# Copyright Notice

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

64x64 × 3

Cat? (0/1)

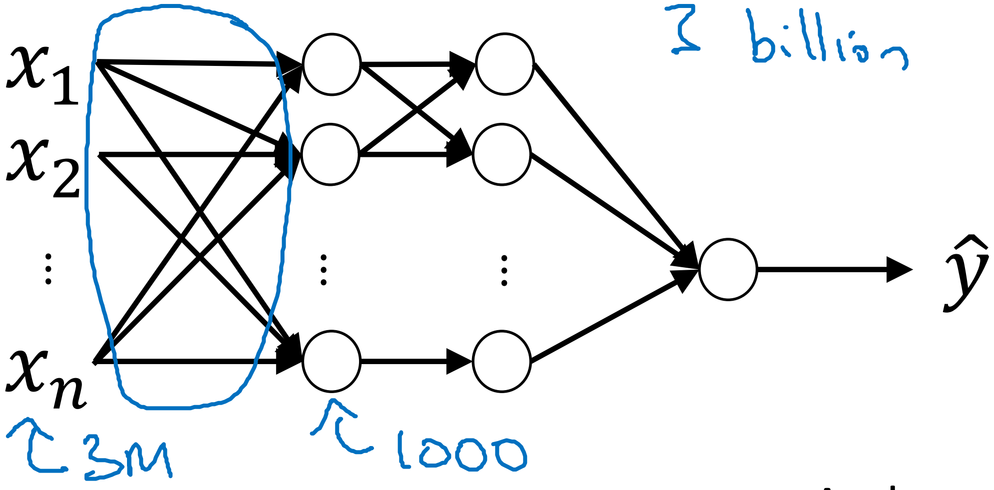
12288

1000×1000×3
= 3 million

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

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

3 billion



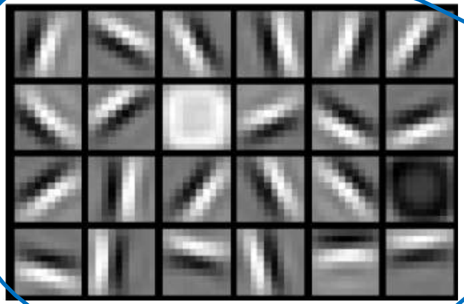

Andrew Ng

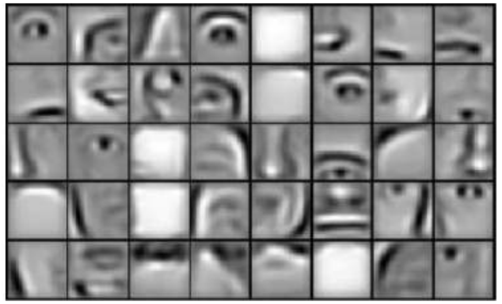
deeplearning.ai

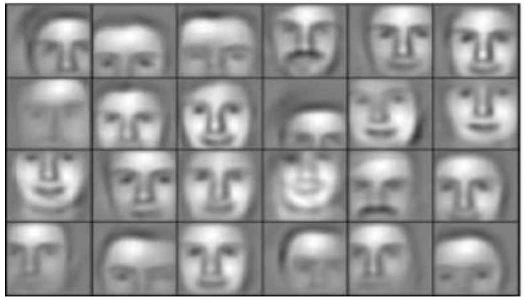
# Convolutional Neural Networks

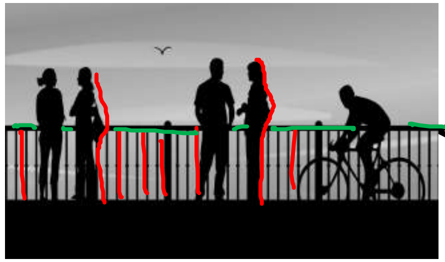
## Edge detection example

# Computer Vision Problem

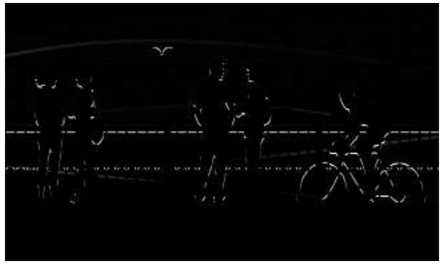








vertical edges



horizontal edges

# Vertical edge detection

$3\times1 + 1\times1 + 2\times1 + 0\times0 + 5\times0 + 7\times0 + 1\times-1 + 8\times-1 + 2\times-1 = -5$

| | | | | | |
|---|---|---|---|---|---|
| 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

=

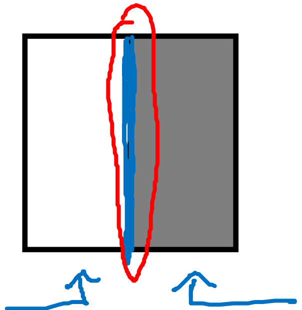
| | | | |
|---|---|---|---|
| -5 | -4 | 0 | 8 |
| -10 | -2 | 2 | 3 |
| 0 | -2 | -4 | -7 |
| -3 | -2 | -3 | -16 |

4×4

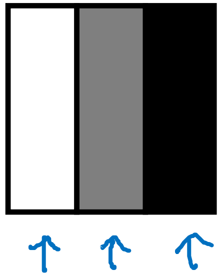
Andrew Ng

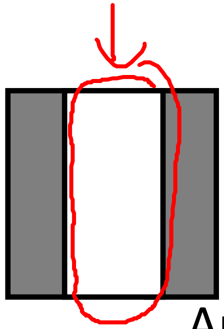
# Vertical edge detection

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



*





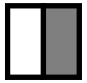
deeplearning.ai

# Convolutional Neural Networks

## More edge detection

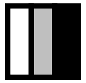
# Vertical edge detection examples

| | | | | | |
|---|---|---|---|---|---|
| 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 |



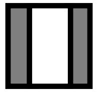
$*$

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |



$=$

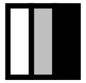
| | | | |
|---|---|---|---|
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |
| 0 | 30 | 30 | 0 |



| | | | | | |
|---|---|---|---|---|---|
| 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 |

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 |

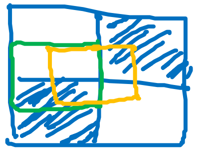
6 x 6

*

| | | |
|---|---|---|
| 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

| | | |
|---|---|---|
| 1 | 0 | -1 |
| 1 | 0 | -1 |
| 1 | 0 | -1 |

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

Sobel filter

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

Scharr filter

| | | | | | |
|---|---|---|---|---|---|
| 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 |

convolution

$*$

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

3x3

these parameters can also be learnt vs. hard coding, allowing NN to learn various edges (low level features)

=

45°
70°
73°

| | | | |
|---|---|---|---|
| | | | |
| | | | |
| | | | |
| | | | |

Andrew Ng

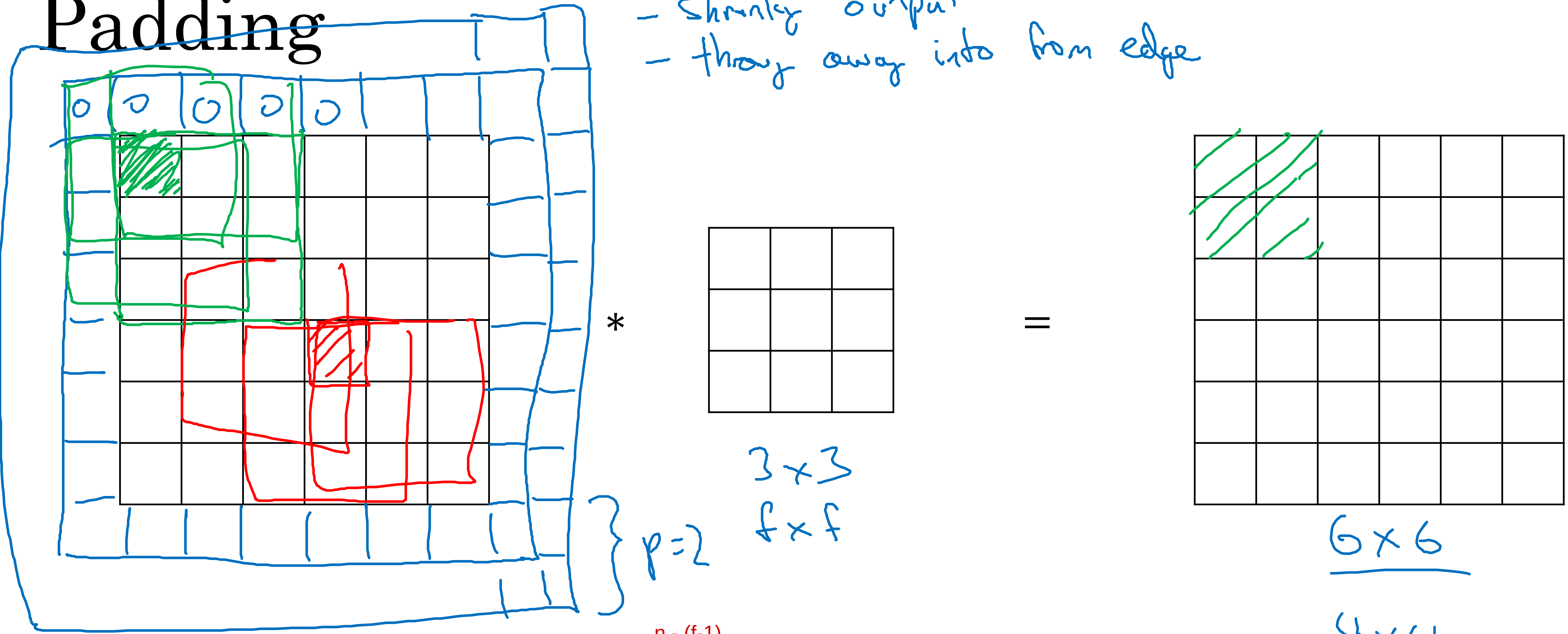
# Convolutional Neural Networks

deeplearning.ai

## Padding

# Padding

- shrinky output
- throw away info from edge



$3 \times 3$

$f \times f$

$p=2$

$6 \times 6$

$\cancel{4 \times 4}$

$6 \times 6 \rightarrow 8 \times 8$

$n \times n$

n - (f-1)

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

$6-3+1=4$

$p = \text{padding} = 1$

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

$6+2-3+1 \times \_\_ = 6 \times 6$

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.

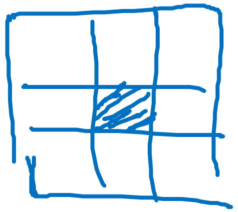
$n+2p-f+1 \times n+2p-f+1$

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

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

f is usually odd

$1 \times 1$
$3 \times 3$
$5 \times 5$
$7 \times 7$



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 |

$7 \times 7$

$*$

| | | |
|---|---|---|
| 3 | 4 | 4 |
| 1 | 0 | 2 |
| -1 | 0 | 3 |

$3 \times 3$

$=$

| | | |
|---|---|---|
| 91 | 100 | 83 |
| 69 | 91 | 127 |
| 44 | 72 | 74 |

$3 \times 3$

stride $= 2$

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

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

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:

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

Andrew Ng

# Technical note on cross-correlation vs. convolution

In Maths:
- convolution = flipping the matrix and then multiplying
- multiplying step is called cross-correlation

In ML - we skip the flipping step, but still call it convolution (and not cross-correlation)

#tradition

## 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 |

flipped across horizontal & vertical should be:
7 9 -1
2 0 1
5 4 3

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

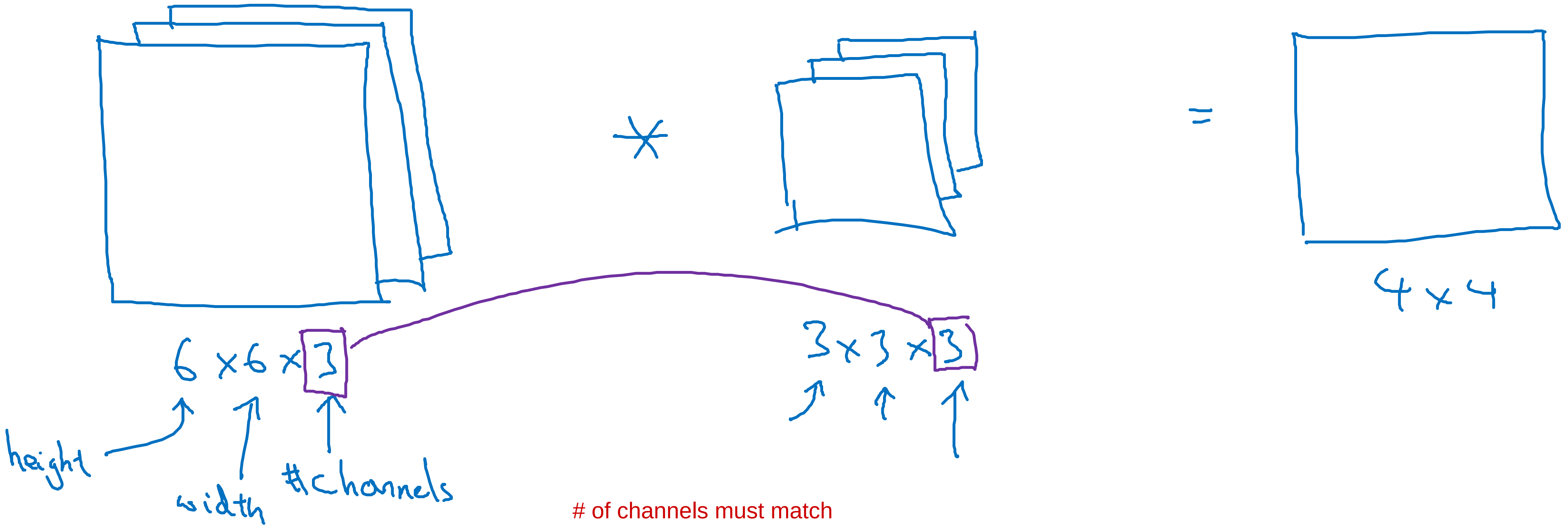
Andrew Ng

deeplearning.ai

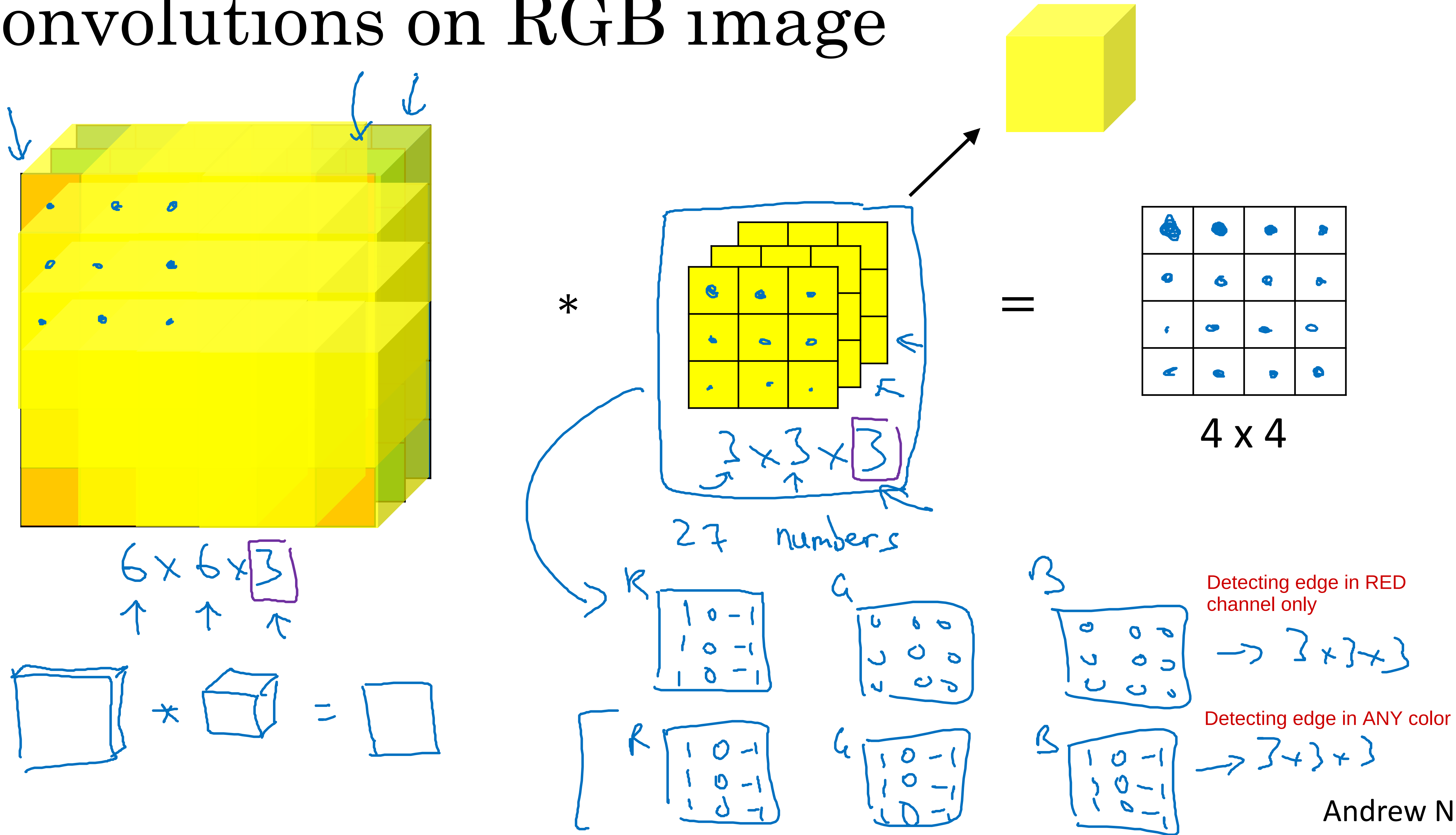
# Convolutional Neural Networks

## Convolutions over volumes

# Convolutions on RGB images

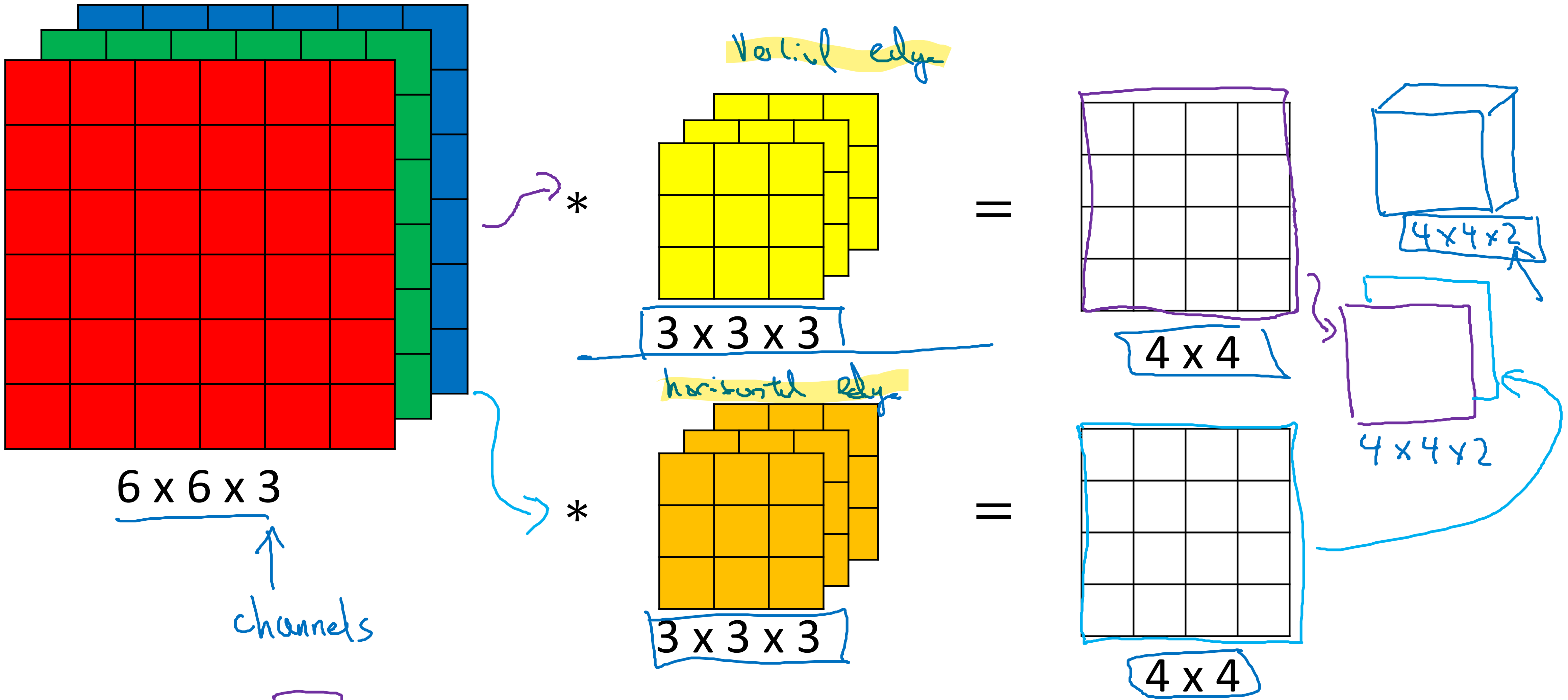


# Convolutions on RGB image




Andrew Ng

# Multiple filters




Summary: $n \times n \times \boxed{n_c} \ * \ f \times f \times \boxed{n_c} \ \longrightarrow \ \underset{4}{n-f+1} \times \underset{4}{n-f+1} \times \underset{2}{n_c'}$ ← #filters

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

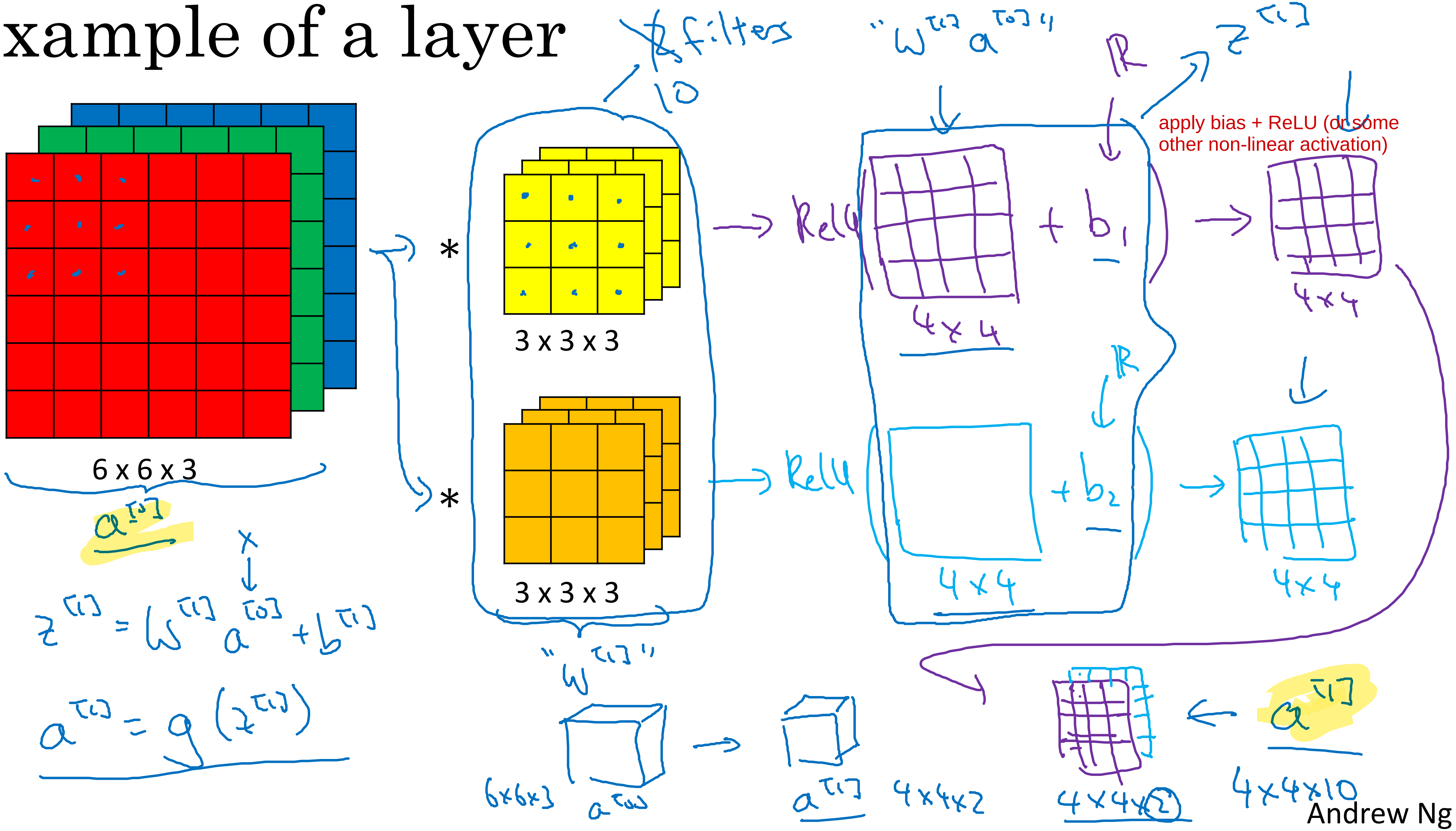
Andrew Ng

deeplearning.ai

# Convolutional Neural Networks

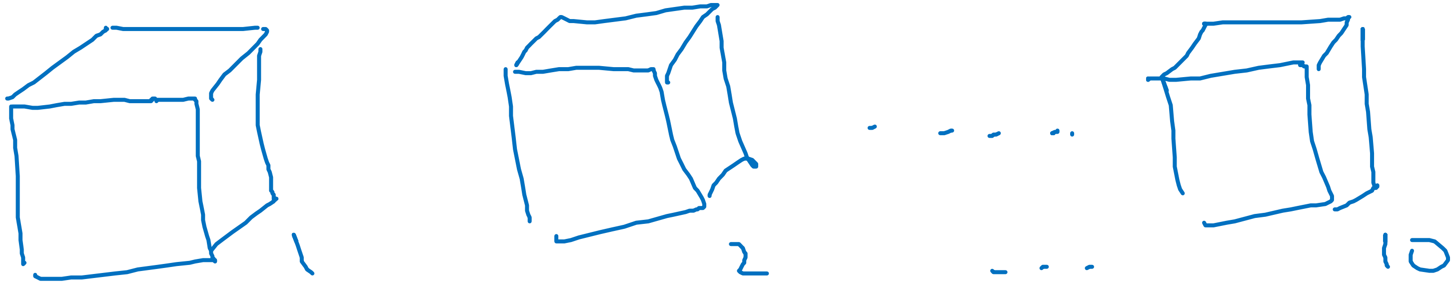
## One layer of a convolutional network

# Example of a layer



# Number of parameters in one layer

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




3x3x3

27 parameters.

+ bias

→ 28 parameters.

280 parameters.

Andrew Ng

# Summary of notation

## If layer l is a convolution layer:

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

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

→ Each filter is: $f^{[l]} \times f^{[l]} \times \boxed{n_c^{[l-1]}}$

Activations: $a^{[l]} \to n_H^{[l]} \times n_W^{[l]} \times n_c^{[l]}$

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

bias: $n_c^{[l]}$ — $(1,1,1,n_c^{[l]})$

← #filters in layer l.

Input: $n_H^{[l-1]} \times n_W^{[l-1]} \times \boxed{n_c^{[l-1]}}$ ←

Output: $n_H^{[l]} \times n_W^{[l]} \times \boxed{n_c^{[l]}}$ ←

$$n_{\cancel{H}W}^{[l]} = \left\lfloor \frac{n_{\cancel{H}W}^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor$$

# of samples is the first dimension now

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

$n_c \times n_H \times n_W$

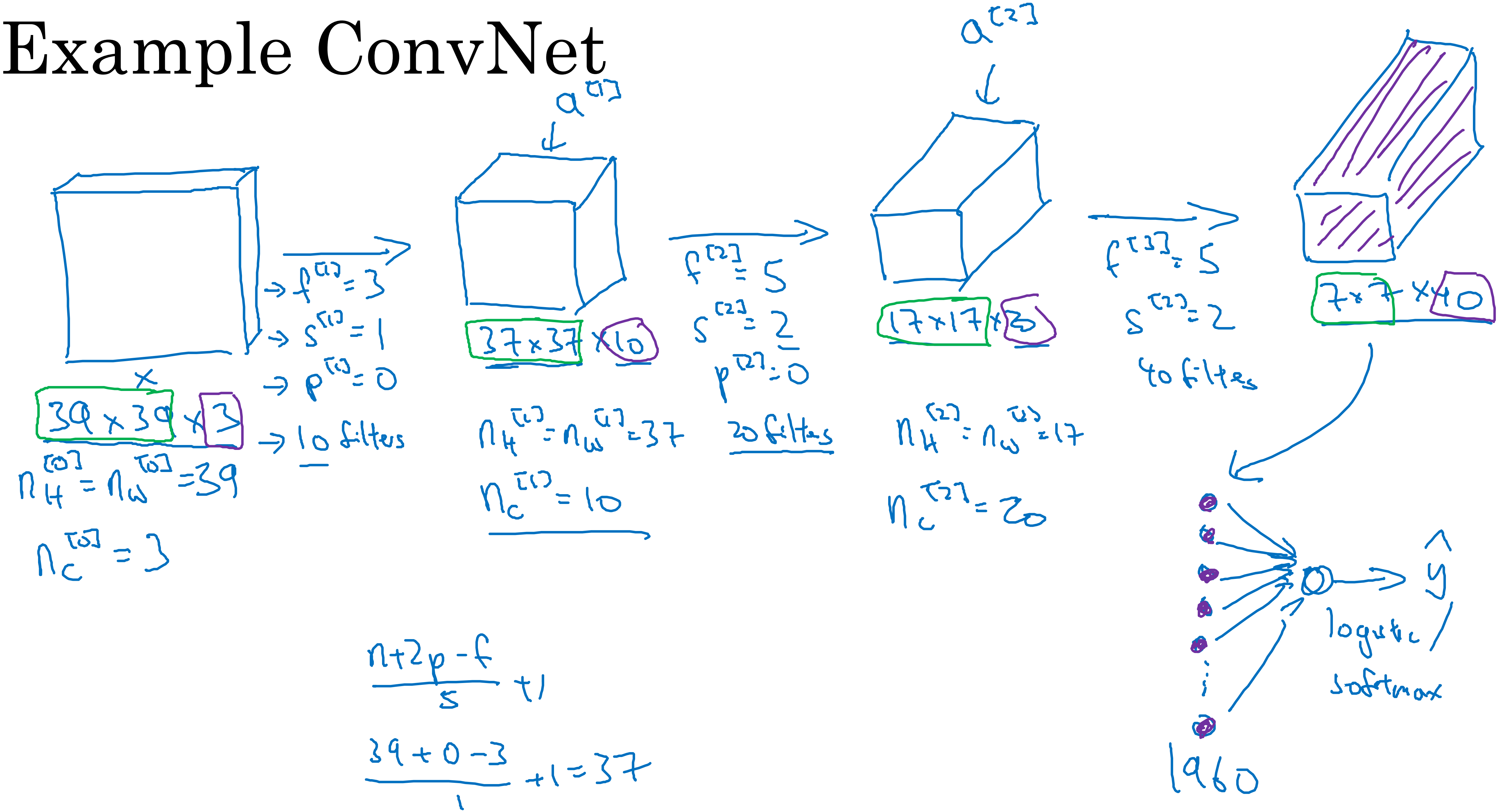
Andrew Ng

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)

# Convolutional Neural Networks

deeplearning.ai

## 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

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 2 | 1 | 3 |
| 2 | 9 | 1 | 1 | 5 |
| 1 | 3 | 2 | 3 | 2 |
| 8 | 3 | 5 | 1 | 0 |
| 5 | 6 | 1 | 2 | 9 |

$5 \times 5 \times \cancel{3}\ n_c$

| | | |
|---|---|---|
| 9 | 9 | 5 |
| 9 | 9 | 5 |
| 8 | 6 | 9 |

$3 \times 3 \times \cancel{3}\ n_c$

$$\boxed{\begin{array}{c} f = 3 \\ \hline s = 1 \end{array}}$$

max pooling calculation is done independently on each channel

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

# Pooling layer: Average pooling

$$\begin{bmatrix} 1 & 3 & 2 & 1 \\ 2 & 9 & 1 & 1 \\ 1 & 4 & 2 & 3 \\ 5 & 6 & 1 & 2 \end{bmatrix} \longrightarrow \begin{bmatrix} 3.75 & 1.25 \\ 4 & 2 \end{bmatrix}$$

$f = 2$

$s = 2$

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

# Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

$f=2, s=2$

$f=3, s=2$

$\rightarrow$ ~~p: padding.~~

No parameters to learn!

$n_H \times n_W \times n_C$

$\downarrow$

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

$\times n_C$

Andrew Ng

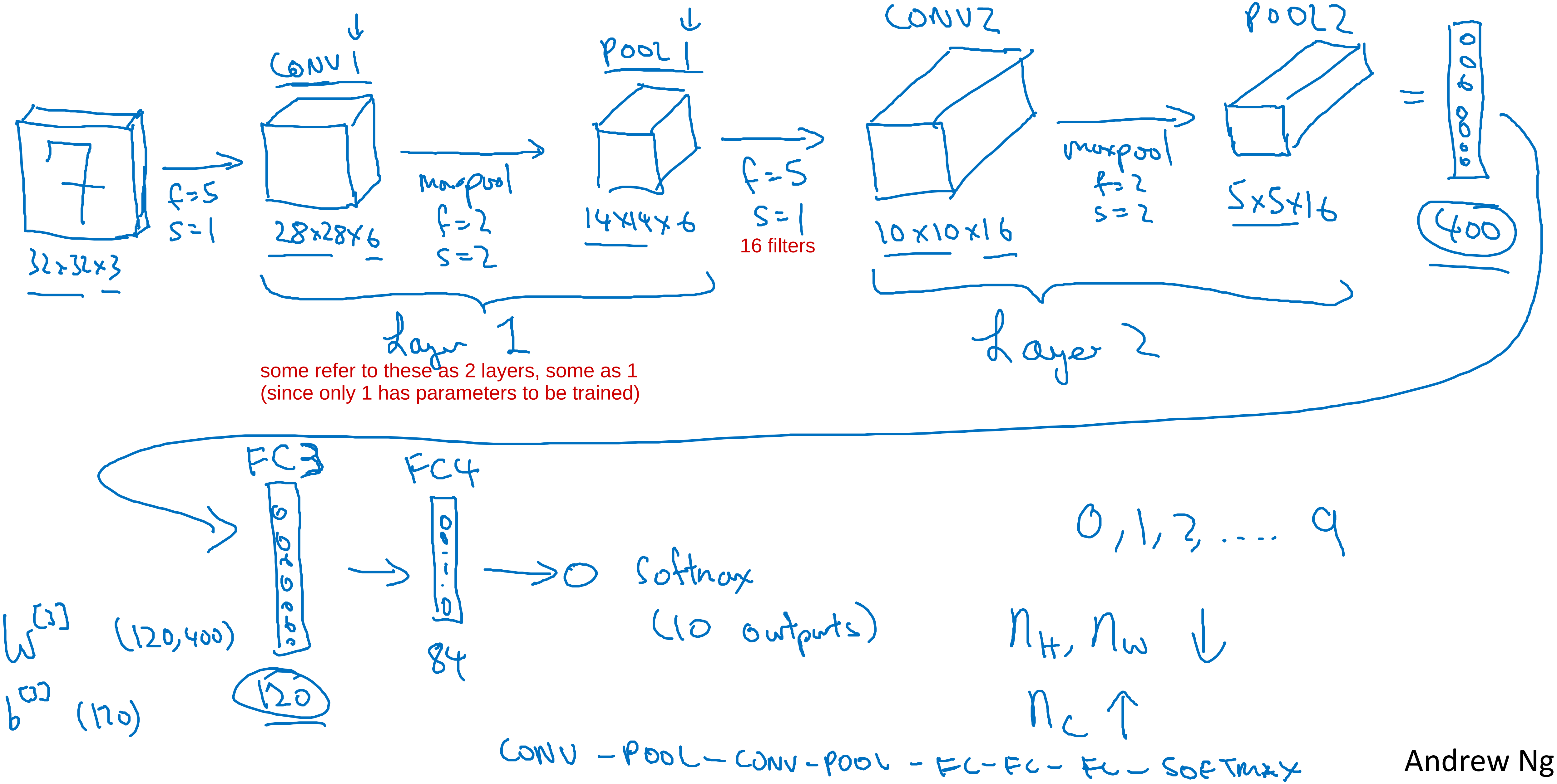
deeplearning.ai

# Convolutional Neural Networks

---

## Convolutional neural network example

# Neural network example (LeNet-5)



# Neural network example

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

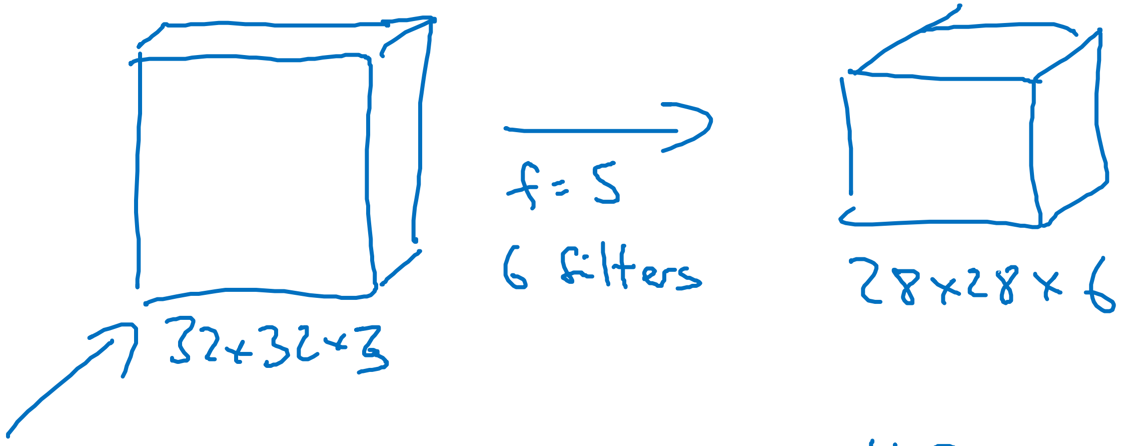
Andrew Ng

deeplearning.ai

# Convolutional Neural Networks

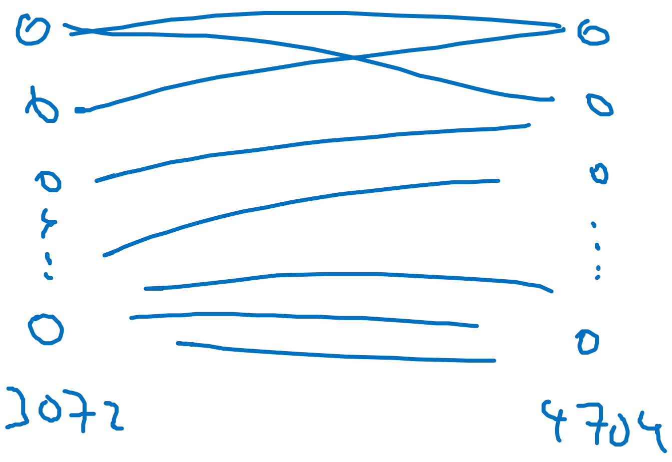
## Why convolutions?

