



deeplearning.ai

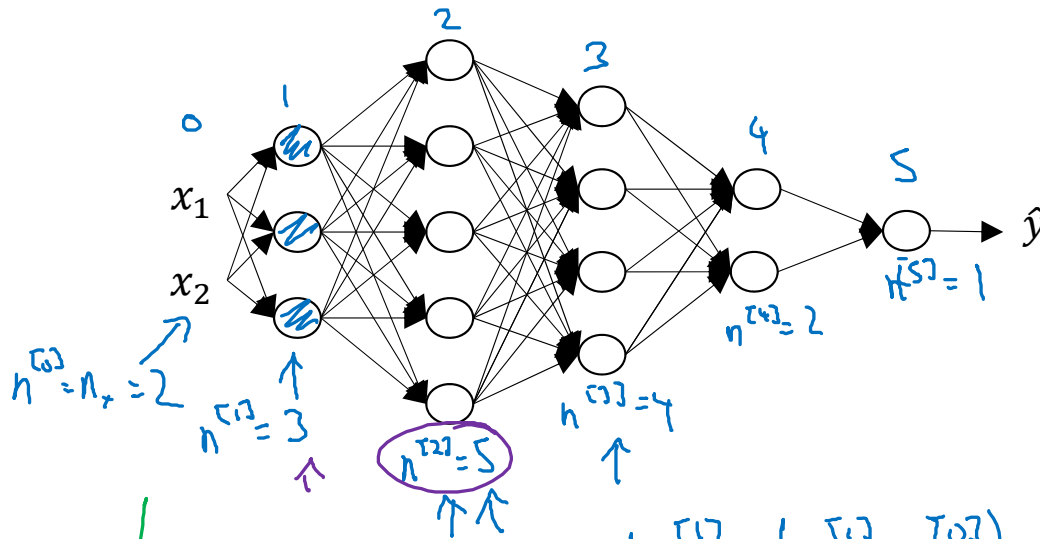
# Deep Neural Networks

---

Getting your matrix  
dimensions right

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

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



$L=5$

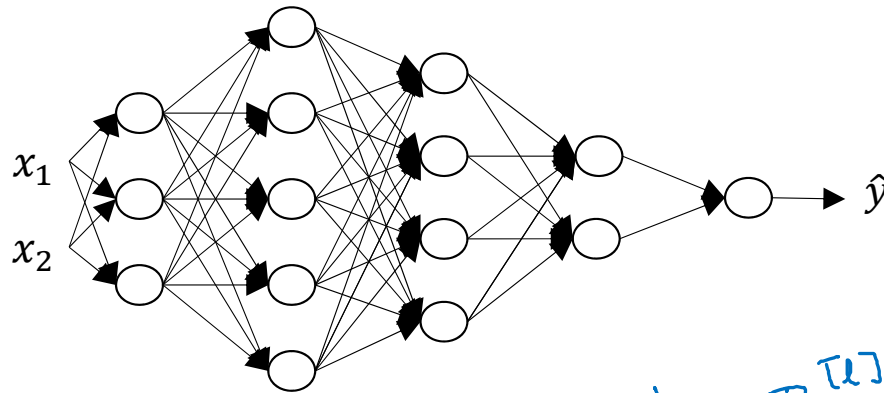
$\rightarrow W^{[L]}: (n^{[L]}, n^{[L-1]})$   
 $\rightarrow b^{[L]}: (n^{[L]}, 1)$   
 $\rightarrow \Delta W^{[L]}: (n^{[L]}, n^{[L-1]})$   
 $\rightarrow \Delta b^{[L]}: (n^{[L]}, 1)$

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

$\begin{bmatrix} \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} \vdots \\ \vdots \end{bmatrix} \begin{bmatrix} \vdots \end{bmatrix}$

