Unit - 4

Autoencoders

- Autoencoders are a type of artificial neural network used for unsupervised learning, primarily for
- **dimensionality reduction, feature learning, and data compression**.
- They are designed to encode input data into a lower-dimensional representation and then reconstruct the original data from this compressed form.
- **Dimensionality Reduction:**
- **Simplification of Data**: High-dimensional data (e.g., images, audio) can be reduced to a more manageable size while retaining important features.
- **Visualization**: It is useful for visualizing complex data in a low-dimensional space, often for exploration or analysis (similar to PCA).

- **Data Compression:**
- Autoencoders are effective at compressing data without losing much information.
- Helpful for storage and transmission purposes.
- **Noise Reduction (Denoising Autoencoders):**
- They can learn to remove noise from data, making them useful for image or audio denoising.
- **Anomaly Detection:**
- Autoencoders learn patterns in the data during training, they can be used to detect anomalies by reconstructing inputs.
- If an input deviates significantly from what the autoencoder has learned, the reconstruction error will be large, indicating a possible anomaly.

- **Feature Extraction:**
- Autoencoders can be used to learn efficient and informative feature representations from raw data,
- Helpful in classification or clustering.
- **Data Generation:**
- Autoencoders can be used to generate new data samples that are similar to the input data.
- Useful in tasks such as generating images, text, or other types of data.
- **Pretraining for Deep Networks:**
- Autoencoders can be used as a pretraining method for deep networks

- **Medical and Industrial Applications:**
- Medical imaging or manufacturing, autoencoders can be used for image reconstruction, defect detection, and pattern recognition, helping in diagnostics and quality control.
- **Learn two functions:**
- An **encoding function** that transforms input data into a lower-dimensional representation, and
- A **decoding function** that recreates the input data from the encoded representation.

Autoencoder

Key Components of an Autoencoder

Encoder:

- This part of the network compresses the input data into a lower-dimensional representation.
- Often called a "latent space" or "bottleneck."
- The encoder learns to capture the most important features of the input data while reducing its dimensionality.

Decoder:

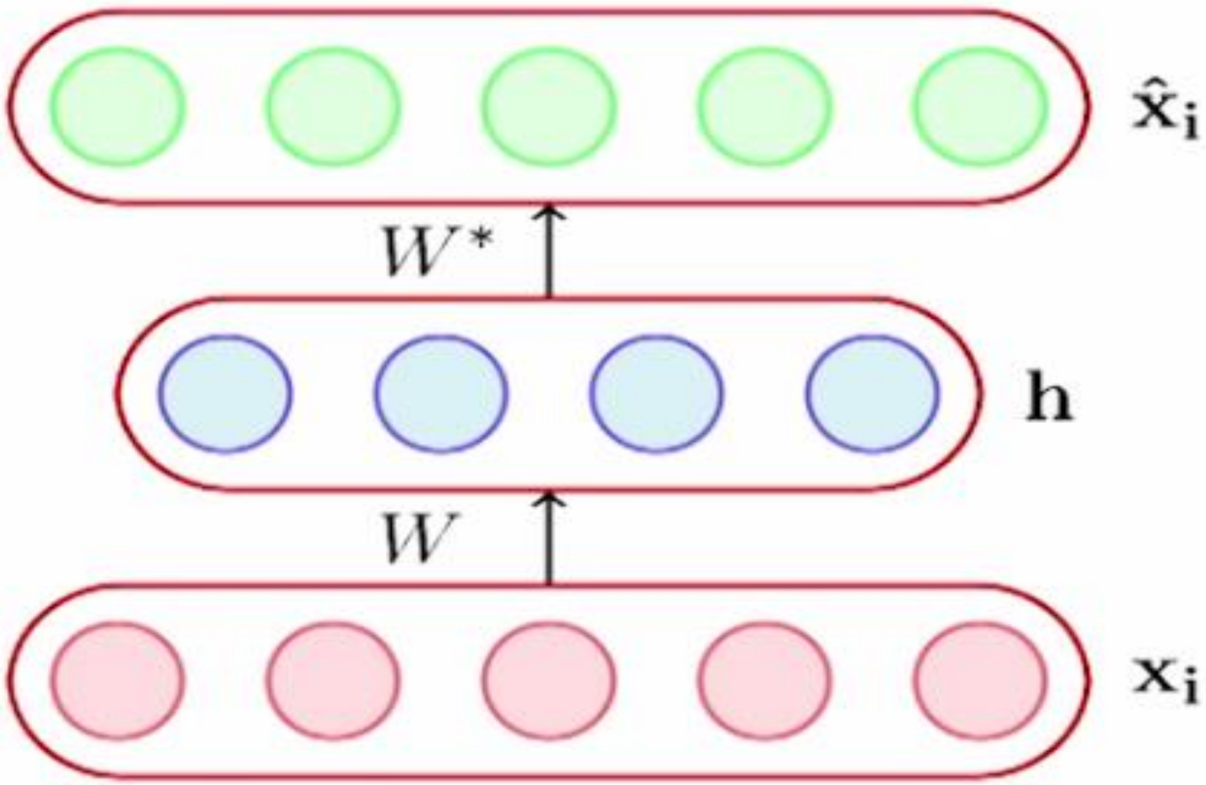
- This part of the network takes the compressed representation and attempts to reconstruct the original input data.
- The goal is to make the output of the decoder as close as possible to the input data.

- **Structure of an Autoencoder**
- **Input Layer**:
- The input layer takes the raw data, such as images, text, or numerical data.
- **Hidden Layers**:
- The encoder and decoder each have one or more hidden layers.
- These layers learn the compressed representation and the reconstruction of the data, respectively.
- **Output Layer**:
- The output layer has the same dimension as the input layer, and it represents the reconstructed version of the input.

- **Encoding Phase**:
- The input data X is passed through the encoder network, which maps it to a lower-dimensional latent space Z.
- This can be represented as:
  - Z=f(X)
- where f is a function learned by the encoder neural network.
- **Decoding Phase**:
- The latent representation Z is then passed through the decoder network, which attempts to reconstruct the input data X^ from Z.
- This can be represented as:
  - X^=g(Z)
- where g is a function learned by the decoder neural network.

- **Training Objective**:
- The autoencoder is trained to minimize the reconstruction error.
- Measures the difference between the input data X and the reconstructed data X^.
- The loss function L can be defined as:
  - $L(X,X^{\wedge})=\|X-X^{\wedge}\|2$
- The objective is to minimize this loss function during training.

