Variational Autoencoder (VAE)

Thanh Bui Ngoc Thien
University of Information Technology - VNU
23521436@gm.uit.edu.vn

Nam Nguyen Pham Phuong
University of Information Technology - VNU
23520978@gm.uit.edu.vn

Cuong Dang Quoc
University of Information Technology - VNU
23520192@gm.uit.edu.vn

Abstract

Variational Autoencoders (VAEs) are a powerful probabilistic framework that bridges mathematical modeling and deep learning. This paper delves into the mathematical foundations of VAEs, with a focus on deriving the Evidence Lower Bound (ELBO) as a core optimization objective. Key mathematical constructs such as the Kullback-Leibler divergence and the reparameterization trick are analyzed, enabling efficient gradient computation and stable training. Additionally, we provide rigorous insights into how latent space modeling with Gaussian distributions supports data generation and reconstruction tasks. The study highlights the interplay between mathematical rigor and computational efficiency, making VAEs an essential tool in dimensionality reduction, anomaly detection, and generative modeling. This work aims to deepen the understanding of VAEs' mathematical principles, offering a strong foundation for further exploration in both academic and applied contexts.

1. Introduction

1.1. Basic Structure of VAE

VAE consists of three main components: the encoder, the latent space, and the decoder [2].

- **Encoder:** The encoder takes input data and transforms it into a representation within the *latent space*. Unlike traditional AE, the encoder in VAE learns a probabilistic distribution instead of fixed points, allowing for more flexible and uncertainty-aware data representation.
- **Latent Space:** The latent space in VAE is a continuous and well-organized space. Each point in this space represents a latent state of the data. The continuity and smoothness of the latent space allow the model to generate new data by sampling points from it.
- **Decoder:** The decoder receives information from the latent space and reconstructs the original data. Additionally, it can generate new data by sampling points from the latent space, making VAE a highly effective generative model.

The key strength of VAE lies in its ability to balance accurate data reconstruction with the organization of the latent space, facilitating meaningful data generation.

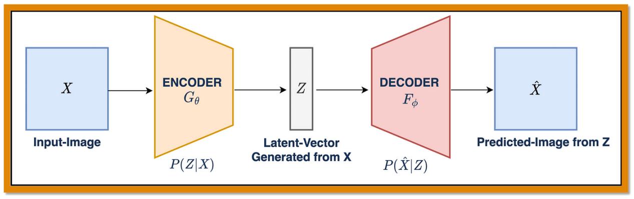


Figure 1. VAE Structure

1.2. Why Choose VAE

While the traditional Autoencoder excels at compressing and reconstructing data, it has limitations that VAE addresses. Here are the main reasons why VAE is preferred:

- **Data Generation:** Traditional AE only reconstructs learned data, whereas VAE can generate new data by sampling from the latent space.
- **Smooth Latent Space:** The latent space of VAE is continuous and smooth, ensuring that generated samples are reasonable even when sampled randomly.
- **Better Generalization:** VAE utilizes regularization techniques to avoid overfitting and improve its generalization capabilities on unseen data.
- **Uncertainty Handling:** VAE can learn and represent uncertainty in the data, a capability absent in traditional AE.

Thanks to these advantages, VAE serves as a powerful tool for data compression, reconstruction, and diverse applications such as image synthesis, anomaly detection, and data augmentation.

2. Some mathematical symbols are used

- $p(\mathbf{x})$: The true distribution of $\mathbf{x}$. It is never known. The whole universe of diffusion models is to find ways to draw samples from $p(\mathbf{x})$. If we knew $p(\mathbf{x})$ (say, we have a formula that describes $p(\mathbf{x})$), we can just draw a sample $\mathbf{x}$ that maximizes $\log p(\mathbf{x})$.
- $p(z)$: The distribution of the latent variable. Typically, we make it a zero-mean unit-variance Gaussian $\mathcal{N}(0, \mathbf{I})$. One reason is that linear transformation of a Gaussian remains a Gaussian, and so this makes the data processing easier. It was mentioned that any distribution can be generated by mapping a Gaussian through a sufficiently complicated function.
- $p(z|\mathbf{x})$: The conditional distribution associated with the *encoder*, which tells us the likelihood of z when given $\mathbf{x}$. We have no access to it. $p(z|\mathbf{x})$ itself is not the encoder, but the encoder has to do something so that it will behave consistently with $p(z|\mathbf{x})$.
- $p(\mathbf{x}|z)$: The conditional distribution associated with the *decoder*, which tells us the posterior probability of getting $\mathbf{x}$ given z. Again, we have no access to it.