# Why convolutions




3,072

4,704




5×5 − 25

26

~~6 × 26~~ = ~~156~~ parameters

6 * (5*5*3 +1) = 456

+ Parameter sharing
+ Sparcity of connections

3,072 × 4,704 ≈ 14M

Andrew Ng

# Why convolutions

translation invariance

Image shifted a few pixels over should still results in similar features and output matrix

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

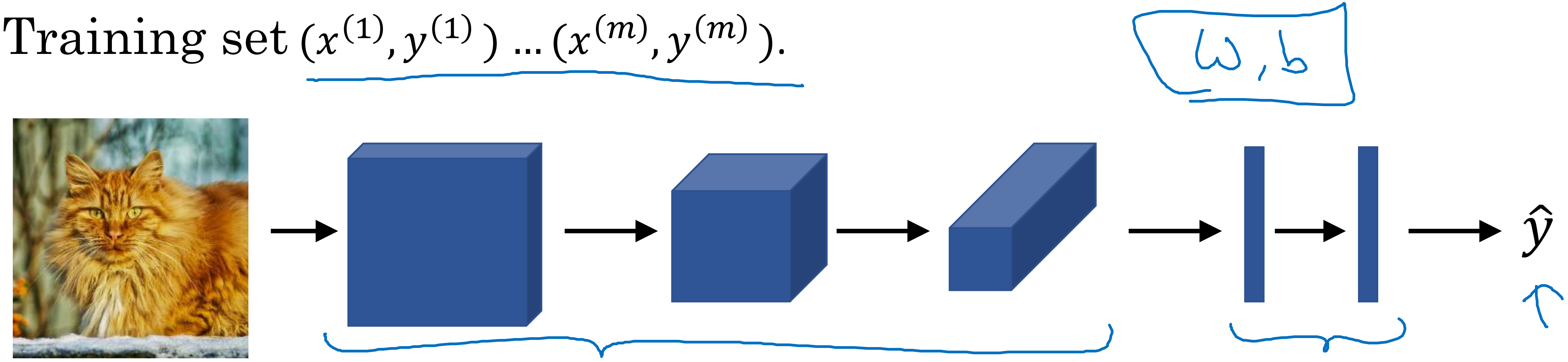
**Parameter sharing:** A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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

Andrew Ng

# Putting it together

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




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

Use gradient descent to optimize parameters to reduce $J$

# Copyright Notice

deeplearning.ai

# Case Studies

## Why look at case studies?

# Outline

## Classic networks:

- LeNet-5
- AlexNet
- VGG

ResNet (152)

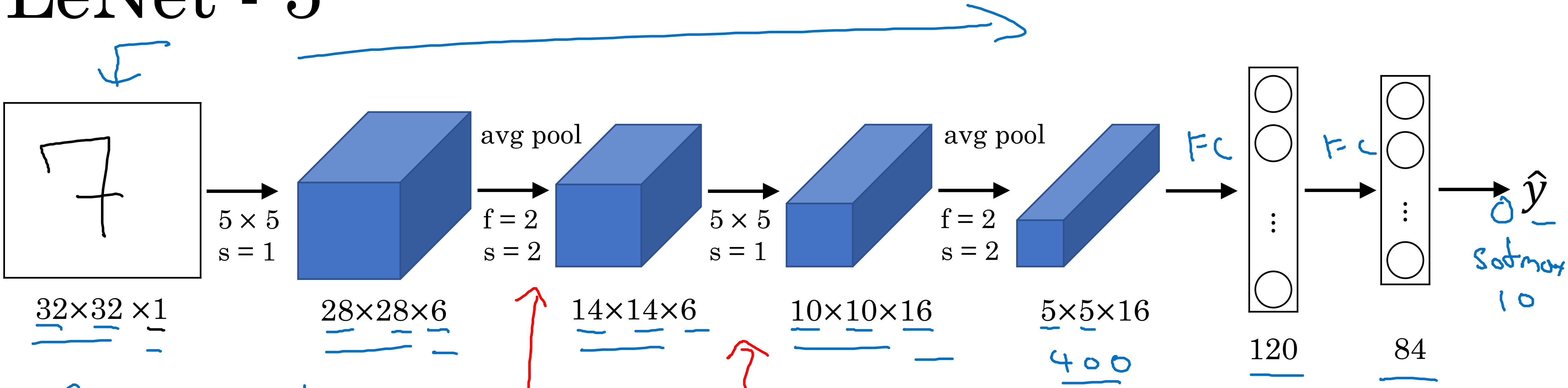
Inception

deeplearning.ai

# Case Studies

## Classic networks

# LeNet - 5




60K parameters.

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

conv pool conv pool fc fc output

non-linearity after pooling

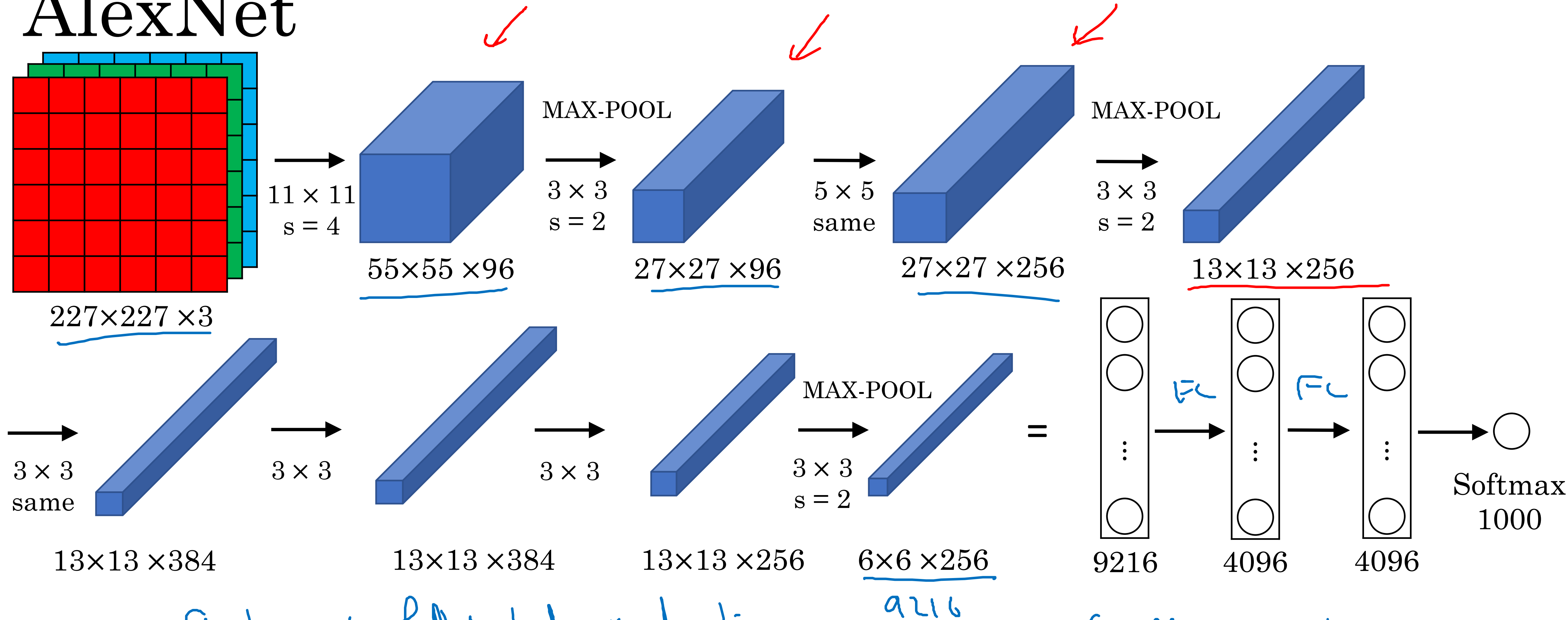
$n_H \times n_W \times n_C$ $f \times f \times n_C$

Advanced: sigmoid/tanh ReLU

$\boxed{\text{II}}$ III.

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

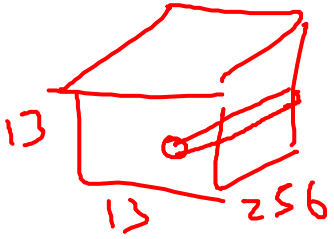
# AlexNet




- Similar to LeNet, but much bigger.

~60M parameters

- ReLU
- Multiple GPUs.
- Local Response Normalization (LRN)



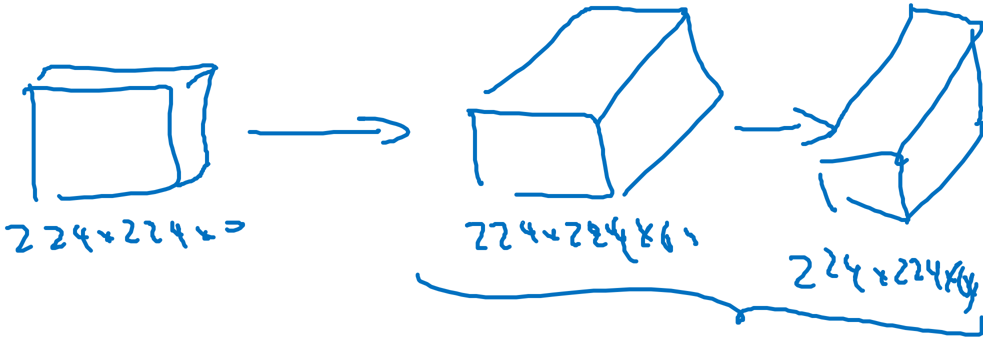

[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

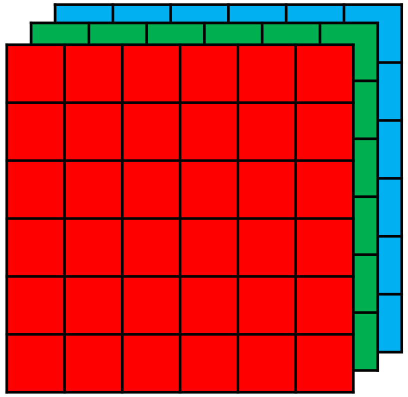
# VGG - 16

VGG-19

Use same filter config

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]

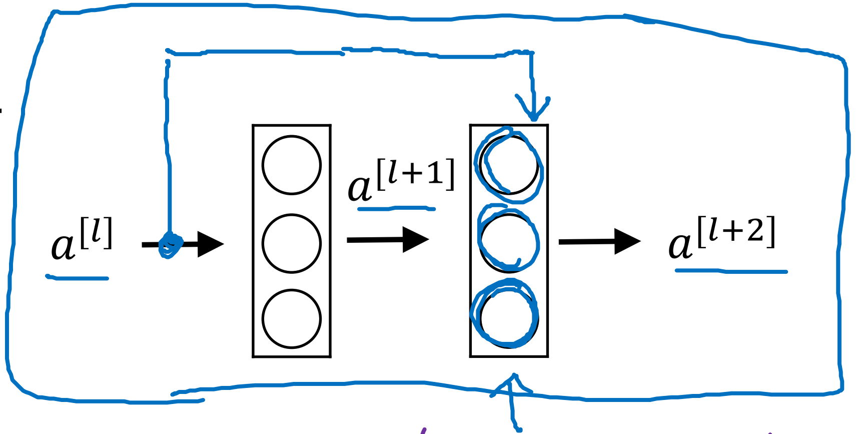
Andrew Ng

deeplearning.ai

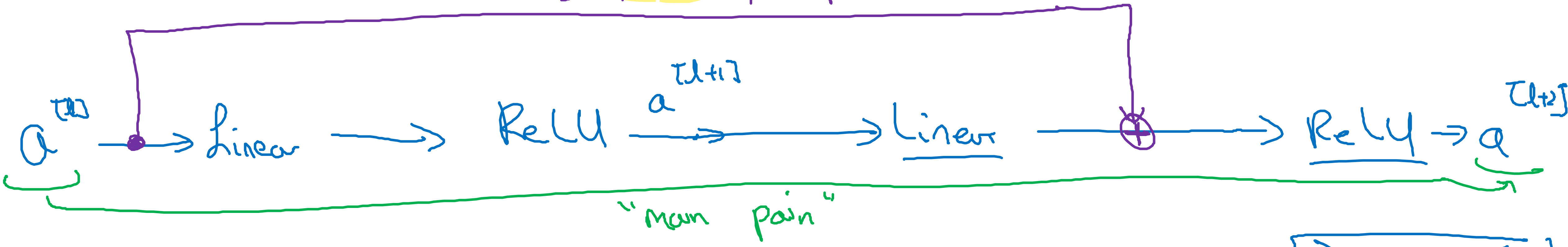
# Case Studies

# Residual Networks (ResNets)

# Residual block




"short cut" / skip connection




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

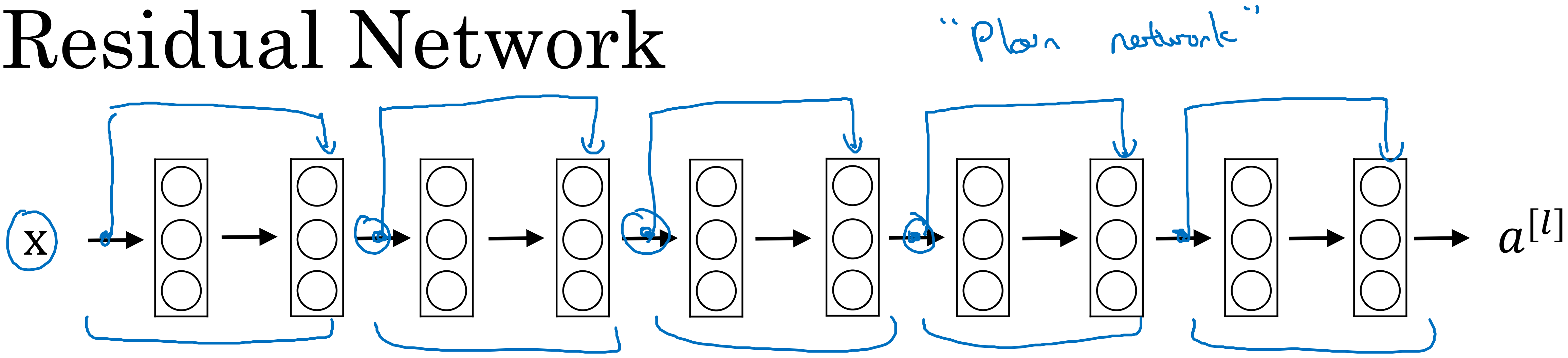
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

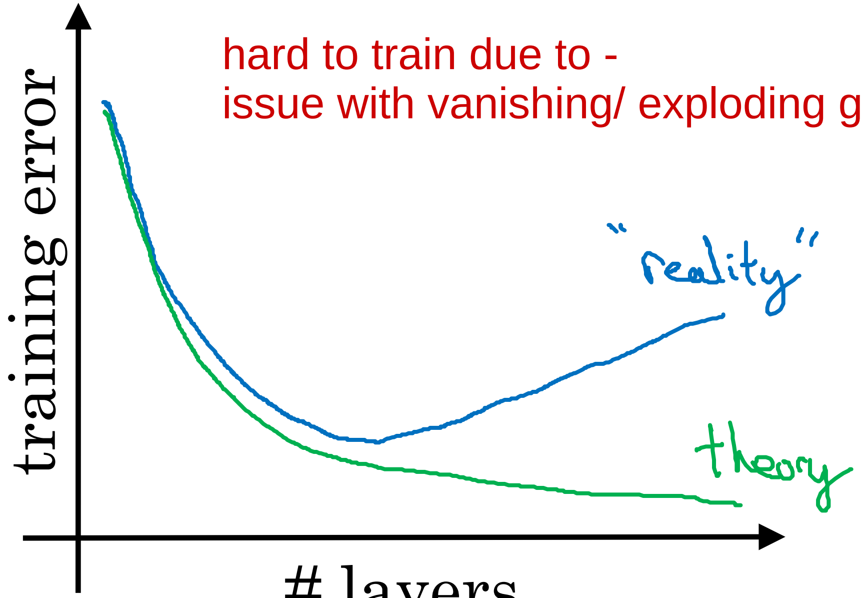
'a' added to 'z'
dimensions are kept same

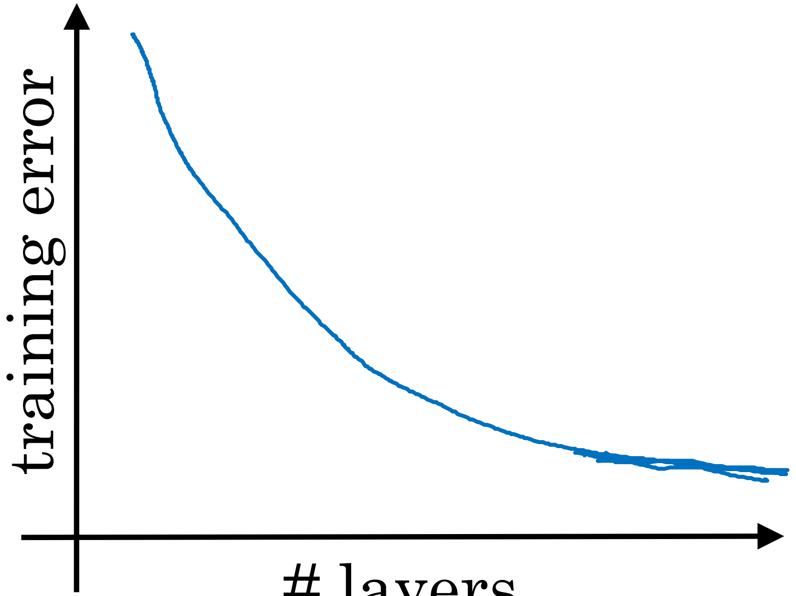
[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

# Residual Network




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

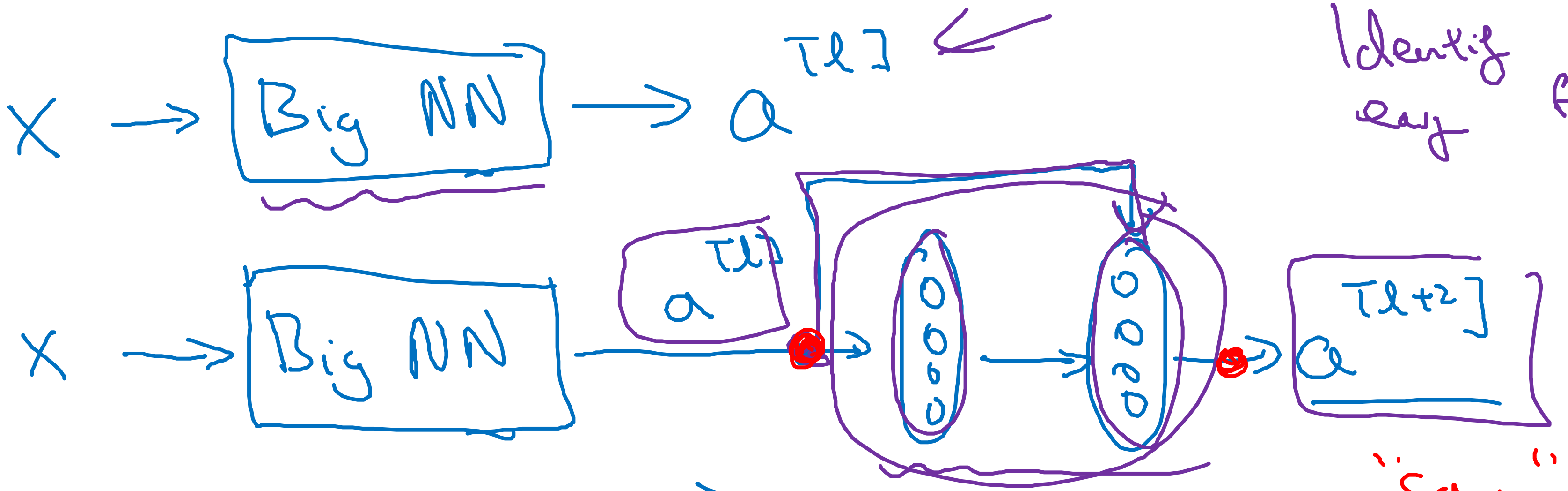
Andrew Ng

deeplearning.ai

# Case Studies

## Why ResNets work

# Why do residual networks work?




Identity function is easy for Residual block to learn!

Neural Net can 'easily' learn to just be an identify, i.e. a(L+2) = a (L).. so in deep network they won't deteriorate performance

ReLU. $a \geq 0$

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

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

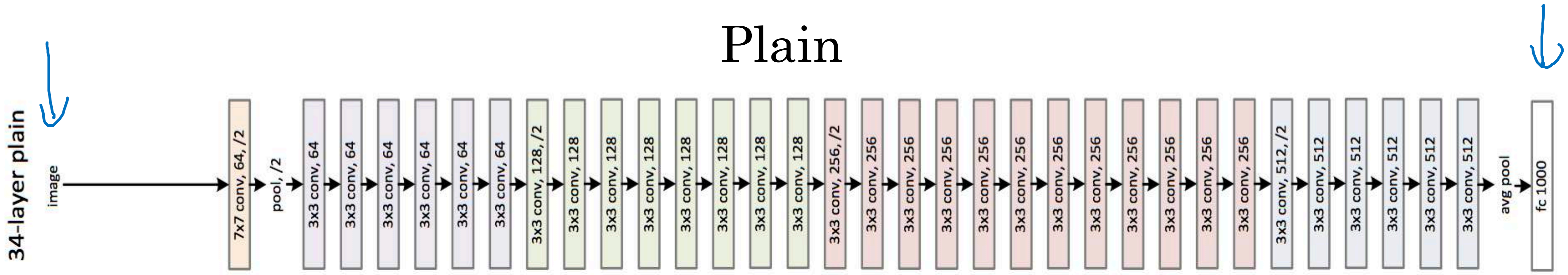
256    128    $\mathbb{R}^{256 \times 128}$

If $W^{[l+2]} = 0$, $b^{[l+2]} = 0$

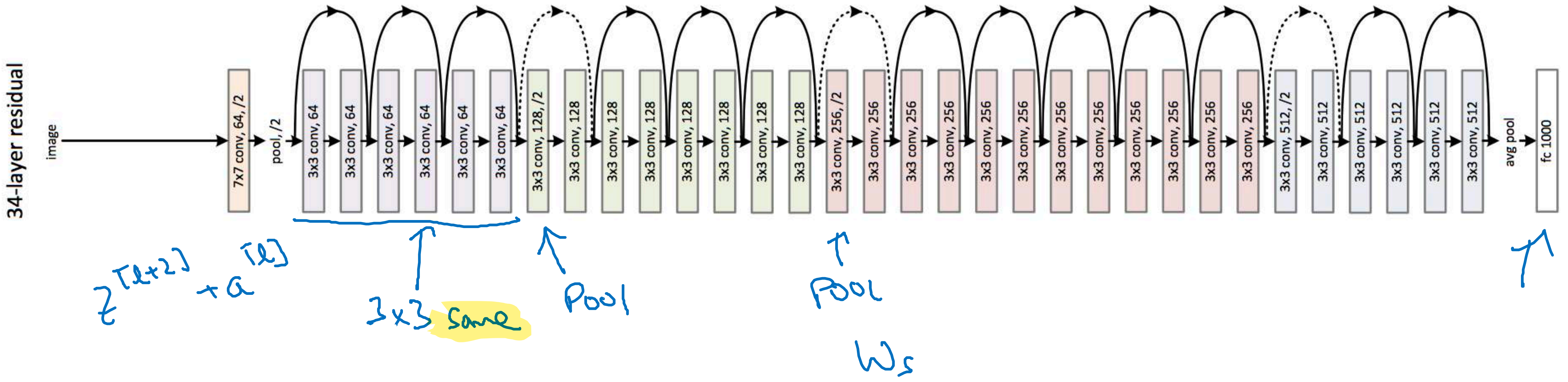
Ws - matrix used in case a(L) and a (L+2) are different dimensions. it can be parameter to be learnt, or padding matrix

# ResNet

## Plain




## ResNet




[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?

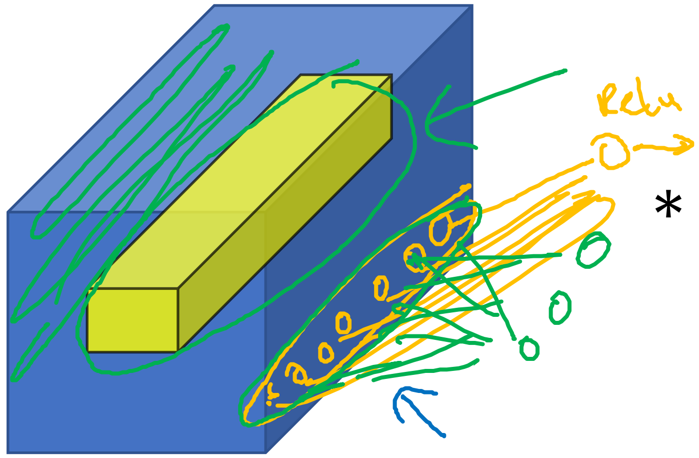
| | | | | | |
|---|---|---|---|---|---|
| 1 | 2 | 3 | 6 | 5 | 8 |
| 3 | 5 | 5 | 1 | 3 | 4 |
| 2 | 1 | 3 | 4 | 9 | 3 |
| 4 | 7 | 8 | 5 | 7 | 9 |
| 1 | 5 | 3 | 7 | 4 | 8 |
| 5 | 4 | 9 | 8 | 3 | 5 |

6 × 6 × 1

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

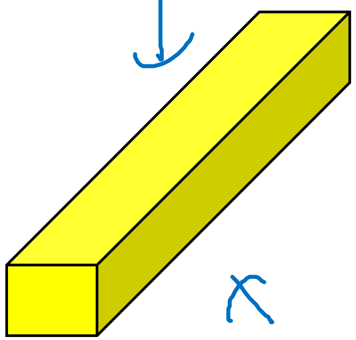
| | | | | | |
|---|---|---|---|---|---|
| 2 | 4 | 6 | ... | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |
| | | | | | |

32 → # filters. $n_c^{[l+1]}$




6 × 6 × 32

$*$

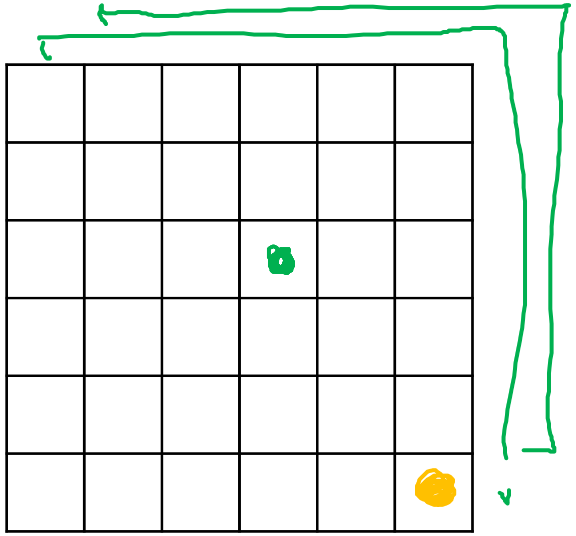


1 × 1 × 32

$=$

ReLU

Network in Network

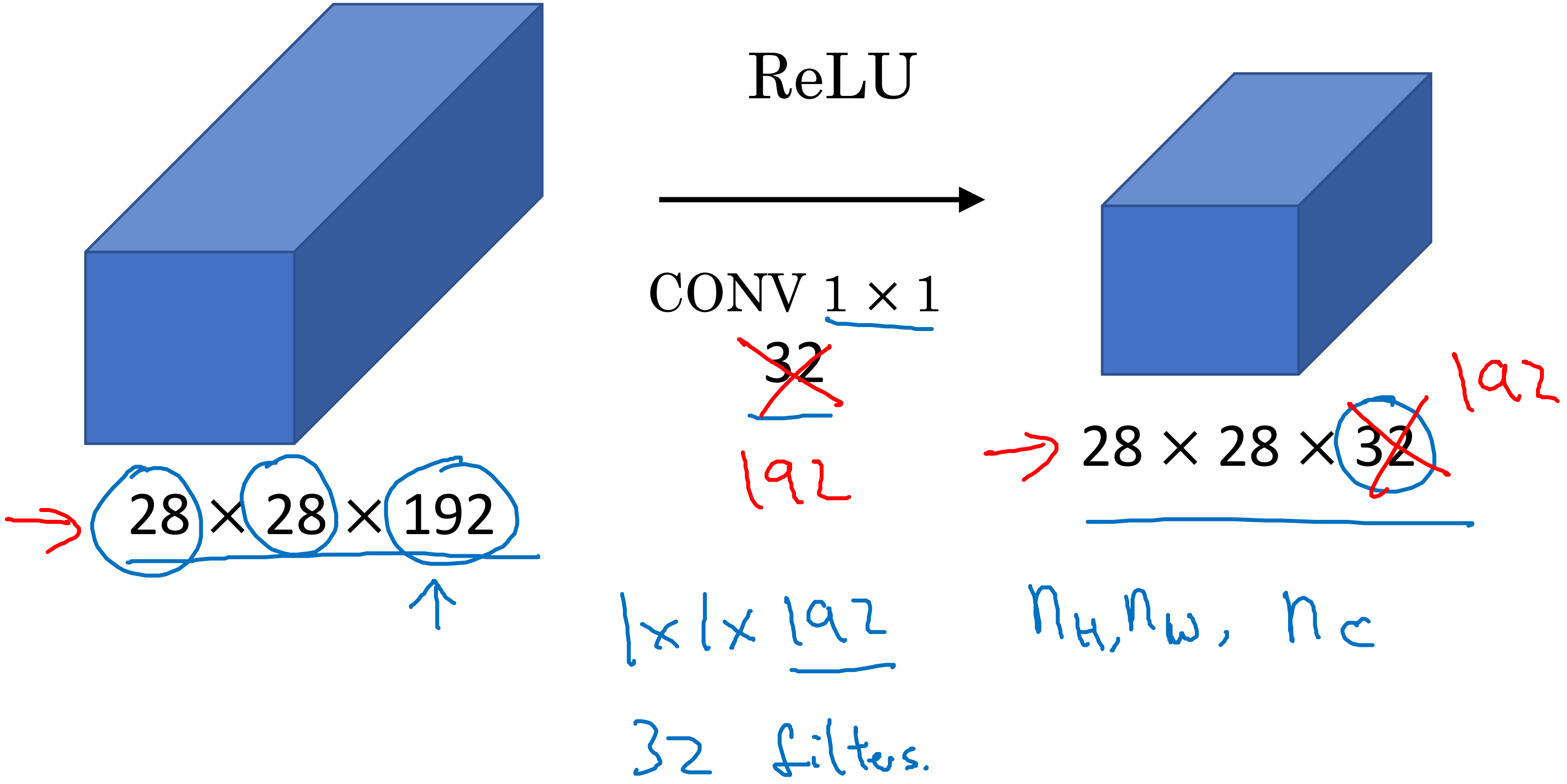


6 × 6 × # filters

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

Andrew Ng

# Using 1×1 convolutions



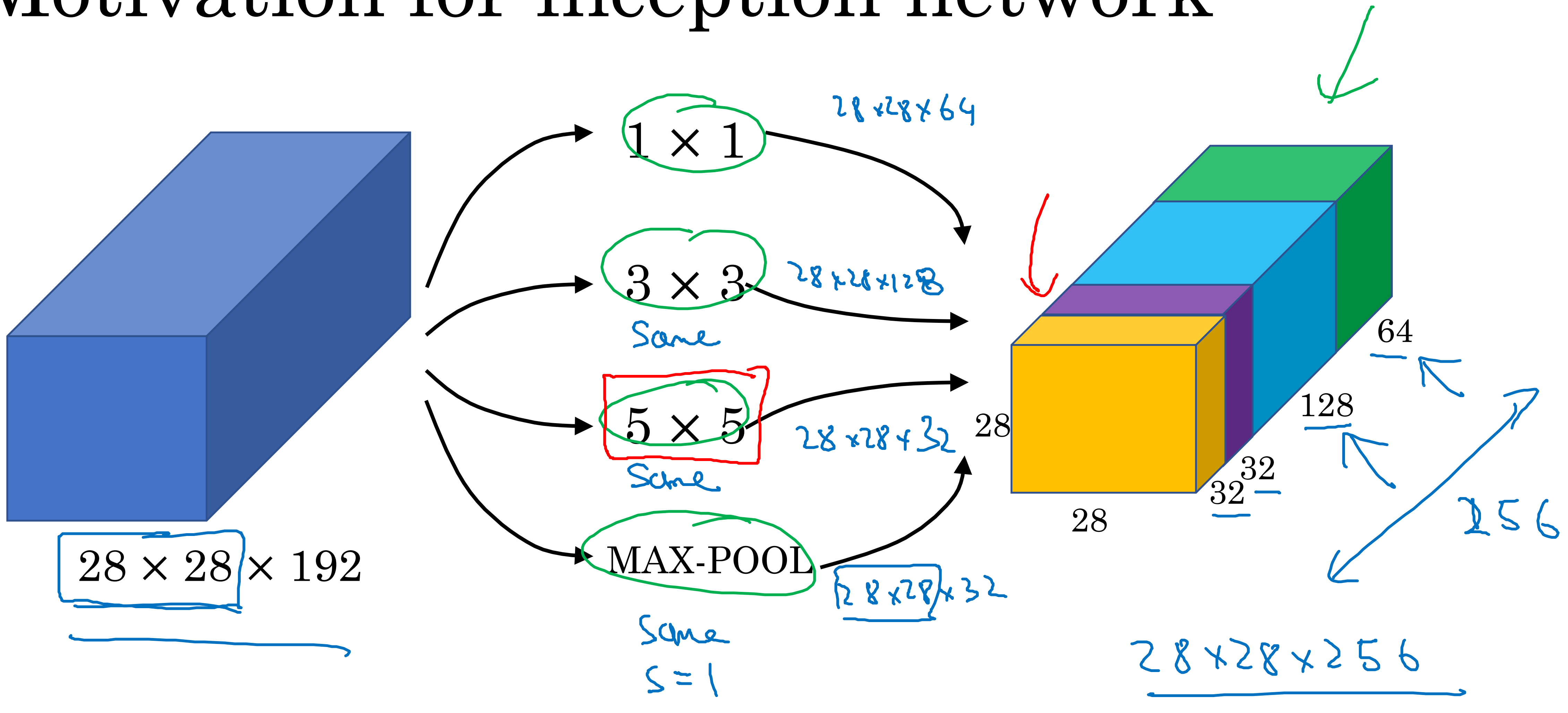

[Lin et al., 2013. Network in network]

deeplearning.ai

# Case Studies

# Inception network motivation

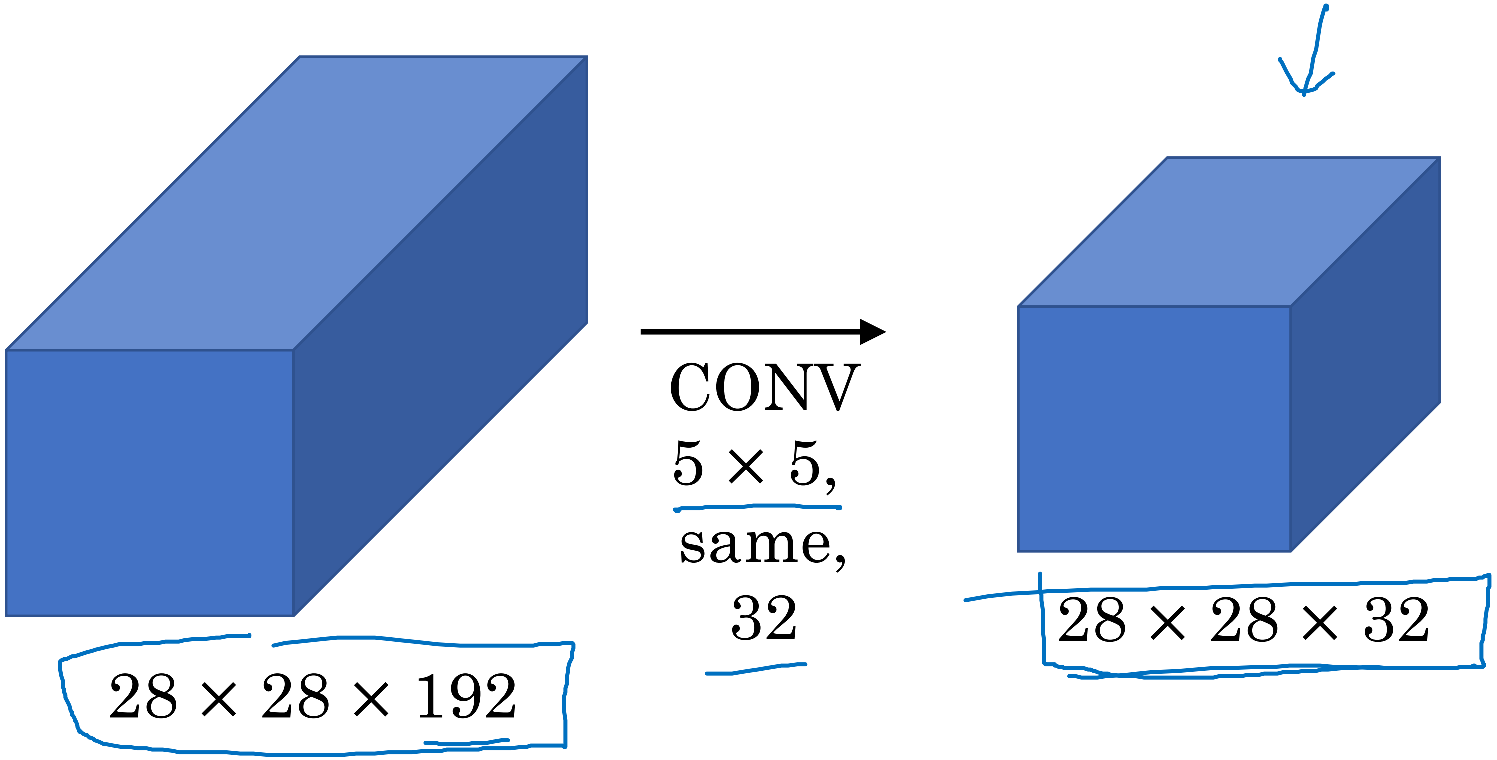
# Motivation for inception network




Combine multiple approaches in a single step
Pooling will require padding

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

# The problem of computational cost




32 filters. filters are $5 \times 5 \times 192$.

