# Neural Networks I

WIN 2019-20

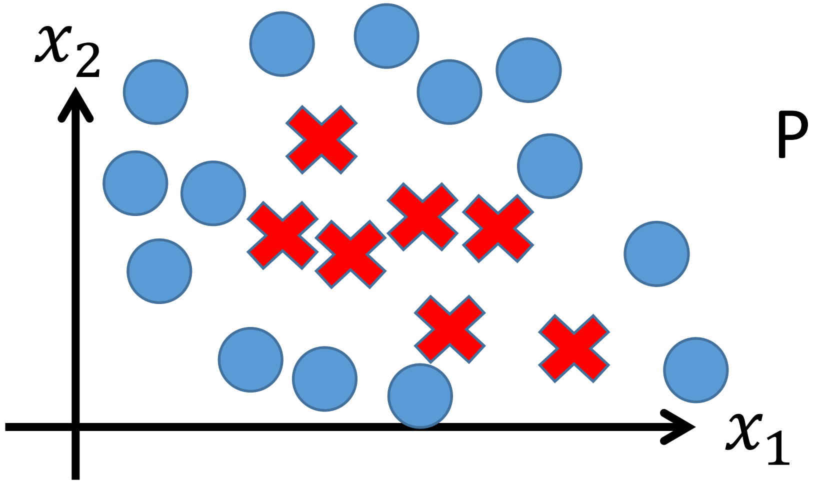
# Neural Networks

- Why neural networks?
- Model representation
- Examples and intuitions
- Multi-class classification
- Back propagation algorithm
- Implementation & Applications

# Neural Networks

- **Why neural networks?**

# Non-linear classification




Predict $y = 1$ if

$$g(\theta_0 + \theta_1 x_1 + \theta_2 x_2 + \theta_3 x_1 x_2 + \theta_4 x_1^2 + \theta_5 x_2^2 + \cdots) \geq 0$$

$x_1$= size

$x_2$= #bedrooms

$x_3$= #floors

$x_4$= age

…

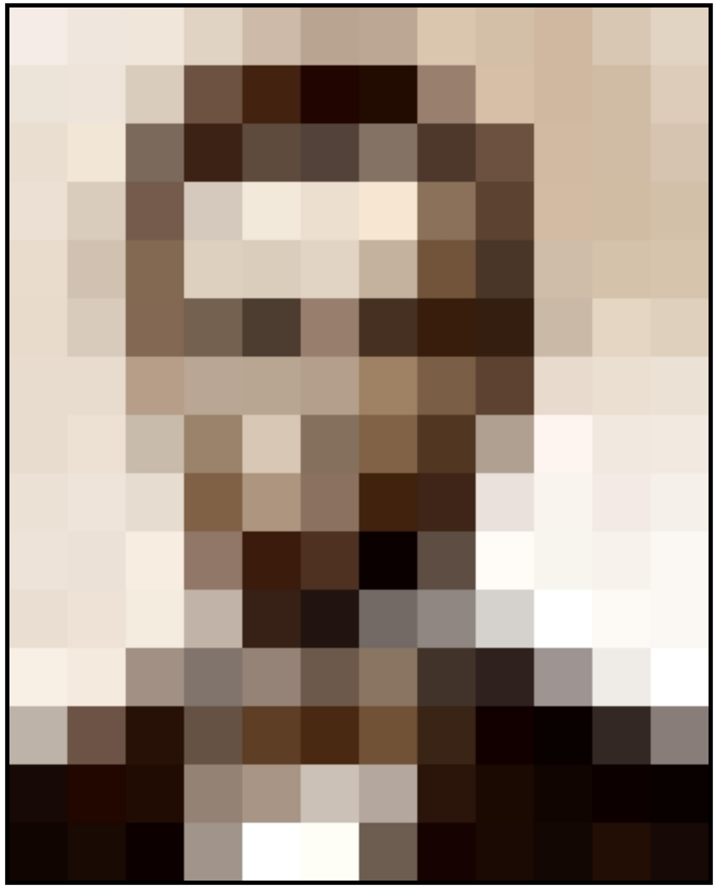
$x_{100}$

# Quadratic features?

# Cubic features?

Slide credit: Andrew Ng

# What humans see



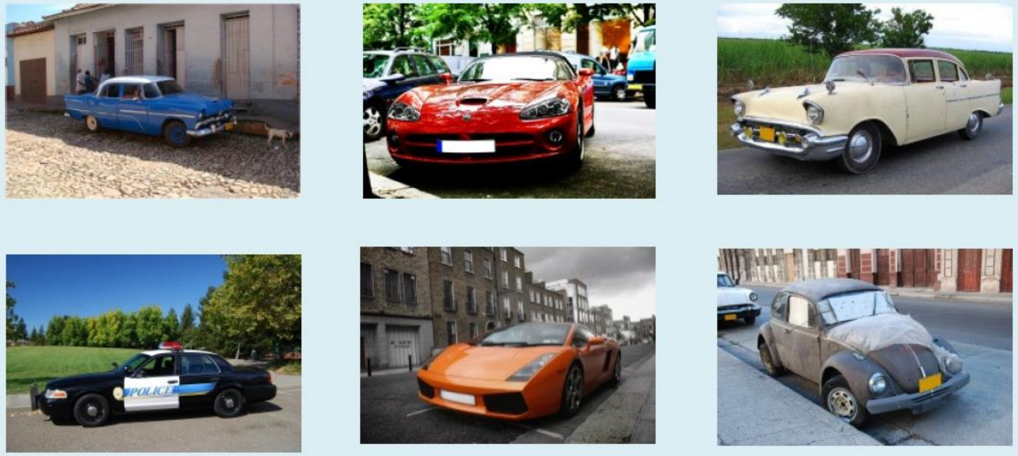
Slide credit: Larry Zitnick

# What computers see

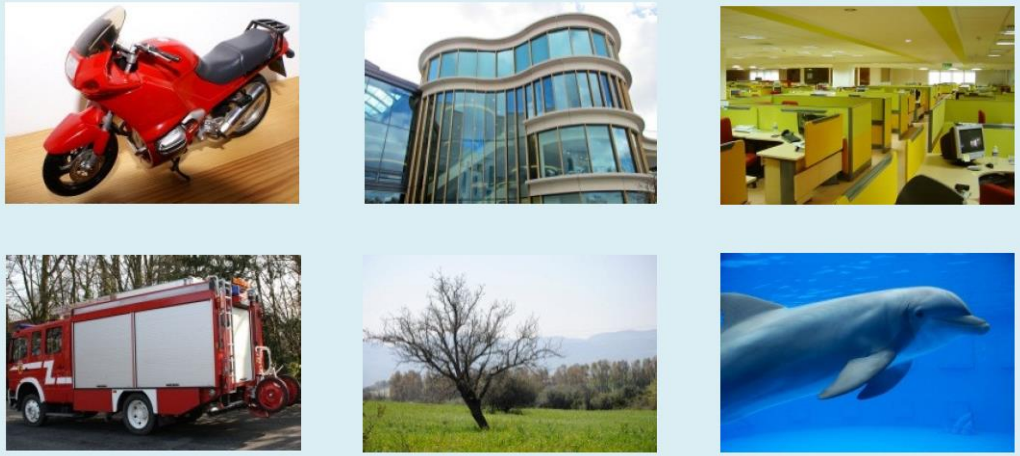
| | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| 243 | 239 | 240 | 225 | 206 | 185 | 188 | 218 | 211 | 206 | 216 | 225 |
| 242 | 239 | 218 | 110 | 67 | 31 | 34 | 152 | 213 | 206 | 208 | 221 |
| 243 | 242 | 123 | 58 | 94 | 82 | 132 | 77 | 108 | 208 | 208 | 215 |
| 235 | 217 | 115 | 212 | 243 | 236 | 247 | 139 | 91 | 209 | 208 | 211 |
| 233 | 208 | 131 | 222 | 219 | 226 | 196 | 114 | 74 | 208 | 213 | 214 |
| 232 | 217 | 131 | 116 | 77 | 150 | 69 | 56 | 52 | 201 | 228 | 223 |
| 232 | 232 | 182 | 186 | 184 | 179 | 159 | 123 | 93 | 232 | 235 | 235 |
| 232 | 236 | 201 | 154 | 216 | 133 | 129 | 81 | 175 | 252 | 241 | 240 |
| 235 | 238 | 230 | 128 | 172 | 138 | 65 | 63 | 234 | 249 | 241 | 245 |
| 237 | 236 | 247 | 143 | 59 | 78 | 10 | 94 | 255 | 248 | 247 | 251 |
| 234 | 237 | 245 | 193 | 55 | 33 | 115 | 144 | 213 | 255 | 253 | 251 |
| 248 | 245 | 161 | 128 | 149 | 109 | 138 | 65 | 47 | 156 | 239 | 255 |
| 190 | 107 | 39 | 102 | 94 | 73 | 114 | 58 | 17 | 7 | 51 | 137 |
| 23 | 32 | 33 | 148 | 168 | 203 | 179 | 43 | 27 | 17 | 12 | 8 |
| 17 | 26 | 12 | 160 | 255 | 255 | 109 | 22 | 26 | 19 | 35 | 24 |

Slide credit: Larry Zitnick

# Computer Vision: Car detection



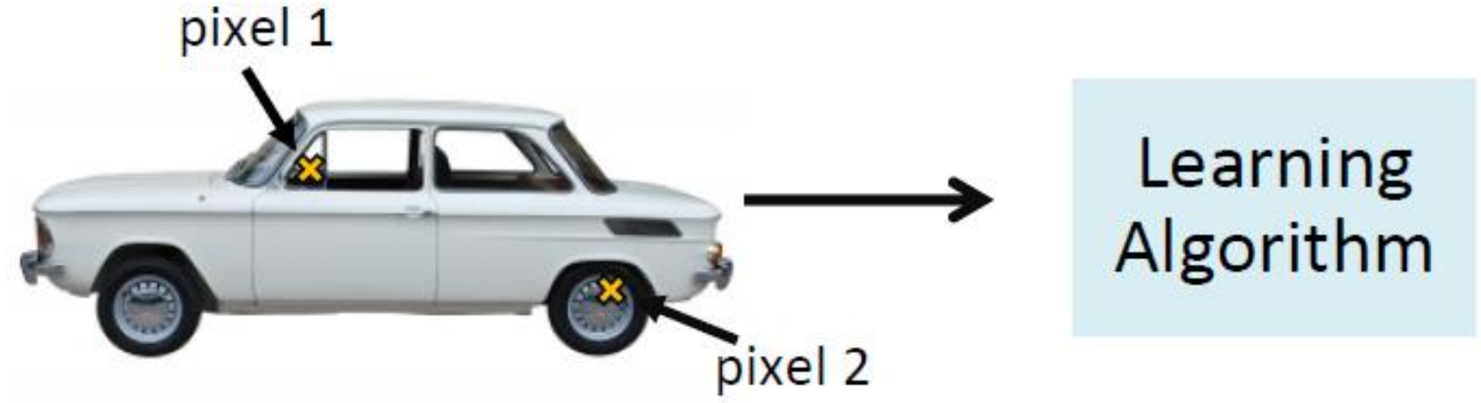
Cars

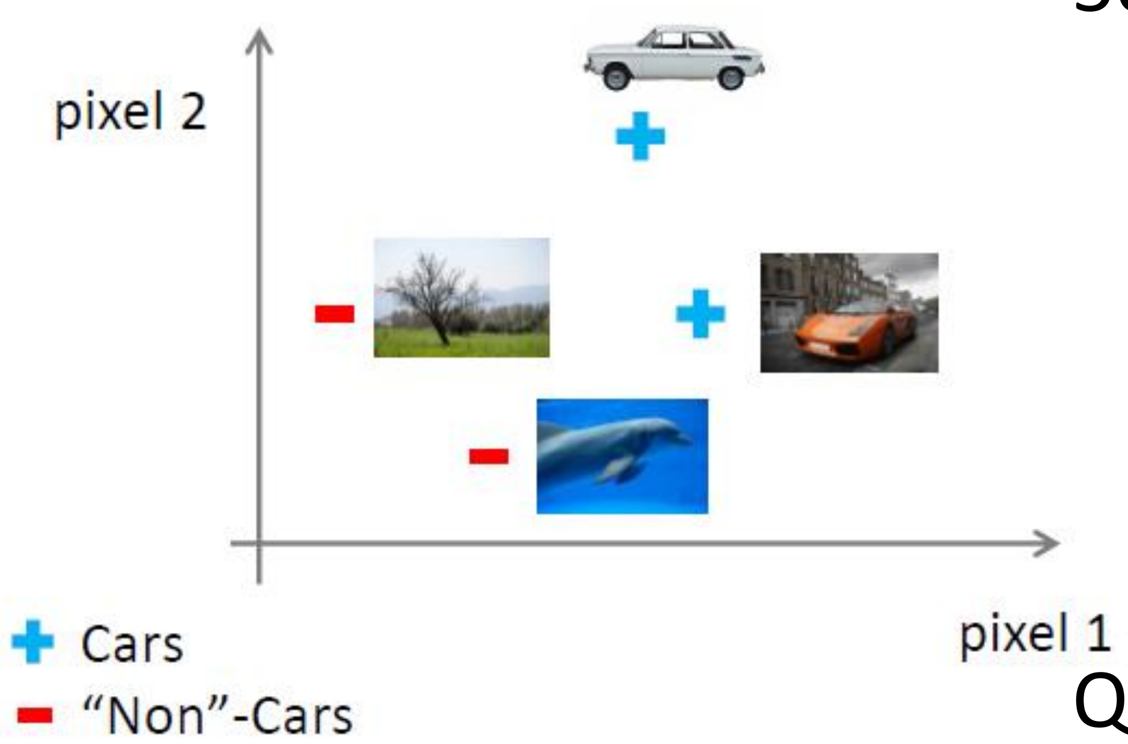


Not a car

Testing:

