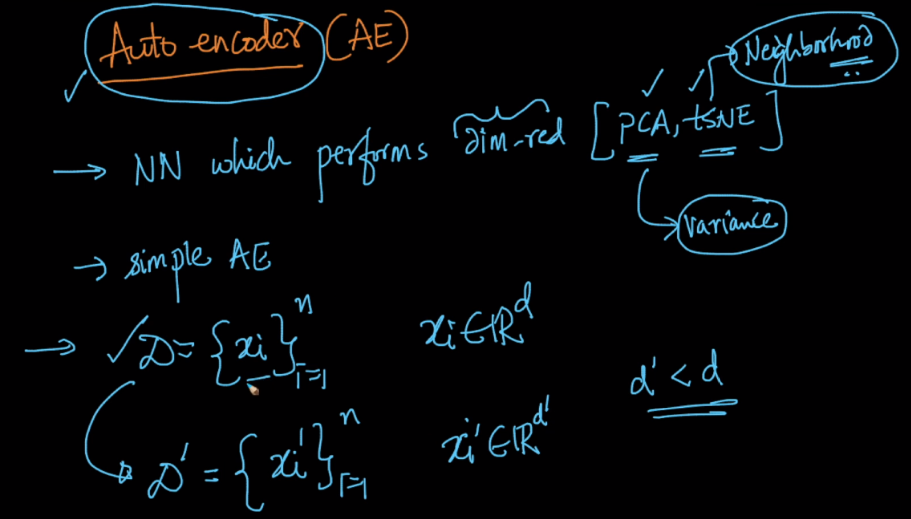
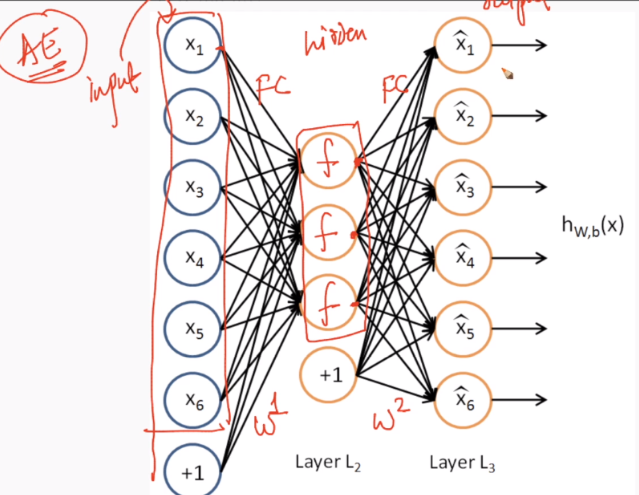
Autoencoders are mainly a dimensionality reduction (or compression) or we can say it’s a neural network which performs dimension reduction like PCA or T-SNE

Autoencoders are a specific type of feedforward neural networks where the input is the same as the output. They compress the input into a lower-dimensional *code*and then reconstruct the output from this representation. The code is a compact “summary” or “compression” of the input, also called the *latent-space representation.*



