Autoencoders and latent spaces

Raoul Grouls, 6 January 2026

Motivation, part 1

Motivation for autoencoders

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

- Dimensionality reduction
- Anomaly detection
- Denoising
- Data Compression

Recap

Data

$$X = \{\overrightarrow{x_1}, \ldots, \overrightarrow{x_n} \,|\, \vec{x} \in \mathbb{R}^d\}$$

$$y = \{y_1, \ldots, y_n \,|\, y \in \{0,1\}\}$$

(Non)linearity

$$l(X) = WX + b \qquad \sigma(X) = max(0,X)$$

Predict

$$\hat{y} = l_n \circ \sigma \circ l_{n-1} \circ \ \ldots \ \circ \sigma \circ l_1$$

Loss

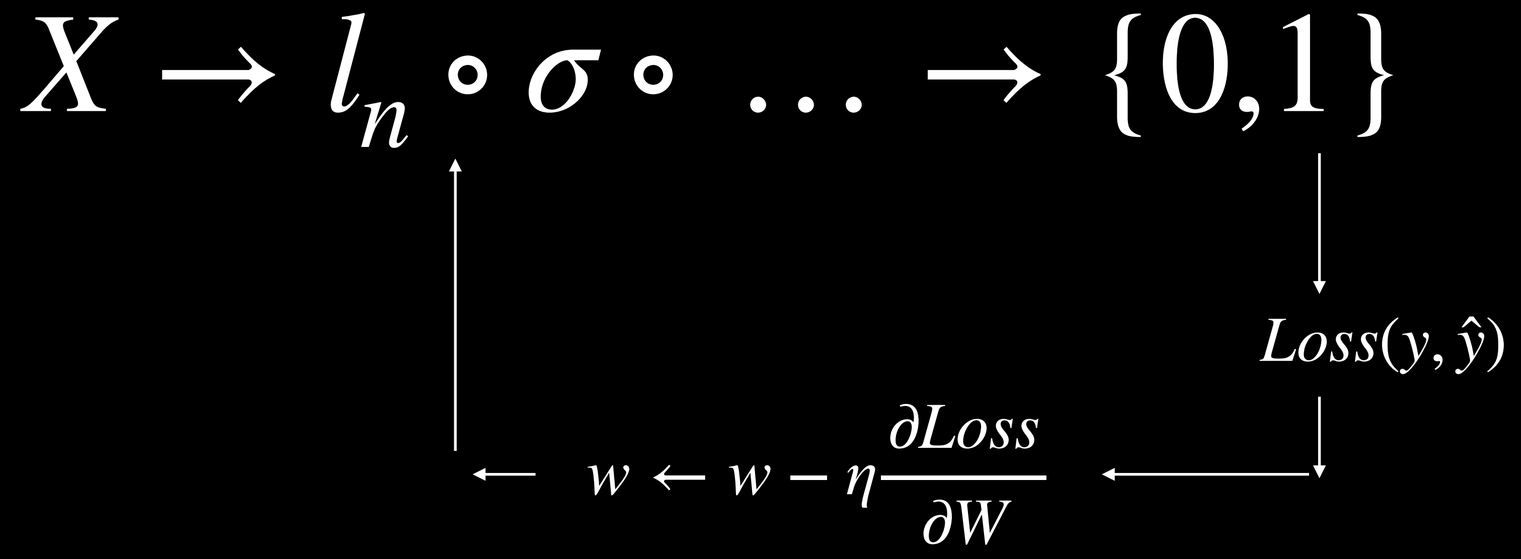
$$Loss(y, \hat{y})$$

Optimize

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

Ps: I am using the letter "l" instead of "f" to avoid notational confusion in the other slides

Recap


$X \rightarrow l_n \circ \sigma \circ \ \ldots \ \rightarrow \{0,1\}$
$Loss(y, \hat{y})$
$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$

Contrast with supervised learning

Latent space

We are going to learn what a latent space is by looking at the differences with the supervised learning we have been studying the last 5 lessons.

This strategy should remind you of a deep learning technique…

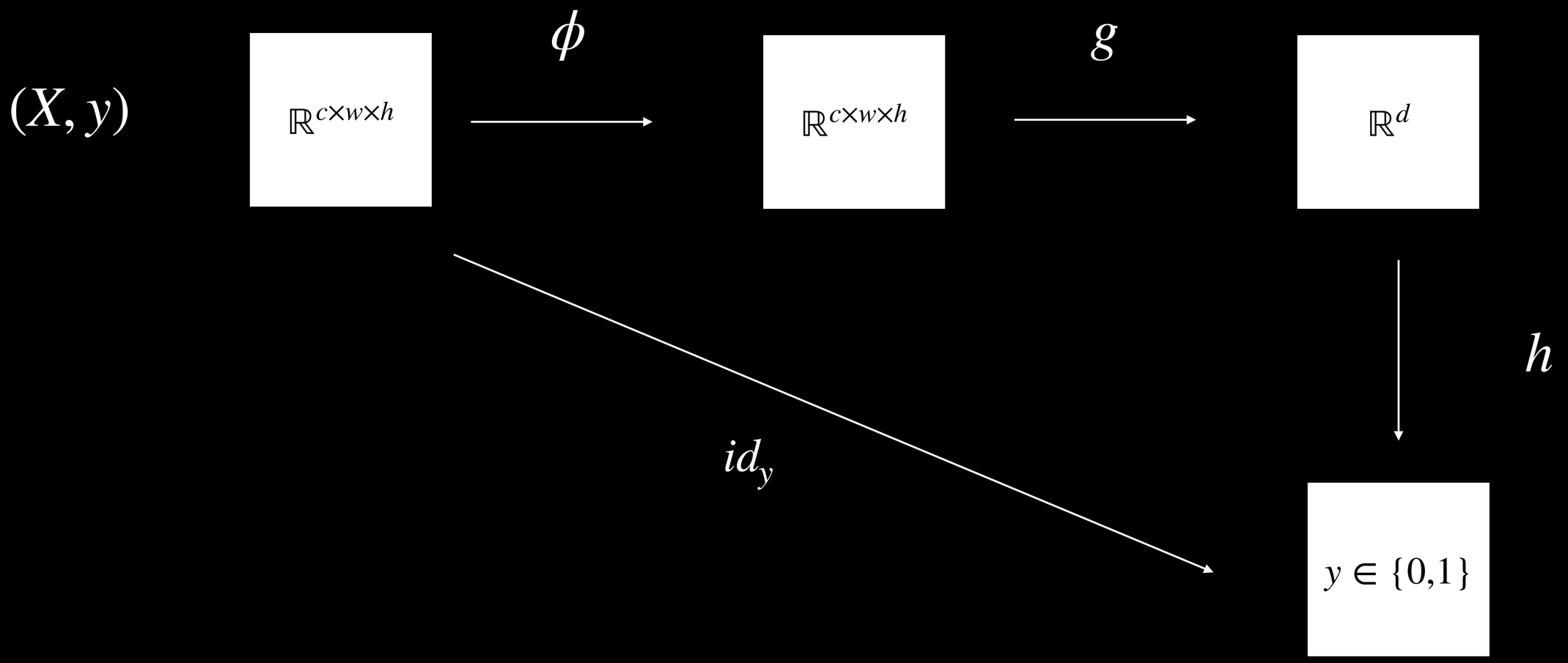
Supervised learning

Lets asume we have some labeled dataset for the next few slides:

$$X = \{x_1, \ldots, x_n \,|\, x \in \mathbb{R}^{c \times w \times h}\}$$

$$y = \{y_1, \ldots, y_n \,|\, y \in \{0,1\}\}$$

Supervised learning



$$id = f = h \circ g \circ \phi$$

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.

It is just a different name for what we have been using the last 5 lessons.

