# Introduction to Deep Learning for Computer Vision

Adhyayan '23 - ACA Summer School
Department of Computer Science and Engineering
Indian Institute of Technology Kanpur

Lecture 6

# Unsupervised Learning: Learning without Labels!

# Unsupervised Learning

- **Definition**: Learning from *unlabeled data*, where the goal is to *discover patterns, structure, or representations* without *explicit labels or supervision*.

# Unsupervised Learning

- **Definition**: Learning from *unlabeled data*, where the goal is to *discover patterns, structure, or representations* without *explicit labels or supervision*.
- **Advantages**:

# Unsupervised Learning

- **Definition**: Learning from *unlabeled data*, where the goal is to *discover patterns, structure, or representations* without *explicit labels or supervision*.
- **Advantages**:
  - Utilizing Unlabeled Data

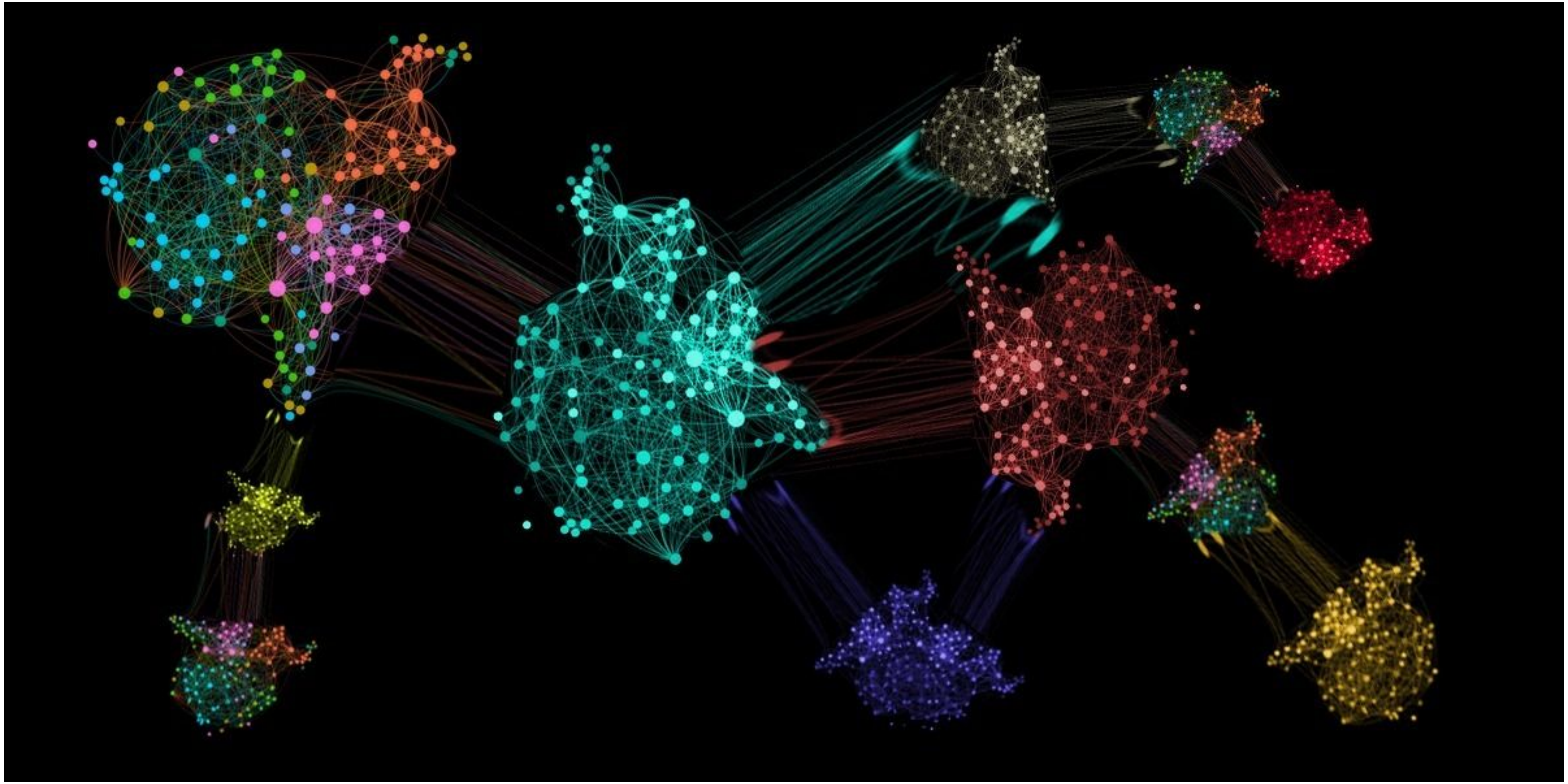
# Unsupervised Learning

- **Definition**: Learning from *unlabeled data*, where the goal is to *discover patterns, structure, or representations* without *explicit labels or supervision*.
- **Advantages**:
  - Utilizing Unlabeled Data
  - Discovery of Hidden Patterns

