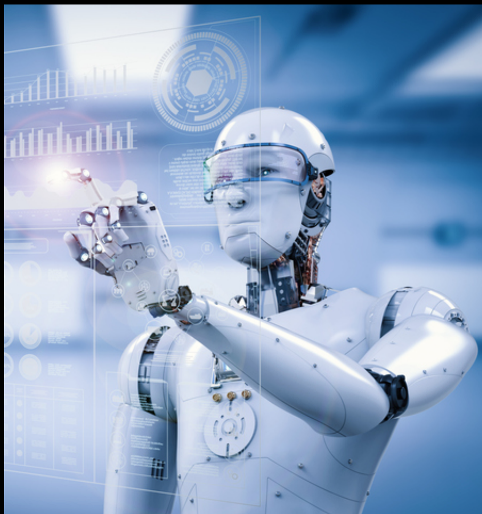
CS 66: Machine Learning

Prof. Sara Mathieson

Spring 2019

SWARTHMORE

MACHINE LEARNING



What society thinks I do

$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

This implies that

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

As for the derivative with respect to b, we obtain

$$\frac{\partial}{\partial b}\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0.$$

If we take the definition of w in Equation (9) and plug that back int
Lagrangian (Equation 8), and simplify, we get

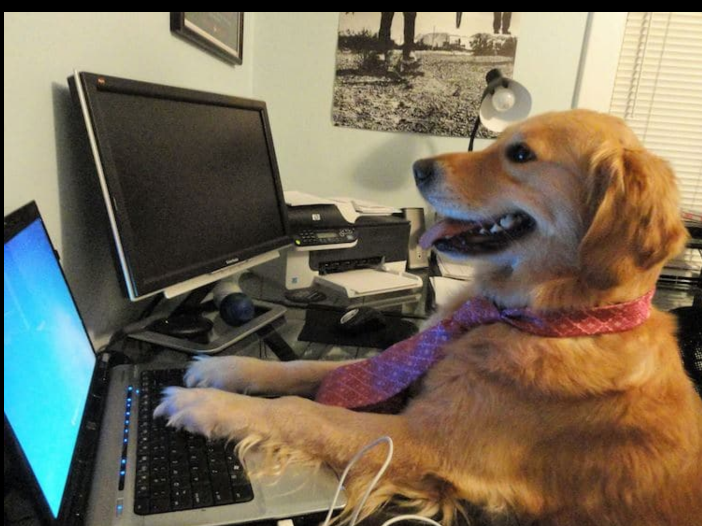
$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^{m} \alpha_i y^{(i)}.$$

But from Equation (10), the last term must be zero, so we obtain

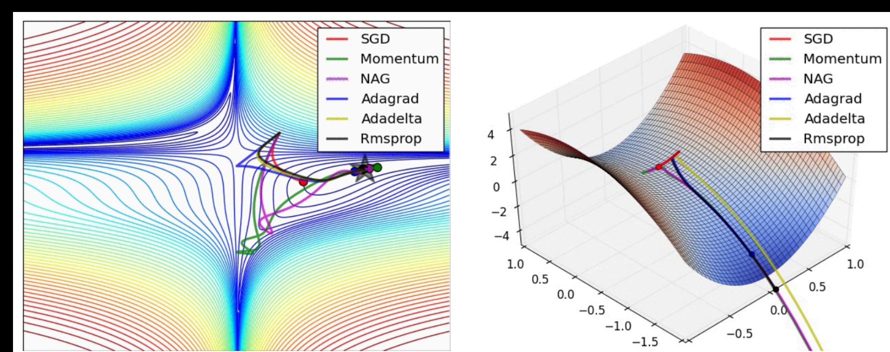
$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2} \sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)}.$$

What my boss thinks I do

What other computer scientists think I do



What mathematicians think I do



What I think I do

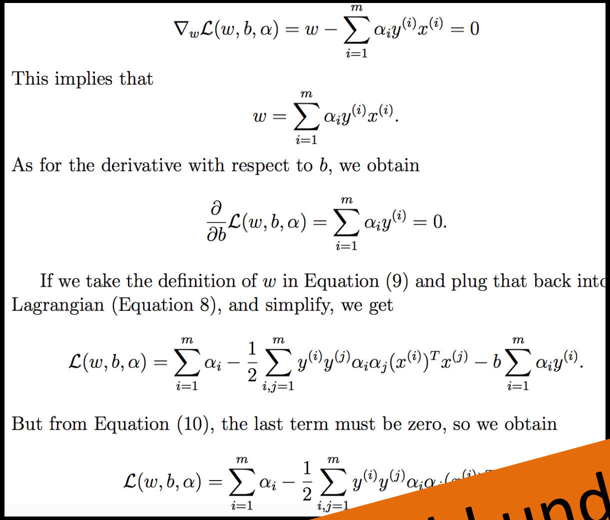
```
>>> from sklearn import svm
>>> import tensorflow as tf
```

What I really do

Adapted from: "Know Your Meme"

MACHINE LEARNING

What society thinks I do



$$\nabla_w \mathcal{L}(w,b,\alpha) = w - \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)} = 0$$

This implies that

$$w = \sum_{i=1}^{m} \alpha_i y^{(i)} x^{(i)}.$$

As for the derivative with respect to b, we obtain

$$\frac{\partial}{\partial b}\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i y^{(i)} = 0.$$

If we take the definition of w in Equation (9) and plug that back into Lagrangian (Equation 8), and simplify, we get

$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j (x^{(i)})^T x^{(j)} - b \sum_{i=1}^{m} \alpha_i y^{(i)}.$$

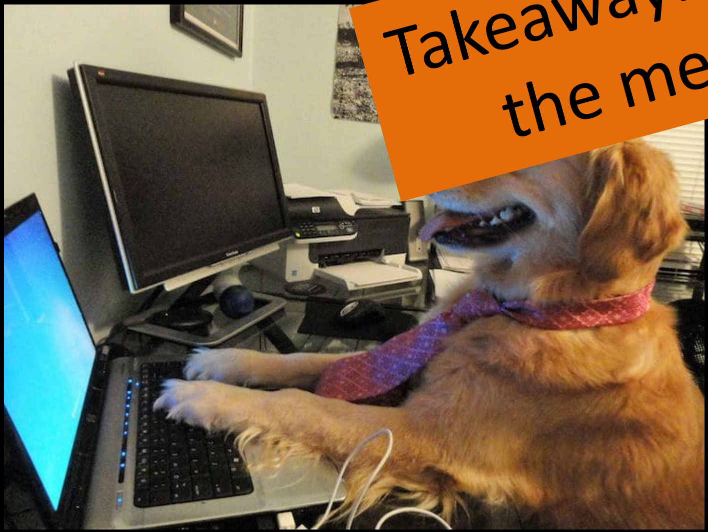
But from Equation (10), the last term must be zero, so we obtain

$$\mathcal{L}(w,b,\alpha) = \sum_{i=1}^{m} \alpha_i - \frac{1}{2}\sum_{i,j=1}^{m} y^{(i)} y^{(j)} \alpha_i \alpha_j$$

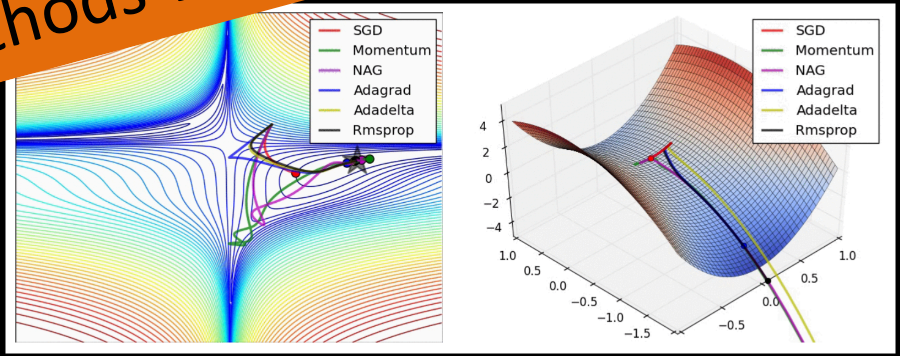
Wha

t other computer scientists think I do

Takeaway: we should understand the methods we are using!



What mathematicians think I do



What I think I do

```
>>> from sklearn import svm
>>> import tensorflow as tf
```

What I really do

Lab 7 getting started

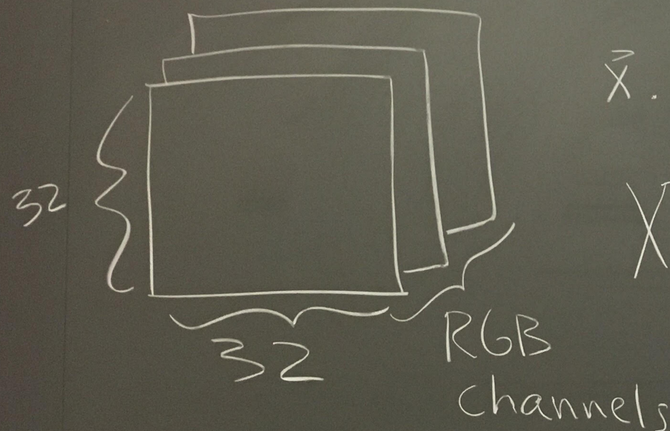
- It is helpful to have our data be zero-centered, so we will subtract off the mean

- It is also helpful to have the features be on the same scale, so we will divide by the standard deviation

- We will compute the mean and std with respect to the *training data*, then apply the same transformation to all datasets

Lab 7 getting started

- Input is now itself a multi-dimensional array
- For images, often the shape of each image will be (width, height, 3) for RGB channels
- Need to “*flatten*” or “unravel” for fully connected networks



$\vec{x}.\text{shape} = (32, 32, 3)$

$X.\text{shape} = (n, \underbrace{32, 32, 3})$

"flatten", "unravel"

$32 \cdot 32 \cdot 3 = \underbrace{3072}_{p}$

Lab 7 getting started

- So far in this class, we have considered *stochastic gradient descent*, where one data point is used to compute the gradient and update the weights
- On the flipside is *batch gradient descent*, where we compute the gradient with respect to all the data, and then update the weights
- A middle ground uses *mini-batches* of examples before updating the weights. This is the approach we will use in Lab 7.

