

目录

1. Standard notations for Deep Learning	2
1 Neural Networks Notations.	2
2 Deep Learning representations	3
2. Binary_Classification	4
3. Logistic_Regression	5
4. Logistic_Regression_Cost_Function	6
C1_W1	7
C1_W2	25
C1_W3	71
C1_W4	107
C2_W1	130
C2_W2	170
C2_W3	202
C3_W1	234
C3_W2	274
C4_W1	309
C4_W2	351
C4_W3	402
C4_W4	449
C5_W1	502
C5_W2	545
C5_W3	582
C5_W4	624

Standard notations for Deep Learning

This document has the purpose of discussing a new standard for deep learning mathematical notations.

1 Neural Networks Notations.

General comments:

- superscript (i) will denote the i^{th} training example while superscript [l] will denote the l^{th} layer

Sizes:

· m : number of examples in the dataset

· n_x : input size

· n_y : output size (or number of classes)

· $n_h^{[l]}$: number of hidden units of the l^{th} layer

In a for loop, it is possible to denote $n_x = n_h^{[0]}$ and $n_y = n_h^{[\text{number of layers} + 1]}$.

· L : number of layers in the network.

Objects:

· $X \in \mathbb{R}^{n_x \times m}$ is the input matrix

· $x^{(i)} \in \mathbb{R}^{n_x}$ is the i^{th} example represented as a column vector

· $Y \in \mathbb{R}^{n_y \times m}$ is the label matrix

· $y^{(i)} \in \mathbb{R}^{n_y}$ is the output label for the i^{th} example

· $W^{[l]} \in \mathbb{R}^{\text{number of units in next layer} \times \text{number of units in the previous layer}}$ is the weight matrix, superscript [l] indicates the layer

· $b^{[l]} \in \mathbb{R}^{\text{number of units in next layer}}$ is the bias vector in the l^{th} layer

· $\hat{y} \in \mathbb{R}^{n_y}$ is the predicted output vector. It can also be denoted $a^{[L]}$ where L is the number of layers in the network.

Common forward propagation equation examples:

$a = g^{[l]}(W_x x^{(i)} + b_1) = g^{[l]}(z_1)$ where $g^{[l]}$ denotes the l^{th} layer activation function

$\hat{y}^{(i)} = \text{softmax}(W_h h + b_2)$

· General Activation Formula: $a_j^{[l]} = g^{[l]}(\sum_k w_{jk}^{[l]} a_k^{[l-1]} + b_j^{[l]}) = g^{[l]}(z_j^{[l]})$

· $J(x, W, b, y)$ or $J(\hat{y}, y)$ denote the cost function.

Examples of cost function:

· $J_{CE}(\hat{y}, y) = -\sum_{i=0}^m y^{(i)} \log \hat{y}^{(i)}$

· $J_1(\hat{y}, y) = \sum_{i=0}^m |y^{(i)} - \hat{y}^{(i)}|$

2 Deep Learning representations

For representations:

- nodes represent inputs, activations or outputs
- edges represent weights or biases

Here are several examples of Standard deep learning representations

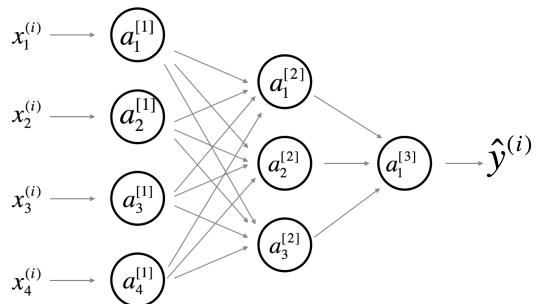


Figure 1: Comprehensive Network: representation commonly used for Neural Networks. For better aesthetic, we omitted the details on the parameters ($w_{ij}^{[l]}$ and $b_i^{[l]}$ etc...) that should appear on the edges

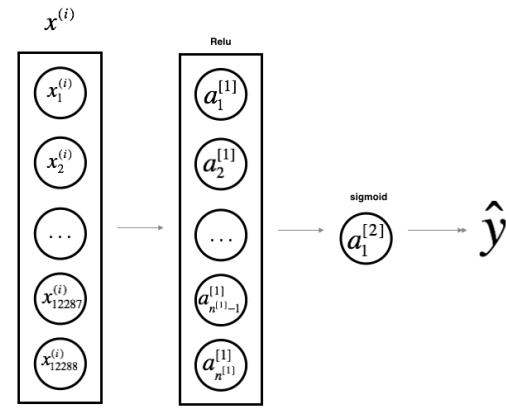


Figure 2: Simplified Network: a simpler representation of a two layer neural network, both are equivalent.

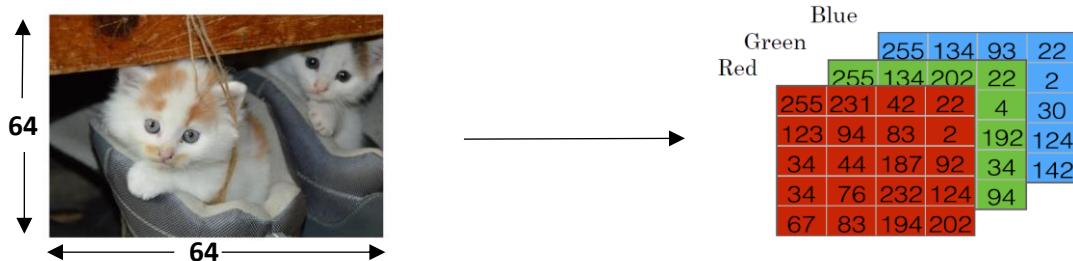
Binary Classification

In a binary classification problem, the result is a discrete value output.

- For example
- account hacked (1) or compromised (0)
 - a tumor malign (1) or benign (0)

Example: Cat vs Non-Cat

The goal is to train a classifier that the input is an image represented by a feature vector, x , and predicts whether the corresponding label y is 1 or 0. In this case, whether this is a cat image (1) or a non-cat image (0).



An image is stored in the computer in three separate matrices corresponding to the Red, Green, and Blue color channels of the image. The three matrices have the same size as the image, for example, the resolution of the cat image is 64 pixels X 64 pixels, the three matrices (RGB) are 64 X 64 each.

The value in a cell represents the pixel intensity which will be used to create a feature vector of n-dimension. In pattern recognition and machine learning, a feature vector represents an object, in this case, a cat or no cat.

To create a feature vector, x , the pixel intensity values will be “unroll” or “reshape” for each color. The dimension of the input feature vector x is $n_x = 64 \times 64 \times 3 = 12,288$.

$$x = \begin{bmatrix} 255 \\ 231 \\ 42 \\ \vdots \\ 255 \\ 134 \\ 202 \\ \vdots \\ 255 \\ 134 \\ 93 \\ \vdots \end{bmatrix} \leftarrow \begin{array}{l} \text{red} \\ \text{green} \\ \text{blue} \end{array}$$

Logistic Regression

Logistic regression is a learning algorithm used in a supervised learning problem when the output y are all either zero or one. The goal of logistic regression is to minimize the error between its predictions and training data.

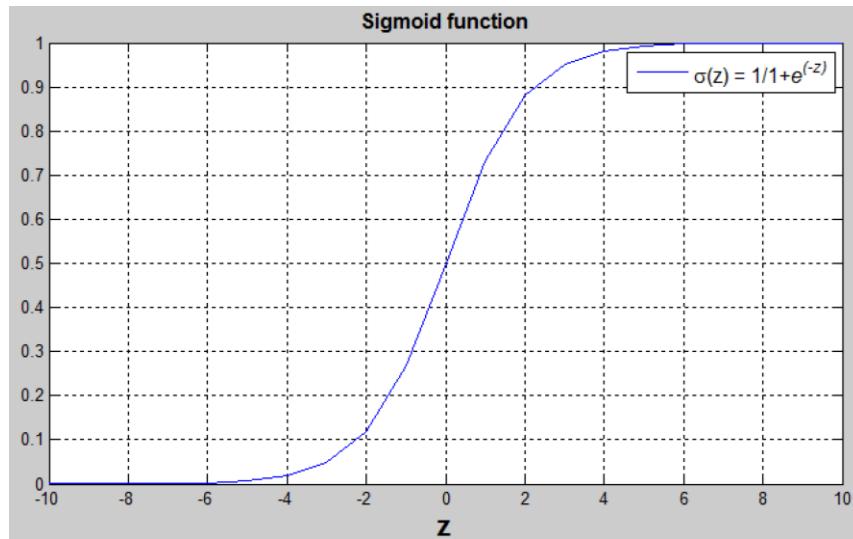
Example: Cat vs No - cat

Given an image represented by a feature vector x , the algorithm will evaluate the probability of a cat being in that image.

$$\text{Given } x, \hat{y} = P(y = 1|x), \text{ where } 0 \leq \hat{y} \leq 1$$

The parameters used in Logistic regression are:

- The input features vector: $x \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The training label: $y \in \{0,1\}$
- The weights: $w \in \mathbb{R}^{n_x}$, where n_x is the number of features
- The threshold: $b \in \mathbb{R}$
- The output: $\hat{y} = \sigma(w^T x + b)$
- Sigmoid function: $s = \sigma(w^T x + b) = \sigma(z) = \frac{1}{1+e^{-z}}$



$(w^T x + b)$ is a linear function ($ax + b$), but since we are looking for a probability constraint between $[0,1]$, the sigmoid function is used. The function is bounded between $[0,1]$ as shown in the graph above.

Some observations from the graph:

- If z is a large positive number, then $\sigma(z) = 1$
- If z is small or large negative number, then $\sigma(z) = 0$
- If $z = 0$, then $\sigma(z) = 0.5$

Logistic Regression: Cost Function

To train the parameters w and b , we need to define a cost function.

Recap:

$$\hat{y}^{(i)} = \sigma(w^T x^{(i)} + b), \text{ where } \sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}}$$

$x^{(i)}$ the i-th training example

Given $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$, we want $\hat{y}^{(i)} \approx y^{(i)}$

Loss (error) function:

The loss function measures the discrepancy between the prediction ($\hat{y}^{(i)}$) and the desired output ($y^{(i)}$). In other words, the loss function computes the error for a single training example.

$$L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{2}(\hat{y}^{(i)} - y^{(i)})^2$$

$$L(\hat{y}^{(i)}, y^{(i)}) = -(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))$$

- If $y^{(i)} = 1$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(\hat{y}^{(i)})$ where $\log(\hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 1
- If $y^{(i)} = 0$: $L(\hat{y}^{(i)}, y^{(i)}) = -\log(1 - \hat{y}^{(i)})$ where $\log(1 - \hat{y}^{(i)})$ and $\hat{y}^{(i)}$ should be close to 0

Cost function

The cost function is the average of the loss function of the entire training set. We are going to find the parameters w and b that minimize the overall cost function.

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m [-(y^{(i)} \log(\hat{y}^{(i)}) + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)}))]$$

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

Introduction to Deep Learning

Welcome



- AI is the new Electricity
- Electricity had once transformed countless industries: transportation, manufacturing, healthcare, communications, and more
- AI will now bring about an equally big transformation.

What you'll learn



Courses in this sequence (Specialization):

1. Neural Networks and Deep Learning
2. Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
3. Structuring your Machine Learning project
4. Convolutional Neural Networks CNN
5. Natural Language Processing: Building sequence models

train / dev / test
end-to-end

RNN, LSTM

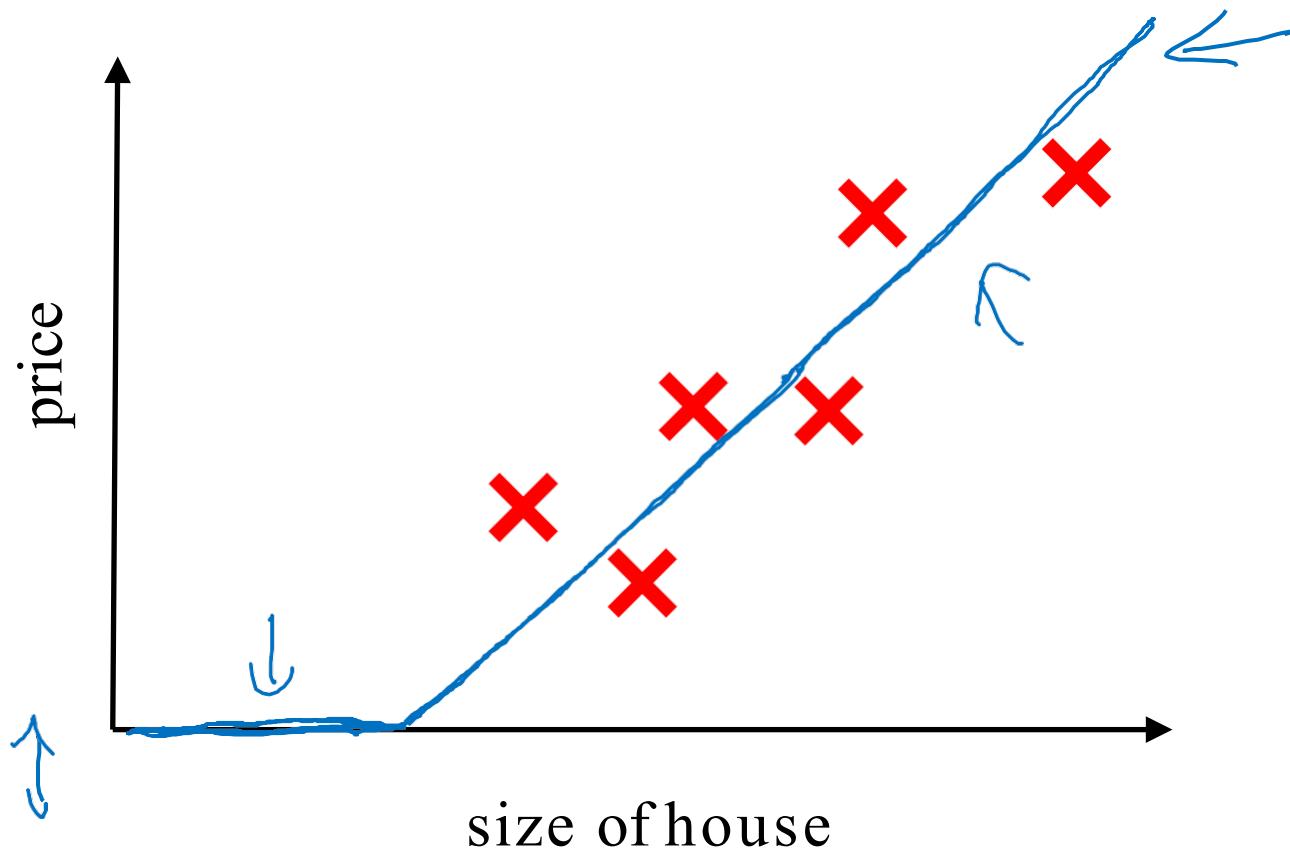


deeplearning.ai

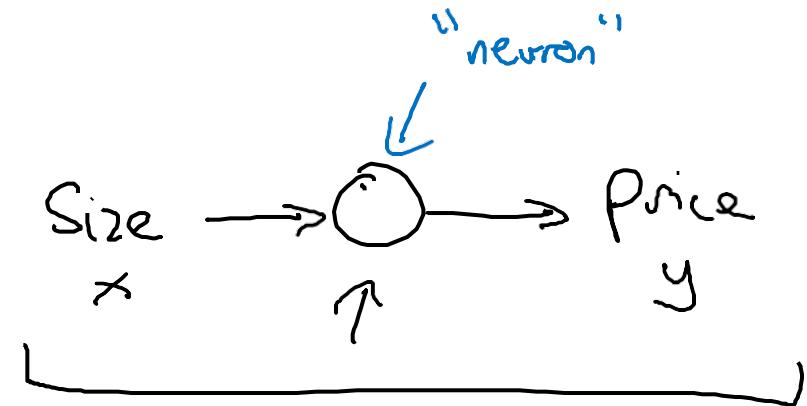
Introduction to Deep Learning

What is a Neural Network?

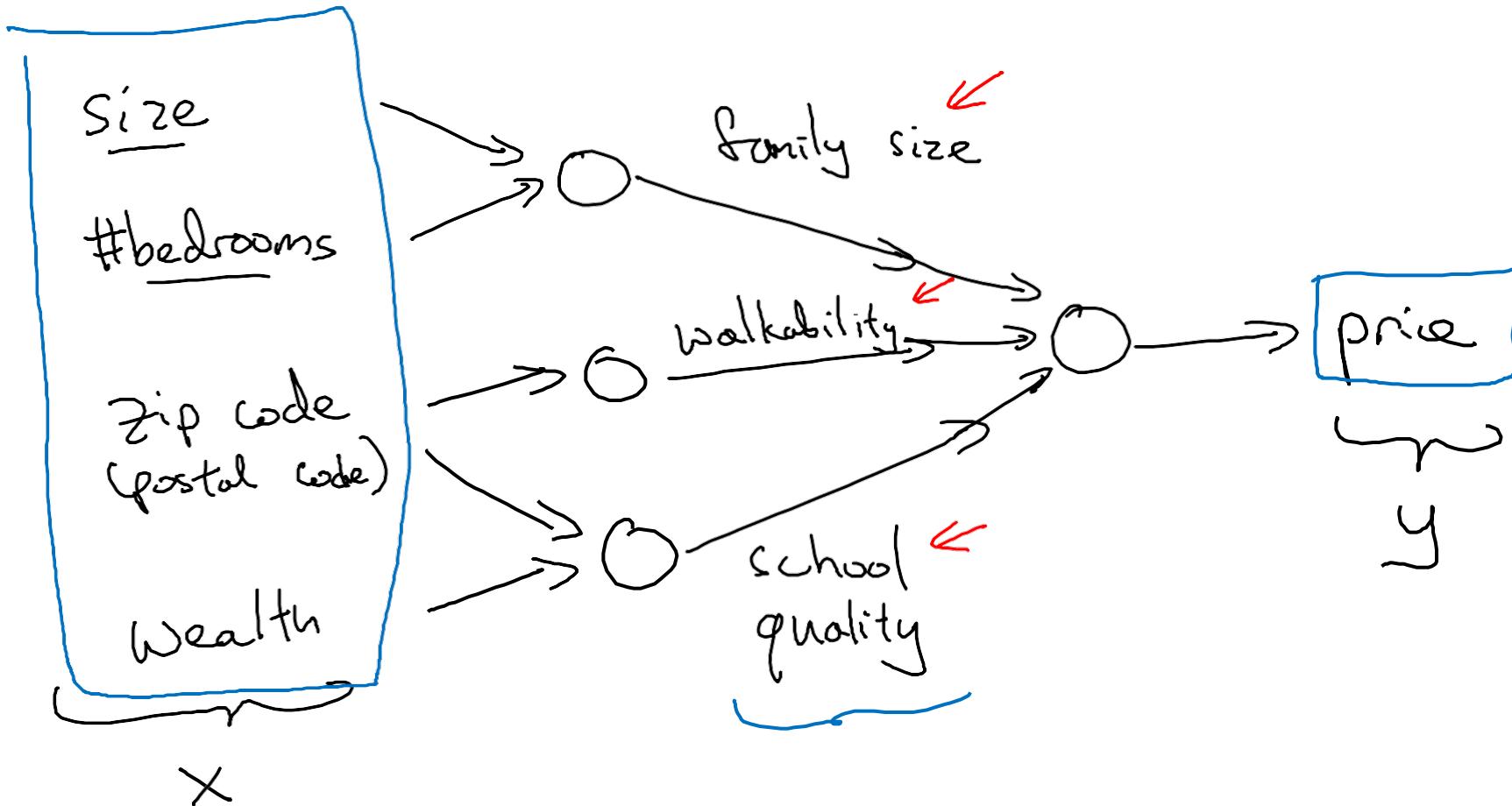
Housing Price Prediction



ReLU
Rectified
Linear
Unit

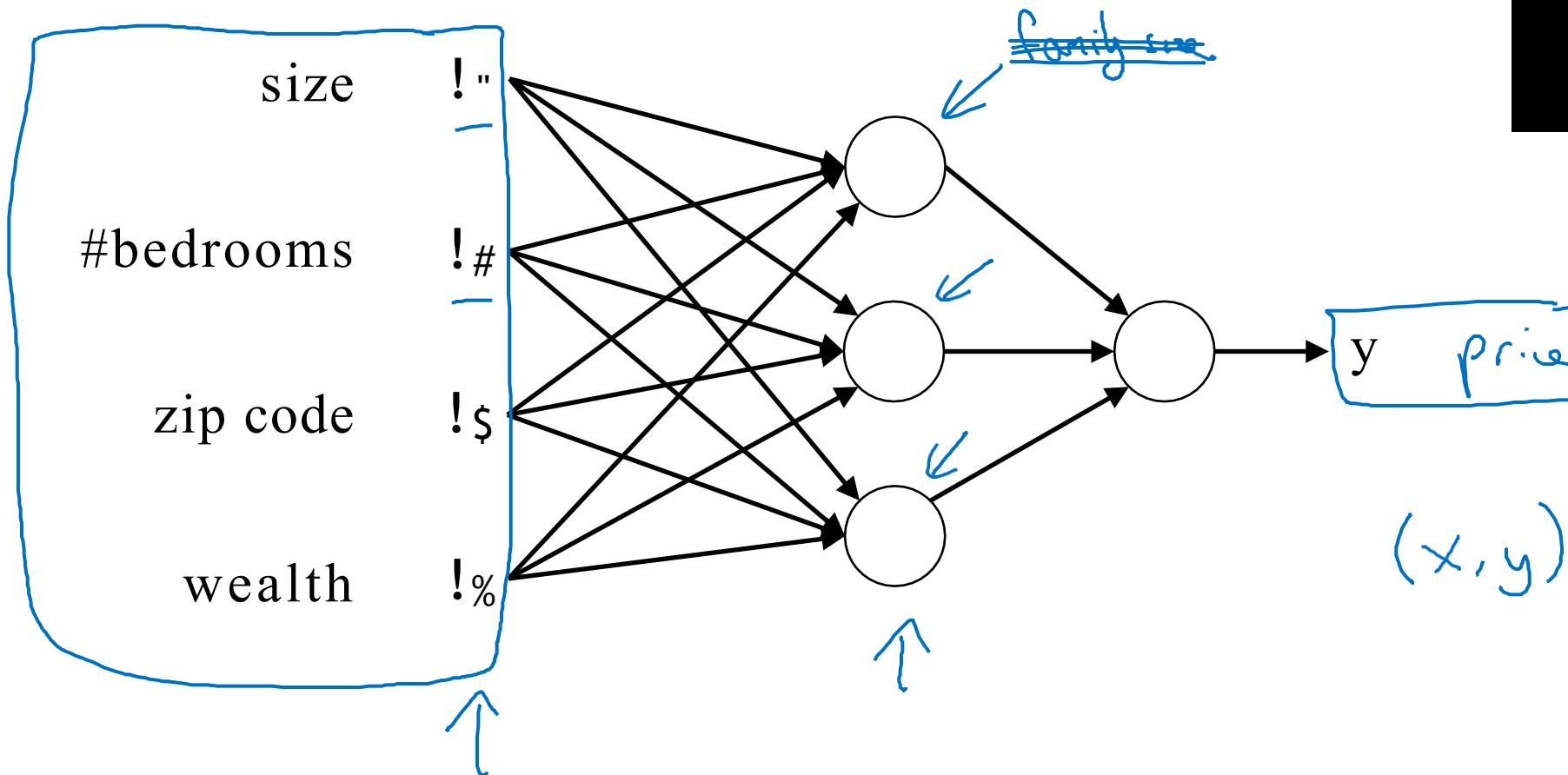


Housing Price Prediction



Housing Price Prediction

Drawing of
previous Image





deeplearning.ai

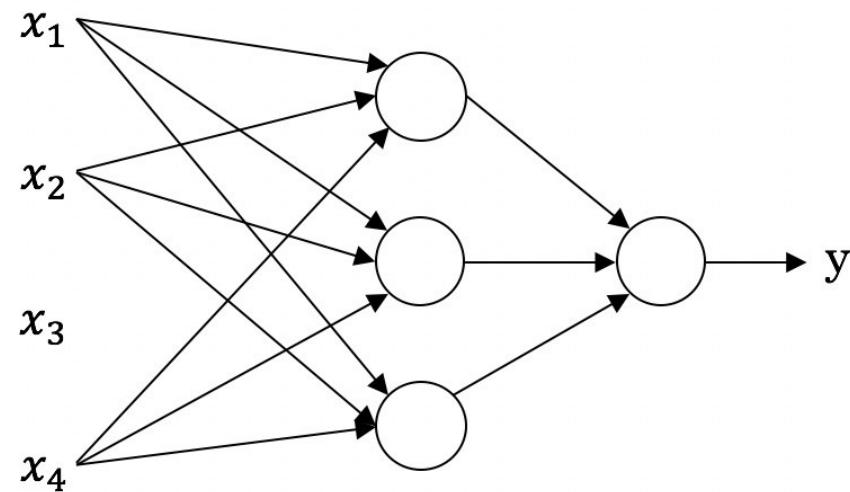
Introduction to Deep Learning

Supervised Learning with Neural Networks

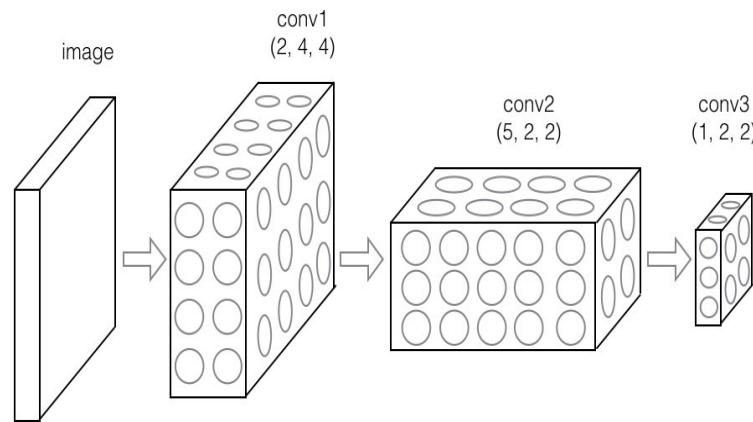
Supervised Learning

Input(x)	Output (y)	Application
Home features	Price	Real Estate
Ad, user info	Click on ad? (0/1)	Online Advertising
Image	Object (1,...,1000)	Photo tagging
Audio	Text transcript	Speech recognition
English	Chinese	Machine translation
Image, Radar info	Position of other cars	Autonomous driving

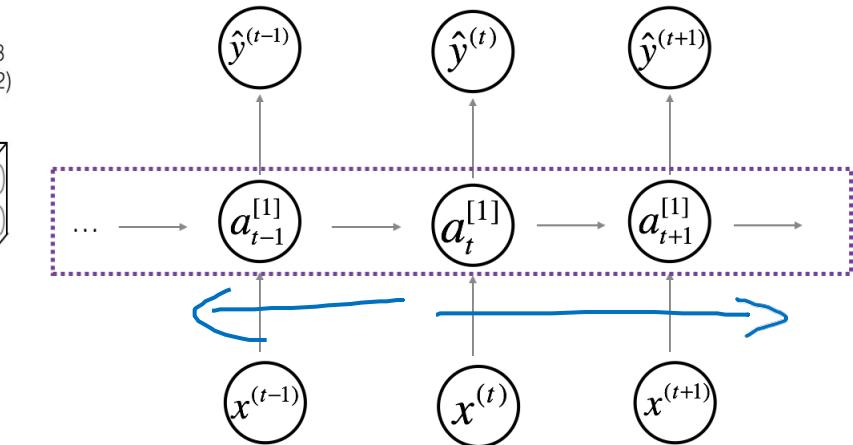
Neural Network examples



Standard NN



Convolutional NN



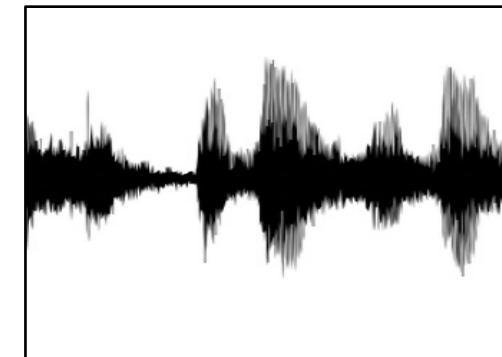
Recurrent NN

Supervised Learning

Structured Data

Size	#bedrooms	...	Price (1000\$)
2104	3		400
1600	3		330
2400	3		369
...
3000	4		540

Unstructured Data



Audio

Image

User Age	Ad Id	...	Click
41	93242		1
80	93287		0
18	87312		1
...
27	71244		1

Four scores and seven years ago...

Text

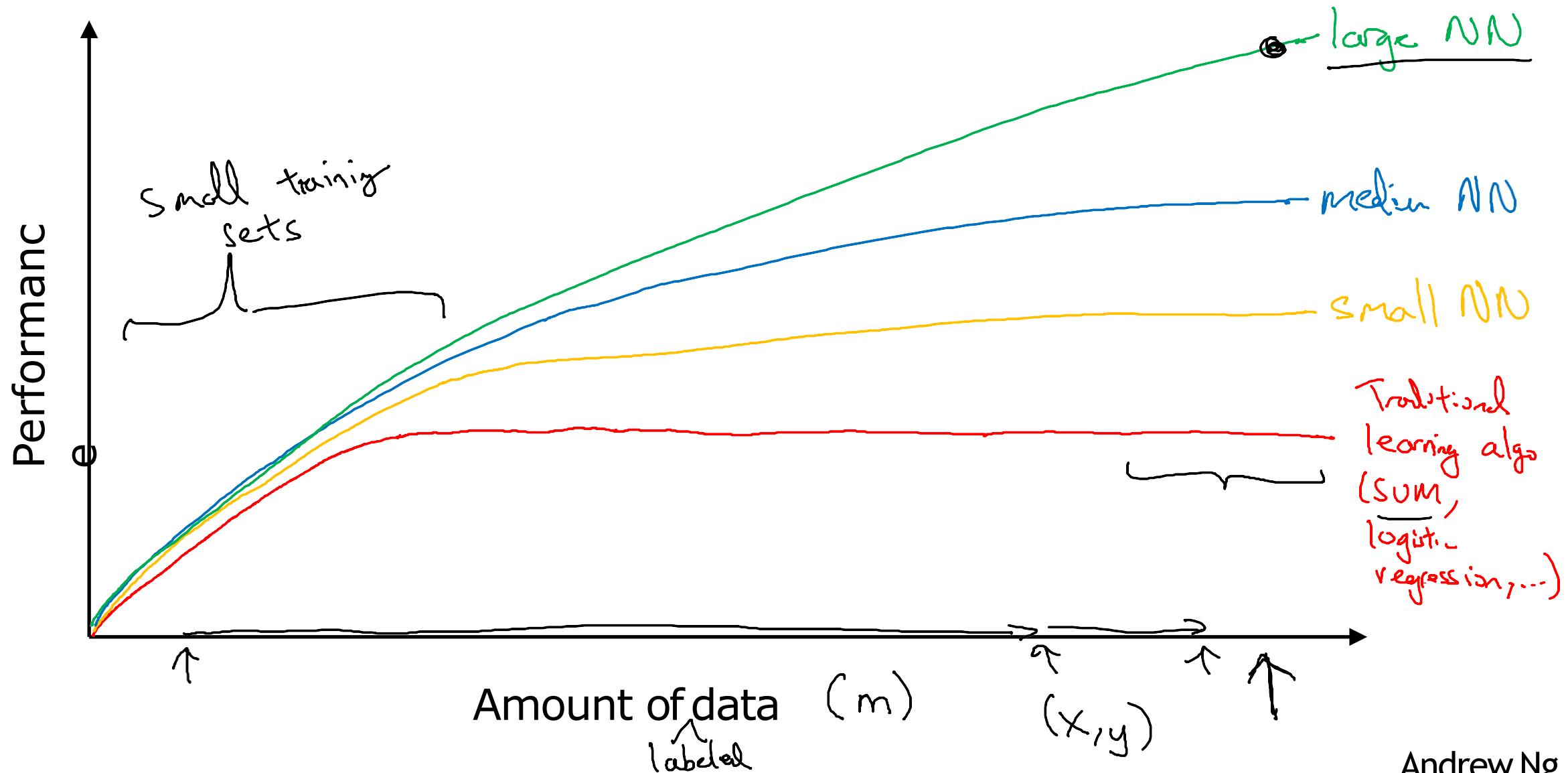


deeplearning.ai

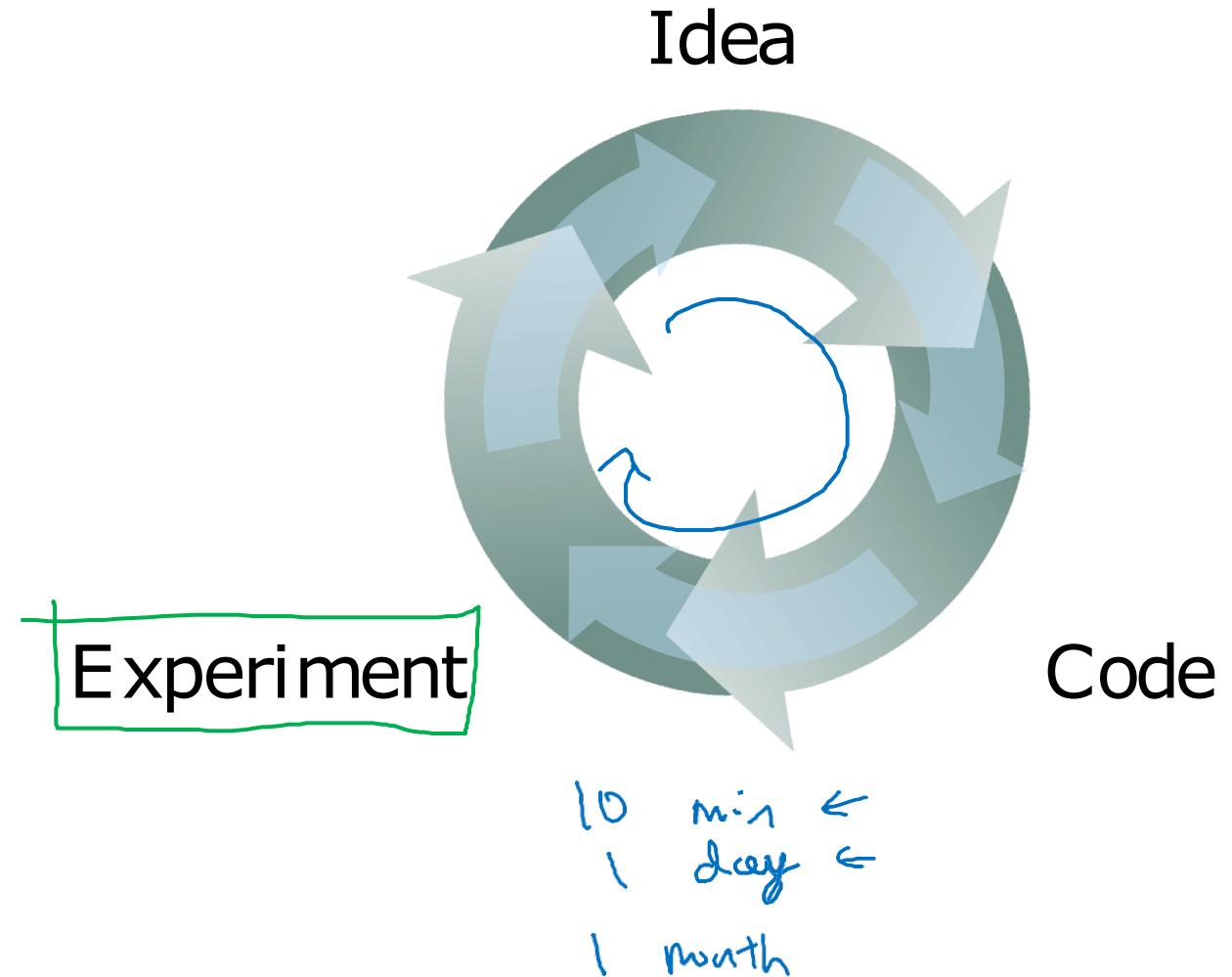
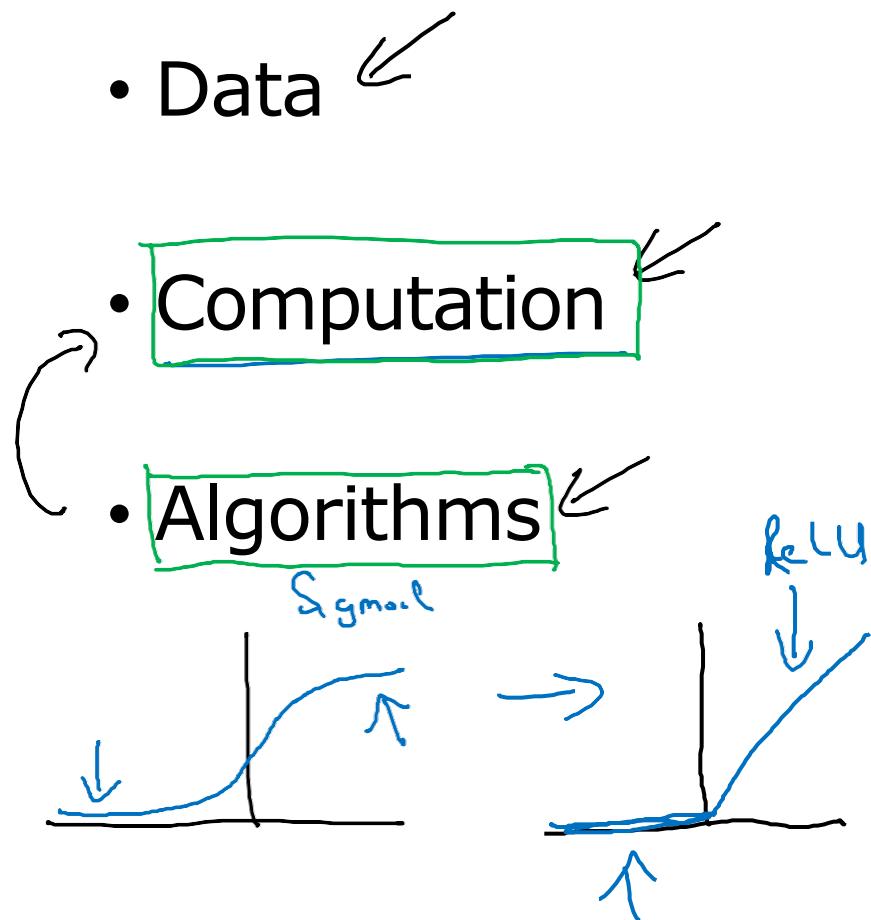
Introduction to Neural Networks

Why is Deep Learning taking off?

Scale drives deep learning progress



Scale drives deep learning progress





deeplearning.ai

Introduction to Neural Networks

About this Course

Courses in this Specialization

1. Neural Networks and DeepLearning ←
2. Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
3. Structuring your Machine Learning project
4. Convolutional Neural Networks
5. Natural Language Processing: Building sequence models

Outline of this Course

Week 1: Introduction

Week 2: Basics of Neural Network programming

Week 3: One hidden layer Neural Networks

Week 4: Deep Neural Networks

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>



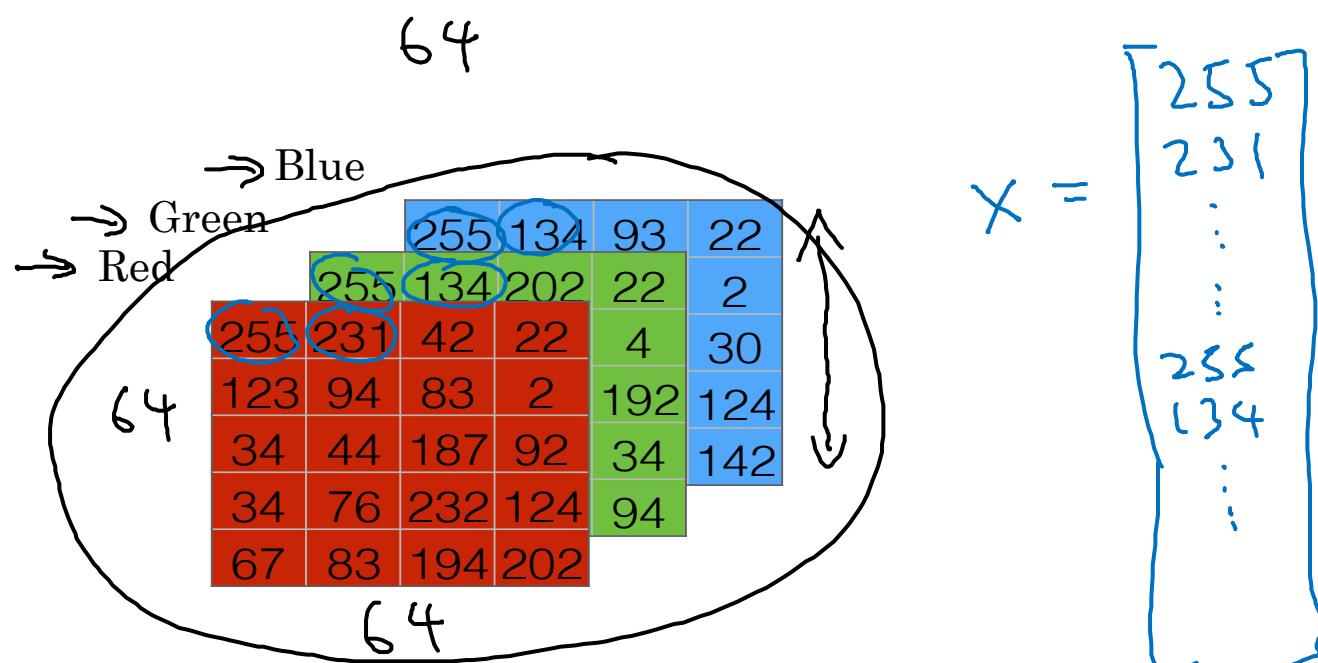
deeplearning.ai

Basics of Neural Network Programming

Binary Classification

Binary Classification

64 →  → 1 (cat) vs 0 (non cat)



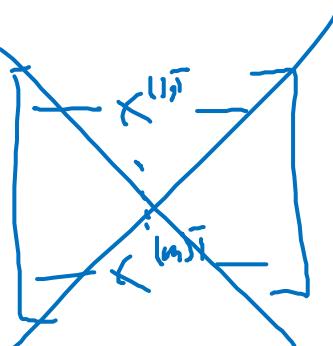
$$64 \times 64 \times 3 = 12288$$

$$n = n_x = 12288$$

$$X \rightarrow y$$

Notation

(x, y) $x \in \mathbb{R}^{n_x}$, $y \in \{0, 1\}$
 m training examples : $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$
 $M = M_{\text{train}}$ $M_{\text{test}} = \# \text{test examples.}$

$$X = \begin{bmatrix} | & | & & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & & | \end{bmatrix}^{n_x \times m}$$

$$Y = [y^{(1)} \ y^{(2)} \ \dots \ y^{(m)}]$$
$$Y \in \mathbb{R}^{1 \times m}$$
$$Y.\text{shape} = (1, m)$$

$X \in \mathbb{R}^{n_x \times m}$ $X.\text{shape} = (n_x, m)$



deeplearning.ai

Basics of Neural Network Programming

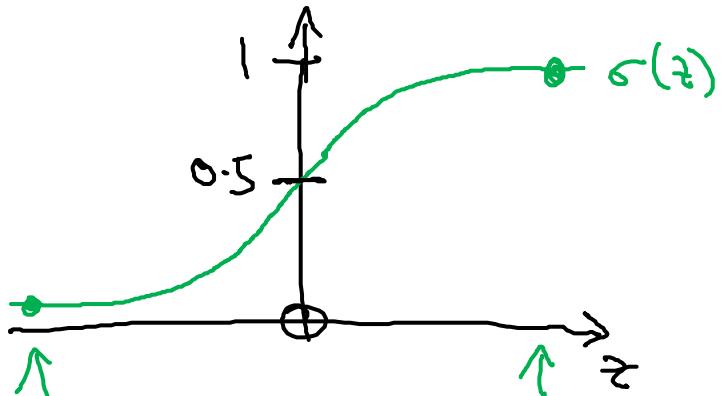
Logistic Regression

Logistic Regression

Given x , want $\hat{y} = P(y=1 | x)$
 $x \in \mathbb{R}^{n_x}$

Parameters: $w \in \mathbb{R}^{n_x}$, $b \in \mathbb{R}$.

Output $\hat{y} = \sigma(w^T x + b)$



$$x_0 = 1, \quad x \in \mathbb{R}^{n_x+1}$$

$$\hat{y} = \sigma(w^T x)$$

$$w = \begin{bmatrix} w_0 \\ w_1 \\ w_2 \\ \vdots \\ w_{n_x} \end{bmatrix} \quad \left. \begin{array}{l} \{w_0\} \rightarrow b \\ \{w_1, w_2, \dots, w_{n_x}\} \rightarrow w \end{array} \right.$$

$$\sigma(z) = \frac{1}{1 + e^{-z}}$$

If z large $\sigma(z) \approx \frac{1}{1+0} = 1$

If z large negative number

$$\sigma(z) = \frac{1}{1 + e^{-z}} \approx \frac{1}{1 + \text{BigNum}} \approx 0$$



deeplearning.ai

Basics of Neural Network Programming

Logistic Regression cost function

Logistic Regression cost function

$$\rightarrow \hat{y}^{(i)} = \sigma(w^T \underline{x}^{(i)} + b), \text{ where } \sigma(z^{(i)}) = \frac{1}{1+e^{-z^{(i)}}} \quad z^{(i)} = w^T \underline{x}^{(i)} + b$$

Given $\{(x^{(1)}, y^{(1)}), \dots, (x^{(m)}, y^{(m)})\}$, want $\hat{y}^{(i)} \approx y^{(i)}$.

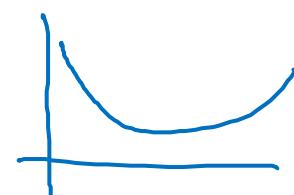
Loss (error) function:

$$L(\hat{y}, y) = \frac{1}{2} (\hat{y} - y)^2$$

$$L(\hat{y}, y) = - (y \log \hat{y} + (1-y) \log (1-\hat{y})) \leftarrow$$

$x^{(i)}$
 $y^{(i)}$
 $z^{(i)}$

i -th example.



If $y=1$: $L(\hat{y}, y) = -\log \hat{y} \leftarrow$ want $\log \hat{y}$ large, want \hat{y} large.

If $y=0$: $L(\hat{y}, y) = -\log (1-\hat{y}) \leftarrow$ want $\log (1-\hat{y})$ large ... want \hat{y} small

Cost function: $J(w, b) = \frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)}) = \frac{1}{m} \sum_{i=1}^m [y^{(i)} \log \hat{y}^{(i)} + (1-y^{(i)}) \log (1-\hat{y}^{(i)})]$



deeplearning.ai

Basics of Neural Network Programming

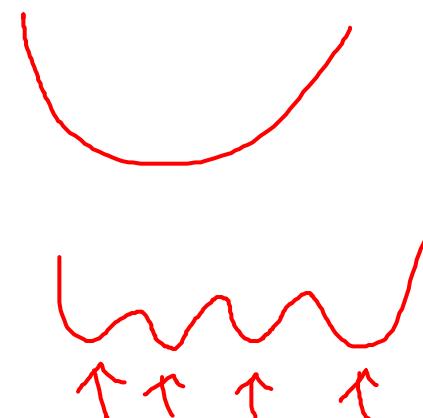
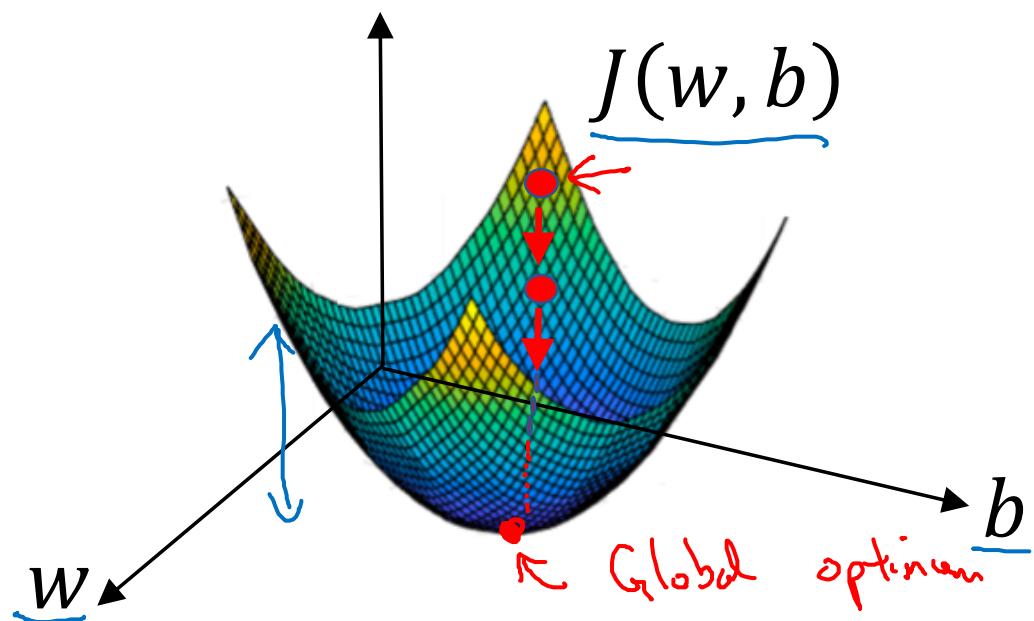
Gradient Descent

Gradient Descent

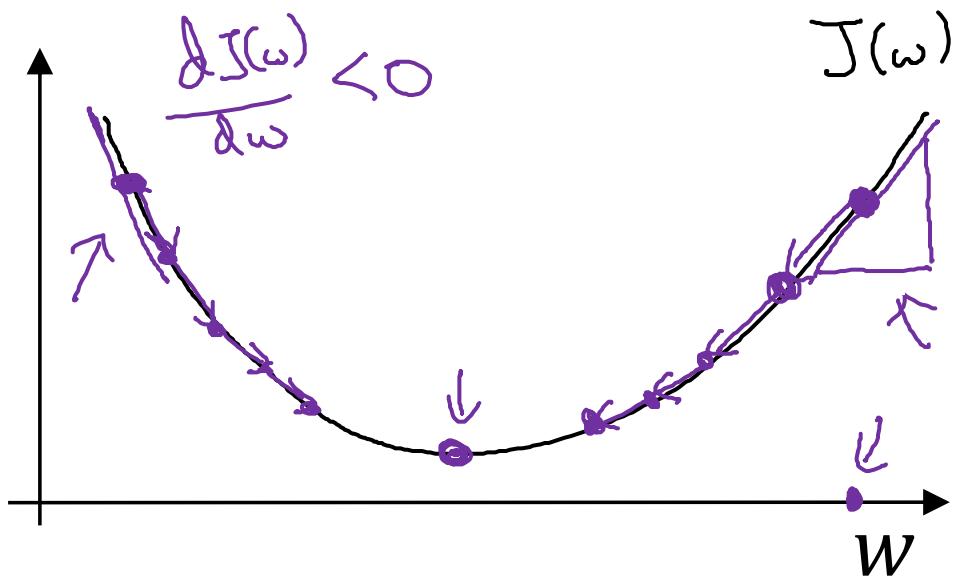
Recap: $\hat{y} = \sigma(w^T x + b)$, $\sigma(z) = \frac{1}{1+e^{-z}}$ 

$$\underline{J(w, b)} = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) = -\frac{1}{m} \sum_{i=1}^m y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})$$

Want to find w, b that minimize $J(w, b)$



Gradient Descent



Repeat {

$$w := w - \alpha \frac{dJ(w)}{dw}$$

}

$w := w - \alpha \frac{dJ(w)}{dw} = ?$

learning rate

"dw"

$$J(w, b)$$

$$w := w - \alpha \frac{dJ(w, b)}{dw}$$

$$b := b - \alpha \frac{dJ(w, b)}{db}$$

$$\frac{dJ(w, b)}{dw}$$

$$\frac{\partial J(w, b)}{\partial w}$$

$$\frac{\partial J(w, b)}{\partial b}$$

$$\frac{\partial}{\partial w}$$

$$\frac{\partial}{\partial b}$$

"partial derivative"
 J

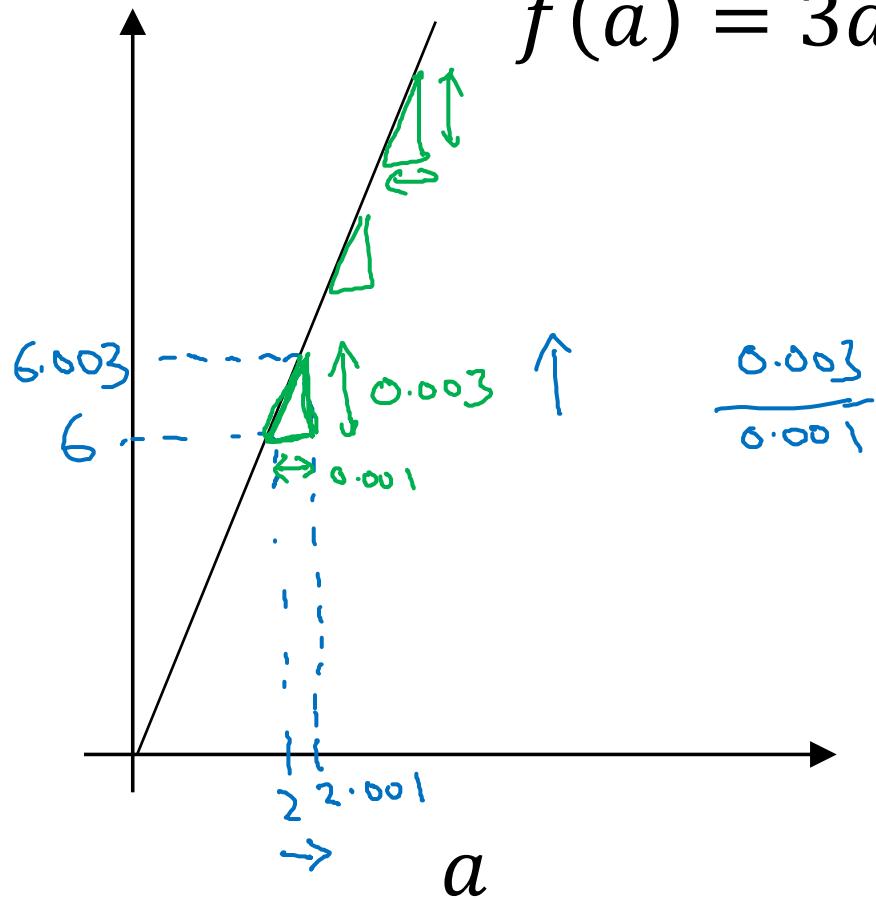


deeplearning.ai

Basics of Neural Network Programming

Derivatives

Intuition about derivatives



$$f(a) = 3a$$

$$\rightarrow a = 2$$

$$f(a) = 6$$

$$a = 2.001$$

$$f(a) = 6.003$$

height
width

slope (derivative) of $f(a)$

at $a=2$ is 3

$$\rightarrow a = 5$$

$$f(a) = 15$$

$$a = 5.001$$

$$f(a) = 15.003$$

slope at $a=5$ is also 3

$$\frac{d f(a)}{da} = 3 = \frac{d}{da} f(a)$$

0.001 ←
0.00000001
0.0000000001

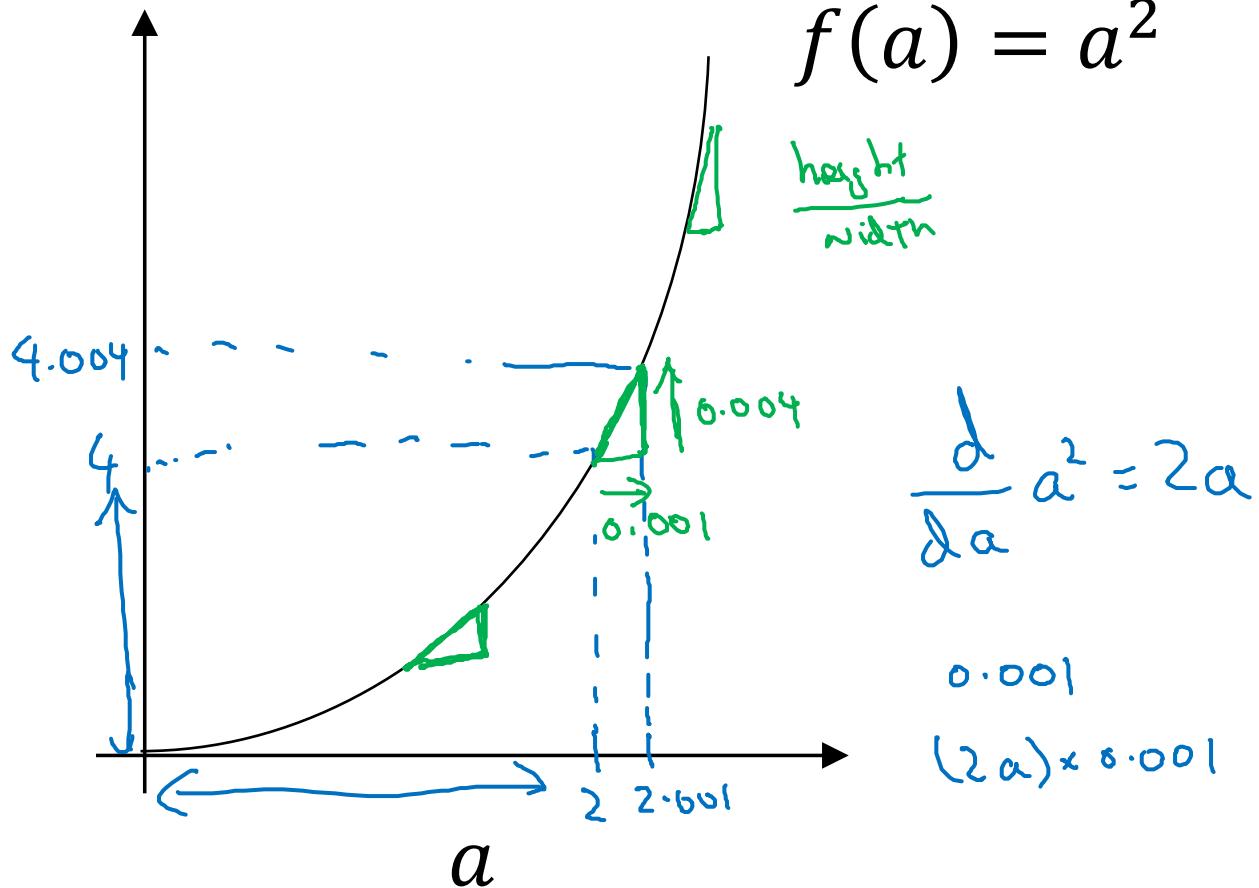


deeplearning.ai

Basics of Neural Network Programming

More derivatives
examples

Intuition about derivatives



$a = 2$ $f(a) = 4$

$a = 2.001$ $f(a) \approx 4.004$

$\frac{(4.004 - 4)}{0.001}$

slope (derivative) of $f(a)$ at $a=2$ is 4.

$\frac{d}{da} f(a) = 4$ when $a=2$.

$a = 5$ $f(a) = 25$

$a = 5.001$ $f(a) \approx 25.010$

$\frac{d}{da} f(a) = 10$ when $a=5$

$\frac{d}{da} f(a) = \frac{d}{da} a^2 = 2a$

More derivative examples

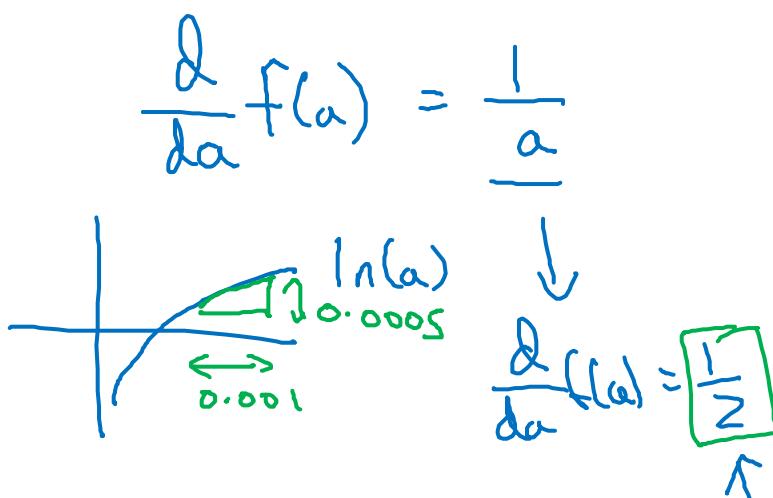
$$f(a) = a^2$$

$$\frac{\partial}{\partial a} f(a) = \underbrace{2a}_{4}$$

$$f(a) = a^3$$

$$\frac{\partial}{\partial a} f(a) = \underbrace{3a^2}_{3 \times 2^2} = 12$$

$$f(a) = \begin{matrix} \log_e(a) \\ \ln(a) \end{matrix}$$



$$a = 2$$

$$a = 2.001$$

$$f(a) = 4$$

$$f(a) \approx 4.004$$

$$a = 2$$

$$a = \underline{2.001}$$

$$f(a) = 8$$

$$f(a) \approx \underline{8.012}$$

$$a = 2$$

$$a = \underline{2.001}$$

$$f(a) \approx 0.69315$$

$$f(a) \approx \underline{0.69365}$$

$$\frac{0.0005}{0.0005}$$



deeplearning.ai

Basics of Neural Network Programming

Computation Graph

Computation Graph

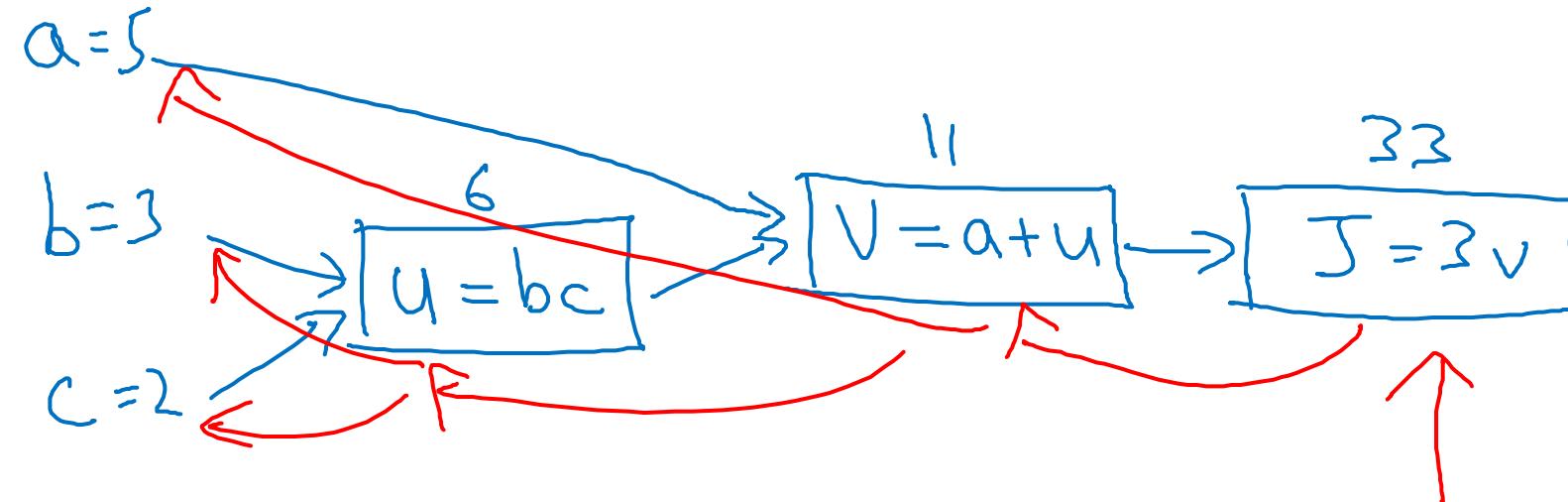
$$J(a, b, c) = 3(u + bc) = 3(5 + 3 \times 2) = 33$$

$\underbrace{u}_{\downarrow}$
 $\underbrace{v}_{\downarrow}$
 $\underbrace{J}_{\downarrow}$

$$u = bc$$

$$v = a + u$$

$$J = 3v$$



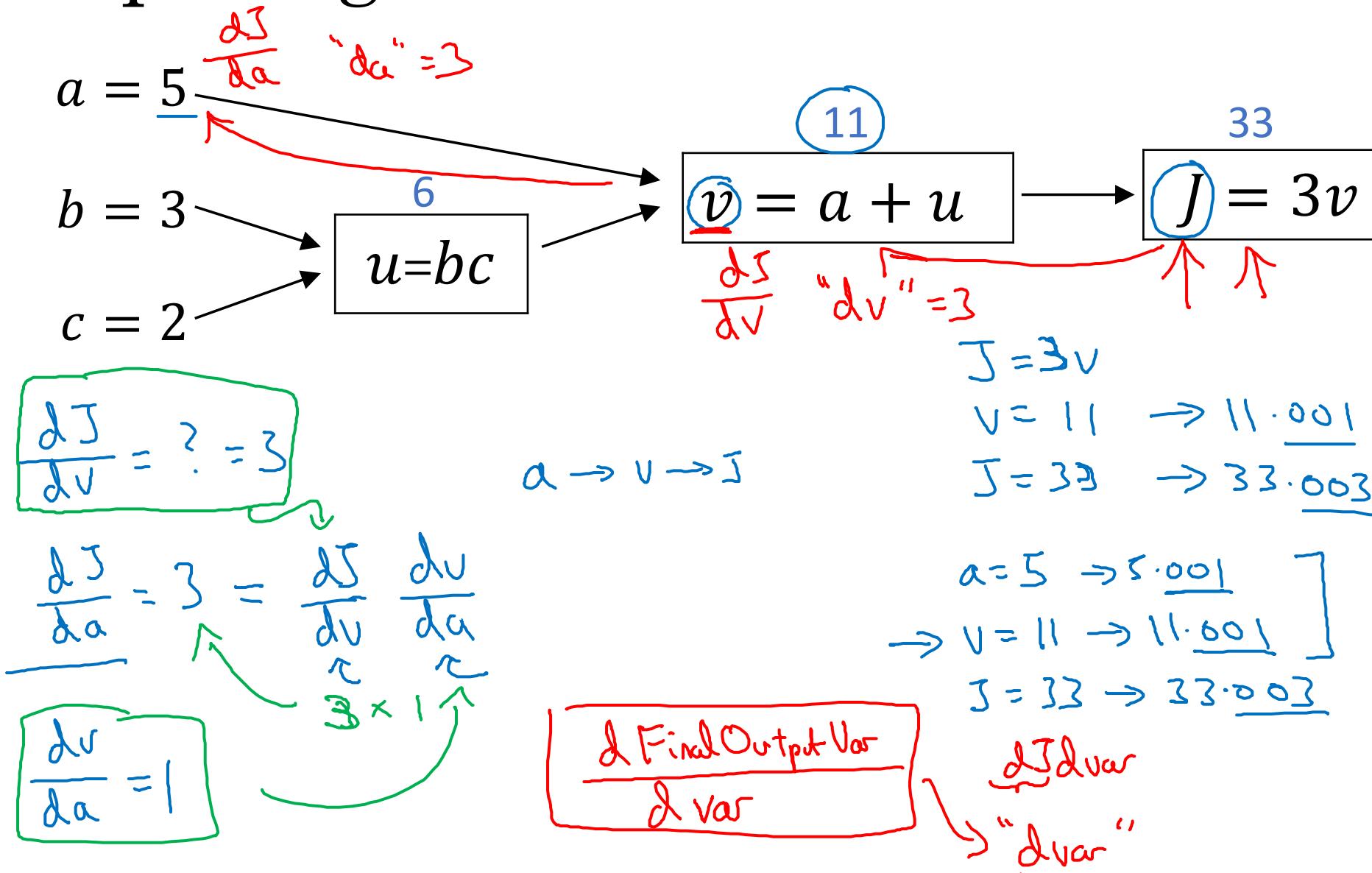


deeplearning.ai

Basics of Neural Network Programming

Derivatives with a Computation Graph

Computing derivatives



$f(a) = 3a$
 $\frac{df(b)}{da} = \frac{df}{da} = 3$
 $J = 3v$
 $\frac{dJ}{dv} = 3$

Computing derivatives

$\frac{\partial J}{\partial a} \rightarrow \underline{a = 5}$
 $\frac{\partial J}{\partial b} \rightarrow \underline{b = 3}$
 $\frac{\partial J}{\partial c} \rightarrow \underline{c = 2}$
 $\frac{\partial J}{\partial u} = 3$
 $\frac{\partial J}{\partial v} = 3$
 $\frac{\partial J}{\partial J} = 33$

$a = 5 \rightarrow \frac{\partial a}{\partial a} = 3$
 $b = 3 \rightarrow \frac{\partial b}{\partial b} = 6$
 $c = 2 \rightarrow \frac{\partial c}{\partial c} = 9$
 $u = bc \rightarrow \frac{\partial u}{\partial u} = 3$
 $v = a + u \rightarrow \frac{\partial v}{\partial v} = 3$
 $J = 3v \rightarrow \frac{\partial J}{\partial J} = 33$

$u = 6 \rightarrow 6.001$
 $v = 11 \rightarrow 11.001$
 $J = 33 \rightarrow 33.003$

$\frac{\partial J}{\partial b} = \boxed{\frac{\partial J}{\partial u}} \cdot \underbrace{\frac{\partial u}{\partial b}}_{=2} = 6$
 $\frac{\partial J}{\partial a} = \boxed{\frac{\partial J}{\partial u}} \cdot \underbrace{\frac{\partial u}{\partial a}}_{=3} = 9$

$b = 3 \rightarrow 3.001$
 $u = b \cdot c = 6 \rightarrow 6.002$
 $J = 33.006$
 $v = 11.002$
 $J = 3v$



deeplearning.ai

Basics of Neural Network Programming

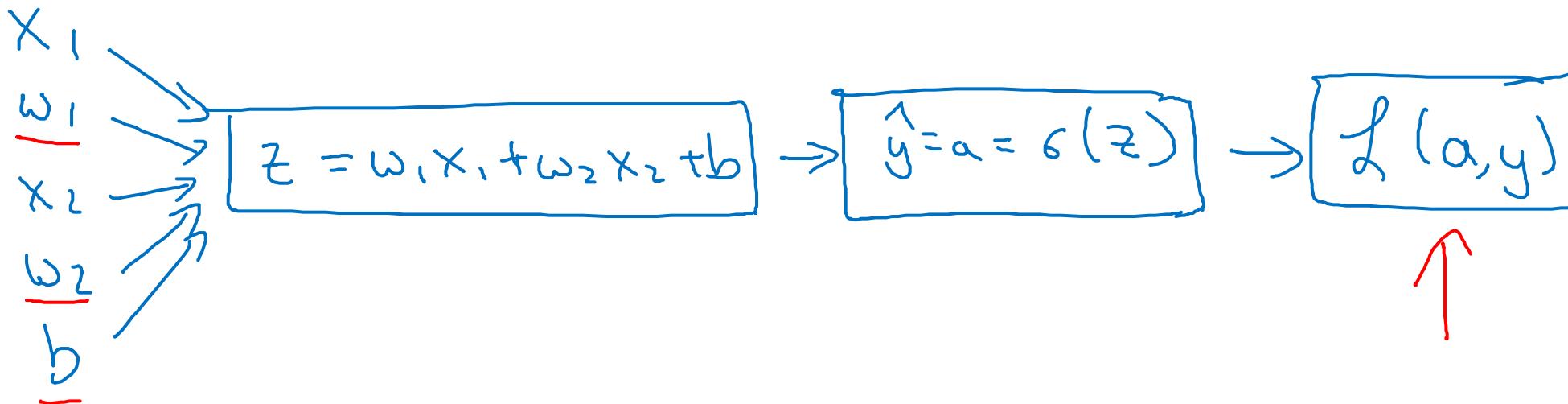
Logistic Regression Gradient descent

Logistic regression recap

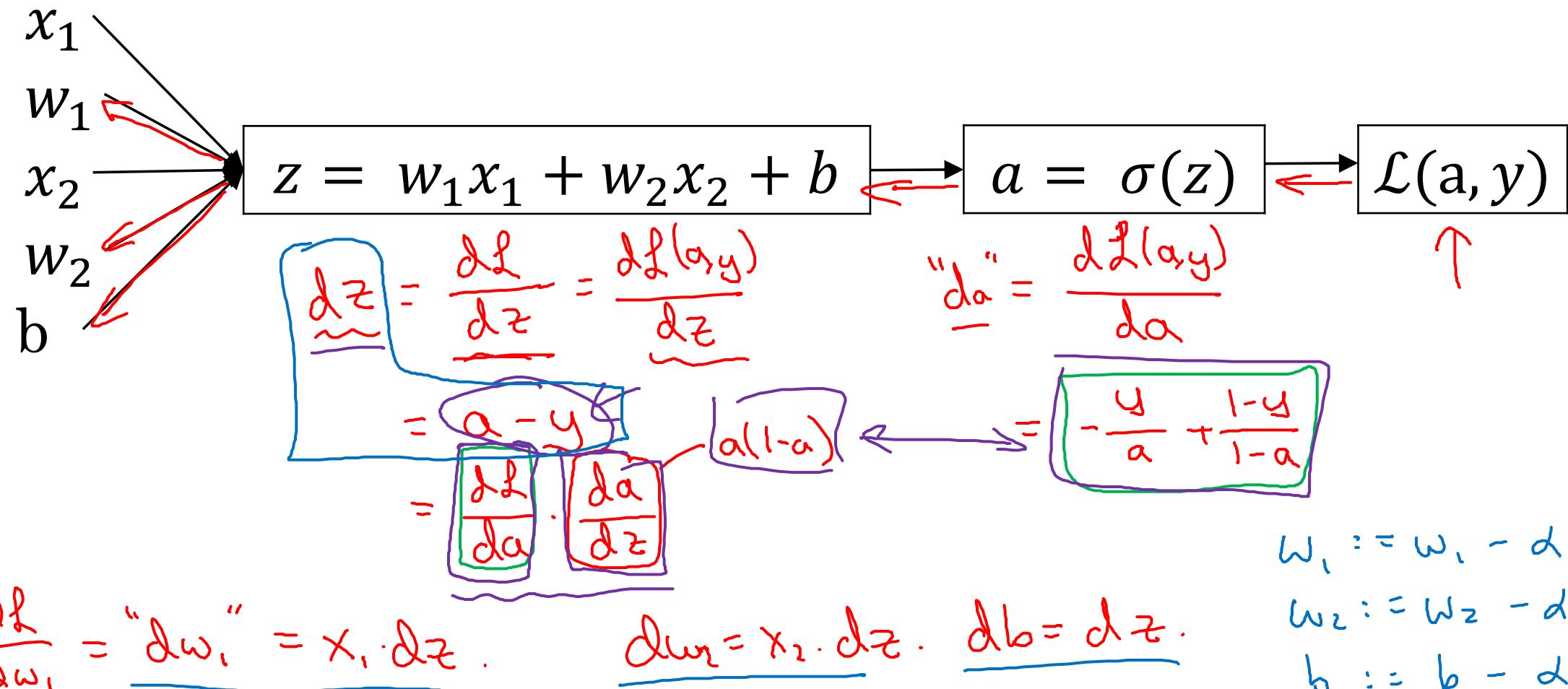
$$\rightarrow z = w^T x + b$$

$$\rightarrow \hat{y} = a = \sigma(z)$$

$$\rightarrow \mathcal{L}(a, y) = -(y \log(a) + (1 - y) \log(1 - a))$$



Logistic regression derivatives



$$\frac{\partial \mathcal{L}}{\partial w_i} = "dw_i" = x_i \cdot dz.$$

$$dw_1 = x_1 \cdot dz. \quad dw_2 = x_2 \cdot dz.$$

$$w_1 := w_1 - \alpha dw_1$$

$$w_2 := w_2 - \alpha dw_2$$

$$b := b - \alpha db.$$



deeplearning.ai

Basics of Neural Network Programming

Gradient descent
on m examples

Logistic regression on m examples

$$\underline{J(\omega, b)} = \frac{1}{m} \sum_{i=1}^m l(a^{(i)}, y^{(i)}) \quad (x^{(i)}, y^{(i)})$$
$$\Rightarrow a^{(i)} = \hat{y}^{(i)} = g(z^{(i)}) = g(\omega^\top x^{(i)} + b) \quad \underline{dw_1^{(i)}}, \underline{dw_2^{(i)}}, \underline{db^{(i)}}$$

$$\underline{\frac{\partial}{\partial \omega_1} J(\omega, b)} = \frac{1}{m} \sum_{i=1}^m \underbrace{\frac{\partial}{\partial \omega_1} l(a^{(i)}, y^{(i)})}_{\underline{dw_1^{(i)}} - (x^{(i)}, y^{(i)})}$$

Logistic regression on m examples

$$J=0; \underline{\Delta w_1}=0; \underline{\Delta w_2}=0; \underline{\Delta b}=0$$

→ For $i = 1$ to m

$$z^{(i)} = \omega^\top x^{(i)} + b$$

$$\alpha^{(i)} = \sigma(z^{(i)})$$

$$J_t = -[y^{(i)} \log \alpha^{(i)} + (1-y^{(i)}) \log(1-\alpha^{(i)})]$$

$$\underline{\Delta z^{(i)}} = \alpha^{(i)} - y^{(i)}$$

$$\begin{aligned} \Delta w_1 &+= x_1^{(i)} \Delta z^{(i)} \\ \Delta w_2 &+= x_2^{(i)} \Delta z^{(i)} \end{aligned}$$

$$\begin{aligned} \Delta b &+= \Delta z^{(i)} \\ \Delta w_3 & \\ \vdots & \\ \Delta w_n & \end{aligned}$$

$$J / m \leftarrow$$

$$\Delta w_1 / m; \Delta w_2 / m; \Delta b / m. \leftarrow$$

$$\Delta w_1 = \frac{\partial J}{\partial w_1}$$

$$w_1 := w_1 - \alpha \underline{\Delta w_1}$$

$$w_2 := w_2 - \alpha \underline{\Delta w_2}$$

$$b := b - \alpha \underline{\Delta b}.$$

Vectorization



deeplearning.ai

Basics of Neural Network Programming

Vectorization

What is vectorization?

$$z = \underbrace{\omega^T x}_{\text{Non-vectorized}} + b$$

Non-vectorized:

$$z = 0$$

for i in range(n - x):
 $z += \omega[i] * x[i]$

$$z += b$$

$$\omega = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix} \quad x = \begin{bmatrix} \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

$$\omega \in \mathbb{R}^{n_x} \quad x \in \mathbb{R}^{n_x}$$

Vectorized

$$z = \underbrace{\text{np.dot}(\omega, x)}_{\omega^T x} + b$$

\rightarrow GPU } SIMD - single instruction
 \rightarrow CPU } multiple data.



deeplearning.ai

Basics of Neural Network Programming

More vectorization examples

Neural network programming guideline

Whenever possible, avoid explicit for-loops.

$$u = Av$$

$$u_i = \sum_j A_{ij} v_j$$

$u = np.zeros((n,))$

for i ...

 for j ...

$u[i] += A[:, i] * v[j]$

$$u = np.dot(A, v)$$

Vectors and matrix valued functions

Say you need to apply the exponential operation on every element of a matrix/vector.

$$v = \begin{bmatrix} v_1 \\ \vdots \\ v_n \end{bmatrix} \rightarrow u = \begin{bmatrix} e^{v_1} \\ e^{v_2} \\ \vdots \\ e^{v_n} \end{bmatrix}$$

```
→ u = np.zeros((n,1))  
for i in range(n): ←  
    → u[i]=math.exp(v[i])
```

```
import numpy as np  
u = np.exp(v) ←  
↑  
np.log(v)  
np.abs(v)  
np.maximum(v, 0)  
v**2  
v/v
```

Logistic regression derivatives

$$J = 0, \boxed{dw_1 = 0, dw_2 = 0}, db = 0$$

$$d\omega = np.zeros((n_x, 1))$$

for i = 1 to n:

$$z^{(i)} = w^T x^{(i)} + b$$

$$a^{(i)} = \sigma(z^{(i)})$$

$$J += -[y^{(i)} \log \hat{y}^{(i)} + (1 - y^{(i)}) \log(1 - \hat{y}^{(i)})]$$

$$dz^{(i)} = a^{(i)}(1 - a^{(i)})$$

for $j=1 \dots n_x$
 $d\omega_j \leftarrow \dots$

$$\boxed{\begin{aligned} dw_1 &+= x_1^{(i)} dz^{(i)} \\ dw_2 &+= x_2^{(i)} dz^{(i)} \\ db &+= dz^{(i)} \end{aligned}}$$

$n_x = 2$

$$d\omega += x^{(i)} dz^{(i)}$$

$$J = J/m, \boxed{dw_1 = dw_1/m, dw_2 = dw_2/m}, db = db/m$$

$$d\omega /= m.$$



deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression

Vectorizing Logistic Regression

$$\begin{aligned} \rightarrow z^{(1)} &= w^T x^{(1)} + b \\ \rightarrow a^{(1)} &= \sigma(z^{(1)}) \end{aligned}$$

$$\begin{aligned} z^{(2)} &= w^T x^{(2)} + b \\ a^{(2)} &= \sigma(z^{(2)}) \end{aligned}$$

$$\begin{aligned} z^{(3)} &= w^T x^{(3)} + b \\ a^{(3)} &= \sigma(z^{(3)}) \end{aligned}$$

$$\underline{\underline{X}} = \begin{bmatrix} x^{(1)} & x^{(2)} & \dots & x^{(m)} \end{bmatrix}$$

$\xrightarrow{\text{---}}$

$$\frac{(n_x, m)}{\mathbb{R}^{n_x \times m}}$$

$$\overline{\omega^T} \begin{bmatrix} 1 & x^{(1)} & x^{(2)} & \dots & x^{(m)} \end{bmatrix}$$

$$\underline{\underline{Z}} = \begin{bmatrix} z^{(1)} & z^{(2)} & \dots & z^{(m)} \end{bmatrix} = \underline{\omega^T X} + \underbrace{\begin{bmatrix} b & b & \dots & b \end{bmatrix}_{1 \times m}}_{\rightarrow} = \begin{bmatrix} \underline{\underline{w^T x^{(1)} + b}} \\ \underline{\underline{w^T x^{(2)} + b}} \\ \vdots \\ \underline{\underline{w^T x^{(m)} + b}} \end{bmatrix}$$

$$\rightarrow \underline{\underline{Z}} = \text{np.dot}(\underline{\omega^T}, \underline{\underline{X}}) + \underline{\underline{b}} \in \mathbb{R}^{(1, m)}$$

"Broadcasting"

$$\underline{\underline{A}} = \begin{bmatrix} a^{(1)} & a^{(2)} & \dots & a^{(m)} \end{bmatrix} = \underline{\underline{\sigma(Z)}}$$



deeplearning.ai

Basics of Neural Network Programming

Vectorizing Logistic Regression's Gradient Computation

Vectorizing Logistic Regression

$$dz^{(1)} = a^{(1)} - y^{(1)} \quad dz^{(2)} = a^{(2)} - y^{(2)}$$

$$dZ = [dz^{(1)} \ dz^{(2)} \ \dots \ dz^{(m)}] \quad | \times m$$

$$A = [a^{(1)} \ \dots \ a^{(m)}]. \quad Y = [y^{(1)} \ \dots \ y^{(m)}]$$

$$\rightarrow dZ = A - Y = [a^{(1)} - y^{(1)} \ a^{(2)} - y^{(2)} \ \dots]$$

$$\begin{aligned} \rightarrow dw &= 0 \\ dw + &= \frac{x^{(1)} dz^{(1)}}{} \\ dw + &= \frac{x^{(2)} dz^{(2)}}{} \\ &\vdots \\ dw &= m \end{aligned}$$

$$\begin{aligned} db &= 0 \\ db + &= dz^{(1)} \\ db + &= dz^{(2)} \\ &\vdots \\ db &= m \end{aligned}$$

$$\begin{aligned} db &= \frac{1}{m} \sum_{i=1}^m dz^{(i)} \\ &= \frac{1}{m} \text{np. sum}(dZ) \end{aligned}$$

$$\begin{aligned} dw &= \frac{1}{m} X dZ^T \\ &= \frac{1}{m} \left[\begin{array}{c|c} x^{(1)} & \cdots & x^{(m)} \\ \hline 1 & & 1 \end{array} \right] \left[\begin{array}{c} dz^{(1)} \\ \vdots \\ dz^{(m)} \end{array} \right] \\ &= \frac{1}{m} \left[\underline{x^{(1)} dz^{(1)}} + \cdots + \underline{x^{(m)} dz^{(m)}} \right] \\ &\qquad\qquad\qquad n \times 1 \end{aligned}$$

Implementing Logistic Regression

$$J = 0, dw_1 = 0, dw_2 = 0, db = 0$$

for i = 1 to m:

$$z^{(i)} = w^T x^{(i)} + b$$

$$a^{(i)} = \sigma(z^{(i)})$$

$$J += -[y^{(i)} \log a^{(i)} + (1 - y^{(i)}) \log(1 - a^{(i)})]$$

$$dz^{(i)} = a^{(i)} - y^{(i)}$$

$$\left. \begin{array}{l} dw_1 += x_1^{(i)} dz^{(i)} \\ dw_2 += x_2^{(i)} dz^{(i)} \end{array} \right\} dw += X^{(i)} * dz^{(i)}$$

$$db += dz^{(i)}$$

$$J = J/m, dw_1 = dw_1/m, dw_2 = dw_2/m$$

$$db = db/m$$

```
for iter in range(1000):  
    z = w^T X + b  
    = np.dot(w.T, X) + b  
    A = sigmoid(z)  
    dZ = A - Y  
    dw = 1/m * X * dZ^T  
    db = 1/m * np.sum(dZ)  
  
    w := w - alpha * dw  
    b := b - alpha * db
```



deeplearning.ai

Basics of Neural Network Programming

Broadcasting in Python

Broadcasting example

Calories from Carbs, Proteins, Fats in 100g of different foods:

	Apples	Beef	Eggs	Potatoes
Carb	56.0	0.0	4.4	68.0
Protein	1.2	104.0	52.0	8.0
Fat	1.8	135.0	99.0	0.9

59 cal $\frac{56}{59} \approx 94.9\%$

$$= A_{(3,4)}$$



Calculate % of calories from Carb, Protein, Fat. Can you do this without explicit for-loop?

`cal = A.sum(axis = 0)`

`percentage = 100*A/(cal.reshape(1,4))`

$\uparrow (3,4) / (1,4)$

Broadcasting example

$$\begin{bmatrix} 1 \\ 2 \\ 3 \\ 4 \end{bmatrix} + \begin{bmatrix} 100 \\ 100 \\ 100 \\ 100 \end{bmatrix} \xrightarrow{100}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 100 & 200 & 300 \\ 100 & 200 & 300 \end{bmatrix} \xleftarrow{(m,n) \quad (2,3)} \xrightarrow{(1,n) \rightsquigarrow (m,n) \quad (2,3)}$$

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \end{bmatrix} + \begin{bmatrix} 100 & 100 & 100 \\ 200 & 200 & 200 \end{bmatrix} = \xleftarrow{(m,n)}$$

General Principle

$$\begin{array}{ccc} \text{(m,n)} & \begin{matrix} + \\ - \\ * \\ / \end{matrix} & \begin{array}{c} \text{(1,n)} \\ \text{(m,1)} \end{array} \end{array} \rightsquigarrow \begin{array}{c} \text{(m,n)} \\ \rightsquigarrow \text{(m,n)} \end{array}$$

$$\begin{array}{ccccc} \text{(m,1)} & + & \mathbb{R} & & \\ \left[\begin{smallmatrix} 1 \\ 2 \\ 3 \end{smallmatrix} \right] & + & 100 & = & \left[\begin{smallmatrix} 101 \\ 102 \\ 103 \end{smallmatrix} \right] \\ \left[\begin{smallmatrix} 1 & 2 & 3 \end{smallmatrix} \right] & + & 100 & = & \left[\begin{smallmatrix} 101 & 102 & 103 \end{smallmatrix} \right] \end{array}$$

Matlab/Octave: bsxfun



deeplearning.ai

Basics of Neural Network Programming

Explanation of logistic
regression cost function
(Optional)

Logistic regression cost function

$$\hat{y} = g(w^T x + b) \quad \text{where} \quad g(z) = \frac{1}{1+e^{-z}}$$

Interpret $\hat{y} = p(y=1|x)$

If $y=1$: $p(y|x) = \hat{y}$

If $y=0$: $p(y|x) = \underline{1 - \hat{y}}$

Logistic regression cost function

$$\begin{aligned} \rightarrow & \boxed{\text{If } y = 1: \quad p(y|x) = \hat{y}} \\ \rightarrow & \boxed{\text{If } y = 0: \quad p(y|x) = 1 - \hat{y}} \end{aligned}$$

$p(y|x)$

$$p(y|x) = \hat{y}^y (1-\hat{y})^{(1-y)}$$

←

$$\text{If } y=1: \quad p(y|x) = \hat{y} \underset{=} {=} 1$$
$$\text{If } y=0: \quad p(y|x) = \hat{y}^0 (1-\hat{y})^{(1-y)} = 1 \times (1-\hat{y}) = 1 - \hat{y}$$
$$\uparrow \log p(y|x) = \log \hat{y}^y (1-\hat{y})^{(1-y)} = y \log \hat{y} + (1-y) \log (1-\hat{y})$$
$$= -\frac{1}{m} \sum_{i=1}^m (\hat{y}_i - y_i)$$

↓

Cost on m examples

$$\log p(\text{labels in training set}) = \log \prod_{i=1}^m p(y^{(i)} | x^{(i)}) \quad \leftarrow$$

$$\log p(\dots) = \sum_{i=1}^m \underbrace{\log p(y^{(i)} | x^{(i)})}_{-L(\hat{y}^{(i)}, y^{(i)})}$$

Maximum likelihood
estimation \nearrow

$$= -\sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$$

Cost: $J(w, b)$ = $\frac{1}{m} \sum_{i=1}^m L(\hat{y}^{(i)}, y^{(i)})$

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>

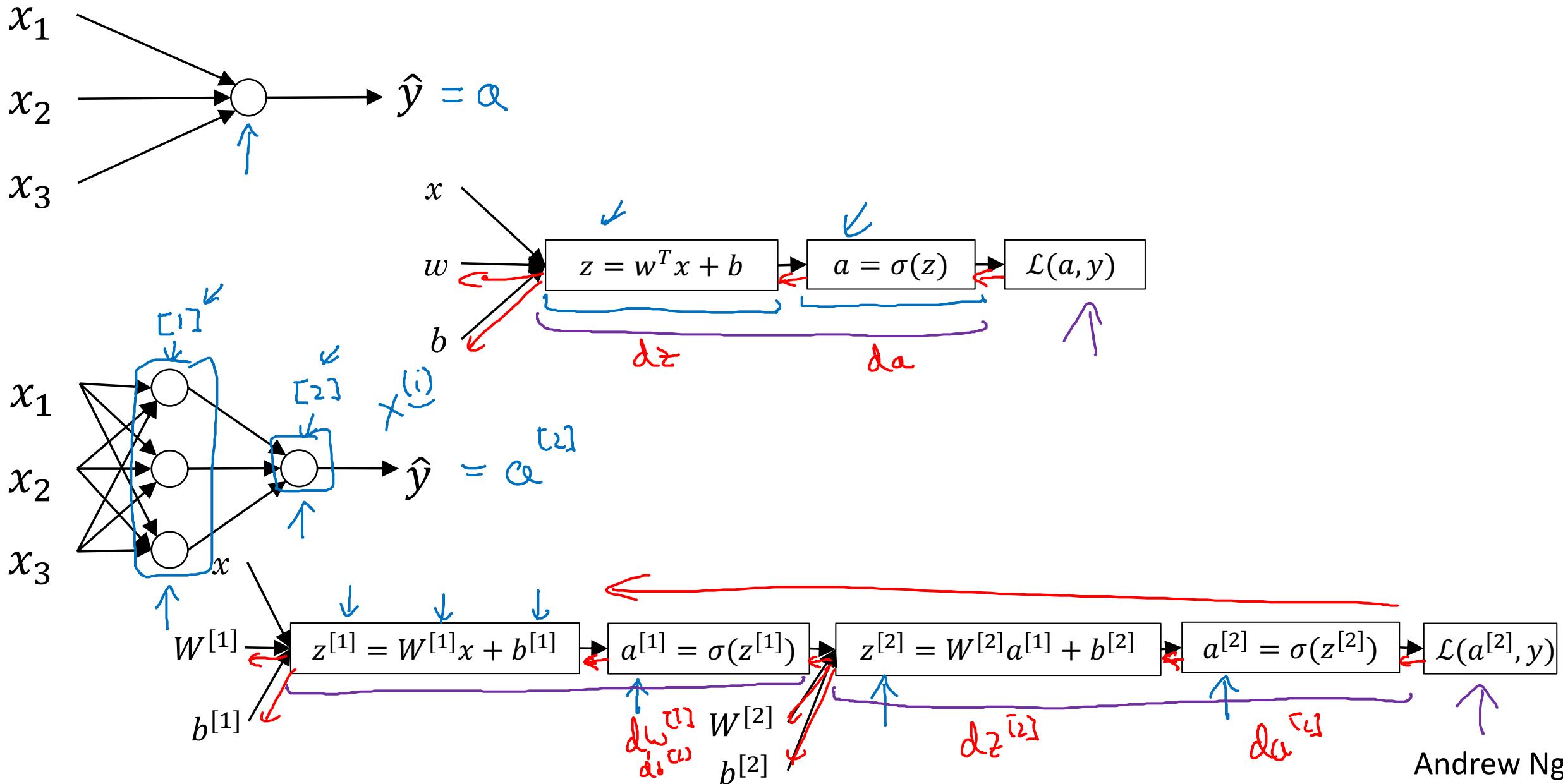


deeplearning.ai

One hidden layer
Neural Network

Neural Networks
Overview

What is a Neural Network?



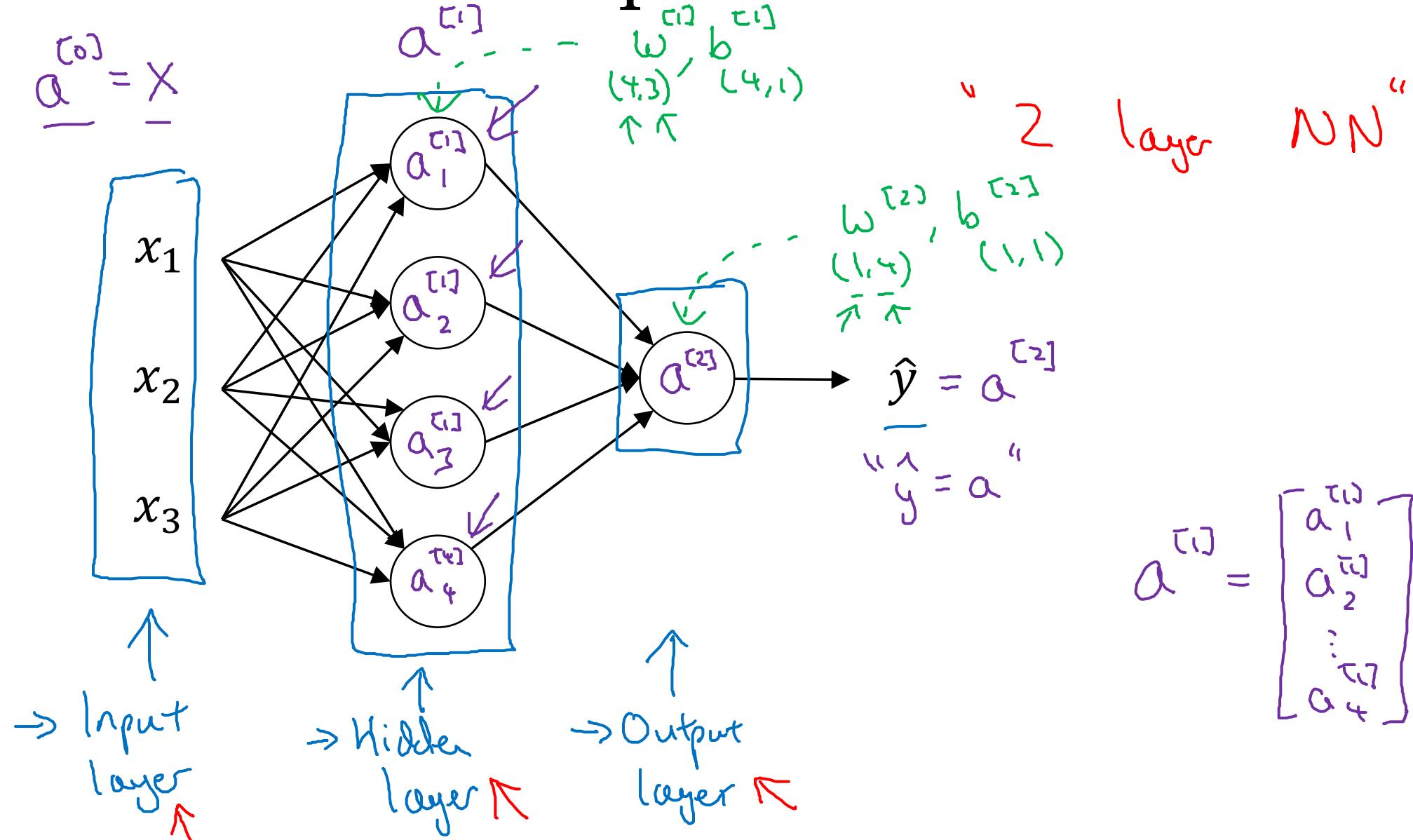


deeplearning.ai

One hidden layer
Neural Network

Neural Network
Representation

Neural Network Representation



$$a^{(1)} = \begin{bmatrix} a_1^{(1)} \\ a_2^{(1)} \\ \vdots \\ a_4^{(1)} \end{bmatrix}$$

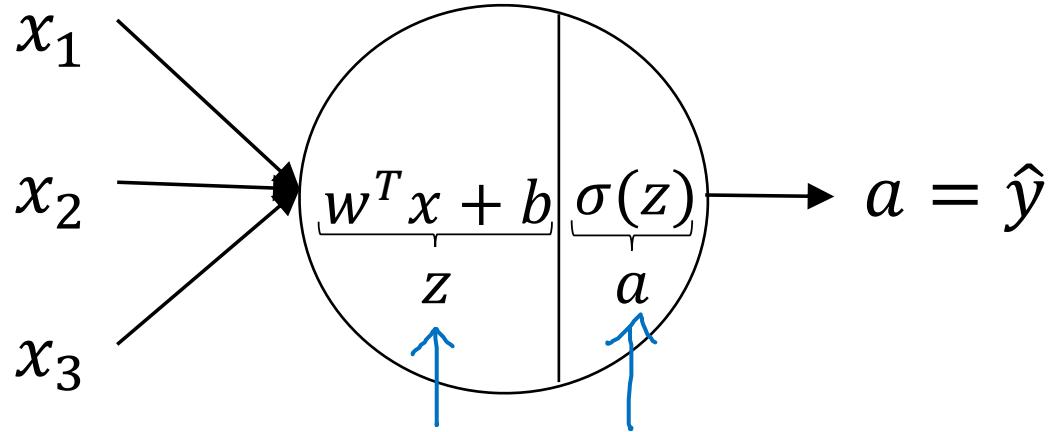


deeplearning.ai

One hidden layer Neural Network

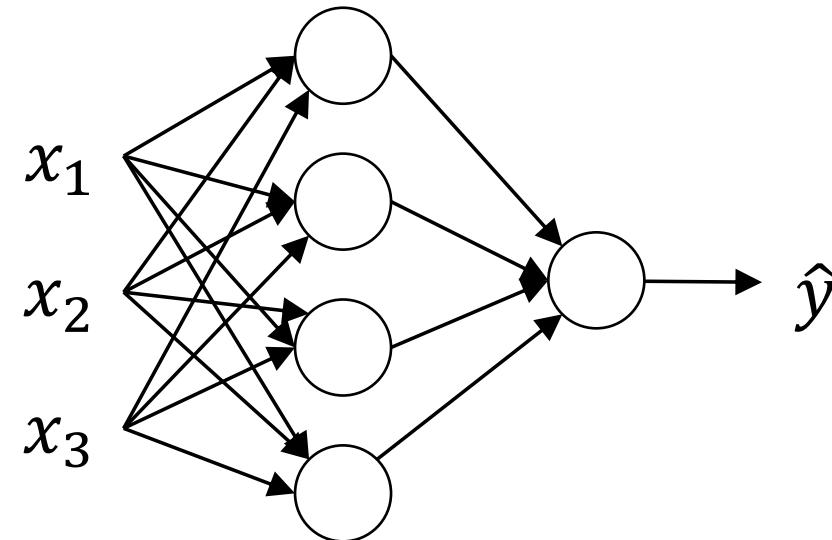
Computing a Neural Network's Output

Neural Network Representation

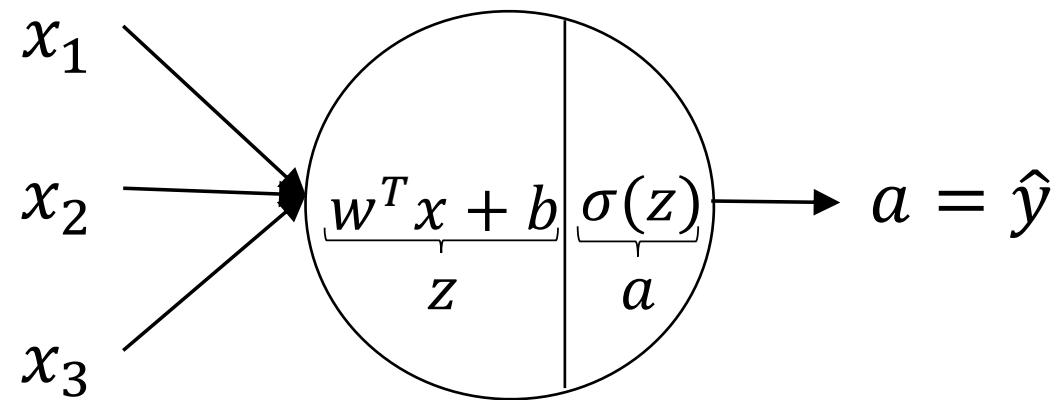


$$z = w^T x + b$$

$$a = \sigma(z)$$

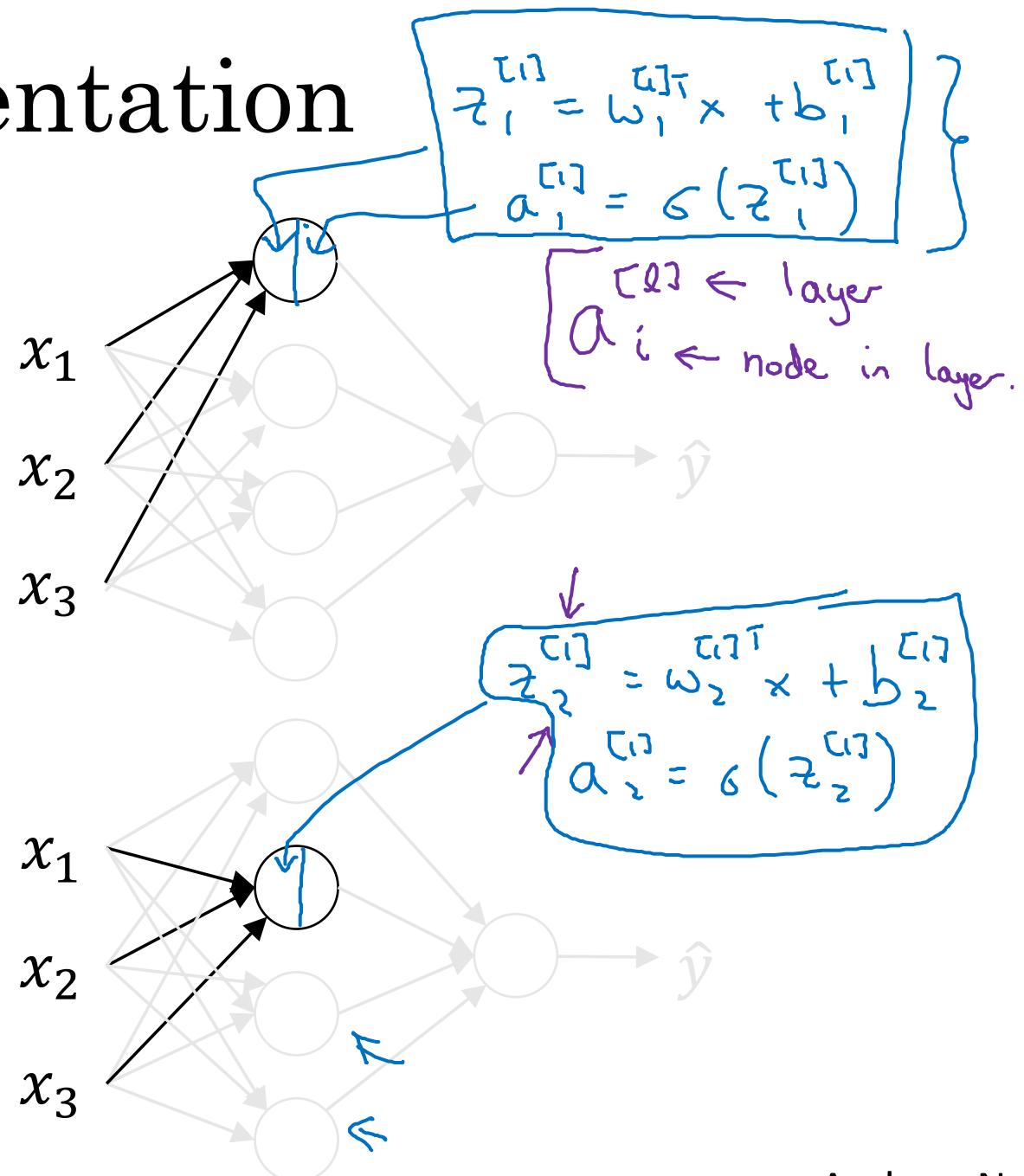


Neural Network Representation

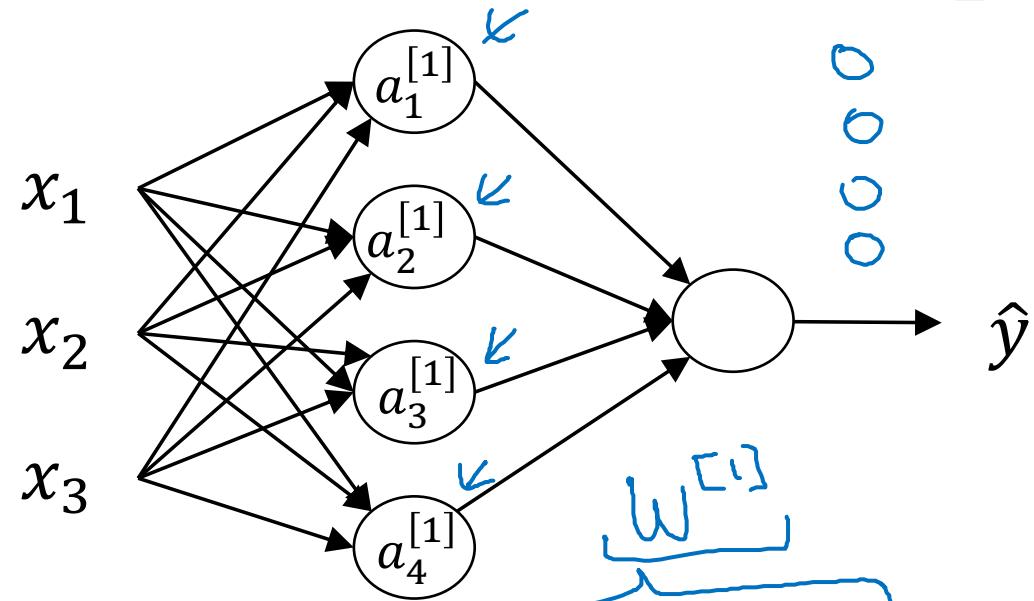


$$z = w^T x + b$$

$$a = \sigma(z)$$



Neural Network Representation



$$\rightarrow z^{[1]} = \begin{bmatrix} w_1^{[1]T} \\ w_2^{[1]T} \\ w_3^{[1]T} \\ w_4^{[1]T} \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} + \begin{bmatrix} b_1^{[1]} \\ b_2^{[1]} \\ b_3^{[1]} \\ b_4^{[1]} \end{bmatrix}$$

$$\rightarrow a^{[1]} = \begin{bmatrix} a_1^{[1]} \\ \vdots \\ a_4^{[1]} \end{bmatrix} = g(z^{[1]})$$

Diagram illustrating the mathematical representation of the neural network layers:

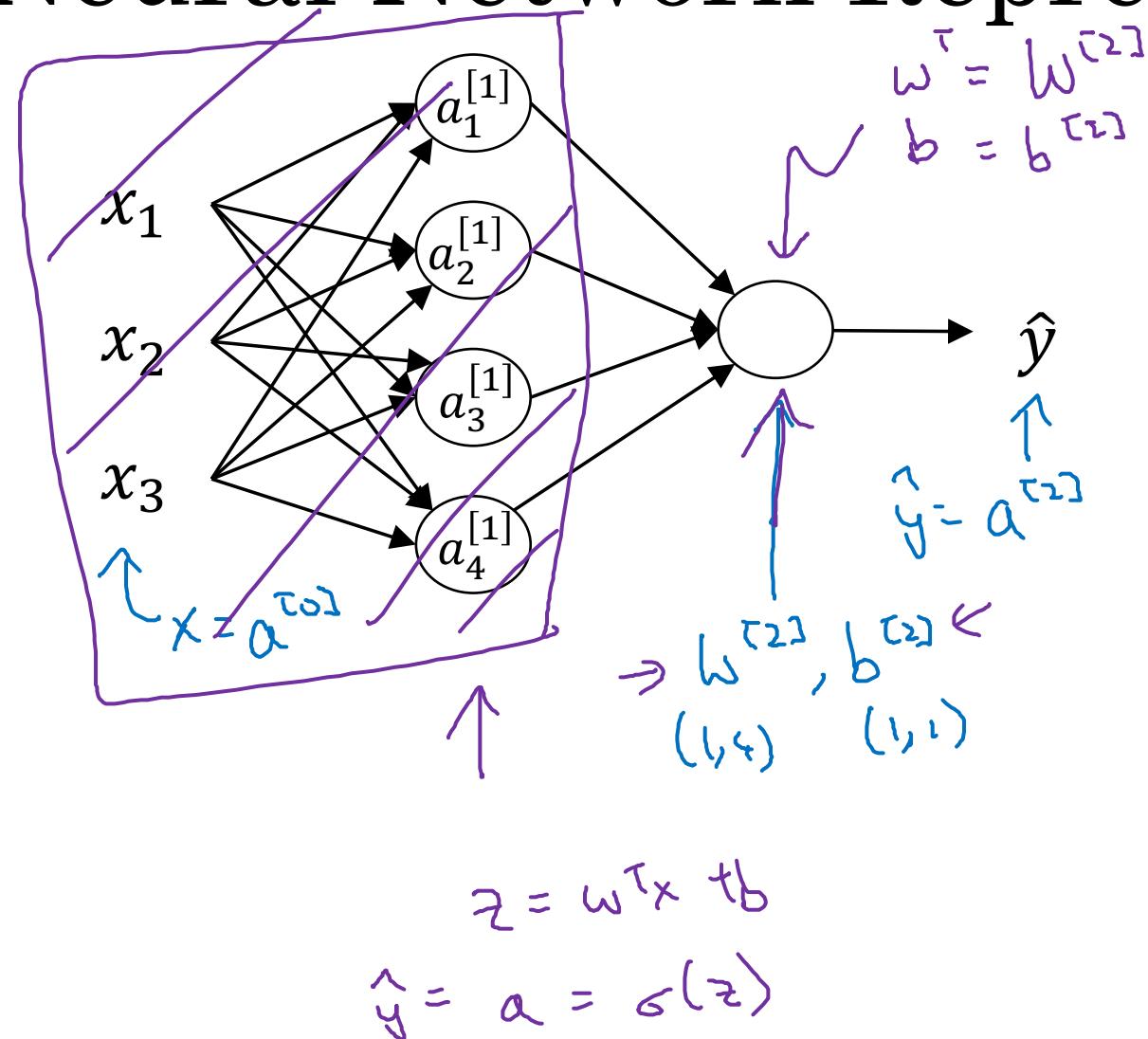
- Layer 1: $z_1^{[1]} = w_1^{[1]T} x + b_1^{[1]}$, $a_1^{[1]} = \sigma(z_1^{[1]})$
- Layer 2: $z_2^{[1]} = w_2^{[1]T} x + b_2^{[1]}$, $a_2^{[1]} = \sigma(z_2^{[1]})$
- Layer 3: $z_3^{[1]} = w_3^{[1]T} x + b_3^{[1]}$, $a_3^{[1]} = \sigma(z_3^{[1]})$
- Layer 4: $z_4^{[1]} = w_4^{[1]T} x + b_4^{[1]}$, $a_4^{[1]} = \sigma(z_4^{[1]})$

Annotations:

- $(w_i^{[1]T} x)$ is highlighted in blue.
- $a^{[1]}$ is highlighted in red.
- $\sigma(z^{[1]})$ is highlighted in red.
- $z^{[1]}$ is highlighted in purple.
- $b^{[1]}$ is highlighted in green.
- $w^{[1]}$ is highlighted in blue.

$$\rightarrow w_1^{[1]T} x + b_1^{[1]} \\ \rightarrow w_2^{[1]T} x + b_2^{[1]} \\ \rightarrow w_3^{[1]T} x + b_3^{[1]} \\ \rightarrow w_4^{[1]T} x + b_4^{[1]} \\ = \begin{bmatrix} z_1^{[1]} \\ z_2^{[1]} \\ z_3^{[1]} \\ z_4^{[1]} \end{bmatrix}$$

Neural Network Representation learning



Given input x :

$$\rightarrow z^{[1]} = W^{[1]} a^{[0]} + b^{[1]}$$

$(4,1)$ $(4,3)$ $(3,1)$ $(4,1)$

$$\rightarrow a^{[1]} = \sigma(z^{[1]})$$

$(4,1)$ $(4,1)$

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

$(1,1)$ $(1,4)$ $(4,1)$ $(1,1)$

$$\rightarrow a^{[2]} = \sigma(z^{[2]})$$

$(1,1)$ $(1,1)$

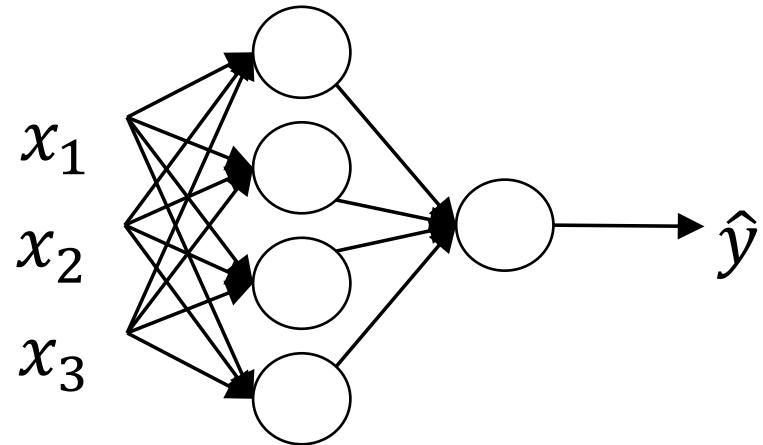


deeplearning.ai

One hidden layer Neural Network

Vectorizing across
multiple examples

Vectorizing across multiple examples



$x \rightarrow a^{[2]} = y$
 $x^{(1)} \rightarrow a^{[2](1)} = y^{(1)}$
 $x^{(2)} \rightarrow a^{2} = y^{(2)}$
 \vdots
 $x^{(n)} \rightarrow a^{[2](m)} = y^{(m)}$

$a^{[2](i)}$ example i
layer 2

$\left\{ \begin{array}{l} z^{[1]} = W^{[1]}x + b^{[1]} \\ a^{[1]} = \sigma(z^{[1]}) \\ z^{[2]} = W^{[2]}a^{[1]} + b^{[2]} \\ a^{[2]} = \sigma(z^{[2]}) \end{array} \right.$

for $i = 1$ to m ,

$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$
 $a^{[1](i)} = \sigma(z^{[1](i)})$
 $z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$
 $a^{[2](i)} = \sigma(z^{[2](i)})$

Vectorizing across multiple examples

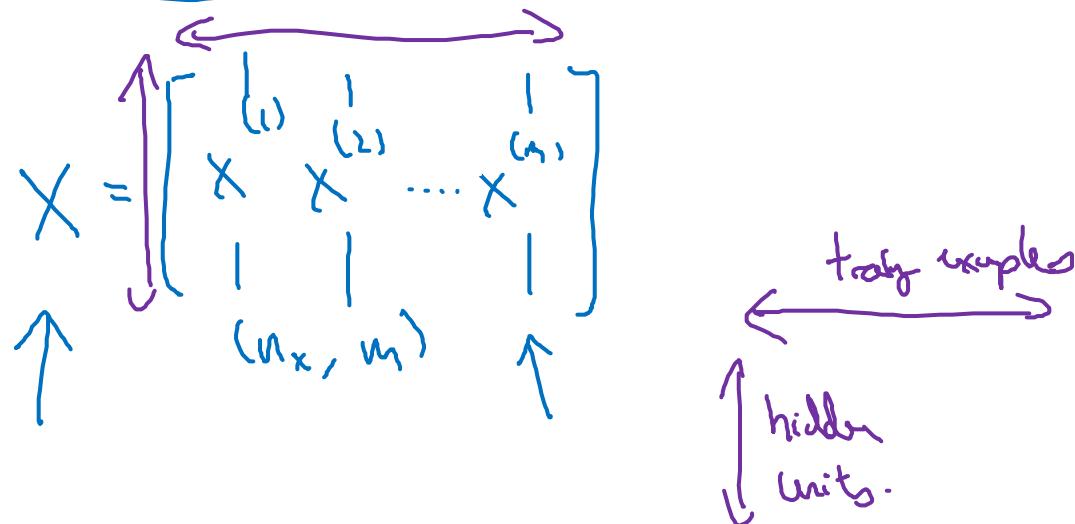
for $i = 1$ to m :

$$z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$$

$$a^{[1](i)} = \sigma(z^{[1](i)})$$

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

$$a^{[2](i)} = \sigma(z^{[2](i)})$$

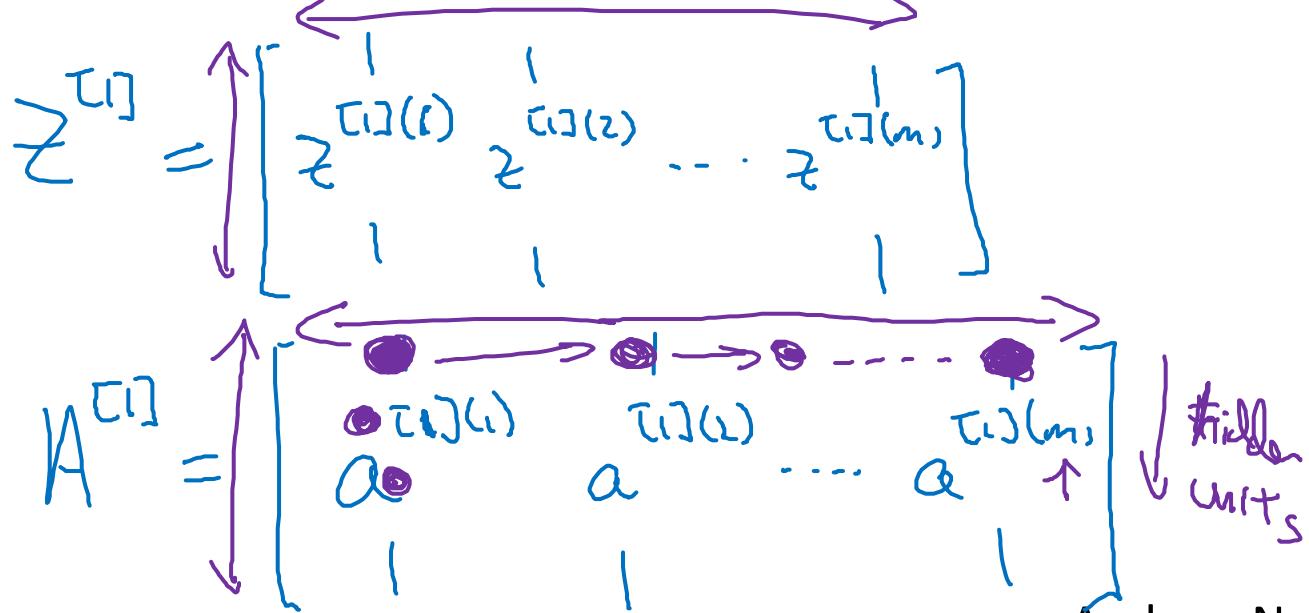


$$z^{[1]} = W^{[1]}X + b^{[1]}$$

$$\rightarrow A^{[1]} = \sigma(z^{[1]})$$

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

$$\rightarrow A^{[2]} = \sigma(z^{[2]})$$





deeplearning.ai

One hidden layer Neural Network

Explanation for vectorized implementation

Justification for vectorized implementation

$$\underline{z}^{1} = \underline{\omega}^{[1]} \times \underline{x}^{(1)} + \cancel{\underline{v}^{[1]}} ,$$

$$\underline{z}^{(1)(2)} = \underline{\omega}^{(1)(2)} \underline{x}^{(2)} + \underline{b}$$

$$\underline{z}^{(1)(3)} = \omega^{(1)} x^{(3)} + \cancel{b^{(1)}}$$

$$w^{(i)} = \begin{bmatrix} \text{---} \\ \text{---} \\ \text{---} \\ \text{---} \end{bmatrix}$$

$$\omega^{[1]} x^{(1)} = \begin{bmatrix} c \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

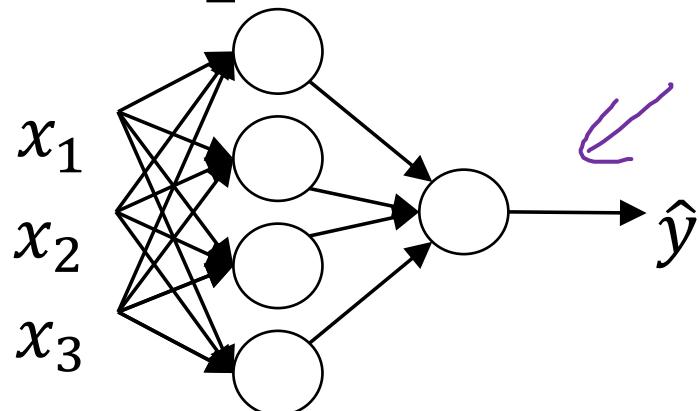
$$\omega^{(i)} x^{(i)} = \begin{bmatrix} \cdot \\ \cdot \\ \cdot \\ \cdot \\ \cdot \end{bmatrix}$$

$$\omega^{(1)} x^{(3)} = \begin{bmatrix} 1 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$z^{[i]} = \tilde{w}^{[i]} X + b^{[i]}$$

$\tilde{w}^{[i]}$ $\begin{bmatrix} 1 & | & | & | \\ X^{(1)} & X^{(2)} & X^{(3)} \dots & | \\ | & | & | & | \end{bmatrix}$ = $\begin{bmatrix} \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \end{bmatrix}$ = $\begin{bmatrix} 1 & | & | & | \\ z^{[1]1} & z^{[1]2} & z^{[1]3} \dots & | \\ | & | & | & | \end{bmatrix}$ = $\tilde{z}^{[i]}$
 $\tilde{w}^{[i]} X^{(i)} = z^{[i]}$

Recap of vectorizing across multiple examples



$$X = \begin{bmatrix} | & | & | \\ x^{(1)} & x^{(2)} & \dots & x^{(m)} \\ | & | & | \end{bmatrix}$$

$$\underline{A^{[1]}} = \begin{bmatrix} | & | & | \\ a^{1} & a^{[1](2)} & \dots & a^{[1](m)} \\ | & | & | \end{bmatrix}$$

for i = 1 to m

$\rightarrow z^{[1](i)} = W^{[1]}x^{(i)} + b^{[1]}$

$\rightarrow a^{[1](i)} = \sigma(z^{[1](i)})$

$\rightarrow z^{[2](i)} = W^{[2]}a^{[1](i)} + b^{[2]}$

$\rightarrow a^{[2](i)} = \sigma(z^{[2](i)})$

$x = a^{[0]}$ $x^{(i)} = a^{[0](i)}$

$Z^{[1]} = W^{[1]}X + b^{[1]} \leftarrow w^{[1]}A^{[0]} + b^{[1]}$

$A^{[1]} = \sigma(Z^{[1]})$

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

$A^{[2]} = \sigma(Z^{[2]})$

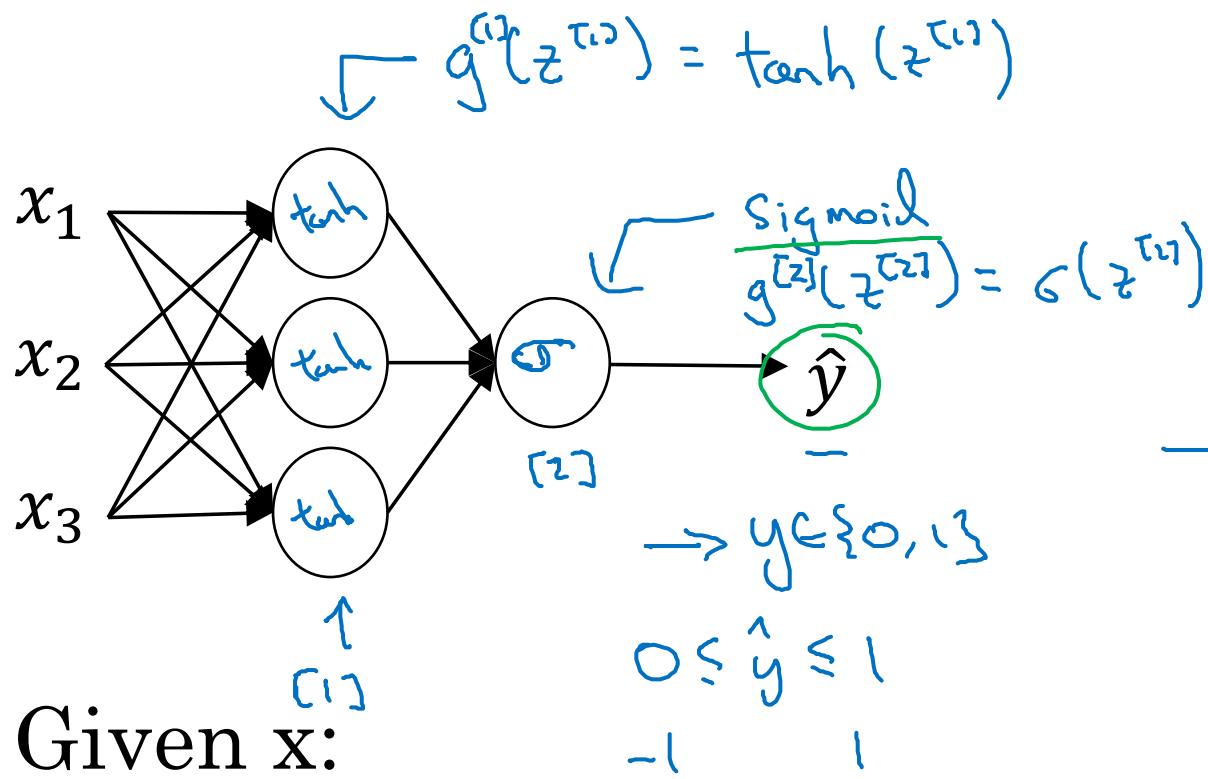


deeplearning.ai

One hidden layer Neural Network

Activation functions

Activation functions

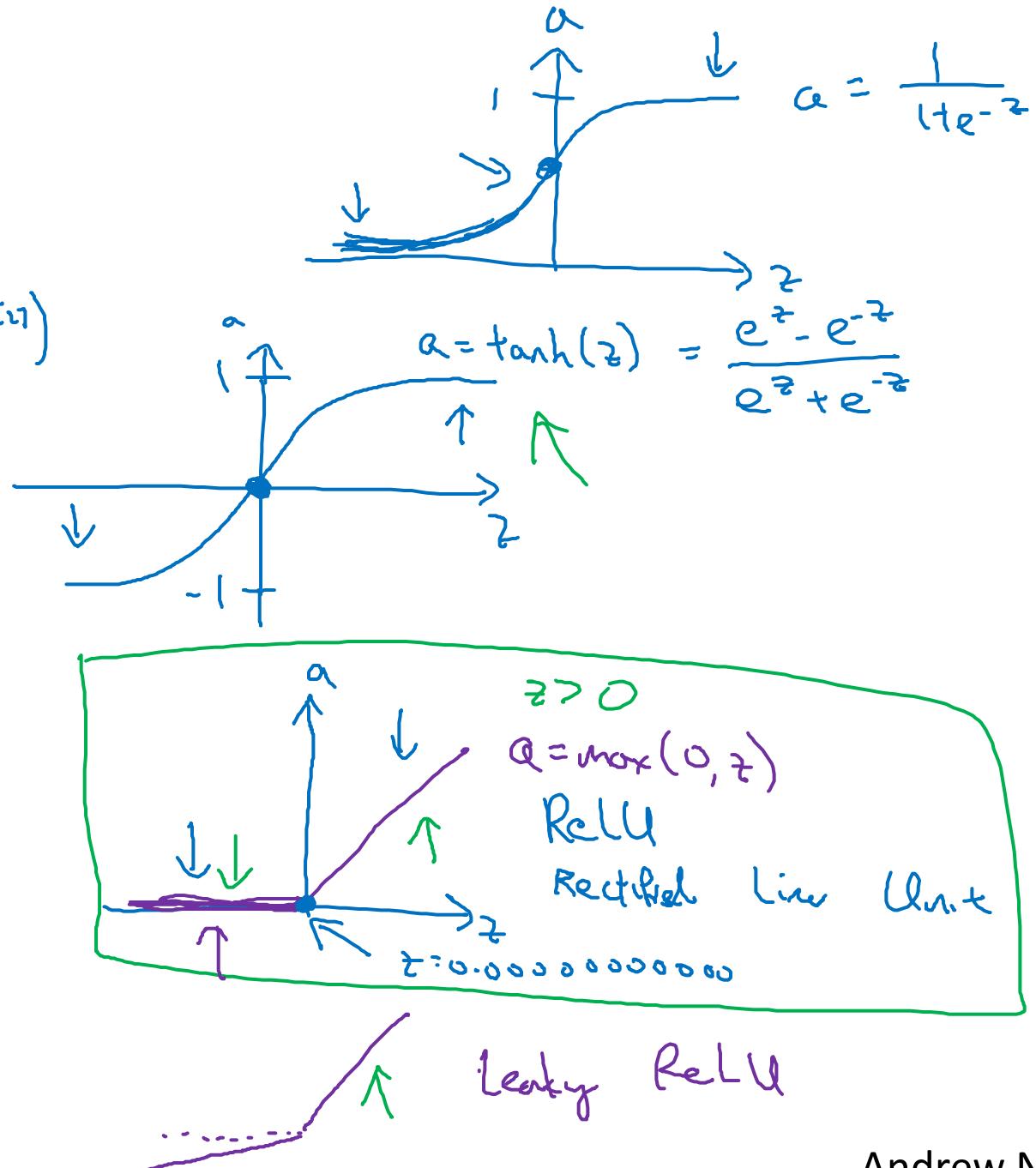


$$z^{[1]} = W^{[1]}x + b^{[1]}$$

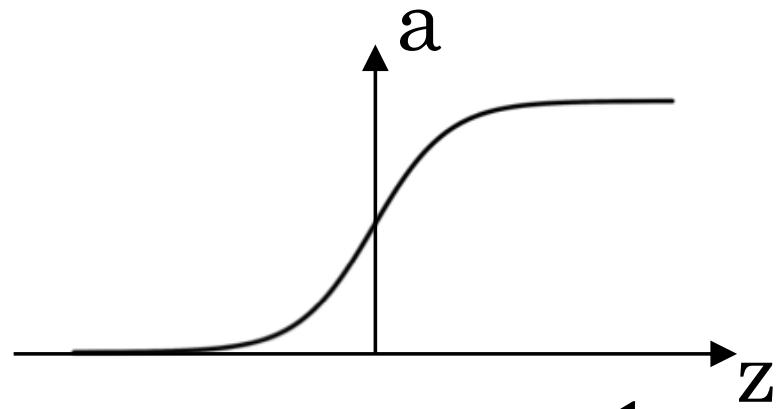
$$\rightarrow a^{[1]} = \sigma(z^{[1]})$$

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

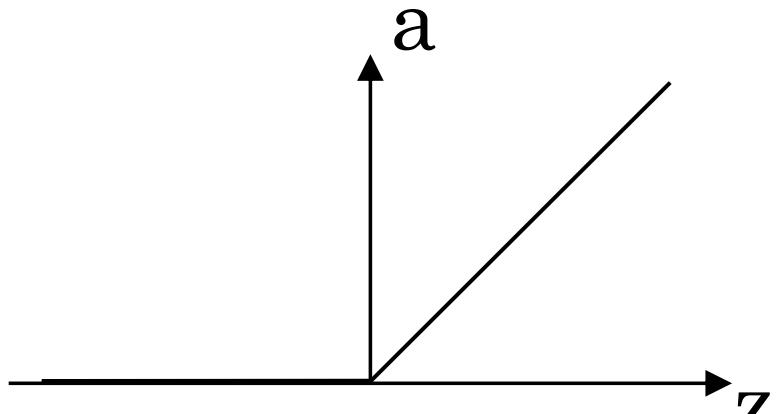
$$\rightarrow a^{[2]} = \sigma(\cancel{z^{[2]}}) \quad \cancel{g^{[2]}(z^{[2]})}$$



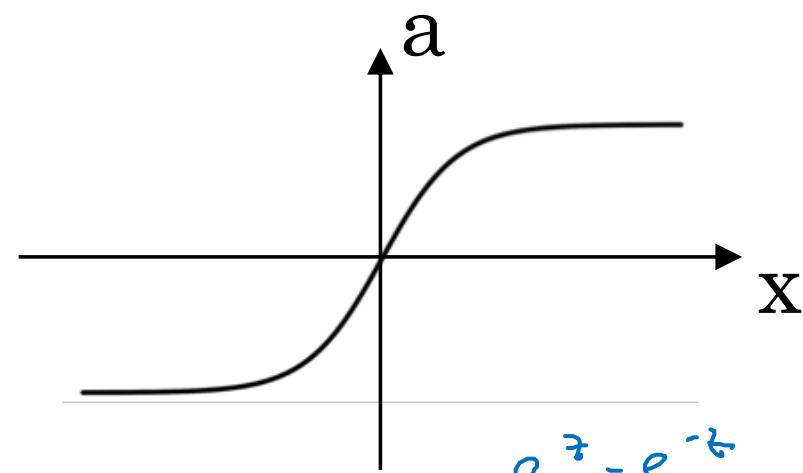
Pros and cons of activation functions



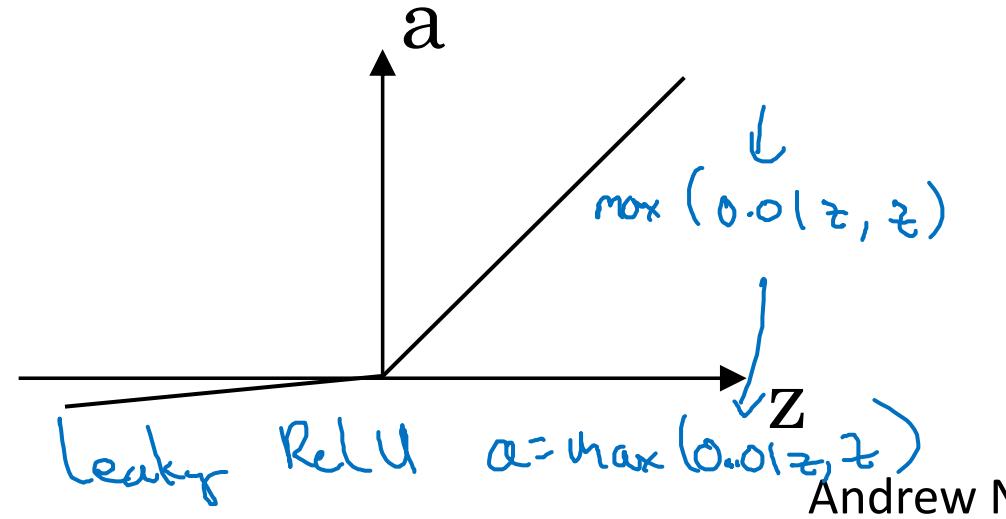
$$\text{sigmoid: } a = \frac{1}{1 + e^{-z}}$$



$$\text{ReLU} \quad a = \max(0, z)$$



$$\tanh: \quad a = \frac{e^z - e^{-z}}{e^z + e^{-z}}$$



$$\text{Leaky ReLU} \quad a = \max(0.01z, z) \quad \text{Andrew Ng}$$

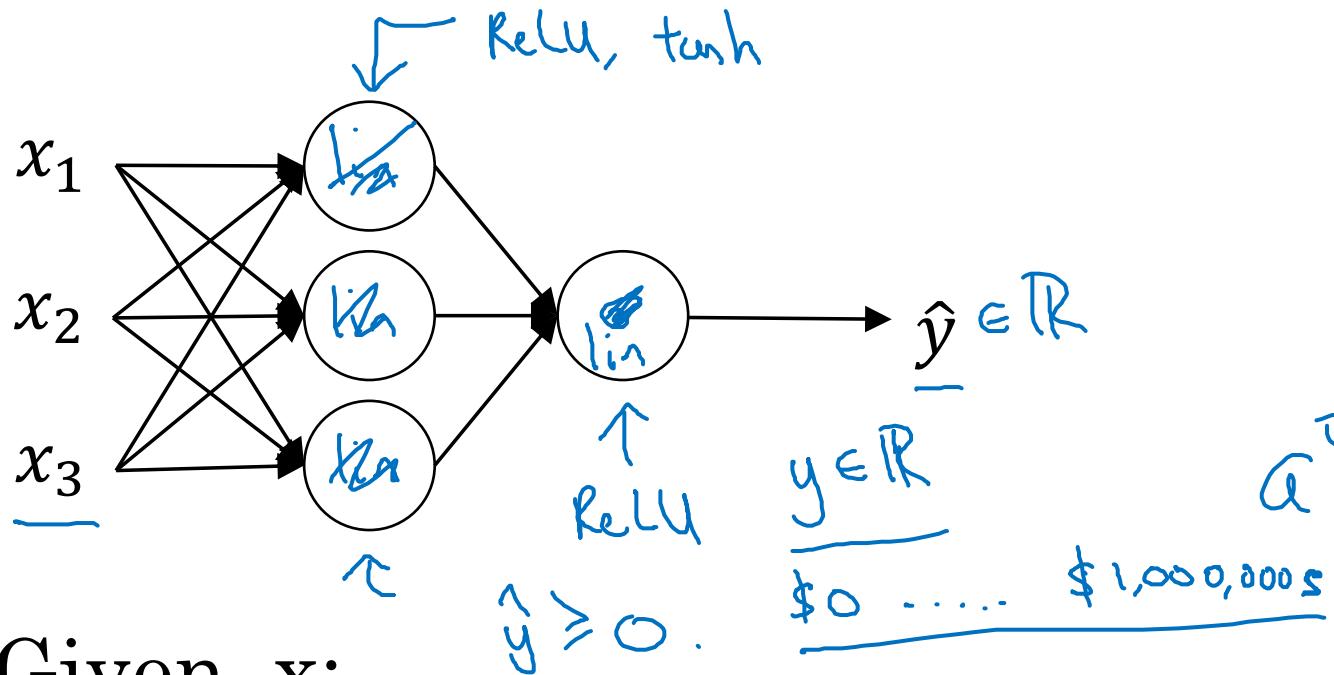


deeplearning.ai

One hidden layer Neural Network

Why do you
need non-linear
activation functions?

