Autoencoders and latent spaces

Raoul Grouls, 8 januari 2024

Overview

Motivation for autoencoders

An autoencoder learns a “latent space” with efficient encodings of <u>unlabeled data</u>. Some applications:

- Dimensionality reduction
- Anomaly detection
- Denoising
- Data Compression

Recap

Data	$X = \{\overrightarrow{x_1}, \ldots, \overrightarrow{x_n} \mid \vec{x} \in \mathbb{R}^d\}$ $y = \{y_1, \ldots, y_n \mid y \in \{0,1\}\}$
(Non)linearity	$f(X) = WX + b \quad \sigma(X) = max(0,X)$
Predict	$\hat{y} = f_n \circ \sigma \circ f_{n-1} \circ \ \ldots \ \circ \sigma \circ f_1$
Loss	$Loss(y, \hat{y})$
Optimize	$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$

Recap

$$X \rightarrow f_n \circ \sigma \circ \; \dots \rightarrow Loss(y, \hat{y})$$

$$\leftarrow \quad w \leftarrow w - \eta \frac{\partial Loss}{\partial W} \quad \longleftarrow$$

Contrast with supervised learning

Latent space

A latent space, also known as a feature space or hidden space, refers to a vectorspace $\mathbb{R}^d$ where the data's features are represented in a way that is not directly observable in the input space.

For autoencoders, the dimensionality is typically much lower than that of the input.

Contrast with supervised learning

Encoder - decoder

- Instead of: $X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_n} \to \{0,1\}$ the idea is to map the input X back to itself. Let's split the network conceptually into an encoder-decoder architecture:
  - An encoder $e = f_n \circ \sigma \circ f_{n-1} \circ \ \ldots \ \circ \sigma \circ f_1$ that maps

$$e : X \to \mathbb{R}^d$$

  - A decoder $d = f_m \circ \sigma \circ f_{m-1} \circ \ \ldots \ \circ \sigma \circ f_1$ that reconstructs input:

$$d : \mathbb{R}^d \to X$$

Contrast with supervised learning

Reducing dimensionality

An autoencoder is a network $AE(x) = d(e(x))$, which gives us:

$$AE\colon X \to \mathbb{R}^d \to X$$

The encoder maps input space X to latent space, that typically involves a reduction in dimensionality: $dim(Z) < dim(X)$

