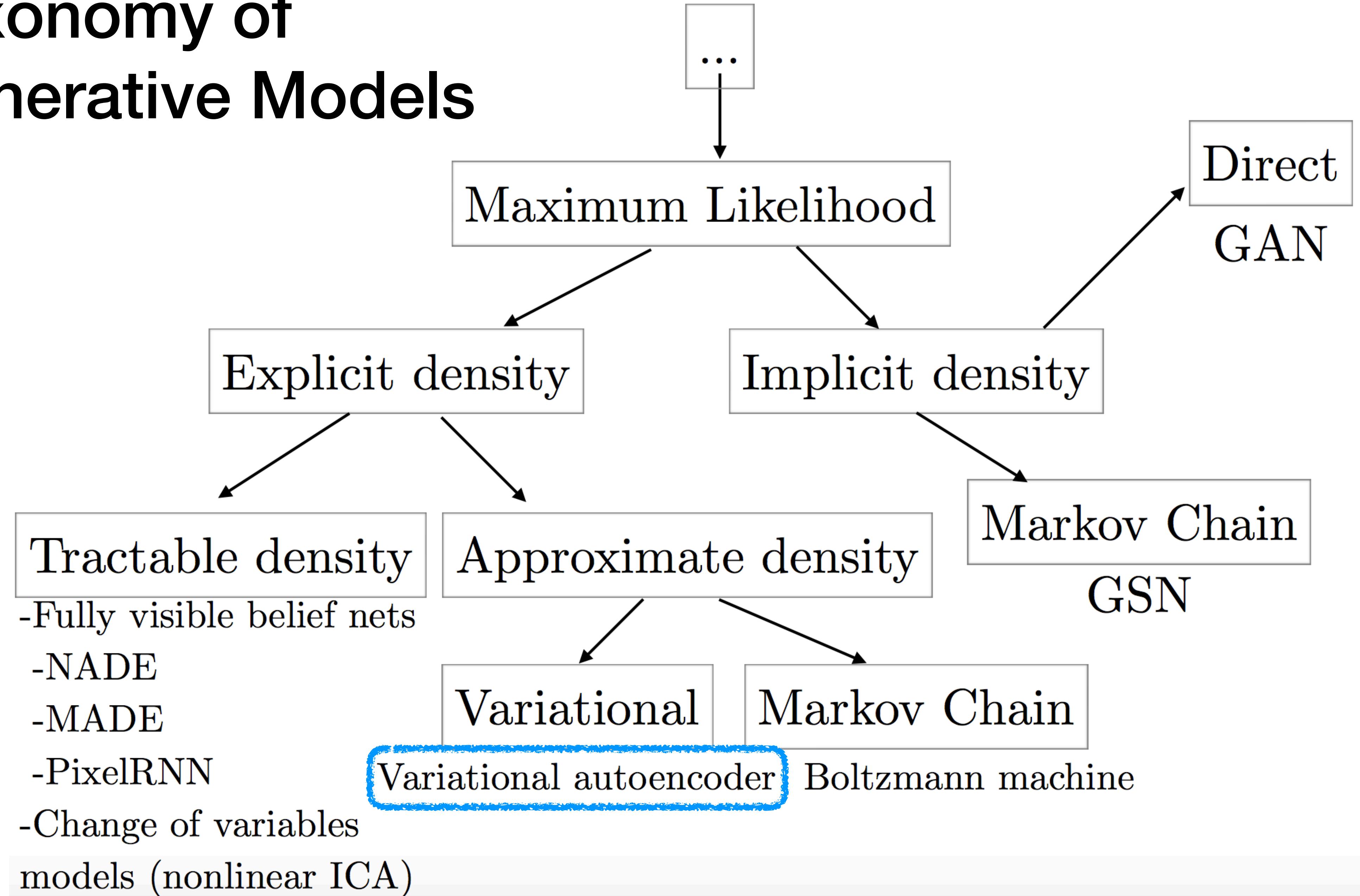


# Latent Variable Models

2021  
Multicampus

**SooChul Park**

# Taxonomy of Generative Models



# Variational Auto-Encoders

# Variational AutoEncoders - Preview

- Variational AutoEncoders는 Auto-Encoding Variational Bayes 논문에서 제안된 모델입니다.
- VAE는 주어진 data를 z space에 encoding하는 encoder와 encoding된 z vector를 다시 data로 복원하는 decoder로 구성됩니다.
- 일반적인 auto encoder와 구조는 거의 동일하지만, VAE는 z space에서 encoding된 z vector들의 분포를 Gaussian distribution 등으로 제한할 수 있다는 장점을 가지고 있습니다.
- z space 상의 분포를 제한할 수 있으면 해당 분포에서 z vector를 샘플링하고 decoder를 이용하여 data를 복원할 수 있습니다. 따라서 auto encoder와 달리 vae는 generative model이라고 할 수 있습니다.



# Variational AutoEncoders - Preview

- **Maximum likelihood**

VAE에서 data  $\mathbf{x}$ 의 marginal likelihood  $p_\theta(\mathbf{x})$ 는 prior  $p_\theta(\mathbf{z})$ 에 대한 likelihood  $p_\theta(\mathbf{x} | \mathbf{z})$ 의 expectation으로 계산합니다.

$$\bullet \quad p_\theta(\mathbf{x}) = \int p_\theta(\mathbf{z})p_\theta(\mathbf{x} | \mathbf{z})d\mathbf{z}$$

- 통상적으로 prior  $p_\theta(\mathbf{z})$ 는 standard Gaussian distribution  $\mathbf{N}(\mathbf{0}, \mathbf{I})$ 으로 두고, likelihood  $p_\theta(\mathbf{x} | \mathbf{z})$ 는 Gaussian distribution이나 Bernoulli distribution으로 설정합니다.
- Likelihood  $p_\theta(\mathbf{x} | \mathbf{z})$ 의 mean을  $\mathbf{z}$ 의 linear transform된 형태 즉,  $\mathbf{W}\mathbf{z} + \boldsymbol{\mu}$ 으로 설정하는 linear-Gaussian model의 경우, 위의 marginal likelihood 적분식을 closed form으로 계산할 수 있습니다. (Pattern recognition and machine learning p.93)
- 하지만 VAE와 같이 neural network와 같이 복잡한 함수를 사용하여  $\mathbf{z}$ 를 transform하는 경우, 적분식을 계산하는 것은 어렵습니다. (intractable)

# Variational AutoEncoders - Preview

- **EM algorithm**

주어진 data  $\mathbf{x}$ 를 낳게한  $\mathbf{z}$  vector의 분포인 posterior  $p_{\theta}(\mathbf{z} | \mathbf{x})$ 를 구할 수 있다면 EM algorithm을 사용하여 marginal likelihood  $p_{\theta}(\mathbf{x})$ 를 최대화 할 수 있습니다.

[https://en.wikipedia.org/wiki/Expectation–maximization\\_algorithm](https://en.wikipedia.org/wiki/Expectation–maximization_algorithm)

- posterior  $p_{\theta}(\mathbf{z} | \mathbf{x})$ 는 다음과 같이 prior  $p_{\theta}(\mathbf{z})$ 와 likelihood  $p_{\theta}(\mathbf{x} | \mathbf{z})$ 를 곱하고 evidence  $p_{\theta}(\mathbf{x})$ 로 나누어 계산할 수 있습니다.

$$p_{\theta}(\mathbf{z} | \mathbf{x}) = \frac{p_{\theta}(\mathbf{z})p_{\theta}(\mathbf{x} | \mathbf{z})}{p_{\theta}(\mathbf{x})}$$

- 앞서본 바와 마찬가지로 Likelihood  $p_{\theta}(\mathbf{x} | \mathbf{z})$ 의 mean을  $\mathbf{z}$ 의 linear transform된 형태 즉,  $\mathbf{Wz} + \boldsymbol{\mu}$ 으로 설정하는 linear-Gaussian model의 경우, posterior도 Gaussian 꼴로 나오고 이를 closed form으로 구할 수 있습니다.

(Pattern recognition and machine learning p.93)

- 하지만 VAE와 같이 neural network와 같이 복잡한 함수를 사용하여  $\mathbf{z}$ 를 transform하는 경우, posterior를 closed form으로 구하는 것은 어렵습니다. (intractable)



# Variational AutoEncoders - Preview

- **Variational Inference**

VAE에서는 Maximum likelihood을 사용하기 위해서 marginal likelihood  $p_\theta(\mathbf{x})$ 를 구하는 것도 어렵고, EM algorithm을 사용하기 위해서 posterior  $p_\theta(\mathbf{z} | \mathbf{x})$ 를 구하는 것도 어렵습니다.

- 이를 해결하기 위해 EM algorithm의 확장된 형태인 variational inference, variational EM algorithm의 방법을 사용합니다.
- Variational inference에서는 true posterior  $p_\theta(\mathbf{z} | \mathbf{x})$ 를 직접 구하는 대신 variational posterior  $q_\phi(\mathbf{z} | \mathbf{x})$ 를 사용합니다.
- 데이터  $\mathbf{x}$ 가 주어져 있을 때, log marginal likelihood  $\log p_\theta(\mathbf{x})$ 는 다음과 같이 전개됩니다.

$$\log p_\theta(\mathbf{x}) = ELBO + D_{KL}(q_\phi(\mathbf{z} | \mathbf{x}) \| p_\theta(\mathbf{z} | \mathbf{x}))$$

$$ELBO = \mathbf{E}_{q_\phi(\mathbf{z} | \mathbf{x})}[p_\theta(\mathbf{x}, \mathbf{z}) / q_\phi(\mathbf{z} | \mathbf{x})]$$

- 데이터  $\mathbf{x}$ 와 model parameter  $\theta$ 가 주어졌을 때,  $\log p_\theta(\mathbf{x})$ 는 고정된 값이므로 variational parameter  $\phi$ 를 조정하여 ELBO를 최대화하면 variational posterior  $q_\phi(\mathbf{z} | \mathbf{x})$ 를 true posterior  $p_\theta(\mathbf{z} | \mathbf{x})$ 에 가깝게 만듭니다.
- 비슷하게 model parameter  $\theta$ 를 조정하여 ELBO를 최대화하면 ELBO는 marginal likelihood  $\log p_\theta(\mathbf{x})$ 의 하한이므로 marginal likelihood를 최대화하는 결과를 낳습니다.



# Variational AutoEncoders - Preview

- VAE에서는 variational inference를 이용하여 marginal likelihood가 아닌 ELBO를 objective function으로 갖습니다.
- Encoder는 variational posterior  $q_{\phi}(z | x)$ 를 구현합니다. 즉, data  $x$ 를 encoder의 입력으로 받고 z space상의 vector  $z$ 의 분포의 parameter mean  $\mu$ 와 standard deviation  $\sigma$ 를 출력합니다.
- Decoder는 conditional distribution  $p_{\theta}(x | z)$ 를 구현합니다. 즉, z space상의 vector  $z$ 를 입력 받아, x space 상의 data  $x$ 의 분포의 parameter를 출력합니다.
- data  $x$ 의 분포는 Bernoulli나 Gaussian을 주로 사용합니다. Gaussian의 경우 standard deviation  $\sigma$ 의 값은 트레이닝 과정에서 매우 작아지므로 무시하고, 주로 mean  $\mu$ 만을 구합니다.



# Variational AutoEncoders - Preview

- Gaussian Model

Maximum likelihood를 이용하여 주어진 dataset  $\mathbf{X} = \{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_3, \dots, \mathbf{x}_N\}$ 에 대해 Gaussian distribution의 parameter를 최적화하는 방법에 대해서 알아봅니다.

- Probabilistic PCA

Prior  $p_\theta(\mathbf{z})$ 에서  $\mathbf{z}$  vector를 샘플링하고 decoder로 구현되는 conditional distribution  $p_\theta(\mathbf{x} | \mathbf{z})$ 을 통해 data를 생성한다는 점에서 VAE와 목적이 같은 모델입니다. 하지만 conditional distribution  $p_\theta(\mathbf{x} | \mathbf{z})$ 이 linear-Gaussian으로 주어지므로써 marginal likelihood와 true posterior를 closed form으로 구할 수 있다는 특징이 있습니다.

- AutoEncoder

Probabilistic PCA나 VAE와 다르게 deterministic 모델이며,  $\mathbf{z}$  space에서  $\mathbf{z}$  vector들의 분포를 한정시키지 않습니다. 그러므로  $\mathbf{z}$  space 상에서 샘플링이 불가능하면 generative model이 될 수 없습니다.

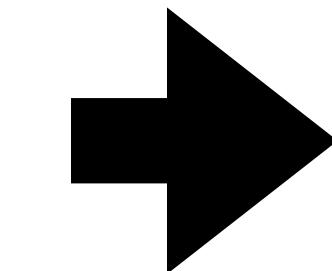
- Variational AutoEncoder

Probabilistic PCA가 prior  $p_\theta(\mathbf{z})$ 를 가지고 있다는 장점과, auto encoder가 neural network라는 장점을 혼합하여 표현력 높은 generative model을 만듭니다.



# Variational AutoEncoders

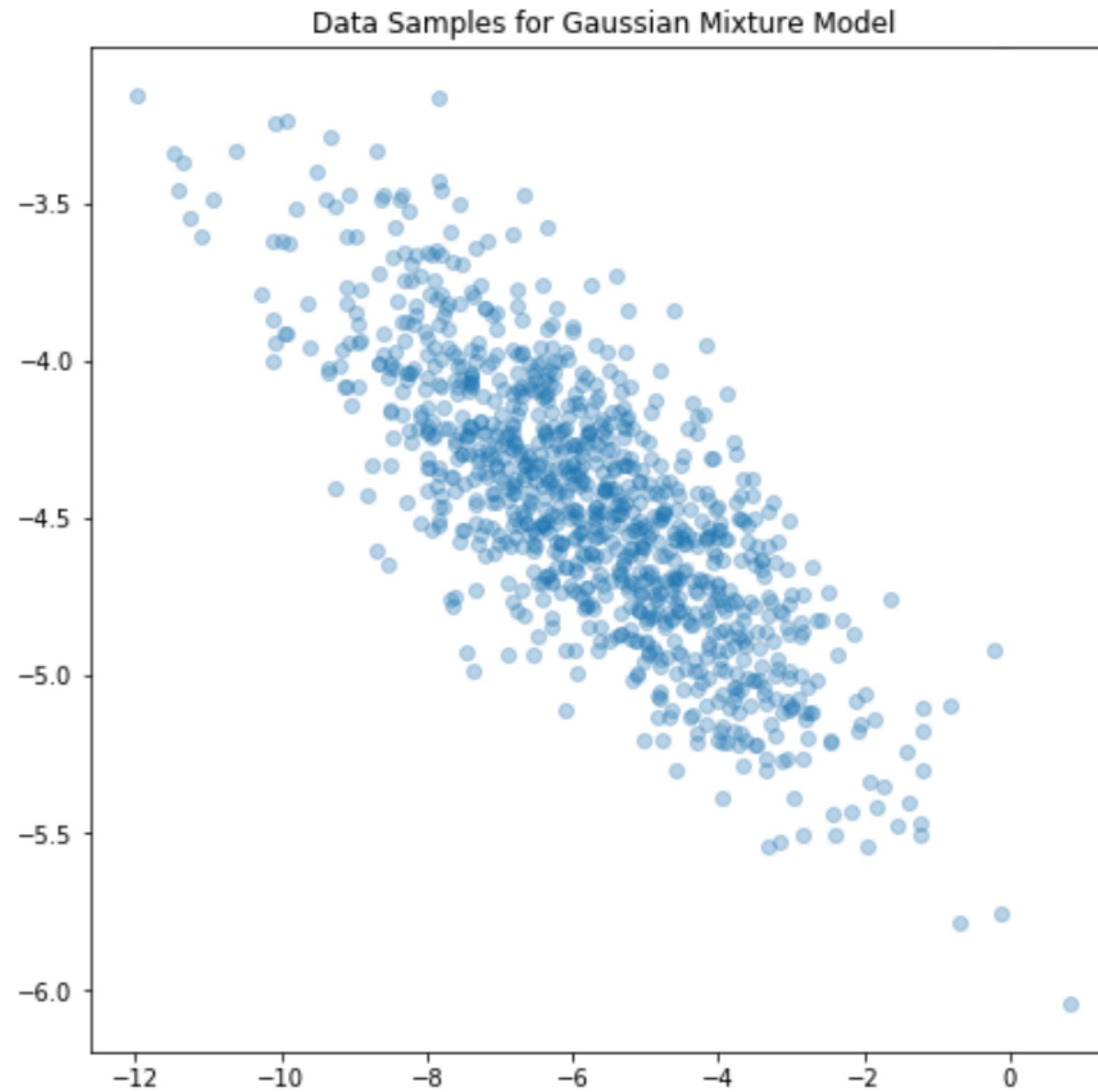
	Probabilistic PCA	Auto-Encoder
장점	Prior에서 샘플링할 수 있어 generative model임	Encoder와 decoder가 non-linear한 neural network로 data를 압축하고 복원하는데 좋은 성능을 지님
단점	Decoder를 linear transform에 한정시켜서 data를 복원하기에 능력이 부족	Z space에서 z vector들의 분포를 설정할 수 없어 sampling이 불가능하다. (별도로 z vector들의 분포를 GMM등으로 모델링하면 가능)



**Variational  
Auto-  
Encoder**

# Gaussian Model

# Gaussian Model



# Gaussian Model

- Single multivariate Gaussian distribution을 이용해 dataset의 distribution을 모델링하고자 합니다.

- 

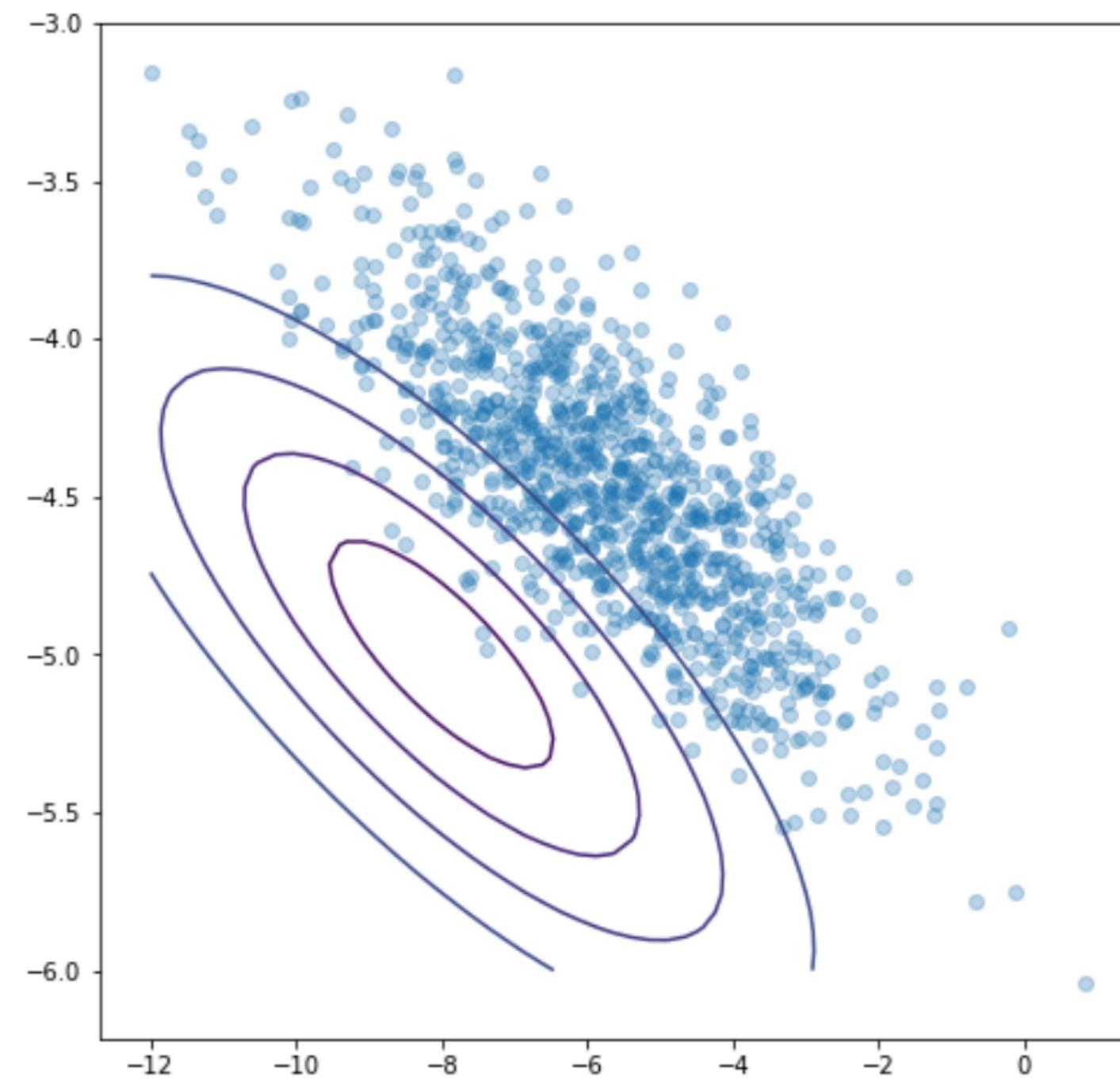
$$\begin{aligned} p_{\theta}(x) &= N(x | \mu, \Sigma) \\ &= \frac{1}{\sqrt{(2\pi)^D |\Sigma|}} \exp \left[ -\frac{1}{2}(x - \mu)\Sigma^{-1}(x - \mu)^T \right] \end{aligned}$$

- 이를 위해서 maximum likelihood 방식을 이용합니다. 즉, dataset의 likelihood를 최대화하는 parameters  $\theta = \{\mu, \Sigma\}$ 를 구합니다.

