deeplearning.ai

Convolutional Neural Networks

Computer vision

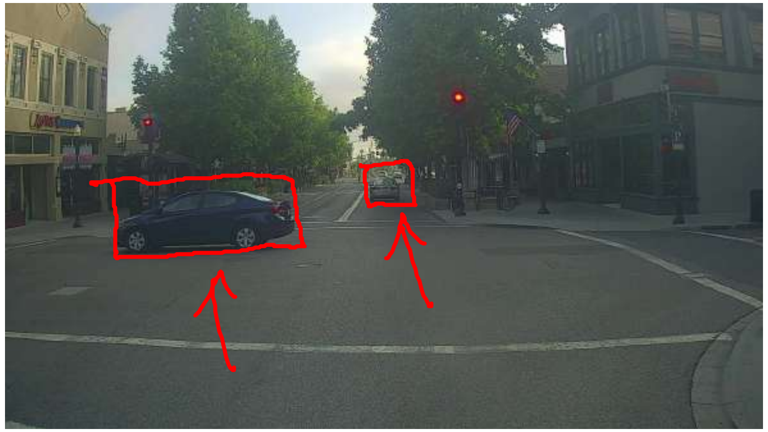
Computer Vision Problems

Image Classification

64x64

Cat? (0/1)

Object detection



Neural Style Transfer

Deep Learning on large images

Cat? (0/1)

64x64 × 3

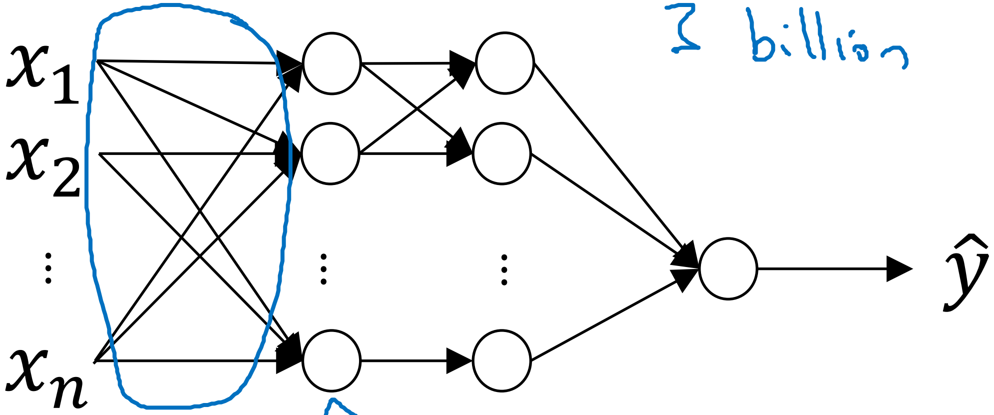
12288

1000×1000×3
= 3 million

$X \in \mathbb{R}^{3M}$

$W^{[1]}$ (1000, 3M)

3 billion



3M

1000

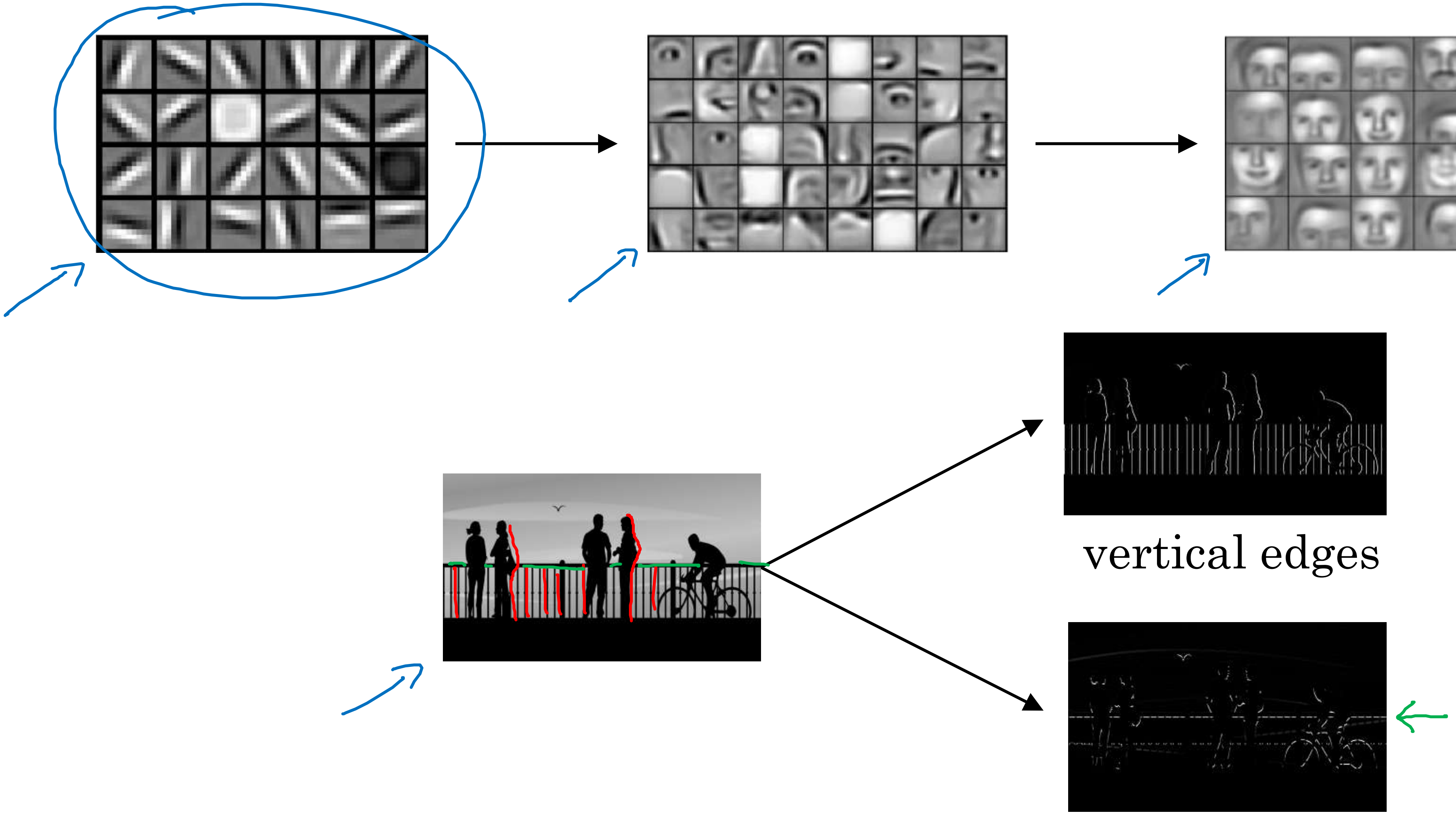
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

Edge detection example

Computer Vision Problem



Andrew Ng

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

"convolution"

*

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

3×3

→ filter
kernel

=

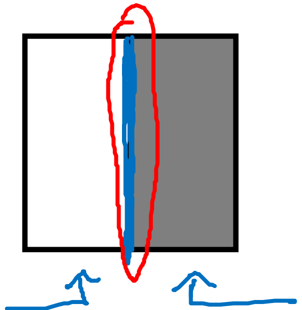
-5	-4	0	8
-10	-2	2	3
0	-2	-4	-7
-3	-2	-3	-16

4×4

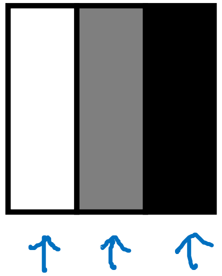
Andrew Ng

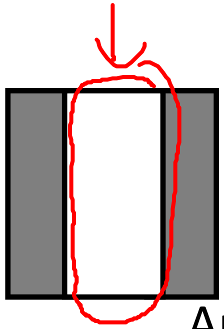
Vertical edge detection

$$\underset{6\times 6}{\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix}} * \underset{3\times 3}{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}} = \underset{4\times 4}{\begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}}$$



*





deeplearning.ai

Convolutional Neural Networks

More edge detection

Vertical edge detection examples

$$\begin{array}{|c|c|c|c|c|c|}\hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline 10 & 10 & 10 & 0 & 0 & 0 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \boxed{\begin{array}{|c|c|c|c|}\hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline 0 & 30 & 30 & 0 \\ \hline \end{array}}$$

$$\begin{array}{|c|c|c|c|c|c|}\hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline 0 & 0 & 0 & 10 & 10 & 10 \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} = \boxed{\begin{array}{|c|c|c|c|}\hline 0 & -30 & -30 & 0 \\ \hline 0 & -30 & -30 & 0 \\ \hline 0 & -30 & -30 & 0 \\ \hline 0 & -30 & -30 & 0 \\ \hline \end{array}}$$

Andrew Ng

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

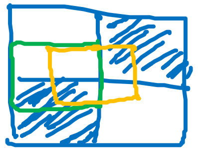
6 x 6

*

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



Andrew Ng

Learning to detect edges

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

$$\begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$$

Sobel filter

$$\begin{bmatrix} 3 & 0 & -3 \\ 10 & 0 & -10 \\ 3 & 0 & -3 \end{bmatrix}$$

Scharr filter

$$\begin{bmatrix} 3 & 0 & 1 & 2 & 7 & 4 \\ 1 & 5 & 8 & 9 & 3 & 1 \\ 2 & 7 & 2 & 5 & 1 & 3 \\ 0 & 1 & 3 & 1 & 7 & 8 \\ 4 & 2 & 1 & 6 & 2 & 8 \\ 2 & 4 & 5 & 2 & 3 & 9 \end{bmatrix} * \underbrace{\begin{bmatrix} w_1 & w_2 & w_3 \\ w_4 & w_5 & w_6 \\ w_7 & w_8 & w_9 \end{bmatrix}}_{3\times 3} =$$

convolution

45°
70°
73°

Andrew Ng

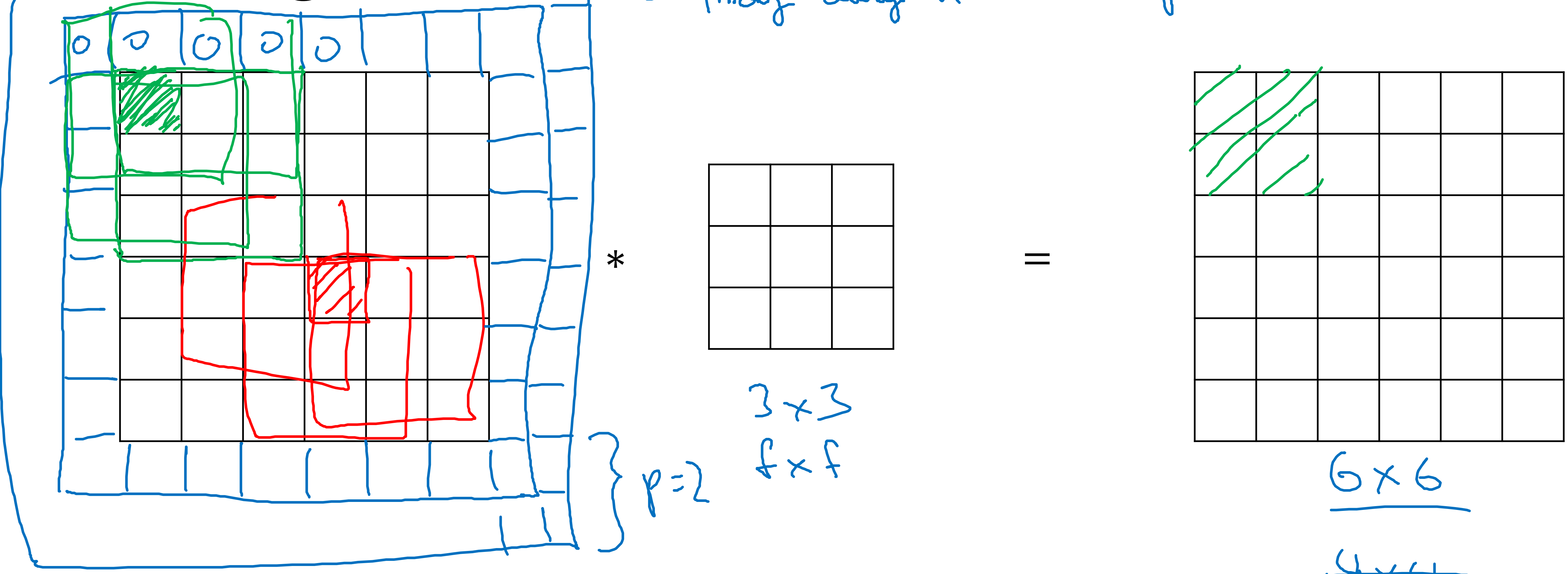
deeplearning.ai

Convolutional Neural Networks

Padding

Padding

- shrinky output
- throw away info from edge



6x6 → 8x8

n x n

n-f+1 x n-f+1

6-3+1=4

p = padding = 1

n+2p-f+1 x n+2p-f+1

6+2-3+1 x ___ = 6x6

Andrew Ng

Valid and Same convolutions

→ no padding

"Valid": $n \times n \; * \; f \times f \rightarrow n-f+1 \times n-f+1$

$6 \times 6 \; * \; 3 \times 3 \rightarrow 4 \times 4$

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

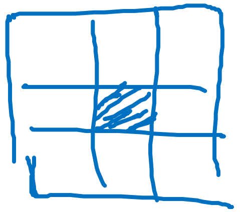
$n+2p-f+1 \times n+2p-f+1$

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

$3 \times 3 \quad p = \frac{3-1}{2} = 1 \quad | \quad 5 \times 5 \;\; f=5 \quad p=2$

f is usually odd

1×1
3×3
5×5
7×7



Andrew Ng

deeplearning.ai

Convolutional Neural Networks

Strided convolutions

Strided convolution

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

7×7

$*$

3	4	4
1	0	2
-1	0	3

3×3

stride $= 2$

$=$

91	100	83
69	91	127
44	72	74

3×3

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

$n \times n \quad * \quad f \times f$

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 $\qquad f \times f$ filter

padding p $\qquad$ stride s

Output size:

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

Andrew Ng

Technical note on cross-correlation vs. convolution

Convolution in math textbook:

2^{7}	3^{2}	7^{5}	4	6	2
6^{9}	6^{0}	9^{4}	8	7	4
3^{-1}	4^{1}	8^{3}	3	8	9
7	8	3	6	6	3
4	2	1	8	3	4
3	2	4	1	9	8

*

3	4	5
1	0	2
-1	9	7

=

7	2	5
9	0	4
-1	1	3

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

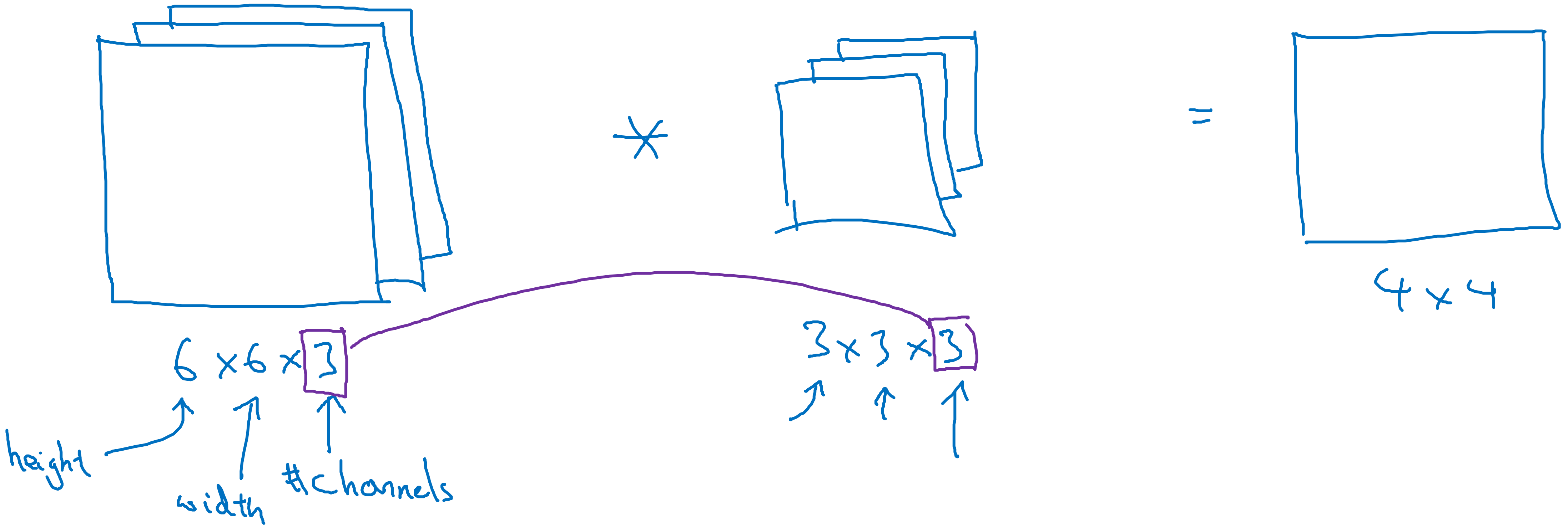
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

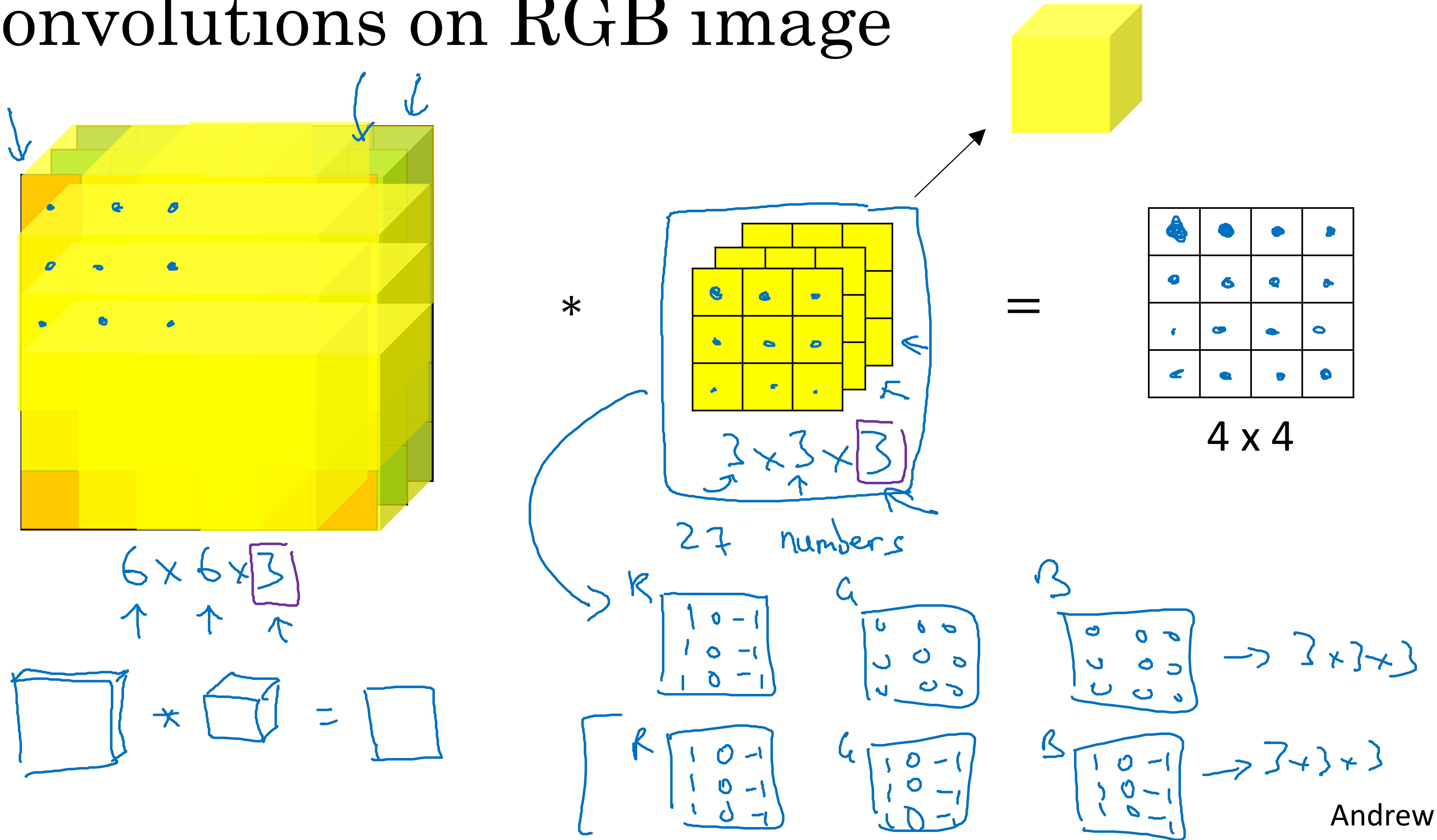
Convolutions over volumes

Convolutions on RGB images

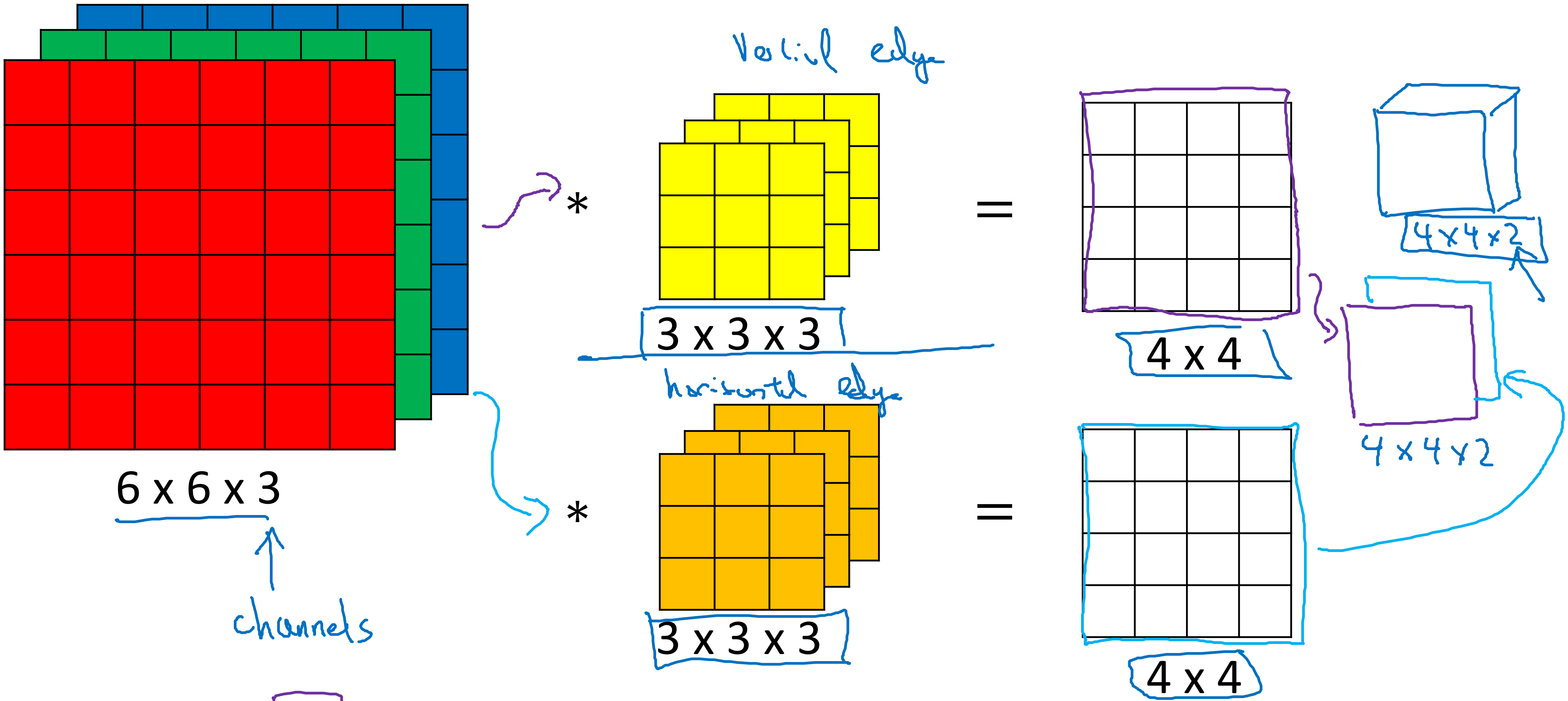


Andrew Ng

Convolutions on RGB image



Multiple filters



Summary: $n \times n \times \boxed{n_c} \ * \ f \times f \times \boxed{n_c} \rightarrow \underline{n-f+1} \times \underline{n-f+1} \times n_c'$

