

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>

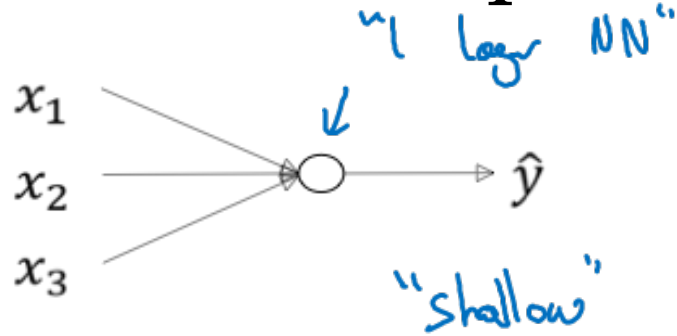


deeplearning.ai

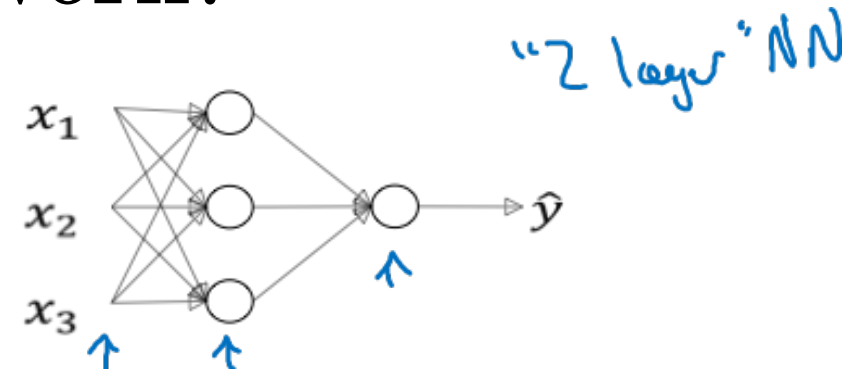
Deep Neural Networks

Deep L-layer
Neural network

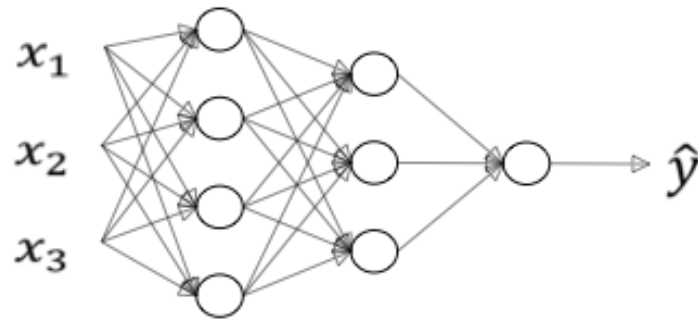
What is a deep neural network?



