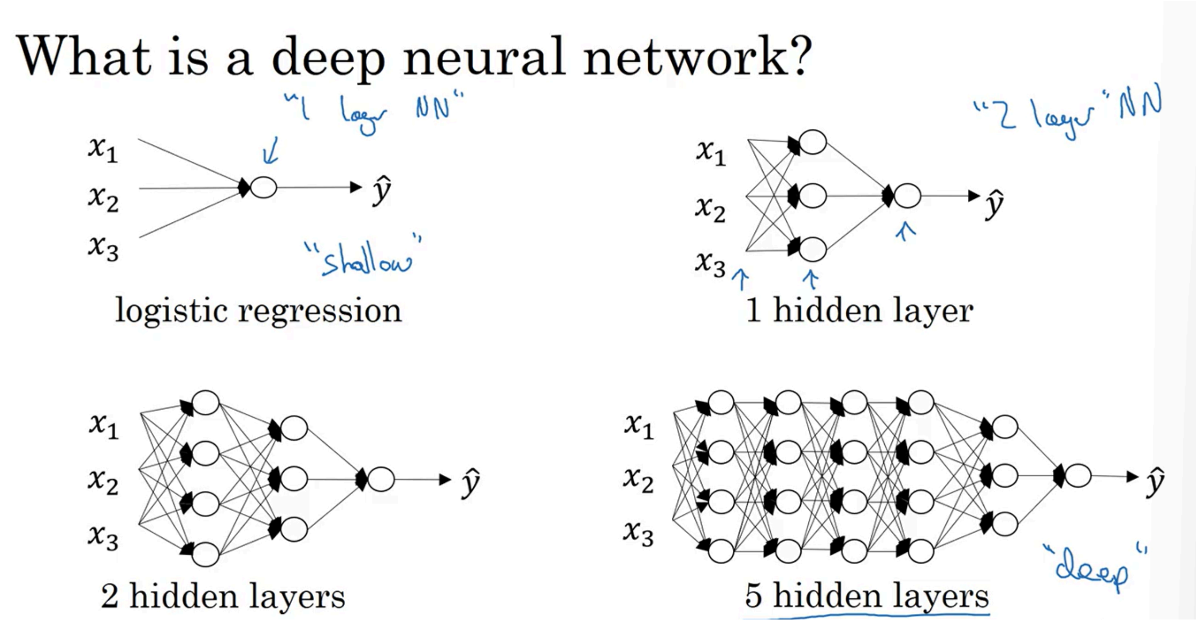
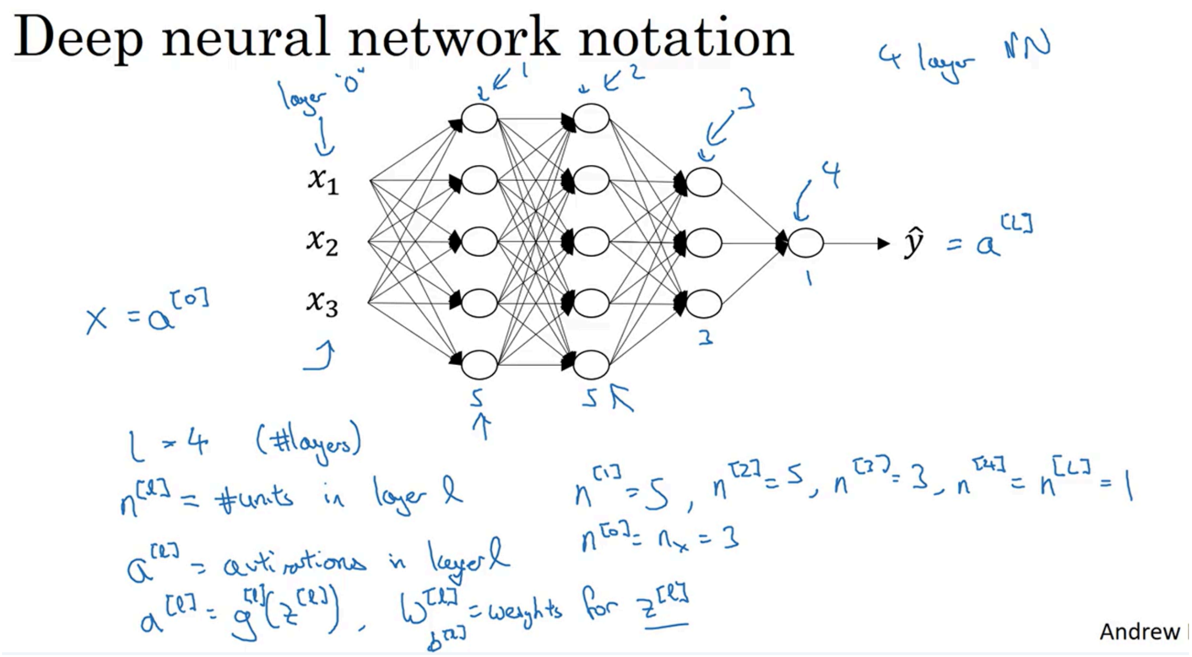
Week 4: Deep Neural Networks





