IDML 730 E
Assignment 4: AutoEncoder — Denoising and Neural Inpainting

IIT2015113 - Faheem Hassan Zunjani
IIT2015505 - Souvik Sen
IIT2015511 - Sayantan Chatterjee

October 8, 2018

Abstract

In this experiment we studied the denoising functionality of an AutoEncoder and neural inpainting functionality of a Variational AutoEncoder(VAE) for Image Completion on the MNIST dataset.

1 Introduction

An autoencoder is a type of artificial neural network used to learn efficient data codings in an unsupervised manner. The aim of an autoencoder is to learn a representation (encoding) for a set of data, typically for dimensionality reduction. Recently, the autoencoder concept has become more widely used for learning generative models of data. Some of the most powerful AI in the 2010s have involved sparse autoencoders stacked inside of deep neural networks.

2 Architecture

An autoencoder learns to compress data from the input layer into a short code, and then uncompress that code into something that closely matches the original data. This forces the autoencoder to engage in dimensionality reduction, for example by learning how to ignore noise. Some architectures use stacked sparse autoencoder layers for image recognition. The first autoencoder might learn to encode easy features like corners, the second to analyze the first layer's output and then encode less local features, the third might encode a segment of the image representing a whole feature, until the final autoencoder encodes the whole image into a code that matches the concept of the label.

The detailed archiecture of an AutoEncoder is illustrated in Figure 1.

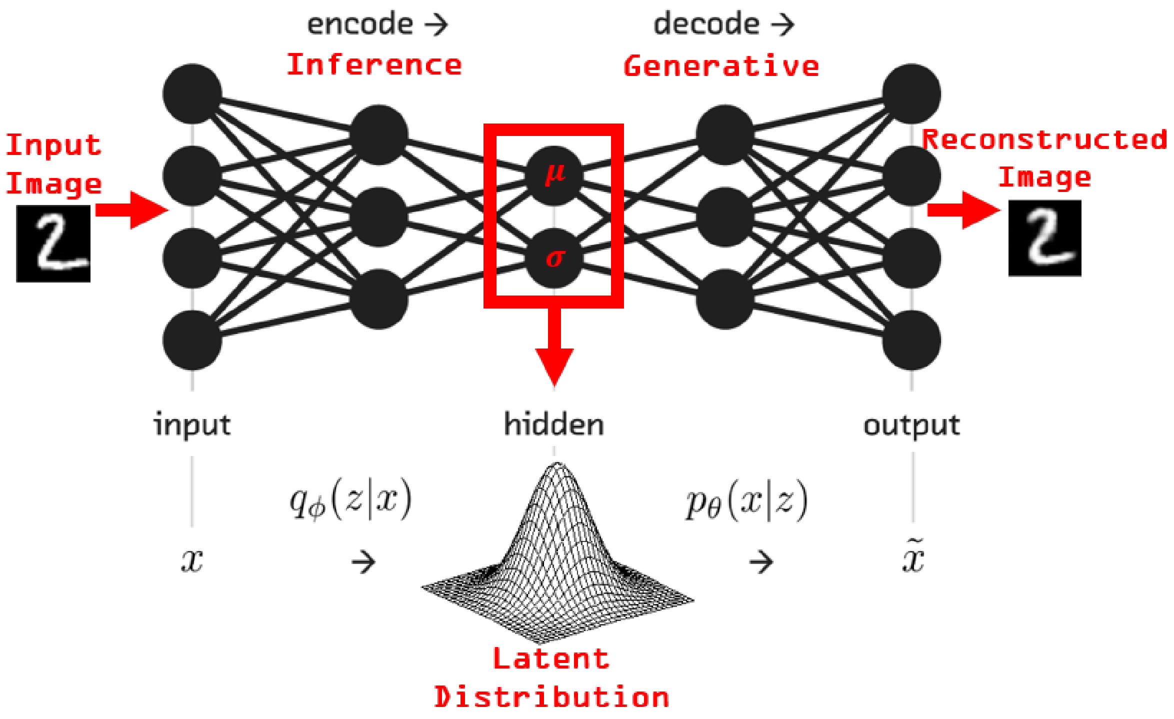


Figure 1: Architecture of an AutoEncoder

Architecturally, the simplest form of an autoencoder is a feedforward, non-recurrent neural network very similar to the many single layer perceptrons which makes a multilayer perceptron (MLP) having an input layer, an output layer and one or more hidden layers connecting them but with the output layer having the same number of nodes as the input layer, and with the purpose of reconstructing its own inputs (instead of predicting the target value Y given inputs X). Therefore, autoencoders are unsupervised learning models.