Outline for April 8

- Choice of weight initialization
- Regularization
- Autoencoders and unsupervised pre-training
- Begin: convolutional neural networks

  - Lab 7 check-in (fully connected NN part) on WED
  - Lab 7 due Monday (week from today)
  - Office hours TODAY 12:30-2pm

Outline for April 8

- Choice of weight initialization
- Regularization
- Autoencoders and unsupervised pre-training
- Begin: convolutional neural networks

Weight initialization

- All 0's initialization is bad! Causes nodes to compute the same outputs, so then the weights go through the same updates during gradient descent

- Need asymmetry! => usually use small random values

Weight initialization

- Issue: nodes with more randomly initialized inputs will have a higher variance in their output

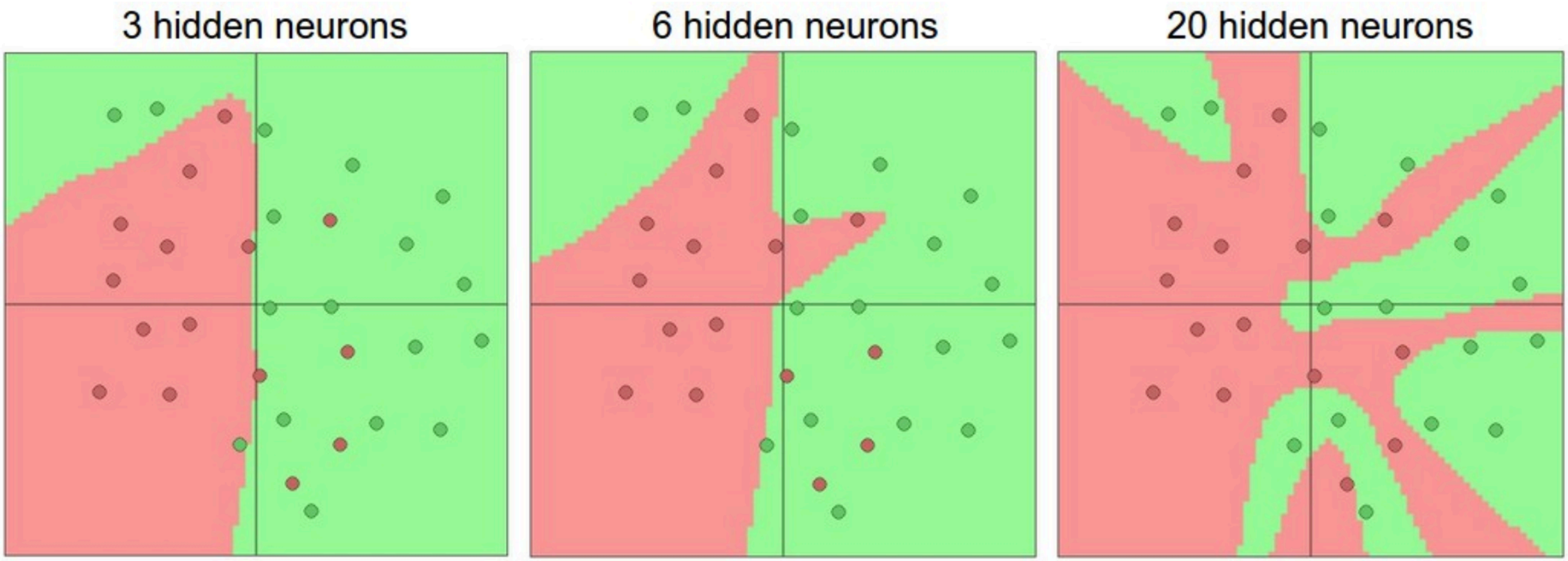
- Solution: divide by the sqrt(n) where n is the "fan-in" (number of inputs)

More info:
http://cs231n.github.io/neural-networks-1/

Outline for April 8

- Choice of weight initialization
- Regularization
- Autoencoders and unsupervised pre-training
- Begin: convolutional neural networks

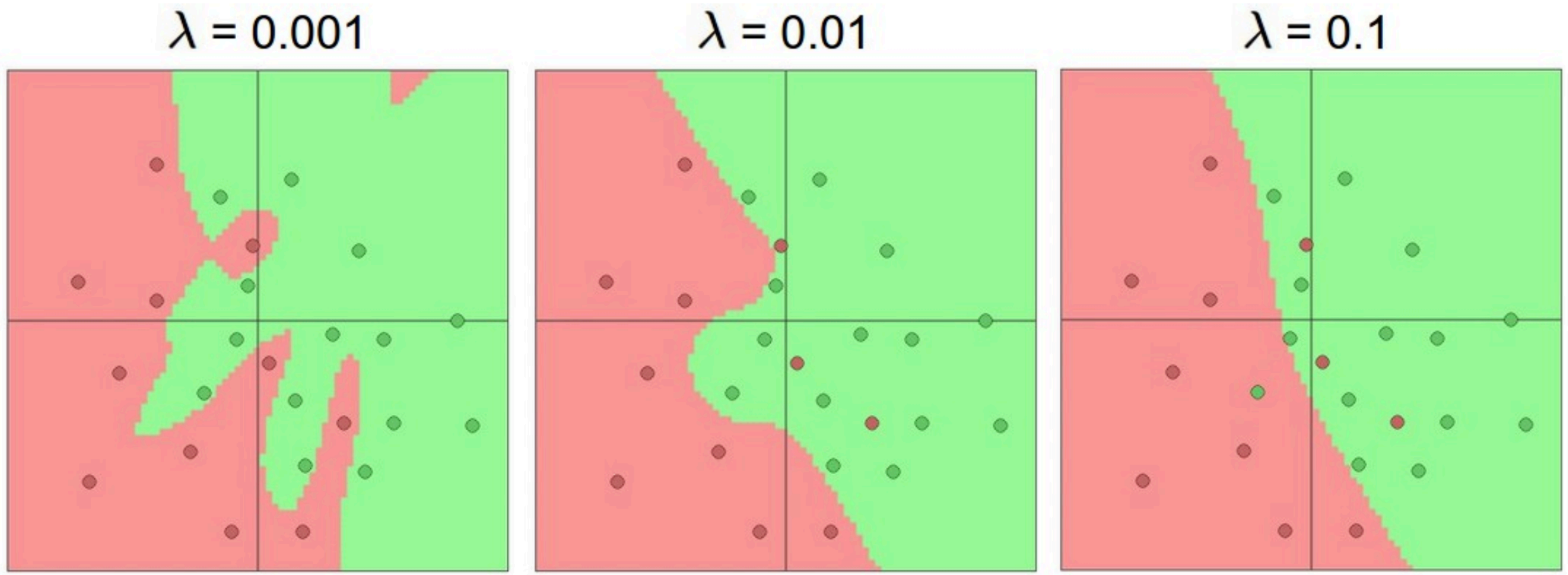
More hidden units can contribute to overfitting



Larger Neural Networks can represent more complicated functions. The data are shown as circles colored by their class, and the decision regions by a trained neural network are shown underneath. You can play with these examples in this ConvNetsJS demo.

Image from: http://cs231n.github.io/neural-networks-1/

However! It is always better to use a larger network and regularize in other ways



The effects of regularization strength: Each neural network above has 20 hidden neurons, but changing the regularization strength makes its final decision regions smoother with a higher regularization. You can play with these examples in this ConvNetsJS demo.

Image from: http://cs231n.github.io/neural-networks-1/

Handout 15

① (a) sigmoid $\boxed{\frac{1}{2}}$ (b)

tanh $\boxed{0}$

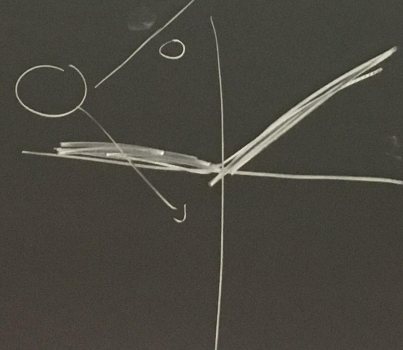
ReLU $\boxed{0}$

$$\sigma'(0) = \frac{1}{4} = \frac{1}{2}\cdot\frac{1}{2}$$

$$\xrightarrow[\boxed{-\frac{3}{4}}]{0} \sigma \xrightarrow[-3]{\frac{1}{2}}$$

(c) $\boxed{\xrightarrow[0]{-1} \text{relu} \xrightarrow[-3]{0}}$

bad if we can't move on from this.



