



deeplearning.ai

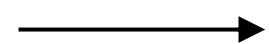
# Convolutional Neural Networks

---

## Computer vision

# Computer Vision Problems

Image Classification



Cat? (0/1)



Neural Style Transfer



Object detection



Andrew Ng

# Deep Learning on large images



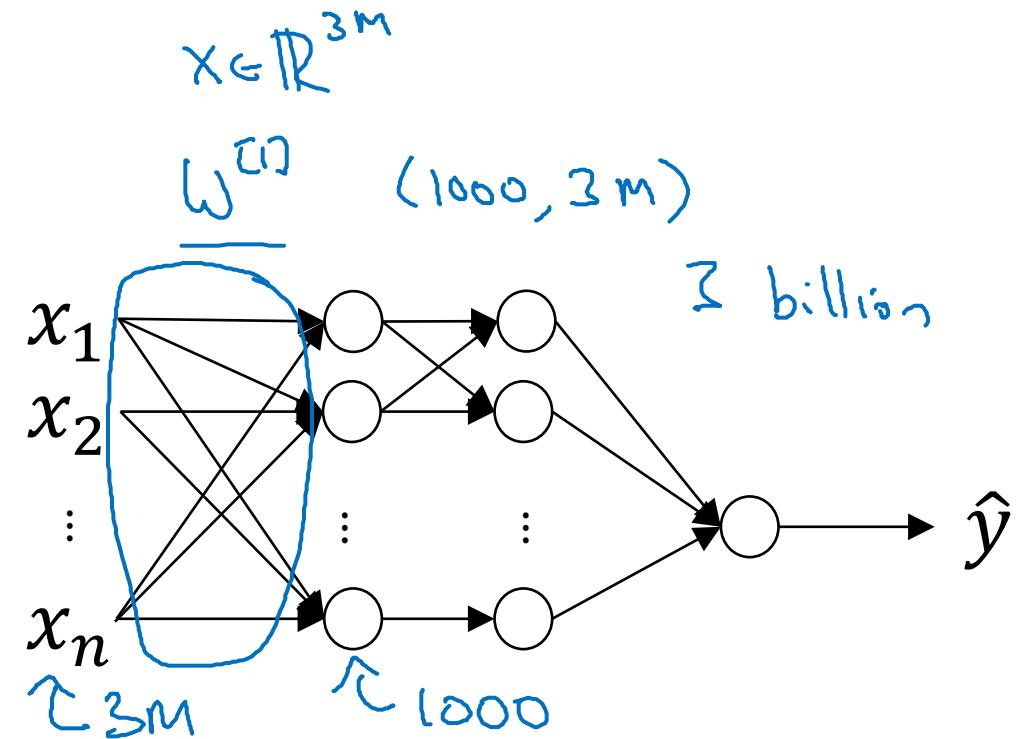
$64 \times 64 \times 3$

→ Cat? (0/1)

12288



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





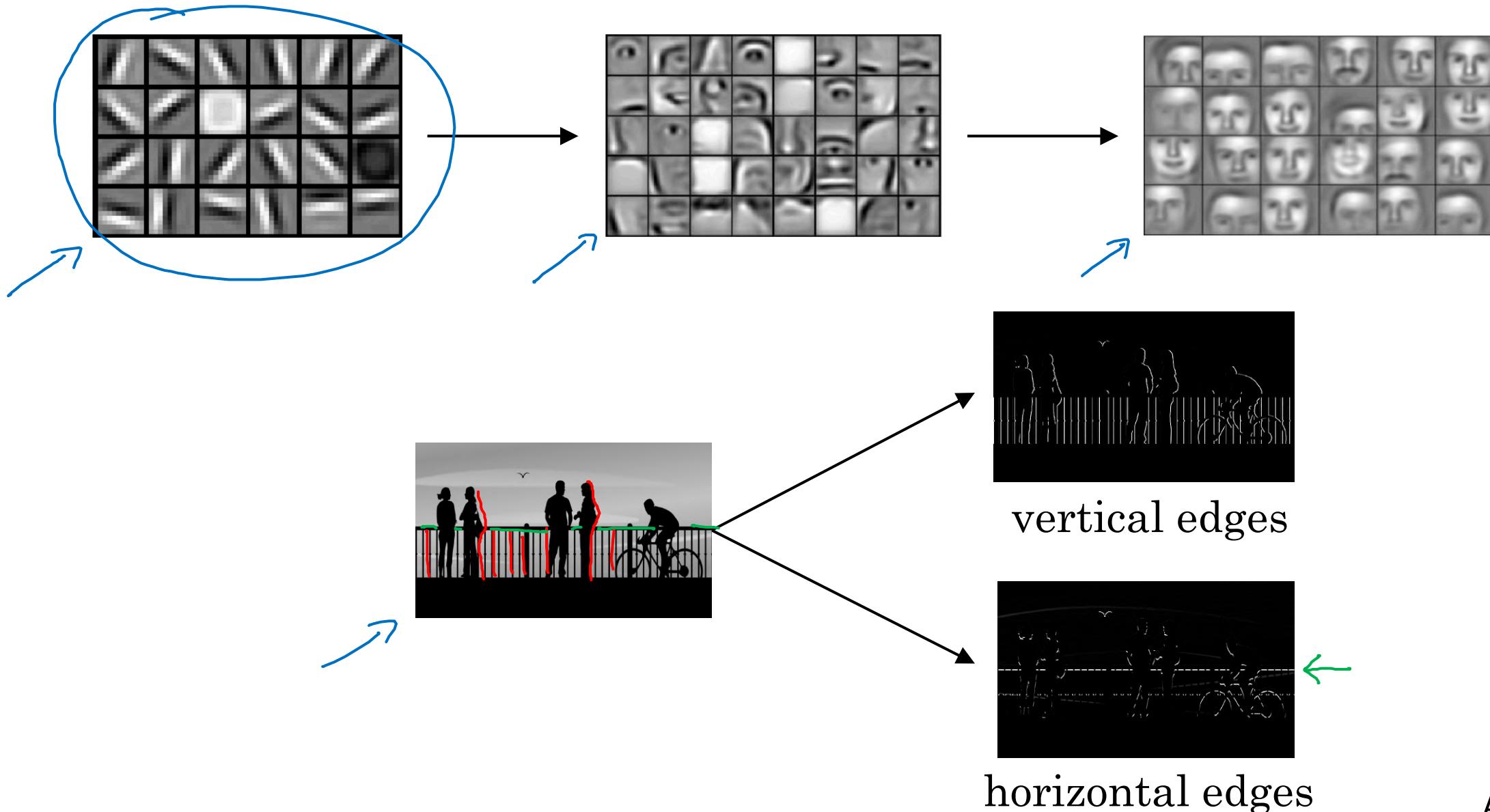
deeplearning.ai

# Convolutional Neural Networks

---

## Edge detection example

# Computer Vision Problem

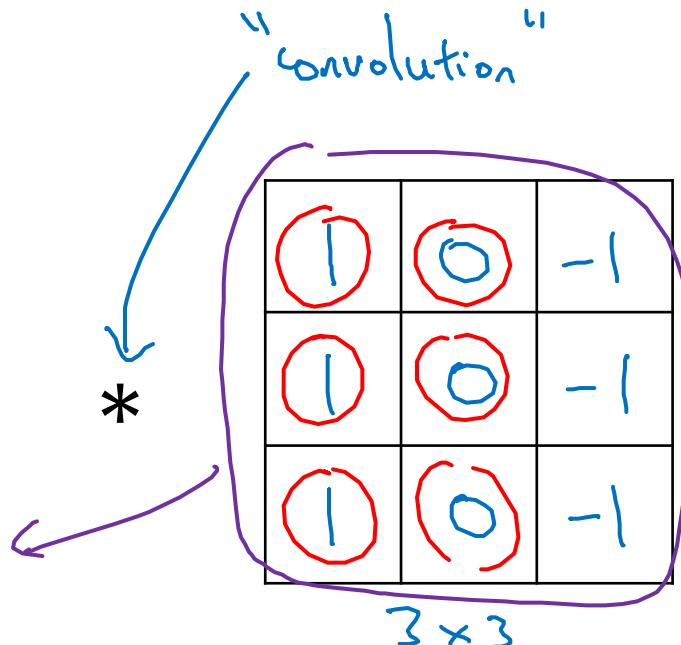


# Vertical edge detection

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

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

$6 \times 6$



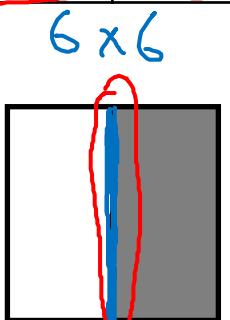
=

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

$4 \times 4$

# 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



\*

$$\begin{matrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{matrix}$$

3x3

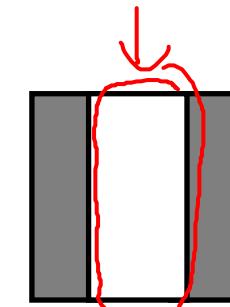
=

$$\begin{matrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{matrix}$$

4x4

\*

↑ ↑ ↑



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

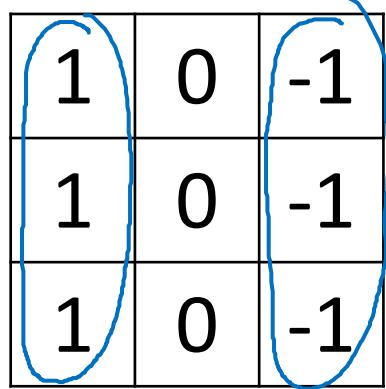


=

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

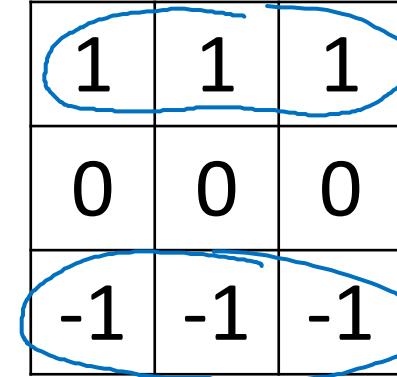


# Vertical and Horizontal Edge Detection



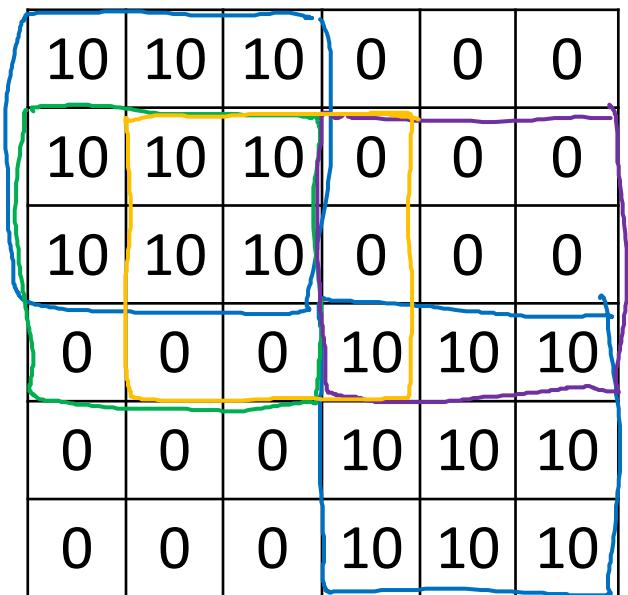
1	0	-1
1	0	-1
1	0	-1

Vertical



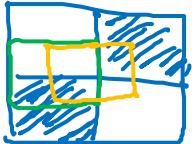
1	1	1
0	0	0
-1	-1	-1

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

$6 \times 6$



\*

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

×

$w_1$	$w_2$	$w_3$
$w_4$	$w_5$	$w_6$
$w_7$	$w_8$	$w_9$

$3 \times 3$

=

$45^\circ$   
 $70^\circ$   
 $73^\circ$

↑


Scharr filter



deeplearning.ai

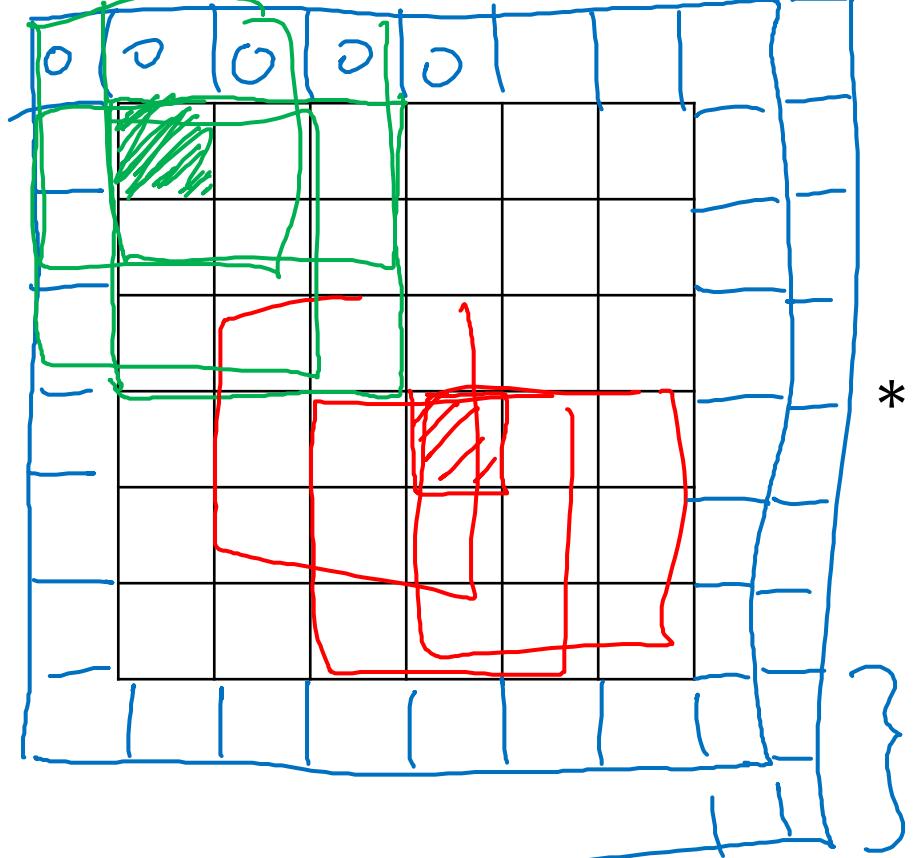
# Convolutional Neural Networks

---

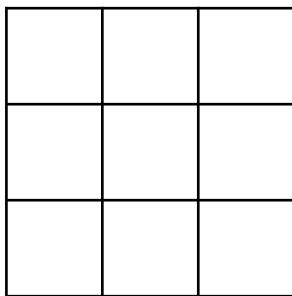
## Padding

# Padding

- shrinky output
- throw away info from edge



\*



$3 \times 3$   
 $f \times f$

 $p=2$ 

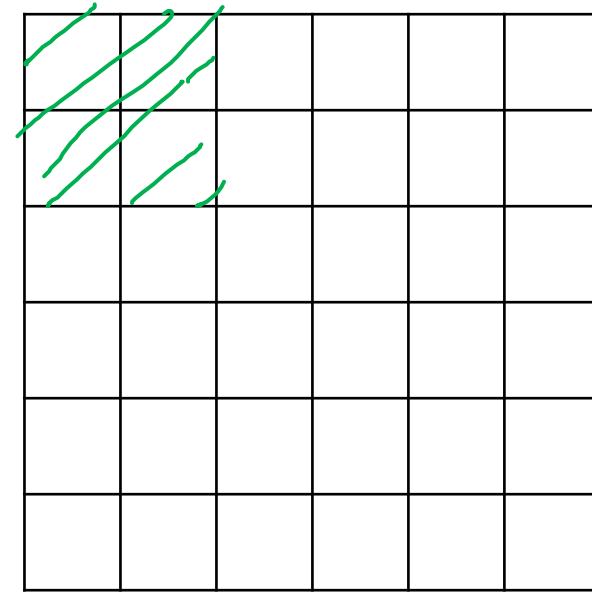
$$\frac{6 \times 6}{n \times n} \rightarrow 8 \times 8$$

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

$$6-3+1 = 4$$

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

=

 $\underline{6 \times 6}$ 

$$\xrightarrow{\quad} \underline{4 \times 4}$$

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

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

# Valid and Same convolutions

→ n → padding

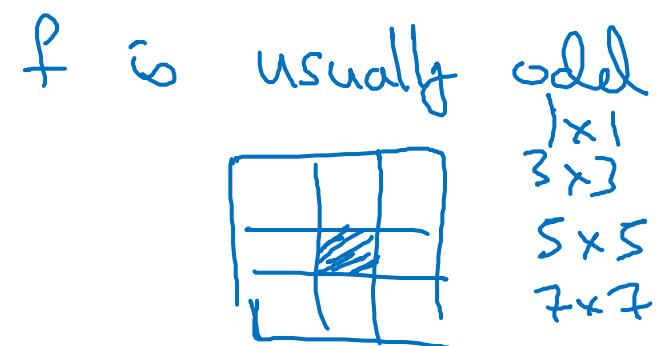
“Valid”:  $n \times n \quad * \quad f \times f \quad \rightarrow \frac{n-f+1}{f} \times \frac{n-f+1}{f}$

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

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

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

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





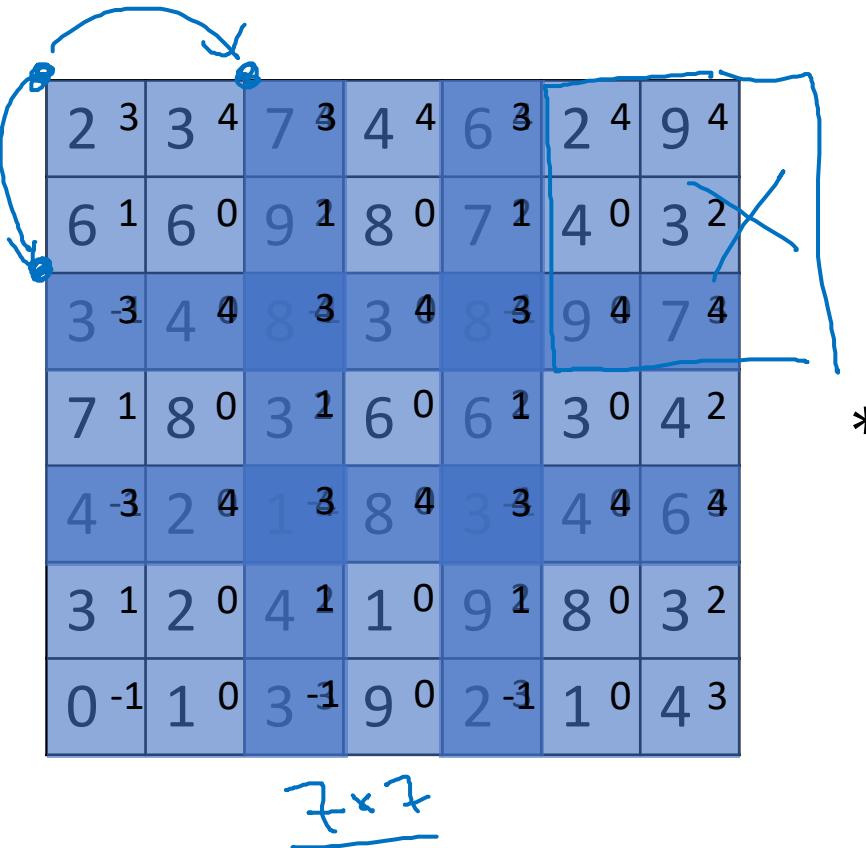
deeplearning.ai

# Convolutional Neural Networks

---

## Strided convolutions

# Strided convolution



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

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

$3 \times 3$

stride = 2

=

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

$3 \times 3$

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

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

$$\left\lfloor \frac{7+0-3}{2} + 1 \right\rfloor = \frac{4}{2} + 1 = 3$$

# 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	

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

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

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



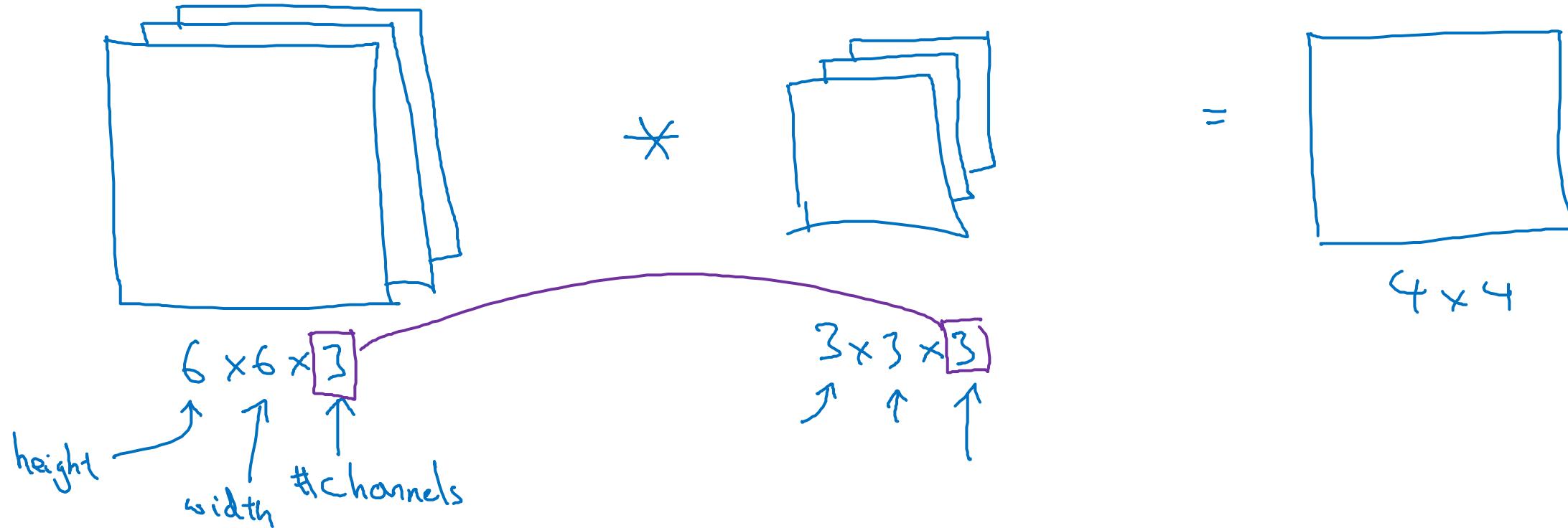
deeplearning.ai

# Convolutional Neural Networks

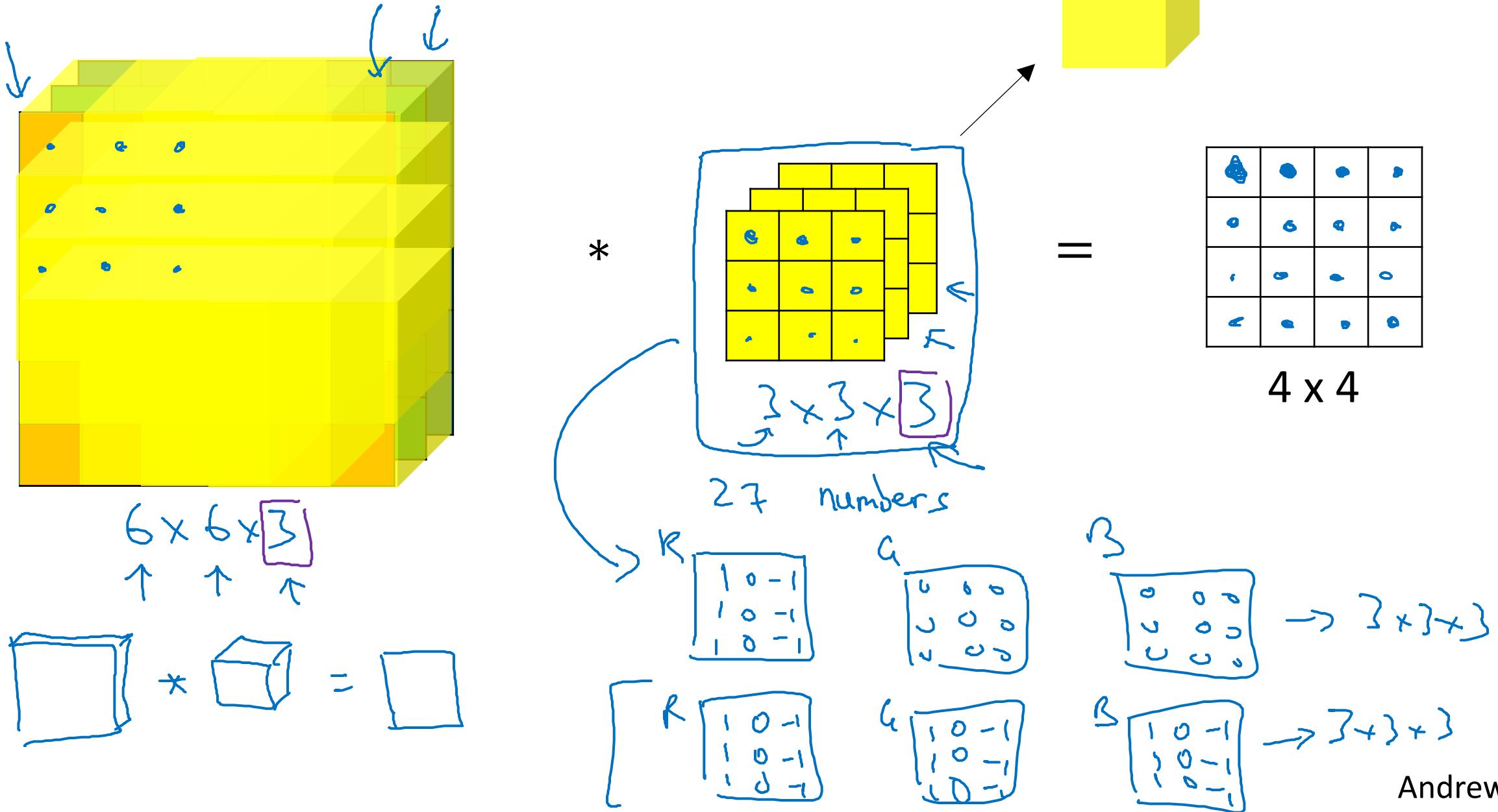
---

## Convolutions over volumes

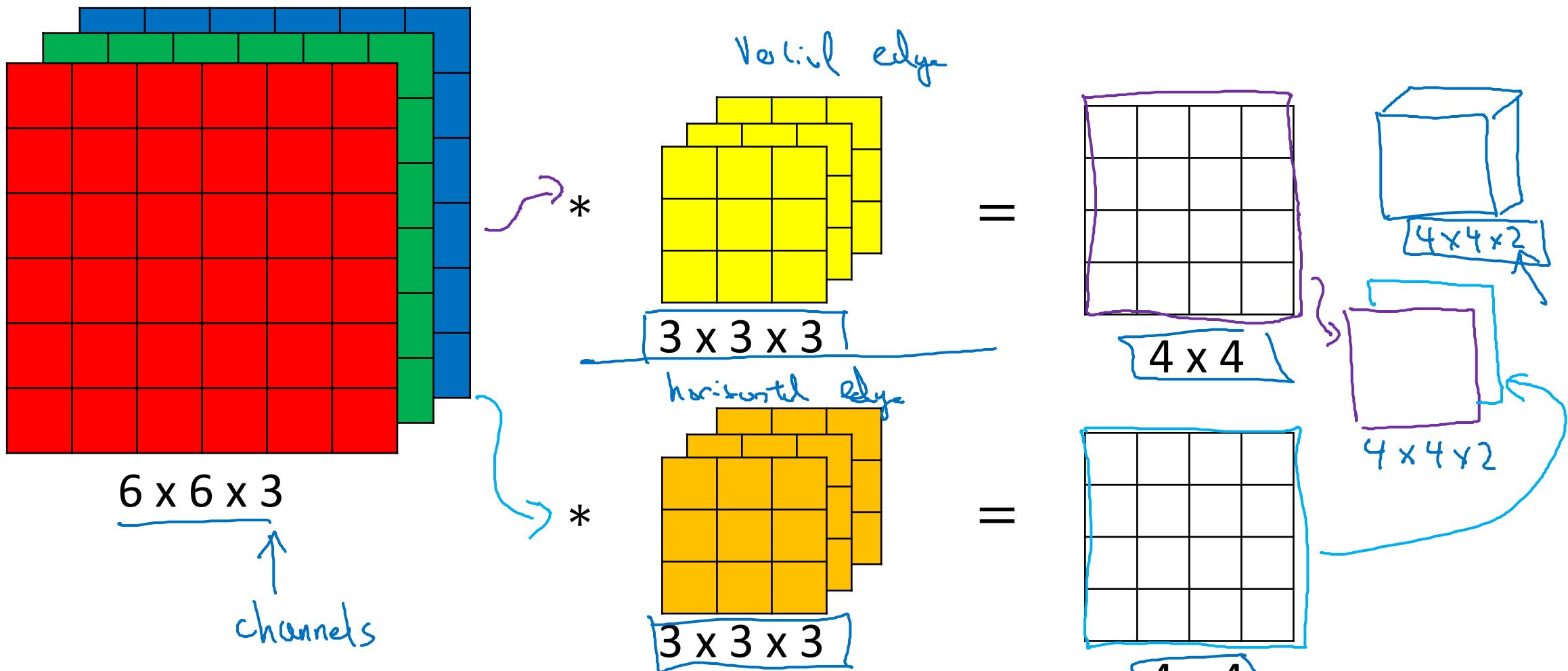
# Convolutions on RGB images



# Convolutions on RGB image



# Multiple filters



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

$6 \times 6 \times 3 \quad 3 \times 3 \times 3 \quad \frac{4-3+1}{4} \times \frac{4-3+1}{4} \times 2$