Activation function



Given x :

$$\rightarrow z^{[1]} = W^{[1]}x + b^{[1]}$$

$$\rightarrow a^{[1]} = \underline{g^{[1]}(z^{[1]})} \geq^{[1]}$$

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

$$\rightarrow a^{[2]} = \underline{g^{[2]}(z^{[2]})} \geq^{[2]}$$

$g(z) = z$
"linear activation
function"

$$a^{[1]} = z^{[1]} = \underbrace{W^{[1]}x + b^{[1]}}_{a^{[1]}}$$

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

$$a^{[2]} = W^{[2]} \left(\underbrace{W^{[1]}x + b^{[1]}}_{a^{[1]}} \right) + b^{[2]}$$

$$= (\underbrace{W^{[2]} W^{[1]}}_{w'})x + (\underbrace{W^{[2]} b^{[1]} + b^{[2]}}_{b'})$$

$$= \underline{w'x + b'}$$

$$g(z) = z$$

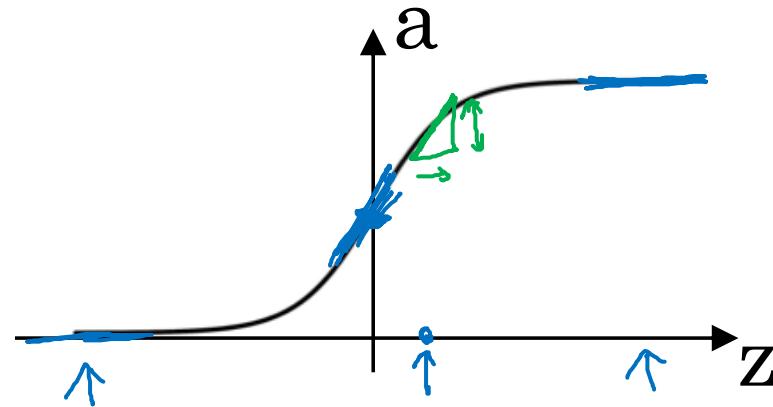


deeplearning.ai

One hidden layer Neural Network

Derivatives of activation functions

Sigmoid activation function



$$g(z) = \frac{1}{1 + e^{-z}}$$

$$a = g(z) = \frac{1}{1 + e^{-z}}$$

$$\begin{aligned} g'(z) &= \boxed{\frac{d}{dz} g(z)} \\ &= \frac{1}{1 + e^{-z}} \left(1 - \frac{1}{1 + e^{-z}} \right) \\ &= g(z) \left(1 - g(z) \right) \quad \leftarrow \begin{array}{l} | \\ g'(z) = a(1-a) \end{array} \\ &= \boxed{a(1-a)} \end{aligned}$$

$$z = 10, \quad g(z) \approx 1$$

$$\frac{d}{dz} g(z) \approx 1(1-1) \approx 0$$

$$z = -10, \quad g(z) \approx 0$$

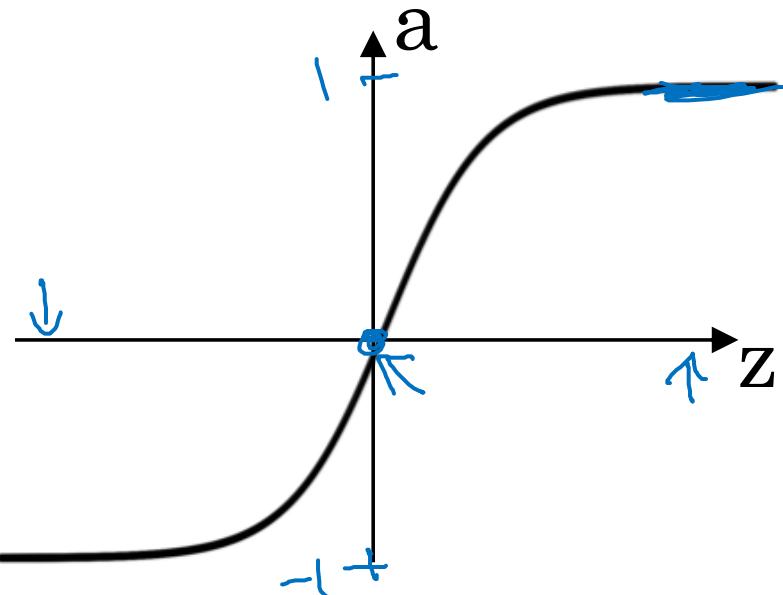
$$\frac{d}{dz} g(z) \approx 0 \cdot (1-0) \approx 0$$

$$z = 0, \quad g(z) = \frac{1}{2}$$

$$\frac{d}{dz} g(z) = \frac{1}{2}(1-\frac{1}{2}) = \frac{1}{4}$$

Andrew Ng

Tanh activation function



$$g(z) = \tanh(z)$$

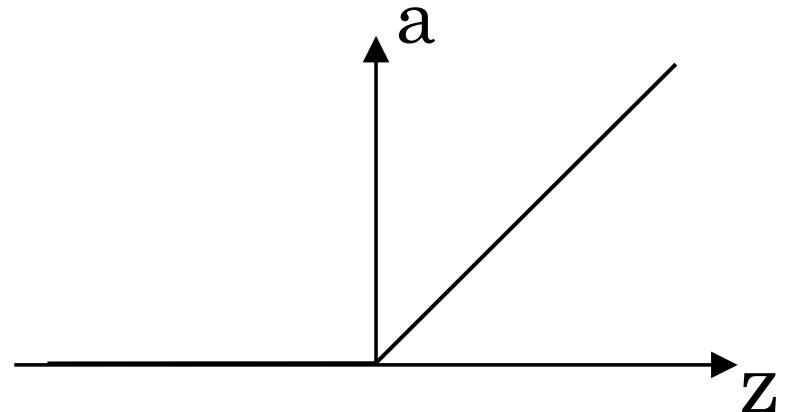
$$= \frac{e^z - e^{-z}}{e^z + e^{-z}}$$

$$\begin{aligned} g'(z) &= \frac{d}{dz} g(z) = \text{slope of } g(z) \text{ at } z \\ &= \underline{\underline{1 - (\tanh(z))^2}} \leftarrow \end{aligned}$$

$$a = g(z), \quad g'(z) = 1 - a^2$$

$$\begin{cases} z = 10 & \tanh(z) \approx 1 \\ g'(z) \approx 0 \\ z = -10 & \tanh(z) \approx -1 \\ g'(z) \approx 0 \\ z = 0 & \tanh(z) = 0 \\ g'(z) = 1 \end{cases}$$

ReLU and Leaky ReLU

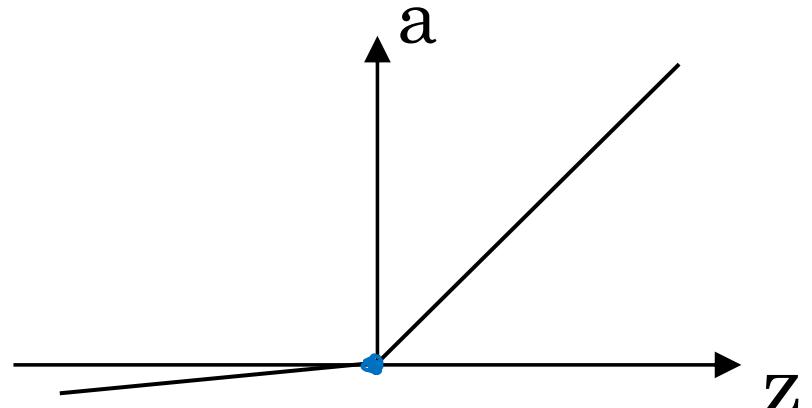


ReLU

$$g(z) = \max(0, z)$$

$$\Rightarrow g'(z) = \begin{cases} 0 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$

~~undefined if $z = 0$~~



Leaky ReLU

$$g(z) = \max(0.01z, z)$$

$$g'(z) = \begin{cases} 0.01 & \text{if } z < 0 \\ 1 & \text{if } z \geq 0 \end{cases}$$



deeplearning.ai

One hidden layer
Neural Network

Gradient descent for
neural networks

Gradient descent for neural networks

Parameters: $(\omega^{[1]}, b^{[1]}, \dots, \omega^{[L]}, b^{[L]})$ $n_x = n^{[0]}, n^{[1]}, \dots, \underline{n^{[L]}} = 1$

Cost function: $J(\underbrace{\omega^{[1]}, b^{[1]}, \dots, \omega^{[L]}, b^{[L]}}, \underbrace{\hat{y}_i}_{\in \mathcal{A}^{[L]}}) = \frac{1}{m} \sum_{i=1}^m l(\hat{y}_i, y_i)$

Gradient Descent:

→ Repeat {

→ Compute predict $(\hat{y}^{(i)}, i=1 \dots m)$

$$\frac{\partial J}{\partial \omega^{[l]}} = \frac{\partial J}{\partial \omega^{[l]}} , \quad \frac{\partial J}{\partial b^{[l]}} = \frac{\partial J}{\partial b^{[l]}} , \dots$$

$$\omega^{[l]} := \omega^{[l]} - \alpha \frac{\partial J}{\partial \omega^{[l]}}$$

$$b^{[l]} := b^{[l]} - \alpha \frac{\partial J}{\partial b^{[l]}}$$

↳

Formulas for computing derivatives

Forward propagation:

$$z^{[1]} = w^{[1]}X + b^{[1]}$$

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

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

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

Back propagation:

$$dz^{[2]} = A^{[2]} - Y \leftarrow$$

$$dW^{[2]} = \frac{1}{m} dz^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} \underline{\underline{\text{np.sum}(dz^{[2]}, \text{axis}=1, \text{keepdims=True})}}$$

$$dz^{[1]} = \underbrace{(w^{[2]T} dz^{[2]})}_{(n^{[2]}, m)} \times \underbrace{g^{[2]'}(z^{[2]})}_{\text{element-wise product}} \underbrace{(n^{[1]}, m)}$$

$$dW^{[1]} = \frac{1}{m} dz^{[1]} X^T$$

$$\cancel{db^{[1]} = \frac{1}{m} \underline{\underline{\text{np.sum}(dz^{[1]}, \text{axis}=1, \text{keepdims=True})}}} \quad \cancel{(n^{[1]}, 1)} \quad (n^{[1]}, 1) \quad \cancel{\text{reshape} \uparrow}$$

$$Y = [y^{(1)}, y^{(2)}, \dots, y^{(m)}]$$

$$(n^{[1]}) \leftarrow$$

$$\cancel{\downarrow} (n^{[2]}, 1) \leftarrow$$



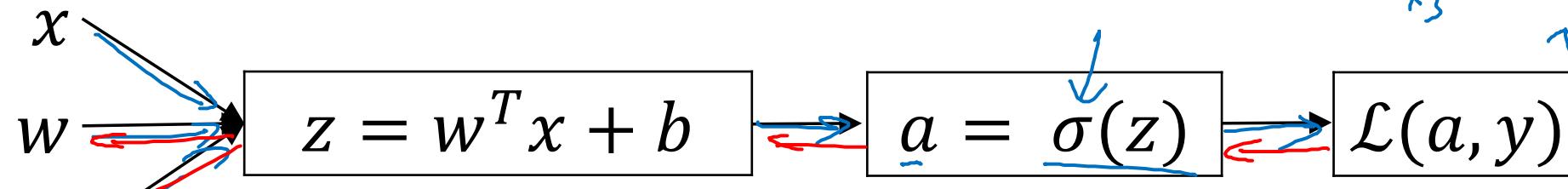
deeplearning.ai

One hidden layer
Neural Network

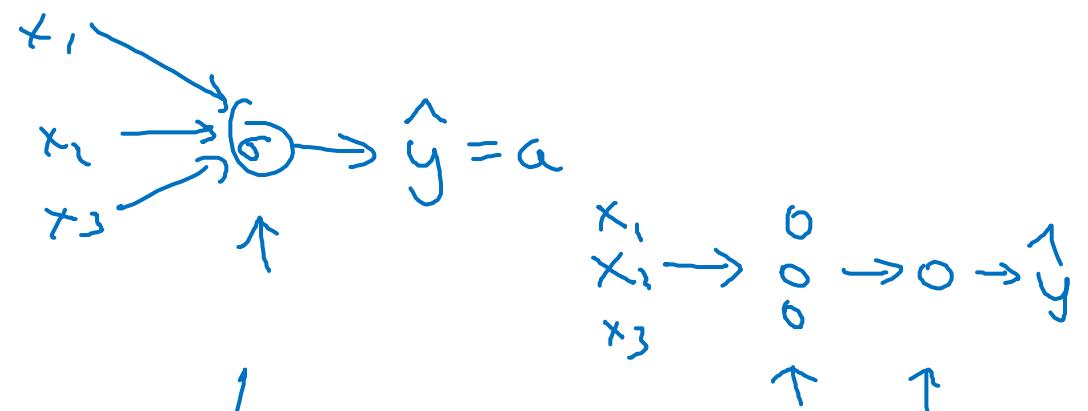
Backpropagation
intuition (Optional)

Computing gradients

Logistic regression



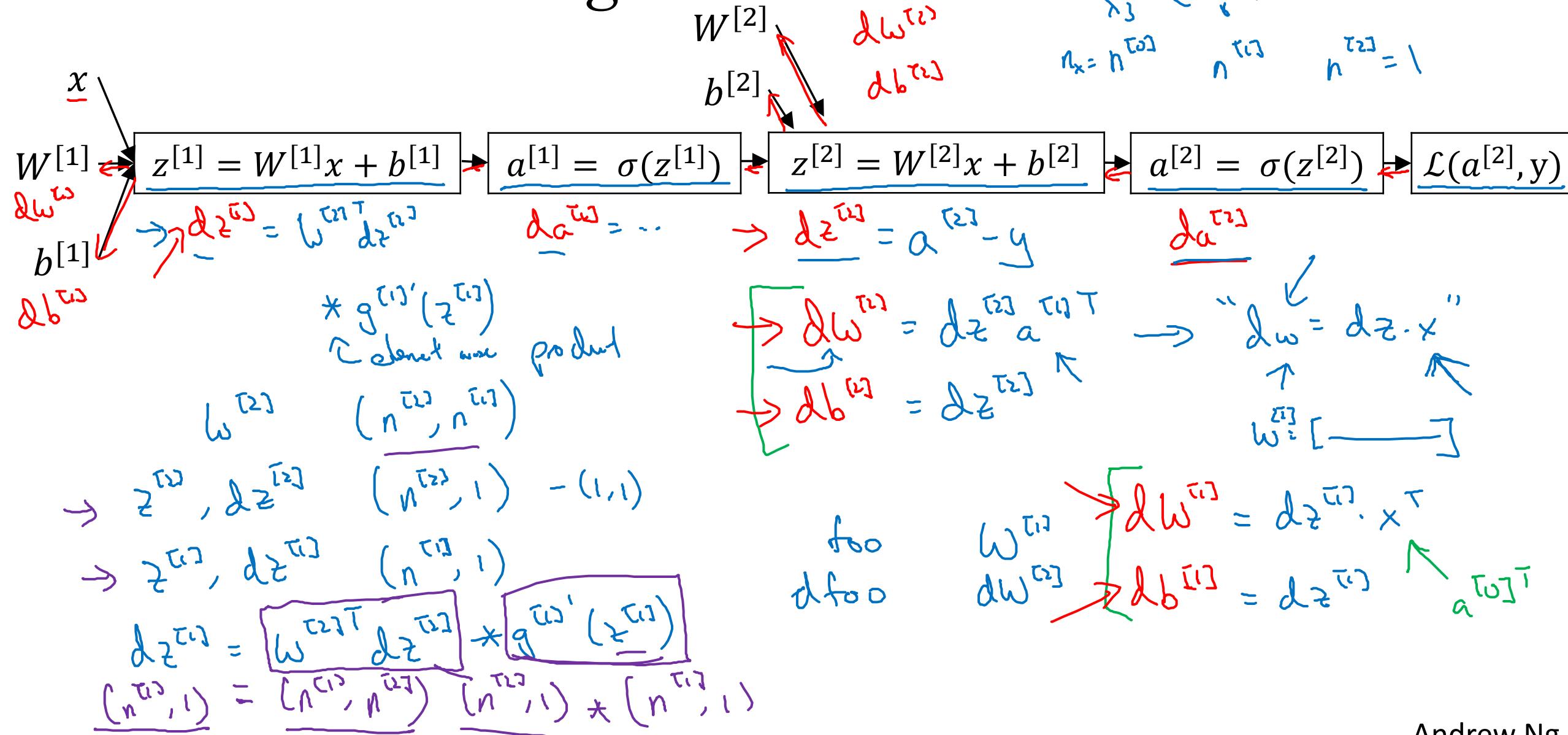
$$\begin{aligned} \frac{\partial z}{\partial w} &= x \\ \frac{\partial z}{\partial b} &= 1 \\ \frac{\partial z}{\partial a} &= g'(z) \\ g(z) &= \sigma(z) \\ \frac{\partial \mathcal{L}}{\partial z} &= \frac{\partial \mathcal{L}}{\partial a} \cdot \frac{\partial a}{\partial z} \\ "dz" &= "da" \end{aligned}$$



$$\begin{aligned} \frac{\partial \mathcal{L}}{\partial a} &= \frac{d}{da} \mathcal{L}(a, y) = -y \log a - (1-y) \log(1-a) \\ &= -\frac{y}{a} + \frac{1-y}{1-a} \end{aligned}$$

$$\frac{d}{dz} g(z) = g'(z)$$

Neural network gradients



Summary of gradient descent

$$dz^{[2]} = a^{[2]} - y$$

$$dW^{[2]} = dz^{[2]} a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

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

$$dW^{[1]} = dz^{[1]} x^T$$

$$db^{[1]} = dz^{[1]}$$

Vectorized implementation:

$$\begin{aligned} z^{[1]} &= \underbrace{w^{[1]} x + b^{[1]}}_{\text{Implementation}} \\ a^{[1]} &= g^{[1]}(z^{[1]}) \end{aligned}$$

$$z^{[1]} = \begin{bmatrix} 1 & z^{1} & z^{[1](2)} & \dots & z^{[1](n)} \\ | & | & | & \dots & | \end{bmatrix}$$

$$\begin{aligned} z^{[1]} &= w^{[1]} x + b^{[1]} \\ A^{[1]} &= g^{[1]}(z^{[1]}) \end{aligned}$$

Summary of gradient descent

$$\underline{dz}^{[2]} = \underline{a}^{[2]} - \underline{y}$$

$$dW^{[2]} = dz^{[2]} a^{[1]T}$$

$$db^{[2]} = dz^{[2]}$$

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

(n^{T₁}, 1)

$$dW^{[1]} = dz^{[1]} X^T$$

$$db^{[1]} = dz^{[1]}$$

$$\underline{dZ}^{[2]} = \underline{A}^{[2]} - \underline{Y}$$

$$dW^{[2]} = \frac{1}{m} dZ^{[2]} A^{[1]T}$$

$$db^{[2]} = \frac{1}{m} np.sum(dZ^{[2]}, axis = 1, keepdims = True)$$

$$dZ^{[1]} = \underbrace{W^{[2]T} dZ^{[2]}}_{(n^{T₂}, m)} * \underbrace{g^{[1]'}(Z^{[1]})}_{(n^{T₁}, m)}$$

elementwise product

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

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

$$J(\cdot) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}_i, y_i)$$

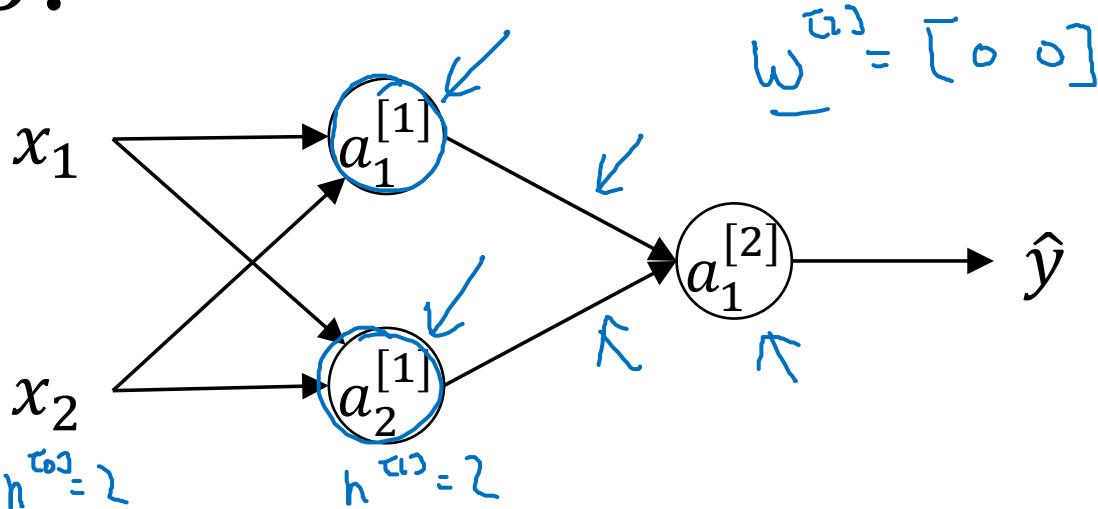


deeplearning.ai

One hidden layer
Neural Network

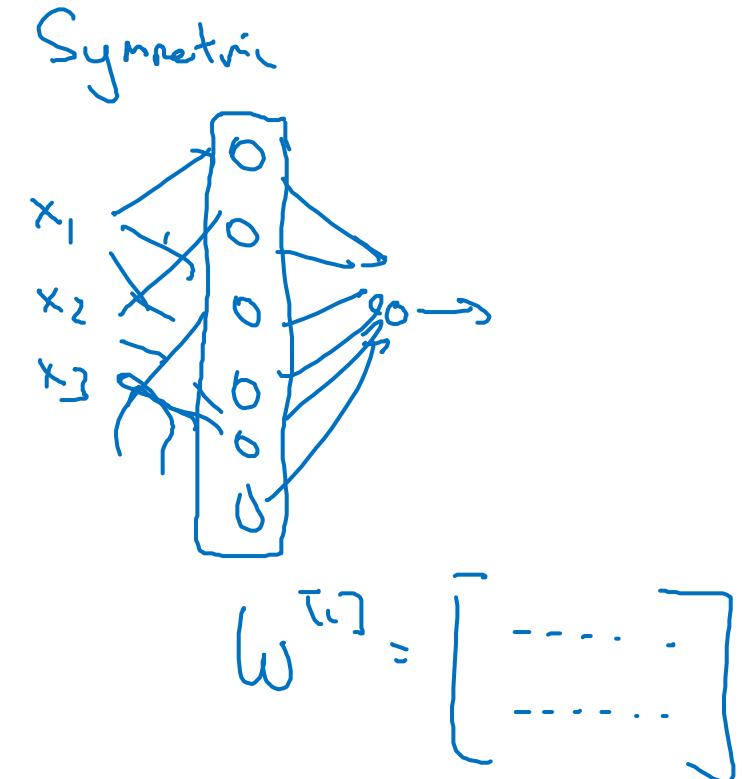
Random Initialization

What happens if you initialize weights to zero?

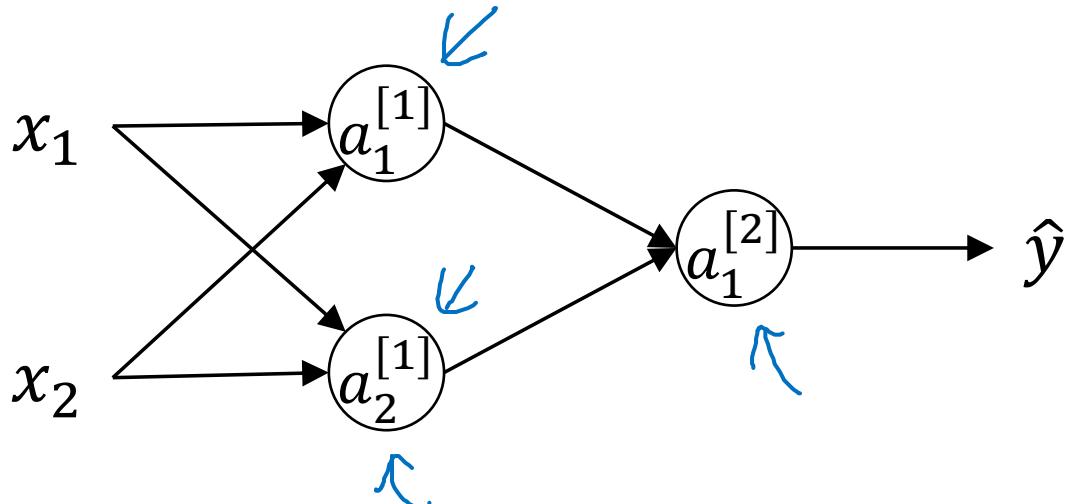


$$\Delta w = \begin{bmatrix} u & v \\ u & v \end{bmatrix}$$

$$w^0 = w^{\text{initial}} - \lambda \Delta w$$

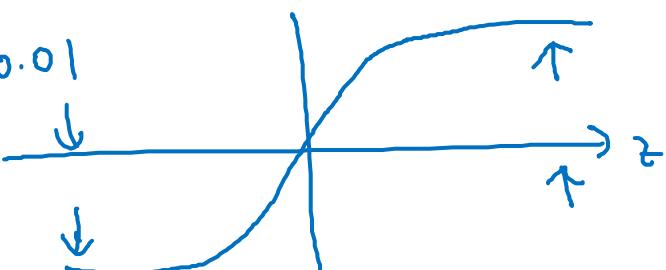


Random initialization



$$\begin{aligned} \rightarrow w^{[1]} &= \text{np.random.randn}(2, 2) \times \frac{0.01}{100?} \\ b^{[1]} &= \text{np.zeros}(2, 1) \\ w^{[2]} &= \text{np.random.randn}(1, 2) \times 0.01 \\ b^{[2]} &= 0 \end{aligned}$$

$$\begin{aligned} z^{[1]} &= w^{[1]} \times + b^{[1]} \\ a^{[1]} &= g^{[1]}(z^{[1]}) \end{aligned}$$



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>

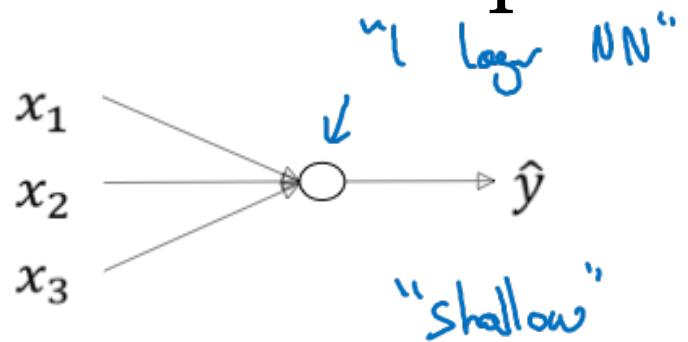


deeplearning.ai

Deep Neural Networks

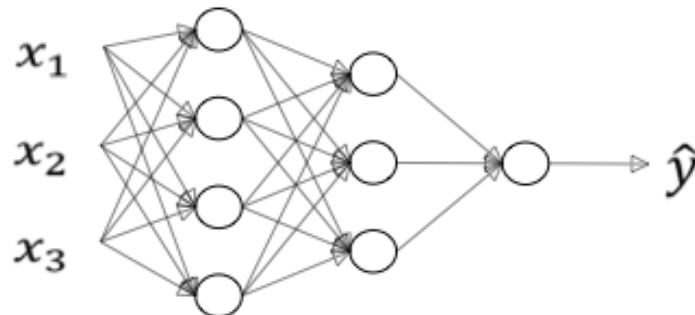
Deep L-layer Neural network

What is a deep neural network?

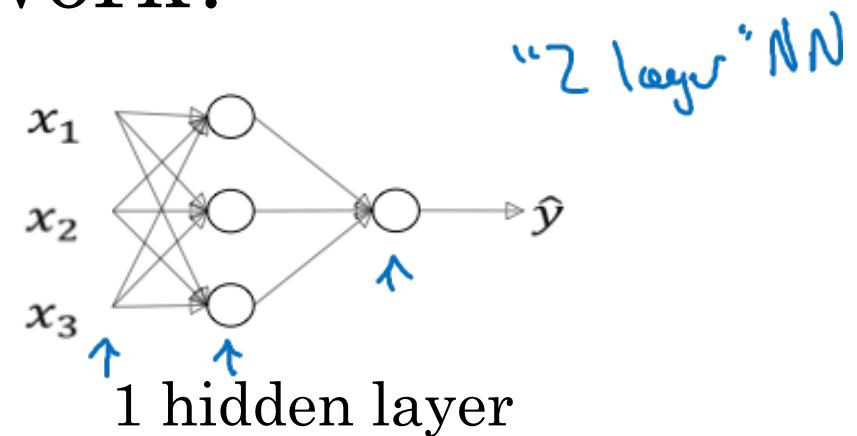


"Shallow"

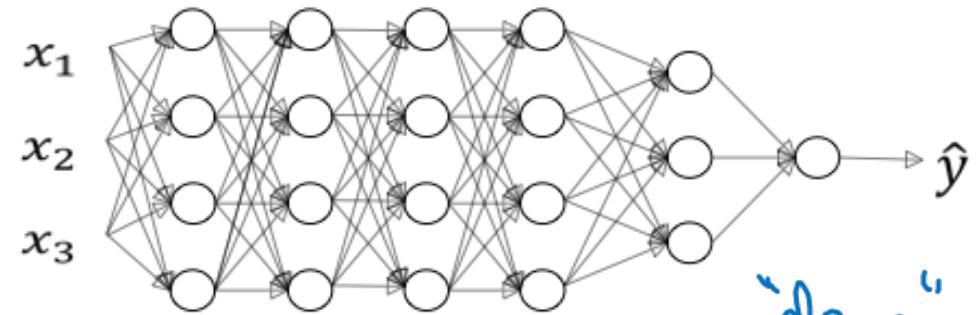
logistic regression



2 hidden layers



1 hidden layer

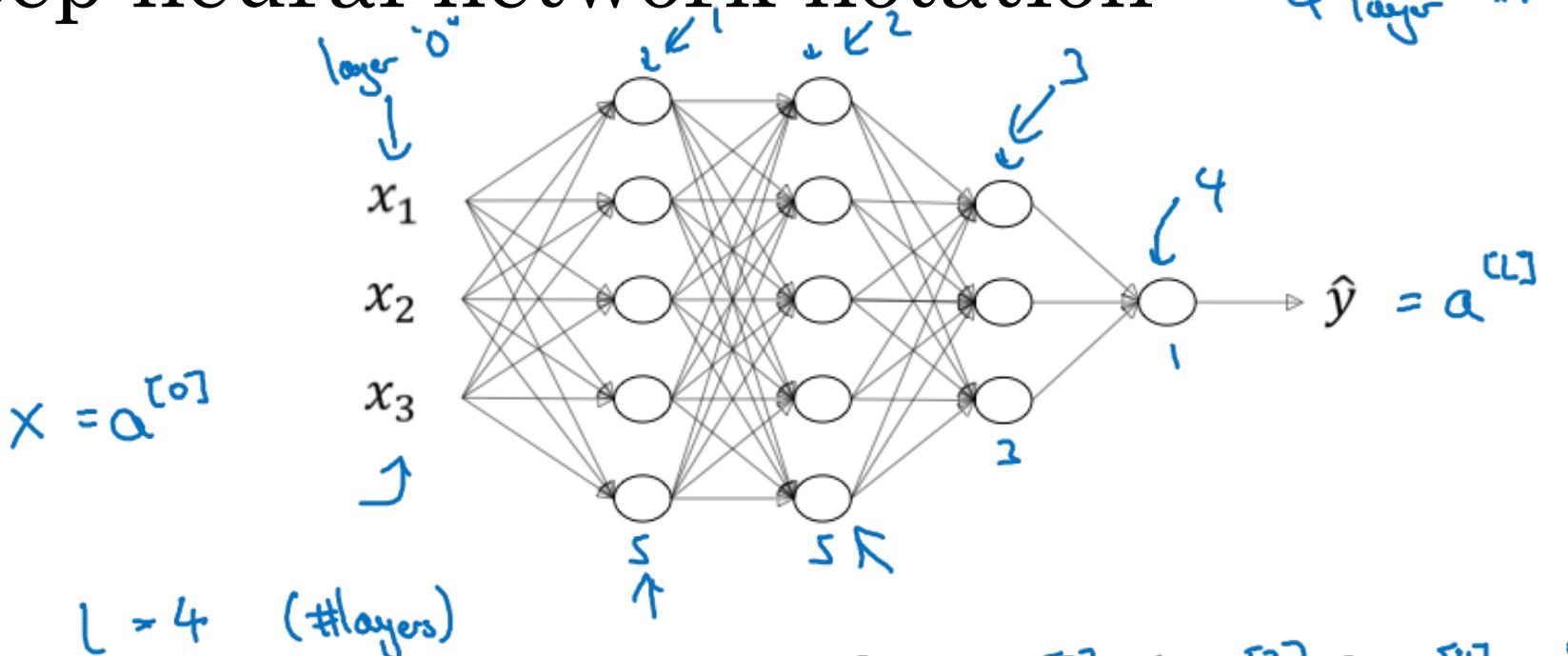


5 hidden layers

"deep"

Andrew
Ng

Deep neural network notation



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

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

Andrew
Ng

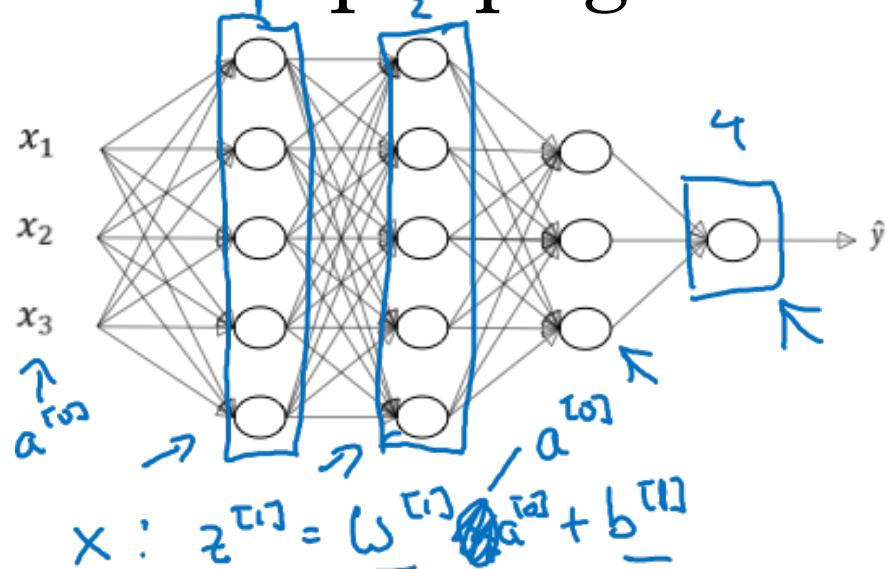


deeplearning.ai

Deep Neural Networks

Forward Propagation in a Deep Network

Forward propagation in a deep network



$A^{[0]} = X$

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

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

Vertikal:

$\rightarrow z^{[1]} = (W^{[1]} \otimes A^{[0]} + b^{[1]})$

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

$\rightarrow z^{[2]} = (W^{[2]} \otimes A^{[1]} + b^{[2]})$

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

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

for $l=1..4$

Andrew



deeplearning.ai

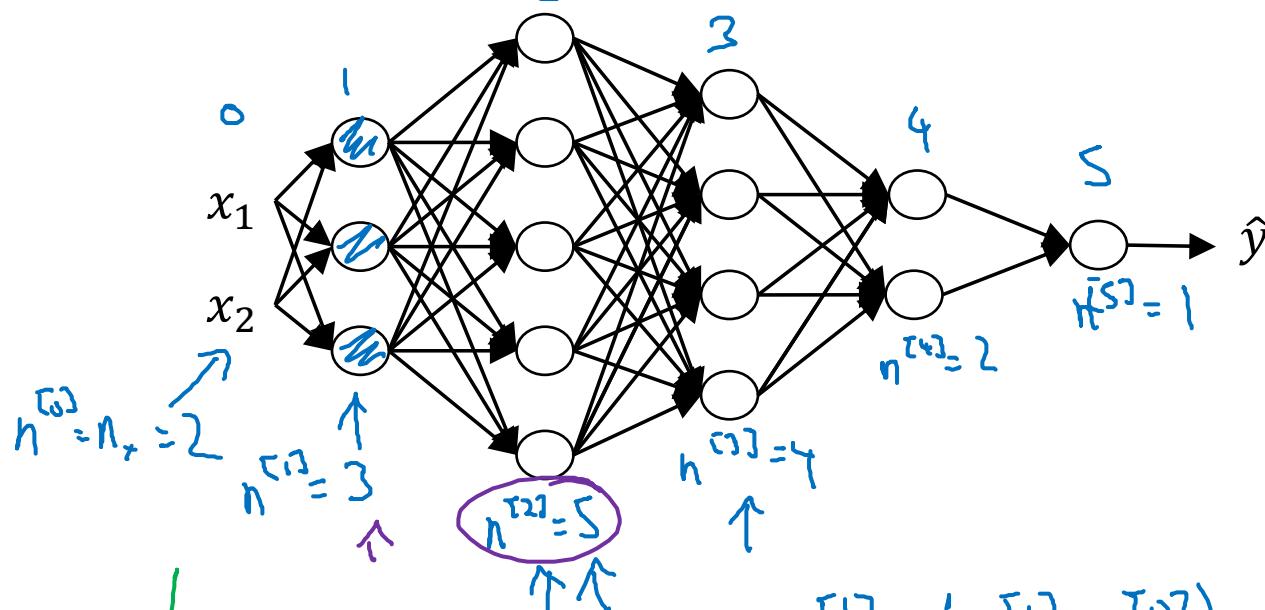
Deep Neural Networks

Getting your matrix dimensions right

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

$L = 5$

$$\begin{array}{c} \downarrow \\ z^{[0]} = g^{[0]}(a^{[0]}) \\ \uparrow \\ \theta^{[0]} \end{array}$$



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

$$[\vdots] = [\vdots \vdots] \quad [\vdots]$$

$$W^{[1]}: (n^{[1]}, n^{[0]})$$

$$W^{[2]}: (5, 3) \quad (n^{[2]}, n^{[1]})$$

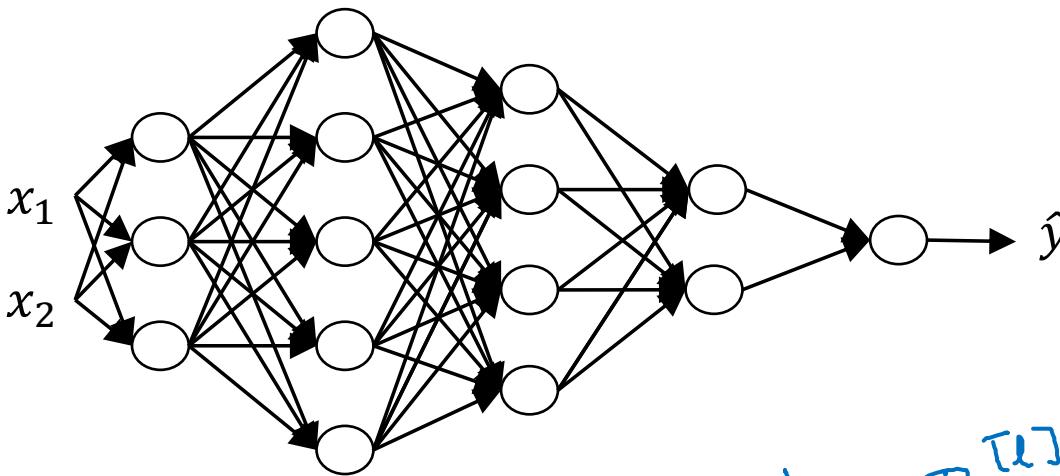
$$\begin{array}{c} z^{[2]} = \boxed{\underline{W^{[2]}} \cdot \underline{a^{[1]}}} + \boxed{\underline{b^{[2]}}} \\ \uparrow \quad \uparrow \quad \uparrow \quad \uparrow \\ (5,1) \quad (5,3) \quad (2,1) \quad (5,1) \\ (\underline{n^{[2]}}, 1) \quad (\underline{n^{[1]}}, \underline{n^{[3]}}) \quad (\underline{n^{[3]}}, 1) \end{array}$$

$$W^{[3]}: (4, 5)$$

$$W^{[4]}: (2, 4), \quad W^{[5]}: (1, 2)$$

$$\begin{cases} W^{[l]}: (n^{[l]}, n^{[l-1]}) \\ b^{[l]}: (n^{[l]}, 1) \\ \delta W^{[l]}: (n^{[l]}, n^{[l-1]}) \\ \delta b^{[l]}: (n^{[l]}, 1) \end{cases}$$

Vectorized implementation



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

$$(n^{[l]}, 1) \quad (n^{[l]}, n^{[l+1]}) \quad (n^{[l]}, 1)$$

$$\begin{bmatrix} z^{1} & z^{[1](2)} & \dots & z^{[1](m)} \end{bmatrix}$$

$$\sum^{[l]} = w^{[l]} \cdot X + b^{[l]}$$

$$\underbrace{(n^{[l]}, m)}_{\uparrow} \quad \underbrace{(n^{[l]}, n^{[l+1]})}_{\uparrow} \quad \underbrace{(n^{[l]}, m)}_{\uparrow} \quad \underbrace{(n^{[l]}, 1)}_{(n^{[l]}, m)}$$

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

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

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

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

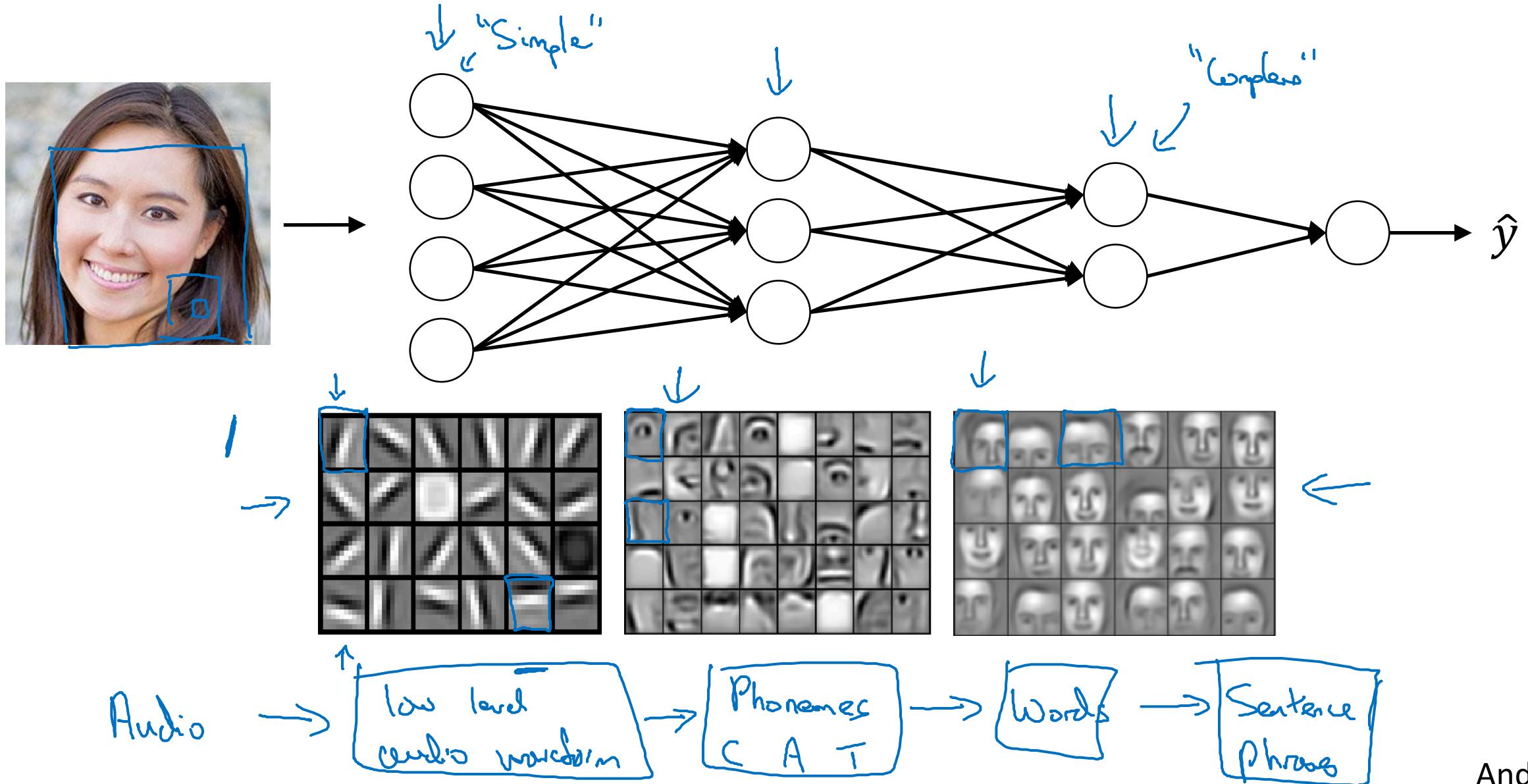


deeplearning.ai

Deep Neural Networks

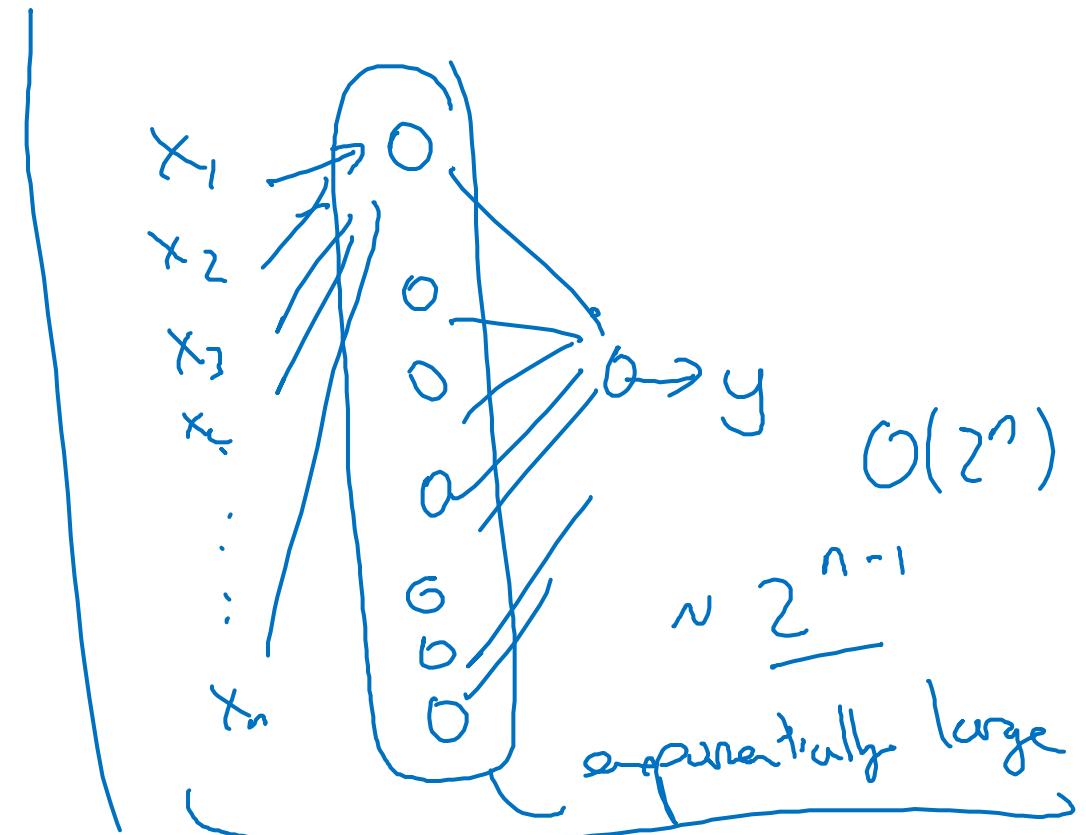
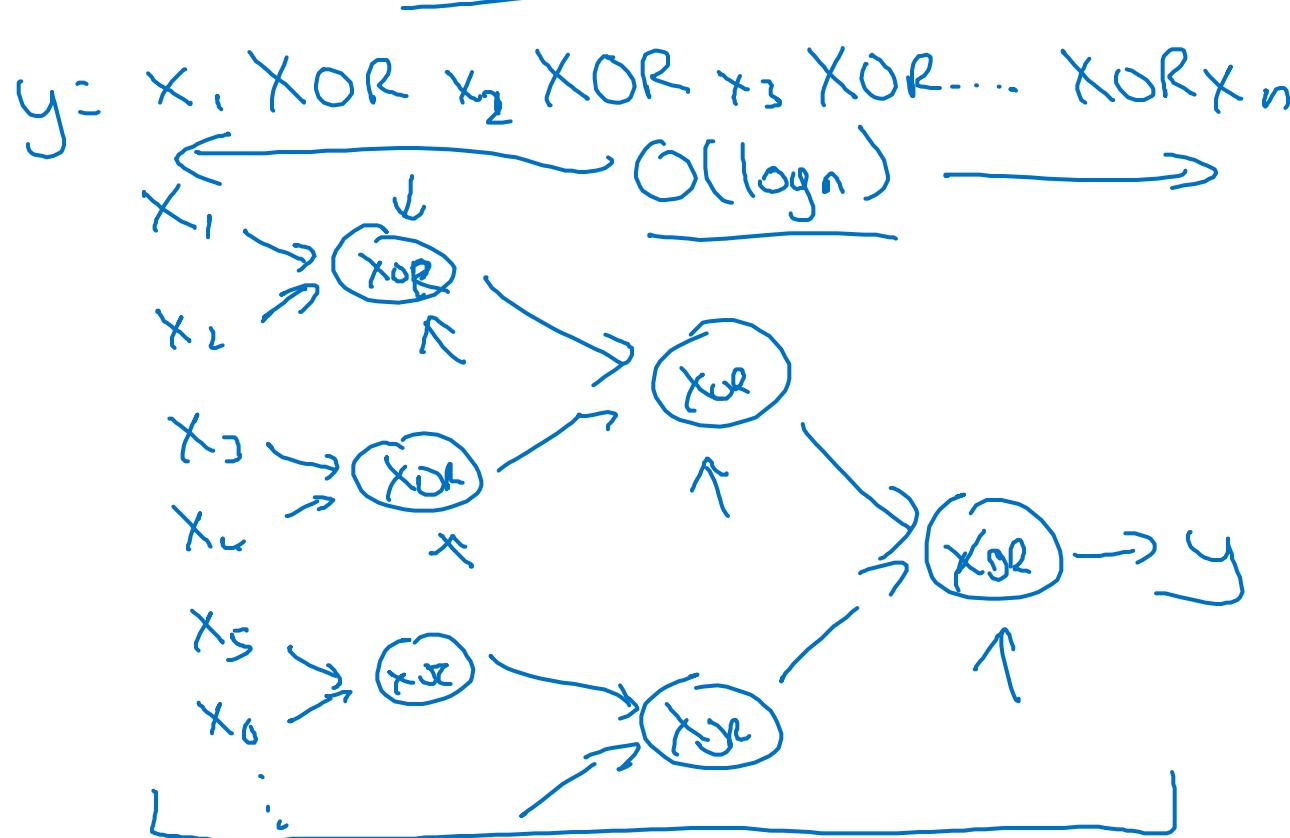
Why deep representations?

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.



Andrew Ng

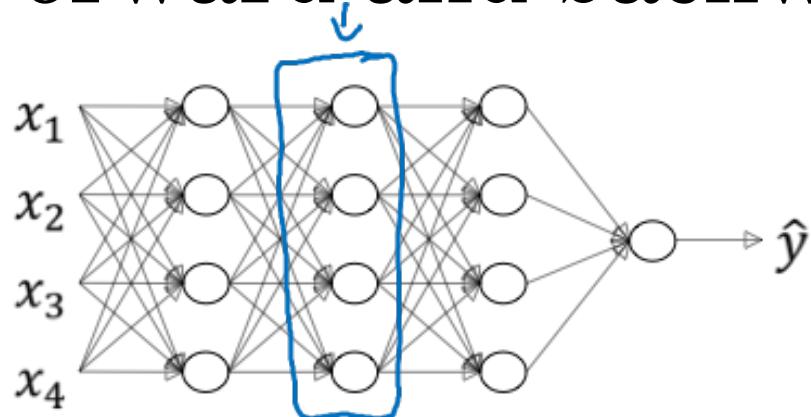


deeplearning.ai

Deep Neural Networks

Building blocks of
deep neural networks

Forward and backward functions



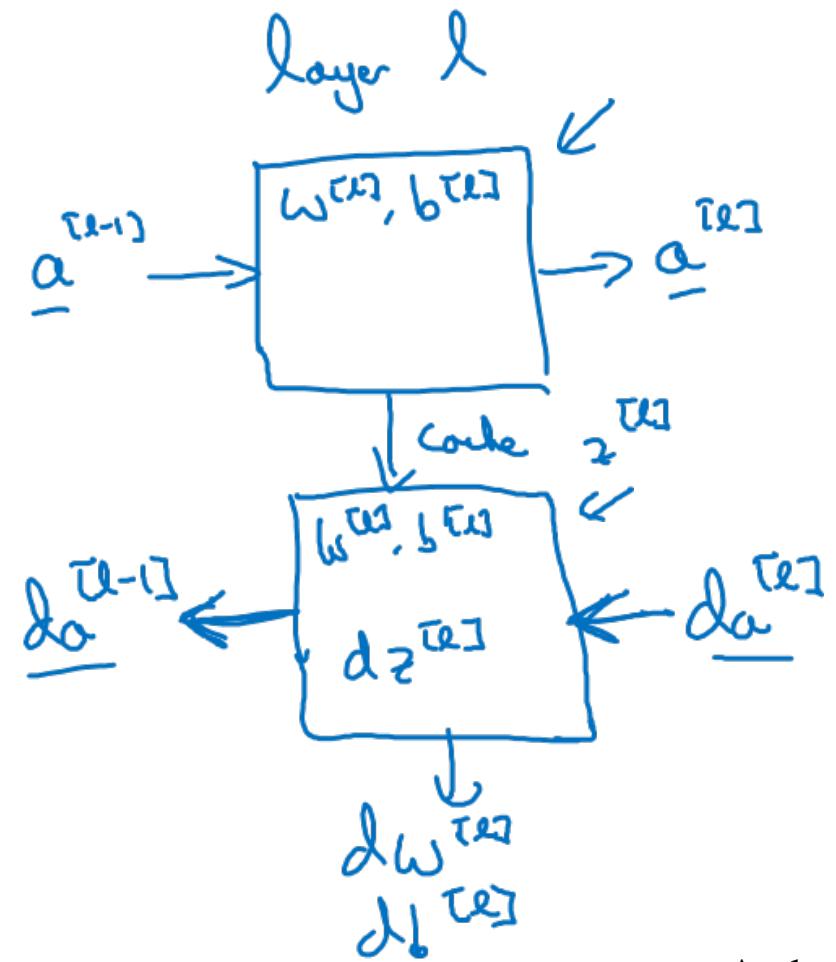
layer l : $w^{[l]}, b^{[l]}$

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

$$\underline{z}^{[l]} = (w^{[l]} \underline{a}^{[l-1]} + b^{[l]}) \text{ cache } \underline{z}^{[l]}$$

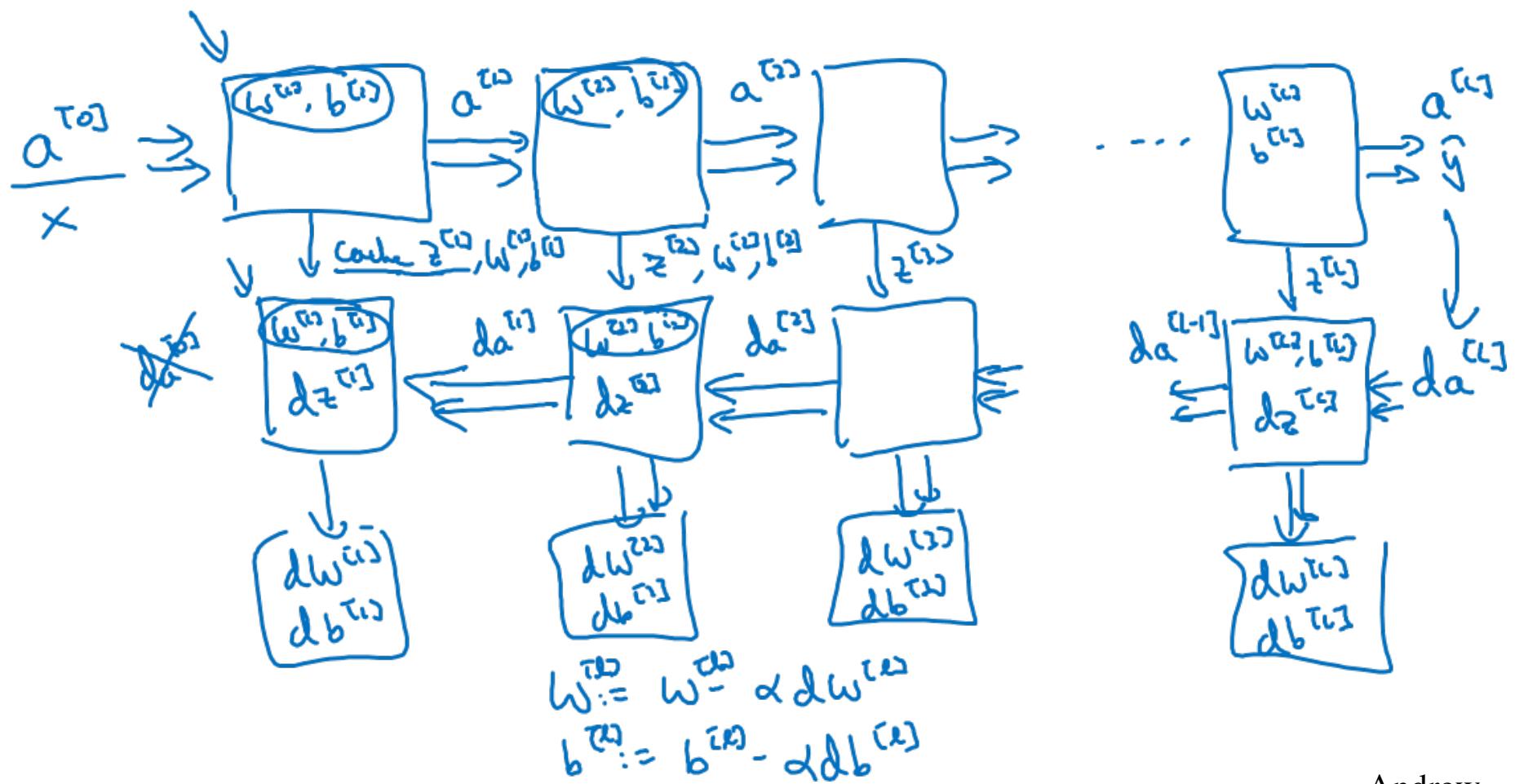
$$\underline{a}^{[l]} = g^{[l]}(\underline{z}^{[l]})$$

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



Andrew
Ng

Forward and backward functions



Andrew
Ng



deeplearning.ai

Deep Neural Networks

Forward and backward propagation

Backward propagation for layer l

→ Input $da^{[l]}$

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

$$dz^{[l]} = \underbrace{da^{[l]}}_{\text{green}} * g^{[l]}(z^{[l]})$$

$$dW^{[l]} = dz^{[l]} \cdot \underbrace{a^{[l-1]}}_{\text{blue}}$$

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

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

$$dz^{[l-1]} = \underbrace{W^{[l]T}}_{\text{green}} \underbrace{dz^{[l]}}_{\text{green}} * g^{[l-1]}(z^{[l]})$$

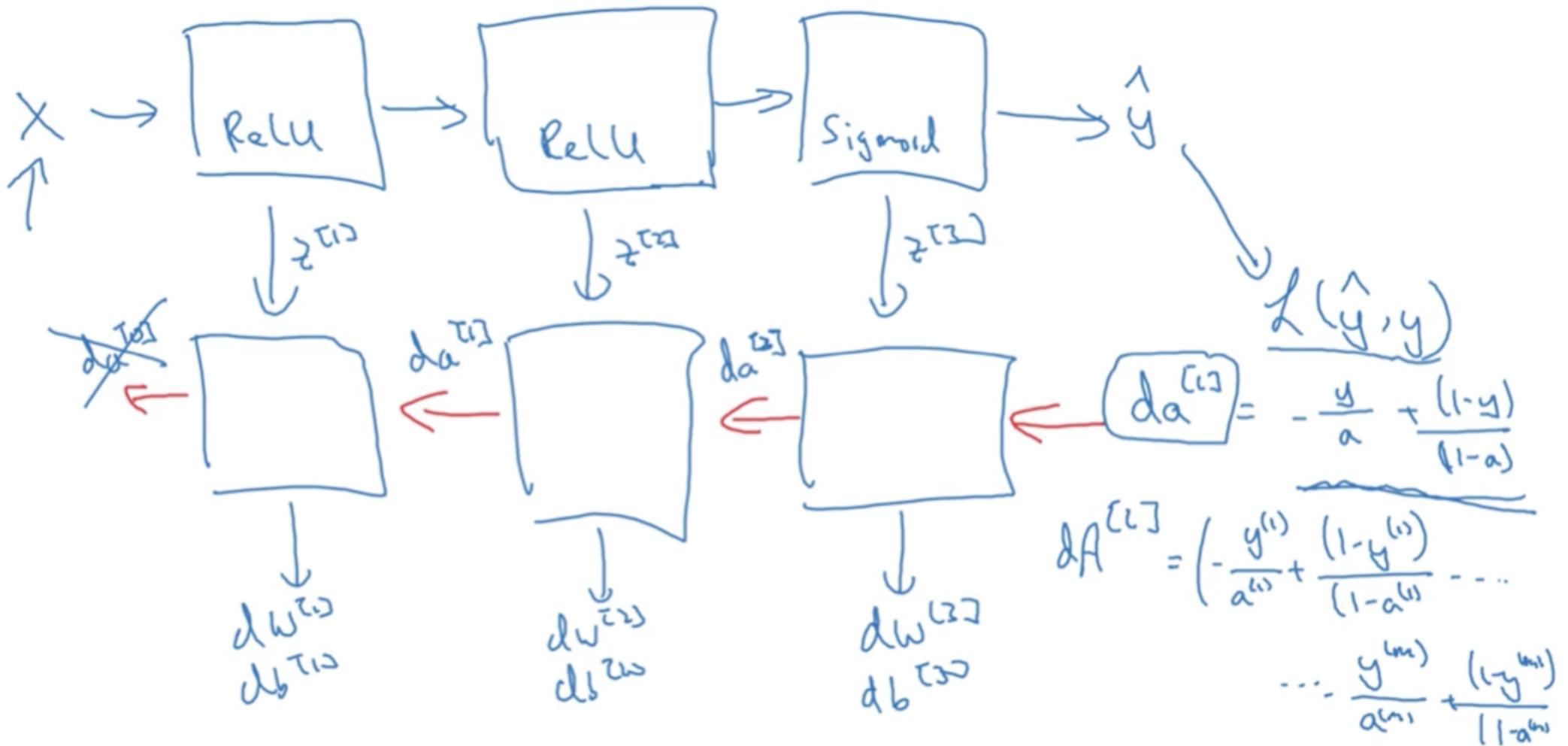
$$dz^{[l]} = dA^{[l]} * g^{[l]}(z^{[l]})$$

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

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

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

Summary





deeplearning.ai

Deep Neural Networks

Parameters vs Hyperparameters

What are hyperparameters?

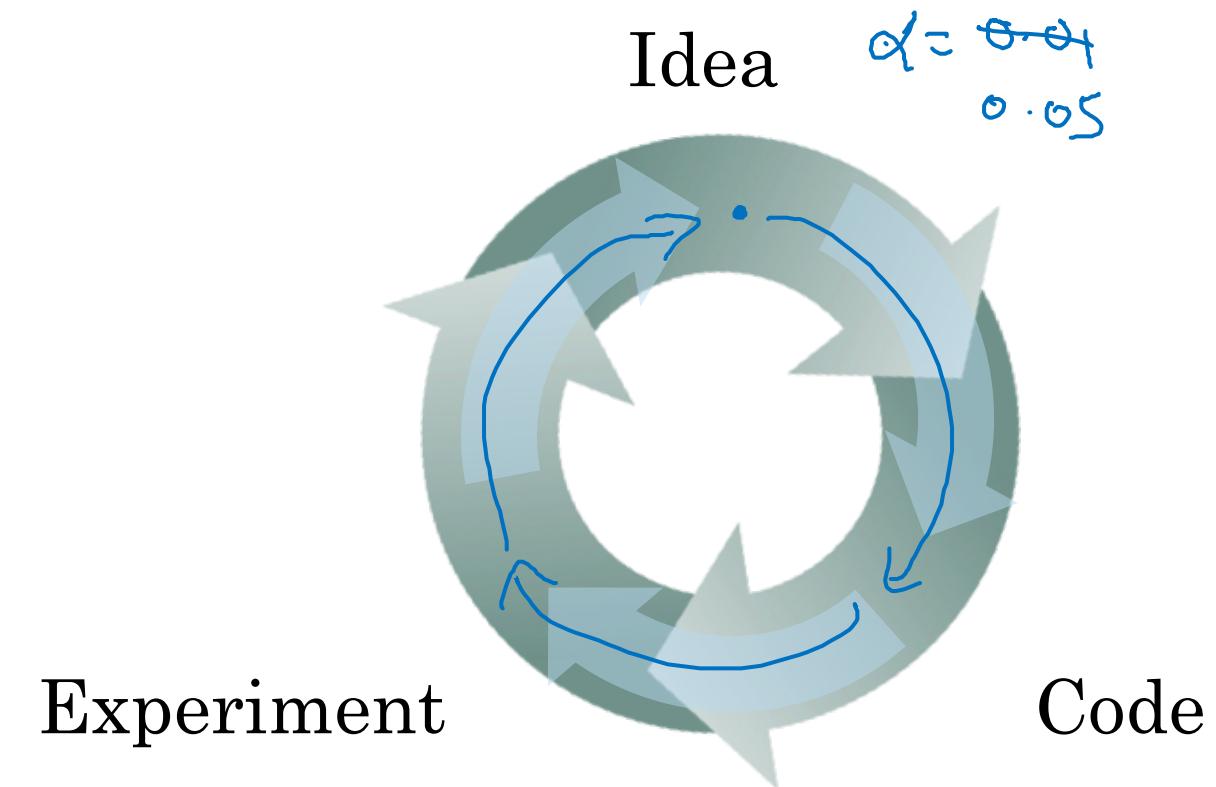
Parameters: $W^{[1]}, b^{[1]}, W^{[2]}, b^{[2]}, W^{[3]}, b^{[3]} \dots$

Hyperparameters:

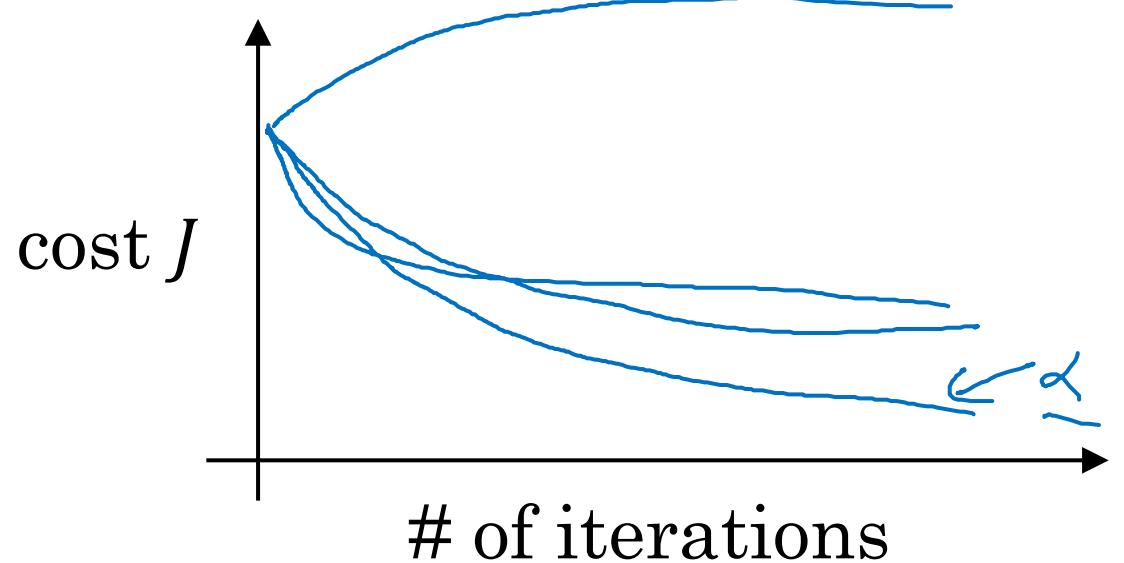
- learning rate $\frac{\alpha}{n}$
- #iterations
- #hidden layers L
- #hidden units $n^{[1]}, n^{[2]}, \dots$
- choice of activation function

Later: Momentum, minibatch size, regularizations, ...

Applied deep learning is a very empirical process



Vision, Speech, NLP, Ad, Search, Reinforcement.





deeplearning.ai

Deep Neural Networks

What does this
have to do with
the brain?

Forward and backward propagation

$$Z^{[1]} = W^{[1]}X + b^{[1]}$$

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

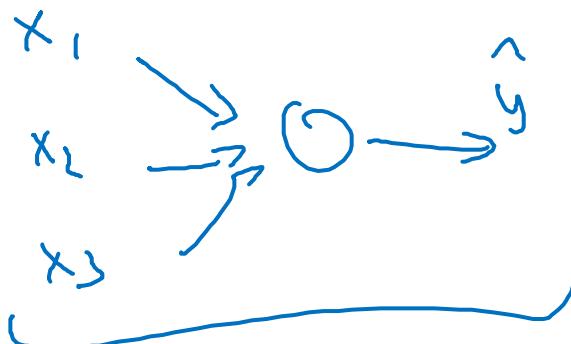
$$Z^{[2]} = W^{[2]}A^{[1]} + b^{[2]}$$

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

:

$$A^{[L]} = g^{[L]}(Z^{[L]}) = \hat{Y}$$

"It's like the brain"



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

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

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

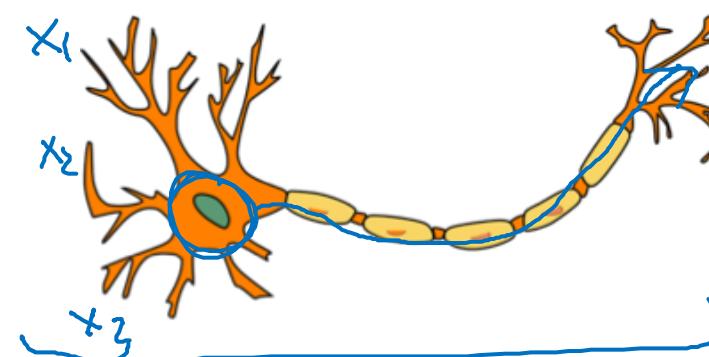
$$dZ^{[L-1]} = dW^{[L]T} dZ^{[L]} g'^{[L]}(Z^{[L-1]})$$

$$\vdots$$

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

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

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



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>

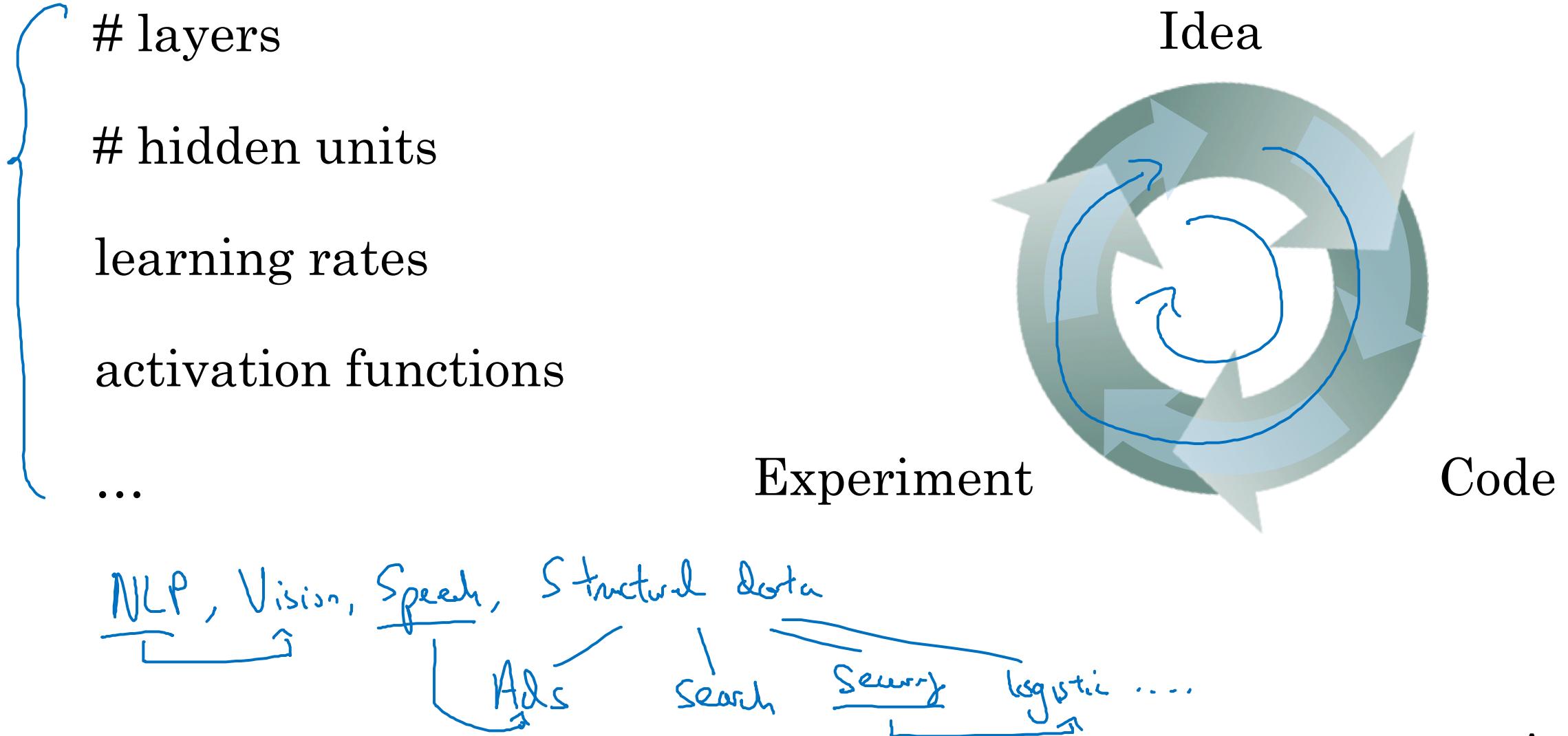


deeplearning.ai

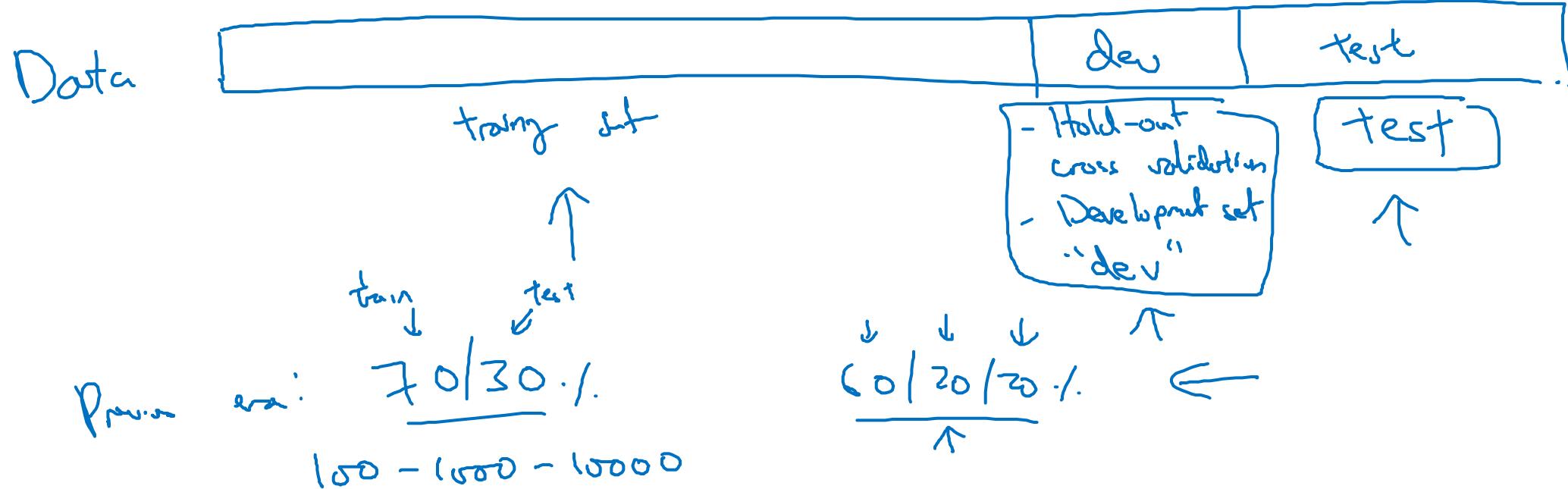
Setting up your
ML application

Train/dev/test
sets

Applied ML is a highly iterative process



Train/dev/test sets



Big data! 1,000,000

10,000 10,000

98 / 1 / 1 ./.
99.5 { 25 / 25
 { 4 { -1 ./.

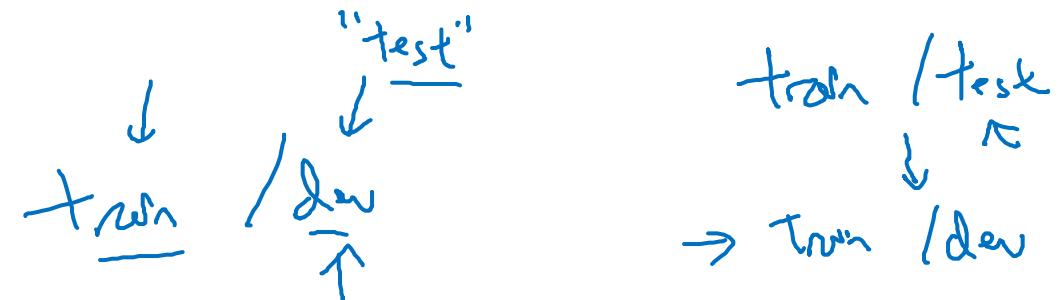
Mismatched train/test distribution

Conts

Training set:
Cat pictures from }
webpages

Dev/test sets:
Cat pictures from }
users using your app

→ Make sure dev and test come from same distribution.



Not having a test set might be okay. (Only dev set.)

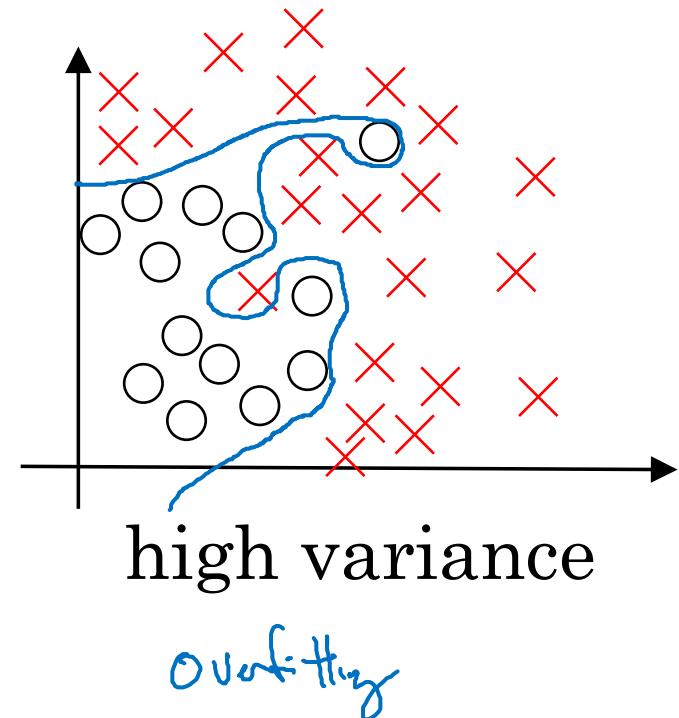
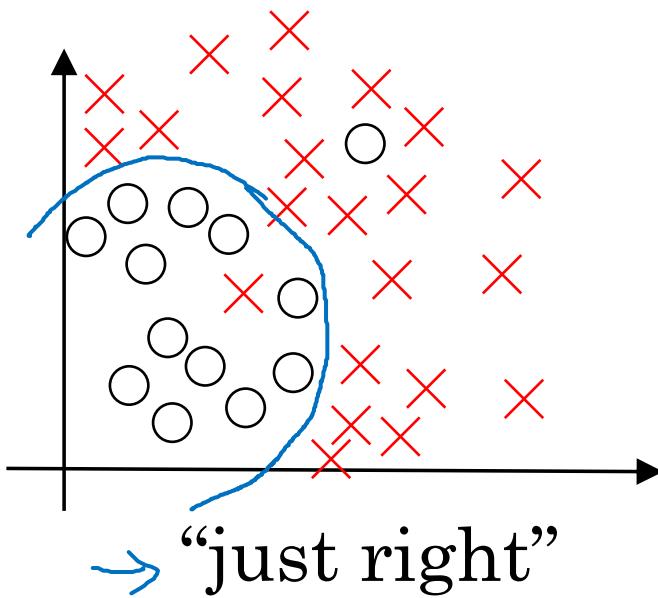
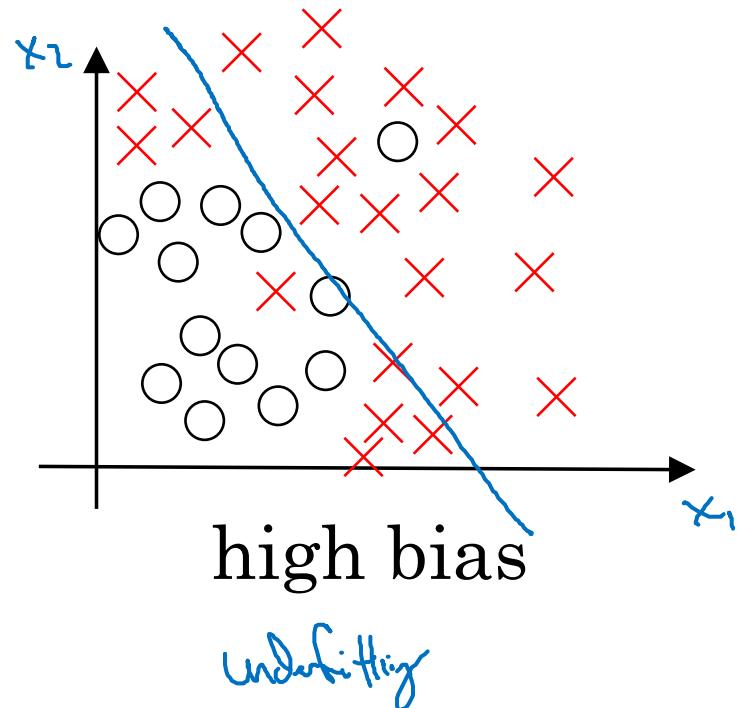


deeplearning.ai

Setting up your
ML application

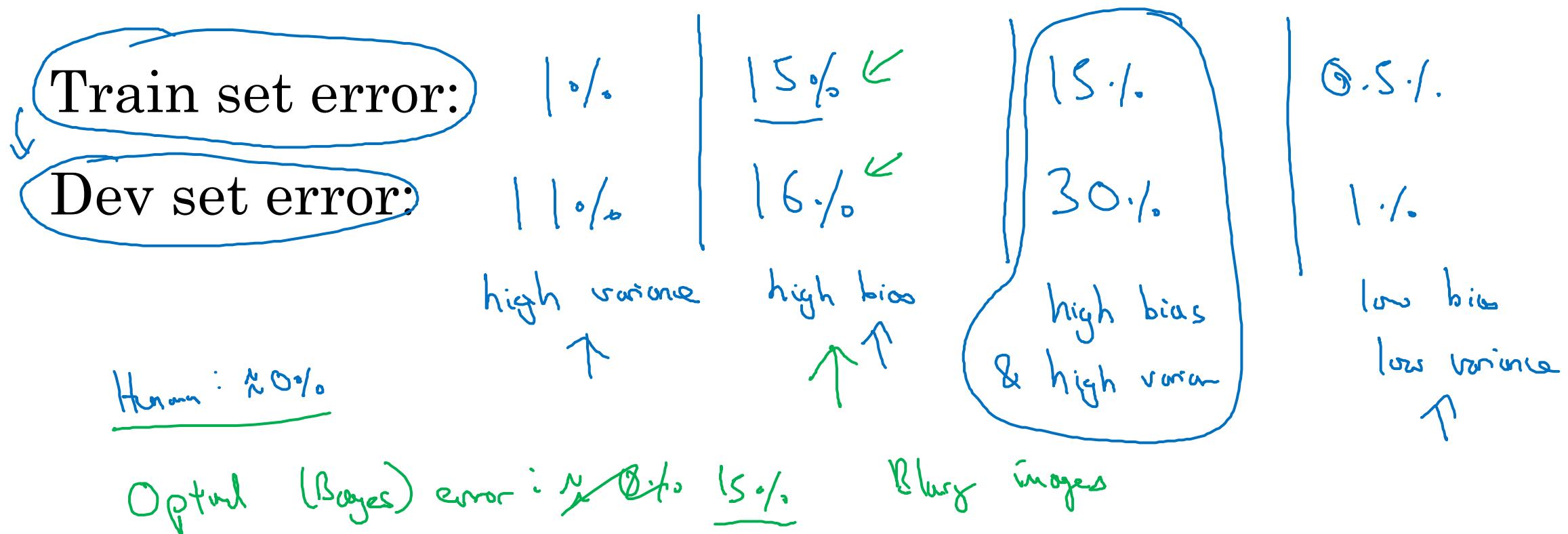
Bias/Variance

Bias and Variance

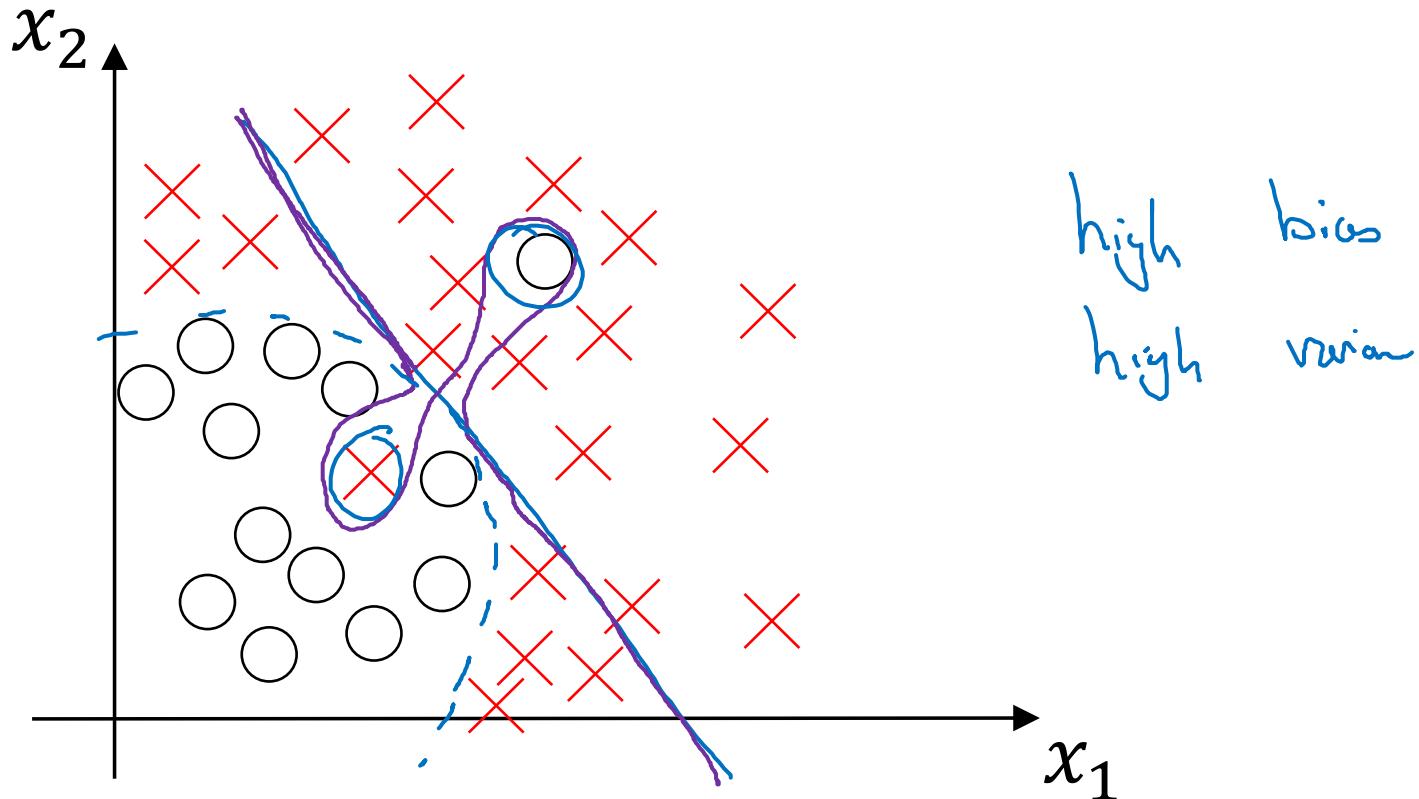


Bias and Variance

Cat classification



High bias and high variance



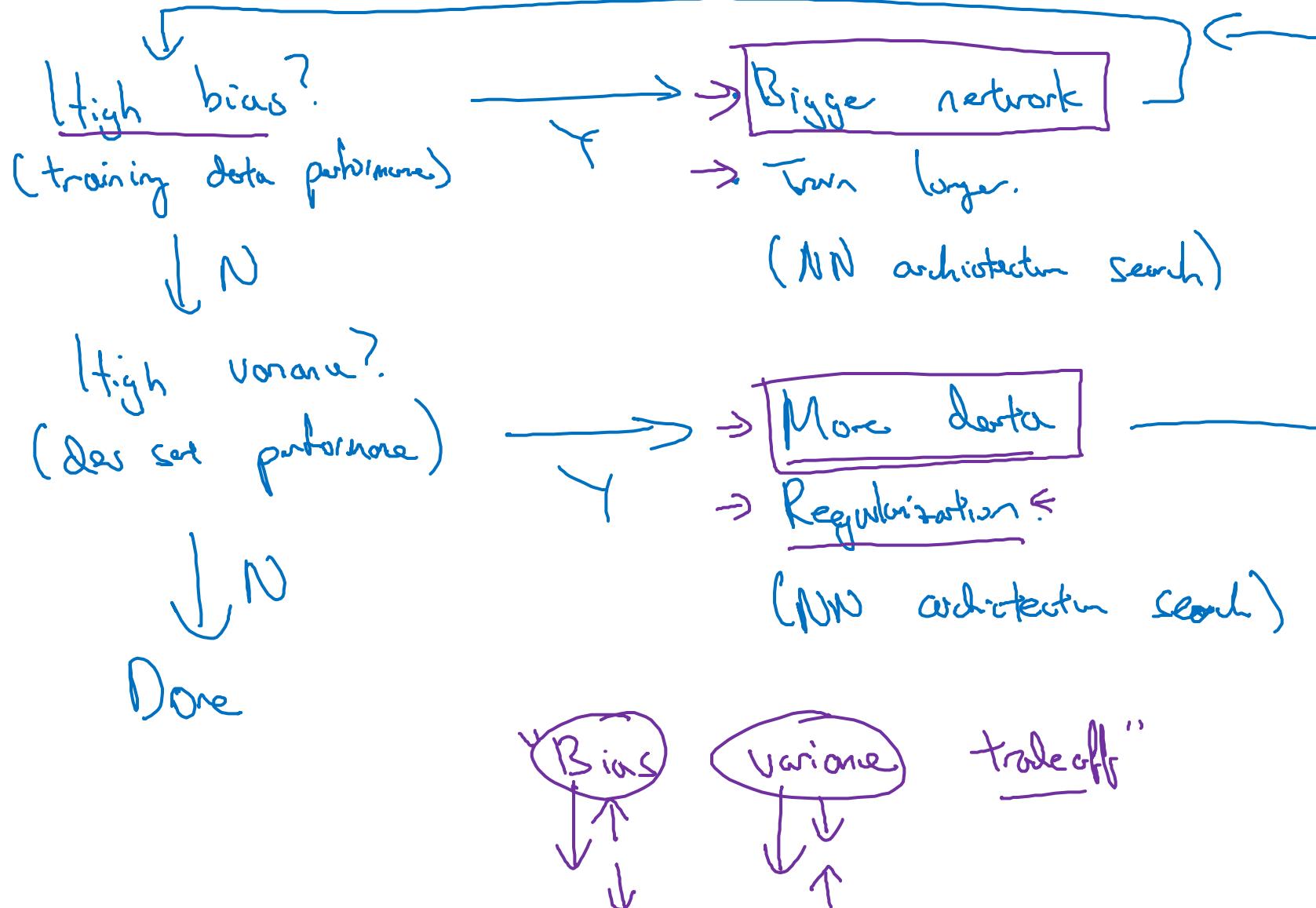


deeplearning.ai

Setting up your ML application

Basic “recipe” for machine learning

Basic recipe for machine learning





deeplearning.ai

Regularizing your
neural network

Regularization

Logistic regression

$$\min_{w,b} J(w, b)$$

$$w \in \mathbb{R}^{n_x}, b \in \mathbb{R}$$

λ = regularization parameter
lambda lambd

$$J(w, b) = \underbrace{\frac{1}{m} \sum_{i=1}^m \ell(\hat{y}^{(i)}, y^{(i)})}_{\text{L}_2 \text{ regularization}} + \frac{\lambda}{2m} \|w\|_2^2$$

$$+ \cancel{\frac{\lambda}{2m} b^2}$$

omit

$$\|w\|_2^2 = \sum_{j=1}^{n_x} w_j^2 = w^T w \leftarrow$$

L₁ regularization

$$\frac{\lambda}{2m} \sum_{j=1}^{n_x} |w_j| = \frac{\lambda}{2m} \|w\|_1$$

w will be sparse

Neural network

$$\rightarrow J(w^{(1)}, b^{(1)}, \dots, w^{(L)}, b^{(L)}) = \underbrace{\frac{1}{m} \sum_{i=1}^m f(\hat{y}^{(i)}, y^{(i)})}_{\text{loss function}} + \underbrace{\frac{\lambda}{2m} \sum_{l=1}^L \|w^{(l)}\|_F^2}_{\text{regularization}}$$

$$\|w^{(l)}\|_F^2 = \sum_{i=1}^{n^{(l)}} \sum_{j=1}^{n^{(l+1)}} (w_{ij}^{(l)})^2$$

$w^{(l)}: (n^{(l)}, n^{(l+1)})$

"Frobenius norm"

$$\|\cdot\|_2^2$$

$$\|\cdot\|_F^2$$

$$dW^{(l)} = \boxed{(\text{from backprop}) + \frac{\lambda}{m} w^{(l)}}$$

$$\rightarrow w^{(l)} := w^{(l)} - \alpha dW^{(l)}$$

$$\frac{\partial J}{\partial w^{(l)}} = dw^{(l)}$$

"Weight decay"

$$w^{(l)} := w^{(l)} - \alpha \left[(\text{from backprop}) + \frac{\lambda}{m} w^{(l)} \right]$$

$$= w^{(l)} - \frac{\alpha \lambda}{m} w^{(l)} - \alpha (\text{from backprop})$$

$$= \underbrace{\left(1 - \frac{\alpha \lambda}{m}\right)}_{< 1} \underbrace{w^{(l)}}_{> 0} - \alpha (\text{from backprop})$$

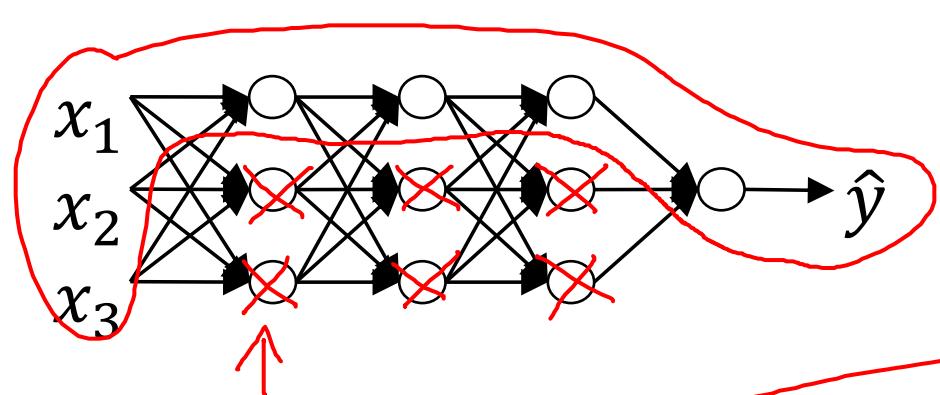


deeplearning.ai

Regularizing your neural network

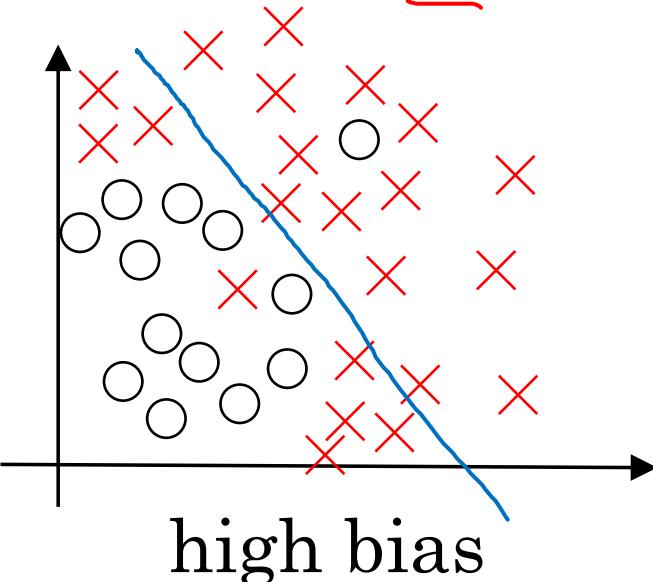
Why regularization reduces overfitting

How does regularization prevent overfitting?

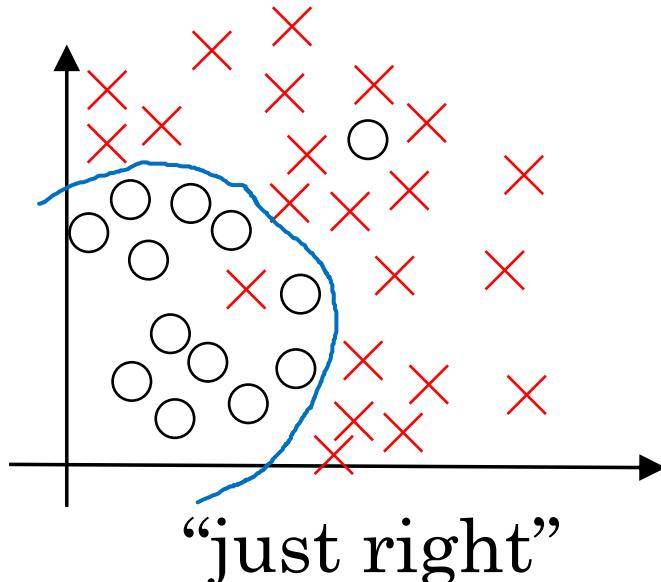


$$J(\boldsymbol{\theta}^{(m)}, \boldsymbol{b}^{(m)}) = \frac{1}{m} \sum_{i=1}^m \ell(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m} \sum_{l=1}^L \|\boldsymbol{w}^{(l)}\|_F^2$$

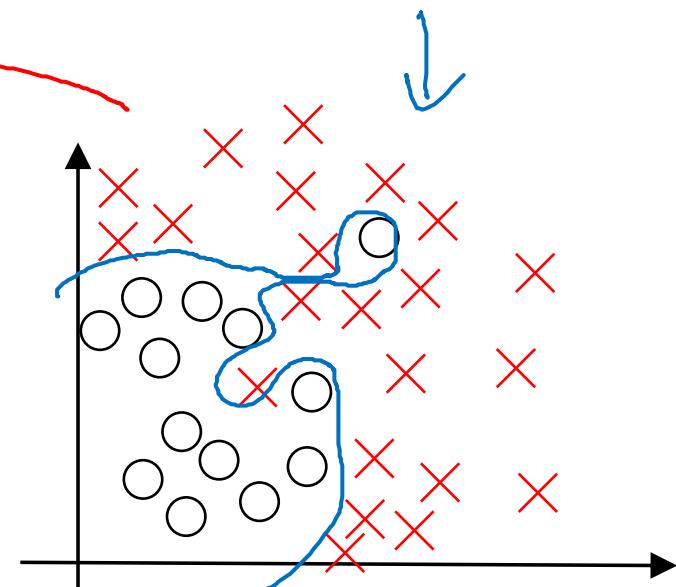
$\boldsymbol{w}^{(l)} \approx 0$



high bias

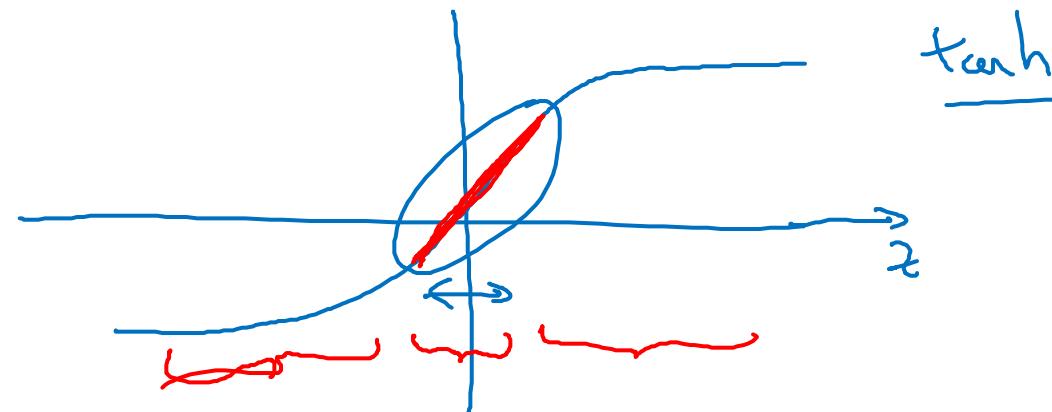


"just right"



high variance

How does regularization prevent overfitting?



$$g(z) = \tanh(z)$$

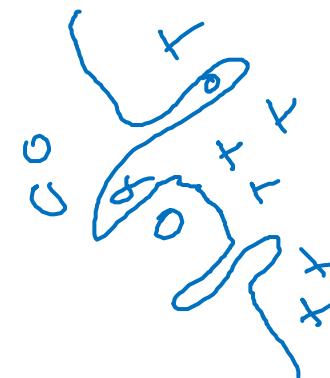
$$\lambda \uparrow$$

$$\underline{w^{[l]}} \downarrow$$

$$z^{[l]} = \underline{w^{[l]}} \underline{a^{[l-1]}} + b^{[l]}$$

Every layer \approx linear.

$$J(\dots) = \left[\sum_i L(\hat{y}^{(i)}, y^{(i)}) \right] + \frac{\lambda}{2m} \sum_l \|w^{[l]}\|_F^2$$



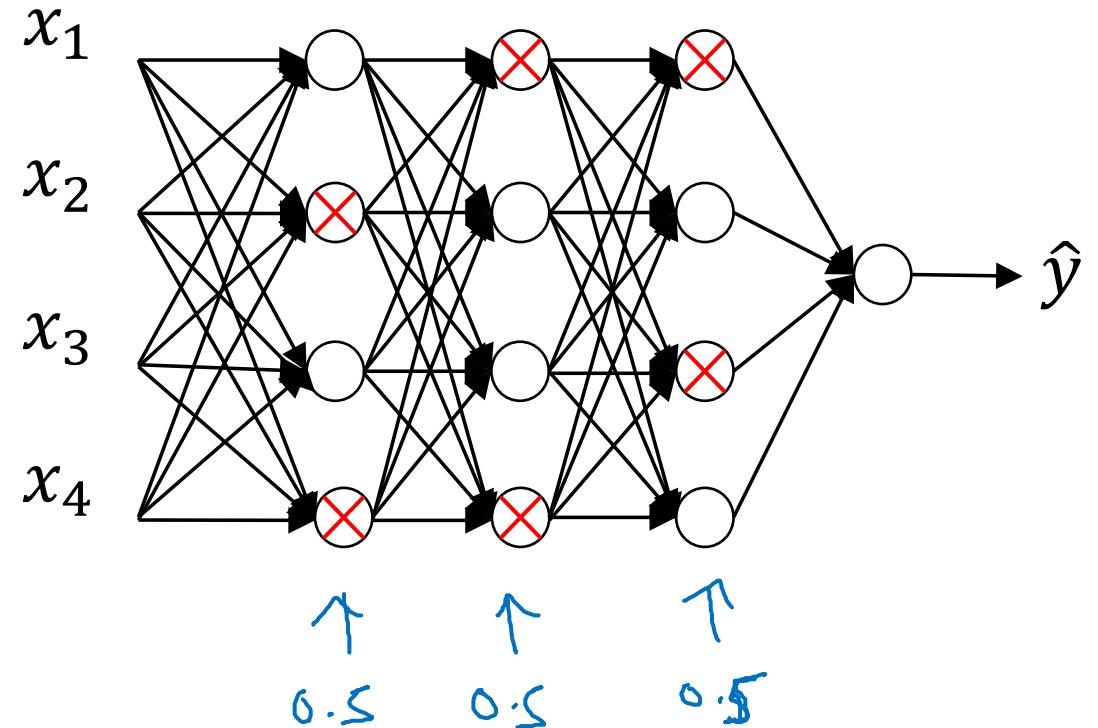
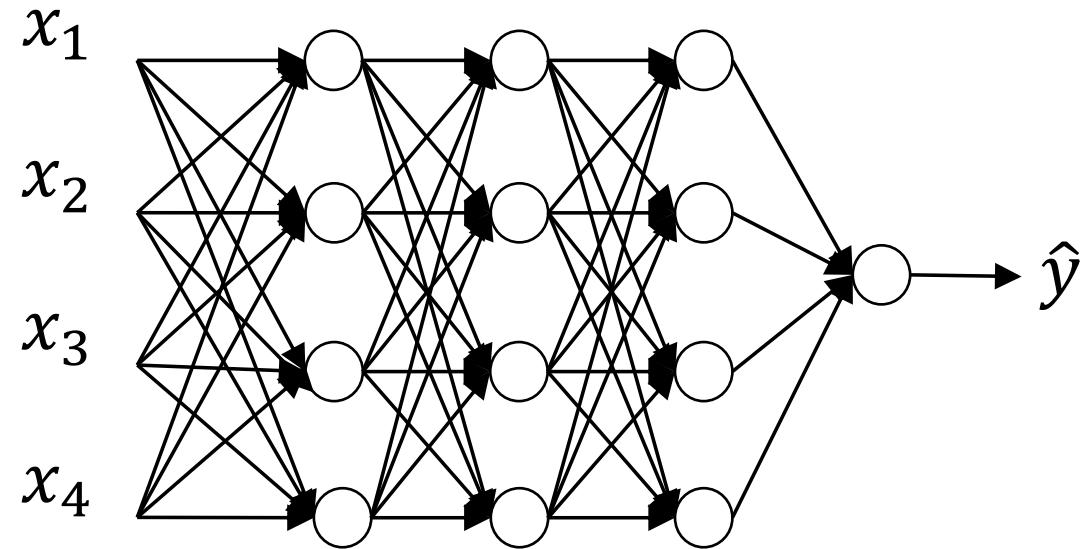


deeplearning.ai

Regularizing your
neural network

Dropout
regularization

Dropout regularization



Implementing dropout (“Inverted dropout”)

Illustrate with layer $l=3$. $\text{keep-prob} = \frac{0.8}{x}$ $\underline{\underline{0.2}}$

$$\rightarrow d_3 = \underbrace{\text{np.random.rand}(a_3.shape[0], a_3.shape[1]) < \text{keep-prob}}_{\text{d3}}$$

$$\underbrace{a_3}_{\text{a3}} = \text{np.multiply}(a_3, d_3) \quad \# a_3 * d_3.$$

$$\rightarrow \underbrace{a_3 /=\cancel{0.8} \text{ keep-prob}}_{\text{a3}} \leftarrow$$

50 units. \rightsquigarrow 10 units shut off

$$z^{(4)} = w^{(4)} \cdot \underbrace{\frac{a^{(3)}}{x}}_{\text{reduced by } 20\%} + b^{(4)}$$

Test

$$x = \underline{\underline{0.8}}$$

Making predictions at test time

$$a^{(0)} = X$$

No drop out.

$$\uparrow z^{(1)} = w^{(1)} \underline{a^{(0)}} + b^{(1)}$$

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

$$z^{(2)} = w^{(2)} \underline{a^{(1)}} + b^{(2)}$$

$$a^{(2)} = \dots$$

$$\downarrow \hat{y}$$

λ = keep-prob



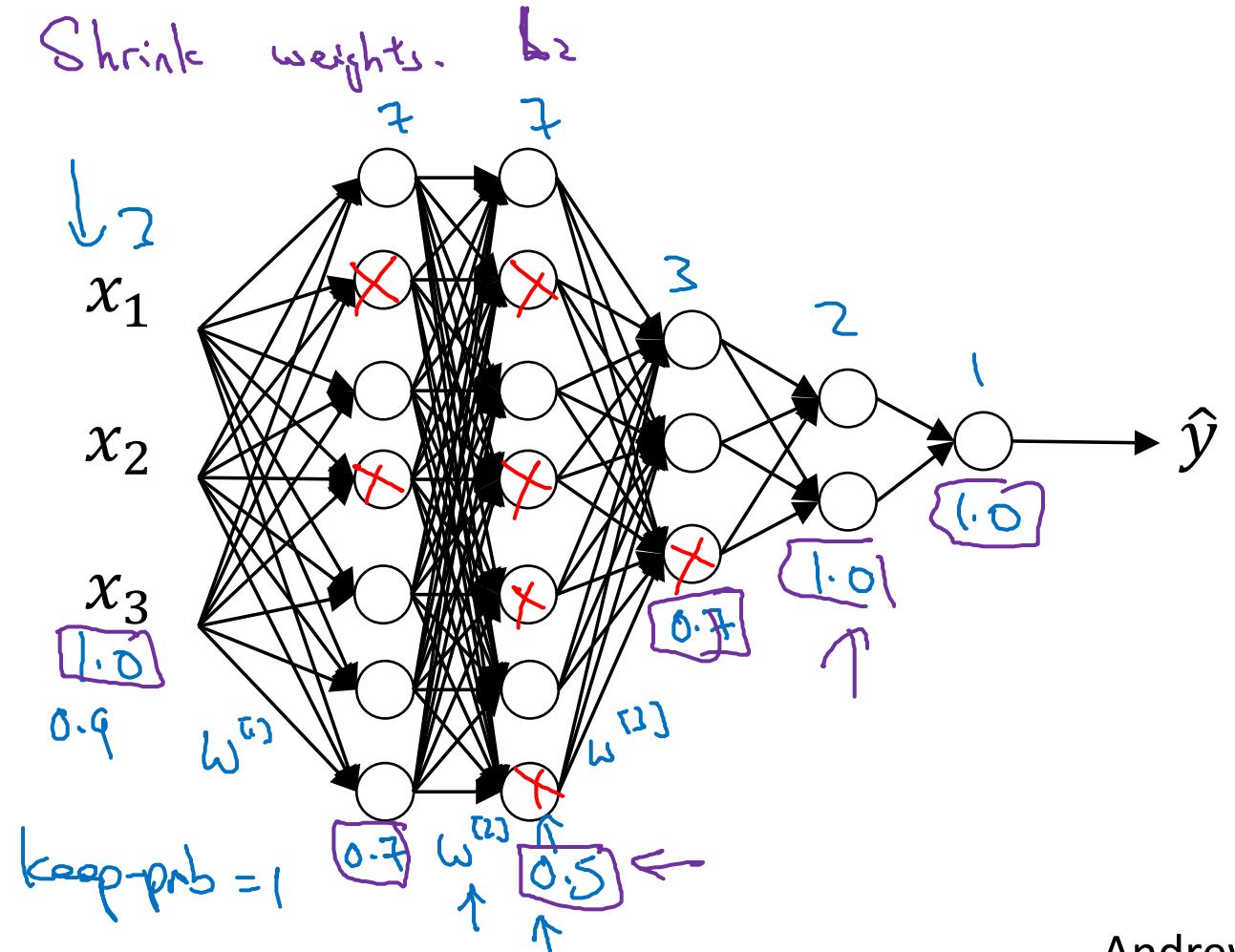
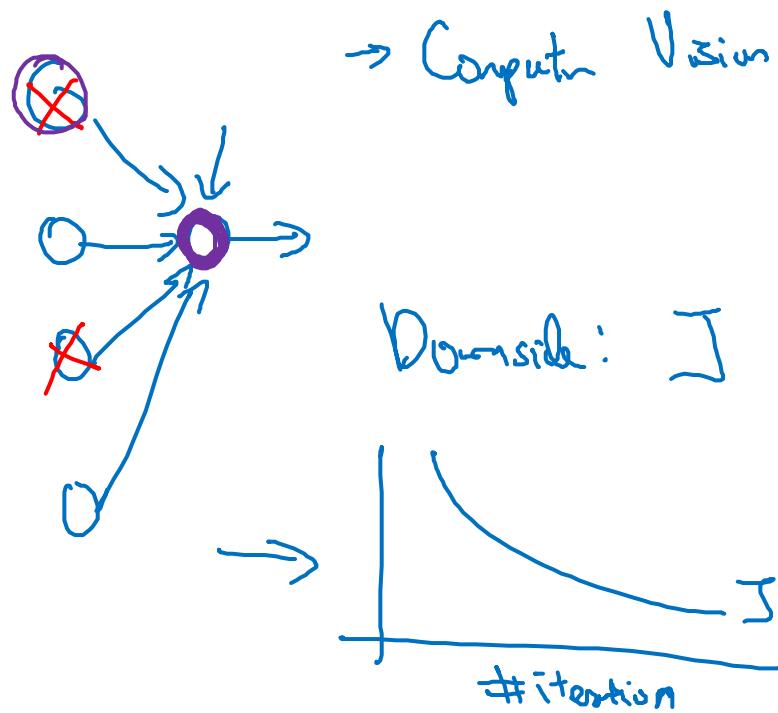
deeplearning.ai

Regularizing your
neural network

Understanding
dropout

Why does drop-out work?

Intuition: Can't rely on any one feature, so have to spread out weights. \rightsquigarrow Shrink weights.





deeplearning.ai

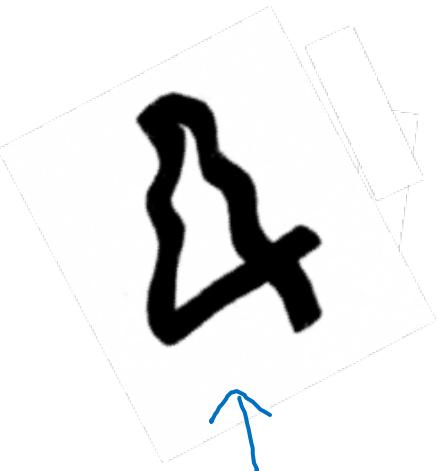
Regularizing your neural network

Other regularization methods

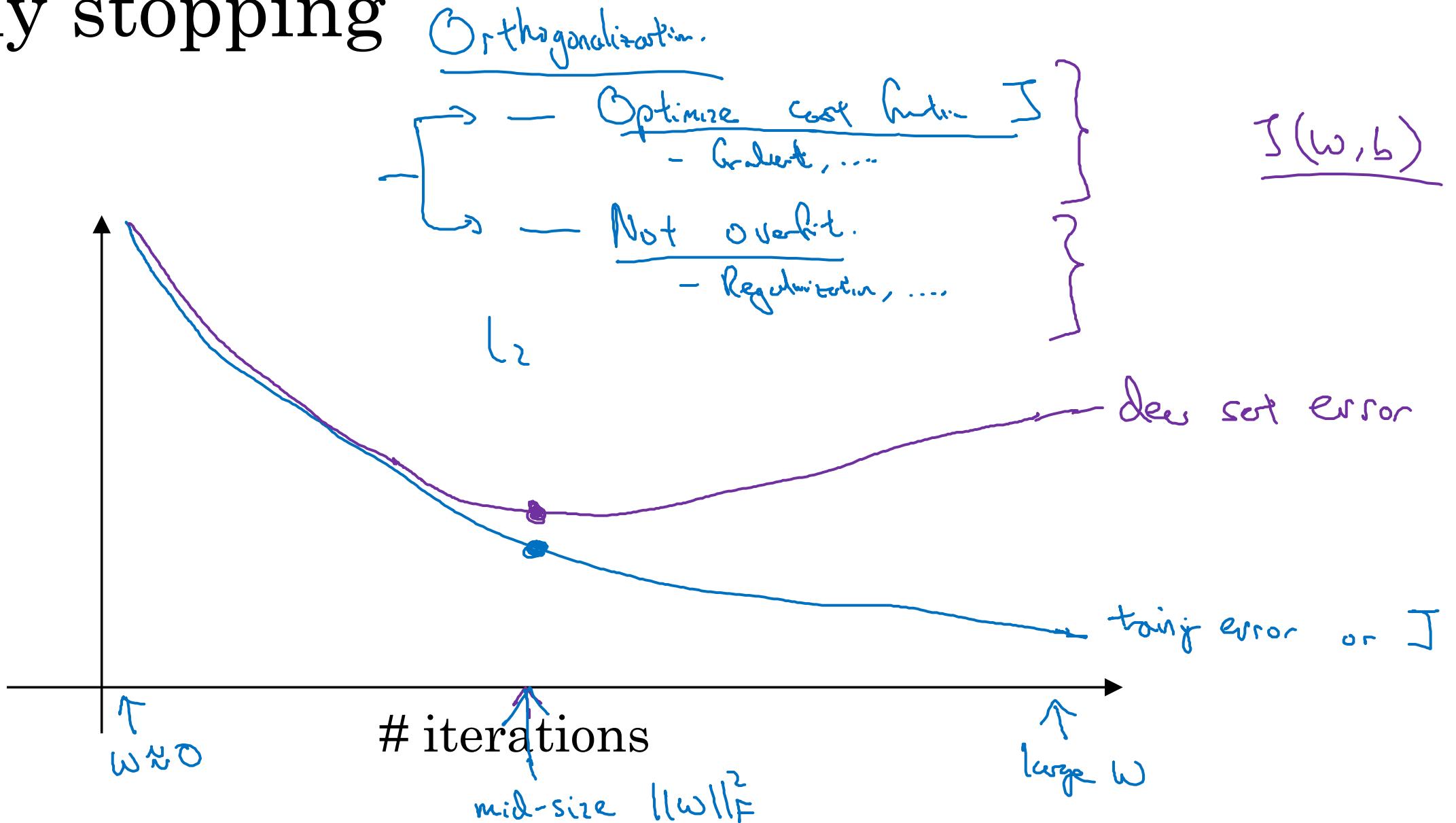
Data augmentation



4



Early stopping





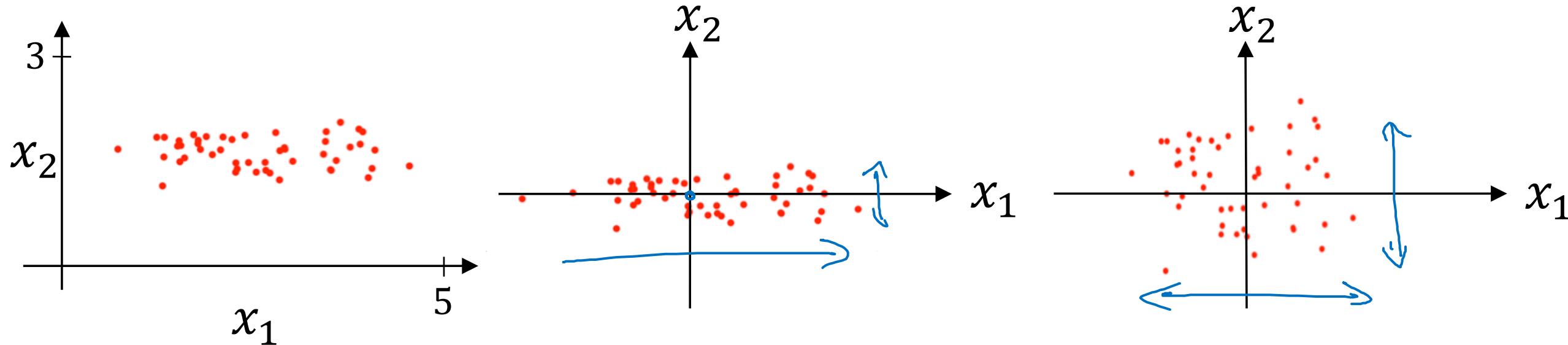
deeplearning.ai

Setting up your
optimization problem

Normalizing inputs

Normalizing training sets

$$x = \begin{bmatrix} x_1 \\ x_2 \end{bmatrix}$$



Subtract mean:

$$\mu = \frac{1}{m} \sum_{i=1}^m x^{(i)}$$

$$\underline{x := x - \mu}$$

Normalize variance

$$\sigma^2 = \frac{1}{m} \sum_{i=1}^m x^{(i)} * x^{(i)}$$

~ element-wise

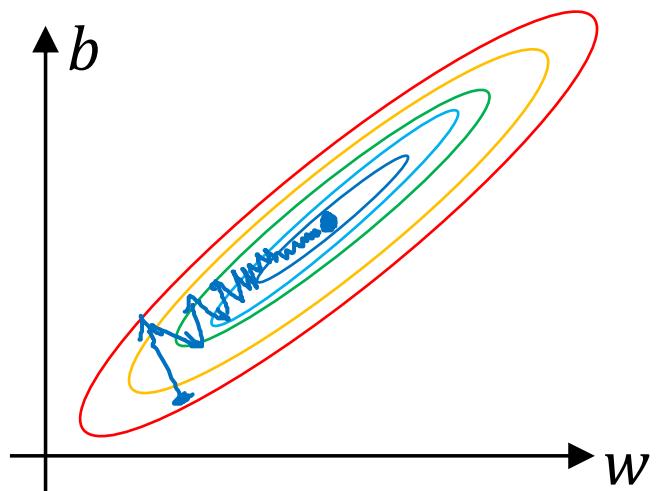
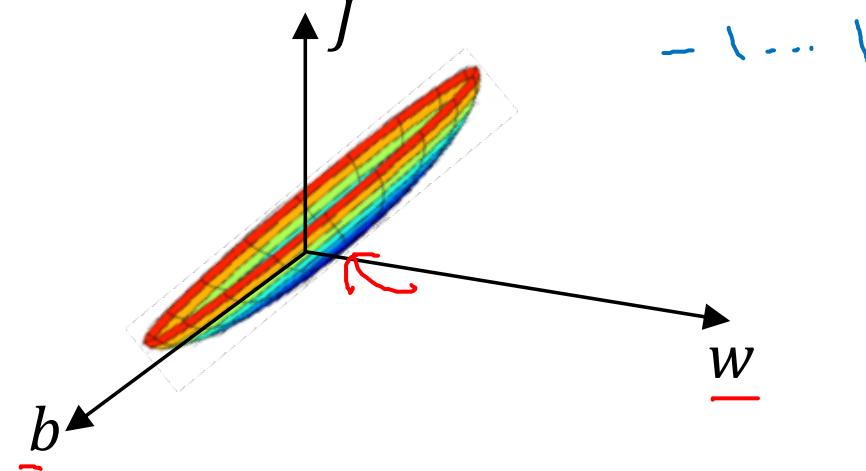
$$\underline{x / \sigma^2}$$

Use same μ, σ^2 to normalize test set.

Why normalize inputs?

