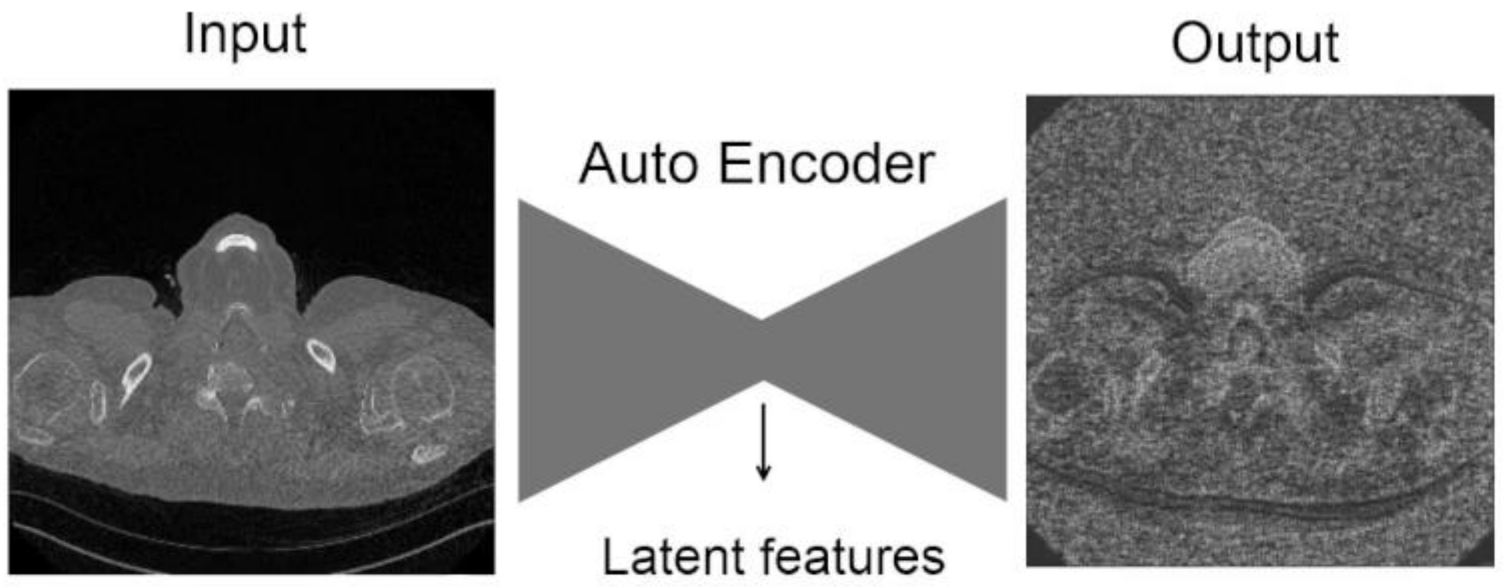
Module - 6

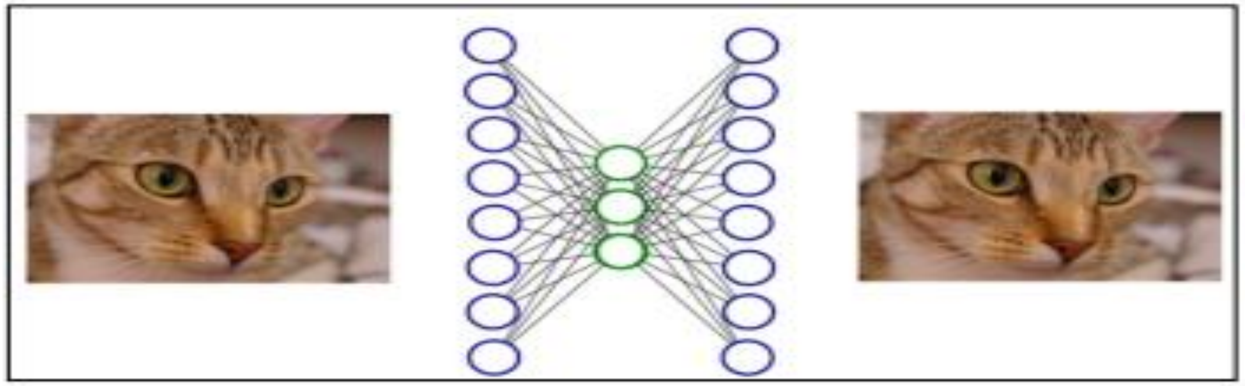
Autoencoders

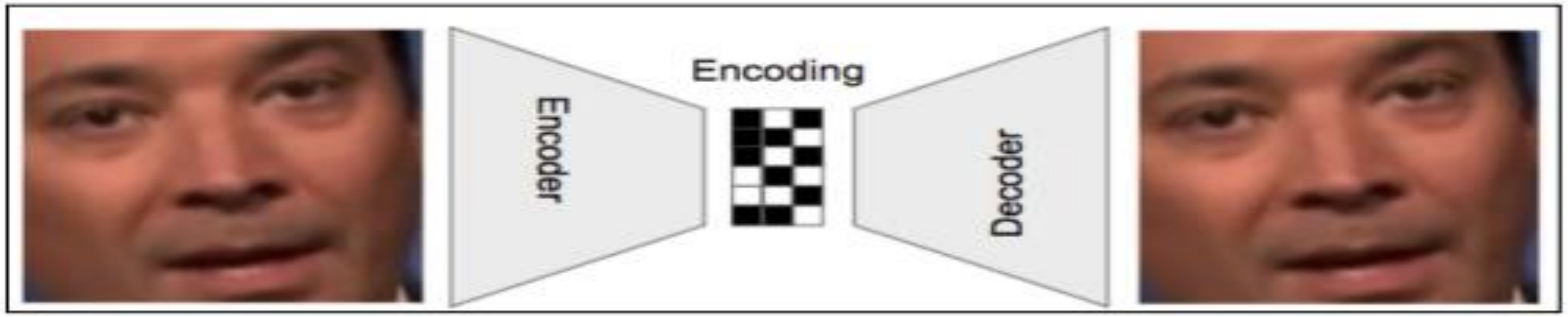


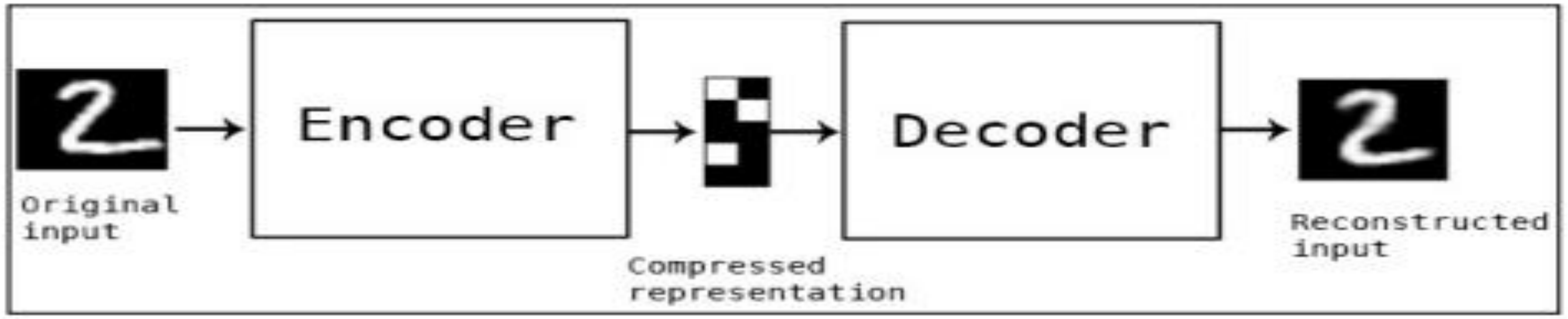
Autoencoders

- Autoencoders are self-supervised machine learning models which are used to reduce the size of input data by recreating it.
- These models are trained as supervised machine learning models and during inference, they work as unsupervised models that's why they are called self-supervised models.
- Autoencoders are very useful in the field of unsupervised machine learning. You can use them to compress the data and reduce its dimensionality.
- The aim of an autoencoder is to learn a lower-dimensional representation (encoding) for higher-dimensional data, typically for dimensionality reduction, by training the network to capture the most important parts of the input image.

Autoencoders







Applications of Autoencoders

- **1. File Compression**: Primary use of Autoencoders is that they can reduce the dimensionality of input data which we in common refer to as file compression. Autoencoders works with all kinds of data like Images, Videos, and Audio, this helps in sharing and viewing data faster than we could do with its original file size.
- **2. Image De-noising**: Autoencoders are also used as noise removal techniques (Image De-noising), what makes it the best choice for De-noising is that it does not require any human interaction, once trained on any kind of data it can reproduce that data with less noise than the original image.
- **3. Image Transformation**: Autoencoders are also used for image transformations, which is typically classified under GAN(Generative Adversarial Networks) models. Using these we can transform B/W images to colored one and vice versa, we can up-sample and down-sample the input data, etc.

