Auto-Encoding Variational Bayes (Variational Autoencoder)

ICLR 2014

Diederik P. Kingma , Max Welling

리뷰 발표자: 장형원

Contents

Math review for Bayes' rule

Math Review : Conditional Probability and Bayes' Rule

- **Conditional Probability for events A and B**

$$P(B|A)P(A) = P(A \cap B)$$

- **Bayes' Rule**

$$P(B|A) = \frac{P(A|B)P(B)}{P(A)}$$

Math Review : Random variable

- **Discrete random variable(이산 확률 변수)**
  - **Probability mass function(확률 질량 함수)**

$$p_X : \mathbb{R} \to [0,1]$$

$$p_X(x) = P(X = x)$$

  - p, non-negative, sum up to 1.

- **Conditional random variable(연속 확률 변수)**
  - **Probability density function(확률 밀도 함수)**

$$P(X \in A) = \int_A f\,dx$$

  - f, non-negative Lebesgue-integrable function, integral 1.

Math Review : Conditional Probability and Bayes' Rule (2)

- **Conditional Probability Distribution for continuous random variables**
  - p(x|y) : probability density function of X given the occurrence of the value y of Y.

$$P(X \in A | Y = y) = \int_A p(x|y)dx$$

$$p(x|y) = \frac{p(x,y)}{p_Y(y)}$$

Math Review : Conditional Probability and Bayes' Rule (2)

