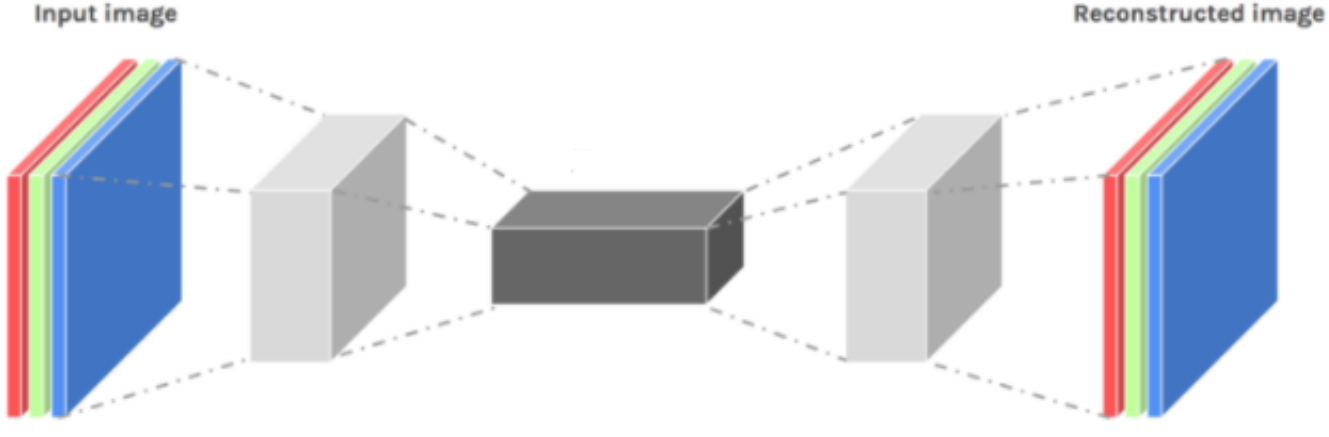
Autoencoders

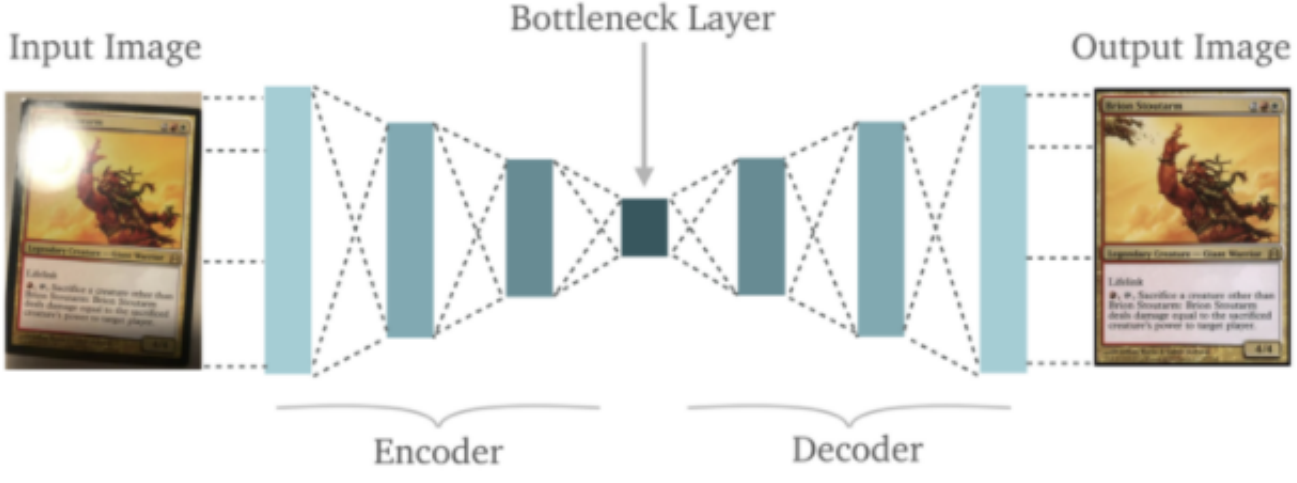
Introduction

- A specific type of feedforward neural networks where the input is same as the output.
- Mainly used to reduce the size of the inputs into a smaller representation.
- The inputs is compressed into a lower-dimensional code and the original data can be reconstructed from the compressed data.
- The code is a compact summary of the input called as the **latent-space representation.**