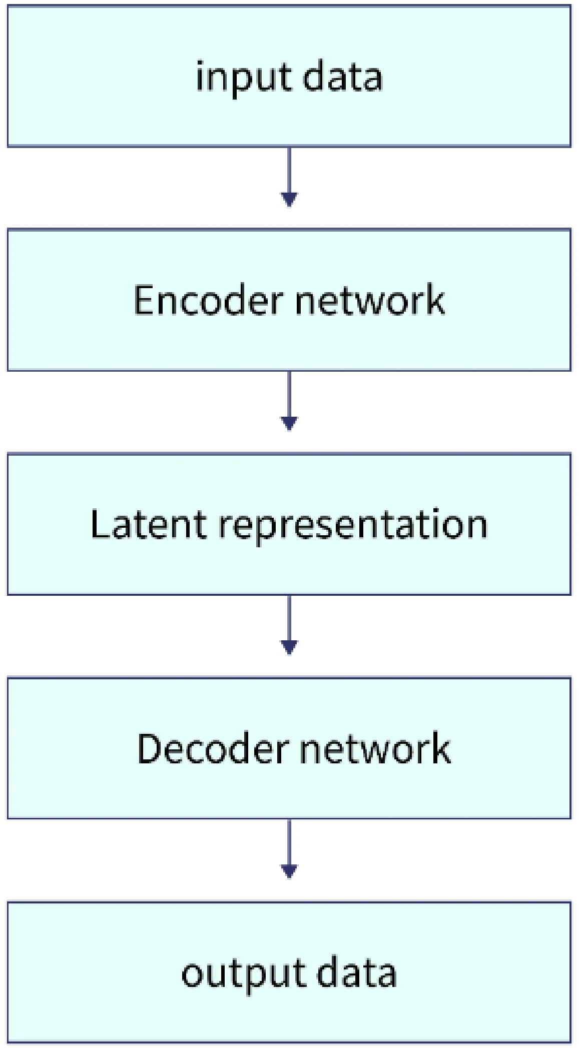
Normal Data Compression Vs Autoencoders

- Data compression is reducing the amount of data needed to represent a given piece of information.

- Autoencoders can be used for data compression by learning a compressed representation of the input data through the encoder and then reconstructing the original data through the decoder.

- Autoencoders are data-specific
  - i.e., only able to compress data similar to what they have been trained on.
  - This is different from, say, MP3 or JPEG compression algorithm-Which make general assumptions about "sound/images", but not about specific types of sounds/images.
  - Autoencoder for pictures of cats would do poorly in compressing pictures of trees Because features it would learn would be cat-specific.
- Autoencoders are lossy - which means that the decompressed outputs will be degraded compared to the original inputs (similar to MP3 or JPEG compression). This differs from lossless arithmetic compression
- Autoencoders are learnt

Architecture of Autoencoders



- The architecture of an autoencoder typically consists of two parts: an encoder and a decoder.
- The encoder maps the input data to a lower-dimensional latent space, and the decoder maps the latent representation back to the original data space.
- The encoder and decoder can be implemented as feedforward neural networks. The input to the encoder is the data you want to compress, and the output is the latent representation.
- The input to the decoder is the latent representation, and the output is the reconstructed data.

Architecture of Autoencoders

Encoder

- The encoder is a set of convolutional blocks followed by pooling modules that compress the input to the model into a compact section called the bottleneck.
- The bottleneck is followed by the decoder which consists of a series of upsampling modules to bring the compressed feature back into the form of an image.
- In case of simple autoencoders, the output is expected to be the same as the input with reduced noise.

Architecture of Autoencoders

Bottleneck

- The most important part of the neural network, and ironically the smallest one, is the bottleneck.
- The bottleneck exists to restrict the flow of information to the decoder from the encoder, thus, allowing only the most vital information to pass through.
- It helps us form a knowledge representation of the input because the maximum information is captured.
- Thus, the encoder-decoder structure helps us extract the most from an image in the form of data and establish useful correlations between various inputs within the network.
- A bottleneck as a compressed representation of the input further prevents the neural network from memorizing the input and overfitting the data.
- As a rule of thumb, remember this: The smaller the bottleneck, the lower the risk of overfitting.

