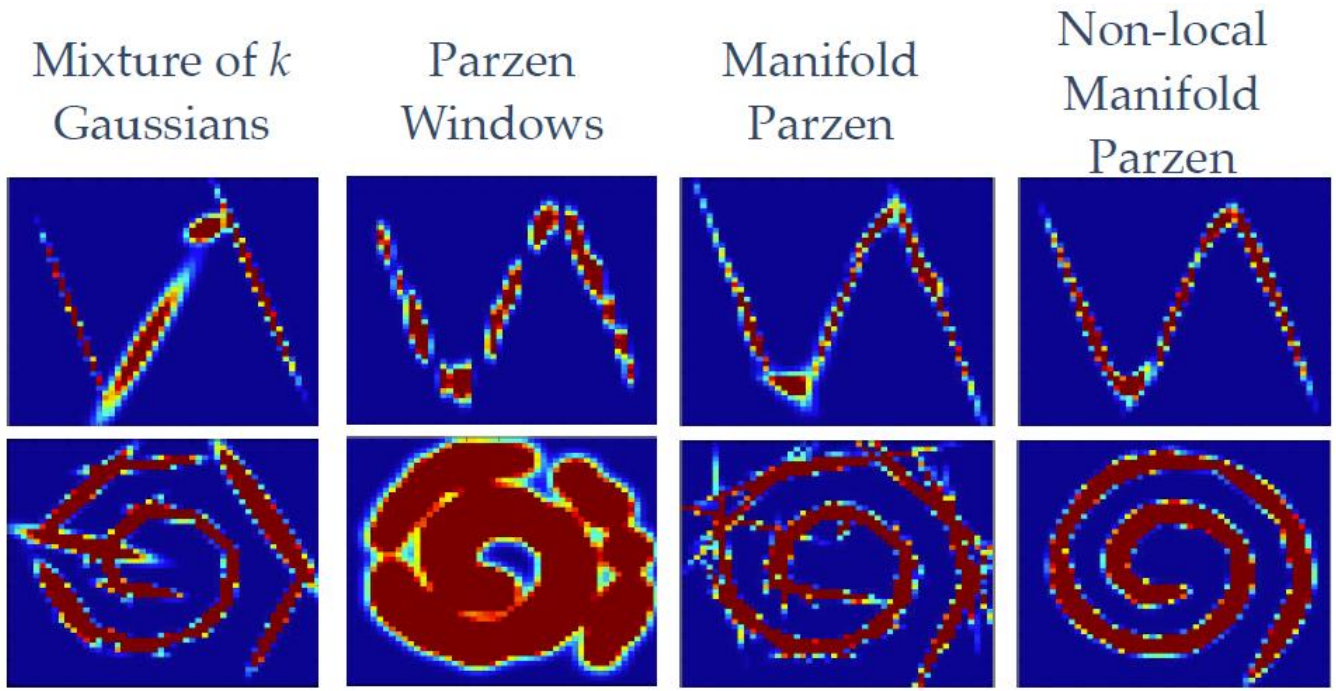
$$\hat{p}(x) = \frac{1}{n}\sum_{i=1}^{n} \mathcal{N}(x; \mu(x_i), C(x_i))$$

-High variance directions and center output by neural network trained to maximize likelihood of k nearest neighbors

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

DENSITY ESTIMATION

Parzen WIndows



Use in Bayes classifier on USPS

Algorithm	Valid.	Test	Hyper-Parameters
SVM	1.2%	4.68%	$C = 100, \sigma = 8$
Parzen Windows	1.8%	5.08%	$\sigma = 0.8$
Manifold Parzen	0.9%	4.08%	$d = 11, k = 11, \sigma_0^2 = 0.1$
Non-local MP	**0.6%**	**3.64% (-1.5218)**	$d = 7, k = 10, k_\mu = 10,$ $\sigma_0^2 = 0.05, n_{hid} = 70$
Non-local MP*	**0.6%**	**3.54% (-1.9771)**	$d = 7, k = 10, k_\mu = 4,$ $\sigma_0^2 = 0.05, n_{hid} = 30$

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

LIMITATION

Dimensionality Reduction

- Isomap
- Locally-linear embedding (LLE)

Non-parametric Density Estimation

- Isotropic parzen window
- Manifold parzen window
- Non-local manifold parzen window

Neighborhood based training !!!

- 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.

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

01. Revisit Deep Neural Networks

02. Manifold Learning

03. Autoencoders

- Autoencoder (AE)
- Denosing AE (DAE)
- Contractive AE (CAE)

Autoencoder를 설명하고 이와 유사해 보이는 PCA, RBM과의 차이점을 설명한다.

Autoencoder의 입력에 Stochastic perturbation을 추가한 Denoising Autoencoder, perturbation을 analytic regularization term으로 바꾼 Contractive Autoencoder에 대해서 설명한다.

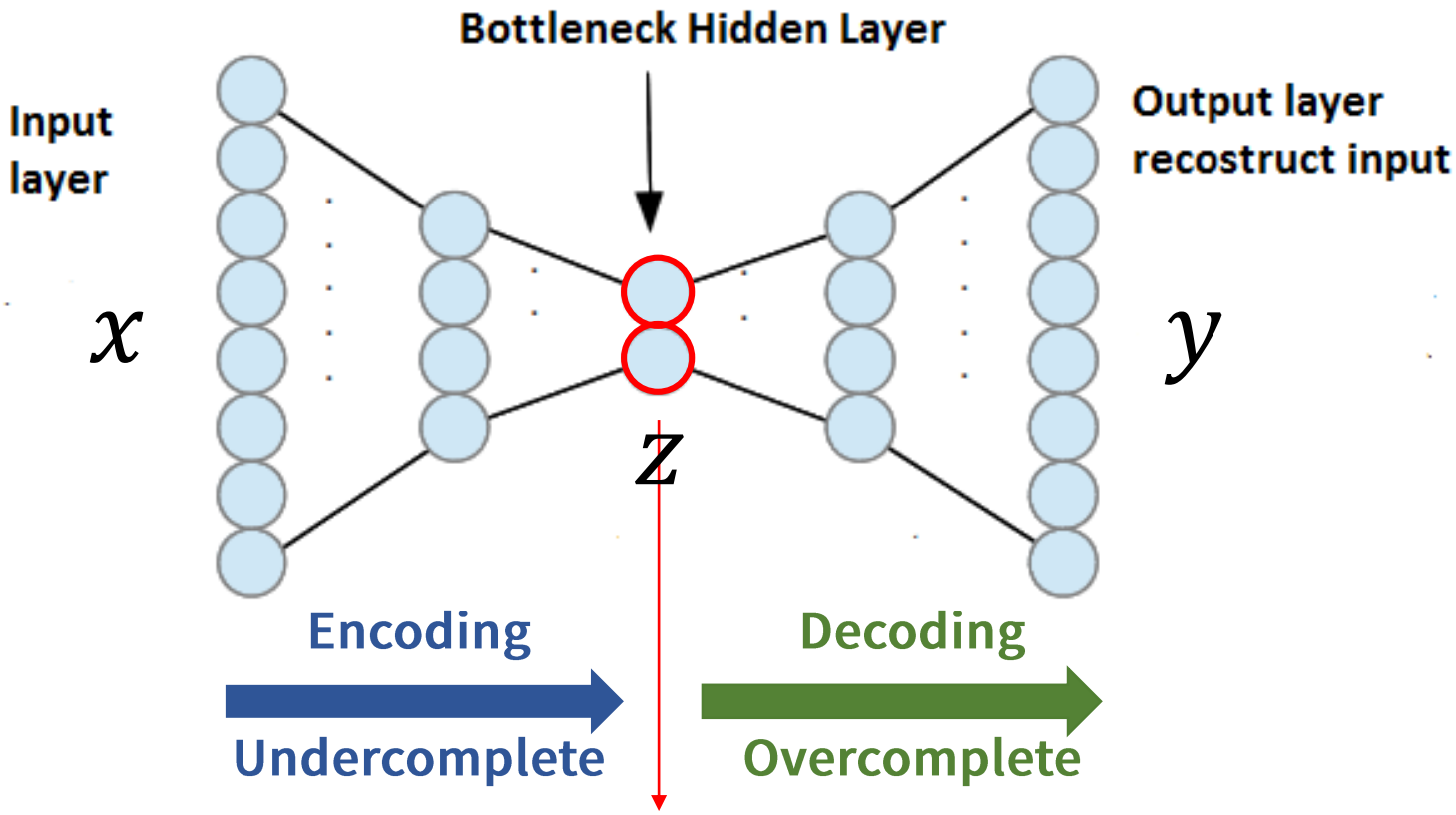
04. Variational Autoencoders

05. Applications

Autoencoders

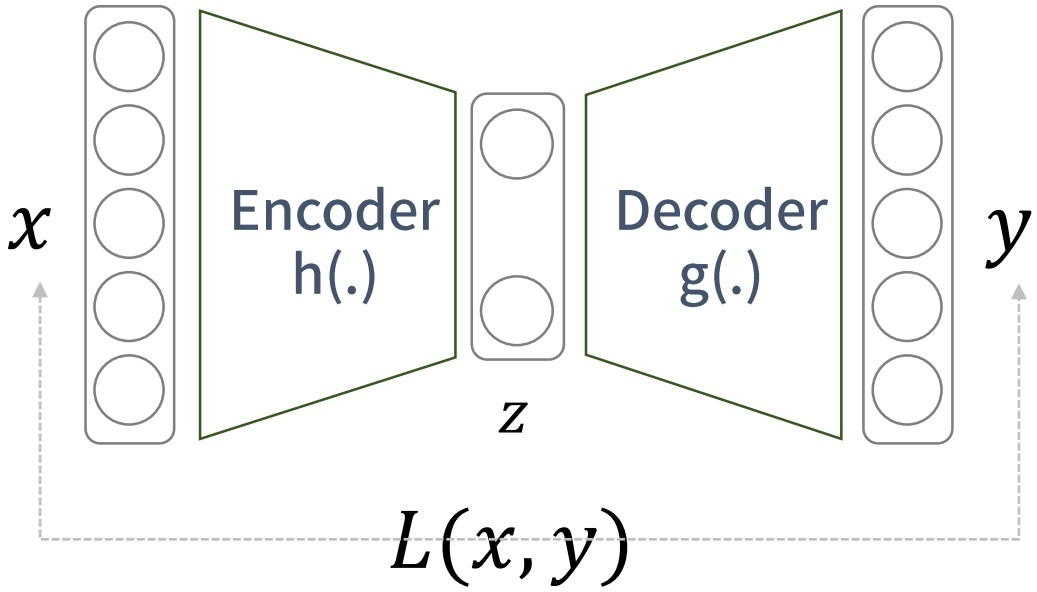
= Auto-associators
= Diabolo networks
= Sandglass-shaped net

Diabolo



- Code
- Latent Variable
- Feature
- Hidden representation

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



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

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

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

- Make output layer same size as input layer

$$x, y \in \mathbb{R}^d$$

- Loss encourages output to be close to input

$$L(x, y)$$

입출력이 동일한 네트워크

- Unsupervised Learning → Supervised Learning

비교사 학습 문제를 교사 학습 문제로 바꾸어서 해결

Decoder가 최소한 학습 데이터는 생성해 낼 수 있게 된다.
→ 생성된 데이터가 학습 데이터 좀 닮아 있다.

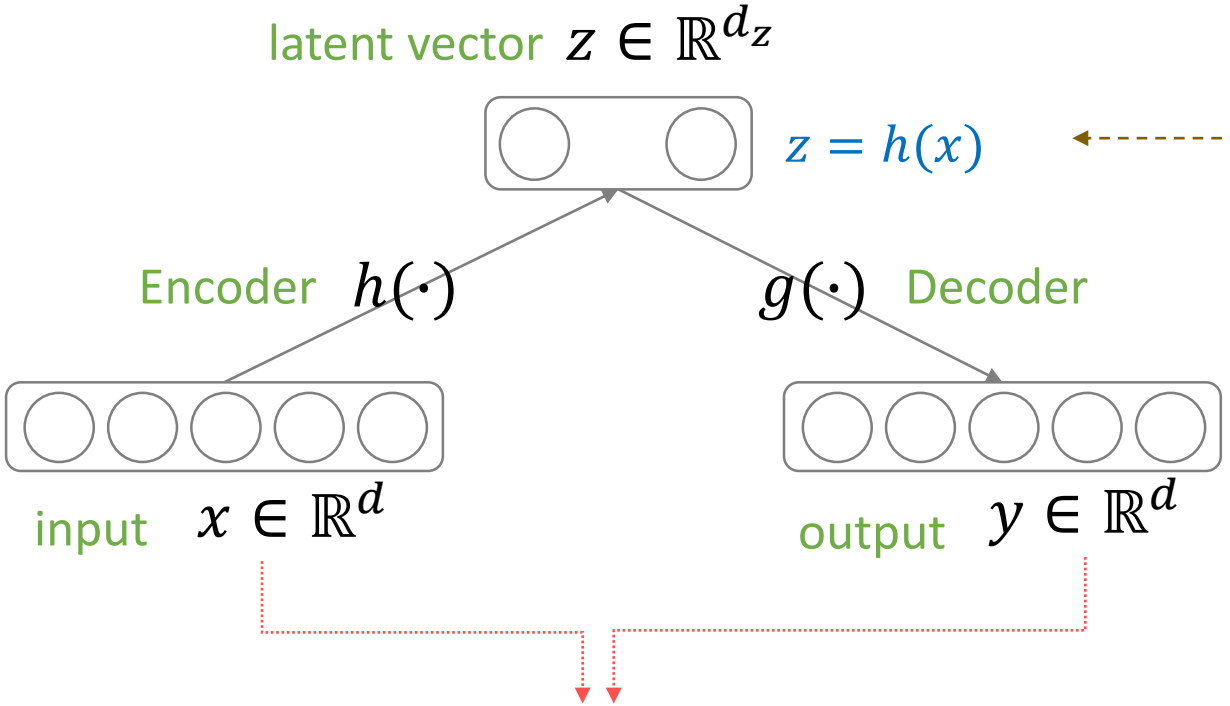
Encoder가 최소한 학습 데이터는 잘 latent vector로 표현할 수 있게 된다.
→ 데이터의 추상화를 위해 많이 사용된다.

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

LINEAR AUTOENCODER

Multi-Layer Perceptron

General Autoencoder



Minimize $L_{AE} = \sum_{x \in D} L(x, g(h(x)))$

Linear Autoencoder

$h(x) = W_e x + b_e$

$g(h(x)) = W_d z + b_d$

$\|x - y\|^2$ or cross-entropy

Hidden layer 1개이고 레이어 간 fully-connected로 연결된 구조

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

LINEAR AUTOENCODER | Connection to PCA & RBM

Principle Component Analysis

- For bottleneck structure : $d_z < d$
- With linear neurons and squared loss, autoencoder learns same subspace as PCA

$h = f_\theta(x) = W(x - \mu)\ with\ \theta = \{W, \mu\}$ in PCA Slide

- Also true with a single sigmoidal hidden layer, if using linear output neurons with squared loss and untied weights.
- Won't learn the exact same basis as PCA, but W will span the same subspace.

Baldi, Pierre, & Hornik, Kurt. 1989. Neural networks and principal component analysis: Learning from examples without local minima. Neural networks, 2(1), 53–58.

Restricted Boltzman Machine

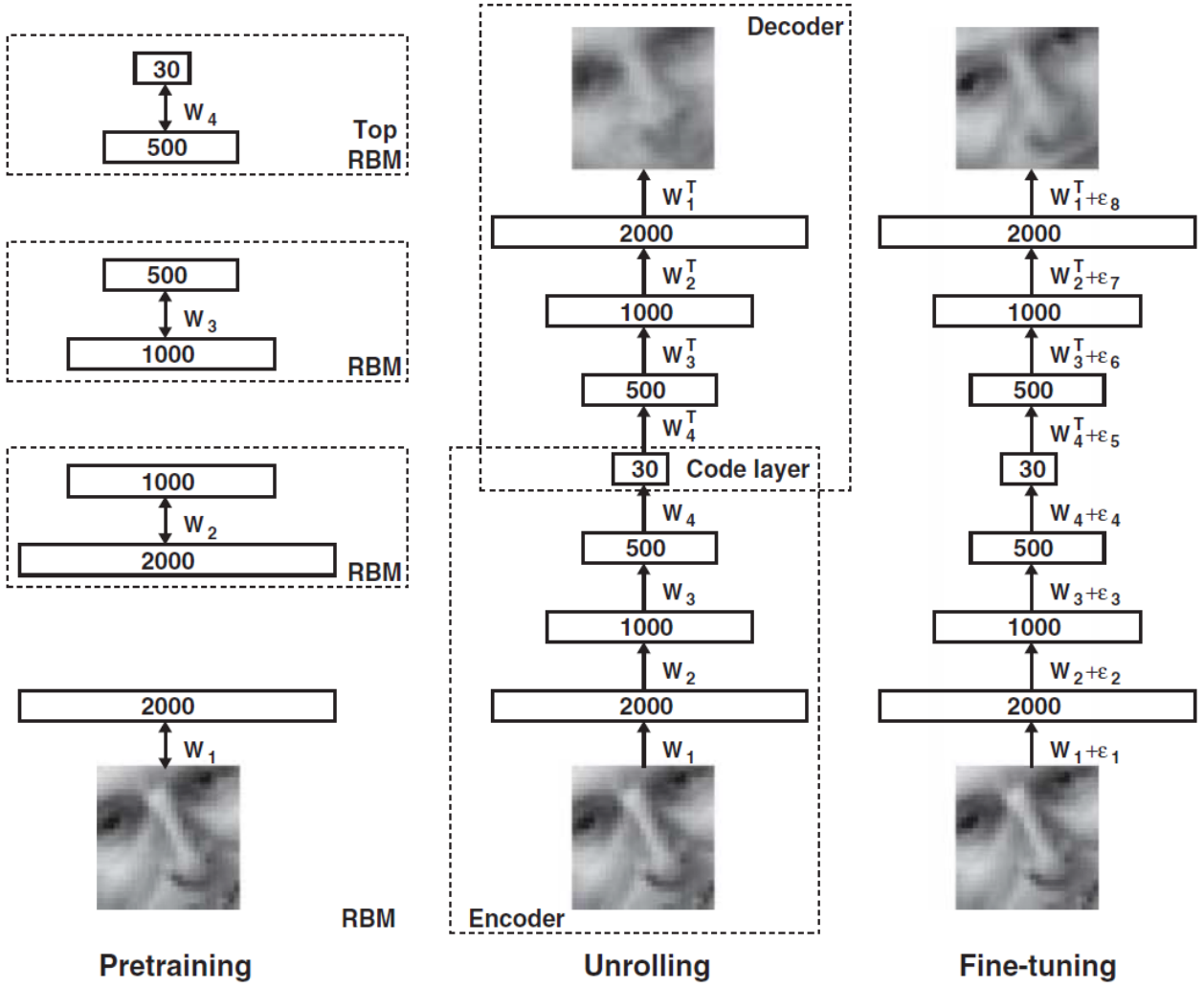
- With a single hidden layer with sigmoid non-linearity and sigmoid output non-linearity.
- Tie encoder and decoder weights: $W_d = W_e^T$

Autoencoder	*RBM*
$z_i = \sigma(W_{ei}x + b_{ei})$	$P(h_i = 1\|v) = \sigma(W_{ei}v + b_{ei})$
$y_j = \sigma(W_{ej}^T z + b_{dj})$	$P(v_j = 1\|h) = \sigma(W_{ej}^T h + b_{dj})$
Determinisitc mapping z is a function x	Stochastic mapping z is a random variable

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

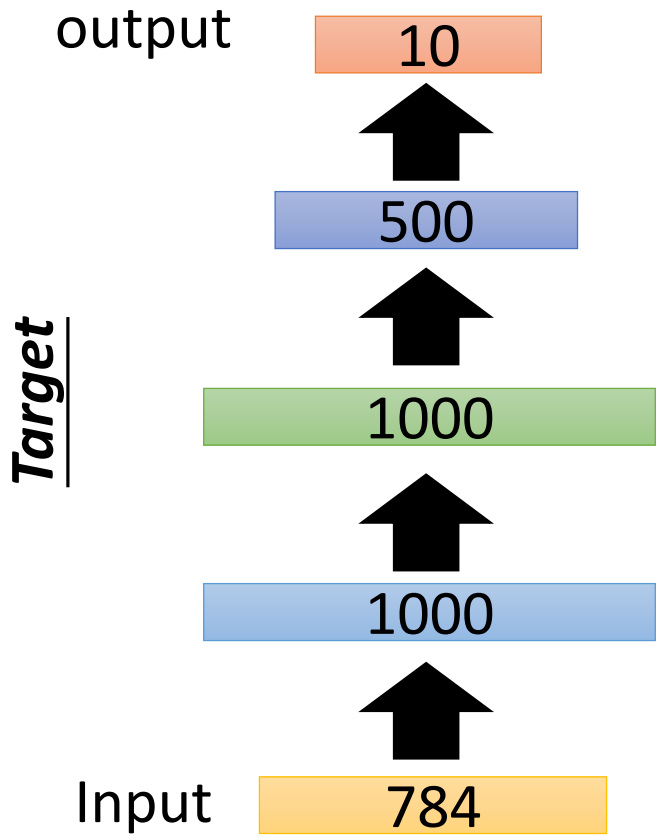
Stacking RBM → Deep Belief Network (DBN)

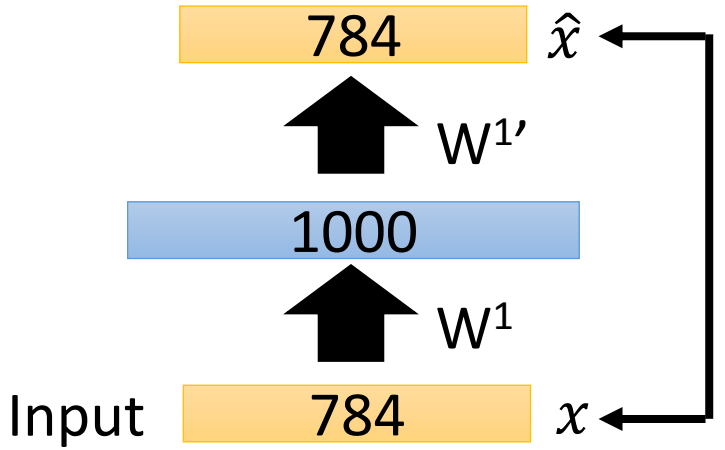
Reducing the Dimensionality of Data with Neural Networks