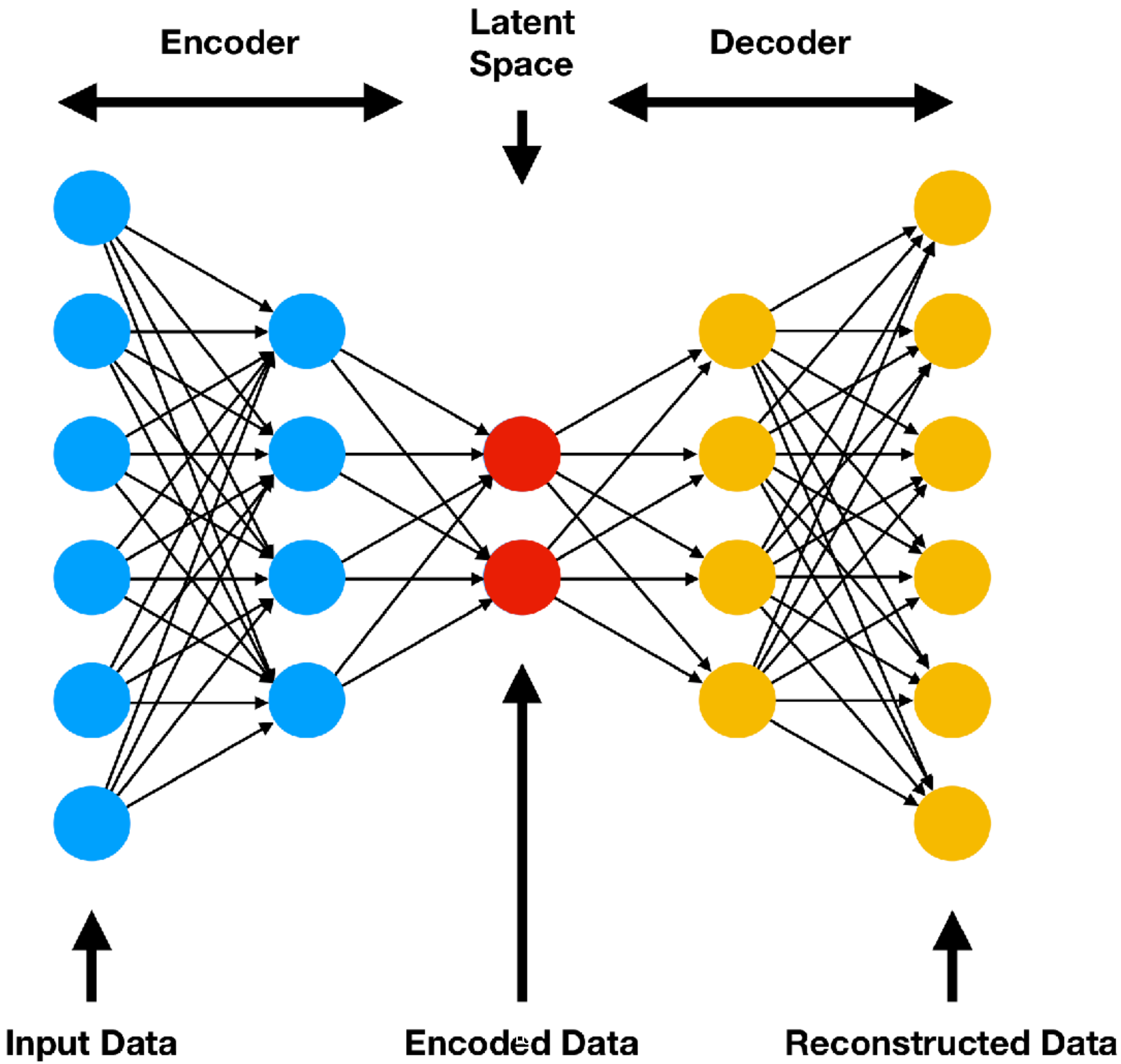
Contrast with supervised learning

Minimize reconstruction error

Instead of minimizing the error between $\hat{y}$ and y, the goal is to minimize the reconstruction error between $d(e(x))$ and x