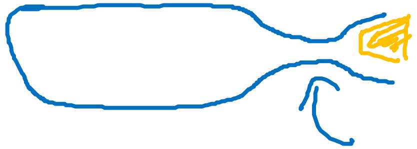
$28 \times 28 \times 32 \times 5 \times 5 \times 192 = 120M$.

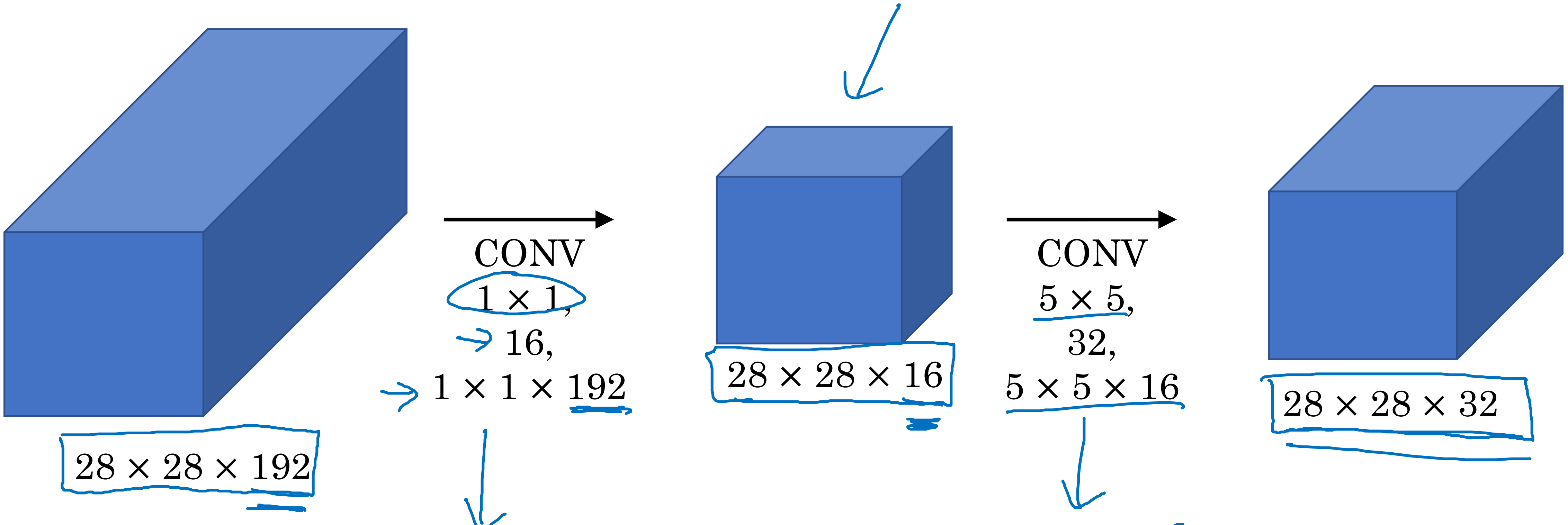
\# of multiplications

Andrew Ng

# Using 1×1 convolution

"bottleneck layer"






28x28x16 × 192 = 2.4M

28x28x32 × 5x5x16 = 10.0M

12.4M

120M

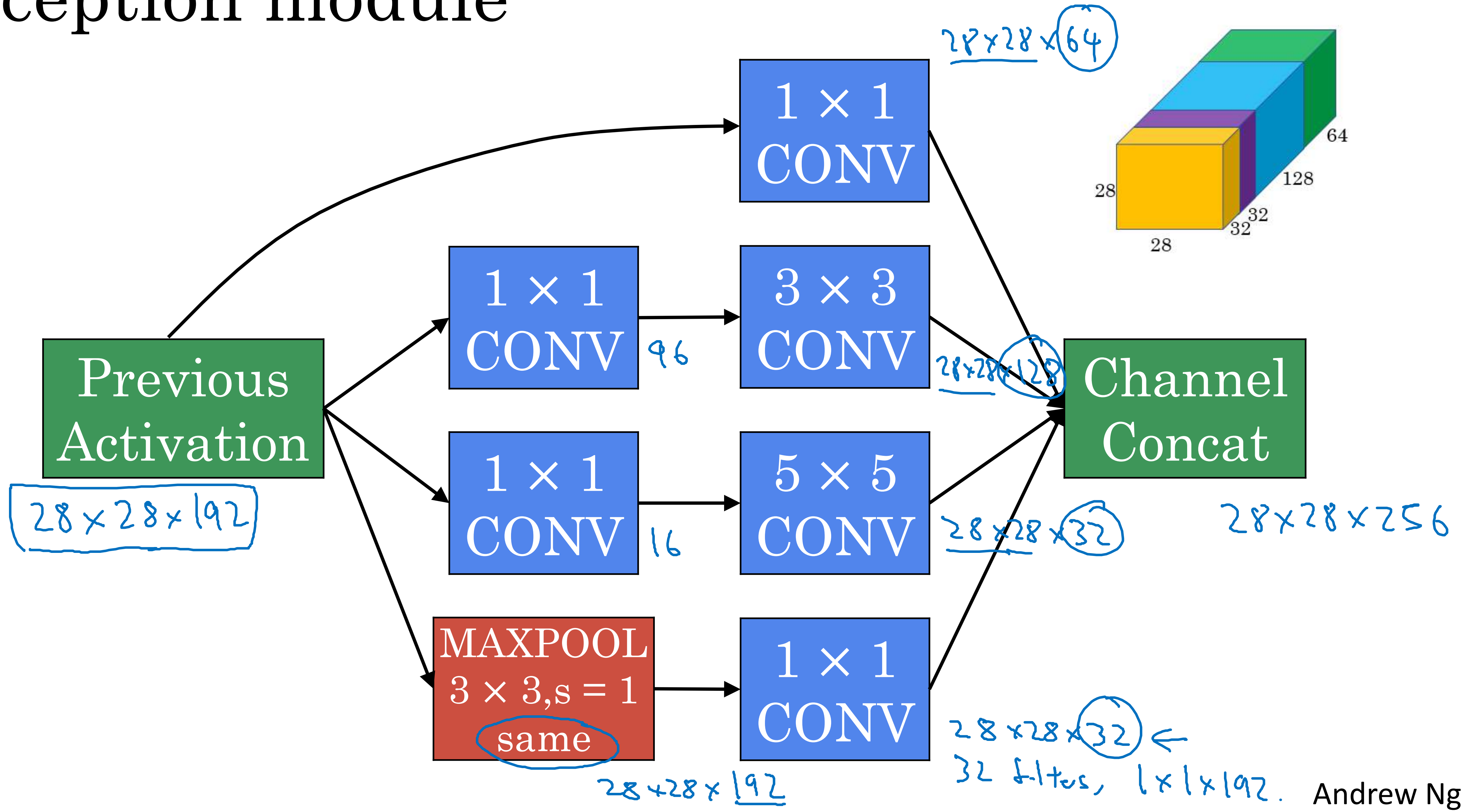
Andrew Ng

deeplearning.ai

# Case Studies

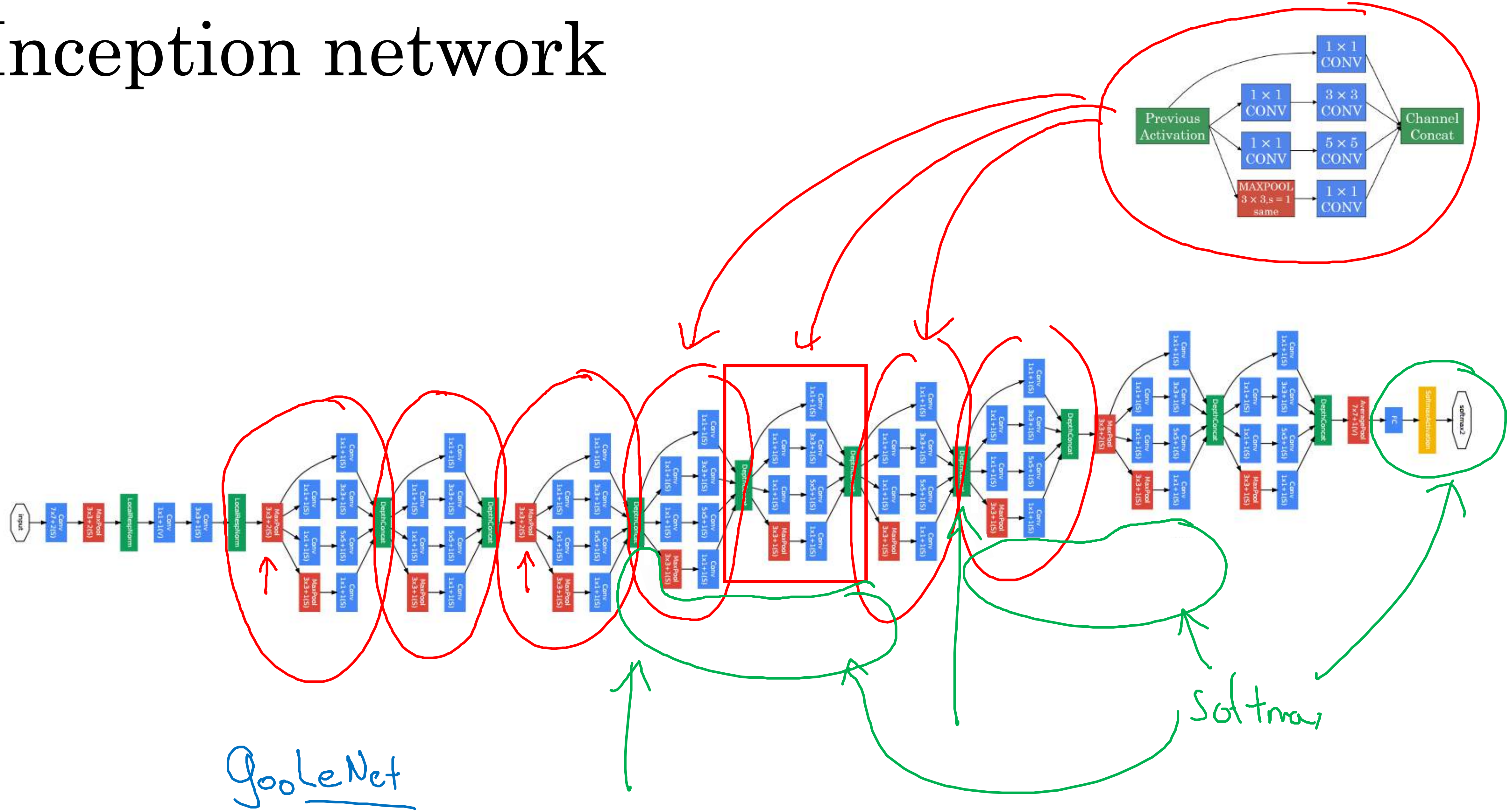
# Inception network

# 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

# Convolutional Neural Networks

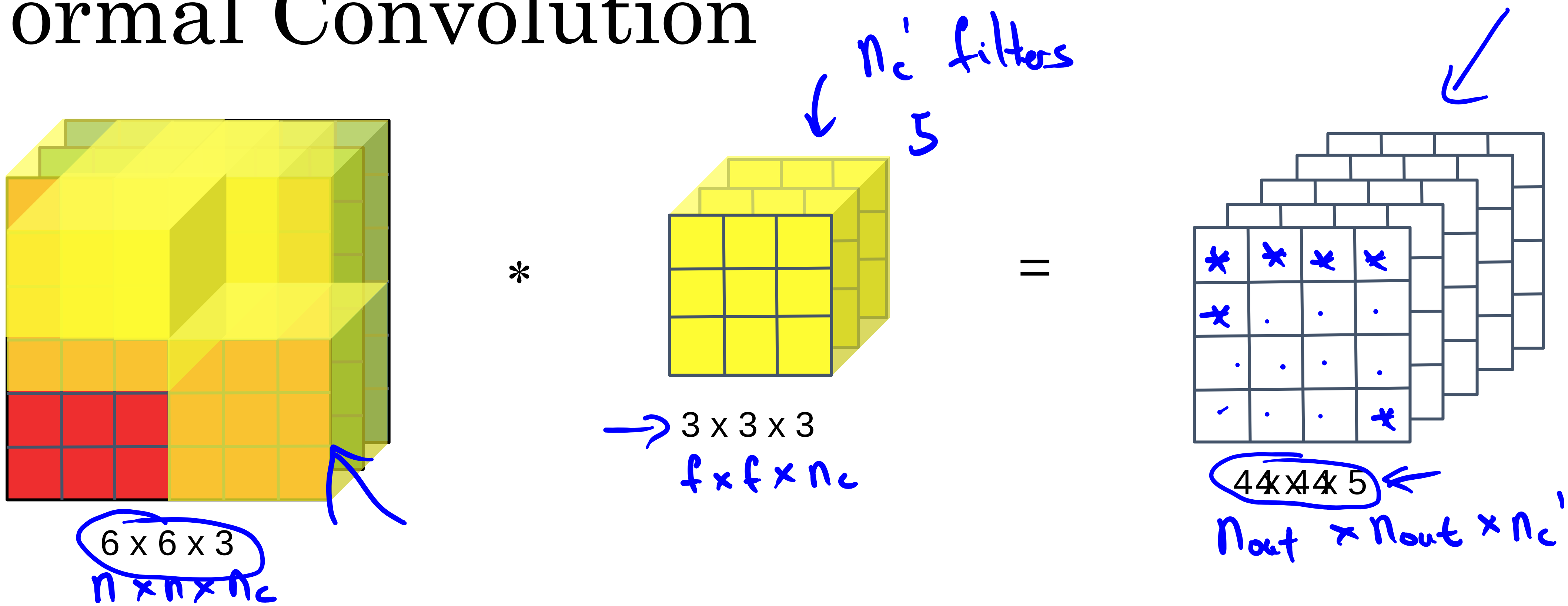
## MobileNet

# Motivation for MobileNets

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


[Howard et al. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications]

# Normal Convolution

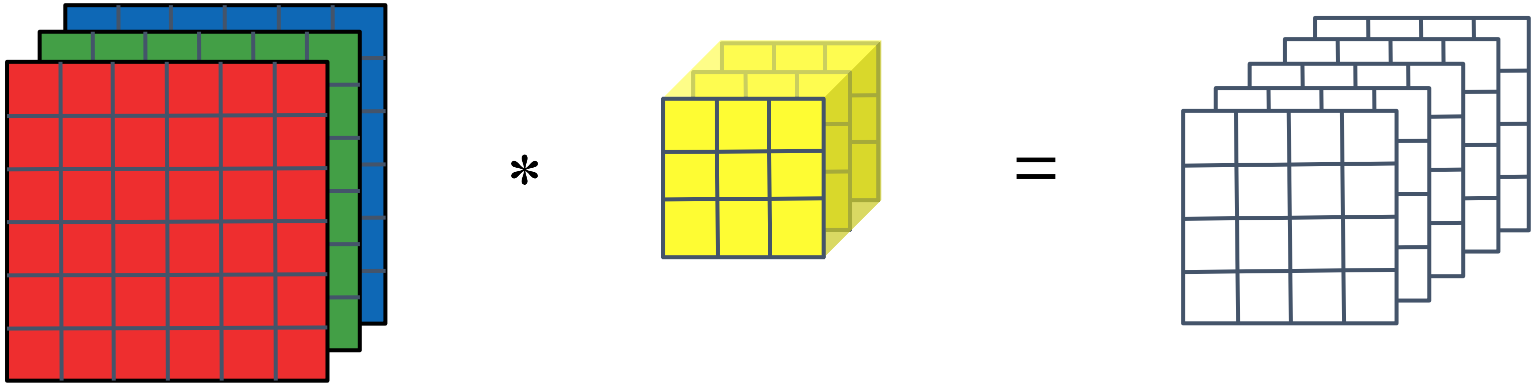



Computational cost = #filter params x # filter positions x # of filters

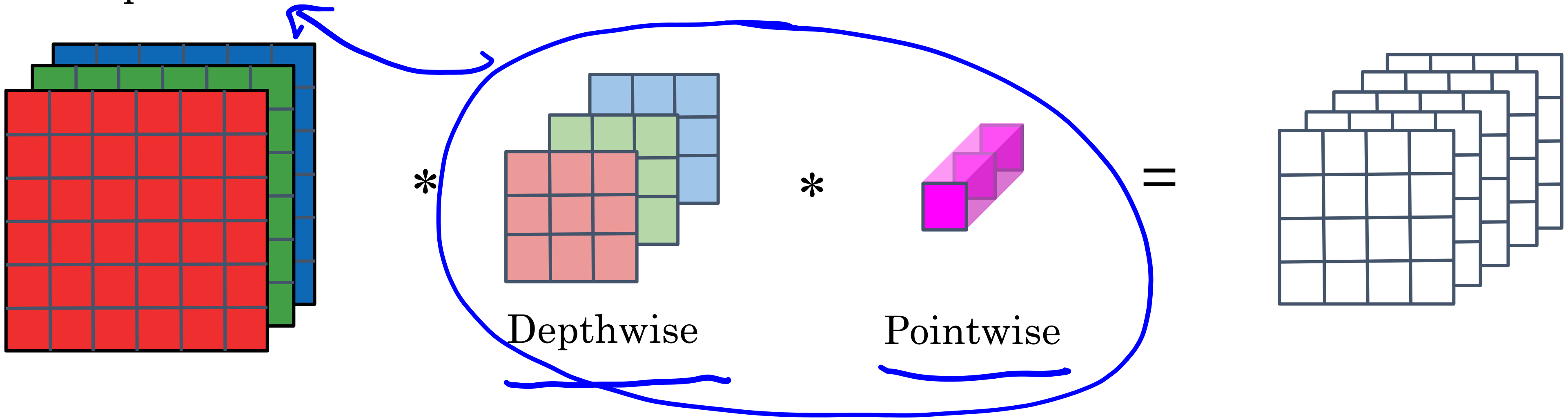
$2160 = 3 \times 3 \times 3 \times 4 \times 4 \times 5$

# Depthwise Separable Convolution

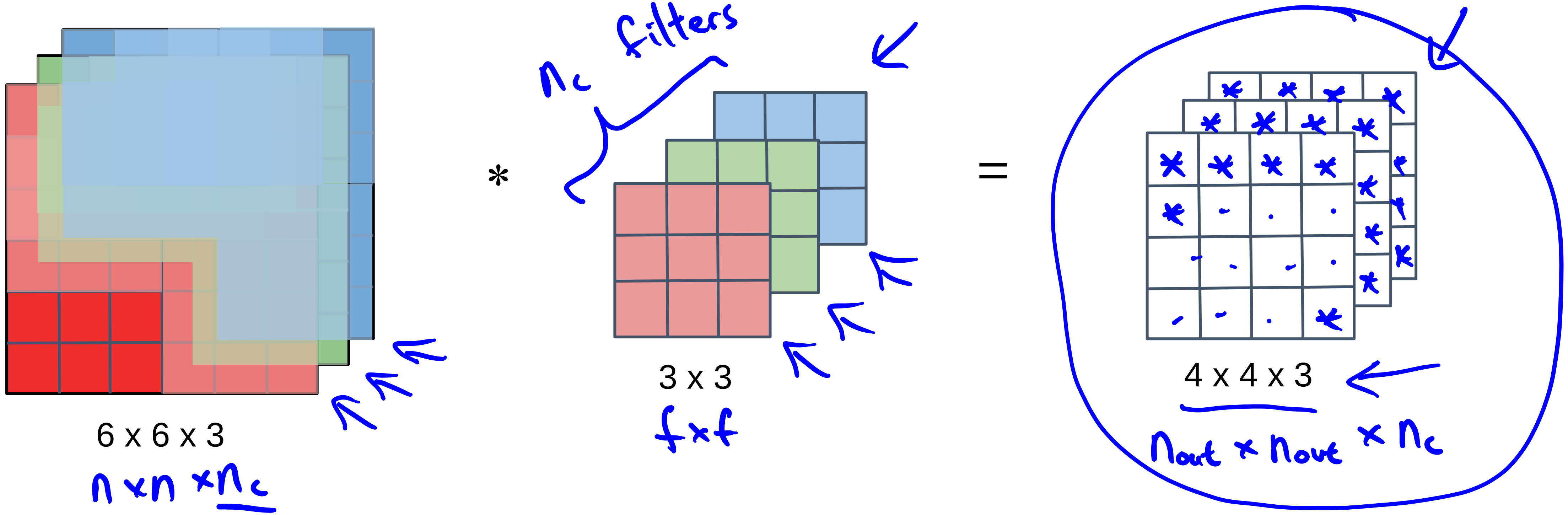
Normal Convolution



Depthwise Separable Convolution



# Depthwise Convolution



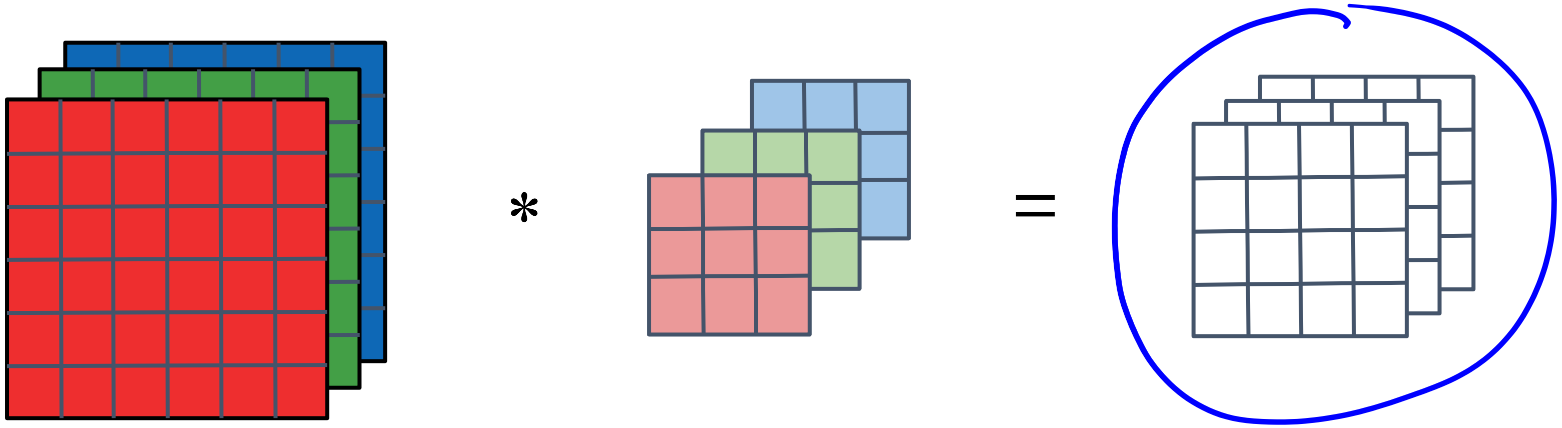

Computational cost = #filter params x # filter positions x # of filters

432 = 3×3 × 4×4 × 3

Andrew Ng

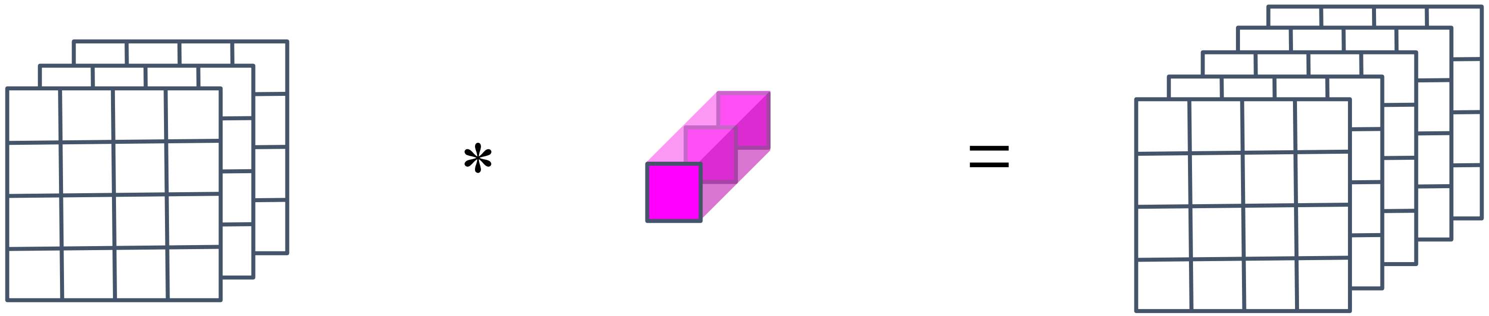
# Depthwise Separable Convolution

## Depthwise Convolution



432

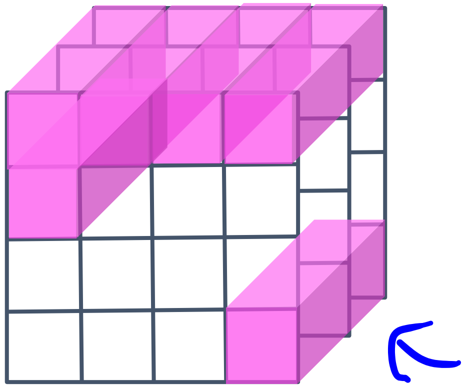
## Pointwise Convolution



4×4×3 4×4×5

Andrew Ng

# Pointwise Convolution



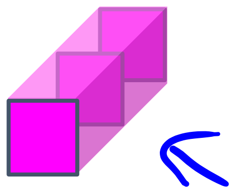
4 x 4 x 3

$n_{out} \times n_{out} \times n_c$

$*$

$n_c'$ filters

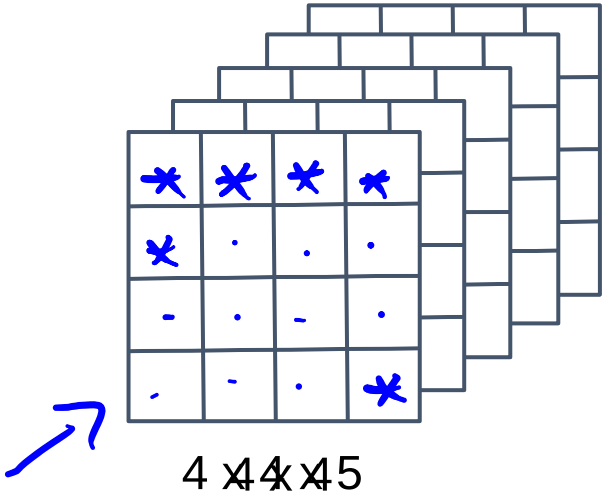
5



1 x 1 x 3

$1 \times 1 \times n_c$

=



4 x 4 x 5

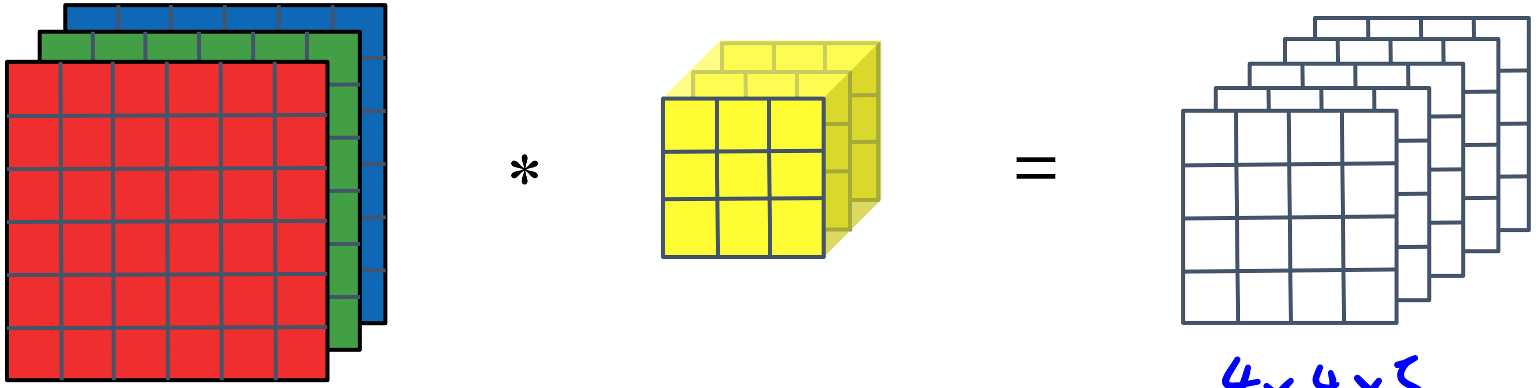
$n_{out} \times n_{out} \times n_c'$

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Computational cost | = | #filter params | x | # filter positions | x | # of filters |
| 240 | = | $1 \times 1 \times 3$ | $\times$ | $4 \times 4$ | $\times$ | 5 |

# Depthwise Separable Convolution

Normal Convolution

4x4x5 x 3x3x3 => 4x4x3 x (5 x 9) = 2160



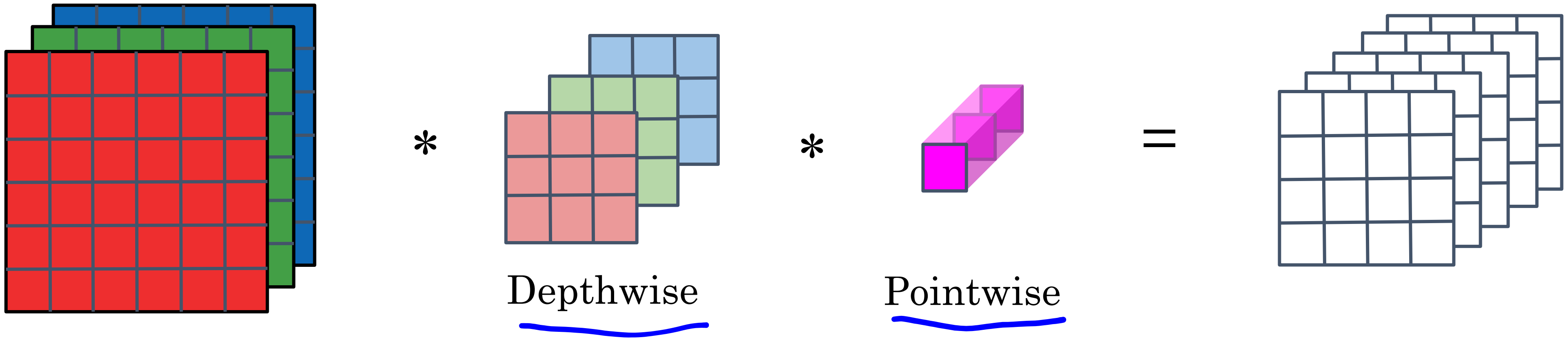

4x4x5

changed from
x to +

ratio
= 1/5 + 1/9
= 1/nc' + 1/f**2

Depthwise Separable Convolution

4x4x3 x 3x3 + 4x4x5 x 3 => 4x4x3 x (5 + 9) = 672




Andrew Ng

# Cost Summary

Cost of normal convolution 2160

Cost of depthwise separable convolution

depthwise + pointwise

432 + 240 = 672

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

$$= \frac{1}{n_c'} + \frac{1}{f^2}$$

$$\frac{1}{5} + \frac{1}{9}$$

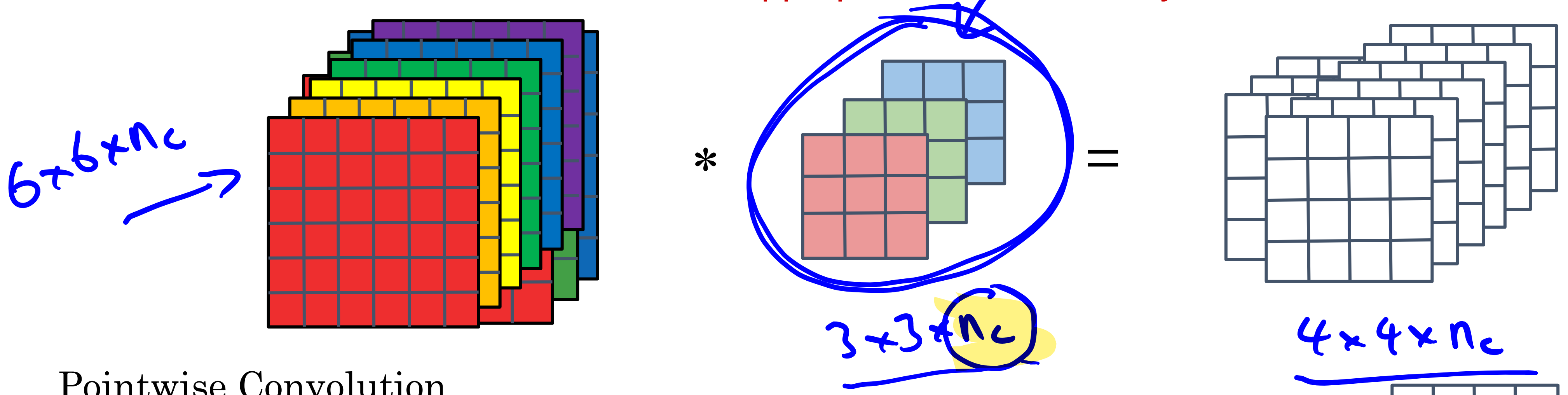
$$= \frac{1}{512} + \frac{1}{3^2}$$

~10 times cheaper

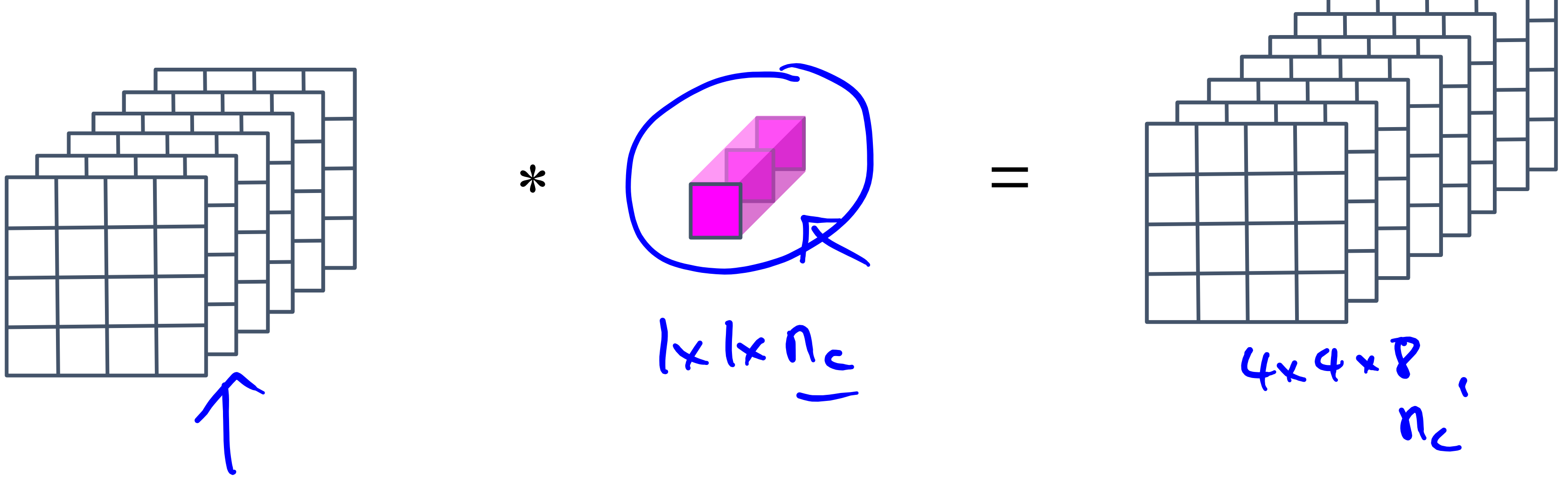
[Howard et al. 2017, MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications]

