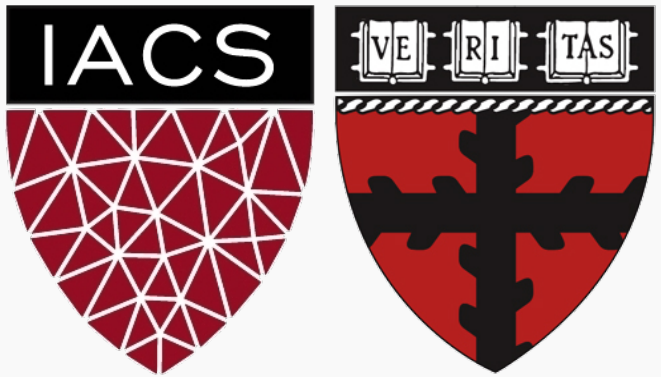
Lecture 18: Autoencoders

CS109B Data Science 2

Pavlos Protopapas and Mark Glickman



- "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**
- Unsupervised/Predictive Learning (cake)
  - The machine predicts any part of its input for any observed part.
  - Predicts future frames in videos
  - **Millions of bits per sample**
- (Yes, I know, this picture is slightly offensive to RL folks. But I'll make it up)

Original LeCun cake analogy slide presented at NIPS 2016, the highlighted area has now been updated.

How Much Information is the Machine Given during Learning?

Y. LeCun

- **"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**

LeCun updated his cake recipe last week 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.

Outline

- Quick review of representation learning
- Brief history of encoding/decoding
- Autoencoder Variations:
  - Convolutional Autoencoders
  - Denoising Autoencoders
  - Sparse Autoencoders

Quick Review. Neural Networks as function approximation.

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

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

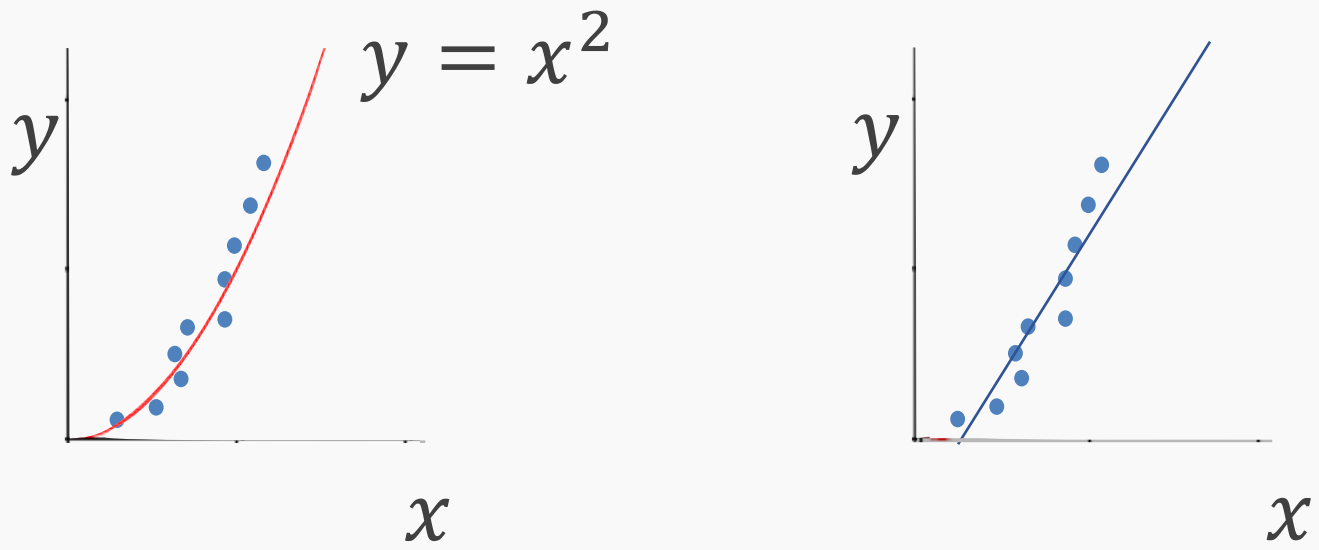
Our goal is to find an estimate of $f(x)$ which we will call $\hat{f}(x)$.

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 $\hat{f}(x)$.

