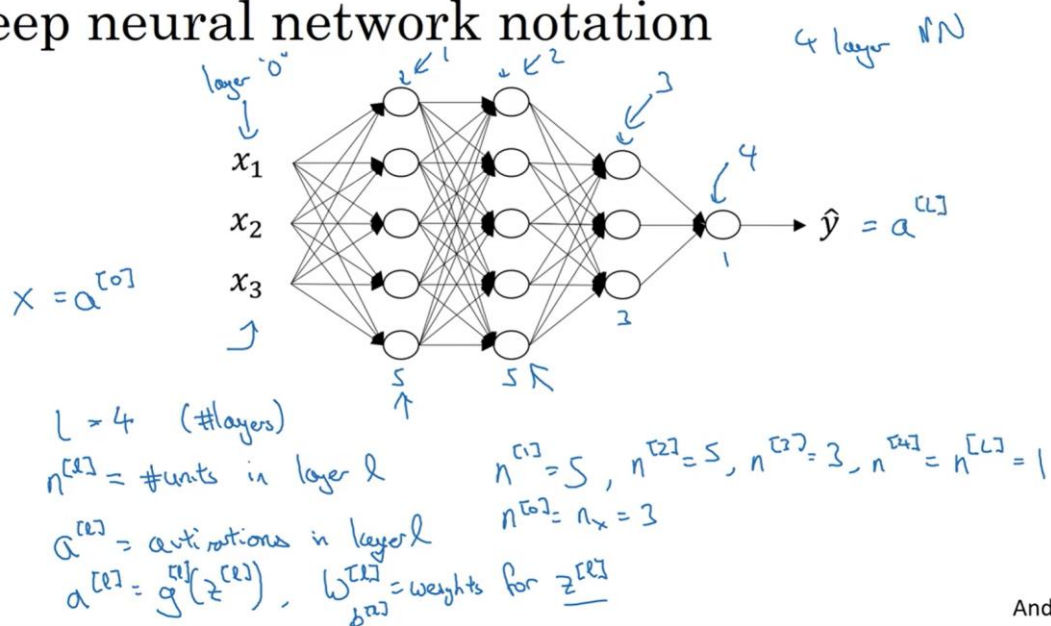


# COURSE 1 W4

## Vid 36

### Deep L-Layer Neural Network

#### Deep neural network notation



Andrew Ng

## Vid 37

### Forward and Backward Propagation

#### Forward propagation for layer $l$

→ Input  $a^{[l-1]}$

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

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

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

Verzorgd:

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

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

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

Andrew Ng

## Backward propagation for layer $l$

→ Input  $da^{[l]}$

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

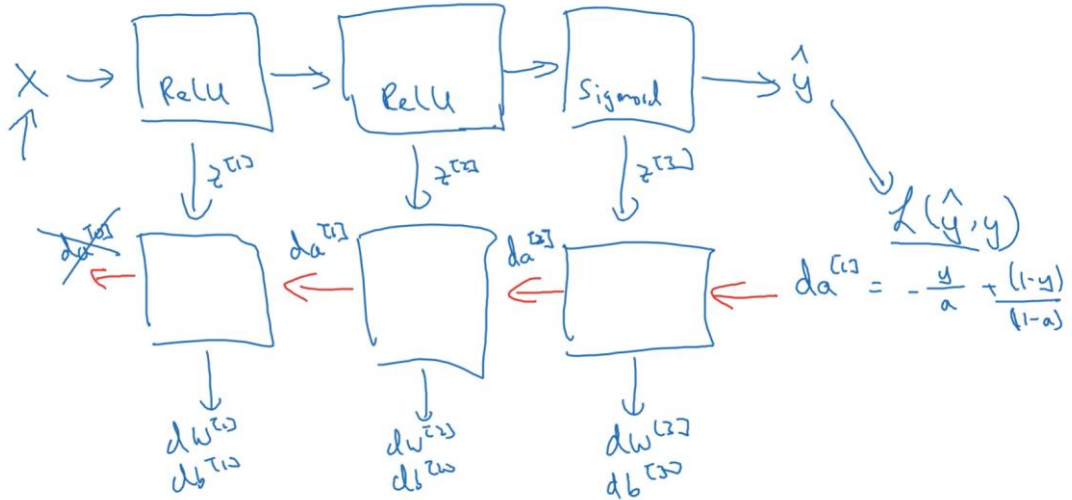
$$\begin{aligned} dz^{[l]} &= da^{[l]} * g'^{[l]}(z^{[l]}) \\ dW^{[l]} &= dz^{[l]} \cdot a^{[l-1]} \\ db^{[l]} &= dz^{[l]} \\ da^{[l-1]} &= W^{[l]T} \cdot dz^{[l]} \\ dz^{[l]} &= W^{[l+1]T} dz^{[l+1]} * g'^{[l+1]}(z^{[l+1]}) \end{aligned}$$

