AUTOENCODER

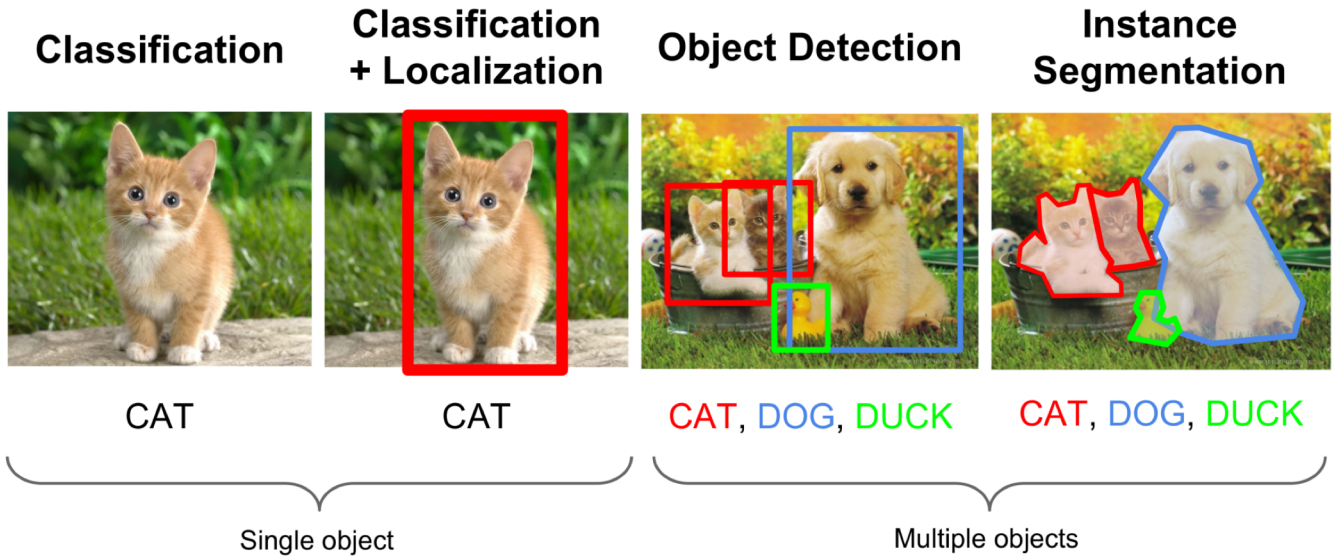
Narges Norouzi

AGENDA

- Unsupervised Learning (Introduction)
- Autoencoder (AE)
- Convolutional AE
- Denoising AE
- Stacked AE

SUPERVISED LEARNING

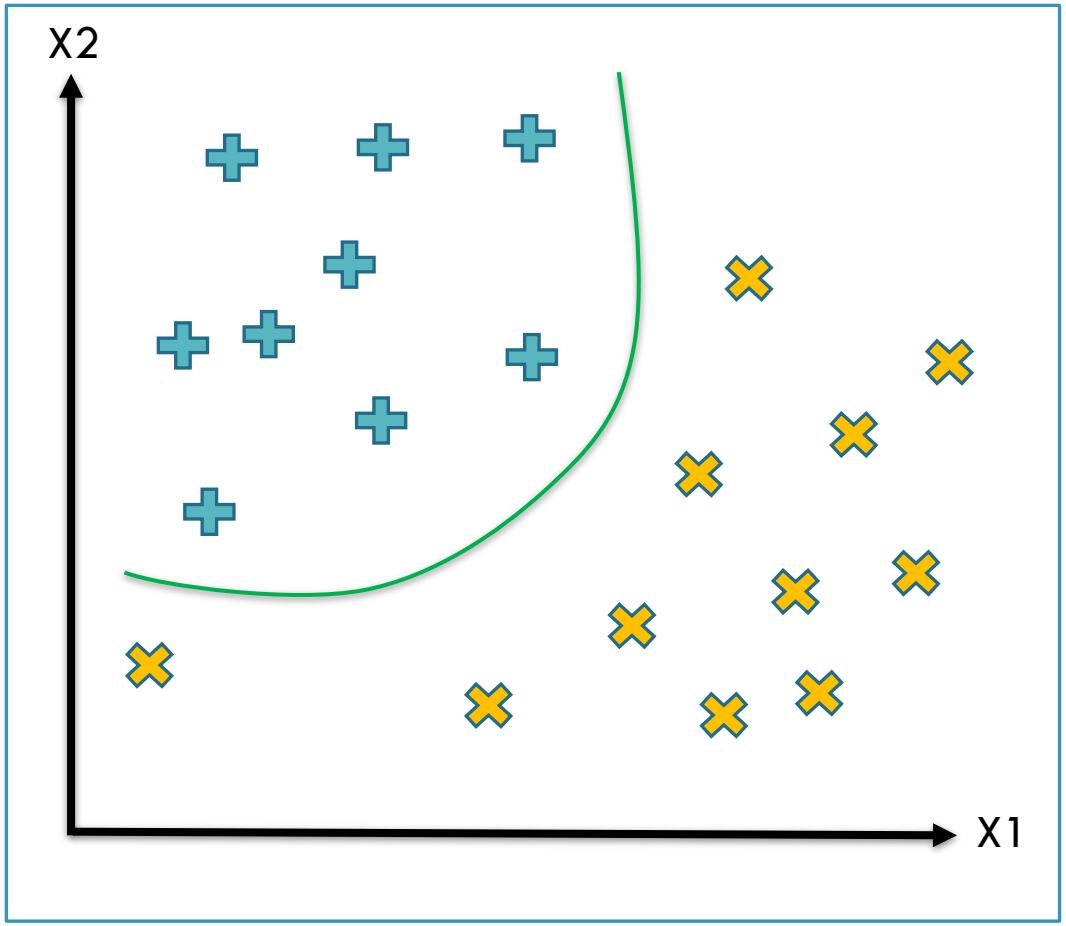
- Supervised Learning
- Data: (X, Y)
- Goal: Learn a Mapping Function f where: $f(X) = Y$



SUPERVISED LEARNING

- Examples: Classification

Decision Trees	Naïve Bayes
KNN	SVM
Perceptron	Multi Layer Perceptron



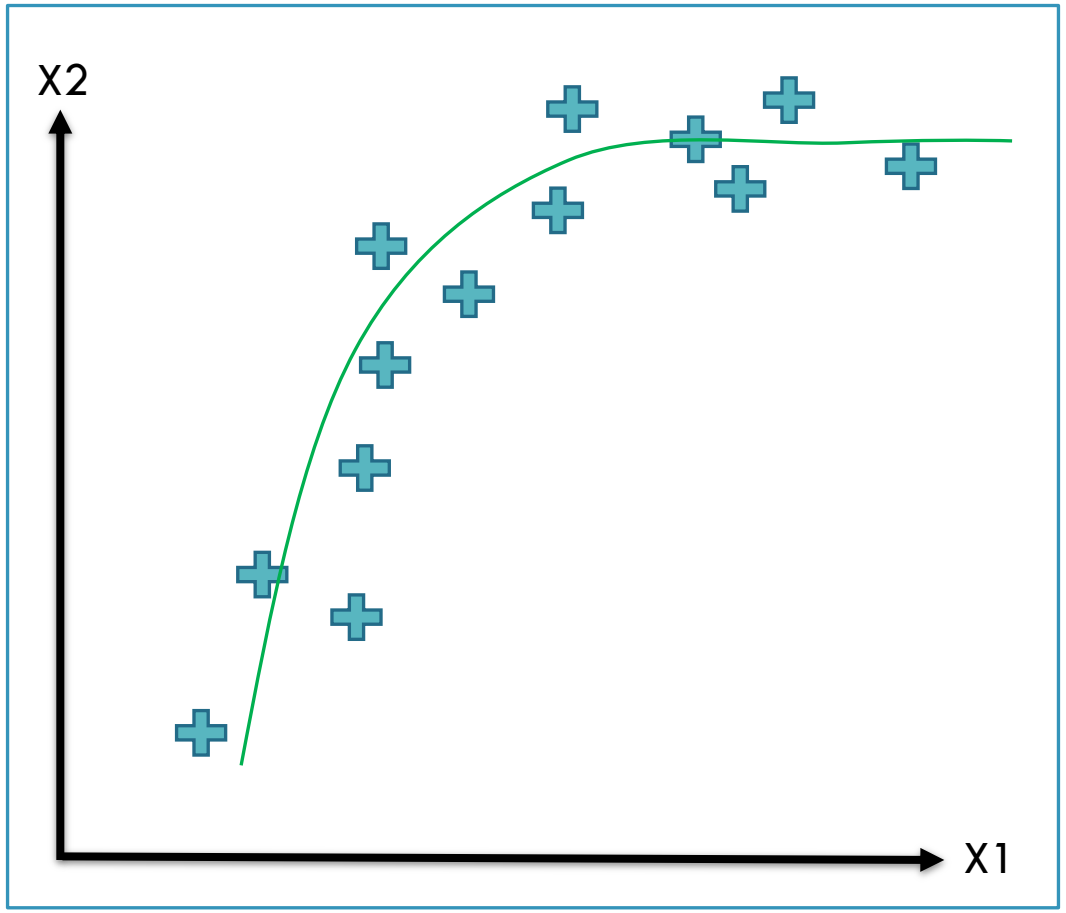
Classification

SUPERVISED LEARNING

- Examples: Regression

Linear Regression

Logistic Regression



Regression

SUPERVISED VS UNSUPERVISED LEARNING

01

What happens when our labels are noisy?

