AutoEncoders

Achint Soni and Sai Keerthi PV

What are Autoencoders?

We all know about neural networks. If the neural network is given some input x, it passes through the network and produces some output y, which essentially is either a class number, regression, feature vector, embedding etc.
Now, the task at hand is data compression or dimensionality reduction.
Goal: Data compression or dimensionality reduction because we know that a data lies in a low dimensional manifold.
Example 1: Let's say our data contains some images like this:

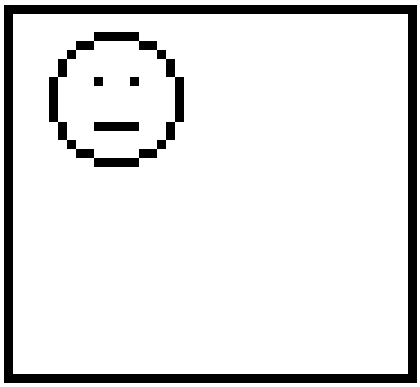

Figure 1: Image 1

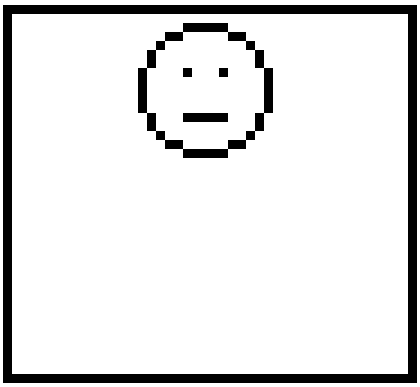

Figure 2: Image 2

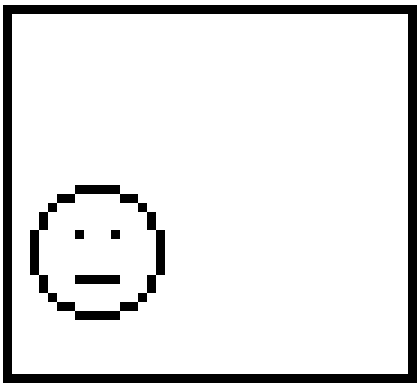

Figure 3: Image 3

If we want to draw 3, we can simply take 1 and move the face downwards. Which means that if we know the coordinates of the centre of faces of the two images, those two dimensions are sufficient to draw an image.

Example 2: For the given images: The data lies on very small manifold because several things

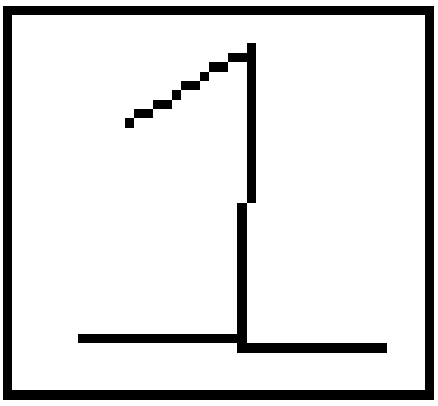

Figure 4: Example 2, image 1

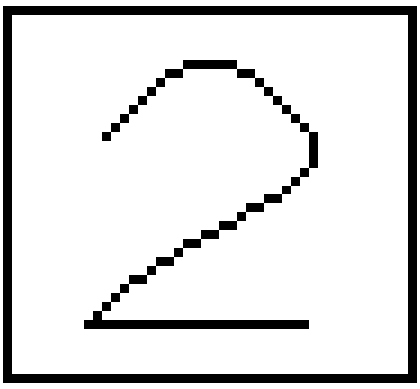

Figure 5: Example 2, image 2

are not possible at all. Let's say,

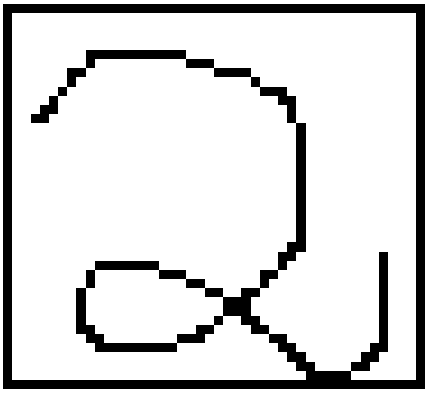

Figure 6: This is not possible.

