Variational Autoencoder (VAE)

Cuong Dang, Nam Nguyen, Thanh Bui[1]

[1]CS Department
Ho Chi Minh City University Information of Technology (UIT)

Overview

I. Introduction

Introduction to Variational Autoencoders (VAE)

- **Autoencoders (AE)**:
  - Autoencoders are deep learning models used for learning representations of data in a lower-dimensional latent space.
  - They consist of two parts: the *Encoder* (which encodes data) and the *Decoder* (which reconstructs data).
  - However, AE models are limited in their ability to generate new data in a smooth and natural way.

Introduction to Variational Autoencoders (VAE)

- **Autoencoders (AE)**:
  - Autoencoders are deep learning models used for learning representations of data in a lower-dimensional latent space.
  - They consist of two parts: the *Encoder* (which encodes data) and the *Decoder* (which reconstructs data).
  - However, AE models are limited in their ability to generate new data in a smooth and natural way.
- **Variational Autoencoders (VAE)**:
  - VAE is an extension of Autoencoders where the latent space is modeled as a probabilistic distribution.
  - The goal of VAE is to learn a probability distribution over the latent space, enabling the generation of new data samples.
  - VAE uses *Variational Inference* methods to approximate the posterior distribution.

Differences Between Autoencoder (AE) and Variational Autoencoder (VAE)

- **Latent Space Representation:**
  - **AE:** Encodes data into a fixed latent vector.
  - **VAE:** Models the latent space as a probabilistic distribution (e.g., Gaussian distribution).

Differences Between Autoencoder (AE) and Variational Autoencoder (VAE)

- **Latent Space Representation:**
  - **AE:** Encodes data into a fixed latent vector.
  - **VAE:** Models the latent space as a probabilistic distribution (e.g., Gaussian distribution).
- **Goal:**
  - **AE:** Focuses on reconstructing input data as accurately as possible.
  - **VAE:** Learns a distribution over the latent space for generating new data samples.

Differences Between Autoencoder (AE) and Variational Autoencoder (VAE)

- **Mathematical Framework:**
  - **AE:** Uses a standard loss function (e.g., Mean Squared Error) for reconstruction.
  - **VAE:** Combines a reconstruction loss and a regularization term (Kullback-Leibler divergence).

Differences Between Autoencoder (AE) and Variational Autoencoder (VAE)

- **Mathematical Framework:**
  - **AE:** Uses a standard loss function (e.g., Mean Squared Error) for reconstruction.
  - **VAE:** Combines a reconstruction loss and a regularization term (Kullback-Leibler divergence).
- **Data Generation:**
  - **AE:** Limited ability to generate new data.
  - **VAE:** Generates smooth and realistic data by sampling from the learned distribution.

Why Choose Variational Autoencoders (VAE)?

- **Dimensionality Reduction:**
  - One of the main advantages of autoencoders is their ability to reduce the dimensionality of data.
  - They transform the data into a smaller latent representation while retaining the important information.

Why Choose Variational Autoencoders (VAE)?

- **Dimensionality Reduction:**
  - One of the main advantages of autoencoders is their ability to reduce the dimensionality of data.
  - They transform the data into a smaller latent representation while retaining the important information.
- **Feature Learning for Classification:**
  - Autoencoders can serve as a feature learning tool.
  - The latent space representation can be used as input for downstream classification tasks.

II. Structure of VAE

Basic Structure of VAE

- **Encoder:**
  - Takes the input data x and encodes it into a probabilistic representation in the latent space z.
- **Latent Space:**
  - A space where data is represented probabilistically, capturing the underlying structure of the data.
- **Decoder:**
  - Decodes the latent variable z back into the original data x, reconstructing the input.
- **Objective:**
  - The goal is to learn to reconstruct the input data and optimize the distribution in the latent space.