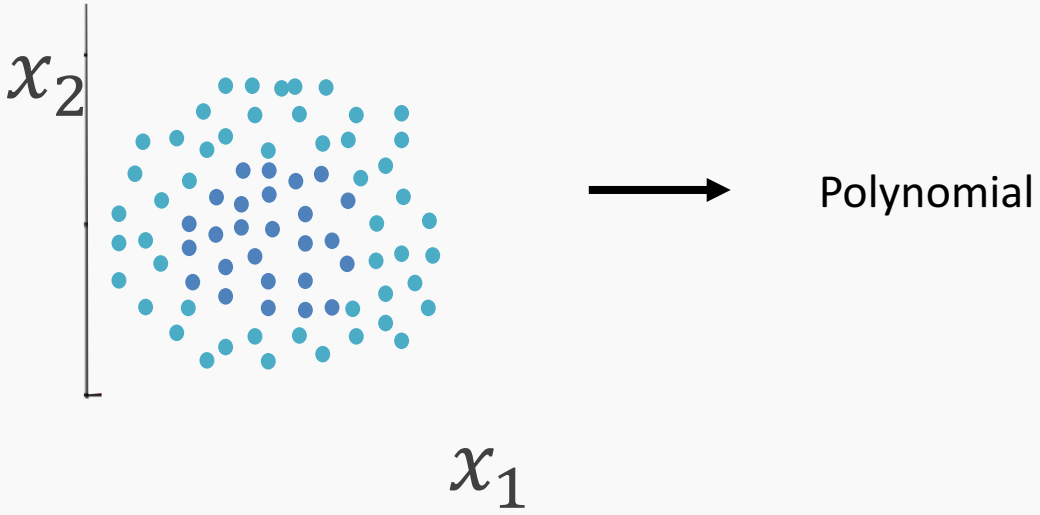
Representational Learning

Representation Matters



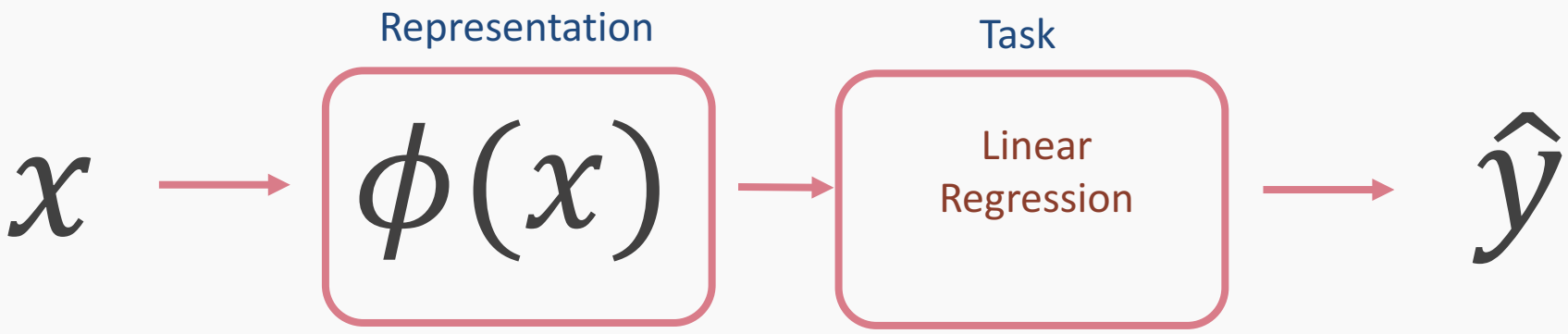
Representational Learning

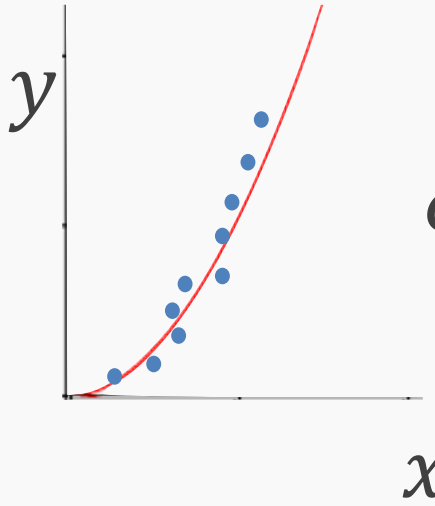
Representation Matters



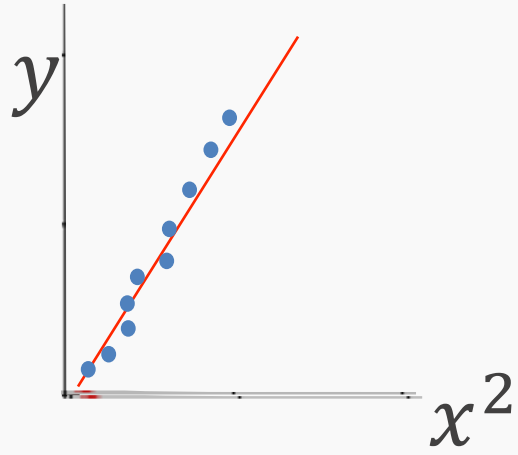
Representational Learning

Representation Matters



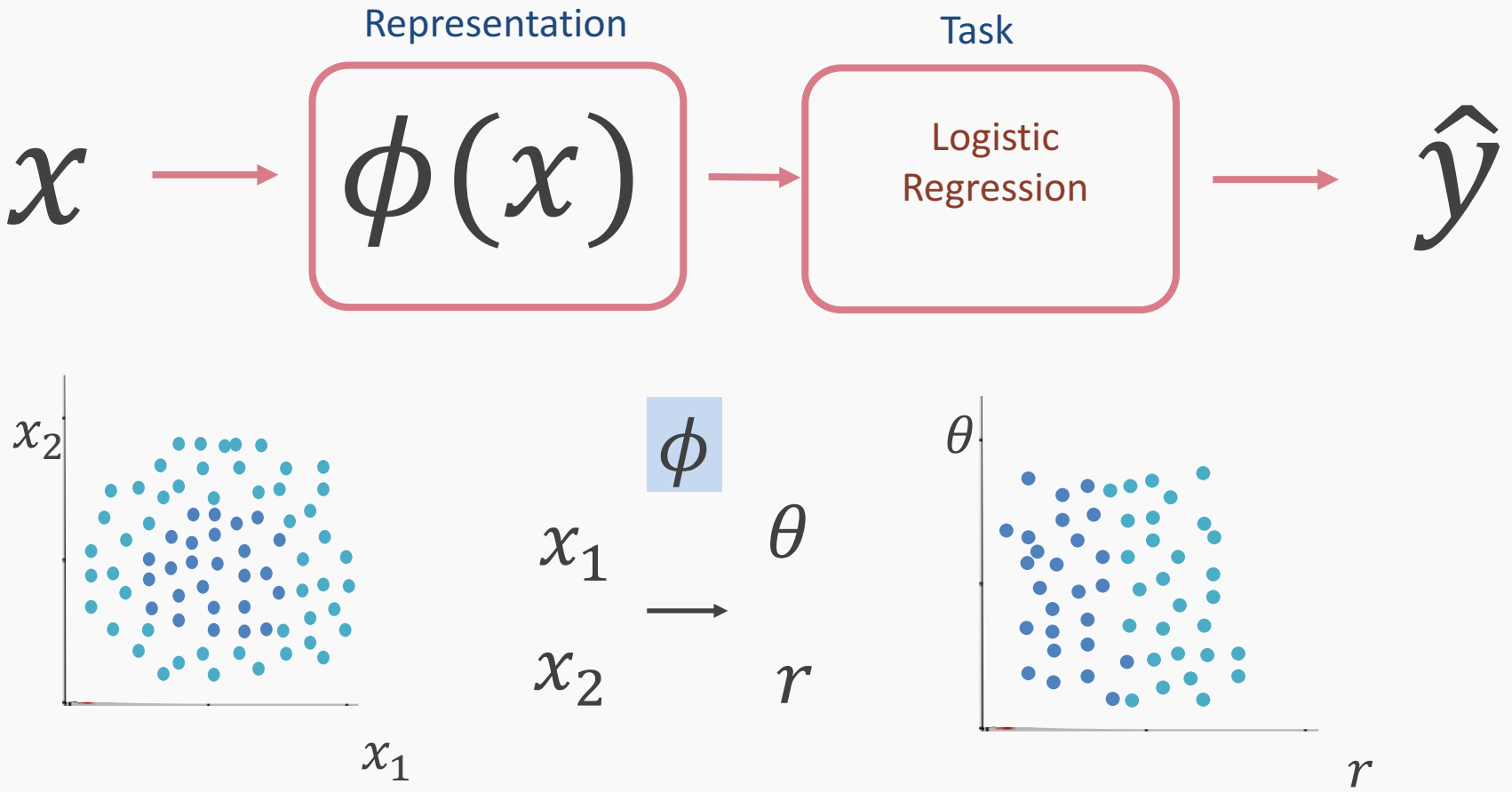


$$\phi(x) = x^2$$



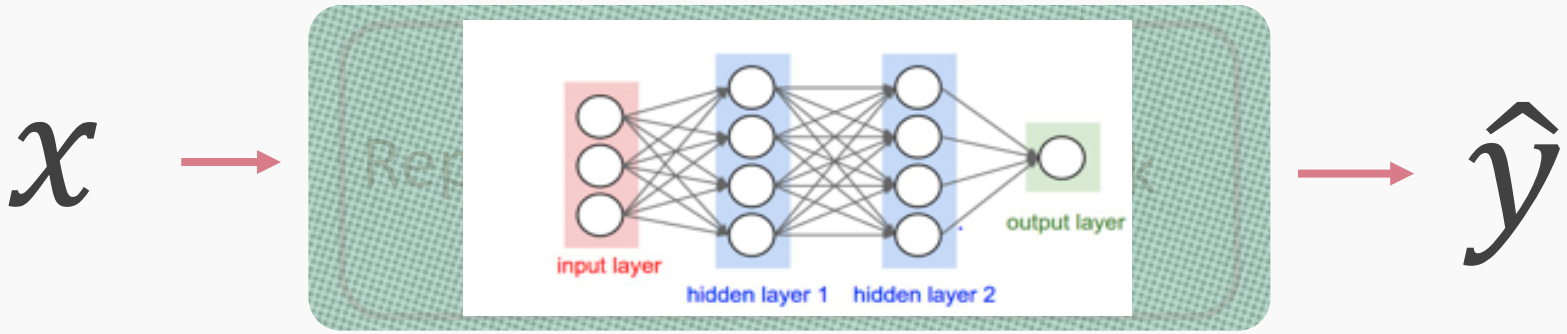
Representational Learning

Representation Matters

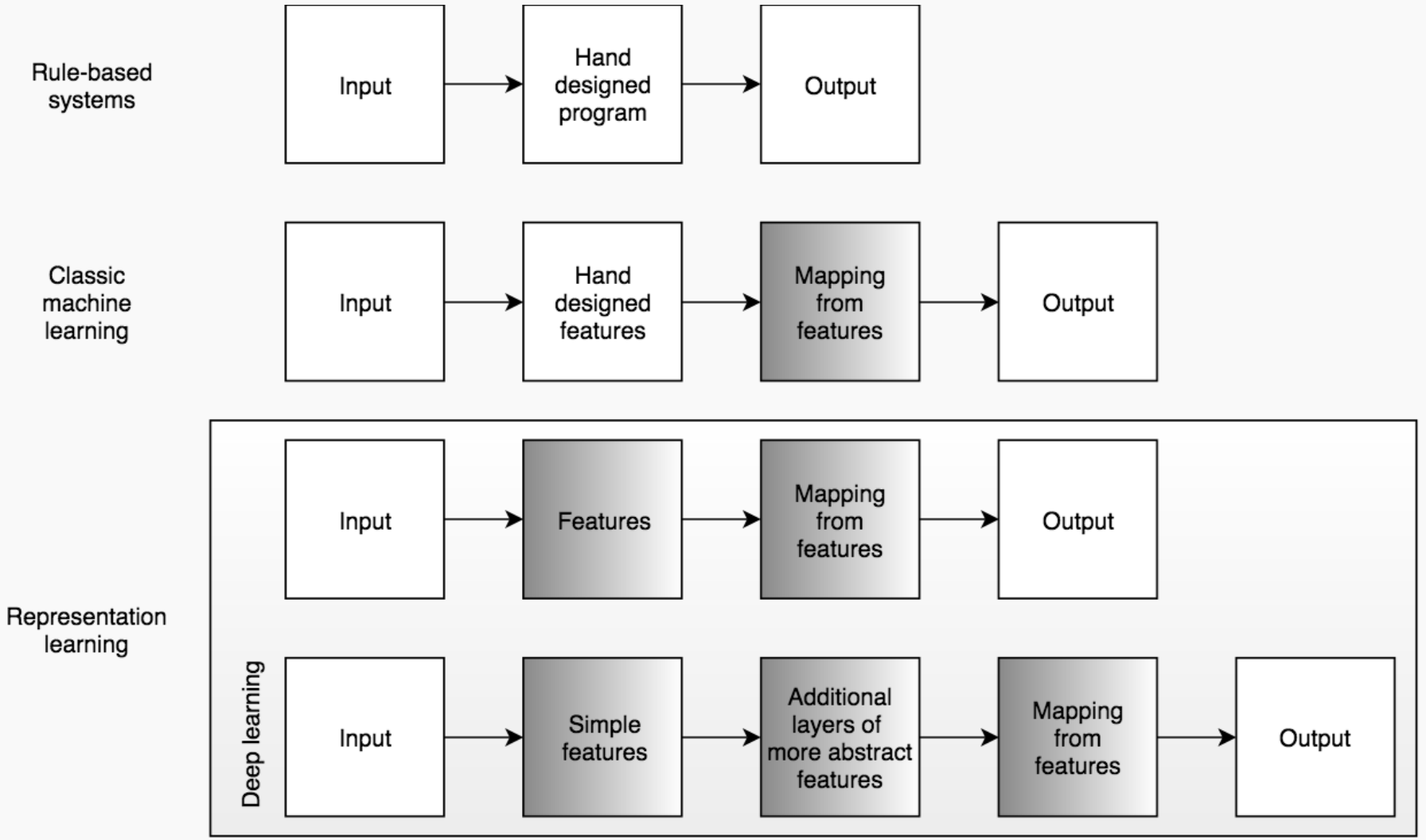


Representational Learning

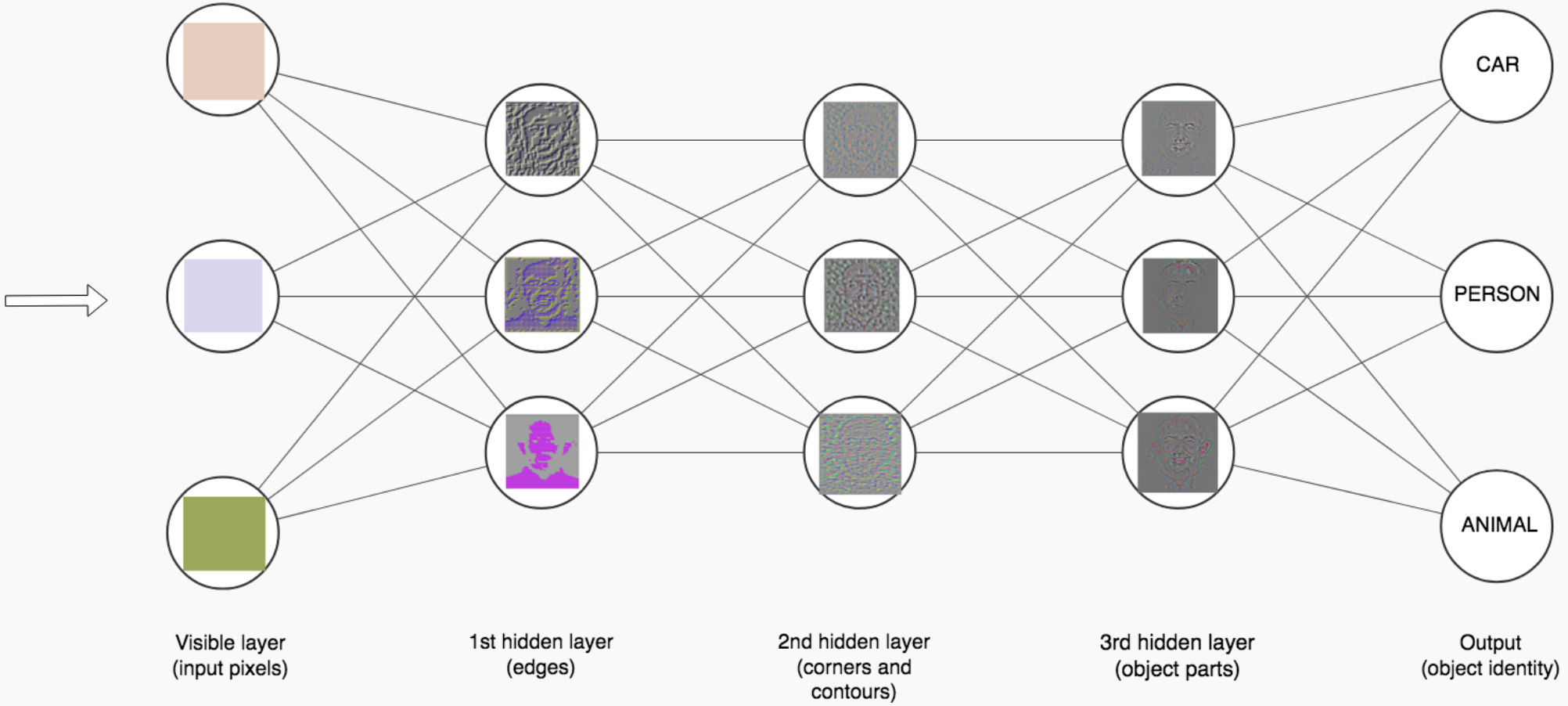
Representation Matters



Representational Learning (cont)

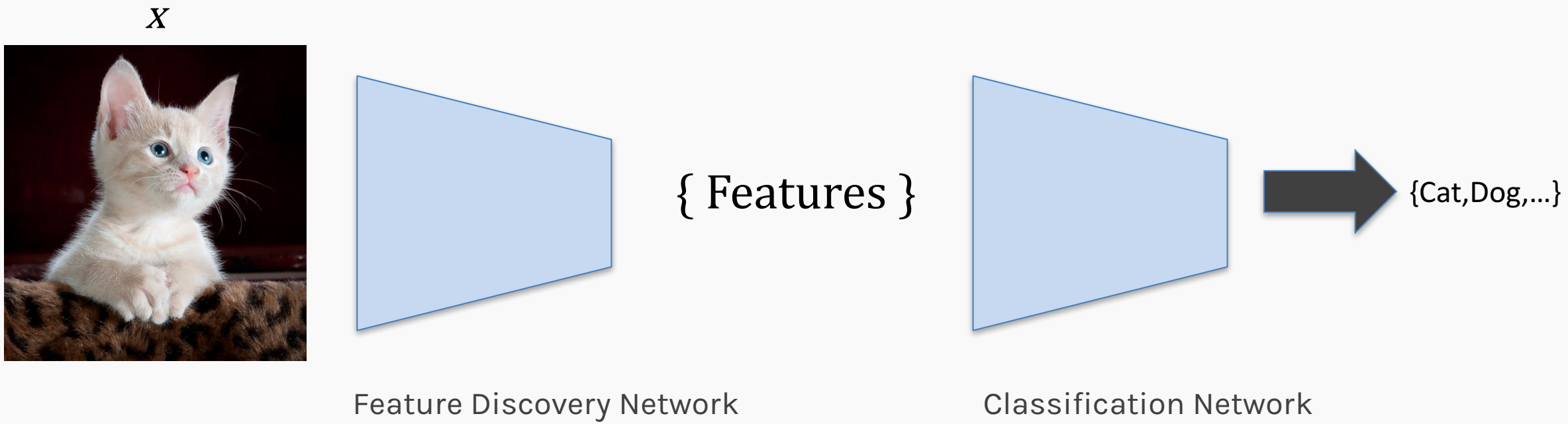


Representational Learning (cont)



Representational Learning: Supervised Learning

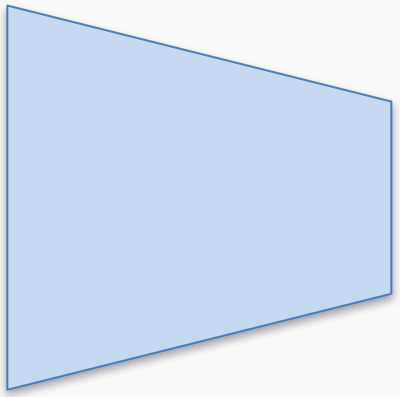
We train the two networks by minimizing the loss function (cross entropy loss)



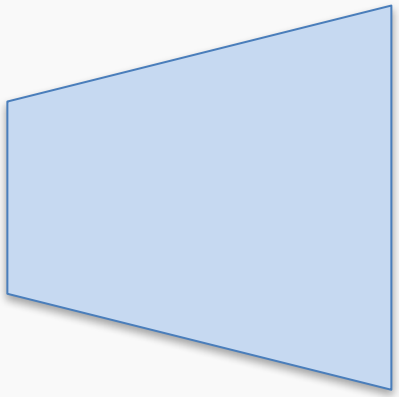
Representational Learning: Self-supervised Learning

We train the two networks by minimizing the reconstruction loss function: $\mathcal{L} = \sum(x_i - \hat{x}_i)^2$

x



{ Features }



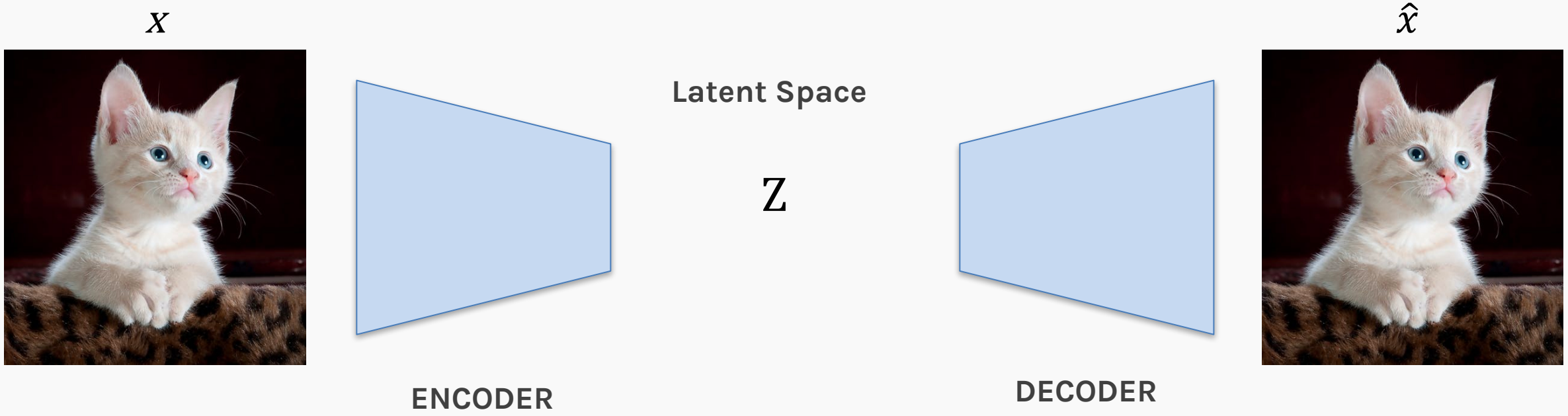
$\hat{x}$

Feature Discovery Network

Second Network

Representational Learning: Self-supervised Learning

AUTOENCODER



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

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)

We say that both MP3 and JPG take an input (a music or image file), and **encode** it into a **compressed** form.

Then we **decode** or **decompress** the intermediate version to some lower quality original version.

What are autoencoders?

- A particular kind of learning architecture
- A mechanism of compressing inputs into a form that can later be decompressed
- Similar to the way MP3 compresses audio and JPG compresses images
- Autoencoders are more general than either MP3 or JPG
- They are usually used to …
  - reduce data dimensionality or find a better suited representation for another task
  - blend inputs from one input to another.
  - denoise, infill, etc.

Lossless and Lossy Encoding

The greater the difference between the original version and the version post-decompression the greater the **loss**.

For example imagine you are in Boston and you want to write a birthday text to a special friend.

HANNAH HAPPY BIRTHDAY I LOVE YOU DAN

In the freezing (-20C) Boston winter you do not want to have your hands out in the open, so you shorten the message as much as possible using text-speak:

H HBD ILY D

Your 36 characters message is compressed to 11 characters

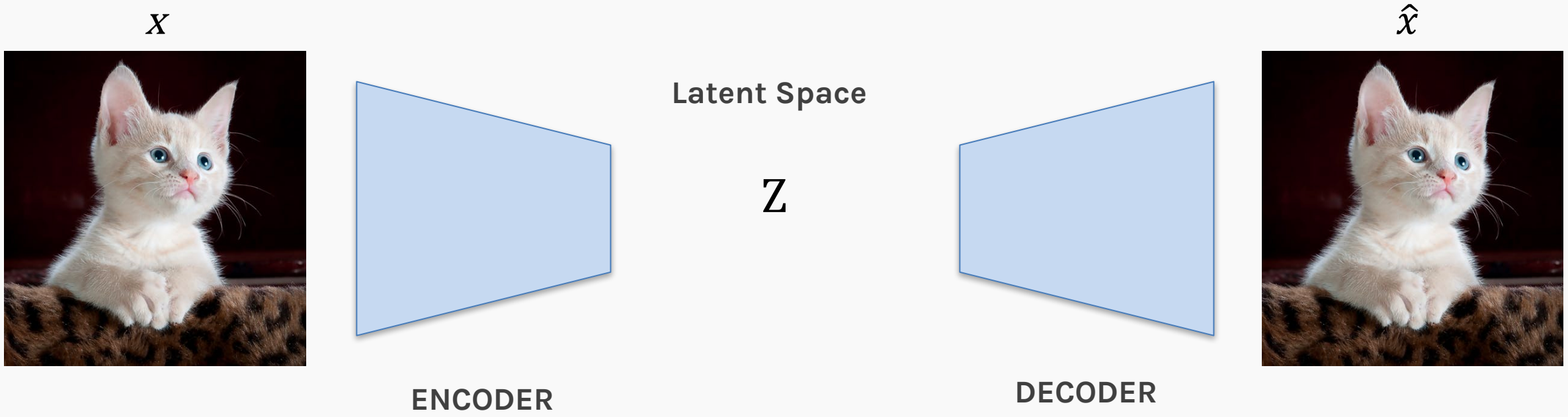
Lossless and Lossy Encoding (cont)

HPD is unambiguous given that it is her birthday today, but **D** could mean **Dan or David or Donny …** You can imagine the potential drama. (Is this an example lossy or lossless compression?)