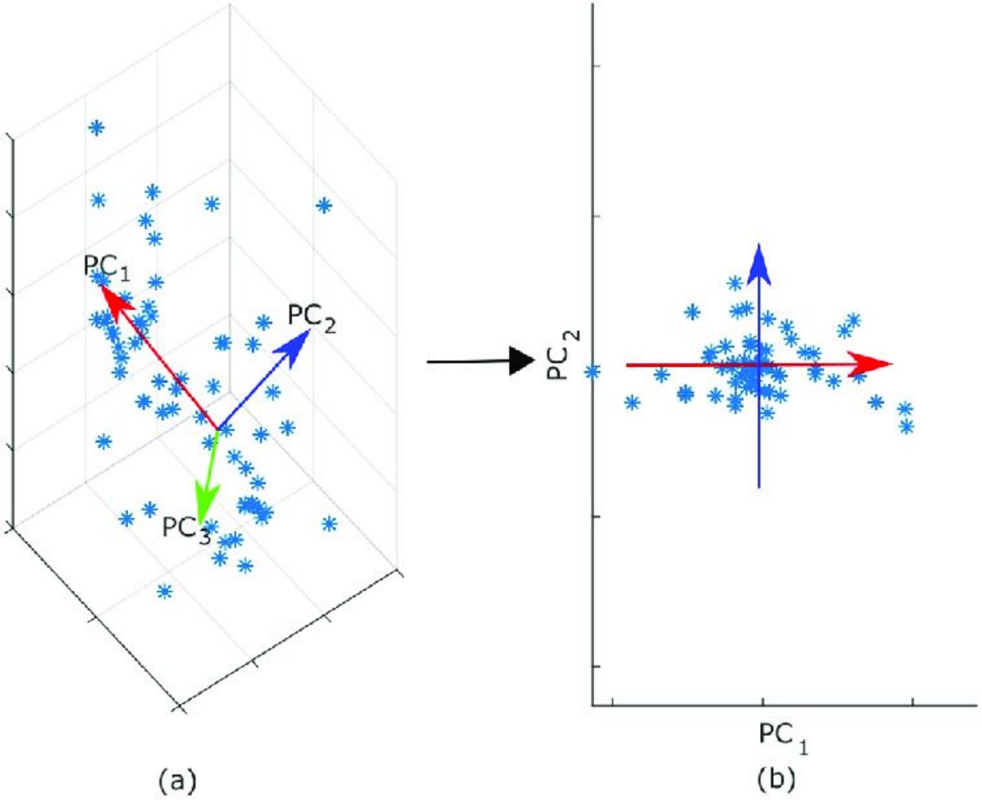
# Unsupervised Learning

- **Definition**: Learning from *unlabeled data*, where the goal is to *discover patterns, structure, or representations* without *explicit labels or supervision*.
- **Advantages**:
  - Utilizing Unlabeled Data
  - Discovery of Hidden Patterns
  - Handling Unlabeled or Scarce Labeled Data