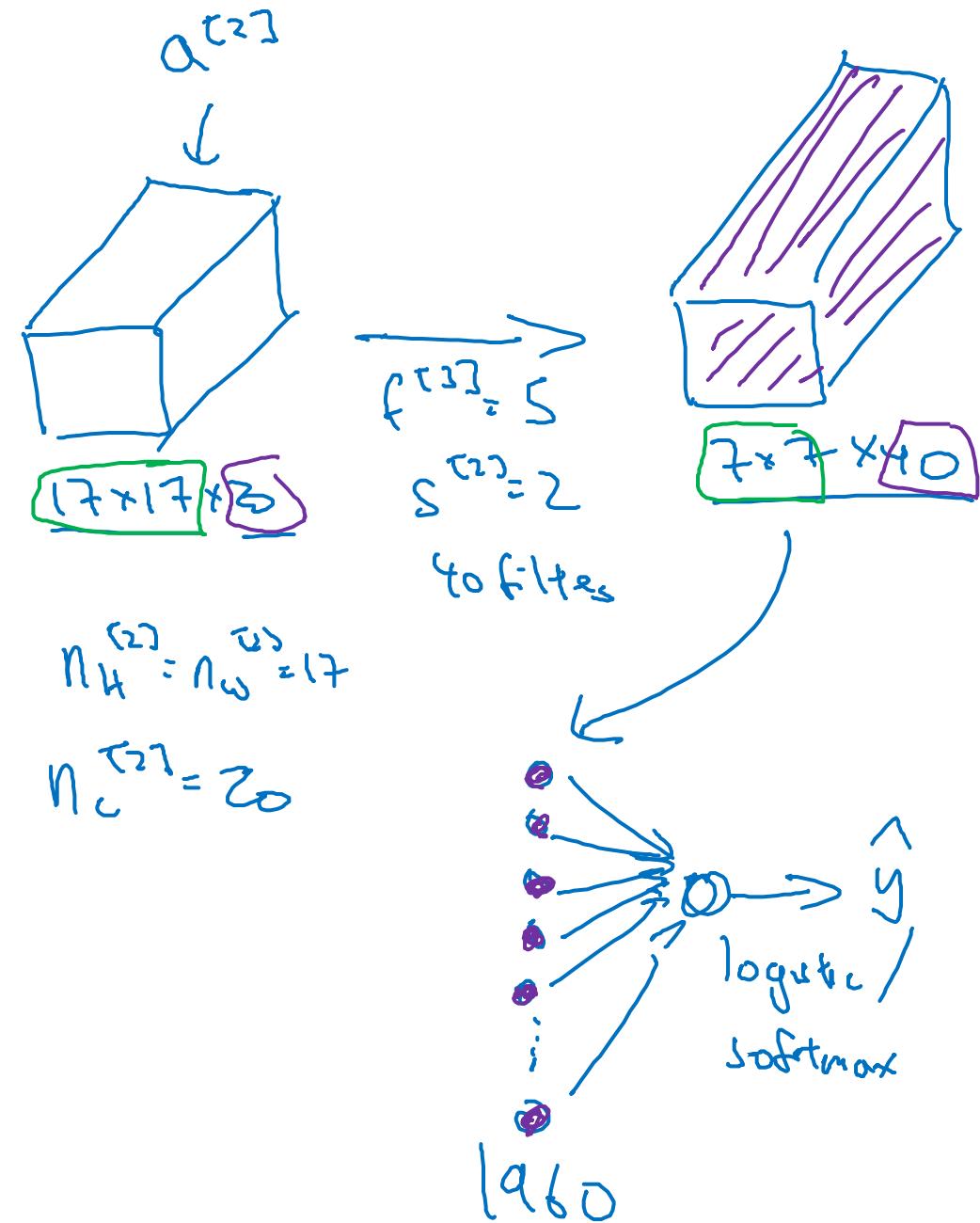
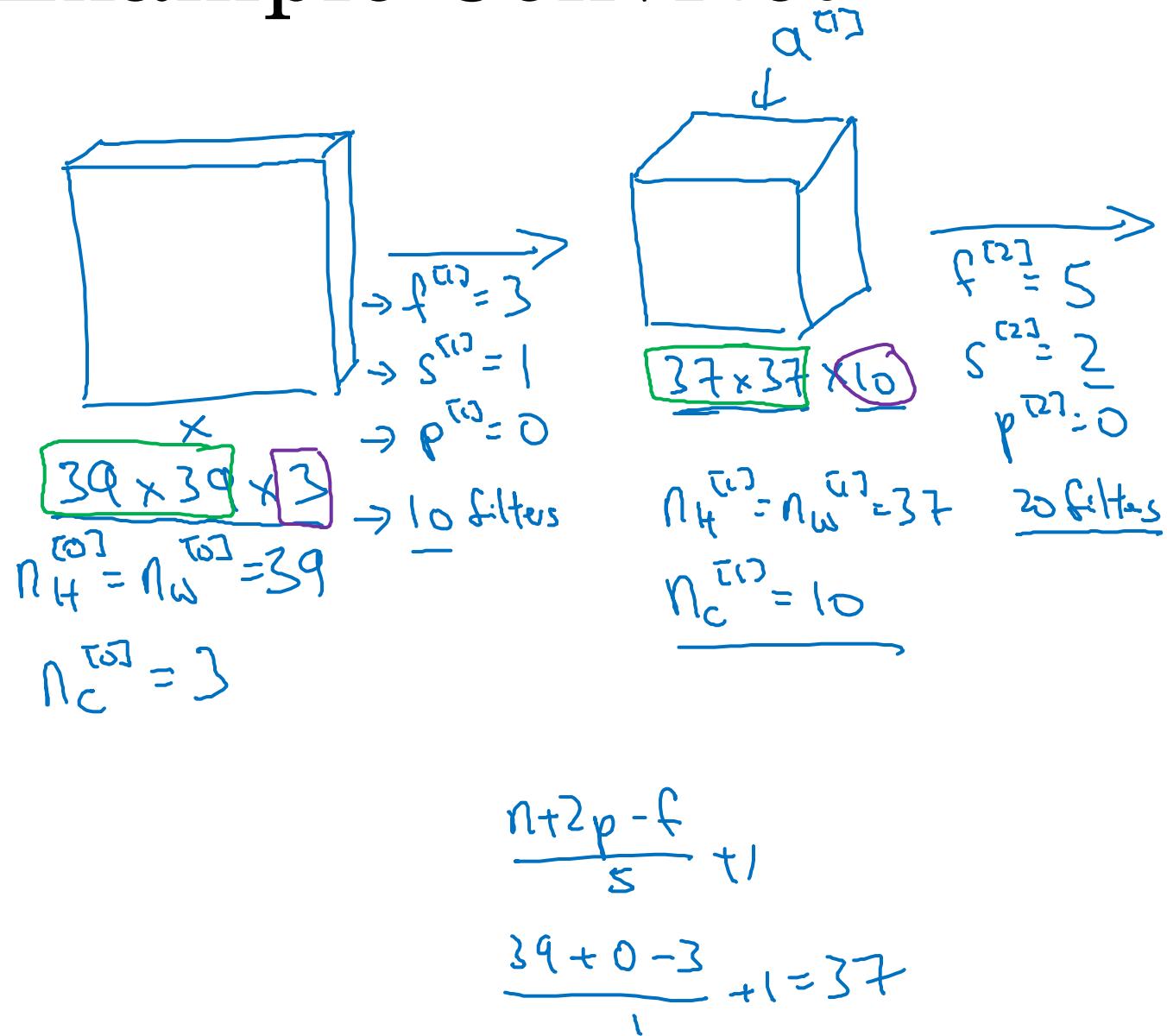
deeplearning.ai

# Convolutional Neural Networks

---

## A simple convolution network example

# Example ConvNet



# Types of layer in a convolutional network:

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



deeplearning.ai

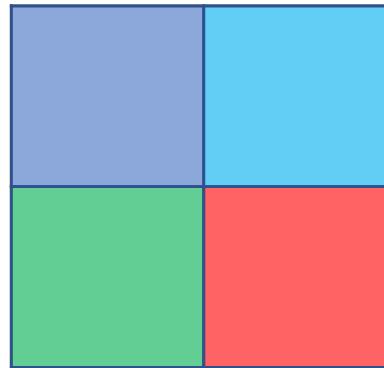
# Convolutional Neural Networks

---

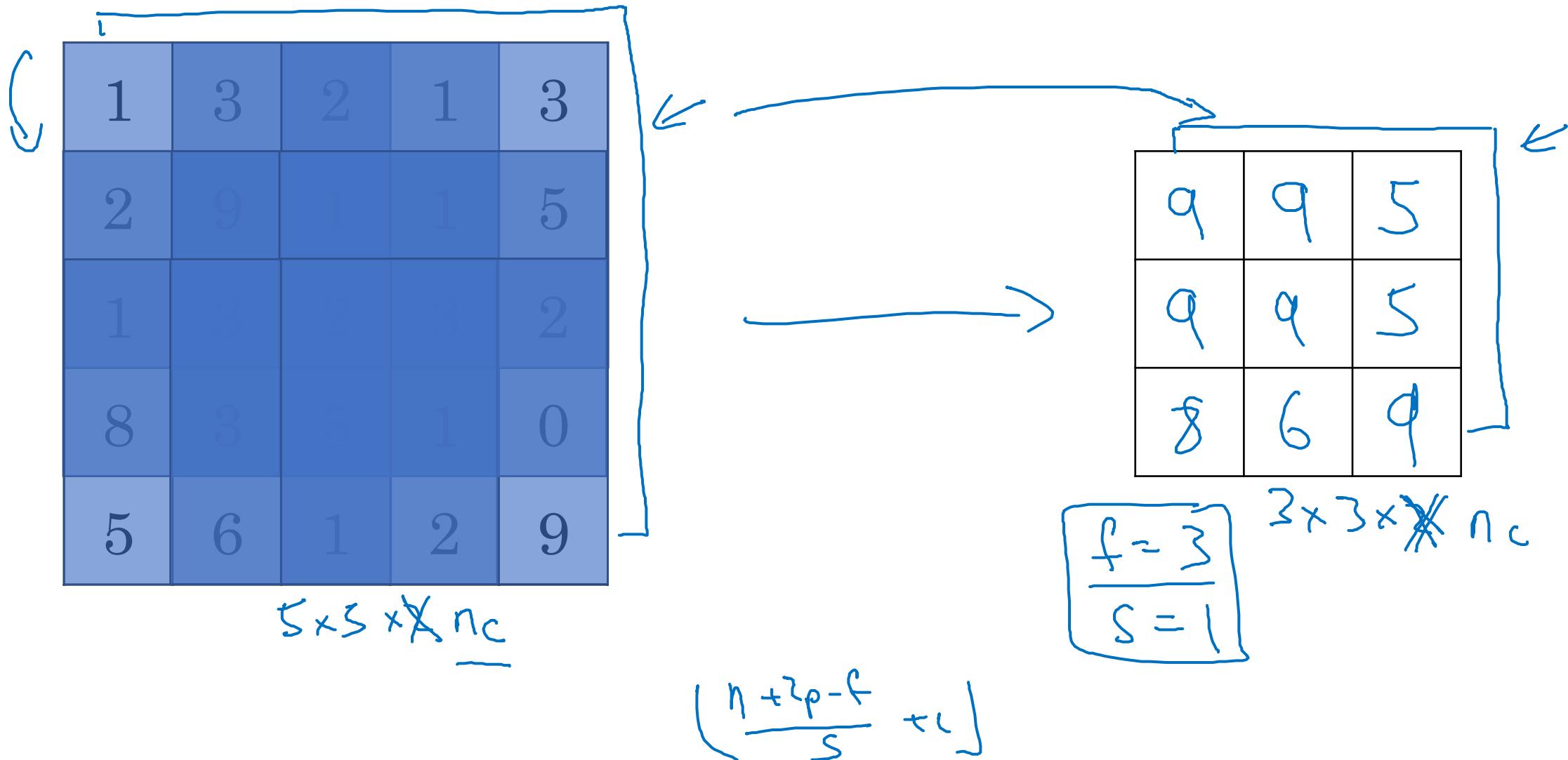
## Pooling layers

# 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

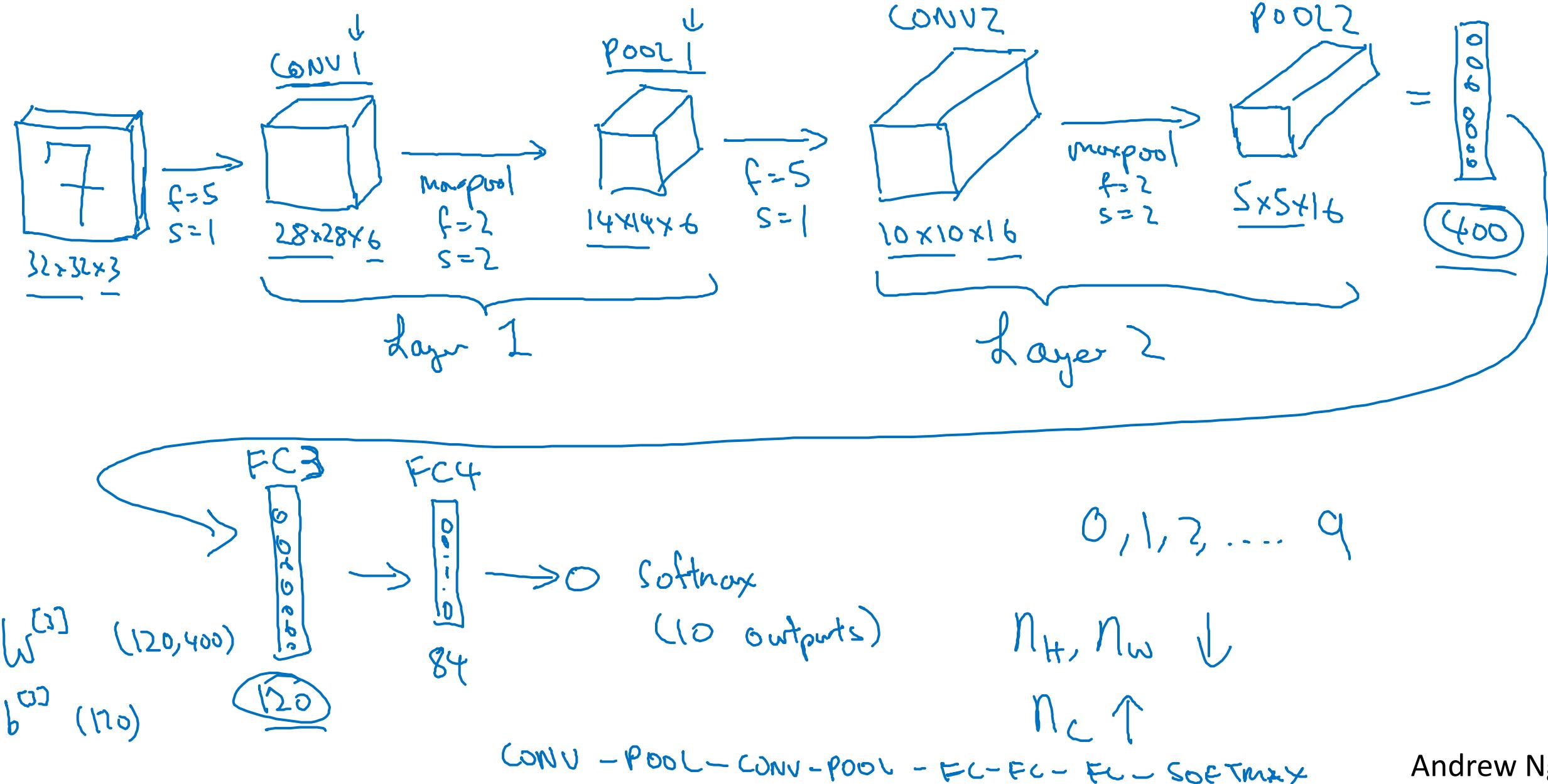
# Convolutional Neural Networks

---

## Convolutional neural network example

# Neural network example

(LeNet-5)



# Neural network example

	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	— 3,072 $a^{[32]}$	0
		—	←
		—	←
		—	←
		—	←
		—	←
		—	—
		—	—
		—	—
		—	—



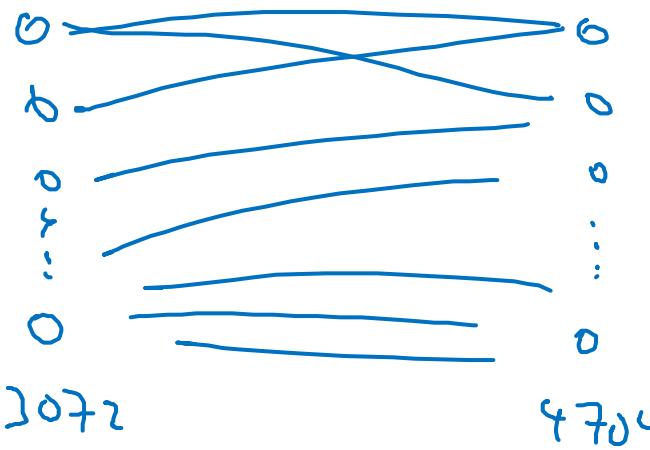
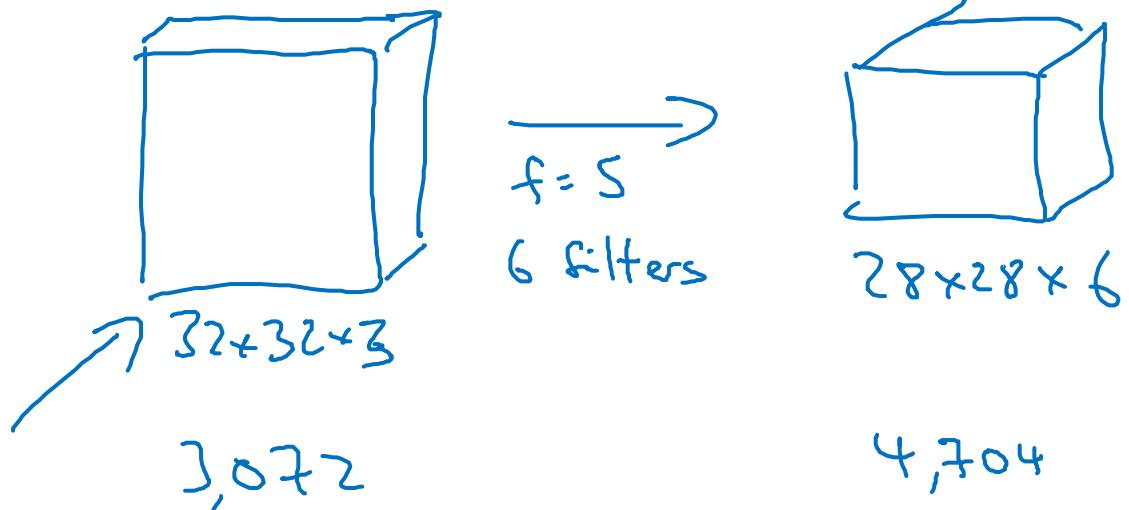
deeplearning.ai

# Convolutional Neural Networks

---

## Why convolutions?

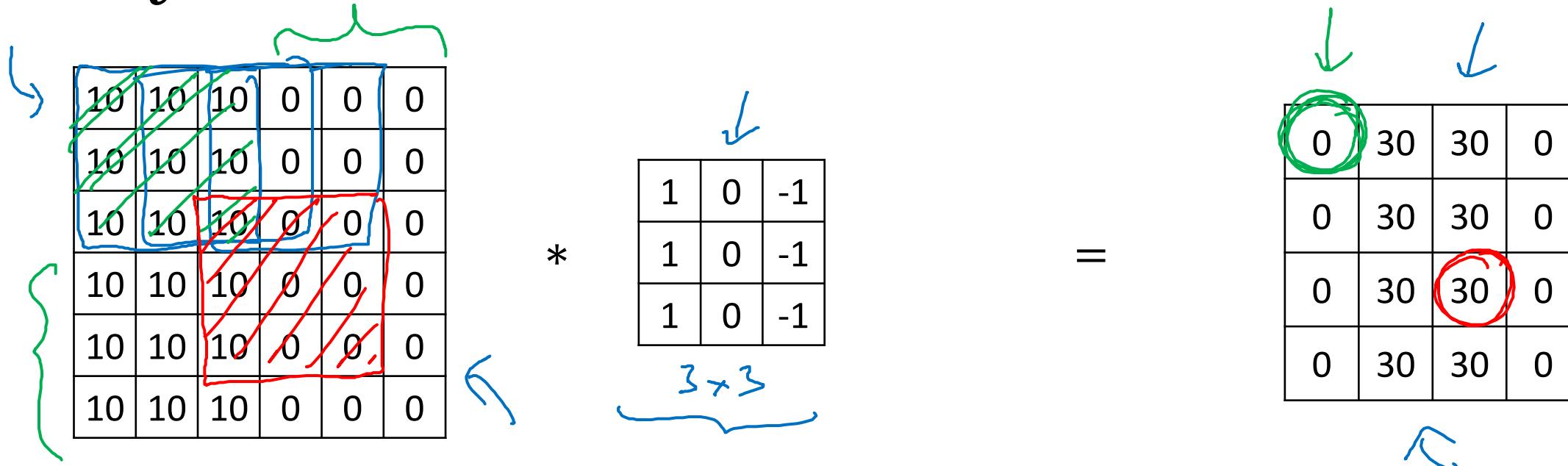
# Why convolutions



$$5 \times 5 - 25$$
$$26$$
$$6 \times 26 = 156 \text{ Parameters}$$

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

# Why convolutions

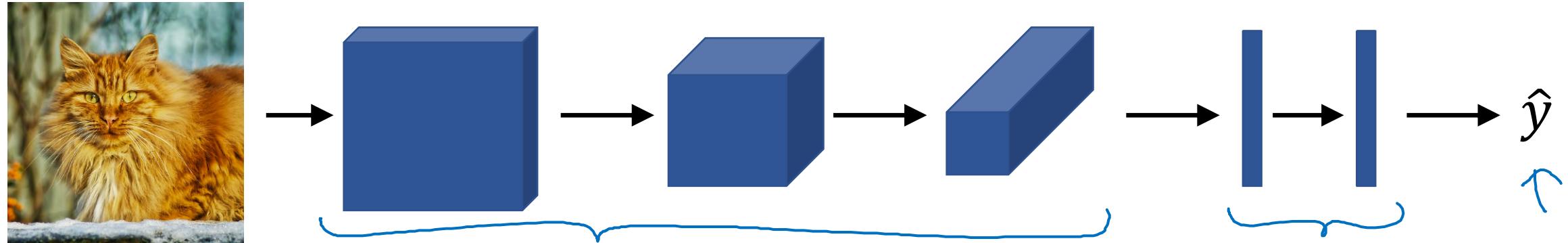


**Parameter sharing:** A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

→ **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  $J$



deeplearning.ai

# Case Studies

---

Why look at  
case studies?

# Outline

Classic networks:

- LeNet-5 ←
- AlexNet ←
- VGG ←

ResNet (152)

Inception



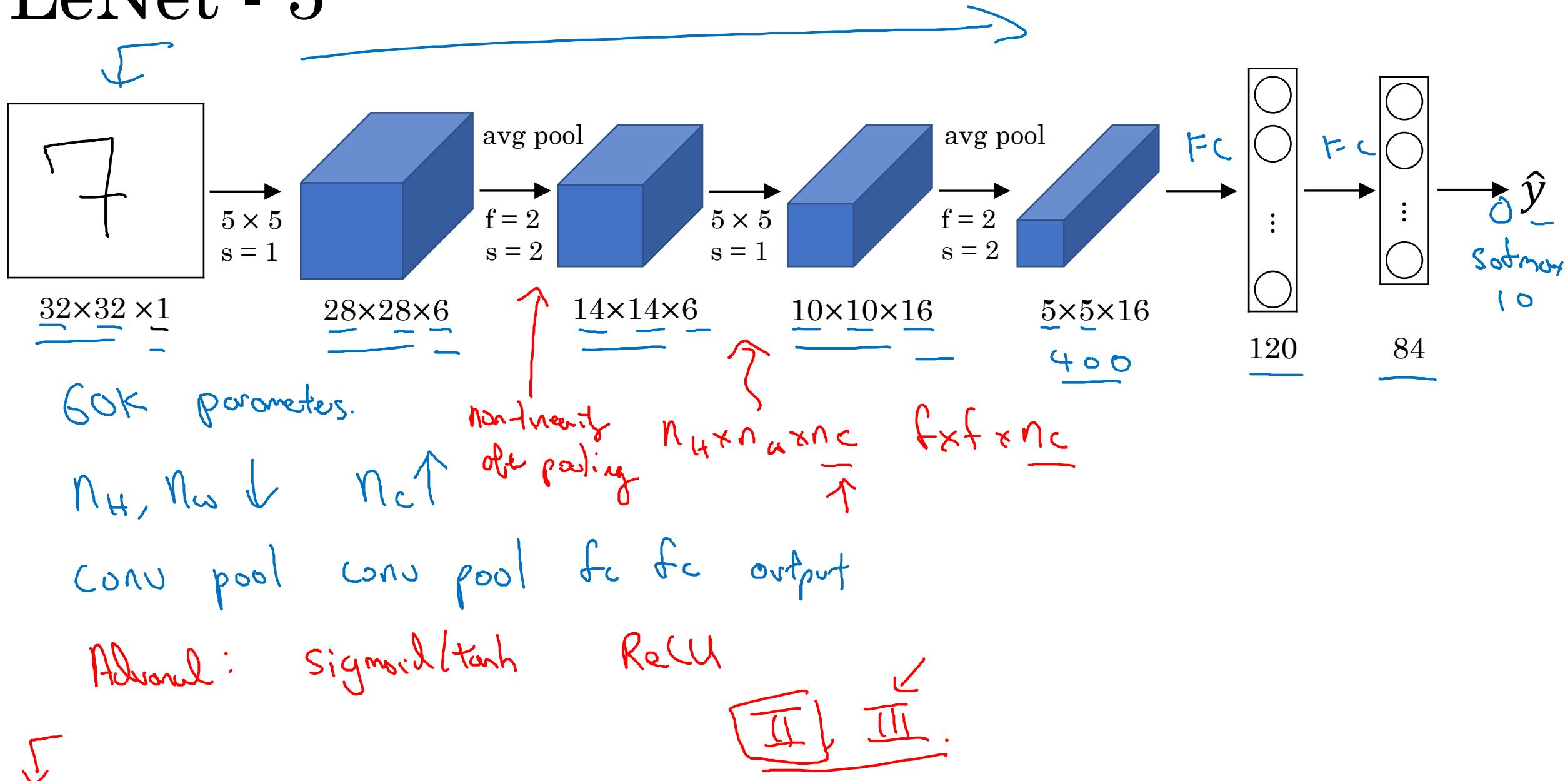
deeplearning.ai

# Case Studies

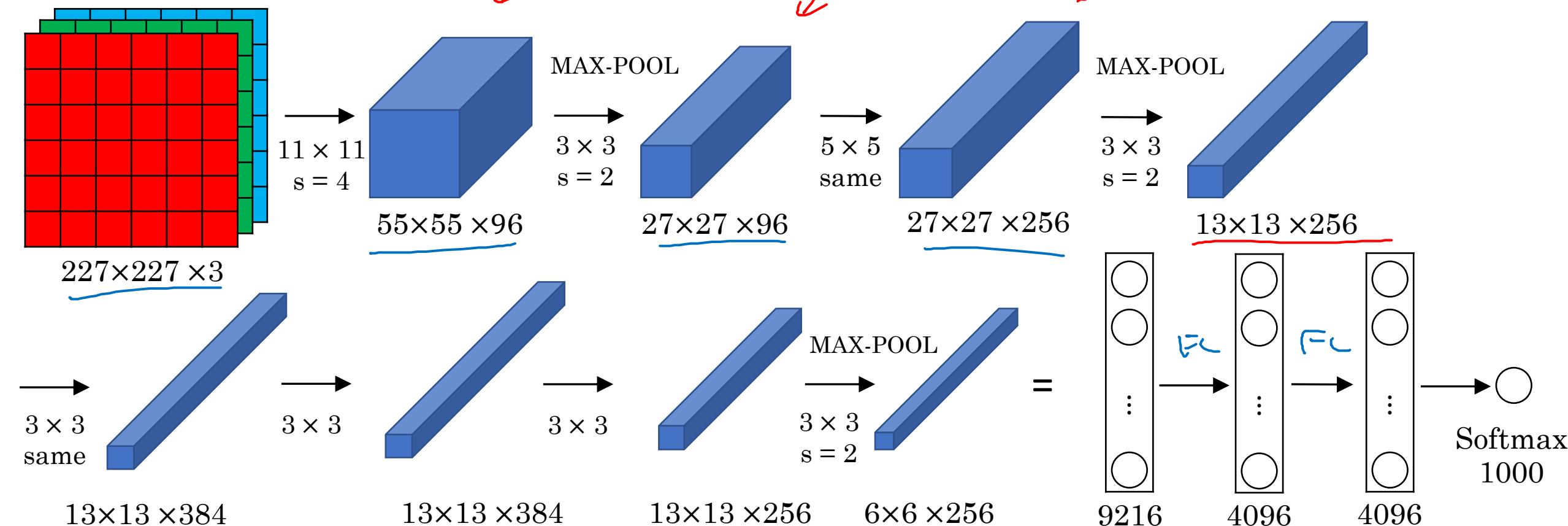
---

## Classic networks

# LeNet - 5

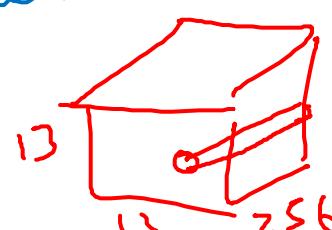


# AlexNet



- Similar to LeNet, but much bigger.
- ReLU

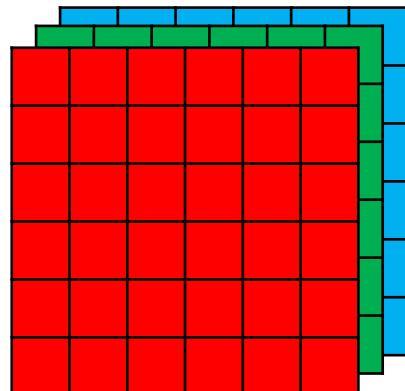
- Multiple GPUs.  
- Local Response Normalization (LRN)



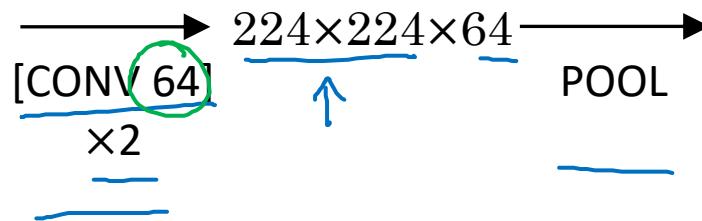
~60M Parameters

# VGG - 16

CONV =  $3 \times 3$  filter,  $s = 1$ , same



$224 \times 224$



POOL

Intermediate layer dimensions:  $112 \times 112 \times 64$ . The feature map is pooled to produce a smaller feature map of size  $112 \times 112 \times 64$ .

MAX-POOL =  $2 \times 2$ ,  $s = 2$

$224 \times 224$

[CONV 128]  $\times 2$

Intermediate layer dimensions:  $112 \times 112 \times 128$ . The input is processed by two convolutional layers with  $128 \times 3 \times 3$  filters and stride 1, resulting in a feature map of size  $112 \times 112 \times 128$ .

POOL

$224 \times 224$

$224 \times 224 \times 64$

$224 \times 224 \times 64$

[CONV 256]  $\times 3$

Intermediate layer dimensions:  $56 \times 56 \times 256$ . The input is processed by three convolutional layers with  $256 \times 3 \times 3$  filters and stride 2, resulting in a feature map of size  $56 \times 56 \times 256$ .

POOL

Intermediate layer dimensions:  $28 \times 28 \times 256$ . The feature map is pooled to produce a smaller feature map of size  $28 \times 28 \times 256$ .

[CONV 512]  $\times 3$

Intermediate layer dimensions:  $28 \times 28 \times 512$ . The input is processed by three convolutional layers with  $512 \times 3 \times 3$  filters and stride 2, resulting in a feature map of size  $28 \times 28 \times 512$ .

POOL

Intermediate layer dimensions:  $14 \times 14 \times 512$ . The feature map is pooled to produce a smaller feature map of size  $14 \times 14 \times 512$ .

[CONV 512]  $\times 3$

Intermediate layer dimensions:  $14 \times 14 \times 512$ . The input is processed by three convolutional layers with  $512 \times 3 \times 3$  filters and stride 2, resulting in a feature map of size  $7 \times 7 \times 512$ .

POOL

Intermediate layer dimensions:  $7 \times 7 \times 512$ . The feature map is pooled to produce a smaller feature map of size  $7 \times 7 \times 512$ .