https://www.cs.toronto.edu/~hinton/science.pdf

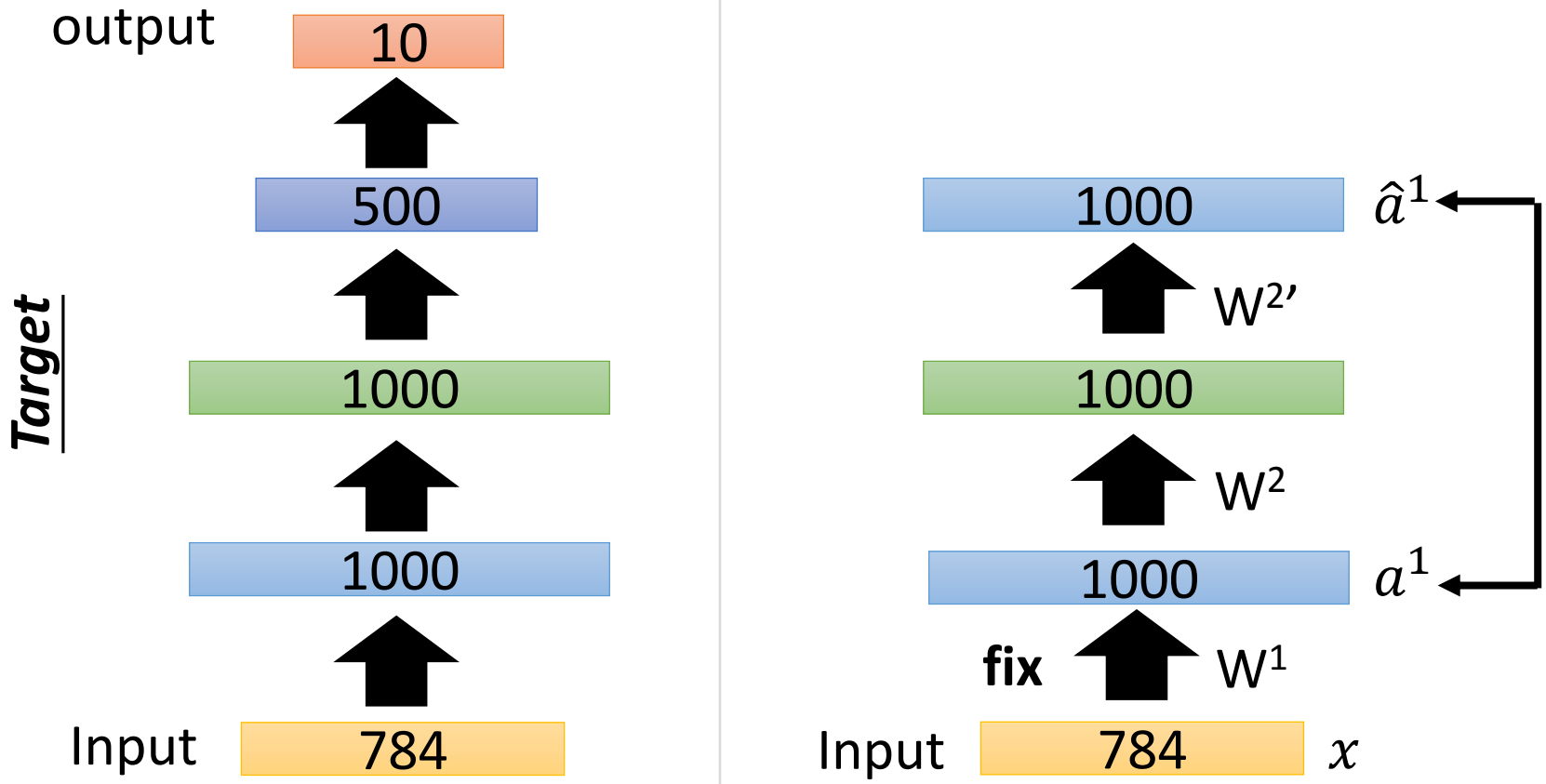
Stacking Autoencoder



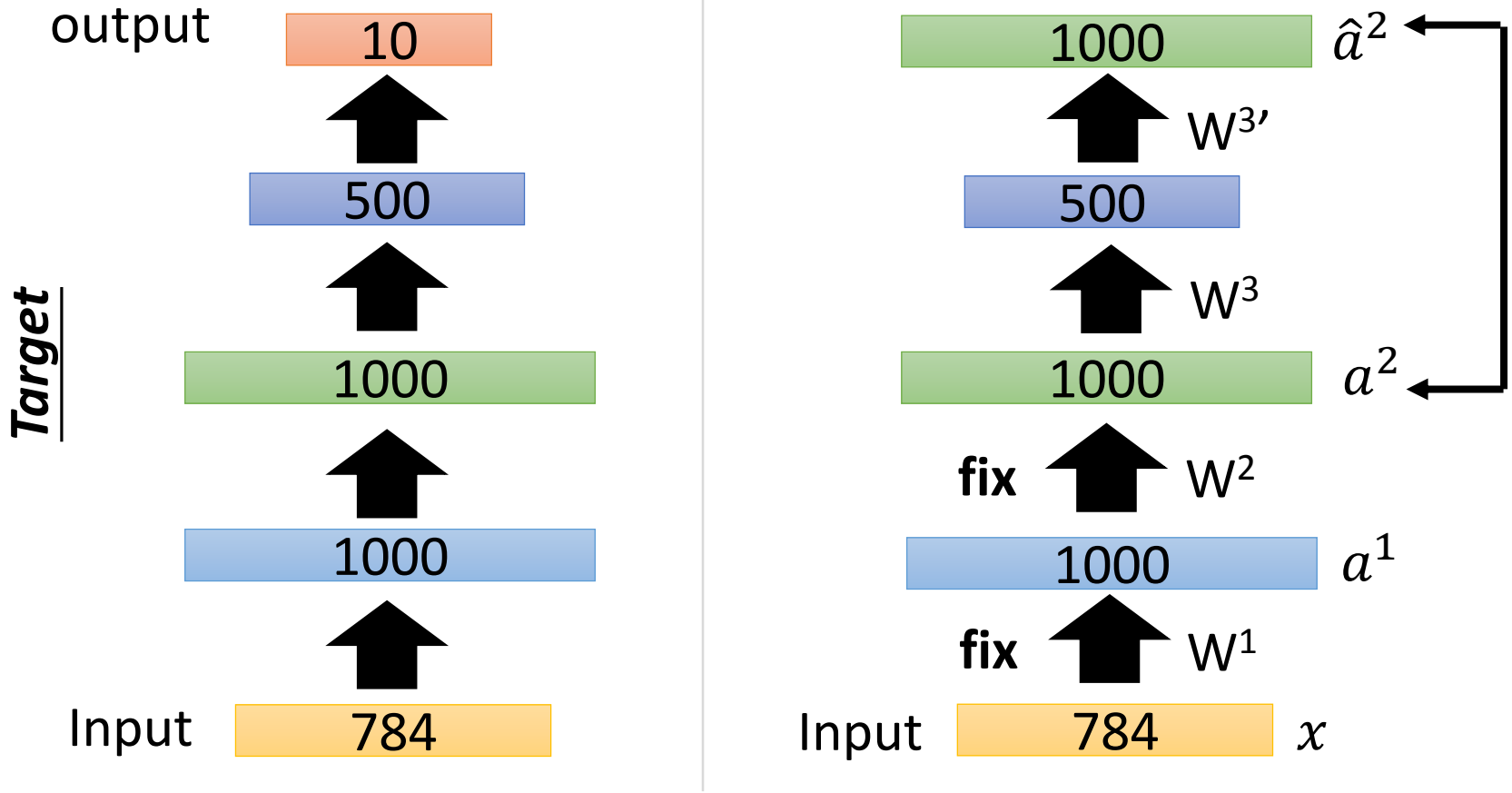


http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/auto.pptx

Stacking Autoencoder

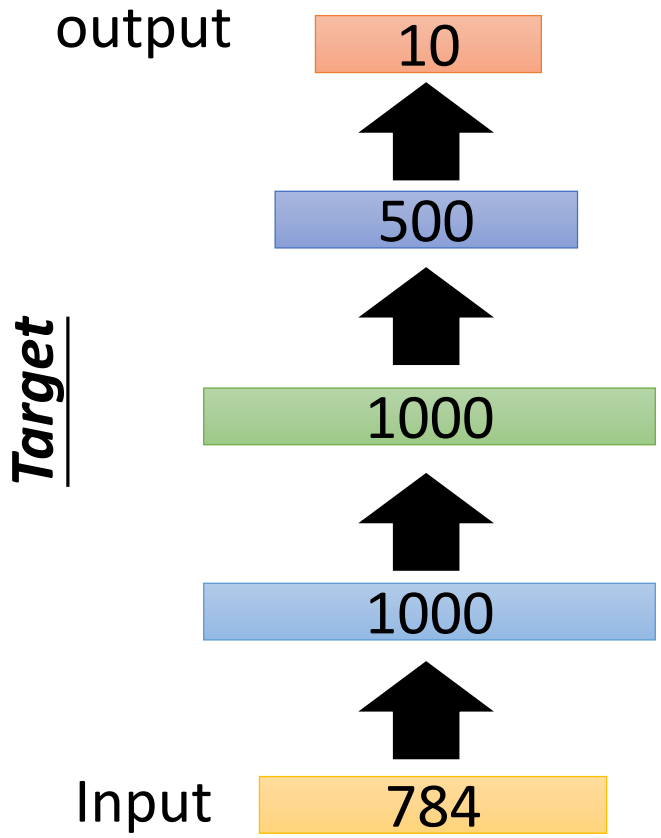


Stacking Autoencoder

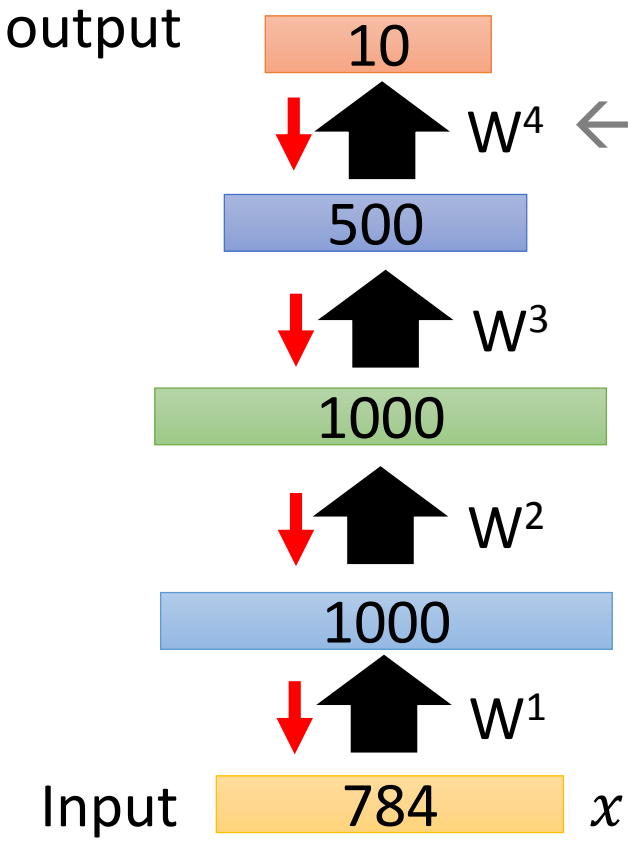


http://speech.ee.ntu.edu.tw/~tlkagk/courses/ML_2017/Lecture/auto.pptx

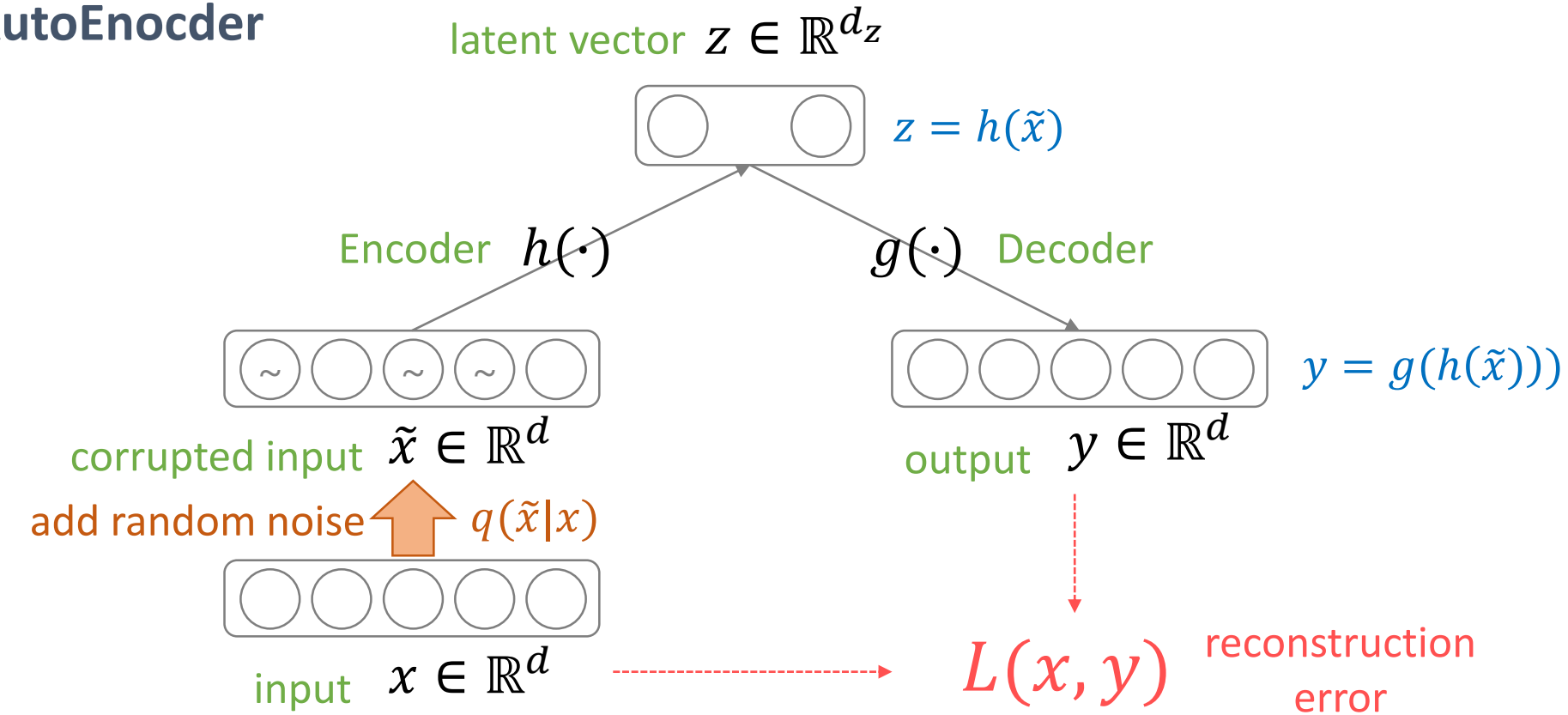
Stacking Autoencoder



Fine-tuning by backpropagation



Denoising AutoEnocder



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

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

DAE | Adding noise

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

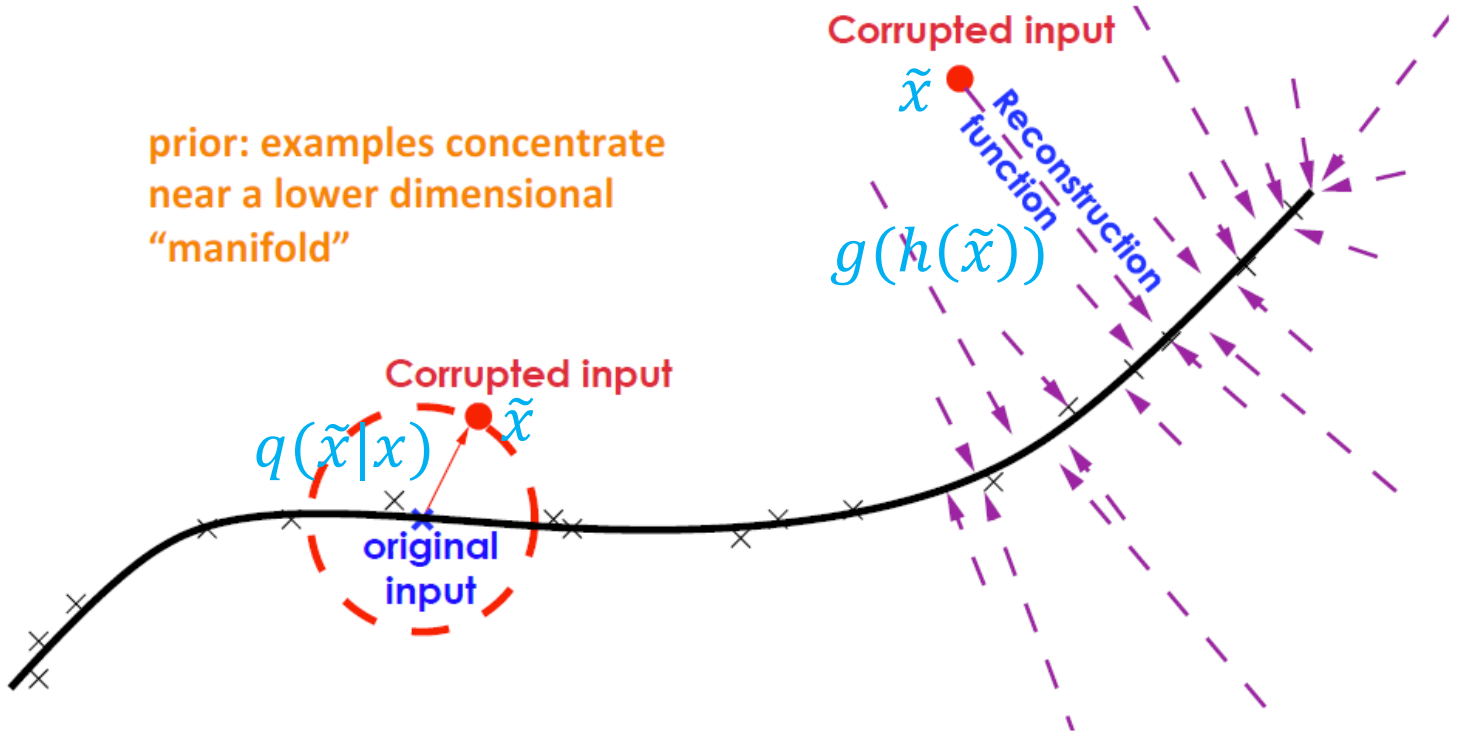
Cannot compute expectation exactly

- Use sampling corrupted inputs

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

L개 샘플에 대한 평균으로 대체

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



- 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.

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

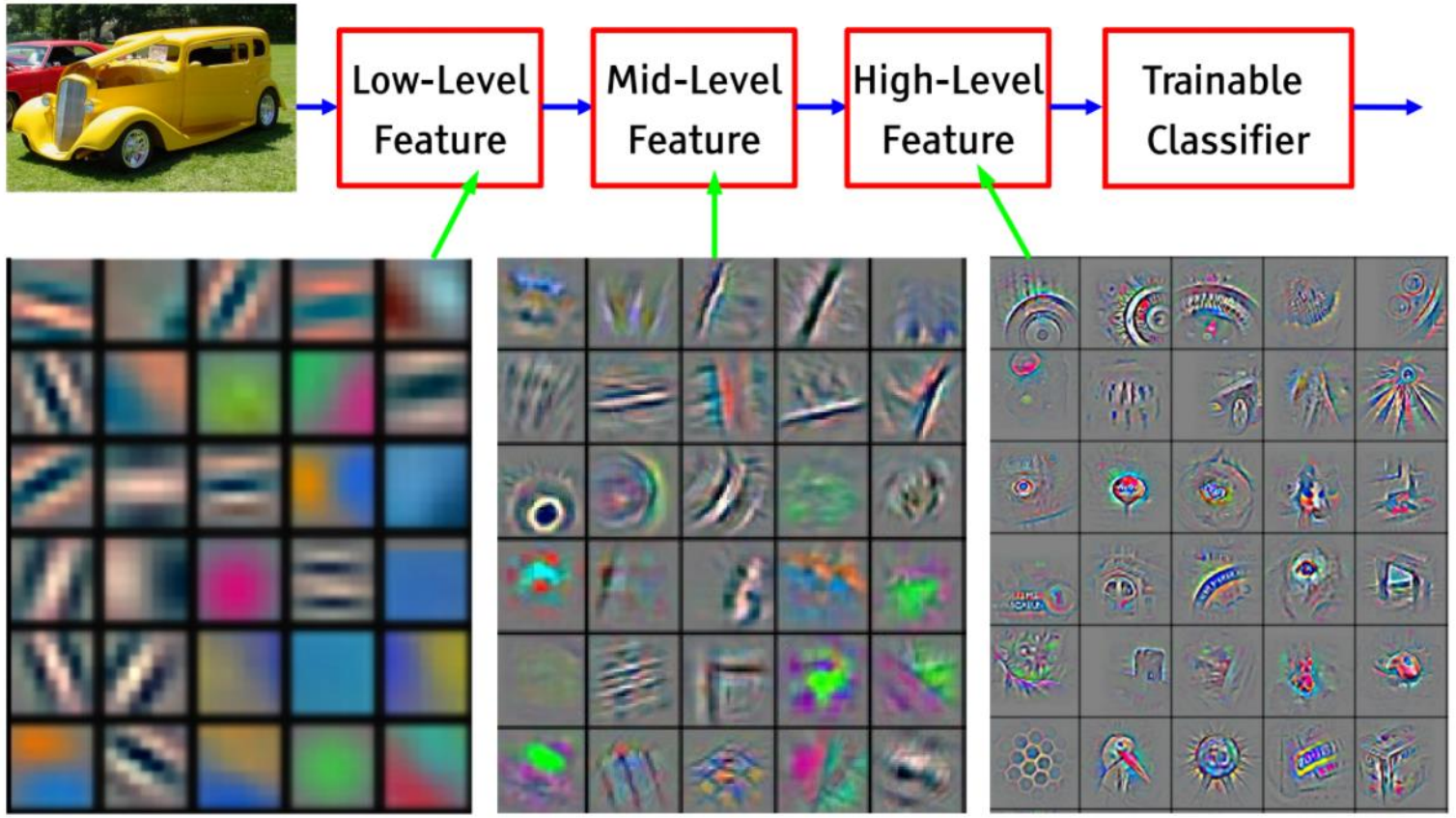
Filters in 1 hidden architecture must capture low-level features of images.

$$y = \sigma_d(W^T z + b_d)$$

$$z = \sigma_e(Wx + b_e)$$

x

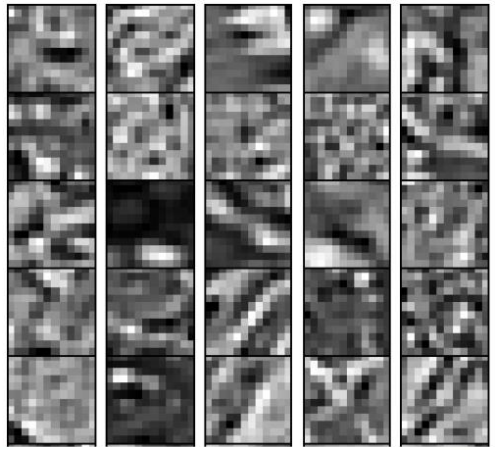
AutoEncoder
with 1hidden layer



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

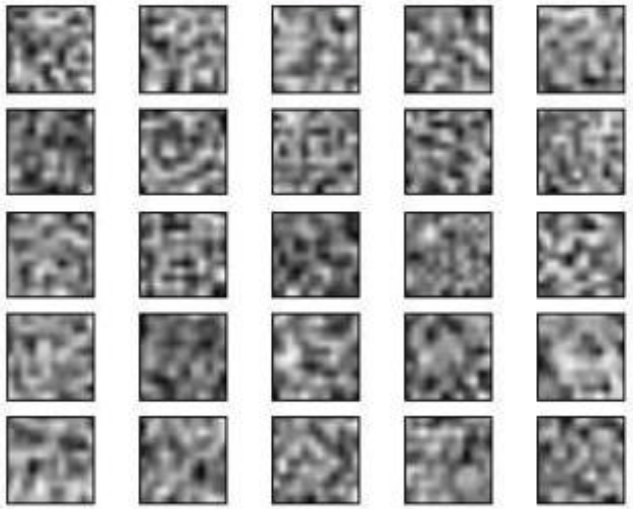
Natural image patches (12x12 pixels) : 100 hidden units

랜덤값으로 초기화하였기 때문에 노이즈처럼 보이는 필터일 수록 학습이 잘 안 된 것이고 edge filter와 같은 모습일 수록 학습이 잘 된 것이다.

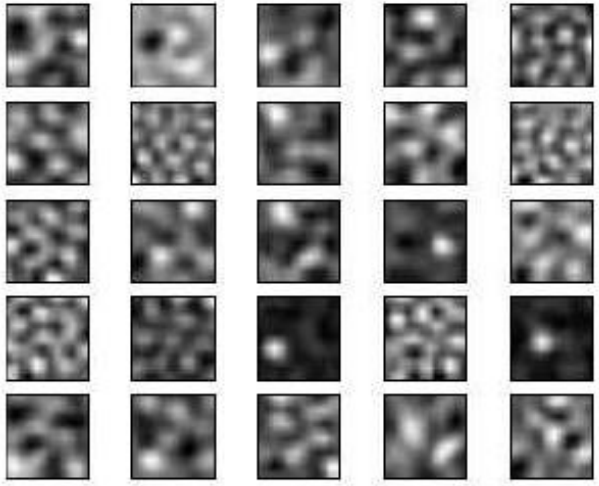


- Mean Squared Error
- 100 hidden units
- Salt-and-pepper noise

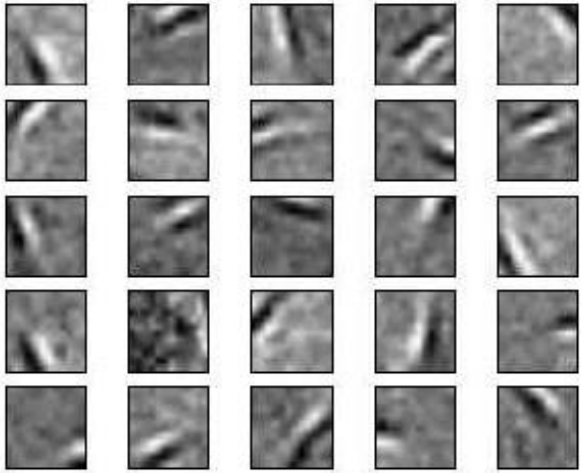
AE



AE with weight decay



DAE



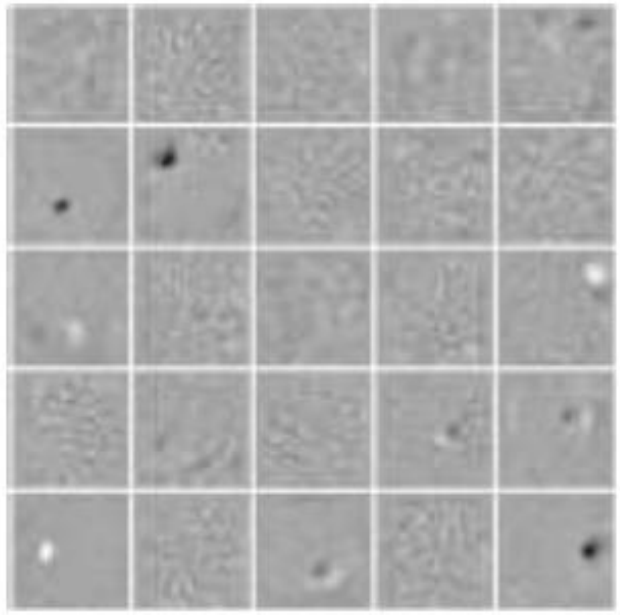
10% salt-and-pepper noise

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

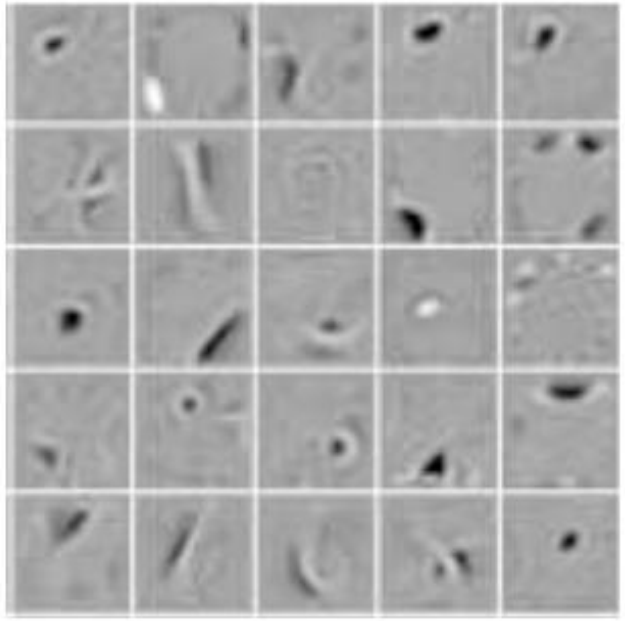
MNIST digits (64x64 pixels)

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

AE

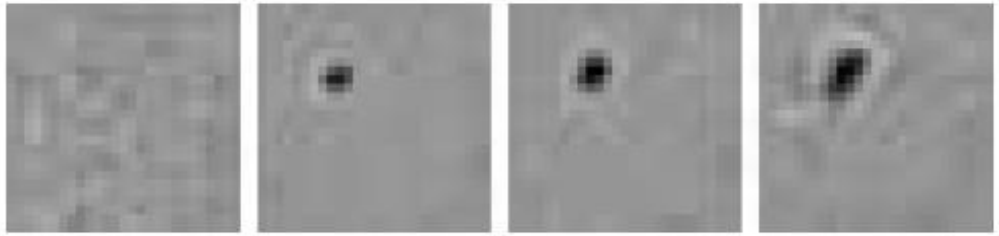


DAE



25% corruption

Increasing noise

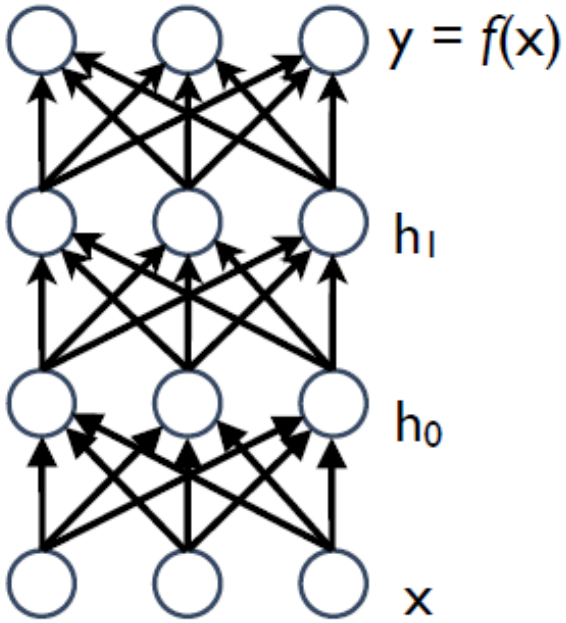


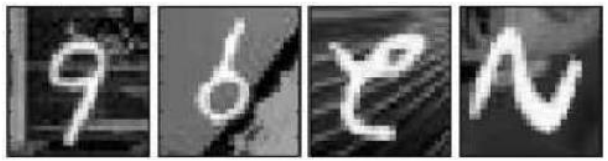
(d) Neuron A (0%, 10%, 20%, 50% corruption)

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

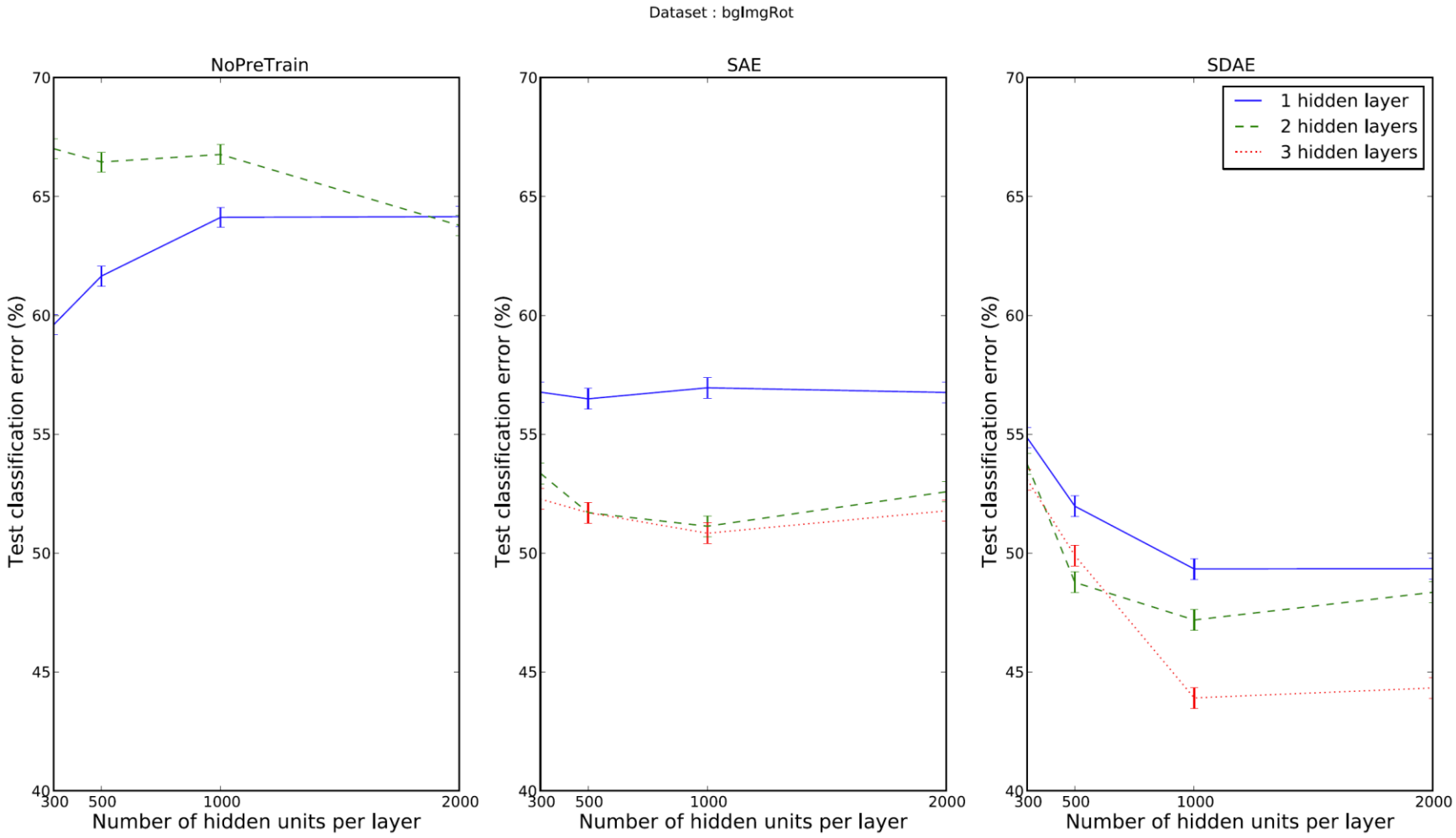
http://videolectures.net/deeplearning2015_vincent_autoencoders/?q=vincent%20autoencoder

Stacked Denoising Auto-Encoders (SDAE)



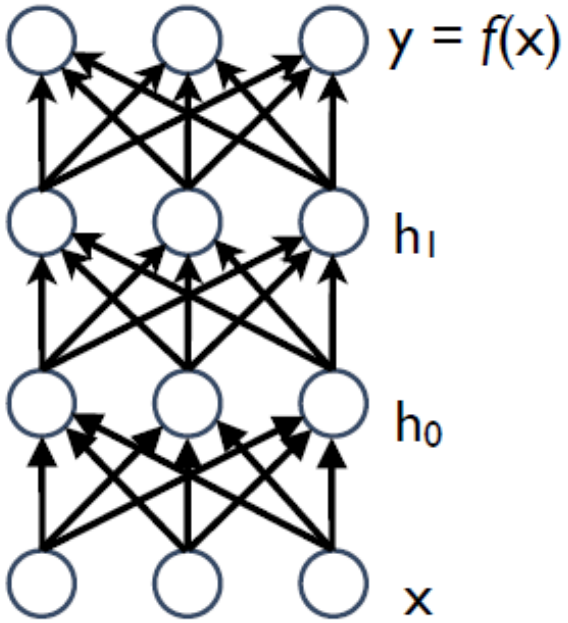


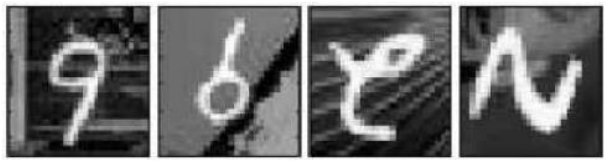
bgImgRot Data
Train/Valid/Test : 10k/2k/20k



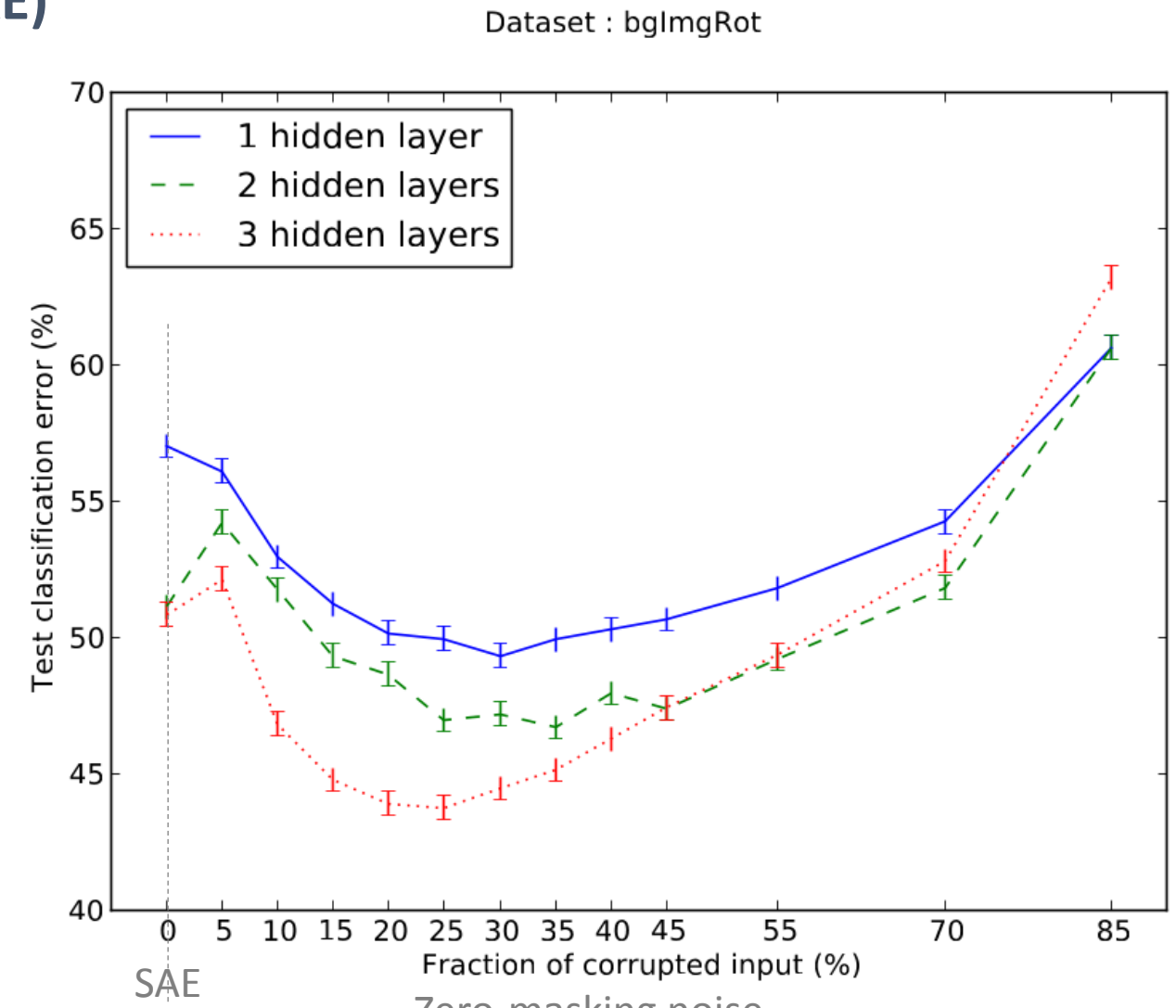
Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion

Stacked Denoising Auto-Encoders (SDAE)

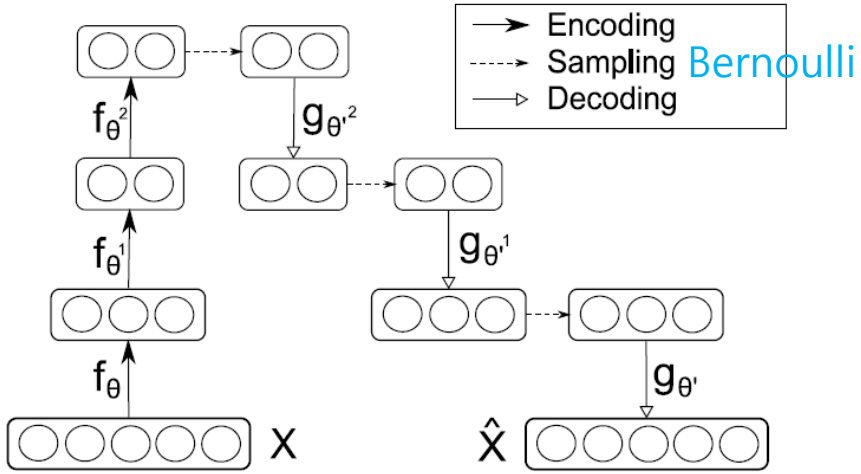


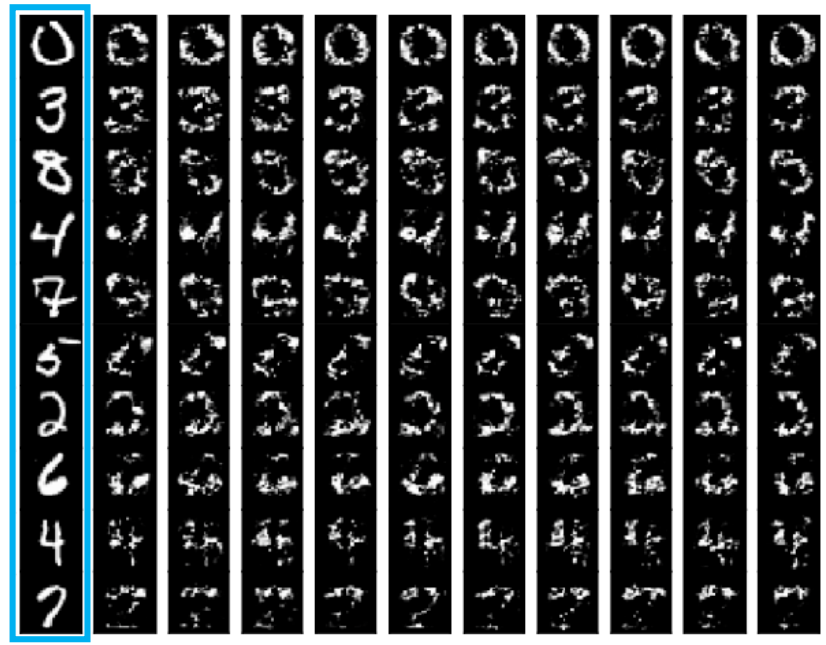


bgImgRot Data
Train/Valid/Test : 10k/2k/20k



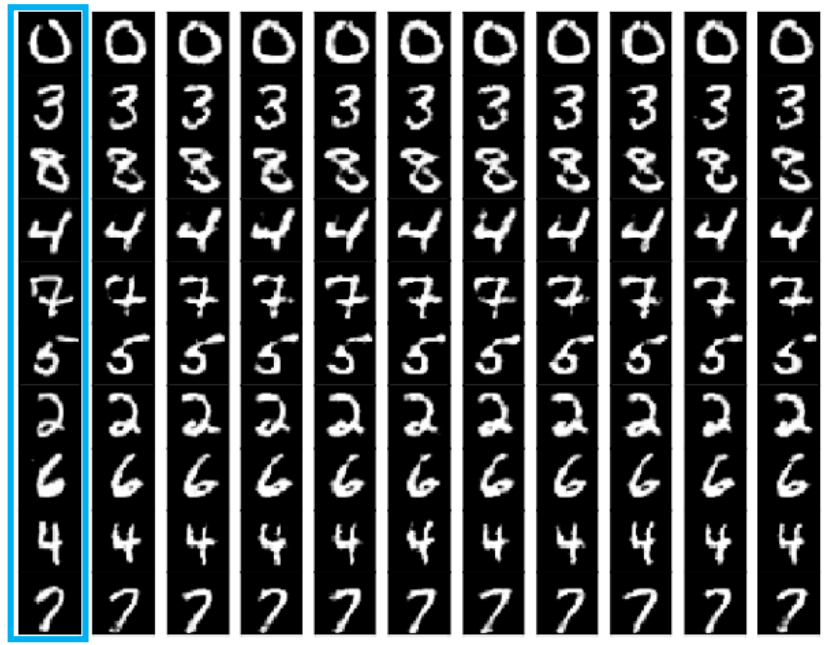
Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion





input

(a) SAE



input

(b) SDAE

Stacked Denoising Autoencoders: Learning Useful Representations in a Deep Network with a Local Denoising Criterion

Stochastic Contractive AutoEncoder (SCAE)

DAE encourages reconstruction to be insensitive to input corruption

DAE의 loss의 해석은 g,h중에서 특히 h는 데이터 x가 조금 바뀌더라도
매니폴드 위에서 같은 샘플로 매칭이 되도록 학습 되어야 한다라고 볼 수 있다.

