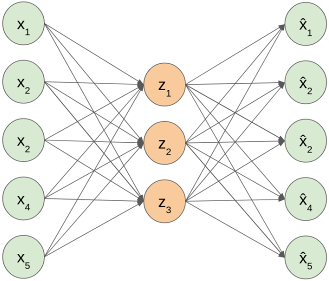
Deep Learning

Autoencoders - Basic Principle



Learning goals

- Task and structure of an AE
- Undercomplete AEs
- Relation of AEs and PCA

AUTOENCODER-TASK AND STRUCTURE

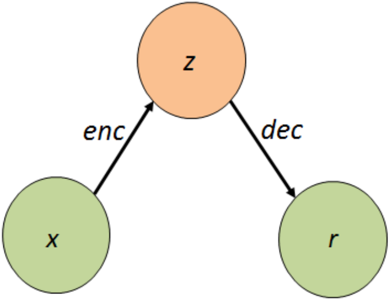
- Autoencoders (AEs) are NNs for unsupervised learning of a lower dimensional feature representation from unlabeled training data.
- Task: Learn a compression of the data.
- Autoencoders consist of two parts:
  - **encoder** learns mapping from the data $\mathbf{x}$ to a low-dimensional latent variable $\mathbf{z} = enc(\mathbf{x})$.
  - **decoder** learns mapping back from latent $\mathbf{z}$ to a reconstruction $\hat{\mathbf{x}} = dec(\mathbf{z})$ of $\mathbf{x}$.
- Loss function does not use any labels and measures the quality of the reconstruction compared to the input:

$$L\left(\mathbf{x}, dec(enc(\mathbf{x}))\right)$$

- Goal: Learn good **representation z** (also called **code**).

AUTOENCODER (AE)- COMPUTATIONAL GRAPH

The general structure of an AE as a computational graph:

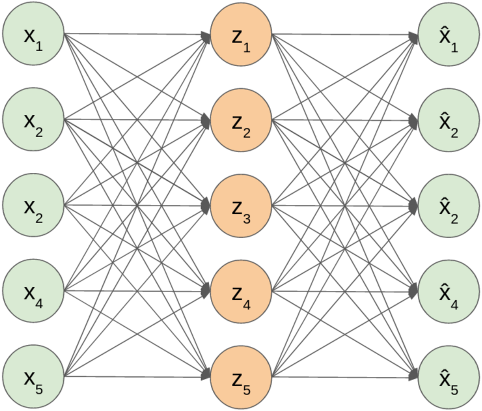


- An AE has two computational steps:
  - the encoder *enc*, mapping $\mathbf{x}$ to $\mathbf{z}$.
  - the decoder *dec*, mapping $\mathbf{z}$ to $\hat{\mathbf{x}}$.

Undercomplete Autoencoders

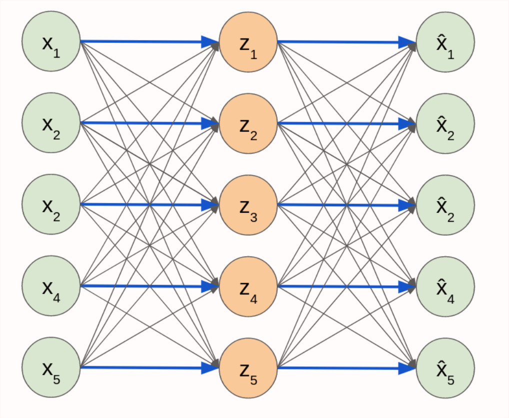
UNDERCOMPLETE AUTOENCODERS

- A naive implementation of an autoencoder would simply learn the identity $dec(enc(\mathbf{x})) = \mathbf{x}$.
- This would not be useful.



UNDERCOMPLETE AUTOENCODERS

- A naive implementation of an autoencoder would simply learn the identity $dec(enc(\mathbf{x})) = \mathbf{x}$.
- This would not be useful.

