Understanding How Neural Networks See

SML201: Introduction to Data Science, Spring 2020

Michael Guerzhoy

Recent successes of neural networks

- Can recognize what object is in the a photo
- Can tell bad Go positions/shapes from good Go positions
- Cen tell a self-driving car where to go
- Can decide on what key to press to win at a video game by looking at the screen
- Can diagnose lung disease better than a radiologist

About this lecture

- A *very* brief introduction to artificial neural networks (ANNs)
  - Why and how ANNs work
- A *very* brief introduction to Explainable AI
  - Understanding how “black box” models work

"Review:" Supervised Machine Learning

› Training set:

- Training example 1: $\mathrm{x}^{(1)} = \left(x_1^{(1)}, x_2^{(1)}, \dots, x_m^{(1)}\right)$ output: $y^{(1)}$
- Training example 2: $\mathrm{x}^{(2)} = \left(x_1^{(2)}, x_2^{(2)}, \dots, x_m^{(2)}\right)$ output: $y^{(2)}$

…

- Training example N: $\mathrm{x}^{(\mathrm{N})} = \left(x_1^{(N)}, x_2^{(N)}, \dots, x_m^{(N)}\right)$ output: $y^{(N)}$

› Test set:

- Test Example 1: $\mathrm{x}^{(N+1)} = \left(x_1^{(N+1)}, x_2^{(N+1)}, \dots, x_m^{(N+1)}\right)$ output: $y^{(N+1)}$
- Test Example 2: $\mathrm{x}^{(N+2)} = \left(x_1^{(N+2)}, x_2^{(N+2)}, \dots, x_m^{(N+2)}\right)$ output: $y^{(N+2)}$
- …
- Test Example K: $\mathrm{x}^{(N+K)} = \left(x_1^{(N+K)}, x_2^{(N+K)}, \dots, x_m^{(N+K)}\right)$ output: $y^{(N+K)}$

- Goal: Find a θ such that $h_\theta\left(x^{(i)}\right) \approx y^{(i)}$ for $i \in 1, \dots, N$
- Hope: $h_\theta\left(x^{(i)}\right) \approx y^{(i)}$ for any i
- For new input x, predict $h_\theta(x)$

Shotgun debugging

Machine Learning vs. Intro to Programming

- Programming ***done badly***

```
CountryMaxIncome <- function(gap):
  return(min(gap$gdpPercap)

> CountryMaxIncome(gapminder)
10
```

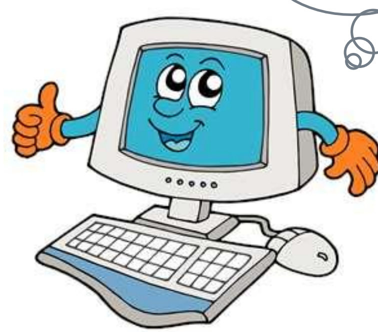
Change the `min` to `max`?

- Machine Learning ***done right***

```
>>> h_(0,1.2,0.1)([0, 0])
[0, 0]
>>> h_(0,1.2,0.1)([1, 2])
[1.3, 2.8]
```

$$h_{(\theta_1,\theta_2,\theta_3)}(x) = \theta_1 + \theta_2 x + \theta_3 x^2$$

Change θ_2 to 1.3?