$$\begin{aligned} dz^{[l]} &= dA^{[l]} * g'^{[l]}(z^{[l]}) \\ dW^{[l]} &= \frac{1}{n} dz^{[l]} \cdot A^{[l-1]T} \\ db^{[l]} &= \frac{1}{n} np.sum(dz^{[l]}, axis=1, keepdims=True) \\ dA^{[l-1]} &= W^{[l]T} \cdot dz^{[l]} \end{aligned}$$

Andrew Ng

Sağdaki vectorized version.

## Summary



Andrew Ng

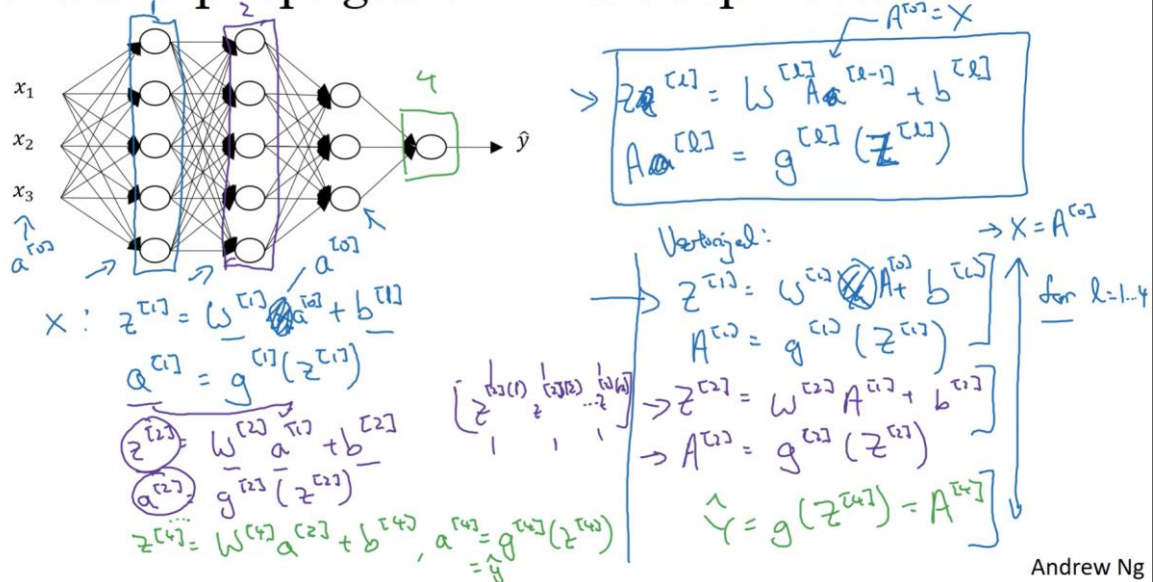
Özetle önce forward propagation ile  $\hat{y}$  ve loss hesaplanıyor daha sonra, aynı yapı üzerinde geriye doğru  $dW_3$   $db_3$  den başlayarak geriye doğru geliniyor,  $z_1, z_2, z_3$  cache de tutuluyor sanırım türev hesaplamaları için kullanılıyor. Backwarda başlarken  $da^{(l)}$  ile başlıyoruz, bu da loss'un  $\hat{y}$ 'e göre türevinden başka bir şey değil.

