Lecture 26: Autoencoders

Marios Mattheakis

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, Chris Tanner

Pavlos game #4275

WHO IS MOST SIMILAR TO PAVLOS?

Option a

Option B

Option C

WHO IS MOST SIMILAR TO PAVLOS?

COSINE SIMILARITY ( ) = 0.987 ✓

COSINE SIMILARITY ( ) = 0.912

COSINE SIMILARITY ( ) = 0.826

Winner !

The prize of the competition was to give a lecture for the CS109B

Research Associate at IACS

Doing research in the intersection of data science and applied physics developing deep neural network architectures for:

- Solving differential equations
- Eigenvalue quantum problems
- Material science (crystal structures)
- Hamiltonian Networks
- Inverse problems
- …

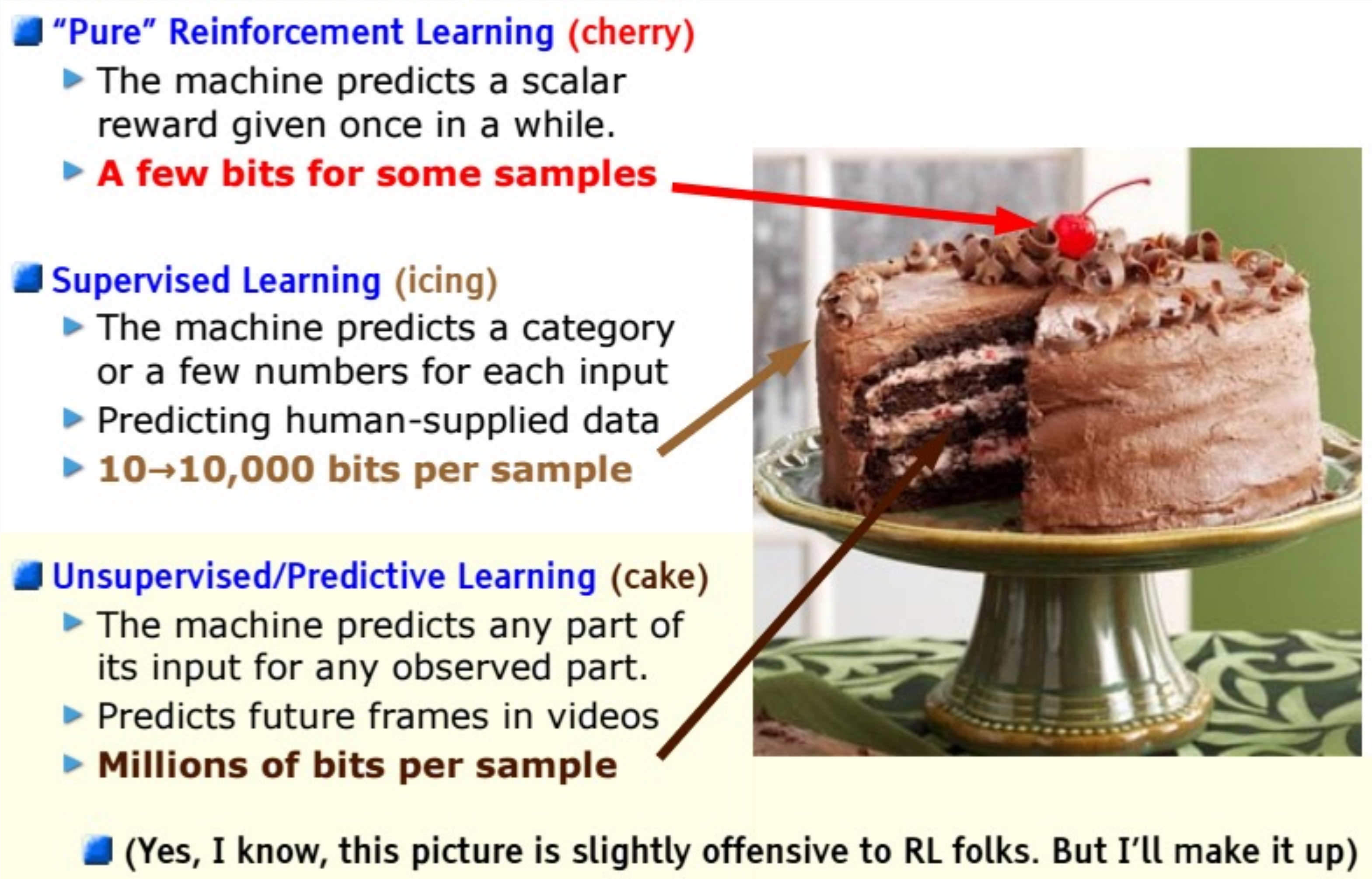
https://scholar.harvard.edu/marios_matthaiakis/home

Lecture 26: Autoencoders

Marios Mattheakis

CS109B Data Science 2

Pavlos Protopapas, Mark Glickman, Chris Tanner



Original Yann LeCun cake analogy slide presented at NeurIPS 2016.
The highlighted area has now been updated.

Y. LeCun

How Much Information is the Machine Given during Learning?

- **"Pure" Reinforcement Learning (cherry)**
  - The machine predicts a scalar reward given once in a while.
  - **A few bits for some samples**
- **Supervised Learning (icing)**
  - The machine predicts a category or a few numbers for each input
  - Predicting human-supplied data
  - **10→10,000 bits per sample**
- **Self-Supervised Learning (cake génoise)**
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**

 1.1: Deep Learning Hardware: Past, Present, & Future 59

LeCun updated his cake recipe at the 2019 International Solid-State Circuits Conference (ISSCC) in San Francisco, replacing "unsupervised learning" with "self-supervised learning," a variant of unsupervised learning where the data provides the supervision.

Unsupervised or self-supervised learning

Self-supervised learning can be challenging

Autoencoders Part A

Outline

- **What are autoencoders?**
- **Brief history of encoding/decoding.**
- **Inside autoencoders.**
- Convolutional autoencoders.
- Regularization of autoencoders.
- Applications
  - Denoising
  - Blending

Neural Networks as universal function approximators

Given an input x and an output y there exists a mapping from input space to output space as follows:

$$x \rightarrow y$$
$$y = f(x) + \epsilon$$

Our goal is to find an estimate of $f(x)$ which we will call $\hat{f}(x)$.

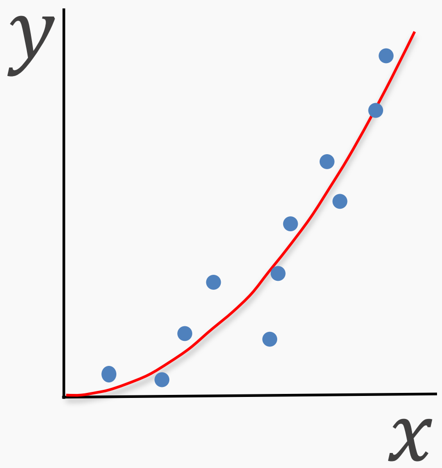
Statistical learning or modeling is the process of finding $\hat{f}(x)$.

Neural networks are one of many possible methods we can use to obtain the estimate $\hat{f}(x)$.

Linear Regression

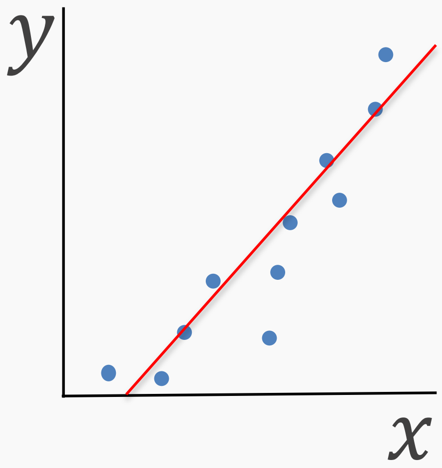
$x \rightarrow$ Linear Regression $\rightarrow \hat{y}$

Fit quadratic function



$$y = x^2$$

Do your best !