Slide credit: Andrew Ng




50x50 pixel images -> 2500 pixels

$$x = \begin{bmatrix} \text{pixel 1 intensity} \\ \text{pixel 2 intensity} \\ \vdots \\ \text{pixel 2500 intensity} \end{bmatrix}$$

Quadratic features $(x_i x_j)$~
3 million features

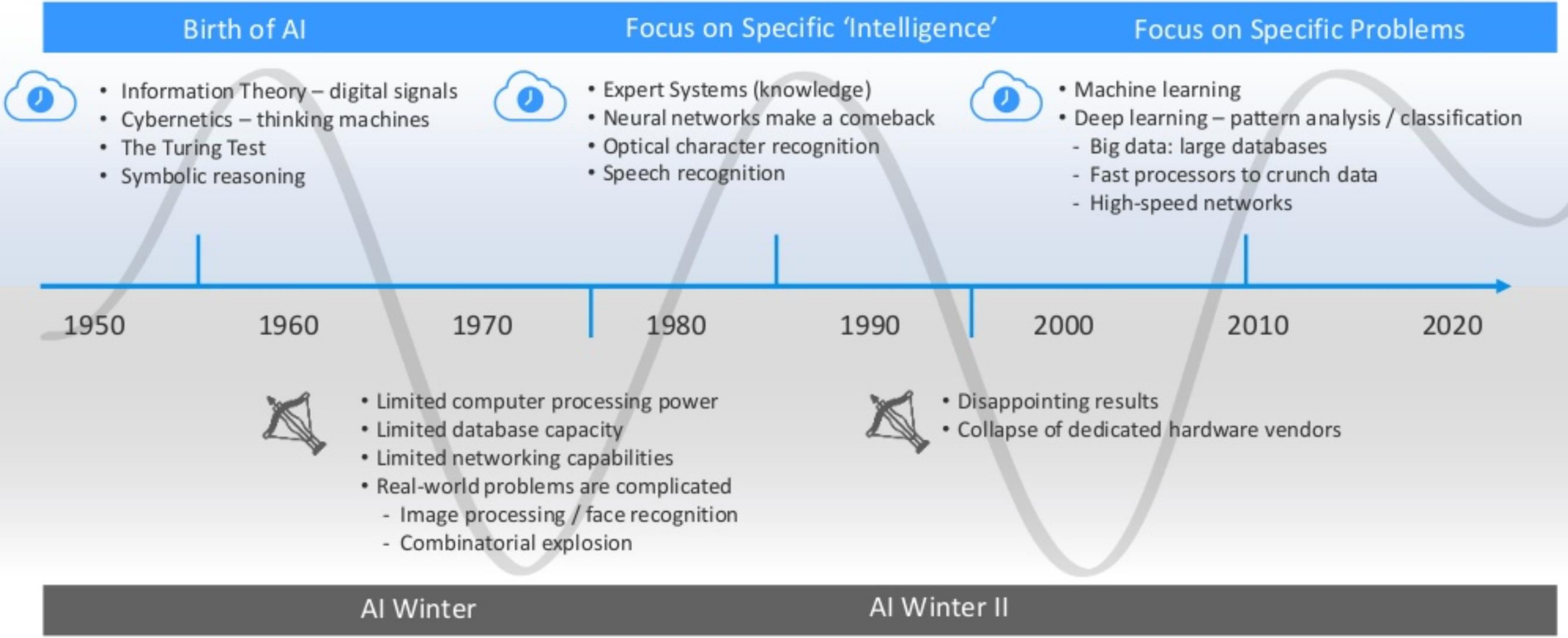
Slide credit: Andrew Ng

# Neural Networks

- Origins: Algorithms that try to mimic the brain.
- Was very widely used in 80s and early 90s; popularity diminished in late 90s.
- Recent resurgence: State-of-the-art technique for many applications

Slide credit: Andrew Ng

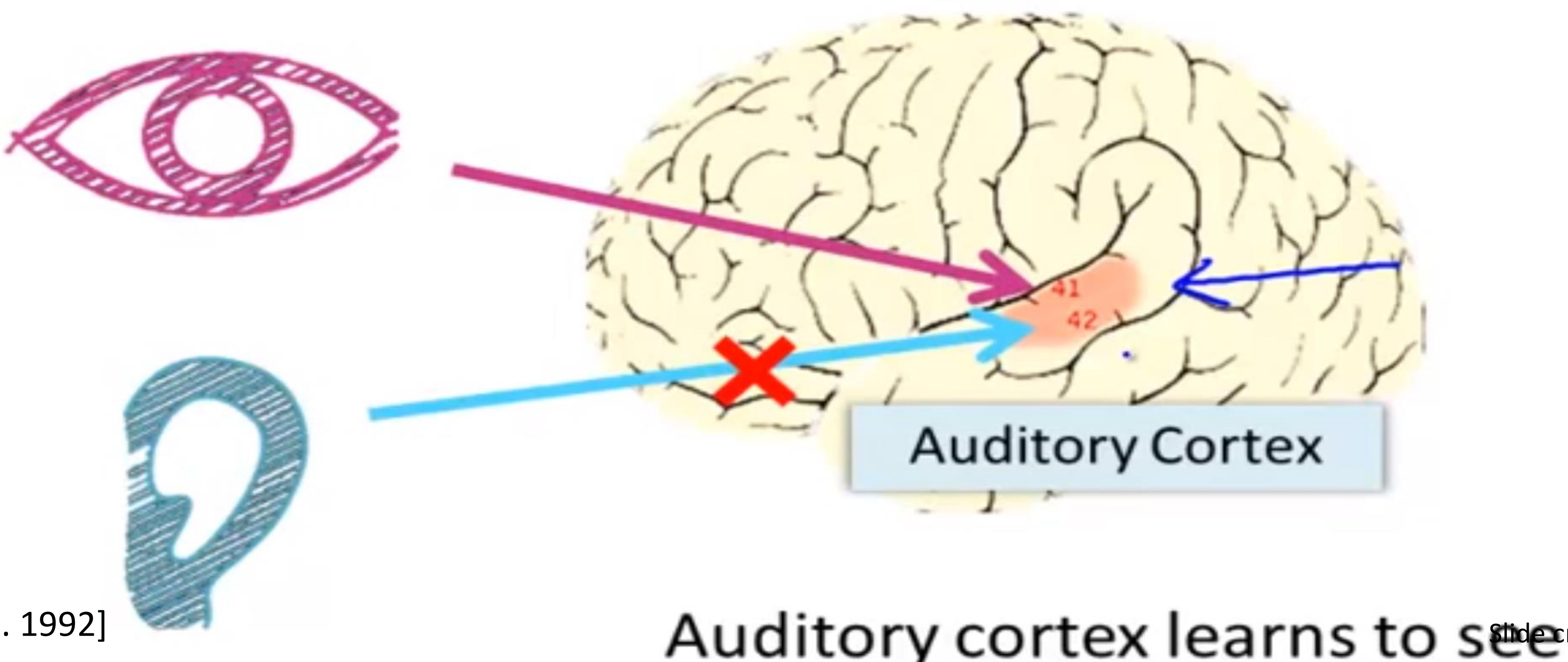
# An AI Timeline




https://www.slideshare.net/dlavenda/ai-and-productivity

# The "one learning algorithm" hypothesis

- Auditory cortex --> takes sound signals If you cut the wiring from the ear to the auditory cortex
- Re-route optic nerve to the auditory cortex
- Auditory cortex learns to see

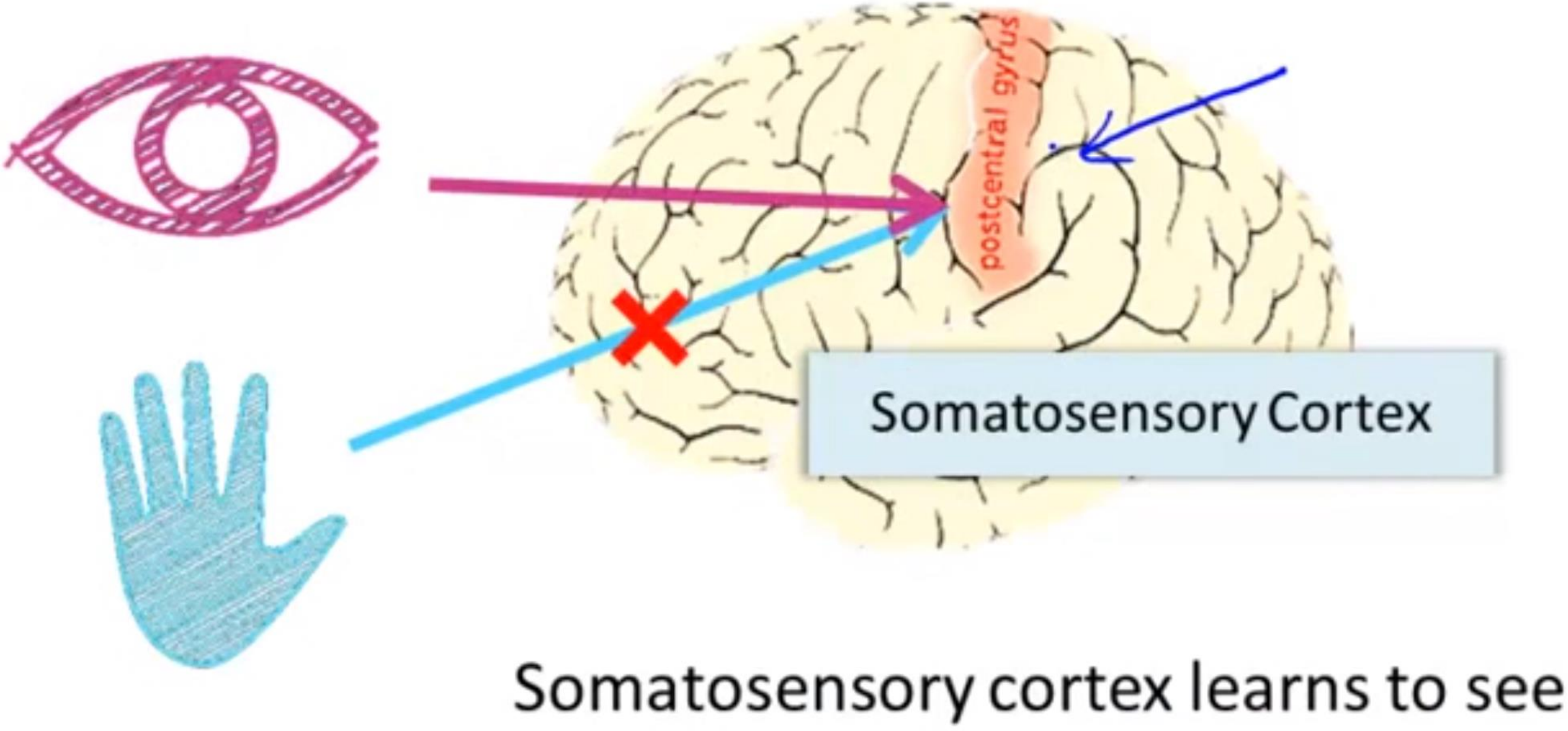



[Roe et al. 1992]

Slide credit: Andrew Ng

# The “one learning algorithm” hypothesis

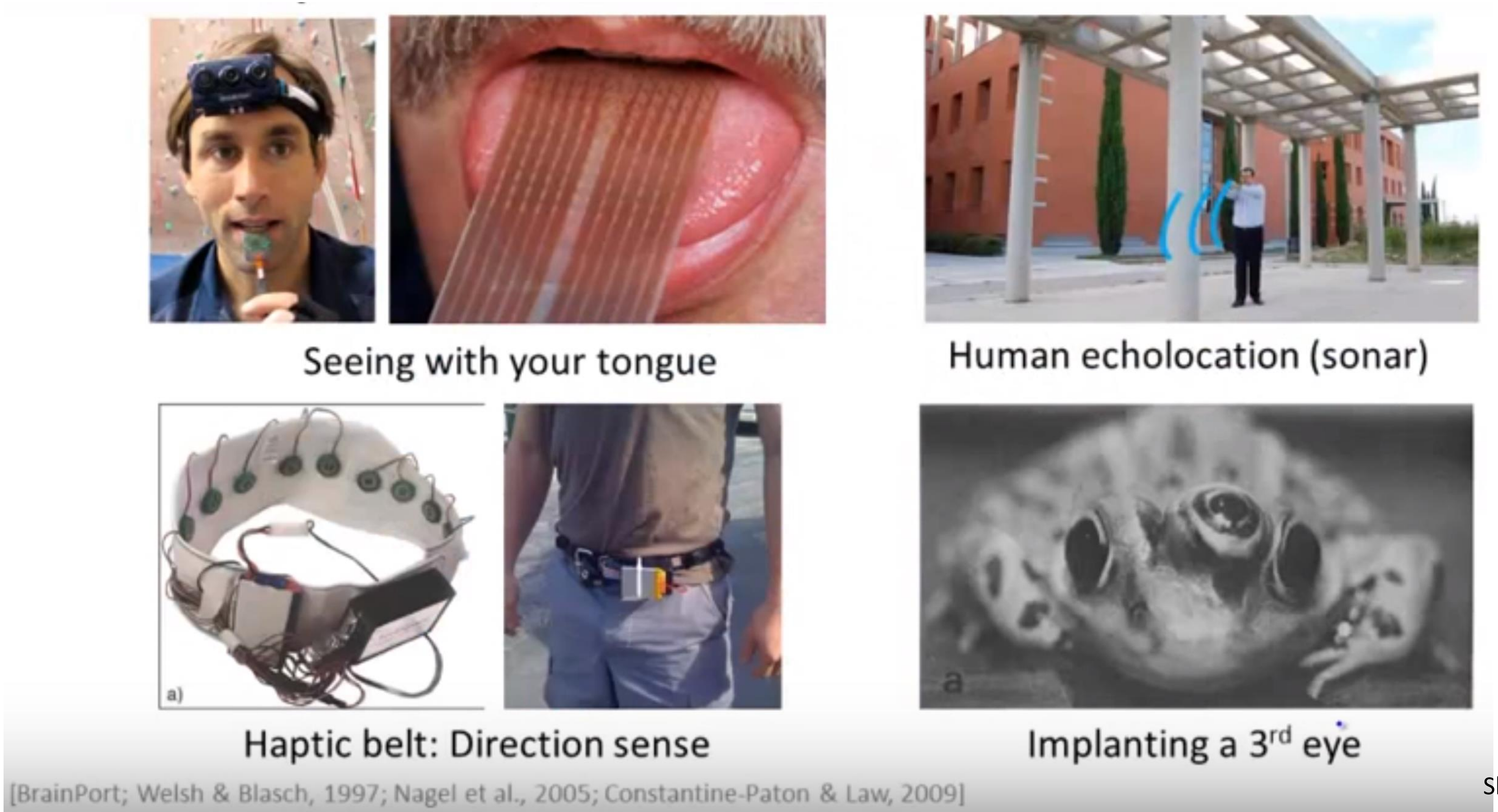
- Somatosensory context (touch processing)
- If you rewrite optic nerve to somatosensory cortex then it learns to see