An autoencoder always consists of two parts, the encoder and the decoder, which can be defined as transitions ϕ and ψ.

2.1 Denoising AutoEncoder

Denoising autoencoders take a partially corrupted input whilst training to recover the original undistorted input. This technique has been introduced with a specific approach to good representation. A good representation is one that can be obtained robustly from a corrupted input and that will be useful for recovering the corresponding clean input. This definition contains the following implicit assumptions:

- The higher level representations are relatively stable and robust to the corruption of the input;
- It is necessary to extract features that are useful for representation of the input distribution.

To train an autoencoder to denoise data, it is necessary to perform preliminary stochastic mapping $x \to \tilde{x}$ in order to corrupt the data and use $\tilde{x}$ as input for a normal autoencoder, with the only exception being that the loss should be still computed for the initial input $\mathcal{L}(x, \tilde{x}')$ instead of $\mathcal{L}(\tilde{x}, \tilde{x}')$

2.2 Variational AutoEncoder

Variational autoencoder models inherit autoencoder architecture, but make strong assumptions concerning the distribution of latent variables. They use variational approach for latent representation learning, which results in an additional loss component and specific training algorithm called Stochastic Gradient Variational Bayes (SGVB). It assumes that the data is generated by a directed graphical model $p(x|z)$ and that the encoder is learning an approximation $q_\phi(z|x)$ to the posterior distribution $p_\theta(z|x)$ where ϕ and θ denote the parameters of the encoder (recognition model) and decoder (generative model) respectively. The objective of the variational autoencoder in this case has the following form:

$$\mathcal{L}(\phi, \theta, \mathbf{x}) = D_{\mathrm{KL}}(q_\phi(\mathbf{z}|\mathbf{x}) \| p_\theta(\mathbf{z})) - E_{q_\phi(\mathbf{z}|\mathbf{x})}\big(\log p_\theta(\mathbf{x}|\mathbf{z})\big)$$

3 Experiment and Results

Our experiment in this assignment is twofold — denoising images of MNIST dataset after studying its representation, implementing an autoencoder for neural inpainting.

3.1 Implementing the AutoEncoder

A VAE has three basic parts:

- An encoder that that learns the parameters (mean and variance) of the underlying latent distribution
- A means of sampling from that distribution
- A decoder that can turn the sample back into an image.

3.1.1 Sampling Function

We create a function to sample from the distribution we just learned the parameters of. epsilon is a tensor of small random normal values. One of the assumptions underlying a VAE like this is that our data arose from a random process and is normally distributed in the latent space.

3.1.2 Loss

The VAE is trained using a loss function with two components:

- **'Reconstruction loss'** - This is the cross-entropy describing the errors between the decoded samples from the latent distribution and the original inputs.
- The **Kullback-Liebler divergence** between the latent distribution and the prior (this acts as a sort of regularization term).

3.2 Studying the representation of MNIST handwritten digits

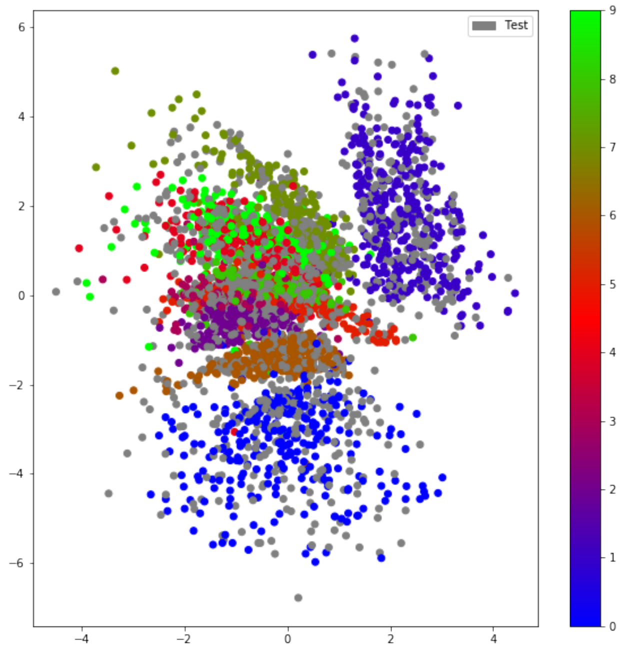


Figure 2: Representation of the MNIST handwritten digits and overlapping the test results on the digit clusters

