Objectives

In this chapter, you should be able to learn the following,

- Background and motivation behind autoencoders
- Introduction to autoencoders
- Structures of autoencoders
- Autoencoder types
- Applications

Background and motivation

Let us begin with the ubiquitous example to understand the motivation behind the autoencoders.

Suppose, you have given a task to classify an image, whether it is a cat or a dog, how will you build a model for this?

First of all, you need data, right? Generally, data should have images, and it's respective labels so that you can apply your machine learning algorithms, such as SVM or Neural nets. This is how we build a model to classify an image.

Figure 1: (a) Cat image (b) Dog image

But what if the data doesn't have any labels - only images. In this case, how can we build a model that classifies the image without labels? Here comes a handy approach, called autoencoders.

In the real-world, data generation and preparation take a lot of time and cost. Labeling and categorizing the data also take a significant amount of time. Unfortunately, most of the machine learning algorithms require labels for training the model. Hence, the autoencoder opens the door for tackling this kind of problem.

Let's see another problem from the same above example. Suppose you have to transfer the high dimensional image over the network. Generally, raw images aren't in compact form, and then it will take much time, right? So, what may be the solution to this problem, because modern world image processing techniques and video streaming with low-speed rate are not plausible? One way to solve this problem is, compress the image at the source end, then transfer this compressed image across the network and finally reconstruct it in the receiver end. Here, the same thing does by the autoencoder.

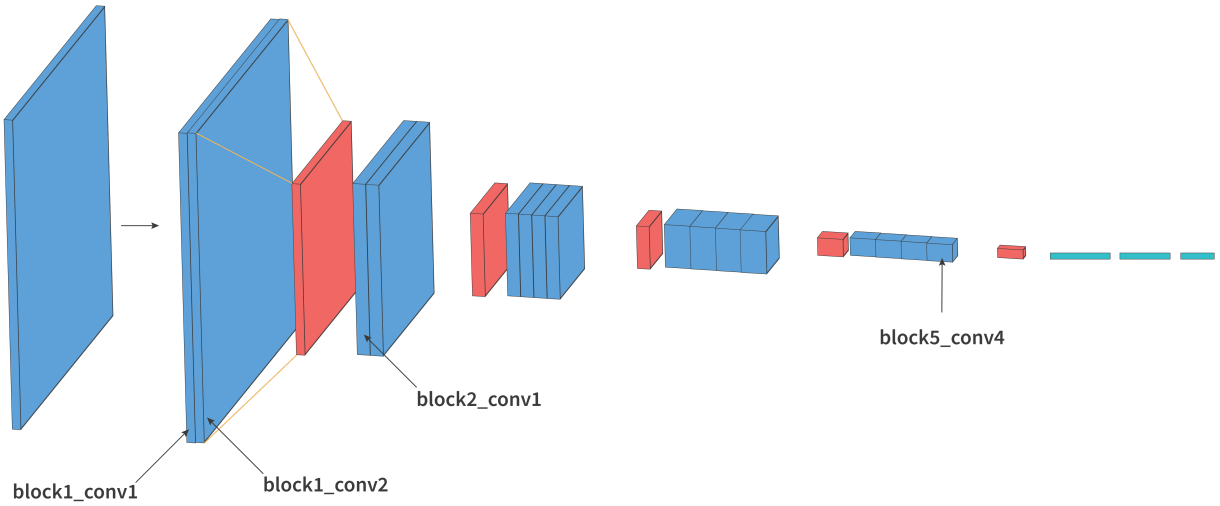

Figure 2: Block diagram for dimensionality reduction

To summarize, autoencoders learn the representation of data (without labels), especially by dimensionality reduction such as PCA, to ignore the noise.

Introduction

Simply, autoencoders are unsupervised neural networks. Unlike neural networks, its output is compared with input (due to no labels). It reduces the dimension of input data, transfers over the network, and later reconstructs the data as close to the original input. This is how autoencoder works.