- Missing values
- Labeled incorrectly

02

What happens where we don't have labels for training **at all**?

SUPERVISED VS UNSUPERVISED LEARNING

Up until now we have encountered mostly **Supervised Learning** problems and algorithms.

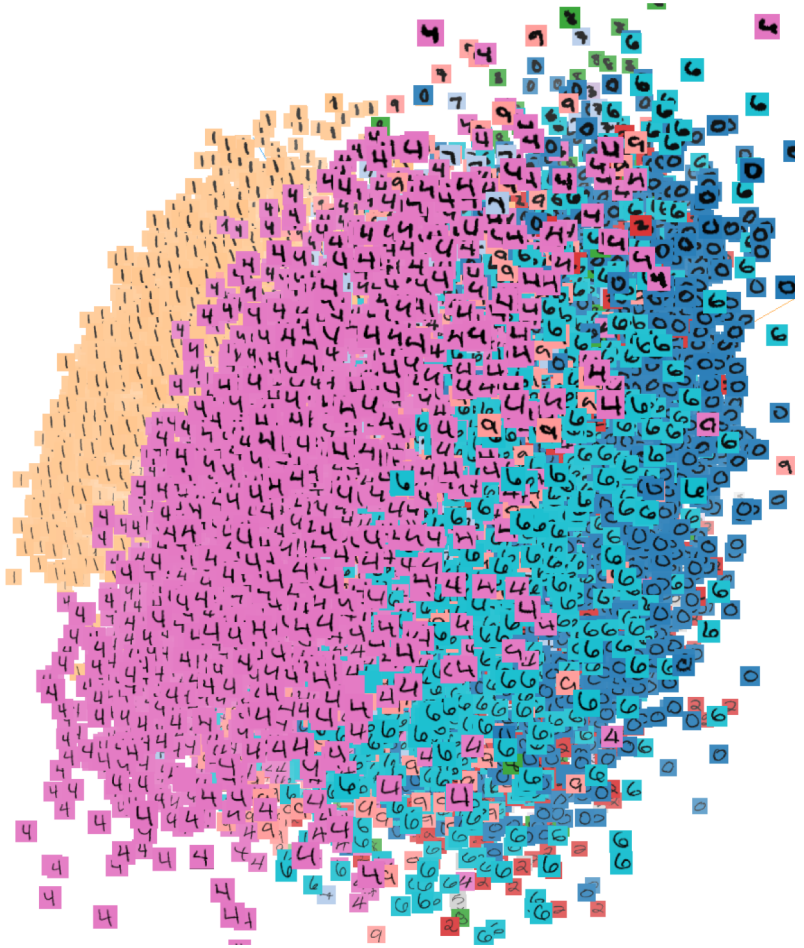
Lets talk about **Unsupervised Learning**

UNSUPERVISED LEARNING

Unsupervised Learning

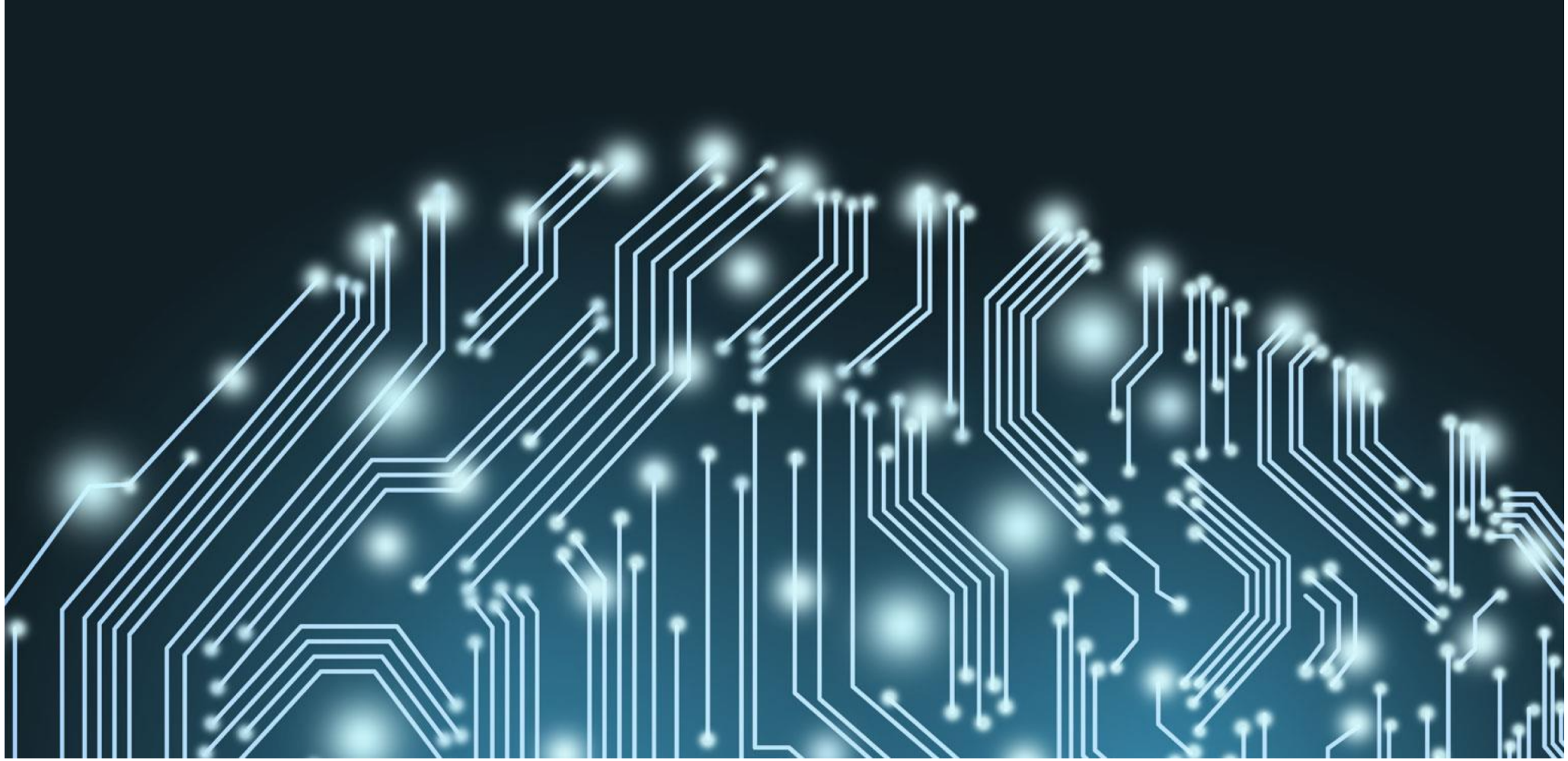
Data: X (no labels!)