$6 \times 6 \times 3 \qquad 3 \times 3 \times 3 \qquad 4 \times 4 \times 2$ ← # filters

Andrew Ng

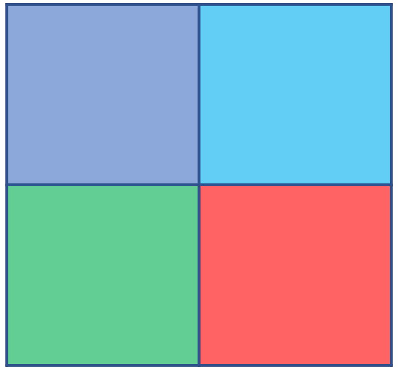
deeplearning.ai

Convolutional Neural Networks

Pooling layers

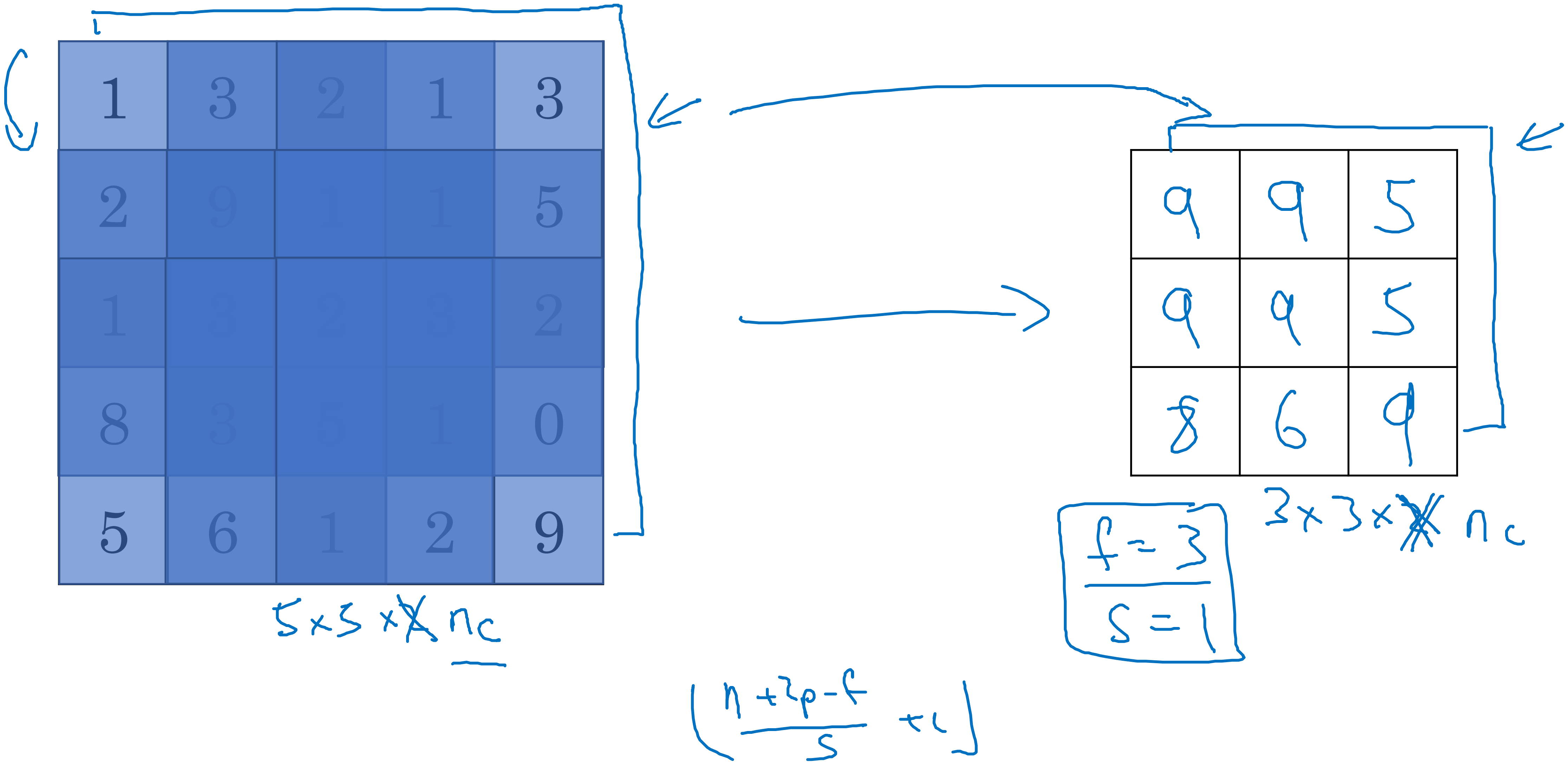
Pooling layer: Max pooling

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



Andrew Ng

Pooling layer: Max pooling



Andrew Ng

Pooling layer: Average pooling

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

$\rightarrow$

3.75	1.25
4	2

$f = 2$

$s = 2$

$\underline{7 \times 7} \times 1000 \longrightarrow 1 \times 1 \times 1000$

Andrew Ng

Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

$f=2, s=2$

$f=3, s=2$

→ ~~p: padding~~

No parameters to learn!

$n_H \times n_W \times n_C$

↓

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

$\times n_C$

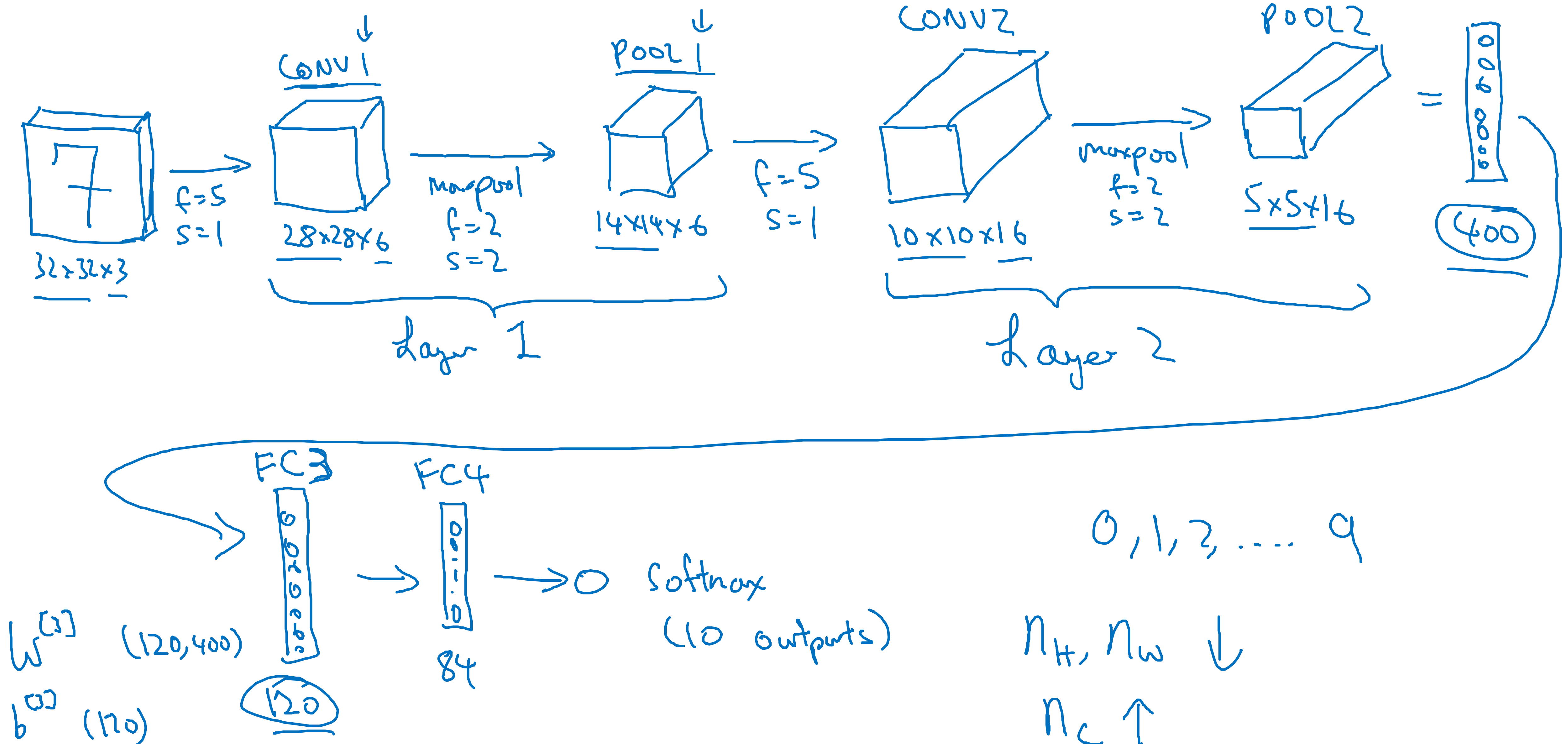
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

Convolutional neural network example

Neural network example (LeNet-5)



CONV - POOL - CONV - POOL - FC - FC - FC - SOFTMAX

Andrew Ng

Neural network example

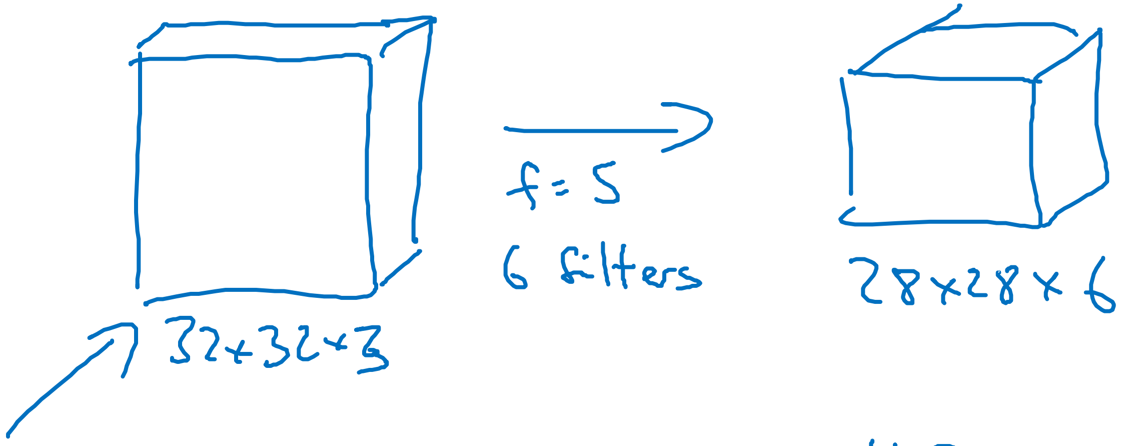
	Activation shape	Activation Size	# parameters
Input:	(32,32,3)	3,072 $a^{[0]}$	0

deeplearning.ai

Convolutional Neural Networks

Why convolutions?

Why convolutions



3,072 4,704

$5 \times 5 = 25$

26

$6 \times 26 = 156$ parameters

$3{,}072 \times 4{,}704 \approx \underline{14M}$

Andrew Ng

Why convolutions

translation invariance

$$\begin{bmatrix} 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \\ 10 & 10 & 10 & 0 & 0 & 0 \end{bmatrix} * \underbrace{\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}}_{3 \times 3} = \begin{bmatrix} 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \\ 0 & 30 & 30 & 0 \end{bmatrix}$$

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.

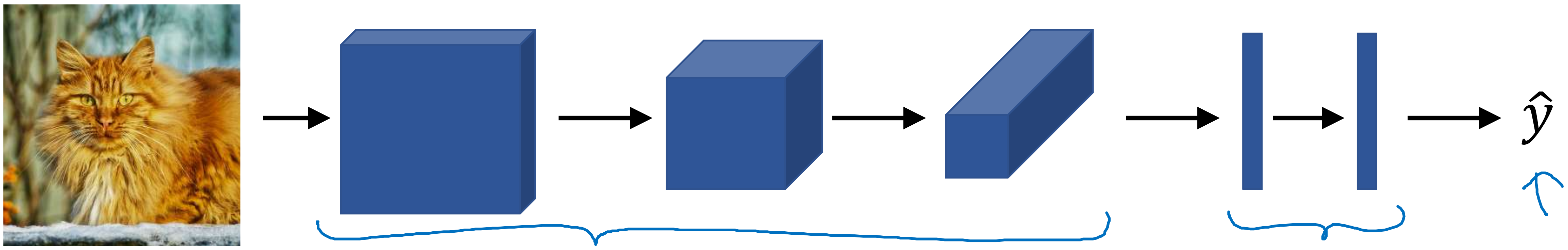
Sparsity of connections: In each layer, each output value depends only on a small number of inputs.

Andrew Ng

Putting it together

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

$\boxed{w, b}$



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)

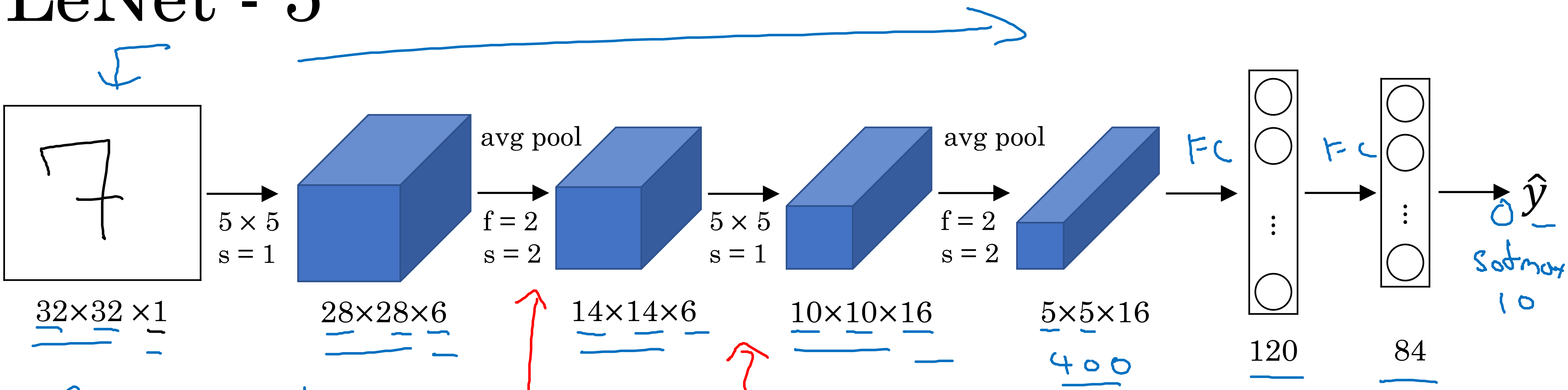
Inception

deeplearning.ai

Case Studies

Classic networks

LeNet - 5



60K parameters

$n_H, n_W \downarrow$ $n_C \uparrow$

non-linearity after pooling

$n_H \times n_W \times n_C$ $f \times f \times n_C$

conv pool conv pool fc fc output

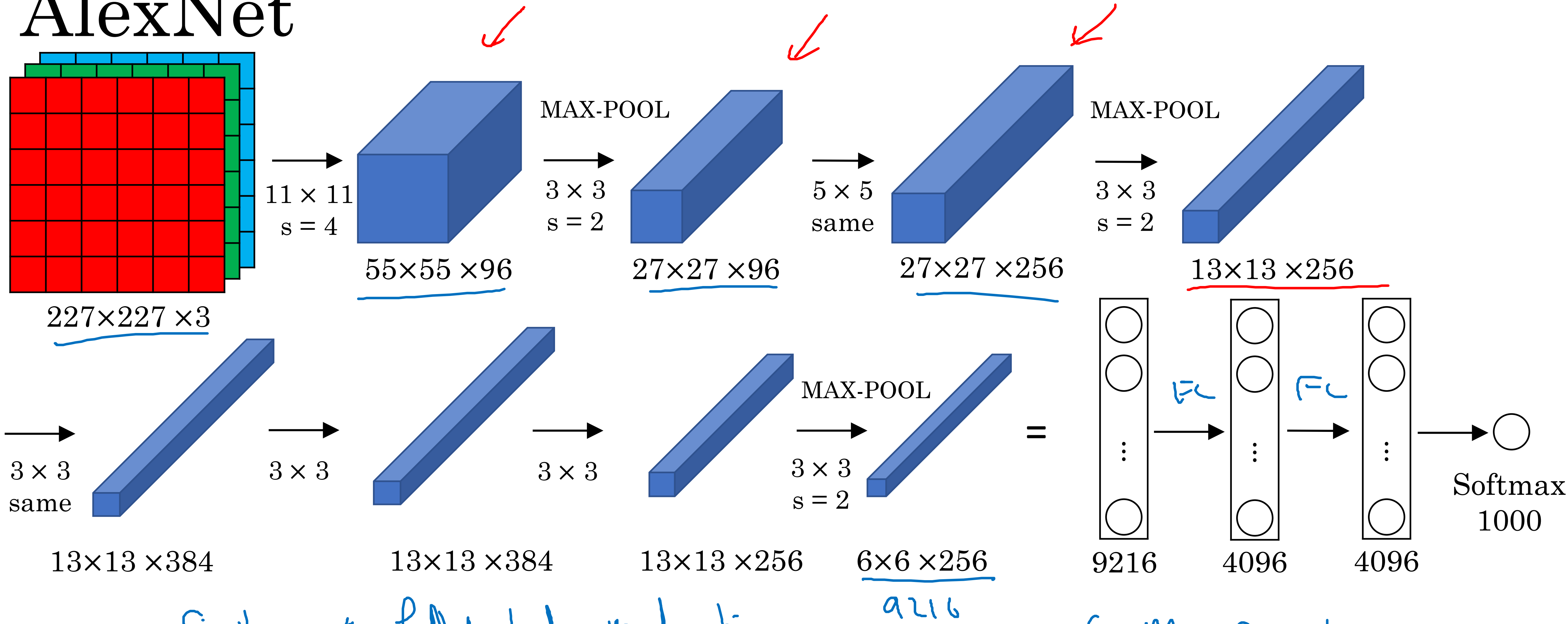
Advanced: sigmoid/tanh ReLU

II III.

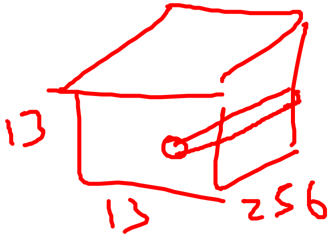
[LeCun et al., 1998. Gradient-based learning applied to document recognition]

Andrew Ng

AlexNet



- Similar to LeNet, but much bigger. ~60M parameters
- ReLU
- Multiple GPUs.
- Local Response Normalization (LRN)



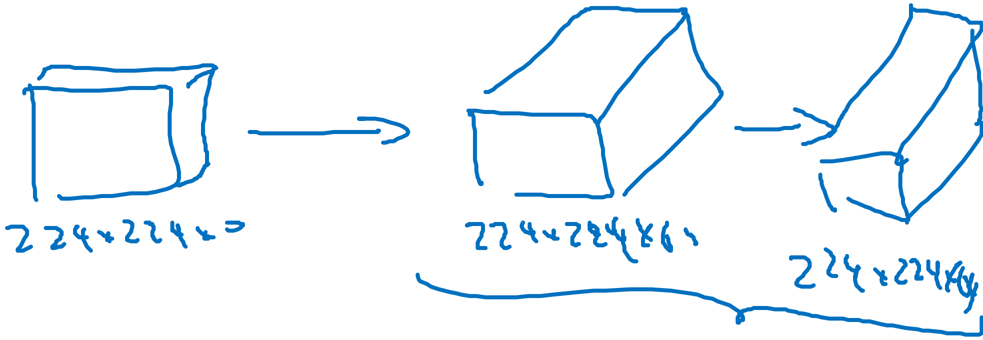
[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

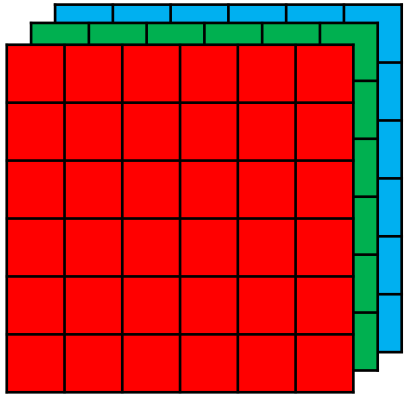
VGG - 16

VGG-19

CONV = 3×3 filter, s = 1, same

MAX-POOL = 2×2 , s = 2





224×224 ×3

→ [CONV 64] ×2 → 224×224×64 → POOL → 112×112 ×64 → [CONV 128] ×2 → 112×112 ×128 → POOL → 56×56 ×128

→ [CONV 256] ×3 → 56×56 ×256 → POOL → 28×28 ×256 → [CONV 512] ×3 → 28×28 ×512 → POOL → 14×14×512

→ [CONV 512] ×3 → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

$n_H, n_W \downarrow$ $n_c \uparrow$

~138M

[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

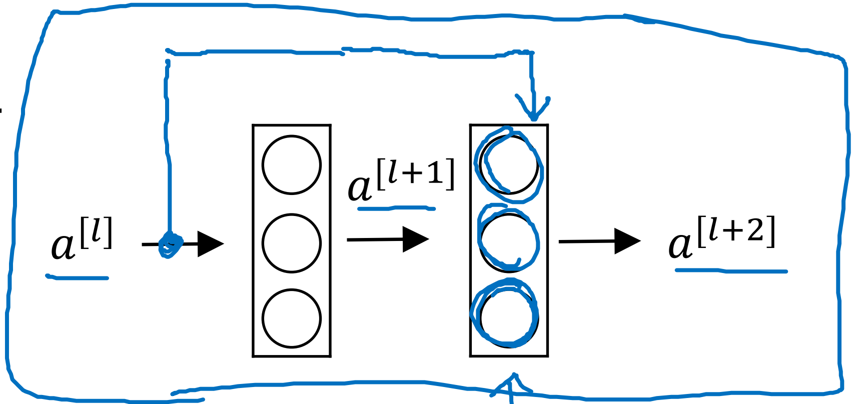
Andrew Ng

deeplearning.ai

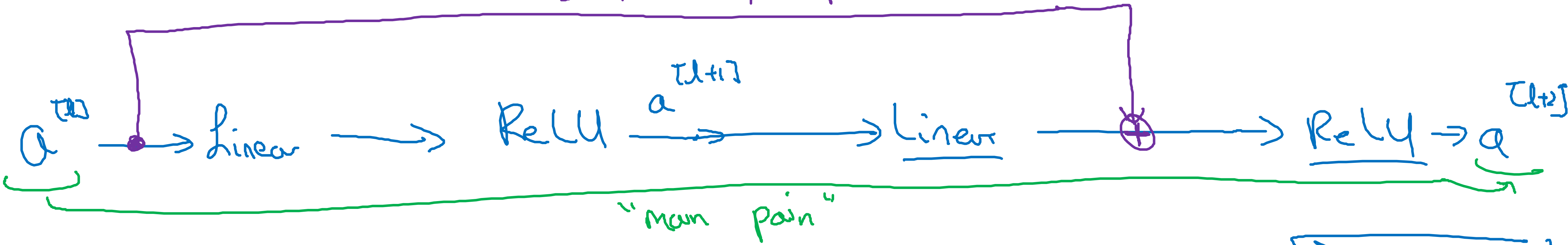
Case Studies

Residual Networks (ResNets)

