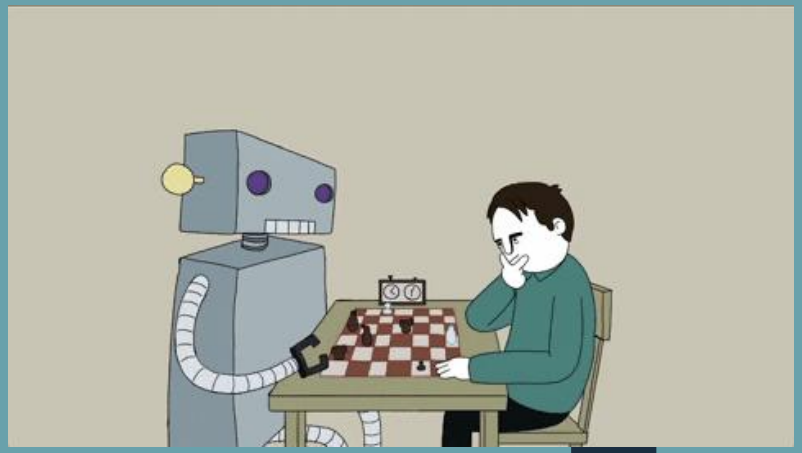
CSE4006
DEEP LEARNING



Dr K G Suma
Associate Professor
School of Computer Science and Engineering

Module No. 6
VAEs and GANS
9 Hours

- Variational Autoencoders
- Generative Adversarial Networks
- Multi-task Deep Learning
- Multi-view Deep Learning

- Various Applications - speech, text, image and video

Variational Autoencoders

- A Variational Autoencoder (VAE) is a deep learning model that generates new data by learning a **probabilistic representation** of input data.
- Unlike standard autoencoders, VAEs encode inputs into a **latent space** as probability distributions (mean and variance) rather than fixed points.
- The encoder compresses data into this space, while the decoder reconstructs it by sampling from the distribution.
- VAEs include a regularization term to ensure smooth, structured latent space, enabling realistic data generation.
- Used in image synthesis, anomaly detection, and data compression, VAEs excel in unsupervised learning by producing diverse, high-quality outputs from learned data distributions.

Variational Autoencoders

Autoencoders are a type of neural network designed to learn efficient data representations, primarily for the purpose of **dimensionality reduction** or **feature learning**.

Autoencoders consist of two main parts:

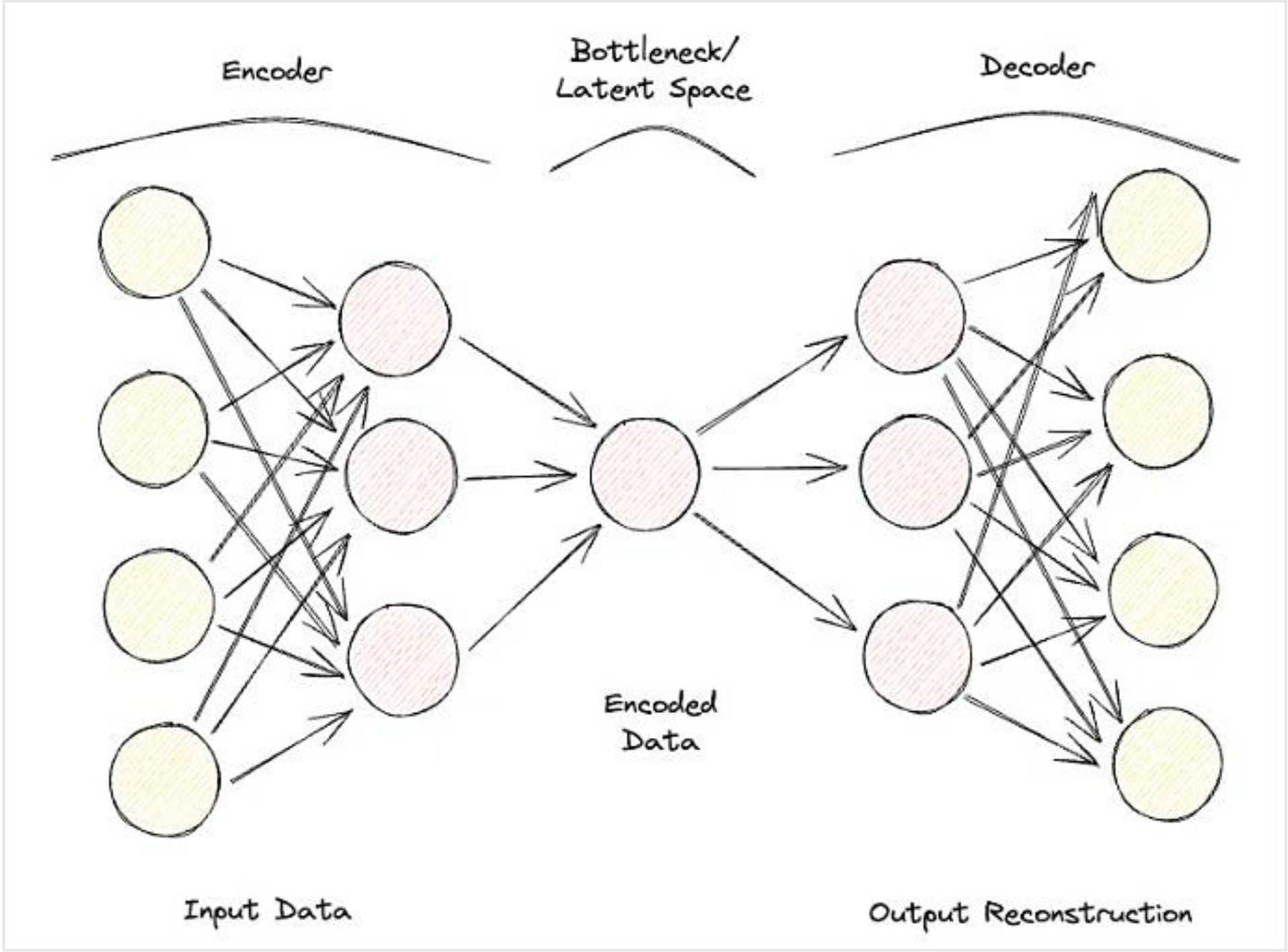
1. **The encoder**: Compresses the input data into a lower-dimensional latent space.
2. **The decoder**: Reconstructs the original data from this compressed representation.

