Copyright Notice

deeplearning.ai

Convolutional Neural Networks

Computer vision

Computer Vision Problems

Image Classification

64x64

Cat? (0/1)

Neural Style Transfer

Object detection

Deep Learning on large images

64x64 x 3

Cat? (0/1)

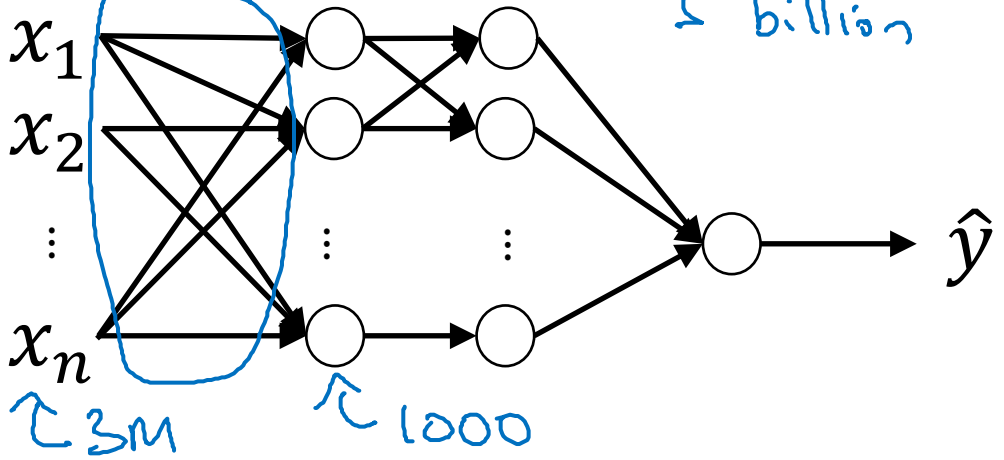
12288

1000 x 1000 x 3
= 3 million

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

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

3 billion



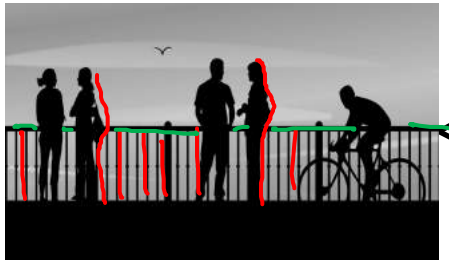
Andrew Ng

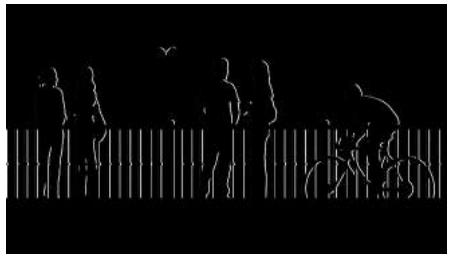
deeplearning.ai

Convolutional Neural Networks

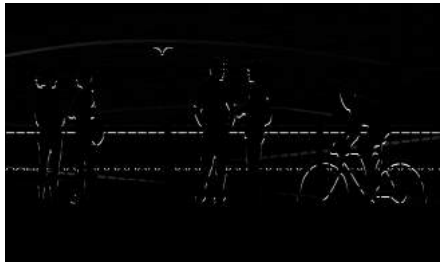
Edge detection example

Computer Vision Problem





vertical edges



horizontal edges

Vertical edge detection

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

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 x 6

"convolution"

*

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

3 x 3

→ filter kernel

=

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

4 x 4

Vertical edge detection

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

6 x 6

*

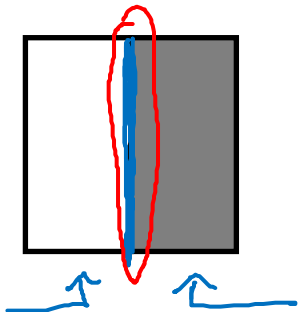
1	0	-1
1	0	-1
1	0	-1

3 x 3

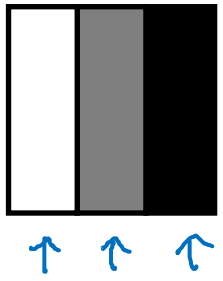
=

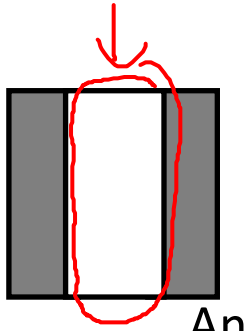
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

