

Deep Generative Models

Prepared by Dr. Sherif Mostafa

AI314

Master of Science in Artificial Intelligence

Lecture 3



Variational Autoencoders (VAEs)

Introduction

Autoencoders

Vibrational Autoencoders

Training the Vibrational autoencoders

Application the Vibrational autoencoders

What are Autoencoders?

Autoencoders are a specialized class of algorithms that can learn efficient representations of input data with no need for labels. It is a class of artificial neural networks designed for unsupervised learning. Learning to compress and effectively represent input data without specific labels is the essential principle of an automatic decoder.

What are Autoencoders?

Autoencoders are used to learn generative models of data in an unsupervised manner. Autoencoders usually consist of one hidden layer that has smaller number of neurons than input or output layers. The goal of autoencoders is to reconstruct the input data at the output layer, and the outputs of hidden layer can be considered as compressed representation of data. Autoencoders can have more than one hidden layer, but they are usually treated as single-hidden-layer autoencoders stacked together, and they are trained one by one

Architecture of Autoencoder in Deep Learning

Encoder

- Input layer take raw input data
- The hidden layers progressively reduce the dimensionality of the input, capturing important features and patterns. These layer compose the encoder.
- The bottleneck layer (latent space) is the final hidden layer, where the dimensionality is significantly reduced. This layer represents the compressed encoding of the input data.

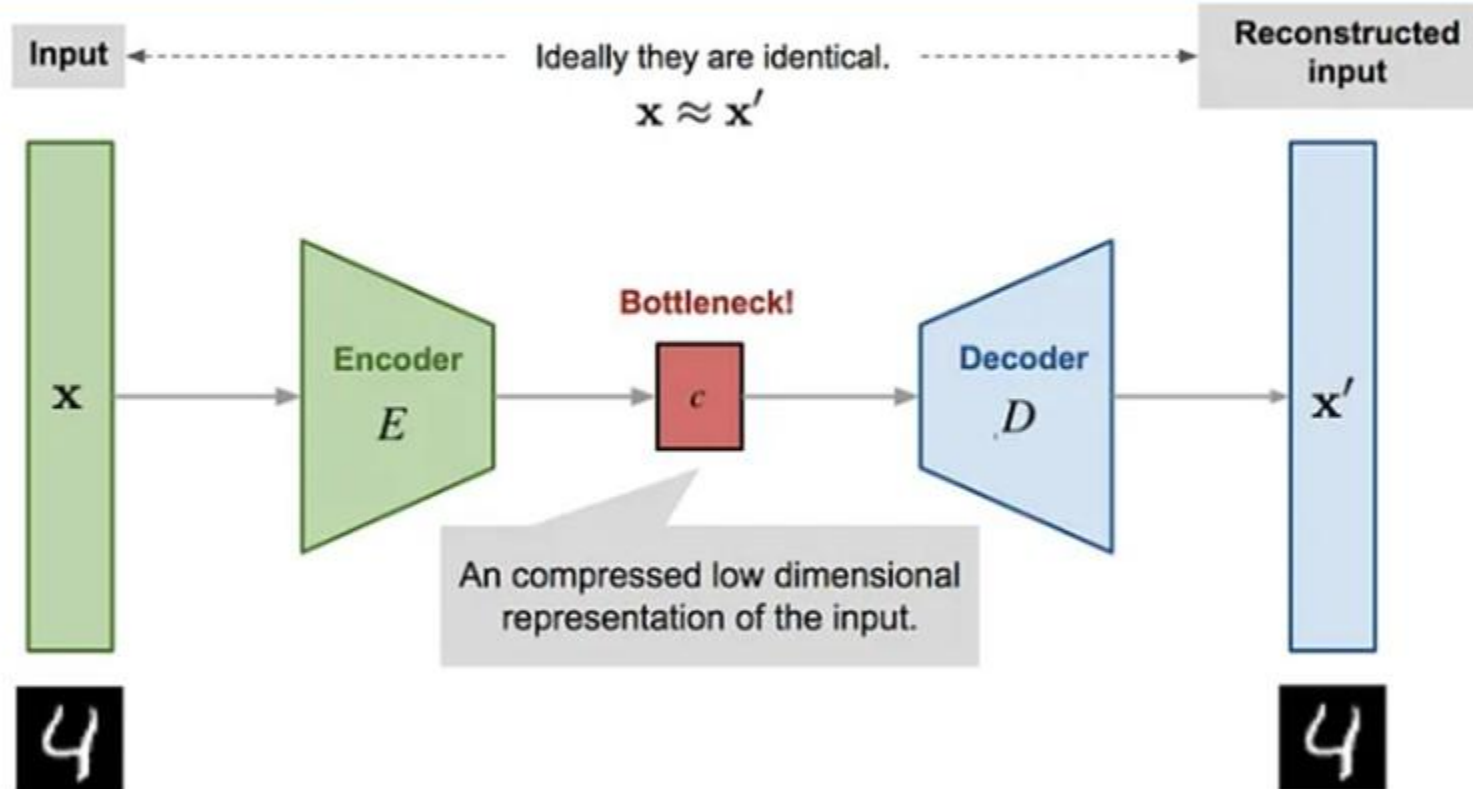
Architecture of Autoencoder in Deep Learning

Decoder

- The bottleneck layer takes the encoded representation and expands it back to the dimensionality of the original input.
- The hidden layers progressively increase the dimensionality and aim to reconstruct the original input.
- The output layer produces the reconstructed output, which ideally should be as close as possible to the input data.