② 1 2 3 } class label

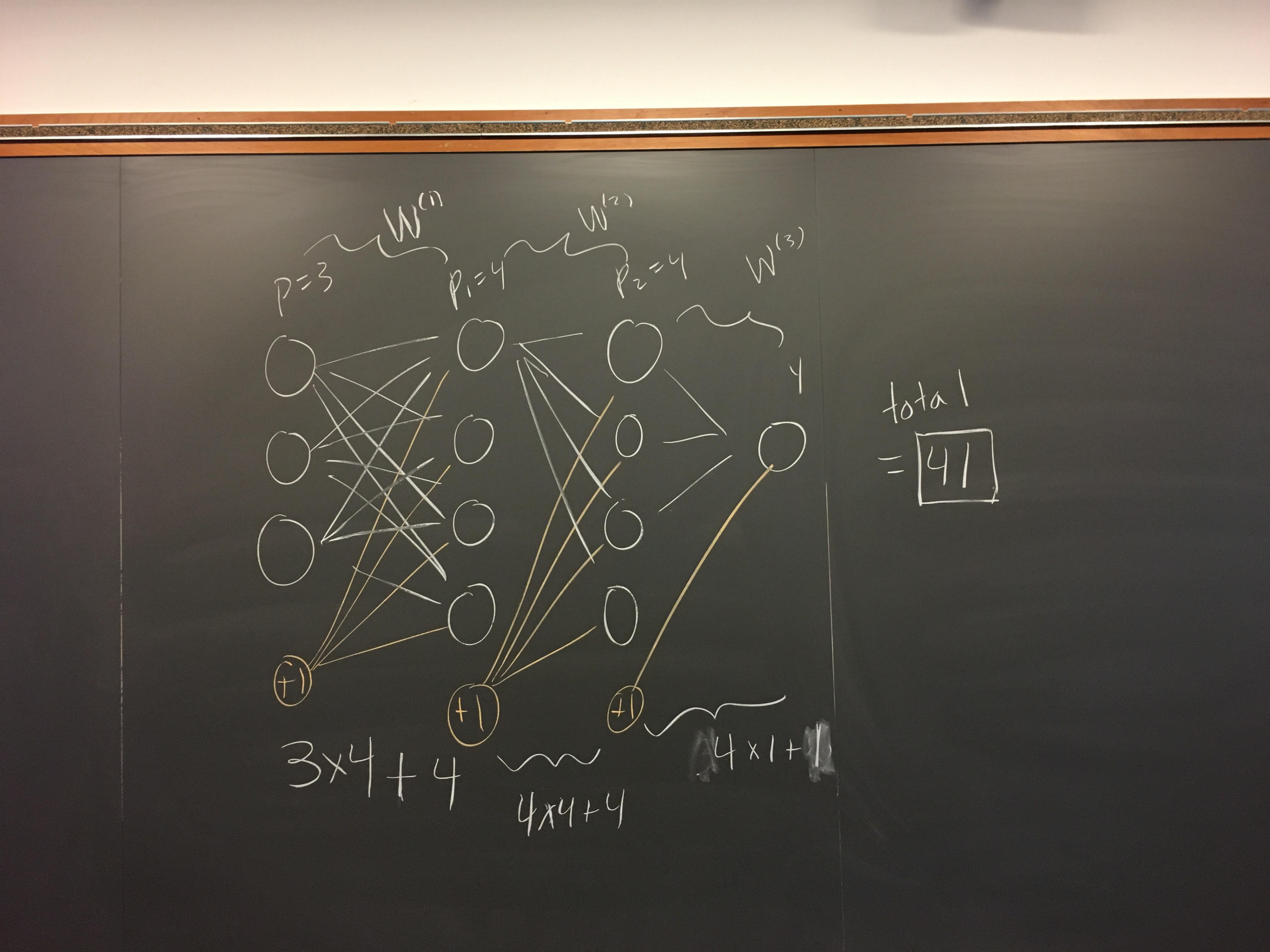
$$y = [0, 1, 0]$$

$$\hat{y} = \left[\frac{1}{4}, \frac{1}{2}, \frac{1}{4}\right] \text{ or } \hat{y} = \left[\frac{3}{8}, \frac{1}{8}, \frac{1}{2}\right]$$

$$H(y, \hat{y}) = -\log_2\left(\frac{1}{2}\right) = \boxed{1}$$

second

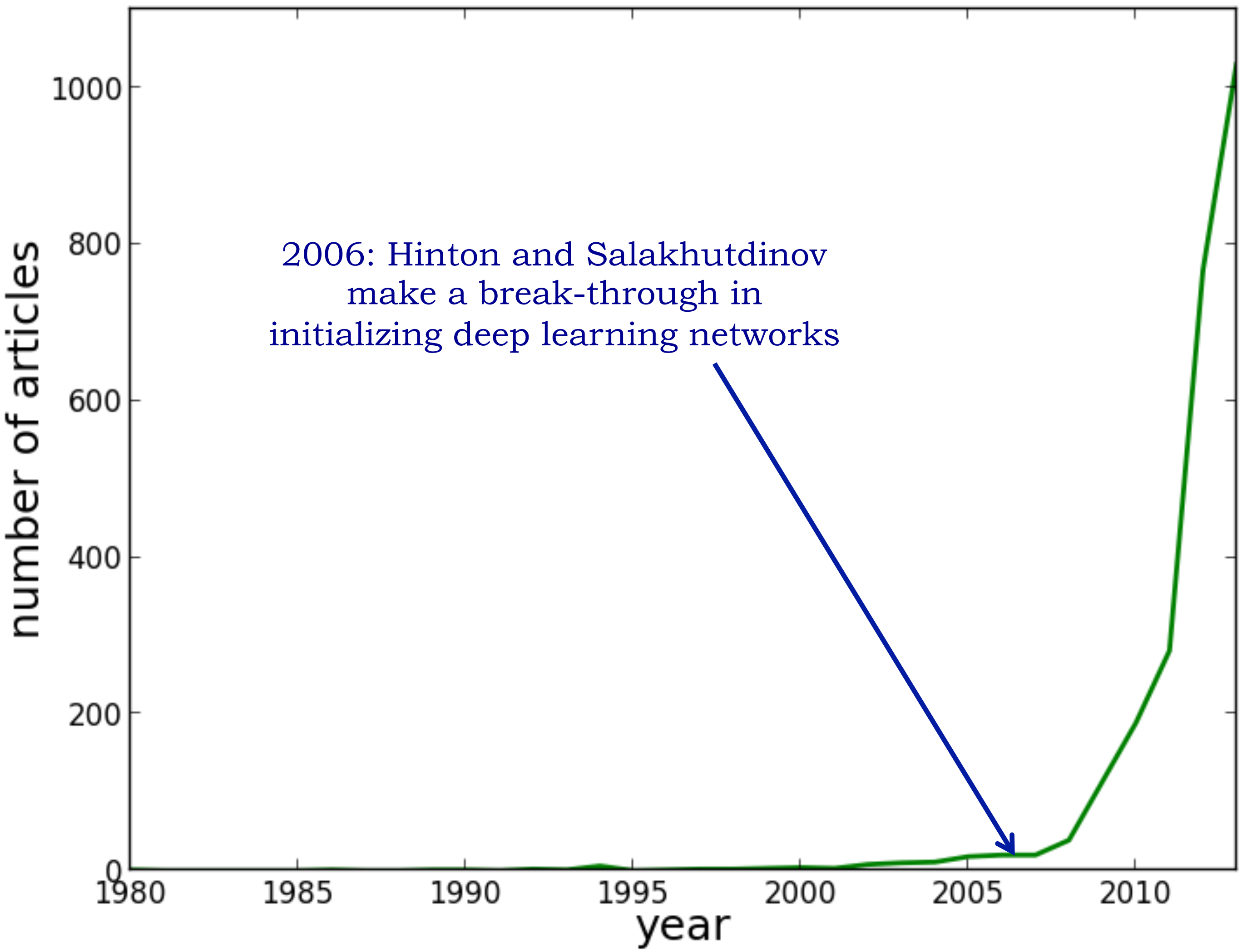
$$H(y, \hat{y}) = -\log_2\left(\frac{1}{8}\right) = \boxed{3} \cdot \frac{1}{1}$$


$W^{(1)}$
$W^{(2)}$
$W^{(3)}$
$p=3$
$p_1=4$
$p_2=4$
y
+1
+1
+1
$3\times4+4$
$4\times4+4$
$4\times1+1$
total
$=\boxed{41}$