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

# Vectorized implementation



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

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

$[z^{[1]}, z^{[2]}, \dots, z^{[L]}]$

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

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

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


---


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

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

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



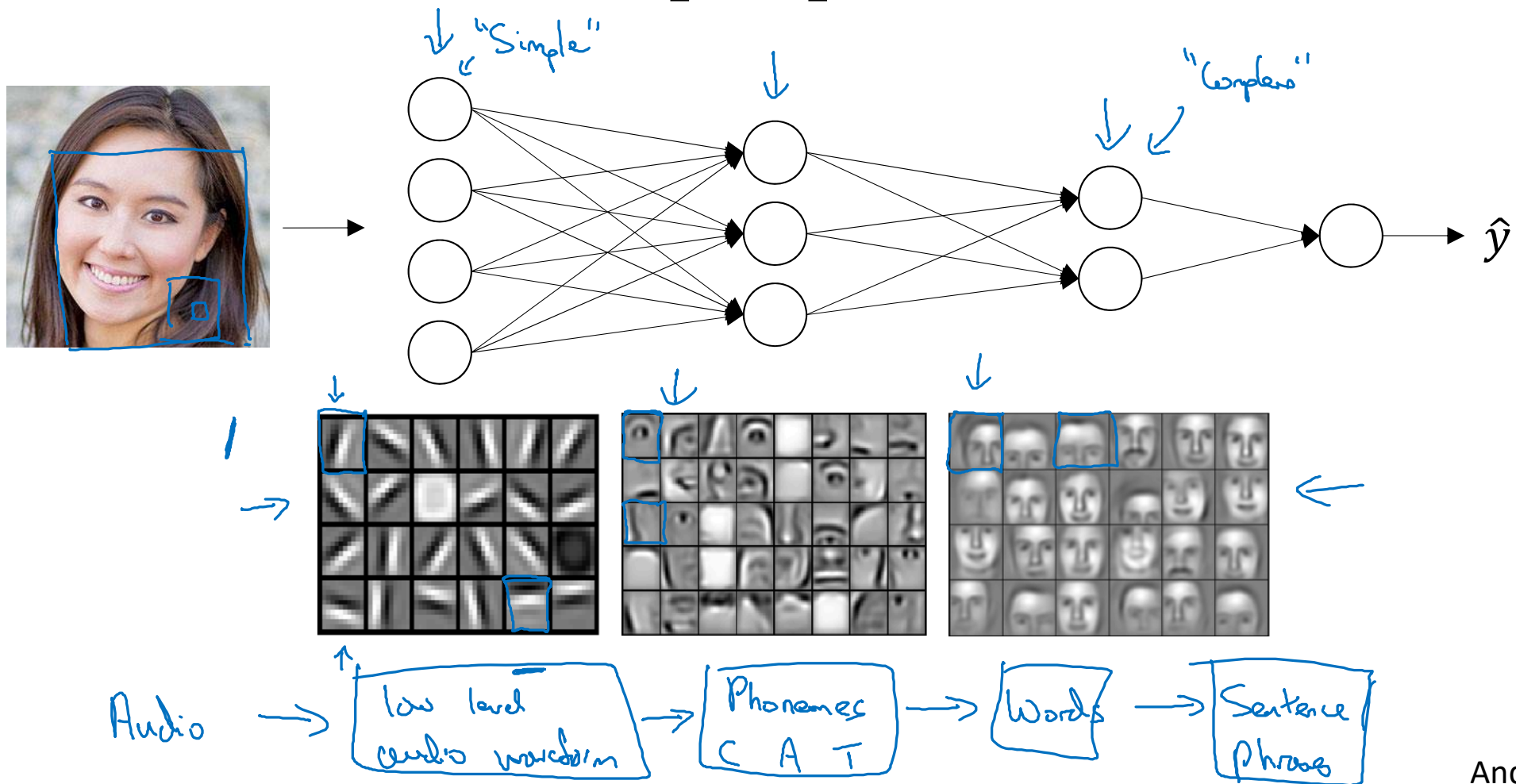
deeplearning.ai

# Deep Neural Networks

---

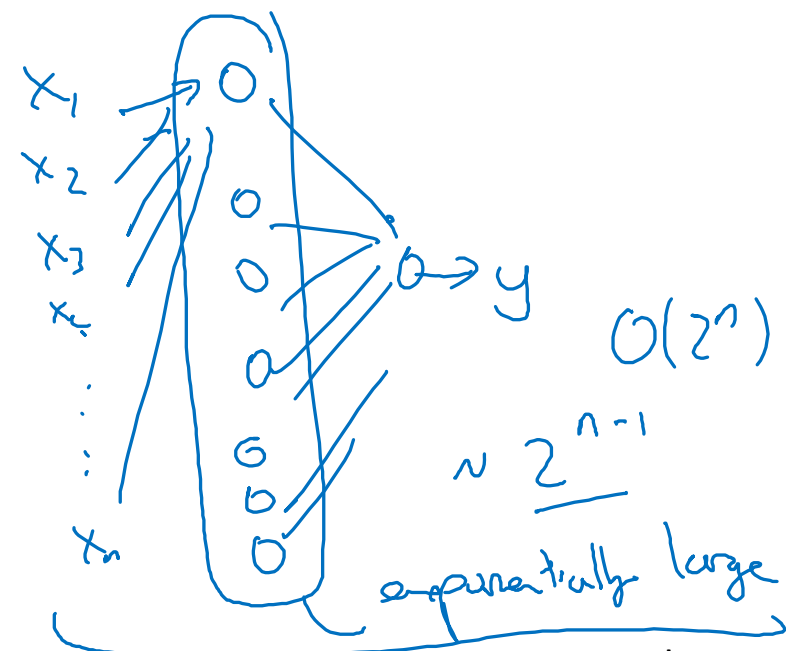
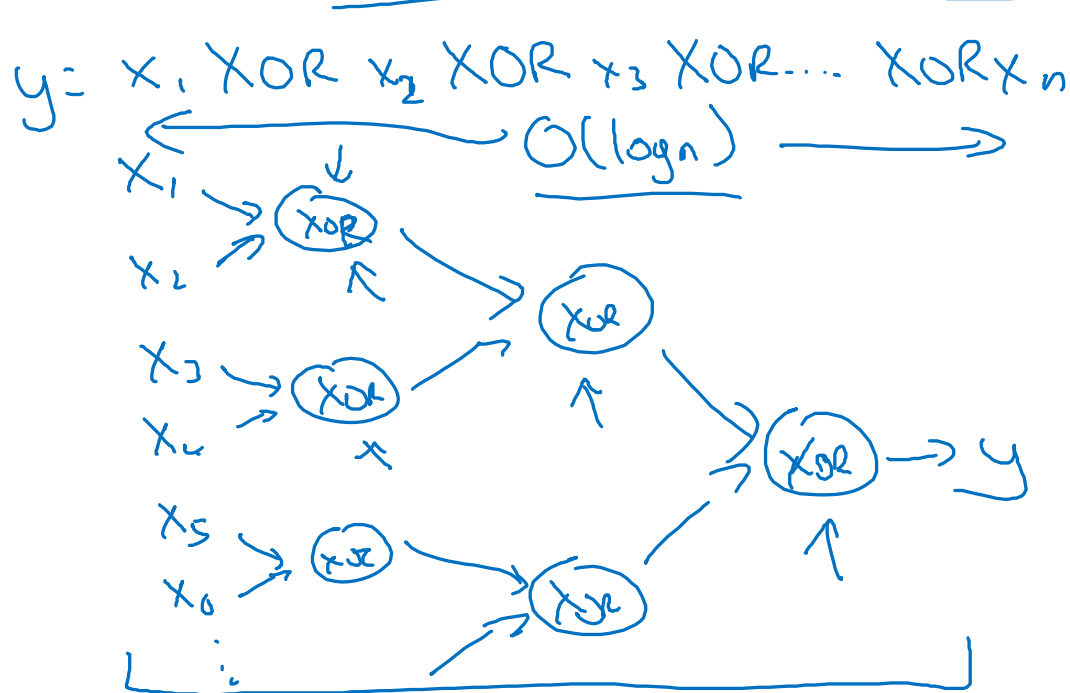
Why deep  
representations?