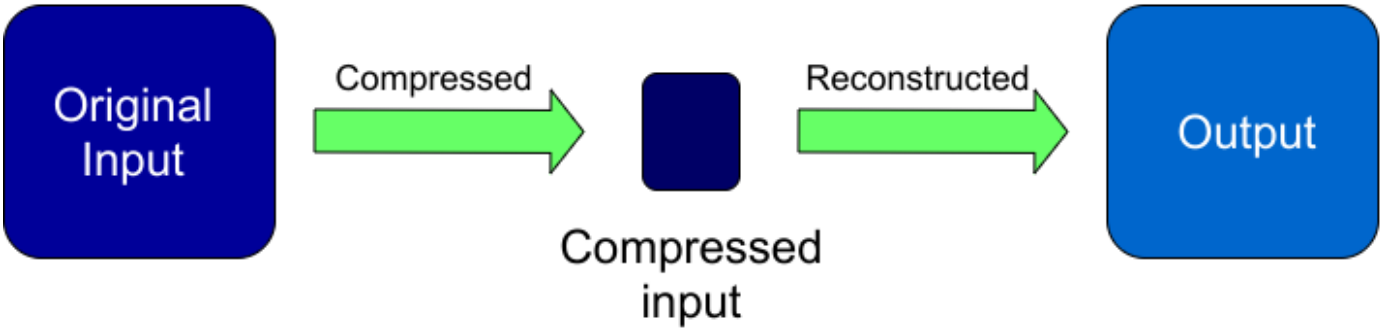

Figure 3: Simple autoencoder diagram

Structures

By definition, an autoencoder is a technique to encode data automatically. It has mainly three components,