Autoencoders



$$\mathbf{h} = g(W\mathbf{x}_\mathbf{i} + \mathbf{b})$$

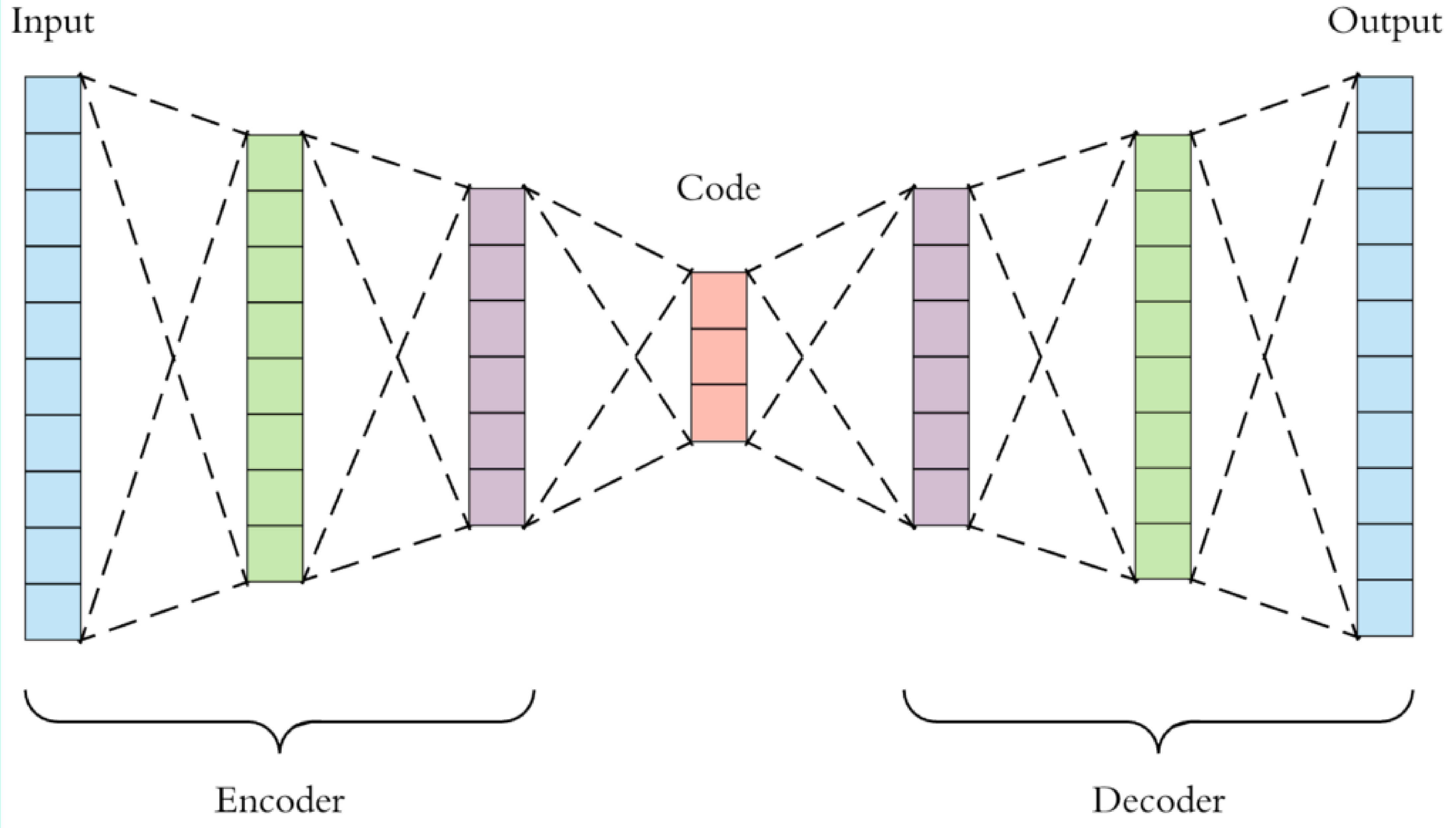
$$\hat{\mathbf{x}}_\mathbf{i} = f(W^*\mathbf{h} + \mathbf{c})$$

Types of Autoencoders:

- Under Complete Autoencoders.
- Sparse Autoencoders
- Contractive Autoencoders
- Denoising Autoencoders
- Variational Autoencoders

Under Complete Autoencoders

- Undercomplete Autoencoders are a class of neural networks designed for unsupervised learning tasks like
- dimensionality reduction, feature extraction, and data reconstruction.
- Primary function is to learn a compressed representation of input data by minimizing reconstruction error.
- **The term "undercomplete" refers to the architecture of the network, where the hidden layer (or bottleneck layer) contains fewer neurons than the input layer.**
- **Forcing the network to compress the data into a smaller, more efficient representation.**
- The objective of undercomplete autoencoder is to capture the most important features present in the data.


Input
Output
Code
Encoder
Decoder

- **Architecture of Undercomplete Autoencoders**
- Consists of two main parts:
- **Encoder**: This part compresses the input into a lower-dimensional latent space (bottleneck).
- It maps the input x to a latent representation z.
- The encoder function can be denoted as:

$$z=f\theta(x)$$

where fθ is the mapping with parameters θ.

- **Decoder**: This part reconstructs the original input from the compressed latent representation z.
- The decoder attempts to reverse the compression performed by the encoder and maps the latent code back to the input space:

$$x^{\wedge}=g\phi(z)$$

where gϕ is the decoder function with parameters ϕ and x^ is the reconstructed version of x.

- The model is trained to minimize the reconstruction error between the original input x and its reconstructed output x^.

- **Loss Function**
- The autoencoder is trained by minimizing the reconstruction loss between the input x and the output x^.
- For continuous data, the most common loss function is the Mean Squared Error (MSE):

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

- where xi and xi^ are the original and reconstructed inputs for each data point i, and n is the number of data points.

- **Applications of Undercomplete Autoencoders:**
- Data Compression.
- Anomaly Detection.
- Denoising.
- Feature Extraction.
- Medical Imaging.

Regularized Autoencoders

- **Regularized Autoencoders** are a variant of autoencoders designed with regularization techniques to prevent overfitting.
- Ensure the learned latent representations are robust, meaningful, and useful for downstream tasks.
- **Undercomplete Autoencoders**, which force learning by constraining the size of the bottleneck layer.
- It is seen that these autoencoders fail to learn anything useful if the encoder and decoder are given too much capacity.
- Regularized autoencoders introduce **penalties or constraints** to improve the learning process without necessarily reducing the dimensionality of the latent space.