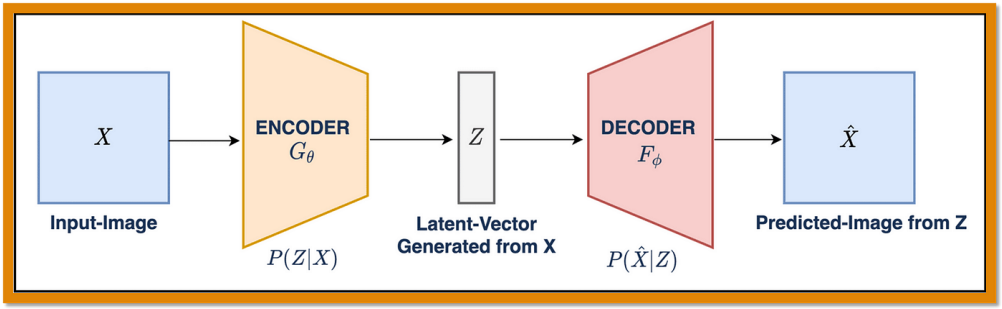


Figure 1: Structure of VAE

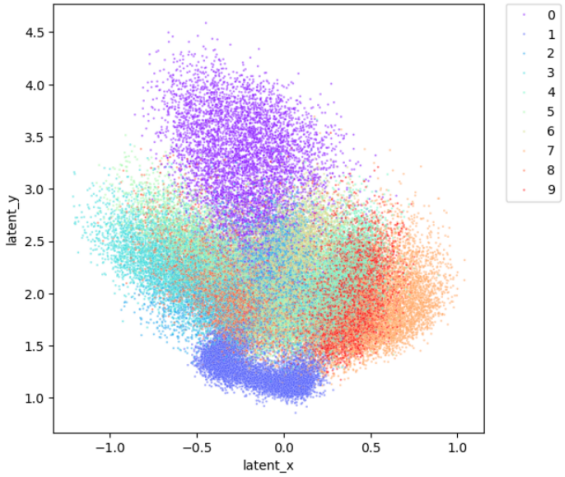


Figure 2: Latent space

III. Mathematical Symbols in VAE

- $p(\mathbf{x})$: The true distribution of $\mathbf{x}$. The goal of diffusion models is to sample from this distribution, which is typically unknown.
- $p(z)$: The distribution of the latent variable, usually a zero-mean unit-variance Gaussian: $\mathcal{N}(0, \mathbf{I})$. This simplifies processing, as any distribution can be generated by mapping a Gaussian through a complex function.
- $p(z|\mathbf{x})$: The conditional distribution in the *encoder* — the likelihood of z given $\mathbf{x}$. This distribution is not directly accessible.
- $p(\mathbf{x}|z)$: The conditional distribution in the *decoder* — the probability of $\mathbf{x}$ given z. This distribution is also not directly accessible.

- $q_\phi(z|\mathbf{x})$: The approximate distribution for $p(z|\mathbf{x})$, parameterized by a neural network. Define:

$$(\mu, \sigma^2) = \text{EncoderNetwork}_\phi(\mathbf{x}),$$

$$q_\phi(z|\mathbf{x}) = \mathcal{N}(z \mid \mu, \text{diag}(\sigma^2)).$$

- $q_\phi(z|\mathbf{x})$: The approximate distribution for $p(z|\mathbf{x})$, parameterized by a neural network. Define:

$$(\mu, \sigma^2) = \text{EncoderNetwork}_\phi(\mathbf{x}),$$

$$q_\phi(z|\mathbf{x}) = \mathcal{N}(z \mid \mu, \text{diag}(\sigma^2)).$$

- $p_\theta(\mathbf{x}|z)$: The approximate distribution for $p(\mathbf{x}|z)$, parameterized by a neural network. Define:

$$f_\theta(z) = \text{DecoderNetwork}_\theta(z),$$

$$p_\theta(\mathbf{x}|z) = \mathcal{N}(\mathbf{x} \mid f_\theta(z), \sigma^2_{\text{dec}}\mathbf{I}).$$

The relationship between the input x and the latent z, as well as the conditional distributions, are summarized below

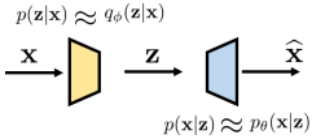


Figure 3: $\mathbf{x}$ and z are connected by $p(z|\mathbf{x})$ and $p(\mathbf{x}|z)$ in a VAE.

IV. Build construct in VAE

Encoding and Decoding

Suppose that we have a random variable $x \in \mathbb{R}^d$ and a latent variable $z \in \mathbb{R}^d$ such that:

$$x \sim p(x) = \mathcal{N}(x \mid \mu, \sigma^2 \mathbf{I}),$$

$$z \sim p(z) = \mathcal{N}(z \mid 0, \mathbf{I}).$$

We want to construct a Variational Autoencoder. By this, we mean that we want to build two mappings: the encoder Encoder$(\cdot)$ and the decoder Decoder$(\cdot)$. The encoder will take a sample x and map it to the latent variable z, whereas the decoder will take the latent variable z and map it to the generated variable $\hat{x}$.