1. Encoder
2. Code (Bottleneck)
3. Decoder

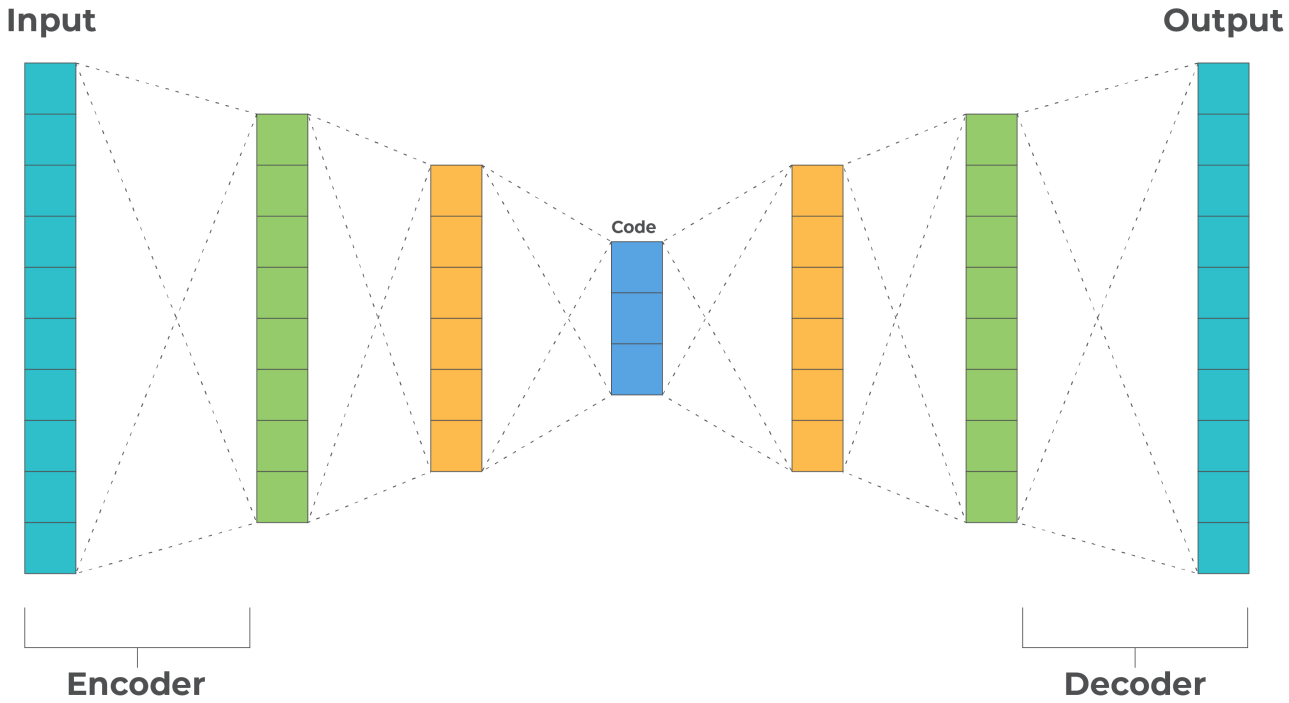

Figure 4: Autoencoder with it's layers shown

1. **Encoder:** This is the first component of autoencoder, which encodes the input data into smaller dimension i.e., data compression.
2. **Code (Bottleneck)**: This component is also known as latent structure. It is the lowest possible compressed structure of input data.
3. **Decoder**: The compressed structure (code) is then reconstructed by the decoder to produce output data as original as input data.

Properties

Autoencoders primary purpose is a compression (dimensionality reduction), along with this it has the following properties:

1. Data-specific

> Autoencoder network is only efficient at compressing/reconstructing the data that it has been trained on because, during the training, it learns the feature of the training data. Autoencoders trained on face image compression are not proficient at handling compression/reconstruction of landscape images. These types of architecture are different from standard compression algorithms such as zip, 7z, etc.

2. Unsupervised

> Throw some unlabeled data as the input to the network and train the autoencoders for satisfactory outputs. This is why these networks are called unsupervised network. The self-supervised network is another name of autoencoders.

3. Lossy Nature

> Autoencoders are lossy, and they cannot get an exact reconstruction of the input data. There will always be lossy compression applied to the input, so the output is a degraded version of the input. If the user wants lossless compression, they are not the choice of architecture.

Hyperparameters

- Code size
  - Smaller size results in more compression.
- Number of layers
  - It can have as many layers as wanted or required.
- Number of nodes per layer
  - Generally, decreases in the encoder part and again increase in decoder part due to symmetricity.
- Loss function
  - It can use both binary cross-entropy as well as mean squared error.
  - If an error is between 0 to 1, binary cross-entropy is preferred. Otherwise, the mean square error is preferred.

Types based on the structure

There are different types of autoencoders based on their three underlying structures. Here, we mainly discuss two types of autoencoders,

1. Undercomplete autoencoder

- In this architecture, code (latent feature or hidden structure) has a smaller dimension than an input.
- One of the benefits of this model is that it does not require any regularization. It maximizes the probability of the data rather than directly copying input to the output.

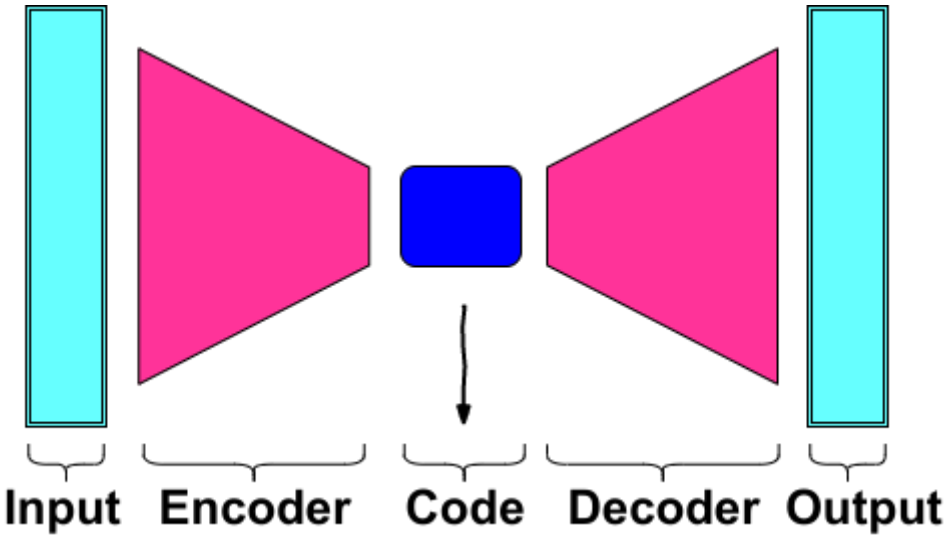

Figure 5: Undercomplete autoencoder block diagram

2. Overcomplete autoencoder

- In this architecture, code has higher or equal dimension than the input.
- Since, there is no compreesion in the hidden layer, this encoder model doesn't learn anything meaningful features rather than just copying the input to the output.
- This problem can be solve by introducing some regularisation term.