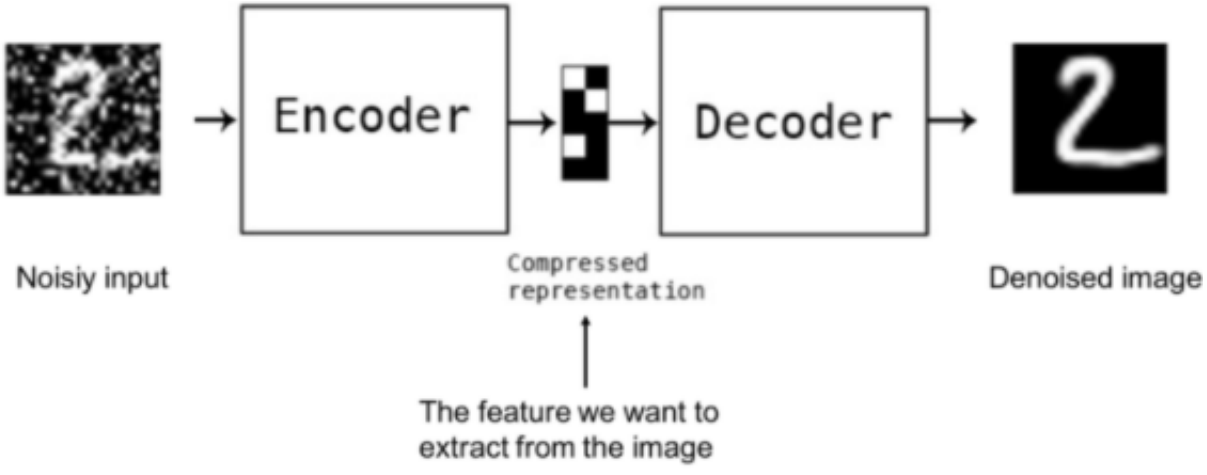


Applications

- **Image coloring**
- **Feat**



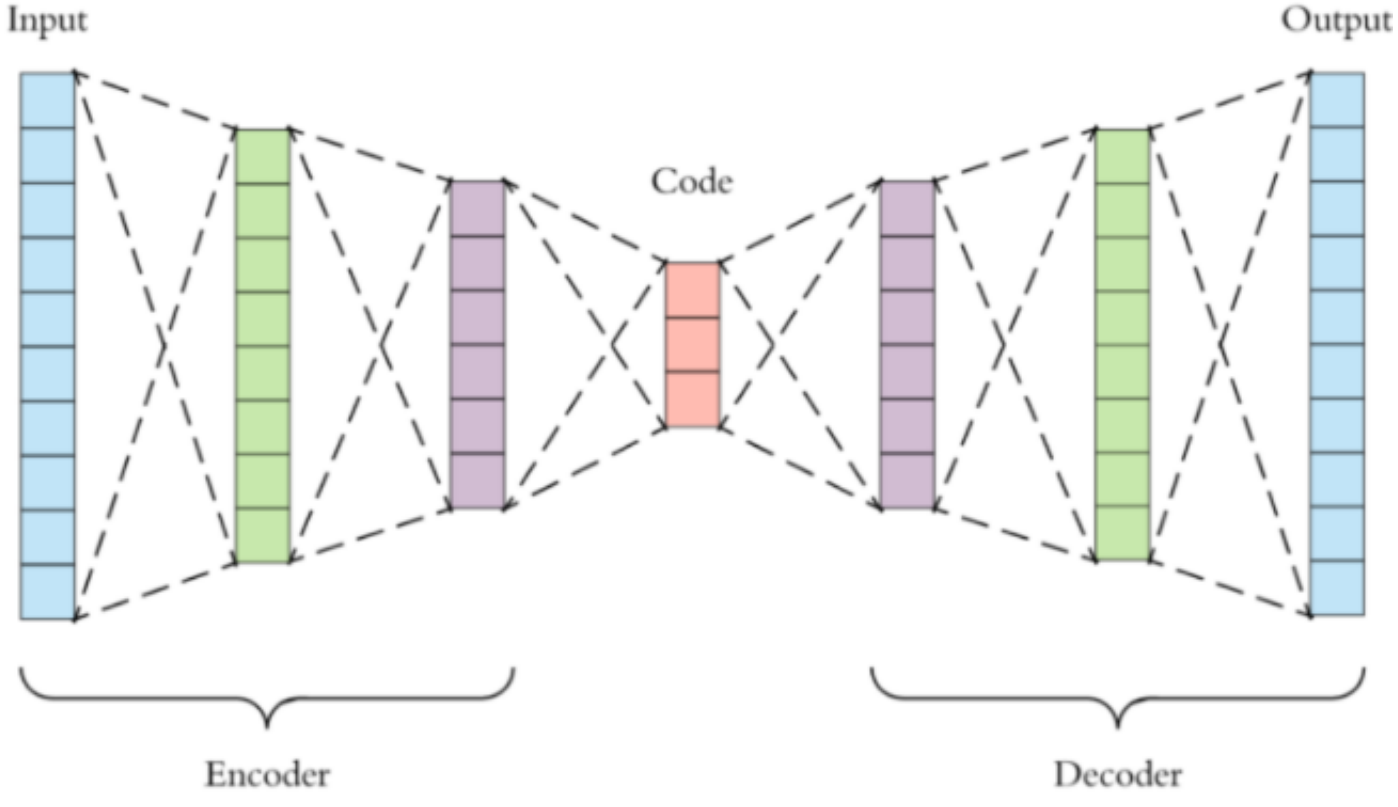
- **Image denoising**



- **Watermark removal**

Architecture of Autoencoders

Autoencoders have 3 layers: Encoder, Code, Decoder



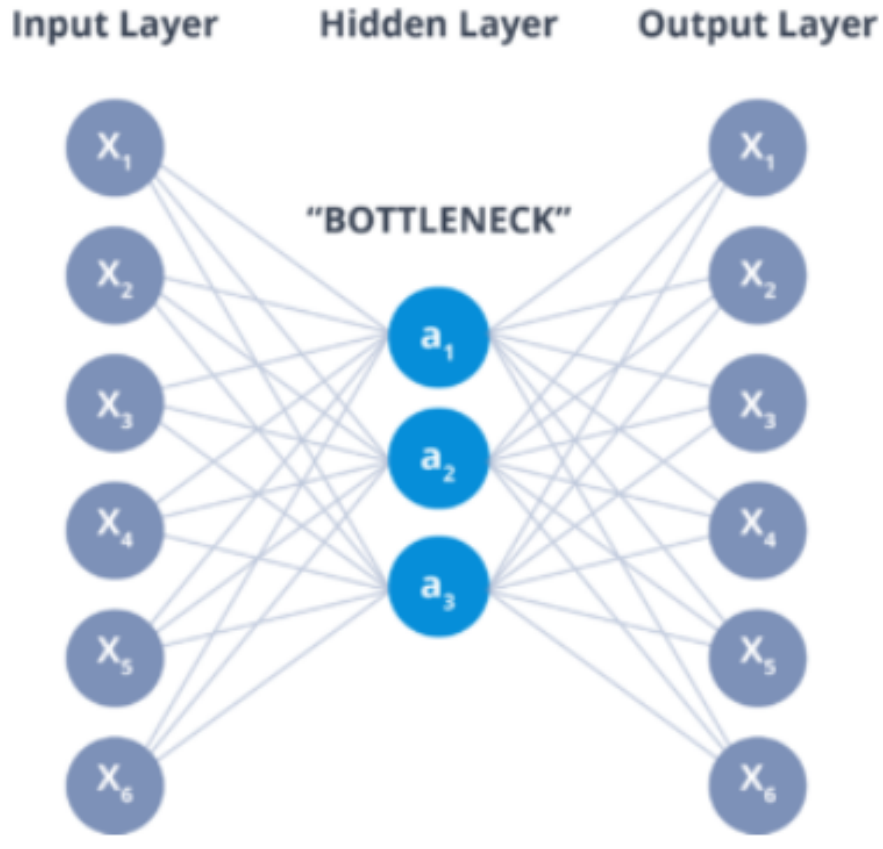
- **Encoder**

  Compresses the input into a latent space representation. Encoder layers encodes the input image to a compressed representation in a reduced dimension. This will be a distorted version of the original image.
- **Code**

  Represents the compressed input which is fed to the decoder.
- **Decoder**