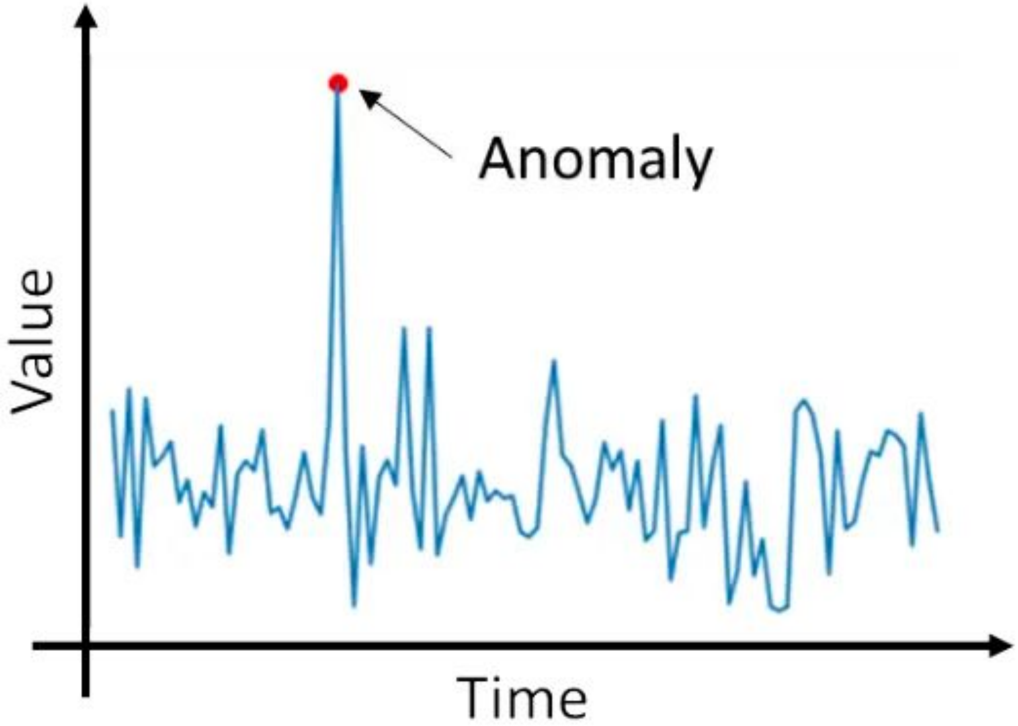
# Applications: Clustering



# Applications: Dimensionality Reduction



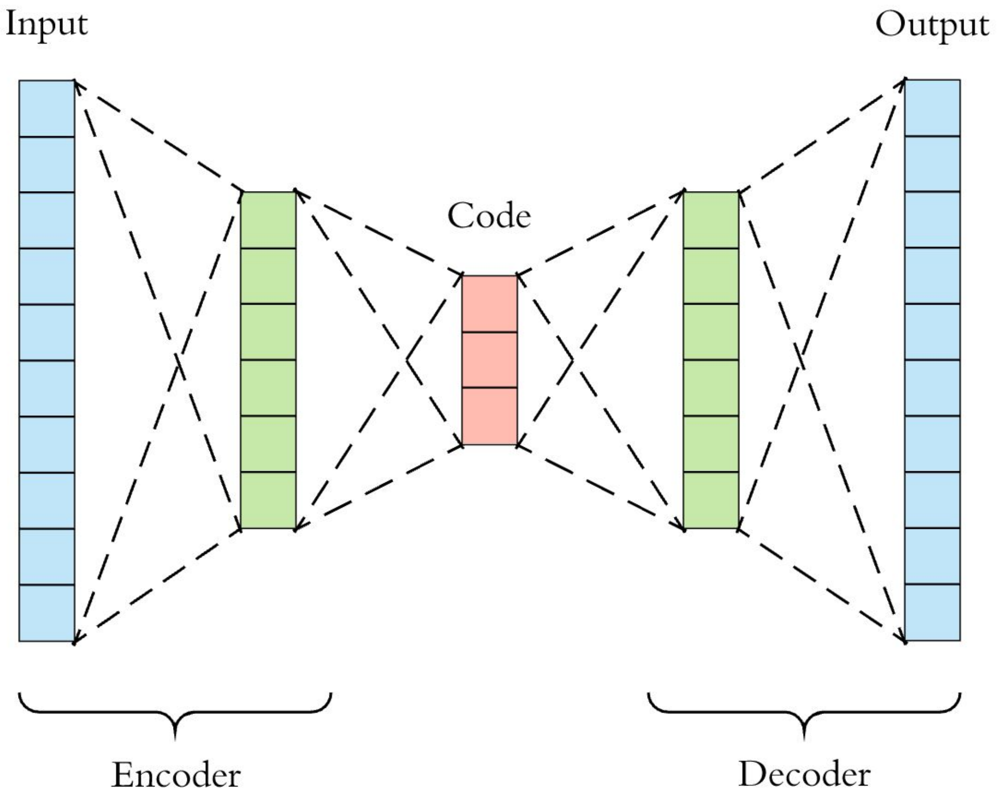
# Applications: Anomaly Detection



# Applications: Generative Modelling

# Unsupervised Deep Learning!

# Autoencoders



# Autoencoders

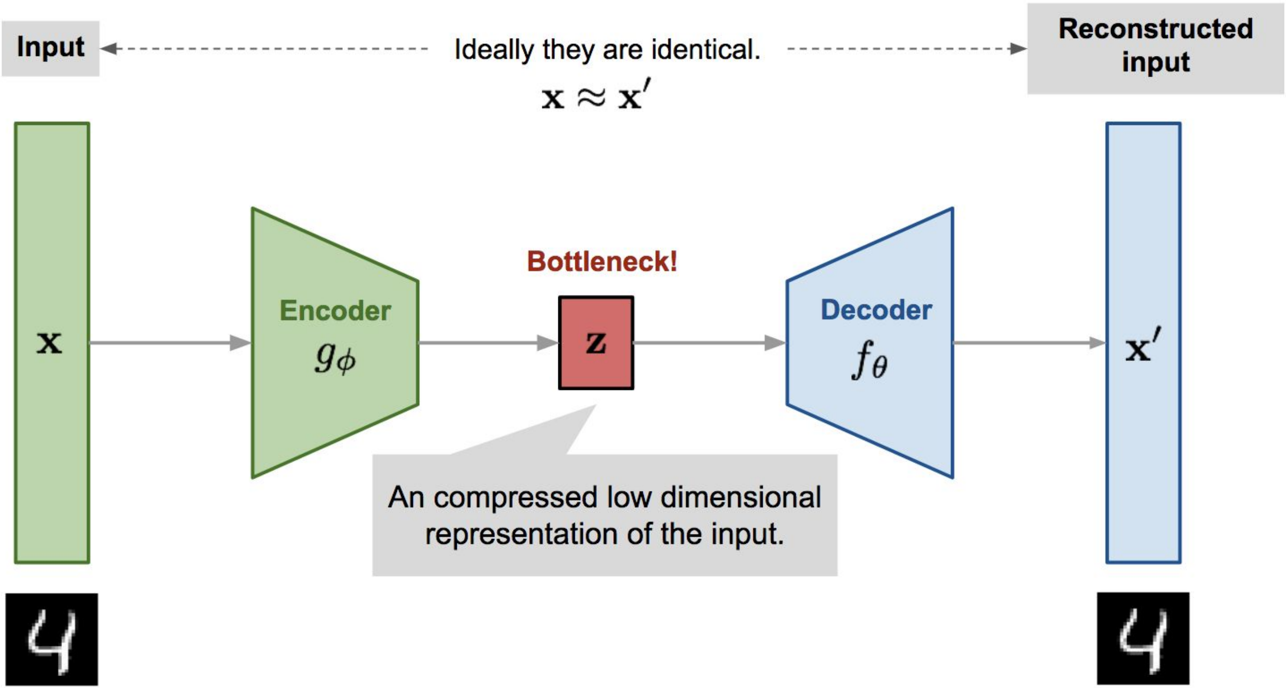
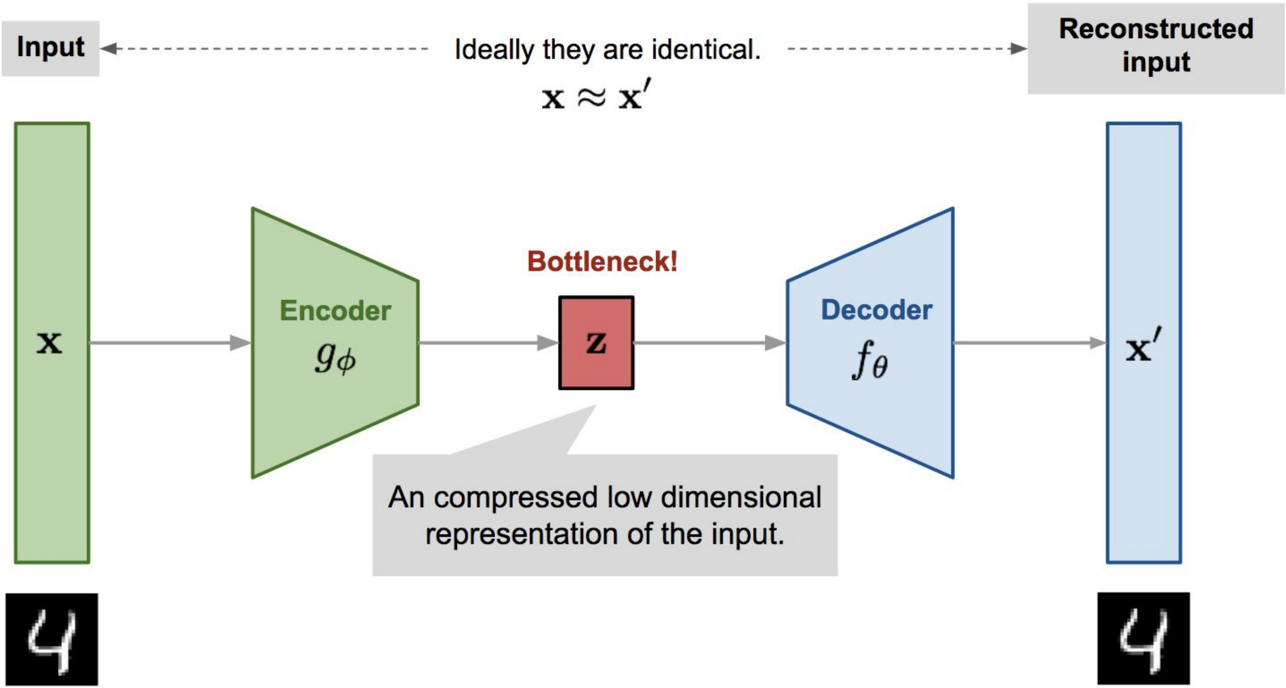