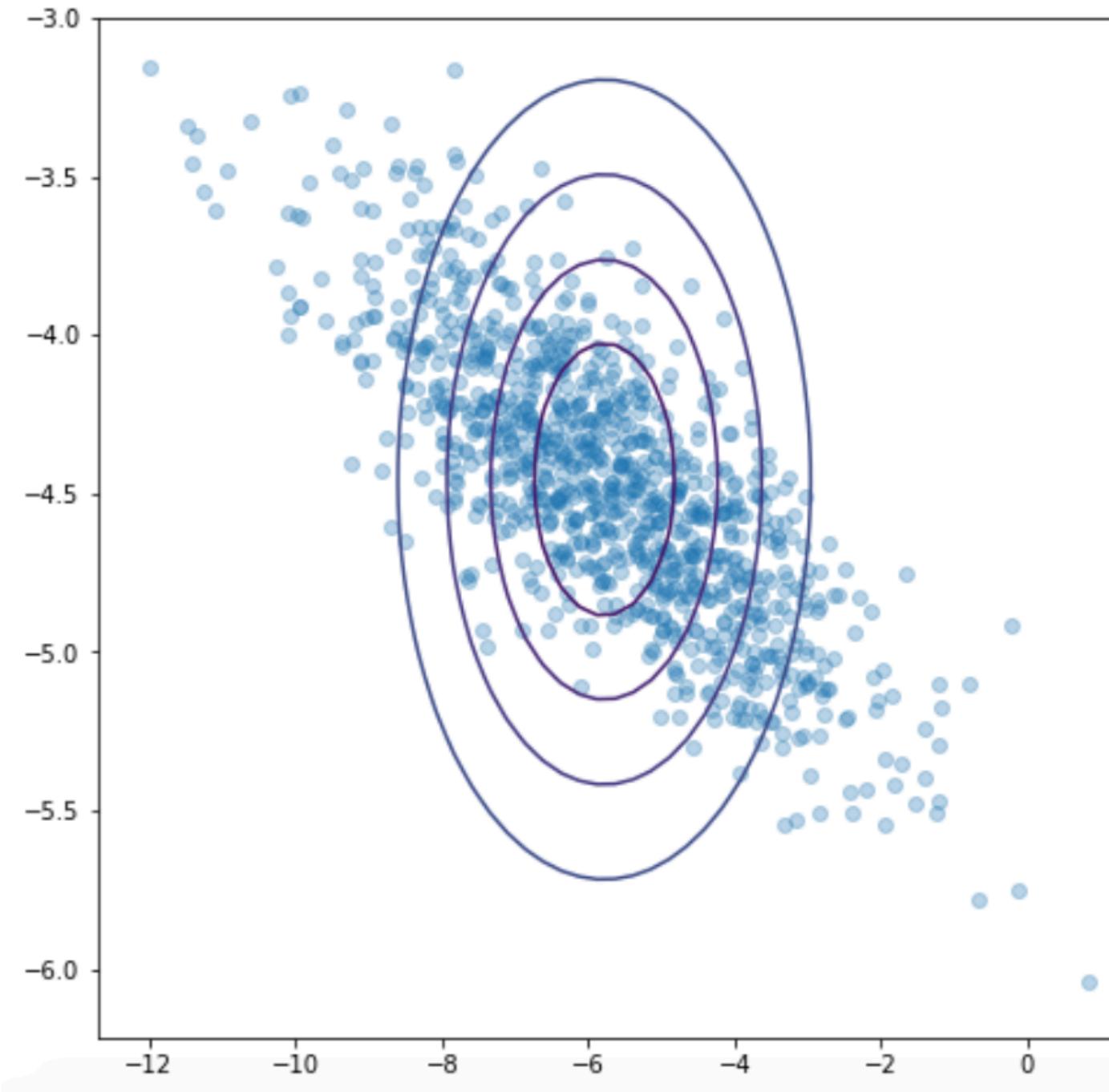
- 

$$\arg \max_{\theta} p_{\theta}(X), \text{ where } X = \{x_1, x_2, \dots, x_N\}$$

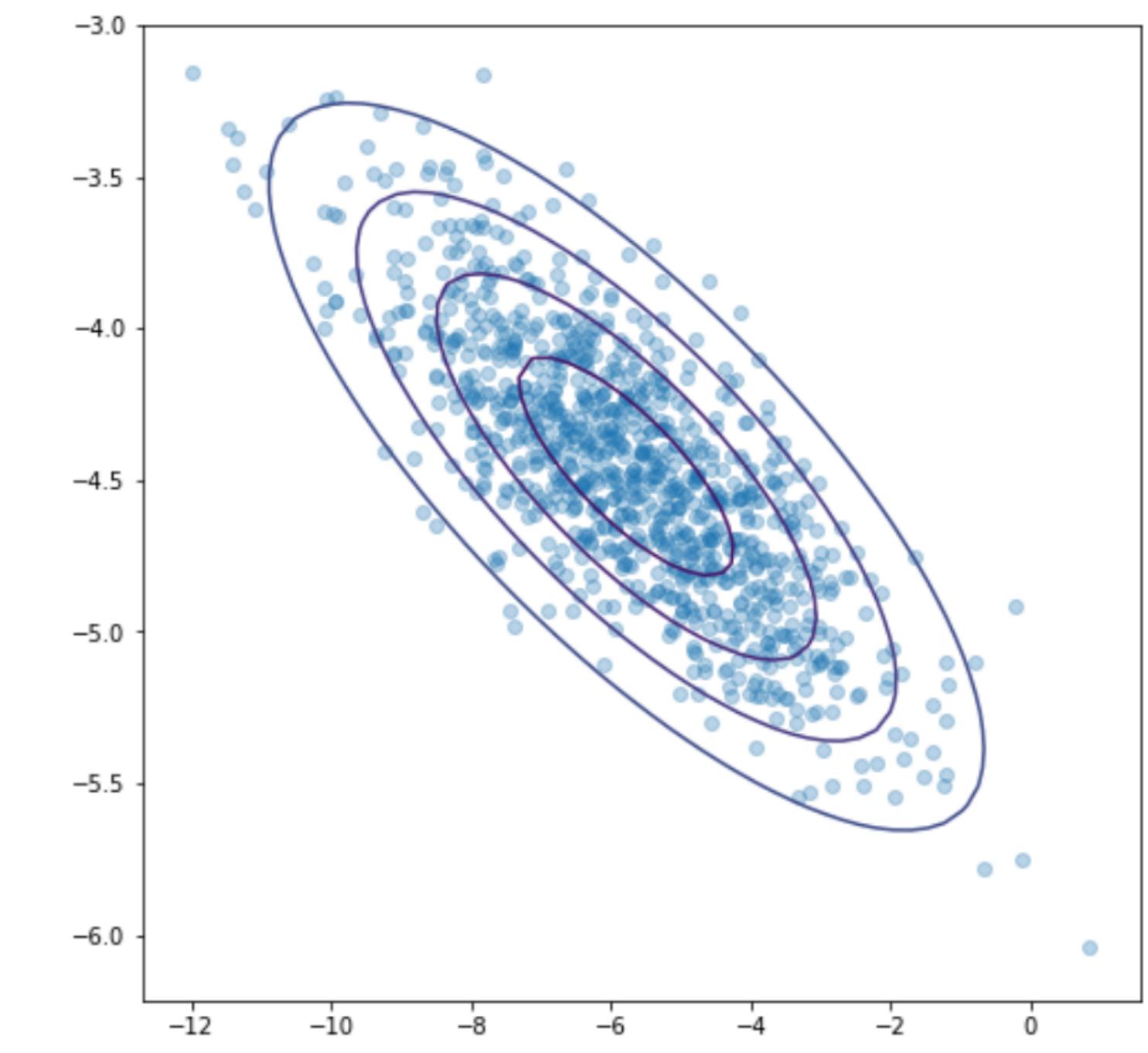
# Gaussian Model



낮은 likelihood를 갖는  $\mu, \Sigma$ 가 맞춰진 상황



likelihood가 최대가 되도록  $\mu$ 가 맞춰졌으나  $\Sigma$ 가 적합하지 않은 상황



likelihood가 최대가 되도록  $\mu$ 와  $\Sigma$ 가 맞춰진 상황

# Gaussian Model

- Distribution  $p_\theta(x)$ 에서 data samples  $X$  전체의 likelihood는 다음과 같습니다. 이 때, 각 samples들은 I.I.D. (independent and identically distributed)로 가정합니다.

•

$$p(X) = \prod_{n=1}^N p(x_n), X = \{x_1, x_2, \dots, x_N\}$$

- 위 식에 log를 씌워 Dataset 전체의 log-likelihood를 구하면 다음과 같습니다.

•

$$\log p_\theta(X) = \log \prod_{n=1}^N p_\theta(x_n) = \sum_{n=1}^N \log p_\theta(x_n)$$

- Single multivariate Gaussian distribution에서 data point 하나의 log-likelihood는 다음과 같습니다.

•

$$\log p_\theta(x_n) = -\frac{D}{2} \log 2\pi - \log |\Sigma| - \frac{1}{2}(x_n - \mu)\Sigma^{-1}(x_n - \mu)^T$$

- 따라서 dataset 전체의 log-likelihood는 다음과 같이 계산됩니다.

•

$$\log p_\theta(X) = \sum_{n=1}^N -\frac{D}{2} \log 2\pi - \log |\Sigma| - \frac{1}{2}(x_n - \mu)\Sigma^{-1}(x_n - \mu)^T$$

# Gaussian Model

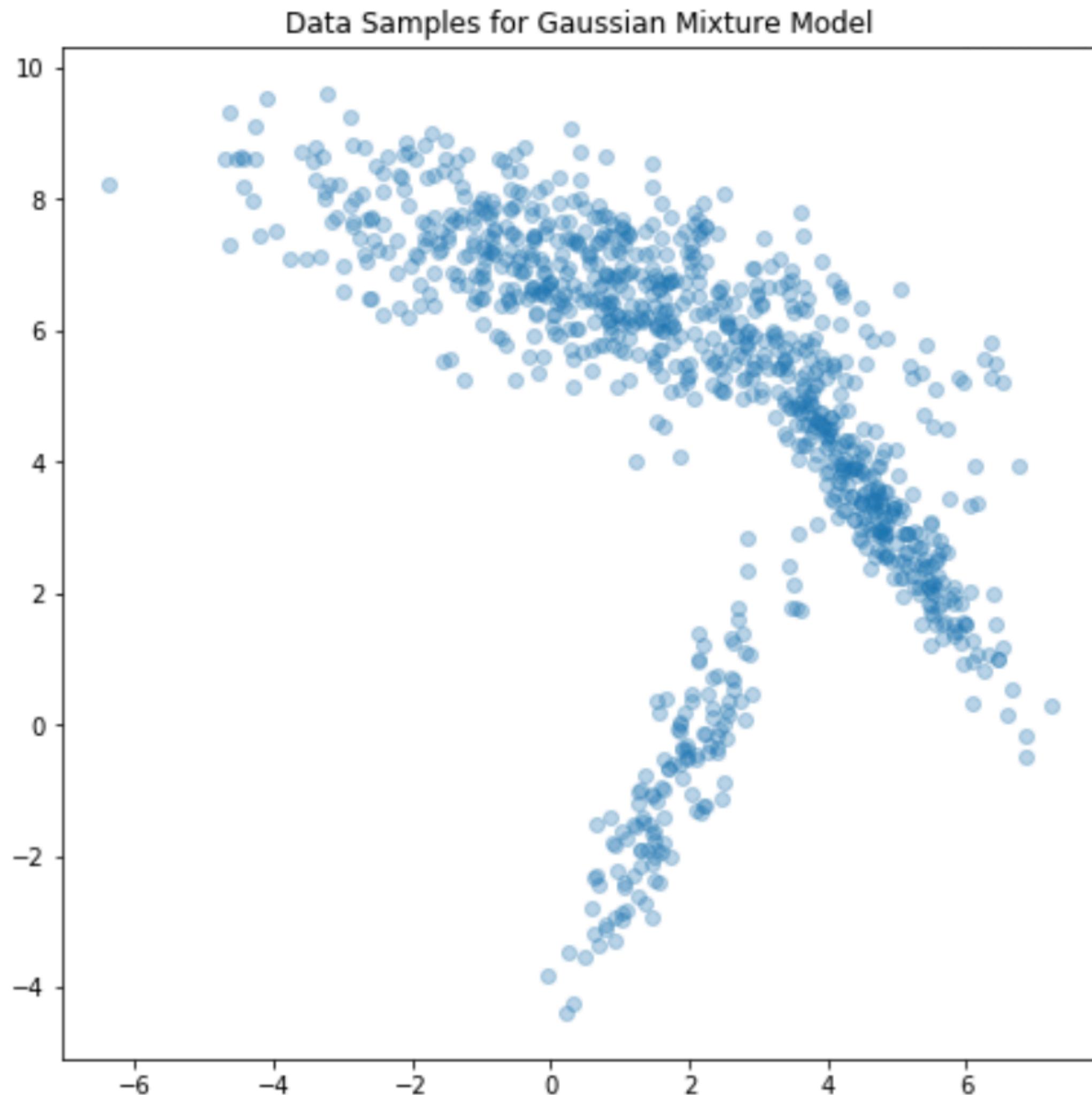
- $\log p_\theta(X)$ 가 최대가 되도록하는 parameters  $\mu, \Sigma$ 를 다음과 같이 closed form solution을 구할 수 있습니다.
- 

$$\mu_{ML} = \frac{1}{N} \sum_{n=1}^N x_n$$

$$\Sigma_{ML} = \frac{1}{N} \sum_{n=1}^N (x_n - \mu)(x_n - \mu)^T$$

# Gaussian Mixture Model

# Gaussian Mixture Models



# Gaussian Mixture Models

- 여러 Multivariate Gaussian distributions를 이용해 dataset의 distribution을 모델링하고자 합니다.

- 

$$p_{\theta}(x) = \sum_{k=1}^K \pi_k N(x | \mu_k, \Sigma_k)$$

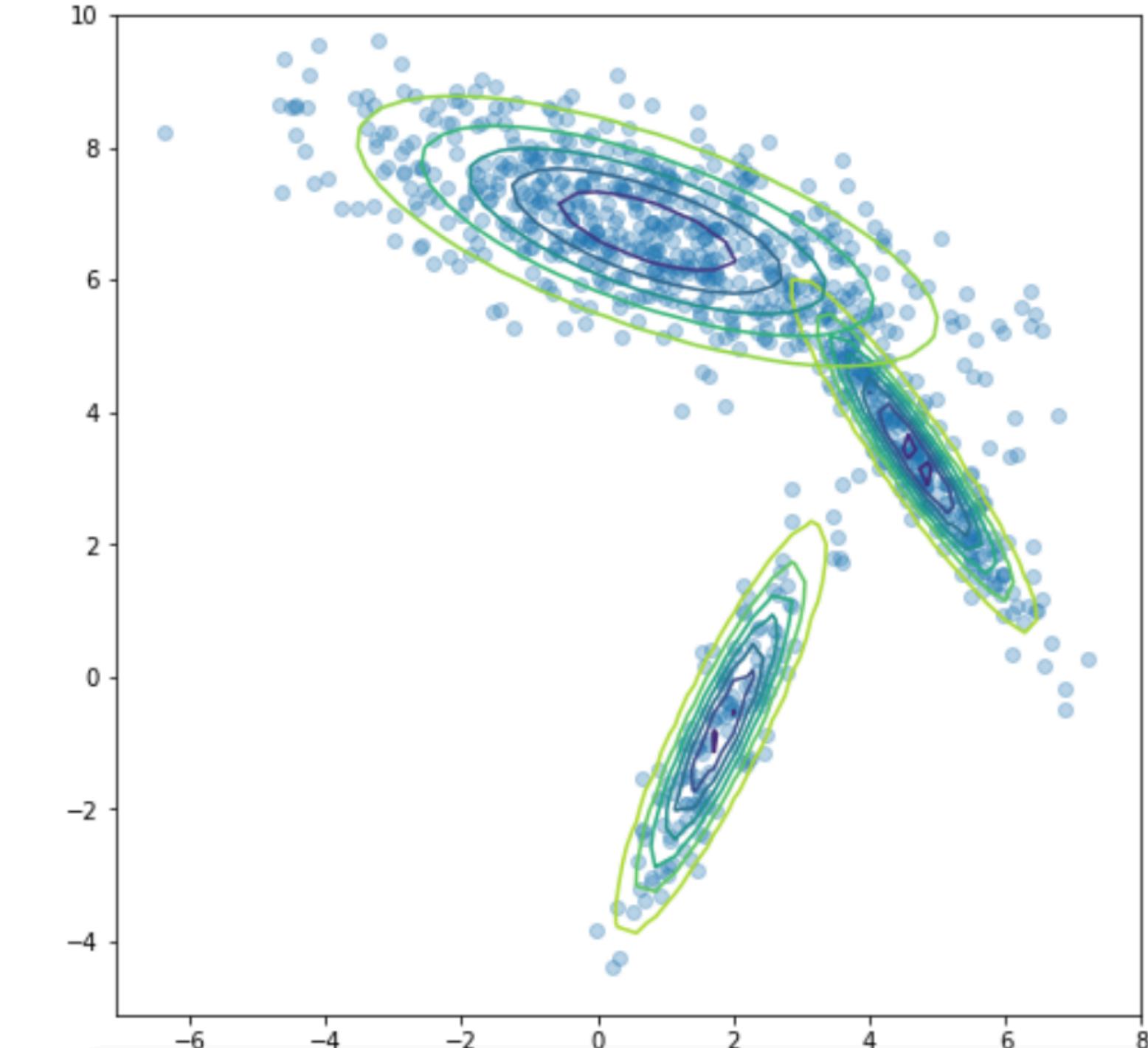
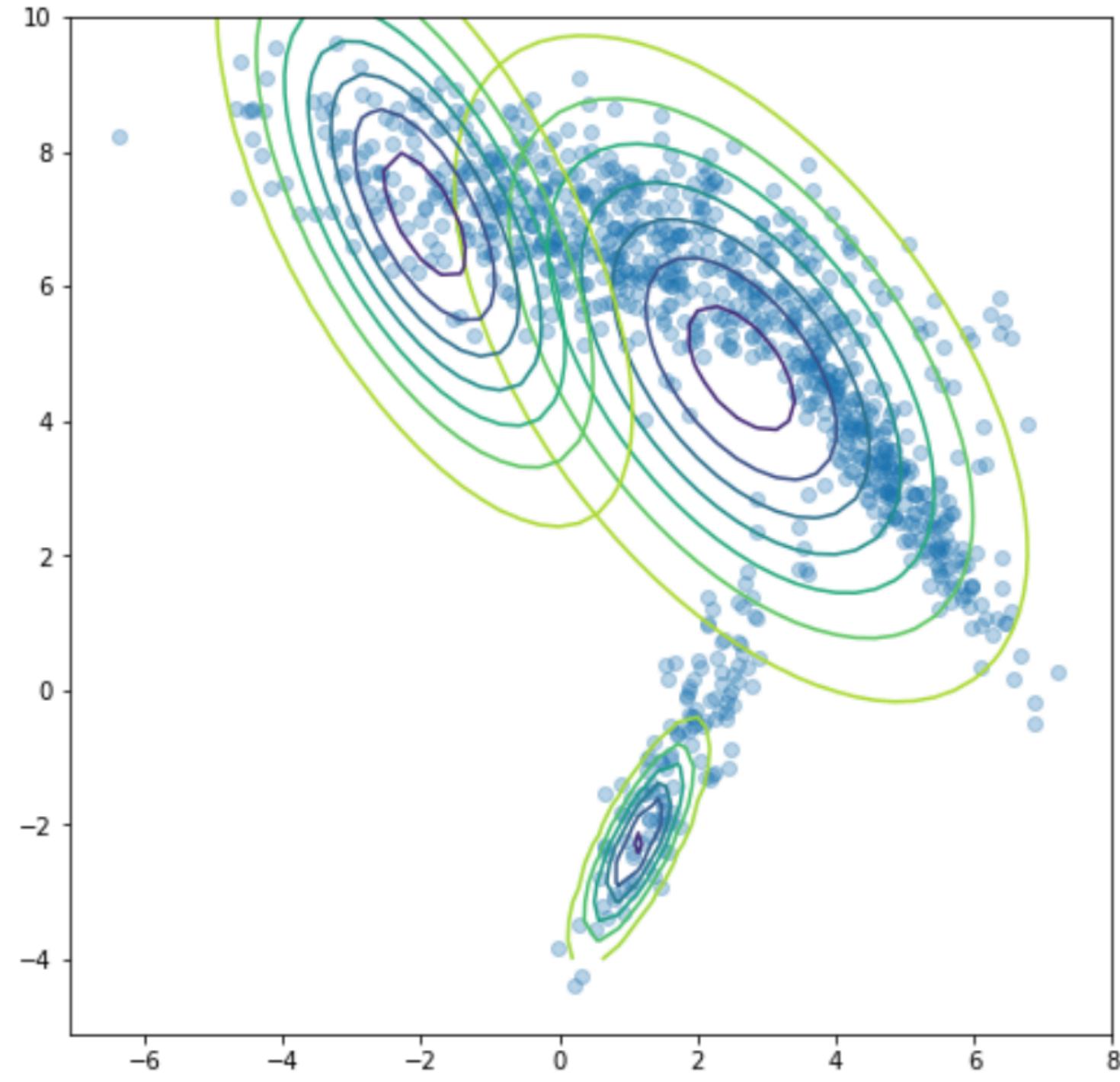
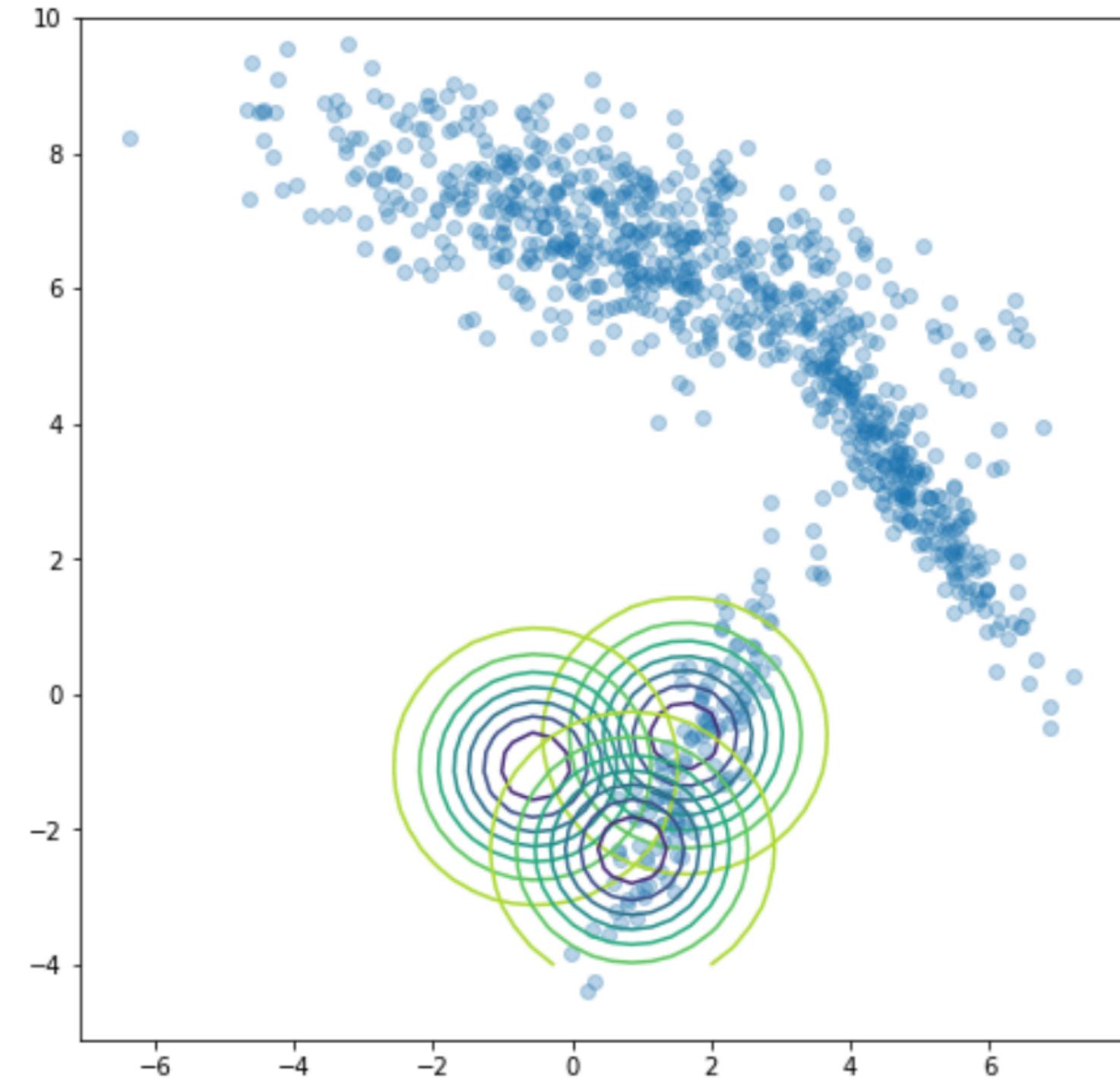
- 위 식에서  $K$ 는 Gaussian distribution의 mixture 갯수를 뜻합니다. 각 Gaussian component 별로 parameters인 mixing coefficient  $\pi_k$ 와 mean  $\mu_k$ , covariance matrix  $\Sigma_k$ 가 존재합니다.
- Maximum likelihood 방식을 이용하여 dataset에 맞는 distribution의 parameters를 찾습니다. 즉, dataset의 likelihood를 최대화하는 parameters  $\theta = \{\pi_k, \mu_k, \Sigma_k\}$ 를 구합니다.

