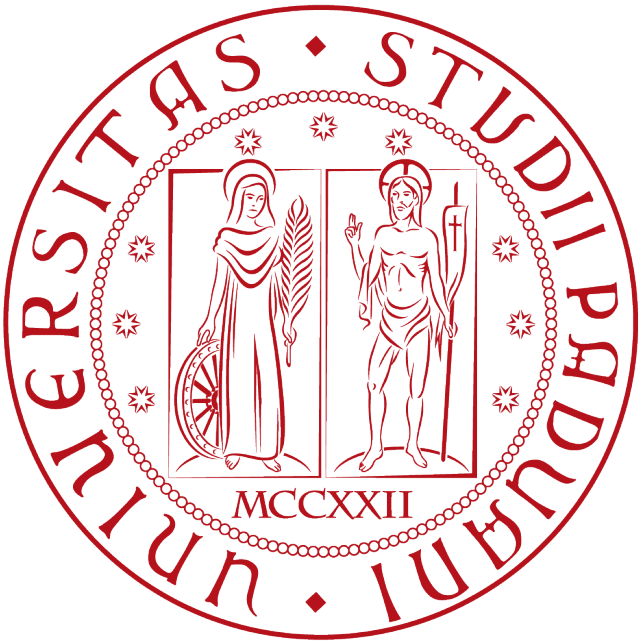
University of Padua

Deep Learning and Neural Networks

Unsupervised Deep Learning

Second Assignment Report

Author:
Ufuk Baran Karakaya (1215960)
ufukbaran.karakaya@studenti.unipd.it



Introduction

Unsupervised Learning is a machine learning technique in which the users do not need to supervise the model. Instead, it allows the model to work on its own to discover patterns and information that was previously undetected. It mainly deals with the unlabelled data. Unsupervised Learning Algorithms allow users to perform more complex processing tasks compared to supervised learning. Although, unsupervised learning can be more unpredictable compared with other natural learning methods. Unsupervised learning algorithms include clustering, anomaly detection, neural networks, etc.

In addition, unsupervised learning provides us some advantages including;

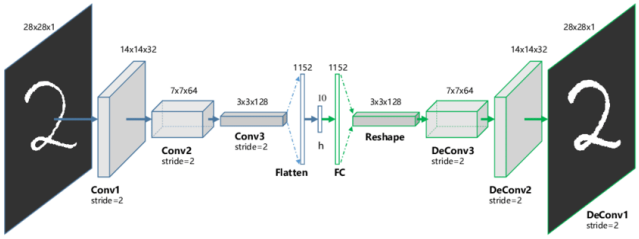
- Unsupervised machine learning finds all kind of unknown patterns in data.
- Unsupervised methods help you to find features which can be useful for categorization.
- It is taken place in real time, so all the input data to be analyzed and labeled in the presence of learners.
- It is easier to get unlabeled data from a computer than labeled data, which needs manual intervention.

On the other hand, it may lead some drawbacks including;

- You cannot get precise information regarding data sorting, and the output as data used in unsupervised learning is labeled and not known
- Less accuracy of the results is because the input data is not known and not labeled by people in advance. This means that the machine requires to do this itself.
- The spectral classes do not always correspond to informational classes.
- The user needs to spend time interpreting and label the classes which follow that classification.
- Spectral properties of classes can also change over time so you can't have the same class information while moving from one image to another.

Approach & Algorithm

In this project, the program implements Convolutional Autoencoder. To simplify the definition of method; an autoencoder is a type of artificial neural network used to learn efficient codings of unlabeled data (unsupervised learning). The encoding is validated and refined by attempting to regenerate the input from the encoding. The autoencoder learns a representation (encoding) for a set of data, typically for dimensionality reduction, by training the network to ignore insignificant data ("noise").



Convolutional Autoencoder is a variant of Convolutional Neural Networks that are used as the tools for unsupervised learning of convolution filters. They are generally applied in the task of image reconstruction to minimize reconstruction errors by learning the optimal filters. Once they are trained in this task, they can be applied to any input in order to extract features. Convolutional Autoencoders are general-purpose feature extractors differently from general autoencoders that completely ignore the 2D image structure. In autoencoders, the image must be unrolled into a single vector and the network must be built following the constraint on the number of inputs.

The algorithm is based on two main steps such as initialization of layers (encoder layers and decoder layers), and forwarding part. The algorithm follows the steps which are given the below.