4 x 4



*





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

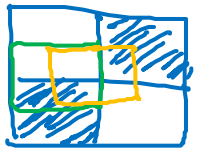
Horizontal

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

3×3

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

=

45°
70°
73°

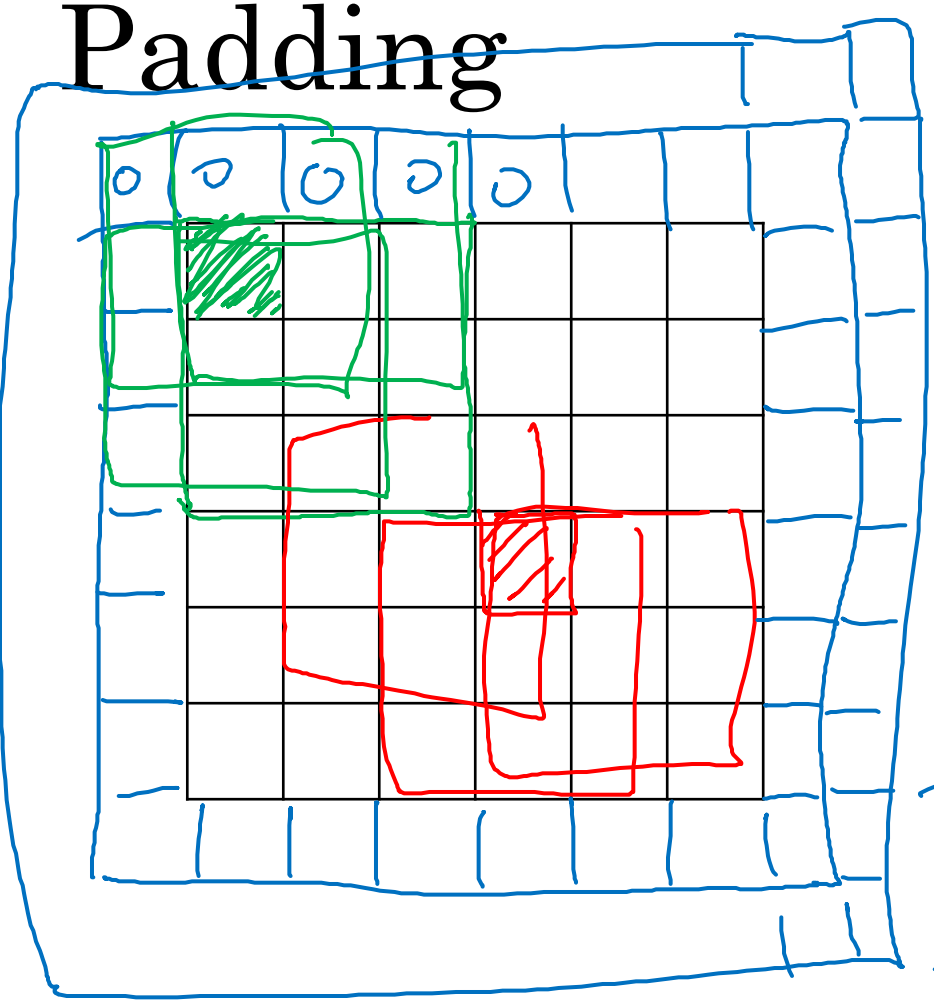
Andrew Ng

deeplearning.ai

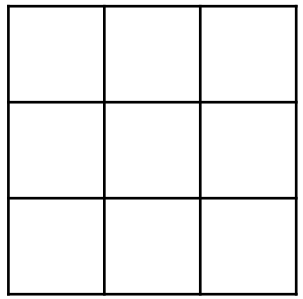
Convolutional Neural Networks

Padding

Padding



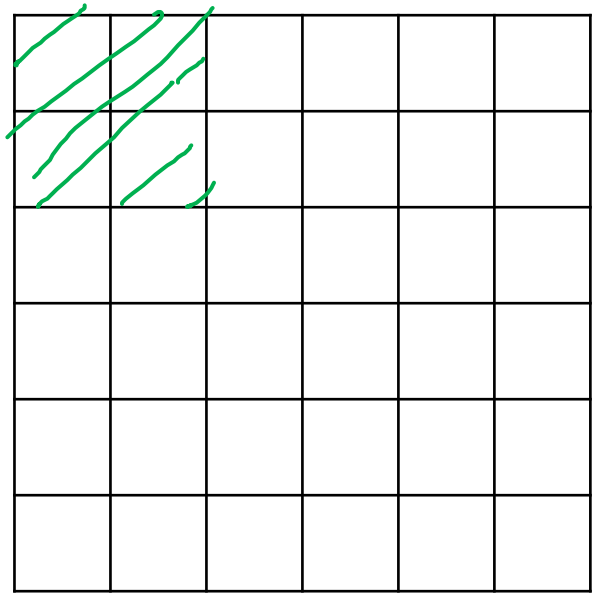
- shrinky output
- throw away into from edge



3×3
f×f

*

=



6×6

p=2

6×6 → 8×8
n×n

n - (f-1)

n-f+1 × n-f+1

6-3+1=4

4×4

p = padding = 1

n+2p-f+1 × n+2p-f+1

6+2-3+1 × ___ = 6×6

Andrew Ng

Valid and Same convolutions

↳ no padding

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

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

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

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

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

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

$5 \times 5 \quad p = 2$

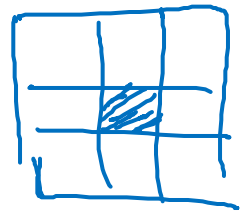
f is usually odd

1×1

3×3

5×5

7×7

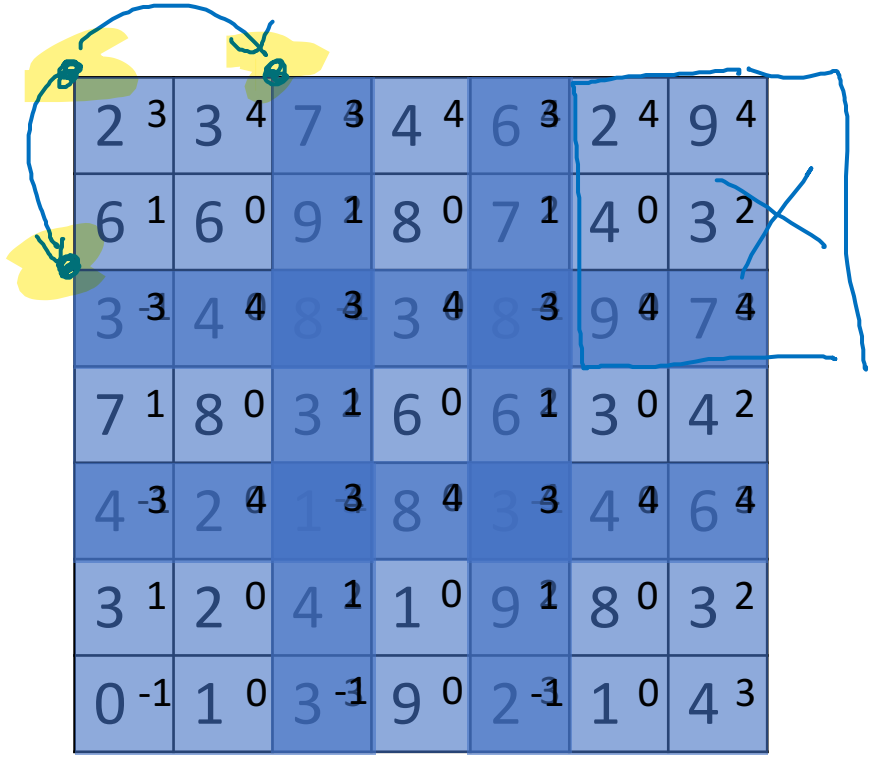


deeplearning.ai

Convolutional Neural Networks

Strided convolutions

Strided convolution



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

$s = 2$

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

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

Andrew Ng

Summary of convolutions

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

padding p $\quad$ stride s

Output size:

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

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

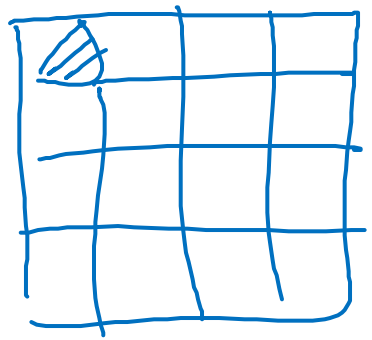
Convolution in math textbook:

2	3	7	4	6	2
6	6	9	8	7	4
3	4	8	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

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

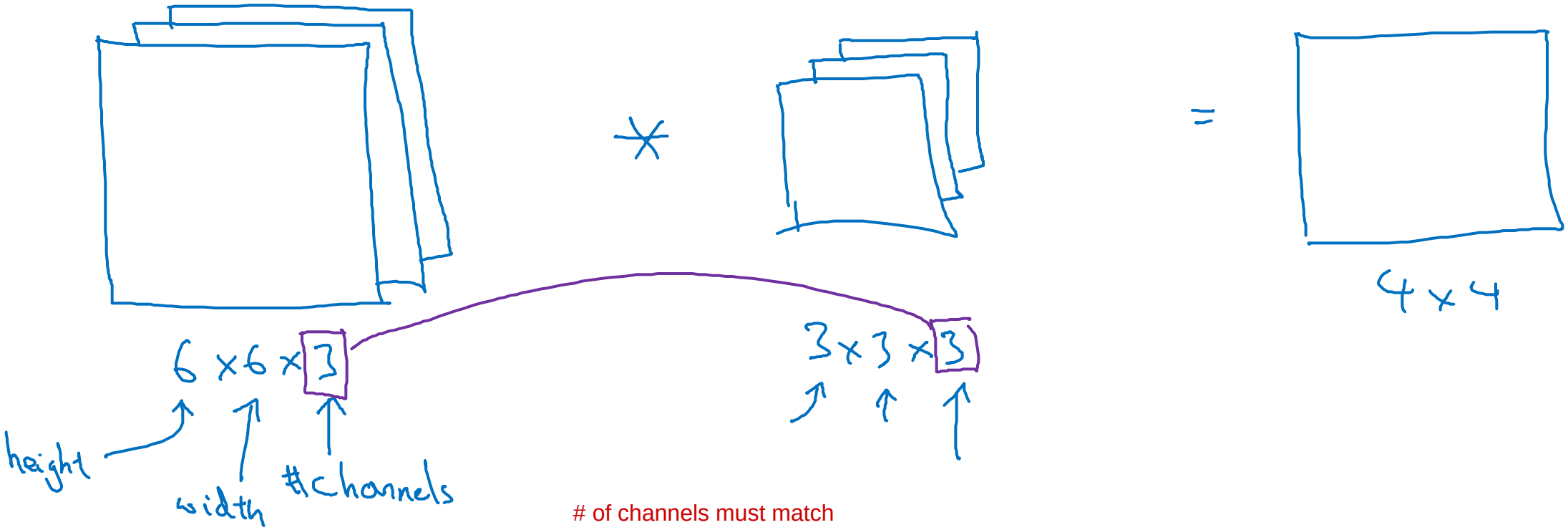
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