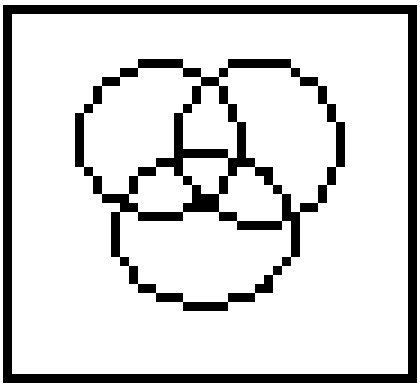

Figure 7: This is not possible.

Both these images are not possible to create from the given dataset of images containing figure 4 and 5. If the images are 28×28, than it means that our data is not 28×28 complete dimensions.
Example 3:
PCA on this data gives us a line and all the data lies on this line because we know that this is the

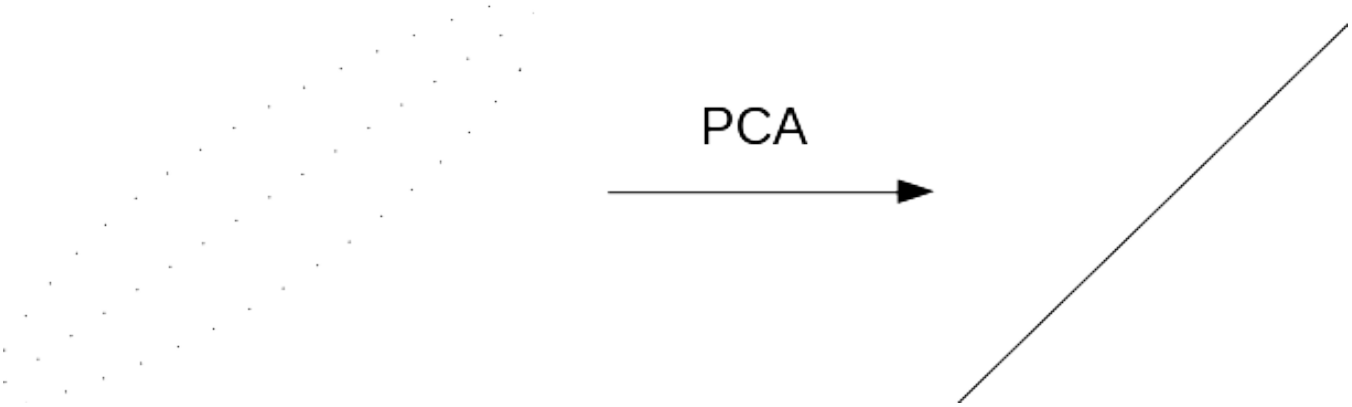

Figure 8: PCA

dimension which has got the maximum amount of information.

When we know our data is not uniformly spread on the entire space, how can we find out smaller subspace where our data is lying or how can we find a low dimensional representation of data?
That is the main goal of autoencoders.
So basically, autoencoders is a concept which implements similar concept of data compression or dimensionality reduction.

How does it do that?

Let's say we have some input vector x and we pass that x through a neural network to map it to x itself. Let's say that x has 100 dimensions. But in autoencoders, we make a bottleneck layer(a layer of smaller dimensions).
The dimensionality of z, which we can also say as length as z is very very small as compared to length of x.