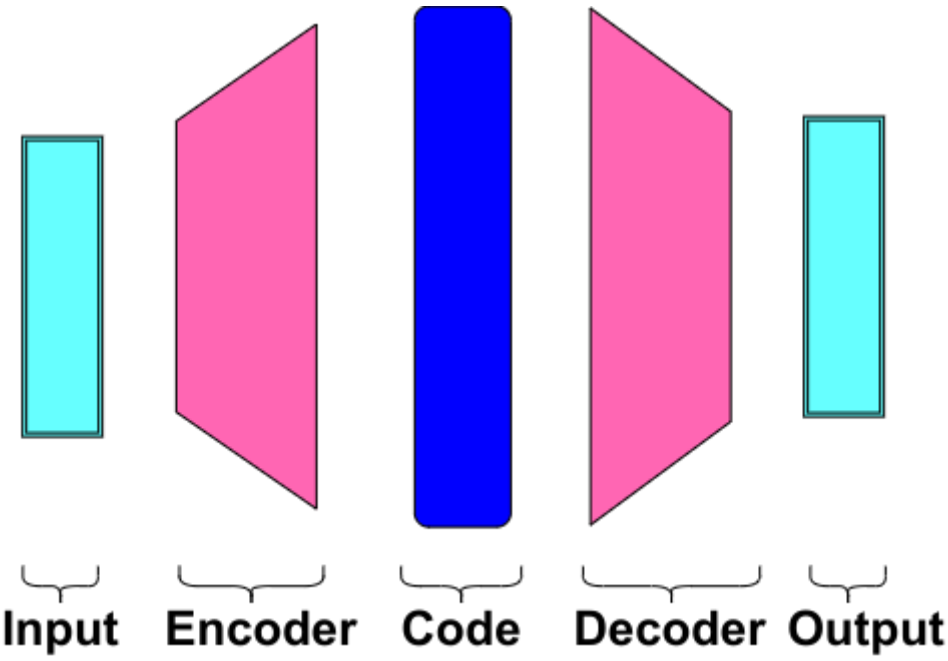

Figure 6: Overcomplete autoencoder block diagram

Types based on functionality

Based on functionality, types of neurons in the hidden layer (i.e., latent structure), etc., autoencoder can be of the following types,

- ## Sparse autoencoders

- These encoders generally have more hidden units than inputs, but only a small number of units are activated to learn the features from the data. The sparsity constraint introduced in the hidden layer to prevent overfitting. This forces model to prevent the output layer copy the input layer.