Outline for April 8

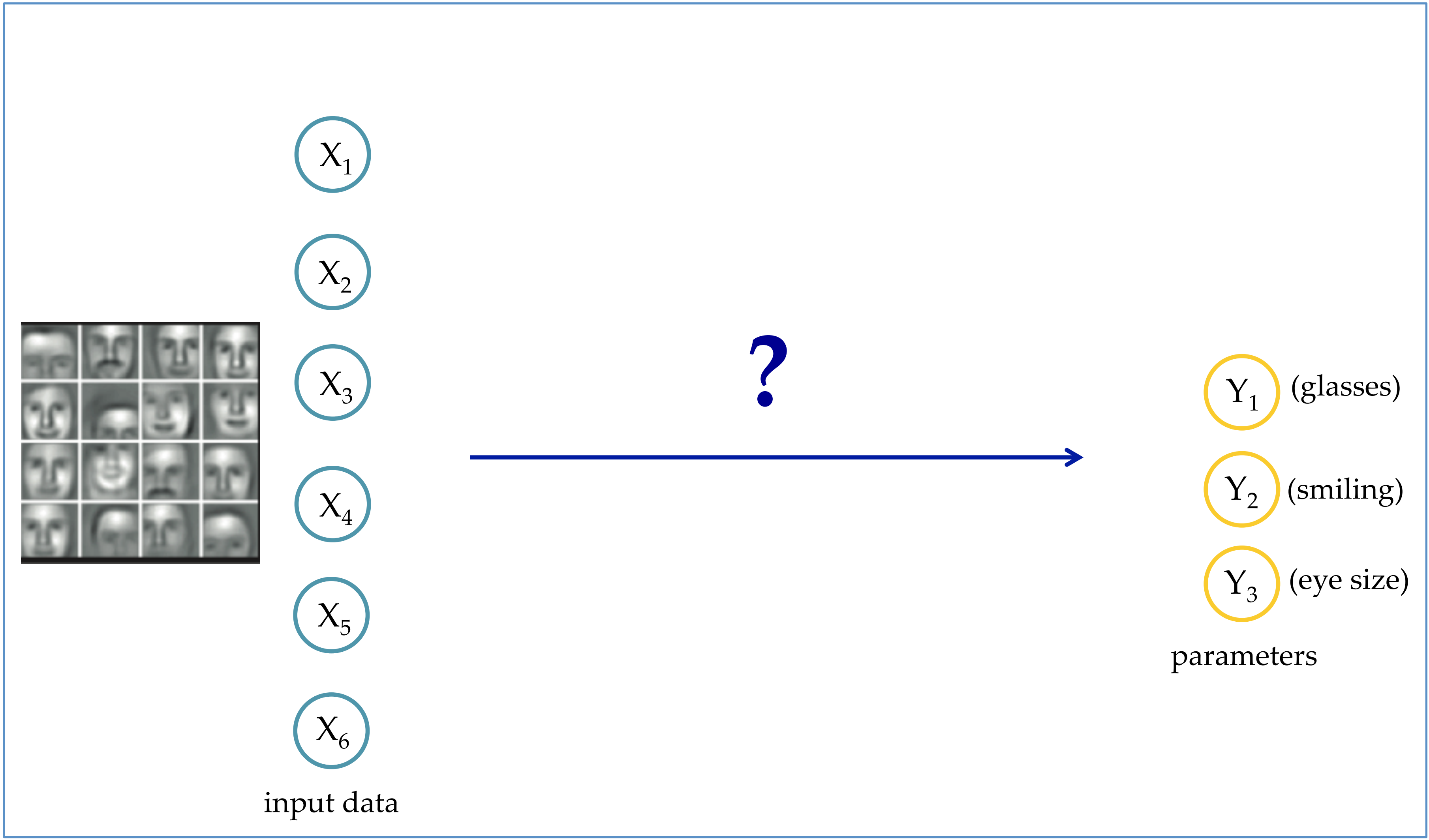
- Choice of weight initialization
- Regularization
- Autoencoders and unsupervised pre-training
- Begin: convolutional neural networks

What was this breakthrough in deep learning?

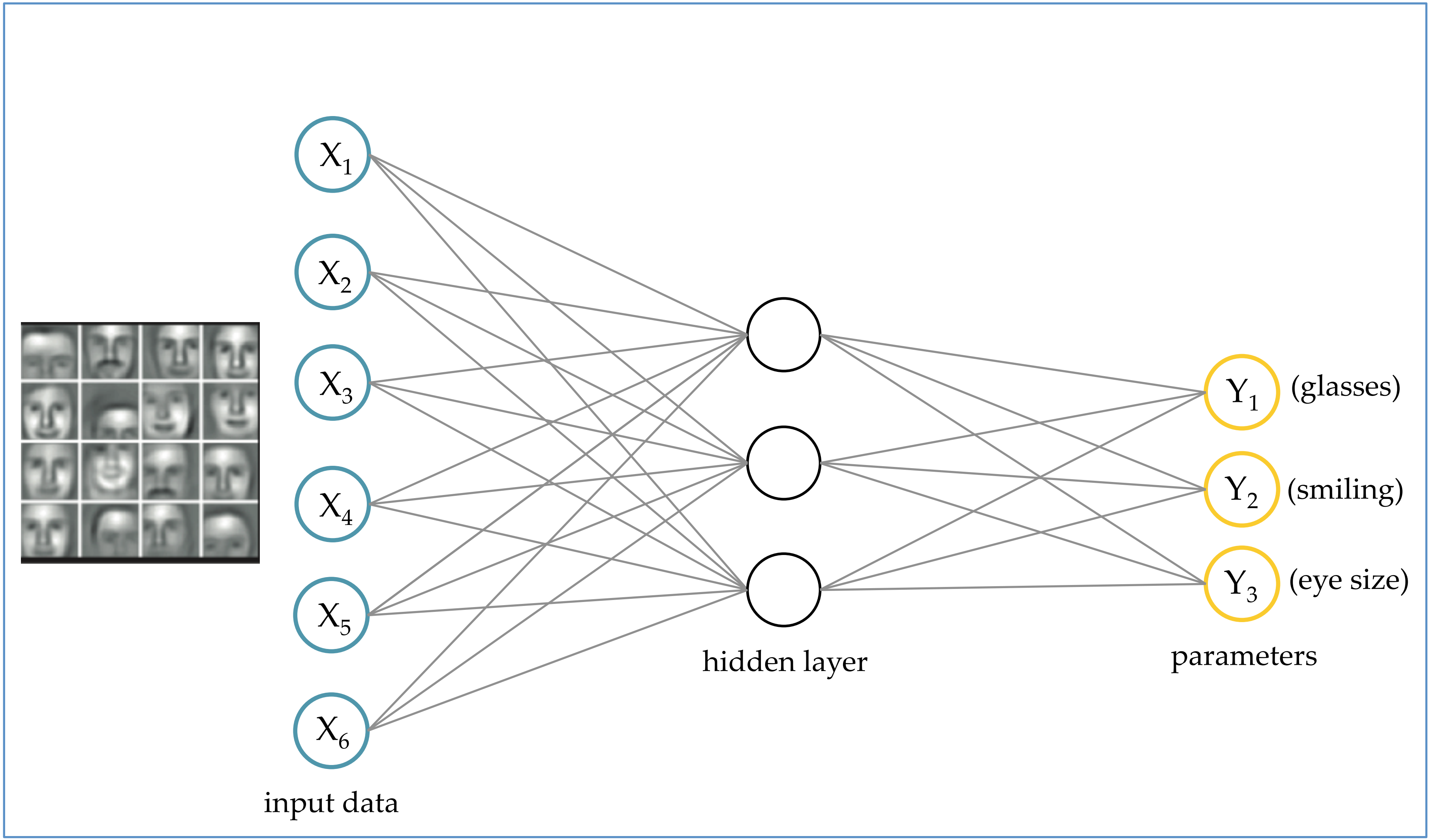


Images on next slides from
Hinton & Salakhutdinov (2006)

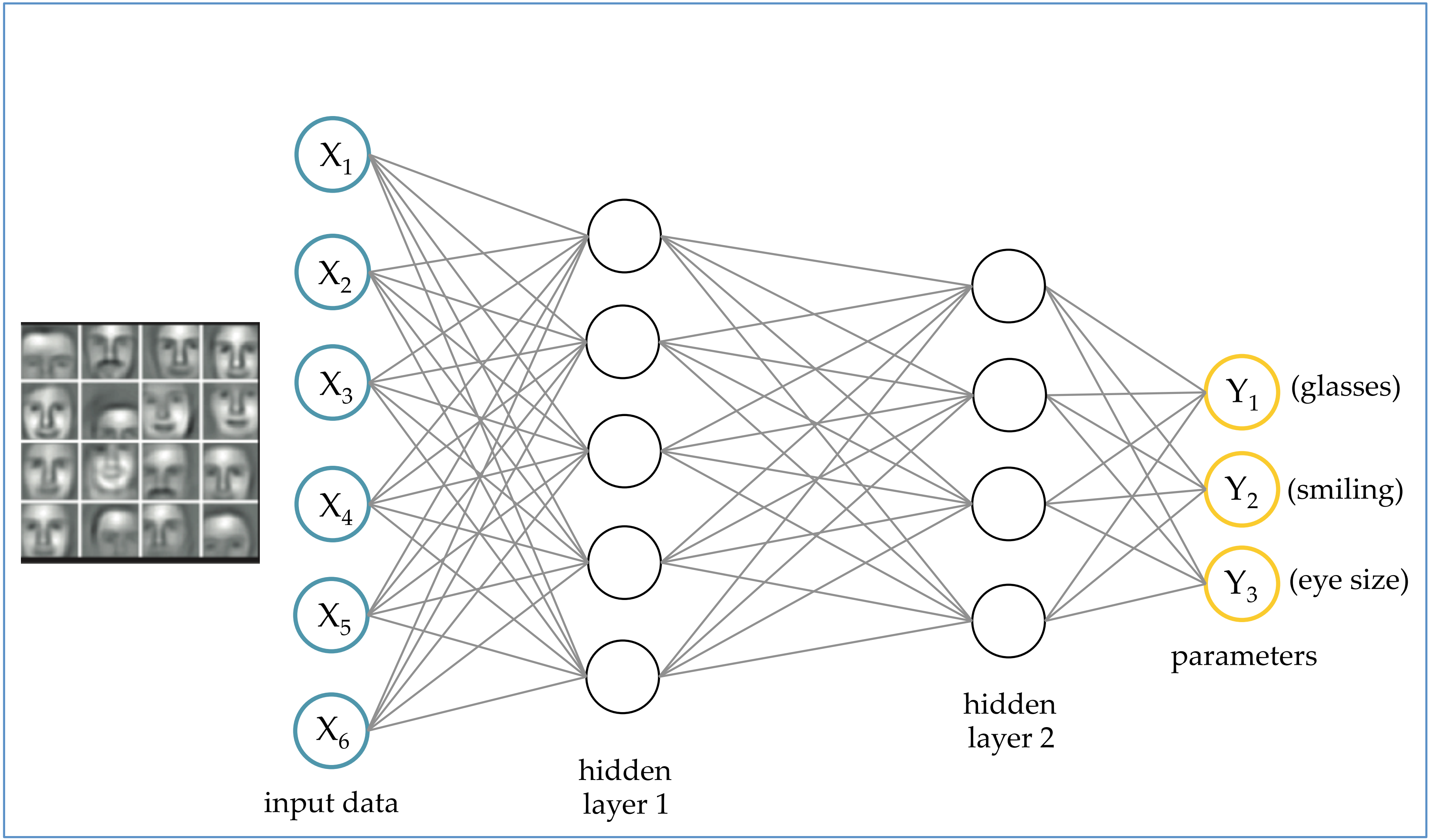
Goal: find a function between input and output