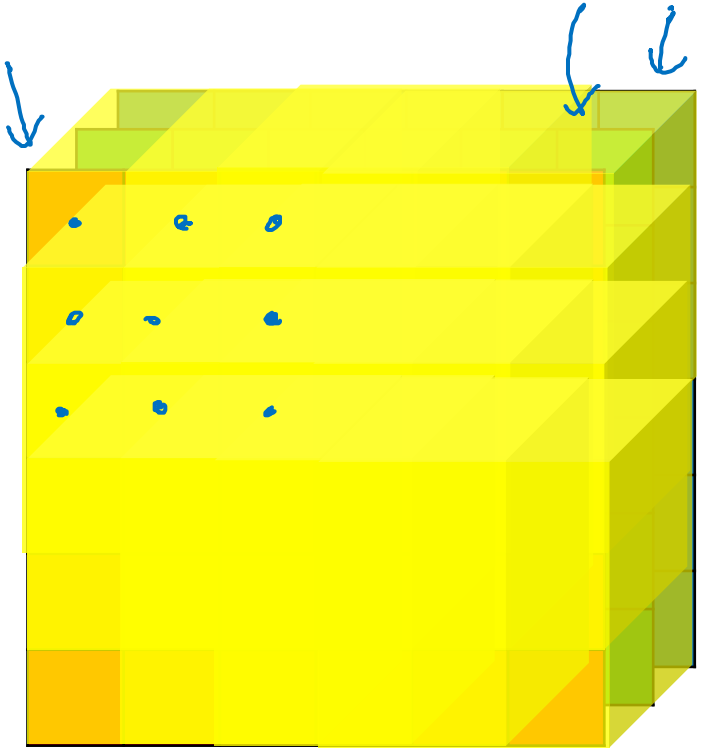
Convolutions over volumes

Convolutions on RGB images

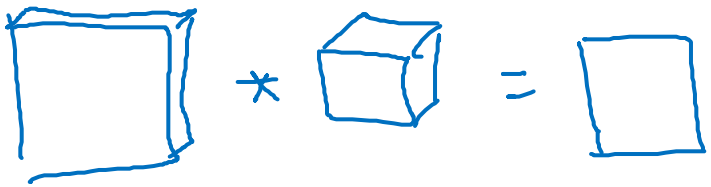


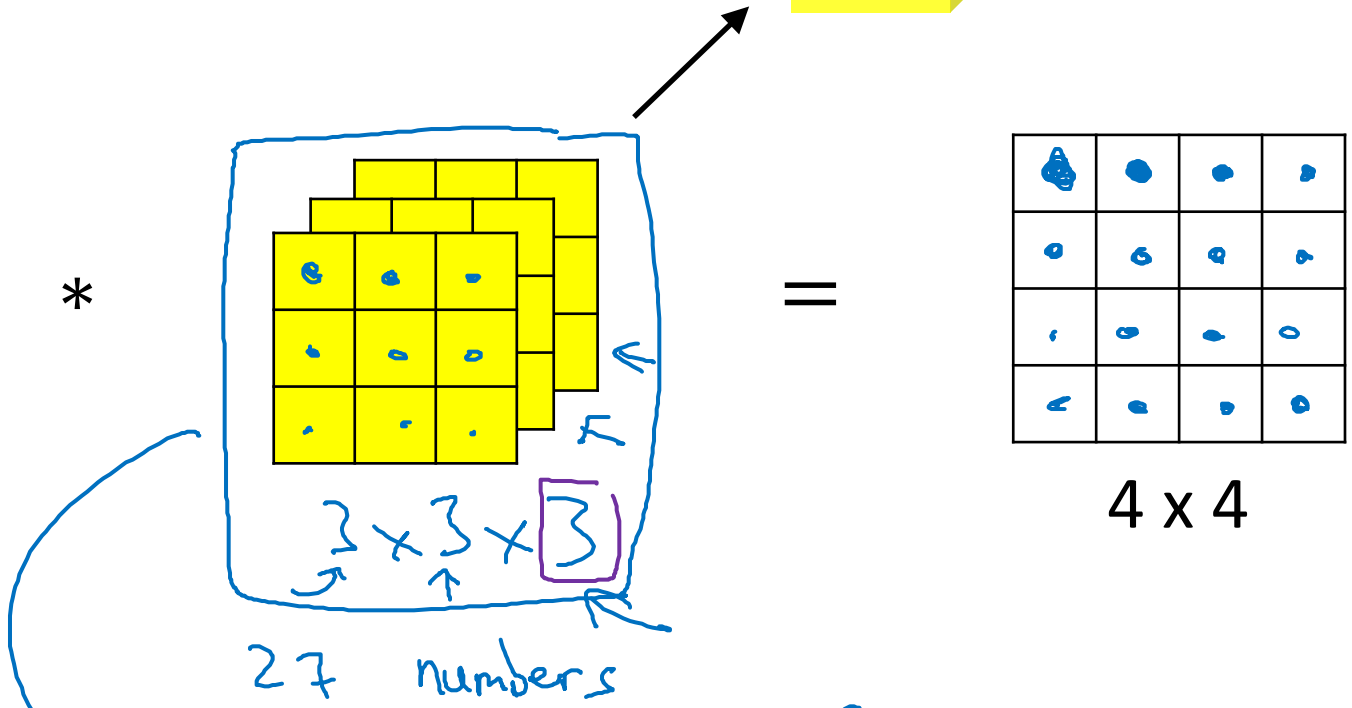
# of channels must match

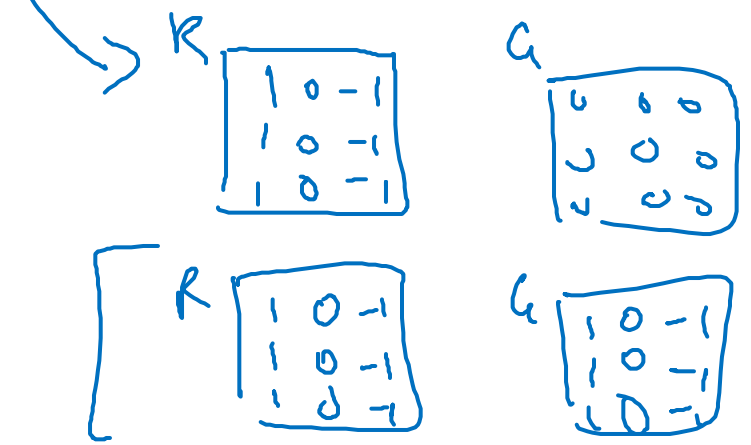
Convolutions on RGB image



6 x 6 x 3

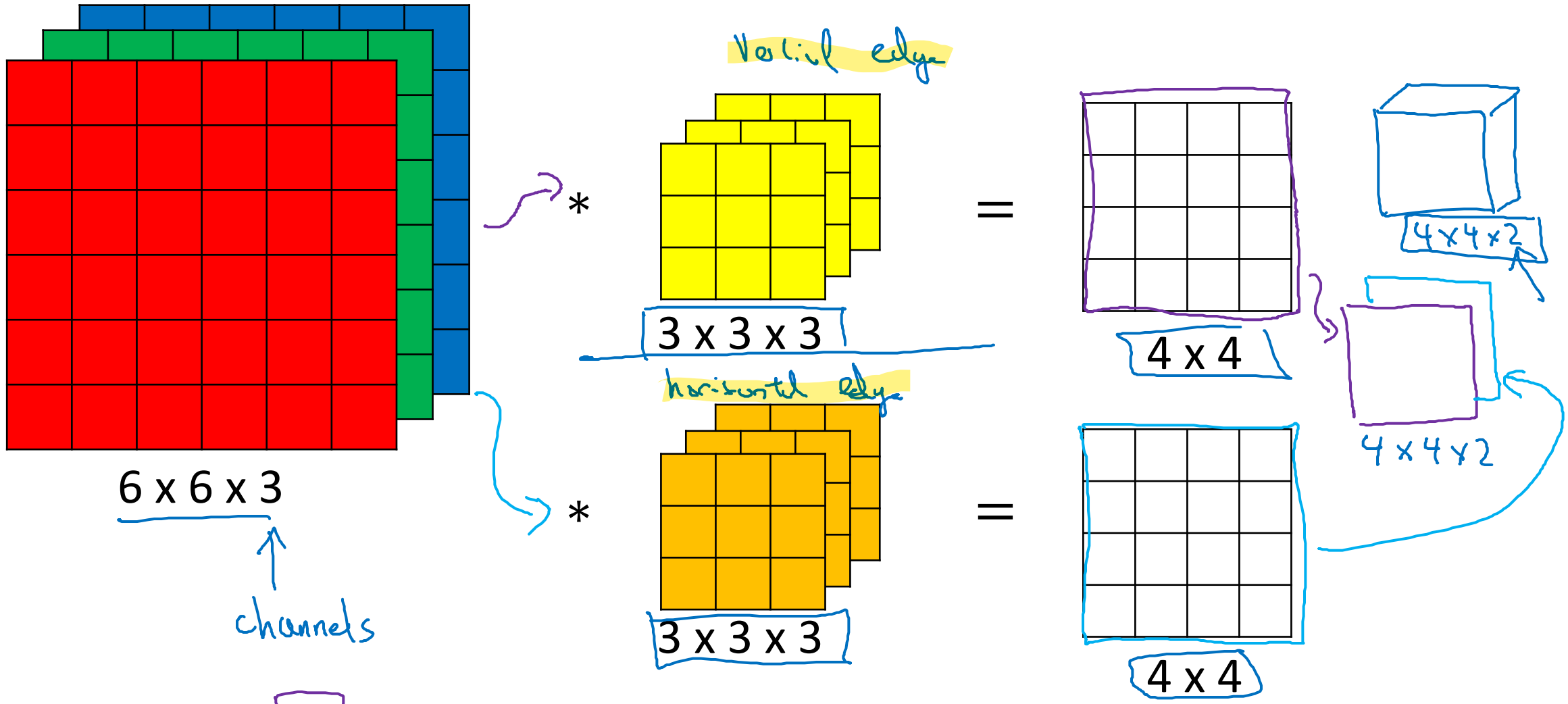