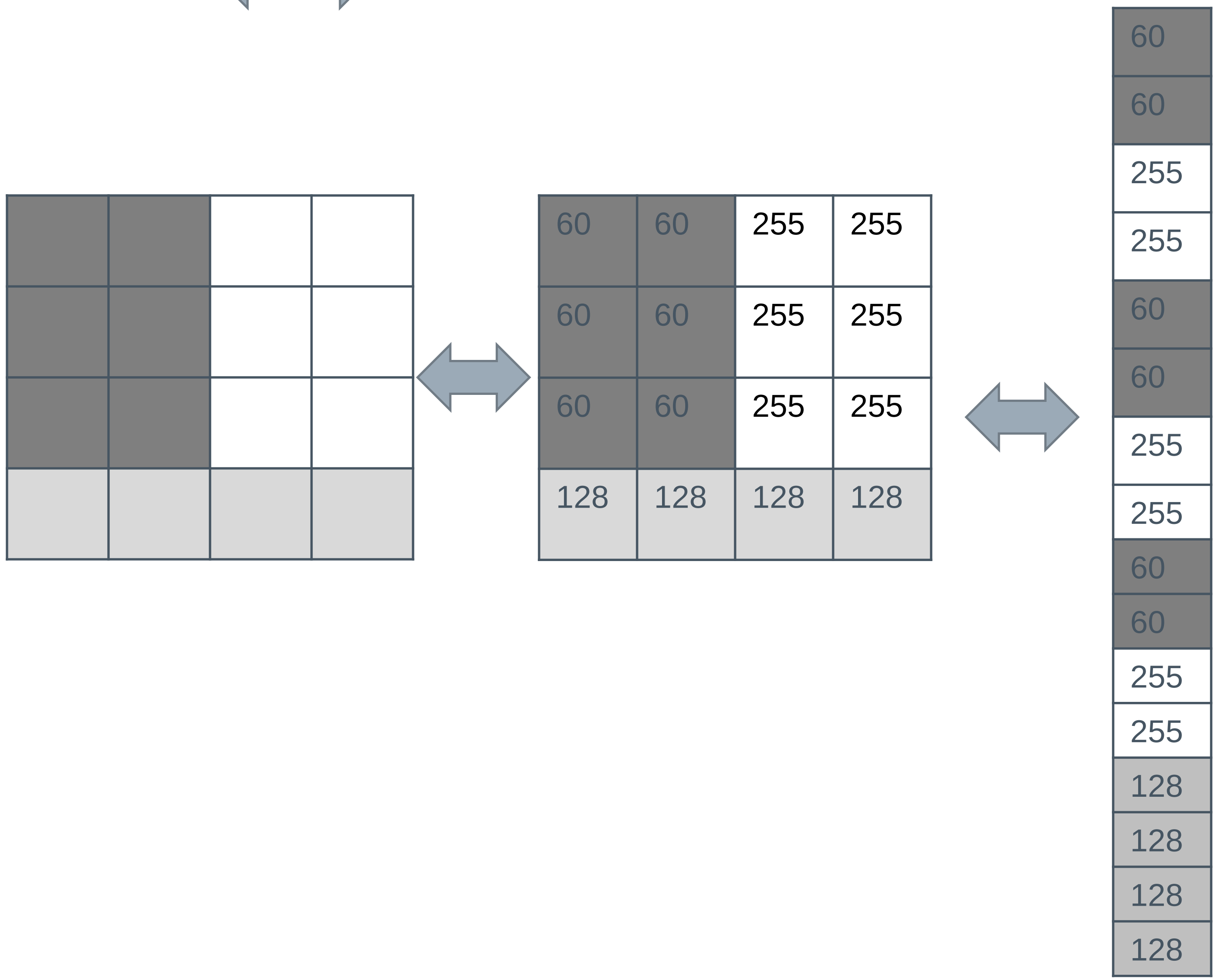


Machine learning (kind of)

Sample ML task: Recognizing Justin Bieber

What Justin Bieber looks like to a computer

Images ⟷ Vectors



The Face Recognition Task

- Training set:
  - $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$
    - $x^{(i)}$ is a k-dimensional vector consisting of the intensities of all the pixels in in the i-th photo (20×20 photo $\rightarrow x^{(i)}$ is 400-dimensional)
    - $y^{(i)}$ is the *label* (i.e., name)
- Test phase:
  - We have an input vector x, and want to assign a label y to it
    - Whose photo is it?

Face Recognition using 1-Nearest Neighbors (1NN)

– Training set: $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(N)}, y^{(N)})\}$

– Input: x

– 1-Nearest Neighbor algorithm:

› Find the training photo/vector $x^{(i)}$ that's as "close" as possible to x, and output the label $y^{(i)}$

Input x

Closest training image to the input x

Output: Paul

Supervised Machine Learning

- Training set:
  - Training example 1: $\mathrm{x}^{(1)} = \left(x_1^{(1)}, x_2^{(1)}, \ldots, x_m^{(1)}\right)$ output: $y^{(1)}$
  - Training example 2: $\mathrm{x}^{(2)} = \left(x_1^{(2)}, x_2^{(2)}, \ldots, x_m^{(2)}\right)$ output: $y^{(2)}$

  …
  - Training example N: $\mathrm{x}^{(\mathrm{N})} = \left(x_1^{(N)}, x_2^{(N)}, \ldots, x_m^{(N)}\right)$ output: $y^{(N)}$
- Test set:
  - Test Example 1: $\mathrm{x}^{(N+1)} = \left(x_1^{(N+1)}, x_2^{(N+1)}, \ldots, x_m^{(N+1)}\right)$ output: $y^{(N+1)}$
  - Test Example 2: $\mathrm{x}^{(N+2)} = \left(x_1^{(N+2)}, x_2^{(N+2)}, \ldots, x_m^{(N+2)}\right)$ output: $y^{(N+2)}$
  - …
  - Test Example K: $\mathrm{x}^{(N+K)} = \left(x_1^{(N+K)}, x_2^{(N+K)}, \ldots, x_m^{(N+K)}\right)$ output: $y^{(N+K)}$

- Goal: Find a θ such that $h_\theta\left(x^{(i)}\right) \approx y^{(i)}$ for $i \in 1, \ldots, N$
- Hope: $h_\theta\left(x^{(i)}\right) \approx y^{(i)}$ for any i
- For new input x, predict $h_\theta(x)$

Are the two images a and b close?

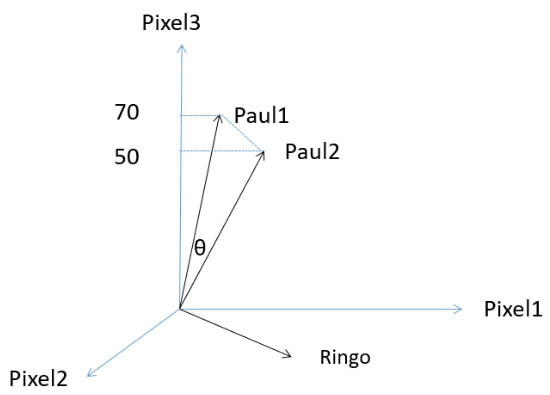
- Key idea: think of the images as *vectors*
  - Reminder: to turn an image into a vector, simply “flatten” all the pixels into a 1D vector
- Is the distance between the endpoints of vectors a and b small?

$$|a - b| = \sqrt{\sum_i (a_i - b_i)^2} \text{ small}$$

- Is the cosine of the angle between the vectors a and b large?

By the law of cosines

$$\cos\theta_{ab} = \frac{a \cdot b}{|a||b|} = \frac{\sum_i a_i b_i}{\sqrt{\sum_i a_i^2}\sqrt{\sum_i b_i^2}} \text{ large}$$



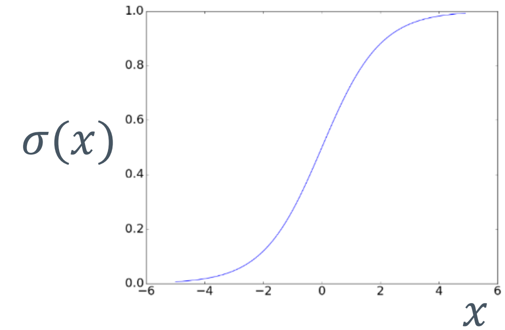
- Is $a \cdot b = \sum_i a_i b_i$ large?
  - Assume $|a| \approx |b| \approx const$