Key differences with supervised learning

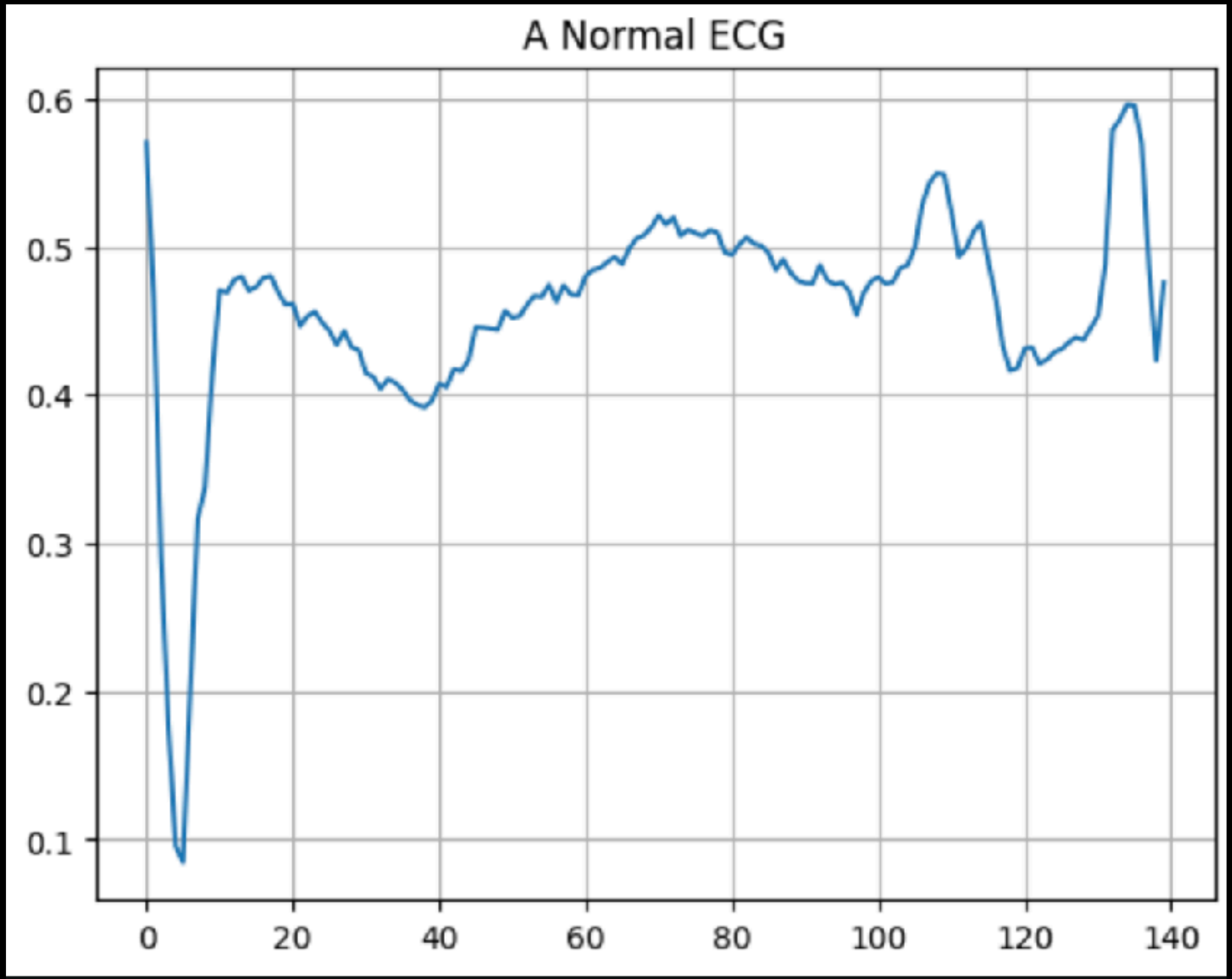
- We dont need external labels
- By restricting the dimensionality of Z, we force the model to learn to be as efficient as possible (make summaries). We dont focus on accuracy perse, but on efficiency (in terms of our endgoal)
- Generative AI explores the latent space as a source of creativity
- Sometimes we just want the encoder or decoder, instead of using the full model for inference.