Variational Autoencoders

The primary objective of autoencoders is to minimize the difference between the input and the reconstructed output, thus learning a compact representation of the data.

While standard autoencoders map inputs to fixed latent representations, VAEs introduce a probabilistic approach where the encoder outputs a distribution over the latent space, typically modeled as a **Gaussian distribution/ multivariate Gaussian**. This allows VAEs to sample from this distribution during the decoding process, leading to the generation of new data instances.

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.


Encoder
Bottleneck/
Latent Space
Decoder
Encoded
Data
Input Data
Output Reconstruction

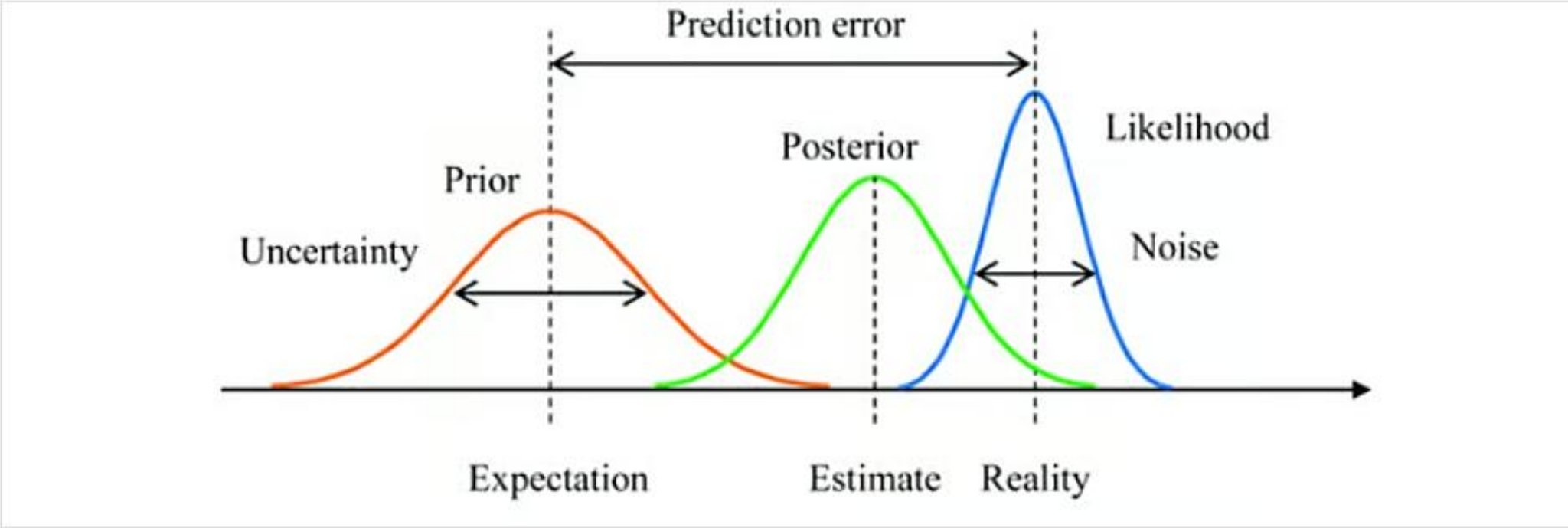
VAE with Probability distributions

The variational approach is a technique used to approximate complex probability distributions. In the context of VAEs, it involves approximating the **true posterior distribution of latent variables** given the data, which is often intractable.

The VAE learns an **approximate posterior distribution**. The goal is to make this approximation as close as possible to the true posterior.

Bayesian inference is a method of updating the probability estimate for a hypothesis as more evidence or information becomes available. In VAEs, **Bayesian inference** is used to estimate the distribution of latent variables.

By integrating prior knowledge (prior distribution) with the observed data (**likelihood**), VAEs adjust the latent space representation through the learned posterior distribution.



Bayesian inference with a prior distribution, posterior distribution, and likelihood function.

VAE Process flow

Here is how the process flow looks:

1. The input data x is fed into the encoder, which outputs the parameters of the latent space distribution $q(z/x)$ (mean μ and variance σ^2).
2. Latent variables z are sampled from the distribution $q(z/x)$ using techniques like the reparameterization trick.
3. The sampled z is passed through the decoder to produce the reconstructed data $\hat{x}$, which should be similar to the original input x.