Goal: Learn the structure of the data (learn correlations between features)



UNSUPERVISED LEARNING

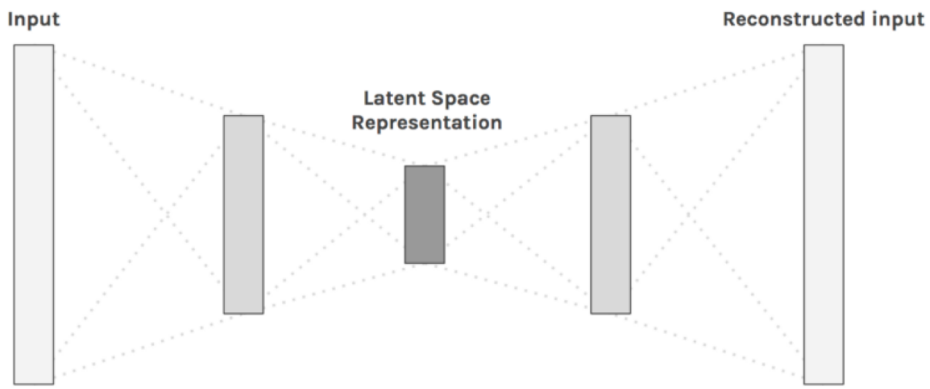
Examples: Clustering, **Compression, Feature & Representation learning, Dimensionality reduction,** Generative models ,etc.



AUTOENCODERS

COMPONENTS OF AE

- Encoder: encodes input to a latent space representation
- Code: latent space representation
- Decoder: reconstruction of input from the latent space features



SIMPLE IDEA

Given data x (no labels) we would like to learn the functions f (encoder) and g (decoder) where:

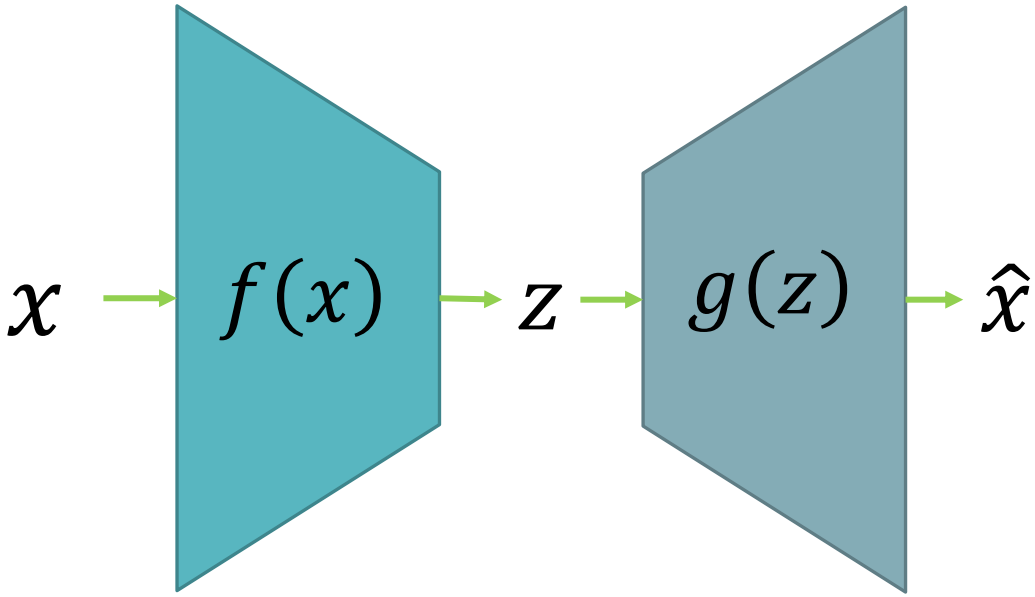
$$f(x) = s(wx + b) = z$$

and

$$g(z) = s(w'z + b') = \hat{x}$$

s.t $h(x) = g\big(f(x)\big) = \hat{x}$

where h is an **approximation** of the identity function.



(z is some **latent** representation or **code** and s is a non-linearity such as the sigmoid)

($\hat{x}$ is x's reconstruction)

SIMPLE IDEA

Learning the identity function seems trivial, but with added constraints on the network (such as limiting the number of hidden neurons or regularization) we can learn information about the structure of the data.

Trying to capture the distribution of the data (data-specific!)

TRAINING THE AE

Using **Gradient Descent** we can simply train the model as any other FC NN with:

- Traditionally with *squared error loss* function

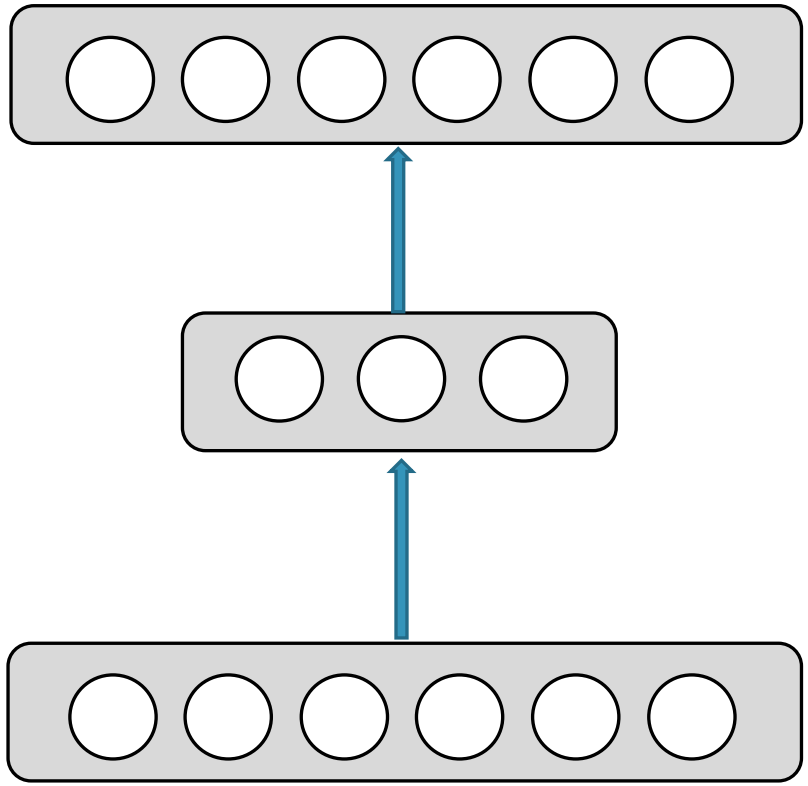
$$L(x, \hat{x}) = \|x - \hat{x}\|^2$$

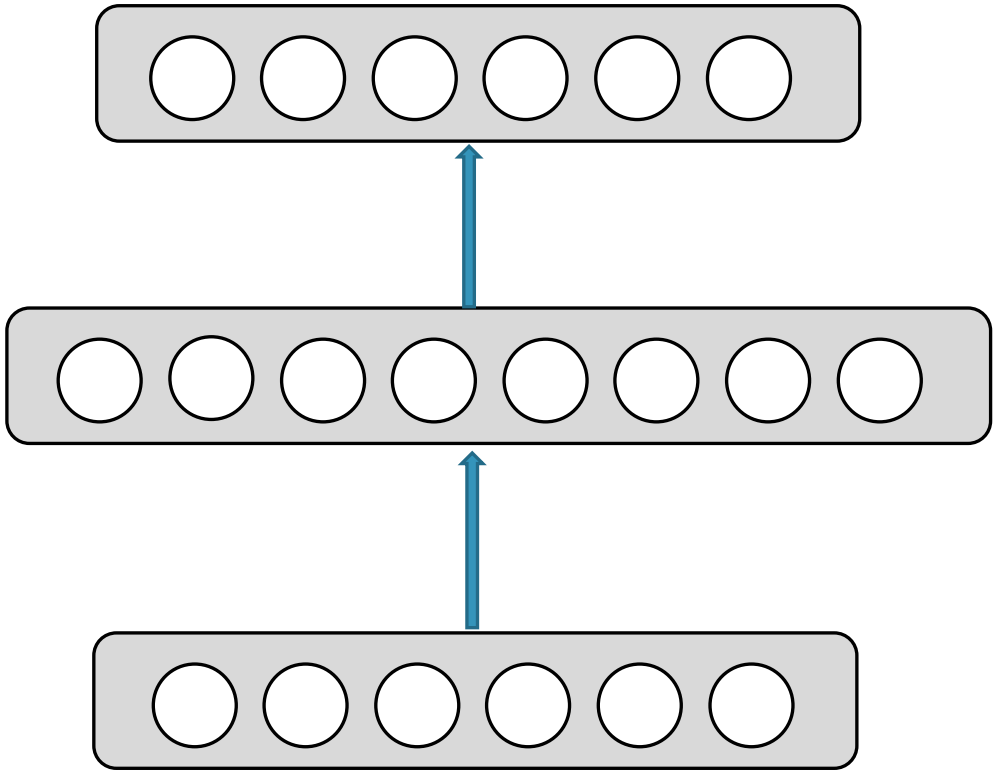
- If our input is interpreted as bit vectors or vectors of bit probabilities the *cross entropy* can be used

$$H(p, q) = -\sum_{x} p(x) \log q(x)$$

UNDERCOMPLETE AE VS OVERCOMPLETE AE

We distinguish between two types of AE structures:

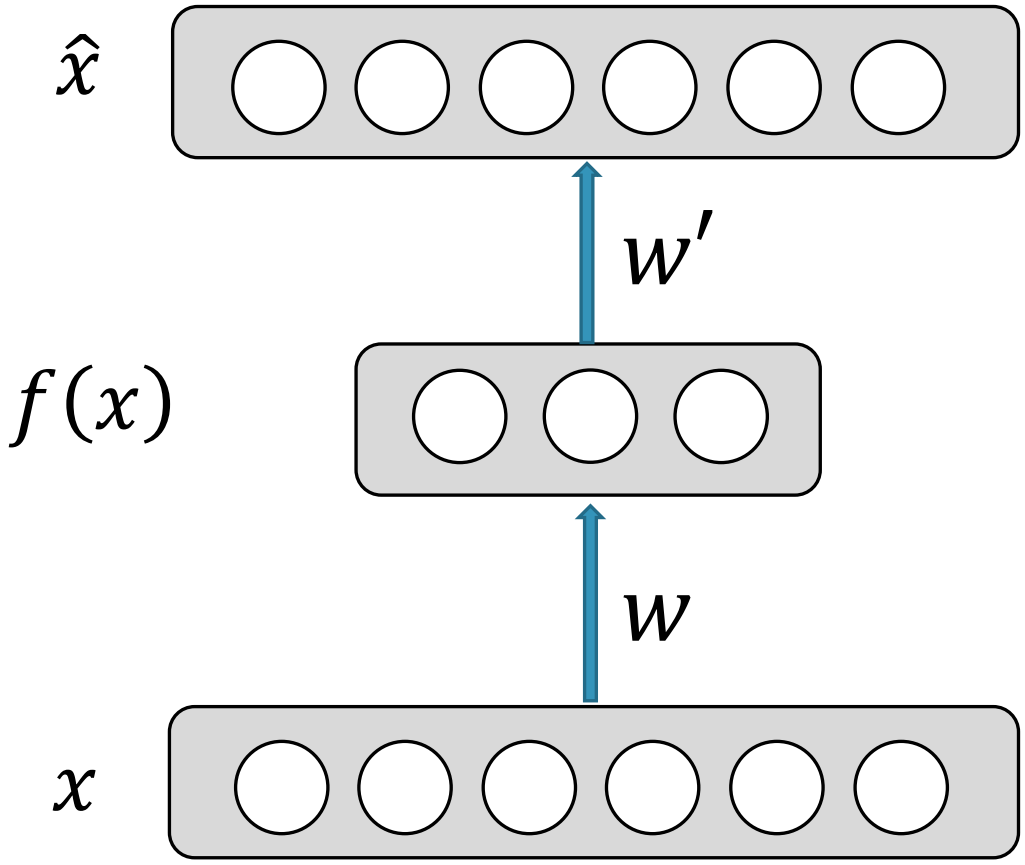




UNDERCOMPLETE AE

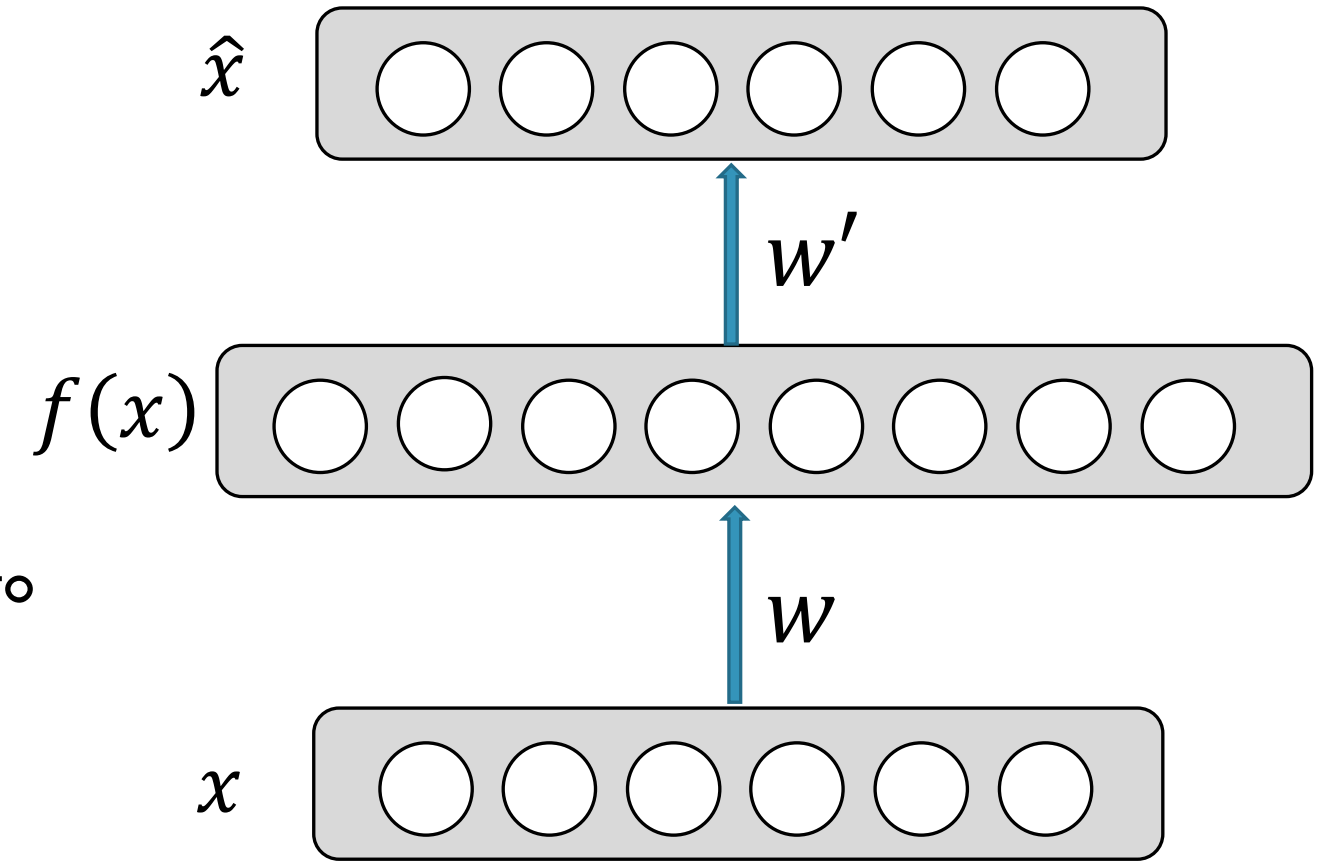
- Hidden layer is **Undercomplete** if code is smaller than the input layer
  - Compresses the input
  - Compresses well only for the training distribution.

- Hidden nodes will be
  - Good features for the training distribution.
  - Bad for other types on input



OVERCOMPLETE AE

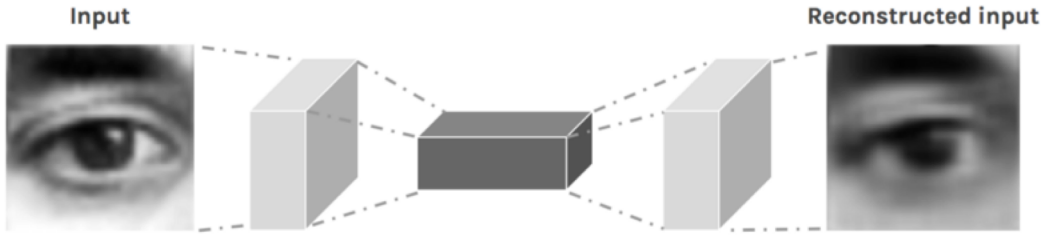
- Hidden layer is **Overcomplete** if greater than the input layer
  - No compression in hidden layer.
  - Each hidden unit could copy a different input component.
- No guarantee that the hidden units will extract meaningful structure.
- A higher dimension code helps model to learn a more complex distribution.