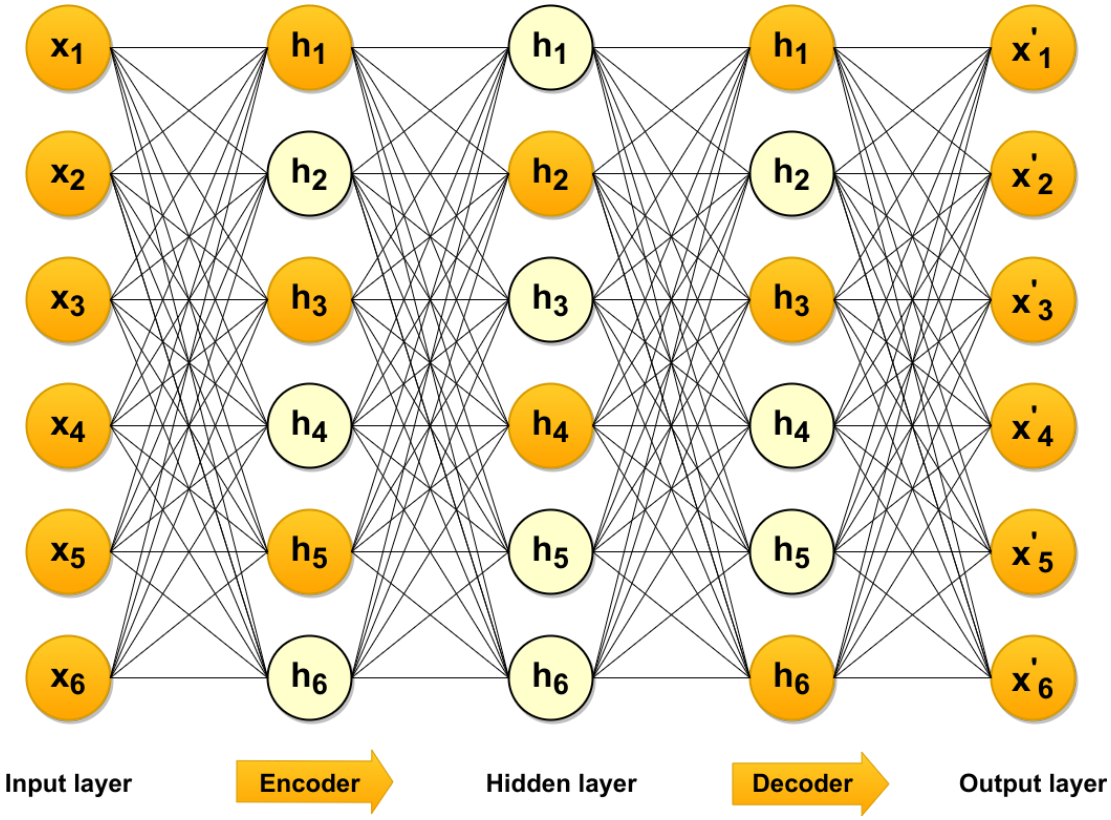

Figure 7: Sparse autoencoder

Convolutional autoencoders

- Instead of using a simple dimensionality technique, these encoders use convolution layers to extract essential features from the input and similar structure for reconstructing it. The main benefits of having convolution layers are,
  - Due to convolution nature, realistic-sized high dimensional images can be well scaled
  - Can reconstruct the missing part as well as remove noise from the images

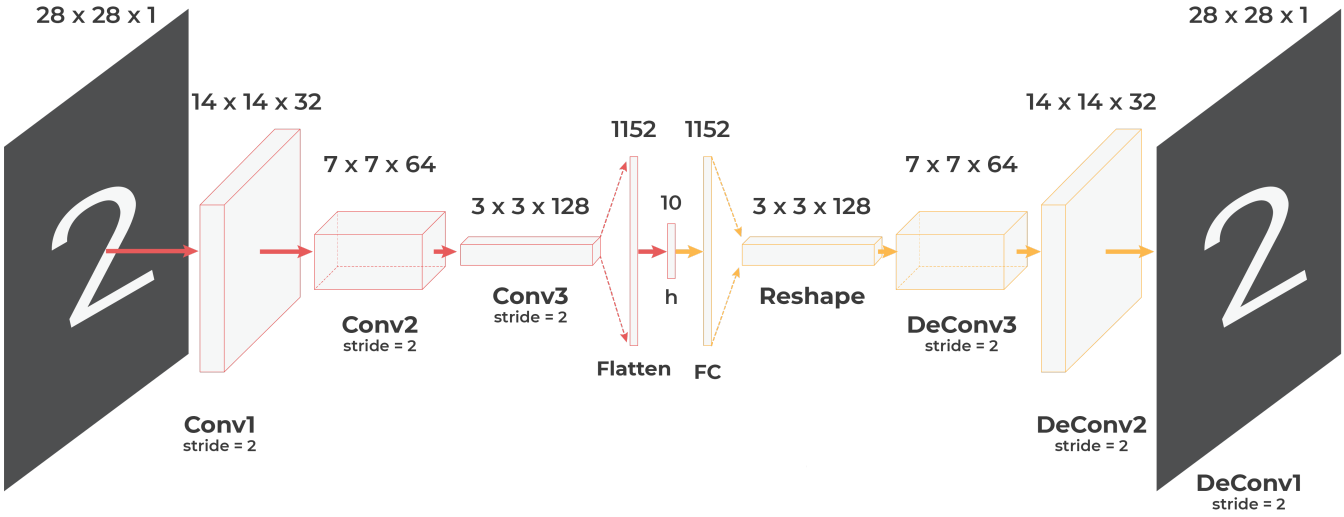

Figure 8: Convolutional autoencoder

Variational autoencoders

- Variational autoencoders are the generative models, unlike the classical models (sparse, denoising, etc.) autoencoders. VAE introduces probabilistic spin on autoencoders to let them generate new data by sampling.