Variational Autoencoder vs Traditional Autoencoder

Architecture comparison

Traditional autoencoders consist of an encoder network that maps the input data *x* to a fixed, lower-dimensional latent space representation *z*. This process is deterministic, meaning each input is encoded into a specific point in the latent space. The decoder network then reconstructs the original data from this fixed latent representation, aiming to minimize the difference between the input and its reconstruction.

Traditional autoencoders' latent space is a compressed representation of the input data without any probabilistic modeling, which limits their ability to generate new, diverse data since they lack a mechanism to handle uncertainty.

Variational Autoencoder vs Traditional Autoencoder

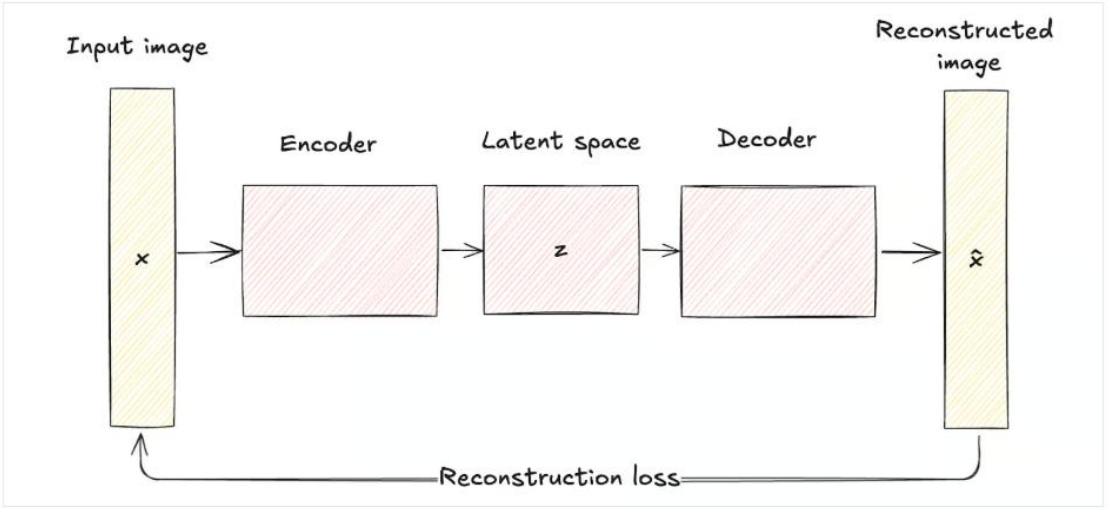
Architecture comparison

VAEs introduce a probabilistic element into the encoding process. Namely, the encoder in a VAE maps the input data to a probability distribution over the latent variables, typically modeled as a Gaussian distribution with mean μ and variance σ^2.

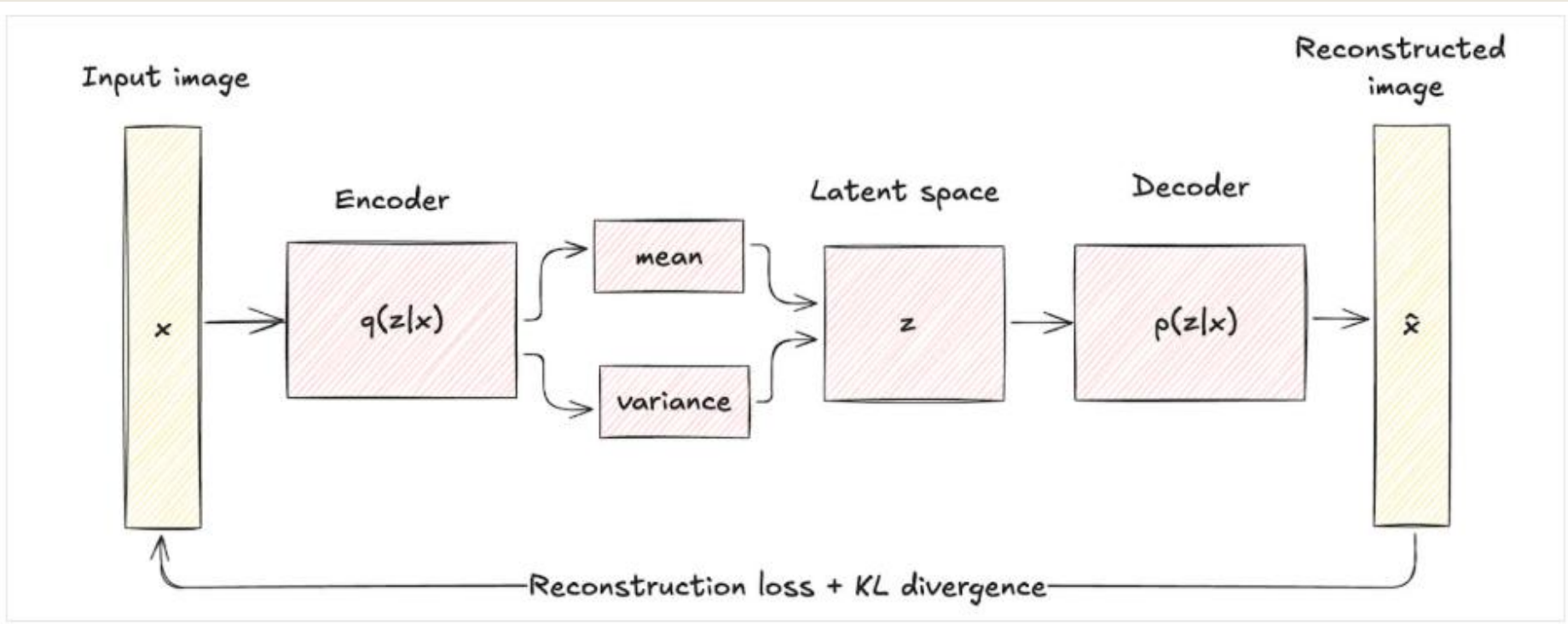
This approach encodes each input into a distribution rather than a single point, adding a layer of variability and uncertainty.