- **Bayes' Rule**

$$p(x|y) = \frac{p(y|x)p_X(x)}{p_Y(y)}$$

  - **Marginal density function: Integrate without it**

$$p_X(x) = \int p(x,y)dy$$

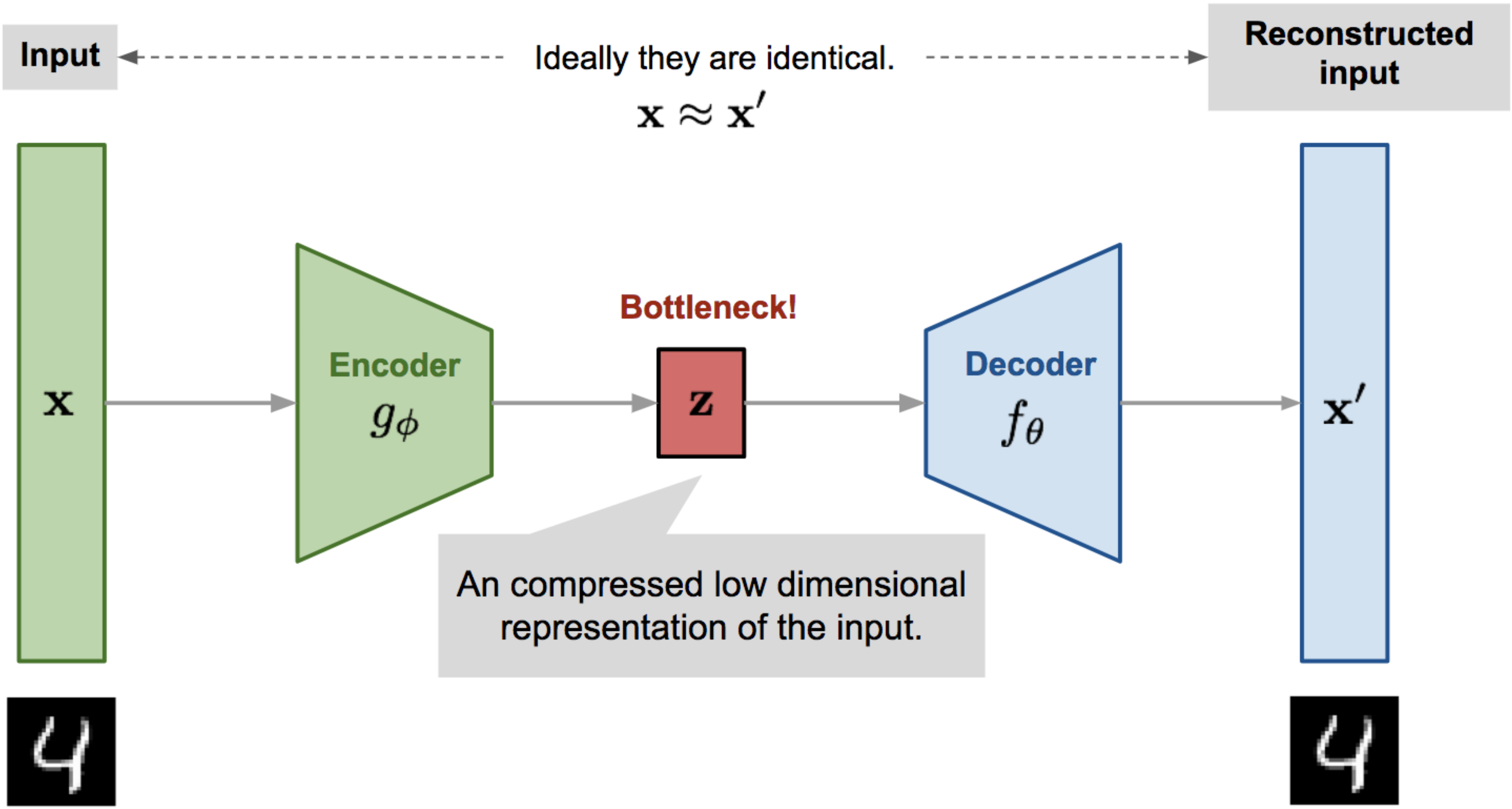
Introduction

Probabilistic generative models

- Probabilistic generative model learns a distribution p_θ from X1, X2, … ~ p_{true}
- We can generate new samples X ~ p_θ

