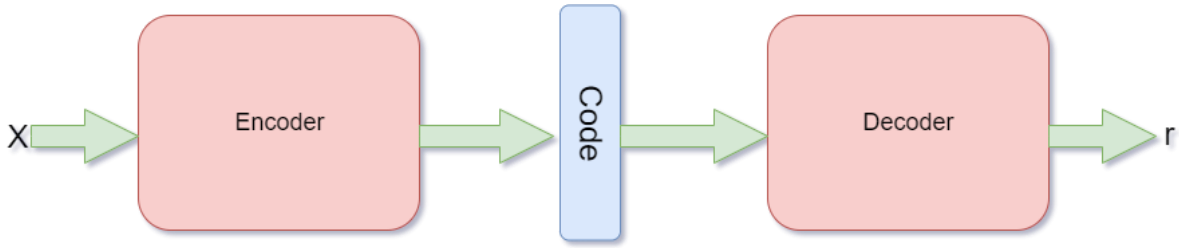
What are Autoencoders?

- Autoencoders are an unsupervised learning technique that we can use to learn efficient data encodings.
- Basically, autoencoders can learn to map input data to the output data.
- While doing so, they learn to encode the data. And the output is the compressed representation of the input data.

→The main aim while training an autoencoder neural network is dimensionality reduction.

"Autoencoding" is a data compression algorithm where the compression and decompression functions are 1) data-specific, 2) lossy, and 3) learned automatically from examples rather than engineered by a human.

The following image shows the basic working of an autoencoder. It is just a basic representation of the working of the autoencoder.



1.An autoencoder is a feed-forward neural net whose job it is to take an input x and predict x.

2 In another words, autoencoders are neural networks that are trained to copy their inputs to their outputs.

3 It consists of Encoder h = f (x) Decoder r = g(h)

h = f (x) and r = g(h)

Autoencoders consist of an encoder h = f (x) taking an input x to the hidden representation h and a decoder xˆ = g(x) mapping the hidden representation h to the input xˆ

- An autoencoder is a data compression algorithm.
- A hidden layer describes the code used to represent the input.
- It maps input to output through a compressed representation code.