$\omega_1 \quad x_1: \frac{1 \dots 1000}{0 \dots 1} \leftarrow$
 $\omega_2 \quad x_2: \frac{0 \dots 1}{-1 \dots 1} \leftarrow$

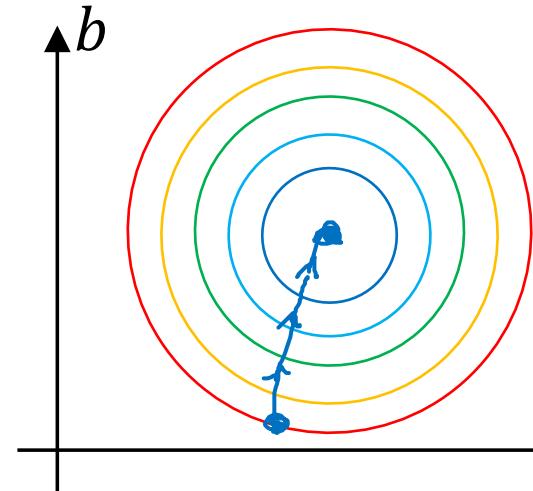
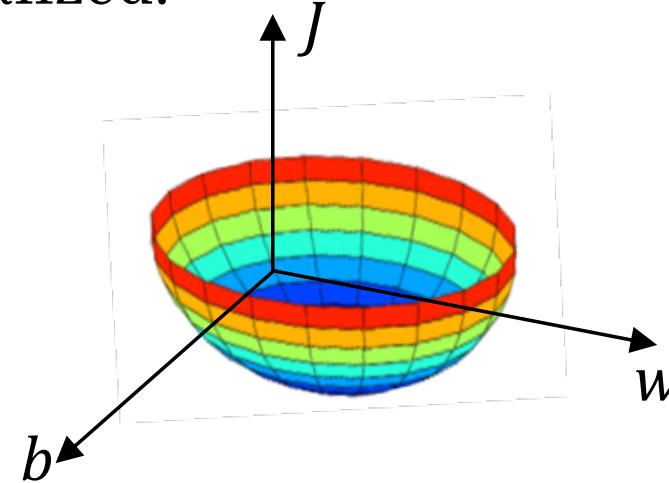
Unnormalized:



$x_1: 0 \dots 1$
 $x_2: -1 \dots 1$
 $x_3: 1 \dots 2$

$$J(w, b) = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Normalized:



w Andrew Ng



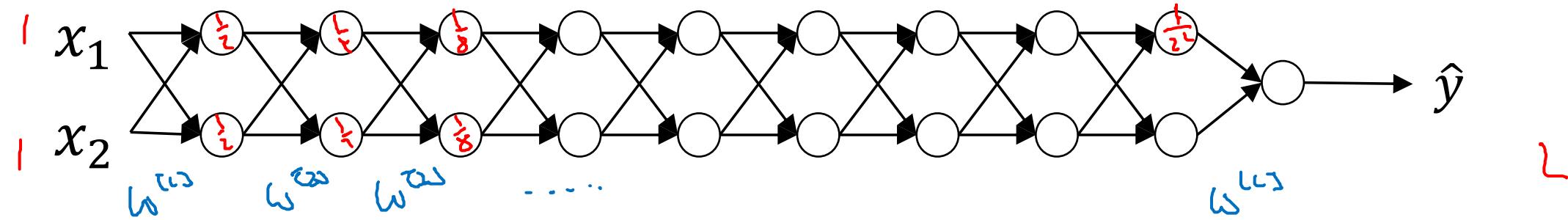
deeplearning.ai

Setting up your
optimization problem

Vanishing/exploding
gradients

Vanishing/exploding gradients

$L=150$



$$\underline{g(z) = z} . \quad b^{[L]} = 0 .$$



$$w^{[1]} > I$$

$$w^{[2]} < I \quad \begin{bmatrix} 0.9 & \\ & 0.9 \end{bmatrix}$$

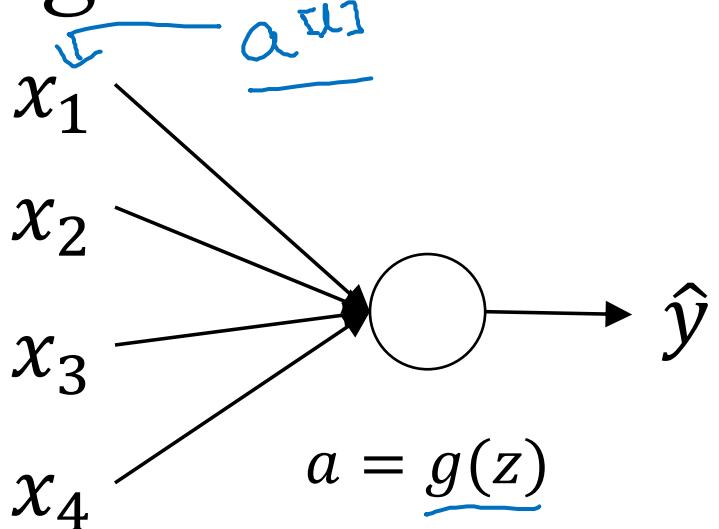
$$w^{[L]} = \begin{bmatrix} 1.5 & 0 \\ 0 & 6.5 \end{bmatrix}$$

$$\hat{y} = w^{[L]} \begin{bmatrix} 0.5 & \\ & 1.5 & 0 \\ 0 & 0.5 & 6.5 \end{bmatrix}^{L-1} x$$

$$z^{[L-1]} = w^{[L]} x$$

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

Single neuron example



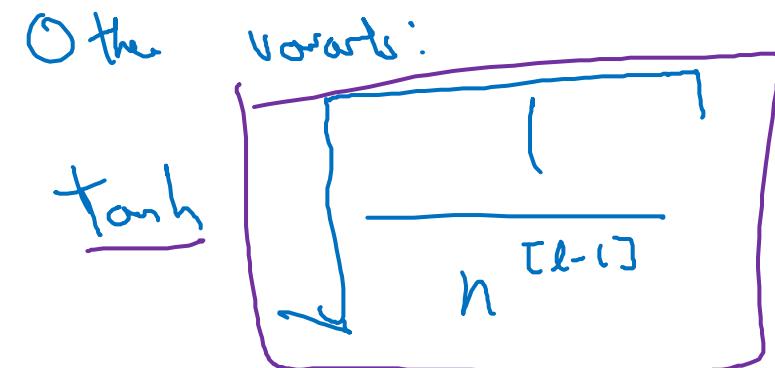
$$z = \underline{w_1}x_1 + \underline{w_2}x_2 + \cdots + \underline{w_n}x_n \quad \cancel{\text{if } n \text{ is large}}$$

\downarrow
Large $n \rightarrow$ Smaller w_i

$$\text{Var}(w_i) = \frac{1}{n} \frac{2}{n}$$

$$\underline{w^{[l]}} = \text{np.random.randn}(\text{shape}) * \text{np.sqrt}\left(\frac{2}{n^{[l-1]}}\right)$$

\downarrow
 ReLU $\underline{g^{[l]}}(z) = \text{ReLU}(z)$



$$\frac{2}{n^{[l-1]} + n^{[l]}}$$

\uparrow



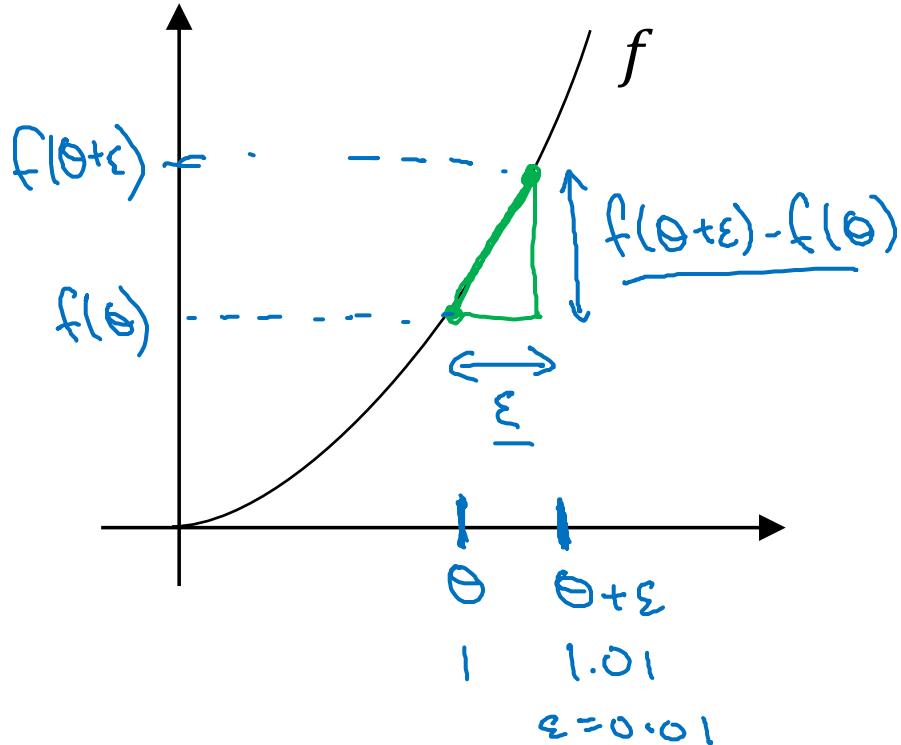
deeplearning.ai

Setting up your optimization problem

Numerical approximation of gradients

Checking your derivative computation

$$\begin{aligned} f(\theta) &= \underline{\theta^3} \\ \theta &\in \mathbb{R}. \\ \text{I} \end{aligned}$$



$$\begin{aligned} g(\theta) &= \frac{d}{d\theta} f(\theta) = f'(\theta) \\ g(\theta) &= 3\theta^2. \\ g(1) &= 3 \cdot (1)^2 = 3 \\ \text{when } \theta &= 1 \\ \frac{dw}{db} \end{aligned}$$

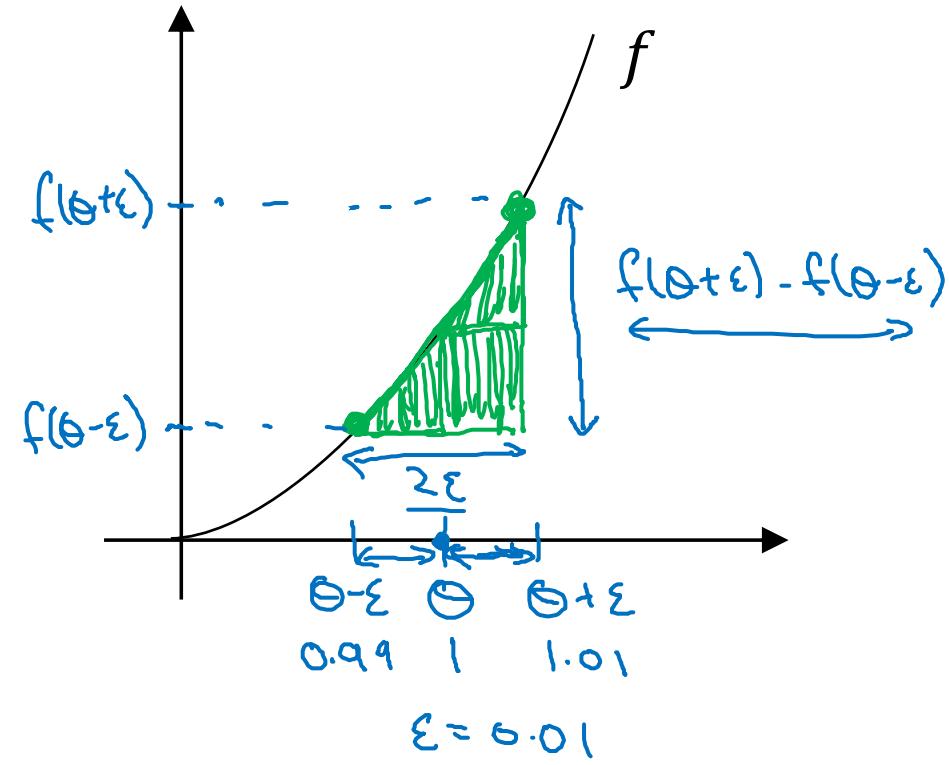
$$\begin{aligned} \frac{f(\theta + \epsilon) - f(\theta)}{\epsilon} &\approx g(\theta) \\ \frac{(1.01)^3 - 1^3}{0.01} &= \frac{3.0301}{0.01} \\ &\approx 3 \end{aligned}$$

$$\begin{aligned} \theta &= 1 \\ \theta + \epsilon &= 1.01 \end{aligned}$$

$$\begin{aligned} 3.1 \\ 3.2 \\ 0.0301 \end{aligned}$$

Checking your derivative computation

$$\underline{f(\theta) = \theta^3}$$



$$\left[\frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon} \right] \approx g(\theta)$$

$$\frac{(1.01)^3 - (0.99)^3}{2(0.01)} = 3.0001 \approx 3$$

$$g(\theta) = 3\theta^2 = 3$$

approx error: 0.0001

(prev slide: 3.0301. error: 0.03)

$\left\{ f'(\theta) = \lim_{\epsilon \rightarrow 0} \frac{f(\theta + \epsilon) - f(\theta - \epsilon)}{2\epsilon} \right.$	$\frac{\mathcal{O}(\epsilon^2)}{0.01} = \underline{0.0001}$	$\left \frac{f(\theta + \epsilon) - f(\theta)}{\epsilon} \right. \uparrow \qquad \text{error: } \mathcal{O}(\epsilon) \right. \uparrow \qquad 0.01$
--	---	--



deeplearning.ai

Setting up your
optimization problem

Gradient Checking

Gradient check for a neural network

Take $\underline{W^{[1]}, b^{[1]}, \dots, W^{[L]}, b^{[L]}}$ and reshape into a big vector $\underline{\theta}$.

$$J(\underline{w^{[1]}, b^{[1]}, \dots, w^{[L]}, b^{[L]}}) = J(\underline{\theta})$$

Take $\underline{dW^{[1]}, db^{[1]}, \dots, dW^{[L]}, db^{[L]}}$ and reshape into a big vector $\underline{d\theta}$.

Is $d\theta$ the gradient of $J(\theta)$?

Gradient checking (Grad check)

$$J(\theta) = J(\theta_0, \theta_1, \theta_2, \dots)$$

for each i :

$$\rightarrow \underline{d\theta_{\text{approx}}[i]} = \frac{J(\theta_0, \theta_1, \dots, \theta_i + \varepsilon, \dots) - J(\theta_0, \theta_1, \dots, \theta_i - \varepsilon, \dots)}{\varepsilon}$$

$$\approx \underline{d\theta[i]} = \frac{\partial J}{\partial \theta_i}$$

$$d\theta_{\text{approx}} \stackrel{?}{\approx} d\theta$$

Check

$$\rightarrow \frac{\|d\theta_{\text{approx}} - d\theta\|_2}{\|d\theta_{\text{approx}}\|_2 + \|d\theta\|_2}$$

$$\varepsilon = 10^{-7}$$

$$\approx \boxed{10^{-7} - \text{great!}} \leftarrow 10^{-5}$$

$$\rightarrow 10^{-3} - \text{worry.} \leftarrow$$



deeplearning.ai

Setting up your optimization problem

Gradient Checking implementation notes

Gradient checking implementation notes

- Don't use in training – only to debug

$$\frac{\partial \theta_{\text{approx}}^{[i]}}{\uparrow} \longleftrightarrow \frac{\partial \theta^{[i]}}{\uparrow}$$

- If algorithm fails grad check, look at components to try to identify bug.

$$\frac{\partial b^{[l]}}{\uparrow} \quad \frac{\partial w^{[l]}}{\uparrow}$$

- Remember regularization.

$$J(\theta) = \frac{1}{m} \sum_i f(y^{(i)}, \hat{y}^{(i)}) + \frac{\lambda}{2m} \sum_l \|w^{(l)}\|_F^2$$

$\frac{\partial \theta}{\partial \theta} = \text{gradient of } J \text{ wrt. } \theta$

- Doesn't work with dropout.

$$J \quad \underline{\text{keep-prob} = 1.0}$$

- Run at random initialization; perhaps again after some training.

$$\underline{w, b \text{ no}}$$

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

Optimization Algorithms

Mini-batch gradient descent

Batch vs. mini-batch gradient descent

X, Y

$X^{\{t\}}, Y^{\{t\}}$

Vectorization allows you to efficiently compute on m examples.

$$X = \begin{bmatrix} X^{(1)} & X^{(2)} & X^{(3)} & \dots & X^{(500)} & | & X^{(1001)} & \dots & X^{(2000)} & | & \dots & | & \dots & X^{(m)} \end{bmatrix}$$

(n_x, m)

$\underbrace{X^{\{1\}}}_{(n_x, 1000)}$ $(n_x, 1000)$ $\underbrace{X^{\{2\}}}_{(n_x, 1000)}$ $(n_x, 1000)$ \dots $\underbrace{X^{\{5,000\}}}_{(n_x, 1000)}$ $(n_x, 1000)$

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & y^{(3)} & \dots & y^{(1000)} & | & y^{(1001)} & \dots & y^{(2000)} & | & \dots & | & \dots & y^{(m)} \end{bmatrix}$$

$(1, m)$

$\underbrace{Y^{\{1\}}}_{(1, 1000)}$ $(1, 1000)$ $\underbrace{Y^{\{2\}}}_{(1, 1000)}$ $(1, 1000)$ \dots $\underbrace{Y^{\{5,000\}}}_{(1, 1000)}$ $(1, 1000)$

What if $m = 5,000,000$?

5,000 mini-batches of 1,000 each

Mini-batch t : $X^{\{t\}}, Y^{\{t\}}$

$X^{(i)}$
 $Z^{[l]}$
 $X^{\{t\}}, Y^{\{t\}}$

Mini-batch gradient descent

repeat {
for $t = 1, \dots, 5000$ {

Forward prop on $X^{\{t\}}$.

$$Z^{(l)} = W^{(l)} X^{\{t\}} + b^{(l)}$$

$$A^{(l)} = g^{(l)}(Z^{(l)})$$

:

$$A^{(L)} = g^{(L)}(Z^{(L)})$$

Compute cost $J^{\{t\}} = \frac{1}{1000} \sum_{i=1}^L f(\hat{y}^{(i)}, y^{(i)}) + \frac{\lambda}{2 \cdot 1000} \sum_l \|W^{(l)}\|_F^2$.

Backprop to compute gradients wrt $J^{\{t\}}$ (using $(X^{\{t\}}, Y^{\{t\}})$)

$$W^{(l)} := W^{(l)} - \alpha \nabla W^{(l)}, \quad b^{(l)} := b^{(l)} - \alpha \nabla b^{(l)}$$

3 } 3 }

"1 epoch"
└ pass through training set.

1 step of gradient descent
using $\frac{X^{\{t+1\}}}{Y^{\{t+1\}}}$
(as if $t=5000$)

X, Y



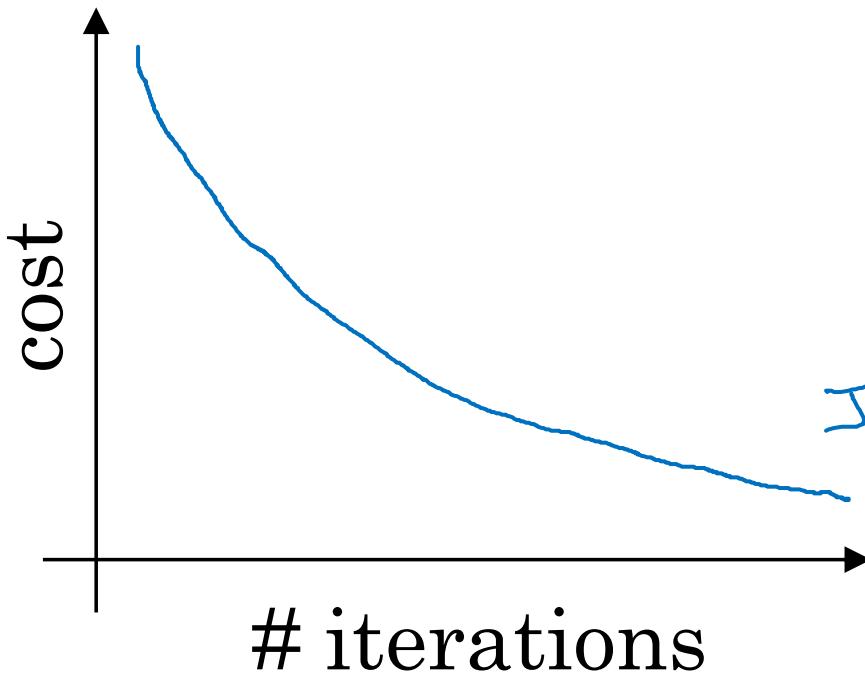
deeplearning.ai

Optimization Algorithms

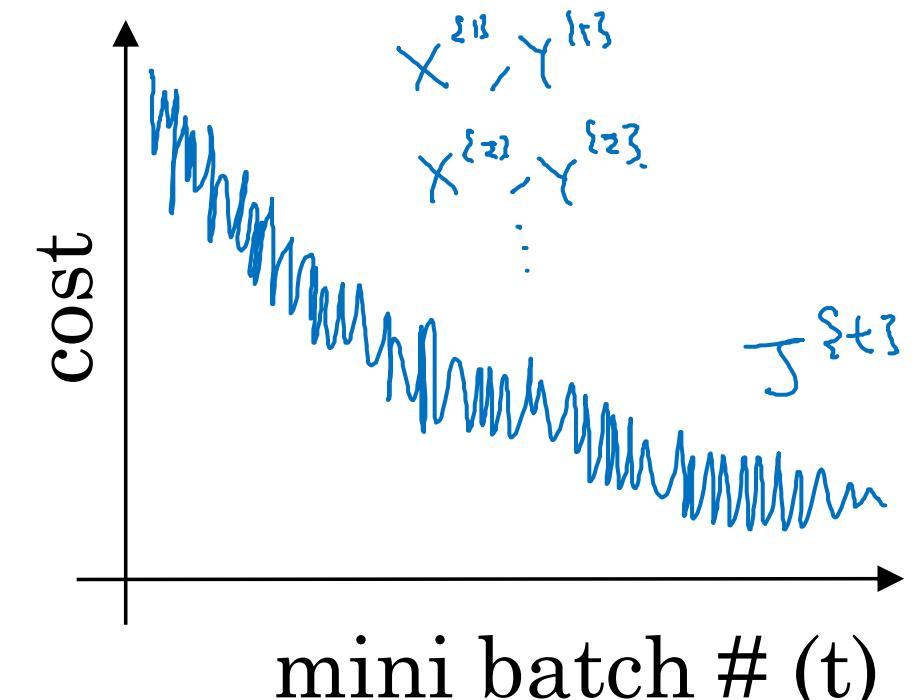
Understanding mini-batch gradient descent

Training with mini batch gradient descent

Batch gradient descent



Mini-batch gradient descent



Plot J^{st} computed using $x^{\{t\}}, y^{\{t\}}$

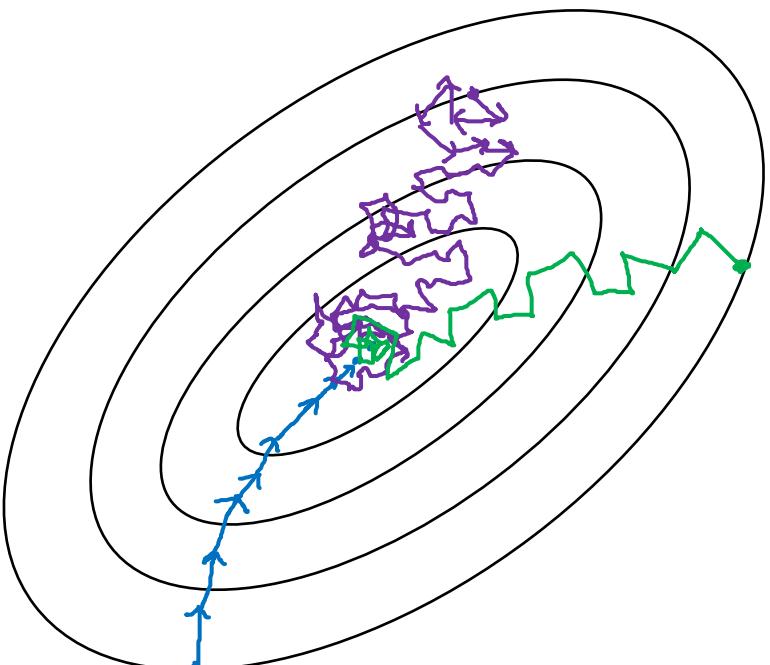
Choosing your mini-batch size

→ If mini-batch size = m : Batch gradient descent.

$$(X^{\{1\}}, Y^{\{1\}}) = (X, Y)$$

→ If mini-batch size = 1 : Stochastic gradient descent. Every example is its own
 $(X^{(1)}, Y^{(1)}) = (x^{(1)}, y^{(1)}) \dots (x^{(n)}, y^{(n)})$ mini-batch.

In practice: Somehw in-between 1 and m



Stochastic
gradient
descent

{
Use speedup
from vectorization

In-between
(mini-batch size
not too big/small)

{
Faster learning.

- Vectorization.
($n \approx 1000$)
- Make passes without
processing entire training set.

Batch
gradient descent
(mini-batch size = m)

{
Two long
per iteration

Choosing your mini-batch size

If small training set : Use batch gradient descent.
 $(m \leq 2000)$

Typical mini-batch sizes:

$$\rightarrow 64, 128, 256, 512 \quad \frac{1024}{2^{10}}$$

$2^6 \quad 2^7 \quad 2^8 \quad 2^9$



Make sure mini-batch fits in CPU/GPU memory.

$$X^{\{t\}}, Y^{\{t\}}$$



deeplearning.ai

Optimization Algorithms

Exponentially weighted averages

Temperature in London

$$\theta_1 = 40^{\circ}\text{F} \quad 4^{\circ}\text{C} \quad \leftarrow$$

$$\theta_2 = 49^{\circ}\text{F} \quad 9^{\circ}\text{C}$$

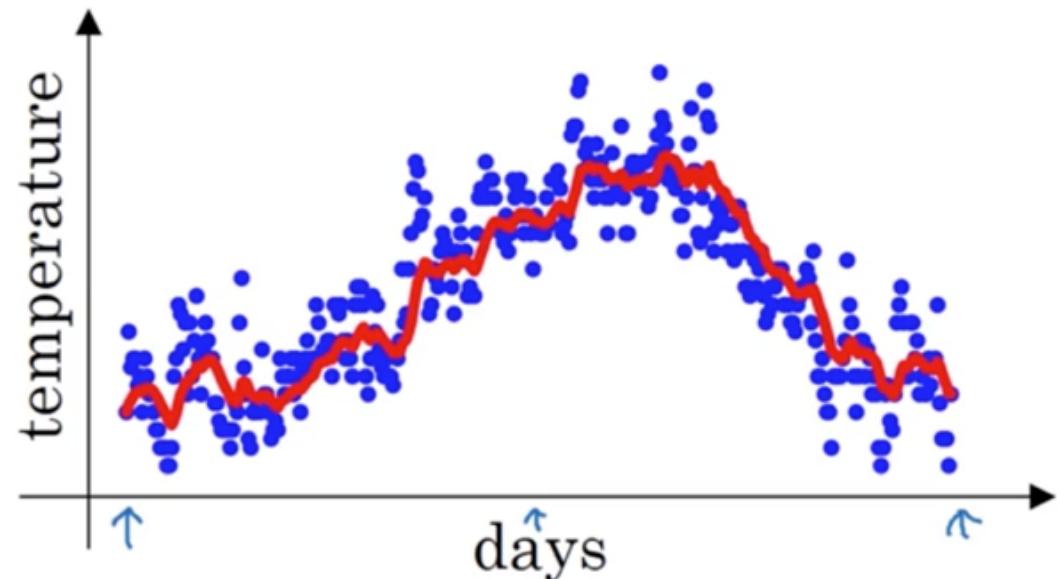
$$\theta_3 = 45^{\circ}\text{F} \quad \vdots$$

\vdots

$$\theta_{180} = 60^{\circ}\text{F} \quad 15^{\circ}\text{C}$$

$$\theta_{181} = 56^{\circ}\text{F} \quad \vdots$$

\vdots



$$V_0 = 0$$

$$V_1 = 0.9 V_0 + 0.1 \theta_1$$

$$V_2 = 0.9 V_1 + 0.1 \theta_2$$

$$V_3 = 0.9 V_2 + 0.1 \theta_3$$

\vdots

$$V_t = 0.9 V_{t-1} + 0.1 \theta_t$$

Exponentially weighted averages

$$\underline{V}_t = \beta \underline{V}_{t-1} + (1-\beta) \underline{\theta}_t \leftarrow$$

$\beta = 0.9$: ≈ 10 days' temper.

$\beta = 0.98$: ≈ 50 days

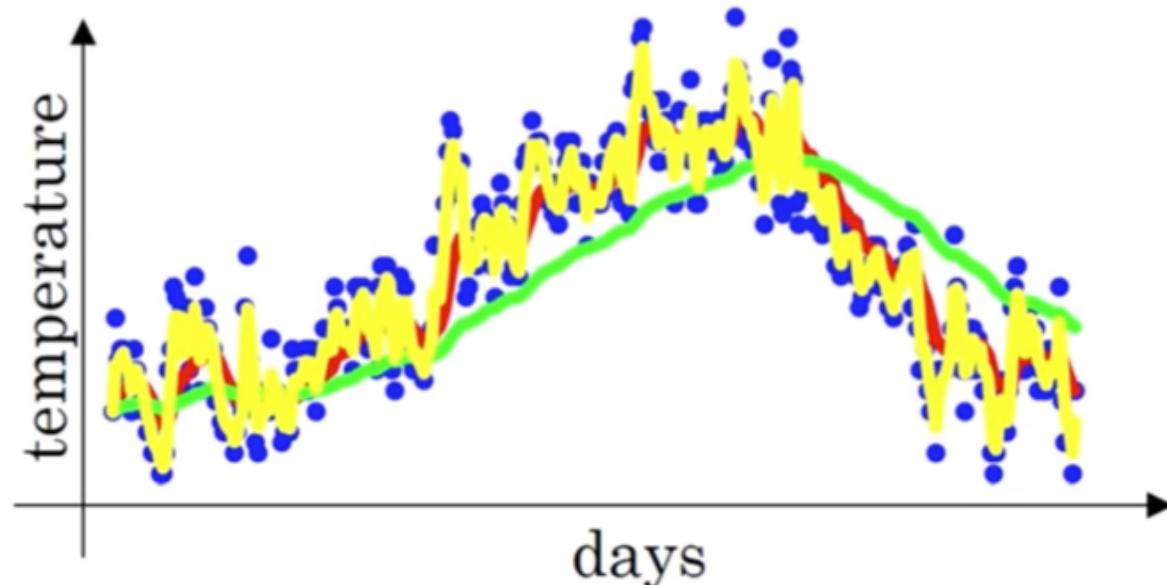
$\beta = 0.5$: ≈ 2 days

V_t is approximately

average over

$\rightarrow \approx \frac{1}{1-\beta}$ days' temperature.

$$\frac{1}{1-0.98} = 50$$





deeplearning.ai

Optimization Algorithms

Understanding exponentially weighted averages

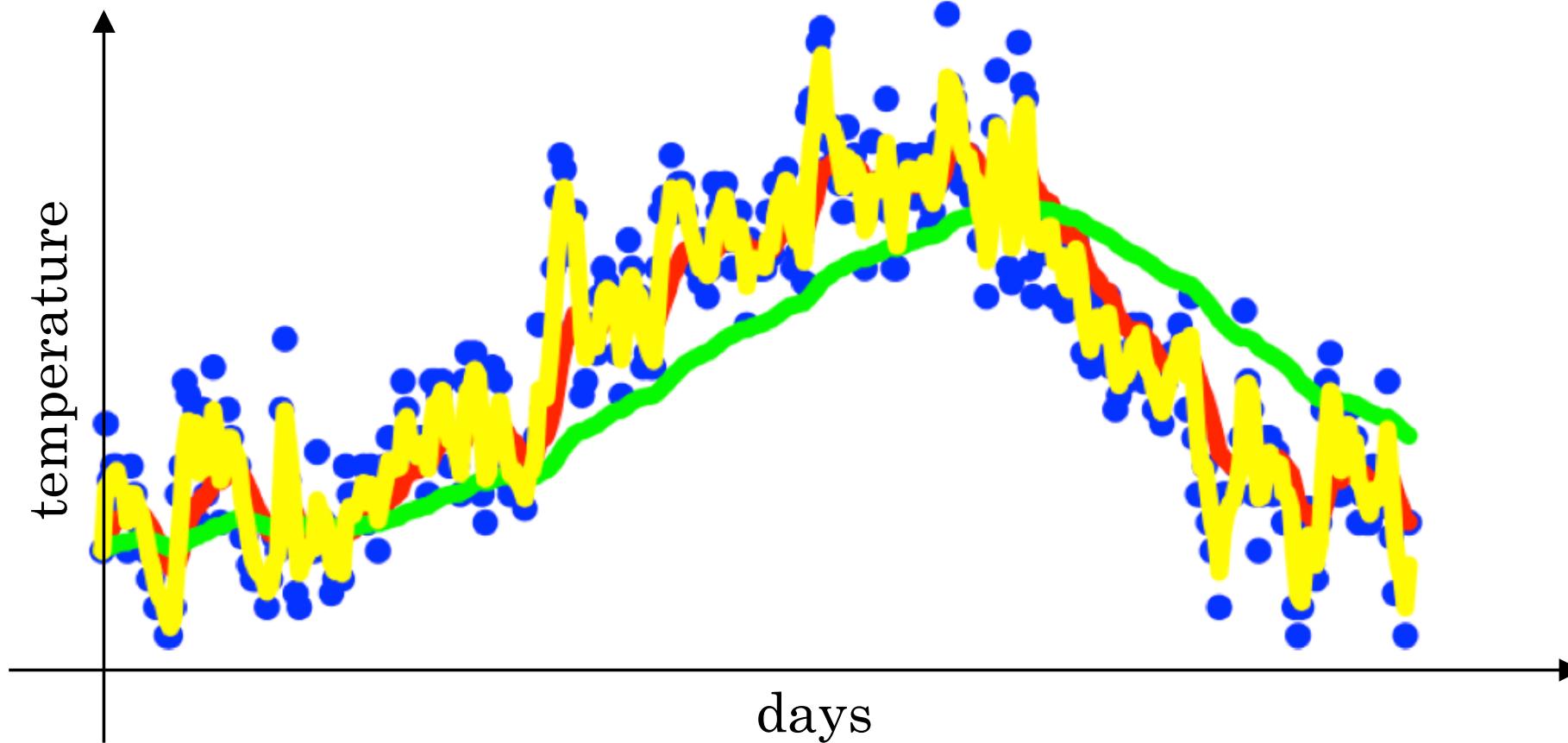
Exponentially weighted averages

$$v_t = \beta v_{t-1} + (1 - \beta) \theta_t$$

$$\beta = 0.9$$

$$0.98$$

$$0.5$$



Exponentially weighted averages

$$v_t = \beta v_{t-1} + (1 - \beta) \theta_t$$

$$v_{100} = 0.9v_{99} + 0.1\theta_{100}$$

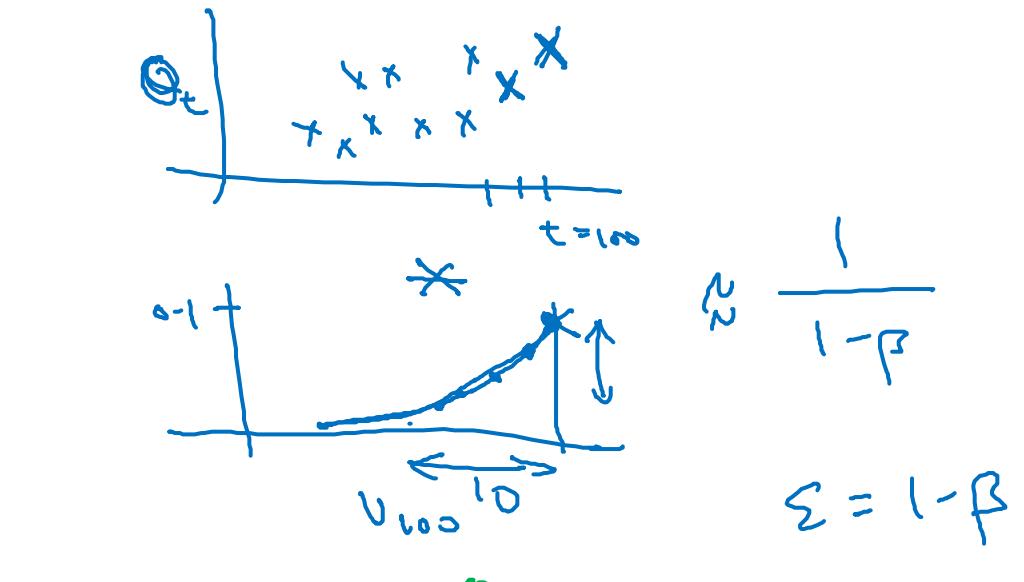
$$v_{99} = 0.9v_{98} + 0.1\theta_{99}$$

$$v_{98} = 0.9v_{97} + 0.1\theta_{98}$$

...

$$\begin{aligned} \underline{v_{100}} &= 0.1 \underline{\theta_{100}} + 0.9 \cancel{(0.1 \underline{\theta_{99}})} + 0.9 \cancel{(0.1 \underline{\theta_{98}})} + 0.9 \cancel{(0.1 \underline{\theta_{97}})} + 0.9 \cancel{(0.1 \underline{\theta_{96}})} \\ &= 0.1 \underline{\theta_{100}} + \underline{0.1 \times 0.9 \cdot \theta_{99}} + \underline{0.1 (0.9)^2 \theta_{98}} + \underline{0.1 (0.9)^3 \theta_{97}} + \underline{0.1 (0.9)^4 \theta_{96}} + \dots \end{aligned}$$

$$\underline{0.9^{10}} \approx \underline{0.35} \approx \frac{1}{e}$$



$$\begin{aligned} \frac{(1-\epsilon)^{1/\epsilon}}{0.9} &= \frac{1}{e} \\ \epsilon = 0.02 &\rightarrow \underline{0.98^{50}} \approx \frac{1}{e} \end{aligned}$$

Implementing exponentially weighted averages

$$v_0 = 0$$

$$v_1 = \beta v_0 + (1 - \beta) \theta_1$$

$$v_2 = \beta v_1 + (1 - \beta) \theta_2$$

$$v_3 = \beta v_2 + (1 - \beta) \theta_3$$

...

$$V_0 := 0$$

$$V_0 := \beta V + (1-\beta) \theta_1$$

$$V_0 := \beta V + (1-\beta) \theta_2$$

:

$$\rightarrow V_0 = 0$$

Repeat {

Get next θ_t

$$V_0 := \beta V_0 + (1-\beta) \theta_t \leftarrow$$

}

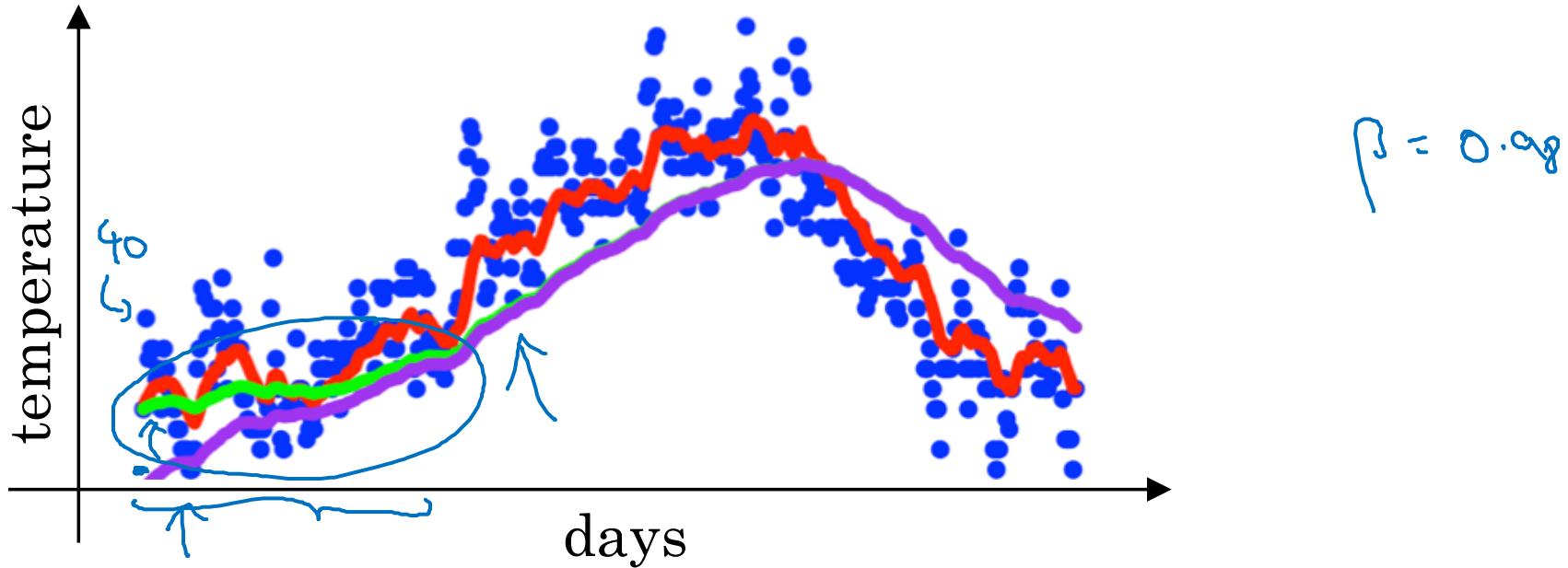


deeplearning.ai

Optimization Algorithms

Bias correction
in exponentially
weighted average

Bias correction



$$\rightarrow v_t = \beta v_{t-1} + (1 - \beta) \theta_t$$

$$v_0 = 0$$

$$v_1 = \cancel{0.98v_0} + \underline{0.02\theta_1}$$

$$\begin{aligned} v_2 &= 0.98 v_1 + 0.02 \theta_2 \\ &= 0.98 \times 0.02 \times \theta_1 + 0.02 \theta_2 \\ &= \underline{0.0196\theta_1} + \underline{0.02\theta_2} \end{aligned}$$

$$\frac{v_t}{1 - \beta^t}$$

$$t=2: 1 - \beta^t = 1 - (0.98)^2 = 0.0396$$

$$\frac{v_2}{0.0396} =$$

$$\frac{\underline{0.0196\theta_1} + \underline{0.02\theta_2}}{0.0396}$$

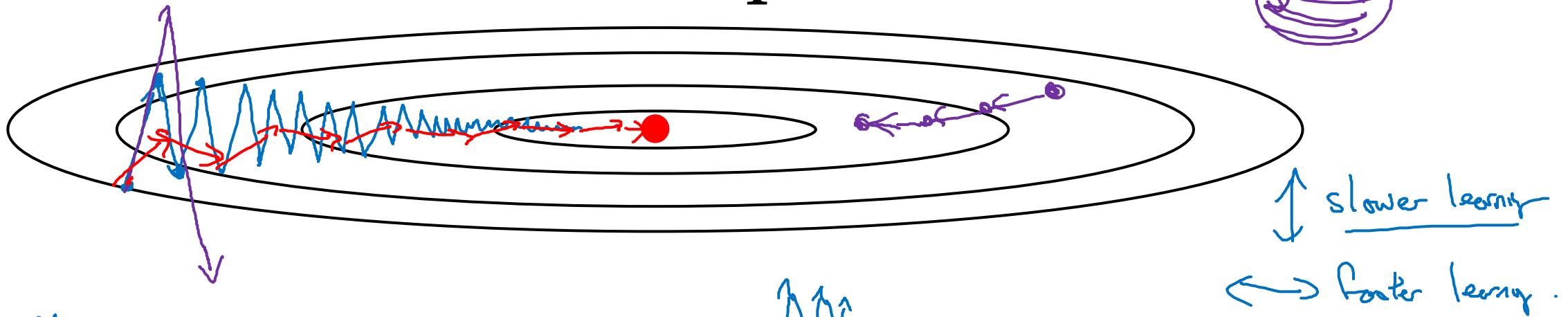


deeplearning.ai

Optimization Algorithms

Gradient descent with momentum

Gradient descent example



Momentum:

On iteration t :

Compute $\Delta w, \Delta b$ on current mini-batch.

$$v_{dw} = \beta v_{dw} + (1-\beta) \frac{\Delta w}{\text{acceleration}}$$

$$v_{db} = \beta v_{db} + (1-\beta) \frac{\Delta b}{\text{acceleration}}$$

Friction ↑ velocity

$$w := w - \alpha v_{dw}, \quad b := b - \alpha v_{db}$$

$$"v_\theta = \beta v_\theta + (1-\beta) \theta_t"$$

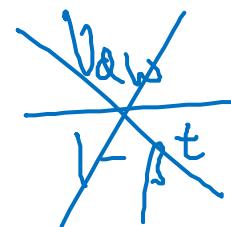
Implementation details

$$v_{dw} = 0, v_{db} = 0$$

On iteration t :

Compute dW, db on the current mini-batch

$$\begin{aligned} \rightarrow v_{dw} &= \beta v_{dw} + (1 - \beta) dW \\ \rightarrow v_{db} &= \beta v_{db} + (1 - \beta) db \end{aligned} \quad \left| \begin{array}{l} v_{dw} = \beta v_{dw} + dW \leftarrow \\ v_{db} = \beta v_{db} + db \end{array} \right.$$
$$W = W - \underbrace{\alpha v_{dw}}, b = \underbrace{b - \alpha v_{db}}$$



Hyperparameters: α, β

$$\beta = 0.9$$

average over last ≈ 10 gradients

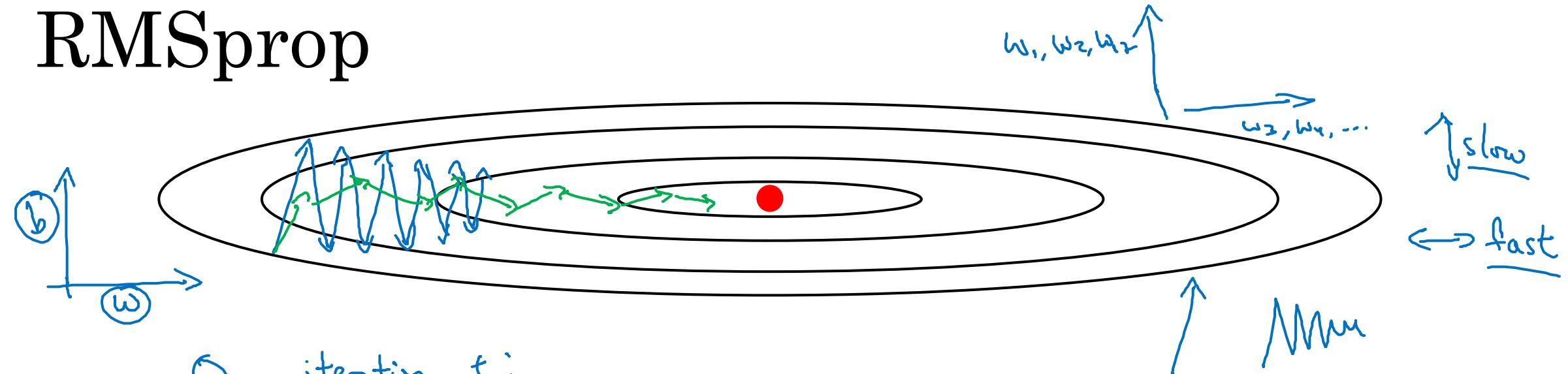


deeplearning.ai

Optimization Algorithms

RMSprop

RMSprop



On iteration t :

Compute dW, db on current mini-batch

$$\underline{S_{dw}} = \beta_2 \underline{S_{dw}} + (1-\beta_2) \underline{dW^2} \quad \begin{matrix} \text{element-wise} \\ \leftarrow \text{small} \end{matrix}$$

$$\rightarrow \underline{S_{db}} = \beta_2 \underline{S_{db}} + (1-\beta_2) \underline{d b^2} \quad \leftarrow \text{large}$$

$$w := w - \frac{\alpha dW}{\sqrt{\underline{S_{dw}} + \epsilon}} \quad \leftarrow$$

$$b := b - \frac{\alpha db}{\sqrt{\underline{S_{db}} + \epsilon}} \quad \leftarrow$$

$$\epsilon = 10^{-8}$$



deeplearning.ai

Optimization Algorithms

Adam optimization algorithm

Adam optimization algorithm

$$V_{dw} = 0, S_{dw} = 0. \quad V_{db} = 0, S_{db} = 0$$

On iteration t :

Compute $\delta w, \delta b$ using current mini-batch

$$V_{dw} = \beta_1 V_{dw} + (1 - \beta_1) \delta w, \quad V_{db} = \beta_1 V_{db} + (1 - \beta_1) \delta b \quad \leftarrow \text{"moment"} \beta_1$$

$$S_{dw} = \beta_2 S_{dw} + (1 - \beta_2) \delta w^2, \quad S_{db} = \beta_2 S_{db} + (1 - \beta_2) \delta b \quad \leftarrow \text{"RMSprop"} \beta_2$$

$yhat = np.array([.9, 0.2, 0.1, .4, .9])$

$$V_{dw}^{corrected} = V_{dw} / (1 - \beta_1^t), \quad V_{db}^{corrected} = V_{db} / (1 - \beta_1^t)$$

$$S_{dw}^{corrected} = S_{dw} / (1 - \beta_2^t), \quad S_{db}^{corrected} = S_{db} / (1 - \beta_2^t)$$

$$w := w - \alpha \frac{V_{dw}^{corrected}}{\sqrt{S_{dw}^{corrected}} + \epsilon}$$

$$b := b - \alpha \frac{V_{db}^{corrected}}{\sqrt{S_{db}^{corrected}} + \epsilon}$$

Hyperparameters choice:

- α : needs to be tune
- β_1 : 0.9 $\rightarrow (\underline{dw})$
- β_2 : 0.999 $\rightarrow (\underline{dw^2})$
- ϵ : 10^{-8}

Adam: Adaptive moment estimation



Adam Coates



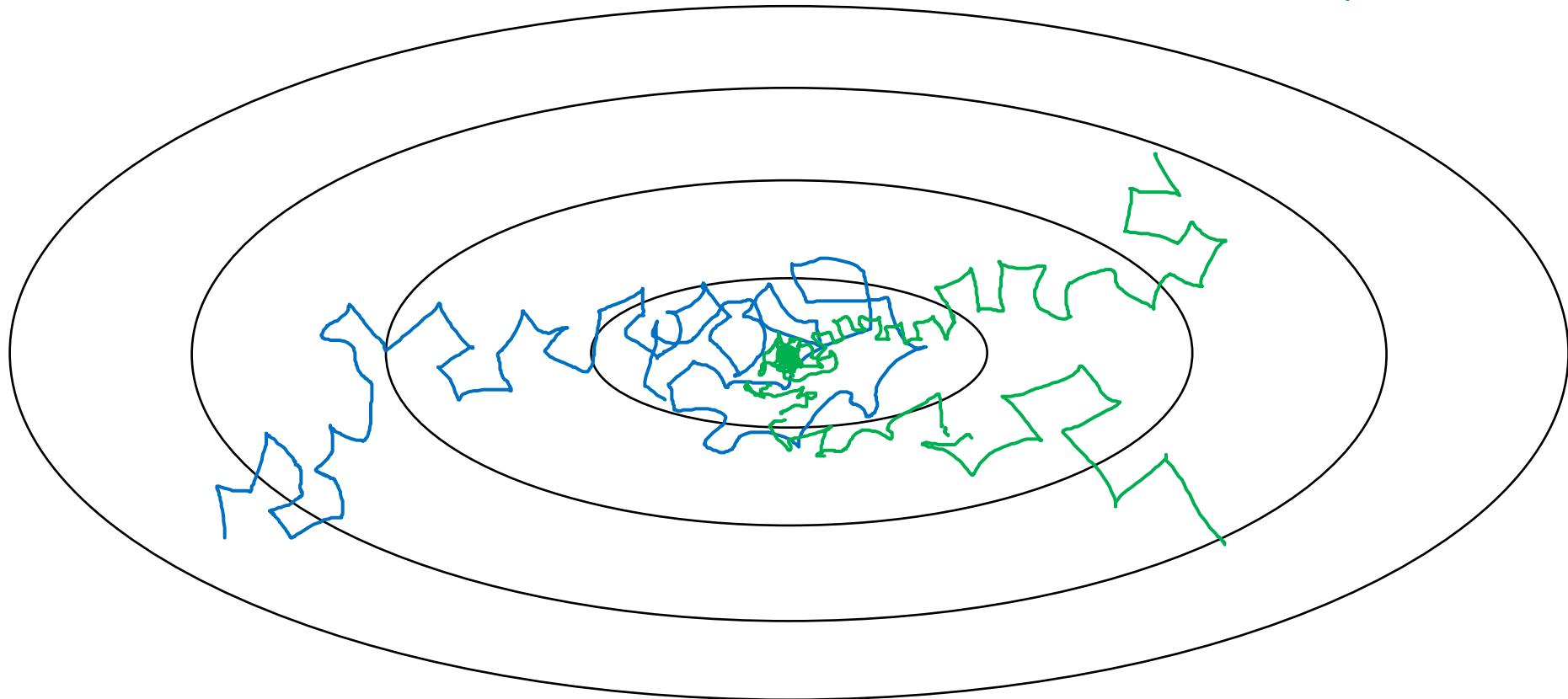
deeplearning.ai

Optimization Algorithms

Learning rate decay

Learning rate decay

Slowly reduce λ

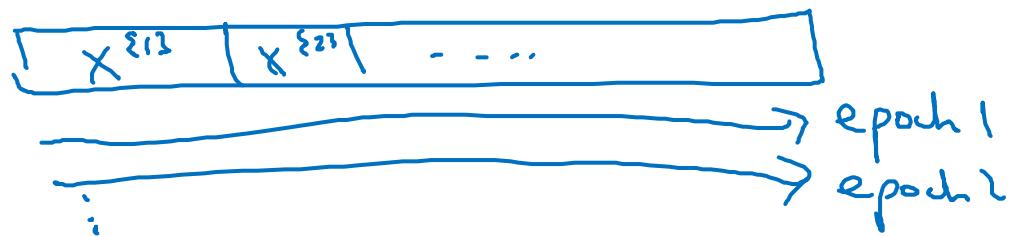


Learning rate decay

1 epoch = 1 pass through data.

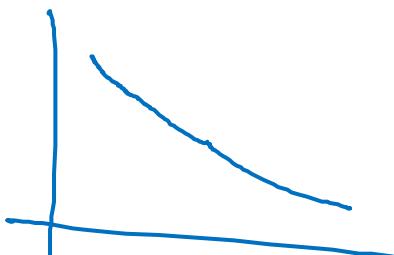
$$\alpha = \frac{\alpha_0}{1 + \text{decay-rate} * \text{epoch-num}}$$

Epoch	α
1	0.1
2	0.67
3	0.5
4	0.4
:	:

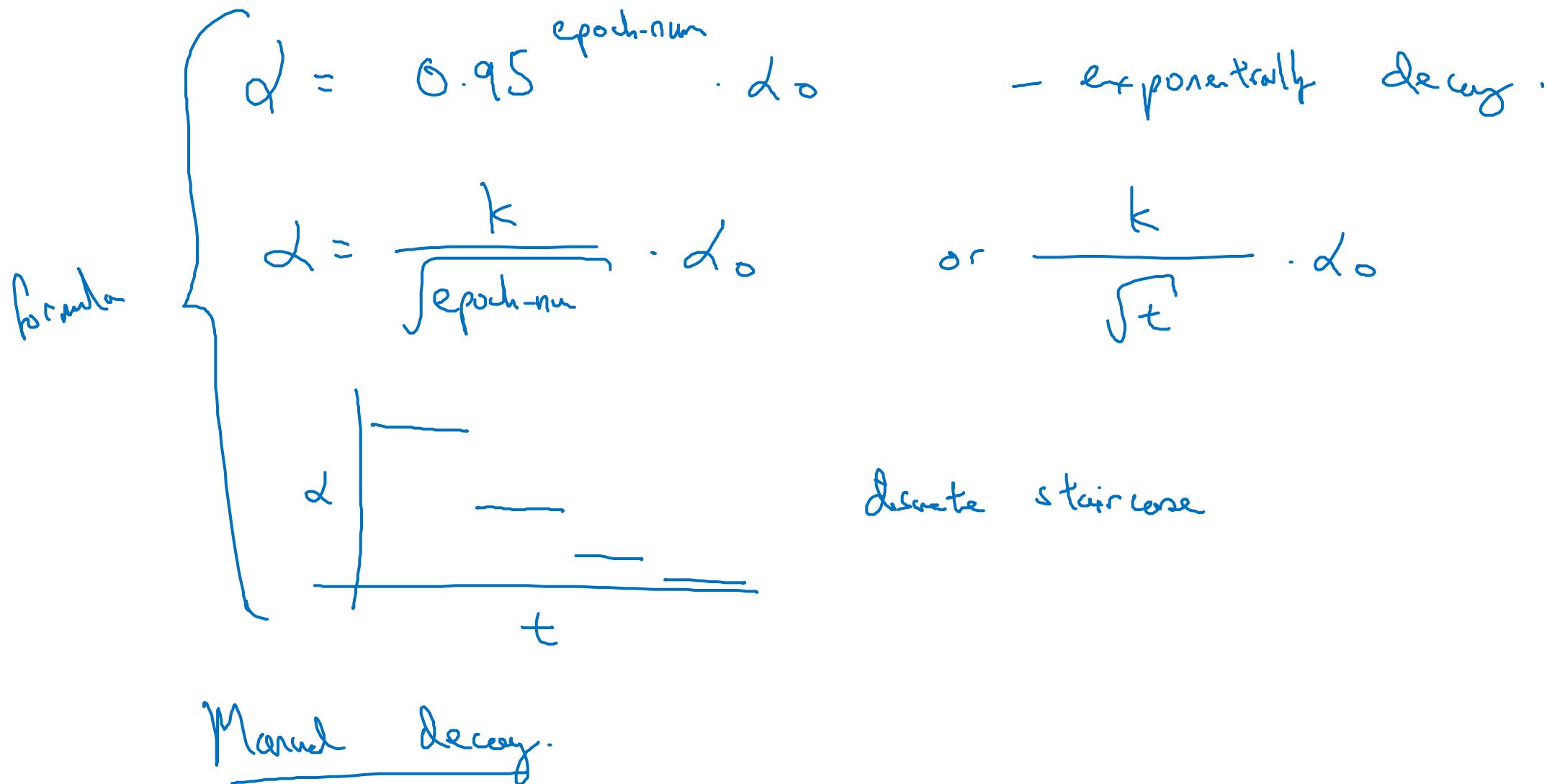


$$\alpha_0 = 0.2$$

$$\text{decay-rate} = 1$$



Other learning rate decay methods



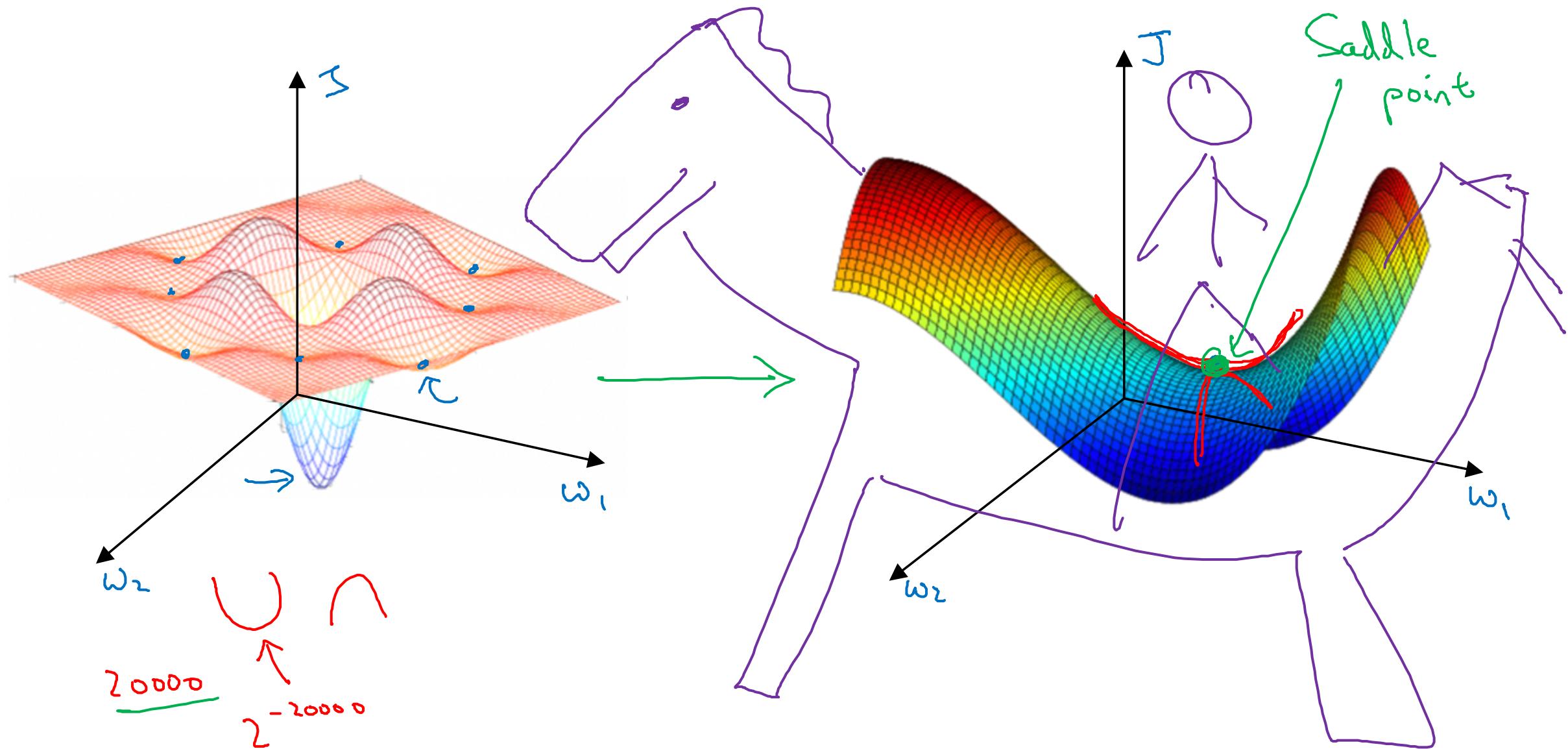


deeplearning.ai

Optimization Algorithms

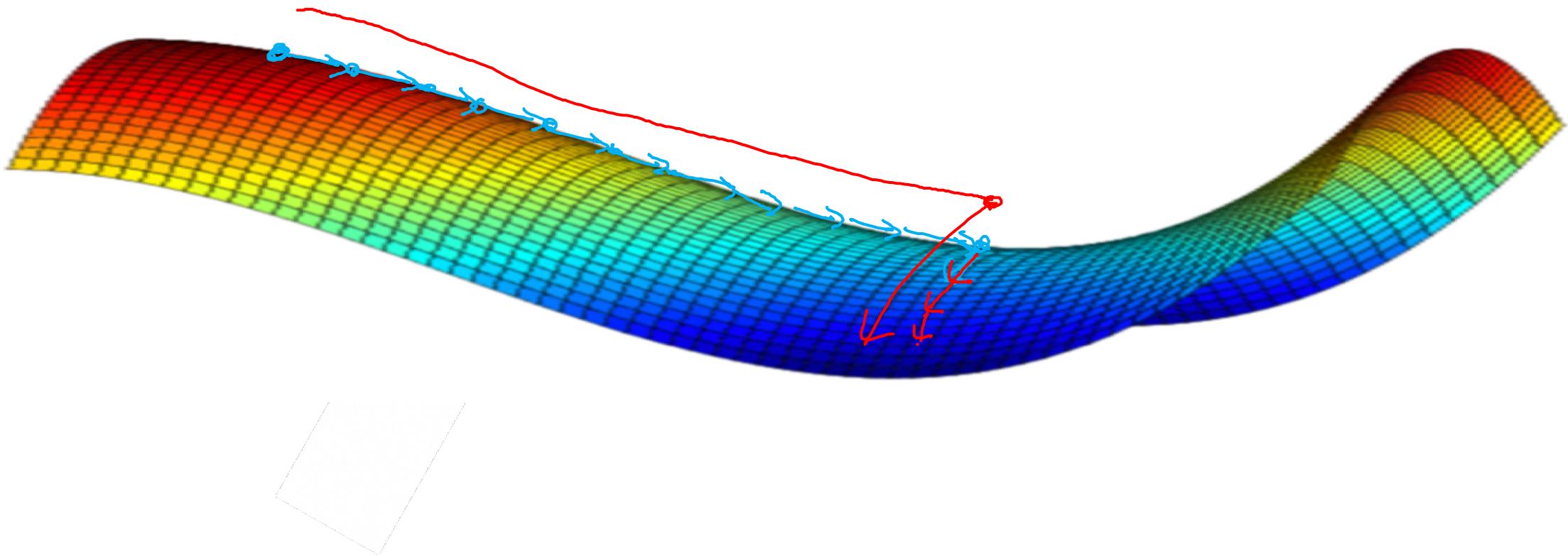
The problem of local optima

Local optima in neural networks



Andrew Ng

Problem of plateaus



- Unlikely to get stuck in a bad local optima
- Plateaus can make learning slow

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>

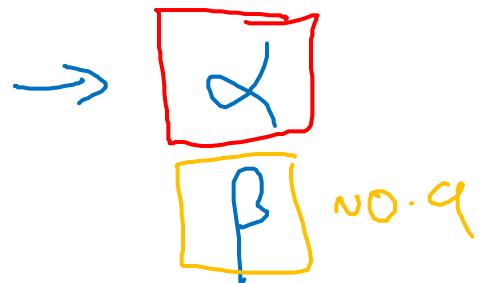


deeplearning.ai

Hyperparameter tuning

Tuning process

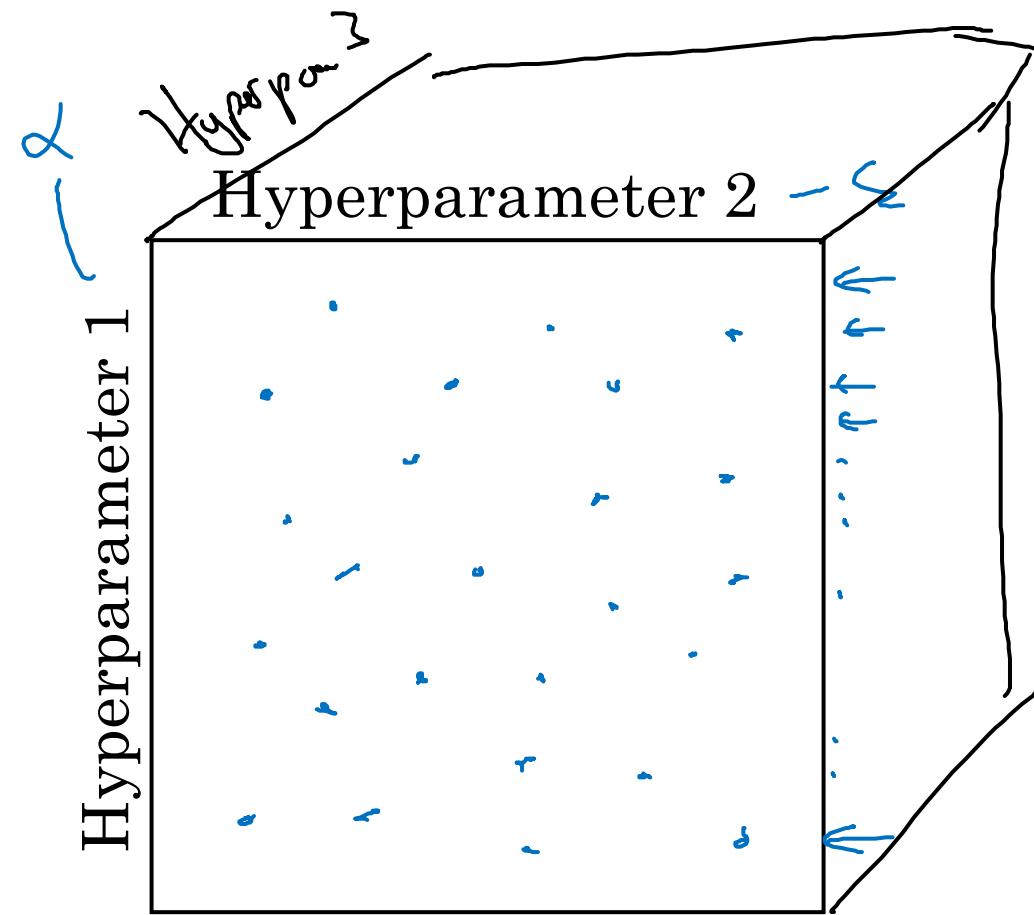
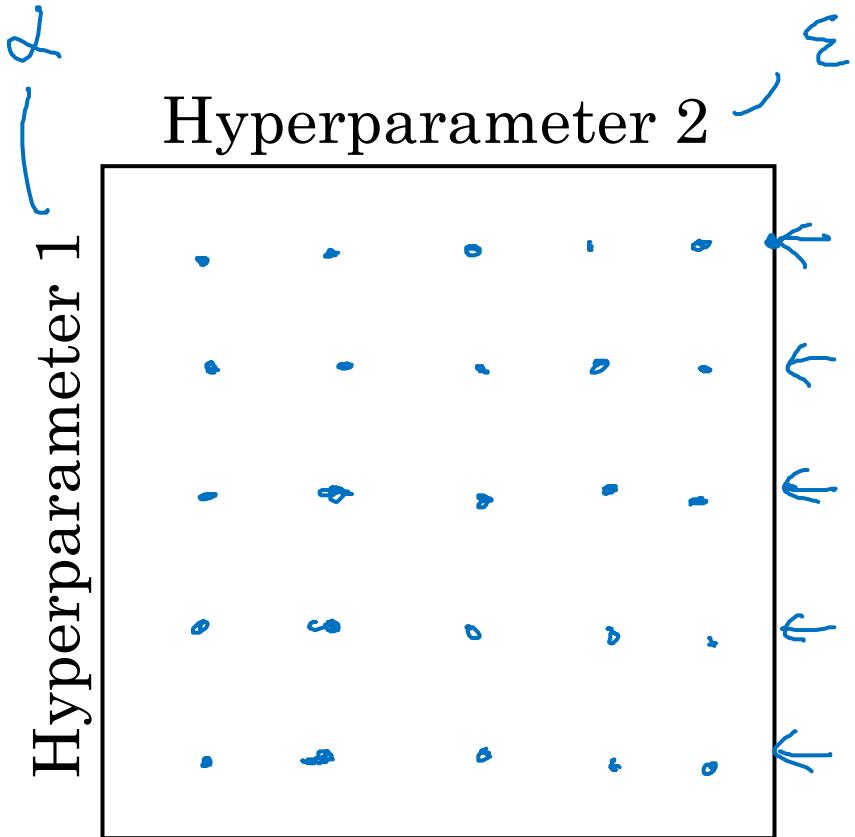
Hyperparameters



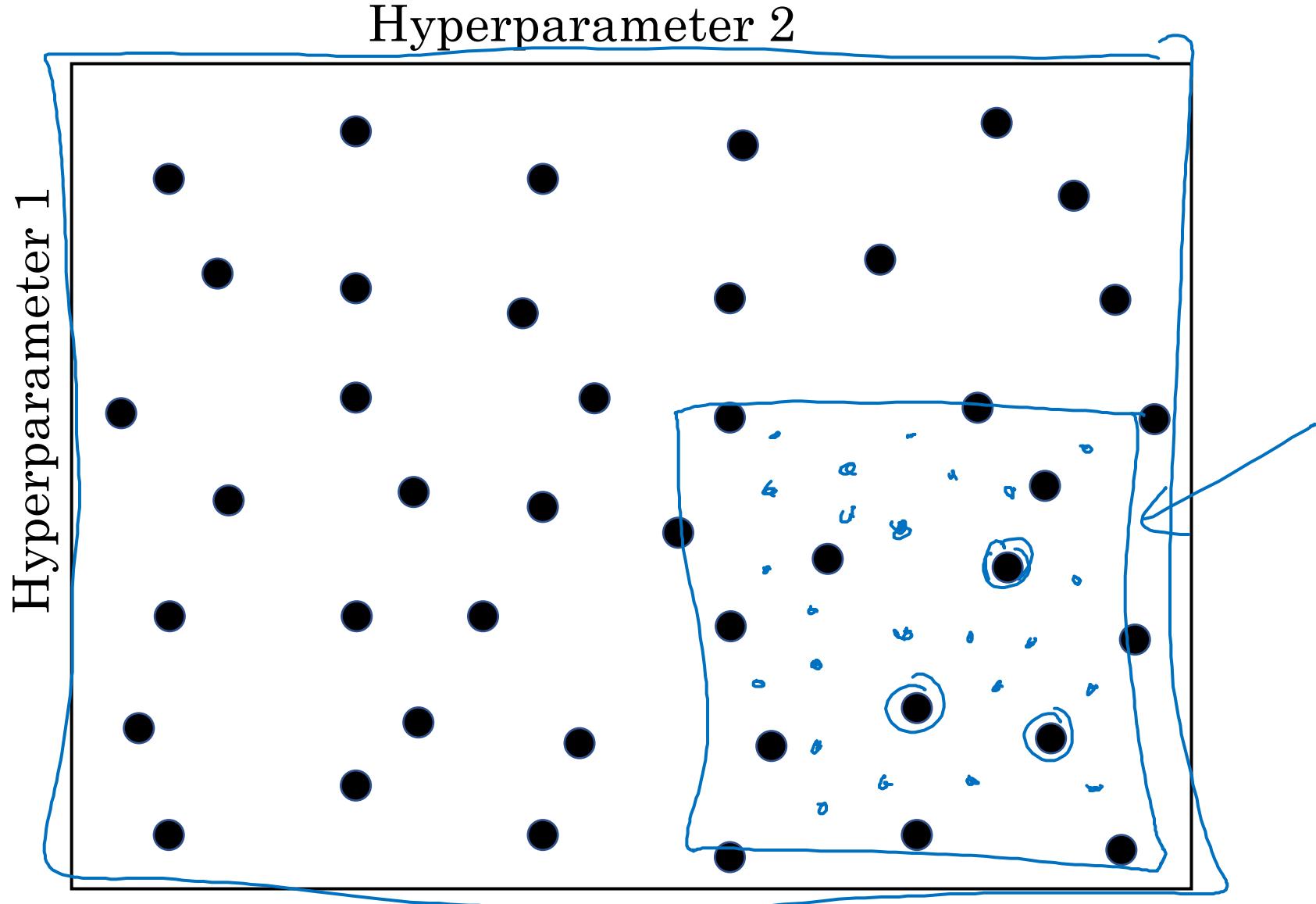
$\beta_1, \beta_2, \epsilon$
0.9 0.999 10^{-8}

- # layers
- # hidden units
- learning rate decay
- mini-batch size

Try random values: Don't use a grid



Coarse to fine





deeplearning.ai

Hyperparameter tuning

Using an appropriate
scale to pick
hyperparameters

Picking hyperparameters at random

→ $n^{[l]} = 50, \dots, 100$

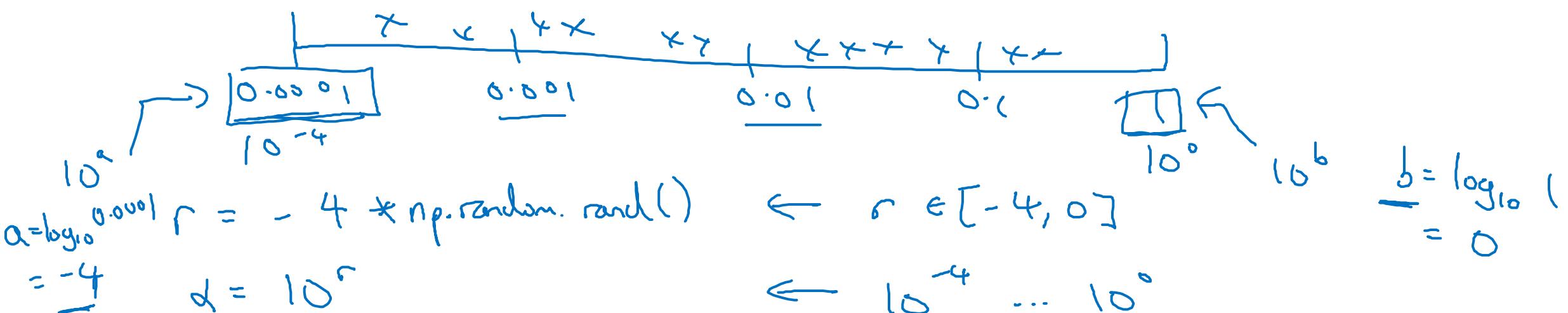
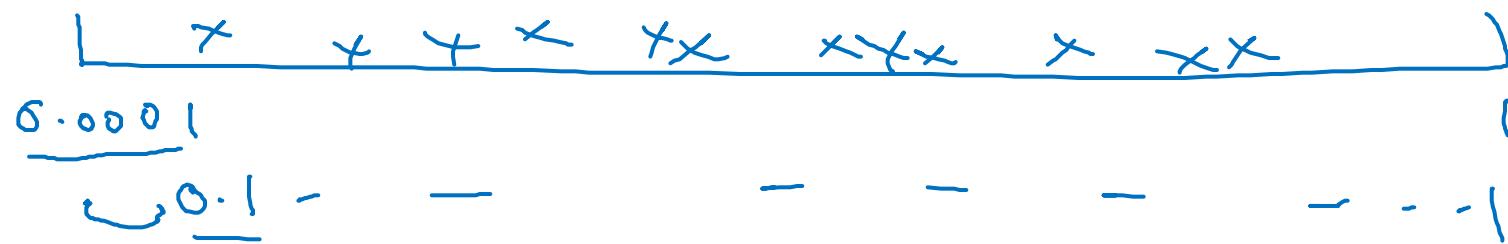


→ #layers $L : 2 - 4$

2, 3, 4

Appropriate scale for hyperparameters

$$\lambda = 0.0001, \dots, 1$$



$$10^a \dots 10^b$$

$$\frac{r \in [a, b]}{[-4, 0]}$$

$$\lambda = 10^r$$

Hyperparameters for exponentially weighted averages

$$\beta = 0.9 \dots 0.999$$

\downarrow \downarrow
 10 1000

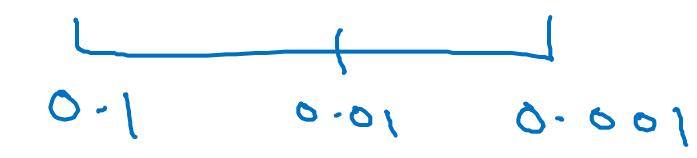
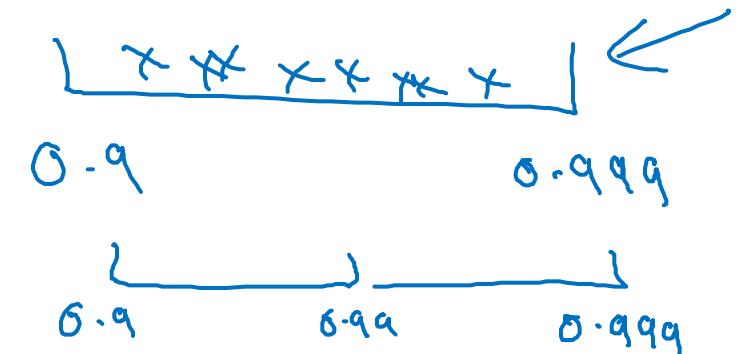
$$1-\beta = 0.1 \dots 0.001$$

$$\beta: 0.900 \rightarrow 0.9005 \quad \} \sim 10$$

$$\beta: 0.999 \rightarrow 0.9995$$

~ 1000 ~ 2000

$$\frac{1}{1-\beta}$$



$$\frac{10^{-1}}{1-\beta} \quad \frac{10^{-3}}{1-\beta}$$

$r \in [-3, -1]$

$$1-\beta = 10^r$$
$$\beta = 1 - 10^r$$

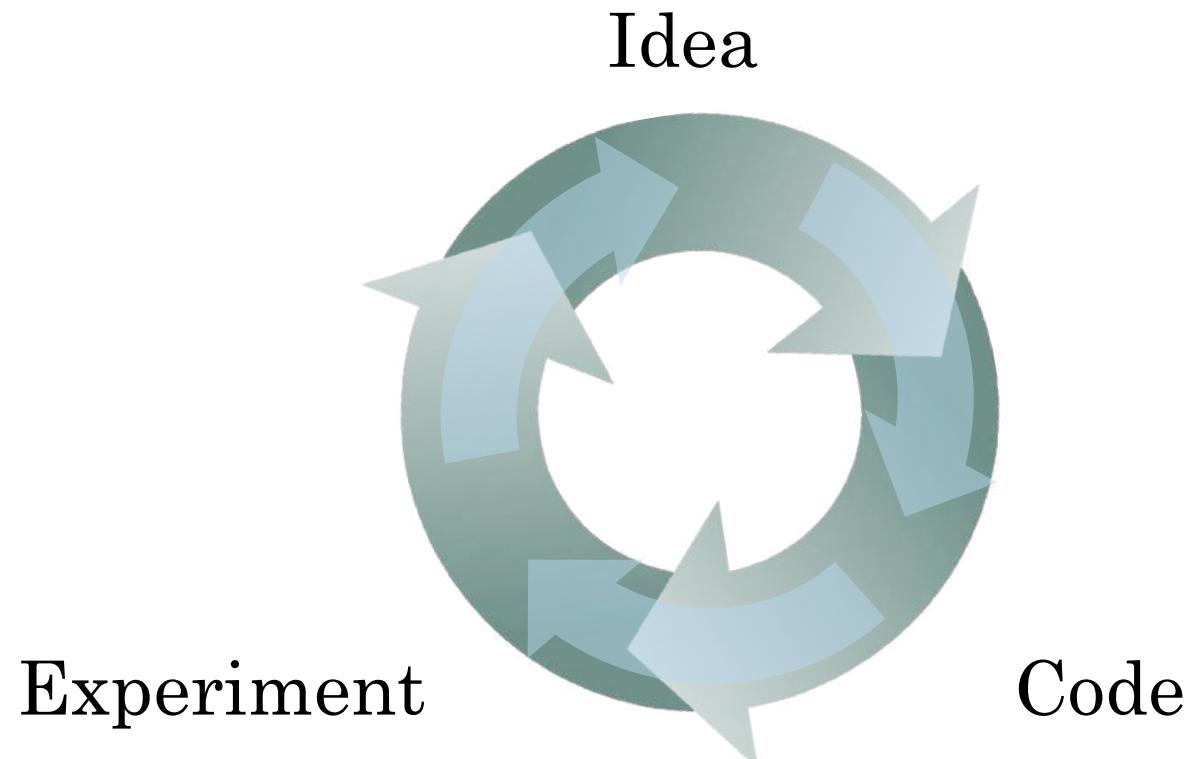


deeplearning.ai

Hyperparameters tuning

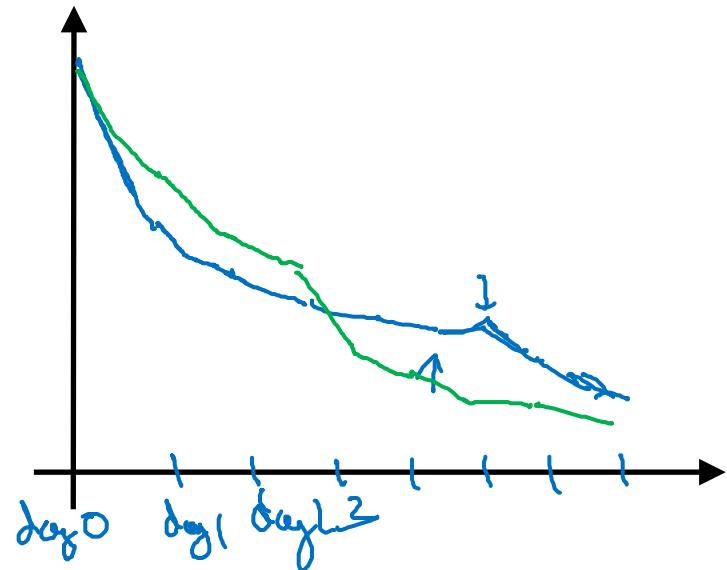
Hyperparameters tuning in practice: Pandas vs. Caviar

Re-test hyperparameters occasionally



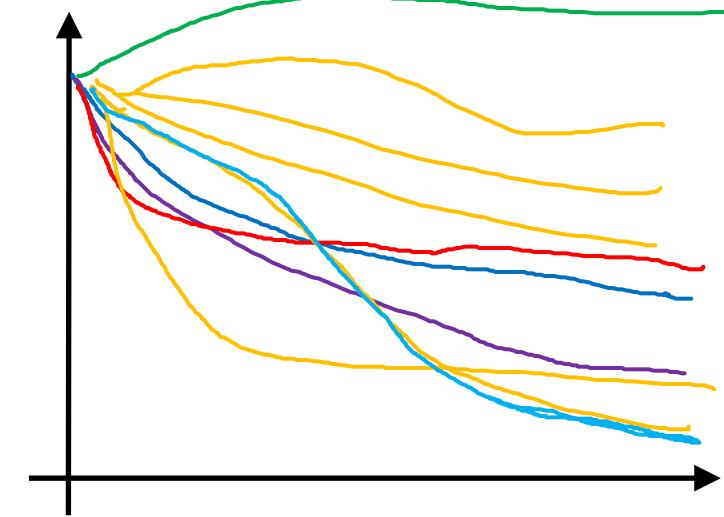
- NLP, Vision, Speech,
Ads, logistics,
- Intuitions do get stale.
Re-evaluate occasionally.

Babysitting one model



Panda ↵

Training many models in parallel



Caviar ↵

Andrew Ng

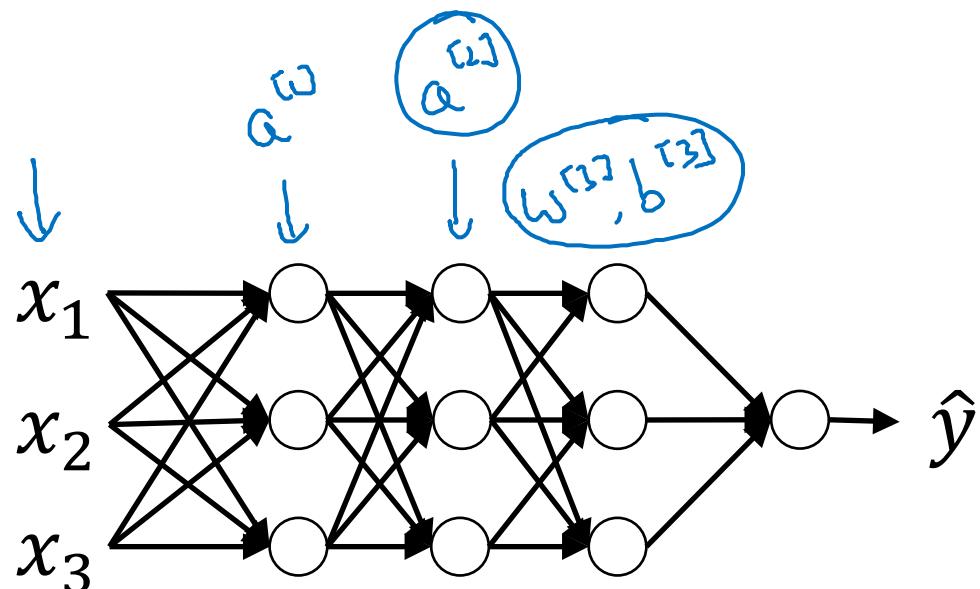
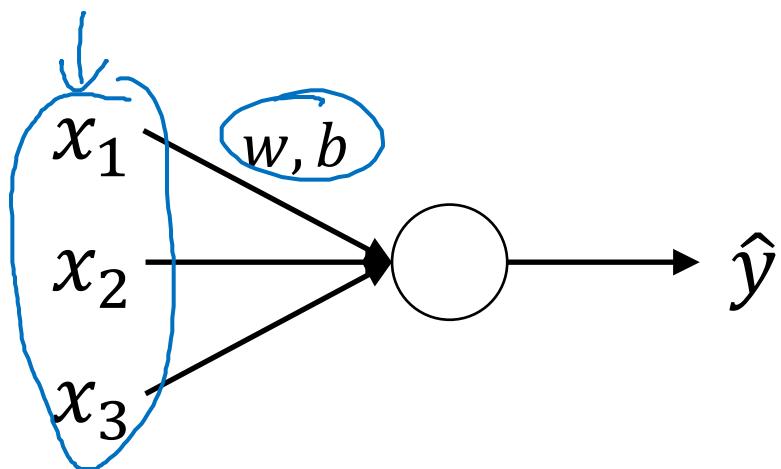


deeplearning.ai

Batch Normalization

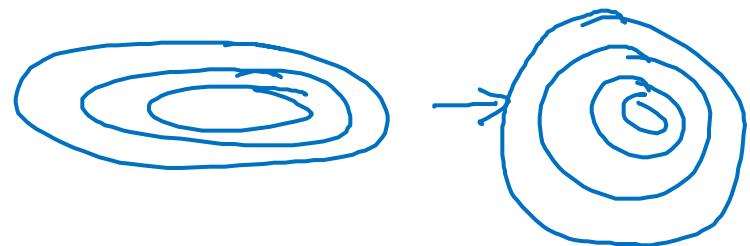
Normalizing activations in a network

Normalizing inputs to speed up learning



$$\mu = \frac{1}{m} \sum_i x^{(i)}$$
$$X = X - \mu$$
$$\sigma^2 = \frac{1}{m} \sum_i (x^{(i)})^2$$
$$X = X / \sigma^2$$

electr-wiss



Can we normalize $\frac{a^{[2]}}{w^{[2]}, b^{[2]}}$ so
as to train $w^{[2]}, b^{[2]}$ faster

Normalize $\frac{z^{[2]}}{\uparrow}$

Implementing Batch Norm

Given some intermediate values in NN

$$\mu = \frac{1}{m} \sum_i z^{(i)}$$

$$\sigma^2 = \frac{1}{m} \sum_i (z^{(i)} - \mu)^2$$

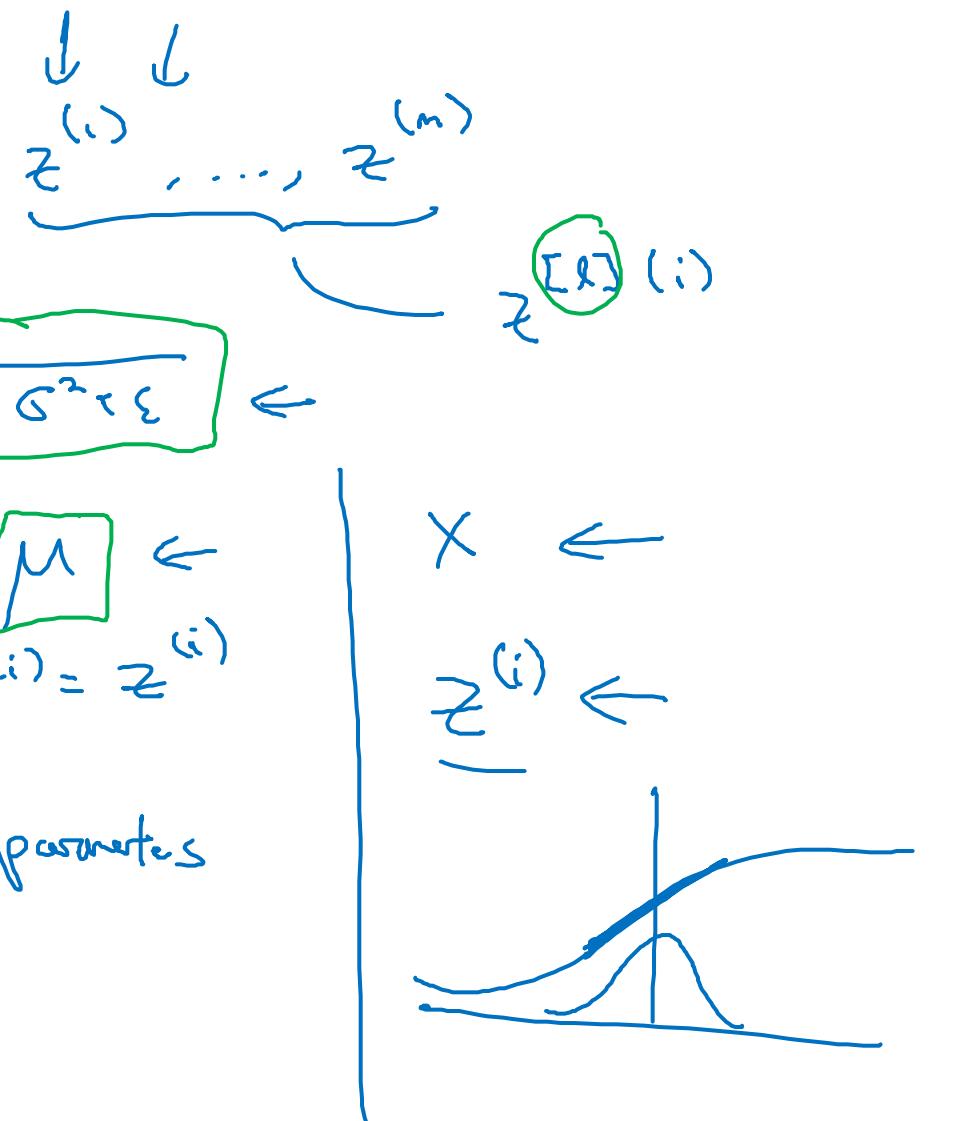
$$z_{\text{norm}}^{(i)} = \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \epsilon}}$$

$$\hat{z}^{(i)} = \gamma z_{\text{norm}}^{(i)} + \beta$$

Use $\hat{z}^{(i)}$ instead of $z^{(i)}$.

If $\gamma = \sqrt{\sigma^2 + \epsilon}$ ←
then $\beta = \mu$ ←
 $\hat{z}^{(i)} = z^{(i)}$

learnable parameters
of model.



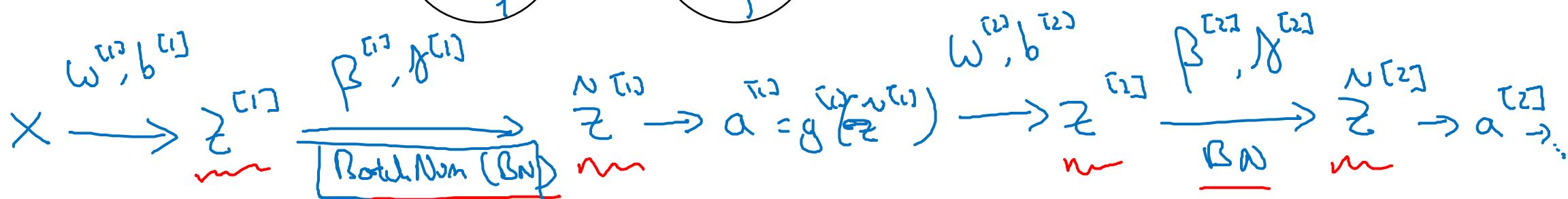
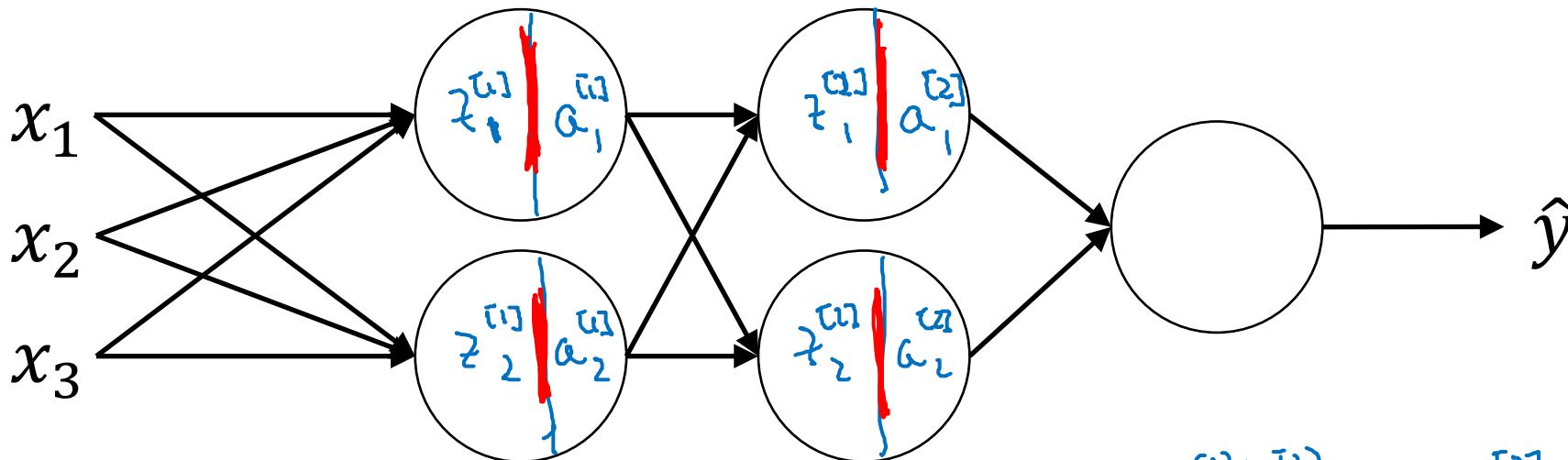


deeplearning.ai

Batch Normalization

Fitting Batch Norm into a neural network

Adding Batch Norm to a network



Parameters:

$$\left\{ w^{[1]}, b^{[1]}, w^{[2]}, b^{[2]}, \dots, w^{[L]}, b^{[L]}, \right.$$

$$\left. \rightarrow \beta^{[1]}, \gamma^{[1]}, \beta^{[2]}, \gamma^{[2]}, \dots, \beta^{[L]}, \gamma^{[L]} \right\}$$

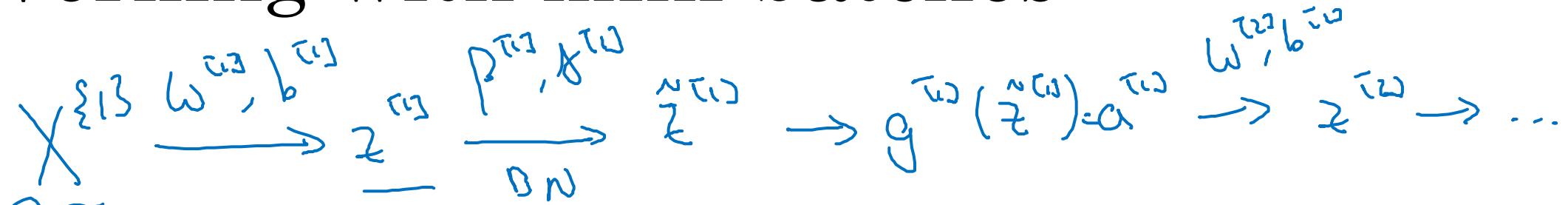
$$\rightarrow \beta$$

$$\frac{d\beta^{[l]}}{d\beta} = \frac{\tau^{[l]}}{\tau^{[l]}}$$

$$\beta = \beta - \frac{1}{n} \sum \frac{\tau^{[l]}}{\tau^{[l]}}$$

`tf.nn.batch_normalization` ←

Working with mini-batches



$X^{S2} \rightarrow \dots$

Parameters: $\omega^{[l]}, \cancel{b^{[l]}}, \beta^{[l]}, \gamma^{[l]}$.

$$\underline{z}^{[l]} \\ (n^{[l]}, 1)$$

$$\begin{matrix} \cancel{b^{[l]}} \\ | \\ (n^{[l]}, 1) \end{matrix} \quad \begin{matrix} \beta^{[l]} \\ | \\ (n^{[l]}, 1) \end{matrix} \quad \begin{matrix} \gamma^{[l]} \\ | \\ (n^{[l]}, 1) \end{matrix}$$

$$\begin{aligned} \underline{z}^{[l]} &= \omega^{[l]} \underline{a}^{[l-1]} + \cancel{b^{[l]}} \\ \underline{z}^{[l]} &= \omega^{[l]} \underline{a}^{[l-1]} \\ \underline{z}^{[l]}_{norm} &= \gamma^{[l]} \underline{z}^{[l]}_{norm} + \beta^{[l]} \end{aligned}$$

Andrew Ng

Implementing gradient descent

for $t = 1 \dots \text{num MiniBatches}$
Compute forward prop on $X^{[t]}$.

In each hidden layer, use BN to replace $\underline{z}^{[l]}$ with $\hat{\underline{z}}^{[l]}$.

Use backprop to compute $\underline{dw}^{[l]}$, ~~$\underline{db}^{[l]}$~~ , $\underline{d\beta}^{[l]}$, $\underline{dg}^{[l]}$

Update parameters $\left. \begin{array}{l} w^{[l]} := w^{[l]} - \alpha \underline{dw}^{[l]} \\ \beta^{[l]} := \beta^{[l]} - \alpha \underline{d\beta}^{[l]} \\ g^{[l]} := \dots \end{array} \right\} \leftarrow$

Works w/ momentum, RMSprop, Adam.

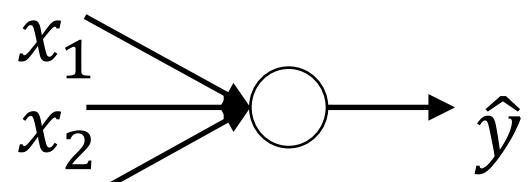


deeplearning.ai

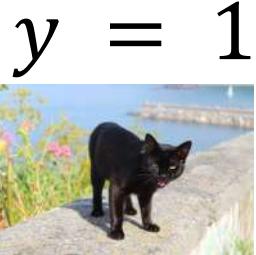
Batch Normalization

Why does
Batch Norm work?

Learning on shifting input distribution



Cat



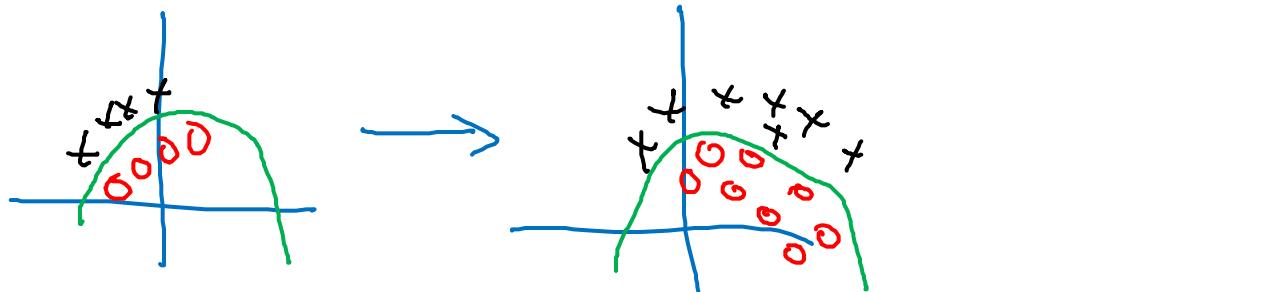
Non-Cat



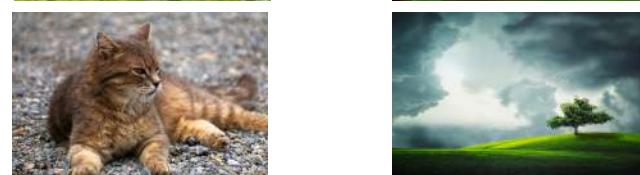
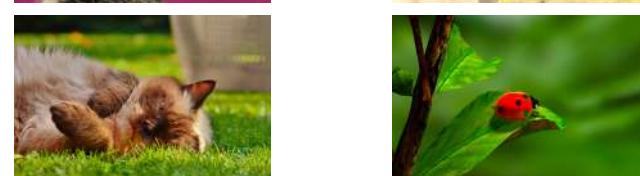
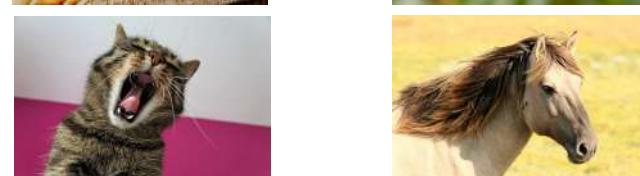
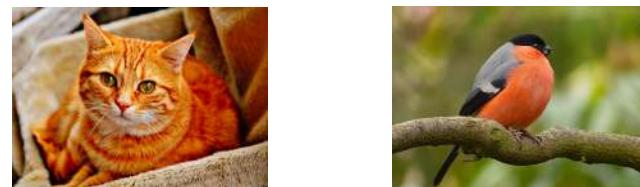
$$y = 1$$



$$y = 0$$



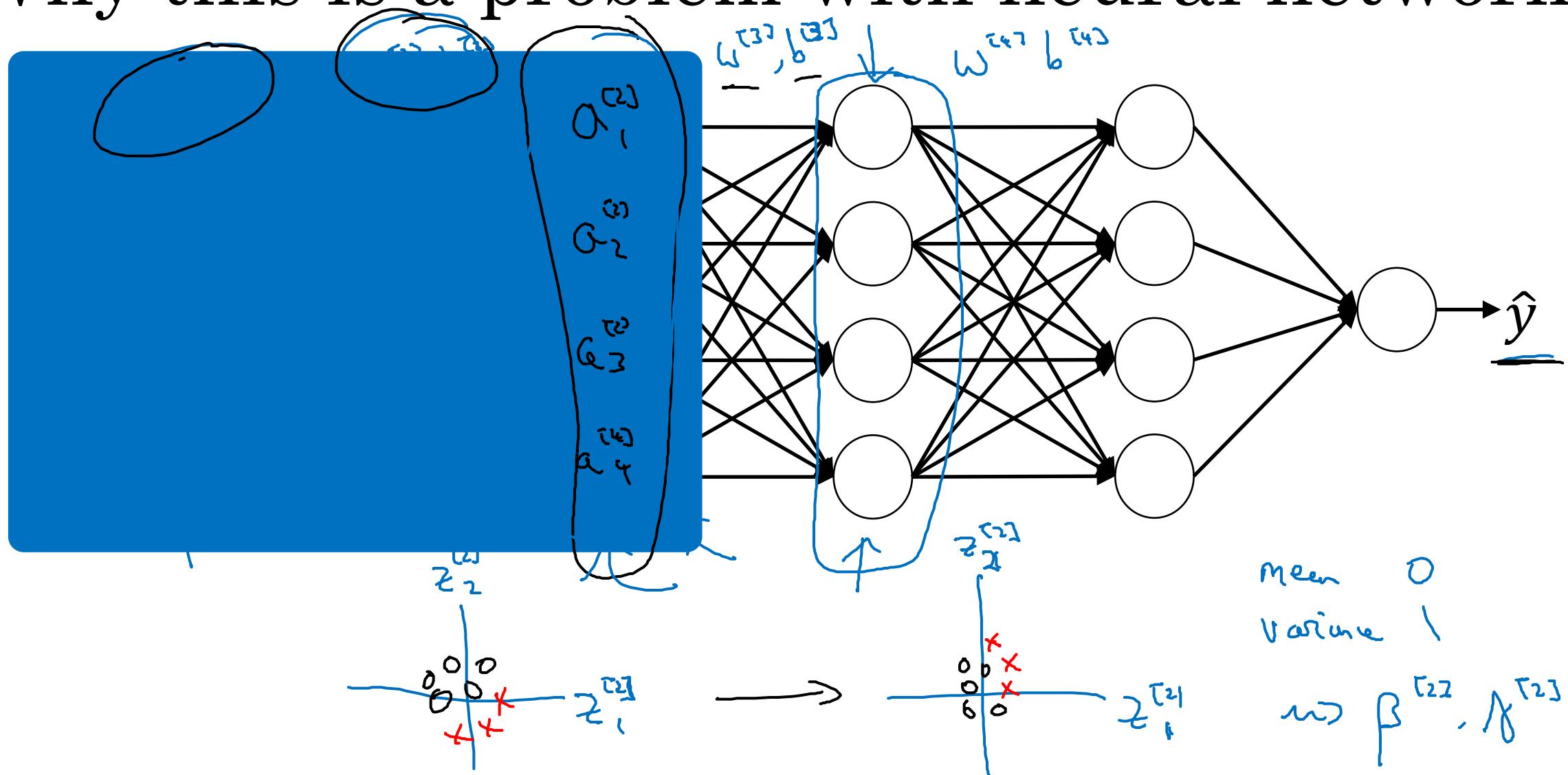
$$y = 1 \quad y = 0$$



"Covariate shift"

$$\underline{x} \rightarrow y$$

Why this is a problem with neural networks?



Batch Norm as regularization

X

- Each mini-batch is scaled by the mean/variance computed on just that mini-batch.
 $\xrightarrow{\hat{z}^{[l]}}$ μ, σ^2 $\{z^{[l]}\}$
 $\hat{z}^{[l]}$ $\underline{64}, \underline{128}$
- This adds some noise to the values $z^{[l]}$ within that minibatch. So similar to dropout, it adds some noise to each hidden layer's activations.
 μ, σ^2
- This has a slight regularization effect.

mini-batch : 64 \longrightarrow 512



deeplearning.ai

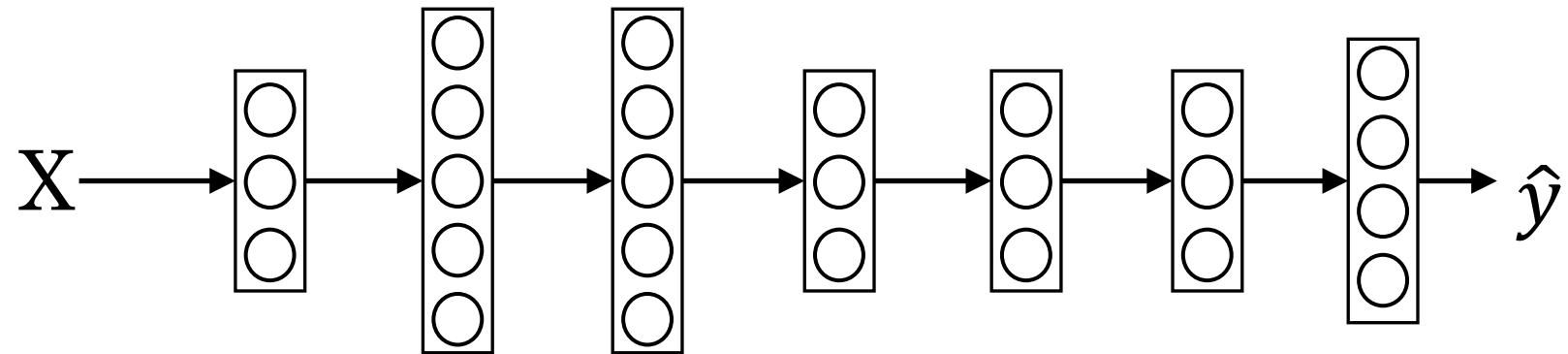
Multi-class
classification

Softmax regression

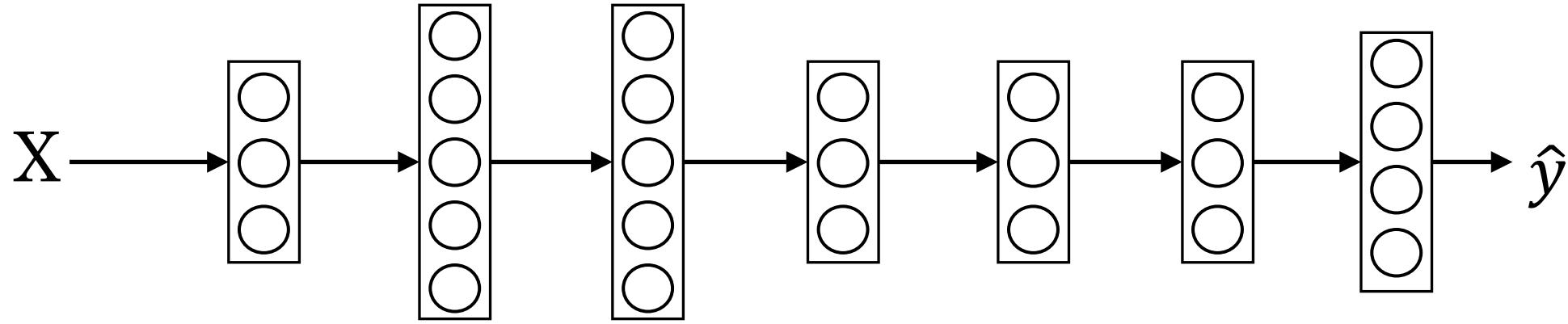
Recognizing cats, dogs, and baby chicks



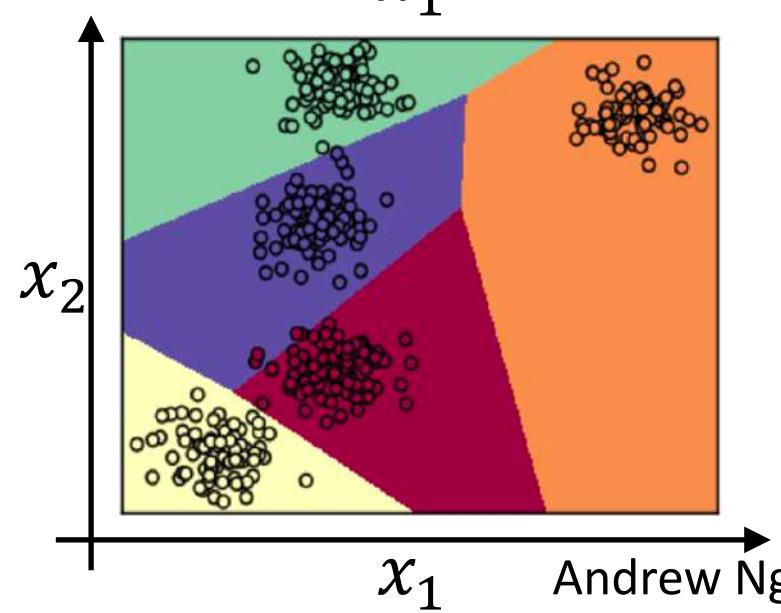
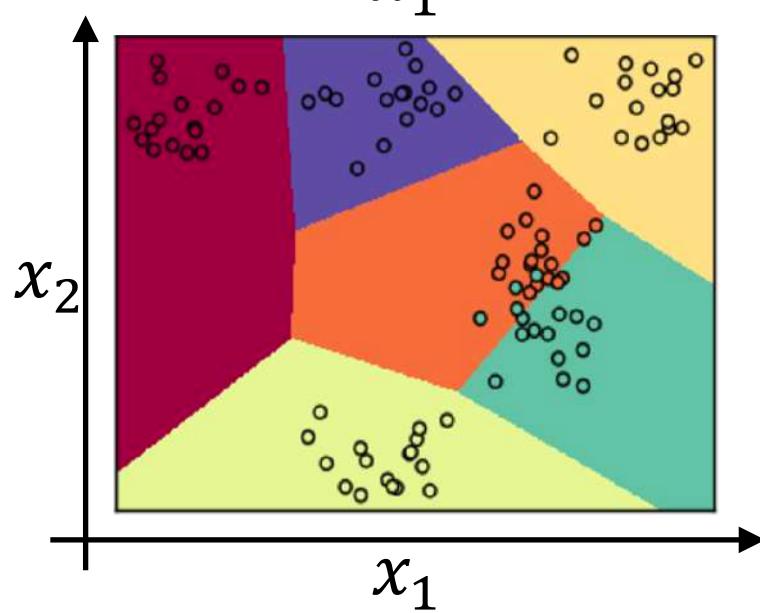
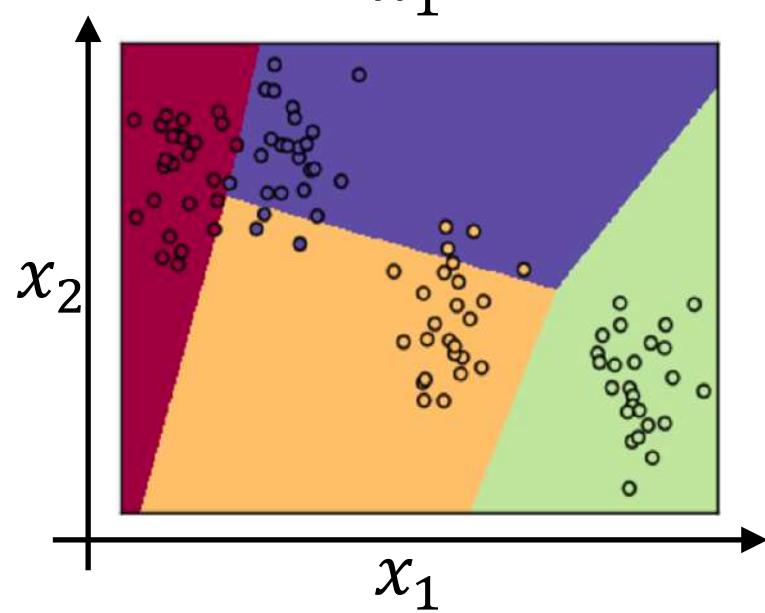
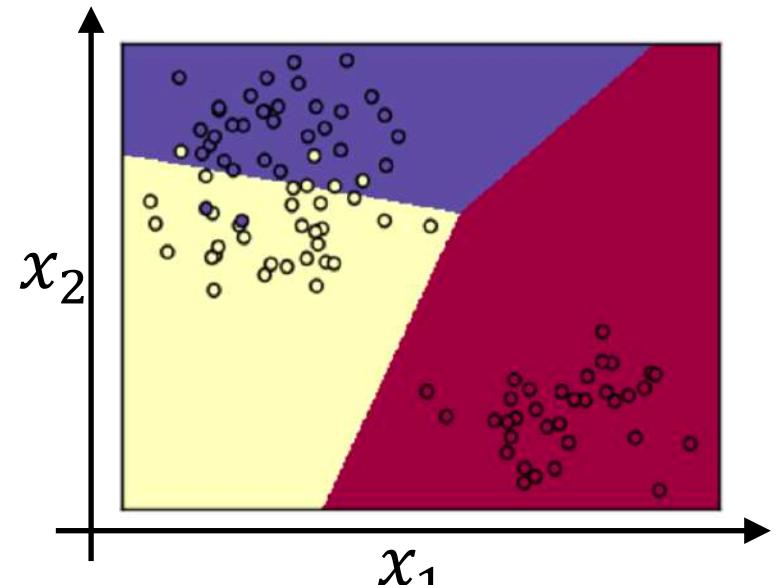
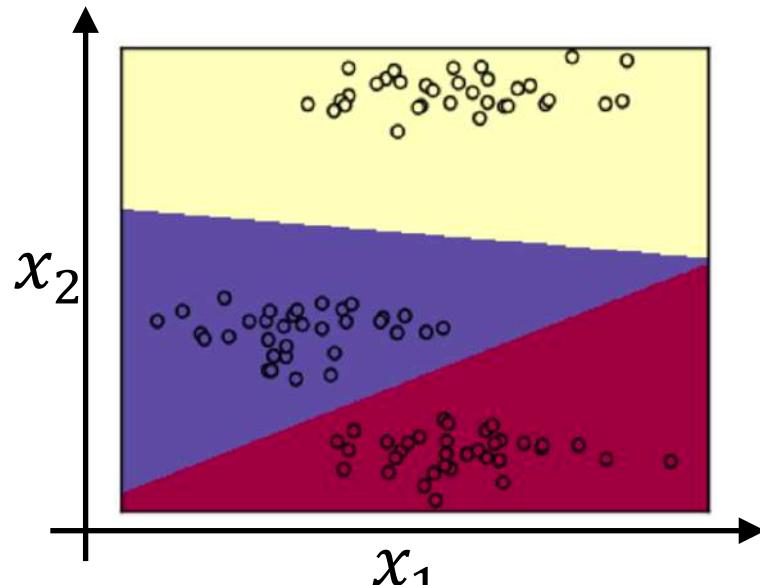
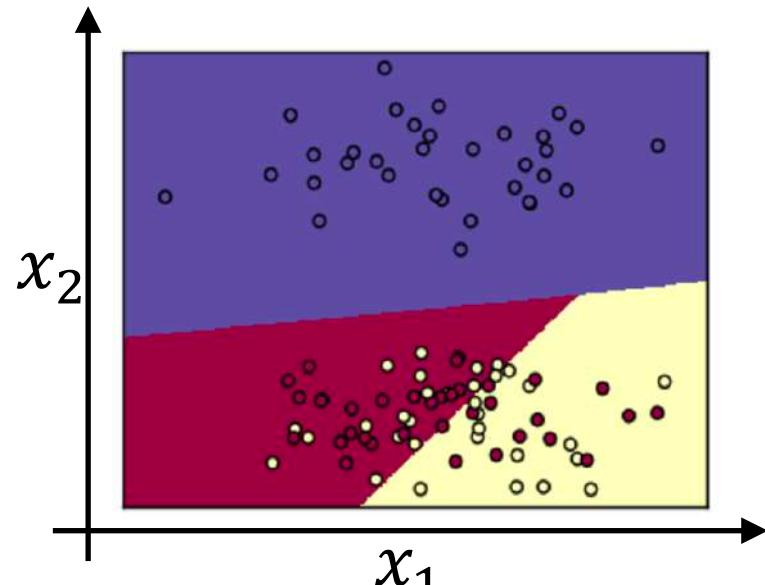
3 1 2 0 3 2 0 1



Softmax layer



Softmax examples





deeplearning.ai

Programming Frameworks

Deep Learning frameworks

Deep learning frameworks

- Caffe/Caffe2
- CNTK
- DL4J
- Keras
- Lasagne
- mxnet
- PaddlePaddle
- TensorFlow
- Theano
- Torch

Choosing deep learning frameworks

- Ease of programming (development and deployment)
 - Running speed
- - Truly open (open source with good governance)



deeplearning.ai

Programming Frameworks

TensorFlow

Motivating problem

$$J(\omega) = \frac{(\omega^2 - 10\omega + 25)}{(\omega - 5)^2}$$

$\omega = 5$

$J(\omega, b)$

↑ ↑

Code example

```
import numpy as np  
import tensorflow as tf  
  
coefficients = np.array([[1], [-20], [25]])
```

```
w = tf.Variable([0], dtype=tf.float32)
```

```
x = tf.placeholder(tf.float32, [3,1])
```

```
cost = x[0][0]*w**2 + x[1][0]*w + x[2][0] # (w-5)**2
```

```
train = tf.train.GradientDescentOptimizer(0.01).minimize(cost)
```

```
init = tf.global_variables_initializer()
```

```
session = tf.Session()
```

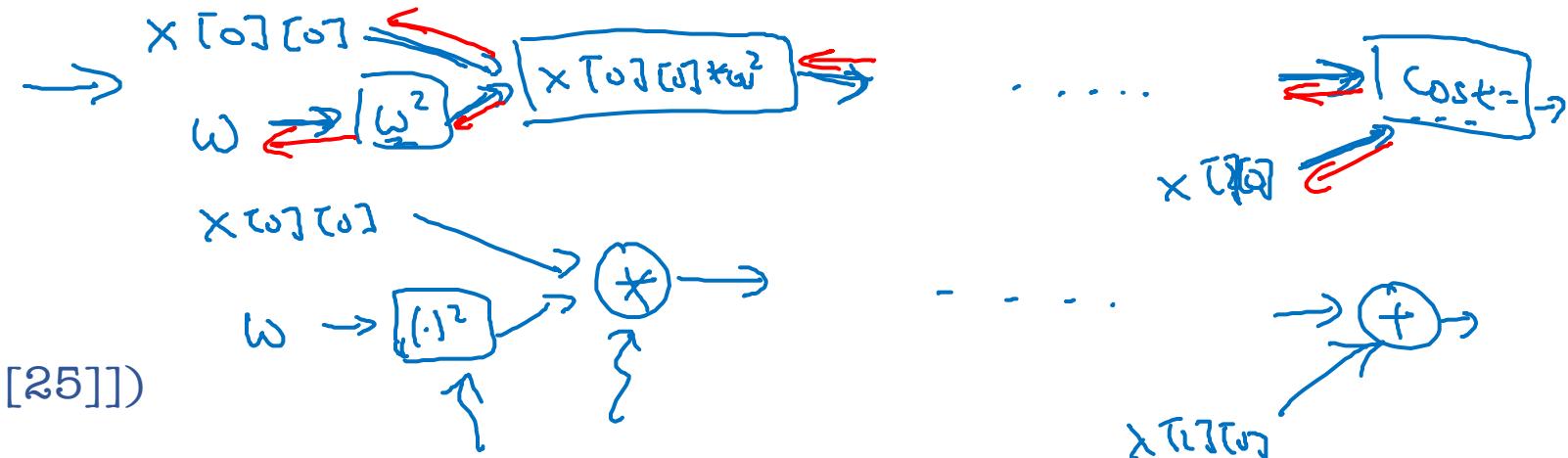
```
session.run(init)
```

```
print(session.run(w))
```

```
for i in range(1000):
```

```
    session.run(train, feed_dict={x:coefficients})
```

```
print(session.run(w))
```



```
with tf.Session() as session:
```

```
    session.run(init)
```

```
    print(session.run(w))
```

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

Introduction to ML strategy

Why ML Strategy?

Motivating example



96%.

Ideas:

- Collect more data ←
- Collect more diverse training set
- Train algorithm longer with gradient descent
- Try Adam instead of gradient descent
- Try bigger network
- Try smaller network
- Try dropout
- Add L_2 regularization
- Network architecture
 - Activation functions
 - # hidden units
 - ...

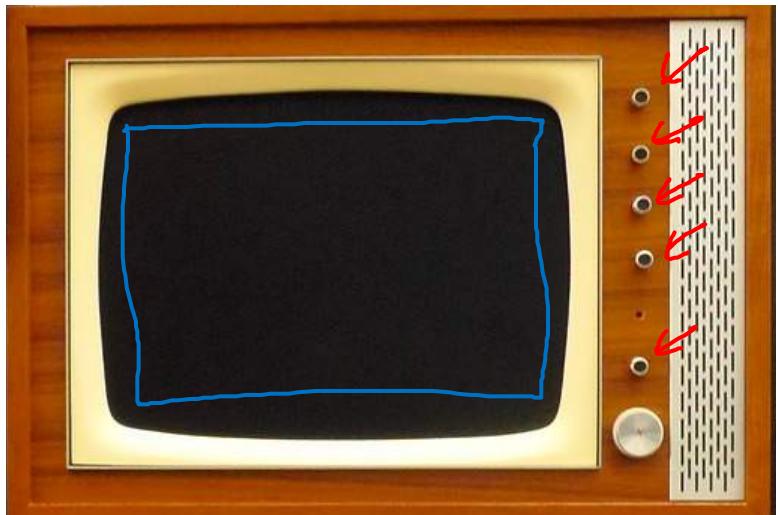


deeplearning.ai

Introduction to ML strategy

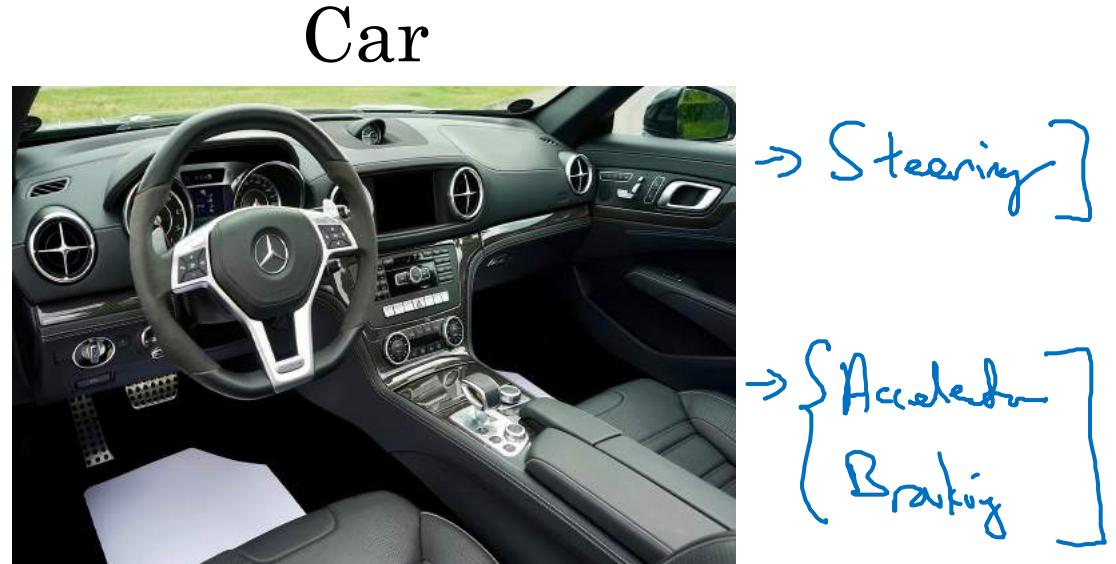
Orthogonalization

TV tuning example



Orthogonalization

$$\begin{aligned} & 0.1 \times \boxed{\uparrow \downarrow} \\ & + 0.3 \times \boxed{\leftarrow \rightarrow} \\ & - 1.7 \times \boxed{-} \\ & + 0.8 \times \boxed{\leftarrow \rightarrow} \\ & + \dots \end{aligned}$$



Car

\rightarrow Steering]
 \rightarrow Acceleration [Braking]

$$\rightarrow \underline{0.3 \times \text{angle}} - 0.8 \times \text{speed}$$

$$\rightarrow 2 \times \text{angle} + 0.9 \times \text{speed}.$$

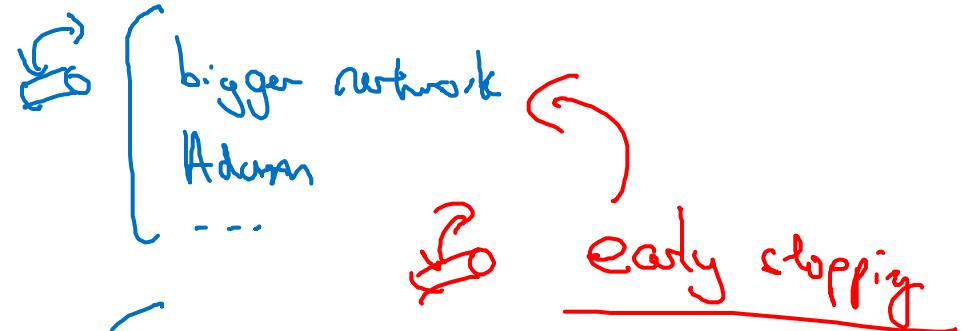


Chain of assumptions in ML

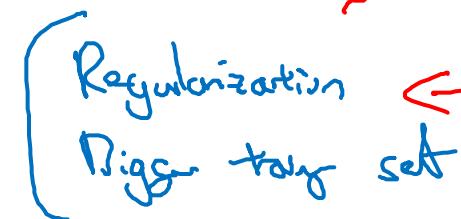
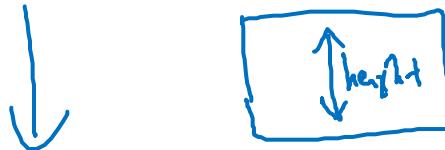
→ Fit training set well on cost function



(≈ human-level performance)



→ Fit dev set well on cost function



→ Fit test set well on cost function



Bigger dev set

→ Performs well in real world

(Happy cat pic off users.)

Change dev set or
cost function

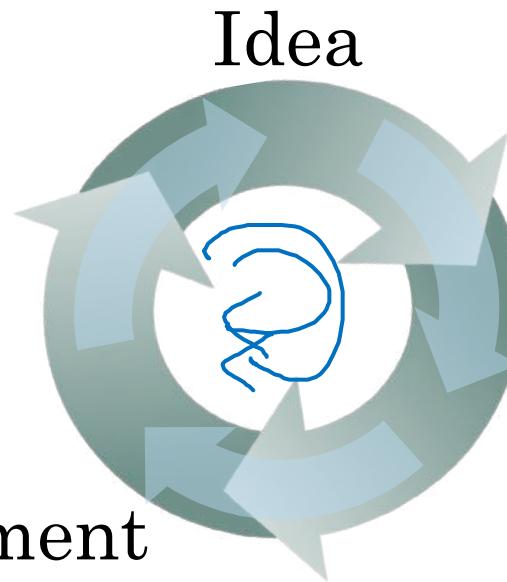


deeplearning.ai

Setting up
your goal

Single number
evaluation metric

Using a single number evaluation metric



Code

→ Of examples recognized as cert.,
what % actually are certs?

→ what % of actual certs
are correctly recognized

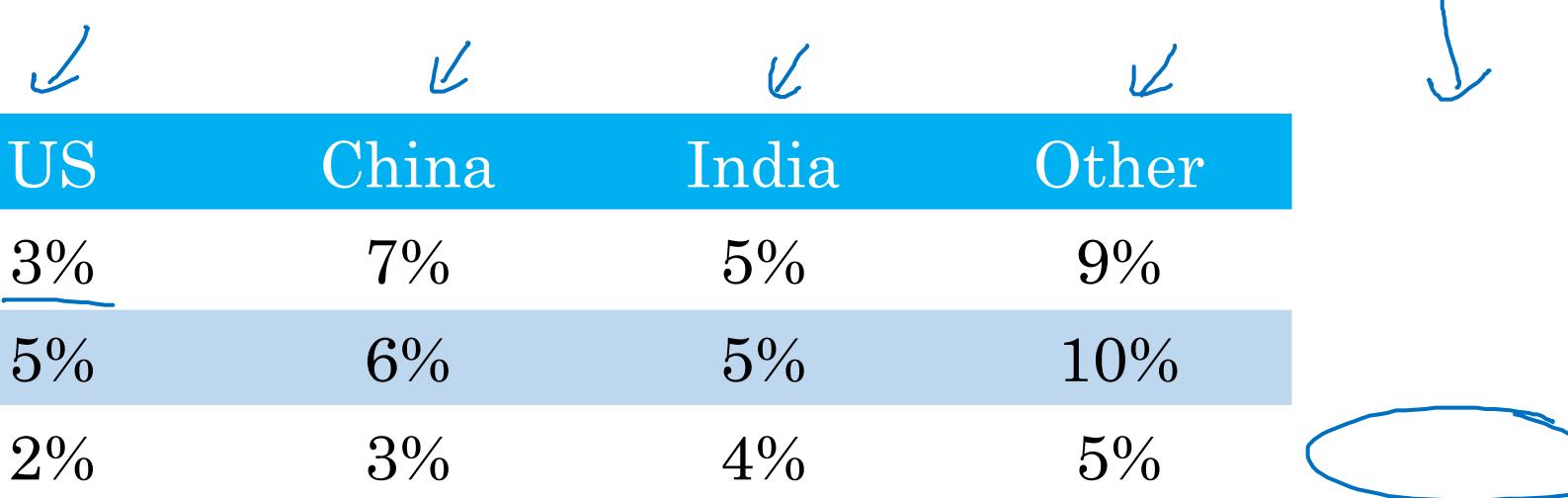
Classifier	Precision	Recall
A	<u>95%</u>	90%
B	98%	85%

F₁ Score = "Average" of P and R.

$$\left(\frac{2}{\frac{1}{P} + \frac{1}{R}} . \text{ "Harmonic Mean"} \right)$$

Dev set + Single number evaluation metric

Another example



Algorithm	US	China	India	Other
A	<u>3%</u>	7%	5%	9%
B	5%	6%	5%	10%
C	2%	3%	4%	5%
D	5%	8%	7%	2%
E	4%	5%	2%	4%
F	7%	11%	8%	12%



deeplearning.ai

Setting up
your goal

Satisficing and
optimizing metrics

Another cat classification example

Classifier	Accuracy	Running time
A	90%	80ms
B	92%	95ms
C	95%	1,500ms

$$\text{Cost} = \underline{\text{accuracy}} - 0.5 \times \underline{\text{running Time}}$$

Maximize accuracy

Subject to running Time \leq 100 ms.

N metrics : 1 optimizing

N-1 satisfying

optimizing



satisfying

Wakewords / Trigger words

Alexa, OK Google,

Hey Siri, nihao baidu

你好 百度

accuracy.