Chapter
03

Architecture of Autoencoder in Deep Learning



The general architecture of an autoencoder includes an encoder, decoder, and bottleneck layer

Architecture of Autoencoder in Deep Learning

So, we can sum up that an Autoencoder has two distinct components :

- **An encoder:** This part of the model takes in parameter the input data and compresses it. $E(x) = c$ where x is the input data, c the latent representation and E our encoding function.
- **A decoder:** This part takes in parameter the latent representation and try to reconstruct the original input. $D(c) = x'$ where x' is the output of the decoder and D our decoding function

The network is trained to minimize the difference between the input and the reconstructed output.

Architecture of Autoencoder in Deep Learning

Autoencoders are primarily used for unsupervised learning, as they do not require labeled data during the training phase. However, they can also be adapted for semi-supervised or supervised tasks by incorporating labeled information into the training process.

Architecture of Autoencoder in Deep Learning

The loss function used during training is typically a reconstruction loss, measuring the difference between the input and the reconstructed output. Common choices include mean squared error (MSE) for continuous data or binary cross-entropy for binary data.

During training, the autoencoder learns to minimize the reconstruction loss, forcing the network to capture the most important features of the input data in the bottleneck layer.

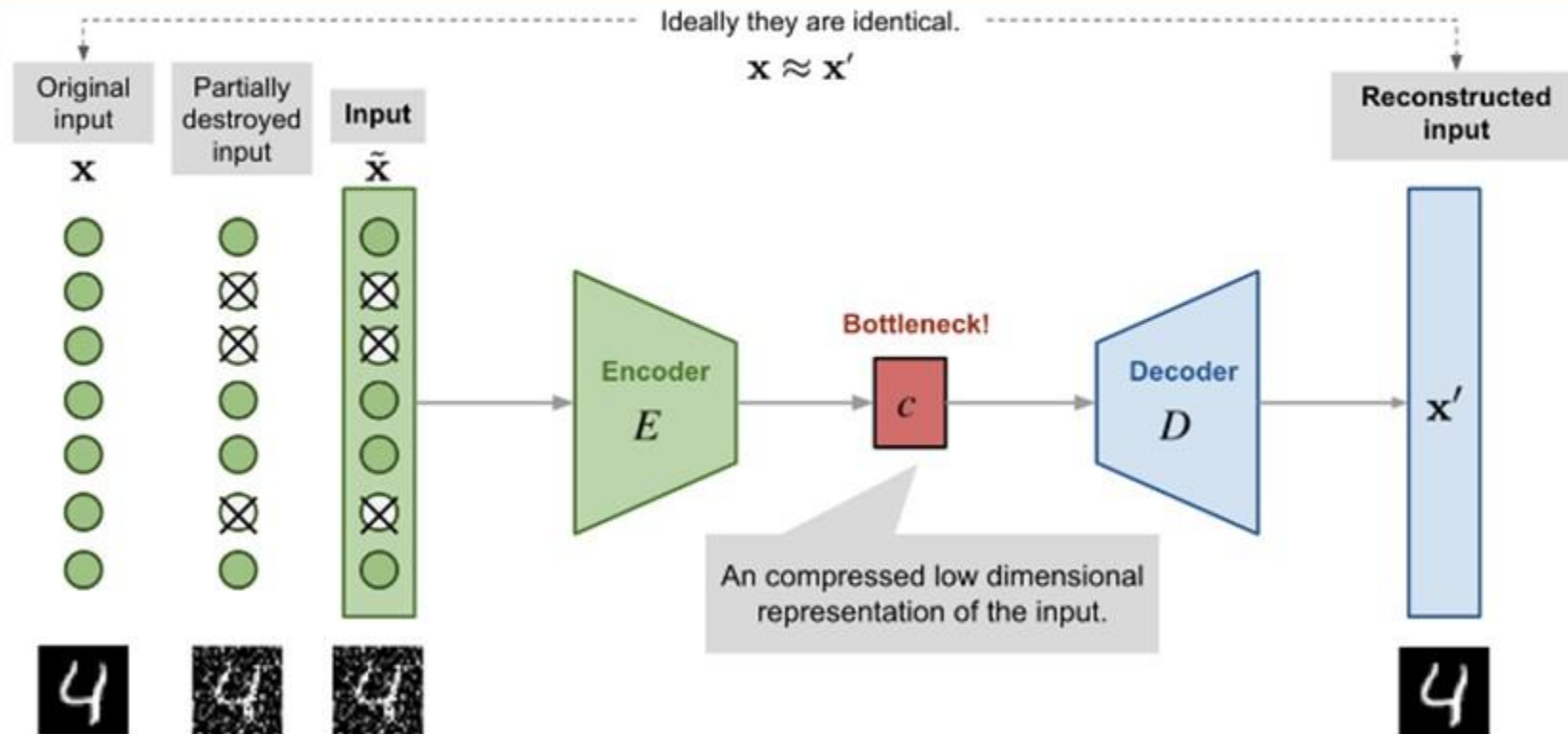
Types of Autoencoders

Denoising autoencoder

works on a partially corrupted input and trains to recover the original undistorted image. This method is an effective way to constrain the network from simply copying the input and thus learn the underlying structure and important features of the data.