Residual block



"short cut" / skip connection

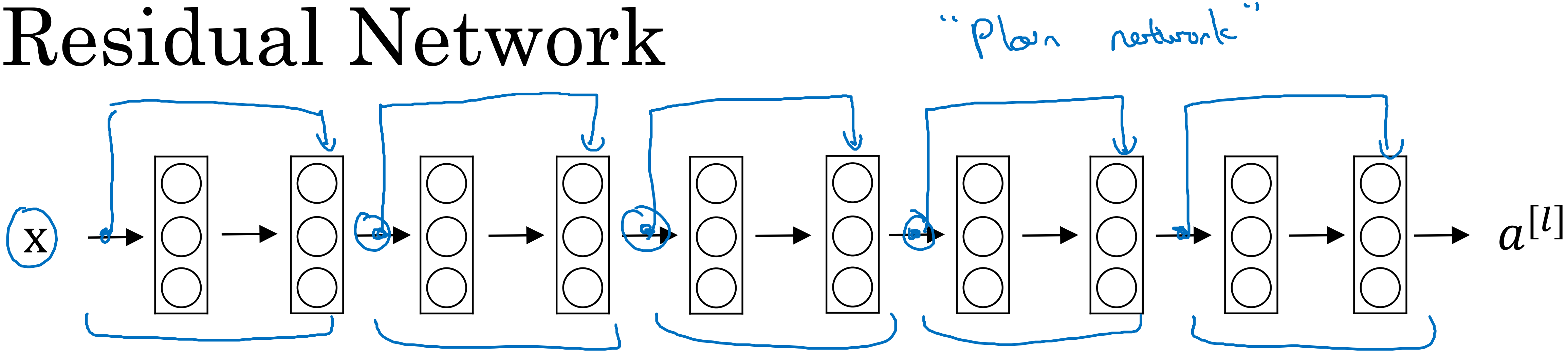


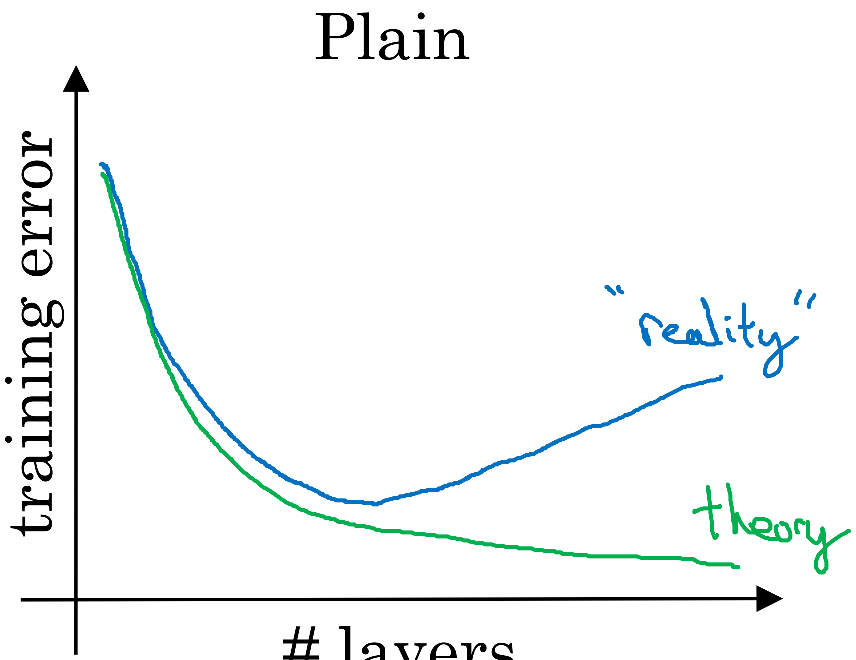
$z^{[l+1]} = W^{[l+1]} a^{[l]} + b^{[l+1]}$ $\quad a^{[l+1]} = g(z^{[l+1]})$ $\quad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$ $\quad \boxed{\cancel{a^{[l+2]} = g(z^{[l+2]})}}$

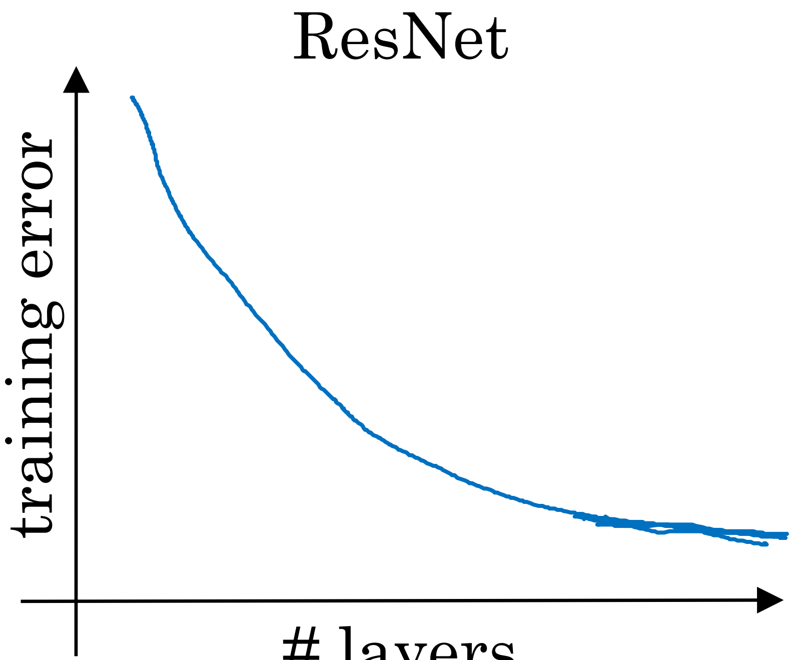
$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

[He et al., 2015. Deep residual networks for image recognition]

Residual Network







[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

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

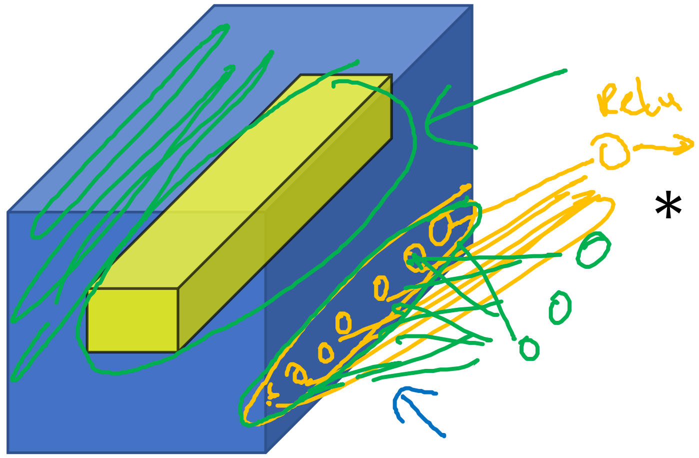
6 × 6 × 1

$*$ $\boxed{2}$ $=$

2	4	6	...		

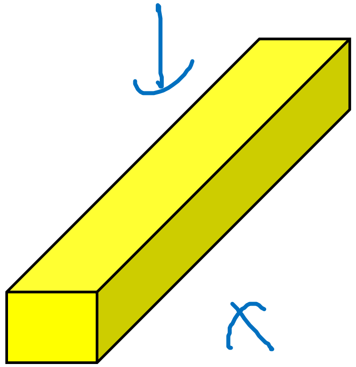
32 → # filters.

$n_c^{[l+1]}$



6 × 6 × 32

$*$

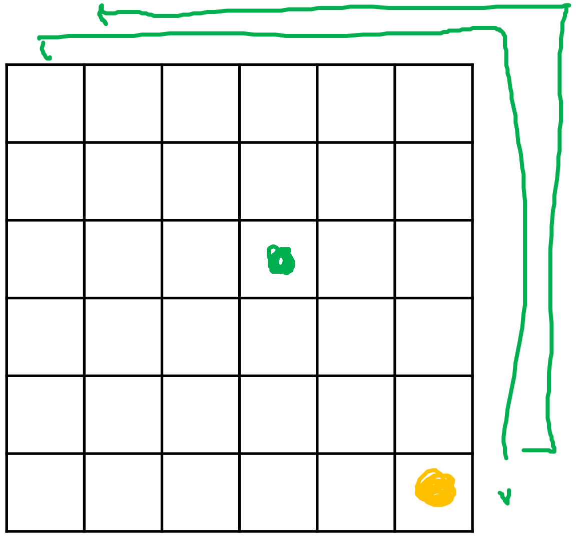


1 × 1 × 32

$=$

ReLU

Network in Network

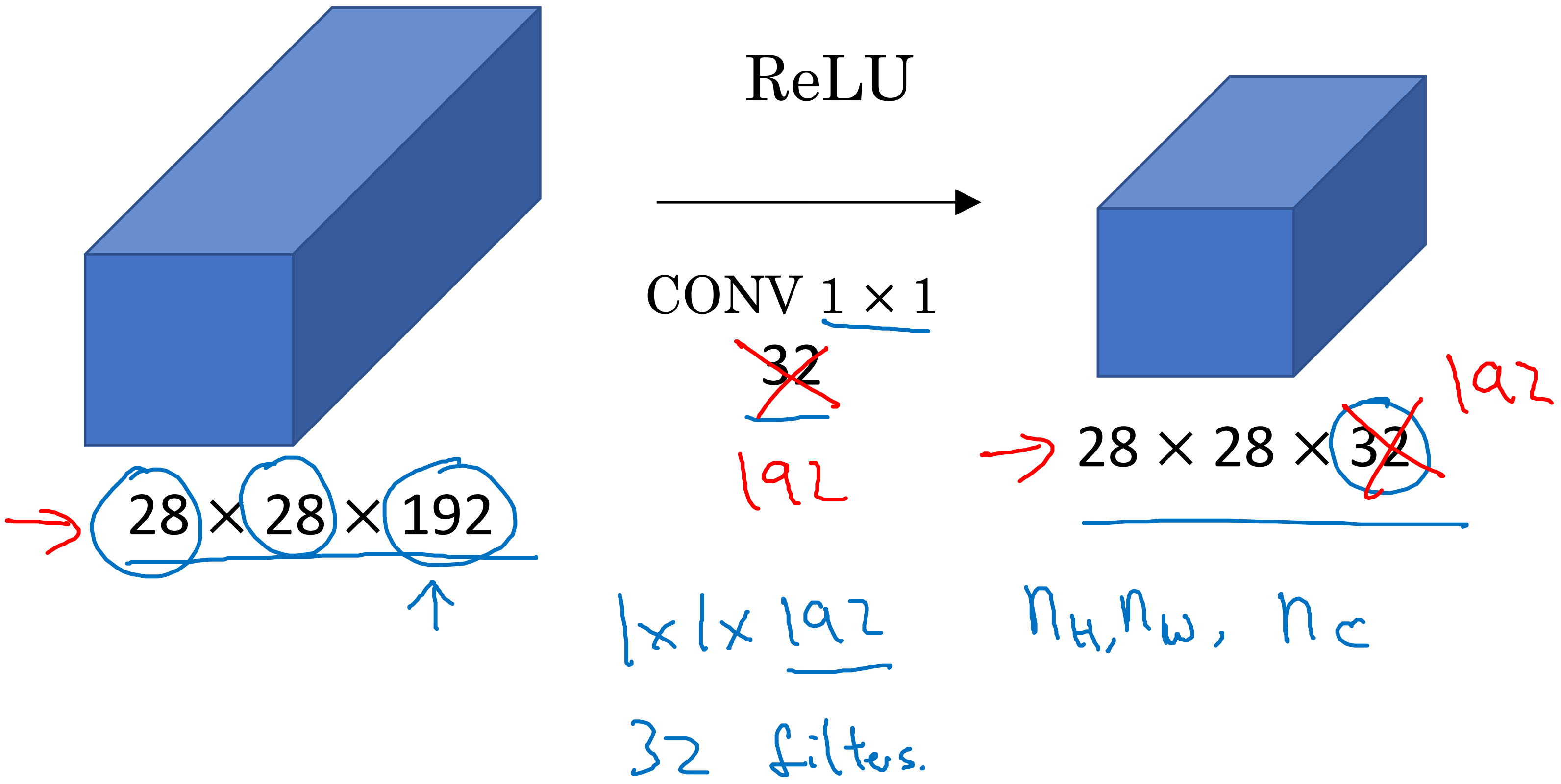


6 × 6 × # filters

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

Andrew Ng

Using 1×1 convolutions



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

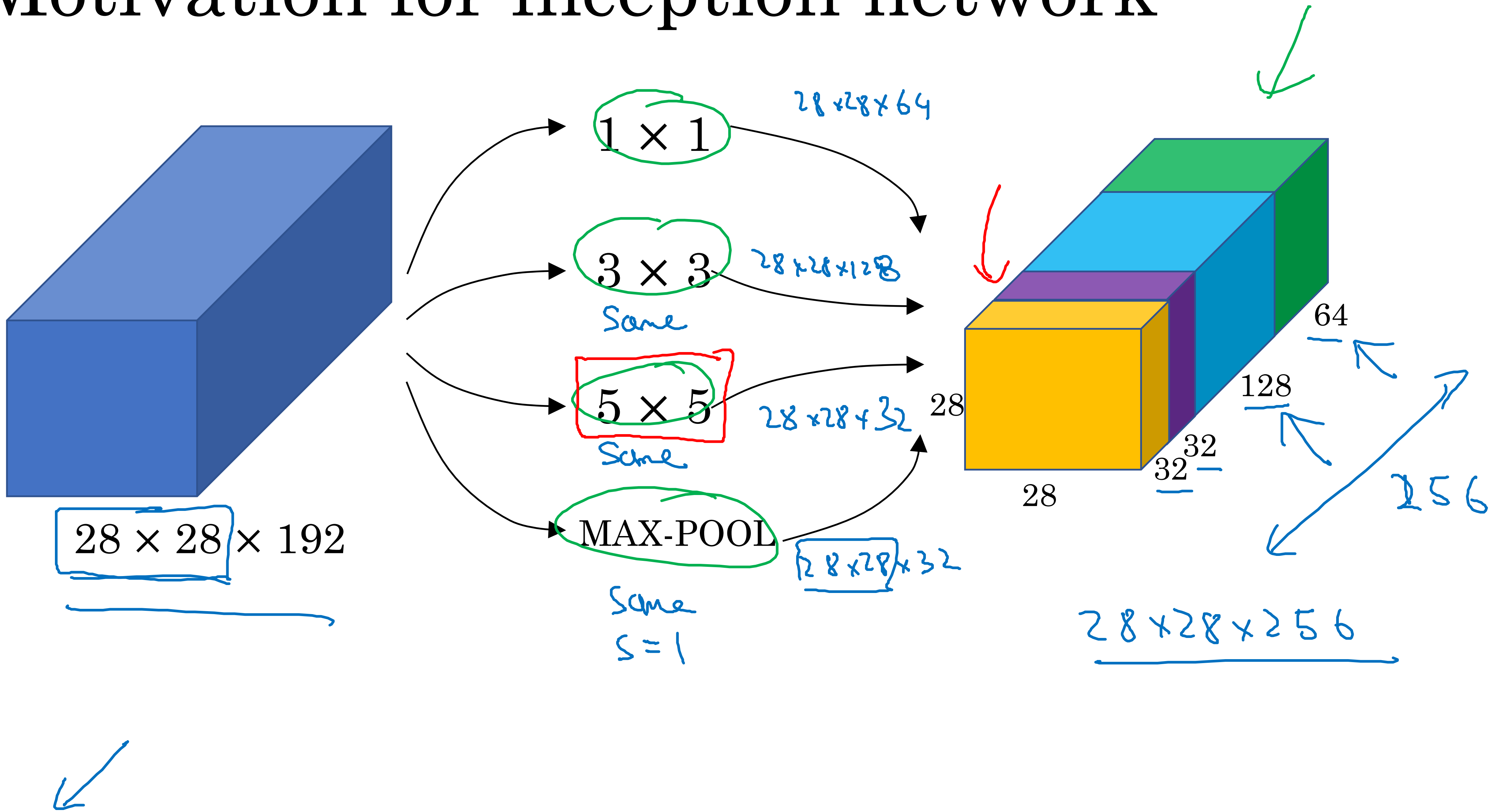
Andrew Ng

deeplearning.ai

Case Studies

Inception network motivation

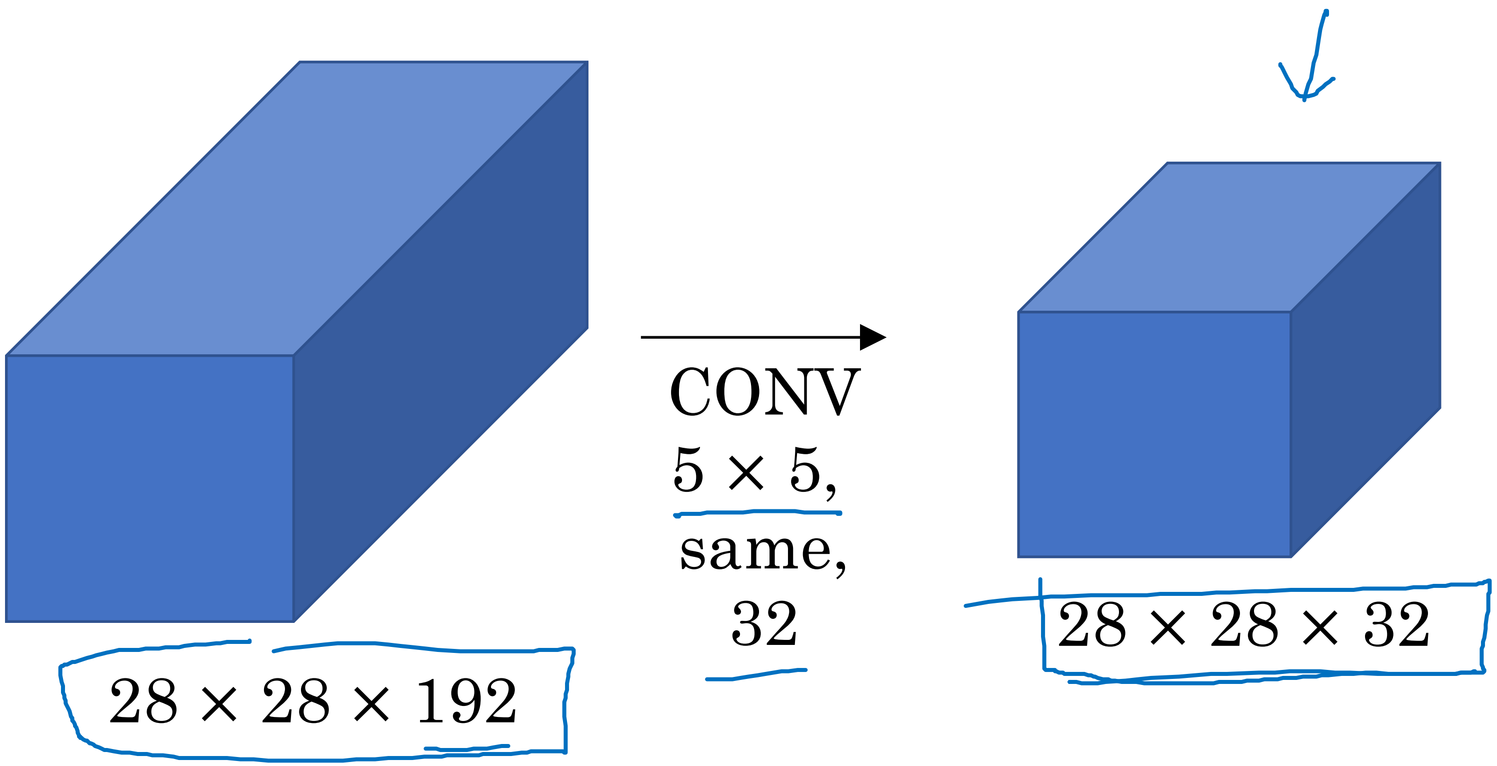
Motivation for inception network



[Szegedy et al. 2014. Going deeper with convolutions]

Andrew Ng

The problem of computational cost



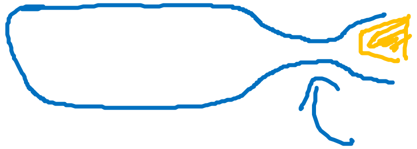
32 filters. filters are 5×5×192.

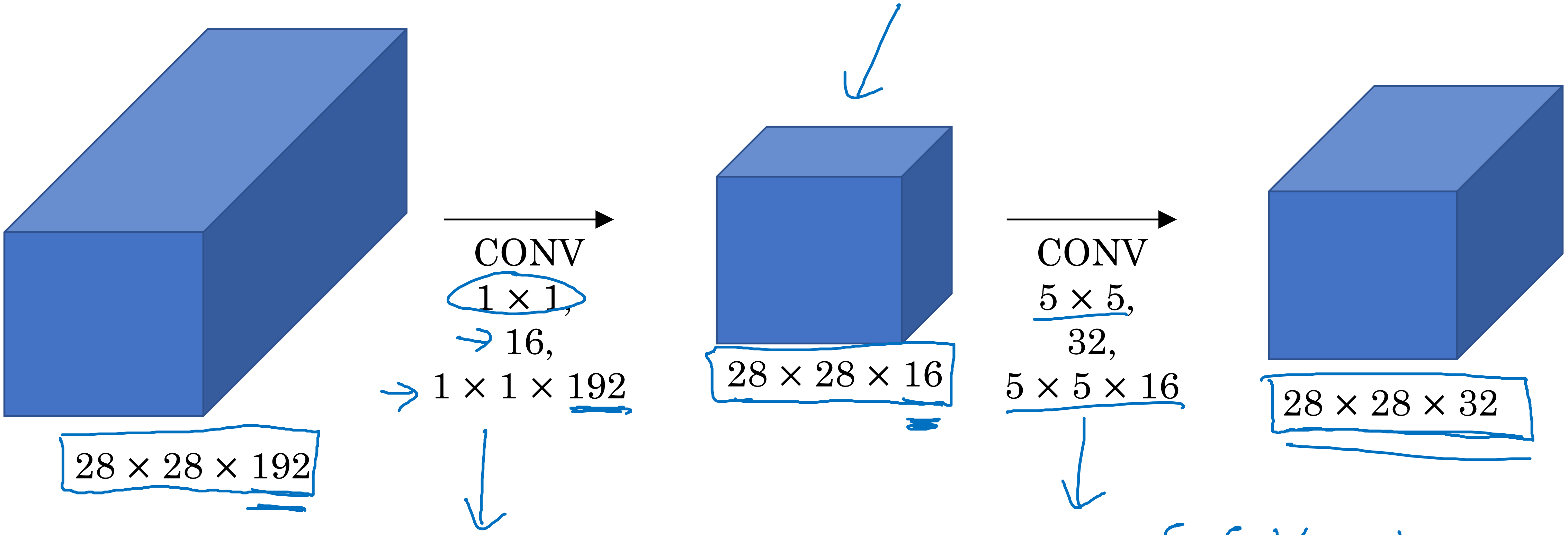
28×28×32 × 5×5×192 = 120M.