Andrew Ng

Multiple filters



Summary: $n \times n \times n_c$ * $f \times f \times n_c$ → $(n-f+1) \times (n-f+1) \times n_c'$

$6 \times 6 \times 3$ * $3 \times 3 \times 3$ → $4 \times 4 \times 2$ (n_c' = # filters)

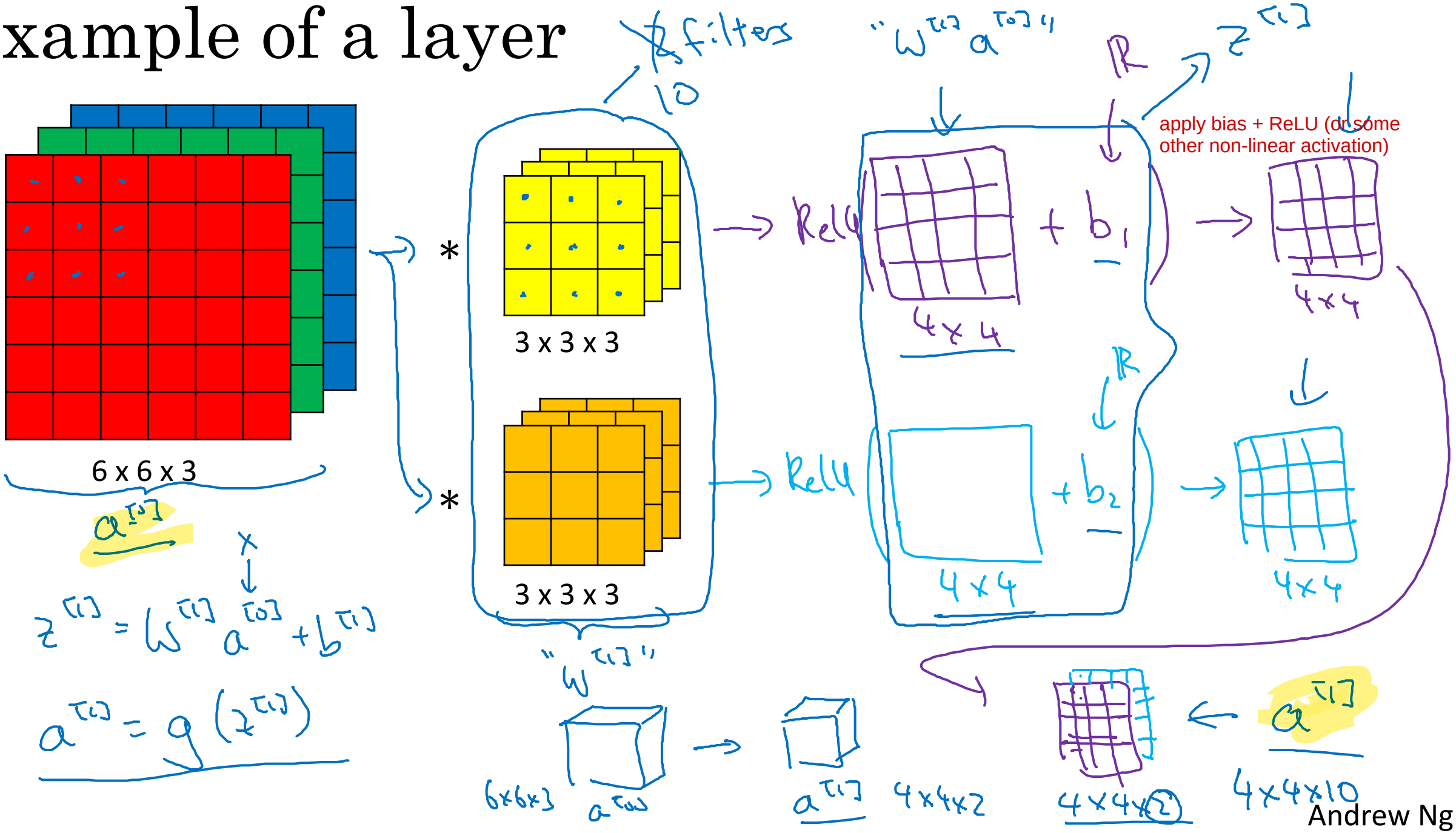
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