A way to test if a transformation is **lossy** or **lossless** is to consider if it can be inverted, or run backwards, to provide us with the original data.

Autoencoder

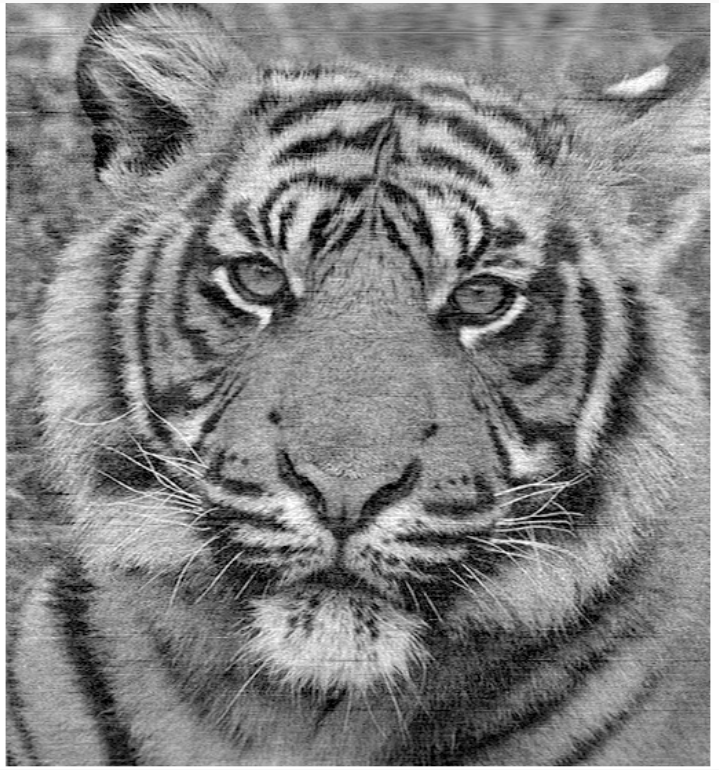
We train the two networks by minimizing the reconstruction loss function: $\mathcal{L} = \sum(x_i - \hat{x}_i)^2$



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.

MP3 and JPG Image Compression

original



MP3

JPG

Original image 256x256=262,000, MP3=37,000, and 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

The simplest autoencoder

Encode with a simple fully connected network (FCN)

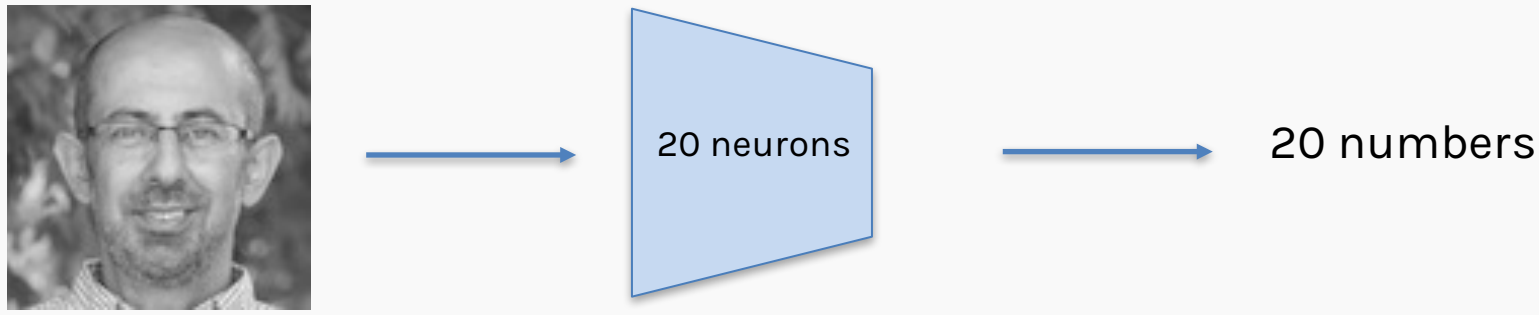
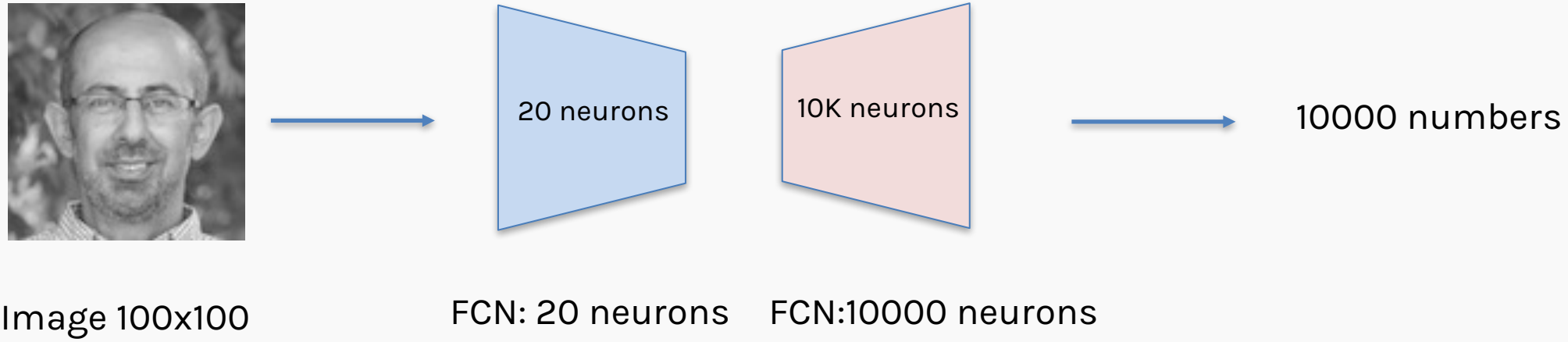


Image 100x100

FCN with 20 neurons

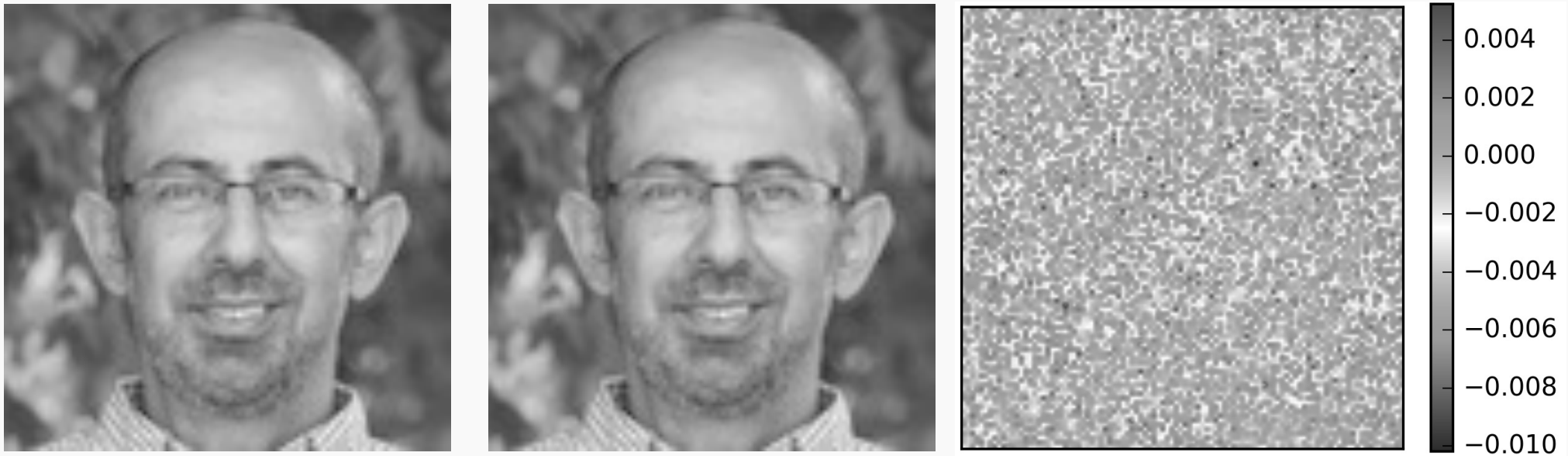
The simplest autoencoder

Encode and decode



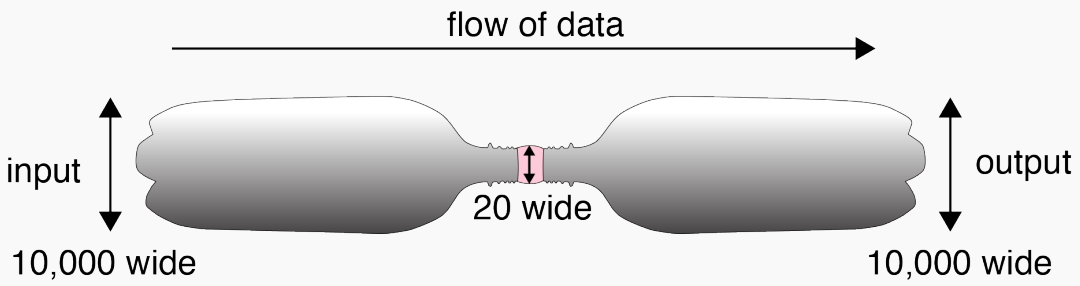
Autoencoders in action

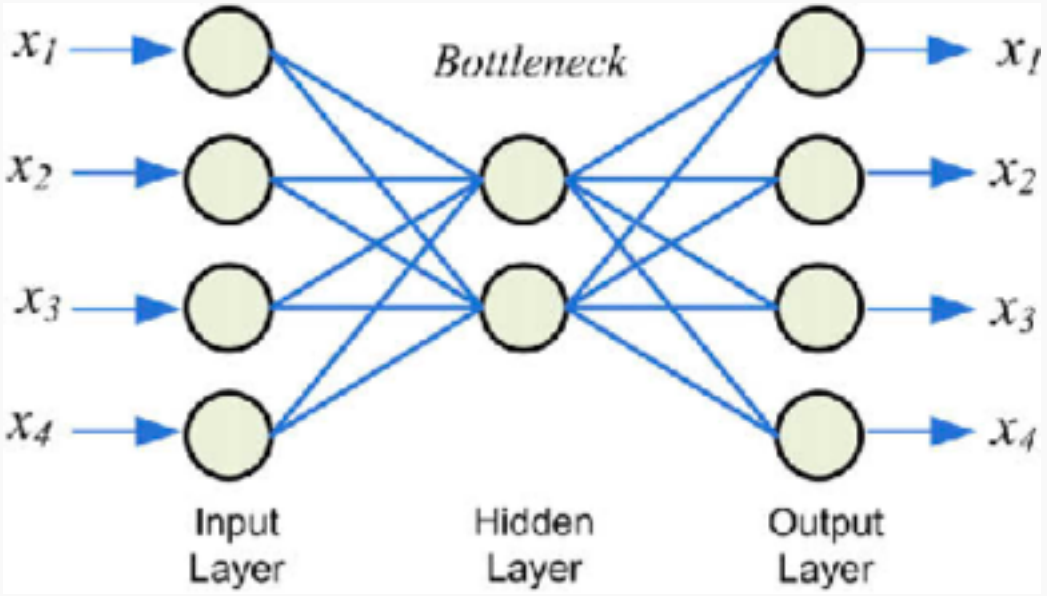
Comparing the input and output pixel by pixel.



Bottleneck

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



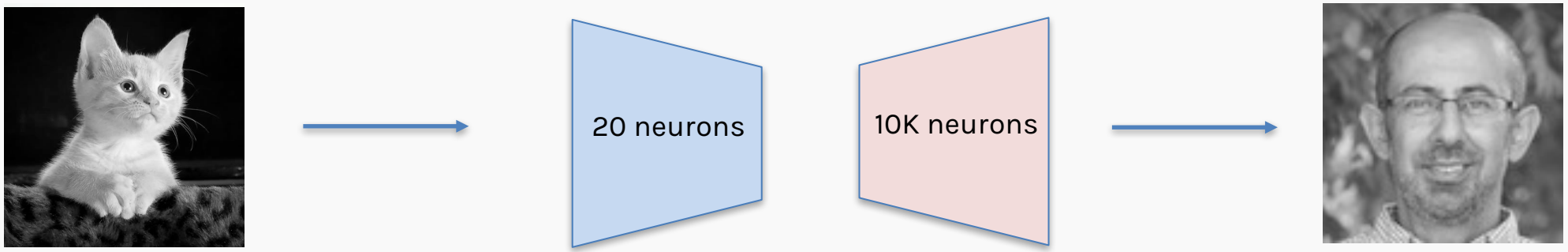


Latent variables and latent layer

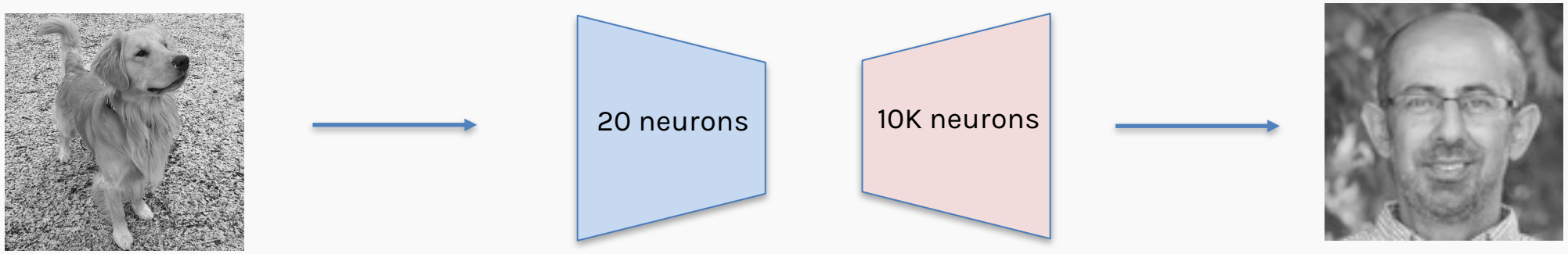
We say that an autoencoder is an example of **semi-supervised or self-supervised** learning.