# Depthwise Separable Convolution

Depthwise Convolution




Pointwise Convolution



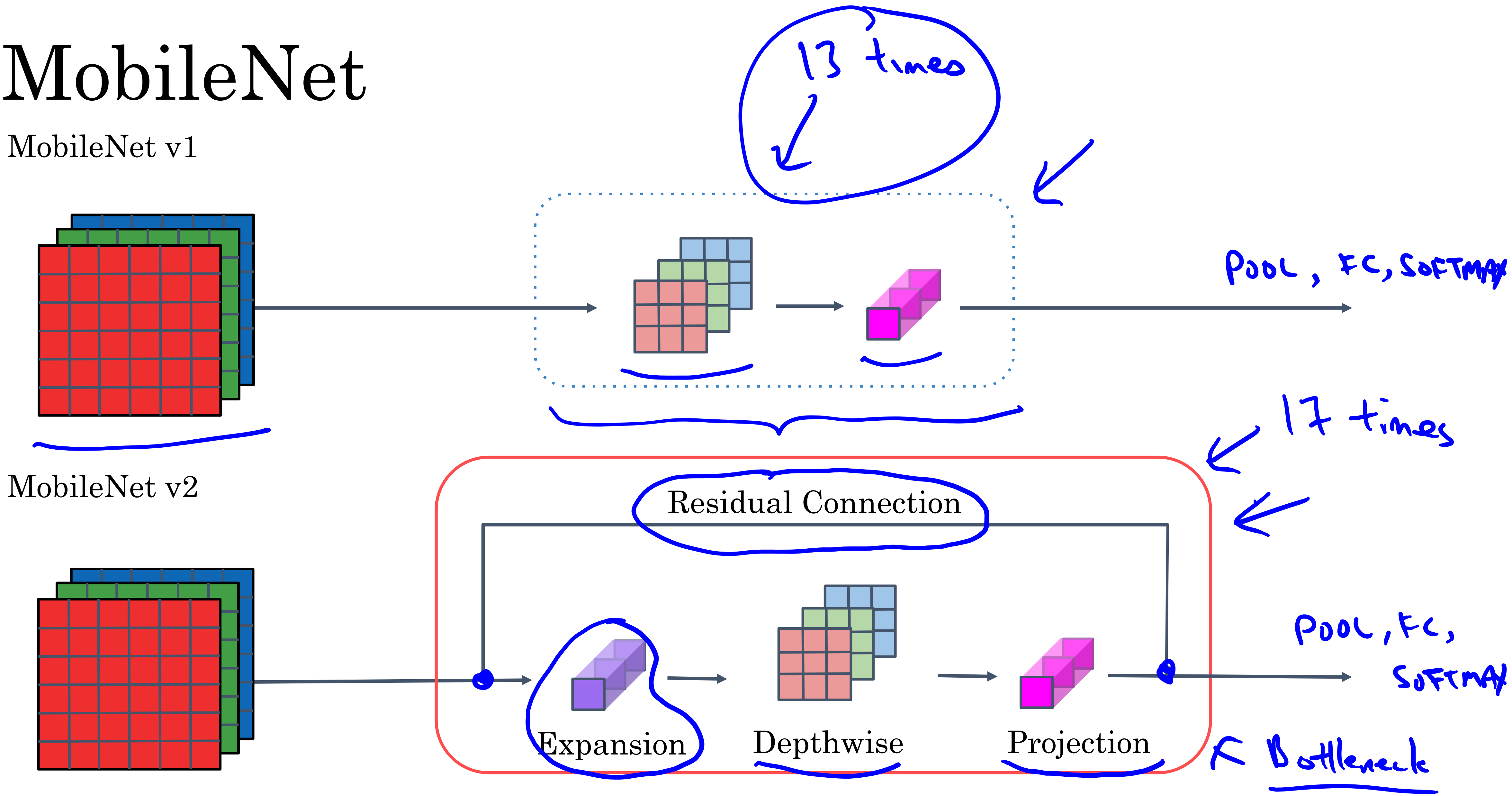

Andrew Ng

deeplearning.ai

Convolutional Neural Networks

# MobileNet Architecture

# MobileNet

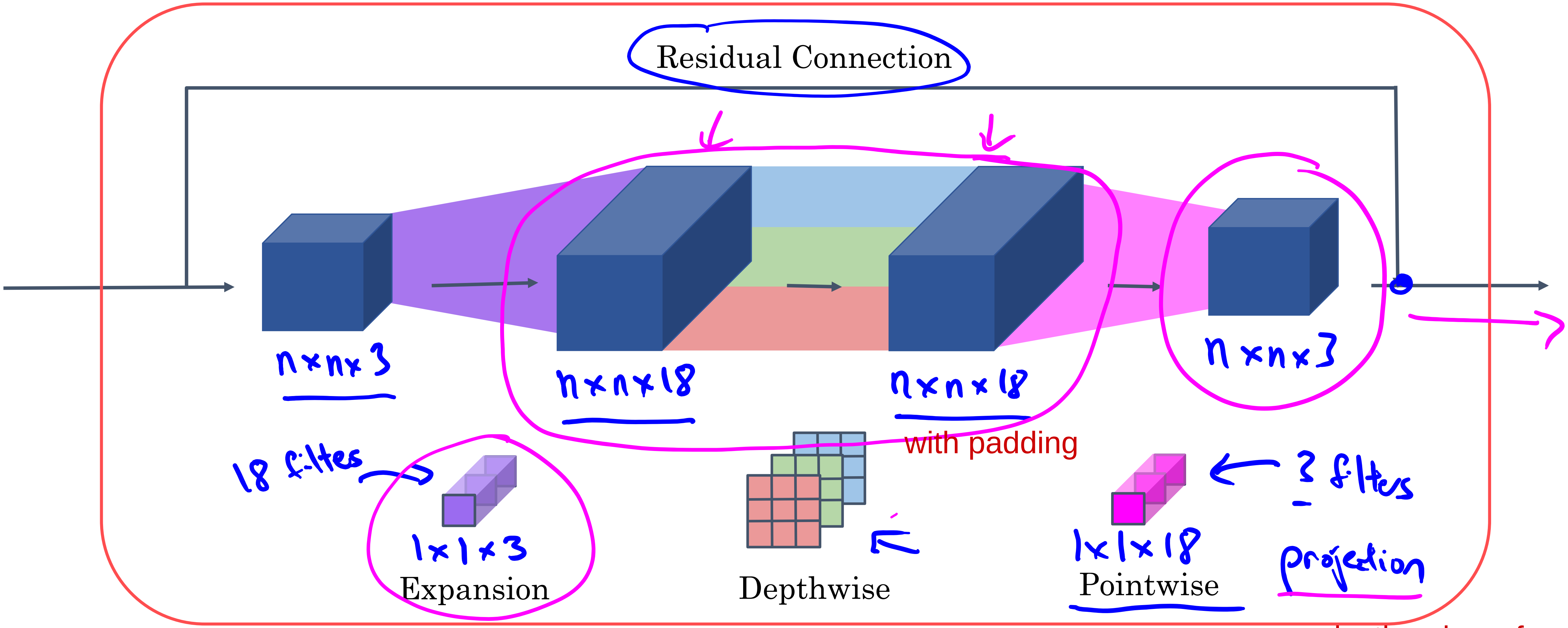



[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

Andrew Ng

# MobileNet v2 Bottleneck

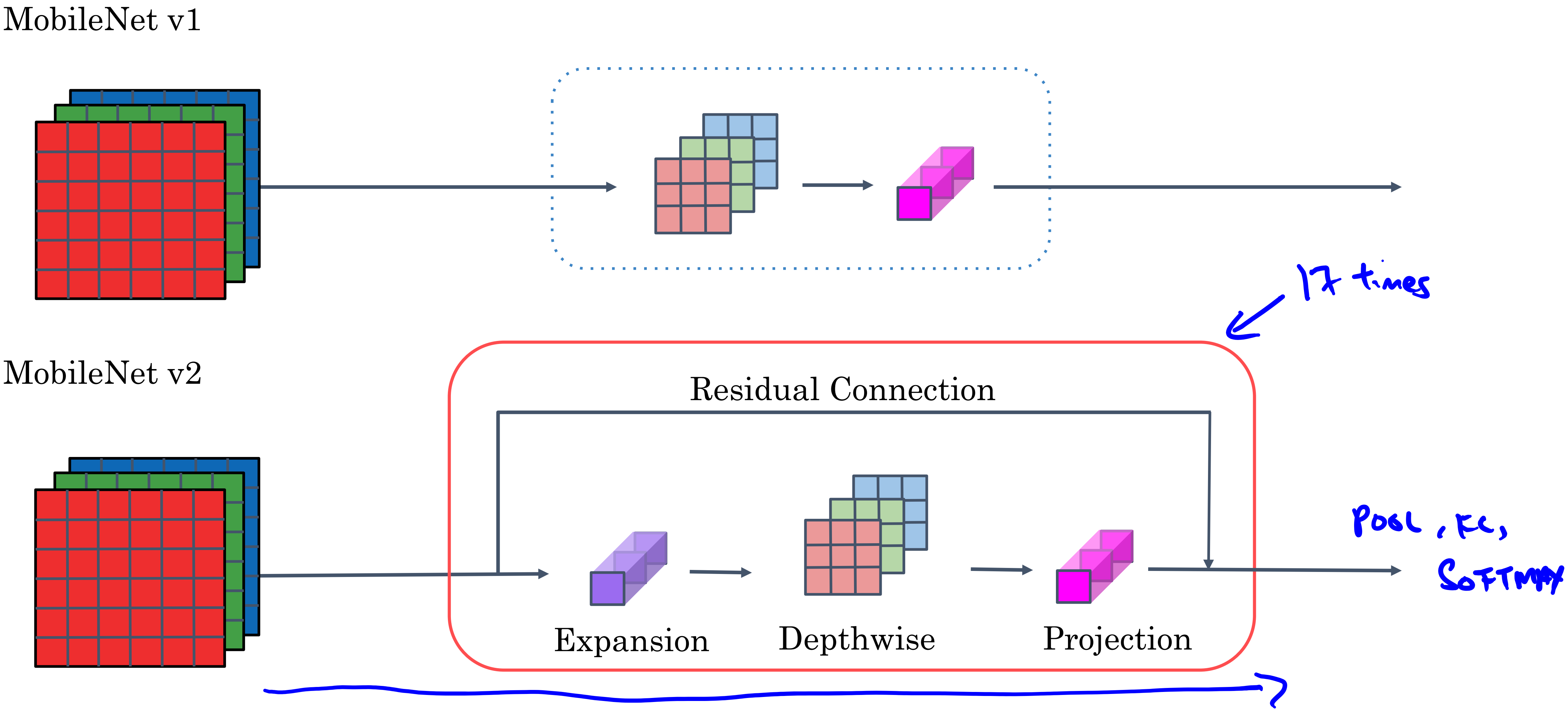
clever idea here is - keep memory low (in term of activations moved from L to L+1), by projecting down... but it allows NN to learn a richer / complex function within the layer




projecting down from nxnx18 to nxnx3

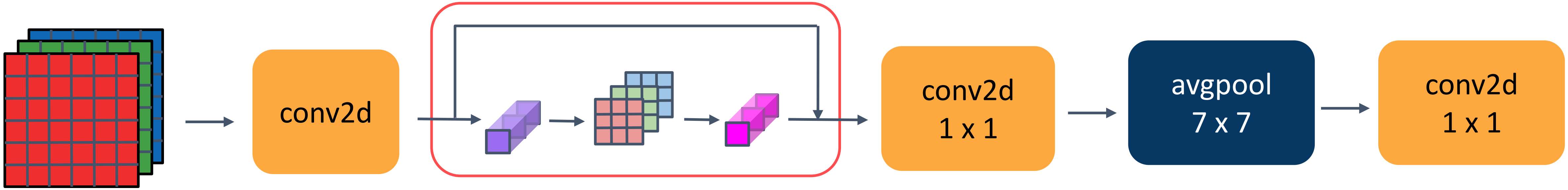
[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

# MobileNet




[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

# MobileNet v2 Full Architecture




[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

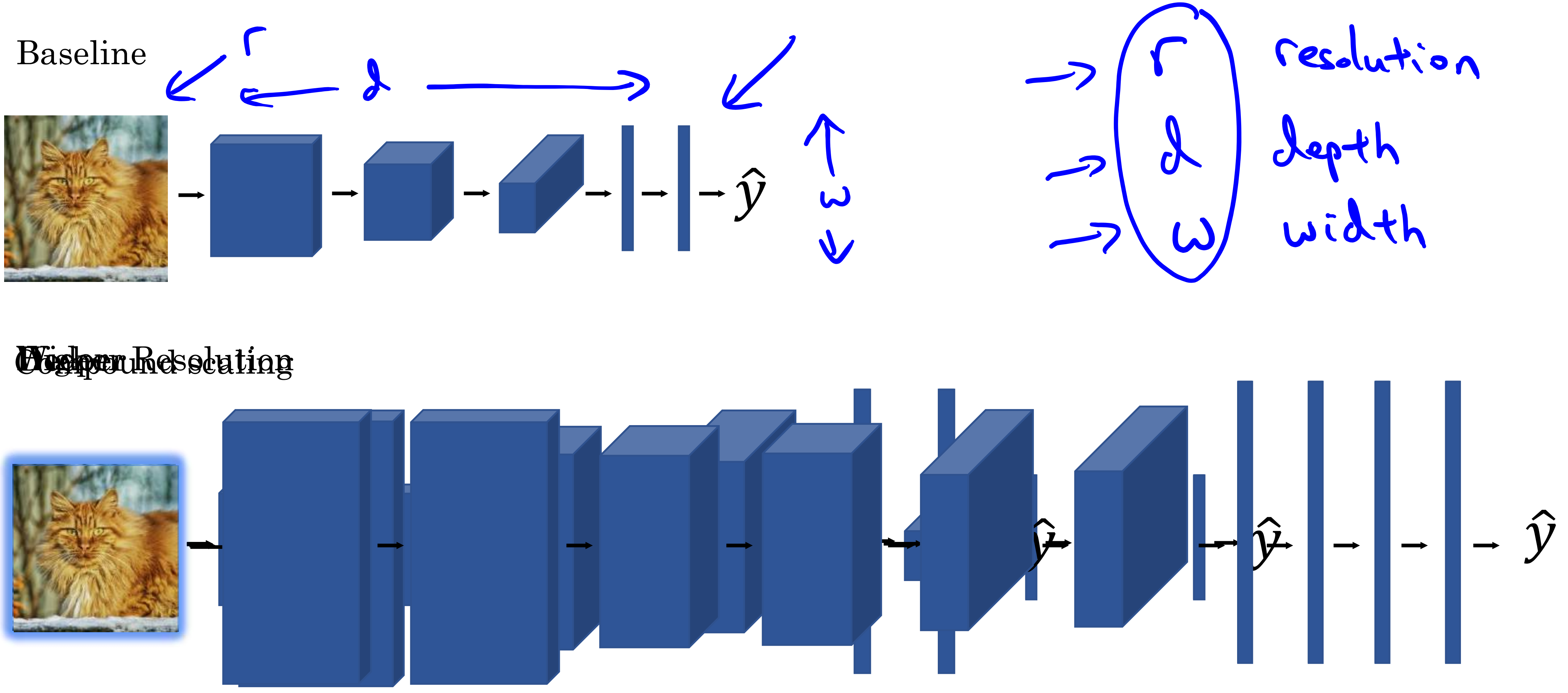
deeplearning.ai

# Convolutional Neural Networks

## EfficientNet

How to scale the network up or down based on the compute available on a device - think in terms of r/ d/ w and look for an open source implementation of efficient net to calibrate

# EfficientNet

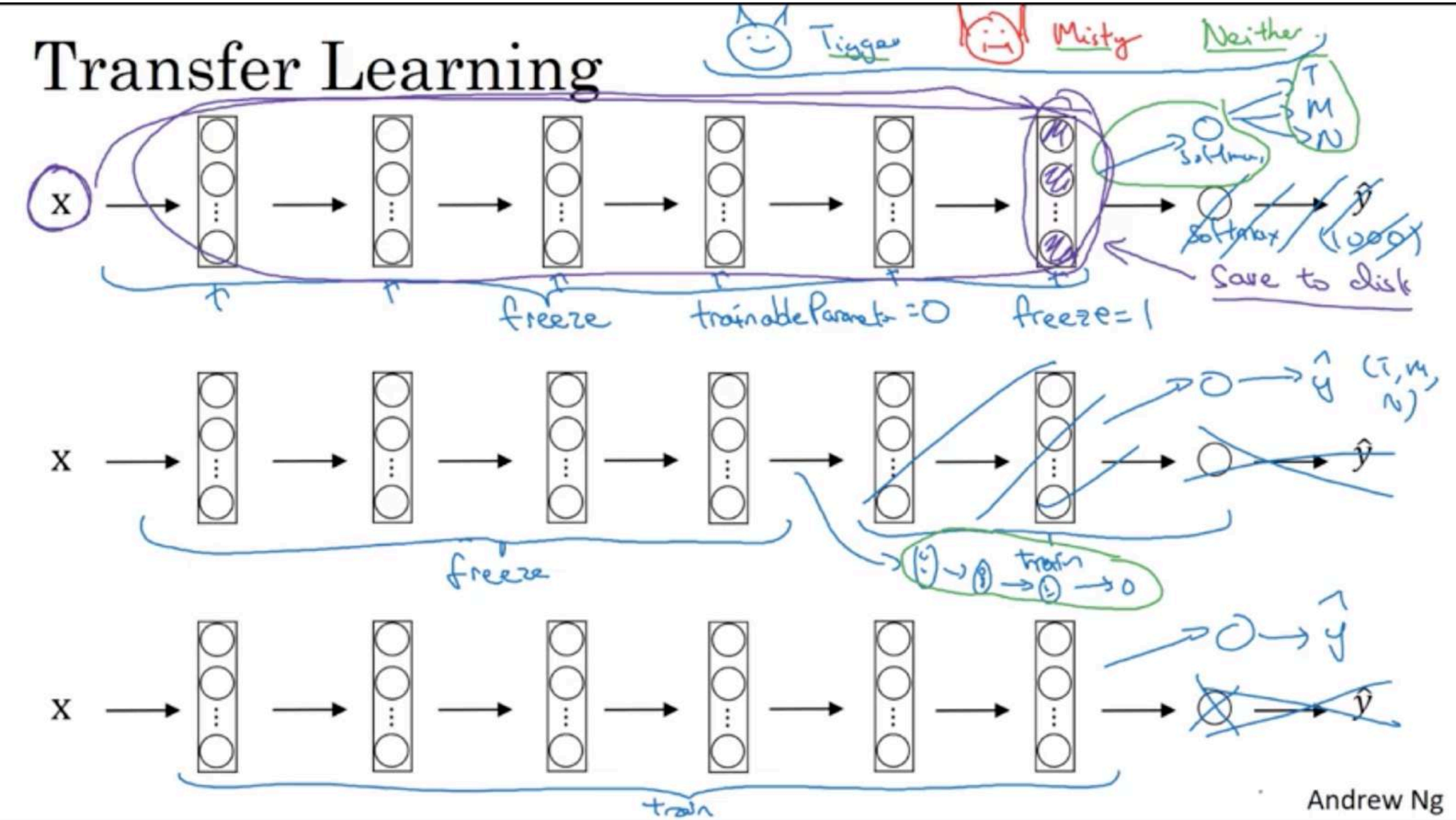



[Tan and Le, 2019, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks]

deeplearning.ai

# Practical advice for using ConvNets

# Transfer Learning




1st approach: few images - take imagenet (1000classes), replace the last layer with 3 unit softmax. Convert X to X' feature set by running it through the network upto the replaced layer and save to disk. Then it's like training a single layer shallow NN using X'. 2nd approach - have more pics to train on, then retrain >1 layers at the end. Extreme case 3rd approach - use trained weights as initialization points, and train entire network.

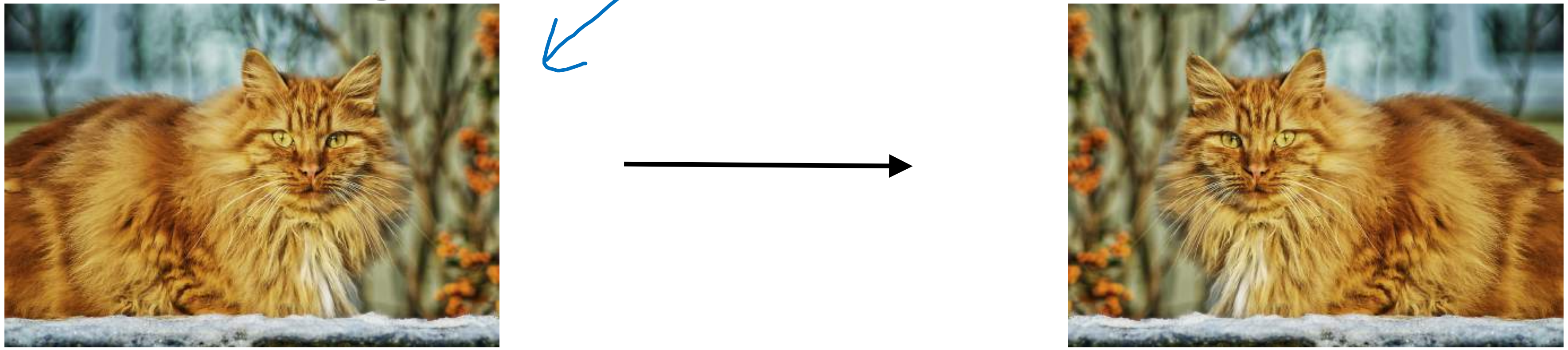
deeplearning.ai

# Practical advice for using ConvNets

## Data augmentation

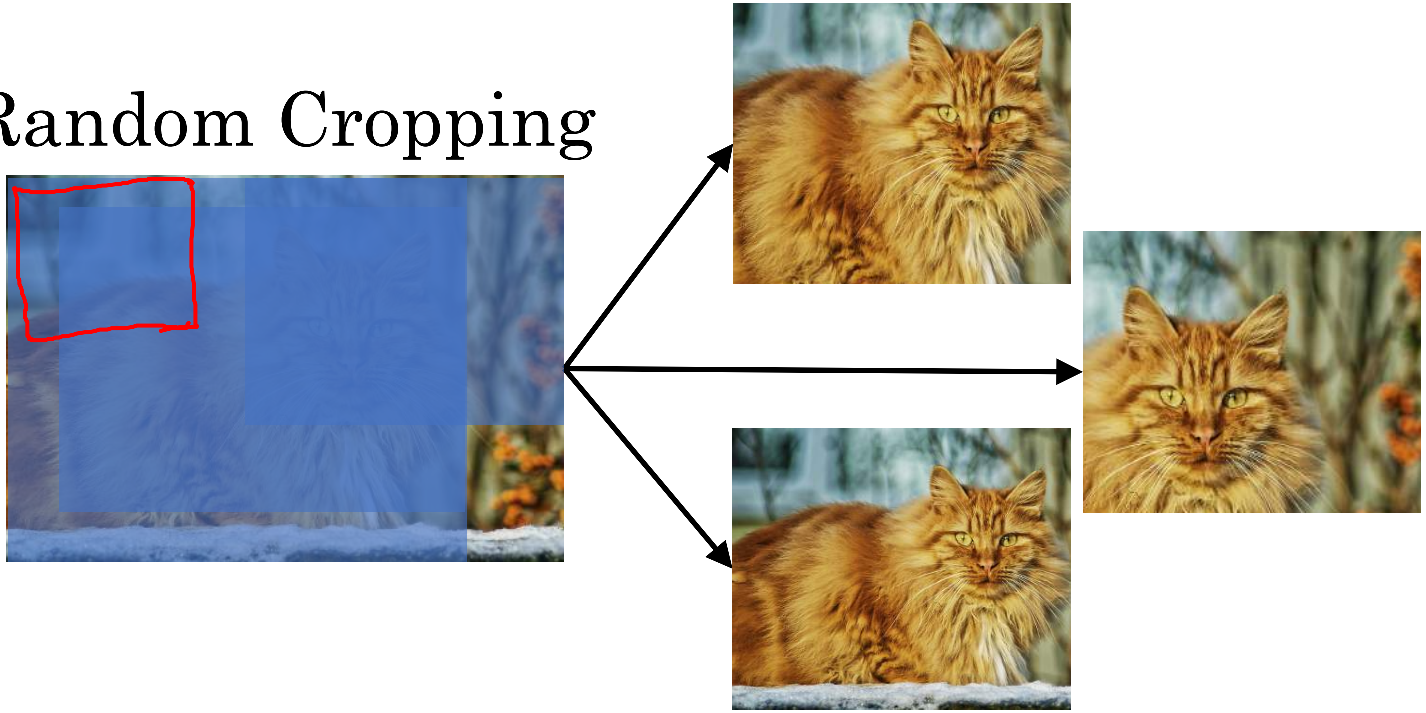
# Common augmentation method

Mirroring



y

Random Cropping



Rotation

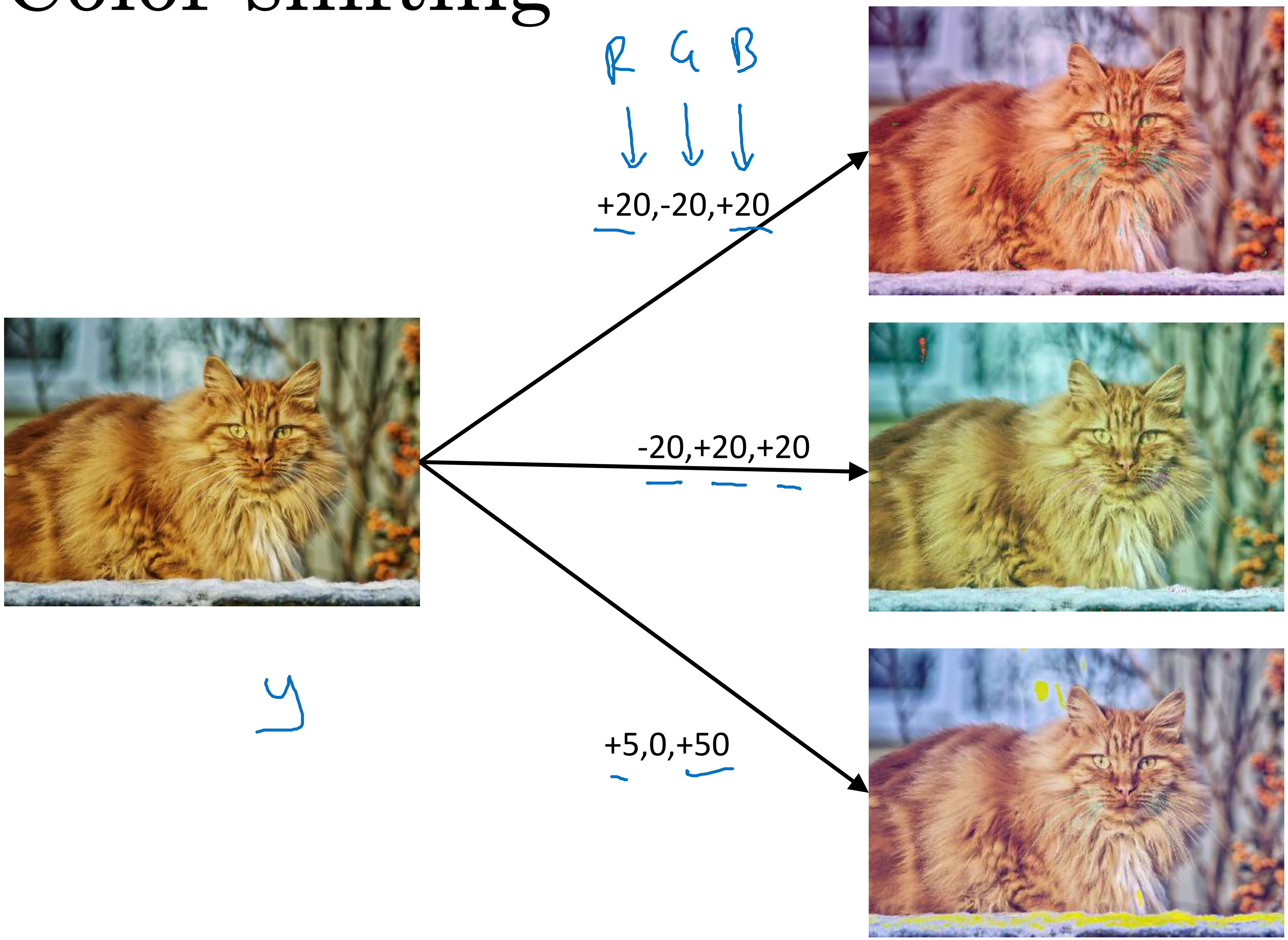
Shearing

Local warping

...

Andrew Ng

# Color shifting




Advanced:

PCA

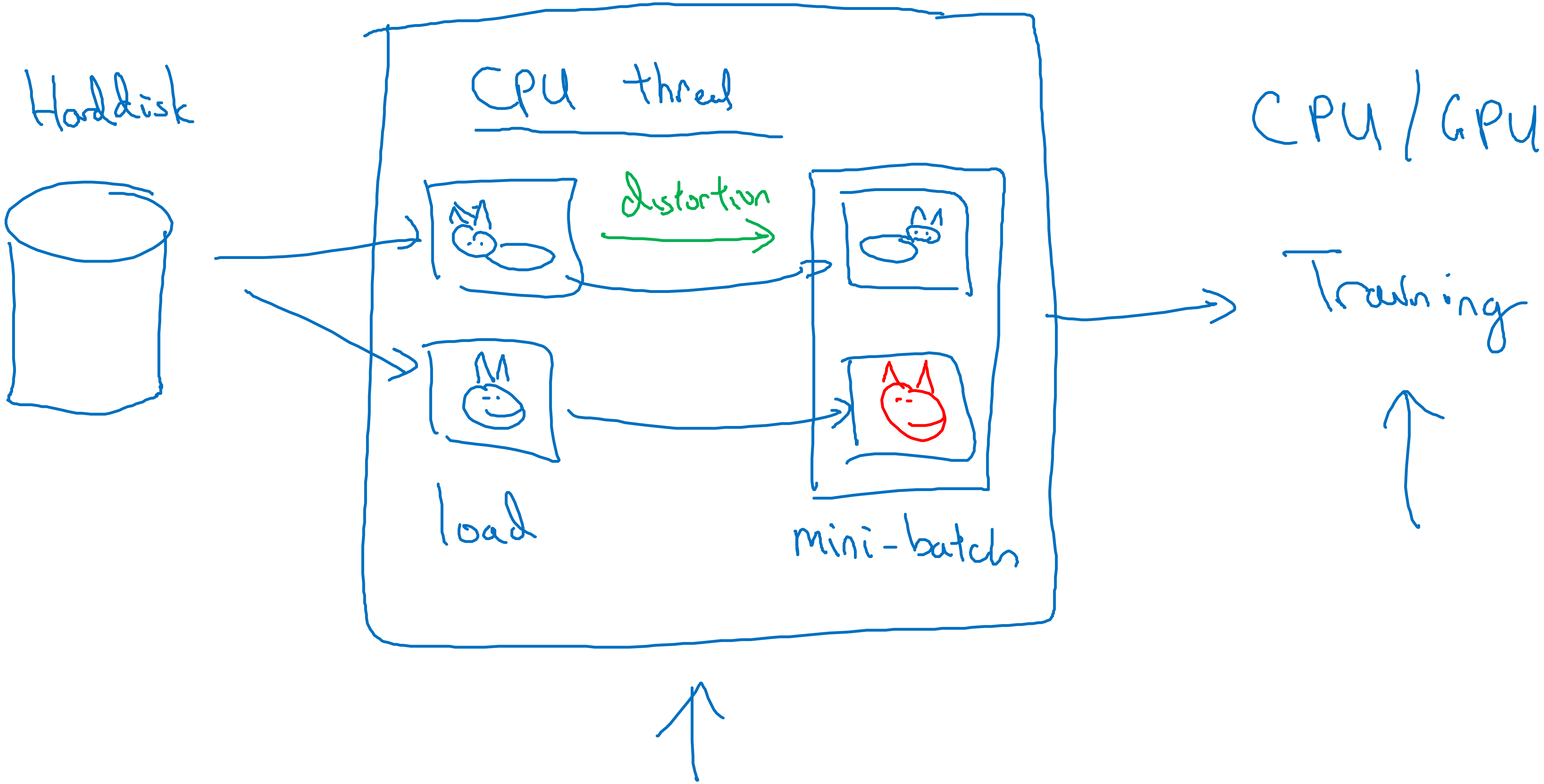
ml-class.org

[AlexNet paper

["PCA color augmentation."

R B G

# Implementing distortions during training



deeplearning.ai

# Practical advice for using ConvNets

# The state of computer vision

# Data vs. hand-engineering

Little data ⟷ Lots of data

More hand-engineering ("hacks")

Transfer learning

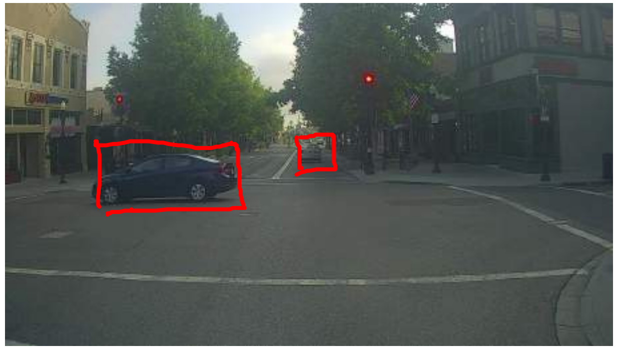
Tigger/Misty/neither.

Object detection

Image recognition

Speech recognition

Simpler algorithms
less hand-engineering



Two sources of knowledge

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

Andrew Ng

# Tips for doing well on benchmarks/winning competitions

## Ensembling

3-15 networks $\to \hat{y}$

- Train several networks independently and average their outputs

Good for competition, not so much for production system, because of high compute requirement + necessity to store data.