- 

$$\arg \max_{\theta} p_{\theta}(X), \text{ where } X = \{x_1, x_2, \dots, x_N\}$$

# Gaussian Mixture Models

- dataset의 likelihood가 커지도록 parameters가 업데이트 되는 모습



# Gaussian Mixture Models

- Data point 하나는 다음과 같은 두 순서 의해서 생성되었다고 볼 수 있습니다.
  1.  $K$ 개의 components(Gaussian distributions) 중 하나를 선택
  2. 선택한 component에서 data point 샘플링
- Mixture의 각 component들이 선택될 확률을 categorical distribution으로 나타냅니다.
- $$p_{\theta}(z) = \prod_{k=1}^K \pi_k^{z_k}$$
- 이 때,  $z$ 는 선택된 component를 나타내는 one-hot vector로 표현됩니다. 예를 들어 총 5개의 components가 있고 3번째 component가 선택되었다면  $z = (0,0,1,0,0)^T$ 를 갖게 됩니다.  $z_k$ 는 one-hot vector의  $k$ 번째 element를 뜻합니다.
- Component가 정해졌을 때, 즉  $z$ 가 주어졌을 때 data point의 likelihood는 Gaussian distribution으로 나타냅니다.

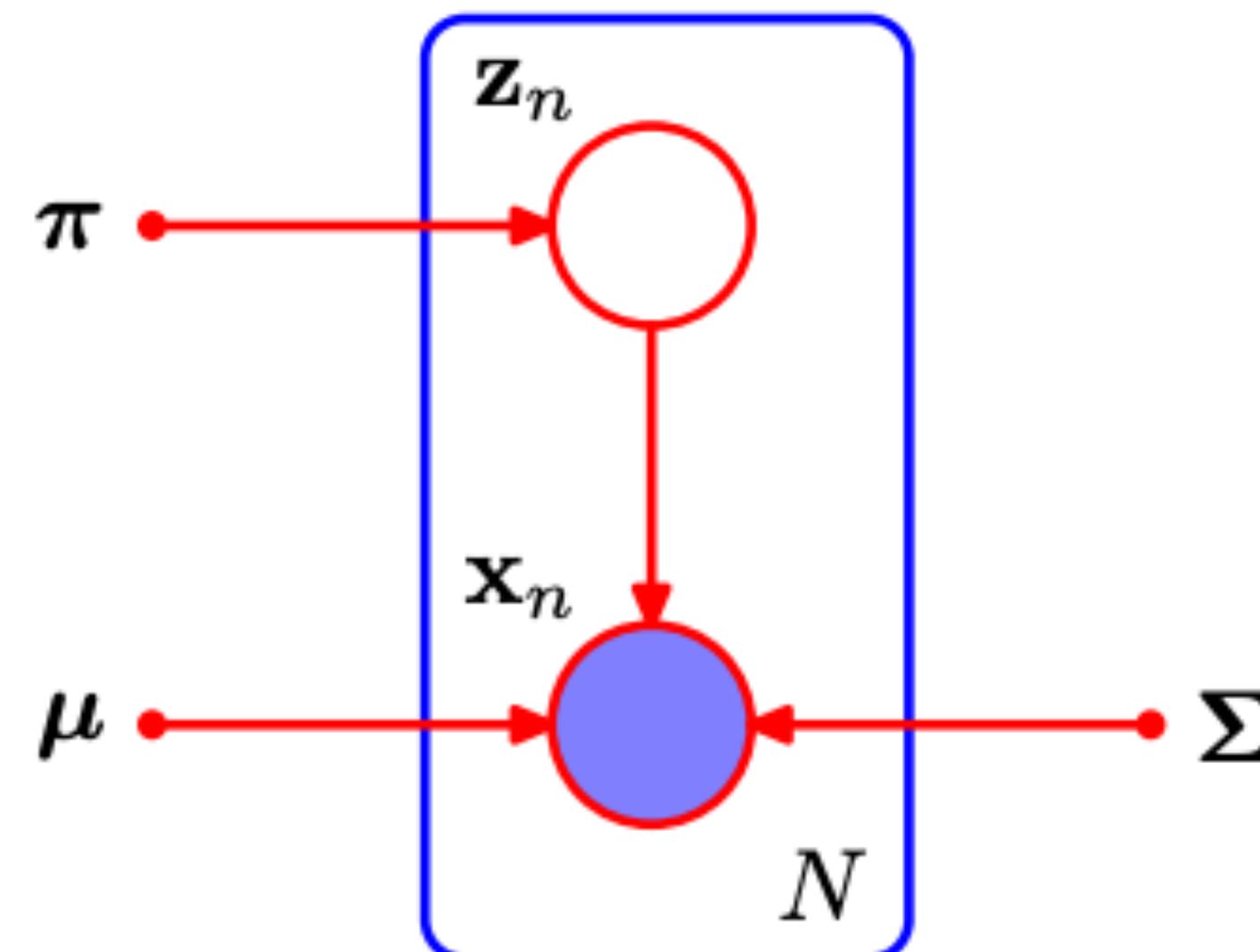
$$p_{\theta}(x | z) = \prod_{k=1}^K N(x | \mu_k, \Sigma_k)^{z_k}$$

# Gaussian Mixture Models

- $z$ 를 특정하지 않은 marginal likelihood는 다음과 같이 구할 수 있습니다.

$$p_{\theta}(x) = \sum_z p_{\theta}(z)p_{\theta}(x|z) = \sum_{k=1}^K \pi_k N(x|\mu_k, \Sigma_k)$$

- Gaussian mixture models을 graphical representation으로 나타낼 수 있습니다.



# Gaussian Mixture Models

- Single Gaussian Model의 경우와 마찬가지로 I.I.D. dataset 전체의 likelihood는 다음과 같습니다.

- 

$$p_{\theta}(X) = \prod_{n=1}^N p_{\theta}(x_n), X = \{x_1, x_2, \dots, x_N\}$$

- Gaussian mixture models 경우 log-likelihood는 다음과 같이 전개됩니다.

- 

$$\log p_{\theta}(X) = \log \prod_{n=1}^N p_{\theta}(x_n) = \sum_{n=1}^N \log p_{\theta}(x_n) = \sum_{n=1}^N \log \sum_{k=1}^K \pi_k N(x_n | \mu_k, \Sigma_k)$$

- Single Gaussian model과 다르게 log-likelihood를 최대화하는 parameters를 closed form으로 구할 수 없습니다.

# Gaussian Mixture Models

- log-likelihood가 최대가 되는 mean  $\mu_k$ 을 구해보기 위해  $\mu_k$ 로 미분하고 이를 0으로 두면 다음과 같습니다.

•

$$0 = \sum_{n=1}^N \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n | \mu_j, \Sigma_j)} \Sigma^{-1}(x_n - \mu_k)$$

- 만약 위 식에서  $\frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n | \mu_j, \Sigma_j)}$  부분을 어떤 값으로 고정한다면  $\mu_k$ 를 구할 수 있습니다.

$$0 = \sum_{n=1}^N \gamma(z_{nk}) \Sigma^{-1}(x_n - \mu_k)$$

$\Sigma$ 를 양변에 곱하면

$$0 = \sum_{n=1}^N \gamma(z_{nk}) x_n - \sum_{n=1}^N \gamma(z_{nk}) \mu_k$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n, \text{ where } N_k = \sum_{n=1}^N \gamma(z_{nk})$$

# Gaussian Mixture Models

- 앞에서 고정한 값은 사실 어떤 data point  $x_n$ 이 주어졌을 때  $k$ 번째 component로부터 생성되었을 확률을 나타내는 posterior로 해석할 수 있습니다. GMM에서 이를 responsibility라고 말하기도 합니다.

$$0 = \sum_{n=1}^N \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n | \mu_j, \Sigma_j)} \Sigma^{-1}(x_n - \mu_k)$$

↓

Prior  $p(z_k = 1)$       Likelihood  $p(x_n | z_k = 1)$

Marginal Likelihood(Evidence)

$$\sum_{z_j} p(z_j = 1) p(x_n | z_j = 1) = p(x_n)$$

$$\text{Prior} \times \text{Likelihood} / \text{Evidence} = \text{Posterior}$$
$$p(z_{nk} = 1 | x_n)$$

# Gaussian Mixture Models

- posterior를 고정하면 mean  $\mu_k$ 을 비롯해 mixing coefficient  $\pi_k$ 와 covariance matrix  $\Sigma_k$ 도 closed form으로 구할 수 있습니다.

$$\bullet \quad \pi_k = \frac{N_k}{N},$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n,$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T$$

where  $\gamma(z_{nk}) = \sum_{n=1}^N \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n | \mu_j, \Sigma_j)}$  and  $N_k = \sum_{n=1}^N \gamma(z_{nk})$

# Gaussian Mixture Models

- Parameters  $\theta = \{\pi_k, \mu_k, \Sigma_k\}$ 를 구하고 나면 그에 따라 posterior인 responsibility  $\gamma(z_{nk})$ 값도 변하게 됩니다. 따라서 EM 알고리즘이라 불리는 iteration을 통해 최적화를 진행하게 됩니다.
- Expectation-Maximization Algorithm for GMM

1. parameters  $\theta = \{\pi_k, \mu_k, \Sigma_k\}$ 를 임의의 값으로 초기화 한다.

2. E-step : responsibility를 구한다.

$$\gamma(z_{nk}) = \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n | \mu_j, \Sigma_j)}$$

3. M-step : responsibility를 고정하고 parameters를 다시 구한다.

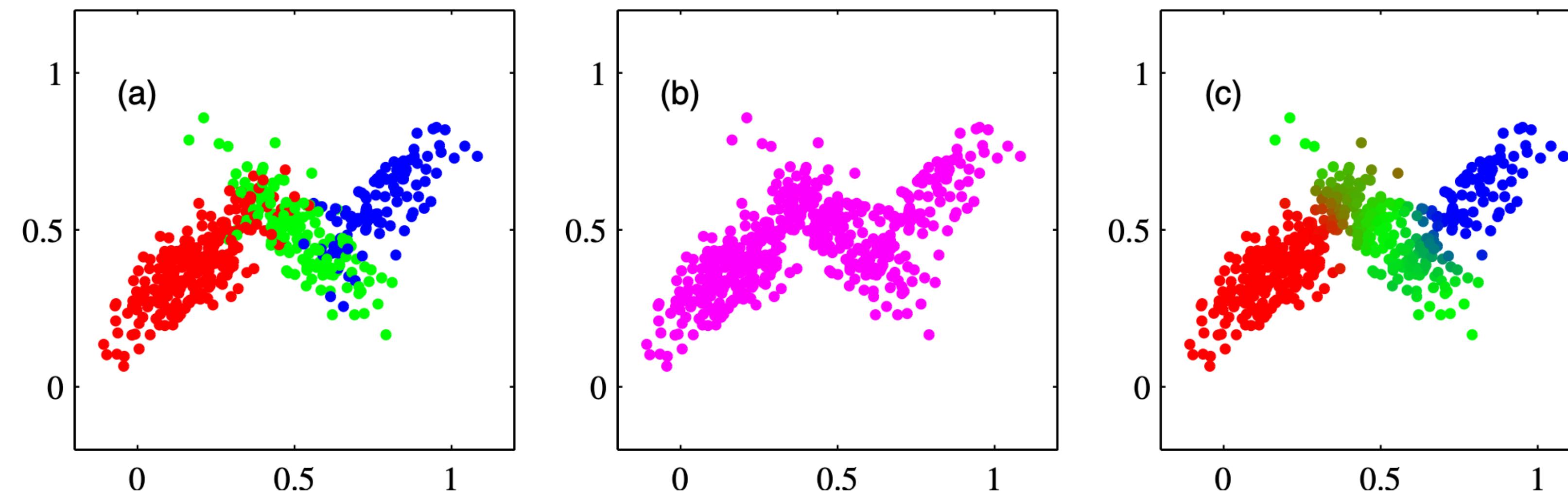
$$\pi_k^{new} = \frac{N_k}{N}, \mu_k^{new} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n, \Sigma_k^{new} = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) (x_n - \mu_k^{new})(x_n - \mu_k^{new})^T$$

$$where N_k = \sum_{n=1}^N \gamma(z_{nk})$$

4. 정해진 step을 돌거나, 정해놓은 수렴 기준에 도달할 때 까지 E-step과 M-step을 반복한다.

# Gaussian Mixture Models

- 현재 가지고 있는 dataset은  $X = \{x_1, x_2, \dots, x_N\}$ 입니다. 이를 incomplete dataset이라고 합니다.
- 모든  $X$ 에 대응하는 모든 latent variables  $Z = \{z_1, z_2, \dots, z_N\}$ 도 가지고 있다면 이를 complete dataset이라고 할 수 있습니다.



**Figure 9.5** Example of 500 points drawn from the mixture of 3 Gaussians shown in Figure 2.23. (a) Samples from the joint distribution  $p(z)p(x|z)$  in which the three states of  $z$ , corresponding to the three components of the mixture, are depicted in red, green, and blue, and (b) the corresponding samples from the marginal distribution  $p(x)$ , which is obtained by simply ignoring the values of  $z$  and just plotting the  $x$  values. The data set in (a) is said to be *complete*, whereas that in (b) is *incomplete*. (c) The same samples in which the colours represent the value of the responsibilities  $\gamma(z_{nk})$  associated with data point  $x_n$ , obtained by plotting the corresponding point using proportions of red, blue, and green ink given by  $\gamma(z_{nk})$  for  $k = 1, 2, 3$ , respectively

# Gaussian Mixture Models

- Dataset으로  $X = \{x_1, x_2, \dots, x_N\}$ 와  $Z = \{z_1, z_2, \dots, z_N\}$ 가 주어진 경우 complete data log-likelihood를 다음과 같이 쓸 수 있습니다.

$$\log p_\theta(X, Z) = \log p_\theta(Z) + \log p_\theta(X | Z) = \sum_{n=1}^N [\log p_\theta(z_n) + \log p_\theta(x_n | z_n)]$$

- $p_\theta(z)$ 는 categorical distribution,  $p_\theta(x | z)$ 는 Gaussian distribution이므로 complete data-log-likelihood를 최대화하는 것은 앞에서 보인 것처럼 쉽게 전개할 수 있습니다.

- $\log p_\theta(Z)$ 를 최대화하는 parameter  $\pi_k$ 를 구하면 다음과 같습니다.

$$\pi_k = \frac{N_k}{N}, \text{ where } N_k = \sum_{n=1}^N z_{nk}$$

- $Z$ 가 주어졌을 때  $\log p_\theta(X | Z)$ 를 최대화하는 parameters  $\mu_k, \Sigma_k$ 를 구하면 다음과 같습니다.

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N z_{nk} x_n,$$
$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N z_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$$

# Gaussian Mixture Models

Posterior(responsibility)가 주어졌을 때  
EM 알고리즘의 M-step

$$\pi_k = \frac{N_k}{N},$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk}) x_n,$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N \gamma(z_{nk})(x_n - \mu_k)(x_n - \mu_k)^T$$

$$\text{where } \gamma(z_{nk}) = \sum_{n=1}^N \frac{\pi_k N(x_n | \mu_k, \Sigma_k)}{\sum_{j=1}^K \pi_j N(x_n | \mu_j, \Sigma_j)}$$

$$\text{and } N_k = \sum_{n=1}^N \gamma(z_{nk})$$

Complete dataset  $X, Z$ 가 주어졌을 때  
complete data log-likelihood의 최대화

$$\pi_k = \frac{N_k}{N},$$

$$\mu_k = \frac{1}{N_k} \sum_{n=1}^N z_{nk} x_n,$$

$$\Sigma_k = \frac{1}{N_k} \sum_{n=1}^N z_{nk} (x_n - \mu_k)(x_n - \mu_k)^T$$

$$\text{where } N_k = \sum_{n=1}^N z_{nk}$$

# Gaussian Mixture Models

- 앞서 EM 알고리즘을 통해 구한 방법은 posterior  $p_{\theta}(Z|X)$ 로 가상의 데이터 z-variables를 만들고 complete data log-likelihood를 최대화한 것과 같다고 할 수 있습니다.
- 따라서 E-step과 M-step은 다음과 같이 일반할 수 있습니다.

E-step : dataset 전체의 posterior  $p_{\theta}(Z|X)$ 를 구한다.

M-step : posterior에 의한 complete data log-likelihood의 기댓값을 최대화하는 parameters를 구한다.

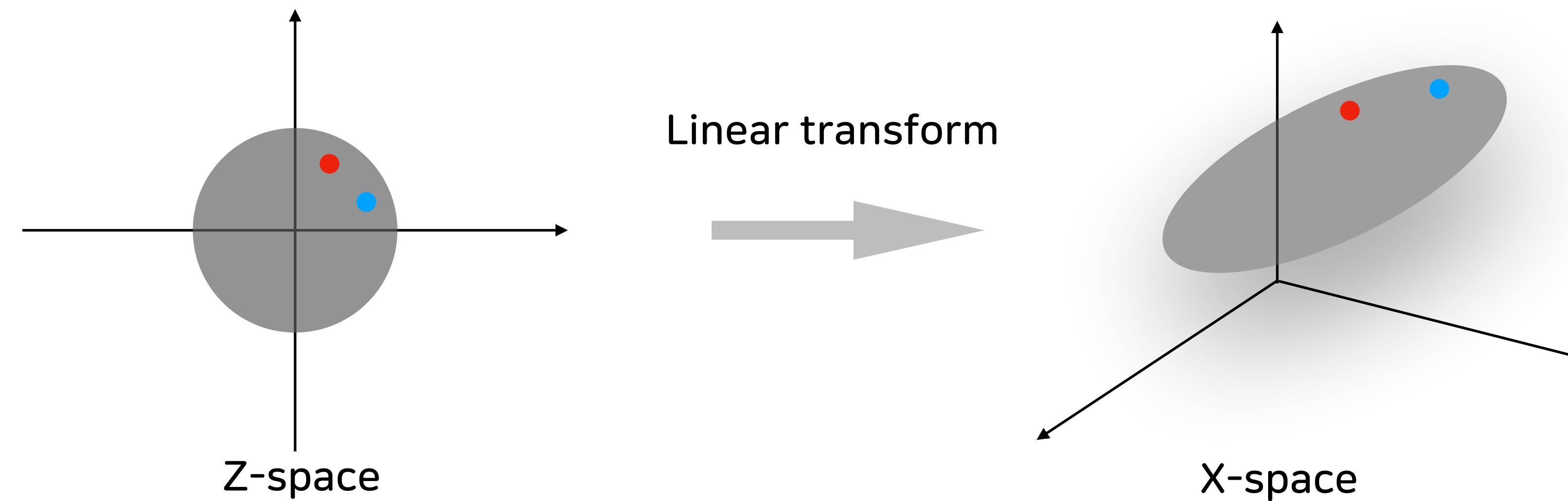
$$\theta^{new} = \arg \max_{\theta} \mathbb{E}_{p_{\theta^{old}}(Z|X)} [\log p_{\theta}(X, Z)]$$

# Probabilistic PCA

# Probabilistic PCA

- PPCA(Probabilistic PCA)는 dimension reduction에 사용되는 PCA의 (semi) Bayesian 버전입니다. parameters를 random variables로 두지 않았으므로 fully Bayesian이라고 볼 수 없습니다.
- GMM에서 discrete한 z-variable을 갖는데 반해, PPCA에서는 continuous한 z-variable을 갖습니다.
- GMM에서 data point 하나를 얻기 위해 다음과 같은 순서에 의해 샘플링 하였습니다.
  1.  $K$ 개의 components(Gaussian distributions) 중 하나를 선택
  2. 선택한 component에서 data point 샘플링
- PPCA에서는 다음과 같은 순서에 의해 샘플링이 이루어집니다.
  1. Z-space의 Gaussian distribution에서 샘플 하나를 선택
  2. 샘플을 linear transform을 통해 X-space로 맵핑
  3. 맵핑된 값을 mean으로 하는 또 다른 Gaussian distribution에서 data point 샘플링

# Probabilistic PCA



Prior  $p_\theta(z) = N(z | 0, I)$

Conditional  $p_\theta(x | z) = N(x | Wz + \mu, \sigma^2 I)$

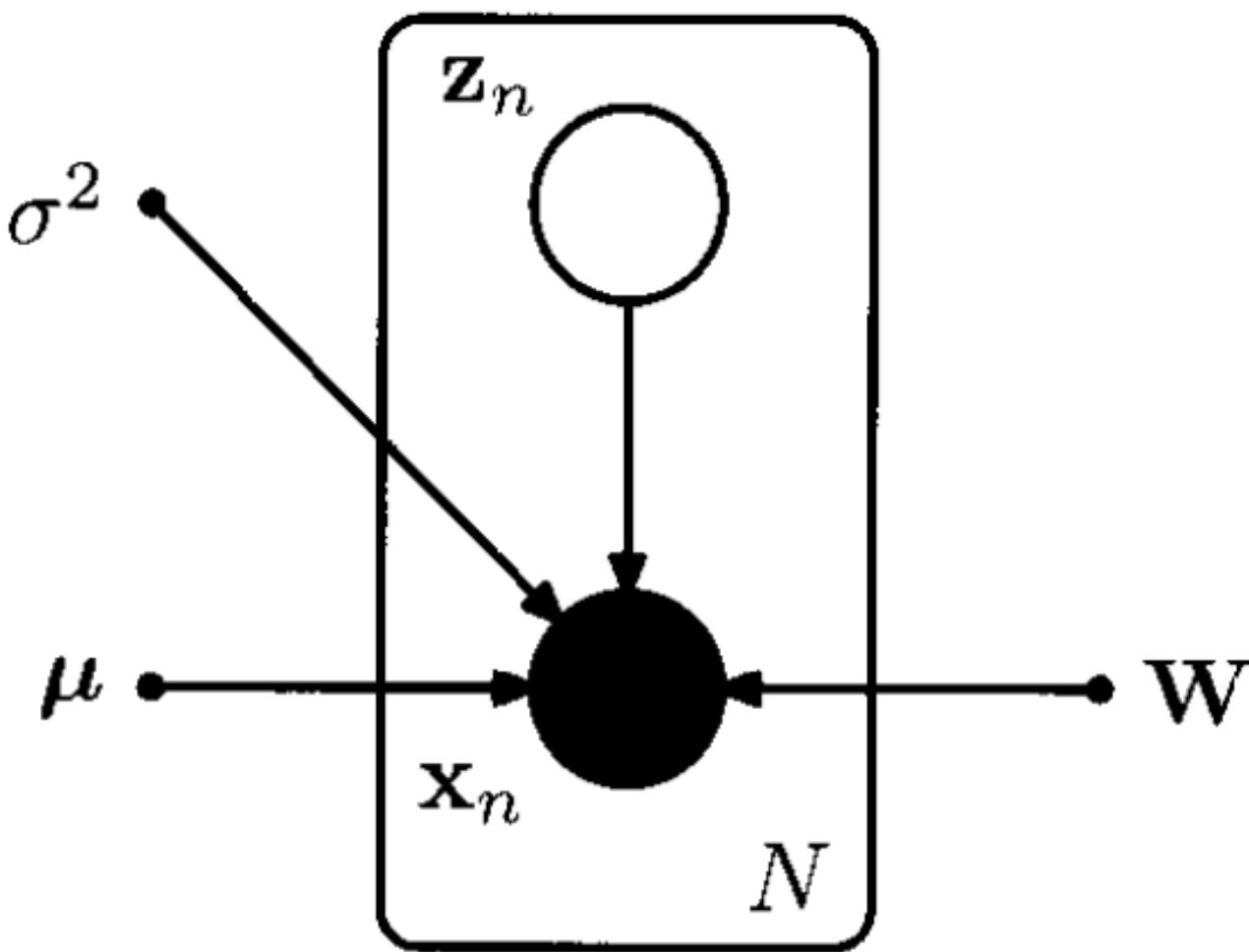
(Linear-Gaussian Model)

References :

Michael E. Tipping and Christopher M. Bishop. Probabilistic Principal Component Analysis. 1999.  
Christopher M. Bishop. Pattern Recognition and Machine Learning. p.570-580

# Probabilistic PCA

## Graphical Representation



Prior  $p_\theta(z) = N(z \mid 0, I)$

Conditional  $p_\theta(x \mid z) = N(x \mid Wz + \mu, \sigma^2 I)$

(Linear-Gaussian Model)

# Probabilistic PCA

- Maximum likelihood를 통해 parameters를 구하기 위해 marginal likelihood를 구합니다.