Alternative: encourage representation to be insensitive

$$L_{SCAE} = \sum_{x \in D} \underbrace{L(x, g(h(x))}_{\text{Reconstruction Error}} + \lambda \underbrace{E_{q(\tilde{x}|x)} \left[\|h(x) - h(\tilde{x})\|^2 \right]}_{\text{Stochastic Regularization}}$$

Tied weights i.e. $W' = W^T$ prevent W from collapsing h to 0.

정규화 항목이 0이 되는 경우는 h가 0인 경우도 있으므로 이를 방지하기 위해 tied weight를 사용한다.

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

Contractive AutoEncoder (CAE)

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

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

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

$$L_{SCAE} = \sum_{x \in D} \underbrace{L(x, g(h(x))}_{\text{Reconstruction Error}} + \lambda \underbrace{\left\| \frac{\partial h}{\partial x}(x) \right\|_F^2}_{\text{Analytic Contractive Regularization}}$$

For training examples, encourages both:

- small reconstruction error
- representation insensitive to small variations around example

Analytic contractive regularization term is

- Frobenius norm of the Jacobian of the non-linear mapping

$$\left\| \frac{\partial h}{\partial x}(x) \right\|_F^2 = \sum_{ij} \left(\frac{\partial z_j}{\partial x_i}(x) \right)^2 = \|J_h(x)\|_F^2$$

Penalizing $\|J_h(x)\|_F^2$ encourages the mapping to the feature space to be contractive in the neighborhood of the training data.

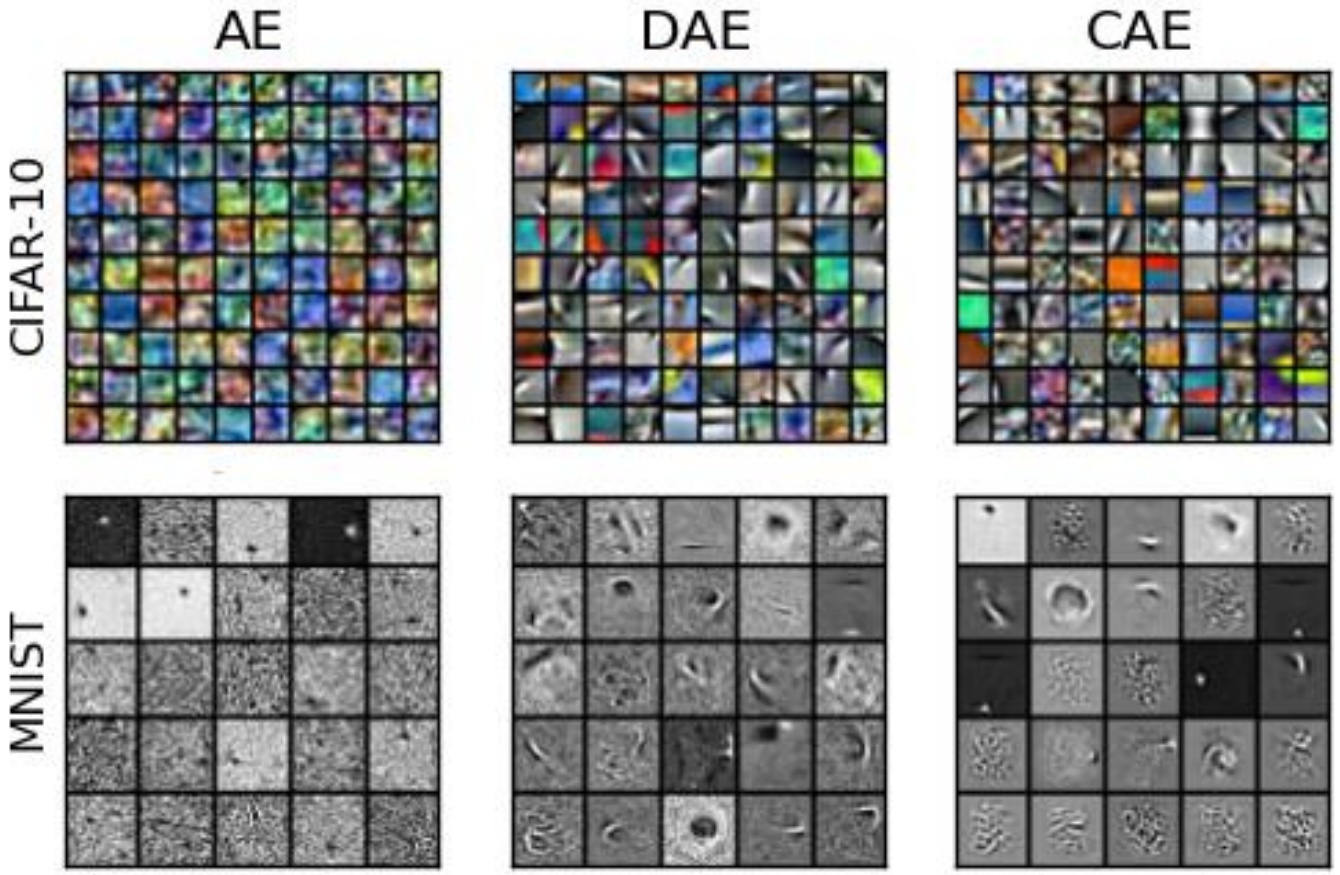
highlights the advantages for representations to be locally invariant in many directions of change of the raw input.

Contractive Auto-Encoders:Explicit Invariance During Feature Extraction

CIFAR-10 (32x32 pixels)
- 4000 hidden units

MNIST digits (64x64 pixels)
- 2000 hidden units

Gaussian noise



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

	Model	Test error	Average $\|J_f(x)\|_F$	SAT
MNIST	CAE	**1.14**	$0.73\ 10^{-4}$	86.36%
	DAE-g	**1.18**	$0.86\ 10^{-4}$	17.77%
	RBM-binary	1.30	$2.50\ 10^{-4}$	78.59%
	DAE-b	1.57	$7.87\ 10^{-4}$	68.19%
	AE+wd	1.68	$5.00\ 10^{-4}$	12.97%
	AE	1.78	$17.5\ 10^{-4}$	49.90%
CIFAR-bw	CAE	**47.86**	$2.40\ 10^{-5}$	85,65%
	DAE-b	**49.03**	$4.85\ 10^{-5}$	80,66%
	DAE-g	54.81	$4.94\ 10^{-5}$	19,90%
	AE+wd	55.03	$34.9\ 10^{-5}$	23,04%
	AE	55.47	$44.9\ 10^{-5}$	22,57%

- DAE-g : DAE with gaussian noise
- DAE-b : DAE with binary masking noise
- CIFAR-bw : gray scale version
- Training/Validation/test : 10k/2k/50k
- SAT : average fraction of saturated units per sample
- 1-hidden layer with 1000 units

Data Set	SVM_{rbf}	SAE-3	RBM-3	DAE-b-3	CAE-1	CAE-2
basic	3.03±0.15	3.46±0.16	3.11±0.15	2.84±0.15	2.83±0.15	**2.48**±0.14
rot	11.11±0.28	10.30±0.27	10.30±0.27	**9.53**±0.26	11.59±0.28	**9.66**±0.26
bg-rand	14.58±0.31	11.28±0.28	**6.73**±0.22	10.30±0.27	13.57±0.30	10.90 ±0.27
bg-img	22.61±0.379	23.00±0.37	16.31±0.32	16.68±0.33	16.70±0.33	**15.50**±0.32
bg-img-rot	55.18±0.44	51.93±0.44	47.39±0.44	**43.76**±0.43	48.10±0.44	45.23±0.44
rect	2.15±0.13	2.41±0.13	2.60±0.14	1.99±0.12	1.48±0.10	**1.21**±0.10
rect-img	24.04±0.37	24.05±0.37	22.50±0.37	**21.59**±0.36	**21.86**±0.36	**21.54**±0.36

- basic: smaller subset of MNIST
- rot: digits with added random rotation
- bg-rand: digits with random noise background
- bg-img: digits with random image background
- bg-img-rot: digits with rotation and image background
- rect: discriminate between tall and wide rectangles (white on black)
- rect-img: discriminate between tall and wide rectangular image on a different background image

http://www.iro.umontreal.ca/~lisa/icml2007
Contractive Auto-Encoders: Explicit Invariance During Feature Extraction

01. Revisit Deep Neural Networks

02. Manifold Learning

03. Autoencoders

04. Variational Autoencoders

05. Applications

- Variational AE (VAE)
- Conditional VAE (CVAE)
- Adversarial AE (AAE)

KEYWORD: Generative model learning

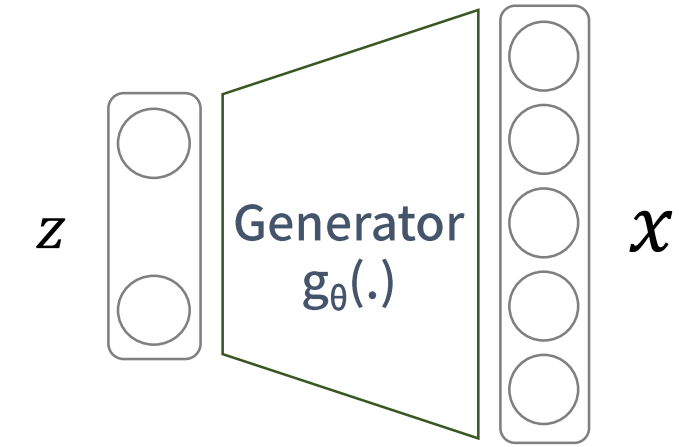
생성 모델로서의 Variational Autoencoder를 설명한다.

이에 생성 결과물을 조절하기 위한 조건을 추가한 Conditional Variational Autoencoder를 설명한다.

또한 일반 prior distributio에 대해서 동작 가능한 Adversarial Autoencder를 설명한다.

GENERATIVE MODEL

Latent Variable Model



$z \sim p(z)$ 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:

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

Question: Is it enough to model p(z) with simple distribution like normal distribution?

Yes!!!

Recall that $p(x|g_\theta(z)) = \mathcal{N}(x|g_\theta(z), \sigma^2 * I)$. If $g_\theta(z)$ is a multi-layer neural network, then we can imagine the network using its first few layers to map the normally distributed z's to the latent values (like digit identity, stroke weight, angle, etc.) with exactly the right statitics. Then it can use later layers to map those latent values to a fully-rendered digit.

Generator가 여러 개의 레이어를 사용할 경우, 처음 몇 개의 레이어들를 통해 복잡할 수 있지만 딱 맞는 latent space로의 맵핑이 수행되고 나머지 레이어들을 통해 latent vector에 맞는 이미지를 생성할 수 있다.

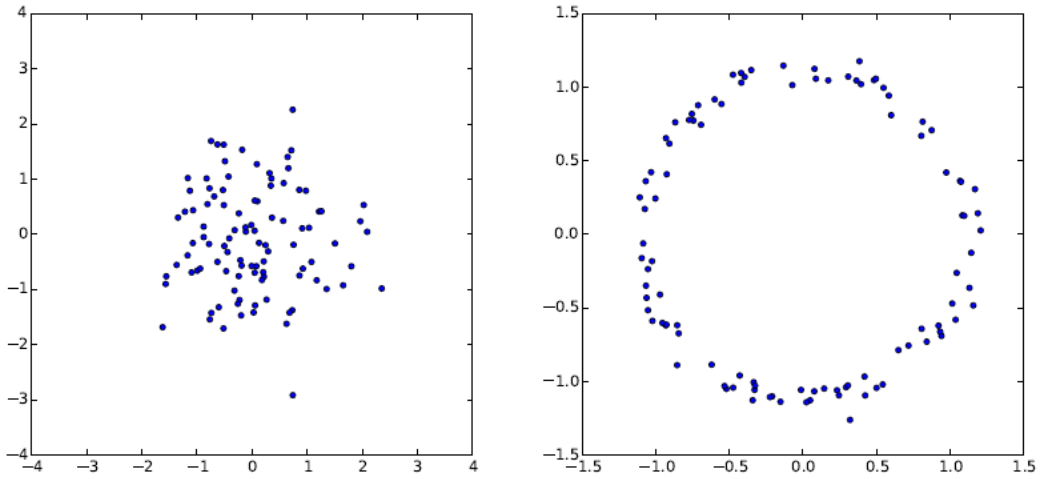


Figure 2: Given a random variable z with one distribution, we can create another random variable $X = g(z)$ with a completely different distribution. Left: samples from a gaussian distribution. Right: those same samples mapped through the function $g(z) = z/10 + z/||z||$ to form a ring. This is the strategy that VAEs use to create arbitrary distributions: the deterministic function g is learned from data.

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

Question: Why don't we use maximum likelihood estimation directly?

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

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.

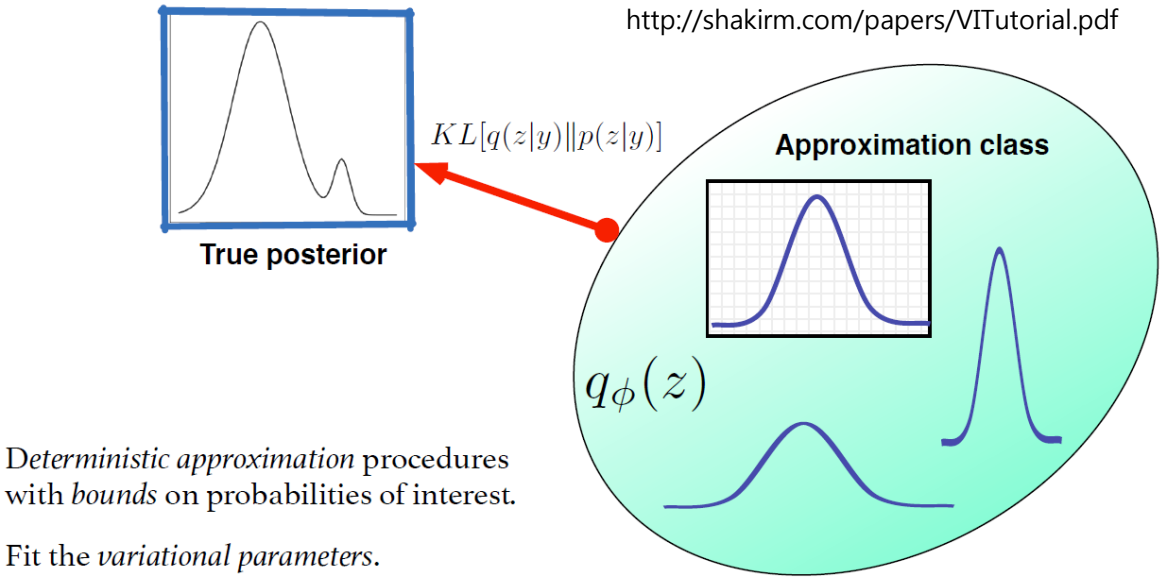
생성기에 대한 확률모델을 가우시안으로 할 경우, MSE관점에서 가까운 것이 더 p(x)에 기여하는 바가 크다.
MSE가 더 작은 이미지가 의미적으로도 더 가까운 경우가 아닌 이미지들이 많기 때문에 현실적으로 올바른 확률값을 구하기가 어렵다.

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

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

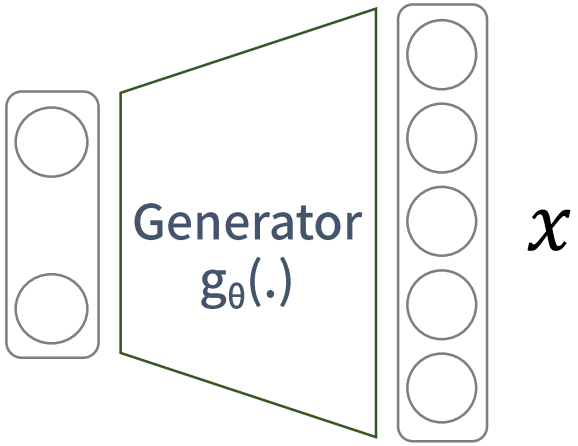
앞 슬라이드에서 Solution2가 가능하게 하는 방법 중 하나는 z를 정규분포에서 샘플링하는 것보다 x와 유의미하게 유사한 샘플이 나올 수 있는 확률분포 $p(z|x)$로 부터 샘플링하면 된다. 그러나 $p(z|x)$ 가 무엇인지 알지 못하므로, 우리가 알고 있는 확률분포 중 하나를 택해서 $(q_\phi(z|x))$ 그것의 파라미터값을 (λ) 조정하여 $p(z|x)$ 와 유사하게 만들어 본다. (Variational Inference)

http://shakirm.com/papers/VITutorial.pdf



[Variational Inference]

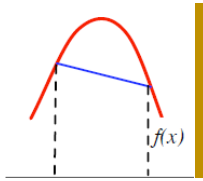
$$p(z|x) \approx q_\phi(z|x) \sim z$$



https://www.slideshare.net/haezoom/variational-autoencoder-understanding-variational-autoencoder-from-various-perspectives

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

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



[Jensen's Inequality]

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

f(.) = log(.) is concave

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

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

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

$ELBO(\phi)$ Variational lower bound
Evidence lower bound (ELBO)

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

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

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

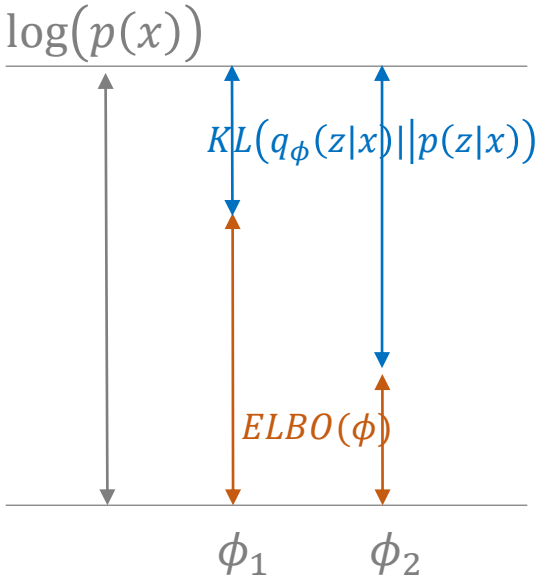
$$= \int \log\left(\frac{p(x,z)}{p(z|x)}\right) q_\phi(z|x)dz \quad \leftarrow p(x) = \frac{p(x,z)}{p(z|x)}$$

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

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

$$ELBO(\phi) \qquad KL\left(q_\phi(z|x) \,\|\, p(z|x)\right)$$

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.

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

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

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

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

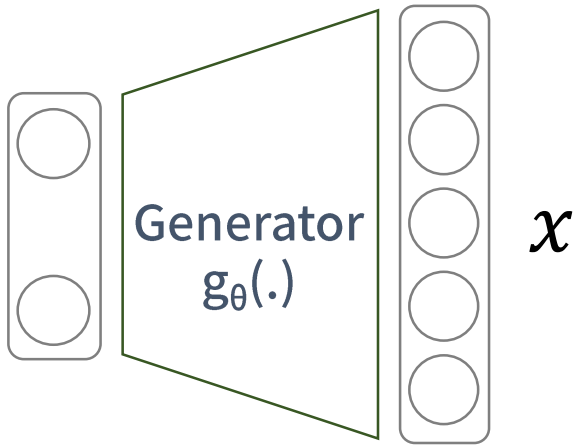
$$= \int \log\left(\frac{p(x|z)/p(z)}{q_\phi(z|x)}\right) q_\phi(z|x)dz$$

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

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

LOSS FUNCTION | Derivation

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



Optimization Problem 1 on ϕ: Variational Inference

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

Optimization Problem 2 on θ: Maximum likelihood

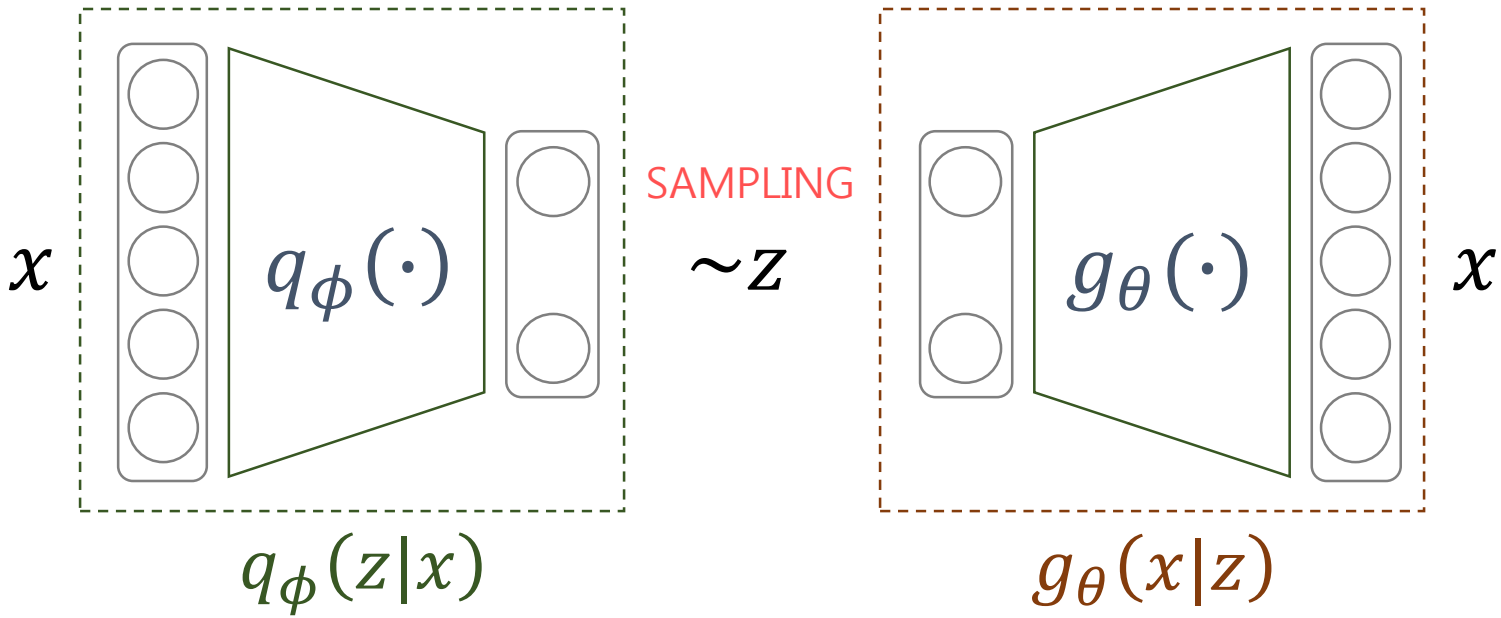
$$-\sum_i \log\big(p(x_i)\big) \leq -\sum_i \mathbb{E}_{q_\phi(z|x_i)}\big[\log\big(p(x_i|g_\theta(z))\big)\big] - KL\big(q_\phi(z|x_i)\big||p(z)\big)$$

Final Optimization Problem

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

LOSS FUNCTION | NeuralNet Perspective

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



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

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

LOSS FUNCTION | Explanation

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

원 데이터에 대한 likelihood

Variational inference를 위한 approximation class 중 선택

다루기 쉬운 확률 분포 중 선택

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

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

LOSS FUNCTION | Explanation

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

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

$$= \underbrace{-\mathbb{E}_{q_\phi(z|x_i)}\left[\log\left(p_\theta(x_i|z)\right)\right]}_{\text{Reconstruction Error}} \underbrace{- H\left(q_\phi(z|x_i)\right)}_{\text{Posterior Entropy}} + \underbrace{H\left(q_\phi(z|x_i), p(z)\right)}_{\text{Cross Entropy}}$$

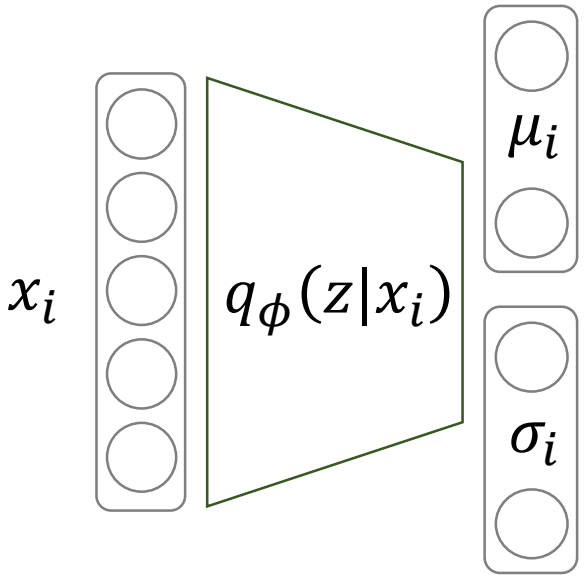
Posterior에서 샘플링 된 z는 최대한 다양해야 한다
(mode collapse 방지 효과)

Posterior와 Prior의 정보량은
유사해야 한다

LOSS FUNCTION | Regularization

Assumptions

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



Assumption 1

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

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

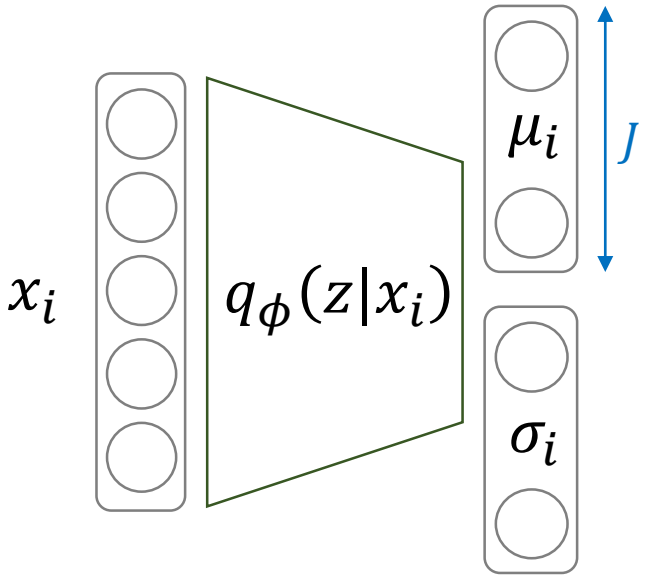
Assumption 2

[prior] multivariate normal distribution

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

LOSS FUNCTION | Regularization

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



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

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

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

posterior prior

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

where k is the dimension of the vector space.