Shaping the latent space to be continuous and well-structured.

The regularization introduced significantly enhances the quality and coherence of the generated samples, surpassing the capabilities of traditional autoencoders.



Autoencoder architecture. Image by author



Variational Autoencoder architecture. Image by author

Variational Autoencoder vs Traditional Autoencoder

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.

- **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.

Variational Autoencoder vs Traditional Autoencoder

Applications of Traditional autoencoders

- **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.

Variational Autoencoder vs Traditional Autoencoder

Applications of Traditional autoencoders

- **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.

Variational Autoencoder vs Traditional Autoencoder

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.

Variational Autoencoder vs Traditional Autoencoder

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.

- **Semi-supervised learning.** In semi-supervised learning scenarios, VAEs can improve classifier performance by using the latent space to capture underlying data structures, thereby enhancing the learning process with limited labeled data.

Variational Autoencoder vs Traditional Autoencoder

Applications of VAEs

- **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.

Types of Variational Autoencoders

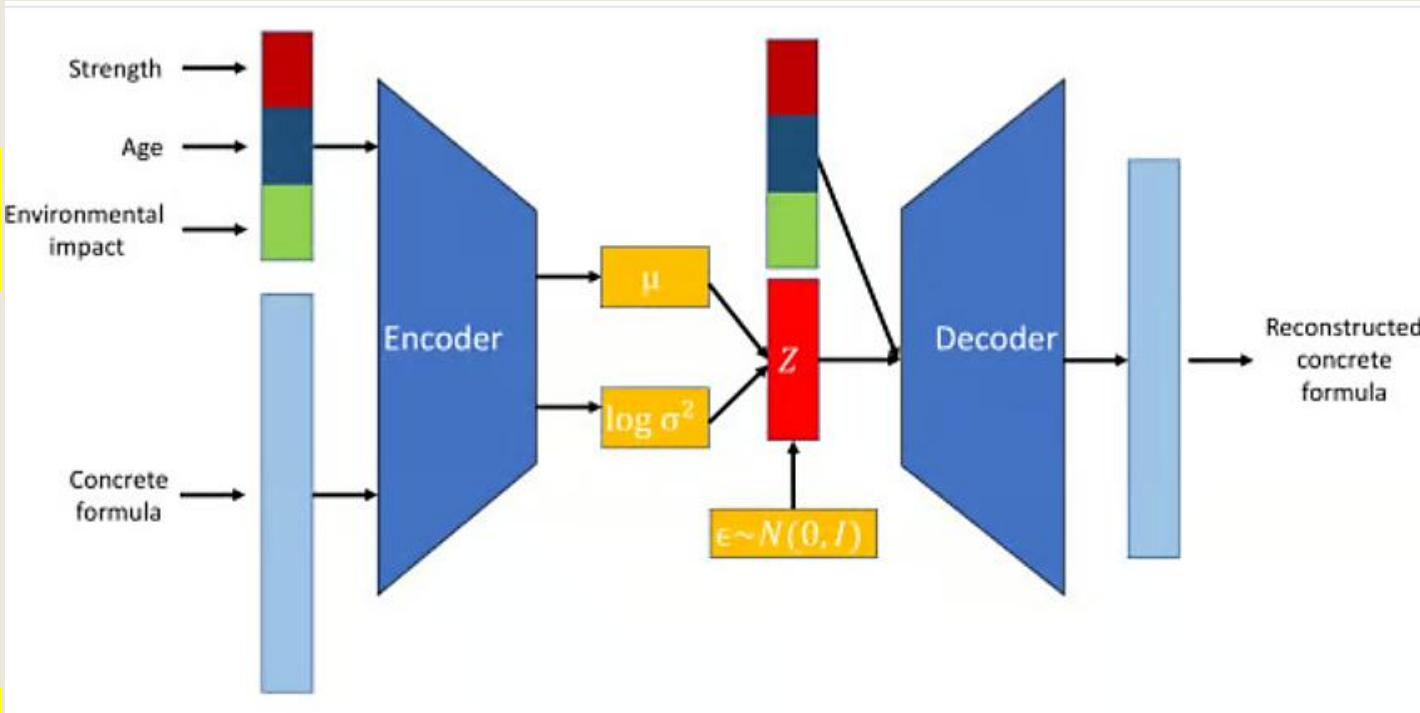
- Conditional Variational AutoEncoder (CVAE)
- Beta-VAEs
- Adversarial Autoencoders (AAEs)
- Variational Recurrent Autoencoders (VRAEs)
- Hierarchical Variational Autoencoders (HVAEs)

Types of Variational Autoencoders

Conditional variational autoencoder

Conditional Variational Autoencoders (CVAEs) are a specialized form of VAEs that enhance the generative process by conditioning on additional information.