  Decodes the image back to the original dimension. This is a lossy reconstruction of the original image and is reconstructed from the latent space representation.

The layer between the encoder and decoder (i.e., the code) is also known as Bottleneck. This decides which aspects of observed data are relevant and what aspects could be discarded.



Properties of Autoencoders

- **Data-specific**: Autoencoders are only able to compress data similar to what they have been trained on. They learn features specific for the given training data.
- **Lossy**: Decompressed outputs will be degraded. It is not ideal for lossless compression.
- **Unsupervised**: Give the raw input data. They don’t need explicit labels to train on. They can be called *self-supervised* since they generate their own labels from the training data.

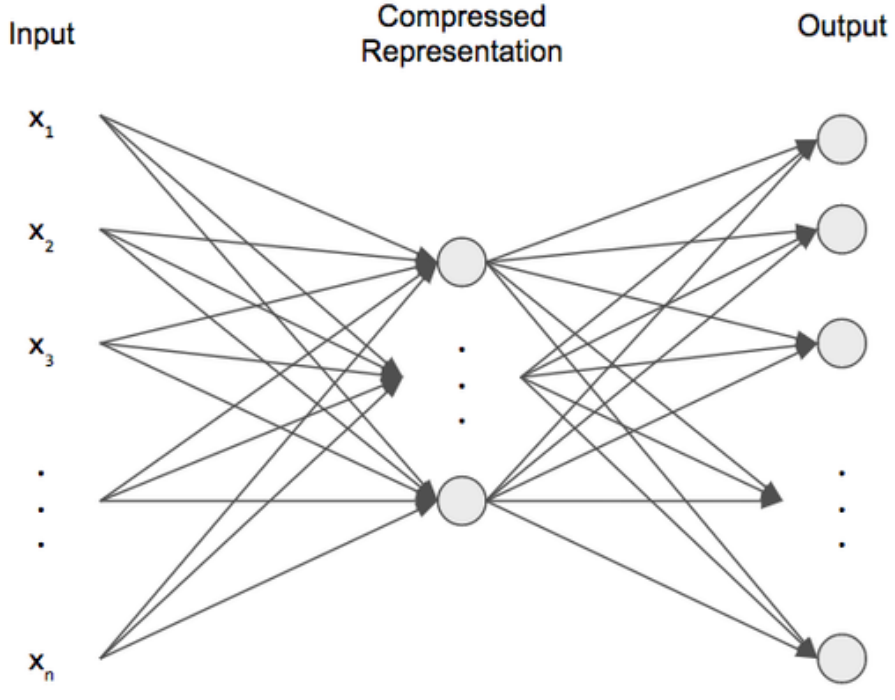
Hyperparameters of autoencoders

- **Code size**: It is the number of nodes in the middle layer. Smaller size results in more compression.
- **Number of layers**: It can consist of as many layers as we want
- **Number of nodes per layer**: It decreases with each subsequent layer of the encoder and increases back in the decoder. The decoder is symmetric to the encoder in terms of layer structure.
- **Loss function**: Either use mse or binary cross-entropy. If input values in range [0,1] typically cross-entropy is used otherwise mse.