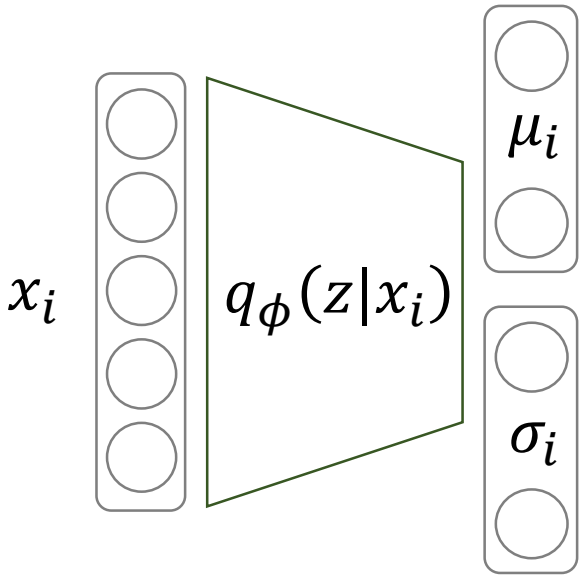
LOSS FUNCTION | Reconstruction error

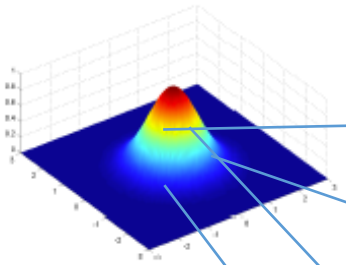
Sampling

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

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

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





SAMPLING

$z^{i,1}$ $\log p_\theta\left(x_i|z^{i,1}\right)$

$z^{i,2}$ $\log p_\theta\left(x_i|z^{i,2}\right)$

$z^{i,l}$ $\log p_\theta\left(x_i|z^{i,l}\right)$

$z^{i,L}$ $\log p_\theta\left(x_i|z^{i,L}\right)$

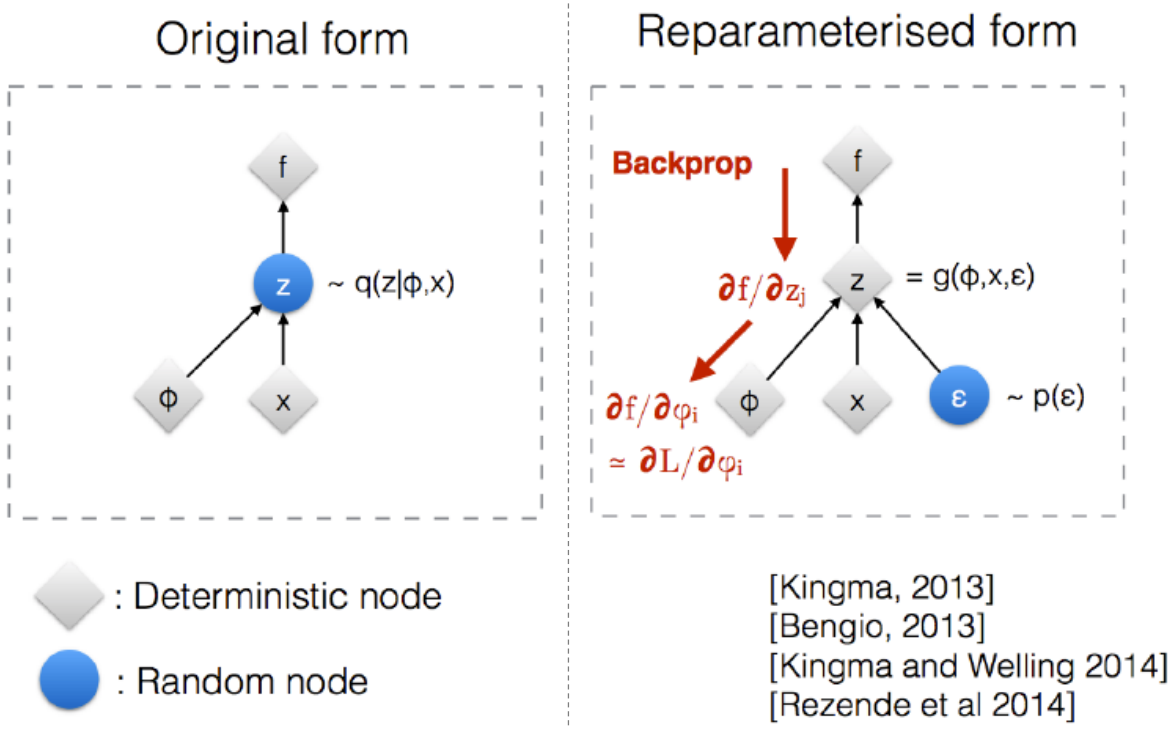
mean → Reconstruction Error

- 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

LOSS FUNCTION | Reconstruction error

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!!

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

LOSS FUNCTION | Reconstruction error

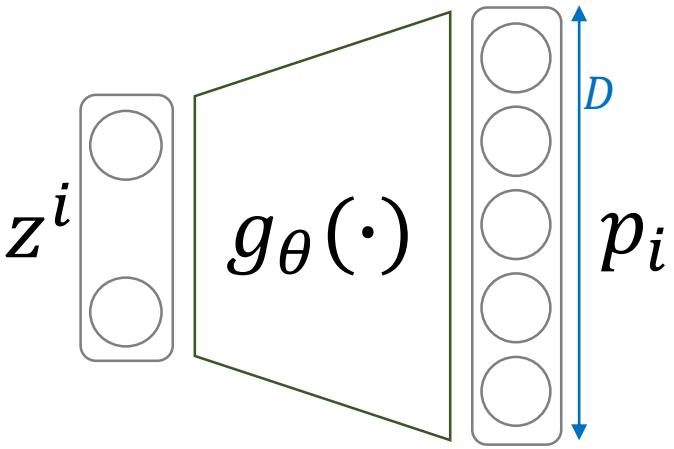
Assumption

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

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

Monte-carlo technique

L=1



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

Assumption 3-1

[Decoder, likelihood]

multivariate bernoulli or gaussain distribution

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

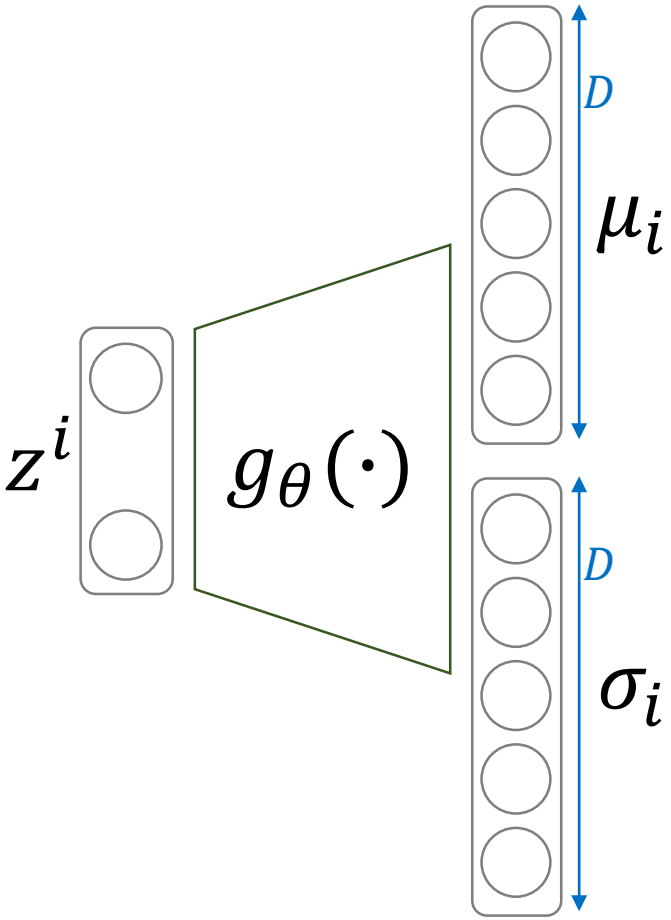
$$= \sum_{j=1}^{D} \log p_{i,j}^{x_{i,j}}\left(1-p_{i,j}\right)^{1-x_{i,j}} \quad \leftarrow p_{i,j} \triangleq network\ output$$

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

LOSS FUNCTION | Reconstruction error

Assumption

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



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

Assumption 3-2

[Decoder, likelihood]
multivariate bernoulli or gaussain distribution

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

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

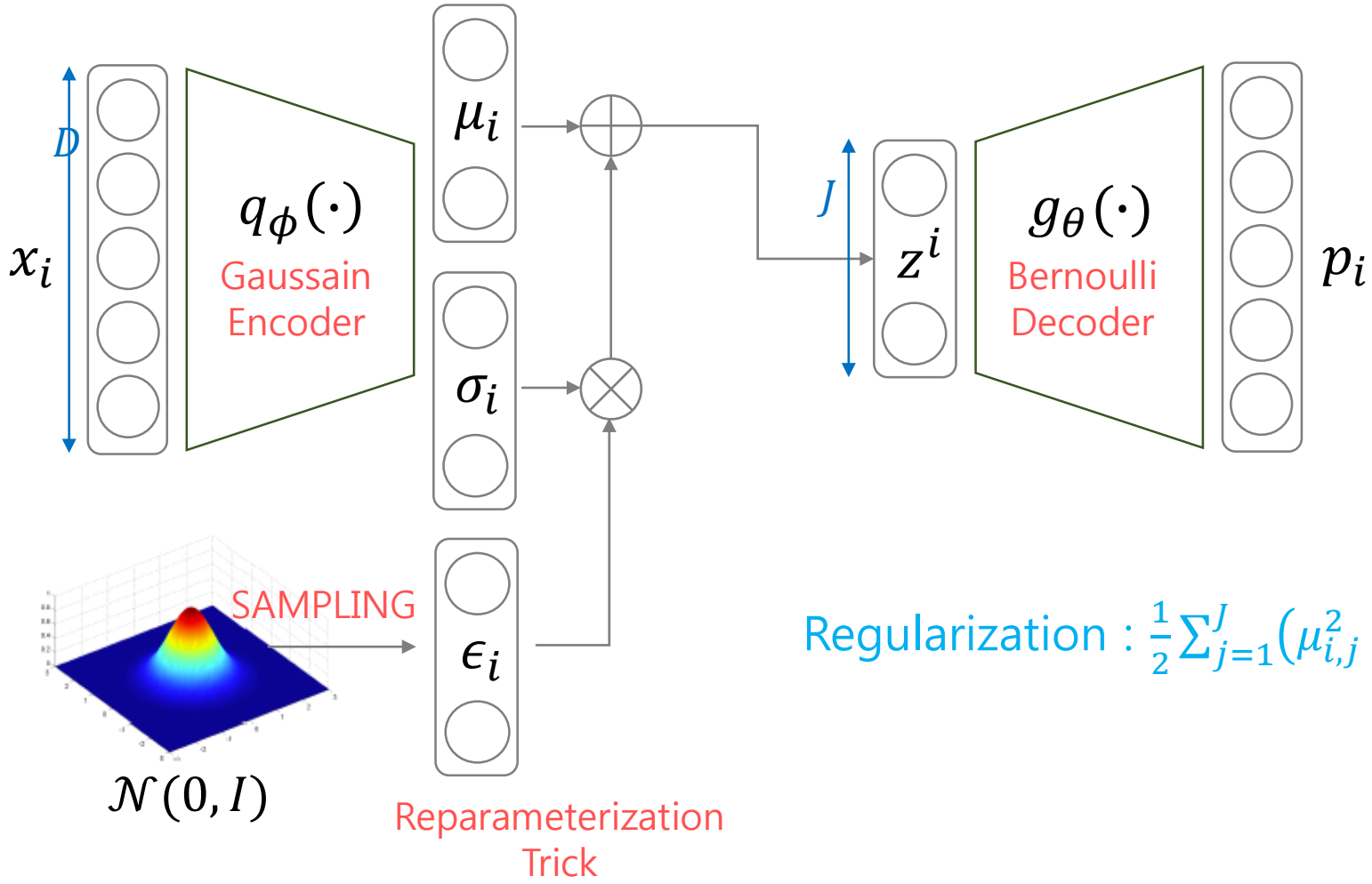
For gaussain distribution with identity covariance

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

STRUCTURE

Default : Gaussian Encoder + Bernoulli Decoder

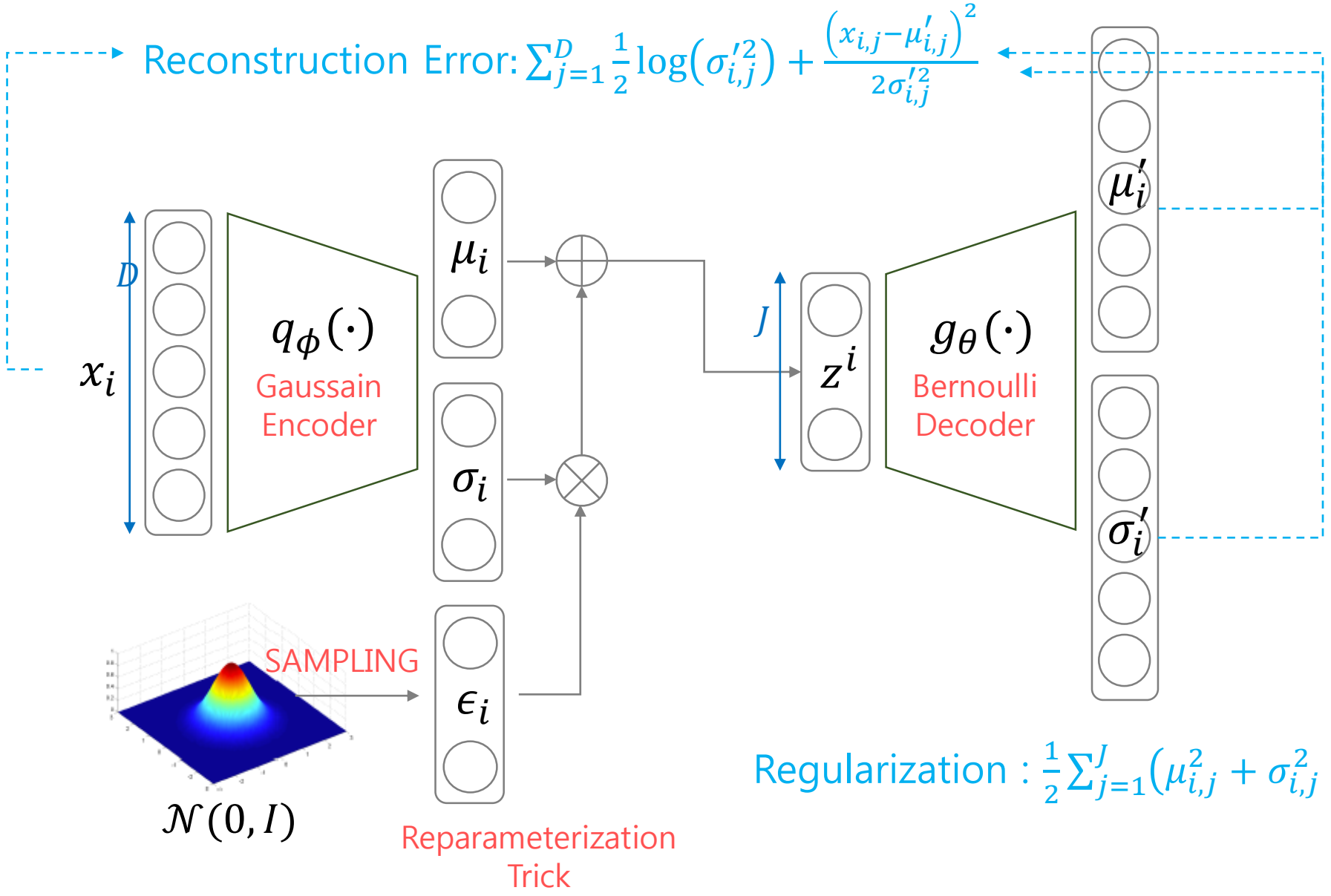
Reconstruction Error: $-\sum_{j=1}^{D} x_{i,j} \log p_{i,j} + (1 - x_{i,j}) \log(1 - p_{i,j})$



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

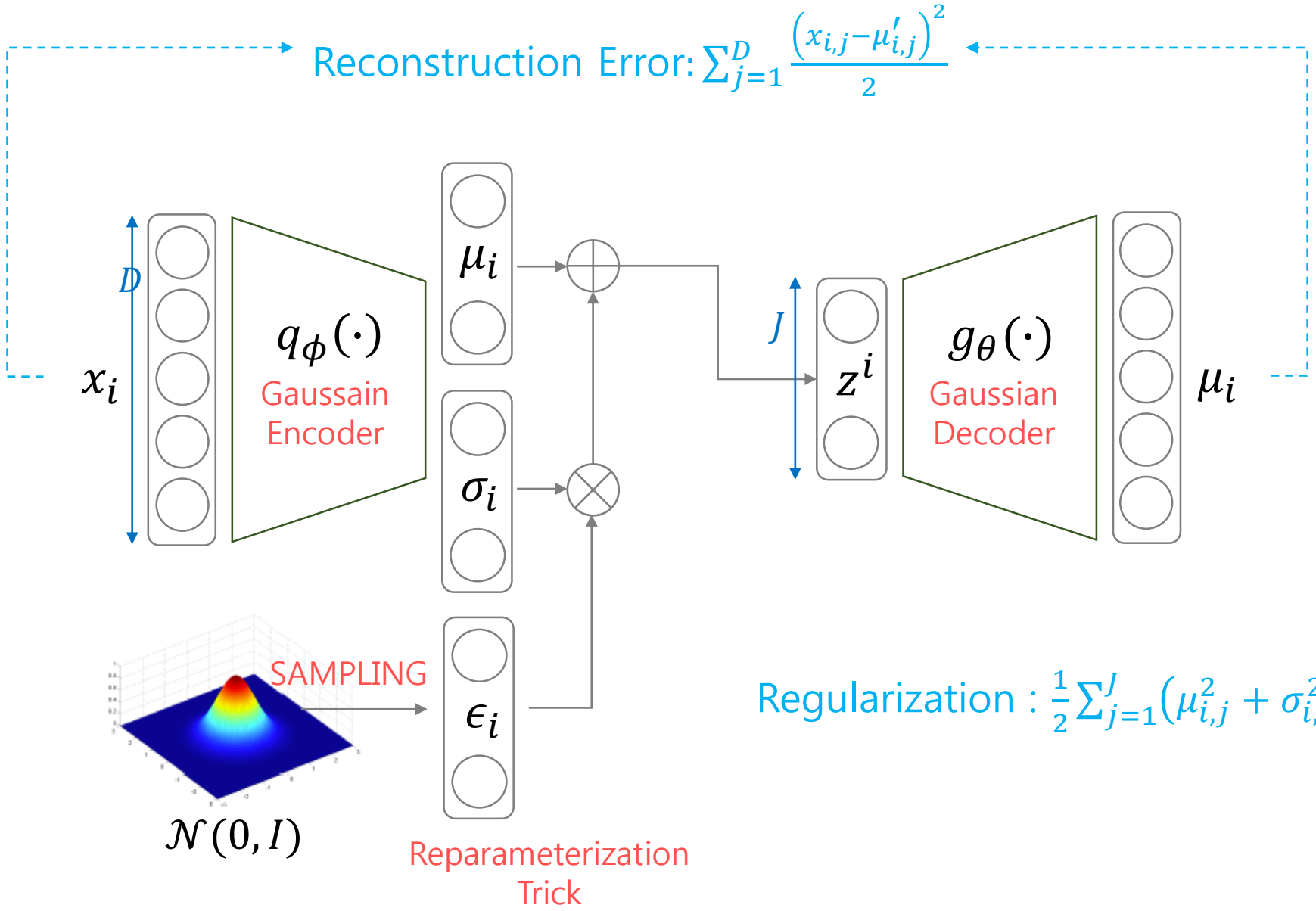
STRUCTURE

Gaussian Encoder + Gaussian Decoder

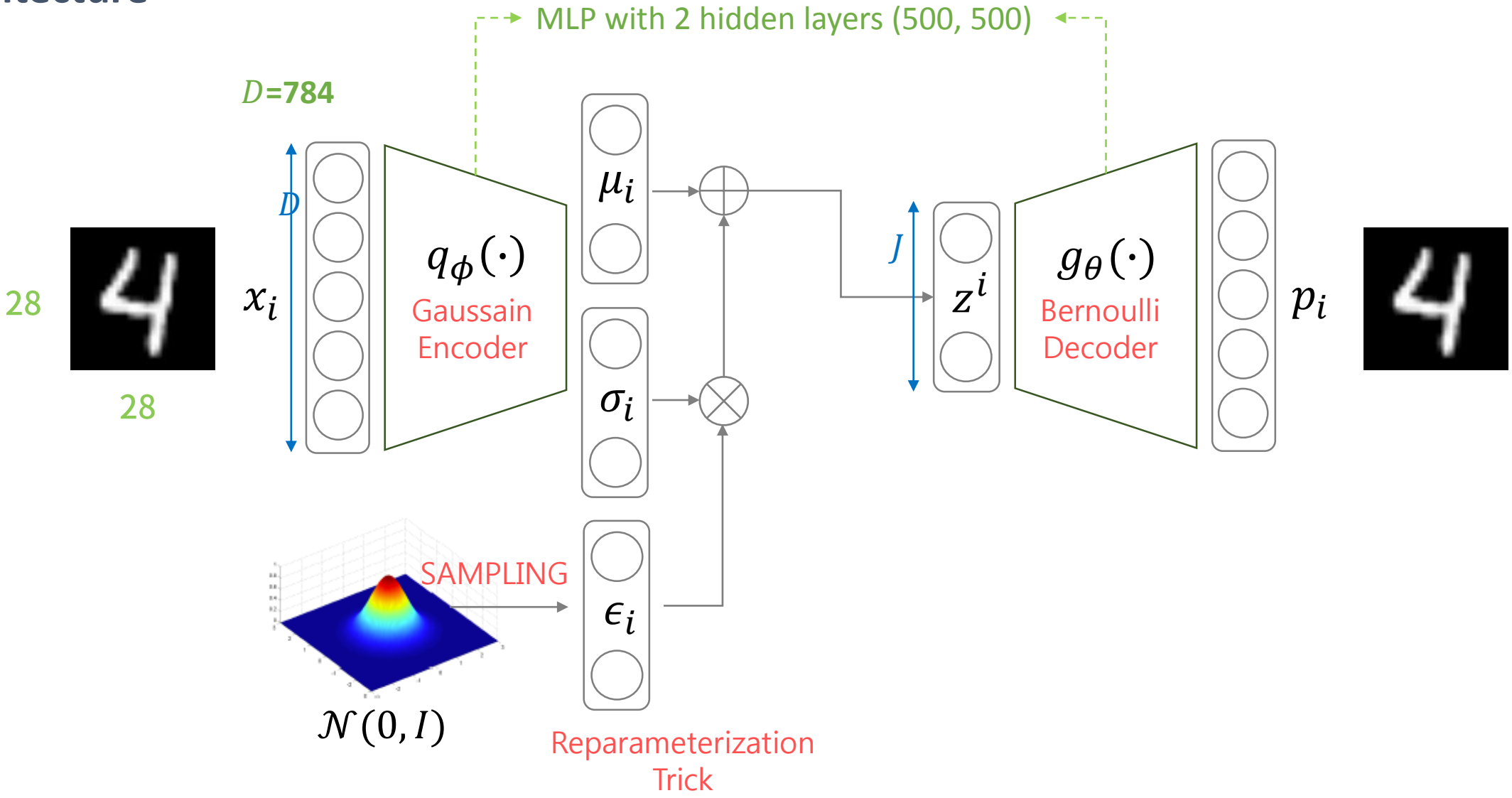


STRUCTURE

Gaussian Encoder + Gaussian Decoder with Identity Covariance



Architecture



Reproduce

Input image

J = |z| =2

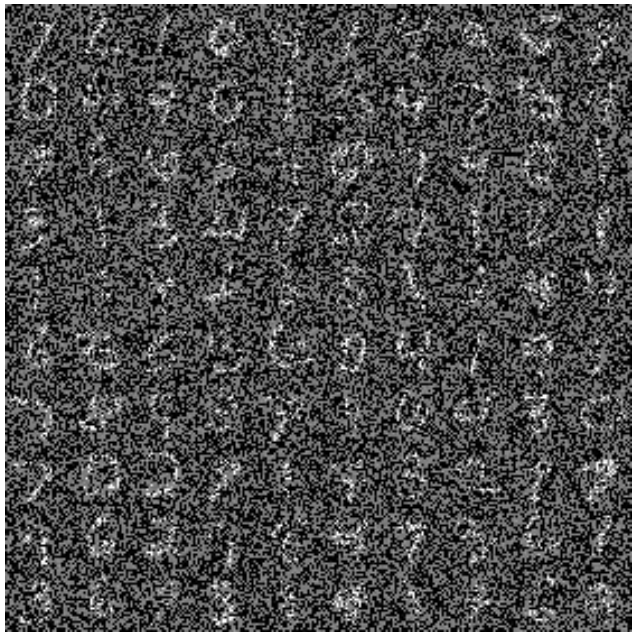
J = |z| =5

J = |z| =20

https://github.com/hwalsuklee/tensorflow-mnist-VAE

Denoising

Input image



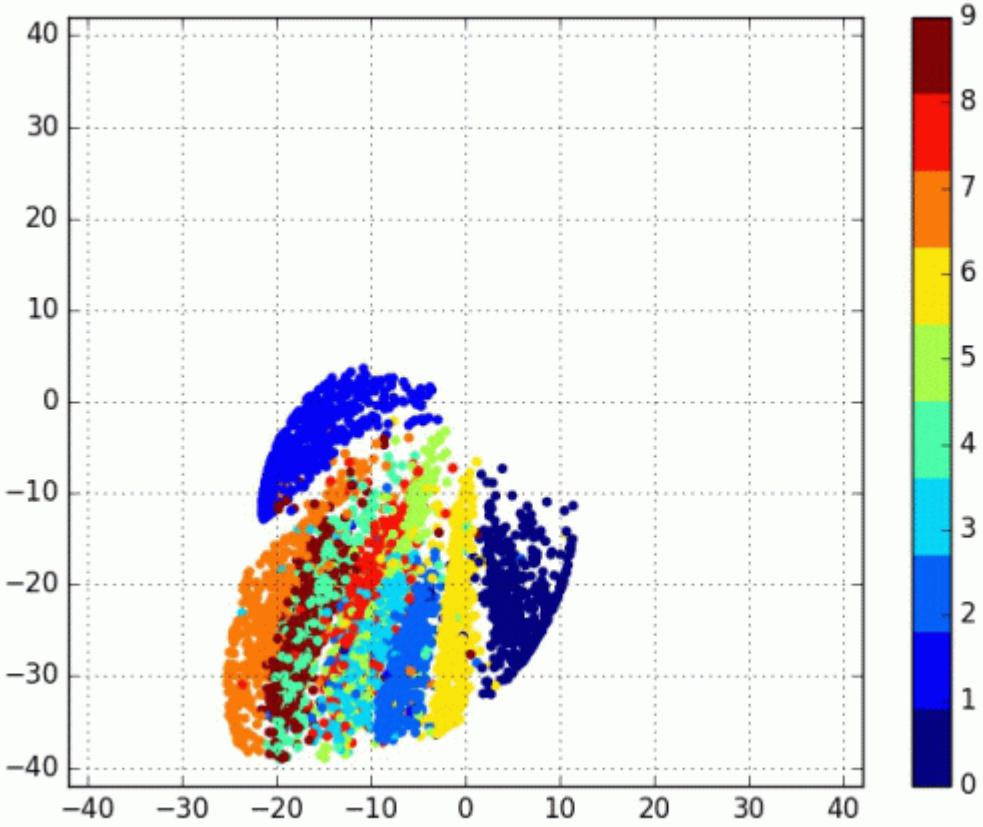
+ zero-masking noise with 50% prob.
+ salt&peppr noise with 50% prob.

Restored image

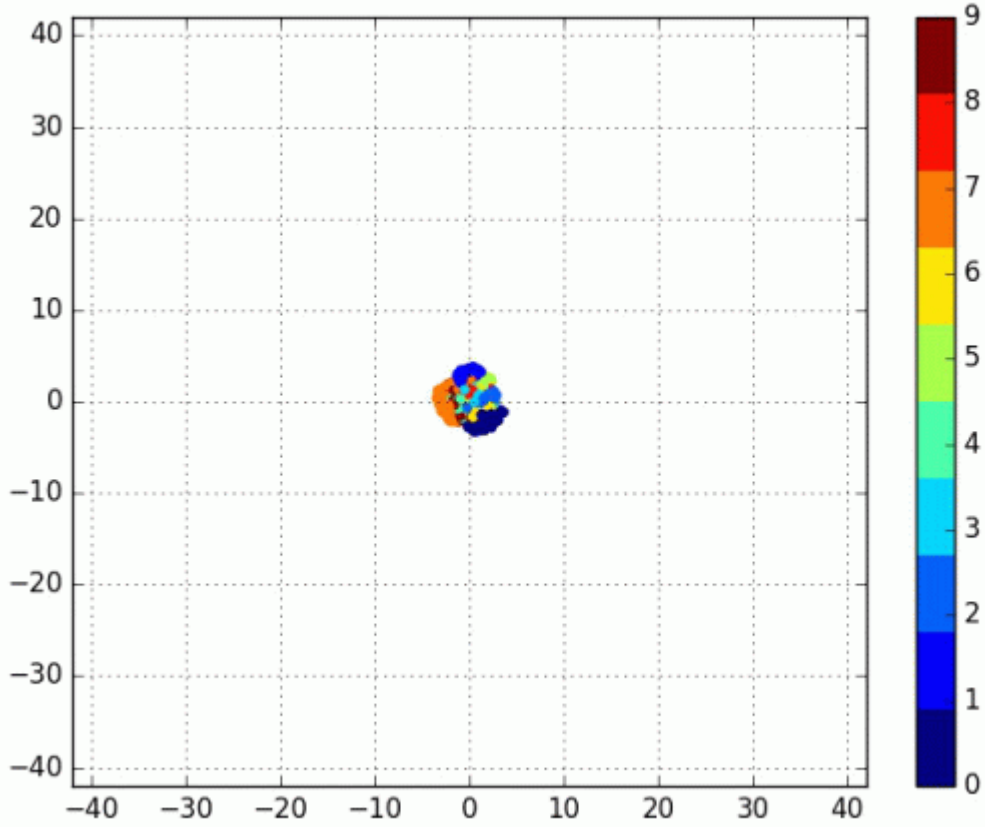
https://github.com/hwalsuklee/tensorflow-mnist-VAE