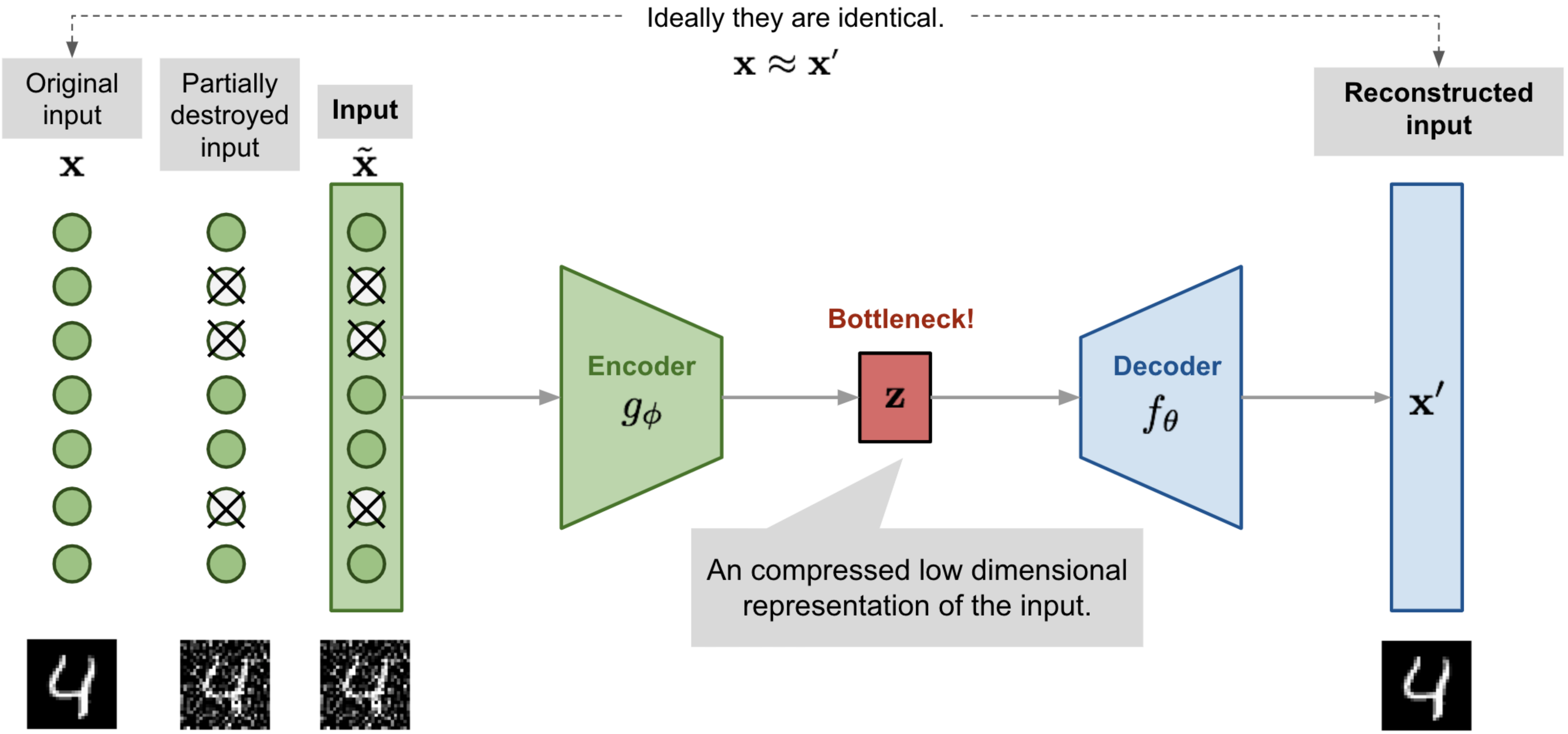
- Ex) Flow models, VAEs, GANs

Autoencoder



Source: https://lilianweng.github.io/posts/2018-08-12-vae/

Autoencoder



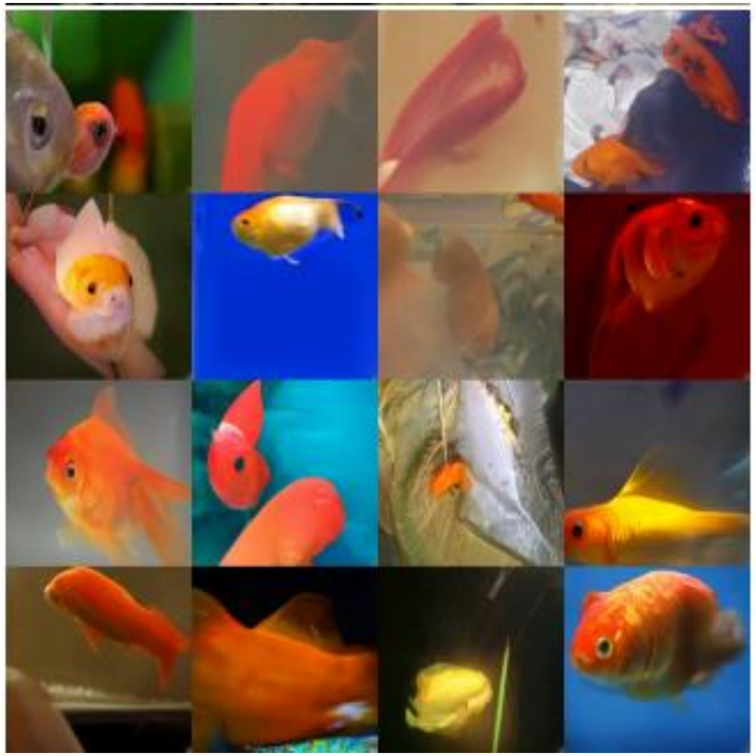
Source: https://lilianweng.github.io/posts/2018-08-12-vae/

What is VAE?

- **VAE(Variational Autoencoder)**
  - Probabilistic Generative model

    - Probabilistic generative model learns distribution from data. we can sample from distribution.
  - Sampling
  - Autoencoder

    - Compressed low dimensional representation.
  - Denoising data



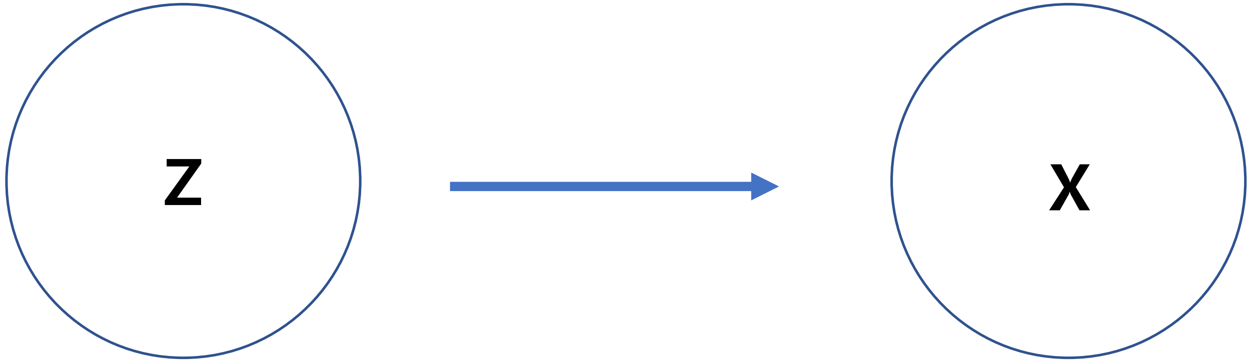
[1] VQ-VAE 2 samples

[1] A. Razavi et al., Generating Diverse High-Fidelity Images with VQ-VAE-2, NeurIPS 2019

Main idea of VAE

Main idea 1: Latent Variable Model

- Z is latent variable (representing essential structure)
- X is observable variable (real data)
- VAE's model is latent variable model with a NN that generates X given Z



Z ~ $p_Z(z)$

X ~ $p_\theta(x|z)$

Main idea 1: Latent Variable Model

- **Training via MLE(Maximum Likelihood Estimation)**
  - Density function(likelihood) for observed data X_i

$$p_\theta(X_i) = \int_z p_Z(z) p_\theta(X_i|z) dz = E_{Z\sim p_z}[p_\theta(X_i|Z)]$$

  - MLE

$$\underset{\theta\in\Theta}{\text{maximize}} \sum_{i=1}^{N} log p_\theta(X_i) = \underset{\theta\in\Theta}{\text{maximize}} \sum_{i=1}^{N} log E_{Z\sim p_z}[p_\theta(X_i|Z)]$$