[Metin and Frost 1989]

Slide credit: Andrew Ng

# Sensor representations in the brain

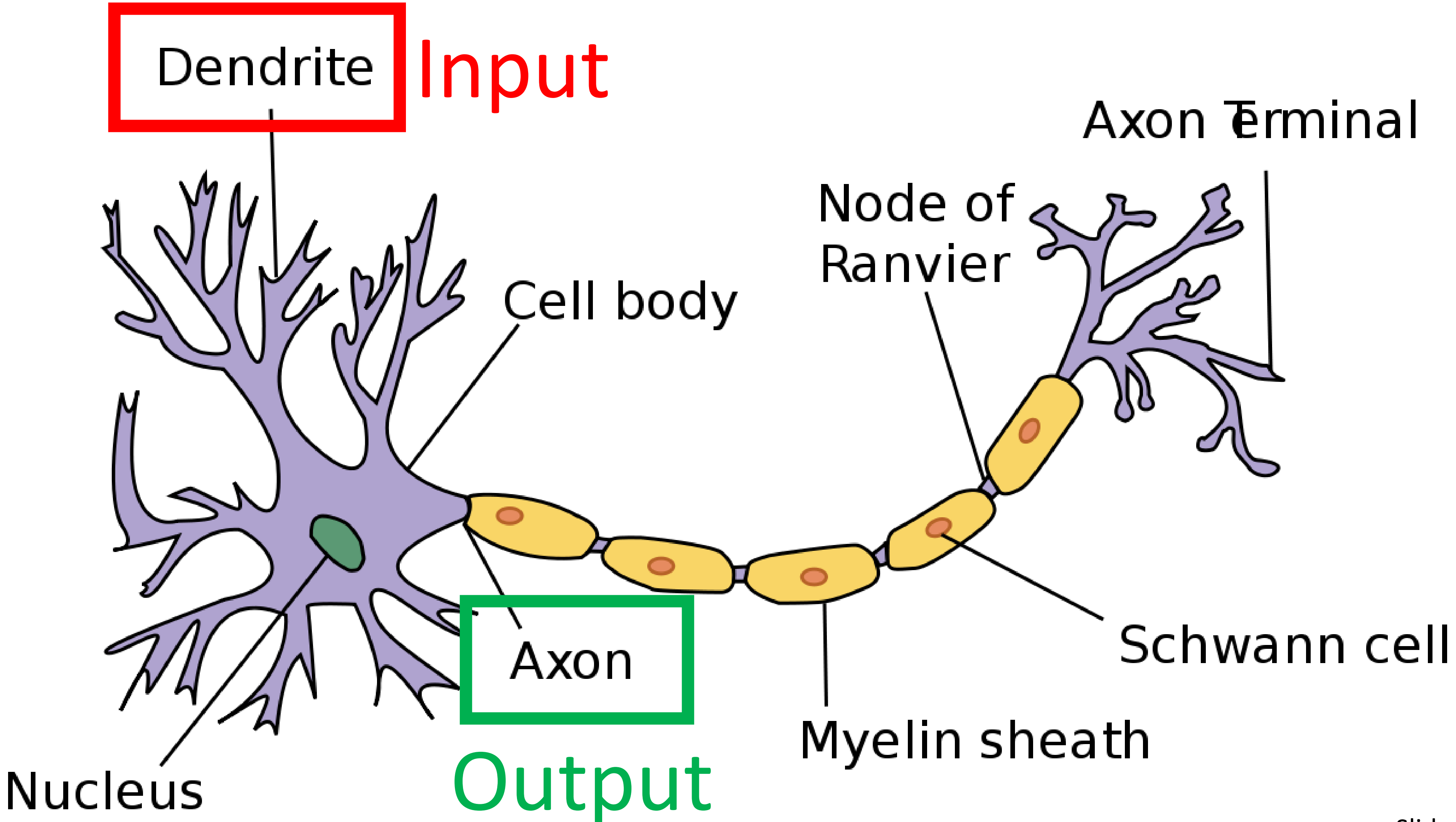



Slide credit: Andrew Ng

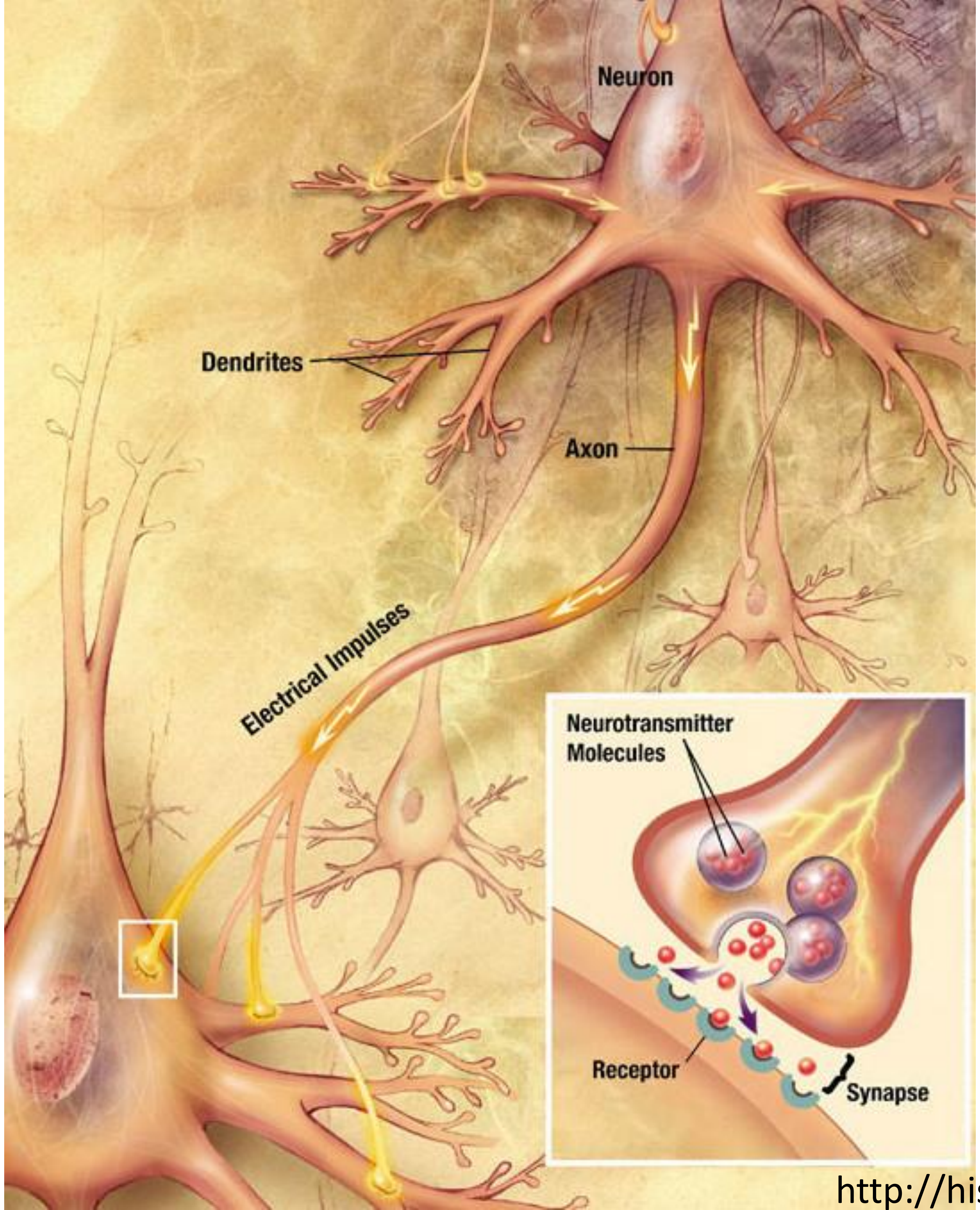
# Neural Networks

- Why neural networks?
- **Model representation**
- Examples and intuitions
- Multi-class classification

# A single neuron in the brain

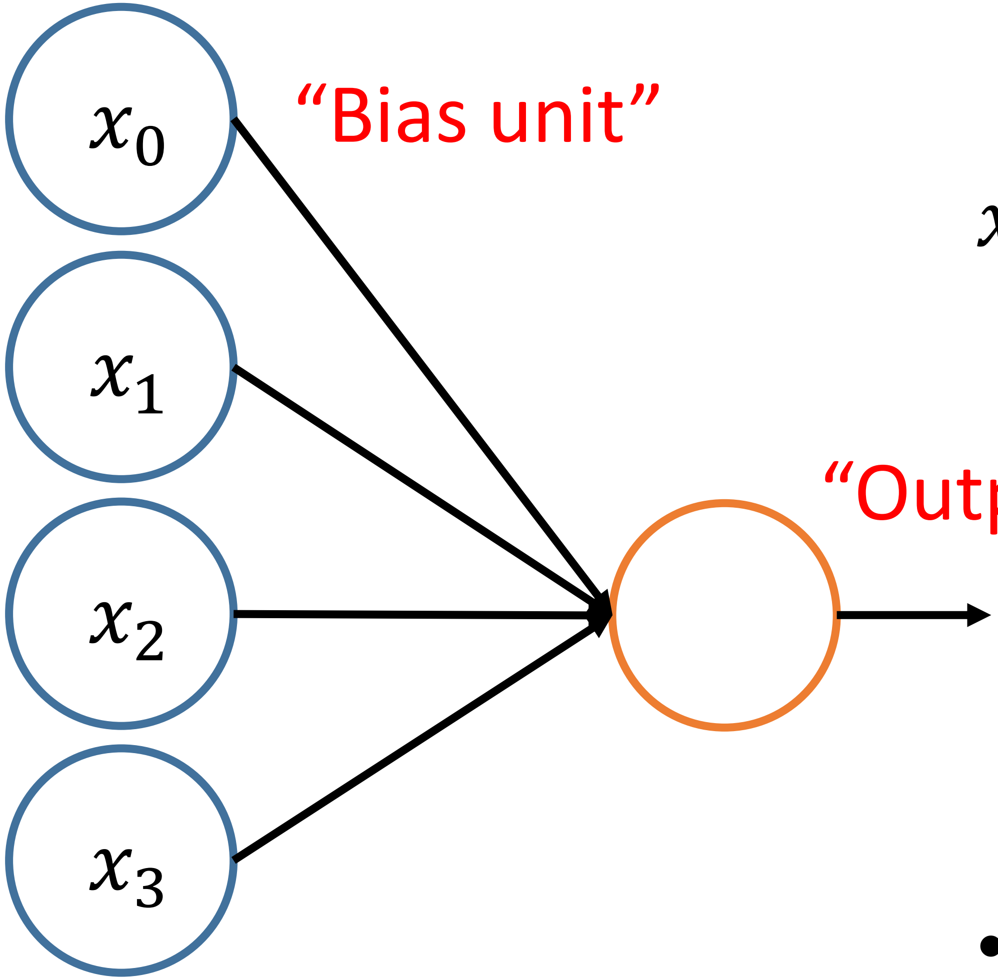



Slide credit: Andrew Ng




http://histoweb.co.za/082/082img002.html

# An artificial neuron: Logistic unit




$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \theta = \begin{bmatrix} \theta_0 \\ \theta_1 \\ \theta_2 \\ \theta_3 \end{bmatrix}$$