Learned Manifold

AE



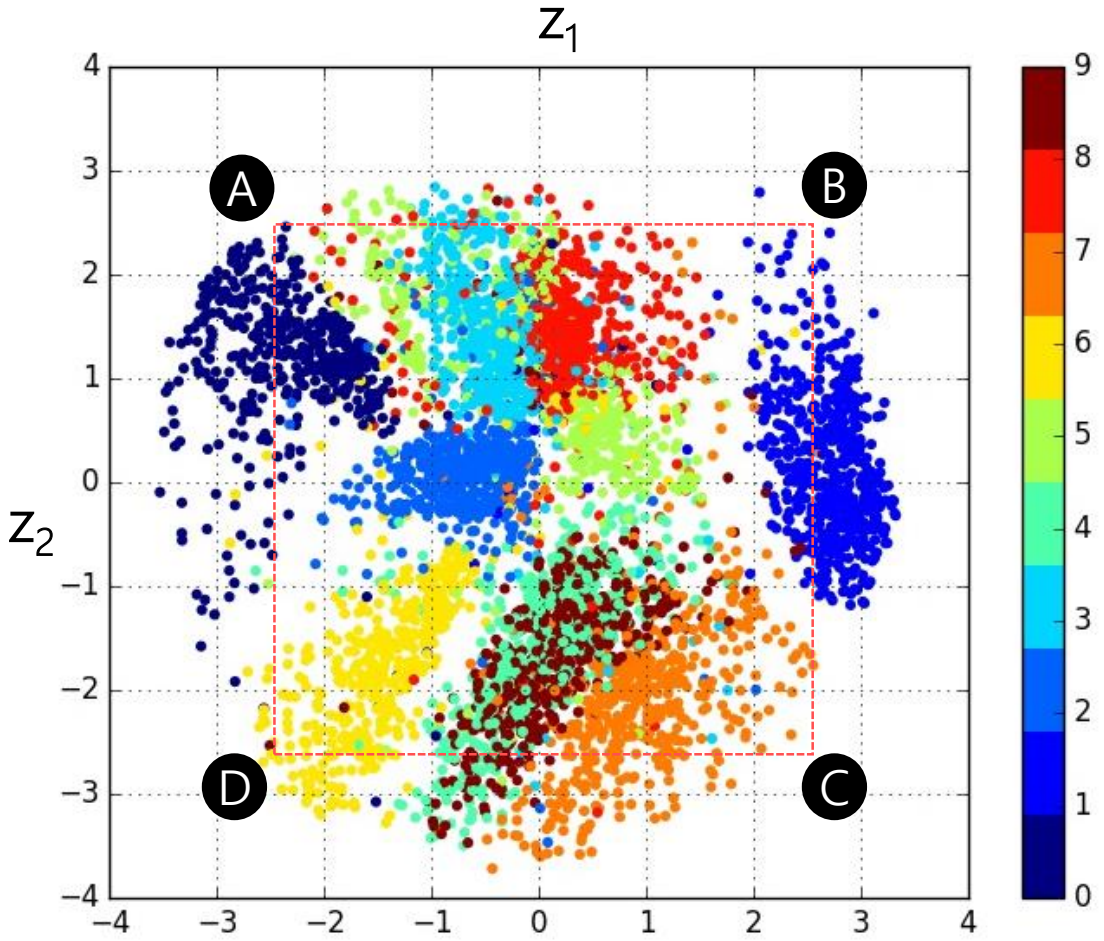
VAE

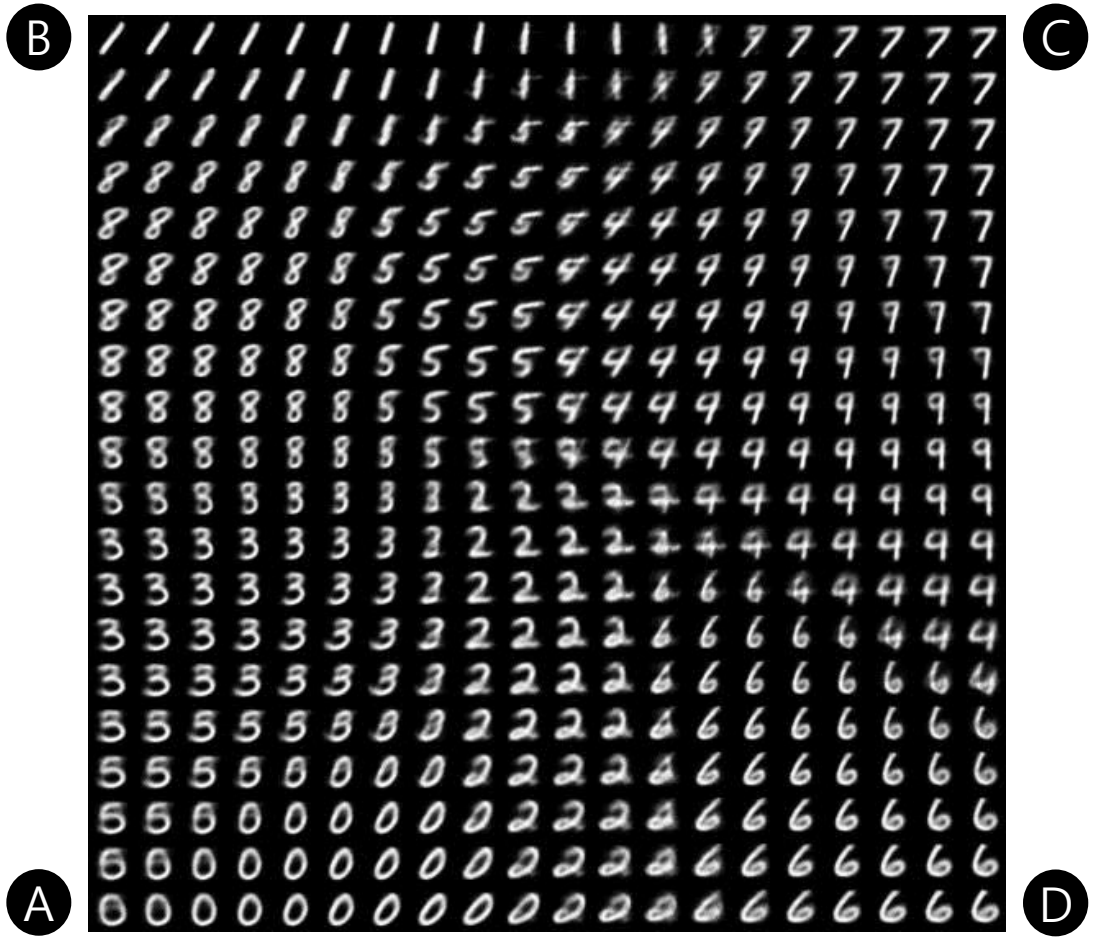


- 테스트 샘플 중 5000개가 매니폴드 어느 위치에 맵핑이 되는 지 보여줌.
- 학습 6번의 결과를 애니메이션으로 표현.
- 생성 관점에서는 다루고자 하는 매니폴드의 위치가 안정적이야 좋음.

https://github.com/hwalsuklee/tensorflow-mnist-VAE

Learned Manifold



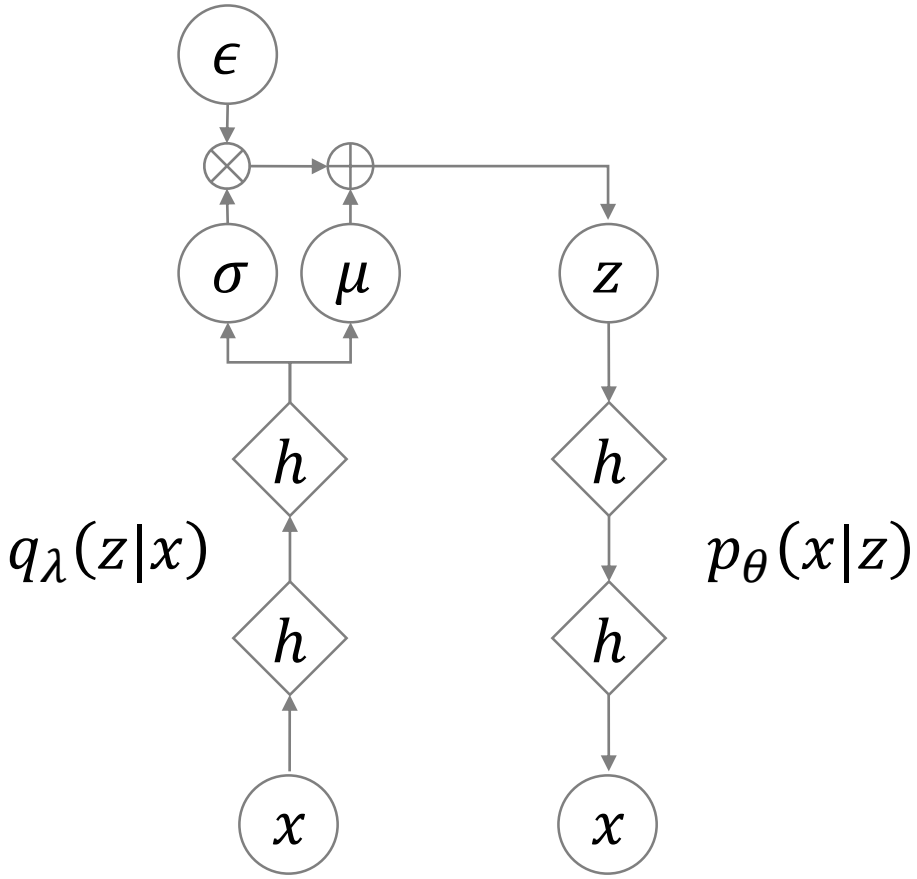


학습이 잘 되었을 수록 2D공간에서 같은 숫자들을 생성하는 z들은 뭉쳐있고,
다른 숫자들은 생성하는 z들은 떨어져 있어야 한다.

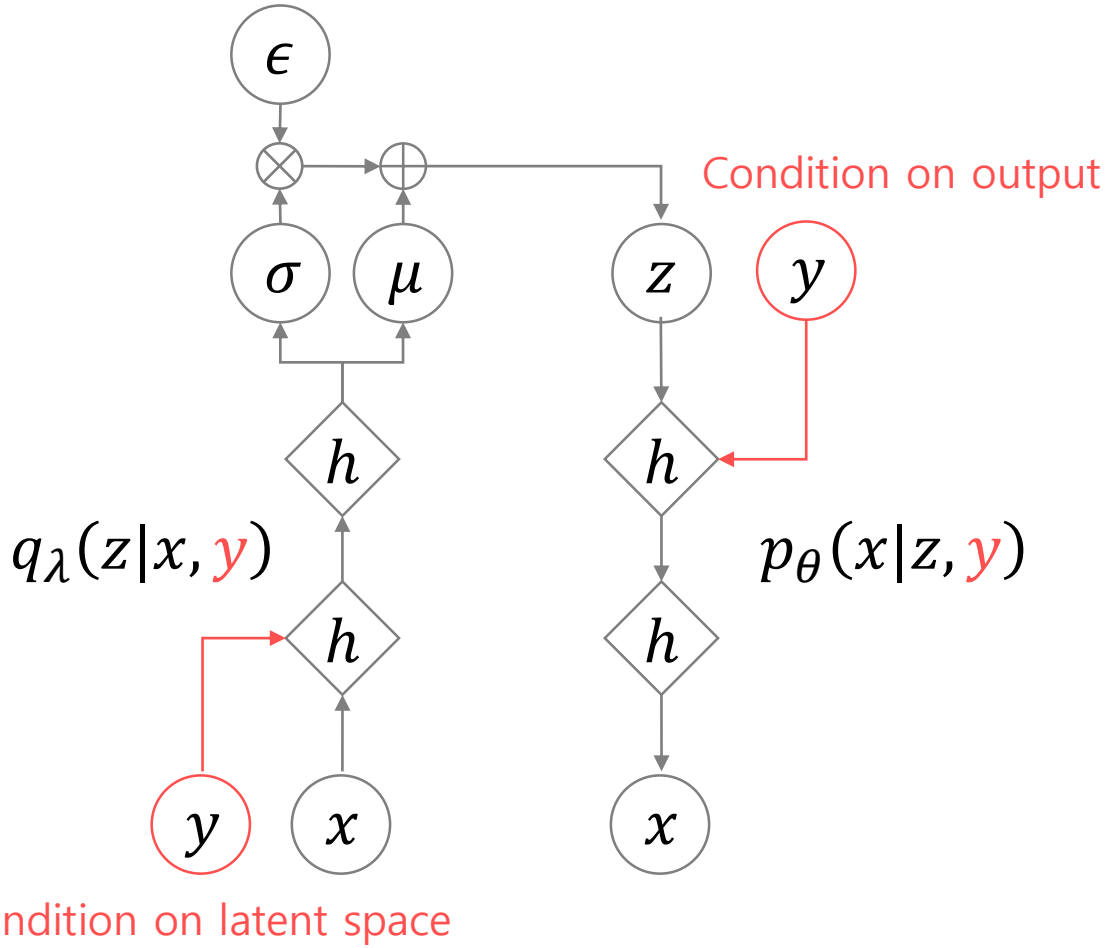
https://github.com/hwalsuklee/tensorflow-mnist-VAE

Conditional VAE

Vanilla VAE (M1)

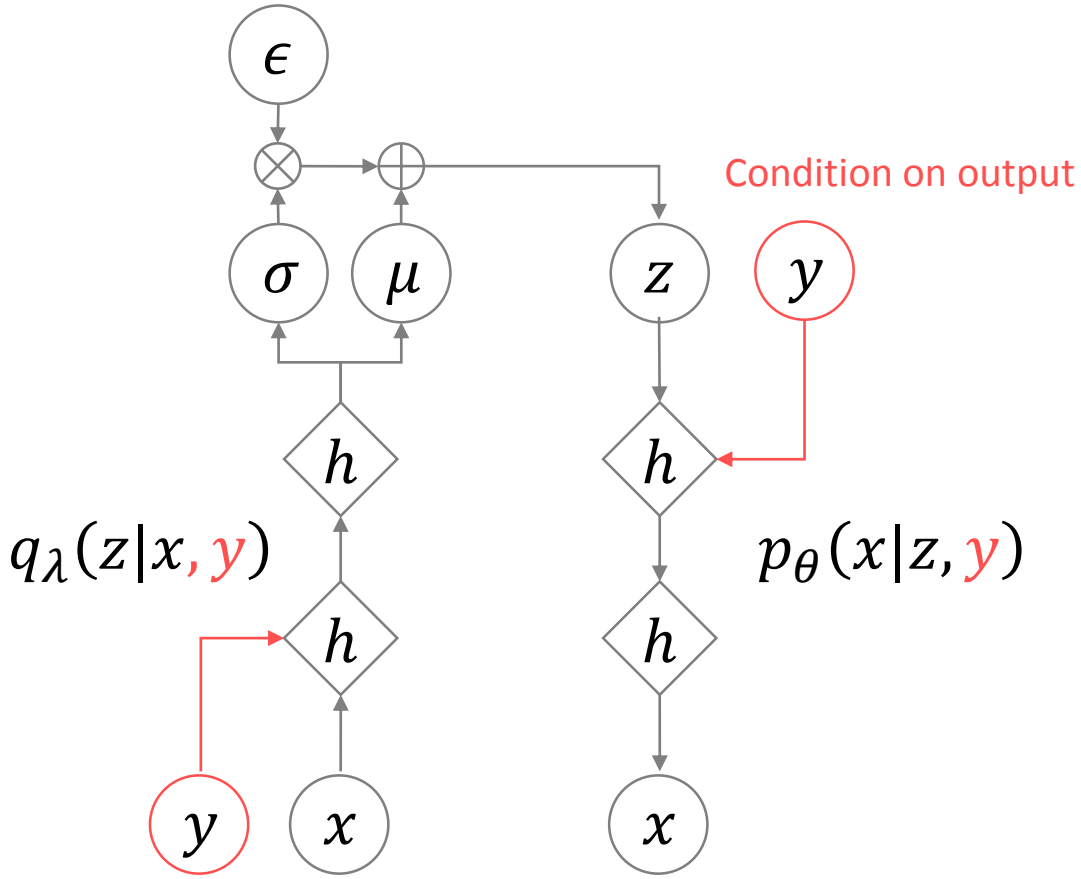


CVAE (M2) : supervised version



Summary

CVAE (M2) : supervised version



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

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

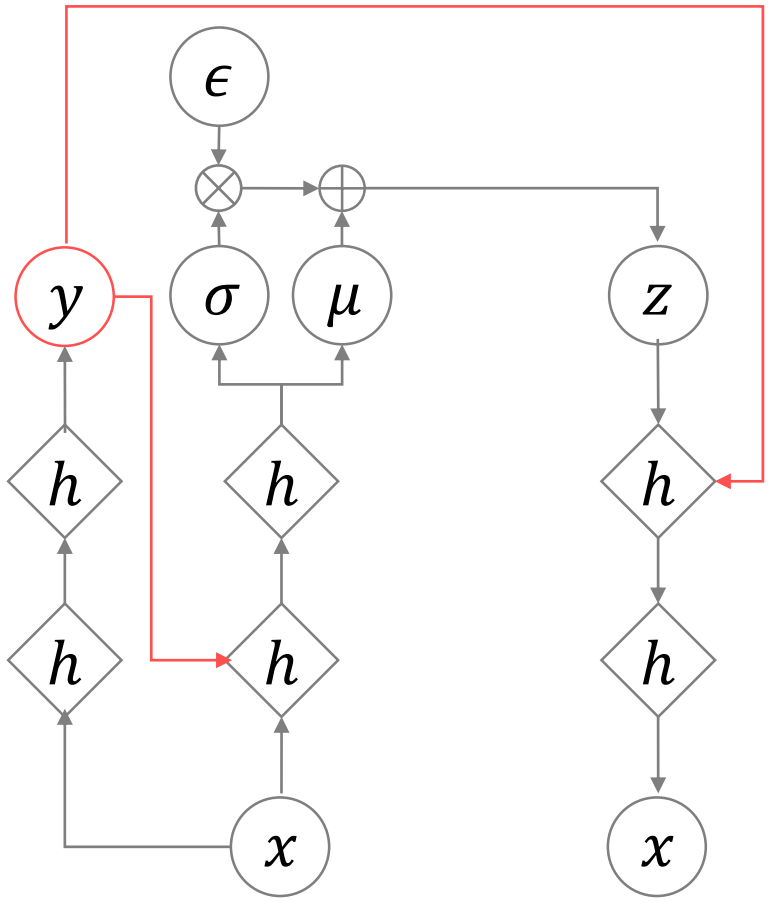
$$= \int \log\left(p_\theta(x|y,z) \frac{p(y)p(z)}{q_\phi(z|x,y)} \right) q_\phi(z|x,y) dz$$

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

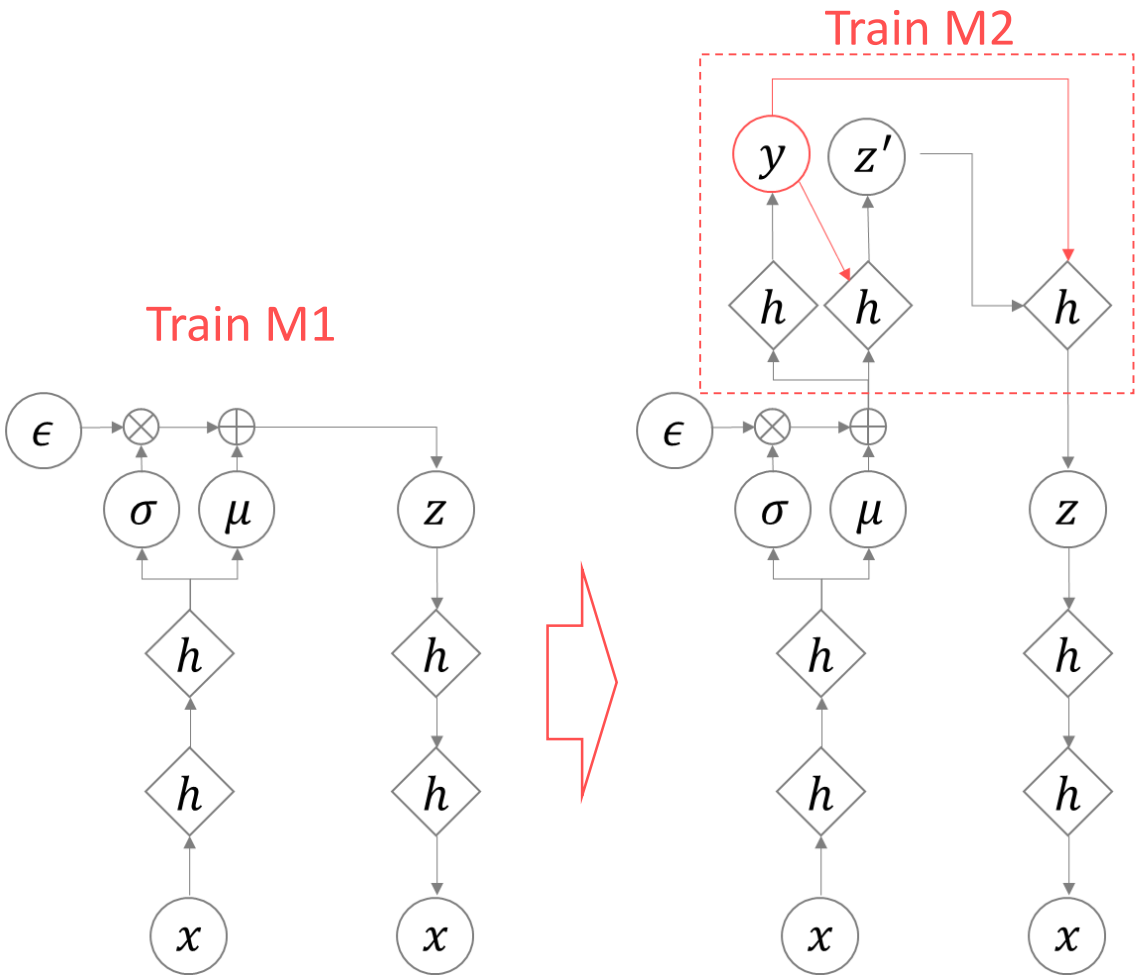
$$= -\mathcal{L}(x,y)$$ ELBO!!

Summary

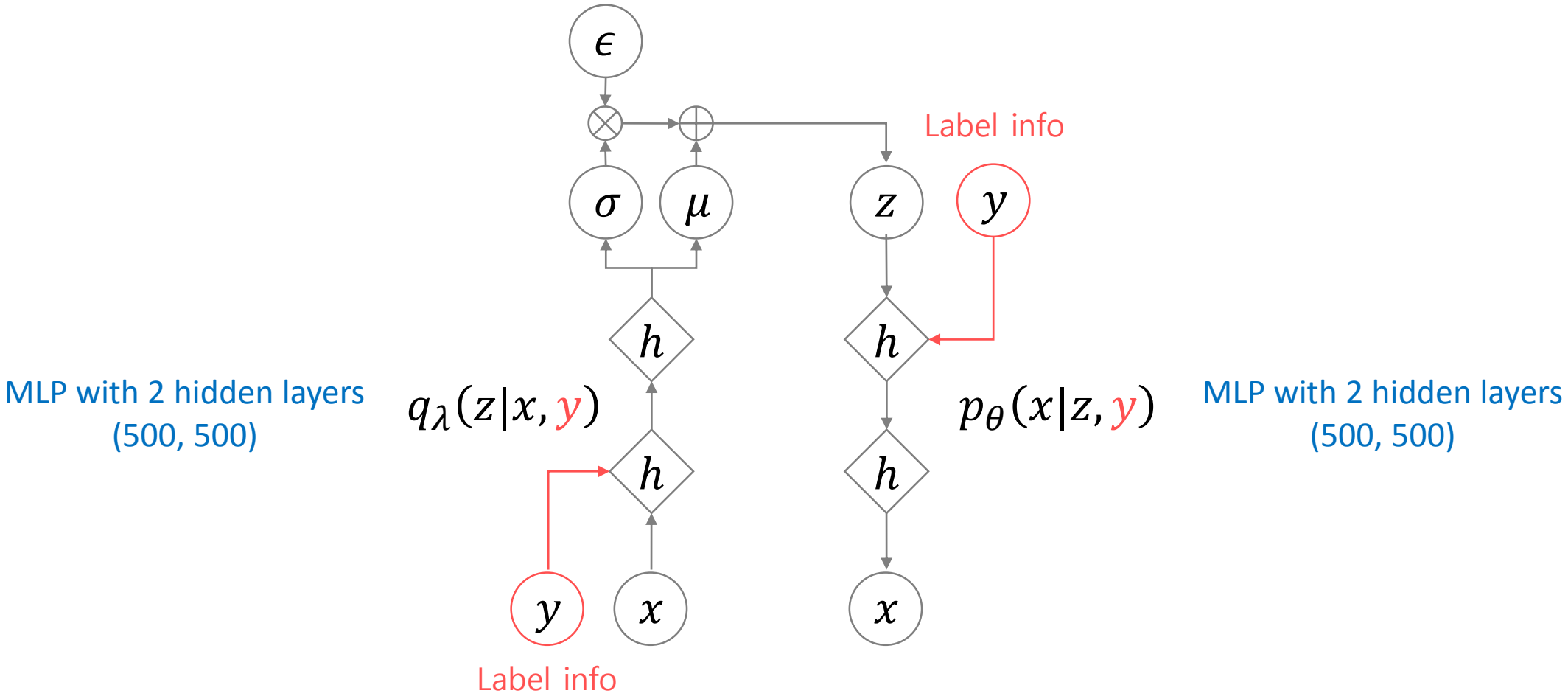
CVAE (M2) : unsupervised version



CVAE (M3)



Architecture : M2 supervised version



Reproduce |z| = 2

input

CVAE, epoch 1

CVAE, epoch 20

VAE, epoch 1

VAE, epoch 20

https://github.com/hwalsuklee/tensorflow-mnist-CVAE

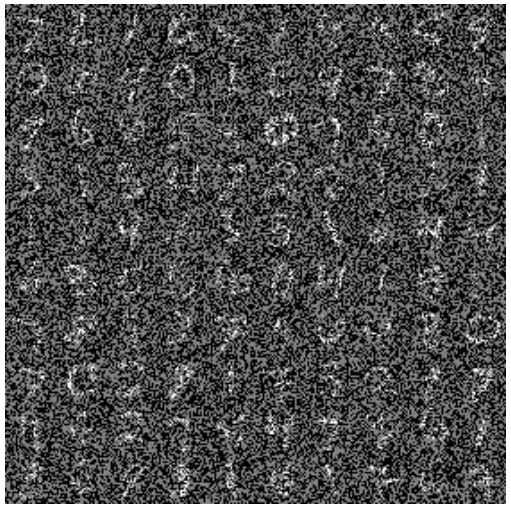
Denoising

$|z| = 2$

input

CVAE, epoch 1

CVAE, epoch 20

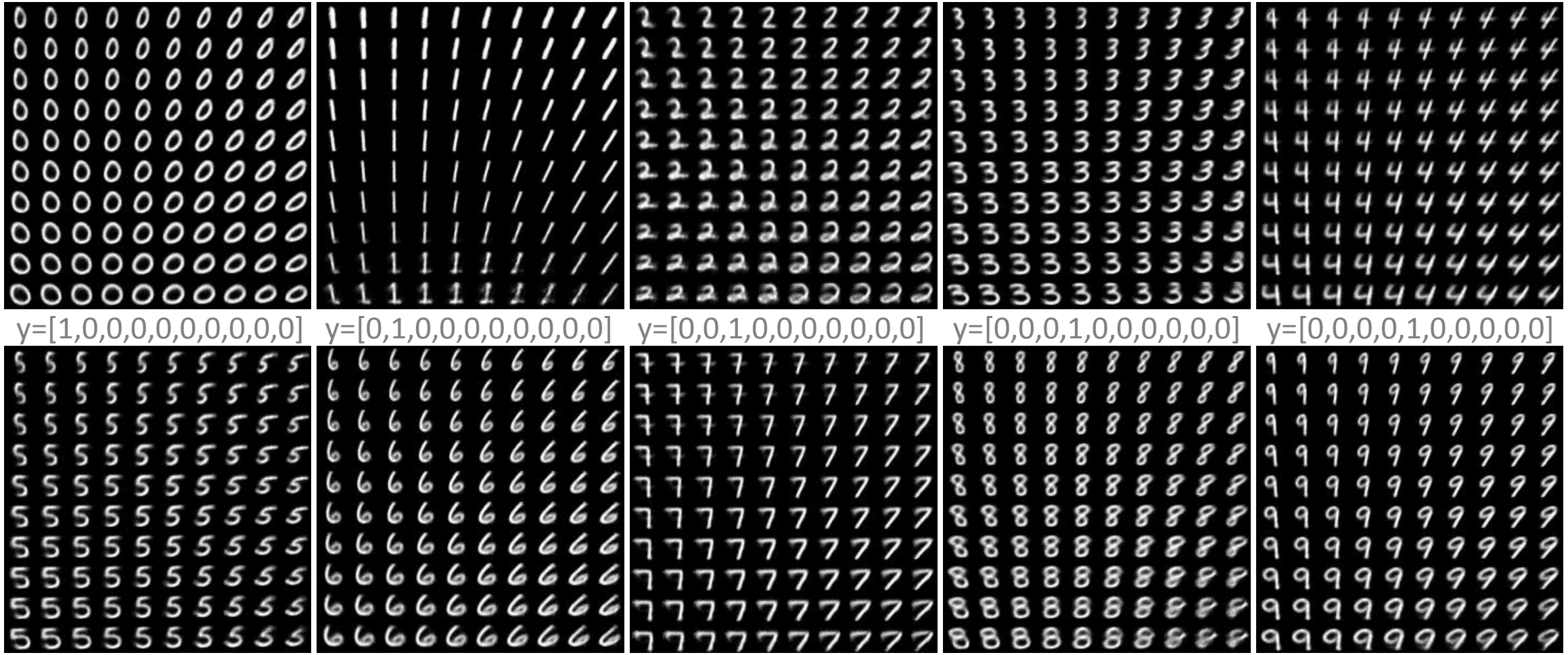


VAE, epoch 1

VAE, epoch 20

https://github.com/hwalsuklee/tensorflow-mnist-CVAE

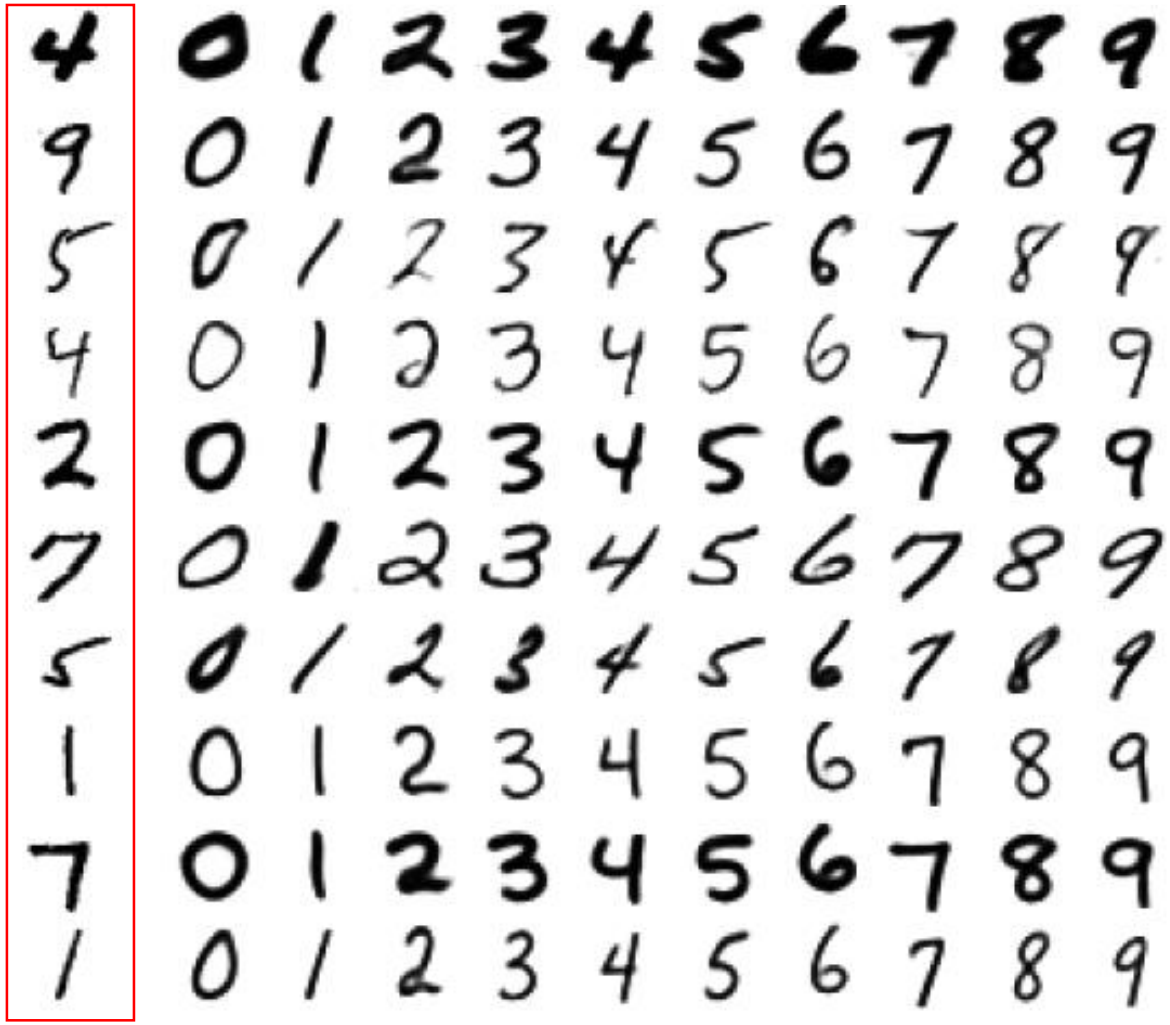
Handwriting styles obtained by fixing the class label and varying z |z| = 2



https://github.com/hwalsuklee/tensorflow-mnist-CVAE

Analogies : Result in paper

각 행 별로, 고정된 z값에 대해서 label정보만 바꿔서 이미지 생성 (스타일 유지하면 숫자만 바뀜)