$$len(z) << len(x)$$

Figure 9: Basic visualization of Autoencoder

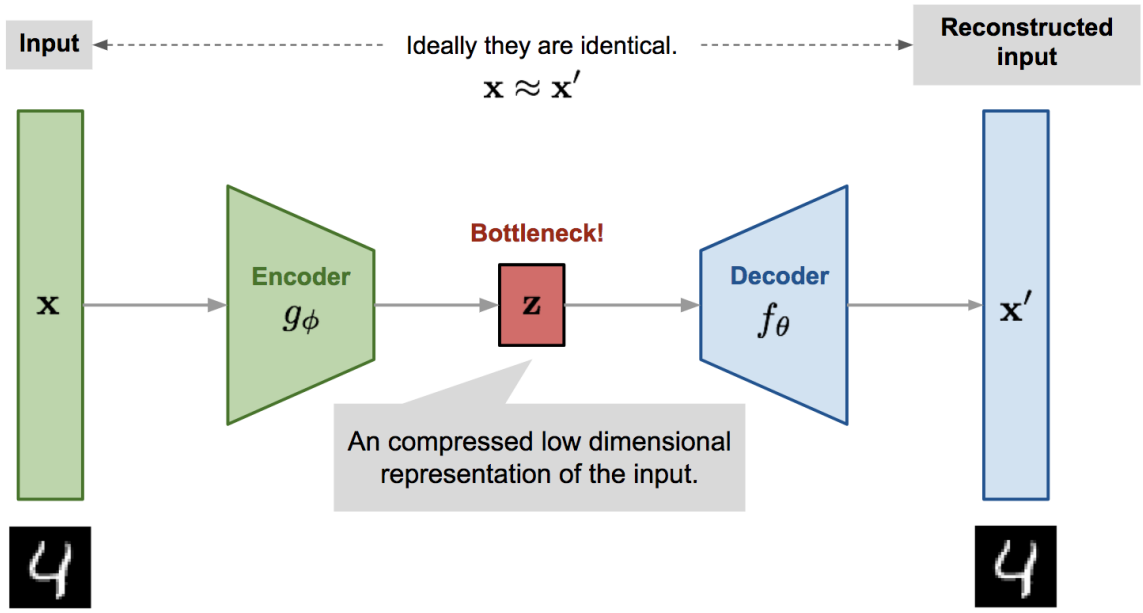

Figure 10: autoencoder architecture

So basically, all the information to reconstruct x is present in z.
In PCA, there's a matrix A such that if x is a 10×1 matrix and if we want y to be a 2×1 matrix, then:

$$A_{2\times10}.x_{10\times1} = y_{2\times1}$$

Further, if we want to reconvert y back into x, there is a matrix B such that:

$$B_{10\times2}.y_{2\times1} = x_{10\times1}$$

11 represents compression of data into a straight line with PCA and reconversion of the straight line back into data but with loss.
Compression of original data is termed as encoding where we take the data and encode it into compressed representation whereas reconstruction of the compressed data to original data is done with the help of decoder.
Similar terminology is used in autoencoders.
10 represents the image of an autoencoder where the part of neural network before the bottleneck is the encoder whereas the part after the bottleneck layer is the decoder.

But why does autoencoder has a term encoder?

Because it is encoding the information into some latent representation.