- **Regularized Autoencoders** provide the ability to :
- Train any architecture of autoencoder successfully, choosing the code dimension and the capacity of the encoder and decoder based on the complexity of data distribution.
- Regularized autoencoders use a loss function that encourages the model to have other properties besides the ability to copy its input to its output.
- Other properties - Sparsity of the representation, smallness of the derivative of the representation, and robustness to noise or to missing inputs.

Sparse Autoencoder

- A sparse autoencoder is simply an autoencoder whose training criterion involves a **sparsity penalty Ω(h)** on the code layer h, in addition to the reconstruction error:
- $L(x, g(f(x))) + \Omega(h).$
- Where g(h) is the decoder output, and typically we have h = f (x), the encoder output.
- Sparse autoencoders are typically used to learn features for another task, such as classification.
- Construct loss function by penalizing **activations** of hidden layers so that only a few nodes are encouraged to activate when a single sample is fed into the network.

- The intuition behind this method is that :
- If a man claims to be an expert in mathematics, computer science, psychology, and classical music, he might be just learning some quite **shallow knowledge** in these subjects.
- However, if he only claims to be devoted to mathematics, we would like to anticipate some **useful insights** from him.
- Same for autoencoders we’re training — fewer nodes activating while still keeping its performance would guarantee that the autoencoder is actually learning **latent representations** instead of redundant information in our input data.
- Two different ways to construct sparsity penalty:
- **L1 regularization** and **KL-divergence**.

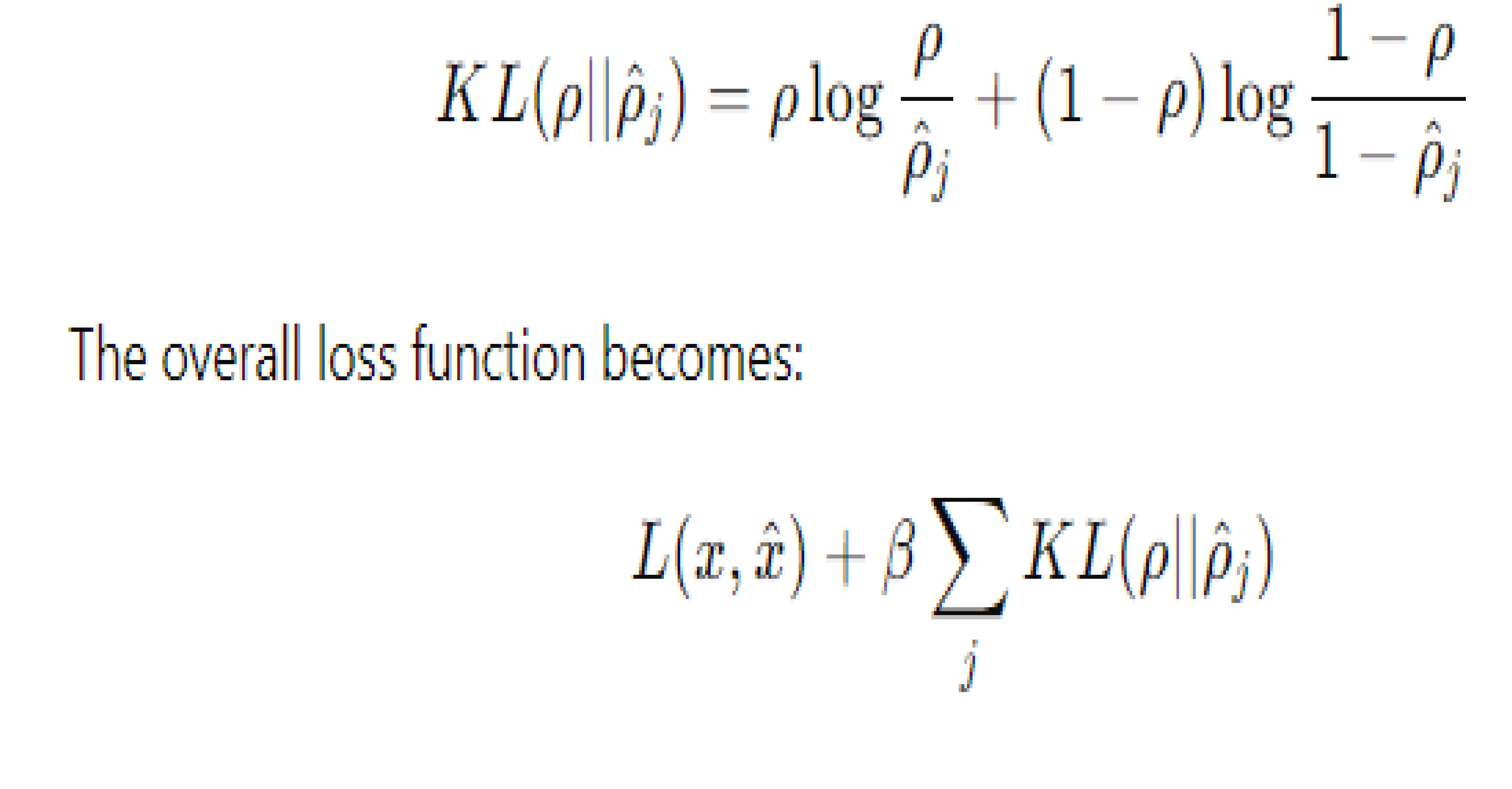
L1 regularization and KL-divergence

- Applying **L1 regularization** directly to the hidden layer's activations.
- The L1 regularization encourages the activations to be sparse by penalizing large values, which has the effect of driving many activations to zero.

$$L(x, \hat{x}) + \lambda \sum_{j} |h_j|$$

- hj are the activations of the hidden layer neurons.
- λ(lambda) is a regularization parameter that controls the strength of the sparsity constraint.