- Hence, VAE can act as a generative model like Generative Adversarial Network (GAN), which gives significant control over the modeling of our latent distribution, unlike other models.

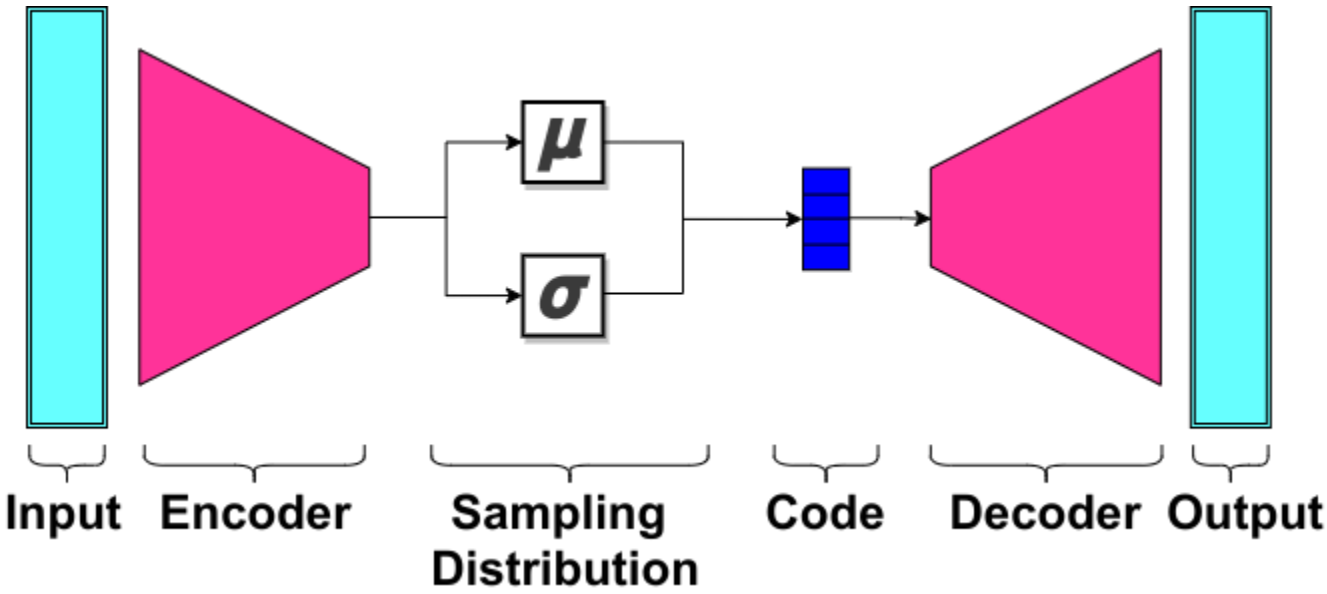

Figure 9: Variational autoencoder

For the detailed explanation about these encoders, you can refer to this link (https://iq.opengenus.org/types-of-autoencoder/).

Applications

1. Dimensionality reduction

> The main application of autoencoders is dimensionality reduction/ feature extraction. Transfering, as well as visualizing high-dimensional data, is challenging. So autoencoders are used as a preprocessing step to reduce dimensionality. This enables the transfer of lower complexity data through a connection. For visualization, this compressed representation is used by t-SNE to visualize the data in 2D space. To learn about t-SNE, Here (https://towardsdatascience.com/an-introduction-to-t-sne-with-python-example-5a3a293108d1) is a link for the explanation (if interested).

2. Image generation

> Let us understand how autoencoders help in the image generation with the figure below.

