

AUTOENCODERS

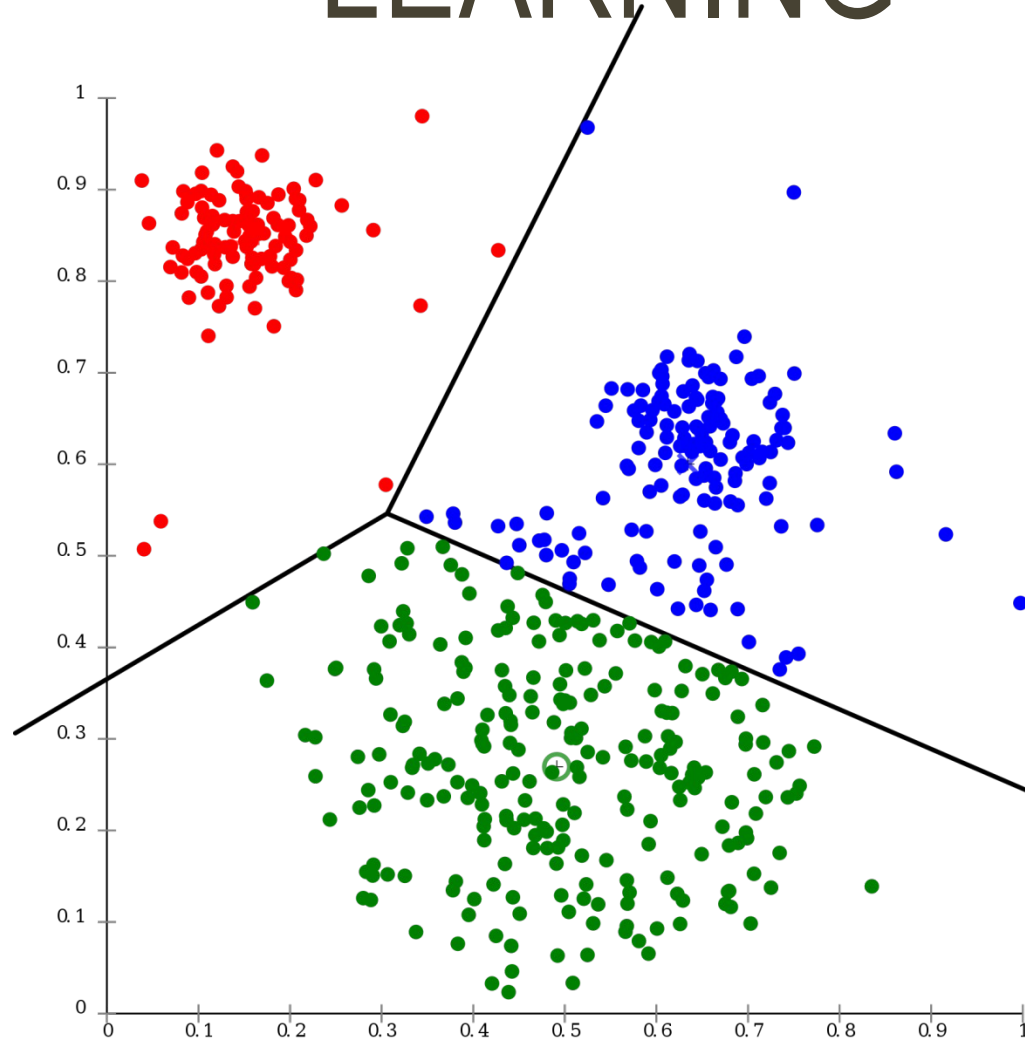
Shangsong Liang
Sun Yat-sen
University

Originally produced by Guy Golan

AGENDA

- Unsupervised Learning (Introduction)
- Autoencoder (AE)
- Convolutional AE
- Regularization: Sparse
- Denoising AE
- Stacked AE
- Contractive AE

INTRODUCTION TO UNSUPERVISED LEARNING



SUPERVISED LEARNING

Supervised Learning

Data: (X,Y)

Goal: Learn a Mapping
Function f where:

$$f(X) = Y$$

Classification



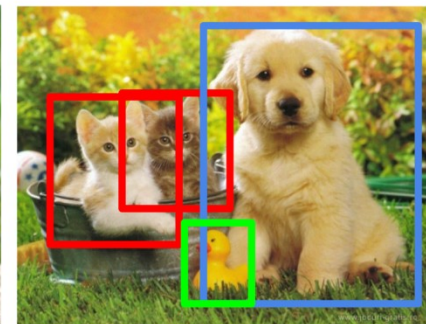
CAT

**Classification
+ Localization**



CAT

Object Detection



CAT, DOG, DUCK

**Instance
Segmentation**



CAT, DOG, DUCK

Single object

Multiple objects

SUPERVISED LEARNING

Examples: Classification.

Decision Trees

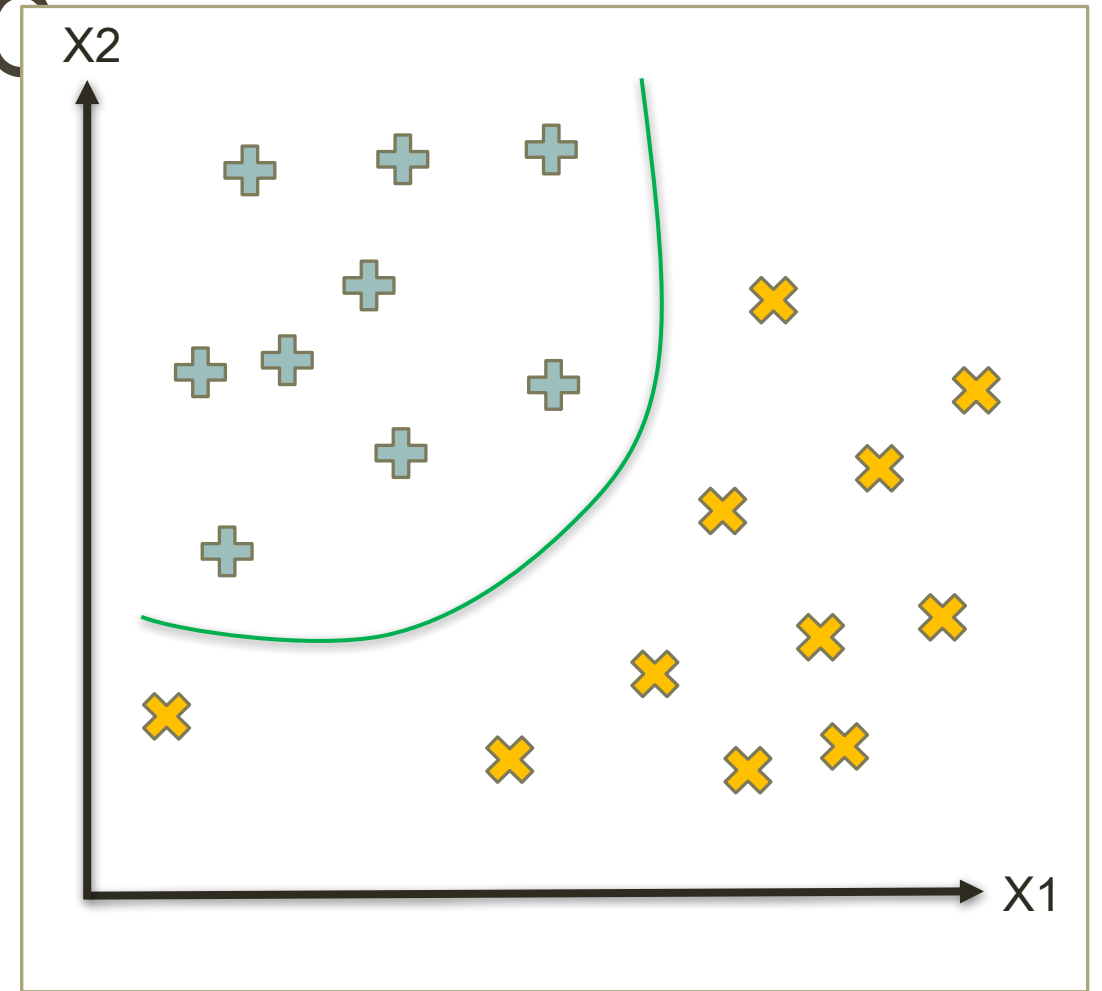
Naïve Bayes

KNN

SVM

Perceptron

Multi Layer
Perceptron



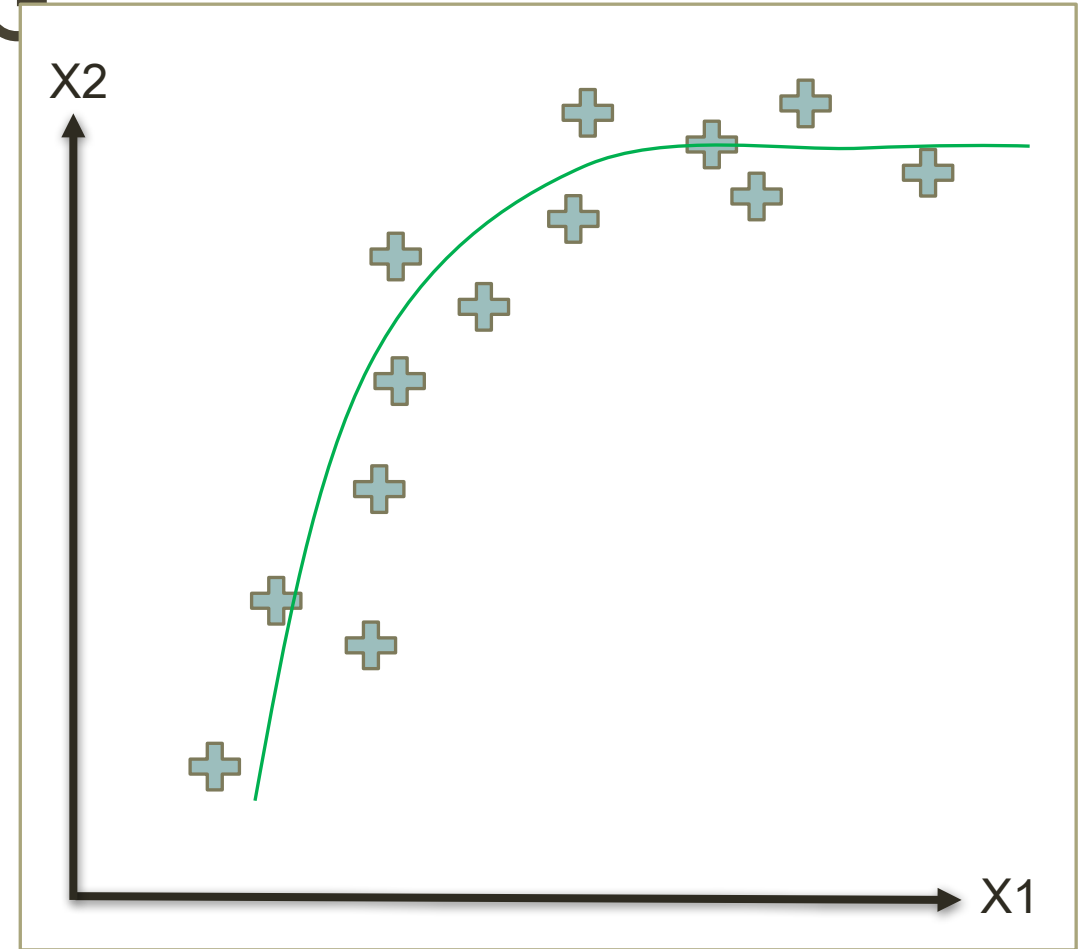
Classification

SUPERVISED LEARNING

Examples: Regression.

Linear Regression

Logistic Regression



Regression

SUPERVISED LEARNING 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 LEARNING VS UNSUPERVISED LEARNING

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

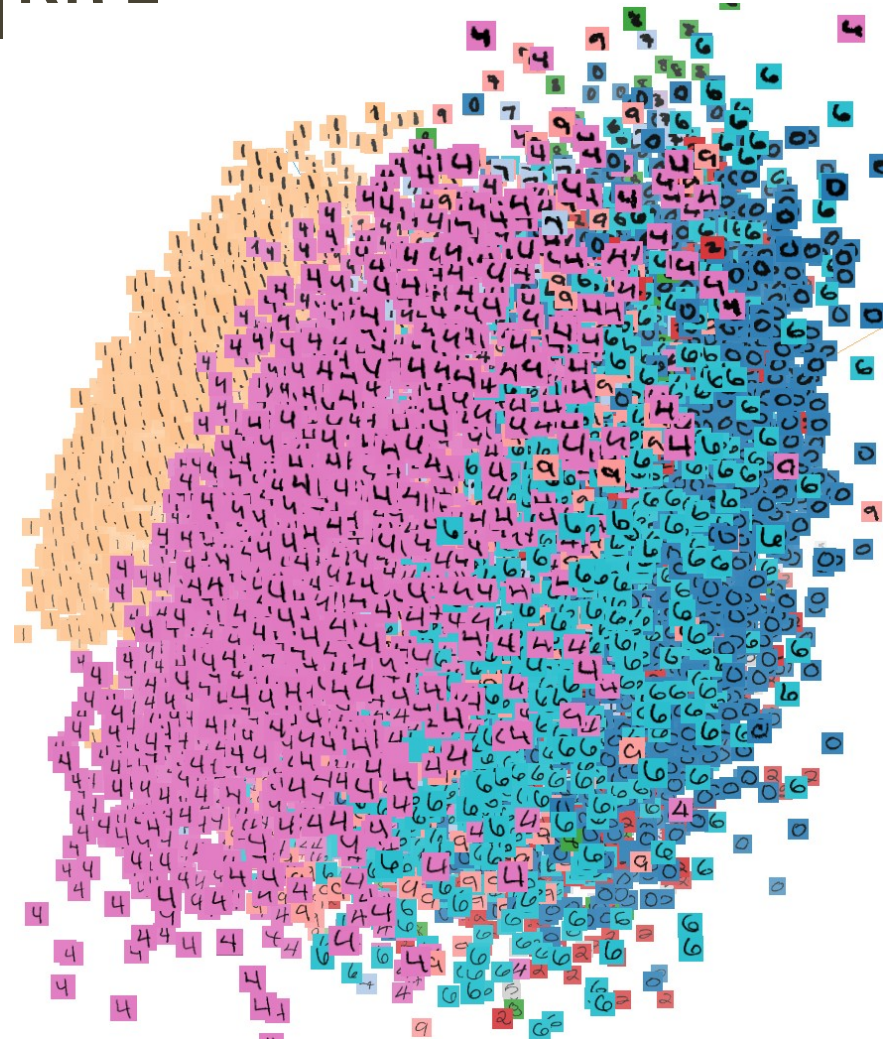
Lets talk about **Unsupervised Learning**

UNSUPERVISED LEARNING

Unsupervised Learning

Data: X (no labels!)