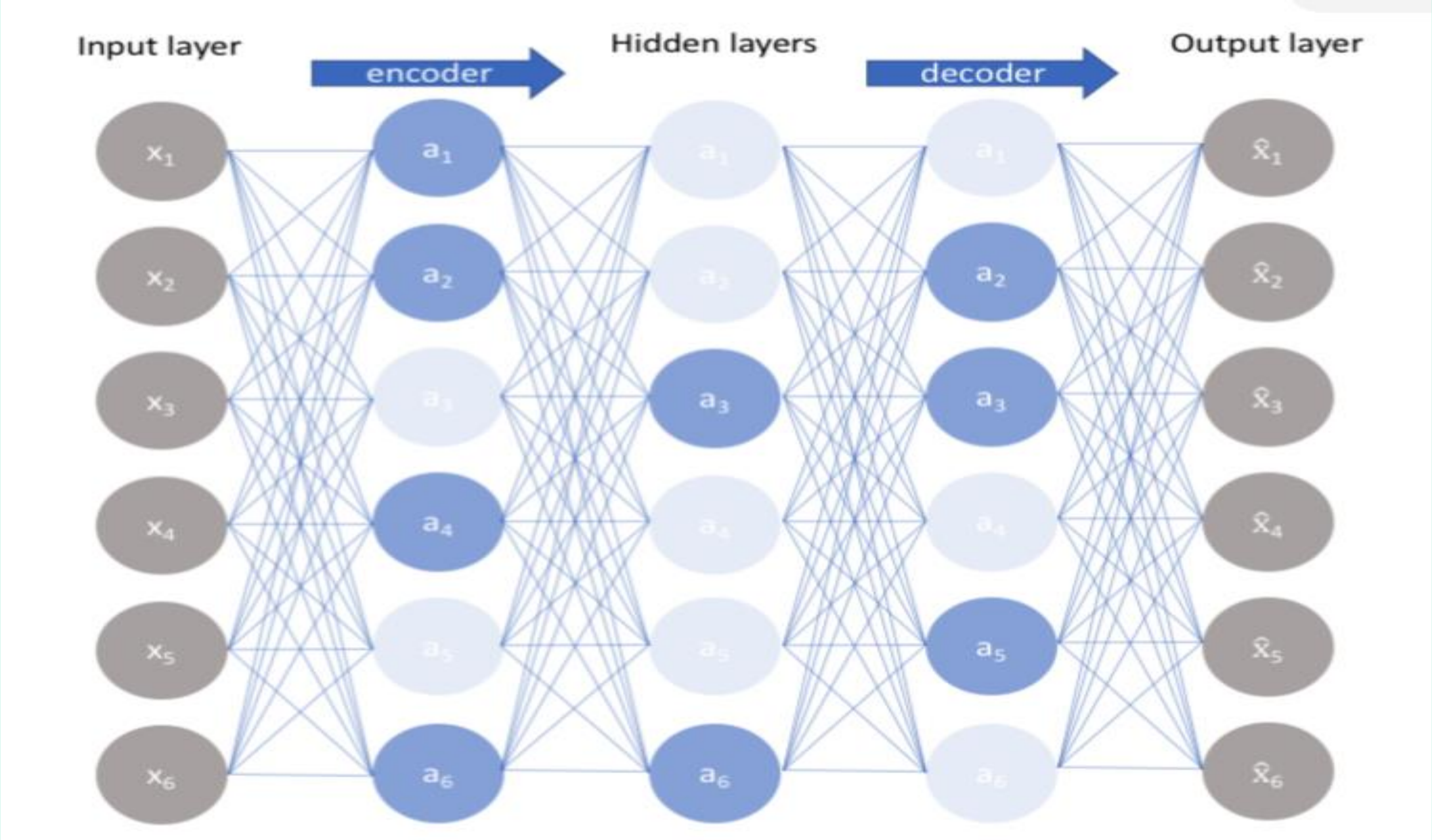
- **KL Divergence (Kullback-Leibler Divergence):**
- The goal is to enforce that the average activation of each neuron in the hidden layer is close to a target sparsity value, denoted as **ρ(rho).**
- Let ρ^j represent the average activation of hidden unit across all training examples.
- The target sparsity, ρ is typically a small value (e.g., 0.05).
- Meaning that on average, only 5% of neurons should be active for any given input.
- The KL divergence between the actual activation ρ^j and the target sparsity ρ is added to the loss function as a regularization term.



$$KL(\rho||\hat{\rho}_j) = \rho \log \frac{\rho}{\hat{\rho}_j} + (1-\rho) \log \frac{1-\rho}{1-\hat{\rho}_j}$$

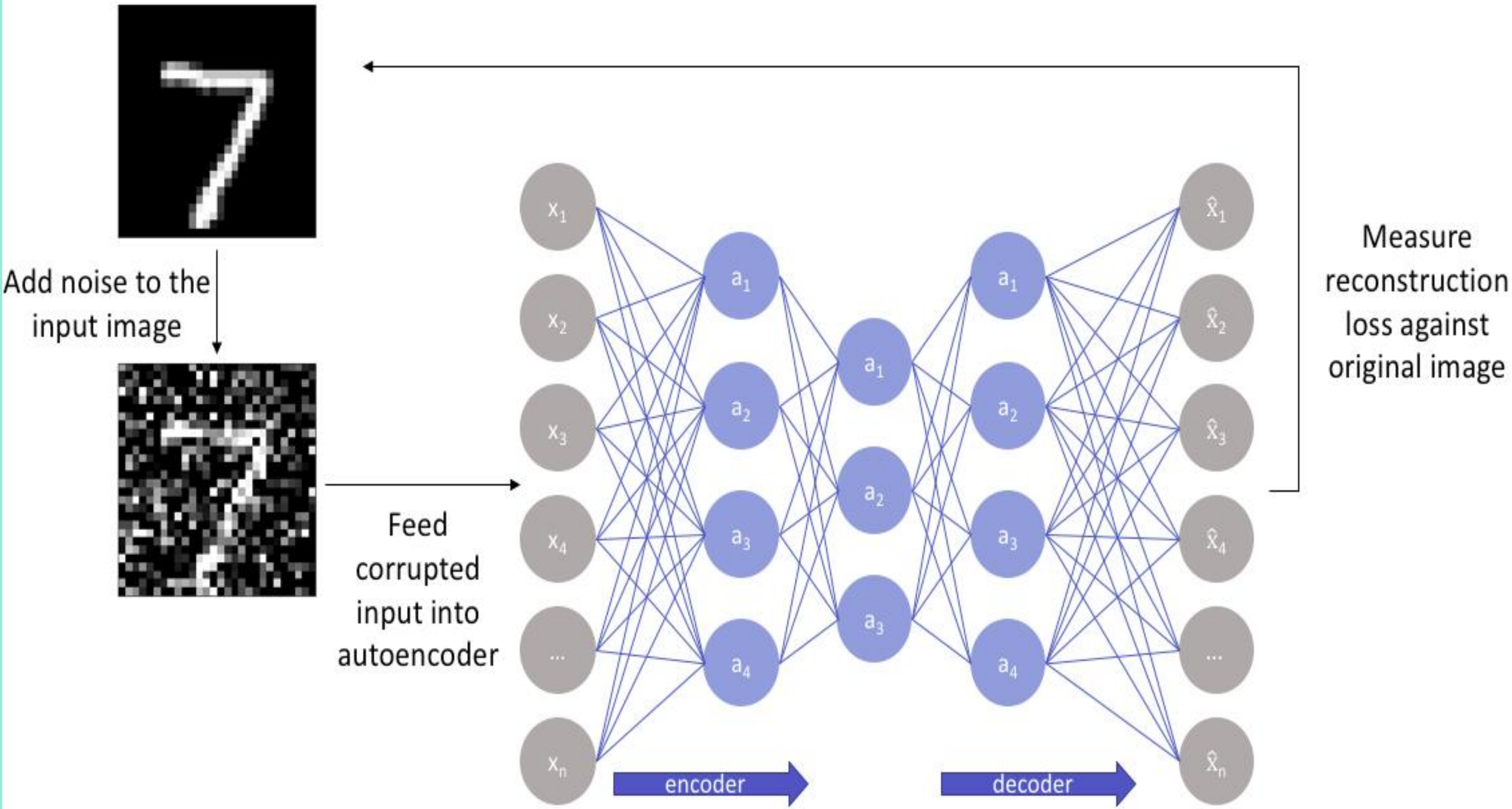
The overall loss function becomes:

$$L(x, \hat{x}) + \beta \sum_j KL(\rho||\hat{\rho}_j)$$


Input layer
encoder
Hidden layers
decoder
Output layer
x_1
x_2
x_3
x_4
x_5
x_6
a_1
a_2
a_3
a_4
a_5
a_6
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_5$
$\hat{x}_6$

Denoising Autoencoders

- Denoising autoencoders, are autoencoders that remove noise from an image.
- Rather than adding a penalty Ω to the cost function, obtain an autoencoder that learns something useful by changing the reconstruction error term of the cost function.
- Traditionally, autoencoders minimize some function
- L(x, g(f(x)))
- A denoising autoencoder (DAE) instead minimizes:
- L(x, g(f(˜x)))
- Where ˜x is a copy of x that has been corrupted by some form of noise.
- where noise has been added via digital alterations.



Add noise to the input image

Feed corrupted input into autoencoder

Measure reconstruction loss against original image

- The denoising autoencoder gets rid of noise by learning a representation of the input where the noise can be filtered out easily.
- While removing noise directly from the image seems difficult,
- The autoencoder performs by mapping the input data into a lower-dimensional manifold.
- Essentially, denoising autoencoders work with the help of non-linear dimensionality reduction.
- Loss function generally used ,networks is L2 or L1 loss.

Contractive Autoencoders

- The idea behind that is to make the autoencoders robust of small changes in the training dataset.
- To deal with the challenge that is posed in basic autoencoders, proposed to add another penalty term to the loss function of autoencoders.

$$\|J_h(X)\|_F^2 = \sum_{ij} \left(\frac{\partial h_j(X)}{\partial X_i}\right)^2$$

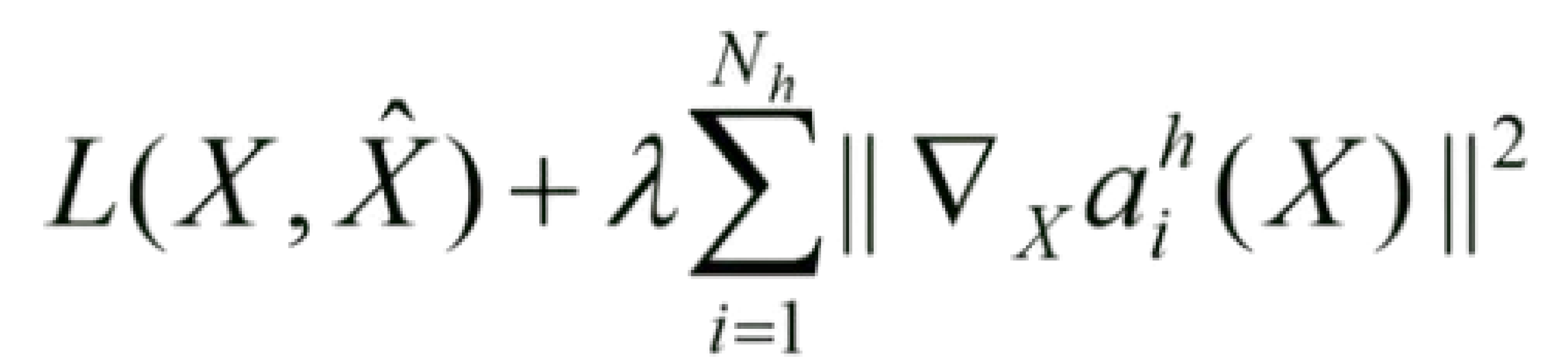
Above penalty term is the Frobinious Norm of the encoder.

$$\|A\|_F = \sqrt{\sum_{j=1}^{m} \sum_{i=1}^{N_h} |a_{ij}|^2}$$

Jacobian Matrix

$$J = \begin{bmatrix} \dfrac{\partial a_1^h(X)}{\partial x_1} & \dfrac{\partial a_1^h(X)}{\partial x_2} & \cdot\cdot & \dfrac{\partial a_1^h(X)}{\partial x_m} \\ \dfrac{\partial a_2^h(X)}{\partial x_1} & \dfrac{\partial a_2^h(X)}{\partial x_2} & \cdot\cdot & \dfrac{\partial a_2^h(X)}{\partial x_m} \\ \cdot\cdot & \cdot\cdot & \cdot\cdot & \cdot\cdot \\ \dfrac{\partial a_{N_h}^h(X)}{\partial x_1} & \dfrac{\partial a_{N_h}^h(X)}{\partial x_2} & \cdot\cdot & \dfrac{\partial a_{N_h}^h(X)}{\partial x_m} \end{bmatrix}$$