Architecture of Autoencoders

Decoder

- Finally, the decoder is a set of upsampling and convolutional blocks that reconstructs the bottleneck's output.
- Since the input to the decoder is a compressed knowledge representation, the decoder serves as a "decompressor" and builds back the image from its latent attributes.

Autoencoders Training

- **Code size:** The code size or the size of the bottleneck is the most important hyperparameter used to tune the autoencoder. The bottleneck size decides how much the data has to be compressed. This can also act as a regularisation term.
- **Number of layers:** Like all neural networks, an important hyperparameter to tune autoencoders is the depth of the encoder and the decoder. While a higher depth increases model complexity, a lower depth is faster to process.
- **Number of nodes per layer:** The number of nodes per layer defines the weights we use per layer. Typically, the number of nodes decreases with each subsequent layer in the autoencoder as the input to each of these layers becomes smaller across the layers.

Autoencoders Training

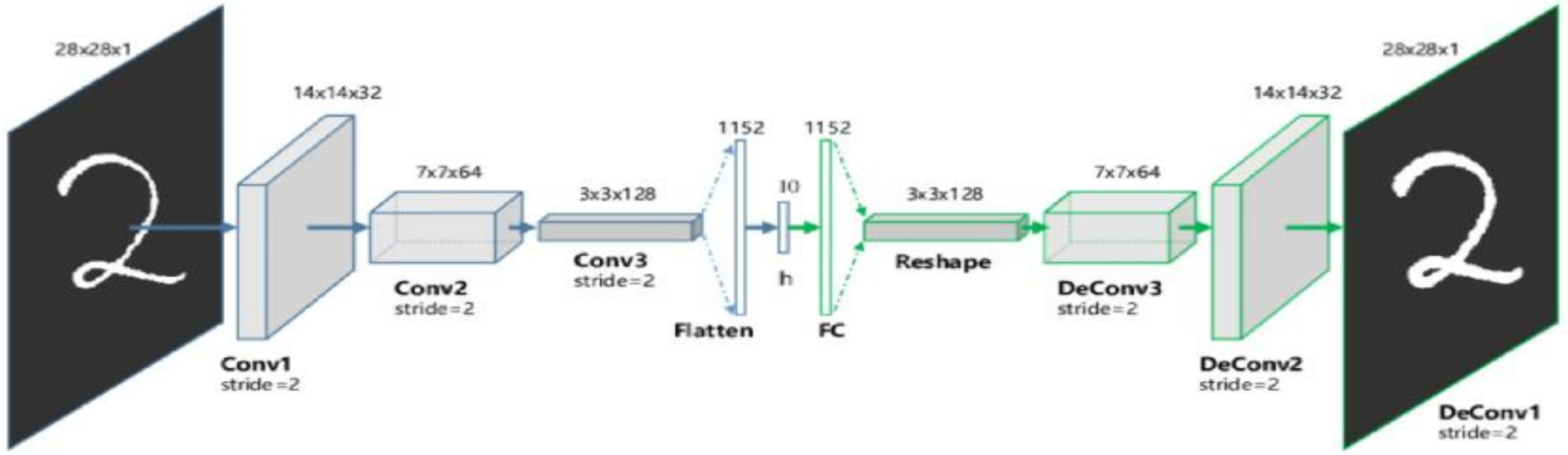
- **Reconstruction Loss:** The loss function we use to train the autoencoder is highly dependent on the type of input and output we want the autoencoder to adapt to.
- If we are working with image data, the most popular loss functions for reconstruction are MSE Loss and L1 Loss.
- In case the inputs and outputs are within the range [0,1], as in MNIST, we can also make use of Binary Cross Entropy as the reconstruction loss.

Types of Autoencoders

- Standard Autoencoder
  - This is the basic form of autoencoder, where the encoder and decoder are composed of a linear stack of fully connected (dense) layers.
- Convolutional Autoencoder
- Deep Autoencoder
- Denoising Autoencoder
- Sparse Autoencoder
- Variational Autoencoder (VAE)