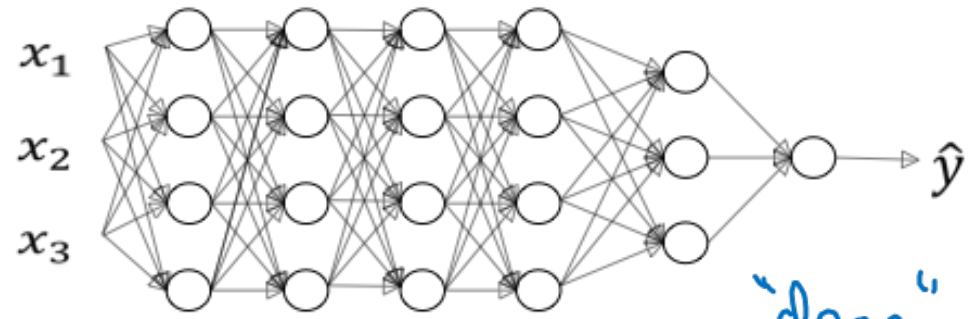
logistic regression



1 hidden layer



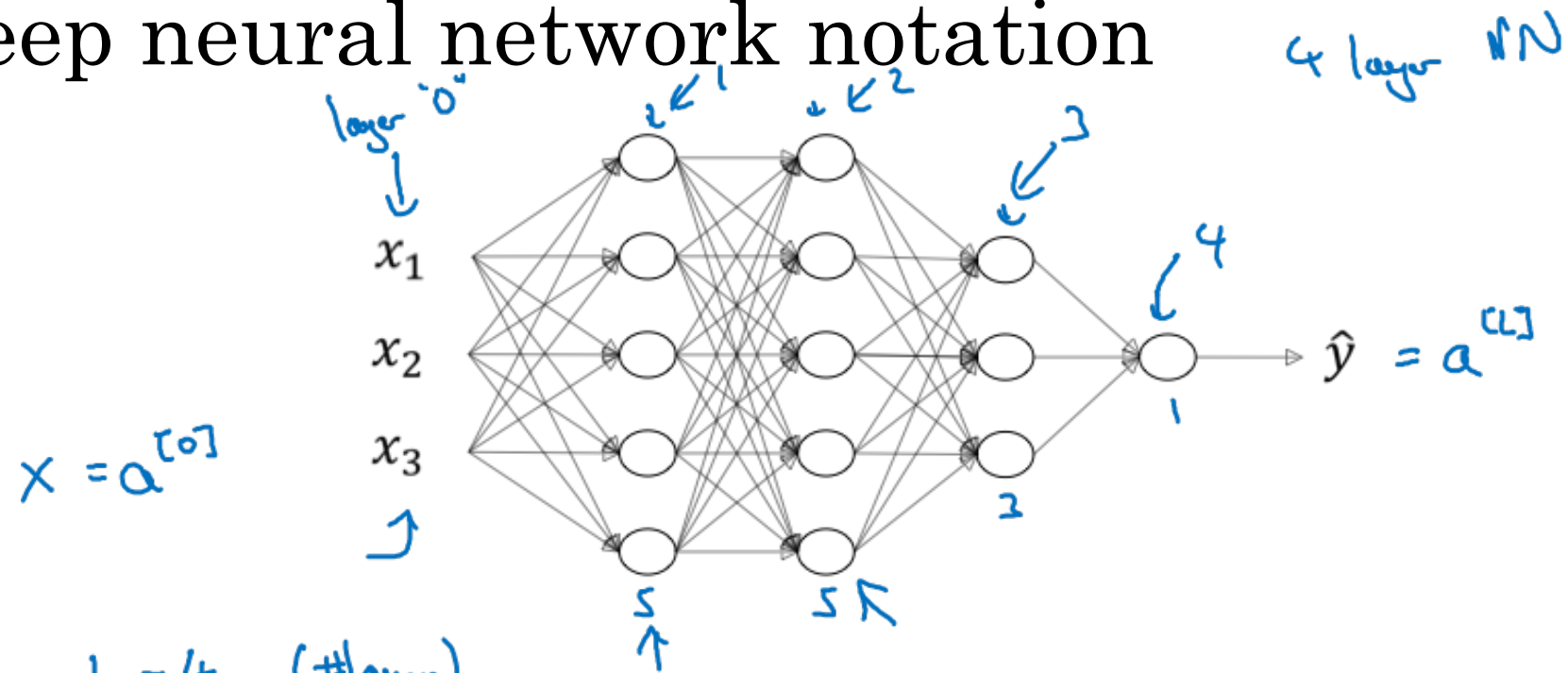
2 hidden layers



5 hidden layers

"deep"

Deep neural network notation



$L = 4$ (#layers)

$n^{[l]} = \# \text{units in layer } l$

$a^{[l]} = \text{activations in layer } l$

$a^{[l]} = g(z^{[l]})$, $w_{\delta a}^{[l]} = \text{weights for } \underline{z^{[l]}}$

$n^{[1]} = 5, n^{[2]} = 5, n^{[3]} = 3, n^{[4]} = n^{[L]} = 1$

$n^{[0]} = n_x = 3$

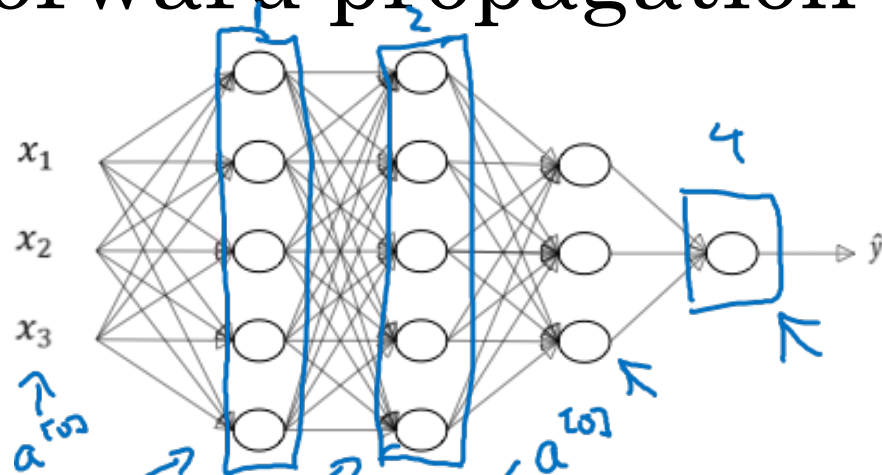


deeplearning.ai

Deep Neural Networks

Forward Propagation in a Deep Network

Forward propagation in a deep network



$$z^{[l]} = W^{[l]} A^{[l-1]} + b^{[l]}$$

$$A^{[l]} = g^{[l]}(z^{[l]})$$

$A^{[0]} = X$

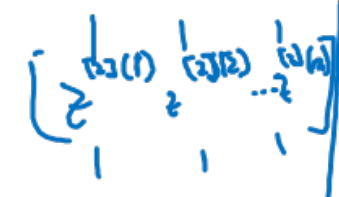
$$X : z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$

$$a^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]} a^{[1]} + b^{[2]}$$

$$a^{[2]} = g^{[2]}(z^{[2]})$$

$$z^{[4]} = W^{[4]} a^{[3]} + b^{[4]}, \quad a^{[4]} = g^{[4]}(z^{[4]})$$



Vertical:

$$z^{[1]} = W^{[1]} A^{[0]} + b^{[1]}$$

$$A^{[1]} = g^{[1]}(z^{[1]})$$

$$z^{[2]} = W^{[2]} A^{[1]} + b^{[2]}$$

$$A^{[2]} = g^{[2]}(z^{[2]})$$

$$\hat{Y} = g^{[4]}(z^{[4]}) = A^{[4]}$$

$\rightarrow X = A^{[0]}$

for $l=1 \dots 4$



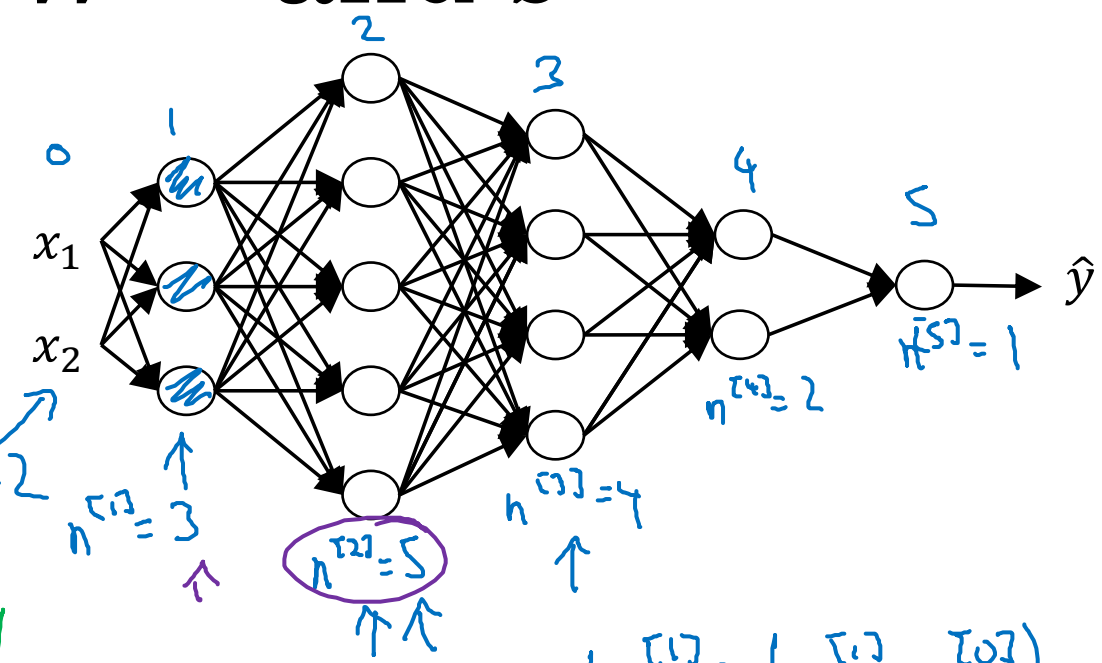
deeplearning.ai

Deep Neural Networks

Getting your matrix
dimensions right

Parameters $W^{[l]}$ and $b^{[l]}$

\downarrow
 $z^{[L]} = g^{[L]}(a^{[L]})$
 \uparrow
 \downarrow
 $a^{[L]}$



$L=5$

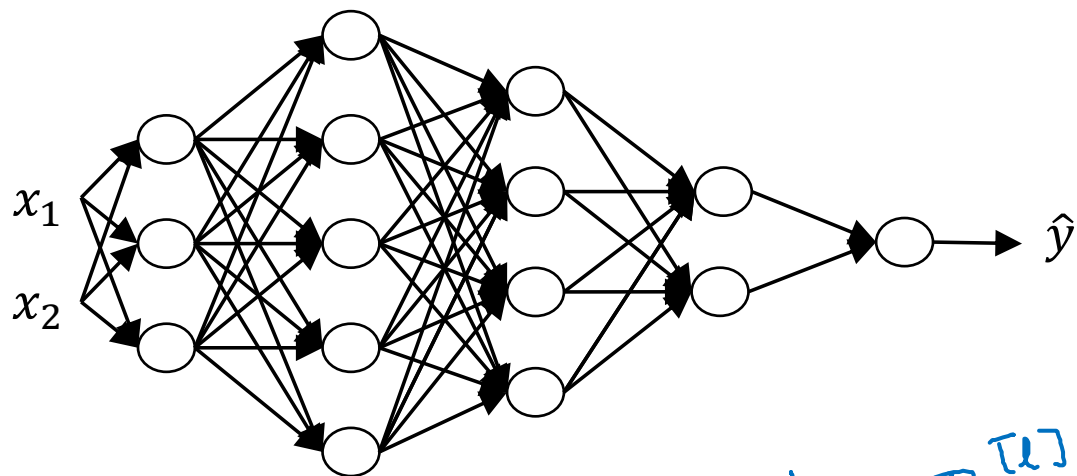
$\rightarrow W^{[L]}: (n^{[L]}, n^{[L-1]})$
 $\rightarrow b^{[L]}: (n^{[L]}, 1)$
 $\rightarrow \Delta W^{[L]}: (n^{[L]}, n^{[L-1]})$
 $\rightarrow \Delta b^{[L]}: (n^{[L]}, 1)$

\downarrow
 $z^{[1]} = \boxed{W^{[1]} \cdot x} + \boxed{b^{[1]}}$
 $(3,1) \leftarrow (3,2) \quad (2,1)$
 $(n^{[1]}, 1) \quad (n^{[1]}, n^{[0]}) \quad (n^{[0]}, 1)$
 $(3,1) \quad (n^{[1]}, 1)$

$W^{[1]}: (n^{[1]}, n^{[0]})$
 $W^{[2]}: (5, 3) \quad (n^{[2]}, n^{[1]})$
 $z^{[2]} = \boxed{W^{[2]} \cdot a^{[1]}} + \boxed{b^{[2]}}$
 $\uparrow \quad \uparrow \quad \uparrow$
 $\rightarrow (5,1) \quad (5,3) \quad (3,1)$
 $(5,1) \quad (n^{[2]}, 1)$
 $W^{[3]}: (4, 5)$
 $W^{[4]}: (2, 4)$, $W^{[5]}: (1, 2)$

$\begin{bmatrix} \cdot \\ \cdot \\ \cdot \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ \cdot & \cdot \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \cdot \\ \cdot \end{bmatrix}$