Problem scenario

- **Intractability of likelihood where true posterior density($p_\theta(z|x)$) is intractable**

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

Main idea 2: Recognition model with IS

- We can approximate intractable objective with a single sample of Z.

$$\sum_{i=1}^{N} logE_{Z\sim p_z}[p_\theta(X_i|Z)] \approx \sum_{i=1}^{N} logp_\theta(X_i|Z_i)\,, Z_i \sim p_Z$$

- By using importance sampling, we can improve accuracy of approximation

$$\sum_{i=1}^{N} logE_{Z\sim p_z}[p_\theta(X_i|Z)] \approx \sum_{i=1}^{N} log\frac{p_\theta(X_i|Z_i)p_Z(Z_i)}{q_i(Z_i)}, Z_i \sim q_i$$

- Calculate mean value of K samples (Importance Weighted AutoEncoder, ICLR 2016)

Importance Sampling

- **Monte Carlo estimation**
  - Using estimate Expectation value of function.

$$I = E_{X \sim f}[\phi(X)] = \int \phi(x) f(x) dx$$

$$\hat{I}_N = \frac{1}{N} \sum_{i=1}^{N} \phi(X_i)$$

$$Var_{X \sim f}(\hat{I}_N) = \sum_{i=1}^{N} Var_{X_i \sim f}(\frac{\phi(X_i)}{N}) = \frac{1}{N} Var_{X \sim f}(\phi(X))$$

Importance Sampling

- **Importance Sampling**
  - Cosider X as a different distribution

$$I = \int \phi(x)f(x)dx = \int \frac{\phi(x)f(x)}{g(x)}g(x)dx = E_{X\sim g}[\frac{\phi(X)f(X)}{g(X)}]$$

$$\hat{I}_N = \frac{1}{N}\sum_{i=1}^{N}\phi(X_i)\frac{f(X_i)}{g(X_i)}$$

$$Var_{X\sim g}(\hat{I}_N) = \sum_{i=1}^{N} Var_{X_i\sim f}(\frac{\phi(X_i)f(X_i)}{Ng(X_i)}) = \frac{1}{N}Var_{X\sim g}(\frac{\phi(X)f(X)}{g(X)}))$$

Importance Sampling

- **Optimal sampling distribution**

$$g(x) = \frac{\phi(x)f(x)}{I}$$

-> $$Var_{X\sim g}\left(\frac{\phi(X)f(X)}{g(X)}\right) = Var_{X\sim g}(I) = 0$$

- **We need to solve optimization problem**

$$\underset{\theta\in\Theta}{\text{minimize}} D_{KL}(g_\theta || \frac{\phi f}{I})$$

Main idea 2: Recognition model with IS

- We can approximate intractable objective with a single sample of Z.

$$\sum_{i=1}^{N} log E_{Z\sim p_z}[p_\theta(X_i|Z)] \approx \sum_{i=1}^{N} log p_\theta(X_i|Z_i)\,,Z_i \sim p_Z$$

- By using importance sampling, we can improve accuracy of approximation

$$\sum_{i=1}^{N} log E_{Z\sim p_z}[p_\theta(X_i|Z)] \approx \sum_{i=1}^{N} log \frac{p_\theta(X_i|Z_i)p_Z(Z_i)}{q_i(Z_i)}\,,Z_i \sim q_i$$

- Calculate mean value of K samples (Importance Weighted AutoEncoder, ICLR 2016)

Main idea 2: Recognition model with IS

- We can't directly use q, so we will use parameterized distribution(Recognition model)

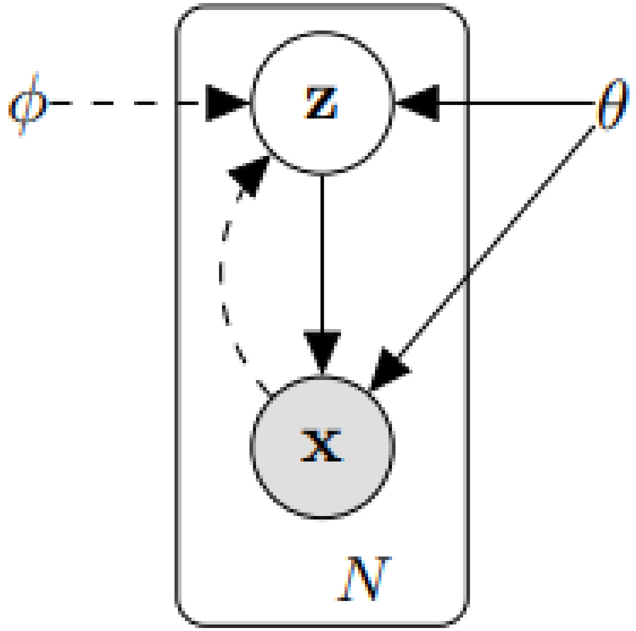
$$q_\phi(z|X_i) \approx q_i(z) = p_\theta(z|X_i)$$