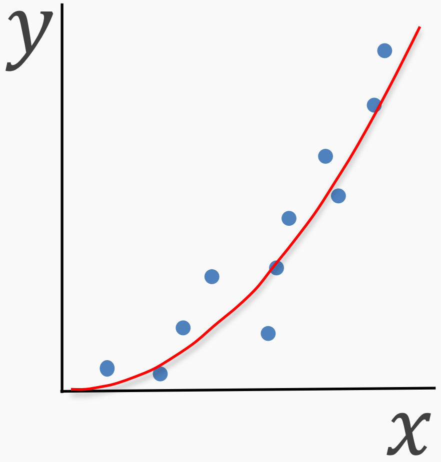


But we know that this is not the best

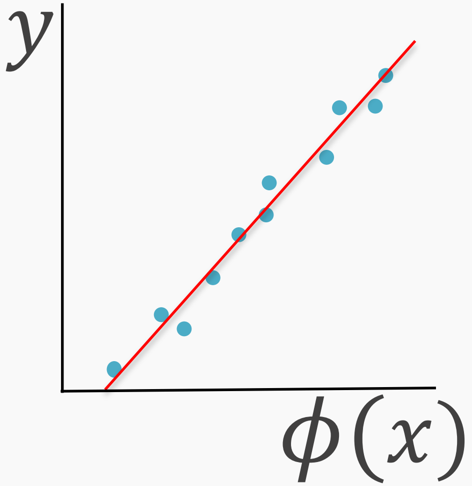
Representational Learning

Representation Matters

$x \rightarrow$ Learn Representation $\rightarrow \phi(x) \rightarrow$ Linear Regression $\rightarrow \hat{y}$



$$\boxed{\phi(x) = x^2}$$



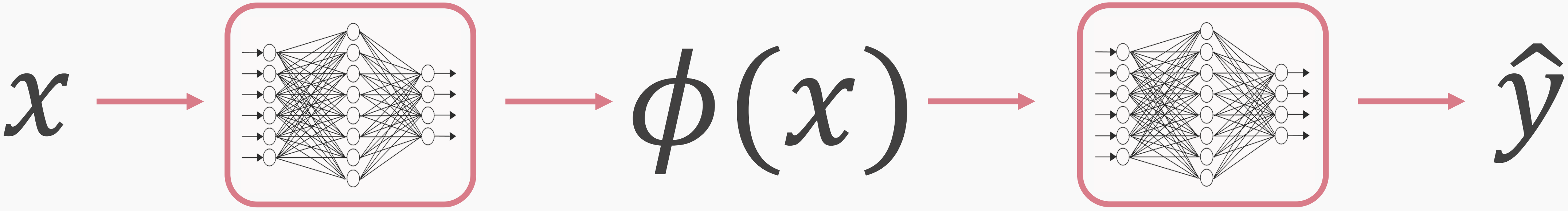
Representational Learning

Representation Matters

$$x \rightarrow \boxed{\text{Learn Representation}} \rightarrow \phi(x) \rightarrow \boxed{\text{Task}} \rightarrow \hat{y}$$

Representational Learning

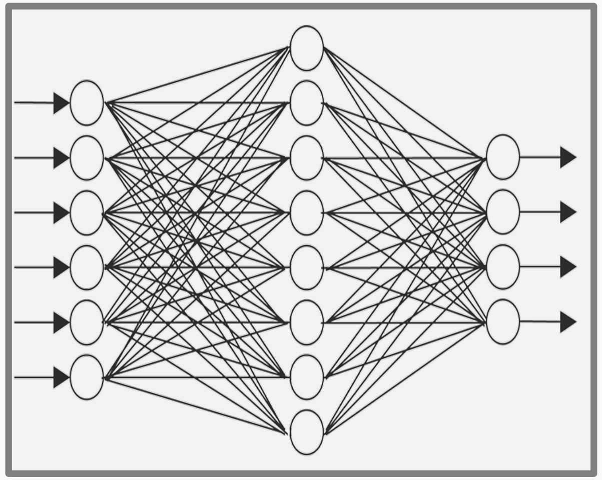
Representation Matters



Representational Learning: **Supervised Learning**

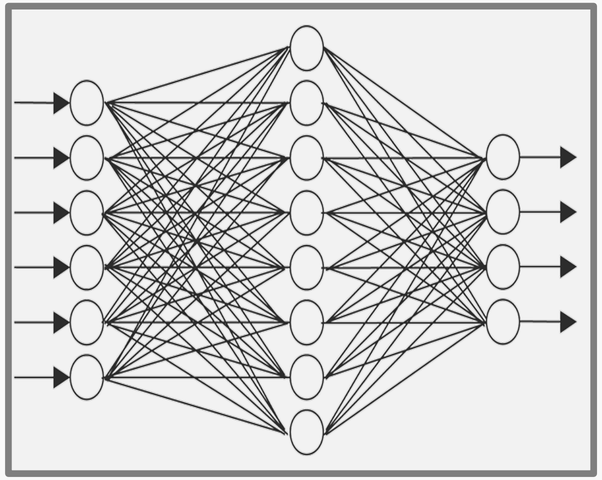
We train the two networks by minimizing the loss function (**cross entropy loss**)

X



Feature Discovery Network

Features
$\phi(x)$



Classification Network

Y

{Cat,Dog}

Representational Learning: **Self-supervised Learning**

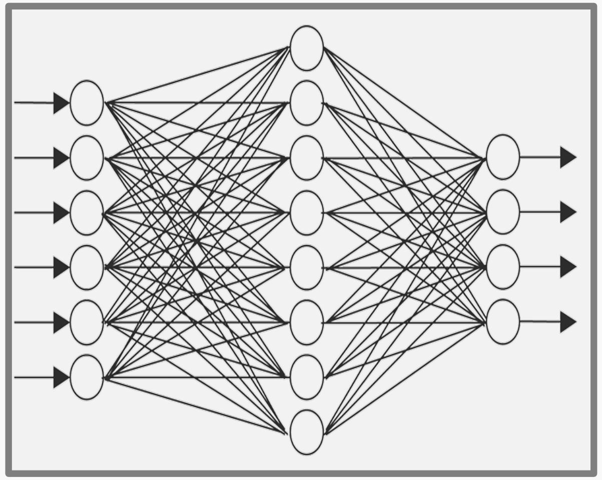
We train the two networks by minimizing th… **entropy loss**)

Specialized features for Task

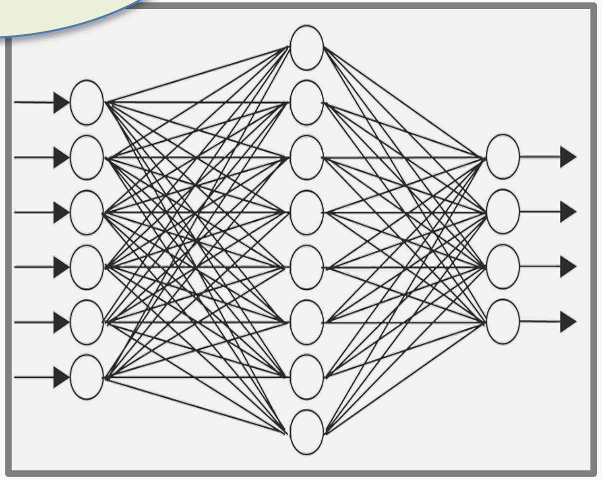
No labels

X

Y



Features $\phi(x)$



{Cat,Dog}

Feature Discovery Network

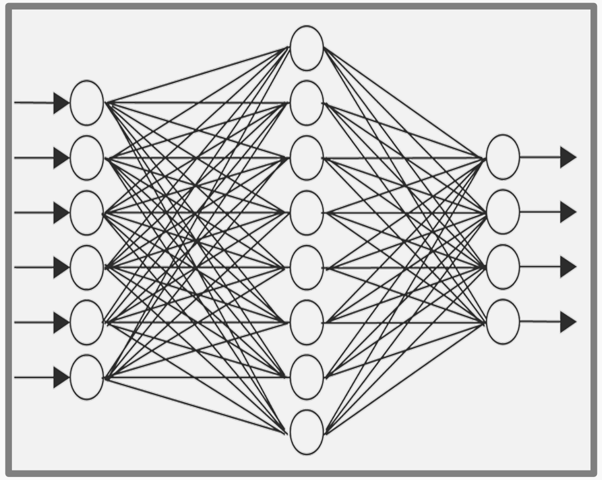
Classification Network

Representational Learning: **Self-supervised Learning**

We train the two networks by minimizing the **reconstruction** loss function:

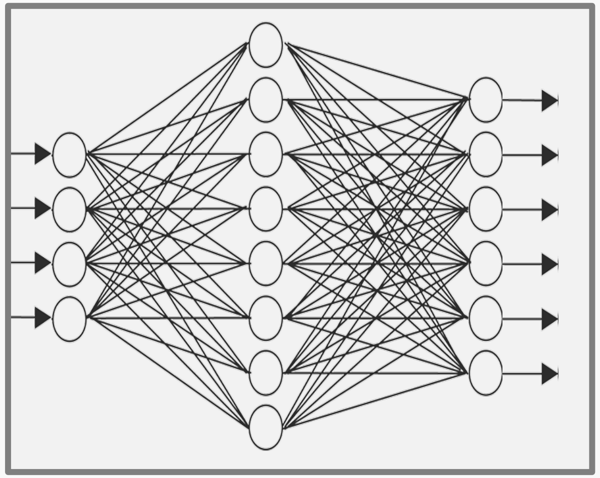
$$\mathcal{L} = \sum (x_i - \hat{x}_i)^2$$

X



Feature Discovery Network

Features $\phi(x)$



Second Network

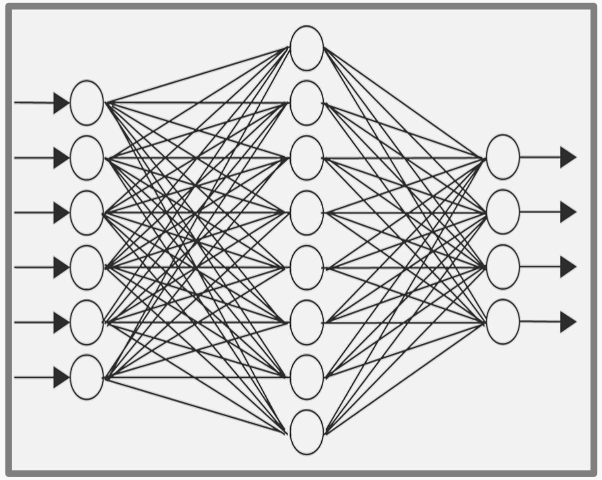
$\hat{X}$

Representational Learning: **Self-supervised Learning**

We train the two networks by minimizing the **reconstruction** loss function:

$$\mathcal{L} = \sum (x_i - \hat{x}_i)^2$$

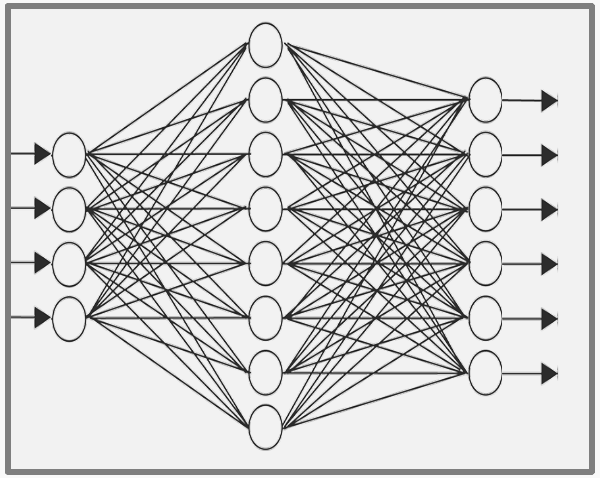
X



ENCODER

Latent Space Z

$\hat{X}$



DECODER

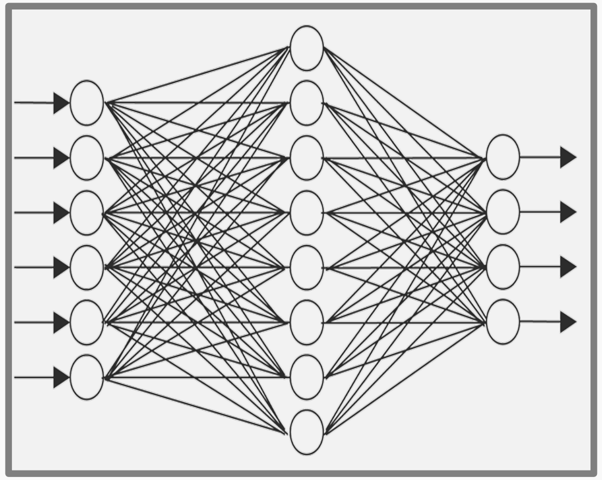
Representational Learning: **Self-supervised Learning**

We train the two networks by minimizing the **reconstruction** loss function:

$$\mathcal{L} = \sum (x_i - \hat{x}_i)^2$$

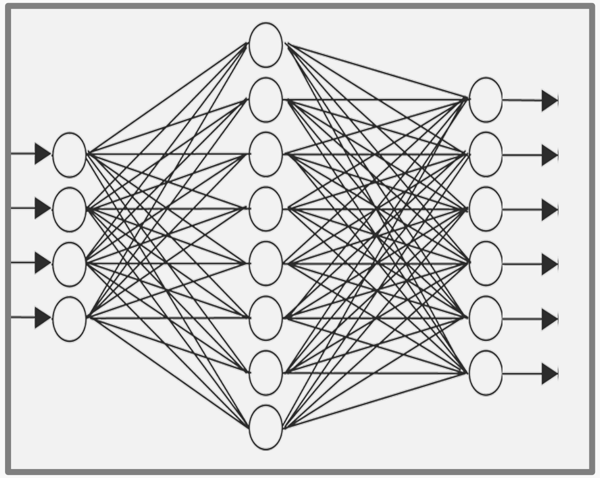
AUTOENCODER

X



ENCODER

Latent Space Z



DECODER

$\hat{X}$

This is an autoencoder. It gets that name because it automatically finds the best way to encode the input so that the decoded version is as close as possible to the input.

Brief history of encoding/decoding

Is this a new idea?

- **MP3** can compress music files by a factor of 10 enabling digital storage and transmission large volumes of audio.
- **JPG** compresses images by a factor of 10-20 and enables storage and transmission of image data.
- These technologies led the way to the image-rich web and abundance of music that we enjoy today.

Brief history of encoding/decoding (cont)

MP3 and JPG take an input and **encode** it into a **compressed** form.

Then they **decode** or **decompress** the compressed version back to the original version.

Do we get the same quality after the decoding?

Lossless and Lossy Encoding

Loss: The difference between the original and post-decompression object

Example: Imagine, I want to make a presentation for upcoming Pavlos' group meeting, so I am asking Pavlos on slack about the topic of the talk

Pavlos, which topic should I present in the next group meeting?

Pavlos is biking (his regular 100Km distance) but also wants to reply at the same time

IMO RL

Immediately, I read (knowing that Pavlos likes representation learning)

In My Opinion Representation Learning

A 33 characters message is compressed to 5 characters (that's a good compression)

Lossless and Lossy Encoding (cont)

Question: Is this an example of lossy or lossless compression?

After a week of hard preparation, I got an excellent presentation on

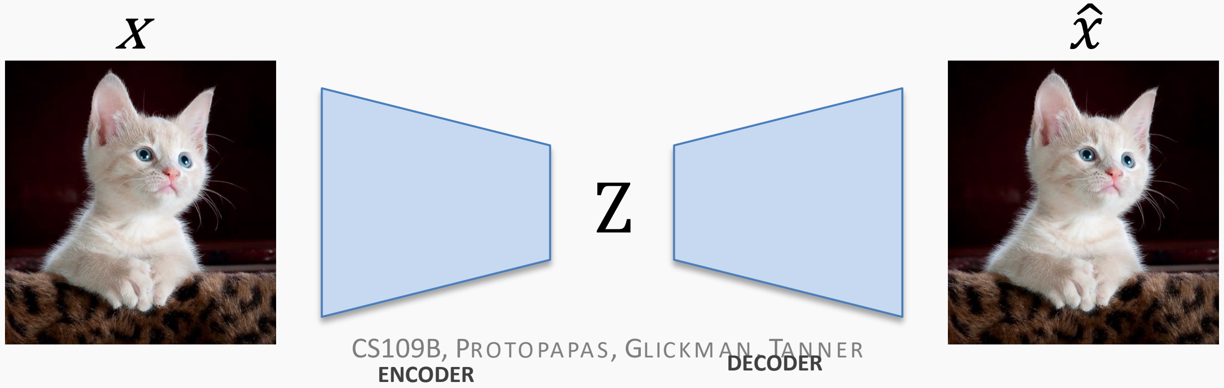
Representation Learning

but Pavlos and the other members of the Lab were expecting a talk on

Reinforcement Learning

A way to test if a transformation is **lossy** or **lossless** is to measure the difference between the reconstructed and original data.
In autoencoders this is the **reconstruction loss function**.