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 isn't supervised learning because we don't have any manually determined labels or targets on the inputs.

Autoencoders in action (cont)

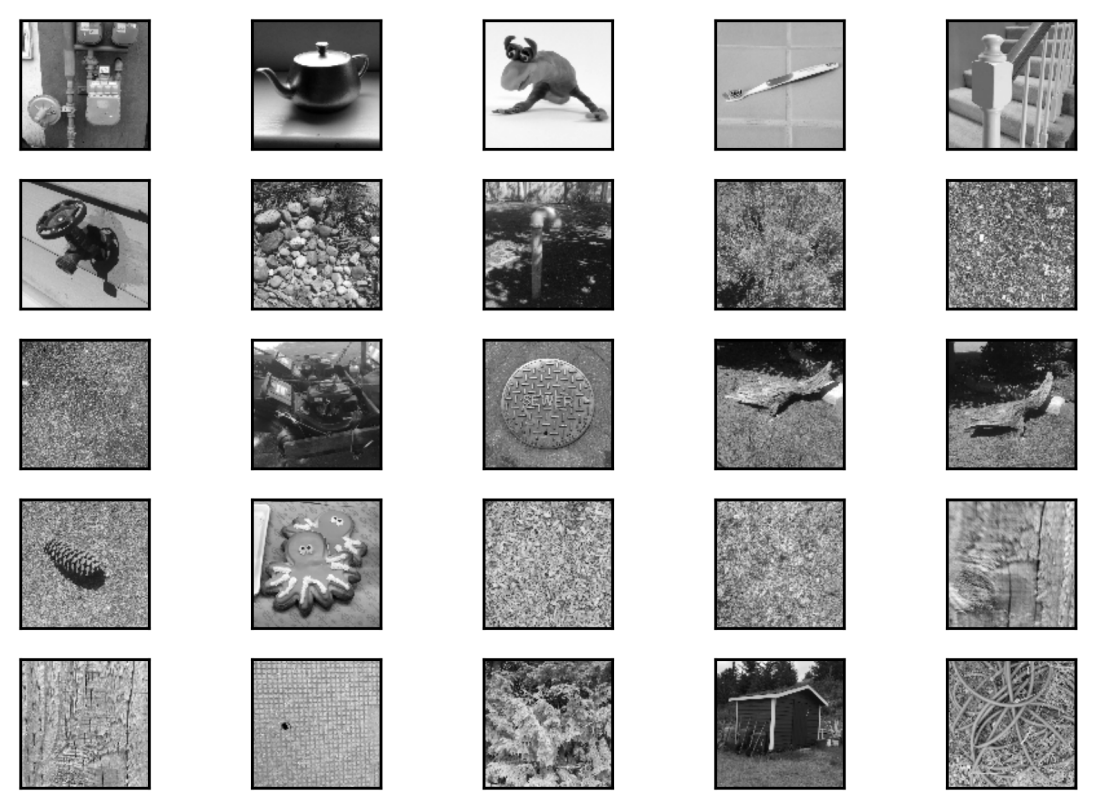


Autoencoders in action (cont)



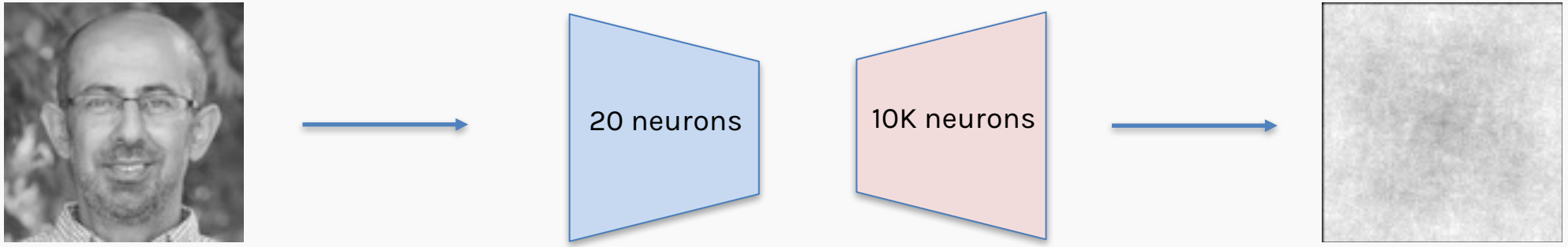
Autoencoders in action (cont)

We must train with a variety of images.

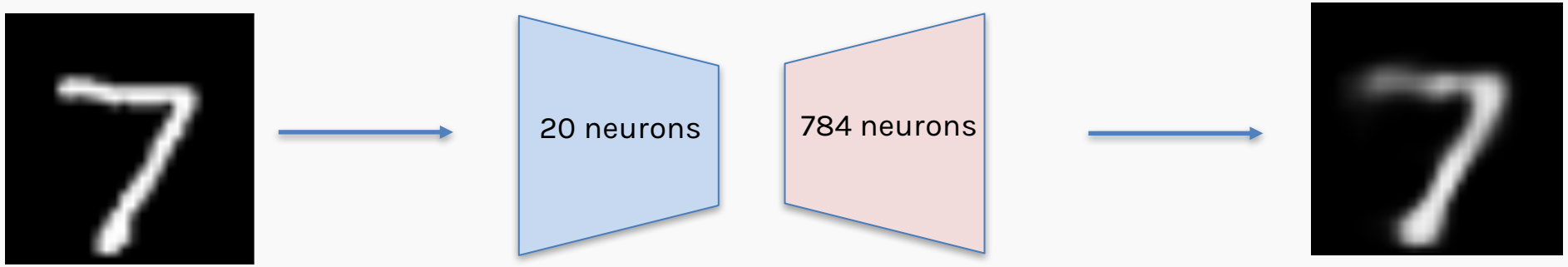


Autoencoders in action (cont)

Testing it again



A Better autoencoder



20 latent variables

original

reconstructed

10 latent variables

original

reconstructed

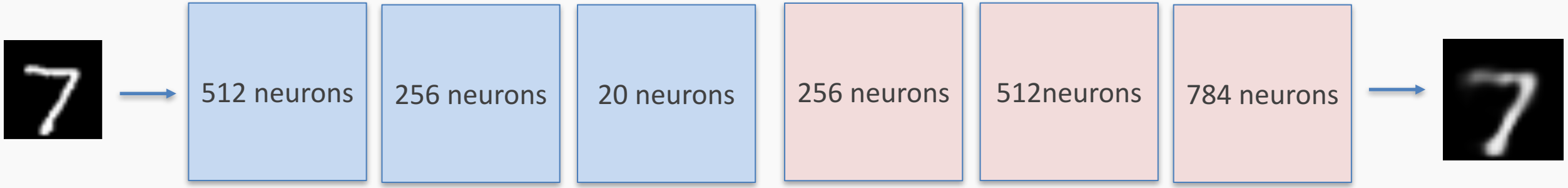
2 latent variables

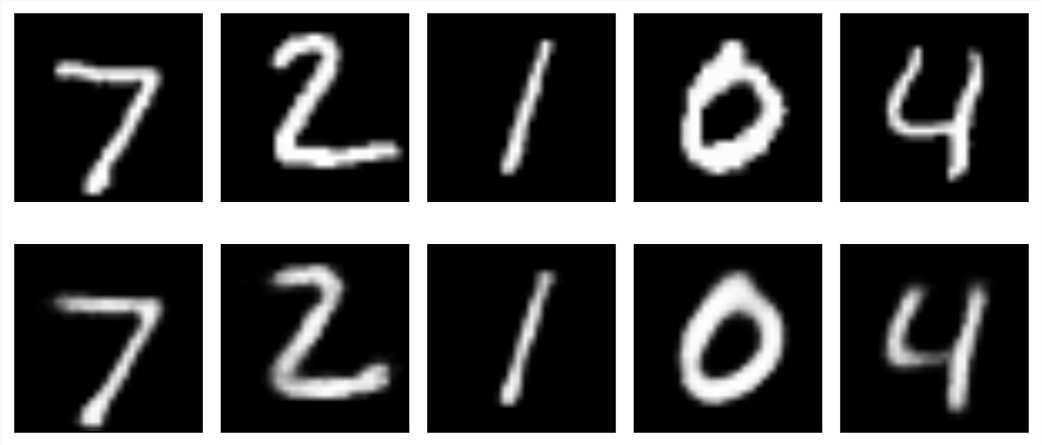
original

reconstructed

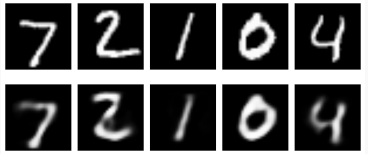
Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Deeper