FC 4096

FC 4096

Softmax 1000

$n_H, n_W \downarrow$

$n_C \uparrow$

$\sim 38M$



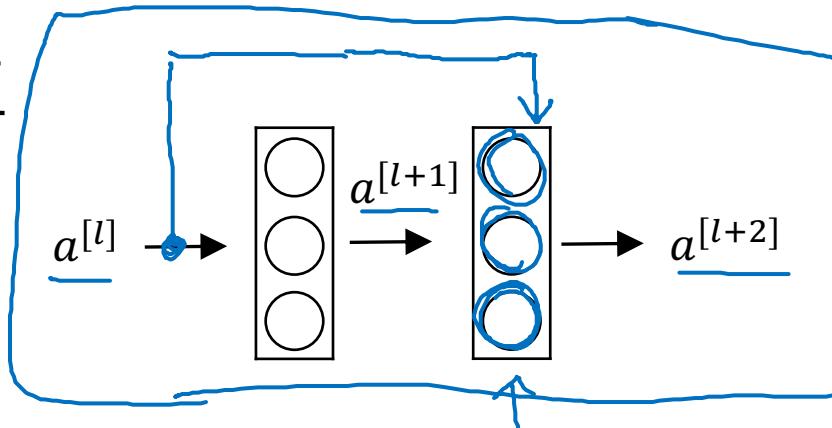
deeplearning.ai

## Case Studies

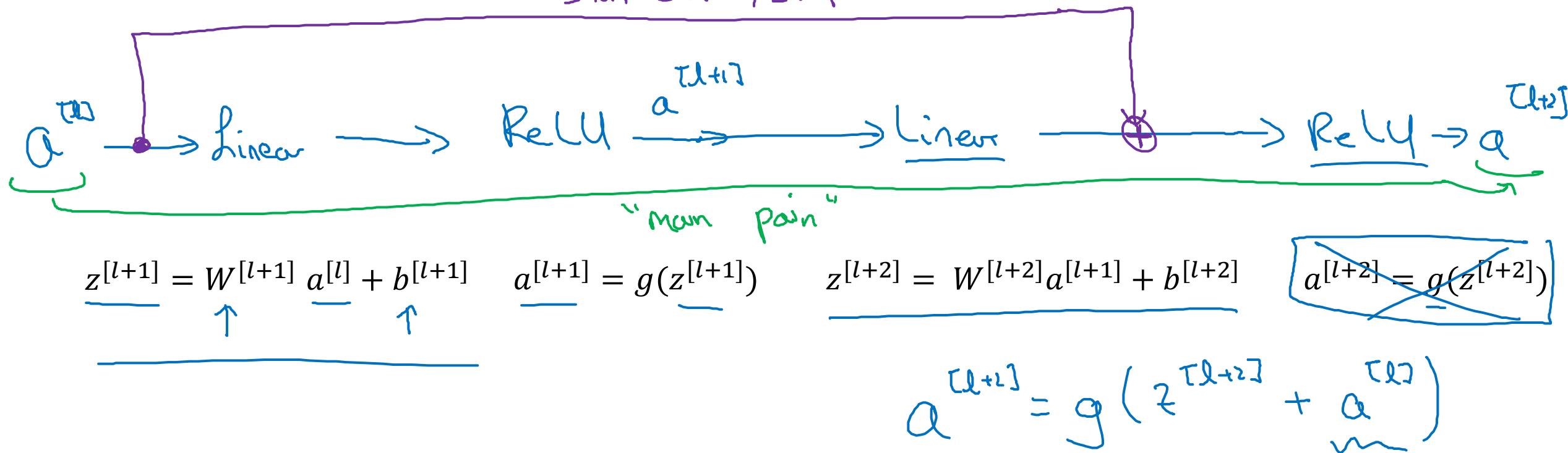
---

# Residual Networks (ResNets)

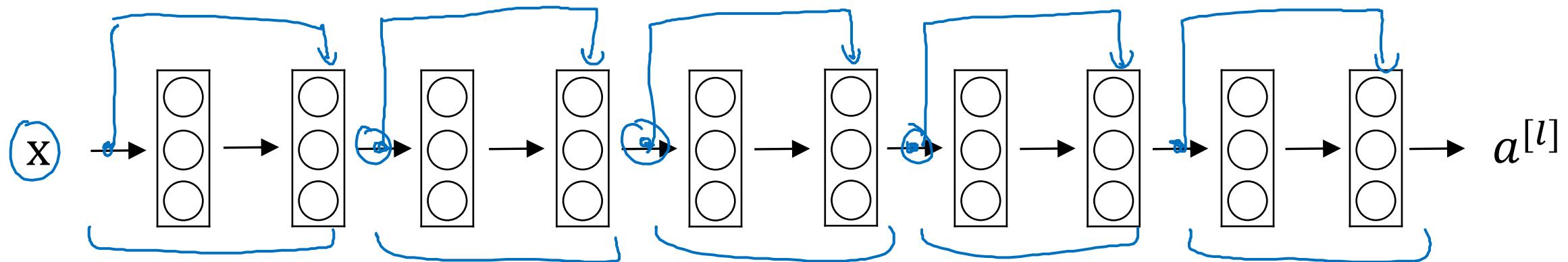
# Residual block



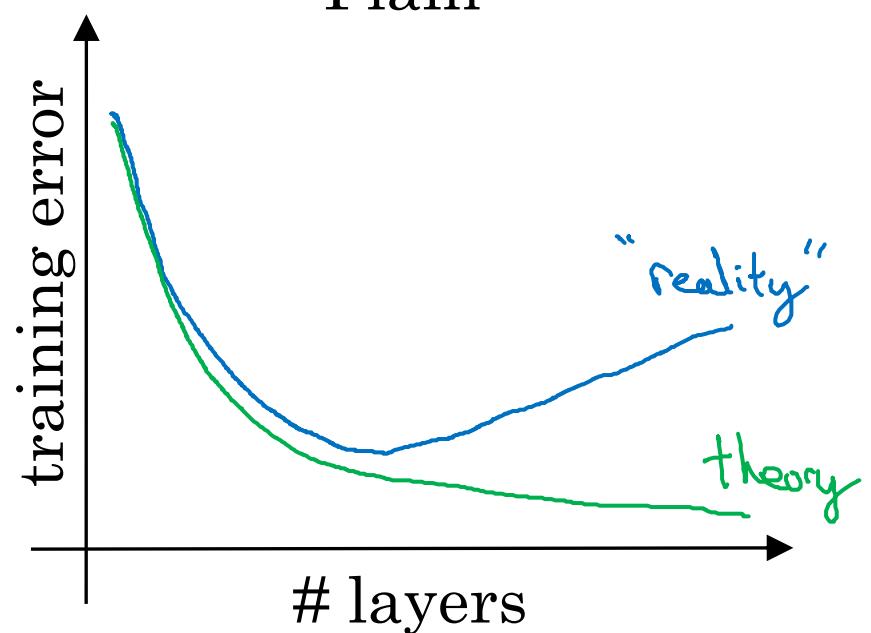
"Short cut" /skip connection



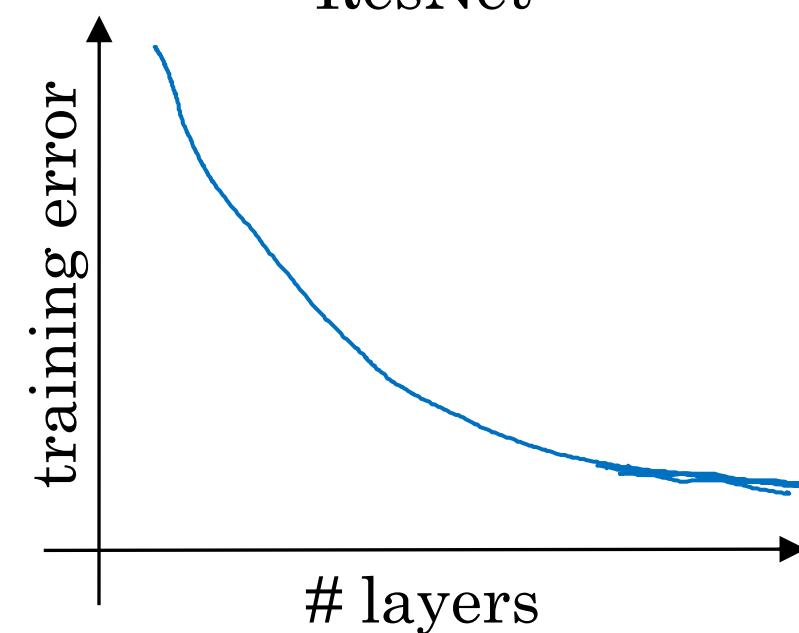
# Residual Network



Plain



ResNet





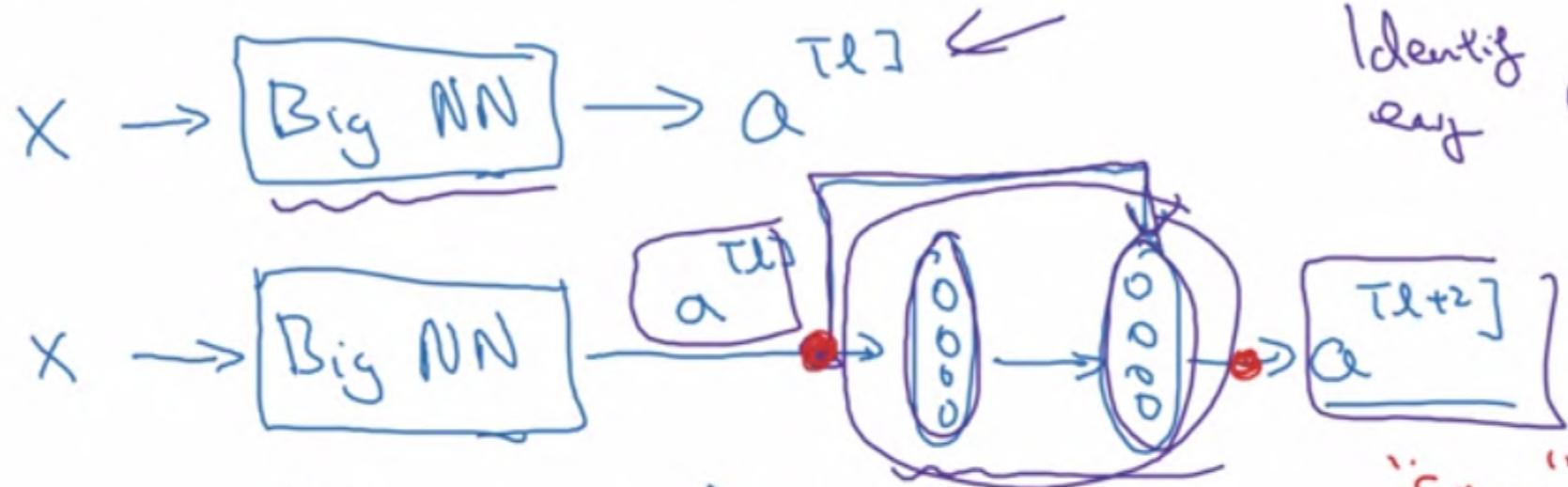
deeplearning.ai

## Case Studies

---

## Why ResNets work

# Why do residual networks work?



ReLU.

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

# ResNet

Plain



ResNet





deeplearning.ai

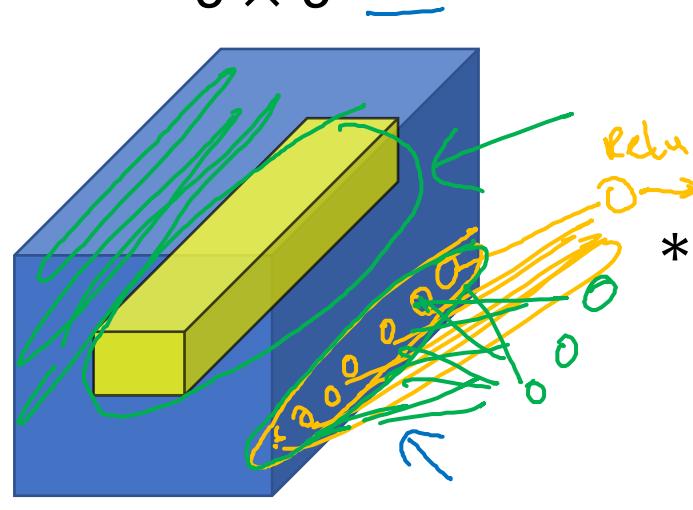
# Case Studies

---

Network in Network and  
 $1 \times 1$  convolutions

# Why does a $1 \times 1$ 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



[Lin et al., 2013. Network in network]

\*

2



=

32 → # filters.

$n_c^{[l+1]}$

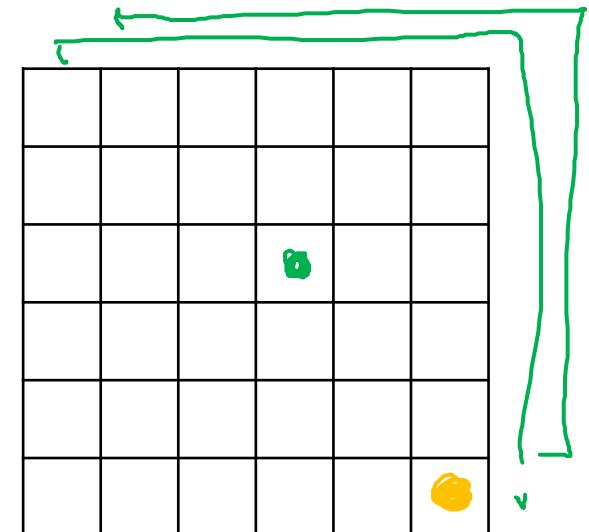
=

ReLU

Network  $\hookrightarrow$  Network

$1 \times 1 \times 32$

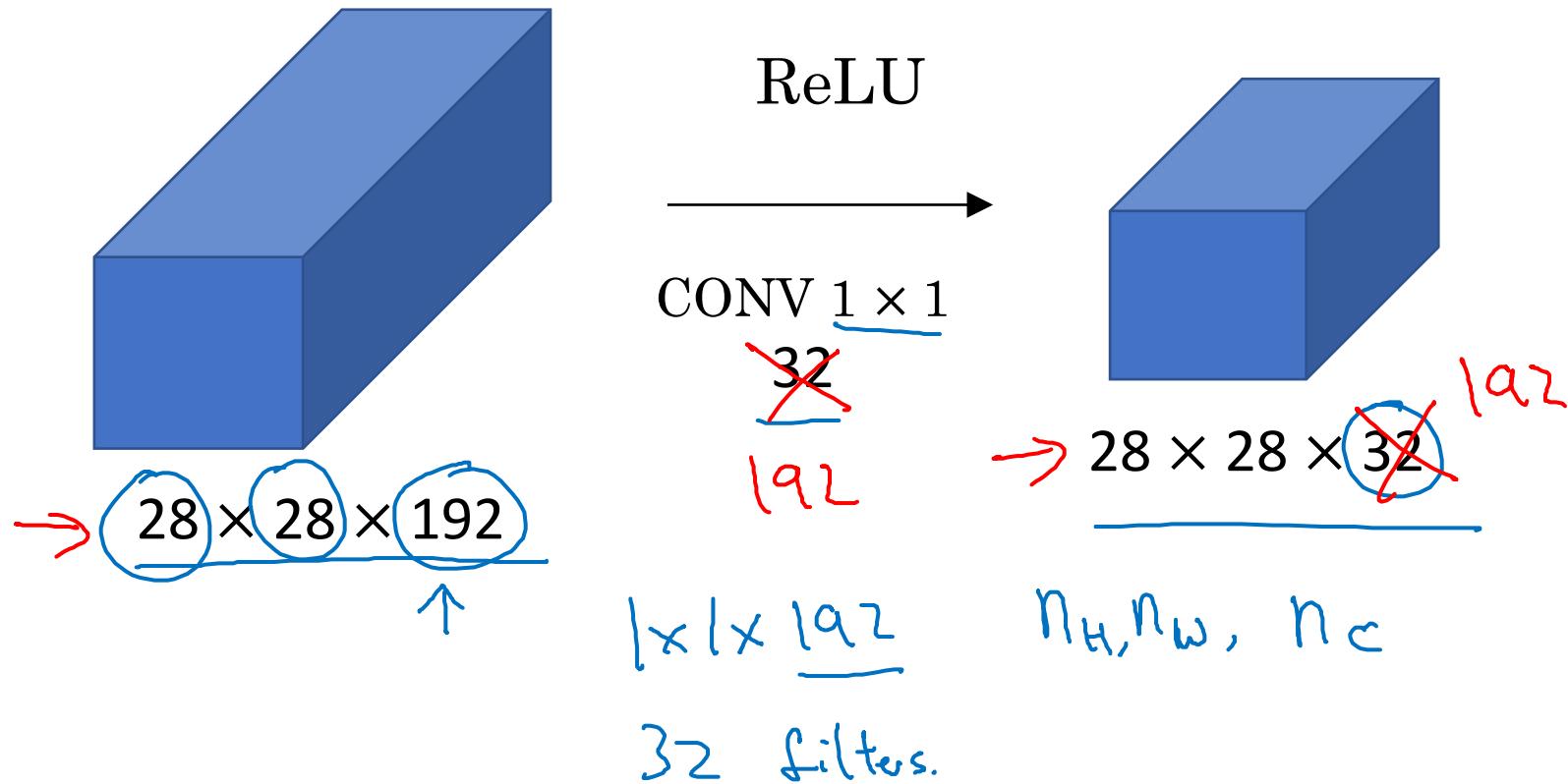
2	4	6	...



$6 \times 6 \times \# \text{ filters}$

Andrew Ng

# Using $1 \times 1$ convolutions





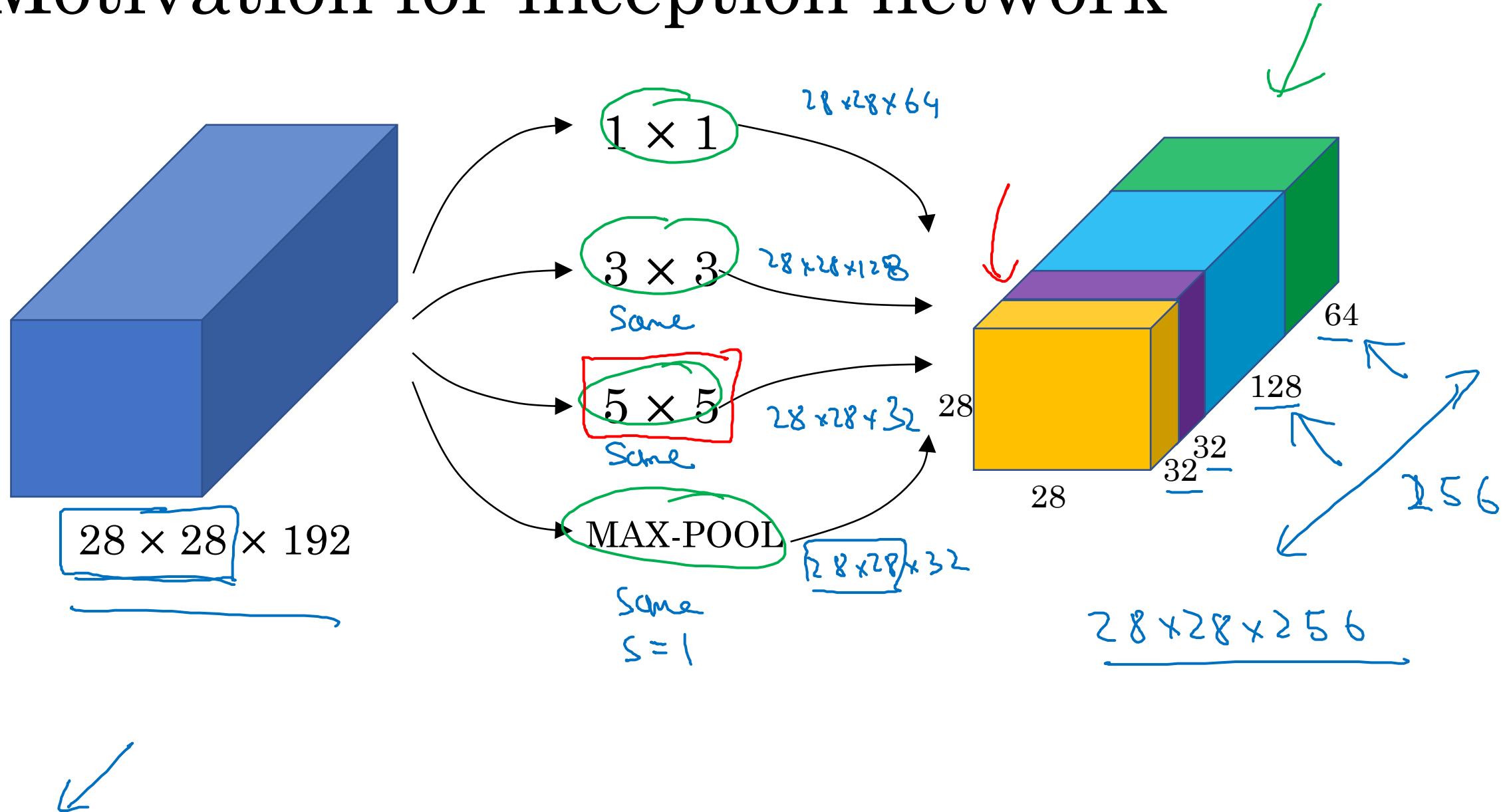
deeplearning.ai

## Case Studies

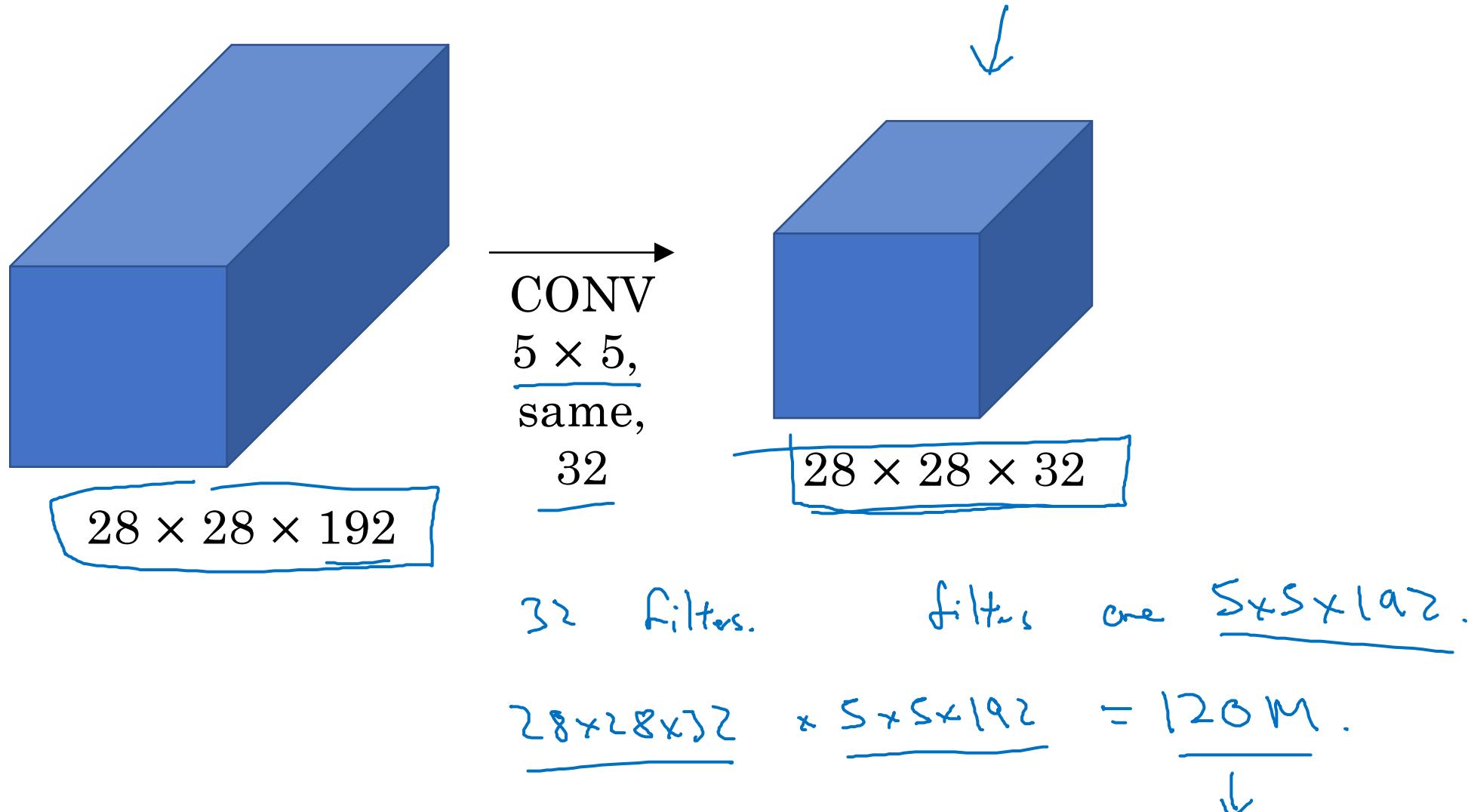
---

Inception network  
motivation

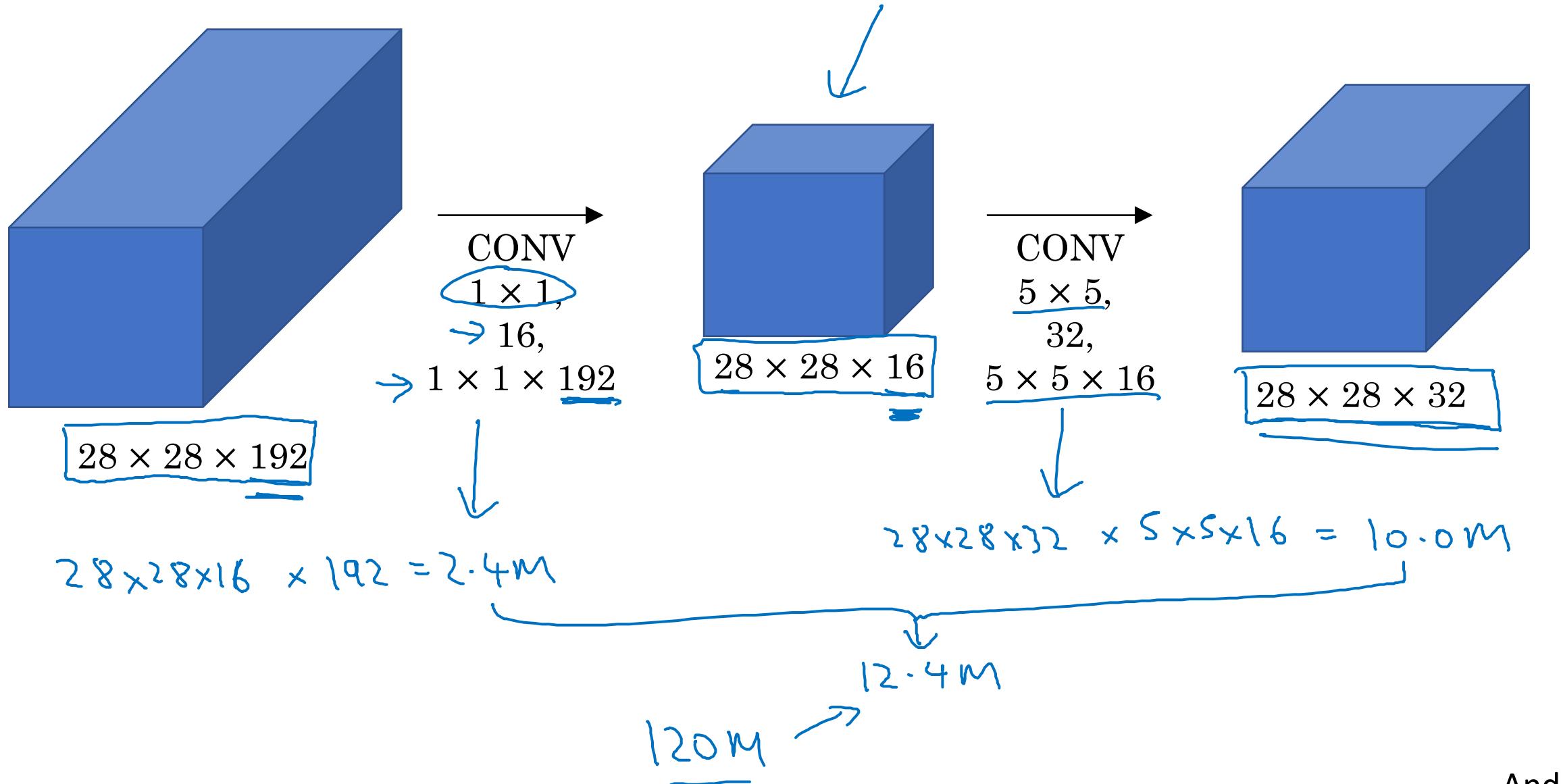
# Motivation for inception network



# The problem of computational cost



# Using $1 \times 1$ convolution





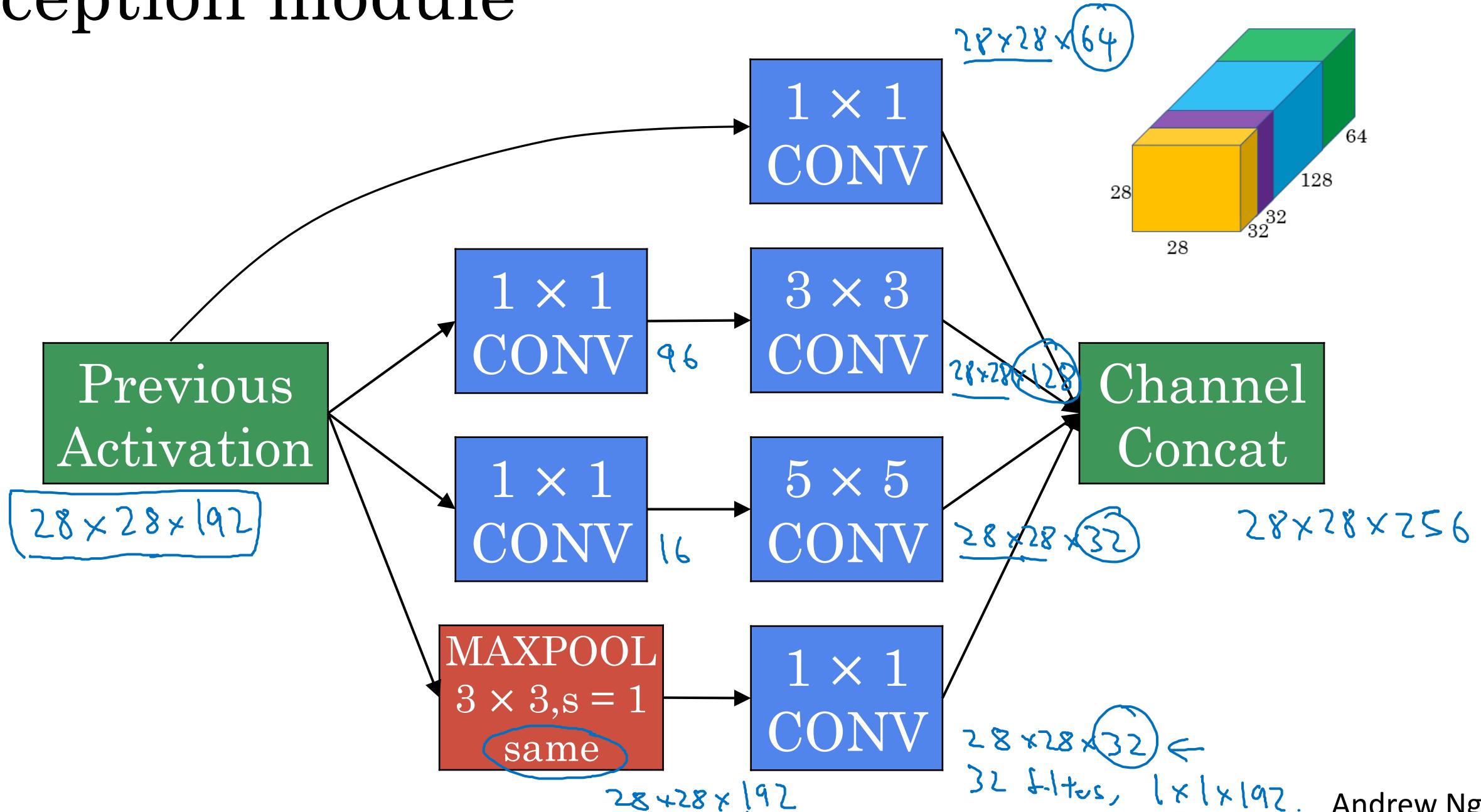
deeplearning.ai

## Case Studies

---

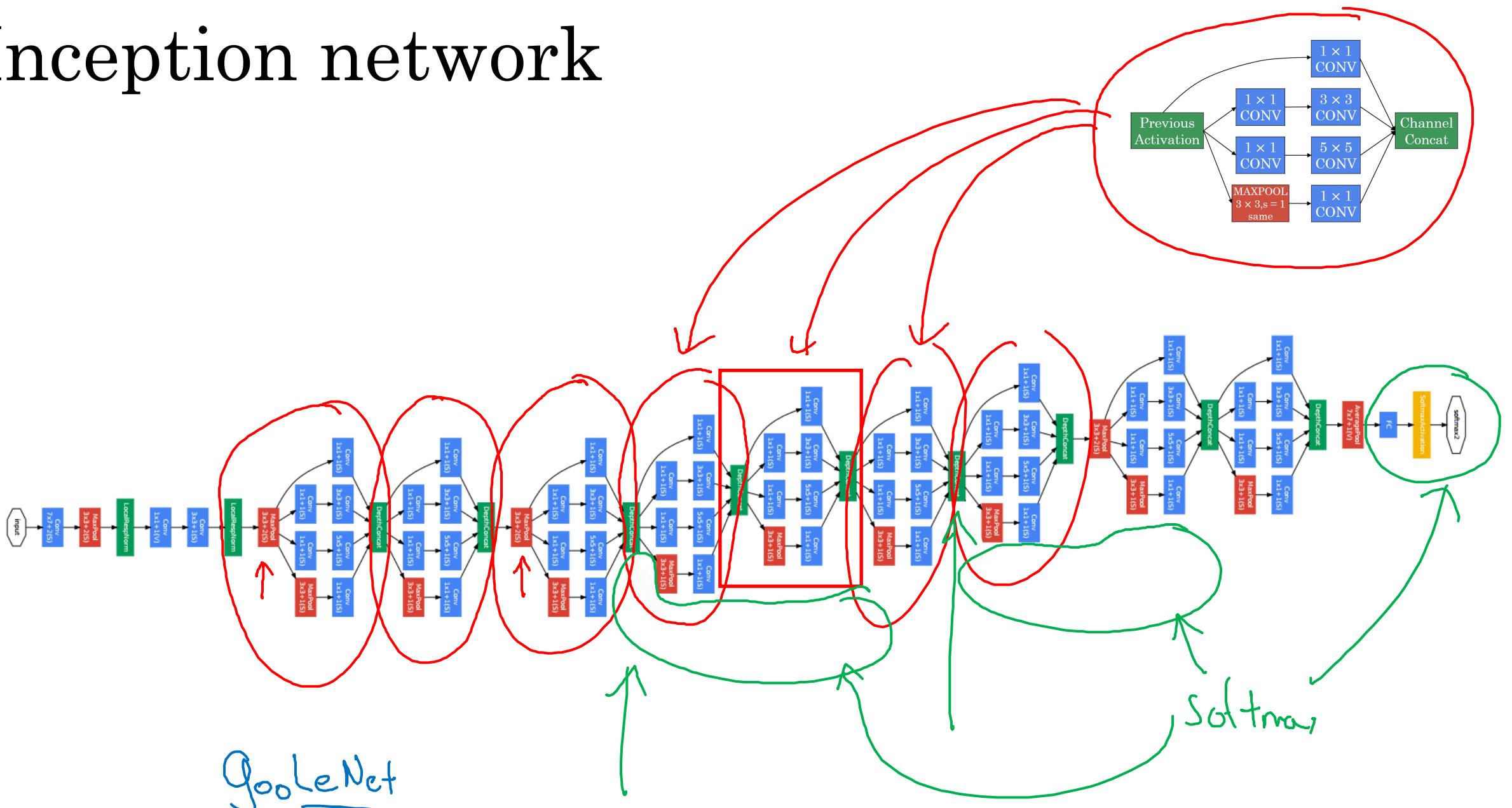
### Inception network

# Inception module



32 filters,  $1 \times 1 \times 192$ . Andrew Ng

# Inception network







deeplearning.ai

Practical advice for  
using ConvNets

---

Using open-source  
implementations

GitHub - KaimingHe/deep-residual-networks

LICENSE Initial commit 2 years ago

README.md Update README.md a year ago

# Deep Residual Networks

By Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun.

Microsoft Research Asia (MSRA).

## Table of Contents

- 0. Introduction
- 1. Citation
- 2. Disclaimer and known issues
- 3. Models
- 4. Results
- 5. Third-party re-implementations

## Introduction

This repository contains the original models (ResNet-50, ResNet-101, and ResNet-152) described in the paper "Deep Residual Learning for Image Recognition" (<http://arxiv.org/abs/1512.03385>). These models are those used in [ILSVRC] (<http://image-net.org/challenges/LSVRC/2015/>) and COCO 2015 competitions, which won the 1st places in: ImageNet classification, ImageNet detection, ImageNet localization, COCO detection, and COCO segmentation.

