Semi-supervised learning via Deep Denoising Autoencoders

Autoencoders in TensorFlow

Angelo Porrello, Davide Abati

December 11, 2018

University of Modena and Reggio Emilia

Agenda

UNIMORE UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

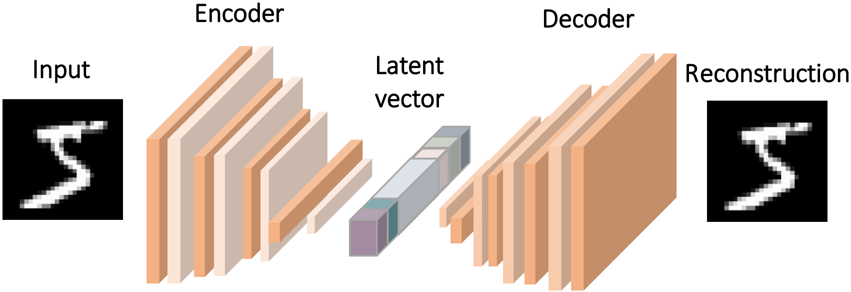
Autoencoders

Semi-supervised learning

Semi-supervised learning on MNIST

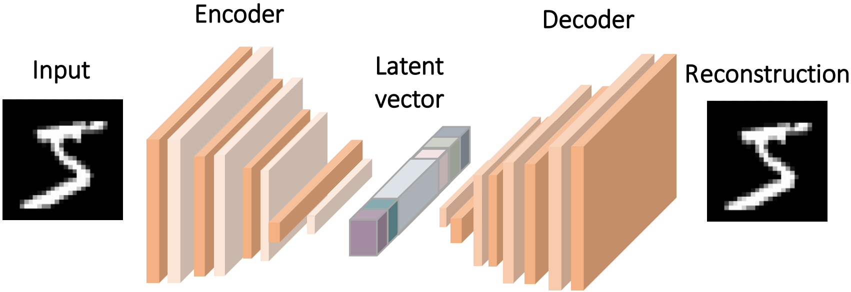
Autoencoders

An **autoencoder** is a feed-forward neural network that is trained to attempt to copy its input to its output. The network may be viewed as consisting of two parts: an encoder function $h = f(x)$ and a decoder that produces a reconstruction $r = g(h)$.

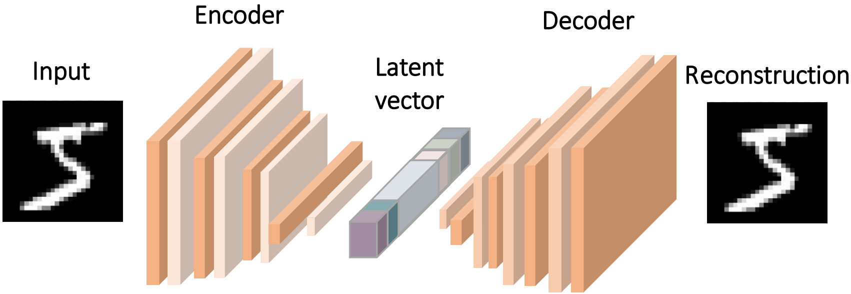


The learning process plans to minimize $\mathcal{L}(g(f(x))$ where $\mathcal{L}$ is a loss function penalizing $g(f(x))$ for being dissimilar from x, e.g. the Mean Square Error (MSE).

$$\mathcal{L}(g(f(x)) = \|x - g(f(x)\|_2 \tag{1}$$

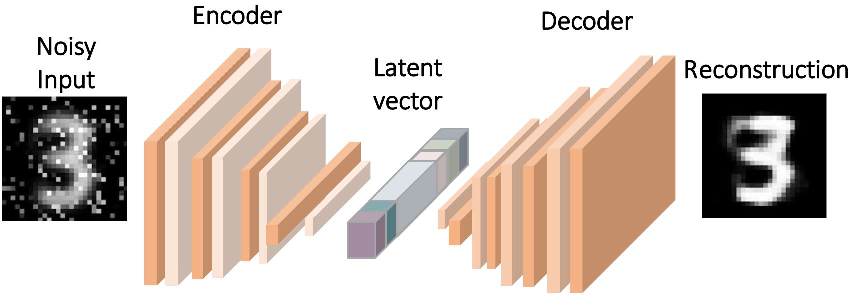


The **aim** is to induce in h useful properties and the most salient features of the training data. For instance, **lower dimensional representations** attempt to compress as much information about x in a smaller representation.