# 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.





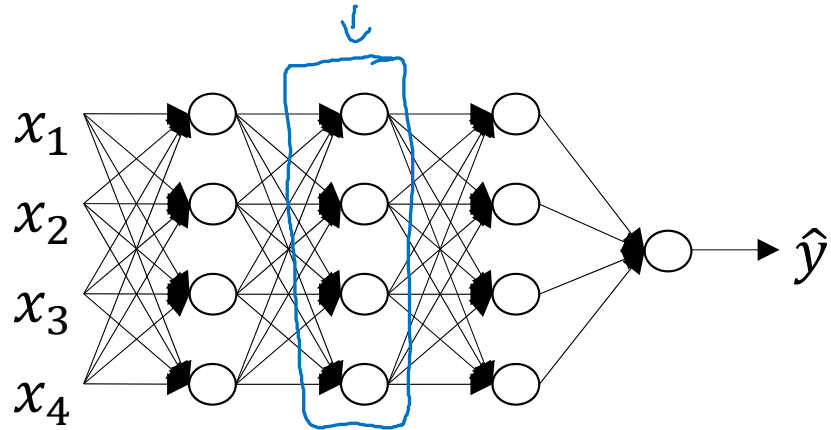
deeplearning.ai

# Deep Neural Networks

---

Building blocks of  
deep neural networks

# Forward and backward functions



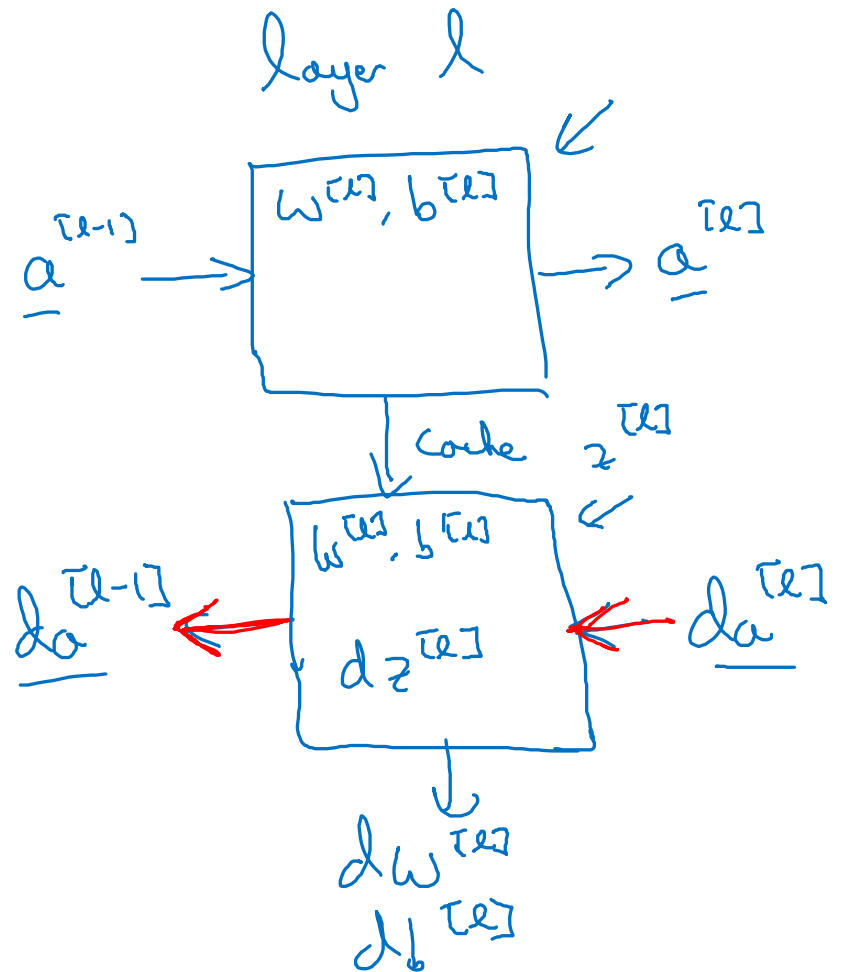
layer  $l$ :  $W^{[l]}, b^{[l]}$

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

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

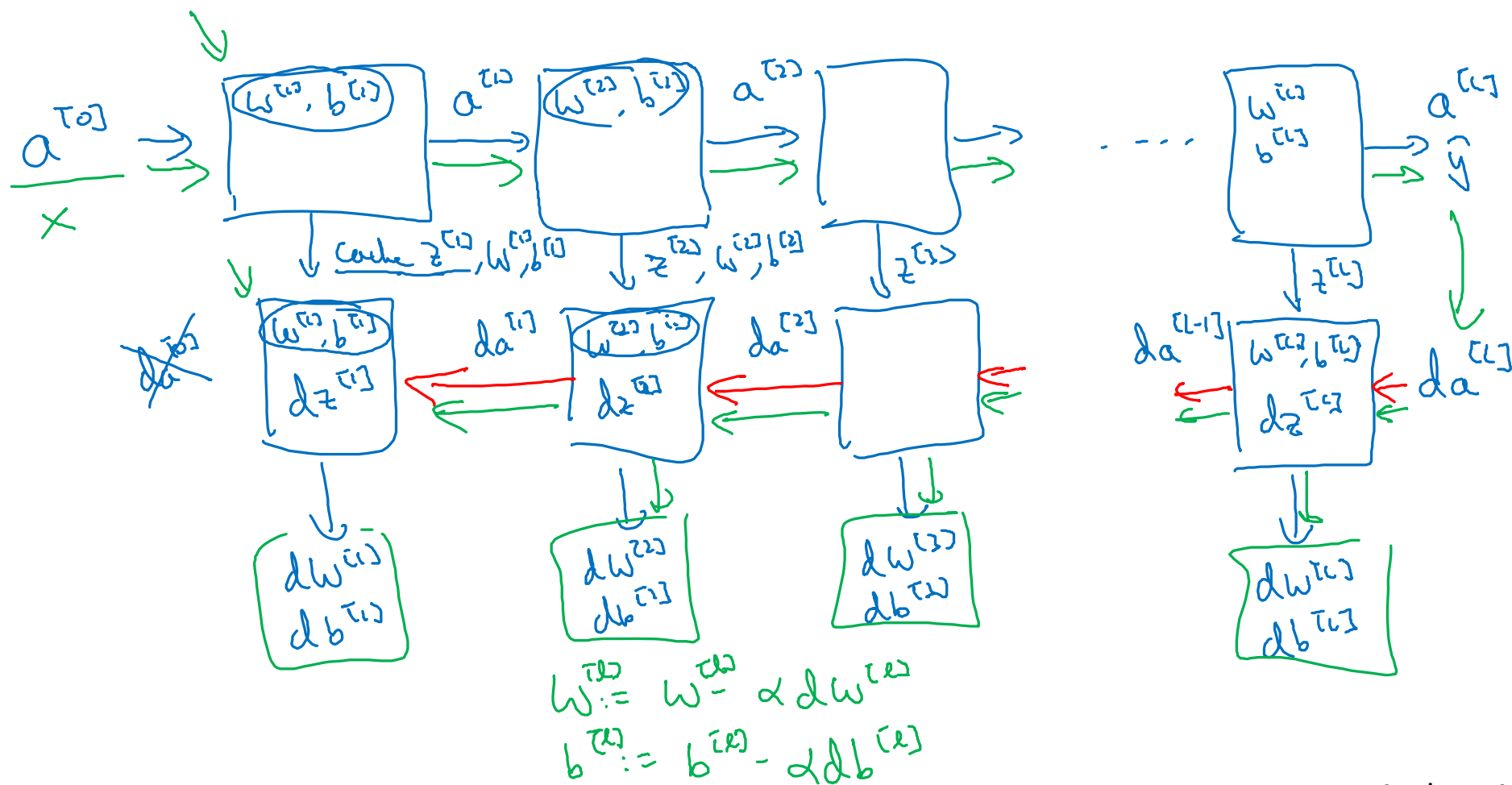
$$a^{[l]} = g^{[l]}(z^{[l]})$$

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





# Forward and backward functions





deeplearning.ai

# Deep Neural Networks

---

Forward and backward  
propagation

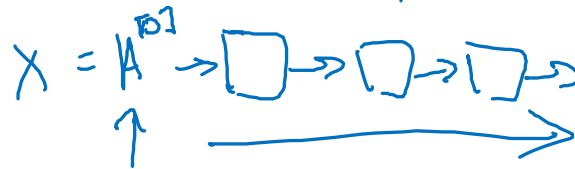
# Forward propagation for layer $l$

→ Input  $a^{[l-1]}$  ←

→ Output  $a^{[l]}$ , cache ( $z^{[l]}$ )

$$z^{[l]} = W^{[l]} \cdot a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

$$a^{[0]}$$
$$A^{[0]}$$



Vectorized:

$$z^{[l]} = W^{[l]} \cdot A^{[l-1]} + b^{[l]}$$
$$A^{[l]} = g^{[l]}(z^{[l]})$$

# Backward propagation for layer $l$

→ Input  $da^{[l]}$

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

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

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

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

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

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

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

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

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

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

# Summary