Andrew Ng

Using 1×1 convolution

"bottleneck layer"





$28\times28\times16 \times 192 = 2.4M$

$28\times28\times32 \times 5\times5\times16 = 10.0M$

$\underbrace{2.4M + 10.0M}_{12.4M}$

$120M \rightarrow 12.4M$

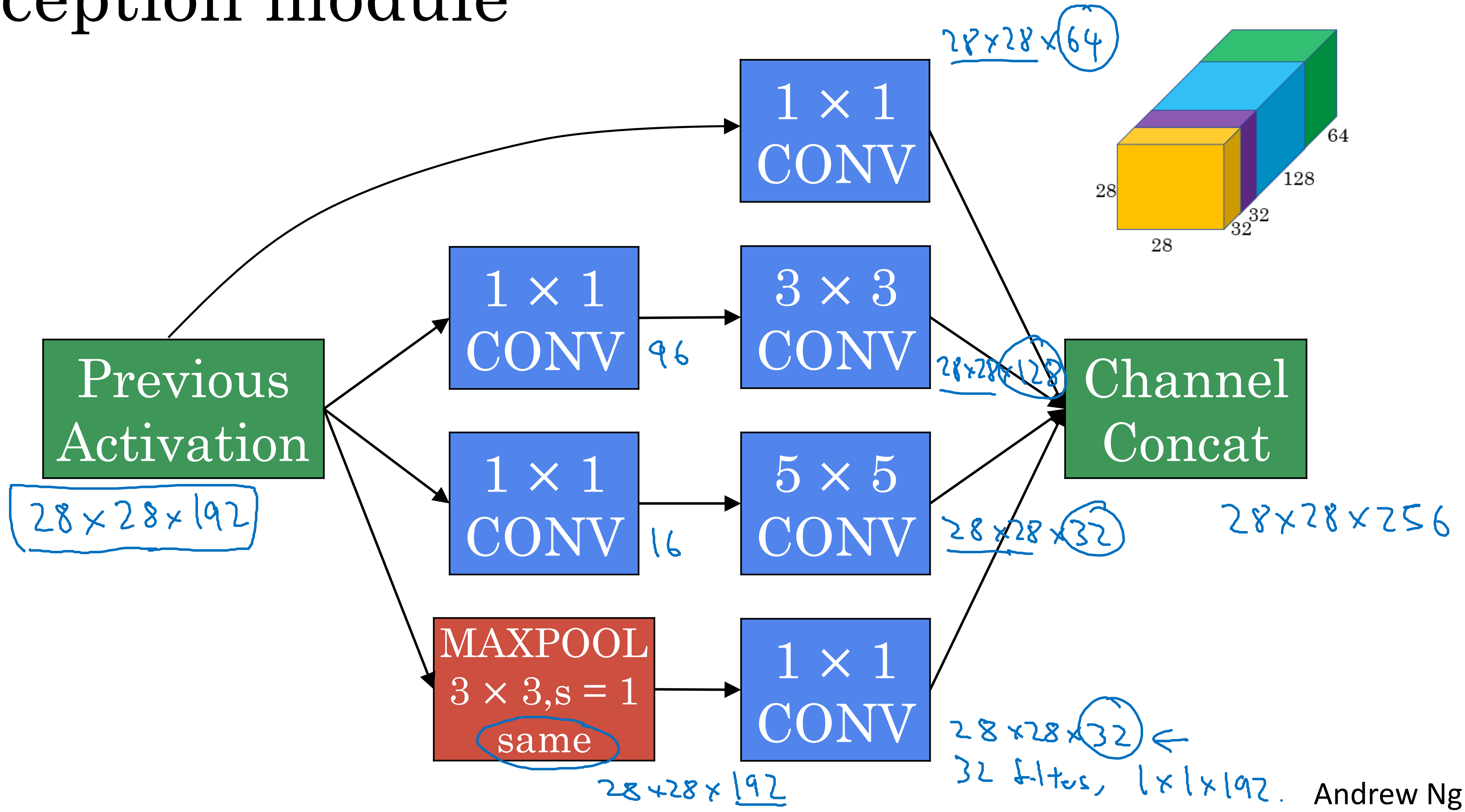
Andrew Ng

deeplearning.ai

Case Studies

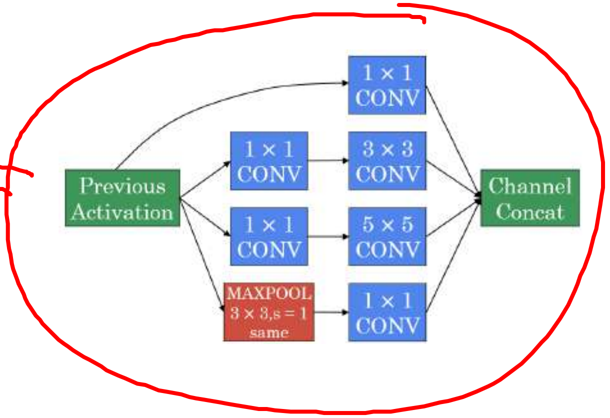
Inception network

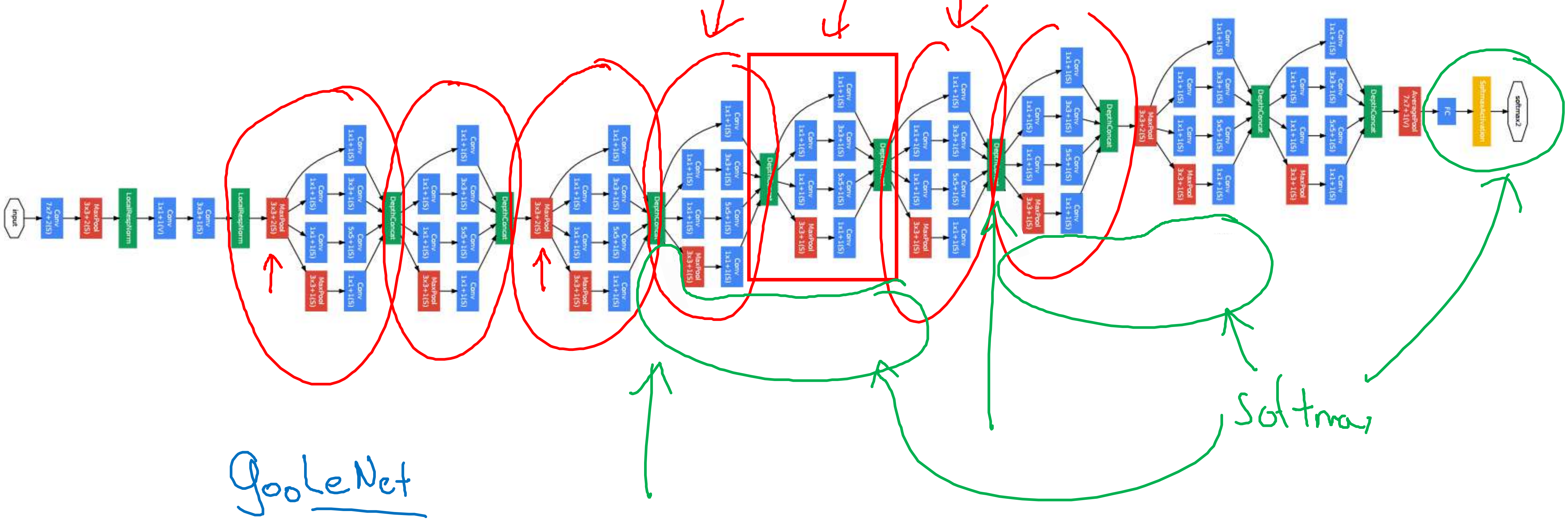
Inception module



Andrew Ng

Inception network





[Szegedy et al., 2014, Going Deeper with Convolutions]

Andrew Ng

http://knowyourmeme.com/memes/we-need-to-go-deeper

deeplearning.ai

Object Detection

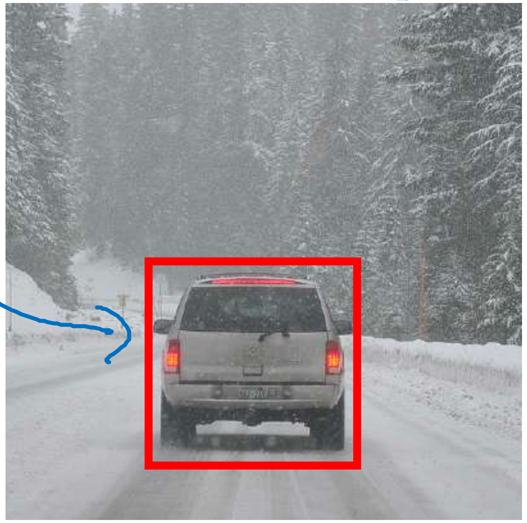
Object localization

What are localization and detection?

Image classification

"Car"

Classification with localization



"Car"

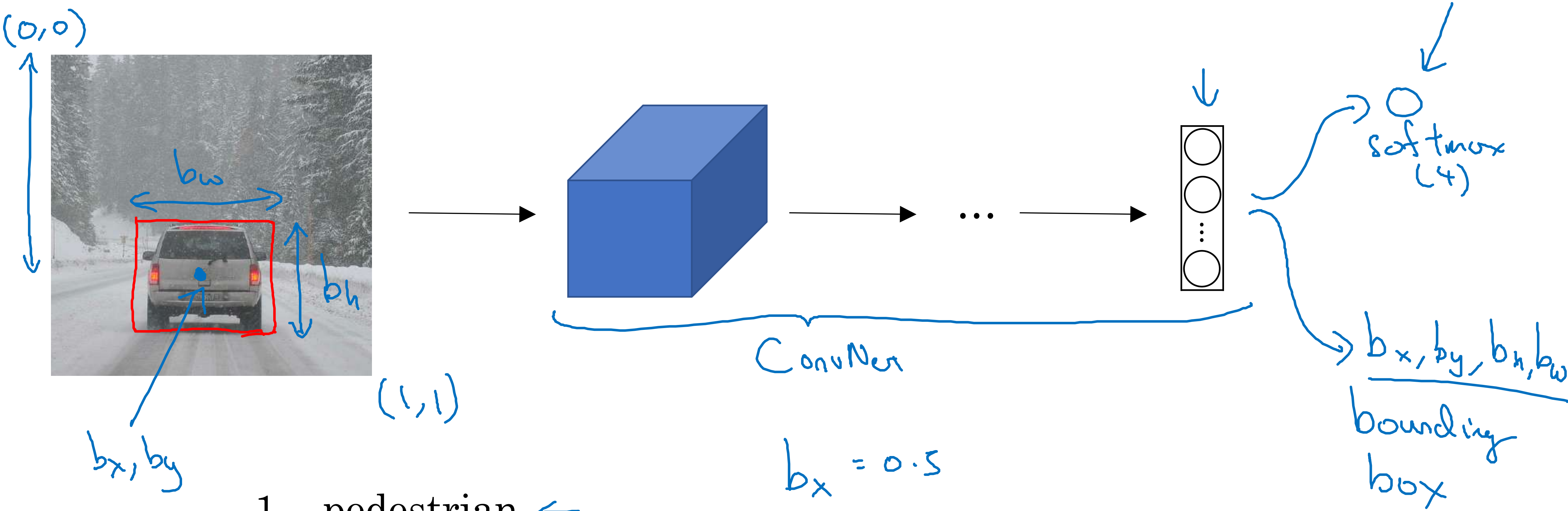
1 object

Detection

multiple objects

Andrew Ng

Classification with localization



1 - pedestrian

2 - car

3 - motorcycle

4 - $\boxed{\text{background}}$

$b_x = 0.5$

$b_y = 0.7$

$b_h = 0.3$

$b_w = 0.4$

Defining the target label y

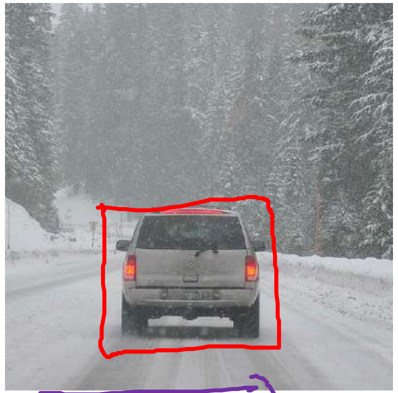
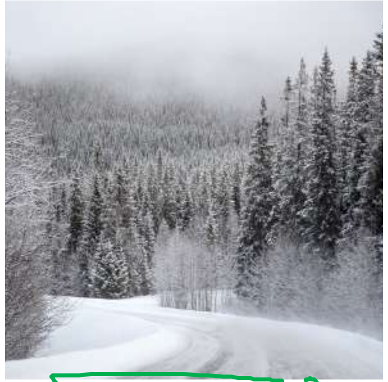
1 - pedestrian
2 - car
3 - motorcycle
4 - background

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

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

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

is there any object?

$x =$  

$$(x, y) \qquad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix}$$

p_c

"don't care"

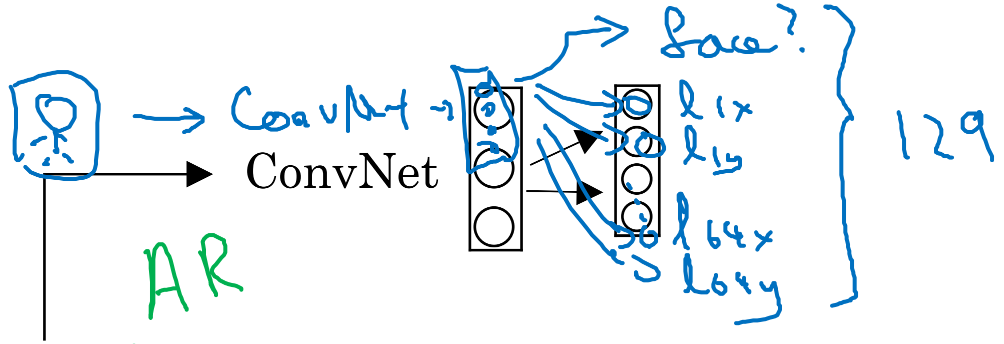
Andrew Ng

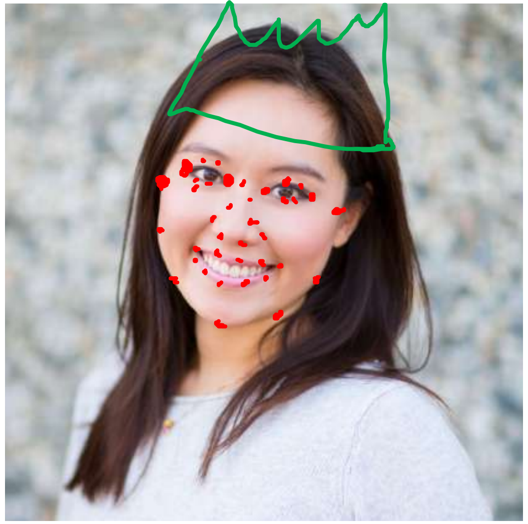
deeplearning.ai

Object Detection

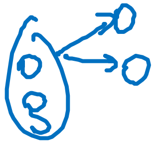
Landmark detection

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}
$\vdots$
l_{64x}, l_{64y}

X, y

$l_{1x}, l_{1y},$
$\vdots$
l_{32x}, l_{32y}

deeplearning.ai

Object Detection

Object detection

Car detection example

Training set:

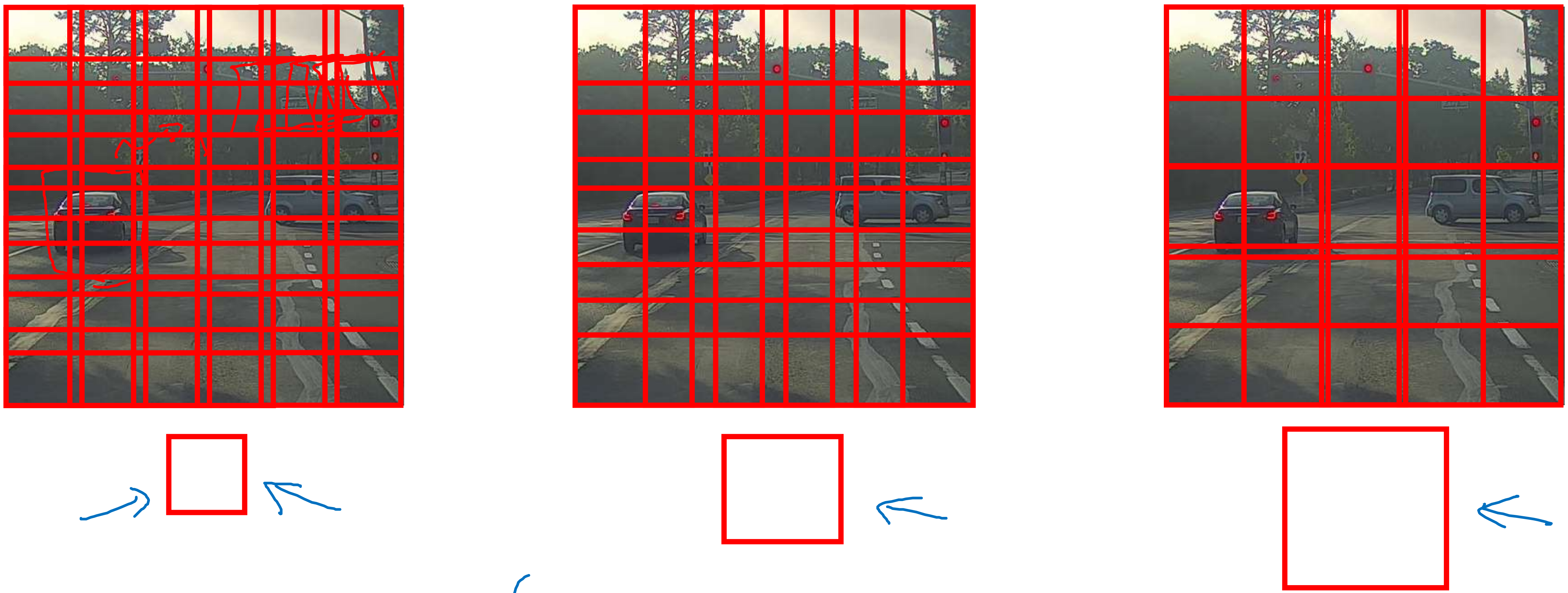
x	y
	1
	1
	1
	0
	0

 → ConvNet → y

Sliding windows detection

→ ConvNet → 0

→ ConvNet



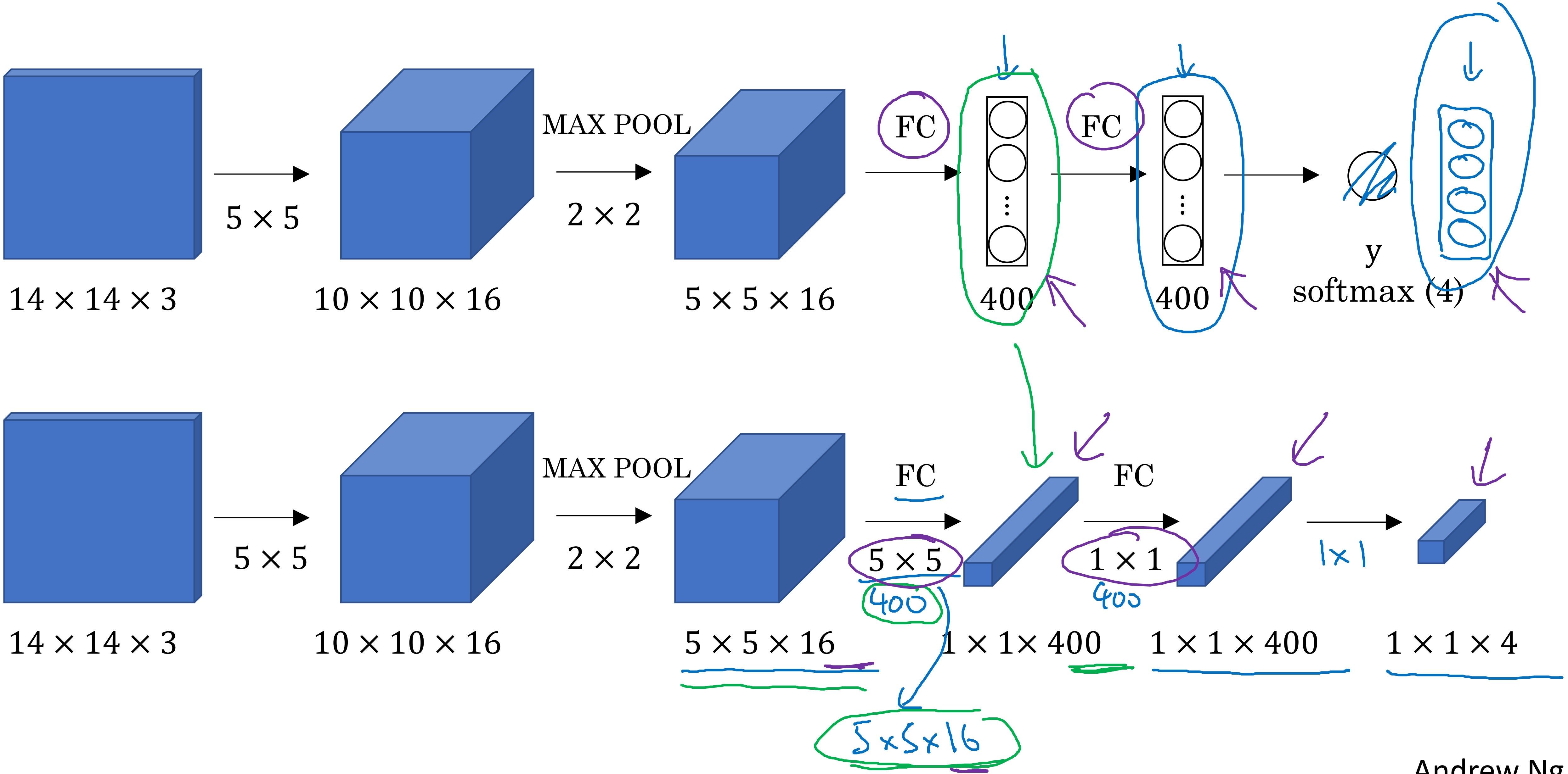
Computation cost

deeplearning.ai

Object Detection

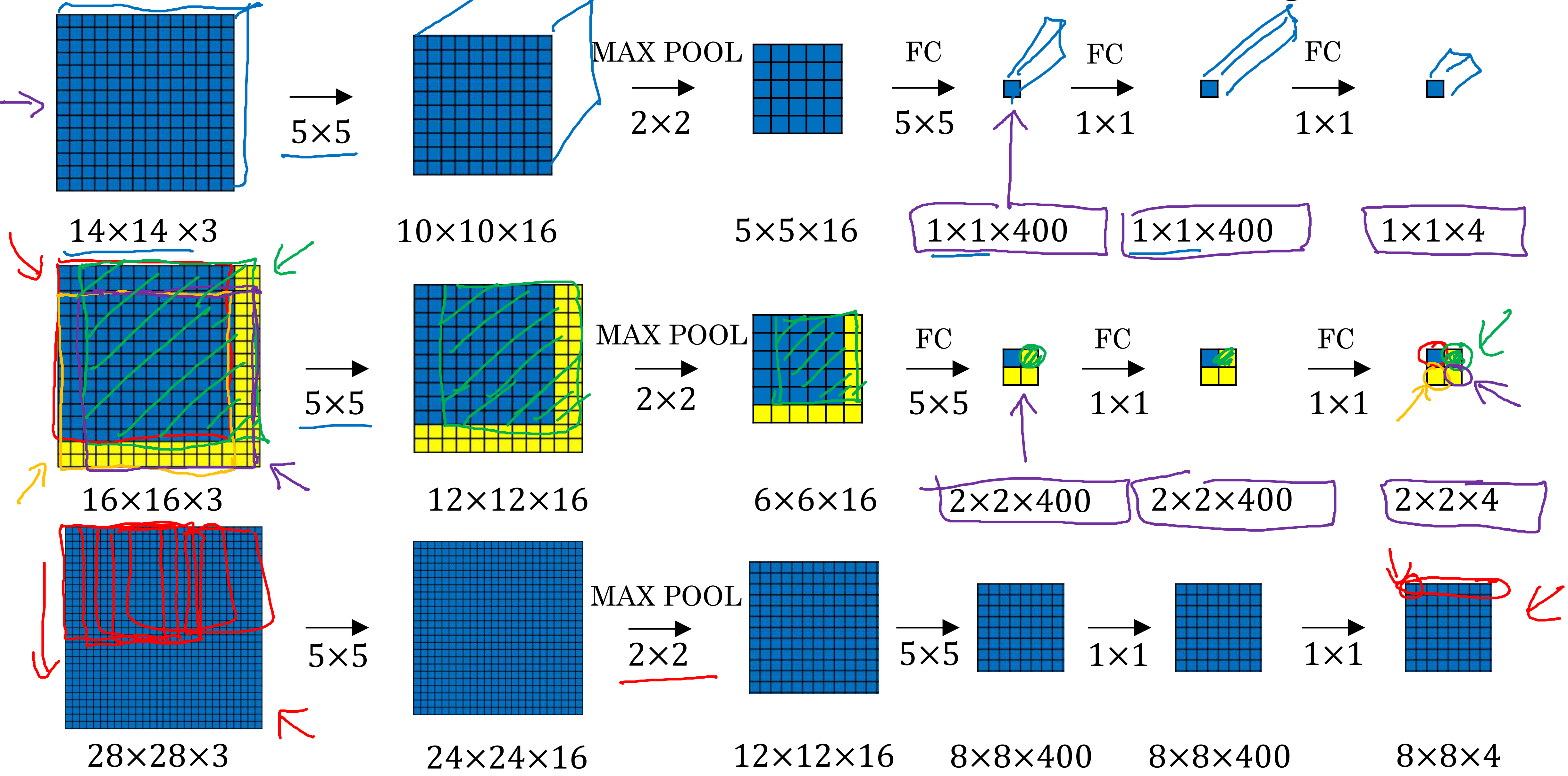
Convolutional implementation of sliding windows