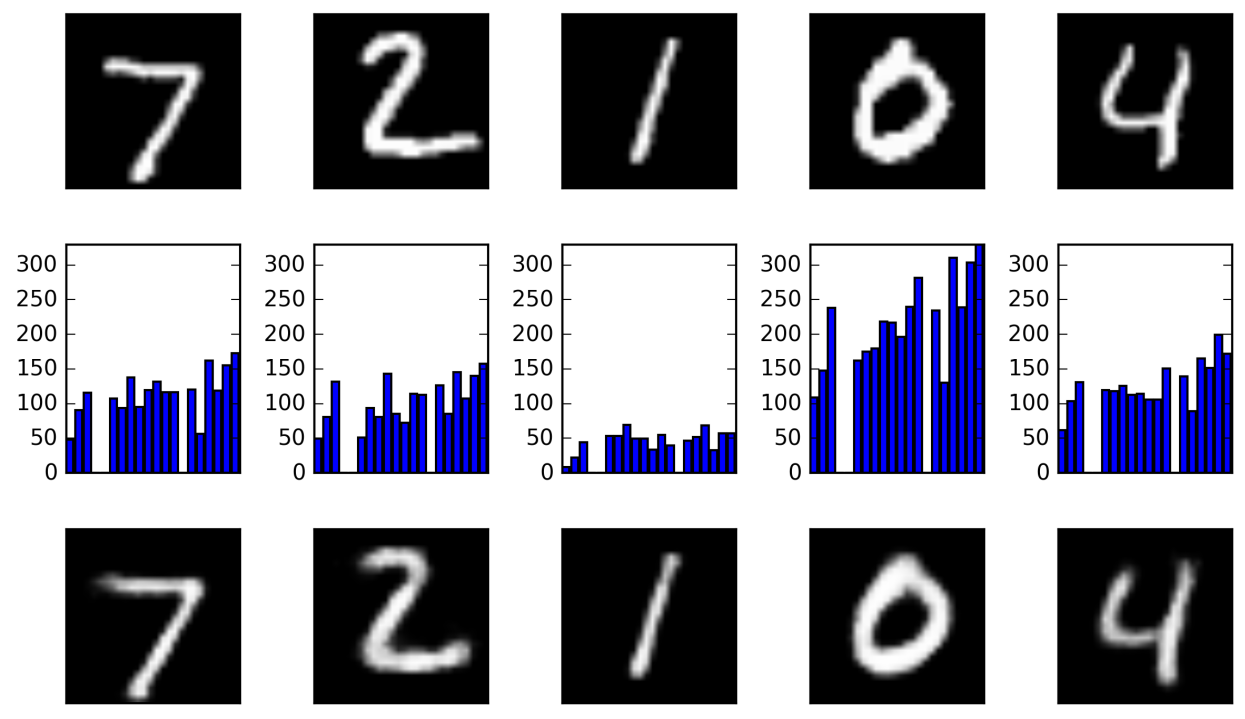




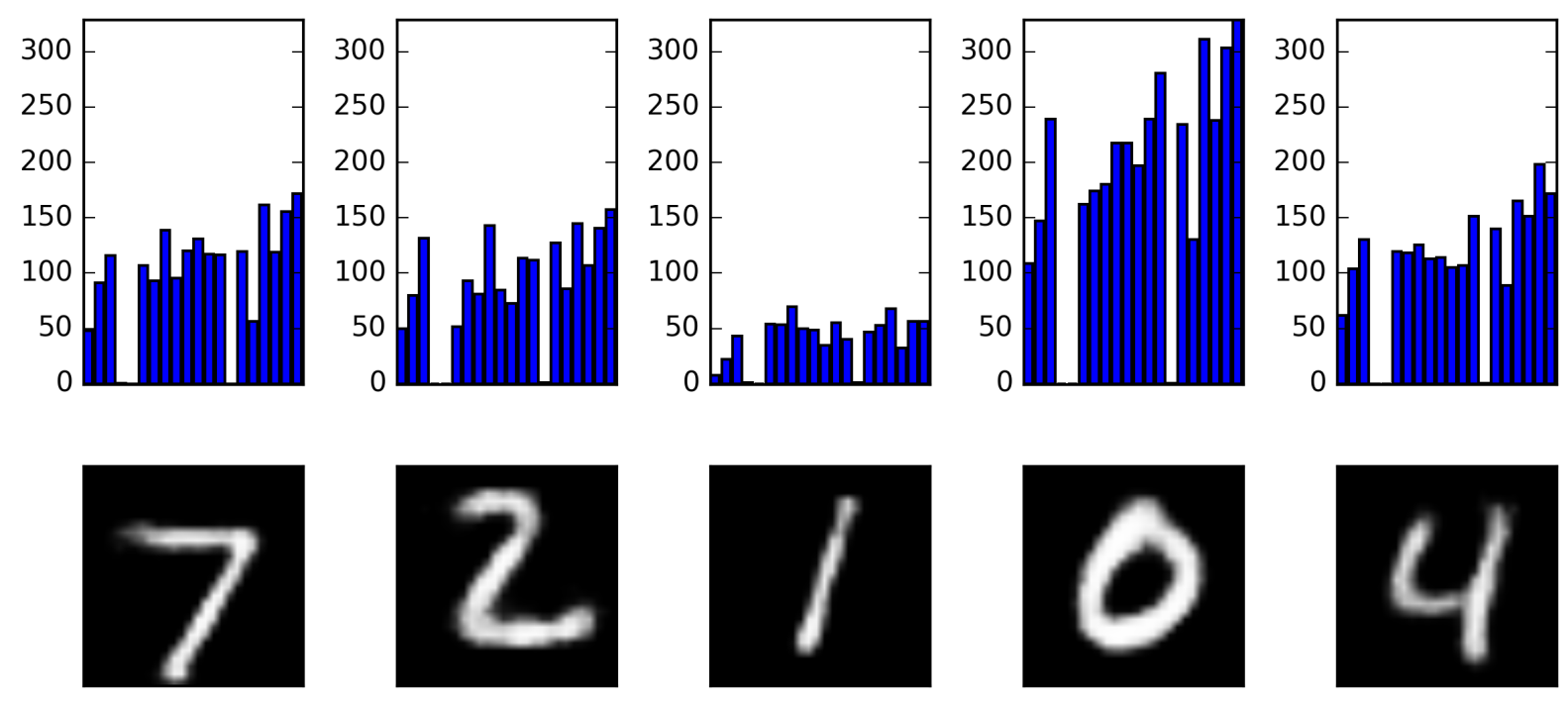
Shallow 20 latent variables



Exploring autoencoders

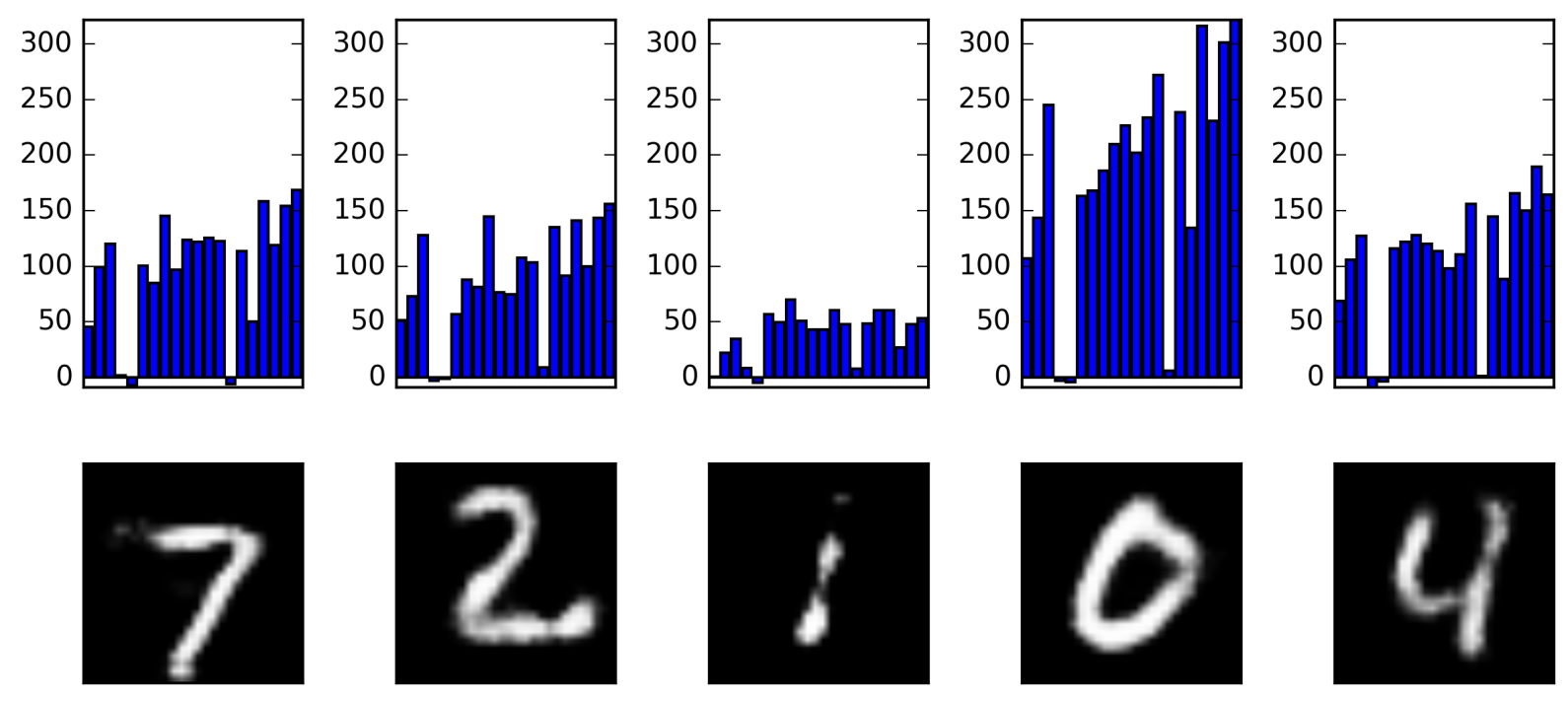


Exploring autoencoders (cont)



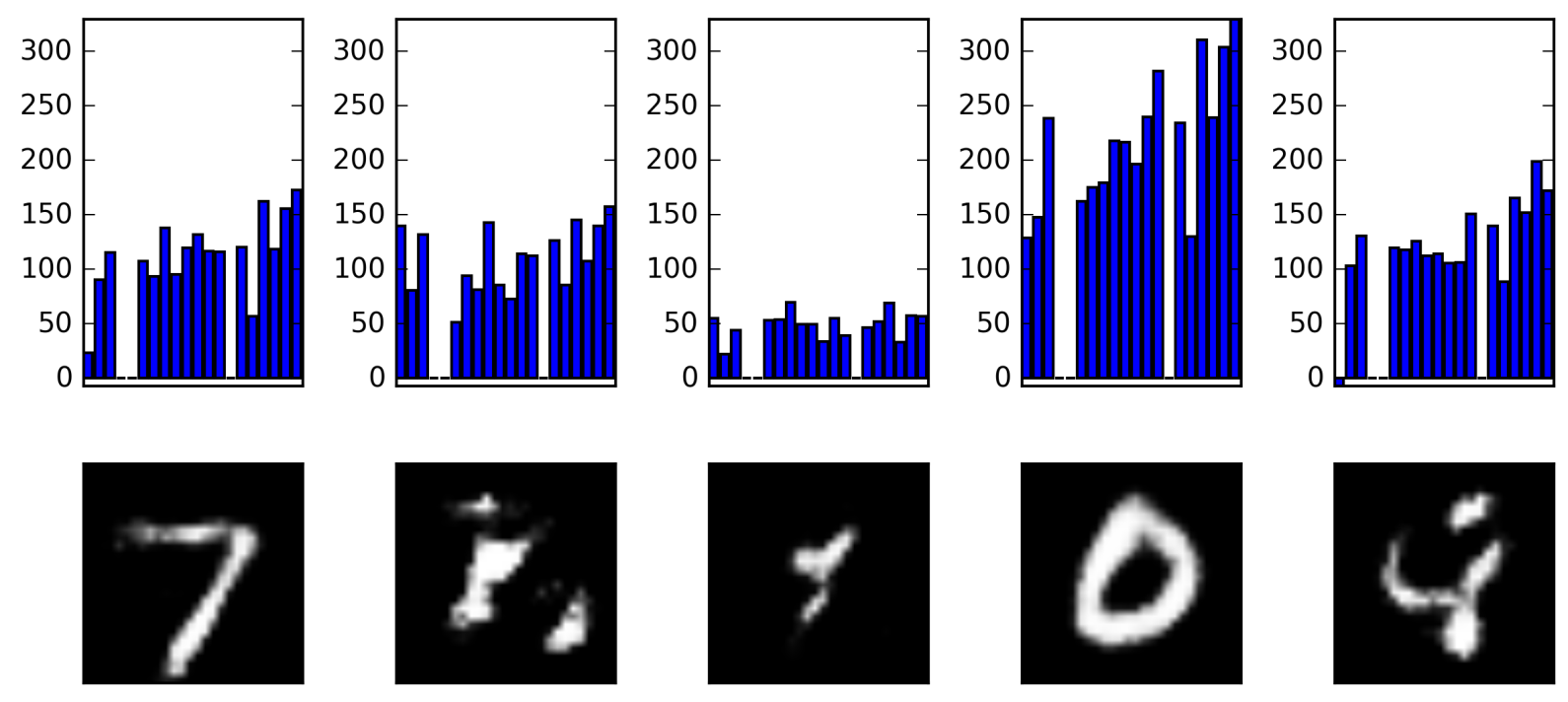
Change +/- 1 the latent variables

Exploring autoencoders (cont)



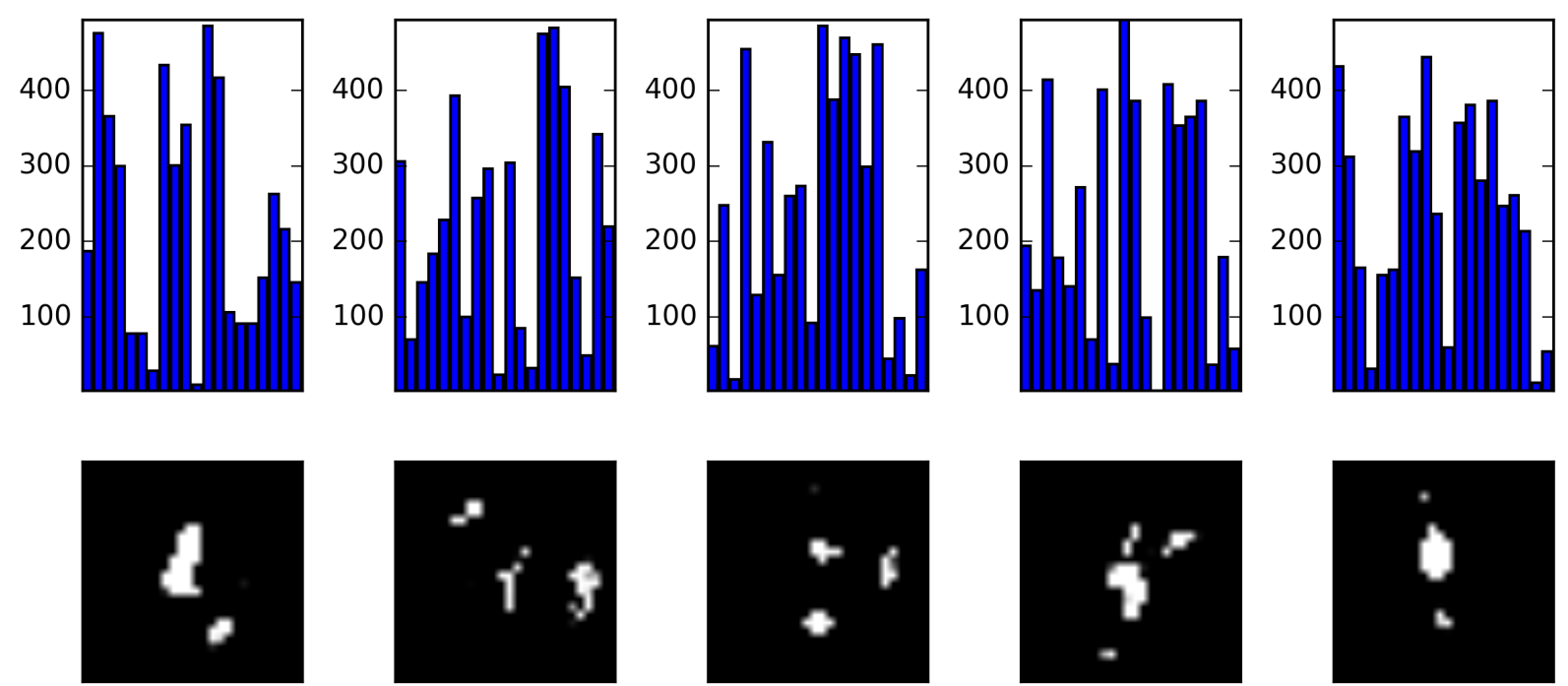
Change +/- 10 the latent variables

Exploring autoencoders (cont)



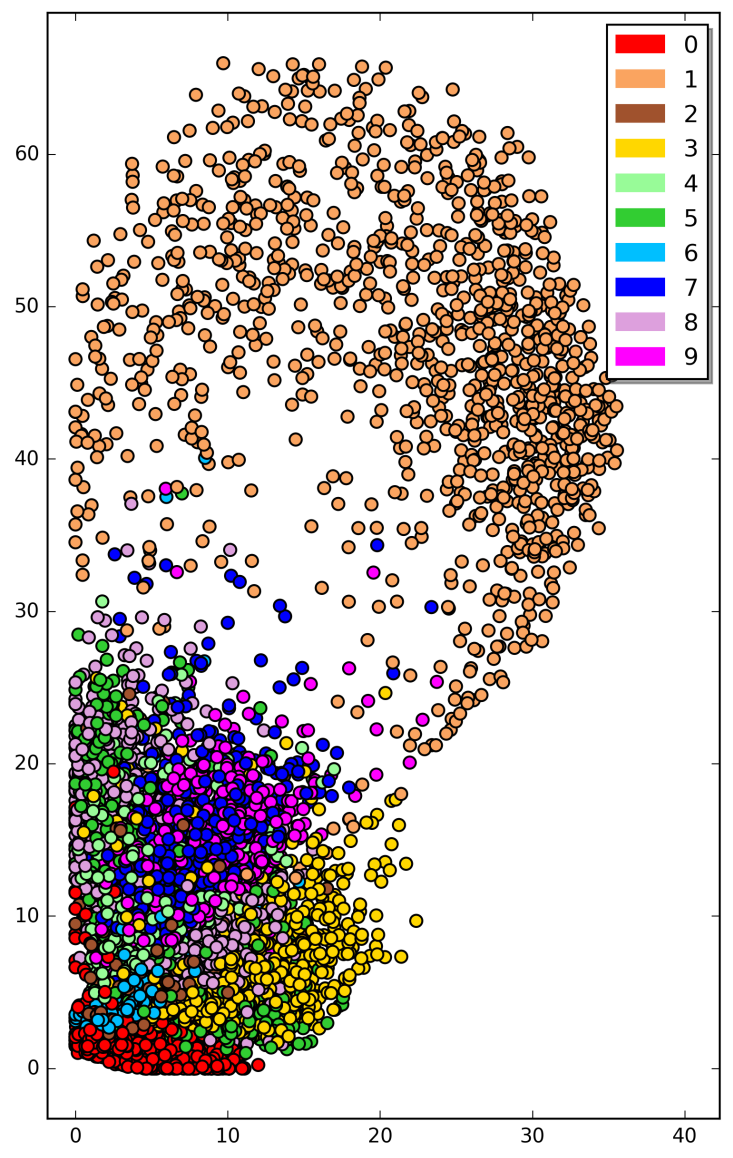
Change +/- 100 the first latent variables

Exploring autoencoders (cont)



Just noise

Parameter space of autoencoder



Blending

We blend inputs to create new data that is similar to the input data, but not exactly the same.

One example of blending is **content blending** where the content of two pieces of data is directly blended. An example is if we overlay images of a cow and zebra.



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

Blending (cont)

Another type of blending is **parametric** or **representation** blending:

In this type of blending we take advantage of parameterization to describe the objects we're interested in. By engaging in blending in the parameter space, we can create results that blend the inherent qualities of the objects of interest.