If we know $p(x)$

- If we know the distribution $p(x)$, then there is a simple solution:

$$z = \frac{x-\mu}{\sigma}, \quad \widehat{x} = \mu + \sigma z.$$

- In this case, the true distributions are expressed using delta functions:

$$p(x \mid z) = \delta\big(x - (\sigma z + \mu)\big),$$

$$p(z \mid x) = \delta\big(z - \frac{x-\mu}{\sigma}\big).$$

Dirac Delta Function:

- The Dirac delta function is often represented as:

$$\delta_a(x) = \frac{1}{|a|\sqrt{\pi}} e^{-\left(\frac{x}{a}\right)^2}.$$

- For computational purposes, it can be approximated by a Gaussian distribution.

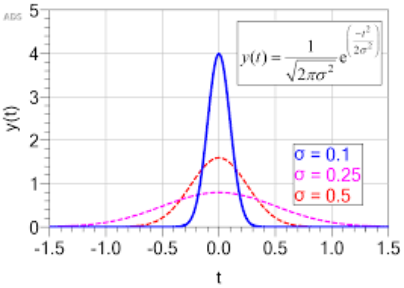


Figure 4: Dirac Delta Function

If we do not know $p(x)$

Let's first define the encoder. Our encoder in this example takes the input x and generates a pair of parameters $\widehat{\mu}(x)$ and $\widehat{\sigma}^2(x)$, denoting the parameters of a Gaussian. Then, we define $q_\phi(z \mid x)$ as a Gaussian:

$$(\widehat{\mu}(x), \widehat{\sigma}^2(x)) = \text{Encoder}_\phi(x),$$

$$q_\phi(z \mid x) = \mathcal{N}(z \mid \widehat{\mu}(x), \widehat{\sigma}^2(x) I).$$

For the purpose of discussion, we assume that $\widehat{\mu}$ is an affine function of x such that: $\widehat{\mu}(x) = ax + b$ for some parameters a and b. Similarly, we assume that:

$$\widehat{\sigma}^2(x) = t^2,$$

for some scalar t. This will give us:

$$q_\phi(z \mid x) = \mathcal{N}(z \mid ax + b, t^2 I).$$

If we do not know $p(x)$

For the decoder, we deploy a similar structure by considering:

$$(\mu_e(z), \sigma_e^2(z)) = \text{Decoder}_\theta(z),$$

$$p_\theta(x \mid z) = \mathcal{N}(x \mid \mu_e(z), \sigma_e^2(z)I).$$

Again, for the purpose of discussion, we assume that μ_e is affine so that: $\mu_e(z) = cz + v$ for some parameters c and v, and:

$$\sigma_e^2(z) = s^2,$$

for some scalar s. Therefore, $p_\theta(x \mid z)$ takes the form of:

$$p_\theta(x \mid z) = \mathcal{N}(x \mid cz + v, s^2 I).$$

V. Evidence Lower Bound

Why?

Variational Autoencoders (VAEs) are designed to model observed data x through a latent variable z. The goal is to maximize the marginal likelihood of the data $p(x)$:

$$p(x) = \int p(x,z)\,dz,$$

where $p(x,z) = p(x \mid z)p(z)$.
Although using $p(x \mid z)$ and $p(z)$ simplifies the complex distribution $p(x,z)$, the integral is difficult to compute, especially when:

- The latent space z is very large or continuous.
- $p(x \mid z)$ and $p(z)$ are nonlinear or complex.

When the space of z is continuous, computing the integral over the entire latent space becomes impractical due to:

- The computational complexity.
- The high dimensionality of z, which makes the processing infeasible.

Define ELBO

Theorem 1.1. Decomposition of Log-Likelihood. The log-likelihood $\log p(x)$ can be decomposed as

$$\log p(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(x,z)}{q_\phi(z|x)}\right] + D_{\mathrm{KL}}(q_\phi(z|x)\|p(z|x)).$$

Proof

The trick is to use our magical proxy $q_\phi(z|x)$ to poke around $p(x)$ and derive the bound:

$$\log p(x) = \log p(x) \times \underbrace{\int q_\phi(z|x)\, dz}_{=1}$$

$$= \int \underbrace{\log p(x)}_{\text{constant w.r.t. } z} \times \underbrace{q_\phi(z|x)}_{\text{distribution in } z} dz \qquad \text{(move } \log p(x) \text{ into the integral)}$$

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

Define the Kullback-Leibler Divergence between two continuous distributions:

$$D_{\mathrm{KL}}(P \parallel Q) = \int_{-\infty}^{\infty} p(x) \log\left(\frac{p(x)}{q(x)}\right) dx$$
$$= \mathbb{E}_{p(x)}\left[\log \frac{p(x)}{q(x)}\right].$$