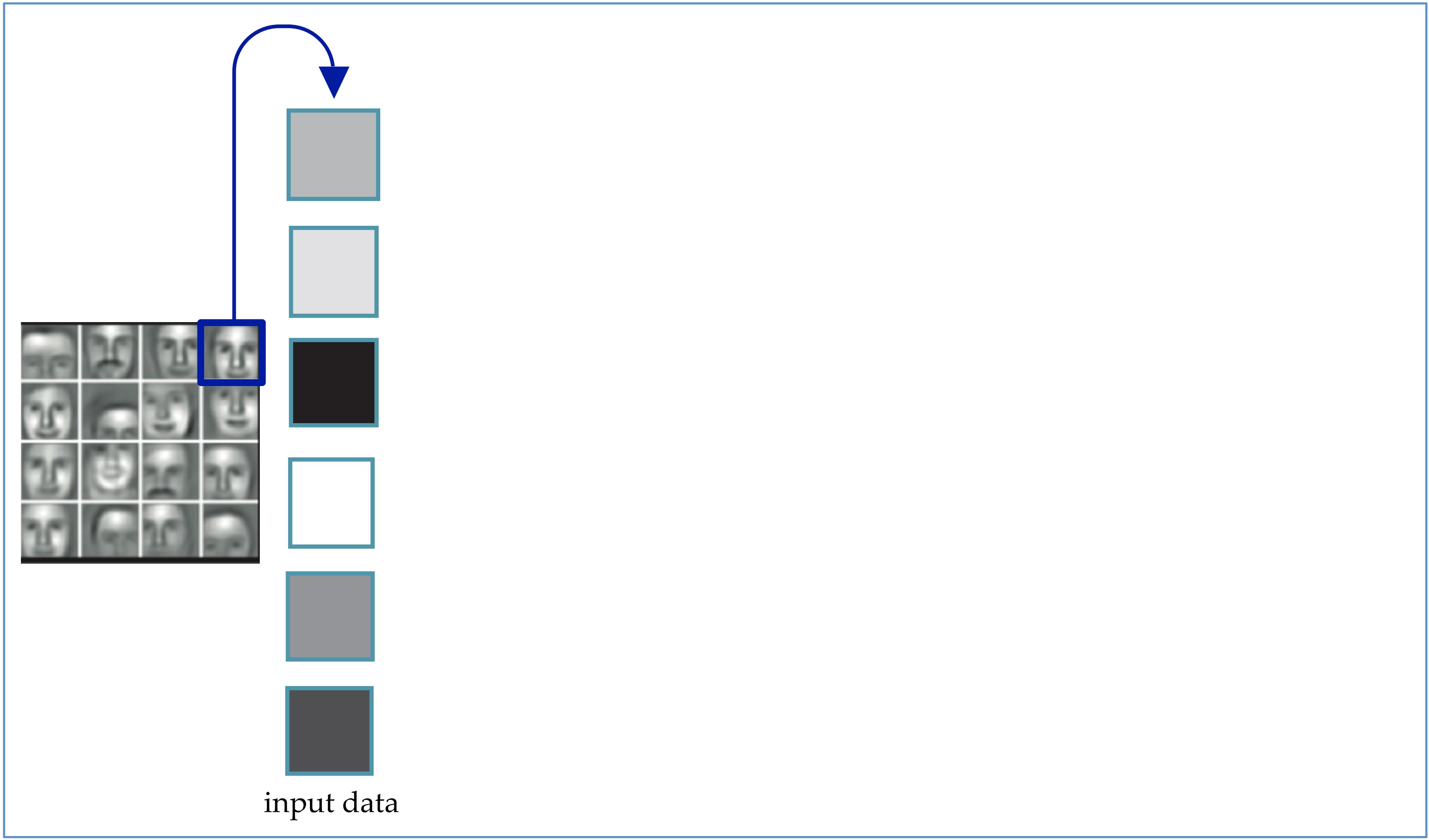
First idea: one hidden layer



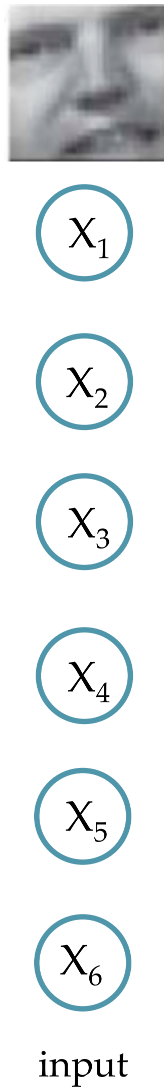
Second idea: more hidden layers ("deep" learning)



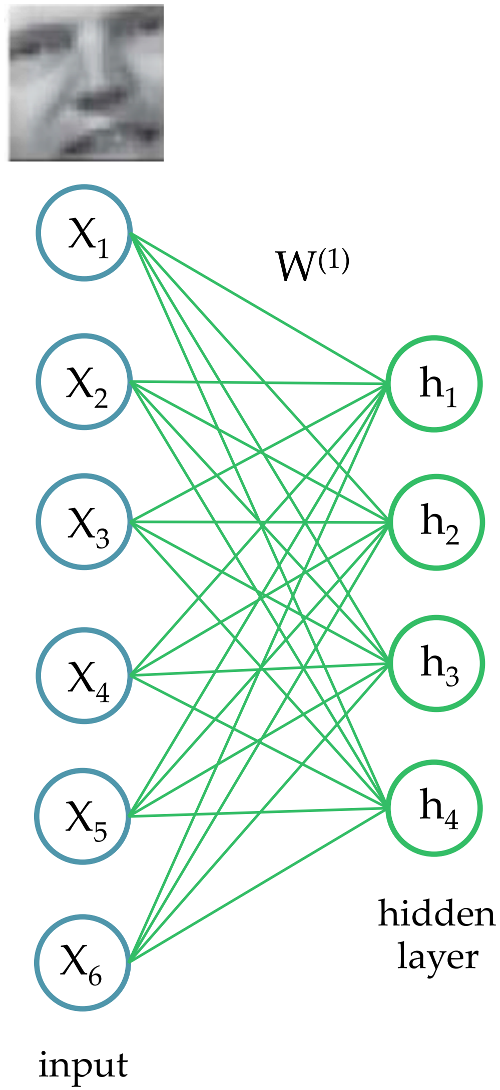
Flatten pixels of image into a single vector