Bunlar zaten bildiğimiz şeyler yeni bir şey değil. Özellikle bakman gerekirse assignment üzerinde dener bakarsın.

## Vid 38

### Forward Propagation in a Deep Neural Network

#### Forward propagation in a deep network



Andrew Ng

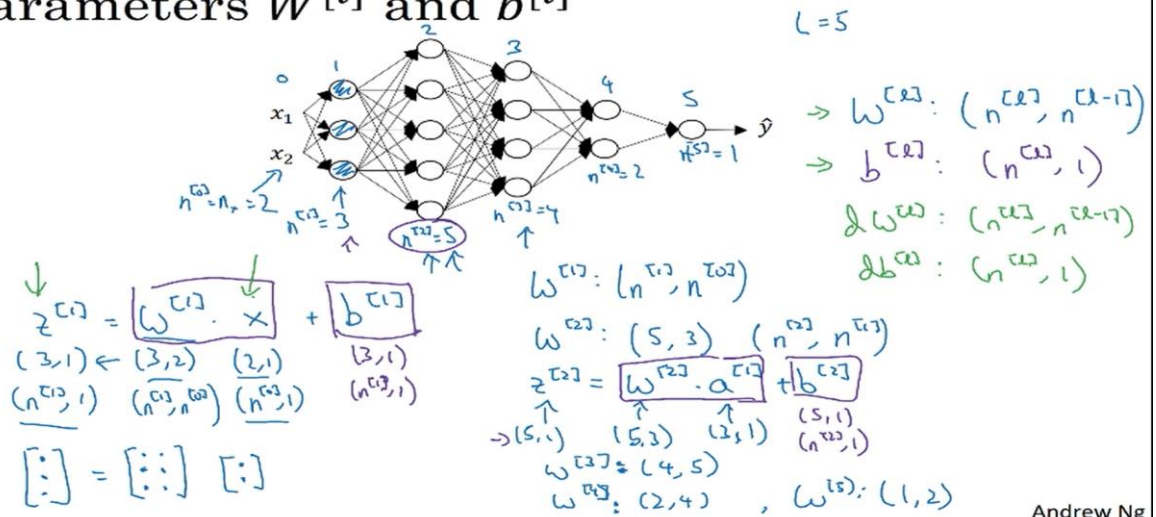
Burada sağ alttaki vectorized version için her layer için dahi her layer'ı dolandığımızı dikkat et yani burada bir for loop söz konusu, bunu vektörize edemiyoruz.

## Vid 39

### Getting Matrix Dimensions Right

NN implement ederken kullanılabilecek debugging tool'lardan biri, matrix dimensionları bir kağıda yazmaktır.