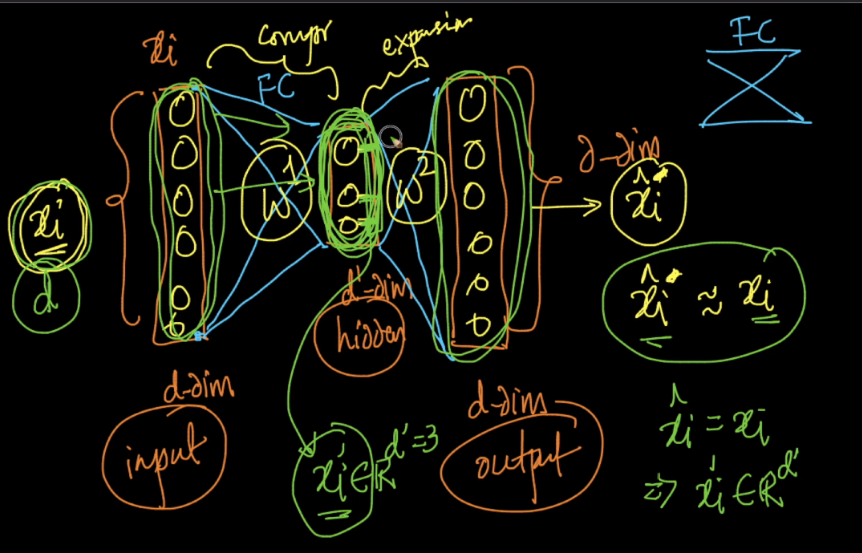


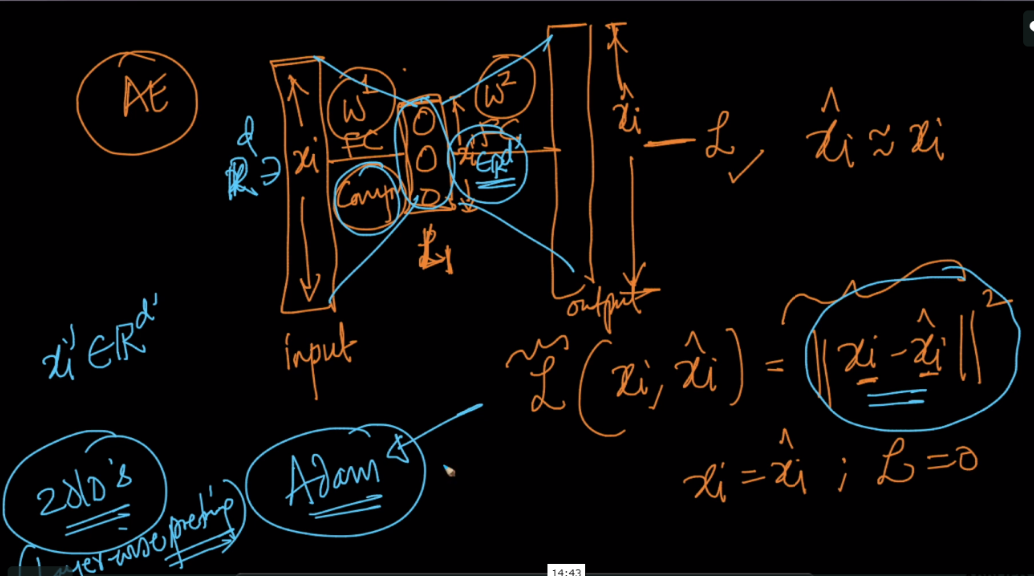
The idea is we have one hidden layer, and we want to match input to output. So basically we are first doing compression from input to hidden layer and the decompression from hidden to output layer.

If input and output become same that means hidden layer is able to reconstruct the original data with less no of feature ( we can have more no. of features in hidden layer than original no. of features, but only problem is that NN might won’t learn something it simply do copy and paste).

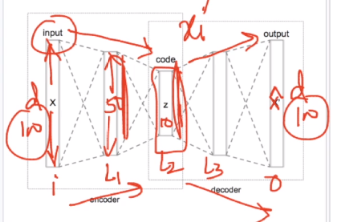
Once we’ve trained model then we’ll remove output layer, and used hidden layer output as data with fewer dimensions.

Since we are trying to make output similar to input so what should be the loss function? The loss function will be (x^ - x)2.





**We’ve seen a AE with one hidden layer, we can make it more intelligent by adding more no. of layers.**

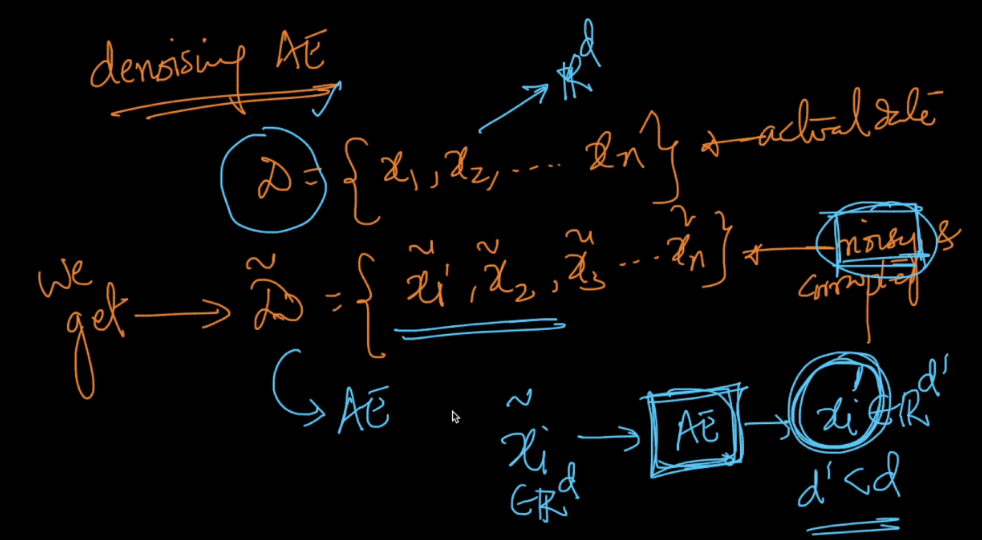


Here L2 is the encoded layer and we’ll use this layer output as data with fewer features.