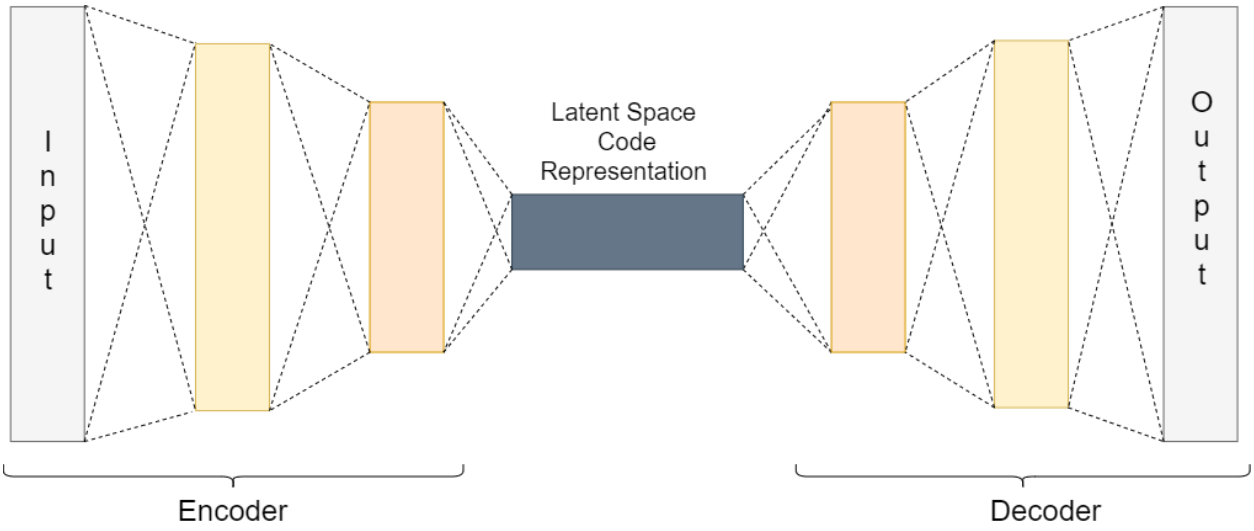
The Principle Behind Autoencoder

- In an autoencoder, there are two parts, an ***encoder,*** and a ***decoder***.
- First, the encoder takes the input and encodes it.
- For example, let the input data be x. Then, we can define the encoded function as f(x).
- Between the encoder and the decoder, there is also an ***internal hidden layer***.
- Let's call this hidden layer h. This hidden layer learns the coding of the input that is defined by the encoder.
- So, basically after the encoding, we get h = f(x).

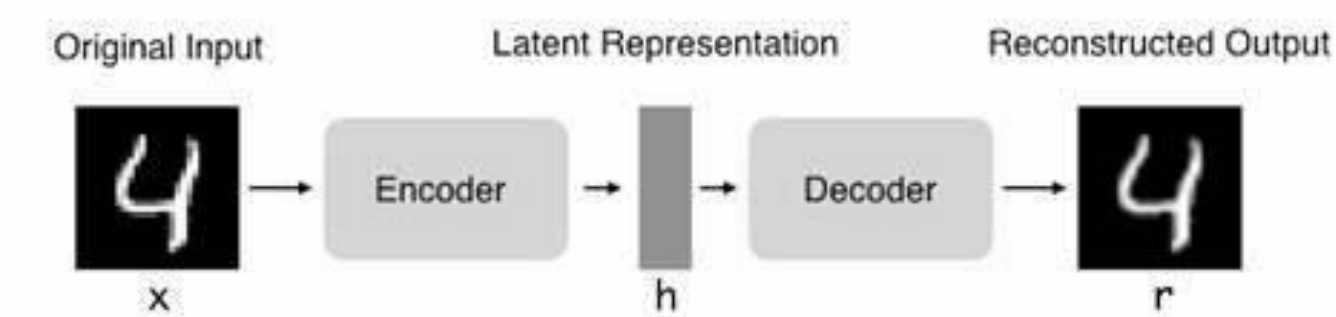
- Finally, the decoder function tries to reconstruct the input data from the hidden layer coding. If we consider the decoder function as g,

- then the reconstruction can be defined as,

$$r=g(f(x)) \qquad r=g(h)$$



The latent space is simply **a representation of compressed data in which similar data points** are closer together in space. Latent space is useful for learning data features and for finding simpler representations of data for analysis.



An autoencoder should be able to reconstruct the input data efficiently but by learning the useful properties rather than memorizing it.

1. Autoencoders are data specific They are 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

2. Autoencoder for pictures of cats would do poorly in compressing pictures of trees Because features it would learn would be cat specific.

3. Autoencoders are lossy

   This means that the decompressed outputs will be degraded compared to the original inputs (similar to MP3 or JPEG compression). This differs from loss less arithmetic compression

Training autoencoder

1 An autoencoder is a feed-forward non-recurrent neural network. With an input layer, an output layer and one or more hidden layers.

2. It can be trained using the following technique. Compute gradients using backpropagation Followed by mini-batch gradient descent.

→There are many ways to capture important properties when training an autoencoder.

1. Undercomplete Autoencoders

- we discussed that we want our autoencoder to learn the important features of the input data.

- It should do that instead of trying to memorize and copy the input data to the output data.

- We can do that if we make the hidden coding data to have less dimensionality than the input data.

- In an autoencoder, when the encoding h has a smaller dimension than x, then it is called an undercomplete autoencoder.

→The above way of obtaining reduced dimensionality data is the same as PCA. In PCA also, we try to try to reduce the dimensionality of the original data.

The loss function for the above process can be described as,

$L(x,r)=L(x,g(f(x)))$

where L is the loss function. This loss function applies when the reconstruction r is dissimilar from the input x.

2.Regularized Autoencoders

In undercomplete autoencoders, we have the coding dimension to be less than the input dimension.