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

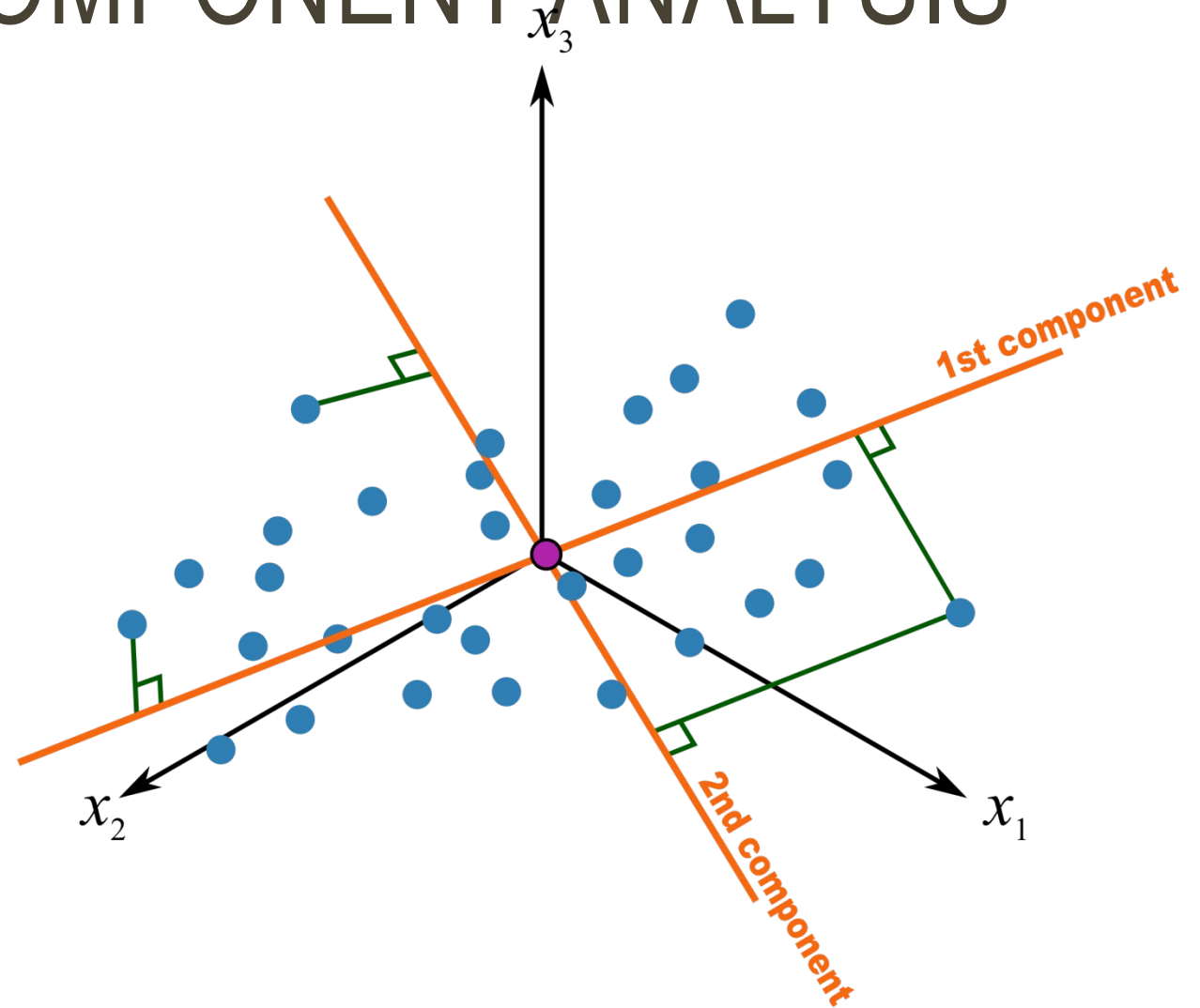


UNSUPERVISED LEARNING

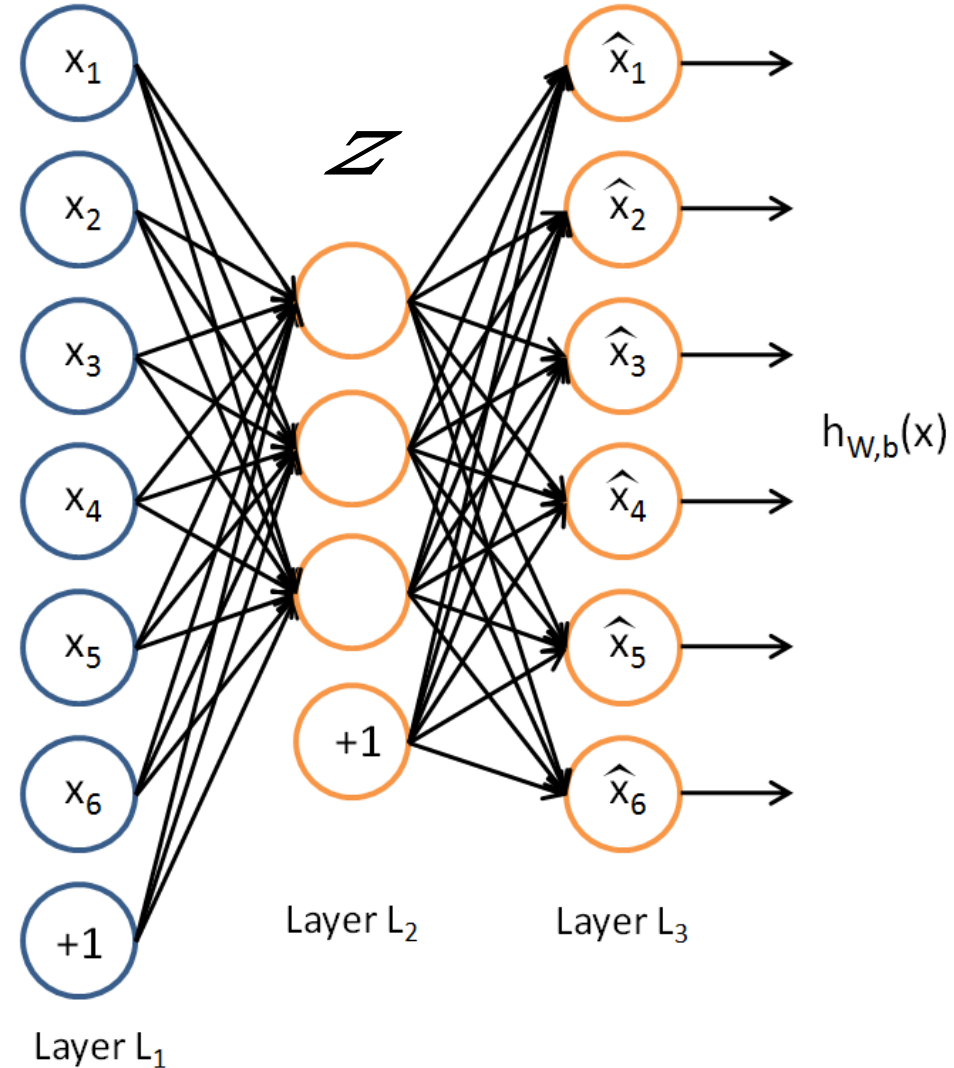
Examples: Clustering, **Compression, Feature & Representation learning, Dimensionality reduction**, Generative models ,etc

PCA – PRINCIPAL COMPONENT ANALYSIS

- Statistical approach for data compression and visualization
- Invented by Karl Pearson in 1901
- Weakness: linear components only.

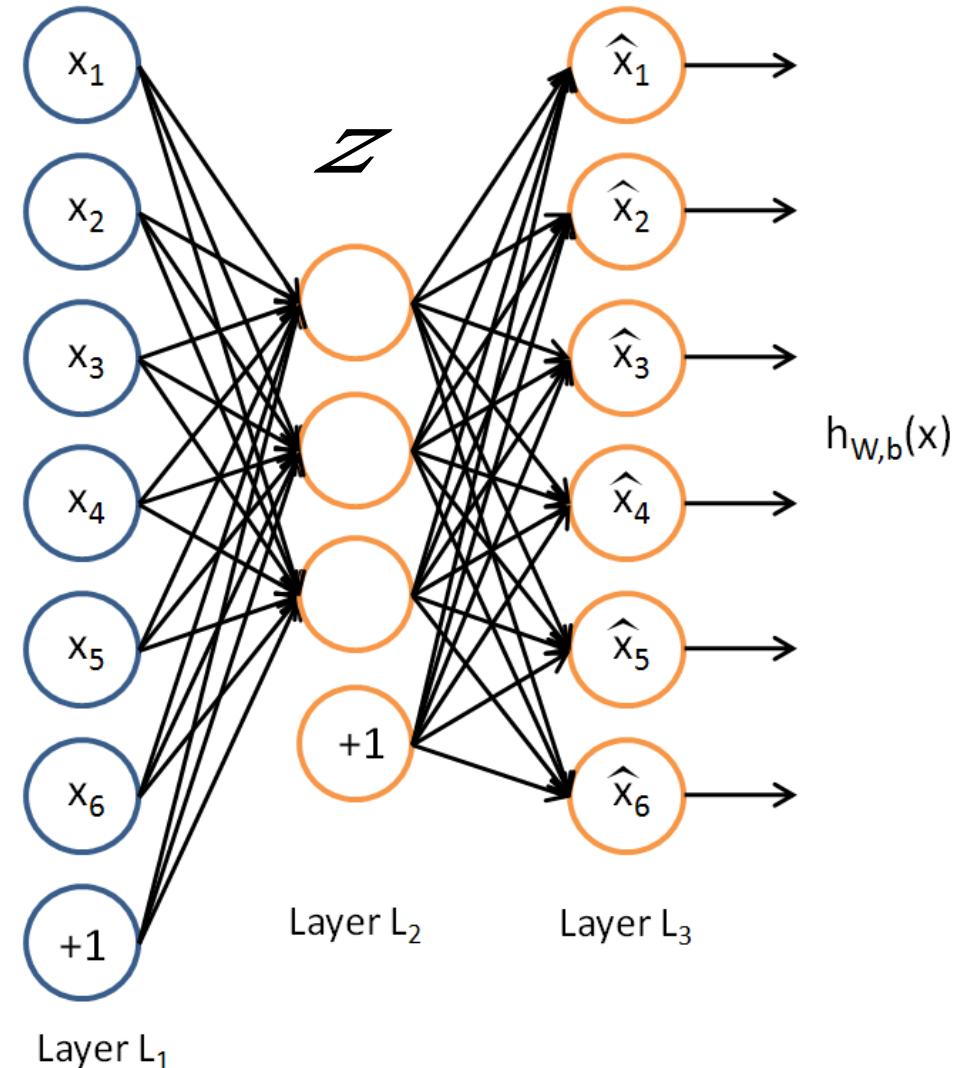


TRADITIONAL AUTOENCODER



TRADITIONAL AUTOENCODER

- Unlike the **PCA** now we can use activation functions to achieve non-linearity.
- It has been shown that an AE without activation functions achieves the **PCA** capacity.