#false positive

Maximize accuracy.

s.t. ≤ 1 false positive
every 24 hours.



deeplearning.ai

Setting up
your goal

Train/dev/test
distributions

Cat classification dev/test sets

↳ development set, hold out cross validation set

Regions:

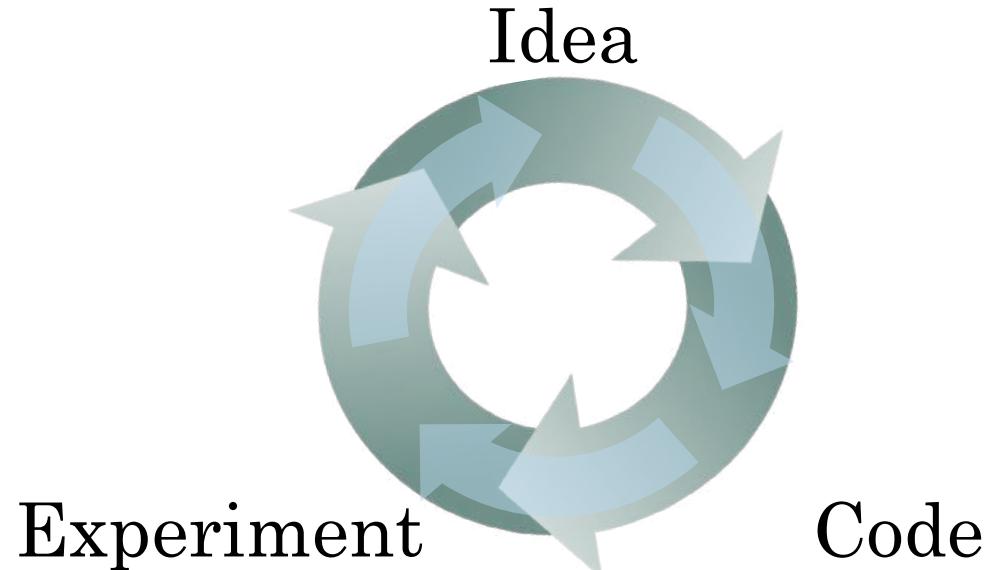
- US
- UK
- Other Europe
- South America
- India
- China
- Other Asia
- Australia



Randomly shuffle into dev/test



dev set
+
metric



True story (details changed)

[Optimizing on dev set on loan approvals for
medium income zip codes

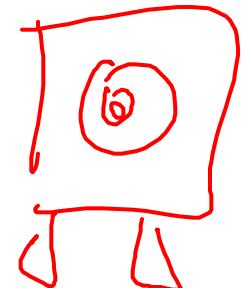


$$x \rightarrow y \text{ (repay loan?)} \quad \text{X} \rightarrow y$$



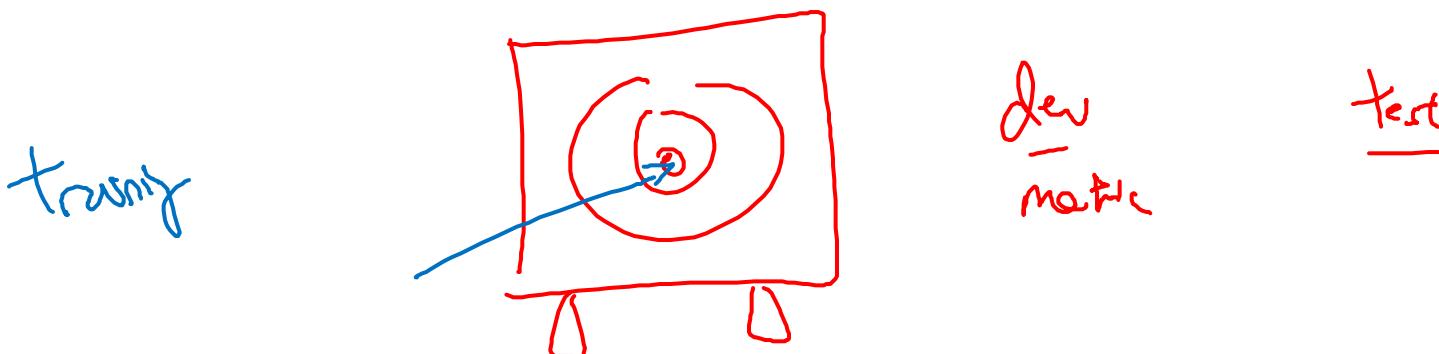
[Tested on low income zip codes

~ 3 month



Guideline

Choose a dev set and test set to reflect data you expect to get in the future and consider important to do well on.



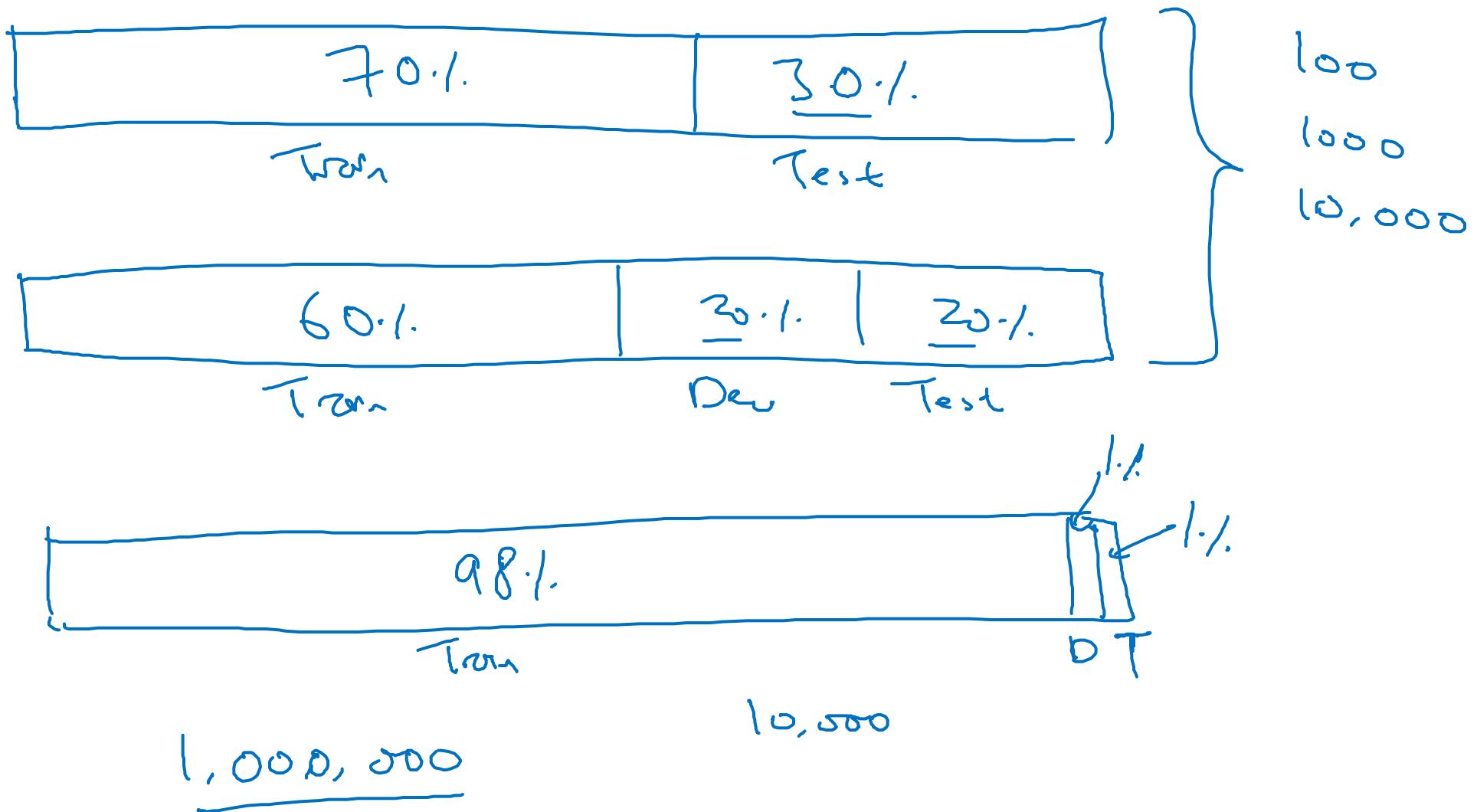


deeplearning.ai

Setting up
your goal

Size of dev
and test sets

Old way of splitting data



Size of dev set

A B

Set your dev set to be big enough to detect differences in
algorithm/models you're trying out.

100: small
 $\frac{1}{100} \approx 1\%$

A B
97% \rightarrow 97.1%
 $\frac{0.1\%}{1}$

1,000

10,000

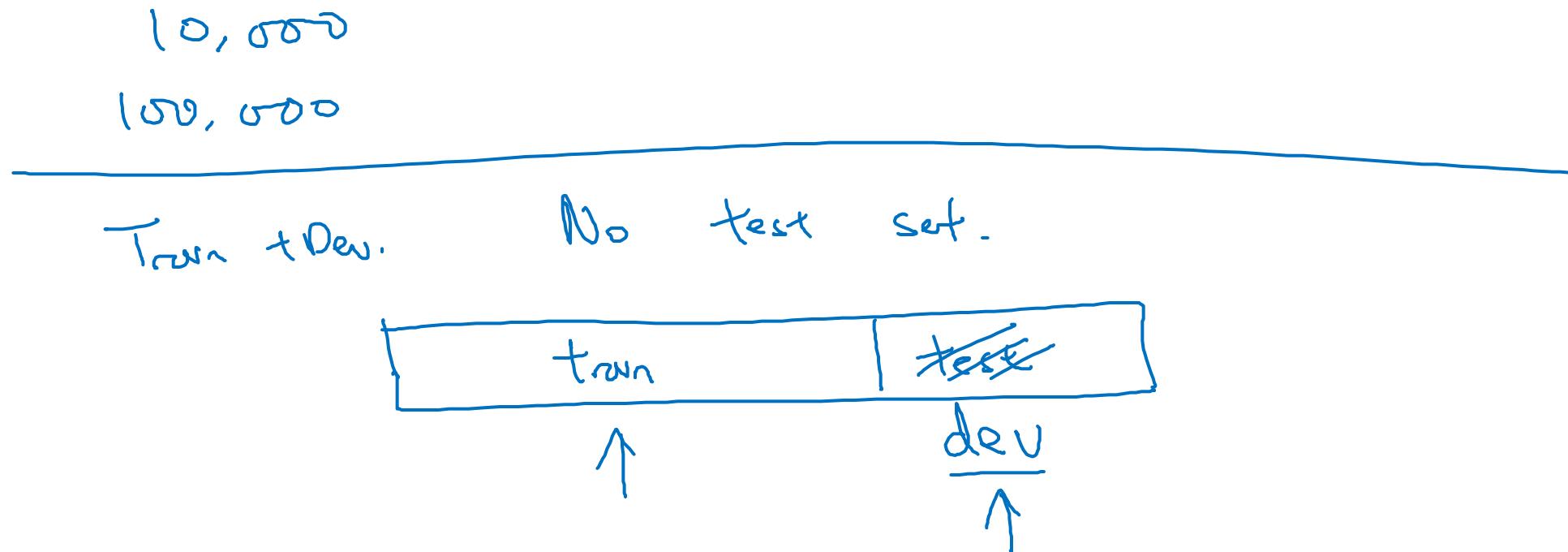
100,000

$\frac{0.01\%}{1}$
0.001%

Online advertising

Size of test set

→ Set your test set to be big enough to give high confidence in the overall performance of your system.





deeplearning.ai

Setting up
your goal

When to change
dev/test sets and
metrics

Cat dataset examples

Metric + Dev : Prefer A
You/users : Prefer B.

→ Metric: classification error

Algorithm A: 3% error → Pornographic

✓ Algorithm B: 5% error

$$\left\{ \begin{array}{l} \text{Error: } \frac{1}{\sum_i \omega^{(i)}} \cancel{\frac{1}{m_{\text{dev}}}} \quad \overbrace{\sum_{i=1}^{m_{\text{dev}}} \omega^{(i)}}^{\downarrow} \quad \left\{ \frac{y_{\text{pred}}^{(i)} + y^{(i)}}{\text{predicted value (0/1)}} \right\} \\ \rightarrow \omega^{(i)} = \begin{cases} 1 & \text{if } x^{(i)} \text{ is man-porn} \\ 10 & \text{if } x^{(i)} \text{ is porn} \end{cases} \end{array} \right.$$

Orthogonalization for cat pictures: anti-porn

- 1. So far we've only discussed how to define a metric to evaluate classifiers. ← Place target 
- 2. Worry separately about how to do well on this metric. 

An (shot at target)

$$\rightarrow J = \frac{1}{\sum w^{(i)}} \sum_{i=1}^m w^{(i)} \ell(\hat{y}^{(i)}, y^{(i)})$$



Another example

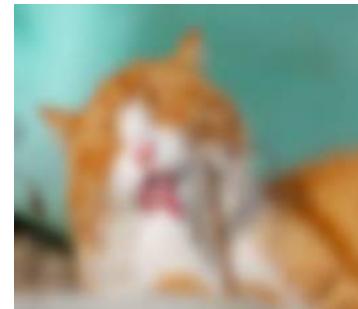
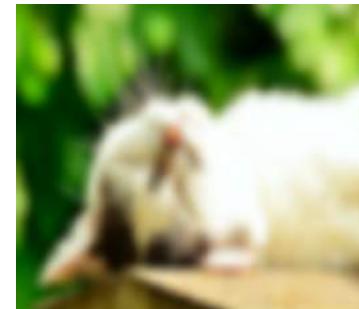
Algorithm A: 3% error

✓ Algorithm B: 5% error ↙

→ Dev/test ↘



→ User images ↗



If doing well on your metric + dev/test set does not correspond to doing well on your application, change your metric and/or dev/test set.

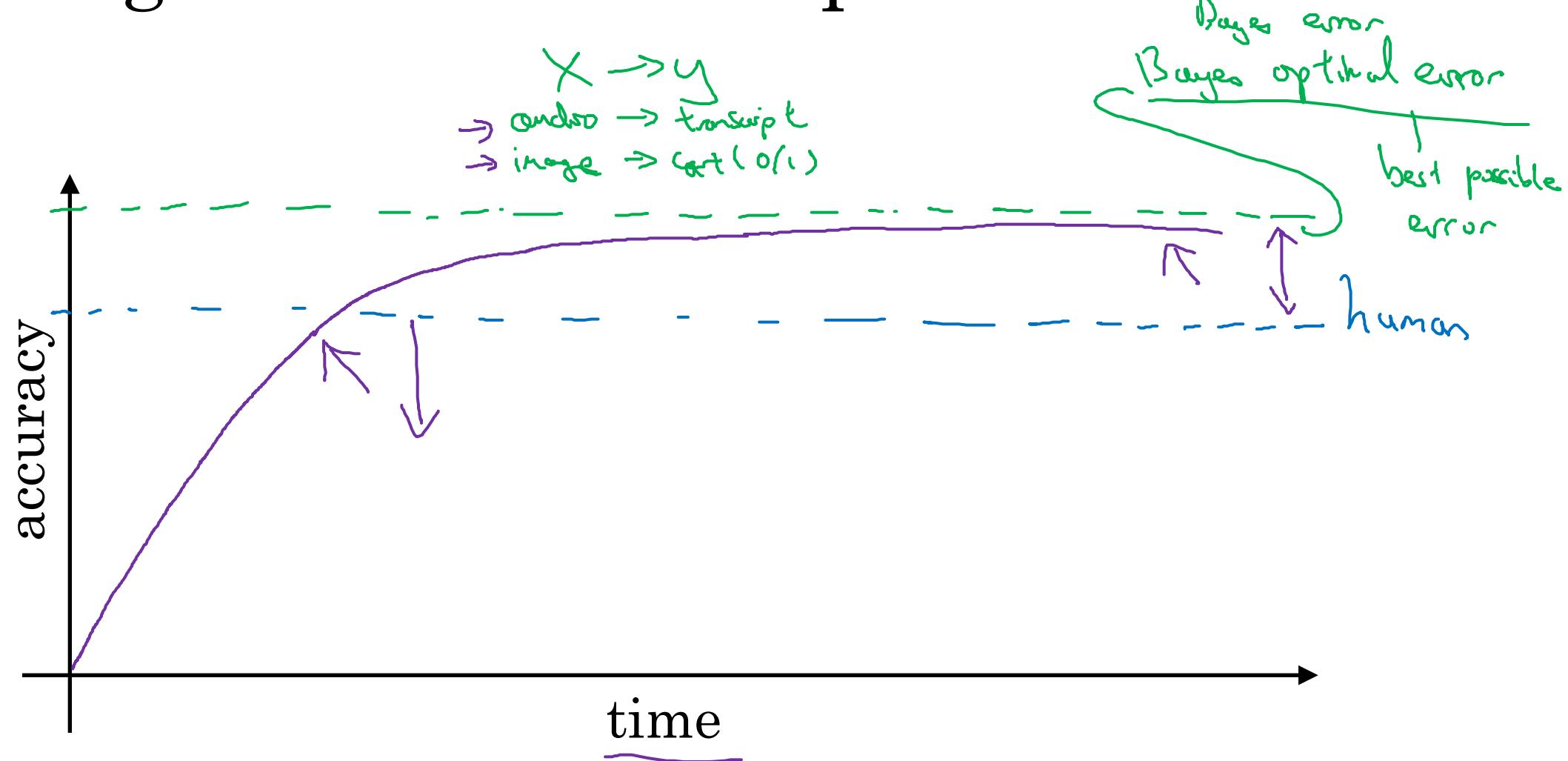


deeplearning.ai

Comparing to human-level performance

Why human-level performance?

Comparing to human-level performance



Why compare to human-level performance

Humans are quite good at a lot of tasks. So long as ML is worse than humans, you can:

- - Get labeled data from humans. (x, y)
- - Gain insight from manual error analysis:
Why did a person get this right?
- - Better analysis of bias/variance.

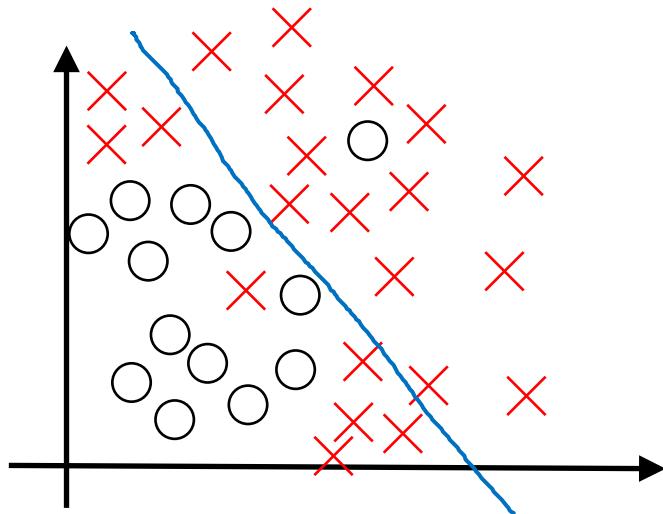


deeplearning.ai

Comparing to human-level performance

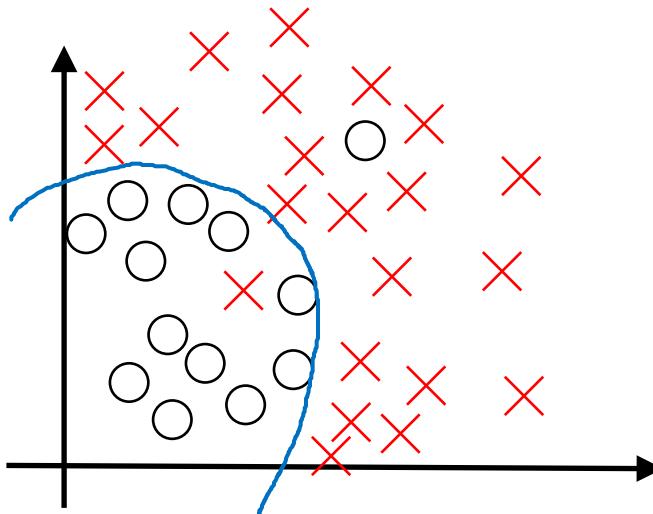
Avoidable bias

Bias and Variance

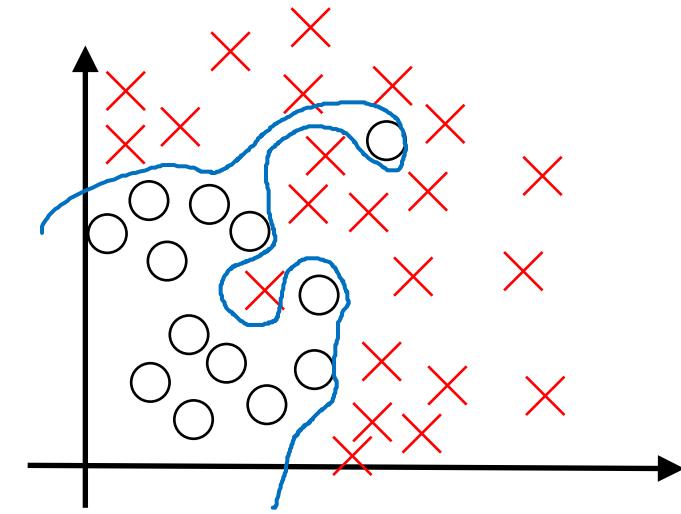


high bias

Underfitting



"just right"



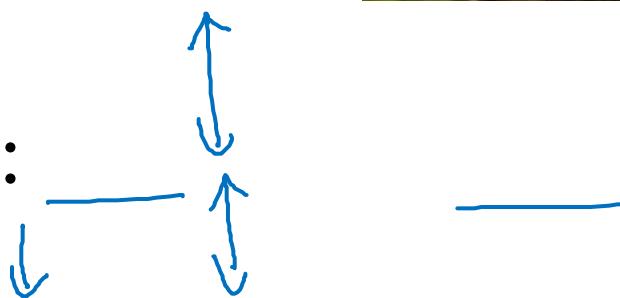
high variance

Overfitting

Bias and Variance

Cat classification

Human-level $\approx 0\%$



Training set error:

Dev set error:

high variance

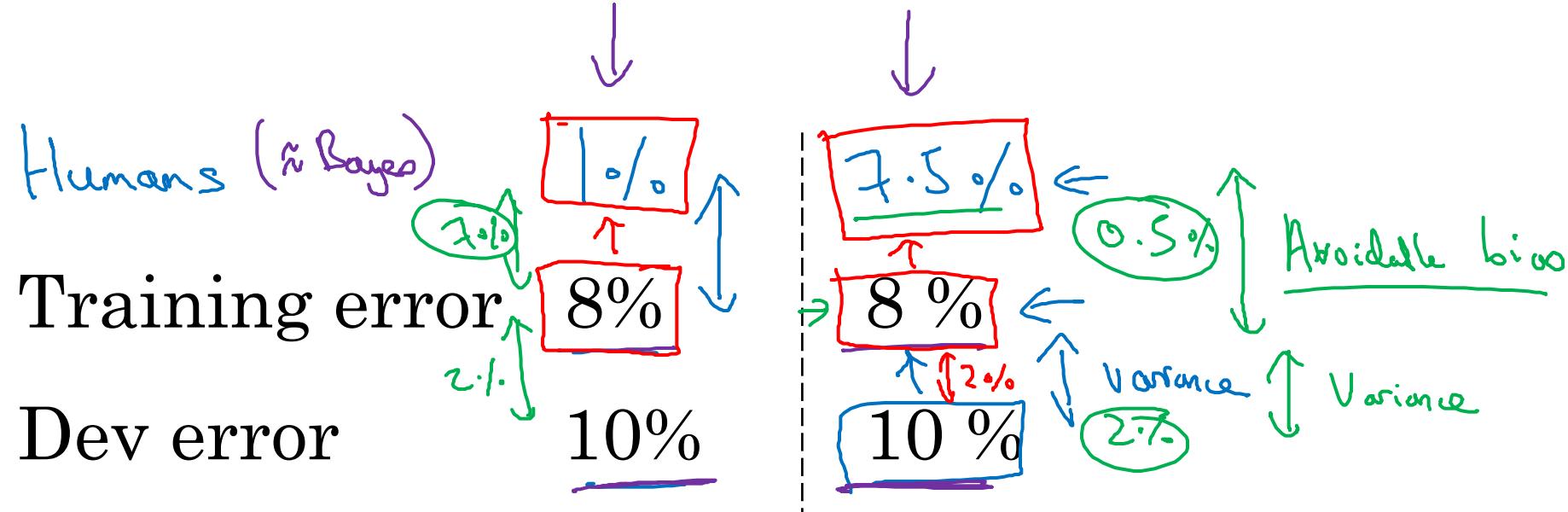
high bias

high bias
high variance

low bias
low variance



Cat classification example



Focus on

bias

Human-level error as a proxy for Bayes error.

Focus on

Variance



deeplearning.ai

Comparing to human-level performance

Understanding
human-level
performance

Human-level error as a proxy for Bayes error

Medical image classification example:

Suppose:

- (a) Typical human 3 % error
- (b) Typical doctor 1 % error
- (c) Experienced doctor 0.7 % error
- (d) Team of experienced doctors .. 0.5 % error



What is “human-level” error?

$$\text{Baye error} \leq \underline{0.5\%}$$

Error analysis example

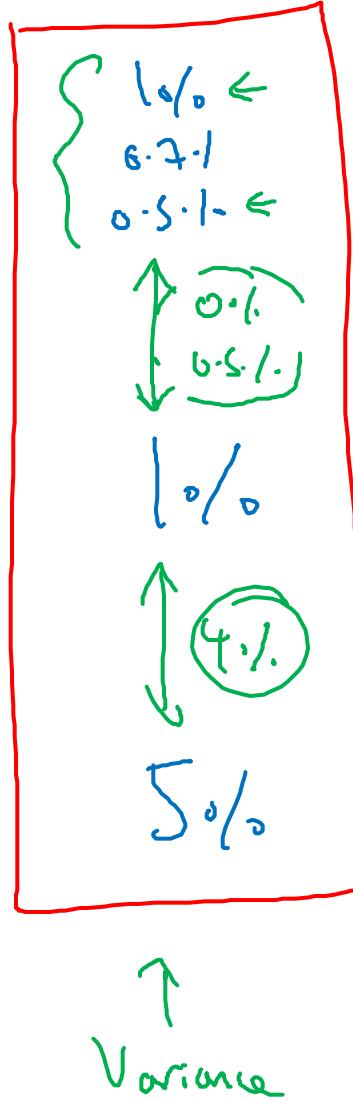
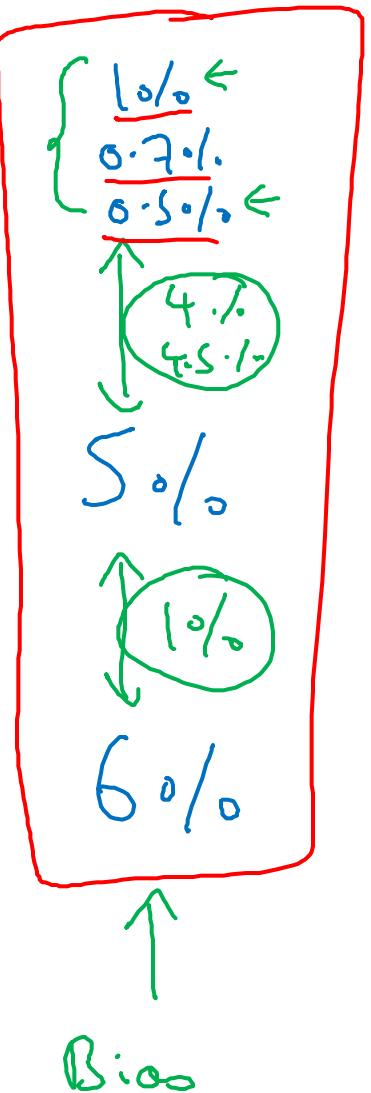
Human (proxy for Bayes error)



Training error

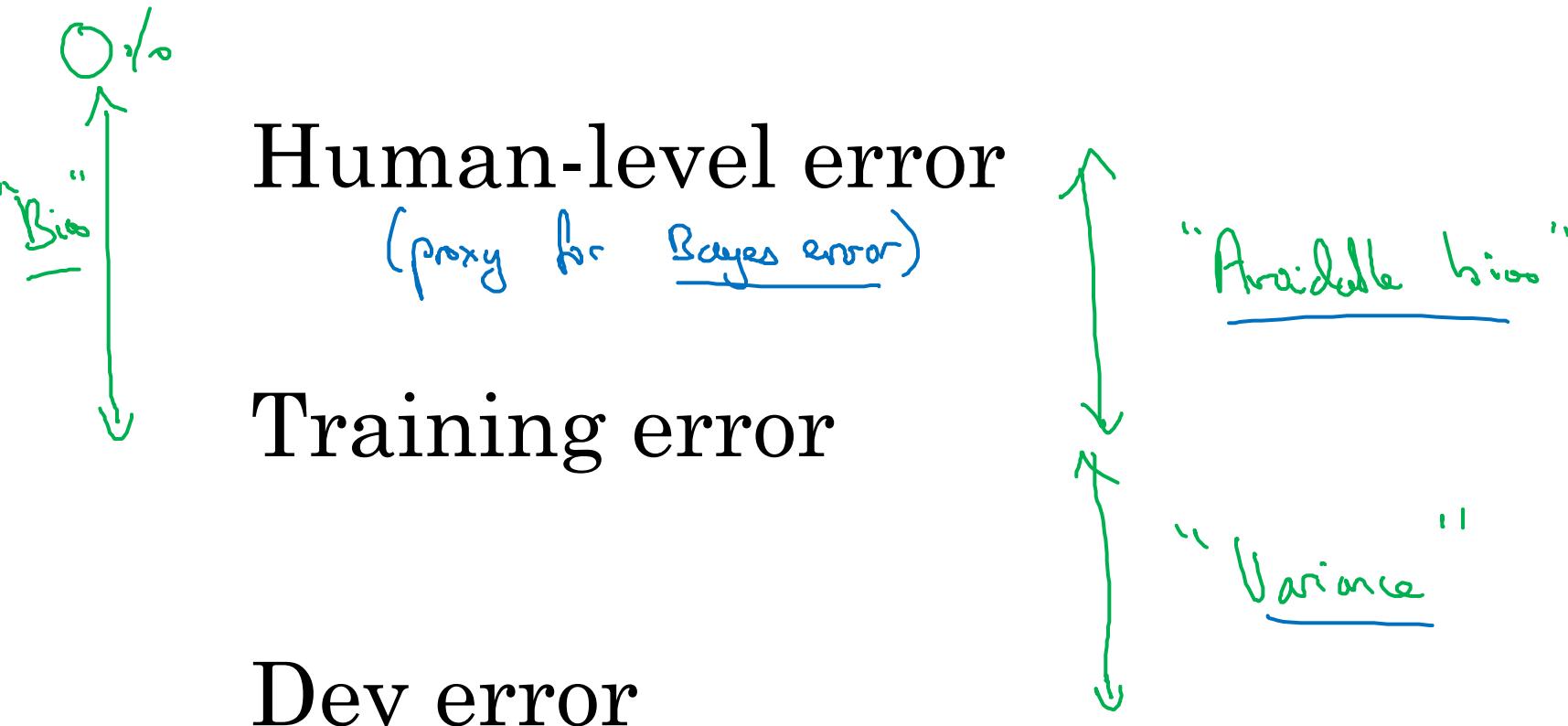


Dev error



$$\begin{aligned} &\rightarrow \frac{0.7\%}{0.5\%} \quad 1\% \\ &\rightarrow \qquad \qquad \qquad \downarrow \\ &0.2\% \quad 0.0\% \\ &\rightarrow \boxed{0.7\%} \quad \downarrow \\ &\qquad \qquad \qquad \downarrow \\ &0.1\% \quad \downarrow \\ &\rightarrow 0.8\% \end{aligned}$$

Summary of bias/variance with human-level performance





deeplearning.ai

Comparing to human-level performance

Surpassing human-level performance

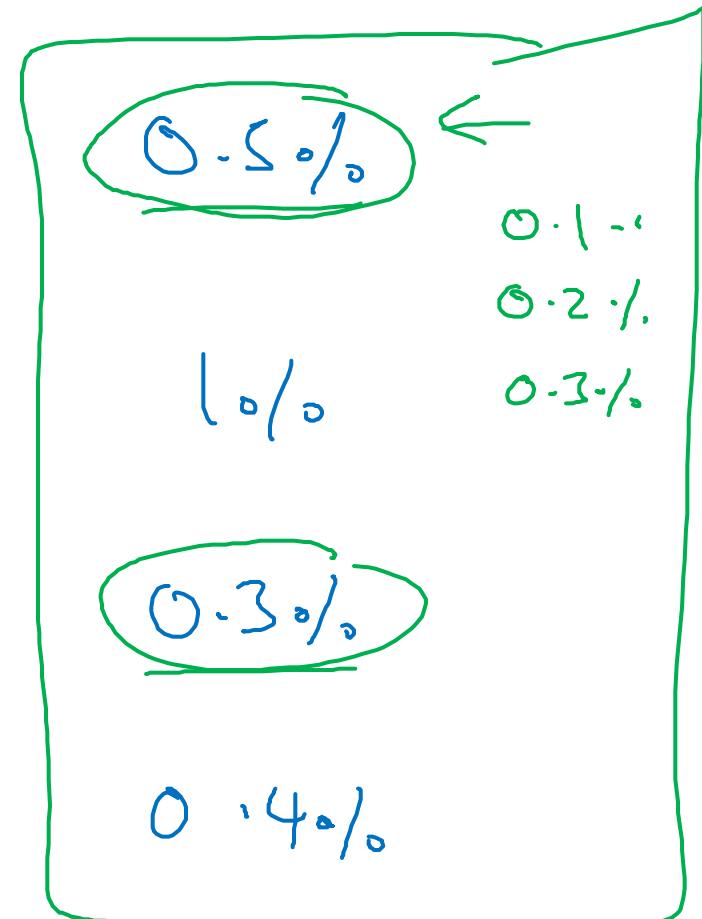
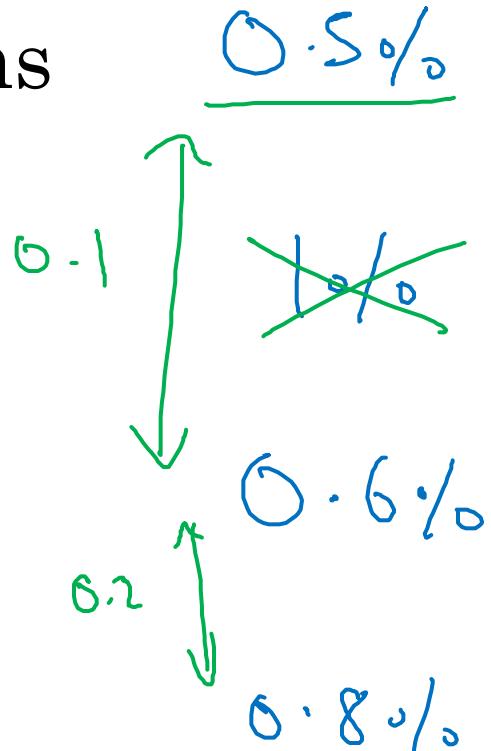
Surpassing human-level performance

Team of humans

One human

Training error

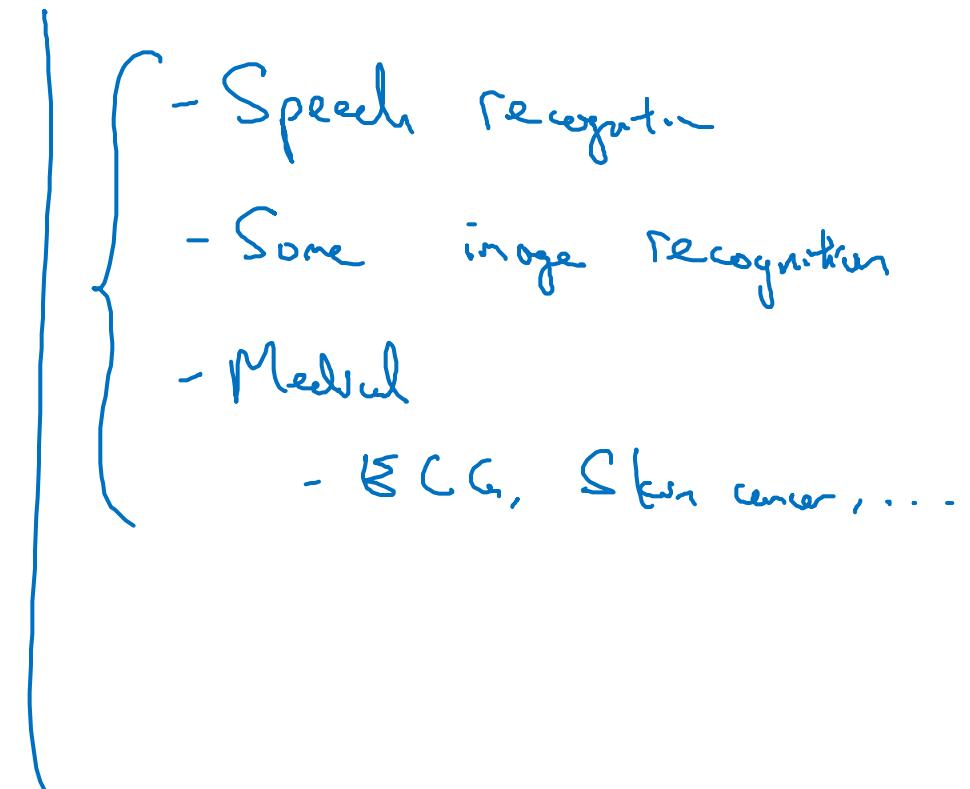
Dev error



What is avoidable bias?

Problems where ML significantly surpasses human-level performance

- - Online advertising
- - Product recommendations
- - Logistics (predicting transit time)
- - Loan approvals



Structural data

Not natural perception

Lots of data



deeplearning.ai

Comparing to human-level performance

Improving your model performance

The two fundamental assumptions of supervised learning

1. You can fit the training set pretty well.



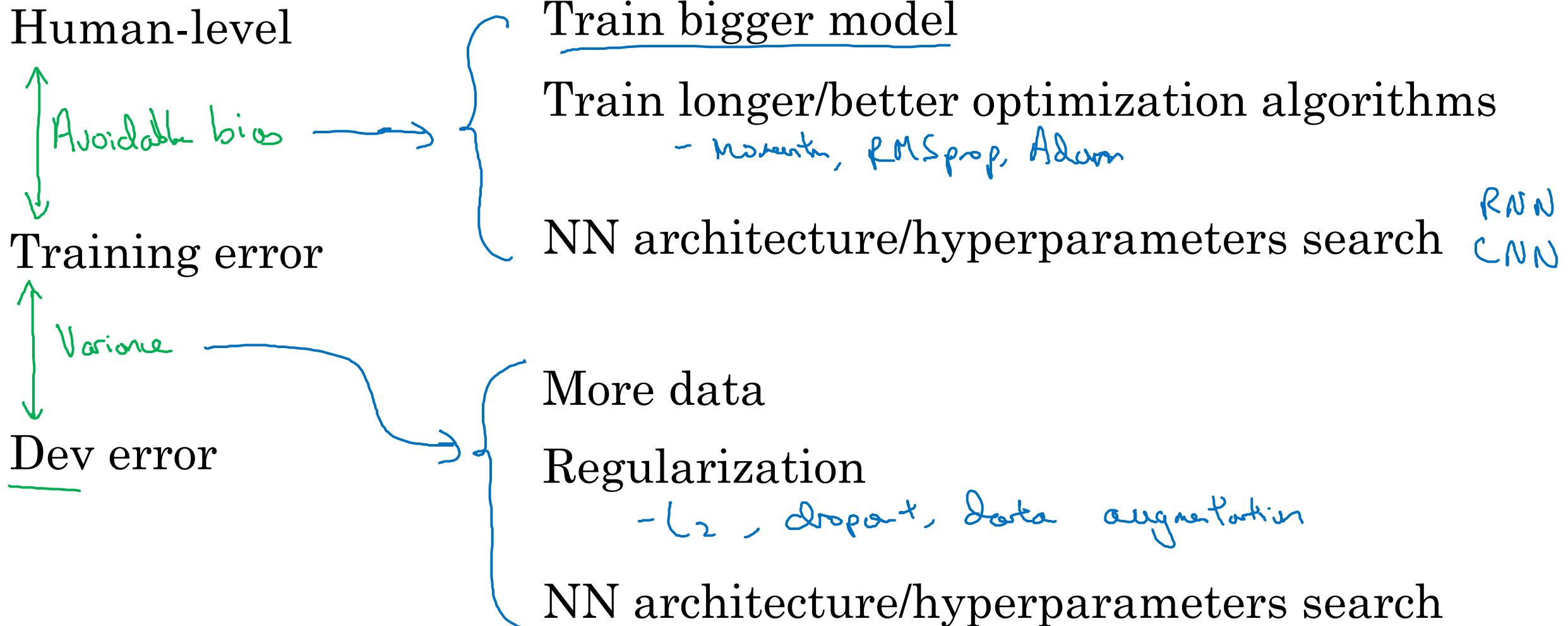
\sim Avoidable bias

2. The training set performance generalizes pretty well to the dev/test set.



\sim Variance

Reducing (avoidable) bias and variance



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

Error Analysis

Carrying out error
analysis

Look at dev examples to evaluate ideas



90% accuracy
→ 10% error

Should you try to make your cat classifier do better on dogs? ←

Error analysis: → 5-10 min

- { • Get ~100 mislabeled dev set examples.
- Count up how many are dogs.

→ 5%
5/100

10%
↓
95%

"Ceiling"
→ 50%
50/100

10%
↓
5%

Evaluate multiple ideas in parallel

Ideas for cat detection:

- Fix pictures of dogs being recognized as cats ←
- Fix great cats (lions, panthers, etc..) being misrecognized ←
- Improve performance on blurry images ← ↴

Image	Dog	Great Cats	Blurry	Instagram	Comments
1	✓			✓	Pitbull
2			✓	✓	
3		✓	✓		Rainy day at zoo
:	:	:	:		
% of total	<u>8%</u>	<u>43%</u>	<u>61%</u>	<u>12%</u>	



deeplearning.ai

Error Analysis

Cleaning up
Incorrectly labeled
data

Incorrectly labeled examples

x



y

1

0

1

1

0

Training set.

DL algorithms are quite robust to random errors in the training set.

Systematic errors

Error analysis



Image	Dog	Great Cat	Blurry	Incorrectly labeled	Comments
...					
98				✓	Labeler missed cat in background
99		✓			
100				✓	Drawing of a cat; Not a real cat.
% of total	<u>8%</u>	<u>43%</u>	<u>61%</u>	<u>6%</u>	

Overall dev set error 100%

Errors due incorrect labels 0.6% ←

Errors due to other causes 9.4% ←

2% ←

0.6% ←

1.4%

2.1%

1.9%

Goal of dev set is to help you select between two classifiers A & B.

Correcting incorrect dev/test set examples

- Apply same process to your dev and test sets to make sure they continue to come from the same distribution
- Consider examining examples your algorithm got right as well as ones it got wrong.
(8.6%) *(20%)*
- Train and dev/test data may now come from slightly different distributions.

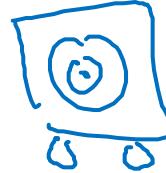


deeplearning.ai

Error Analysis

Build your first system
quickly, then iterate

Speech recognition example



- • Noisy background
 - • Café noise
 - • Car noise
 - • Accent
 - • Far from
 - • Young
 - • Stutter
 - • ...
- Guideline:**
Build your first system quickly, then iterate
- • Set up dev/test set and metric
 - Build initial system quickly
 - Use Bias/Variance analysis & Error analysis to prioritize next steps.



deeplearning.ai

Mismatched training
and dev/test data

Training and testing
on different
distributions

Cat app example

Data from webpages



care about this
Data from mobile app

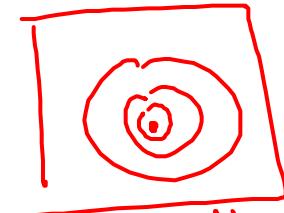


$\rightarrow \approx 200,000$

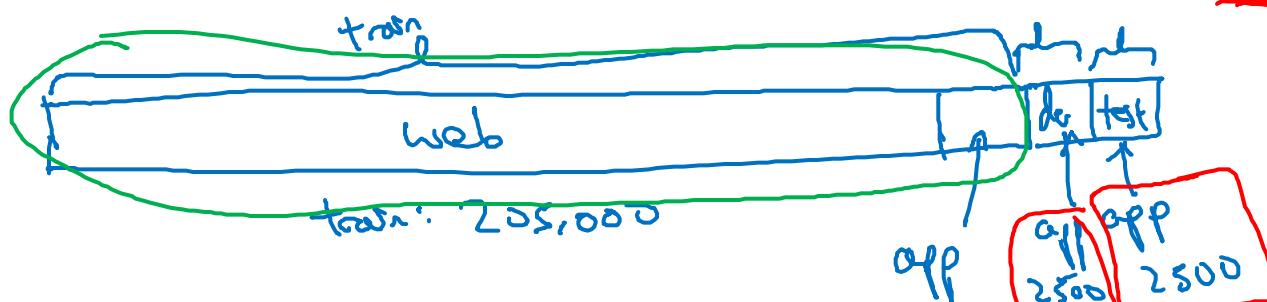
$\approx 210,000$
shuffle

$\rightarrow \approx 10,000$

X Option 1:



Option 2:



$\frac{200K}{210K}$

2381 - web
119 - mobile app



Speech recognition example

Speech activated rearview mirror



Training

Purchased data $\downarrow \downarrow$
 x, y

Smart speaker control

Voice keyboard

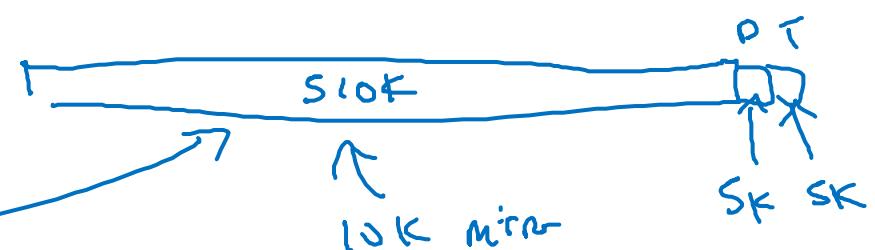
...
500,000

utterances

Dev/test

Speech activated
rearview mirror

$\rightarrow 20,000$





deeplearning.ai

Mismatched training
and dev/test data

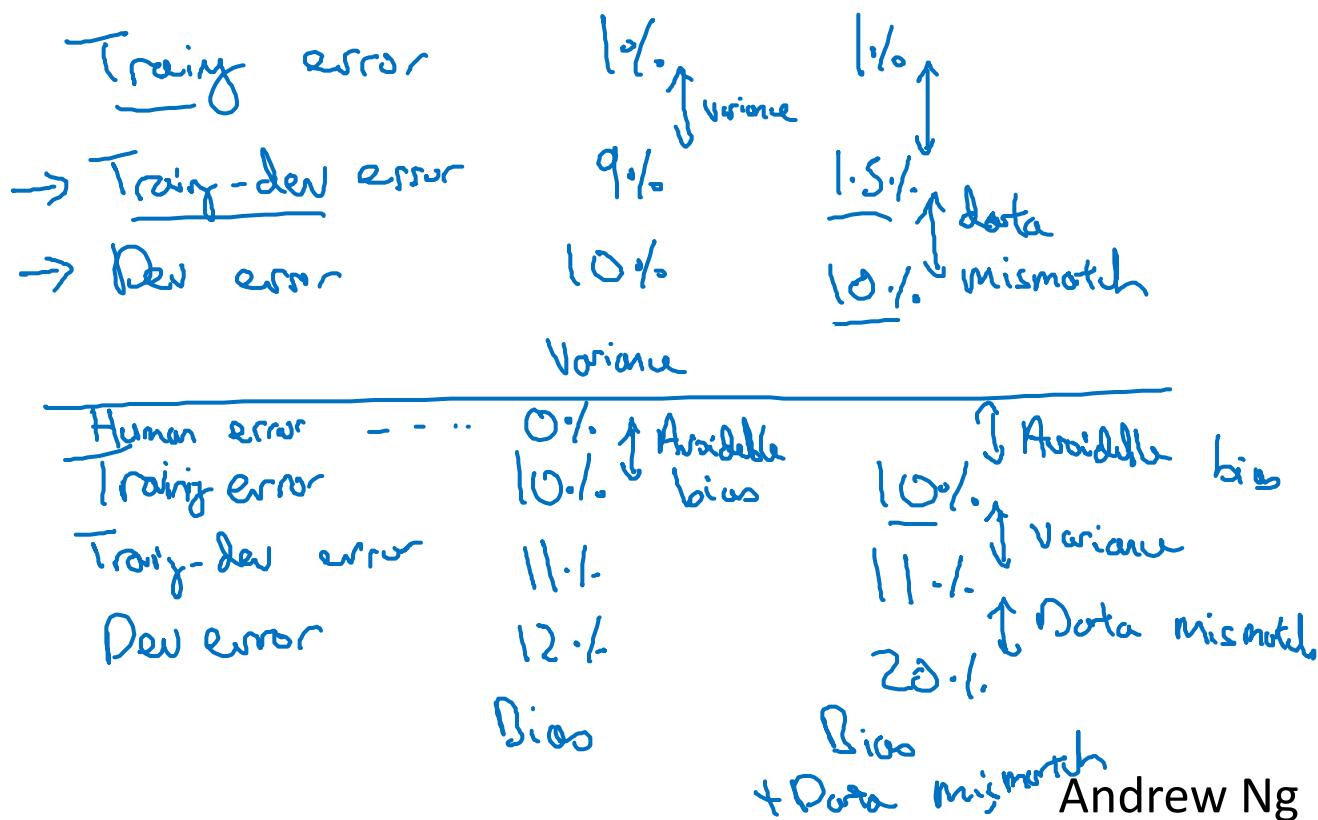
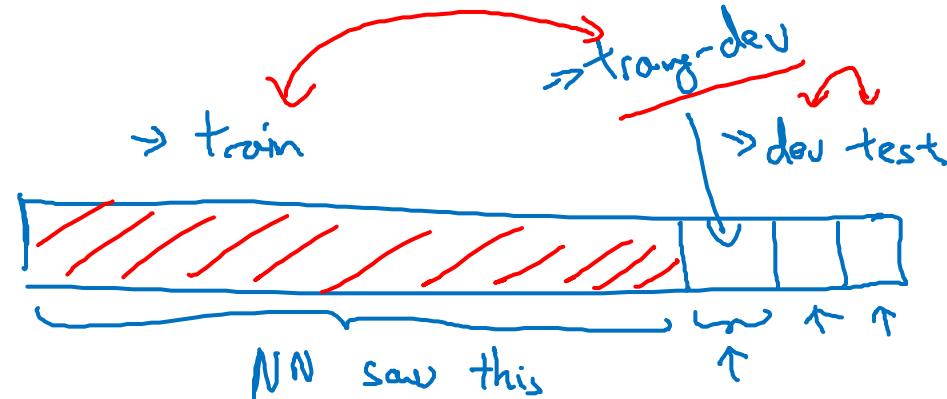
Bias and Variance with
mismatched data
distributions

Cat classifier example

Assume humans get $\approx 0\%$ error.

Training error 1% \downarrow 9%
Dev error 10% \uparrow

Training-dev set: Same distribution as training set, but not used for training



Bias/variance on mismatched training and dev/test sets

Human level

Training set error

Training - dev set error

→ Dev error

→ Test error

4% ↑ avoidable bias

7% ↑ variance

10% ↓ data mismatch

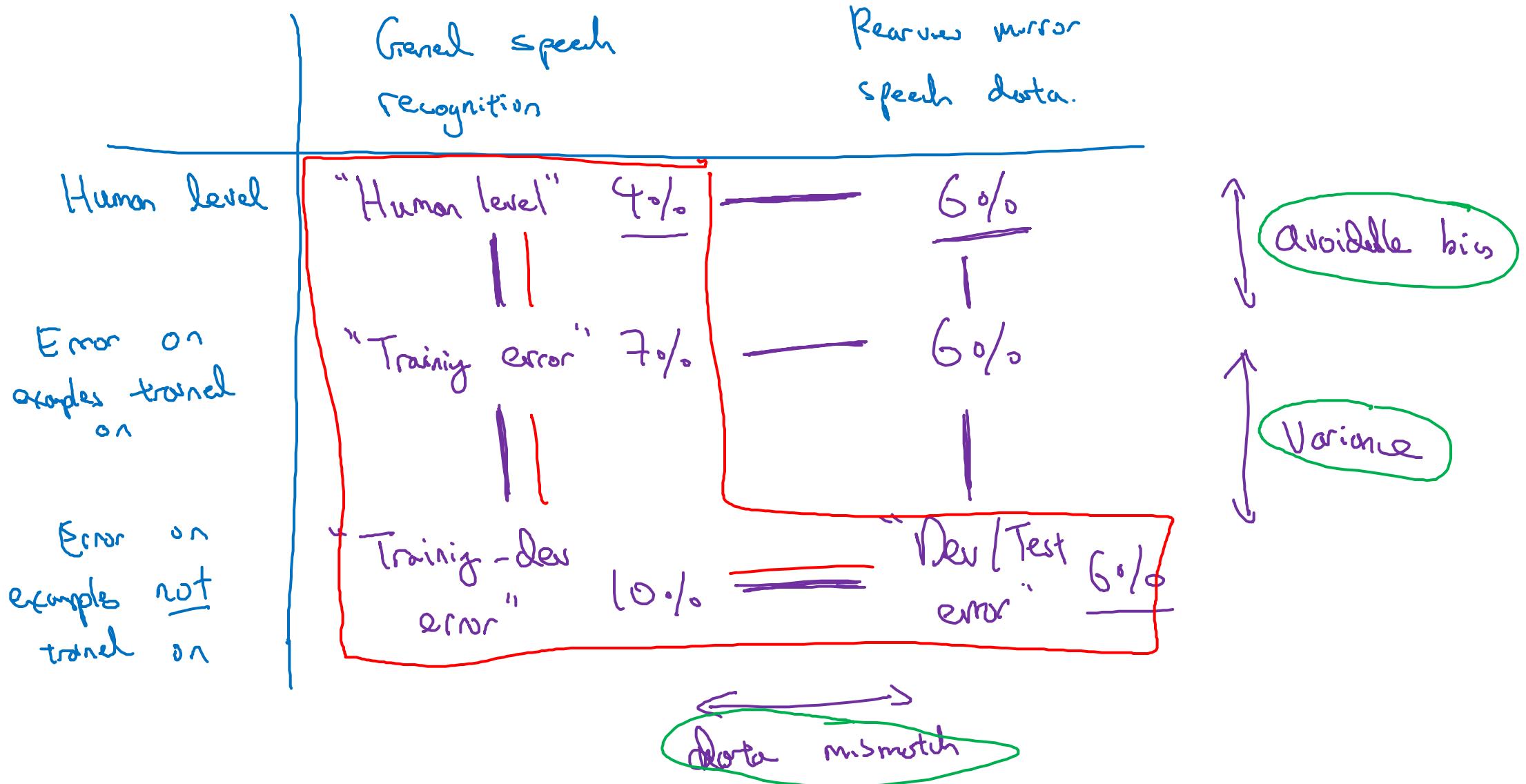
12% ↓ degree of overfitting
to dev set.

4%

7% }
10% }

6% }
6% }

More general formulation





deeplearning.ai

Mismatched training
and dev/test data

Addressing data
mismatch

Addressing data mismatch

- • Carry out manual error analysis to try to understand difference between training and dev/test sets

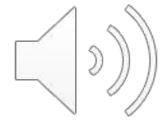
E.g. noisy - car noise

street numbers

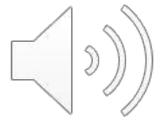
- • Make training data more similar; or collect more data similar to dev/test sets

E.g. Simulate noisy in-car data

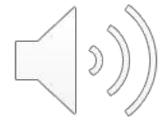
Artificial data synthesis



+

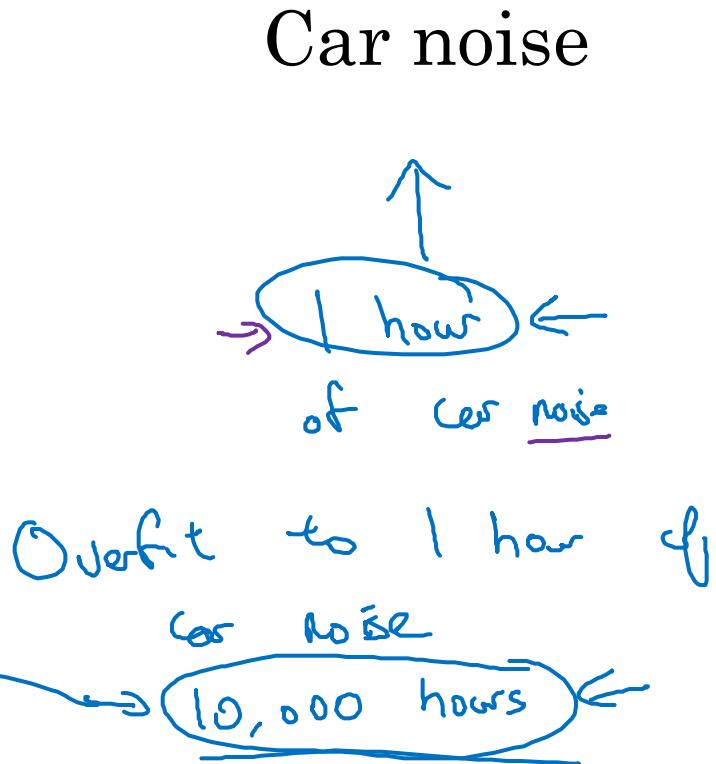


=

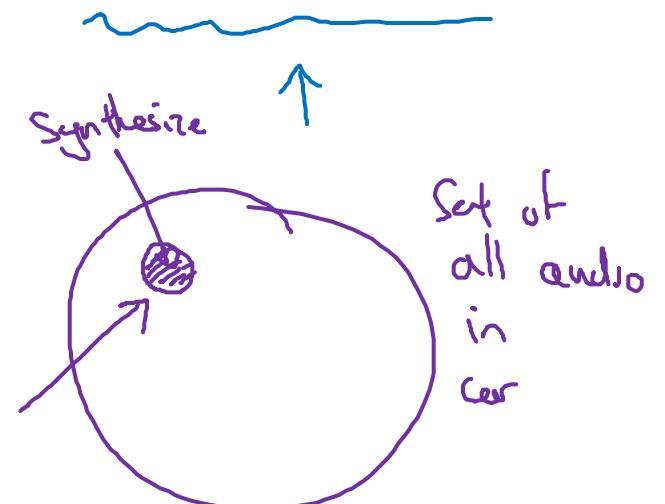


“The quick brown  fox jumps over the lazy dog.”

10,000 hours



Synthesized in-car audio

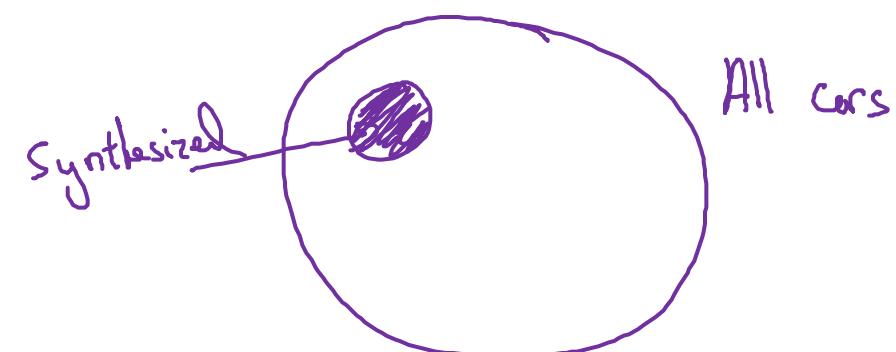


Artificial data synthesis

Car recognition:



$N \approx 20$ cars



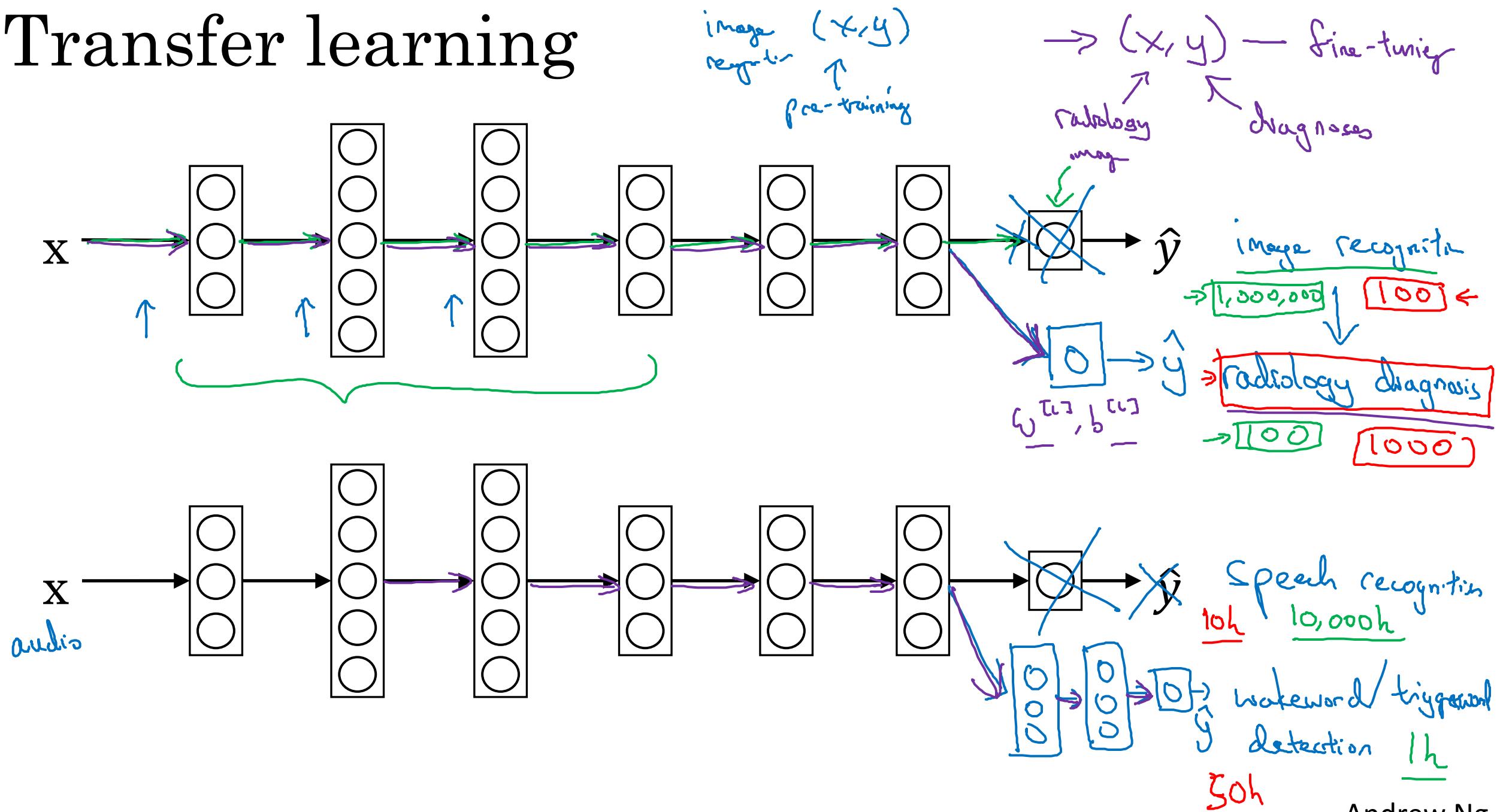


deeplearning.ai

Learning from
multiple tasks

Transfer learning

Transfer learning



When transfer learning makes sense

Transfer from A \rightarrow B

- Task A and B have the same input x .
- You have a lot more data for Task A than Task B.

- Low level features from A could be helpful for learning B.



deeplearning.ai

Learning from
multiple tasks

Multi-task
learning

Simplified autonomous driving example



$x^{(i)}$

Pedestrians

Cars

Stop signs

Traffic lights

⋮

$y^{(i)}$

0

1

1

0

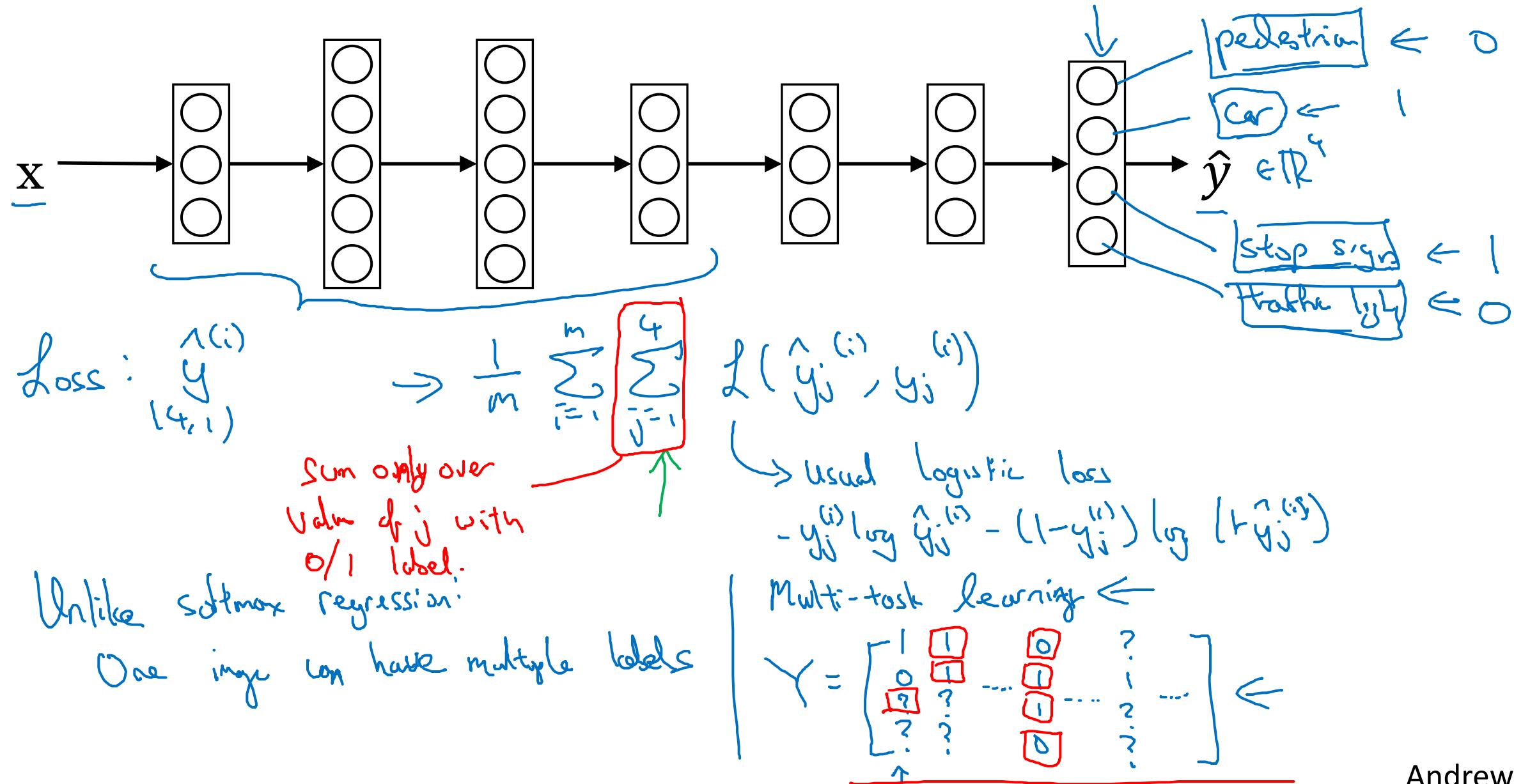
⋮

(4, 1)

$$Y = \begin{bmatrix} y^{(1)} & y^{(2)} & y^{(3)}, \dots, y^{(m)} \end{bmatrix}$$

(4, m)

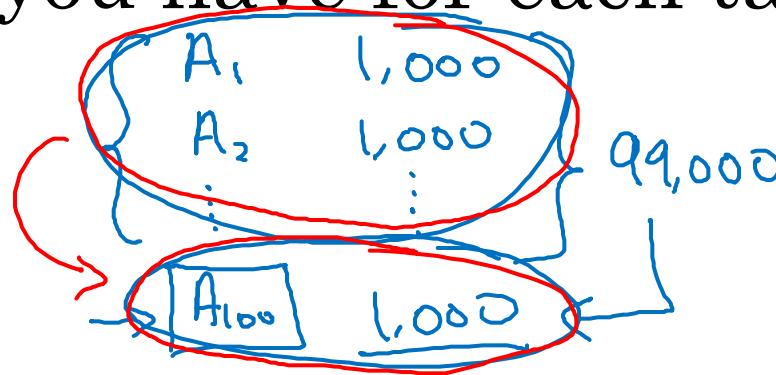
Neural network architecture



When multi-task learning makes sense

- Training on a set of tasks that could benefit from having shared lower-level features.
- Usually: Amount of data you have for each task is quite similar.

$$\begin{array}{ll} A & \underline{1,000,000} \\ \downarrow & \downarrow \\ B & \underline{1,000} \end{array}$$



- Can train a big enough neural network to do well on all the tasks.



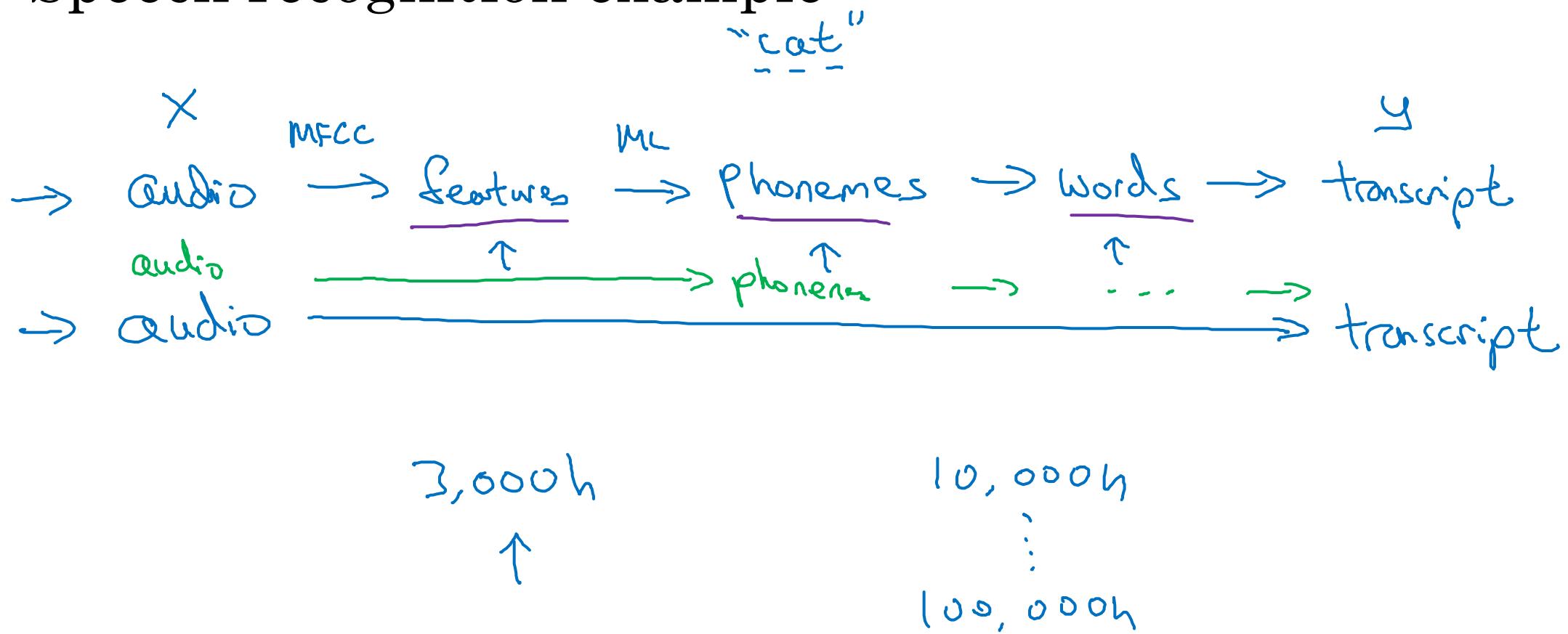
deeplearning.ai

End-to-end deep
learning

What is
end-to-end
deep learning

What is end-to-end learning?

Speech recognition example

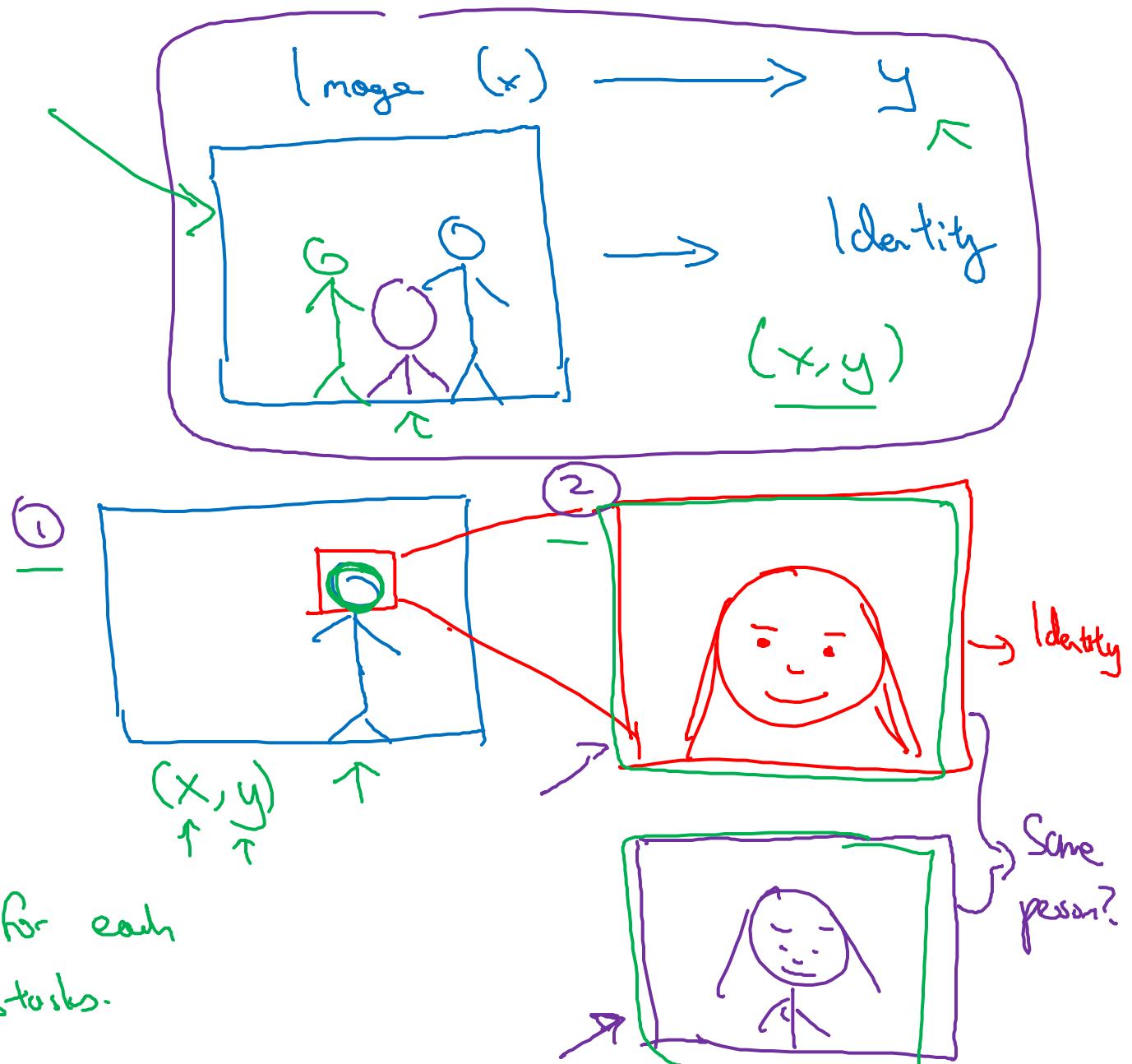


Face recognition



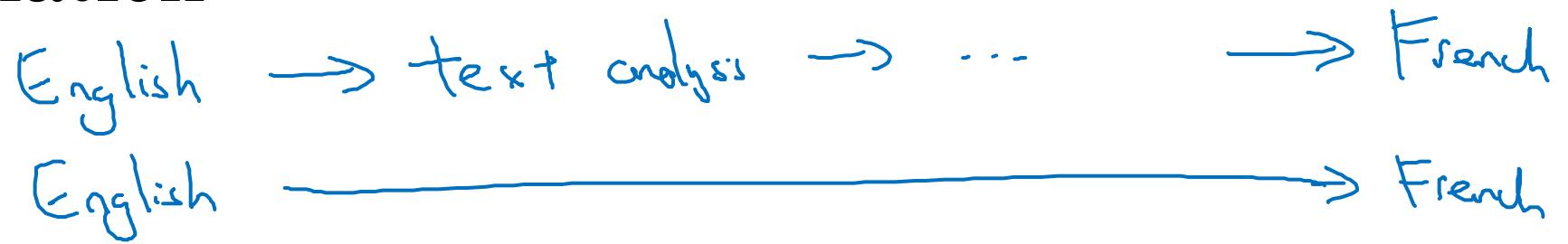
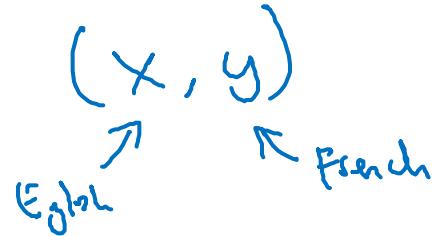
[Image courtesy of Baidu]

Have data for each
of 2 subtasks.

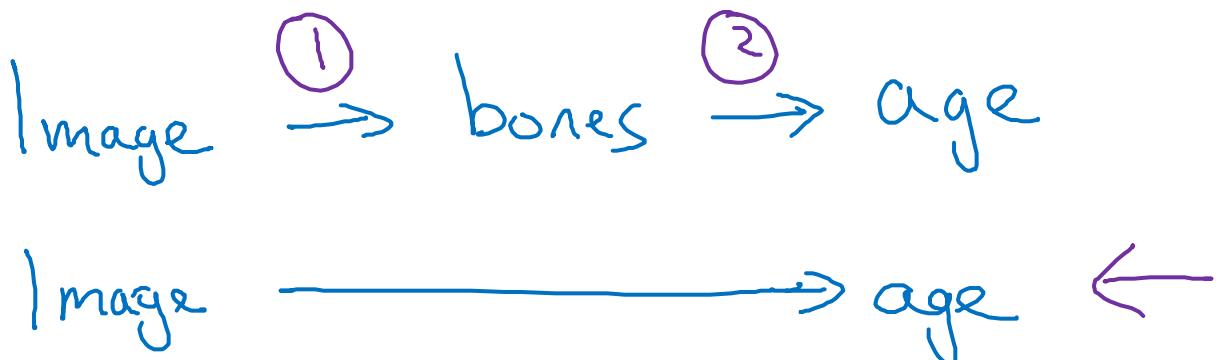


More examples

Machine translation



Estimating child's age:





deeplearning.ai

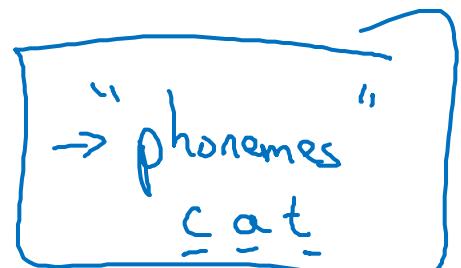
End-to-end deep
learning

Whether to use
end-to-end learning

Pros and cons of end-to-end deep learning

Pros:

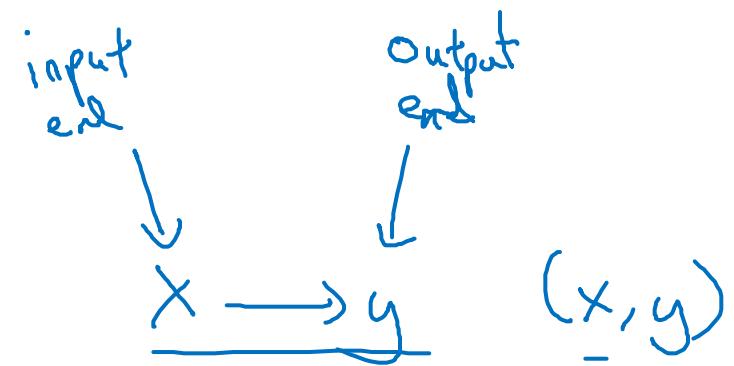
- Let the data speak $x \rightarrow y$
- Less hand-designing of components needed



Cons:

- May need large amount of data
- Excludes potentially useful hand-designed components

$$x - - - - \rightarrow y$$

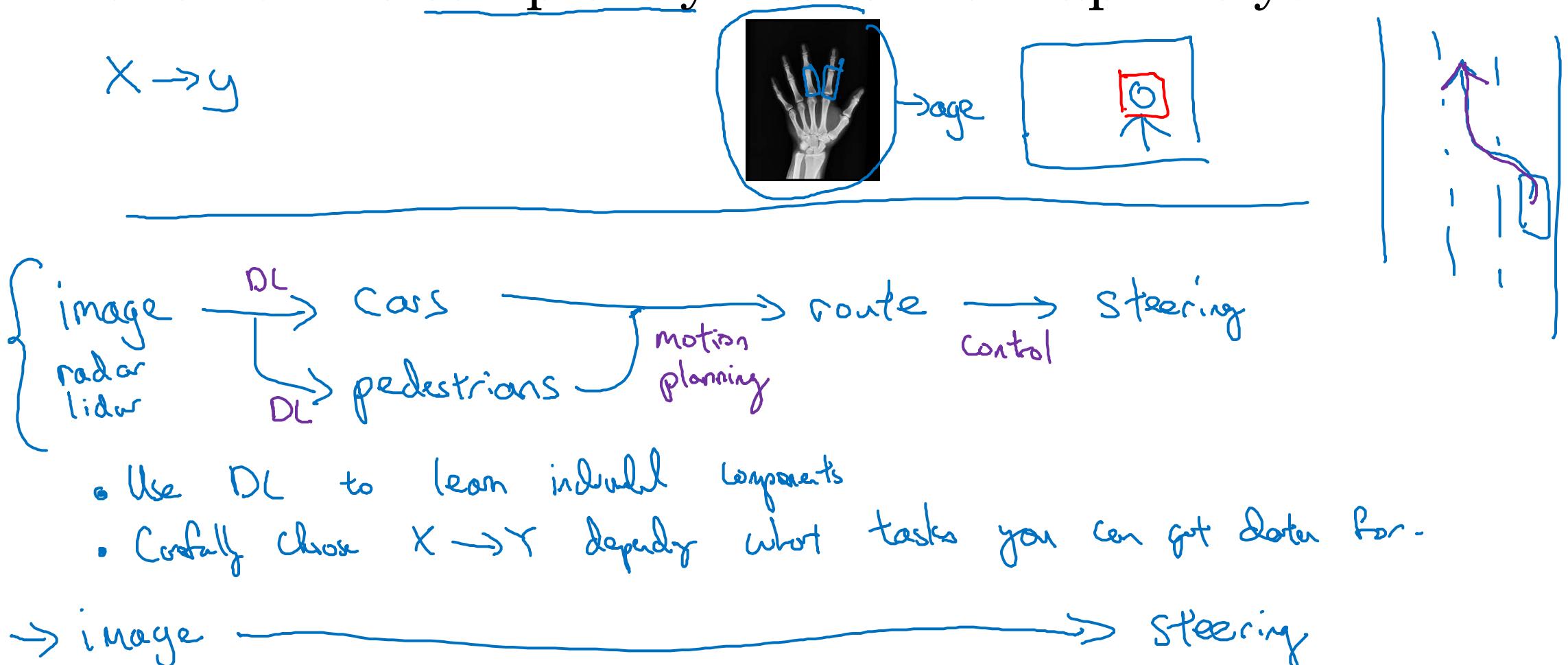


Data

Hand-design.

Applying end-to-end deep learning

Key question: Do you have sufficient data to learn a function of the complexity needed to map x to y ?



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

Convolutional Neural Networks

Computer vision

Computer Vision Problems

Image Classification



→ Cat? (0/1)

Neural Style Transfer



Object detection



Deep Learning on large images



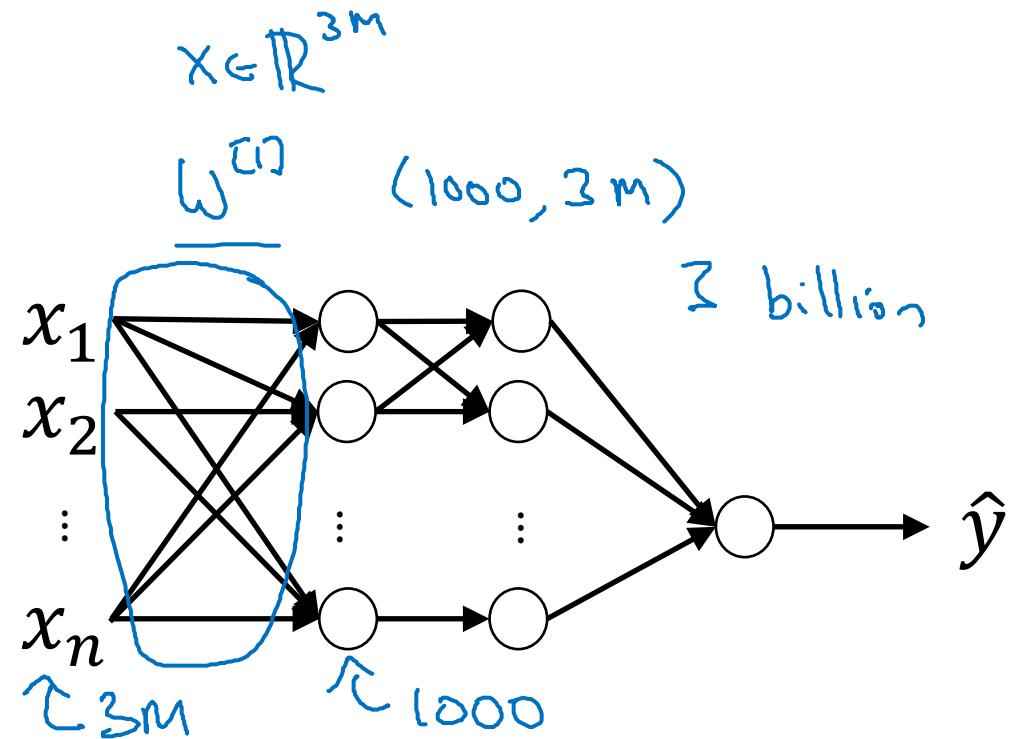
$64 \times 64 \times 3$

→ Cat? (0/1)

12288



$1000 \times 1000 \times 3$
= 3 million



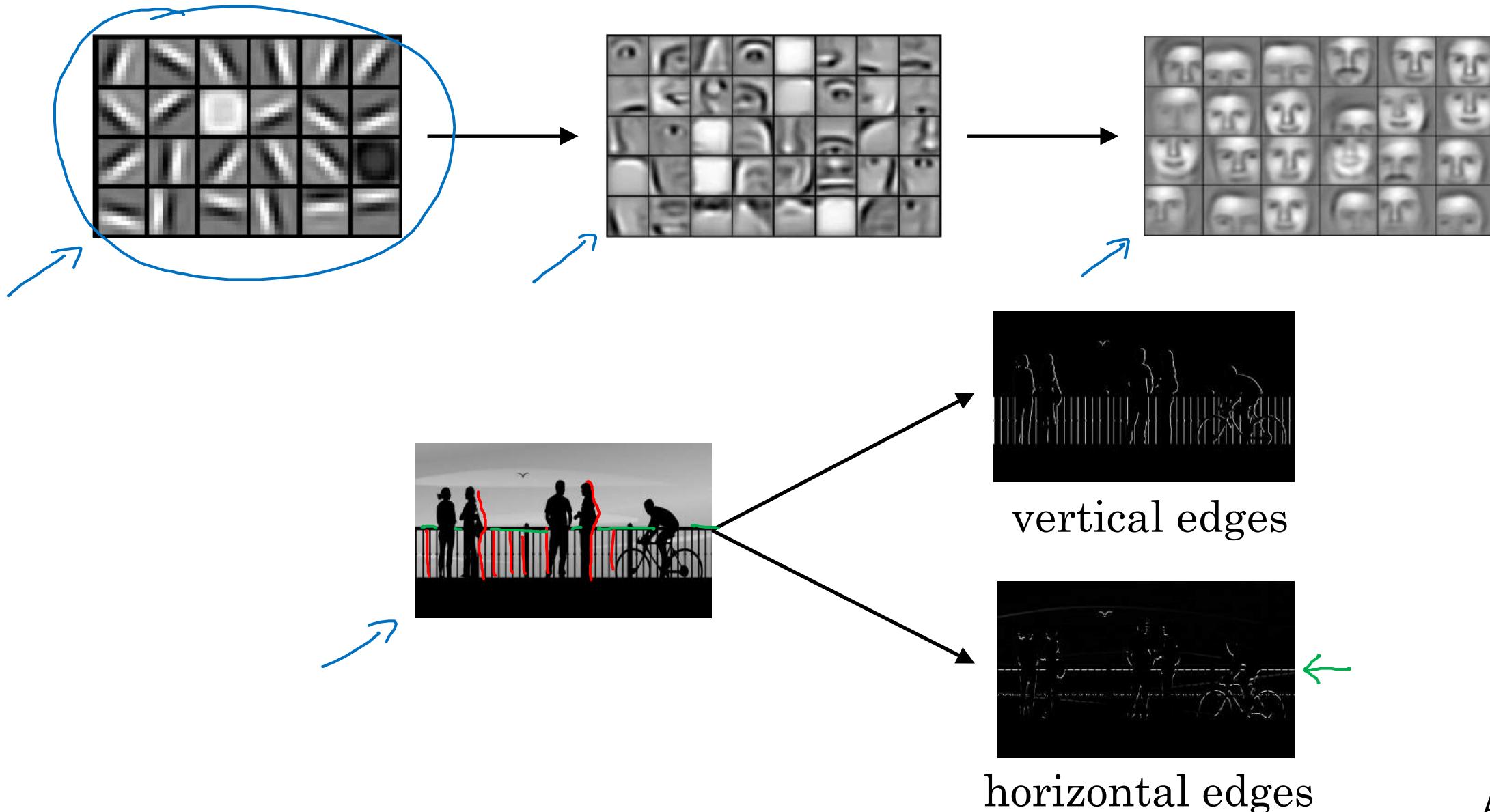


deeplearning.ai

Convolutional Neural Networks

Edge detection example

Computer Vision Problem

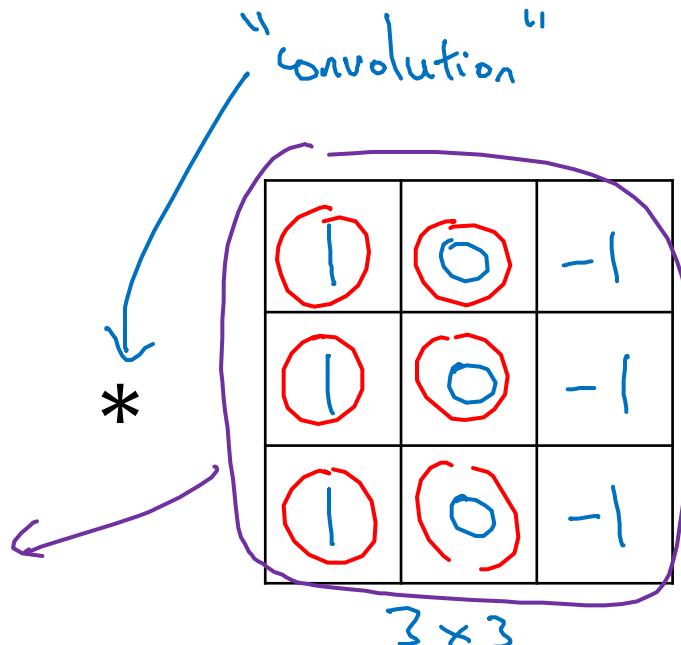


Vertical edge detection

$$3 \times 1 + 1 \times 1 + 2 \times 1 + 0 \times 0 + 5 \times 0 + 7 \times 0 + 1 \times 1 + 8 \times -1 + 2 \times -1 = -5$$

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

6×6



=

-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

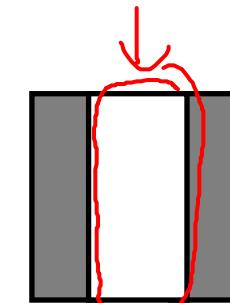
Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



$$\begin{matrix} * & \begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix} \\ & 3 \times 3 \end{matrix}$$

$$= \begin{matrix} \begin{matrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{matrix} \\ 4 \times 4 \end{matrix}$$



$$* \begin{matrix} \uparrow & \uparrow & \uparrow \end{matrix}$$

Andrew Ng



deeplearning.ai

Convolutional Neural Networks

More edge
detection

Vertical edge detection examples

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0



0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10



*

1	0	-1
1	0	-1
1	0	-1



=

0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0



*

1	0	-1
1	0	-1
1	0	-1



=

0	-30	-30	0
0	-30	-30	0
0	-30	-30	0
0	-30	-30	0



Vertical and Horizontal Edge Detection

1	0	-1
1	0	-1
1	0	-1

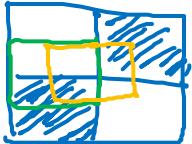
Vertical

1	1	1
0	0	0
-1	-1	-1

Horizontal

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
0	0	0	10	10	10
0	0	0	10	10	10
0	0	0	10	10	10

6×6



*

1	1	1
0	0	0
-1	-1	-1

=

0	0	0	0
30	10	-10	-30
30	10	-10	-30
0	0	0	0

Learning to detect edges

1	0	-1
1	0	-1
1	0	-1

→

1	0	-1
2	0	-2
1	0	-1

3	0	-3
10	0	-10
3	0	-3

↑

3	0	1	2	7	4
1	5	8	9	3	1
2	7	2	5	1	3
0	1	3	1	7	8
4	2	1	6	2	8
2	4	5	2	3	9

Sobel filter

convolution

×

w_1	w_2	w_3
w_4	w_5	w_6
w_7	w_8	w_9

3×3

=

45°
 70°
 73°

↑

Scharr filter



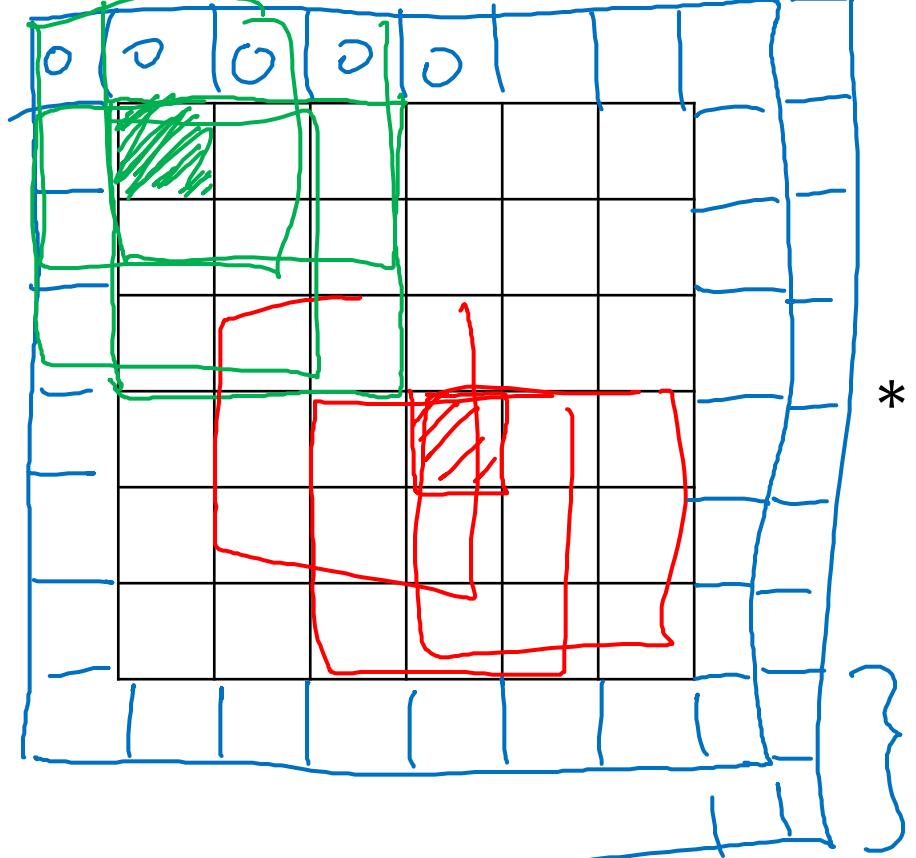
deeplearning.ai

Convolutional Neural Networks

Padding

Padding

- shrinky output
- throw away info from edge



$$\frac{6 \times 6}{n \times n} \rightarrow 8 \times 8$$

$$n-f+1 \times n-f+1$$

$$6-3+1 = 4$$

$$P = \text{padding} = 1$$

$$* \quad \begin{array}{|c|c|c|} \hline & & \\ \hline & & \\ \hline & & \\ \hline \end{array}$$

3×3
 $f \times f$

=

Diagram showing the result of the convolution operation. The 3x3 kernel has been applied to the 4x4 receptive field of the central output unit. The result is a 4x4 output matrix, which is then reduced to a 6x6 output matrix via max pooling.

$$\xrightarrow{\quad} \frac{4 \times 4}{\text{---}}$$

$$n+2p-f+1 \times n+2p-f+1$$

$$6+2-3+1 \times \underline{\quad} = 6 \times 6$$

Valid and Same convolutions

→ n → padding

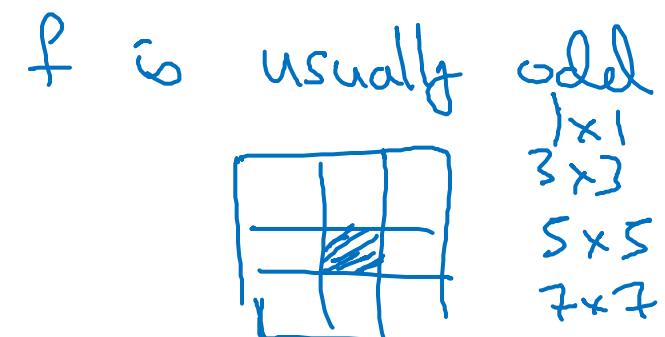
“Valid”: $n \times n \quad * \quad f \times f \quad \rightarrow \frac{n-f+1}{f} \times \frac{n-f+1}{f}$

$$\begin{array}{ccc} n \times n & * & f \times f \\ 6 \times 6 & * & 3 \times 3 \end{array} \rightarrow 4 \times 4$$

“Same”: Pad so that output size is the same as the input size.

$$n + 2p - f + 1 = n \Rightarrow p = \frac{f-1}{2}$$

$$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad \left| \begin{array}{c} S \times S \\ f=5 \end{array} \right. \quad p=2$$



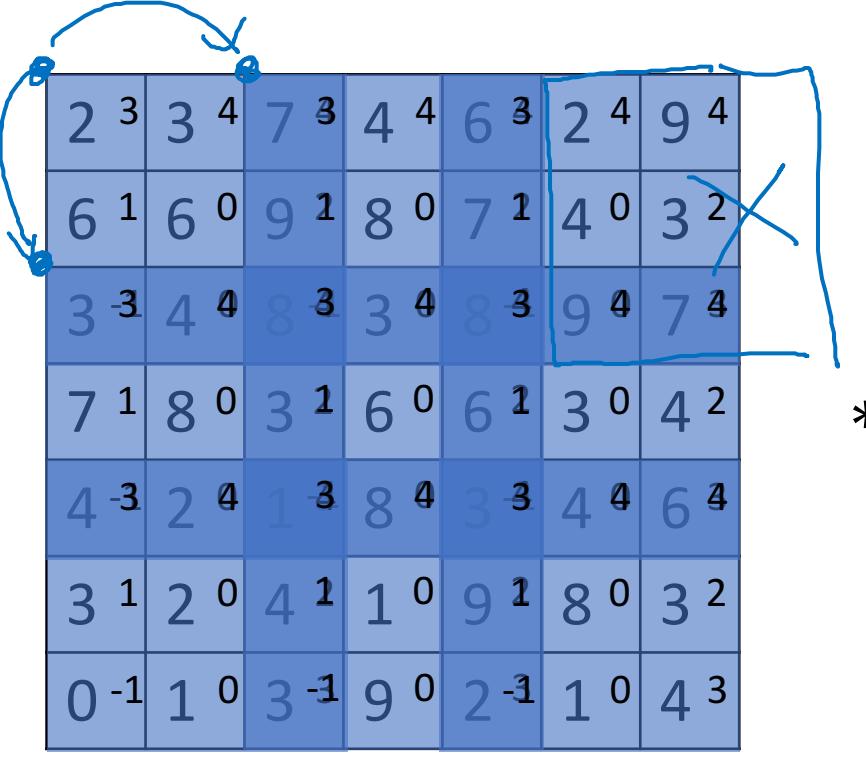


deeplearning.ai

Convolutional Neural Networks

Strided convolutions

Strided convolution



*

$$\begin{array}{|c|c|c|} \hline 3 & 4 & 4 \\ \hline 1 & 0 & 2 \\ \hline -1 & 0 & 3 \\ \hline \end{array}$$

3×3

stride = 2

=

$$\begin{array}{|c|c|c|} \hline 91 & 100 & 83 \\ \hline 69 & 91 & 127 \\ \hline 44 & 72 & 74 \\ \hline \end{array}$$

3×3

$\lfloor \frac{z}{2} \rfloor = \text{floor}(z)$

$n \times n$ * $f \times f$
padding p stride s
 $s=2$

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$

$$\frac{7+0-3}{2} + 1 = \frac{4}{2} + 1 = 3$$

Summary of convolutions

$n \times n$ image $f \times f$ filter

padding p stride s

Output size:

$$\left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor \quad \times \quad \left\lfloor \frac{n+2p-f}{s} + 1 \right\rfloor$$


Technical note on cross-correlation vs. convolution

Convolution in math textbook:

2	3	7	5	4	6	2
6	6	9	4	8	7	4
3	4	8	3	3	8	9
7	8	3	6	6	6	3
4	2	1	8	3	3	4
3	2	4	1	9	8	

$$\begin{matrix} & * & \begin{matrix} 3 & 4 & 5 \\ 1 & 0 & 2 \\ -1 & 9 & 7 \end{matrix} \\ \begin{matrix} 7 & 2 & 5 \\ 9 & 0 & 4 \\ -1 & 1 & 3 \end{matrix} & \end{matrix}$$

$$= \begin{matrix} & \begin{matrix} 1 & 0 & 2 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{matrix} \end{matrix}$$

$$(A * B) * C = A * (B * C)$$

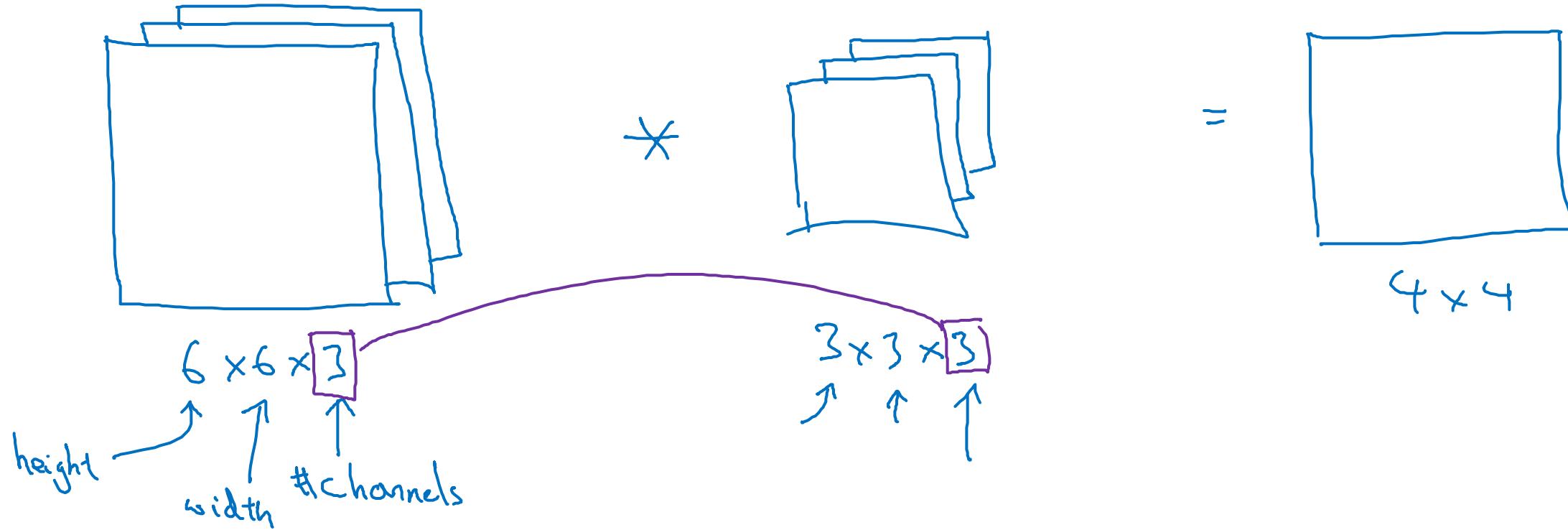


deeplearning.ai

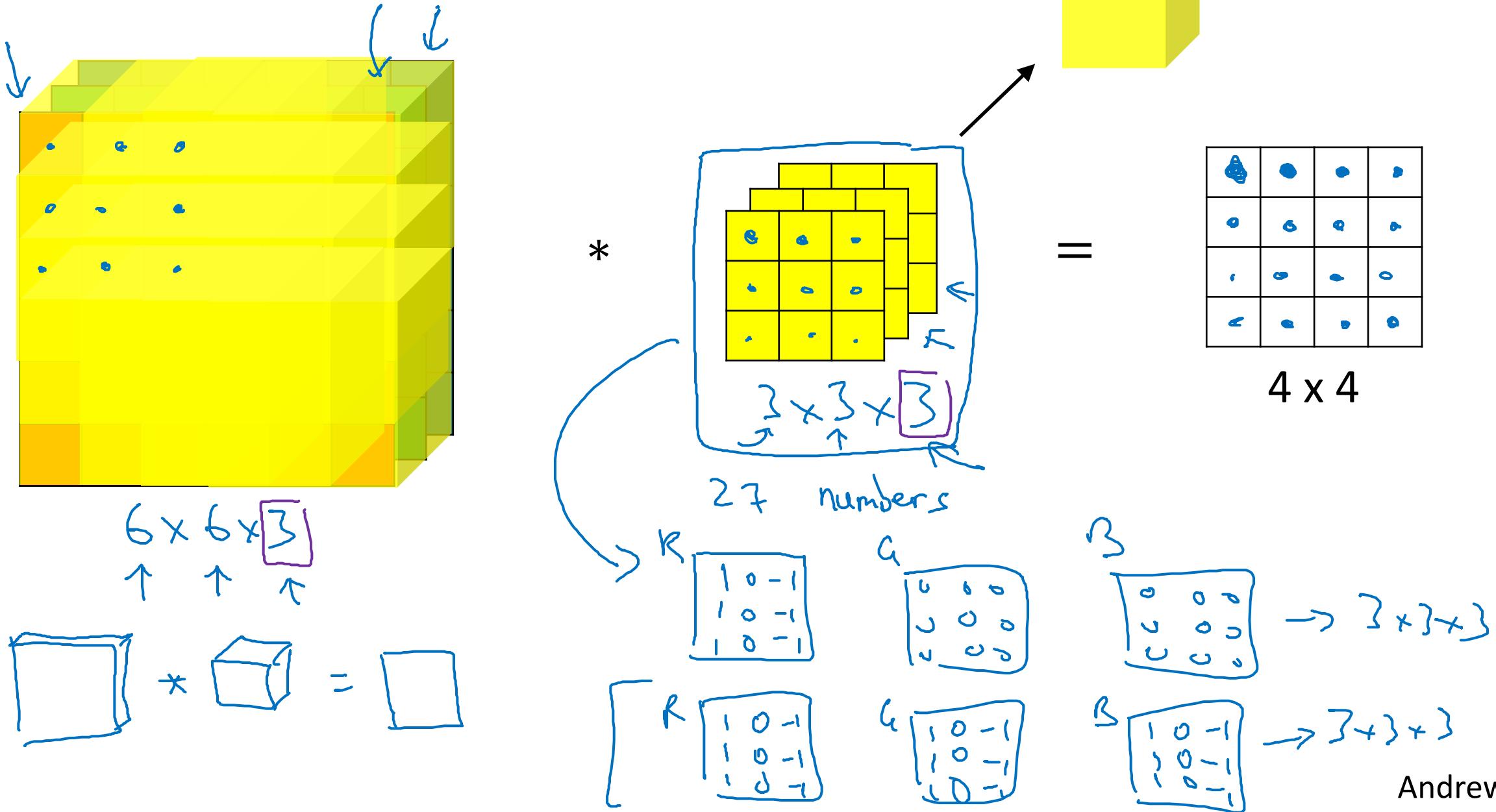
Convolutional Neural Networks

Convolutions over volumes

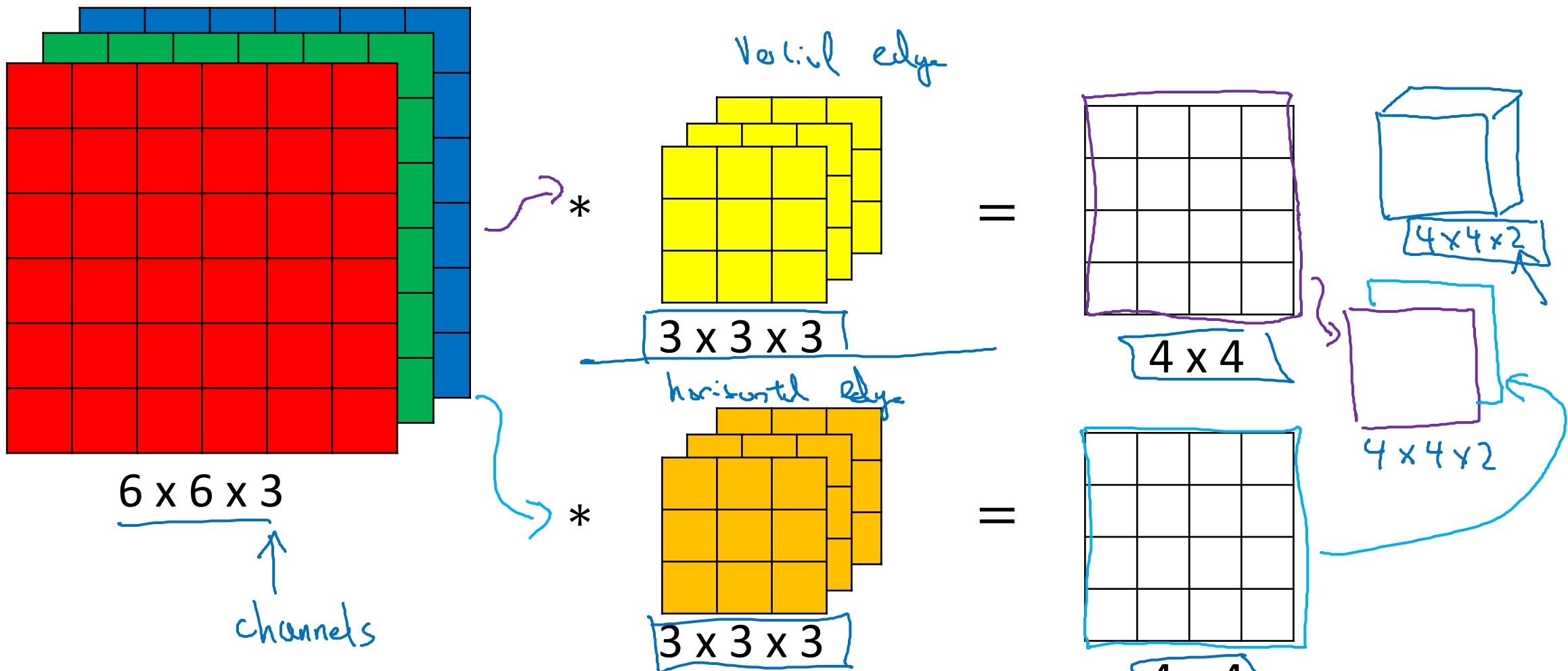
Convolutions on RGB images



Convolutions on RGB image



Multiple filters



Summary: $n \times n \times n_c$ $\times f \times f \times n_c$ $\rightarrow \frac{n-f+1}{4} \times \frac{n-f+1}{4} \times n'_c$ #filters

$6 \times 6 \times 3$ $3 \times 3 \times 3$

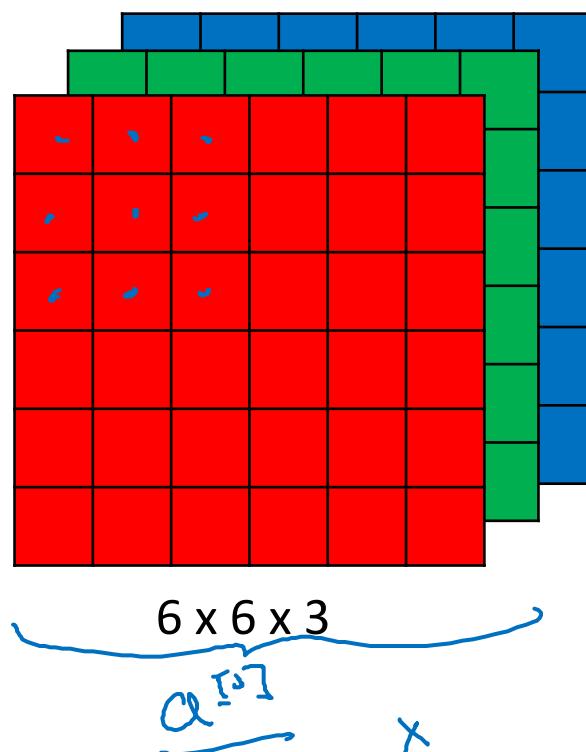


deeplearning.ai

Convolutional Neural Networks

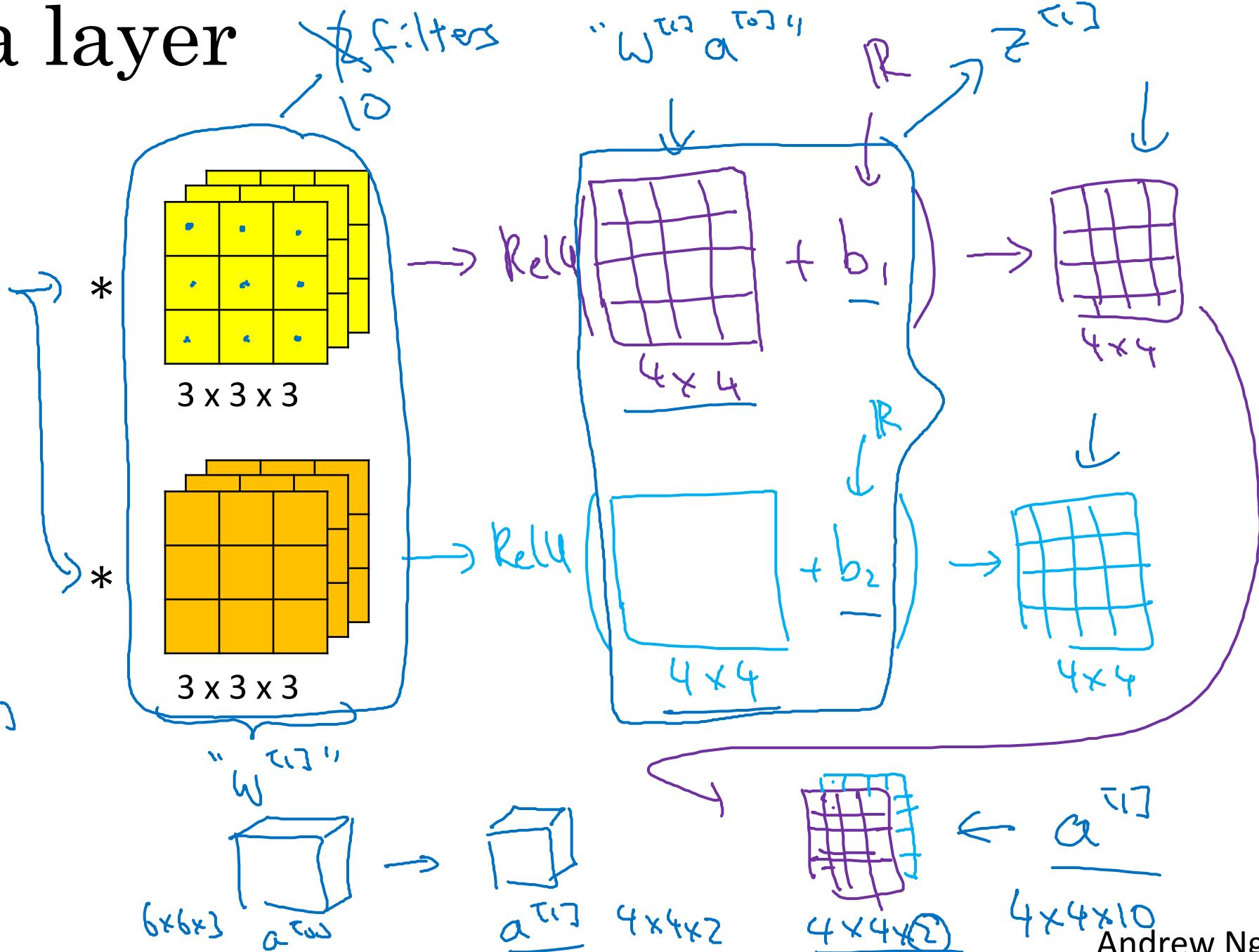
One layer of a
convolutional
network

Example of a layer



$$z^{(i)} = \omega^{(i)} a^{(i)} + b^{(i)}$$

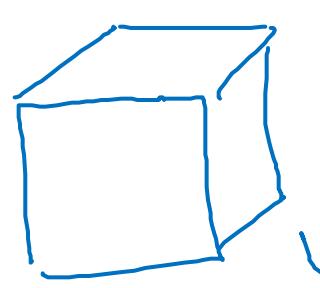
$$a^{[i,j]} = g(z^{[i,j]})$$



Andrew Ng

Number of parameters in one layer

If you have 10 filters that are $3 \times 3 \times 3$ in one layer of a neural network, how many parameters does that layer have?

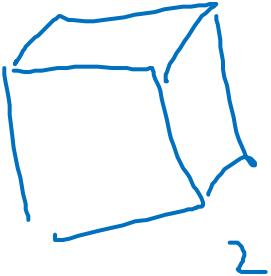


$3 \times 3 \times 3$

27 parameters.

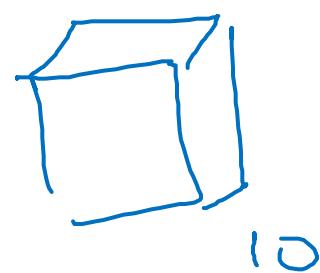
+ bias

→ 28 parameters.



2

...
...



10

280 parameters.

Summary of notation

If layer l is a convolution layer:

$f^{[l]}$ = filter size

$p^{[l]}$ = padding

$s^{[l]}$ = stride

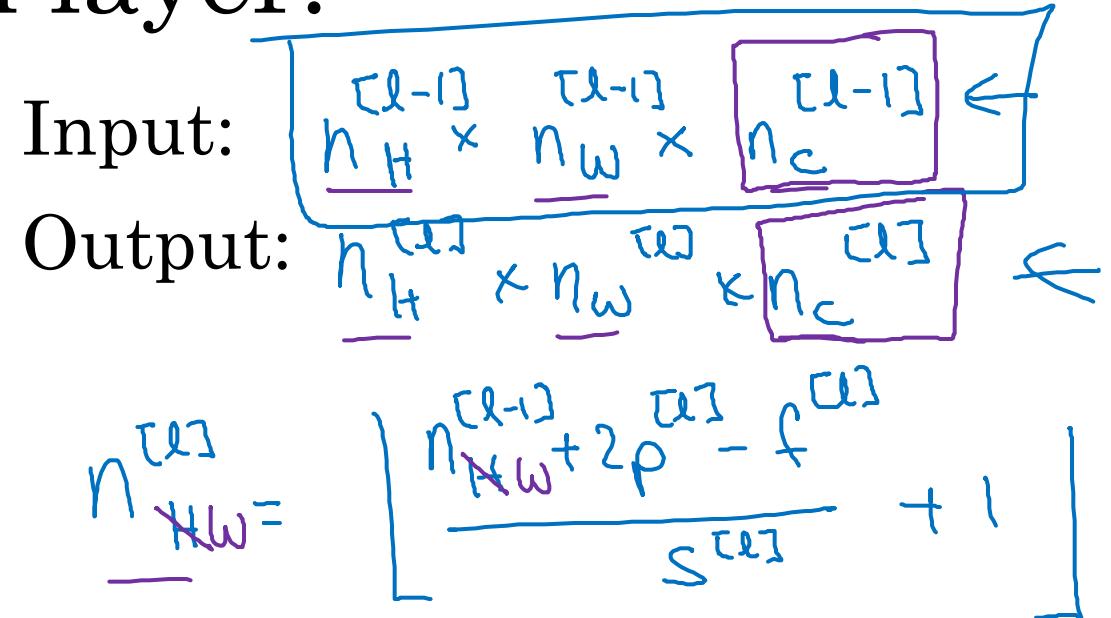
$n_c^{[l]}$ = number of filters

→ Each filter is: $f^{[l]} \times f^{[l]} \times n_c^{[l]}$

Activations: $a^{[l]} \rightarrow n_H^{[l]} \times n_W^{[l]} \times n_C^{[l]}$.

Weights: $f^{[l]} \times f^{[l]} \times n_c^{[l-1]} \times n_c^{[l]}$

bias: $n_c^{[l]} - (1, 1, 1, n_c^{[l]})$ ↪ #f: #ftrs in layer l.



$$A^{[l]} \rightarrow m \times n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$$

$$n_c \times n_H \times n_W$$

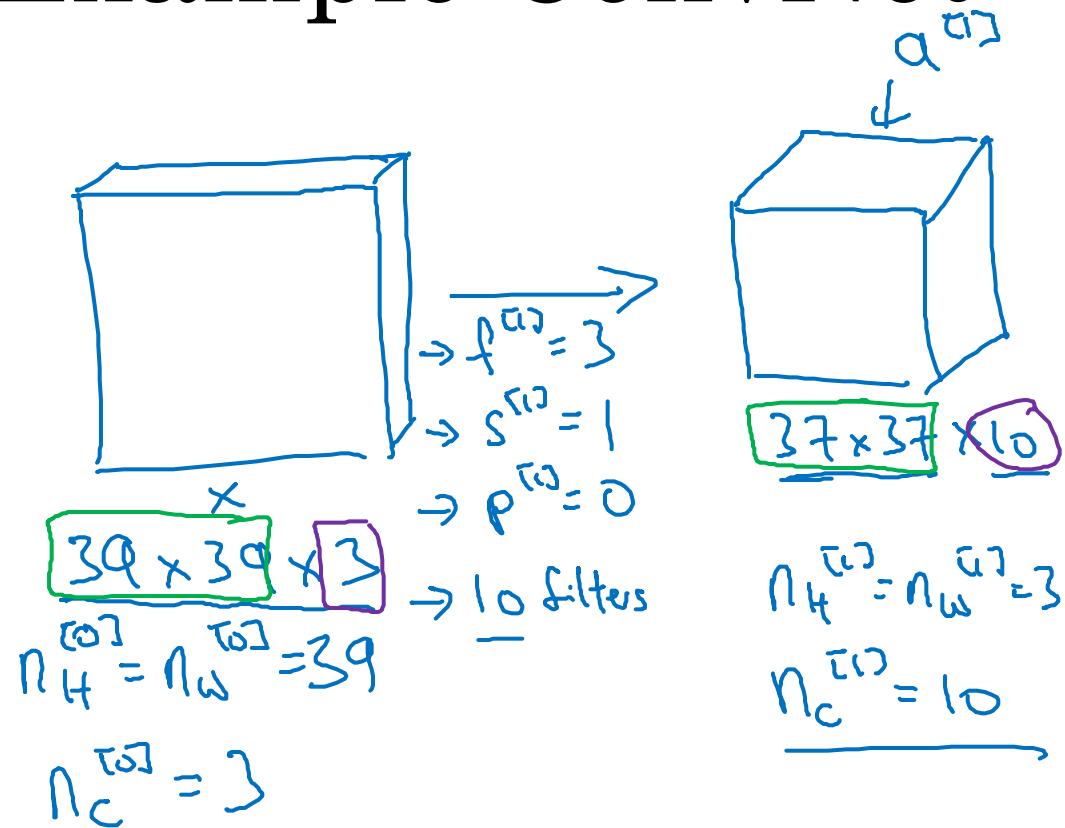


deeplearning.ai

Convolutional Neural Networks

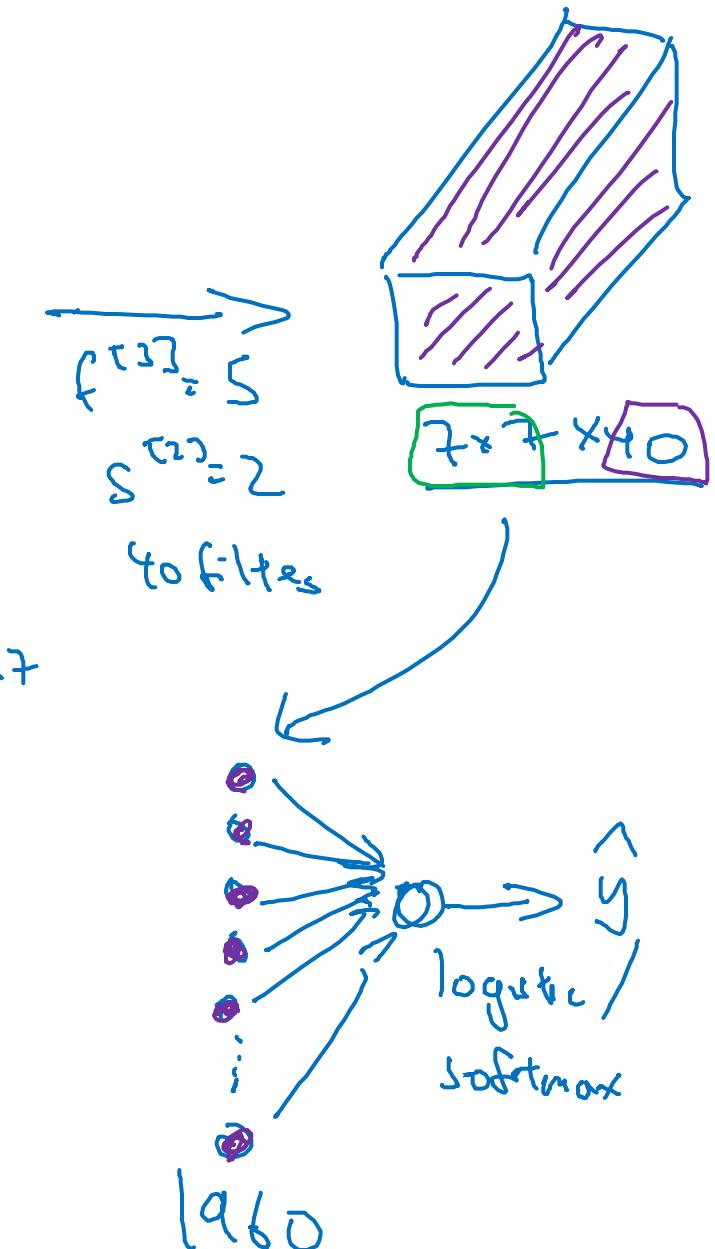
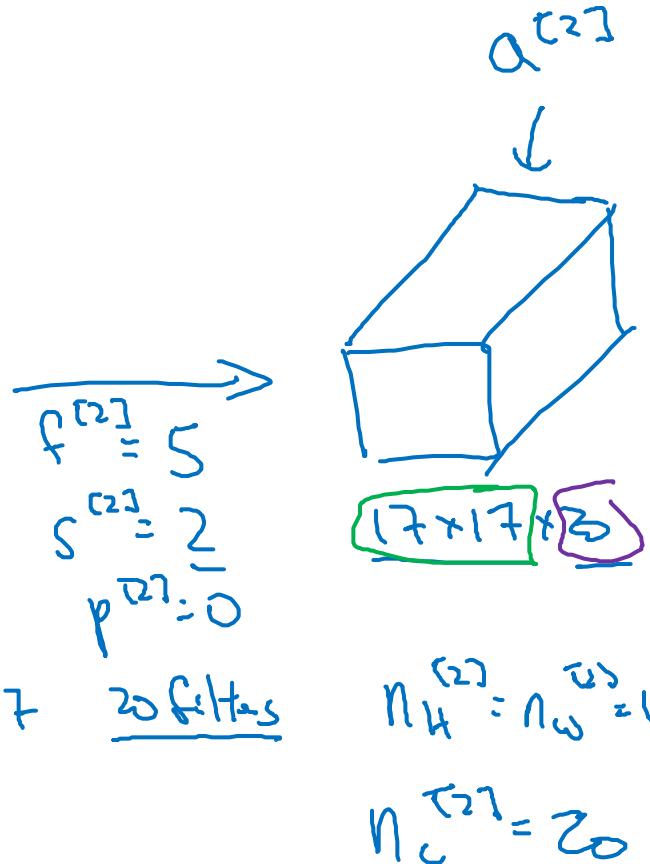
A simple convolution network example

Example ConvNet



$$\frac{n+2p-f}{s} + 1$$

$$\frac{39+0-3}{1} + 1 = 37$$



Types of layer in a convolutional network:

- Convolution (Conv) ←
- Pooling (pool) ←
- Fully connected (Fc) ←



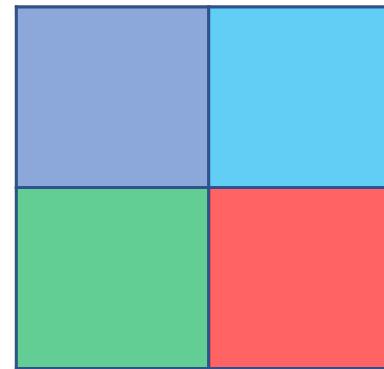
deeplearning.ai

Convolutional Neural Networks

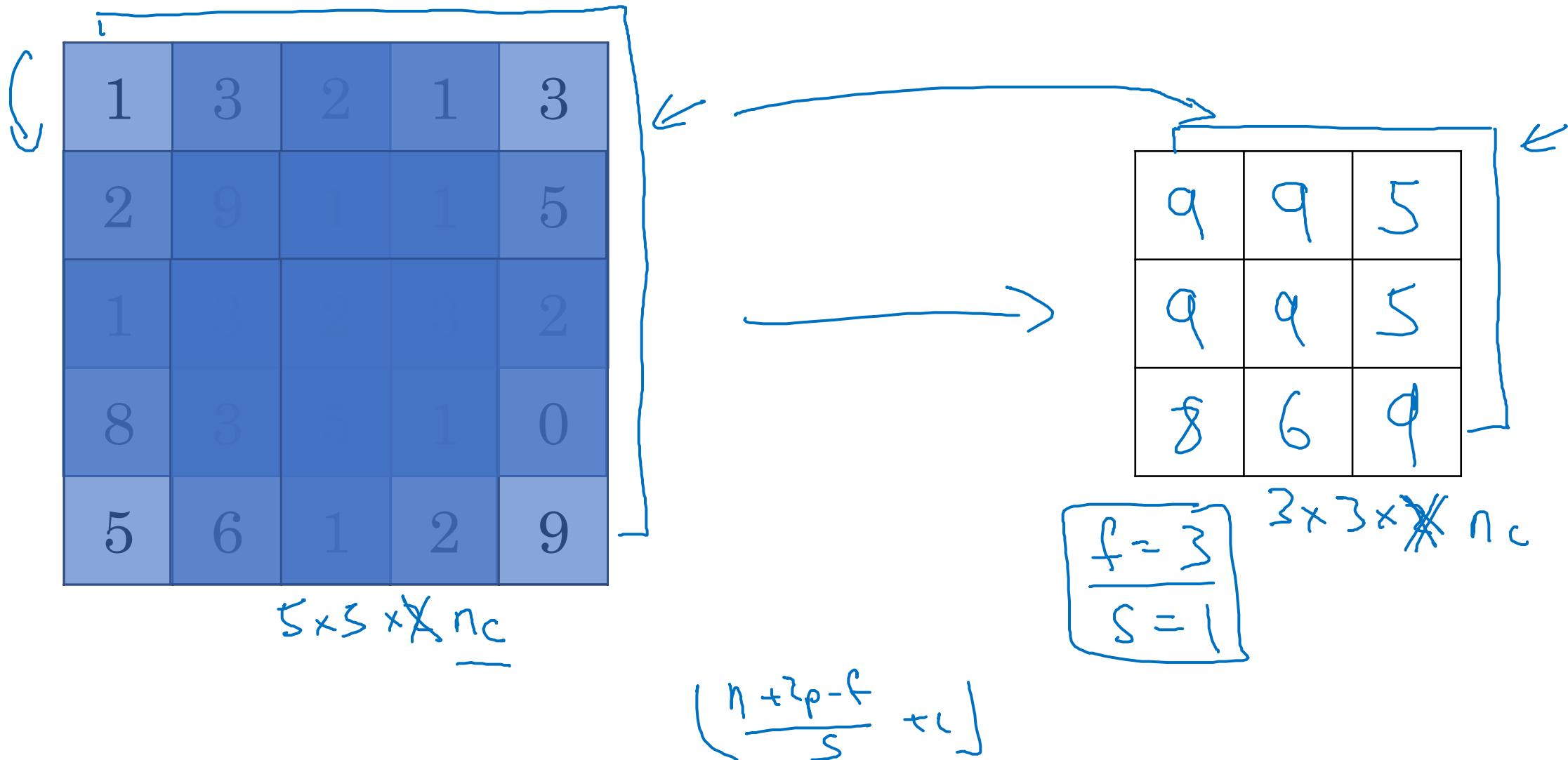
Pooling layers

Pooling layer: Max pooling

1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2



Pooling layer: Max pooling



Pooling layer: Average pooling

1	3	2	1
2	9	1	1
1	4	2	3
5	6	1	2



3.75	1.25
4	2

$$f=2$$

$$s=2$$

$$\underbrace{7 \times 7 \times 1000}_{\rightarrow} \rightarrow 1 \times 1 \times 1000$$

Summary of pooling

Hyperparameters:

f : filter size

$$f=2, s=2$$

s : stride

$$f=3, s=2$$

Max or average pooling

$\rightarrow p$: padding.