An example of KL

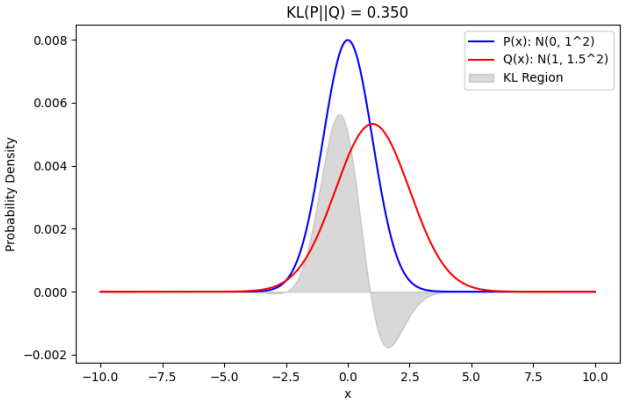


Figure 5: An example of KL

Deriving ELBO with Bayes' Theorem

Using Bayes' theorem: $p(x,z) = p(z|x)p(x)$

$$\begin{aligned}
\mathbb{E}_{q_\phi(z|x)}[\log p(x)] &= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(x,z)}{p(z|x)}\right] \text{(Bayes' theorem)} \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(x,z)}{p(z|x)} \times \frac{q_\phi(z|x)}{q_\phi(z|x)}\right] \text{(Multiply and divide by } q_\phi(z|x)\text{)} \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(x,z)}{q_\phi(z|x)}\right] + \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{q_\phi(z|x)}{p(z|x)}\right] \qquad (8) \\
&= \text{ELBO}(x) + D_{\text{KL}}(q_\phi(z|x) \| p(z|x)).
\end{aligned}$$

Decompose ELBO

Theorem 1.2

We can decompose ELBO as:

$$\text{ELBO}(x) = \mathbb{E}_{q_\phi(z|x)}\Big[\underbrace{\log p_\theta(x|z)}_{\text{a Gaussian (how good of decoder)}}\Big] - \underbrace{D_{\text{KL}}(q_\phi(z|x)\|p(z))}_{\text{a Gaussian (how good of encoder)}}.$$

Proof

$$
\begin{aligned}
\mathrm{ELBO}(x) &\overset{\text{def}}{=} \mathbb{E}_{q_\phi(z|x)}\big[\log \frac{p(x,z)}{q_\phi(z|x)}\big] && \text{(definition)} \\
&= \mathbb{E}_{q_\phi(z|x)}\big[\log \frac{p(x|z)p(z)}{q_\phi(z|x)}\big] && \text{(since } p(x,z) = p(x|z)p(z)\text{)} \\
&= \mathbb{E}_{q_\phi(z|x)}\big[\log p(x|z)\big] + \mathbb{E}_{q_\phi(z|x)}\big[\log \frac{p(z)}{q_\phi(z|x)}\big] && \text{(split expectation)} \\
&= \mathbb{E}_{q_\phi(z|x)}\big[\log p_\theta(x|z)\big] - D_{(\mathrm{KL})}(q_\phi(z|x)\|p(z)), && \text{(definition of KL)}
\end{aligned}
$$

where we replaced the inaccessible $p(x|z)$ by its proxy $p_\theta(x|z)$.

Key Objectives in Variational Autoencoders

Reconstruction:

- The first term focuses on the decoder. We want the decoder to produce a good reconstruction of x when we feed a latent z into it.
- Our goal is to maximize $p_\theta(x|z)$, which is similar to maximum likelihood, where we optimize the model parameters to maximize the likelihood of observing the input x.

Key Objectives in Variational Autoencoders

Prior Matching:

- The second term is the KL divergence for the encoder. We want the encoder to map x into a latent vector z such that z follows a chosen distribution:

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

- More generally, we define $p(z)$ as the target distribution. Since KL divergence measures the distance between two distributions, minimizing it ensures the latent space $q_\phi(z|x)$ becomes similar to $p(z)$.
- **Note**: A negative sign is used in the objective so that minimizing KL divergence aligns the distributions.

VI. Optimization in VAE

Definition

The optimization objective of the VAE is to maximize the ELBO:

$$(\phi, \theta) = \arg\max_{\phi,\theta} \sum_{x \in \mathcal{X}} \text{ELBO}(x),$$

where $\mathcal{X} = \{x^{(\ell)} \mid \ell = 1, \ldots, L\}$ is the training dataset.