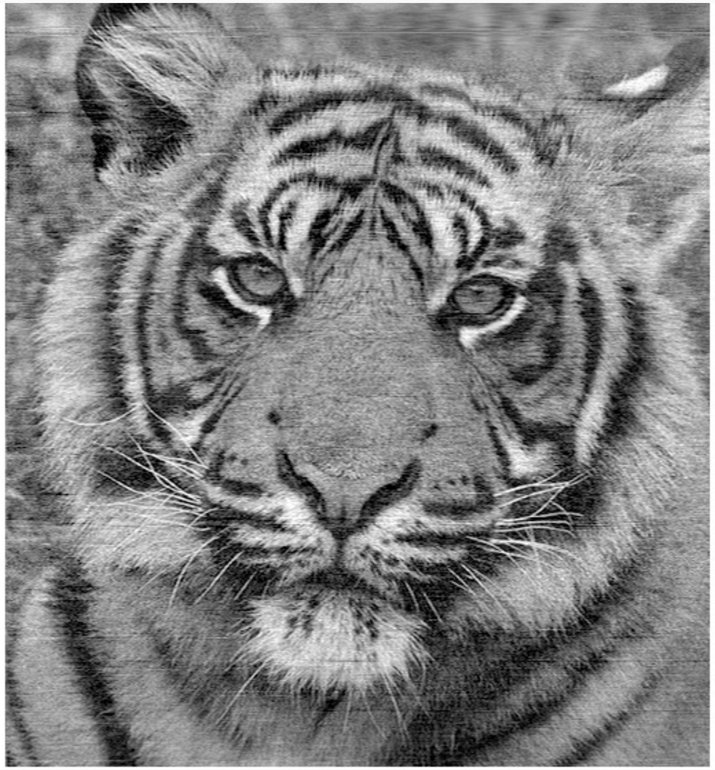
$$\mathcal{L} = \sum (x_i - \hat{x}_i)^2$$



MP3 and JPG Image Compression

original

256x256=262,000



MP3

37,000

JPG

26,000

Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

MP3 and JPG Image Compression(cont)

original

MP3

JPG

Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

What are autoencoders?

- A particular kind of deep learning architecture.
- Compress inputs into a form that can later be decompresses
- Autoencoders are more general than MP3 and JPG
- Able to find a general (abstract) representation of unlabeled data
- They are usually used to …
    - reduce data dimensionality
    - find general representation and underlying relationships
    - Image denoising, infilling, coloring, blending
    - Anomaly detection
    - …

Why a self-supervised learning method?

We say that an autoencoder is an example of **semi-supervised** or **self-supervised** learning.

It sort-of is **supervised** learning because we give the system explicit goal data (the output should be the same as the input),

and

it sort-of is **not supervised** learning because we don't have any manually determined labels or targets on the inputs.

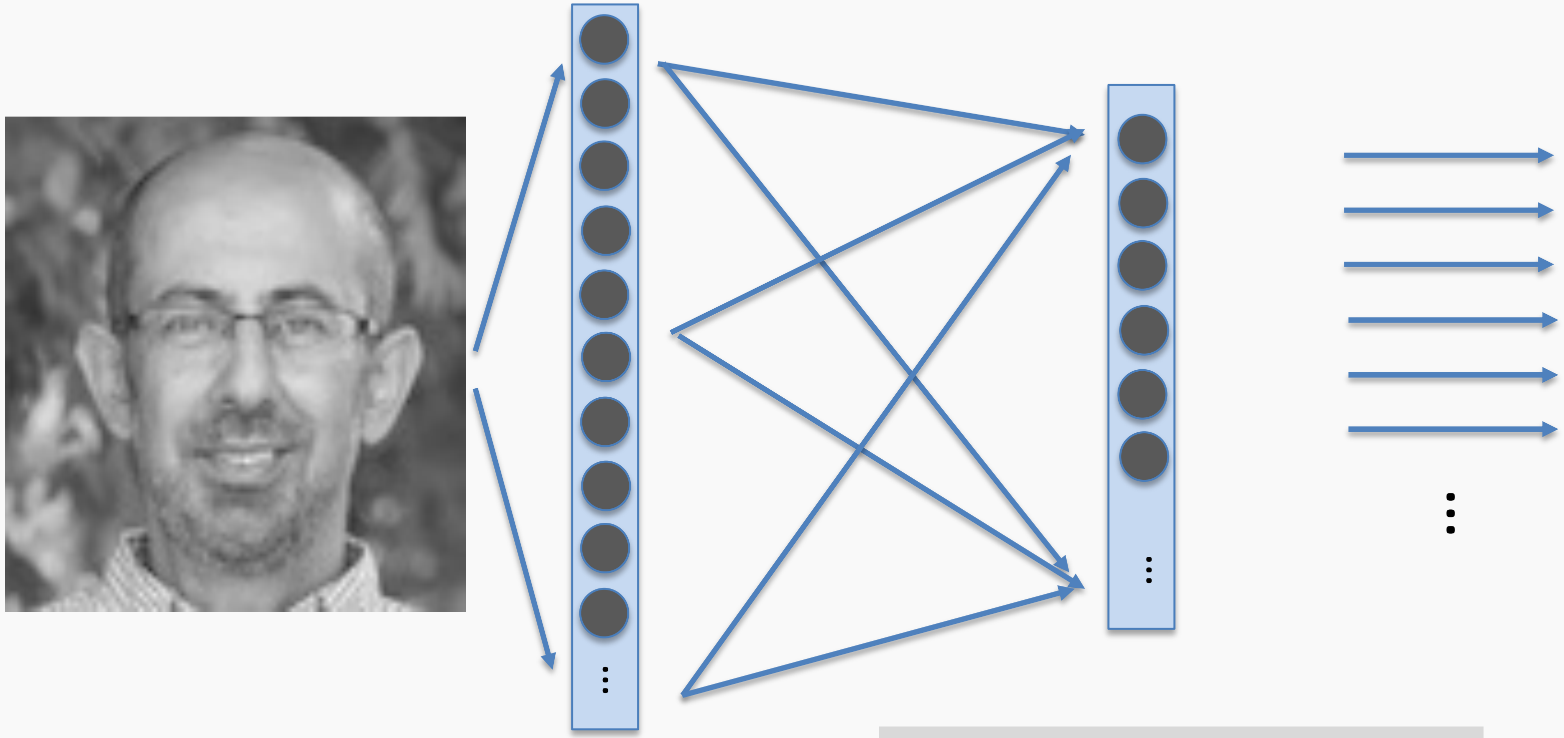
The simplest autoencoder

Encode with a simple fully connected network (FCN)