No parameters to learn!

$$n_H \times n_w \times n_c$$



$$\left\lfloor \frac{n_H-f+1}{s} \right\rfloor \times \left\lfloor \frac{n_w-f}{s} + 1 \right\rfloor$$

$$\times n_c$$



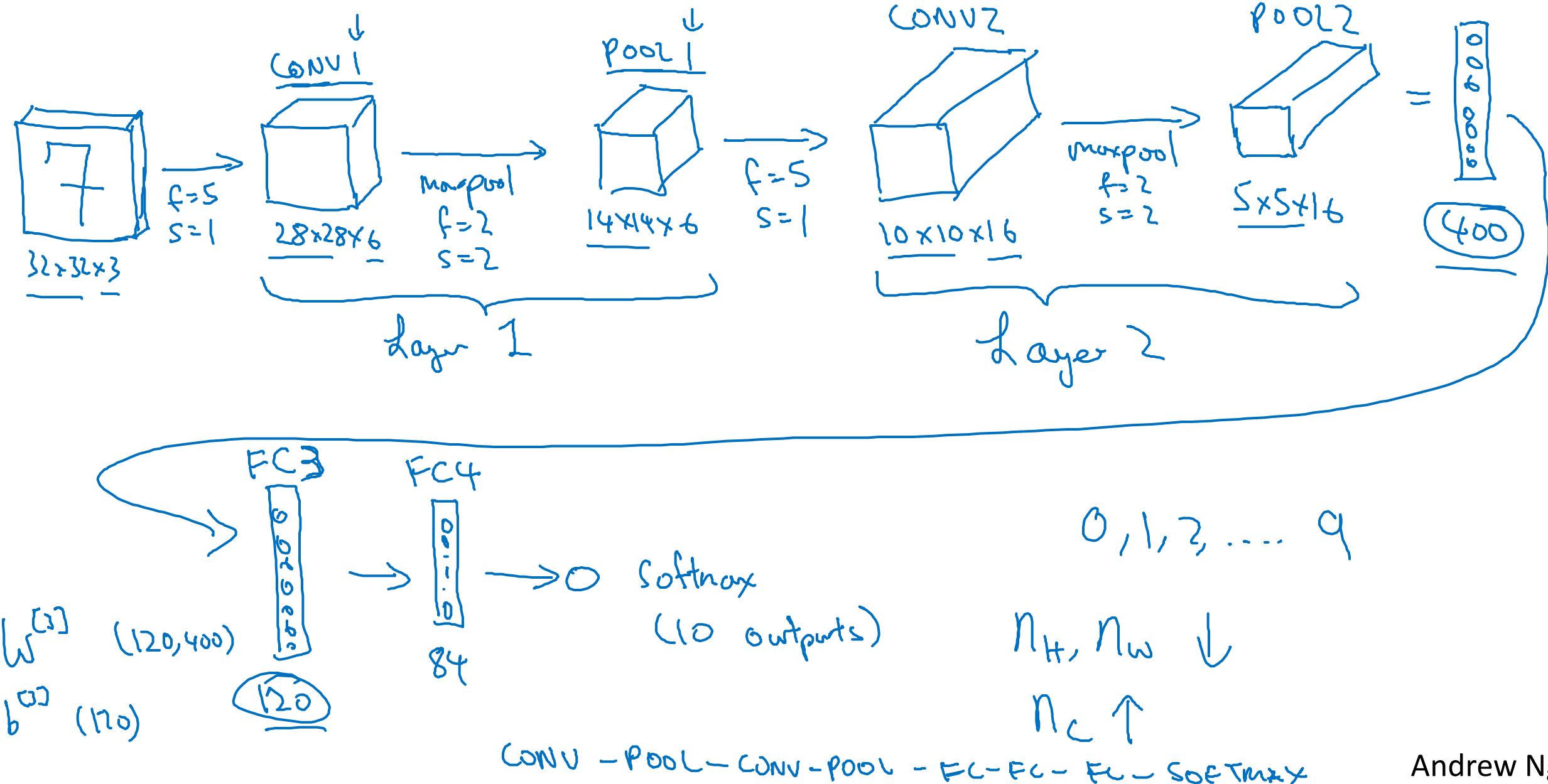
deeplearning.ai

Convolutional Neural Networks

Convolutional neural network example

Neural network example

(LeNet-5)



Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[3]}$	0
CONV1 (f=5, s=1)	(28,28,8)	6,272	608 ←
POOL1	(14,14,8)	1,568	0 ←
CONV2 (f=5, s=1)	(10,10,16)	1,600	3216 ←
POOL2	(5,5,16)	400	0 ←
FC3	(120,1)	120	48120 {
FC4	(84,1)	84	10164 }
Softmax	(10,1)	10	850

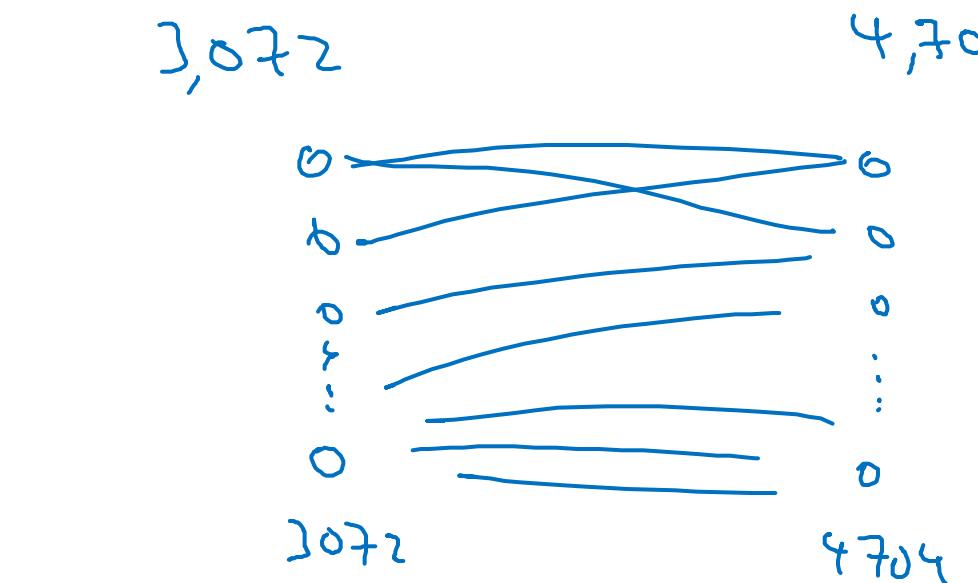
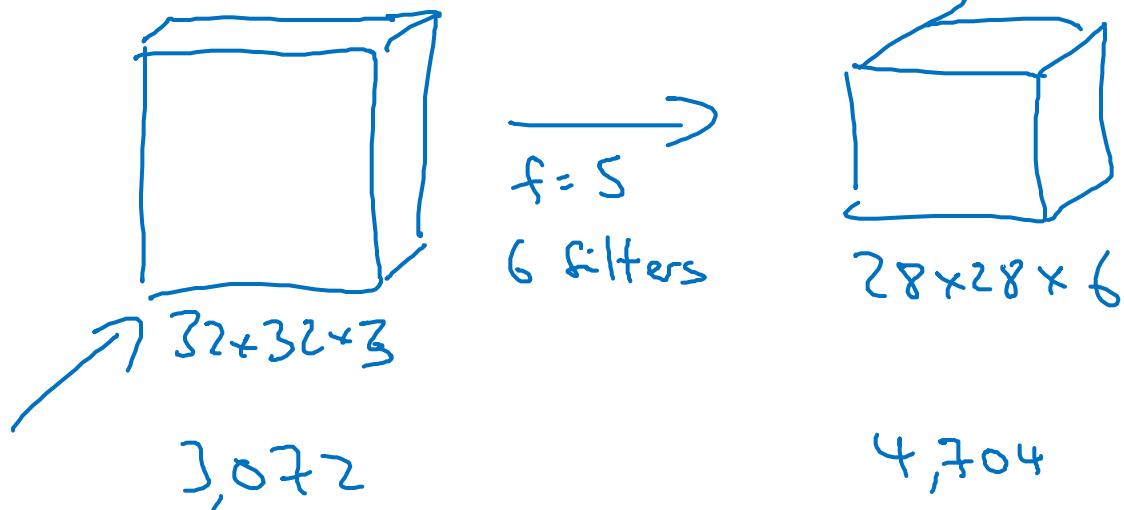


deeplearning.ai

Convolutional Neural Networks

Why convolutions?

Why convolutions



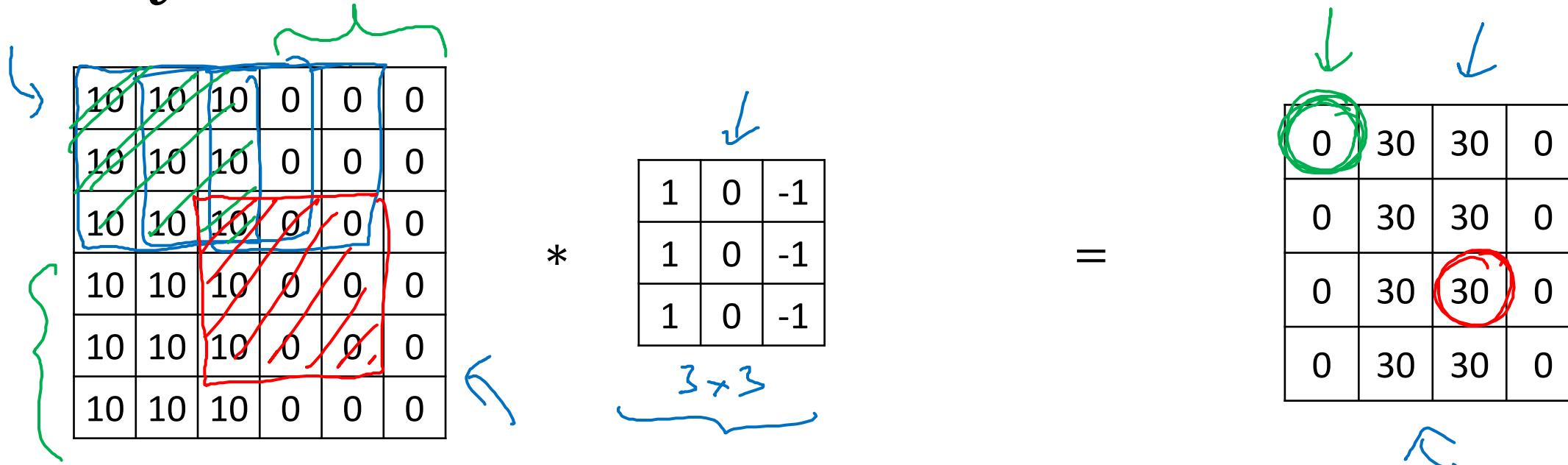
$$5 \times 5 = 25$$

26

$$6 \times 26 = 156 \text{ Parameters}$$

$$3,072 \times 4,704 \approx 14m$$

Why convolutions

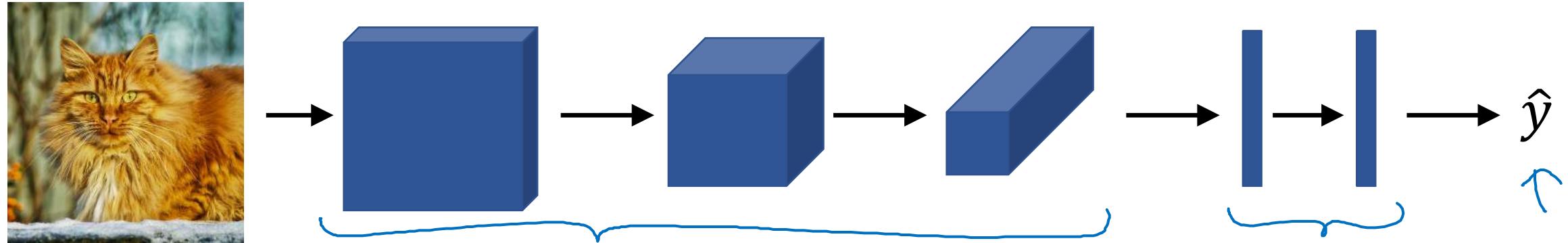


Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **Sparsity of connections:** In each layer, each output value depends only on a small number of inputs.

Putting it together

Training set $(x^{(1)}, y^{(1)}) \dots (x^{(m)}, y^{(m)})$.



$$\text{Cost } J = \frac{1}{m} \sum_{i=1}^m \mathcal{L}(\hat{y}^{(i)}, y^{(i)})$$

Use gradient descent to optimize parameters to reduce J

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

Case Studies

Why look at
case studies?

Outline

Classic networks:

- LeNet-5 ←
- AlexNet ←
- VGG ←

ResNet (152)

Inception

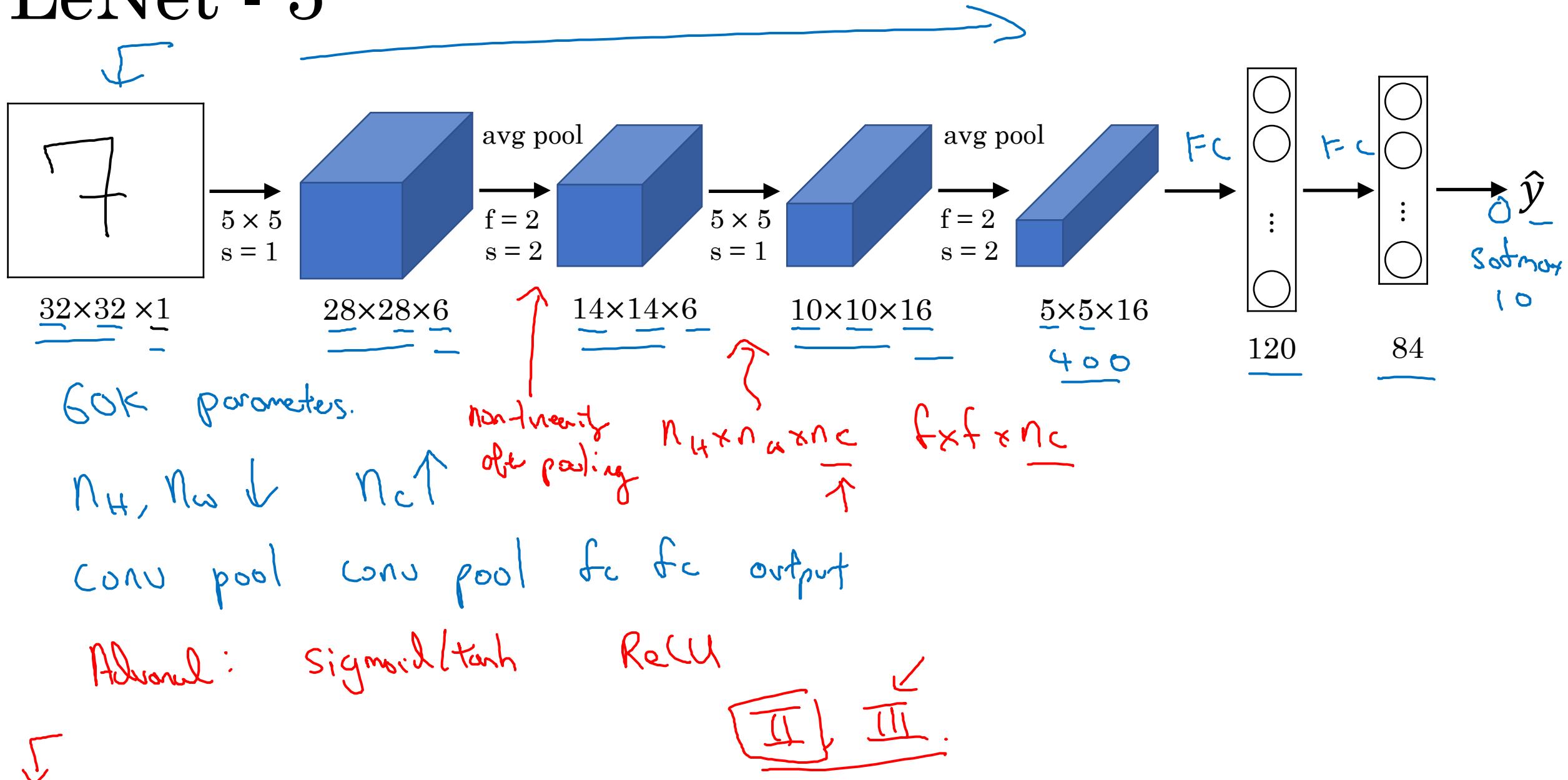


deeplearning.ai

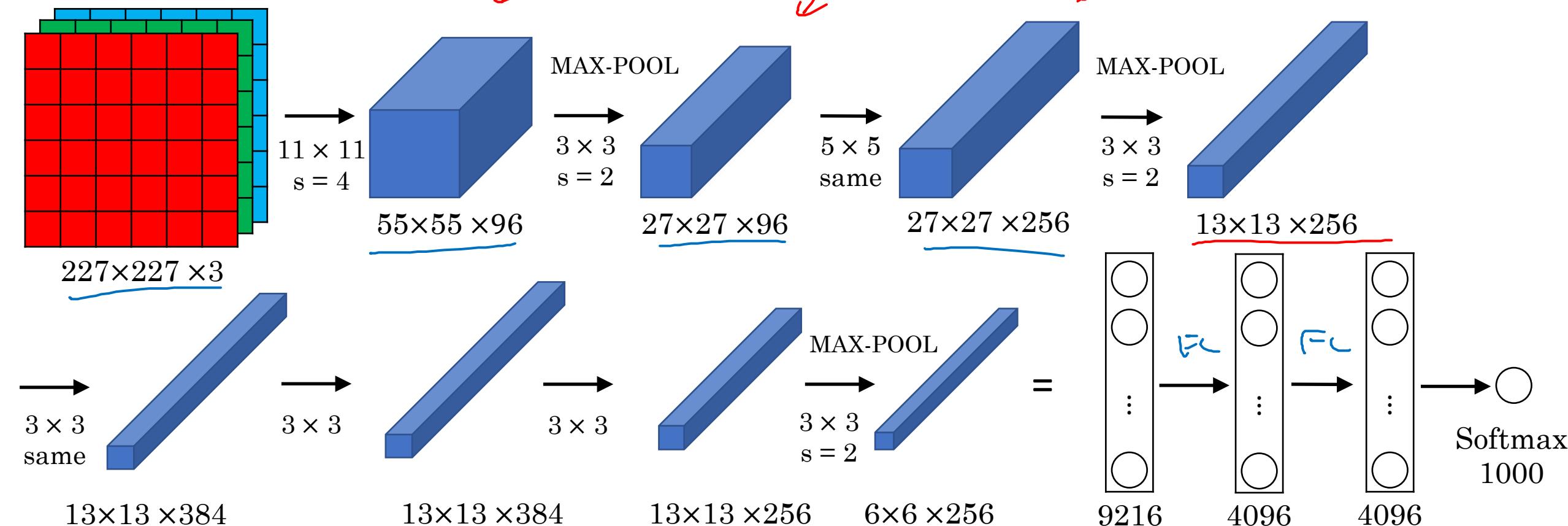
Case Studies

Classic networks

LeNet - 5



AlexNet

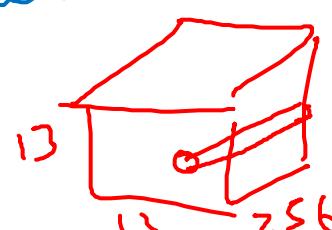


- Similar to LeNet, but much bigger.

- ReLU

- Multiple GPUs.

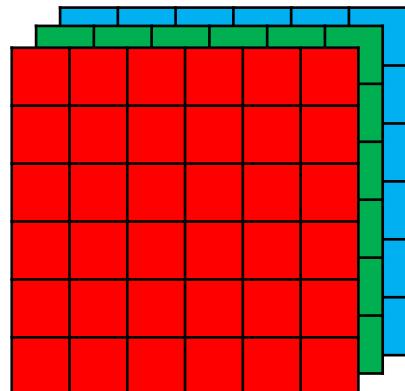
- Local Response Normalization (LRN)



~60M parameters

VGG - 16

CONV = 3×3 filter, $s = 1$, same



224×224

[CONV 64]
 $\times 2$

$224 \times 224 \times 64$ → POOL → $112 \times 112 \times 64$

$112 \times 112 \times 64$

[CONV 128]
 $\times 2$

$112 \times 112 \times 128$ → POOL → $56 \times 56 \times 128$

224×224

$56 \times 56 \times 256$ → POOL → $28 \times 28 \times 256$ → [CONV 256]
 $\times 3$ → $28 \times 28 \times 512$ → [CONV 512]
 $\times 3$ → $14 \times 14 \times 512$ → POOL → $7 \times 7 \times 512$

[CONV 512]
 $\times 3$ → $14 \times 14 \times 512$ → POOL → $7 \times 7 \times 512$ → FC 4096 → FC 4096 → Softmax 1000

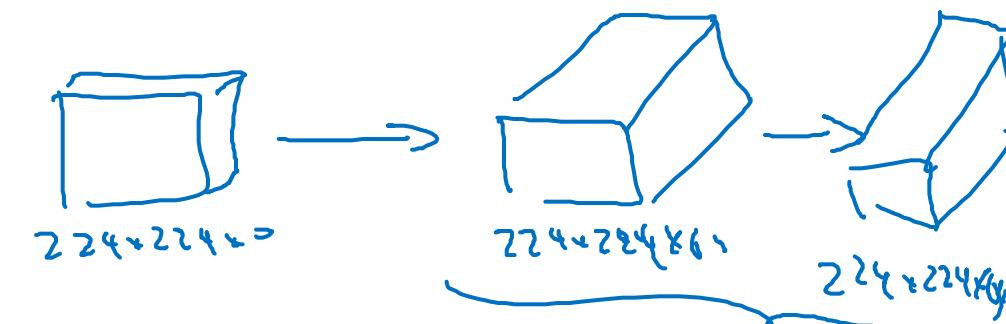
$n_H, n_W \downarrow$

$n_C \uparrow$

$\sim 38M$

VGG-19

MAX-POOL = 2×2 , $s = 2$



$112 \times 112 \times 128$ → POOL → $56 \times 56 \times 128$

POOL

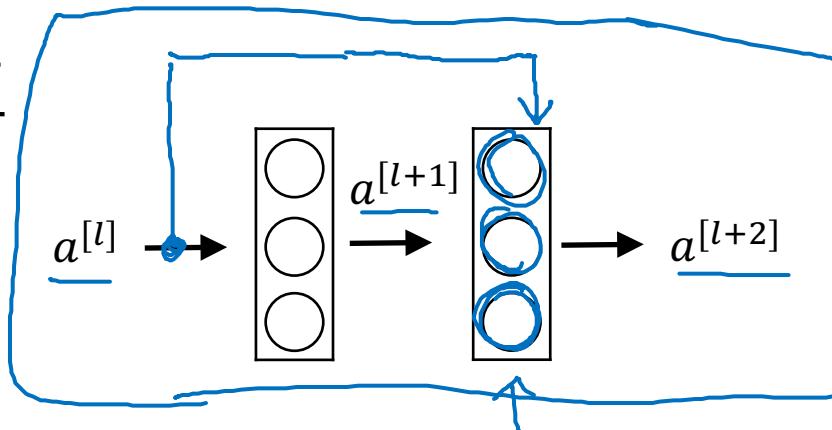


deeplearning.ai

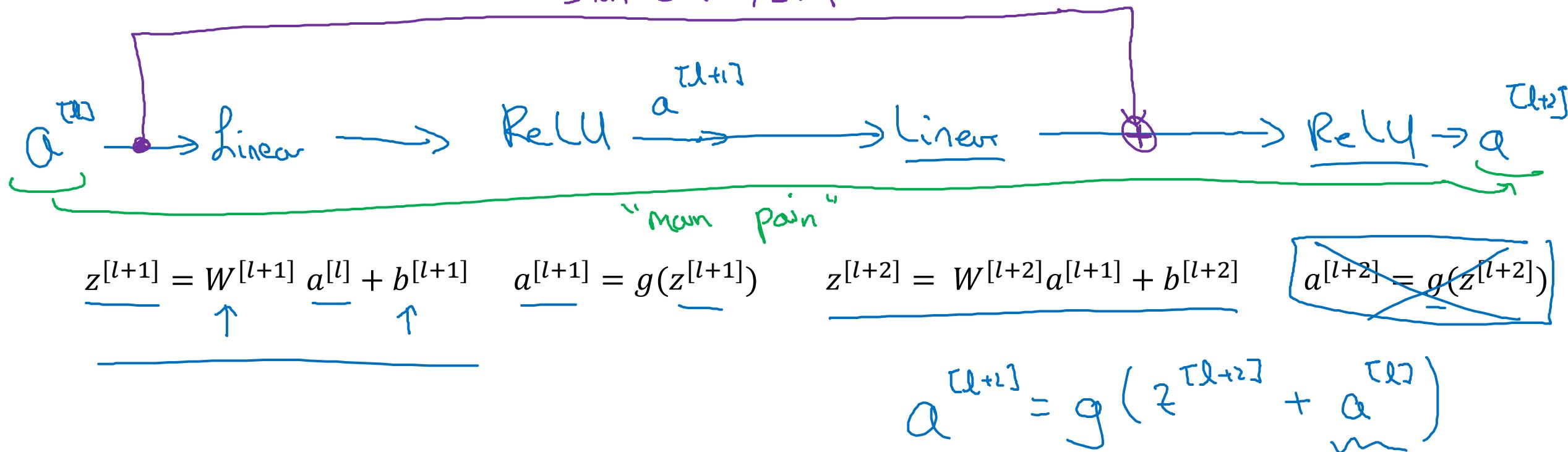
Case Studies

Residual Networks (ResNets)

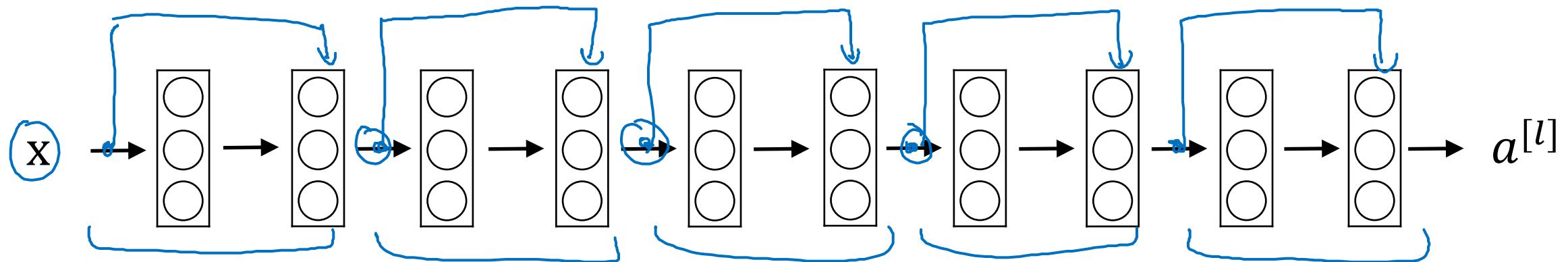
Residual block



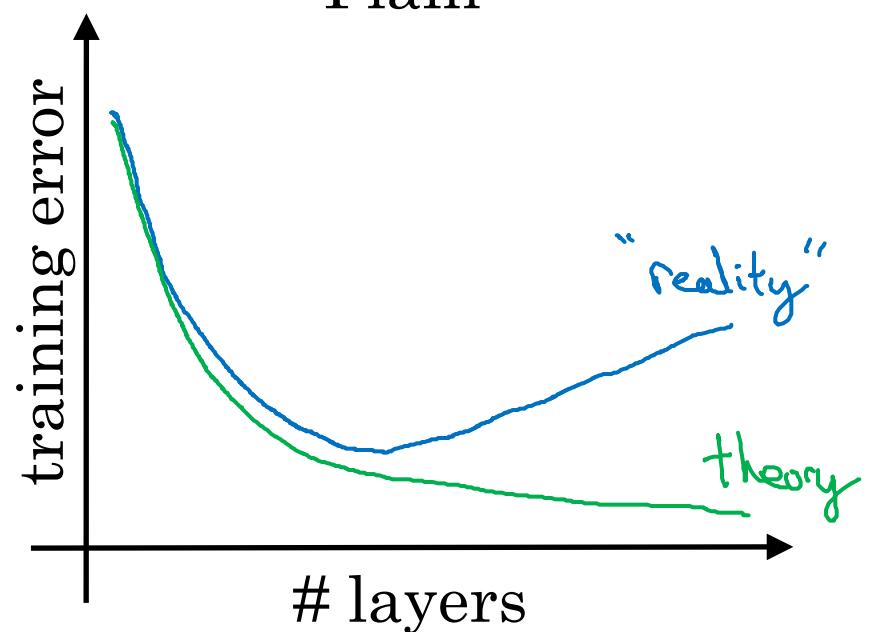
"Short cut" /skip connection



Residual Network

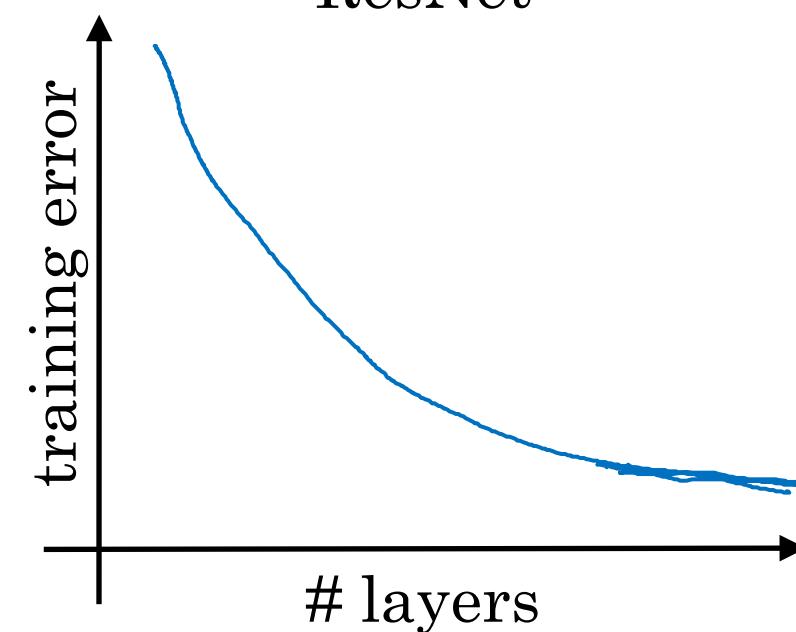


Plain



"Plain network"

ResNet



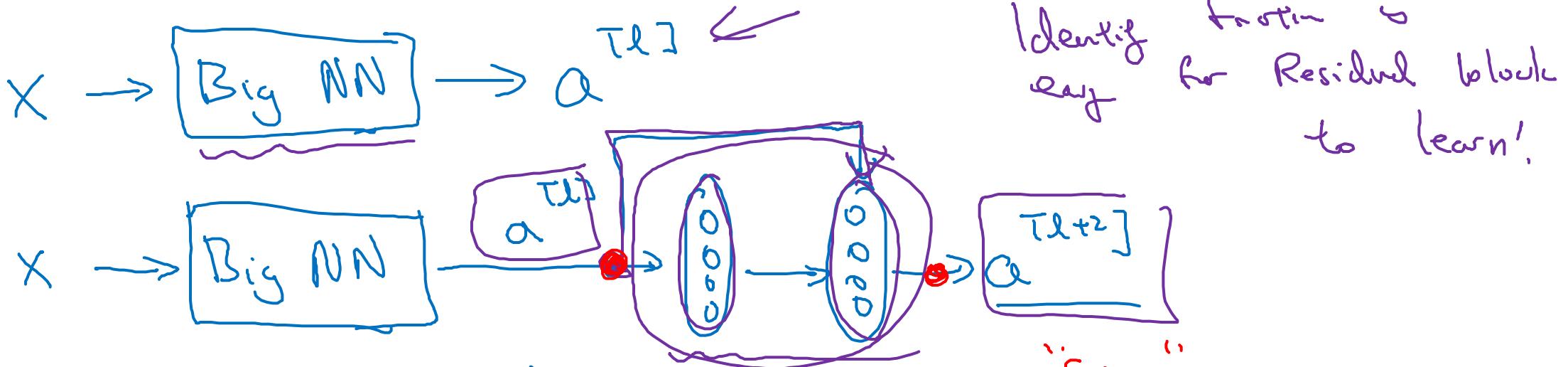


deeplearning.ai

Case Studies

Why ResNets work

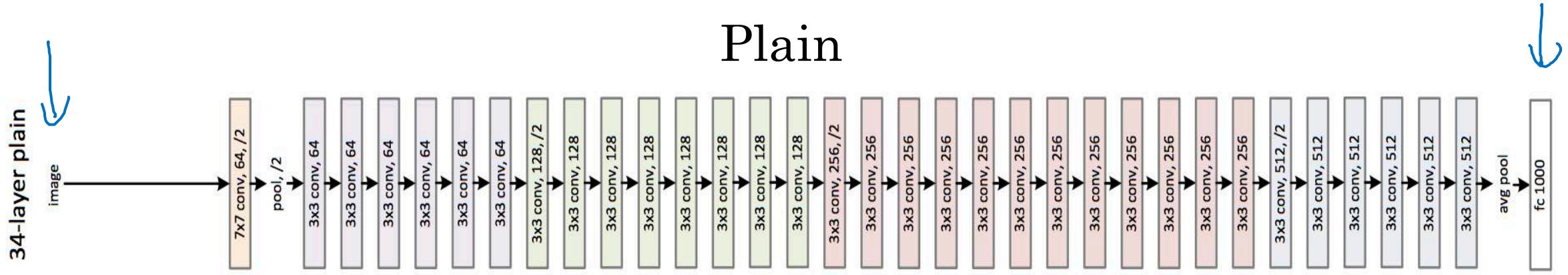
Why do residual networks work?



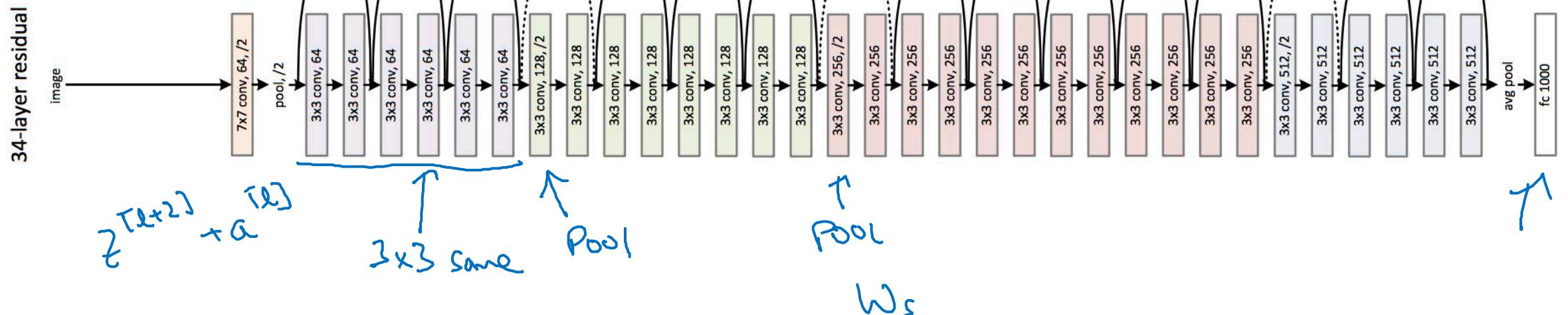
ReLU.

$$\begin{aligned}
 a^{[l+2]} &= g(z^{[l+2]} + a^{[l]}) \\
 &= g(w^{[l+2]} a^{[l+1]} + b^{[l+2]} + a^{[l]}) \\
 &\quad \text{If } w^{[l+2]} = 0, b^{[l+2]} = 0 \quad R^{256 \times 128} \\
 &= g(a^{[l]}) \\
 &= a^{[l]}
 \end{aligned}$$

ResNet



ResNet





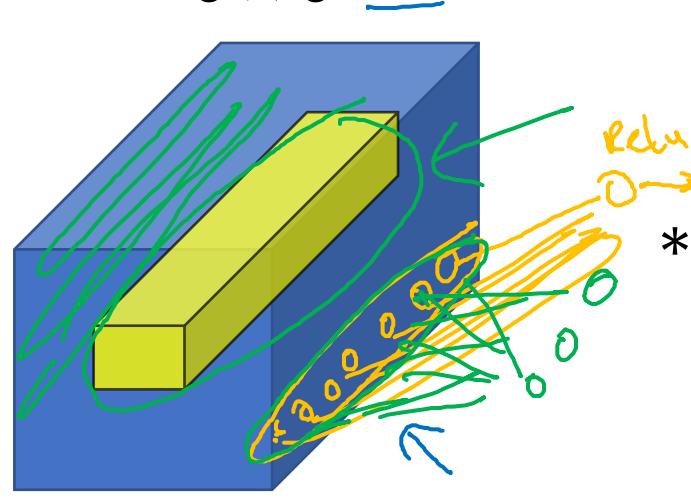
deeplearning.ai

Case Studies

Network in Network and 1×1 convolutions

Why does a 1×1 convolution do?

1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5



*

2



=

32 \rightarrow # filters.

$n_c^{[l+1]}$

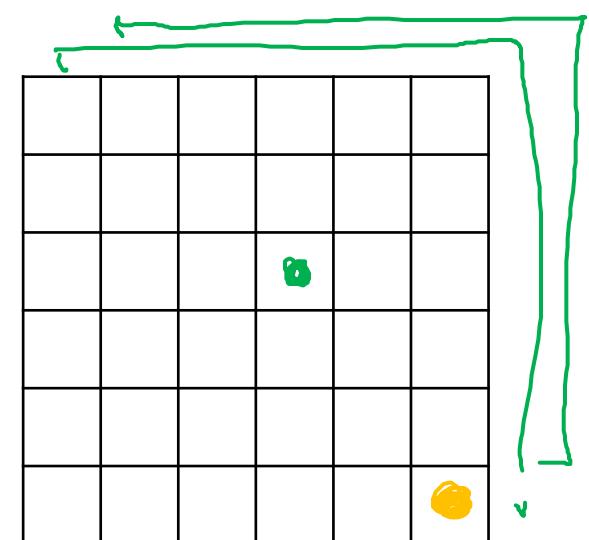
=

ReLU

Network \hookrightarrow Network

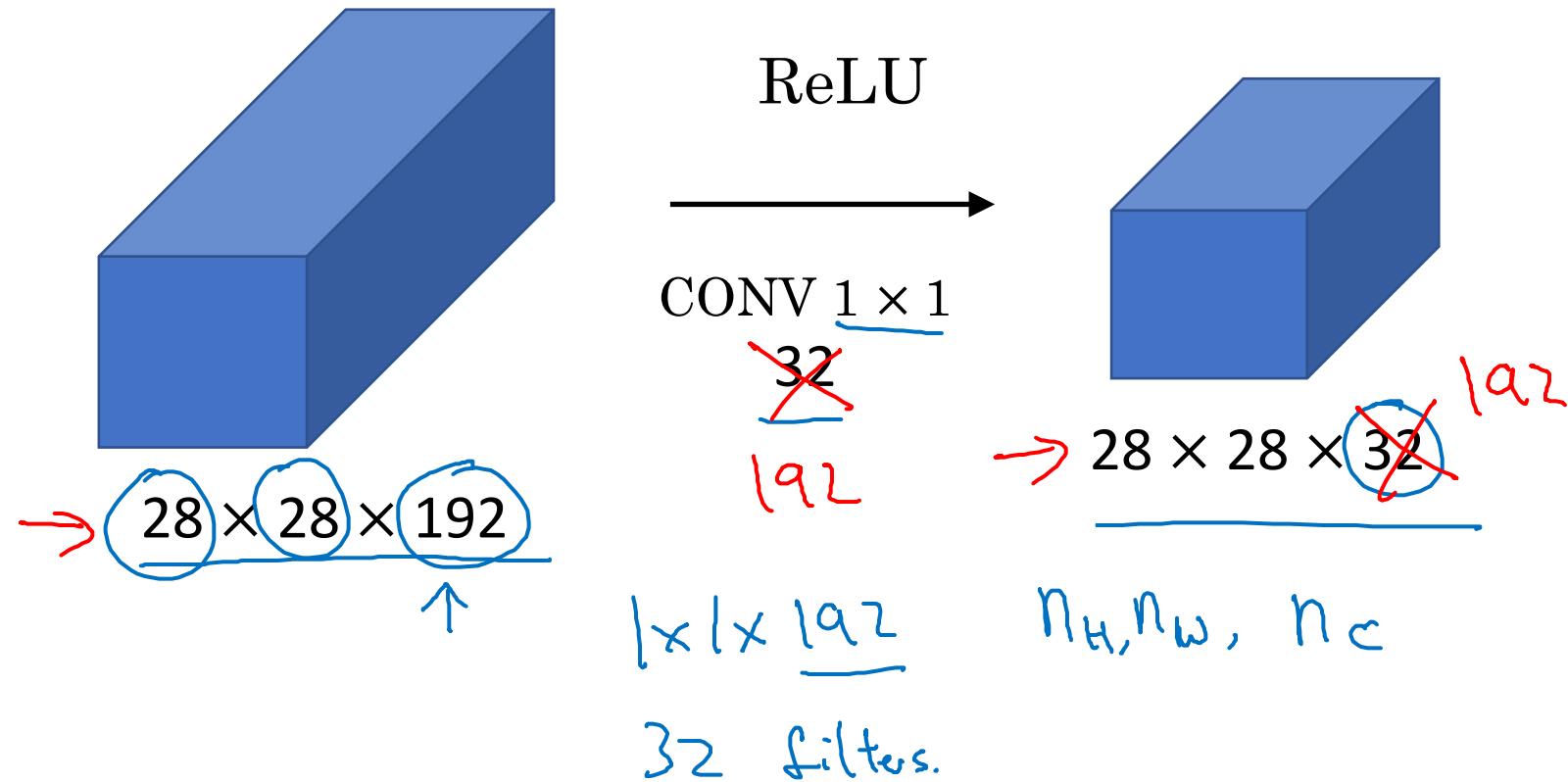
1 \times 1 \times 32

2	4	6	...



Andrew Ng

Using 1×1 convolutions



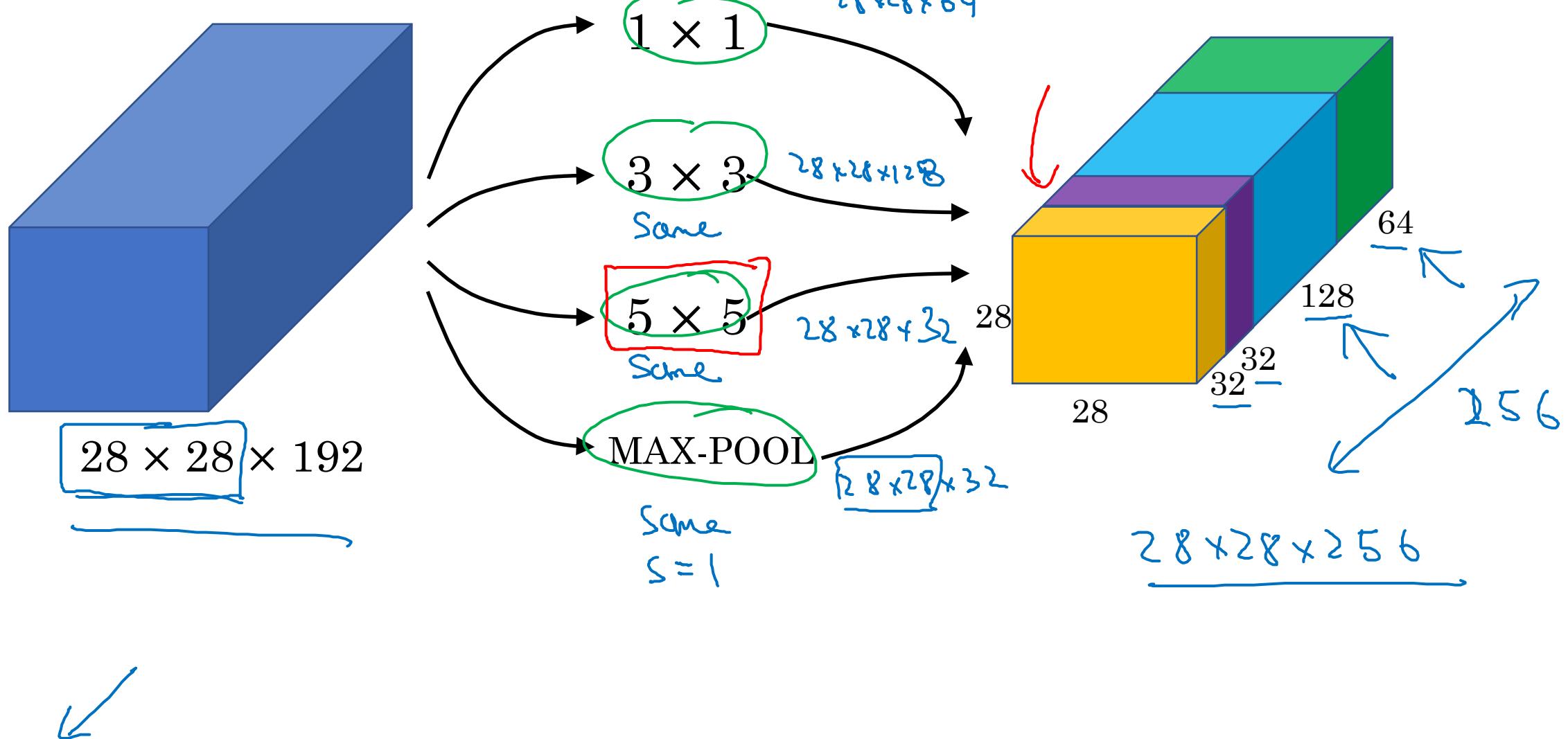


deeplearning.ai

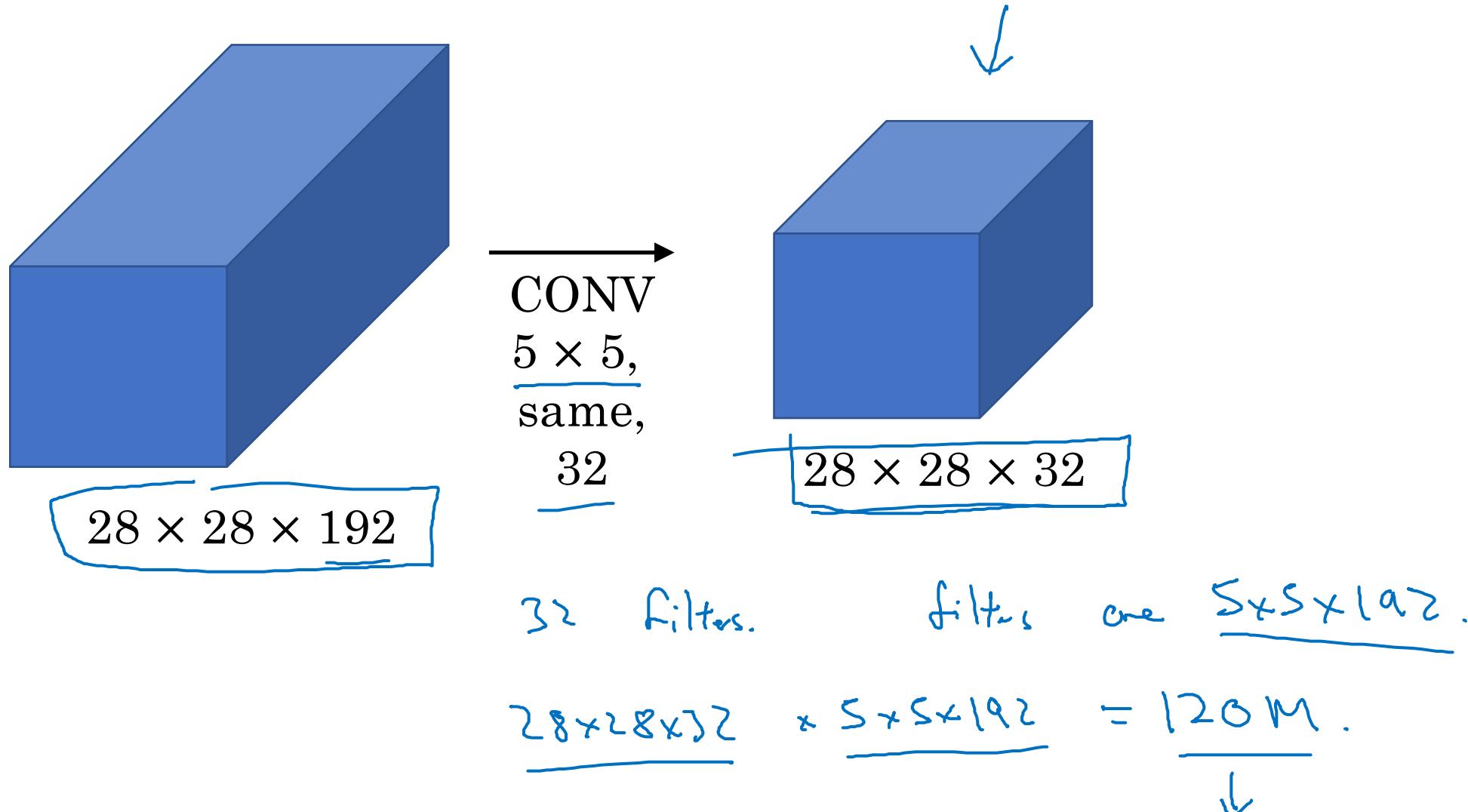
Case Studies

Inception network
motivation

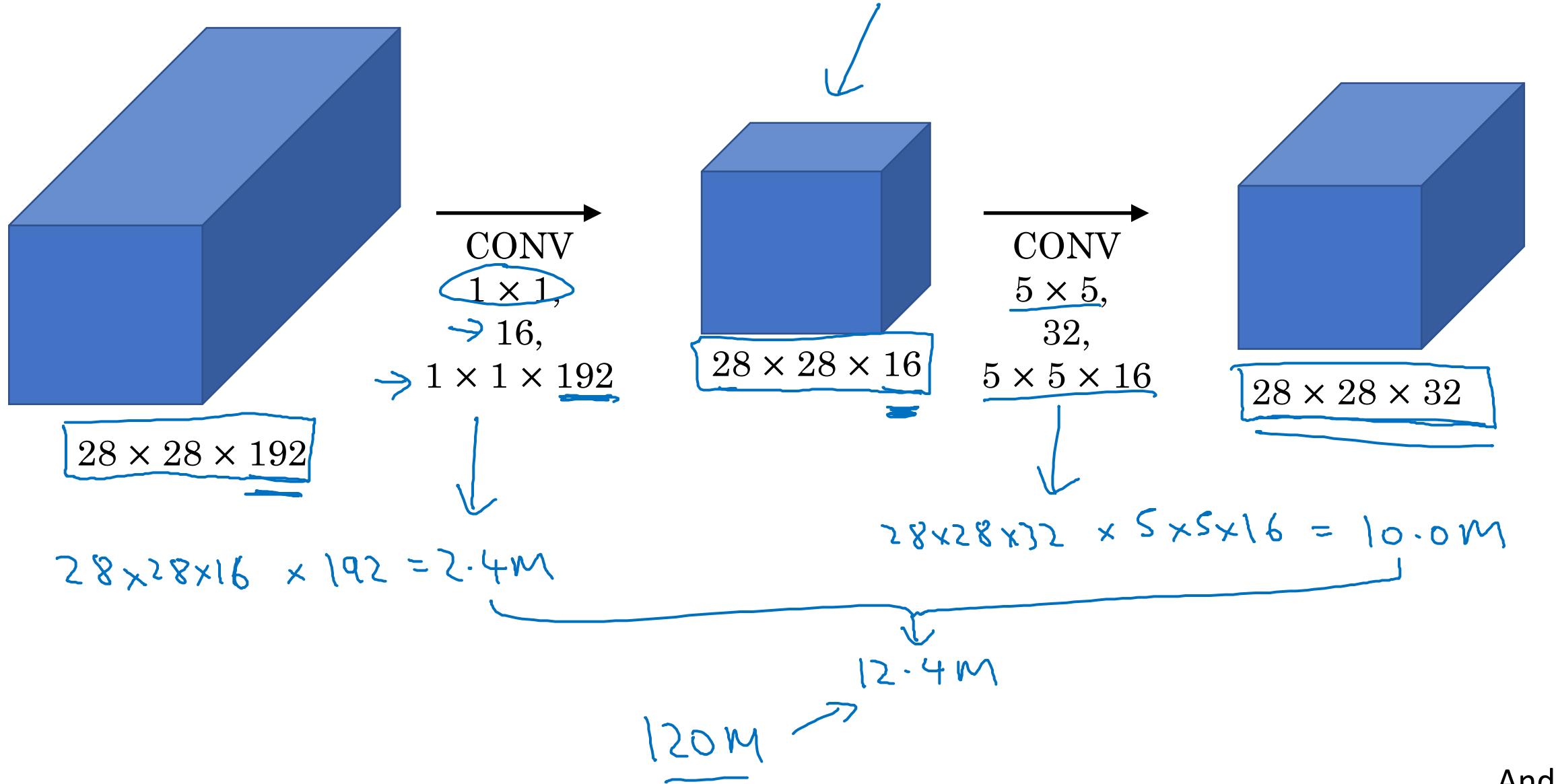
Motivation for inception network



The problem of computational cost



Using 1×1 convolution



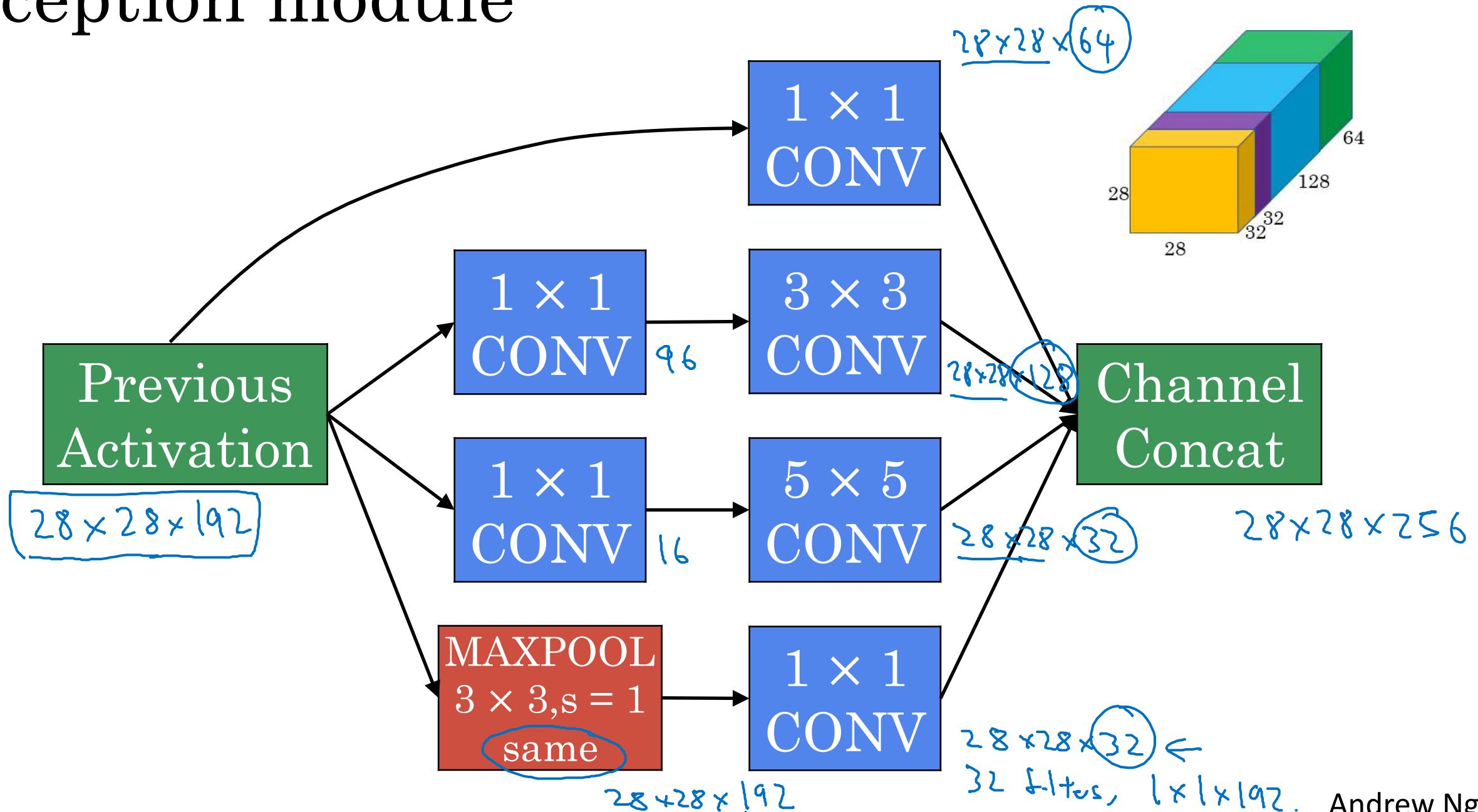


deeplearning.ai

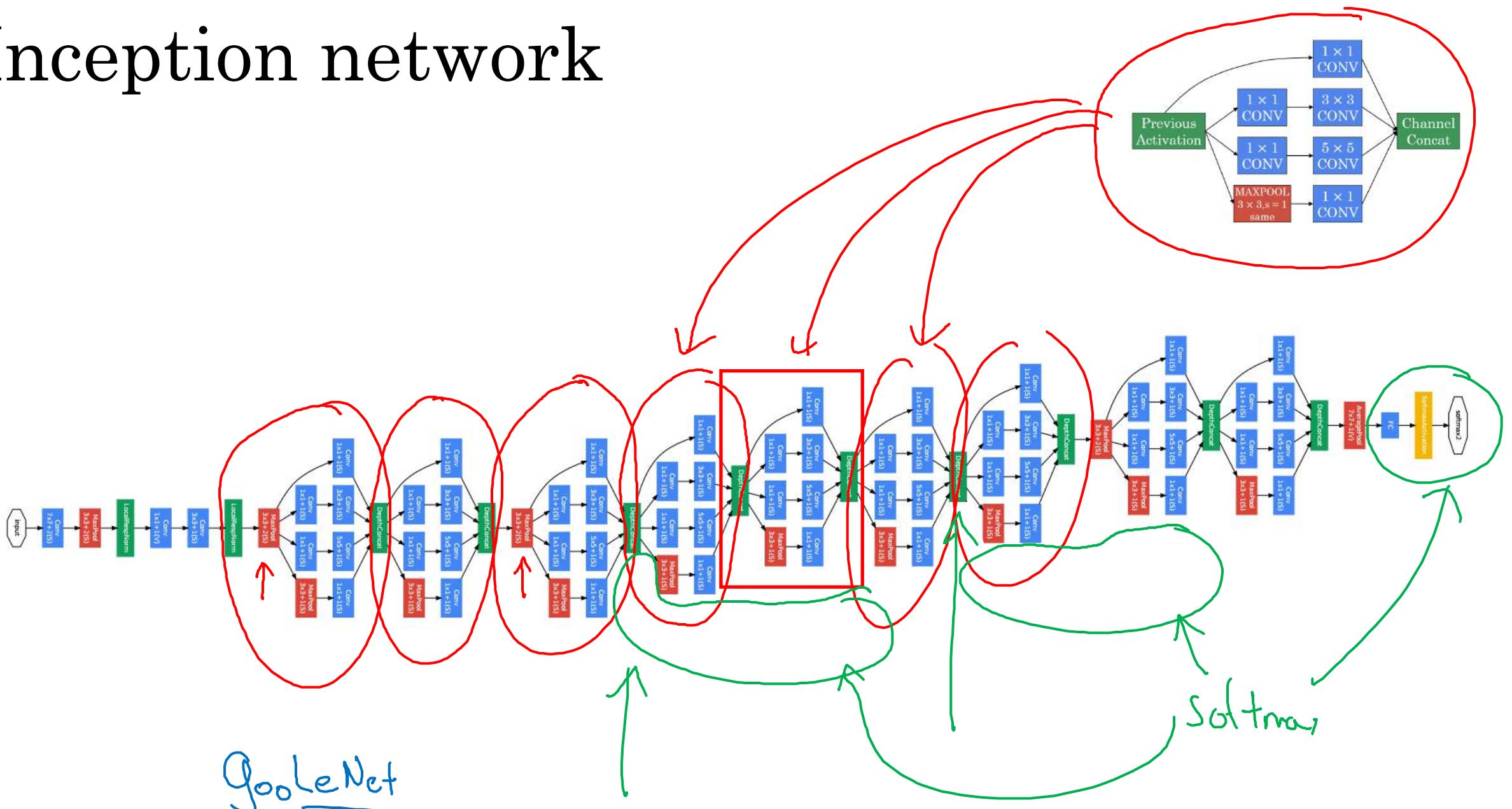
Case Studies

Inception network

Inception module



Inception network







deeplearning.ai

Convolutional Neural Networks

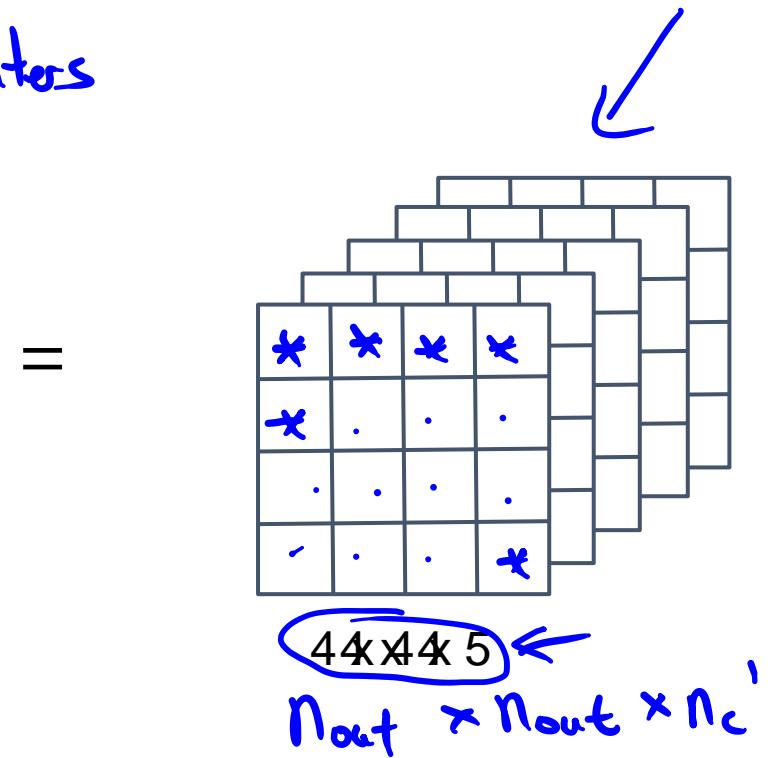
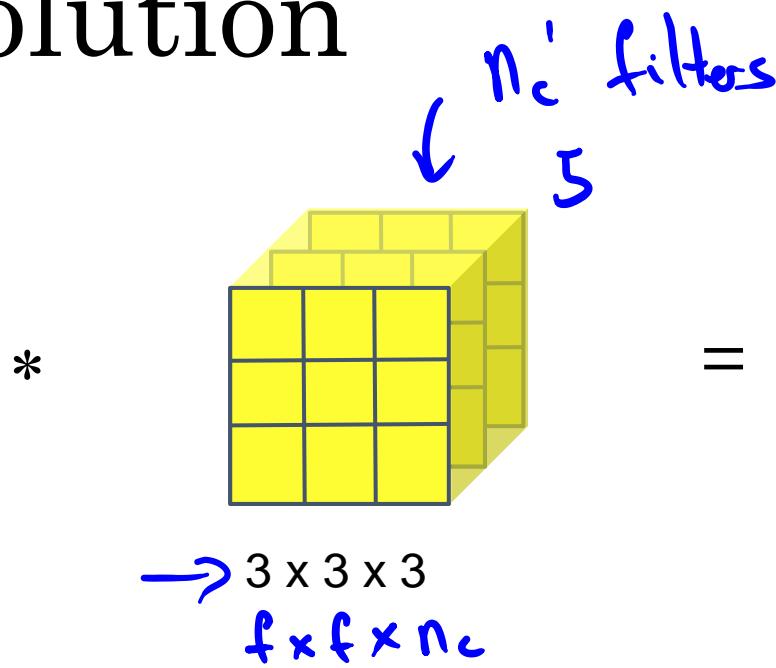
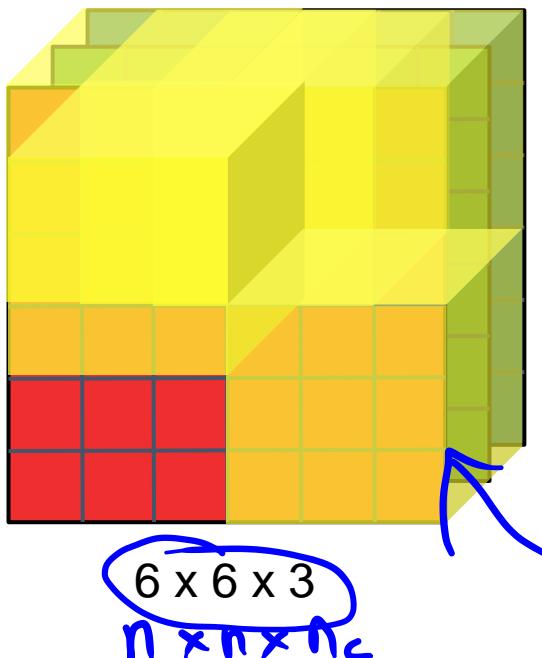
MobileNet

Motivation for MobileNets

- Low computational cost at deployment
- Useful for mobile and embedded vision applications
- Key idea: Normal vs. depthwise-separable convolutions



Normal Convolution



$$\text{Computational cost} = \frac{\# \text{filter params}}{3 \times 3 \times 3} \times \frac{\# \text{filter positions}}{4 \times 4}$$

$$\rightarrow \underline{2160} = \underline{\uparrow} \quad \underline{\uparrow}$$

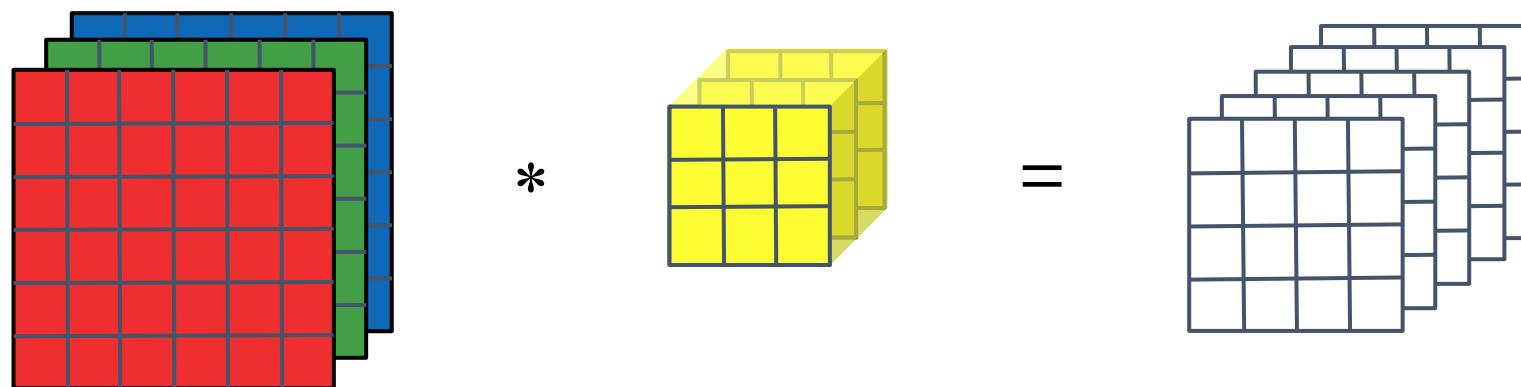
$$x \quad \# \text{of filters}$$

$$\times \quad 5$$

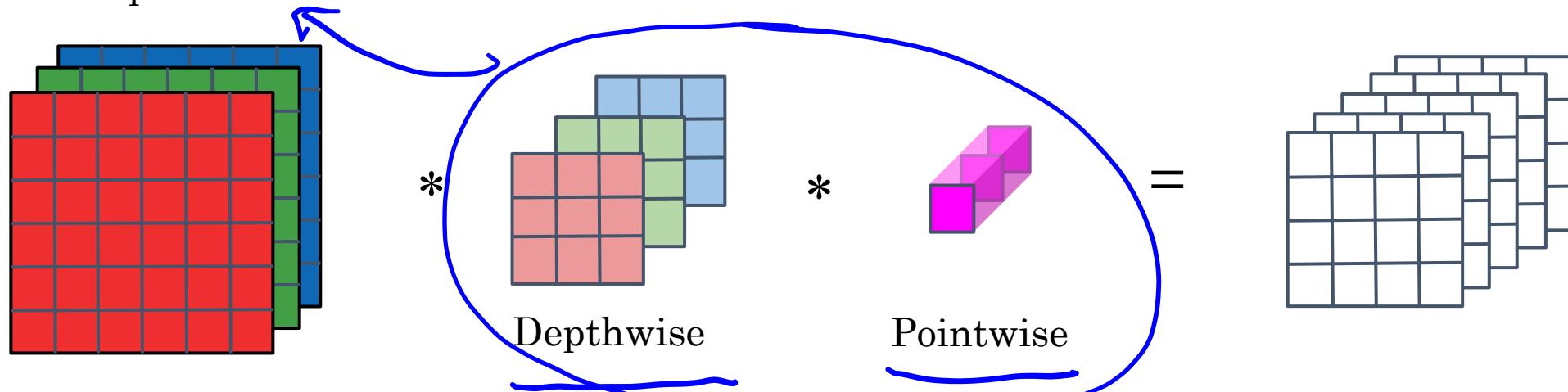
$$\uparrow$$

Depthwise Separable Convolution

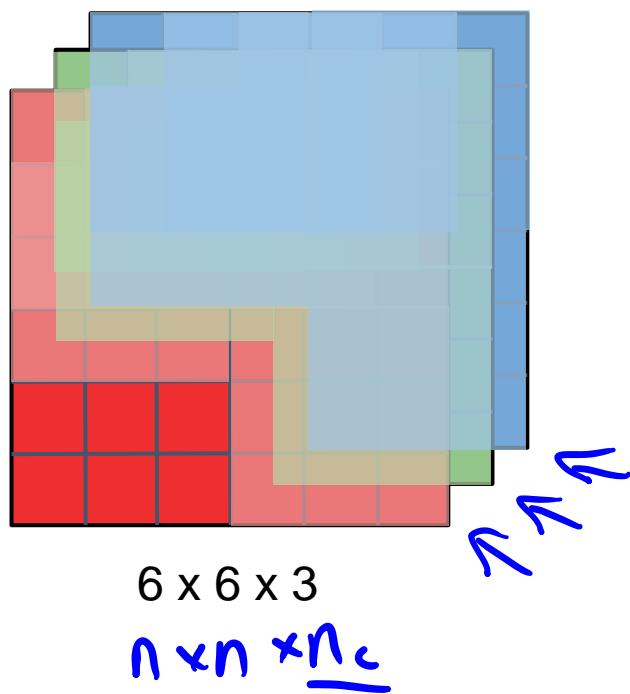
Normal Convolution



Depthwise Separable Convolution

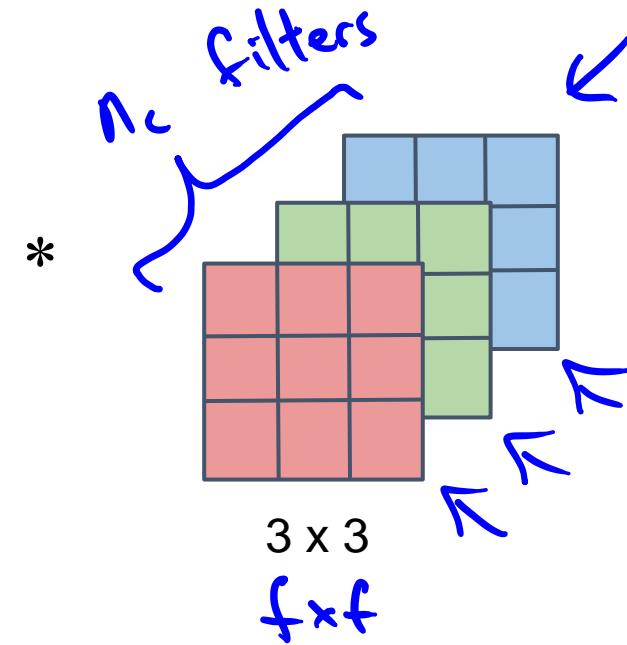


Depthwise Convolution



Computational cost

$$432$$



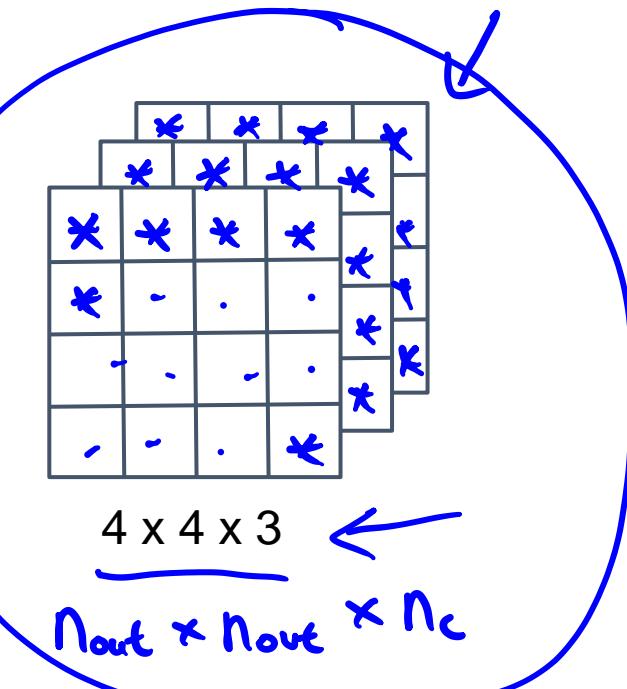
#filter params

$$\underbrace{3 \times 3}_{}$$

filter positions

$$\underbrace{4 \times 4}_{}$$

=

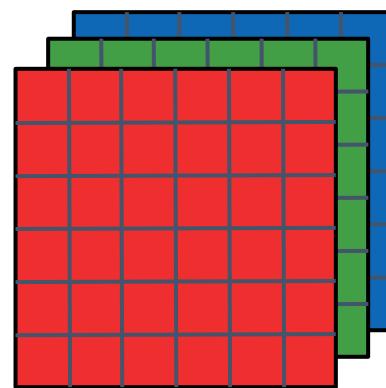


of filters

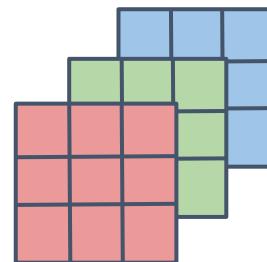
$$\underbrace{3}_{}$$

Depthwise Separable Convolution

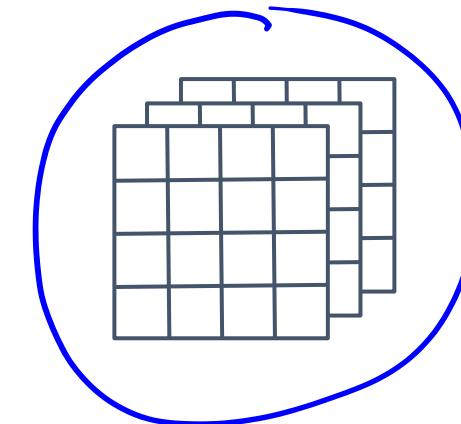
Depthwise Convolution



*

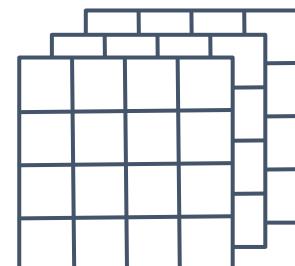


=



432

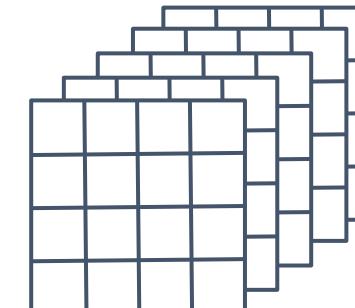
Pointwise Convolution



*

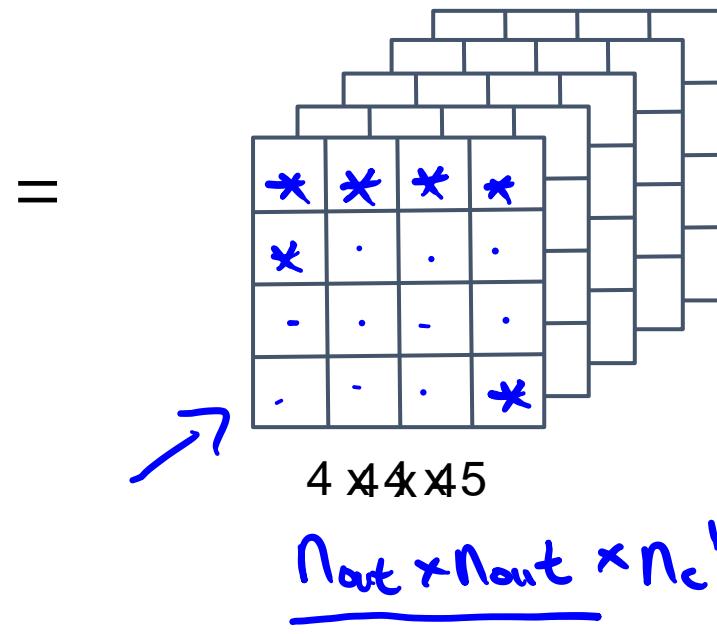
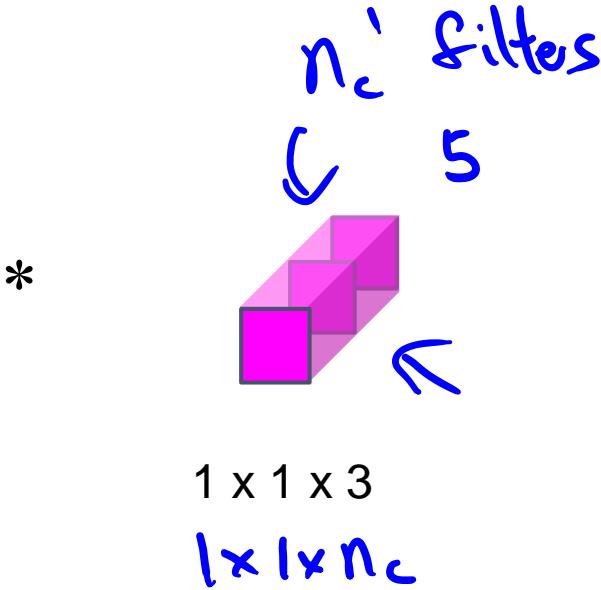
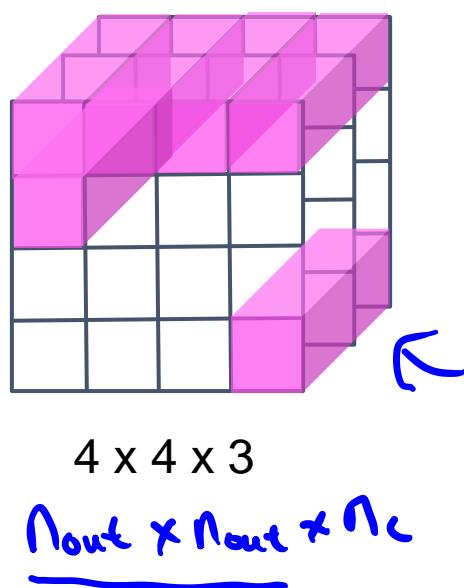


=



4x4x5

Pointwise Convolution

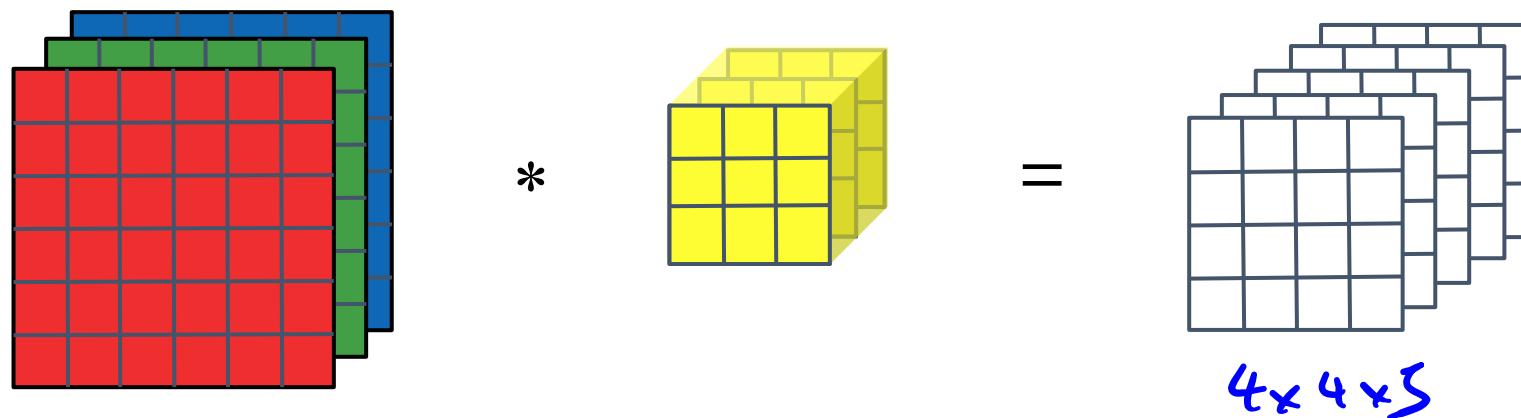


Computational cost = #filter params \times # filter positions \times # of filters

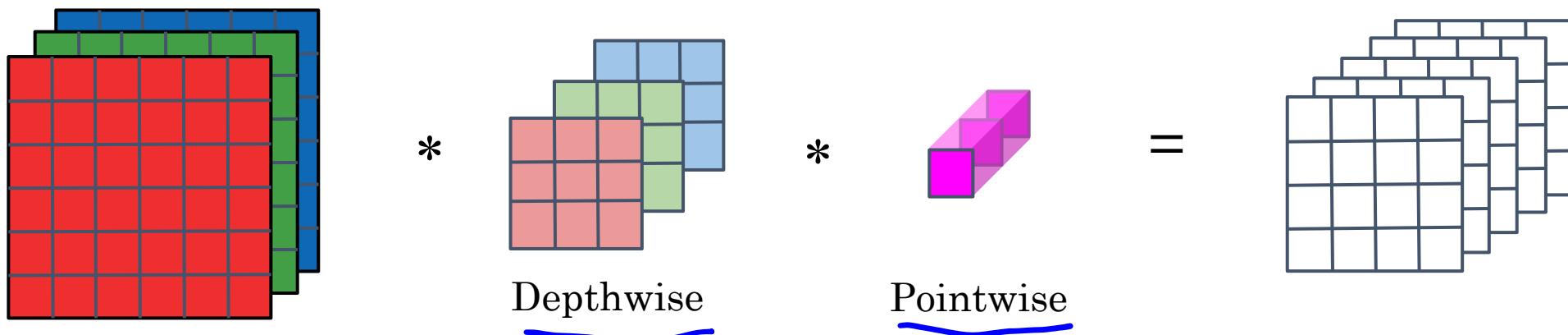
$$240 = 1 \times 1 \times 3 \times 4 \times 4 \times 5$$

Depthwise Separable Convolution

Normal Convolution



Depthwise Separable Convolution



Cost Summary

Cost of normal convolution

2160

Cost of depthwise separable convolution

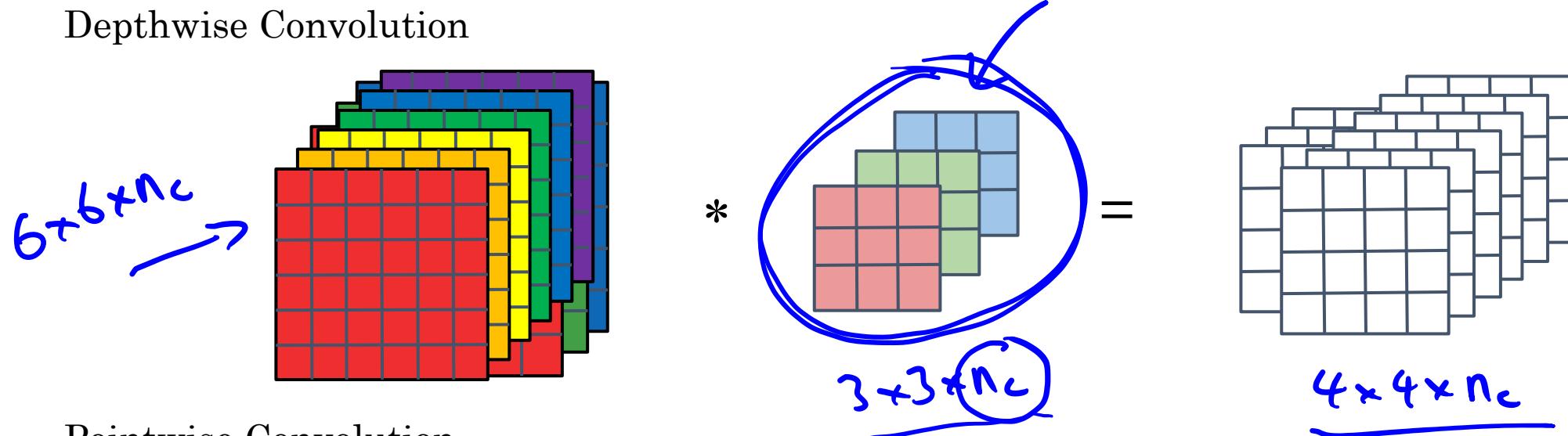
$$\begin{array}{l} \text{depthwise} + \text{pointwise} \\ 432 + 240 = 672 \end{array}$$

$$\frac{672}{2160} = 0.31 <$$

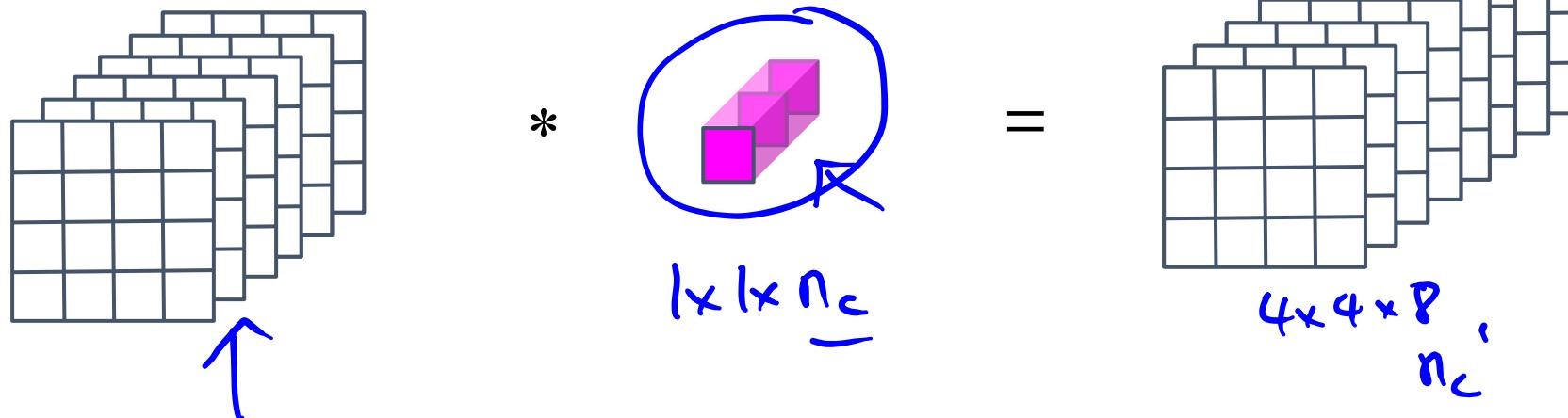
$$\begin{aligned} &= \frac{1}{n_c} + \frac{1}{f^2} \\ &\quad \frac{1}{s} + \frac{1}{q} \\ &= \frac{1}{512} + \frac{1}{3^2} \\ &\quad \nwarrow \quad \swarrow \\ &\text{n10 times cheaper} \end{aligned}$$

Depthwise Separable Convolution

Depthwise Convolution



Pointwise Convolution





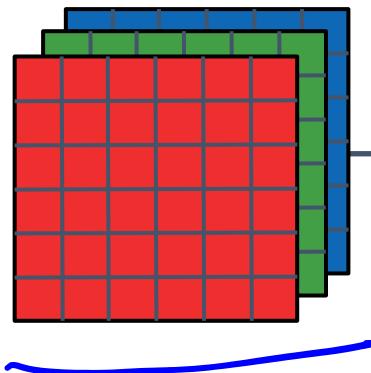
deeplearning.ai

Convolutional Neural Networks

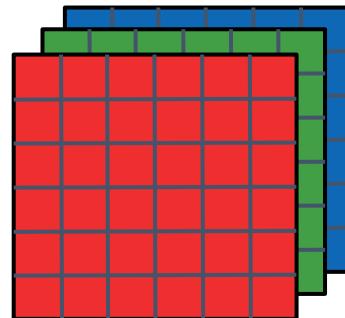
MobileNet Architecture

MobileNet

MobileNet v1



MobileNet v2



13 times

POOL, FC, SOFTMAX

17 times

Residual Connection

POOL, FC,
SOFTMAX

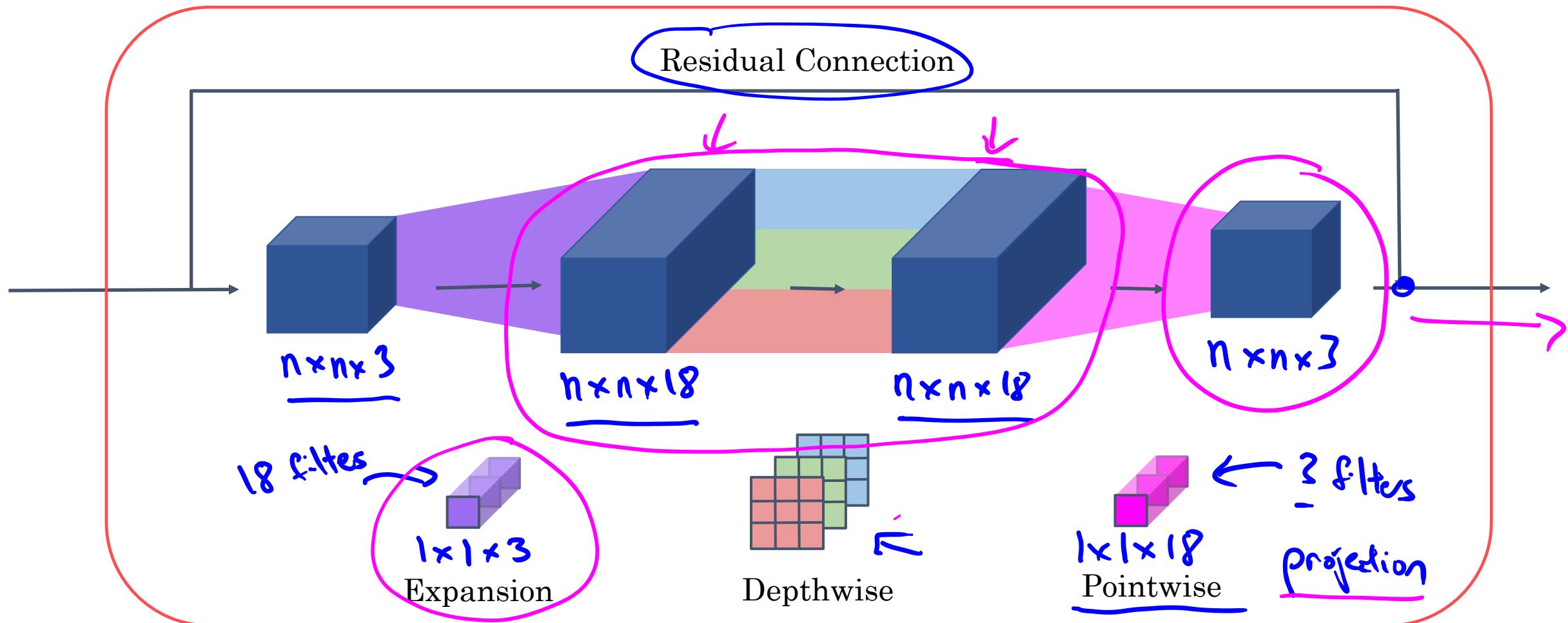
Expansion

Depthwise

Projection

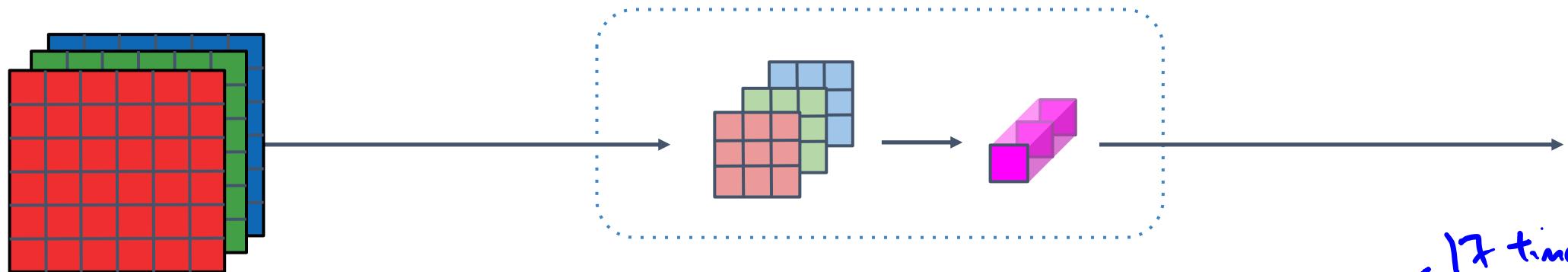
Bottleneck

MobileNet v2 Bottleneck

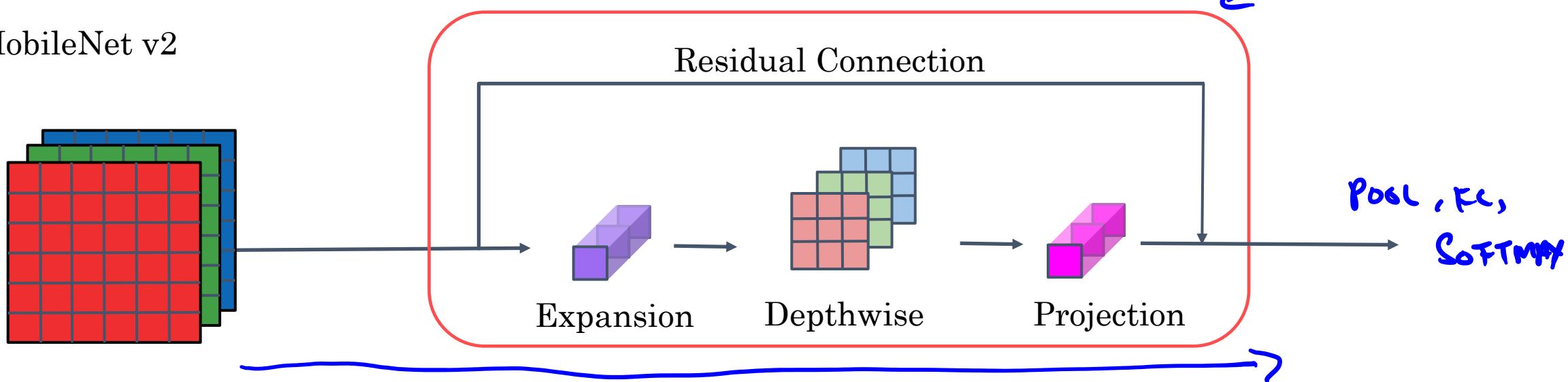


MobileNet

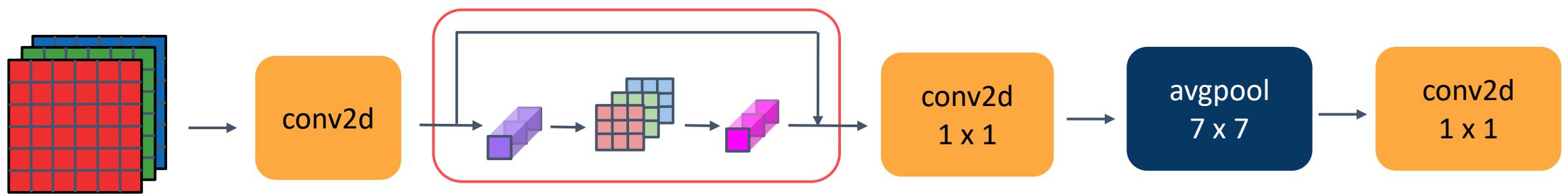
MobileNet v1



MobileNet v2



MobileNet v2 Full Architecture



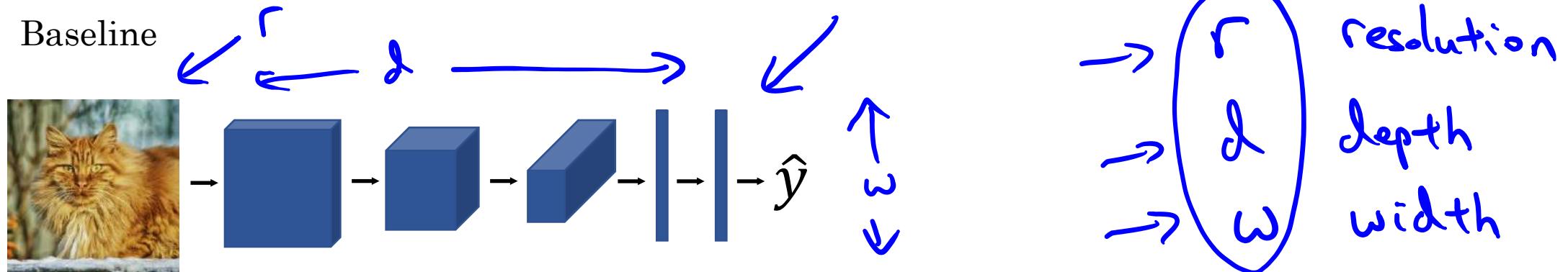


deeplearning.ai

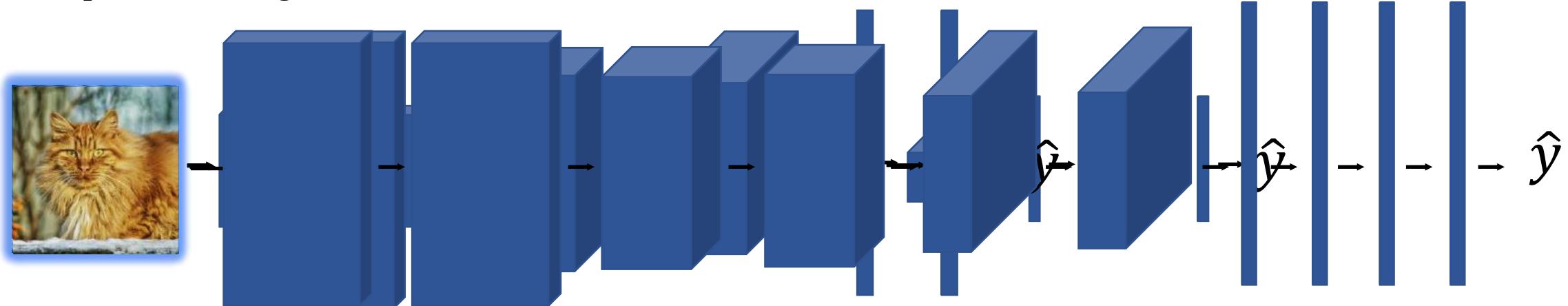
Convolutional Neural Networks

EfficientNet

EfficientNet



Without Rescaling



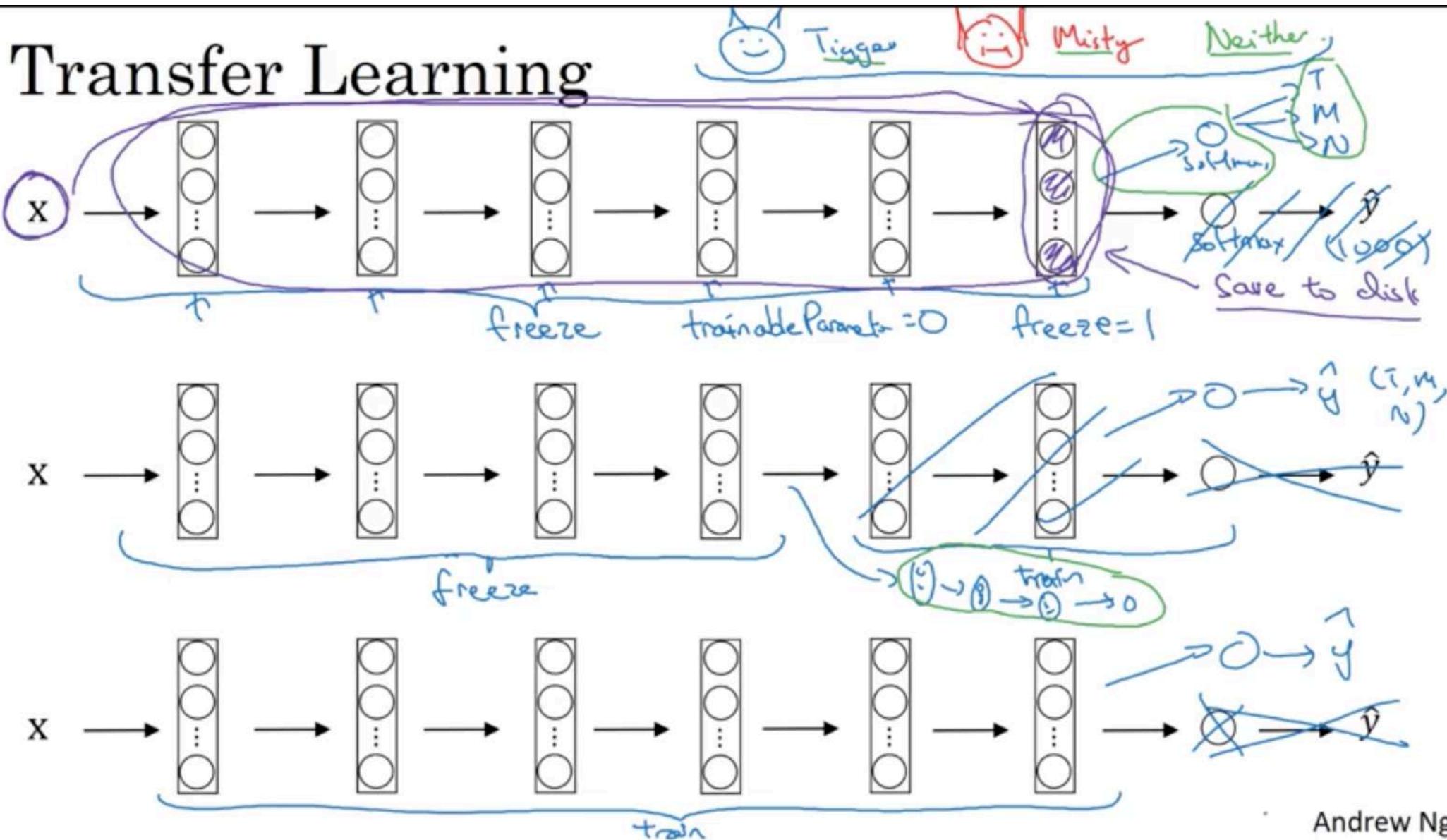


deeplearning.ai

Practical advice for
using ConvNets

Transfer Learning

Transfer Learning



Andrew Ng



deeplearning.ai

Practical advice for
using ConvNets

Data augmentation

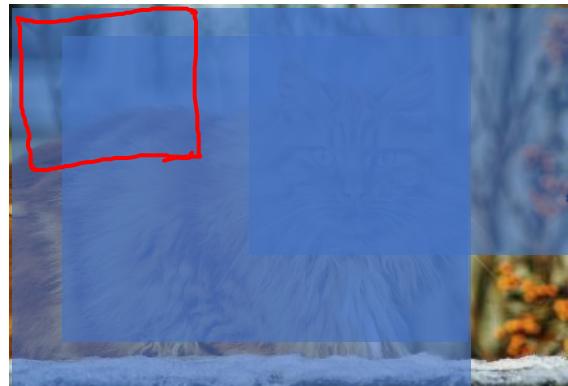
Common augmentation method

Mirroring



yc

Random Cropping

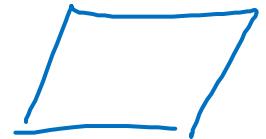


Rotation

Shearing

Local warping

...



Color shifting



R G B
↓ ↓ ↓
+20,-20,+20



-20,+20,+20



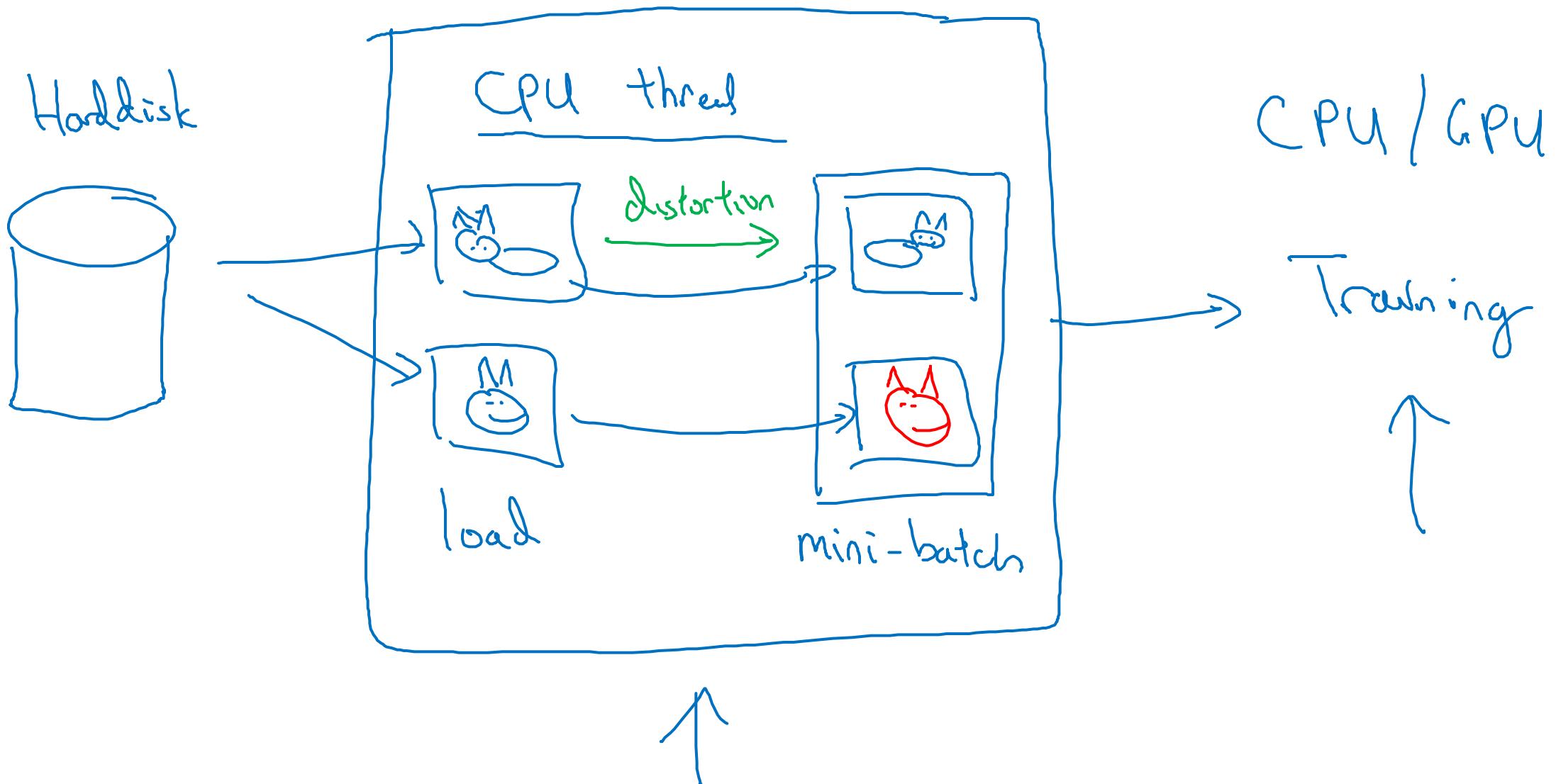
+5,0,+50



y

Advanced:
PCA
ml-class.org
[AlexNet paper
["PCA color augmentation."
R B G

Implementing distortions during training



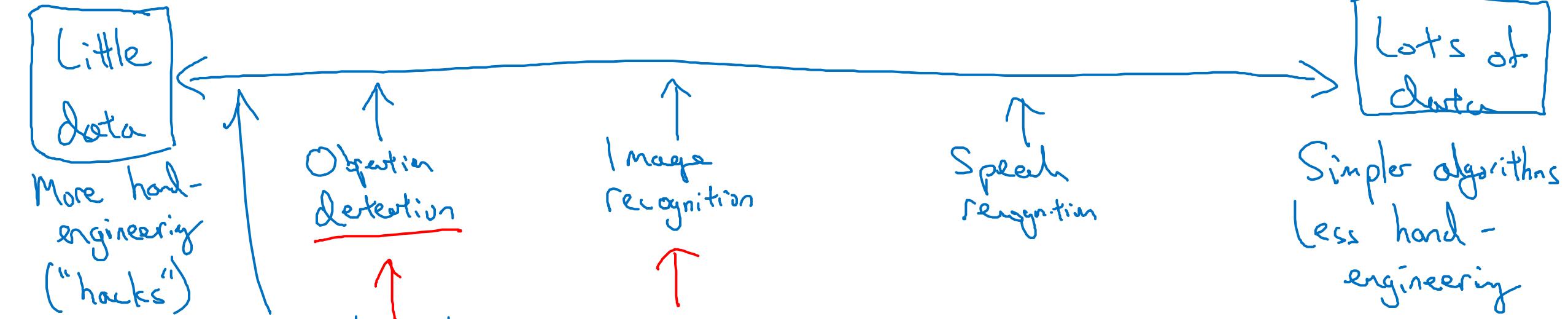


deeplearning.ai

Practical advice for
using ConvNets

The state of
computer vision

Data vs. hand-engineering



Two sources of knowledge

- • Labeled data (x, y)
- • Hand engineered features/network architecture/other components



Tips for doing well on benchmarks/winning competitions

Ensembling

3 - 15 networks

→ \hat{y}

- Train several networks independently and average their outputs

Multi-crop at test time

- Run classifier on multiple versions of test images and average results

10-crop



1



+

4



1

+

4



Andrew Ng

Use open source code

- Use architectures of networks published in the literature
- Use open source implementations if possible
- Use pretrained models and fine-tune on your dataset

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

Object Detection

Object localization

What are localization and detection?

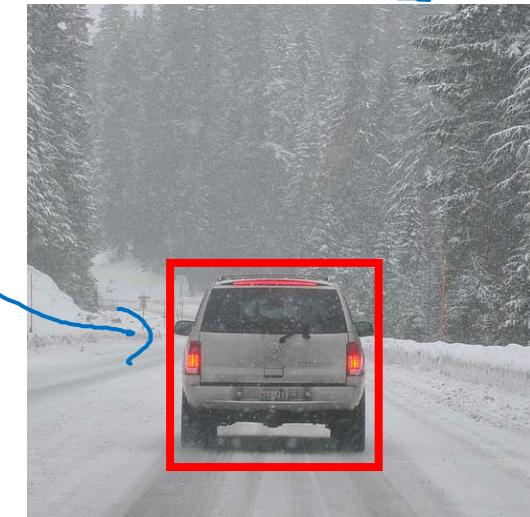
Image classification



"Car"

1 object

Classification with
localization



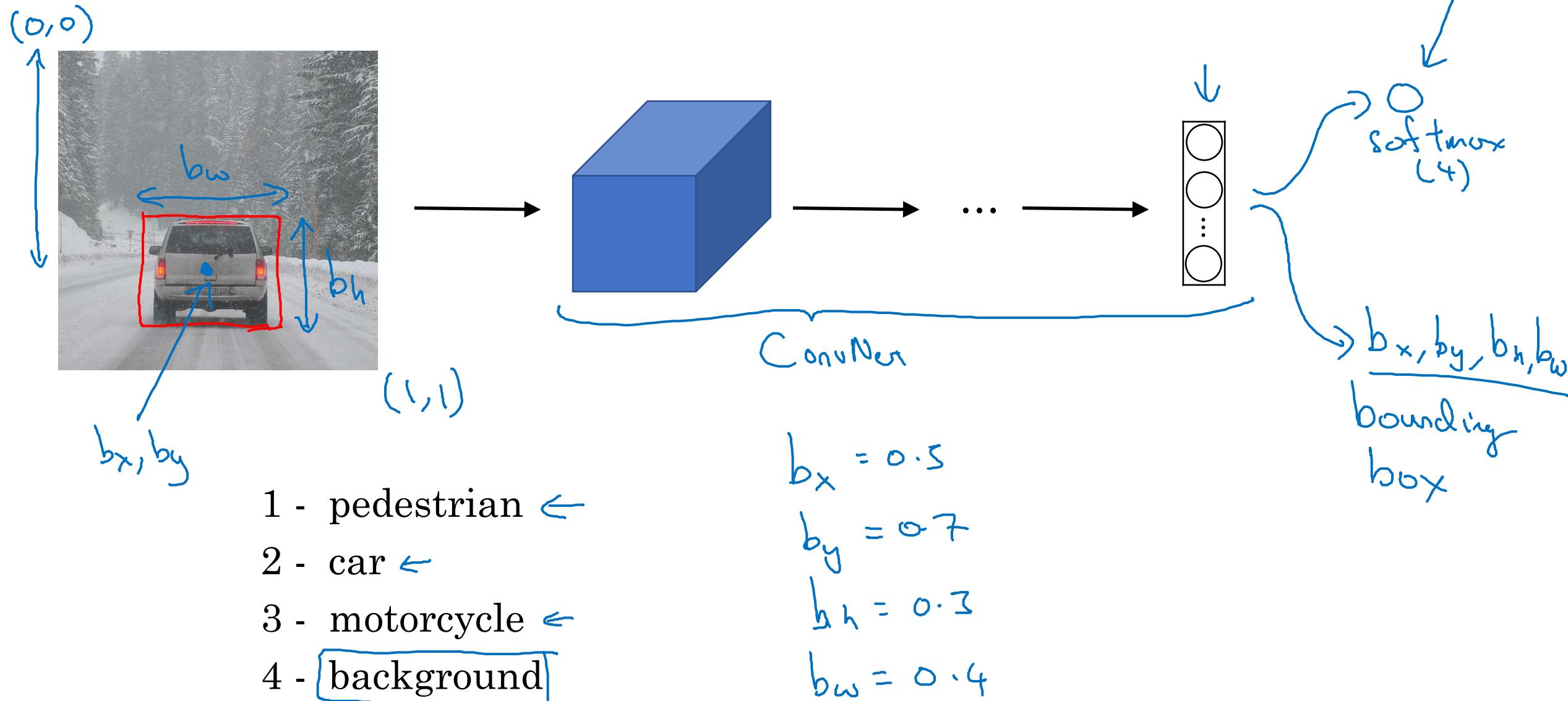
"Car"

Detection



multiple
objects

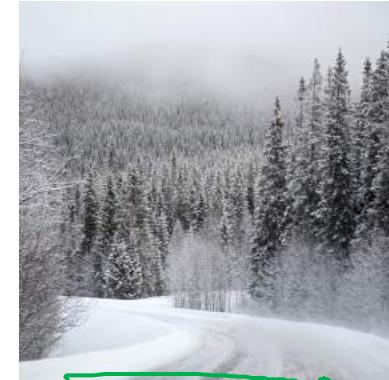
Classification with localization



Defining the target label y

- 1 - pedestrian
- 2 - car ←
- 3 - motorcycle
- 4 - background ←

Need to output b_x, b_y, b_h, b_w , class label (1-4)

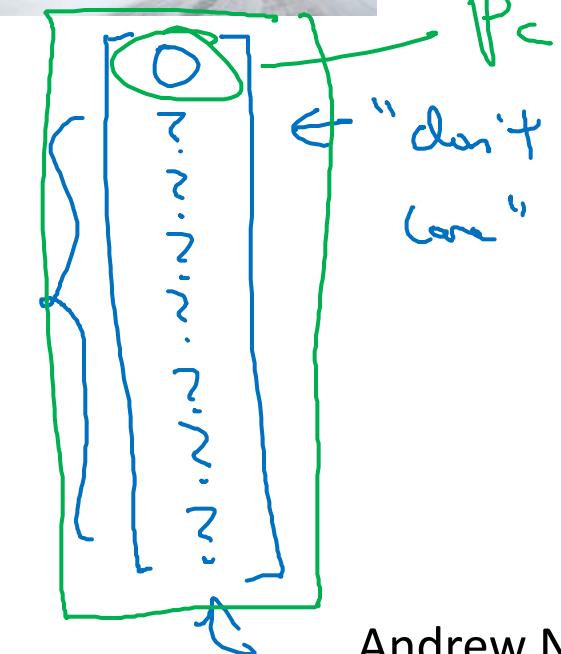
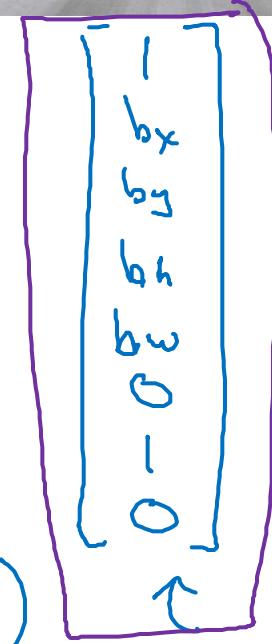


$x =$

$$L(\hat{y}, y) = \begin{cases} (\hat{y}_1 - y_1)^2 + (\hat{y}_2 - y_2)^2 \\ + \dots + (\hat{y}_8 - y_8)^2 & \text{if } y_1 = 1 \\ (\hat{y}_1 - y_1)^2 & \text{if } y_1 = 0 \end{cases}$$

$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \rightarrow \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \quad \left\{ \begin{array}{l} \text{is there any} \\ \text{object?} \end{array} \right.$$

(x, y)



Andrew Ng

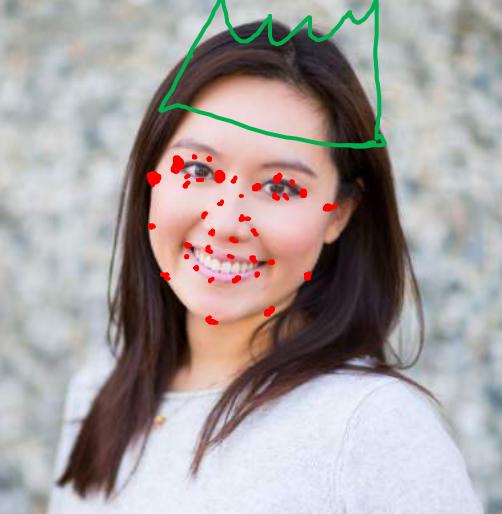
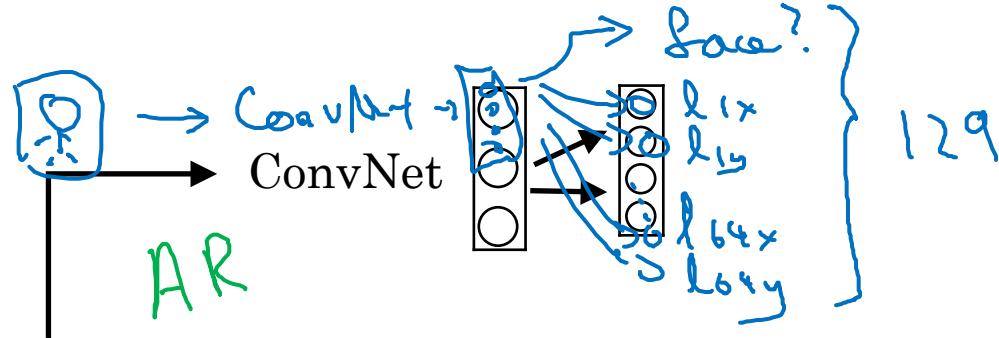


deeplearning.ai

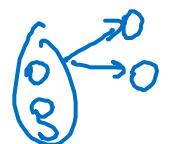
Object Detection

Landmark detection

Landmark detection



b_x, b_y, b_h, b_w



$l_{1x}, l_{1y},$
 $l_{2x}, l_{2y},$
 $l_{3x}, l_{3y},$
 $l_{4x}, l_{4y},$
:
 l_{64x}, l_{64y}

x, y

$l_{1x}, l_{1y},$
:
 l_{32x}, l_{32y}



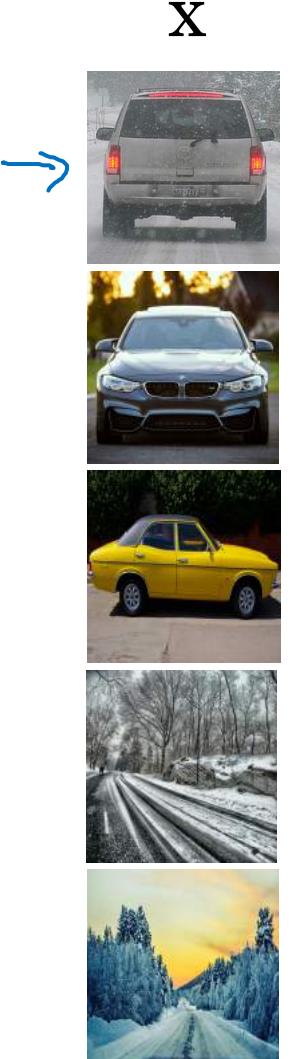
deeplearning.ai

Object Detection

Object detection

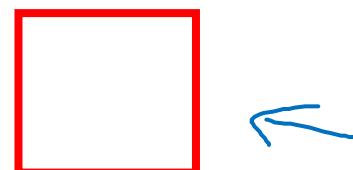
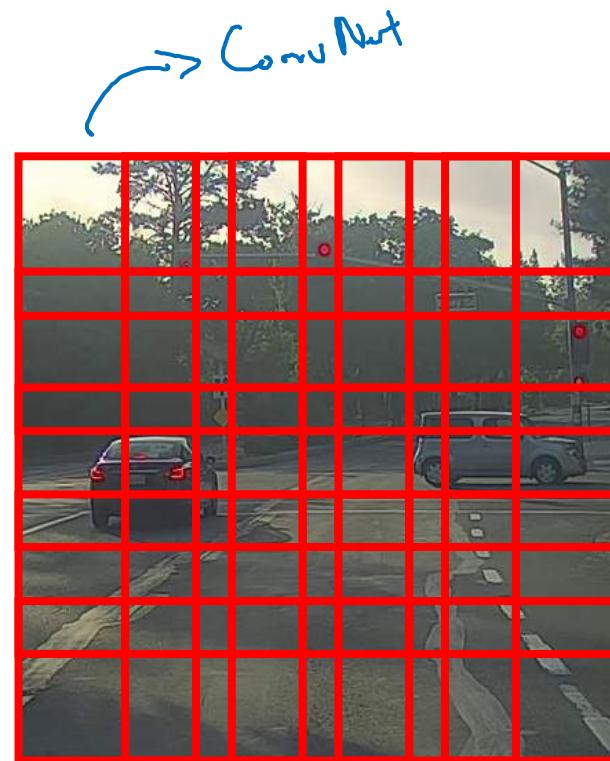
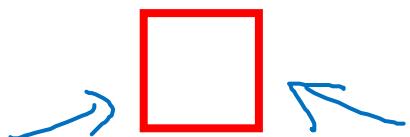
Car detection example

Training set:

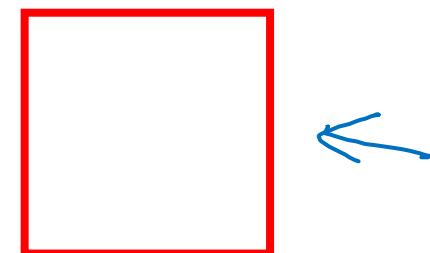


$\xrightarrow{\hspace{1cm}}$ ConvNet $\rightarrow y$

Sliding windows detection



Computation cost



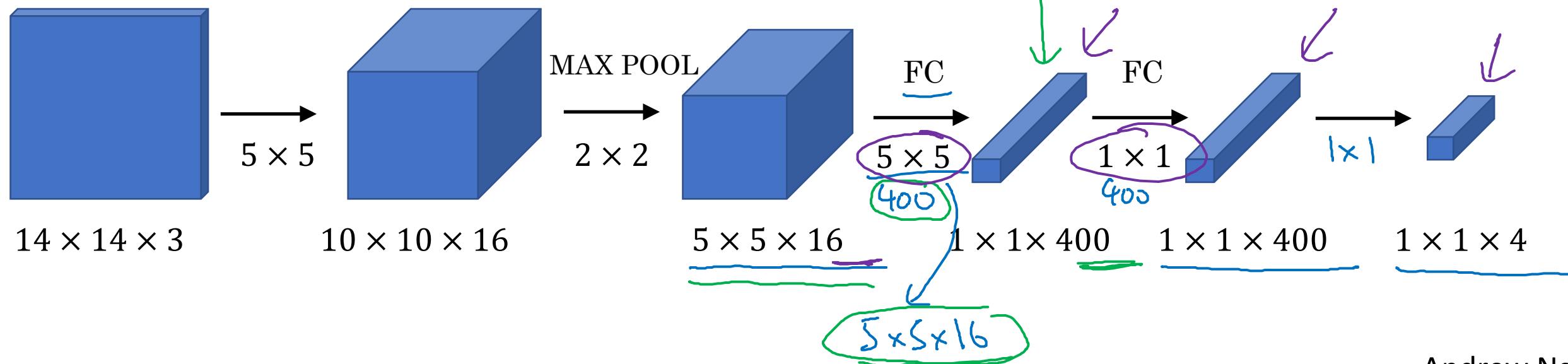
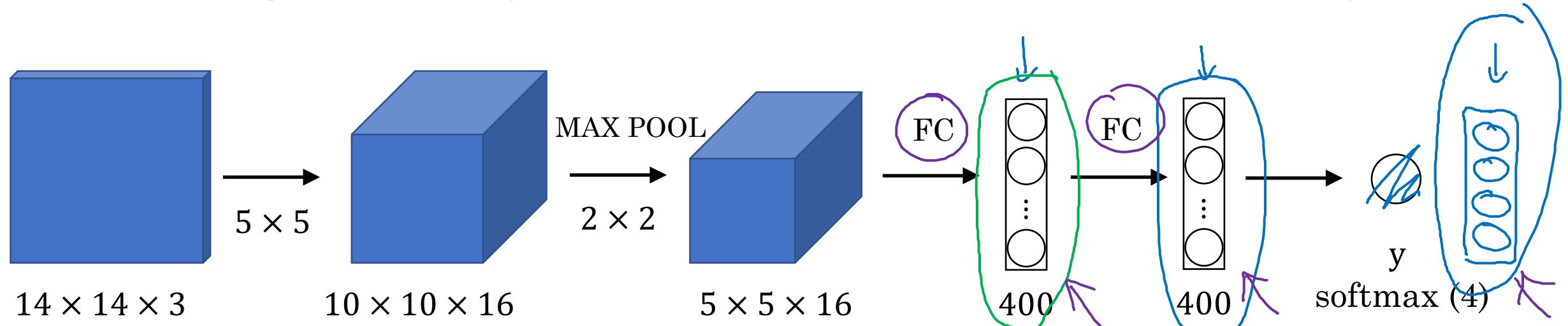


deeplearning.ai

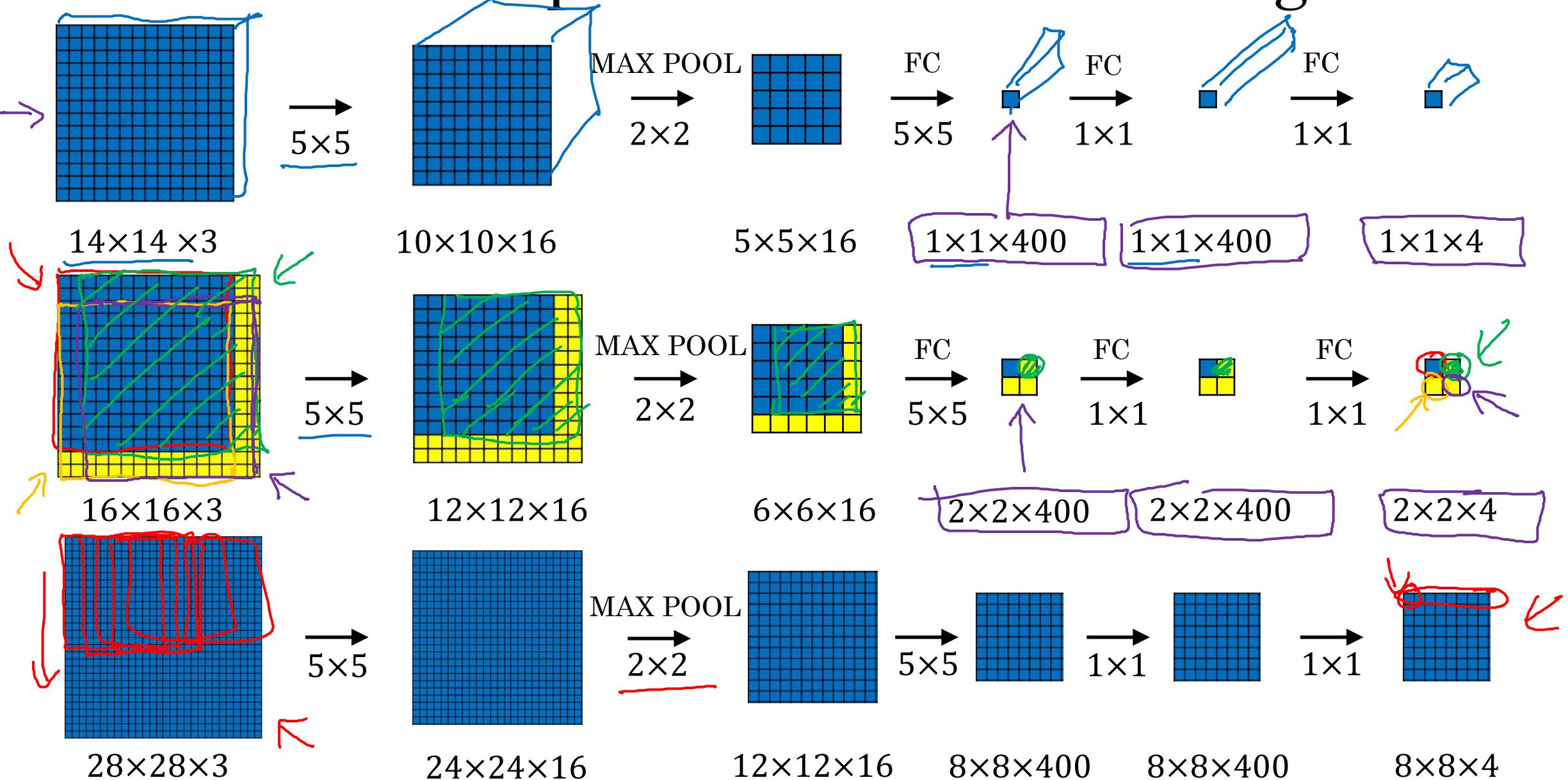
Object Detection

Convolutional implementation of sliding windows

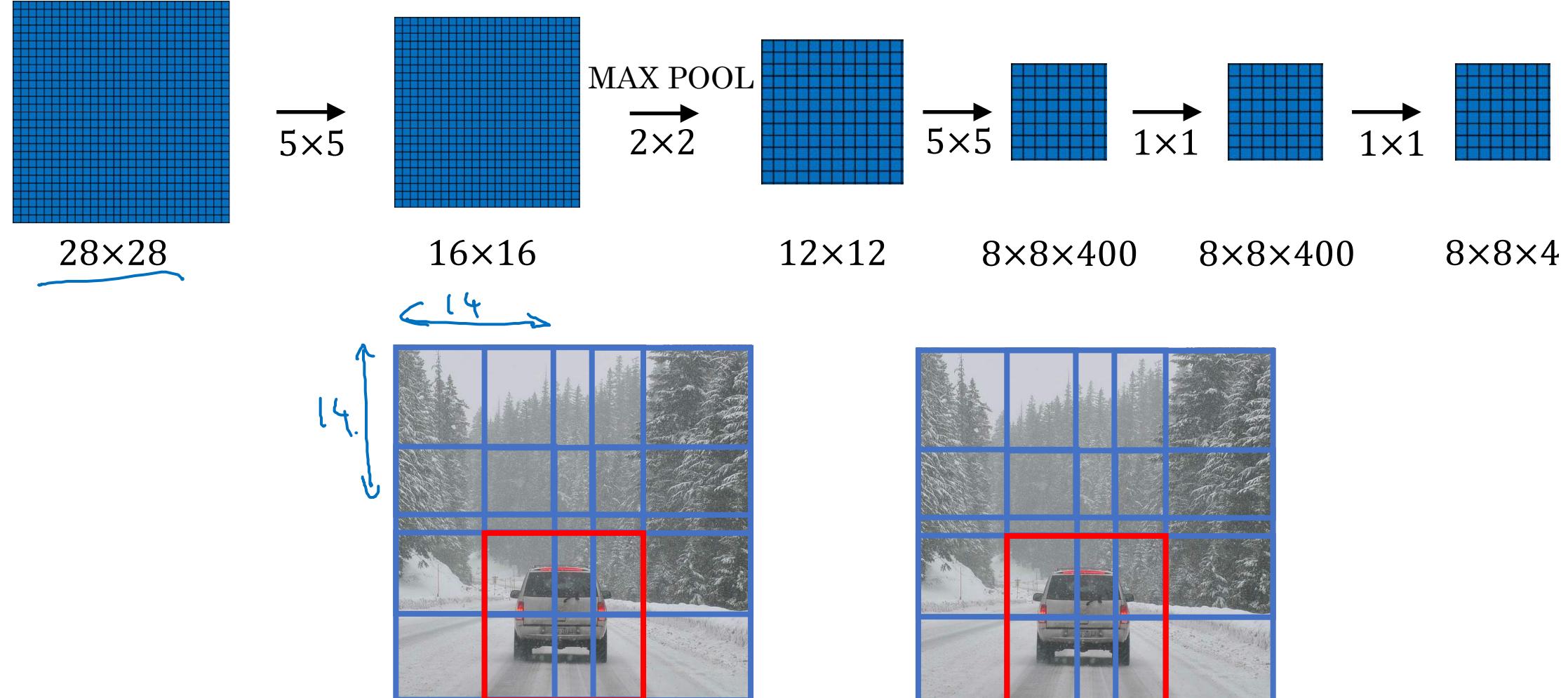
Turning FC layer into convolutional layers



Convolution implementation of sliding windows



Convolution implementation of sliding windows



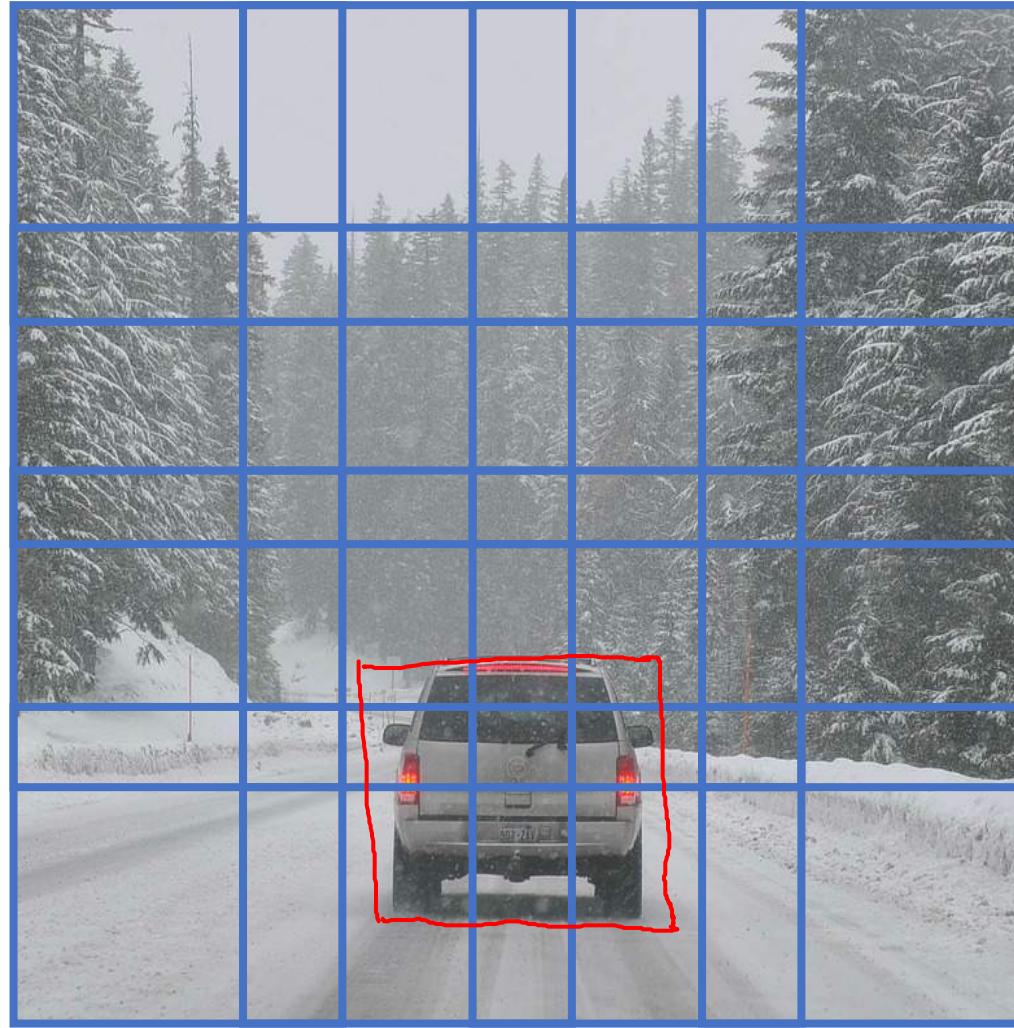


deeplearning.ai

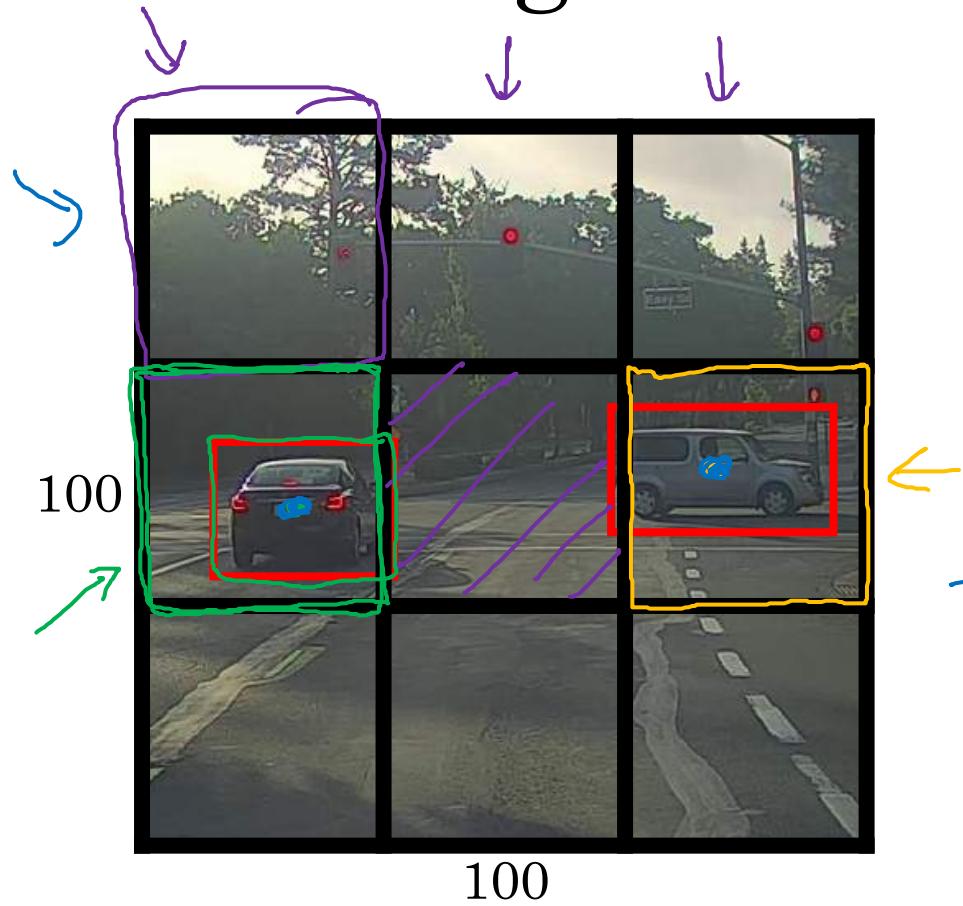
Object Detection

Bounding box
predictions

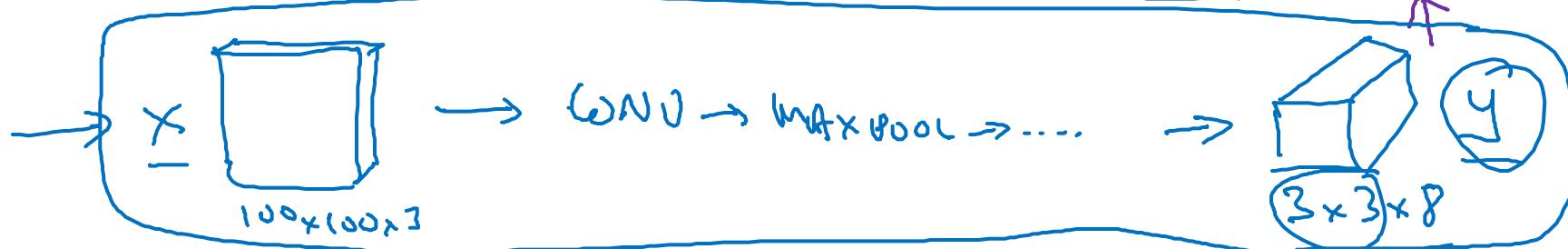
Output accurate bounding boxes



YOLO algorithm



100



Labels for training

For each grid cell:

Target output:

$$3 \times 3 \times 8$$

$y =$

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

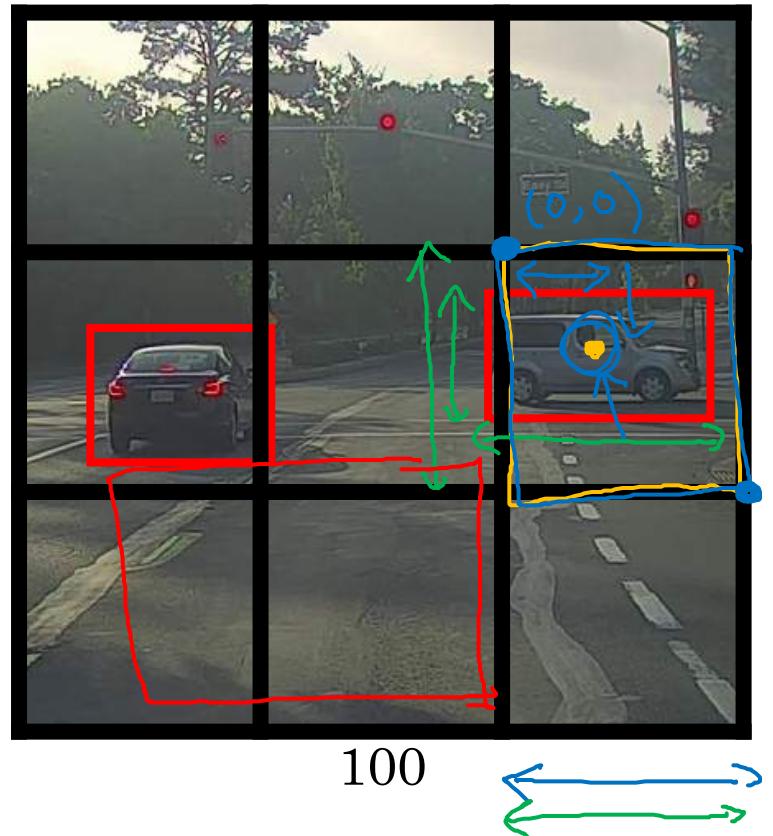
$$\begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 0 \\ 1 \end{bmatrix}$$

Andrew Ng

Specify the bounding boxes



$$y = \begin{bmatrix} b_x \\ b_y \\ b_h \\ b_w \\ o \end{bmatrix}$$

0.4 } between 0 and 1
0.3 }
0.9 }
0.5 } Could be > 1

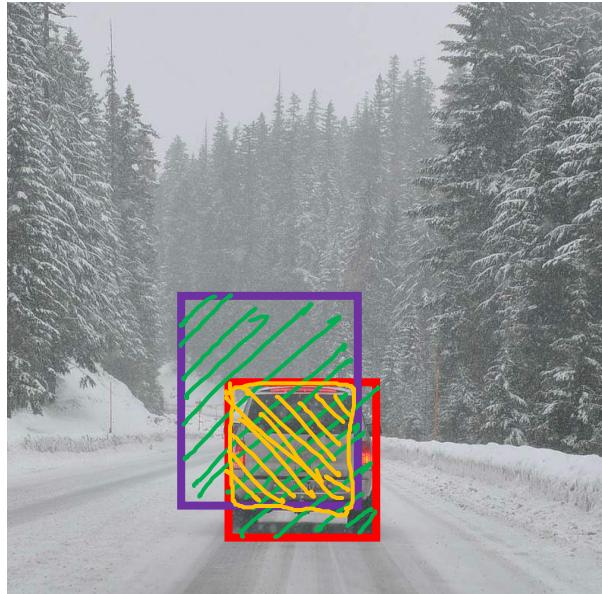


deeplearning.ai

Object Detection

Intersection
over union

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of intersection}}{\text{Size of union}}$$
A diagram illustrating the calculation of IoU. It shows two overlapping rectangles: one yellow and one green. The overlapping area is shaded with diagonal lines, representing the intersection. The total area covered by either rectangle or both is shaded with horizontal lines, representing the union.