Turning FC layer into convolutional layers



Andrew Ng

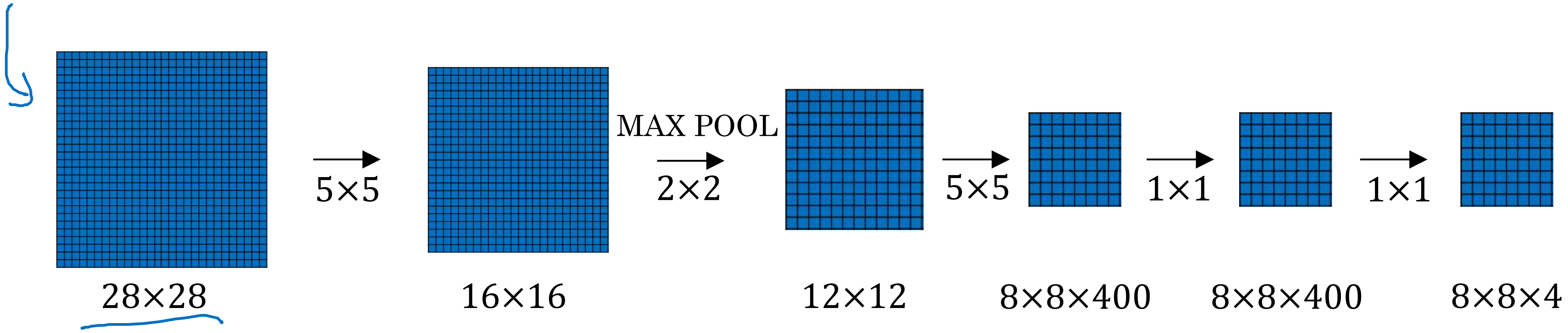
Convolution implementation of sliding windows

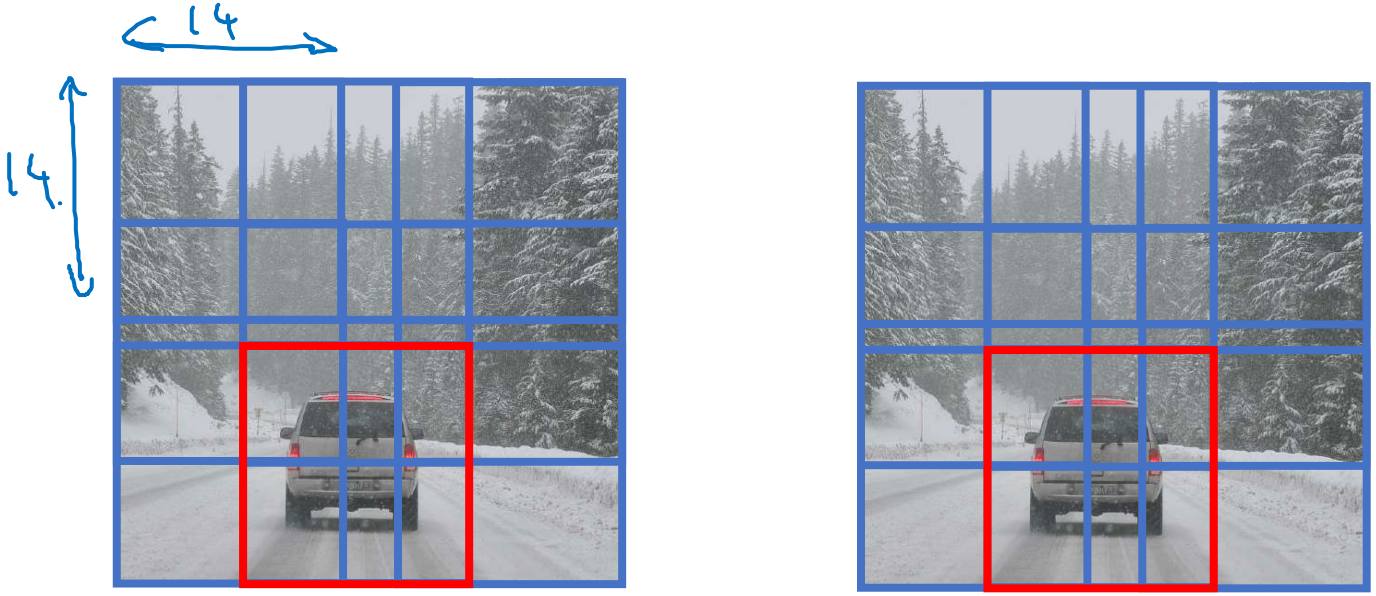


[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

Andrew Ng

Convolution implementation of sliding windows



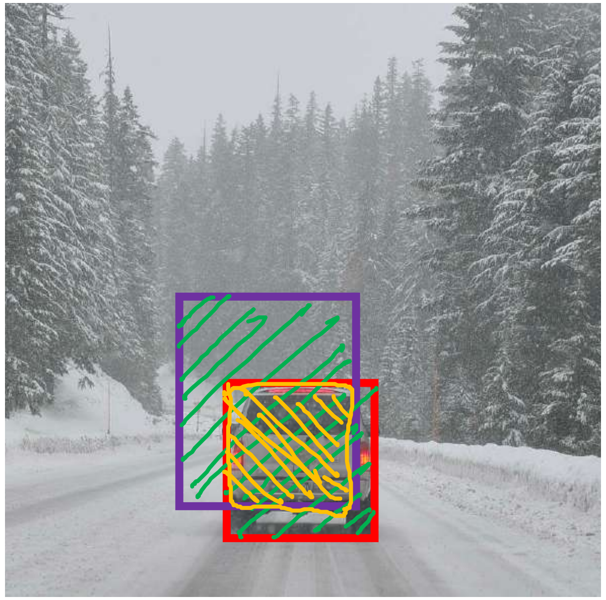


deeplearning.ai

Object Detection

Intersection over union

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{size of}}{\text{size of}}$$

"Correct" if IoU $\geq \boxed{0.5}$

0.6

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

Andrew Ng

deeplearning.ai

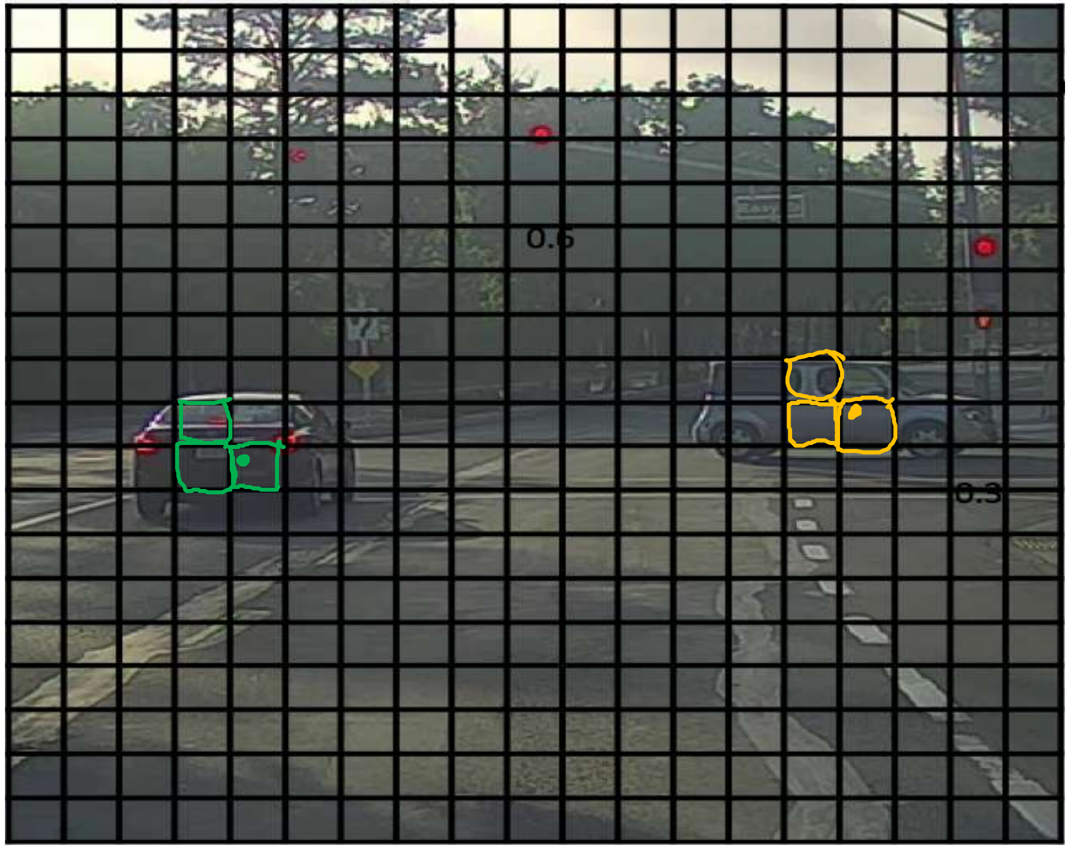
Object Detection

Non-max suppression

Non-max suppression example

Andrew Ng

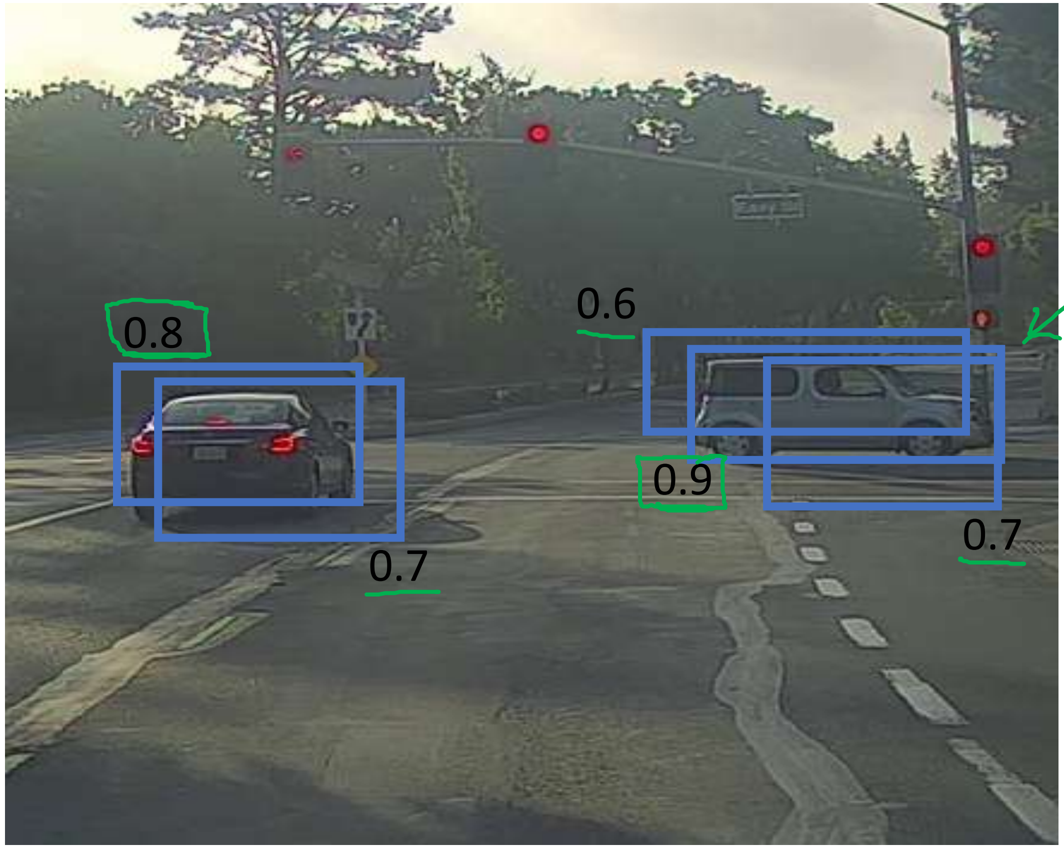
Non-max suppression example



19x19

Andrew Ng

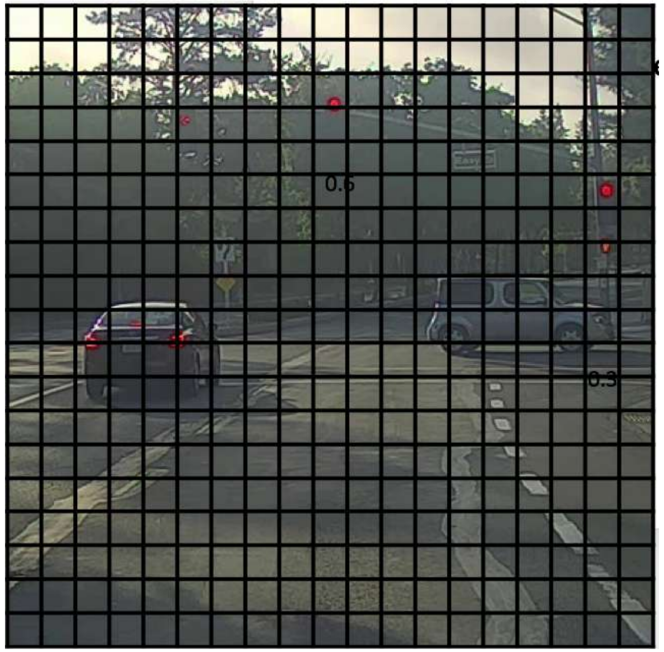
Non-max suppression example



P_c

Andrew Ng

Non-max suppression algorithm



19×19

Each output prediction is: $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$ $\boxed{\begin{matrix} c_1 \\ c_2 \\ c_3 \end{matrix}}$

Discard all boxes with $p_c \leq 0.6$

While there are any remaining boxes:

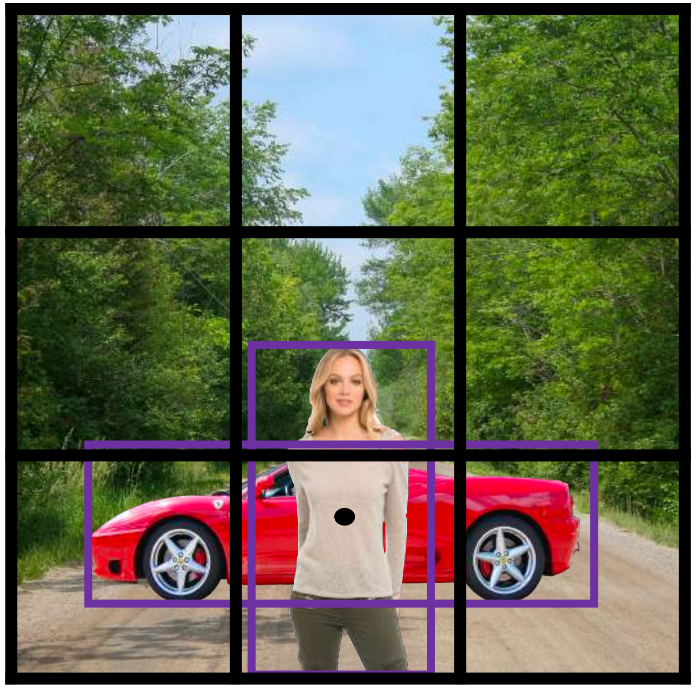
- Pick the box with the largest p_c Output that as a prediction.
- Discard any remaining box with IoU ≥ 0.5 with the box output in the previous step

deeplearning.ai

Object Detection

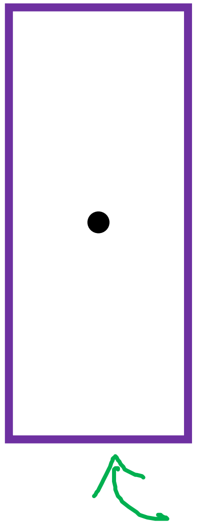
Anchor boxes

Overlapping objects:

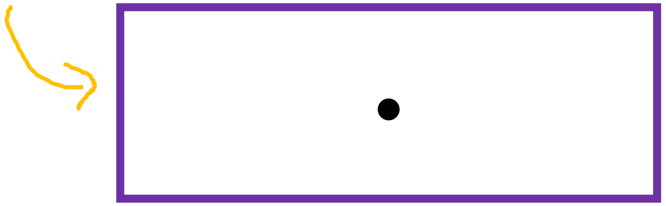


$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

Anchor box 1:



Anchor box 2:



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

anchor box 1

anchor box 2

[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

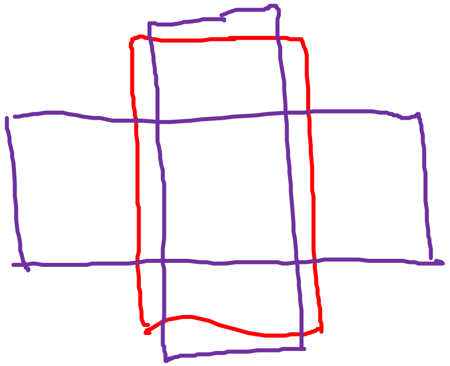
Anchor box algorithm

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.

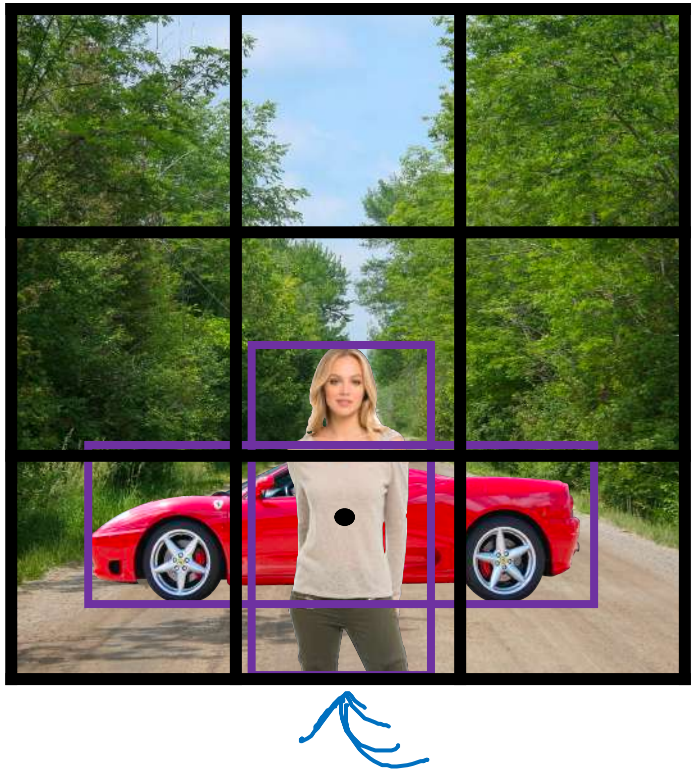
(grid cell, anchor box)

Output y:

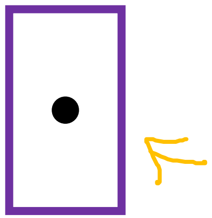
$3 \times 3 \times 16$

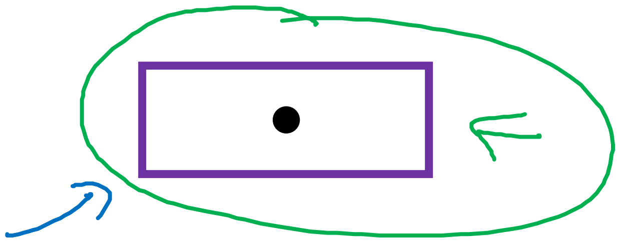
$3 \times 3 \times 2 \times 8$

Anchor box example



Anchor box 1: Anchor box 2:





$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad \begin{bmatrix} 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 1 \\ 0 \\ 0 \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \qquad \begin{array}{c} \text{car only?} \\ \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix} \end{array} \begin{array}{l} \left.\vphantom{\begin{matrix} 0\\?\\?\\?\\?\\?\\?\\? \end{matrix}}\right\} \text{anchor box 1} \\ \left.\vphantom{\begin{matrix} 1\\b_x\\b_y\\b_h\\b_w\\0\\1\\0 \end{matrix}}\right\} \text{anchor box 2} \end{array}$$

Andrew Ng

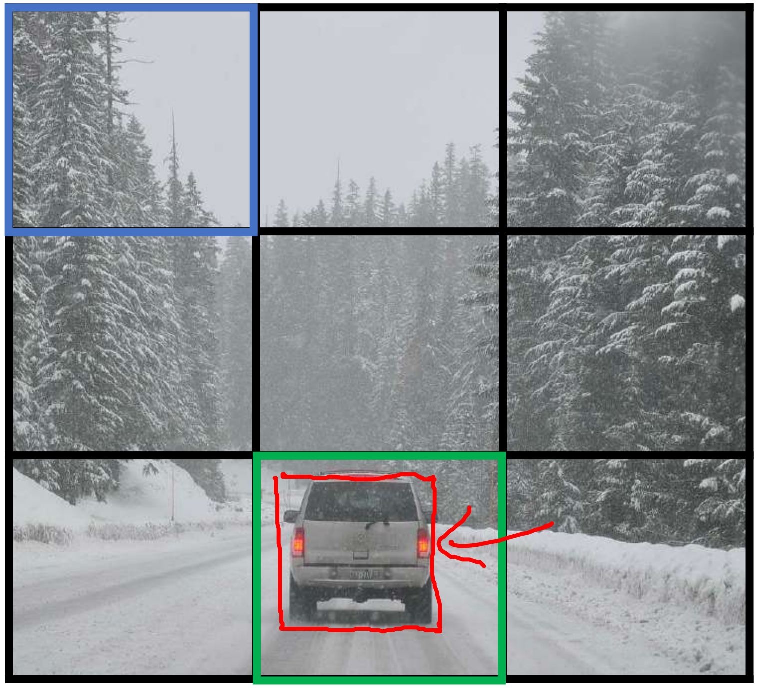
deeplearning.ai

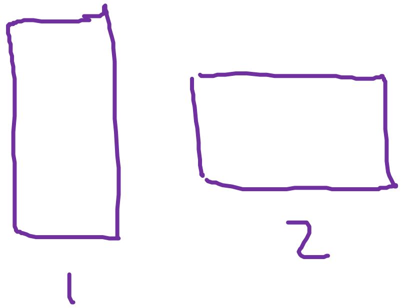
Object Detection

Putting it together: YOLO algorithm

Training

1 - pedestrian
2 - car
3 - motorcycle





$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \end{bmatrix} \qquad \begin{bmatrix} 0 \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ ? \\ 1 \\ b_x \\ b_y \\ b_h \\ b_w \\ 0 \\ 1 \\ 0 \end{bmatrix}$$