SML310 Project 4 task

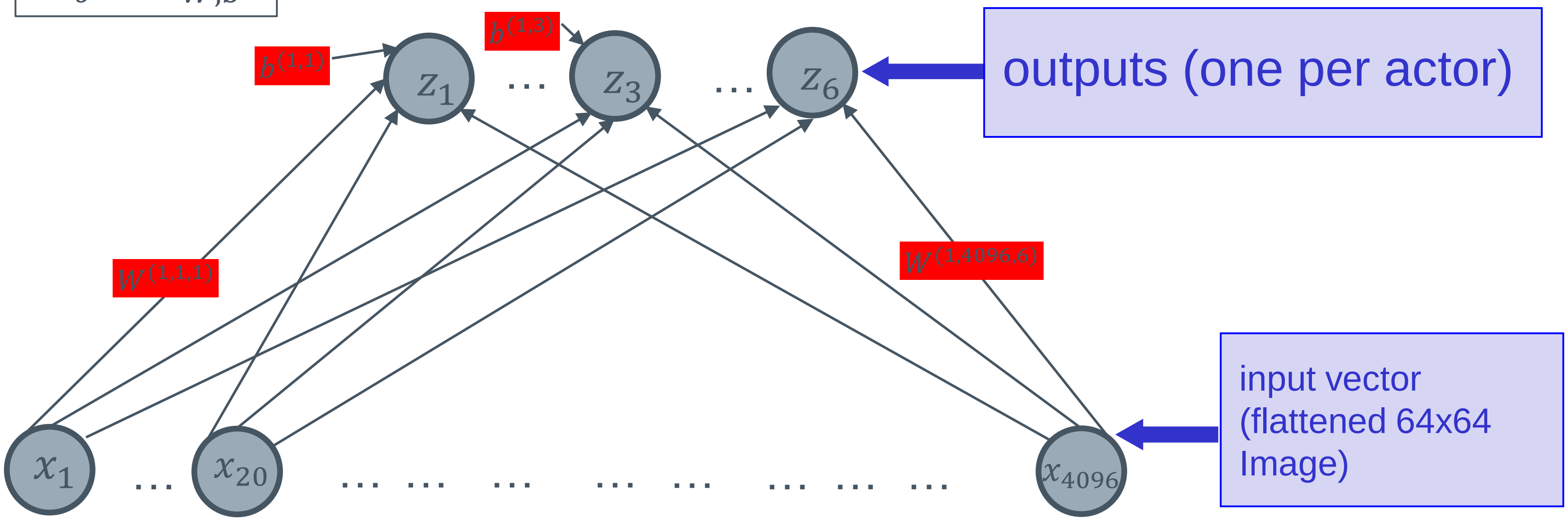
- Training set: 6 actors, with 100 64×64 photos of faces for each
- Test set: photos of faces of the same 6 actors
- Want to classify each face as one of ['Fran Drescher', 'America Ferrera', 'Kristin Chenoweth', 'Alec Baldwin', 'Bill Hader', 'Steve Carell']

The Simplest Possible Neural Network for Face Recognition

$$z_k = \sigma\left(\sum_{j=1}^{4096} W^{(1,j,k)}\, x_j + b^{(1,k)}\right)$$
$$= \sigma\left(W^{(1,*,k)} \cdot x + b^{(1,k)}\right)$$