## Note

```
C:\Users\Andrew Ng>git clone https://github.com/KaimingHe/deep-residual-networks.git
```

```
Cloning into 'deep-residual-networks'...
```

```
remote: Counting objects: 110, done.
```

```
remote: Total 110 (delta 0), reused 0 (delta 0), pack-reused 110R
```

```
Receiving objects: 100% (110/110), 27.37 KiB | 1.19 MiB/s, done.
```

```
Resolving deltas: 100% (55/55), done.
```

```
C:\Users\Andrew Ng>cd
```

C:\Users\Andrew Ng>cd deep-residual-networks

C:\Users\Andrew Ng\deep-residual-networks>dir

Volume in drive C is Windows

Volume Serial Number is 4E6C-6C59

Directory of C:\Users\Andrew Ng\deep-residual-networks

<

10/15/2017	06:30 PM	<DIR>	.
10/15/2017	06:30 PM	<DIR>	..
10/15/2017	06:30 PM		96 .gitmodules
10/15/2017	06:30 PM	<DIR>	caffe
10/15/2017	06:30 PM		1,100 LICENSE
10/15/2017	06:30 PM	<DIR>	prototxt
10/15/2017	06:30 PM		7,160 README.md
		3 File(s)	8,356 bytes
		4 Dir(s)	515,663,204,352 bytes free

C:\Users\Andrew Ng\deep-residual-networks>

```
C:\Users\Andrew Ng\deep-residual-networks>cd prototxt
```

```
C:\Users\Andrew Ng\deep-residual-networks\prototxt>dir
```

```
Volume in drive C is Windows
```

```
Volume Serial Number is 4E6C-6C59
```

```
Directory of C:\Users\Andrew Ng\deep-residual-networks\prototxt
```

10/15/2017 06:30 PM	<DIR>	.
10/15/2017 06:30 PM	<DIR>	..
10/15/2017 06:30 PM		69,987 ResNet-101-deploy.prototxt
10/15/2017 06:30 PM		104,809 ResNet-152-deploy.prototxt
10/15/2017 06:30 PM		34,820 ResNet-50-deploy.prototxt
	3 File(s)	209,616 bytes
	2 Dir(s)	515,663,151,104 bytes free

```
C:\Users\Andrew Ng\deep-residual-networks\prototxt>more ResNet-101-deploy.prototxt
```



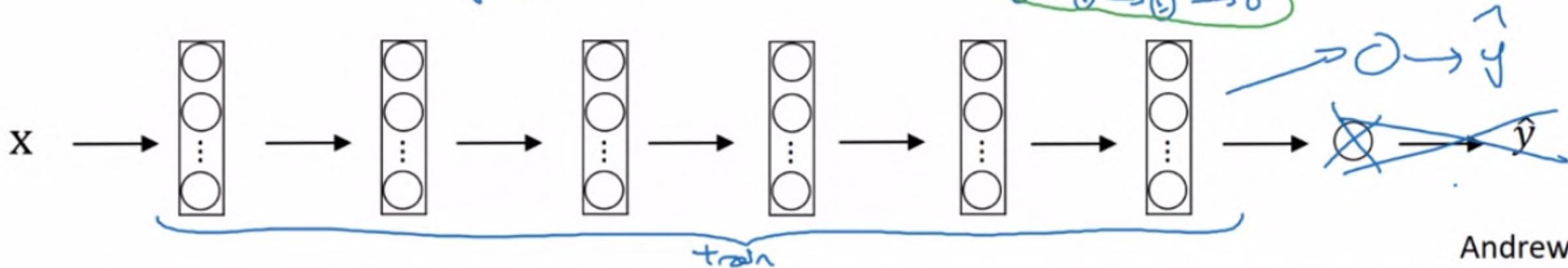
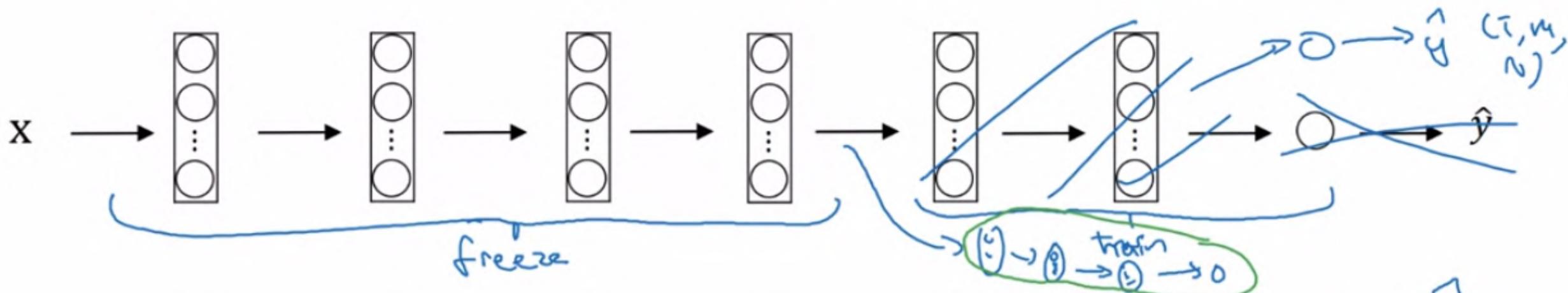
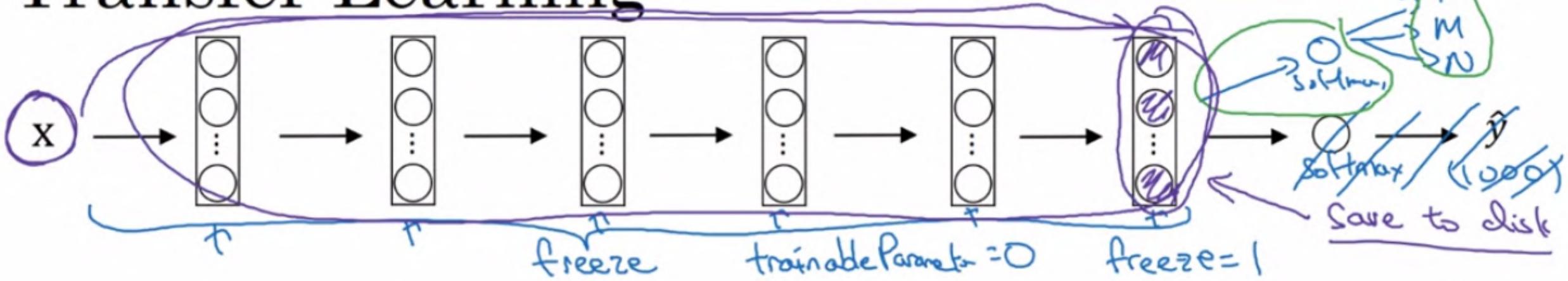
deeplearning.ai

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



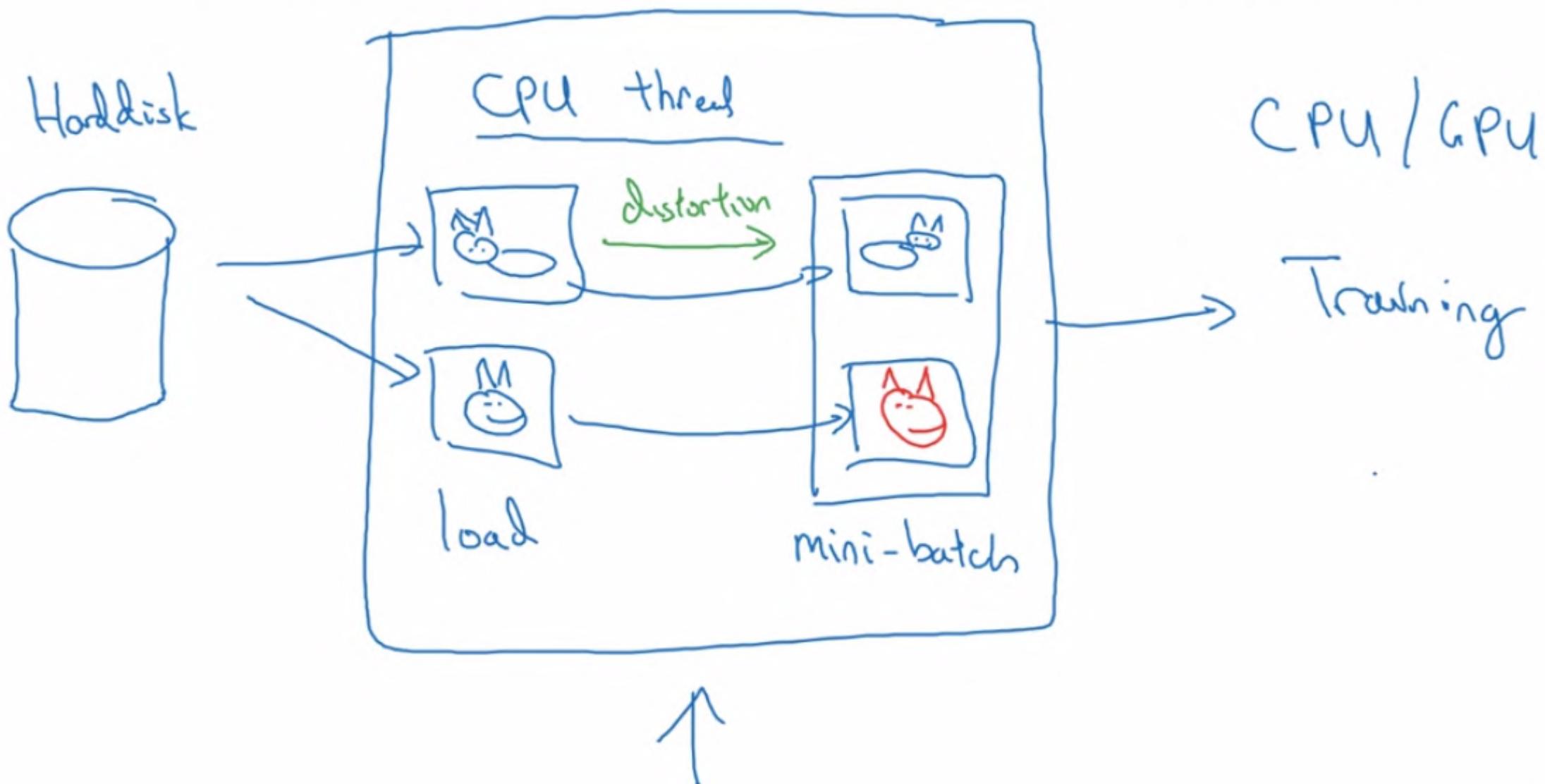
y

Advanced:  
PCA  
[ml-class.org](http://ml-class.org)  
AlexNet paper  
"PCA color augmentation."

R B

G

# Implementing distortions during training





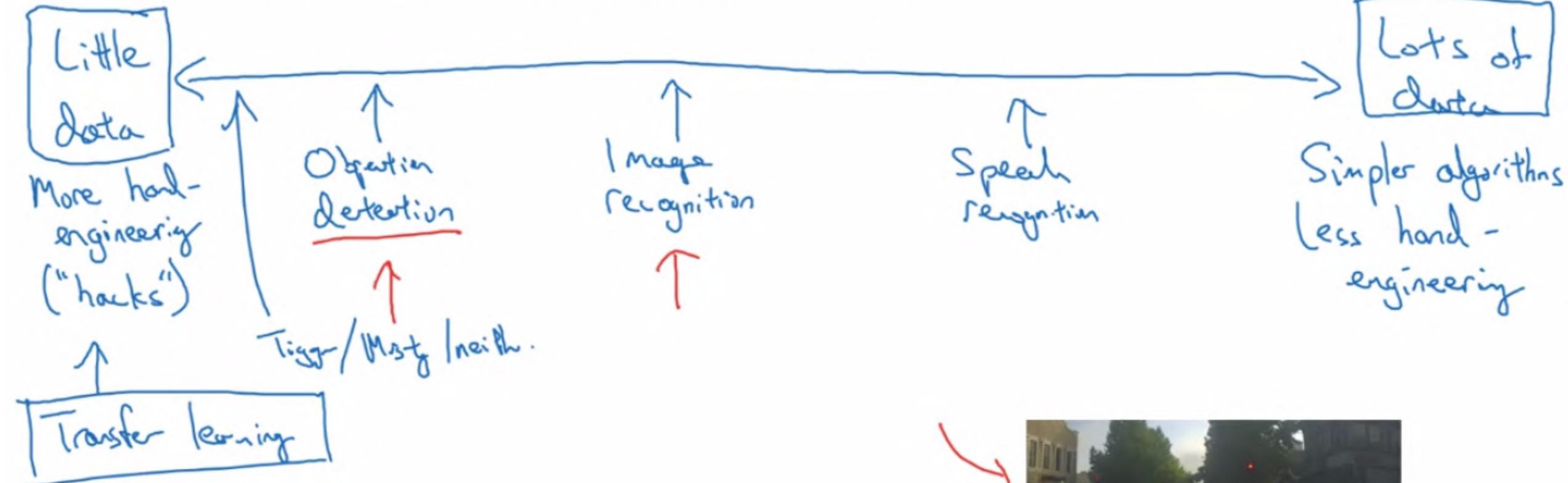
deeplearning.ai

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

$\rightarrow \hat{y}$

- 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

# 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



deeplearning.ai

# Object Detection

---

## Object localization

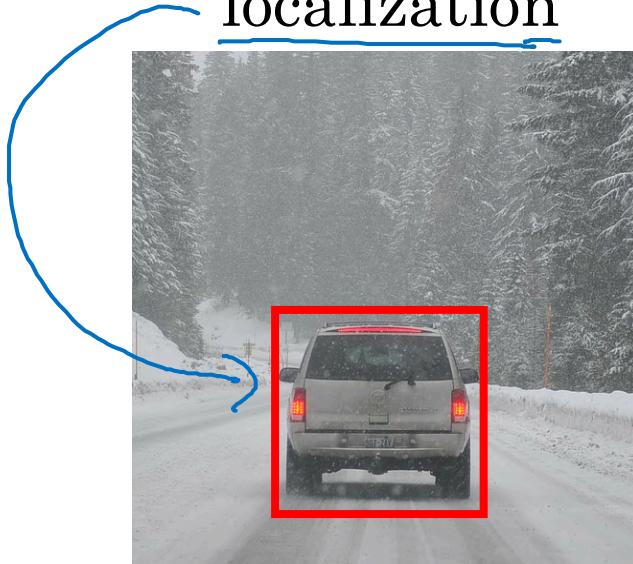
# What are localization and detection?

Image classification



"Car"

Classification with  
localization



"Car"

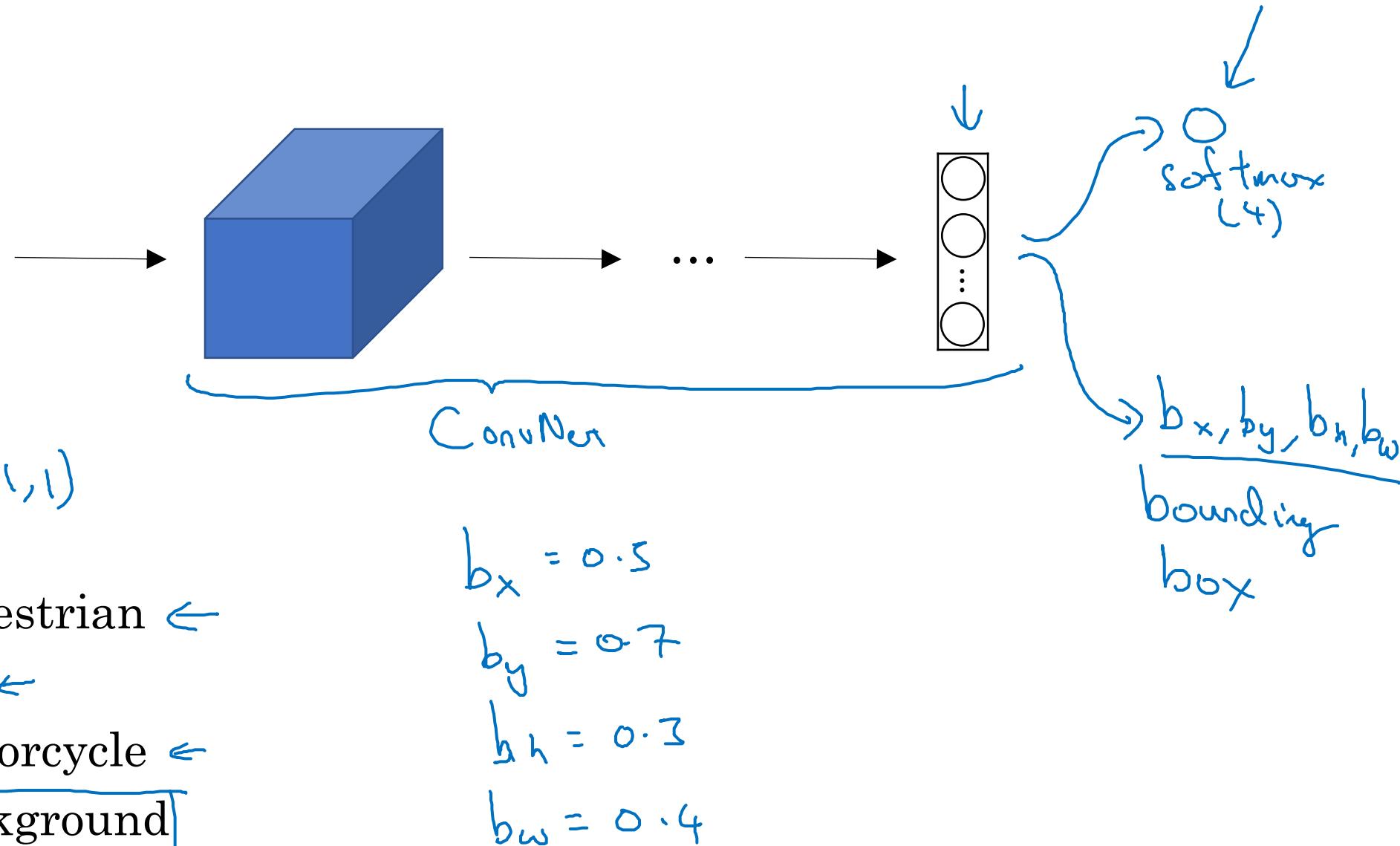
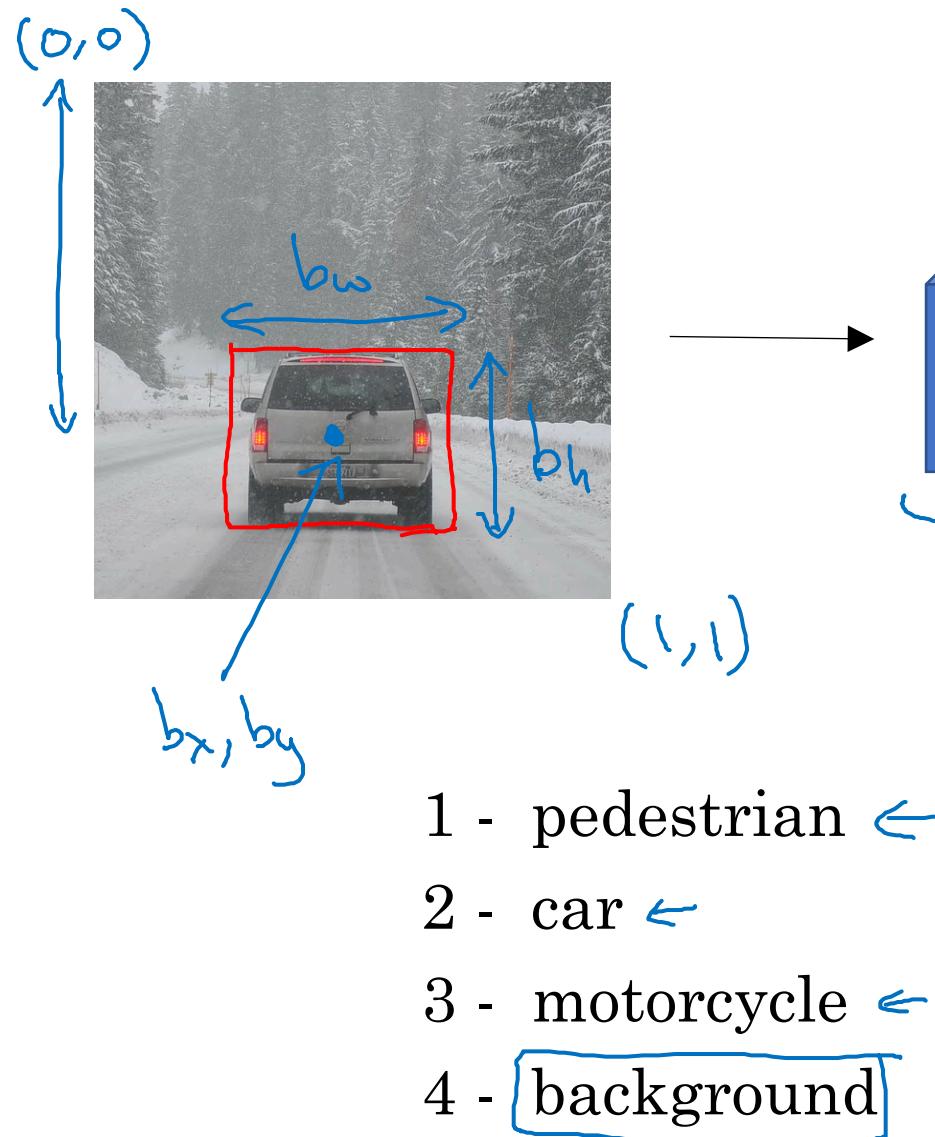
1 object

Detection



multiple  
objects

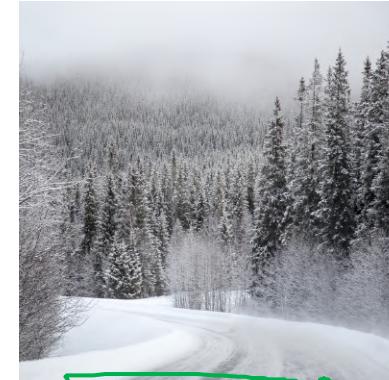
# Classification with localization



# 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)

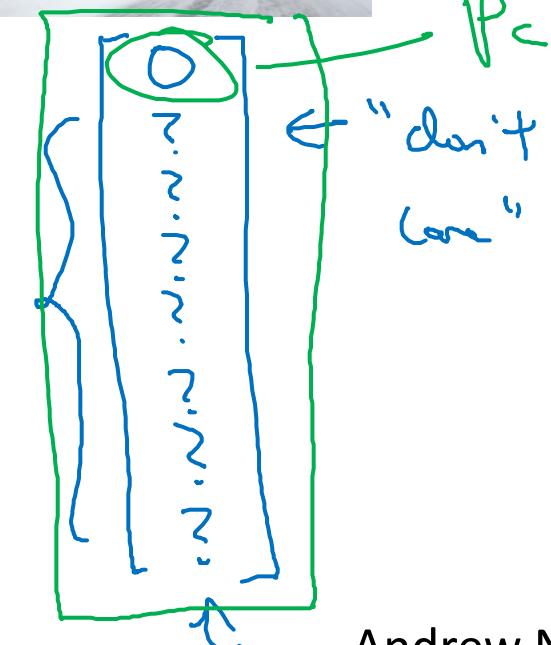
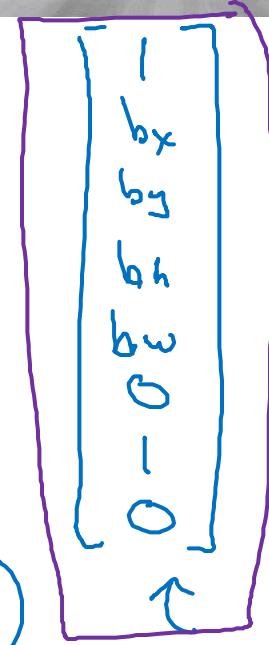


$x =$

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

$$y = \begin{bmatrix} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{bmatrix} \quad \rightarrow \quad \left\{ \begin{array}{l} P_c \\ b_x \\ b_y \\ b_h \\ b_w \\ C_1 \\ C_2 \\ C_3 \end{array} \right\} \text{ is there any object?}$$

$(x, y)$



Andrew Ng



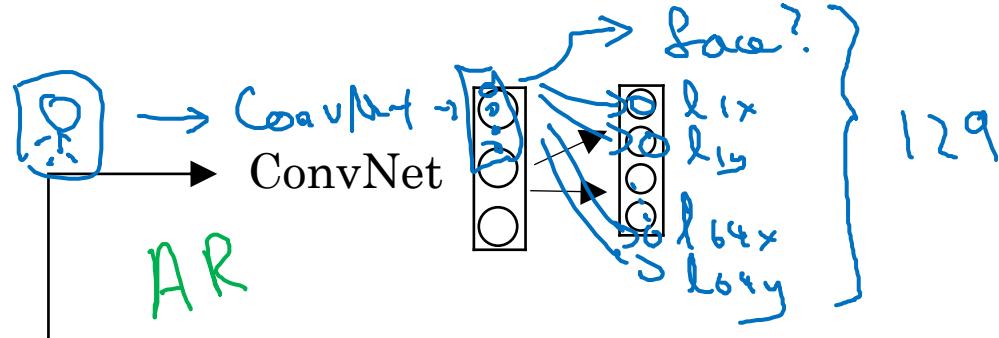
deeplearning.ai

# Object Detection

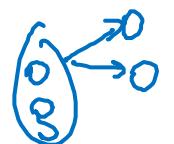
---

## 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

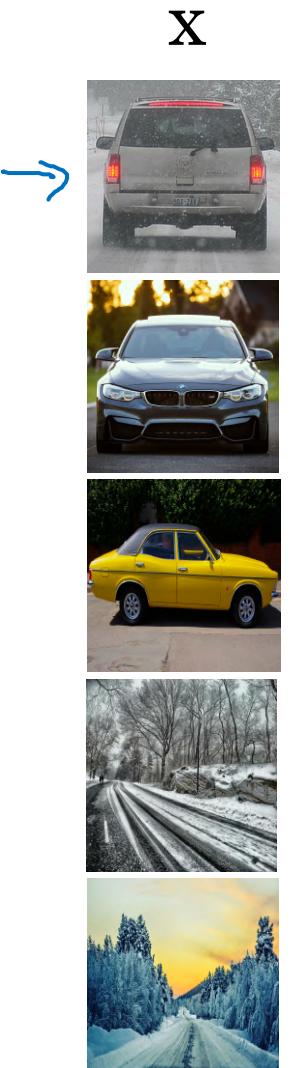
# Object Detection

---

## Object detection

# Car detection example

Training set:



y

1

1

1

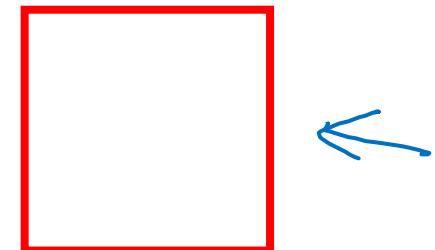
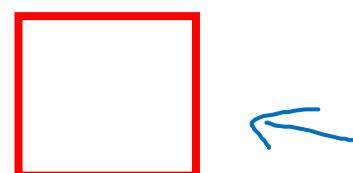
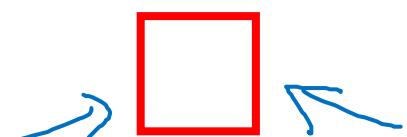
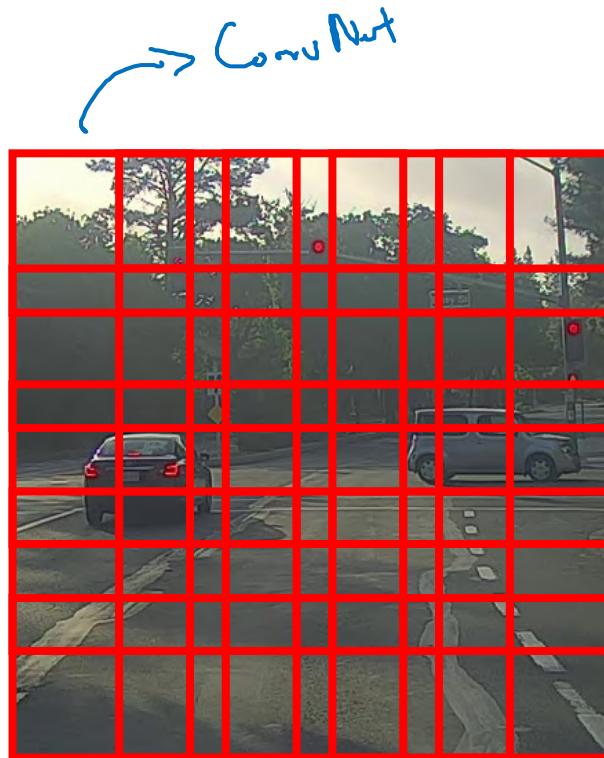
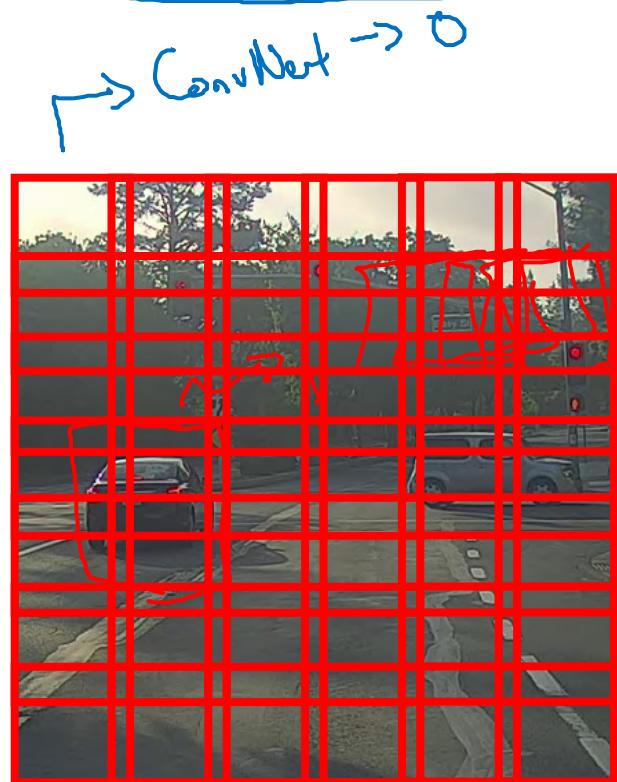
0

0



→ ConvNet → y

# Sliding windows detection



Computation cost



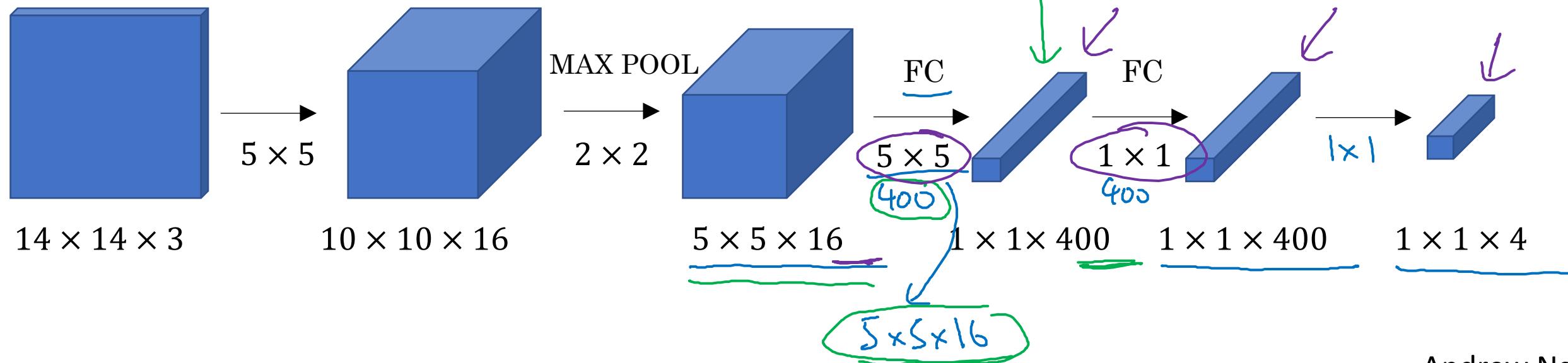
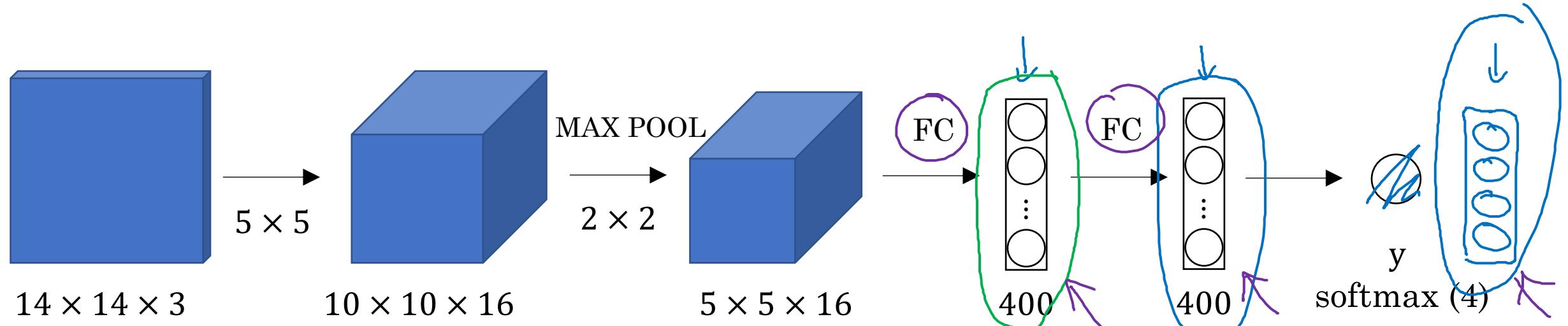
deeplearning.ai