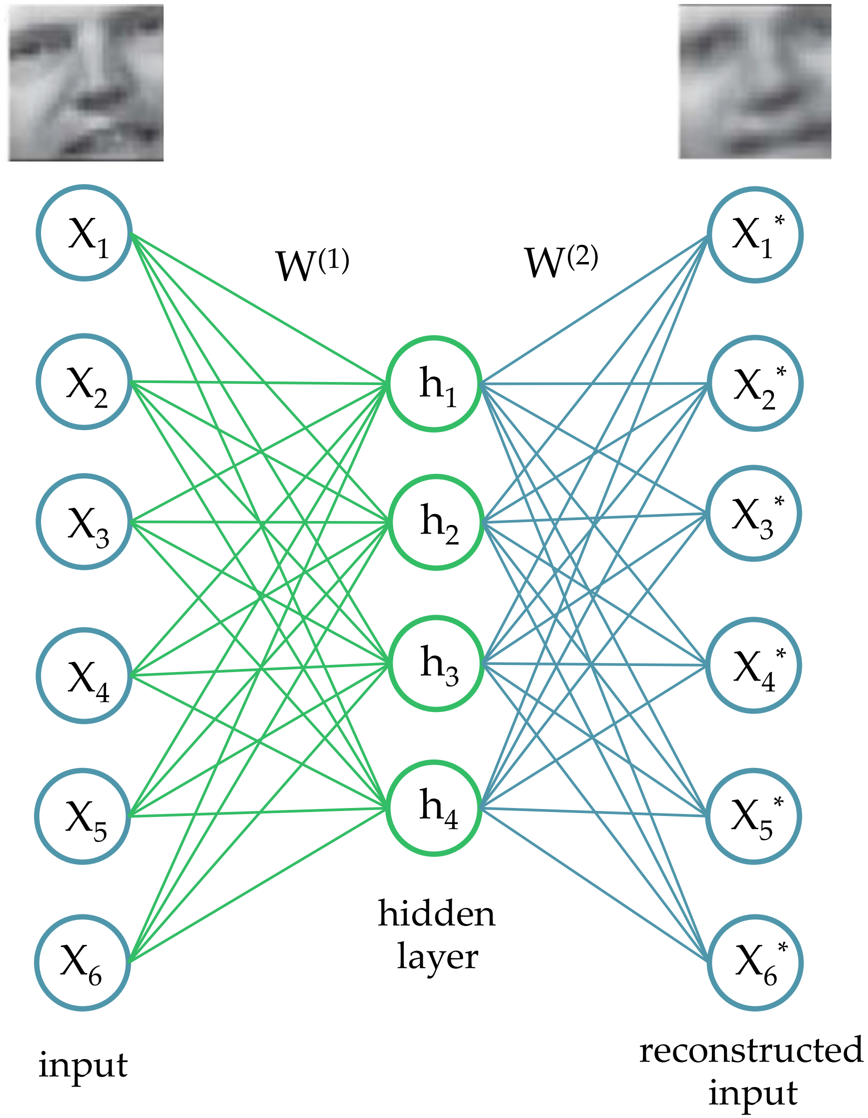
Detour to autoencoders



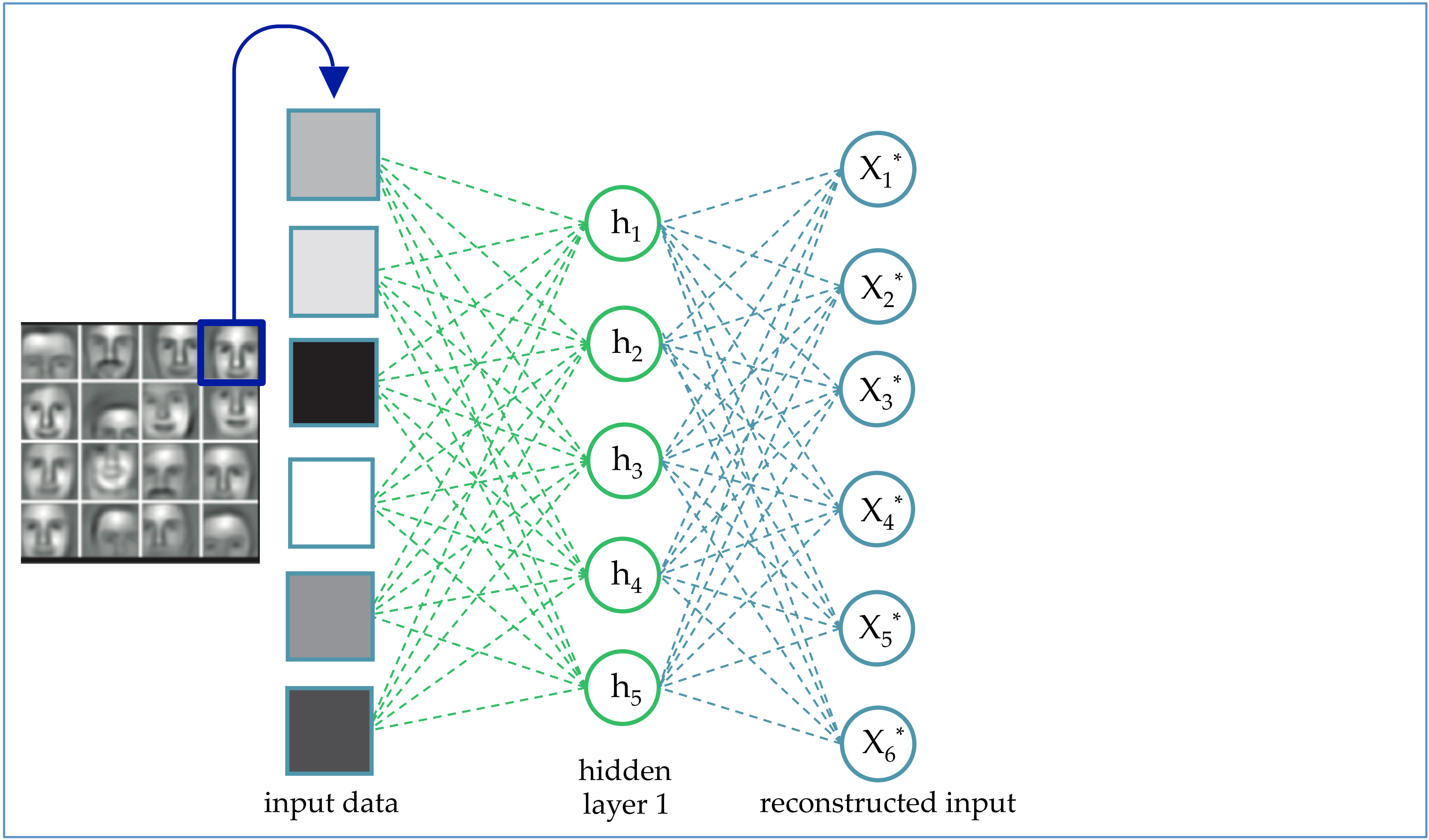
Detour to autoencoders



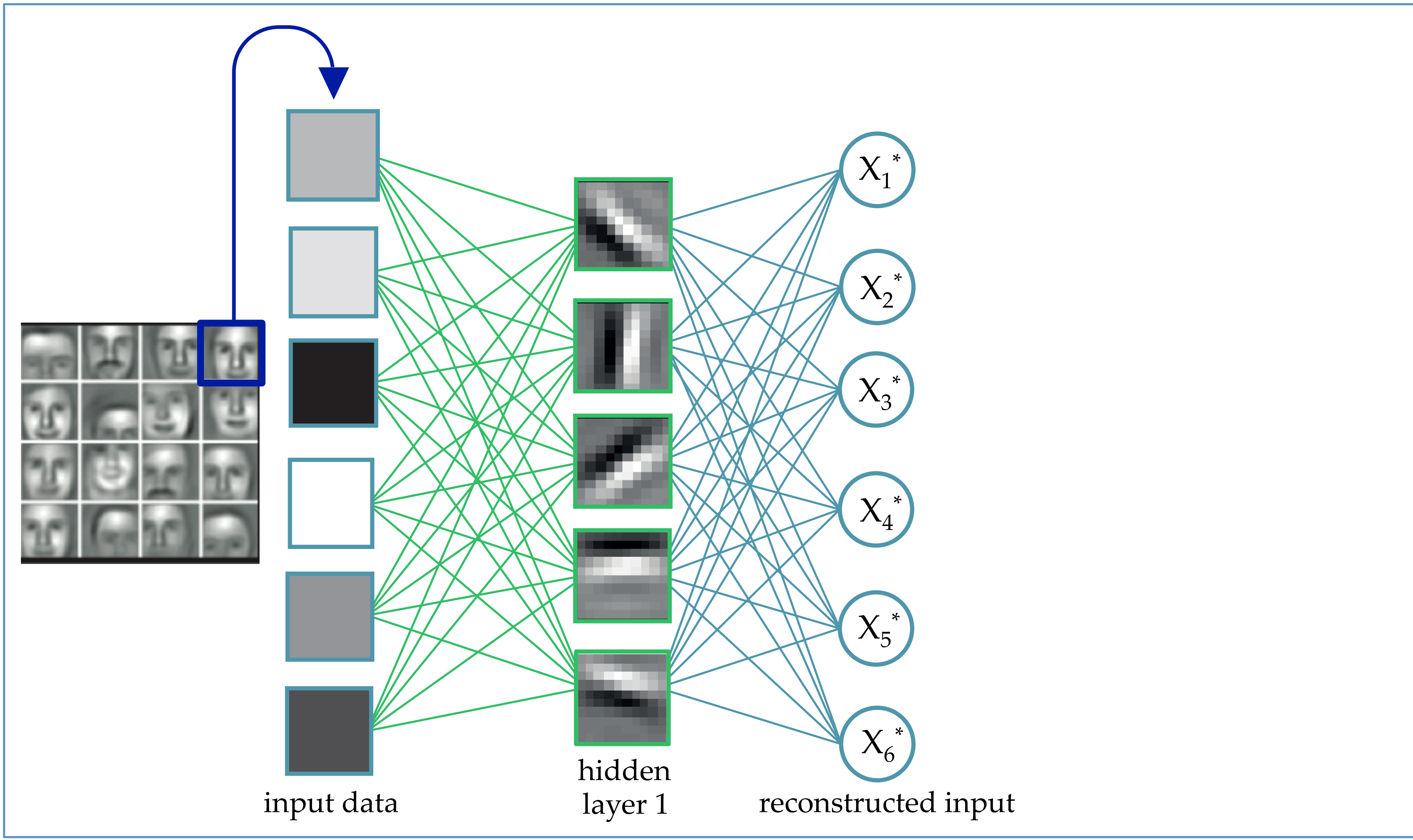
Detour to autoencoders



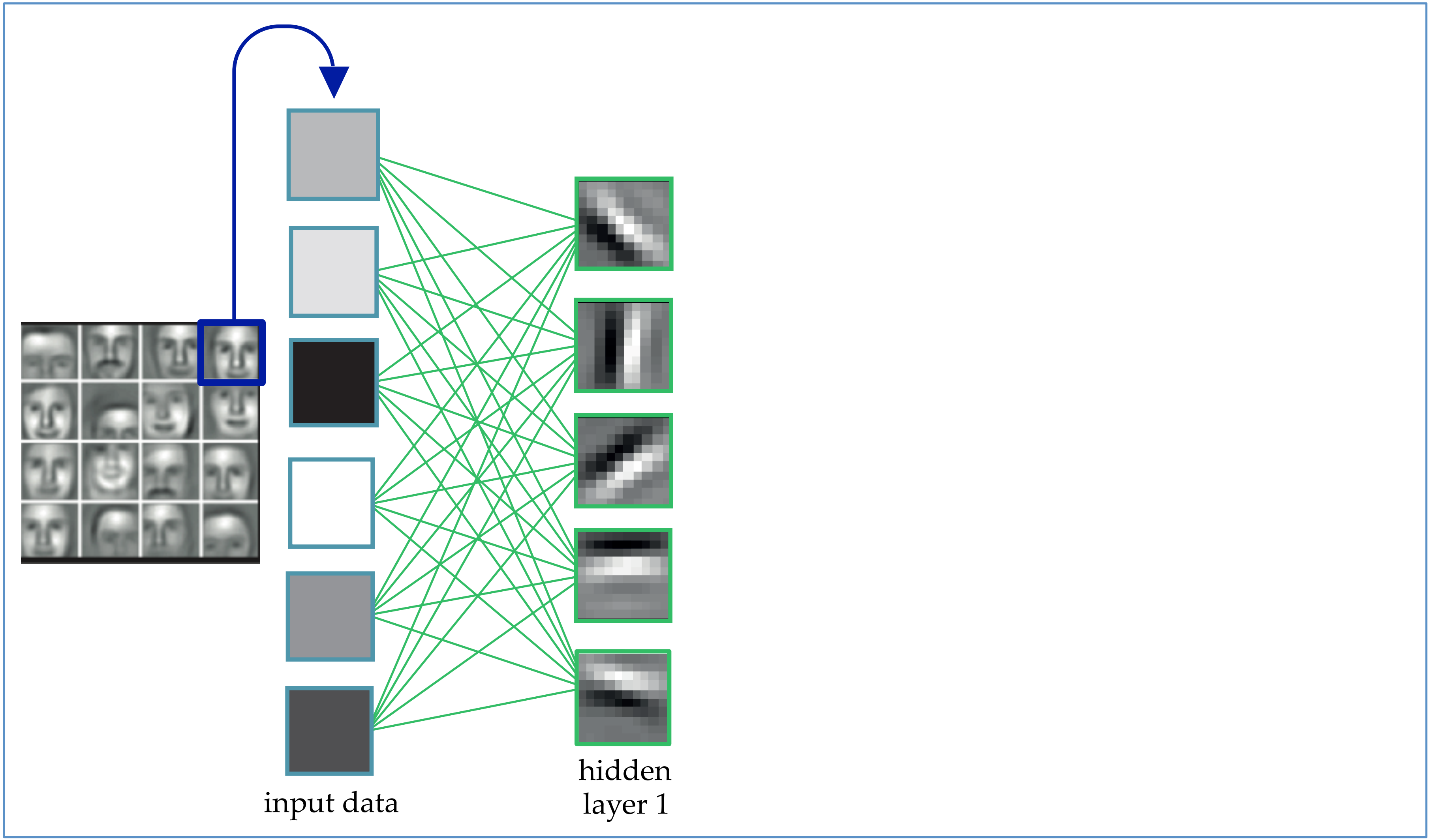
Use unsupervised pre-training to find a function from the input to itself