Types of autoencoders

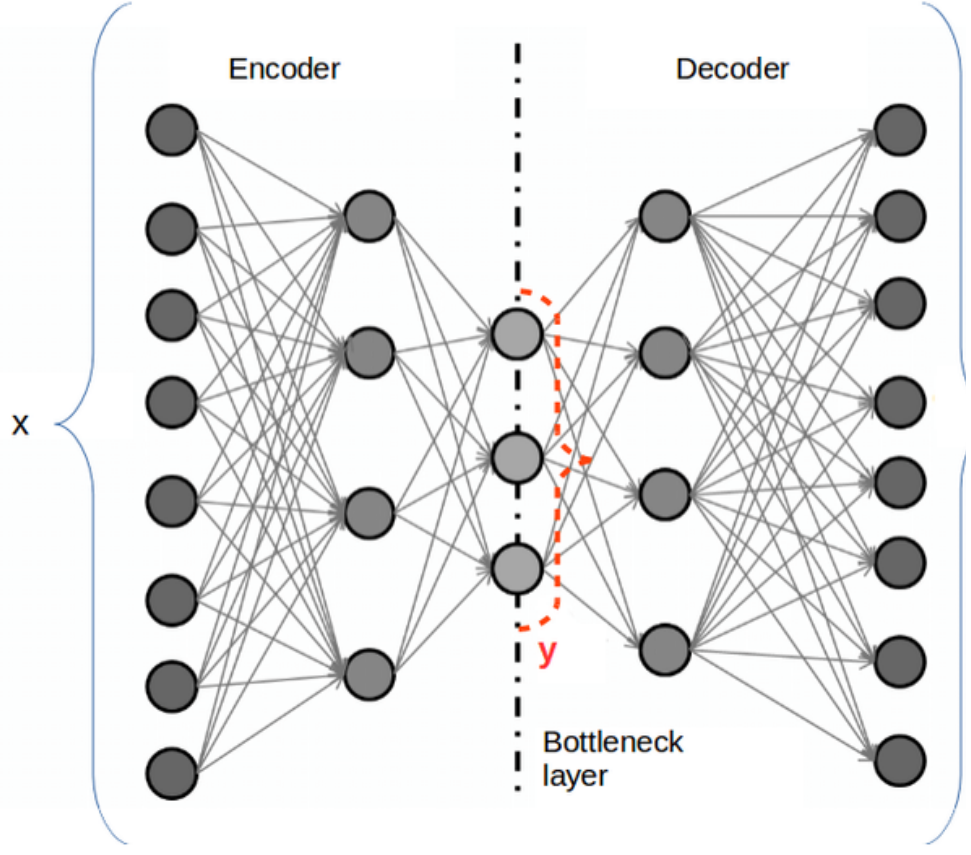
Vanilla autoencoders

Simplest of all encoders. They consist of only one hidden layer.