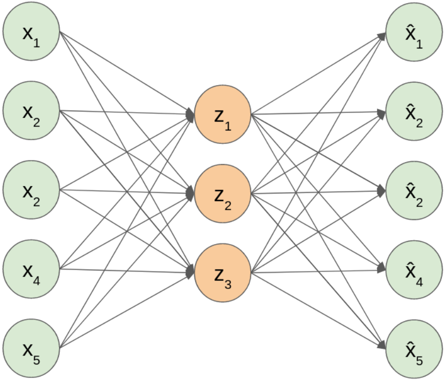


UNDERCOMPLETE AUTOENCODERS

- Therefore we have a “bottleneck” layer: We restrict the architecture, such that

$$\dim(\mathbf{z}) < \dim(\mathbf{x})$$

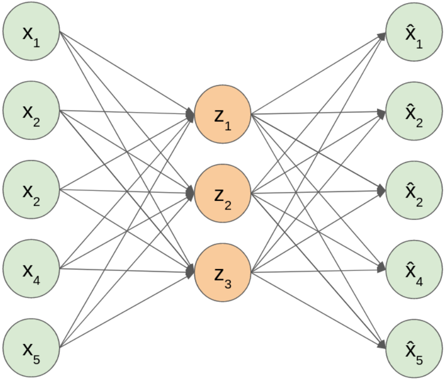
- Such an AE is called **undercomplete**.



UNDERCOMPLETE AUTOENCODERS

- In an undercomplete AE, the hidden layer has fewer neurons than the input layer.

→ That will force the AE to
  - capture only the most salient features of the training data!
  - learn a “compressed” representation of the input.



UNDERCOMPLETE AUTOENCODERS

- Training an AE is done by minimizing the risk with a loss function penalizing the reconstruction $dec(enc(\mathbf{x}))$ for differing from $\mathbf{x}$.
- The L2-loss

$$\|\mathbf{x} - dec(enc(\mathbf{x}))\|_2^2$$

  is a typical choice, but other loss functions are possible.
- For optimization, the same optimization techniques as for standard feed-forward nets are applied (SGD, RMSProp, ADAM,...).

EXPERIMENT: LEARN TO ENCODE MNIST

- Let us try to compress the MNIST data as good as possible.
- We train undercomplete AEs with different dimensions of the internal representation **z** (.i.e. different “bottleneck” sizes).

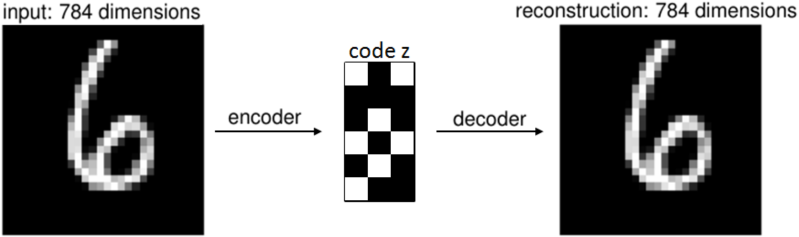


Figure: Flow chart of our our autoencoder: reconstruct the input with fixed dimensions $dim(\mathbf{z}) \leq dim(\mathbf{x})$.

EXPERIMENT: LEARN TO ENCODE MNIST

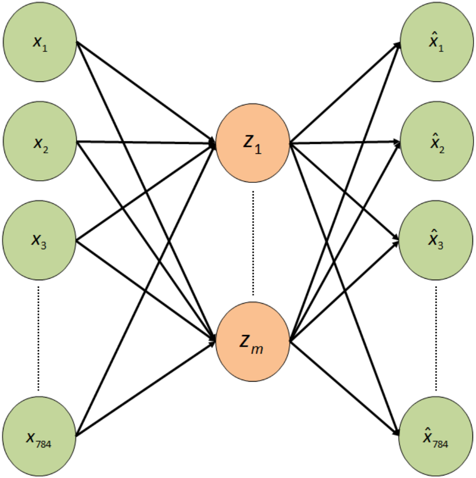


Figure: Architecture of the autoencoder.