## Multi-crop at test time

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

10-crop

Take image - create 1 flipped copy. Create 5 crops for each image = "10-crop"

1 + 4 + 1 + 4

Andrew Ng

# Use open source code

- Use architectures of networks published in the literature

- Use open source implementations if possible

- Use pretrained models and fine-tune on your dataset

deeplearning.ai

# Object Detection

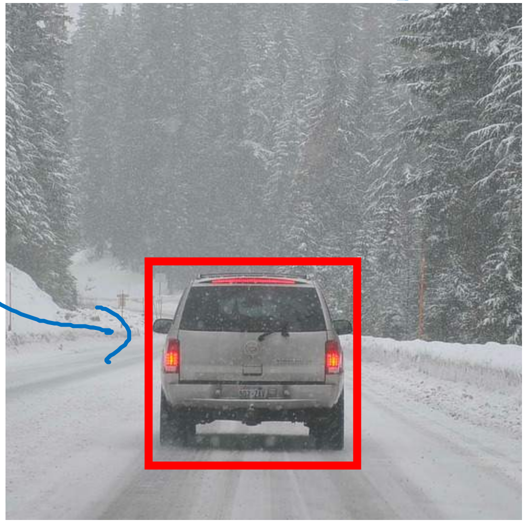
## Object localization

# What are localization and detection?

Image classification

"Car"

Classification with localization



"Car"

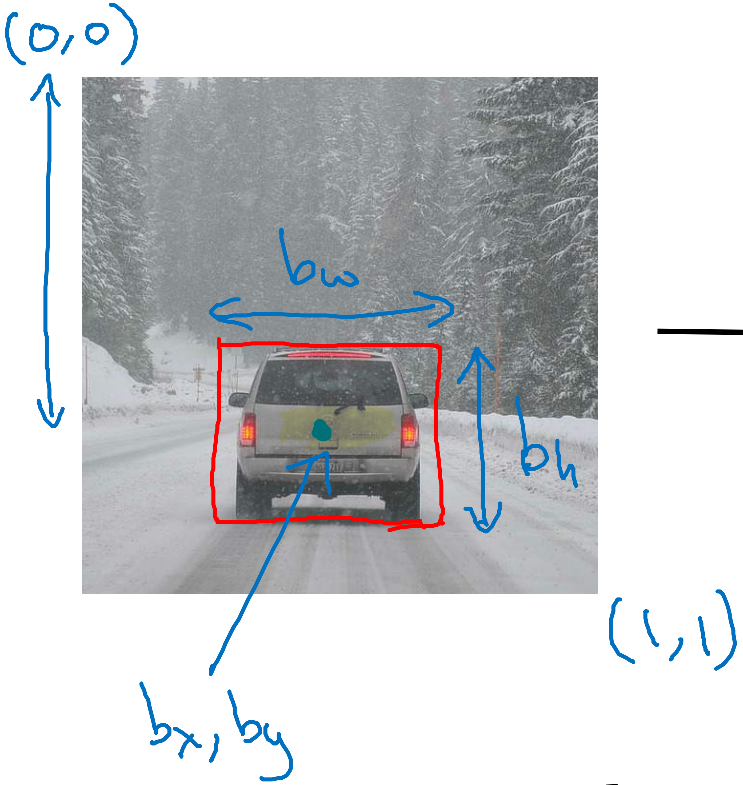
1 object

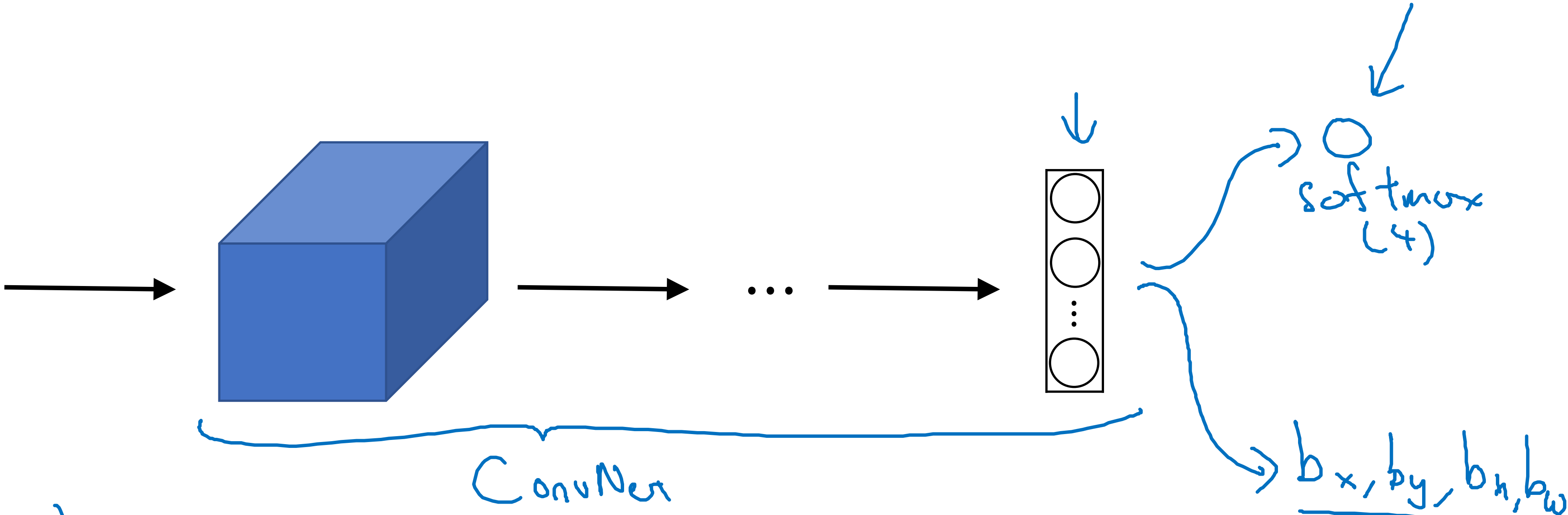
Detection

multiple objects

Andrew Ng

# Classification with localization




1 - pedestrian ←
2 - car ←
3 - motorcycle ←
4 - $\boxed{\text{background}}$

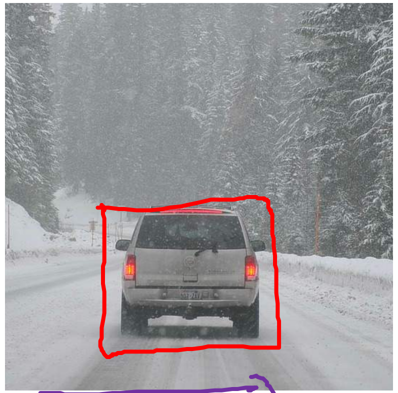
$b_x = 0.5$
$b_y = 0.7$
$b_h = 0.3$
$b_w = 0.4$

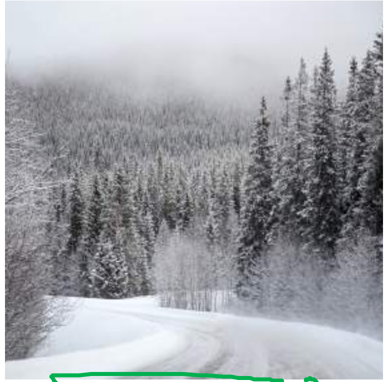
# Defining the target label y

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

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

$x =$





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

RMSE (ideally use RMSE + LogLoss)

in case of Pc = 0 (i.e. background) we only care about Pc, not other values

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

is there any object?

$(x, y)$

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

$P_c$

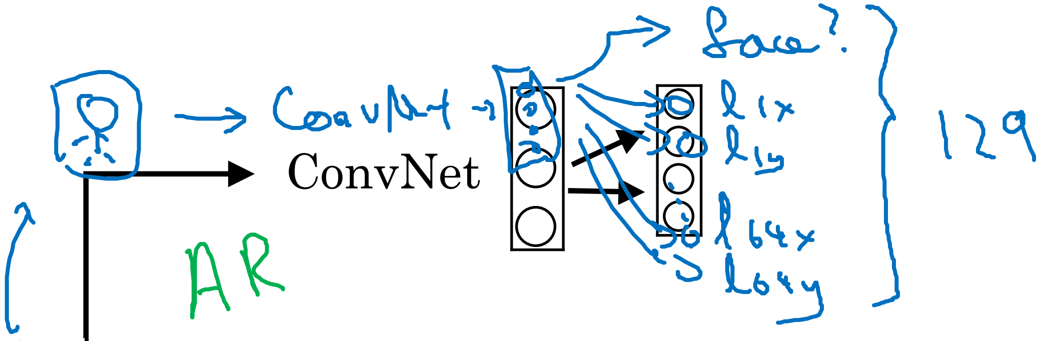
"don't care"

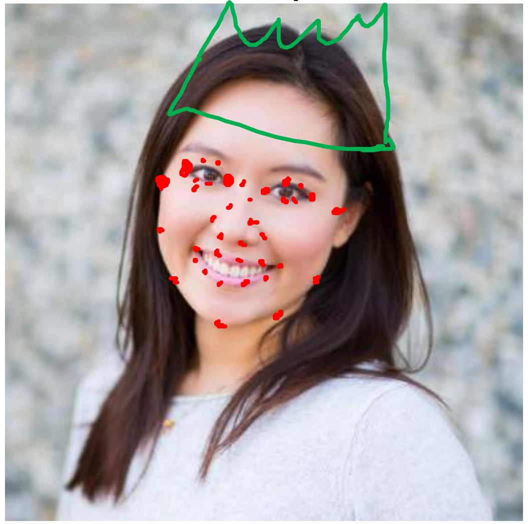
deeplearning.ai

# Object Detection

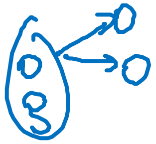
## Landmark detection

# Landmark detection



$b_x, b_y, b_h, b_w$



$$\left.\begin{matrix} l_{1x}, l_{1y}, \\ l_{2x}, l_{2y}, \\ l_{3x}, l_{3y}, \\ l_{4x}, l_{4y} \\ \vdots \\ l_{64}, l_{64y} \end{matrix}\right\} X, y$$

$$\begin{matrix} l_{1x}, l_{1y}, \\ \vdots \\ l_{32x}\ l_{32y} \end{matrix}$$

deeplearning.ai

# Object Detection

## Object detection

# Car detection example

Training set:

| x | y |
|---|---|
|  | 1 |
|  | 1 |
|  | 1 |
|  | 0 |
|  | 0 |

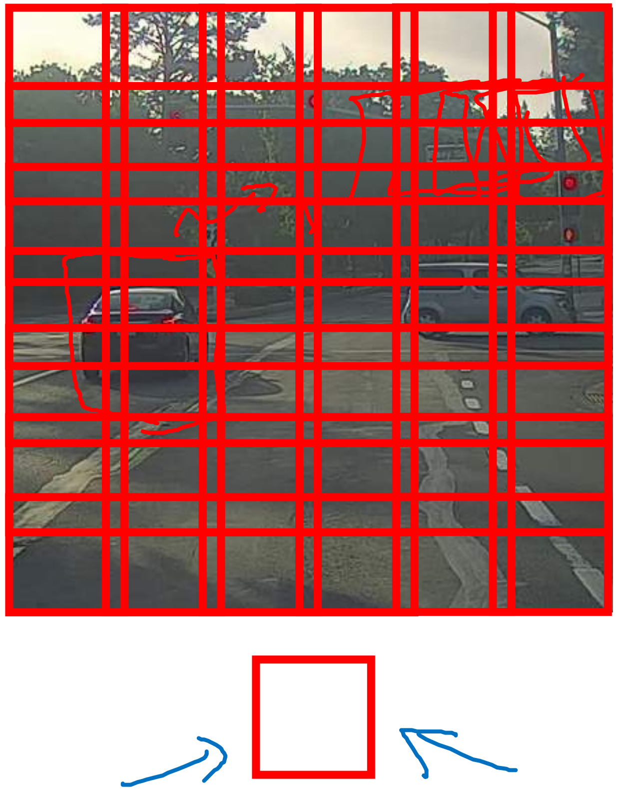
$\rightarrow$ ConvNet $\rightarrow y$

Andrew Ng

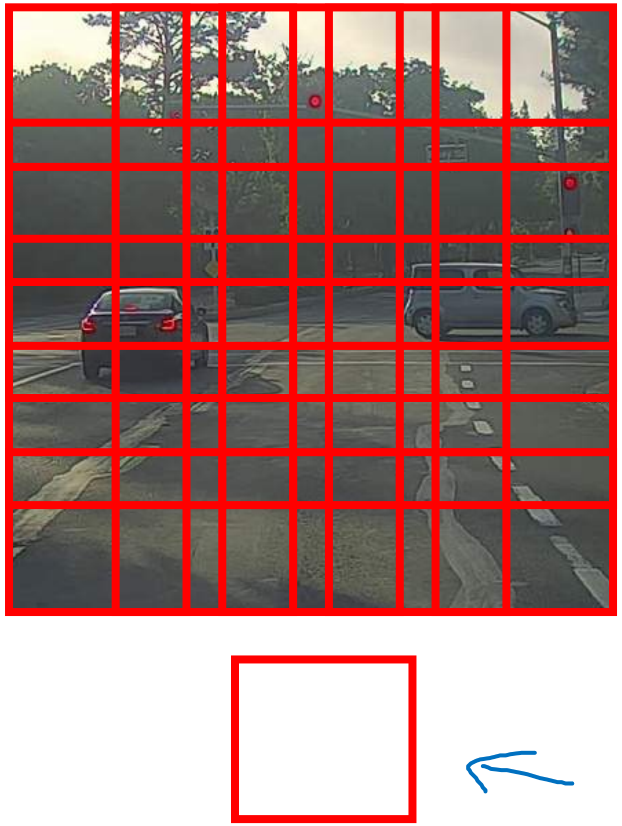
# Sliding windows detection

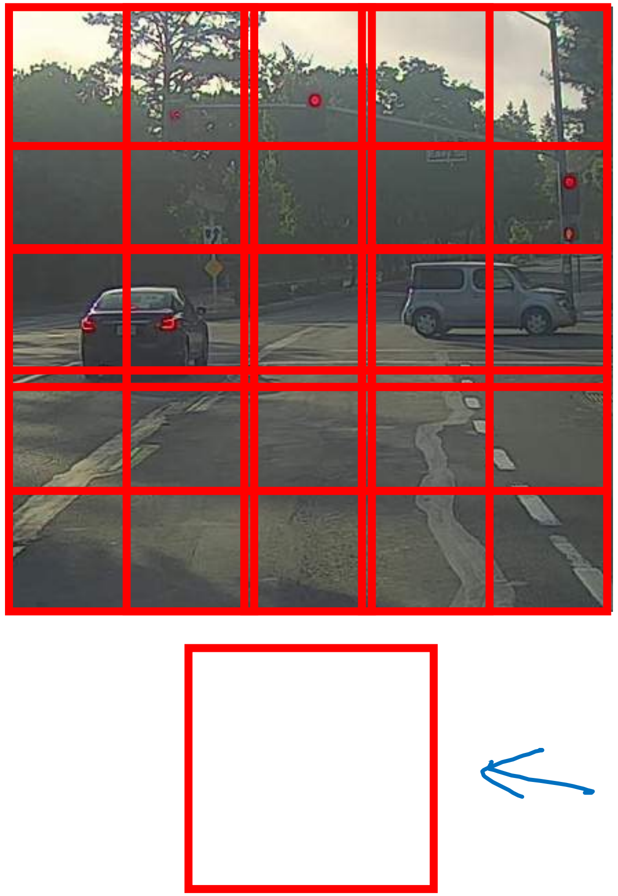
Disadvantage - computational cost of running multiple sliding windows. Originally (pre-CNN), classifiers were simple linear classifiers, so sliding window was feasible. Not so with CNNs.

→ ConvNet → 0



→ ConvNet





Computation cost

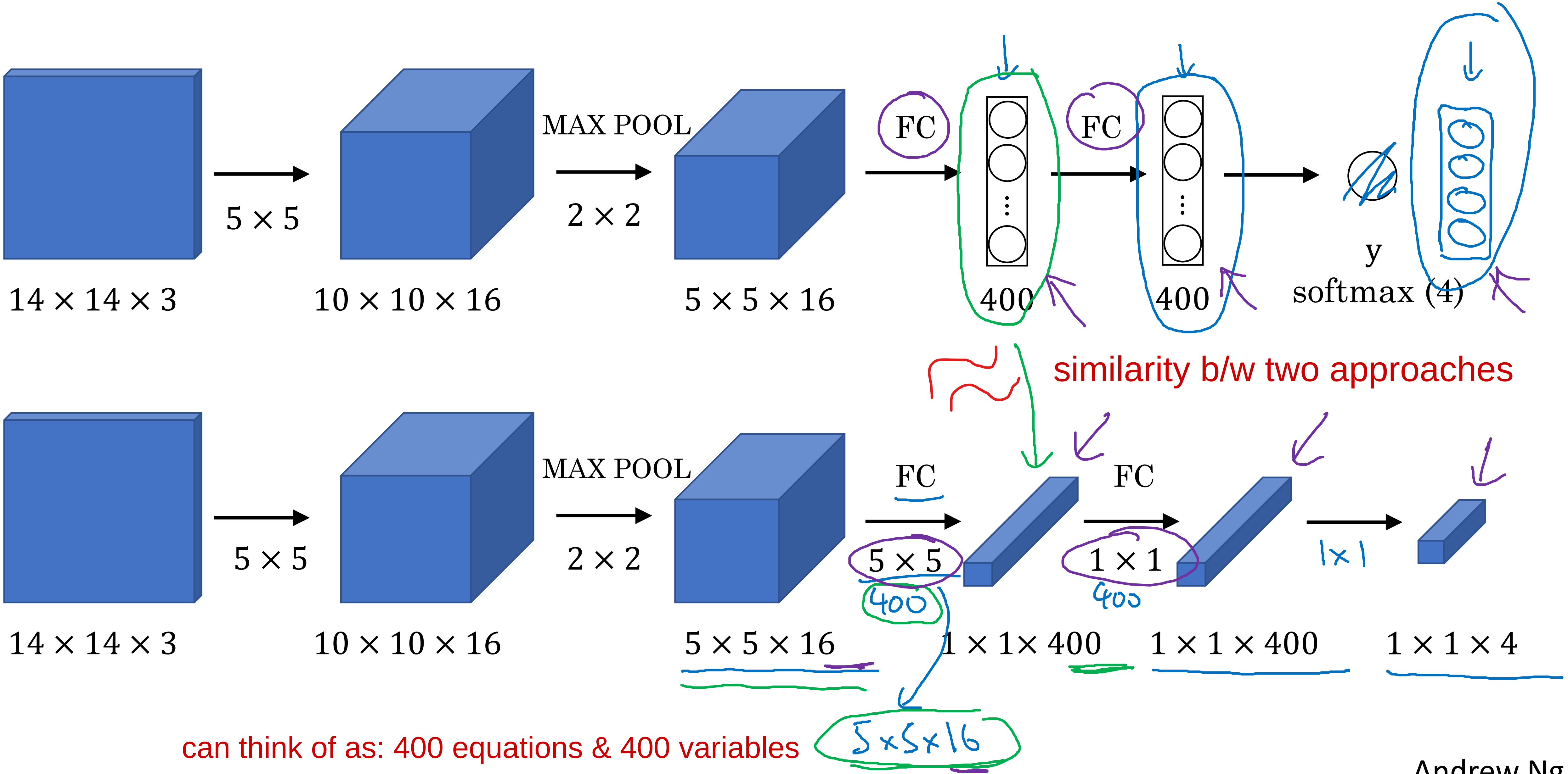
Andrew Ng

deeplearning.ai

# Object Detection

# Convolutional implementation of sliding windows

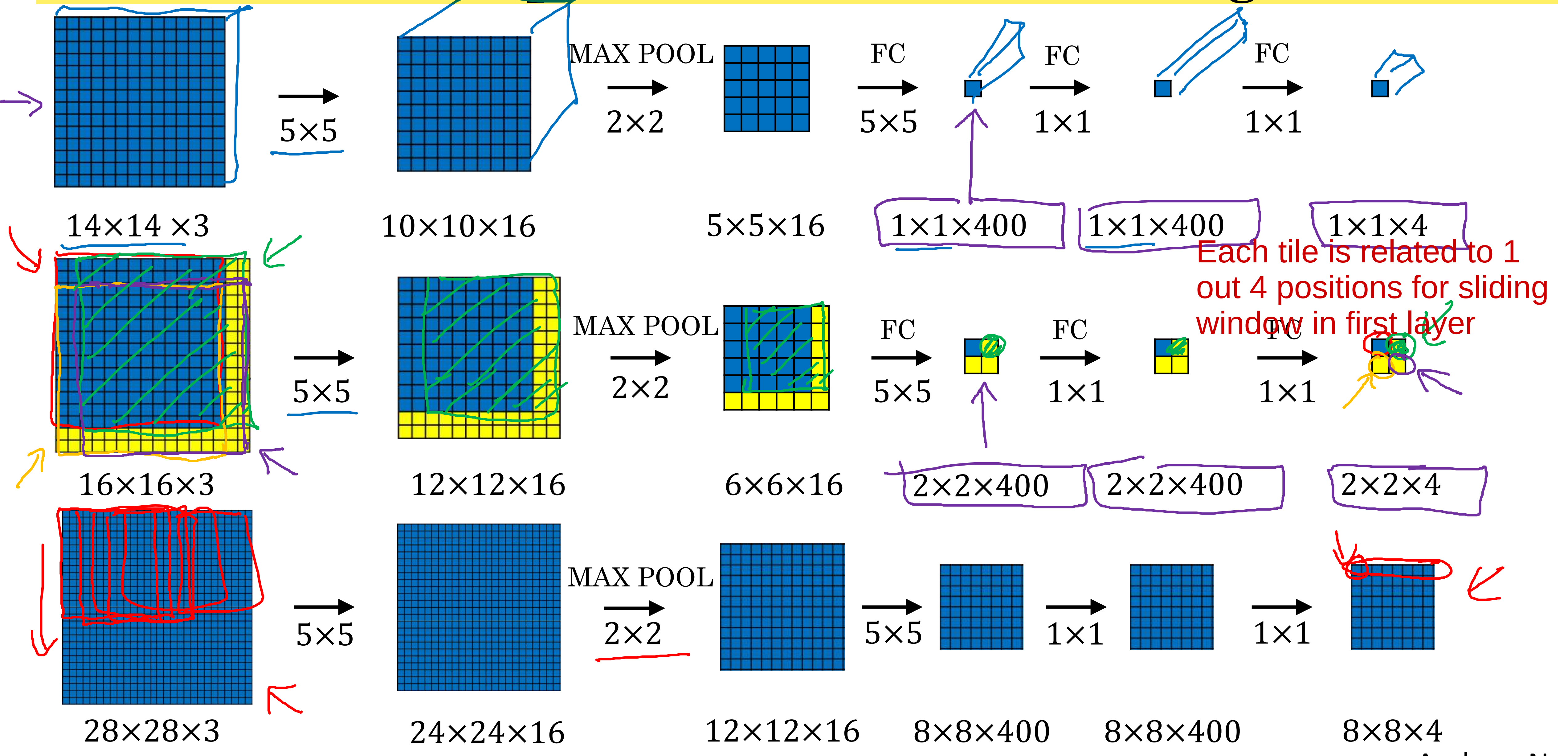
# Turning FC layer into convolutional layers




can think of as: 400 equations & 400 variables

Andrew Ng

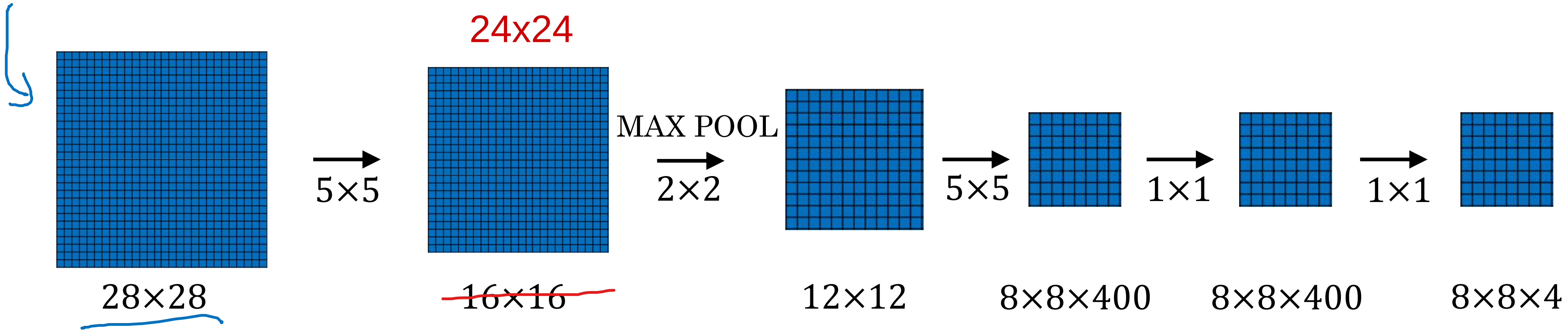
# Convolution implementation of sliding windows



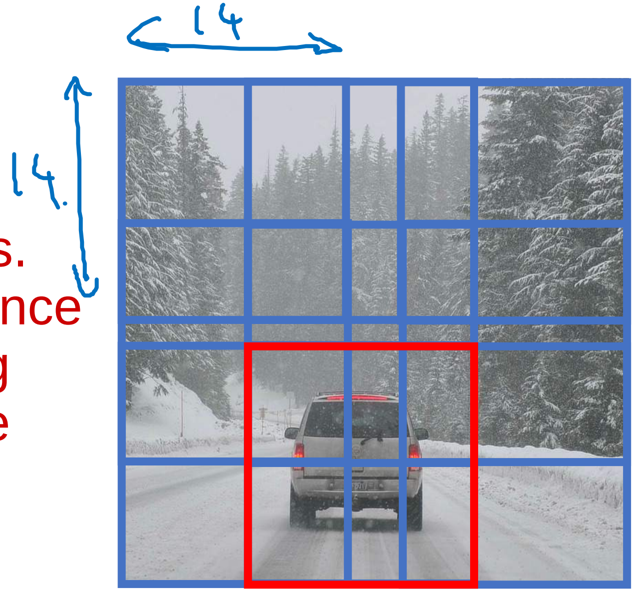

[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

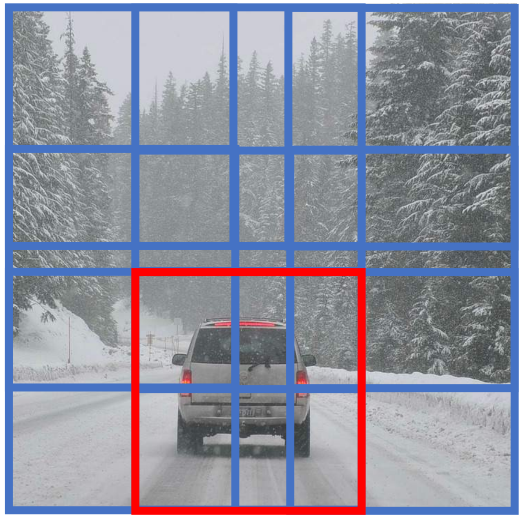
Andrew Ng

# Convolution implementation of sliding windows




Instead of doing sequential calculations. Do the forward pass once for whole image, using convolution, and make predictions in one go.



Negative - position of bounding box is still not going to be very accurate

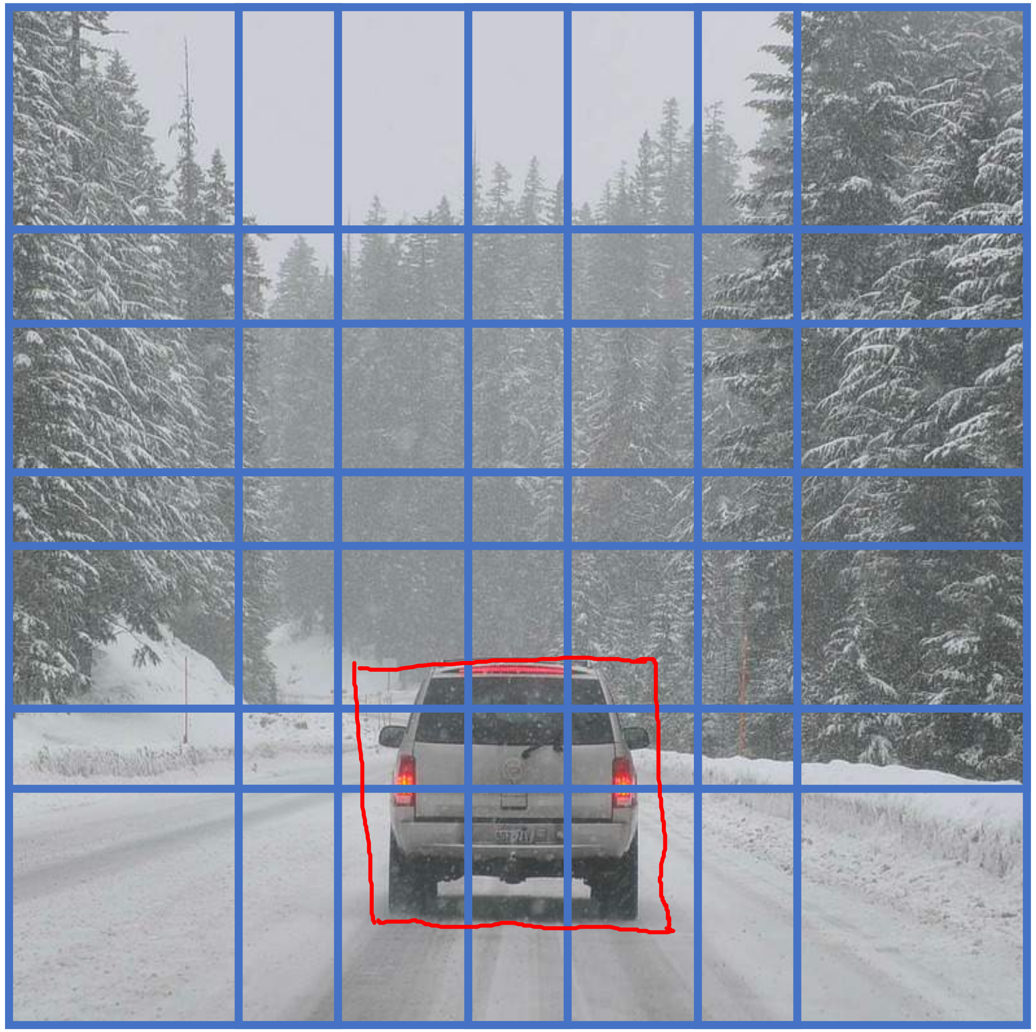
Andrew Ng

deeplearning.ai

# Object Detection

## Bounding box predictions

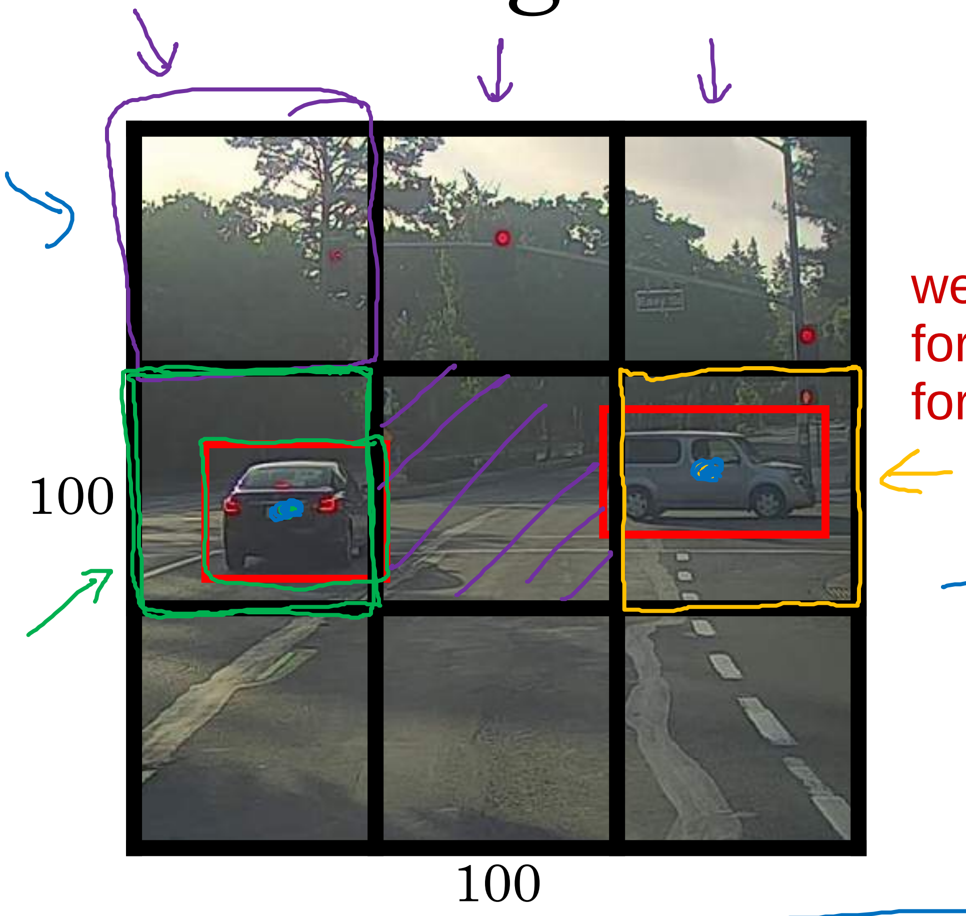
# Output accurate bounding boxes



Andrew Ng

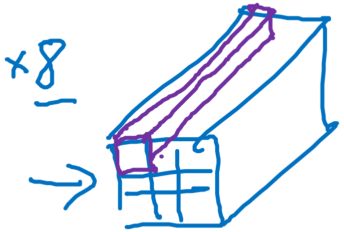
# YOLO algorithm

You Only Look Once
Apply fine grid to the image (19x19) - to each grid apply image classification & localization algorithm.
Object is assigned to the grid where its 'center' lies.




we want 8 o/p values
for each cell on grid,
for total of 3x3x8

Target output:
3×3×8



## Labels for training

For each grid cell:

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

$x$ (100×100×3) → CONV → MAXPOOL → .... → 3×3×8 $y$

→ 19×19×8

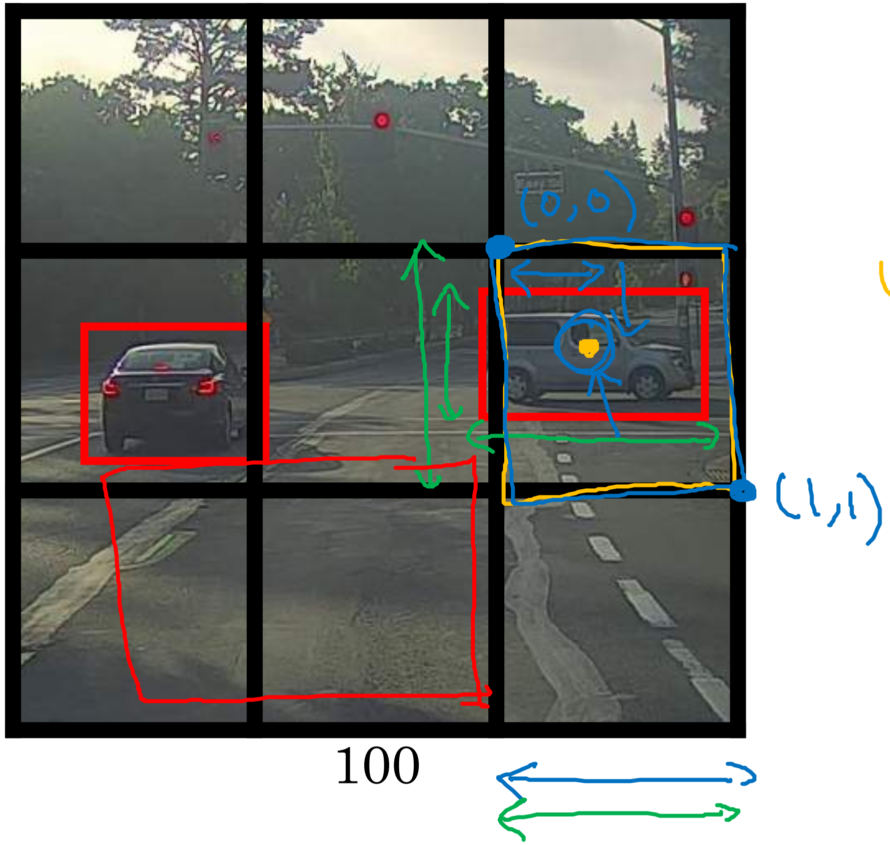
361

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

Andrew Ng

# Specify the bounding boxes

100




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

$\left.\begin{matrix} 0.4 \\ 0.3 \end{matrix}\right\}$ between 0 and 1

$\left.\begin{matrix} 0.9 \\ 0.5 \end{matrix}\right\}$ could be $> 1$

bottom two values can be >1

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

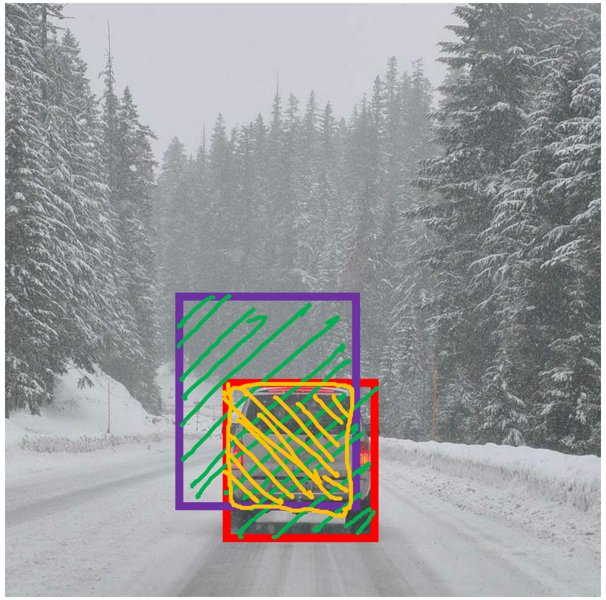
Andrew Ng

# Object Detection

deeplearning.ai

# Intersection over union

# Evaluating object localization



Intersection over Union (IoU)