In reality, $p(z|\mathbf{x})$ and $p(\mathbf{x}|z)$ are very difficult to calculate as well as their complications. Thus, we suggest using two proxy distributions:

- $q_\phi(z|\mathbf{x})$: The proxy for $p(z|\mathbf{x})$, which is also the distribution associated with the *encoder* [4]. $q_\phi(z|\mathbf{x})$ can be any directed graphical model and it can be parameterized using deep neural networks. For example, we can define

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

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

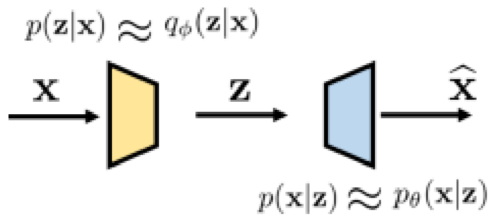
  This model is widely used because of its tractability and computational efficiency.
- $p_\theta(\mathbf{x}|z)$: The proxy for $p(\mathbf{x}|z)$, which is also the distribution associated with the *decoder*. Like the encoder, the decoder can be parameterized by a deep neural network. For example, we can define

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

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

  where σ_{dec} is a hyperparameter that can be pre-determined or it can be learned.

The relationship between the input $\mathbf{x}$ and the latent z, as well as the conditional distributions, are summarized below.



$p(\mathbf{z}|\mathbf{x}) \approx q_\phi(\mathbf{z}|\mathbf{x})$
$\mathbf{X}$
$\mathbf{Z}$
$\widehat{\mathbf{X}}$
$p(\mathbf{x}|\mathbf{z}) \approx p_\theta(\mathbf{x}|\mathbf{z})$

Figure 2. In a variational autoencoder, the variables $\mathbf{x}$ and z are connected by the conditional distributions $p(z|\mathbf{x})$ and $p(\mathbf{x}|z)$.

3. Build Encoder and Decoder in Variational AutoEncoder

Suppose that we have a random variable $x \in \mathbb{R}^d$ and a latent variable $z \in \mathbb{R}^d$ such that [1]:

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

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

We want to construct a Variational Autoencoder. By this, we mean that we want to build two mappings Encoder$(\cdot)$ and 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 knew what $p(x)$ is, then there is a trivial solution where:

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

In this case, the true distributions can be determined and expressed in terms of delta functions:

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

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

3.1. Definition of Dirac Delta Function

- The Dirac delta function [7] as the limit when $a \to 0$ of a sequence of normalized Gaussian distributions centered at 0 is expressed as:

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

- Where:
  - a is the scaling parameter,
  - $|a|$ is the absolute value of a,
  - $\sqrt{\pi}$ is the normalization constant,
  - $e^{-\left(\frac{x}{a}\right)^2}$ is the Gaussian exponential function.

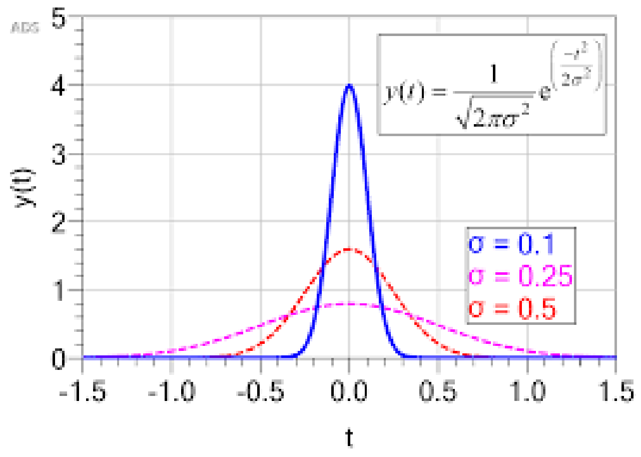


Figure 3. Dirac Delta Function

3.2. Encoder and Decoder Construction

Suppose now that we do not know $p(x)$, so we need to build an encoder and a decoder to estimate z and $\hat{x}$.