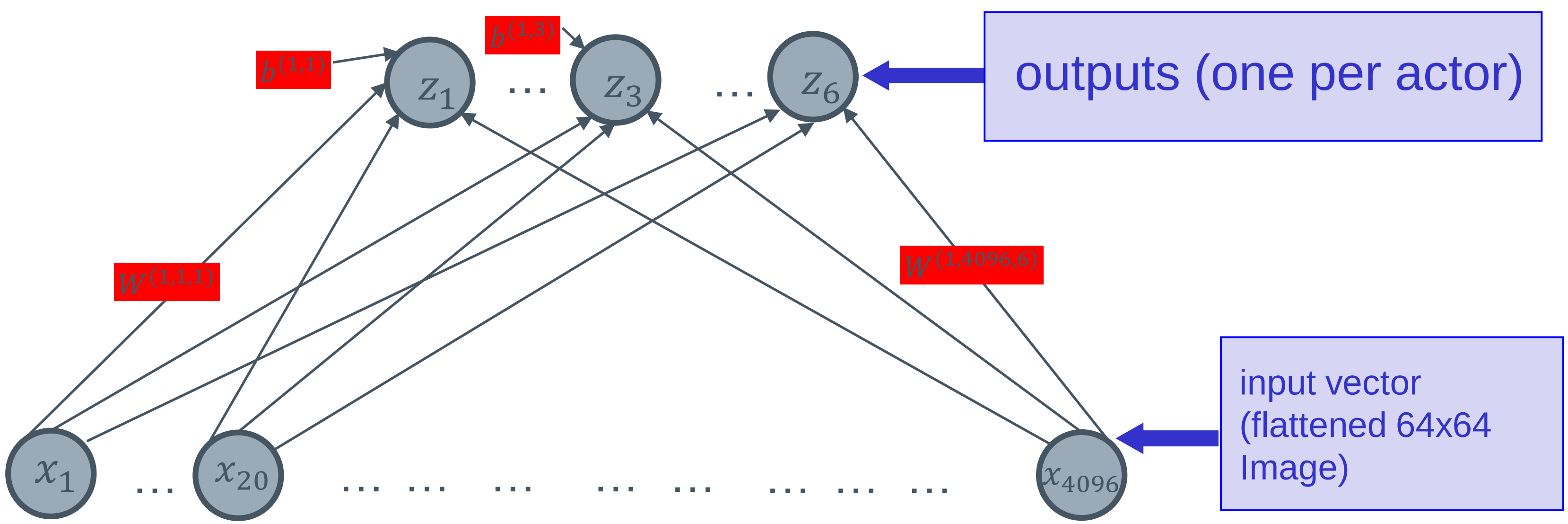
$$\boxed{h_\theta = h_{W,b}}$$



The transformation with σ is not necessary here, but will be useful later

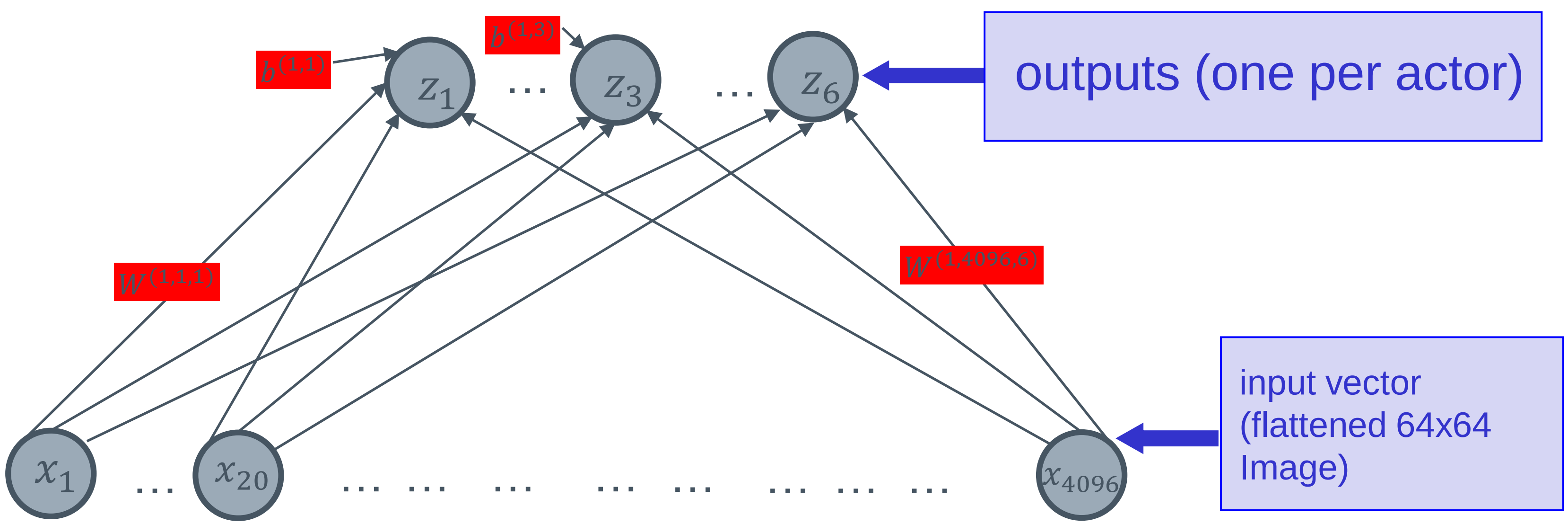
Training a neural network

› Adjust the W's (4096×6 coefs) and b's (6 coefs)
  – Try to make it so that if
    x is an image of actor 1, z is as close as possible to (1, 0, 0, 0, 0, 0)
    x is an image of actor 2, z is as close as possible to (0, 1, 0, 0, 0, 0)
    ……



Face recognition

- Compute the z for a new image x
- If z_k is the largest output, output name *k*



An interpretation

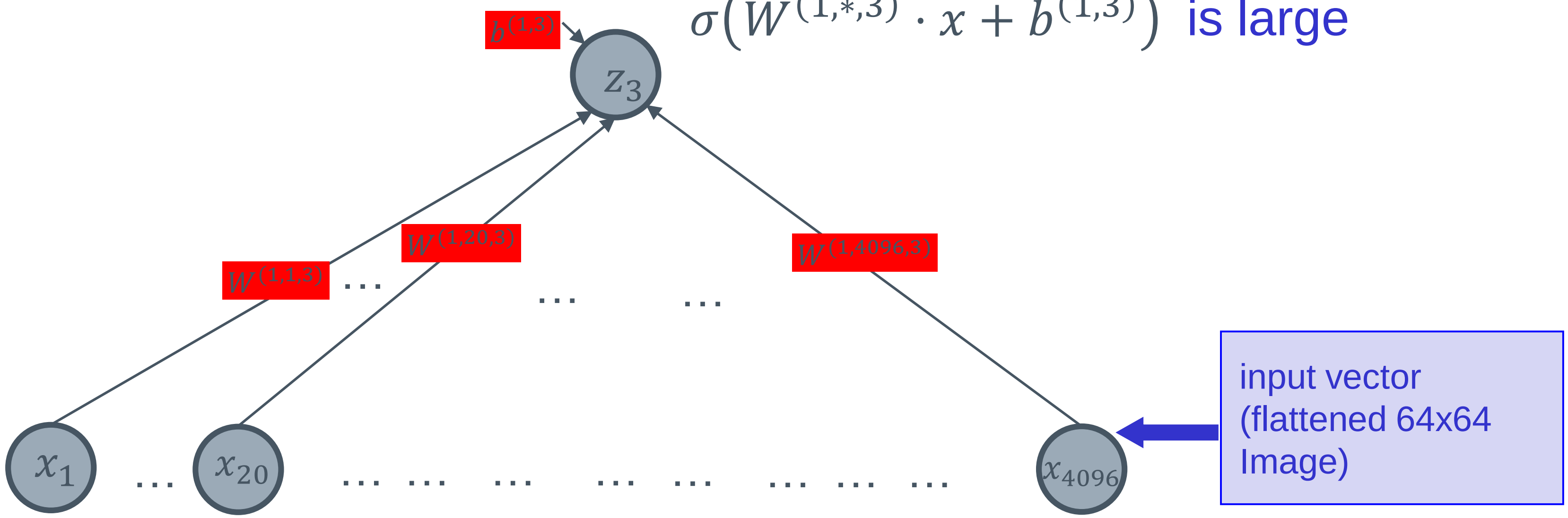
z_1 is large if $W^{(1,*,1)} \cdot x$ is large
z_2 is large if $W^{(1,*,2)} \cdot x$ is large

z_3 is large if $W^{(1,*,3)} \cdot x$ is large

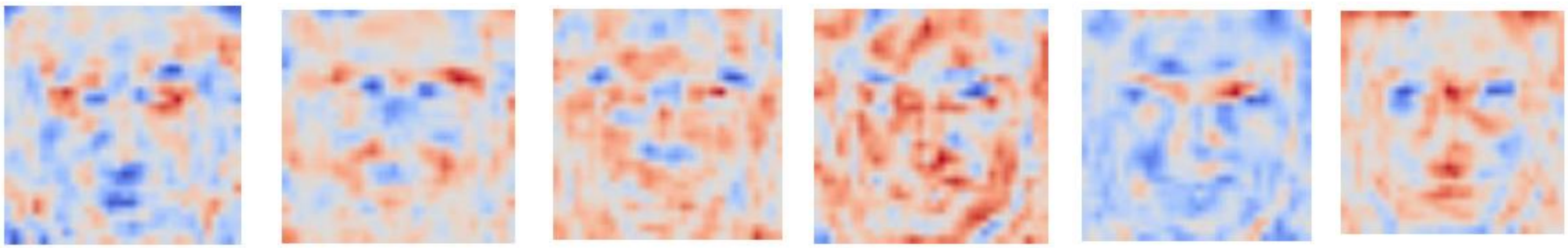
….