To be more precisely, we can show this objective function:

$$\begin{aligned} \nabla_{\theta,\phi} \text{ELBO}(x) &= \nabla_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)} \left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)} \right] \\ &= \nabla_{\theta,\phi} \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x,z) - \log q_\phi(z|x) \right]. \end{aligned}$$

Gradient with parameter θ

Let's first look at θ. We can show that:

$$\begin{aligned}
\nabla_\theta \text{ELBO}(x) &= \nabla_\theta \mathbb{E}_{q_\phi(z|x)} \left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] \\
&= \nabla_\theta \int \left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] q_\phi(z|x)\, dz \\
&= \int \nabla_\theta \left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] q_\phi(z|x)\, dz \\
&= \mathbb{E}_{q_\phi(z|x)} \left[\nabla_\theta \log p_\theta(x,z)\right].
\end{aligned}$$

Using Monte Carlo approximation, we have:

$$\nabla_\theta \text{ELBO}(x) \approx \frac{1}{L} \sum_{\ell=1}^{L} \nabla_\theta \log p_\theta(x, z^{(\ell)}),$$

where $z^{(\ell)} \sim q_\phi(z|x)$.

Gradient with parameter ϕ

The gradient with respect to ϕ is more difficult. We can show that:

$$\begin{aligned}
\nabla_\phi \text{ELBO}(x) &= \nabla_\phi \mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] \\
&= \nabla_\phi \int \left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] q_\phi(z|x)\, dz \\
&= \int \nabla_\phi \left[\log p_\theta(x,z) - \log q_\phi(z|x)\right] q_\phi(z|x)\, dz \\
&\neq \int \left[\nabla_\phi \left(\log p_\theta(x,z) - \log q_\phi(z|x)\right) q_\phi(z|x)\right] dz \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\phi \left(\log p_\theta(x,z) - \log q_\phi(z|x)\right)\right] \\
&= \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\phi \left(-\log q_\phi(z|x)\right)\right] \\
&\approx \frac{1}{L}\sum_{\ell=1}^{L} \nabla_\phi \left(-\log q_\phi(z^{(\ell)}|x)\right), \text{where } z^{(\ell)} \sim q_\phi(z|x).
\end{aligned}$$

Reparameterization Trick

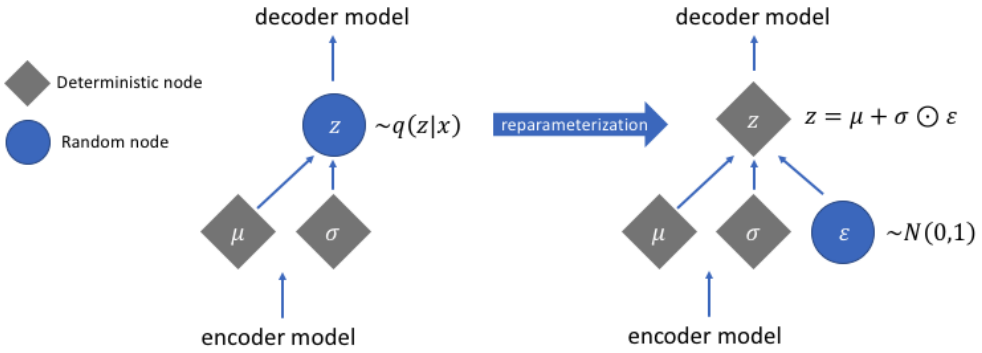


Figure 6: Comparison between Original form and Reparameterized form

Reparameterization Trick

As we can see, $z^{(\ell)} \sim q_\phi(z|x)$ represents a random node, symbolizing the stochastic sampling of the value z.

However, $q_\phi(z|x)$ is parameterized by a neural network (the Encoder network), which makes it difficult to compute a closed-form solution.

Thus, we need to adopt a new approach to address this problem: the **Reparameterization Trick**.

Define

Suppose $z \sim q_\phi(z|x) \overset{\text{def}}{=} \mathcal{N}(z|\mu, \text{diag}(\sigma^2))$. We can define

$$z = g(\epsilon, \phi, x) \overset{\text{def}}{=} \epsilon \odot \sigma + \mu, \tag{1}$$

where $\epsilon \sim \mathcal{N}(0, I)$ and "$\odot$" means elementwise multiplication. The parameter ϕ is $\phi = (\mu, \sigma^2)$. For this choice of the distribution, we can show that by letting $\epsilon = \frac{z-\mu}{\sigma}$:

$$q_\phi(z|x) \cdot \left|\det\left(\frac{\partial z}{\partial \epsilon}\right)\right| = \prod_{i=1}^{d} \frac{1}{\sqrt{2\pi\sigma_i^2}} \exp\left\{-\frac{(z_i - \mu_i)^2}{2\sigma_i^2}\right\} \cdot \prod_{i=1}^{d} \sigma_i$$

$$= \frac{1}{(2\pi)^{d/2}} \exp\left\{-\frac{\|\epsilon\|^2}{2}\right\} = \mathcal{N}(0, I) = p(\epsilon).$$