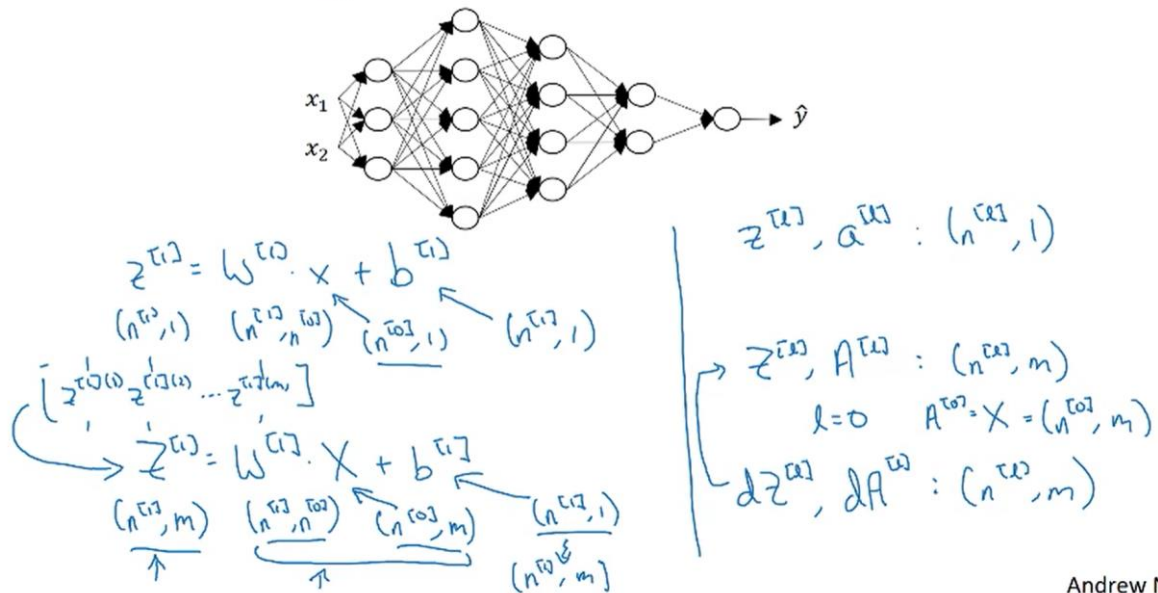
#### Parameters $W^{[l]}$ and $b^{[l]}$



Andrew Ng

Yukarıda sağ üstte görülenler beklenen matris boyutları, kodlama sırasında bunları bilmek yardımcı olur.

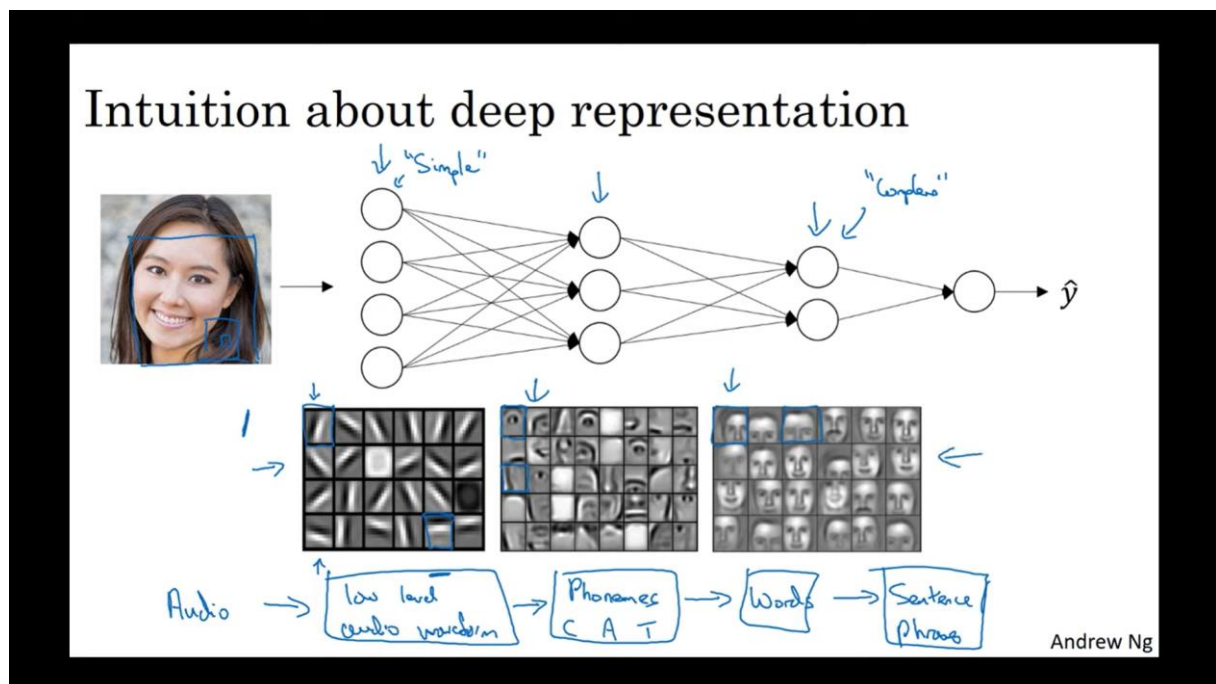
## Vectorized implementation



Benzer şekilde yine sağdaki kısımda beklenen matrix boyutlarını görebiliyoruz.