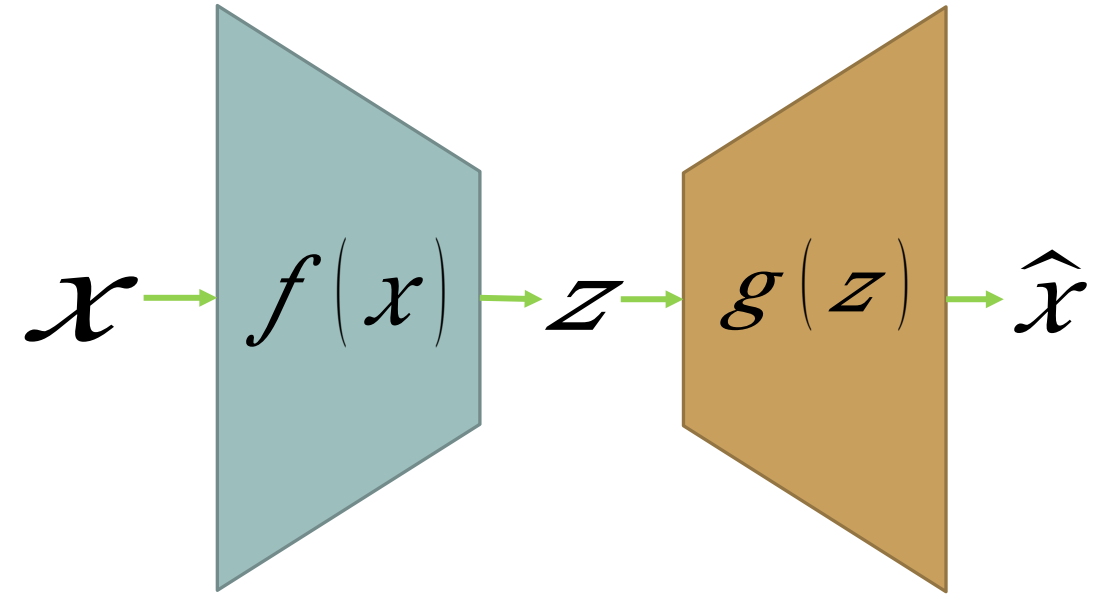


USES

- The autoencoder idea was a part of NN history for decades (LeCun et al, 1987).
 - Traditionally an autoencoder is used for dimensionality reduction and feature learning.
 - Recently, the connection between autoencoders and latent space modeling has brought autoencoders to the front of generative modeling.
- **Not used for compression.**
 - Data specific compression.
 - Lossy.

SIMPLE IDEA

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



and

s.t


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

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

(is '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!)

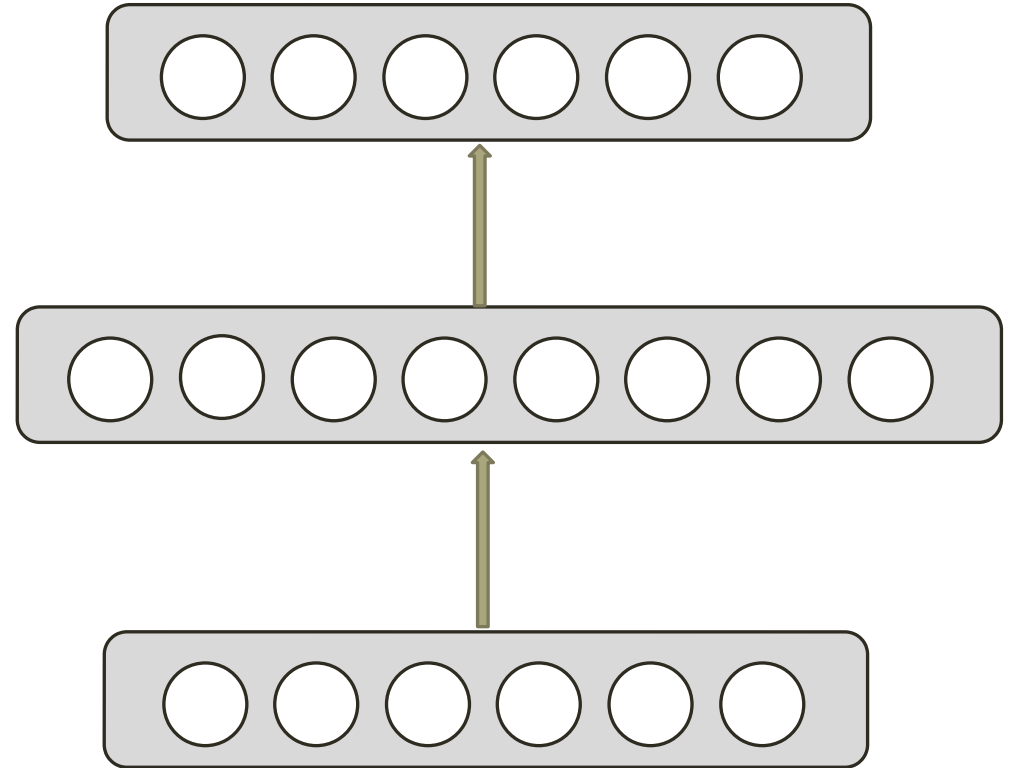
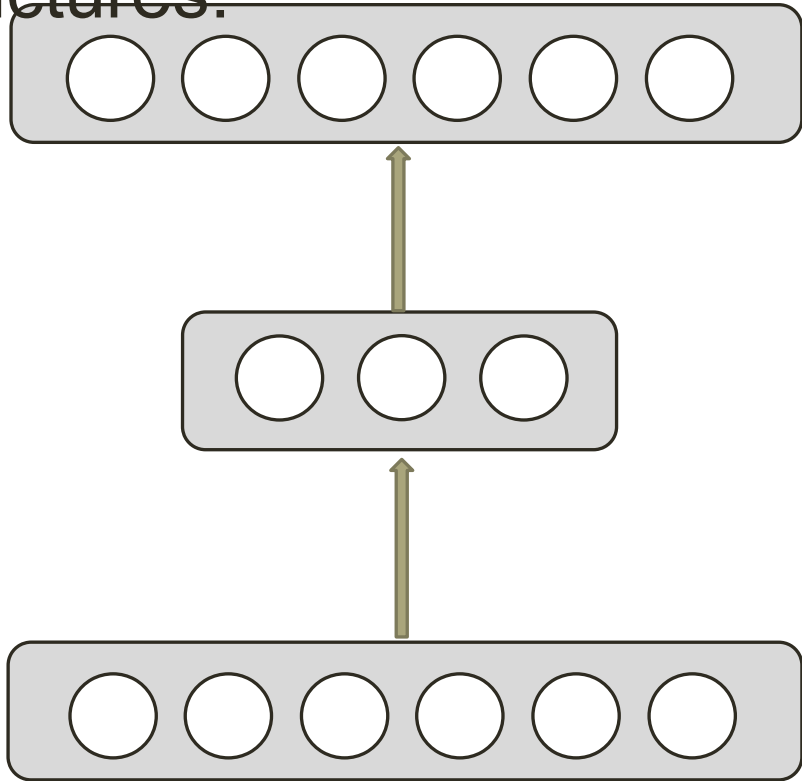
TRAINING THE AE

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

- Traditionally with squared error loss function
- If our input is interpreted as bit vectors or vectors of bit probabilities the cross entropy can be used

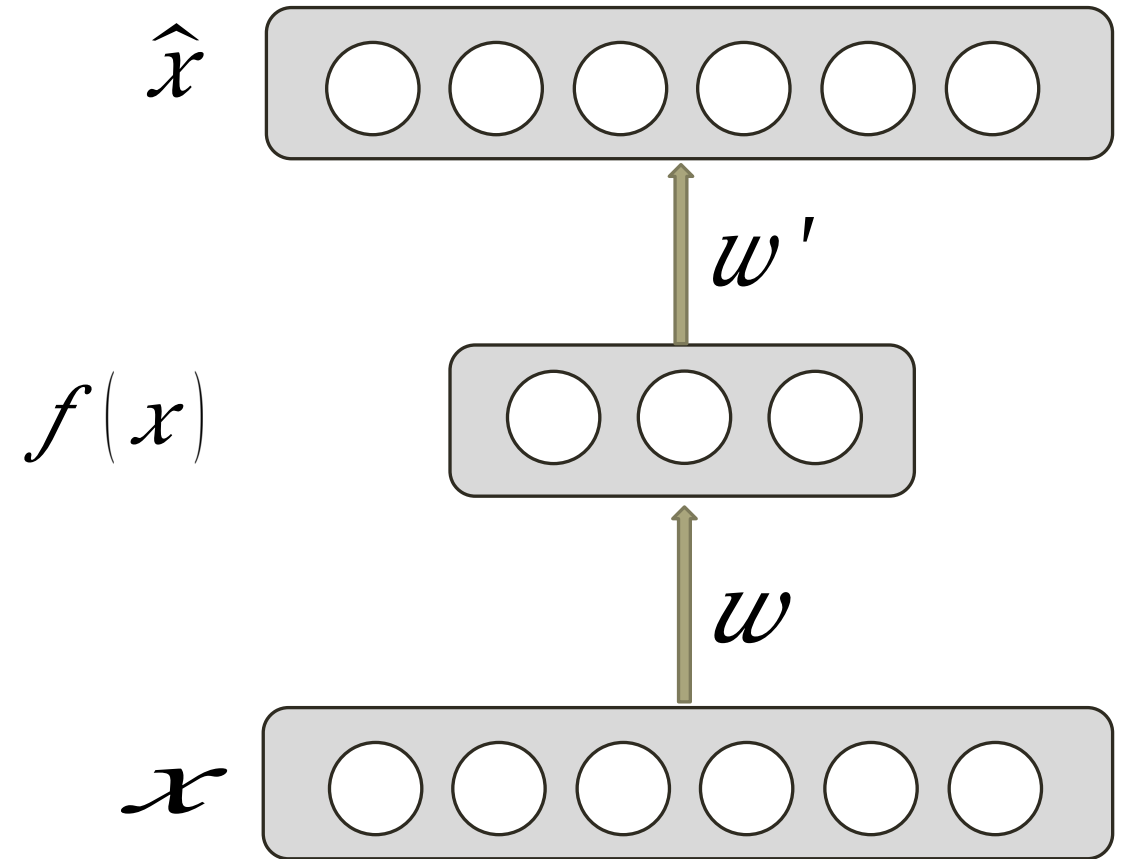
UNDERCOMPLETE AE VS OVERCOMPLETE AE

We distinguish between two types of AE structures:



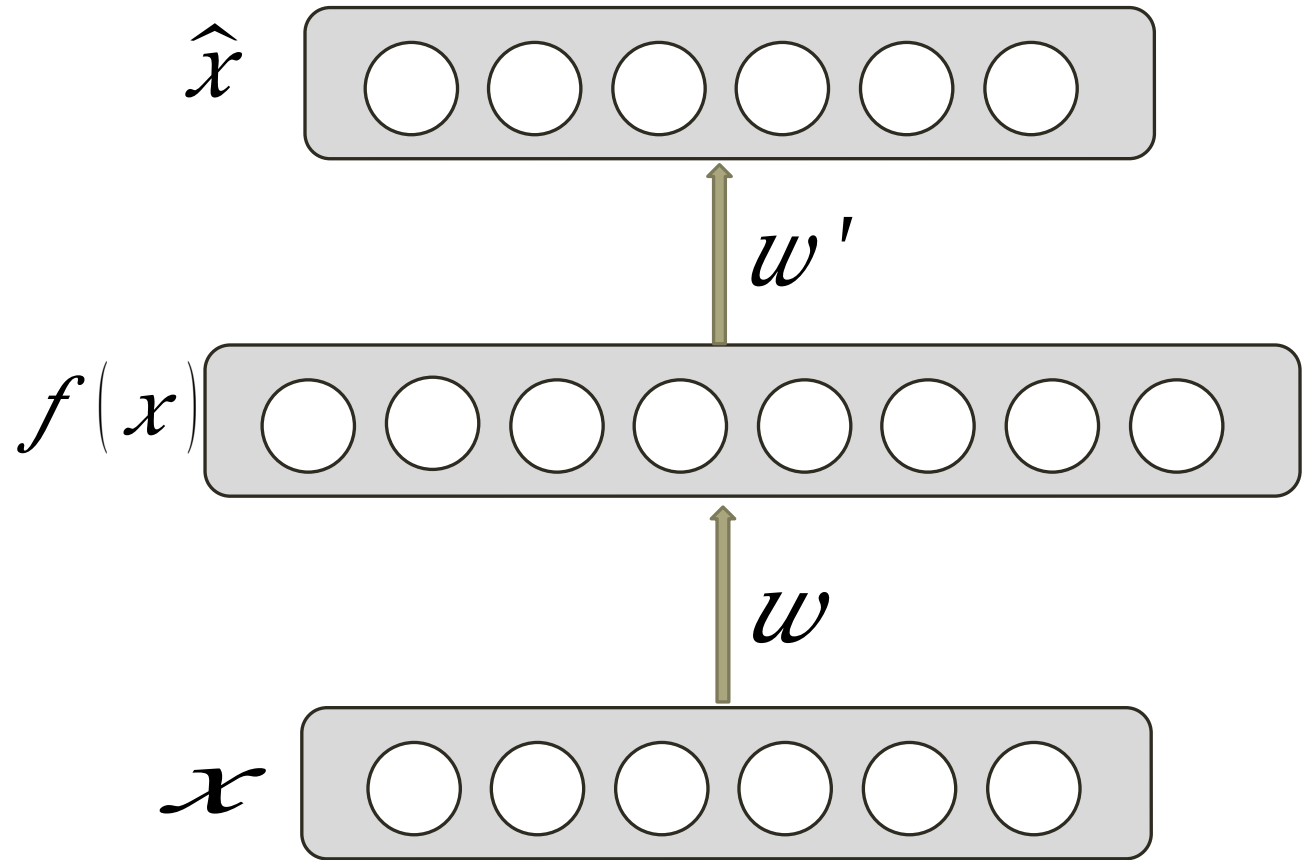
UNDERCOMPLETE AE

- Hidden layer is **Undercomplete** if smaller than the input layer
 - ❑ Compresses the input
 - ❑ Compresses well only for the training dist.
- Hidden nodes will be
 - ❑ Good features for the training distribution.
 - ❑ Bad for other types on input