Jacobian of the Reparameterization Trick

$\frac{\partial z}{\partial \epsilon}$ is the Jacobian, we can define that:

$$\frac{\partial z}{\partial \epsilon} = \begin{bmatrix} \frac{\partial z_1}{\partial \epsilon_1} & \frac{\partial z_1}{\partial \epsilon_2} & \cdots & \frac{\partial z_1}{\partial \epsilon_d} \\ \frac{\partial z_2}{\partial \epsilon_1} & \frac{\partial z_2}{\partial \epsilon_2} & \cdots & \frac{\partial z_2}{\partial \epsilon_d} \\ \vdots & \vdots & \ddots & \vdots \\ \frac{\partial z_d}{\partial \epsilon_1} & \frac{\partial z_d}{\partial \epsilon_2} & \cdots & \frac{\partial z_d}{\partial \epsilon_d} \end{bmatrix}.$$

and expression in each component i is:

$$z_i = \sigma_i \epsilon_i + \mu_i$$

Gradient Computation with Reparameterization

We aim to compute the gradient $\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)]$ for some general function $f(z)$. (We will consider $f(z) = -\log q_\phi(z|x)$.)
For simplicity, we write $g(\epsilon)$ instead of $g(\epsilon, \phi, x)$, even though g has three inputs.

Using the change of variables, we have:

$$\mathbb{E}_{q_\phi(z|x)}[f(z)] = \int f(z) \cdot q_\phi(z|x)\, dz = \int f(g(\epsilon)) \cdot q_\phi(g(\epsilon)|x)\, dg(\epsilon), \quad \text{where } z = g(\epsilon).$$

Incorporating the Jacobian determinant:

$$\mathbb{E}_{q_\phi(z|x)}[f(z)] = \int f(g(\epsilon)) \cdot q_\phi(g(\epsilon)|x) \cdot \det\left(\frac{\partial g(\epsilon)}{\partial \epsilon}\right) d\epsilon.$$

Simplification:

$$\mathbb{E}_{q_\phi(z|x)}[f(z)] = \int f(z) \cdot p(\epsilon)\, d\epsilon = \mathbb{E}_{p(\epsilon)}[f(z)].$$

Gradient Computation for ϕ

To compute the gradient with respect to ϕ, we write:

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)] = \nabla_\phi \mathbb{E}_{p(\epsilon)}[f(z)] = \nabla_\phi \int f(z) \cdot p(\epsilon)\, d\epsilon.$$

Since $p(\epsilon)$ does not depend on ϕ, the gradient only acts on $f(z)$:

$$\nabla_\phi \mathbb{E}_{p(\epsilon)}[f(z)] = \int \nabla_\phi \{f(z) \cdot p(\epsilon)\}\, d\epsilon = \int \{\nabla_\phi f(z)\} \cdot p(\epsilon)\, d\epsilon.$$

Thus:

$$\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[f(z)] = \mathbb{E}_{p(\epsilon)}[\nabla_\phi f(z)].$$

Substitution for $f(z) = -\log q_\phi(z|x)$:

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

Monte Carlo Approximation:

$$\mathbb{E}_{p(\epsilon)}[-\nabla_\phi \log q_\phi(z|x)] \approx -\frac{1}{L}\sum_{\ell=1}^{L} \nabla_\phi \log q_\phi(z^{(\ell)}|x),$$

where $z^{(\ell)} = g(\epsilon^{(\ell)}, \phi, x)$.

Rewriting $q_\phi(z|x)$:

$$q_\phi(z|x) = p(\epsilon) : \det\left(\frac{\partial z}{\partial \epsilon}\right), \quad \log q_\phi(z|x) = \log p(\epsilon) - \log\det\left(\frac{\partial z}{\partial \epsilon}\right).$$

Gradient Simplification: Since $p(\epsilon)$ is independent of ϕ:

$$\nabla_\phi \log q_\phi(z|x) = \nabla_\phi \log\det\left(\frac{\partial z}{\partial \epsilon}\right).$$

Gradient Simplification for the Encoder

Thus:

$$\begin{aligned}\nabla_\phi \mathbb{E}_{q_\phi(z|x)}[-\log q_\phi(z|x)] &\approx \frac{1}{L}\sum_{\ell=1}^{L} \nabla_\phi \log\det\left(\frac{\partial z^{(\ell)}}{\partial \epsilon^{(\ell)}}\right) \\ &= \frac{1}{L}\sum_{\ell=1}^{L} \nabla_\phi \left[\sum_{i=1}^{d} \log \sigma_i\right] \\ &= \nabla_\phi \left[\sum_{i=1}^{d} \log \sigma_i\right] \\ &= \frac{1}{\sigma} \odot \nabla_\phi [\sigma_\phi(x)].\end{aligned}$$

Note: $\sigma_\phi(x)$ is the output of the encoder, which is parameterized by a neural network.