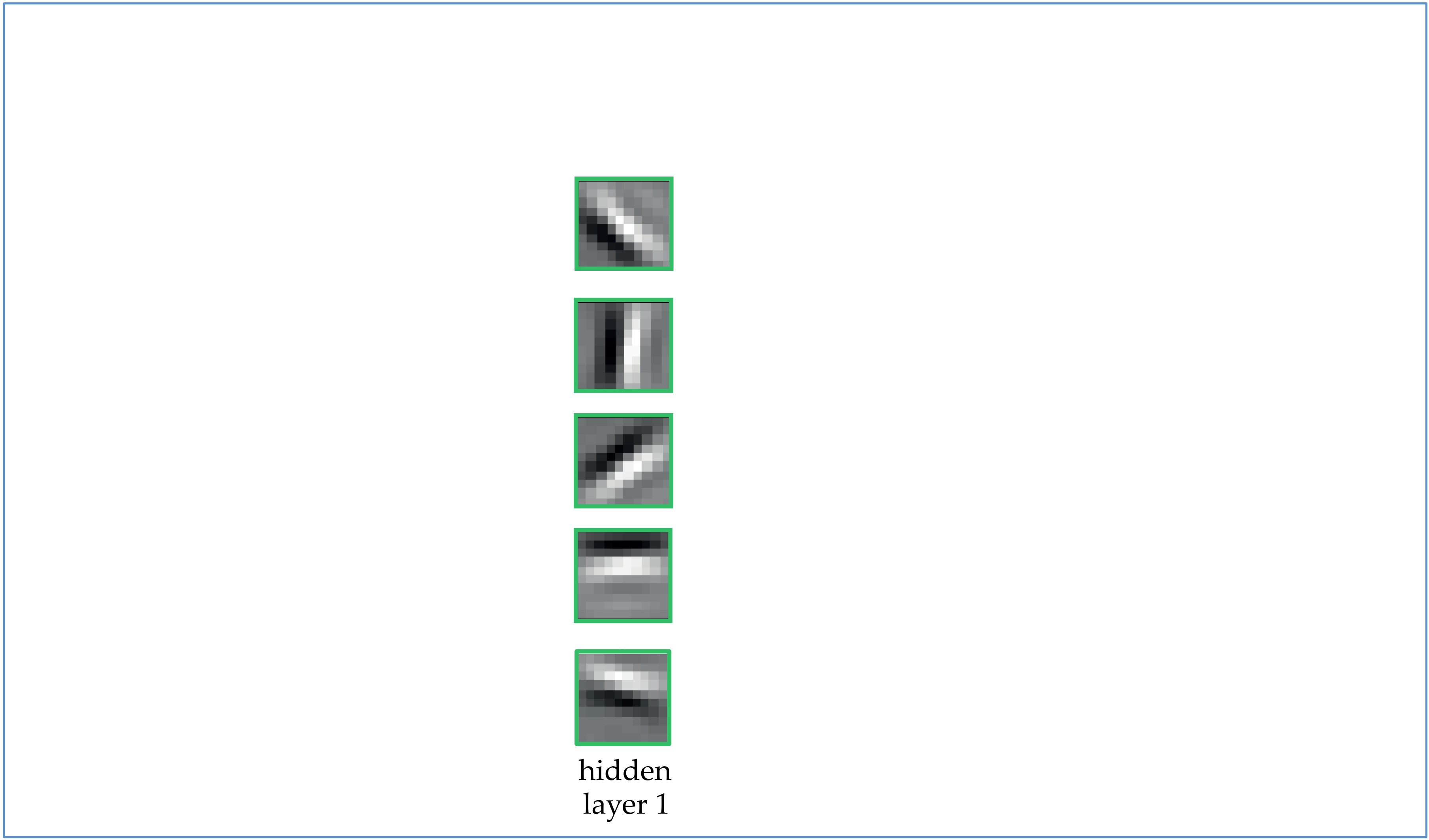
Hidden units can be interpreted as edges



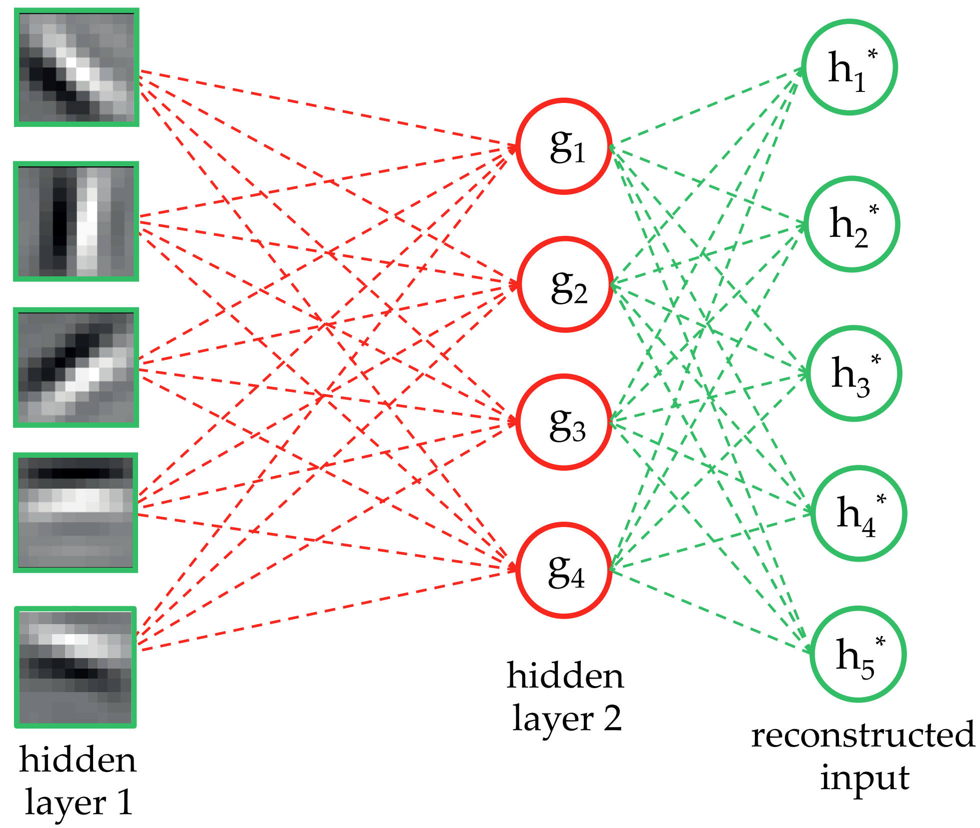
Now: throw away reconstruction and input



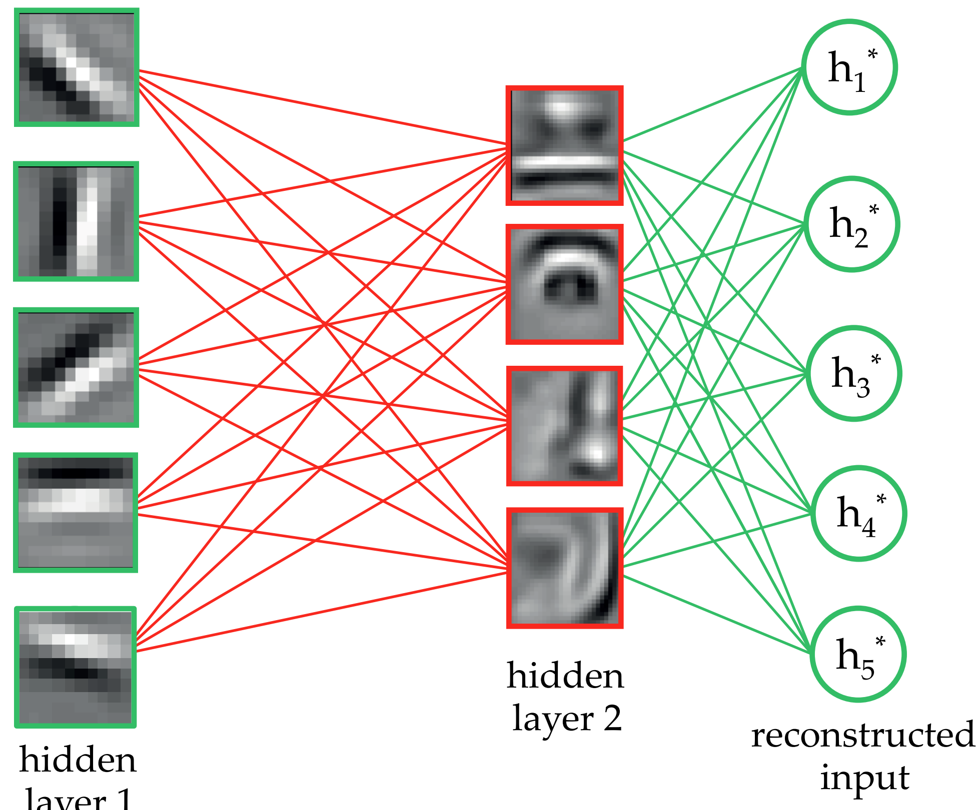
Now: throw away reconstruction and input