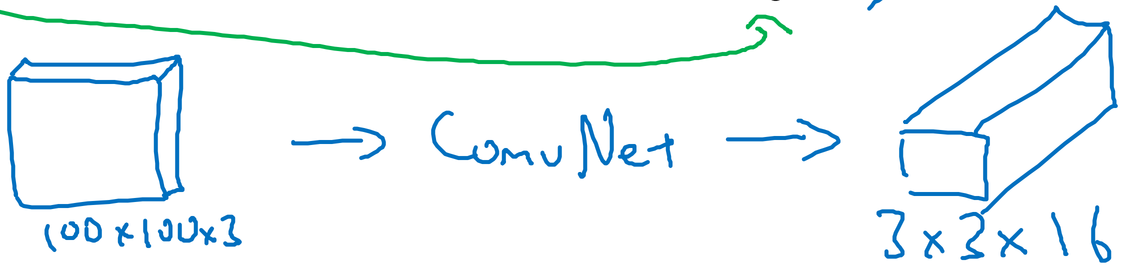
3×3×16

y is $\underbrace{3\times3}\times2\times8$

19×19×16
19×19×40

#anchors

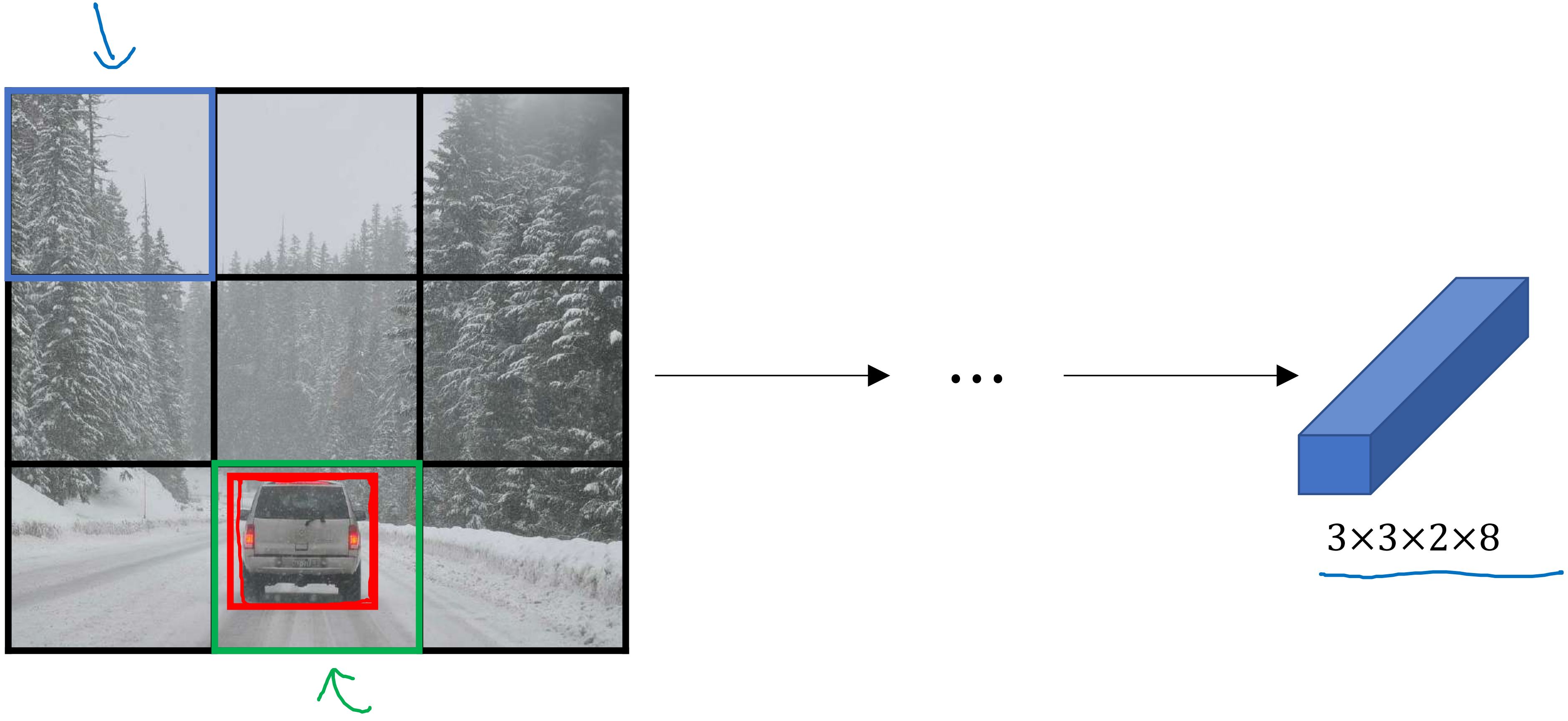
5 + #classes



[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

Andrew Ng

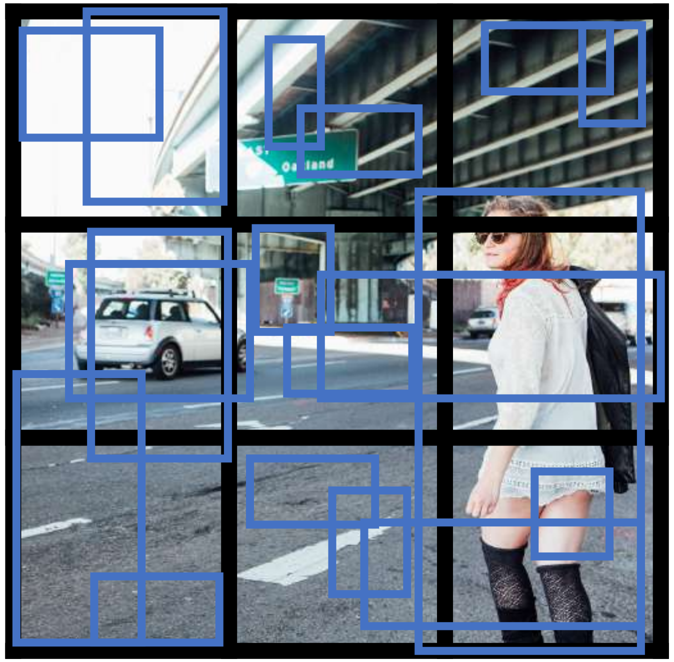
Making predictions



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

Andrew Ng

Outputting the non-max supressed outputs



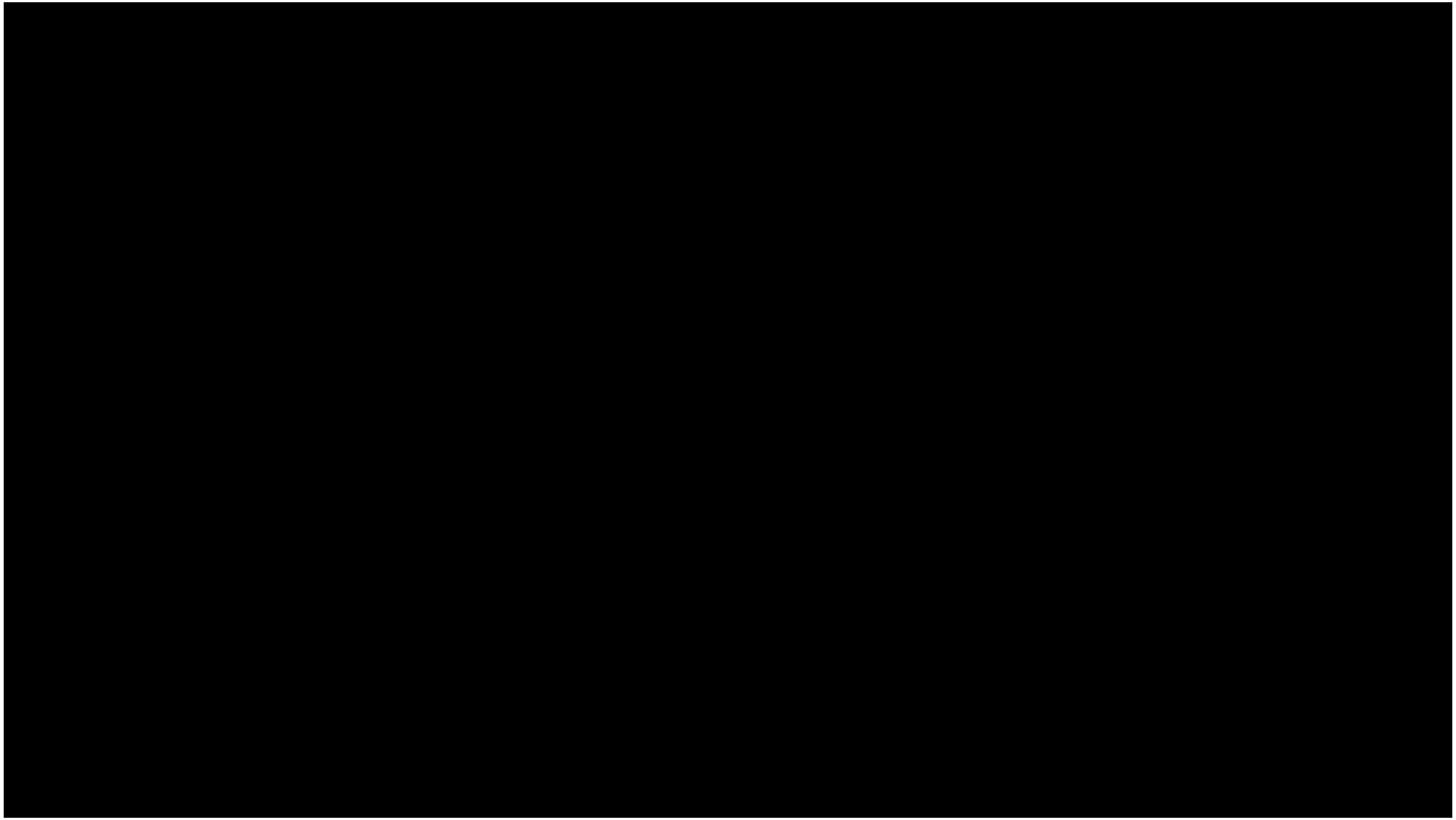
- For each grid call, get 2 predicted bounding boxes.
- Get rid of low probability predictions.
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

deeplearning.ai

Face recognition

What is face recognition?

Face recognition



[Courtesy of Baidu]

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 ←

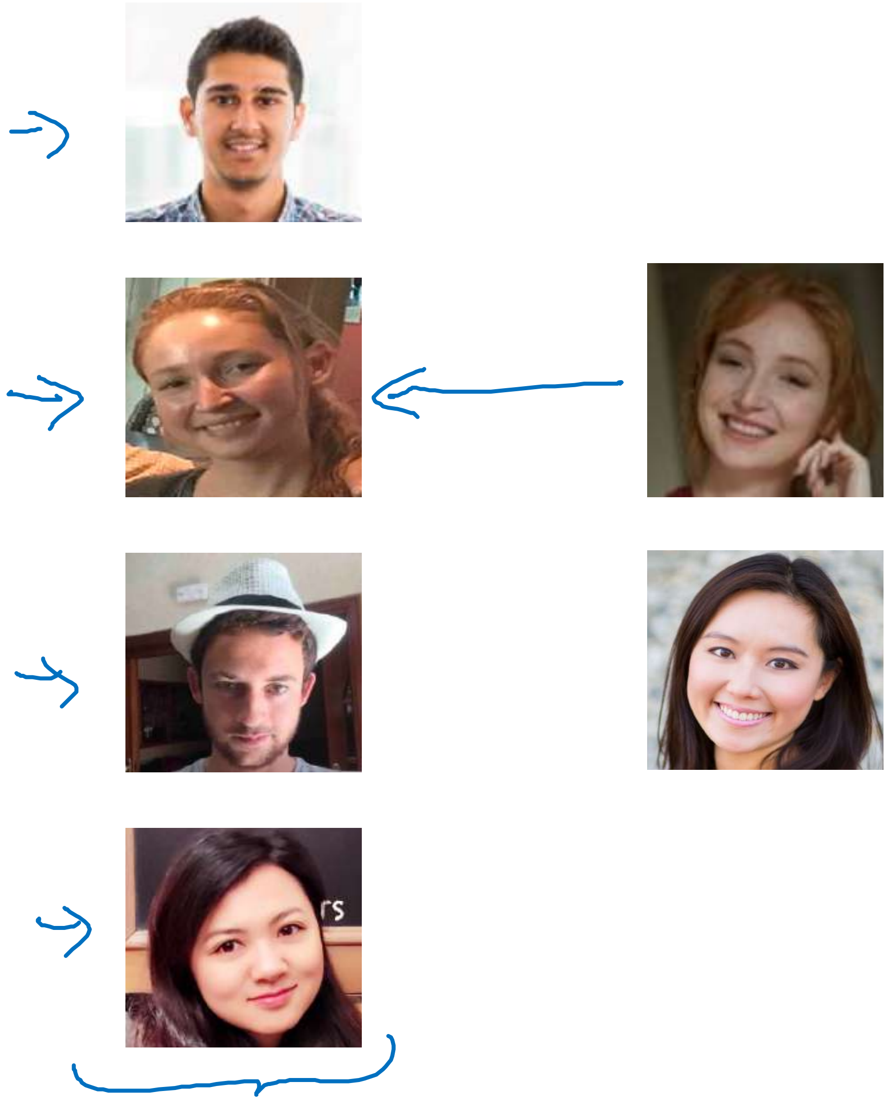
Andrew Ng

deeplearning.ai

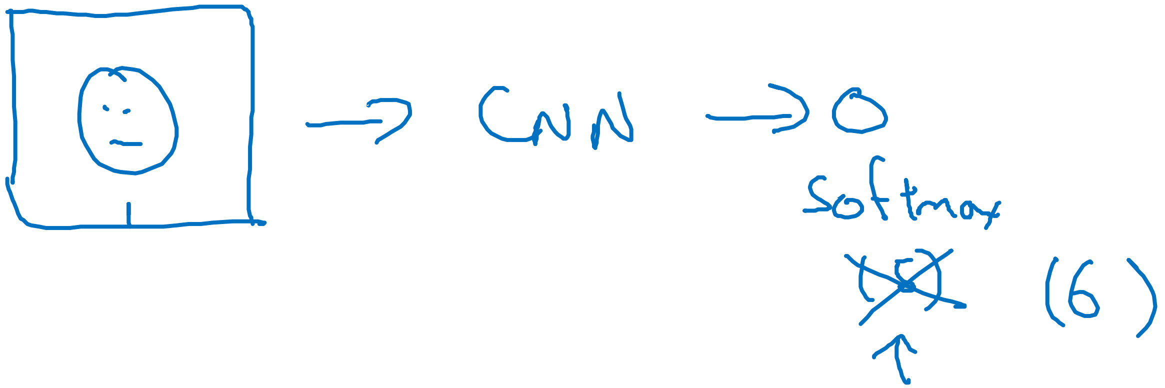
Face recognition

One-shot learning

One-shot learning



Learning from one example to recognize the person again

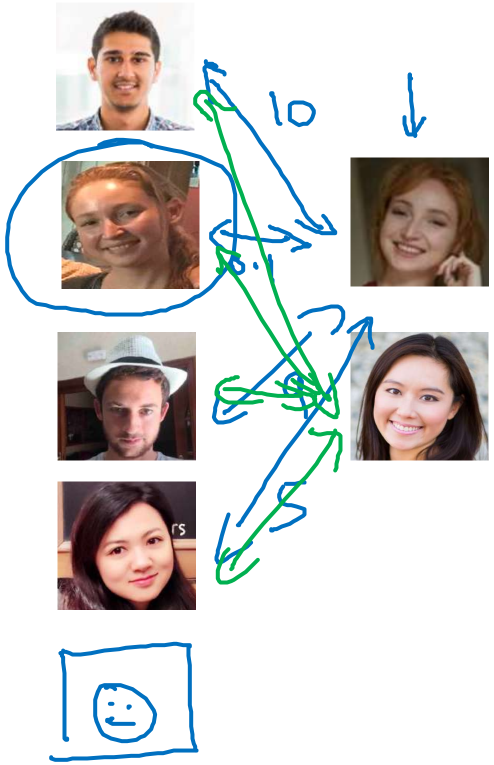


Learning a "similarity" function

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

If $d(\text{img1}, \text{img2}) \le \tau$ "same"
$\phantom{If\ d(\text{img1}, \text{img2})} > \tau$ "different"

Verification.



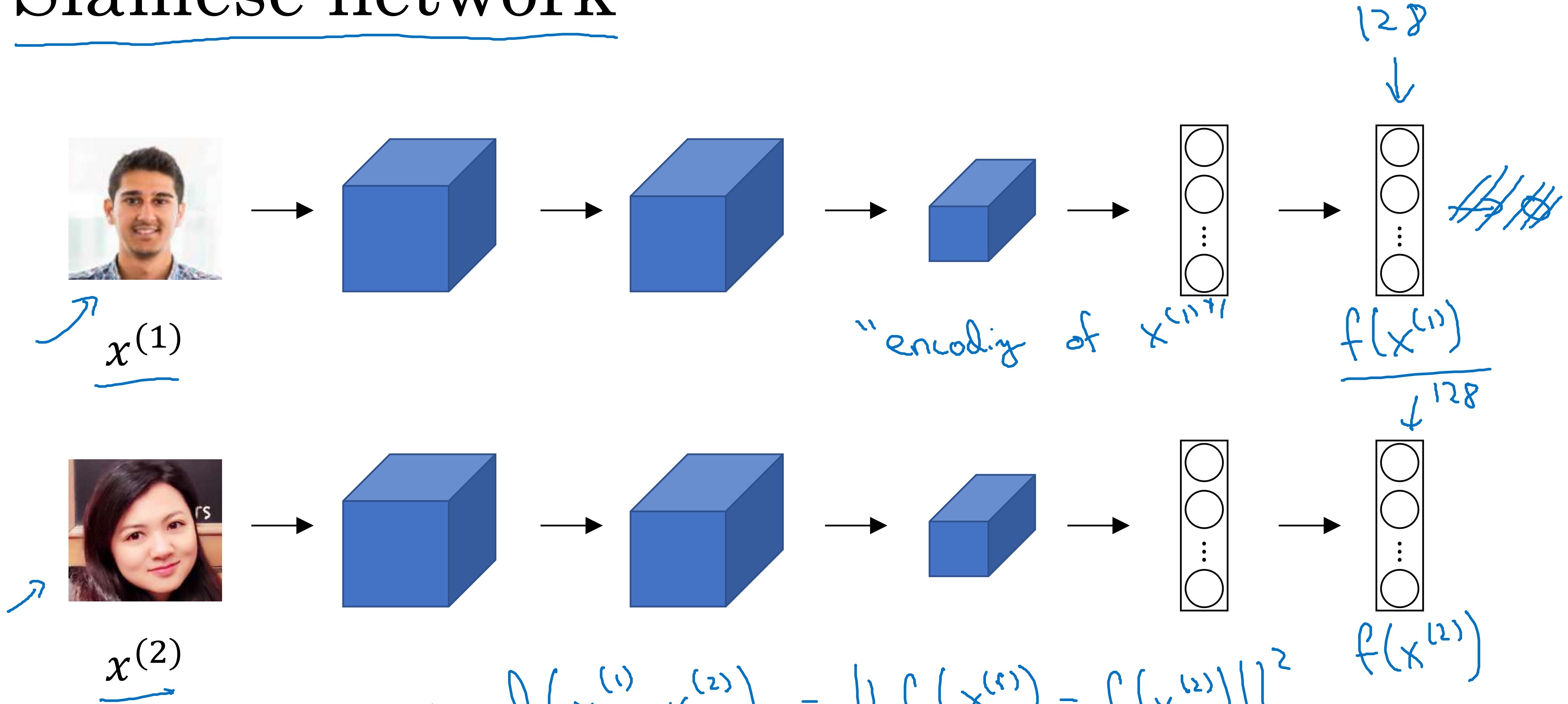
$d(\text{img1}, \text{img2})$

deeplearning.ai

Face recognition

Siamese network

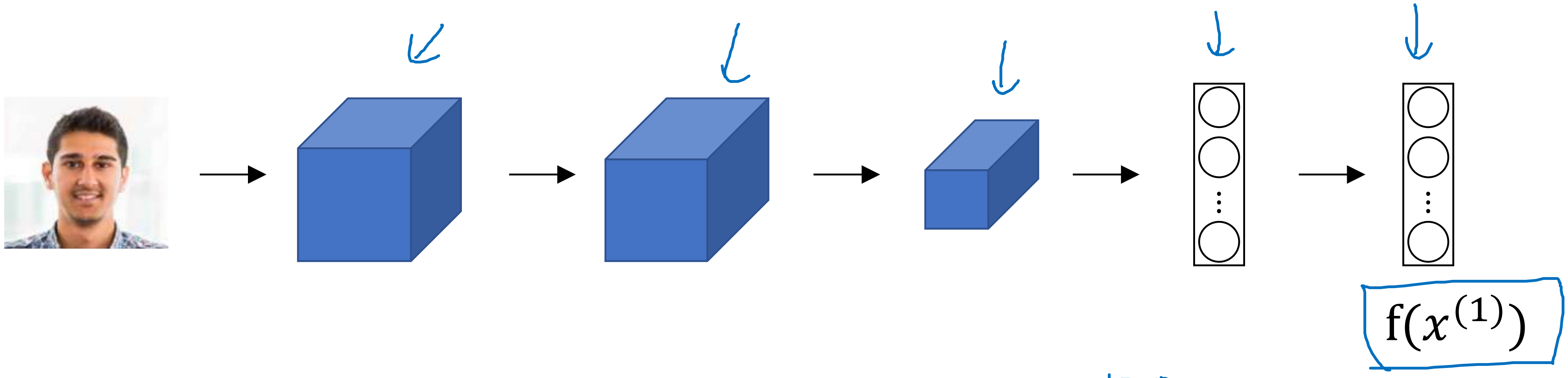
Siamese network



$$\rightarrow d(x^{(1)}, x^{(2)}) = \| f(x^{(1)}) - f(x^{(2)}) \|_2^2$$

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

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.

Face recognition

deeplearning.ai

Triplet loss

Learning Objective

Anchor — Positive

A — P

Anchor — Negative

A — N

$d(A,P) = 0.5$ $\rightarrow 0.2$

$d(A,N) = \cancel{0.51}$ 0.7

Want: $\underbrace{\| f(A) - f(P) \|^2}_{d(A,P)} + \alpha \le \underbrace{\| f(A) - f(N) \|^2}_{d(A,N)}$

$$\boxed{\underbrace{\| f(A) - f(P) \|^2}_{0} - \underbrace{\| f(A) - f(N) \|^2}_{0} + \underbrace{\alpha}_{\text{margin}} \le 0}$$

$f(\text{img}) = \vec{0}$

[Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering]

Loss function

Given 3 images A, P, N:

$$\mathcal{L}(A,P,N) = \max\left(\boxed{\|f(A)-f(P)\|^2 - \|f(A)-f(N)\|^2 + \alpha}, 0\right)$$

$\cancel{\leq 0}$ >0

$$J = \sum_{i=1}^{m} \mathcal{L}(A^{(i)}, P^{(i)}, N^{(i)})$$

A, P

Training set: 10k pictures of 1k persons

[Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering]

Choosing the triplets A,P,N

During training, if A,P,N are chosen randomly, $d(A,P) + \alpha \le d(A,N)$ is easily satisfied.

$\|f(A)-f(P)\|^2 + \alpha \le \|f(A)-f(N)\|^2$

Choose triplets that're "hard" to train on.