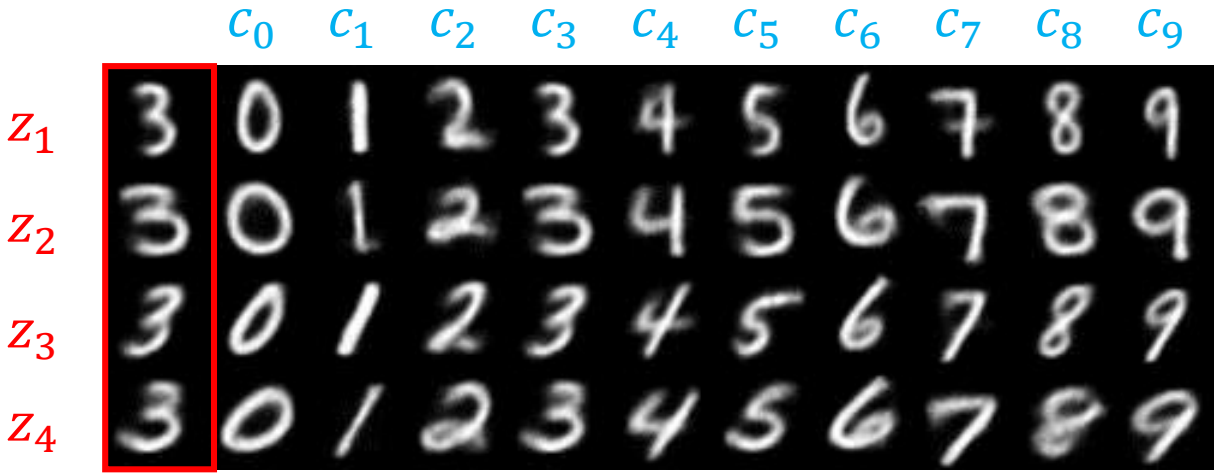


Z-sampling

Semi-Supervised Learning with Deep Generative Models : https://arxiv.org/abs/1406.5298

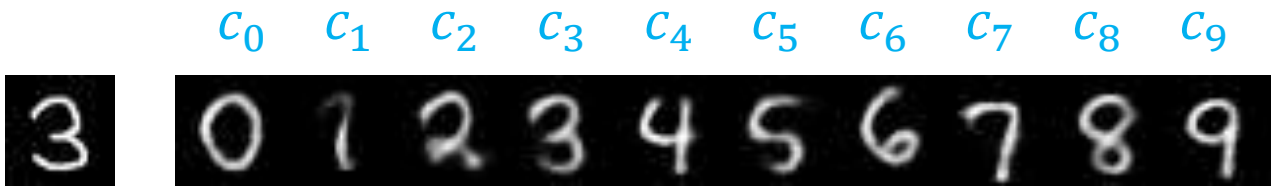
Analogies

|z| = 2



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

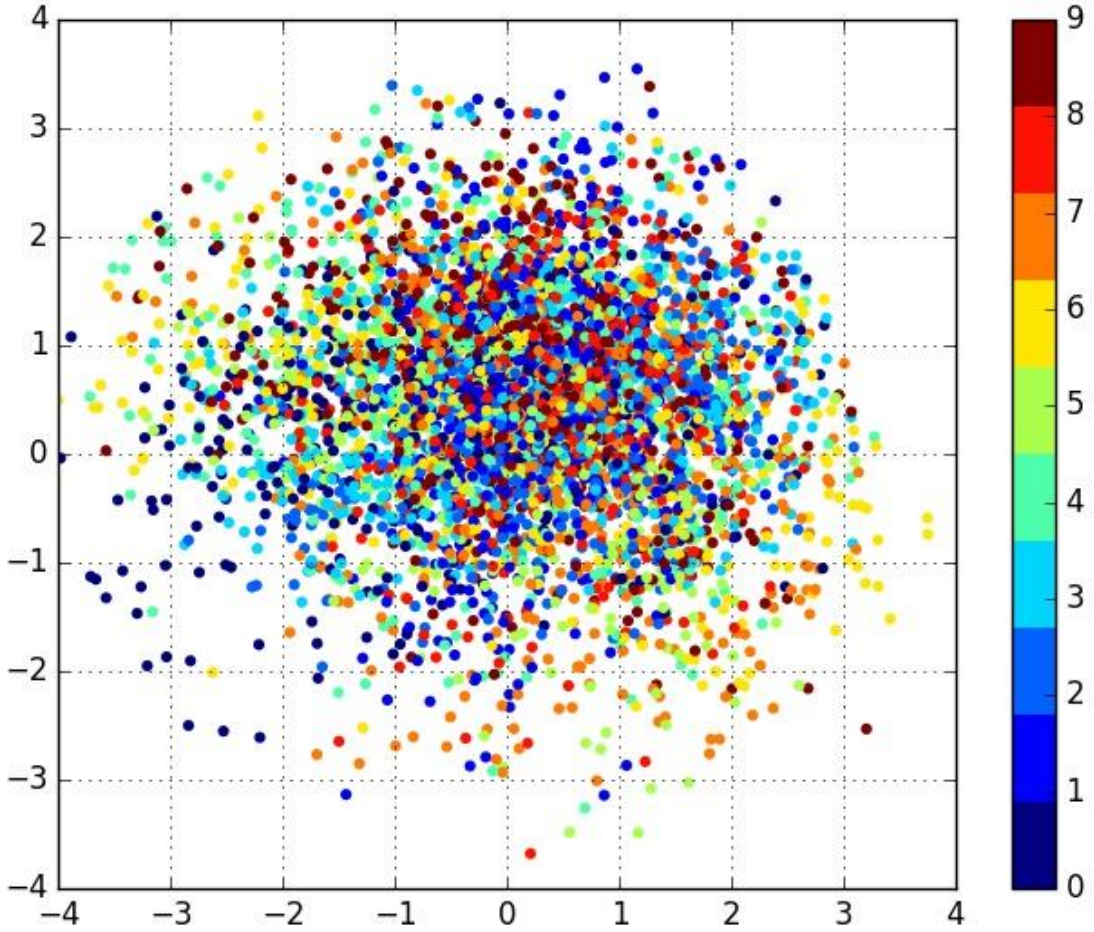
Real handwritten image



실제로 손으로 쓴 글씨 '3'을 CVAE의 label정보와 같이 넣었을 때 얻는
latent vector는 decoder의 고정 입력으로 하고, label정보만 바꿨을 경우

https://github.com/hwalsuklee/tensorflow-mnist-CVAE

Learned Manifold |z| = 2



Things are messy here, in contrast to VAE's Q(z|X), which nicely clusters z.
But if we look at it closely, we could see that given a specific value of c=y, Q(z|X,c=y) is roughly N(0,1)!
It's because, if we look at our objective above, we are now modeling P(z|c), which we infer variationally with a N(0,1).

Q(z|X,c=y)가 N(0,1)에 가까운 모습인데,
P(z|c)가 N(0,1)이고 Q(z|X,c=y)는 P(z|c)와의 KL-Divergence를 최소화하도록 학습이 되기 때문에 바람직한 현상이다.
(P(z|X,c=y) = Q(z|X,c=y)임은 이미지 결과로 확인했음.)

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)

N = 100일 때 직접 돌려보니, 0.9514 → 4.86
(총 50000개 중, 100개만 레이블 사용, 49900개는 미사용)

https://github.com/saemundsson/semisupervised_vae

Semi-Supervised Learning with Deep Generative Models : https://arxiv.org/abs/1406.5298

Adversarial Autoencoder

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

Regularization

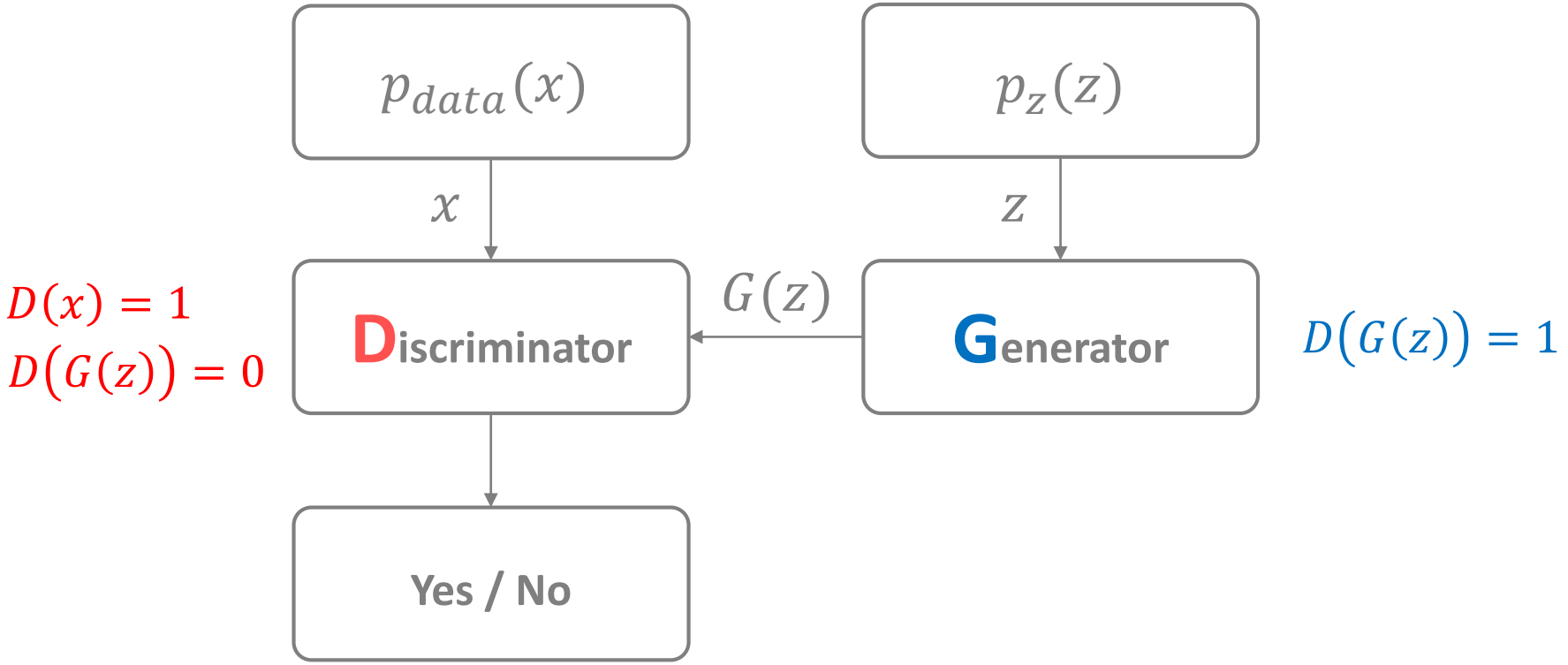
Conditions for $q_\phi(z|x_i)$, $p(z)$
1. Easily draw samples from distribution
2. KL divergence can be calculated

Adversarial Autoencoder (AAE)

Conditions	$q_\phi(z\|x_i)$	$p(z)$
Easily draw samples from distribution	O	O
KL divergence can be calculated	X	X

KL divergence is replaced by discriminator in GAN

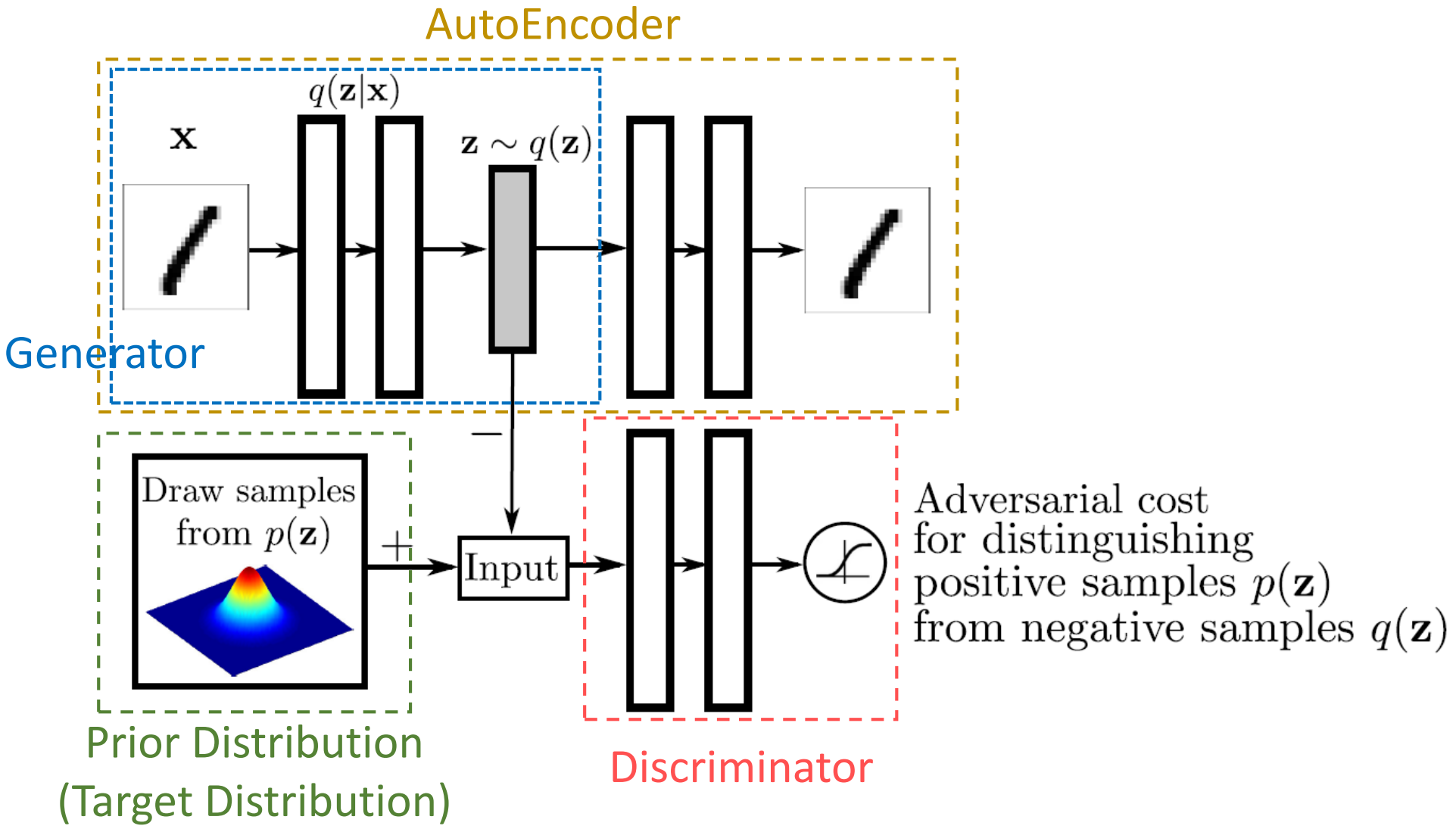
Generative Adversarial Network



Value function of GAN : $V(D,G) = \mathbb{E}_{x\sim p_{data}(x)}[\log D(x)] + \mathbb{E}_{z\sim p_z(z)}\left[\log\left(1 - D\left(G(z)\right)\right)\right]$

Goal : $D^*, G^* = \min_G \max_D V(D,G)$ GAN은 $G(z)\sim p_{data}(x)$로 만드는 것이 목적이다

Overall



Adversarial Autoencoders : https://arxiv.org/abs/1511.05644

Loss Function

GAN loss

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

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

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

*논문에는 로스 정의가 제시되어 있지 않아 새로 정리한 내용

VAE loss

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

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

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

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

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

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

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

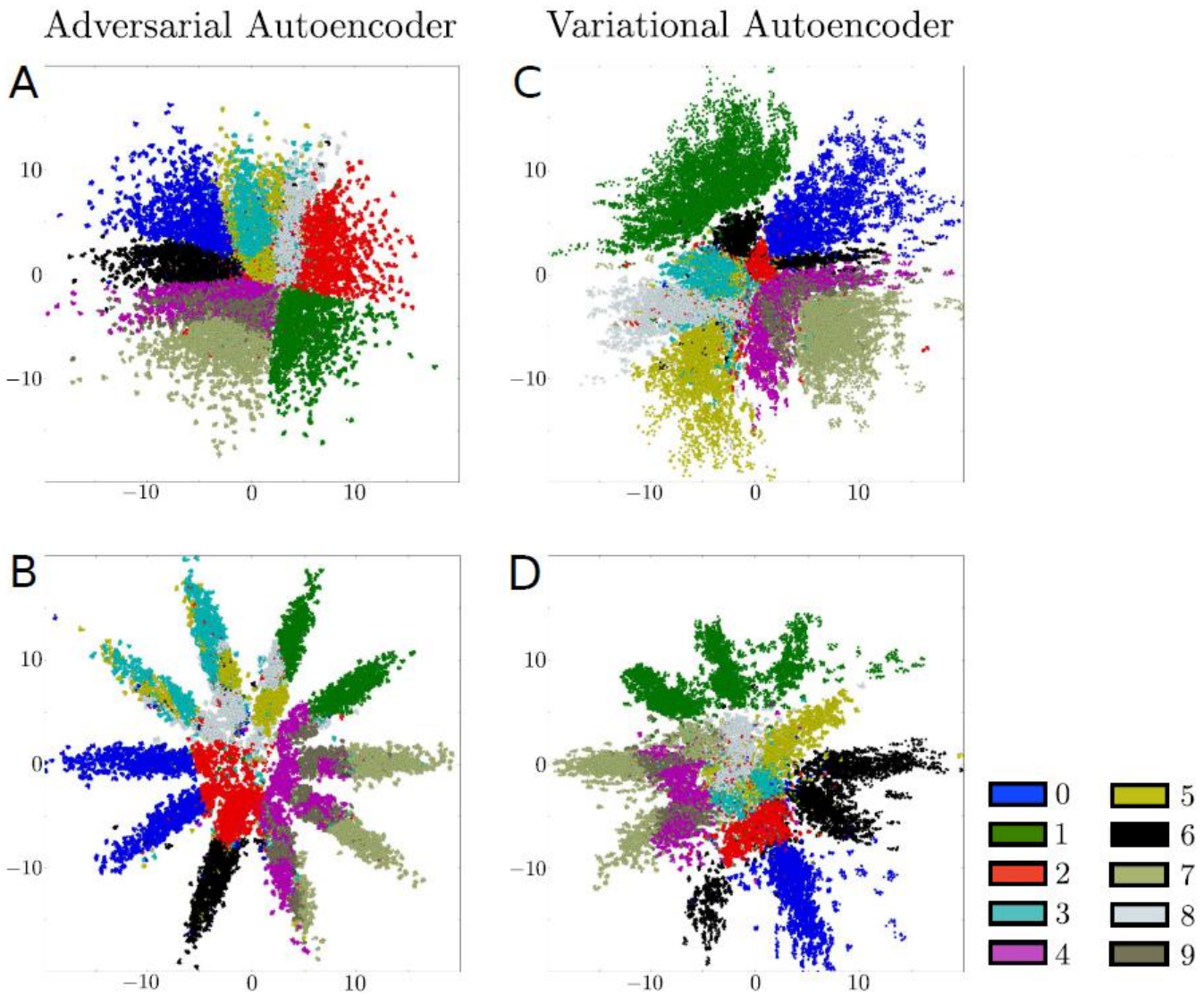
*논문에는 학습 절차 정의가 수식으로 제시되어 있지 않아 새로 정리한 내용

VAE VS AAE

$$p(z): \mathcal{N}(0, 5^2 I)$$

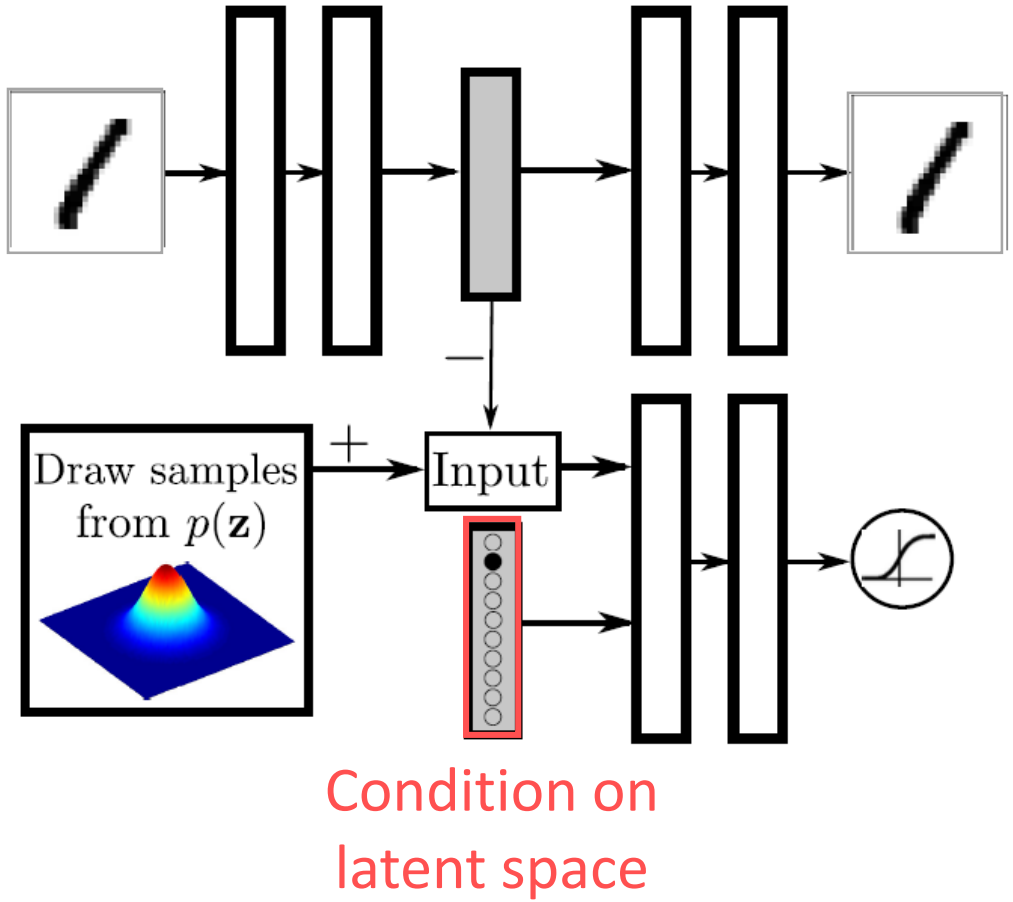
VAE는 자주 나오는 값을 파악하는 것 중시하여 빈 공간이 가끔 있는 반면,
AAE는 분포의 모양을 중시하여 빈 공간이 상대적으로 적다.

$$p(z): mixture\ of\ 10\ gaussians$$



Adversarial Autoencoders : https://arxiv.org/abs/1511.05644

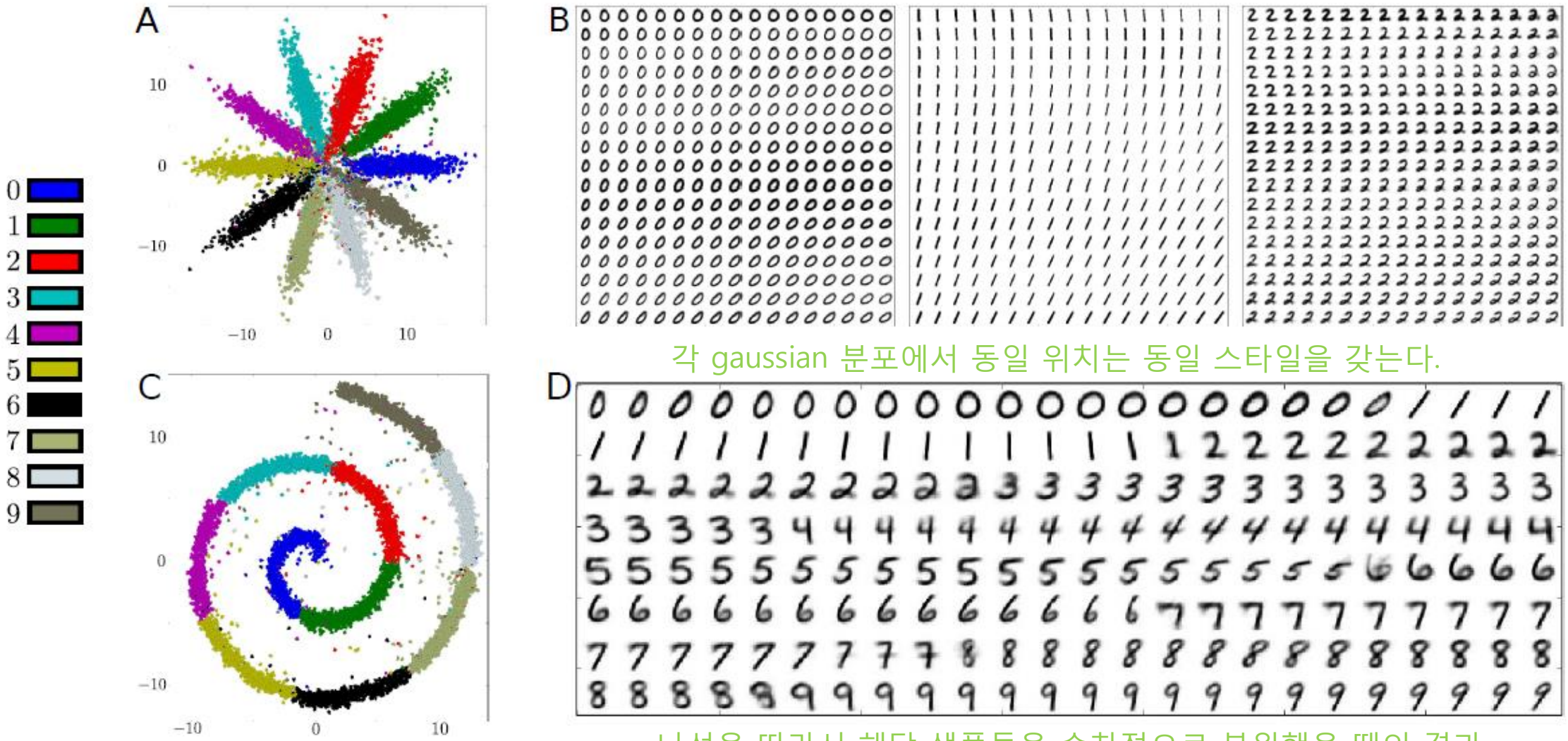
Incorporating Label Information in the Adversarial Regularization



- Discriminator에 prior distribution에서 뽑은 샘플이 입력으로 들어갈 때는 해당 샘플이 어떤 label을 가져야 하는지에 대한 condition을 Discriminator에 넣어준다.
- Discriminator에 posterior distribution에서 뽑은 샘플이 입력으로 들어갈 때는 해당 이미지에 대한 label을 Discriminator에 넣어준다.
- 특정 label의 이미지는 Latent space에서 의도된 구간으로 맵핑된다.

Adversarial Autoencoders : https://arxiv.org/abs/1511.05644

Incorporating Label Information in the Adversarial Regularization



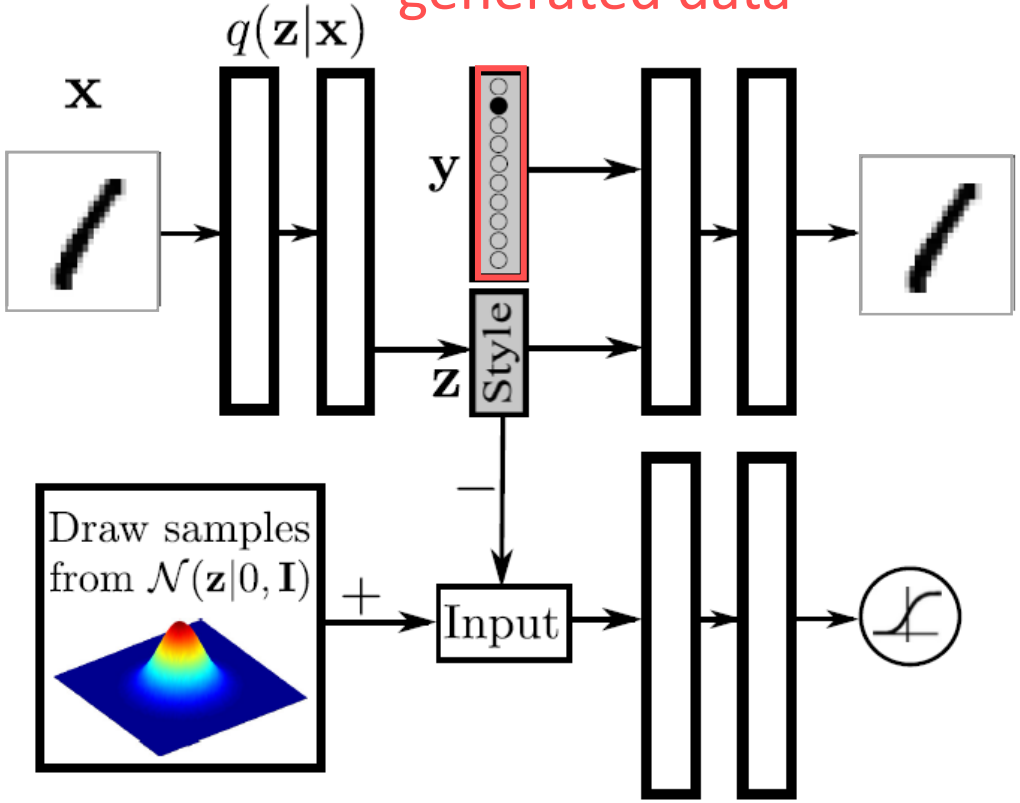
각 gaussian 분포에서 동일 위치는 동일 스타일을 갖는다.