Chapter
03

Types of Autoencoders



Denoising Autoencoder Architecture

Types of Autoencoders (Denoising autoencoder)

Advantages

1. This type of autoencoder can extract important features and reduce the noise or the useless features.
2. Denoising autoencoders can be used as a form of data augmentation, the restored images can be used as augmented data thus generating additional training samples.

Disadvantages

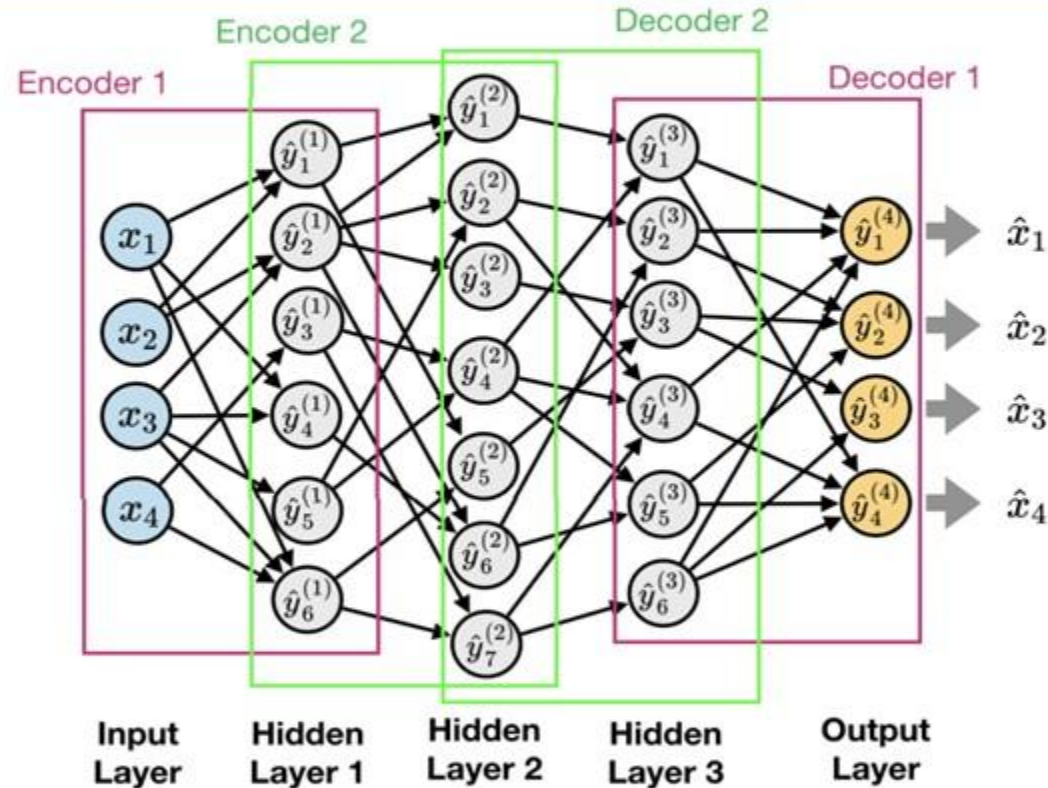
1. Selecting the right type and level of noise to introduce can be challenging and may require domain knowledge.
2. Denoising process can result into loss of some information that is needed from the original input. This loss can impact accuracy of the output.

Types of Autoencoders (Sparse Autoencoder)

Sparse Autoencoder

This type of autoencoder typically contains more hidden units than the input but only a few are allowed to be active at once. This property is called the sparsity of the network. The sparsity of the network can be controlled by either manually zeroing the required hidden units, tuning the activation functions or by adding a loss term to the cost function.

Types of Autoencoders (Sparse Autoencoder)



An autoencoder with three hidden layers as two single-layer autoencoders stacked together.

Types of Autoencoders (Sparse Autoencoder)

Advantages

1. The sparsity constraint in sparse autoencoders helps in filtering out noise and irrelevant features during the encoding process.
2. These autoencoders often learn important and meaningful features due to their emphasis on sparse activations.

Disadvantages

1. The choice of hyperparameters play a significant role in the performance of this autoencoder. Different inputs should result in the activation of different nodes of the network.
2. The application of sparsity constraint increases computational complexity.