We also have ***overcomplete*** autoencoder in which the coding dimension is the same as the input dimension. But this again raises the issue of the model not learning any useful features and simply copying the input.

One solution to the above problem is the use of ***regularized autoencoder***. When training a regularized autoencoder we need not make it undercomplete. We can choose the coding dimension and the capacity for the encoder and decoder according to the task at hand.
To properly train a regularized autoencoder, we choose loss functions that help the model to learn better and capture all the essential features of the input data.

Next, we will take a look at two common ways of implementing regularized autoencoders.

2.1sparse Autoencoders

In ***sparse autoencoders,*** we use a loss function as well as an additional penalty for sparsity. Specifically, we can define the loss function as,

$$L(x,g(f(x))) + \Omega(h)$$

where $\Omega(h)$ is the additional sparsity penalty on the code h.

The following is an image showing MNIST digits. The first row shows the original images, and the second row shows the images reconstructed by a sparse autoencoder.

Working of Sparse Autoencoders (Source)

Adding a penalty such as the sparsity penalty helps the autoencoder to capture many of the useful features of data and not simply copy it.

2.2Denoising Autoencoders

In sparse autoencoders, we have seen how the loss function has an additional penalty for the proper coding of the input data.

But what if we want to achieve similar results without adding the penalty? In that case, we can use something known as ***denoising autoencoder***.

We can change the reconstruction procedure of the decoder to achieve that. Until now we have seen the decoder reconstruction procedure as $r(h) = g(f(x))$ and the loss function as $L(x,g(f(x)))$.

Now, consider adding noise to the input data to make it x~ instead of x. Then the loss function becomes, L(x,g(f(x~)))

The following image shows how denoising autoencoder works. The second row shows the reconstructed images after the decoder has cleared out the noise.

Working of Denoising Autoencoder (Source)

For a proper learning procedure, now the autoencoder will have to minimize the above loss function. And to do that, it first will have to cancel out the noise, and then perform the decoding.

In a denoising autoencoder, the model cannot just copy the input to the output as that would result in a noisy output. While we update the input data with added noise, we can also use overcomplete autoencoders without facing any problems.

3.Variational Autoencoders (VAEs)

VAEs are a type of generative model like **GANs (Generative Adversarial Networks)**.

NOTE:

- A generative adversarial network, or GAN, is a deep neural network framework which is able to learn from a set of training data and generate new data with the same characteristics as the training data.
- For example, a generative adversarial network trained on photographs of human faces can generate realistic-looking faces which are entirely fictitious.
- Generative adversarial networks consist of **two neural networks**, the **generator and the discriminator,** which compete against each other. The generator is trained to produce fake data, and the discriminator is trained to distinguish the generator's fake data from real examples. If the generator produces fake data that the discriminator can easily recognize as implausible, such as an image that is clearly not a face, the generator is

penalized. Over time, the generator learns to generate more plausible examples.

Like other autoencoders, variational autoencoders also consist of an encoder and a decoder. But here, the decoder is the generator model. Variational autoencoders also carry out the reconstruction process from the latent code space. But in VAEs, the latent coding space is continuous.

Applications and Limitations of Autoencoders in Deep Learning

Applications of Deep Learning Autoencoders
First, let's go over some of the applications of deep learning autoencoders.

Dimensionality Reduction and PCA
When we use undercomplete autoencoders, we obtain the latent code space whose ***dimension is less than the input***.

Image Denoising and Image Compression
Denoising autoencoder can be used for the purposes of image denoising. Autoencoders are able to cancel out the noise in images before learning the important features and reconstructing the images.

Autoencoder can also be used for image compression to some extent. More on this in the limitations part.

Information Retrieval
When using deep autoencoders, then reducing the dimensionality is a common approach. This reduction in dimensionality leads the encoder network to capture some really important information

Limitations of Autoencoders

We have seen how autoencoders can be used for image compression and reconstruction of images. But in reality, they are not very efficient in the process of compressing images. Also, they are only efficient when reconstructing images similar to what they have been trained on.

Due to the above reasons, the practical usages of autoencoders are limited.