## Vid 40

### Why Deep Representations?



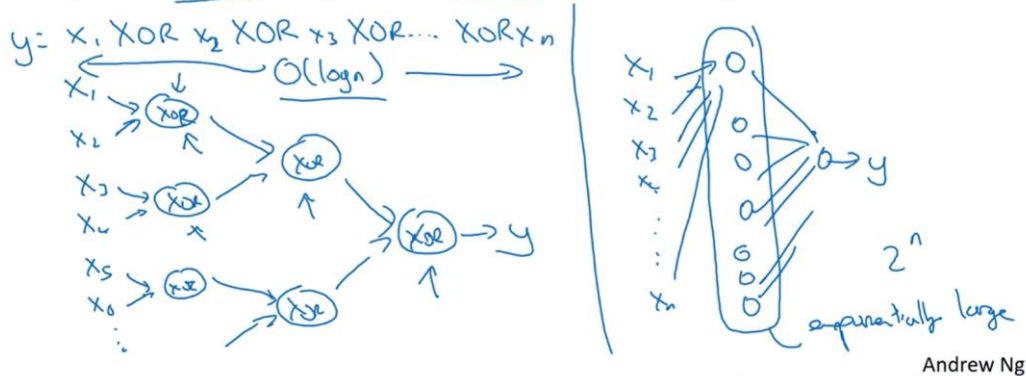


Özünde olay bildiğimiz gibi, NN derinleştikçe daha complex feature'lar ortaya çıkıyor.

Bir başka sebep ise aşağıdaki:

## 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.

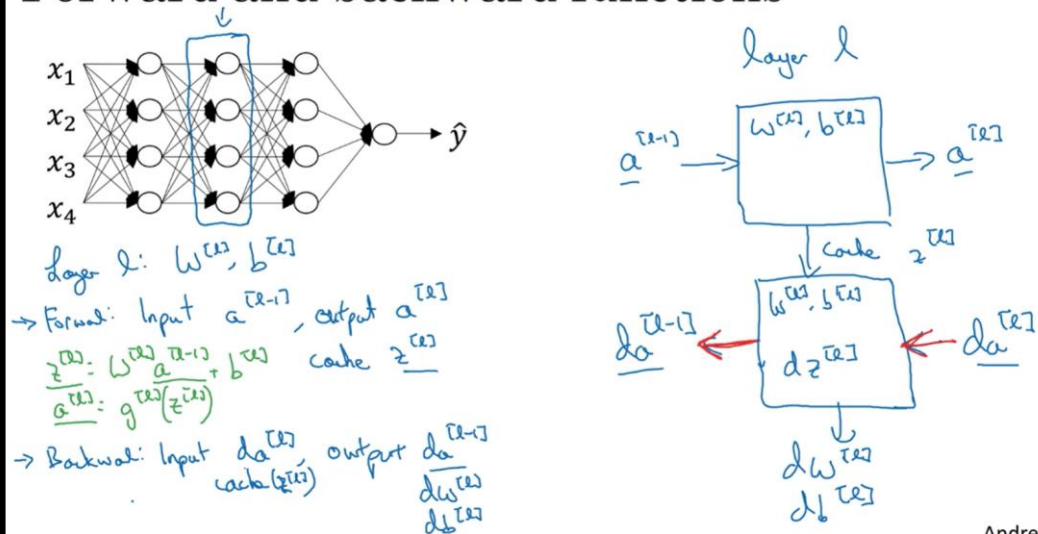


Aynı fonksiyon için shallow NN'de bir sürü unit gerekirken, deep NN için daha az unit ile işi halledebiliriz.