Image: https://lilianweng.github.io/posts/2018-08-12-vae/

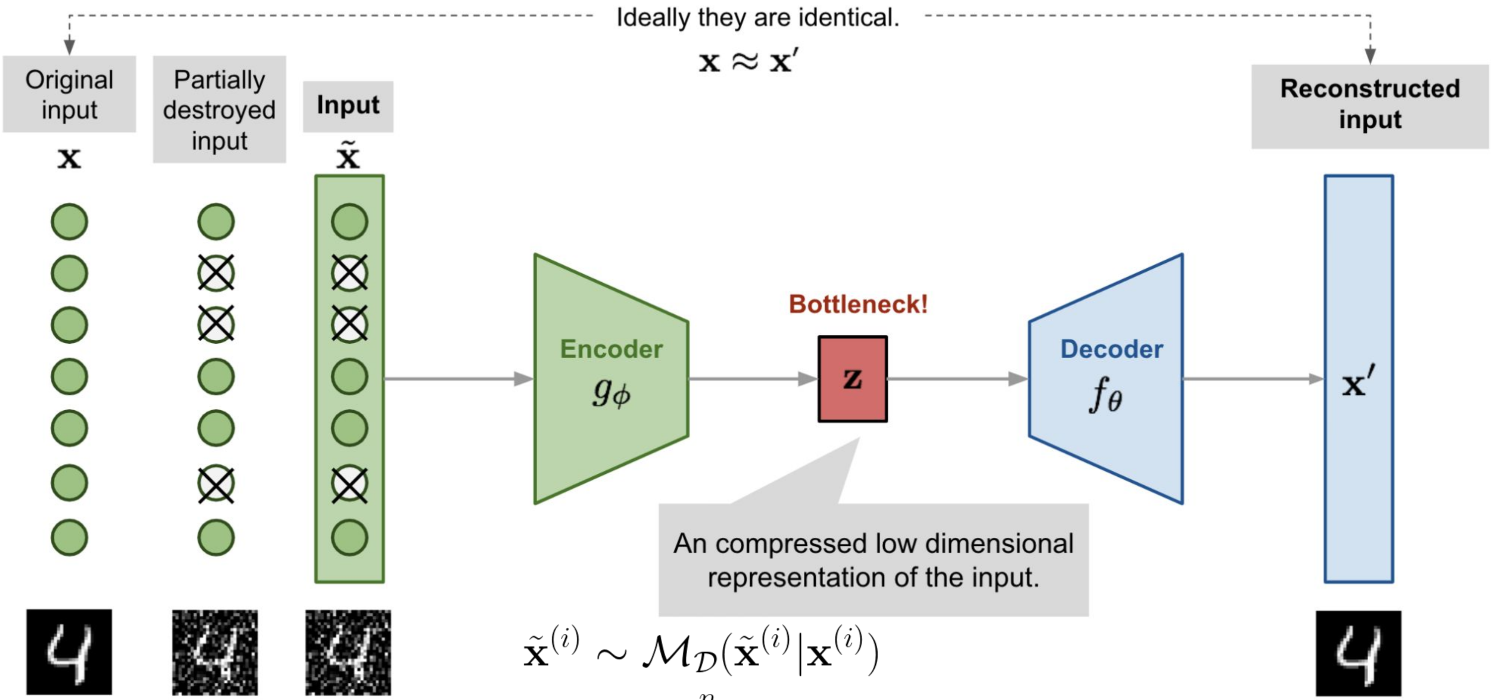
# Autoencoders




$$L_{\text{AE}}(\theta, \phi) = \frac{1}{n}\sum_{i=1}^{n}(\mathbf{x}^{(i)} - f_\theta(g_\phi(\mathbf{x}^{(i)})))^2$$

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

# Denoising Autoencoder




$$\tilde{\mathbf{x}}^{(i)} \sim \mathcal{M}_{\mathcal{D}}(\tilde{\mathbf{x}}^{(i)} | \mathbf{x}^{(i)})$$

$$L_{\text{DAE}}(\theta, \phi) = \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}^{(i)} - f_\theta(g_\phi(\tilde{\mathbf{x}}^{(i)})))^2$$

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

# Variational Autoencoder (VAE)

- **Latent Space**: The latent space is a lower-dimensional representation that captures the underlying structure of the input data. It allows for continuous interpolation and exploration of the data distribution.