# Object Detection

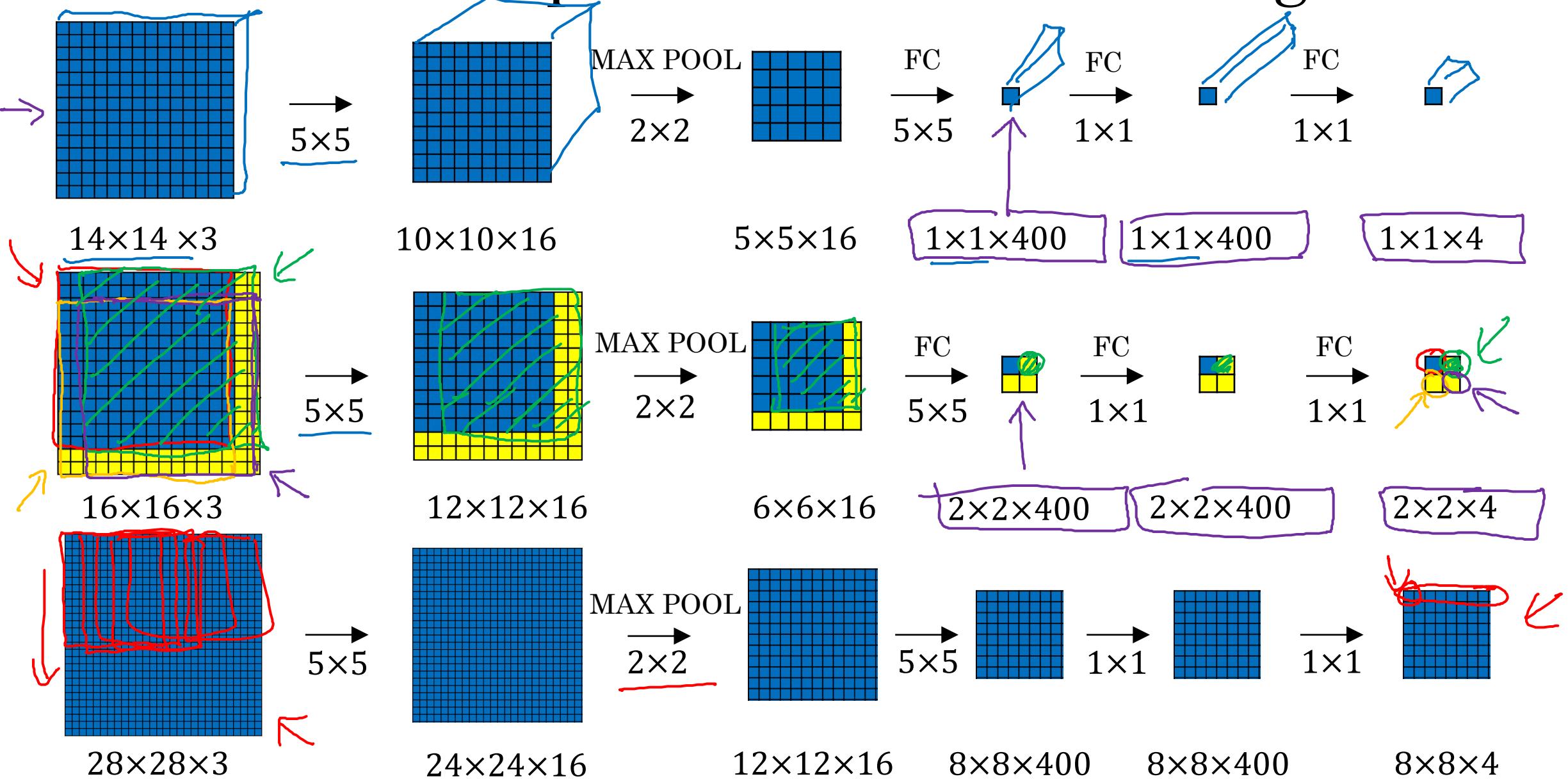
---

## Convolutional implementation of sliding windows

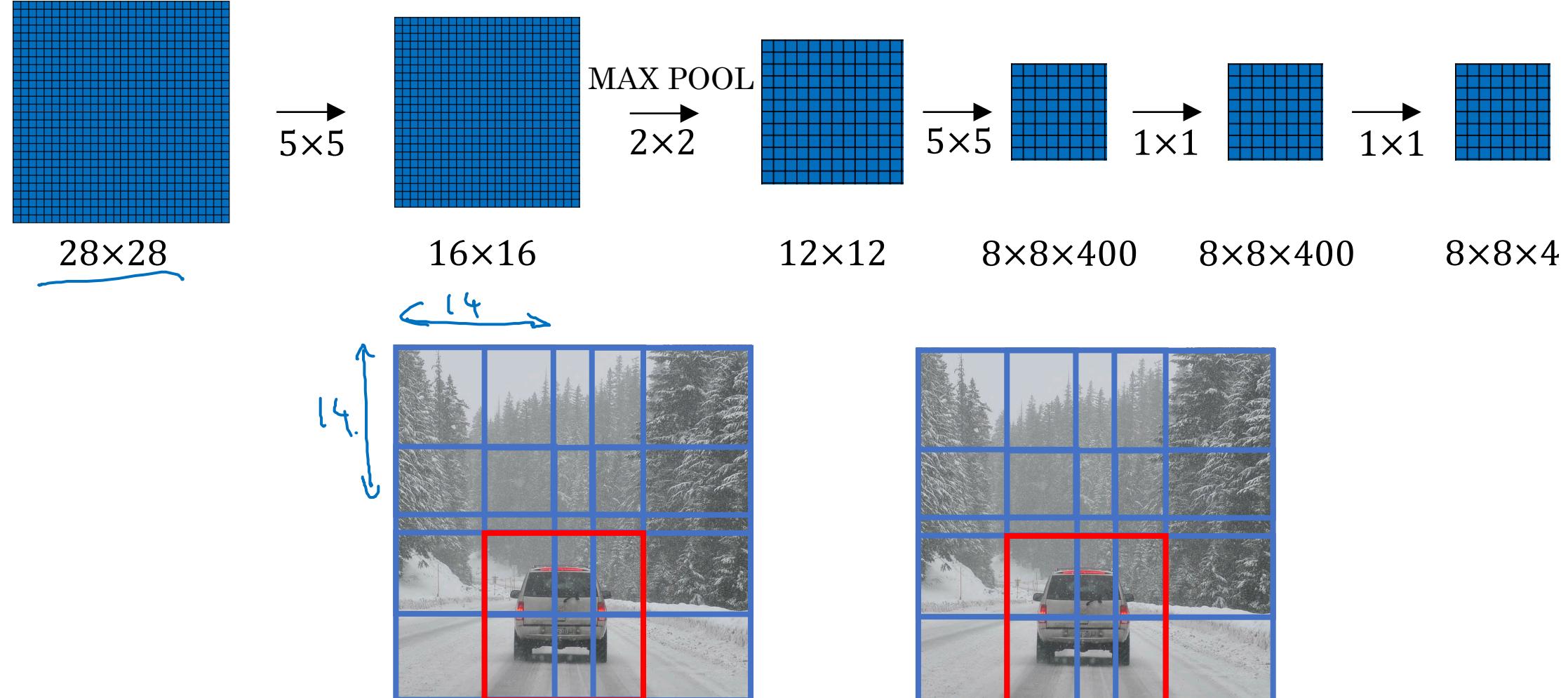
# Turning FC layer into convolutional layers



# Convolution implementation of sliding windows



# Convolution implementation of sliding windows





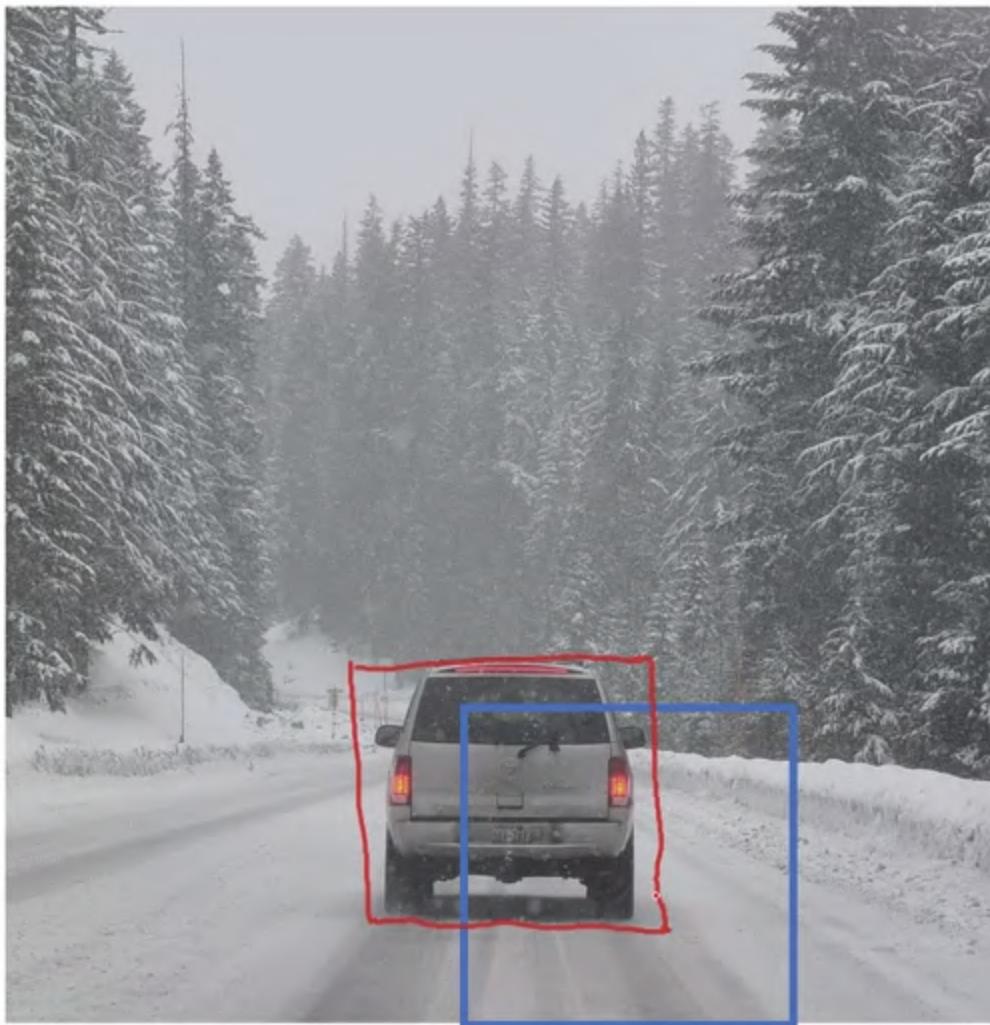
deeplearning.ai

# Object Detection

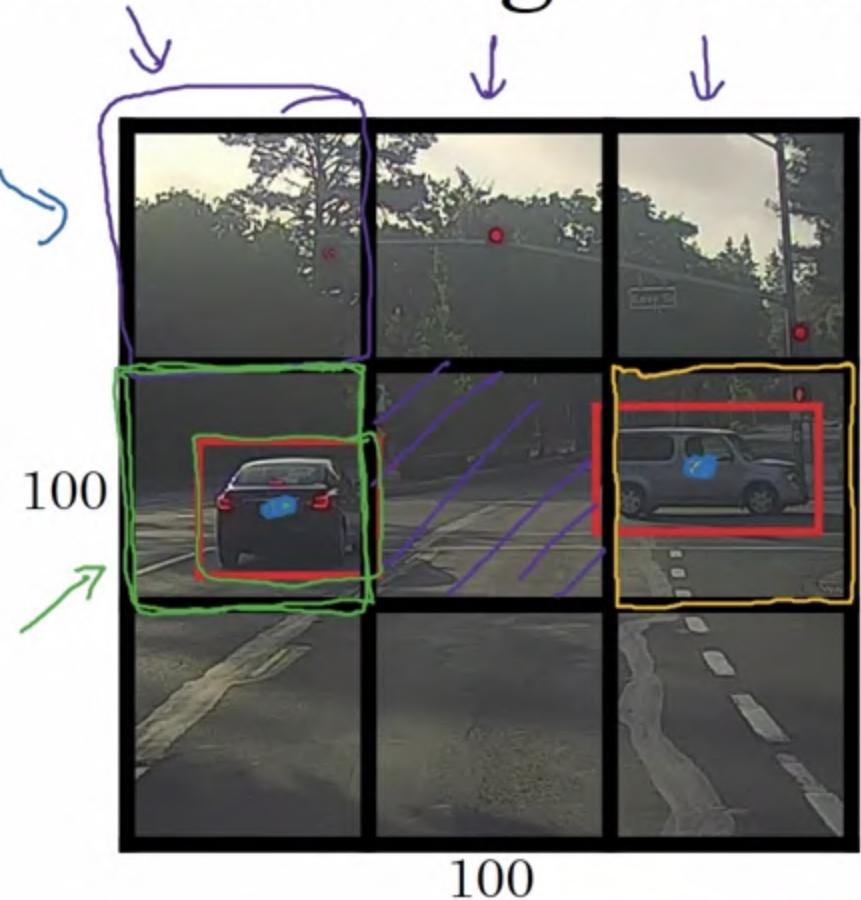
---

Bounding box  
predictions

# Output accurate bounding boxes



# YOLO algorithm



100

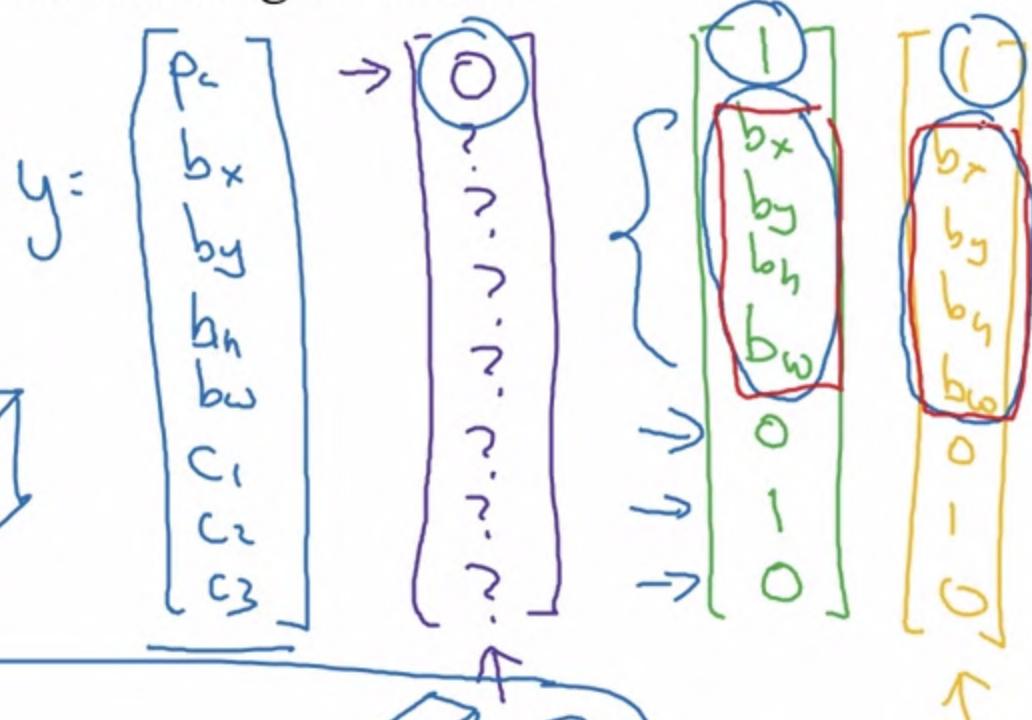
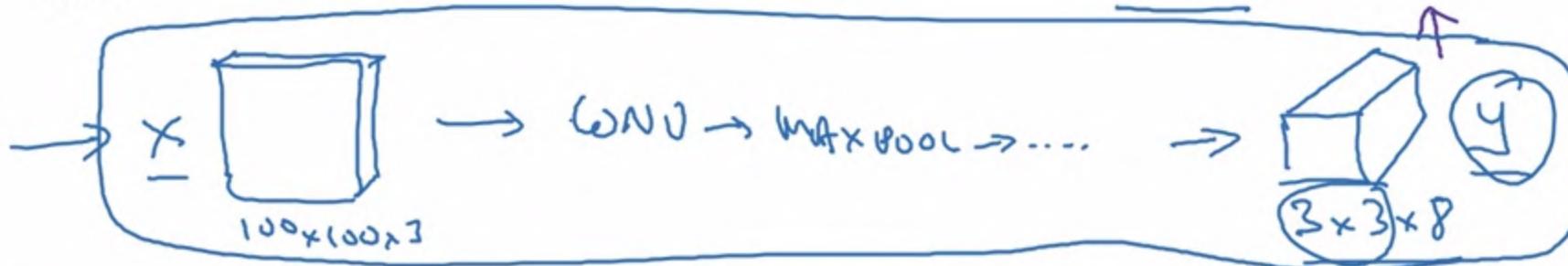
Target output:

$$\underline{3 \times 3 \times 8}$$

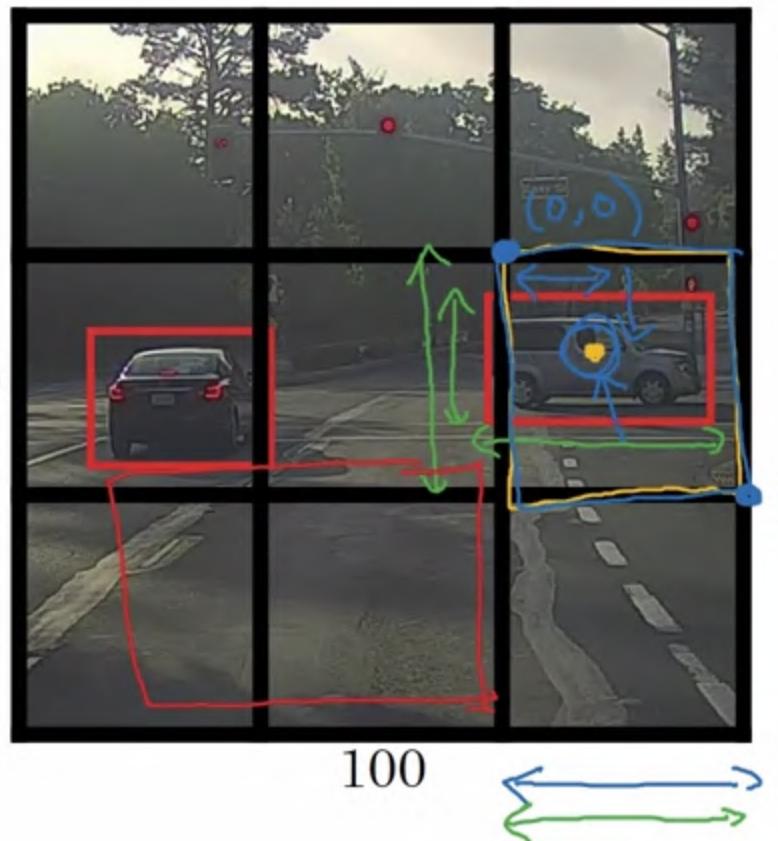


Labels for training

For each grid cell:

361 $\rightarrow 19 \times 19 \times 8$

# Specify the bounding boxes



$$y = \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ o \\ o \end{bmatrix}$$

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



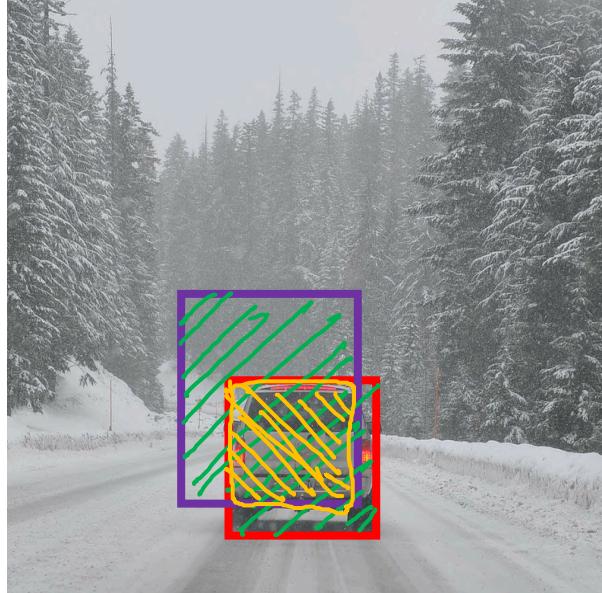
deeplearning.ai

# Object Detection

---

Intersection  
over union

# Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of intersection}}{\text{Size of union}}$$

“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

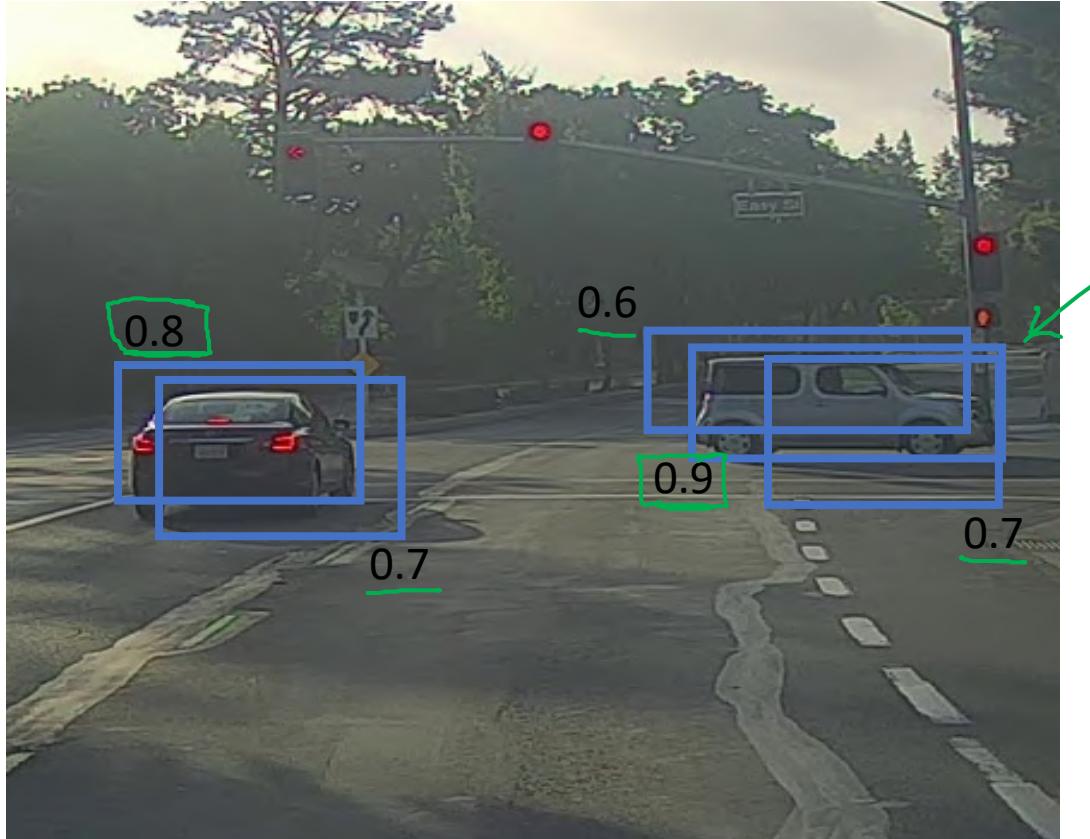


# Non-max suppression example

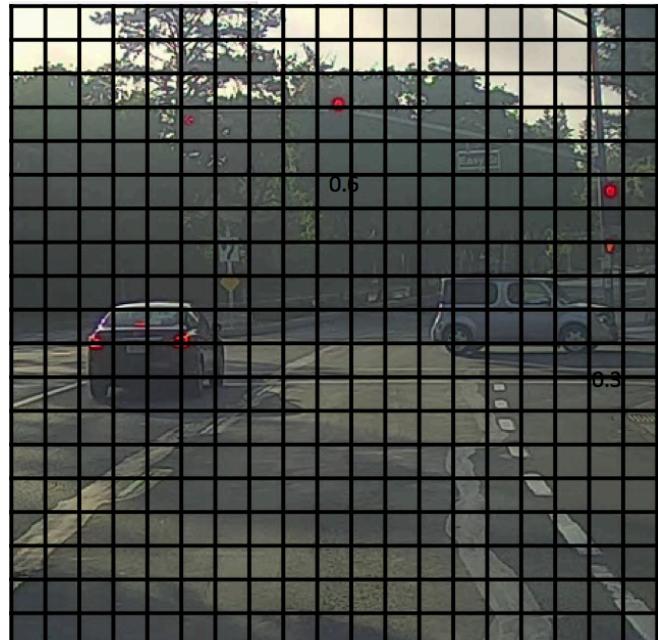


19x19

# Non-max suppression example

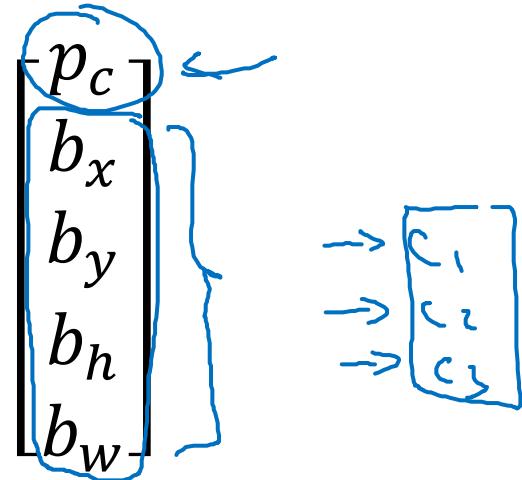


# Non-max suppression algorithm



19×19

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



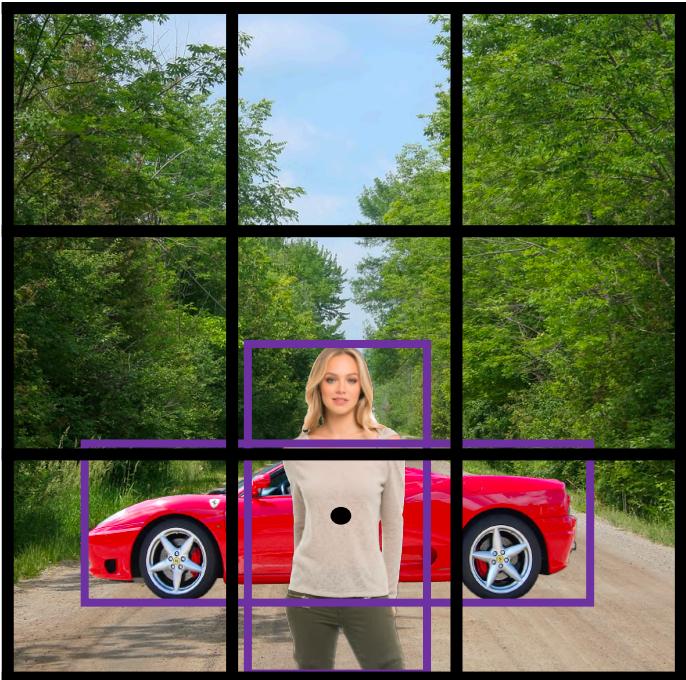
deeplearning.ai

# Object Detection

---

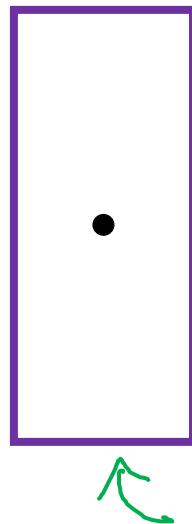
## Anchor boxes

# Overlapping objects:

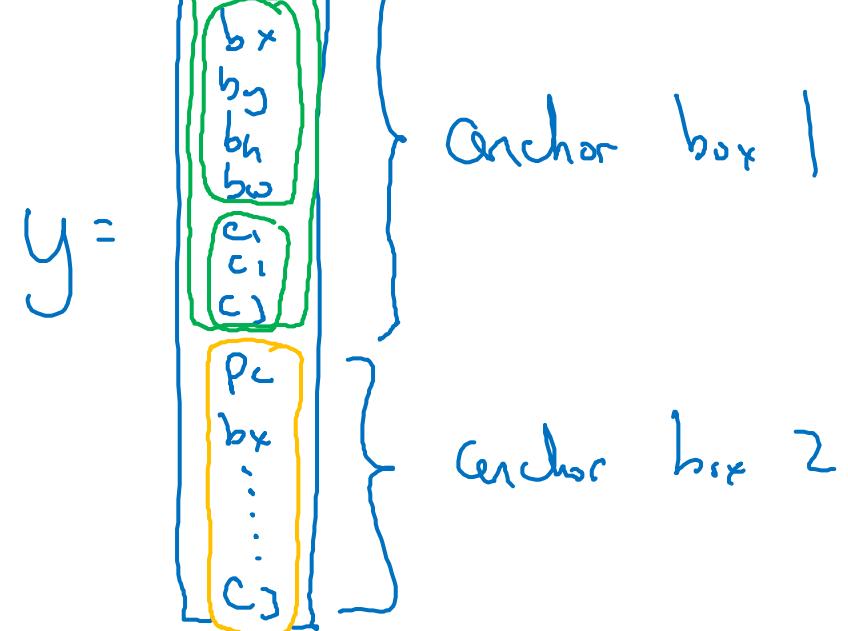
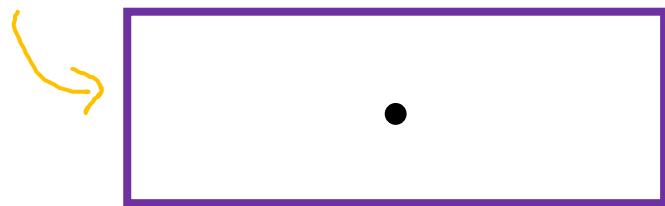


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



Anchor box 2:



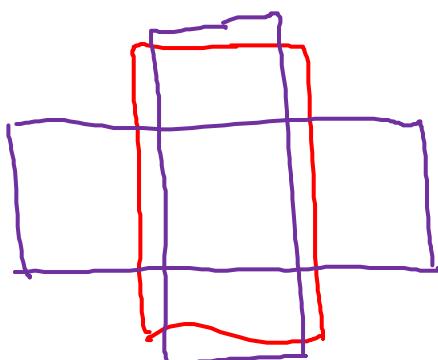
# Anchor box algorithm

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.

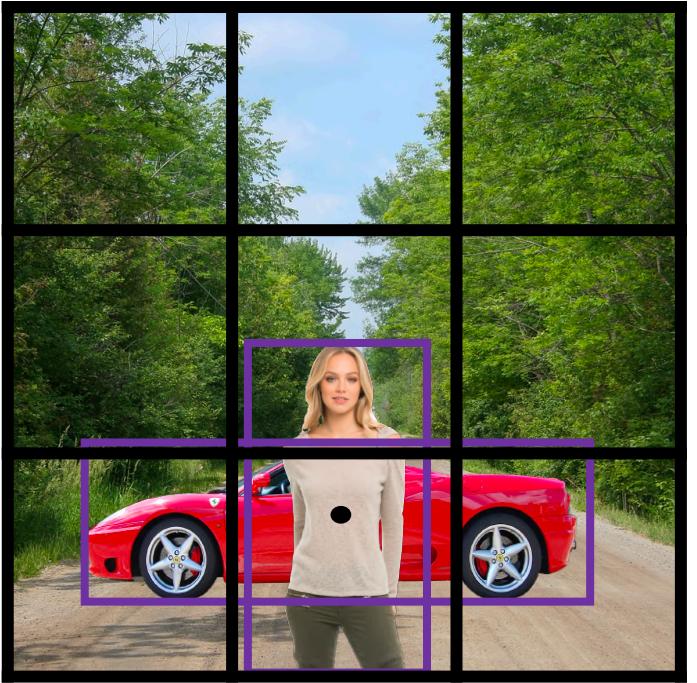
(grid cell, anchor box)

Output y:

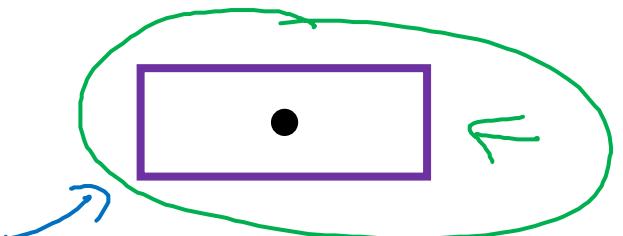
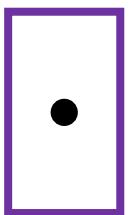
$3 \times 3 \times 16$

$3 \times 3 \times 2 \times 8$

# Anchor box example



Anchor box 1:    Anchor box 2:



$$y =$$

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

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

Car only?