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

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

by convention >0.5 is considered good enough

0.6

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

Andrew Ng

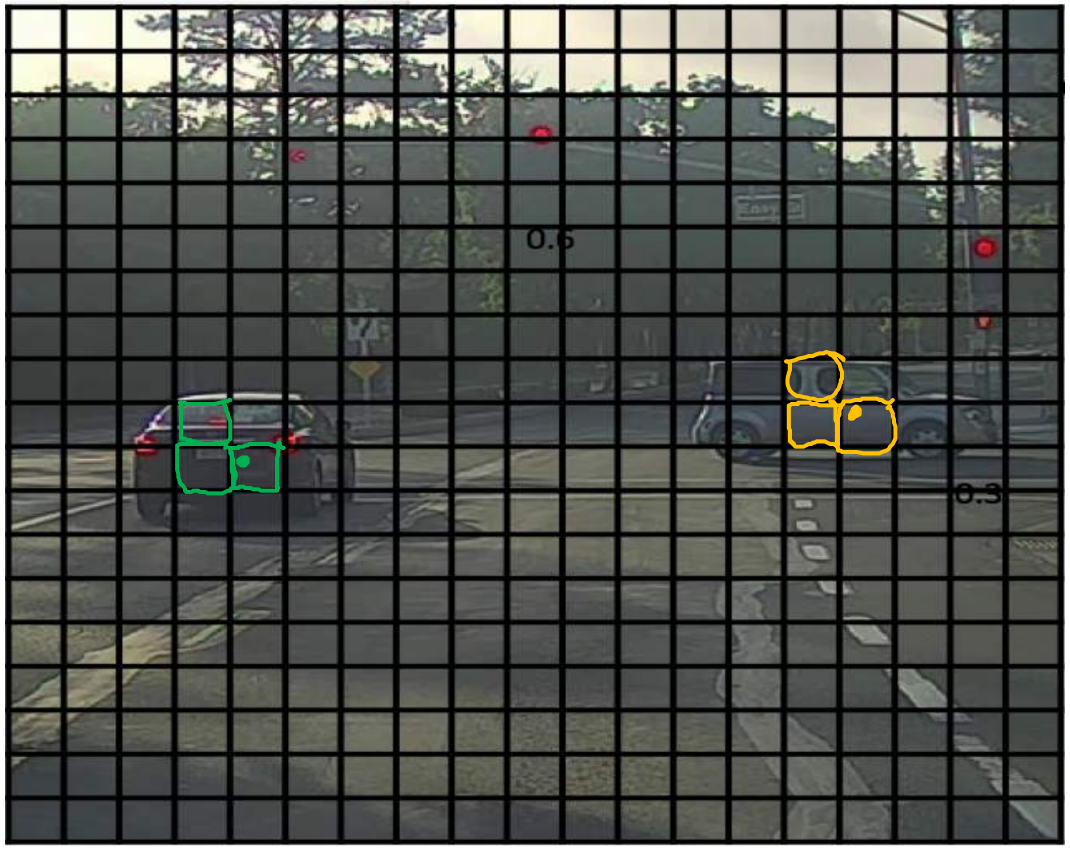
deeplearning.ai

# Object Detection

## Non-max suppression

# Non-max suppression example

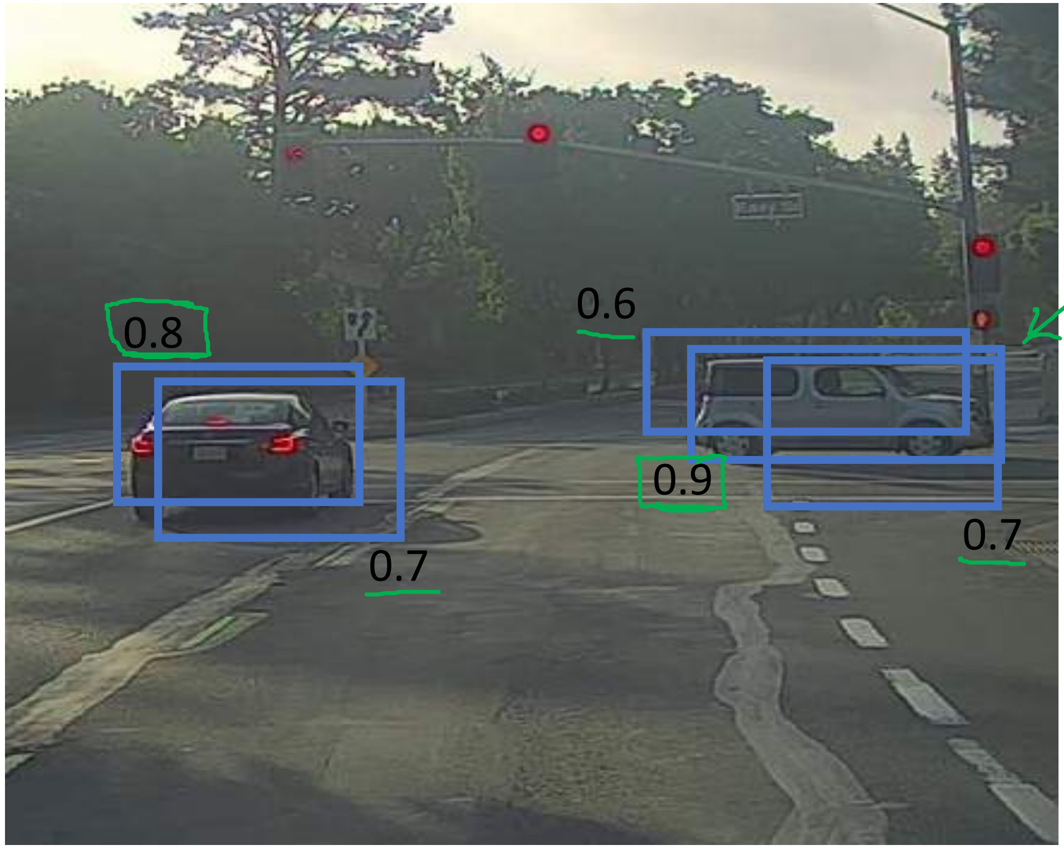
# Non-max suppression example




19x19

Andrew Ng

# Non-max suppression example




$P_c$

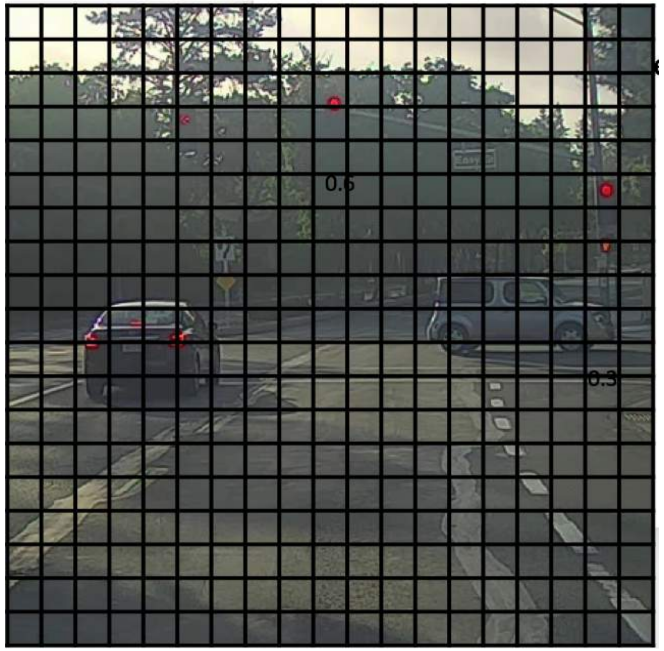
Issue - many boxes will detect any single car and generate Pc > 0

- pick the one with the max value (most confident detection)

- suppress other boxes with significant overlap with the max-box (high IoU)

in the image on left - only two boxes 0.8 and 0.9 will be highlighted

# Non-max suppression algorithm



$19\times19$

Each output prediction is: $\begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \end{bmatrix}$

361 outputs

$\begin{bmatrix} c_1 \\ c_2 \\ c_3 \end{bmatrix}$

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 $\geq 0.5$ with the box output
  in the previous step

Andrew Ng

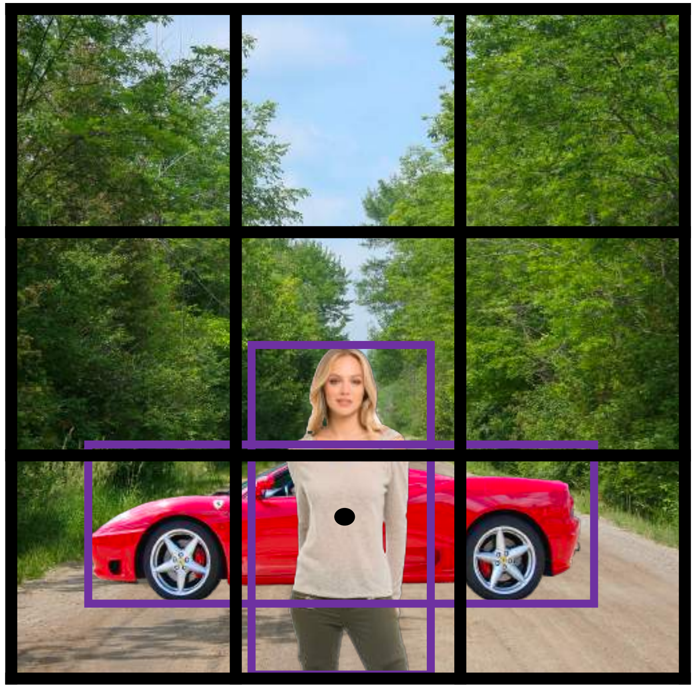
deeplearning.ai

# Object Detection

## Anchor boxes

# Overlapping objects:

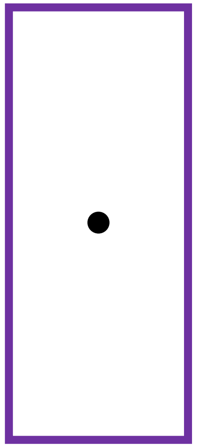
SOLUTION: suppose there can be two kinds of bounding boxes, include both as part the output vector



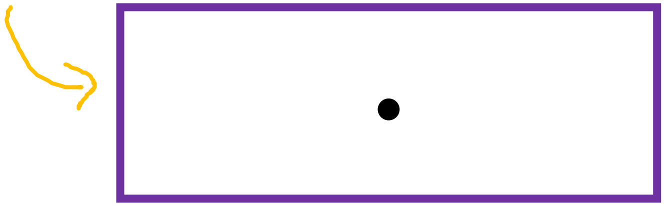
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

ISSUE:
mid-point of both objects fall in same grid - implying we'll know there is a landmark object here, but only be able to pick one

Anchor box 1:



Anchor box 2:



$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ \vdots \\ c_3 \end{bmatrix} \begin{matrix} \left.\vphantom{\begin{matrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{matrix}}\right\} \text{anchor box 1} \\ \left.\vphantom{\begin{matrix} p_c \\ b_x \\ \vdots \\ c_3 \end{matrix}}\right\} \text{anchor box 2} \end{matrix}$$

8x2 o/p vector

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

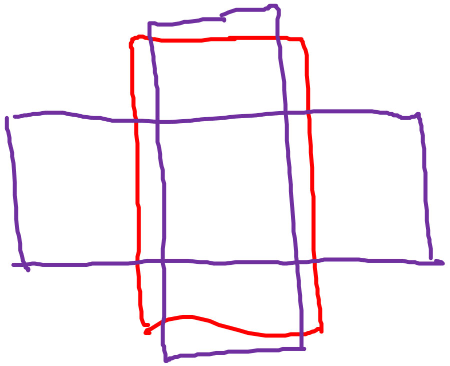
# Anchor box algorithm

## 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)

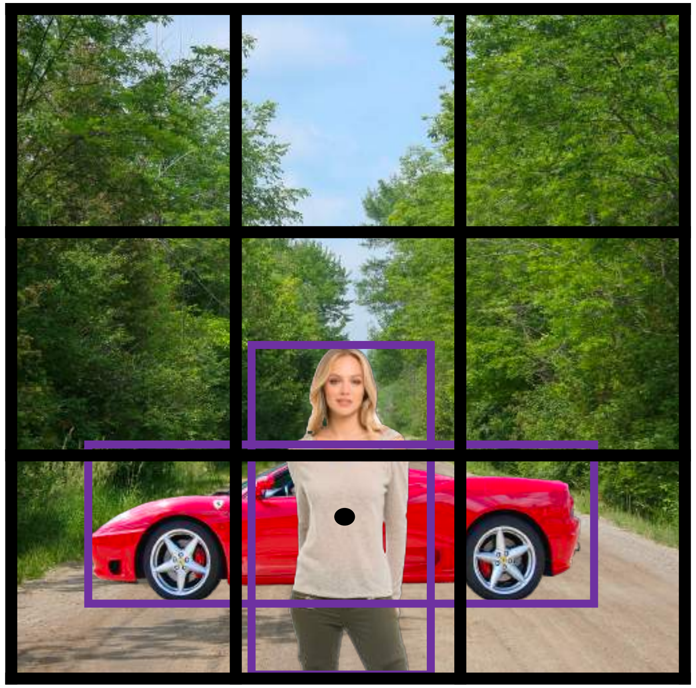
Output y:

$3 \times 3 \times 16$

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

Andrew Ng

# Anchor box example



What it won't capture:
(a) 3 objects in same grid
(b) 2 objects with same anchor box

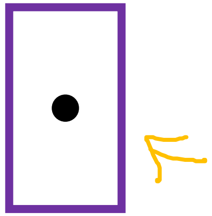
Chances of such collision are reduced as we make the grid finer

In advanced research - people use more than 2 anchor boxes, or use KNN algorithm

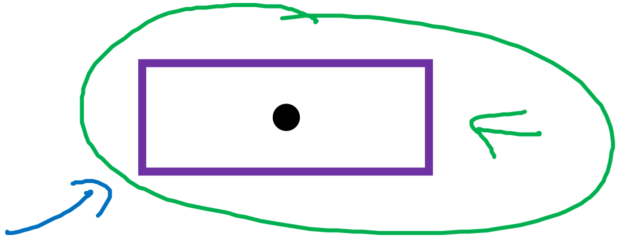
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \\ p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

| $y$ | | | car only? | |
|---|---|---|---|---|
| $p_c$ | 1 | | 0 | anchor box 1 |
| $b_x$ | $b_x$ | | ? | |
| $b_y$ | $b_y$ | | ? | |
| $b_h$ | $b_h$ | | ? | |
| $b_w$ | $b_w$ | | ? | |
| $c_1$ | 1 | person | ? | |
| $c_2$ | 0 | | ? | |
| $c_3$ | 0 | | ? | |
| $p_c$ | 1 | | 1 | anchor box 2 |
| $b_x$ | $b_x$ | | $b_x$ | |
| $b_y$ | $b_y$ | | $b_y$ | |
| $b_h$ | $b_h$ | | $b_h$ | |
| $b_w$ | $b_w$ | | $b_w$ | |
| $c_1$ | 0 | | 0 | |
| $c_2$ | 1 | car | 1 | |
| $c_3$ | 0 | | 0 | |

Anchor box 1:



Anchor box 2:



Andrew Ng

deeplearning.ai

# Object Detection

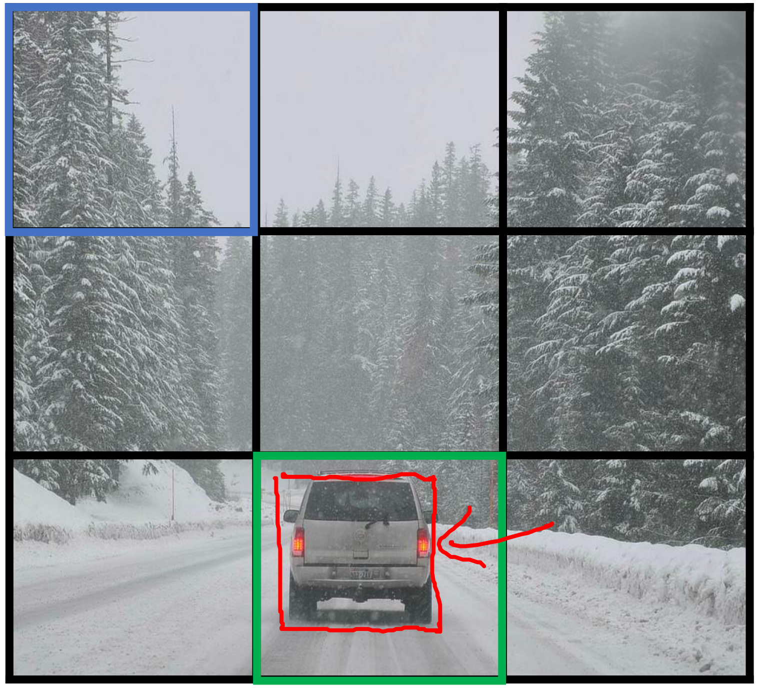
# Putting it together: YOLO algorithm

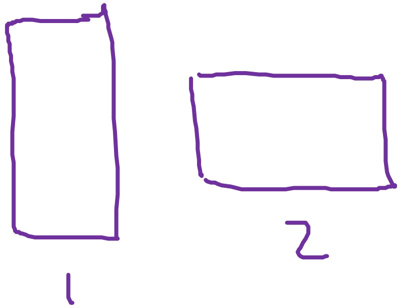
# Training

1 - pedestrian

2 - car

3 - motorcycle






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

more overlap with 2nd anchor box

$3 \times 3 \times 16$

y is $3 \times 3 \times 2 \times 8$ training set for each image

$19 \times 19 \times 16$

$19 \times 19 \times 40$

#anchors

5 + #classes

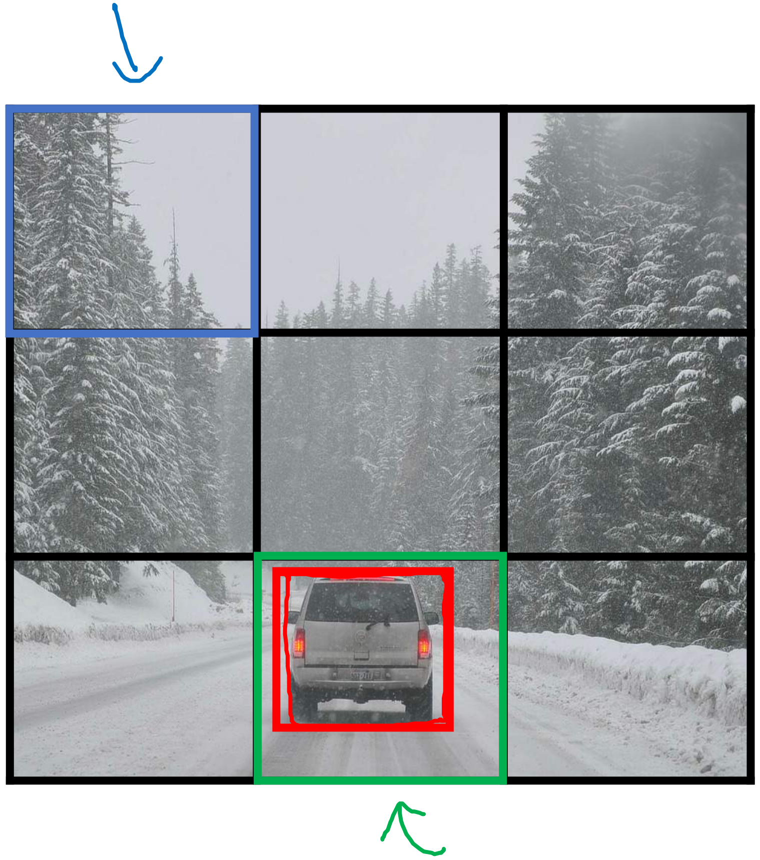
$100 \times 100 \times 3$ → ConvNet → $3 \times 3 \times 16$

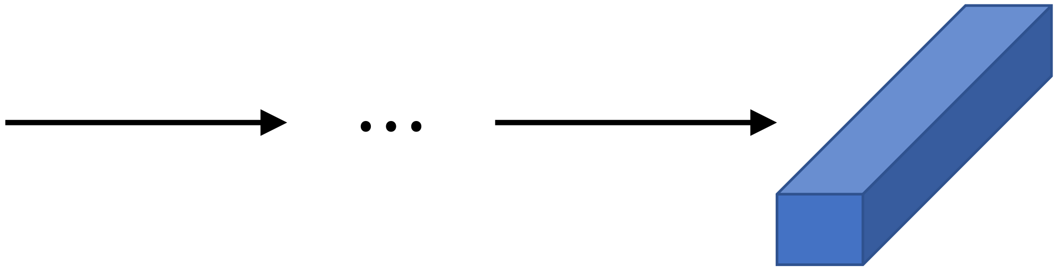
convnet will learn the magic trick

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

Andrew Ng

# Making predictions



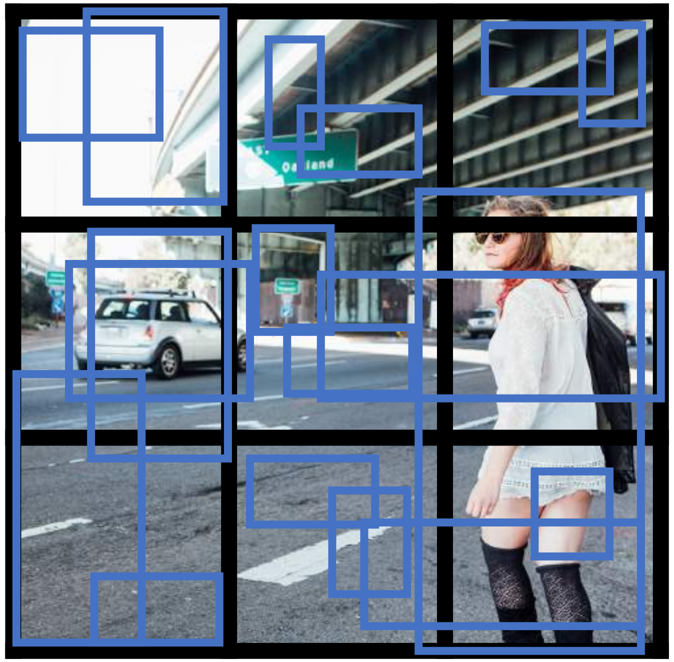


$3\times3\times2\times8$

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

Andrew Ng

# Outputting the non-max supressed outputs



- For each grid c~~a~~ell, get 2 predicted bounding boxes.

- Get rid of low probability predictions. many of these boxes will go away

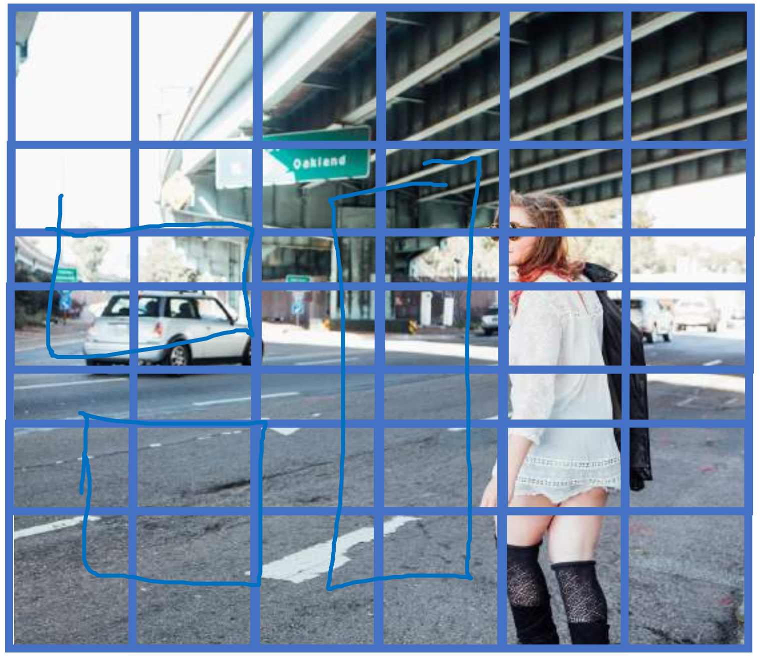
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

deeplearning.ai

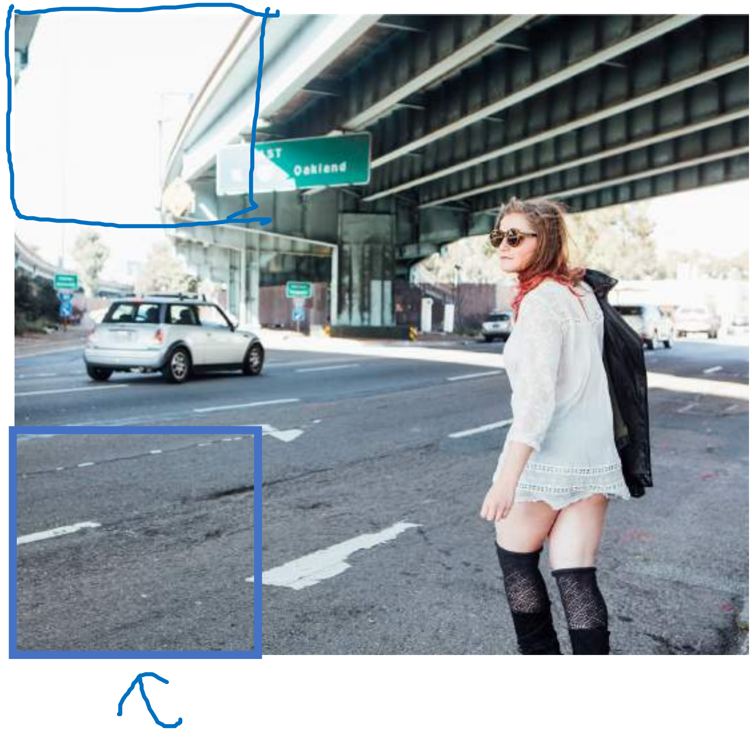
# Object Detection

## Region proposals (Optional)

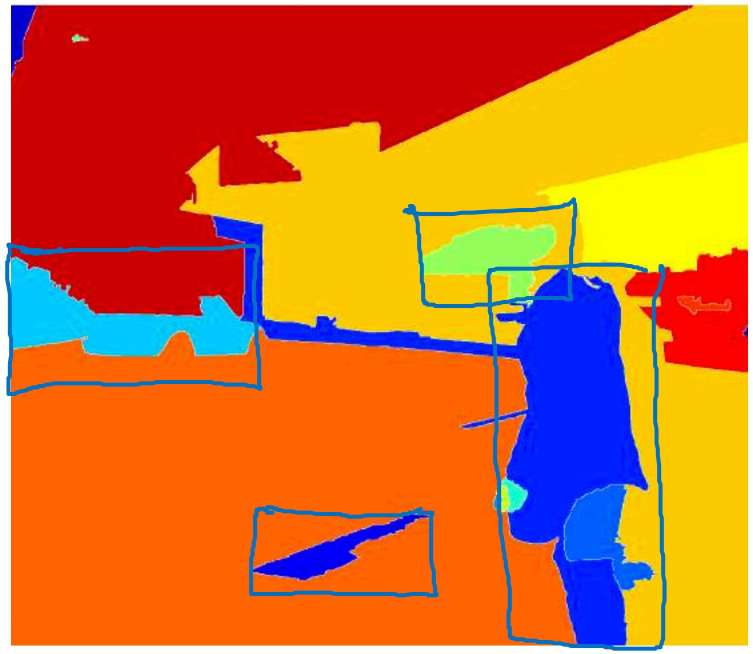
# Region proposal: R-CNN











Segmentation algorithm

~2,000

Instead of running convolution on full image, run a segmentation algorithm to identify interesting regions in the image, and then run convolution on boxes around those regions (~ find maybe 2000 blobs and run convolution on those)

[Girshik et. al, 2013, Rich feature hierarchies for accurate object detection and semantic segmentation] Andrew Ng

# Faster algorithms

Generally, these approaches are still slower than YOLO

R-CNN: Propose regions. Classify proposed regions one at a time. Output label + bounding box.

this approach is still slow

it predicts the bounding box, and doesn't just trust the bounding box it was given from segmentation

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]

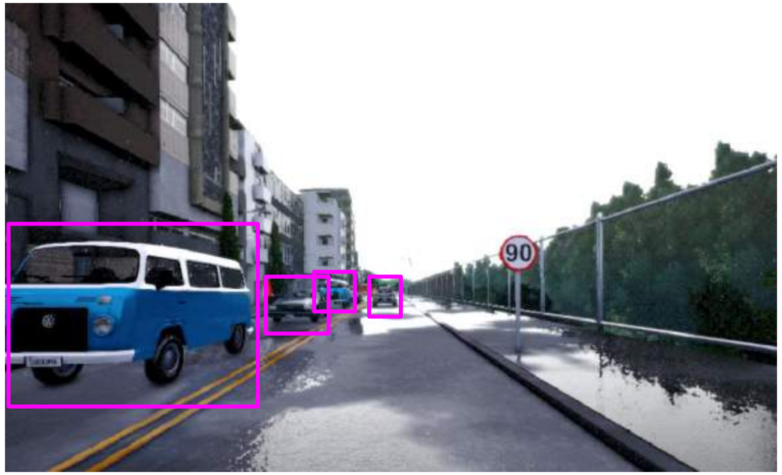
deeplearning.ai

# Convolutional Neural Networks

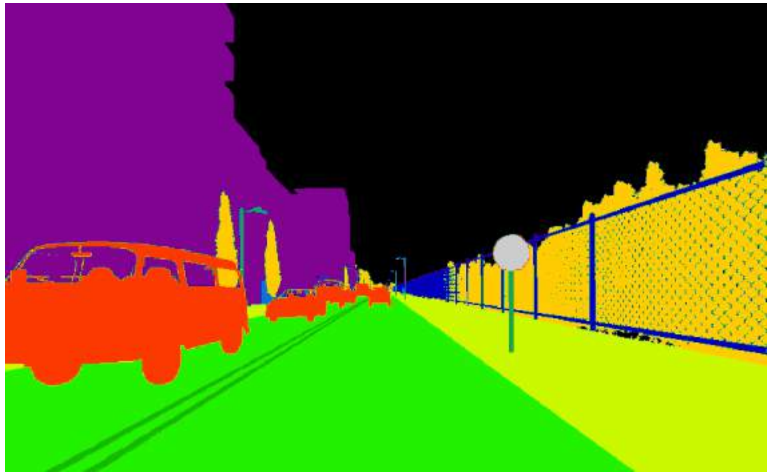
## Semantic segmentation with U-Net

# Object Detection vs. Semantic Segmentation

Input image



Object Detection



Semantic Segmentation

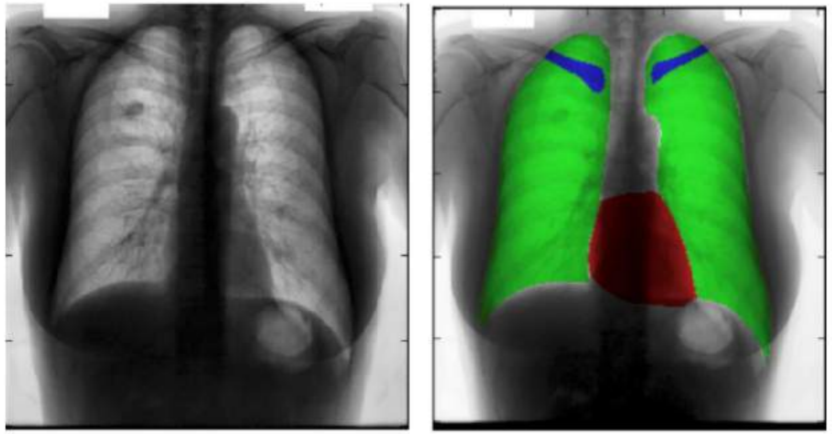
it attempts to segment each pixel - for example, what pixels represent drivable road

Andrew Ng

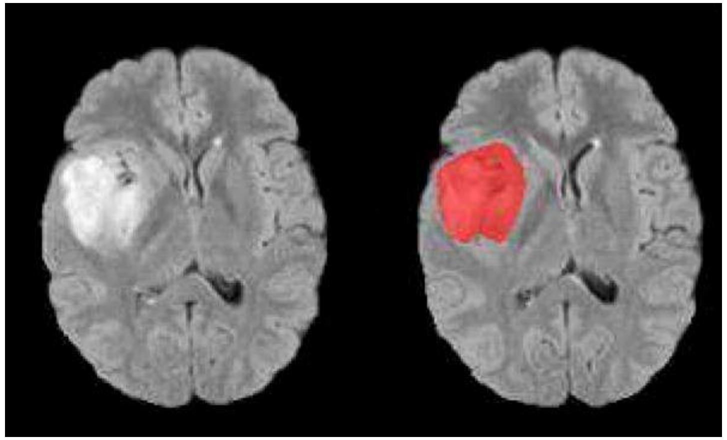
# Motivation for U-Net

algorithm highlights different
parts - heart, lungs, clavicle

the algorithm highlights the area of tumor - which is
useful for surgery planning



Chest X-Ray



Brain MRI

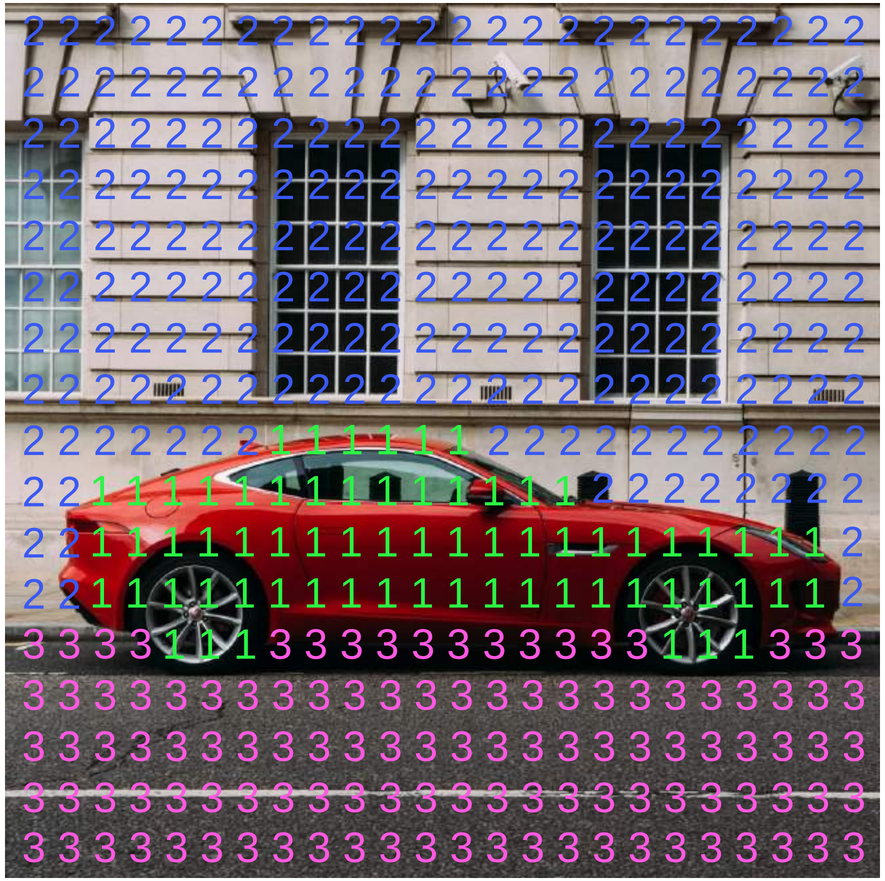
[Novikov et al., 2017, Fully Convolutional Architectures for Multi-Class Segmentation in Chest Radiographs]
[Dong et al., 2017, Automatic Brain Tumor Detection and Segmentation Using U-Net Based Fully Convolutional Networks ]

# Per-pixel class labels


1. Car
0. Not Car

# Per-pixel class labels

Much more complex output in this case
- not just giving object classification (0/1)
- or, just classification & bounding box
but a whole matrix of output



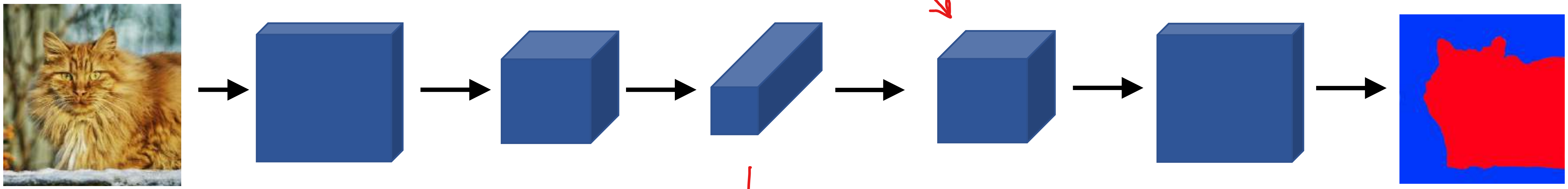
1. Car
2. Building
3. Road

| | | | | | | | | | | | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 2 | 2 | 2 | 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| 2 | 2 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 |
| 3 | 3 | 3 | 3 | 1 | 1 | 1 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 1 | 1 | 1 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |
| 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 | 3 |

Segmentation Map

# Deep Learning for Semantic Segmentation

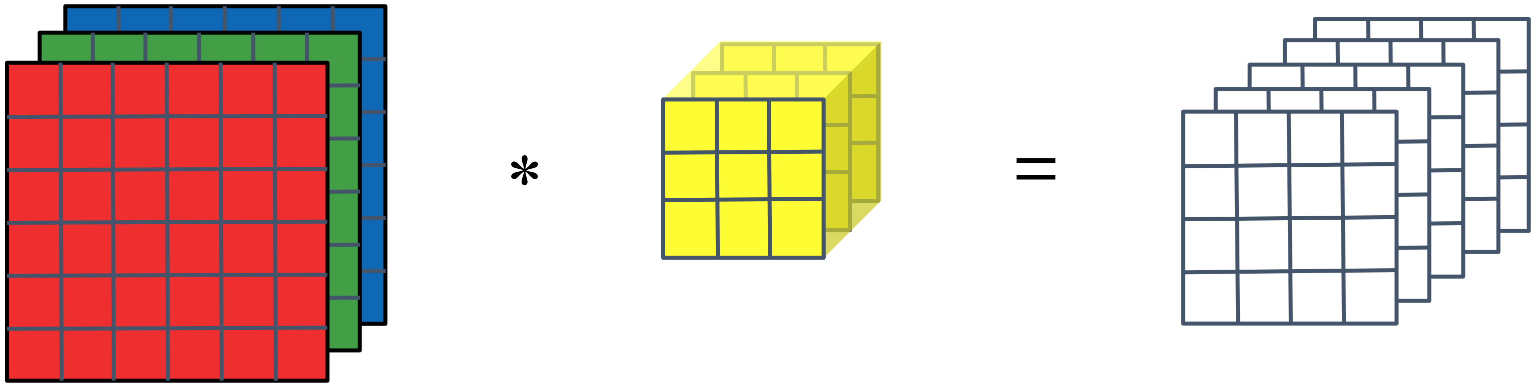
in U-net, volume will again start to go up
but how? answer - transpose convolution. predicting a class for each pixel