A VAE becomes conditional by incorporating additional information, denoted as *c*, into both the encoder and decoder networks. This conditioning information can be any relevant data, such as class labels, attributes, or other contextual data.



CVAE model structure. Image source.

Conditional variational autoencoder

Use cases of CVAEs include:

- **Controlled data generation.** For example, in image generation, a CVAE can create images of specific objects or scenes based on given labels or descriptions.
- **Image-to-image translation.** CVAEs can transform images from one domain to another while maintaining specific attributes. For instance, they can be used to translate black-and-white images to color images or to convert sketches into realistic photos.
- **Text generation.** CVAEs can generate text conditioned on specific prompts or topics, making them useful for tasks like story generation, chatbot responses, and personalized content creation.

The pros and cons are:

- Finer control over generated data
- Improved representation learning
- Increased risk of overfitting

Types of Variational Autoencoders

Beta-VAEs

Disentangled Variational Autoencoders, often called Beta-VAEs, are another type of specialized VAEs. They aim to learn latent representations where each dimension captures a distinct and interpretable factor of variation in the data. This is achieved by modifying the original VAE objective with a **hyperparameter β that balances the reconstruction loss and the KL divergence* term**.

Pros and cons of Beta-VAEs:

- Improved interpretability of latent factors.
- Enhanced ability to manipulate individual features of the generated data.
- Requires careful tuning of the β parameter.
- May result in poorer reconstruction quality if the balance between terms is not optimal.

***Kullback-Leibler (KL) divergence**, also known as relative entropy, is a statistical measure that quantifies the difference between two probability distributions

Types of Variational Autoencoders

Adversarial Autoencoders (AAEs)

AAEs combine the VAE framework with adversarial training principles from Generative Adversarial Networks (GANs). An **additional discriminator** network ensures that the latent representations match a prior distribution, enhancing the model's generative capabilities.

Pros and cons of AAEs:

- Produces high-quality and realistic data samples.
- Effective in regularizing the latent space.
- Increased training complexity due to the adversarial component.
- Potential issues with training stability, similar to GANs.

Types of Variational Autoencoders

Variational Recurrent Autoencoders (VRAEs)

VRAEs extend the VAE framework to sequential data by incorporating **Recurrent Neural Networks** (RNNs) into the encoder and decoder networks. This allows VRAEs to capture temporal dependencies and model sequential patterns.

Pros and cons of VRAEs:

- Effective in handling time-series data and sequential patterns.
- Useful in applications like speech synthesis, music generation, and time-series forecasting.
- Higher computational requirements due to the recurrent nature of the model.

Types of Variational Autoencoders

Hierarchical Variational Autoencoders (HVAEs)

HVAEs introduce multiple layers of latent variables arranged in a hierarchical structure, which allows the model to **capture more complex dependencies and abstractions in the data**.

Pros and cons of HVAEs:

- Capable of modeling complex data distributions with hierarchical structures.
- Provides more expressive latent representations.
- Increased model complexity and computational cost.

Variational Autoencoders Challenges and Solutions

Mode collapse

This is a phenomenon where the VAE fails to capture the full diversity of the data distribution. The result is generated samples representing only a few modes (distinct regions) of the data distribution while ignoring others. This leads to a lack of variety in the generated outputs.

Mode collapse caused by:

- **Poor latent space exploration:** If the latent space is not adequately explored during training, the model might only learn to generate samples from a few regions.
- **Insufficient training data:** Limited or unrepresentative training data can cause the model to overfit to specific modes.

Mode collapse can be mitigated by using:

- **Regularization techniques:** Using techniques like dropout and batch normalization can help improve generalization and reduce mode collapse.
- **Improved training algorithms:** Important-weighted autoencoders (IWAE) can provide better gradient estimates and improve latent space exploration.

Variational Autoencoders Challenges and Solutions

Uninformative latent spaces

In some cases, the latent space learned by a VAE might become uninformative, where the model does not effectively use the latent variables to capture meaningful features of the input data. This can result in poor quality of generated samples and reconstructions.

This typically happens because of the following reasons:

- **Imbalanced loss components:** The trade-off between the reconstruction loss and the KL divergence might not be well-balanced, causing the latent variables to be ignored.
- **Posterior collapse:** The encoder learns to output a posterior distribution that is very close to the prior, leading to a loss of information in the latent space.

Uninformative latent spaces can be fixed by leveraging the warm-up strategy, which involves gradually increasing the weight of the KL divergence during training or by directly modifying the weight of the KL divergence term in the loss function.

Variational Autoencoders Challenges and Solutions

Training instability

Training VAEs can sometimes be unstable, with the loss function oscillating or diverging. This can make it difficult to achieve convergence and obtain a well-trained model.

The reason this occurs is because:

- **Complex loss landscape:** The VAE loss function combines reconstruction and regularization terms, leading to a complex optimization landscape.
- **Hyperparameter sensitivity:** VAEs are sensitive to the choice of hyperparameters, such as the learning rate, the weight of the KL divergence, and the architecture of the neural networks.