Vectorized implementation



$$z^{[l]} = W^{[l]} \cdot x + b^{[l]}$$

$(n^{[l]}, 1)$ $(n^{[l]}, n^{[l-1]})$ $(n^{[l-1]}, 1)$ $(n^{[l-1]}, 1)$

$$\begin{bmatrix} z^{[1]} \\ z^{[2]} \\ \dots \\ z^{[L]} \end{bmatrix}$$

$$Z^{[l]} = W^{[l]} \cdot X + b^{[l]}$$

$(n^{[l]}, m)$ $(n^{[l]}, n^{[l-1]})$ $(n^{[l-1]}, m)$ $(n^{[l-1]}, 1)$

$(n^{[l]}, m)$ $(n^{[l]}, n^{[l-1]})$ $(n^{[l-1]}, m)$ $(n^{[l-1]}, 1)$

$$z^{[l]}, a^{[l]} : (n^{[l]}, 1)$$

$$z^{[l]}, A^{[l]} : (n^{[l]}, m)$$

$l=0 \quad A^{[0]} = X = (n^{[0]}, m)$

$$dz^{[l]}, dA^{[l]} : (n^{[l]}, m)$$

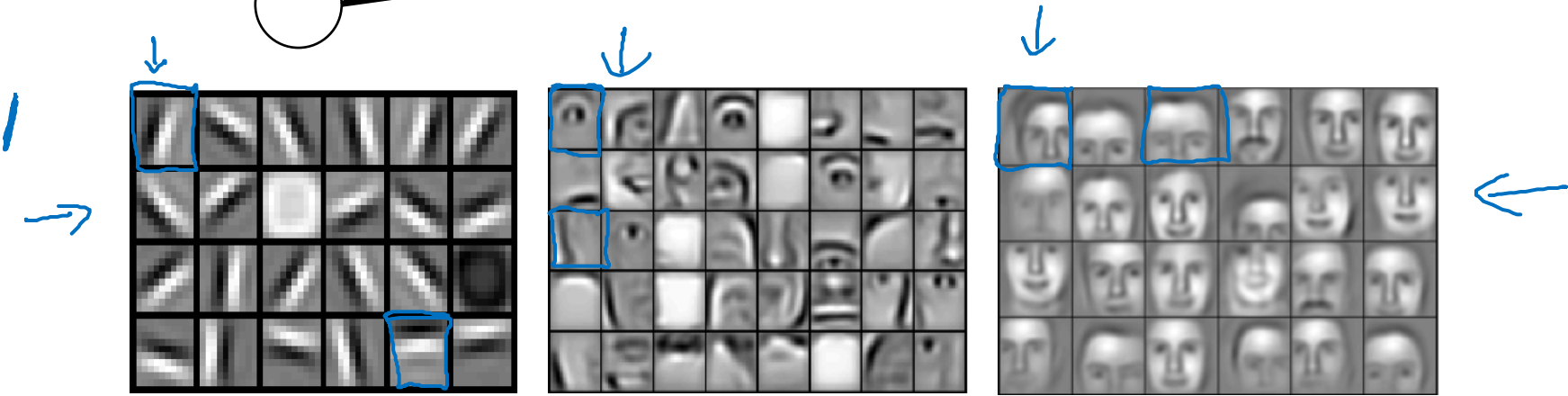
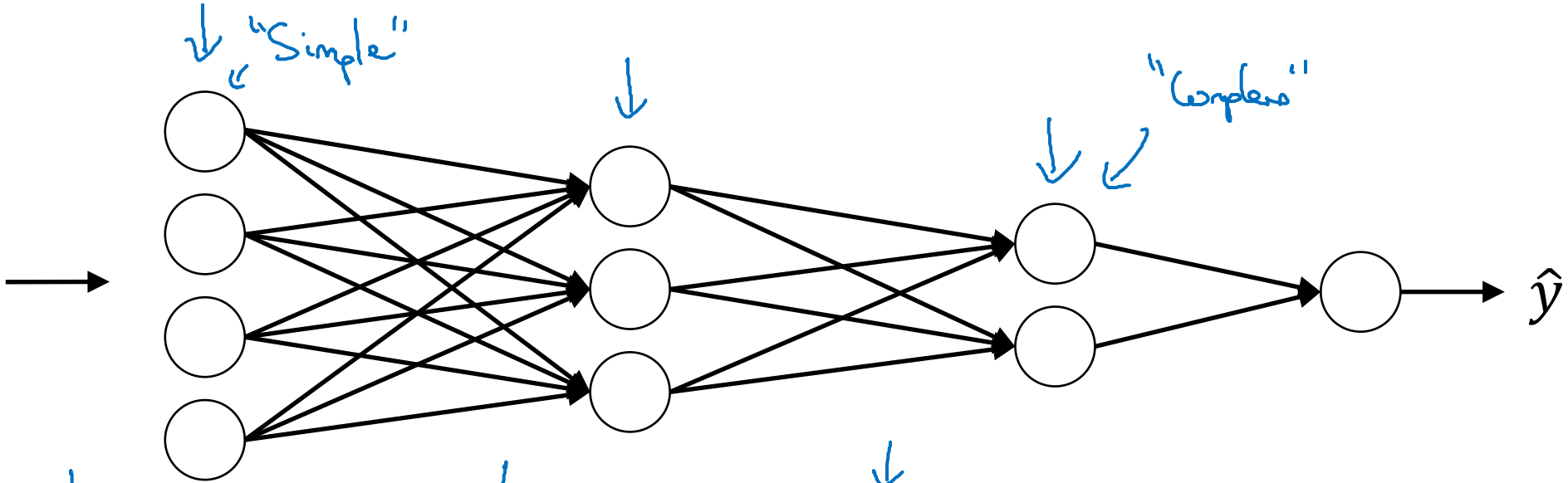
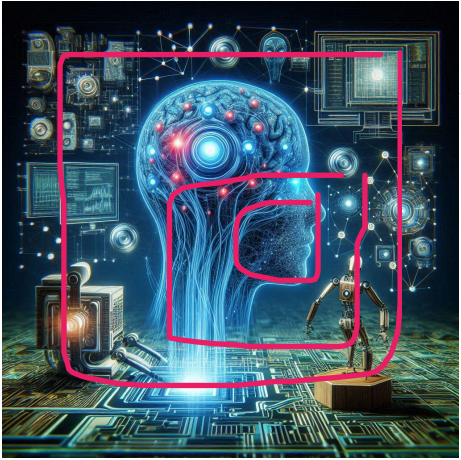