EXPERIMENT: LEARN TO ENCODE MNIST

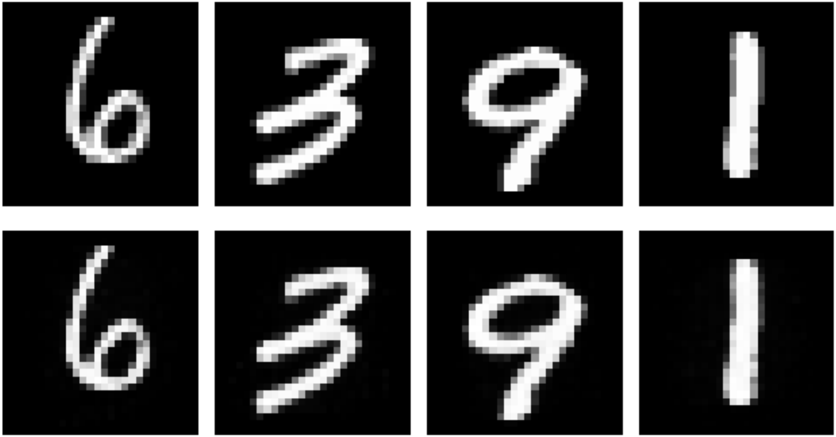


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 784 = dim(\mathbf{x})$.

EXPERIMENT: LEARN TO ENCODE MNIST

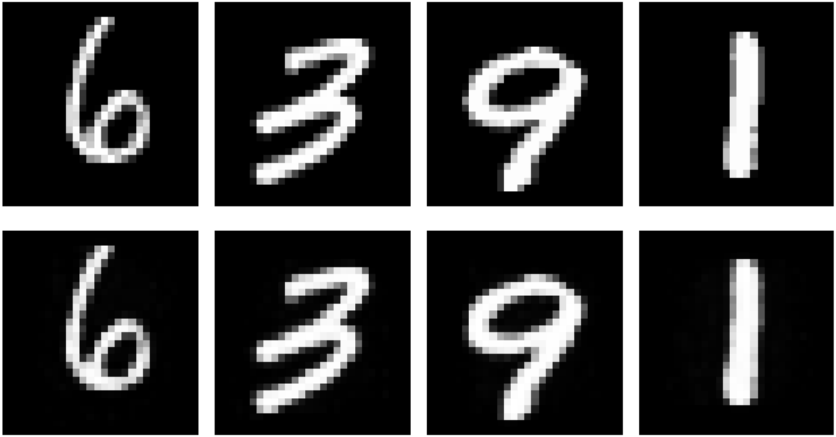


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 256$.

EXPERIMENT: LEARN TO ENCODE MNIST

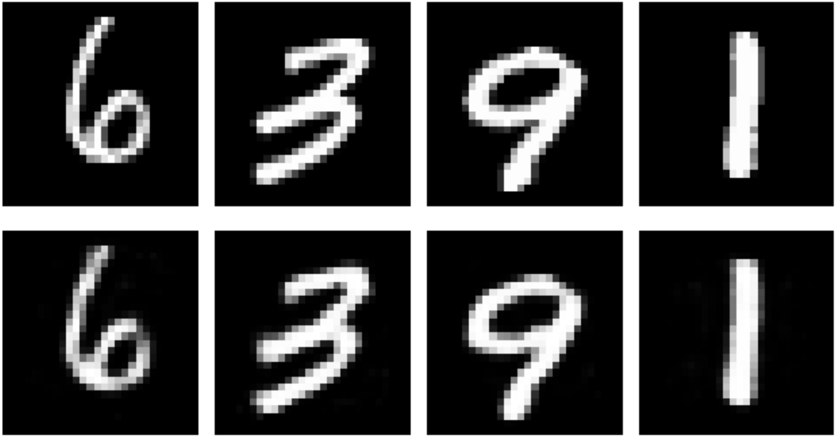


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 64$.