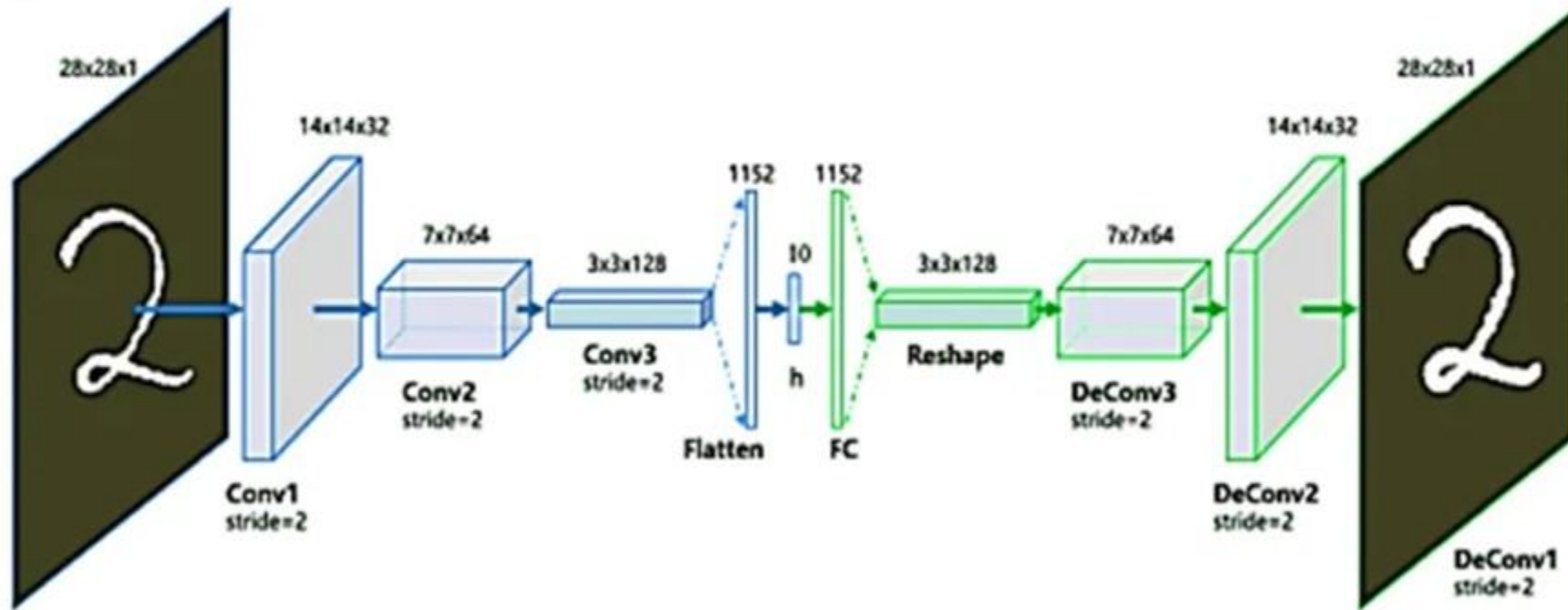
Types of Autoencoders (Convolutional Autoencoder)

Convolutional autoencoders are a type of autoencoder that use convolutional neural networks (CNNs) as their building blocks. The encoder typically consists of a series of convolutional layers, each followed by a pooling layer. The convolutional layers apply filters to the input image to extract features, while the pooling layers reduce the spatial dimensions of the feature maps forming a compressed representation of the input.

The decoder is the reverse of the encoder and typically consists of a series of up-sampling layers followed by convolutional layers. The up-sampling layers increase the spatial dimensions of the feature maps, while the convolutional layers apply filters to reconstruct the original image.

Chapter
03

Types of Autoencoders (Convolutional Autoencoder)



Convolutional autoencoders architecture

Types of Autoencoders (Convolutional Autoencoder)

Advantages

1. Convolutional autoencoder can compress high-dimensional image data into a lower-dimensional data. This improves storage efficiency and transmission of image data.
2. Convolutional autoencoder can reconstruct missing parts of an image. It can also handle images with slight variations in object position or orientation.