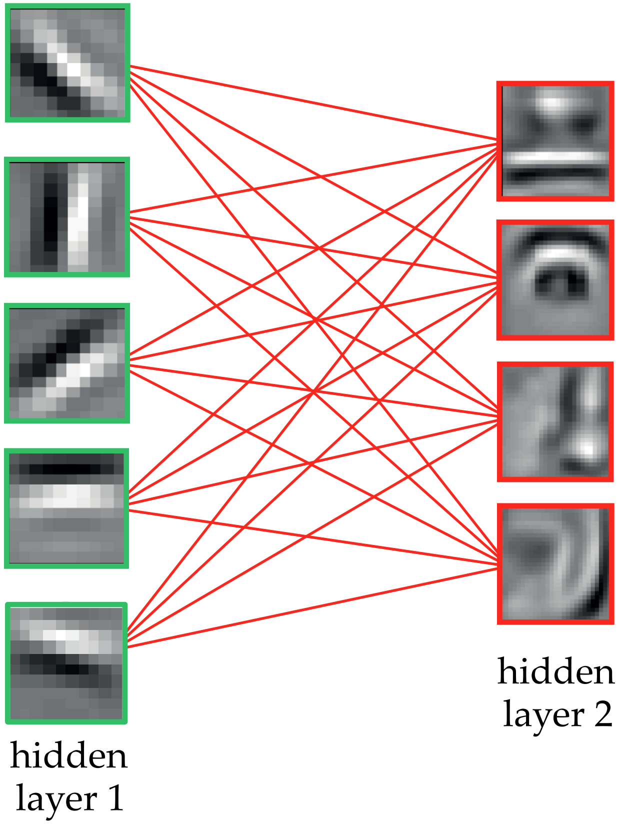
Then repeat the entire process for each layer



Then repeat the entire process for each layer

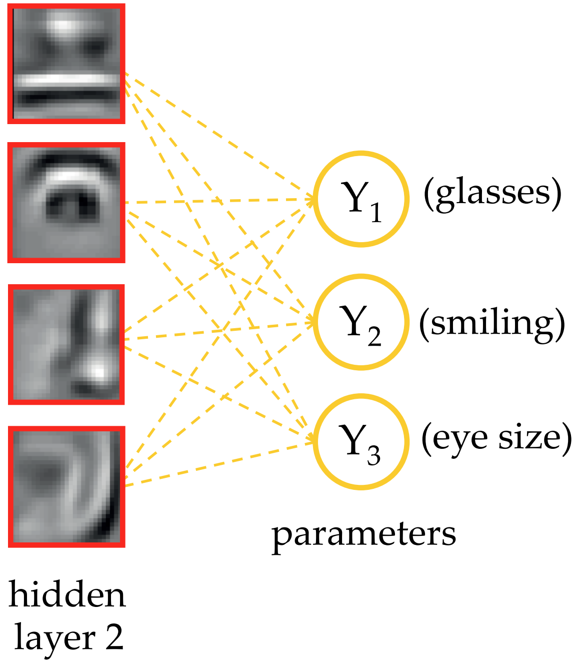


Then repeat the entire process for each layer

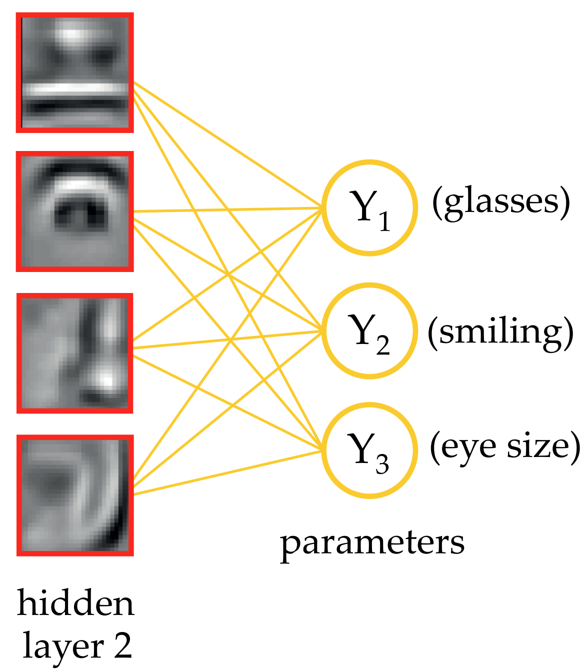


Then repeat the entire process for each layer

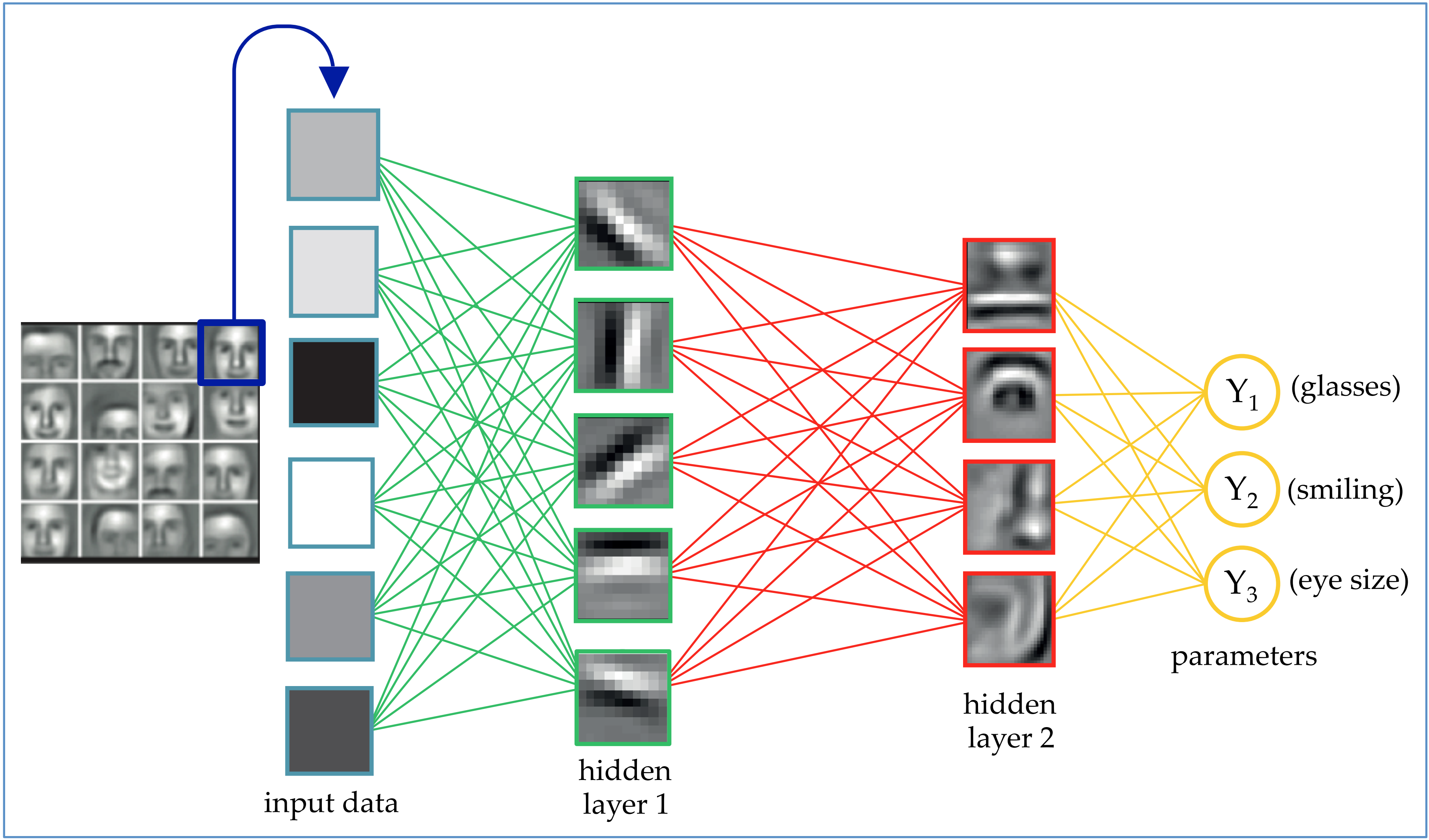
In the last layer, use the outputs (supervised)



In the last layer, use the outputs (supervised)



Finally, “fine-tune” the entire network!



Outline for April 8

- Choice of weight initialization
- Regularization
- Autoencoders and unsupervised pre-training
- Begin: convolutional neural networks

Next time!