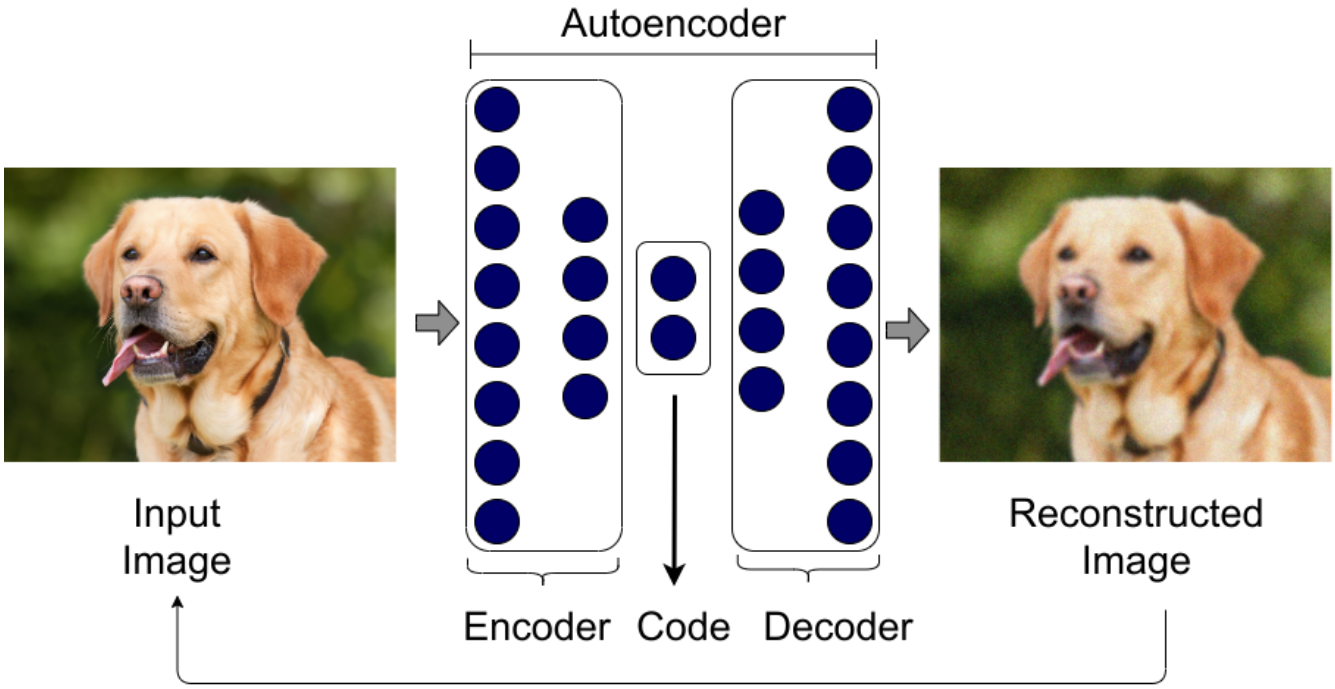

Figure 10: Image generation with autoencoder

Consider, we are generating the image of a dog. Encoder firstly compresses the image into a lower dimension. Unlike others, encoder results mean and variance rather than fixed code. Then, the decoder learns the important features from distribution. After that, the reconstructed image is compared with the original image to calculate loss and train the network accordingly. With the multiple numbers of training and sampling, the autoencoders can generate output images close to the input image.

2. Semantic Segmentation

> Self-driving cars use autoencoders to segment(detect and delineate) different objects in their field of vision. The architecture used here is known as Convolutional encoders, which uses convolutional layers in layers of autoencoders. Arxiv Insight shows the use of semantic segmentation in his youtube video on VAE, Link Here (https://www.youtube.com/watch?v=9zKuYvjFFS8)

3. Denoising task

> Noise present in the data is removed. Denoising autoencoders are used for this type of task. These autoencoders take corrupted input data and try to recover undistorted or clean input data. The main objective of this model is to learn good representation from the data and try to produce clean or original input irrespective of corrupted input data.