Notations

1. **Input & Activations**
    - $X = a^{[0]}$ → Input vector is treated as activation of layer 0.
    - $a^{[l]}$ → Activations (outputs) of layer l.
    - $\hat{y} = a^{[4]}$ → Final output prediction.
2. **Layers & Units**
    - $L = 4$ → Number of layers (here including output layer).
    - $n^{[l]}$ → Number of units (neurons) in layer l.

      From figure:

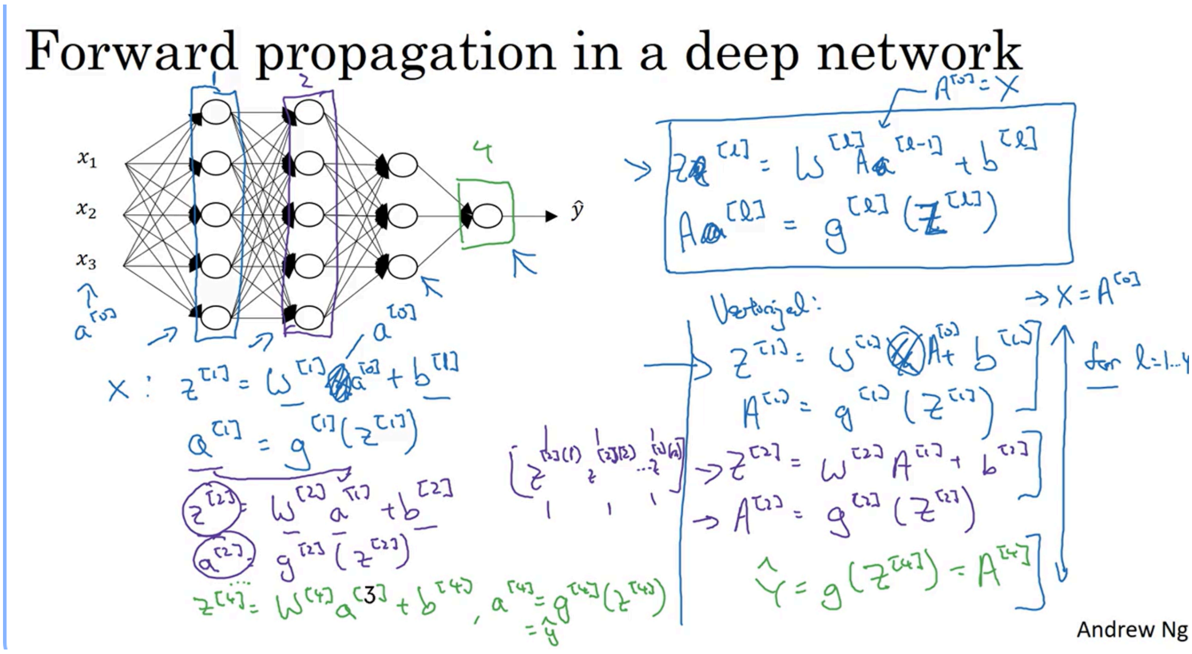
        - $n^{[1]} = 5$ (5 neurons in 1st hidden layer)
        - $n^{[2]} = 5$ (5 neurons in 2nd hidden layer)
        - $n^{[3]} = 3$ (3 neurons in 3rd hidden layer)
        - $n^{[4]} = 1$ (1 neuron in output layer)
        - $n^{[0]} = n_x = 3$ (3 input features).
3. **Forward Propagation Equations**
    - Pre-activation (linear combination):

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

    - Activation:

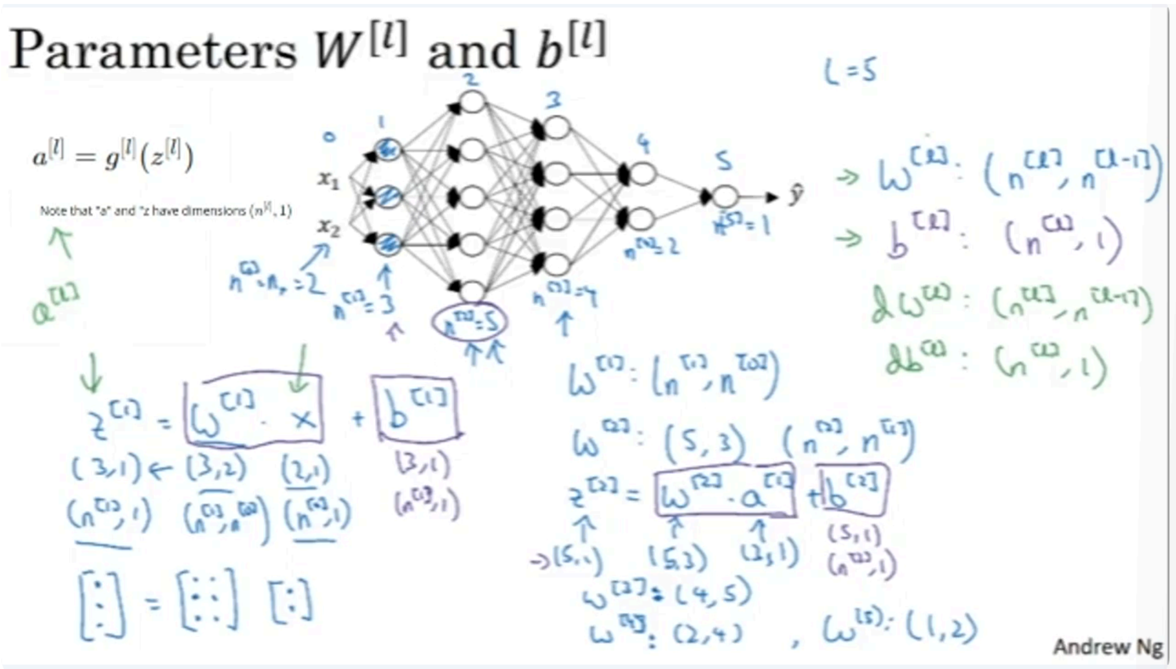
      $$a^{[l]} = g(z^{[l]})$$

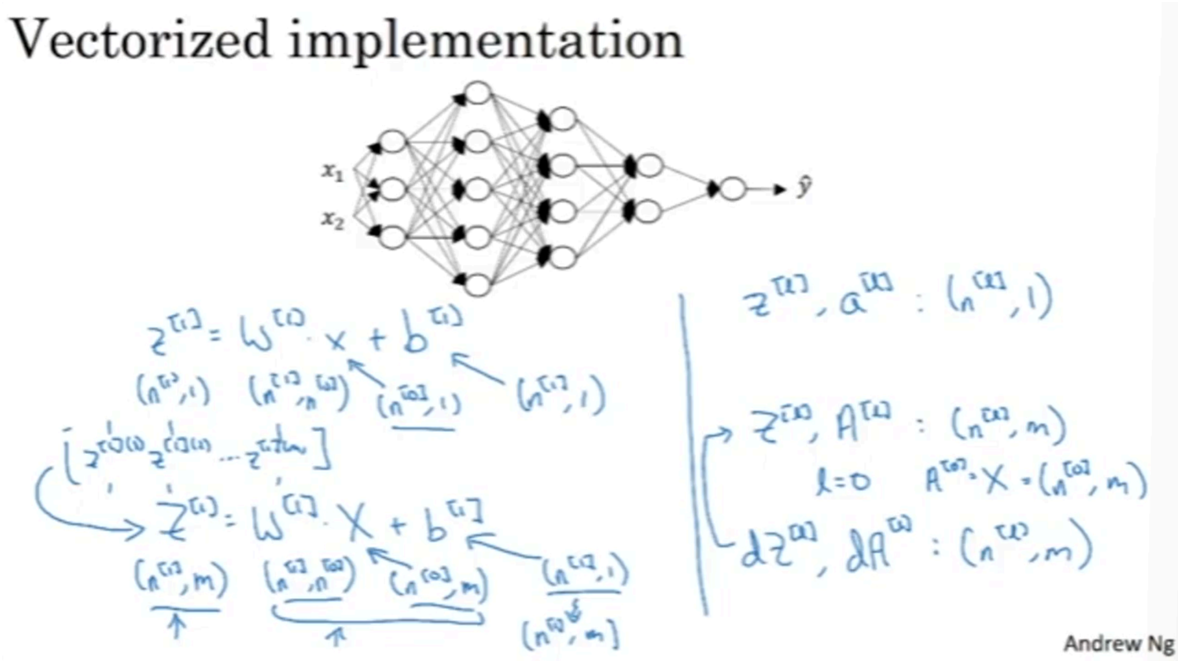
      where g is the activation function.
4. **Parameters**
    - $W^{[l]}$ → Weight matrix for layer l.
    - $b^{[l]}$ → Bias vector for layer l.