Denoising Autoencoders (DAE)

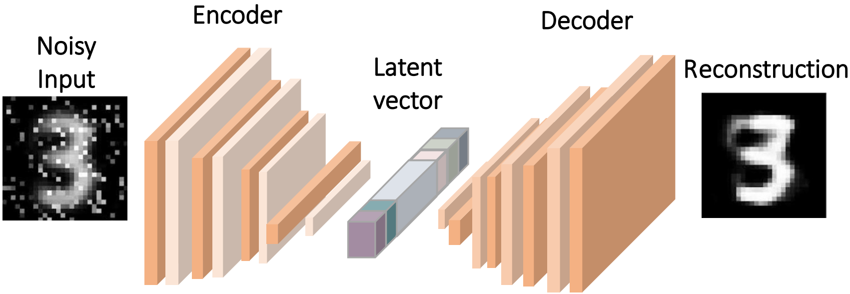
In order to avoid learning the identity function, autoencoders are restricted in ways that allow them to copy only **approximately**. To capture more robust features in the hidden layer, a denoising autoencoder is trained to reconstruct the input x from a **corrupted** version $\tilde{x}$ of it.



Denoising Autoencoders (DAE)

The corrupted input $\tilde{x}$ can be obtained applying on x some form of noise e.g additive white gaussian noise or dropout. Then, the DAE must undo this corruption rather than simply copying their input.

$$\mathcal{L}(g(f(x)) = \|x - g(f(\tilde{x})\|_2 \quad \tilde{x} \sim \mathcal{N}(x, \sigma^2 \mathrm{I}) \tag{2}$$



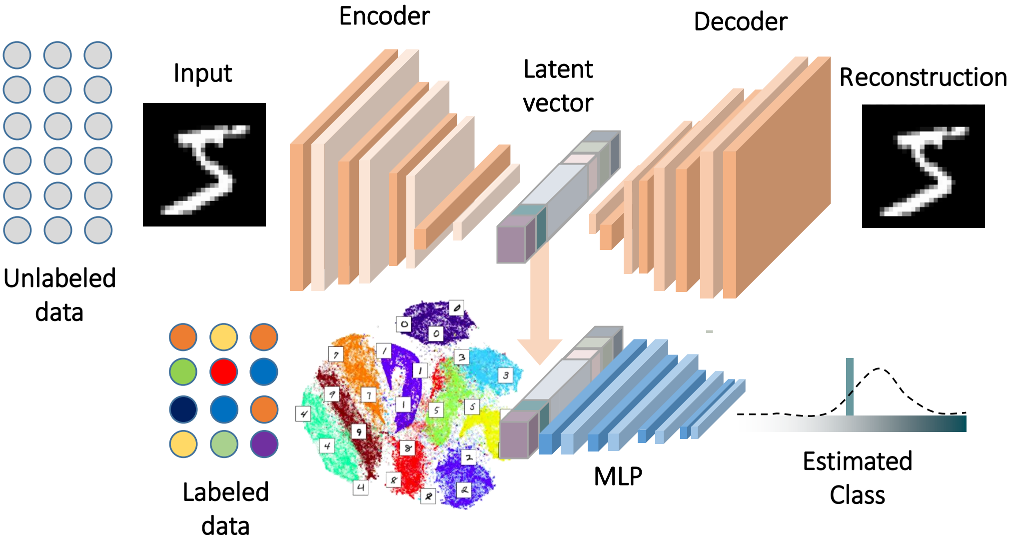
Semi-supervised learning

Semi-supervised learning

Problem: Deeper models lead to more parameters, which implicates the requirement for a high number of training data in order to avoid overfitting.

Semi-supervised learning regards a class of machine learning techniques combining both labeled and unlabeled data.

Goal: taking advantage of a large amount of **unlabeled data** to learn a suitable representation, and then exploit it for training a new classifier, the latter leveraging just few labeled data.