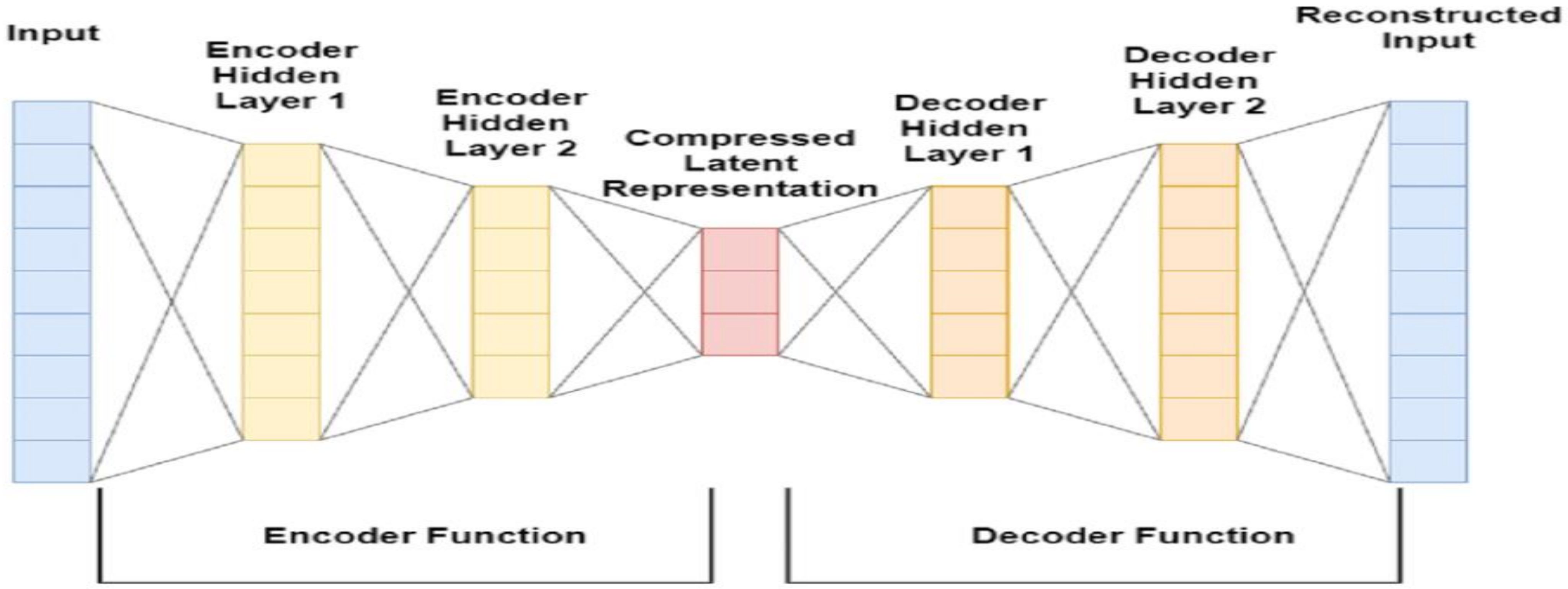
Convolutional Autoencoder

- This type of autoencoder is similar to a standard autoencoder, but the encoder and decoder use convolutional layers instead of dense layers. This is useful for processing data that has a grid-like structure, such as images.