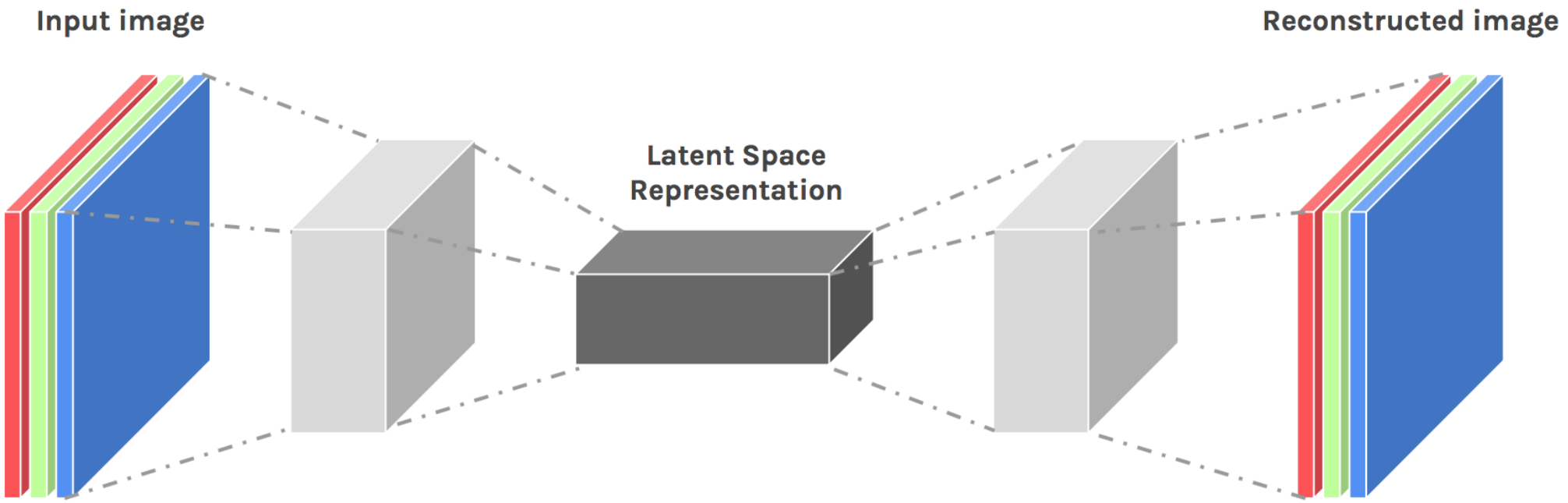
CONVOLUTIONAL AE

CAE

- CNN convert the input from **wide** and **thin** (let's say 100 x 100 px with 3 channels—RGB) to **narrow** and **thick** feature map.
- This helps the network extract **visual features** from the images, and therefore obtain a much more accurate latent space representation.
- The reconstruction process uses *upsampling* and convolutions.



CONVOLUTIONAL AE

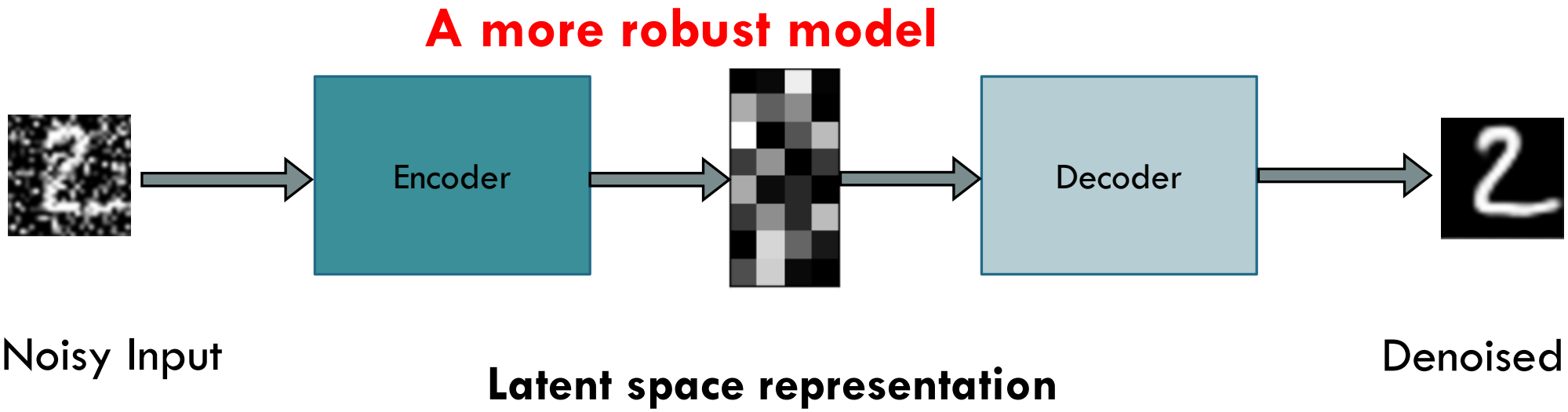


DENOISING AE

DENOISING AUTOENCODERS

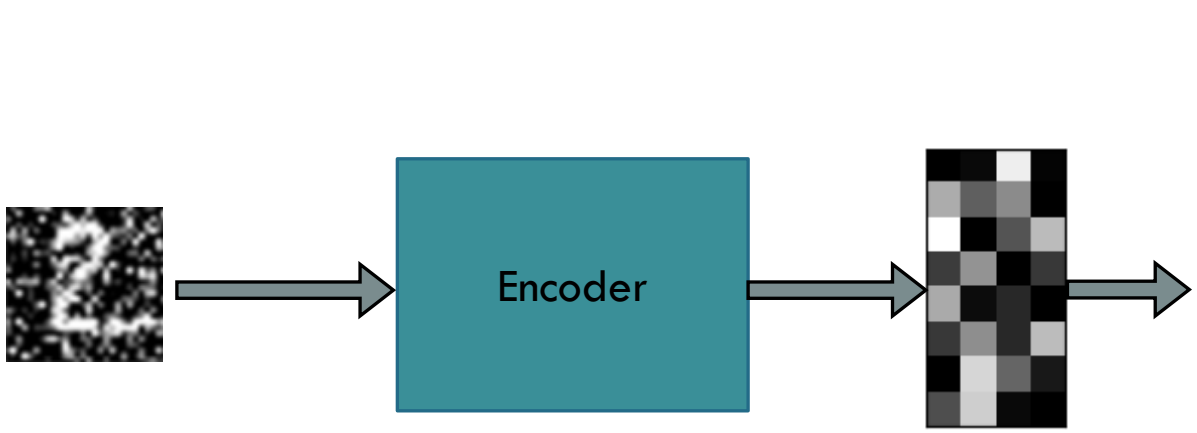
Intuition:

- We still aim to encode the input and to NOT mimic the identity function.
- We try to undo the effect of *corruption* process stochastically applied to the input.



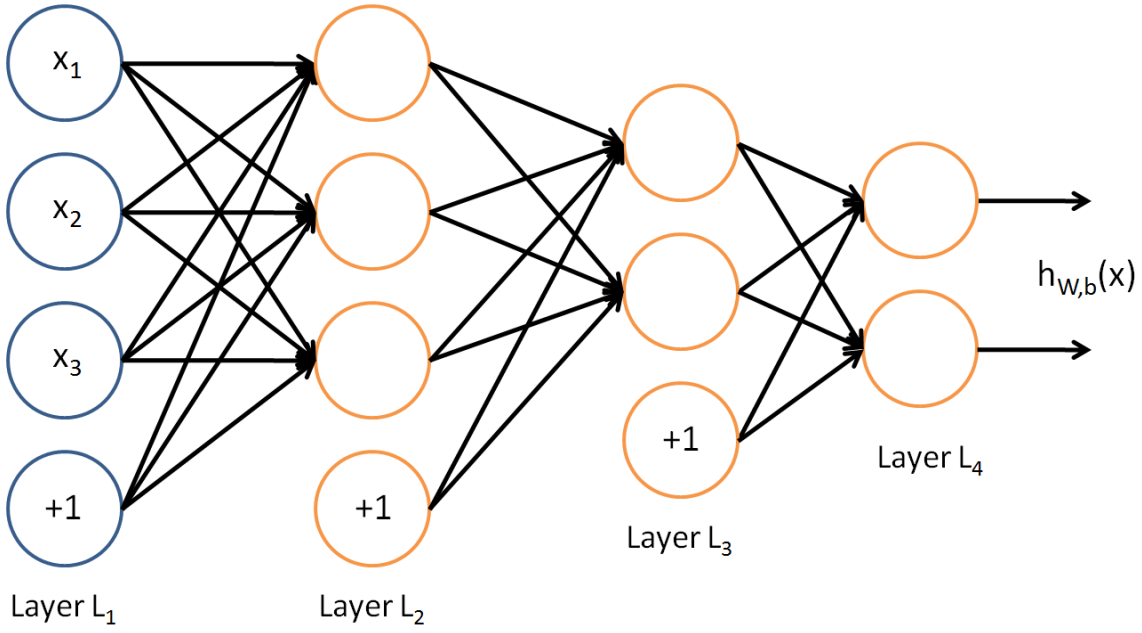
DENOISING AUTOENCODERS

Use Case: Extract robust representation for a NN classifier.