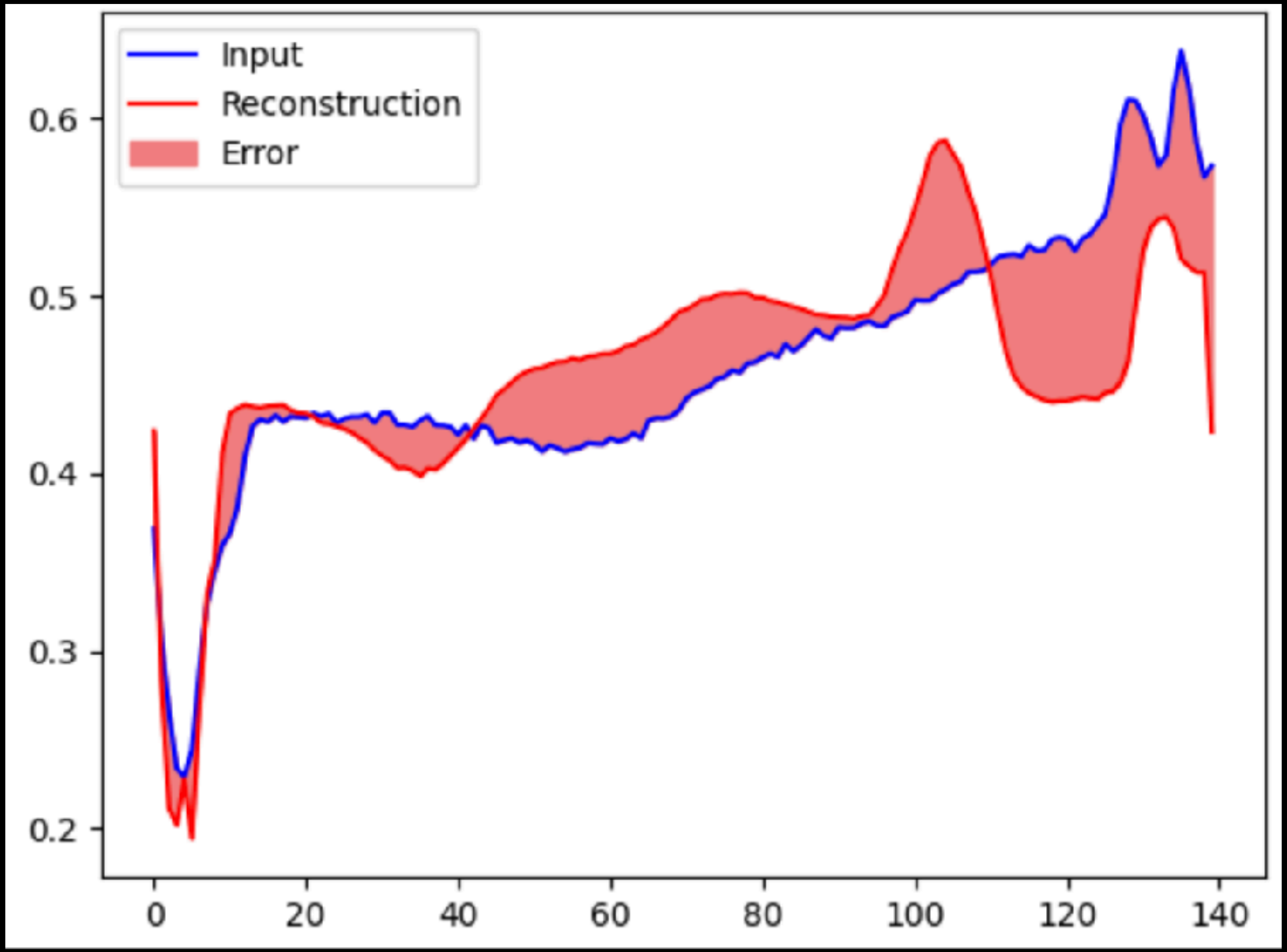

Encoder
Latent
Space
Decoder
Input Data
Encoded Data
Reconstructed Data

Overview, part 2

Extended motivation for autoencoders

- Dimensionality reduction (encoder) : Capture the most significant features, making it easier to visualise and process data.
- Data Compression (encoder): the latent space is compressed, so we can use that in itself.
- Anomaly detection (encoder-decoder): By learning the “normal” pattern of data, the reconstruction error will be bigger with anomalies even though the network hasn’t been trained with labels of anomalies.
- Denoising (encoder-decoder): the latent space is smaller, so has to be more efficient and will remove noise