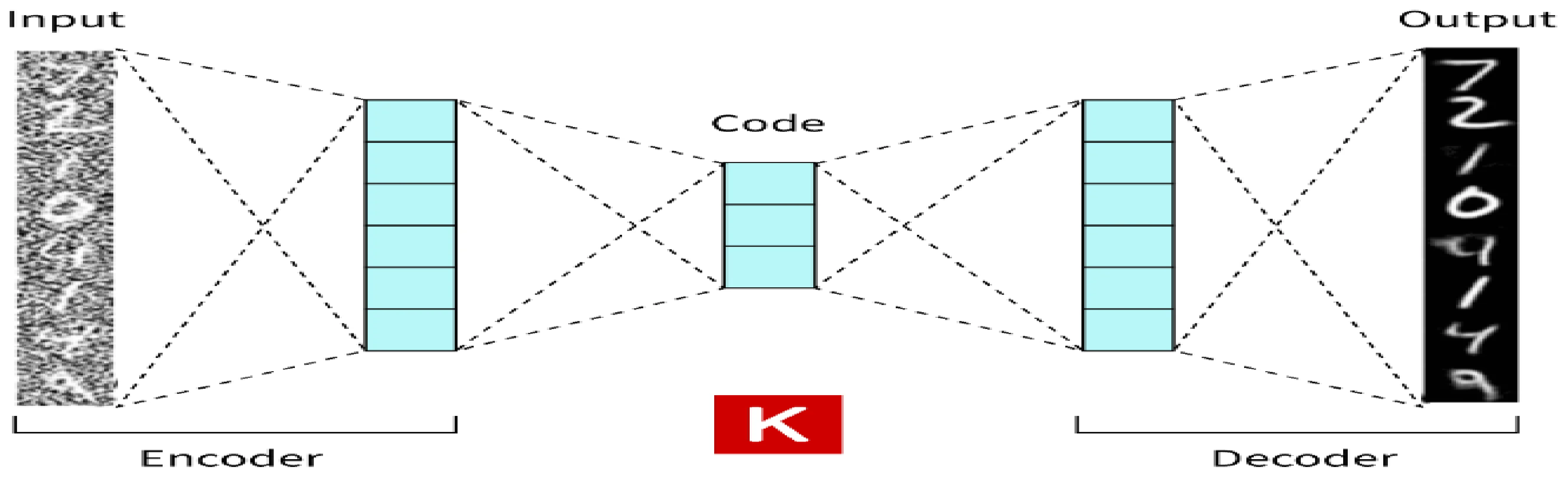
Deep Autoencoder

- This refers to an autoencoder with many layers in the encoder and decoder networks. Deep autoencoders can learn more complex and abstract data representations but are more computationally expensive to train..



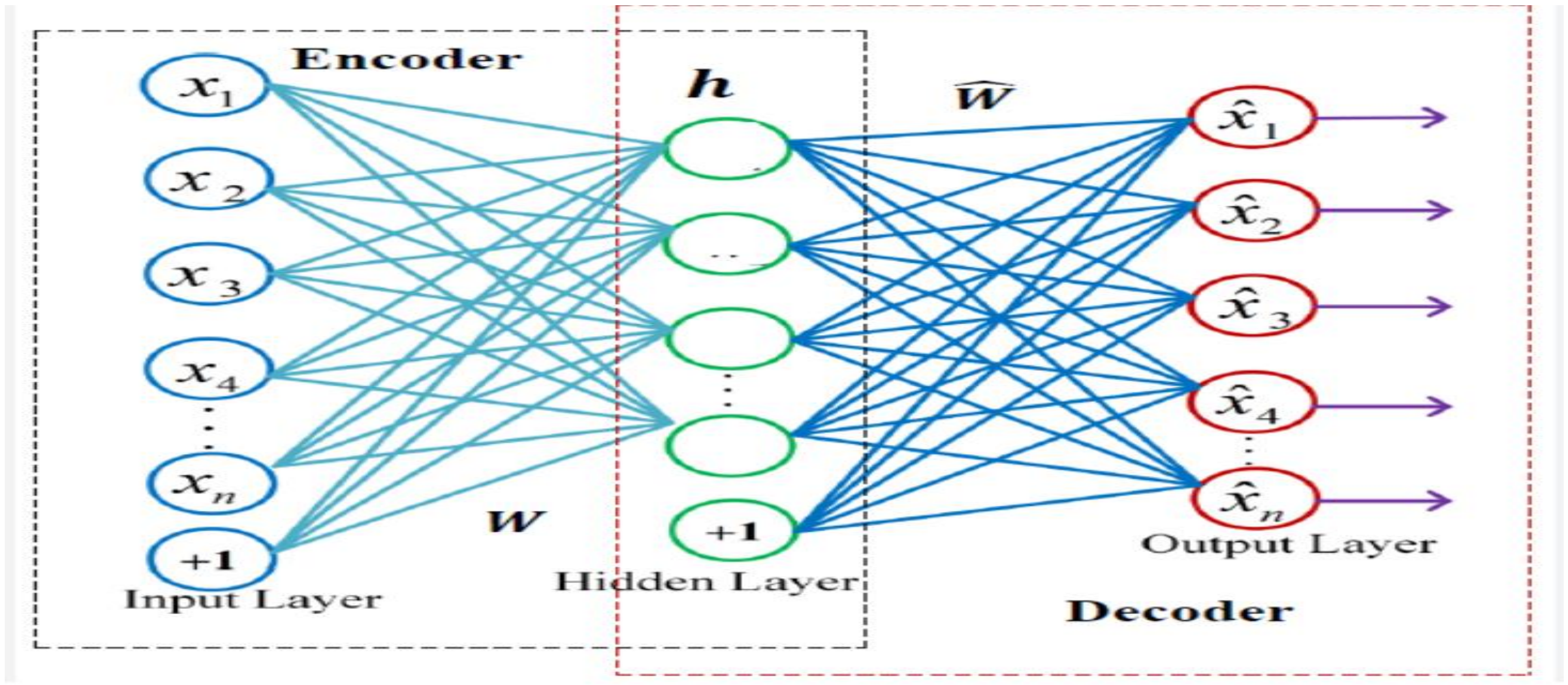
Denoising Autoencoder

- This autoencoder is trained to reconstruct the original data from a corrupted version. The encoder and decoder are still composed of dense or convolutional layers, but the goal is to learn a robust representation of noise.