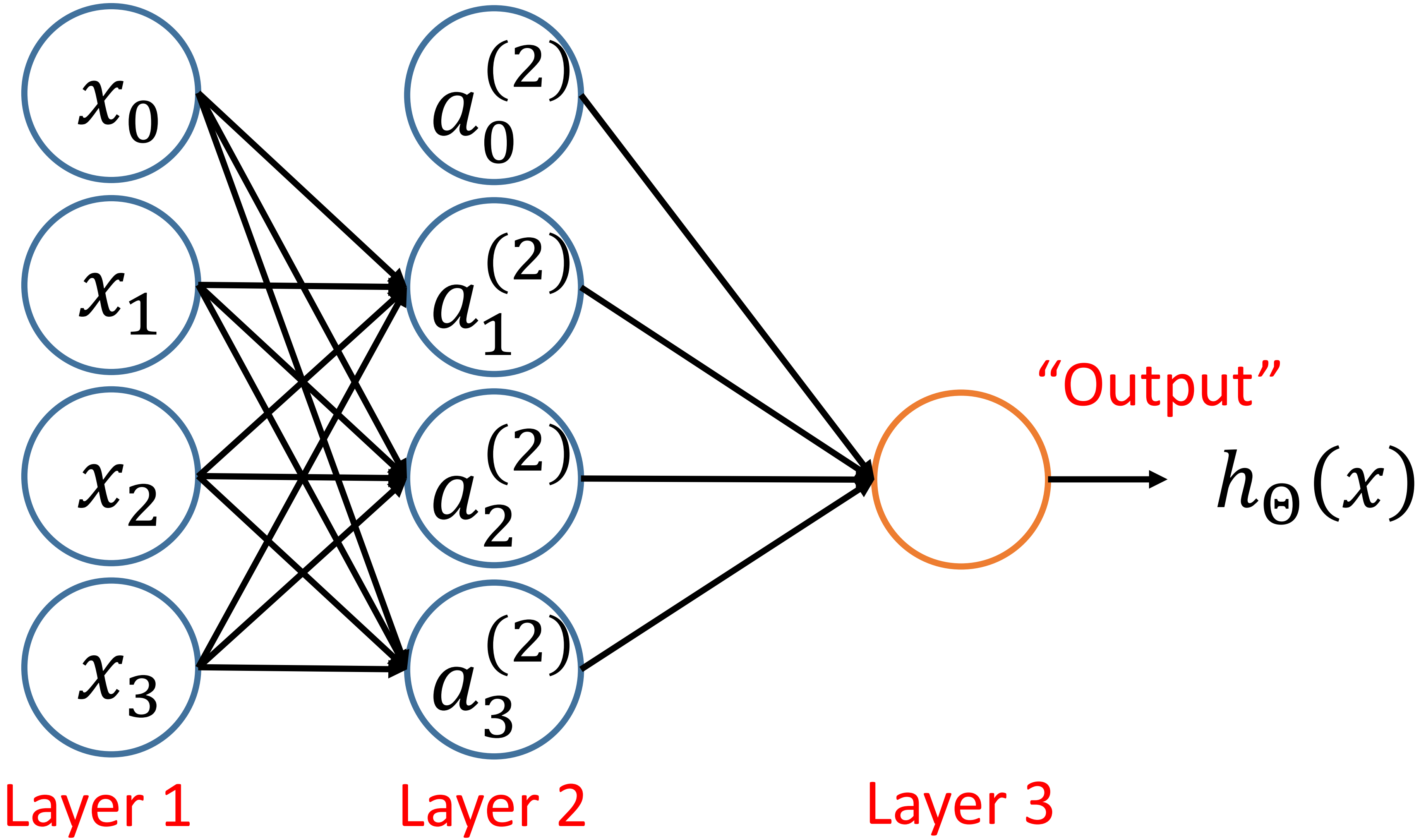
"Weights"
"Parameters"

$$h_\theta(x) = \frac{1}{1 + e^{-\theta^\top x}}$$

- Sigmoid (logistic) activation function

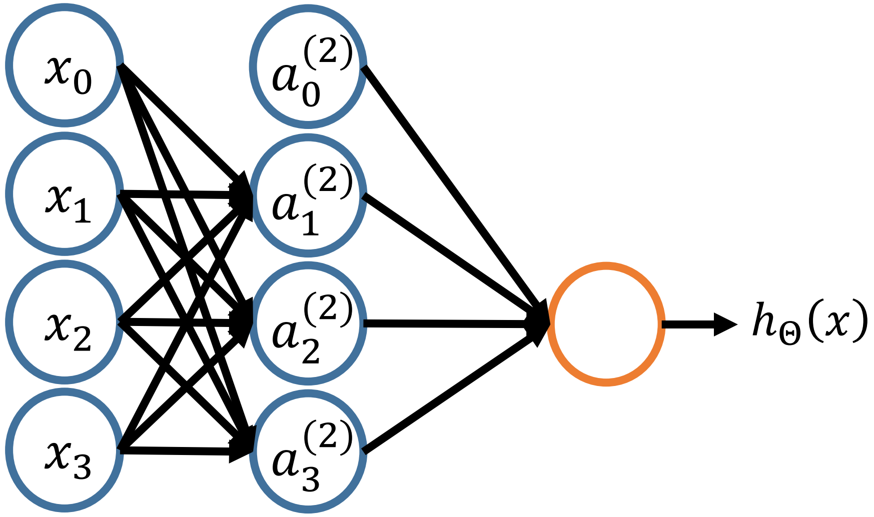
Slide credit: Andrew Ng

# Neural network




Slide credit: Andrew Ng

# Neural network



$a_i^{(j)}$ = "activation" of unit $i$ in layer $j$

$\Theta^{(j)}$ = matrix of weights controlling function mapping from layer $j$ to layer $j+1$

If networks has :

$s_j$ unit in layer $j$

$s_{j+1}$ units in layer $j+1$

Size of $\Theta^{(j)}$?

$s_{j+1} \times (s_j + 1)$

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3\right)$$

$$a_2^{(2)} = g\left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\right)$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3\right)$$

$$h_\Theta(x) = g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(2)} a_1^{(2)} + \Theta_{12}^{(2)} a_2^{(2)} + \Theta_{13}^{(2)} a_3^{(2)}\right)$$

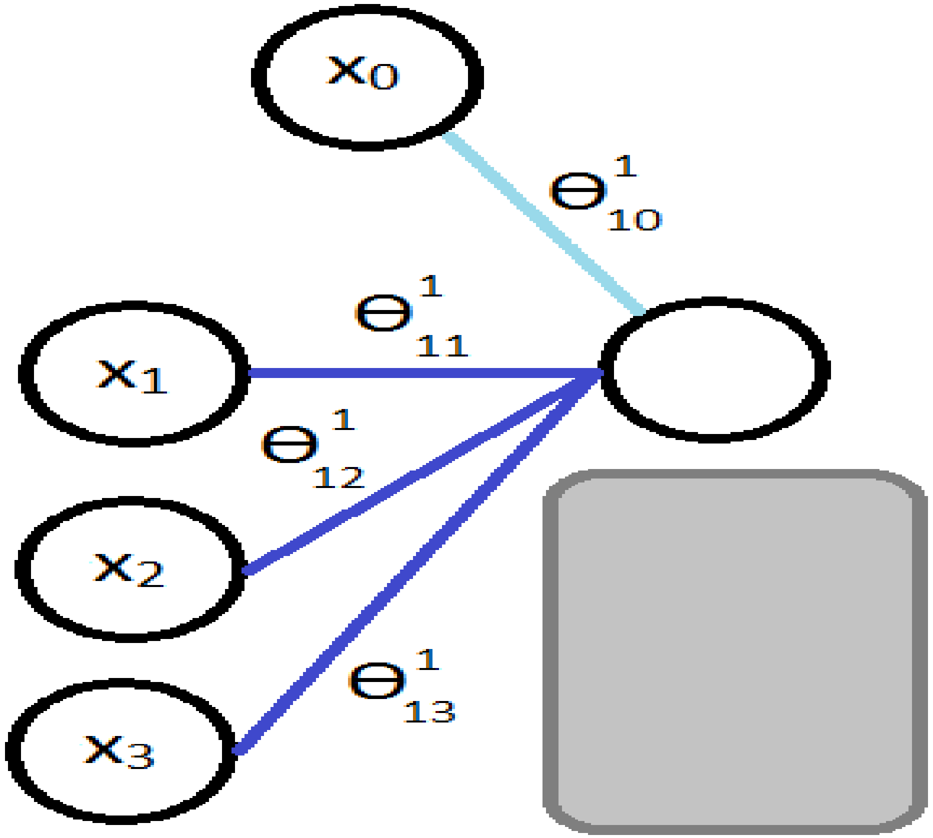
Slide credit: Andrew Ng

# Neural network

- Then $\Theta^j$ will be of dimensions $[s_{j+1} \times s_j + 1]$
  - Because
    - $s_{j+1}$ is equal to the number of units in layer $(j + 1)$
    - is equal to the number of units in layer j, plus an additional unit
- Looking at the ϴ matrix
  Column length is the number of units in the following layer
- Row length is the number of units in the current layer + 1 (because we have to map the bias unit)
  - So, if we had two layers - 110 and 15 units in each
    - Then $\Theta^j$ would be = [15 x 111]

What are the computations ?

- We have to calculate the activation for each node
- That activation depends on
  - The input(s) to the node
  - The parameter associated with that node (from the ϴ vector associated with that layer)

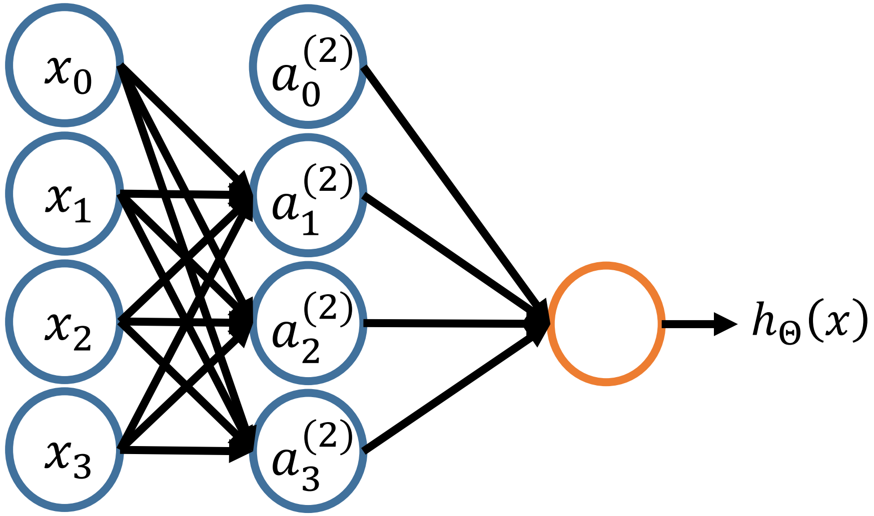



$$g(\Theta_{10}{}^1 x_0 + \Theta_{11}{}^1 x_1 + \Theta_{12}{}^1 x_2 + \Theta_{13}{}^1 x_3)$$

- For example $\Theta_{13}{}^{1}$ = means
  - **1** - we're mapping to node 1 in layer l+1
  - **3** - we're mapping from node 3 in layer l
  - **1** - we're mapping from layer 1

# Neural network

"Pre-activation"




$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad \mathrm{z}^{(2)} = \begin{bmatrix} \mathrm{z}_1^{(2)} \\ \mathrm{z}_2^{(2)} \\ \mathrm{z}_3^{(2)} \end{bmatrix}$$

$$a_1^{(2)} = g\left(\Theta_{10}^{(1)} x_0 + \Theta_{11}^{(1)} x_1 + \Theta_{12}^{(1)} x_2 + \Theta_{13}^{(1)} x_3\right) = g(\mathrm{z}_1^{(2)})$$

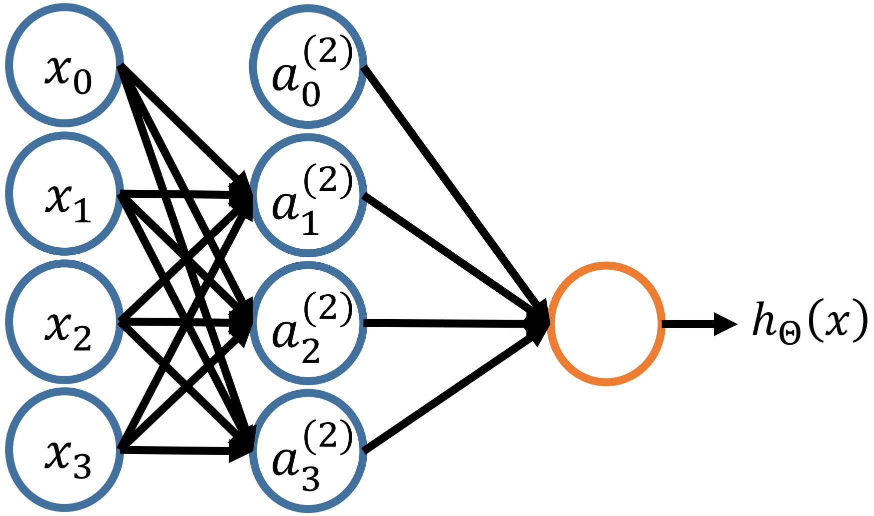
$$a_2^{(2)} = g\left(\Theta_{20}^{(1)} x_0 + \Theta_{21}^{(1)} x_1 + \Theta_{22}^{(1)} x_2 + \Theta_{23}^{(1)} x_3\right) = g(\mathrm{z}_2^{(2)})$$

$$a_3^{(2)} = g\left(\Theta_{30}^{(1)} x_0 + \Theta_{31}^{(1)} x_1 + \Theta_{32}^{(1)} x_2 + \Theta_{33}^{(1)} x_3\right) = g(\mathrm{z}_3^{(2)})$$

$$h_\Theta(x) = g\left(\Theta_{10}^{(2)} a_0^{(2)} + \Theta_{11}^{(1)} a_1^{(2)} + \Theta_{12}^{(1)} a_2^{(2)} + \Theta_{13}^{(1)} a_3^{(2)}\right) = g(z^{(3)})$$

Slide credit: Andrew Ng

# Neural network




"Pre-activation"

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

$$a_1^{(2)} = g(z_1^{(2)})$$
$$a_2^{(2)} = g(z_2^{(2)})$$
$$a_3^{(2)} = g(z_3^{(2)})$$
$$h_\Theta(x) = g(z^{(3)})$$

$$z^{(2)} = \Theta^{(1)}x = \Theta^{(1)}a^{(1)}$$
$$a^{(2)} = g(z^{(2)})$$
$$\text{Add } a_0^{(2)} = 1$$
$$z^{(3)} = \Theta^{(2)}a^{(2)}$$
$$h_\Theta(x) = a^{(3)} = g(z^{(3)})$$