OVERCOMPLETE AE

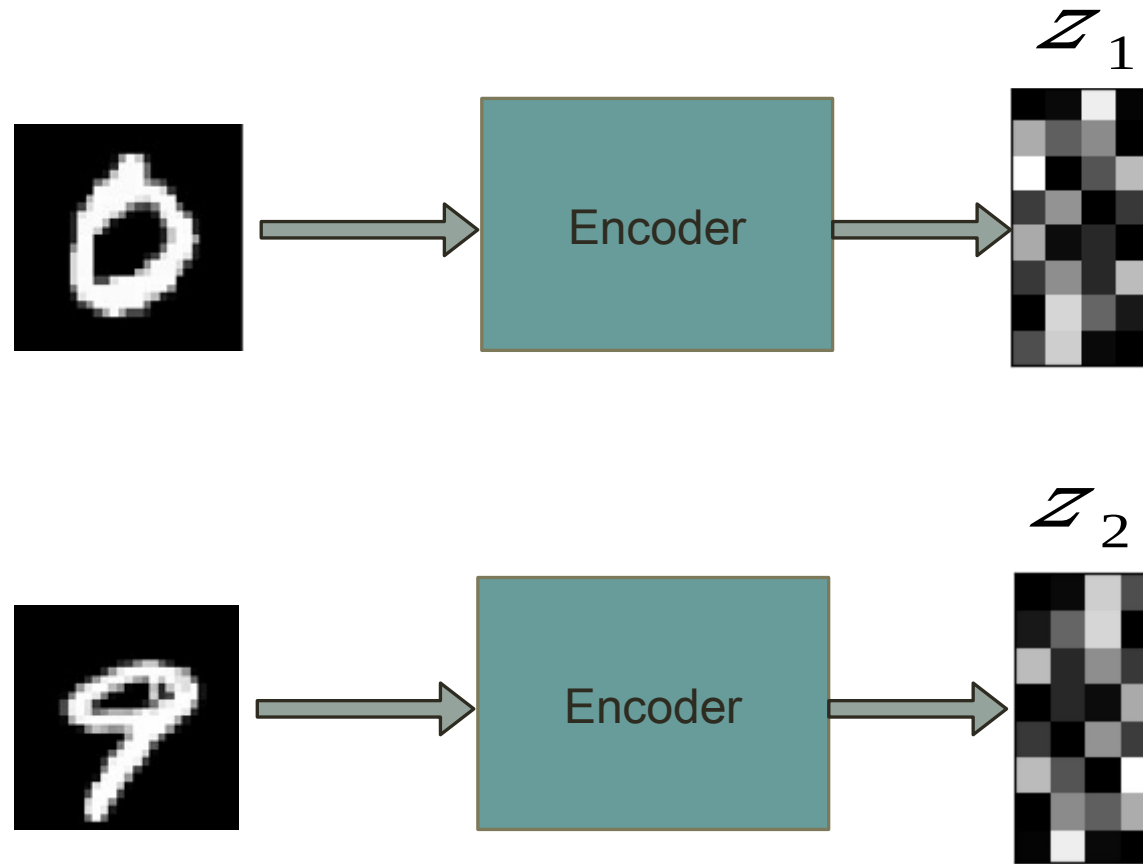
- Hidden layer is **Overcomplete** if greater than the input layer
 - ❑ No compression in hidden layer.
 - ❑ Each hidden unit could copy a different input component.
- No guarantee that the hidden units will extract meaningful structure.
- Adding dimensions is good for training a linear classifier (XOR case example).
- A higher dimension code helps model a more complex distribution



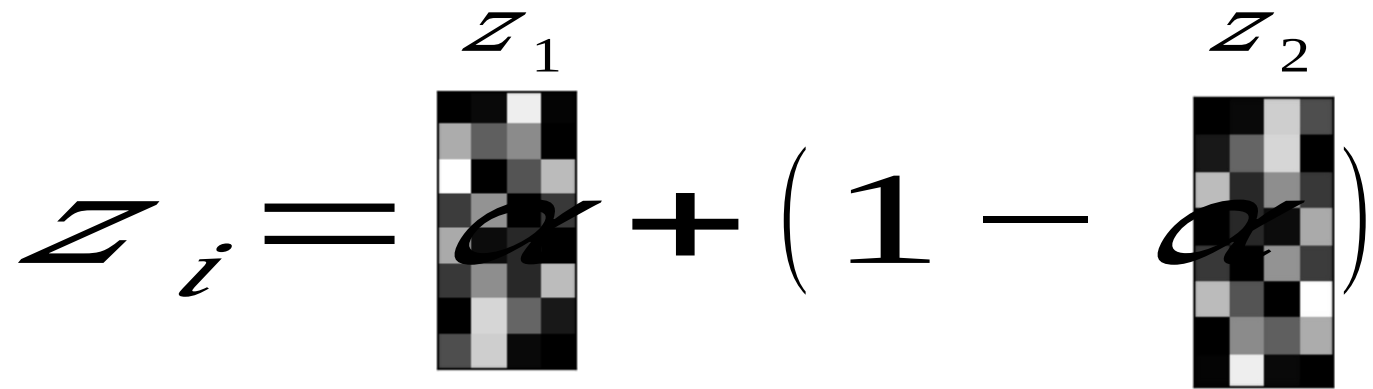
DEEP AUTOENCODER EXAMPLE

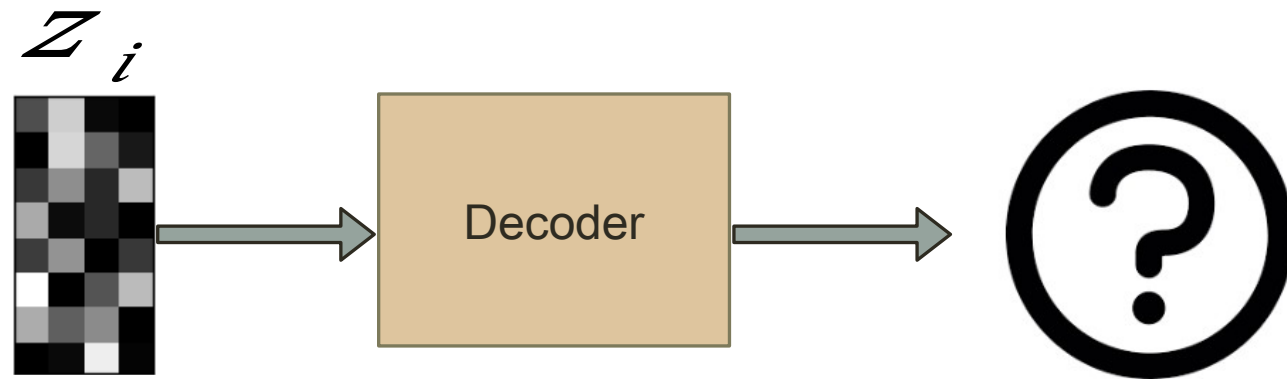
<https://cs.stanford.edu/people/karpathy/convnetjs/demo/autoencoder.html> - By
Andrej Karpathy


SIMPLE LATENT SPACE INTERPOLATION - KERAS



SIMPLE LATENT SPACE INTERPOLATION - KERAS

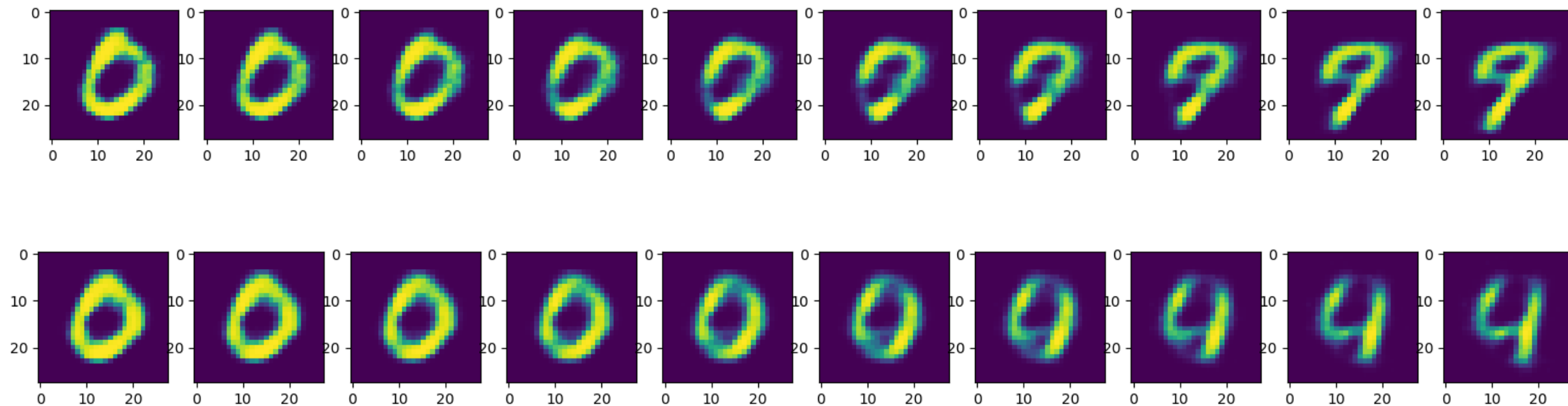
$$Z_i = \cancel{\alpha} Z_1 + (1 - \cancel{\alpha}) Z_2$$




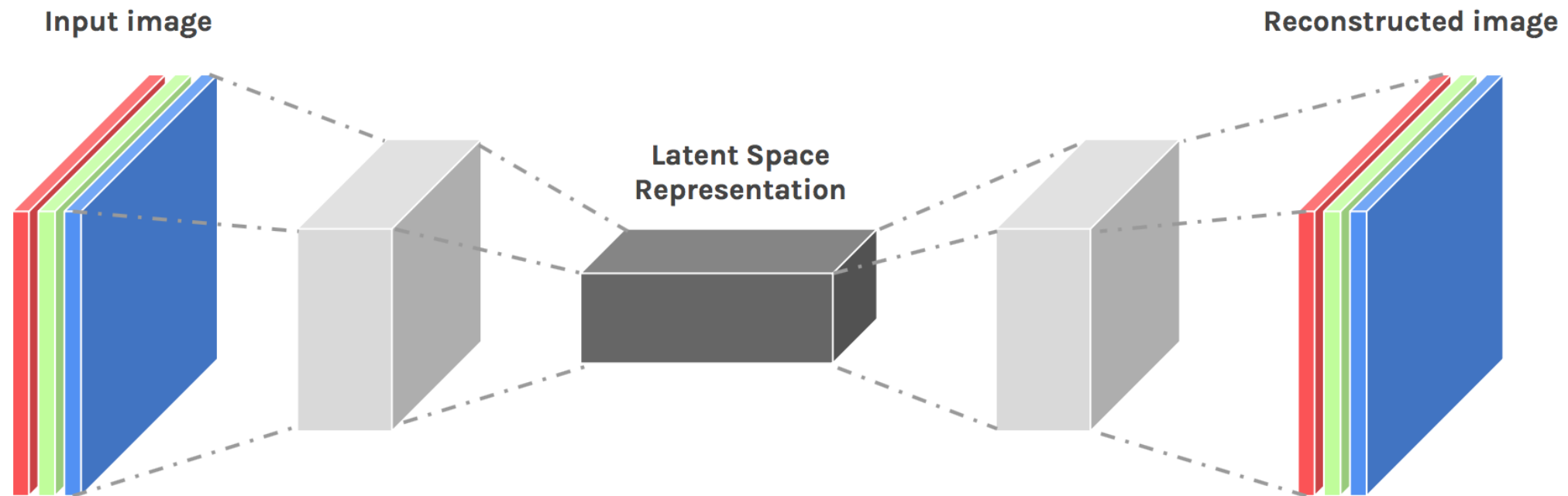


SIMPLE LATENT SPACE INTERPOLATION – KERAS CODE EXAMPLE

SIMPLE LATENT SPACE – INTERPOLATION - KERAS

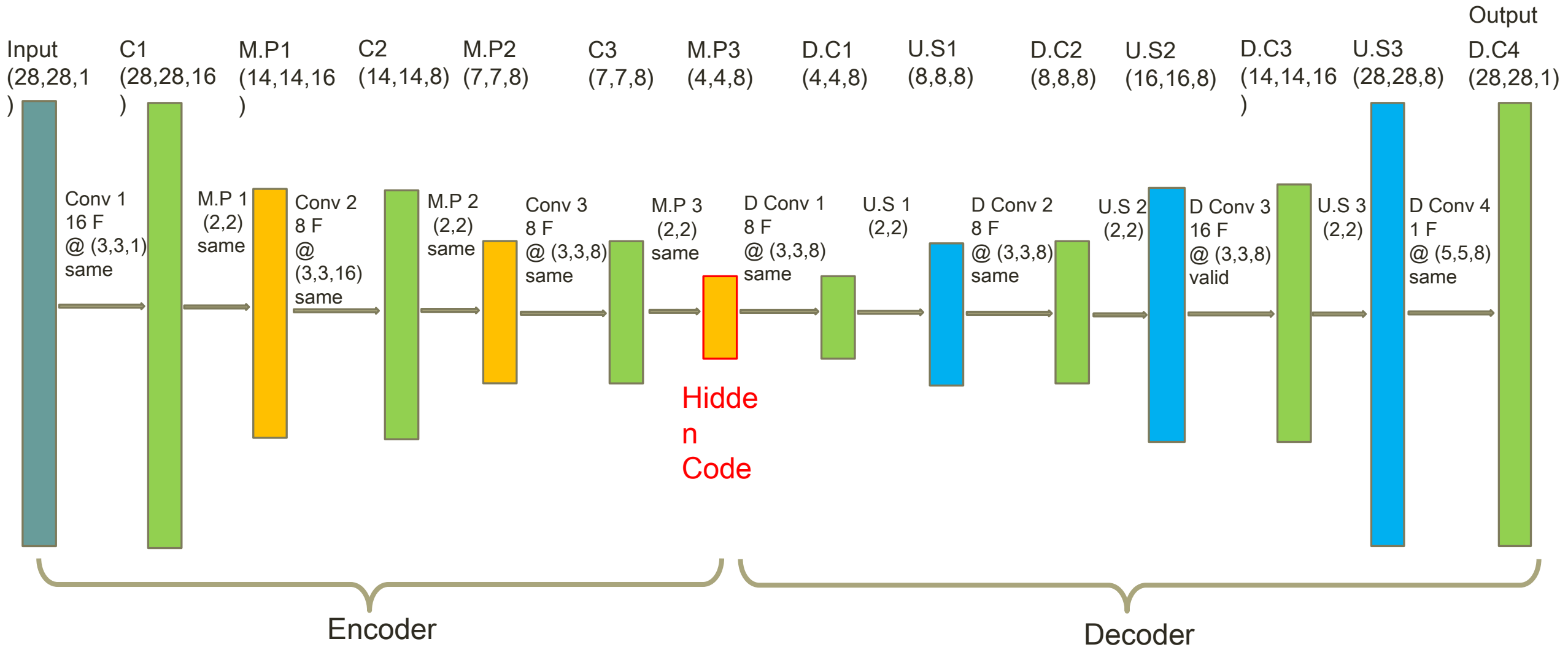


CONVOLUTIONAL AE



- * Input values are normalized
- * All of the conv layers activation functions are relu except for the last conv which is sigmoid