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

Contrast with supervised learning

Encoder - decoder

- Let's start with writing the mapping f: $X \rightarrow \{0,1\}$ a bit more verbose:

$$X \rightarrow \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2} \rightarrow \ldots \rightarrow \mathbb{R}^{d_m} \rightarrow \ldots \rightarrow \mathbb{R}^{d_n} \rightarrow \{0,1\}$$

Contrast with supervised learning

Encoder - decoder

- Let's start with writing the mapping f: $X \rightarrow \{0,1\}$ a bit more verbose:

$$X \rightarrow \mathbb{R}^{d_1} \rightarrow \mathbb{R}^{d_2} \rightarrow \ldots \rightarrow \mathbb{R}^{d_m} \rightarrow \ldots \rightarrow \mathbb{R}^{d_n} \rightarrow \{0,1\}$$

- Now, instead of mapping to some label $\{0,1\}$, the idea is to map the input X back to itself. Let's split the network conceptually into an encoder-decoder architecture:

Contrast with supervised learning

Encoder - decoder

- An encoder $g = l_m \circ \sigma \circ l_{m-1} \circ \ \ldots \ \circ \sigma \circ l_1$ that maps

$$g : X \to \mathbb{R}^{d_m}$$

Contrast with supervised learning

Encoder - decoder

- An encoder $g = l_m \circ \sigma \circ l_{m-1} \circ \ \ldots \ \circ \sigma \circ l_1$ that maps

$$g : X \to \mathbb{R}^{d_m}$$

- And, instead of $h : \mathbb{R}^d \to \{0,1\}$, we will now create a decoder

$$h : \mathbb{R}^{d_m} \to \mathbb{R}^{c \times w \times h}$$

- that reconstructs the *input:* $h = f_n \circ \sigma \circ f_{n-1} \circ \ \ldots \ \circ \sigma \circ f_{m+1}$

Contrast with supervised learning

Encoder - decoder

Supervised

$$X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_m} \to \ldots \to \mathbb{R}^{d_n} \to \{0,1\}$$

Contrast with supervised learning

Encoder - decoder

Supervised

$$X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_m} \to \ldots \to \mathbb{R}^{d_n} \to \{0,1\}$$

Unsupervised

$$X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_m} \to \ldots \to \mathbb{R}^{d_n} \to X$$

Contrast with supervised learning

Encoder - decoder

Supervised

$$X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_m} \to \ldots \to \mathbb{R}^{d_n} \to \{0,1\}$$

Unsupervised

$$X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_m} \to \ldots \to \mathbb{R}^{d_n} \to X$$

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

Contrast with supervised learning

Encoder - decoder

Supervised

$$X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_m} \to \ldots \to \mathbb{R}^{d_n} \to \{0,1\}$$

Unsupervised

$$X \to \mathbb{R}^{d_1} \to \mathbb{R}^{d_2} \to \ldots \to \mathbb{R}^{d_m} \to \ldots \to \mathbb{R}^{d_n} \to X$$

$$AE\colon \mathbb{R}^{c \times w \times h} \to \mathbb{R}^{d} \to \mathbb{R}^{c \times w \times h}$$

Contrast with supervised learning

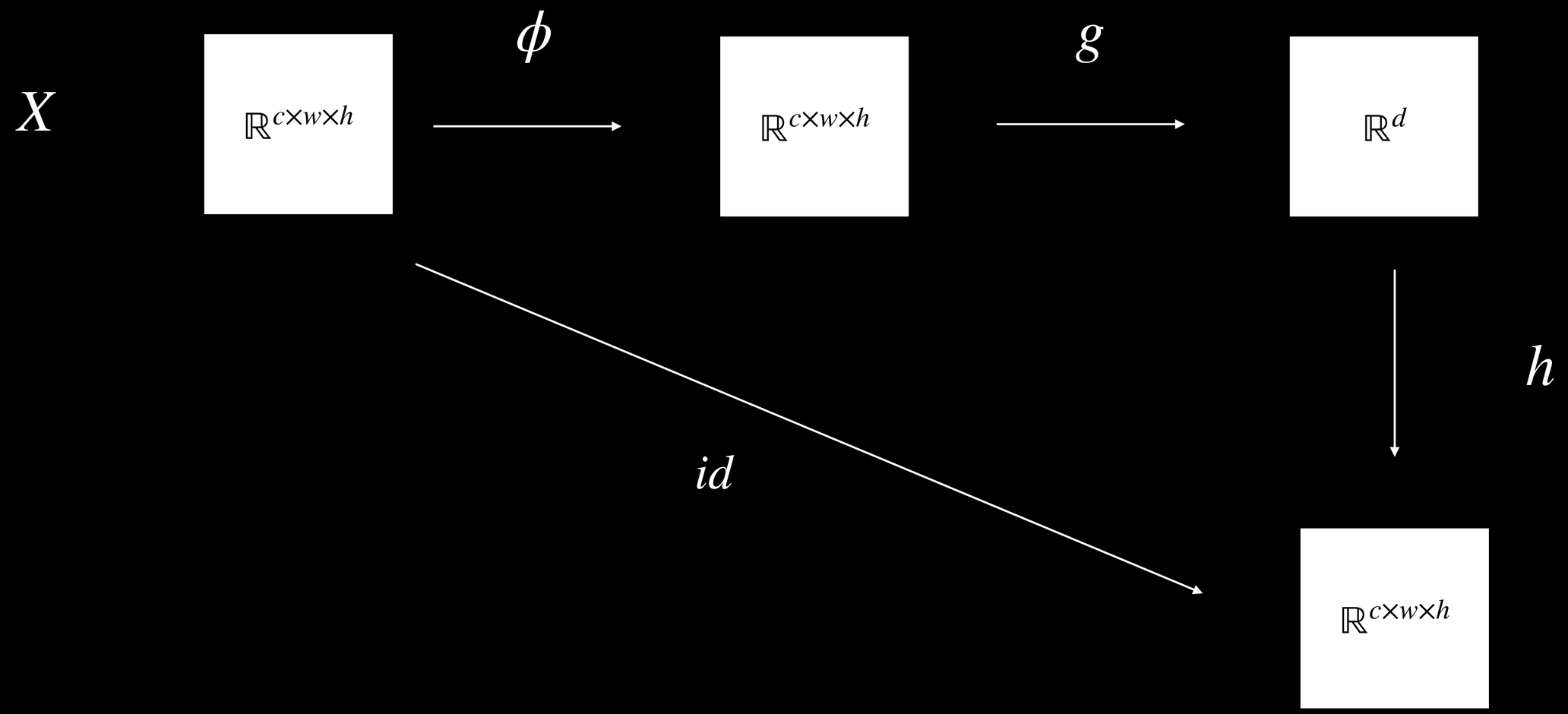
Reducing dimensionality

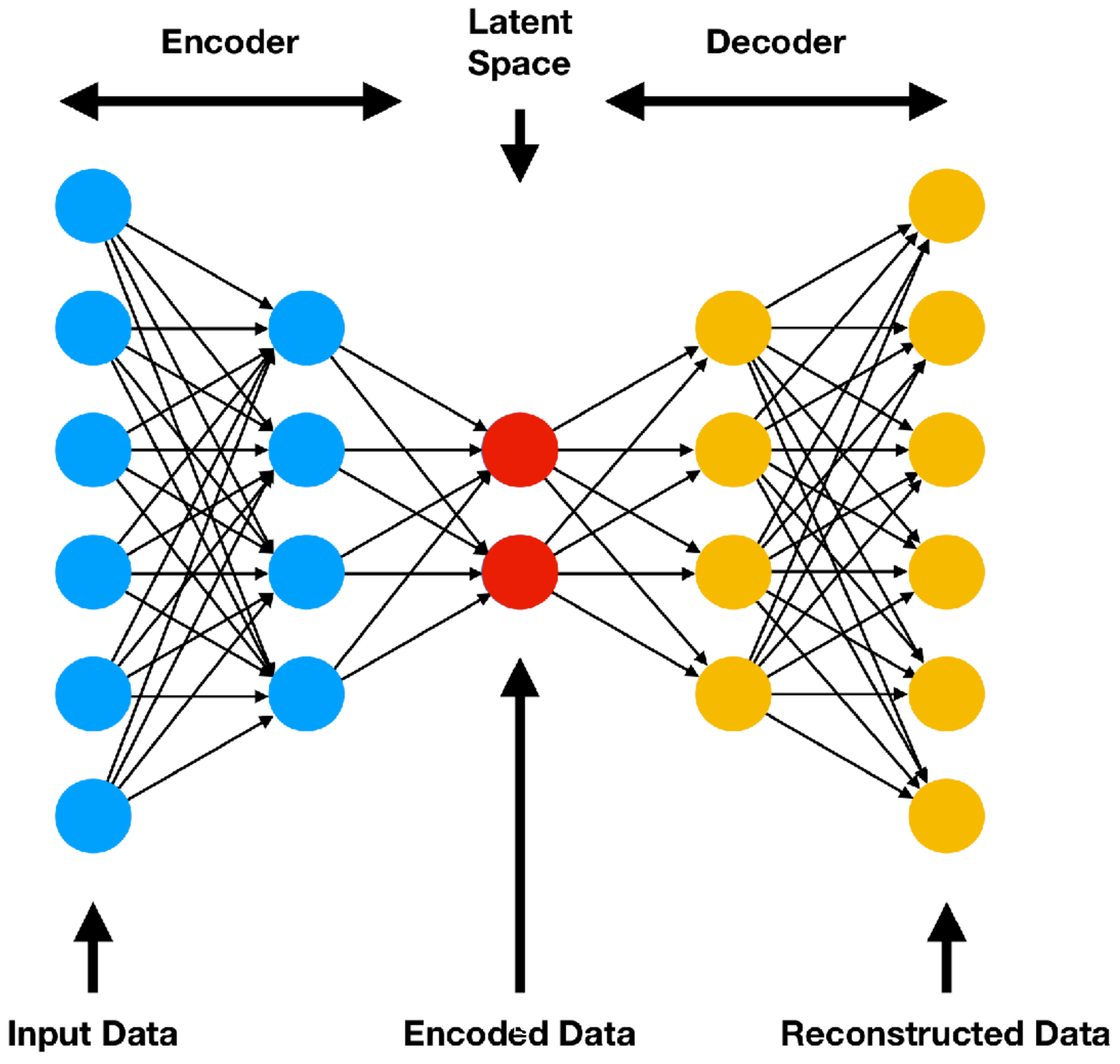
An autoencoder is a network $AE(x) = h(g(x))$, which gives us:

