Neural Networks for Signal Processing-I

Prof. Shayan Srinivasa Garani

Department of Electronic System Engineering

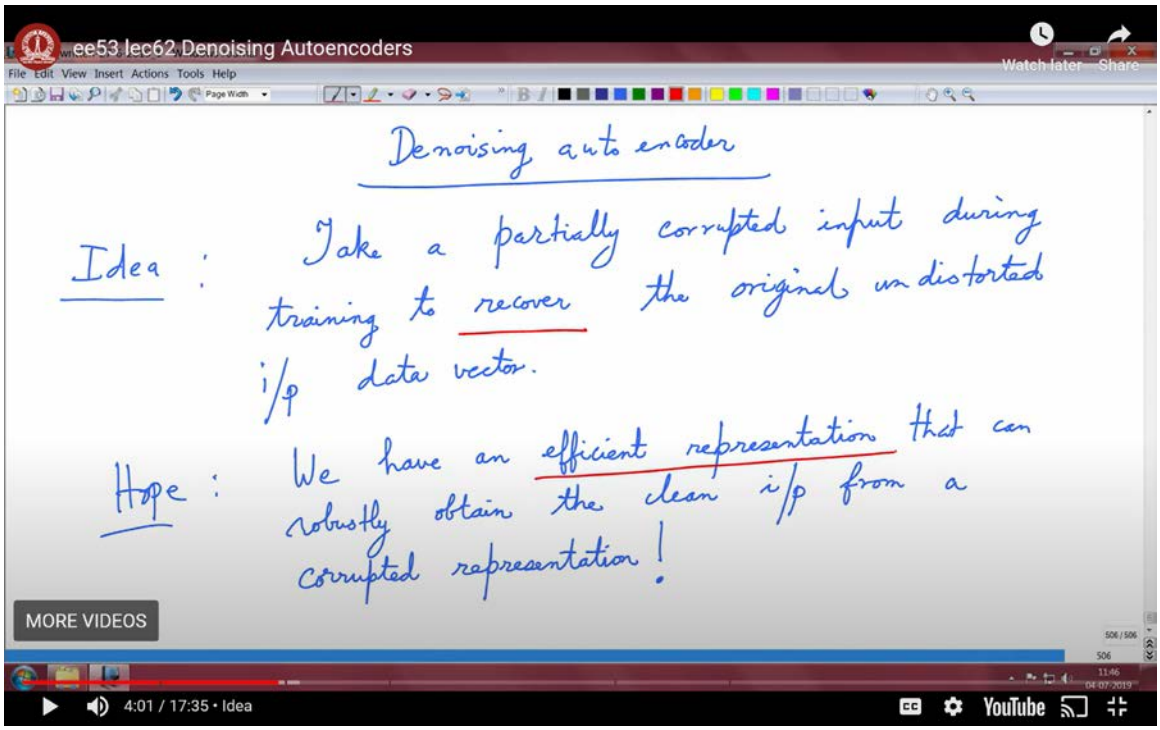
Indian Institute of Science, Bengaluru

Lecture – 62

Denoising Autoencoders

Having explored autoencoders in a qualitative manner and understood their structure and mappings, let's delve into denoising autoencoders. The concept is quite intuitive. Instead of working with the original, clean data vector, we introduce a vector that has been corrupted by noise. The goal is to learn mappings from these noisy inputs, aiming to create a representation that is robust to noise. During the decoding phase, the idea is to effectively "denoise" the input, projecting the noisy data onto a learning manifold to recover the clean input.

(Refer Slide Time: 04:01)