“Correct” if $\text{IoU} \geq 0.5$

0.6

More generally, IoU is a measure of the overlap between two bounding boxes.



deeplearning.ai

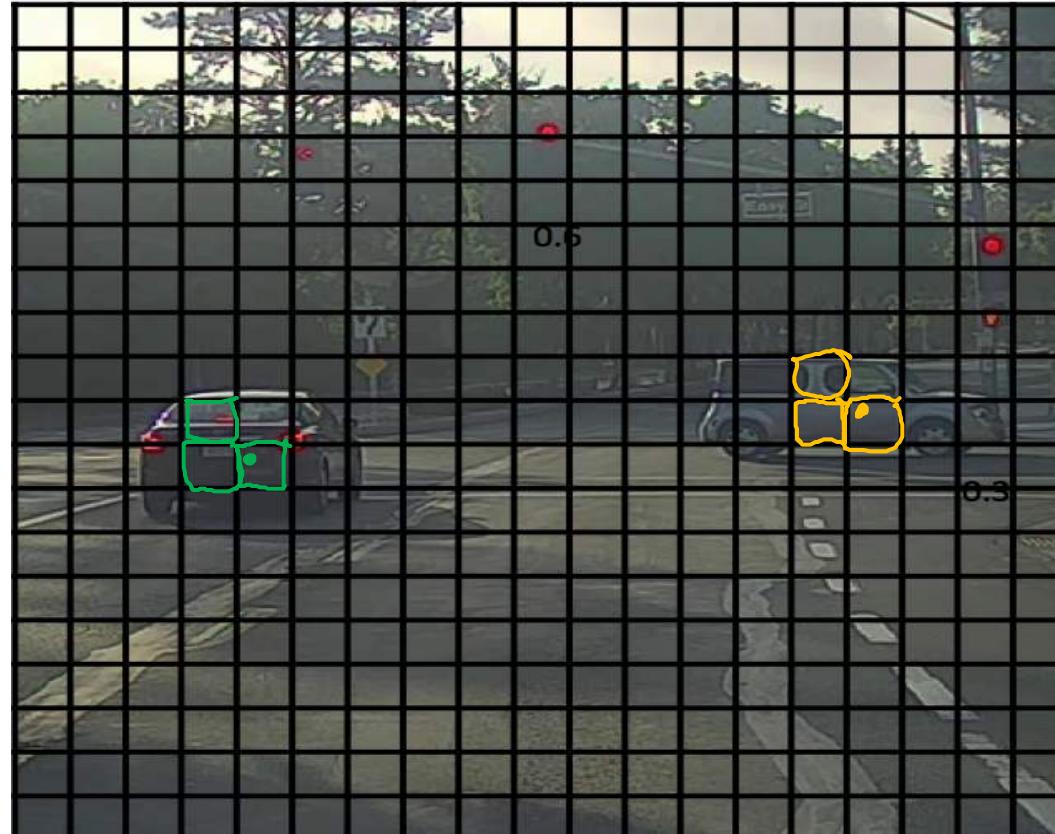
Object Detection

Non-max suppression

Non-max suppression example

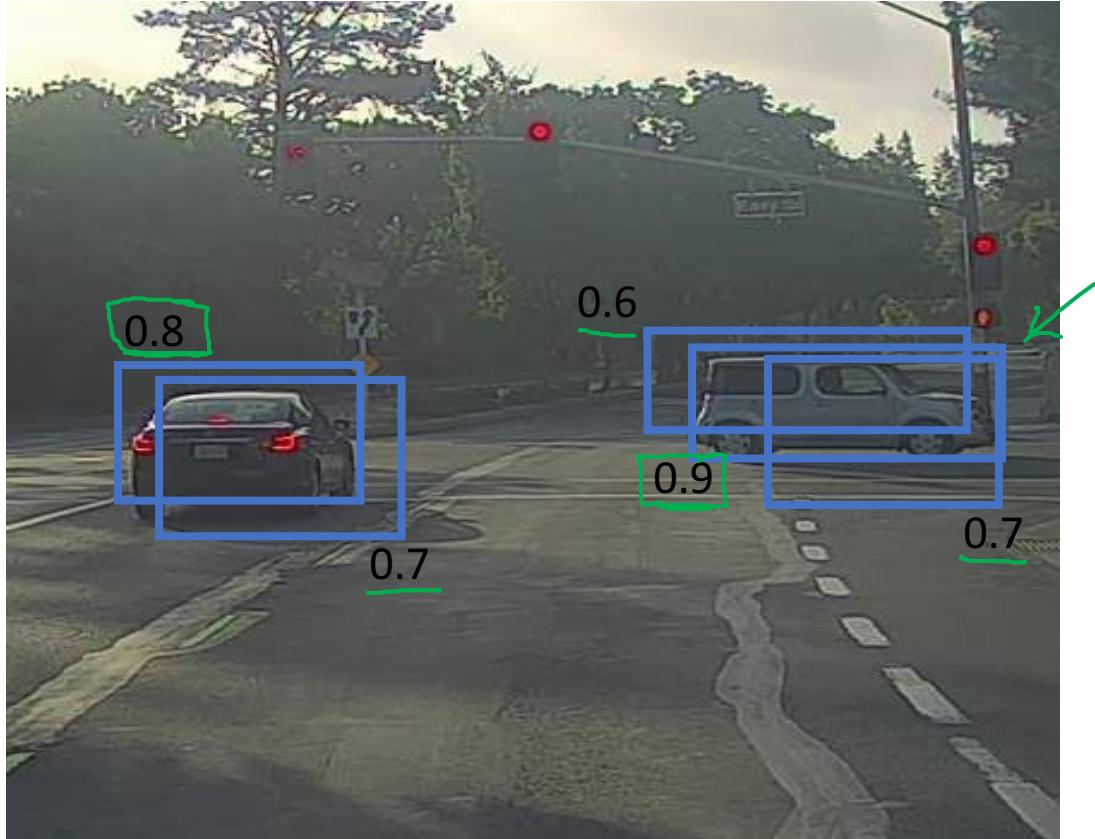


Non-max suppression example

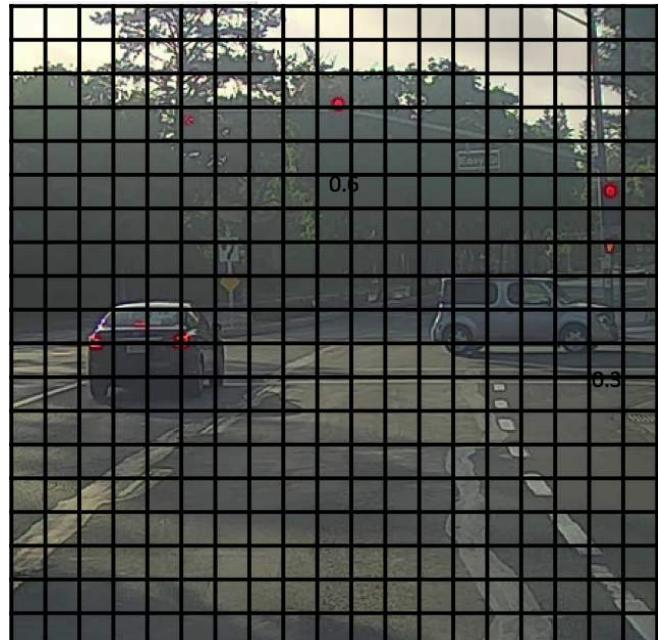


19x19

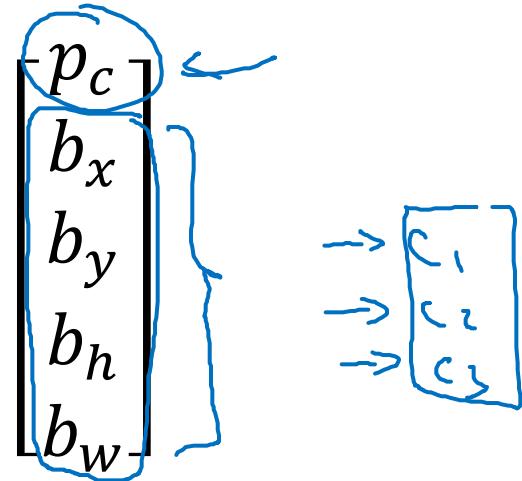
Non-max suppression example



Non-max suppression algorithm



Each output prediction is:



Discard all boxes with $p_c \leq 0.6$

→ While there are any remaining boxes:

- Pick the box with the largest p_c . Output that as a prediction.
- Discard any remaining box with $\text{IoU} \geq 0.5$ with the box output in the previous step

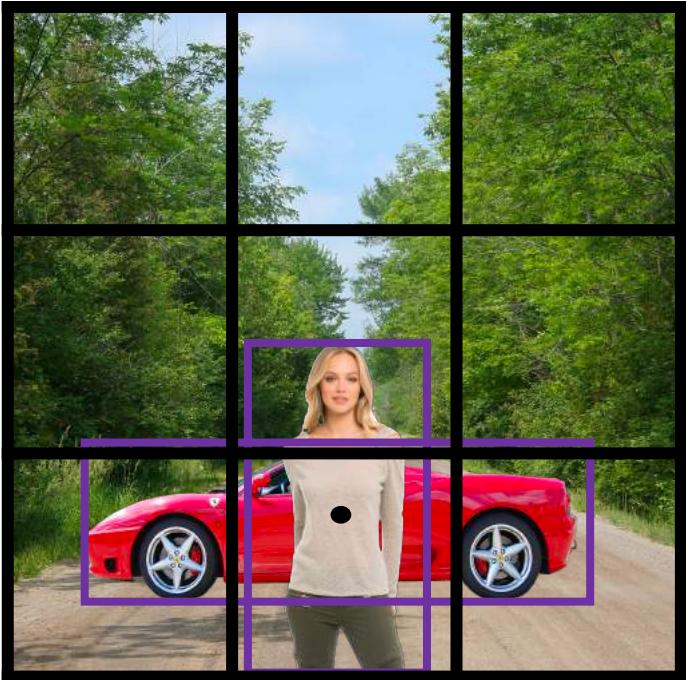


deeplearning.ai

Object Detection

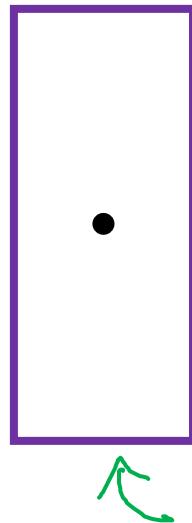
Anchor boxes

Overlapping objects:

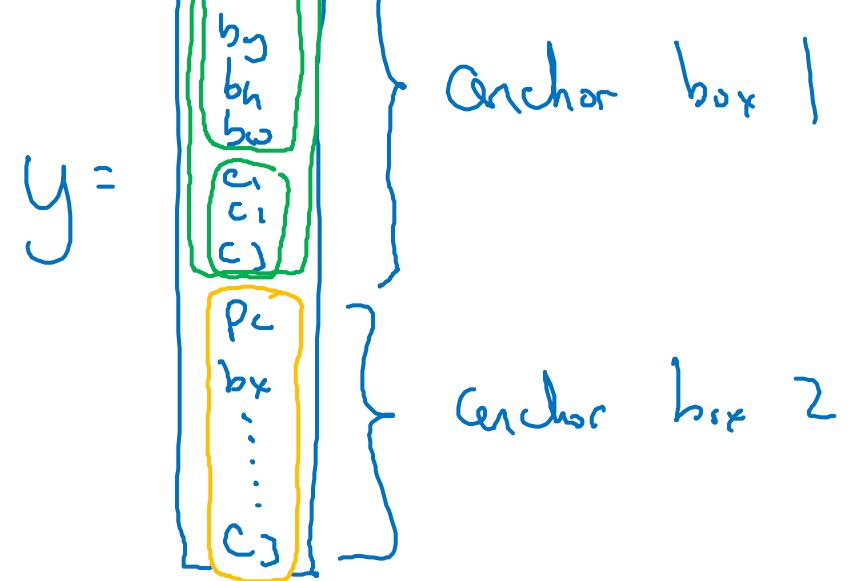
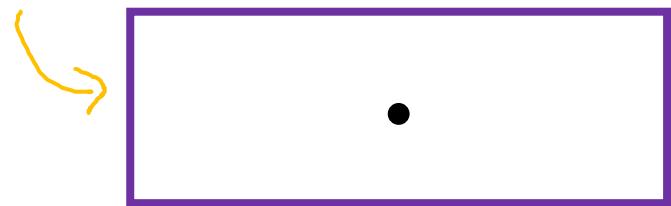


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



Anchor box 2:



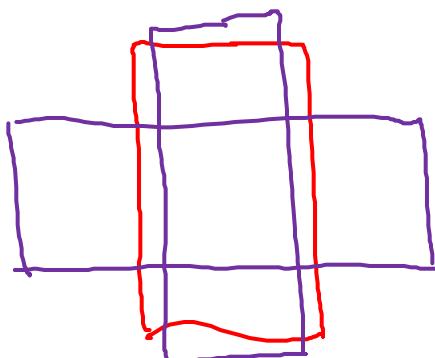
Anchor box algorithm

Previously:

Each object in training image is assigned to grid cell that contains that object's midpoint.

Output y:

$3 \times 3 \times 8$



With two anchor boxes:

Each object in training image is assigned to grid cell that contains object's midpoint and anchor box for the grid cell with highest IoU.

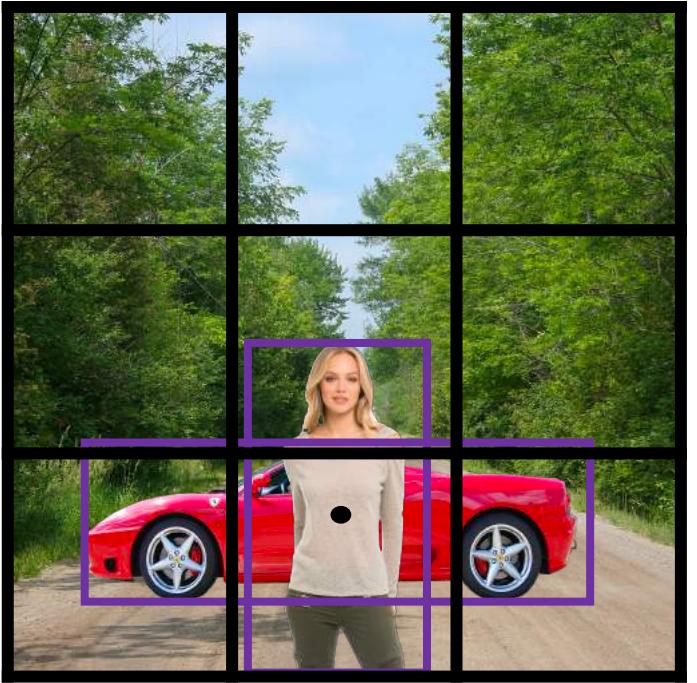
(grid cell, anchor box)

Output y:

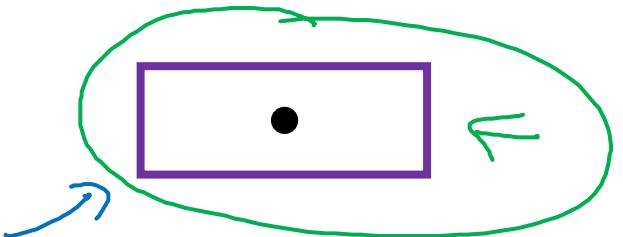
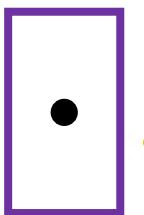
$3 \times 3 \times 16$

$3 \times 3 \times 2 \times 8$

Anchor box example



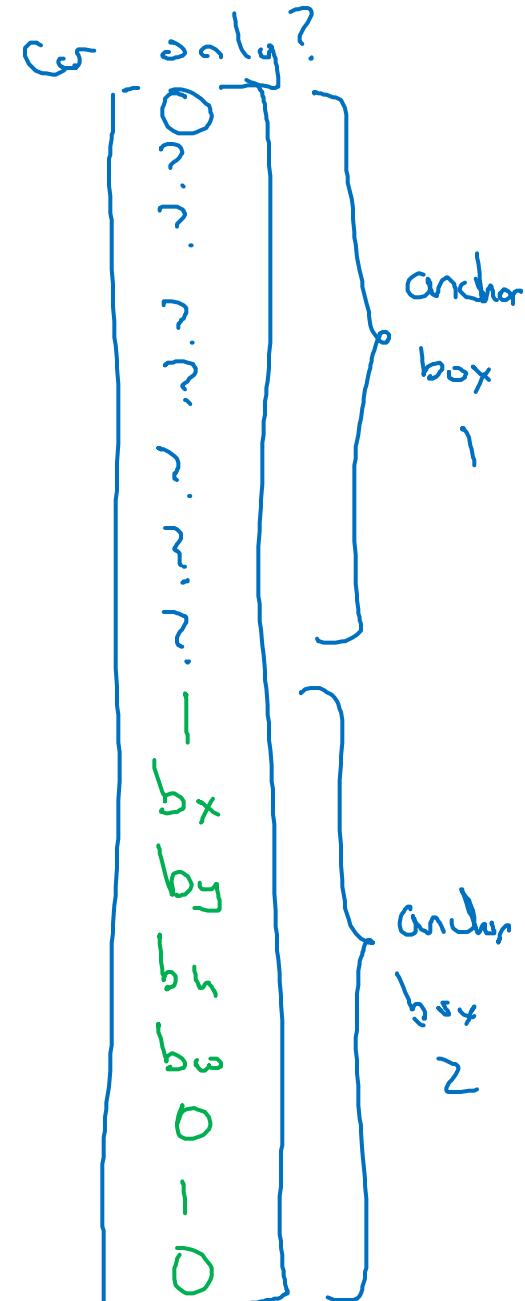
Anchor box 1: Anchor box 2:



$$y =$$

$$\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

$$\begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ - \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$



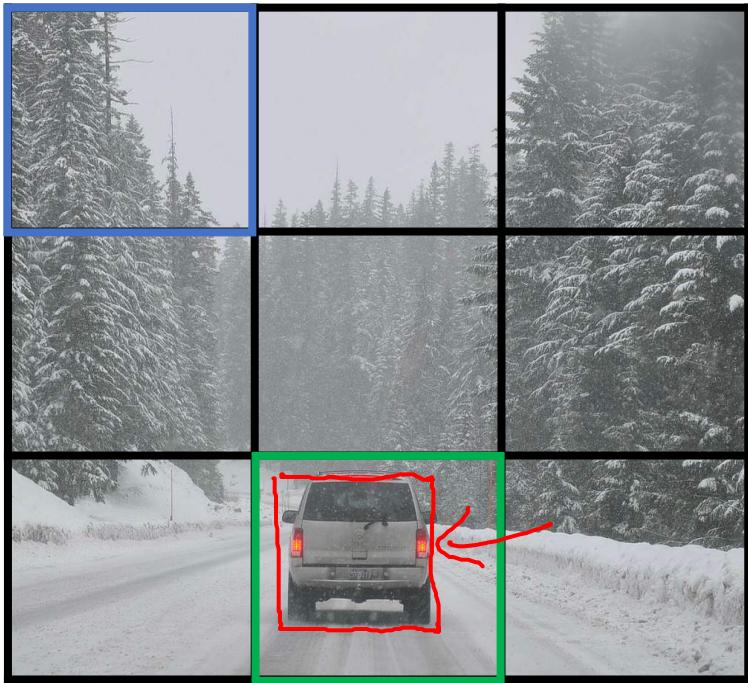


deeplearning.ai

Object Detection

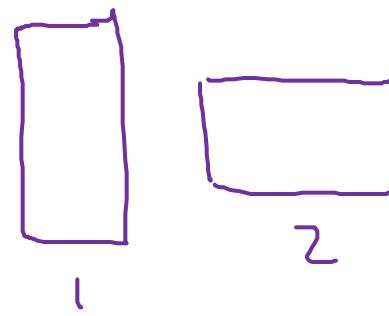
Putting it together:
YOLO algorithm

Training



- 1 - pedestrian
- 2 - car ←
- 3 - motorcycle

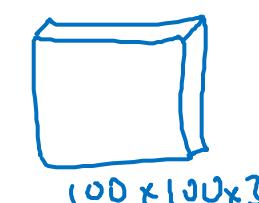
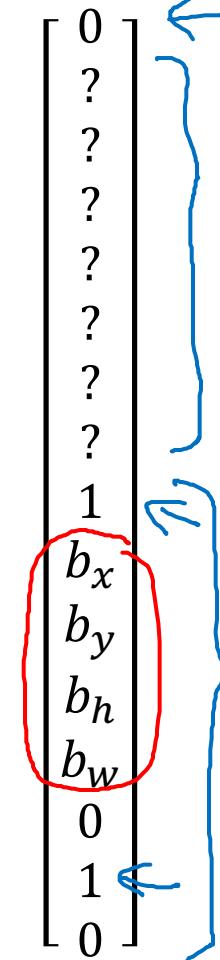
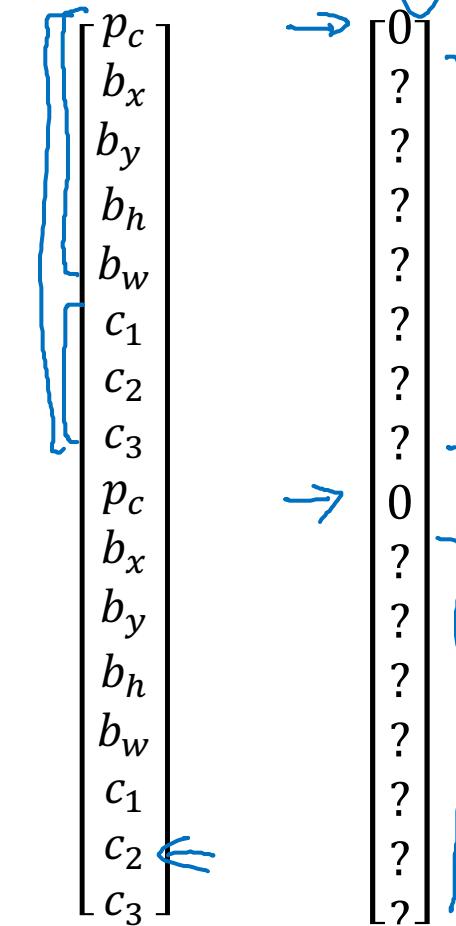
$y =$



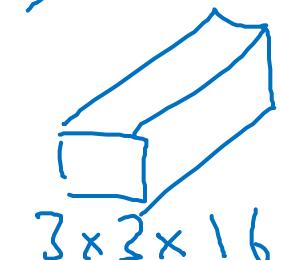
y is $3 \times 3 \times 2 \times 8$

$19 \times 19 \times 16$
 $19 \times 19 \times 40$

#anchors → $5 + \#classes$

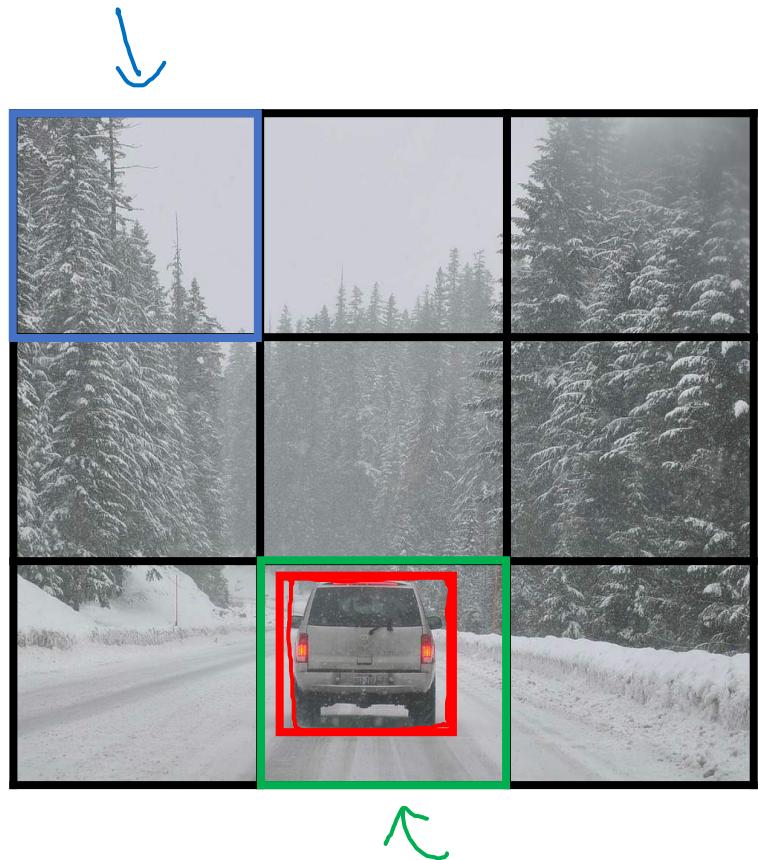


→ ConvNet



Andrew Ng

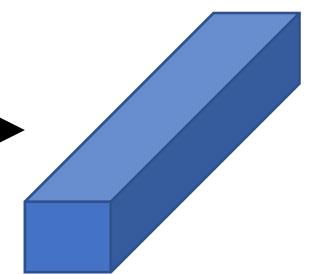
Making predictions



→

...

→



$y =$

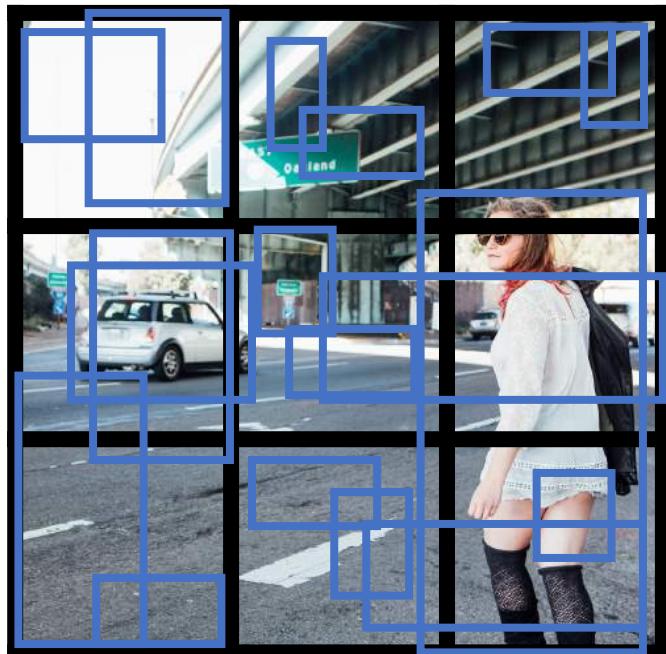
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Diagram illustrating the output vector y for object detection. The vector is composed of 16 elements, grouped into three main sections: confidence scores (p_c), bounding box coordinates (b_x, b_y, b_h, b_w), and class probabilities (c_1, c_2, c_3). Handwritten annotations in blue and red highlight specific elements:

- Blue annotations: p_c (top element), b_x , b_y , b_h , b_w , c_1 , c_2 , c_3 (bottom section).
- Red annotations: b_x , b_y , b_h , b_w (middle section).

Arrows point from the handwritten labels to their corresponding positions in the vector. A blue arrow also points to the bottom of the vector, indicating the continuation of the sequence.

Outputting the non-max suppressed outputs



- For each grid call, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

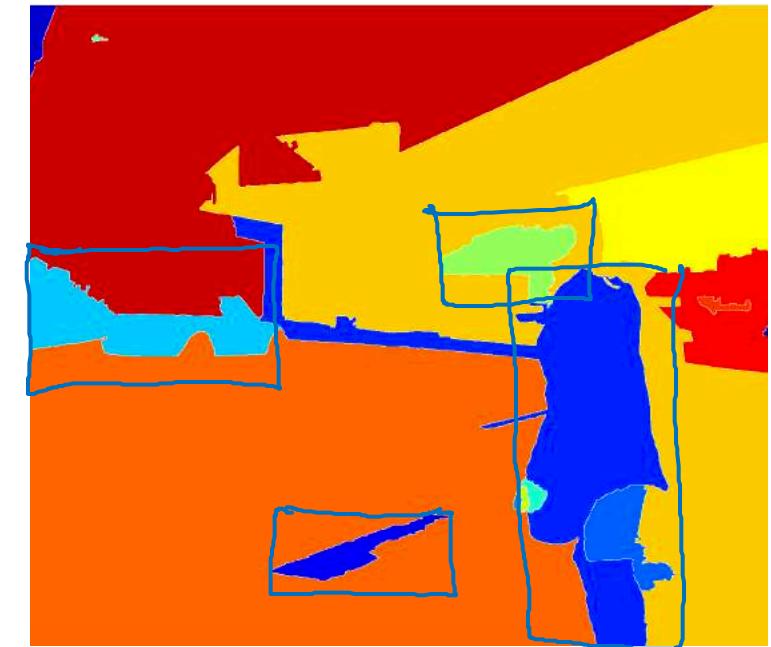
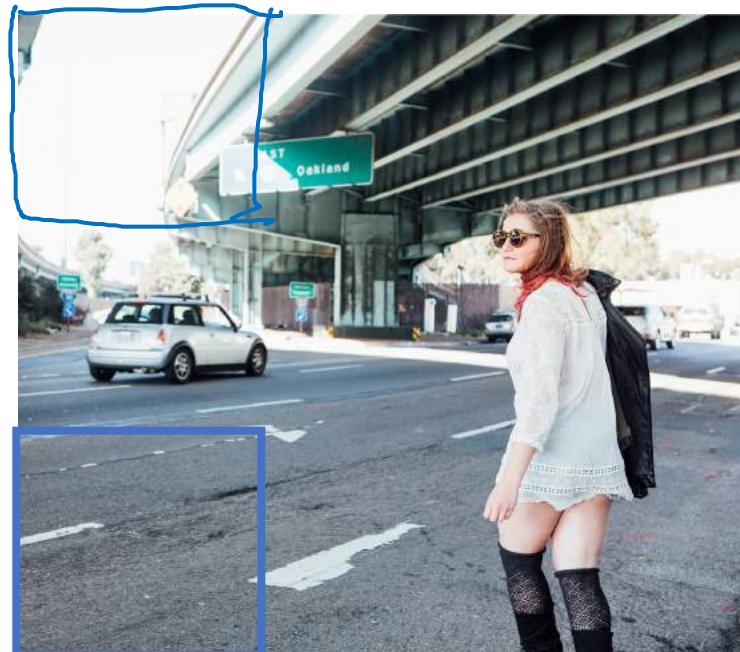
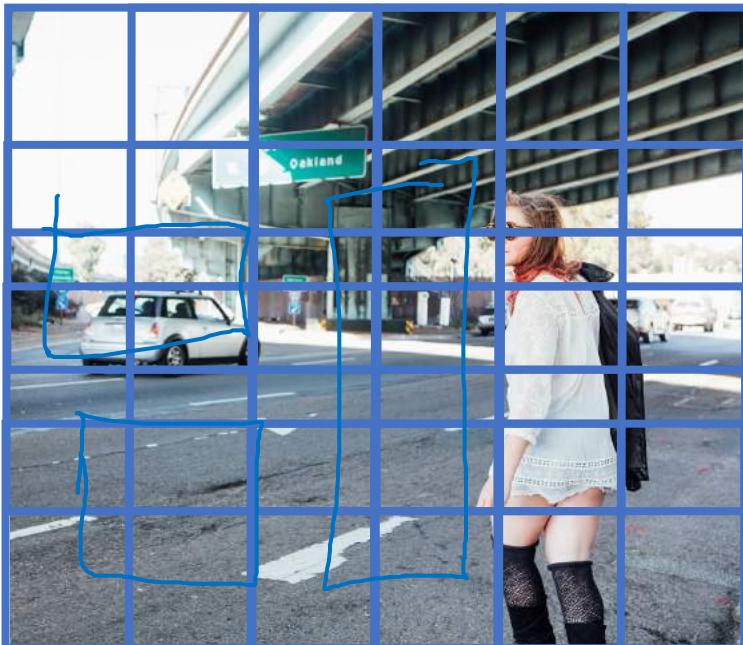


deeplearning.ai

Object Detection

Region proposals
(Optional)

Region proposal: R-CNN



Segmentation algorithm

~2,000

Faster algorithms

- R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box. ←
- Fast R-CNN: Propose regions. Use convolution implementation of sliding windows to classify all the proposed regions. ←
- Faster R-CNN: Use convolutional network to propose regions.

[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]

[Girshik, 2015. Fast R-CNN]

[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]

Andrew Ng



deeplearning.ai

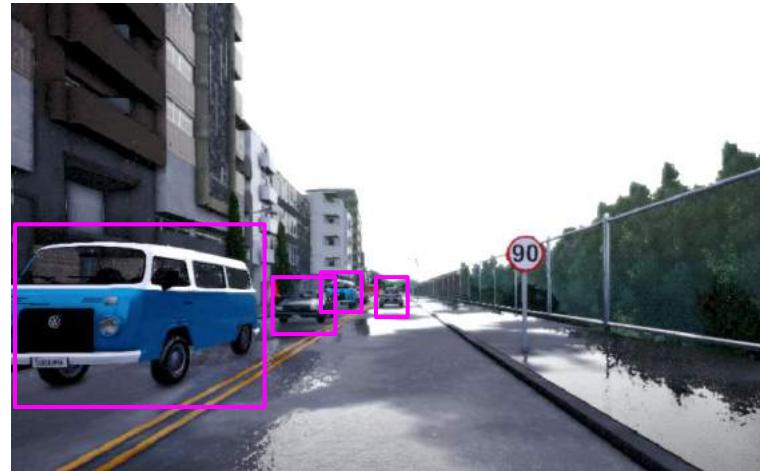
Convolutional Neural Networks

Semantic segmentation with U-Net

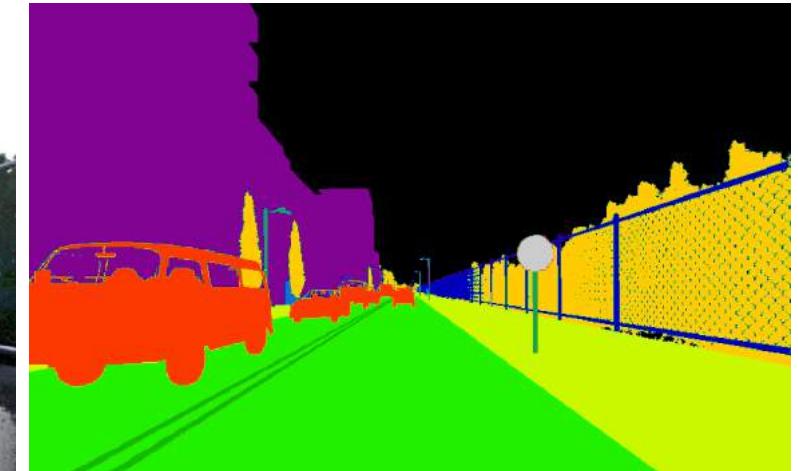
Object Detection vs. Semantic Segmentation



Input image

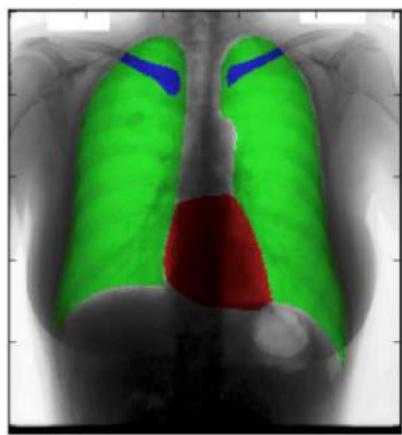


Object Detection

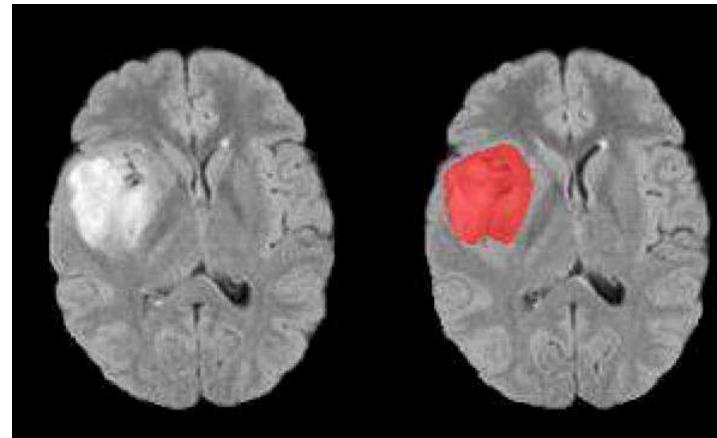


Semantic Segmentation

Motivation for U-Net



Chest X-Ray



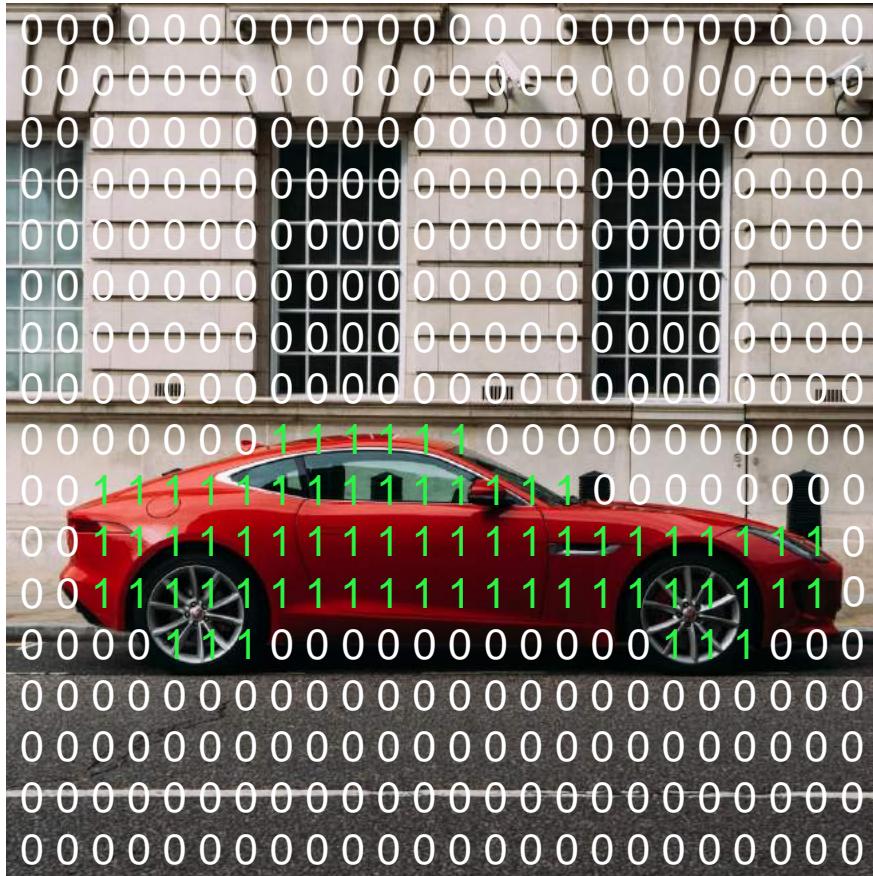
Brain MRI

[Novikov et al., 2017, Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs]

[Dong et al., 2017, Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks]

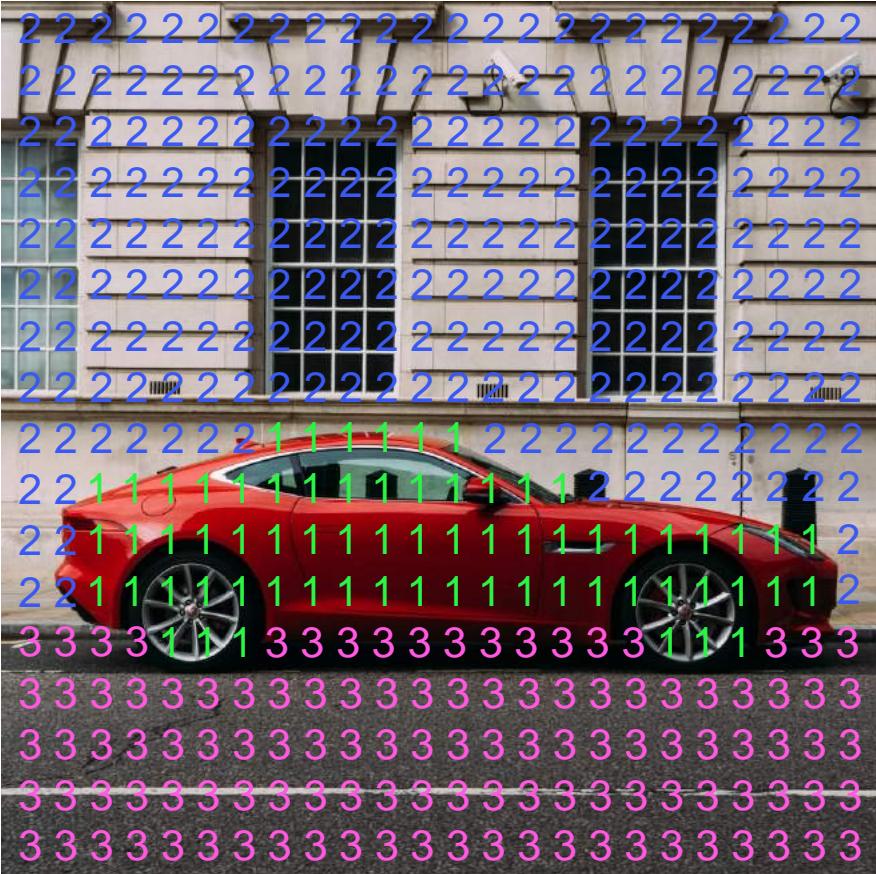
Andrew Ng

Per-pixel class labels



- 1. Car
- 0. Not Car

Per-pixel class labels

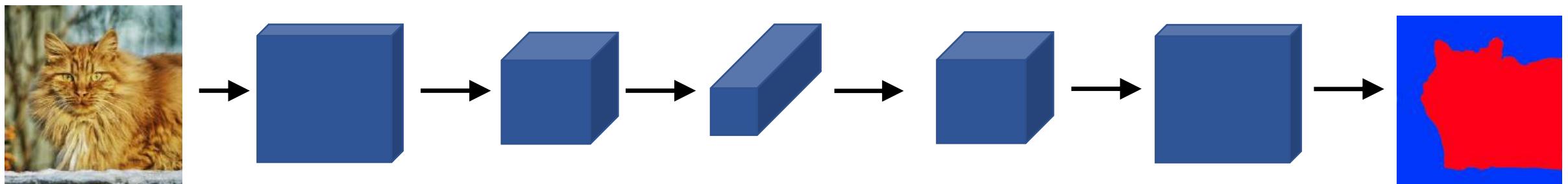


1. Car
 2. Building
 3. Road

2
2
2
2
2
2
2
2
2
2
2
2
2
2 2 1 2
2 2 1 2
3 3 3 3 1 1 1 3 3 3 3 3 3 3 3 3 3 3 3 3 3 1 1 1 3 3 3
3
3
3 3

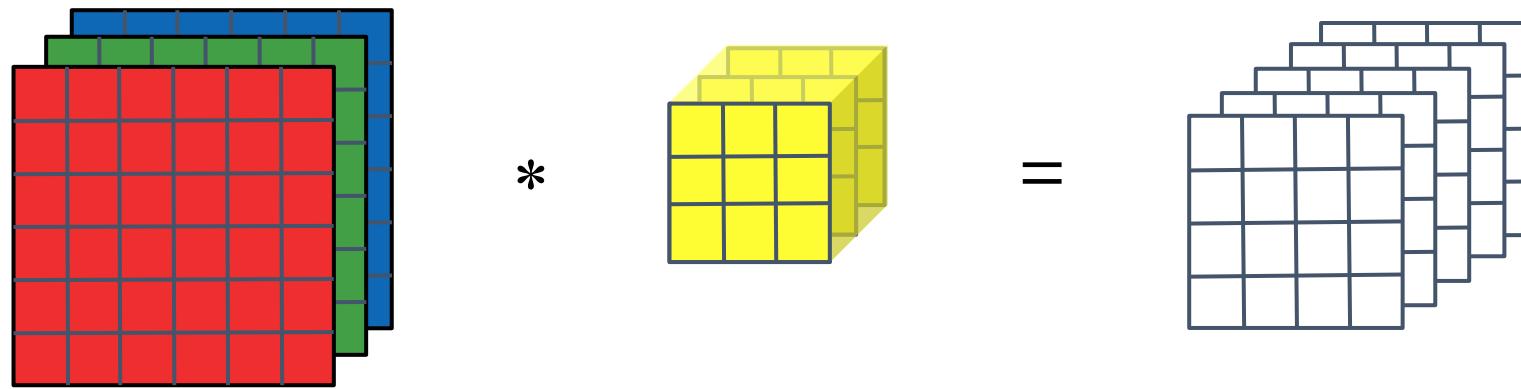
Segmentation Map

Deep Learning for Semantic Segmentation

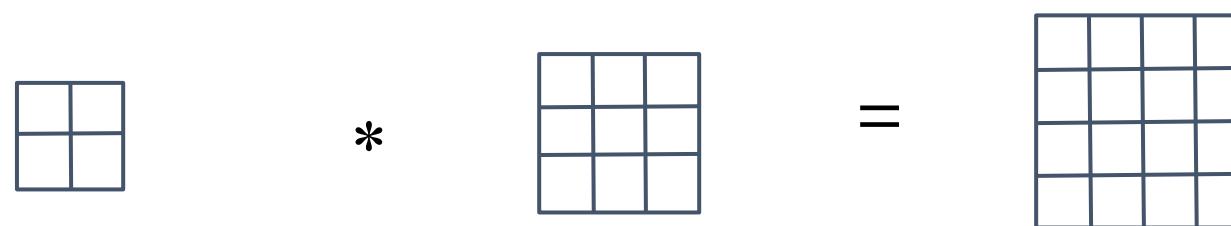


Transpose Convolution

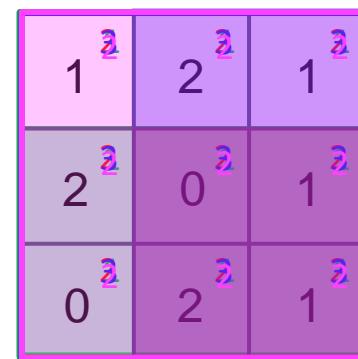
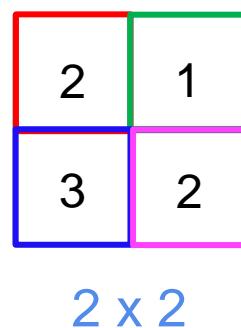
Normal Convolution



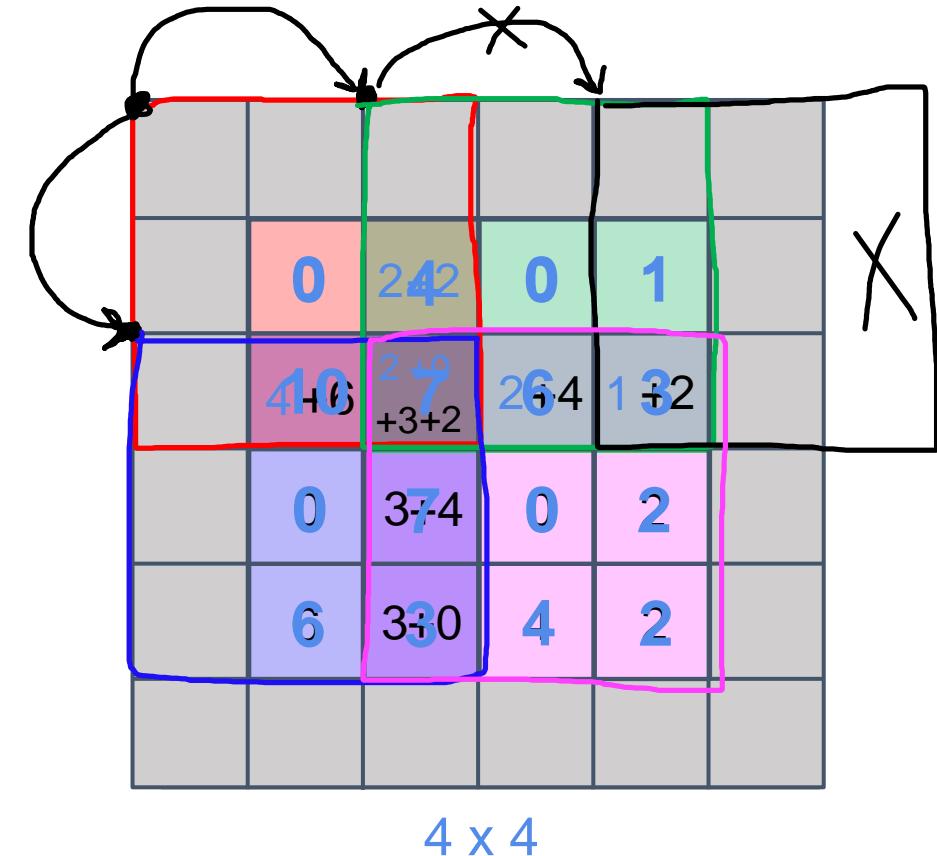
Transpose Convolution



Transpose Convolution



weight filter

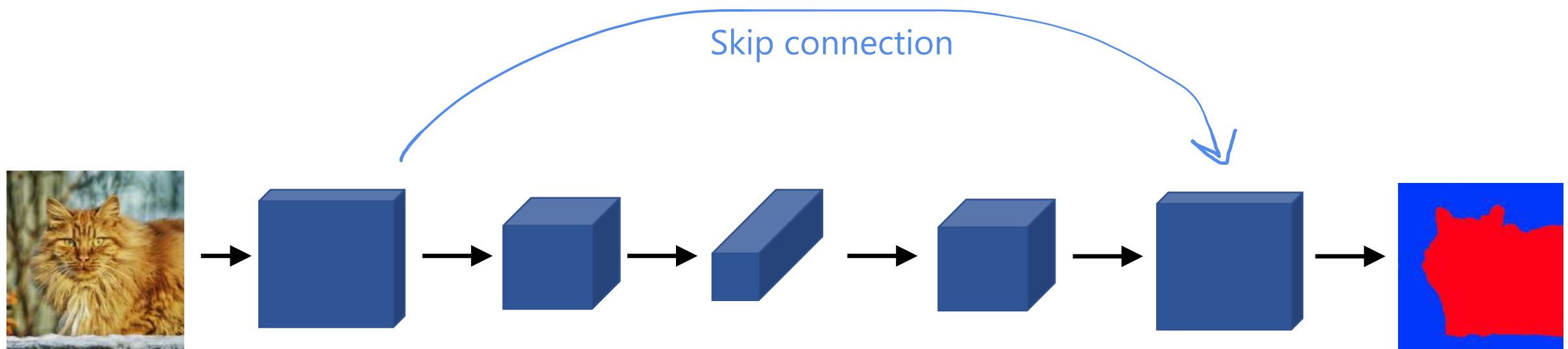


filter $f \times f = 3 \times 3$

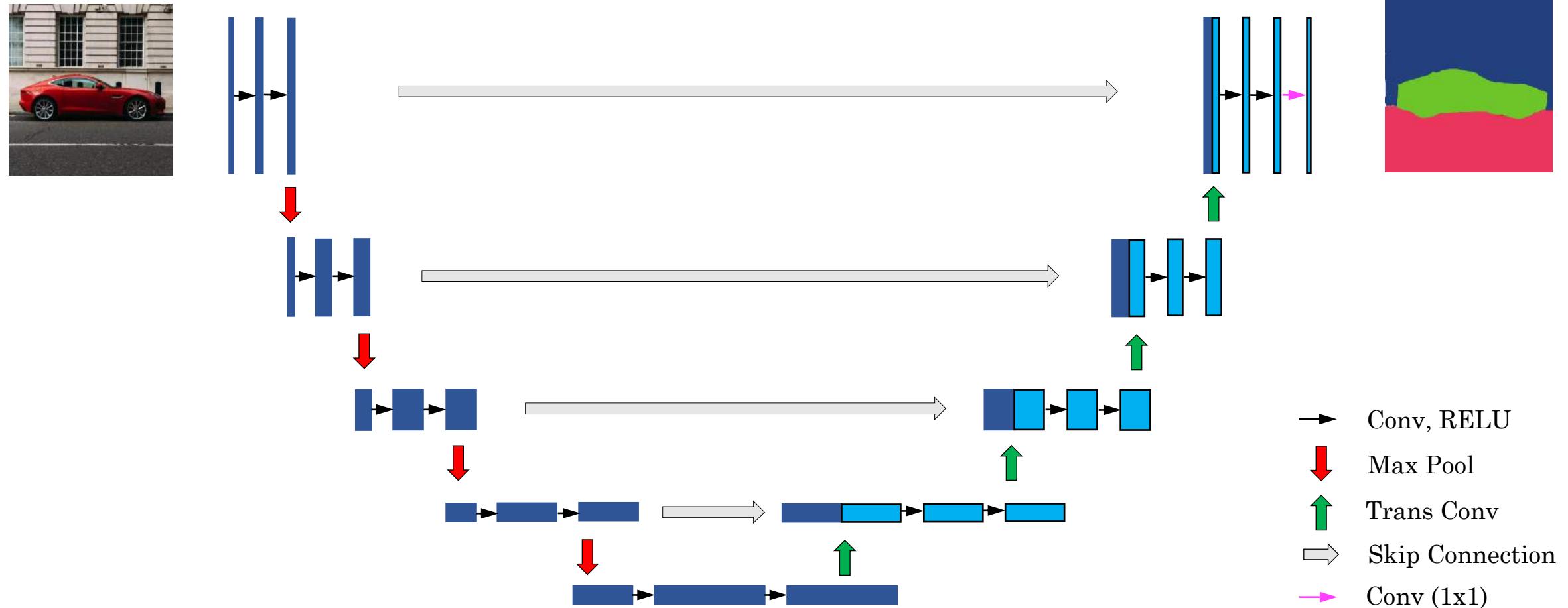
padding $p = 1$

stride $s = 2$

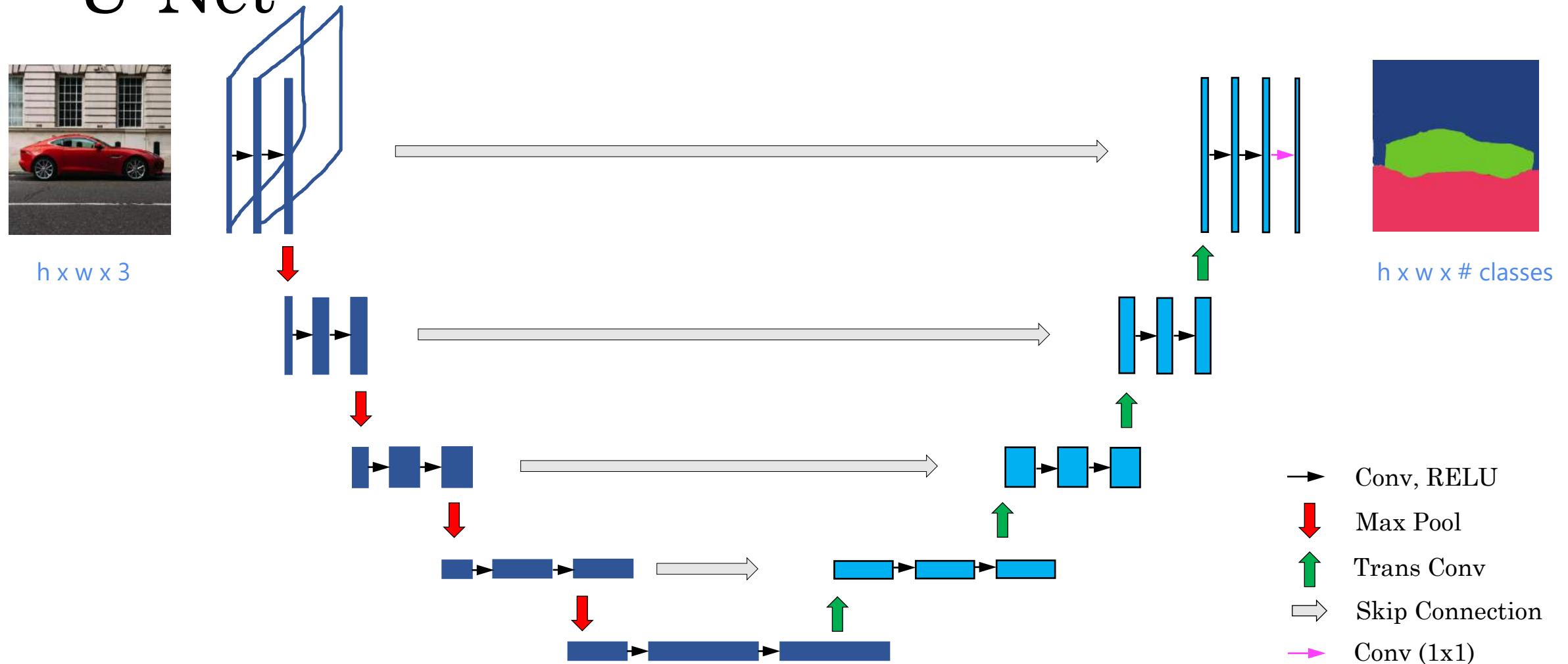
Deep Learning for Semantic Segmentation



U-Net



U-Net



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>

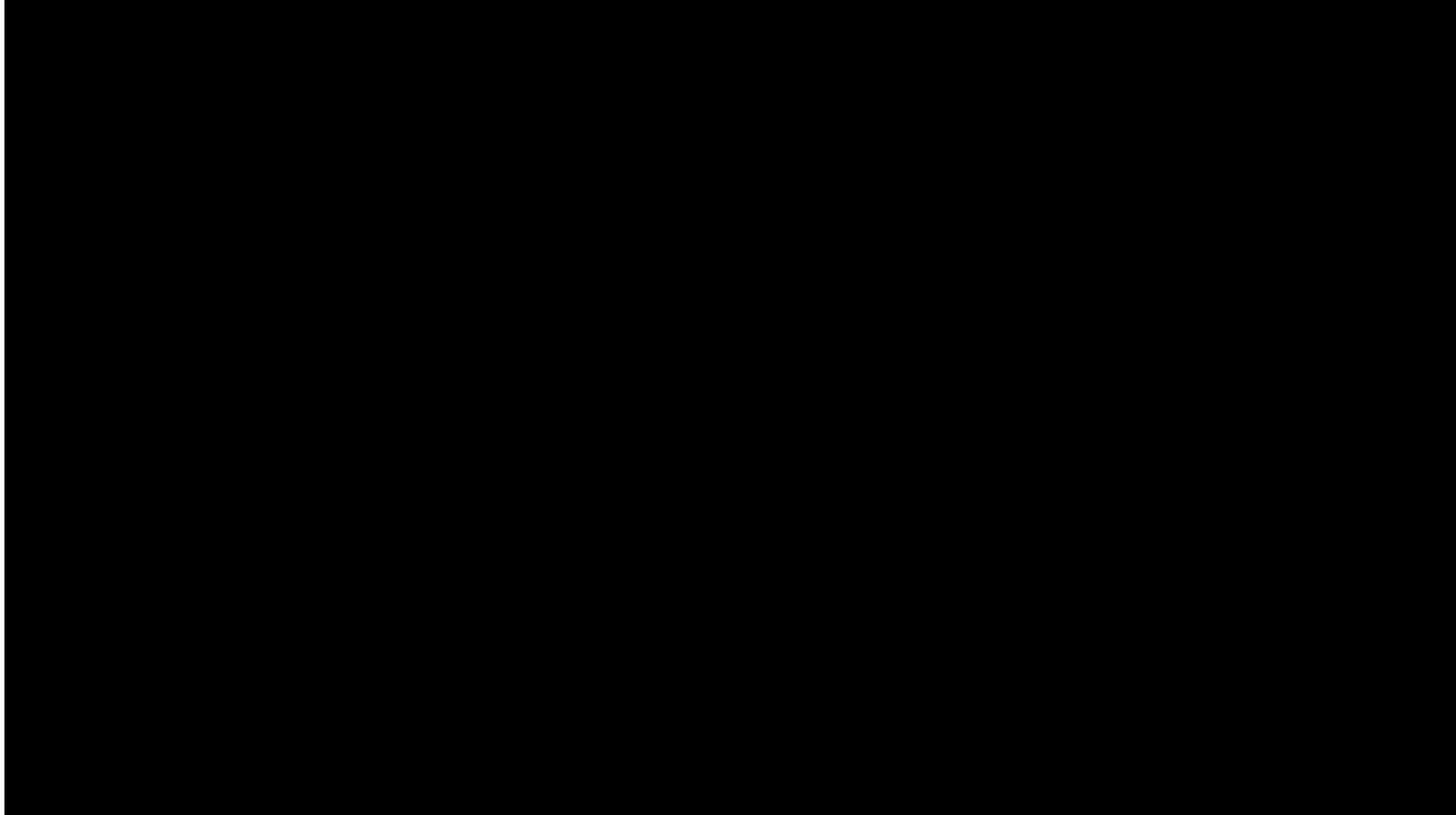


deeplearning.ai

Face recognition

What is face
recognition?

Face recognition



Face verification vs. face recognition

→ Verification

- Input image, name/ID
- Output whether the input image is that of the claimed person