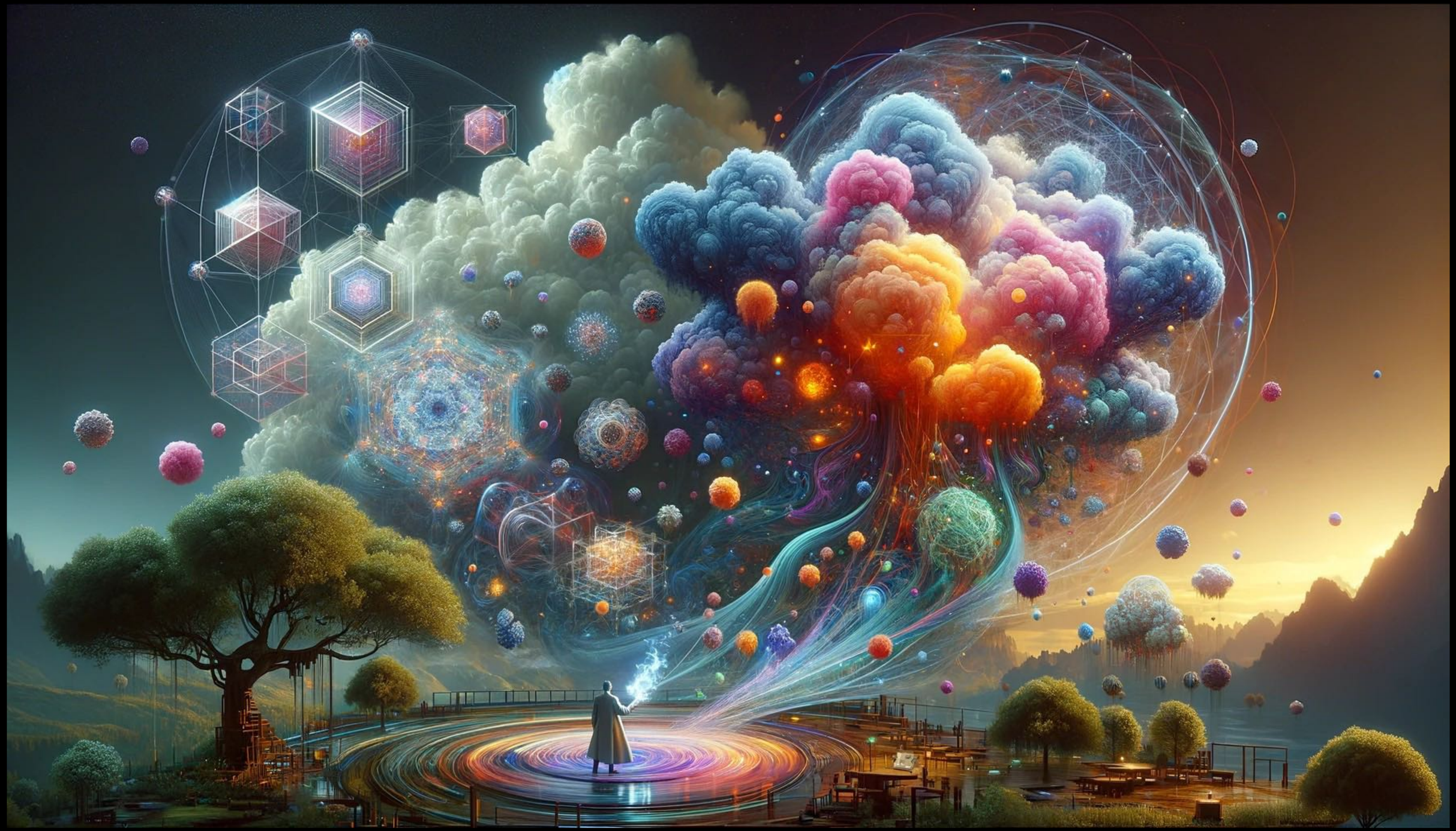
$$AE: X \to \mathbb{R}^d \to X$$

The encoder maps input space X to latent space, that typically involves a reduction in dimensionality: $dim(Z) < dim(X)$, and then maps back to the original.

Auto Encoder / generative




Encoder
Latent Space
Decoder
Input Data
Encoded Data
Reconstructed Data



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 $h(g(x))$ and x

Contrast with supervised learning

Minimize reconstruction error

Why do we do this? We already have X, so isn't it pointless to predict X?

Well, it's not the output we are after, but it is actually what happens in the latent space what we find to be valuable!

Key differences with supervised learning

- By restricting the dimensionality of Z, we force the model to learn to be as efficient as possible and make summaries.

Key differences with supervised learning

- By restricting the dimensionality of Z, we force the model to learn to be as efficient as possible and make summaries.
- We dont need external labels

Key differences with supervised learning

- By restricting the dimensionality of Z, we force the model to learn to be as efficient as possible and make summaries.
- We dont need external labels
- We dont focus on accuracy perse, but on usefull summarisation (in terms of our endgoal). We dont want a perfect reconstruction, but a latent space that captures the essence!

Key differences with supervised learning

- By restricting the dimensionality of Z, we force the model to learn to be as efficient as possible and make summaries.
- We dont need external labels
- We dont focus on accuracy perse, but on usefull summarisation (in terms of our endgoal). We dont want a perfect reconstruction, but a latent space that captures the essence!
- Generative AI explores the latent space as a source of creativity

Key differences with supervised learning

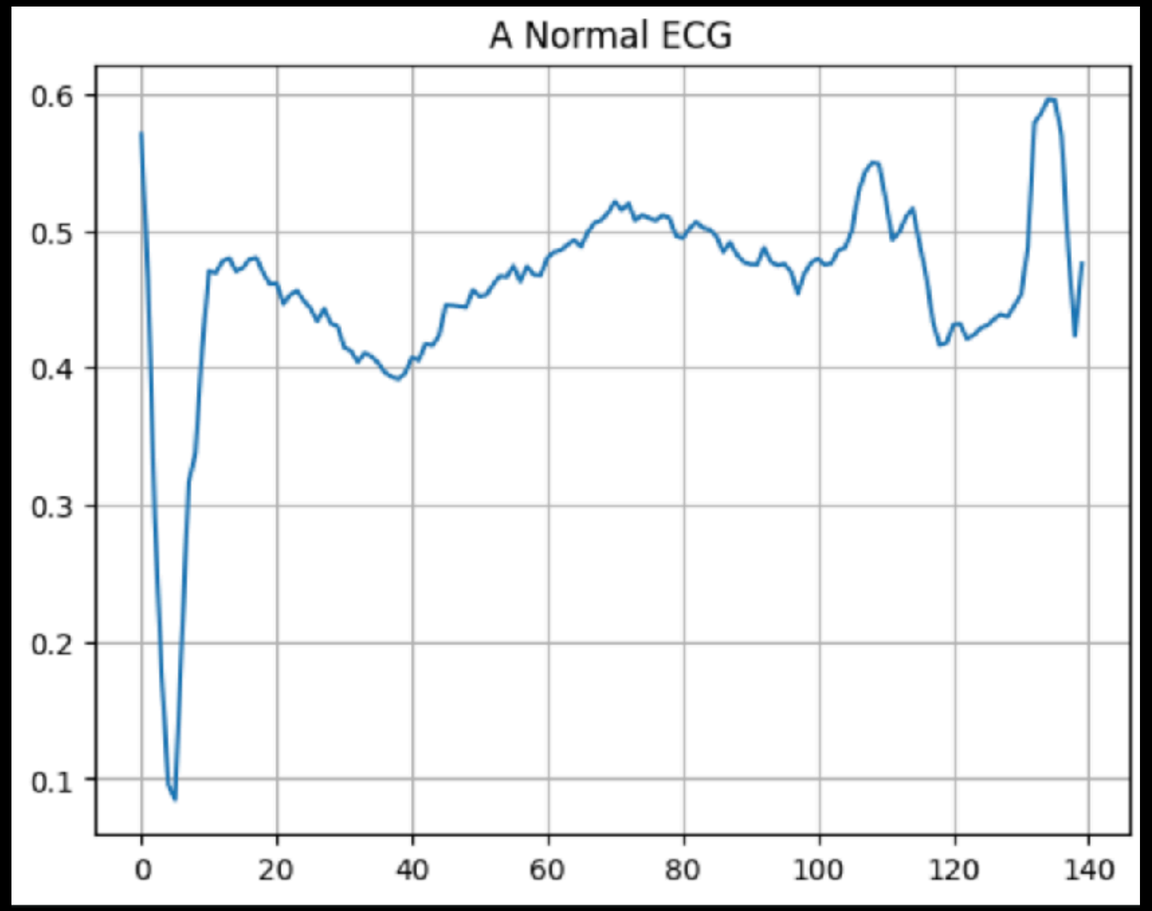
- By restricting the dimensionality of Z, we force the model to learn to be as efficient as possible and make summaries.
- We dont need external labels
- We dont focus on accuracy perse, but on usefull summarisation (in terms of our endgoal). We dont want a perfect reconstruction, but a latent space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