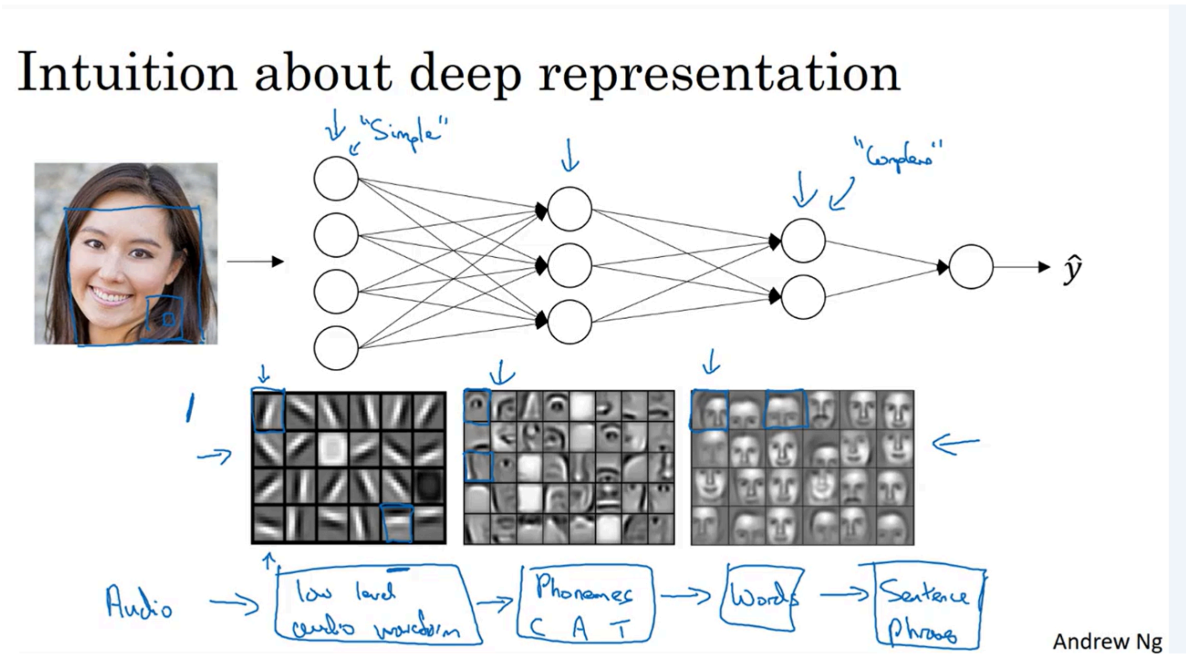
we can have one explicit for loop here

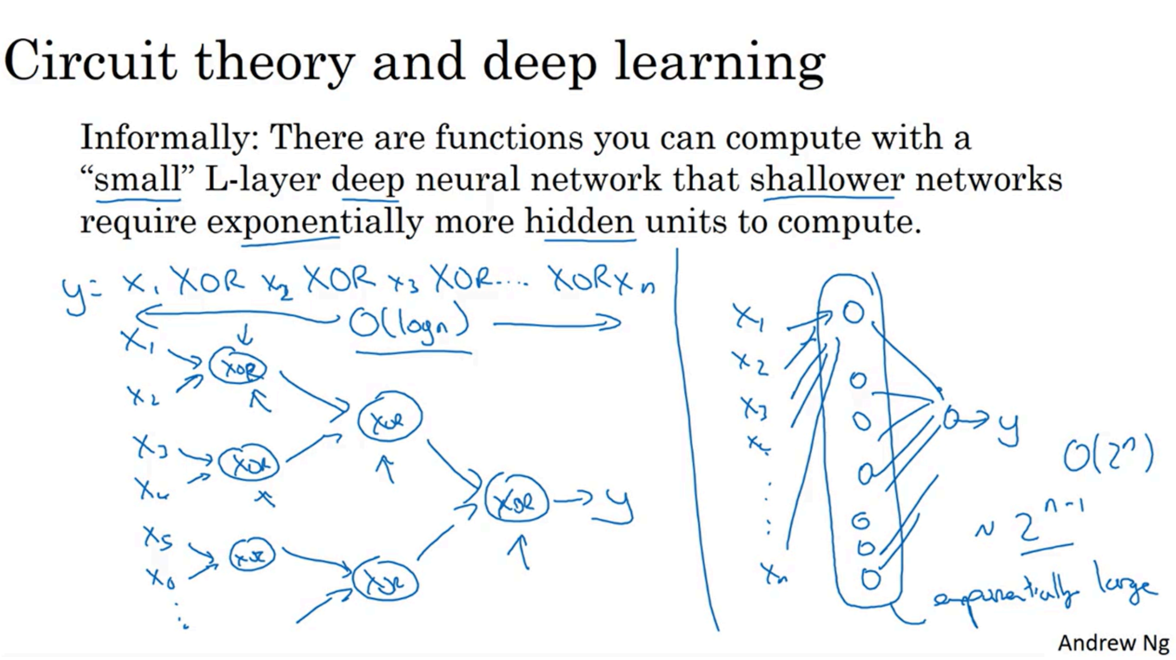