Slide credit: Andrew Ng

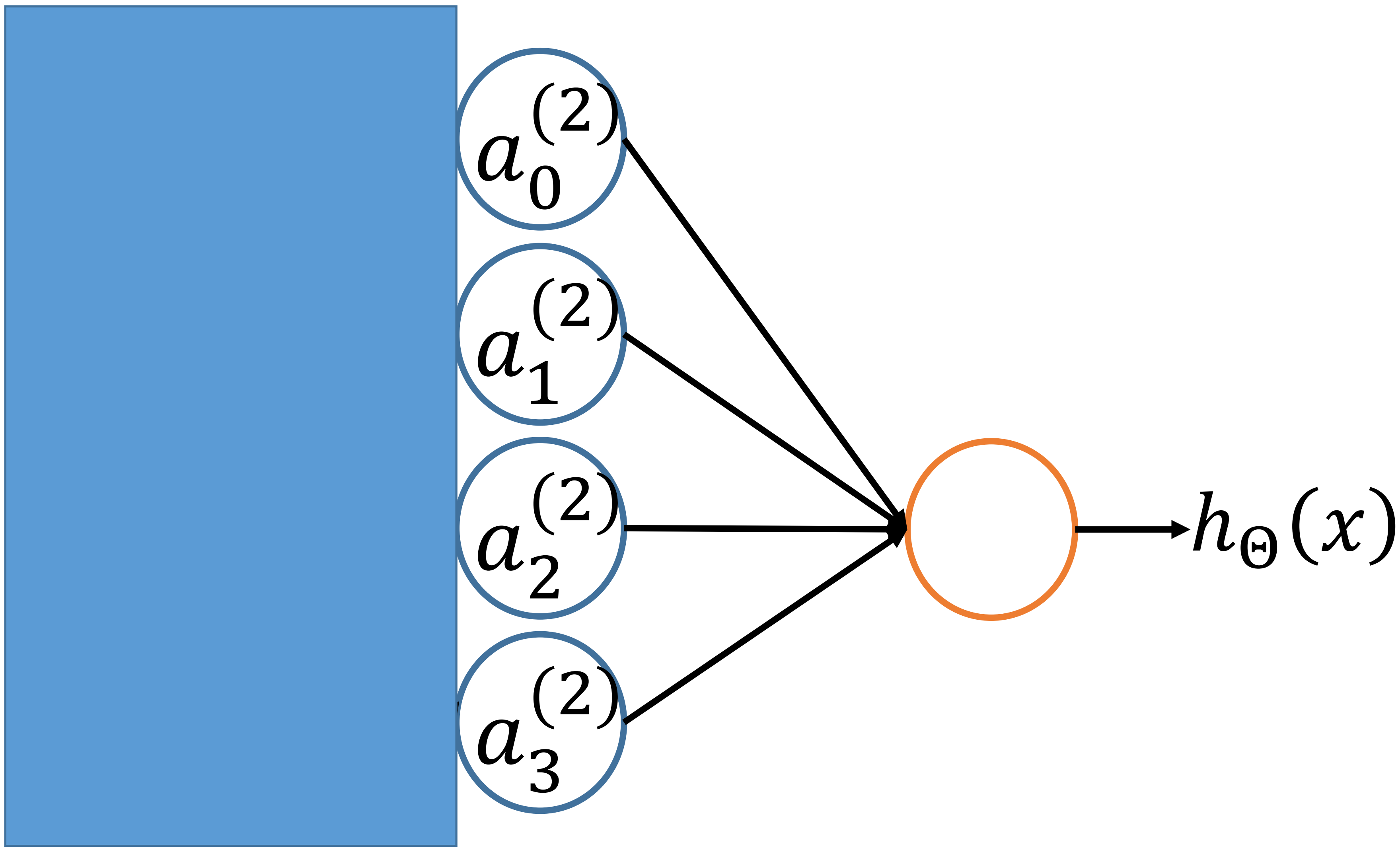
# Forward Propagation

- This process is also called **forward propagation**
  - Start off with activations of input unit
    - i.e. the x vector as input
  - Forward propagate and calculate the activation of each layer sequentially
  - This is a vectorised version of this implementation

$$x = \begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \end{bmatrix} \quad z^{(2)} = \begin{bmatrix} z_1^{(2)} \\ z_2^{(2)} \\ z_3^{(2)} \end{bmatrix}$$

# Neural network learning its own features

Diagram below looks a lot like logistic regression

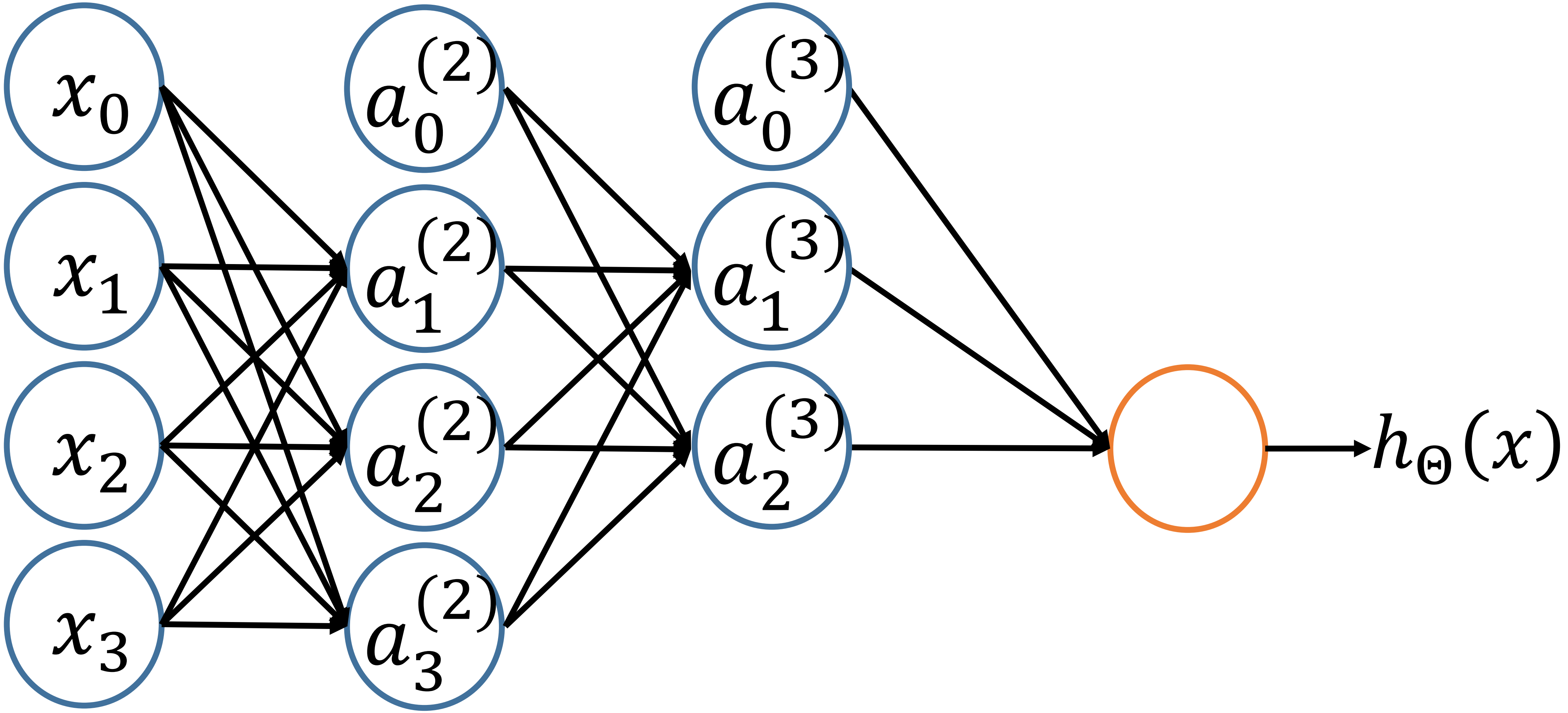



Slide credit: Andrew Ng

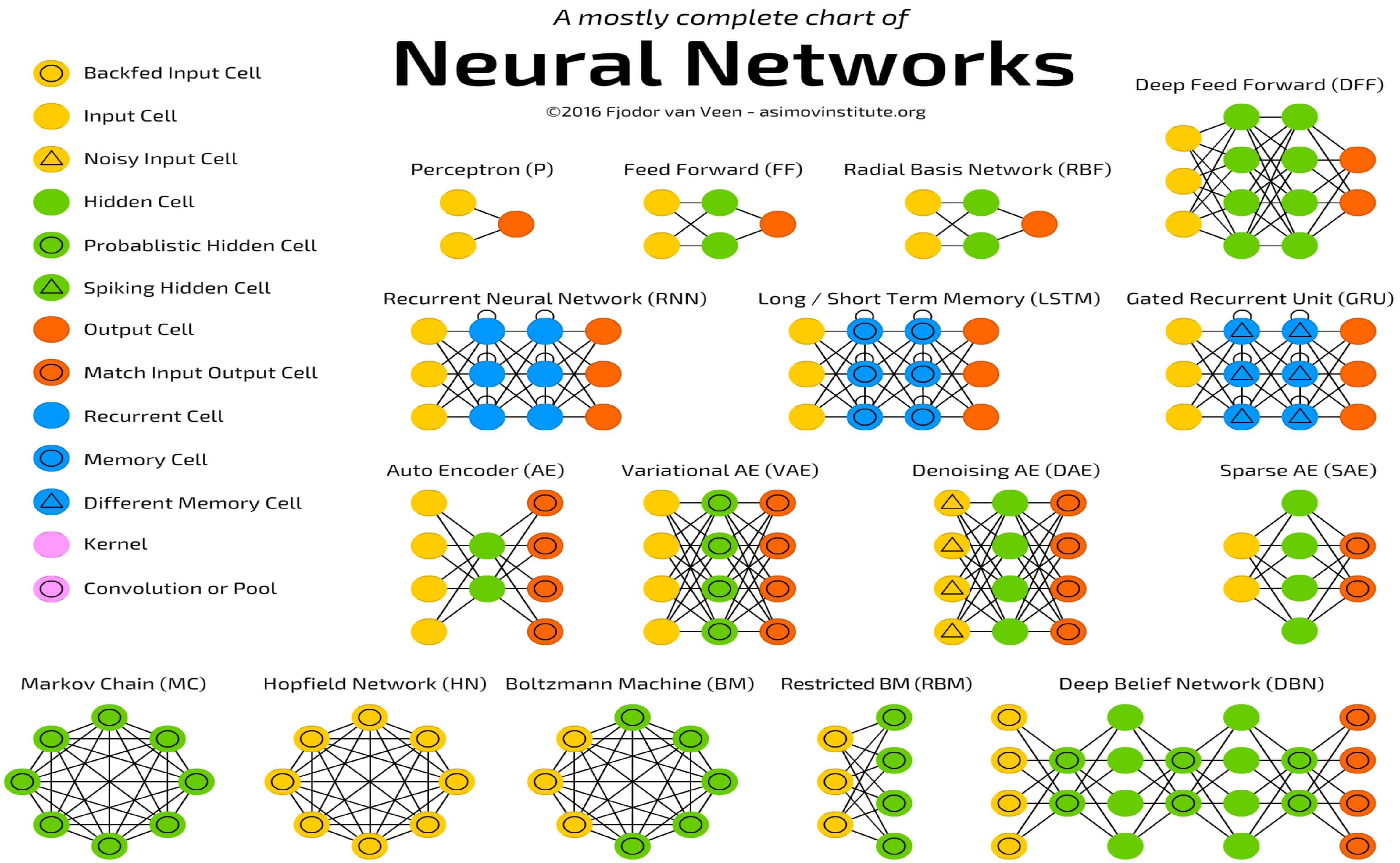
# Neural network learning its own features