Image 100x100

Flatten Image 10,000

FCN with 20 neurons



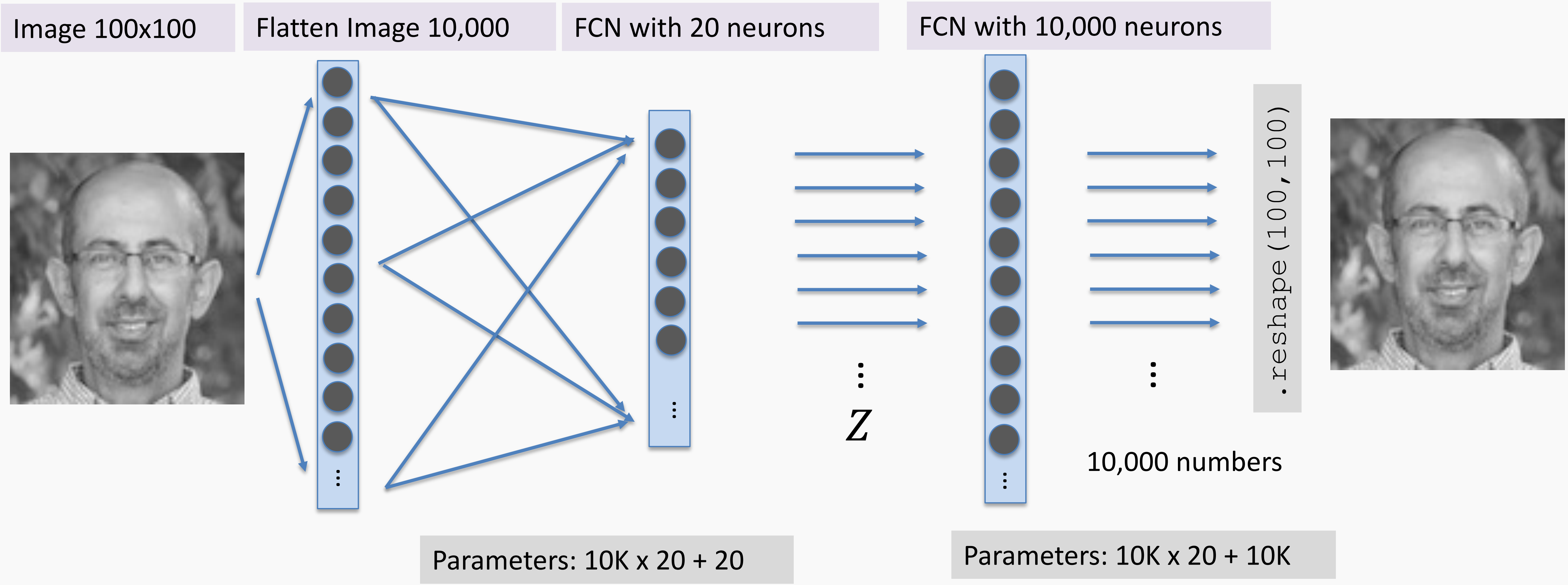
20 numbers

z

Parameters: 10K x 20 + 20

The simplest autoencoder

Encode and decode together after training



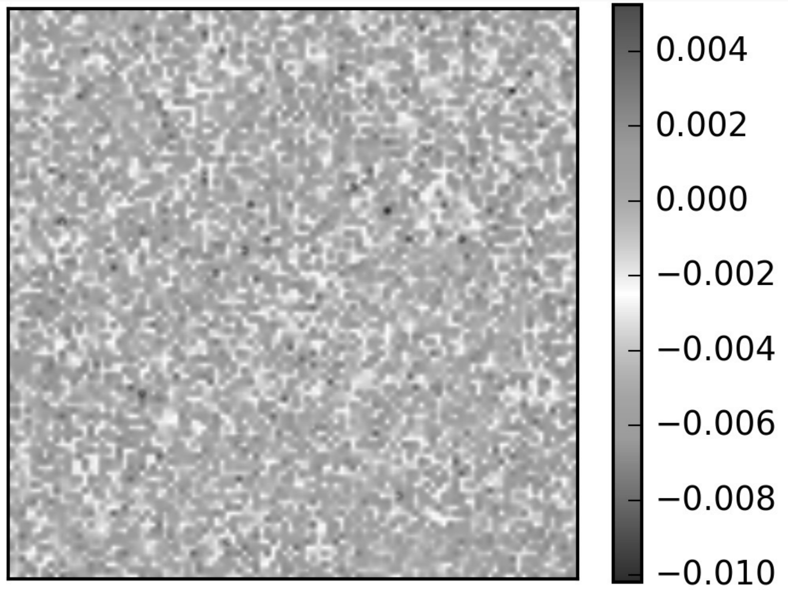
Autoencoders in action

Comparing the input and output pixel by pixel.

Input Image 100x100

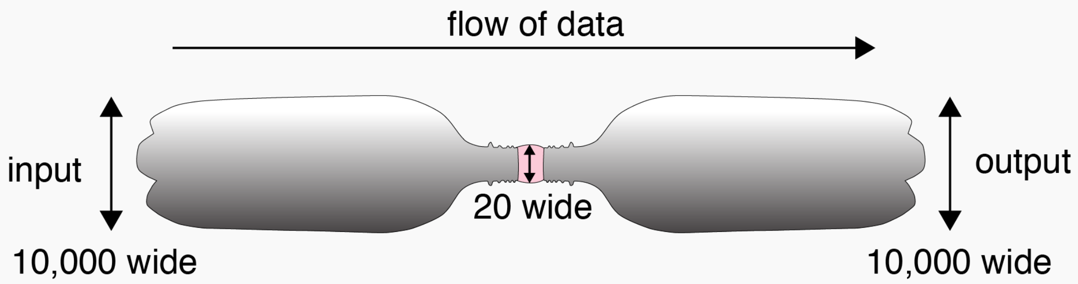
Output Image 100x100

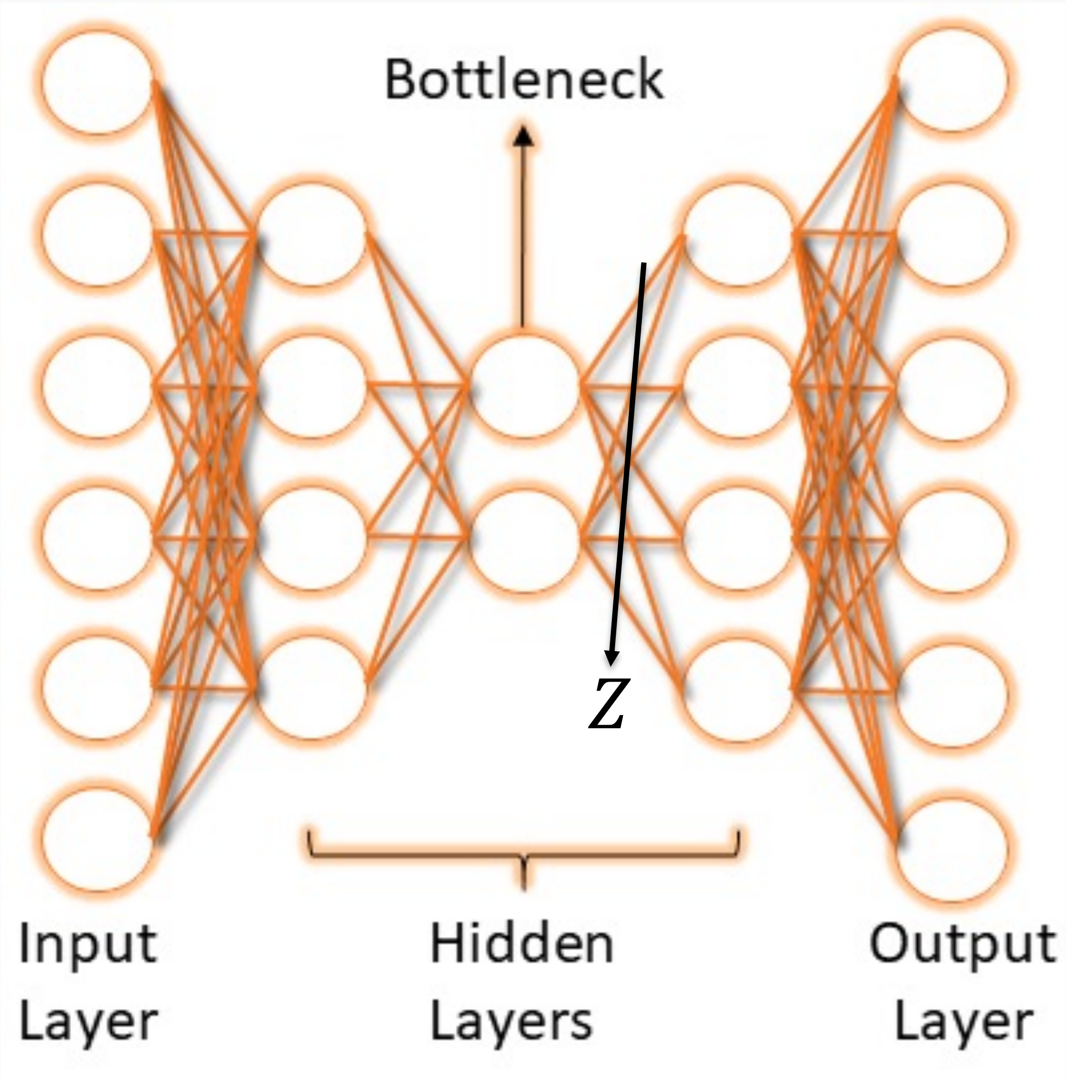
Residuals 100x100



Bottleneck

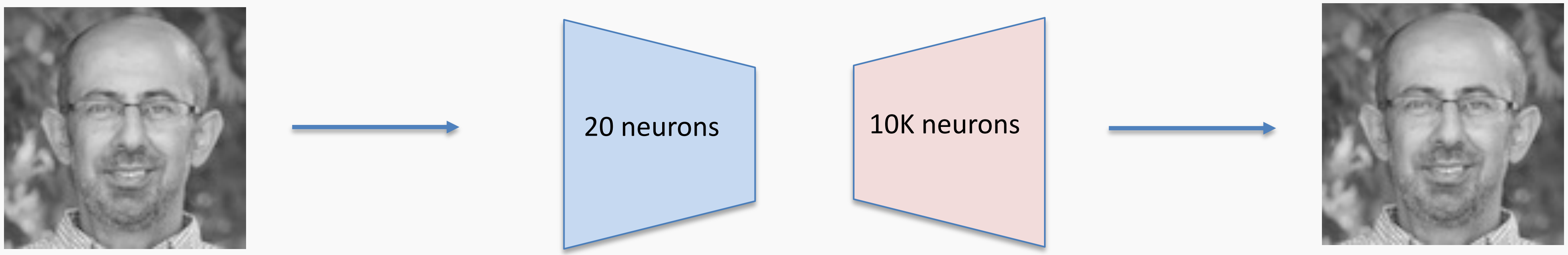
- We start with 10,000 elements
- We have 20 in the middle
- And 10,000 elements again at the end