Steps to mitigate training instability involve either using:

- **Careful hyperparameter tuning:** Systematic exploration of hyperparameters can help find stable configurations for training.
- **Advanced optimizers:** Using adaptive learning rate optimizers like Adam can help navigate the complex loss landscape more effectively.

Variational Autoencoders Challenges and Solutions

Computational costs

Training VAEs, especially with large and complex datasets, can be computationally expensive. This is due to the need for sampling and backpropagation through stochastic layers.

The cause of high computational costs include:

- **Large networks:** The encoder and decoder networks can become large and deep, increasing the computational burden.
- **Latent space sampling:** Sampling from the latent space and calculating gradients through these samples can add to the computational cost.

These are some mitigation actions:

- **Model simplification:** Reducing the complexity of the encoder and decoder networks can help reduce computational costs.
- **Efficient sampling techniques:** Using more efficient sampling methods or approximations can reduce the computational load.

Conclusion

Variational Autoencoders (VAEs) have proven to be a groundbreaking advancement in the realm of machine learning and data generation.

By introducing probabilistic elements into the traditional autoencoder framework, VAEs enable the generation of new, high-quality data and provide a more structured and continuous latent space. This unique capability has opened up a wide array of applications, from generative modeling and anomaly detection to data imputation and semi-supervised learning.

How Autoencoders Work - Image Denoising

- An autoencoder is a type of neural network that learns to compress and reconstruct input data.
- It consists of an encoder that compresses the data into a lower-dimensional representation, and a decoder that reconstructs the original data from the compressed representation.
- The model is trained using unsupervised learning, aiming to minimize the difference between the input and the reconstructed output. Autoencoders are useful for tasks such as dimensionality reduction, data denoising, and anomaly detection. They are effective when working with unlabeled data and can learn meaningful representations from large datasets.
- The network is provided with original images *x*, as well as their noisy version *x*~. The network tries to reconstruct its output *x'* to be as close as possible to the original image *x*. By doing so, it learns how to denoise images.

How Autoencoders Work - Image Denoising

As depicted in the illustration, the encoder model turns the input into a small dense representation. The decoder model can be seen as a generative model which is able to generate specific features.

Both encoder and decoder networks are usually trained as a whole. The loss function penalizes the network for creating output *x'* that differs from the original input *x*.

By doing so the encoder learns to preserve as much of the relevant information needed in the limitation of the latent space, and cleverly discard irrelevant parts, e.g. noise. The decoder learns to take the compressed latent information and reconstruct it into a full error-free input.

How Autoencoders Work - Image Denoising

Let’s implement an autoencoder to denoise hand-written digits. The input is a 28x28 grey scaled image, building a 128-elements vector.

The encoder layer is responsible for transforming the input images into a compressed representation in the latent space. It consists of a series of convolutional and fully connected layers. This compressed representation contains essential features of the input images that capture their underlying patterns and structures. ReLU is used as the activation function in the encoder layer. It applies an element-wise activation function, setting the output to zero for negative inputs and leaving positive inputs unchanged.

The goal of using ReLU in the encoder layer is to introduce non-linearity, allowing the network to learn complex representations and extract important features from the input data.

How Autoencoders Work - Image Denoising

The decoder layer in the code is responsible for reconstructing the images from the compressed representation in the latent space. It mirrors the structure of the encoder layer and consists of a series of fully connected and transpose convolutional layers.

The decoder layer takes the compressed representation from the latent space and reconstructs the images by inverting the operations performed by the encoder layer.

It gradually upsamples the compressed representation using transpose convolutional layers and ultimately generates output images with the same dimensions as the input images. Sigmoid and ReLU activations are used in the decoder layer.

Sigmoid activation squashes the input values between 0 and 1, mapping the output of each neuron to a probability-like value. The goal of using sigmoid in the decoder layer is to produce reconstructed output values in the range [0, 1]. Since the input data in this code represents binary images, sigmoid is a suitable activation function for reconstructing pixel values.

How Autoencoders Work - Image Denoising

By using appropriate activation functions in the encoder and decoder layers, the autoencoder model can effectively learn to compress the input data into a lower-dimensional latent space and then reconstruct the original input data from the latent space.