# Variational Autoencoder (VAE)

- **Latent Space**: The latent space is a lower-dimensional representation that captures the underlying structure of the input data. It allows for continuous interpolation and exploration of the data distribution.
- **Key Idea**: Instead of mapping the input into a *fixed vector*, we want to map it into a *distribution*.

# Variational Autoencoder (VAE)

- **Latent Space**: The latent space is a lower-dimensional representation that captures the underlying structure of the input data. It allows for continuous interpolation and exploration of the data distribution.
- **Key Idea**: Instead of mapping the input into a *fixed vector*, we want to map it into a *distribution*.
- **Notations**:
  - Data: x

# Variational Autoencoder (VAE)

- **Latent Space**: The latent space is a lower-dimensional representation that captures the underlying structure of the input data. It allows for continuous interpolation and exploration of the data distribution.
- **Key Idea**: Instead of mapping the input into a *fixed vector*, we want to map it into a *distribution*.
- **Notations**:
  - Data: $x$
  - Latent Space: $z$

# Variational Autoencoder (VAE)

- **Latent Space**: The latent space is a lower-dimensional representation that captures the underlying structure of the input data. It allows for continuous interpolation and exploration of the data distribution.
- **Key Idea**: Instead of mapping the input into a *fixed vector*, we want to map it into a *distribution*.
- **Notations**:
  - Data: x
  - Latent Space: z
  - Prior Distribution: $p_\theta(\mathbf{z})$

# Variational Autoencoder (VAE)

- **Latent Space**: The latent space is a lower-dimensional representation that captures the underlying structure of the input data. It allows for continuous interpolation and exploration of the data distribution.
- **Key Idea**: Instead of mapping the input into a *fixed vector*, we want to map it into a *distribution*.
- **Notations**:
  - Data: x
  - Latent Space: z
  - Prior Distribution: $p_\theta(\mathbf{z})$
  - Likelihood: $p_\theta(\mathbf{x}|\mathbf{z})$

# Variational Autoencoder (VAE)

- **Latent Space**: The latent space is a lower-dimensional representation that captures the underlying structure of the input data. It allows for continuous interpolation and exploration of the data distribution.
- **Key Idea**: Instead of mapping the input into a *fixed vector*, we want to map it into a *distribution*.
- **Notations**:
  - Data: x
  - Latent Space: z
  - Prior Distribution: $p_\theta(\mathbf{z})$
  - Likelihood: $p_\theta(\mathbf{x}|\mathbf{z})$
  - Posterior: $p_\theta(\mathbf{z}|\mathbf{x})$

# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.

# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.
- We should be able to sample new data from a VAE. How?

# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.
- We should be able to sample new data from a VAE. How?
  - Sample $\mathbf{z}^{(i)}$ from the prior distribution $p_{\theta^*}(\mathbf{z})$

# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.
- We should be able to sample new data from a VAE. How?
  - Sample $\mathbf{z}^{(i)}$ from the prior distribution $p_{\theta^*}(\mathbf{z})$
  - $\mathbf{x}^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$

# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.
- We should be able to sample new data from a VAE. How?
  - Sample $\mathbf{z}^{(i)}$ from the prior distribution $p_{\theta^*}(\mathbf{z})$
  - $\mathbf{x}^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$
- Optimal Parameter $\theta^*$ is the one that maximizes the probability of generating real data samples: $\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)})$

# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.
- We should be able to sample new data from a VAE. How?
  - Sample $\mathbf{z}^{(i)}$ from the prior distribution $p_{\theta^*}(\mathbf{z})$
  - $\mathbf{x}^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$
- Optimal Parameter $\theta^*$ is the one that maximizes the probability of generating real data samples: $\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)})$
- Commonly we use the log probabilities to convert the product on RHS to a sum: $\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(\mathbf{x}^{(i)})$

# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.
- We should be able to sample new data from a VAE. How?
  - Sample $\mathbf{z}^{(i)}$ from the prior distribution $p_{\theta^*}(\mathbf{z})$
  - $\mathbf{x}^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$
- Optimal Parameter $\theta^*$ is the one that maximizes the probability of generating real data samples: $\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)})$
- Commonly we use the log probabilities to convert the product on RHS to a sum: $\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(\mathbf{x}^{(i)})$
- Let's expand on the probability of generating real samples:

$$p_\theta(\mathbf{x}^{(i)}) = \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

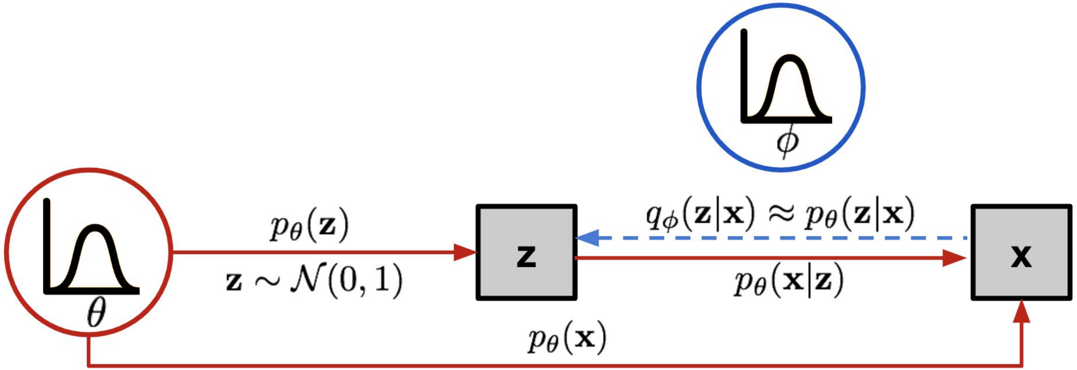
# Variational Autoencoder (VAE)