Encoder Design: The encoder takes the input x and generates a pair of parameters $\hat{\mu}(x)$ and $\hat{\sigma}^2(x)$, denoting the parameters of a Gaussian [8]. Then, we define $q_\phi(z \mid x)$ as:

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

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

For simplicity:

$$\hat{\mu}(x) = ax + b, \quad \hat{\sigma}^2(x) = t^2,$$

where a, b, and t are learned parameters. Substituting, we get:

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

Decoder Design: The decoder takes z and reconstructs x using:

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

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

Similarly, for simplicity:

$$\mu_e(z) = cz + v, \quad \sigma_e^2(z) = s^2,$$

where c, v, and s are learned parameters. Substituting, we get:

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

4. Evidence Lower Bound

If we treat ϕ and θ as two optimization variables, we need an objective function (Loss function) to optimize these two parameters. The Loss function we use here is called Evidence Lower Bound (ELBO).

Definition (Evidence Lower Bound):

$$\text{ELBO}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p(x,z)}{q_\phi(z|x)}\right]. \quad (1)$$

As mentioned earlier, $p(x)$ is the marginal probability of the observation variable x, calculated as:

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

- We need to optimize this probability to motivate the model to achieve the best data distribution, thereby simulating and understanding the data. However, calculating this integral is very complicated due to the large space of z and the lack of a closed-form solution for $p(x \mid z)$ and $p(z)$.
- To address this, we build the relationship between $p(x)$ and $\text{ELBO}(x)$. We suggest the following theorem:

Theorem 1.1. Decomposition of Log-Likelihood:

The log-likelihood $\log p(x)$ can be decomposed as:

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

Proof:

Using $q_\phi(z|x)$ as a proxy for $p(x)$, we have:

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

This implies:

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

We define the Kullback-Leibler Divergence [6] between two continuous distributions as:

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

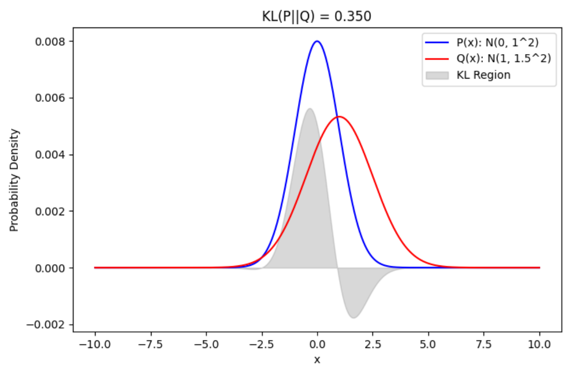


Figure 4. Kullback-Leibler Divergence

Using Bayes' theorem ($p(x,z) = p(z|x)p(x)$) [3]:

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

Rewriting:

$$\mathbb{E}_{q_\phi(z|x)}[\log p(x)] = \mathbb{E}_{q_\phi(z|x)}\Big[\log \frac{p(x,z)}{q_\phi(z|x)}\Big] + \mathbb{E}_{q_\phi(z|x)}\Big[\log \frac{q_\phi(z|x)}{p(z|x)}\Big]. \quad (2)$$

This simplifies to:

$$\log p(x) = \text{ELBO}(x) + D_{\text{KL}}(q_\phi(z|x) \| p(z|x)).$$

Theorem 1.2. ELBO Decomposition:

The ELBO can be expressed as:

$$\text{ELBO}(x) = \mathbb{E}_{q_\phi(z|x)}\big[\log p_\theta(x|z)\big] - D_{\text{KL}}(q_\phi(z|x) \| p(z)). \quad (3)$$

Proof:

Breaking the ELBO into two terms:

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

ELBO Terms:

- **Reconstruction:** The first term, $\mathbb{E}_{q_\phi(z|x)}[\log p_\theta(x|z)]$, measures the decoder's ability to reconstruct x from z. This term is maximized when the decoder is accurate.
- **Prior Matching:** The second term, $D_{\text{KL}}(q_\phi(z|x) \| p(z))$, regularizes $q_\phi(z|x)$ to align with the prior $p(z)$, typically $\mathcal{N}(0, I)$.