VII. Specific Structure of VAE

Encoder

$$(\mu, \sigma^2) = \text{EncoderNetwork}_\phi(x)$$

$$q_\phi(z|x) = \mathcal{N}(z|\mu, \sigma^2 I)$$

The parameters μ and σ are technically neural networks because they are outputs of $\text{EncoderNetwork}_\phi(\cdot)$.

We denote them as:

$$\mu = \mu_\phi(x) \quad \text{and} \quad \sigma^2 = \sigma^2_\phi(x).$$

Given the ℓ-th training sample $x^{(\ell)}$, the latent variable $z^{(\ell)}$ is sampled as:

$$z^{(\ell)} \sim \mathcal{N}\left(z \mid \mu_\phi(x^{(\ell)}), \sigma^2_\phi(x^{(\ell)}) I\right).$$

From the Gaussian, we draw a sample $z^{(\ell)}$:

$$z^{(\ell)} = \mu_\phi(x^{(\ell)}) + \sigma_\phi(x^{(\ell)})\epsilon, \quad \epsilon \sim \mathcal{N}(0, \mathbf{I}).$$

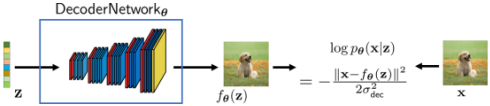


Figure 7: Implementation of a VAE encoder. The neural network takes x and estimates μ_ϕ and σ^2_ϕ.

Theorem 1.3: KL-Divergence of Two Gaussians

The KL divergence for two d-dimensional Gaussian distributions $\mathcal{N}(\mu_0, \Sigma_0)$ and $\mathcal{N}(\mu_1, \Sigma_1)$ is:

$$\begin{aligned} D_{\mathrm{KL}}\left(\mathcal{N}(\mu_0, \Sigma_0) \,\|\, \mathcal{N}(\mu_1, \Sigma_1)\right) = \frac{1}{2}\Big[& \operatorname{Tr}(\Sigma_1^{-1}\Sigma_0) - d \\ & + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0) \\ & + \log \frac{\det \Sigma_1}{\det \Sigma_0}\Big]. \end{aligned}$$

Substituting Specific Distributions:

$$\mu_0 = \mu_\phi(x), \quad \Sigma_0 = \sigma_\phi^2(x)\mathbf{I}, \quad \mu_1 = 0, \quad \Sigma_1 = \mathbf{I}.$$

Simplified KL Divergence:

$$D_{\mathrm{KL}}\left(q_\phi(z \mid x) \parallel p(z)\right) = \frac{1}{2}\left[\sigma_\phi^2(x)d - d + \|\mu_\phi(x)\|^2 - 2\log\sigma_\phi(x)\right].$$

Here, d is the dimension of the latent vector z.

Decoder

The decoder network, $\text{DecoderNetwork}_\theta(\cdot)$, maps a latent variable z to a generated image:

$$f_\theta(z) = \text{DecoderNetwork}_\theta(z).$$

The distribution $p_\theta(x \mid z)$ is:

$$p_\theta(x \mid z) = \mathcal{N}(x \mid f_\theta(z), \sigma_{\text{dec}}^2 I),$$

where σ_{dec} is a hyperparameter. Using the reparameterization trick:

$$\widehat{x} = f_\theta(z) + \sigma_{\text{dec}}\epsilon, \quad \epsilon \sim \mathcal{N}(0, I).$$

Log-Likelihood of $p_\theta(x \mid z)$

The log-likelihood can be written as:

$$\begin{aligned}
&\log p_\theta(x|z) = \log \mathcal{N}(x|f_\theta(z), \sigma^2_{\text{dec}} I) \\
&= \log \left(\frac{1}{\sqrt{(2\pi\sigma^2_{\text{dec}})^d}} \right) \exp\left(-\frac{\|x - f_\theta(z)\|^2}{2\sigma^2_{\text{dec}}} \right) \\
&= -\frac{\|x - f_\theta(z)\|^2}{2\sigma^2_{\text{dec}}} - \log\left((2\pi\sigma^2_{\text{dec}})^d \right)
\end{aligned}$$

The constant term $\log \sqrt{(2\pi\sigma^2_{\text{dec}})^d}$ is independent of θ and can be dropped.