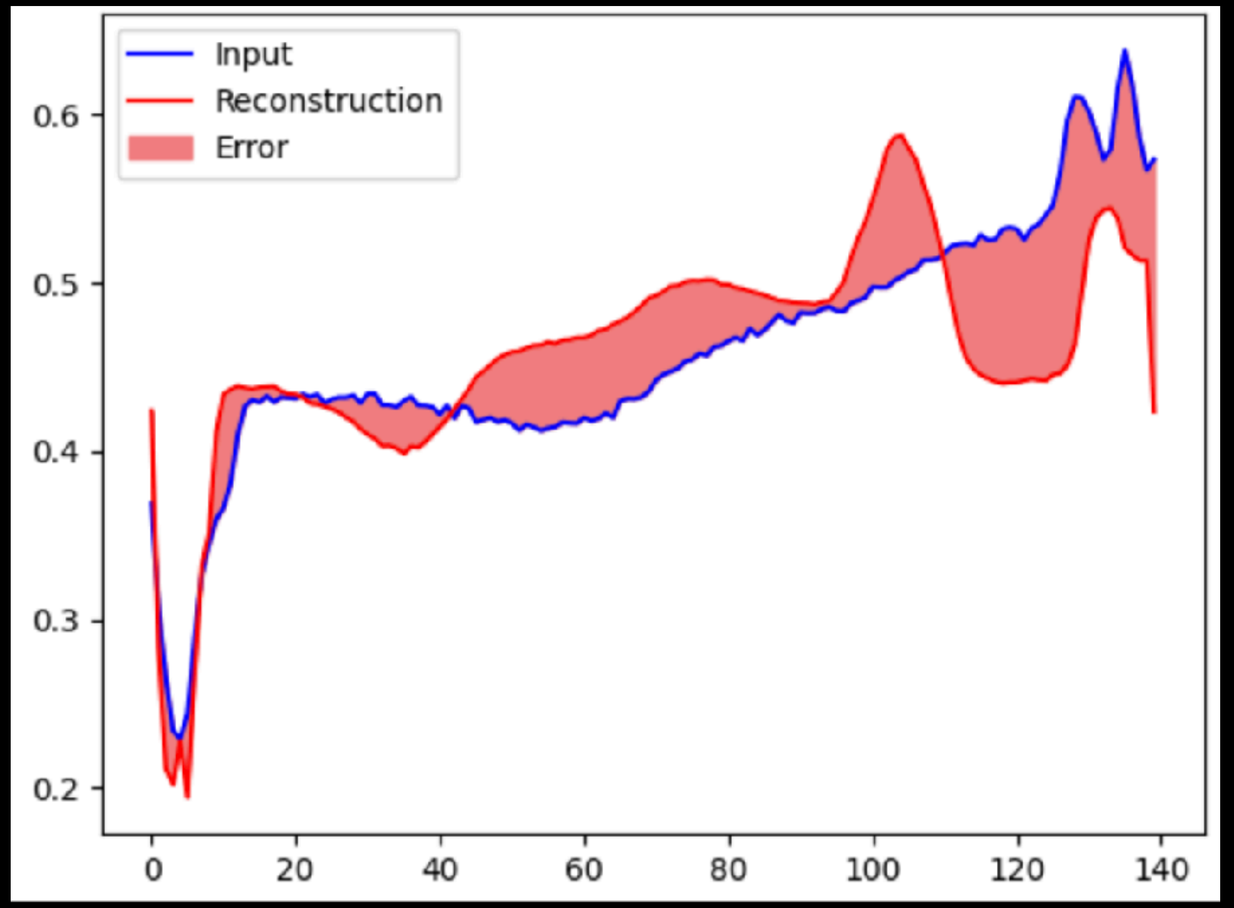
Motivation, part 2

motivation for autoencoders

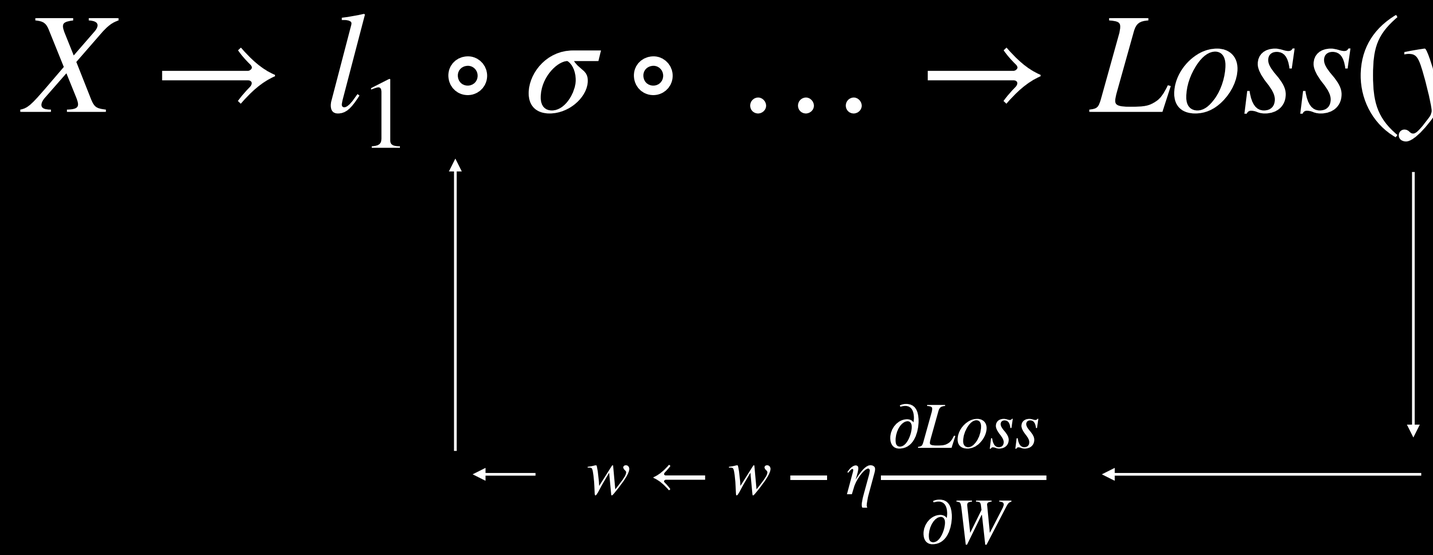
- Dimensionality reduction (encoder) : Capture the most significant features, making it easier to visualise and process data.
- Data Compression (encoder): the latent space is compressed, so we can use that in itself.
- Anomaly detection (encoder-decoder): By learning the “normal” pattern of data, the reconstruction error will be bigger with anomalies even though (more precise, exactly because of this) 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