Deep autoencoders

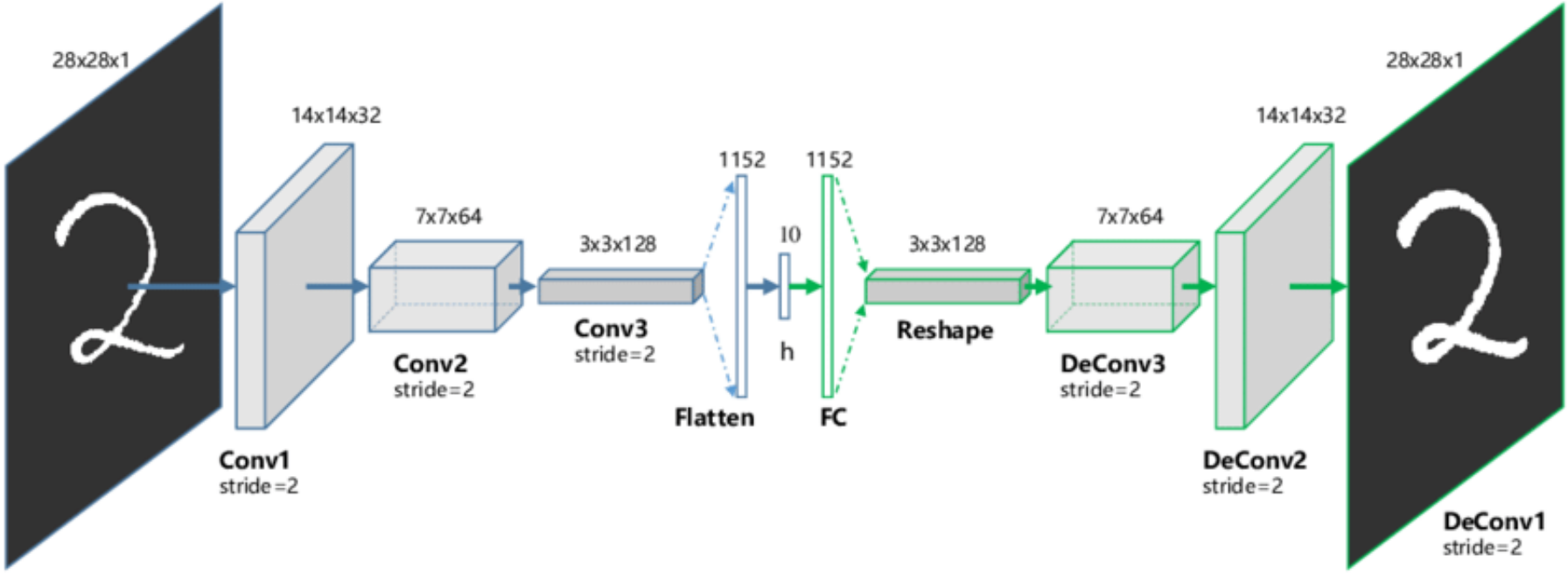
More hidden layers can be used in the autoencoder to reduce and reconstruct the input. The hidden layers have a symmetry where we keep reducing the dimensionality at each layer (the encoder) until we get to the encoding size, then, we expand back up, symmetrically, to the output size (the decoder).



Covolutional autoencoders

Instead of using fully-connected layers, convolutional and pooling layers are used to encode the input. Autoencoders applied to images are convolutional.

The encoder section consists of conv2D and maxpooling (downsampling) layers whereas the decoder section consists of conv2D and upsampling layers.