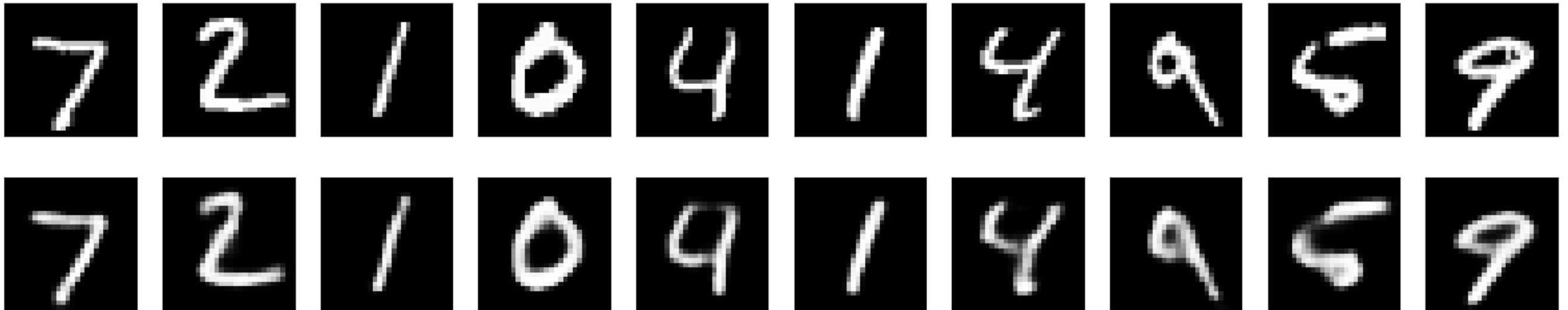
CONVOLUTIONAL AE



CONVOLUTIONAL AE – KERAS EXAMPLE

CONVOLUTIONAL AE – KERAS EXAMPLE RESULTS

- 50 epochs.
- 88% accuracy on validation set.



REGULARIZATION

Motivation:

- We would like to learn meaningful features **without** altering the code's dimensions (Overcomplete or Undercomplete).

The solution: imposing other constraints on the network.

SPARSELY REGULATED AUTOENCODERS

A bad example:

Activation
Maps



SPARSELY REGULATED AUTOENCODERS

- We want our learned features to be as **sparse** as possible.
- With sparse features we can generalize better.

$$\begin{aligned} \boxed{7} &= 1 * \boxed{\text{feature 1}} + 1 * \boxed{\text{feature 2}} + 1 * \boxed{\text{feature 3}} + 1 * \boxed{\text{feature 4}} + 1 * \boxed{\text{feature 5}} \\ &+ 1 * \boxed{\text{feature 6}} + 1 * \boxed{\text{feature 7}} + 0.8 * \boxed{\text{feature 8}} + 0.8 * \boxed{\text{feature 9}} \end{aligned}$$

SPARSELY REGULATED AUTOENCODERS

Recall:

a_j is defined to be the activation of the j th hidden unit (bottleneck) of the autoencoder.

Let a_j be the activation of this specific node on a given input x .

SPARSELY REGULATED AUTOENCODERS

Further let,

be the average activation of hidden unit (over the training set).

Thus we would like to force the constraint:

where α is a “sparsity parameter”, typically small. In other words, we want the average activation of each neuron to be close to α .

SPARSELY REGULATED AUTOENCODERS

- We need to penalize for deviating from .
- Many choices of the penalty term will give reasonable results.

- For example:

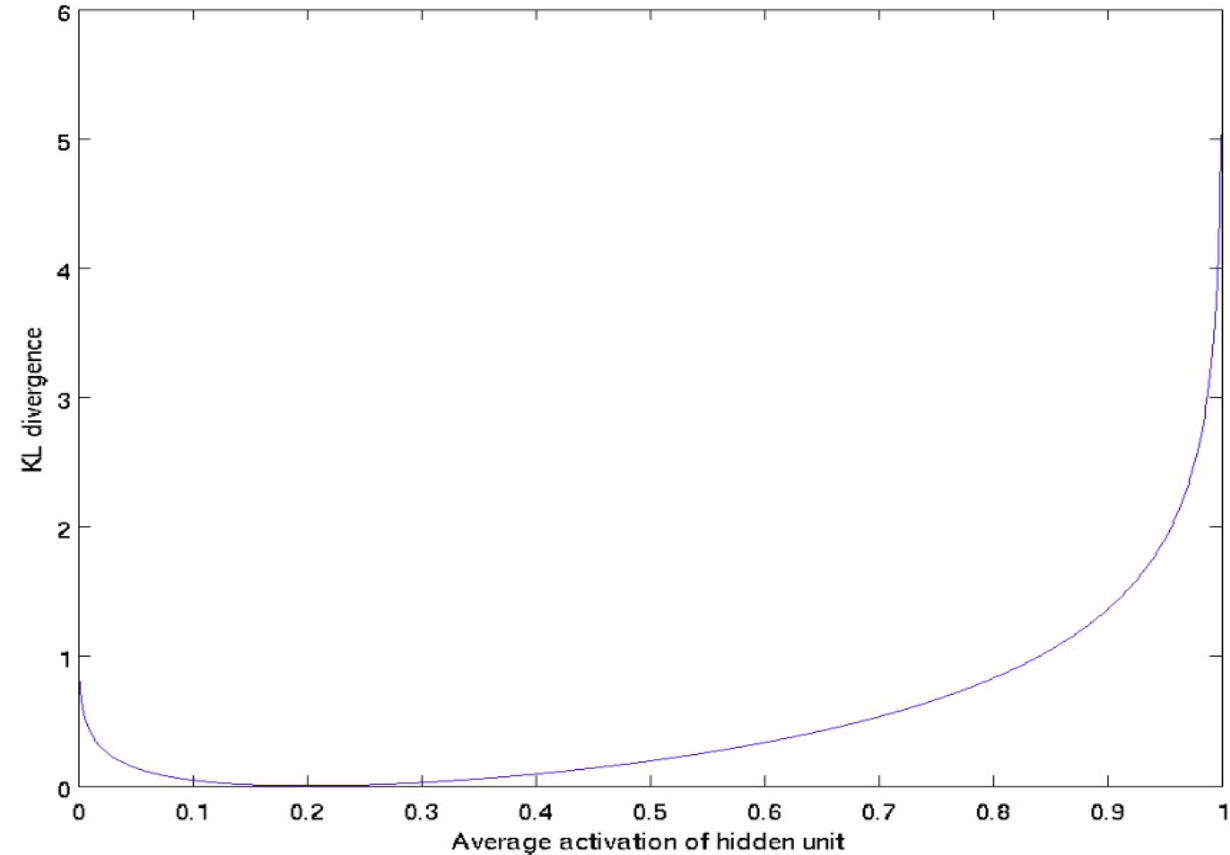
where is a Kullback-Leibler divergence function.

SPARSELY REGULATED AUTOENCODERS

- A reminder:
 - KL is a standard function for measuring how different two distributions are, which has the properties:

= 0 if

otherwise it is increased monotonically.



$$\rho = 0.2$$

SPARSELY REGULATED AUTOENCODERS

- Our overall cost functions is now:

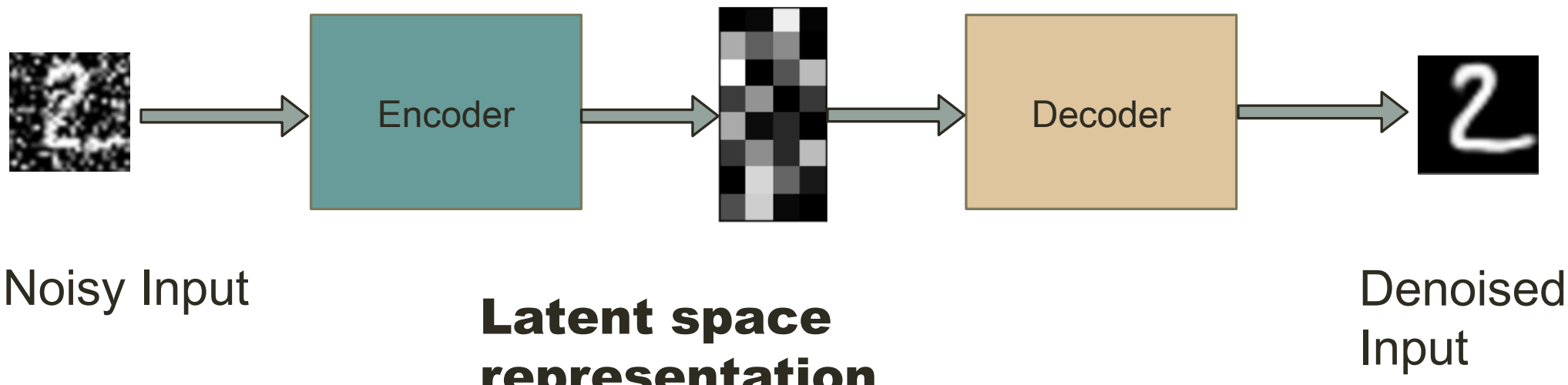
*Note: We need to know before hand,
so we have to compute a forward pass on all the
training set.

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.

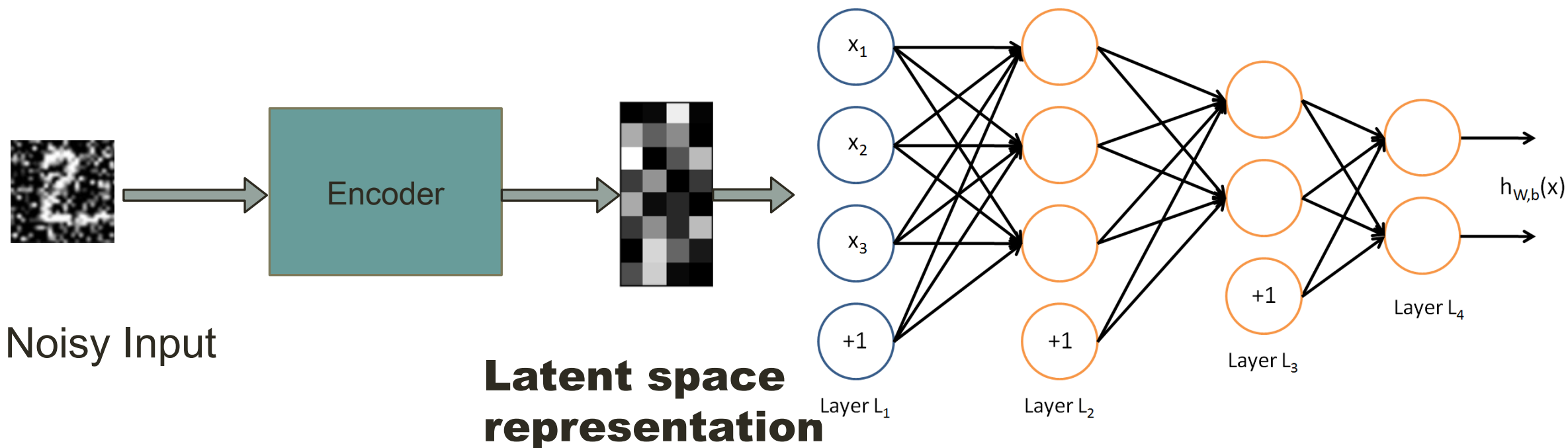
A more robust model



DENOISING AUTOENCODERS

Use Case:

- Extract robust representation for a NN classifier.



DENOISING AUTOENCODERS

Instead of trying to mimic the identity function by minimizing:

where L is some loss function

A **DAE** instead minimizes:

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 .