- Why not $q_1(z), q_2(z), ..., q_N(z)$??
- **optimization problem**

$$\underset{\phi\in\Phi}{\text{minimize}} D_{KL}(q_\phi(\cdot|X_i))||p_\theta(\cdot|X_i))$$

Graphical model of VAE

- $q_\phi(z|x)$, **Variational approximation to the intractable posterior**



[1] graphical model of VAE

[1] Kingma et al., Auto-Encoding Variational Bayes, ICLR, 2014

Training VAE

Variational lower bound

- Encoder and Decoder have same objective function.

$$
\begin{aligned}
\log p_{\boldsymbol{\theta}}(\mathbf{x}) &= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x})\right] && (2.5)\\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] && (2.6)\\
&= \mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right] && (2.7)\\
&= \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z})}{q_\phi(\mathbf{z}|\mathbf{x})}\right]\right]}_{\substack{=\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x})\\ \text{(ELBO)}}} + \underbrace{\mathbb{E}_{q_\phi(\mathbf{z}|\mathbf{x})}\left[\log\left[\frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x})}\right]\right]}_{=D_{KL}(q_\phi(\mathbf{z}|\mathbf{x})||p_{\boldsymbol{\theta}}(\mathbf{z}|\mathbf{x}))} && (2.8)
\end{aligned}
$$

[1]

[1] Kingma et al., An Introduction to Variational Autoencoders, 2019

Computational flow

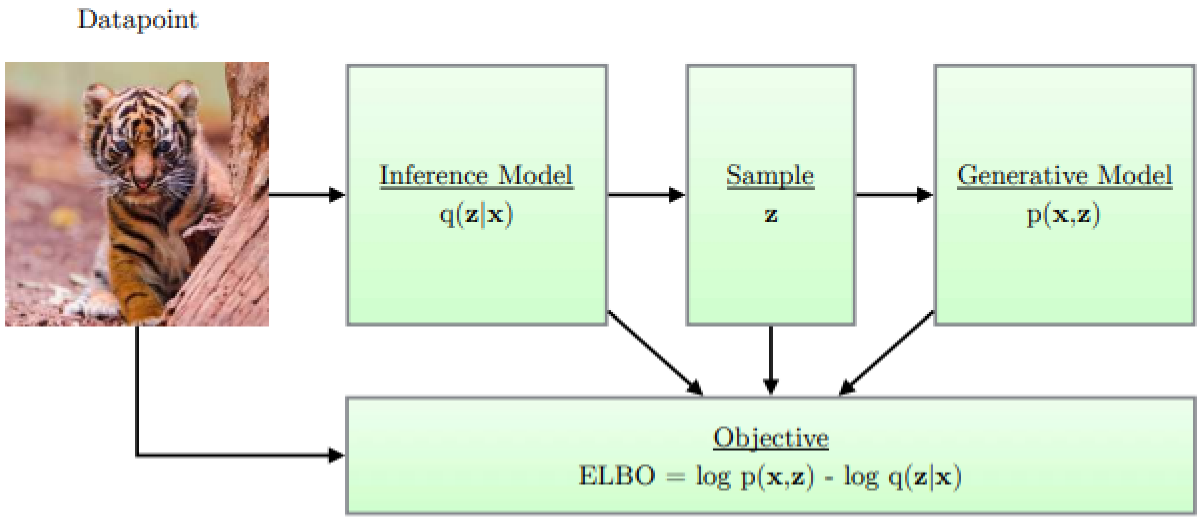
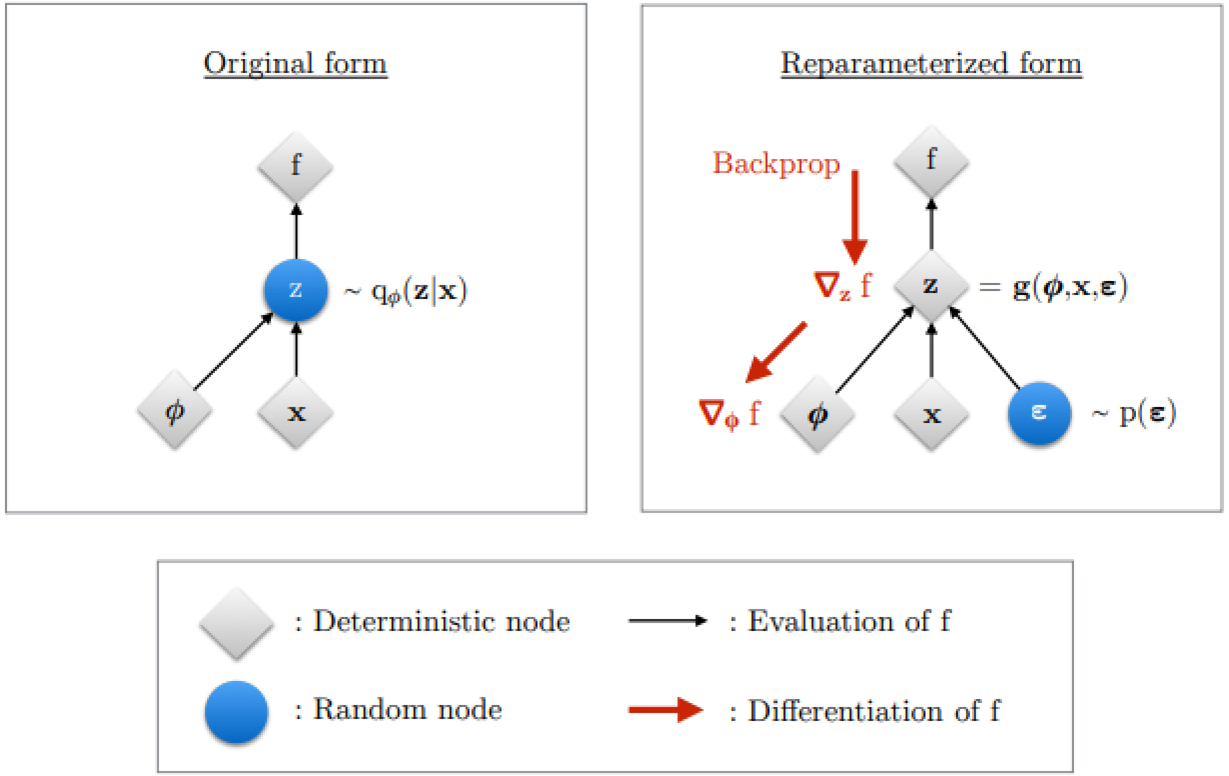