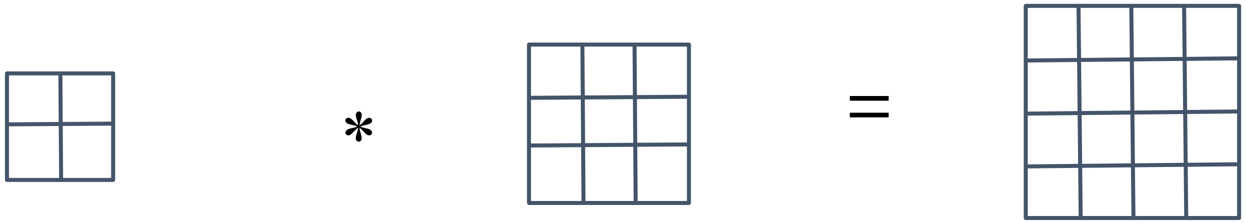
in previous architectures, volume goes down as we progress, until it is flattened and prepared for prediction

# Transpose Convolution

Normal Convolution

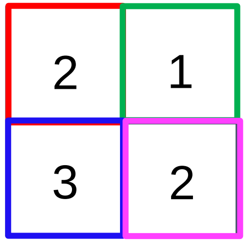


Transpose Convolution

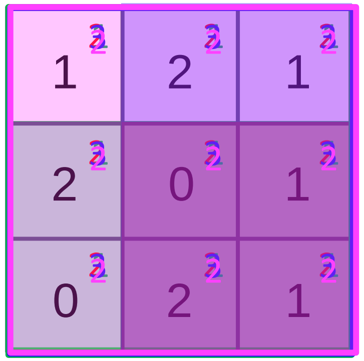


# Transpose Convolution

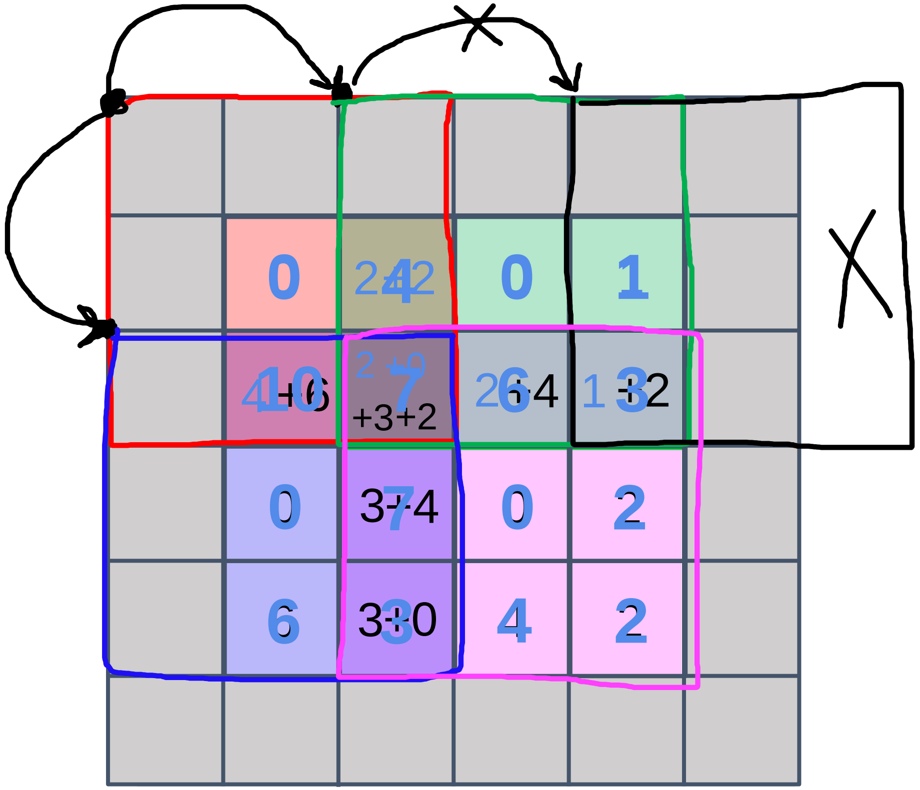
Multiple first value in A (=2) with Filter (B) and fill C (non-padded region)
- then repeat with A (=1), filling C after stride 2. Add values in overlapping boxes




2 x 2




weight filter



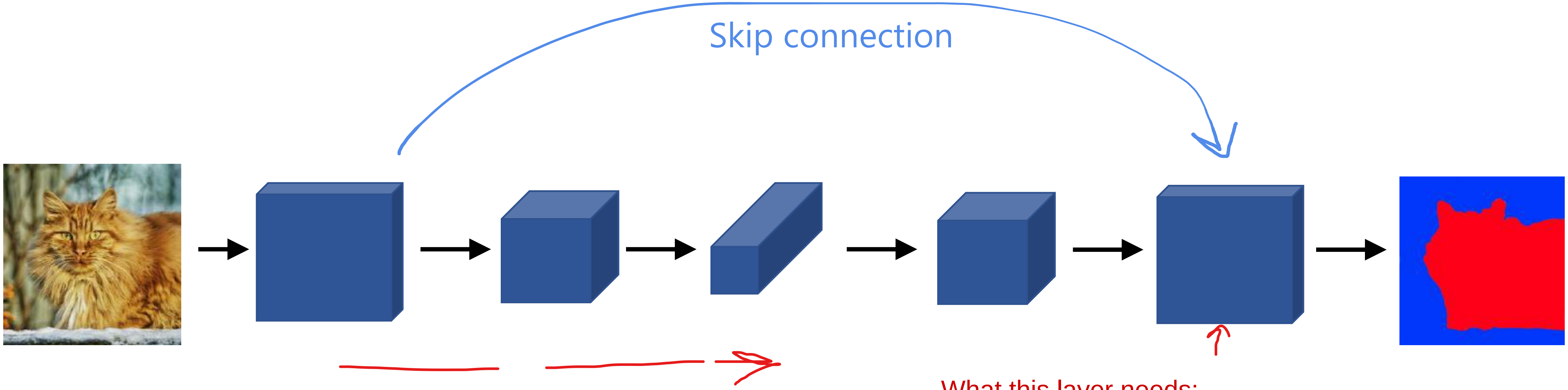

4 x 4

output:
0 4 0 1
10 7 6 3
0 7 0 2
6 3 4 2

filter f x f = 3 x 3 padding p = 1 stride s = 2

Andrew Ng

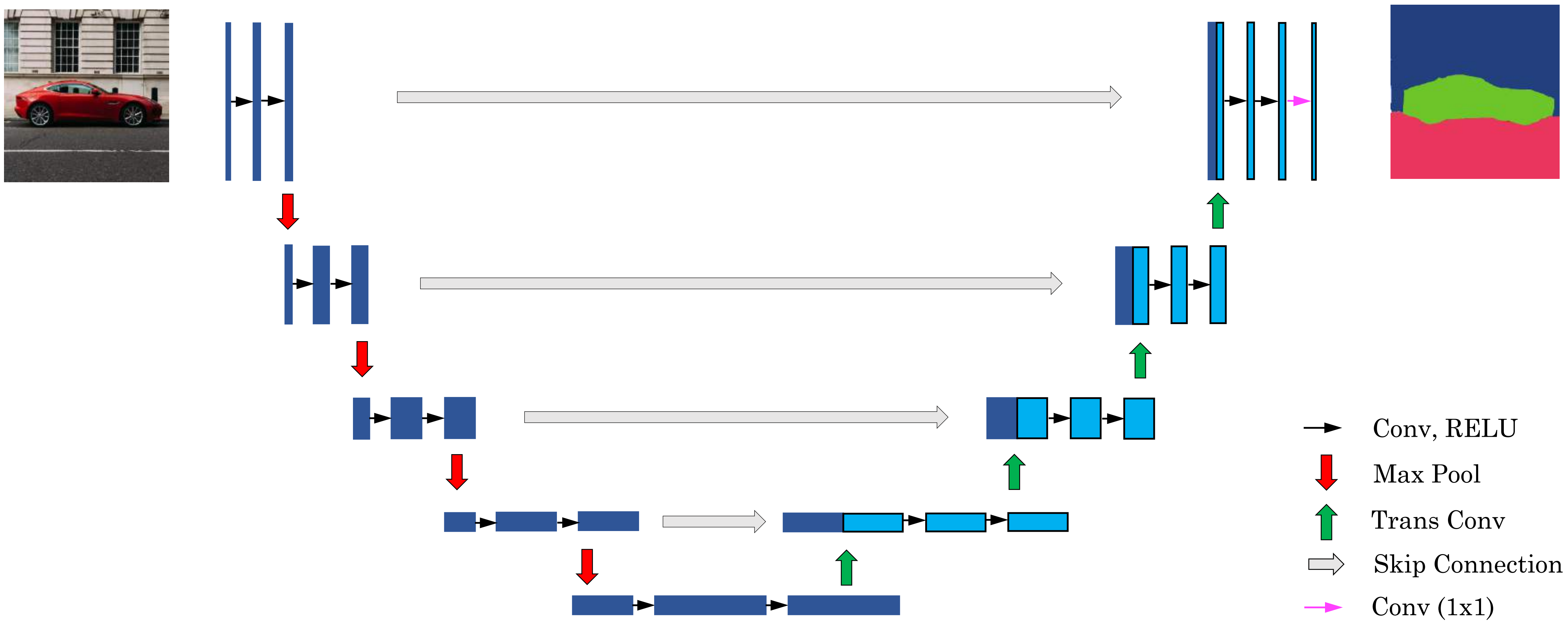
# Deep Learning for Semantic Segmentation




normal convolution:
we lose spatial information

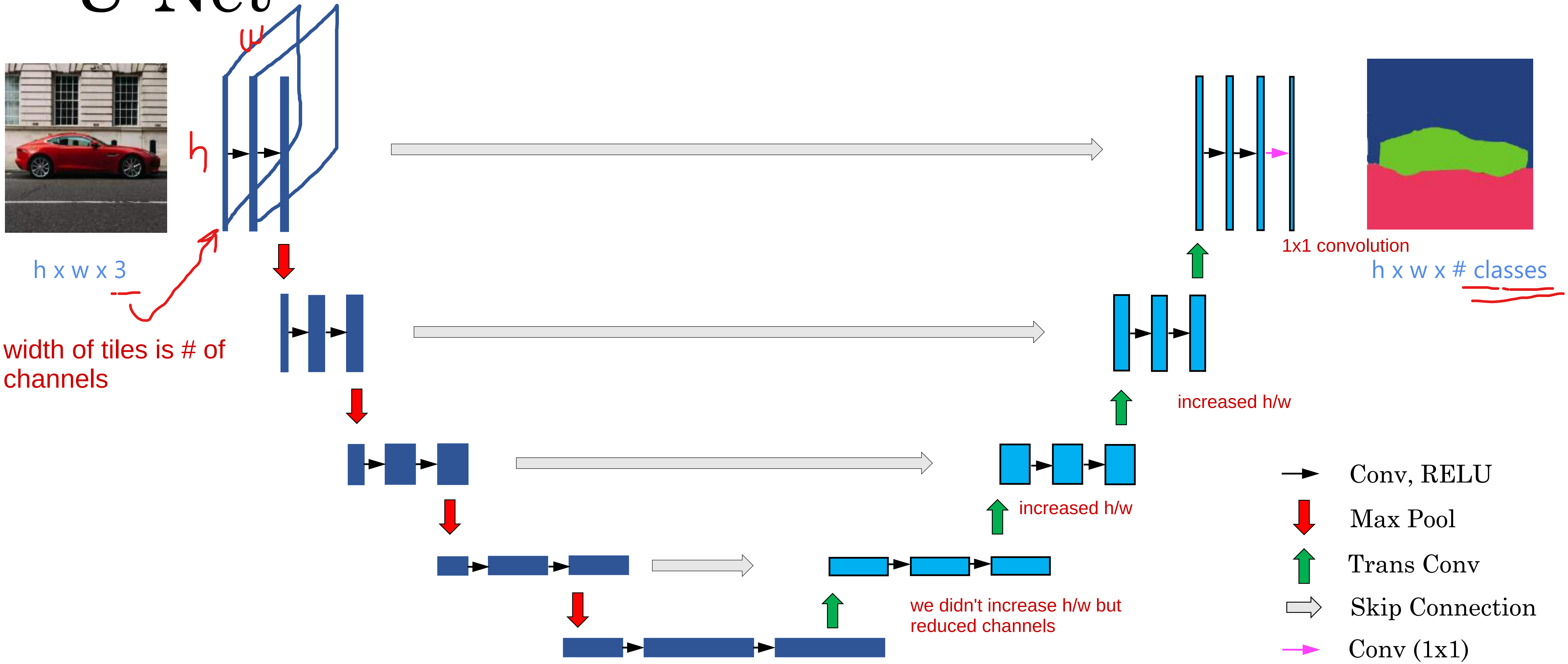
What this layer needs:
(i) high level contextual information (via previous layer)
(ii) detailed spatial / textured information (via skip)

# U-Net




[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

# U-Net




[Ronneberger et al., 2015, U-Net: Convolutional Networks for Biomedical Image Segmentation]

Andrew Ng

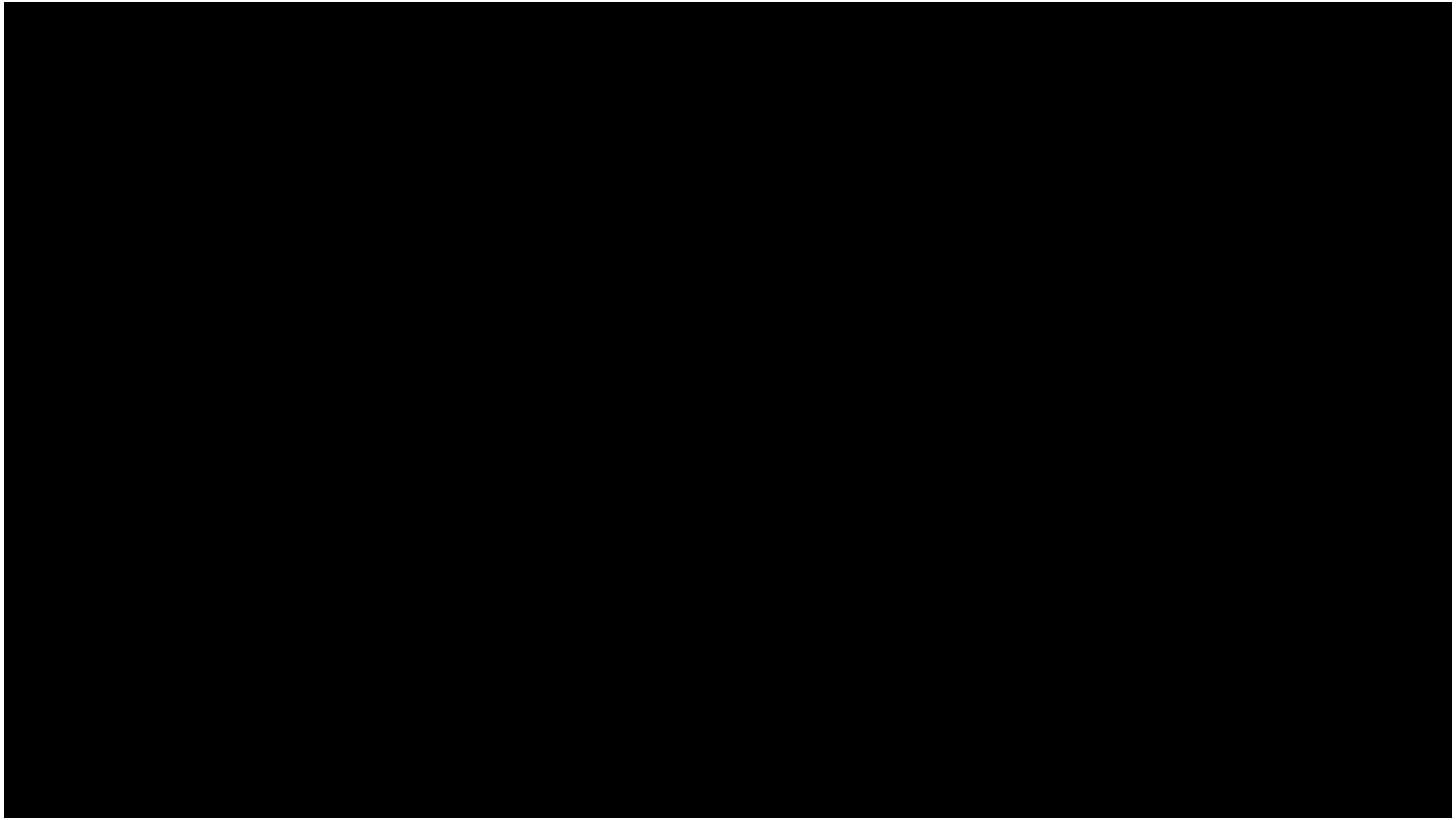
# Copyright Notice

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 ←

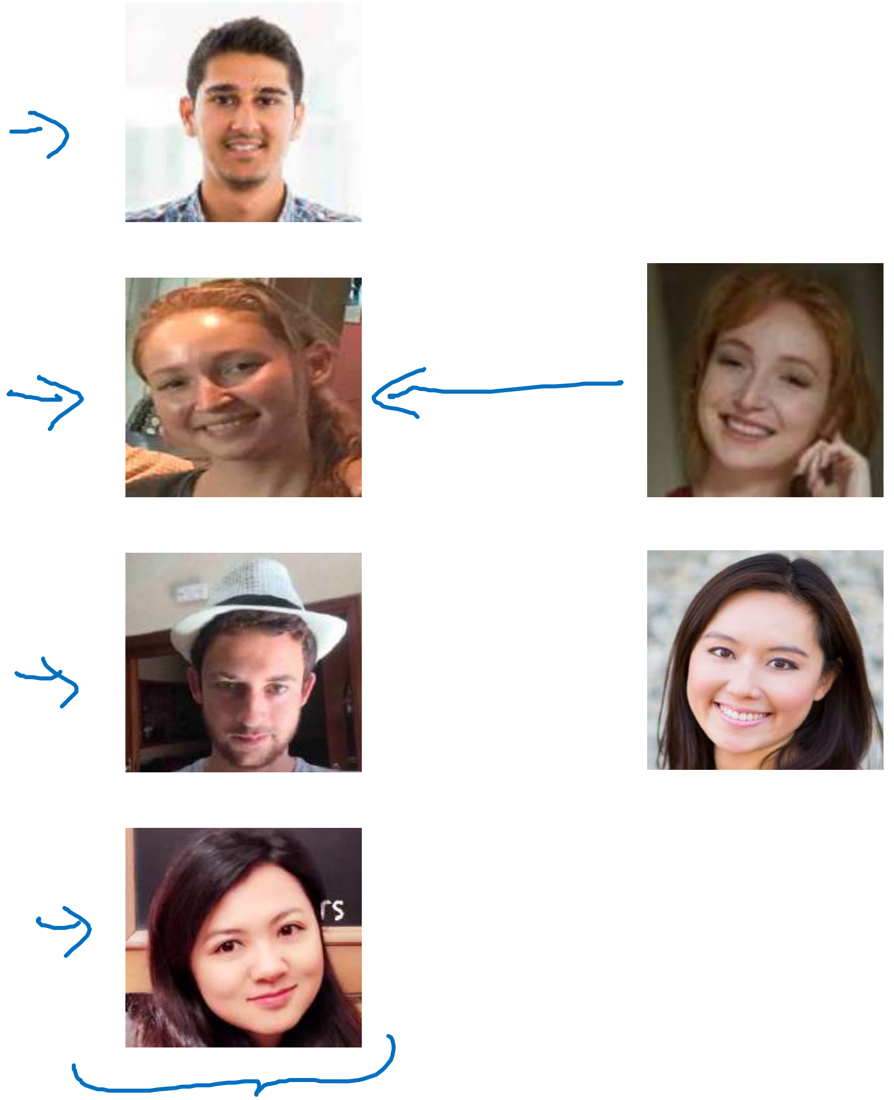
Andrew Ng

# Face recognition

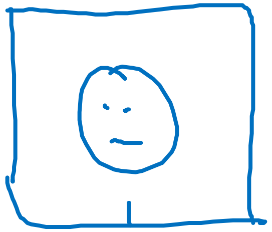
## One-shot learning

deeplearning.ai

# One-shot learning



Learning from one example to recognize the person again



→ CNN → o

softmax

~~5~~ (6)

not a good approach - will need to expand last layer and retrain network every time someone new joins the team

Andrew Ng

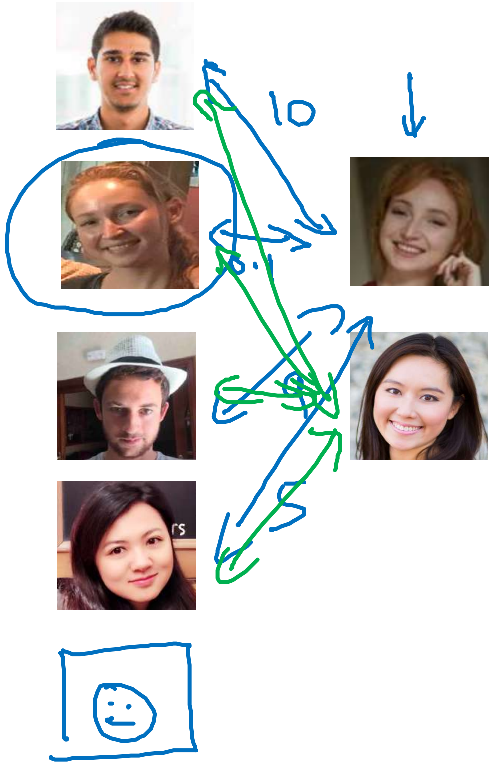
# Learning a “similarity” function

- d(img1,img2) = degree of difference between images

If d(img1,img2) $\le \tau$ "same"

$> \tau$ "different"

Verification.




d(img1, img2)

Key idea - learn a "similarity function"

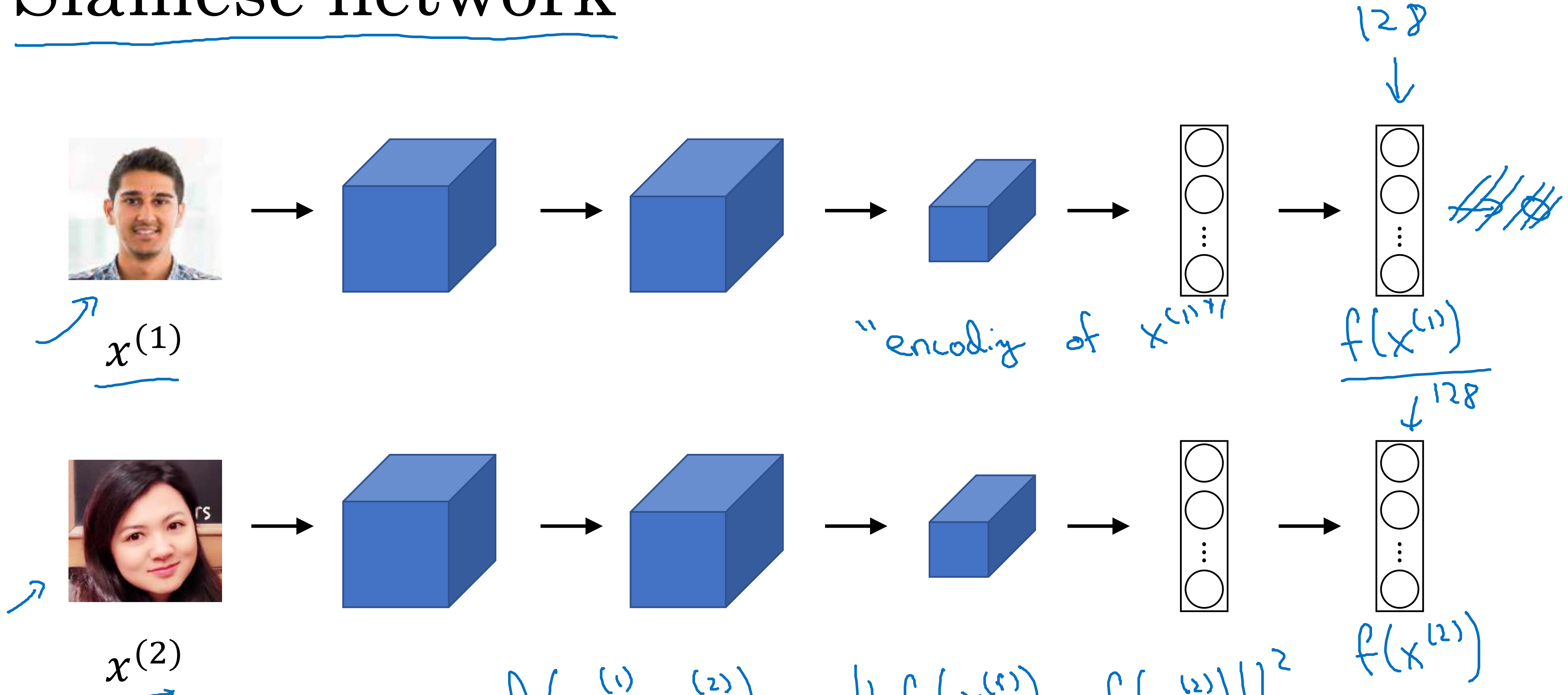
Andrew Ng

# Face recognition

deeplearning.ai

## Siamese network

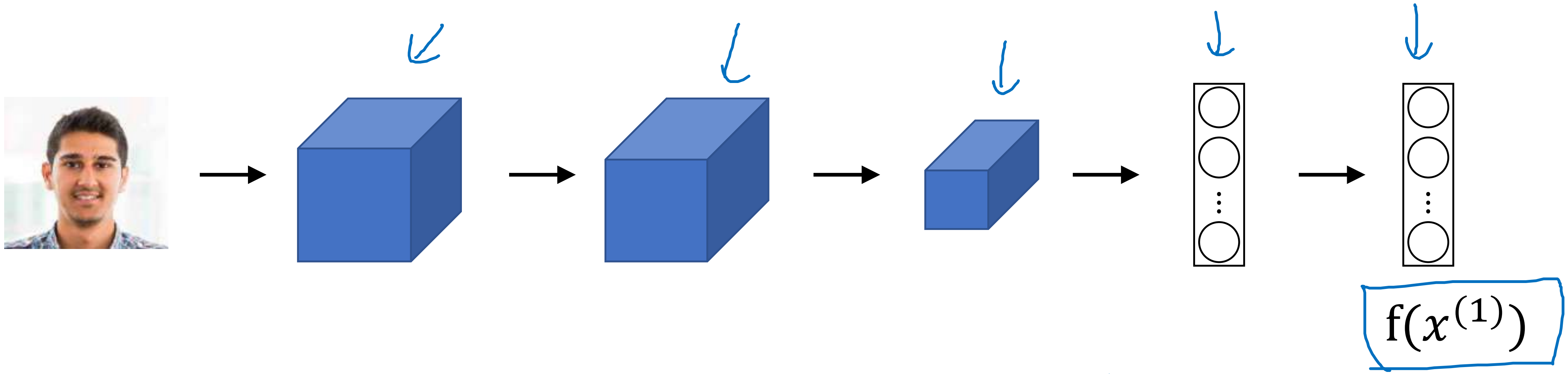
# Siamese network




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

[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, $\left\|f(x^{(i)}) - f(x^{(j)})\right\|^2$ is small.

If $x^{(i)}, x^{(j)}$ are different persons, $\left\|f(x^{(i)}) - f(x^{(j)})\right\|^2$ is large.

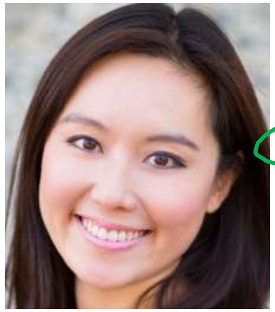
# Face recognition

deeplearning.ai

## Triplet loss

# Learning Objective

Triplet loss - you are looking at 3 images at each
alpha (margin) is added to ensure 'trivial' solution of 0 is eliminated




Anchor Positive

A P

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


Anchor Negative

A N

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

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]

Andrew Ng

# Loss function

- Loss is either 'positive' or 'zero'
- if it is negative, we've reached out objective and no longer want to optimize

Given 3 images $A, P, N$:

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

$\leq 0$ $> 0$

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

$A, P$

Training set: 10k pictures of 1k persons

you need >1 picture for each person for training

[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,
$d(A,P) + \alpha \leq d(A,N)$ is easily satisfied.

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

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

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

$\underline{d(A,P)} \approx \underline{d(A,N)}$

$\downarrow \qquad \uparrow$

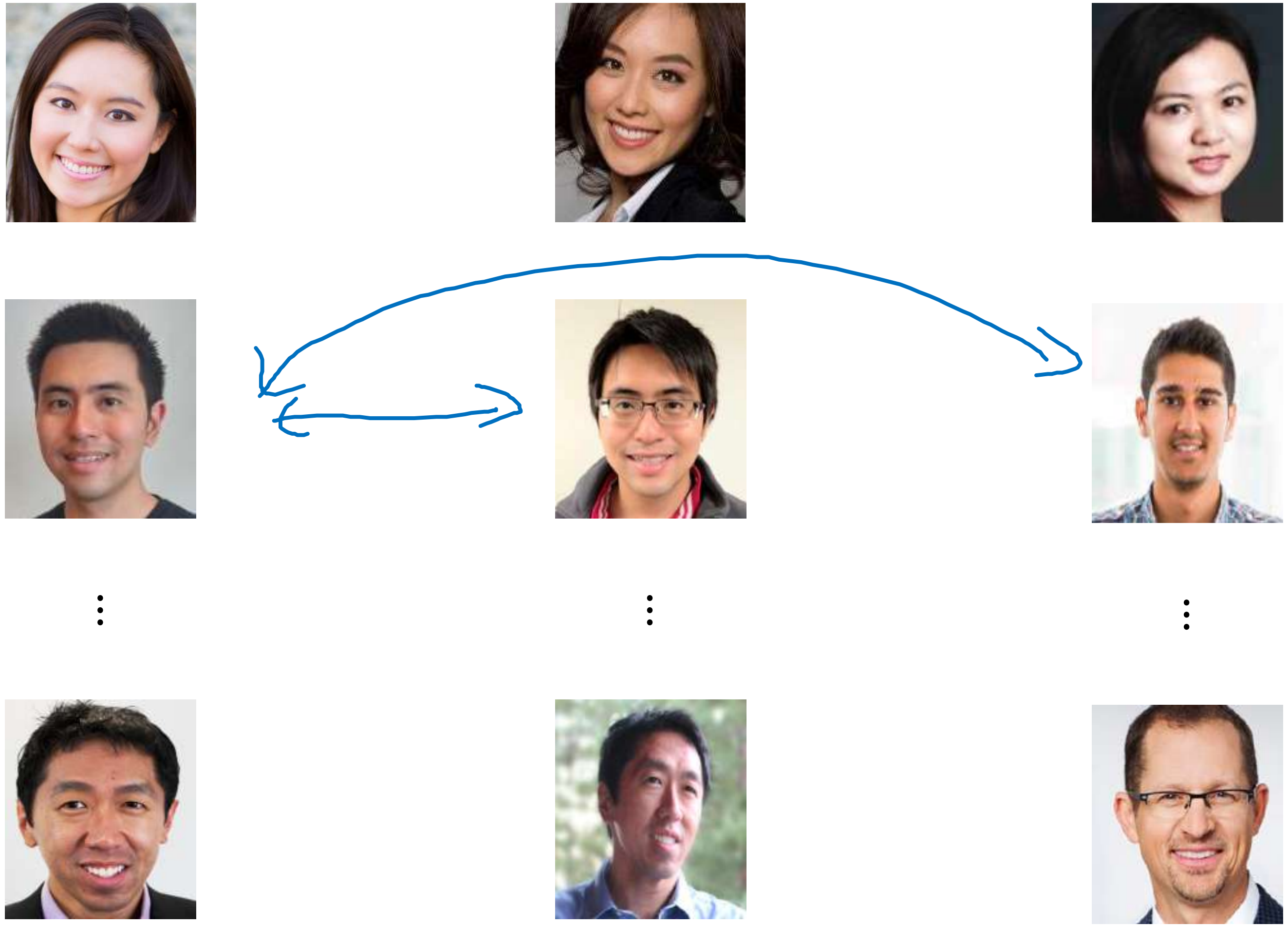
<u>Face</u> Net

Deep <u>Face</u>

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

# Training set using triplet loss

Anchor | Positive | Negative



$J$

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

Andrew Ng

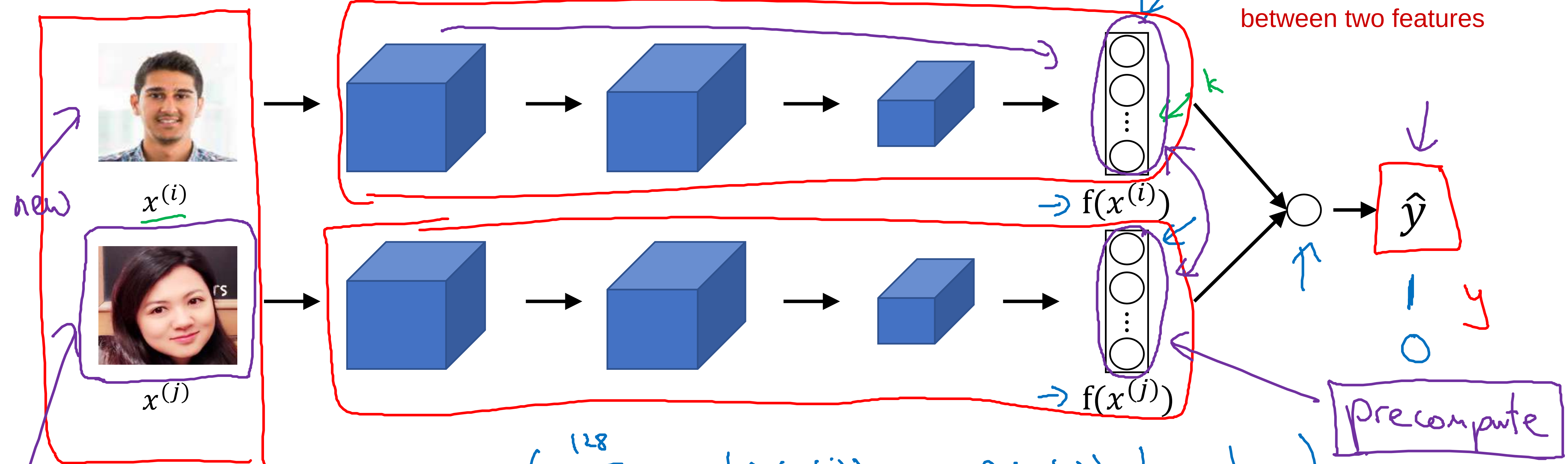
deeplearning.ai

# Face recognition

## Face verification and binary classification

# Learning the similarity function

Input to logistic regression can be element-wise difference between two features




Database

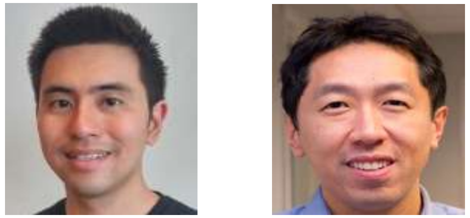
$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k \underbrace{\left| f(x^{(i)})_k - f(x^{(j)})_k \right|}_{\frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k}} + b\right)$$

$\chi^2$

[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

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)

[Images generated by Justin Johnson]

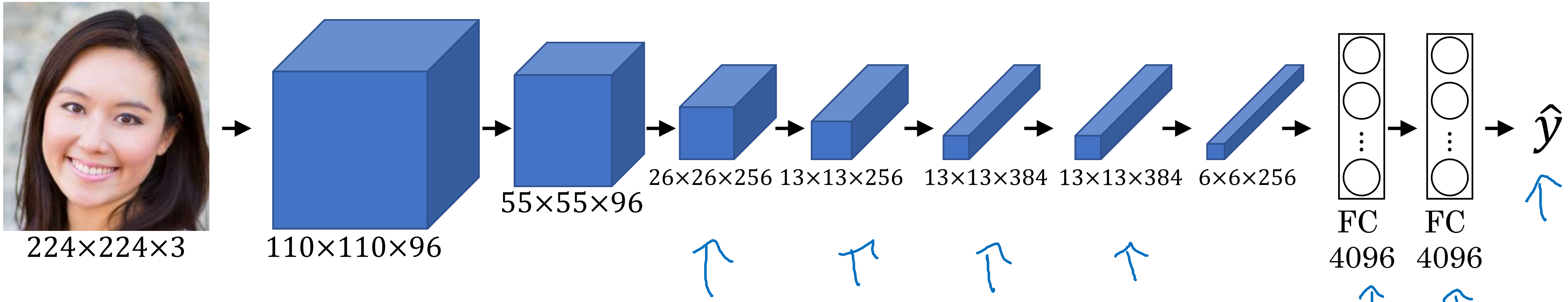
Andrew Ng

# Neural Style Transfer

deeplearning.ai

## What are deep ConvNets learning?

# Visualizing what a deep network is learning

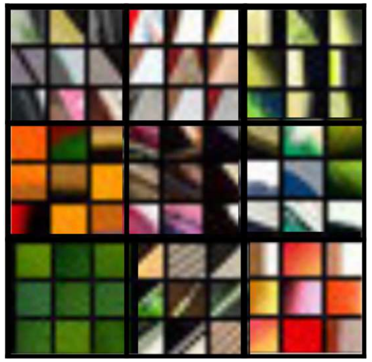



Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

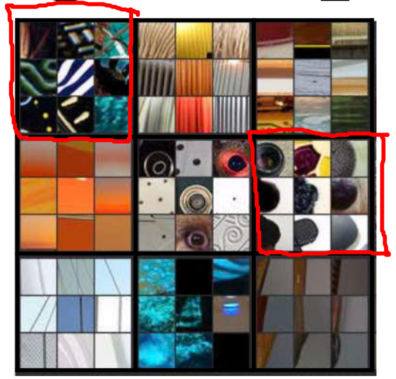
Repeat for other units.