Blending (cont)

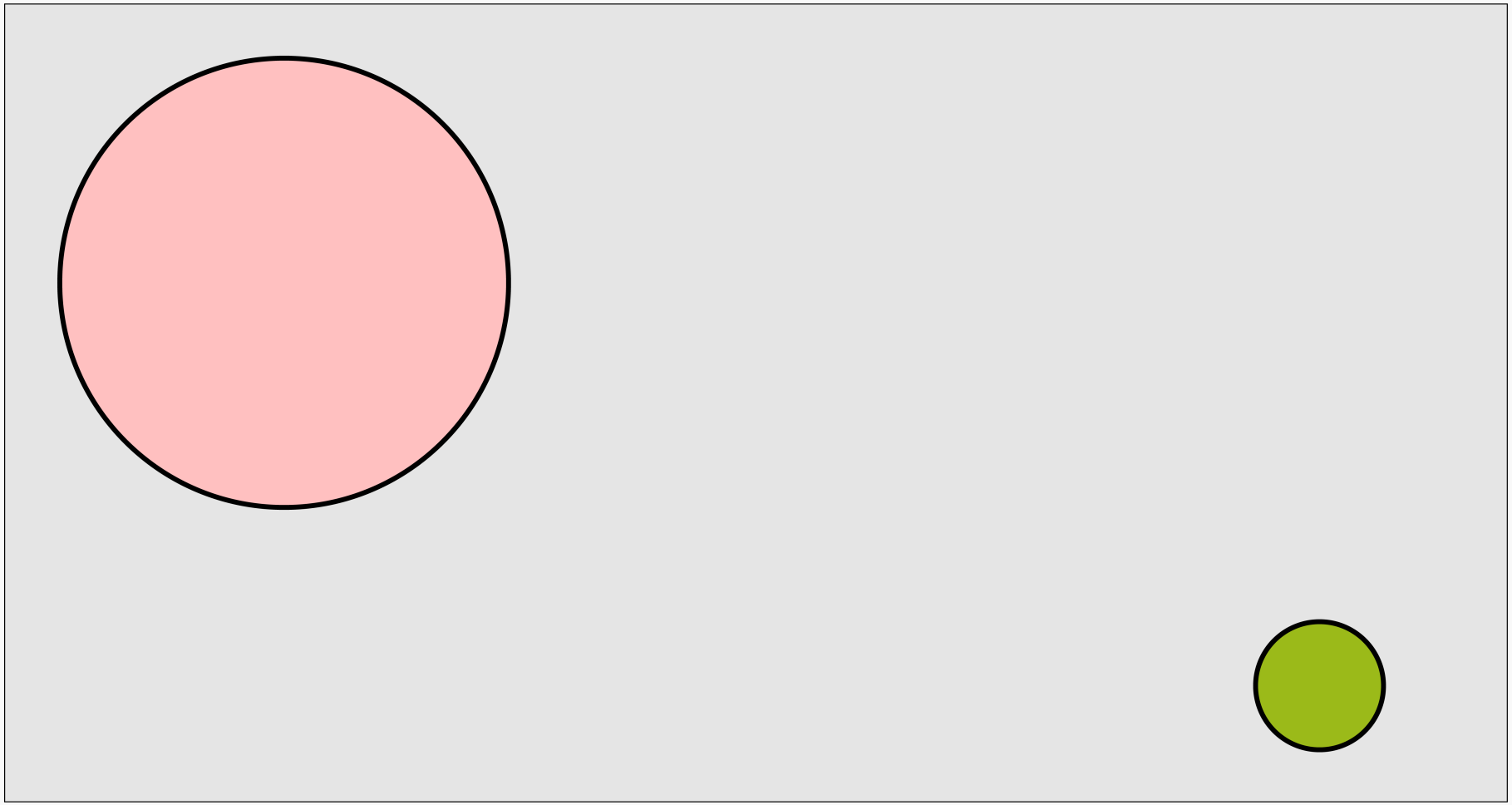


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Content blending

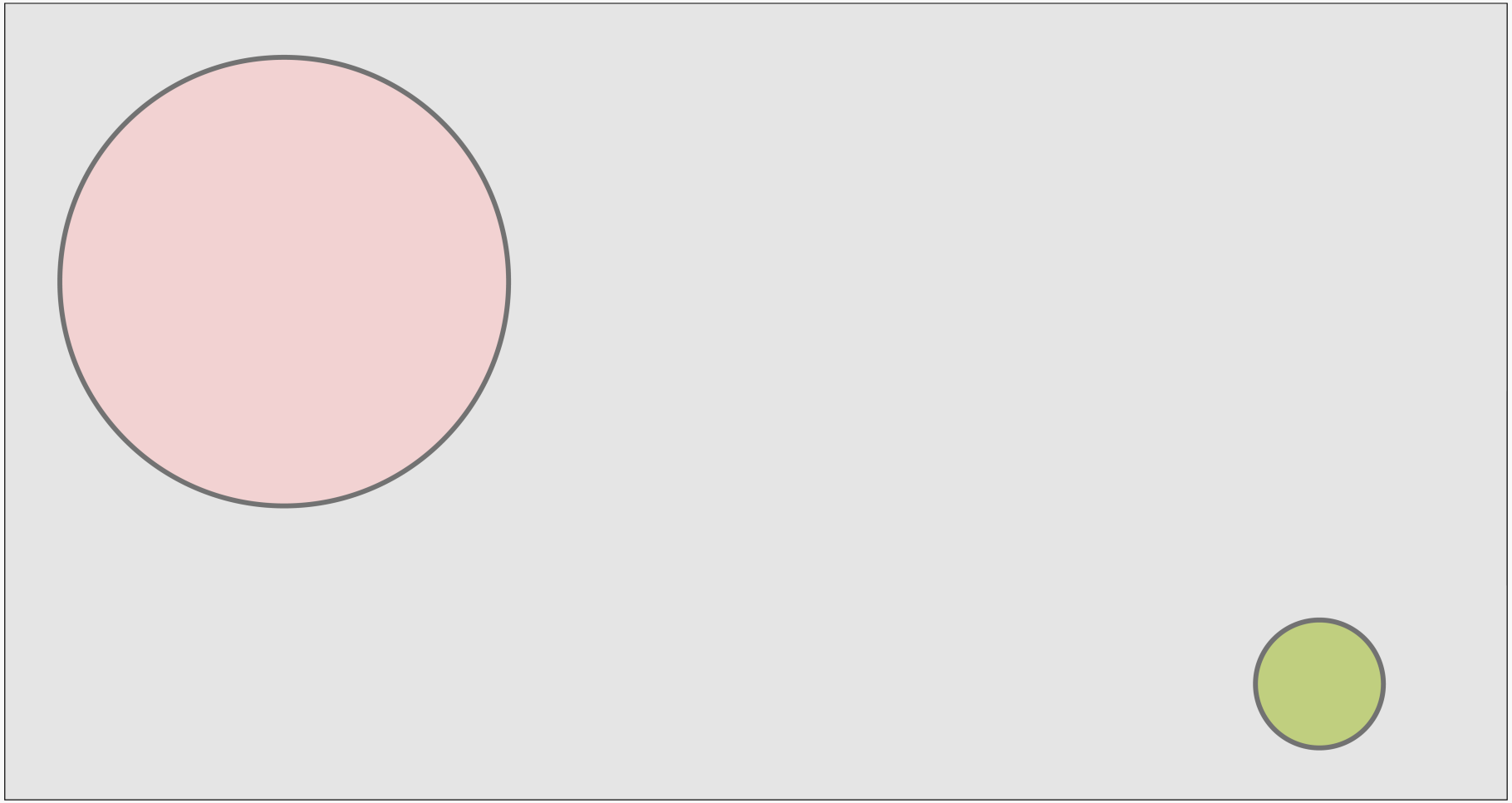


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Parametric or representation blending

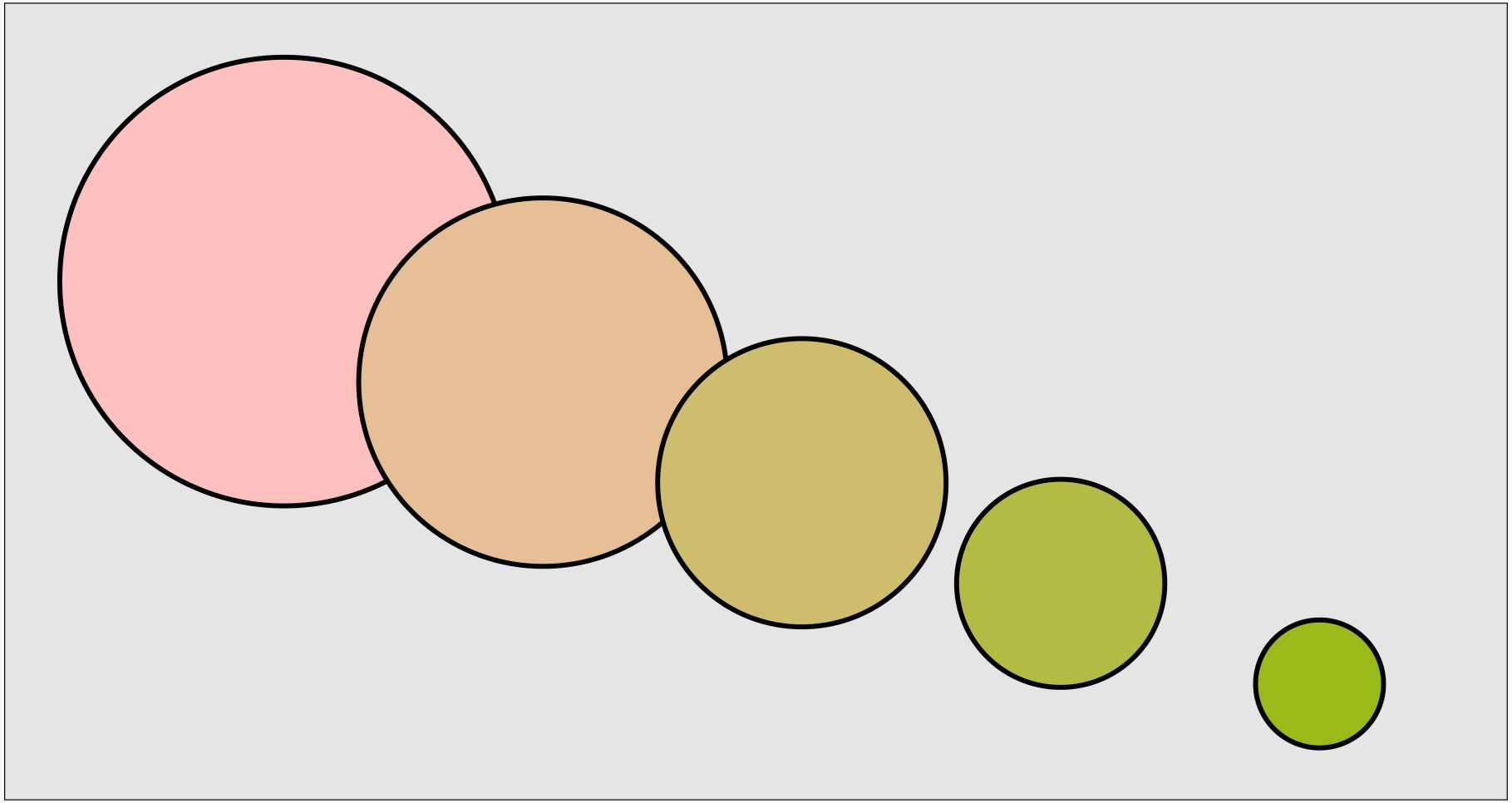


Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice

Blending in compressed representation

While the blending we've described works well for uncompressed objects what happens when compression is involved?

The compressed form may not be the best representation with what we would like to blend the objects.

For example, let's take the sounds of the words *cherry* and *orange*.

We can blend these sounds together or we can *compress them* into written words.

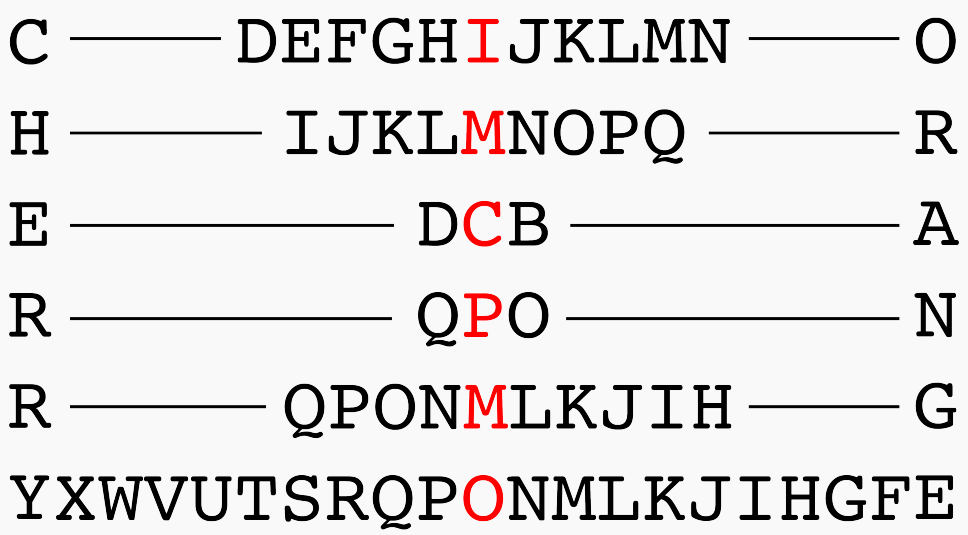
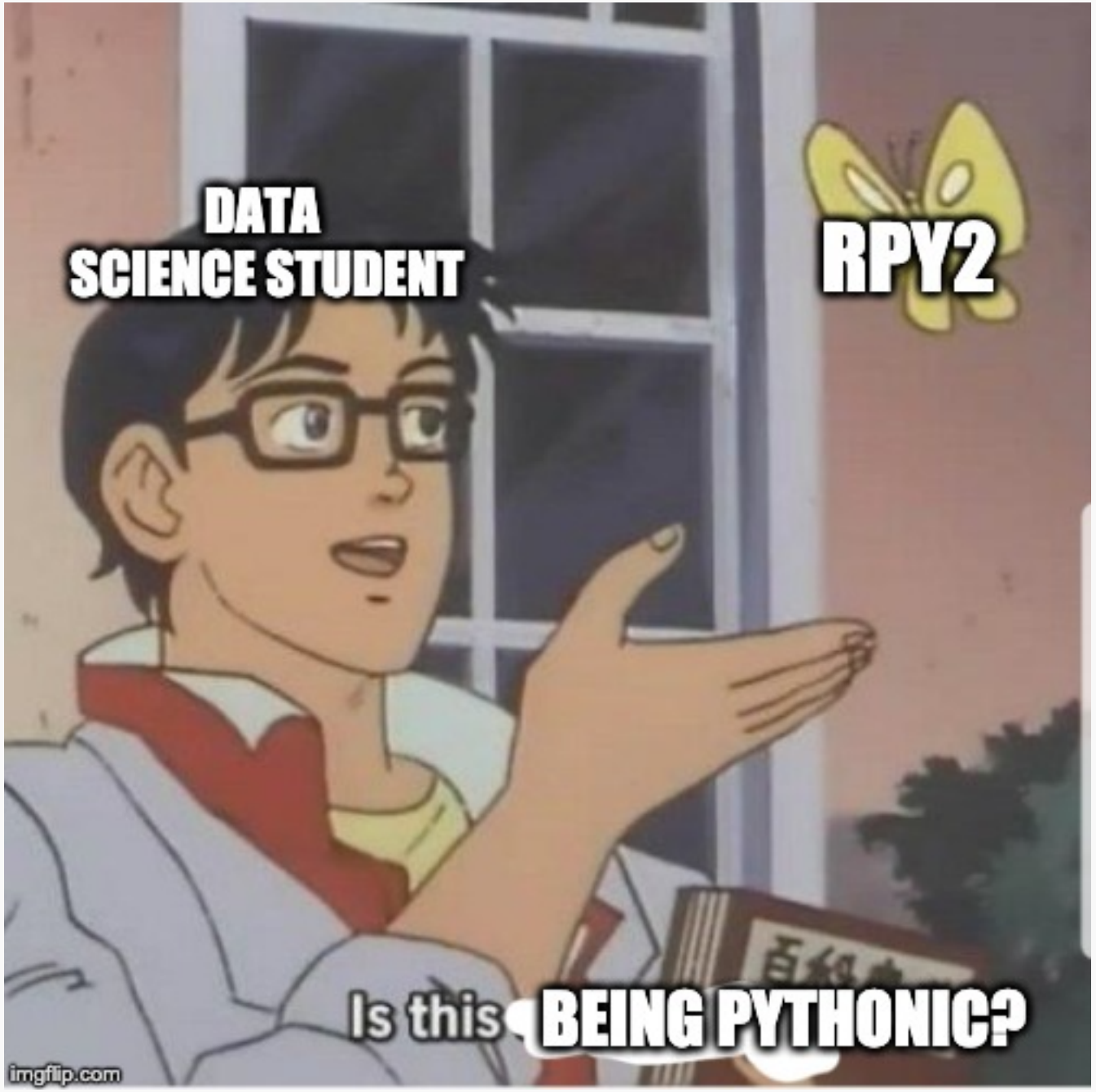
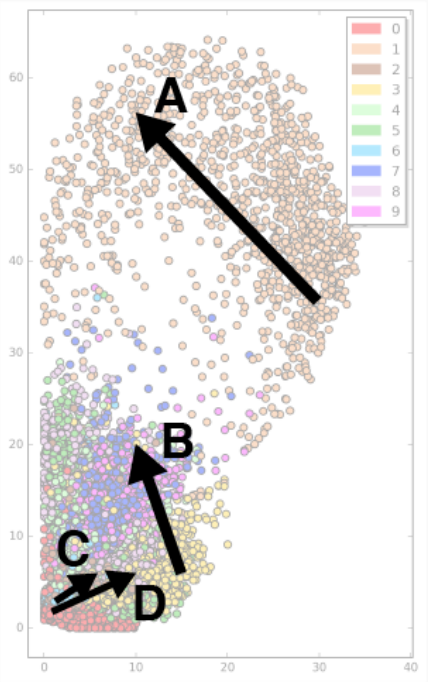