Anomaly detection



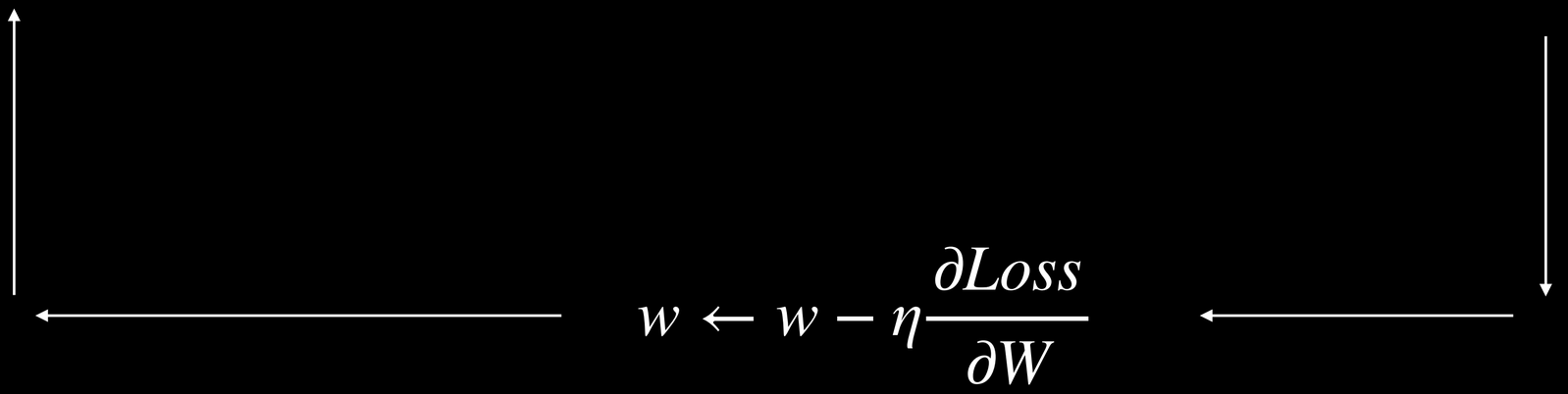


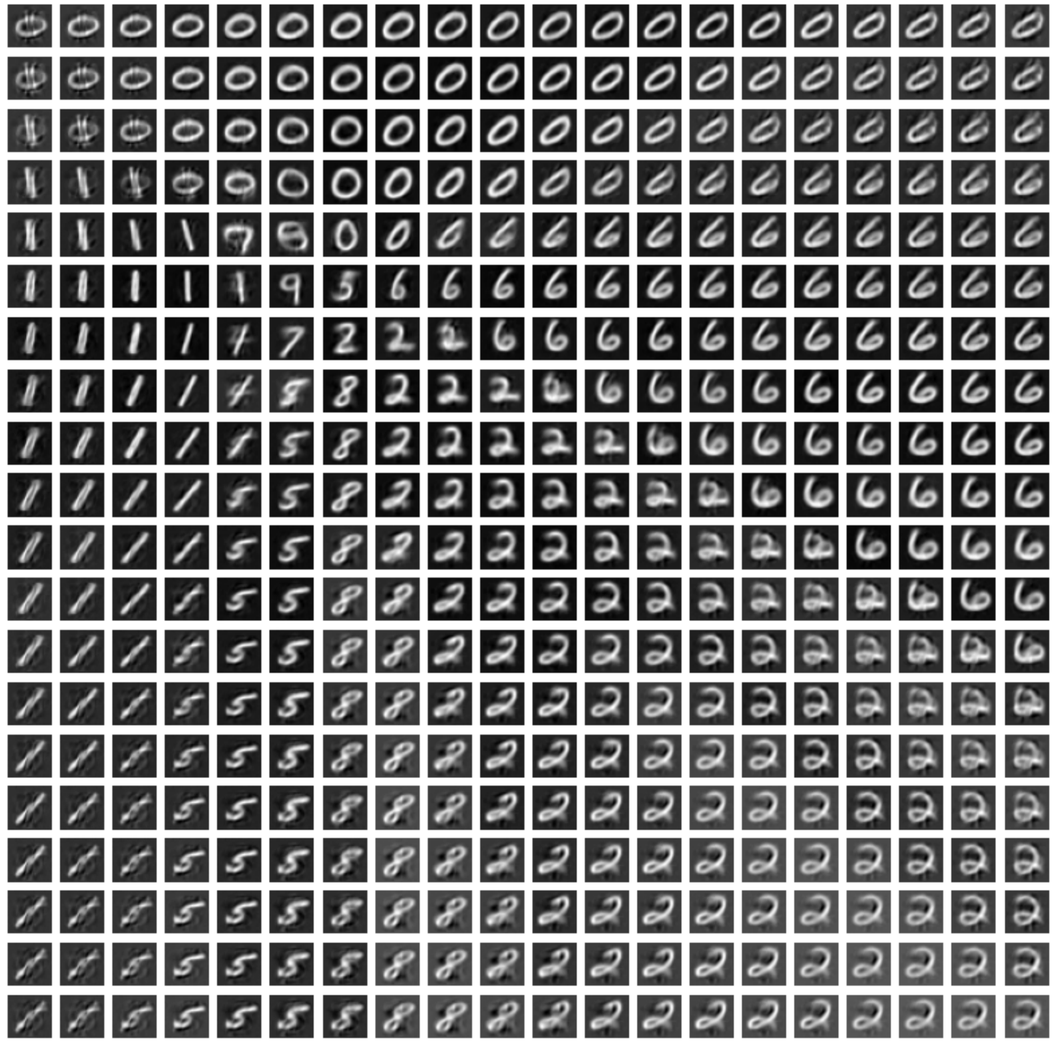
Supervised


$X \rightarrow l_1 \circ \sigma \circ \ \ldots \ \rightarrow Loss(y, \hat{y})$
$w \leftarrow w - \eta \frac{\partial Loss}{\partial W}$

Autoencoder

$$X \to l_m \circ \sigma \circ \ \ldots \to Z \to l_n \circ \sigma \circ \ \ldots \to \hat{X} \to Loss(X, \hat{X})$$