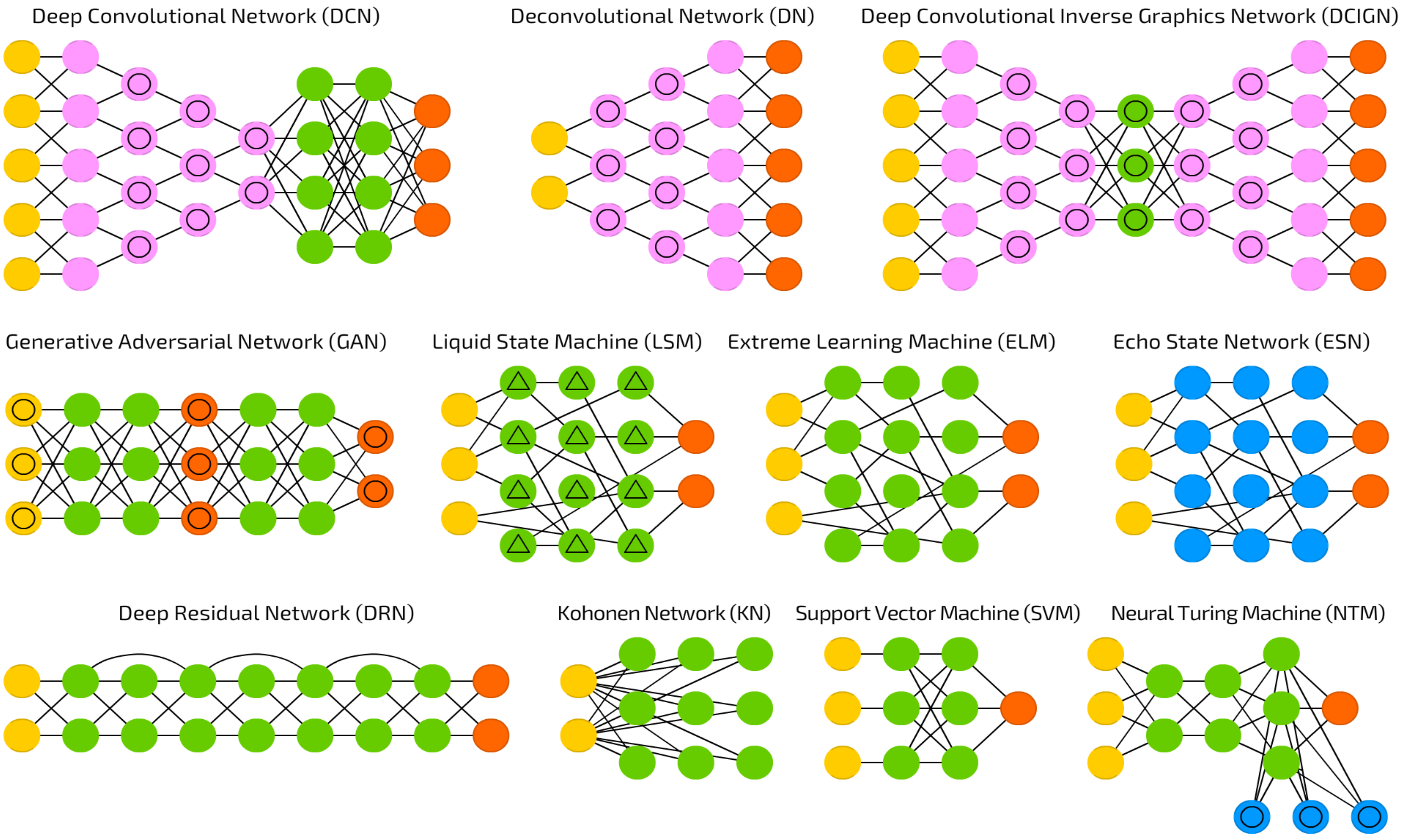
- Layer 3 is a logistic regression node
  - The hypothesis output = $g(\Theta_{10}^{2} a_0^{2} + \Theta_{11}^{2} a_1^{2} + \Theta_{12}^{2} a_2^{2} + \Theta_{13}^{2} a_3^{2})$
  - This is just logistic regression
    - The only difference is, instead of input a feature vector, the features are just values calculated by the hidden layer
- The features $a_1^2$, $a_2^2$, and $a_3^2$ are calculated/learned - not original features
- So the mapping from layer 1 to layer 2 (i.e. the calculations which generate the $a^2$ features) is determined by another set of parameters - $\Theta^1$
  - So instead of being constrained by the original input features, a neural network can learn its own features to feed into logistic regression
  - Depending on the $\Theta^1$ parameters you can learn some interesting things
    - Flexibility to learn whatever features it wants to feed into the final logistic regression calculation
      - So, if we compare this to previous logistic regression, you would have to calculate your own exciting features to define the best way to classify or describe something

# Other network architectures




Slide credit: Andrew Ng


A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)
Markov Chain (MC)
Hopfield Network (HN)
Boltzmann Machine (BM)
Restricted BM (RBM)
Deep Belief Network (DBN)



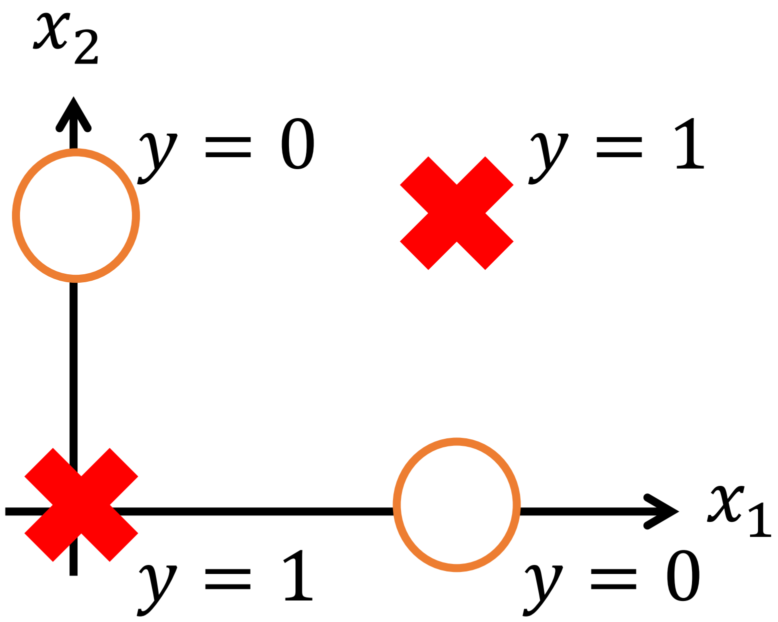
Deep Convolutional Network (DCN)
Deconvolutional Network (DN)
Deep Convolutional Inverse Graphics Network (DCIGN)
Generative Adversarial Network (GAN)
Liquid State Machine (LSM)
Extreme Learning Machine (ELM)
Echo State Network (ESN)
Deep Residual Network (DRN)
Kohonen Network (KN)
Support Vector Machine (SVM)
Neural Turing Machine (NTM)

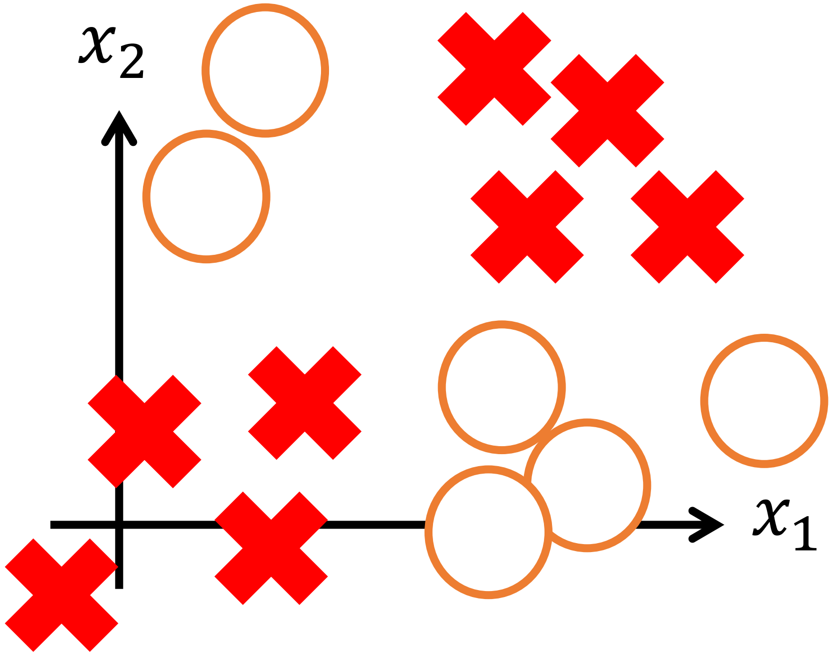
# Neural Networks

- Why neural networks?
- Model representation
- **Examples and intuitions**
- Multi-class classification

# A complex Non-linear classification example using Neural Networks: XOR/XNOR

- $x_1, x_2$ are binary (0 or 1)
- $y = XOR(x_1, x_2)$

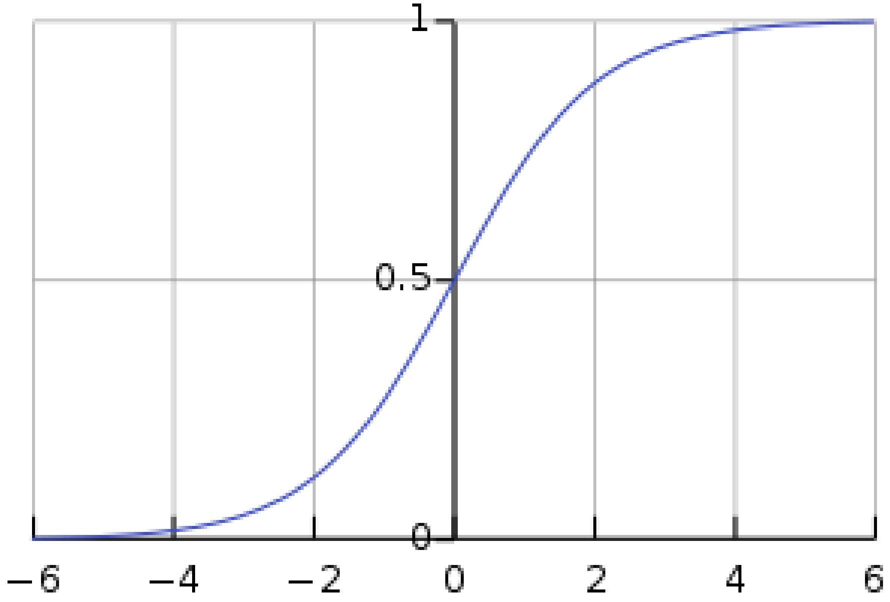


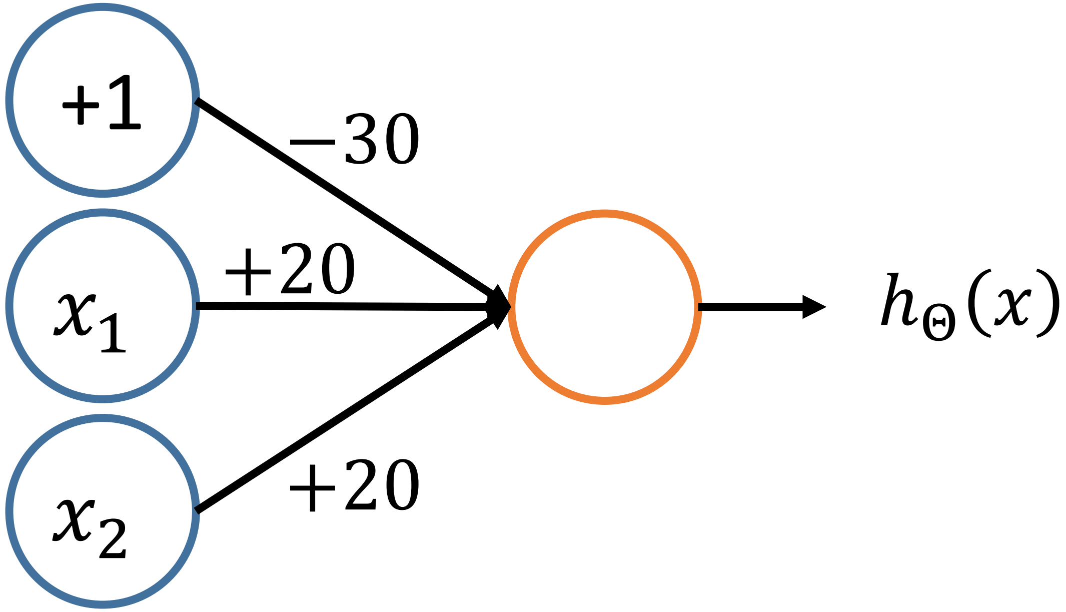




Slide credit: Andrew Ng

# Simple example: AND

- $x_1, x_2 \in \{0, 1\}$
- $y = x_1 \text{ AND } x_2$




| $x_1$ | $x_2$ | $h_\Theta(x)$ |
|---|---|---|
| 0 | 0 | $g(-30) \approx 0$ |
| 0 | 1 | $g(-10) \approx 0$ |
| 1 | 0 | $g(-10) \approx 0$ |
| 1 | 1 | $g(10) \approx 1$ |

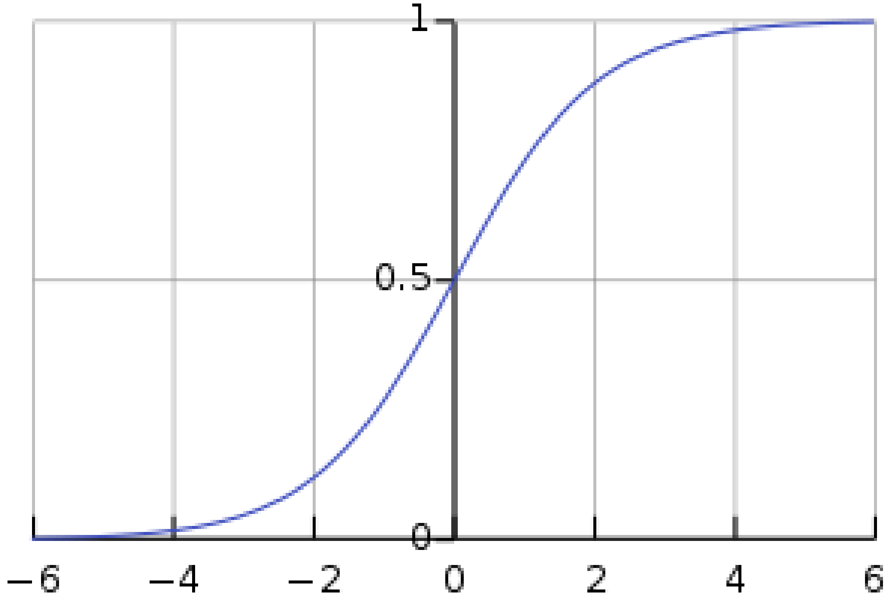
$$h_\Theta(x) = g(-30 + 20x_1 + 20x_2)$$