Figure 2.2: Simple schematic of computational flow in a variational autoencoder.

[1]

[1] Kingma et al., An Introduction to Variational Autoencoders, 2019

Reparameterization Trick



[1]

Reparameterization Trick

$$\mathcal{L}_{\boldsymbol{\theta},\boldsymbol{\phi}}(\mathbf{x}) = \mathbb{E}_{q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right] \tag{2.25}$$

$$= \mathbb{E}_{p(\boldsymbol{\epsilon})}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\boldsymbol{\phi}}(\mathbf{z}|\mathbf{x})\right] \tag{2.26}$$

where $\mathbf{z} = g(\boldsymbol{\epsilon}, \boldsymbol{\phi}, \mathbf{x})$.

[1]

[1] Kingma et al., An Introduction to Variational Autoencoders, 2019

Reparameterization Trick

$$
\begin{aligned}
\mathbb{E}_{p(\epsilon)}\left[\nabla_{\boldsymbol{\theta},\phi}\tilde{\mathcal{L}}_{\boldsymbol{\theta},\phi}(\mathbf{x};\boldsymbol{\epsilon})\right] &= \mathbb{E}_{p(\epsilon)}\left[\nabla_{\boldsymbol{\theta},\phi}(\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x}))\right] && (2.30)\\
&= \nabla_{\boldsymbol{\theta},\phi}(\mathbb{E}_{p(\epsilon)}\left[\log p_{\boldsymbol{\theta}}(\mathbf{x},\mathbf{z}) - \log q_{\phi}(\mathbf{z}|\mathbf{x})\right]) && (2.31)\\
&= \nabla_{\boldsymbol{\theta},\phi}\mathcal{L}_{\boldsymbol{\theta},\phi}(\mathbf{x}) && (2.32)
\end{aligned}
$$

[1]

[1] Kingma et al., An Introduction to Variational Autoencoders, 2019

Reparameterization Trick

- **Let approximate posterior is Gaussian**

$$E_{Z\sim\mathcal{N}(\mu,\sigma^2)}[f(z)] = E_{\epsilon\sim\mathcal{N}(0,1)}[f(\mu+\sigma\epsilon)] \approx \frac{1}{L}\sum_{l=1}^{L} f(\mu+\sigma\epsilon_l)$$