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

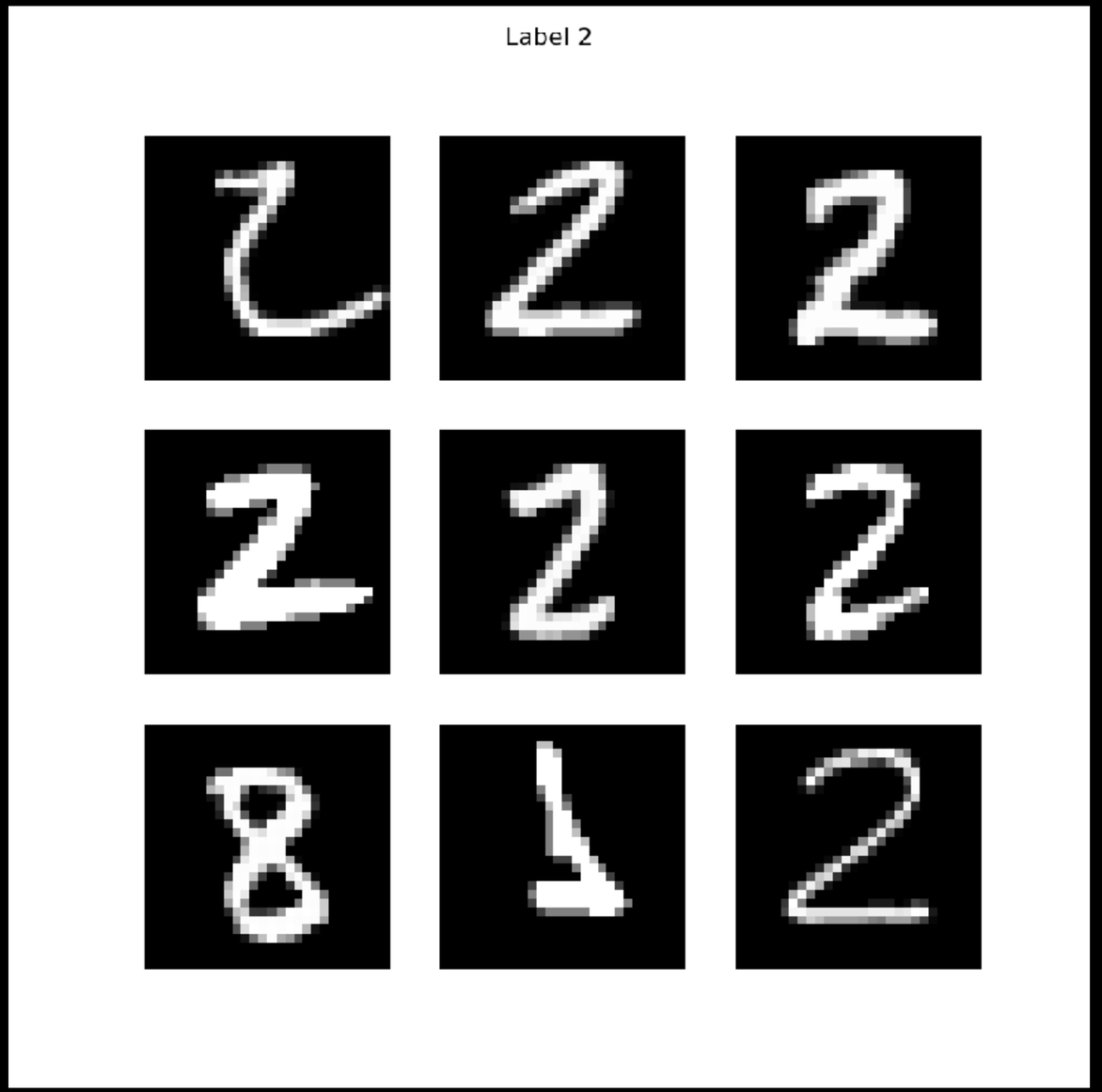


Fig: the 9 items closest to the new input

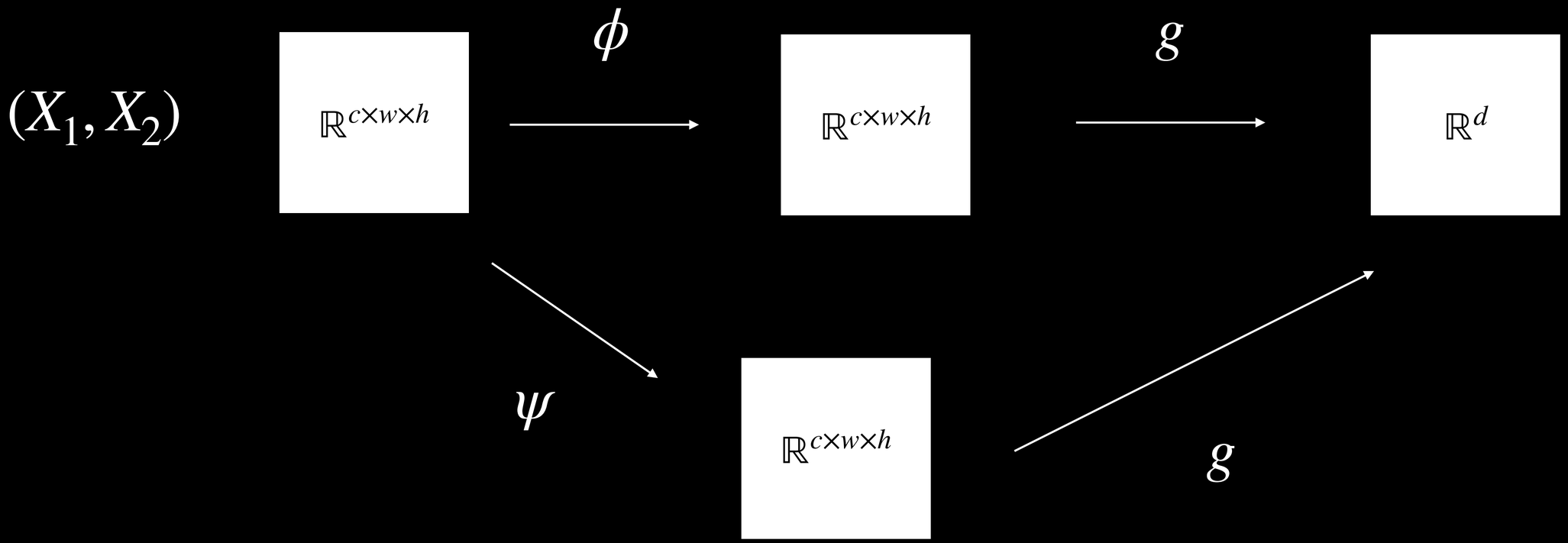
Contrastive learning

Siamese networks

Semisupervised

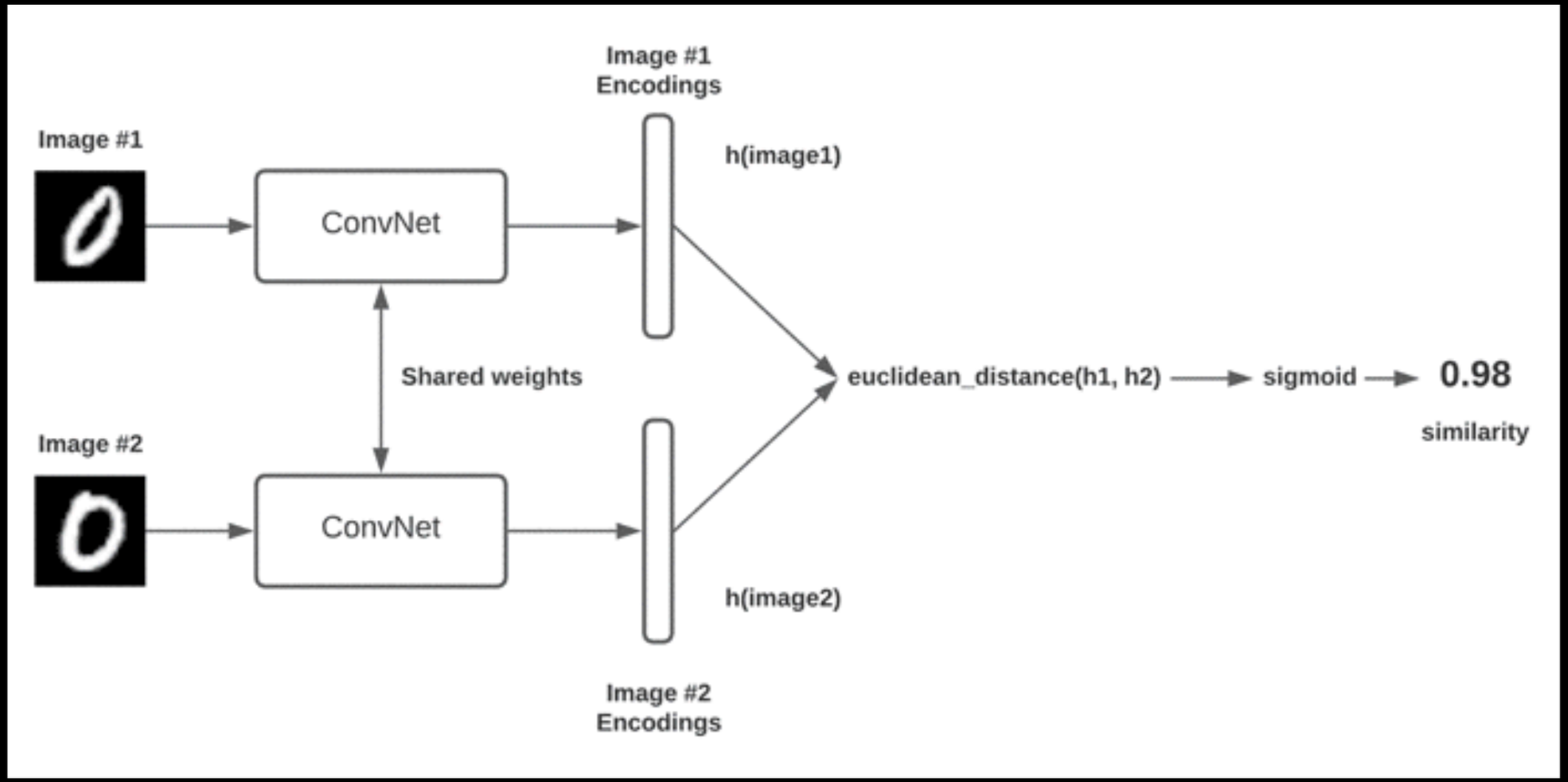
- A typical motivation for contrastive learning, like siamese networks, is to check against a ground truth, instead of the usual classification
- For example, testing if a signature on a document is the same as on an id, or check if a face matches and id picture
- The usual supervised approach would not work, a siamese network makes this much easier.

Contrastive



Siamese networks

Semisupervised



Siamese networks / contrastive networks

Semisupervised

- $X = \{x_1, \ldots, x_j | x \in \mathbb{R}^D\}$
- A labeling function $i: X \times X \rightarrow \{0,1\}$ defined as $i(x_i, x_j) = \begin{cases} 1 & \text{if } x_i \sim x_j \\ 0 & \text{if } x_i \neq x_j \end{cases}$
- An encoder $g: x \rightarrow Z$ 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.

Self-Supervised learning - JEPA

Joint-Embedding Predictive Architecture

Cognitive learning theories have suggested that a driving mechanism behind representation learning in biological systems is

- the adaptation of an internal model to predict sensory input responses

See: *Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture*, Assran et al. (2023)

Joint-Embedding Predictive Architecture

Compare this with this definition of intelligence:

- intelligence as the capacity to accumulate evidence for a generative model of one’s sensed world

See: Friston, Karl J., et al. "Designing ecosystems of intelligence from first principles." *Collective Intelligence* 3.1 (2024): 26339137231222481.