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

Anchor box 1

Anchor box 2



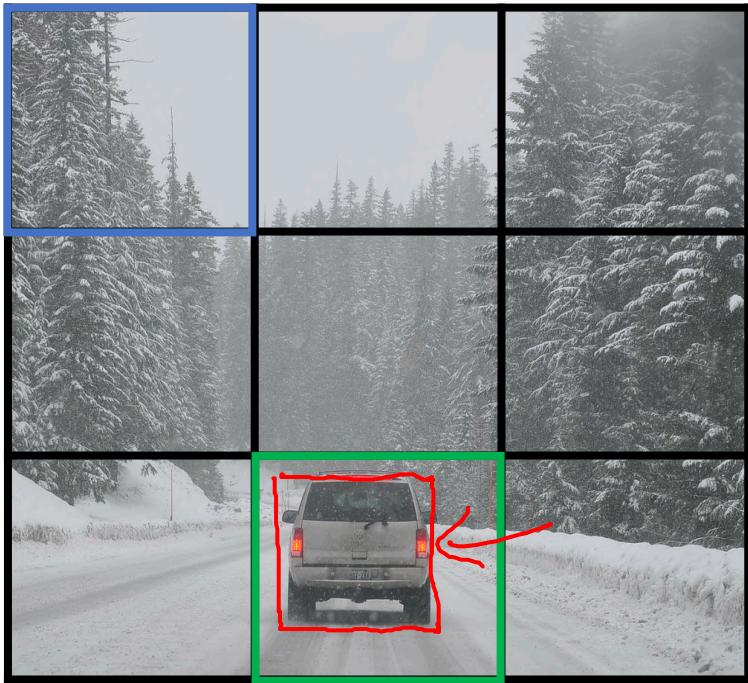
deeplearning.ai

# Object Detection

---

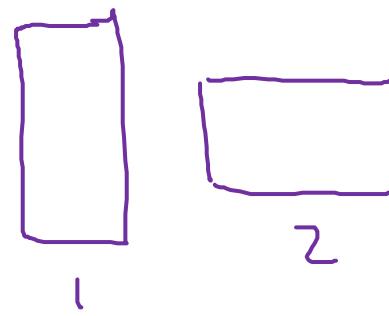
Putting it together:  
YOLO algorithm

# Training



- 1 - pedestrian
- 2 - car
- 3 - motorcycle

$y =$

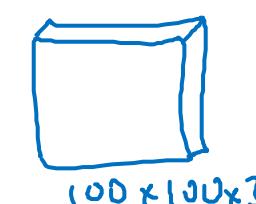
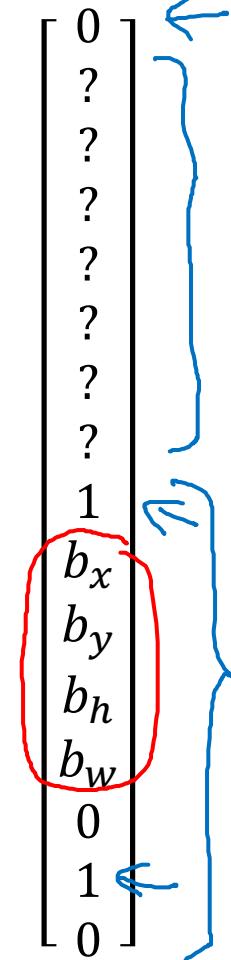
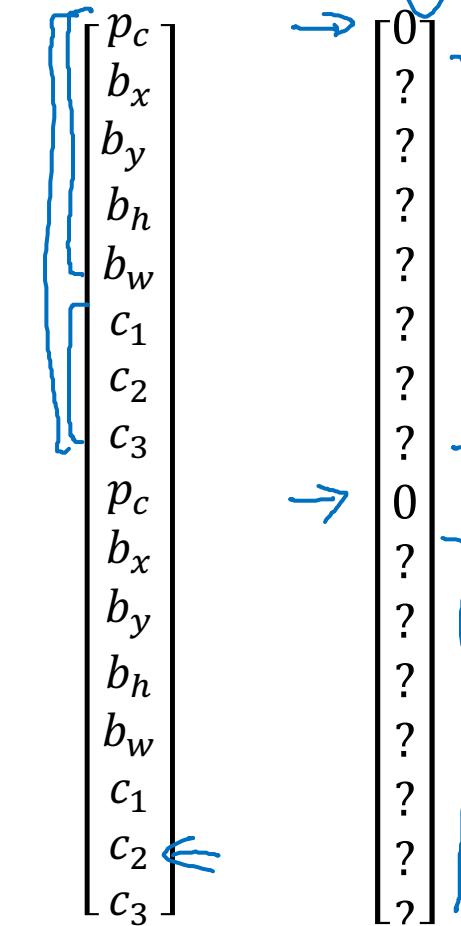


$y$  is  $3 \times 3 \times 2 \times 8$

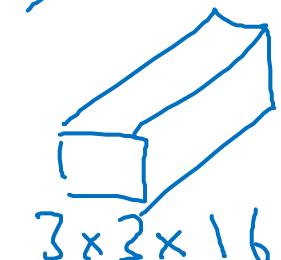
$3 \times 3 \times 16$

$19 \times 19 \times 16$   
 $19 \times 19 \times 40$

#anchors  $\rightarrow 5 + \#classes$

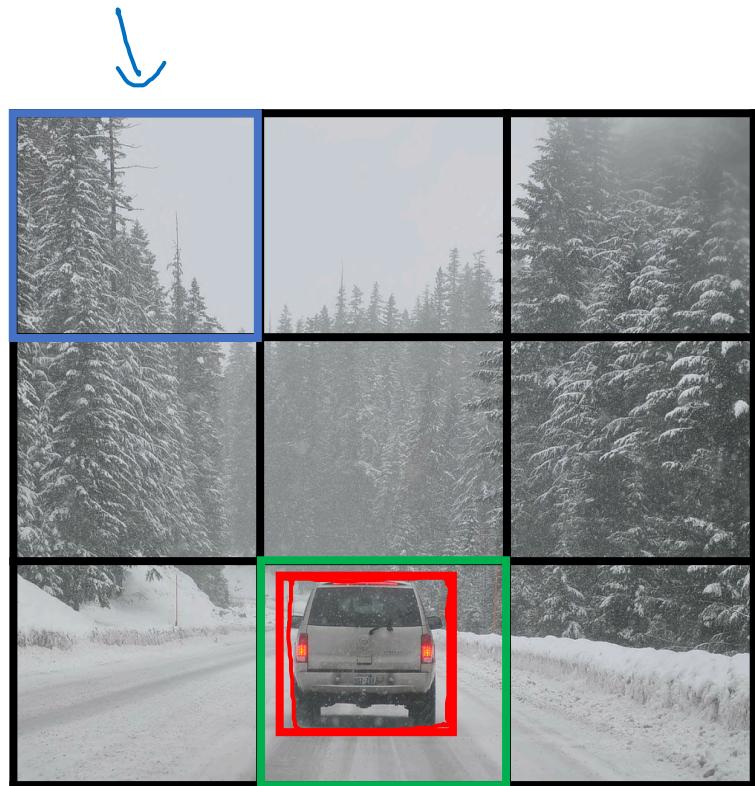


$\rightarrow$  ConvNet

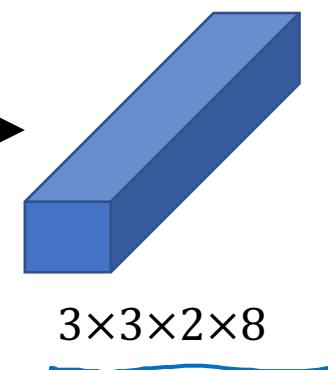


Andrew Ng

# Making predictions



→ ... →



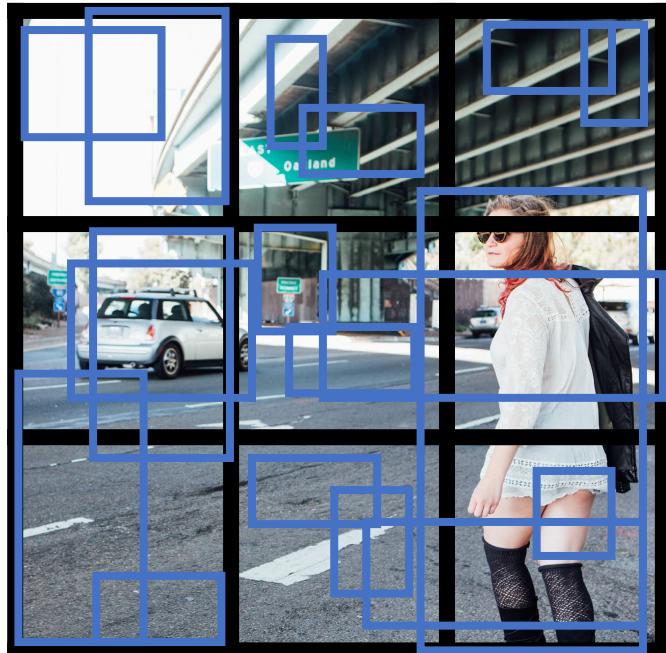
$y =$

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

Annotations on the right side of the equation:

- A blue bracket groups the first five elements:  $p_c, b_x, b_y, b_h, b_w$ . An arrow points from this group to the first row of the matrix.
- A green bracket groups the next three elements:  $c_1, c_2, c_3$ . An arrow points from this group to the second row of the matrix.
- Red annotations label the last four elements:
  - $b_x$  is written vertically next to  $b_x$ .
  - $b_y$  is written vertically next to  $b_y$ .
  - $b_h$  is written vertically next to  $b_h$ .
  - $b_w$  is written vertically next to  $b_w$ .
- A blue arrow points upwards from the bottom of the matrix towards the dimension labels.

# 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.



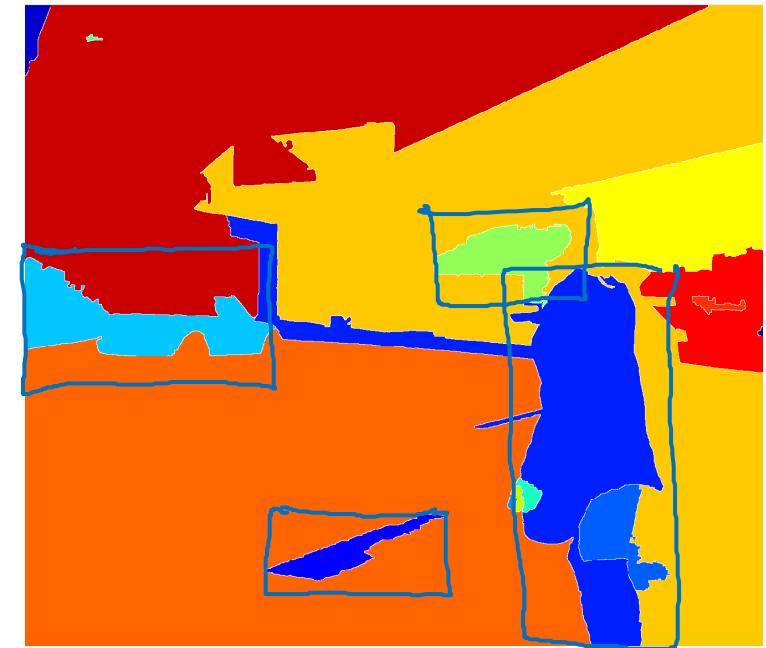
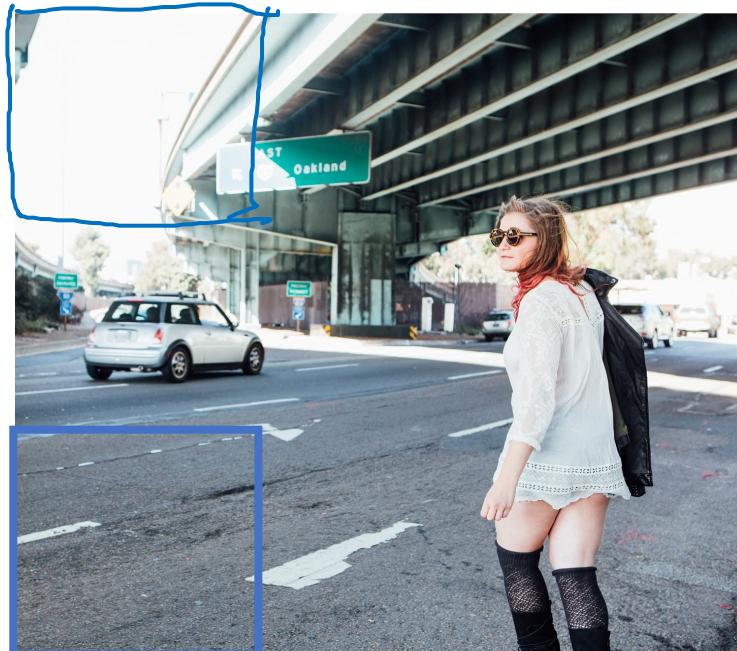
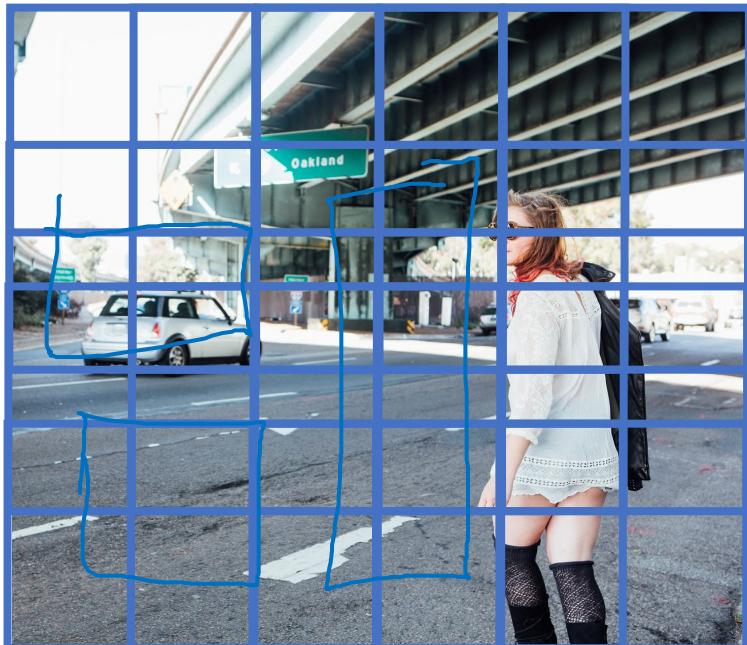
deeplearning.ai

# 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



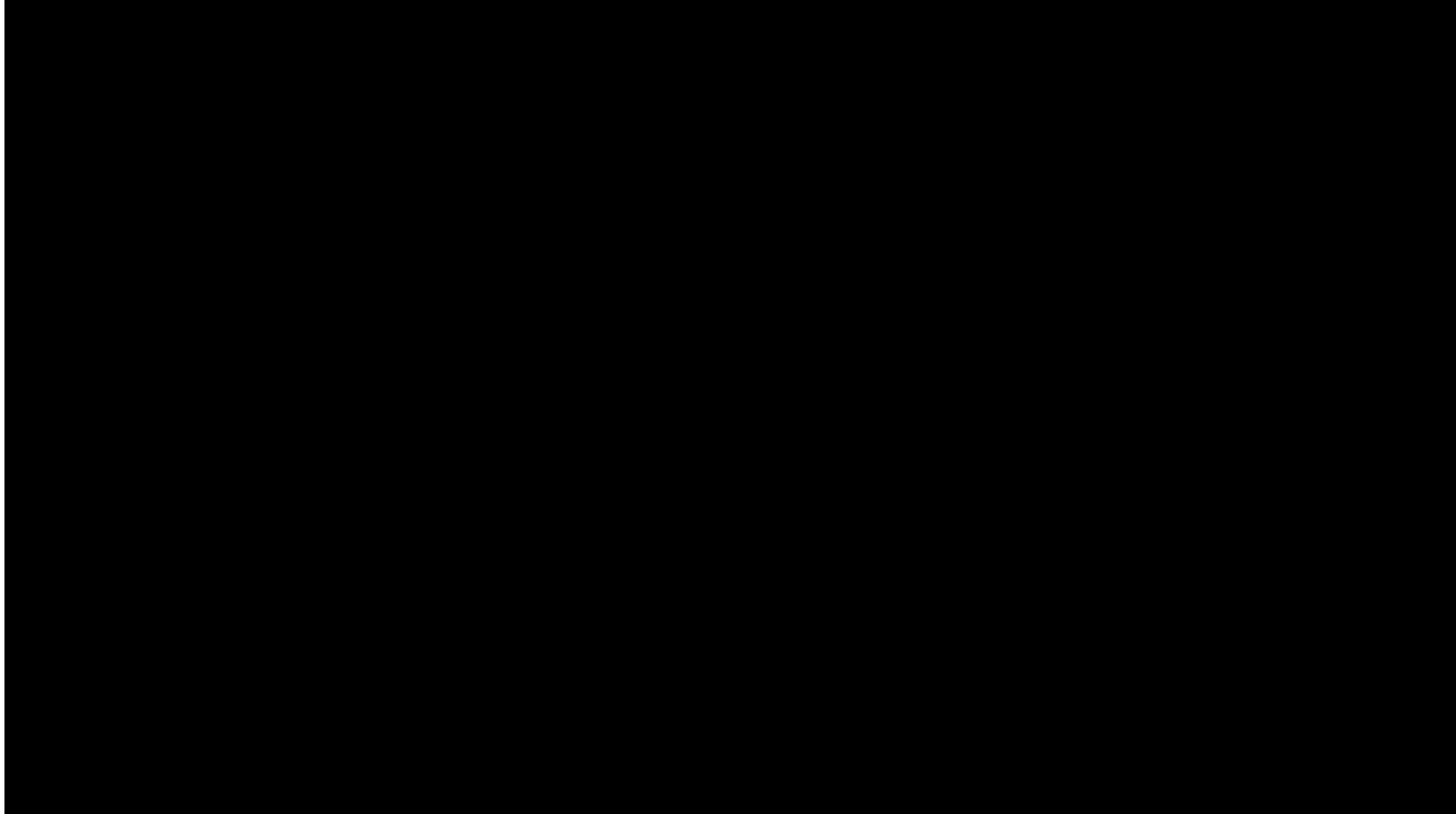
deeplearning.ai

# 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 ←



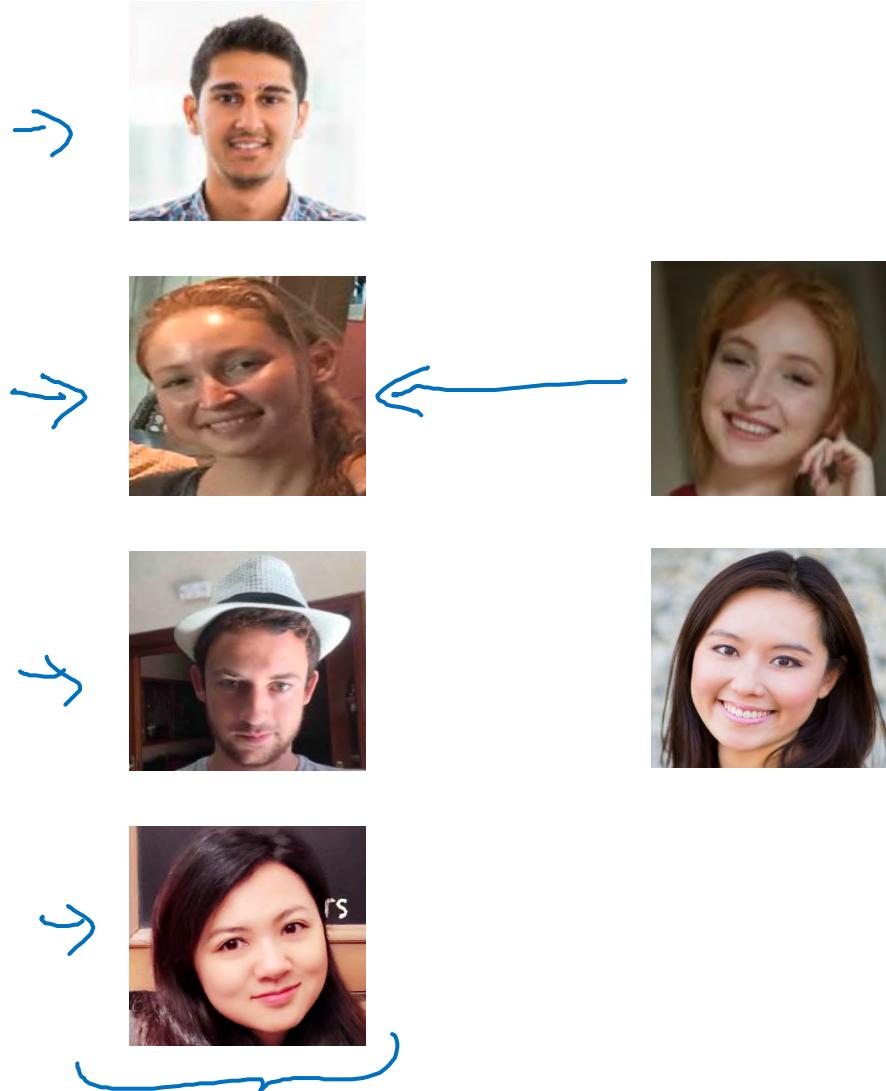
deeplearning.ai

# Face recognition

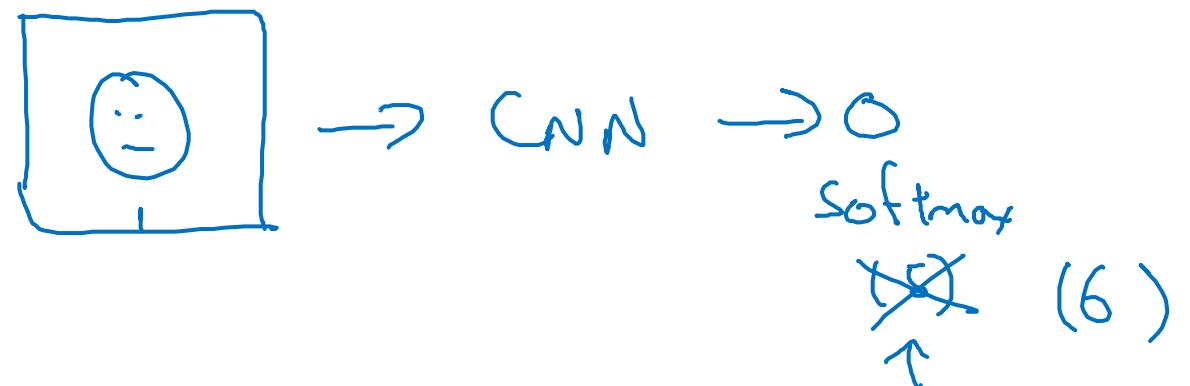
---

# One-shot learning

# One-shot learning



Learning from one example to recognize the person again



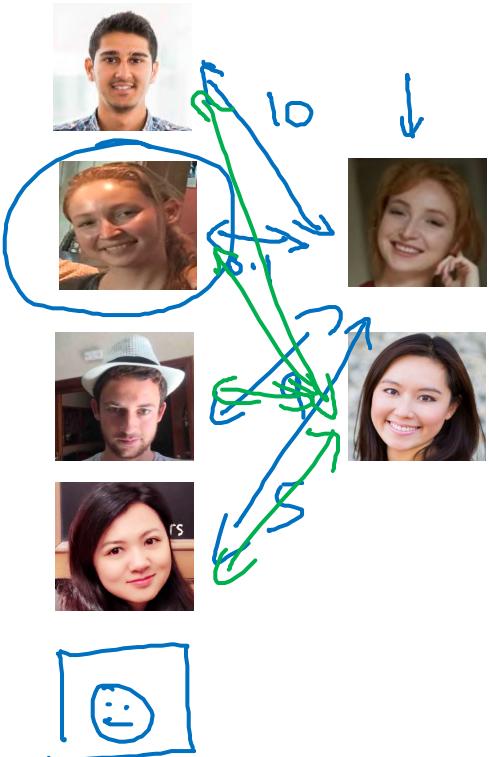
# Learning a “similarity” function

→  $d(\underline{\text{img1}}, \underline{\text{img2}})$  = degree of difference between images

If  $d(\text{img1}, \text{img2}) \leq \tau$   
 $> \tau$

“some”  
“different”

} Verification.



$d(\text{img1}, \text{img2})$



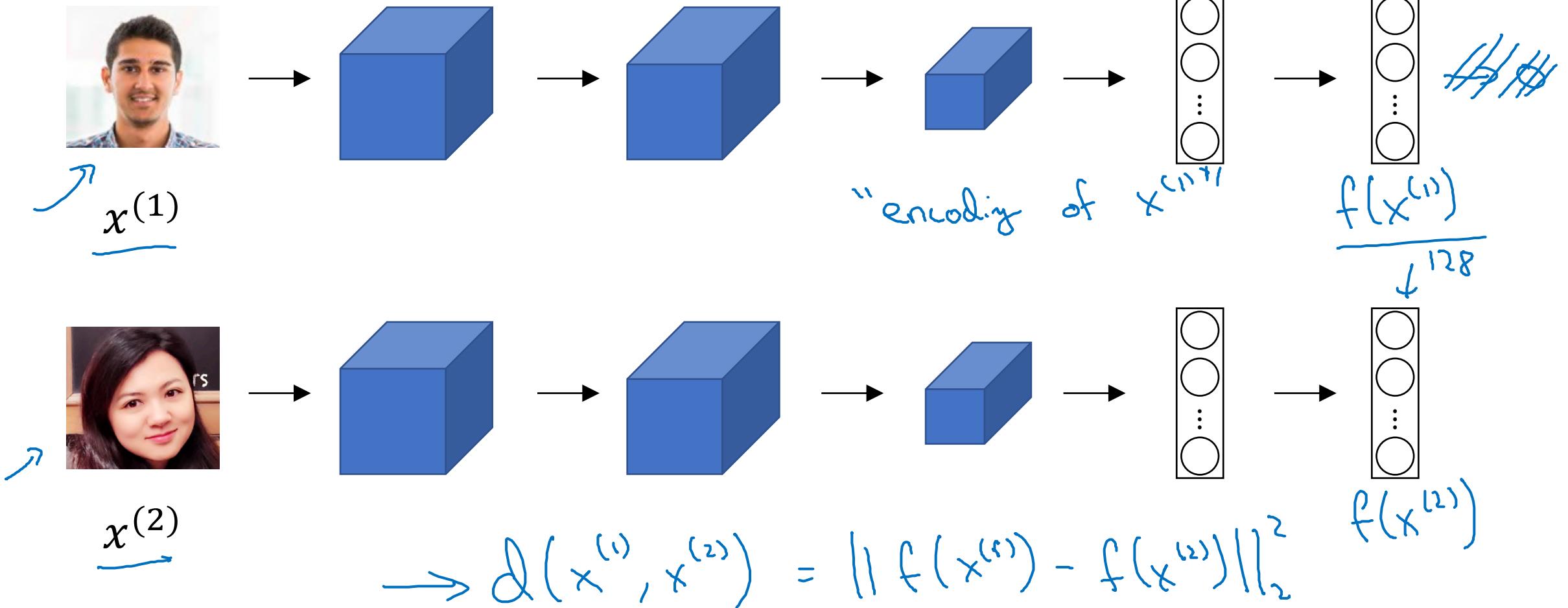
deeplearning.ai

# Face recognition

---

## Siamese network

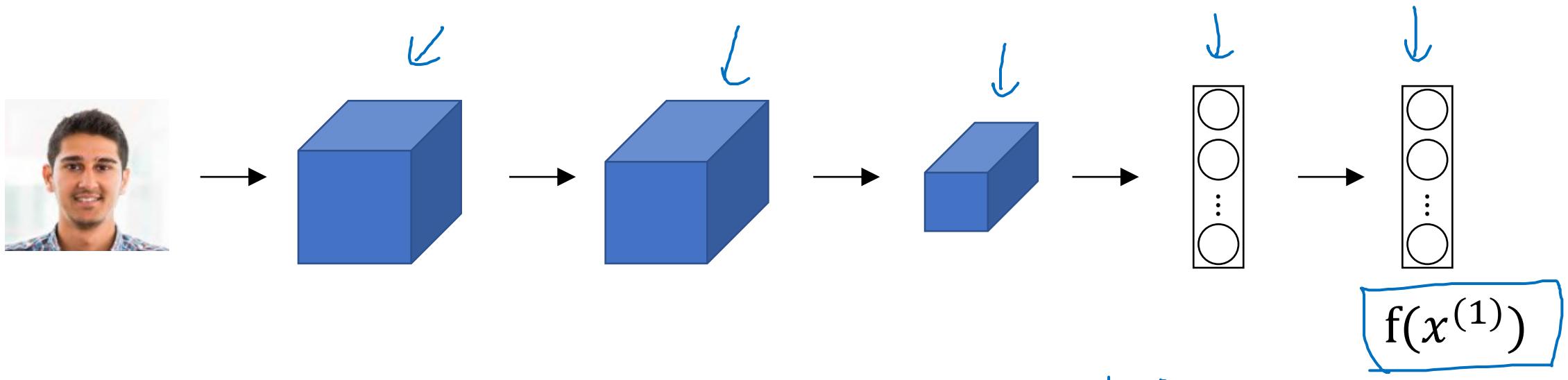
# Siamese network



[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.