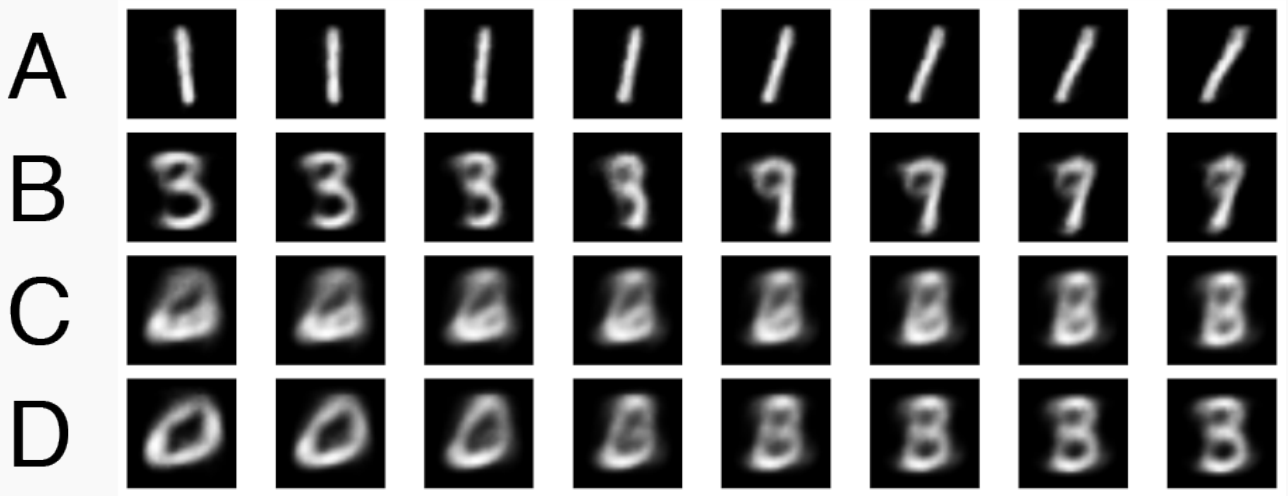
Image taken from A. Glassner, Deep Learning, Vol. 2: From Basics to Practice


DATA
SCIENCE STUDENT
RPY2
Is this BEING PYTHONIC?
imgflip.com

Blending Latent Variables

Back to the example of MNIST

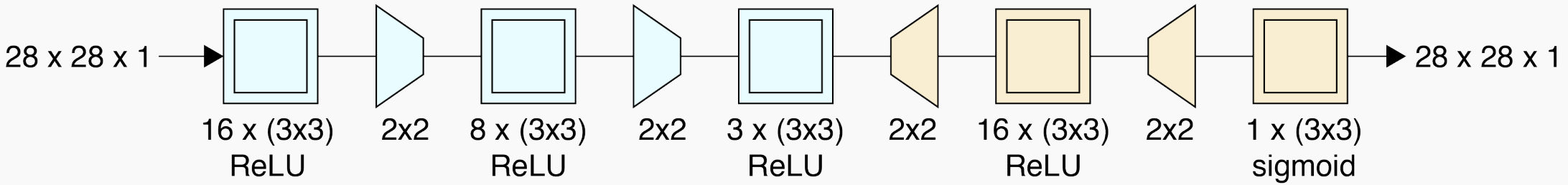


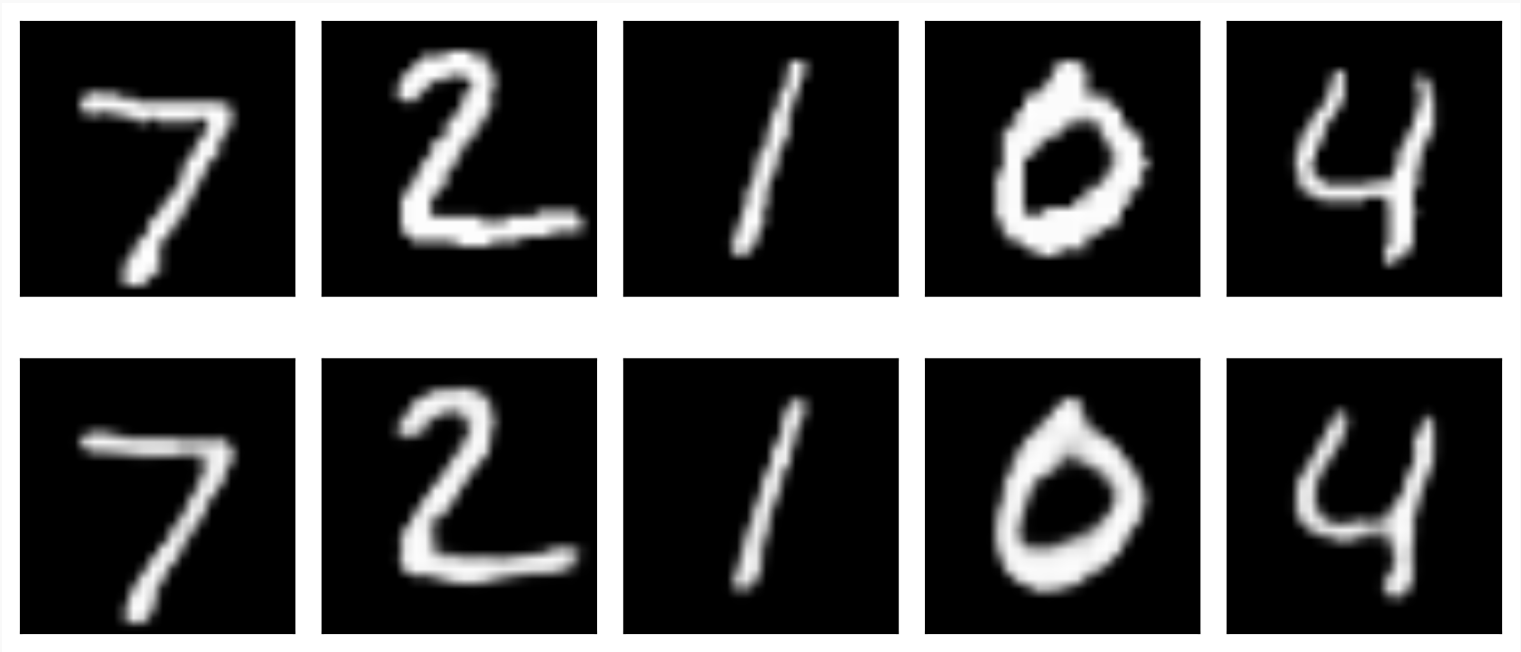


Blending Latent Variables (cont)

Applying to novel input

Convolutional Autoencoders

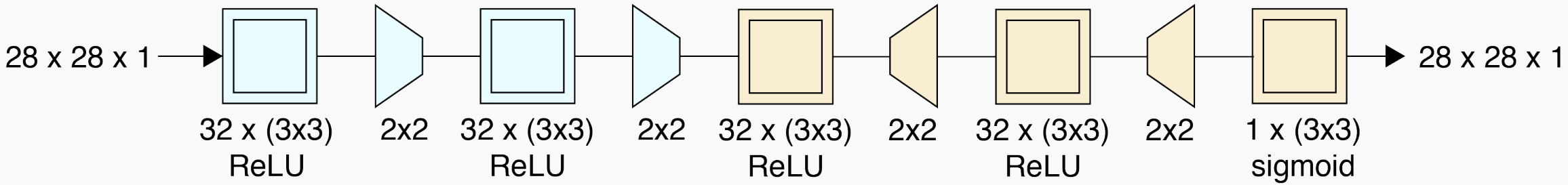




Denoising

A popular use of autoencoders is to remove noise from samples.

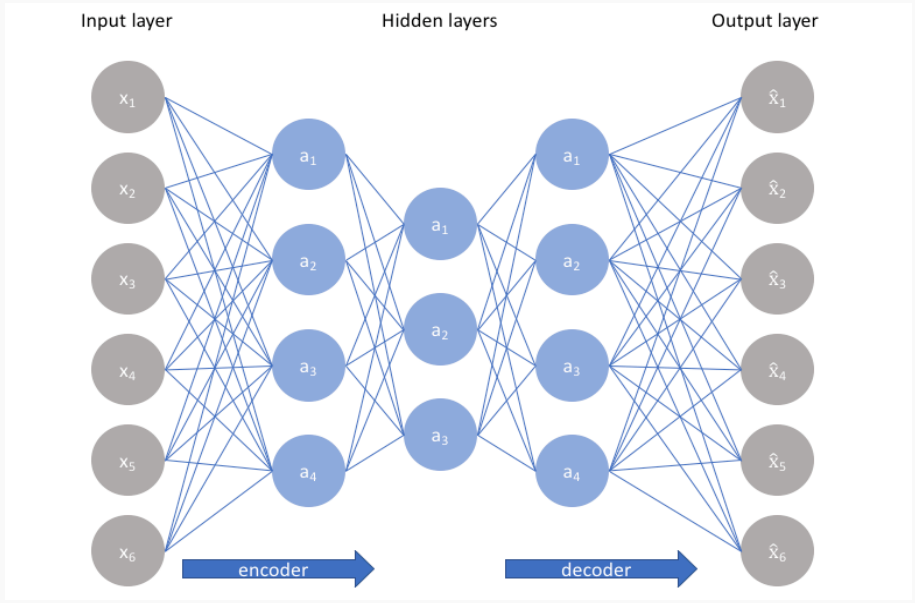
Denoising (cont)

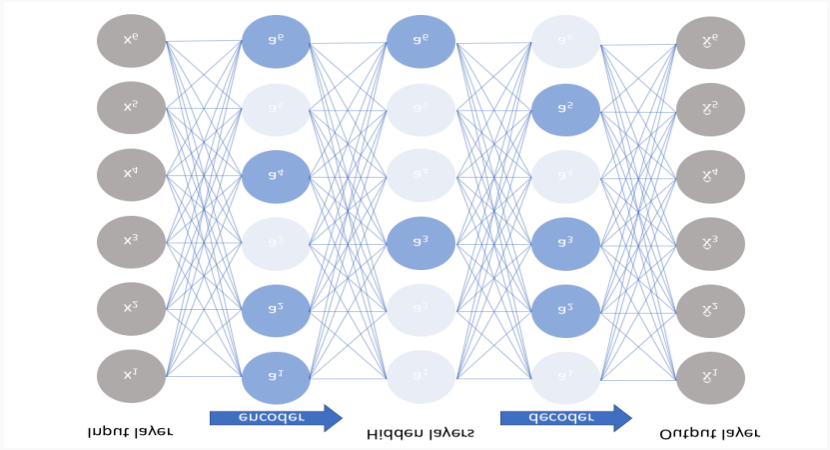


Note that we start with a clean image, add noise and train our networks to return a clean decoded image.

Sparse Autoencoder

- We've assumed so far that the size of the bottleneck is smaller than the size of the inputs – this is called an **undercomplete autoencoder**
- The case in which the size of the bottleneck is greater than or equal to the number of inputs we call an **overcomplete autoencoder**





Sparse Autoencoder (cont)

The size of the bottleneck (i.e. the number of latent variables) makes a difference!

20 latent variables

original

reconstructed

2 latent variables

original

reconstructed

56

Regularized Autoencoders

- Sparse autoencoders
- Denoising autoencoders
- Autoencoders with dropout on z
- Contractive autoencoders

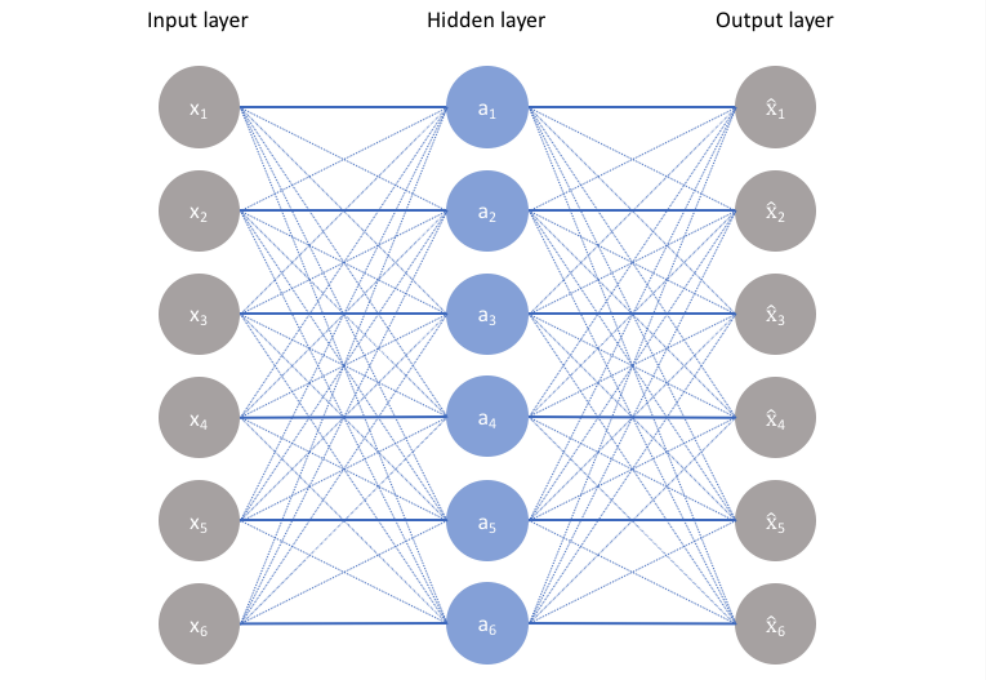
Overcomplete Autoencoders

z has greater dimension than x

Autoencoder may simply copy input to output without learning anything useful

The ideal autoencoder model balances the following:

1. Sensitive to the inputs enough to accurately build a reconstruction.
2. Insensitive enough to the inputs that the model doesn't simply memorize or overfit the training data.