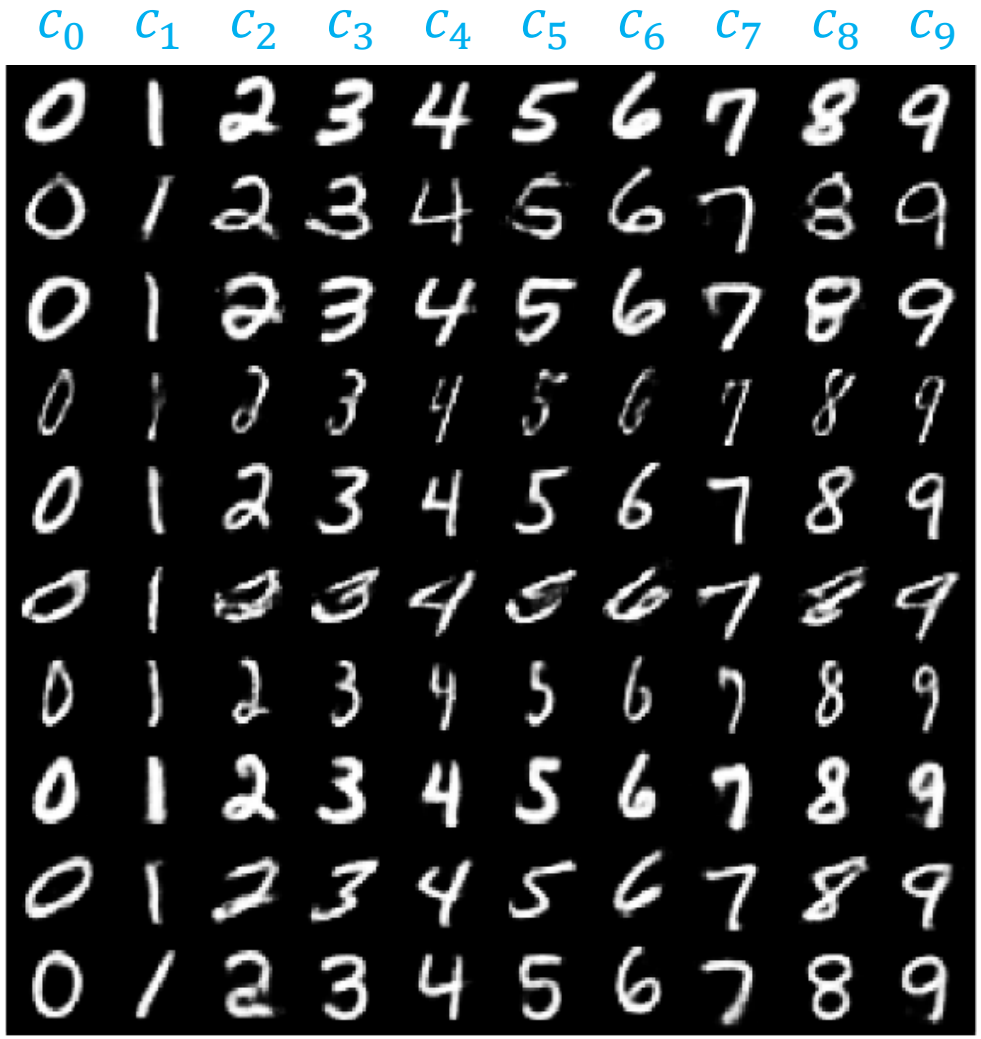
나선을 따라서 해당 샘플들을 순차적으로 복원했을 때의 결과

Adversarial Autoencoders : https://arxiv.org/abs/1511.05644

Supervised Adversarial Autoencoders

Condition on generated data

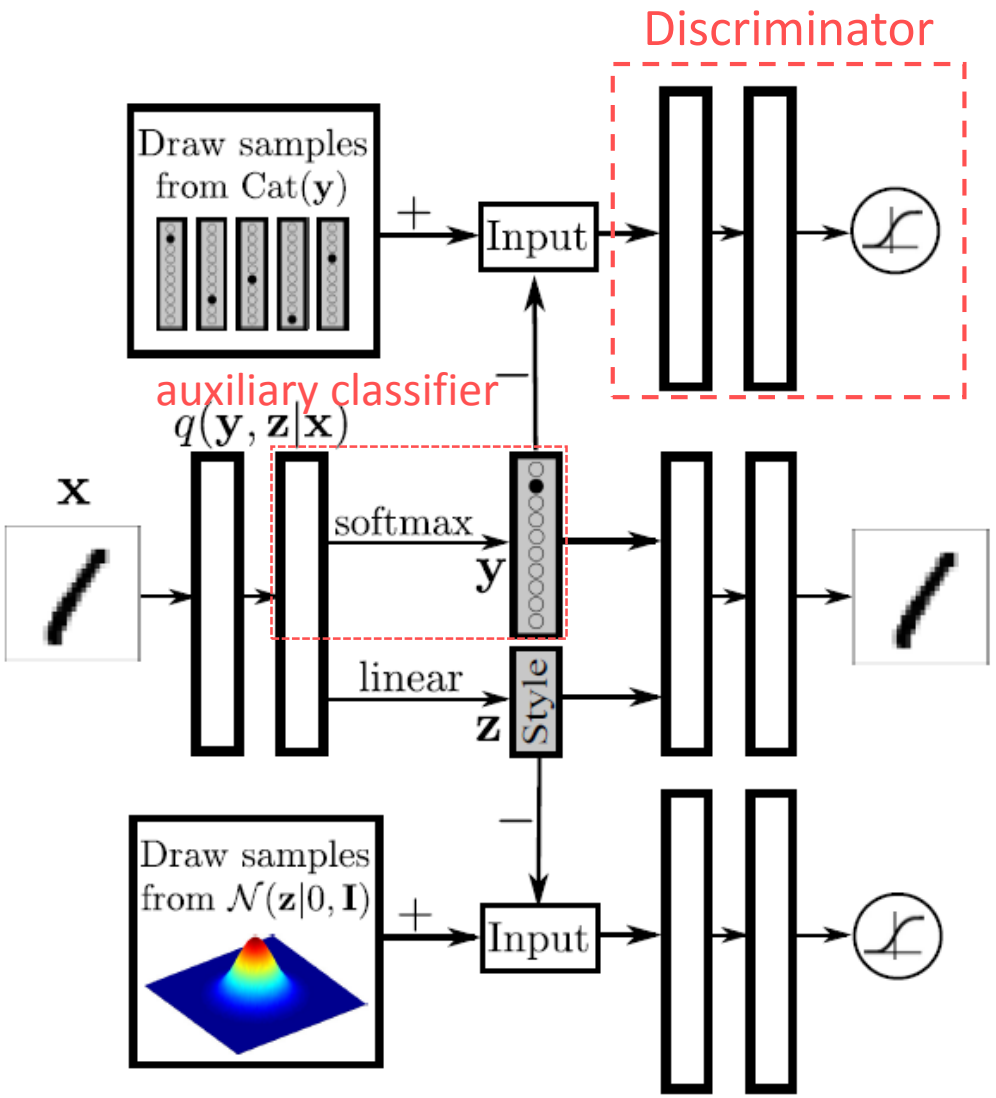




각 행은 동일 z값

Adversarial Autoencoders : https://arxiv.org/abs/1511.05644

Semi-Supervised Adversarial Autoencoders



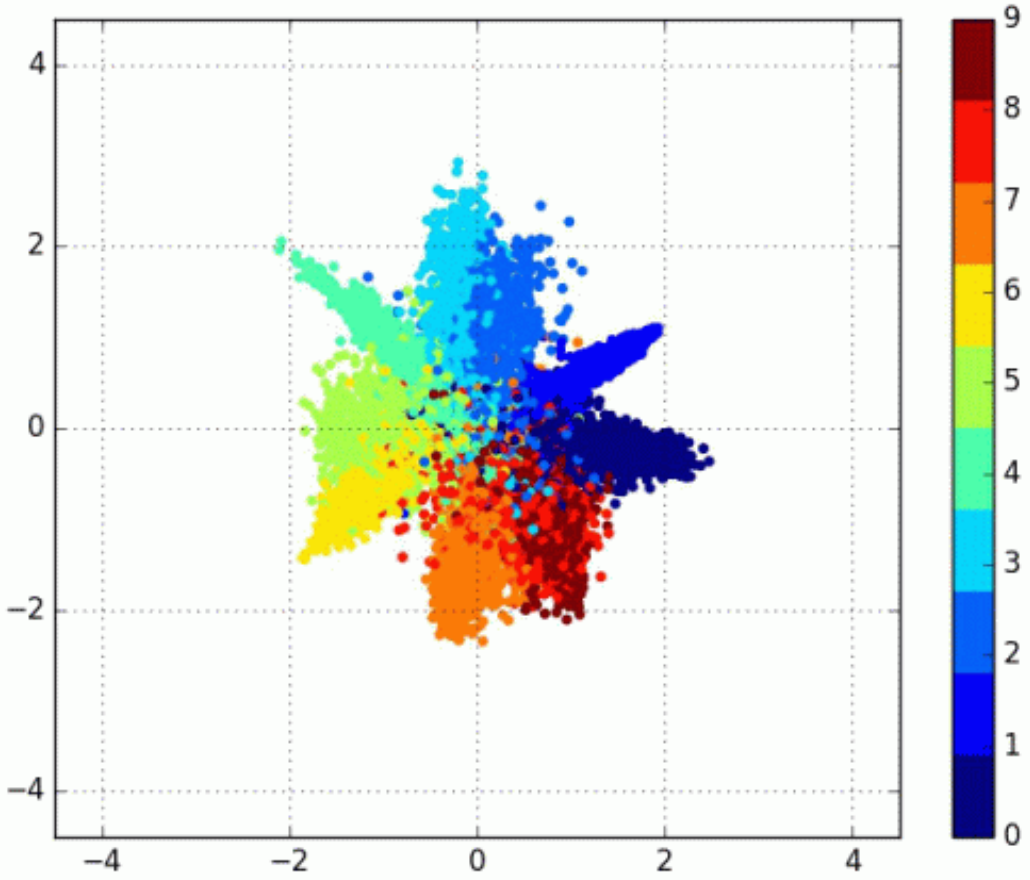
- Label이 제공되지 않을 경우에는 auxiliary classifier를 통해 label을 예측하고 이 예측이 맞는지를 확인하기 위한 discriminator를 추가로 학습시킨다.

	MNIST (100)	MNIST (1000)	MNIST (All)
NN Baseline	25.80	8.73	1.25
VAE (M1) + TSVM	11.82 (±0.25)	4.24 (±0.07)	-
VAE (M2)	11.97 (±1.71)	3.60 (±0.56)	-
VAE (M1 + M2)	3.33 (±0.14)	2.40 (±0.02)	0.96
VAT	2.33	1.36	0.64 (±0.04)
CatGAN	1.91 (±0.1)	1.73 (±0.18)	0.91
Ladder Networks	1.06 (±0.37)	0.84 (±0.08)	0.57 (±0.02)
ADGM	0.96 (±0.02)	-	-
Adversarial Autoencoders	1.90 (±0.10)	1.60 (±0.08)	0.85 (±0.02)

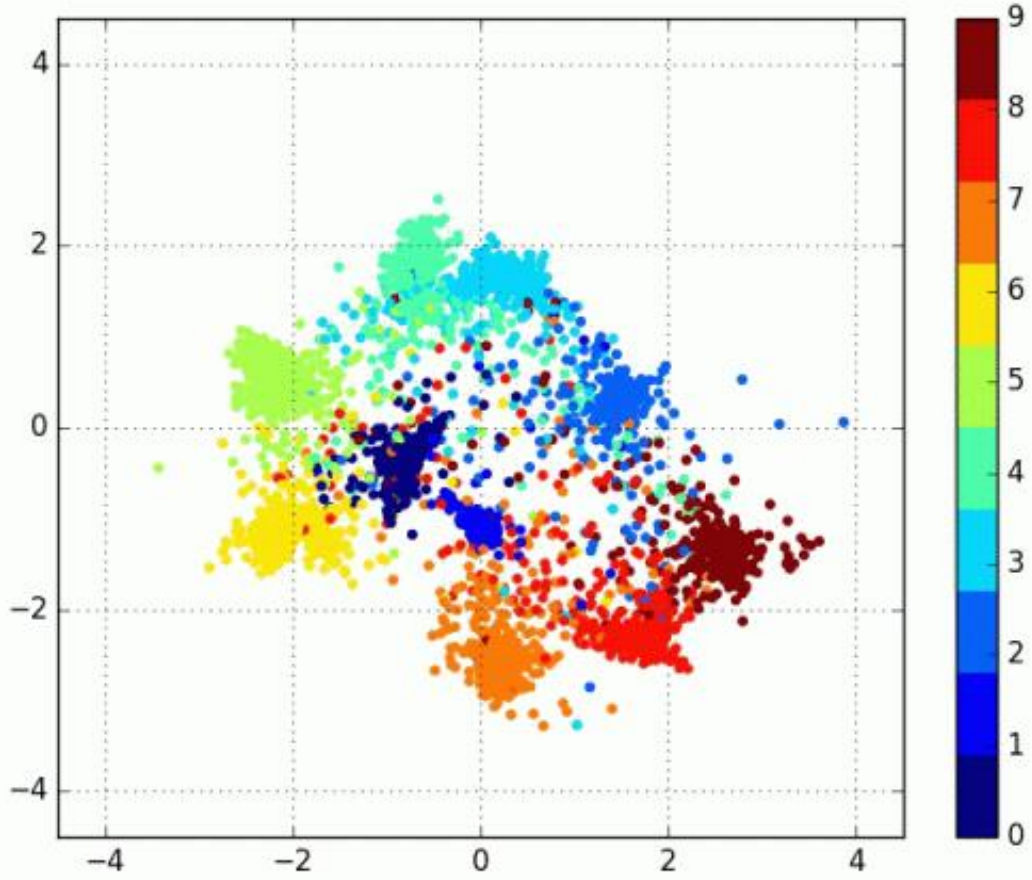
Adversarial Autoencoders : https://arxiv.org/abs/1511.05644

Incorporating Label Information in the Adversarial Regularization

실제 실험 결과 : mixture of 10 gaussians

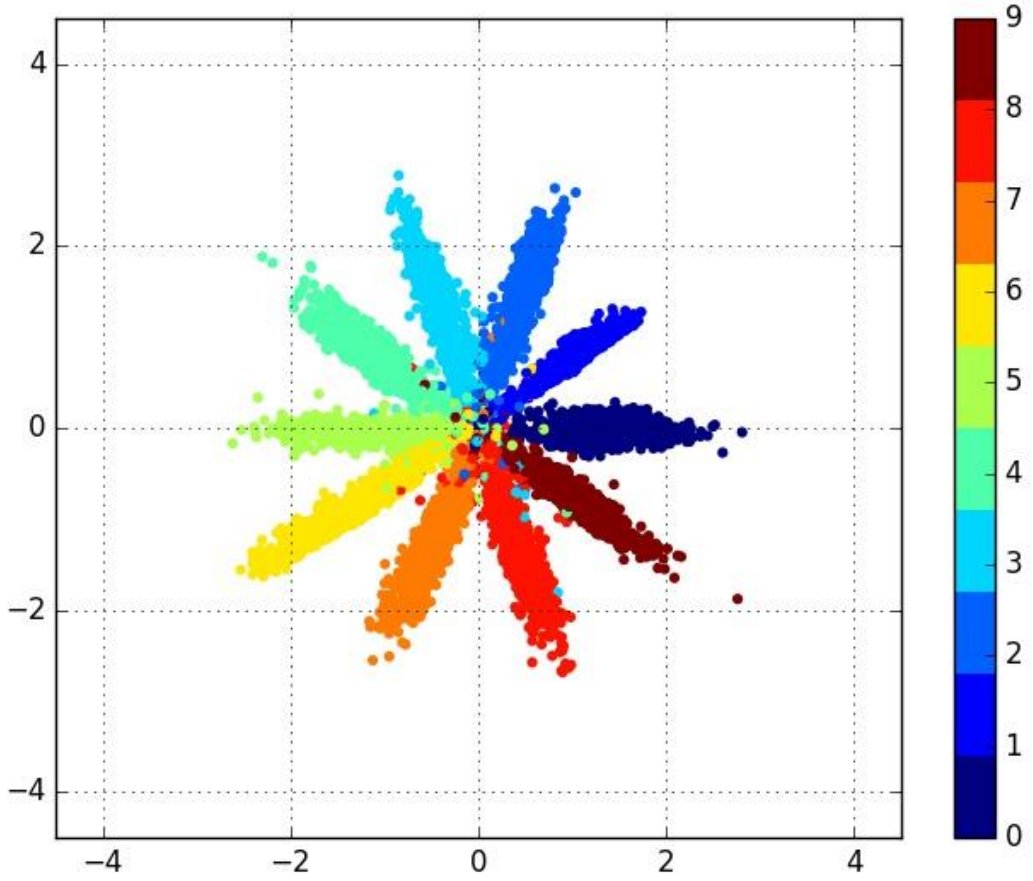


실제 실험 결과 : swiss roll

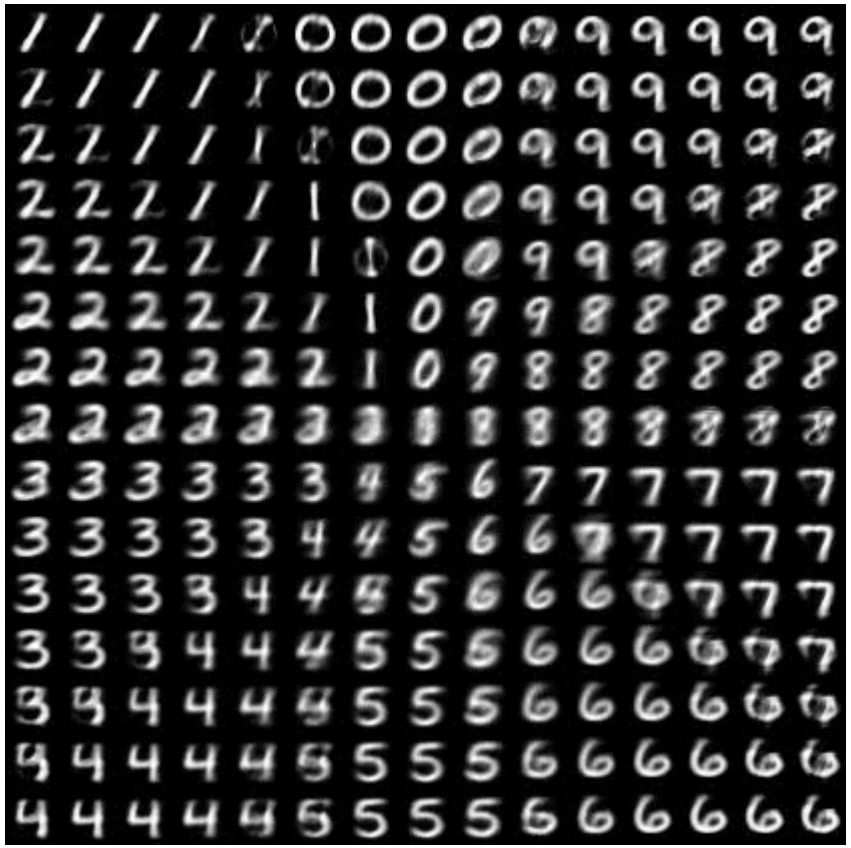


Incorporating Label Information in the Adversarial Regularization

실제 실험 결과 : mixture of 10 gaussians



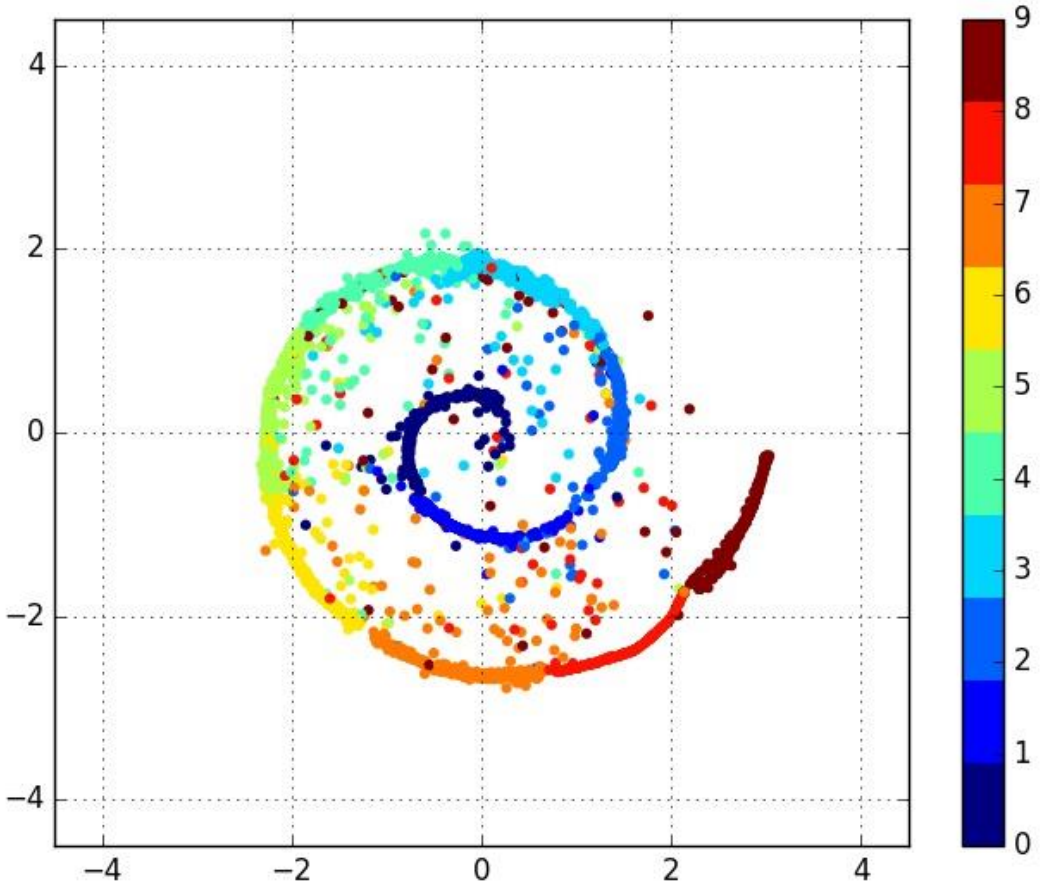
Learned Manifold



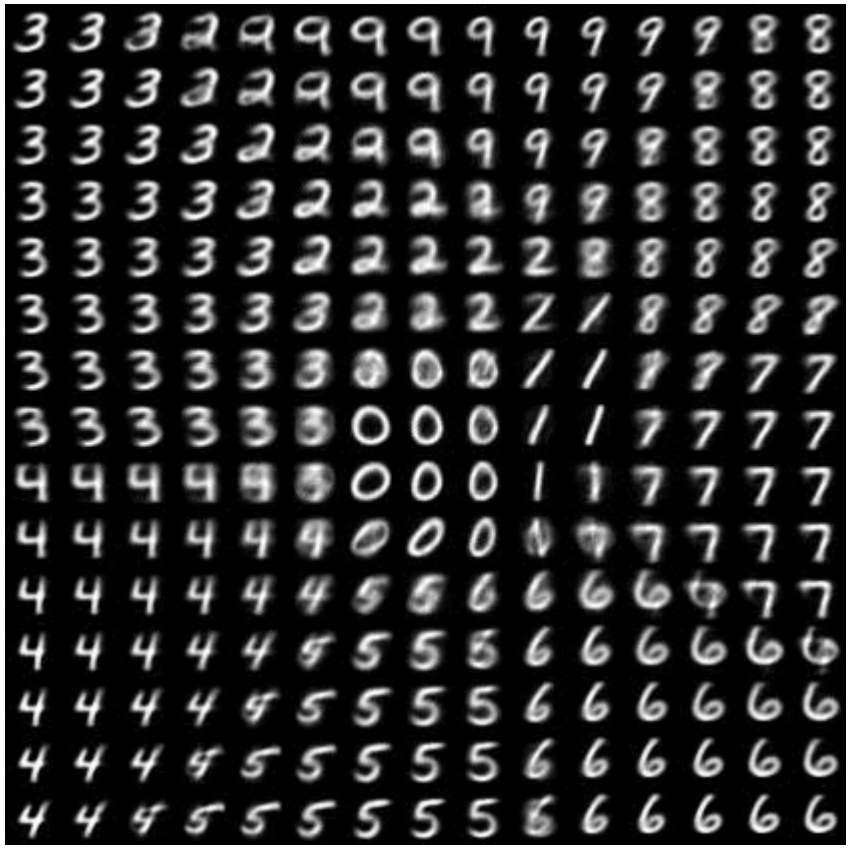
Generation

Incorporating Label Information in the Adversarial Regularization

실제 실험 결과 : swiss roll



Learned Manifold



Generation

01. Revisit Deep Neural Networks

02. Manifold Learning

03. Autoencoders

04. Variational Autoencoders

05. Applications

- Retrieval
- Generation
- Regression
- GAN+VAE

데이터 압축의 주 사용처인 retrieval에 관한 사례, 생성 모델로서의 VAE를 사용한 사례, 고차원 데이터의 regression을 위해 VAE를 사용한 사례 및 GAN 방법과 결합하여 사용되는 VAE들을 소개한다.

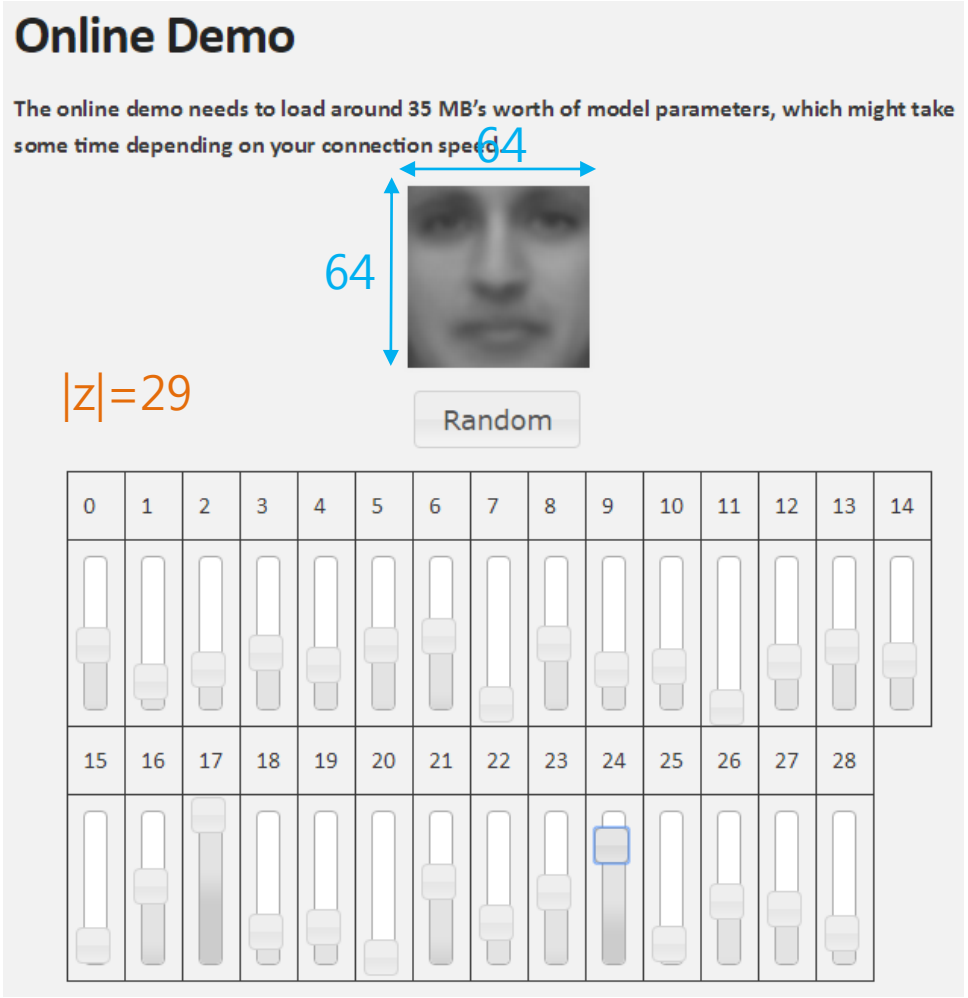
- Text
  - Semantic Hashing
    (Link) http://www.cs.utoronto.ca/~rsalakhu/papers/semantic_final.pdf
  - Dynamic Auto-Encoders for Semantic Indexing
    (Link) http://yann.lecun.com/exdb/publis/pdf/mirowski-nipsdl-10.pdf

- Image
  - Using Very Deep Autoencoders for Content-Based Image Retrieval
    (Link) http://nuyoo.utm.mx/~jjf/rna/A6%20Using%20Very%20Deep%20Autoencoders%20for%20Content-Based%20Image%20Retrieval.pdf
  - Autoencoding the Retrieval Relevance of Medical Images
    (Link) https://arxiv.org/pdf/1507.01251.pdf

- Sound
  - Retrieving Sounds by Vocal Imitation Recognition
    (Link)
    http://www.ece.rochester.edu/~zduan/resource/ZhangDuan_RetrievingSoundsByVocalImitationRecognition_MLSP15.pdf

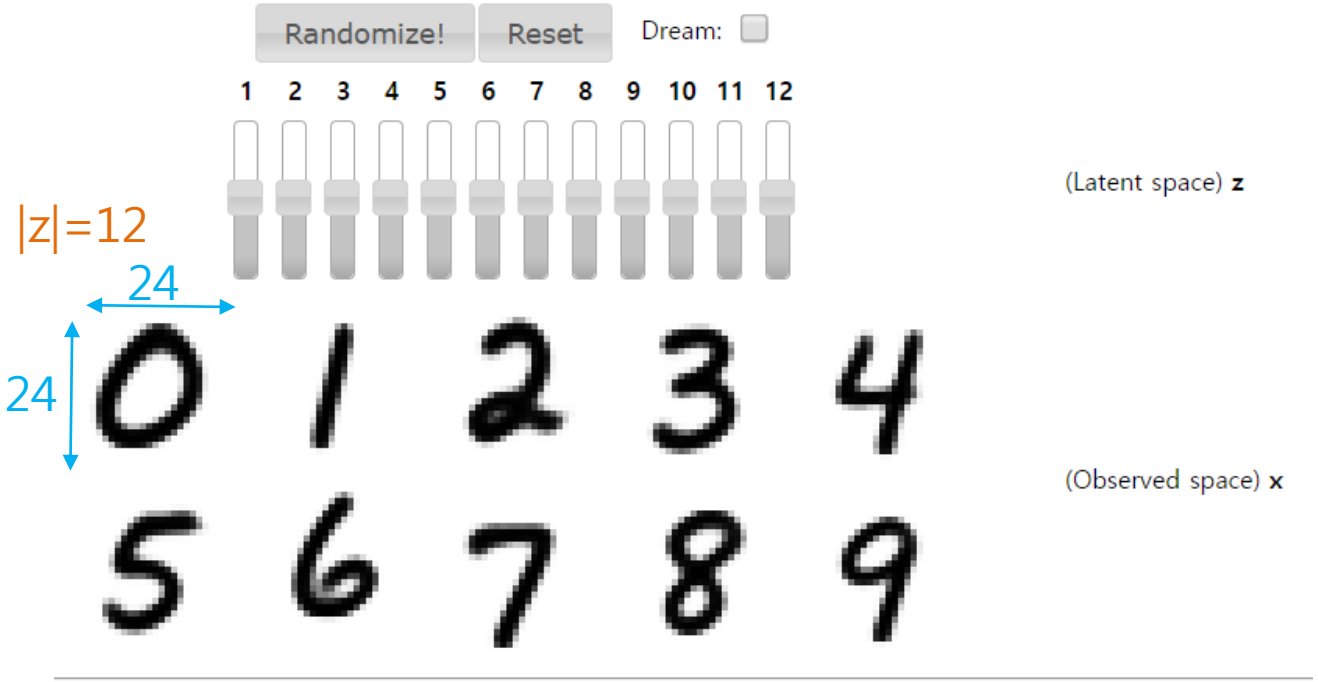
- 3D model
  - Deep Learning Representation using Autoencoder for 3D Shape Retrieval
    (Link) https://arxiv.org/pdf/1409.7164.pdf
  - Deep Signatures for Indexing and Retrieval in Large Motion Databases
    (Link) http://web.cs.ucdavis.edu/~neff/papers/MIG_2015_DeepSignature.pdf
  - DeepShape: Deep Learned Shape Descriptor for 3D Shape Matching and Retrieval
    (Link) http://www.cv-foundation.org/openaccess/content_cvpr_2015/papers/Xie_DeepShape_Deep_Learned_2015_CVPR_paper.pdf
- Multi-modal
  - Cross-modal Retrieval with Correspondence Autoencoder
    (Link) https://people.cs.clemson.edu/~jzwang/1501863/mm2014/p7-feng.pdf
  - Effective multi-modal retrieval based on stacked autoencoders
    (Link) http://www.comp.nus.edu.sg/~ooibc/crossmodalvldb14.pdf