3.3 Implementing a Denoising AutoEncoder

We use the aforementioned autoencoder model to be trained with the original MNIST images and their noisy counterparts subjected to a Gaussian noise as shown in figures 3 and 4:

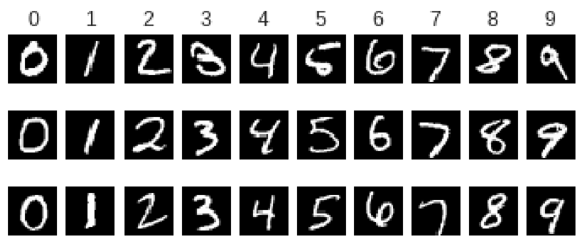


Figure 3: Original Images

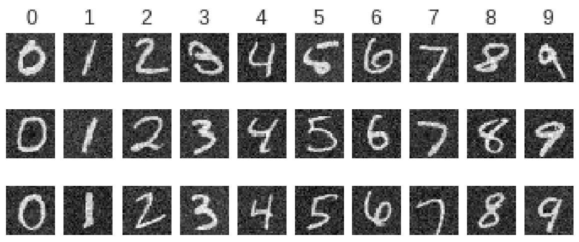


Figure 4: Noisy Images (Gaussian, $\sigma = 0.1$)

The autoencoder is then tested to output the reconstructed images as illustrated in figure 5:

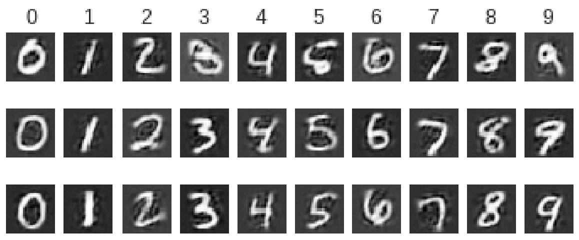


Figure 5: Reconstructed Images

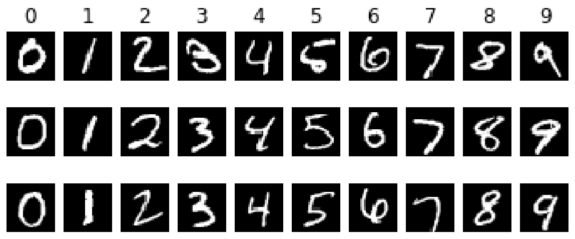
3.4 Implementing Neural Inpainting with AutoEncoder

There are 4 types of VAE we implemented for this experiment:

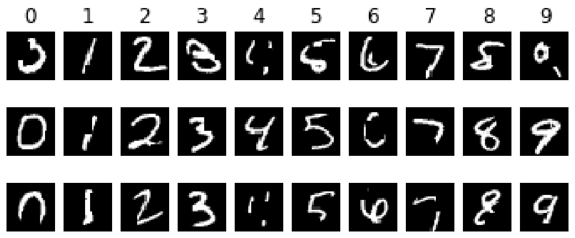
- Vanilla Variational AutoEncoder
- VAE with complete images as inputs
- Conditional VAE with labels as condition
- Conditional VAE with incomplete images as condition

The loss functions used in this part are **Binary Cross Entropy(BCE)** and **Mean Squared Error(MSE)**.

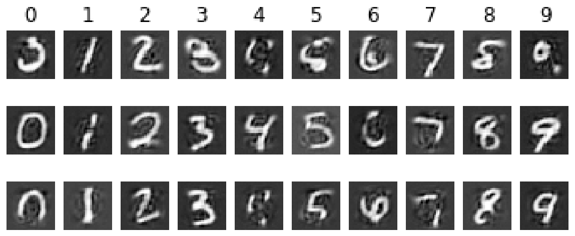
The original, corrupted and reconstructed images are showing in figure 6.



(a) Original Images



(b) Corrupted Images



(c) Reconstructed Images

Figure 6: Neural Inpainting

As illustrated in figure 7, the training losses of each model keep decreasing. In particular, we noticed the two CVAE models with MSE loss rendered both the lowest loss and the best reconstructed images.

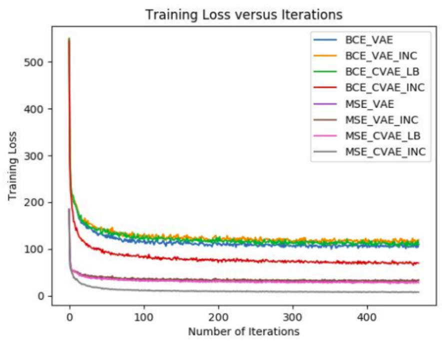


Figure 7: Training Loss vs Number of Iterations