Disadvantages

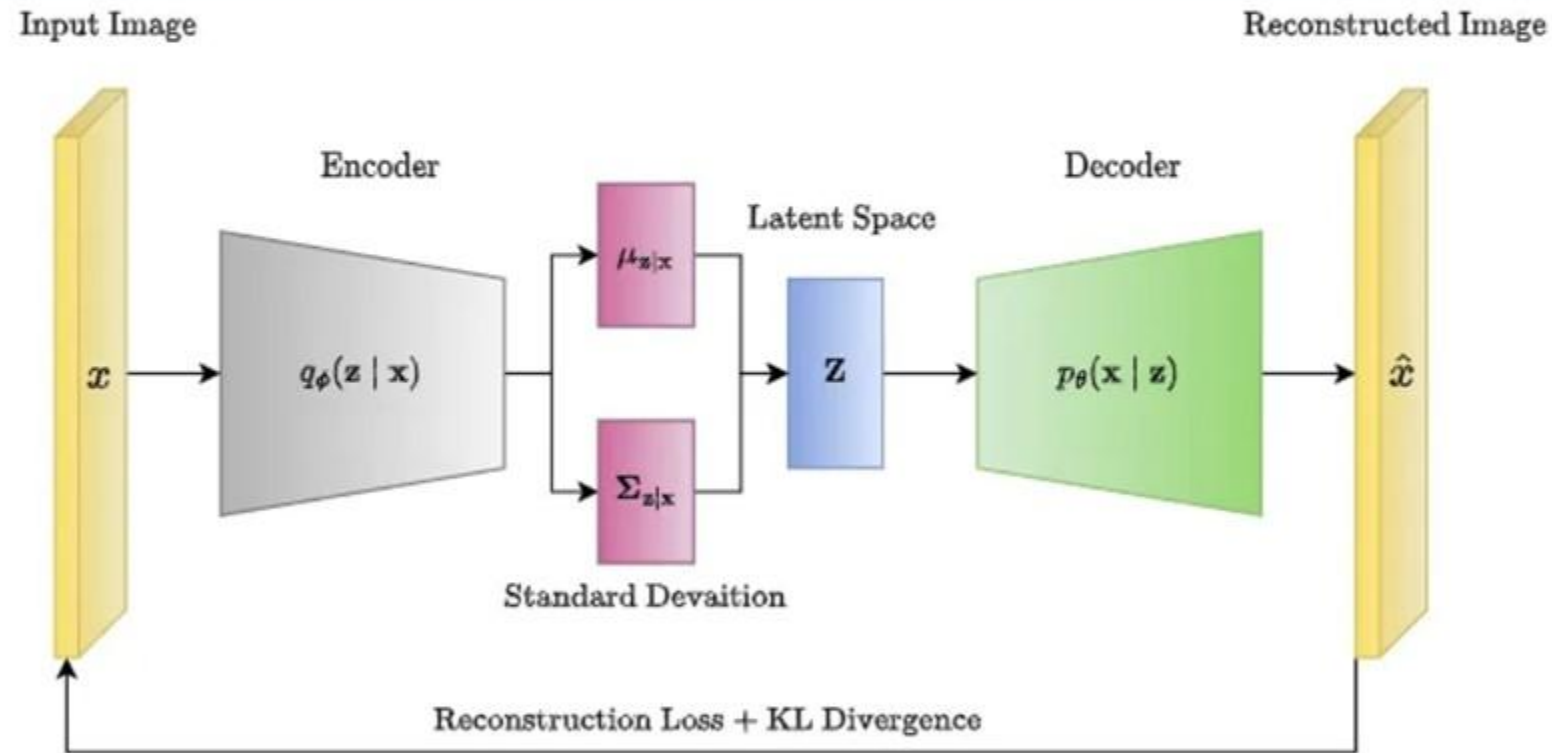
1. These autoencoder are prone to overfitting. Proper regularization techniques should be used to tackle this issue.
2. Compression of data can cause data loss which can result in reconstruction of a lower quality image.

Types of Autoencoders (Variational Autoencoder)

Variational autoencoder makes strong assumptions about the distribution of latent variables and uses the Stochastic Gradient Variational Bayes estimator in the training process. It assumes that the data is generated by a Directed Graphical Model and tries to learn an approximation to $q_{\phi}(Z|X)$ to the conditional property $q_{\theta}(Z|X)$ where ϕ and θ are the parameters of the encoder and the decoder respectively.