$d(A,P) + \alpha \le d(A,N)$

$d(A,P) \approx d(A,N)$

$\downarrow \qquad \uparrow$

Face Net

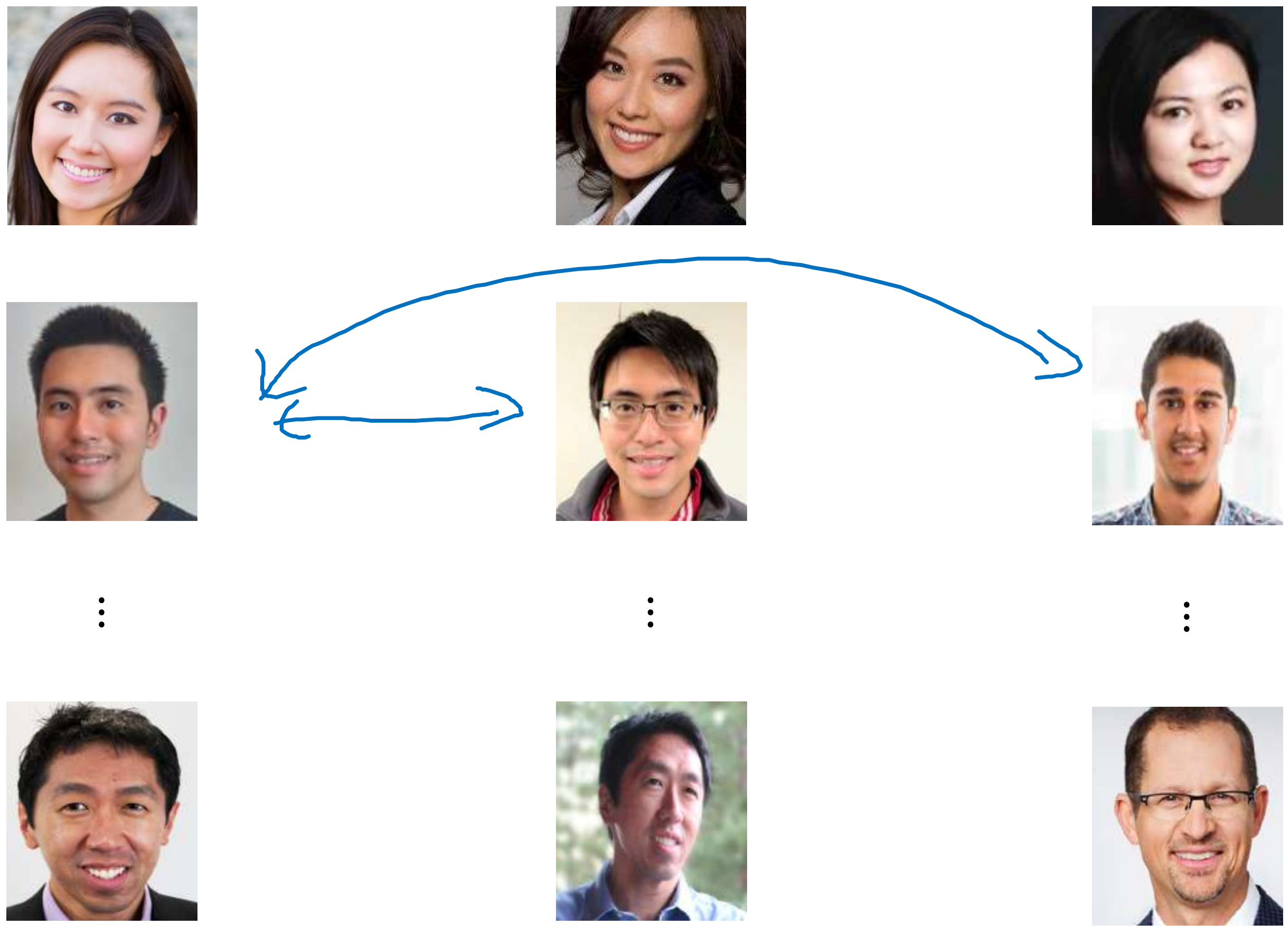
Deep Face

[Schroff et al.,2015, FaceNet: A unified embedding for face recognition and clustering]

Andrew Ng

Training set using triplet loss

Anchor | Positive | Negative



J

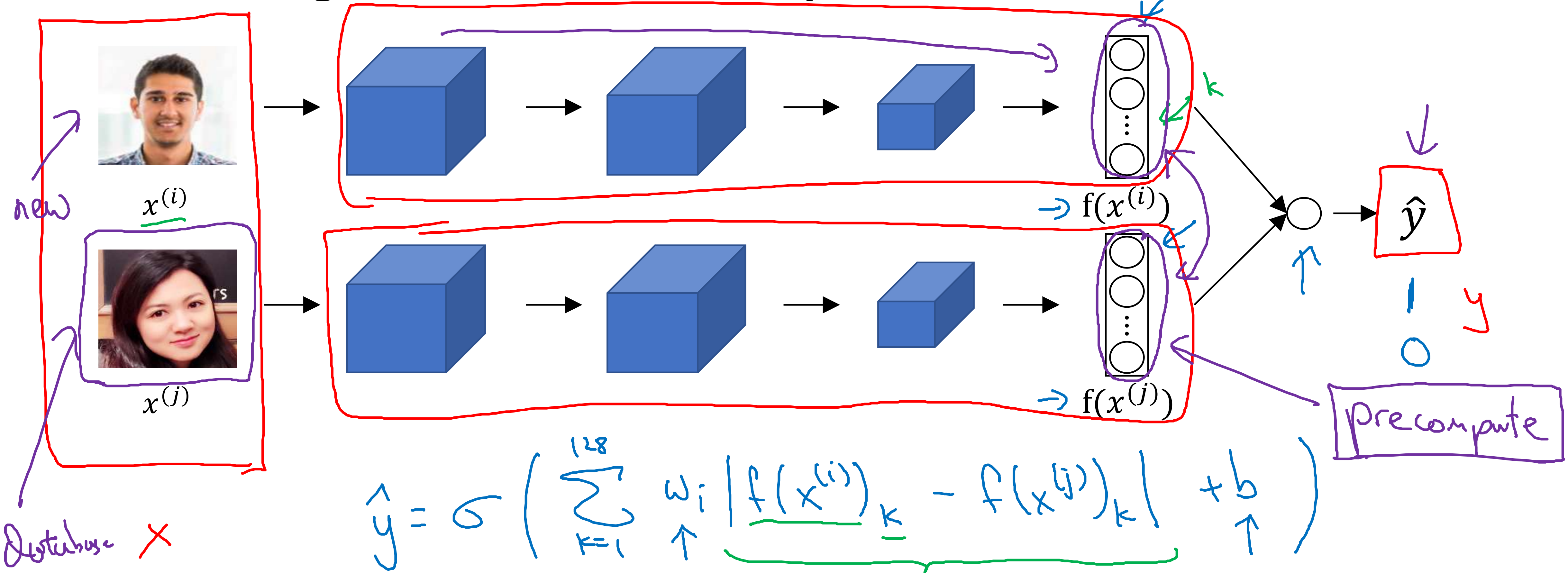
$d(x^{(i)}, x^{(j)})$

deeplearning.ai

Face recognition

Face verification and binary classification

Learning the similarity function



$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_i \underbrace{\left| f(x^{(i)})_k - f(x^{(j)})_k \right|}_{\frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k}} + b\right)$$

χ^2

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

Andrew Ng

Face verification supervised learning

x y

1 "same"

0 "different"

0

1

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

Andrew Ng

Neural Style Transfer

deeplearning.ai

What is neural style transfer?

Neural style transfer


Content (C) Style (S)

Generated image (G)


Content (C) Style (S)

Generated image (G)

[Images generated by Justin Johnson]

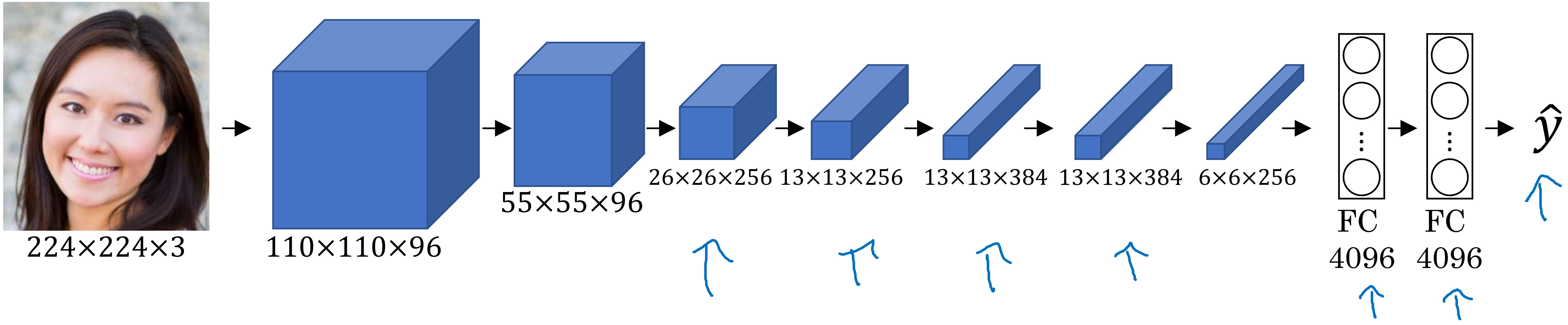
Andrew Ng

deeplearning.ai

Neural Style Transfer

What are deep ConvNets learning?

Visualizing what a deep network is learning



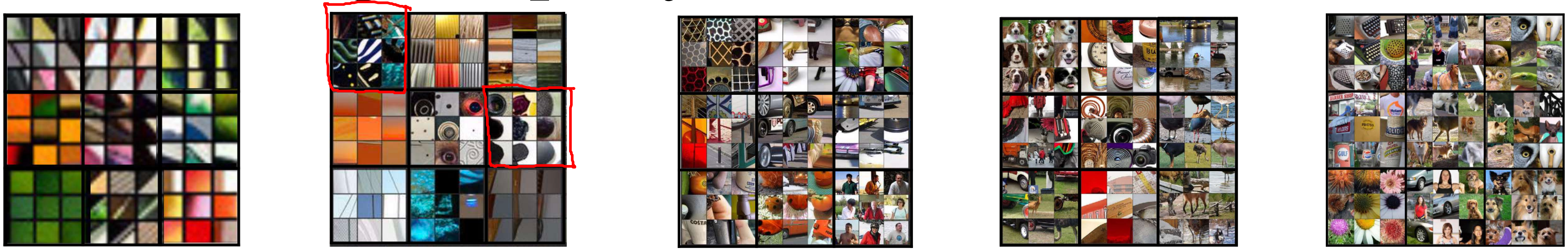
Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

[Zeiler and Fergus., 2013, Visualizing and understanding convolutional networks]

Andrew Ng

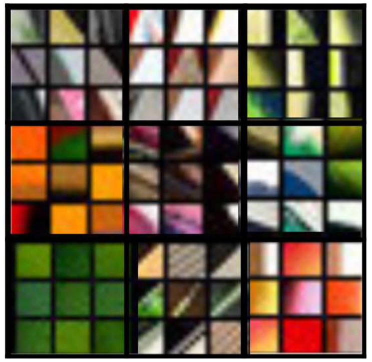
Visualizing deep layers



Layer 1 Layer 2 Layer 3 Layer 4 Layer 5

Andrew Ng

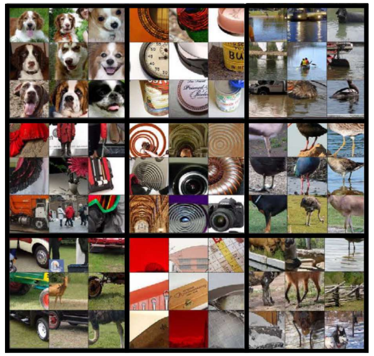
Visualizing deep layers: Layer 1



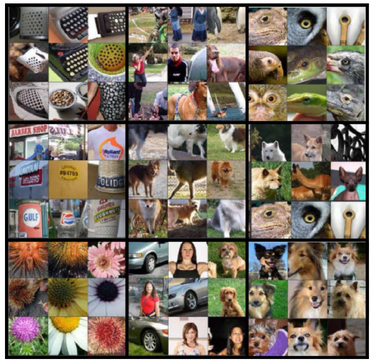
Layer 1

Layer 2

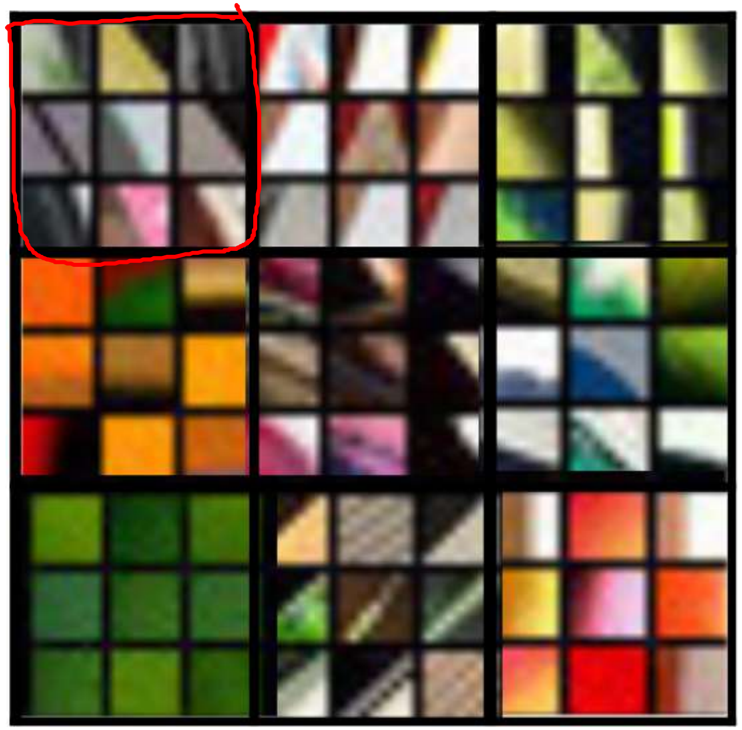
Layer 3

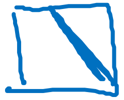


Layer 4

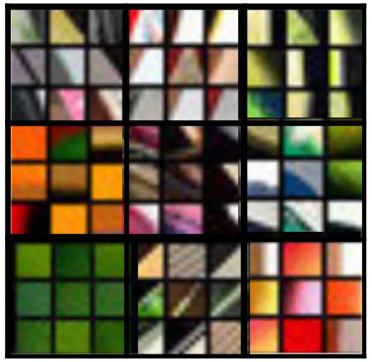


Layer 5





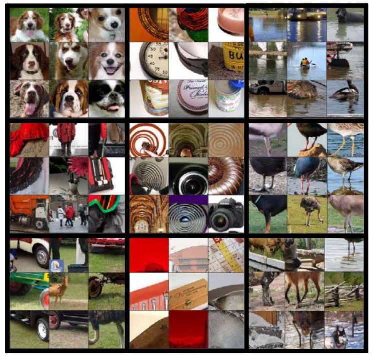
Visualizing deep layers: Layer 2



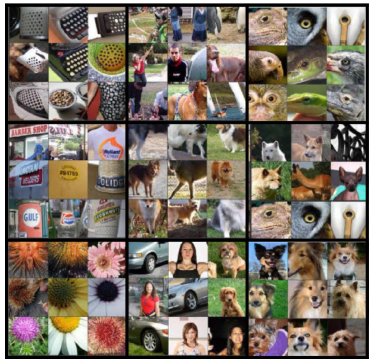
Layer 1

Layer 2

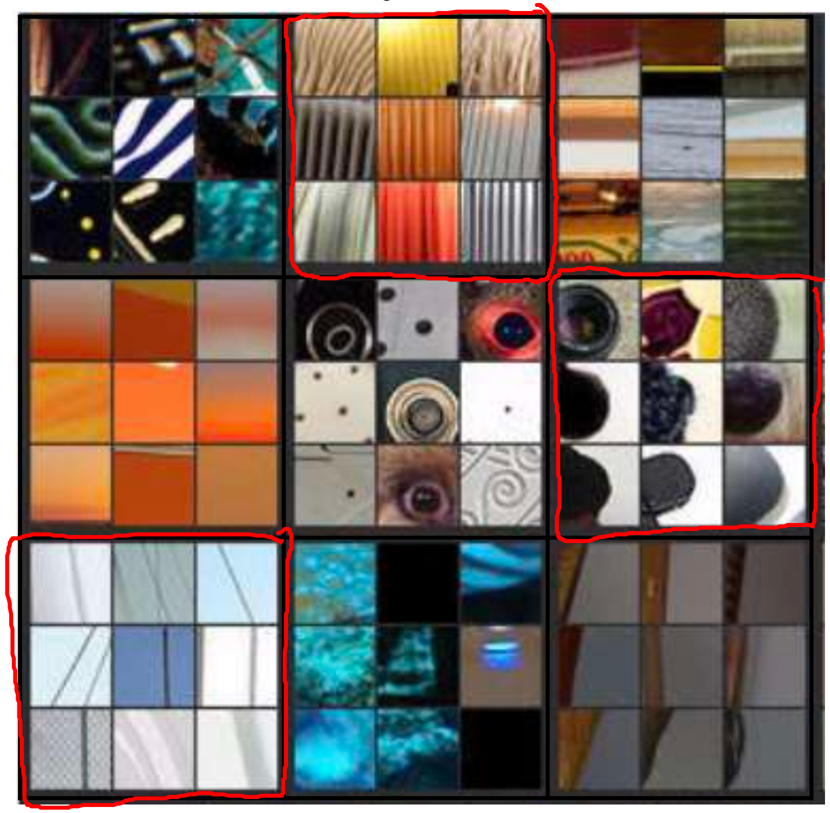
Layer 3



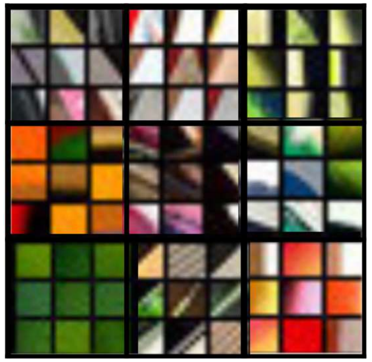
Layer 4



Layer 5

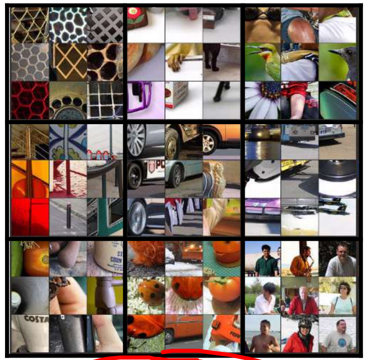


Visualizing deep layers: Layer 3

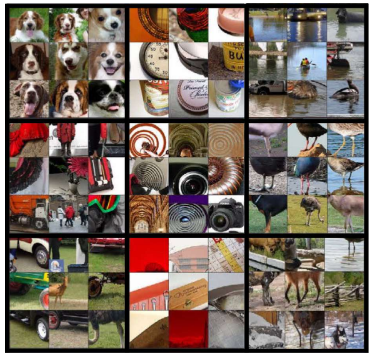


Layer 1

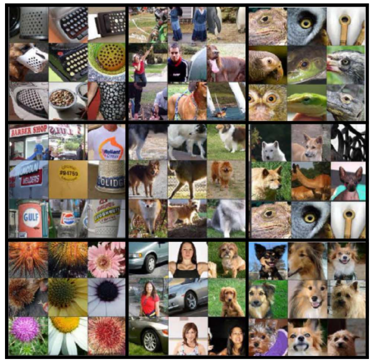
Layer 2



Layer 3



Layer 4



Layer 5

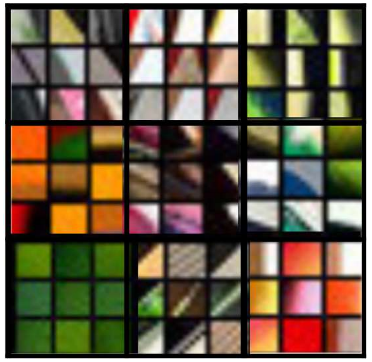
Andrew Ng

Visualizing deep layers: Layer 3

Visualizing deep layers: Layer 4

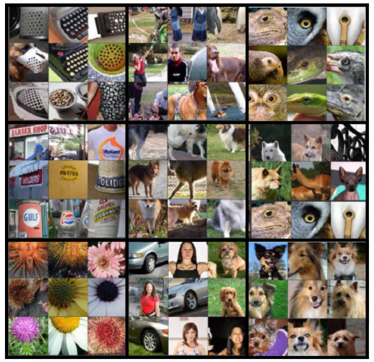
Andrew Ng

Visualizing deep layers: Layer 5



Layer 1

L



Layer 5

Andrew Ng

deeplearning.ai

Neural Style Transfer

Cost function

Neural style transfer cost function

Content C

Style S

Generated image G

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

[Gatys et al., 2015. A neural algorithm of artistic style. Images on slide generated by Justin Johnson]

Find the generated image G

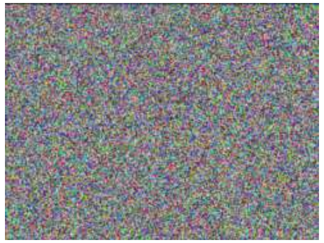
1. Initiate G randomly

   G: 100×100×3

   RGB

2. Use gradient descent to minimize $J(G)$

$$G := G - \frac{\partial}{\partial G} J(G)$$


[Gatys et al., 2015. A neural algorithm of artistic style]

deeplearning.ai

Neural Style Transfer

Content cost function

Content cost function

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

- Say you use hidden layer l to compute content cost.
- Use pre-trained ConvNet. (E.g., VGG network)
- Let $a^{[l](C)}$ and $a^{[l](G)}$ be the activation of layer l on the images
- If $a^{[l](C)}$ and $a^{[l](G)}$ are similar, both images have similar content

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

[Gatys et al., 2015. A neural algorithm of artistic style]

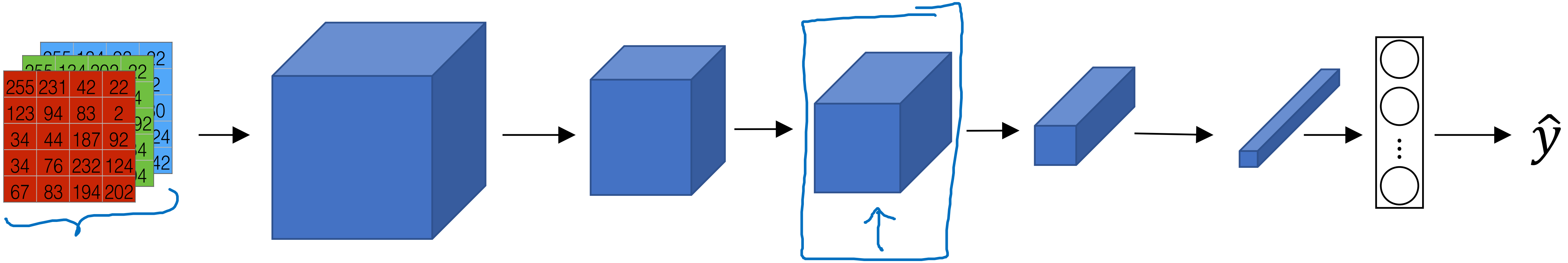
Andrew Ng

Neural Style Transfer

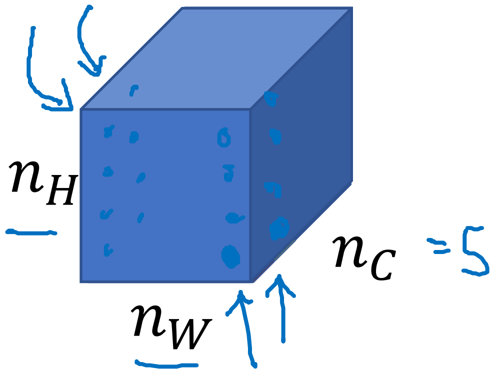
Style cost function

deeplearning.ai

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.