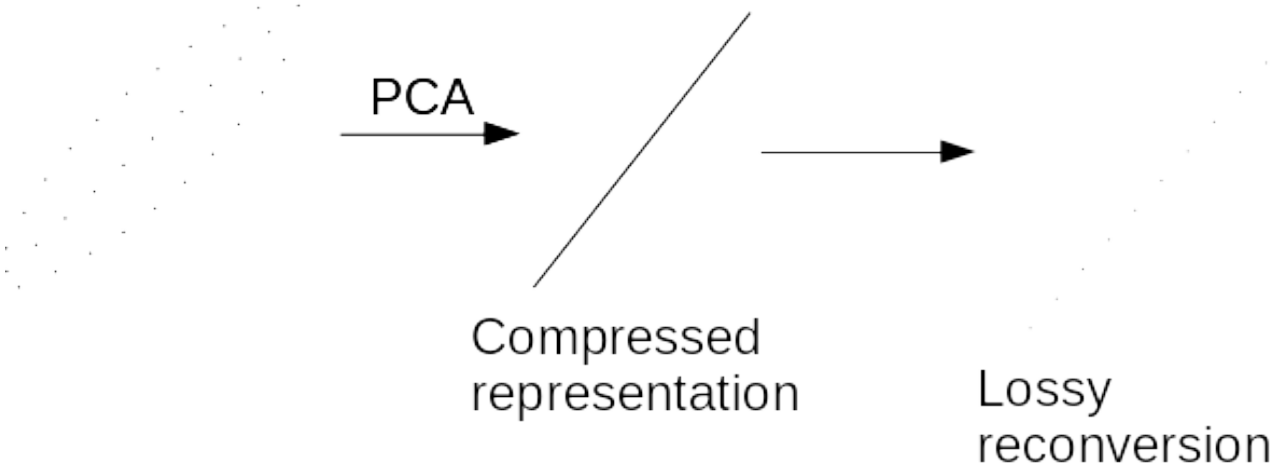

Figure 11: PCA on data and reconversion

Then why is it called autoencoder?

Because it is kind of encoding itself and then it can be decoded to retrieve the same x.

What are the limitations of this model?
The major drawback of autoencoders is that it does not tell us the probability density function(pdf) of the data.

GANs

Autoencoders have encoder and decoder part in the neural network whereas GANs only comprise of the decoder part. GANs are relatively new content compared to autoencoders.

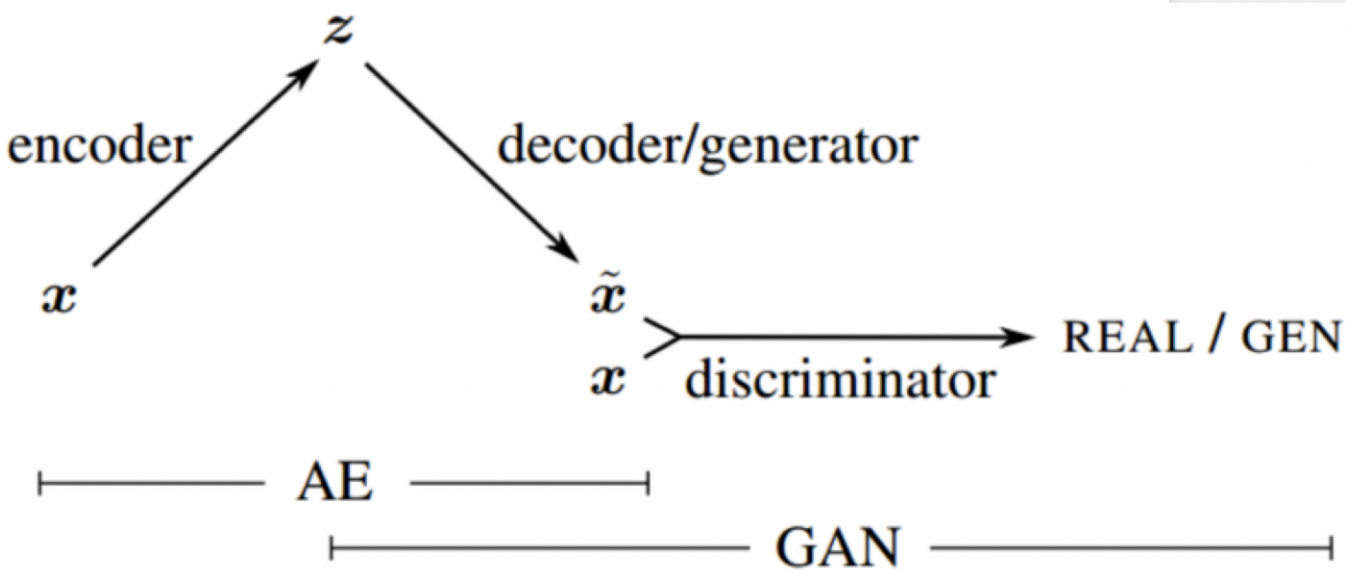

Figure 12: AE and GAN relation

Modelling the autoencoders

To model the autoencoders, we hae to define a simple NN which could have convolutional layers, dense layers, recurrent layers if the data is sequential etc.