- 

$$\begin{aligned} p_{\theta}(x) &= \int p_{\theta}(x, z) dz = \int p_{\theta}(z)p_{\theta}(x|z) dz = \int N(z|0, I)N(x|Wz + \mu, \sigma^2 I) dz \\ &= N(x|\mu, C), \text{ where } C = WW^T + \sigma^2 I \end{aligned}$$

- 위 식을 통해 dataset  $X$ 의 marginal log-likelihood를 구합니다.

- 

$$\log p_{\theta}(X) = \sum_{n=1}^N \log p_{\theta}(x_n)$$

$$= -\frac{ND}{2} \log(2\pi) - \frac{N}{2} \log |C| - \frac{1}{2} \sum_{n=1}^N (x_n - \mu)^T C^{-1} (x_n - \mu)$$

$$p(x) = N(x|\mu, \Lambda^{-1})$$

$$p(y|x) = N(y|Ax + b, L^{-1})$$

$$p(y) = N(y|A\mu + b, L^{-1} + A\Lambda^{-1}A^T)$$

PRML p.93

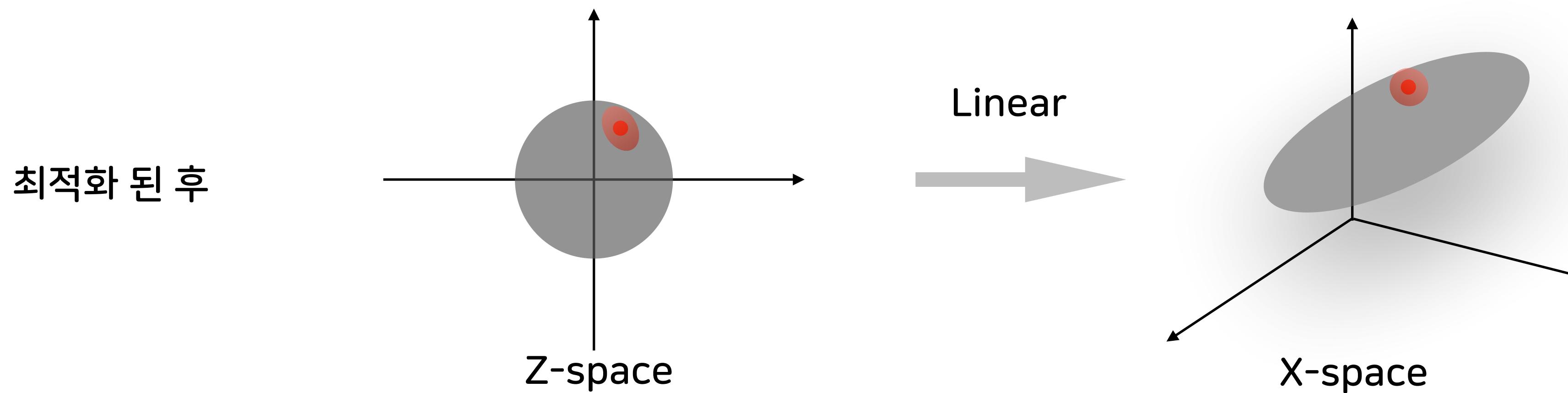
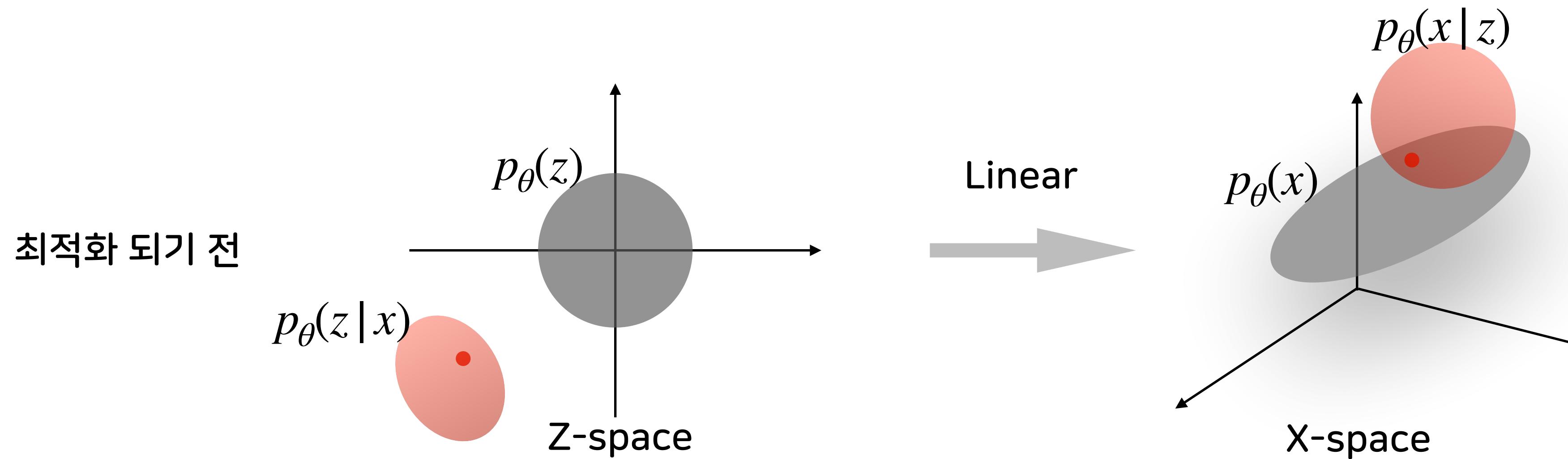
# Probabilistic PCA Maximum Likelihood

- GMM에서와 다르게 marginal likelihood가 Gaussian이 되고, 이를 최대화하는 parameters를 다음과 같이 closed form으로 구할 수 있습니다.
- $W = U_M(L_M - \sigma^2 I)^{1/2}R$ 
  - $U_M L_M U_M^T = S$  (*data covariance matrix*)
  - $U_M = [v_1 \ v_2 \ \cdots \ v_M]$  , where  $v_1, v_2, \dots, v_M$  = eigenvectors given by the data covariance matrix
  - $L_M = \begin{bmatrix} \lambda_1 & & & \\ & \lambda_2 & & \\ & & \ddots & \\ & & & \lambda_M \end{bmatrix}$ , where  $\lambda_1, \lambda_2, \dots, \lambda_M$  = the corresponding eigenvalues
  - $R_M$  = an arbitrary  $M \times M$  orthogonal matrix
- $\mu = \frac{1}{N} \sum_{n=1}^N x_n$
- $\sigma^2 = \frac{1}{D-M} \sum_{i=M+1}^D \lambda_i$

# Probabilistic PCA Expectation-Maximization

- 또는 GMM에서 했듯이 EM알고리즘으로 iteration을 통해 구할 수도 있습니다.
- PPCA에서 sampling은 두 번 이루어집니다.
  1. Prior  $p_\theta(z)$ 에서 sampling
  2. Linear transform후 얻어진 conditional distribution  $p_\theta(x | z)$ 에서 sampling
- 위 sampling을 통해 가지고 있는 dataset이 잘 나올 수 있도록 하기 위해서는 다음과 같은 두가지 조건을 만족해야 합니다.
  1. posterior  $p_\theta(z | x)$ 에서 뽑은 sample이 prior  $p_\theta(z)$ 에서 높은 likelihood를 가져야 한다.
  2. posterior  $p_\theta(z | x)$ 에서 뽑은 sample로 linear transform을 시킨 후 얻은 conditional distribution  $p_\theta(x | z)$ 에서 가지고 있는 dataset이 높은 likelihood를 가져야 한다.

# Probabilistic PCA Expectation-Maximization



# Probabilistic PCA Expectation-Maximization

- 위 두가지 조건을 만족하도록 하는 parameters를 구하기 위해 수식을 전개하면 다음과 같이 됩니다.

- 

$$\theta^{new} = \arg \max_{\theta} \mathbb{E}_{p_{\theta old}(Z|X)} [\log p_{\theta}(Z) + \log p_{\theta}(X|Z)]$$

1. 2.

- 1. posterior  $p_{\theta}(z|x)$ 에서 뽑은 sample이 prior  $p_{\theta}(z)$ 에서 높은 likelihood를 가져야 한다.  
2. posterior  $p_{\theta}(z|x)$ 에서 뽑은 sample로 linear transform을 시킨 후 얻은 conditional distribution  $p_{\theta}(x|z)$ 에서 가지고 있는 dataset이 높은 likelihood를 가져야 한다.
- 이는 곧 GMM에서 했던 EM 알고리즘의 M-step과 같은 식이 됩니다.

- 

$$\theta^{new} = \arg \max_{\theta} \mathbb{E}_{p_{\theta old}(Z|X)} [\log p_{\theta}(X, Z)]$$

# Probabilistic PCA Expectation-Maximization

- M-step에서는 posterior에 의해 가상으로 만들어진 complete data의 log-likelihood를 최대화 합니다. complete data log-likelihood는 다음과 같이 계산됩니다.

$$\log p_\theta(X, Z) = \log p_\theta(Z) + \log p_\theta(X | Z)$$

$$= \log \prod_{n=1}^N p_\theta(z_n) + \log \prod_{n=1}^N p_\theta(x_n | z_n)$$

$$= \sum_{n=1}^N \log p_\theta(z_n) + \sum_{n=1}^N \log p_\theta(x_n | z_n)$$

$$= \sum_{n=1}^N \log N(z_n | 0, I) + \sum_{n=1}^N \log N(x_n | Wz_n + \mu, \sigma^2 I)$$

- 또한 posterior  $p_\theta(Z | X)$ 에 대한 complete data log-likelihood  $\log p_\theta(X, Z)$ 의 기댓값은 다음과 같습니다.

•

$$\begin{aligned}\mathbb{E}_{p_\theta(Z|X)} [\ln p_\theta(\mathbf{X}, \mathbf{Z})] &= - \sum_{n=1}^N \left\{ \frac{D}{2} \ln (2\pi\sigma^2) + \frac{1}{2} \text{Tr} \left( \mathbb{E} [\mathbf{z}_n \mathbf{z}_n^T] \right) \right. \\ &\quad + \frac{1}{2\sigma^2} \| \mathbf{x}_n - \boldsymbol{\mu} \|^2 - \frac{1}{\sigma^2} \mathbb{E} [\mathbf{z}_n]^T \mathbf{W}^T (\mathbf{x}_n - \boldsymbol{\mu}) \\ &\quad \left. + \frac{1}{2\sigma^2} \text{Tr} \left( \mathbb{E} [\mathbf{z}_n \mathbf{z}_n^T] \mathbf{W}^T \mathbf{W} \right) \right\}\end{aligned}$$

# Probabilistic PCA Expectation-Maximization

- E-step : 각 data point마다 posterior  $p_\theta(z|x)$ 를 구합니다.

$$p_\theta(z|x) = N(z|M^{-1}W^T(x - \mu), \sigma^2 M^{-1})$$

where  $M = W^T W + \sigma^2 I$

- M-step : posterior를 이용해 parameters를 업데이트 합니다.  $\mu$ 는 sample mean이 자명하므로 closed form으로 구합니다.

$$W_{new} = \left[ \sum_{n=1}^N (x_n - \bar{x}) E[z_n]^T \right] \left[ \sum_{n=1}^N E[z_n z_n^T] \right]^{-1}$$

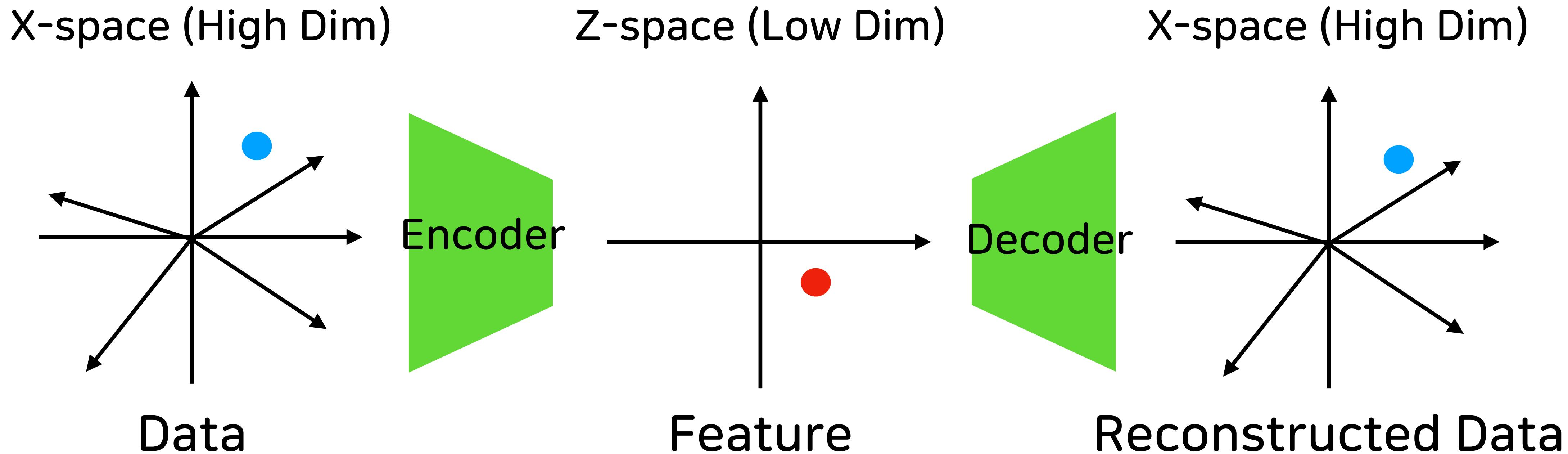
$$\sigma_{new}^2 = \frac{1}{ND} \sum_{n=1}^N \left\{ \|x_n - \bar{x}\|^2 - 2E[z_n]^T W_{new}^T (x_n - \bar{x}) + Tr(E[z_n z_n^T] W_{new}^T W_{new}) \right\}$$

where  $E[z_n z_n^T] = \sigma^2 M^{-1} + E[z_n] E[z_n]^T$

# Auto-Encoder

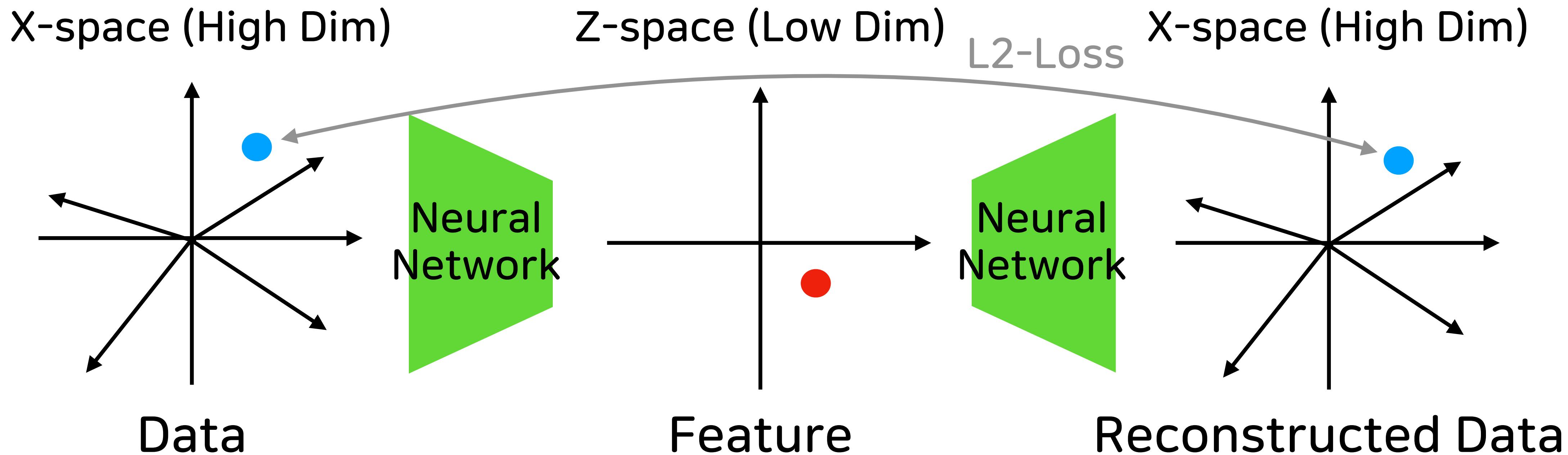
# Auto-Encoder(개념)

- Auto-Encoder는 Encoder를 통해 고차원의 데이터를 저차원으로 맵핑하고, Decoder를 통해 다시 복원하는 네트워크입니다.
- 이렇게 저차원으로 맵핑된 정보는 데이터를 나타내는 feature로 활용될 수 있습니다.



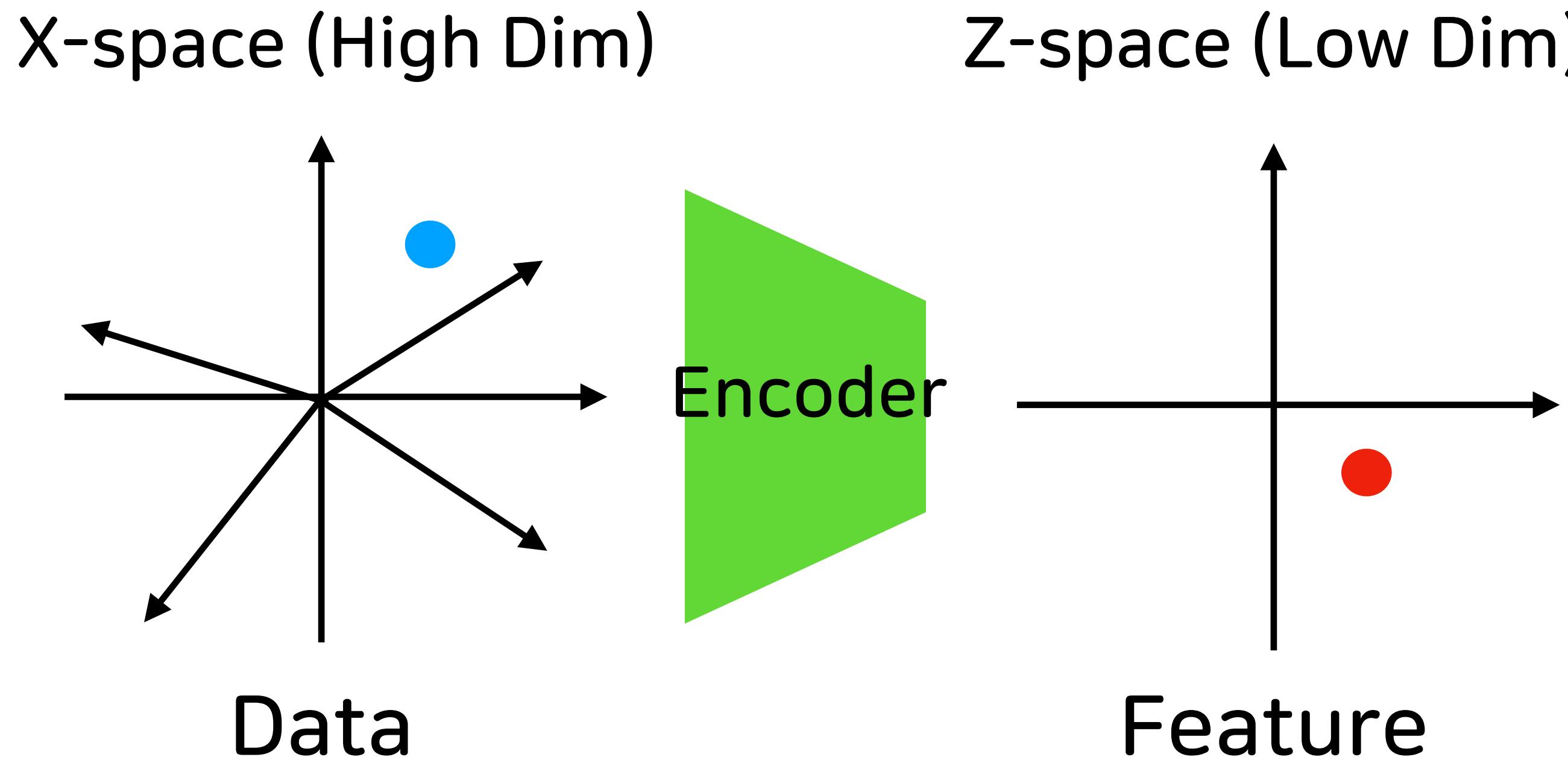
# Auto-Encoder(구성)

- Encoder와 Decoder는 Fully-Connected Layers 또는 Convolution Layers 등 Neural Networks로 구성합니다.
- input과 output 간에 L2 Loss를 취해 reconstruction을 할 수 있도록 합니다.