Final Loss Function

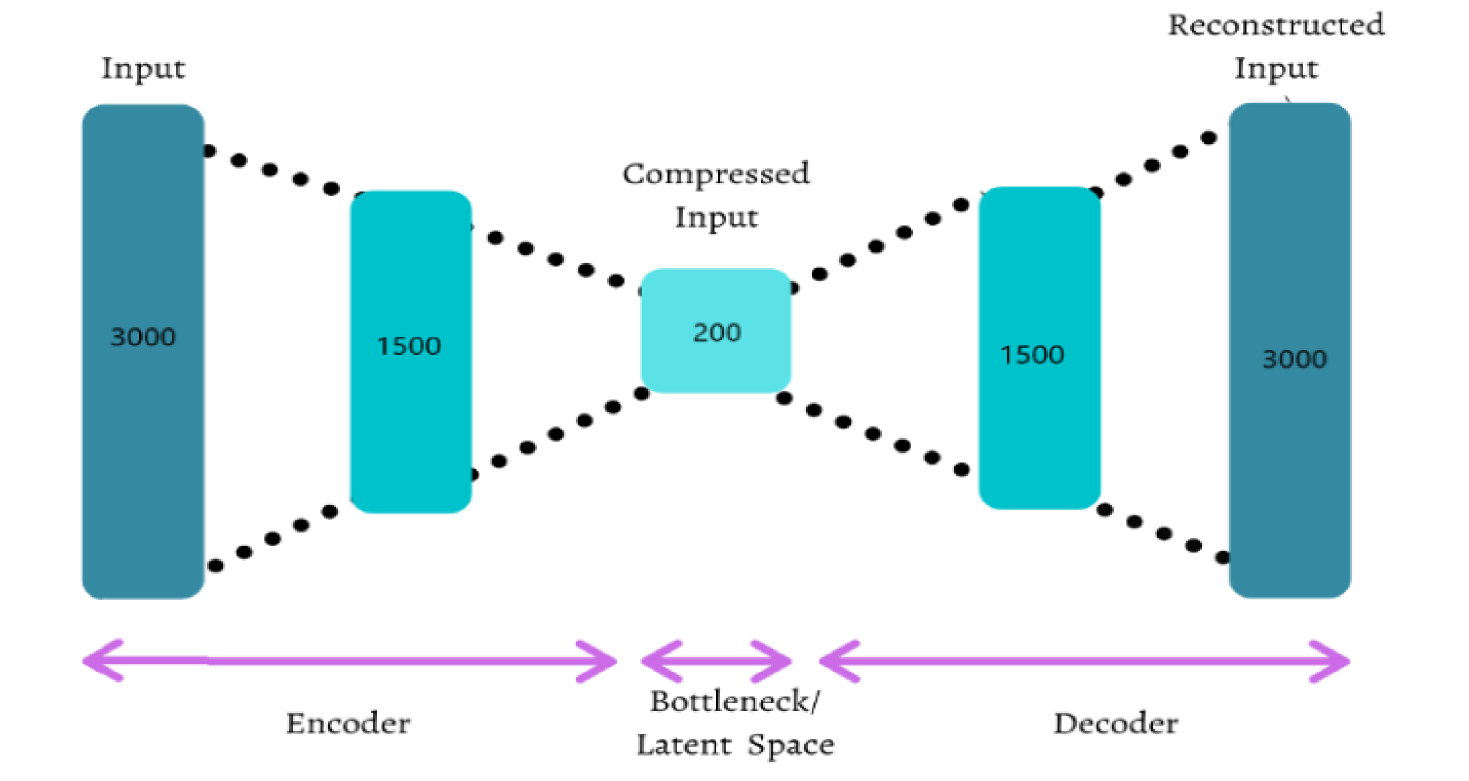


$$L(X,\hat{X})+\lambda\sum_{i=1}^{N_h}\|\nabla_X a_i^h(X)\|^2$$

Applications of Autoencoders

Dimensionality Reduction :

- Autoencoders train the network to explain the natural structure in the data into efficient lower-dimensional representation.
- It does this by using decoding and encoding strategy to minimize the reconstruction error.



Feature Extraction:

- Autoencoders can be used as a feature extractor for classification or regression tasks.
- After training an autoencoder network using a sample of training data, can ignore the decoder part of the autoencoder.
- Only use the encoder to convert raw input data of higher dimension to a lower dimension encoded space.
- Lower dimension of data can be used as a feature for supervised tasks.

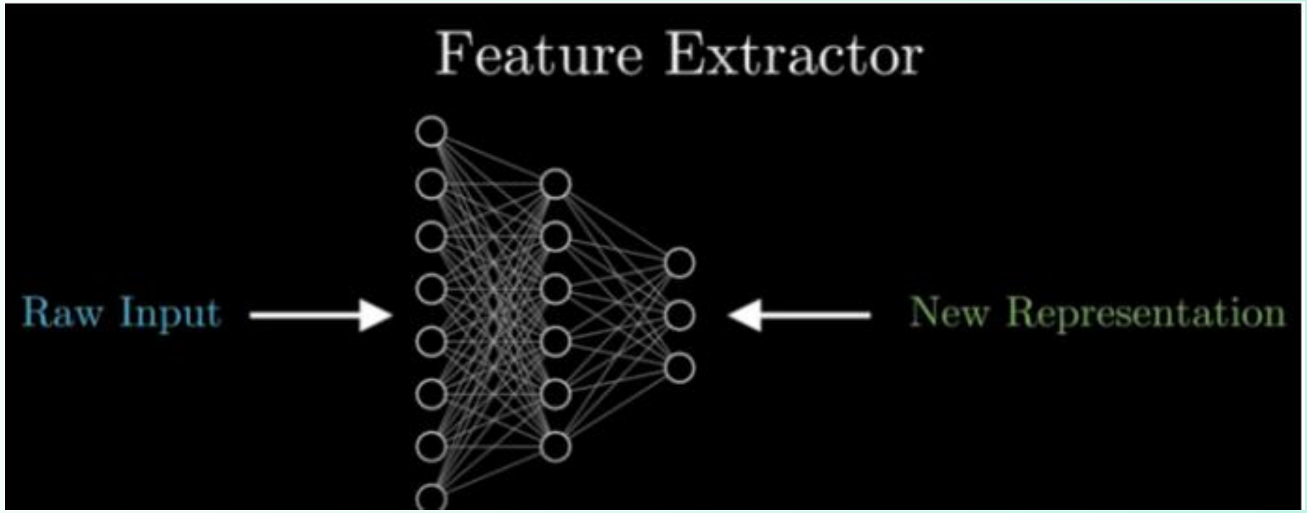


Image Denoising:

- The real-world raw input data is often noisy in nature.
- To train a robust supervised model requires cleaned and noiseless data.
- Autoencoders can be used to denoise the data.