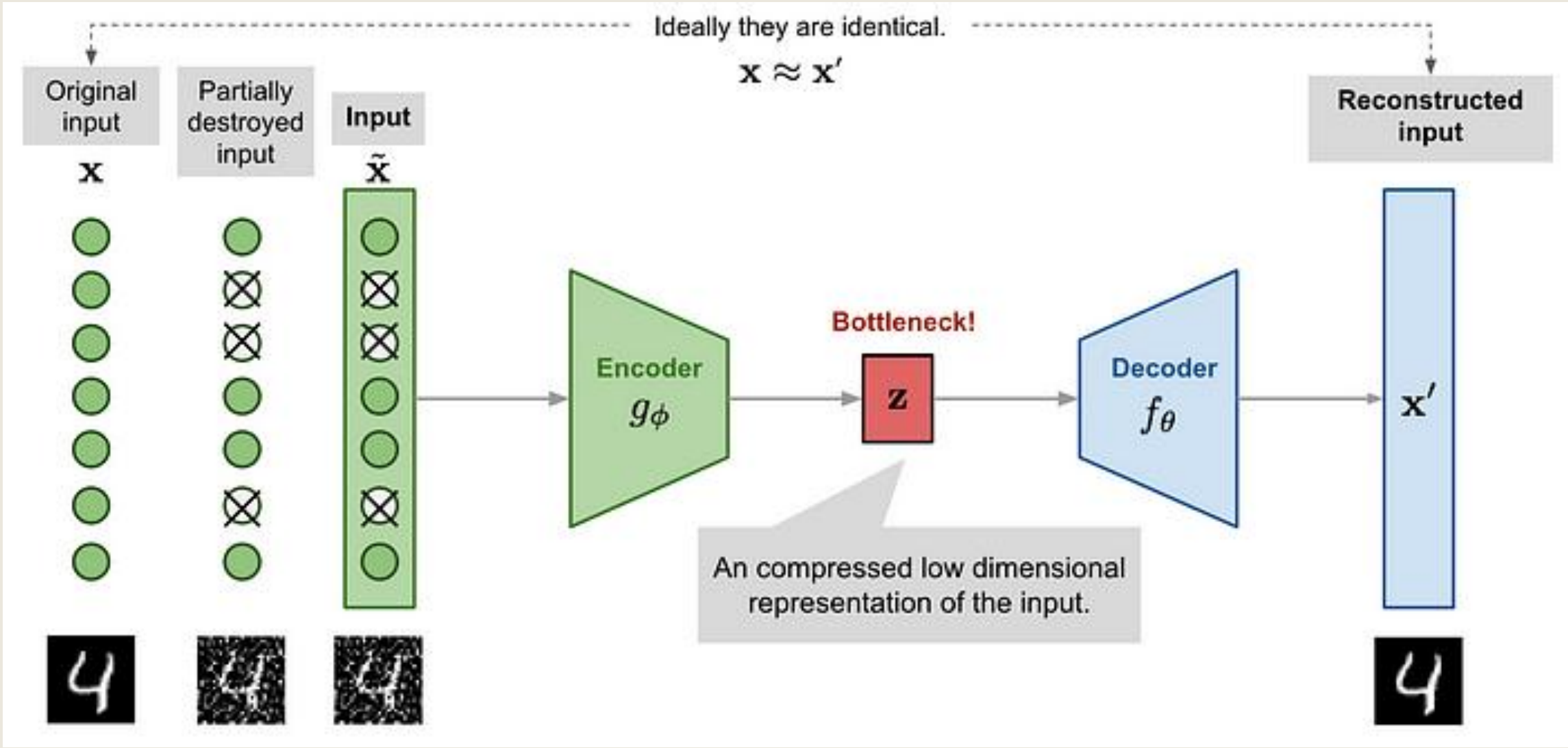
Binary cross-entropy is used as a loss function and Adam as an optimizer for minimizing the loss function.

The “binary_crossentropy” loss function is commonly used for binary classification tasks and is suitable for reconstructing binary images in this case. It measures the similarity between the predicted output and the true target output.

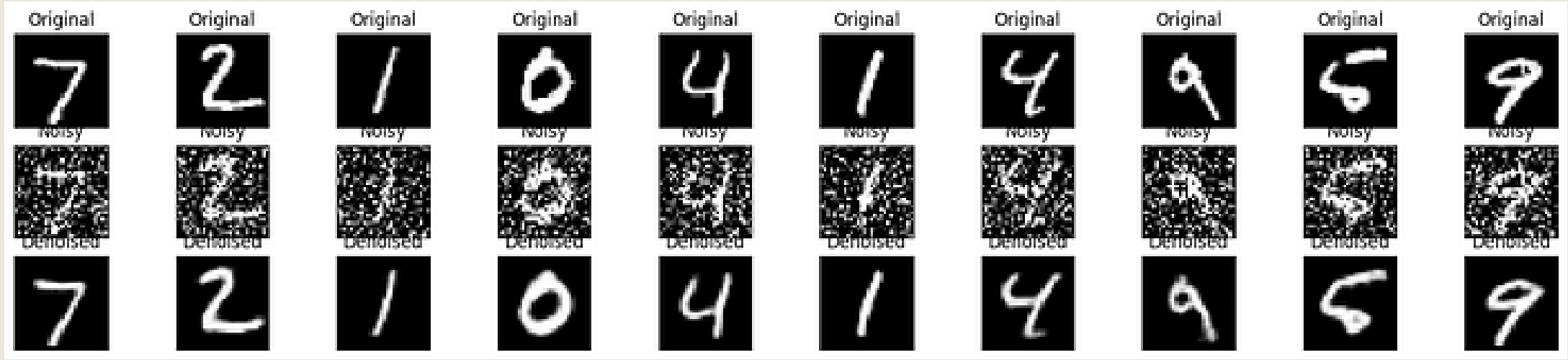
The “adam” optimizer is used to update the weights and biases of the model during training. Adam (short for Adaptive Moment Estimation) is an optimization algorithm that combines the benefits of both the RMSprop optimizer and the momentum-based optimizer. It adapts the learning rate for each weight parameter individually and uses the first and second moments of the gradients to update the parameters efficiently.

the autoencoder model aims to minimize the reconstruction error and optimize the model’s parameters to generate accurate reconstructions of the input data.