VAE standard instance

Standard VAE setup

- **Latent variable :**

$$p_Z \sim \mathcal{N}(0, I)$$

- **Variational approximate posterior(encoder) :**

$$q_\phi(z|x) \sim \mathcal{N}(\mu_\phi(x), \textstyle\sum_\phi(x))$$

  - With diagonal variance

- **Conditional distribution(decoder) :**

$$p_\theta(x|z) \sim \mathcal{N}(f_\theta(z), \sigma^2 I)$$

- $\mu_\phi(x), \sum_\phi(x), f_\theta(z)$ **are deterministic NN**

Standard Training Objective

- **Objective Function: (maximize)**

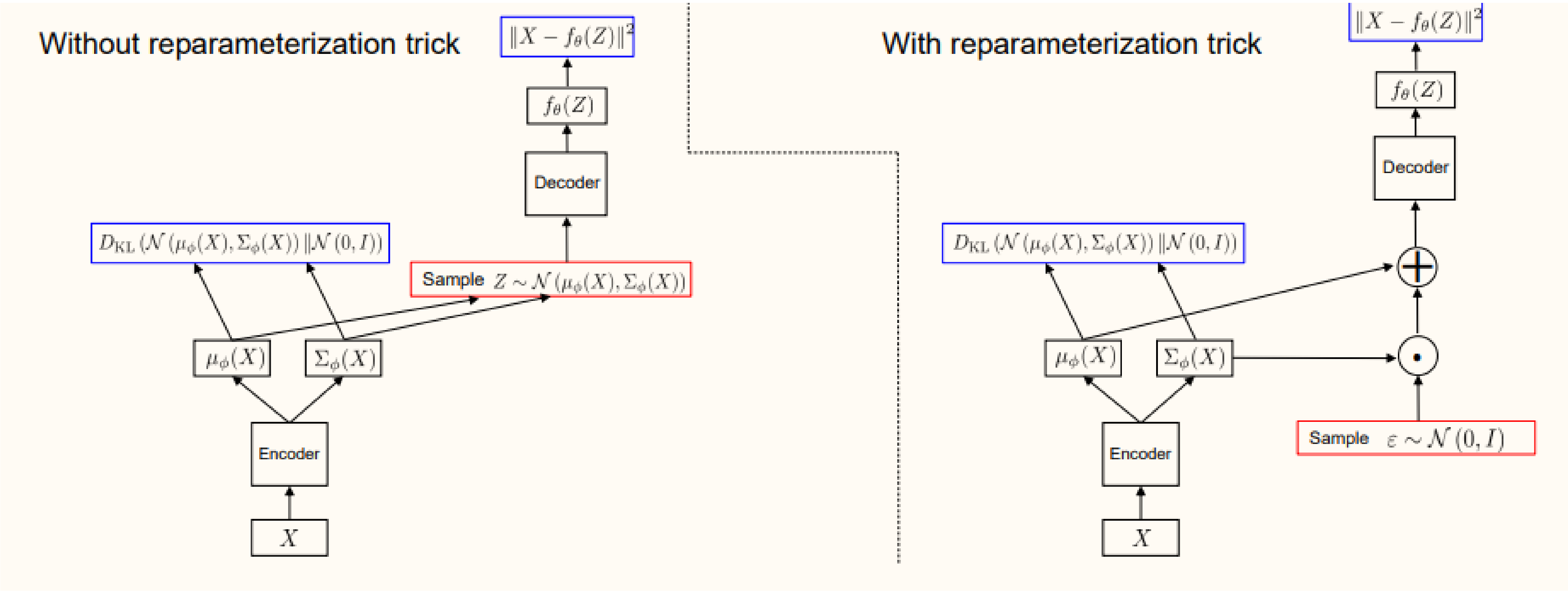
$$\sum_{i=1}^{N} VLB_{\theta,\phi}(X_i) = \sum_{i=1}^{N} E_{Z\sim q_\phi(z|X_i)}[log p_\theta(X_i|Z)] - D_{KL}(q_\phi(\cdot|X_i)||p_Z(\cdot))$$

$$= -\frac{1}{2\sigma^2}\sum_{i=1}^{N} E_{Z\sim\mathcal{N}(\mu_\phi(X_i),\sum_\phi(X_i))}||X_i - f_\theta(Z)||^2 - D_{KL}(q_\phi(\cdot|X_i)||p_Z(\cdot))$$

- **Using reparameterization trick**

$$= \underbrace{-\frac{1}{2\sigma^2}\sum_{i=1}^{N} E_{\epsilon\sim\mathcal{N}(0,I)}||X_i - f_\theta(\mu_\phi(X_i) + \sum_\phi^{\frac{1}{2}}(X_i)\epsilon)||^2}_{\text{Reconstruction loss}} \underbrace{- D_{KL}(q_\phi(\cdot|X_i)||p_Z(\cdot))}_{\text{Regularization}}$$