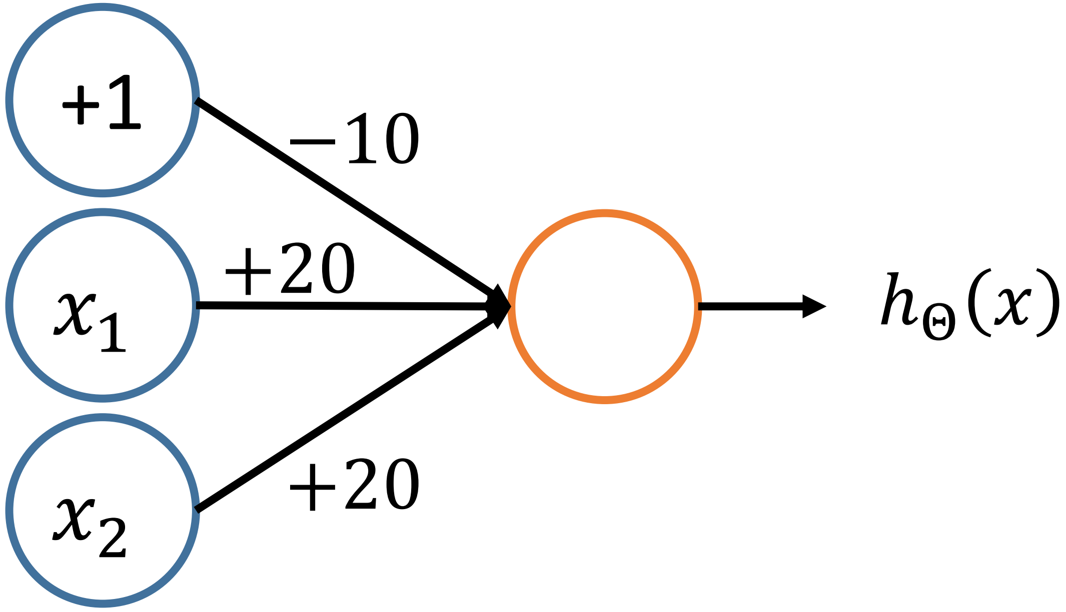
$$h_\Theta(x) \approx x_1 \text{ AND } \mathrm{x}_2$$

Slide credit: Andrew Ng

# Simple example: OR

- $x_1, x_2 \in \{0, 1\}$
- $y = x_1 \text{ AND } x_2$




| $x_1$ | $x_2$ | $h_\Theta(x)$ |
|---|---|---|
| 0 | 0 | $g(-10) \approx 0$ |
| 0 | 1 | $g(10) \approx 1$ |
| 1 | 0 | $g(10) \approx 1$ |
| 1 | 1 | $g(30) \approx 1$ |

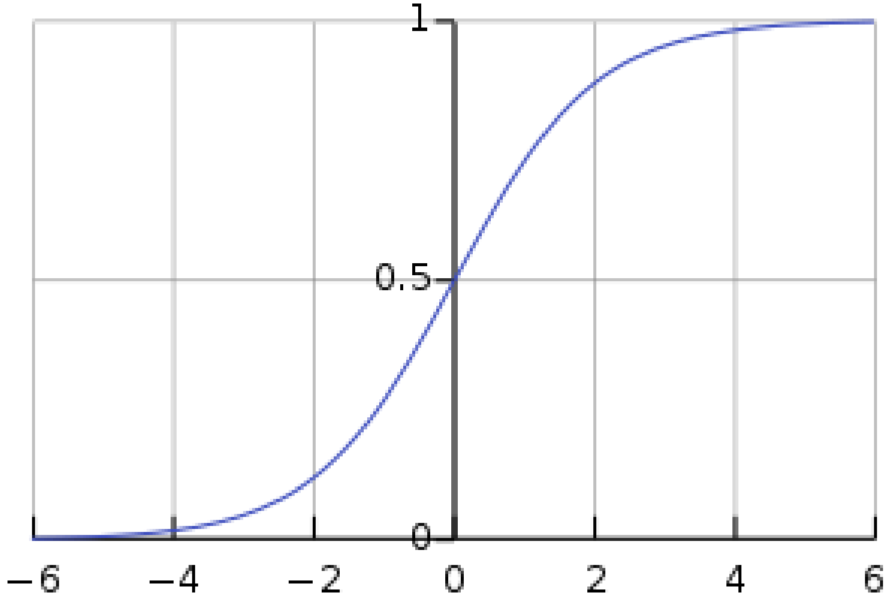
$$h_\Theta(x) = g(-10 + 20x_1 + 20x_2)$$

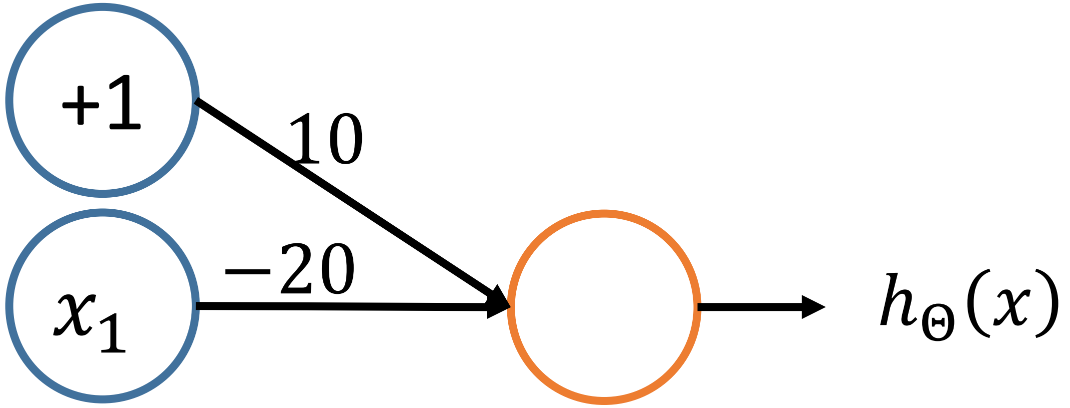
$$h_\Theta(x) \approx x_1 \text{ OR } x_2$$

Slide credit: Andrew Ng

# Simple example: NOT

- $x_1, x_2 \in \{0, 1\}$
- $y = x_1 \text{ AND } x_2$



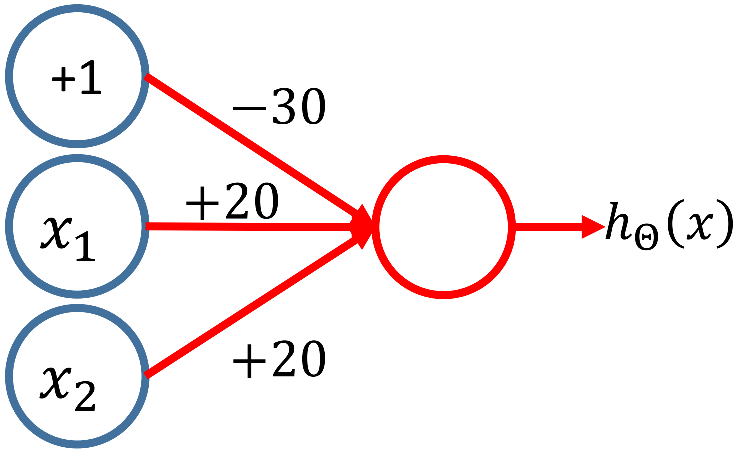

| $x_1$ | $h_\Theta(x)$ |
|---|---|
| 0 | $g(10) \approx 1$ |
| 1 | $g(-10) \approx 0$ |

$$h_\Theta(x) = g(10 - 20x_1)$$

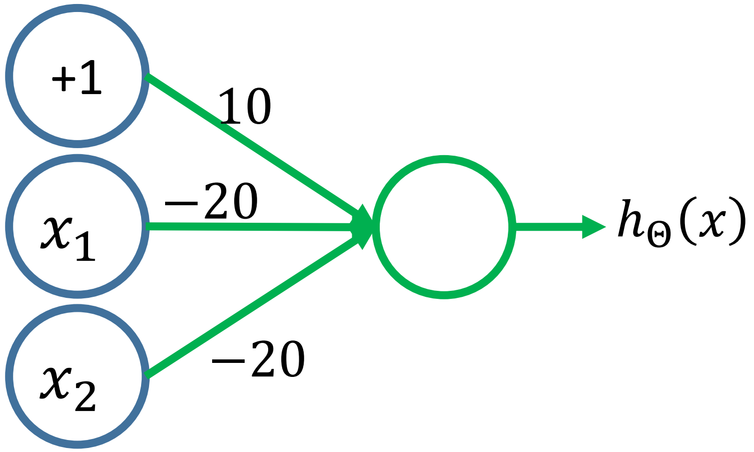
$$h_\Theta(x) \approx \text{NOT } x_1$$

Slide credit: Andrew Ng

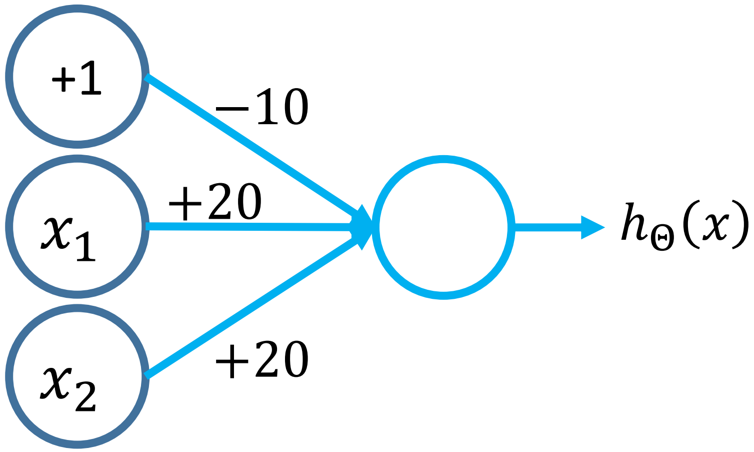
# Putting it together: $x_1$ XNOR $x_2$



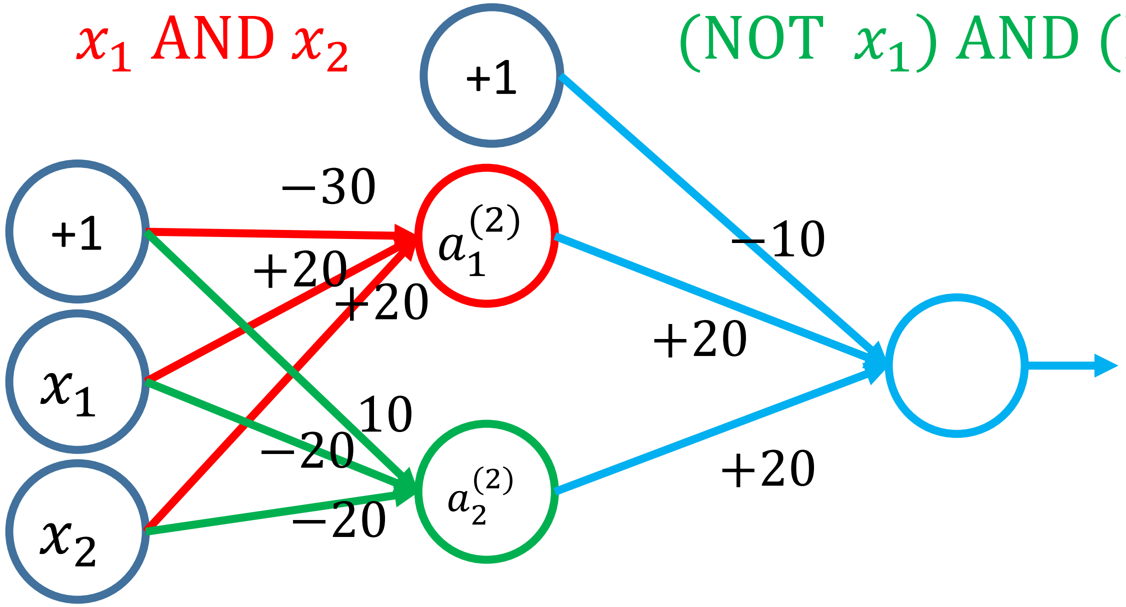

$x_1$ AND $x_2$




(NOT $x_1$) AND (NOT $x_2$)




$x_1$ OR $x_2$



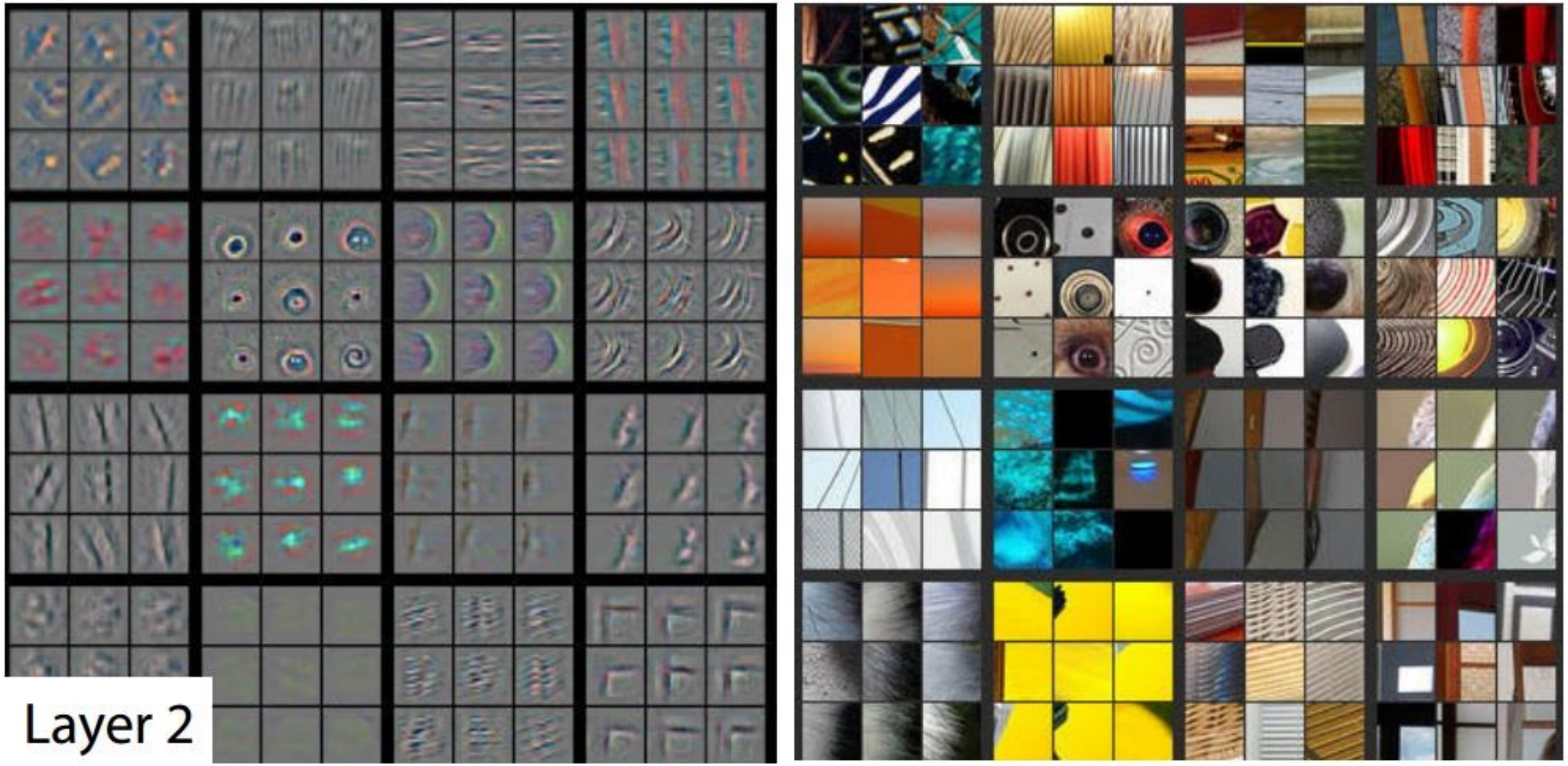

| $x_1$ | $x_2$ | $a_1^{(2)}$ | $a_2^{(2)}$ | $h_\Theta(x)$ |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 |
| 1 | 1 | 1 | 0 | 1 |