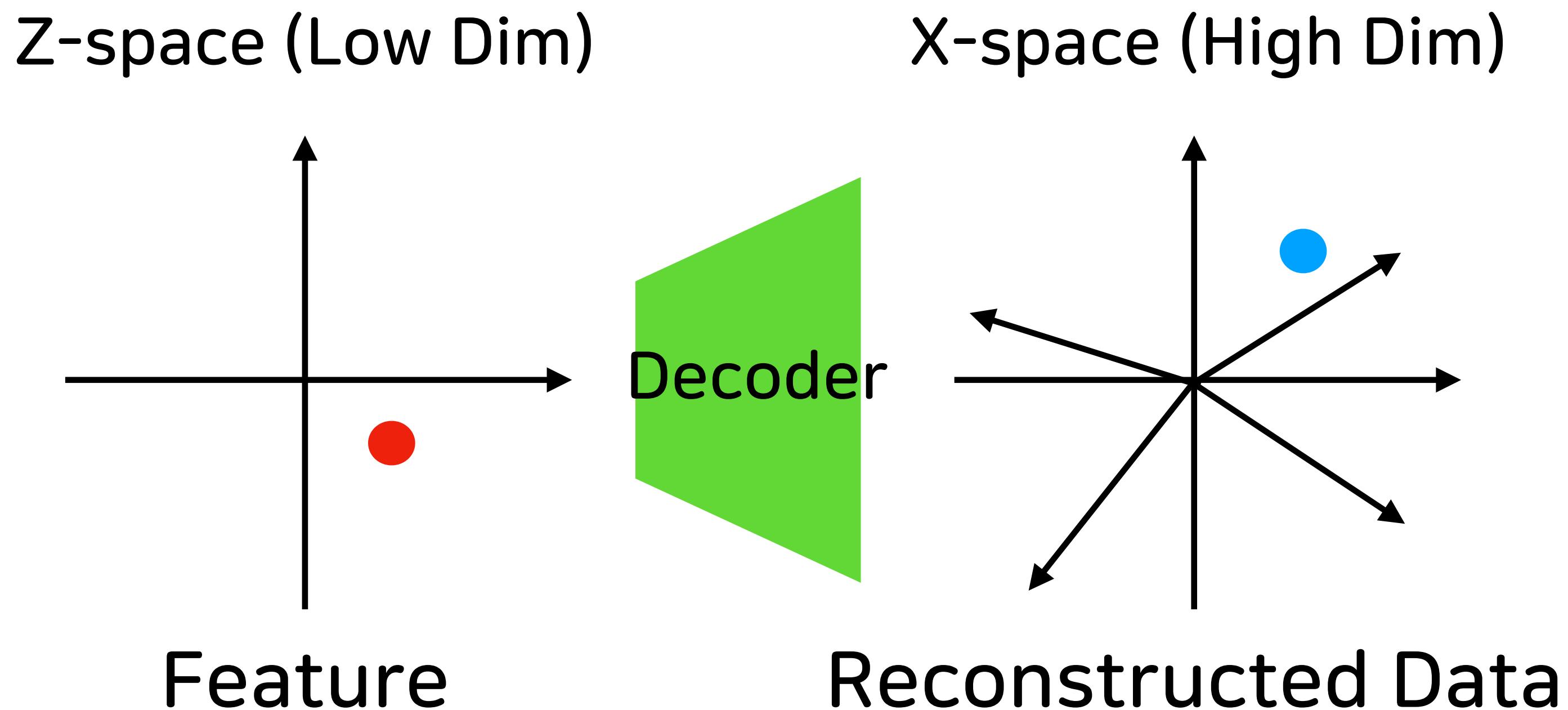
# Auto-Encoder(Encoder)

- 트레이닝 후 Encoder는 Feature Extraction의 용도로 활용될 수 있습니다.
- 이 때, Encoder는 고차원의 데이터를 압축하여 feature를 추출해내는 역할을 합니다.



# Auto-Encoder(Decoder)

- Z-space에서 벡터 하나를 샘플링하여 Decoder를 통해 X-space의 벡터로 복원하면 데이터를 생성해 낼 수 있습니다.
- 하지만 Z-space에서 벡터들의 집합이 어떤 분포를 가지고 있는지 Auto-Encoder 만으로는 알 수 없으므로 생성모델이라 부를 수 없습니다.



# Expectation & Maximization

# Expectation & Maximization

- 앞서 GMM과 PPCA에서 해왔던 EM을 일반화한 형태를 써보면 다음과 같습니다.
- 1. Parameter  $\theta_{old}$ 를 초기화 한다.
- 2. E-step : posterior  $p(\mathbf{Z} | \mathbf{X}, \theta^{old})$ 를 구한다.
- 3. M-step : 새로운 parameter  $\theta^{new}$ 를 구한다.

$$\theta^{new} = \arg \max_{\theta} Q(\theta, \theta^{old})$$

where  $Q(\theta, \theta^{old}) = \mathbb{E}_{p(\mathbf{Z} | \mathbf{X}, \theta^{old})}[p(\mathbf{X}, \mathbf{Z} | \theta)]$

$$= \sum_{\mathbf{Z}} p(\mathbf{Z} | \mathbf{X}, \theta^{old}) \ln p(\mathbf{X}, \mathbf{Z} | \theta)$$

- 4. Log likelihood나 parameter 값이 수렴할 때까지  $\theta^{old}$ 를  $\theta^{new}$ 로 대체하고 E-step과 M-step을 반복한다.

# General EM-posterior 계산의 문제점

- EM 알고리즘은 E-step에서 posterior  $p_{\theta}(\mathbf{Z} | \mathbf{X})$ 를 구하는 과정이 필수적입니다.

- 

$$p_{\theta}(\mathbf{Z} | \mathbf{X}) = \frac{p_{\theta}(\mathbf{Z})p_{\theta}(\mathbf{X} | \mathbf{Z})}{p_{\theta}(\mathbf{X})}$$

- 그러나 분모에 있는  $p_{\theta}(\mathbf{X})$ 를 구하기 위해 적분 또는 summation을 해야 하는데 실제로 어려운 (intractable) 경우가 있습니다.

- 

$$p_{\theta}(\mathbf{X}) = \int p_{\theta}(\mathbf{Z})p_{\theta}(\mathbf{X} | \mathbf{Z})d\mathbf{Z}, \mathbf{Z} \text{가 continuous인 경우},$$

$$= \sum_{\mathbf{Z}} p_{\theta}(\mathbf{Z})p_{\theta}(\mathbf{X} | \mathbf{Z}), \mathbf{Z} \text{가 discrete인 경우}$$

# General EM-Variational Distribution의 도입

- E-step에서 posterior  $p_{\theta}(\mathbf{Z} | \mathbf{X})$ 를 구하는 것은 다음과 같이 KL-divergence를 최소화 하는 variational distribution  $q_{\phi}^{*}(\mathbf{Z})$ 를 구하는 것으로 대체할 수 있습니다.

$$q_{\phi}^{*}(\mathbf{Z}) = \arg \min_{q_{\phi}(\mathbf{Z})} \text{KL}(q || p)$$

where  $\text{KL}(q || p) = - \mathbb{E}_{q_{\phi}(\mathbf{Z})} [\ln p_{\theta}(\mathbf{Z} | \mathbf{X}) - q_{\phi}(\mathbf{Z})]$

$$= - \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \left\{ \frac{p_{\theta}(\mathbf{Z} | \mathbf{X})}{q_{\phi}(\mathbf{Z})} \right\}$$

- $\text{KL}(q || p)$ 은  $q_{\phi}(\mathbf{Z})$ 와  $p_{\theta}(\mathbf{Z} | \mathbf{X})$ 가 같을 때 0이 되므로  $p_{\theta}(\mathbf{Z} | \mathbf{X})$ 를 대신해  $q_{\phi}^{*}(\mathbf{Z})$ 를 사용할 수 있습니다.

# General EM-ELBO 사용

- posterior  $p_{\theta}(\mathbf{Z} | \mathbf{X})$ 를 구하기 어려운 경우  $\text{KL}(q||p)$  식을 직접 계산할 수 없으므로 우회하여 구해봅니다.
- Marginal log-likelihood  $\ln p_{\theta}(\mathbf{X})$ 와 ELBO라 불리는  $\mathcal{L}(q, \theta)$ 와 KL-divergence  $\text{KL}(q||p)$  간에는 다음과 같은 식이 성립합니다.
- 

$$\ln p_{\theta}(\mathbf{X}) = \mathcal{L}(q, \theta) + \text{KL}(q||p)$$

$$\text{where } \mathcal{L}(q, \theta) = \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \left\{ \frac{p_{\theta}(\mathbf{X}, \mathbf{Z})}{q_{\phi}(\mathbf{Z})} \right\}$$

$$\text{KL}(q||p) = - \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \left\{ \frac{p_{\theta}(\mathbf{Z} | \mathbf{X})}{q_{\phi}(\mathbf{Z})} \right\}$$

# General EM-ELBO 유도

- 

$$\ln p_{\theta}(\mathbf{X}) = \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln p_{\theta}(\mathbf{X})$$

$$= \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \frac{p_{\theta}(\mathbf{X}, \mathbf{Z})}{p_{\theta}(\mathbf{Z} | \mathbf{X})}$$

$$= \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \frac{p_{\theta}(\mathbf{X}, \mathbf{Z})}{q_{\phi}(\mathbf{Z})} - \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \frac{p_{\theta}(\mathbf{Z} | \mathbf{X})}{q_{\phi}(\mathbf{Z})}$$

$$= \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \frac{p_{\theta}(\mathbf{X}, \mathbf{Z})}{q_{\phi}(\mathbf{Z})} - \sum_{\mathbf{Z}} q_{\phi}(\mathbf{Z}) \ln \frac{p_{\theta}(\mathbf{Z} | \mathbf{X})}{q_{\phi}(\mathbf{Z})}$$

$$= \mathcal{L}(q, \theta) + \text{KL}(q \| p)$$

# General EM-ELBO 사용

- Parameter  $\theta$ 가 고정되어 있을 때 marginal log-likelihood  $\ln p_\theta(\mathbf{X})$ 는 고정이므로 KL-divergence  $\text{KL}(q\|p)$ 를 최소화하는 것은 ELBO  $\mathcal{L}(q, \theta)$ 를 최대화하는 것이 됩니다.
- 이점을 이용해 이용해  $\text{KL}(q\|p)$ 를 최소화하는 것이 아닌  $\mathcal{L}(q, \theta)$ 를 최대화하는 방향으로  $p_\theta(\mathbf{Z} | \mathbf{X})$ 를 근사하는  $q_\phi(\mathbf{Z})$ 를 구할 수 있습니다.
- 

$$q_\phi^*(\mathbf{Z}) = \arg \min_{q_\phi(\mathbf{Z})} \text{KL}(q\|p) = \arg \max_{q_\phi(\mathbf{Z})} \mathcal{L}(q, \theta)$$

# General EM-Variational EM

- EM 알고리즘에서 posterior를 구하는 E-step과 model parameter를 업데이트하는 M-step은 variational posterior  $q_\phi(\mathbf{Z})$ 를 이용하여 다음과 같이 일반화 할 수 있습니다.

## (Classic) EM Algorithm

- E-step : dataset 전체의 posterior  $p_\theta(\mathbf{Z}|\mathbf{X})$ 를 구한다.
- M-step : posterior에 의한 complete data log-likelihood의 기댓값을 최대화하는 parameters를 구한다.

$$\theta^{new} = \arg \max_{\theta} \mathbb{E}_{p_{\theta old}(\mathbf{Z}|\mathbf{X})} [\log p_\theta(\mathbf{X}, \mathbf{Z})]$$

## Variational EM Algorithm

- E-step : dataset 전체의 variational posterior를 구한다.  
$$q_\phi^*(\mathbf{Z}) = \arg \max_{q_\phi(\mathbf{Z})} \mathcal{L}(q, \theta)$$
- M-step : variational posterior에 의한 complete data log-likelihood의 기댓값을 최대화하는 parameters를 구한다.

$$\theta^{new} = \arg \max_{\theta} \mathbb{E}_{q_\phi^*(\mathbf{Z})} [\log p_\theta(\mathbf{X}, \mathbf{Z})]$$

# Variational Auto-Encoders

# Variational Auto-Encoders

- VAE (Variational Auto-Encoders)는 GAN과 더불어 딥러닝의 대표적인 생성모델(Generative Models)입니다.
- Auto-Encoder가 Z-variables의 분포를 알 수 없는 단점을 개선한 모델입니다.
- Z-variables의 분포를 Isotropic Gaussian 등으로 제한하여 자유롭게 컨트롤할 수 있다는 특징을 가집니다.
- VAE는 또한 Probabilistic PCA의 conditional distribution이 linear transform 모델에 제한된다는 단점을 개선한 모델로 생각할 수 있습니다.
- Neural Networks의 non-linearity를 posterior와 conditional distribution을 형성하는데 사용해 기존 머신러닝에서 보여줄 수 없었던 유연한 표현 능력을 가집니다.

# Variational Auto-Encoders

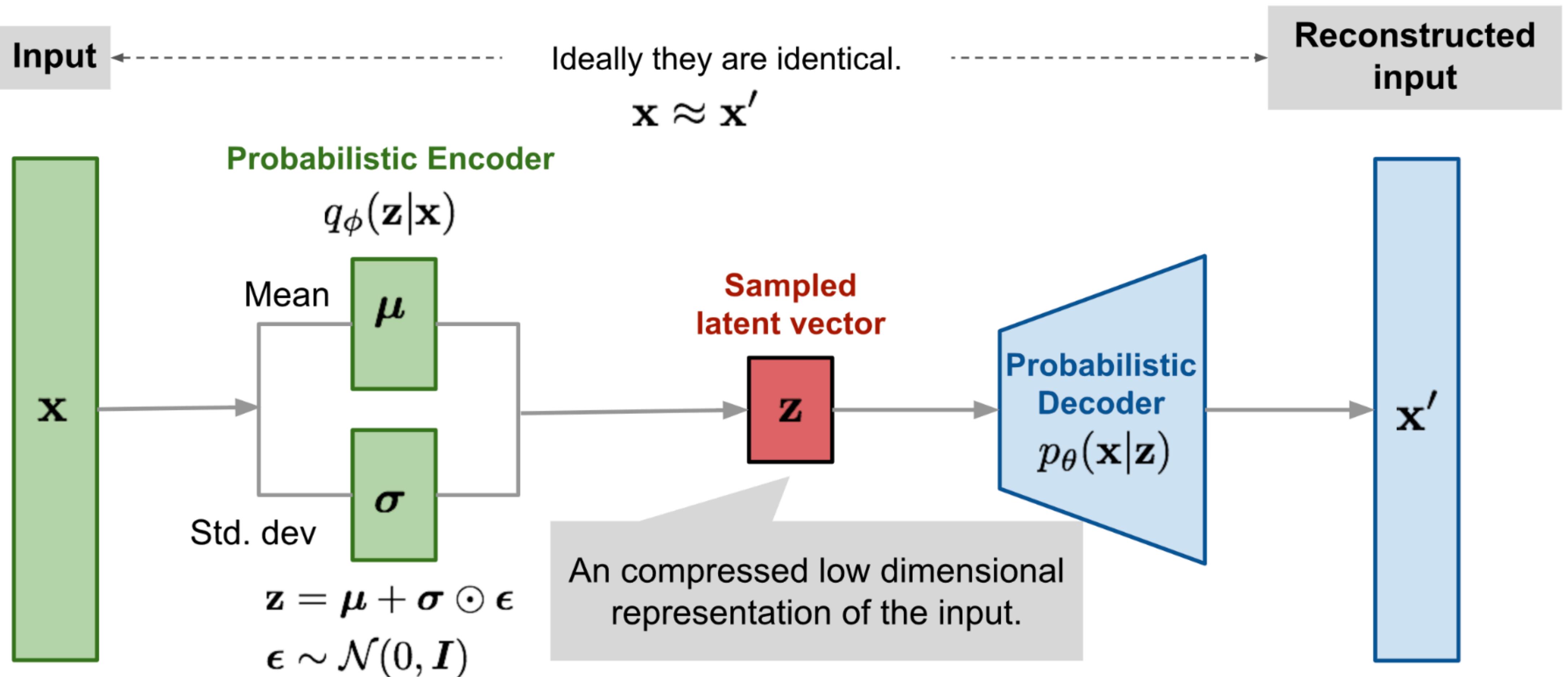


Fig. 9. Illustration of variational autoencoder model with the multivariate Gaussian assumption.

# Variational Auto-Encoders

Called a variational method because it derives from the  
**Calculus of Variations.**

## Functions:

- Variables as input, output is a value.
- Full and partial derivatives  $\frac{df}{dx}$
- E.g., Maximise likelihood  $p(x|\theta)$  w.r.t. parameters  $\theta$

## Functionals:

- Functions as input, output is a value.
- Functional derivatives  $\frac{\delta F}{\delta f}$
- E.g., Maximise the entropy  $H[p(x)]$  w.r.t.  $p(x)$

*We exploit both types of derivatives  
in variational inference.*

# Variational Auto-Encoders

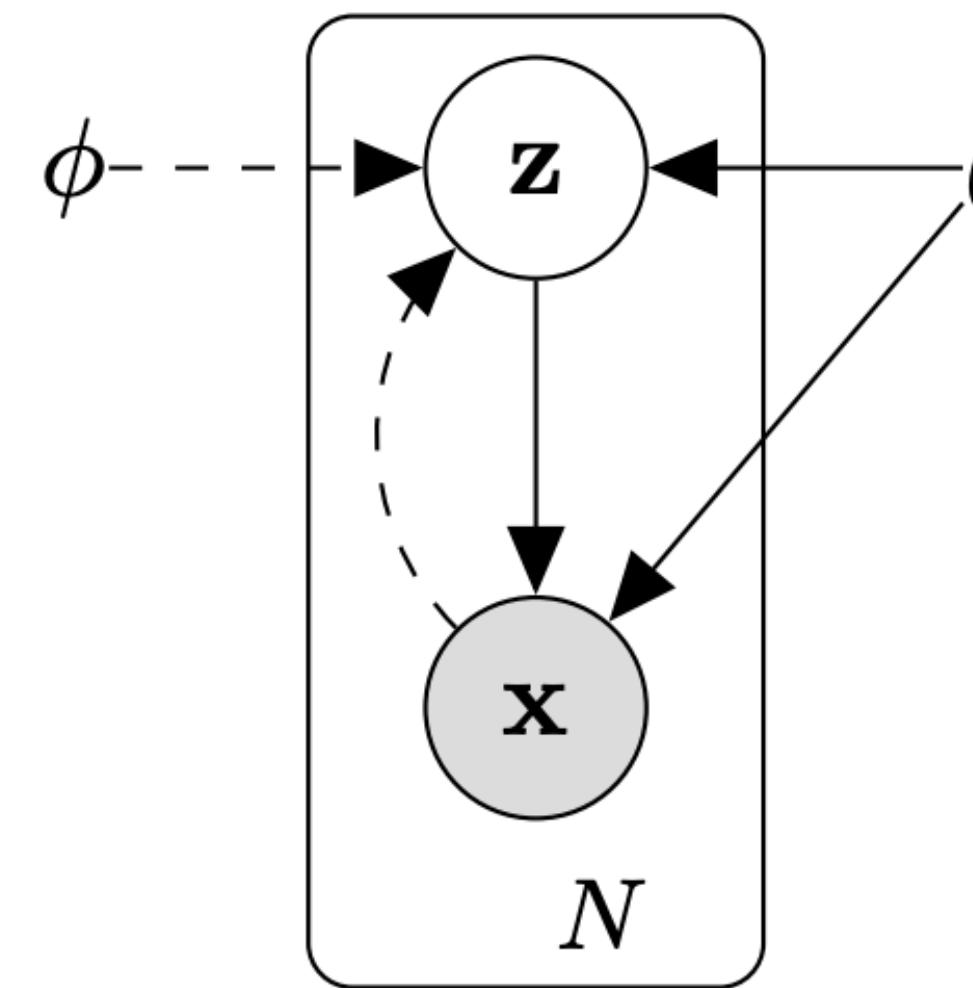


Figure 1: The type of directed graphical model under consideration. Solid lines denote the generative model  $p_\theta(\mathbf{z})p_\theta(\mathbf{x}|\mathbf{z})$ , dashed lines denote the variational approximation  $q_\phi(\mathbf{z}|\mathbf{x})$  to the intractable posterior  $p_\theta(\mathbf{z}|\mathbf{x})$ . The variational parameters  $\phi$  are learned jointly with the generative model parameters  $\theta$ .