}



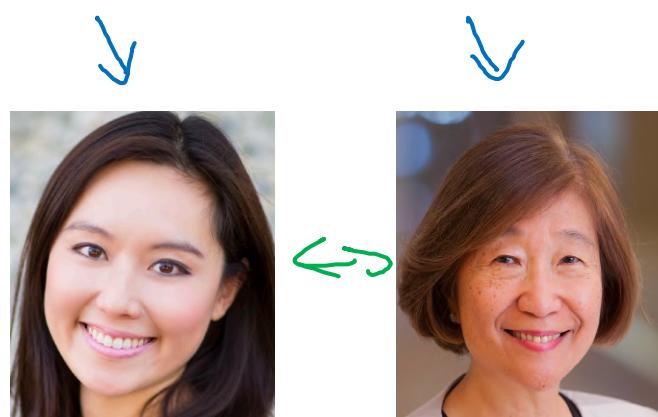
deeplearning.ai

# 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.

$$\underbrace{\|f(A) - f(P)\|^2}_{\text{Distance between } A \text{ and } P} + \alpha \leq \underbrace{\|f(A) - f(N)\|^2}_{\text{Distance between } A \text{ and } N}$$

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

$$d(x^{(i)}, x^{(j)})$$



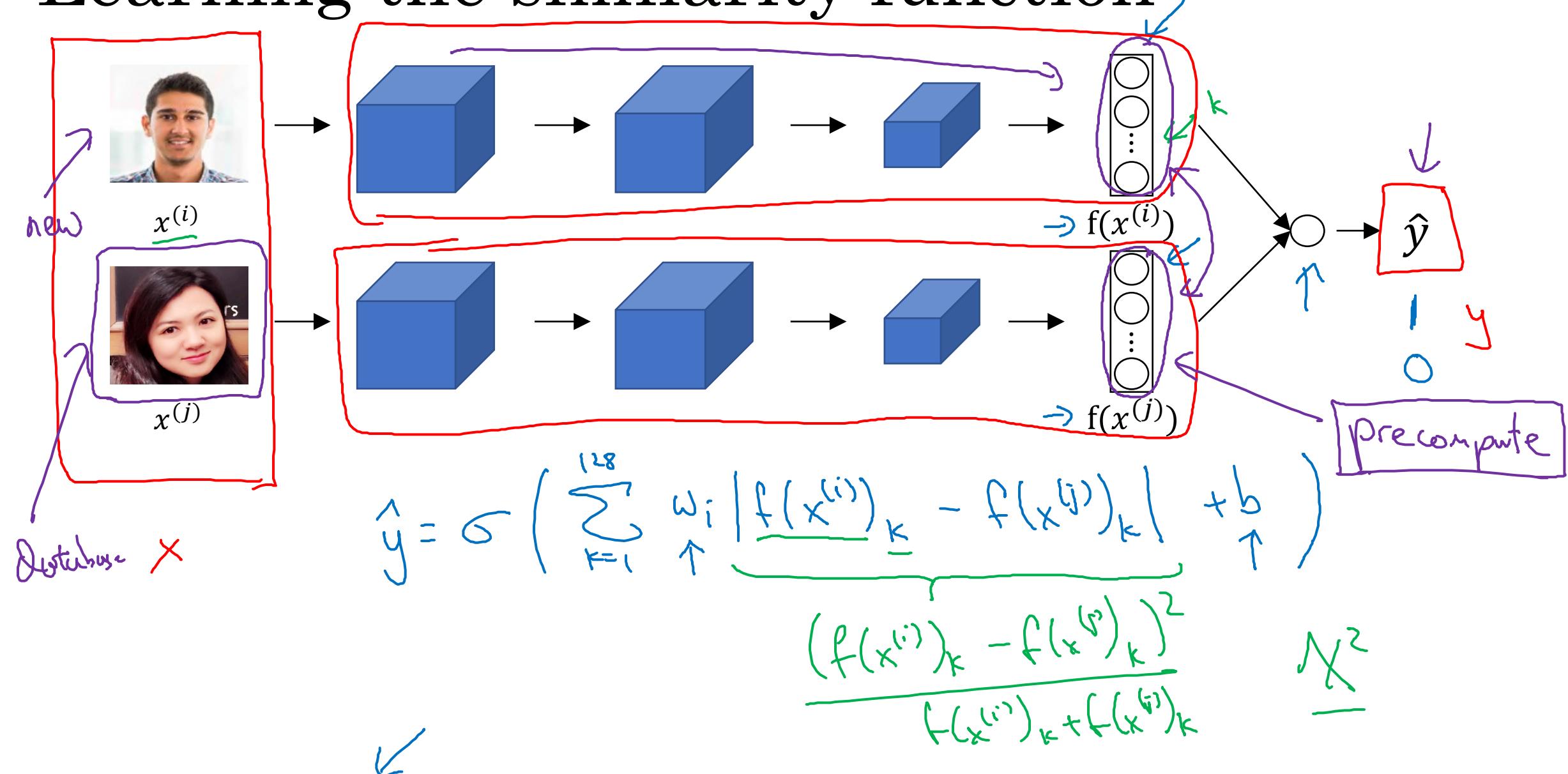
deeplearning.ai

# Face recognition

---

# Face verification and binary classification

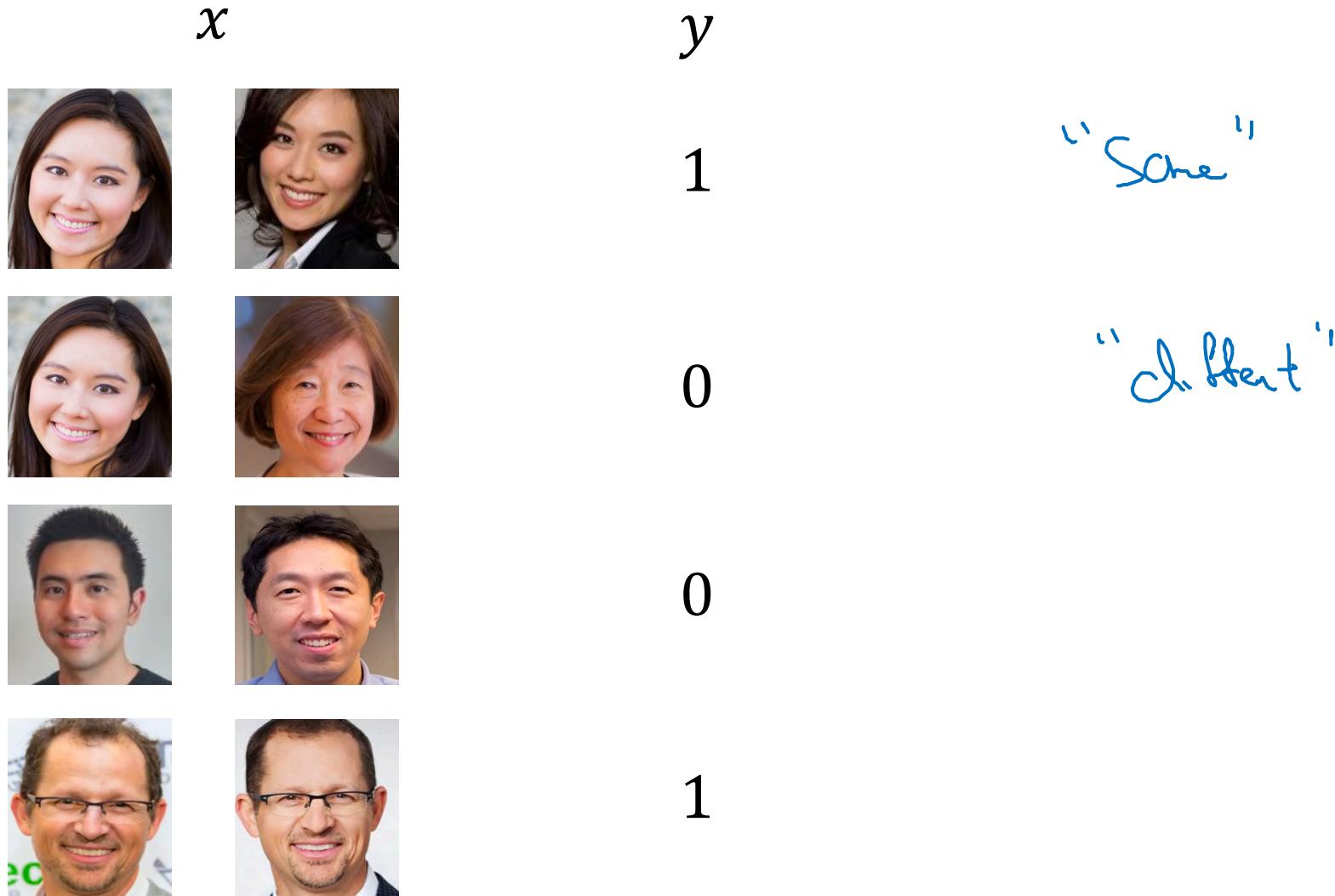
# Learning the similarity function



[Taigman et. al., 2014. DeepFace closing the gap to human level performance]

Andrew Ng

# Face verification supervised learning





deeplearning.ai

# Neural Style Transfer

---

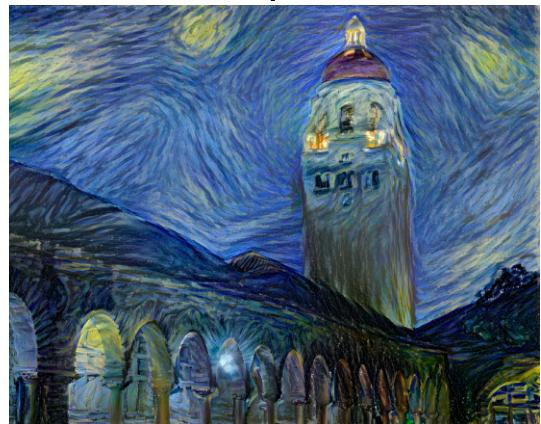
What is neural style  
transfer?

# Neural style transfer



Content ( $c$ )

Style ( $s$ )



Generated image ( $G$ )



Content ( $c$ )

Style ( $s$ )



Generated image ( $G$ )



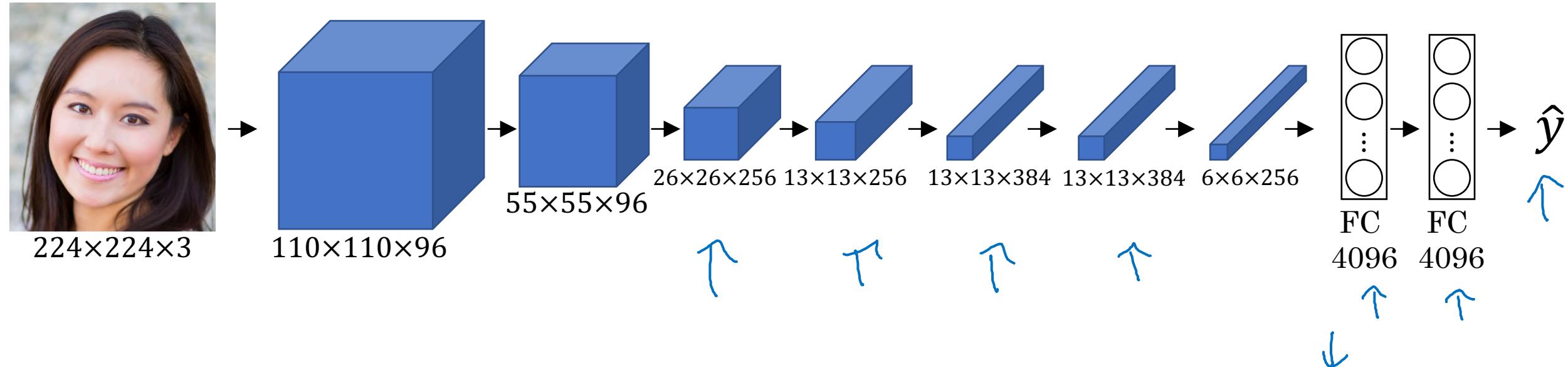
deeplearning.ai

# Neural Style Transfer

---

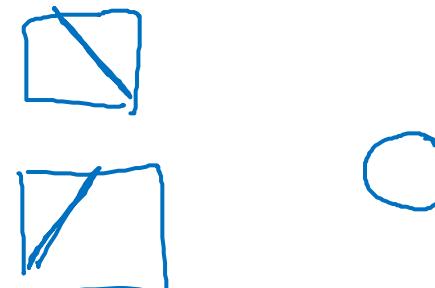
What are deep  
ConvNets learning?

# Visualizing what a deep network is learning



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

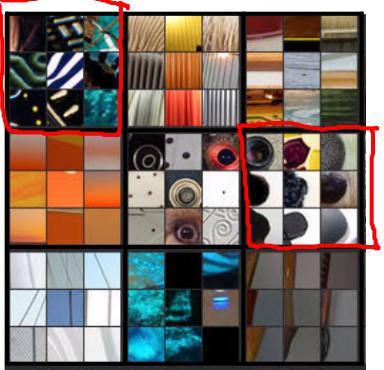
Repeat for other units.



# Visualizing deep layers



Layer 1



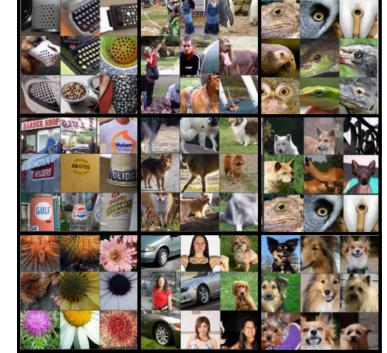
Layer 2



Layer 3

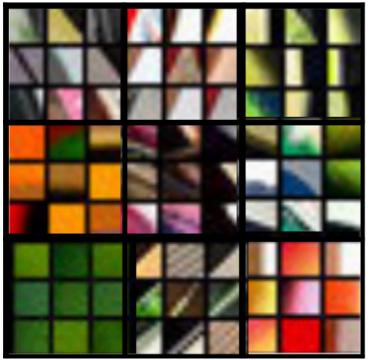


Layer 4



Layer 5

# Visualizing deep layers: Layer 1



Layer 1



Layer 2



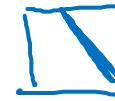
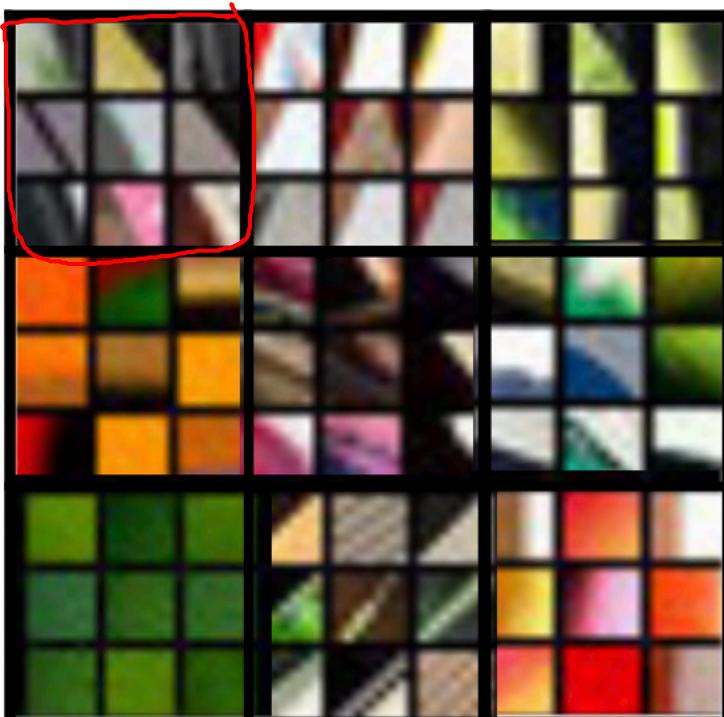
Layer 3



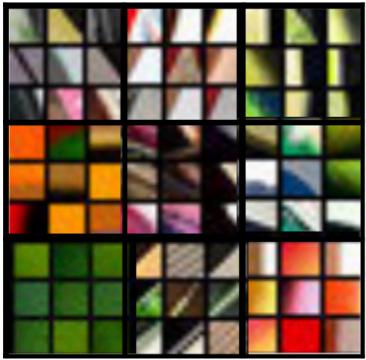
Layer 4



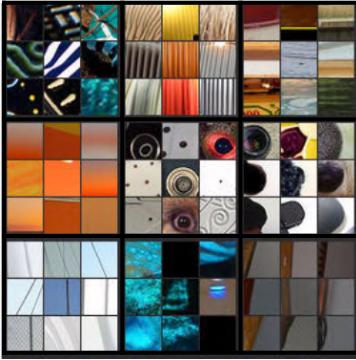
Layer 5



# Visualizing deep layers: Layer 2



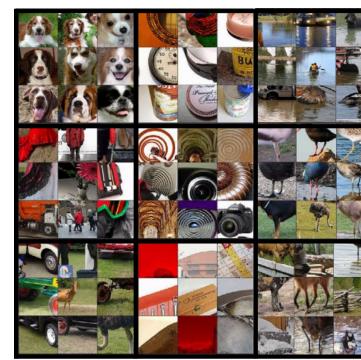
Layer 1



Layer 2



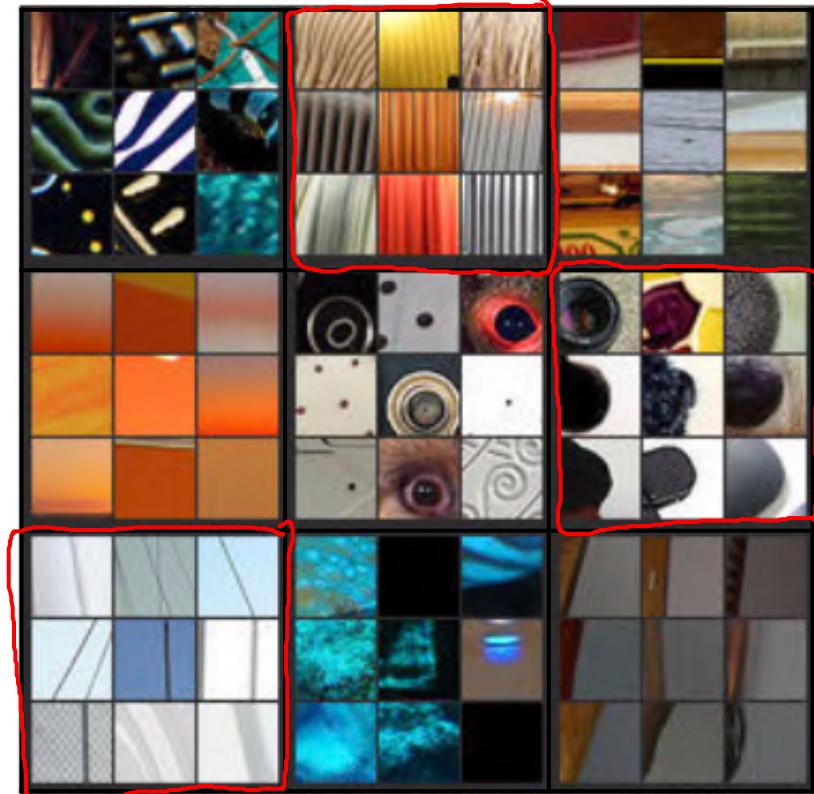
Layer 3



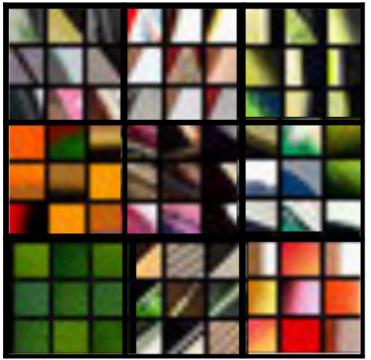
Layer 4



Layer 5



# Visualizing deep layers: Layer 3



Layer 1



Layer 2



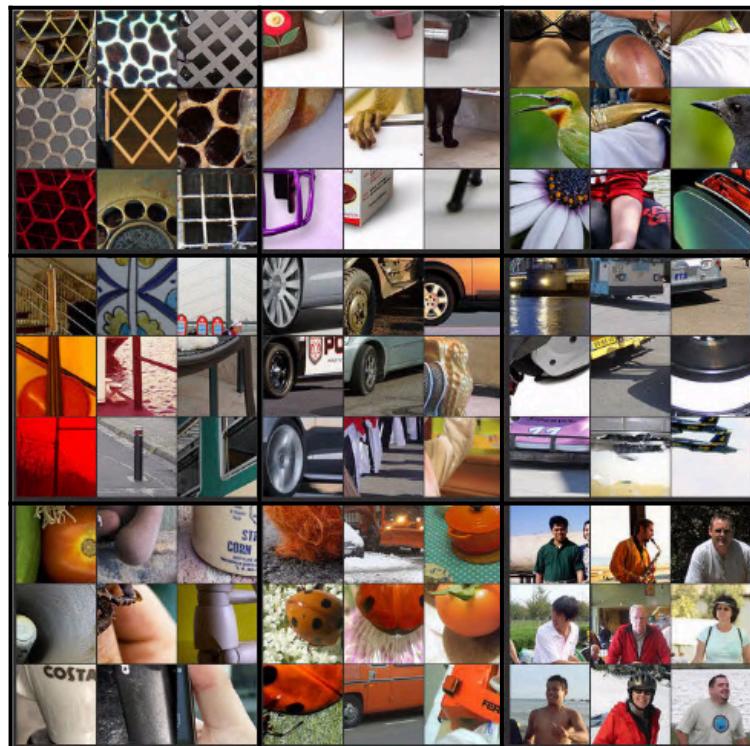
Layer 3



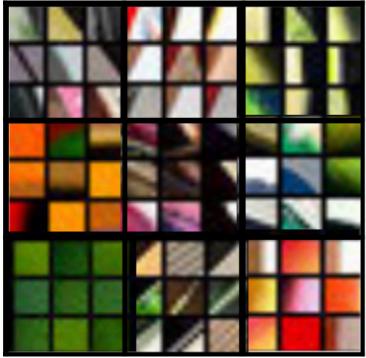
Layer 4



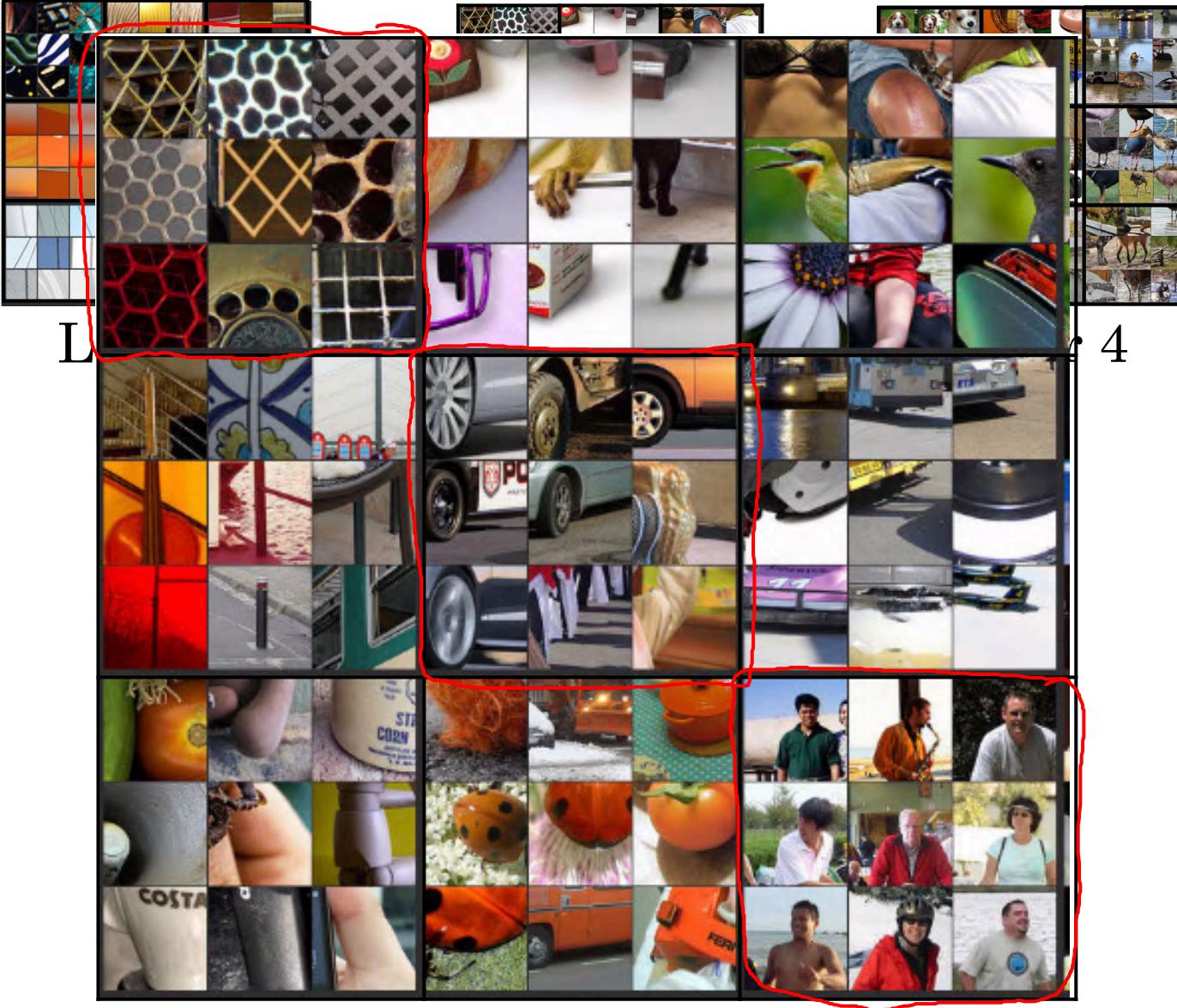
Layer 5



# Visualizing deep layers: Layer 3

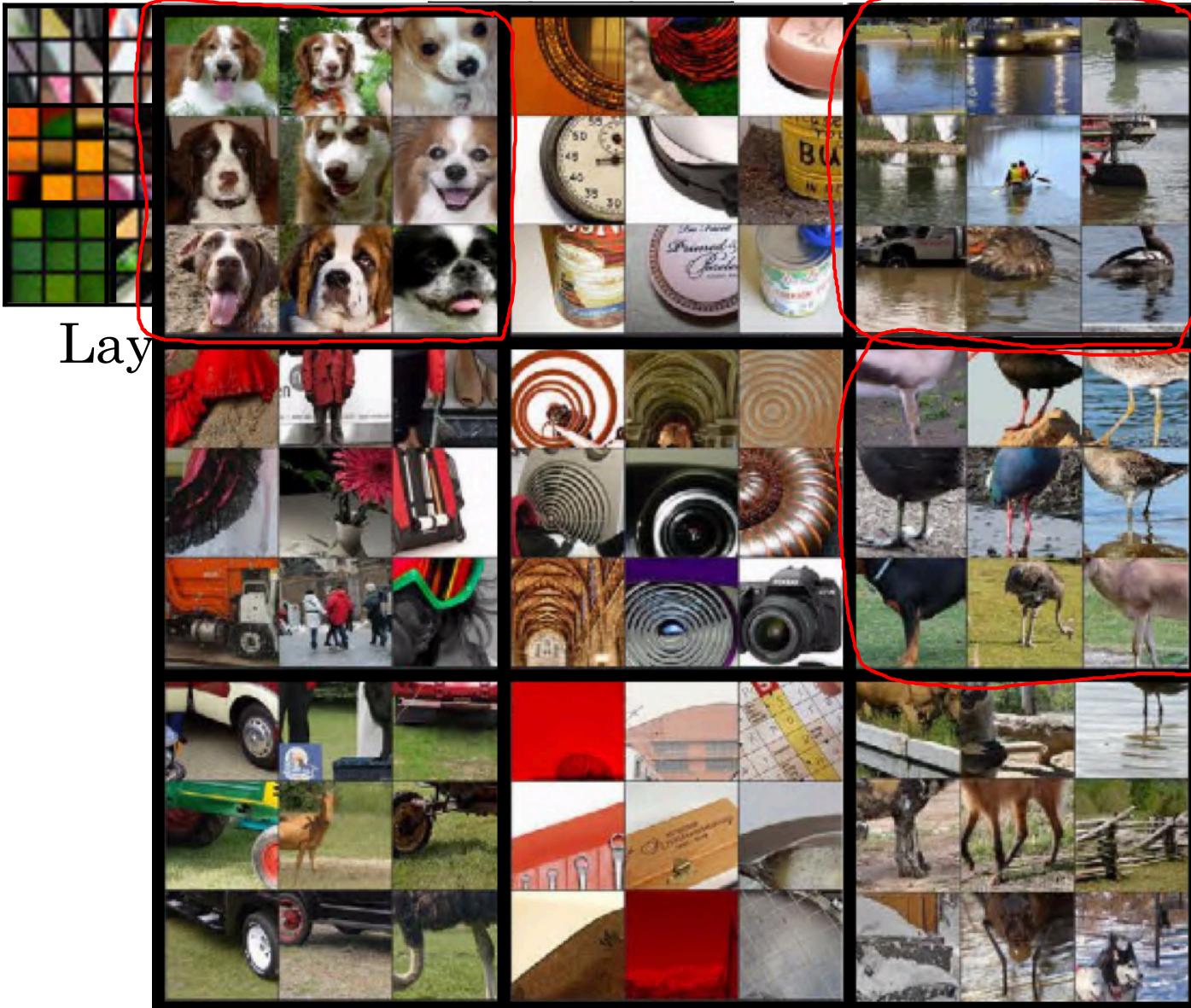


Layer 1



Layer 5

# Visualizing deep layers: Layer 4



Layer 4

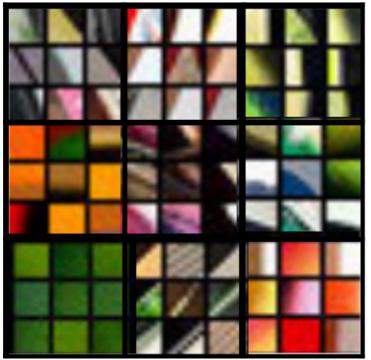


Layer 4



Layer 5

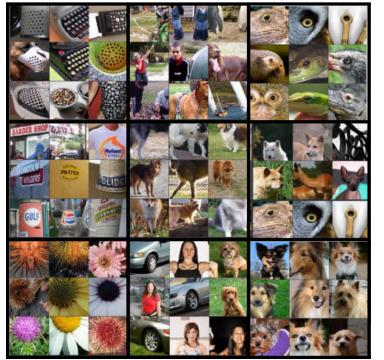
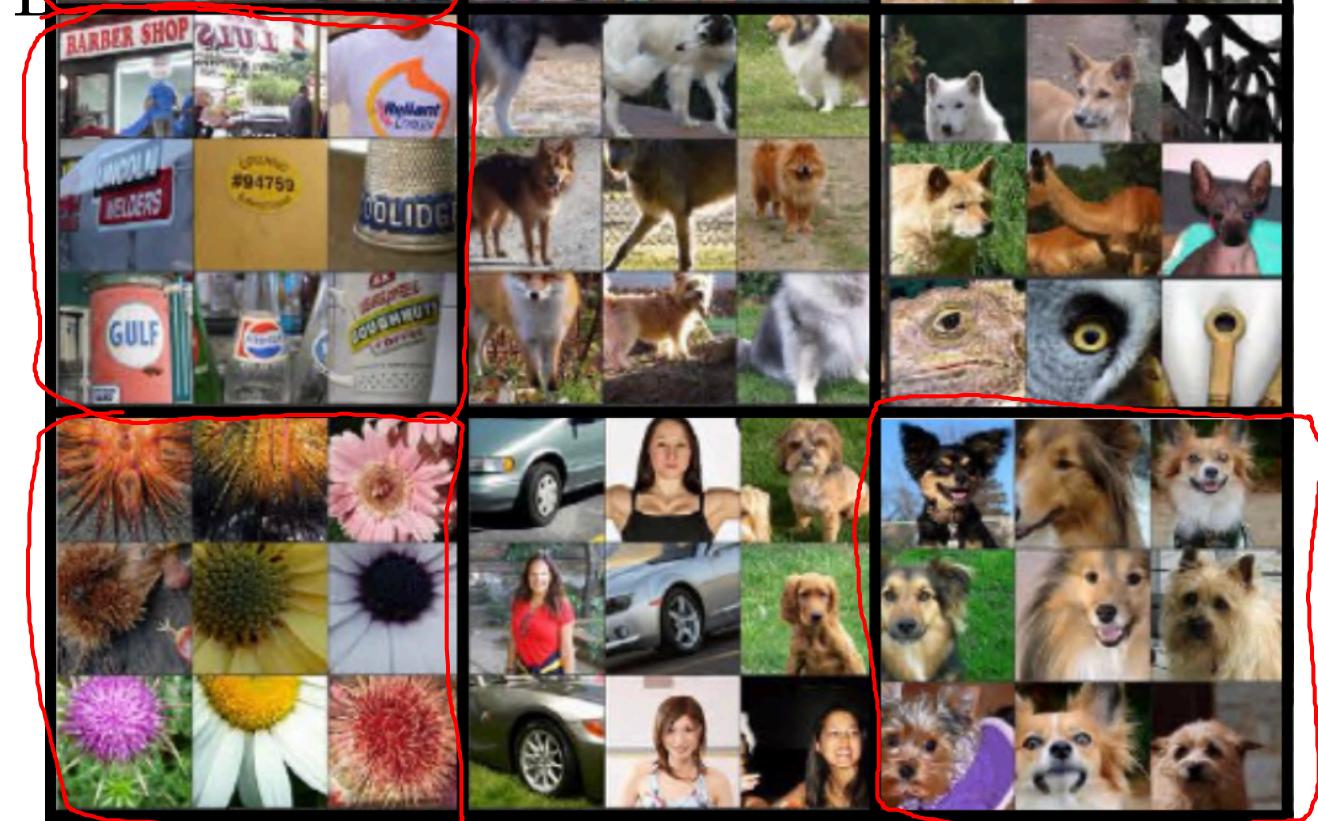
# Visualizing deep layers: Layer 5



Layer 1



L



Layer 5



deeplearning.ai

# Neural Style Transfer

---

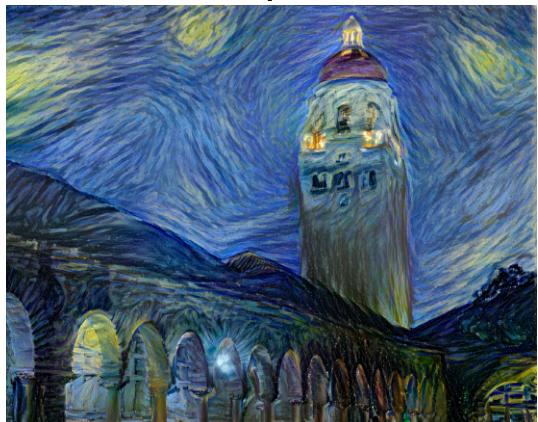
## Cost function

# 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

$G: \underbrace{100 \times 100}_{\text{---}} \times \underbrace{3}_{\text{---}}$

$\uparrow_{\text{RGB}}$



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

$$J_{content}(C, G) = \frac{1}{2} \left\| \underline{a}^{[l](C)} - \underline{a}^{[l](G)} \right\|^2$$



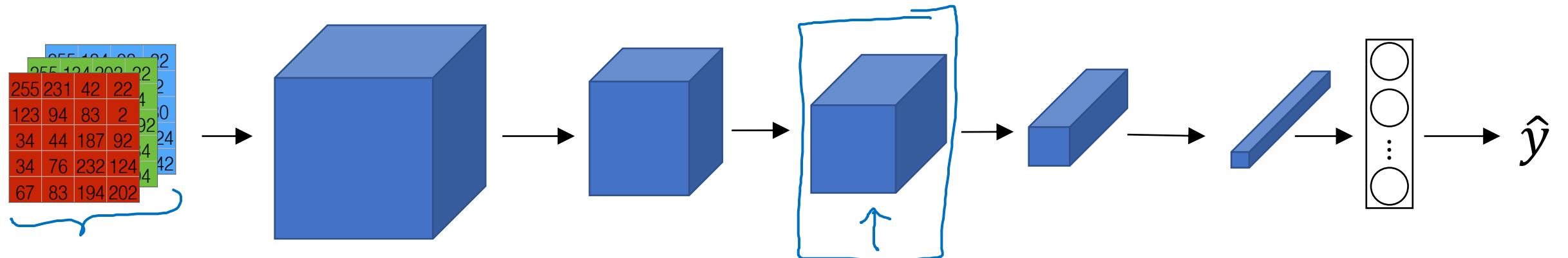
deeplearning.ai

# Neural Style Transfer

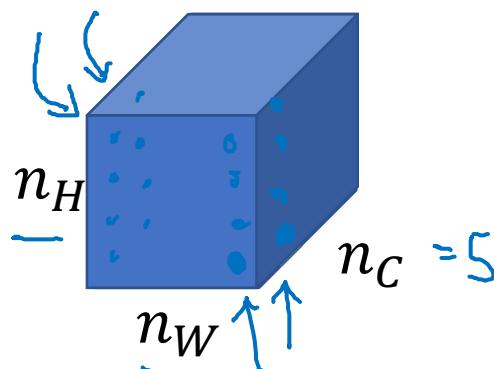
---

## Style cost function

# Meaning of the “style” of an image

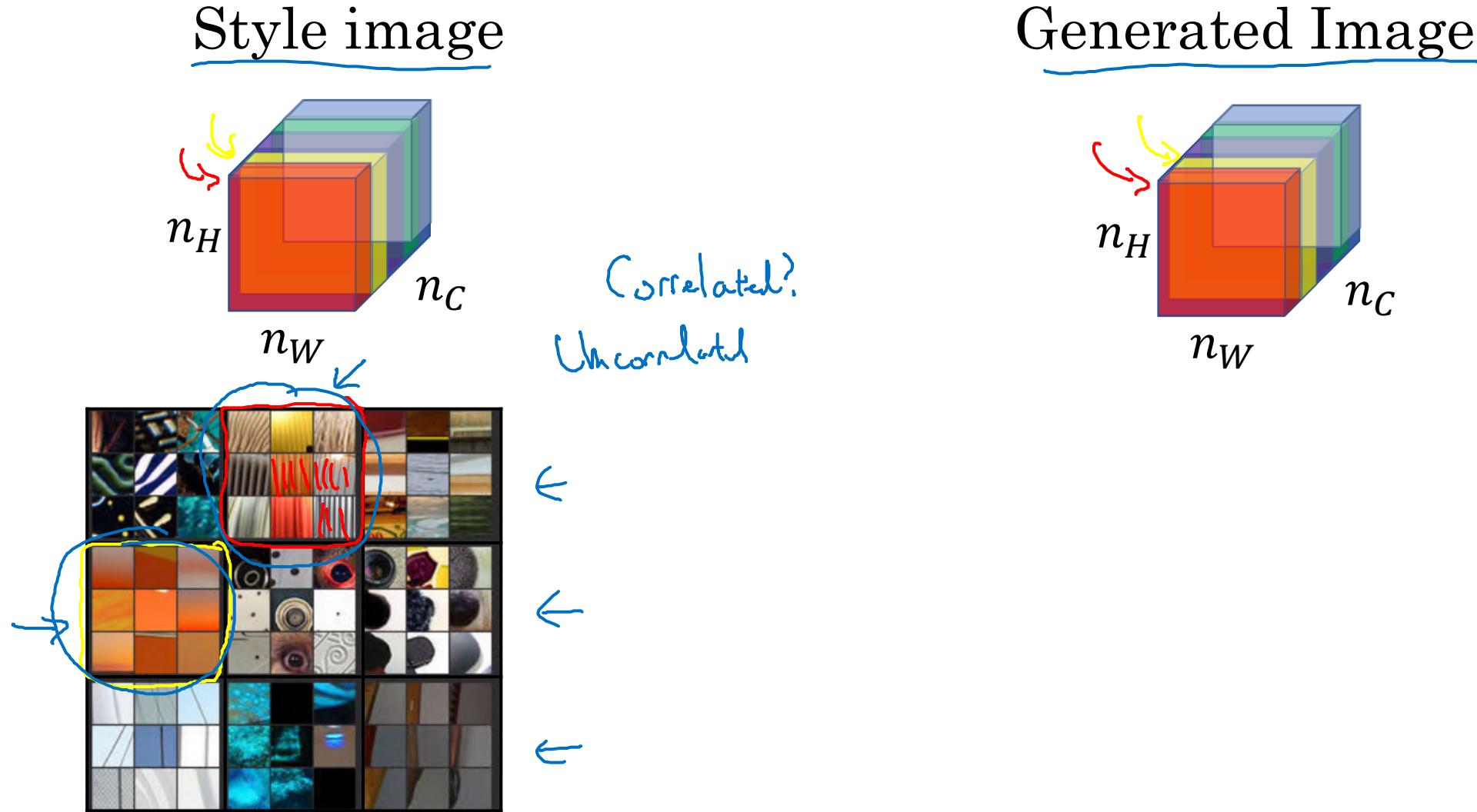


Say you are using layer  $l$ 's activation to measure “style.”  
Define style as correlation between activations across channels.