$W^{(1,*,1)}, W^{(1,*,2)}, \ldots, W^{(1,*,6)}$ are *templates* for the faces of actor 1, actor 2, …, actor 6

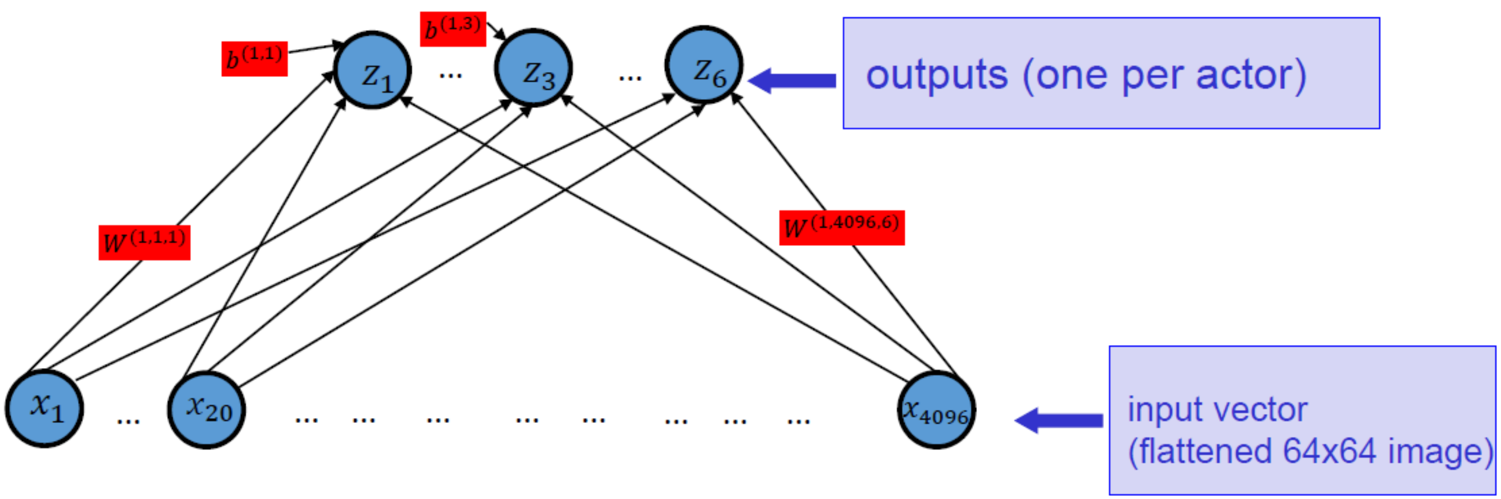
Actor 3 neuron activated:
$\sigma\left(W^{(1,*,3)} \cdot x + b^{(1,3)}\right)$ is large