Sparse Autoencoder

- This autoencoder is trained to learn a representation with a small number of active units (i.e., units with non-zero activations). The goal is to encourage the autoencoder to learn a more efficient and compact data representation.

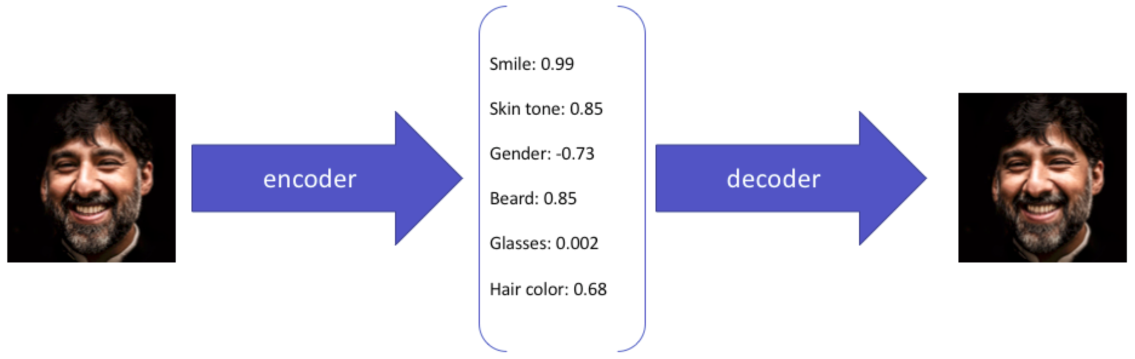


Variational Autoencoder (VAE)

- A VAE is a generative model that can generate new data samples similar to the training data.
- It consists of an encoder network and a decoder network, but the encoder outputs two values: a mean and a variance.
- These values are used to sample a latent code from a normal distribution, passed through the decoder to generate the output data.
- The VAE is trained to minimize the difference between the original data and the reconstructed data, allowing it to learn the underlying distribution of the data and generate new samples that follow that same distribution.

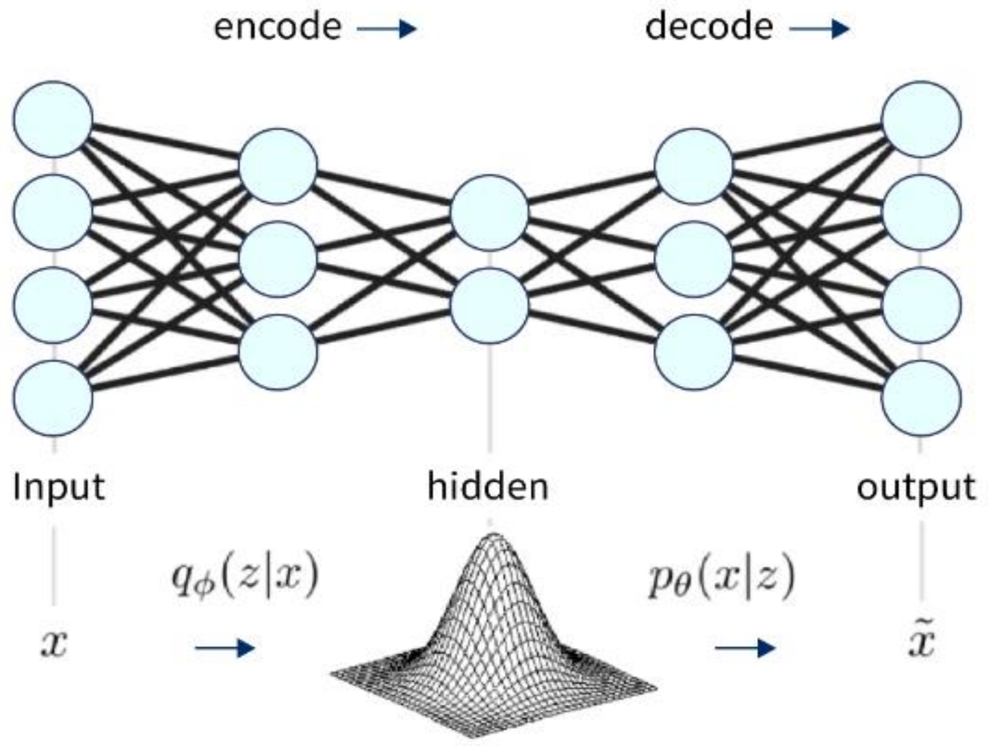
Variational Autoencoder (VAE)

- VAEs are the regularization applied to the latent code.
- In a VAE, the latent code is regularized using a Gaussian distribution with fixed mean and variance. This regularization helps to prevent overfitting by encouraging the latent code to have a smooth distribution rather than memorizing the training data.
- This regularization also allows the VAE to generate new data samples that are smoothly interpolated between the training data points. This makes VAEs a powerful tool for generating new data samples similar to the training data.