Slide credit: Andrew Ng

# Layer 1

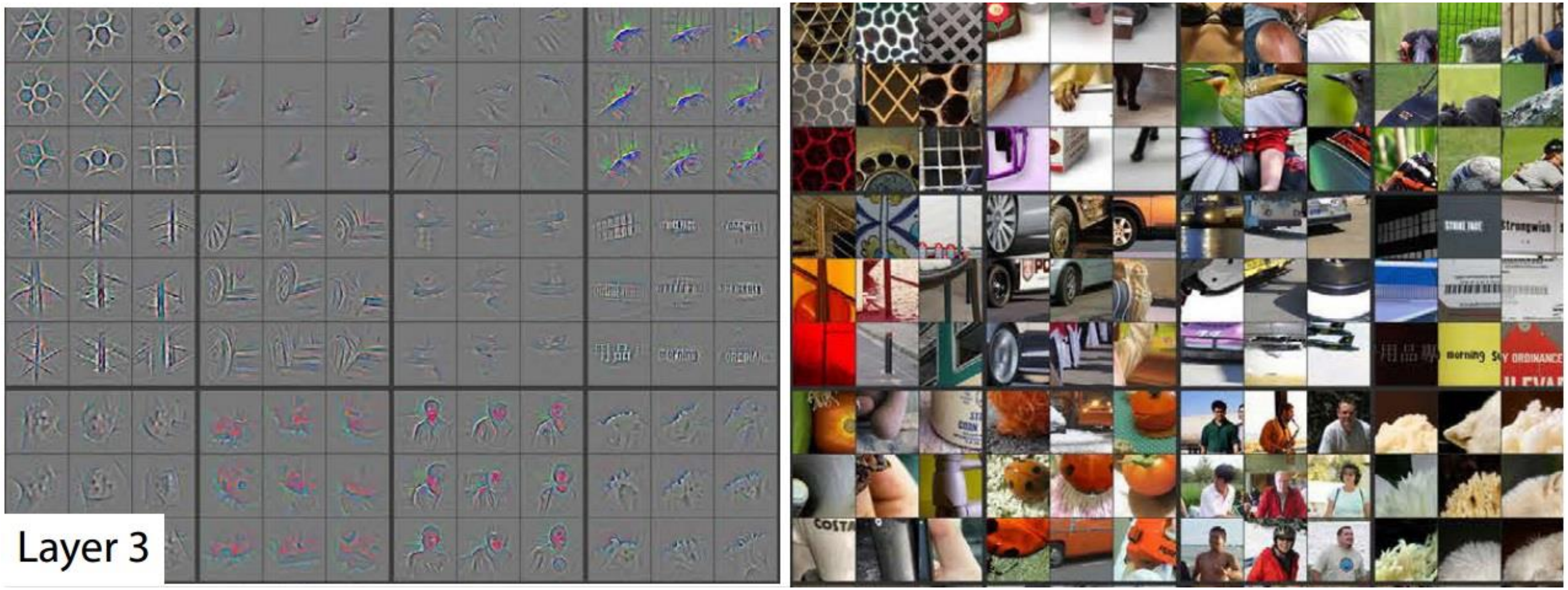

Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

# Layer 2



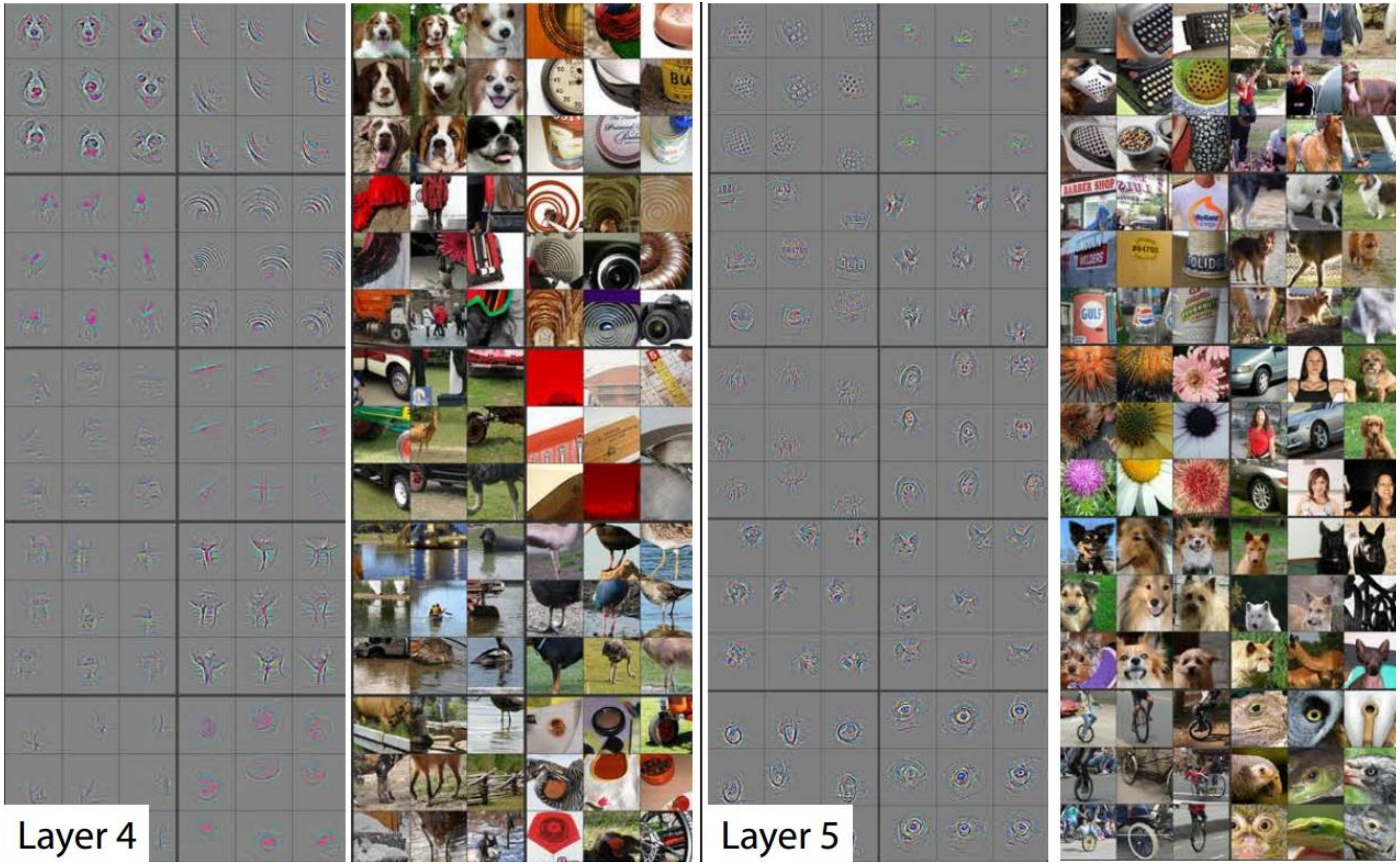

Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

# Layer 3




Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

# Layer 4 and 5




Visualizing and Understanding Convolutional Networks [Zeiler and Fergus, ECCV 2014]

# Neural Networks

- Why neural networks?
- Model representation
- Examples and intuitions
- **Multi-class classification**

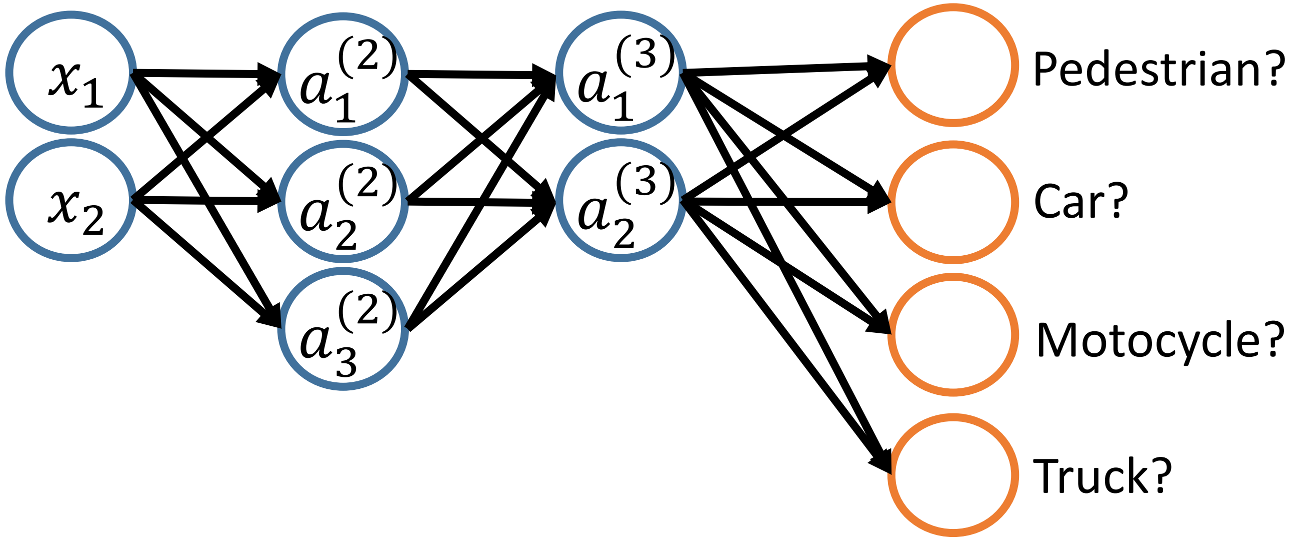
# Multiple output units: One-vs-all

Pedestrian

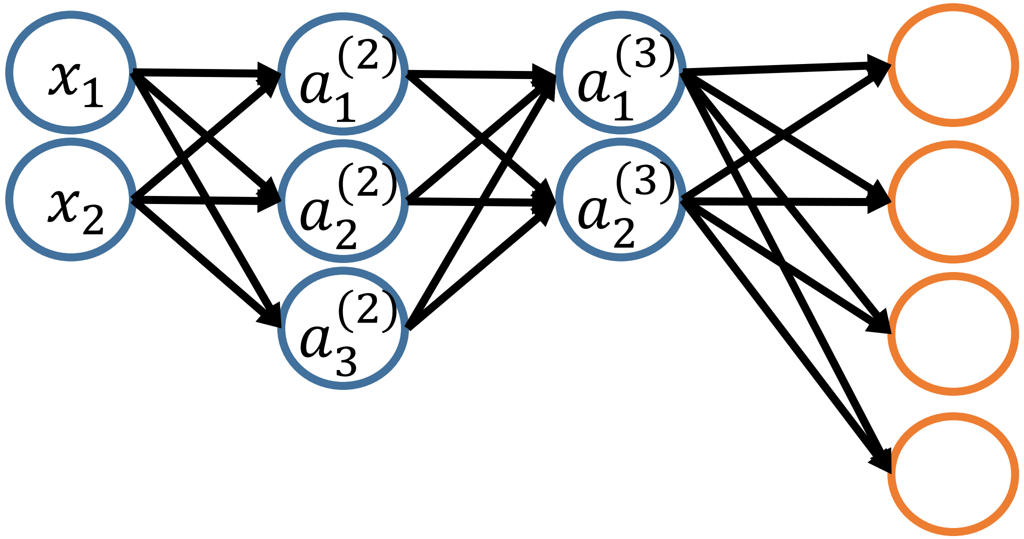
Car

Motorcycle

Truck



# Multiple output units: One-vs-all




$h_\Theta(x) \in R^4$

Training set : $\left(x^{(1)}, y^{(1)}\right), \left(x^{(2)}, y^{(2)}\right), \cdots \left(x^{(m)}, y^{(m)}\right),$

$y^{(i)}$ one of $\begin{bmatrix}1\\0\\0\\0\end{bmatrix}, \begin{bmatrix}0\\1\\0\\0\end{bmatrix}, \begin{bmatrix}0\\0\\1\\0\end{bmatrix}, \begin{bmatrix}0\\0\\0\\1\end{bmatrix}$

# Things to remember

- Why neural networks?
- Model representation
- Examples and intuitions
- Multi-class classification