deeplearning.ai

Deep Neural Networks

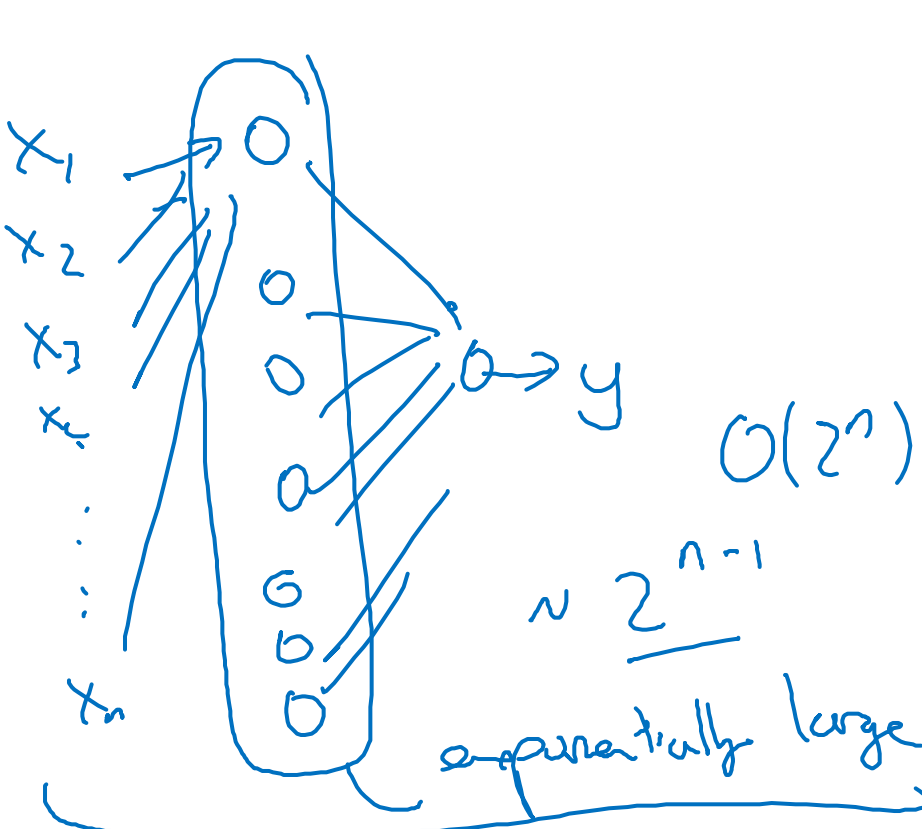
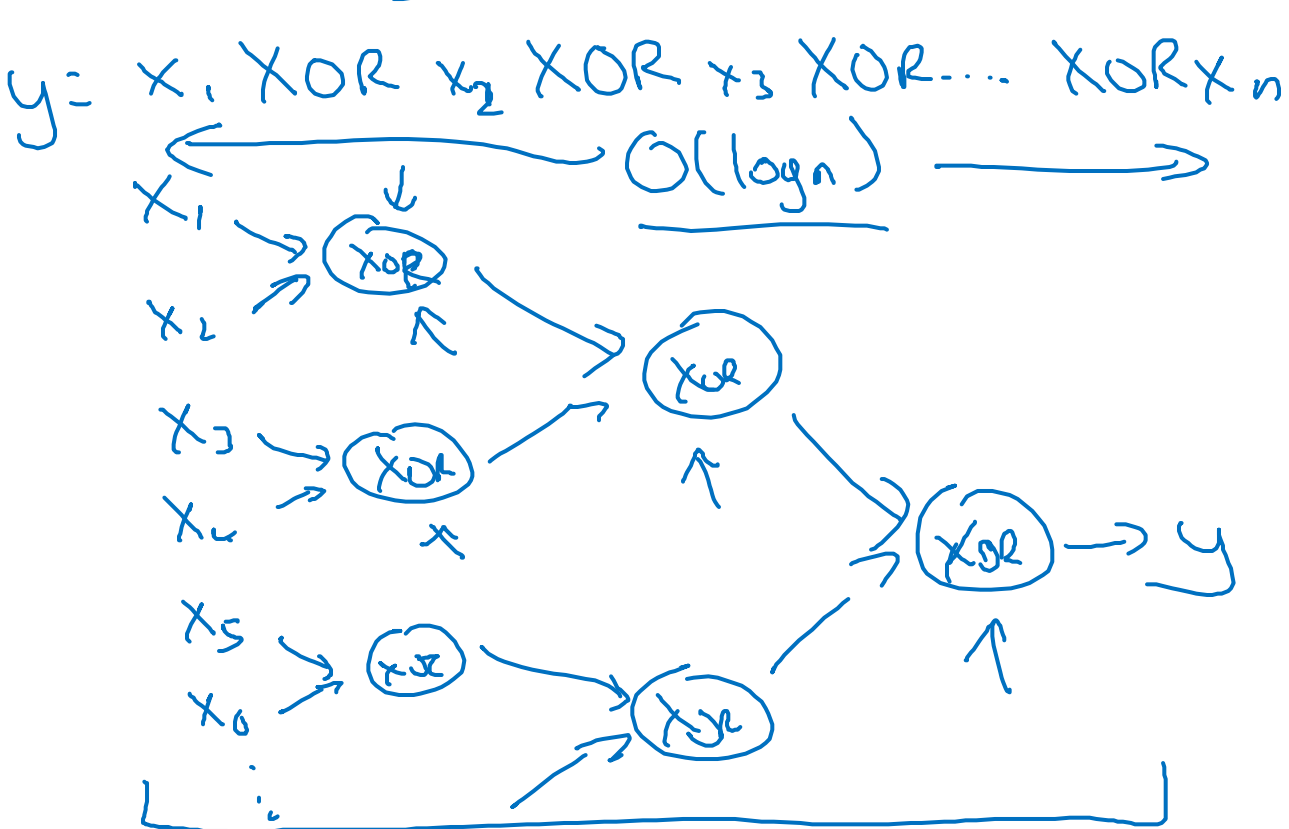
Why deep
representations?

Intuition about deep representation



Circuit theory and deep learning

Informally: There are functions you can compute with a “small” L-layer deep neural network that shallower networks require exponentially more hidden units to compute.