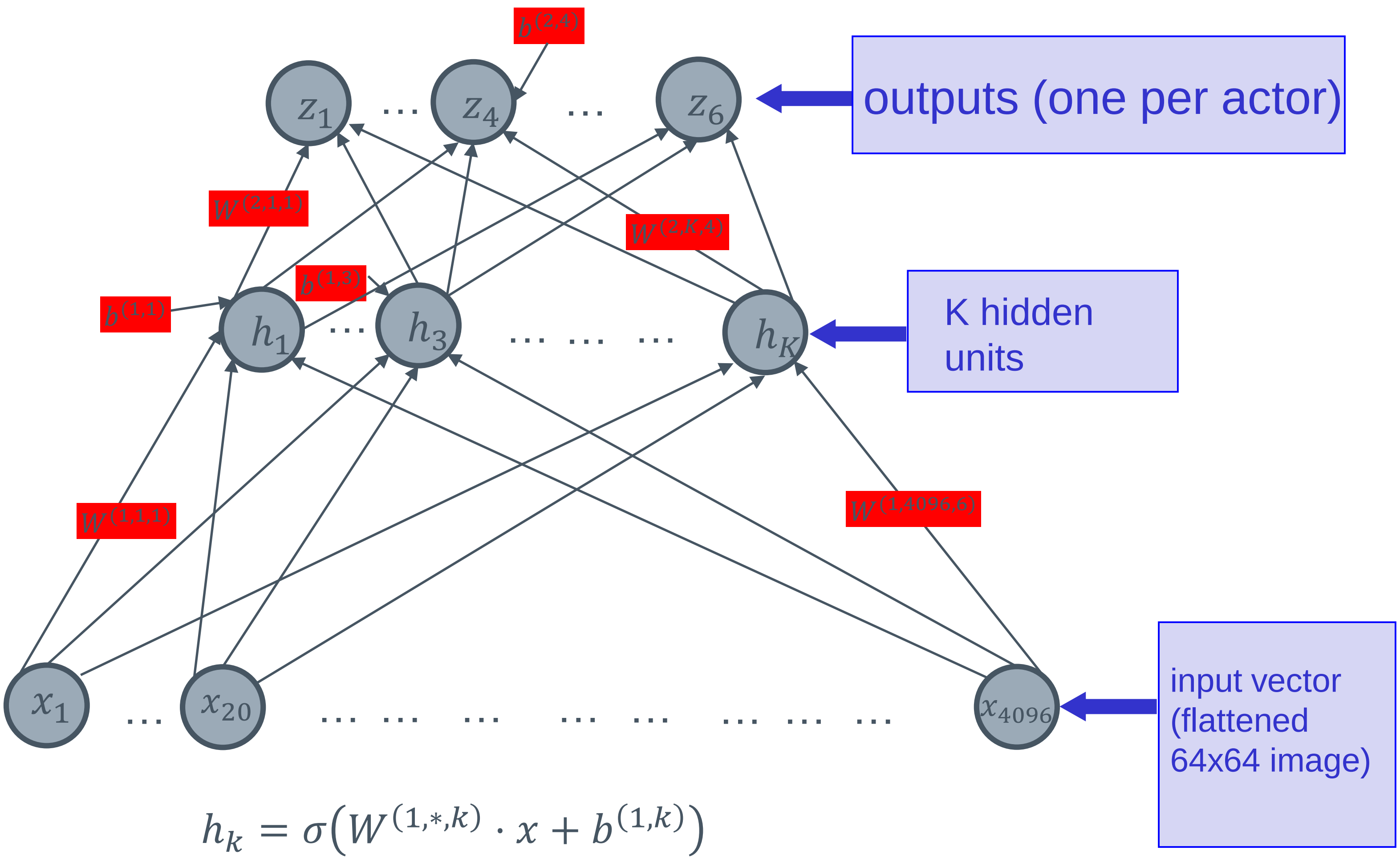
Visualizing the parameters W



Baldwin $W^{(1,*,1)}$ | Carrel $W^{(1,*,2)}$ | Hader $W^{(1,*,3)}$ | Ferrera $W^{(1,*,4)}$ | Drescher $W^{(1,*,5)}$ | Chenoweth $W^{(1,*,6)}$



Deep Neural Networks: Introducing Hidden Layers

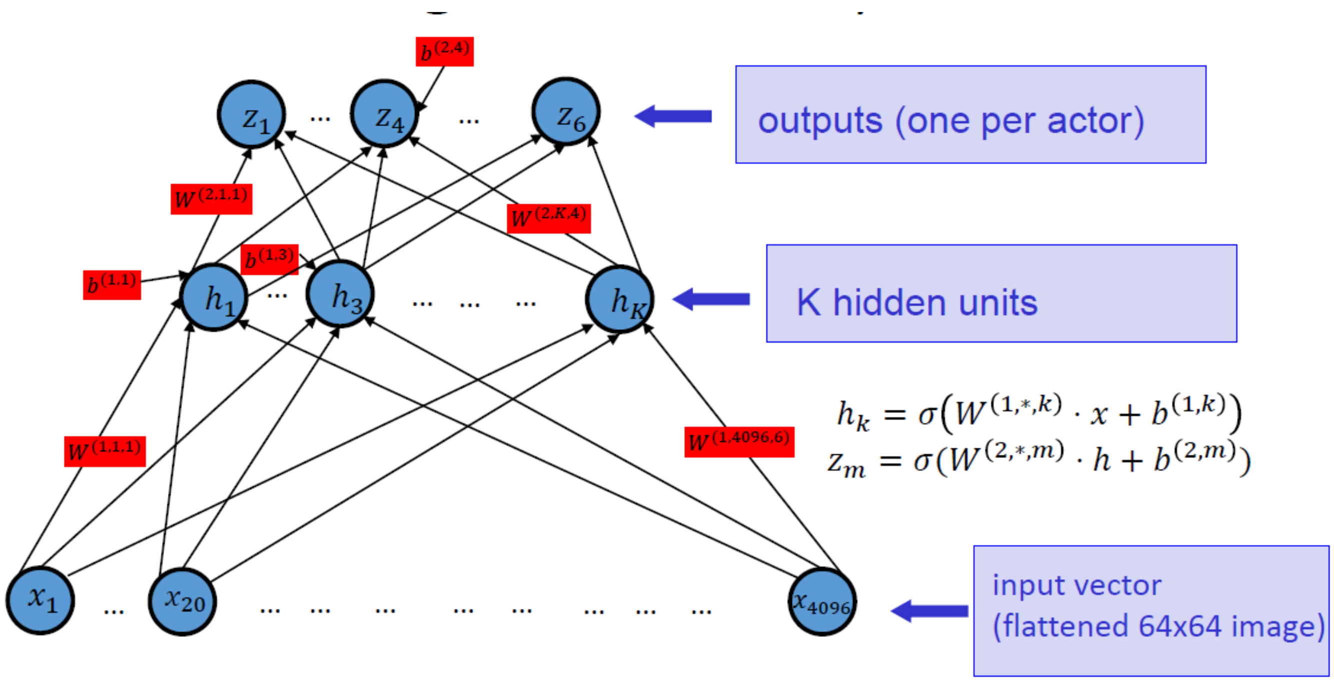


$$h_k = \sigma\left(W^{(1,*,k)} \cdot x + b^{(1,k)}\right)$$

$$z_m = \sigma(W^{(2,*,m)} \cdot h + b^{(2,m)})$$

Why a hidden layer?

- Instead of checking whether x looks like one of 6 templates, we'll be checking whether x looks like one of K templates, for a large K
  - If template k (i.e., $W^{(1,*,k)}$) looks like actor 6, $W^{(2,k,6)}$ will be large