How Autoencoders Work - Image Denoising



How Autoencoders Work - Image Denoising



How Autoencoders Work - Image Denoising

- Part 1: Importing Libraries and Modules
- Part 2: Loading and Preprocessing the Dataset
- Part 3: Preprocessing the Dataset
- Part 4: Adding Random Noise to the Training Set
- Part 5: Creating the Autoencoder Model
- Part 6: Compiling the Autoencoder Model
- Part 7: Adding Early Stopping
- Part 8: Training the Autoencoder
- Part 9: Denoising Test Images and Displaying Results

Mathematical Formulation

Latent Variables:

The inclusion of keywords such as “latent distribution,” “latent variable z,” “deep generative models,” and “random variable” is pivotal in facilitating their incorporation within a model structured around a simpler (usually exponential) conditional distribution pertaining to the observable variable. This setup revolves around a probability distribution involving two variables: p(x, z). While the variable x is readily observable in the dataset being analyzed, the variable z remains concealed.

The overall probability distribution can be expressed as $p(x, z) = p(x|z)p(z)$.

Observed Variables:

- We have an observed variable x, which is assumed to follow a likelihood distribution $p(x|z)$ (for example, a Bernoulli distribution).

Mathematical Formulation

Likelihood Distribution

L(x, z) is a function that depends on two variables. If we set the value of x, the likelihood function can be understood as a distribution representing the probability distribution of z for that particular fixed x. However, if we set the value of z, the likelihood function should not be regarded as a distribution for x. In most cases, it does not adhere to the characteristics of a distribution, such as summing up to 1. Nevertheless, certain scenarios exist where the likelihood function can formally meet the distribution criteria and satisfy the requirement of summing to 1.

- The main objective of a Variational Autoencoder (VAE) is to learn the true posterior distribution of the latent variables: $p(z|x)$
- A VAE achieves this by using an encoder network to approximate the true posterior with a learned distribution: $q(z|x)$

Mathematical Formulation

Posterior Distribution

In Bayesian statistics, a posterior probability refers to the adjusted or updated probability of an event happening in light of newly acquired information. Update the prior probability by applying Bayes' theorem to calculate the posterior probability.

The Variational Autoencoder (VAE) learns model parameters by maximizing the Evidence Lower Bound (ELBO): **ELBO = E[log(p(x|z))] – KL(q(z|x) || p(z))**

The ELBO consists of **two main components**:

1. **Reconstruction term**: **E[log(p(x|z))]** Measures how well the VAE can **reconstruct the input data** from the latent representation.
2. **KL divergence term**: **KL(q(z|x) || p(z))** Quantifies how much the learned posterior **q(z|x)** deviates from the prior **p(z)**. Using a probabilistic framework, VAEs generate data by assuming that the input originates from a latent space governed by specific probability distributions. The key goal is to learn the true posterior distribution by maximizing the likelihood of the observed input data.

Mathematical Formulation

Variational Inference Formulation

The formulation of Variational Inference in a VAE is as follows:

- **Approximate posterior distribution:** We have an approximation of the posterior distribution q(z|x).
- **True posterior distribution:** We have the true posterior distribution p(z|x).

The aim is to find a similar distribution (q(z|x)) that approximates the true distribution (p(z|x)) as closely as possible, using the KL divergence method.

The KL variance equation compares two probability distributions, **q(z|x) and p(z|x)**, to measure their differences.

During VAE training, we try to minimize the KL divergence by increasing the evidence of lower boundary (ELBO), a combination of the reconstruction term and the KL divergence. The reconstruction term assesses the model's ability to reconstruct input data, while the KL divergence measures the difference between the approximate and actual distributions.

Mathematical Formulation

Neural Networks in the Model

- Neural networks are commonly used to implement VAEs, where both the encoder and decoder components are implemented as neural networks. During the training process, the VAE adjusts the parameters of the encoder and decoder networks to minimize two key components: the reconstruction error and the KL divergence between the variational distribution and the true posterior distribution. This optimization task is often accomplished using techniques like stochastic gradient descent or other suitable optimization algorithms.

Latent Space - Key Concepts

Dimensionality Reduction: Latent space is often created by reducing the number of dimensions in the original data. This process compresses the data while retaining the most important information.

Feature Extraction: The latent space captures the underlying features and relationships within the data.

Representation Learning: Neural networks learn to encode the input data into this latent space, creating a representation that is useful for various downstream tasks like classification, regression, or generation.

Embedding Space: Latent space is often used interchangeably with embedding space, which refers to the space where data points are represented as vectors, with similar data points being closer together.

Manifolds: In some cases, latent space can be visualized as a manifold, which is a lower-dimensional subspace that represents the underlying structure of the data.

Latent Space - Working

Encoding:
The input data is passed through a neural network, which transforms it into a lower-dimensional representation in the latent space.

Latent Space Operations:
Once in the latent space, the model can perform various operations, such as calculating distances between data points, finding patterns, or generating new data.

Decoding:
The representation in the latent space can be decoded back into the original data format, allowing the model to reconstruct or generate new instances.

Latent Space - Examples

Autoencoders:
Autoencoders are neural networks designed to learn a compressed representation of the input data in the latent space and then reconstruct the original data from this representation.

Variational Autoencoders (VAEs):
VAEs are a type of autoencoder that learns a probabilistic distribution over the latent space, allowing for the generation of new data samples.

Generative Adversarial Networks (GANs):
GANs also utilize latent space to generate new data, where the latent space is used as a seed for generating images, text, or other types of data.