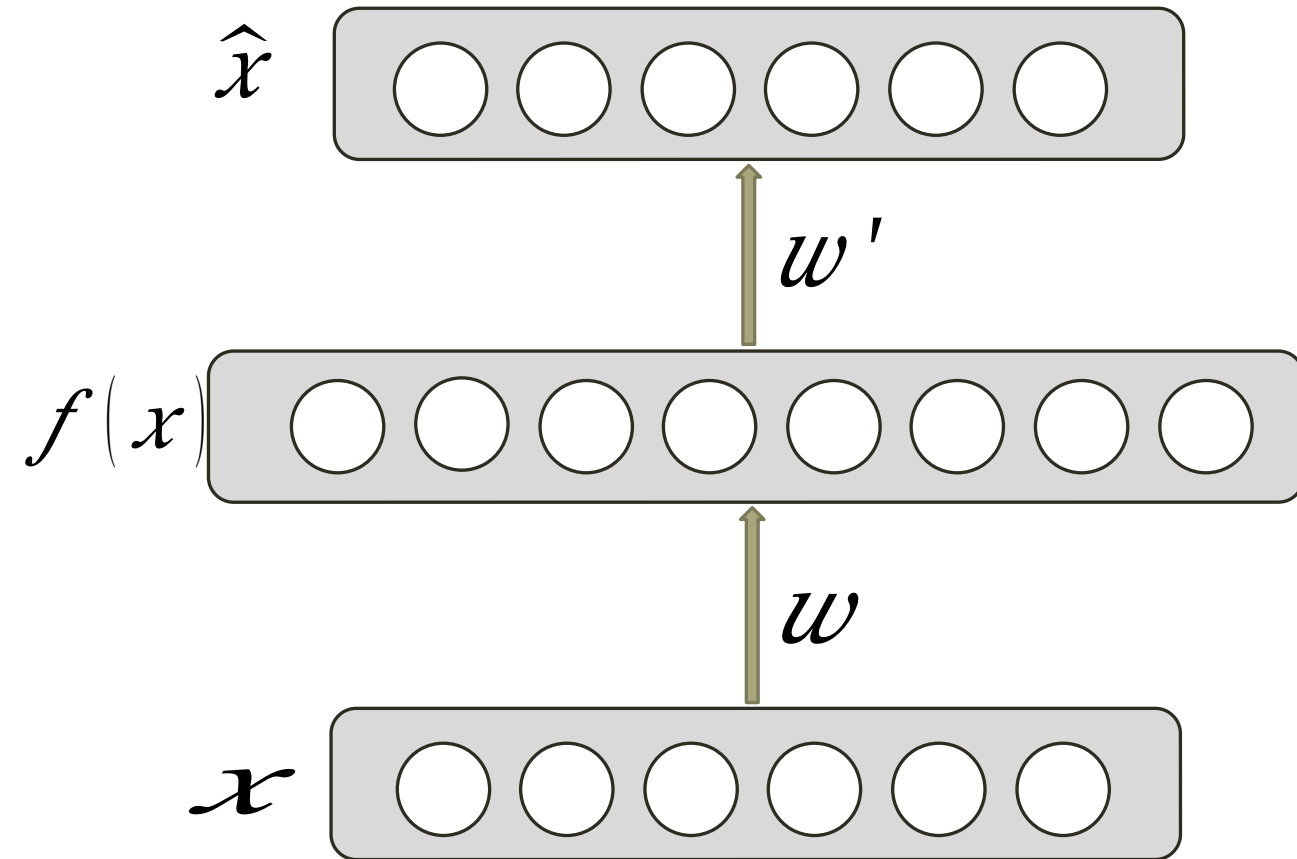


(a)



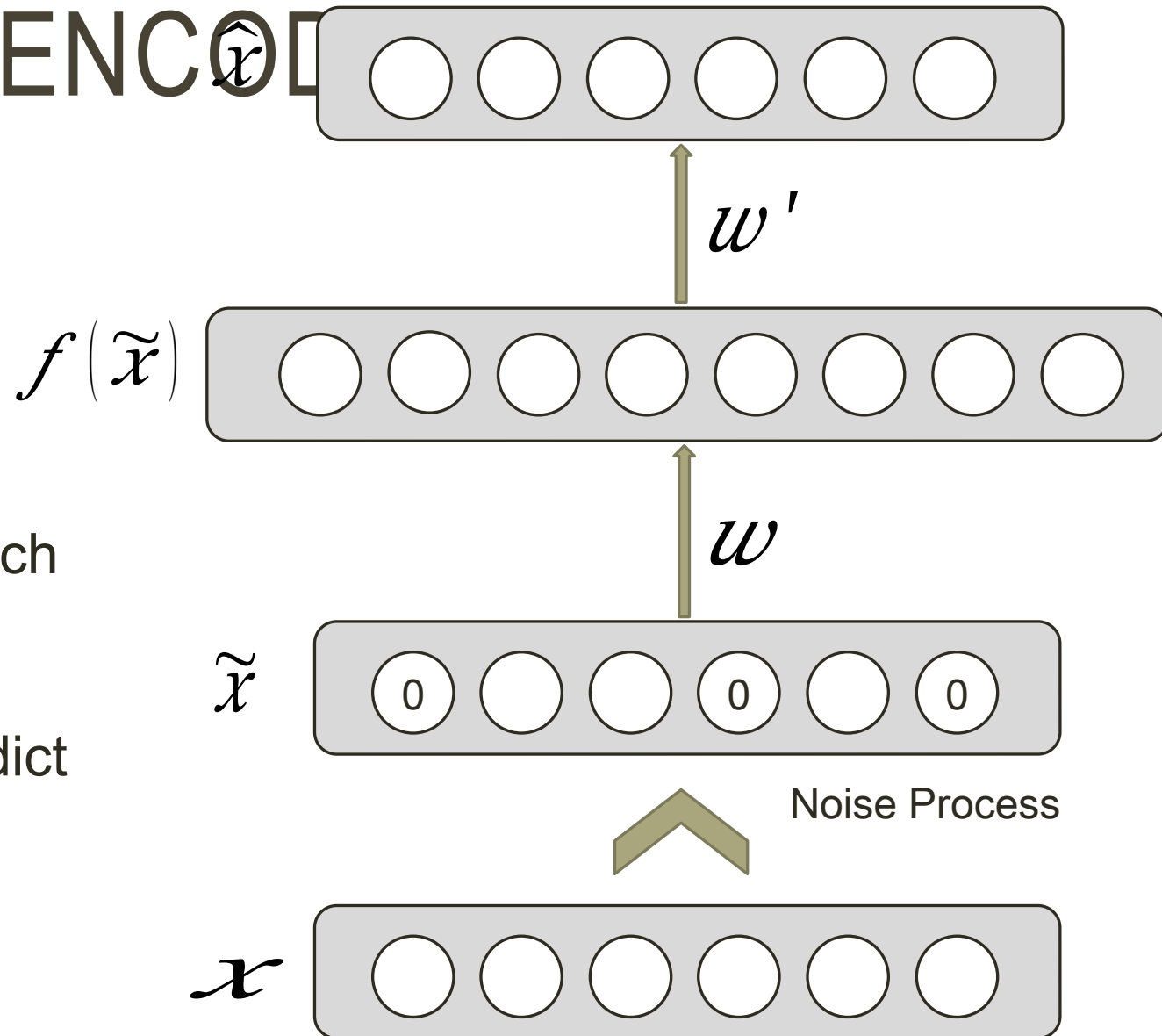
(b)

ise.



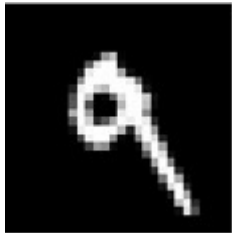
DENOISING AUTOENCODER

- Reconstruction computed from the corrupted input .
- Loss function compares reconstruction with the noiseless .
- ❖ The autoencoder cannot fully trust each feature of independently so it must learn the correlations of 's features.
- ❖ Based on those relations we can predict a more 'not prone to changes' model.
- We are forcing the hidden layer to learn a generalized structure of the