EXPERIMENT: LEARN TO ENCODE MNIST

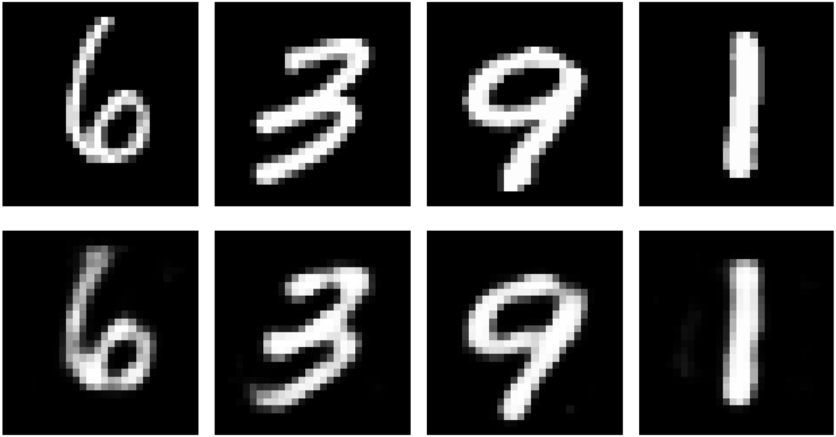


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 32$.

EXPERIMENT: LEARN TO ENCODE MNIST

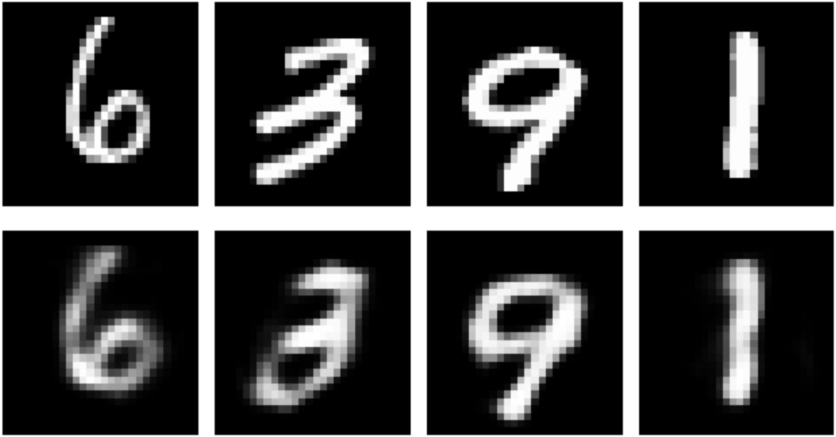


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 16$.

EXPERIMENT: LEARN TO ENCODE MNIST

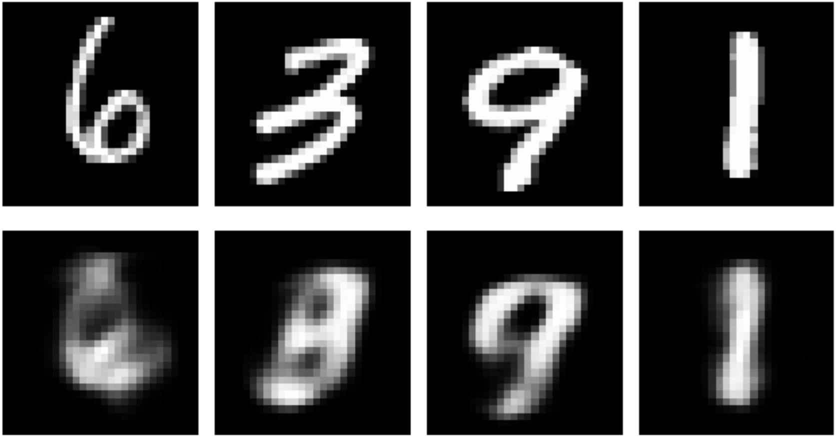


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 8$.