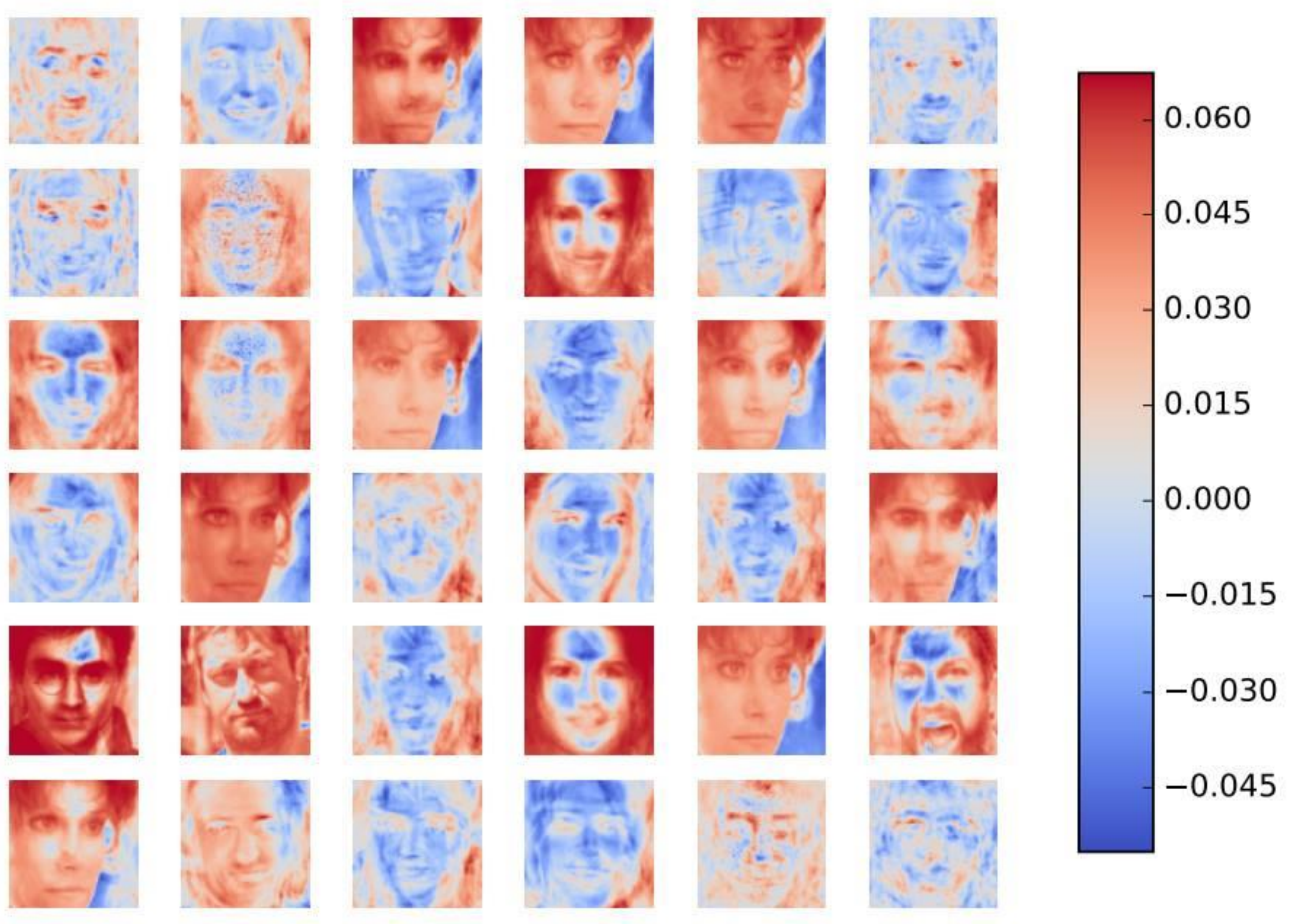
Recap: Face Recognition with ML

- 1-Nearest-Neighbor: match x to all the images in the training set
- 0-hidden-layer neural network*: match x to several templates, with one template per actor
  - The templates work better than any individual photo
- 1-hidden-layer neural network: match x to K templates
  - The templates work better than any individual photo
  - More templates means better accuracy on the training set