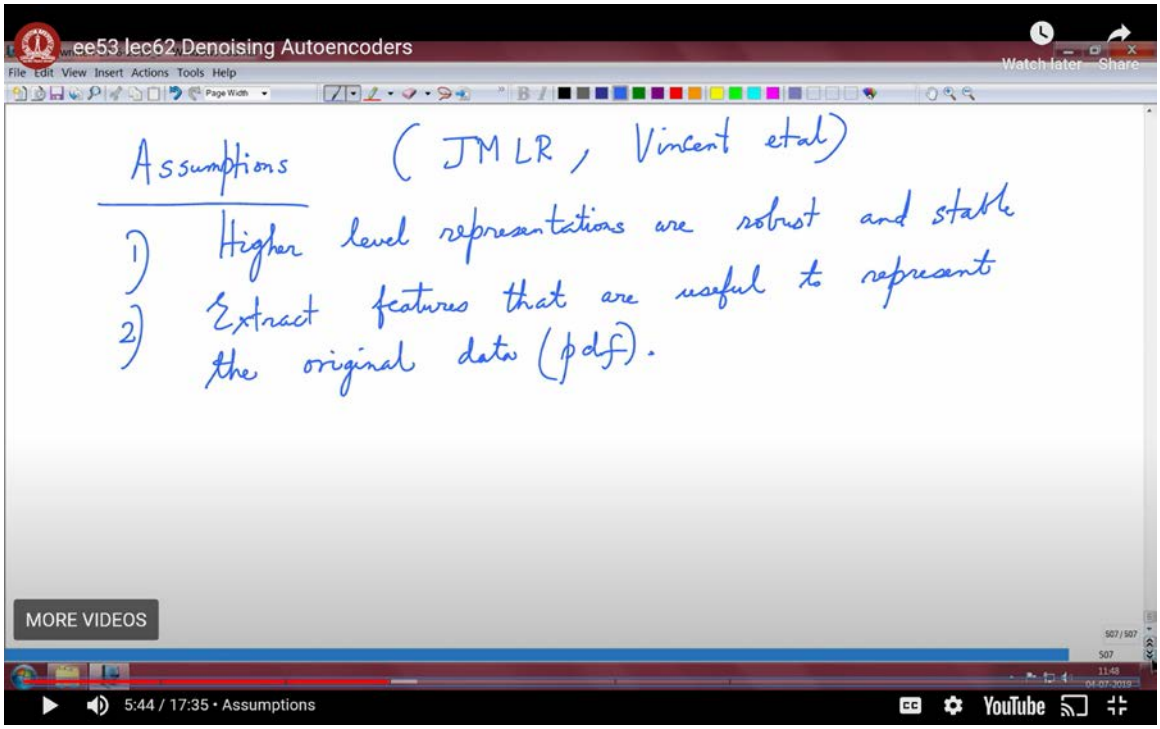


The key difference between traditional autoencoders and denoising autoencoders is that in the latter, we start with corrupted inputs during training, rather than clean ones. This approach is intended to yield a more robust mapping, improving the model's ability to handle noisy data.

Here’s the crux of denoising autoencoders:

- During training, we use partially corrupted inputs to train the model to recover the original, undistorted data vector.
- The objective is to develop an efficient representation that can reliably reconstruct the clean input from its corrupted version.

(Refer Slide Time: 05:44)



Several assumptions underpin this approach:

1. Higher-level representations are robust and stable.

2. The features extracted are effective in representing the original data distribution.

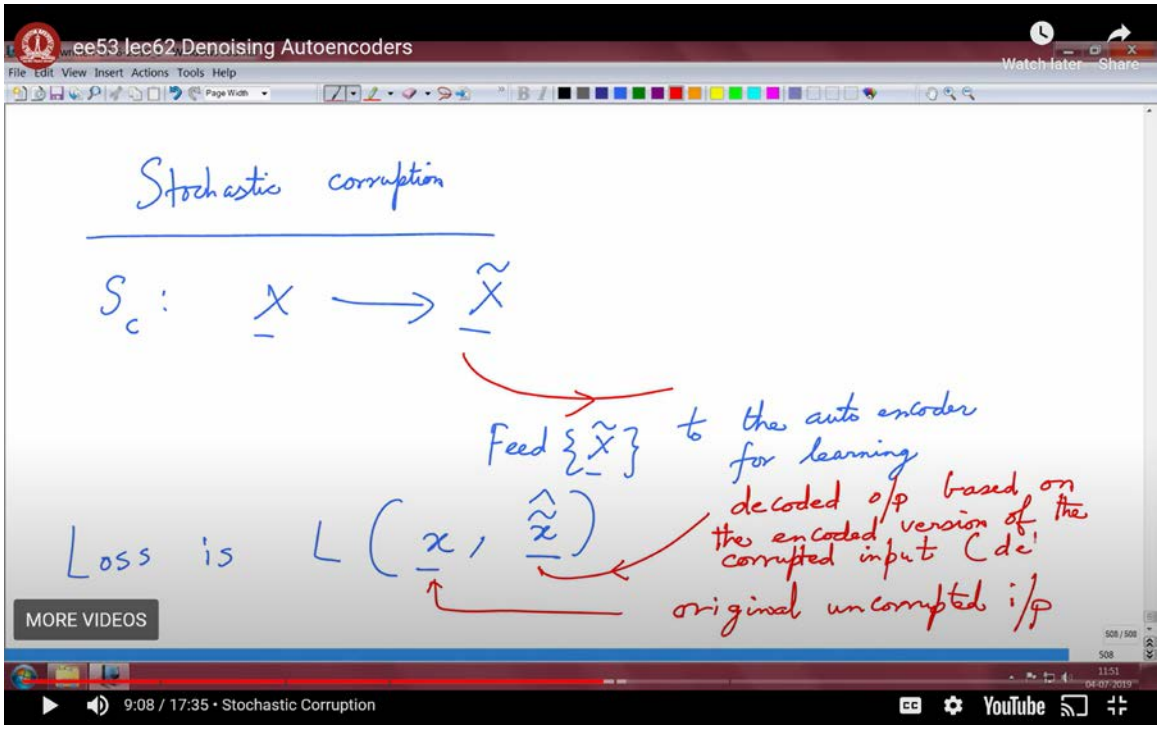
These assumptions are discussed in the paper by Vincent et al., published in the Journal of Machine Learning Research. For further details, you might refer to that paper or consult a comprehensive book on deep learning.