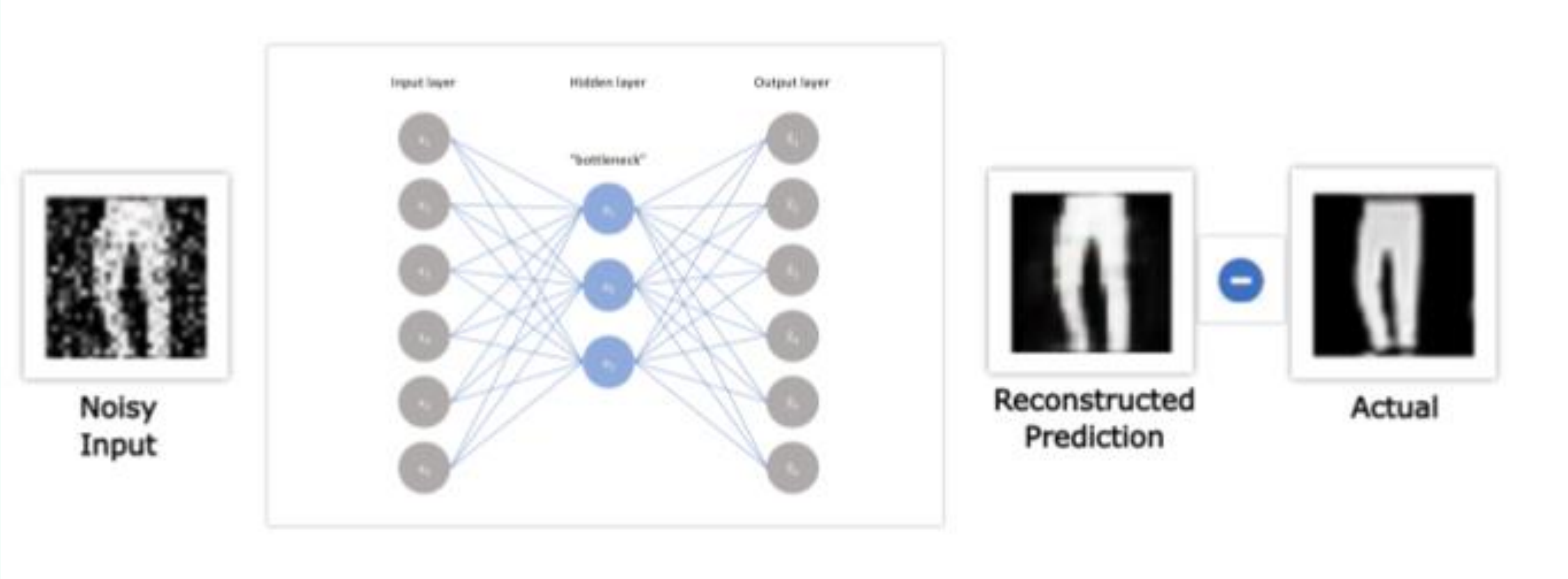


Image Compression:

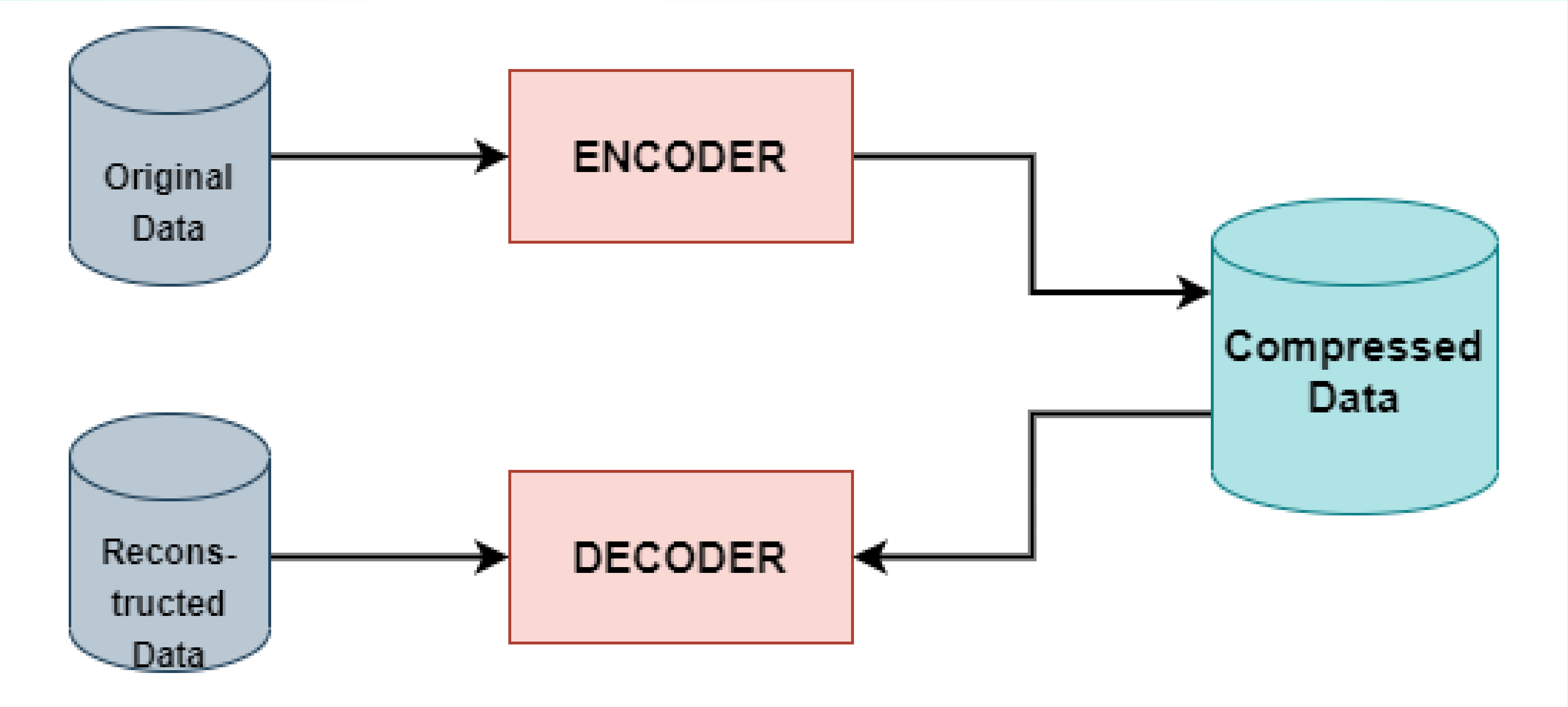
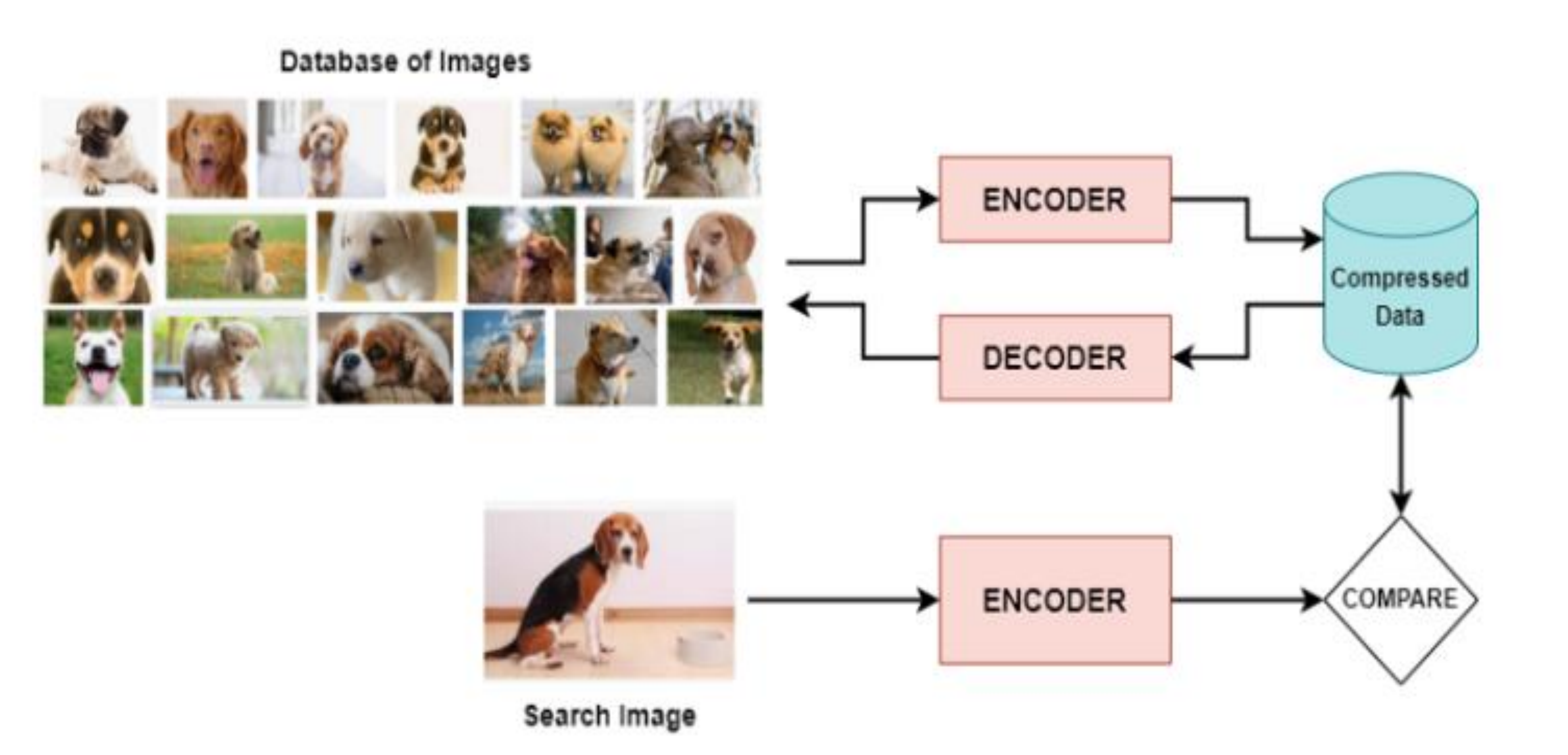