Regularized Autoencoders (cont)

This trade-off requires the model to maintain only the variations in the data required to reconstruct the input without holding on to redundancies within the input.

Question: How to achieve this?

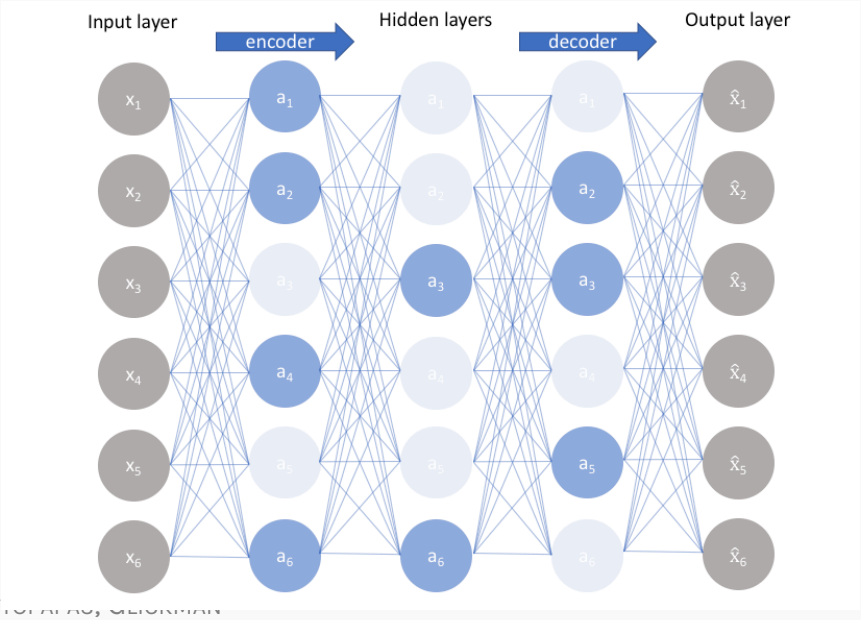
For most cases, this involves constructing a loss function where one term encourages our model to be sensitive to the inputs (ie. reconstruction loss $\mathcal{L}(x, \hat{x})$ and a second term discourages memorization/overfitting (ie. an added regularizer).

$$\mathcal{L}\left(x, g\left(f(x)\right)\right) + \Omega(z)$$

Sparse Autoencoders

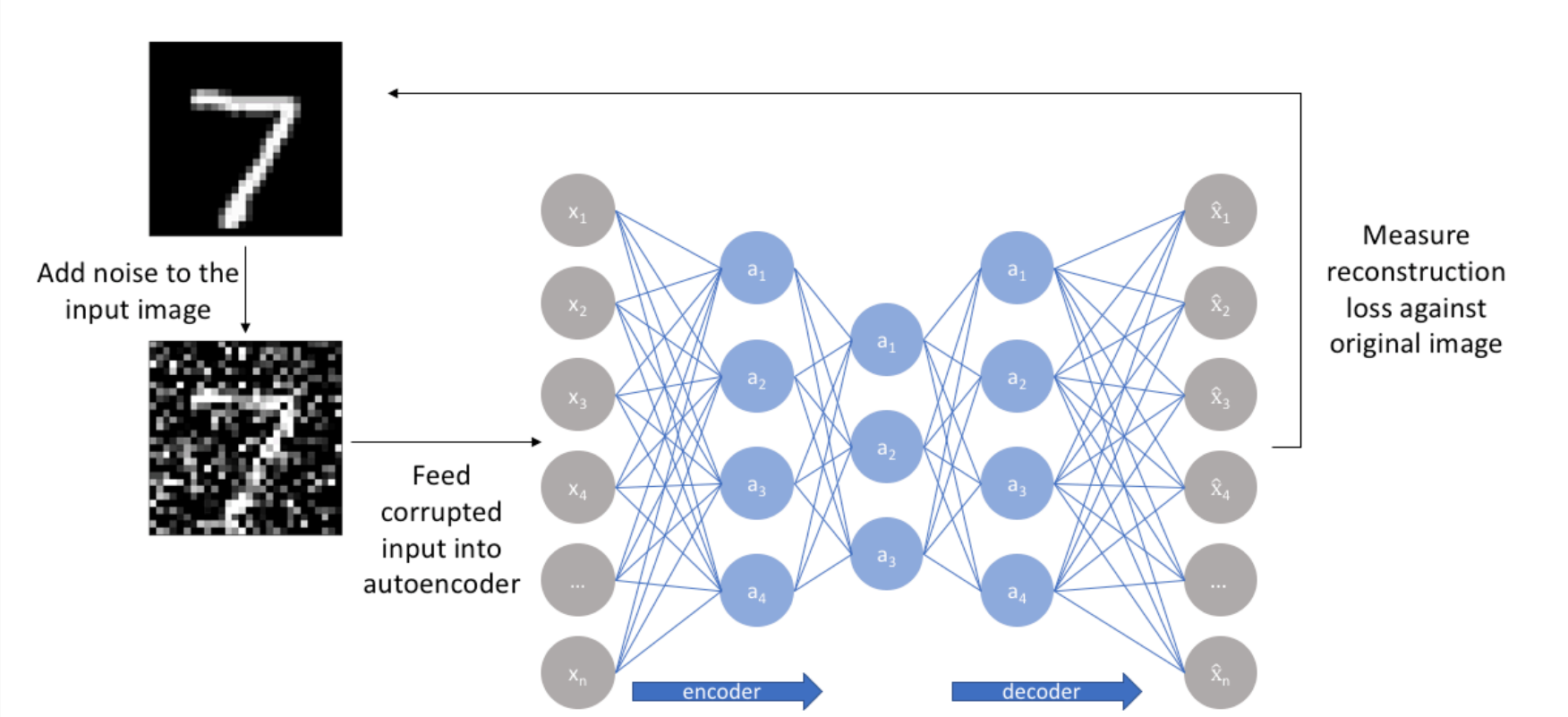
We **allow our network to sensitize individual hidden layer nodes toward specific attributes of the input data**.
A sparse autoencoder is selectively activate regions of the network depending on the input data.
Limiting the network's capacity to memorize the input data without limiting the networks capability to extract features from the data.

$$\mathcal{L}(x, \hat{x}) + \lambda \sum_i |z_i|$$



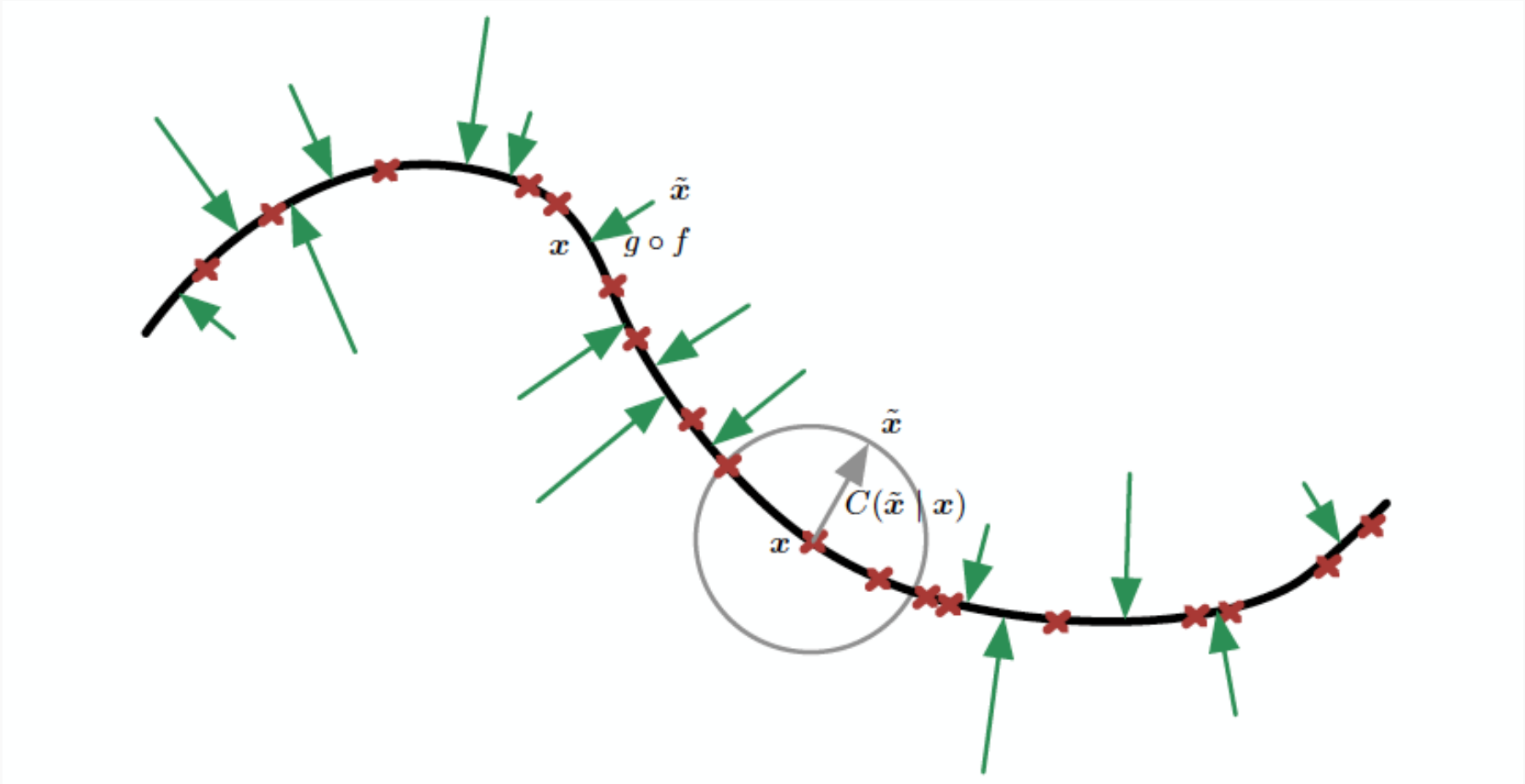
Denoising Autoencoders

Trained with corrupted data points, but to reconstruct original



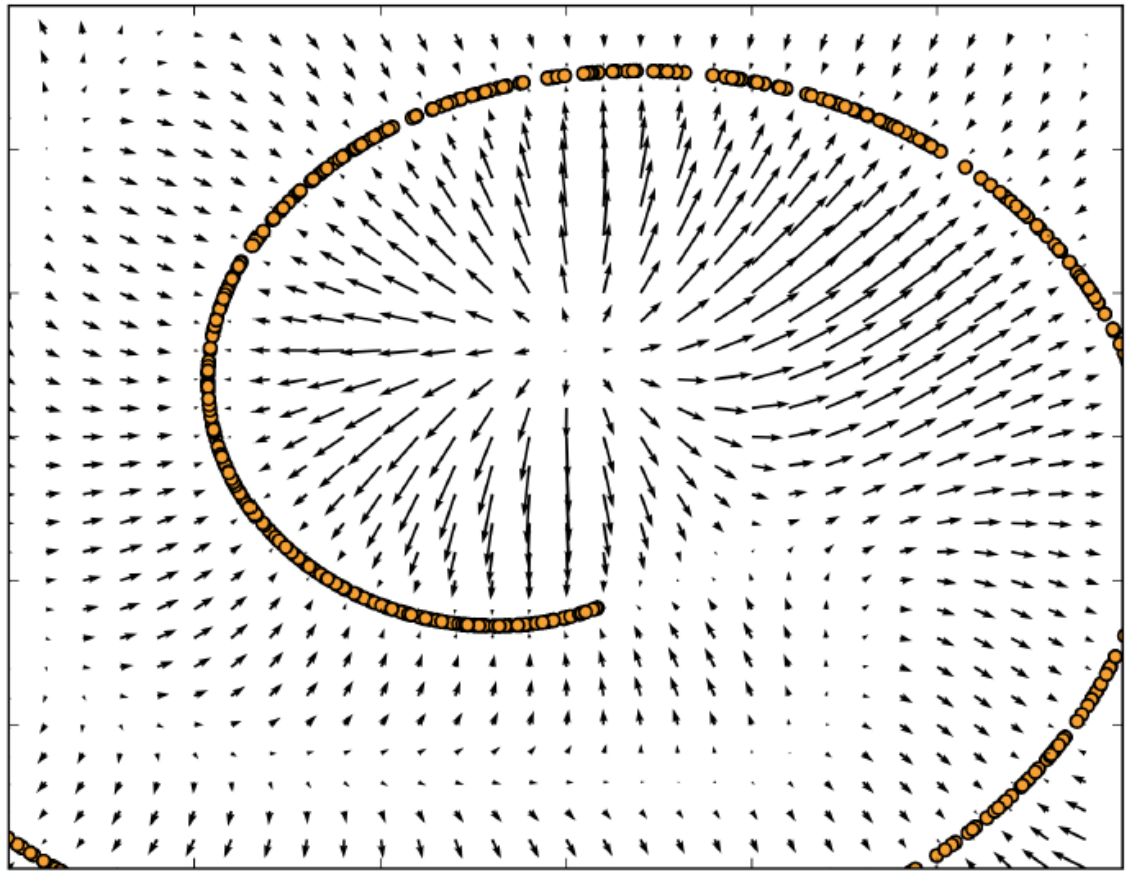
Denoising Autoencoders (cont)

Denoising autoencoders learn a manifold. Vector field learned by denoising autoencoder. Each arrow is proportional to $g(f(x))$ $- x$



Denoising Autoencoders (cont)

Vector field learned by denoising autoencoder. Each arrow is proportional to $g(f(x)) - x$



Contractive Autoencoders

One would expect that **for very similar inputs, the learned encoding would also be very similar**.

We can explicitly train our model in order for this to be the case by requiring that the *derivative of the hidden layer activations are small* with respect to the input.

Question: How do we find how much the encoded space would change if the input changes?

Derivatives

$$L(x, g(f(x))) + \lambda \left\| \frac{\partial f(x)}{\partial x} \right\|_F^2$$

Problems with Autoencoders

- Gaps in the latent space
- Separability in the latent space
- Discrete latent space