The process involves stochastic corruption of the data. Specifically:

1. A data vector x is transformed into $\tilde{x}$ through a stochastic corruption function SC.

2. The corrupted version $\tilde{x}$ is then fed into the autoencoder for training.

(Refer Slide Time: 09:08)



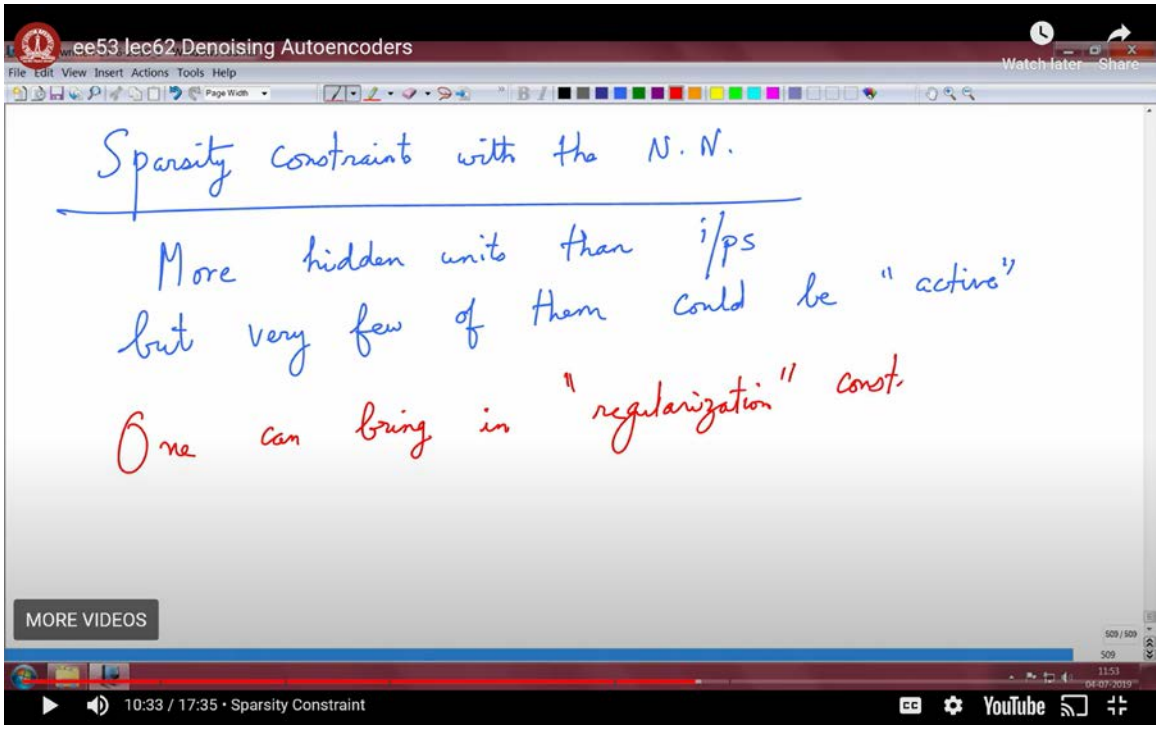
The autoencoder learns to reconstruct the original, clean data from this corrupted input. However, the loss function is not measured between the corrupted input and its decoded output. Instead, the loss is computed between the original clean input and the decoded output obtained from the corrupted input.

To express this formally, the loss function is denoted as $L(x, \hat{x})$, where x is the original, clean input, and $\hat{x}$ is the decoded output based on the encoded version of the corrupted

input $\tilde{x}$. This setup ensures that the model is optimized to recover the original data from the noisy input, thereby achieving effective denoising.

Incorporating sparsity constraints into neural networks can be quite beneficial. For instance, it's possible to have a network with more hidden units than input units, but only a small subset of these hidden units might be active at any given time. This can result in what are known as "dead neurons", hidden units that do not contribute to the learning process.

(Refer Slide Time: 10:33)



To address this issue, one can introduce regularization constraints into the optimization function to eliminate these non-contributing neurons. This is an important consideration when designing and solving network problems or research tasks. While I've discussed this qualitatively, you can set up a constraint optimization problem to find the optimal parameters for your network, ensuring the removal of inactive neurons.