EXPERIMENT: LEARN TO ENCODE MNIST

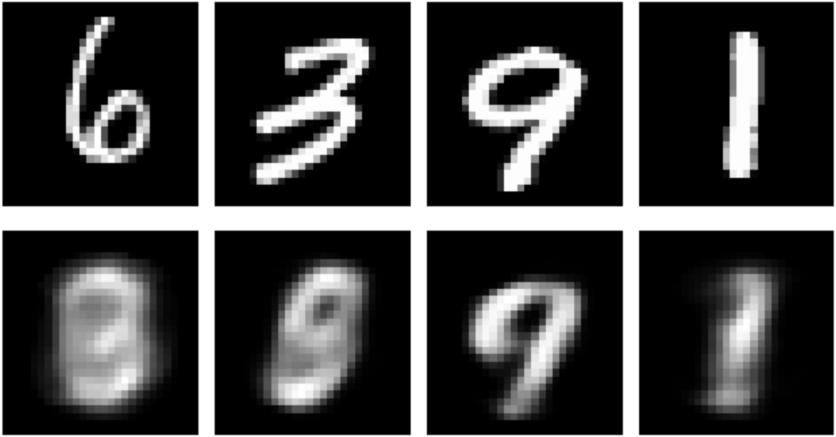


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 4$.

EXPERIMENT: LEARN TO ENCODE MNIST

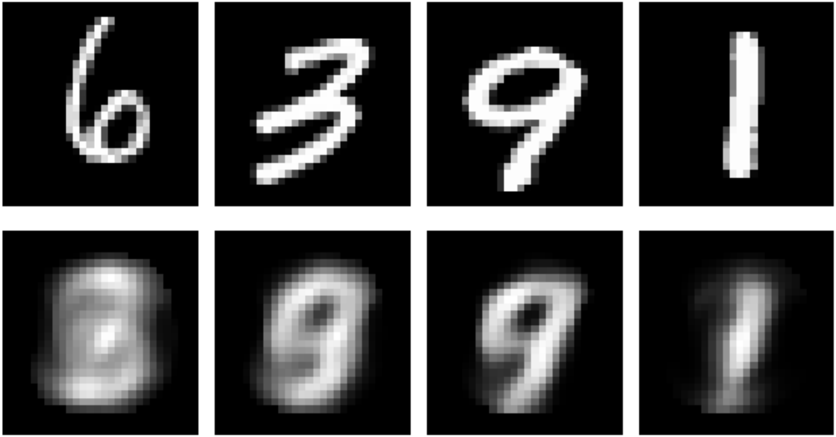


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 2$.

EXPERIMENT: LEARN TO ENCODE MNIST

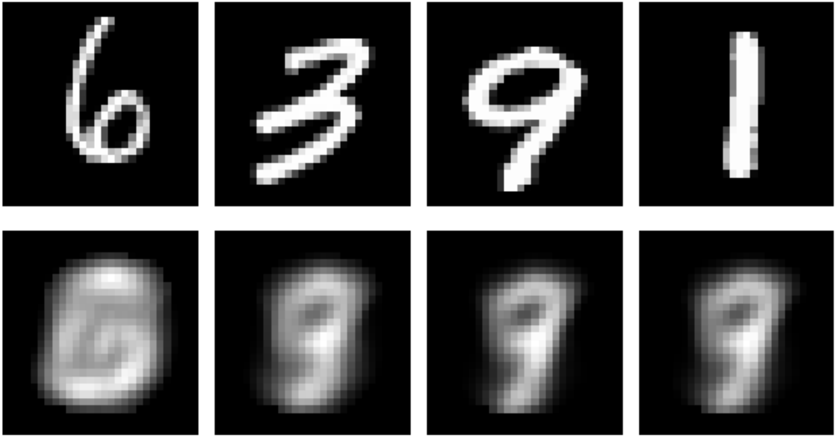
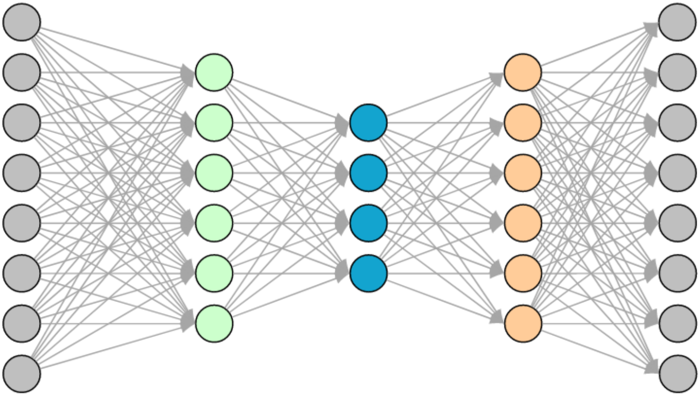