Encoder
Decoder
Input
Latent vector
Reconstruction
Unlabeled data
Labeled data
MLP
Estimated Class

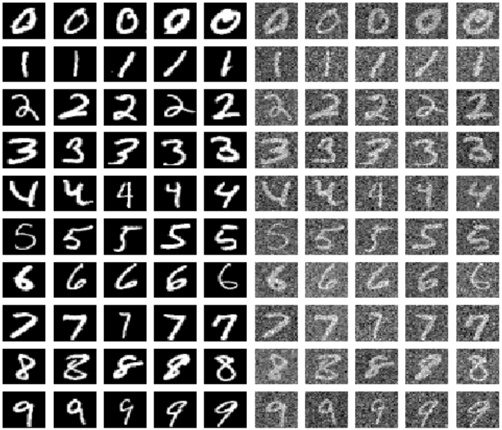
Semi-supervised learning on MNIST

Semi-supervised learning on MNIST

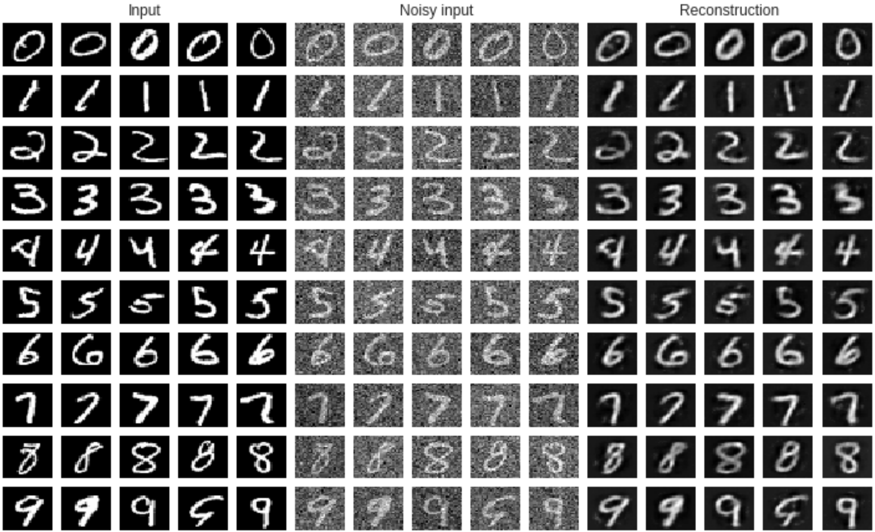
1. Given the MNIST training set, discard the labels and train a denoising autoencoder on it. As a starting point, provide a DAE with just dense layers, batch normalization and the non-linearity you prefer.

Once it has been trained, extract bottleneck activations for both training and test set.

Class-wise Clean vs Noisy MNIST images



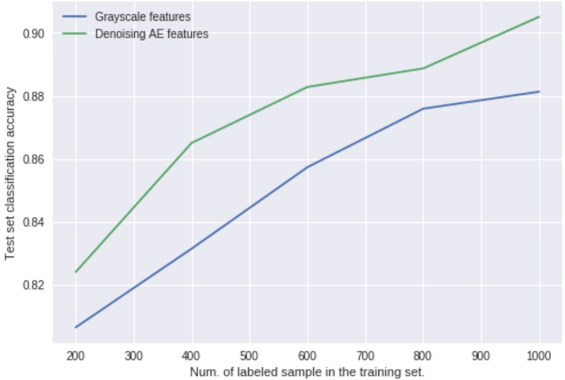
Semi-supervised learning on MNIST



2. To emulate a setting with few labeled samples, pick a small subset of the original and fully-labeled training set (e.g. comprising just 200 randomly drawn samples among the 60000 available).

3. Train a simple classifier (e.g. K-NN) from both the grayscale features and the hidden activations.

4. Compare the test set classification accuracies arising from the two strategies.



Useful Functions

To this purpose, you may find useful the following functions:

- `tf.cond`
- `tf.random_normal`
- `tf.layers.batch_normalization`
- `tf.layers.dense`

Please refer to the docs to know the exact API.

Optionally

- Replace the dense layers with 2D-convolutions.
- Compare the results w.r.t. a **naive** autoencoder (where the input has not been corrupted).
- Compare the results w.r.t. a **sparse** autoencoder.
- Conduct experiments on a more challenging dataset (e.g. CIFAR-10).

Good Luck!