Constraints

- Must have same input and output dimensions.
  #neurons in output layer = #neurons in input layer.
- There is a bottleneck layer which is actually the encoded representation. The length of that layer must be smaller than the length of the input layer.

$$len(z)<<len(x)$$

Training autoencoders

In the beginning, the weights are random and x will map to some arbitrary $\hat{x}$ which has no relation to x.
If,

$$x \in \mathbb{R}^D$$

We can define a reconstruction loss of the form

$$(\hat{x} - x)^2$$

This is a element wise square and then we sum up all the elements.
Non linearity of the output layer for this loss function can be Leaky ReLU.

But is Leaky ReLU a good option?

x can have both positive and negative values. It will have a kind of uniform distribution. If we normalize the values of x in the beginning such that mean is 0 and standard deviation is 1, then x is a uniformly spread distribution. But unlike x, leaky ReLU is skewed since it is $y = x$ for $x > 0$ and $y = \alpha x$ where $\alpha \approx 0$ for $x < 0$.
It would make more sense if we use a linear function in the output layer instead of non linearity.

NOTE:WE ARE TALKING ABOUT USING LINEAR FUNCTION ONLY IN THE **OUTPUT LAYER**, NOT INTERMEDIATE LAYERS. They can have whatever non linearity we want.

This was the can when

$$x \in \mathbb{R}^D$$

But if,

$$x \in \{0, 1\}^D$$

Example for binary data is MNIST dataset.
Reconstruction loss that works well for binary data is binary cross entropy loss. Whereas non linearity in the output layer can be sigmoid function.

Applications of Autoencoders

1. Anomaly detection: Property of the Autoencoder as a generative model which encodes input into a low dimensional latent representation which then can be decoded to generate back the original data with some loss is used for this application.
   Consider the example of PCA in 13 Here the input is in 2 Dimensional space. Encoder maps the points to some points on 1 Dimensional space. Here the output is similar to training data but not to input data. The red point in the 13 will not be regenerated, that information is lost.

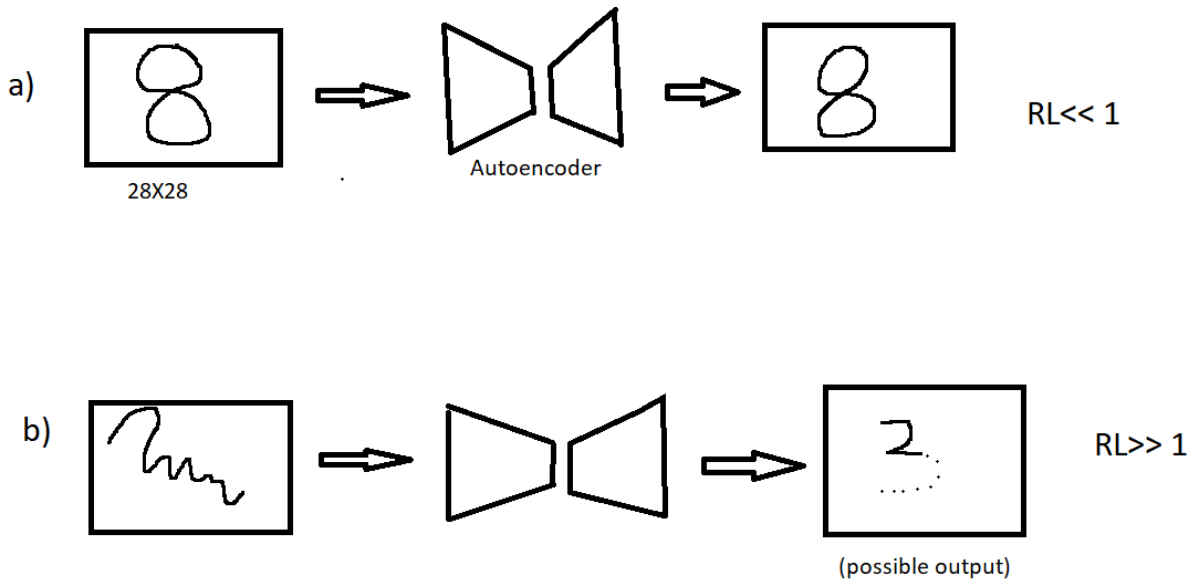

Figure 13: Autoencoder testing on MNIST data

Taking from above consider the example of training Autoencoder on MNIST data. The results while testing new inputs is given in 14 As we can observe in b) the decoder will map to a low dimensional manifold of original training set, so the mapping is to digits 2 or 3.
Reconstruction loss for the above example defined by

$$\sum_{d}(\hat{x} - x)^2$$

in a) is less than 1 whereas it is much greater than 1 in b).Reconstruction loss for anomalous data is greater than non anomalous data.