One layer of a convolutional network

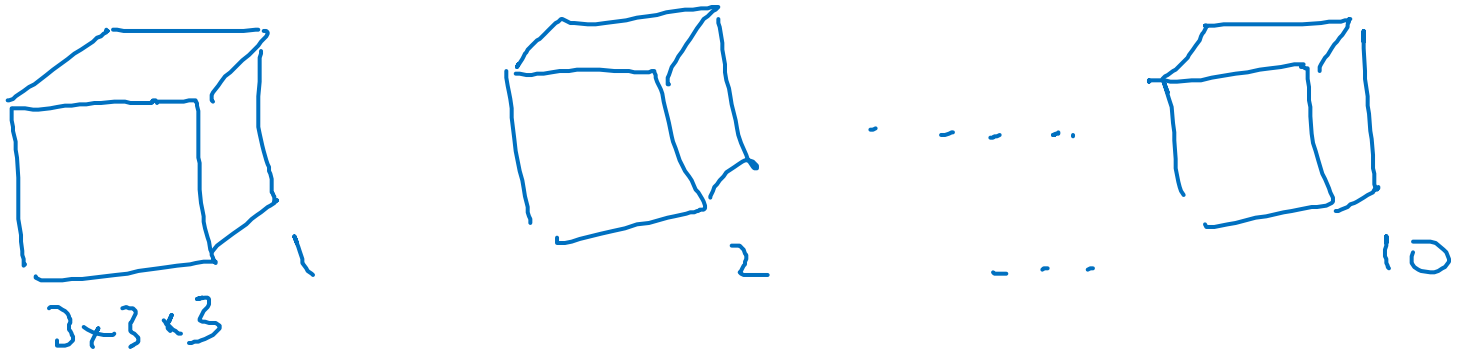
Example of a layer



Andrew Ng

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?



Andrew Ng

Summary of notation

If layer l is a convolution layer:

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

$p^{[l]}$ = padding

$s^{[l]}$ = stride

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

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

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

Weights: $(f^{[l]} \times f^{[l]} \times n_c^{[l-1]}) \times n_c^{[l]}$ ← #filters in layer l.