A Normal ECG
0.6
0.5
0.4
0.3
0.2
0.1
0
20
40
60
80
100
120
140


Input
Reconstruction
Error
0.6
0.5
0.4
0.3
0.2
0
20
40
60
80
100
120
140

Supervised

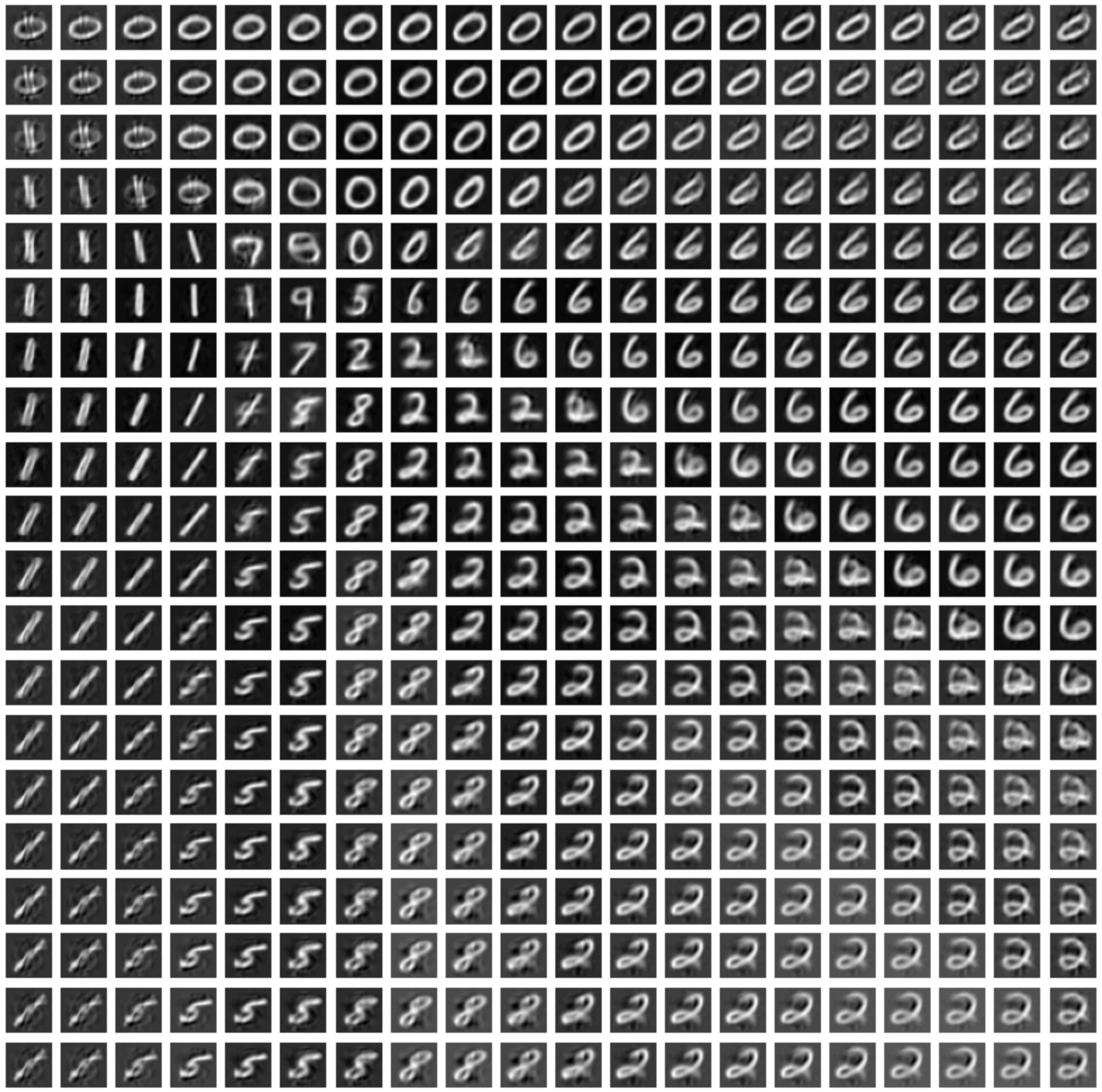
$$X \to f_1 \circ \sigma \circ \ \dots \to Loss(y, \hat{y})$$

$$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$$

Autoencoder

$$X \rightarrow f_n \circ \sigma \circ \; \ldots \rightarrow Z \rightarrow f_m \circ \sigma \circ \; \ldots \rightarrow \hat{X} \rightarrow Loss(X, \hat{X})$$

$$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$$



Unsupervised Classification

- Map your unlabeled training data to Z
- Map the new, unlabeled input to the latent space Z
- Find the k items in your trainingsdata that are closest in Z