## Vid 41

### Building Blocks of a Deep Neural Network

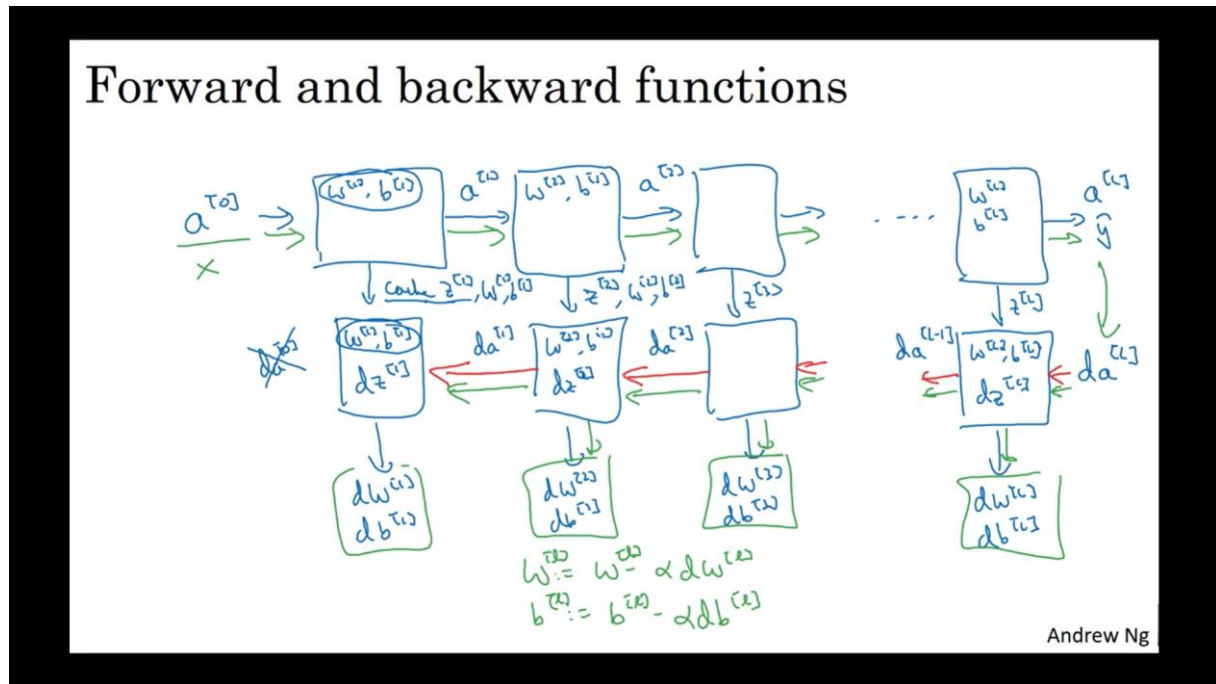
## Forward and backward functions



Seçilen layer için forward ve backward computations'ı tekrar gözden geçiriyoruz. Sonuçta forward için  $a_{l-1}$ 'den  $a_l$  hesaplanıyor ve bunun için  $w_l$  ve  $b_l$  bilgisini kullanıyoruz bu sırada backward computation için kullanılmak üzere  $z_l$ 'i cache'liyoruz.

Backward için  $da_l$ 'den  $da_{l-1}$ , hesaplıyoruz bu sırada  $w_l, b_l, dz_l$  kullanılıyor ve  $dw_l, db_l$  de output olarak verilebilir.

Sonuçta forward ve backward computation'ı her layer için hesapladığımızı düşünersek aşağıdaki diagram ortaya çıkar.