Noisy Input

Latent space representation



DENOISING AUTOENCODERS

Instead of trying to mimic the identity function by minimizing:

$$L\left(x, g\left(f(x)\right)\right)$$

where L is some loss function

A **DAE** instead minimizes:

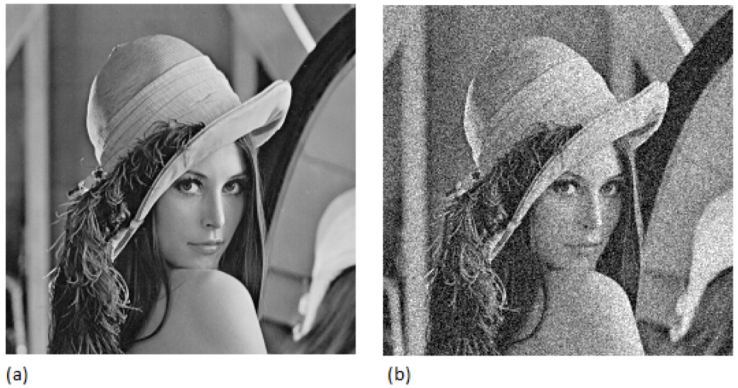
$$L\left(x, g\left(f(\tilde{x})\right)\right)$$

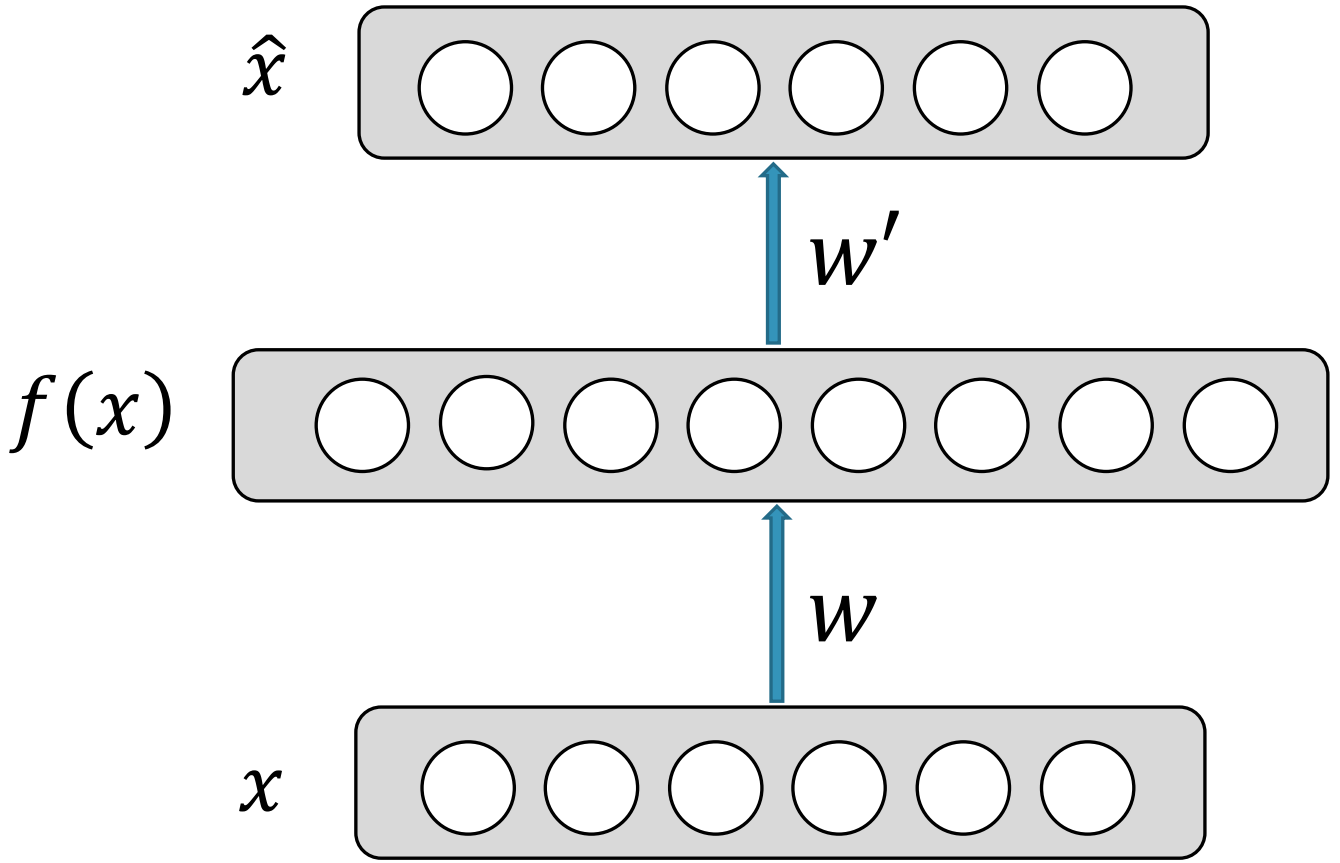
where $\tilde{x}$ is a copy of x that has been corrupted by some form of noise.

DENOISING AUTOENCODERS

Idea: A robust representation against noise:

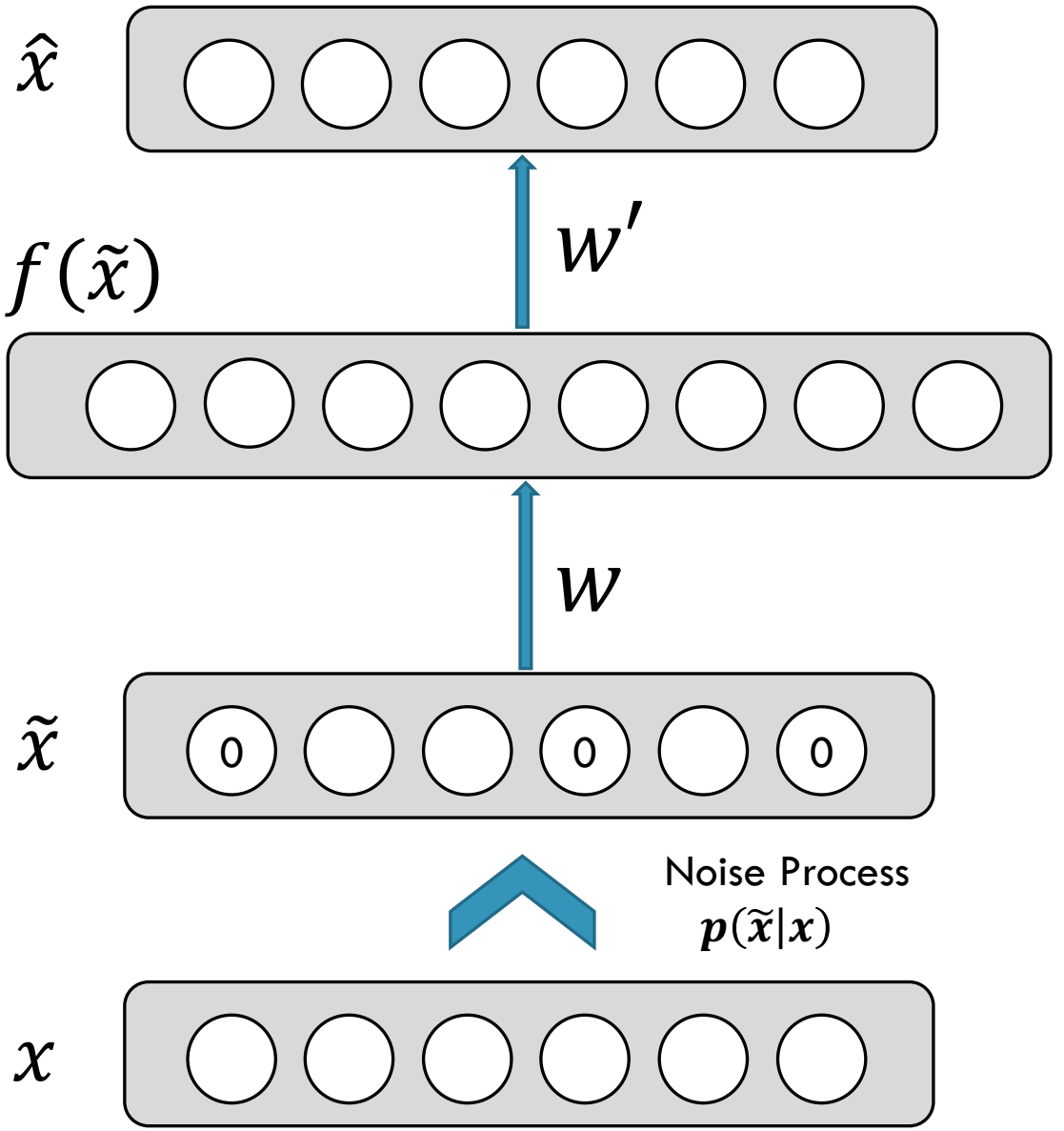
- Random assignment of subset of inputs to 0, with probability v.
- Gaussian additive noise.