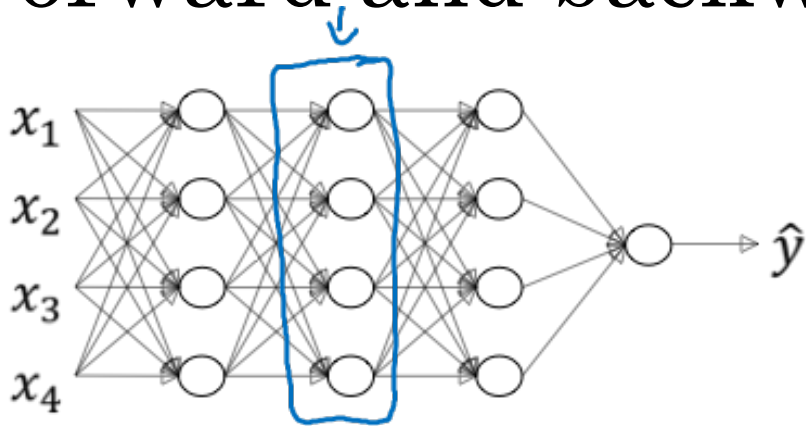


deeplearning.ai

Deep Neural Networks

Building blocks of
deep neural networks

Forward and backward functions



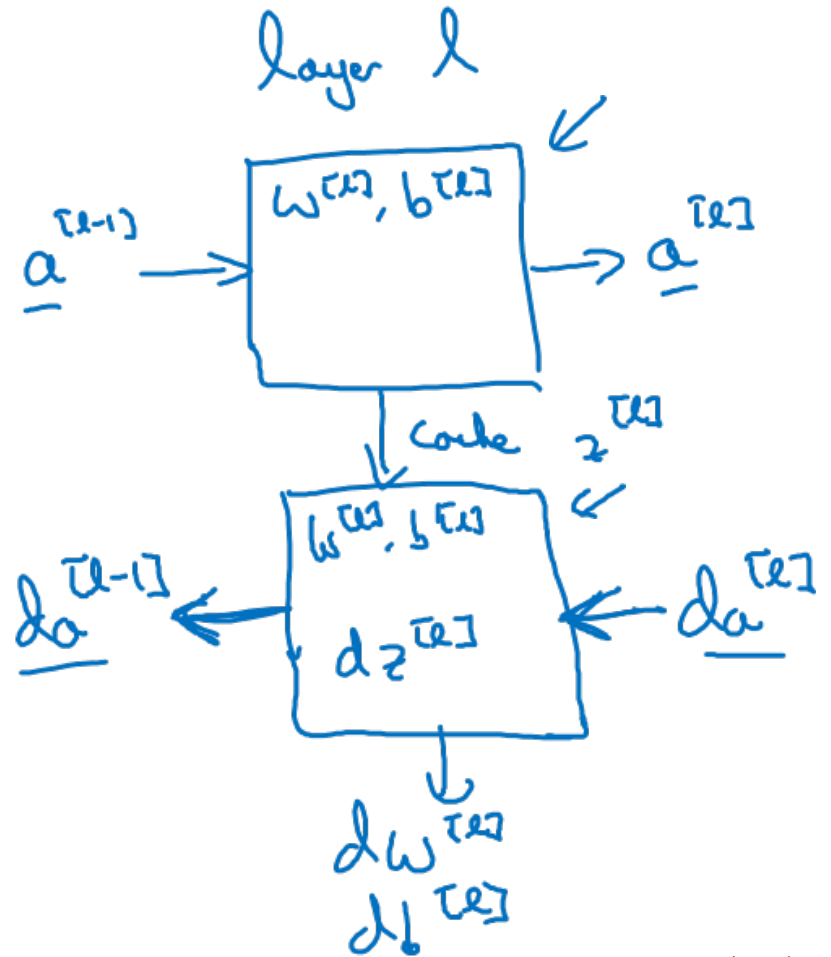
layer l : $W^{[l]}, b^{[l]}$

→ Forward: Input $a^{[l-1]}$, output $a^{[l]}$

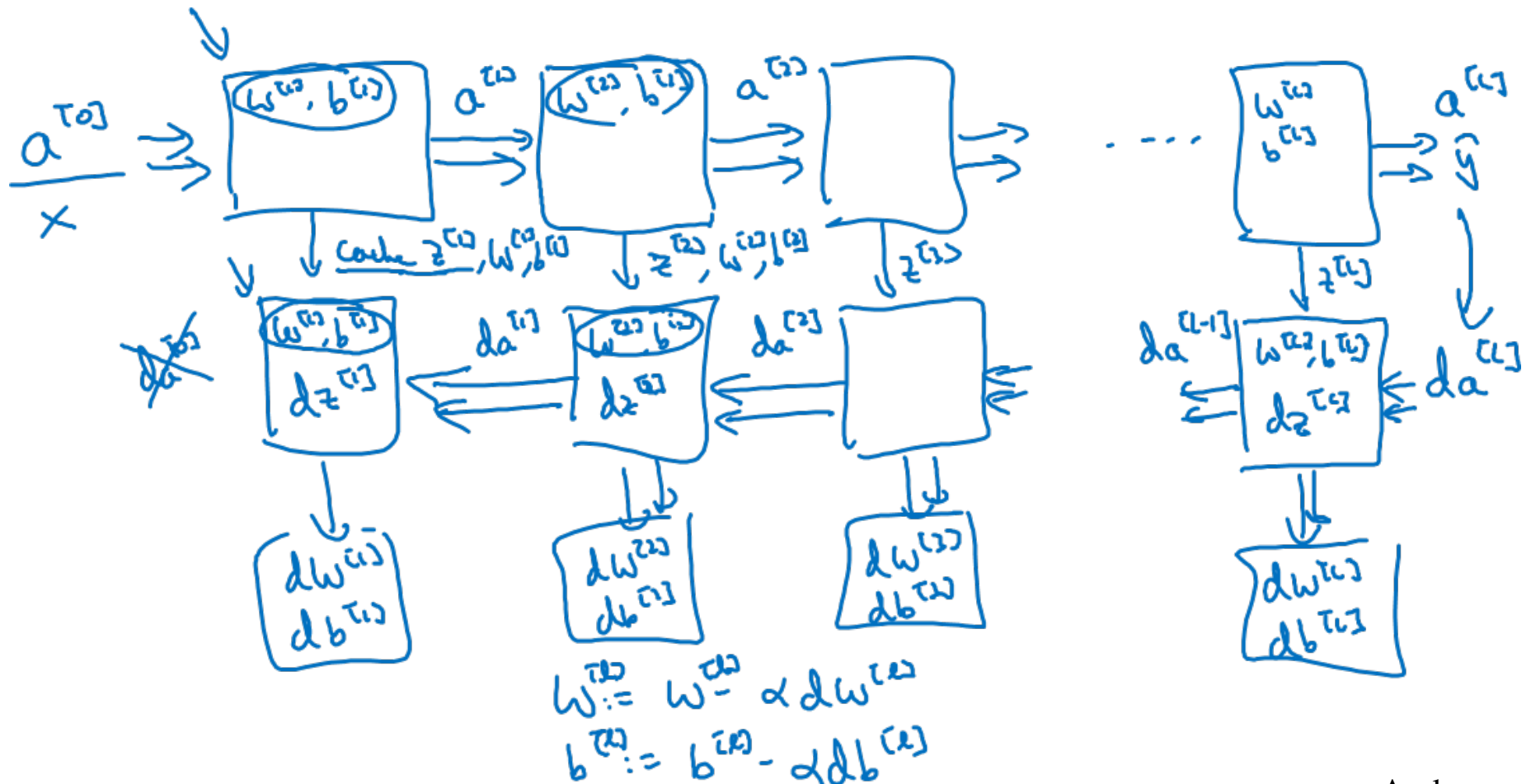
$$z^{[l]} = W^{[l]} a^{[l-1]} + b^{[l]} \quad \text{cache } z^{[l]}$$

$$a^{[l]} = g^{[l]}(z^{[l]})$$

→ Backward: Input $da^{[l]}$, cache $z^{[l]}$, output $da^{[l-1]}$
 $\frac{dL}{dW^{[l]}}$
 $\frac{dL}{db^{[l]}}$