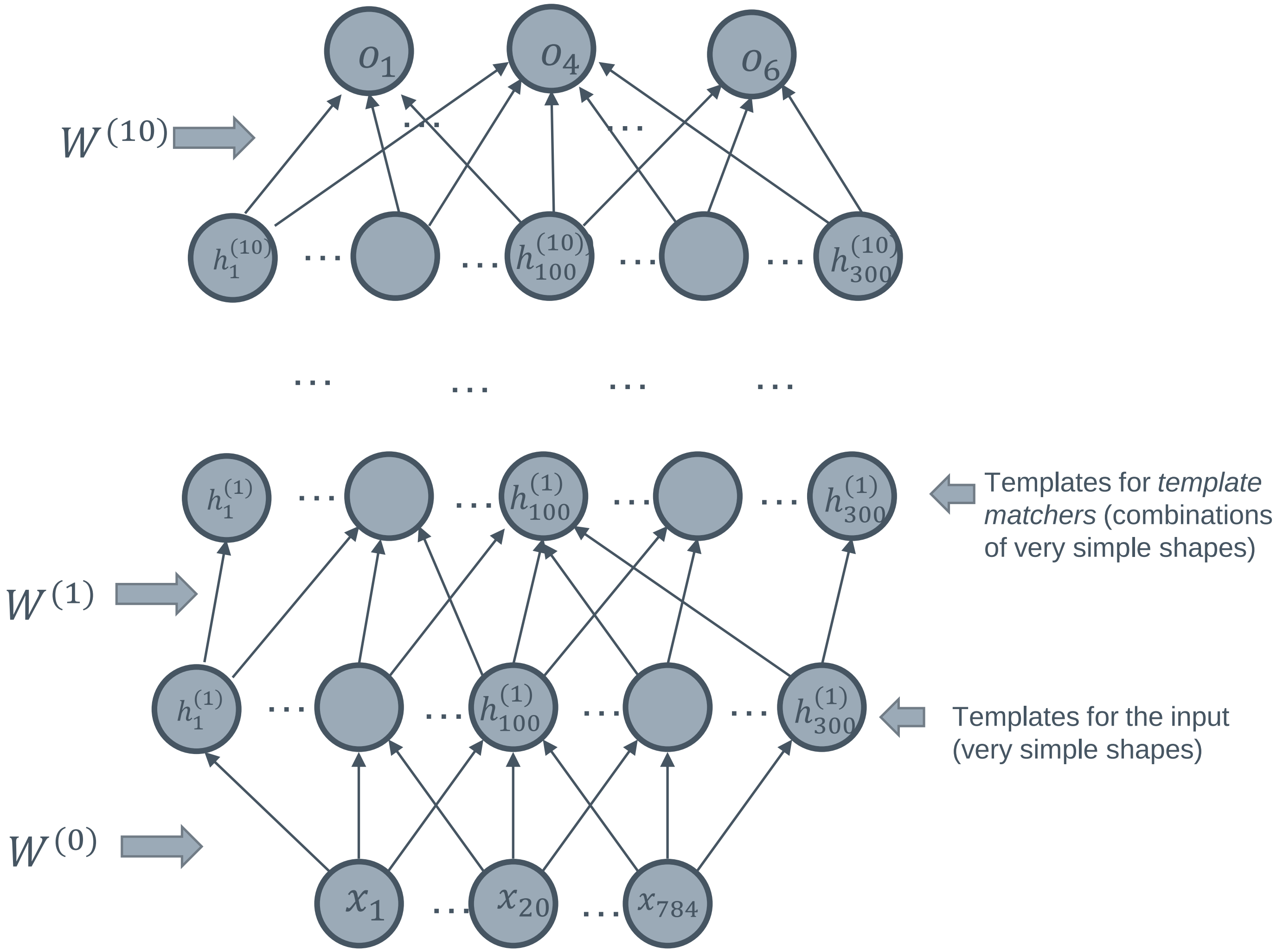
*A.K.A. multinomial logistic regression

** With minor modifications made to make this lecture clearer

Visualizing a One-Hidden-Layer NN



Deep Learning: More hidden layers!



Deep Neural Networks as a Model of Computation

- Most people's first instinct when building a face classifier is to write a complicated computer program
- A deep neural network *is* a computer program:

```
h1 = f1(x)
h2 = f2(h1)
h3 = f3(h2)
…
h9 = f9(h8)
```

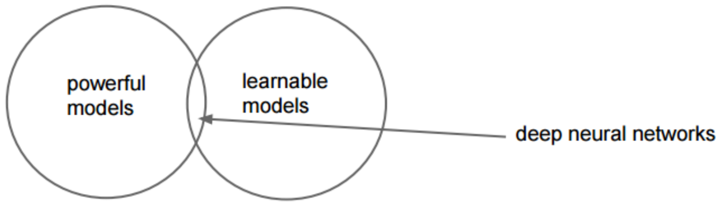
- Can think of every layer of a neural network as one step of a parallel computation
- Features/templates are the functions that are applied to the previous layers
- Learning features ⇔ Learning what function to apply at step t of the algorithm

Deep Neural Networks

› Can perform a wide range of computation

› Can be learned automatically

– (using gradient descent)



- Powerful but not (computer) learnable: R/Java/Python
  - Can't make a learning R/Java/Python
  - that takes lots of inputs and outputs and produces Python code that generates the outputs on new inputs
    - (But can do it with simpler languages!)
- Learnable but not powerful:
  - Logistic regression
  - Deep Neural Networks that aren't deep enough

Graphic and idea by Ilya Sutskever

The *Deep Learning Hypothesis*

- Human perception is fast
  - (Human) neurons fire at most 100 times a second
  - Humans can solve simple perceptual tasks in 0.1 seconds
    - So out neurons fire in a sequence of 10 times at most

> Anything a human can do in 0.1 seconds, a big 10-layer neural network can do, too!

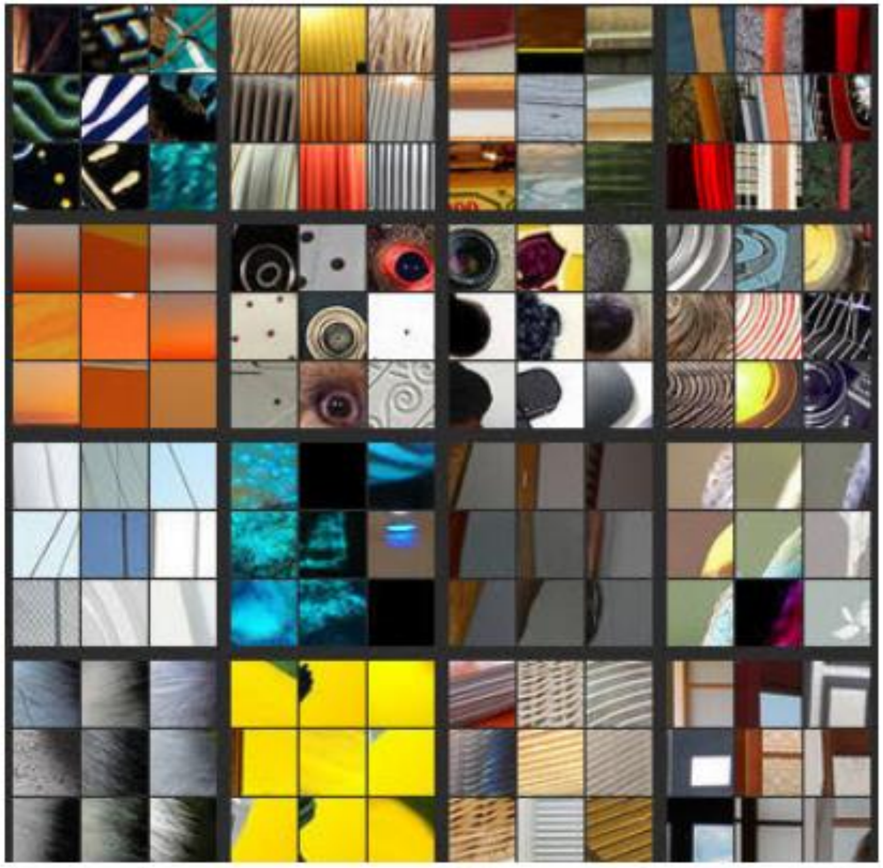
- Success stories:
  - Classifying images of objects
  - Classifying Go positions as good or bad

What are the hidden units doing?

What are the hidden units doing?

- Find the images in the dataset that activate the units the *most*
- *Let’s see some visualizations of neurons of a large deep network trained to recognize objects in images*
  - The network classifies images as one of 1000 objects (sample objects: toy poodle, flute, forklift, goldfish…)
  - The network has 8 layers
  - Note: more tricks were used in designing the networks than we have time to mention. In particular, a *convolutional* architecture is crucial

Units in Layer 3



Matthew Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks” (ECCV 2014)

Units in Layer 4

Matthew Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks” (ECCV 2014)

Units in Layer 5

Matthew Zeiler and Rob Fergus, “Visualizing and Understanding Convolutional Networks” (ECCV 2014)

Questions?