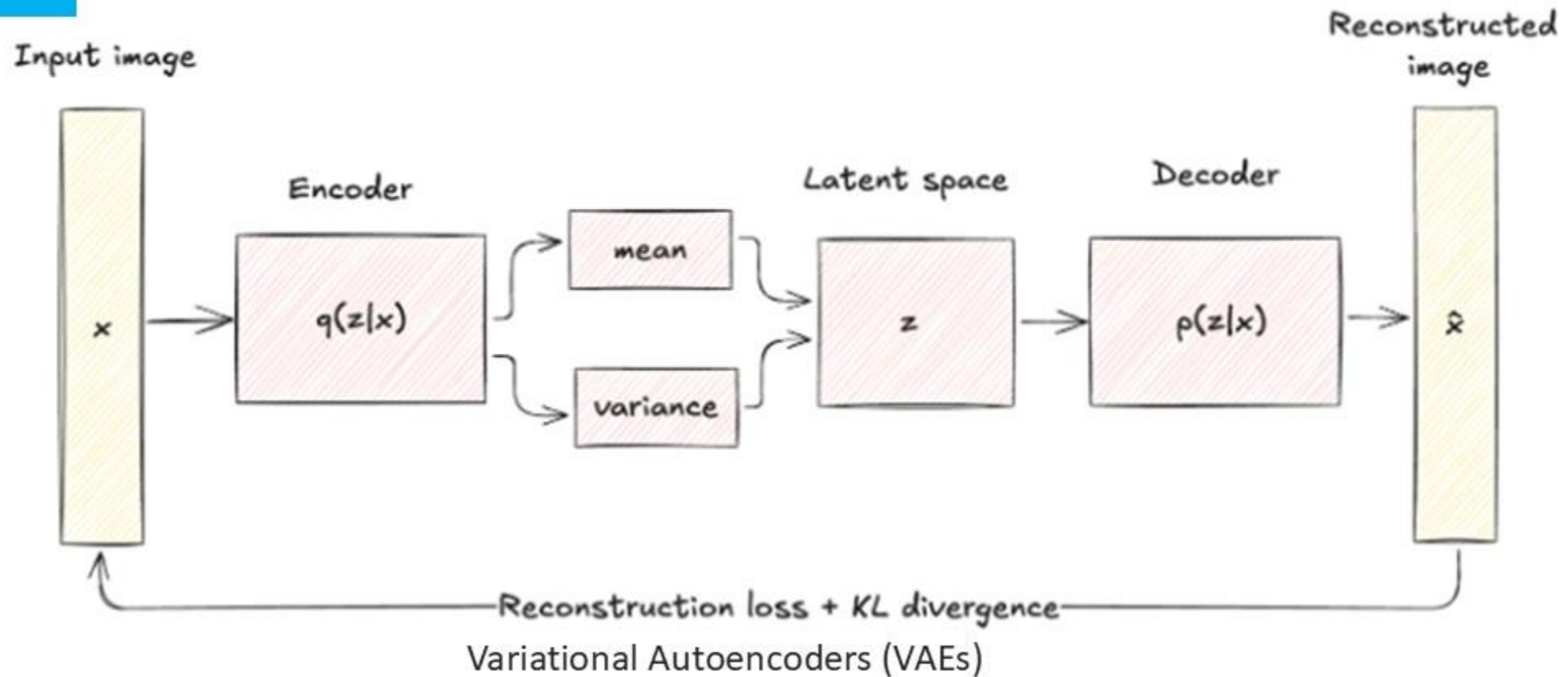
Chapter
03

Types of Autoencoders (Variational Autoencoder)



Variational Autoencoders (VAEs)

Types of Autoencoders (Variational Autoencoder)



Types of Autoencoders (Variational Autoencoder)

Advantages

VE used to generate new data points that resemble the original training data. These samples are learned from the latent space.

Variational Autoencoder is probabilistic framework that is used to learn a compressed representation of the data that captures its underlying structure and variations, so it is useful in detecting anomalies and data exploration.

Disadvantages

VA use approximations to estimate the true distribution of the latent variables. This approximation introduces some level of error, which can affect the quality of generated samples. The generated samples may only cover a limited subset of the true data distribution. This can result in a lack of diversity in generated samples.

Architecture of Autoencoder

In the most basic form of an autoencoder, the encoder and decoder are typically composed of fully connected layers (MLP or CNN). The objective of training an autoencoder is to minimize the difference between the input data and its reconstructed output, which is typically measured using a loss function such as mean squared error

$$Loss_{MSE} = \frac{1}{n} \sum_{i=1}^n (x_i - \hat{x}_i)^2$$

Where x_i is the input image and $\hat{x}_i$ is the reconstructed one.

Variational Autoencoder

Variational Autoencoders (VAEs) are a type of autoencoder that was introduced to overcome some limitations of traditional AE. VAEs extend the traditional AE architecture by introducing a probabilistic framework for generating the compressed representation of the input data.

Variational Autoencoder

In VAEs, the encoder still maps the input data to a lower-dimensional latent space, but instead of a single point in the latent space, the encoder generates a probability distribution over the latent space. The decoder then samples from this distribution to generate a new data point. This probabilistic approach to encoding the input allows VAEs to learn a more structured and continuous latent space representation, which is useful for generative modeling and data synthesis.

Variational Autoencoder

To go from a traditional autoencoder to a VAE, we need to make two key modifications. First, we need to replace the encoder's output with a probability distribution. Instead of the encoder outputting a point in the latent space, it outputs the parameters of a probability distribution, such as mean and variance. This distribution is typically a multivariate Gaussian distribution but can be some other distribution as well (e.g., Bernoulli).

Variational Autoencoder

Second, VAEs introduce a new term in the loss function called the *Kullback-Leibler (KL) divergence*. This term measures the difference between the learned probability distribution over the latent space and a predefined prior distribution (usually a standard normal distribution).

Kullback-Leibler divergence metric (relative entropy) is a statistical measurement from information theory that is commonly used to quantify the difference between one probability distribution from a reference probability distribution.

Kullback-Leibler divergence

KL divergence is defined as the number of bits required to convert one distribution into another. The lower bound value is zero and is achieved when the distributions under observation are identical.

It is often denoted with the following notation " $D_{KL}(P||Q)$ "

Kullback-Leibler divergence

$$D_{\text{KL}}(P \parallel Q) = \sum_{x \in \mathcal{X}} P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

In the formula above:

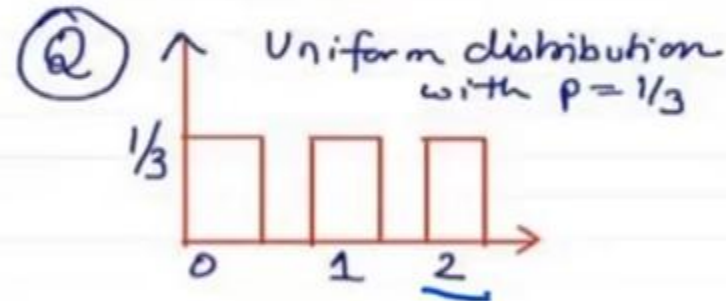
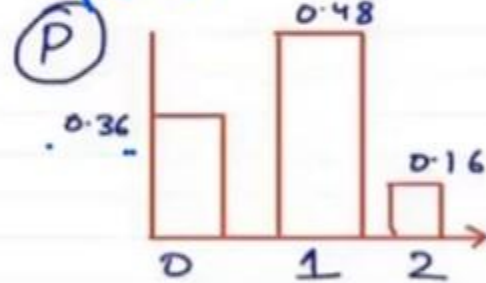
- x is a possible event in the probability space.
- $P(x)$ and $Q(x)$ are the probabilities of x in the distribution of P and Q , respectively.
- The ratio $P(x)/Q(x)$ represents the relative likelihood of event " x " according to P compared to Q .

Kullback-Leibler divergence (Example)

$$D_{KL}(P \parallel Q) = \sum_x P(X=x) \log \left(\frac{P(X=x)}{Q(X=x)} \right)$$

$$Q(x) \log \frac{Q(x)}{P(x)} = \sum_x P(x) \log \left(\frac{P(x)}{Q(x)} \right)$$

Example:



$$D_{KL}(Q \parallel P) = \frac{1}{3} \ln \left(\frac{0.333}{0.36} \right) + \frac{1}{3} \ln \left(\frac{0.333}{0.48} \right) + \frac{1}{3} \ln \left(\frac{0.333}{0.16} \right)$$

$$= 0.09637 \text{ nats}$$

Variational Autoencoder

The KL divergence term ensures that the learned distribution over the latent space is close to the prior distribution, which helps regularize the model and ensures that the latent space has a meaningful structure.

$$\log p_{\theta} (x^{(i)}) \geq \mathcal{L} (x^{(i)}, \theta, \phi) = \underbrace{\mathbb{E}_z \left[\log p_{\theta} (x^{(i)} | z) \right]}_{\text{Reconstruct the Input Data}} - \underbrace{D_{KL} \left(q_{\phi} (z | x^{(i)}) || p_{\theta} (z) \right)}_{\text{KL Divergence}}$$

$$\theta^*, \phi^* = \arg \max_{\theta, \phi} \sum_{i=1}^N \mathcal{L} (x^{(i)}, \theta, \phi)$$

How Does a VAE Work?

1.The Distribution Trick: Instead of a point in the latent space, the encoder of a VAE outputs the parameters that define a probability distribution (usually mean and variance). During training, we sample a point from this distribution to feed into the decoder.

2.The Reparameterization Trick: This is the clever part. Directly backpropagating gradients through random sampling is tricky. The reparameterization trick lets us express the sampled point in the latent space as a deterministic function of the distribution parameters and an external random variable. This allows for proper training.

How Does a VAE Work?

1. Loss Function:

Beyond Reconstruction: The VAE's loss function has two parts:

- **Reconstruction Loss:** Just like in an autoencoder, this part ensures that the decoder accurately reconstructs the input.
- **KL Divergence:** This is where the probabilistic twist comes in. The Kullback-Leibler (KL) divergence measures how much the encoder's learned distribution diverges from a standard prior distribution (often a standard Gaussian). This encourages a well-structured, regular latent space.

How Do You Calculate KL Divergence?

KL divergence is a non-symmetric metric that measures the relative entropy or difference in information represented by two distributions. It can be thought of as measuring the distance between two data distributions showing how different the two distributions are from each other.

KL Divergence is utilized to ensure that input or output data in production doesn't drastically change from a baseline. The baseline can be a training production window of data or a training or validation dataset.

The Reparameterization Trick

The core difficulty lies in the concept of backpropagation, a training technique in neural networks. Backpropagation allows us to adjust the network's internal parameters based on the difference between the predicted output and the actual output. However, it struggles when dealing with random sampling, which is inherent in VAEs.

The Reparameterization Trick

The Traditional (Broken) Approach

Imagine the encoder outputs the mean (μ) and standard deviation (σ) of a Gaussian distribution. We then sample a random variable from the distribution as:

$z = \text{random_sample}(\mu, \sigma)$

This seems straightforward, but there's a catch. Backpropagation requires calculating the gradients with respect to the model's parameters (μ and σ in this case). However, calculating the gradient with respect to z (a randomly sampled variable) is mathematically undefined. This throws a wrench in the training process.

The reparameterization trick offers an elegant solution

Instead of directly sampling z from its distribution $N(\mu, \sigma^2)$, we **decompose z into a deterministic component and a stochastic component** that is independent of the parameters we want to optimize.

Here's the magic:

1. Introduce a new random variable: We introduce another independent random variable, typically another standard normal variable (ϵ). This new variable (ϵ) serves as a source of randomness entirely separate from the sampling process within the VAE.