- VAEs are **Generative Models**.
- We should be able to sample new data from a VAE. How?
  - Sample $\mathbf{z}^{(i)}$ from the prior distribution $p_{\theta^*}(\mathbf{z})$
  - $\mathbf{x}^{(i)}$ is generated from a conditional distribution $p_{\theta^*}(\mathbf{x}|\mathbf{z} = \mathbf{z}^{(i)})$
- Optimal Parameter $\theta^*$ is the one that maximizes the probability of generating real data samples: $\theta^* = \arg\max_\theta \prod_{i=1}^{n} p_\theta(\mathbf{x}^{(i)})$
- Commonly we use the log probabilities to convert the product on RHS to a sum: $\theta^* = \arg\max_\theta \sum_{i=1}^{n} \log p_\theta(\mathbf{x}^{(i)})$
- Let's expand on the probability of generating real samples:

$$p_\theta(\mathbf{x}^{(i)}) = \int p_\theta(\mathbf{x}^{(i)}|\mathbf{z}) p_\theta(\mathbf{z}) d\mathbf{z}$$

Not easy to compute.
Need to approximate!

# Variational Autoencoder (VAE)




- Unfortunately it is not easy to compute $p_\theta(\mathbf{x}^{(i)})$
- **Better Idea**: Introduce a new approximation function to output what is a likely code given an input, $\mathbf{X}$ $q_\phi(\mathbf{z}|\mathbf{x})$, parameterized by $\phi$
- Now the structure resembles an Autoencoder:
  - $p_\theta(\mathbf{x}|\mathbf{z})$ is similar to the decoder $f_\theta(\mathbf{x}|\mathbf{z})$. Also known as *probabilistic decoder*.
  - $q_\phi(\mathbf{z}|\mathbf{x})$ is similar to the encoder $g_\phi(\mathbf{z}|\mathbf{x})$. Also known as *probabilistic encoder*.

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

# Variational Autoencoder (VAE)

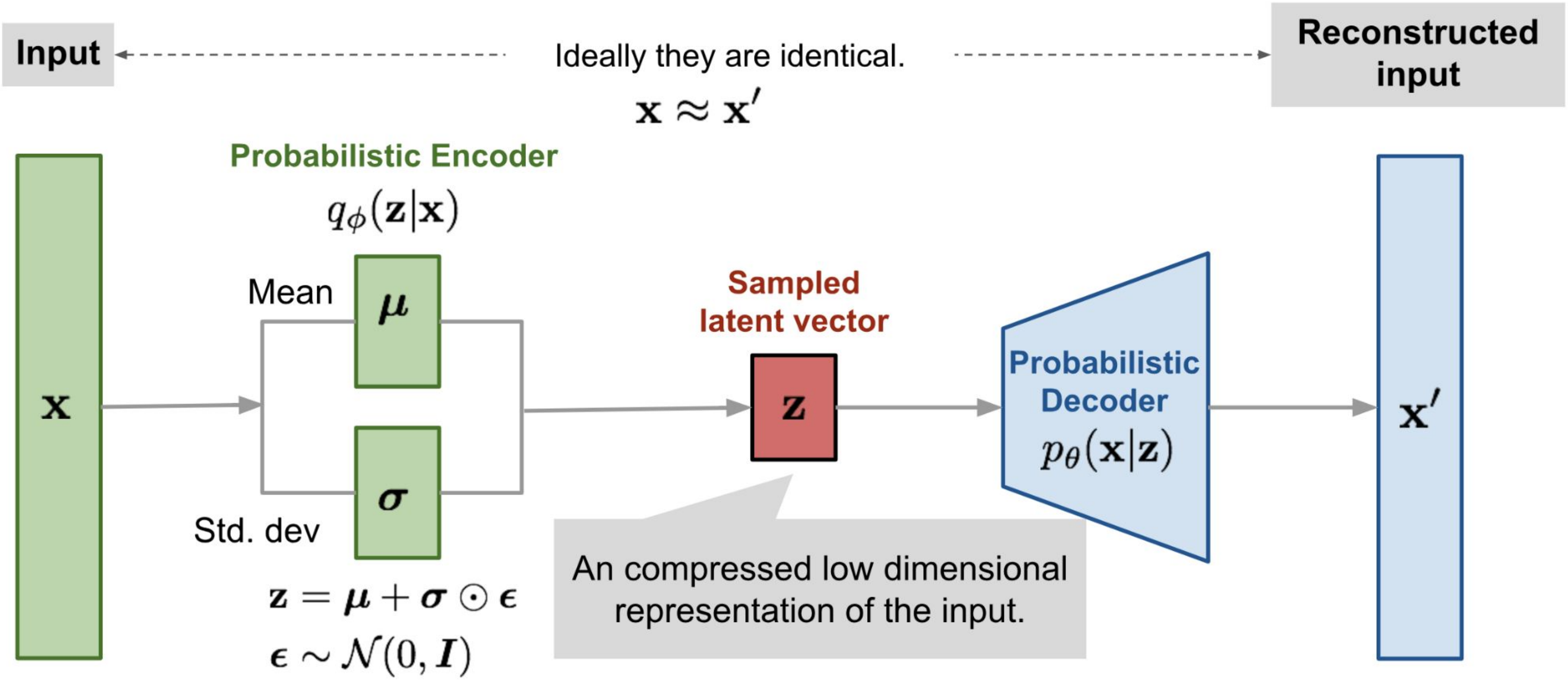



Image: https://lilianweng.github.io/posts/2018-08-12-vae/

# Loss Function: KL Divergence

- Target: Estimated Posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should be very close to the real one $p_\theta(\mathbf{z}|\mathbf{x})$

# Loss Function: KL Divergence

- Target: Estimated Posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should be very close to the real one $p_\theta(\mathbf{z}|\mathbf{x})$
- How do we **quantify closeness** of distributions?