- Reconstruction $\hat{x}$ computed from the corrupted input $\tilde{x}$.
- Loss function compares $\hat{x}$ reconstruction with the noiseless x.

- The autoencoder cannot fully trust each feature of x independently so it must learn the correlations of x's features.
- Based on those relations we can predict a more 'not prune to changes' model.
- We are forcing the hidden layer to learn a generalized structure of the data.



DENOISING AUTOENCODERS - PROCESS

Taken some input x

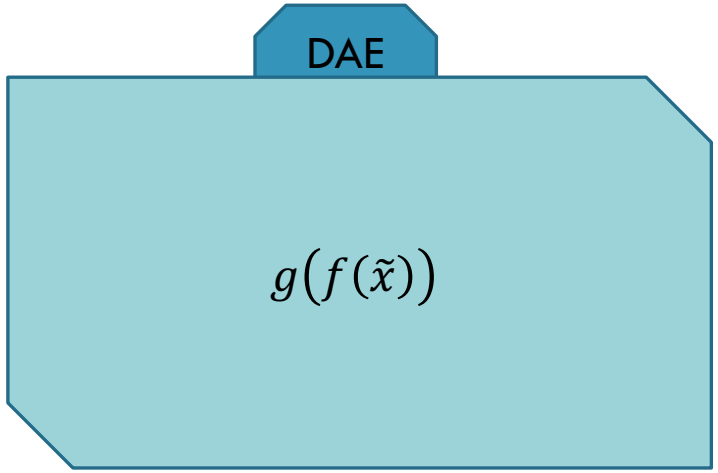
Apply Noise

$\tilde{x}$

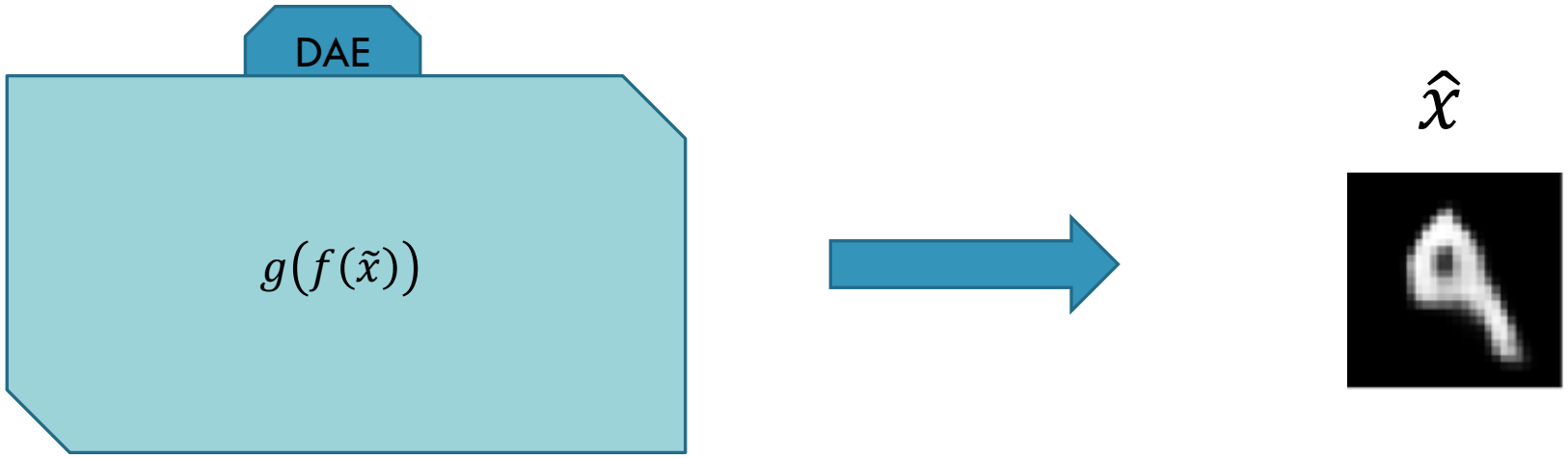
DENOISING AUTOENCODERS - PROCESS

$\tilde{x}$

Encode And Decode

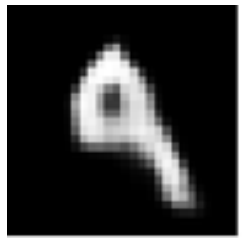


DENOISING AUTOENCODERS - PROCESS



DENOISING AUTOENCODERS - PROCESS

$\hat{x}$



Compare

x

STACKED AE

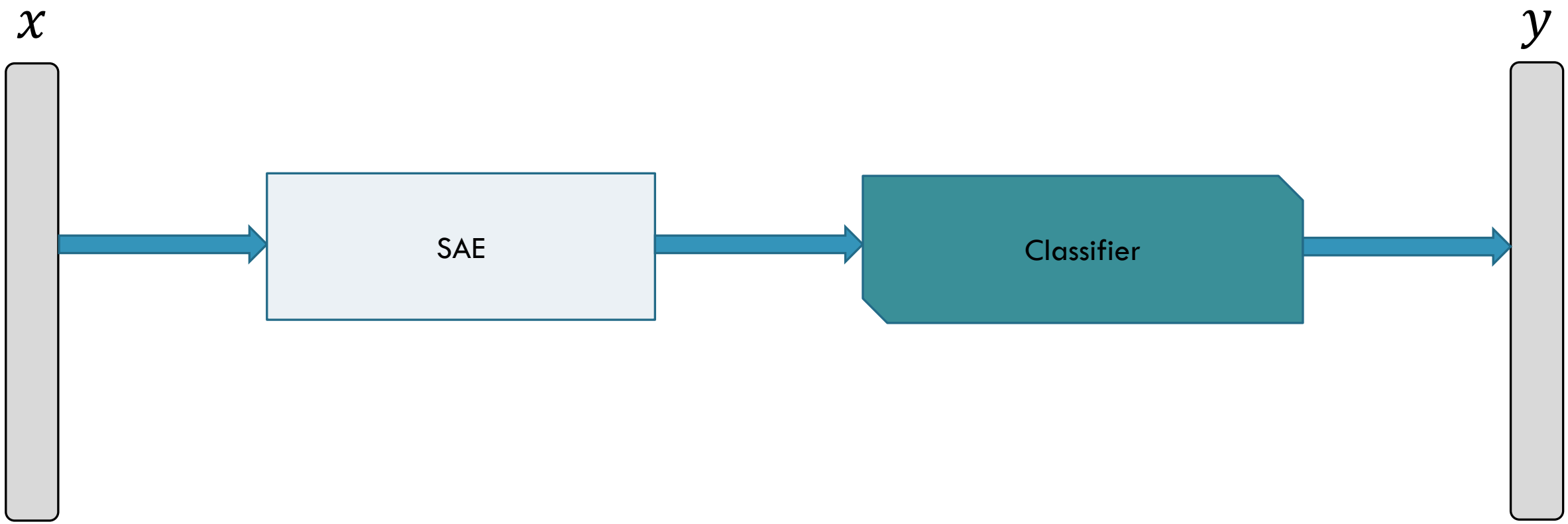
STACKED AE

- Motivation:
- We want to harness the feature extraction quality of a AE for our advantage.
- For example: we can build a deep supervised classifier where its input is the output of a SAE.
- The benefit: our deep model's W are not randomly initialized but are rather "smartly selected"