ELBO with Monte Carlo Approximation

Using the reparameterization trick, the expectation is approximated as:

$$\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x \mid z)] = -\int \frac{\|x - f_\theta(z)\|^2}{2\sigma_{\text{dec}}^2} \cdot \mathcal{N}\left(z \mid \mu_\phi(x), \sigma_\phi(x)^2\right) dz$$
$$\approx -\frac{1}{M} \sum_{m=1}^{M} \frac{\|x - f_\theta(z^{(m)})\|^2}{2\sigma_{\text{dec}}^2},$$

where:

$$z^{(m)} = \mu_\phi(x) + \sigma_\phi(x)\epsilon^{(m)}, \quad \epsilon^{(m)} \sim \mathcal{N}(0, \mathbf{I}).$$

VAE Training: Optimization Objective

To train a VAE, we need to solve the optimization problem:

$$\arg\max_{\theta,\phi} \sum_{x \in \mathcal{X}} \text{ELBO}_{\phi,\theta}(x),$$

where:

$$\text{ELBO}_{\phi,\theta}(x) = \text{Reconstruction Term} + \text{KL Divergence Term}.$$

Reconstruction Term:

$$-\frac{1}{M}\sum_{m=1}^{M} \frac{\|x - f_\theta\left(\mu_\phi(x) + \sigma_\phi(x)\epsilon^{(m)}\right)\|^2}{2\sigma_{\text{dec}}^2}.$$

KL Divergence Term:

$$+\frac{1}{2}\Big(\sigma_\phi^2(x)d - d + \|\mu_\phi(x)\|^2 - 2\log\sigma_\phi(x)\Big).$$

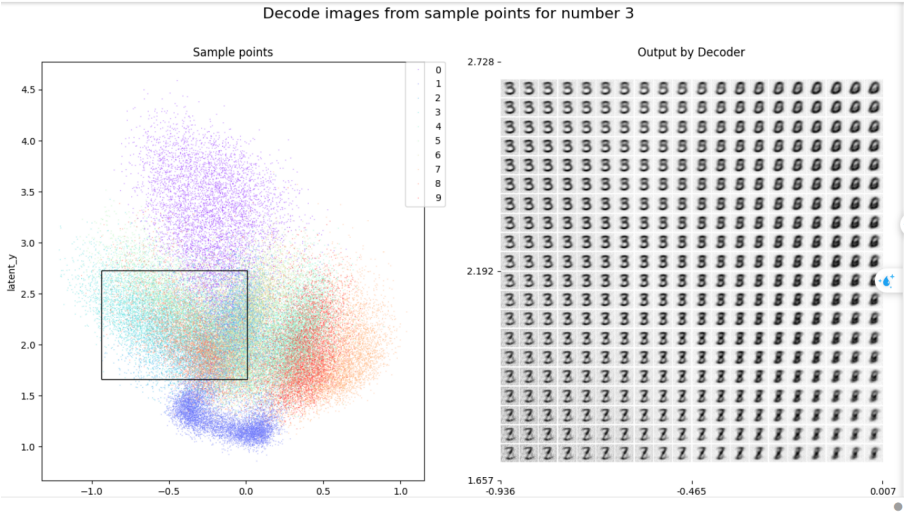
Explanation:

- $\mu_\phi(x)$: Mean output of the encoder.
- $\sigma_\phi(x)$: Standard deviation output of the encoder.
- d: Dimension of the latent variable z.

VIII. Applications of VAE

Applications of VAE: Image Generation

- VAE can generate new images similar to a given dataset.
- Example: Generating new handwritten digits using the MNIST dataset.
- VAE encodes the input into a latent space and decodes it back to generate new images.



Applications of VAE: Anomaly Detection

- VAE learns the normal distribution of data.
- Any data point with low reconstruction probability is flagged as an anomaly.
- Applications:
  - Fraud detection.
  - Industrial defect detection.

Applications of VAE: Data Compression

- The latent space z provides a compressed representation of the data.
- Applications:
  - Reducing storage requirements while retaining meaningful features.
  - Efficient transmission of high-dimensional data.

Applications of VAE: Medicine and Biology

- Generating molecular structures or drug compounds.
- Applications:
  - Drug discovery.
  - Protein structure prediction.