# Loss Function: KL Divergence

- Target: Estimated Posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should be very close to the real one $p_\theta(\mathbf{z}|\mathbf{x})$
- How do we **quantify closeness** of distributions? Ans: **KL Divergence**!

$$D_{\mathrm{KL}}(P|Q) = \mathbb{E}_{z\sim P(z)} \log \frac{P(z)}{Q(z)}$$

# Loss Function: KL Divergence

- Target: Estimated Posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should be very close to the real one $p_\theta(\mathbf{z}|\mathbf{x})$
- How do we **quantify closeness** of distributions? Ans: **KL Divergence**!

$$D_{\mathrm{KL}}(P|Q) = \mathbb{E}_{z\sim P(z)} \log \frac{P(z)}{Q(z)}$$

- $D_{\mathrm{KL}}(X|Y)$ measures how much information is lost if the distribution Y is used to represent X.

# Loss Function: KL Divergence

- Target: Estimated Posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should be very close to the real one $p_\theta(\mathbf{z}|\mathbf{x})$
- How do we **quantify closeness** of distributions? Ans: **KL Divergence**!

$$D_{\mathrm{KL}}(P|Q) = \mathbb{E}_{z\sim P(z)} \log \frac{P(z)}{Q(z)}$$

- $D_{\mathrm{KL}}(X|Y)$ measures how much information is lost if the distribution Y is used to represent X.
- We want to minimize $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})|p_\theta(\mathbf{z}|\mathbf{x}))$ with respect to $\Phi$.

# Loss Function: KL Divergence

- Target: Estimated Posterior $q_\phi(\mathbf{z}|\mathbf{x})$ should be very close to the real one $p_\theta(\mathbf{z}|\mathbf{x})$
- How do we **quantify closeness** of distributions? Ans: **KL Divergence**!

$$D_{KL}(P\|Q) = \int_z P(z)\log\frac{P(z)}{Q(z)}dz$$

- $D_{\mathrm{KL}}(X|Y)$ measures how much information is lost if the distribution Y is used to represent X.
- We want to minimize $D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})|p_\theta(\mathbf{z}|\mathbf{x}))$ with respect to *Φ*.
- Why use $D_{\mathrm{KL}}(q_\phi|p_\theta)$ (reverse KL) instead of $D_{\mathrm{KL}}(p_\theta|q_\phi)$ (forward KL)?
  Ans: [https://blog.evjang.com/2016/08/variational-bayes.html](https://blog.evjang.com/2016/08/variational-bayes.html)

# Loss Function: ELBO

$$L_{\text{VAE}}(\theta, \phi) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

$$\theta^*, \phi^* = \arg\min_{\theta, \phi} L_{\text{VAE}}$$

Reconstruction Loss

Regularization

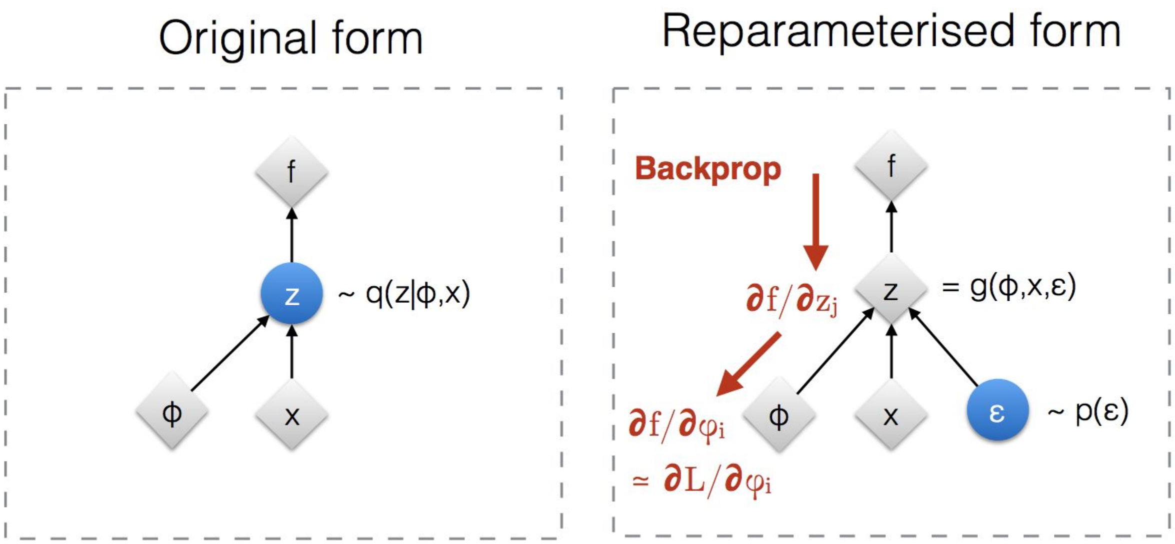
# Reparameterization Trick

- The expectation term in the loss function invokes generating samples from $\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})$
- Sampling is a stochastic process. Backpropagation is not possible.
- To make it trainable, the reparameterization trick is introduced:
  - It is often possible to express the random variable *z* as a deterministic variable $\mathbf{z} = \mathcal{T}_\phi(\mathbf{x}, \boldsymbol{\epsilon})$ where is an auxiliary independent random variable, and the transformation function $\mathcal{T}_\phi$ parameterized by *Φ* converts ε to z.