Latent attributes

Variational Autoencoder (VAE)

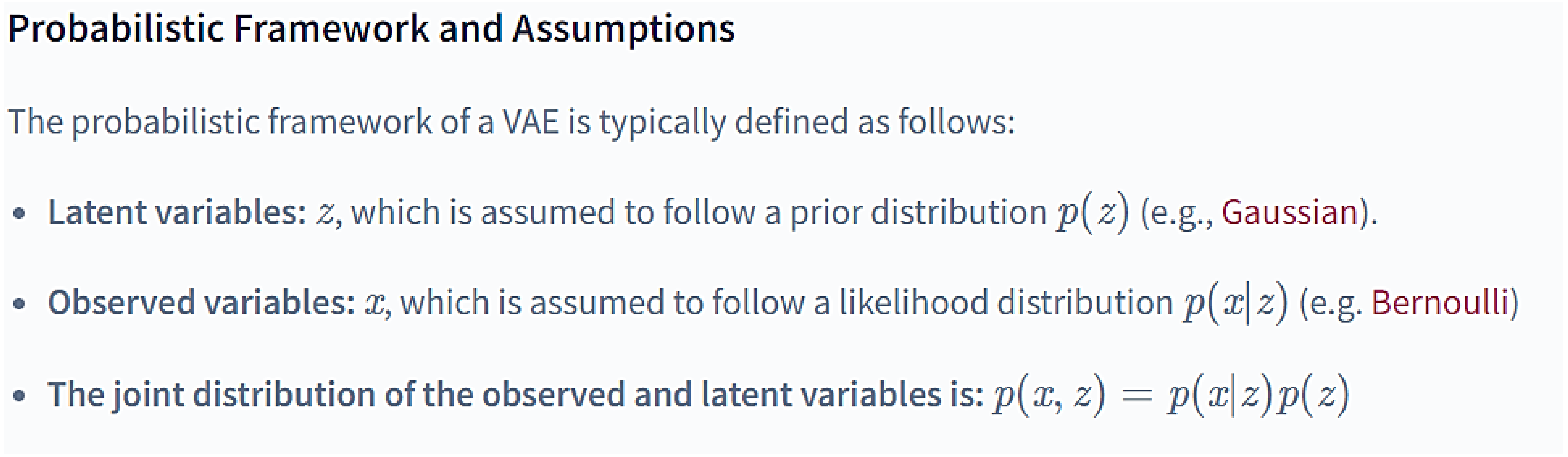


- The encoder network outputs the parameters of a **Gaussian** distribution for the latent code, typically the mean and the log-variance (or standard deviation).
- The latent code is then sampled from this Gaussian distribution.
- The KL divergence between this distribution and a prior (often assumed to be a standard normal distribution) is added as a regularization term to the VAE's loss function.

Formula for KL divergence

$$KL(q(z|x)||p(z)) = E[logq(z|x) - logp(z)]$$

Mathematical Details of VAEs



Probabilistic Framework and Assumptions

The probabilistic framework of a VAE is typically defined as follows:

- **Latent variables:** z, which is assumed to follow a prior distribution $p(z)$ (e.g., Gaussian).
- **Observed variables:** x, which is assumed to follow a likelihood distribution $p(x|z)$ (e.g. Bernoulli)
- **The joint distribution of the observed and latent variables is:** $p(x, z) = p(x|z)p(z)$

- The main goal of the VAE is to learn the true posterior distribution of the latent variables given the observed variables, which are defined as $p(z|x)$.
- The VAE uses an encoder network to approximate the true posterior distribution with a learned approximation $q(z|x)$ to achieve this.

ENCODERS VS VAE

- While in the case of autoencoders, the encoder network maps input data to a fixed point, in the case of variational autoencoders, the encoder network maps input data to a distribution (multivariate normal distribution).
- Additionally, the loss function of the variational autoencoder includes an additional KL divergence term in addition to the reconstruction loss.

Variational Autoencoders as a Generative Model