Gray Face Generation

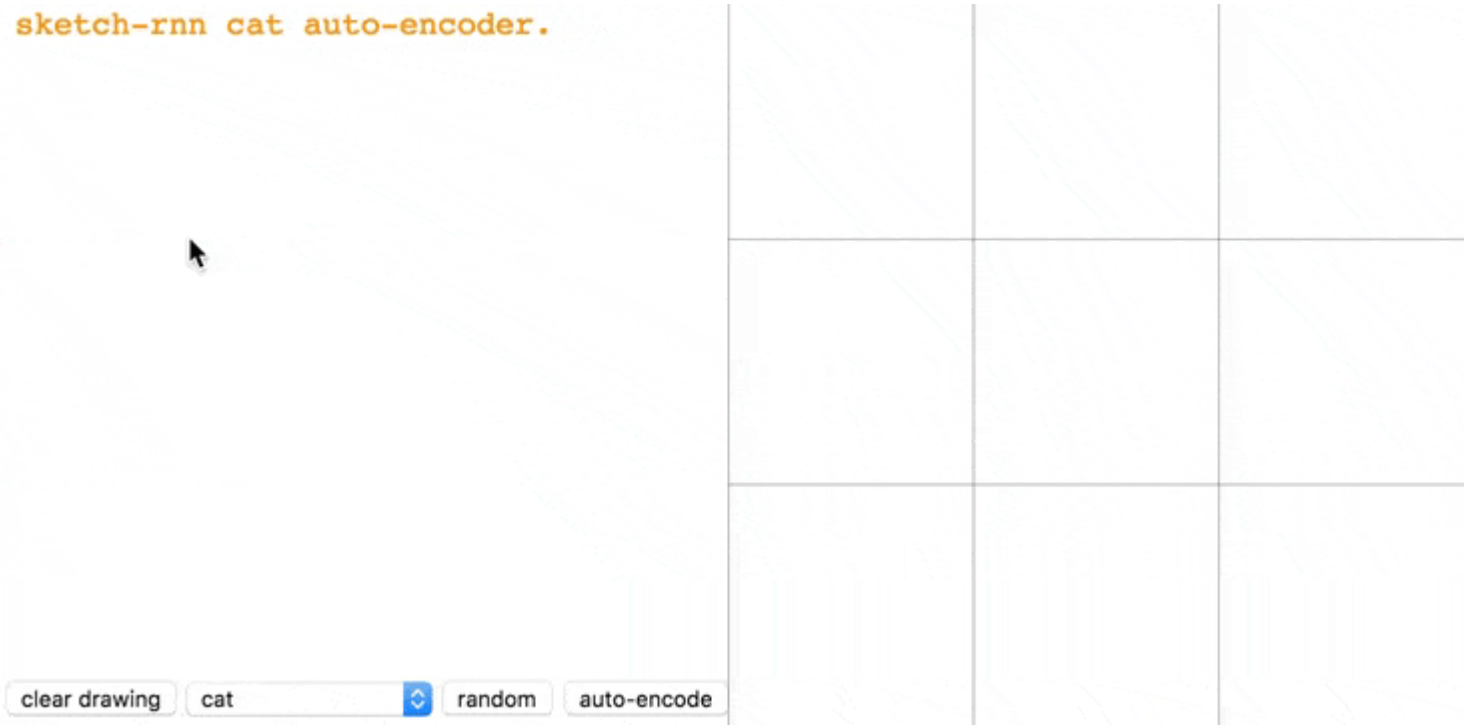


http://vdumoulin.github.io/morphing_faces/online_demo.html

Handwritten Digits Generation



http://www.dpkingma.com/sgvb_mnist_demo/demo.html



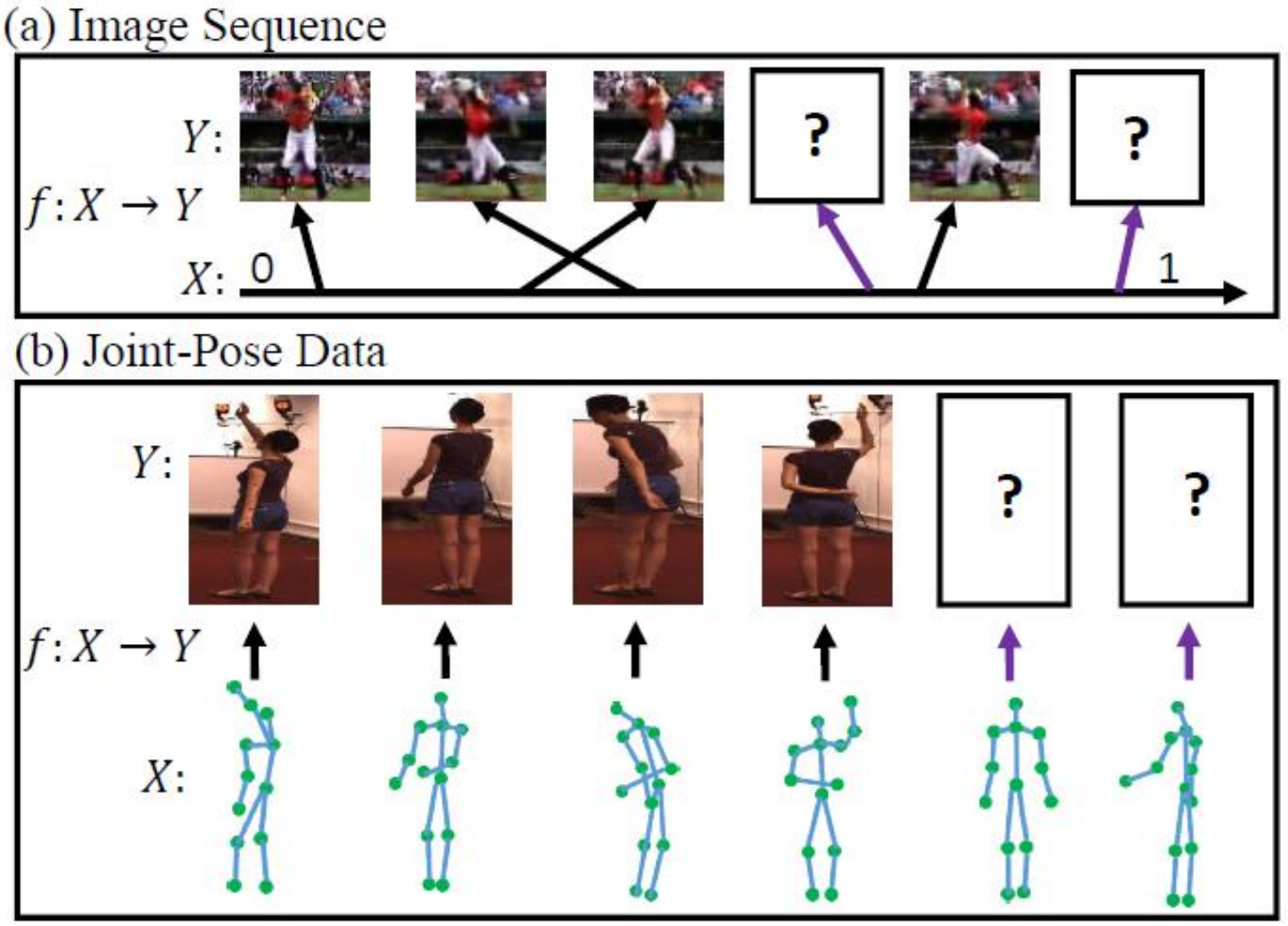
https://magenta.tensorflow.org/assets/sketch_rnn_demo/multi_vae.html

The model can also mimic your drawings and produce similar doodles. In the Variational Autoencoder Demo, you are to draw a **complete** drawing of a specified object. After you draw a complete sketch inside the area on the left, hit the *auto-encode* button and the model will start drawing similar sketches inside the smaller boxes on the right. Rather than drawing a perfect duplicate copy of your drawing, the model will try to mimic your drawing instead.

You can experiment drawing objects that are not the category you are supposed to draw, and see how the model interprets your drawing. For example, try to draw a cat, and have a model trained to draw crabs generate cat-like crabs. Try the Variational Autoencoder demo.

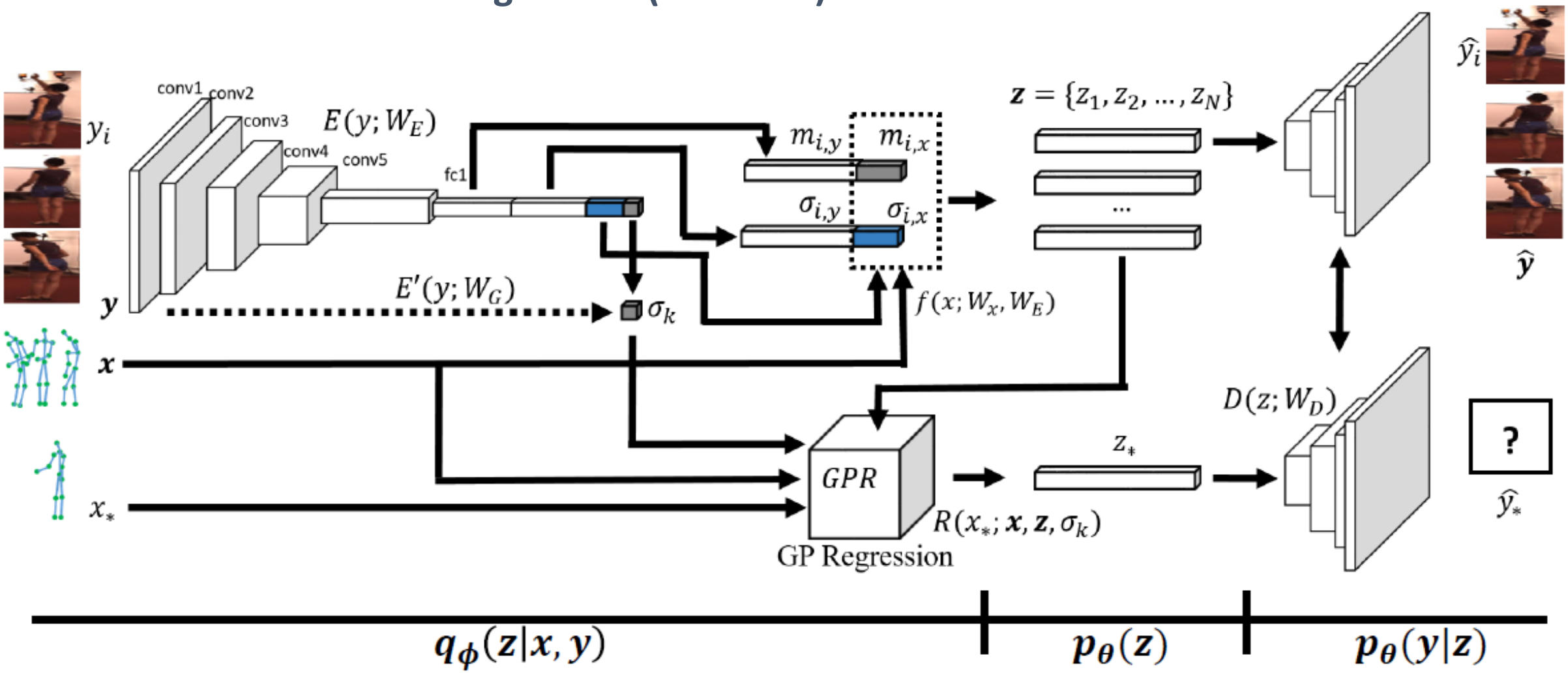
https://magenta.tensorflow.org/sketch-rnn-demo

Regression | Problem



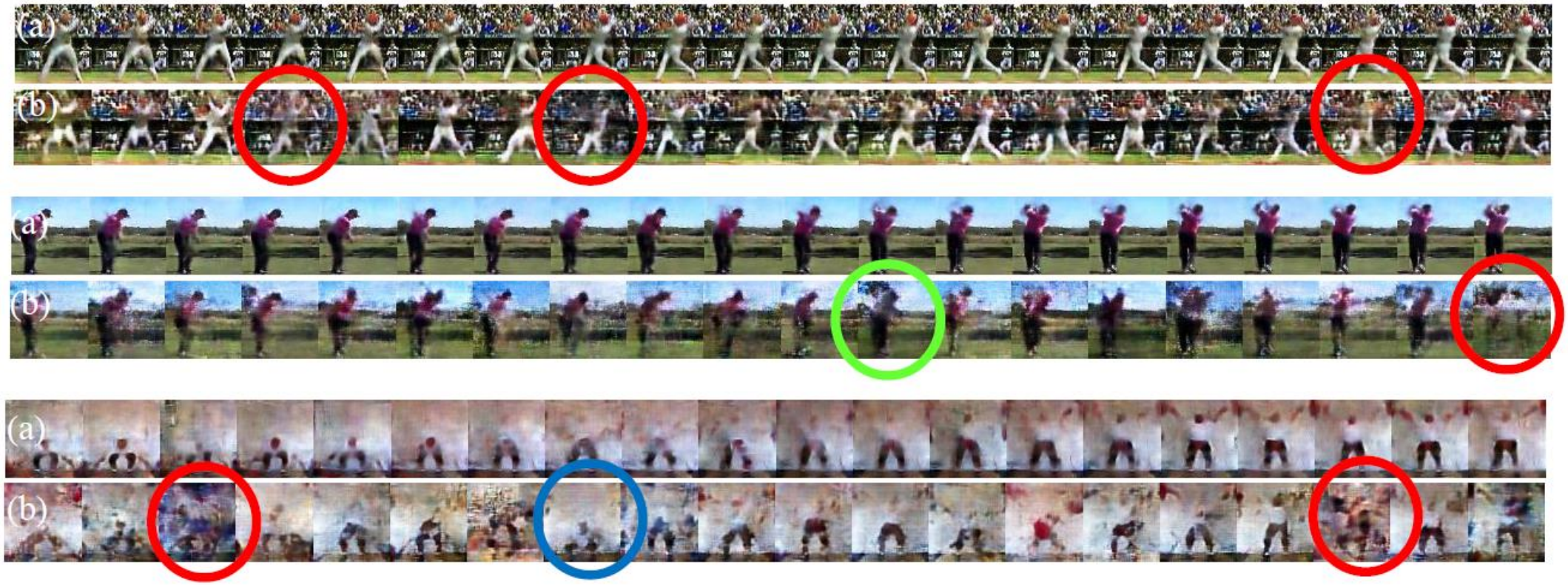
https://hyungjinchang.files.wordpress.com/2015/03/cvpr-2017-variational-paper.pdf

Variational Autoencoded Regression (CVPR '17)



https://hyungjinchang.files.wordpress.com/2015/03/cvpr-2017-variational-paper.pdf

Variational Autoencoded Regression (CVPR '17)

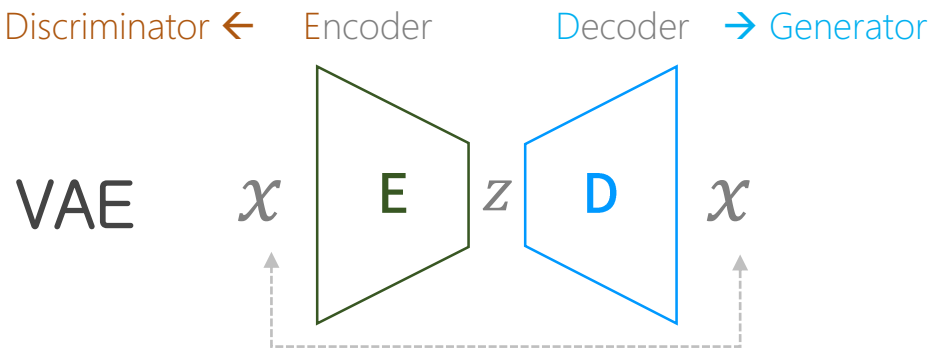


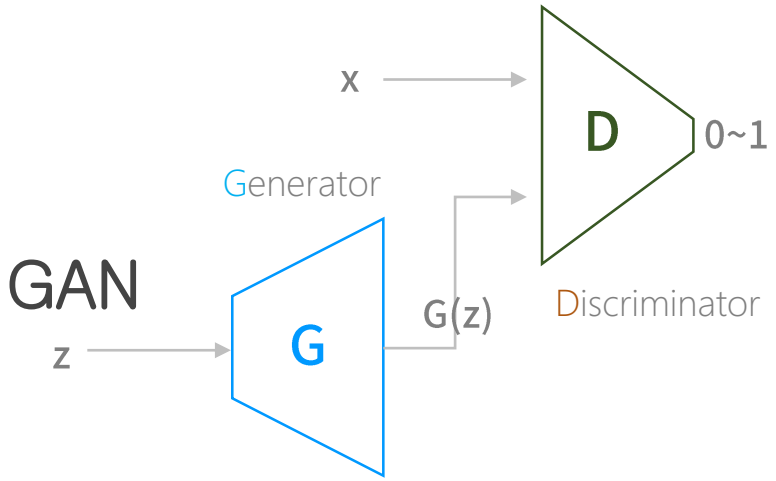
(a) proposed, (b) VAE + GPR

https://drive.google.com/drive/folders/0B8z5oUpB2Dysd3kxTHl0dDdLWUE

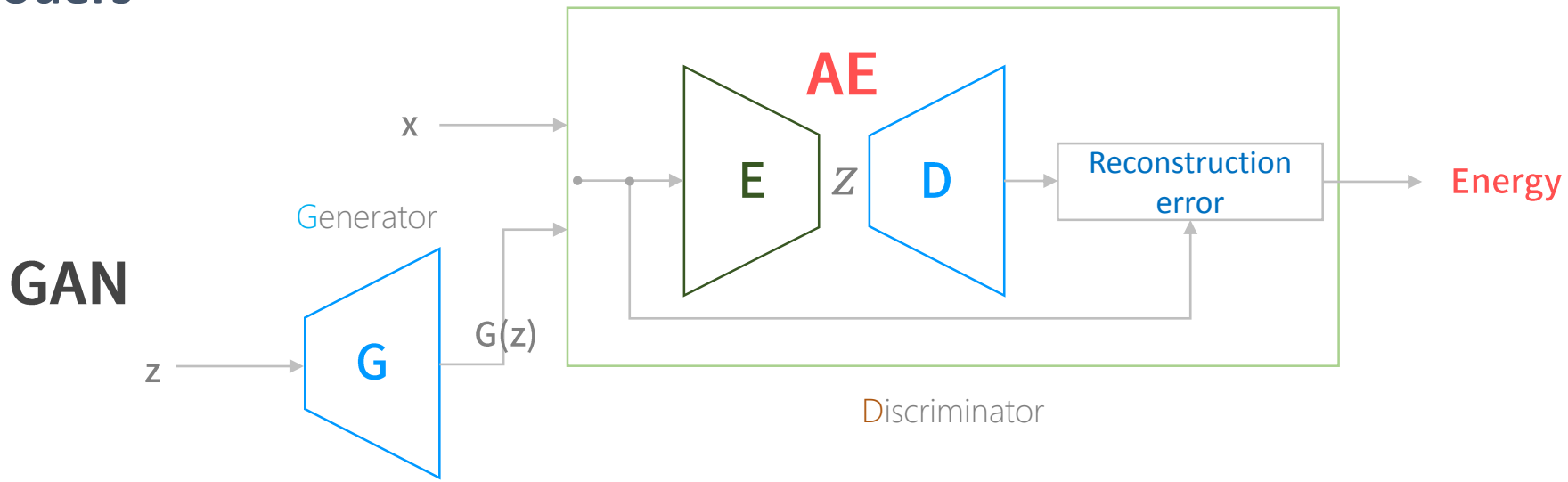
Comparison between VAE vs GAN

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





Regularized Autoencoders



We argue that the energy function (the discriminator) in the EBGAN framework is also seen as being regularized by having a generator producing the contrastive samples, to which the discriminator ought to give high reconstruction energies.

We further argue that the EBGAN framework allows more flexibility from this perspective, because: (i)-the regularizer (generator) is fully trainable instead of being handcrafted; (ii)-the adversarial training paradigm enables a direct interaction between the duality of producing contrastive sample and learning the energy function.

EBGAN : Energy-based Generative Adversarial Network '16.09
BEGAN : Boundary Equilibrium Generative Adversarial Networks '17.03

Multimodal Feature Learner

AI가 생성한 사진을 찾아보세요!

(1) (2) (3)

(1) This flower has overlapping pink pointed petals surrounding a ring of short yellow filaments

이 꽃은 짧은 노란색 필라멘트의 고리를 둘러싼 핑크색 뾰족한 꽃잎이 겹쳐져 있습니다

(2) This flower has upturned petals which are thin and orange with rounded edges

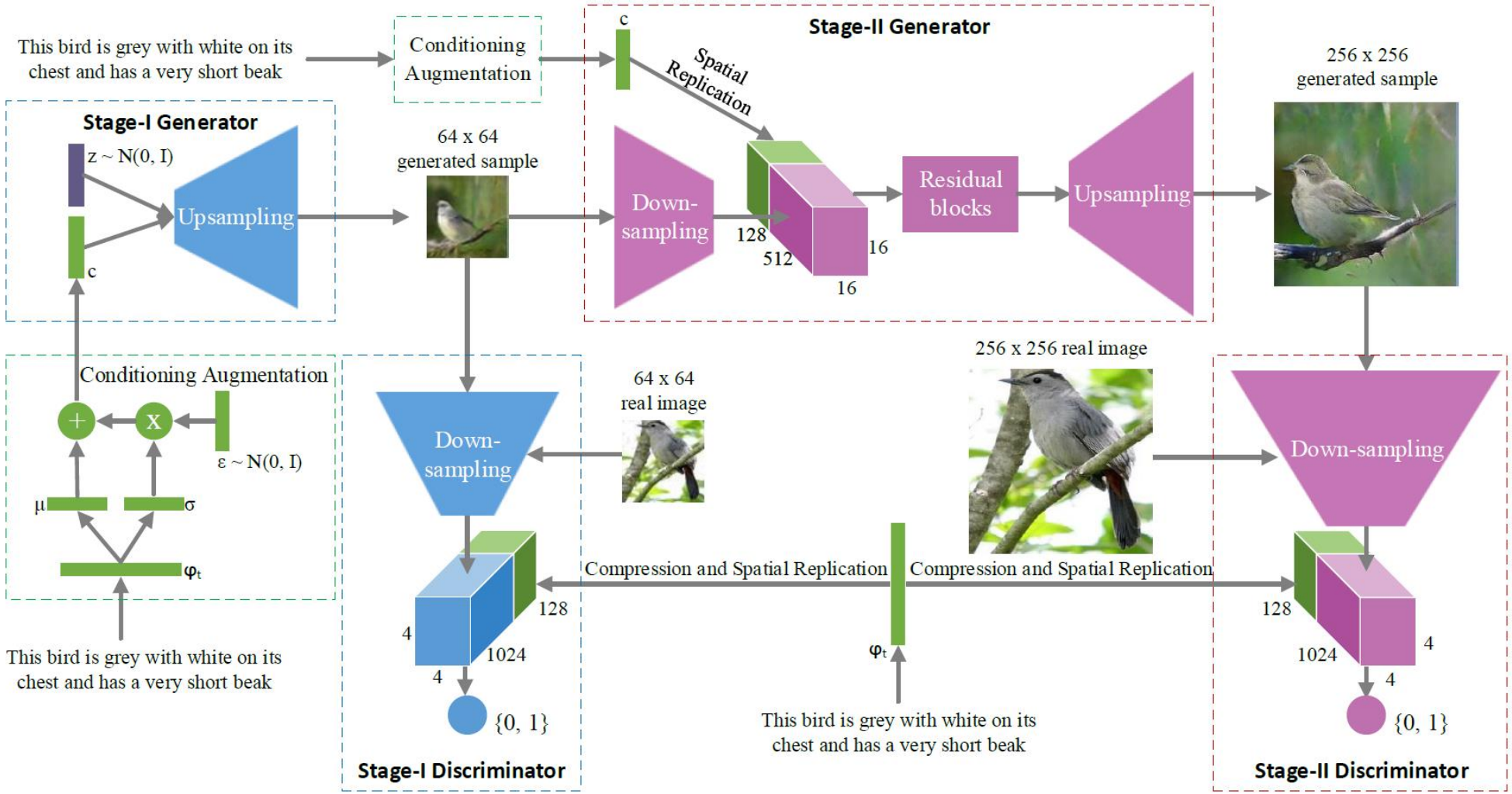
이 꽃은 둥근 모서리를 가진 얇고 오렌지색의 꽃잎이 위로 향해 있습니다

(3) A flower with small pink petals and a massive central orange and black stamen cluster

작은 분홍색 꽃잎들과 중심에 다수의 오렌지색과 검은 색 수술 군집이 있는 꽃

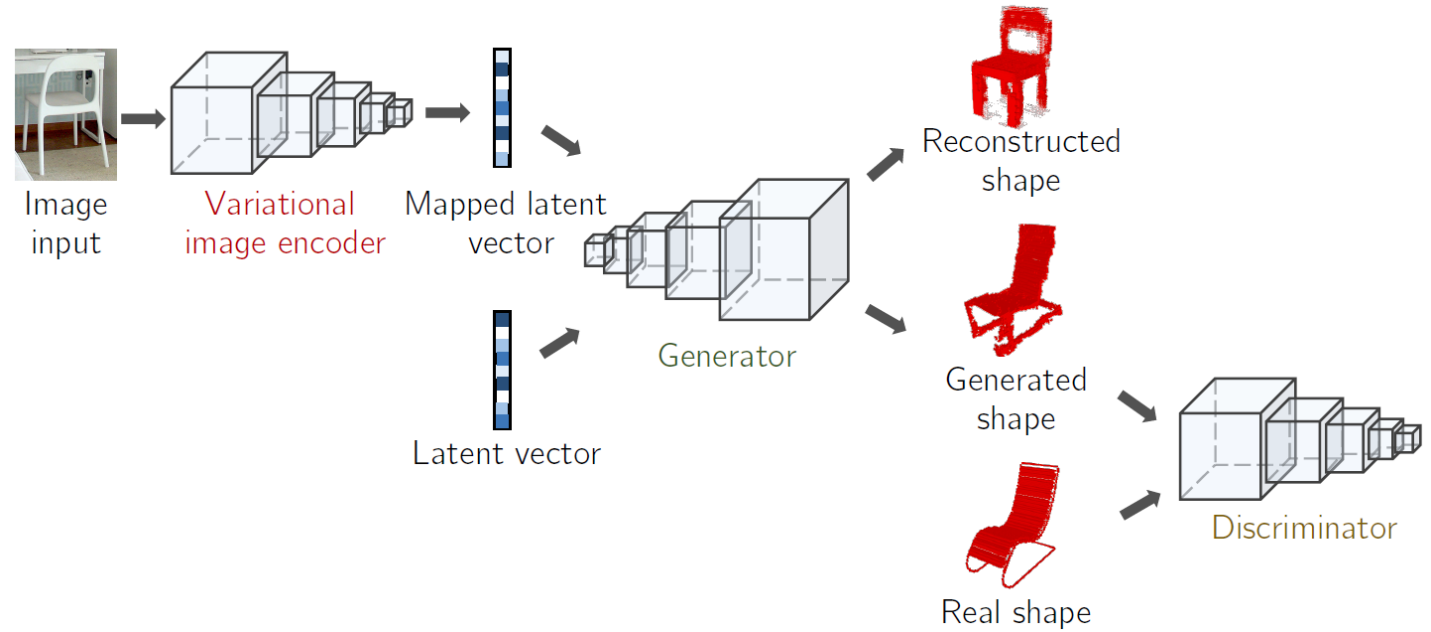
StackGAN : Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks , '16.12

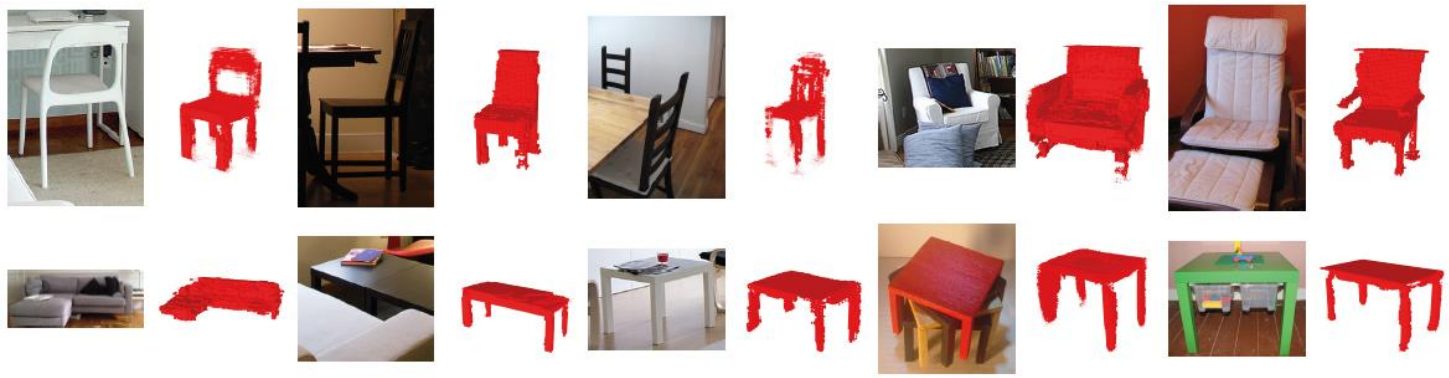
Multimodal Feature Learner



StackGAN : Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks , '16.12

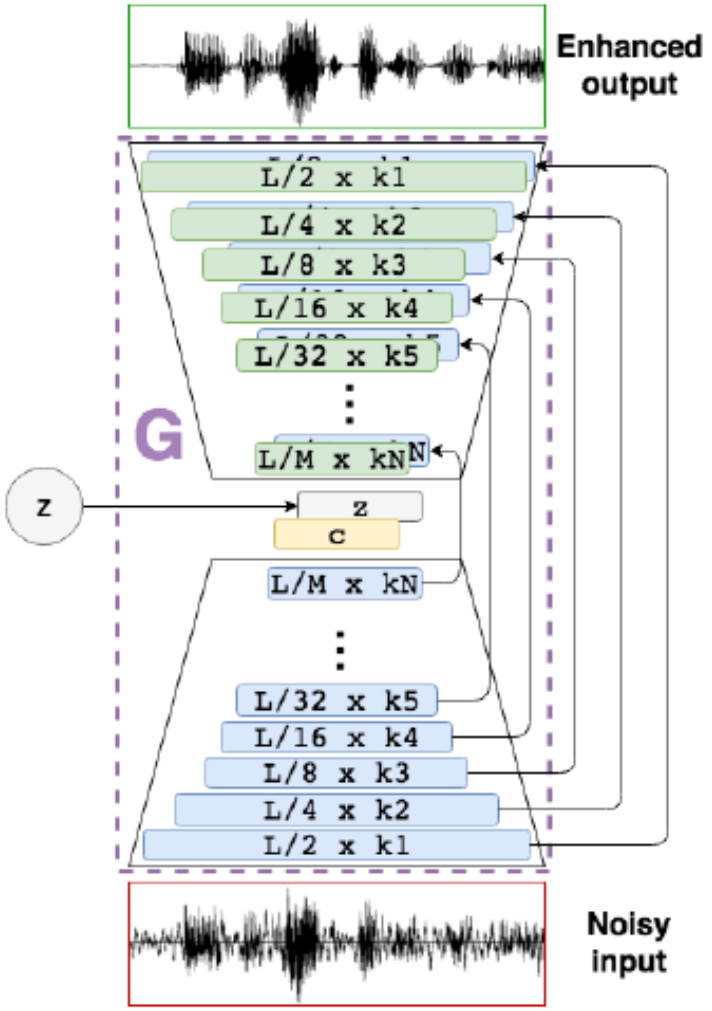
Multimodal Feature Learner

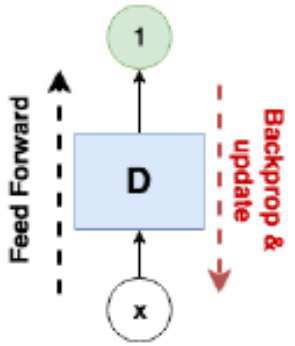


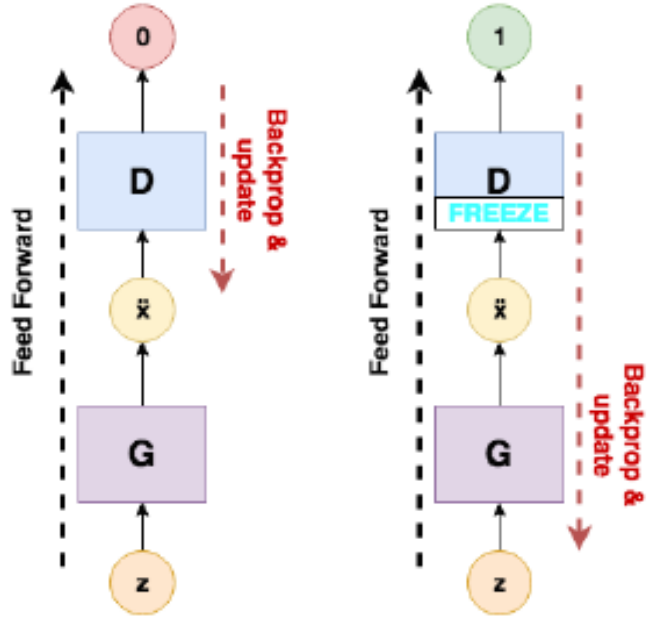


Learning a Probabilistic latent Space of Object Shapes via 3D Generative-Adversarial Modeling (3D-GAN), '16.10

Denoising







There will be no repeat of that performance, that I can guarantee.

before after

Nothing is safe.

before after

SEGAN: Speech Enhancement Generative Adversarial Network '17. 03. 28

End of Document