The Reparameterization Trick

2. Reparametrize the latent variable: Instead of directly adding noise, we create a scaled version of the new random variable (ϵ) using the standard deviation (σ) predicted by the encoder. We achieve this with element-wise multiplication:

$$z = \mu + \sigma * \epsilon$$

This equation might seem very similar to the original equation, but there's a crucial difference. Now, both μ and σ are deterministic outputs from the encoder, and ϵ is a separate source of randomness we can control.

The Reparameterization Trick

Why Does This Work?

Even though new equation introduces a new variable (ϵ), it still captures the essence of the original Gaussian distribution. The scaled ϵ effectively injects randomness while maintaining the relationship between the mean (μ) and standard deviation (σ) defined by the encoder.

Benefits:

- **Backpropagation is Happy:** Because the new equation only involves deterministic operations, we can now calculate gradients with respect to μ and σ during backpropagation. This allows the network to learn and adjust its parameters effectively.

The Reparameterization Trick

- **Preserves Stochasticity:** Even though the reparameterization trick uses a deterministic transformation, the final sampled latent variable (z) retains the desired statistical properties of the original Gaussian distribution. This ensures the VAE maintains its ability to explore the latent space effectively.

The reparameterization trick is a brilliant illustration of how a seemingly simple mathematical transformation can have a profound impact on the training process of complex neural network architectures like VAEs.

Why Use a VAE?

- **Generative Power:** The key advantage of VAEs lies in their generative capabilities. Because a VAE learns to model the underlying probability distribution of the input data, it can **generate new samples** that are similar to the original data. Think generating realistic faces, composing new pieces of music, or crafting creative text.
- **Continuous Latent Space:** Unlike traditional autoencoders, VAEs promote a continuous and smooth latent space. This means that we can interpolate between points in this space to generate smoothly transitioning variations of our data. Imagine smoothly morphing one person's face into another's!

Variational Autoencoders (VAEs) importance

The key innovation of VAEs lies in their ability to generate new, high-quality data by learning a structured, continuous latent space. This is particularly important for generative modeling, where the goal is not just to compress data but to create new data samples that resemble the original dataset.

VAEs have demonstrated significant effectiveness in tasks such as image synthesis, data denoising, and anomaly detection, making them relevant tools for advancing the capabilities of machine learning models and applications.

Applications of VAEs

- **Generative modeling.** The core advantage of VAEs is their ability to generate new data samples that are similar to the training data but not identical to any specific instance. For example, in image synthesis, VAEs can create new images that resemble the training set but with variations, making them useful for tasks like creating new artwork, generating realistic faces, or producing new designs in fashion and architecture.
- **Anomaly detection.** By learning the distribution of normal data, VAEs can identify deviations from this distribution as anomalies. This is particularly useful in applications like fraud detection, network security, and predictive maintenance.

Applications of VAEs

- **Data imputation and denoising.** One of VAEs' strong points is reconstructing data with missing or noisy parts. By sampling from the learned latent distribution, they are able to predict and fill in missing values or remove noise from corrupted data. This makes them valuable in applications such as medical imaging, where accurate data reconstruction is essential, or in restoring corrupted audio and visual data.
- **Latent space manipulation.** VAEs provide a structured and continuous latent space that can be manipulated for various applications. For instance, in image editing, specific features (like lighting or facial expressions) can be adjusted by navigating the latent space. This feature is particularly useful in creative industries for modifying and enhancing images and videos.

Applications of traditional autoencoders

- **Dimensionality reduction.** Traditional autoencoders are widely used to reduce the dimensionality of data. By encoding data into a lower-dimensional latent space and then reconstructing it, they can capture the most important features of the data. This is useful in scenarios such as data visualization, where high-dimensional data needs to be projected into two or three dimensions, and in preprocessing steps for other machine learning models to improve performance and reduce computational costs.

Applications of traditional autoencoders

- **Feature extraction.** By training the encoder to capture the essential aspects of the input data, the latent representations can be used as compact feature vectors for downstream tasks like classification, clustering, and regression. This is particularly beneficial in applications such as image recognition, where the latent space can reveal important visual patterns.

Applications of VAEs

- **Denoising.** Traditional autoencoders are effective in denoising data by learning to reconstruct clean inputs from noisy versions. This application is valuable in scenarios such as image processing, where removing noise from images can enhance visual quality, and in signal processing, where it can improve the clarity of audio signals.
- **Data compression.** The compact latent vectors can be stored or transmitted more efficiently than the original high-dimensional data, and the decoder can reconstruct the data when needed. This is particularly useful in applications like image and video compression.

Applications of VAEs

- **Image reconstruction and inpainting.** Traditional autoencoders can be used to reconstruct missing parts of images. In image inpainting, the autoencoder is trained to fill in missing or corrupted regions of an image based on the context provided by the surrounding pixels. This is useful in fields like computer vision and digital restoration.
- **Sequence learning.** Autoencoders can be adapted to work with sequential data using recurrent or convolutional layers. They can capture temporal dependencies and patterns, making them useful for applications like text generation, speech synthesis, and financial forecasting.

Conclusion

Variational autoencoders provide a fascinating blend of autoencoders and probabilistic modeling. Their ability to learn organized latent spaces and their generative prowess makes them a remarkably versatile tool in the vast landscape of deep learning.



Thank You