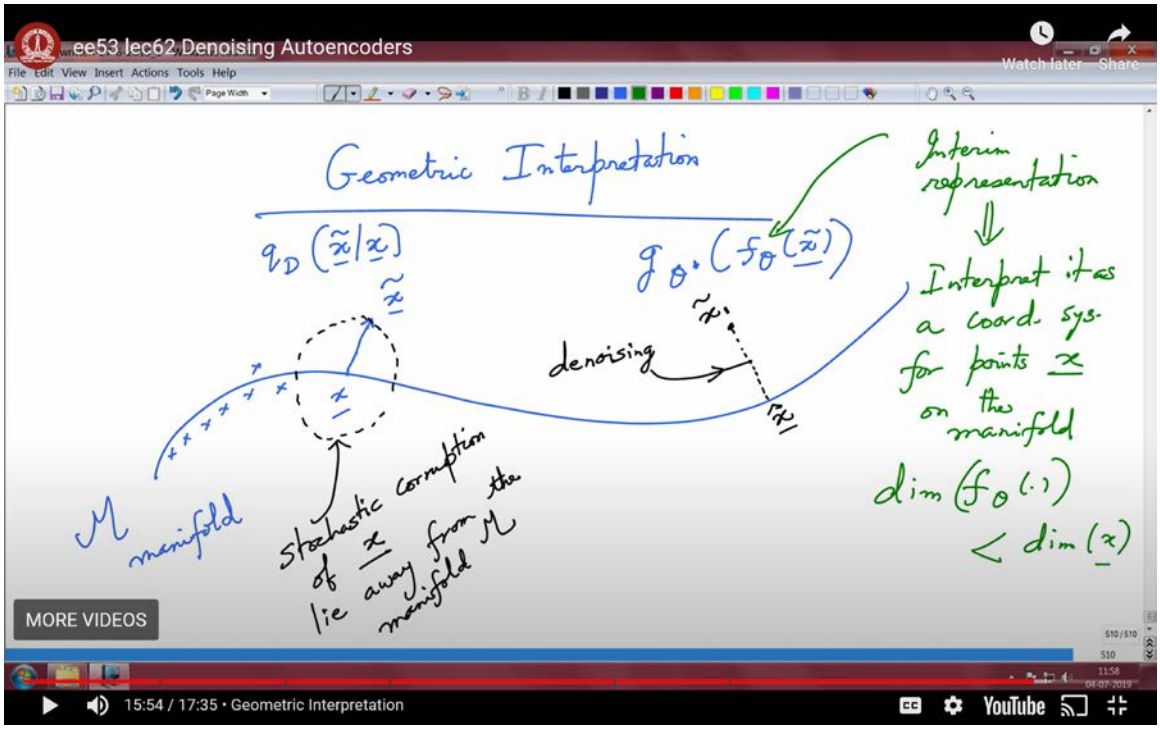
Additionally, there is a fascinating geometric interpretation of how denoising autoencoders operate. Imagine a manifold, and data points situated close to this manifold, reflecting how

they have been learned. Some data points might be corrupted due to noise, which is represented by a mapping denoted as $Q_D(\tilde{x} \mid x)$. This mapping represents the stochastic corruption of the data vector x, causing it to deviate from the manifold $\mathcal{M}$.

Consider a noisy data point $\tilde{x}$, which lies within an ϵ-ball around the original data point due to noise. The goal of a denoising autoencoder is to project this noisy point back onto the manifold to approximate the original clean point x. This projection process is a key aspect of denoising autoencoders.

In this context, you can think of a function $g_{\theta'}$ that acts on the corrupted input, aiming to recover the clean input. Meanwhile, F_θ serves as an intermediate representation, providing a coordinate system for points that lie on the manifold.

(Refer Slide Time: 15:54)



The interim representation, or latent representation, can be viewed as a coordinate system for points on the manifold. Crucially, the dimension of this coordinate system, defined by F_θ, is less than the dimension of the original input x. This subtle detail emphasizes that F_θ provides a more compact representation of the data.

The mapping $g_{\theta'}$ is responsible for projecting the corrupted input back to a point that is closest to the manifold. This process is central to the reconstruction phase. The loss function measures the fidelity between the original input vector x and the reconstructed output $\hat{x}$, and the goal is to optimize the parameters of the neural network over these mappings, from encoding to decoding.

In practice, you optimize these parameters to minimize the loss between the original and the reconstructed inputs. This completes our qualitative discussion on denoising autoencoders. For a deeper understanding, including theoretical analysis and guarantees about the effectiveness of this approach, you should explore research papers and advanced texts on the subject. For the scope of this course, this overview provides a sufficient foundation.