DENOISING AUTOENCODERS - PROCESS

Taken some input



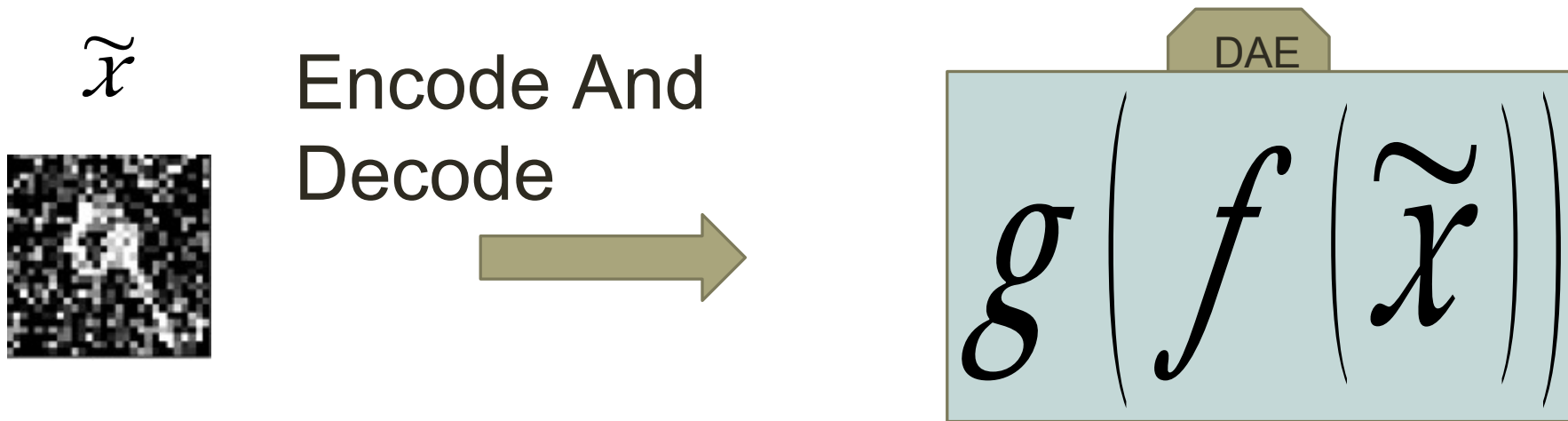
Apply
Noise



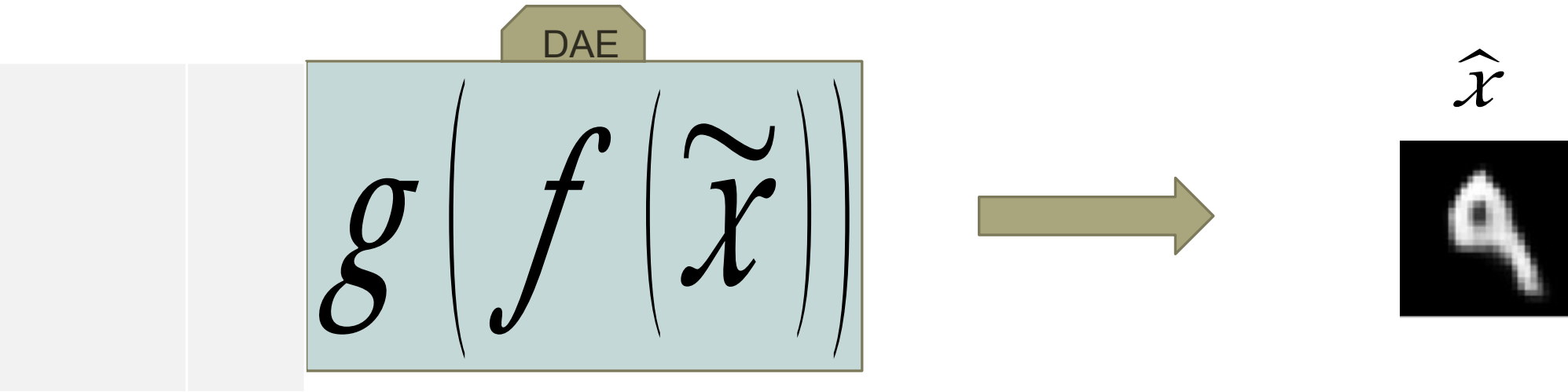
\tilde{x}



DENOISING AUTOENCODERS - PROCESS

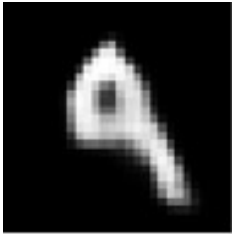


DENOISING AUTOENCODERS - PROCESS

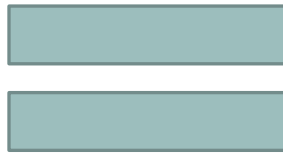


DENOISING AUTOENCODERS - PROCESS

\hat{x}



Compare

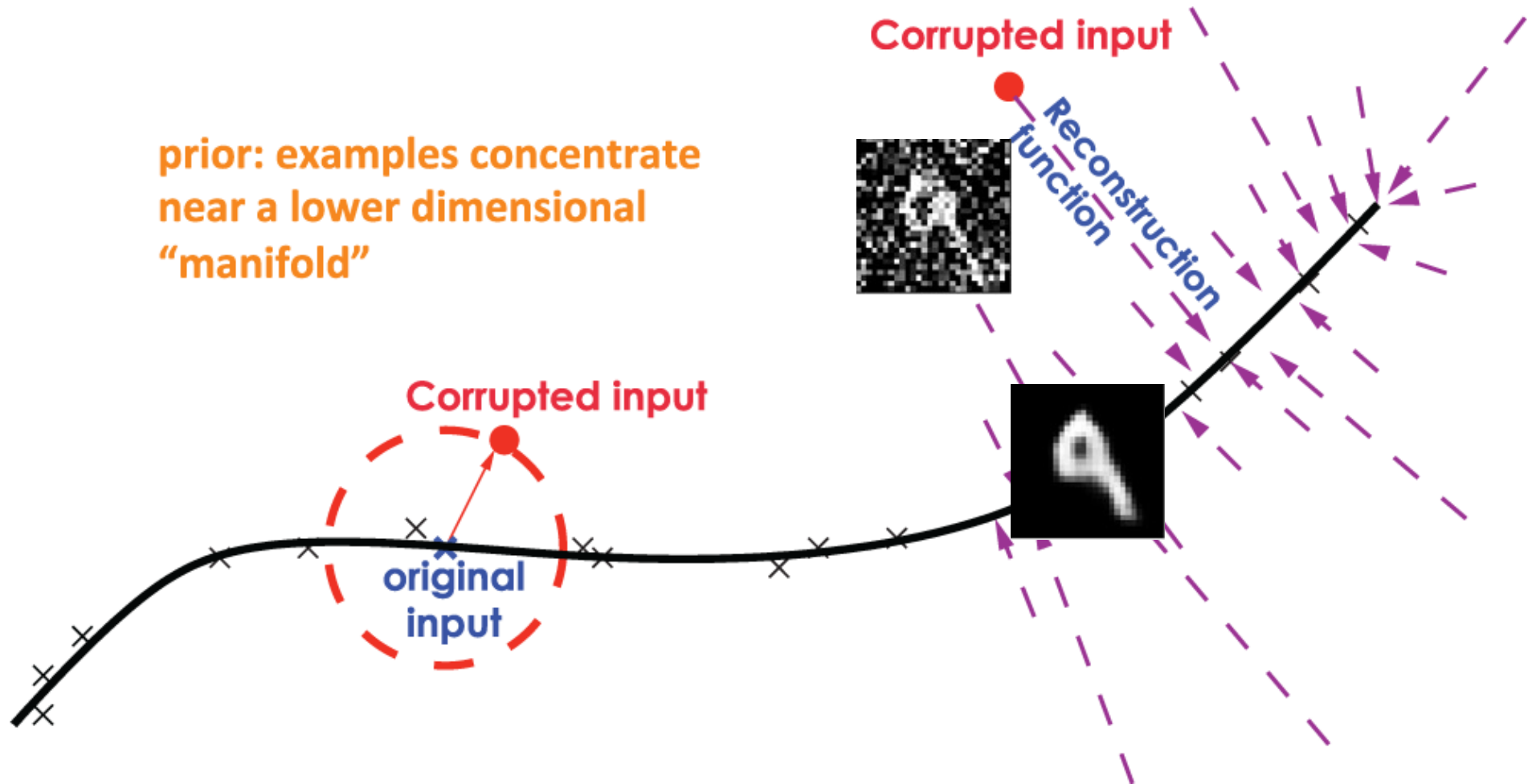


x



DENOISING AUTOENCODERS

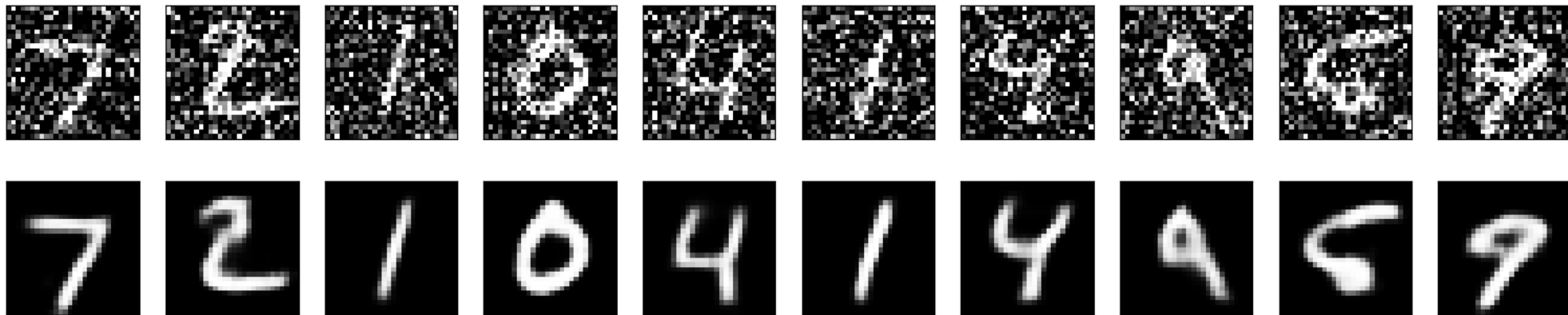
prior: examples concentrate
near a lower dimensional
“manifold”



DENOISING CONVOLUTIONAL AE – KERAS

DENOISING CONVOLUTIONAL AE – KERAS

- 50 epochs.
- Noise factor 0.5
- 92% accuracy on validation set.



STACKED AE

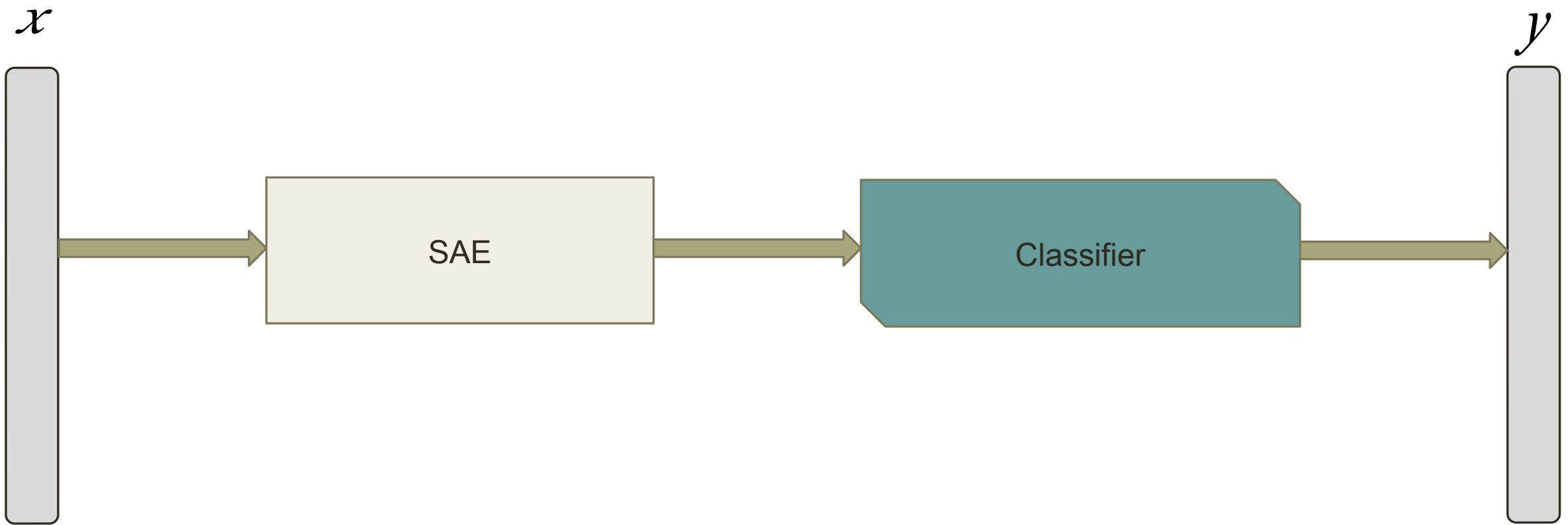
- Motivation:

- ❑ We want to harness the feature extraction quality of an AE for our advantage.
- ❑ For example: we can build a deep supervised classifier where it's input is the output of a SAE.
- ❑ The benefit: our deep model's W are not randomly initialized but are rather "smartly selected"
- ❑ Also using this unsupervised technique lets us have a larger unlabeled dataset.

STACKED AE

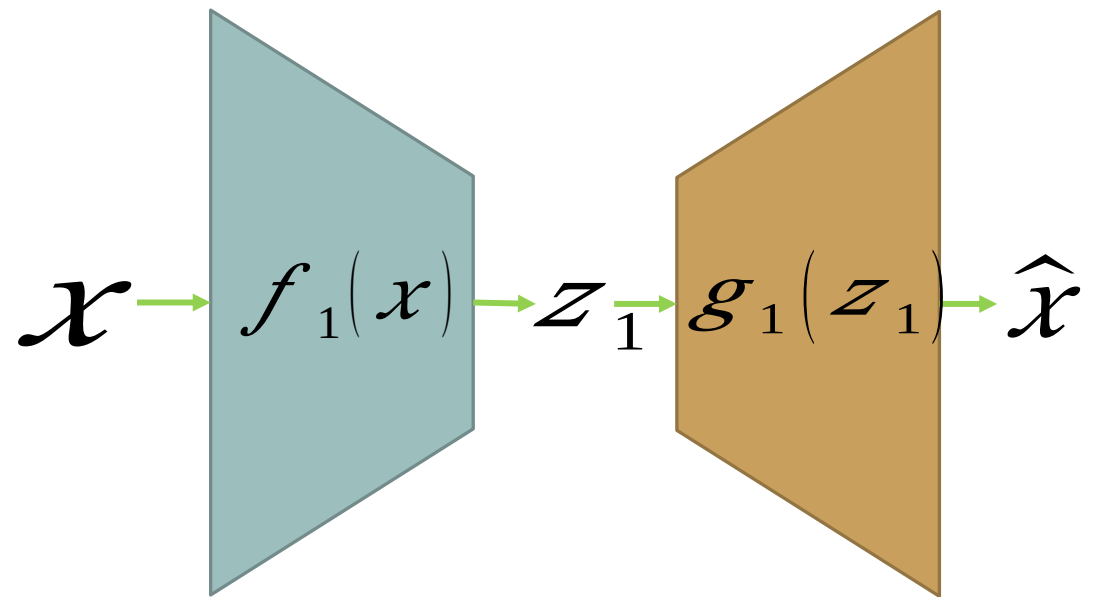
- Building a SAE consists of two phases:
 - 1. Train each AE layer one after the other.
 - 2. Connect any classifier (SVM / FC NN layer etc.)

STACKED AE



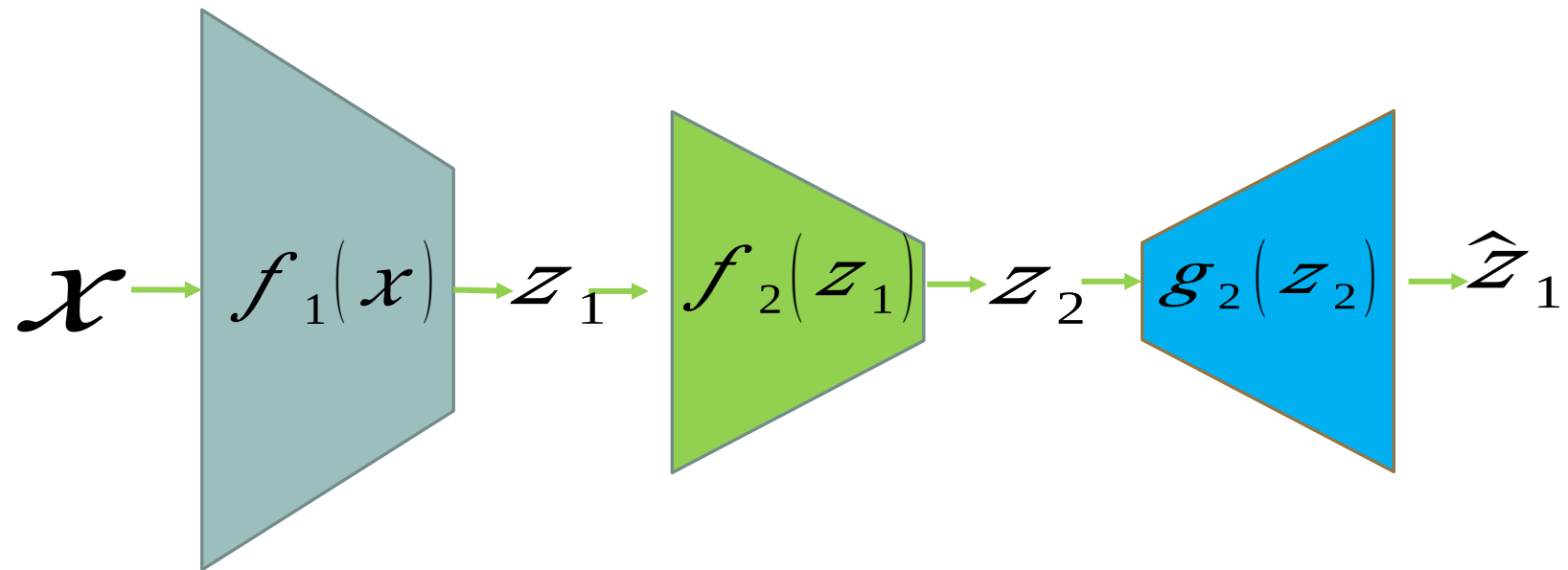
STACKED AE – TRAIN PROCESS

First Layer Training (AE
1)