# Variational Auto-Encoders

- ELBO(Evidence Lower BOund) 설명

## 2.2 The variational bound

The marginal likelihood is composed of a sum over the marginal likelihoods of individual datapoints

$\log p_{\theta}(\mathbf{x}^{(1)}, \dots, \mathbf{x}^{(N)}) = \sum_{i=1}^N \log p_{\theta}(\mathbf{x}^{(i)})$ , which can each be rewritten as:

$$\log p_{\theta}(\mathbf{x}^{(i)}) = D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\theta}(\mathbf{z}|\mathbf{x}^{(i)})) + \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) \quad (1)$$

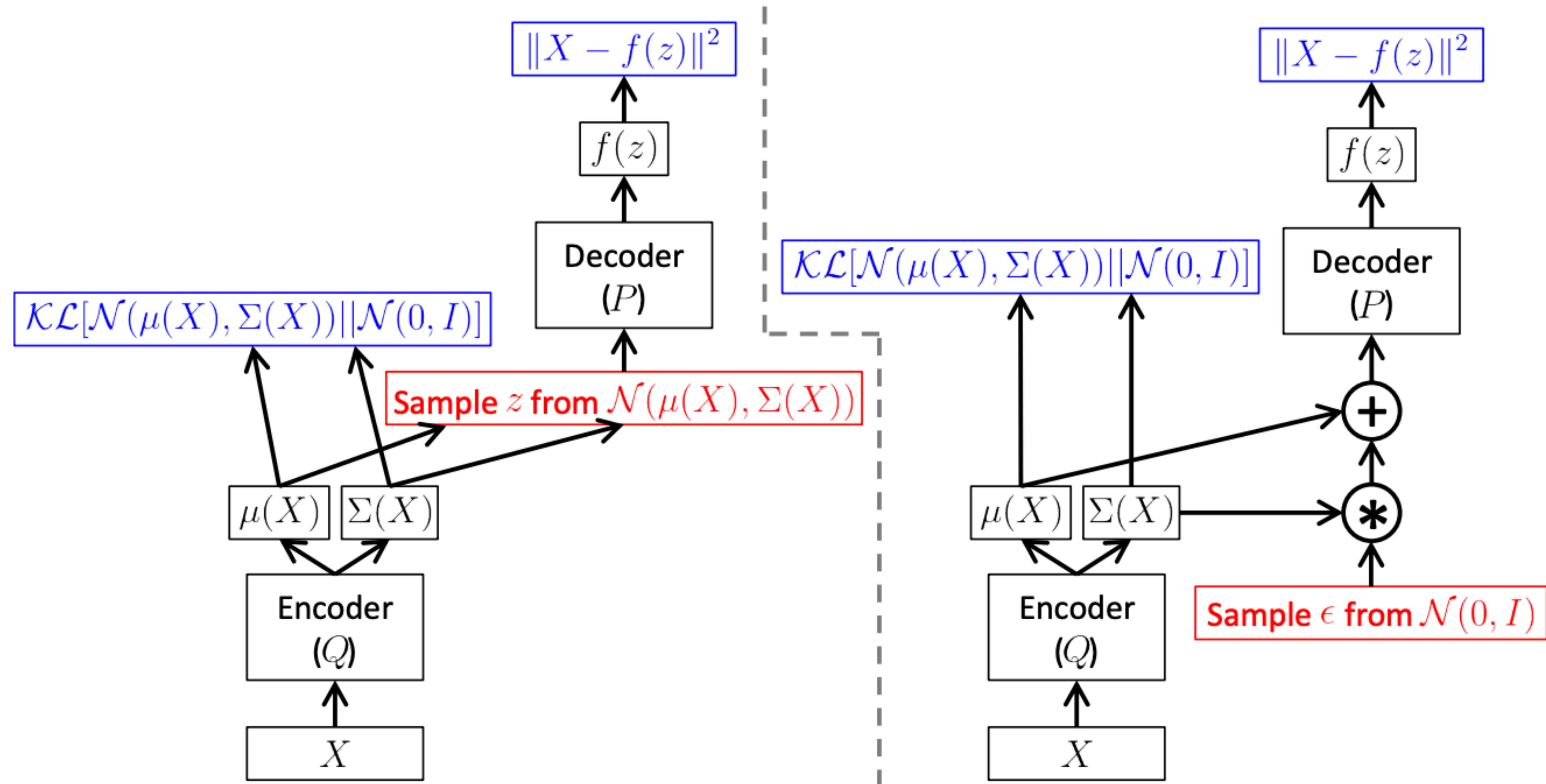
The first RHS term is the KL divergence of the approximate from the true posterior. Since this KL-divergence is non-negative, the second RHS term  $\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)})$  is called the (variational) lower bound on the marginal likelihood of datapoint  $i$ , and can be written as:

$$\log p_{\theta}(\mathbf{x}^{(i)}) \geq \mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x})} [-\log q_{\phi}(\mathbf{z}|\mathbf{x}) + \log p_{\theta}(\mathbf{x}, \mathbf{z})] \quad (2)$$

which can also be written as:

$$\mathcal{L}(\theta, \phi; \mathbf{x}^{(i)}) = -D_{KL}(q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})||p_{\theta}(\mathbf{z})) + \mathbb{E}_{q_{\phi}(\mathbf{z}|\mathbf{x}^{(i)})} [\log p_{\theta}(\mathbf{x}^{(i)}|\mathbf{z})] \quad (3)$$

# Variational Auto-Encoders



# Variational Auto-Encoders

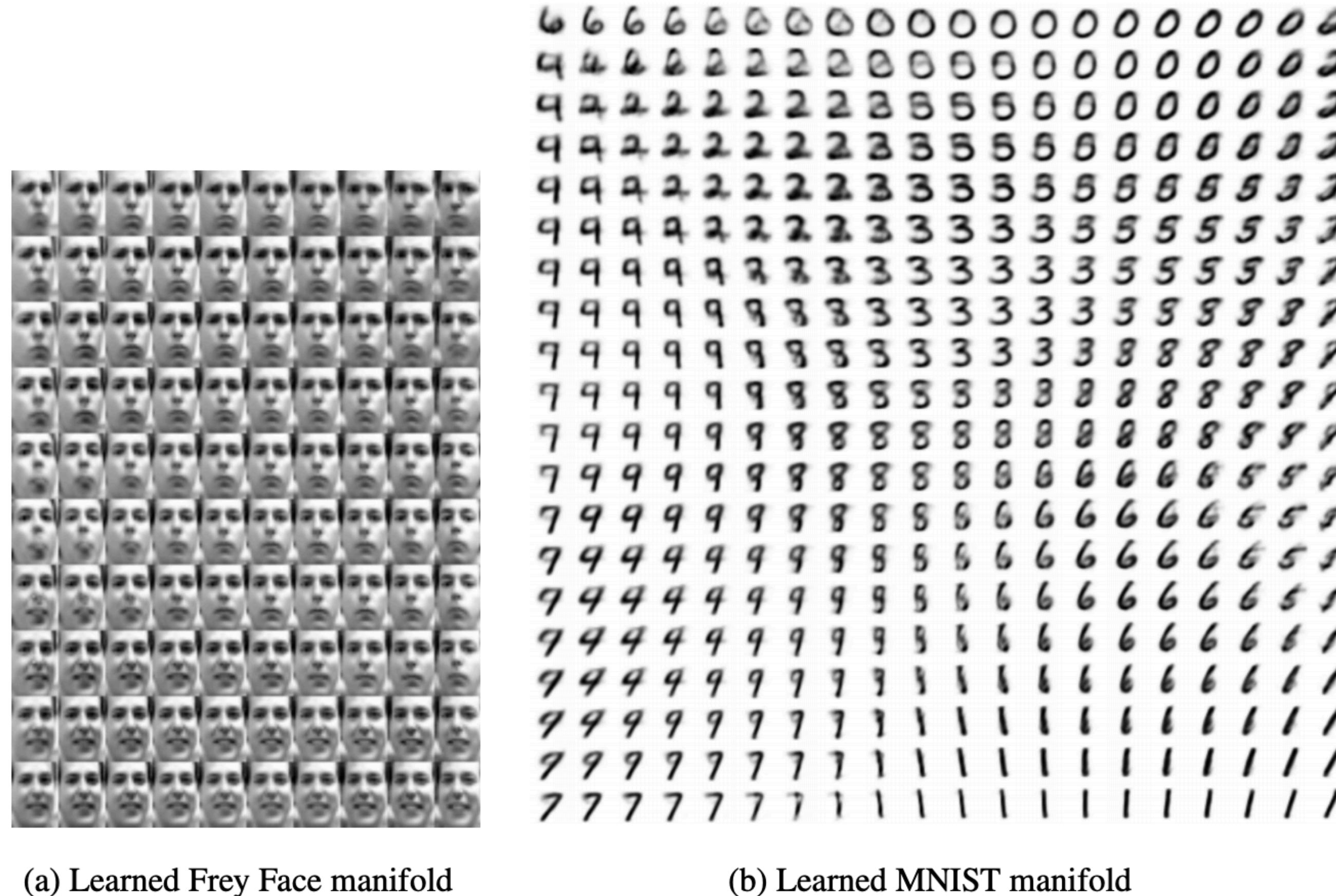


Figure 4: Visualisations of learned data manifold for generative models with two-dimensional latent space, learned with AEVB. Since the prior of the latent space is Gaussian, linearly spaced coordinates on the unit square were transformed through the inverse CDF of the Gaussian to produce values of the latent variables  $\mathbf{z}$ . For each of these values  $\mathbf{z}$ , we plotted the corresponding generative  $p_{\theta}(\mathbf{x}|\mathbf{z})$  with the learned parameters  $\theta$ .

# Variational Auto-Encoders

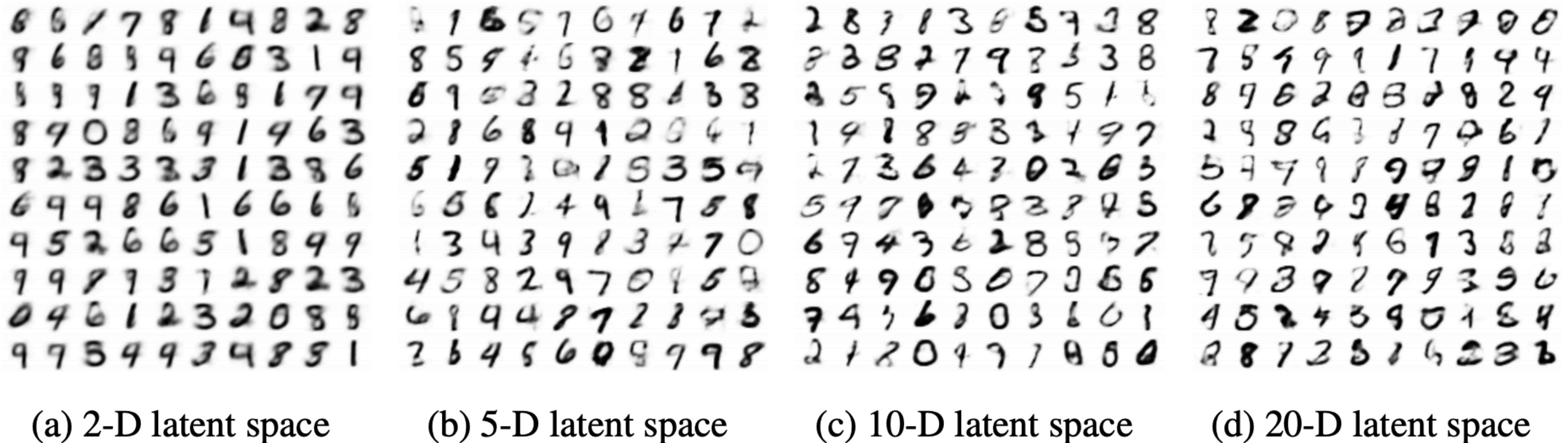
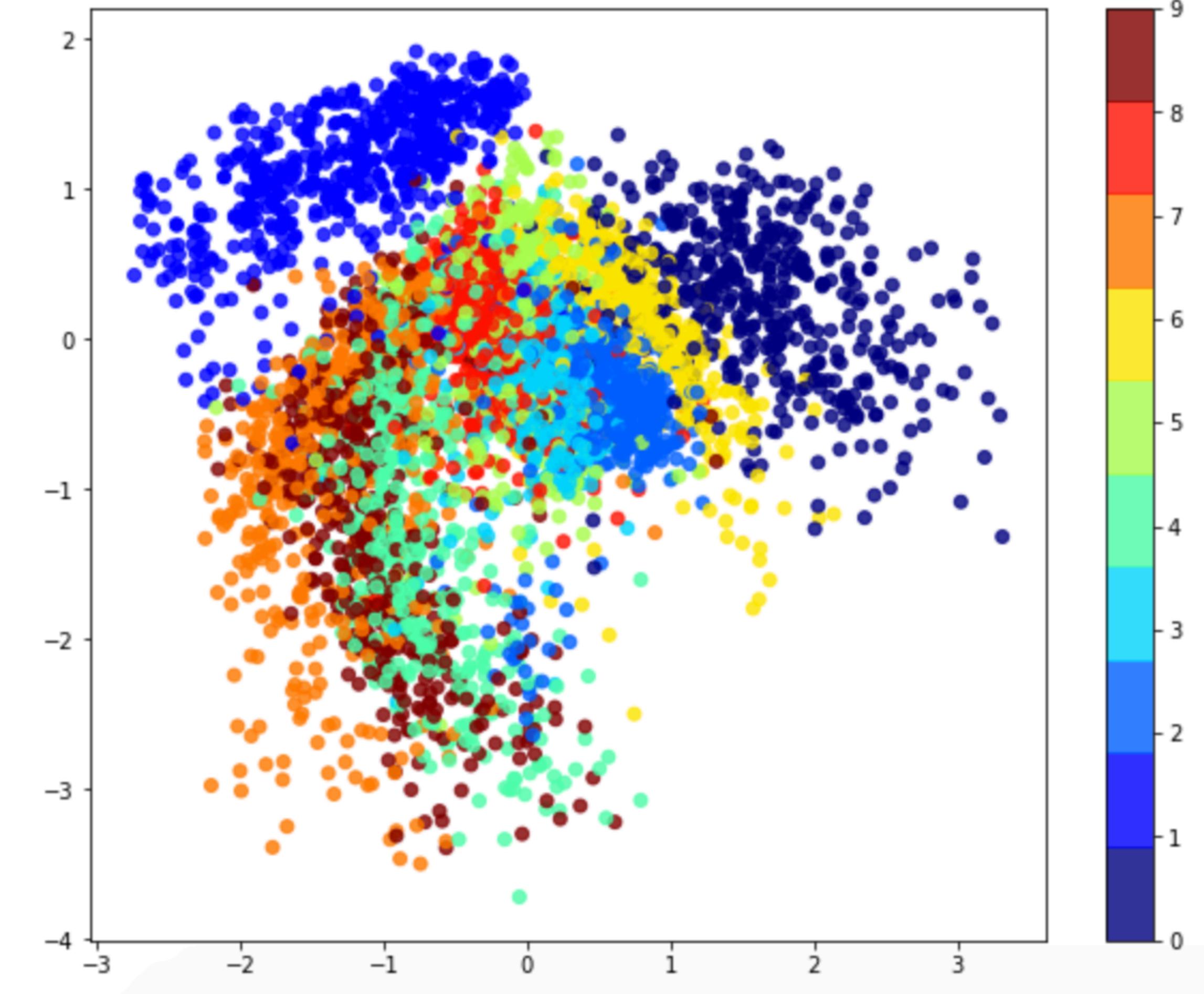
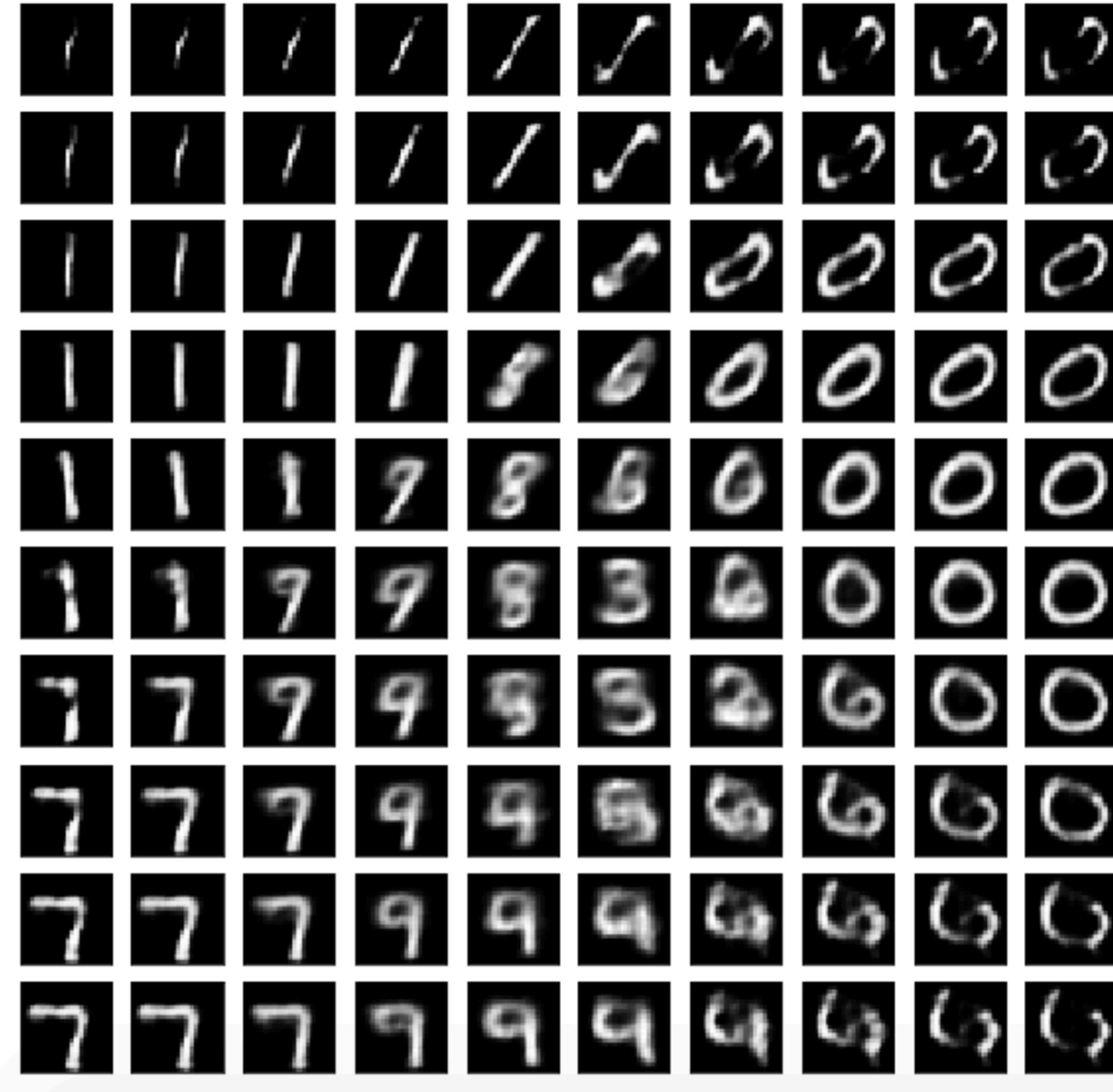


Figure 5: Random samples from learned generative models of MNIST for different dimensionalities of latent space.

# Variational Auto-Encoders



# Hierarchical VAE

# Hierachical VAE

- VAE는 트레이닝이 쉽고 controllable한 latent vector를 얻는다는 점에서 매우 강력한 모델이지만 결과물이 blurry하다는 단점 때문에 generative model로 사용되기 보다는 representation learning의 역할에 치중해왔습니다.
- VAE가 blurry한 결과물을 낳는 이유는 prior  $p_\theta(\mathbf{z})$ 의 분포를 fully-factorized Gaussian으로 정하기 때문입니다.
- 그래서 NVAE (Nouveau VAE), VDVAE (Very Deep VAE) 등, prior  $p_\theta(\mathbf{z})$ 를 계층별로 나누고 상위 계층이 하위 계층에 dependent하도록 만드는 시도들이 생겼습니다.

$$p_\theta(\mathbf{z}) = p_\theta(\mathbf{z}_0) p_\theta(\mathbf{z}_1 \mid \mathbf{z}_0) \dots p_\theta(\mathbf{z}_N \mid \mathbf{z}_{<N})$$

- 이에 따라 variational posterior  $q_\phi(\mathbf{z} \mid \mathbf{x})$ 도 계층간에 dependent하도록 factorize합니다.

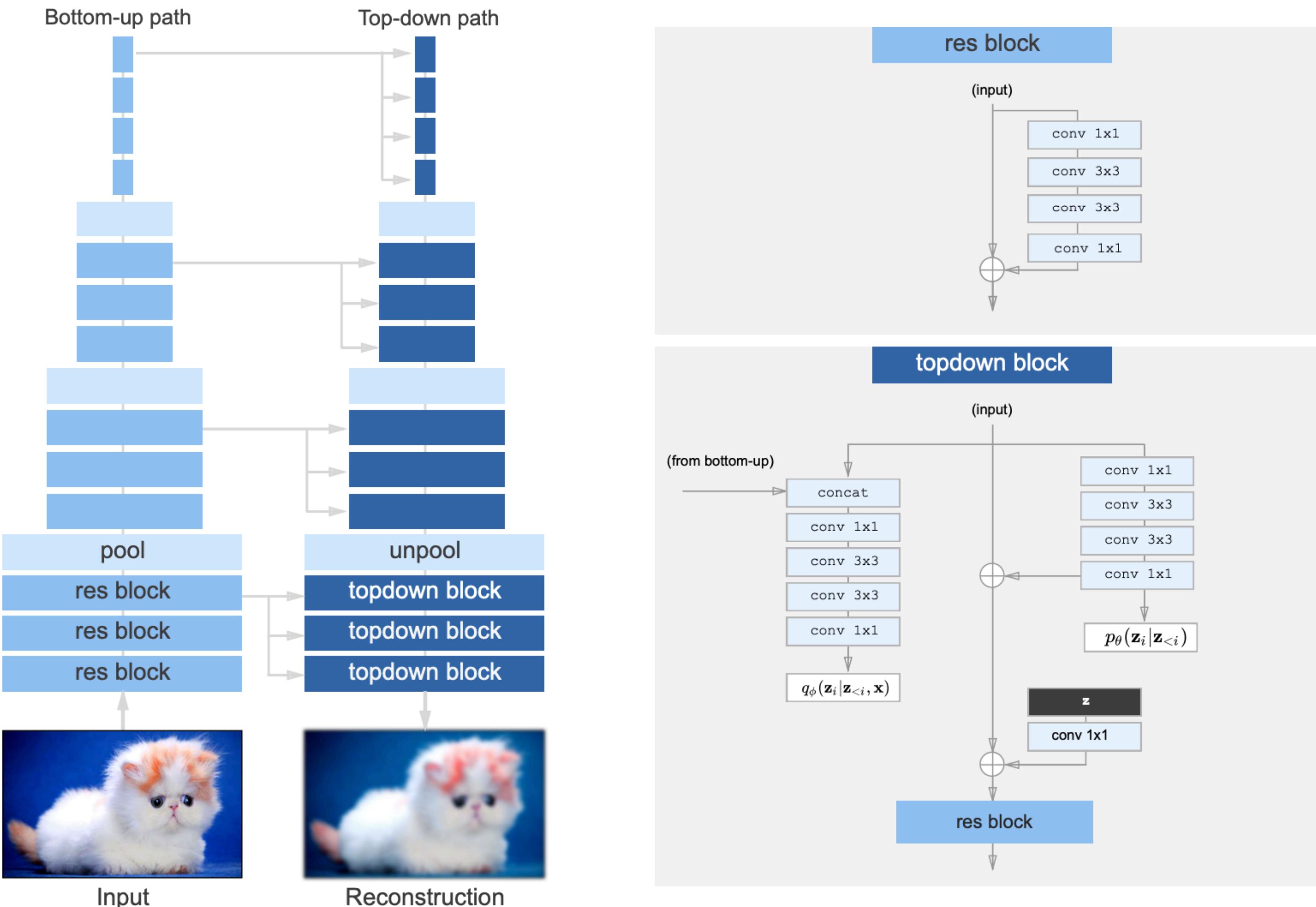
$$q_\phi(\mathbf{z} \mid \mathbf{x}) = q_\phi(\mathbf{z}_0 \mid \mathbf{x}) q_\phi(\mathbf{z}_1 \mid \mathbf{z}_0, \mathbf{x}) \dots q_\phi(\mathbf{z}_N \mid \mathbf{z}_{<N}, \mathbf{x})$$

# Hierachical VAE



Child, Rewon. "Very deep vaes generalize autoregressive models and can outperform them on images." arXiv preprint arXiv:2011.10650 (2020).