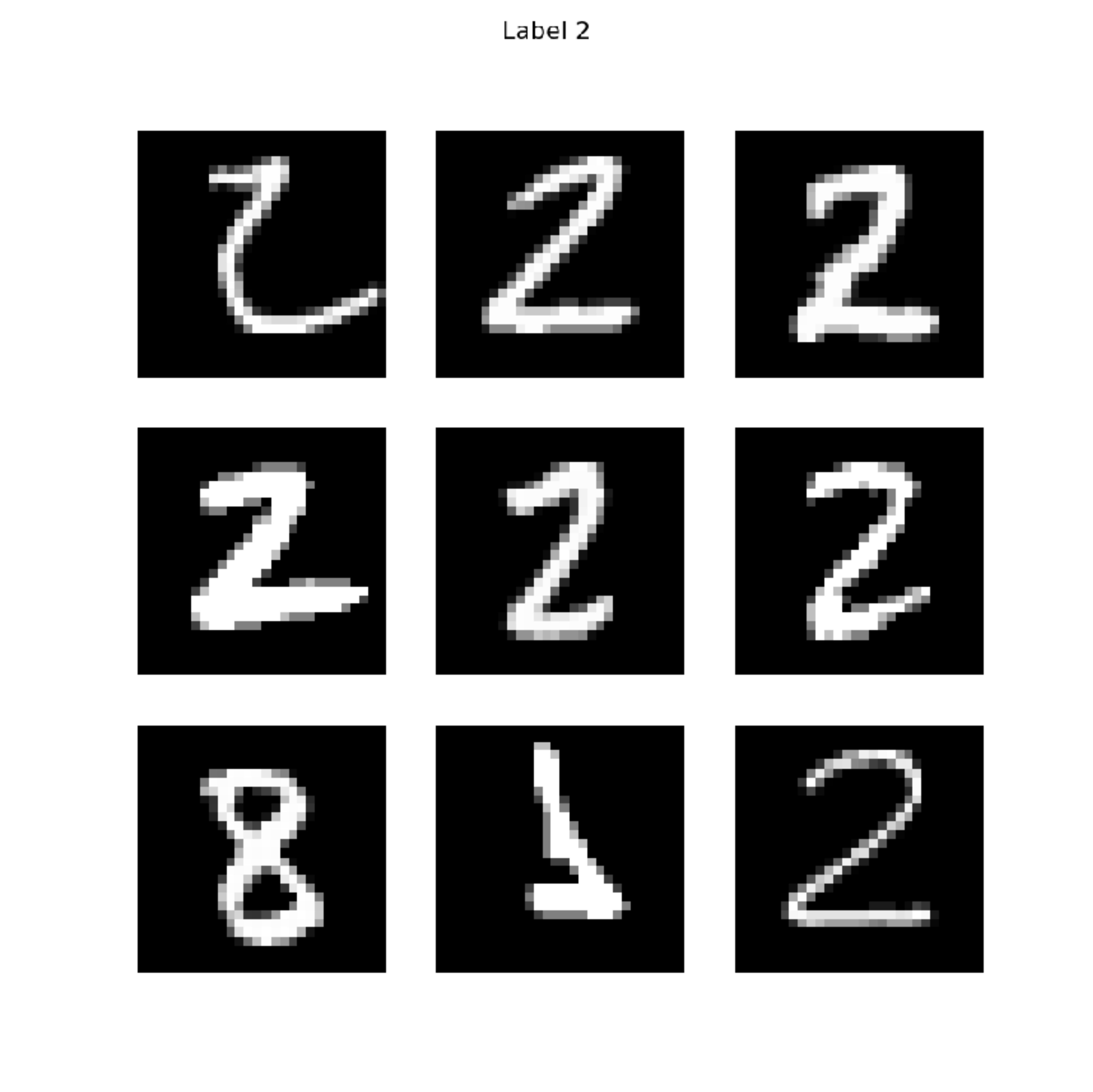


Fig: the 9 items closest to the new input

Siamese networks
Semisupervised

- $X = \{x_1, \ldots, x_j | x \in \mathbb{R}^D\}$
- A labeling function $g: X \times X \to \{0,1\}$ defined as $g(x_i, x_j) = \begin{cases} 1 & \text{if } x_i \sim x_j \\ 0 & \text{if } x_i \neq x_j \end{cases}$
- An encoder f: $x \to Z$ with $Z \subset \mathbb{R}^d$ and $d < D$
- A distance function $s(z_i, z_j)$, eg euclidian distance
- A loss function $Loss(s(z_i, z_j), y)$ that requires the distance to be close if the label is 1.

Siamese networks

Semisupervised