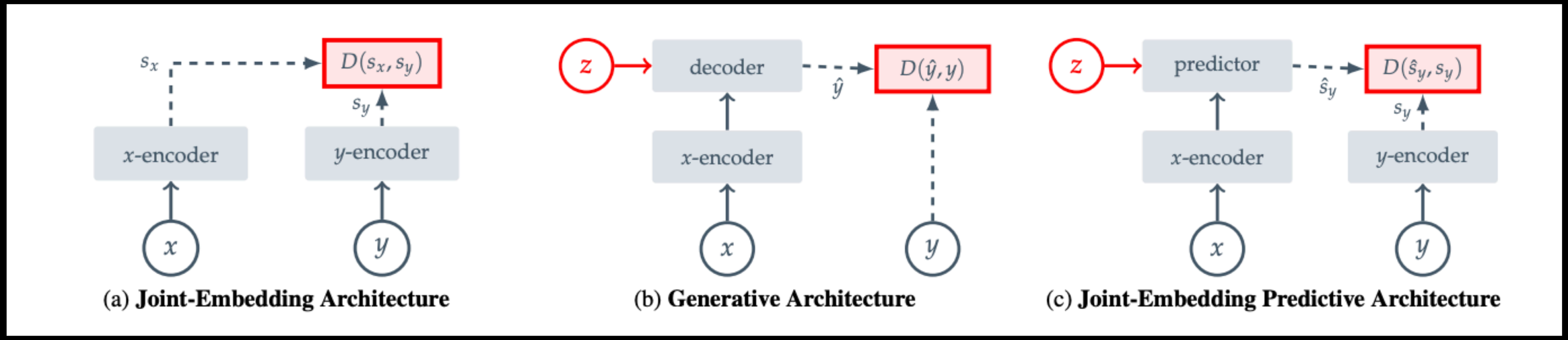
Joint-Embedding Predictive Architecture

Compared to generative methods that predict in pixel/token space, I-JEPA predicts *directly in embedding-space*.

This means it predicts the representation, not the details!

Compare this to planning a journey on the level of all the muscle contractions, versus planning it with a simplified model of the route (go left after the big green tree).

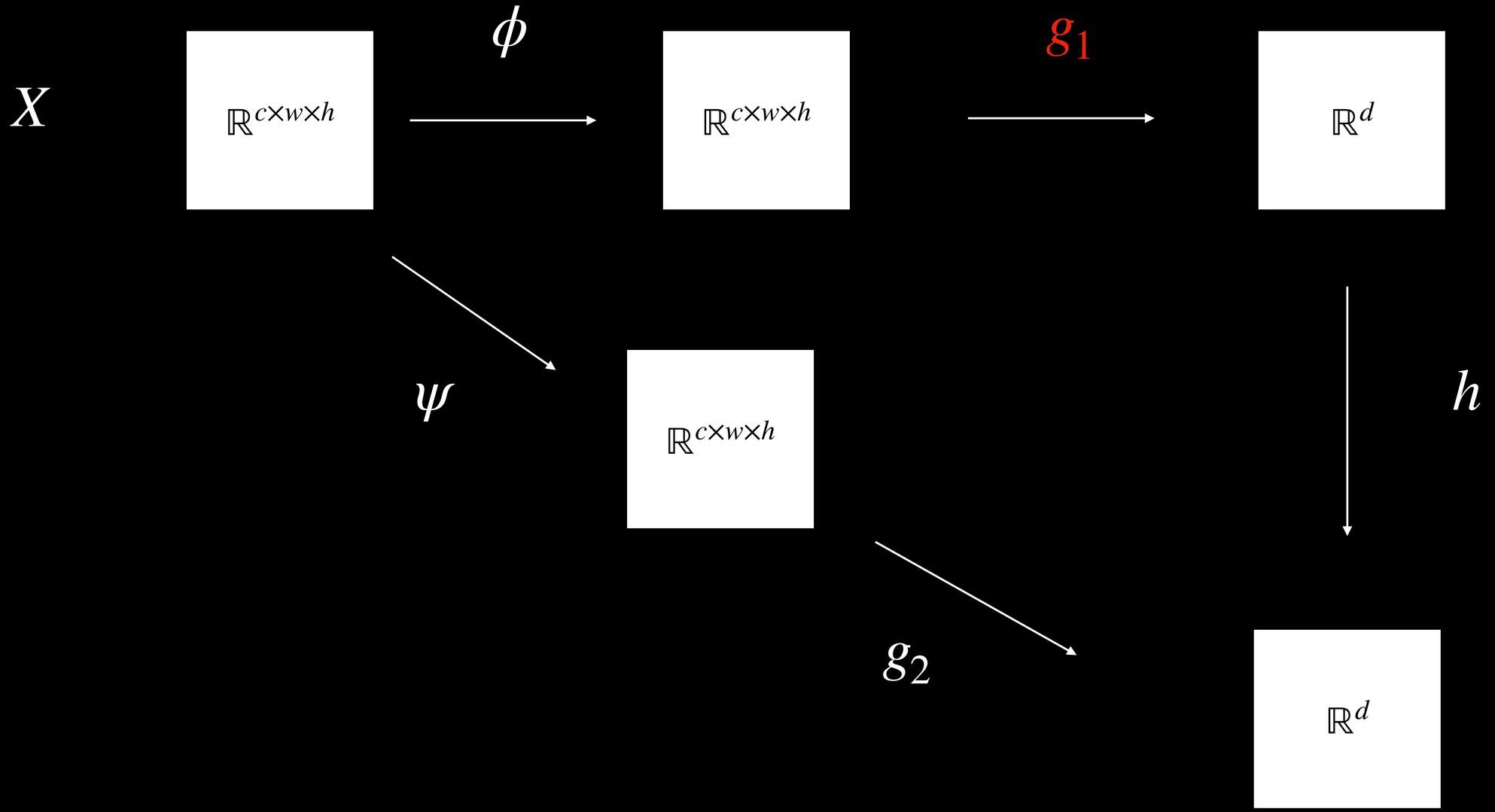
Joint-Embedding Predictive Architecture



Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture, Assran et al. (2023)