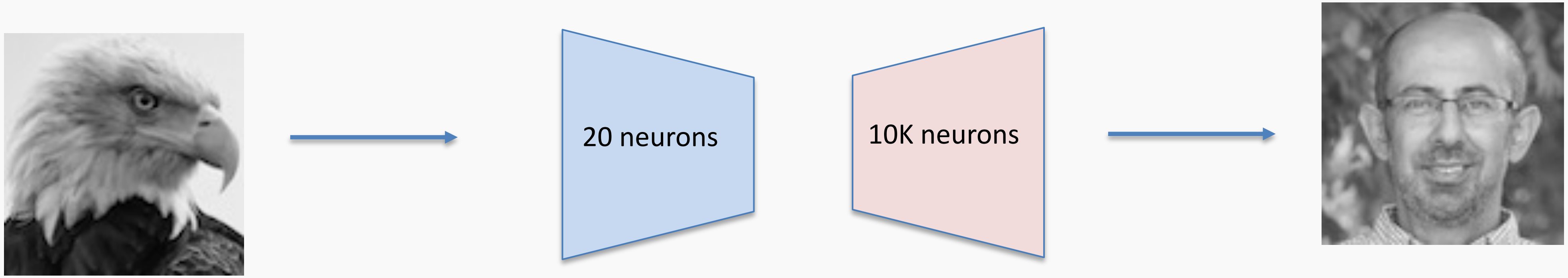


Autoencoders in action (cont)

Passing “Pavlos” to the previously trained autoencoder returns:



How about if we input the “Eagle”?

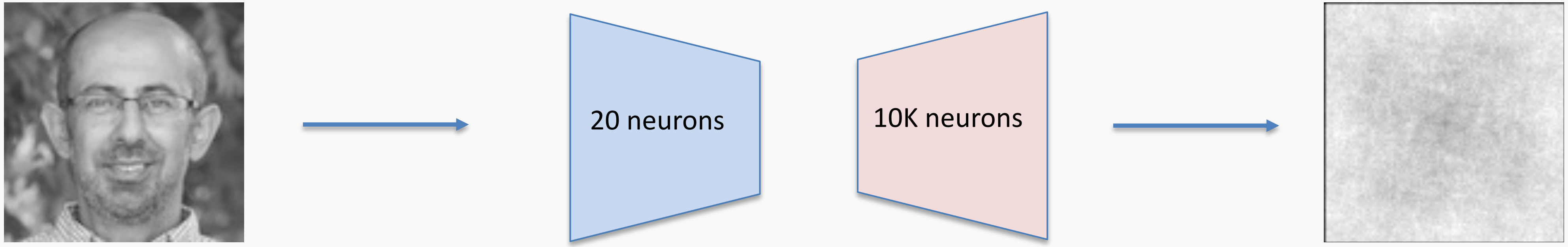


Autoencoders in action (cont)

We should **train** with a variety of images.

Autoencoders in action (cont)

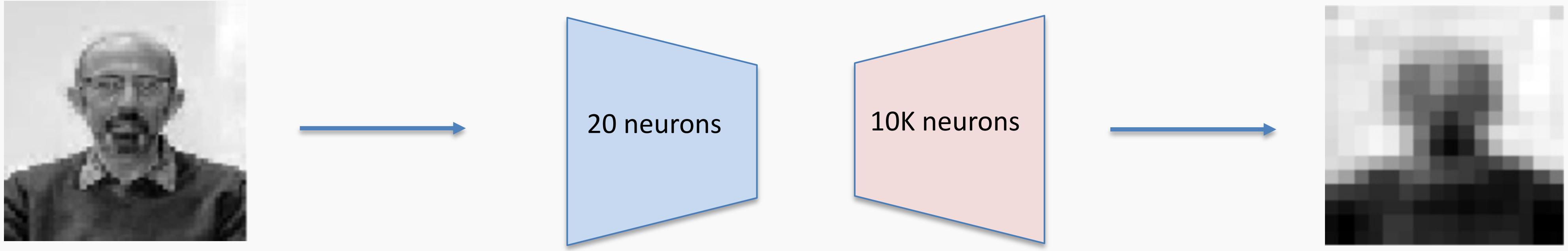
After training with those images, let’s test how well it generalizes:



Network has never seen anything like this, so it is no surprise that could not reconstruct a face.

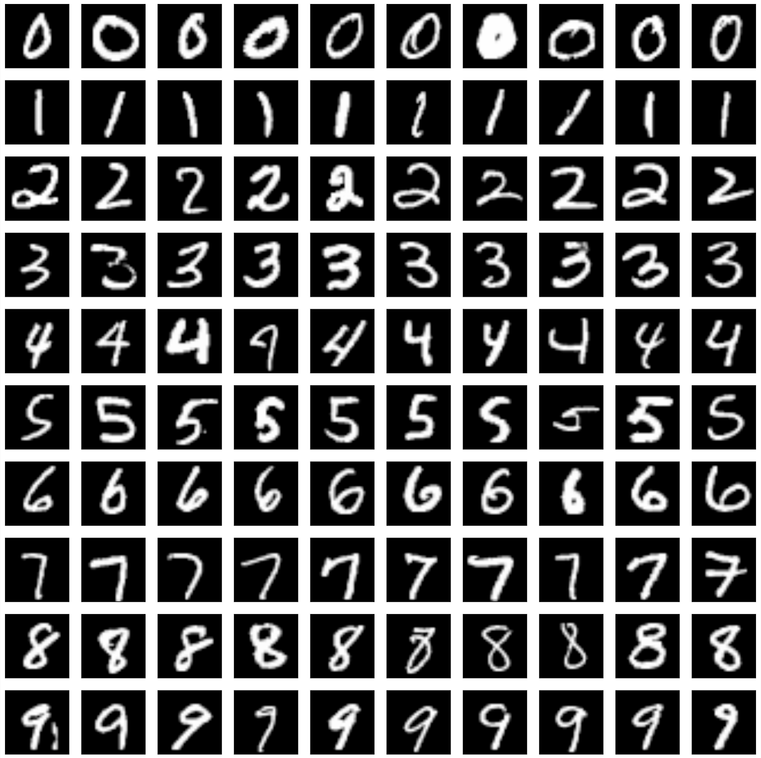
Autoencoders in action (cont)

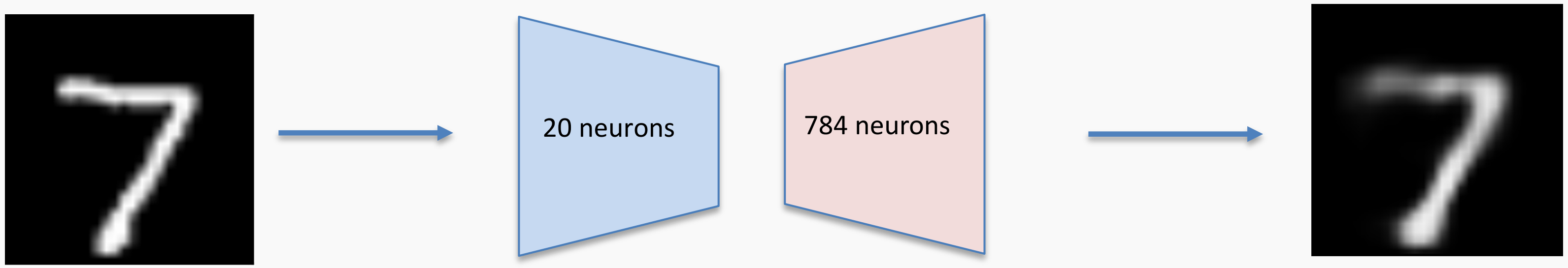
We can use a better training set such as the Olivetti faces