Forward and backward functions





deeplearning.ai

Deep Neural Networks

**Forward and backward
propagation**

Backward propagation for layer l

→ Input $da^{[l]}$

→ Output $da^{[l-1]}, dW^{[l]}, db^{[l]}$

$$dz^{[l]} = da^{[l]} * g^{[l]'}(z^{[l]})$$

$$dW^{[l]} = dz^{[l]} \cdot \underline{a^{[l-1]}}$$

$$db^{[l]} = dz^{[l]}$$

$$da^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$$

$$dz^{[l]} = W^{[l+1]T} dz^{[l+1]} * g^{[l]'}(z^{[l]})$$

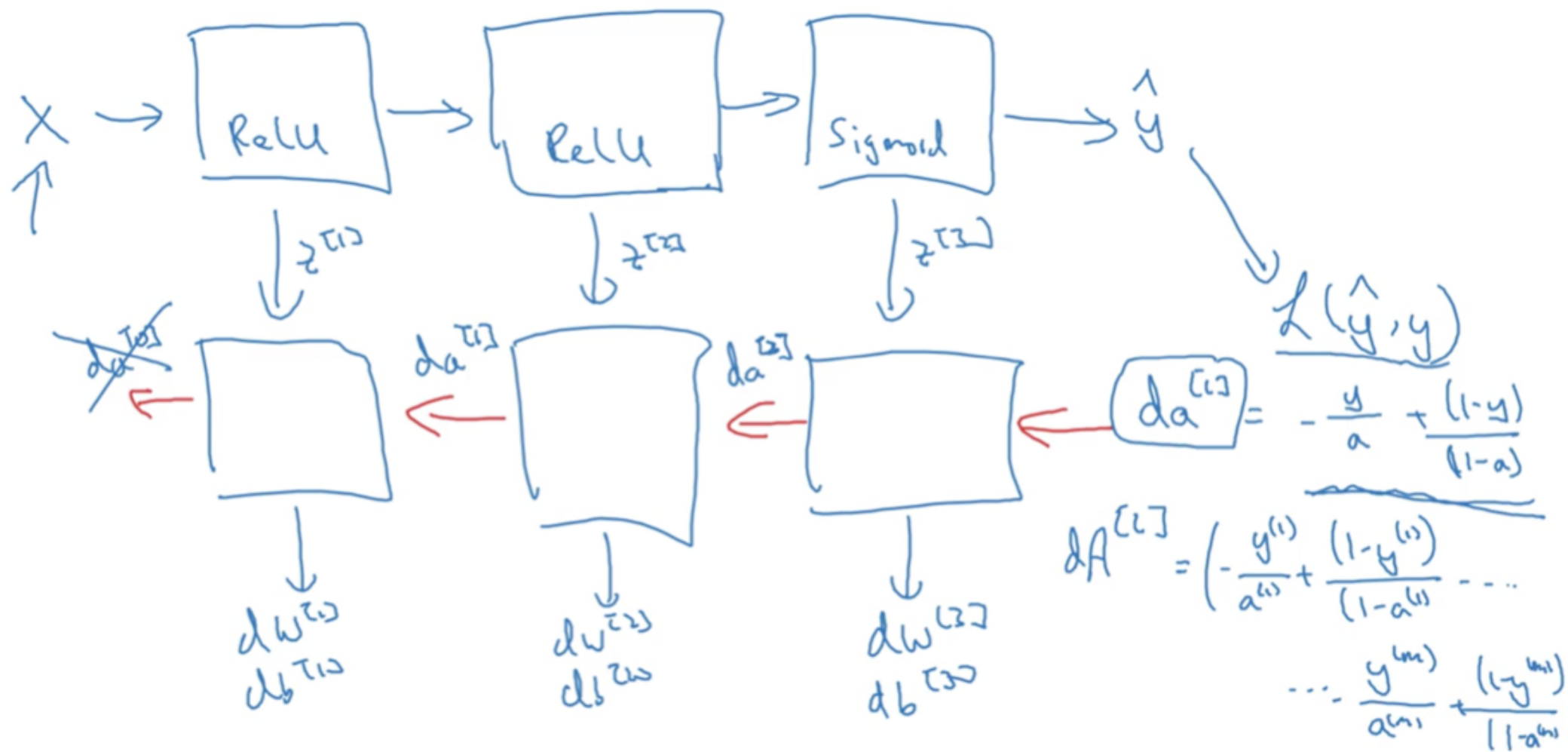
$$dz^{[l]} = dA^{[l]} * g^{[l]'}(z^{[l]})$$

$$dW^{[l]} = \frac{1}{n} dz^{[l]} \cdot A^{[l-1]T}$$

$$db^{[l]} = \frac{1}{n} \text{np.sum}(dz^{[l]}, \text{axis}=1, \text{keepdims}=\text{True})$$

$$dA^{[l-1]} = W^{[l]T} \cdot dz^{[l]}$$

Summary





deeplearning.ai

Deep Neural Networks

Parameters vs Hyperparameters

What are hyperparameters?