Figure: The top row shows the original digits, the bottom row the reconstructed ones.

- $dim(\mathbf{z}) = 1$.

INCREASING THE CAPACTIY OF AES

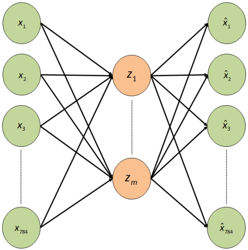
Increasing the number of layers adds capacity to autoencoders:



Autoencoders as Principal Component Analysis

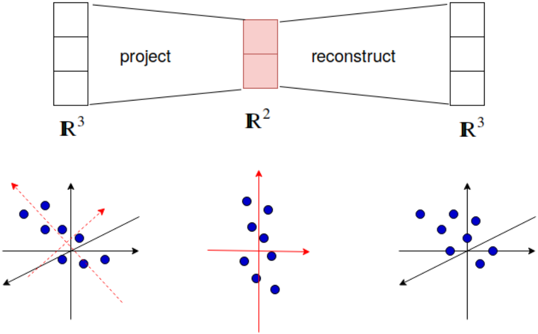
AES AS PRINCIPAL COMPONENT ANALYSIS

- Consider a undercomplete autoencoder with
  - **linear** encoder function $enc(\mathbf{x})$, and
  - **linear** decoder function $dec(\boldsymbol{z})$.
- The L2-loss $\|\mathbf{x} - dec(enc(\mathbf{x}))\|_2^2$ is employed and inputs are normalized to zero mean.
- We want to find the **linear projection** of the data with the minimal L2-reconstruction error.



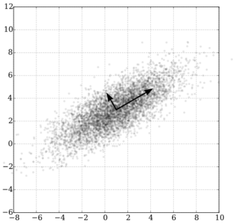
AES AS PRINCIPAL COMPONENT ANALYSIS

- It can be shown that the optimal solution is an **orthogonal** linear transformation (i.e. a rotation of the coordinate system) given by the $\dim(\boldsymbol{z}) = k$ singular vectors with largest singular values.



AES AS PRINCIPAL COMPONENT ANALYSIS

- This is an equivalent formulation to **Principal Component Analysis (PCA)**, which uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called **principal components**.
- The transformation is defined in such a way that the first principal component has the largest possible variance (i.e., accounts for as much of the variability in the data as possible).



AES AS PRINCIPAL COMPONENT ANALYSIS

- The formulations are equivalent: “Find a linear projection into a *k*-dimensional space that ...”
  - “... minimizes the L2-reconstruction error” (AE-based formulation).
  - “... maximizes the variance of the projected datapoints” (statistical formulation).
- An AE with a non-linear decoder/encoder can be seen as a non-linear generalization of PCA.

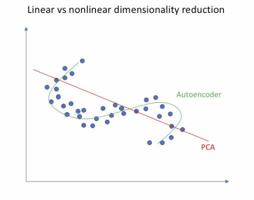


Figure: Credits: Jeremy Jordan “Introduction to autoencoders”

REFERENCES

Ian Goodfellow, Yoshua Bengio and Aaron Courville (2016)
Deep Learning
http://www.deeplearningbook.org/