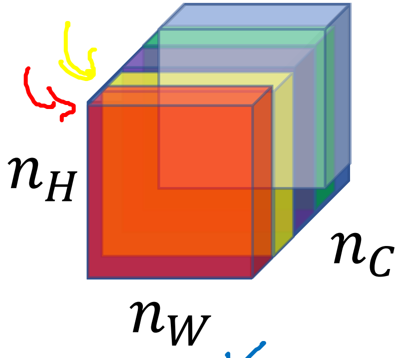
How correlated are the activations across different channels?

[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

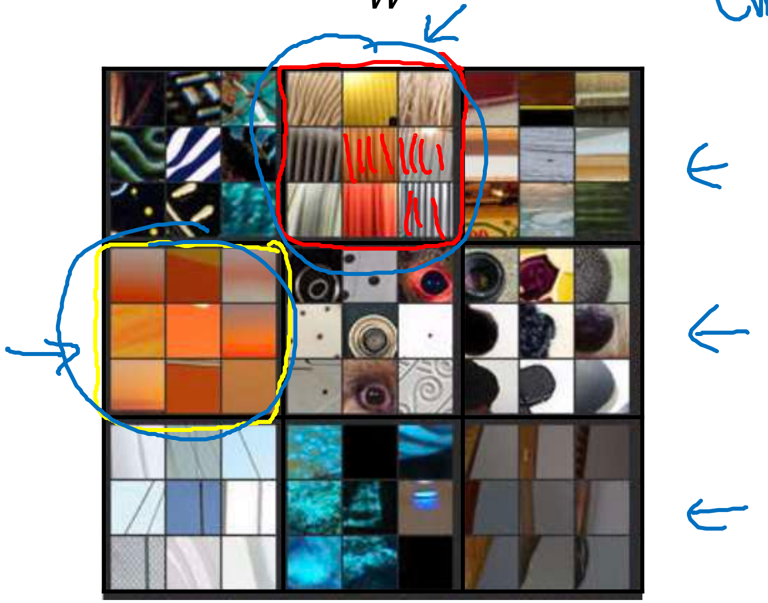
Intuition about style of an image

Style image

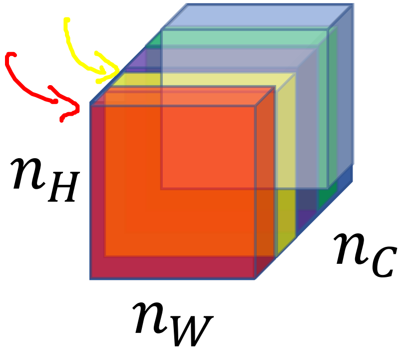


Correlated?

Uncorrelated



Generated Image



[Gatys et al., 2015. A neural algorithm of artistic style]

Style matrix

Let $a^{[l]}_{i,j,k}$ = activation at (i, j, k). $G^{[l]}$ is $n_c^{[l]} \times n_c^{[l]}$

H W C

n_c

$G^{[l]}_{kk'}$

$k = 1, \ldots, n_c^{[l]}$

$$\rightarrow G^{[l](S)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](S)}_{ijk} \, a^{[l](S)}_{ijk'}$$

$$\rightarrow G^{[l](G)}_{kk'} = \sum_{i=1}^{n_H^{[l]}} \sum_{j=1}^{n_W^{[l]}} a^{[l](G)}_{ijk} \, a^{[l](G)}_{ijk}$$

"Gram matrix"

$$J^{[l]}_{style}(S, G) = \frac{1}{(\ldots)} \left\| G^{[l](S)} - G^{[l](G)} \right\|_F^2$$

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

β

[Gatys et al., 2015. A neural algorithm of artistic style]

Andrew Ng

Style cost function

$$\| G^{[l](S)} - G^{[l](G)} \|_F^2$$

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

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

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

[Gatys et al., 2015. A neural algorithm of artistic style]

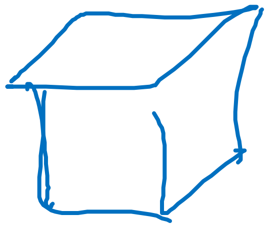
Andrew Ng

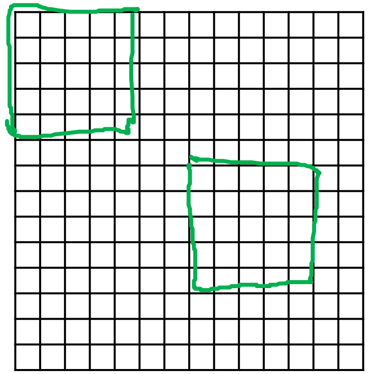
deeplearning.ai

Convolutional Networks in 1D or 3D

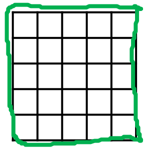
1D and 3D generalizations of models

Convolutions in 2D and 1D





*



2D input image
14×14

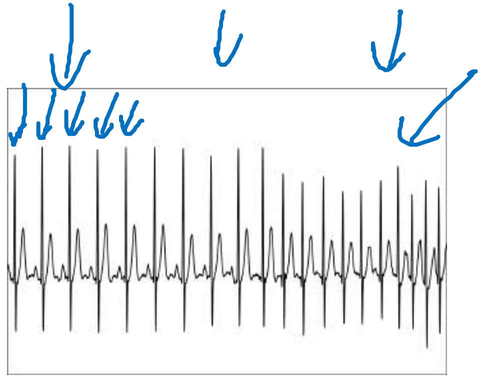
2D filter
5×5

$14 \times 14 \times \underline{3} \ * \ 5 \times 5 \times \underline{3}$

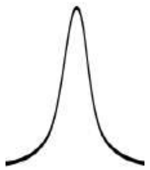
$\rightarrow \underline{10 \times 10} \times 16$

$10 \times 10 \times \underline{16} \ * \ 5 \times 5 \times \underline{16}$

$\rightarrow 6 \times 6 \times \underline{32}$



*



1	20	15	3	18	12	4	17

14

1	3	10	3	1

5

$14 \times \underline{1} \ * \ 5 \times \underline{1}$

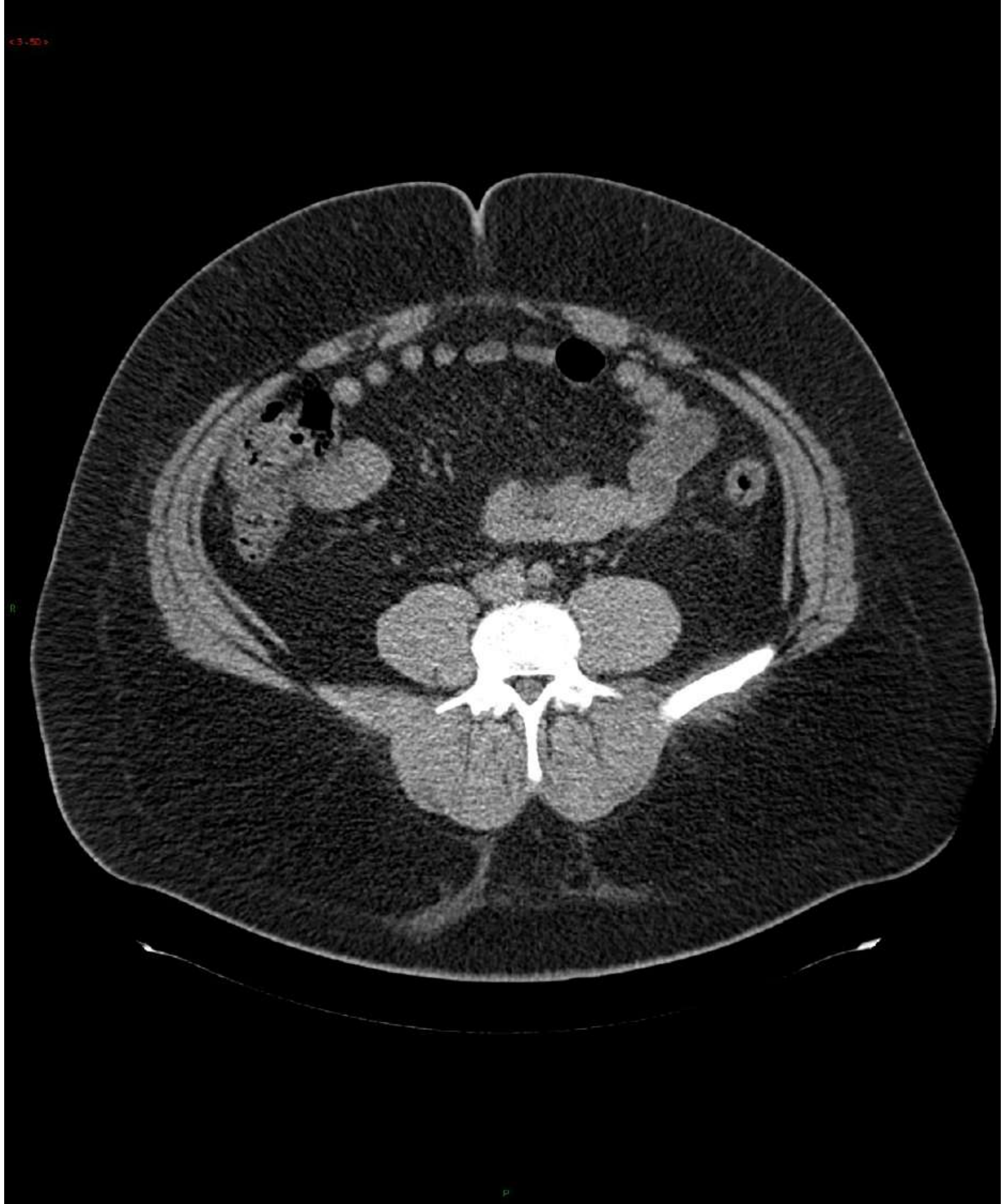
$\rightarrow 10 \times \underline{16}$

$10 \times \underline{16} \ * \ 5 * \underline{16}$

$\rightarrow 6 * \underline{32}$

Andrew Ng

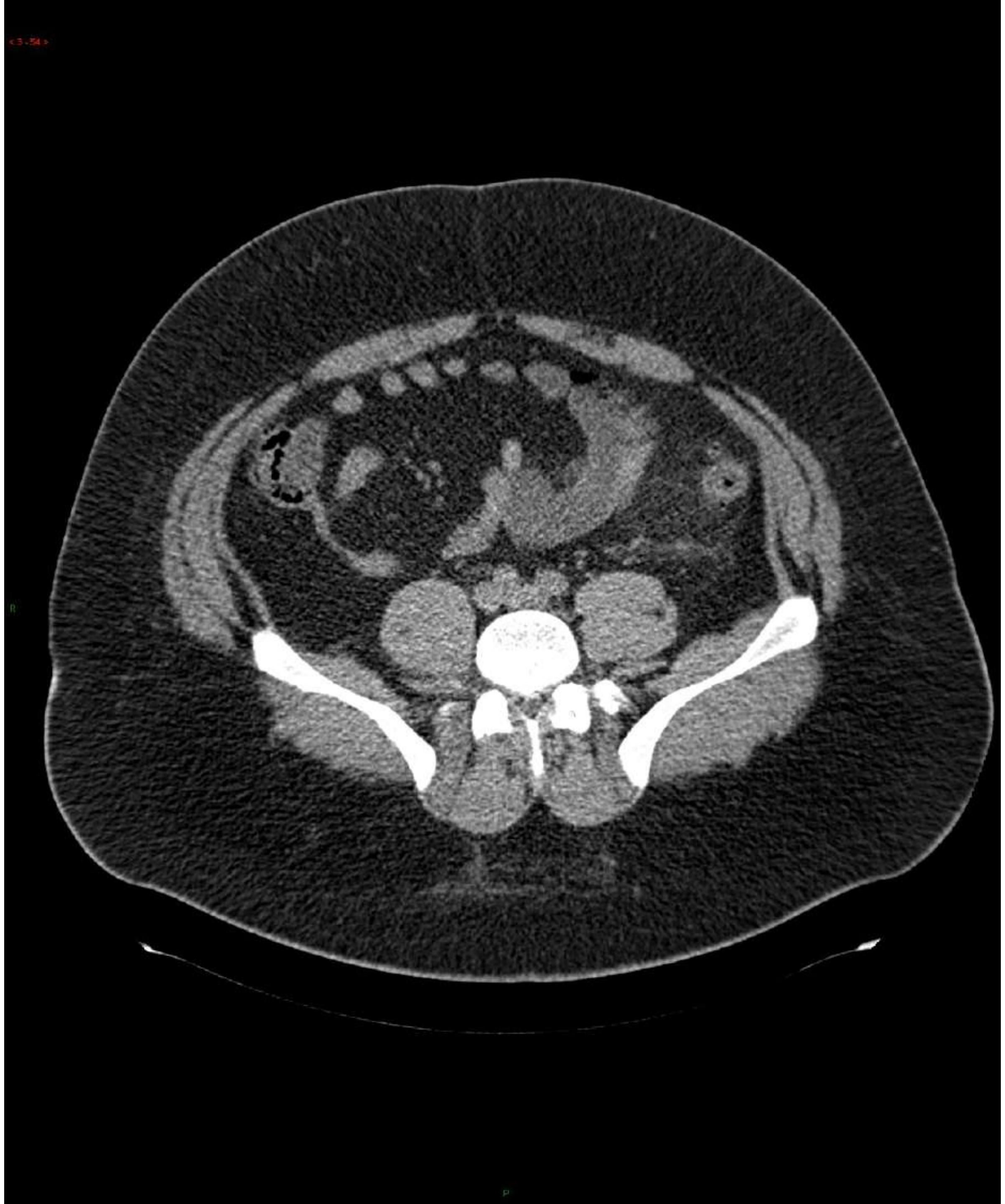
3D data



3D data

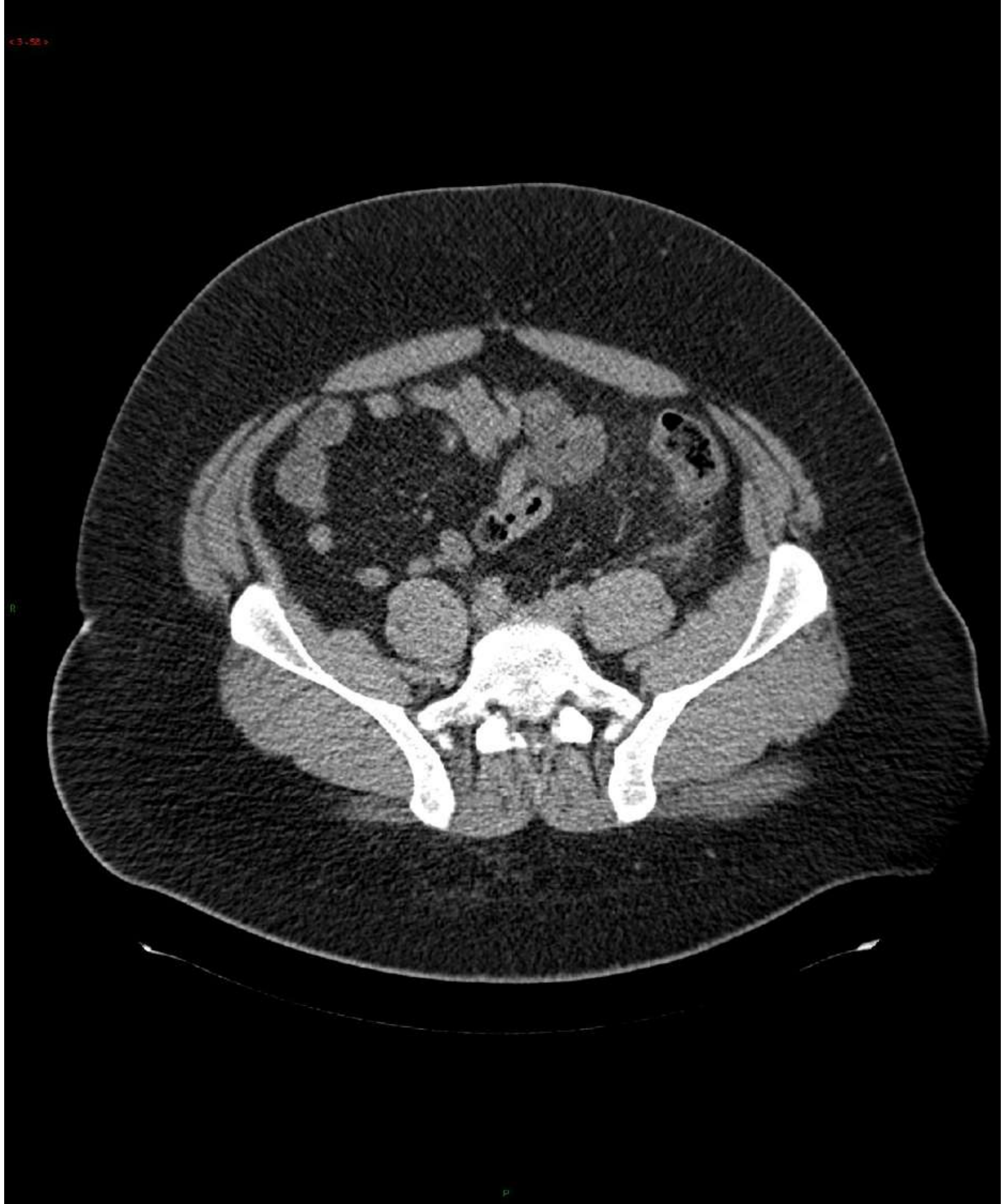
Andrew Ng

3D data

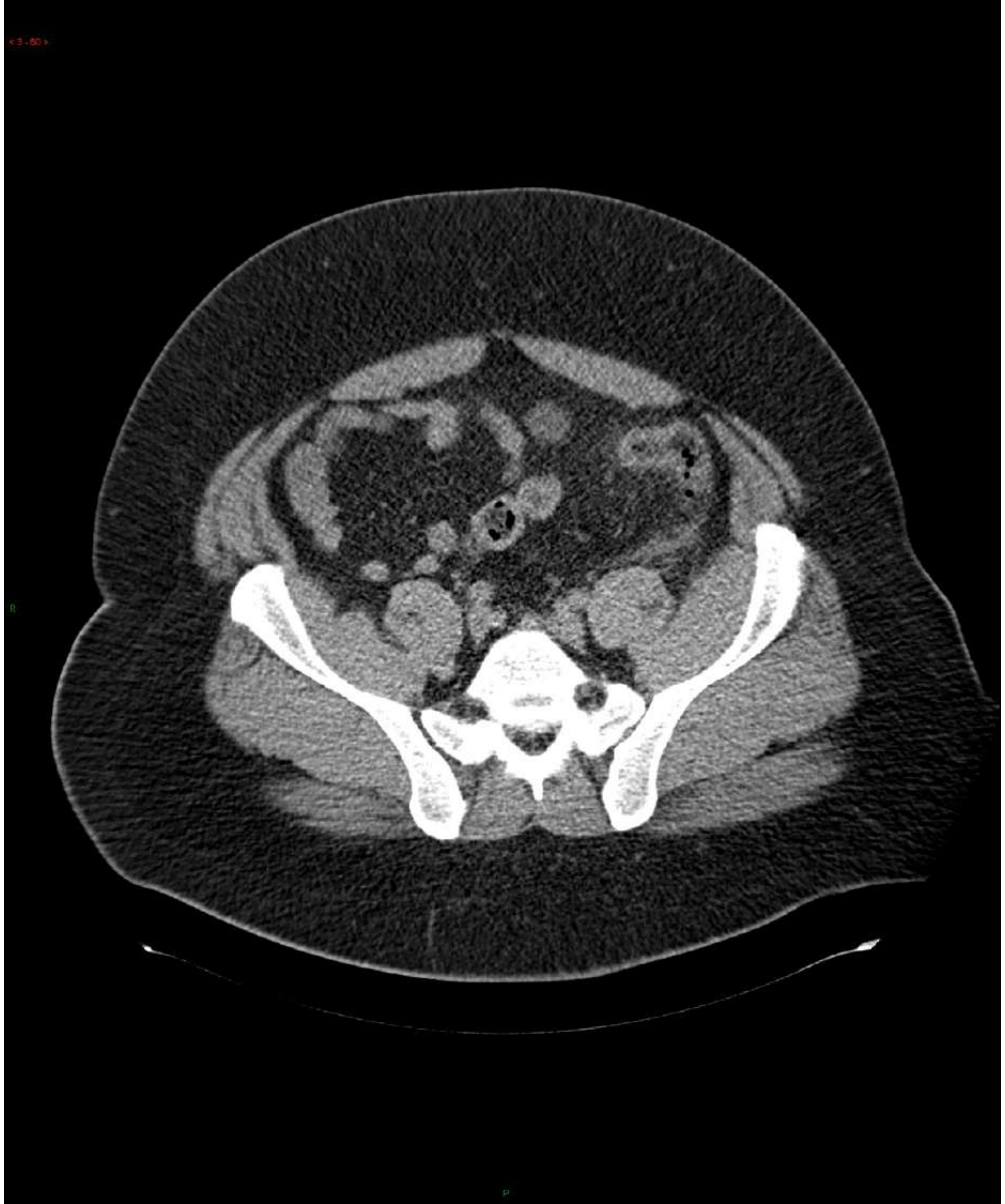


3D data

3D data



3D data



3D data

3D data

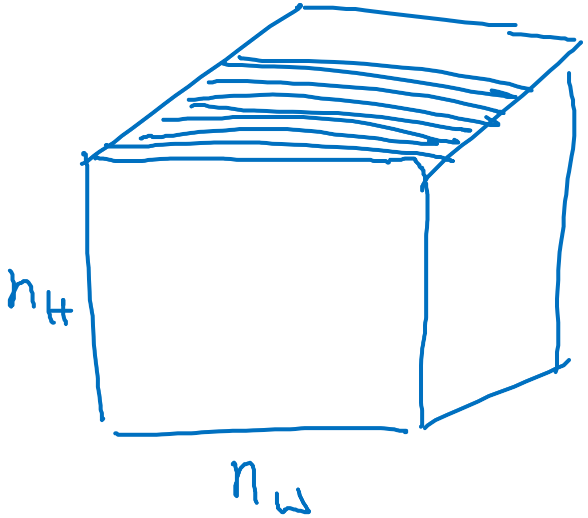
3D data

Andrew Ng

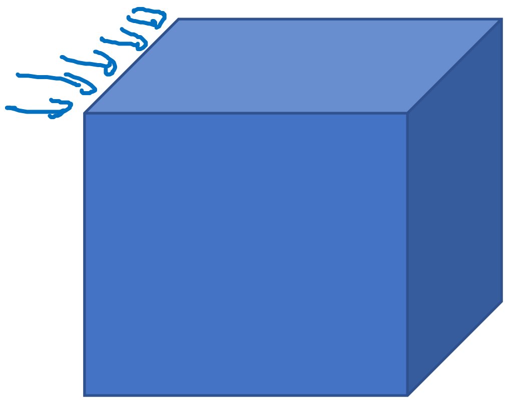
3D data

Andrew Ng

3D data

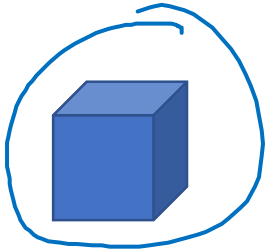


3D convolution



3D volume

*



3D filter

n_c

$14 \times 14 \times 14 \times 1$

$* \ 5 \times 5 \times 5 \times 1$

16 filters

$\rightarrow 10 \times 10 \times 10 \times 16$

$* \ 5 \times 5 \times 5 \times 16$

32 filters

$\rightarrow 6 \times 6 \times 6 \times 32$