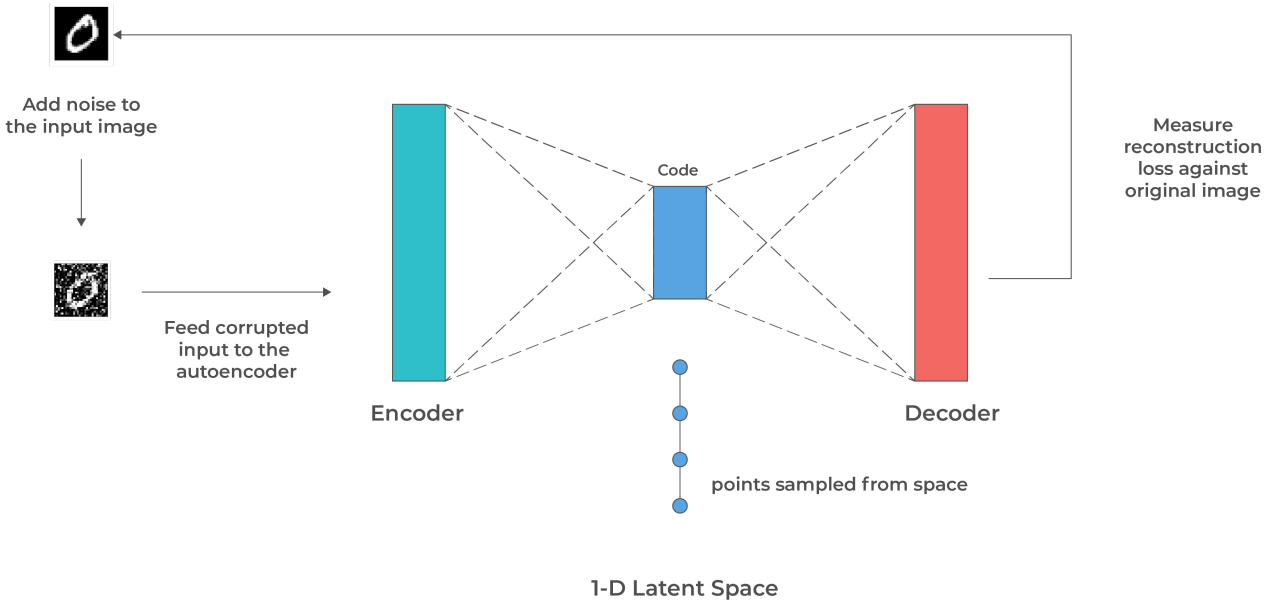

Figure 11: Denoising autoencoder

4. Variational Autoencoders (VAE)

> VAE is a more modern and sophisticated use-case of autoencoders. However, as a quick summary, VAE learns the parameters of the probability distribution modeling the input data instead of learning an arbitrary function in the case of vanilla autoencoders. By sampling points from this distribution, VAE can be used as a generative model.

5. Neural Inpainting

> Removing a part of the original image and replacing it with white or black pixels and later feeding into the network. The network has to reconstruct the original image. This type of approach enables us to remove water-marks, unwanted objects from images.

Key Takeaways

1. Autoencoders are unsupervised neural networks trying to reconstruct output as close to original input by dimensionality reduction in the encoder section of the unlabeled data.
2. Autoencoders are data-specific and lossy.
3. Code size, number of neurons per layer, number of layers, and loss function are the hyperparameters of the autoencoder.
4. Undercomplete autoencoder has a code size dimension lesser than the input size and exactly opposite for the overcomplete autoencoder.
5. In sparse autoencoder, neurons are randomly drop-off in different hidden layers.
6. In convolutional autoencoder, different convolutional layers are included in the encoder and decoder section.
7. Variational autoencoders are the generative models where sampling distribution occurs in between encoder and decoder.

References

- Papers
  - Pierre B. (2012), Autoencoders, Unsupervised Learning, and Deep Architectures (http://proceedings.mlr.press/v27/baldi12a/baldi12a.pdf)
    - Refer this paper to understand linear and non-linear autoencoders mathematically.
- Books
  - Ian Goodfellow, Deep Learning (Adaptive Computation and Machine Learning series) (https://www.amazon.com/Deep-Learning-Adaptive-Computation-Machine/dp/0262035618/ref=sr_1_1?ie=UTF8&qid=1472485235&sr=8-1&keywords=deep+learning+book)
    - Refer this chapter (https://www.deeplearningbook.org/contents/autoencoders.html) to read more about autoencoders mathematically.
- University lectures
  - Jean-Pierre B., Deep Learning Techniques for Music Generation Autoencoder (http://www-desir.lip6.fr/~briot/cours/unirio2/Slides/dlmg-4-autoencoder.pdf)
    - Check this slide to understand how autoencoders can implement in real-world problems.