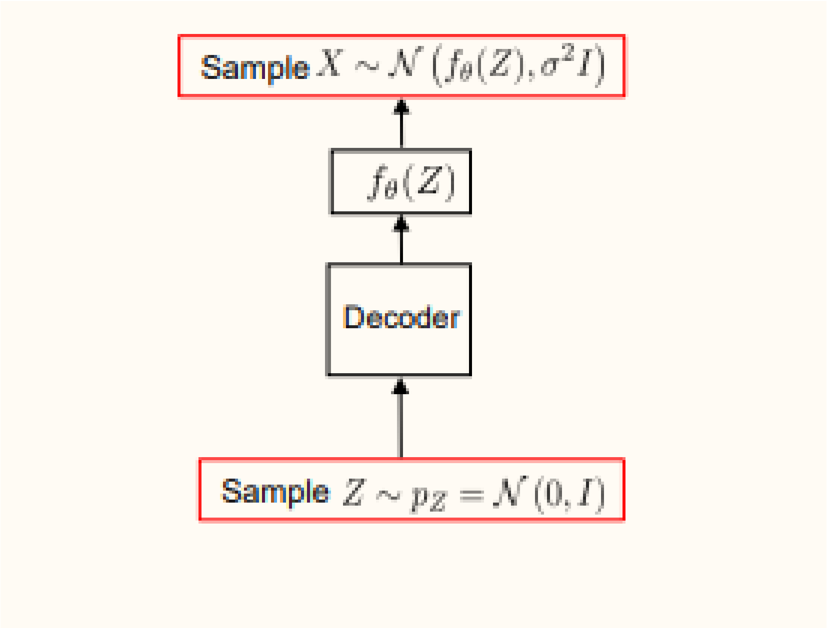
VAE standard instance architecture: Training



[1]

[1] http://www.math.snu.ac.kr/~ernestryu/courses/deep_learning.html

VAE standard instance architecture: Sampling



[1]

[1] http://www.math.snu.ac.kr/~ernestryu/courses/deep_learning.html

Pytorch Code

Experiments

Experiments

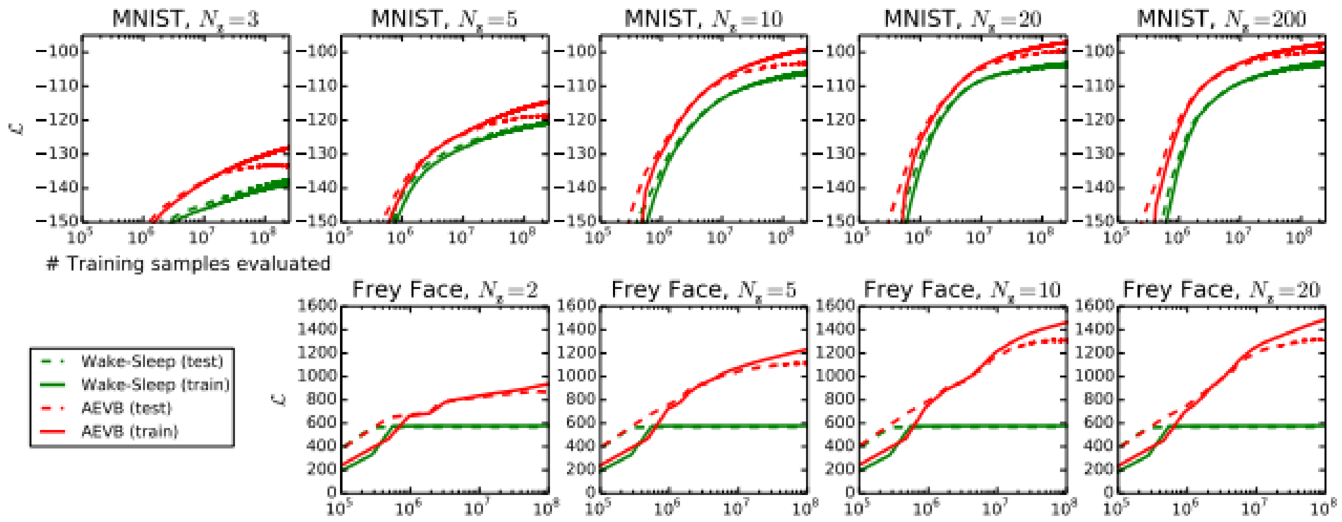


Figure 2: Comparison of our AEVB method to the wake-sleep algorithm, in terms of optimizing the lower bound, for different dimensionality of latent space ($N_\mathbf{z}$). Our method converged considerably faster and reached a better solution in all experiments. Interestingly enough, more latent variables does not result in more overfitting, which is explained by the regularizing effect of the lower bound. Vertical axis: the estimated average variational lower bound per datapoint. The estimator variance was small (< 1) and omitted. Horizontal axis: amount of training points evaluated. Computation took around 20-40 minutes per million training samples with a Intel Xeon CPU running at an effective 40 GFLOPS.

[1]

[1] Kingma et al., Auto-Encoding Variational Bayes, ICLR, 2014

Reference

- http://www.math.snu.ac.kr/~ernestryu/courses/deep_learning.html 오픈소스 강의자료

- Kingma et al., Auto-Encoding Variational Bayes, ICLR, 2014

- Kingma et al., An Introduction to Variational Autoencoders, 2019