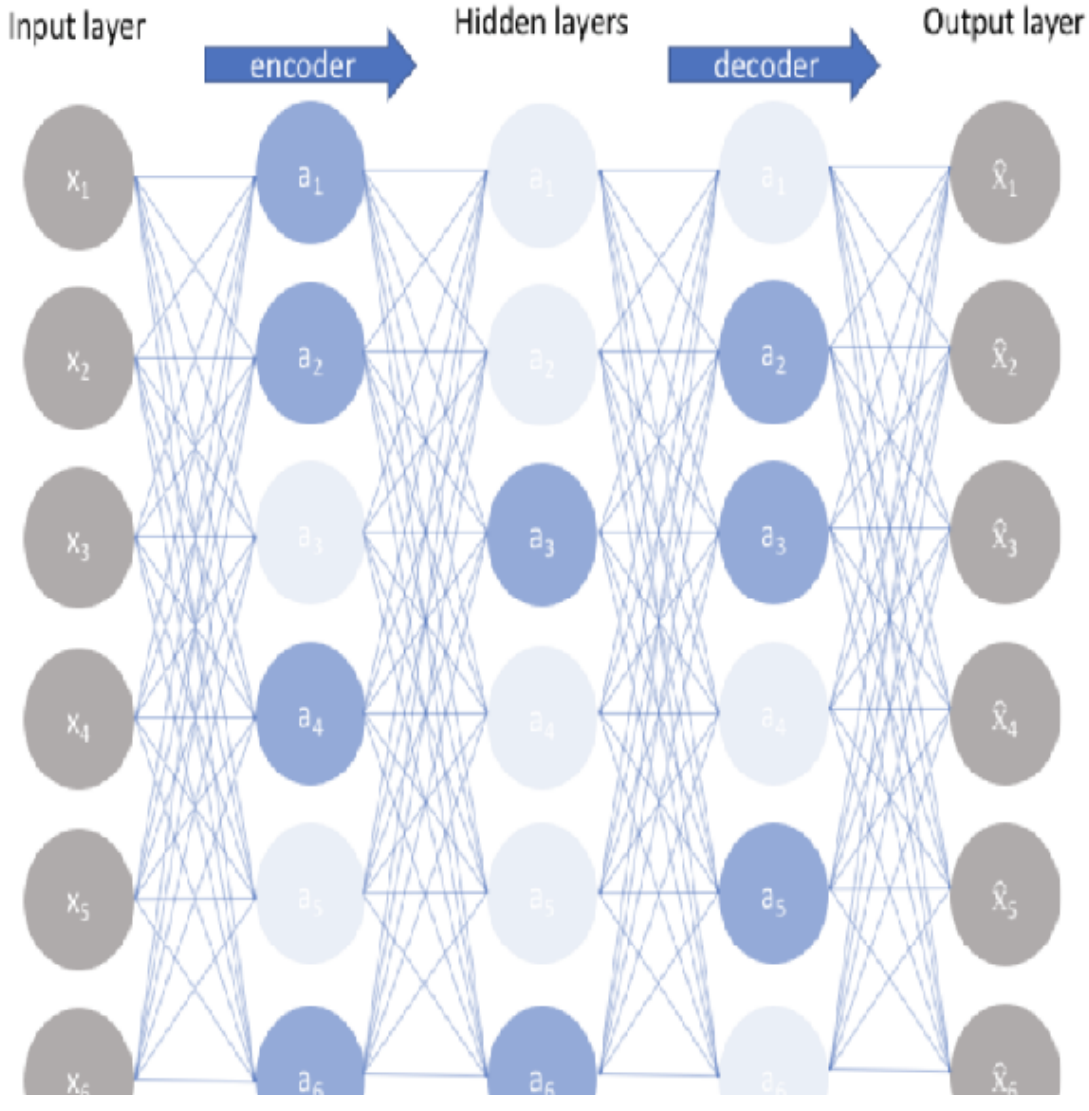
Regularized autoencoders

Regularized autoencoders use a loss function that encourages the model to have other properties besides copying the input to the output.
2 types of regularized autoencoders:

- Sparse autoencoders
- Denoising autoencoders

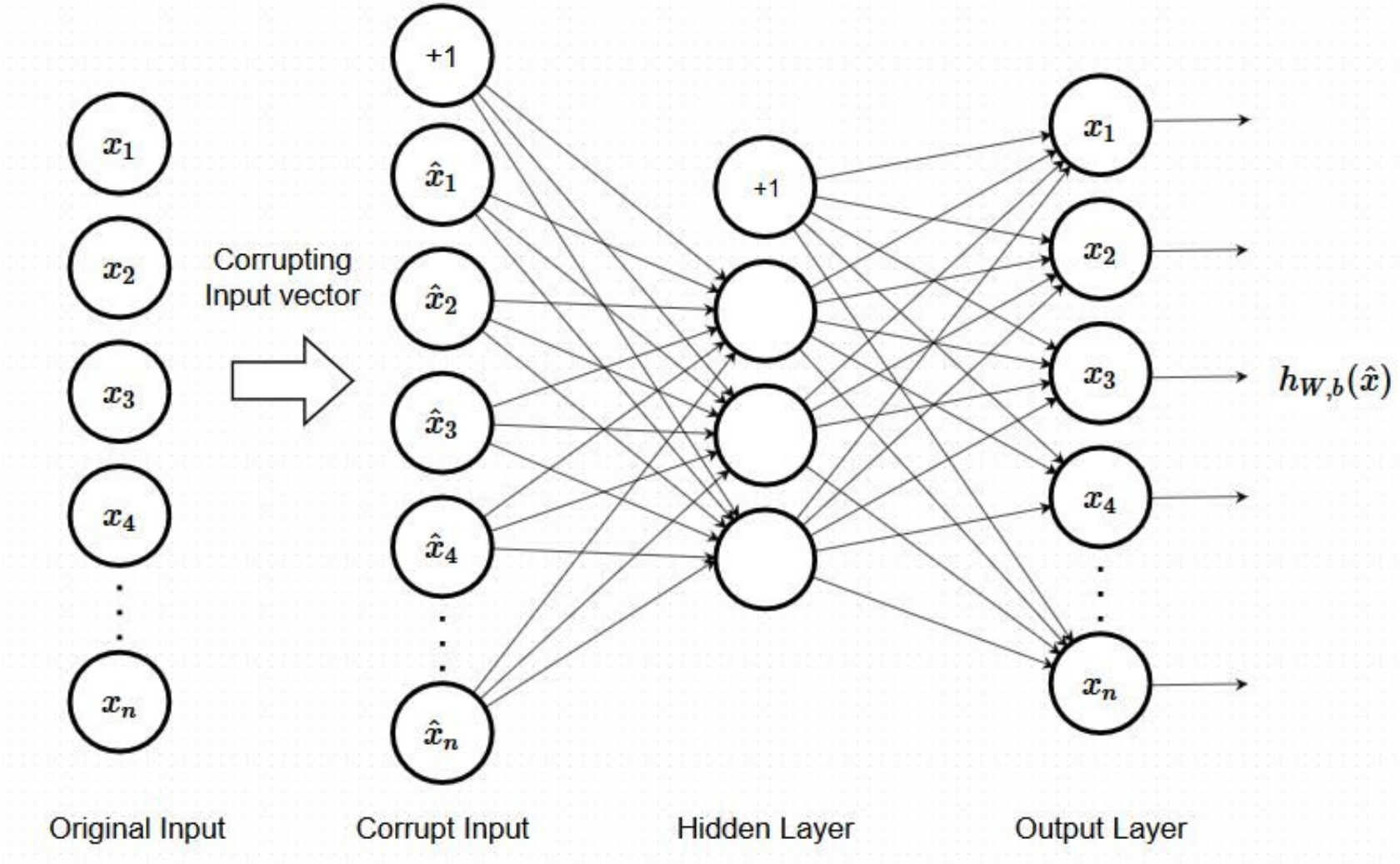
Sparse autoencoders

- The loss function is constructed in such a way that only a few nodes are activated when a single record is given to the network.
- It penalizes the activations within a layer.
- No need of reducing the number of nodes in the hidden layers.
- Activating few nodes will guarantee that the network learns the latent information rather than redundant information.
- Useful in dimensionality reduction.
- The activated nodes of a trained model are data dependent, as different inputs will result in activations of different nodes through the network.


Input layer
encoder
Hidden layers
decoder
Output layer
x1
x2
x3
x4
x5
x6
a1
a2
a3
a4
a5
a6
a1
a2
a3
a4
a5
a6
a1
a2
a3
a4
a5
a6
x̂1
x̂2
x̂3
x̂4
x̂5
x̂6

Denoising autoencoders

- Denoising autoencoders corrupts the data on purpose by randomly turning some of the input values to zero.
- Here, the training data and testing data are different.
- For the training data, we add random, Gaussian noise, and test data will be the original, clean image. This trains the autoencoder to produce clean images given noisy images.
- This is useful in feature extraction and noise removal.


x_1
x_2
x_3
x_4
x_n
Corrupting
Input vector
+1
$\hat{x}_1$
$\hat{x}_2$
$\hat{x}_3$
$\hat{x}_4$
$\hat{x}_n$
+1
x_1
x_2
x_3
x_4
x_n
$h_{W,b}(\hat{x})$
Original Input
Corrupt Input
Hidden Layer
Output Layer