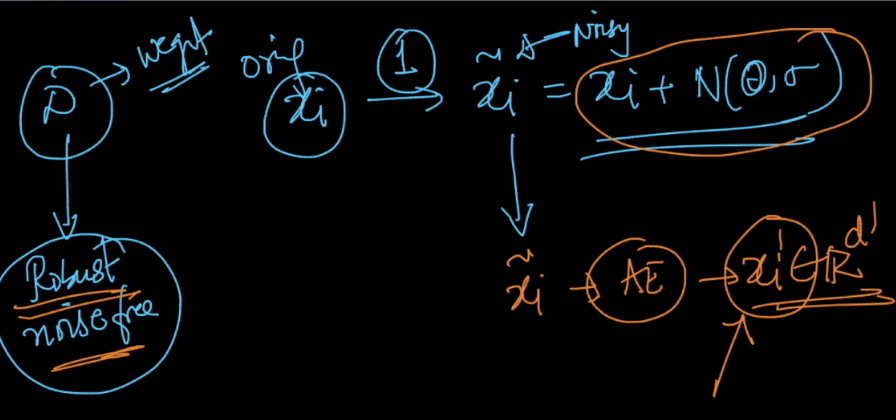
**Denoising AE:**

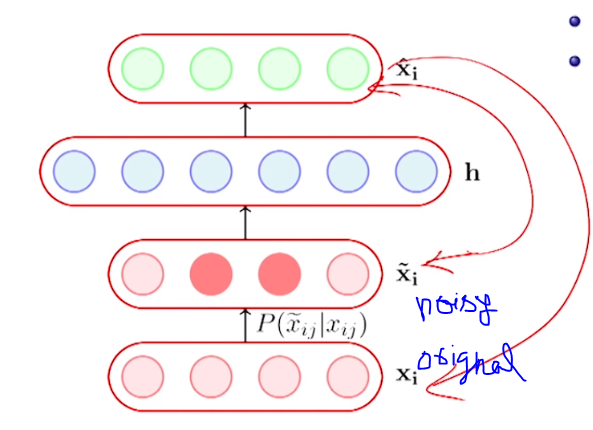
There is one problem with simple AE that it learn on training data but when it’s used on test data it won’t perform good if there is noise, because in real world there is always some noise.

Another problem is when there are more nodes in the hidden layer than there are inputs, the Network is risking to learn the so-called “Identity Function”, also called “Null Function”, meaning that the output equals the input, marking the Autoencoder useless. Denoising Autoencoders solve this problem by corrupting the data or adding some noise to the original data while training. When calculating the Loss function, it is important to compare the output values with the original input, not with the corrupted input because we want the output layer to output same as original input and not as corrupted input.



Adding noise(normal distributed) to real data.

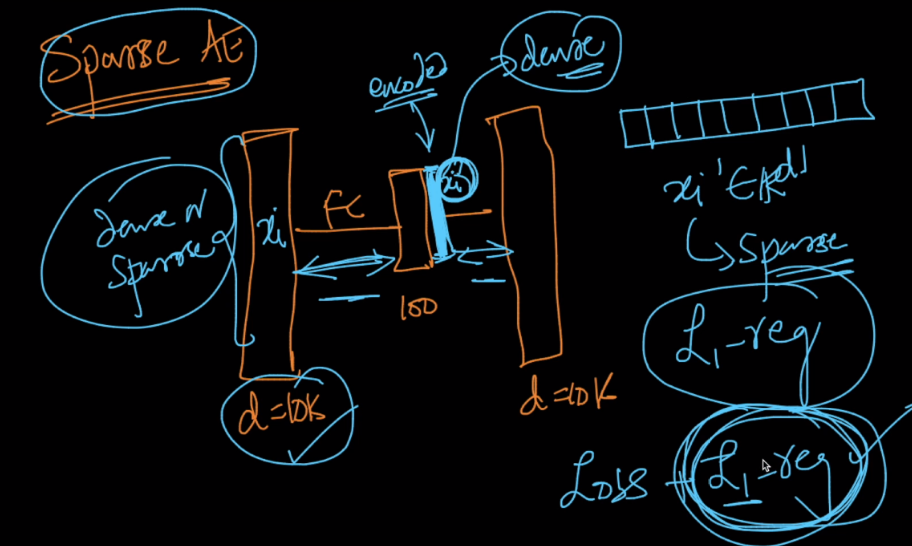




**Sparse AE:**

The another AE is using *regularization*. We can regularize the autoencoder by using a *sparsity constraint* such that only a fraction of the nodes would have nonzero values, called active nodes.

In particular, we add a penalty term to the loss function such that only a fraction of the nodes become active. This forces the autoencoder to represent each input as a combination of small number of nodes, and demands it to discover interesting structure in the data. This method works even if the code size is large, since only a small subset of the nodes will be active at any time.



<https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2>

<https://towardsdatascience.com/denoising-autoencoders-explained-dbb82467fc2>