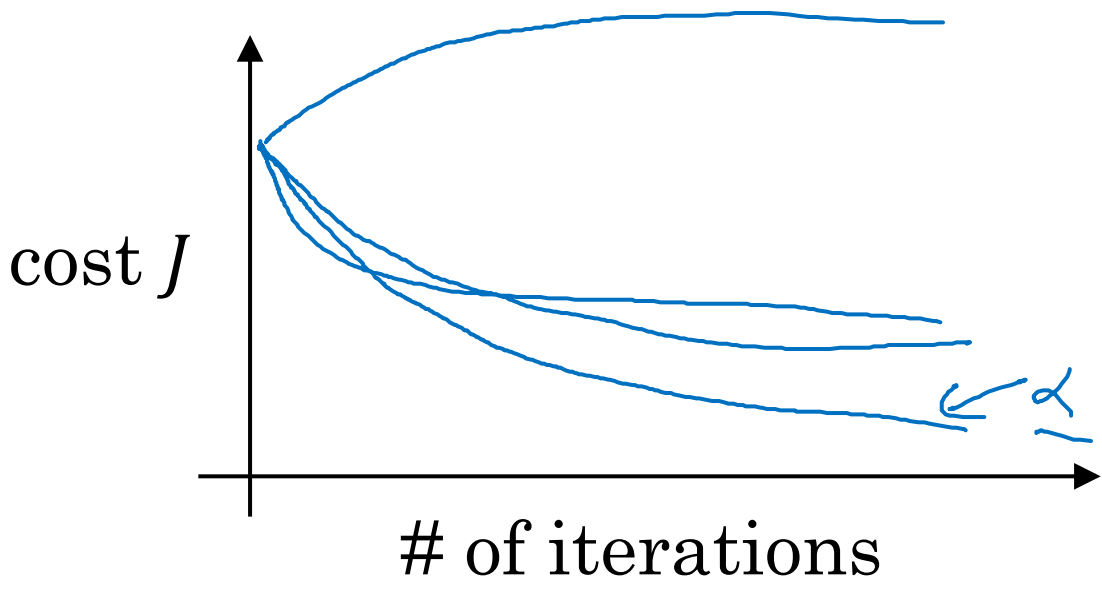
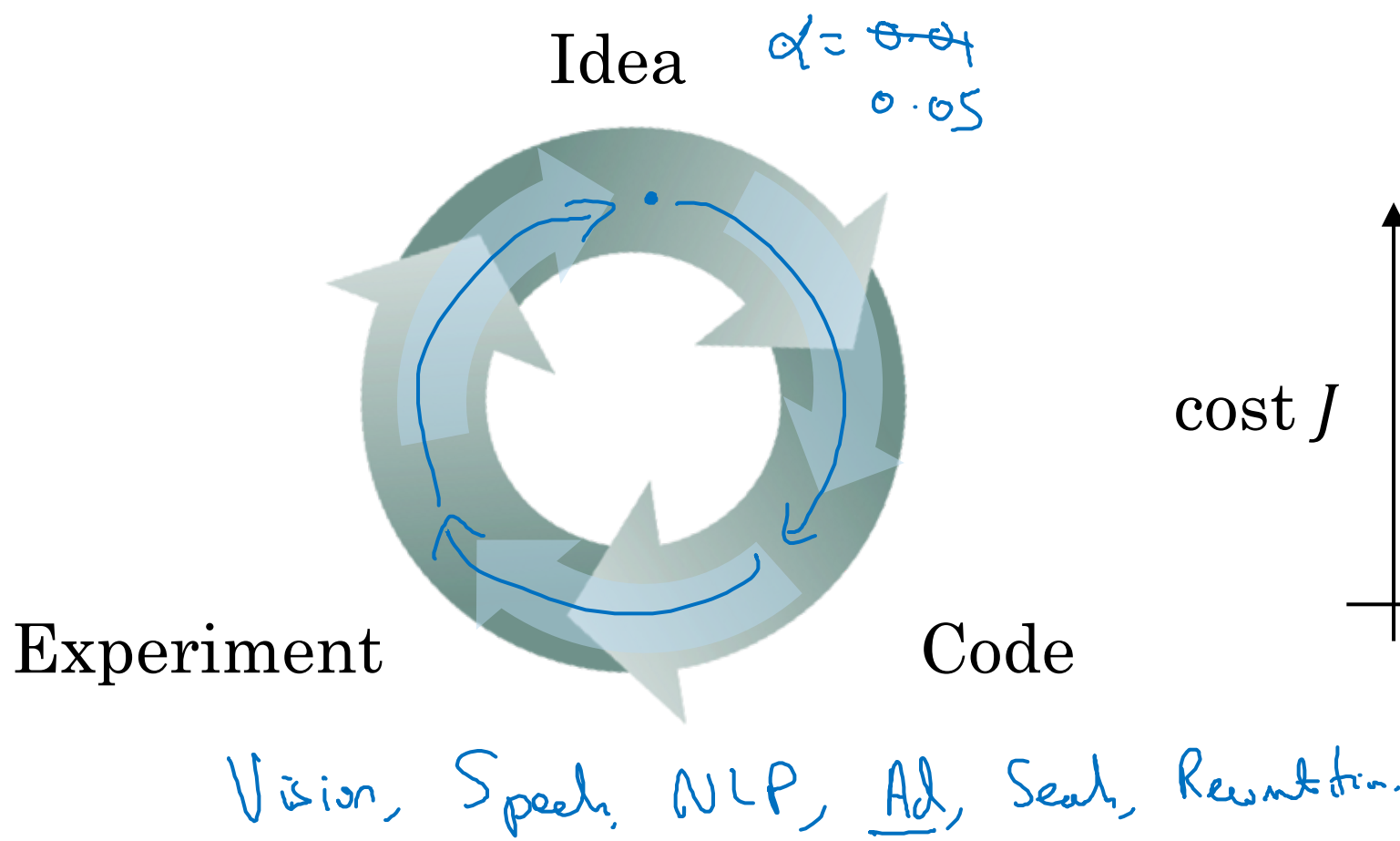
Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]} \dots$

Hyperparameters:

- learning rate α
- # iterations τ
- # hidden layers L
- # hidden units $n^{[1]}, n^{[2]}, \dots$
- choice of activation function

Other: Momentum, mini-batch size, regularizations, ...

Applied deep learning is a very empirical process





deeplearning.ai

Deep Neural Networks

What does this
have to do with
the brain?

Forward and backward propagation

$$\begin{aligned}Z^{[1]} &= W^{[1]}X + b^{[1]} \\A^{[1]} &= g^{[1]}(Z^{[1]}) \\Z^{[2]} &= W^{[2]}A^{[1]} + b^{[2]} \\A^{[2]} &= g^{[2]}(Z^{[2]}) \\&\vdots \\A^{[L]} &= g^{[L]}(Z^{[L]}) = \hat{Y}\end{aligned}$$

"It's like the brain"



$$\begin{aligned}dZ^{[L]} &= A^{[L]} - Y \\dW^{[L]} &= \frac{1}{m} dZ^{[L]} A^{[L]T} \\db^{[L]} &= \frac{1}{m} np.sum(dZ^{[L]}, axis = 1, keepdims = True) \\dZ^{[L-1]} &= dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]}) \\&\vdots \\dZ^{[1]} &= dW^{[L]T} dZ^{[2]} g'^{[1]}(Z^{[1]}) \\dW^{[1]} &= \frac{1}{m} dZ^{[1]} A^{[1]T} \\db^{[1]} &= \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)\end{aligned}$$