Situations where anomaly detection is useful are:

- Face recognition
- Traffic monitoring

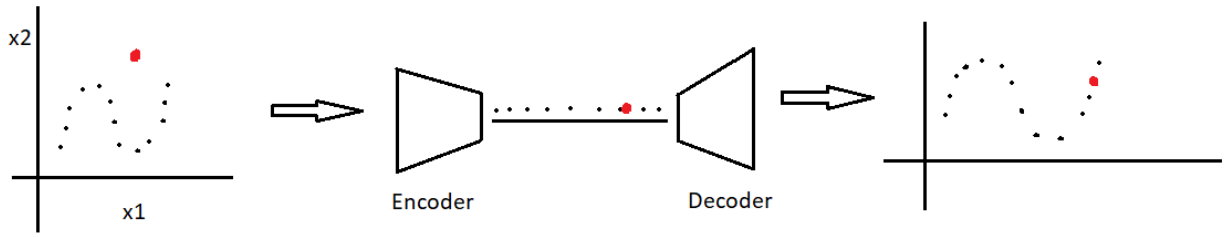

Figure 14: PCA

- Audio monitoring of machines etc.

2. Denoising: Property of the Autoencoder which given any $\vec{x}$ off from from the original manifold can be brought back to that manifold.
   Example: Image denoising application on MNIST dataset given in 15. This process is very ef-

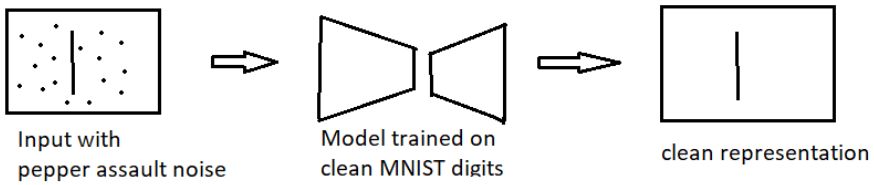

Figure 15: Image Denoising

ficient compared to finding which image input corresponds to by searching through a database.

Another approach to the same is to train the Autoencoder to map noisy to denoised $\vec{x}$ as shown in 16.

Implementation on Tensorflow

Implementation of Autoencoder can be referred to Tensorflow documentation. Here it is implemented on Fashion MNIST dataset which is grey scale images of objects like dress, shoes etc.
Subclass Autoencoder of Model class is defined. latent_dim, input to constructor of Autoencoder class is the size of $\vec{z}$. Encoder and decoder architecture can be chosen. Dense_layer or convolutional networks can be used. Sigmoid activation function is used in output as data values are continuous and lie between 0 and 1.

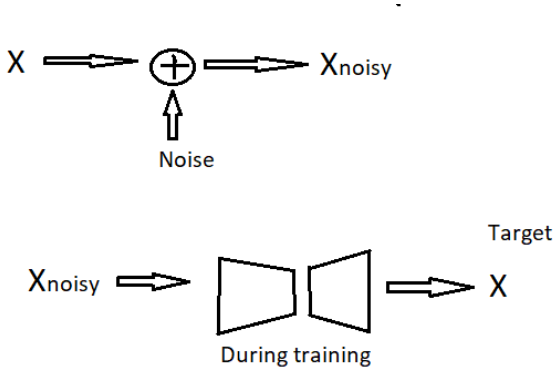

Figure 16: Alternate approach to Image denoising

Optimizer and loss function are specified while compiling the Autoencoder model. Model is fit using gradient descent.
In Image Denoising example, Gaussian noise with some standard deviation is added to input images.