## Vid 42

### Parameters and Hyperparameters

**What are hyperparameters?**

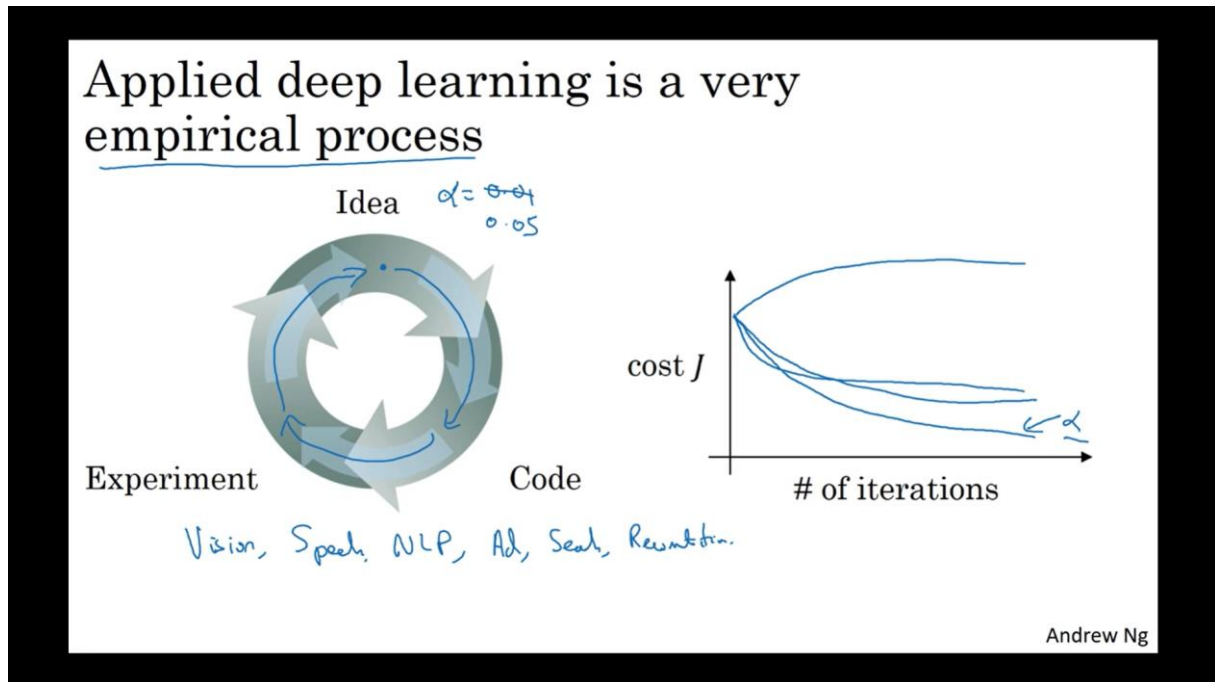
Parameters:  $w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, w^{[3]}, b^{[3]} \dots$

Hyperparameters:  $\left. \begin{array}{l} \text{learning rate } \frac{\alpha}{\tau} \\ \text{\#iterations} \\ \text{\#hidden layers } L \\ \text{\#hidden units } n^{[1]}, n^{[2]}, \dots \\ \text{choice of activation function} \end{array} \right\}$

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

Andrew Ng

Hyperparameters dediğimiz parametreler,  $W$  ve  $b$  gibi final parametreleri etkileyen diğer parametreler olarak düşünülebilir, learning rate'e veya iteration'a göre benim  $W$  ve  $b$  parametrelerim değişecektir. Daha sonra göreceğimiz, momentum, minibatch size, regularization parameters gibi başka hyperparameters da söz konusudur.



Sonuçta uygulama hala empirik yani deneysel gerçekleşiyor, en iyi hyperparameterları bulmanın kesin bir yolu yok, deneme yanılma ile buluruz, daha sonra bu deneme yanılma için de sistematik bazı yollardan bahsedilecek. Ancak bulunan parametre bile bir süre sonra değiştirilebilir, CPU'nun GPU'nun yapısından dolayı dahi sonuçlar değişebilir.

## Vid 43

What does this have to do with brain?

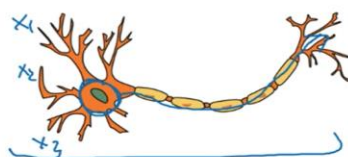
### Forward and backward propagation

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

"It's like the brain."



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



Andrew Ng

Sonuta dediđi Őey Őu, bir log. Reg. Unit neurana benzese de bu pek de nemsenecek bir benzerlik deđil, analoji kurmak dođru olmaz diyor. Tek bir neuronun bile henz tam olarak neler yaptığı aık deđil. Ayrıca beyinde backpropagation gibi bir mekanizma var mı yok mu belirsiz, nasıl đreniyor belirsiz.

Bunun yerine supervised learning'i son derece kompleks fonksiyonların đrenilmesi olarak dŐnmek daha mantıklı, en nihayetinde  $x$  ile  $y$ 'yi map ediyoruz.