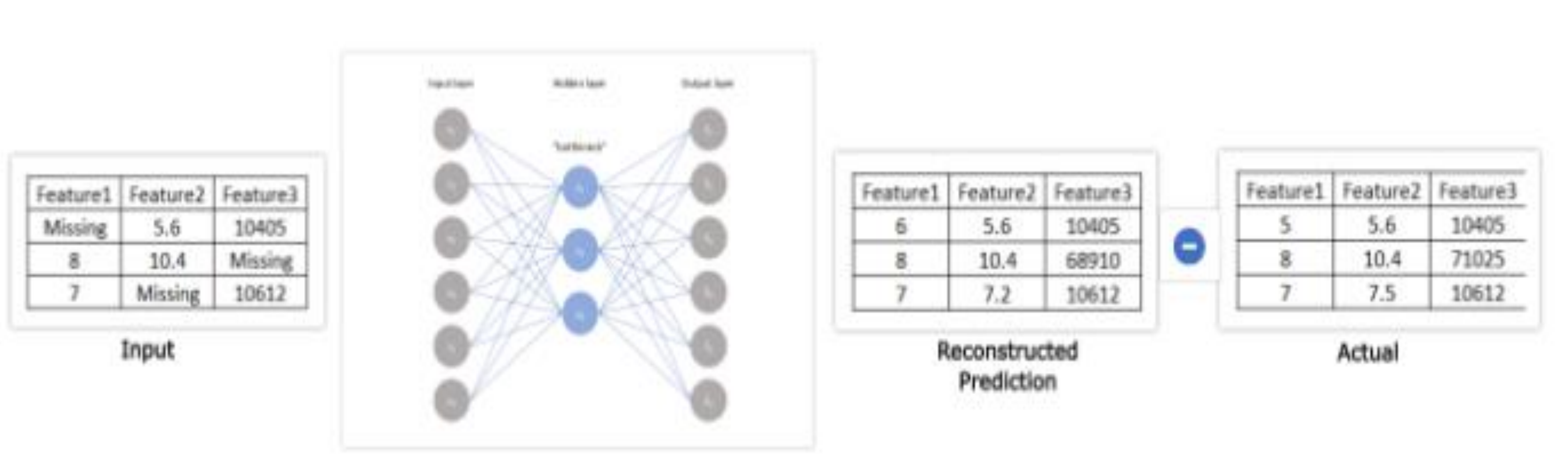
Image Search:



Missing Value Imputation:

Denoising autoencoders can be used to impute the missing values in the dataset.

Randomly placing missing values in the input data and trying to reconstruct the original raw data by minimizing the reconstruction loss.



Feature1	Feature2	Feature3
Missing	5.6	10405
8	10.4	Missing
7	Missing	10612

Input

Feature1	Feature2	Feature3
6	5.6	10405
8	10.4	68910
7	7.2	10612

Reconstructed Prediction

Feature1	Feature2	Feature3
5	5.6	10405
8	10.4	71025
7	7.5	10612

Actual

Anomaly Detection:

- Anomaly detection is another useful application of an autoencoder network.
- An anomaly detection model can be used to detect a fraudulent transaction or any highly imbalanced supervised tasks.