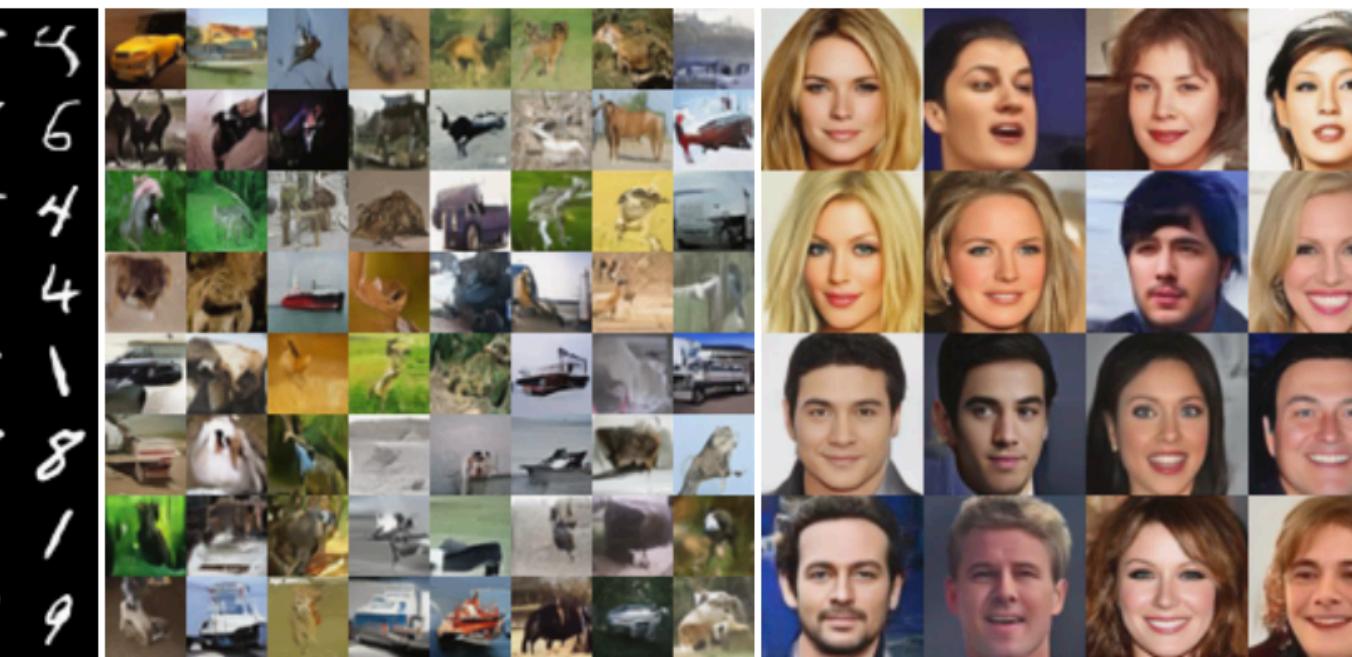
# Hierarchical VAE



Child, Rewon. "Very deep vaes generalize autoregressive models and can outperform them on images." arXiv preprint arXiv:2011.10650 (2020).

# Hierachical VAE

5	8	3	6	7	7	5	5
9	4	3	2	5	9	5	6
1	3	7	9	7	/	5	4
6	1	0	7	3	1	6	4
3	2	8	1	9	4	0	1
4	3	3	2	6	8	5	8
5	5	4	3	4	1	3	1
6	7	3	8	7	4	8	9



(a) MNIST ( $t = 1.0$ )

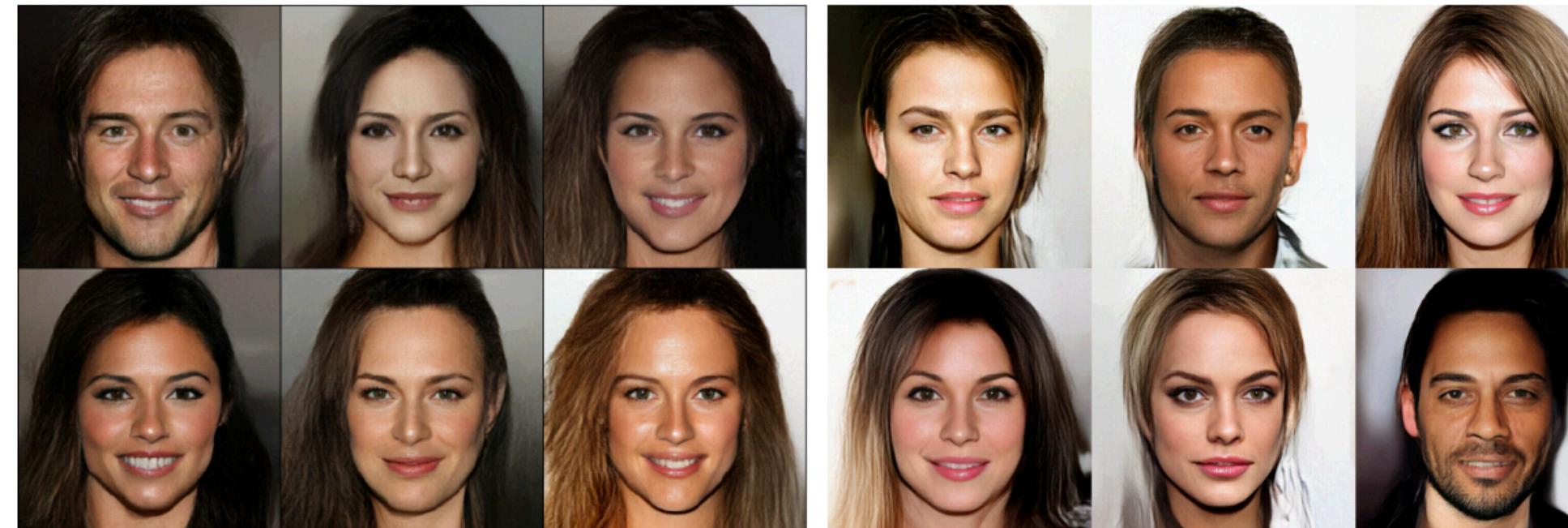
(b) CIFAR-10 ( $t = 0.7$ )

(c) CelebA 64 ( $t = 0.6$ )



(d) CelebA HQ ( $t = 0.6$ )

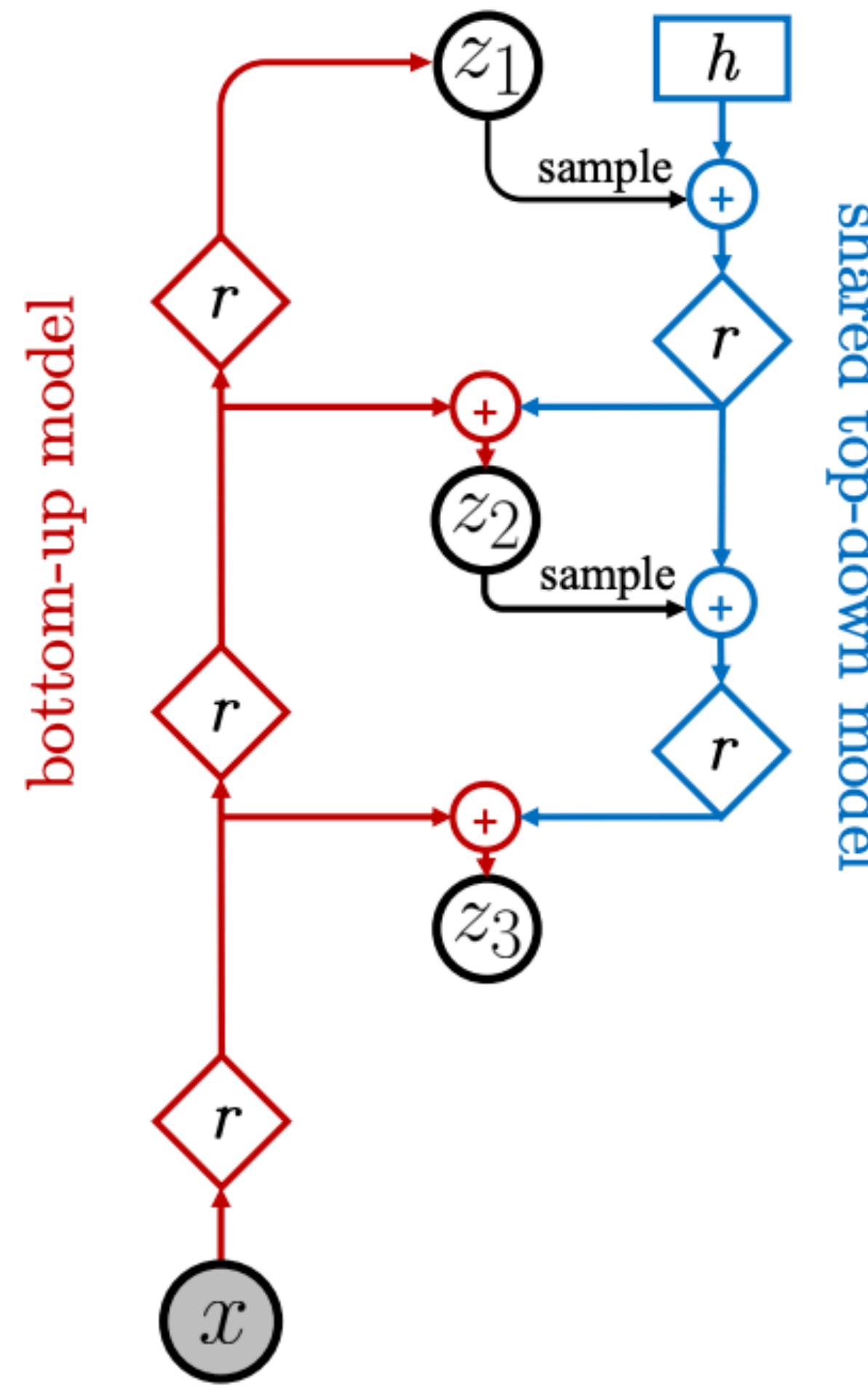
(e) FFHQ ( $t = 0.5$ )



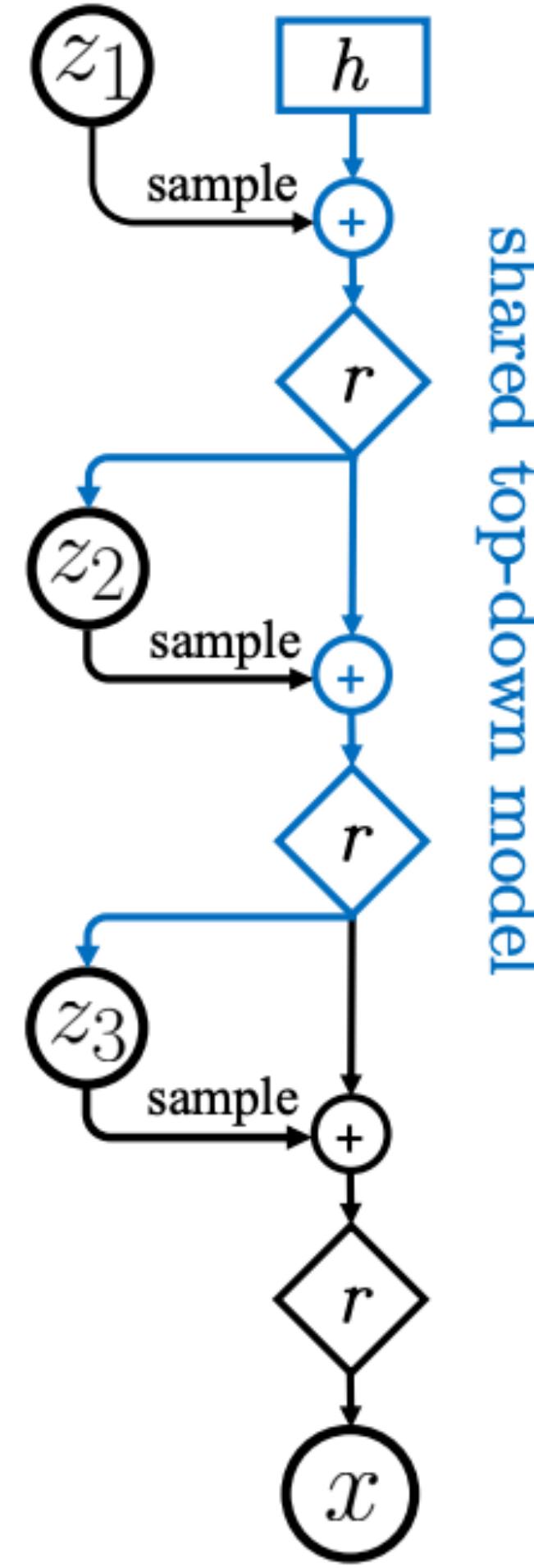
(f) MaCow [67] trained on CelebA HQ ( $t = 0.7$ )

(g) Glow [62] trained on CelebA HQ ( $t = 0.7$ )

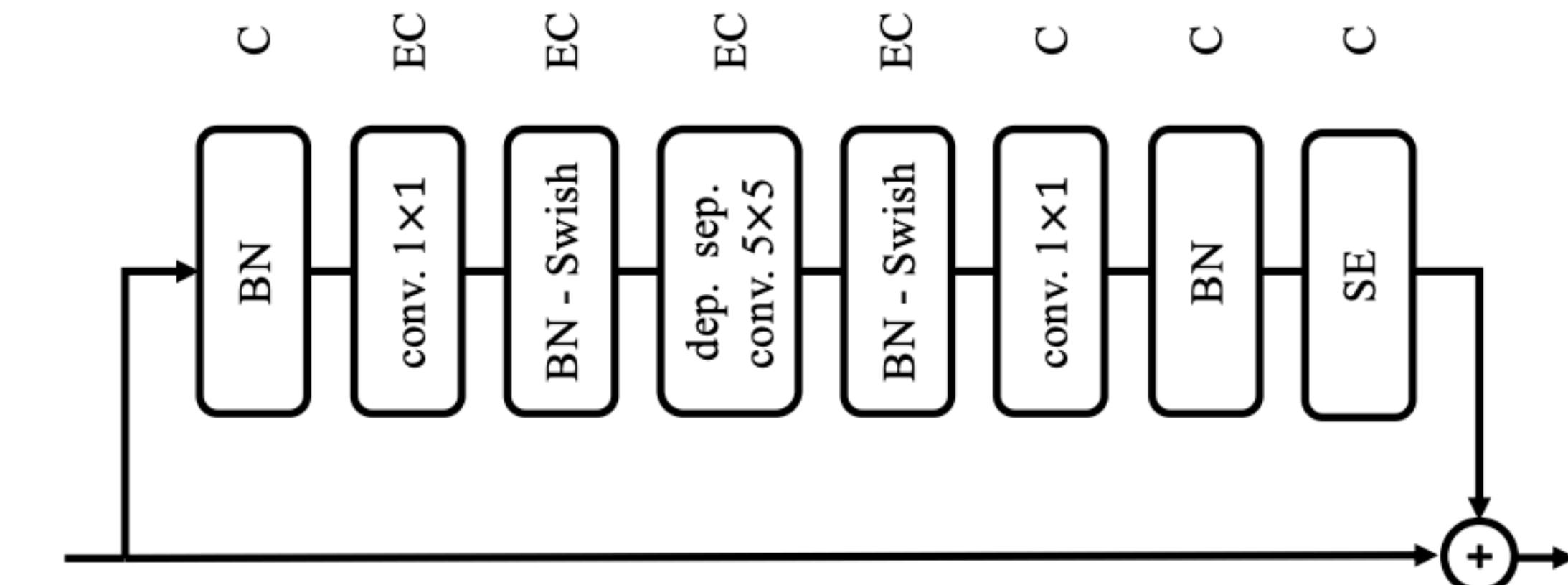
# Hierarchical VAE



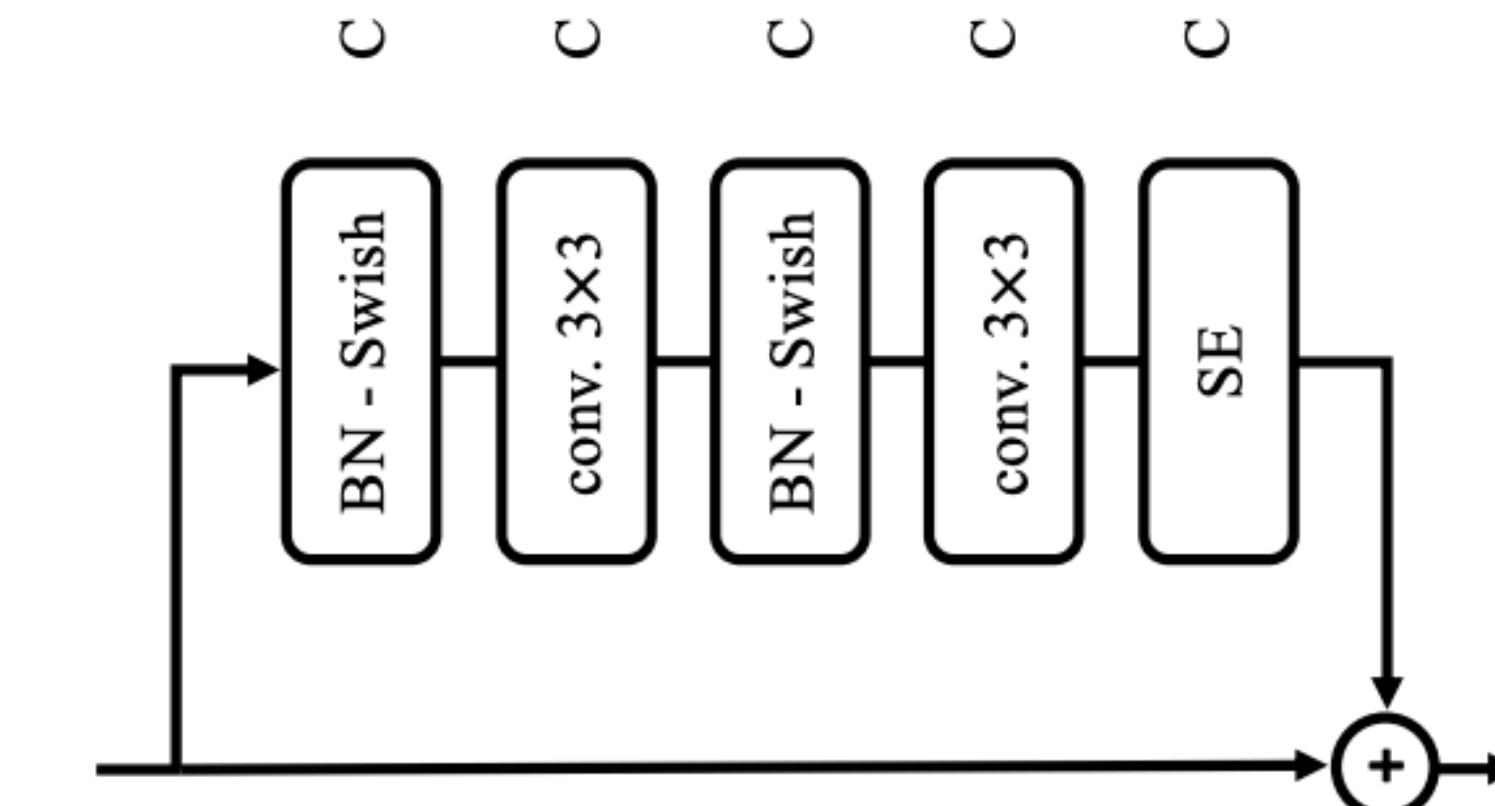
(a) Bidirectional Encoder (b) Generative Model



shared top-down model



(a) Residual Cell for NVAE Generative Model

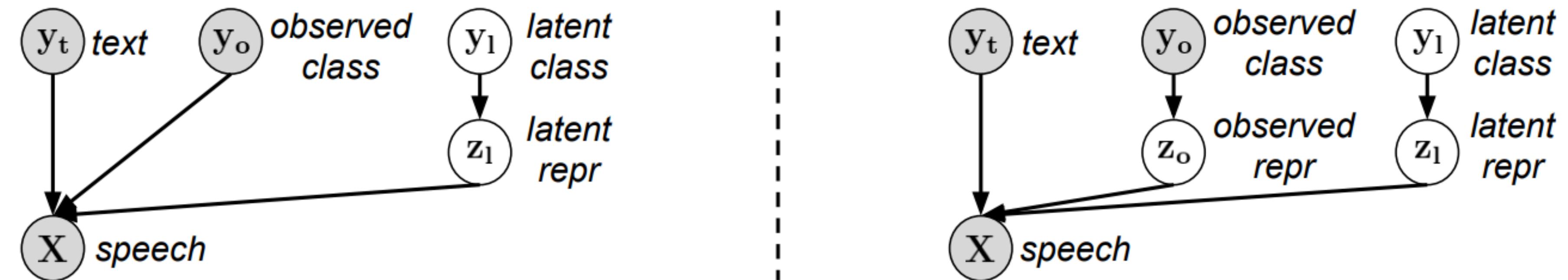


(b) Residual Cell for NVAE Encoder

# VAE in Speech Synthesis

# VAE in Speech Synthesis

- VAE는 음성 합성에서도 자주 접목되어 사용되는 모델입니다.
- 같은 말을 하더라도 사람들마다 서로 다른 빠르기, 악센트, 음고, 리듬을 가지고 말을 하는데 이러한 정보는 data 자체에 annotation되기 힘듭니다.
- VAE는 이러한 정보들을 train때에 z space상에 encoding시키고, inference때에 원하는 특징을 가진 음성을 합성하도록 해줍니다.
- Hierarchical generative modeling for controllable speech synthesis 논문에서는 Gaussian Mixture VAE를 사용하여 다양한 사람들의 목소리 특징을 discrete한 정보와 continuous한 정보로 나누어 분석하였습니다.
- 예제 : [https://google.github.io/tacotron/publications/gmvae\\_controllable\\_tts/index.html](https://google.github.io/tacotron/publications/gmvae_controllable_tts/index.html)

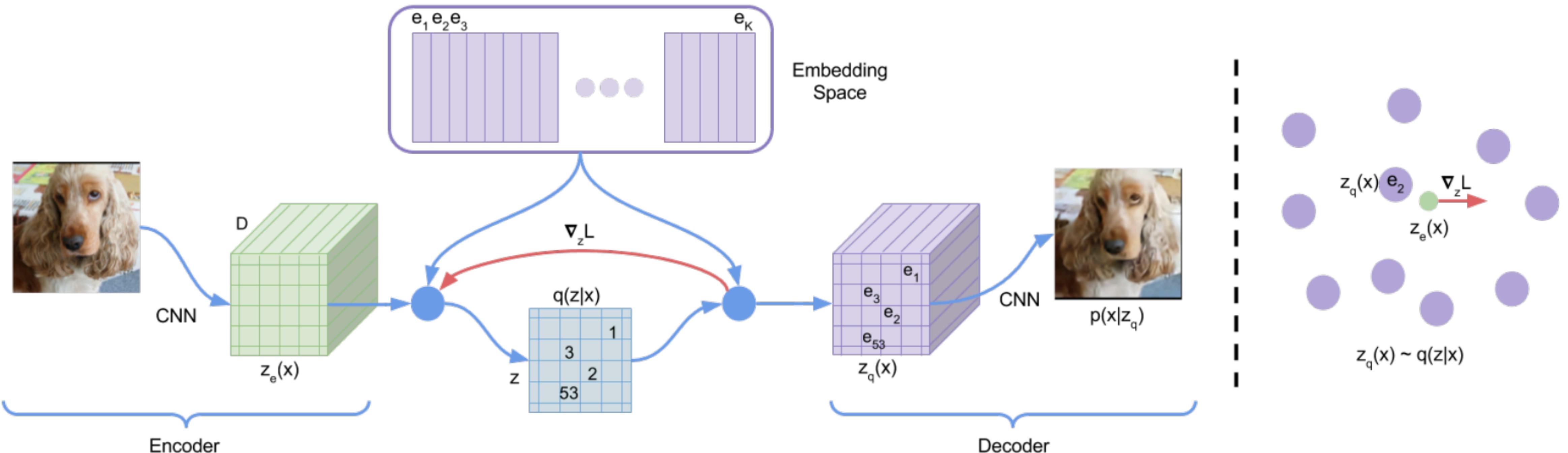


**VQ-VAE**  
(Vector Quantized VAE)

# VQ-VAE

- VQ-VAE는 Neural Discrete Representation Learning 논문에서 제안된 모델입니다.
- Vector quantized-VAE의 약자이며 VAE는 z space가 continuous space인데 반해, VQ-VAE는 discrete space로 설정합니다.
- z space를 discrete space로 설정하면 z vector들을 NLP에서 다루는 token들처럼 다룰 수 있다는 장점이 있습니다.
- VQ-VAE는 VAE와 마찬가지로 encoder와 decoder로 이루어져 있고, 추가적으로 codebook을 사용합니다.
- Encoding: Encoder는 data  $x$ 를 인코딩하여  $z_e(x)$  vector를 만듭니다.
- Quantization:  $z_e(x)$ 와 codebook에서 가장 가까운 vector를  $z_q(x)$ 로 두어 quantizaton을 수행합니다.
- Decoding: Decoder에  $z_q(x)$ 를 입력하고 출력된 data  $x'$ 를 받아 본래 data  $x$ 와 MSE loss를 취해 복원되도록 만듭니다.