1:1

99%

99.9
~~~

## → Recognition

- Has a database of K persons
- Get an input image
- Output ID if the image is any of the K persons (or “not recognized”)

1:K

K=100 ←



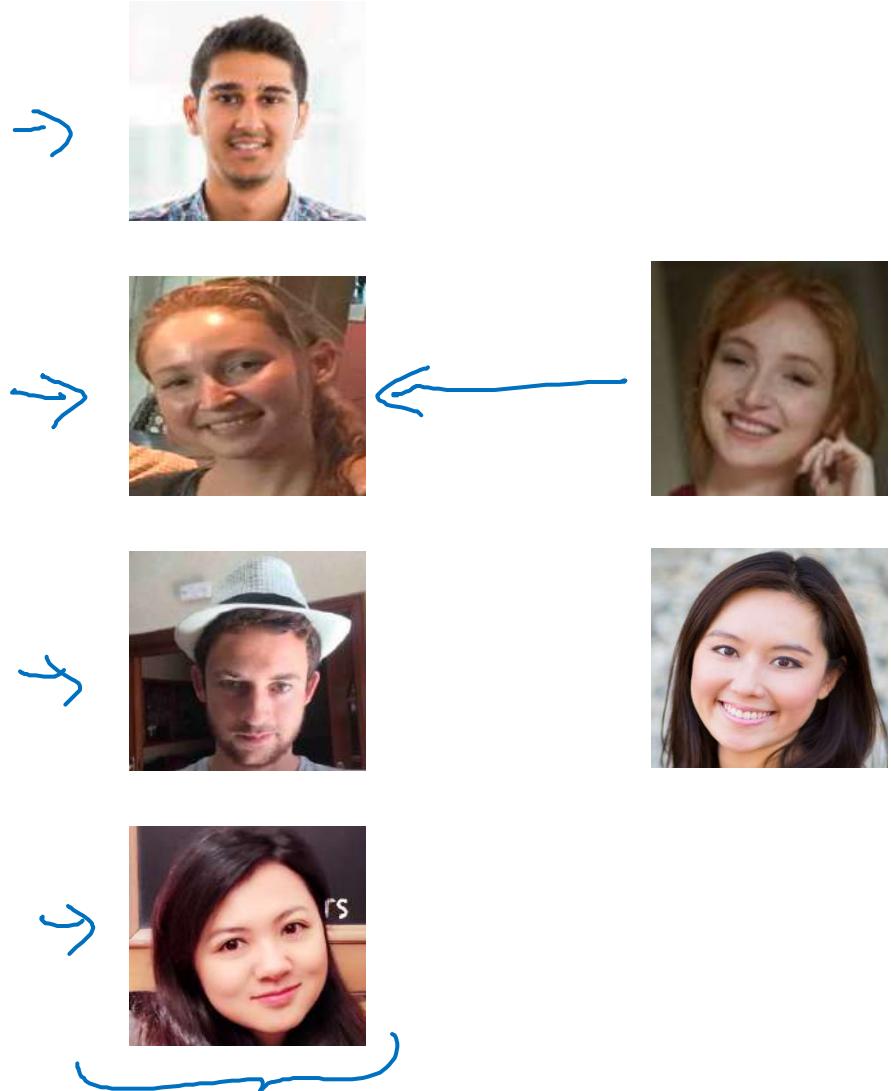
deeplearning.ai

# Face recognition

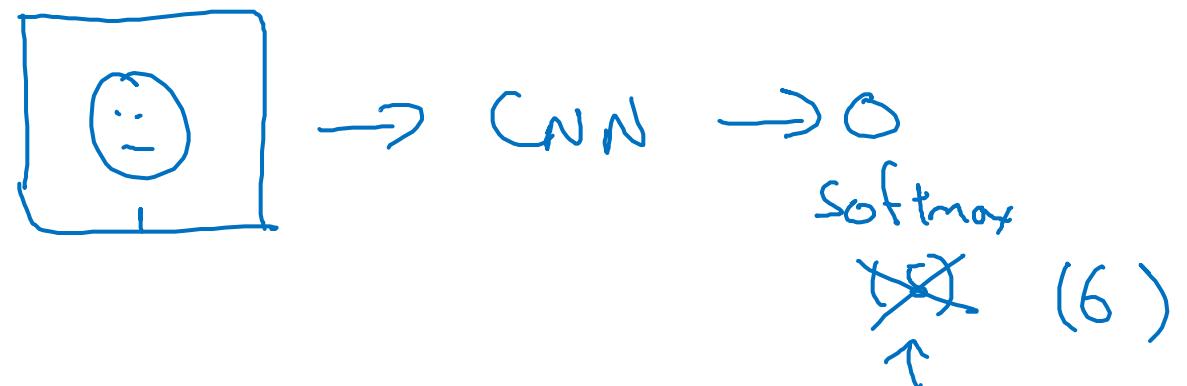
---

# One-shot learning

# One-shot learning



Learning from one example to recognize the person again



# Learning a “similarity” function

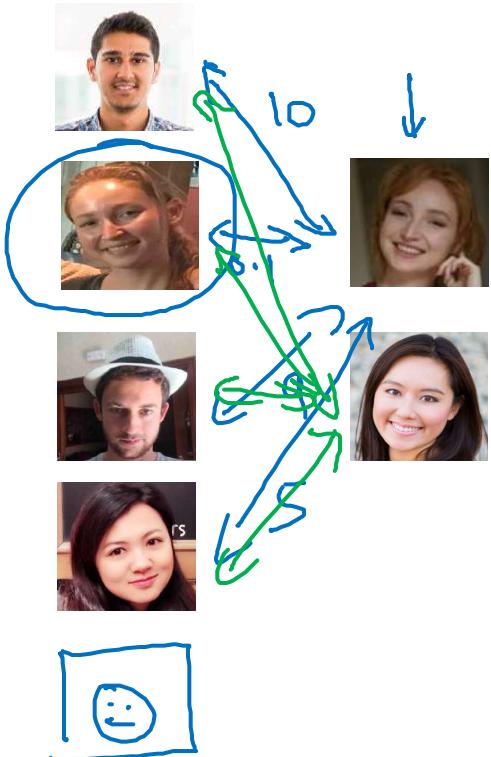
→  $d(\underline{\text{img1}}, \underline{\text{img2}})$  = degree of difference between images

If  $d(\text{img1}, \text{img2}) \leq \tau$

$> \tau$

“some”  
“different”

Verification.



$d(\text{img1}, \text{img2})$



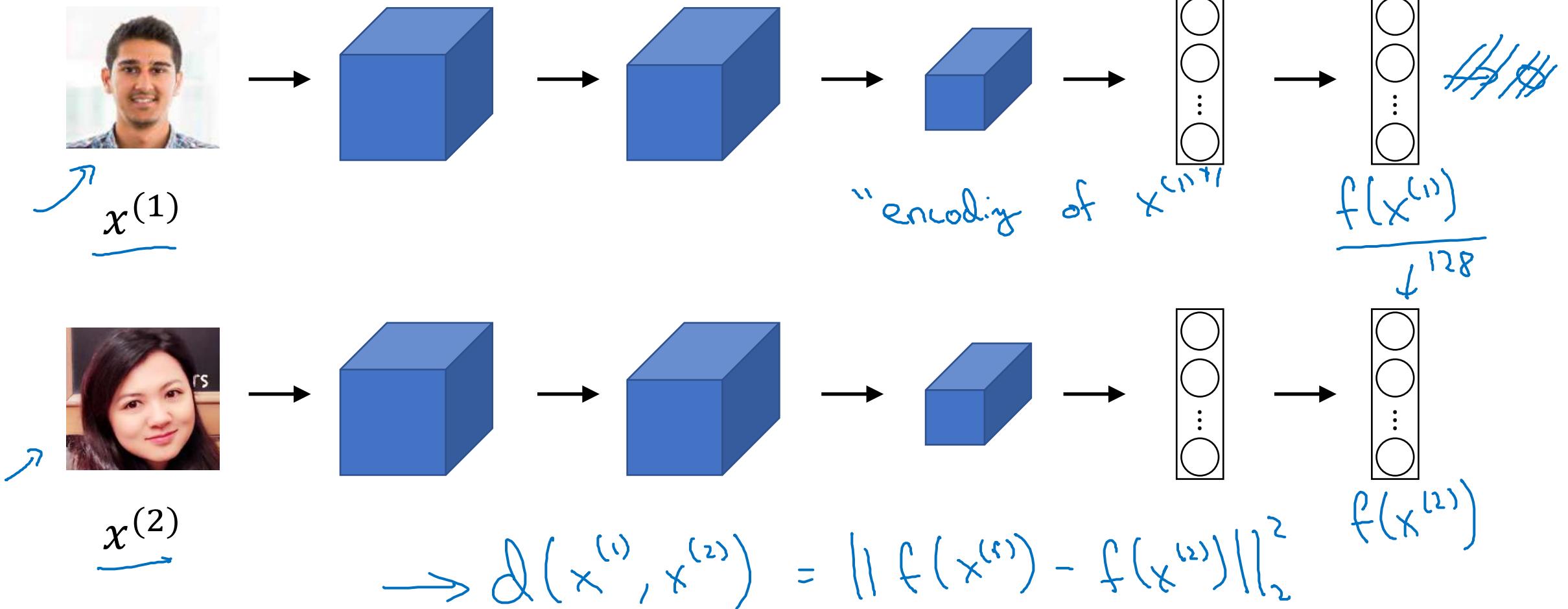
deeplearning.ai

# Face recognition

---

## Siamese network

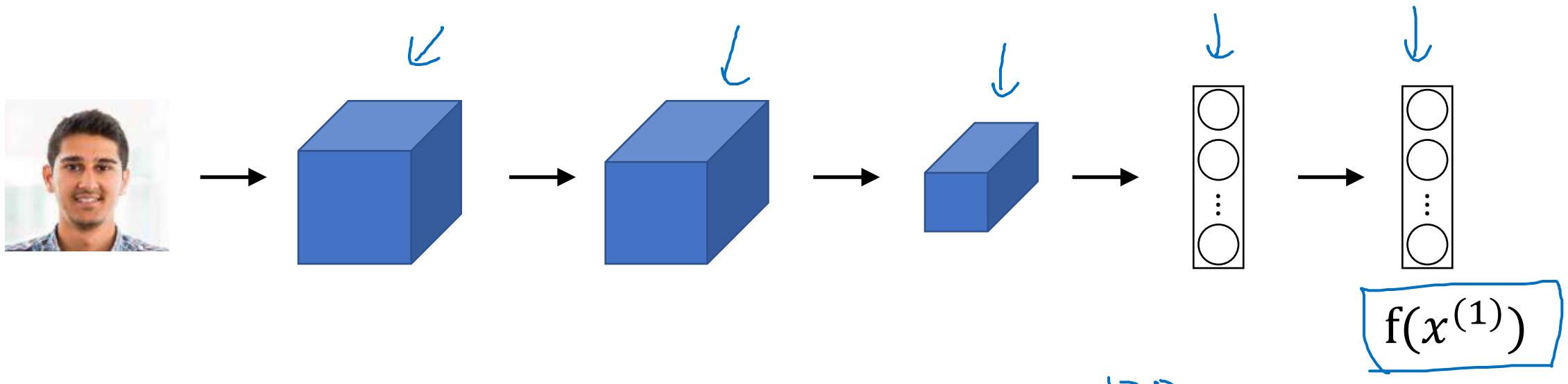
# Siamese network



[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

# Goal of learning



Parameters of NN define an encoding  $f(x^{(i)})$  128

Learn parameters so that:

If  $x^{(i)}, x^{(j)}$  are the same person,  $\|f(x^{(i)}) - f(x^{(j)})\|^2$  is small.

If  $x^{(i)}, x^{(j)}$  are different persons,  $\|f(x^{(i)}) - f(x^{(j)})\|^2$  is large.



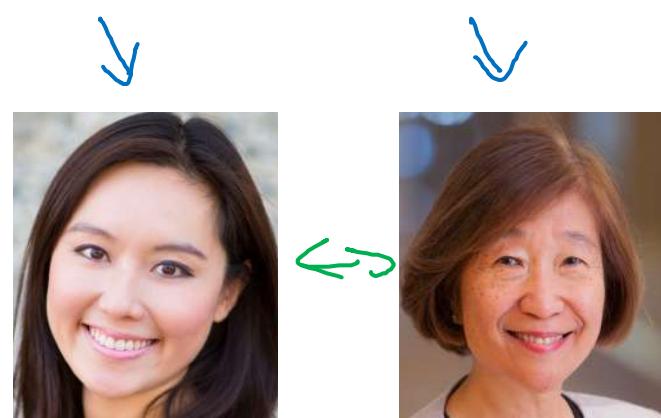
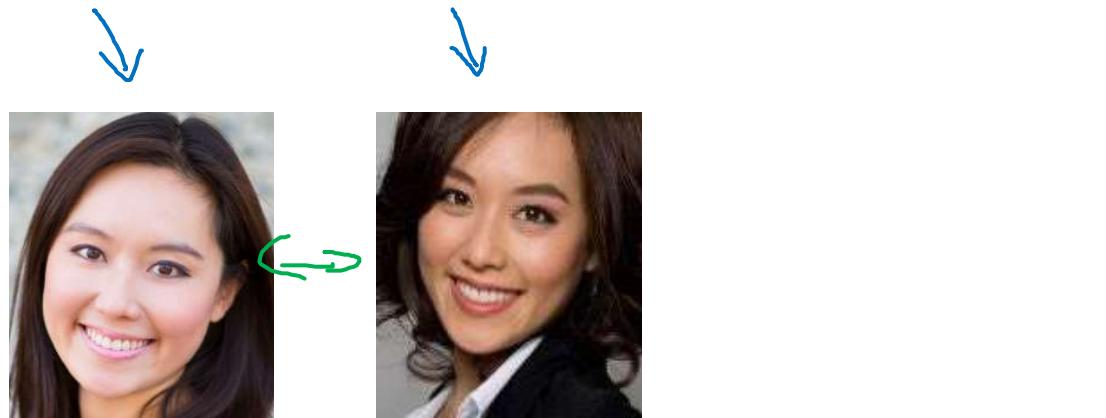
deeplearning.ai

# Face recognition

---

## Triplet loss

# Learning Objective



Anchor      Positive  
 $A$        $P$   
 $d(A, P) = 0.5$

Want:  $\frac{\|f(A) - f(P)\|^2}{d(A, P)} + \gamma \leq \frac{\|f(A) - f(N)\|^2}{d(A, N)}$   $\rightarrow 0.2$

Anchor      Negative  
 $A$        $N$   
 $d(A, N) = 0.5$   $\rightarrow 0.7$

$$\frac{\|f(A) - f(P)\|^2}{\circ} - \frac{\|f(A) - f(N)\|^2}{\circ} + \gamma \leq 0 \quad 4/4 \quad f(\text{img}) = \vec{0}$$

Margin

# Loss function

Given 3 images

$A, P, N$ :

$$\underline{L(A, P, N)} = \max \left( \left[ \|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \lambda \right], 0 \right)$$

$$J = \sum_{i=1}^m L(A^{(i)}, P^{(i)}, N^{(i)})$$

$A, P$   
 $T$

Training set:  $\underbrace{10k}_{\infty}$  pictures of  $\frac{1k}{\infty}$  persons

# Choosing the triplets A,P,N

During training, if A,P,N are chosen randomly,  
 $d(A, P) + \alpha \leq d(A, N)$  is easily satisfied.

$$\underbrace{\|f(A) - f(P)\|^2}_{\text{ }} + \alpha \leq \underbrace{\|f(A) - f(N)\|^2}_{\text{ }}$$

Choose triplets that're “hard” to train on.

$$\begin{aligned} \cancel{d(A, P)} + \alpha &\leq \cancel{d(A, N)} \\ \frac{d(A, P)}{\downarrow} &\approx \frac{d(A, N)}{\uparrow} \end{aligned}$$

Face Net  
Deep Face



# Training set using triplet loss

Anchor



Positive



Negative



:

:

:



J

$$d(x^{(i)}, x^{(j)})$$



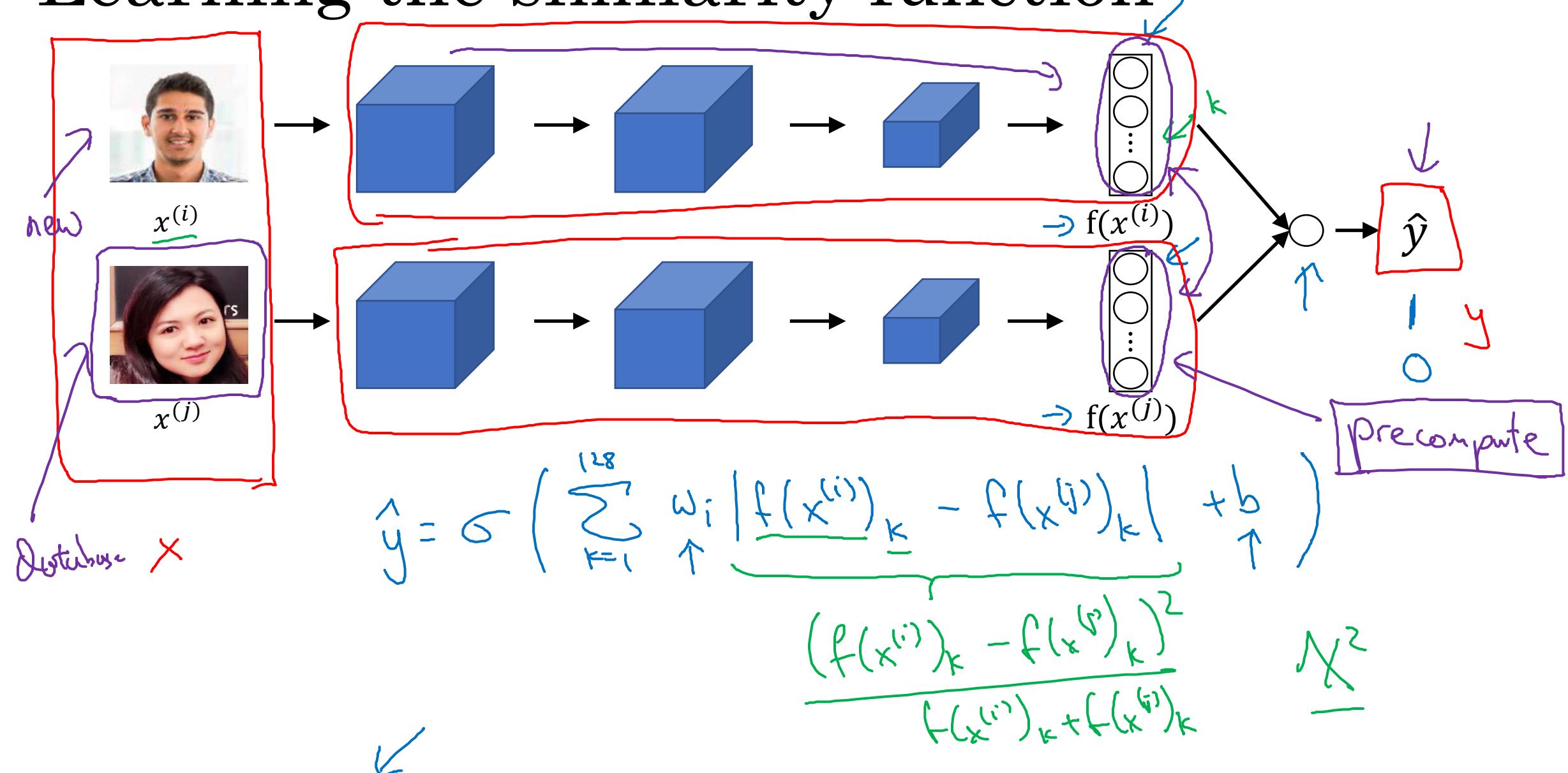
deeplearning.ai

# Face recognition

---

# Face verification and binary classification

# Learning the similarity function

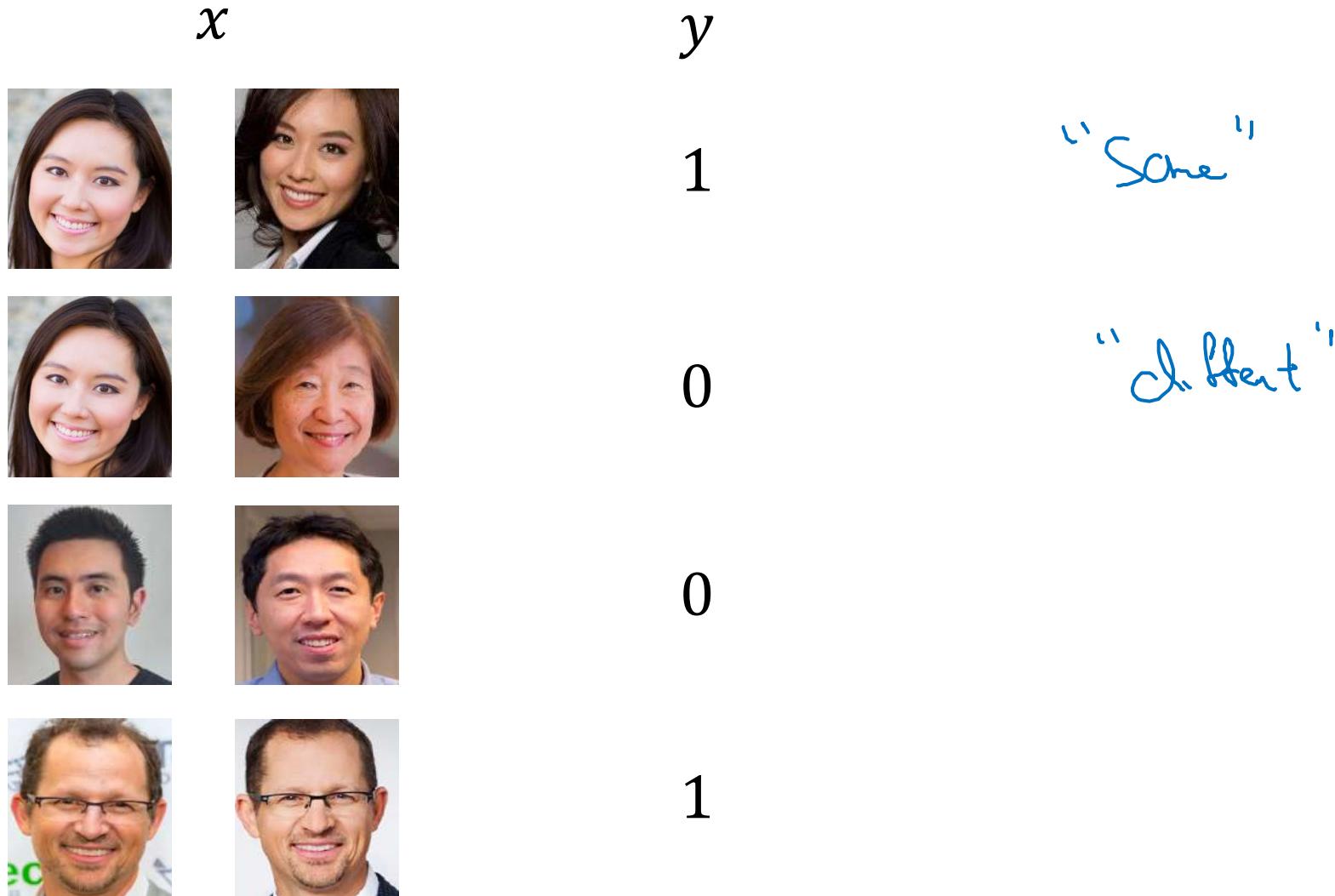


$$\hat{y} = \sigma \left( \sum_{k=1}^{128} w_i |f(x^{(i)})_k - f(x^{(j)})_k| + b \right)$$

$$\text{distance}_k = \frac{(f(x^{(i)})_k - f(x^{(j)})_k)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$$

$$\frac{1}{N^2}$$

# Face verification supervised learning





deeplearning.ai

# Neural Style Transfer

---

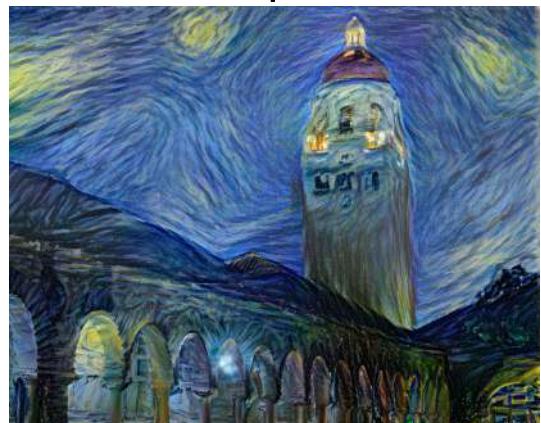
What is neural style  
transfer?

# Neural style transfer



Content ( $c$ )

Style ( $s$ )



Generated image ( $g$ )



Content ( $c$ )

Style ( $s$ )



Generated image ( $g$ )



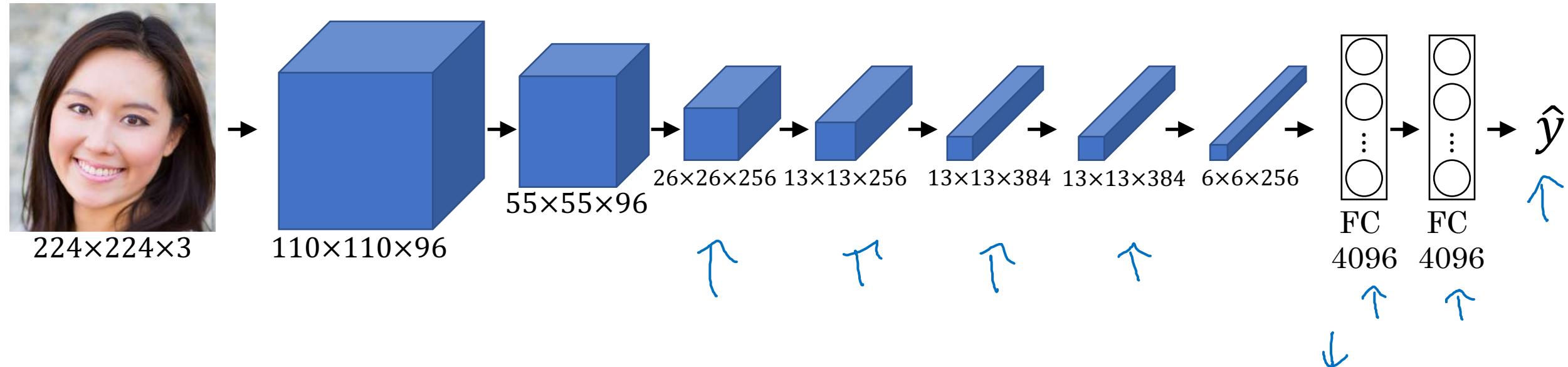
deeplearning.ai

# Neural Style Transfer

---

What are deep  
ConvNets learning?

# Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

# Visualizing deep layers



Layer 1



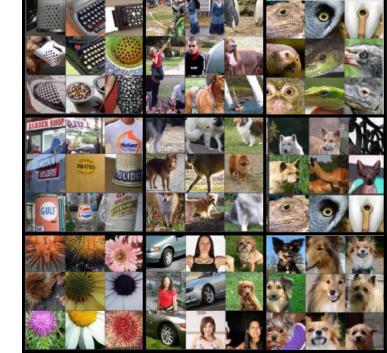
Layer 2



Layer 3

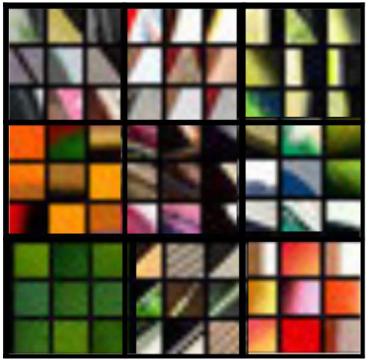


Layer 4



Layer 5

# Visualizing deep layers: Layer 1



Layer 1



Layer 2



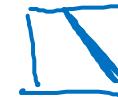
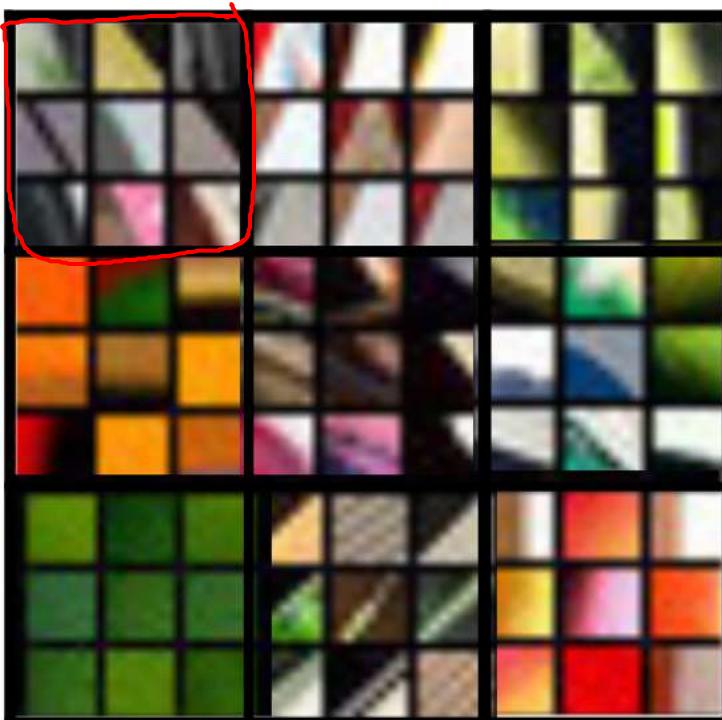
Layer 3



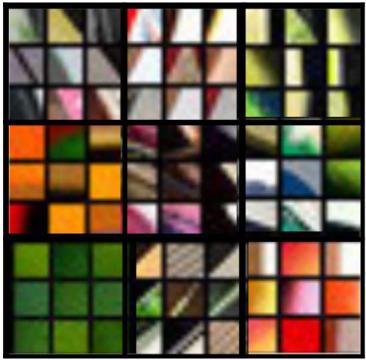
Layer 4



Layer 5



# Visualizing deep layers: Layer 2



Layer 1



Layer 2



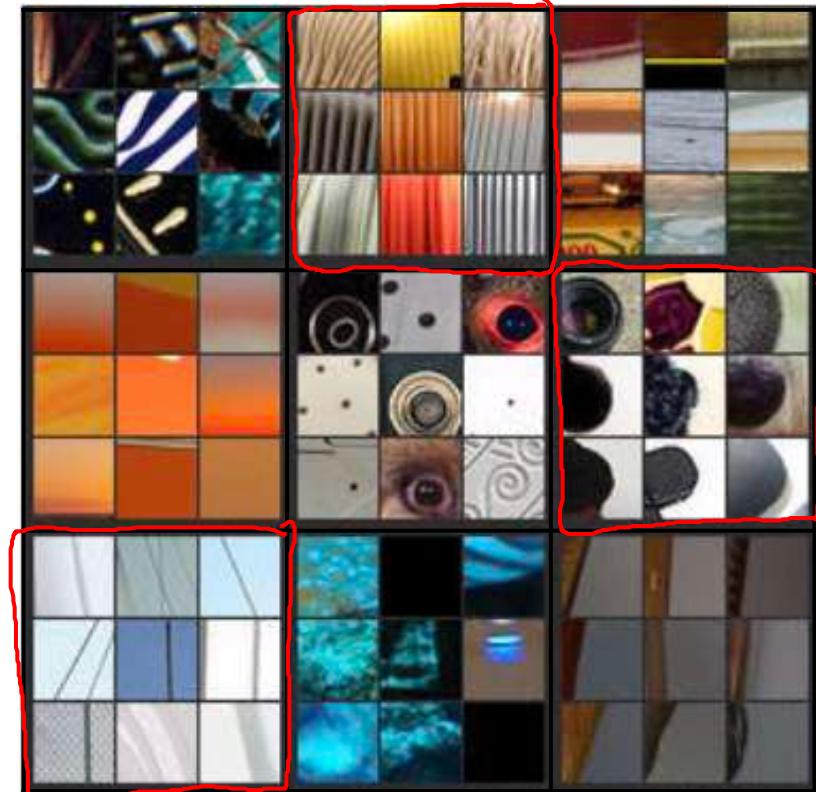
Layer 3



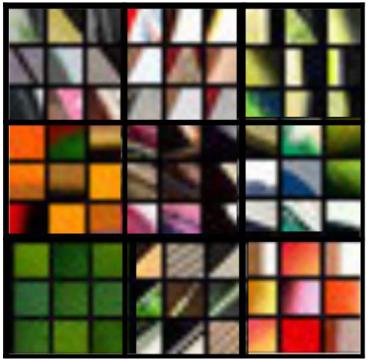
Layer 4



Layer 5



# Visualizing deep layers: Layer 3



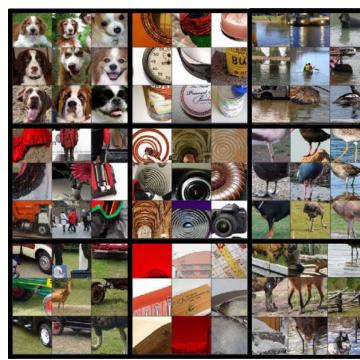
Layer 1



Layer 2



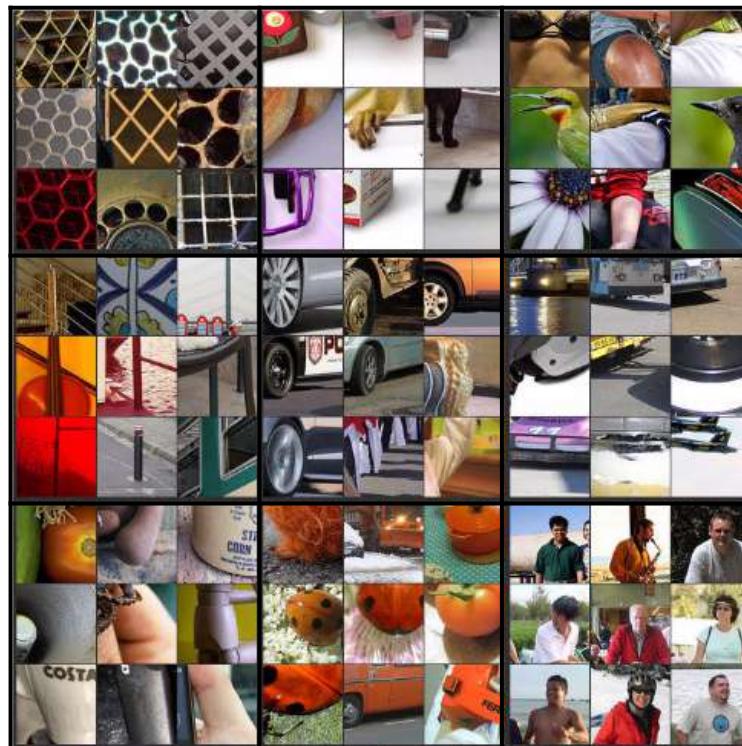
Layer 3



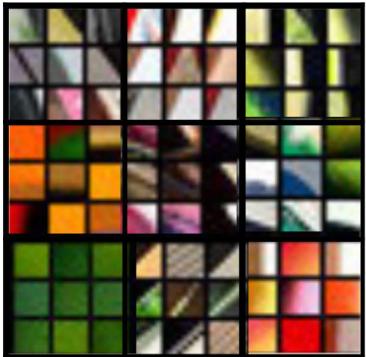
Layer 4



Layer 5



# Visualizing deep layers: Layer 3



Layer 1

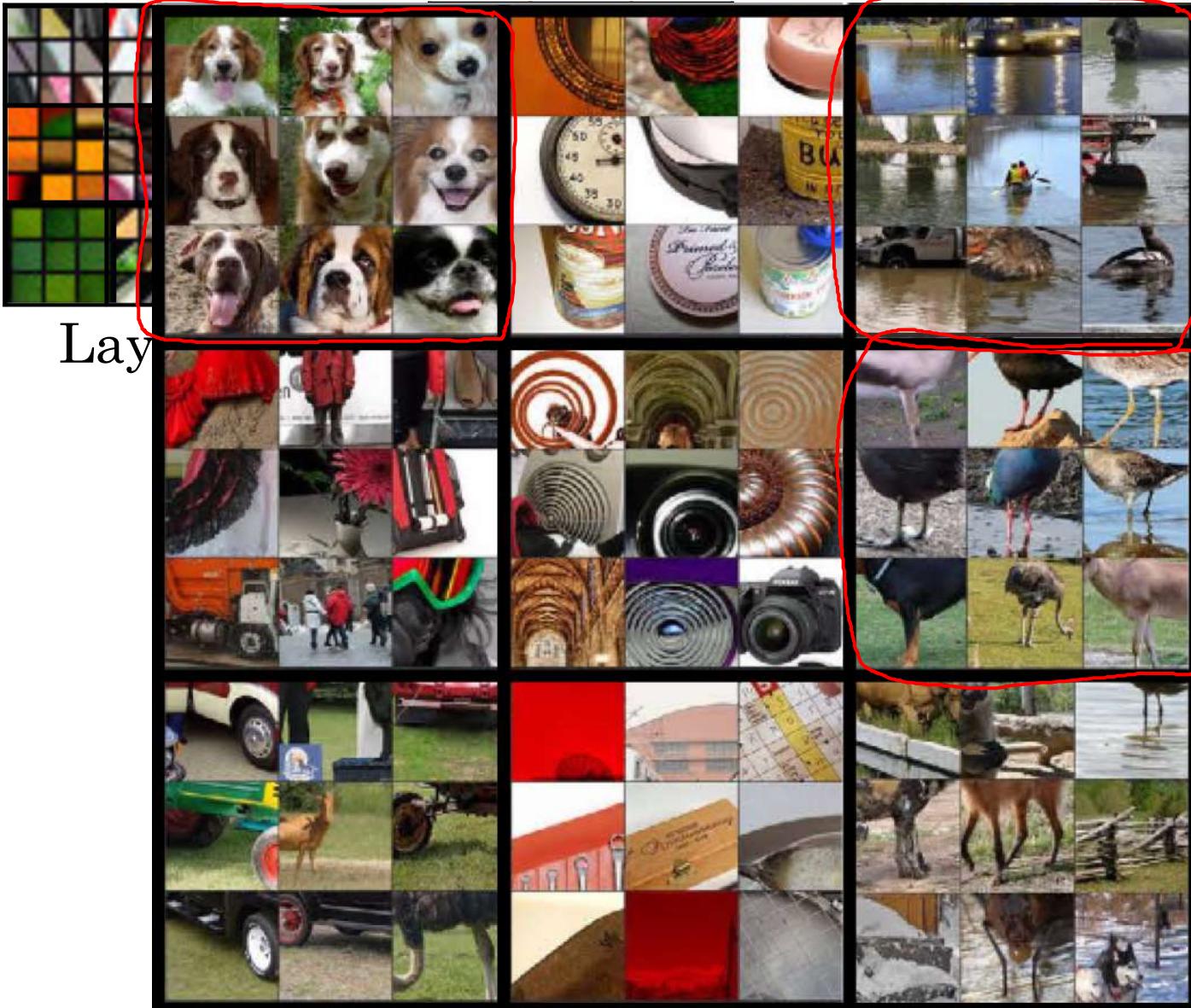


L



Layer 5

# Visualizing deep layers: Layer 4



Layer 4

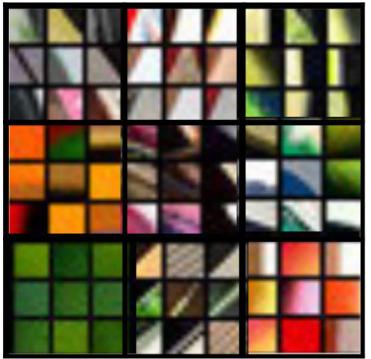


Layer 4



Layer 5

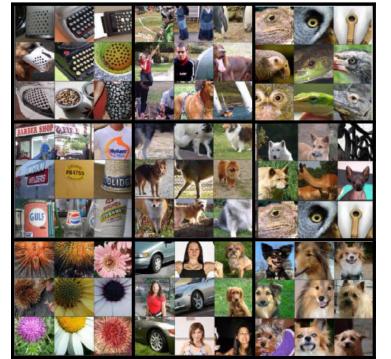
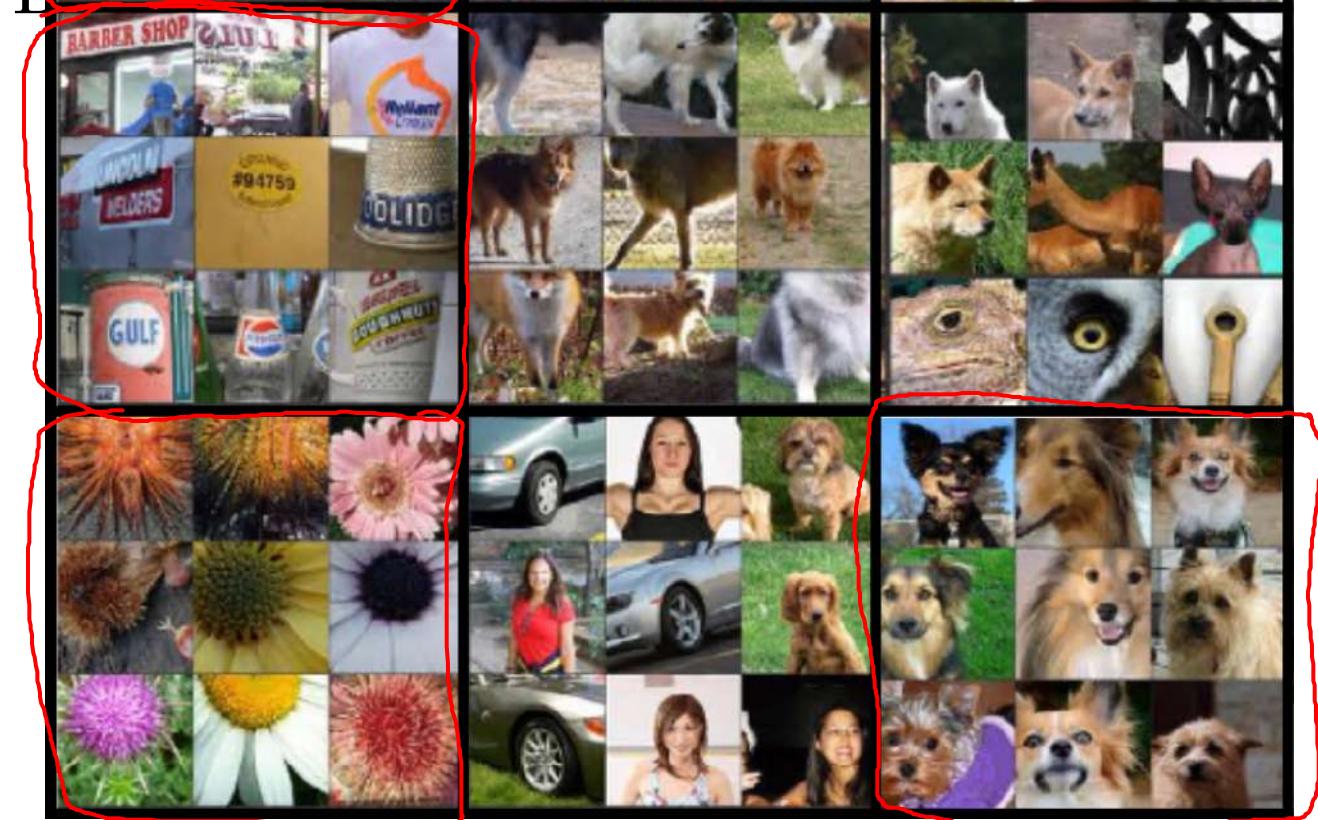
# Visualizing deep layers: Layer 5



Layer 1



Layer 2



Layer 5



deeplearning.ai

# Neural Style Transfer

---

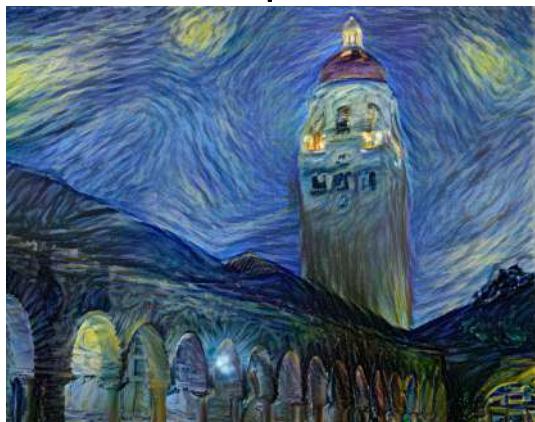
## Cost function

# Neural style transfer cost function



Content C

Style S



Generated image G

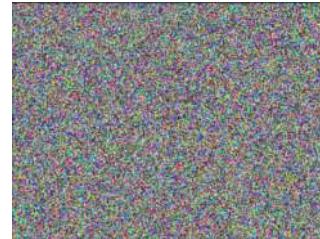
$$J(G) = \alpha J_{\text{Content}}(C, G) + \beta J_{\text{Style}}(S, G)$$

# Find the generated image G

1. Initiate G randomly

$G: \underbrace{100 \times 100}_{\text{---}} \times \underbrace{3}_{\text{---}}$

$\uparrow$   
RGB



2. Use gradient descent to minimize  $\underline{J(G)}$

$$G_t := G - \frac{\partial}{\partial G} J(G)$$





deeplearning.ai

# Neural Style Transfer

---

## Content cost function

# Content cost function

$$\underline{J}(G) = \alpha \underline{J}_{content}(C, G) + \beta J_{style}(S, G)$$

- Say you use hidden layer  $\underline{l}$  to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let  $\underline{a}^{[l](C)}$  and  $\underline{a}^{[l](G)}$  be the activation of layer  $\underline{l}$  on the images
- If  $\underline{a}^{[l](C)}$  and  $\underline{a}^{[l](G)}$  are similar, both images have similar content

$$J_{content}(C, G) = \frac{1}{2} \left\| \underline{a}^{[l](C)} - \underline{a}^{[l](G)} \right\|^2$$



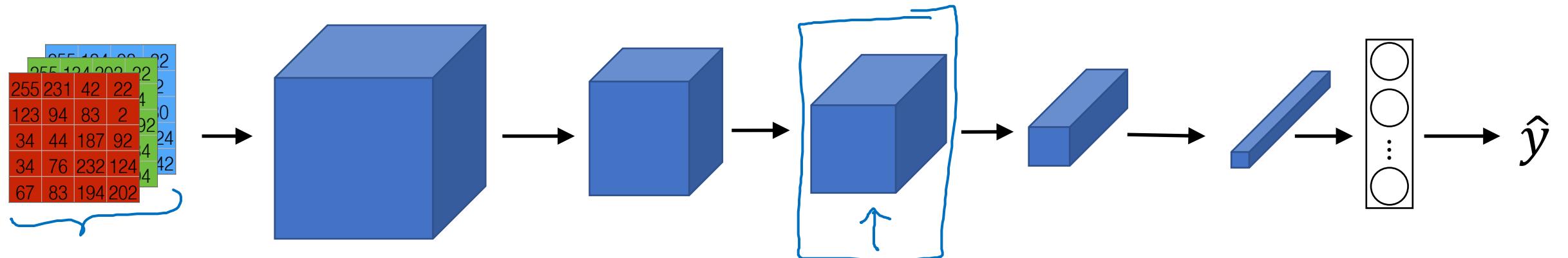
deeplearning.ai

# Neural Style Transfer

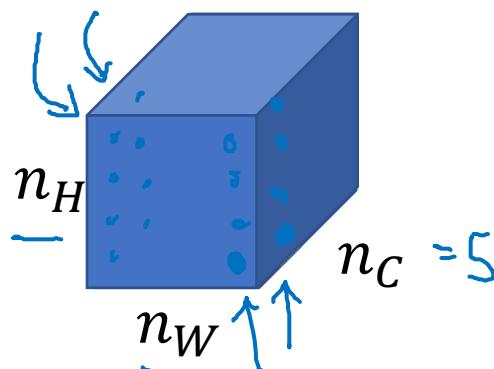
---

## Style cost function

# Meaning of the “style” of an image

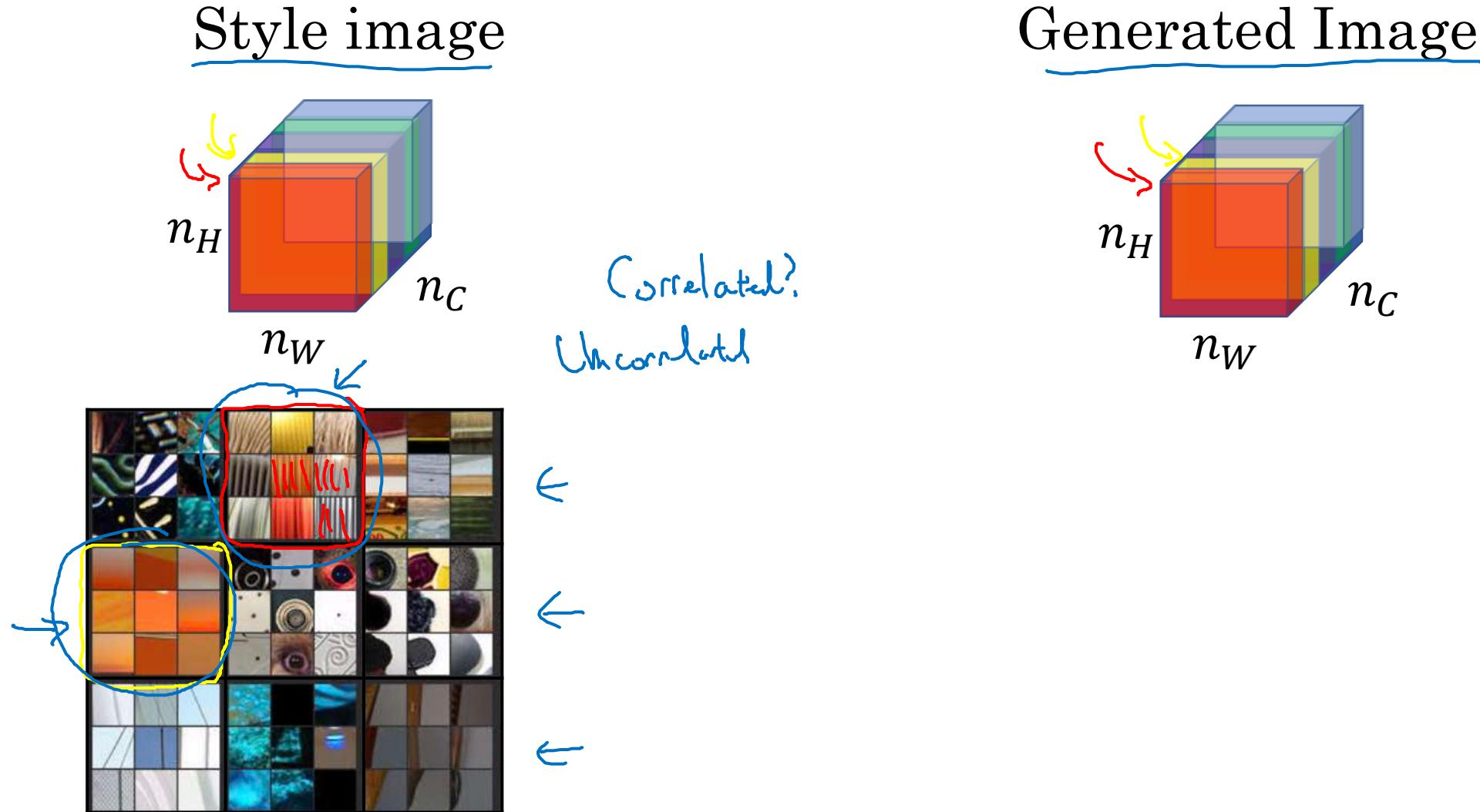


Say you are using layer  $l$ 's activation to measure “style.”  
Define style as correlation between activations across channels.



How correlated are the activations  
across different channels?

# Intuition about style of an image



# Style matrix

Let  $a_{i,j,k}^{[l]}$  = activation at  $(i, j, k)$ .  $G^{[l]}$  is  $n_c^{[l]} \times n_c^{[l]}$

$$\rightarrow G_{kk'}^{[l](s)} = \sum_{i=1}^{n_h^{[l]}} \sum_{j=1}^{n_w^{[l]}} a_{ijk}^{[l](s)} a_{ijk'}^{[l](s)}$$

$$\rightarrow G_{kk'}^{[l](G)} = \sum_{i=1}^{n_h^{[l]}} \sum_{j=1}^{n_w^{[l]}} a_{ijk}^{[l](G)} a_{ijk'}^{[l](G)}$$

H W C  
↓ ↓ ↓

$n_c$

$$G_{kk'}^{[l]} \quad \forall k, k' \in \{1, \dots, n_c\}$$

"Gram matrix"

$$\begin{aligned} J_{style}^{[l]}(S, G) &= \frac{1}{(\dots)} \| G_{kk'}^{[l](s)} - G_{kk'}^{[l](G)} \|_F^2 \\ &= \frac{1}{(2n_h^{[l]} n_w^{[l]} n_c^{[l]})^2} \sum_k \sum_{k'} (G_{kk'}^{[l](s)} - G_{kk'}^{[l](G)})^2 \end{aligned}$$

# Style cost function

$$\left\| G^{[l](s)} - G^{[l](G)} \right\|_F^2$$

$$J_{style}^{[l]}(S, G) = \frac{1}{\left(2n_H^{[l]} n_W^{[l]} n_C^{[l]}\right)^2} \sum_k \sum_{k'} (G_{kk'}^{[l](s)} - G_{kk'}^{[l](G)})$$

$$J_{style}(S, G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S, G)$$

$$\underline{J(G)} = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$



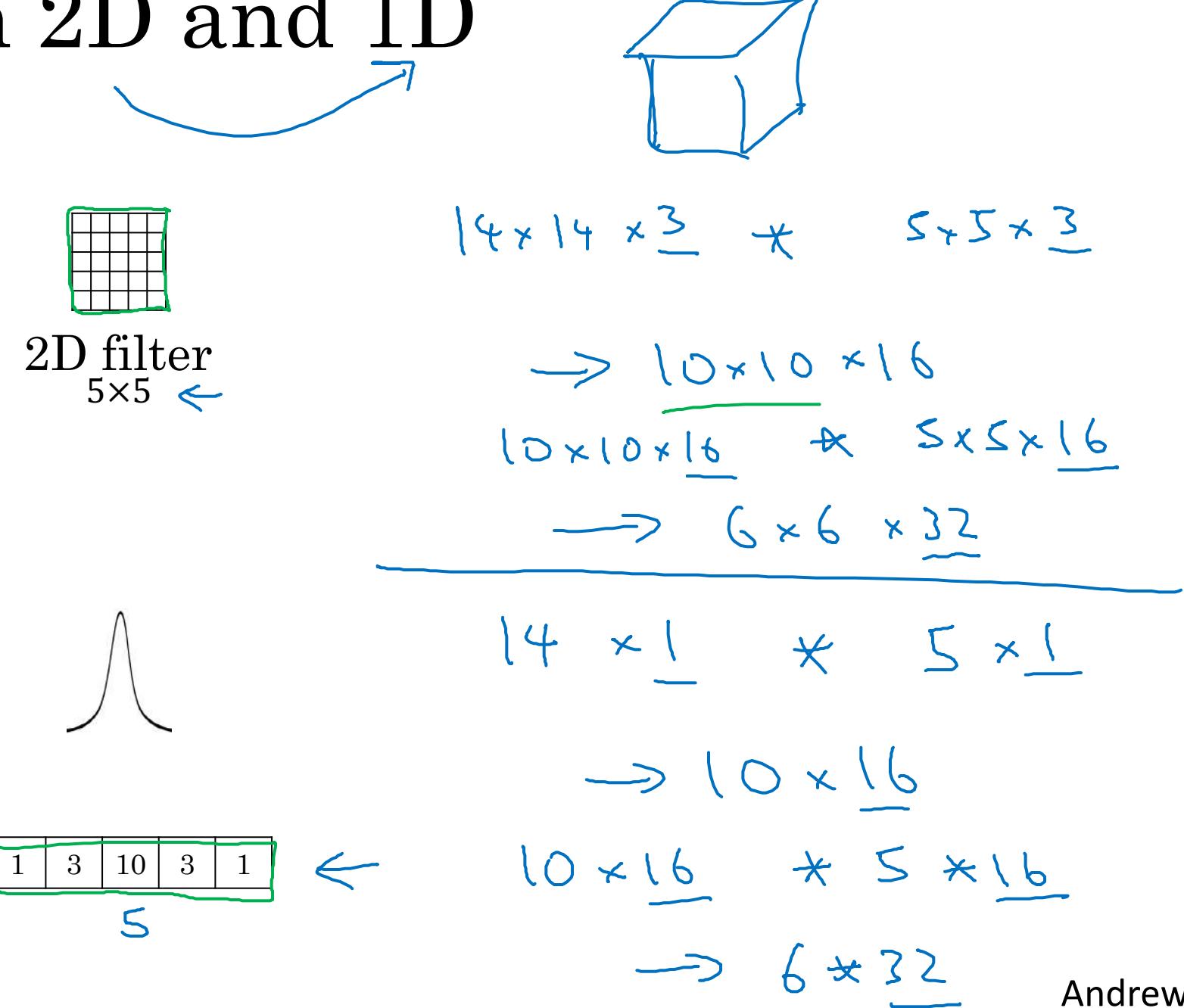
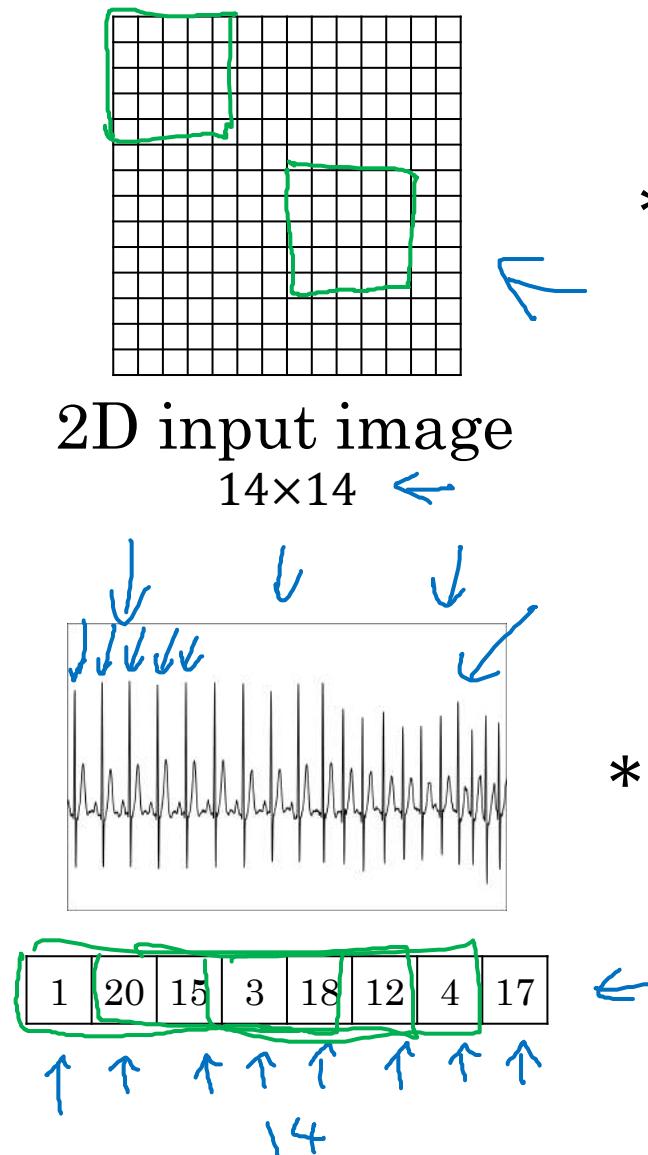
deeplearning.ai

# Convolutional Networks in 1D or 3D

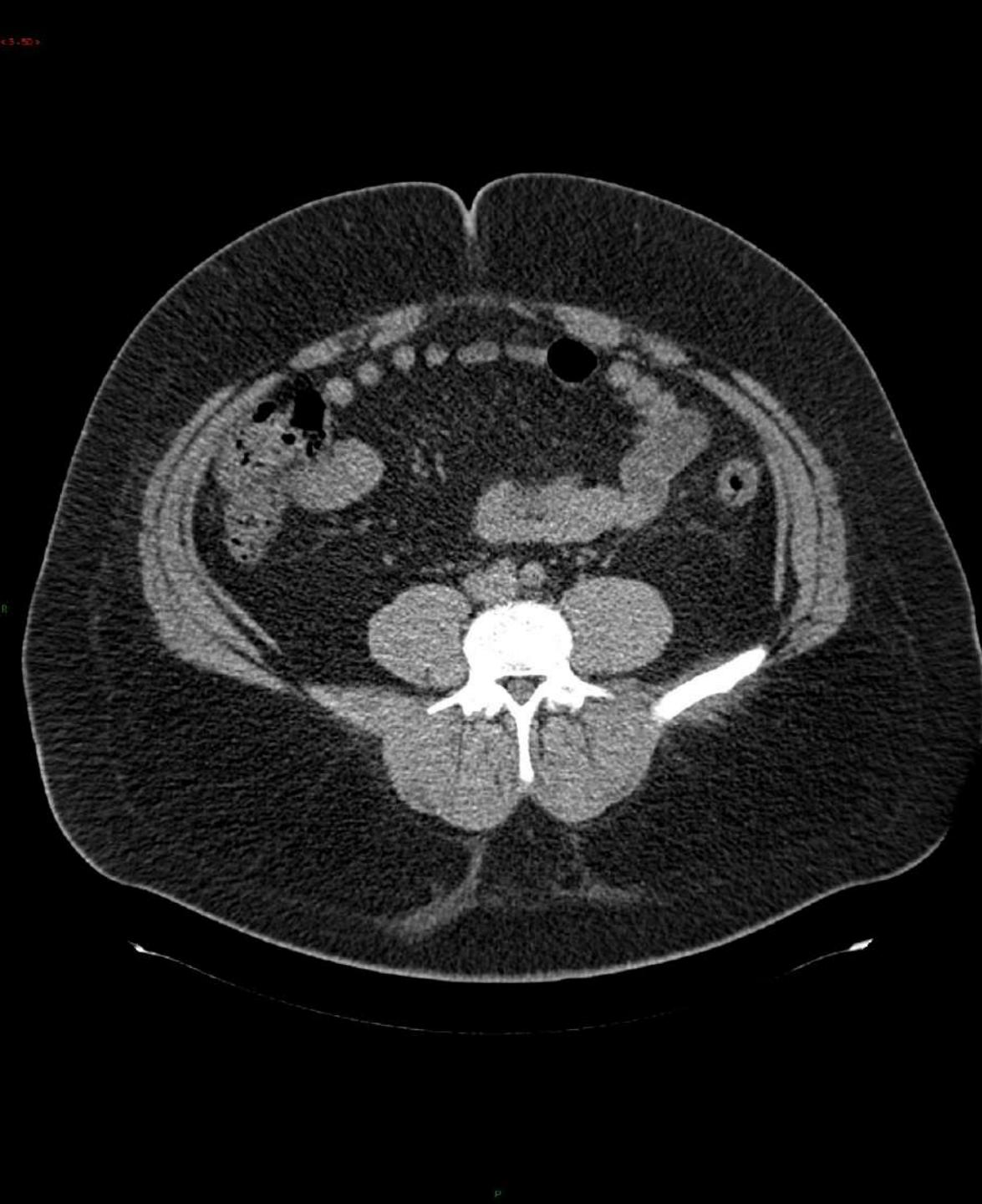
---

1D and 3D  
generalizations of  
models

# Convolutions in 2D and 1D



# 3D data



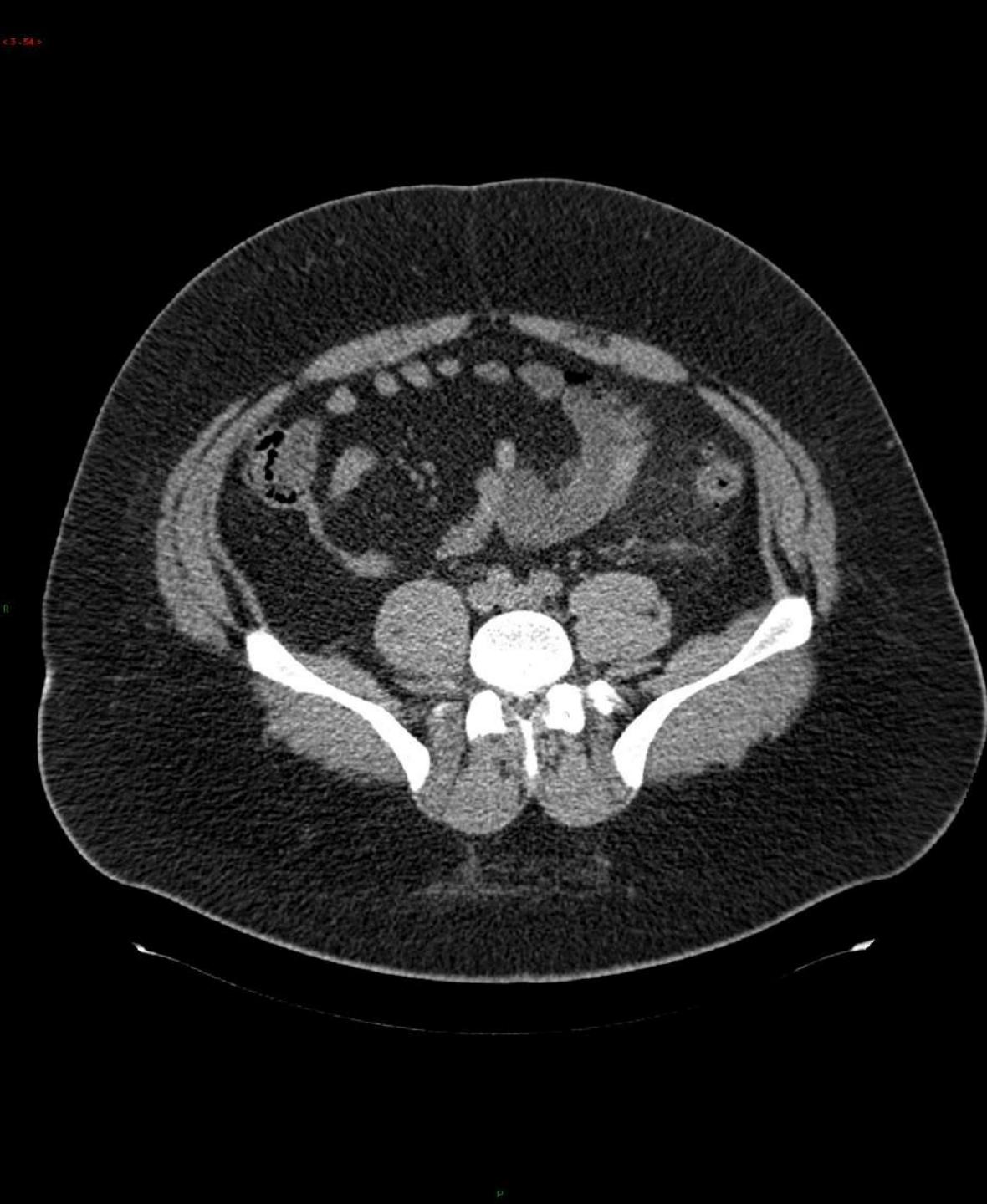
Andrew Ng

3D data



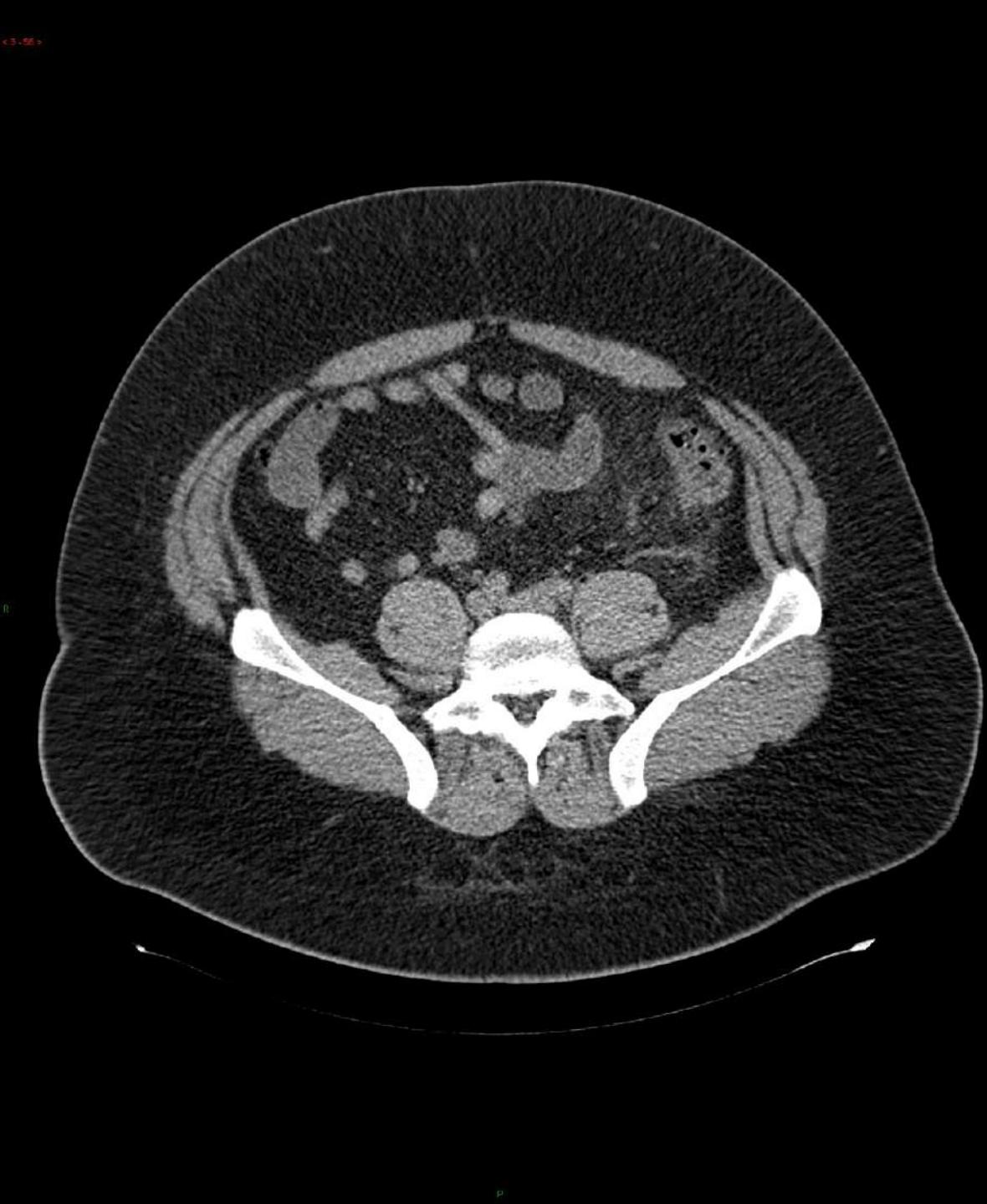
Andrew Ng

# 3D data

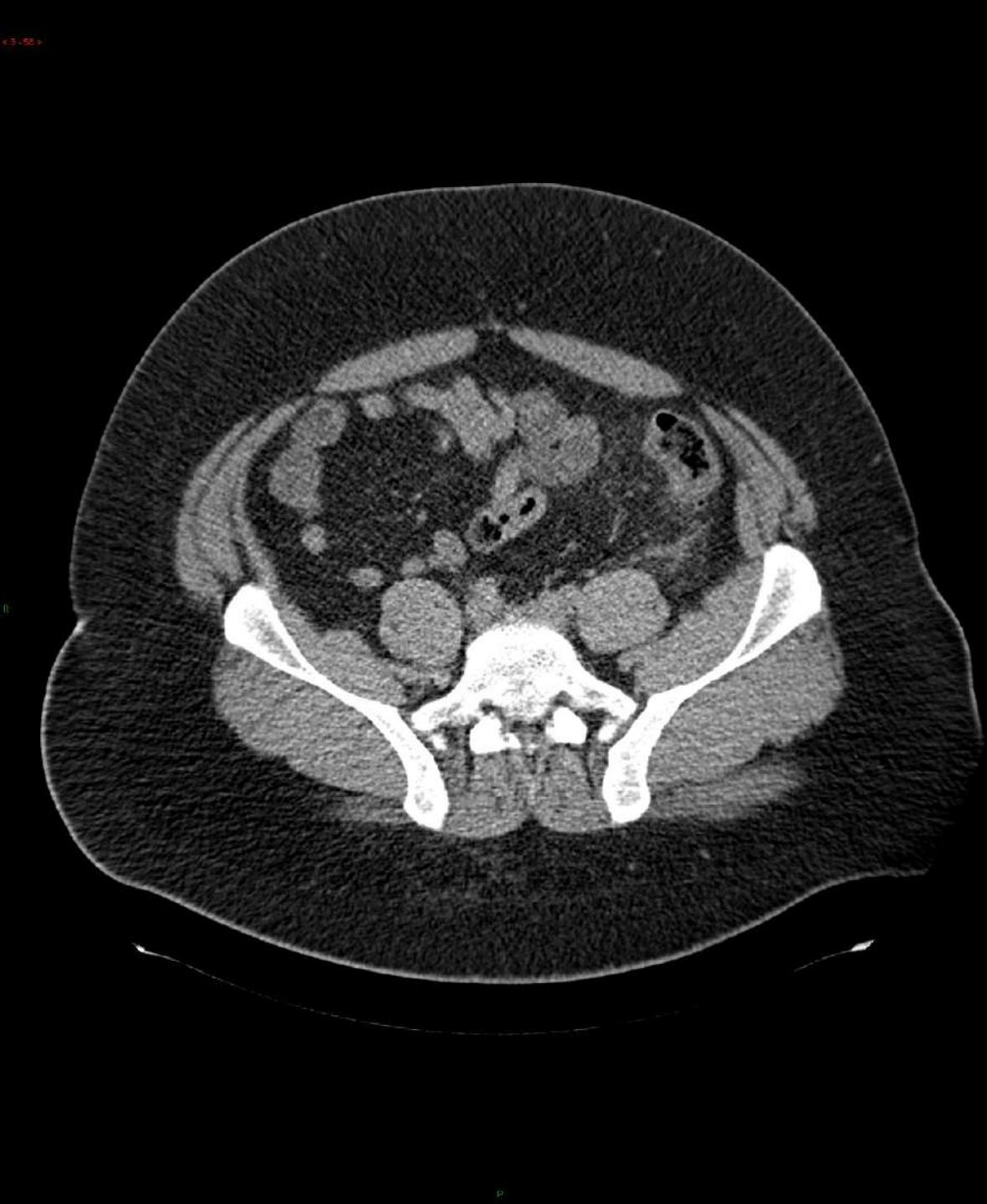


Andrew Ng

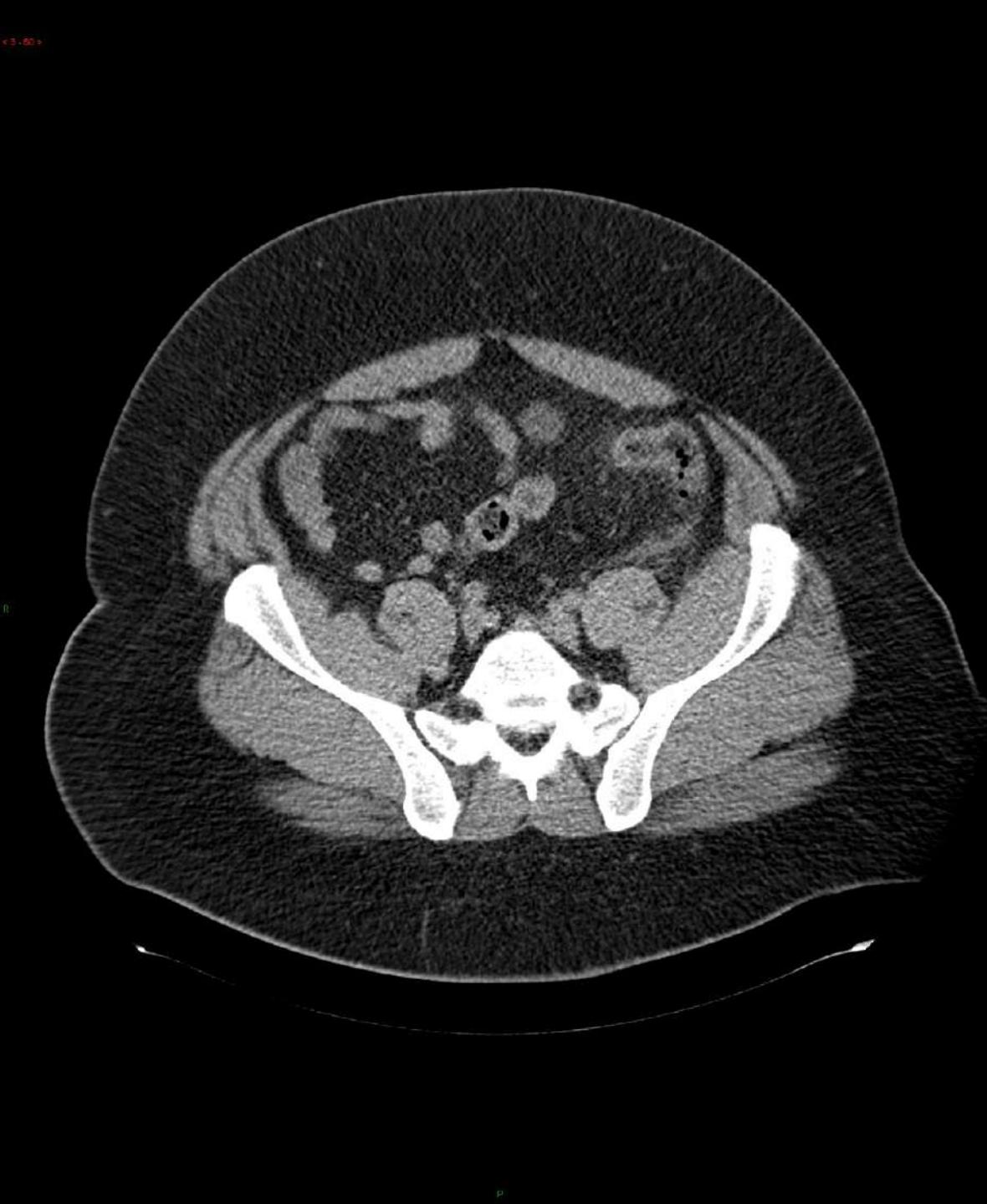
# 3D data



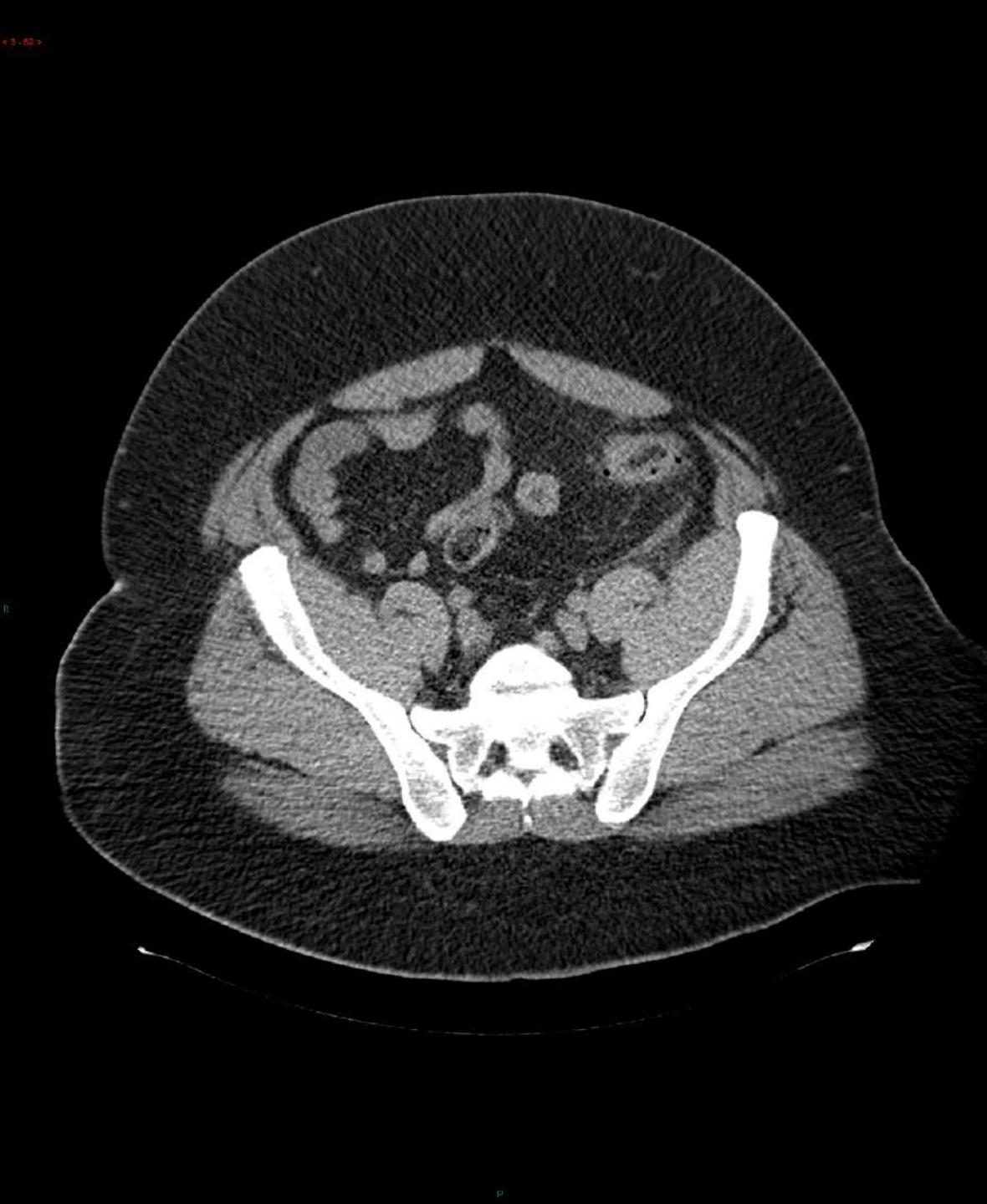
# 3D data



# 3D data



# 3D data



Andrew Ng

# 3D data



Andrew Ng

# 3D data



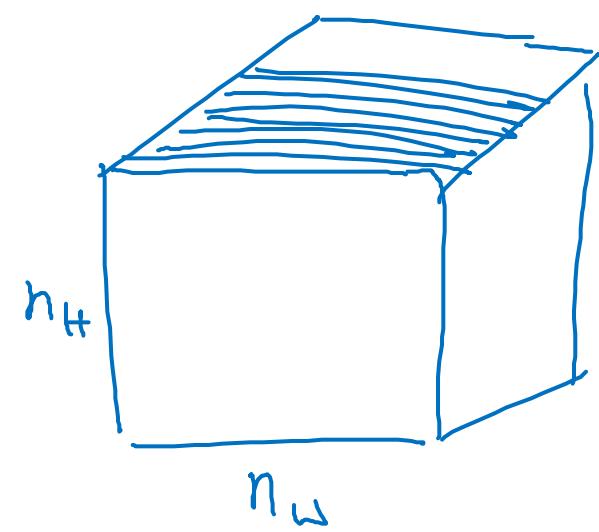
Andrew Ng

# 3D data



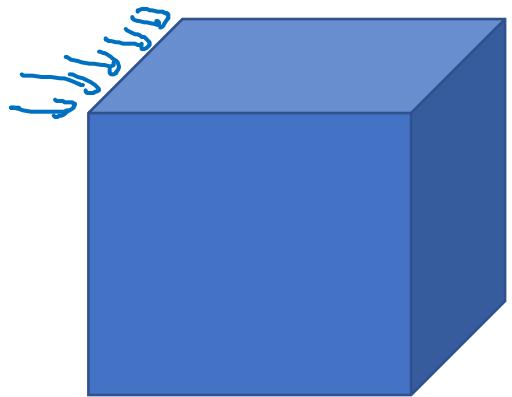
Andrew Ng

# 3D data

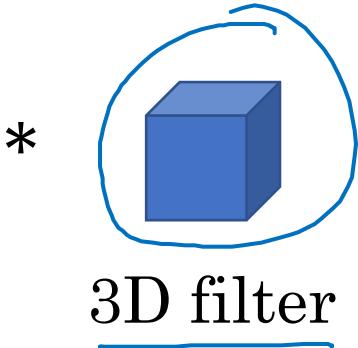


Andrew Ng

# 3D convolution



3D volume



$$\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ \underbrace{4 \times 4 \times 4}_{\text{Input}} \times 1 \\ * \quad \underbrace{5 \times 5 \times 5}_{\text{Filter}} \times 1 \quad 16 \text{ filters.} \\ \rightarrow 10 \times 10 \times 10 \times 16 \\ * \quad \underbrace{5 \times 5 \times 5}_{\text{Stride}} \times 16 \\ \rightarrow 6 \times 6 \times 6 \times 32 \end{array}$$

The diagram illustrates the computation of a 3D convolution. It starts with an input volume of size  $4 \times 4 \times 4$ , which is multiplied by a 3D filter of size  $5 \times 5 \times 5$ . This results in 16 feature maps of size  $10 \times 10 \times 10$ . A second convolution step uses a stride of 2 (indicated by the underbrace) on these 16 maps, resulting in 32 feature maps of size  $6 \times 6 \times 6$ .

# 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

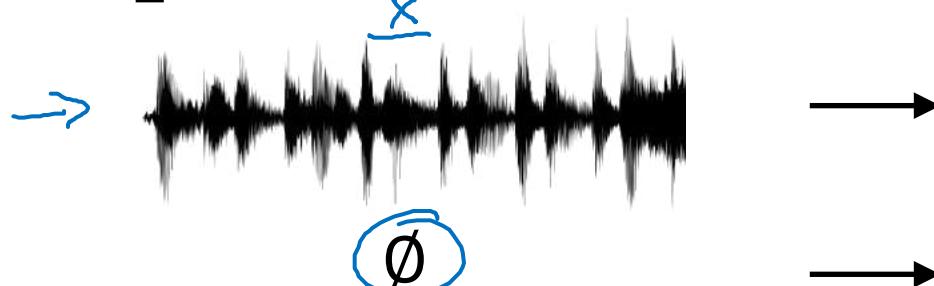
# Recurrent Neural Networks

---

## Why sequence models?

# Examples of sequence data

Speech recognition



$y$   
“The quick brown fox jumped  
over the lazy dog.”

Music generation



Sentiment classification

“There is nothing to like  
in this movie.”



DNA sequence analysis → AGCCCCTGTGAGGAAC TAG



AG $\textcolor{red}{CCCCTGTGAGGAAC}$  TAG

Machine translation

Voulez-vous chanter avec  
moi?



Do you want to sing with  
me?

Video activity recognition



Running

Name entity recognition

Yesterday, Harry Potter  
met Hermione Granger.



Yesterday, **Harry Potter**  
met **Hermione Granger**.

Andrew Ng



deeplearning.ai

# Recurrent Neural Networks

---

## Notation

# Motivating example

NLP

x:

(Harry Potter) and (Hermione Granger) invented a new spell.

$\rightarrow \underline{x^{<1>}}$   $x^{<2>}$   $x^{<3>}$  - - - -  $x^{<t>}$  - - - -  $x^{<q>}$

$$T_x = q$$

$\rightarrow y:$

| | 0 | | 0 0 0 0  
 $y^{<1>}$   $y^{<2>}$   $y^{<3>}$  - - - -  $y^{<q>}$

$$T_y = q$$

$x^{(i)<t>}$

$y^{(i)<t>}$

$$T_x^{(i)} = q$$

$$T_y^{(i)}$$

15

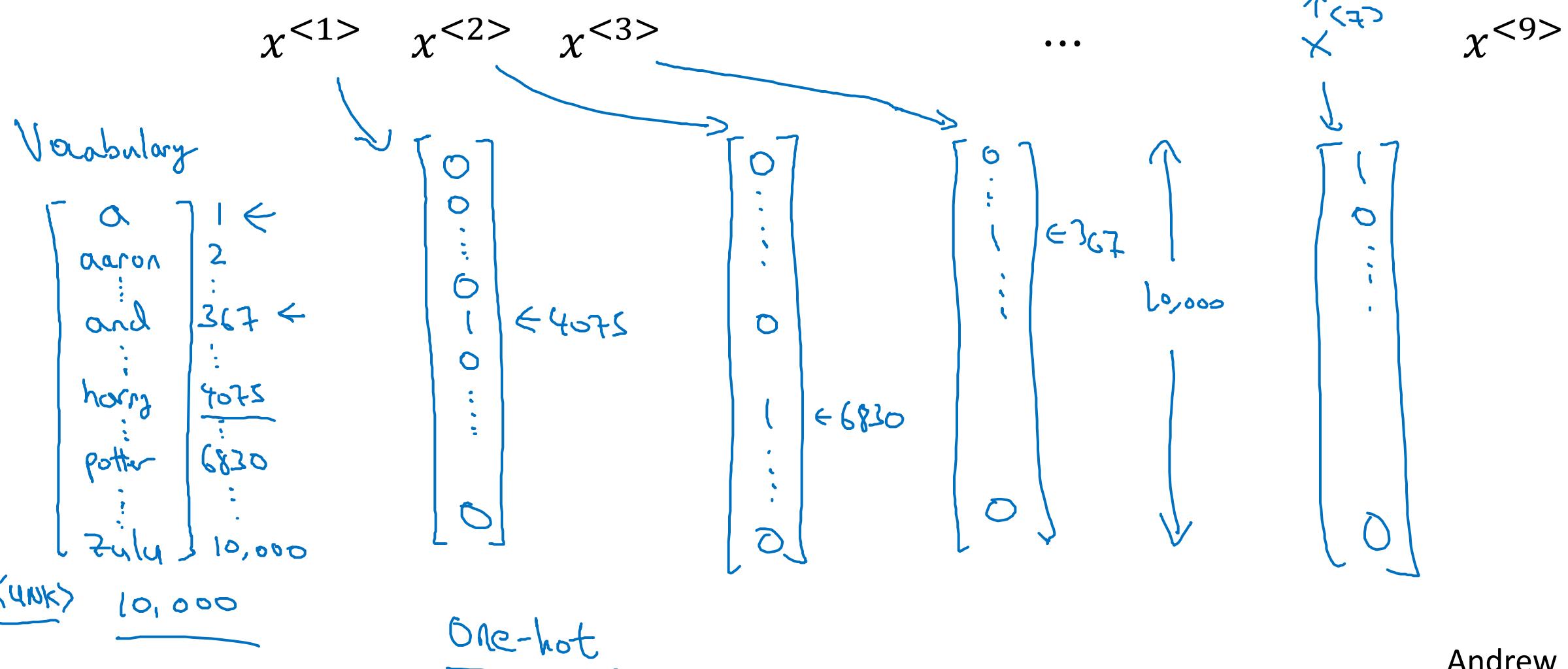
# Representing words

$$x^{<\leftrightarrow>} \quad x \rightarrow y$$

$(x, y)$

x:

Harry Potter and Hermione Granger invented a new spell.



# Representing words

x: Harry Potter and Hermione Granger invented a new spell.

$x^{<1>} \quad x^{<2>} \quad x^{<3>} \quad \dots \quad x^{<9>}$

And = 367  
Invented = 4700  
A = 1  
New = 5976  
Spell = 8376  
Harry = 4075  
Potter = 6830  
Hermione = 4200  
Gran... = 4000



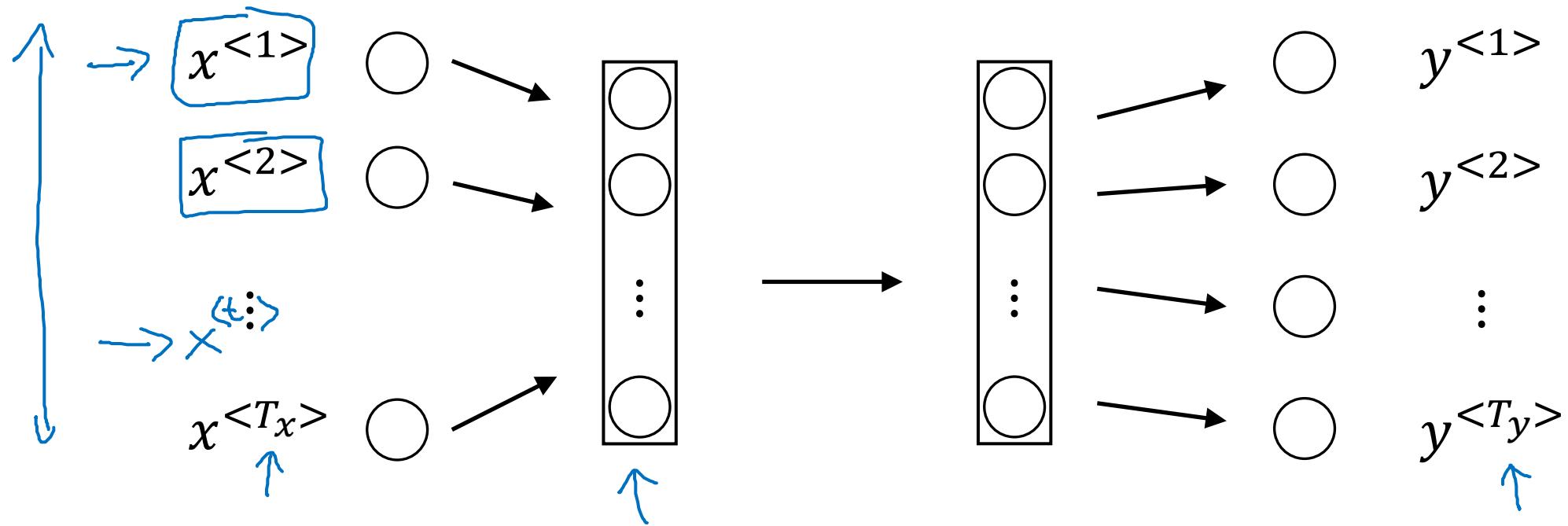
deeplearning.ai

# Recurrent Neural Networks

---

## Recurrent Neural Network Model

# Why not a standard network?

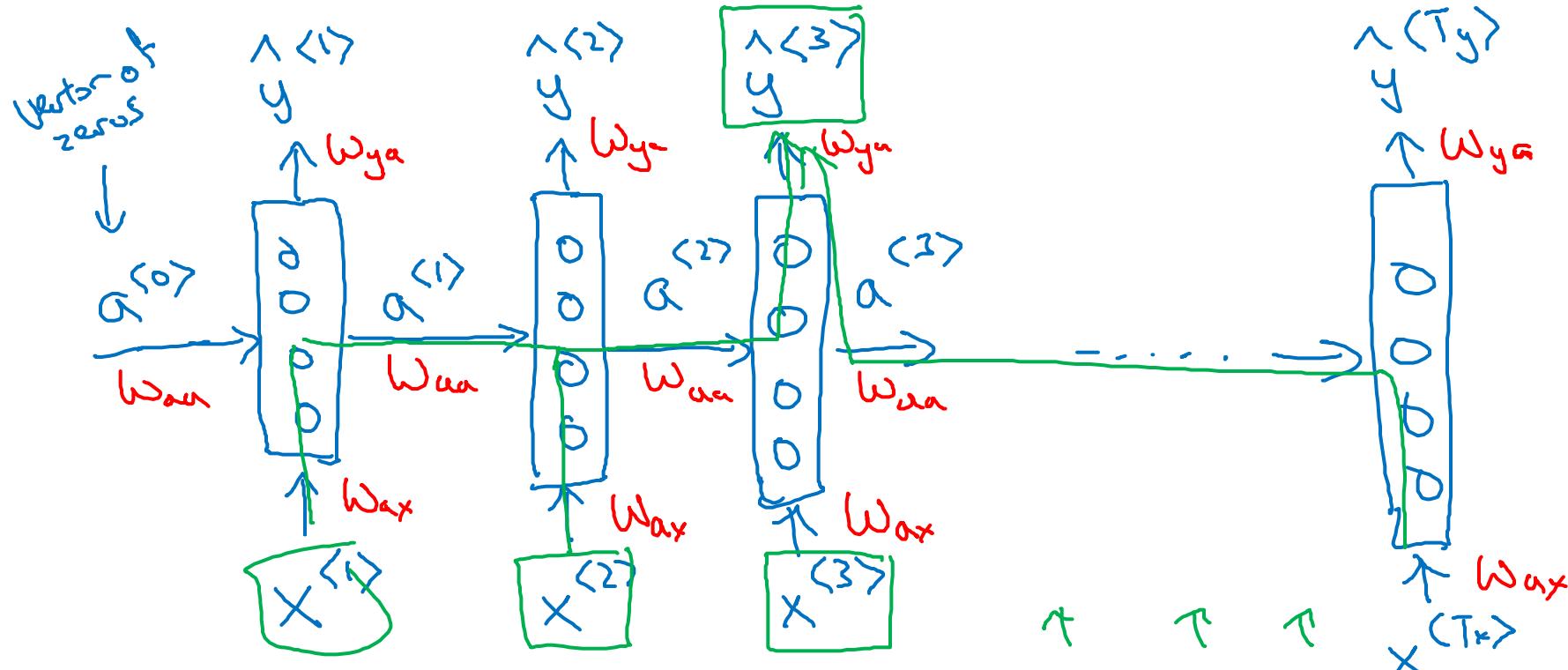


## Problems:

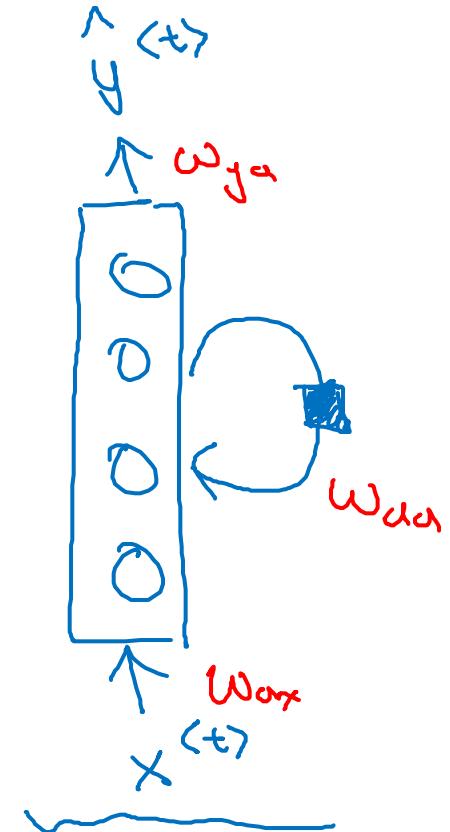
- - Inputs, outputs can be different lengths in different examples.
- - Doesn't share features learned across different positions of text.

# Recurrent Neural Networks

$$\overline{T}_x = \overline{T}_y$$



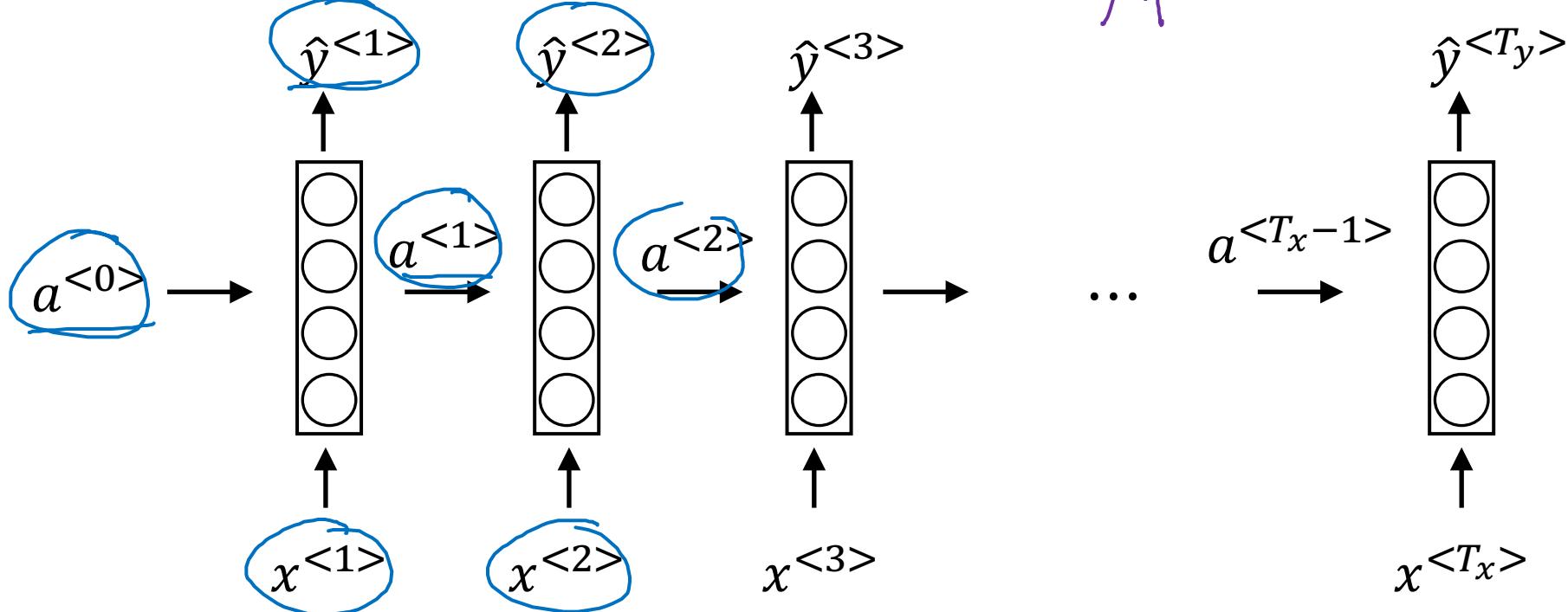
Bidirectional RNN (BRNN)



He said, "Teddy Roosevelt was a great President."

He said, "Teddy bears are on sale!"

# Forward Propagation



$$a^{<0>} = \vec{0}.$$

$$\underline{a^{<t>} = g_t(W_a a^{<t-1>} + W_x x^{<t>} + b_a)} \leftarrow \text{tanh / ReLU}$$

$$\underline{\hat{y}^{<t>} = g_t(W_y a^{<t>} + b_y)} \leftarrow \text{Sigmoid}$$

$$a^{<t>} = g(W_a a^{<t-1>} + W_x x^{<t>} + b_a)$$

$$\hat{y}^{<t>} = g(W_y a^{<t>} + b_y)$$

# Simplified RNN notation

$$a^{(t)} = g(W_{aa}a^{(t-1)} + W_{ax}x^{(t)} + b_a)$$

Dimensions:  
 $W_{aa}$ : (100, 100)  
 $W_{ax}$ : (100, 10,000)

$$\hat{y}^{(t)} = g(W_{ya}a^{(t)} + b_y)$$

$$y^{(t)} = g(W_y a^{(t)} + b_y)$$

Dimensions:  
 $W_y$ : (10, 100)

$$a^{(t)} = g(W_a [a^{(t-1)}, x^{(t)}] + b_a)$$

$$\begin{bmatrix} W_{aa} & W_{ax} \end{bmatrix} = W_a$$

Dimensions:  
 $W_{aa}$ : (100, 100)  
 $W_{ax}$ : (100, 10,000)

$$[a^{(t-1)}, x^{(t)}] = \begin{bmatrix} a^{(t-1)} \\ x^{(t)} \end{bmatrix}$$

Dimensions:  
 $a^{(t-1)}$ : 100  
 $x^{(t)}$ : 10,000  
 $[a^{(t-1)}, x^{(t)}]$ : 10,100

$$[W_{aa}; W_{ax}] \begin{bmatrix} a^{(t-1)} \\ x^{(t)} \end{bmatrix} = W_{aa}a^{(t-1)} + W_{ax}x^{(t)}$$



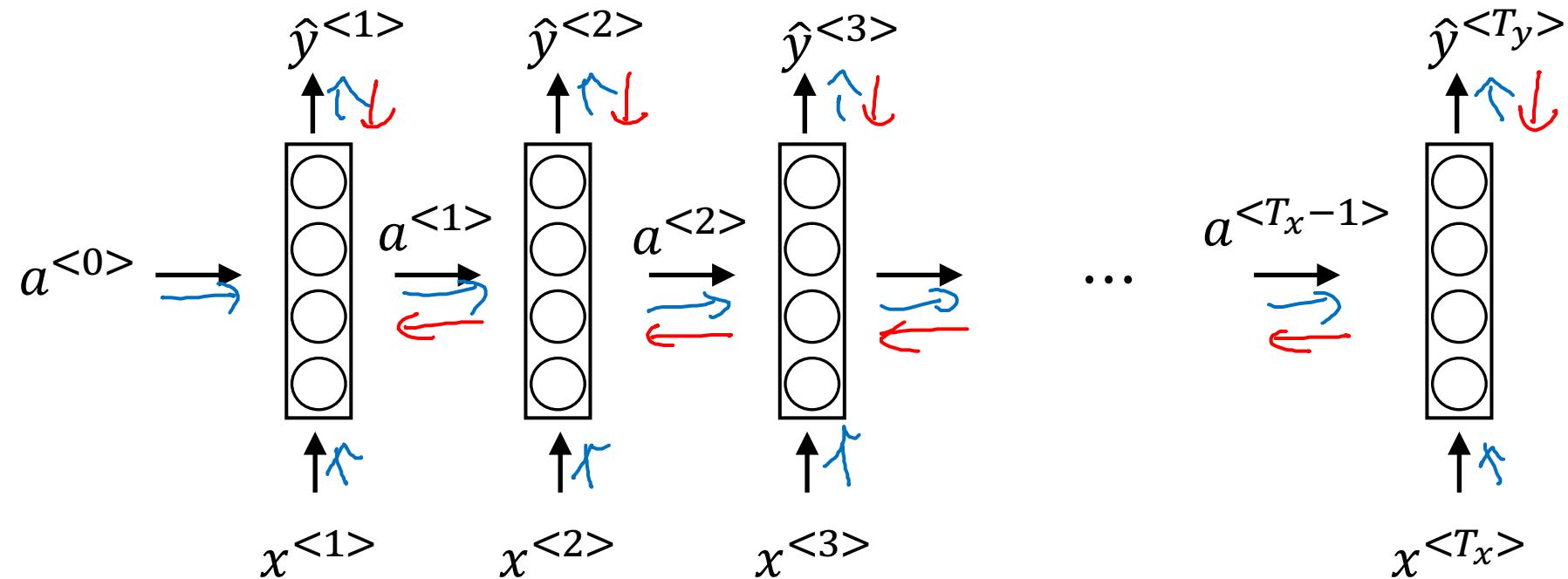
deeplearning.ai

# Recurrent Neural Networks

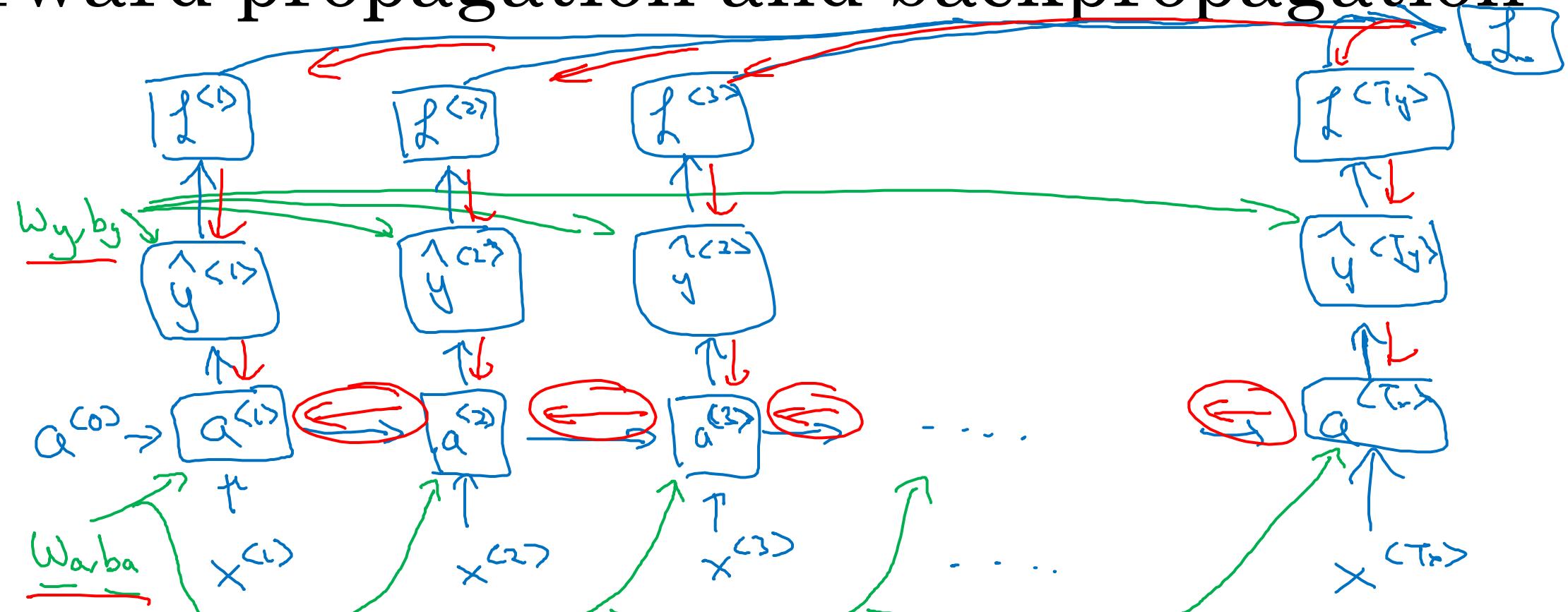
---

## Backpropagation through time

# Forward propagation and backpropagation



# Forward propagation and backpropagation



$$\mathcal{L}^{(t)}(\hat{y}^{(t)}, y^{(t)}) = -y^{(t)} \log \hat{y}^{(t)} - (1-y^{(t)}) \log (1-\hat{y}^{(t)})$$

$$\mathcal{L}(\hat{y}, y) = \sum_{t=1}^{T_y} \mathcal{L}^{(t)}(\hat{y}^{(t)}, y^{(t)})$$

Backpropagation through time



deeplearning.ai

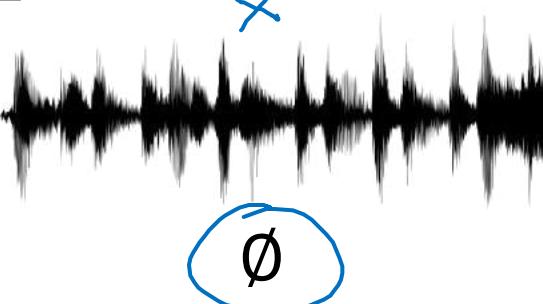
# Recurrent Neural Networks

---

Different types  
of RNNs

# Examples of sequence data

Speech recognition



$T_x$   $T_y$

$y$

“The quick brown fox jumped over the lazy dog.”

Music generation



Sentiment classification

“There is nothing to like  
in this movie.”



DNA sequence analysis

AGCCCCTGTGAGGAAC TAG



AG~~CCCCTGTGAGGAAC~~ TAG

Machine translation

Voulez-vous chanter avec  
moi?



Do you want to sing with  
me?

Video activity recognition



Running

Name entity recognition

Yesterday, Harry Potter  
met Hermione Granger.

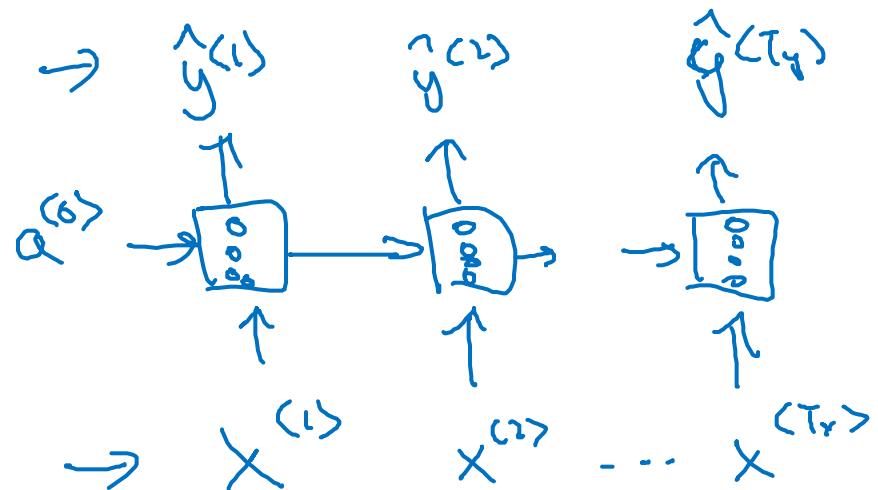


Yesterday, **Harry Potter**  
met **Hermione Granger**.

Andrew Ng

# Examples of RNN architectures

$$T_x = T_y$$

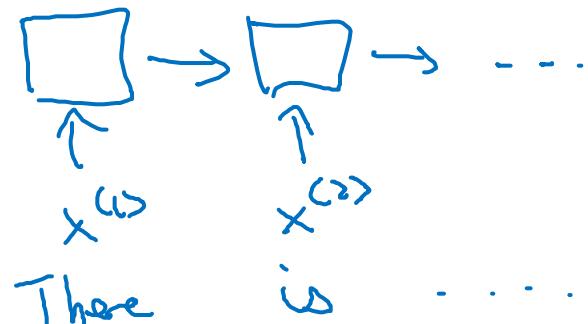


Many-to-many

Sentiment classification

$x = \text{text}$

$y = 0/1 \quad 1 \dots 5$

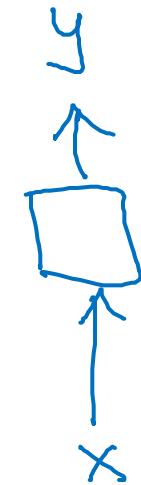
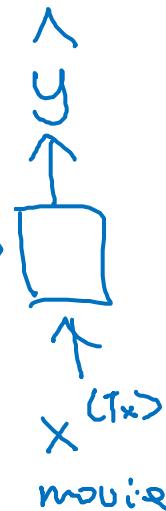


These

is

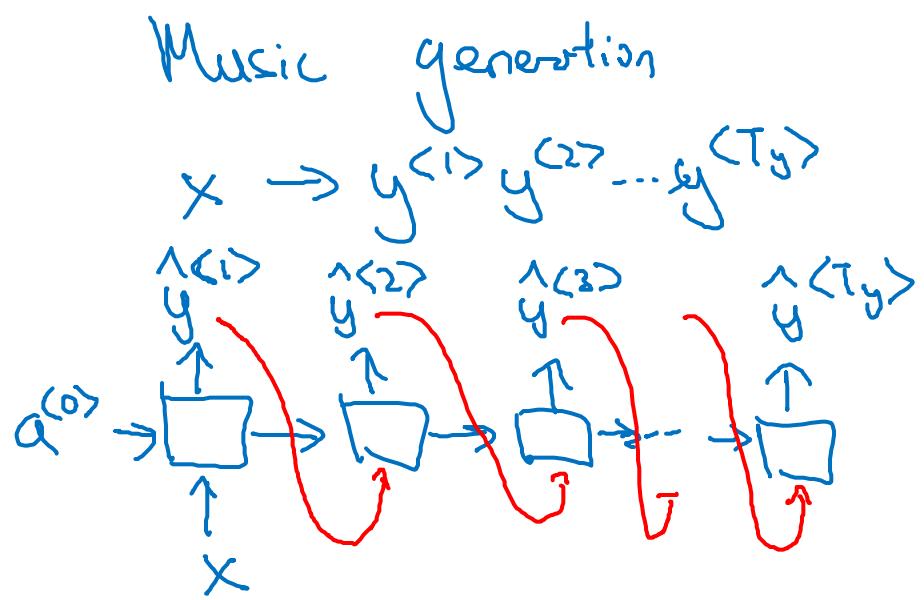
.....

Many - to - one



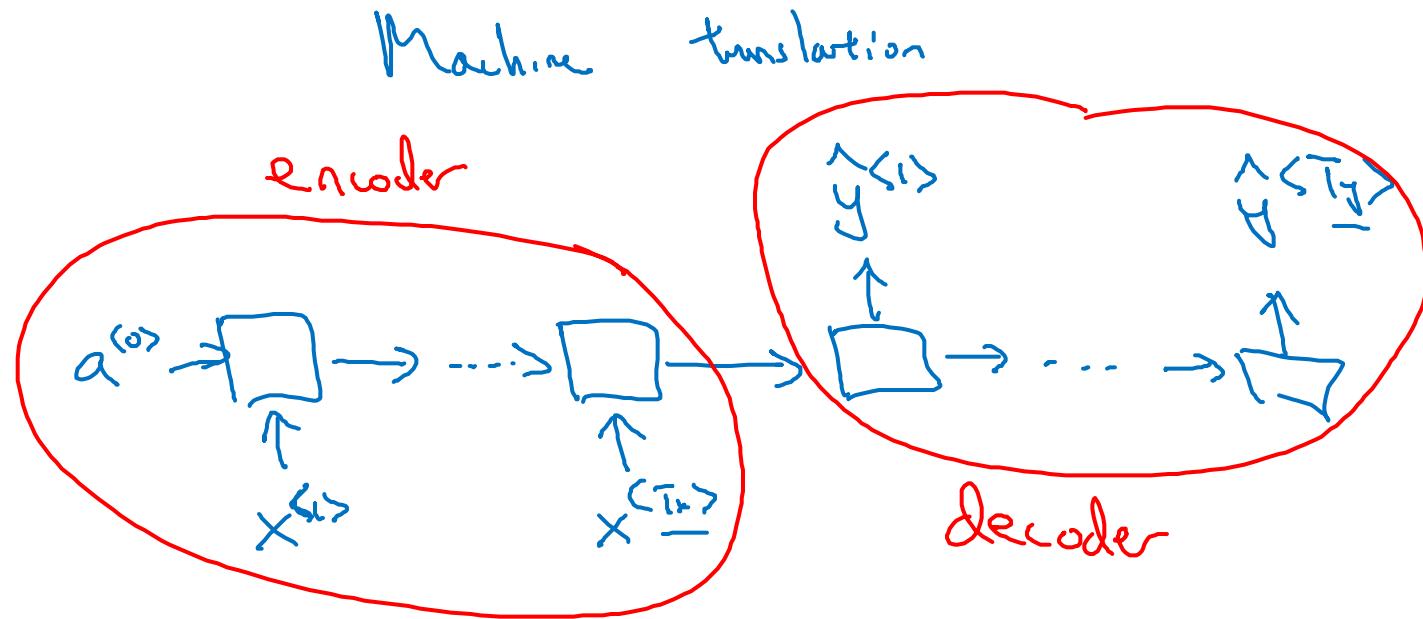
One - to -  
one

# Examples of RNN architectures



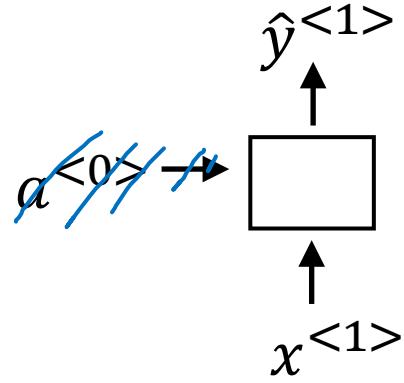
One-to-many

$$x = \phi$$

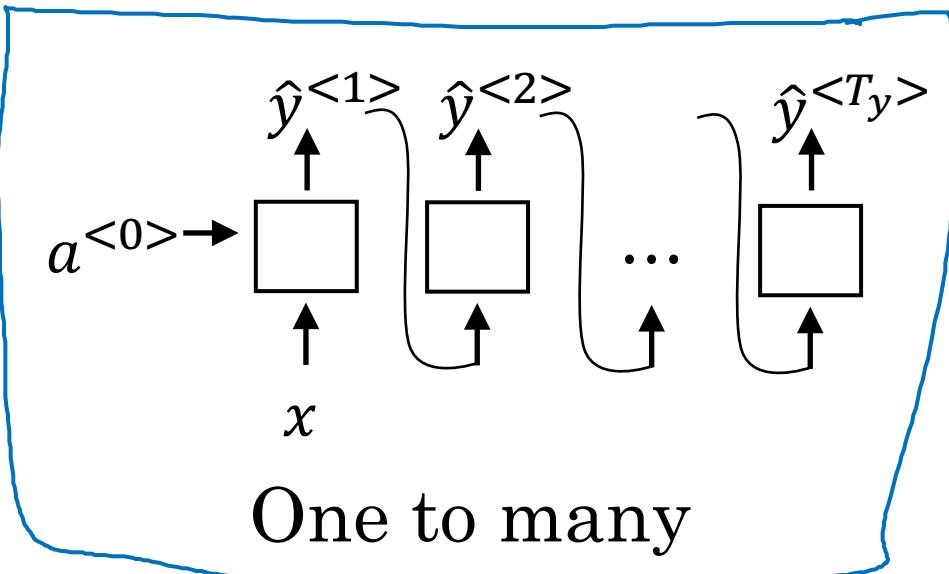


Many - to - many

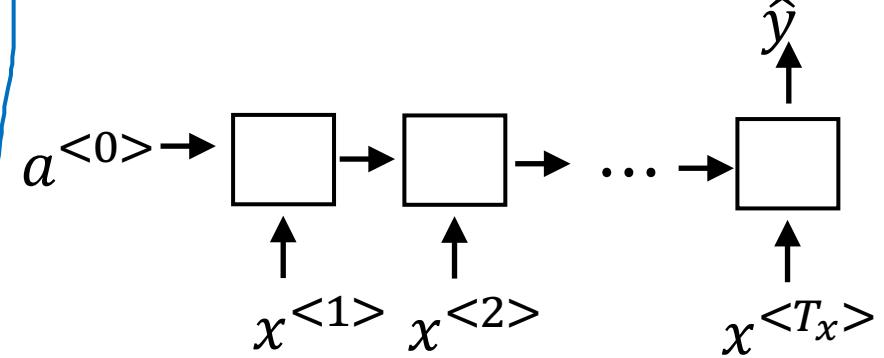
# Summary of RNN types



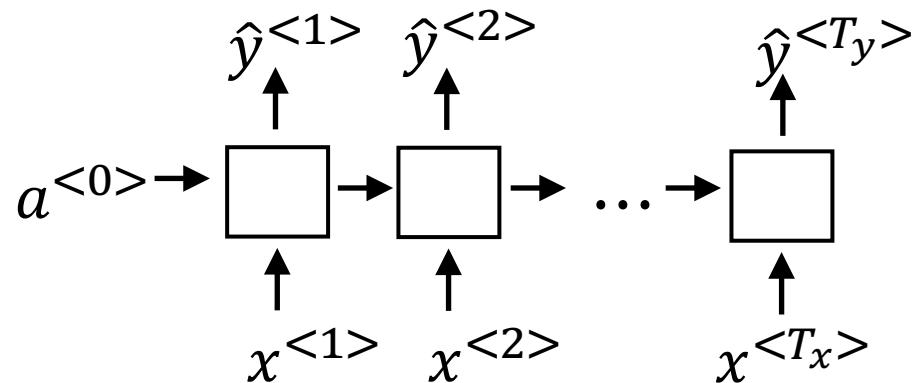
One to one



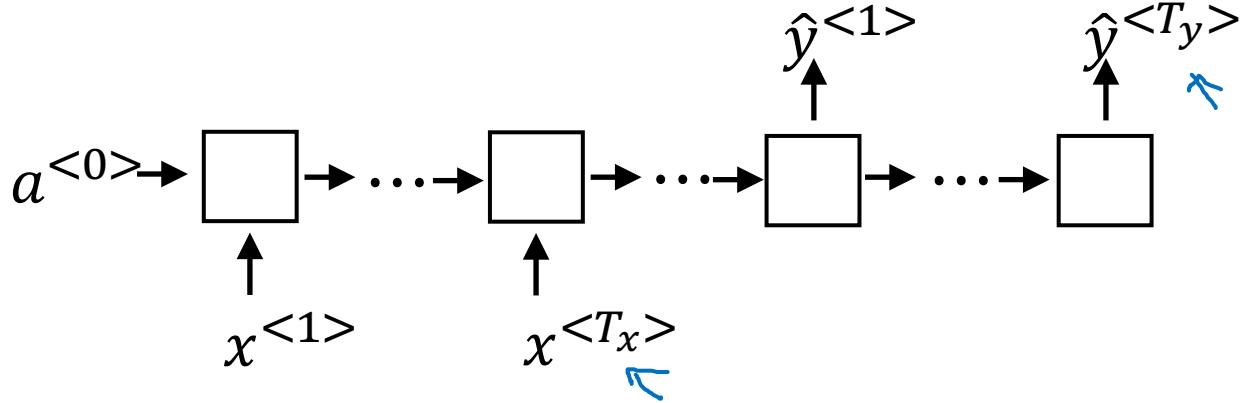
One to many



Many to one



Many to many



Many to many



deeplearning.ai

# Recurrent Neural Networks

---

## Language model and sequence generation

# What is language modelling?

Speech recognition

The apple and pair salad.

→ The apple and pear salad.

$$P(\text{The apple and pair salad}) = 3.2 \times 10^{-3}$$

$$P(\text{The apple and pear salad}) = 5.7 \times 10^{-10}$$

$$P(\text{Sentence}) = ?$$

$$P(y^{(1)}, y^{(2)}, \dots, y^{(T)})$$

# Language modelling with an RNN

Training set: large corpus of english text.

Tokenize

Cats average 15 hours of sleep a day.  $\downarrow <\text{EOS}>$

$y^{<1>}$        $y^{<2>}$        $y^{(3)}$

$x^{<t>} = y^{<t-1>}$

...

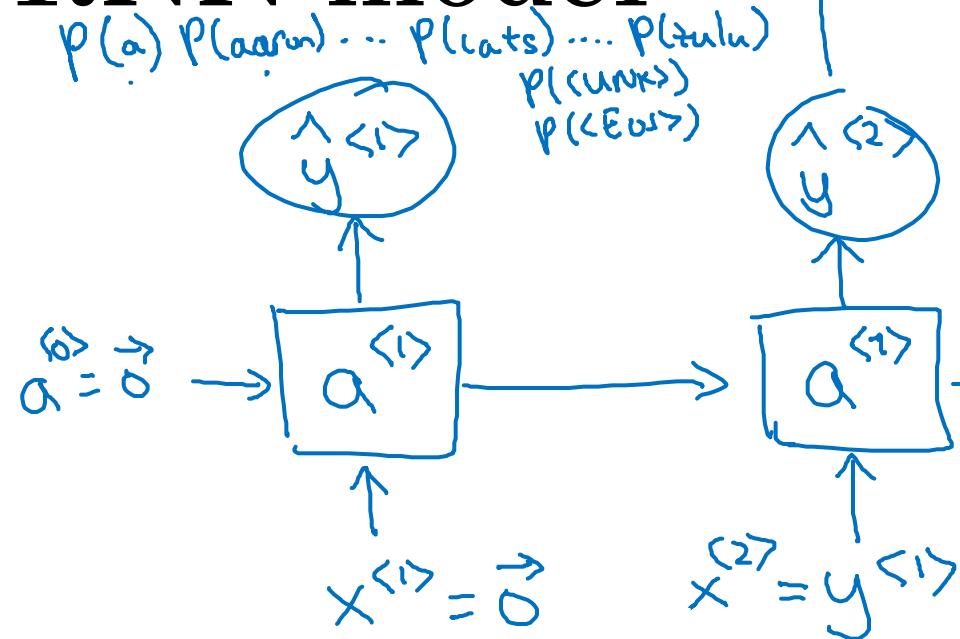
$y^{(8)}$        $y^{(9)}$

The Egyptian ~~Mau~~ is a bread of cat.  $<\text{EOS}>$

$<\text{UNK}>$

10,000

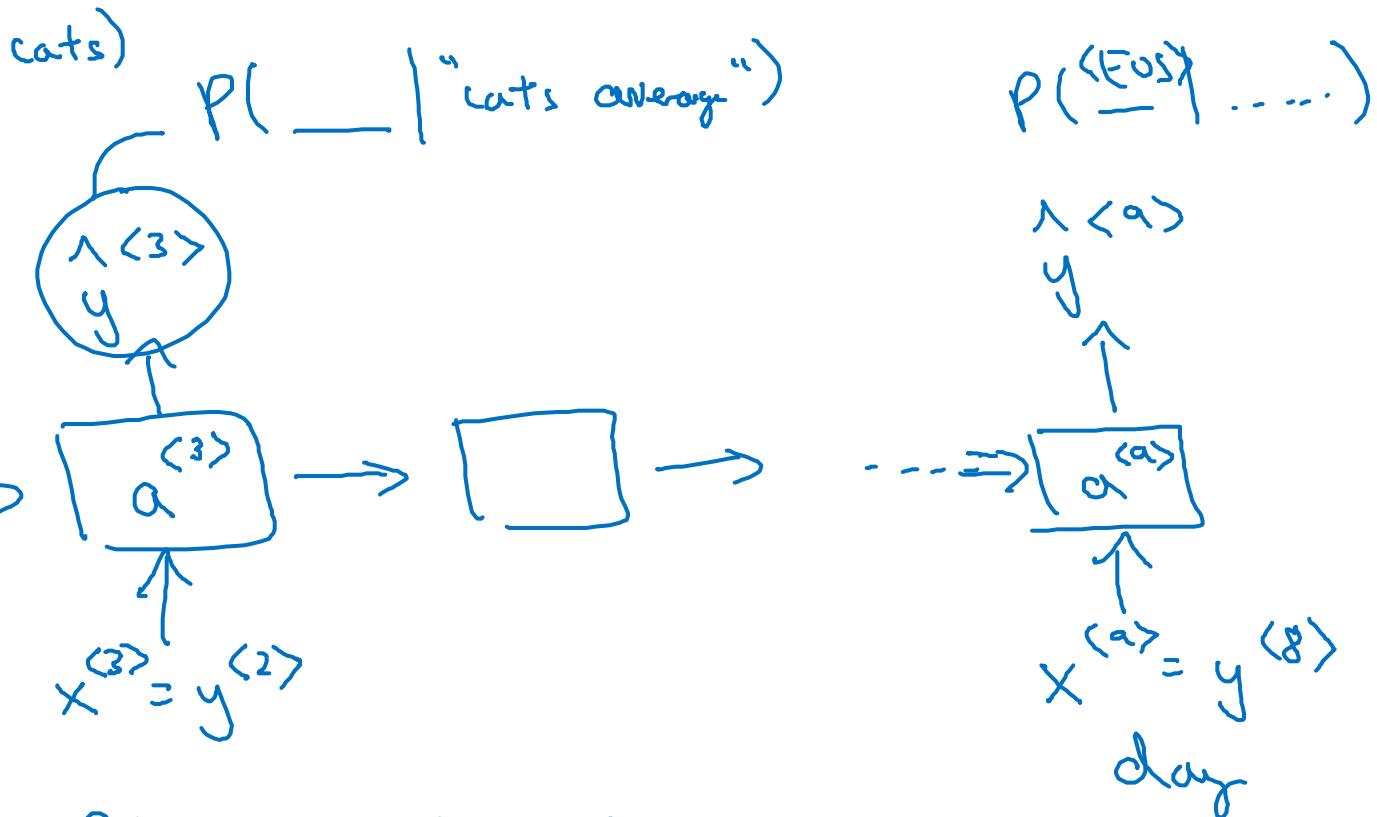
# RNN model



$\rightarrow$  Cats average 15 hours of sleep a day.  $\underline{\text{Cats}}$   $\underline{\text{Average}}$   $\underline{\text{hours}}$   $\underline{\text{day}}$

$$\mathcal{L}(\hat{y}^{(t)}, y^{(t)}) = - \sum_i y_i^{(t)} \log \hat{y}_i^{(t)}$$

$$\mathcal{L} = \sum_t \mathcal{L}^{(t)}(\hat{y}^{(t)}, y^{(t)})$$



$$P(y^{(1)}, y^{(2)}, y^{(3)}) \leftarrow \\ = \frac{p(y^{(1)}) p(y^{(2)} | y^{(1)})}{p(y^{(3)} | y^{(1)}, y^{(2)})}$$



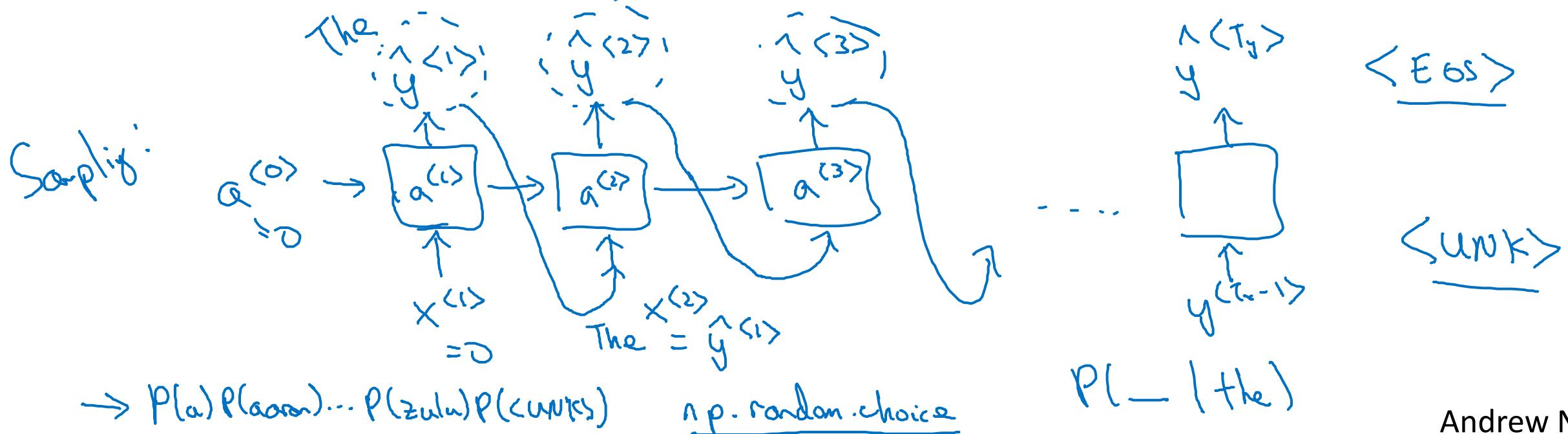
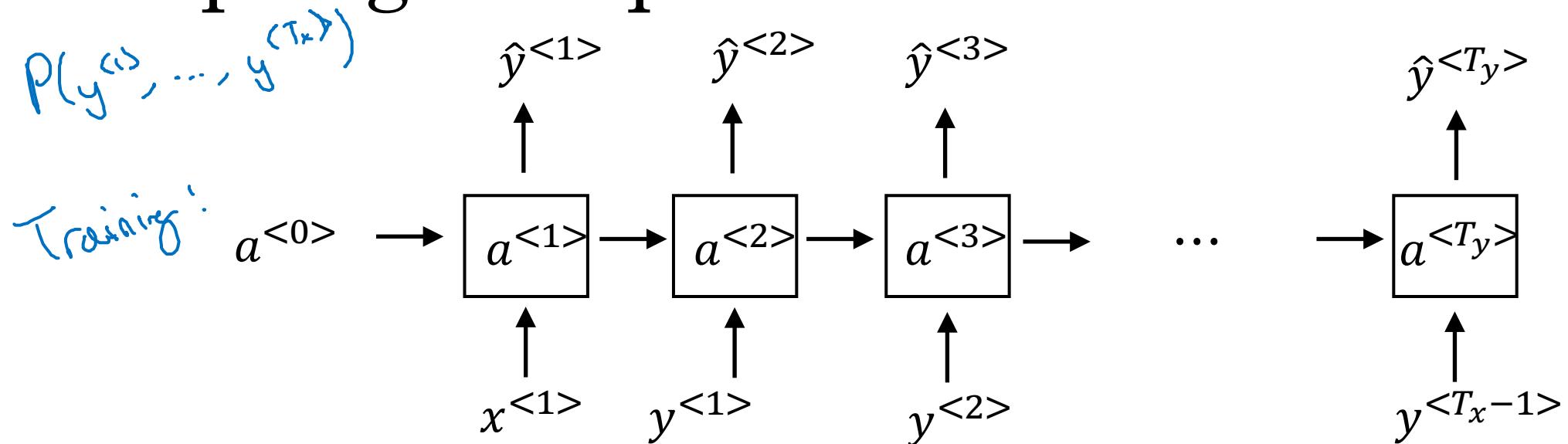
deeplearning.ai

# Recurrent Neural Networks

---

Sampling novel  
sequences

# Sampling a sequence from a trained RNN



# Character-level language model

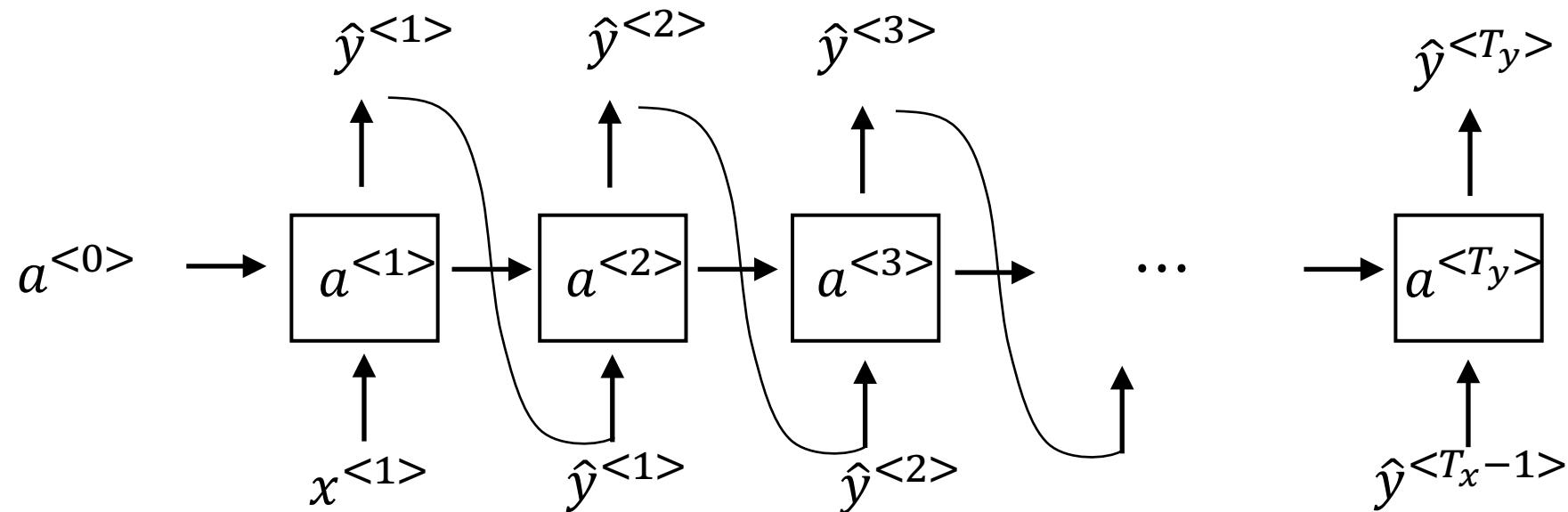
→ Vocabulary = [a, aaron, ..., zulu, <UNK>] ↪

$\rightarrow \text{Vocabulary} = [a, b, c, \dots, z, \cup, \circ, \rightarrow, ;, 0, \dots, 9, A, \dots, Z]$

$$y^{(0)} - y^{(1)} = \frac{y^{(2)}}{y^{(3)}}$$

Cat average  
↑ ↑ ↑ ↑ . . .

Man



# Sequence generation

## News

President enrique peña nieto, announced  
sench's sulk former coming football langston  
paring.

“I was not at all surprised,” said hich langston.

“Concussion epidemic”, to be examined. ←

The gray football the told some and this has on  
the uefa icon, should money as.

## Shakespeare

The mortal moon hath her eclipse in love.  
And subject of this thou art another this fold.

When lesser be my love to me see sabl’s.  
For whose are ruse of mine eyes heaves.



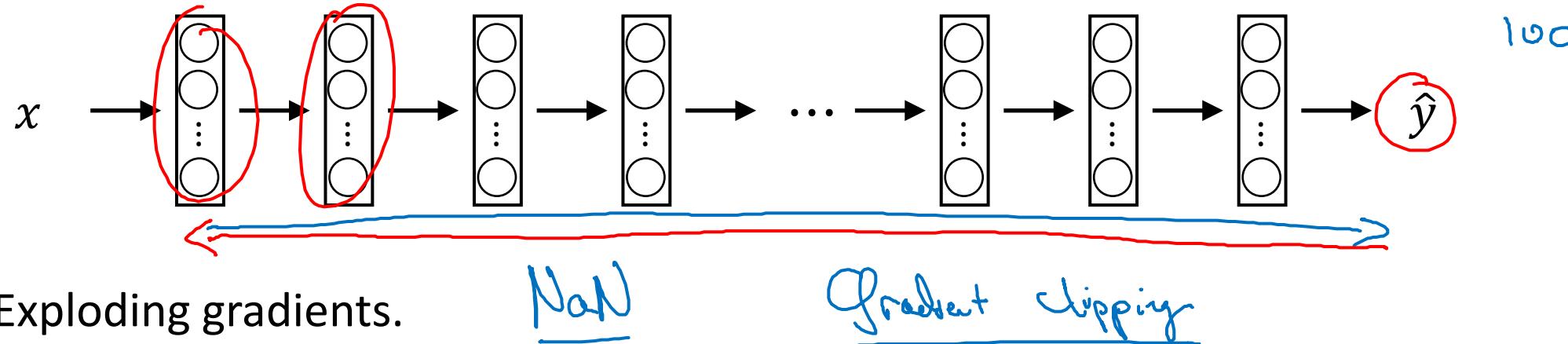
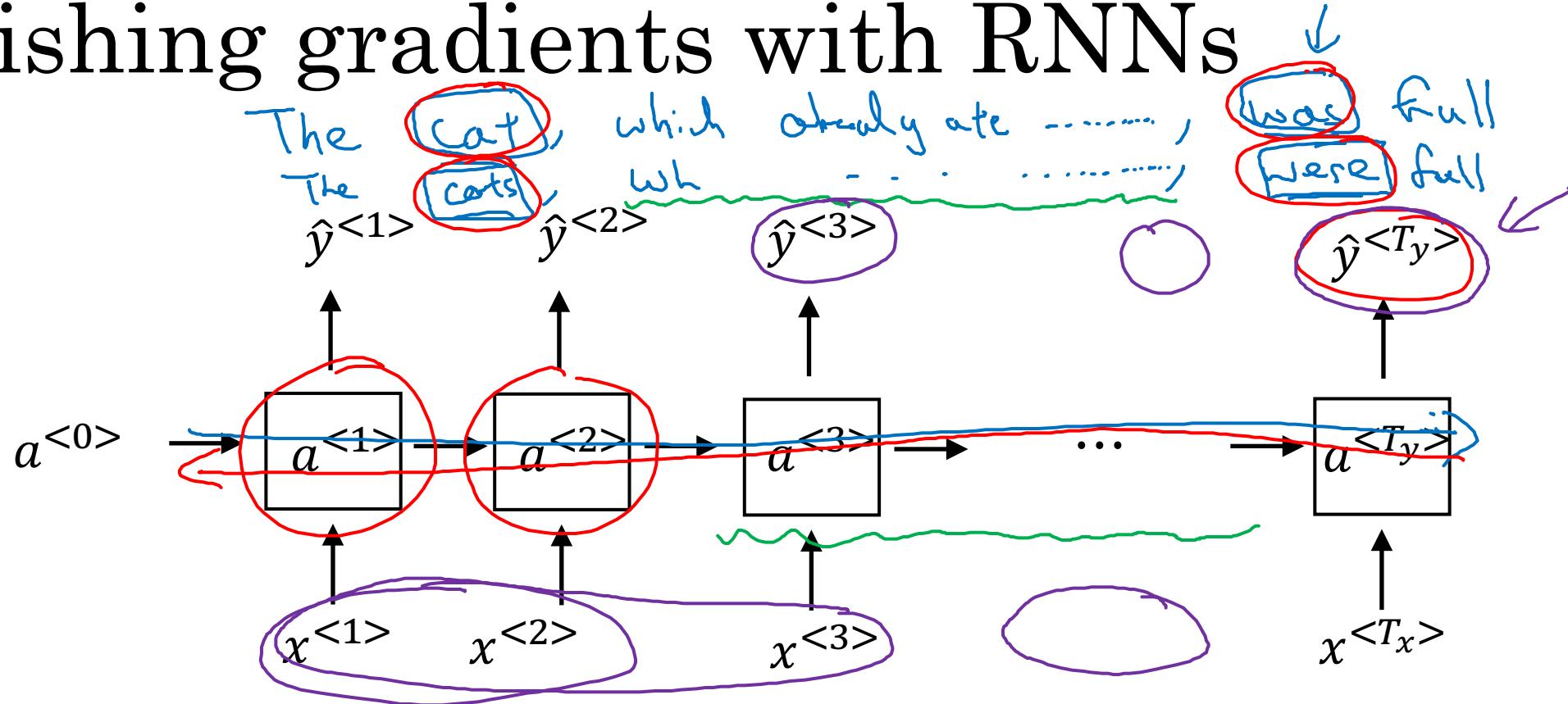
deeplearning.ai

# Recurrent Neural Networks

---

## Vanishing gradients with RNNs

# Vanishing gradients with RNNs





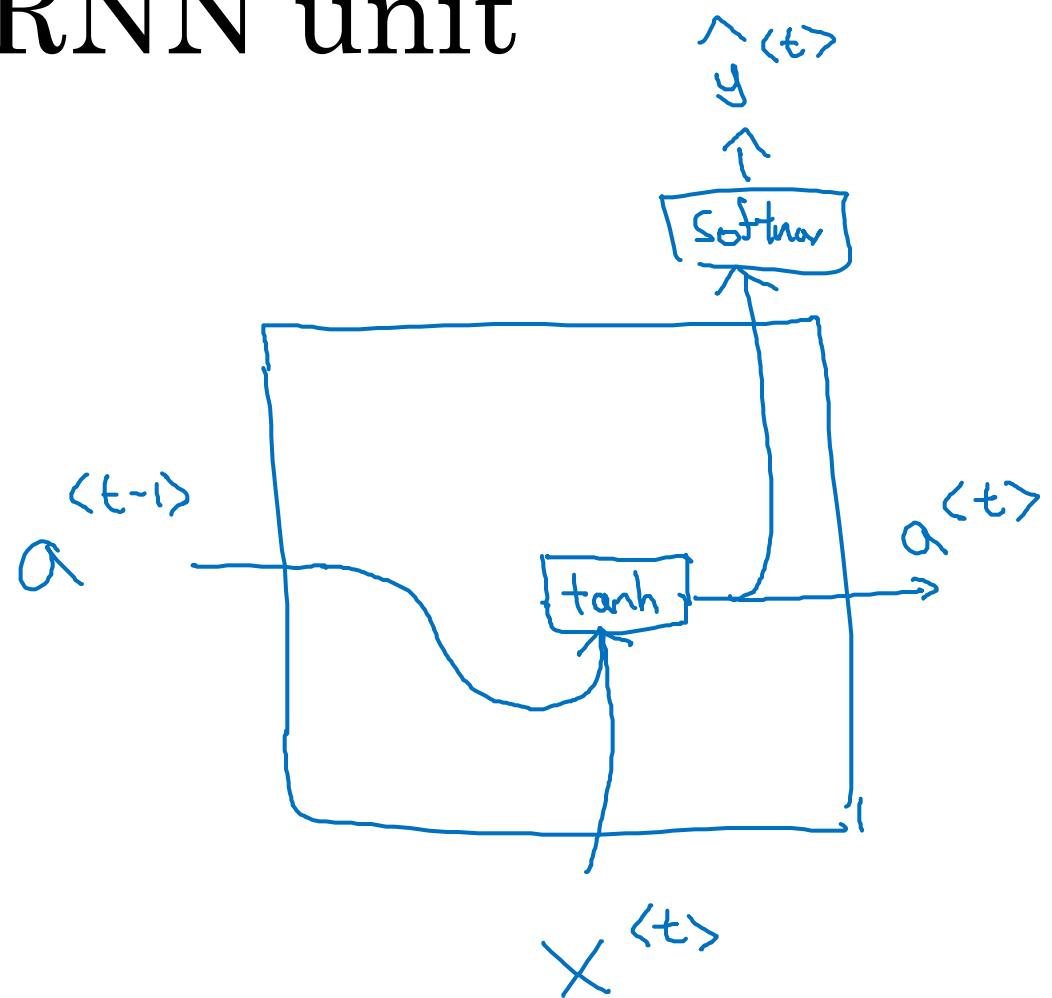
deeplearning.ai

# Recurrent Neural Networks

---

## Gated Recurrent Unit (GRU)

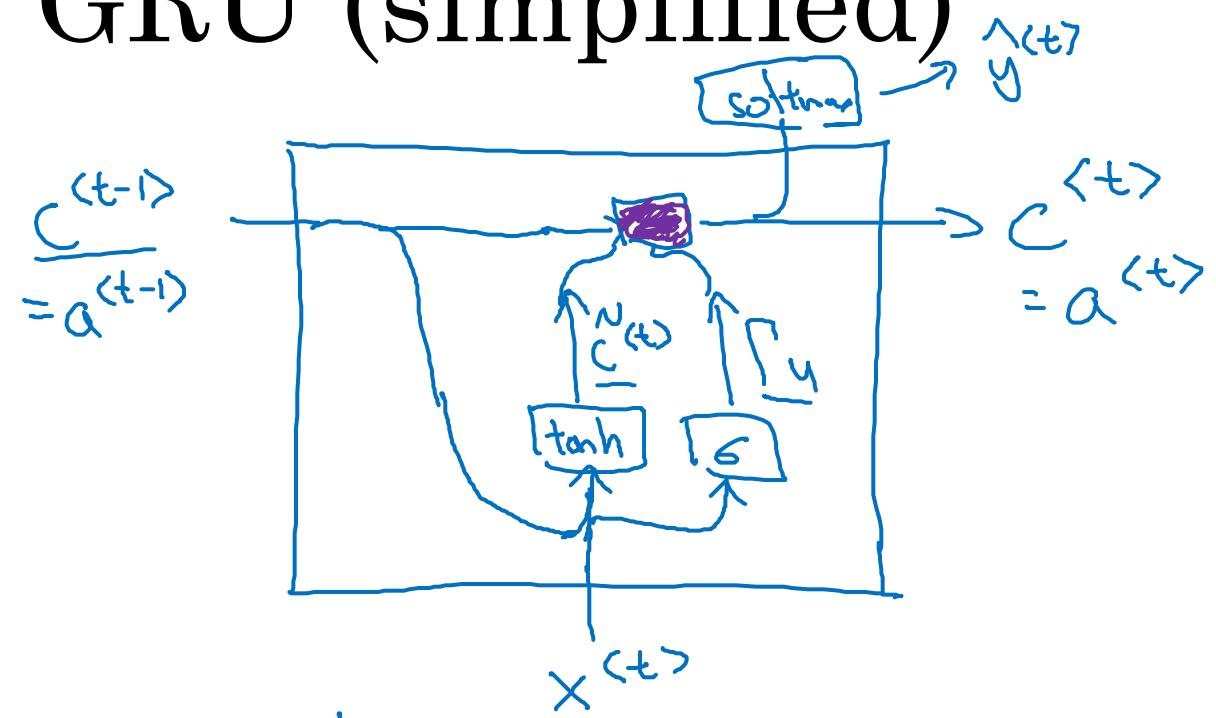
# RNN unit



$$\underline{a}^{(t)} = g(W_a[\underline{a}^{(t-1)}, \underline{x}^{(t)}] + b_a)$$

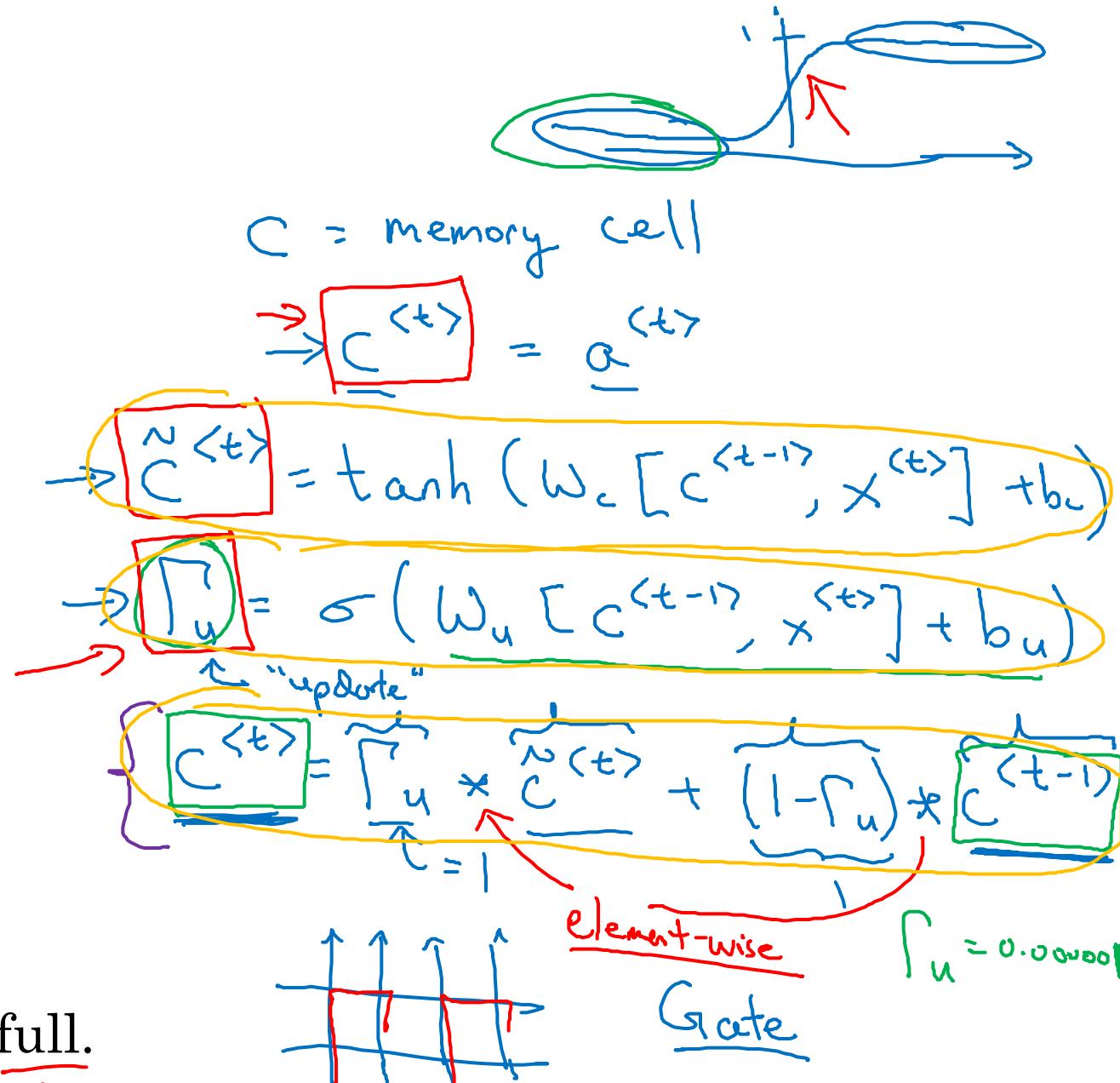
Below the equation, there is a wavy line representing the hidden state  $a^{(t)}$ . A blue bracket under the wavy line indicates its dimensionality. An arrow labeled "tanh" points to the  $\text{tanh}$  function in the diagram above.

# GRU (simplified)



$\Gamma_u = 1$     $\Gamma_a = 0$     $\Gamma_n = 0$     $\Gamma_c = 0$    ...    $\Gamma_u = 1$

The cat, which already ate ..., was full.



[Cho et al., 2014. On the properties of neural machine translation: Encoder-decoder approaches]

[Chung et al., 2014. Empirical Evaluation of Gated Recurrent Neural Networks on Sequence Modeling]

Andrew Ng

# Full GRU

$$\tilde{h} \quad \tilde{c}^{<t>} = \tanh(W_c[\tilde{c}_r^{<t-1>}, \underline{x}^{<t>}] + b_c)$$

$$u \quad \Gamma_u = \sigma(W_u[c^{<t-1>}, \underline{x}^{<t>}] + b_u)$$

$$r \quad \Gamma_r = \sigma(W_r[c^{<t-1>}, \underline{x}^{<t>}] + b_r)$$

LSTM

$$h \quad c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

The cat, which ate already, was full.



deeplearning.ai

# Recurrent Neural Networks

---

LSTM (long short  
term memory) unit

# GRU and LSTM

## GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * \underline{c}^{<t-1>}, x^{<t>}] + b_c)$$

$$\underline{\Gamma_u} = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\underline{\Gamma_r} = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \underline{\Gamma_u} * \tilde{c}^{<t>} + (1 - \underline{\Gamma_u}) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$



## LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\underline{\Gamma_u} = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$\underline{\Gamma_f} = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\underline{\Gamma_o} = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \underline{\Gamma_u} * \tilde{c}^{<t>} + \underline{\Gamma_f} * \underline{c}^{<t-1>}$$

$$a^{<t>} = \underline{\Gamma_o} * c^{<t>}$$

# LSTM units

GRU

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r * c^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[c^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_r = \sigma(W_r[c^{<t-1>}, x^{<t>}] + b_r)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + (1 - \Gamma_u) * c^{<t-1>}$$

$$a^{<t>} = c^{<t>}$$

LSTM

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * c^{<t>}$$

# LSTM in pictures

$$\tilde{c}^{<t>} = \tanh(W_c[a^{<t-1>}, x^{<t>}] + b_c)$$

$$\Gamma_u = \sigma(W_u[a^{<t-1>}, x^{<t>}] + b_u)$$

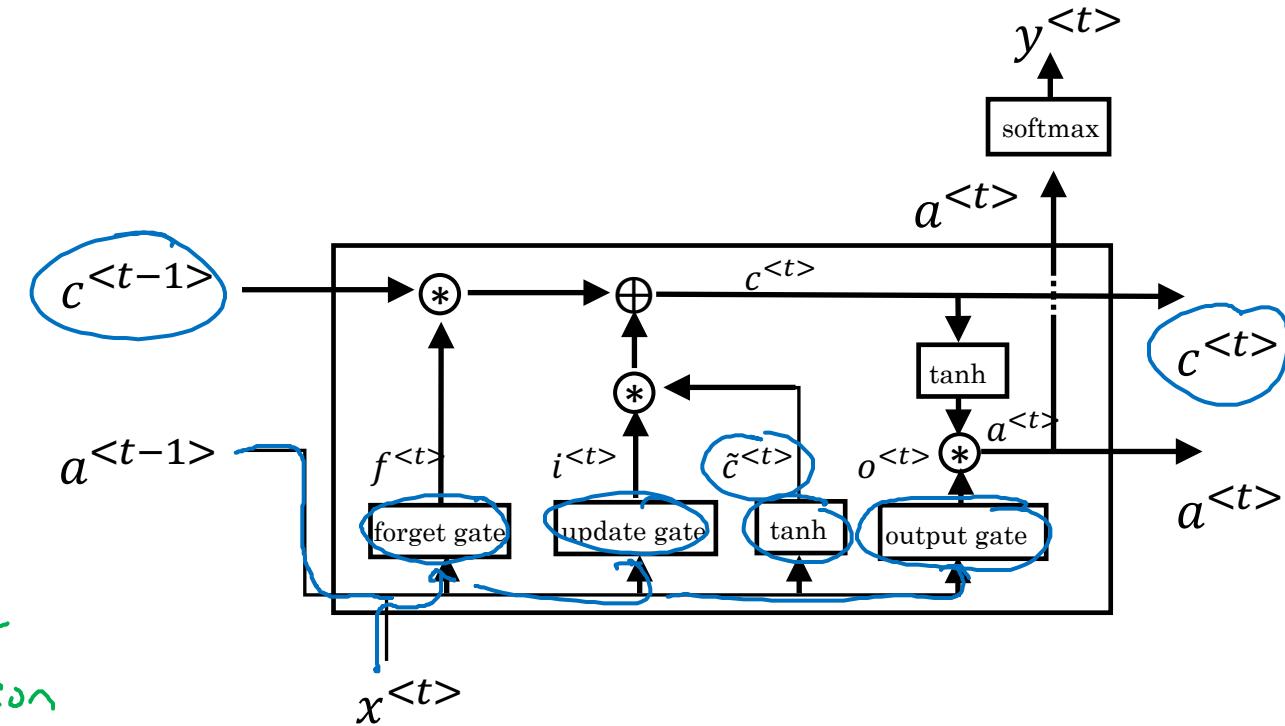
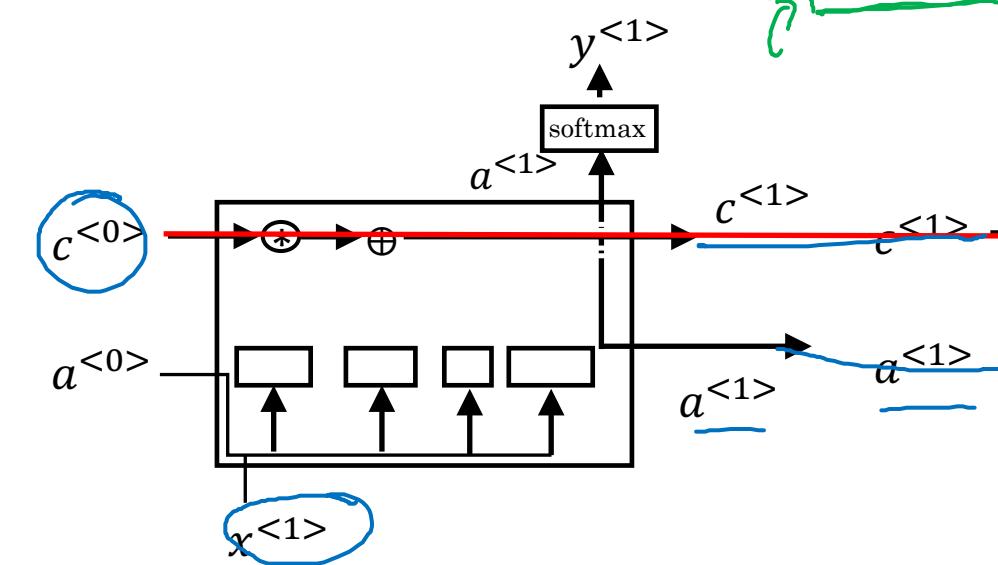
$$\Gamma_f = \sigma(W_f[a^{<t-1>}, x^{<t>}] + b_f)$$

$$\Gamma_o = \sigma(W_o[a^{<t-1>}, x^{<t>}] + b_o)$$

$$c^{<t>} = \Gamma_u * \tilde{c}^{<t>} + \Gamma_f * c^{<t-1>}$$

$$a^{<t>} = \Gamma_o * c^{<t>}$$

peephole connection



Andrew Ng



deeplearning.ai

# Recurrent Neural Networks

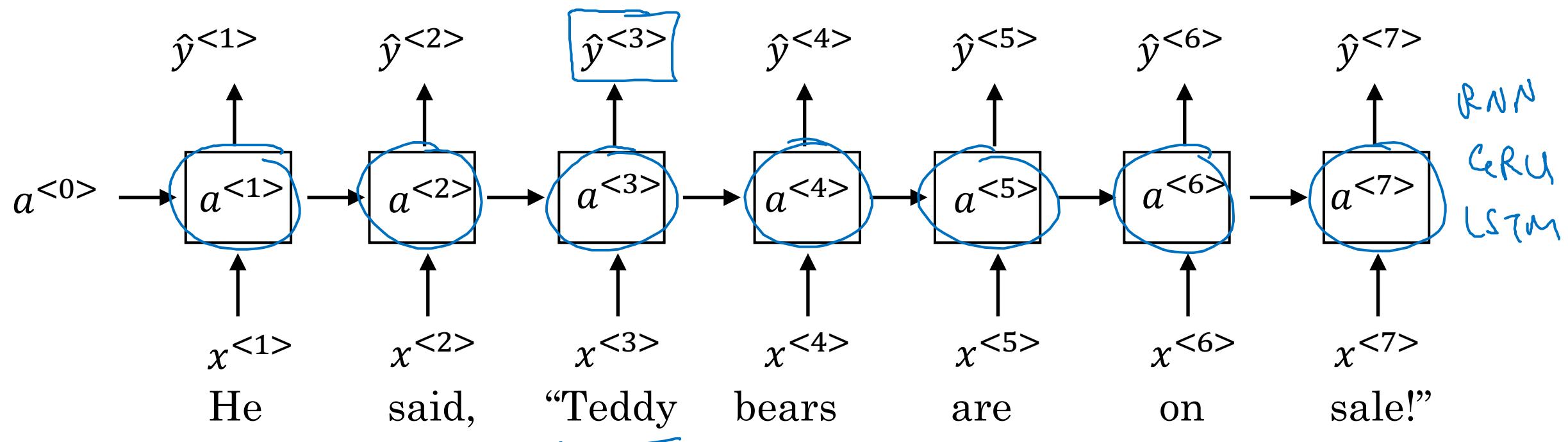
---

## Bidirectional RNN

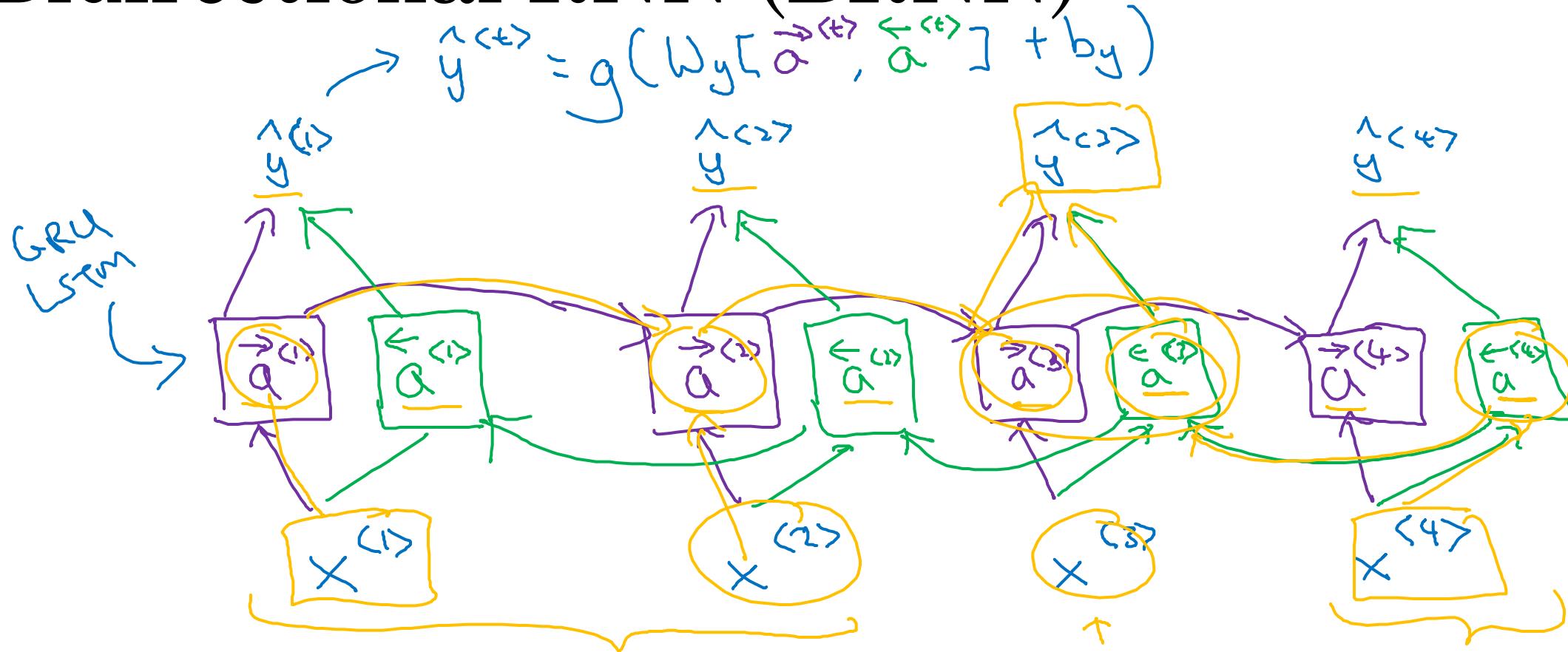
# Getting information from the future

He said, “Teddy bears are on sale!”

He said, “Teddy Roosevelt was a great President!”



# Bidirectional RNN (BRNN)



Acyclic graph

BRNN w/LSTM

He said,

"Teddy Roosevelt ..."



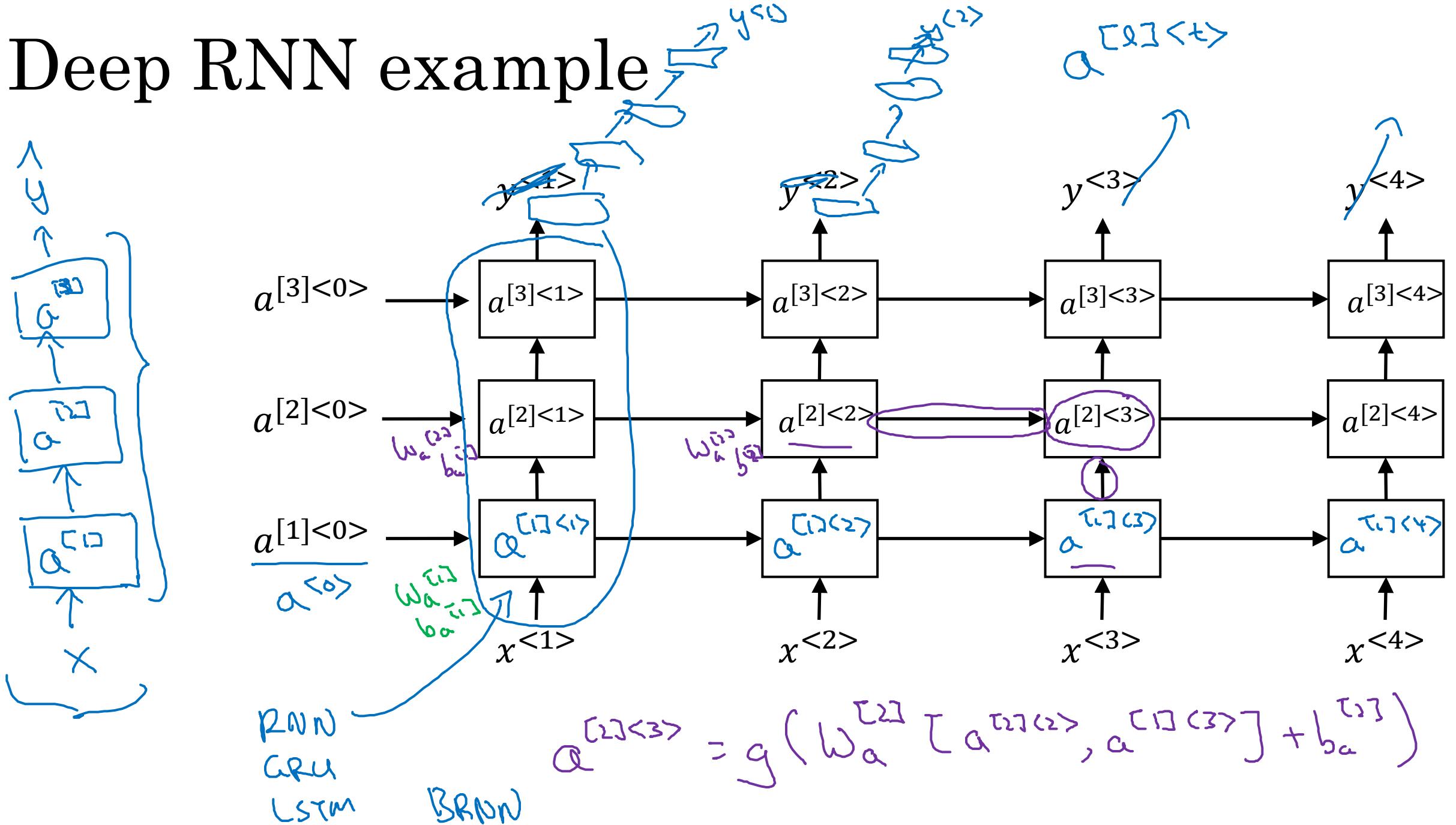
deeplearning.ai

# Recurrent Neural Networks

---

## Deep RNNs

# Deep RNN example



# 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

# NLP and Word Embeddings

---

## Word representation

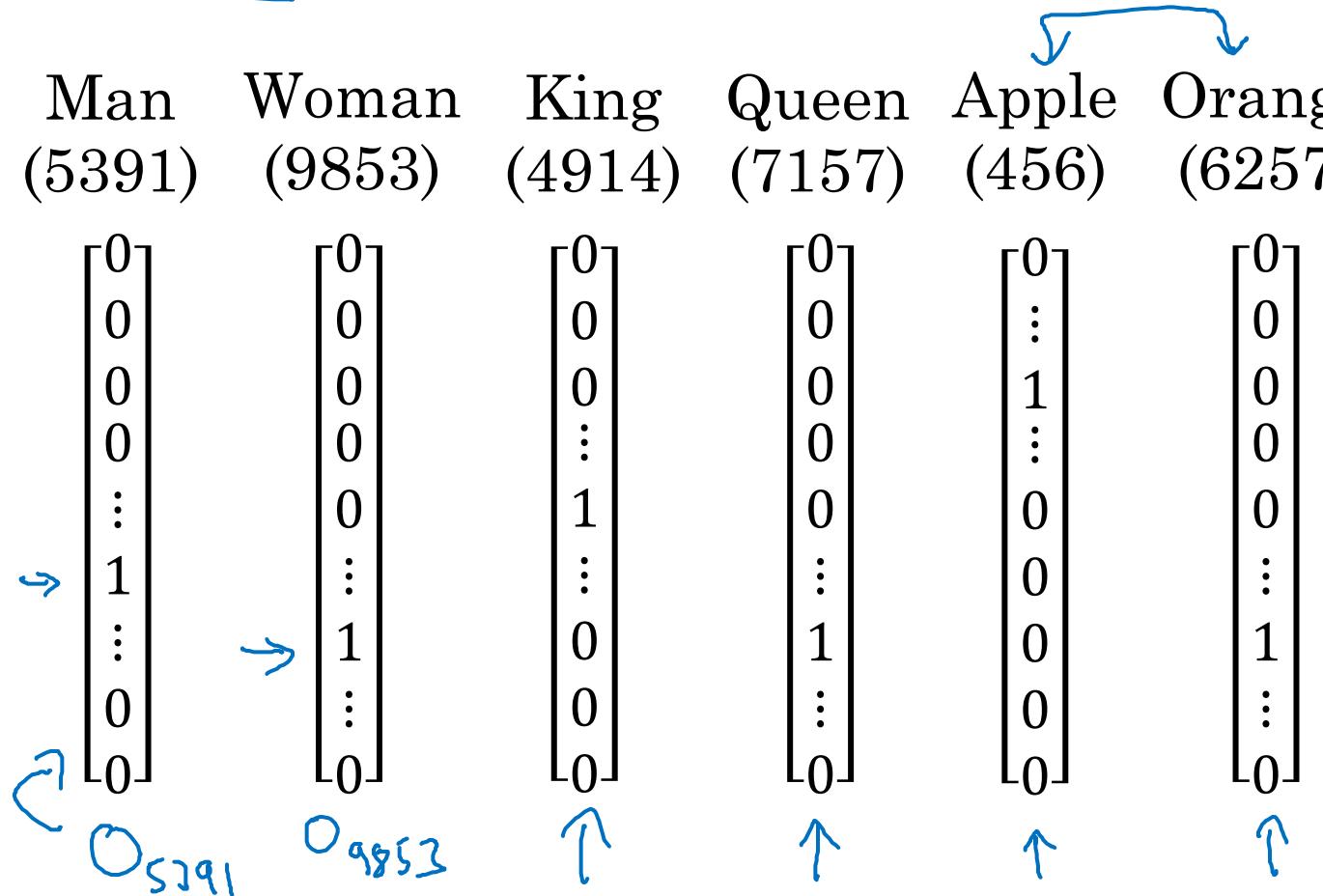
# Word representation

$$V = [a, \text{aaron}, \dots, \text{zulu}, \text{<UNK>}]$$

$$|V| = 10,000$$

1-hot representation

|               |                 |                |                 |                |                  |
|---------------|-----------------|----------------|-----------------|----------------|------------------|
| Man<br>(5391) | Woman<br>(9853) | King<br>(4914) | Queen<br>(7157) | Apple<br>(456) | Orange<br>(6257) |
|---------------|-----------------|----------------|-----------------|----------------|------------------|



I want a glass of orange juice.  
I want a glass of apple ?.

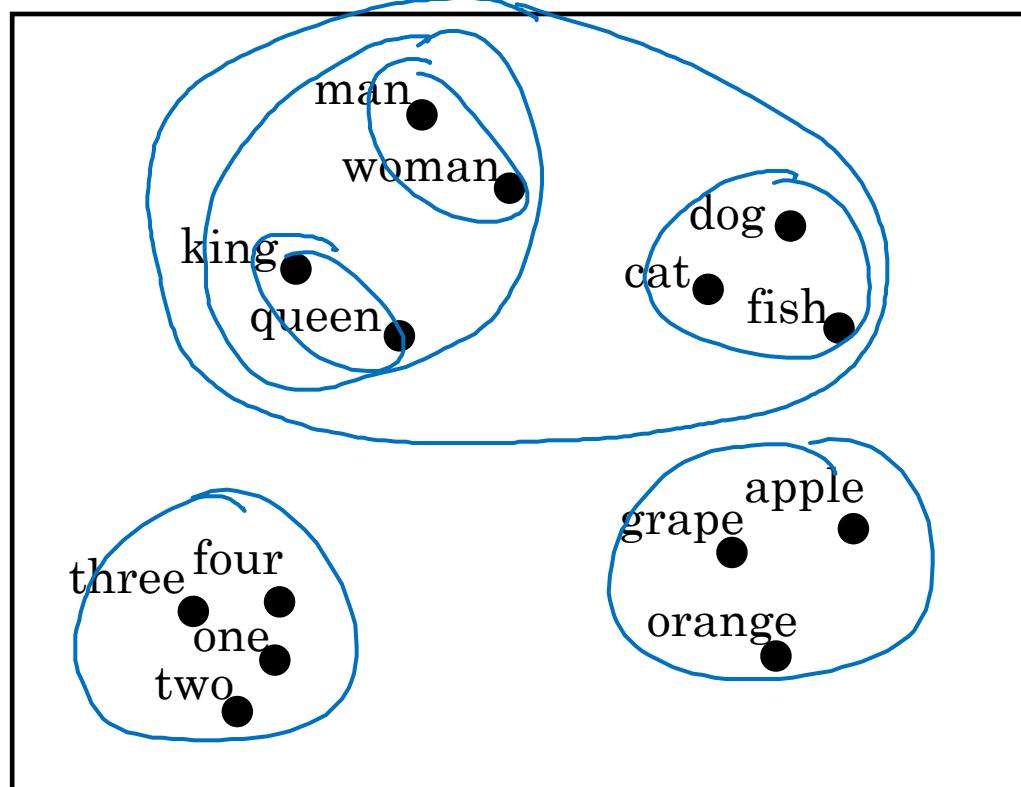
# Featurized representation: word embedding

|        | Man<br>(5391) | Woman<br>(9853) | King<br>(4914) | Queen<br>(7157) | Apple<br>(456) | Orange<br>(6257) |
|--------|---------------|-----------------|----------------|-----------------|----------------|------------------|
| Gender | -1            | 1               | -0.95          | 0.97            | 0.00           | 0.01             |
| Royal  | 0.01          | 0.02            | 0.93           | 0.95            | -0.01          | 0.00             |
| Age    | 0.03          | 0.02            | 0.7            | 0.69            | 0.03           | -0.02            |
| Food   | 0.04          | 0.01            | 0.02           | 0.01            | 0.95           | 0.97             |
| Size   | :             | :               |                |                 |                |                  |