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  
↓ ↓ ↗

$n_c$

$$G_{kk'}^{[l]} \quad \text{for } k, k' = 1, \dots, n_c^{[l]}$$

"Gram matrix"

$$\begin{aligned} J_{\text{style}}^{[l]}(S, G) &= \frac{1}{(\dots)} \| G_{kk'}^{[l](s)} - G_{kk'}^{[l](G)} \|_F^2 \\ &= \frac{1}{(2n_h^{[l]} n_w^{[l]})^2} \sum_k \sum_{k'} (G_{kk'}^{[l](s)} - G_{kk'}^{[l](G)})^2 \end{aligned}$$

# Style cost function

$$\left\| G^{[l](s)} - G^{[l](G)} \right\|_F^2$$

$$J_{style}^{[l]}(S, G) = \frac{1}{\left(2n_H^{[l]} n_W^{[l]} n_C^{[l]}\right)^2} \sum_k \sum_{k'} (G_{kk'}^{[l](s)} - G_{kk'}^{[l](G)})$$

$$J_{style}(S, G) = \sum_l \lambda^{[l]} J_{style}^{[l]}(S, G)$$

$$\underline{J(G)} = \alpha J_{content}(C, G) + \beta J_{style}(S, G)$$



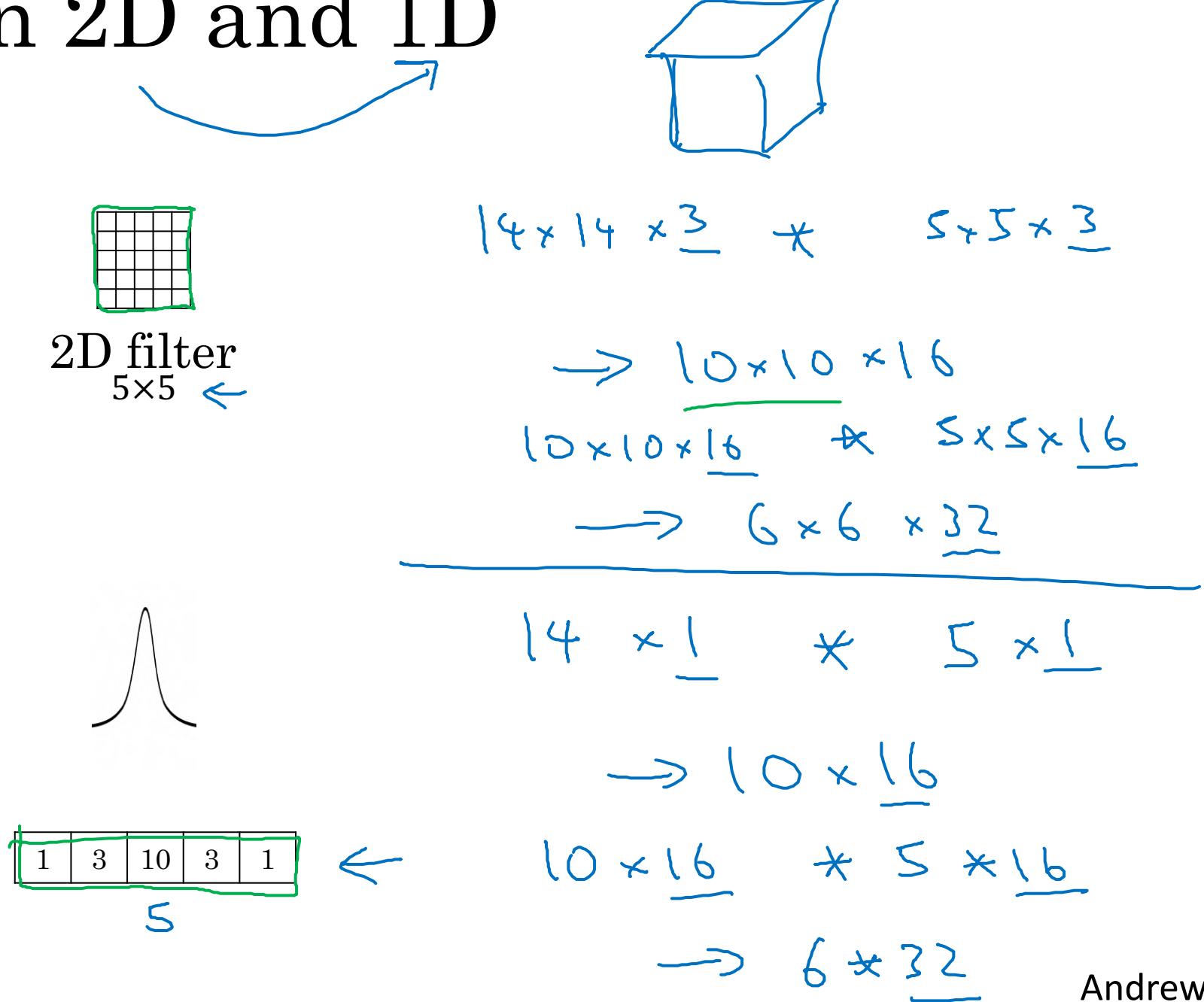
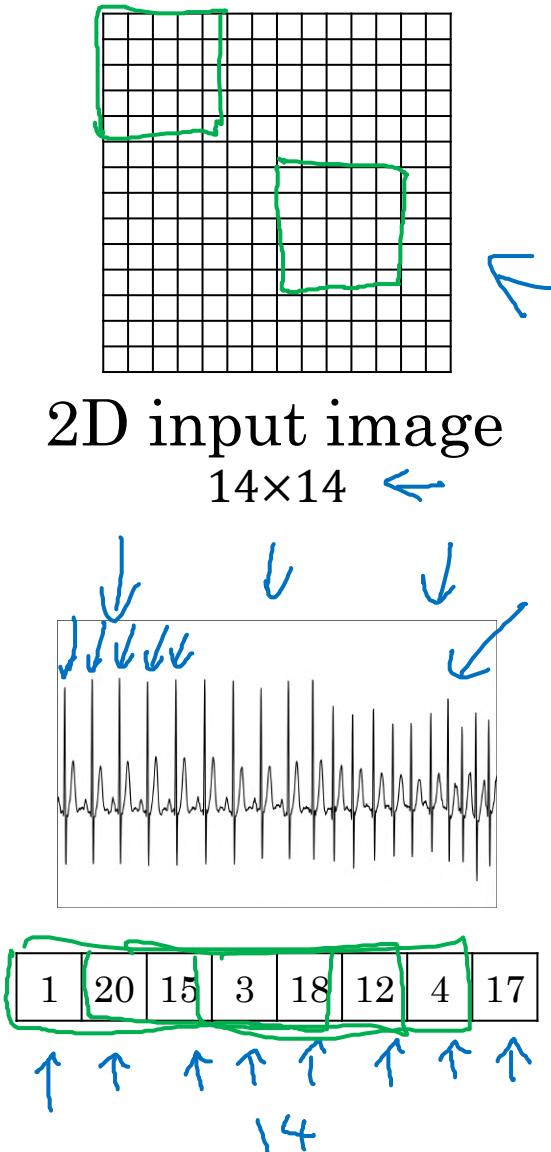
deeplearning.ai

# Convolutional Networks in 1D or 3D

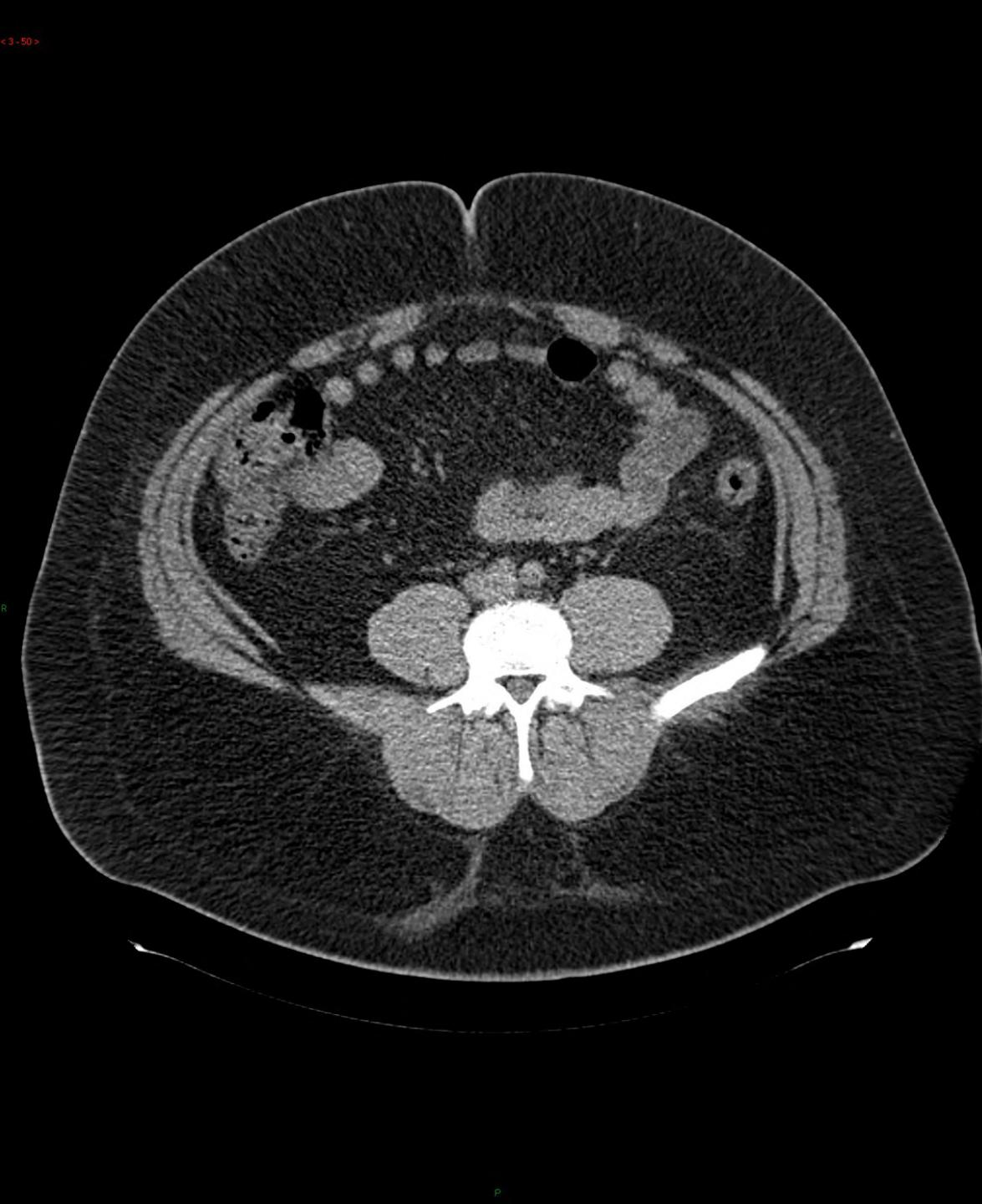
---

1D and 3D  
generalizations of  
models

# Convolutions in 2D and 1D



# 3D data



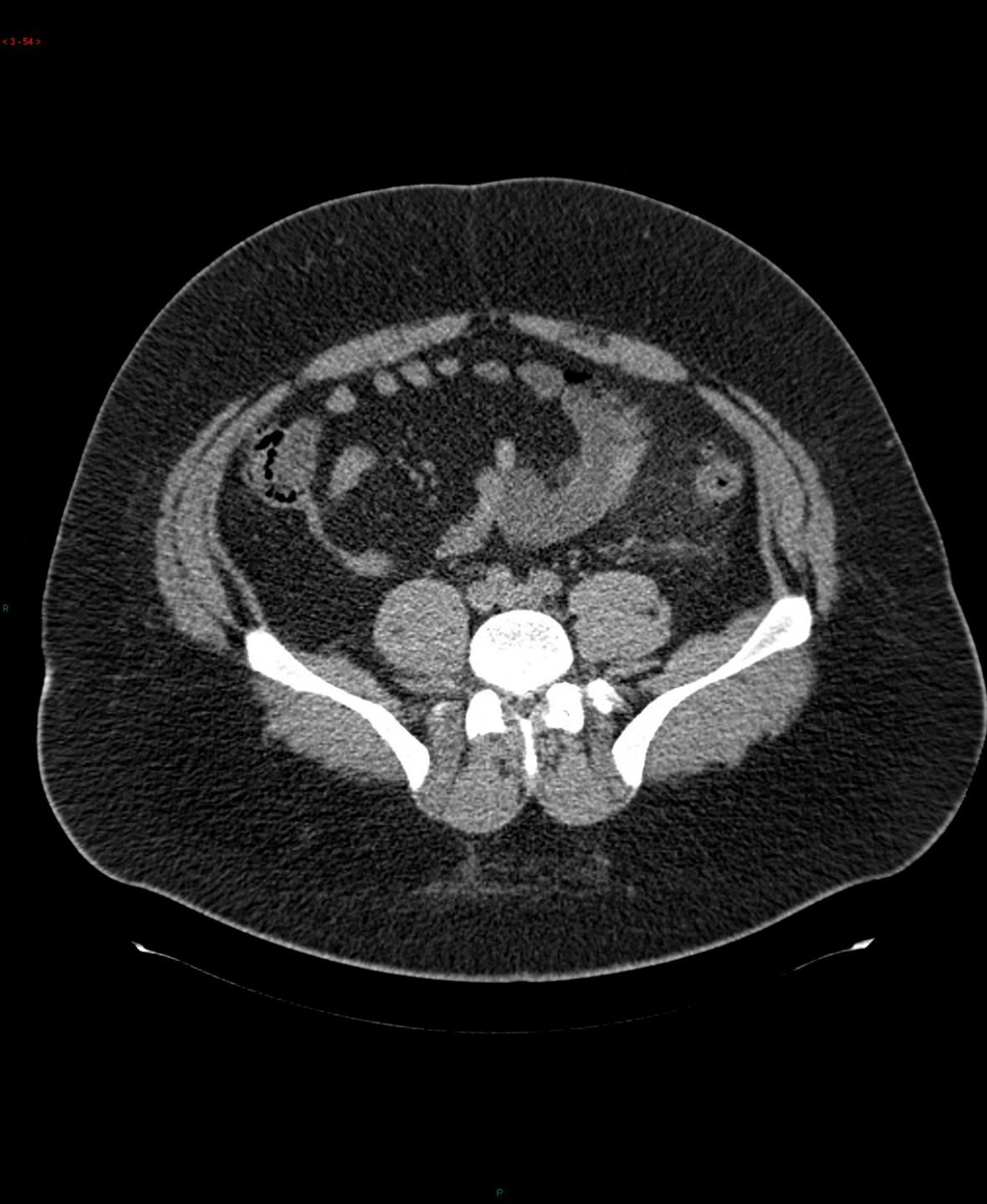
Andrew Ng

# 3D data

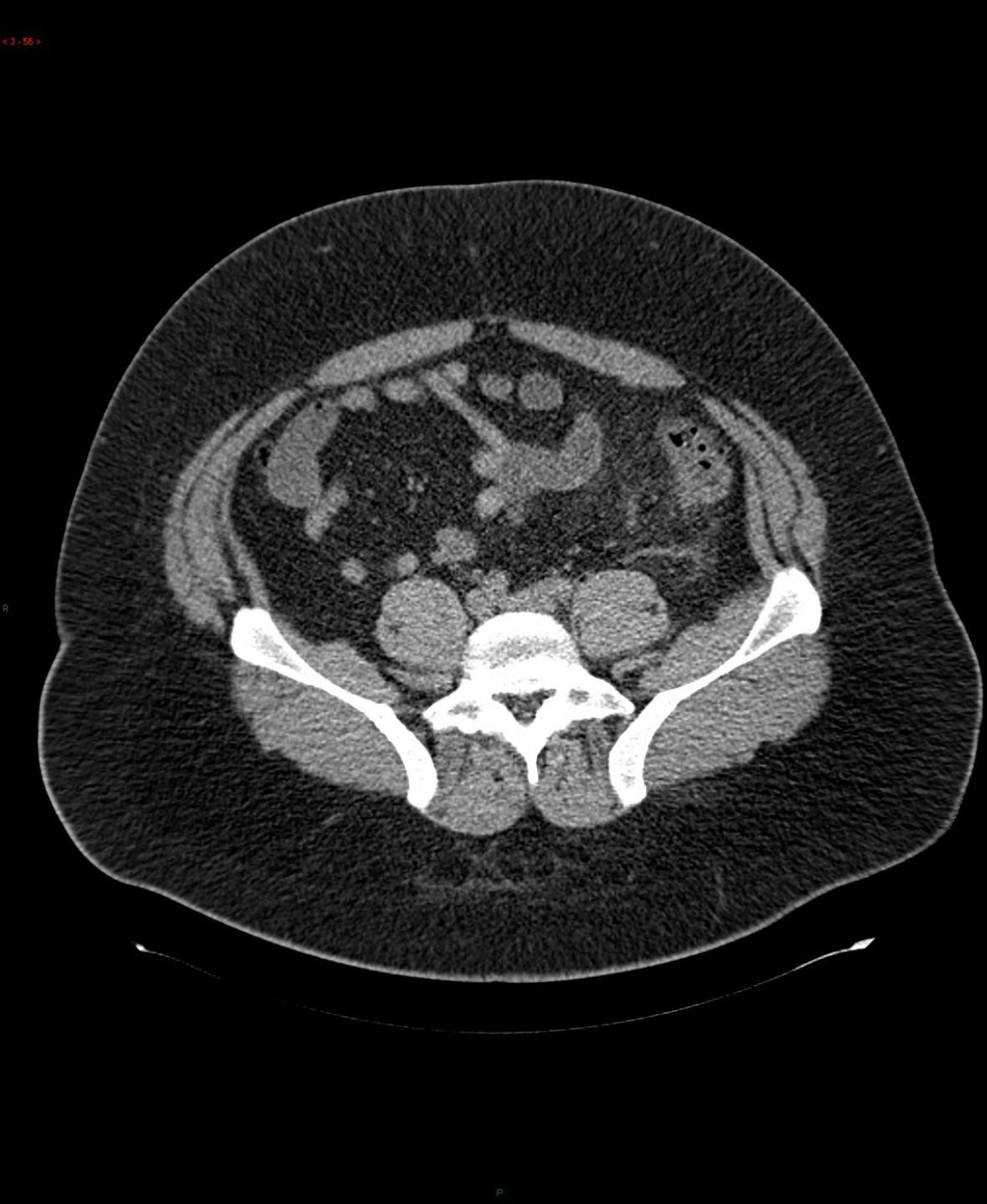


Andrew Ng

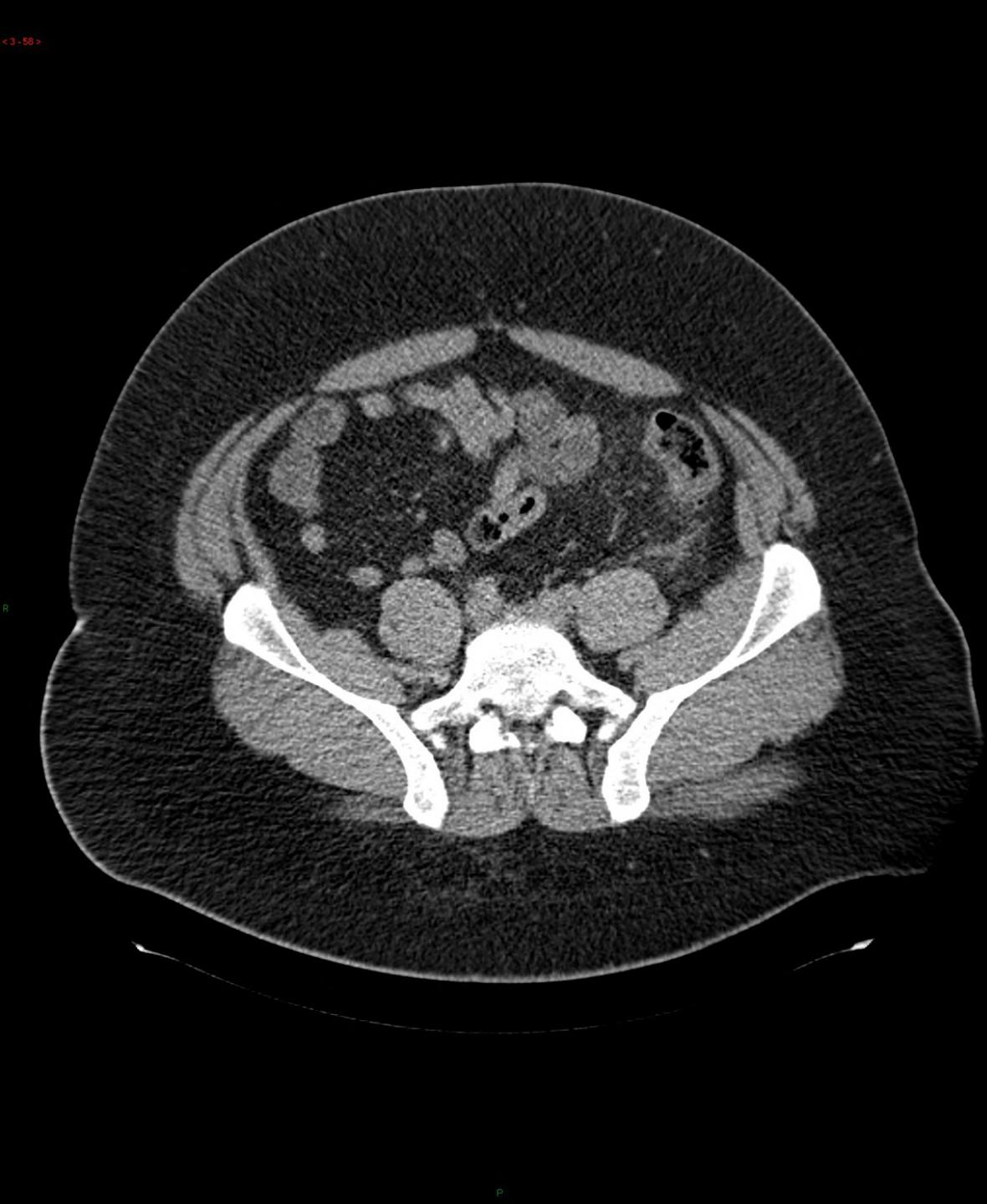
# 3D data



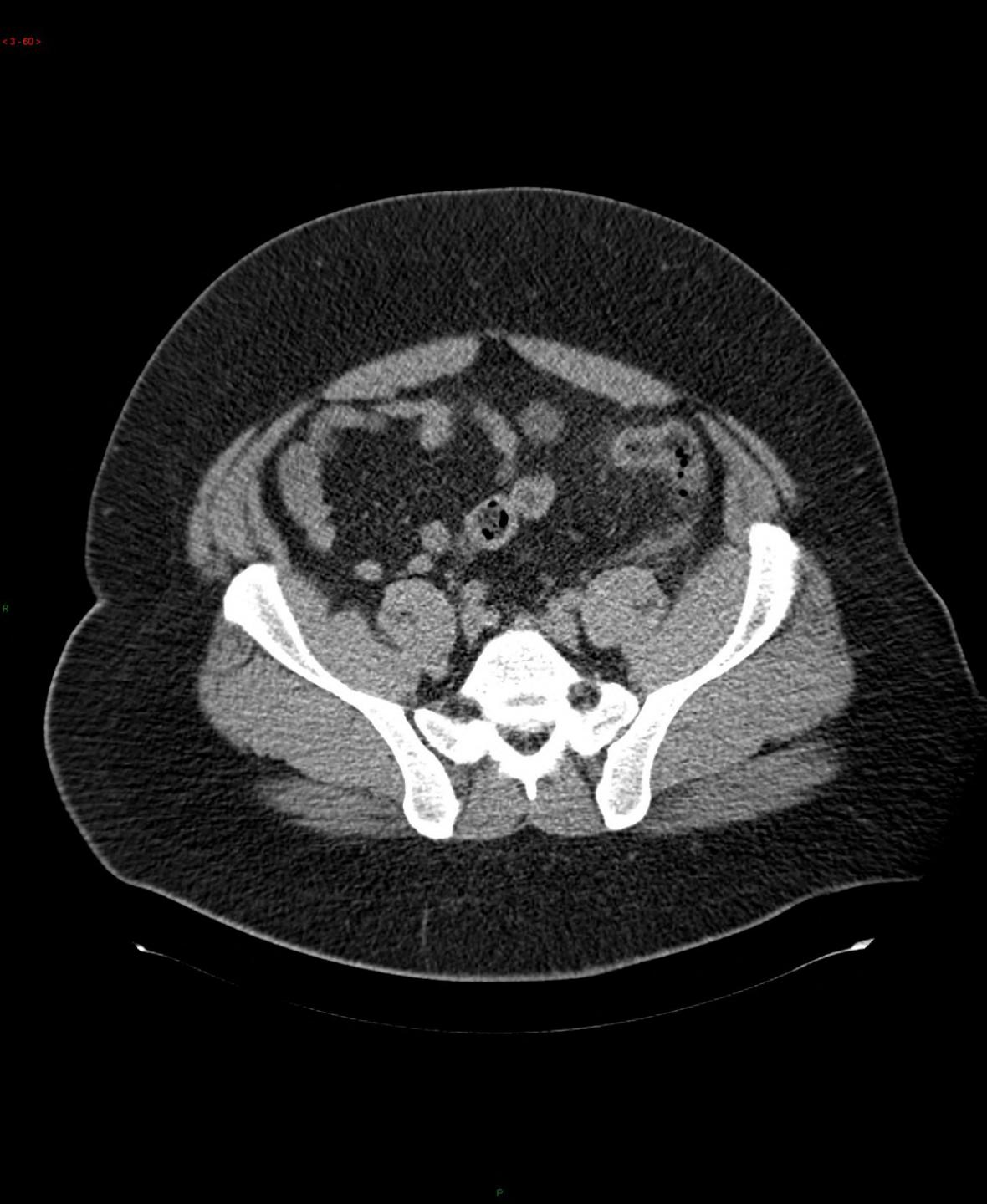
# 3D data



# 3D data



# 3D data



# 3D data



# 3D data



Andrew Ng

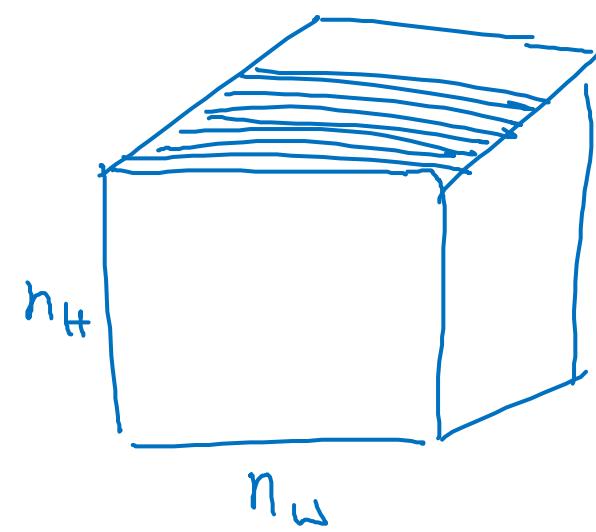
# 3D data



# 3D data

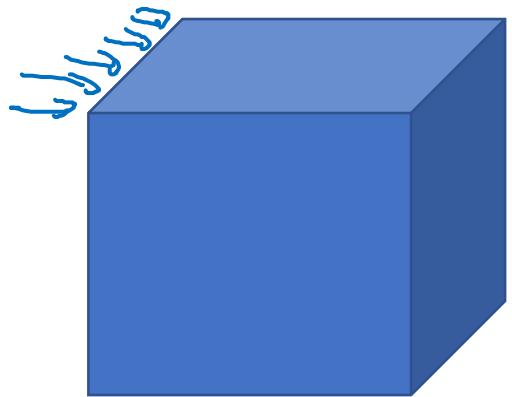


# 3D data

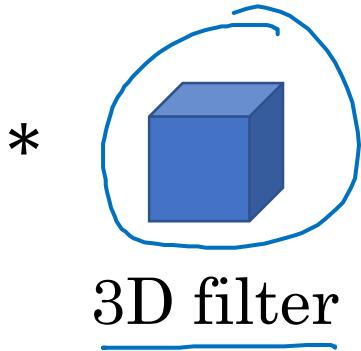


Andrew Ng

# 3D convolution



3D volume



$$\begin{array}{c} \downarrow \quad \downarrow \quad \downarrow \quad \downarrow \\ \cancel{4 \times 4 \times 4} \times \cancel{1} \quad n_c \\ \times \cancel{5 \times 5 \times 5} \times \cancel{1} \quad 16 \text{ filters.} \\ \rightarrow 10 \times 10 \times 10 \times \cancel{16} \\ \times \cancel{5 \times 5 \times 5} \times \cancel{16} \\ \rightarrow 6 \times 6 \times 6 \times 32 \end{array}$$

Diagram illustrating the 3D convolution process:

- The input is a 3D volume of size  $4 \times 4 \times 4$ .
- The input is processed by a 3D filter of size  $5 \times 5 \times 5$ .
- The output is a tensor of size  $10 \times 10 \times 10 \times 16$ , representing 16 feature maps.
- The final output is a tensor of size  $6 \times 6 \times 6 \times 32$ , representing 32 feature maps.