Getting your Matrix Dimensions Right





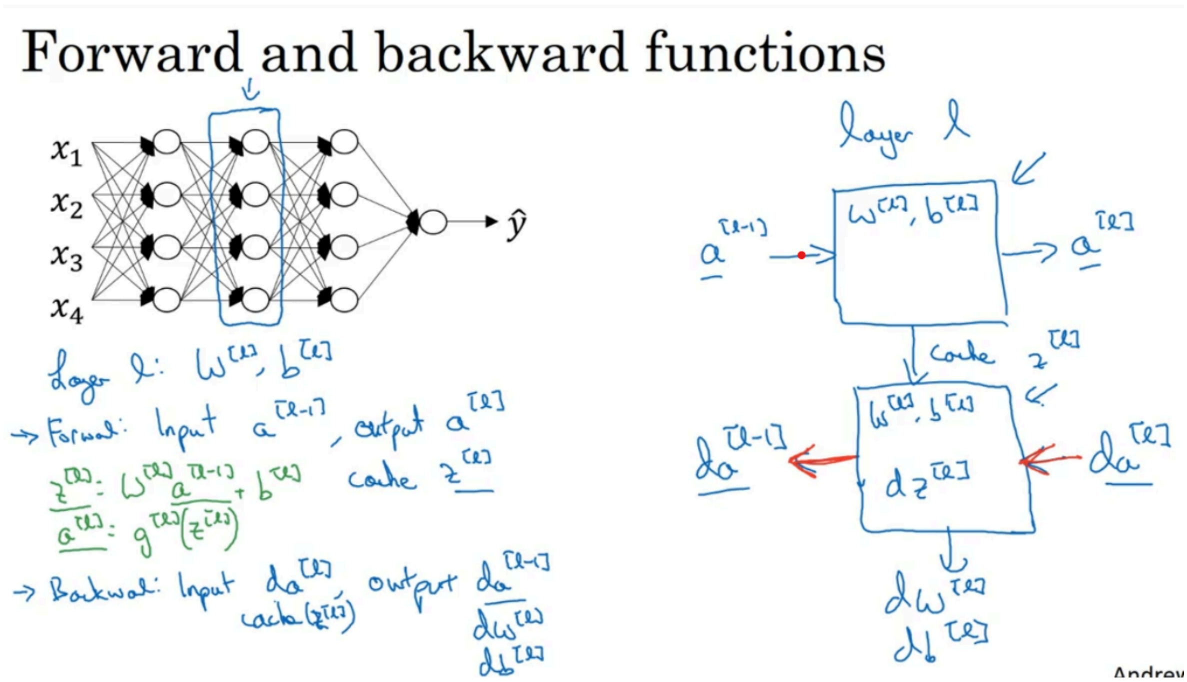
Why Deep Representations?