- First of all, the code imports the related libraries
- After importing the libraries, it will download the MNIST dataset.
- Now, the program will prepare the data loaders that will be used for training and testing.
- The code will print some random images from the training data set.
- In the next step, it will define the Convolutional Autoencoder as a class that will be used to define the final Convolutional Autoencoder model.
- After that, the program will define the loss criterion and optimizer.
- Now, the code will pass our model to the CUDA environment. Make sure that you are using GPU.
- In the next step, it will train the model on MNIST dataset.
- Finally, it will train the convolutional autoencoder model on generating the reconstructed images.

In addition, the code implies k-fold cross validation to increase the robustness of results. During the epochs the loss is calculated and according to loss value, backward step is applied to compute the gradient. After these steps, optimizer updates the parameters.

Denoising Autoencoders

Autoencoders are Neural Networks which are commonly used for feature selection and extraction. However, 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 on purpose by randomly turning some of the input values to zero. In general, the percentage of input nodes which are being set to zero is about 50When calculating the Loss function, it is important to compare the output values with the original input, not with the corrupted input. That way, the risk of learning the identity function instead of extracting features is eliminated.

For denoising autoencoder, the code needs to add the following steps:

- Calling Dropout to randomly turning off neurons.
- Create noise mask
- Create bad images by multiply good images to the binary masks

Variational Autoencoders

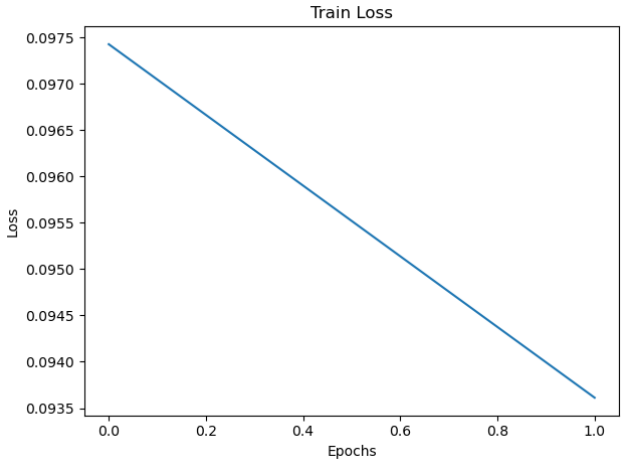
Variational autoencoders use probability modeling in a neural network system to provide the kinds of equilibrium that autoencoders are typically used to produce. The variational autoencoder works with an encoder, a decoder and a loss function. By reconstructing loss aspects, the system can learn to focus on desired likelihoods or outputs, for example, producing remarkable focus in image generation and image processing. For example, tests of these types of networks show their ability to reconstruct and render numerical digits from inputs.

For variational autoencoder, the code needs to add the following steps:

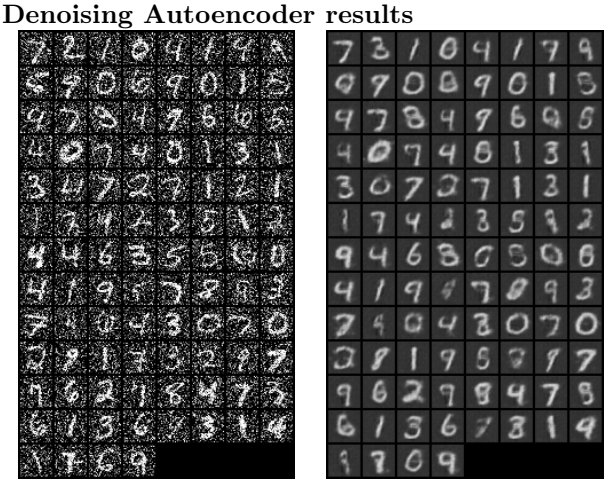
- The input is encoded as distribution over the latent space
- The point from the latent space is sampled from that distribution
- The sampled point is decoded and the reconstruction error can be computed
- The reconstruction error is back-propagated through the network

Results and Analysis

The loss of Convolutional Autoencoders is illustrated on the graph:



Denoising Autoencoder results



Noised Images Denoised Images

cc

In fine-tuning part, the code exploits the pre-trained model to obtain more robust results. This strategy can be considered to eliminate drawbacks of unsupervised algorithms which are caused by unlabelled and less trained data. With help of fine tuning, layers of pre-trained model are integrated the model again more effectively.