5. Optimization in VAE

5.1. Objective Function

The optimization objective of the VAE is to maximize the Evidence Lower Bound (ELBO):

$$(\phi, \theta) = \arg\max_{\phi,\theta} \sum_{x \in \mathcal{X}} \text{ELBO}(x), \quad (4)$$

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

The main challenge in optimizing the VAE lies in the intractability of the ELBO gradient with respect to (ϕ, θ). Since modern neural network optimizers rely on first-order methods like gradient descent and backpropagation, this intractability makes training difficult.

5.2. Gradient Computation for ELBO

The gradient of the ELBO can be expressed as:

$$\begin{aligned}\nabla_{\theta,\phi}\text{ELBO}(x) &= \nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z|x)}\left[\log \frac{p_\theta(x,z)}{q_\phi(z|x)}\right] \\ &= \nabla_{\theta,\phi}\mathbb{E}_{q_\phi(z|x)}\left[\log p_\theta(x,z) - \log q_\phi(z|x)\right].\end{aligned} \quad (5)$$

—

Gradient with Respect to θ

The gradient with respect to θ can be derived as:

$$\nabla_\theta \text{ELBO}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\theta \log p_\theta(x,z)\right]. \quad (6)$$

Using Monte Carlo approximation, we obtain:

$$\nabla_\theta \text{ELBO}(x) \approx \frac{1}{L}\sum_{\ell=1}^{L} \nabla_\theta \log p_\theta(x, z^{(\ell)}), \quad (7)$$

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

—

Gradient with Respect to ϕ

The gradient with respect to ϕ is more challenging:

$$\nabla_\phi \text{ELBO}(x) = \mathbb{E}_{q_\phi(z|x)}\left[\nabla_\phi \log p_\theta(x,z) - \nabla_\phi \log q_\phi(z|x)\right]. \quad (8)$$

Using Monte Carlo approximation for the second term:

$$\nabla_\phi \text{ELBO}(x) \approx \frac{1}{L}\sum_{\ell=1}^{L} \nabla_\phi \left(-\log q_\phi(z^{(\ell)}|x)\right), \quad (9)$$

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

—

5.3. Reparameterization Trick

To simplify gradient computation, the **Reparameterization Trick** is introduced. Recall that the latent variable z is a sample drawn from $q_\phi(z|x)$. The reparameterization trick expresses z as a differentiable and invertible transformation of another random variable ϵ, independent of x and ϕ.

Suppose:

$$z \sim q_\phi(z|x) = \mathcal{N}(z \mid \mu, \text{diag}(\sigma^2)), \quad (10)$$

we define:

$$z = g(\epsilon, \phi, x) = \epsilon \odot \sigma + \mu, \quad \epsilon \sim \mathcal{N}(0, I), \quad (11)$$

where $\odot$ denotes elementwise multiplication, and $\phi = (\mu, \sigma^2)$.

—

Jacobian Simplification

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

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

The expression for each component i is:

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

—

5.4. Gradient Computation with Reparameterization

Using the reparameterization trick, we aim to compute:

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

where $f(z)$ is some function of z (e.g., $f(z) = -\log q_\phi(z|x)$).

Substituting $f(z) = -\log q_\phi(z|x)$, we obtain:

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

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

—

5.5. Monte Carlo Approximation

Using Monte Carlo [4, 5], the expectation can be approximated as:

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

where $z^{(\ell)} = g(\epsilon^{(\ell)}, \phi, x)$ and $\epsilon^{(\ell)} \sim \mathcal{N}(0, I)$.

This reparameterization allows gradients to flow through $\phi = (\mu, \sigma^2)$ efficiently.