STACKED AE

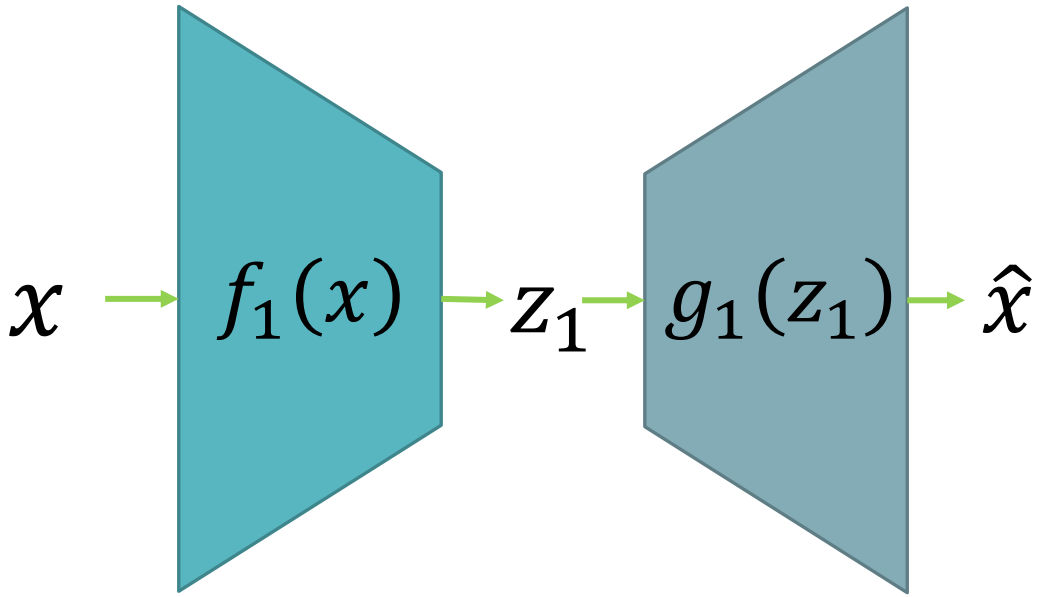
- Building a SAE consists of two phases:
  1) Train each AE layer one after the other.
  2) Connect any classifier (SVM / FC NN layer etc.)

STACKED AE



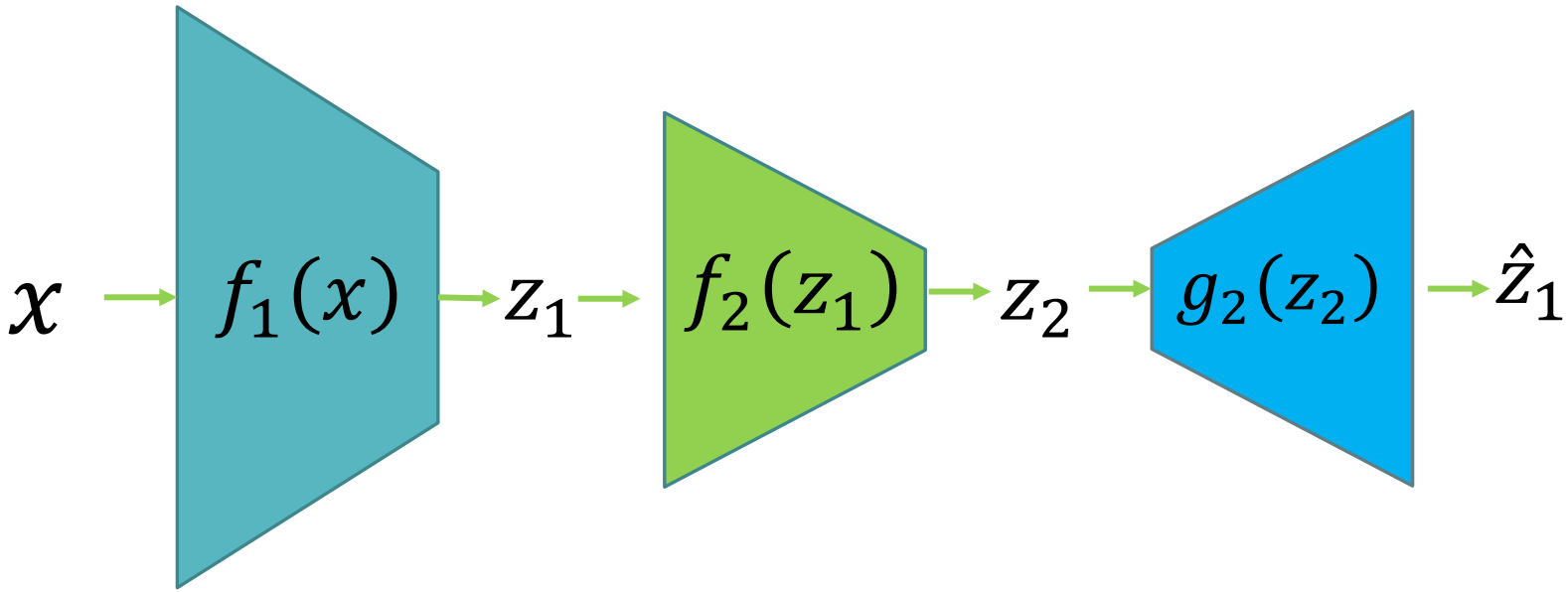
STACKED AE – TRAIN PROCESS

First Layer Training (AE 1)



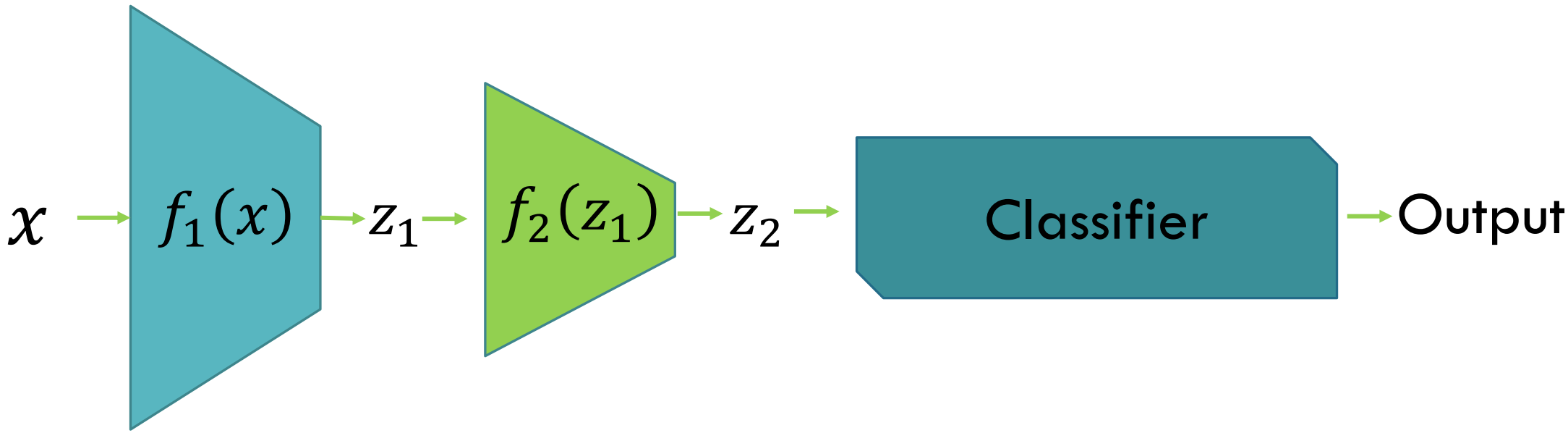
STACKED AE – TRAIN PROCESS

Second Layer Training (AE 2)



STACKED AE – TRAIN PROCESS

Add any classifier



CLASS EXERCISE

bit.ly/ce-ae