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

# Visualizing deep layers



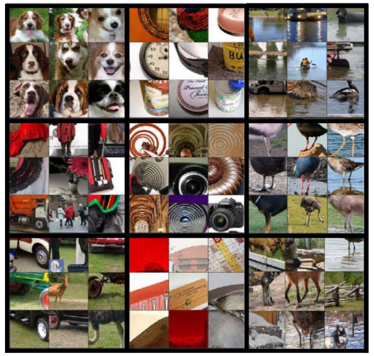
Layer 1



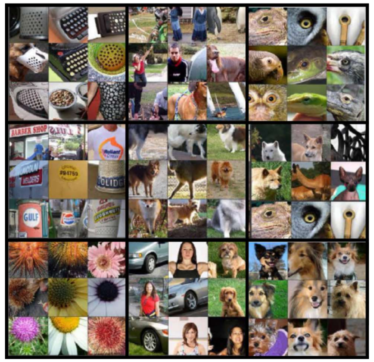
Layer 2



Layer 3



Layer 4

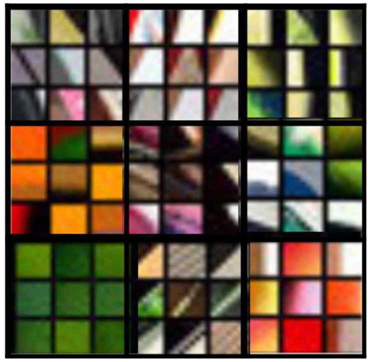




Layer 5

Andrew Ng

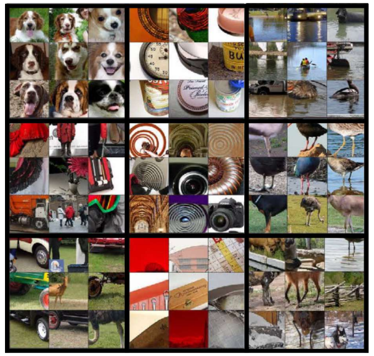
# Visualizing deep layers: Layer 1



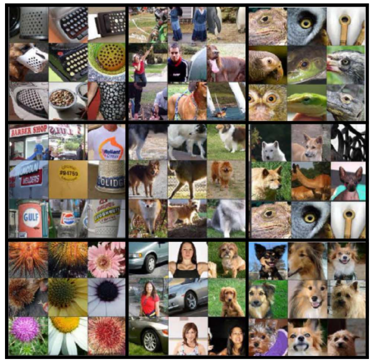
Layer 1

Layer 2

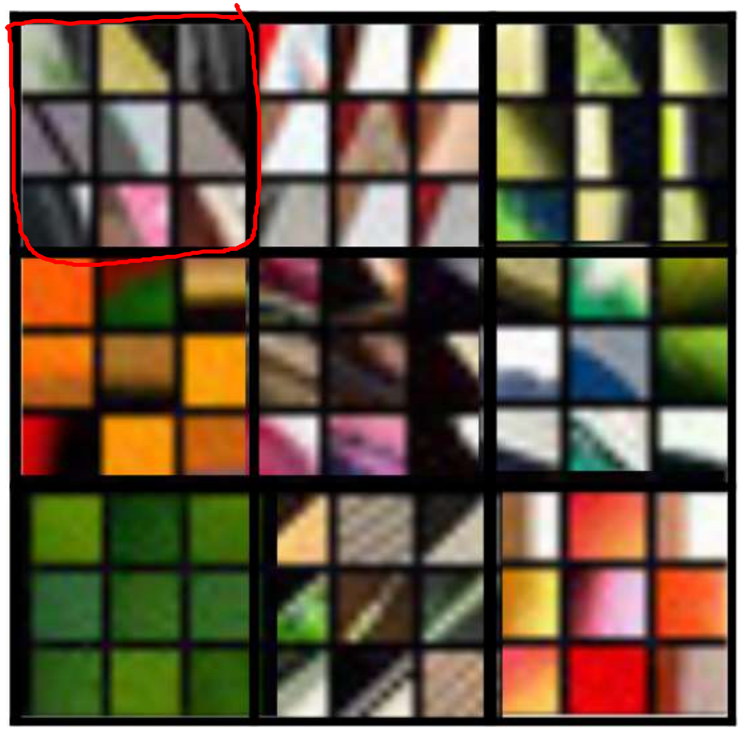
Layer 3

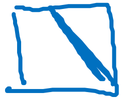


Layer 4

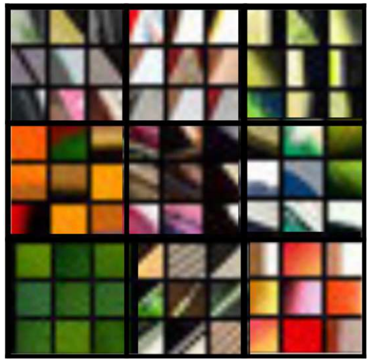


Layer 5





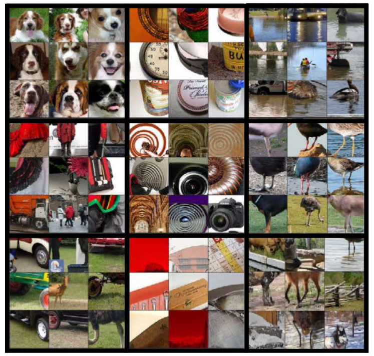
# Visualizing deep layers: Layer 2



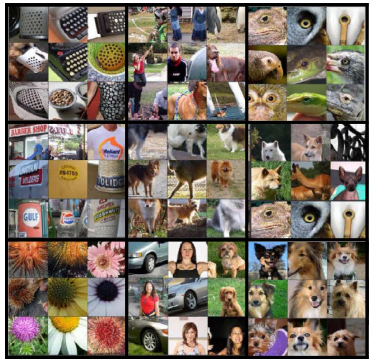
Layer 1

Layer 2

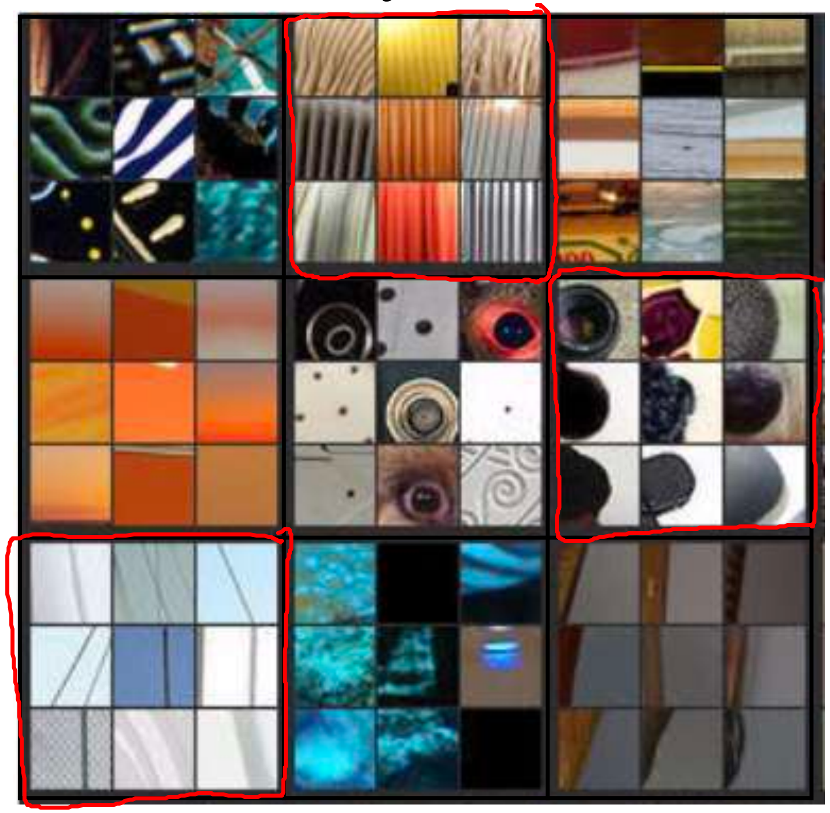
Layer 3



Layer 4

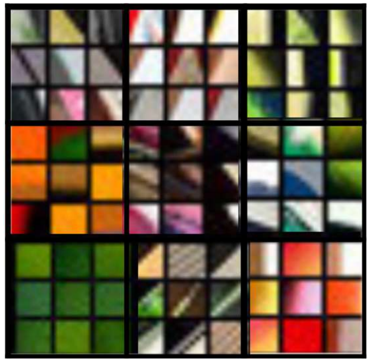


Layer 5



Andrew Ng

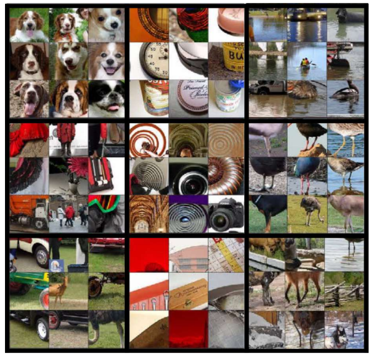
# Visualizing deep layers: Layer 3



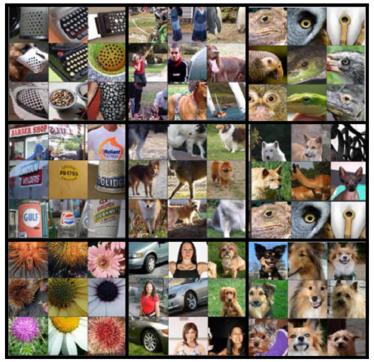
Layer 1

Layer 2

Layer 3



Layer 4



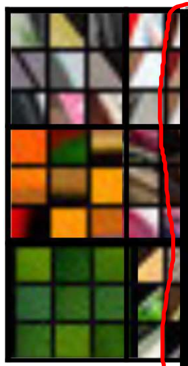
Layer 5

Andrew Ng

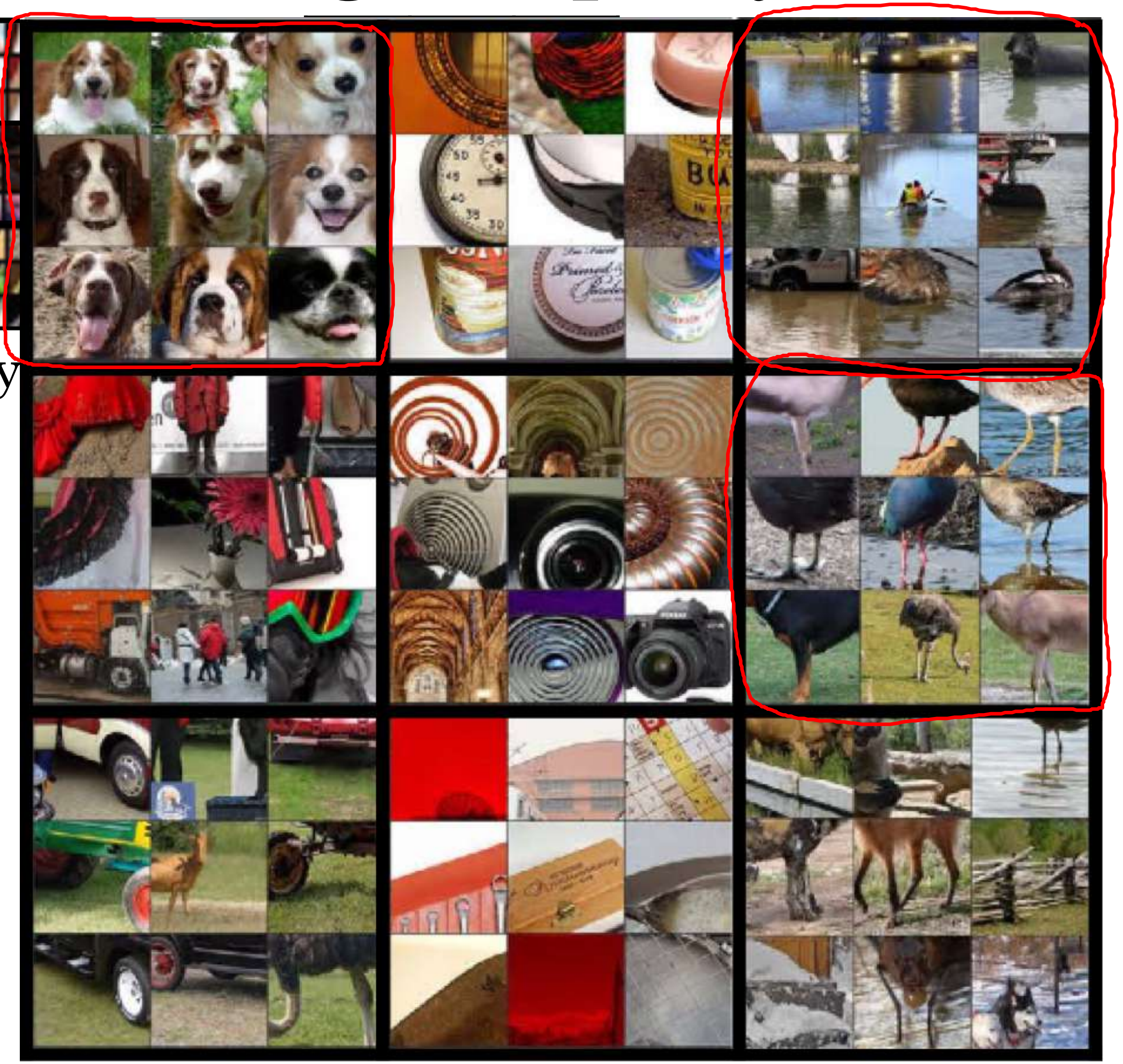
# Visualizing deep layers: Layer 3

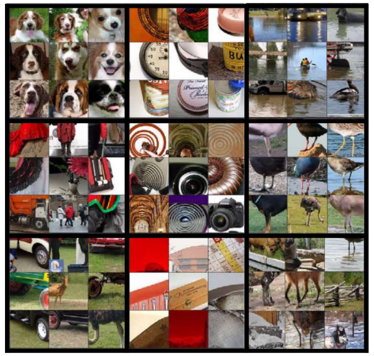

Andrew Ng

# Visualizing deep layers: Layer 4

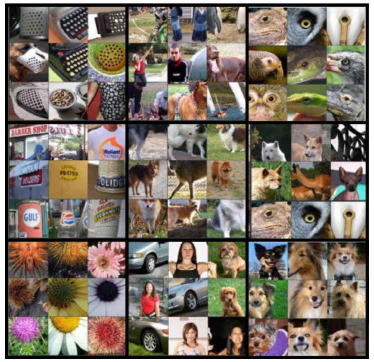


Lay





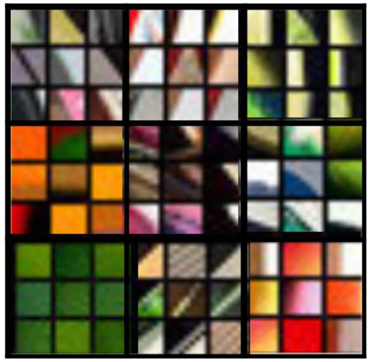
Layer 4



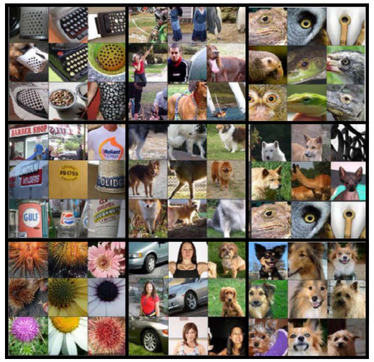
Layer 5

Andrew Ng

# Visualizing deep layers: Layer 5



Layer 1

Layer 5

Andrew Ng

# Neural Style Transfer

## Cost function

deeplearning.ai

# Neural style transfer cost function

Content C

Style S

Generated image G

We define two parts to the cost function: Content & Style

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

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

# Find the generated image G

1. Initiate G randomly

   start with white-noise image

   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]

Andrew Ng

deeplearning.ai

# Neural Style Transfer

## Content cost function

# Content cost function

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

'l' layer is neither too shallow nor too deep

- 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

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

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

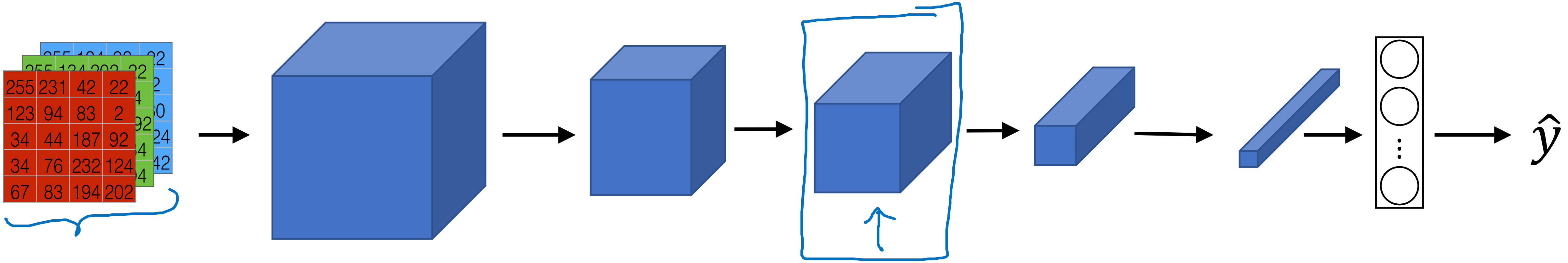
Andrew Ng

deeplearning.ai

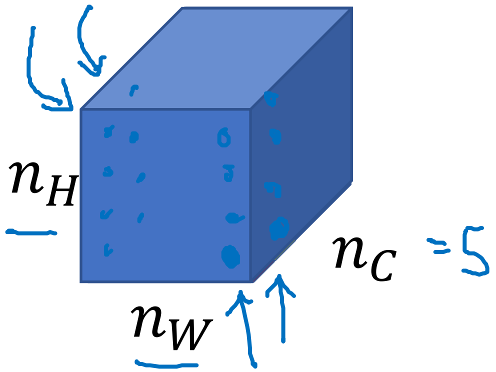
# Neural Style Transfer

## 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.



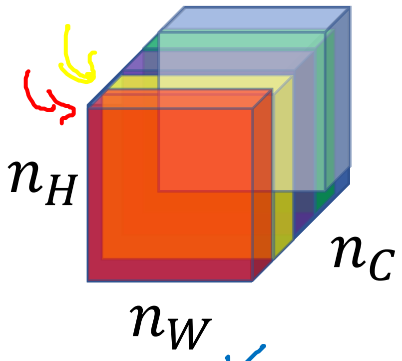

How correlated are the activations across different channels?

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

Andrew Ng

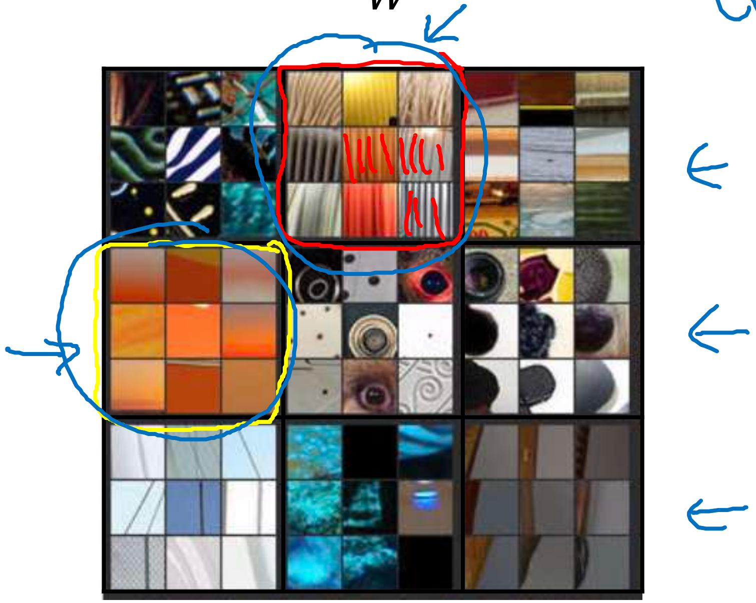
# Intuition about style of an image

## Style image

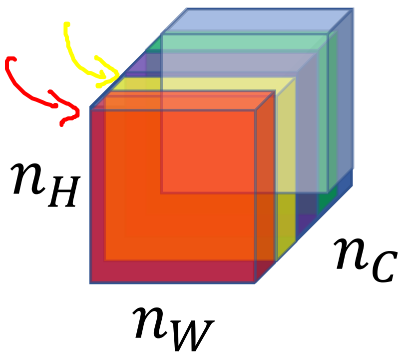



Correlated?

Uncorrelated



## Generated Image




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

# Style matrix

$\begin{matrix} H & W & C \\ \downarrow & \downarrow & \swarrow \end{matrix}$

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

$n_c$

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

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

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

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

"Gram matrix"

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

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

$\uparrow$

$\beta$

[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]

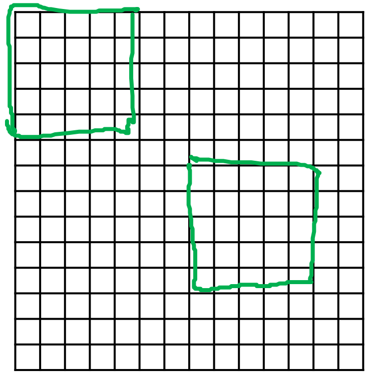
Andrew Ng

deeplearning.ai

# Convolutional Networks in 1D or 3D

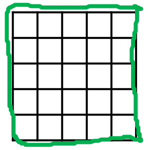
## 1D and 3D generalizations of models

# Convolutions in 2D and 1D

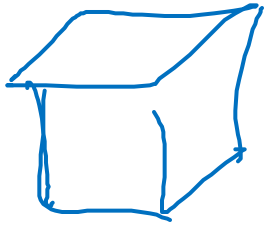


2D input image
14×14

*



2D filter
5×5

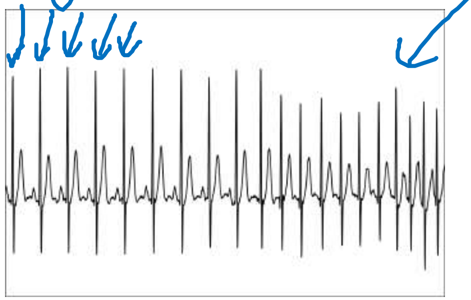


$14 \times 14 \times 3 \ * \ 5 \times 5 \times 3$

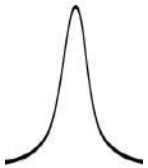
$\rightarrow 10 \times 10 \times 16$

$10 \times 10 \times 16 \ * \ 5 \times 5 \times 16$

$\rightarrow 6 \times 6 \times 32$



*



| | | | | | | | |
|---|---|---|---|---|---|---|---|
| 1 | 20 | 15 | 3 | 18 | 12 | 4 | 17 |

14

| | | | | |
|---|---|---|---|---|
| 1 | 3 | 10 | 3 | 1 |

5

$14 \times 1 \ * \ 5 \times 1$

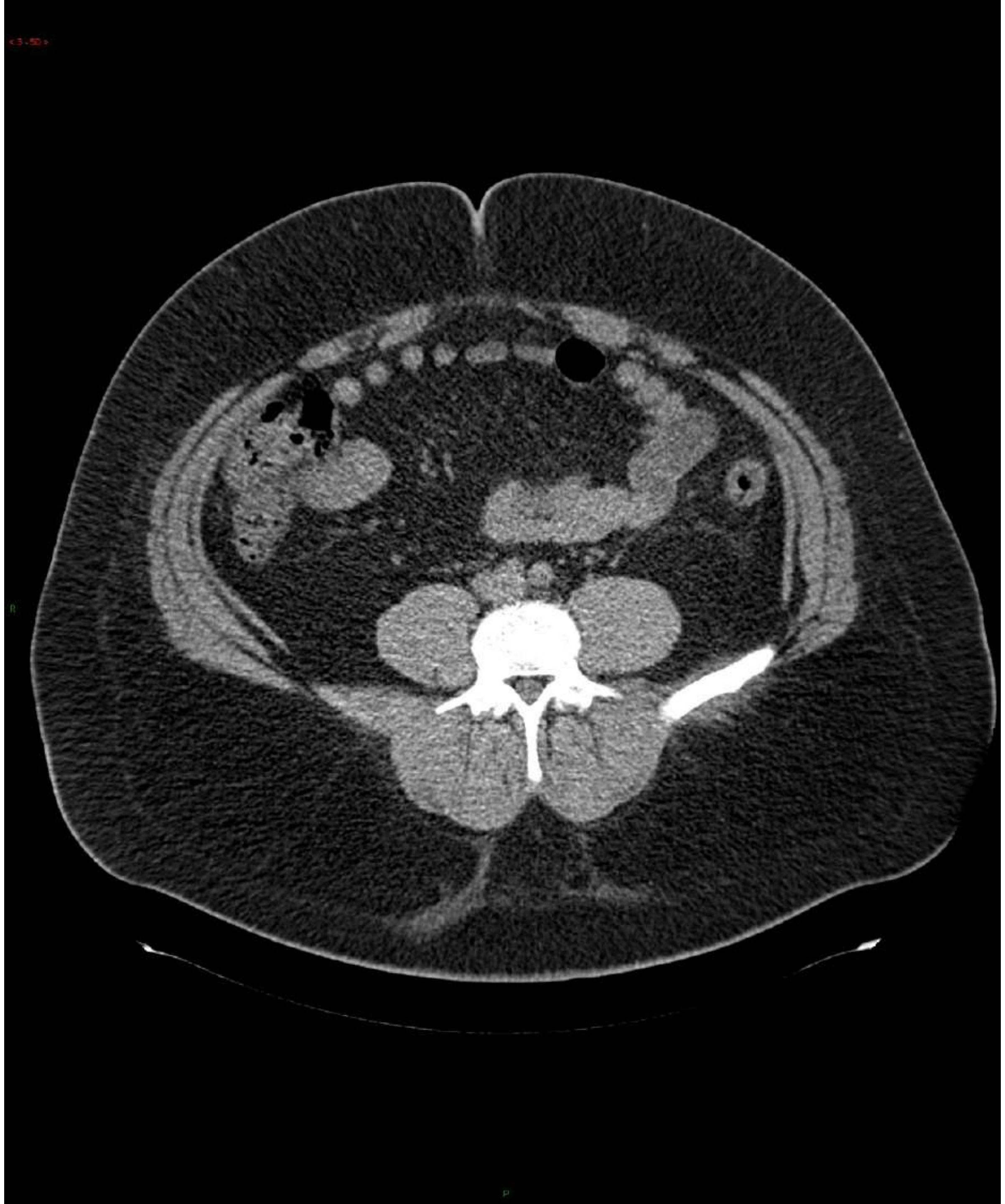
$\rightarrow 10 \times 16$

$10 \times 16 \ * \ 5 \times 16$

$\rightarrow 6 \times 32$

Andrew Ng

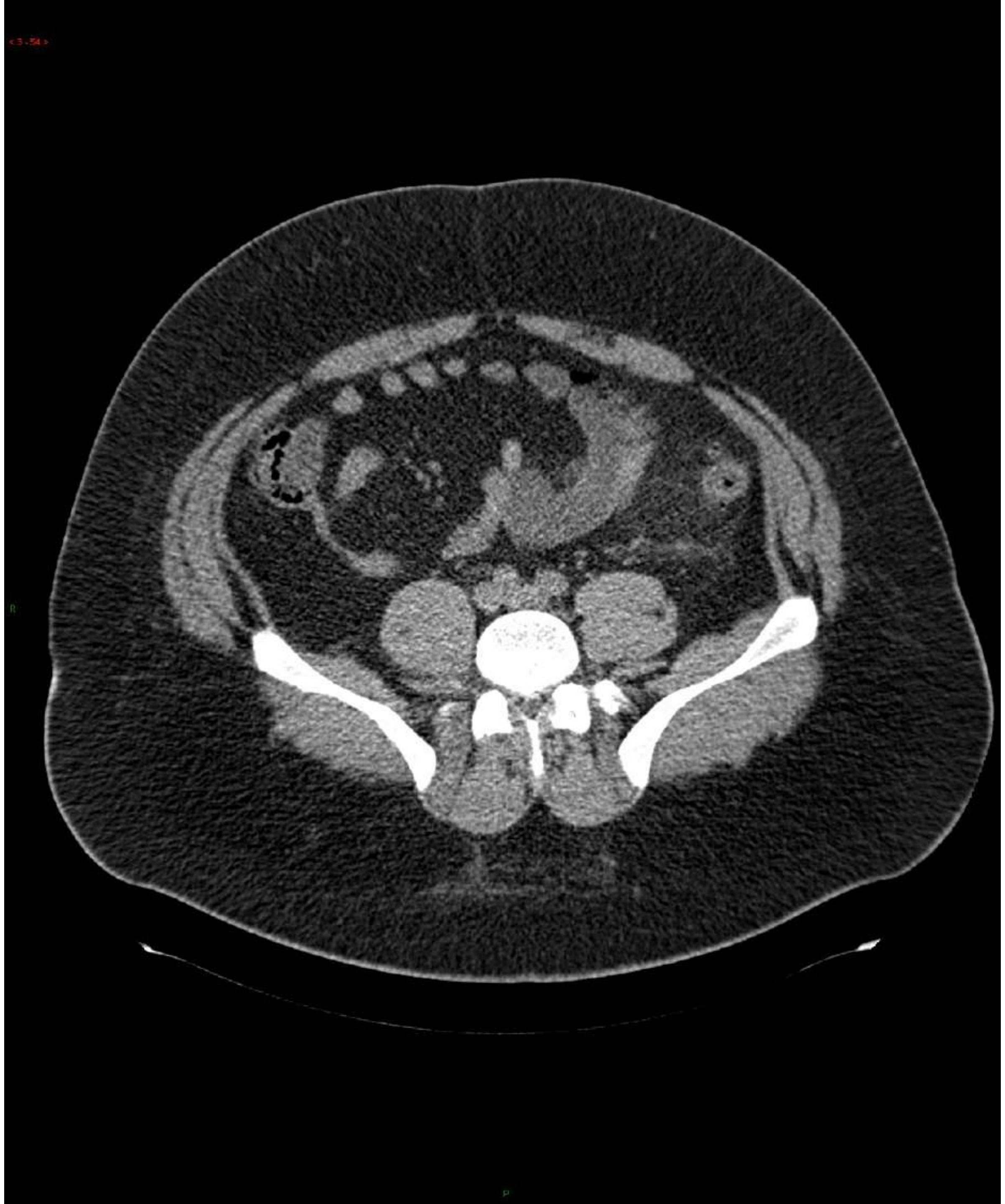
# 3D data



# 3D data

Andrew Ng

# 3D data

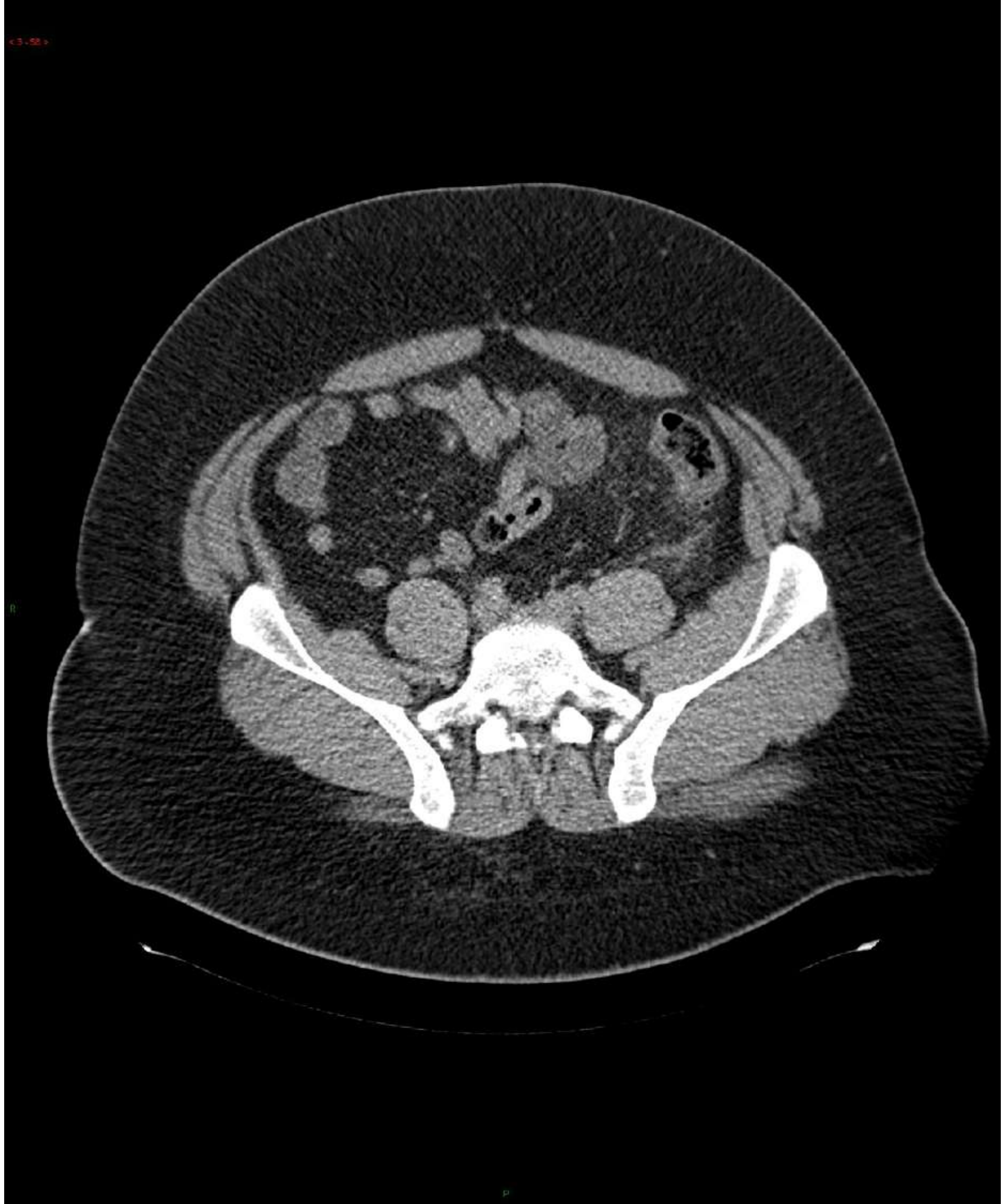


Andrew Ng

# 3D data

Andrew Ng

# 3D data



# 3D data

Andrew Ng

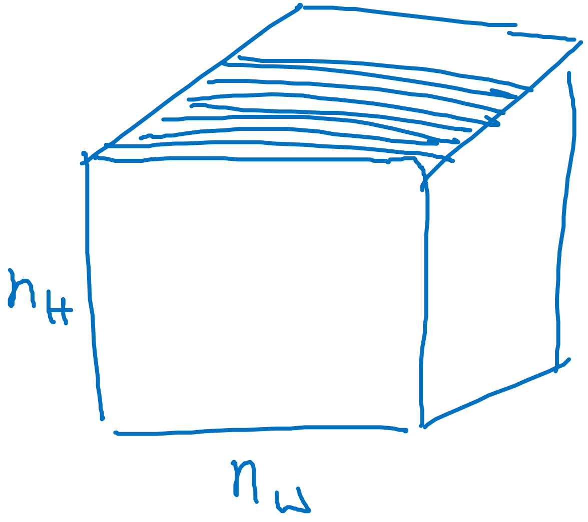
# 3D data

# 3D data

# 3D data

# 3D data

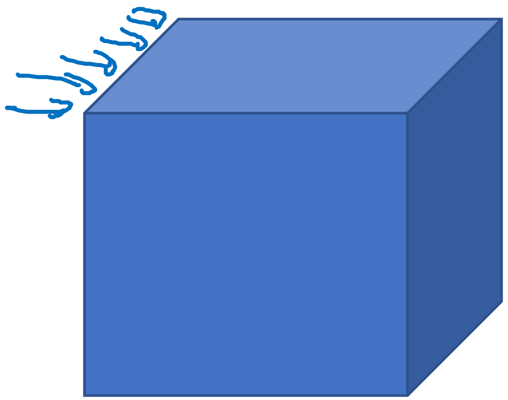
# 3D data



# 3D convolution

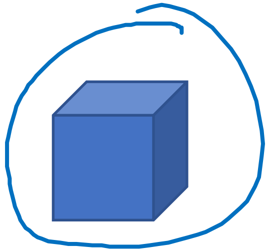
# of channels can be considered 4th dimension

3D filter, with 3rd dimension not same as 3rd dimension of input volume



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

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