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

$$\mathbf{z} = \boldsymbol{\mu} + \boldsymbol{\sigma} \odot \boldsymbol{\epsilon}, \text{ where } \boldsymbol{\epsilon} \sim \mathcal{N}(0, \boldsymbol{I}) \quad ; \text{Reparameterization trick.}$$

# Reparameterization Trick



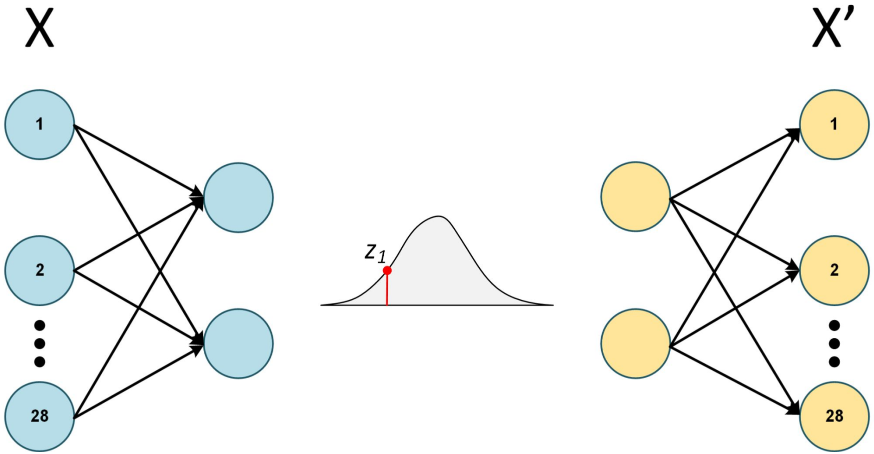

: Deterministic node

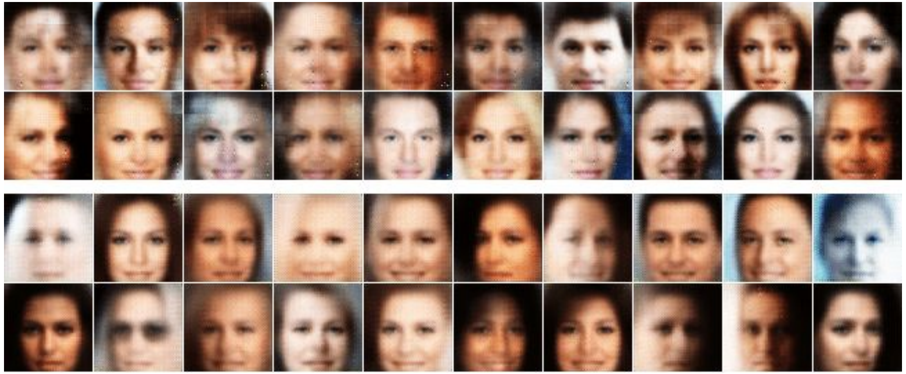
: Random node

[Kingma, 2013]
[Bengio, 2013]
[Kingma and Welling 2014]
[Rezende et al 2014]

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

# Sample Generation using VAE



# β-VAE : Regularizing the Regularizer!

$$L_{\mathrm{BETA}}(\phi, \beta) = -\mathbb{E}_{\mathbf{z} \sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z}) + \beta D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z}))$$

# Next Lecture: GANs, Diffusion Models!

# Expanding the KL Divergence

$$
\begin{aligned}
&D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}|\mathbf{x})} d\mathbf{z} \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x}) p_\theta(\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z} && ;\ \text{Because } p(z|x)=p(z,x)/p(x) \\
&= \int q_\phi(\mathbf{z}|\mathbf{x}) \big( \log p_\theta(\mathbf{x}) + \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} \big) d\mathbf{z} \\
&= \log p_\theta(\mathbf{x}) + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z}, \mathbf{x})} d\mathbf{z} && ;\ \text{Because } \int q(z|x)dz=1 \\
&= \log p_\theta(\mathbf{x}) + \int q_\phi(\mathbf{z}|\mathbf{x}) \log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{x}|\mathbf{z}) p_\theta(\mathbf{z})} d\mathbf{z} && ;\ \text{Because } p(z,x)=p(x|z)p(z) \\
&= \log p_\theta(\mathbf{x}) + \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}[\log \frac{q_\phi(\mathbf{z}|\mathbf{x})}{p_\theta(\mathbf{z})} - \log p_\theta(\mathbf{x}|\mathbf{z})] \\
&= \log p_\theta(\mathbf{x}) + D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})} \log p_\theta(\mathbf{x}|\mathbf{z})
\end{aligned}
$$

# Deriving the Evidence Lower Bound

$$D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) = \log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) - \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z})$$

$$\log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) = \mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}))$$

$$\begin{aligned} L_{\text{VAE}}(\theta, \phi) &= -\log p_\theta(\mathbf{x}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \\ &= -\mathbb{E}_{\mathbf{z}\sim q_\phi(\mathbf{z}|\mathbf{x})}\log p_\theta(\mathbf{x}|\mathbf{z}) + D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z})) \\ \theta^*, \phi^* &= \arg\min_{\theta,\phi} L_{\text{VAE}} \end{aligned}$$

$$-L_{\text{VAE}} = \log p_\theta(\mathbf{x}) - D_{\text{KL}}(q_\phi(\mathbf{z}|\mathbf{x})\|p_\theta(\mathbf{z}|\mathbf{x})) \leq \log p_\theta(\mathbf{x})$$