# VQ-VAE



$$\mathcal{L}(\mathbf{x}, D(\mathbf{e})) = \|\mathbf{x} - D(\mathbf{e})\|_2^2 + \|sg[E(\mathbf{x})] - \mathbf{e}\|_2^2 + \beta \|sg[\mathbf{e}] - E(\mathbf{x})\|_2^2$$

**Reconstruction loss**

**Codebook loss**

**Commitment loss**

# VQ-VAE

- VQ-VAE의 loss는 reconstruction loss와 codebook loss, 그리고 commitment loss로 구성됩니다.
- Reconstruction loss : auto-encoder, VAE와 마찬가지로 decoding된 결과가 원본과 같아지도록 만드는 기능을 합니다.
- Codebook loss : codebook의 vector들이 embedding된 vector들과 가까워지도록 만듭니다.
- Commitment loss : embedding된 vector들이 codebook의 vector들과 가까워지도록 만듭니다.

$$\mathcal{L}(\mathbf{x}, D(\mathbf{e})) = \|\mathbf{x} - D(\mathbf{e})\|_2^2 + \|sg[E(\mathbf{x})] - \mathbf{e}\|_2^2 + \beta \|sg[\mathbf{e}] - E(\mathbf{x})\|_2^2$$

Reconstruction loss

Codebook loss

Commitment loss

# VQ-VAE



Figure 2: Left: ImageNet 128x128x3 images, right: reconstructions from a VQ-VAE with a 32x32x1 latent space, with K=512.

# VQ-VAE

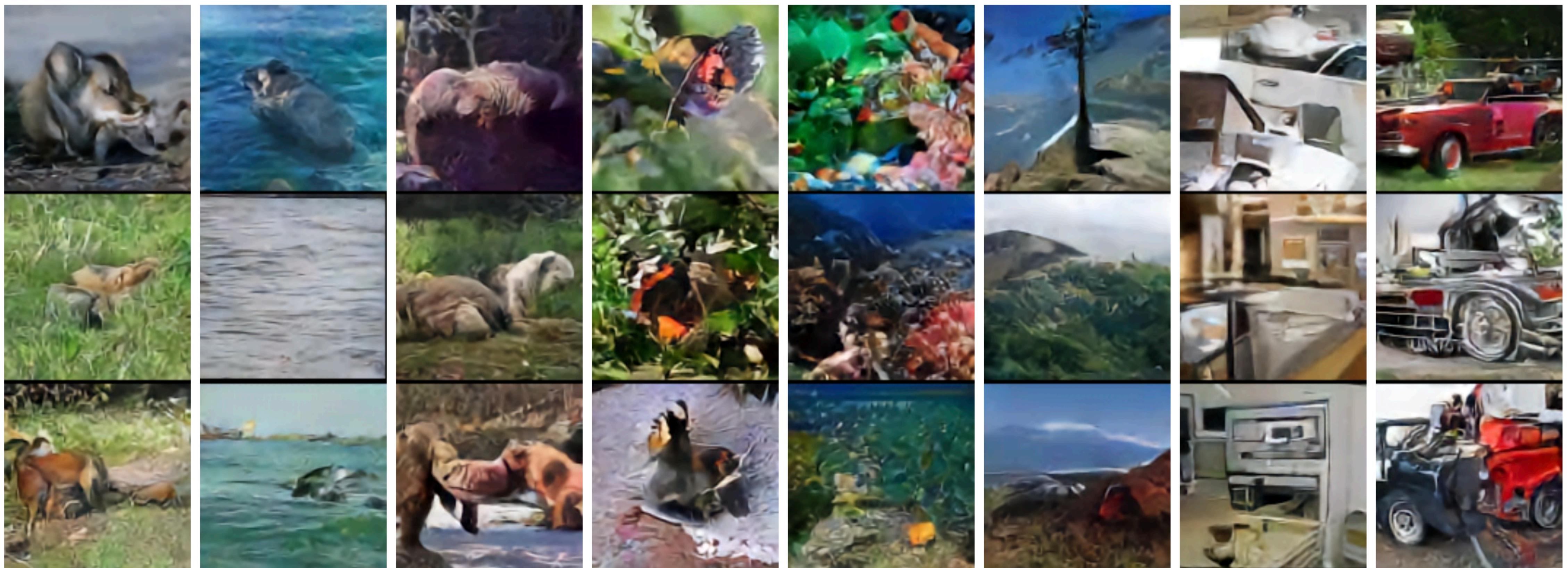
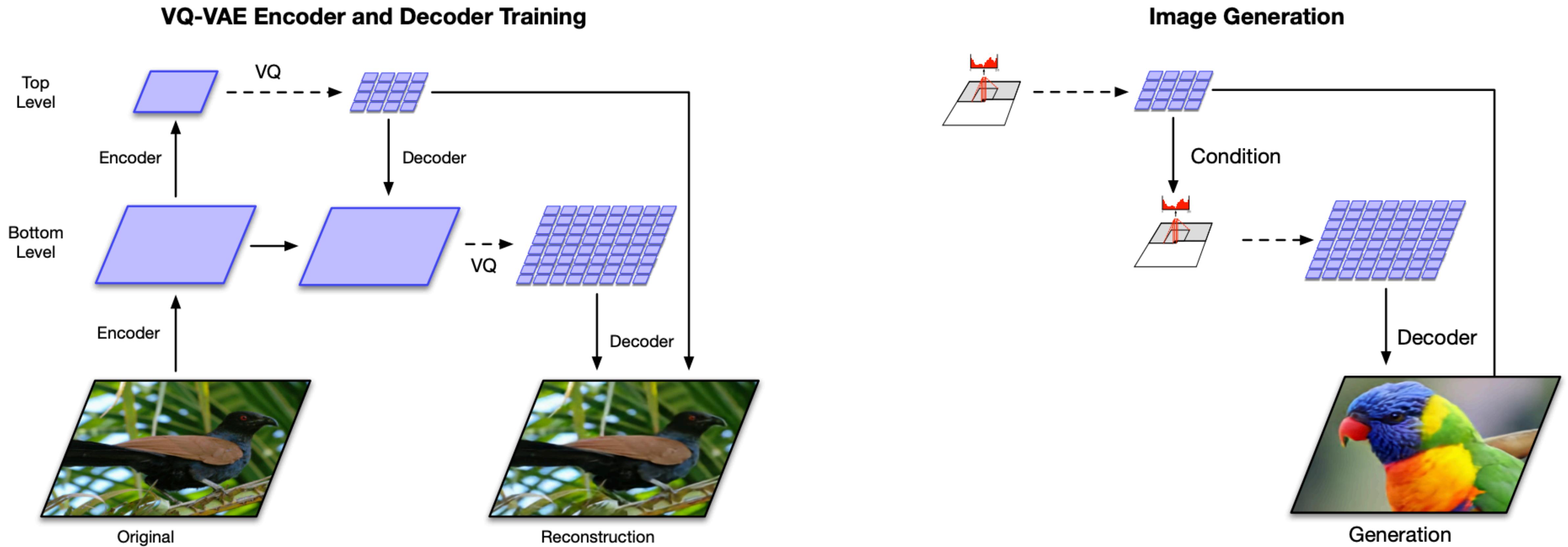


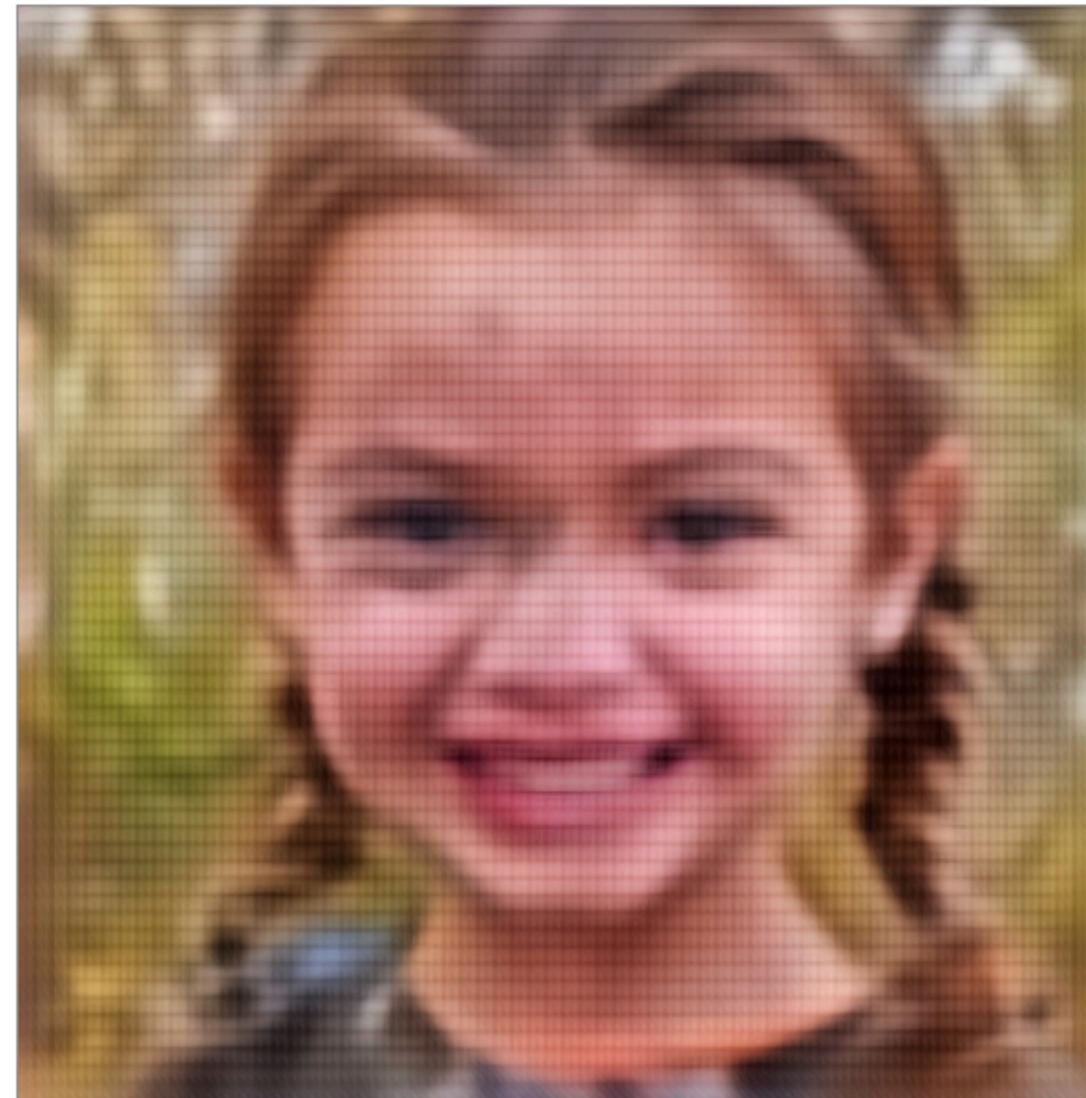
Figure 3: Samples (128x128) from a VQ-VAE with a PixelCNN prior trained on ImageNet images  
From left to right: kit fox, gray whale, brown bear, admiral (butterfly), coral reef, alp, microwave pickup.

# VQ-VAE 2



Razavi, Ali, Aaron van den Oord, and Oriol Vinyals. "Generating diverse high-fidelity images with vq-vae-2." *Advances in neural information processing systems*. 2019.

# VQ-VAE 2



$h_{\text{top}}$



$h_{\text{top}}, h_{\text{middle}}$



$h_{\text{top}}, h_{\text{middle}}, h_{\text{bottom}}$

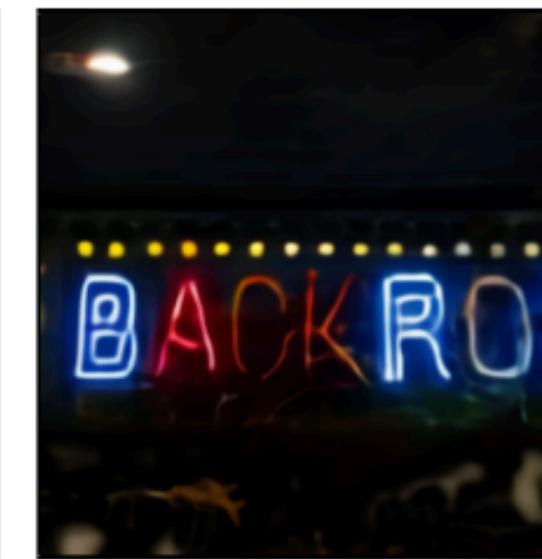
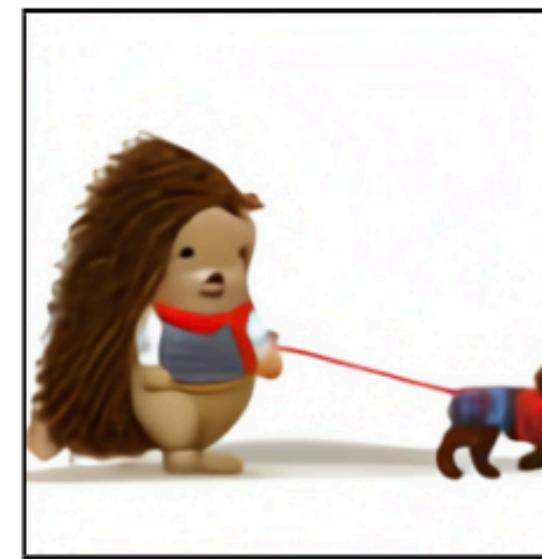
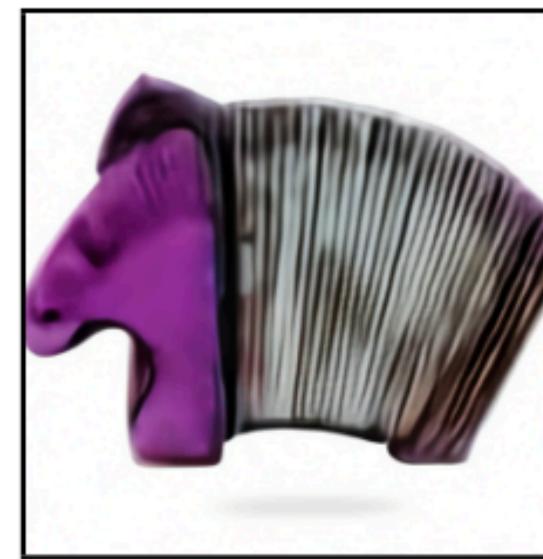
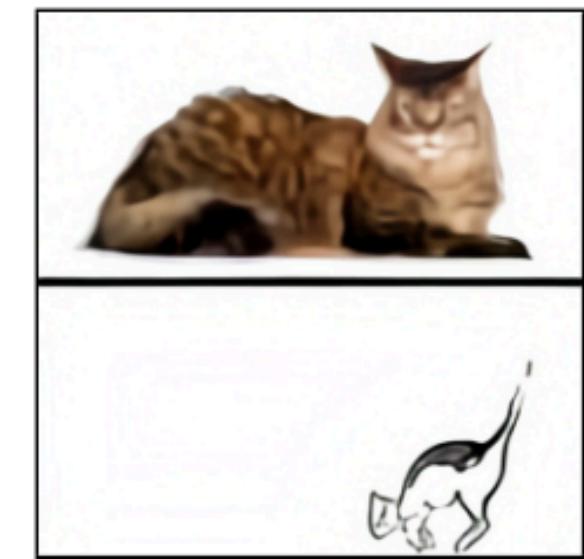
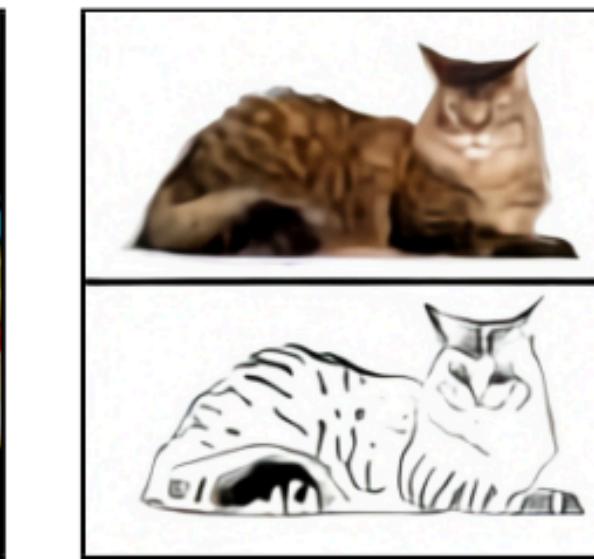
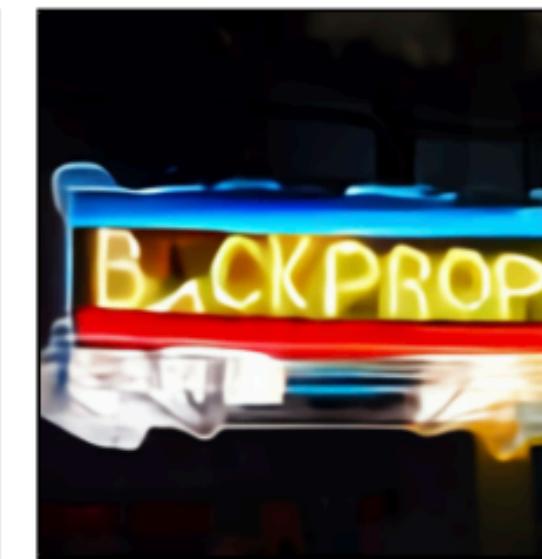
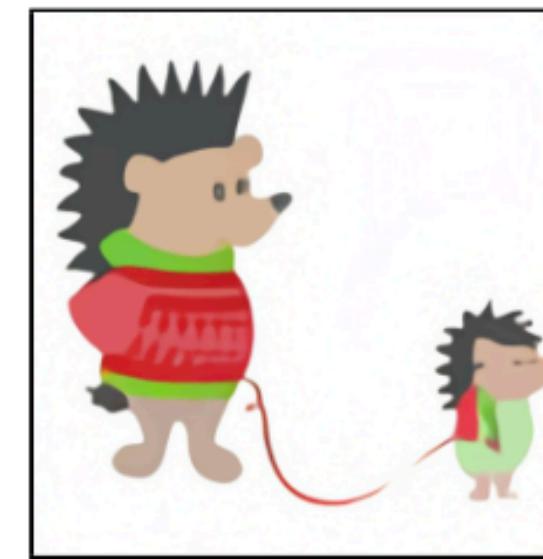
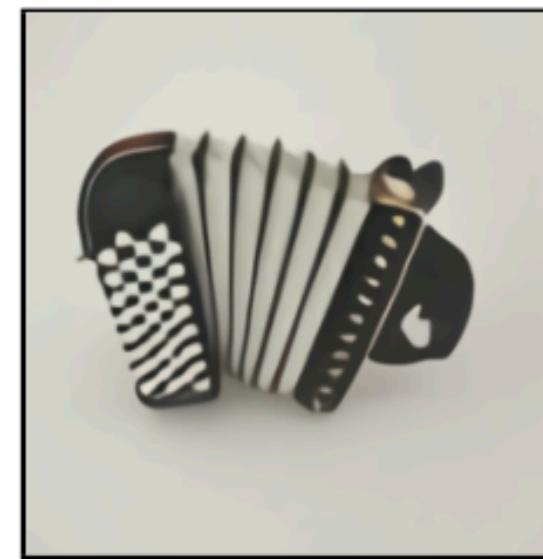


Original

# DALL·E

- DALL·E는 OpenAI에서 Zero-Shot Text-to-Image Generation 논문을 통해 발표된 모델입니다.
- Text를 입력으로 받아 해당하는 설명에 맞는 이미지를 출력해줍니다.
- 트레이닝은 두 개의 stage로 구성되며 다음과 같습니다. (논문에서 인용)
  - “Stage 1. We train a discrete variational autoencoder (dVAE) to compress each  $256 \times 256$  RGB image into a  $32 \times 32$  grid of image tokens, each element of which can assume 8192 possible values. This reduces the context size of the transformer by a factor of 192 without a large degradation in visual quality.”
  - “Stage 2. We concatenate up to 256 BPE-encoded text tokens with the  $32 \times 32 = 1024$  image tokens, and train an autoregressive transformer to model the joint distribution over the text and image tokens.”
- Examples : <https://openai.com/blog/dall-e/>

# DALL·E



(a) a tapir made of accordion.  
a tapir with the texture of an  
accordion.

(b) an illustration of a baby  
hedgehog in a christmas  
sweater walking a dog

(c) a neon sign that reads  
“backprop”. a neon sign that  
reads “backprop”. backprop  
neon sign

(d) the exact same cat on the  
top as a sketch on the bottom

# DALL·E