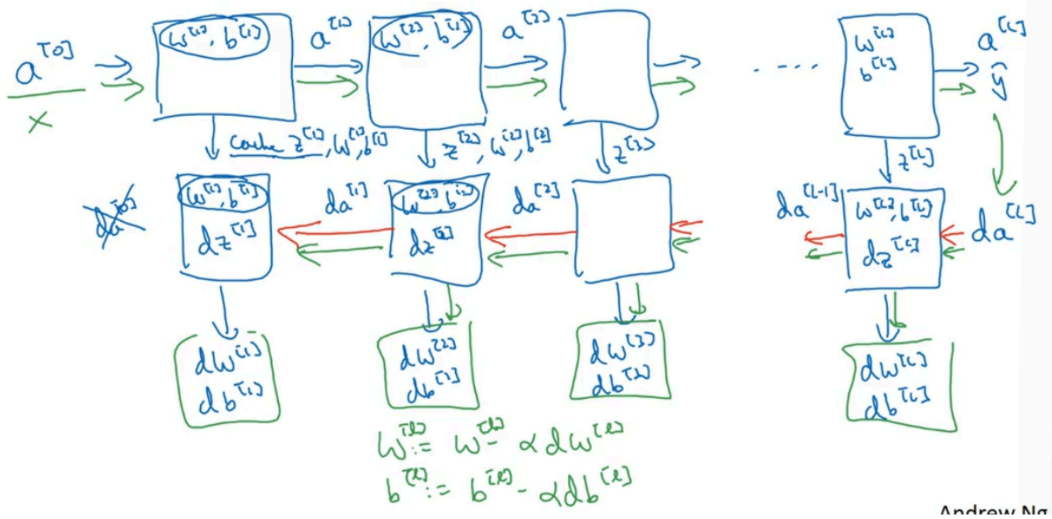


Building Blocks of Deep Neural Networks

for one layer:



Forward and backward functions



Forward propagation for layer l

→ Input $a^{[l-1]}$

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

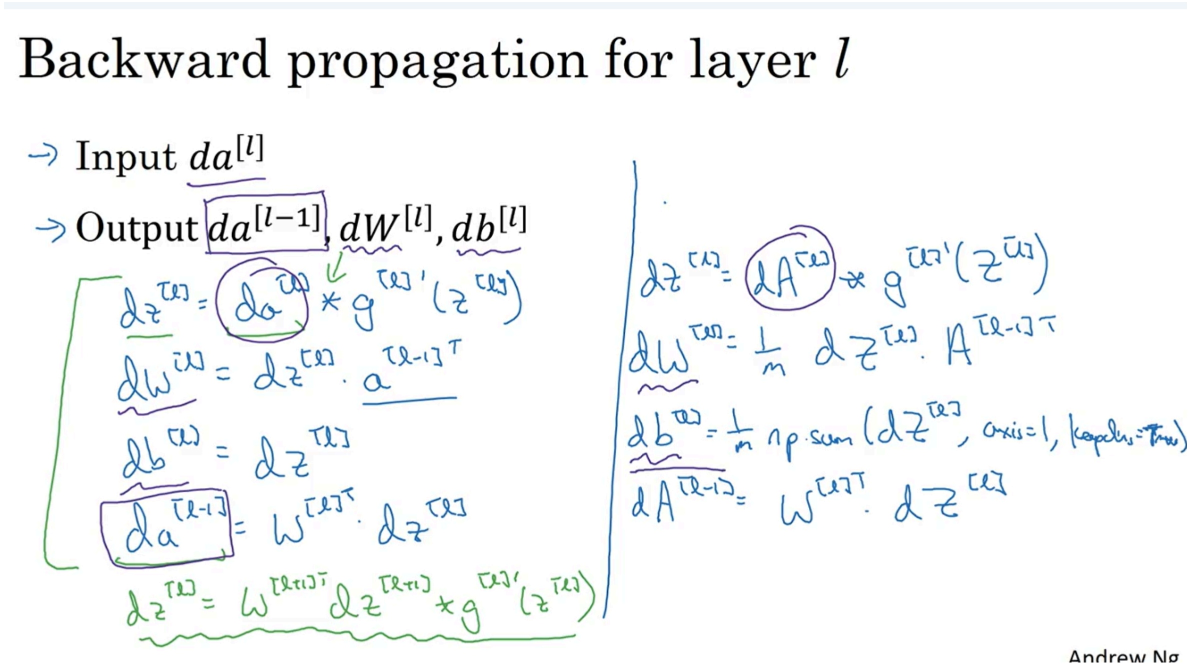
$$z^{[l]} = W^{[l]} \cdot a^{[l-1]} + b^{[l]}$$
$$a^{[l]} = g^{[l]}(z^{[l]})$$