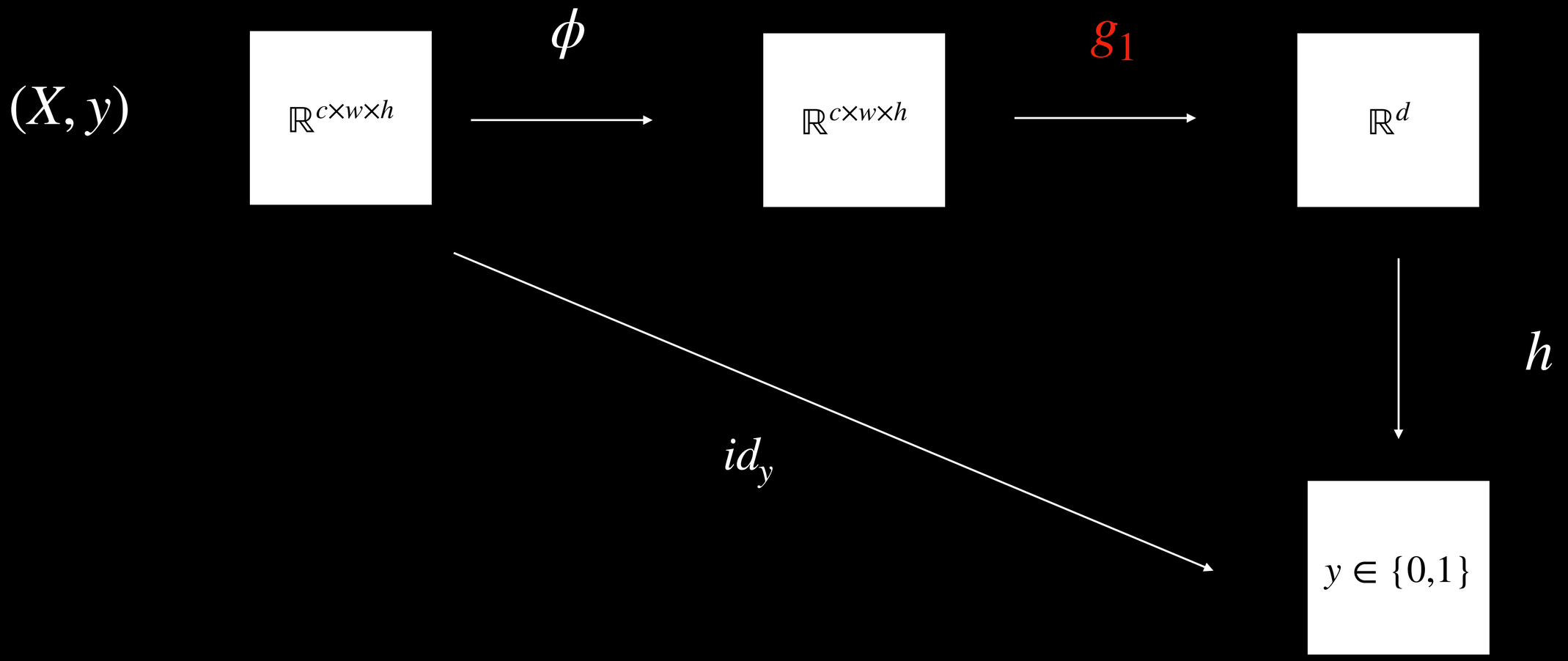
I-JEPA

selfsupervised - big dataset

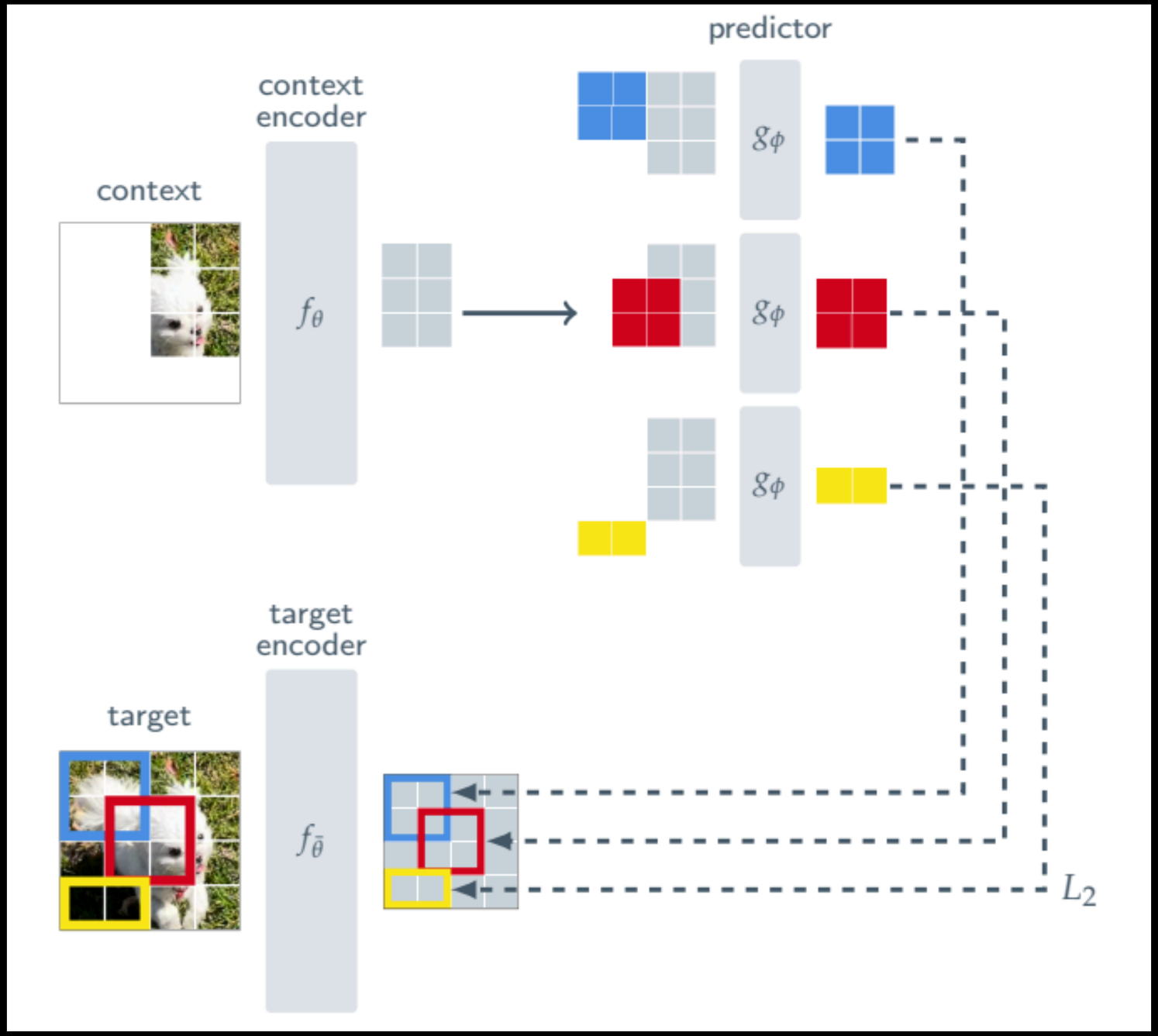


Downstream task

supervised - small dataset



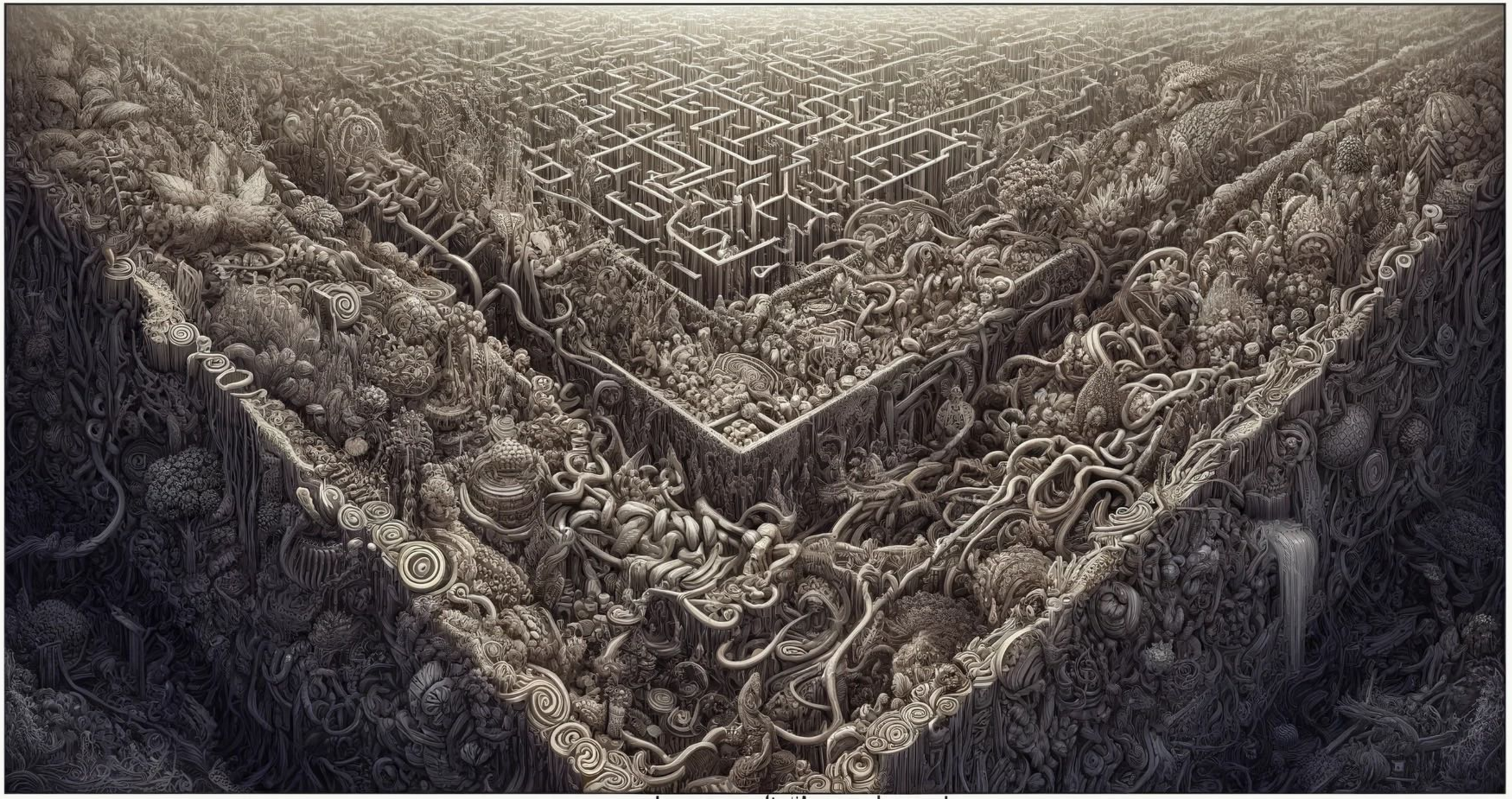
$$id = f = h \circ g \circ \phi$$



Self-Supervised Learning from Images with a Joint-Embedding Predictive Architecture, Assran et al. (2023)

Joint-Embedding Predictive Architecture

Exactly because the input is “damaged”, the model is forced to reduce focus on details and take a higher level of abstraction.


Generative method | To/tonle level