6. Specific Structure of VAE

6.1. Encoder

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

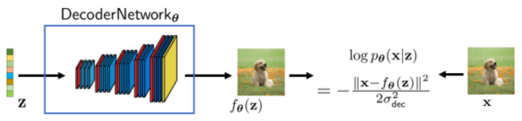
The parameters μ and σ are technically neural networks because they are outputs of $\text{EncoderNetwork}_\phi(\cdot)$.

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:

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



DecoderNetwork$_\theta$
$\mathbf{z}$
$f_\theta(\mathbf{z})$
$\log p_\theta(\mathbf{x}|\mathbf{z})$
$= -\frac{\|\mathbf{x}-f_\theta(\mathbf{z})\|^2}{2\sigma^2_{\text{dec}}}$
$\mathbf{x}$

Figure 5. Implementation of a VAE encoder. The neural network takes x and estimates μ_ϕ and σ^2_ϕ.

6.2. Sampling with Reparameterization Trick

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

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

6.3. KL Divergence of Two Gaussians

Theorem 1.3: KL-Divergence of Two Gaussians [2]
The KL divergence for two d-dimensional Gaussian distributions $\mathcal{N}(\mu_0, \Sigma_0)$ and $\mathcal{N}(\mu_1, \Sigma_1)$ is:

$$D_{\text{KL}}\left(\mathcal{N}(\mu_0, \Sigma_0) \parallel \mathcal{N}(\mu_1, \Sigma_1)\right) = \frac{1}{2}\Big[\text{Tr}(\Sigma_1^{-1}\Sigma_0) - d + (\mu_1 - \mu_0)^T \Sigma_1^{-1}(\mu_1 - \mu_0) + \log \frac{\det \Sigma_1}{\det \Sigma_0}\Big].$$

6.4. KL Divergence in VAE

Substituting Specific Distributions:

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

Simplified KL Divergence:

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

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

6.5. Decoder

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

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

The likelihood of the data given the latent variable, $p_\theta(x \mid z)$, is assumed to follow a Gaussian distribution:

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

where σ_{dec} is a hyperparameter that determines the variance of the Gaussian distribution.

Reparameterization Trick: The generated data point $\widehat{x}$ can be sampled as:

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

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

The log-likelihood of the data given z can be expressed as:

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

where d is the dimensionality of x.

The constant term $-\frac{d}{2}\log(2\pi\sigma^2_{\text{dec}})$ does not depend on the model parameters θ and can be ignored during optimization.

Explanation:

- $f_\theta(z)$: The deterministic mapping from z to the mean of the Gaussian distribution.
- σ^2_{dec}: Variance of the Gaussian distribution, controlling how tightly x is expected to cluster around $f_\theta(z)$.

6.6. VAE Training: Optimization Objective

To train a VAE, we need to solve the optimization problem [2]:

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

where:

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

Reconstruction Term

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

KL Divergence Term

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

Explanation of Terms

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

7. Applications of VAE

- Image Generation
- Anomaly Detection
- Data Compression
- Medicine and Biology

References

[1] A. Taylan Cemgil, Sumedh Ghaisas, Krishnamurthy Dvijotham, Sven Gowal, and Pushmeet Kohli. Autoencoding variational autoencoder. *arXiv preprint arXiv:2012.03715*, 2020. 2

[2] Stanley Chan. Tutorial on diffusion models for imaging and vision, 2025. 1, 5, 6

[3] Max Welling Diederik P. Kingma. An introduction to variational autoencoders, 2019. 4

[4] F. James. Monte carlo theory and practice. *Reports on Progress in Physics*, 43(9):1145, 1980. 2, 5

[5] Dirk P. Kroese, Tim Brereton, Thomas Taimre, and Zdravko I. Botev. Why the monte carlo method is so important today. *WIREs Computational Statistics*, 6(6):386–392, 2014. 5

[6] Jonathon Shlens. Notes on kullback-leibler divergence and likelihood. *arXiv preprint arXiv:1404.2000*, 2014. 3

[7] Nicholas Wheeler. Simplified production of dirac delta function identities. 1997. 2

[8] Yufeng Zhang, Wanwei Liu, Zhenbang Chen, Ji Wang, and Kenli Li. On the properties of kullback-leibler divergence between multivariate gaussian distributions. *arXiv preprint arXiv:1206.05485*, 2015. 3