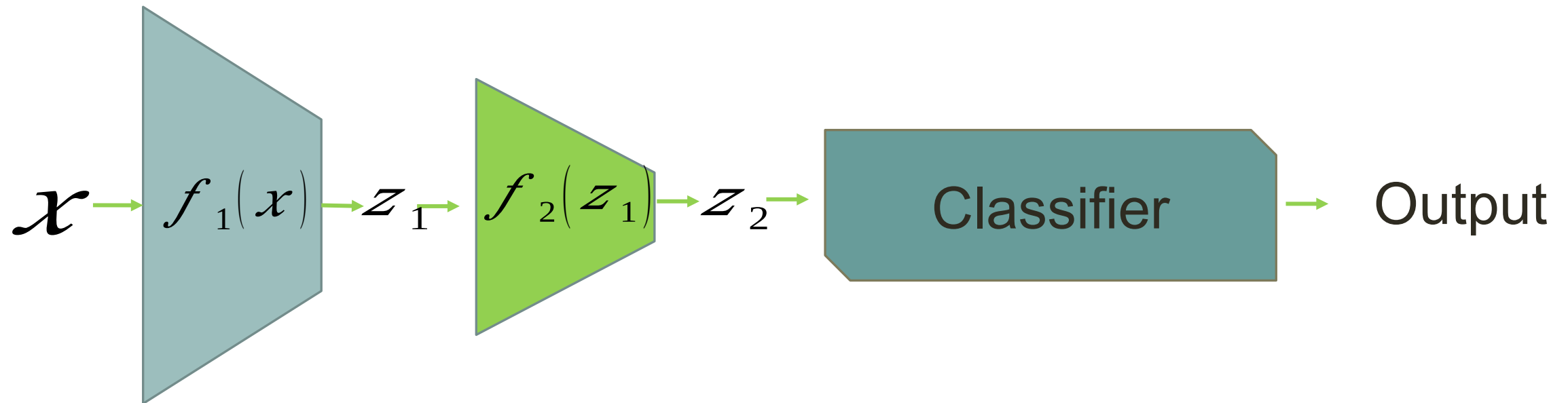
STACKED AE – TRAIN PROCESS

Second Layer Training (AE 2)



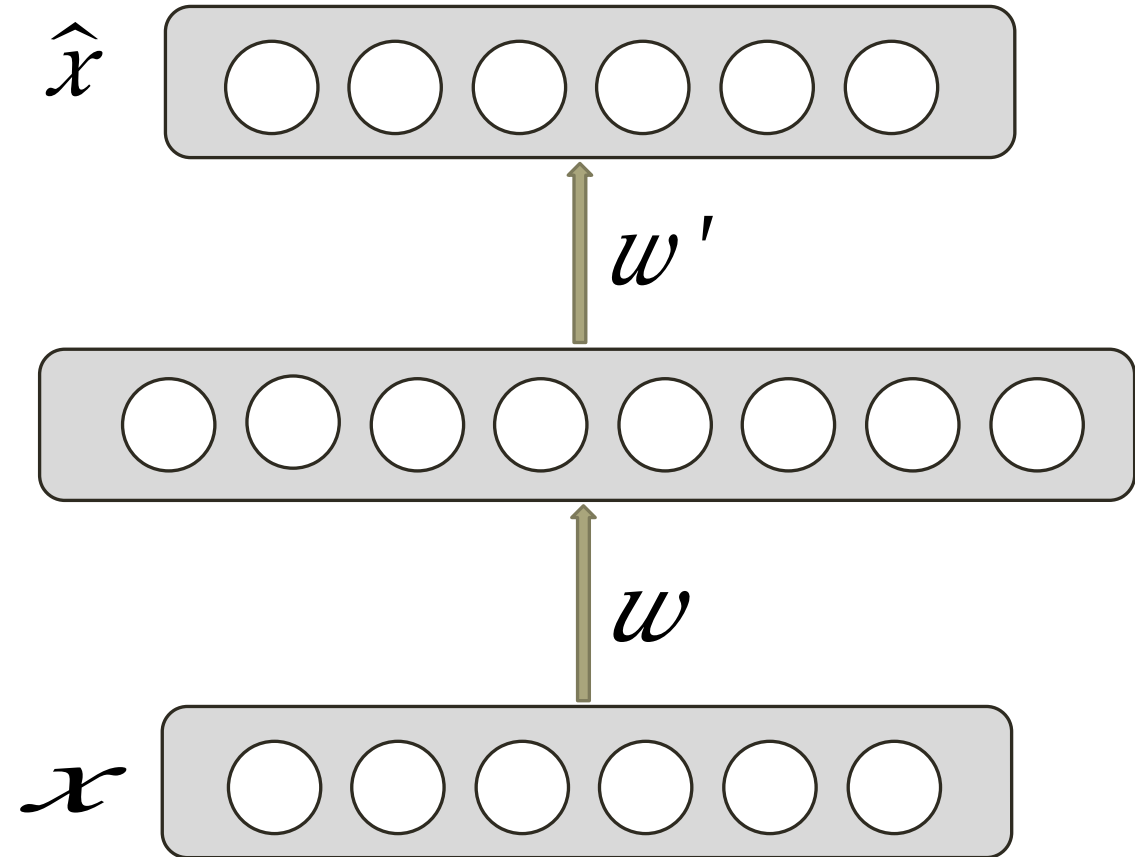
STACKED AE – TRAIN PROCESS

Add any classifier



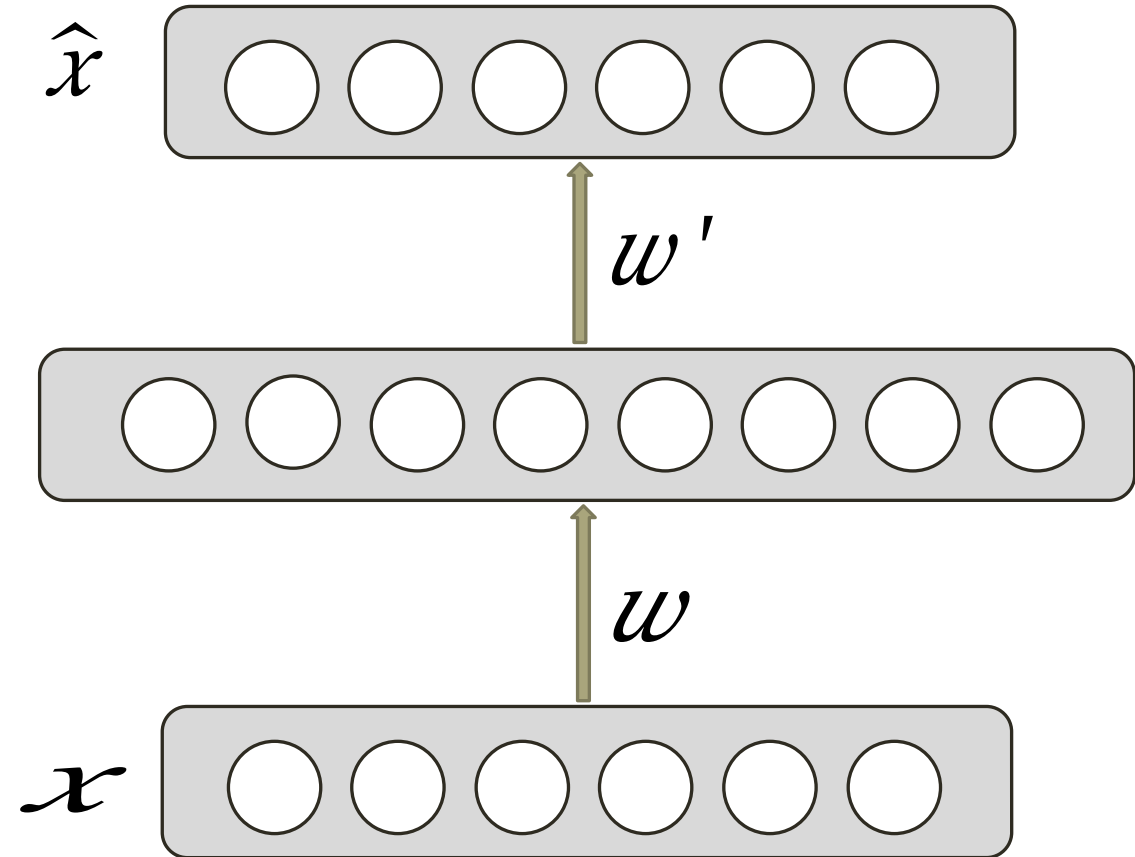
CONTRACTIVE AUTOENCODERS

- We are still trying to avoid uninteresting features.
- Here we add a regularization term to our loss function to limit the hidden layer.



CONTRACTIVE AUTOENCODERS

- Idea: We wish to extract features that **only** reflect variations observed in the training set. We would like to be invariant to the other variations.
- Points close to each other in the input space maintain that property in the latent space.



CONTRACTIVE AUTOENCODERS

Definitions and reminders:

- Frobenius norm (L2):

$$\sqrt{\sum_{i,j} |a_{ij}|^2}$$

- Jacobian Matrix: $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$ $\frac{\partial \mathcal{L}}{\partial \mathbf{z}}$

CONTRACTIVE AUTOENCODERS

Our new loss function would be:

where λ or simply:

and where λ controls the balance of our reconstruction objective and the hidden layer “flatness”.

CONTRACTIVE AUTOENCODERS

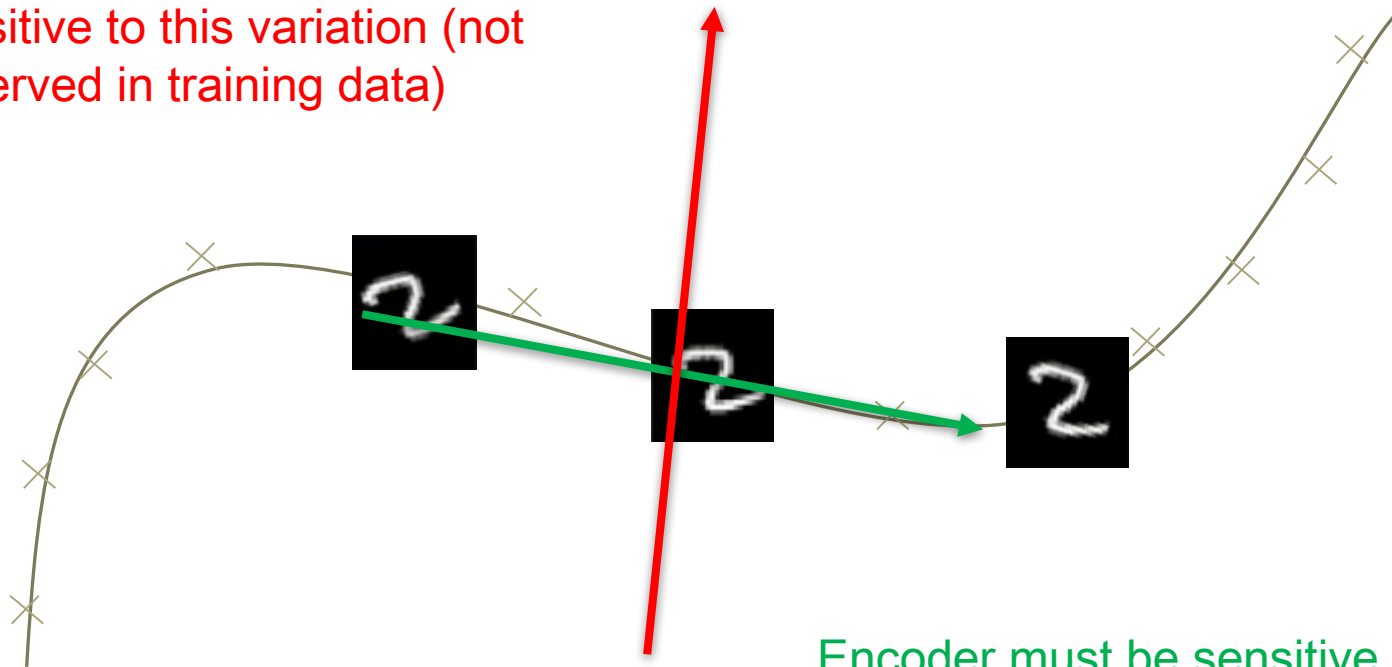
Our new loss function would be:

- would be an encoder that keeps good information ()
- would be an encoder that throws away all information ()

Combination
would be an
encoder that
keeps **only**
good
information.

CONTRACTIVE AUTOENCODERS

Encoder doesn't need to be sensitive to this variation (not observed in training data)



Encoder must be sensitive to this variation to reconstruct well

WHICH AUTOENCODER?

- DAE make the **reconstruction function** resist small, finite sized perturbations in input.
- CAE make the **feature encoding function** resist small, infinitesimal perturbations in input.
- Both denoising AE and contractive AE perform well!

WHICH AUTOENCODER?

- Advantage of DAE: simpler to implement
 - Requires adding one or two lines of code to regular AE.
 - No need to compute Jacobian of hidden layer.
- Advantage of CAE: gradient is deterministic.
 - might be more stable than DAE, which uses a sampled gradient.
 - one less hyper-parameter to tune (noise-factor)

REFERENCES

1. <https://arxiv.org/pdf/1206.5538.pdf>
2. <http://www.deeplearningbook.org/contents/autoencoders.html>
3. <http://deeplearning.net/tutorial/dA.html>
4. <http://ufldl.stanford.edu/tutorial/unsupervised/Autoencoders/>
5. http://ufldl.stanford.edu/wiki/index.php/Stacked_Autoencoders
6. <http://www.jmlr.org/papers/volume11/vincent10a/vincent10a.pdf>
7. <https://codeburst.io/deep-learning-types-and-autoencoders-a40ee6754663>