| Cost   |               |                 |                |                 |                |                  |
| Color  |               |                 |                |                 |                |                  |
| Verb   |               |                 |                |                 |                |                  |

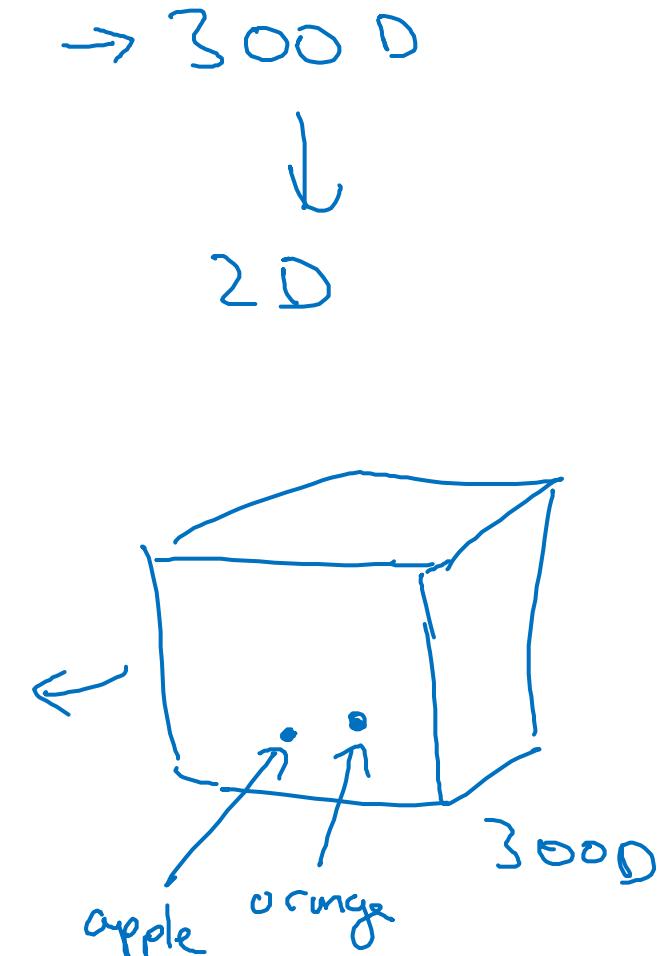
Handwritten annotations:

- Vertical column on the left: Gender, Royal, Age, Food, Size, Cost, Color, Verb.
- Horizontal row labels: Man (5391), Woman (9853), King (4914), Queen (7157), Apple (456), Orange (6257).
- Cells for 'Man' and 'Woman' rows are highlighted with blue boxes.
- Cells for 'Apple' and 'Orange' columns are highlighted with blue boxes.
- Cells for 'King' and 'Queen' rows are underlined.
- Cells for 'Apple' and 'Orange' rows are underlined.
- Cells for 'Apple' and 'Orange' rows are grouped by a blue brace on the right.
- Cells for 'Apple' and 'Orange' rows are grouped by a blue brace at the bottom.
- Text at the bottom right: "I want a glass of orange juice." and "I want a glass of apple juice." followed by "Andrew Ng".

# Visualizing word embeddings



t-SNE





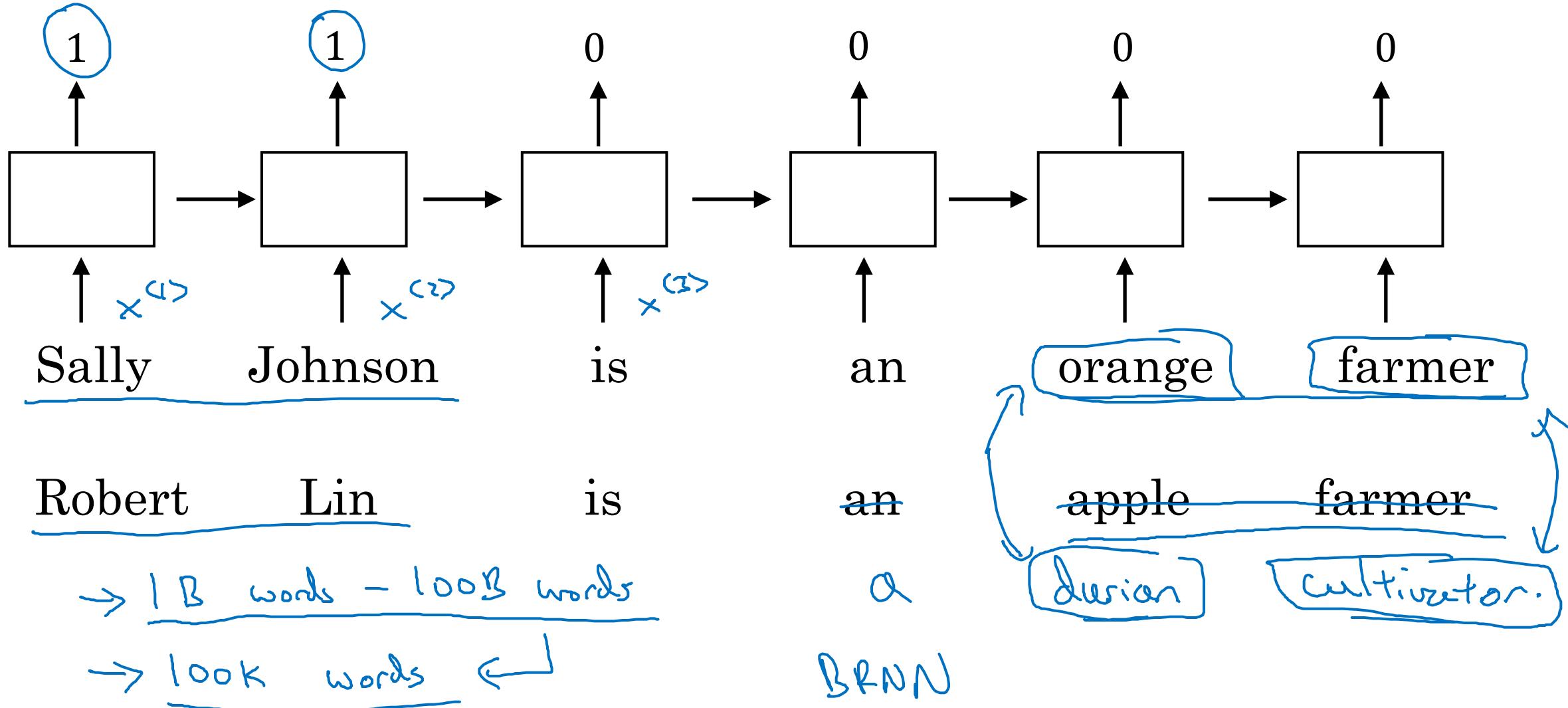
deeplearning.ai

# NLP and Word Embeddings

---

## Using word embeddings

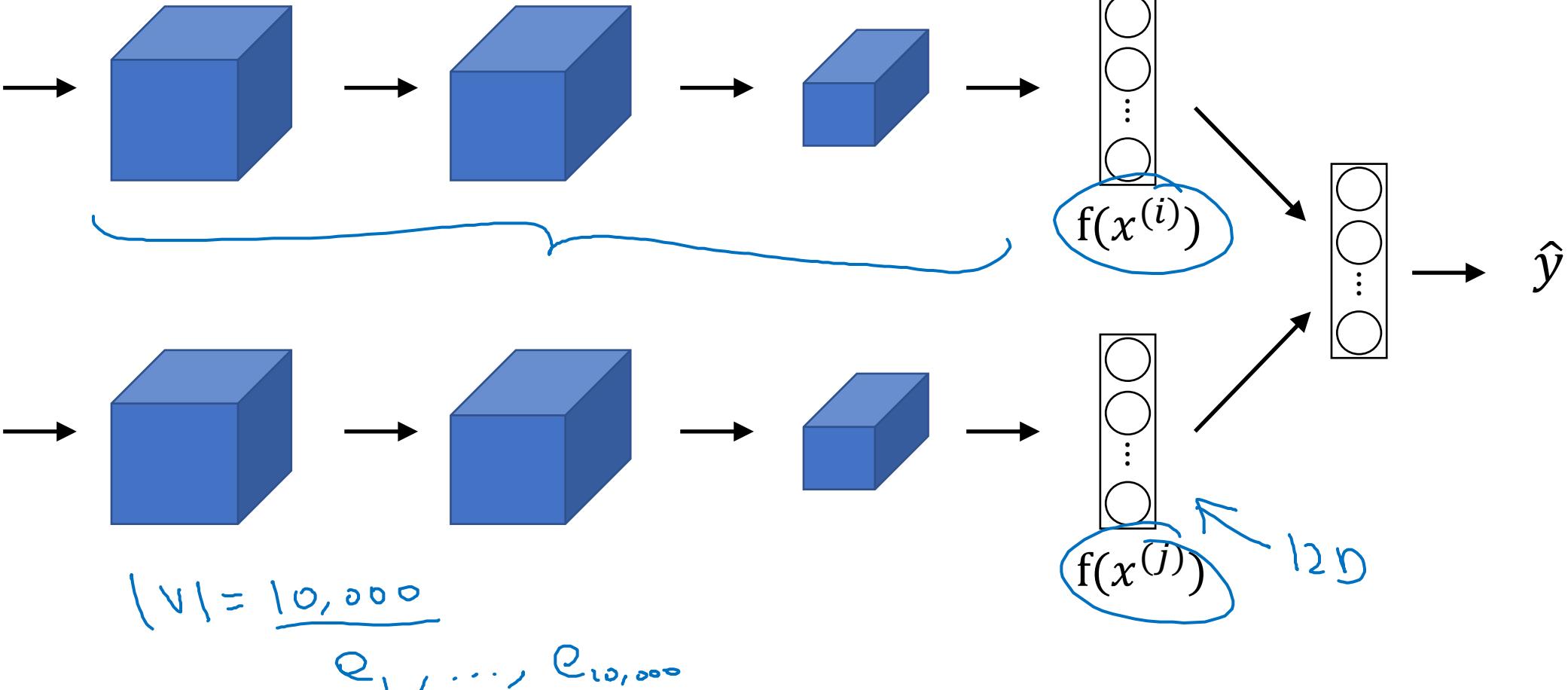
# Named entity recognition example



# Transfer learning and word embeddings

- 
- The diagram consists of two blue-outlined brackets. The top bracket, labeled 'A' in a circle, encloses the first two steps. The bottom bracket, labeled 'B' in a circle, encloses the third step. A vertical blue arrow points downwards from the end of the 'A' bracket to the start of the 'B' bracket.
1. Learn word embeddings from large text corpus. (1-100B words)  
(Or download pre-trained embedding online.)
  2. Transfer embedding to new task with smaller training set.  
(say, 100k words)  $\rightarrow 10,000 \quad \rightarrow 300$
  3. Optional: Continue to finetune the word embeddings with new data.

# Relation to face encoding (embedding) 128D





deeplearning.ai

# NLP and Word Embeddings

---

## Properties of word embeddings

# Analogy

|        | Man<br>(5391) | Woman<br>(9853) | King<br>(4914) | Queen<br>(7157) | Apple<br>(456) | Orange<br>(6257) |
|--------|---------------|-----------------|----------------|-----------------|----------------|------------------|
| Gender | -1            | 1               | -0.95          | 0.97            | 0.00           | 0.01             |
| Royal  | 0.01          | 0.02            | 0.93           | 0.95            | -0.01          | 0.00             |
| Age    | 0.03          | 0.02            | 0.70           | 0.69            | 0.03           | -0.02            |
| Food   | 0.09          | 0.01            | 0.02           | 0.01            | 0.95           | 0.97             |

$$\begin{matrix} e_{5391} \\ e_{\text{man}} \end{matrix}$$

$$\underline{\text{Man} \rightarrow \text{Woman}}$$

$$e_{\text{man}} - e_{\text{woman}}$$

$$e_{\text{woman}}$$

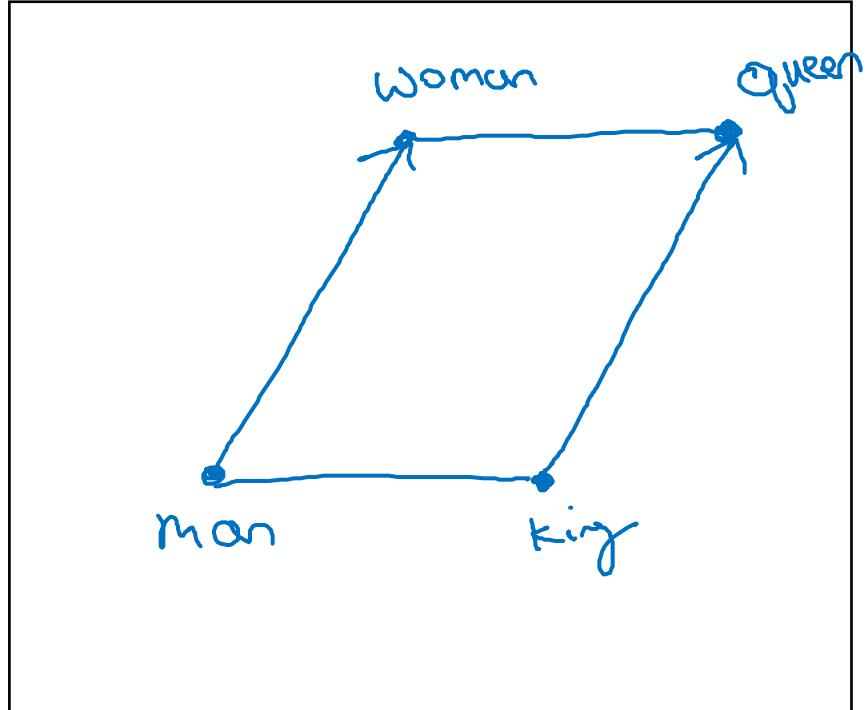
$$\underline{\text{King} \rightarrow ? \text{ Queen}}$$

$$e_{\text{king}} - e_{? \text{ Queen}}$$

$$\underline{e_{\text{man}} - e_{\text{woman}}} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

$$\underline{e_{\text{king}} - e_{\text{queen}}} \approx \begin{bmatrix} -2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

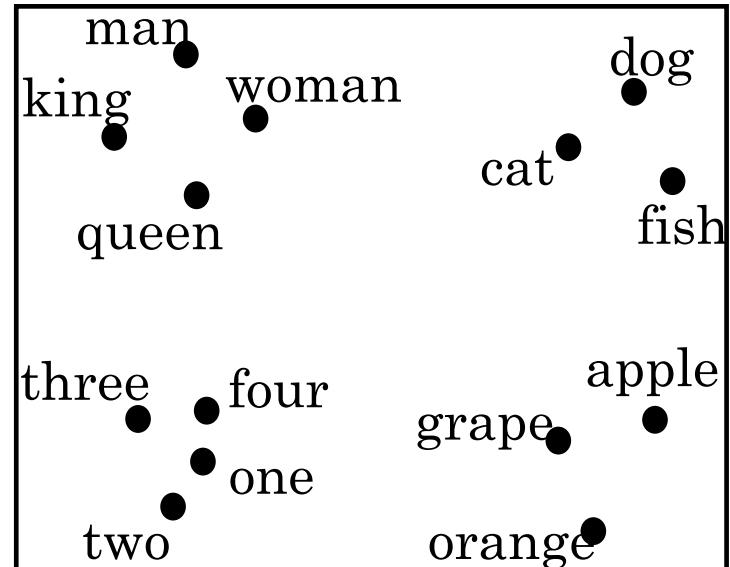
# Analogies using word vectors



300 D

Find word  $w_i: \arg \max_w$

$300D \rightarrow 2D$



t-SNE

$$e_{\text{man}} - e_{\text{woman}} \approx e_{\text{king}} - e_{\underline{\text{?}}} e_w$$

$\underbrace{\hspace{10cm}}$

$\downarrow$

$\text{Sim}(e_w, e_{\text{king}} - e_{\text{man}} + e_{\text{woman}})$

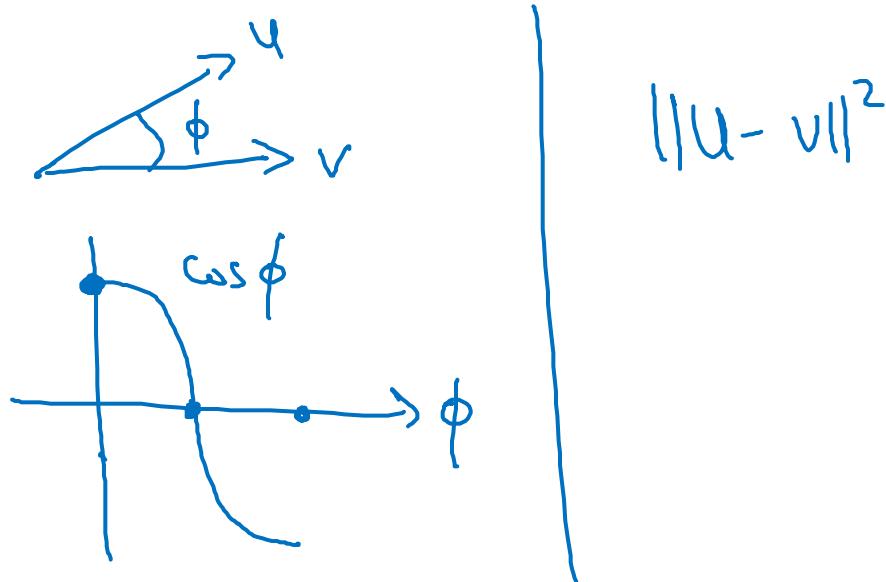
$\uparrow$

30 - 75%

# Cosine similarity

$$\rightarrow \boxed{\text{sim}(e_w, e_{king} - e_{man} + e_{woman})}$$

$$\text{sim}(u, v) = \frac{u^T v}{\|u\|_2 \|v\|_2}$$



Man:Woman as Boy:Girl  
Ottawa:Canada as Nairobi:Kenya  
Big:Bigger as Tall:Taller  
Yen:Japan as Ruble:Russia



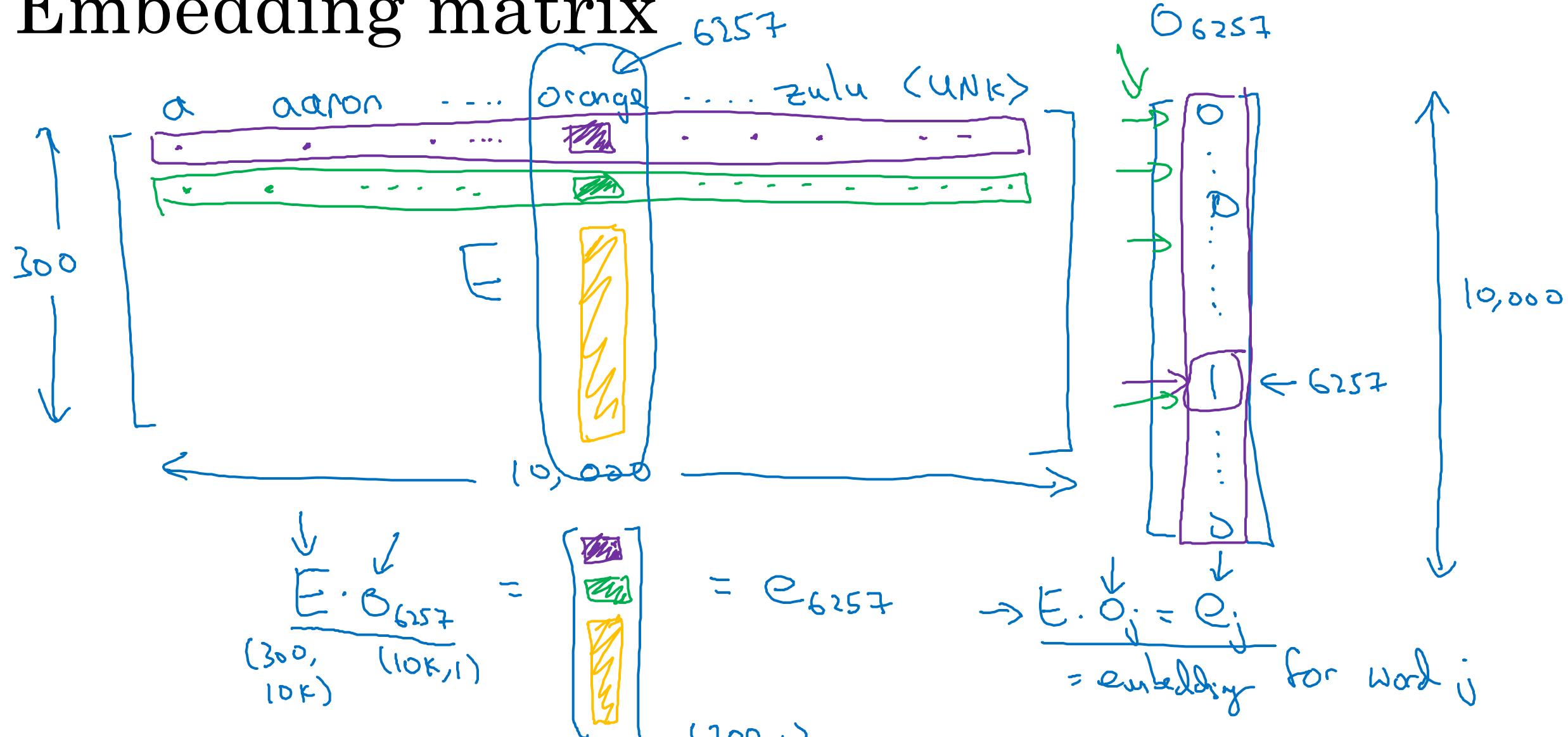
deeplearning.ai

# NLP and Word Embeddings

---

## Embedding matrix

# Embedding matrix



In practice, use specialized function to look up an embedding.  
→ Embedding



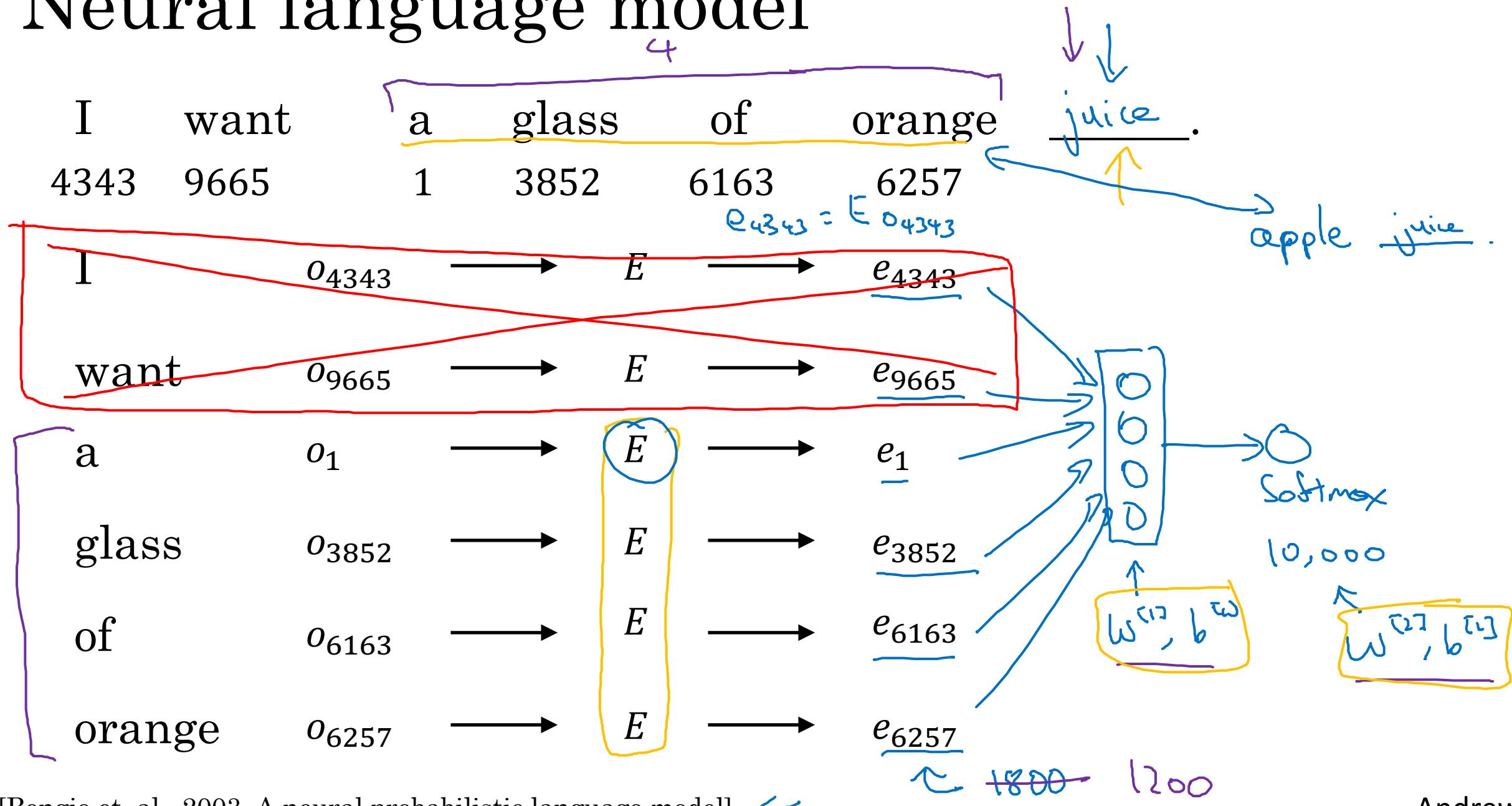
deeplearning.ai

# NLP and Word Embeddings

---

## Learning word embeddings

# Neural language model



# Other context/target pairs

I want a **glass** of **orange** juice to go along with my cereal.

Context: Last 4 words.

4 words on left & right

Last 1 word

Nearby 1 word

skip gram

a glass of orange ? to go along with

orange ?

glass . ?



deeplearning.ai

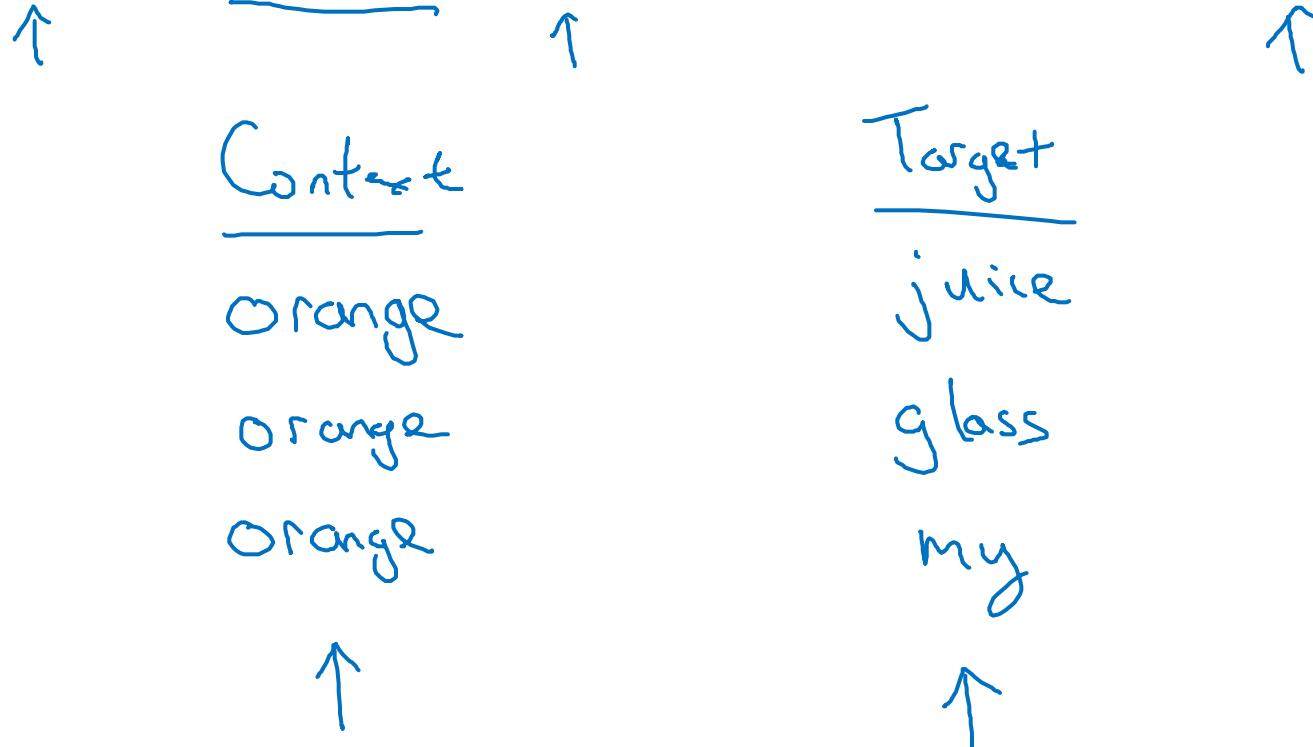
# NLP and Word Embeddings

---

## Word2Vec

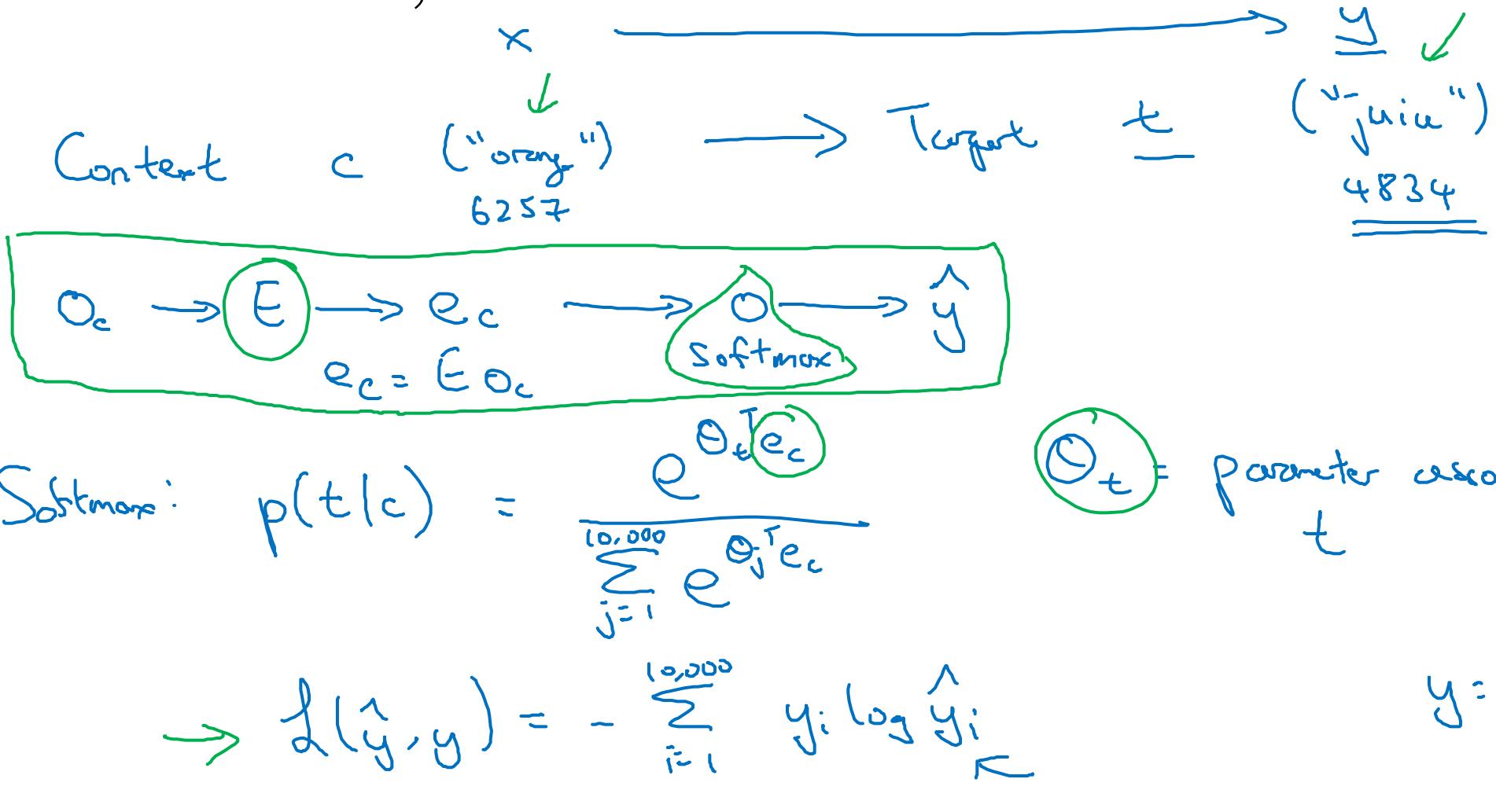
# Skip-grams

I want a glass of orange juice to go along with my cereal.



# Model

Vocab size = 10,000k

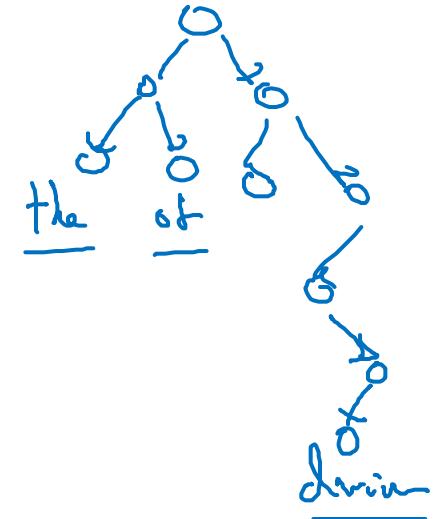
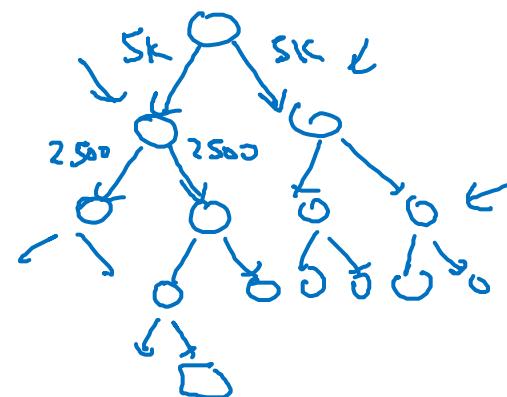


# Problems with softmax classification

$$\underline{p(t|c)} = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

log lvl

Hierarchical software.



# How to sample the context $c$ ?

→ the, of, a, and, to, ...

→ Orange, apple, durian

Qdurian

七

$c \rightarrow t$

P(c)



deeplearning.ai

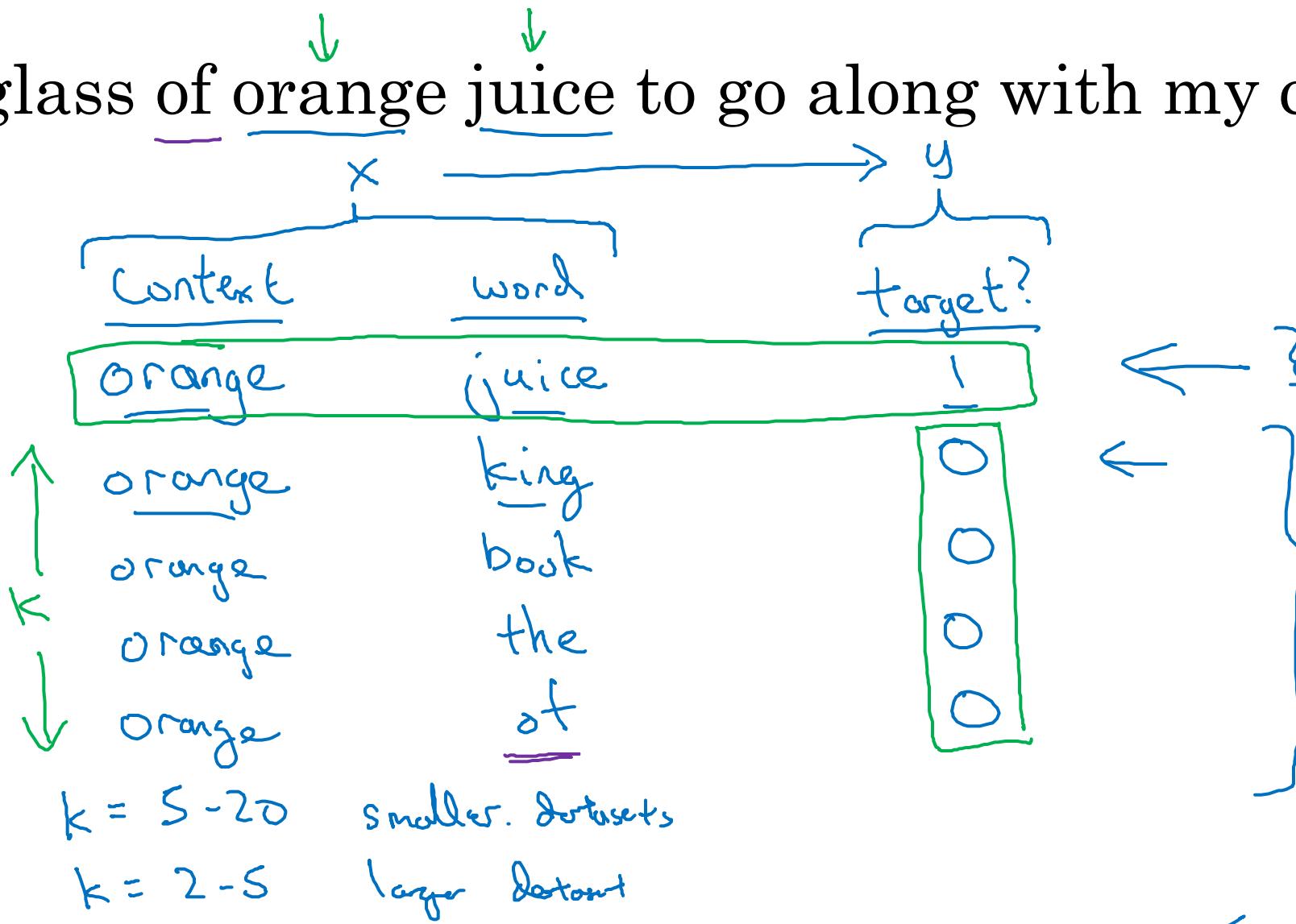
# NLP and Word Embeddings

---

## Negative sampling

# Defining a new learning problem

I want a glass of orange juice to go along with my cereal.



# Model

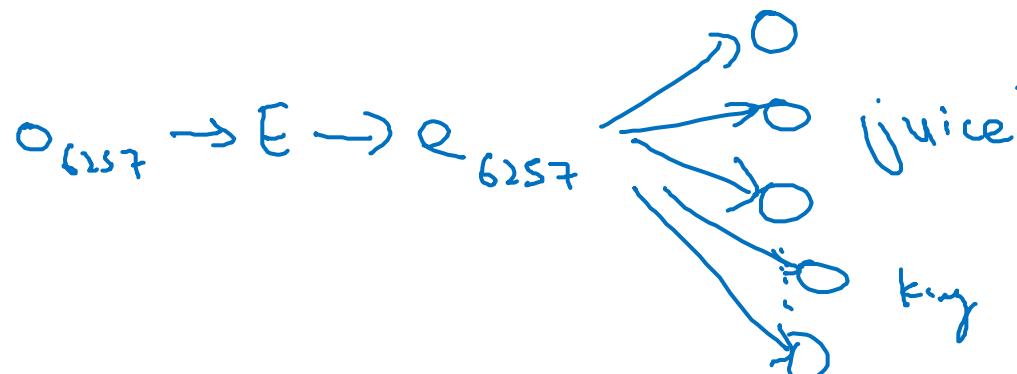
Softmax:

$$p(t|c) = \frac{e^{\theta_t^T e_c}}{\sum_{j=1}^{10,000} e^{\theta_j^T e_c}}$$

10,000-way softmax

$$P(y=1 | c, t) = \sigma(\theta_t^T e_c)$$

Orange  
6257



|         | x     | y |  |
|---------|-------|---|--|
| context |       |   |  |
| word    |       |   |  |
| orange  | juice | 1 |  |
| orange  | king  | 0 |  |
| orange  | book  | 0 |  |
| orange  | the   | 0 |  |
| orange  | of    | 0 |  |

↑  
c  
↑  
t  
↑  
y

10,000 binary classification problem  
k+1

# Selecting negative examples

| <u>context</u> | <u>word</u> | <u>target?</u> |
|----------------|-------------|----------------|
| orange         | juice       | 1              |
| orange         | king        | 0              |
| orange         | book        | 0              |
| orange         | the         | 0              |
| orange         | of          | 0              |

the , of, and, ...

$$P(w_i) = \frac{f(w_i)^{3/4}}{\sum_{j=1}^{10,000} f(w_j)^{3/4}}$$

$$\frac{1}{|V|}$$



deeplearning.ai

# NLP and Word Embeddings

---

## GloVe word vectors

# GloVe (global vectors for word representation)

I want a glass of orange juice to go along with my cereal.

c, t

$x_{i,j} = \# \text{ times } i \text{ appears in context of } j.$

$x_{i,j}$        $i$        $j$   
↑      ↑      ↑  
c      t      c

$$x_{ij} = x_{ji} \leftarrow$$



## Model

Minimize

$$\sum_{i=1}^{10,000} \sum_{j=1}^{100,000} f(x_{ij}) (\theta_i^T e_j + b_i + b_j' - \log \frac{x_{ij}}{t_c})$$

"  $\theta_i^T e_c$ "

*weight<sub>ij</sub>*  
term

$f(x_{ij}) = 0$  or  $x_{ij} = 0$ .      "0 log 0" = 0

this, is, of, a, ...

derian

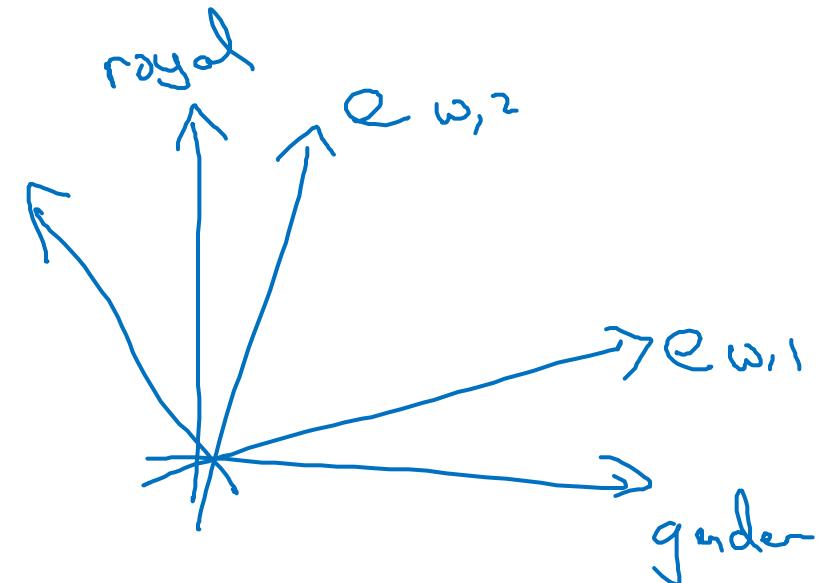
$\theta_i, e_j$  are symmetric

$\theta_w^{(\text{final})} = \frac{\theta_w + \theta_w}{2}$

# Andrew Ng

# A note on the featurization view of word embeddings

|        | Man<br>(5391) | Woman<br>(9853) | King<br>(4914) | Queen<br>(7157) |
|--------|---------------|-----------------|----------------|-----------------|
| Gender | -1            | 1               | -0.95          | 0.97            |
| Royal  | 0.01          | 0.02            | 0.93           | 0.95            |
| Age    | 0.03          | 0.02            | 0.70           | 0.69            |
| Food   | 0.09          | 0.01            | 0.02           | 0.01            |



$$\text{minimize } \sum_{i=1}^{10,000} \sum_{j=1}^{10,000} f(X_{ij}) (\underbrace{\theta_i^T e_j + b_i - b'_j - \log X_{ij}}_{} )^2$$

$$\langle A\theta_i \rangle^T (A^T e_j) = \cancel{\theta_i^T A^T A} \cancel{e_j} = \theta_i^T e_j$$



deeplearning.ai

# NLP and Word Embeddings

---

## Sentiment classification

# Sentiment classification problem



The dessert is excellent.



Service was quite slow.



Good for a quick meal, but nothing special.



Completely lacking in good taste, good service, and good ambience.



10,000  $\rightarrow$  100,000 words

# Simple sentiment classification model

The dessert is excellent  
8928 2468 4694 3180



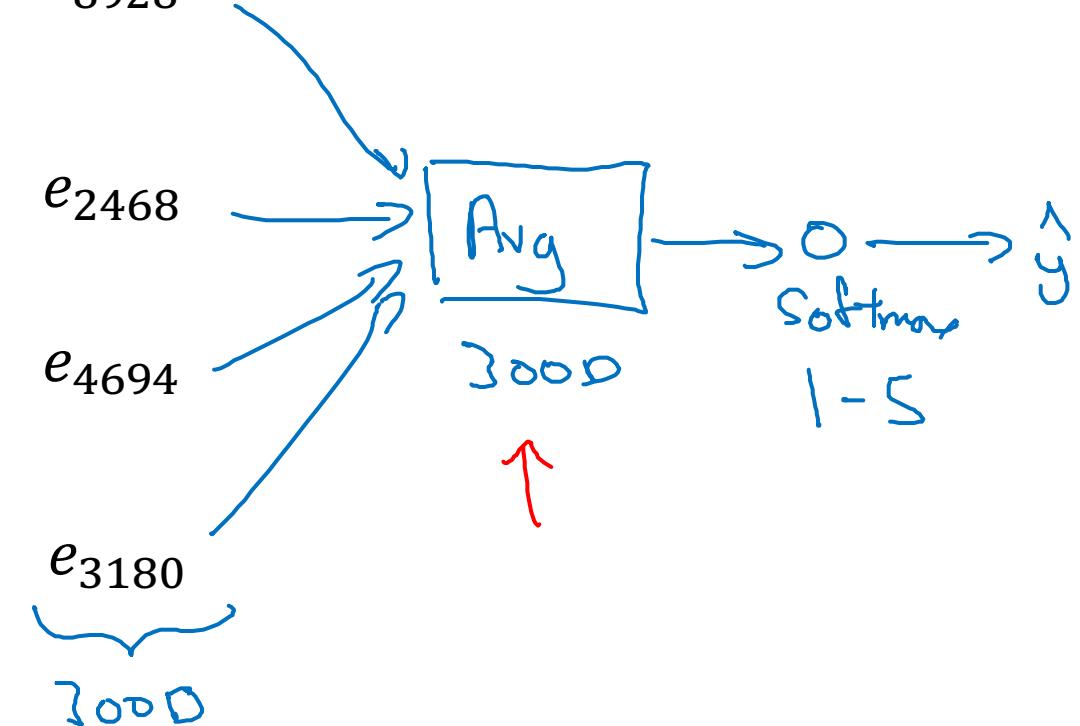
The  $o_{8928}$   $\rightarrow E \rightarrow e_{8928}$

desert  $o_{2468} \rightarrow E \rightarrow e_{2468}$

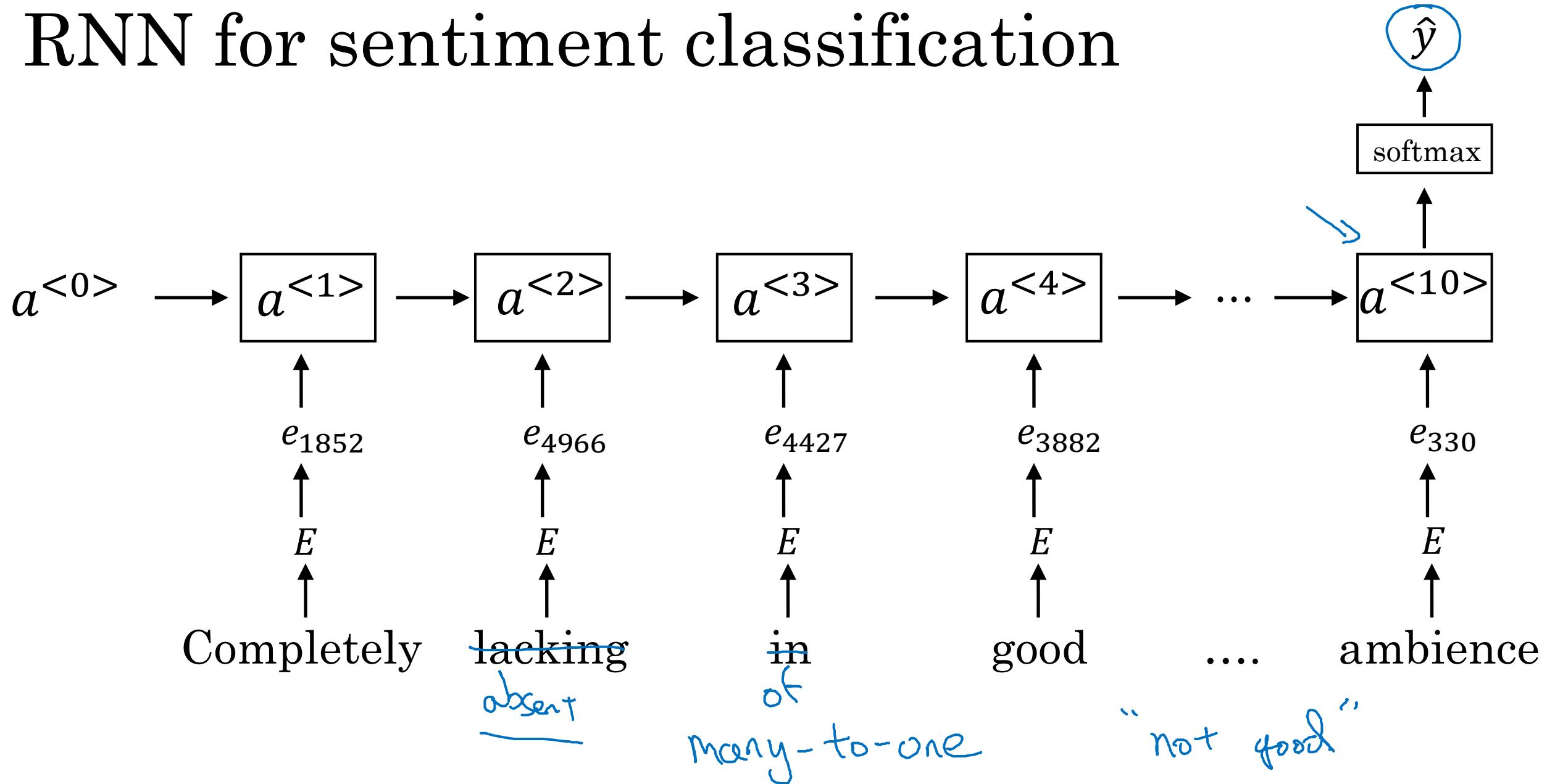
is  $o_{4694} \rightarrow E \rightarrow e_{4694}$

excellent  $o_{3180} \rightarrow E \rightarrow e_{3180}$

“Completely lacking in good taste, good service, and good ambience.”  
↑  
100 B words



# RNN for sentiment classification





deeplearning.ai

# NLP and Word Embeddings

---

## Debiasing word embeddings

# The problem of bias in word embeddings

Man:Woman as King:Queen

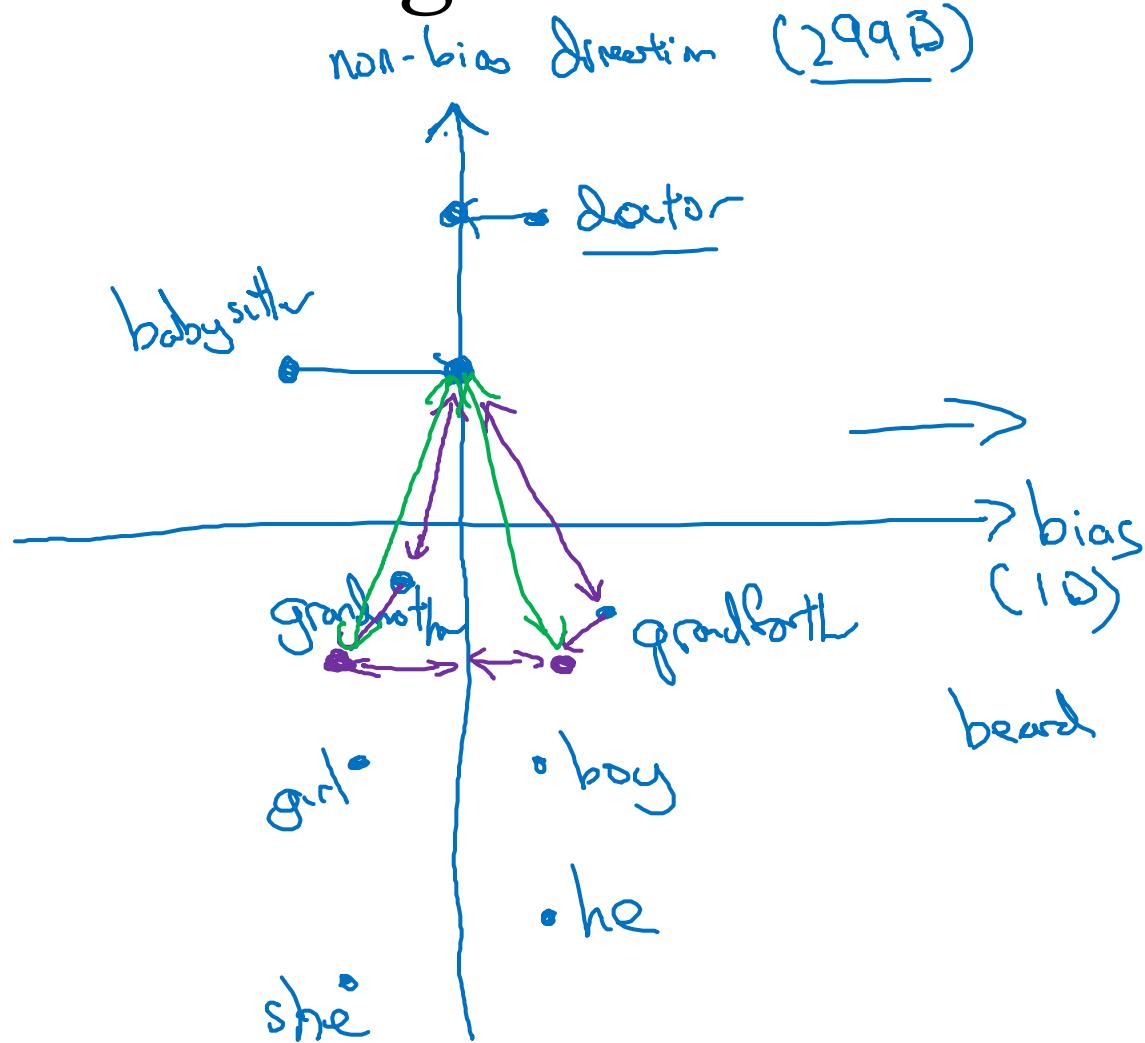
Man:Computer\_Programmer as Woman:Homemaker 

Father:Doctor as Mother:Nurse 

Word embeddings can reflect gender, ethnicity, age, sexual orientation, and other biases of the text used to train the model.



# Addressing bias in word embeddings



1. Identify bias direction.

$$\left\{ \begin{array}{l} e_{\text{he}} - e_{\text{she}} \\ e_{\text{male}} - e_{\text{female}} \\ \vdots \\ \text{average} \end{array} \right.$$

2. Neutralize: For every word that is not definitional, project to get rid of bias.

3. Equalize pairs.

$$\rightarrow \text{grandmother} - \text{grandfather} = \text{boy} - \text{girl}$$

# 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

# Sequence to sequence models

---

## Basic models

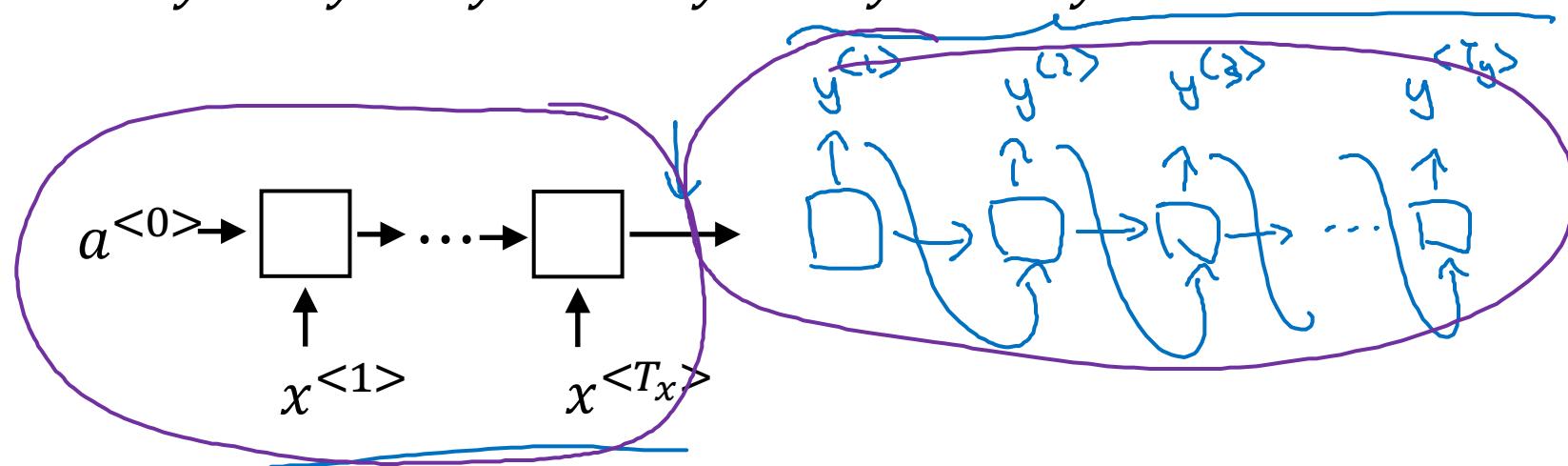
# Sequence to sequence model

$$x^{<1>} \quad x^{<2>} \quad x^{<3>} \quad x^{<4>} \quad x^{<5>}$$

Jane visite l'Afrique en septembre

→ Jane is visiting Africa in September.

$$y^{<1>} \quad y^{<2>} \quad y^{<3>} \quad y^{<4>} \quad y^{<5>} \quad y^{<6>}$$

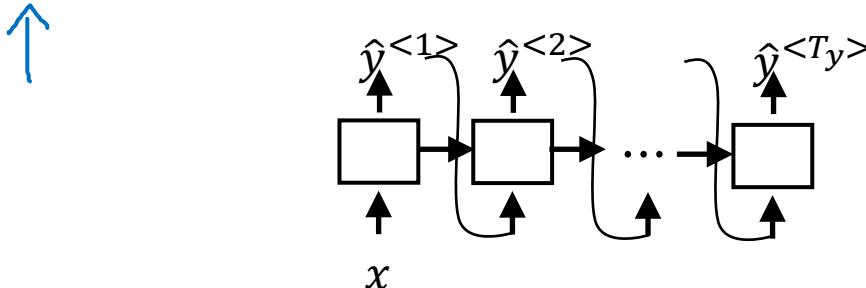
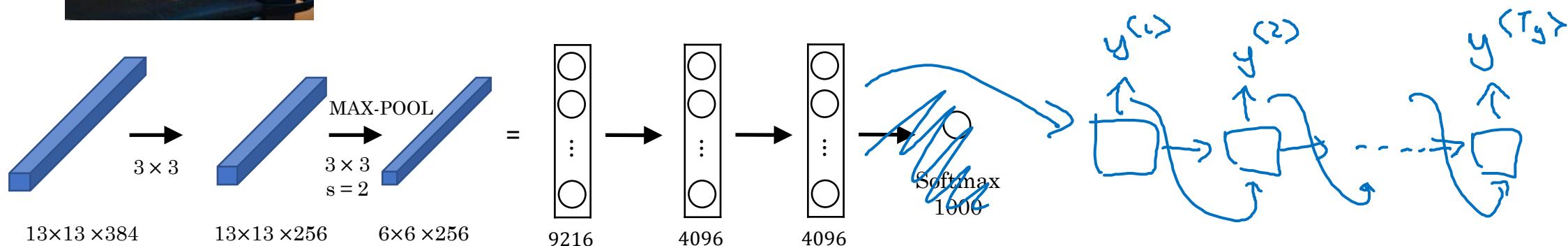
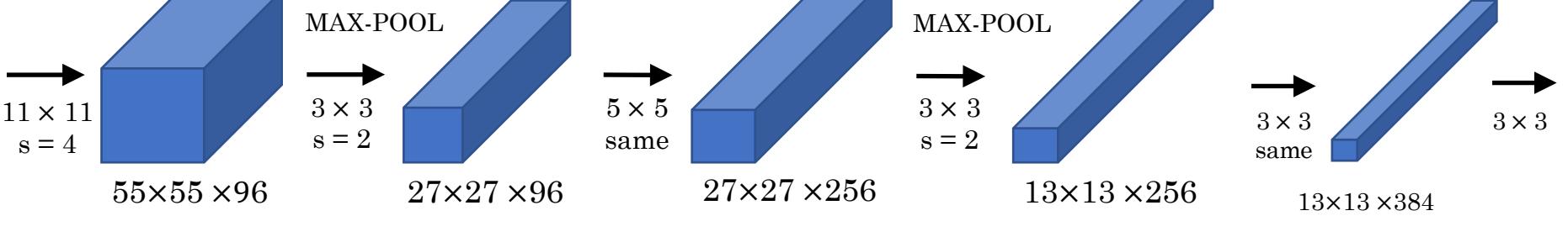
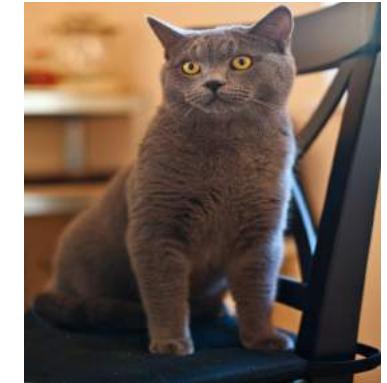


[Sutskever et al., 2014. Sequence to sequence learning with neural networks] ↩

[Cho et al., 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation] ↩

Andrew Ng

# Image captioning



[Mao et. al., 2014. Deep captioning with multimodal recurrent neural networks]

[Vinyals et. al., 2014. Show and tell: Neural image caption generator]

[Karpathy and Li, 2015. Deep visual-semantic alignments for generating image descriptions]

Andrew Ng



deeplearning.ai

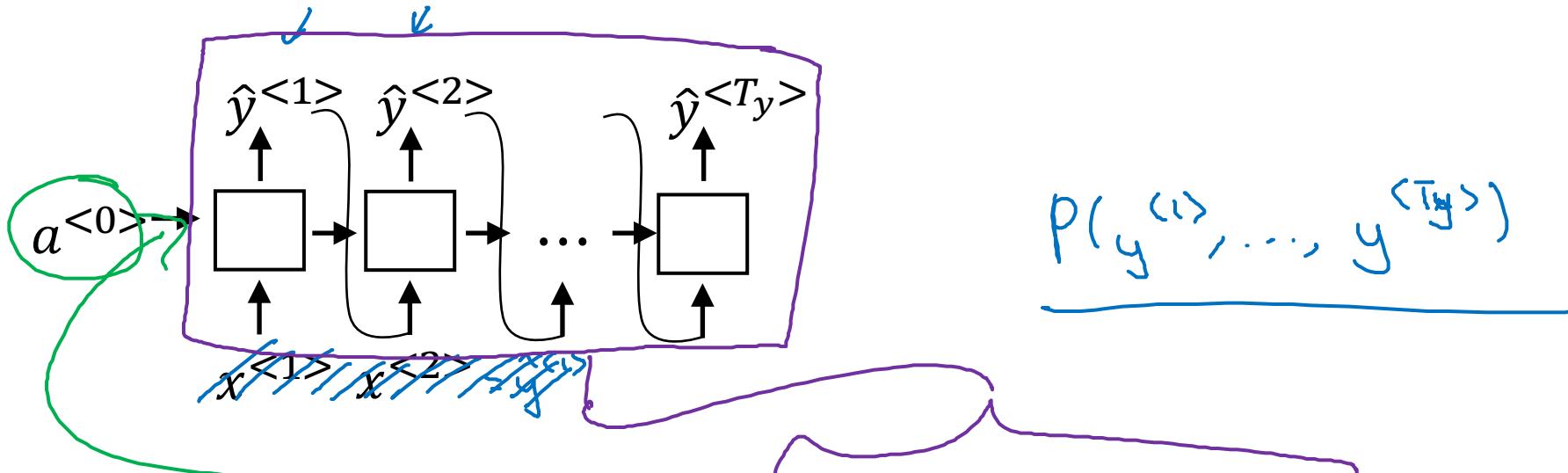
# Sequence to sequence models

---

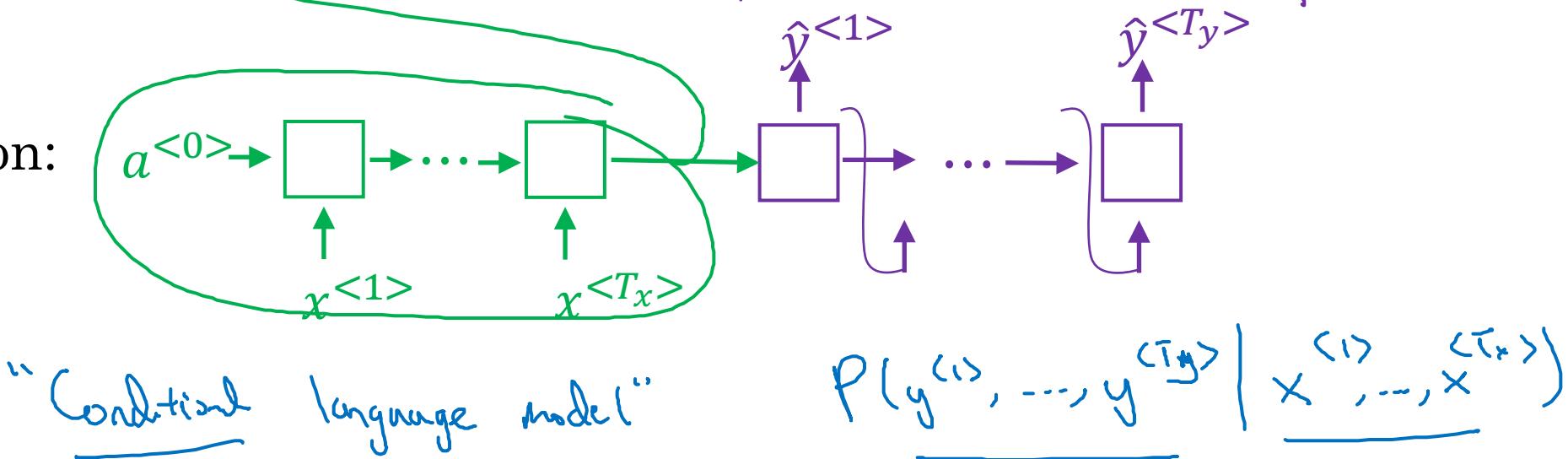
Picking the most  
likely sentence

# Machine translation as building a conditional language model

Language model:



Machine translation:



Andrew Ng

# Finding the most likely translation

Jane visite l'Afrique en septembre.

$$P(y^{<1>} , \dots , y^{<T_y>} | x)$$

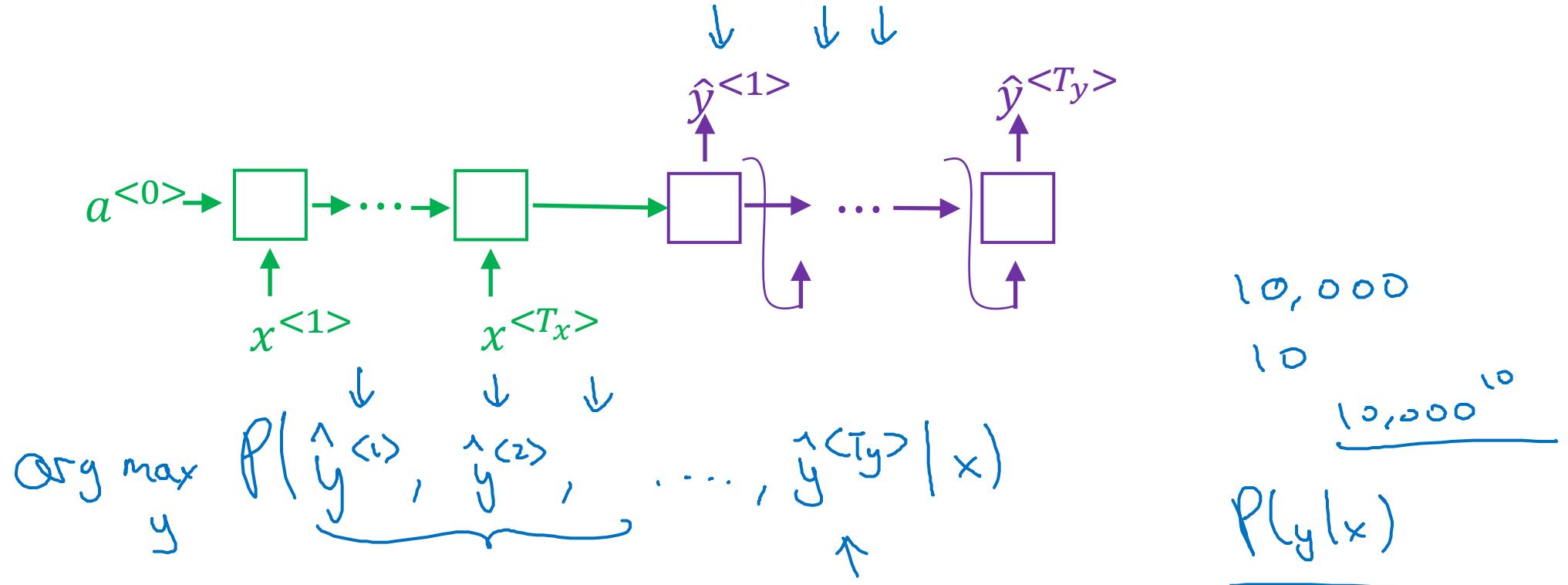
French  
↓  
English

- Jane is visiting Africa in September.
- Jane is going to be visiting Africa in September.
- In September, Jane will visit Africa.
- Her African friend welcomed Jane in September.

$$\arg \max_{y^{<1>} , \dots , y^{<T_y>}} P(y^{<1>} , \dots , y^{<T_y>} | x)$$

# Why not a greedy search?

$$P(\hat{y}^{(1)} | x)$$



$$\frac{10,000}{10} = \underline{\underline{10,000^{10}}} \\ P(y|x)$$

→ Jane is visiting Africa in September.

→ Jane is going to be visiting Africa in September.

$$P(\text{Jane is going } | x) > P(\text{Jane is visiting } | x)$$



deeplearning.ai

# Sequence to sequence models

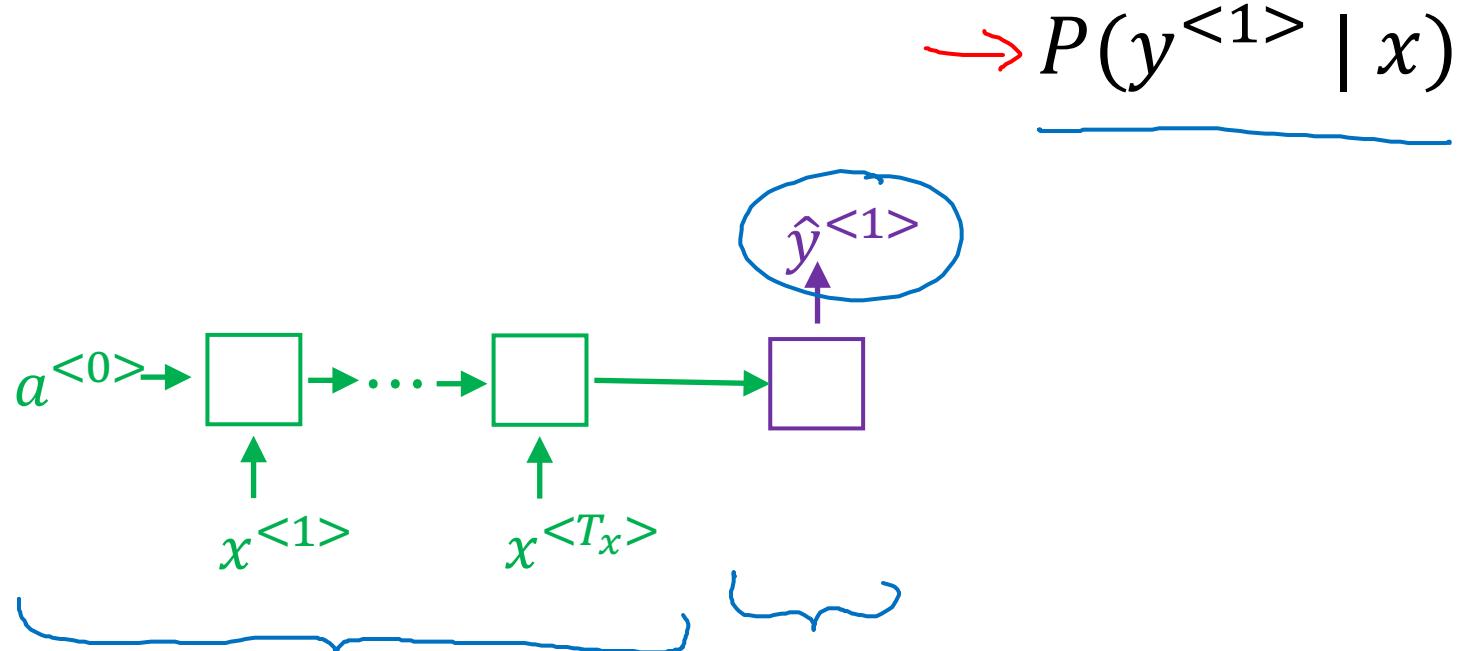
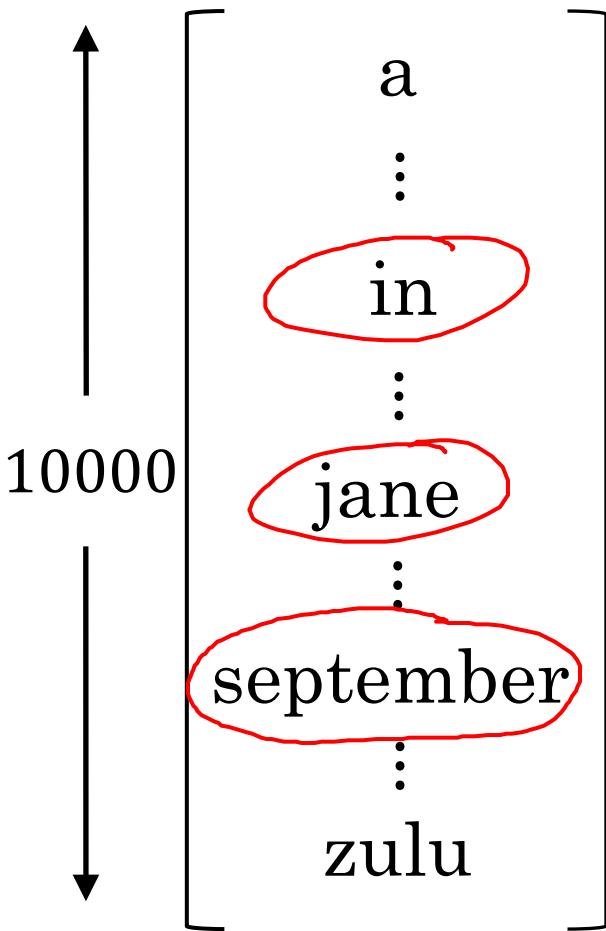
---

## Beam search

# Beam search algorithm

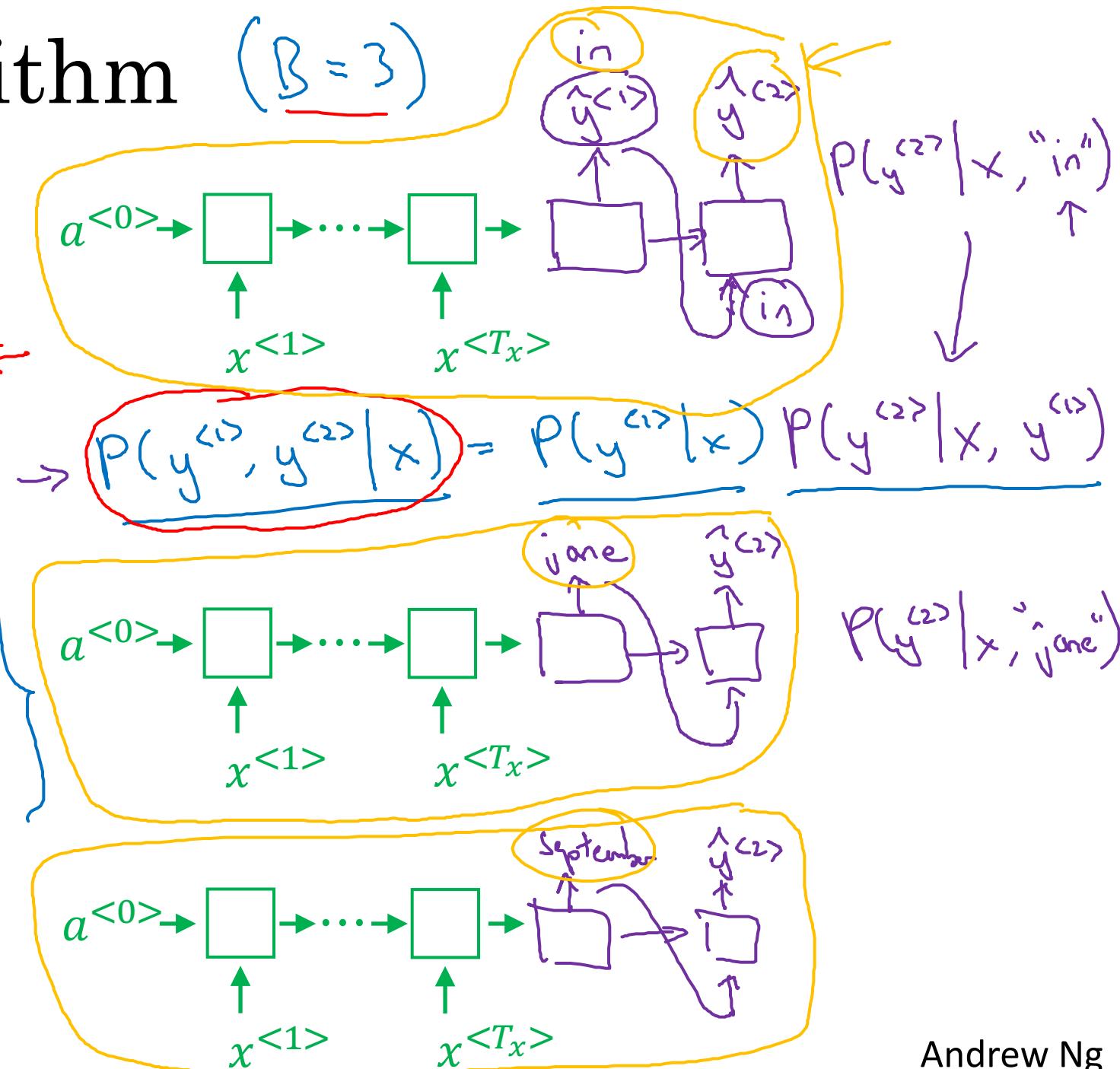
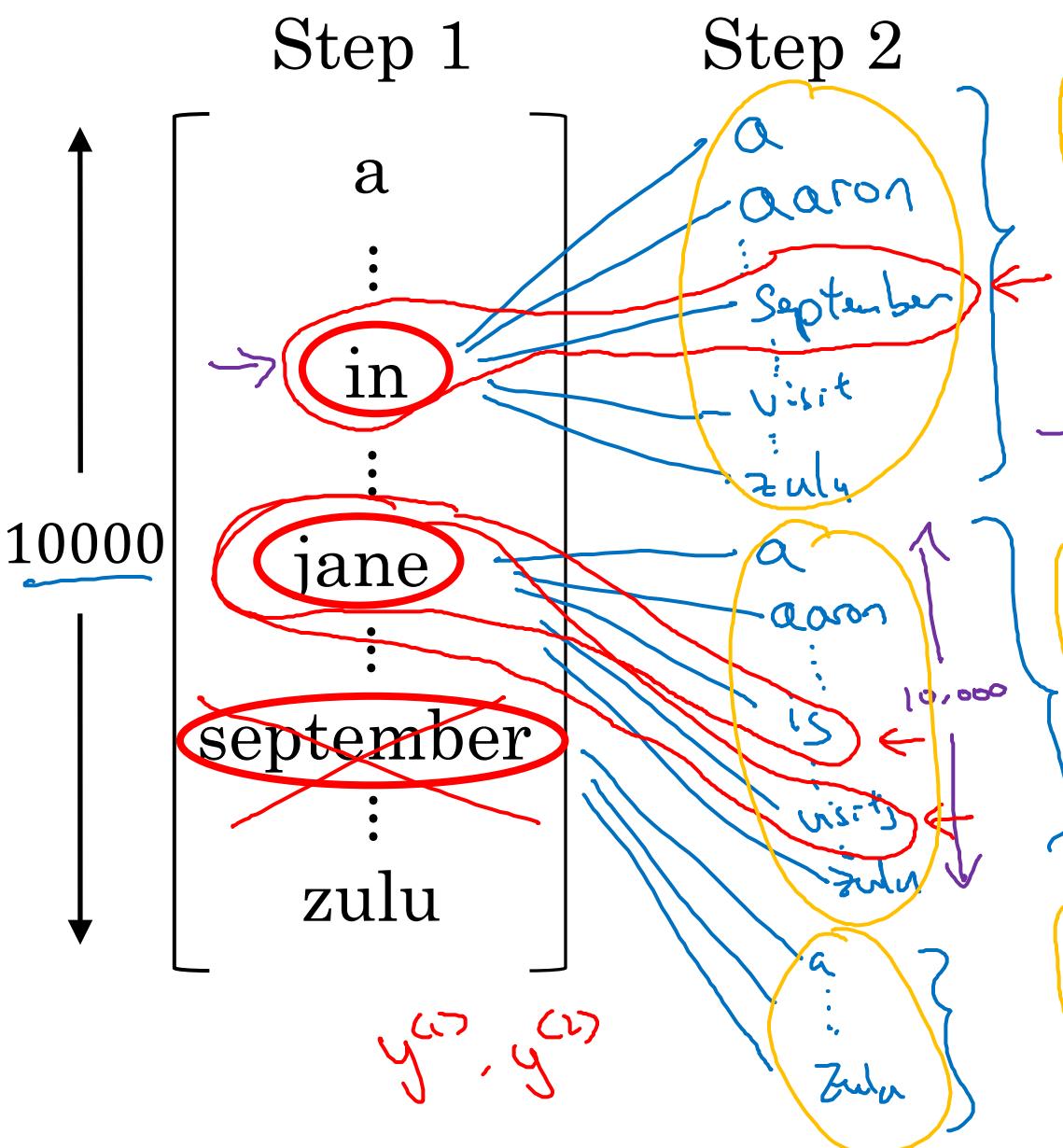
B = 3 (beam width)

Step 1



# Beam search algorithm

$$(B = 3)$$



# Beam search ( $B = 3$ )

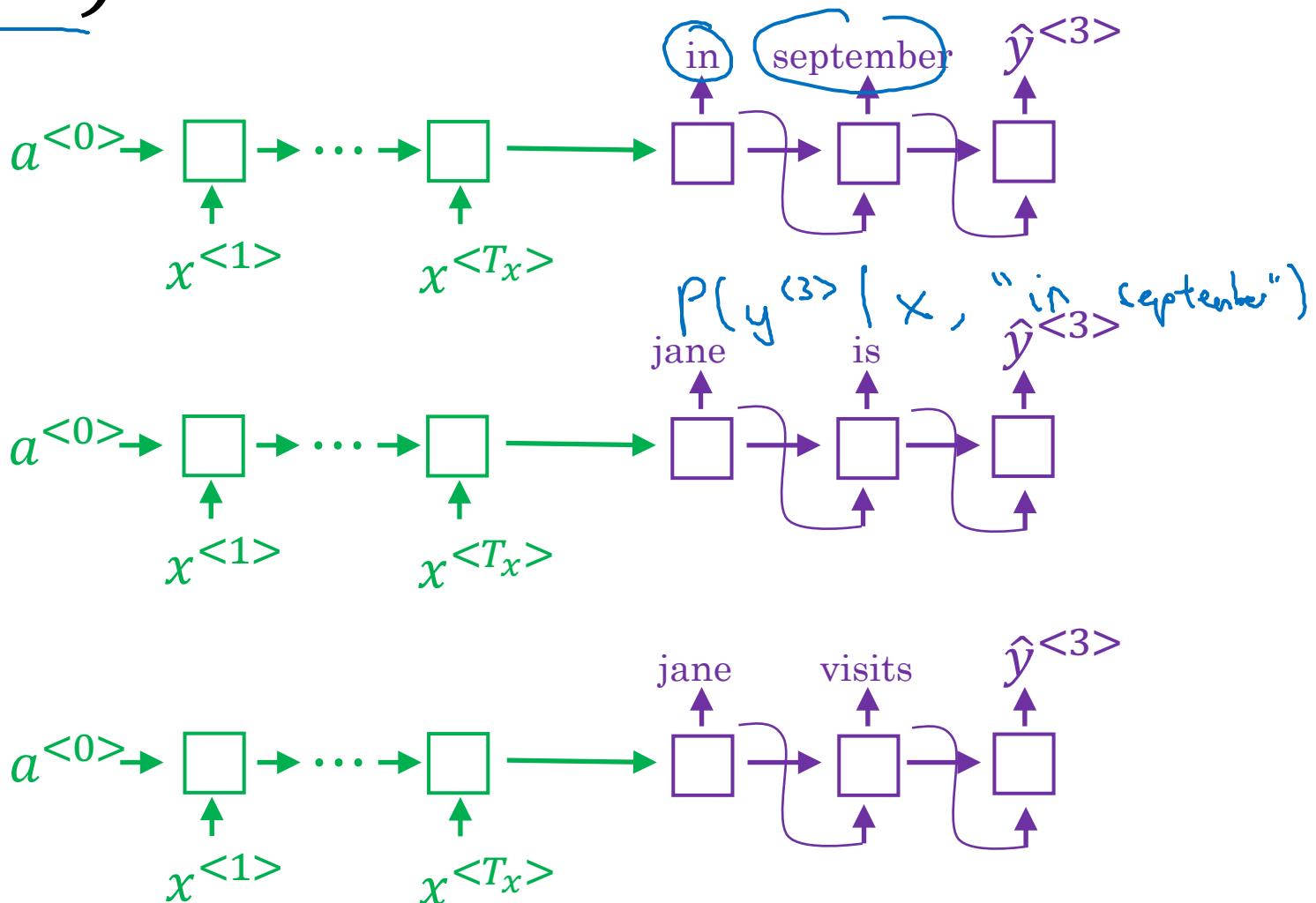
in september

jane is

jane visits

$$P(y^{<1>} , y^{<2>} | x)$$

$B=1 \rightsquigarrow$  greedy search



jane visits africa in september. <EOS>



deeplearning.ai

# Sequence to sequence models

---

## Refinements to beam search

# Length normalization

$$\arg \max_y \prod_{t=1}^{T_y} P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$$

$P(y^{<1>} \dots y^{<T_y>} | x) = P(y^{<1>} | x) P(y^{<2>} | x, y^{<1>}) \dots P(y^{<T_y>} | x, y^{<1>} \dots, y^{<T_y-1>})$

$$\arg \max_y \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})$$

$\overleftarrow{Ty = 1, 2, 3, \dots, 30.}$

$$\rightarrow \boxed{\frac{1}{T_y^\alpha} \sum_{t=1}^{T_y} \log P(y^{<t>} | x, y^{<1>}, \dots, y^{<t-1>})}$$

$$\alpha = 0.7$$

$$\frac{d=1}{d=0}$$

# Beam search discussion

Beam width B?

$1 \rightarrow 3 \rightarrow 10, \quad 100, \quad 1000 \rightarrow 3000$

large B: better result, slower  
small B: worse result, faster

Unlike exact search algorithms like BFS (Breadth First Search) or DFS (Depth First Search), Beam Search runs faster but is not guaranteed to find exact maximum for  $\arg \max_y P(y|x)$ .

$y$



deeplearning.ai

# Sequence to sequence models

---

## Error analysis on beam search

# Example

Jane visite l'Afrique en septembre.

→ RNN

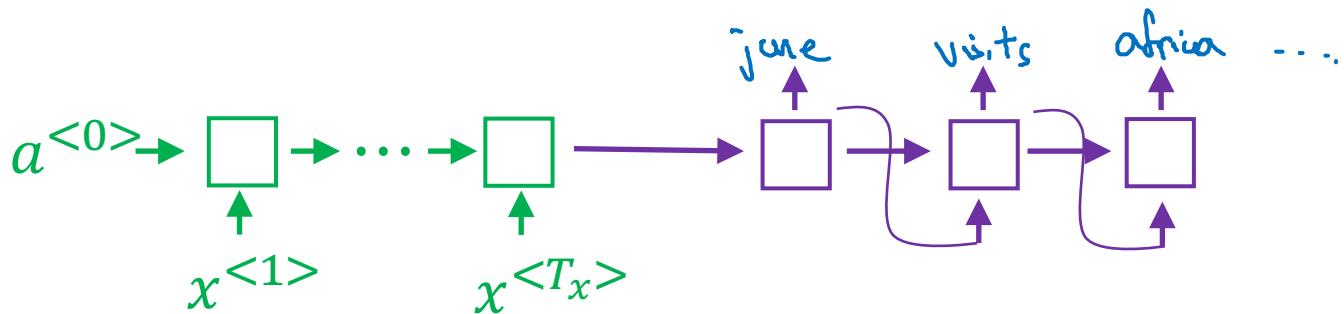
→ Beam Search

B↑

Human: Jane visits Africa in September. ( $y^*$ )

Algorithm: Jane visited Africa last September. ( $\hat{y}$ ) ←

$$\text{RNN} \text{ computes } P(y^*|x) \geq P(\hat{y}|x)$$



# Error analysis on beam search

Human: Jane visits Africa in September. ( $y^*$ )

$$P(y^*|x)$$

Algorithm: Jane visited Africa last September. ( $\hat{y}$ )

$$P(\hat{y}|x)$$

Case 1:  $P(y^*|x) > P(\hat{y}|x) \leftarrow$

$$\arg \max_y P(y|x)$$

Beam search chose  $\hat{y}$ . But  $y^*$  attains higher  $P(y|x)$ .

Conclusion: Beam search is at fault.

Case 2:  $P(y^*|x) \leq P(\hat{y}|x) \leftarrow$

$y^*$  is a better translation than  $\hat{y}$ . But RNN predicted  $P(y^*|x) < P(\hat{y}|x)$ .

Conclusion: RNN model is at fault.

# Error analysis process

| Human                            | Algorithm                           | $P(y^* x)$                            | $P(\hat{y} x)$                        | At fault? |
|----------------------------------|-------------------------------------|---------------------------------------|---------------------------------------|-----------|
| Jane visits Africa in September. | Jane visited Africa last September. | <u><math>2 \times 10^{-10}</math></u> | <u><math>1 \times 10^{-10}</math></u> | B         |
| ...                              | ...                                 | —                                     | —                                     | R         |
| ...                              | ...                                 | —                                     | —                                     | R         |
|                                  |                                     |                                       | —                                     | R         |
|                                  |                                     |                                       |                                       | :         |

Figures out what fraction of errors are “due to” beam search vs. RNN model



deeplearning.ai

# Sequence to sequence models

---

Bleu score  
(optional)

# Evaluating machine translation

French: Le chat est sur le tapis.

Reference 1: The cat is on the mat. ←

Reference 2: There is a cat on the mat. ←

MT output: the the the the the the.

Precision:

Modified precision:

Bleu  
bilingual evaluation understudy

# Bleu score on bigrams

Example: Reference 1: The cat is on the mat. ←

Reference 2: There is a cat on the mat. ←

MT output: The cat the cat on the mat. ←

|         | Count | Count <sub>clip</sub> |  |
|---------|-------|-----------------------|--|
| the cat | 2 ←   | 1 ←                   |  |
| cat the | 1 ←   | 0                     |  |
| cat on  | 1 ←   | 1 ←                   |  |
| on the  | 1 ←   | 1 ←                   |  |
| the mat | 1 ←   | 1 ←                   |  |

$$\frac{4}{6}$$

# Bleu score on unigrams

Example: Reference 1: The cat is on the mat.

$$P_1, P_2 = 1.0$$

Reference 2: There is a cat on the mat.

→ MT output: The cat the cat on the mat. (↑)

$$P_1 = \frac{\sum_{\text{unigrams} \in \hat{y}} \text{count}_{clip}(\text{unigram})}{\sum_{\text{unigrams} \in \hat{y}} \text{count}(\text{unigram})}$$

↑  
Unigram

Count (unigram) Count (unigram)

count<sub>clip</sub>(unigram) count<sub>clip</sub>(unigram)

unigrams n-grams

$$p_n = \frac{\sum_{\text{n-gram} \in \hat{y}} \text{count}_{clip}(\text{n-gram})}{\sum_{\text{n-gram} \in \hat{y}} \text{count}(\text{n-gram})}$$

↑  
n-gram

Count (n-gram) Count (n-gram)

count<sub>clip</sub>(n-gram) count<sub>clip</sub>(n-gram)

n-grams n-grams

# Bleu details

$p_n$  = Bleu score on n-grams only

$P_1, P_2, P_3, P_4$

Combined Bleu score:

$$BP \exp\left(\frac{1}{4} \sum_{n=1}^4 p_n\right)$$

$BP$  = brevity penalty

$$BP = \begin{cases} 1 & \text{if } \underline{\text{MT\_output\_length}} > \underline{\text{reference\_output\_length}} \\ \exp(1 - \text{MT\_output\_length}/\text{reference\_output\_length}) & \text{otherwise} \end{cases}$$





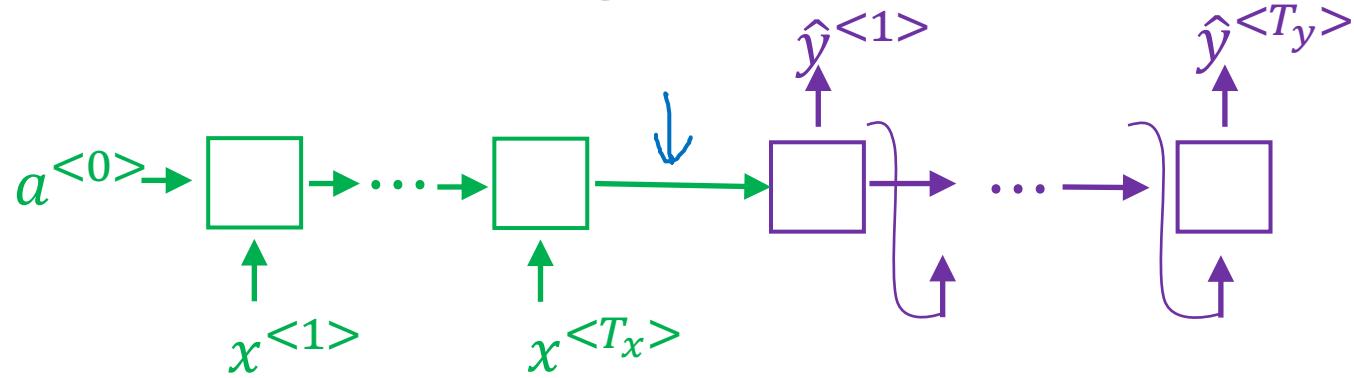
deeplearning.ai

# Sequence to sequence models

---

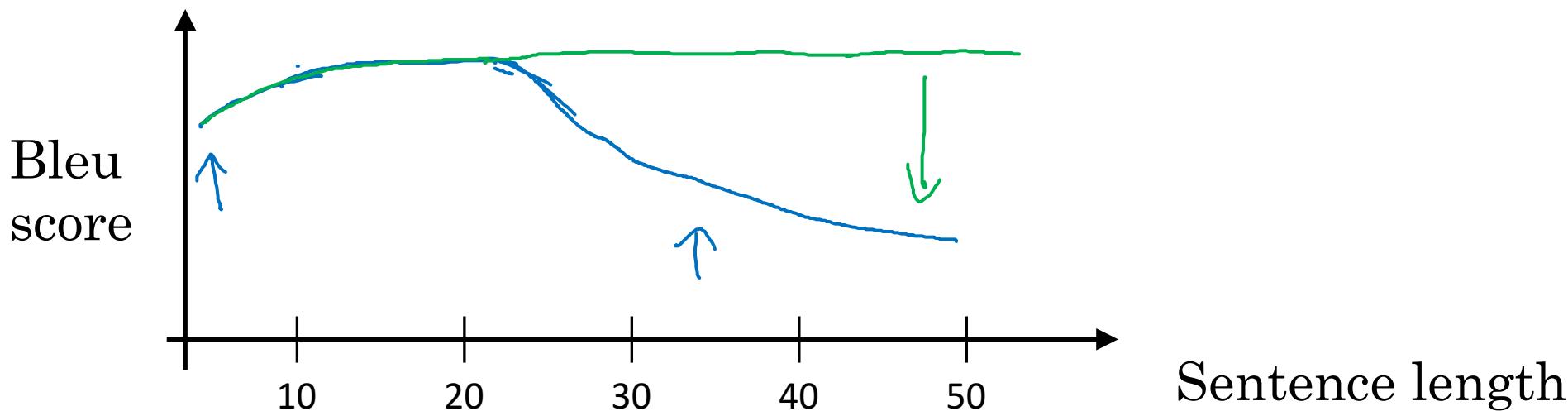
## Attention model intuition

# The problem of long sequences

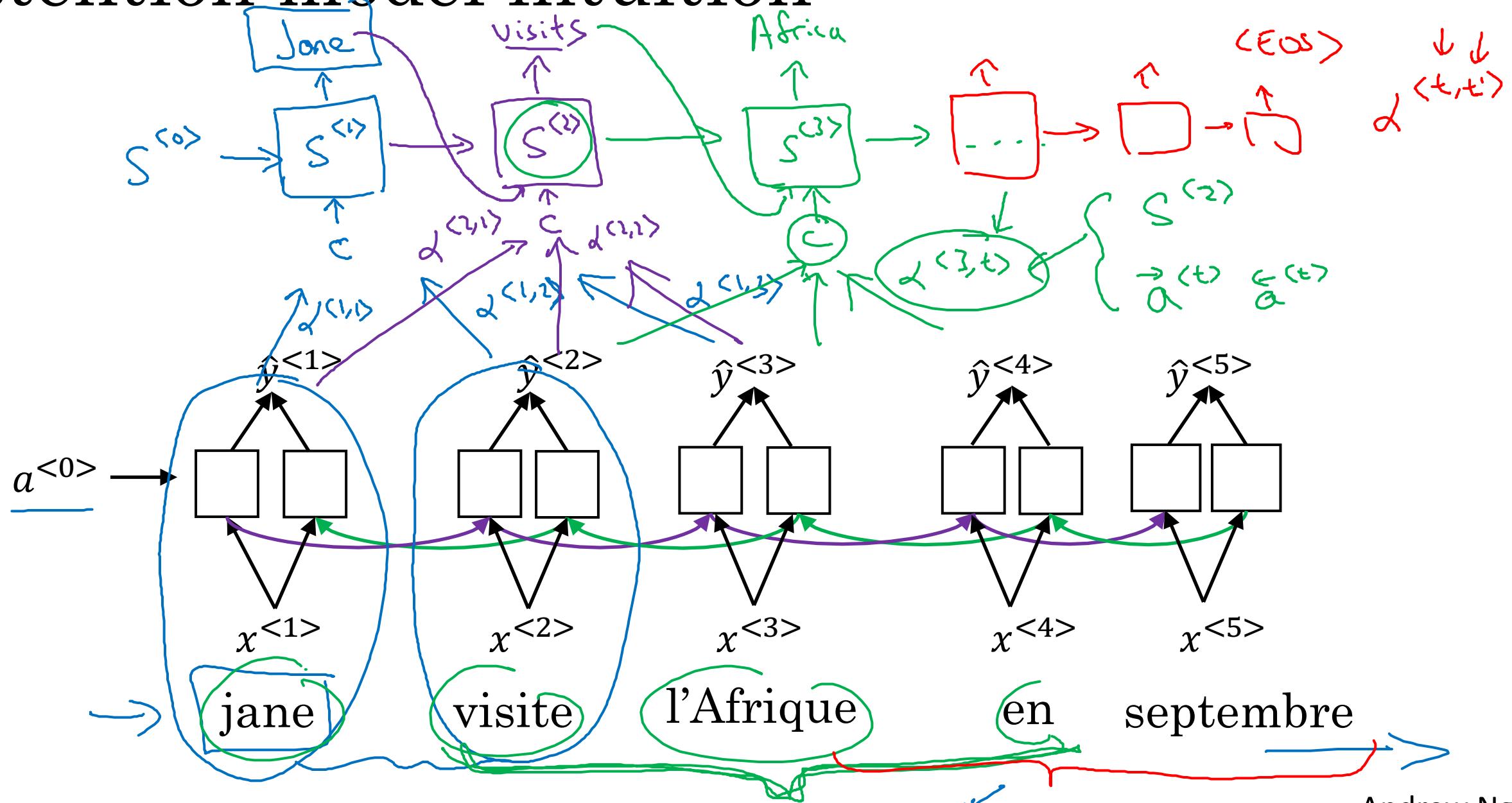


Jane s'est rendue en Afrique en septembre dernier, a apprécié la culture et a rencontré beaucoup de gens merveilleux; elle est revenue en parlant comment son voyage était merveilleux, et elle me tente d'y aller aussi.

Jane went to Africa last September, and enjoyed the culture and met many wonderful people; she came back raving about how wonderful her trip was, and is tempting me to go too.



# Attention model intuition





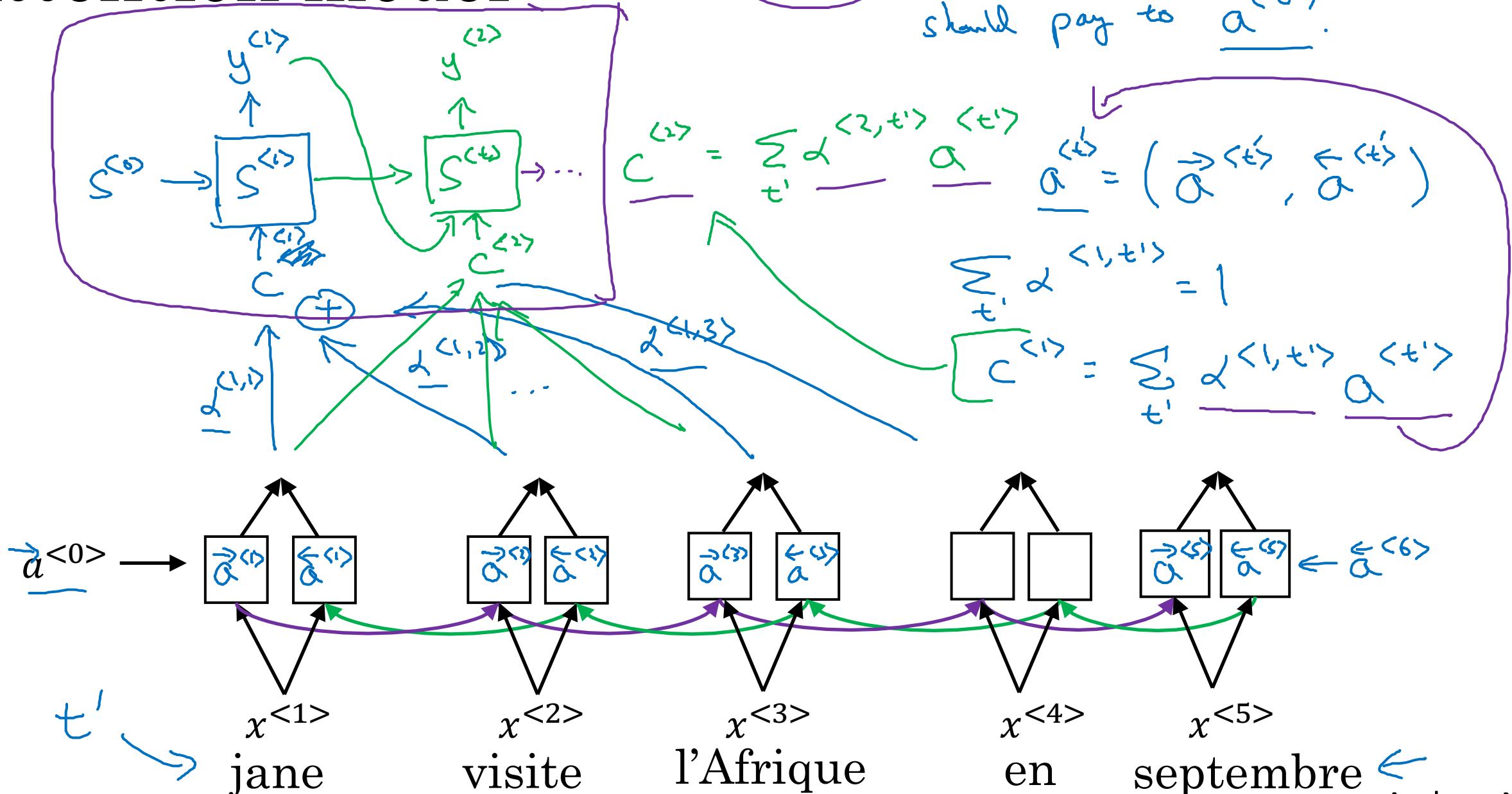
deeplearning.ai

# Sequence to sequence models

---

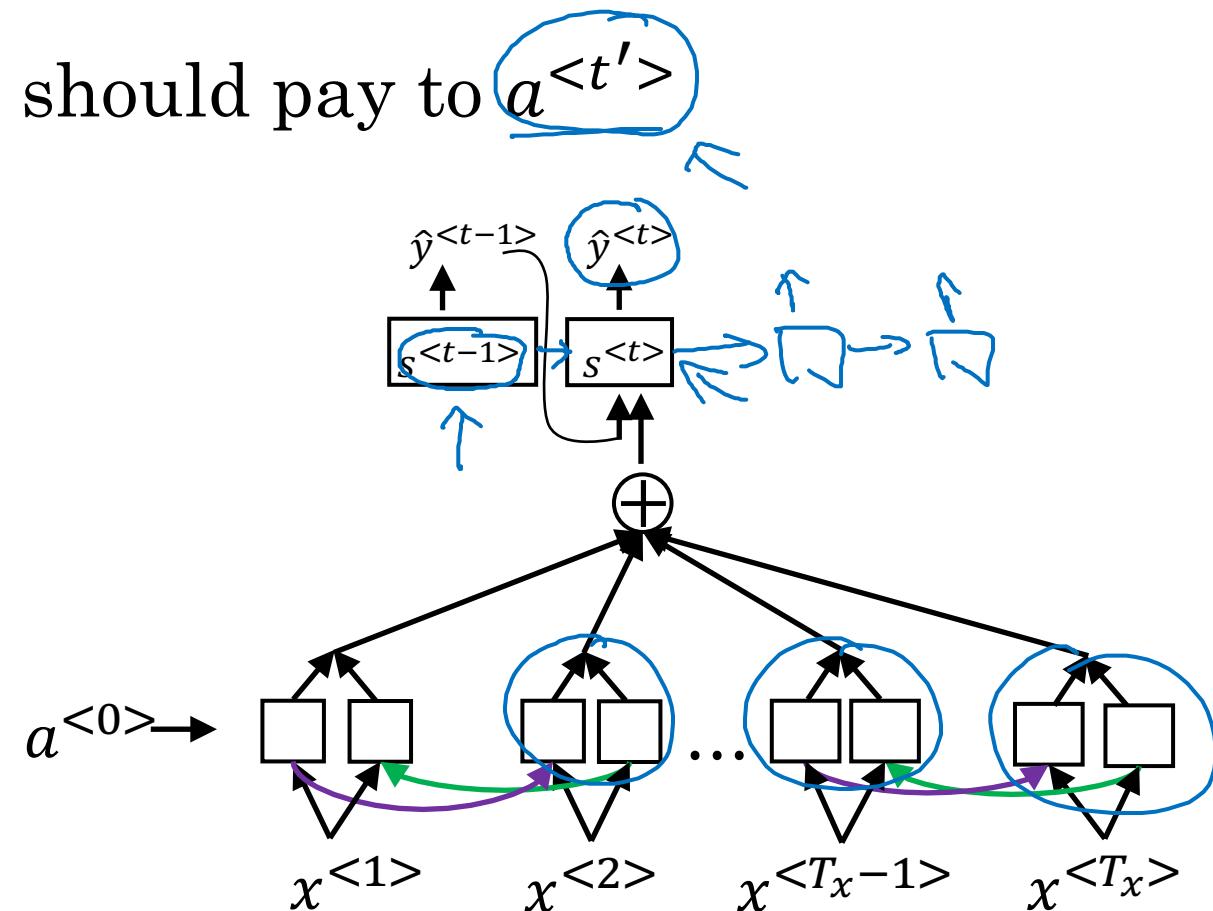
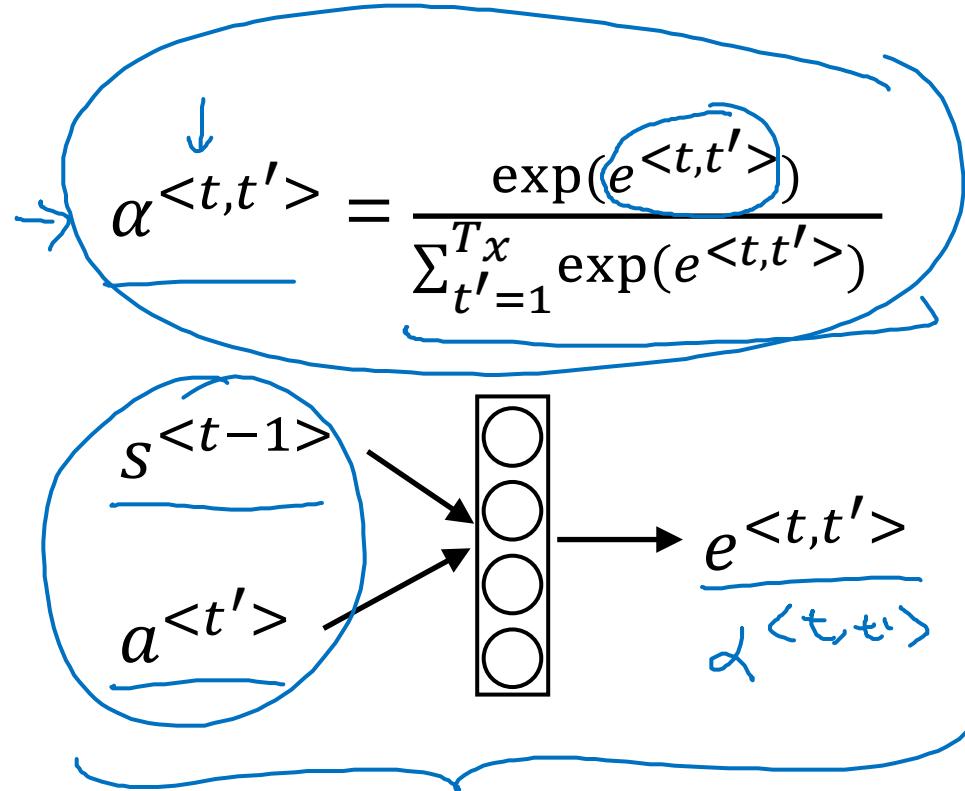
## Attention model

# Attention model



# Computing attention $\underline{\alpha^{'}}$

$\alpha^{'}$  = amount of attention  $y^{'}$  should pay to  $a^{'}$



[Bahdanau et. al., 2014. Neural machine translation by jointly learning to align and translate]

[Xu et. al., 2015. Show, attend and tell: Neural image caption generation with visual attention]

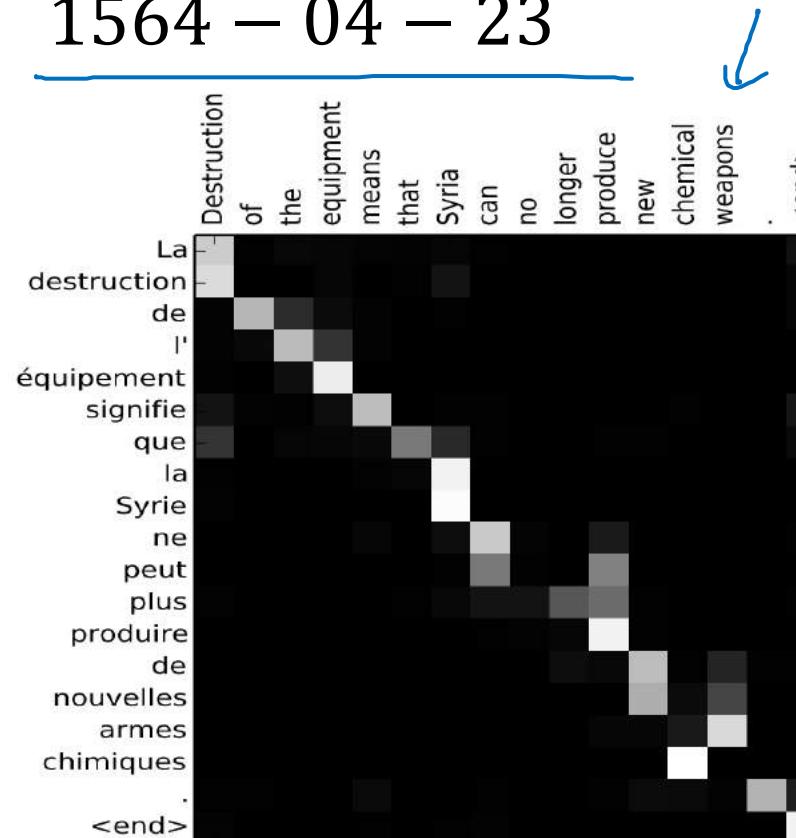
Andrew Ng

# Attention examples

July 20th 1969 → 1969 – 07 – 20

23 April, 1564 → 1564 – 04 – 23

Visualization of  $\alpha^{<t,t'>}$ :





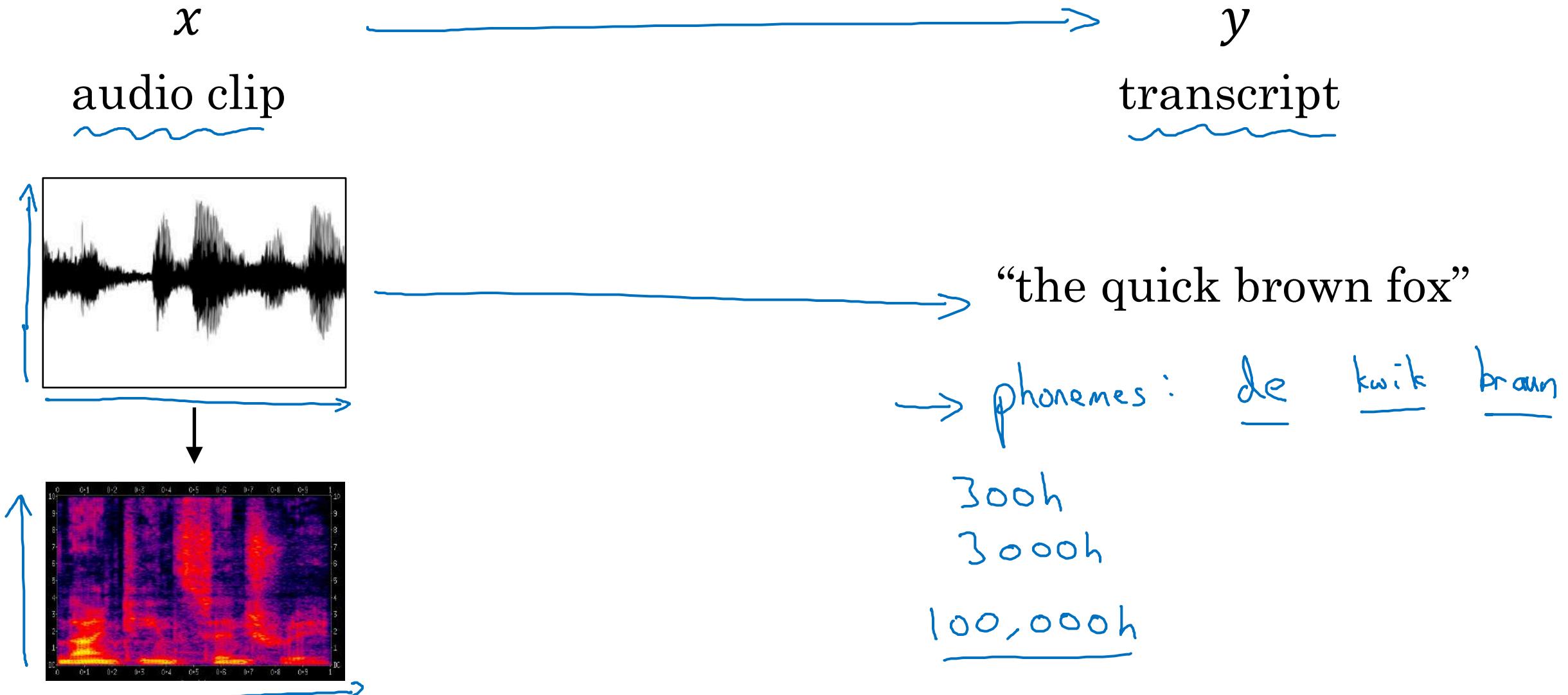
deeplearning.ai

Audio data

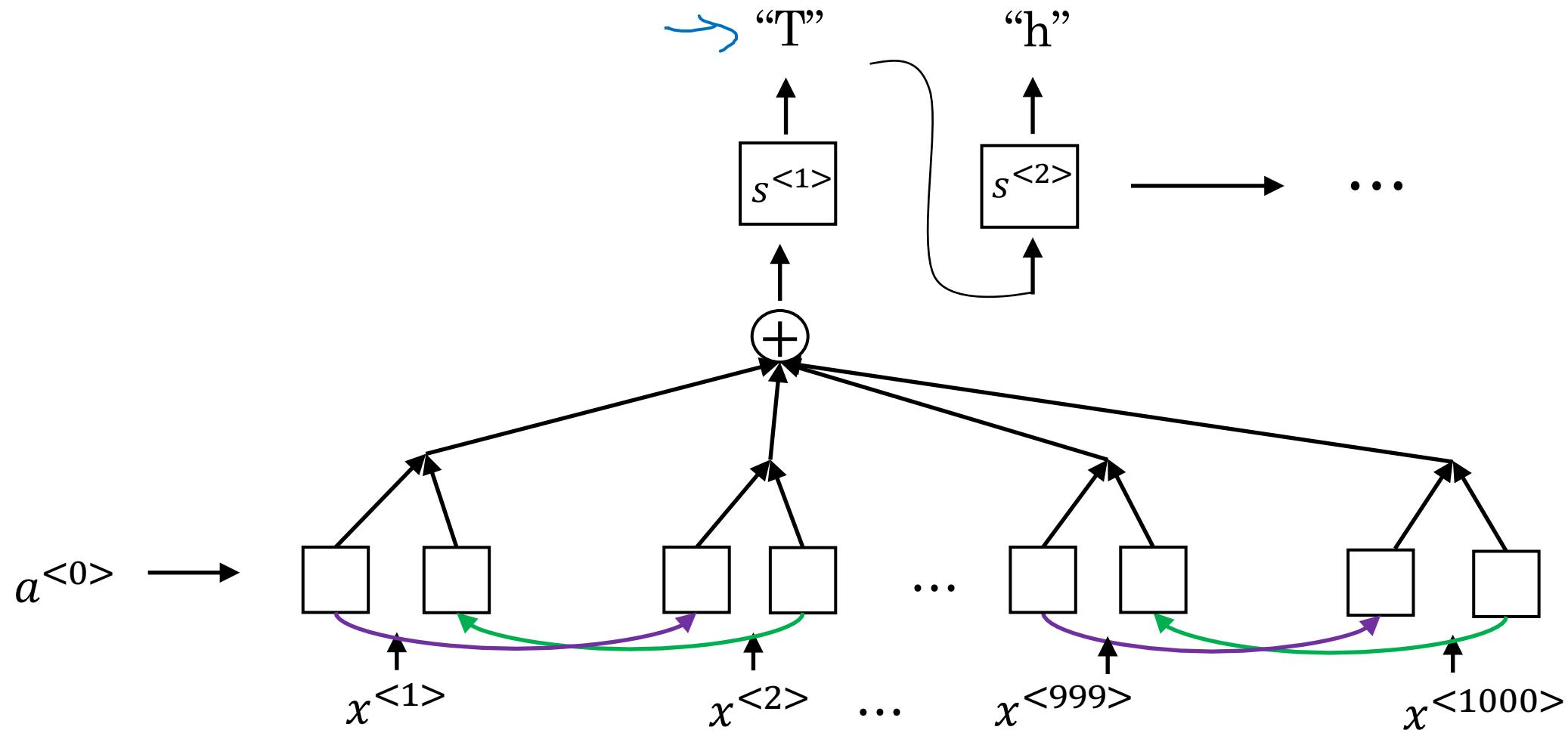
---

Speech recognition

# Speech recognition problem

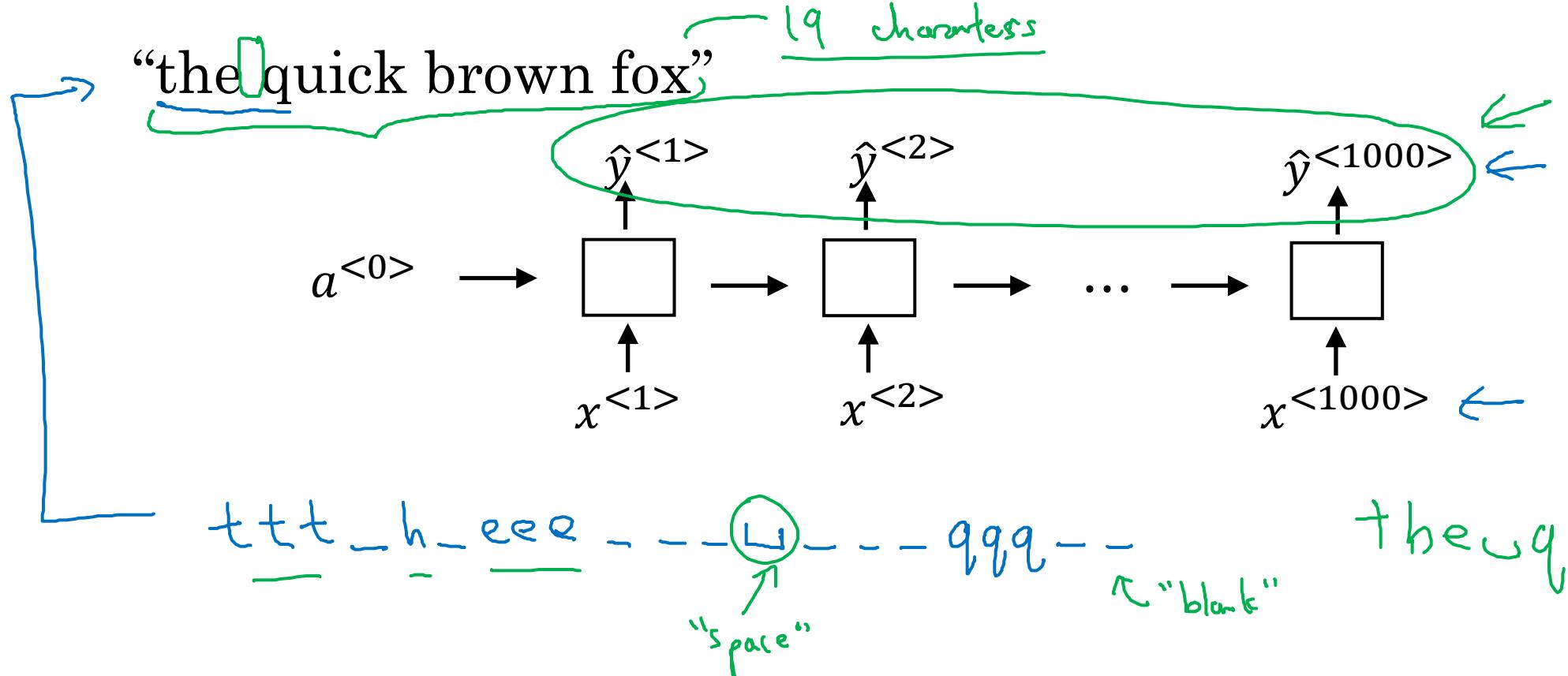


# Attention model for speech recognition



# CTC cost for speech recognition

(Connectionist temporal classification)



Basic rule: collapse repeated characters not separated by “blank” ↴



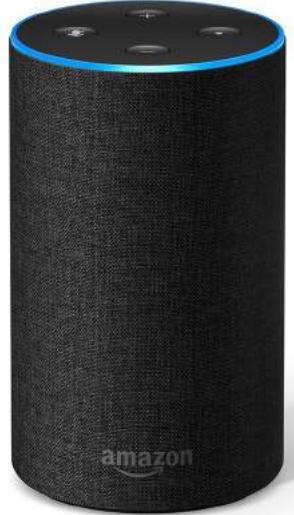
deeplearning.ai

Audio data

---

Trigger word  
detection

# What is trigger word detection?



Amazon Echo  
(Alexa)



Baidu DuerOS  
(xiaodunihao)

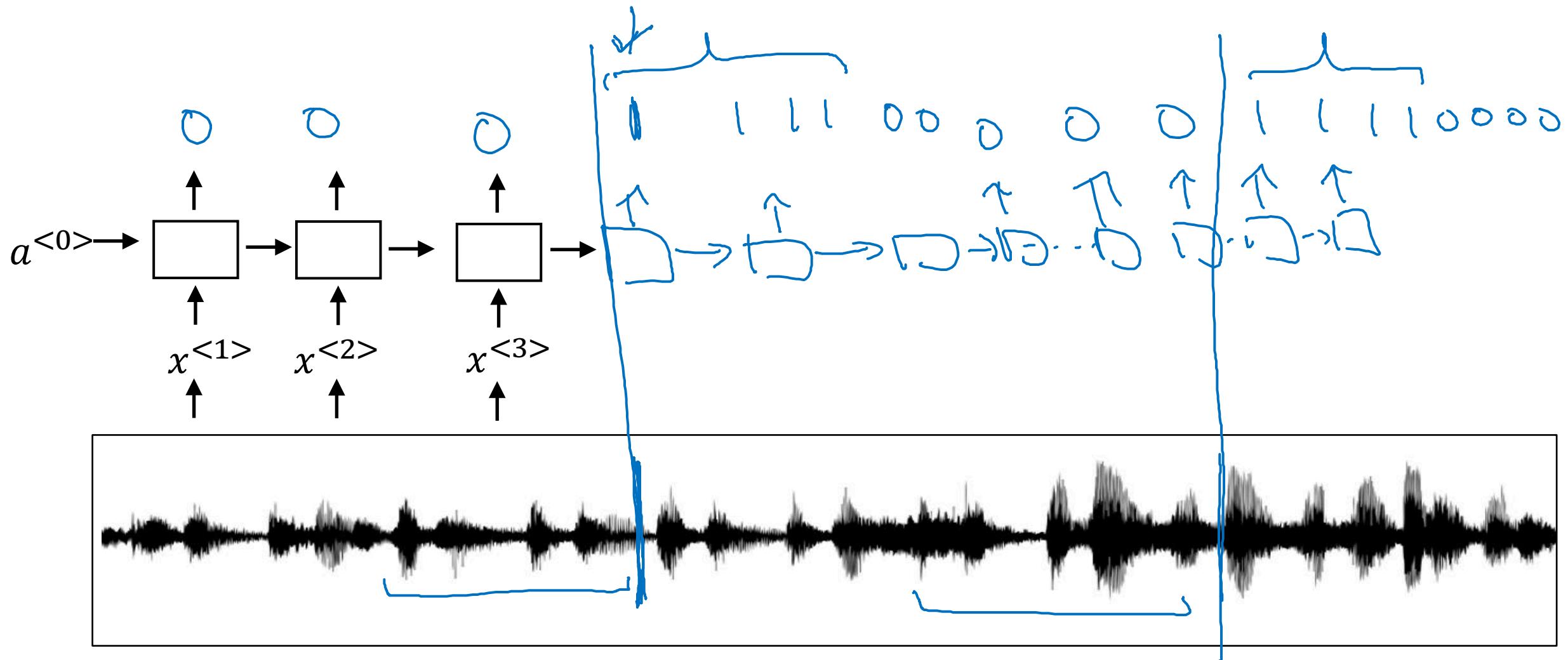


Apple Siri  
(Hey Siri)



Google Home  
(Okay Google)

# Trigger word detection algorithm





deeplearning.ai

# Conclusion

---

## Summary and thank you

# Specialization outline

1. Neural Networks and Deep Learning
2. Improving Deep Neural Networks: Hyperparameter tuning, Regularization and Optimization
3. Structuring Machine Learning Projects
4. Convolutional Neural Networks
5. Sequence Models

# Deep learning is a super power

Please buy this  
from shutterstock  
and replace in  
final video.



[www.shutterstock.com](http://www.shutterstock.com) • 331201091

Thank you.

- Andrew Ng

# 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

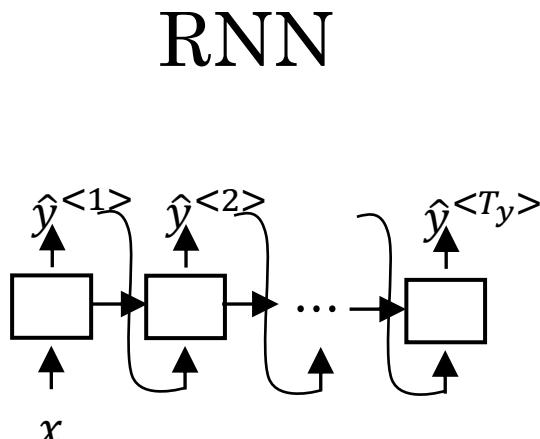
# Sequence to sequence models

---

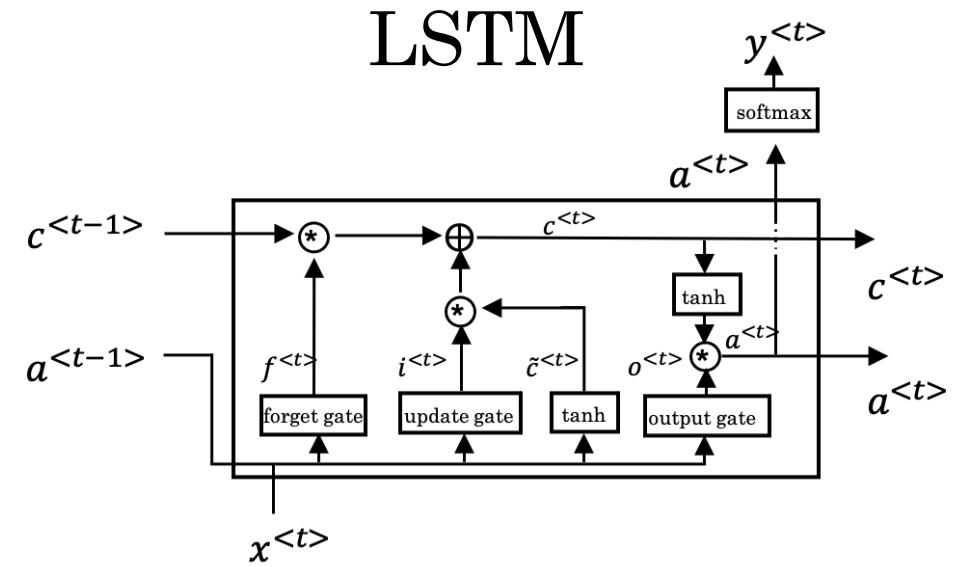
## Transformers Intuition

# Transformers Motivation

Increased complexity,  
sequential

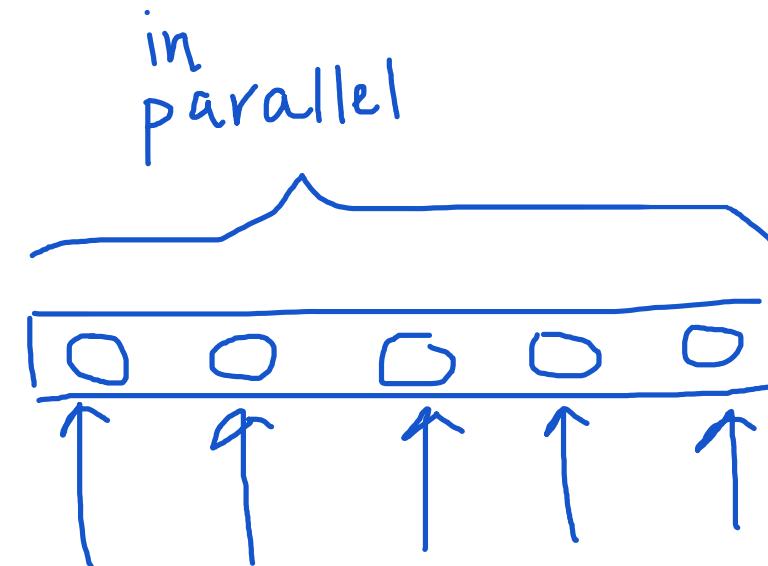
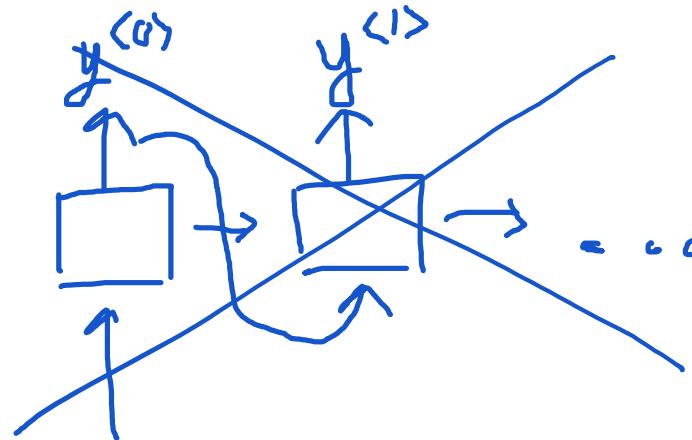


GRU



# Transformers Intuition

- Attention + CNN
  - Self-Attention
  - Multi-Head Attention





deeplearning.ai

# Sequence to sequence models

---

## Self-Attention

# Self-Attention Intuition

$A(q, K, V)$ = attention-based vector representation of a word  
calculate for each word

## RNN Attention

$$\alpha^{<t,t'>} = \frac{\exp(e^{$$

$x^{<1>}$   
Jane

$x^{<2>}$   
visite

$x^{<3>}$   
l'Afrique

$x^{<4>}$   
en

$x^{<5>}$   
septembre

## Transformers Attention

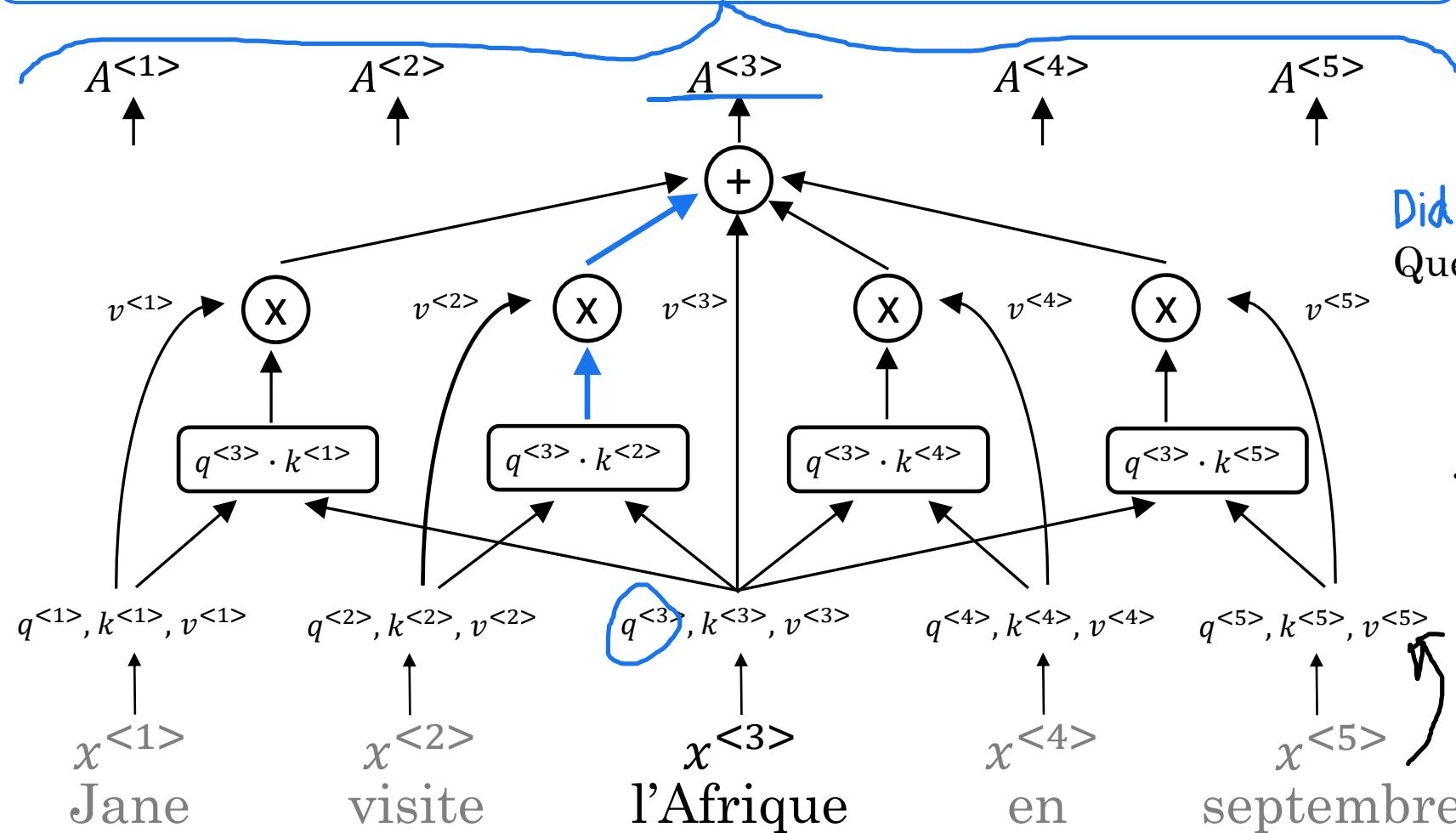
$$A(q, K, V) = \sum_i \frac{\exp(e^{}})}{\sum_j \exp(e^{}})} v^{<i>}$$

# Self-Attention

$$A(q, K, V) = \sum_i \frac{\exp(e^{q \cdot k^{<i>}})}{\sum_j \exp(e^{q \cdot k^{<j>}})} v^{<i>}$$

*Softmax*

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$



Did what?

Query ( $Q$ )

$q^{<1>}$   
 $q^{<2>}$   
 $q^{<3>}$   
 $\underline{q^{<4>}}$   
 $q^{<5>}$

Key ( $K$ )

$k^{<1>}$   
 $k^{<2>}$   
 $k^{<3>}$   
 $\cancel{k^{<4>}}$   
 $k^{<5>}$

Value ( $V$ )

$v^{<1>} \text{ person}$   
 $v^{<2>} \text{ action}$   
 $v^{<3>} \text{ visit}$   
 $v^{<4>}$   
 $v^{<5>}$

What's happening here?



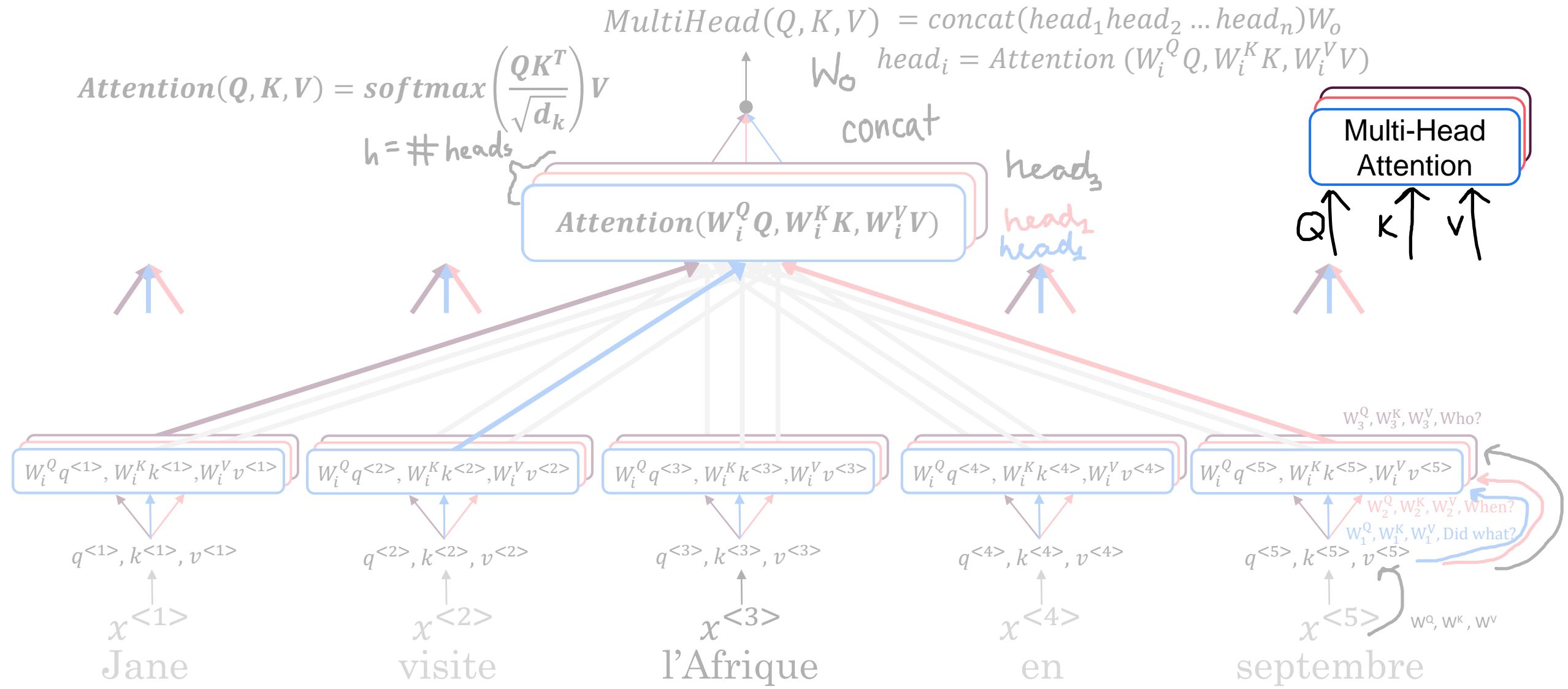
deeplearning.ai

# Sequence to sequence models

---

## Multi-Head Attention

# Multi-Head Attention





deeplearning.ai

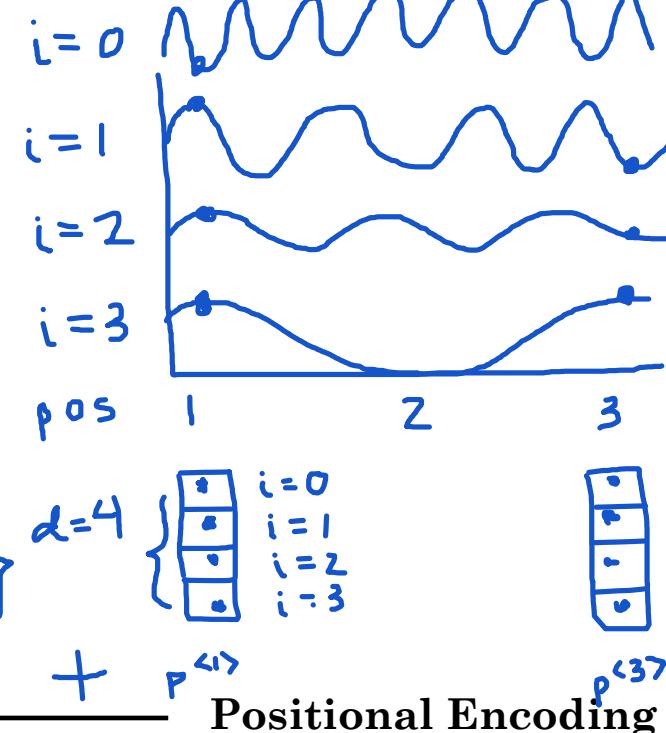
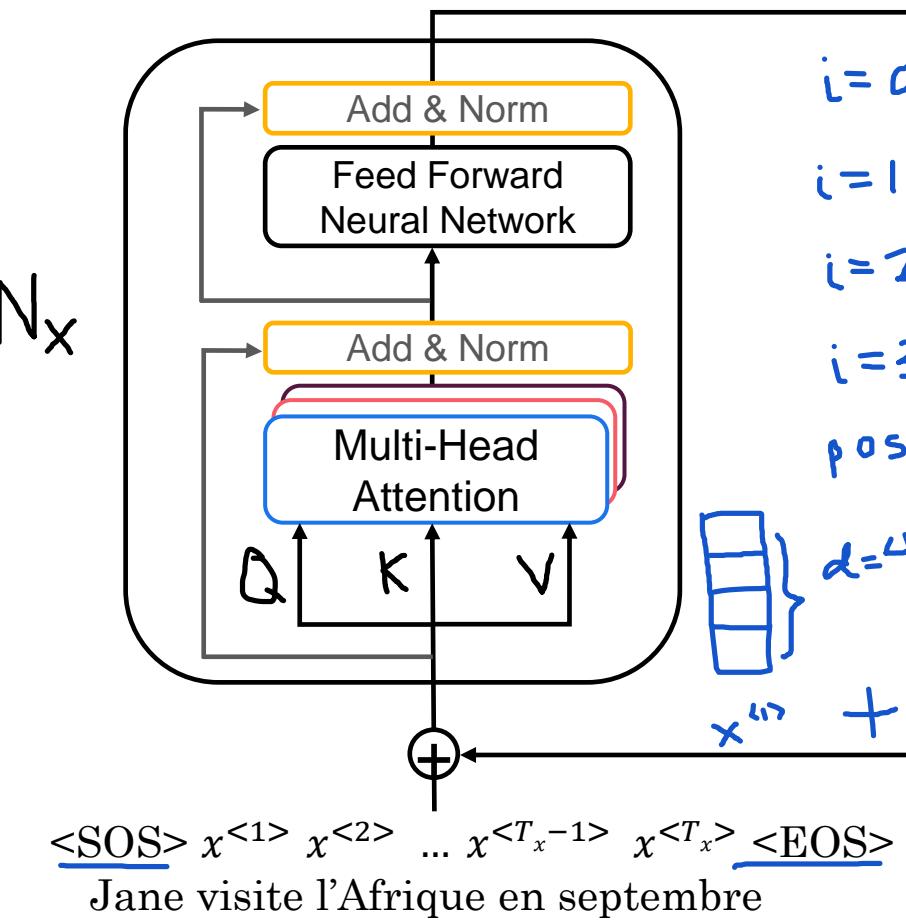
# Sequence to sequence models

---

## Transformers

# Transformer Details

## Encoder

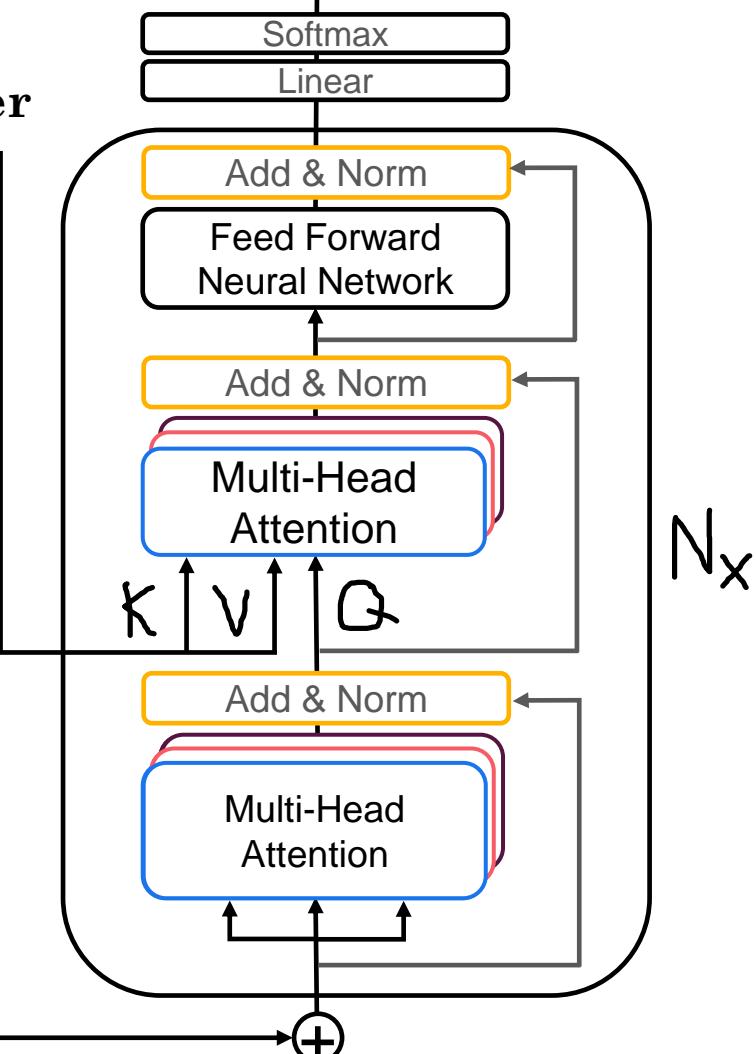


$$PE_{(pos,2i)} = \sin\left(\frac{pos}{1000^d}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{1000^d}\right)$$

<SOS> Jane visits Africa in September <EOS>

## Decoder



<SOS> Jane visits Africa in September