Train on a better dataset yields a better autoencoder

MNIST data: train a simple AE with one-layer FCN encoder and one-layer FCN decoder





20 latent variables

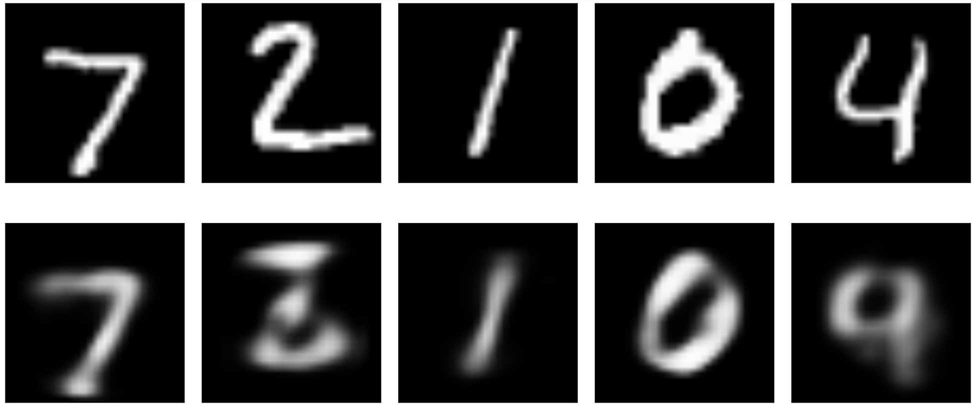
original

reconstructed

10 latent variables

original

reconstructed



WHY DOES IT WORK?

2 latent variables

original

reconstructed

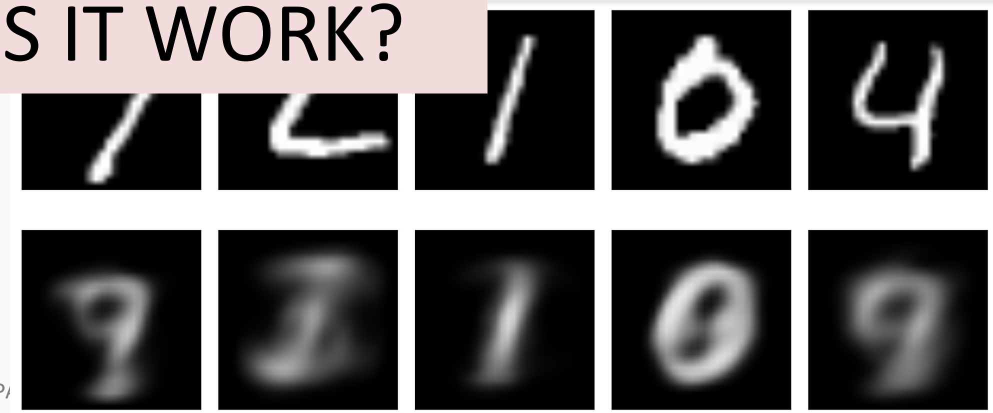


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Encoding is easy:

Considering a number with infinity number of decimals, we can encode the whole universe.

Learning representation is difficult:

We need sufficient latent dimensions to learn the underlying relationships that are hidden in the data.

WHY DOES IT WORK?

2 latent variables

original

reconstructed

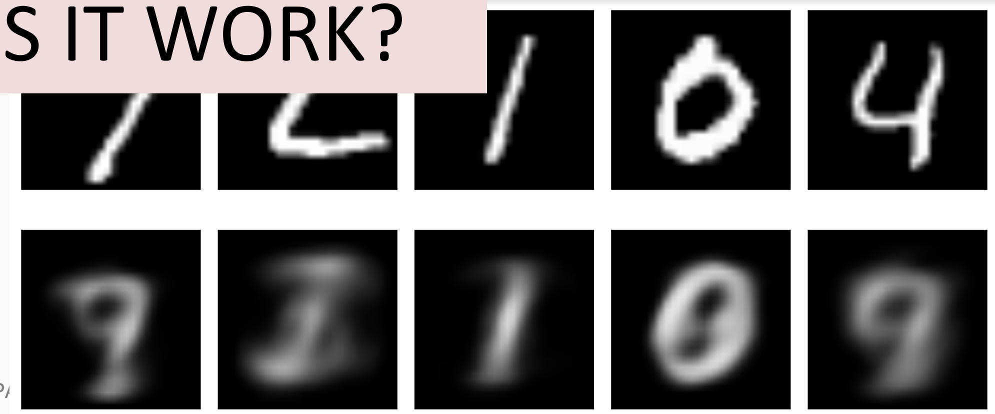
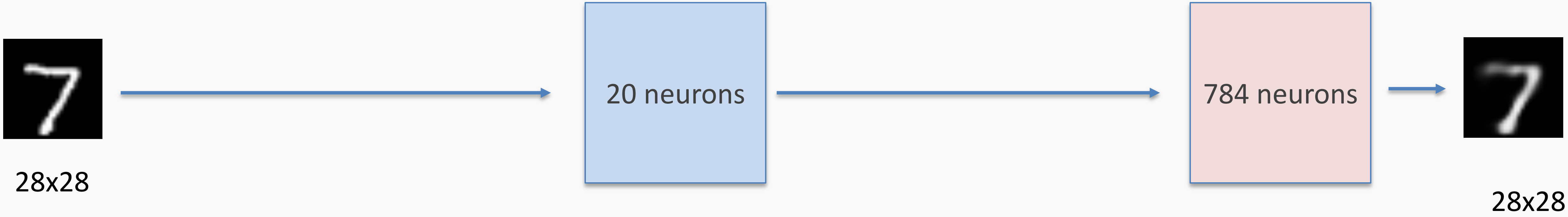


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Deeper

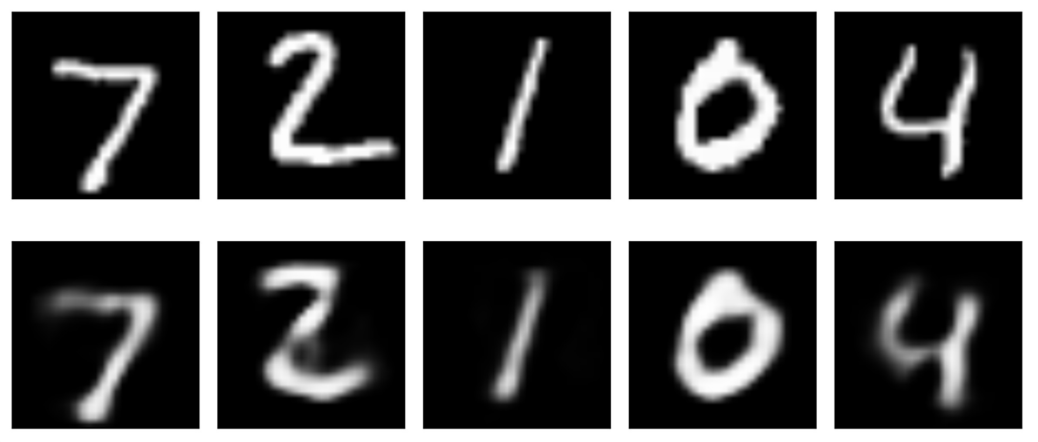
For a better representation we can add neurons in one layer or go deeper.