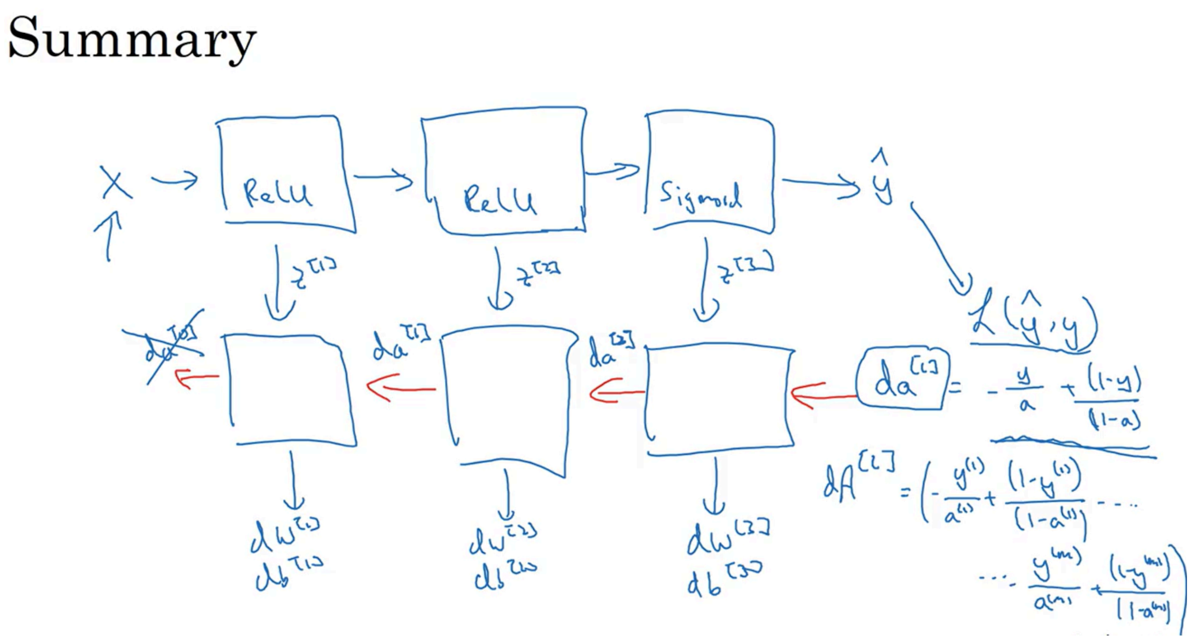
Vectorized:

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

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

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

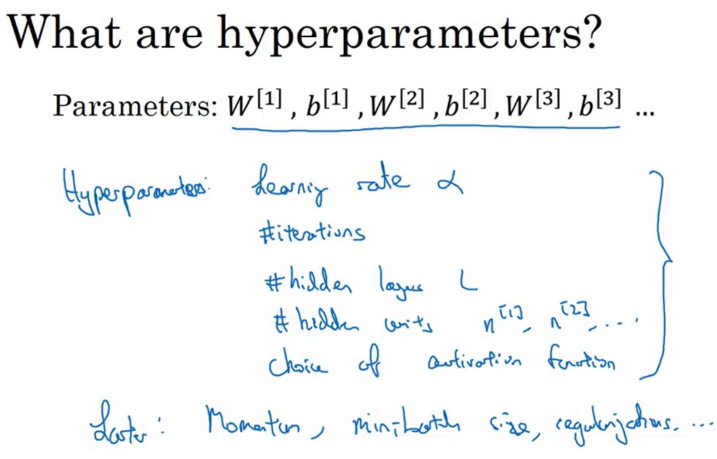




optional notes:

https://community.deeplearning.ai/t/feedforward-neural-networks-in-depth/98811

Parameters vs Hyperparameters

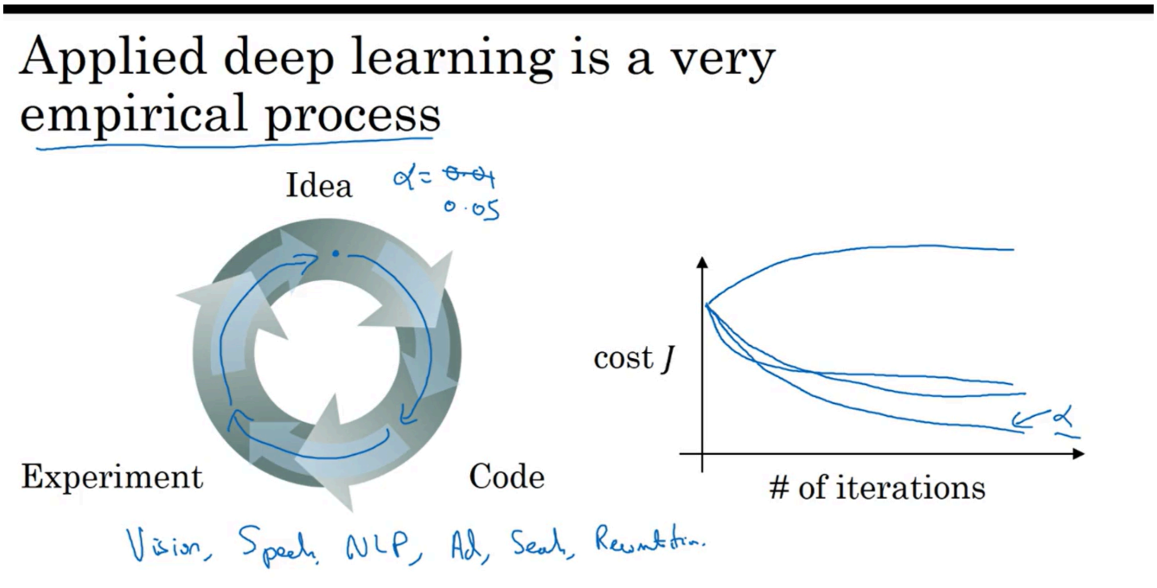


Mnemonic: "Para = Inside, Hyper = Outside

Aspect	Parameters	Hyperparameters
Definition	Internal variables learned by the model	External settings chosen before training
Examples	Weights, Biases, CNN filters, RNN hidden weights	Learning rate, Batch size, Epochs, Dropout rate, Optimizer type
Who decides them?	Learned automatically during training	Set manually or tuned by developer
When set?	Updated throughout training via backpropagation	Fixed before training (can be tuned between runs)
Role	Define the model's knowledge	Control how the model learns
Optimization	Learned using gradient descent	Tuned using Grid Search, Random Search, Bayesian Optimization, etc.
Analogy	What the student learns (knowledge)	The study plan or strategy given by the teacher

Quick Analogy:

- **Parameters = Brain cells' knowledge** (learnt facts).
- **Hyperparameters = Study plan** (how long to study, what method to use).



Clarification For: What does this have to do with the brain?

Note that the formulas shown in the next video have a few typos. Here is the correct set of formulas.

$dZ^{[L]} = A^{[L]} - Y$

$dW^{[L]} = \frac{1}{m} dZ^{[L]} A^{[L-1]^T}$

$db^{[L]} = \frac{1}{m} np.sum(dZ^{[L]}, axis = 1, keepdims = True)$

$dZ^{[L-1]} = W^{[L]^T} dZ^{[L]} * g'^{[L-1]}(Z^{[L-1]})$

Note that * denotes element-wise multiplication)

$\vdots$

$dZ^{[1]} = W^{[2]^T} dZ^{[2]} * g'^{[1]}(Z^{[1]})$

$dW^{[1]} = \frac{1}{m} dZ^{[1]} A^{[0]^T}$

Note that $A^{[0]^T}$ is another way to denote the input features, which is also written as X^T

$db^{[1]} = \frac{1}{m} np.sum(dZ^{[1]}, axis = 1, keepdims = True)$