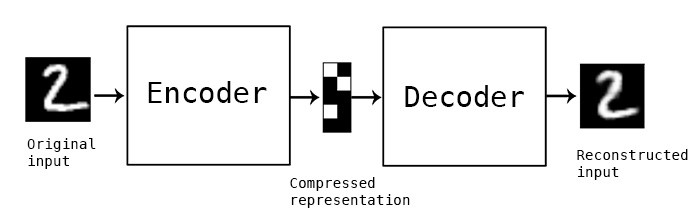
[**https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726**](https://towardsdatascience.com/auto-encoder-what-is-it-and-what-is-it-used-for-part-1-3e5c6f017726)

**Autoencoder**, by design, reduces data dimensions by learning how to ignore the noise in the data.

Autoencoder is an unsupervised artificial neural network that learns how to efficiently compress and encode data then learns how to reconstruct the data back **from** the reduced encoded representation **to** a representation that is as close to the original input as possible.



## Autoencoder Components:

Autoencoders consists of 4 main parts:

1- **Encoder**: In which the model learns how to reduce the input dimensions and compress the input data into an encoded representation.

2- **Bottleneck**: which is the layer that contains the compressed representation of the input data. This is the lowest possible dimensions of the input data.

3- **Decoder**: In which the model learns how to reconstruct the data from the encoded representation to be as close to the original input as possible.

4- **Reconstruction** **Loss**: This is the method that measures measure how well the decoder is performing and how close the output is to the original input.

The training then involves using back propagation in order to minimize the network’s reconstruction loss.

Autoencoder Architecture: FeedForward network, LSTM network or Convolutional Neural Network

# 1- Autoencoder for Anomaly Detection: There are many ways and techniques to detect anomalies and outliers

2- Image Denoising:

<https://towardsdatascience.com/tf-term-frequency-idf-inverse-document-frequency-from-scratch-in-python-6c2b61b78558>

Table of Contents:

Terminology .

* Term Frequency(TF) .
* Document Frequency .
* Inverse Document Frequency .

Implementation in Python .

1 - Terminology :

t — term (word)

d — document (set of words)

N — count of corpus

corpus — the total document set

the number of times a term occurs in a document is called its term frequency

tf(t,d) = count of t in d / number of words in d

df(t) = occurrence of t in documents

Idf(t) = log(N/(df + 1))

tf-idf(t, d) = tf(t, d) \* log(N/(df + 1))

Feature engineering Techniques:-

1.Imputation

2.Handling Outliers

3.Binning

4.Log Transform

5.One-Hot Encoding

6.Grouping Operations

7.Feature Split

8.Scaling

9.Extracting Date

The most simple solution to the missing values is to drop the rows or the entire column. There is not an optimum threshold for dropping but you can use **70%** as an example value and try to drop the rows and columns which have missing values with higher than this threshold.

threshold = 0.7

**#Dropping columns with missing value rate higher than threshold**  
 data = data[data.columns[data.isnull().mean() < threshold]]  
  
**#Dropping rows with missing value rate higher than threshold**  
 data = data.loc[data.isnull().mean(axis=1) < threshold]

Numerical Imputation

Categorical Imputation