Z is 20 dimensional

Original Images

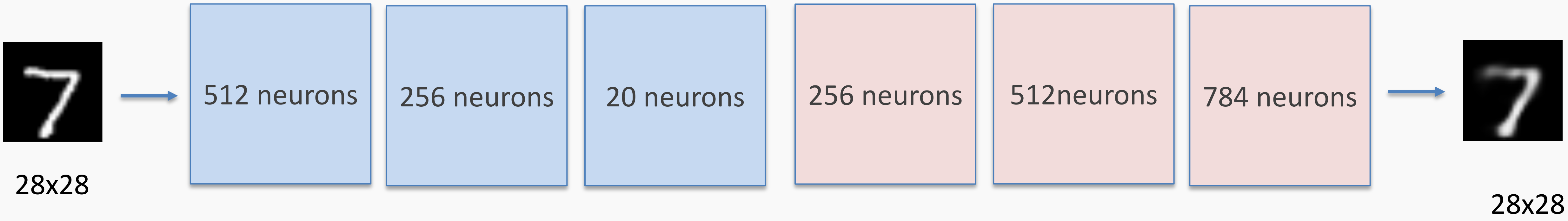
Reconstructed Images



Shallow AE: 20 latent variables

Deeper

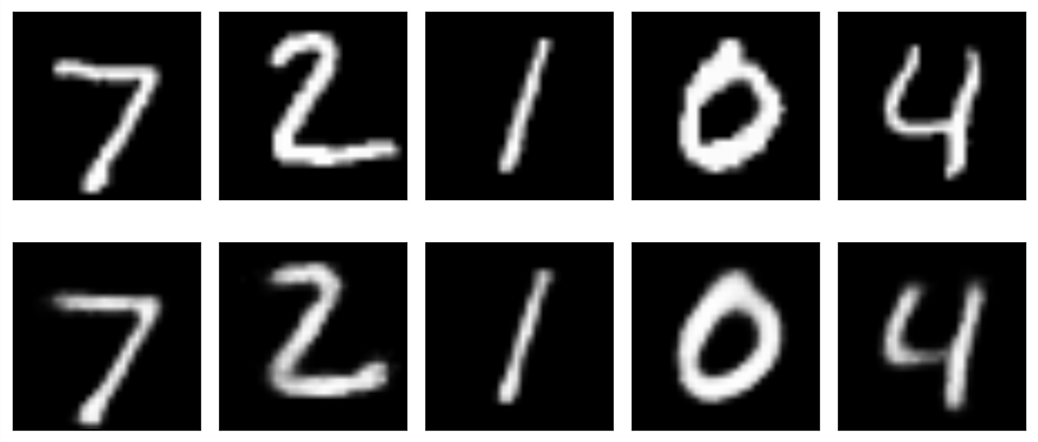
For a better representation we can add neurons in one layer or go deeper.



Z is 20 dimensional

Original Images

Reconstructed Images



Deep AE: 20 latent variables

DEEPER IS BETTER

Latent space of autoencoder

If the AE learns the "essence" of the MNIST images, similar images should be close to each other in the Z space, implying **Contextual learning**

Plot a 2D projection of the latent space to examine the separation (2 PCA components)

Labels (colors) and PCA are used just for visual inspection (evaluation)

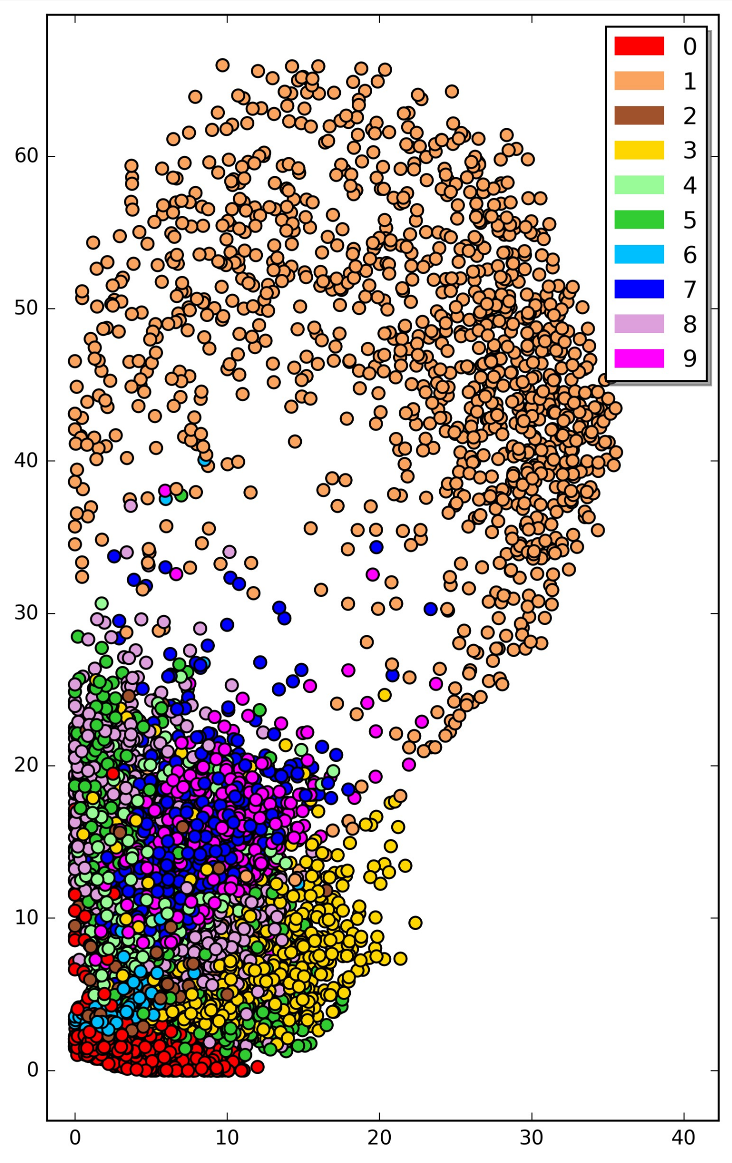
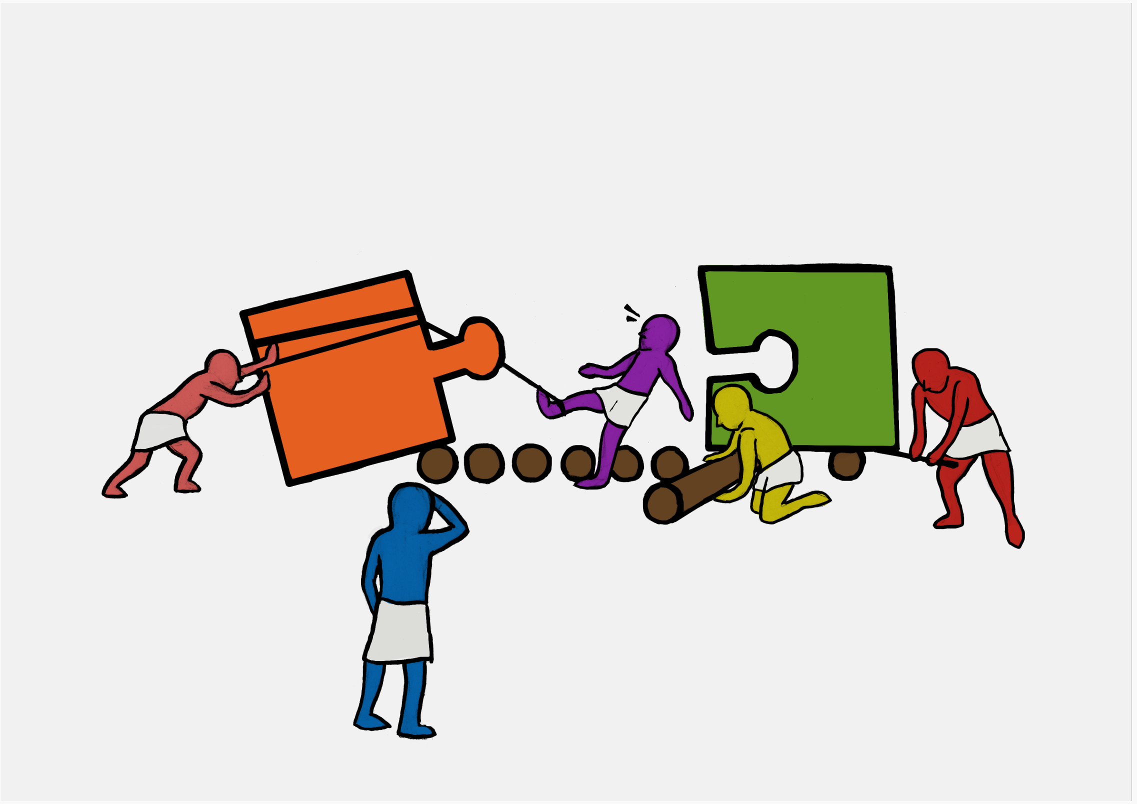


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice



Exercise 1:
Introduction to Autoencoders using the MNIST dataset