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

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

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

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

# of samples is the first dimension now

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

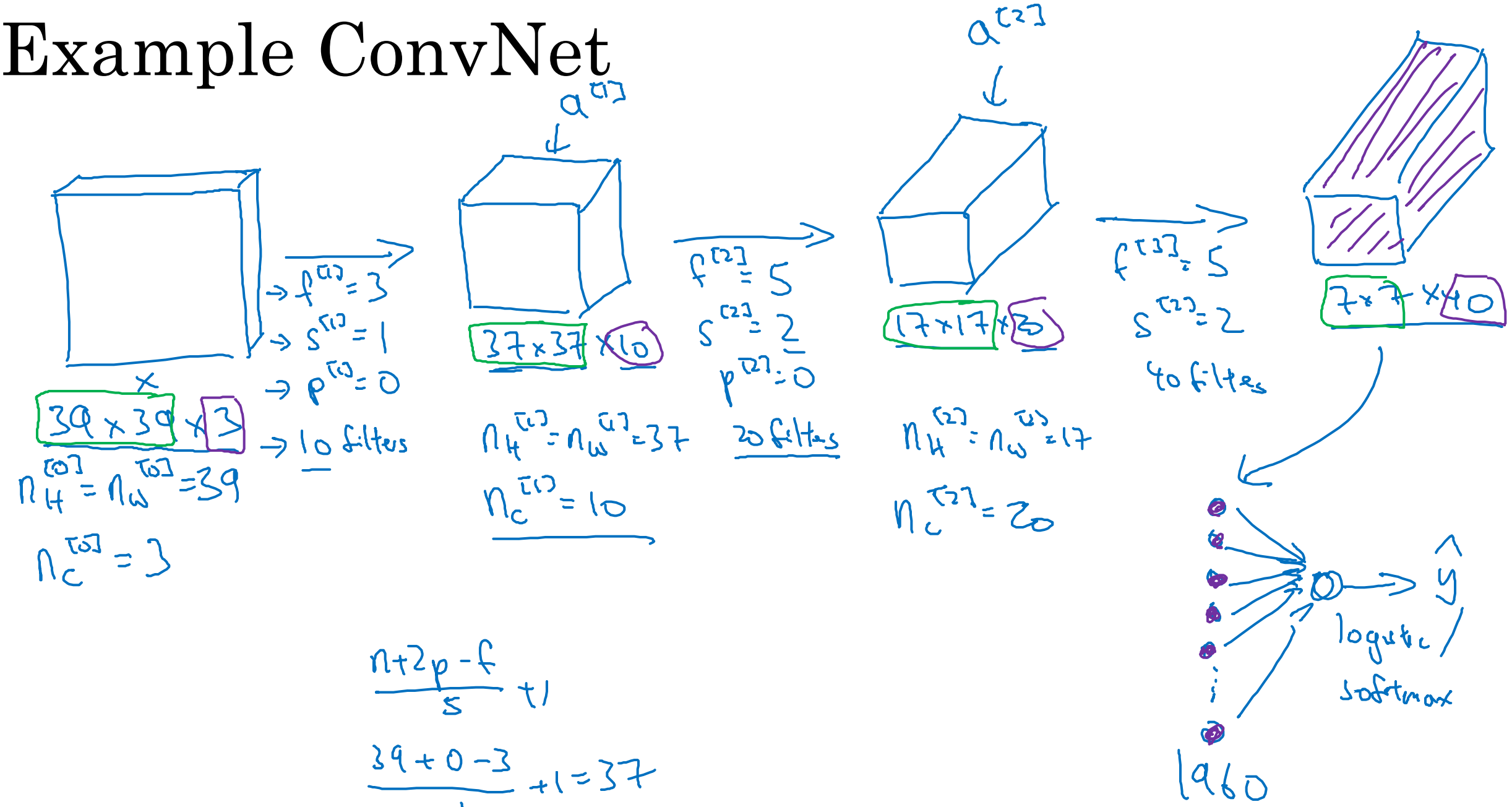
$n_c \times n_H \times n_W$

deeplearning.ai

Convolutional Neural Networks

A simple convolution network example

Example ConvNet



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

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

Andrew Ng

Types of layer in a convolutional network:

- Convolution (CONV) ←
- Pooling (POOL) ←
- Fully connected (FC) ←

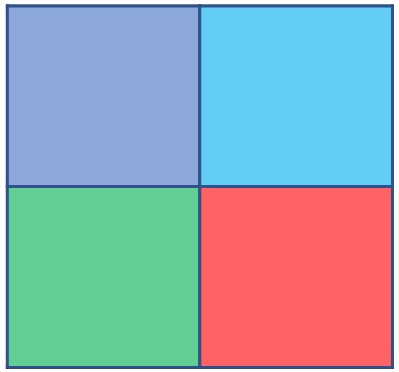
deeplearning.ai

Convolutional Neural Networks

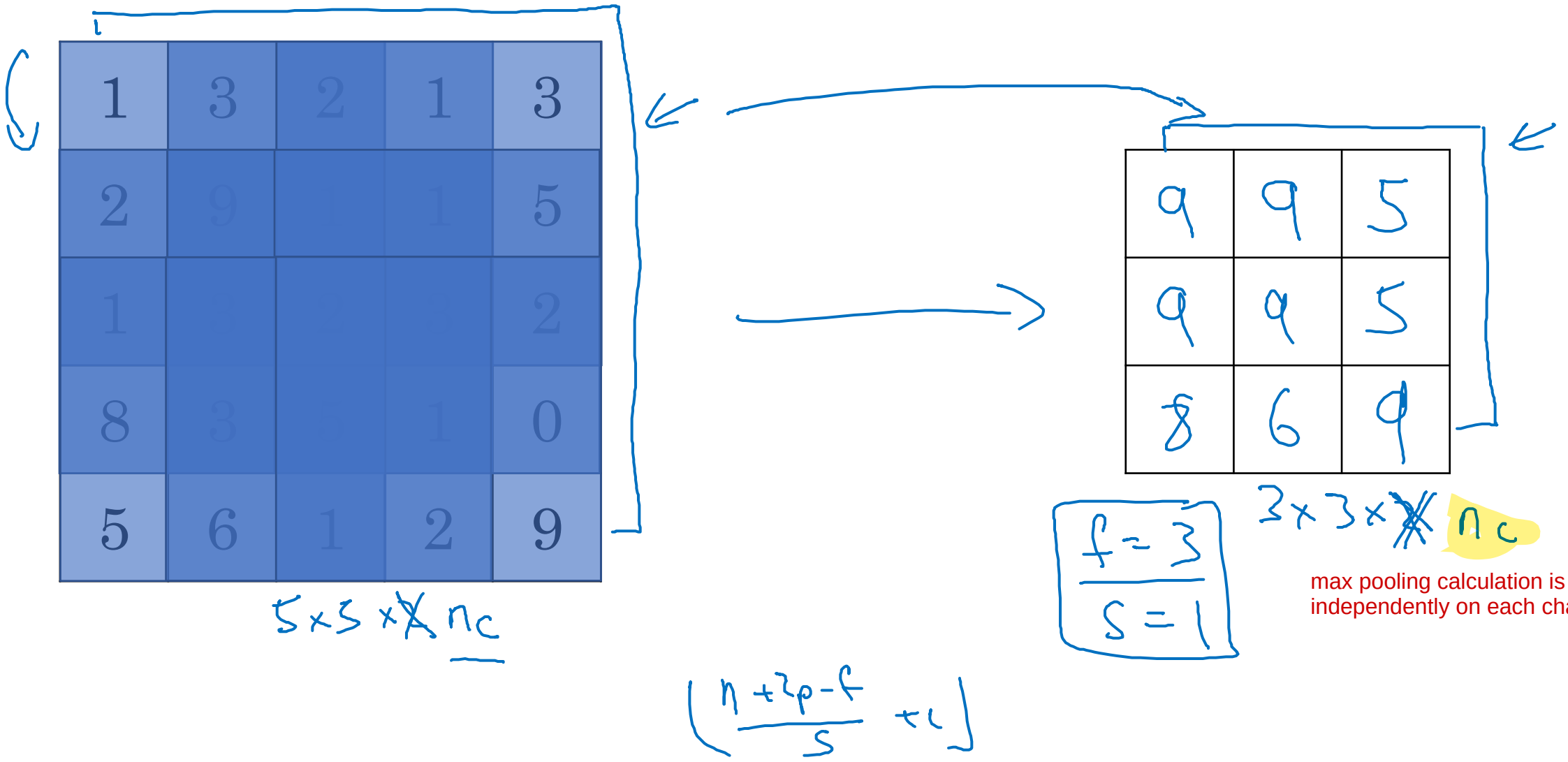
Pooling layers

Pooling layer: Max pooling

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

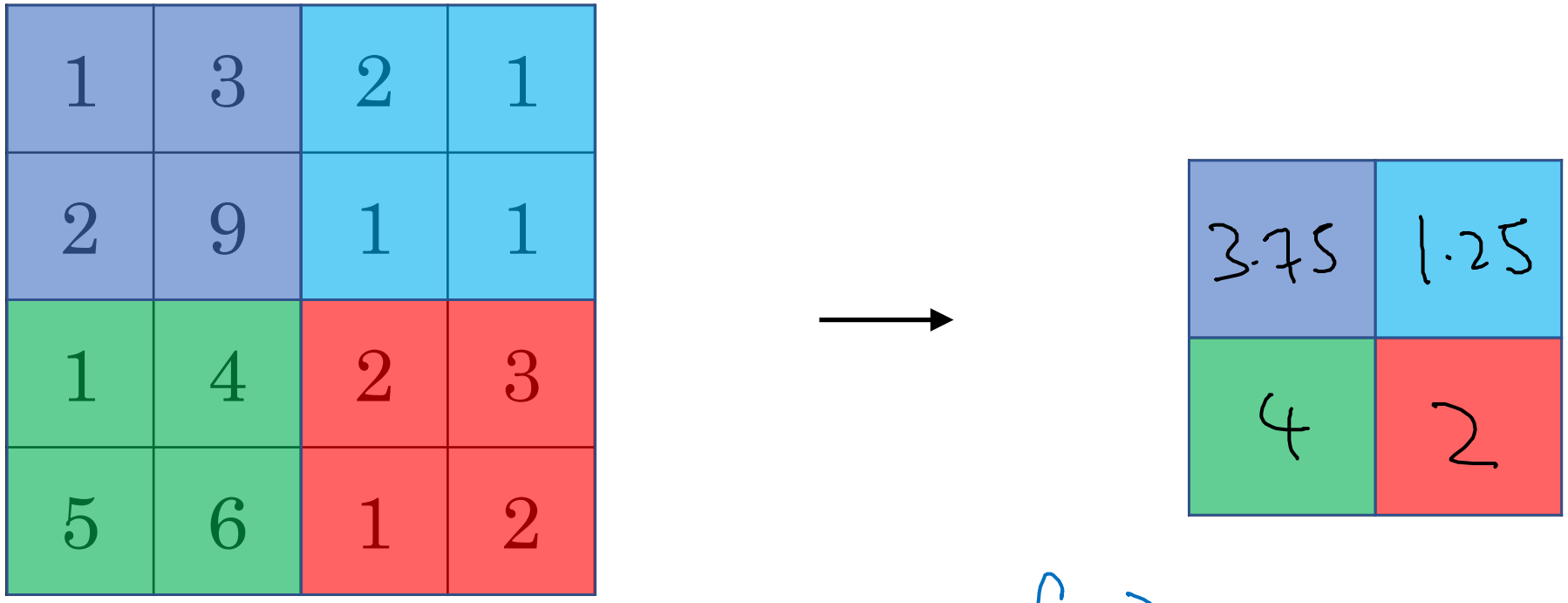


Pooling layer: Max pooling



Andrew Ng

Pooling layer: Average pooling



Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or average pooling

$f=2, s=2$

$f=3, s=2$

$n_H \times n_W \times n_C$

$\downarrow$

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

$\times n_C$

~~p : padding~~

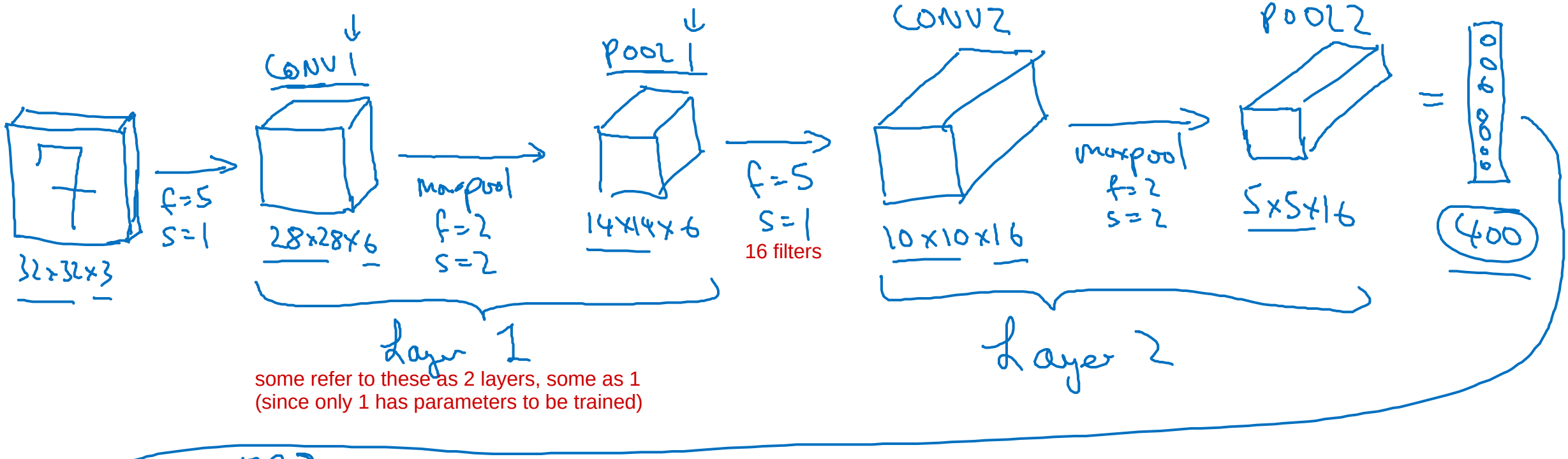
No parameters to learn!

deeplearning.ai

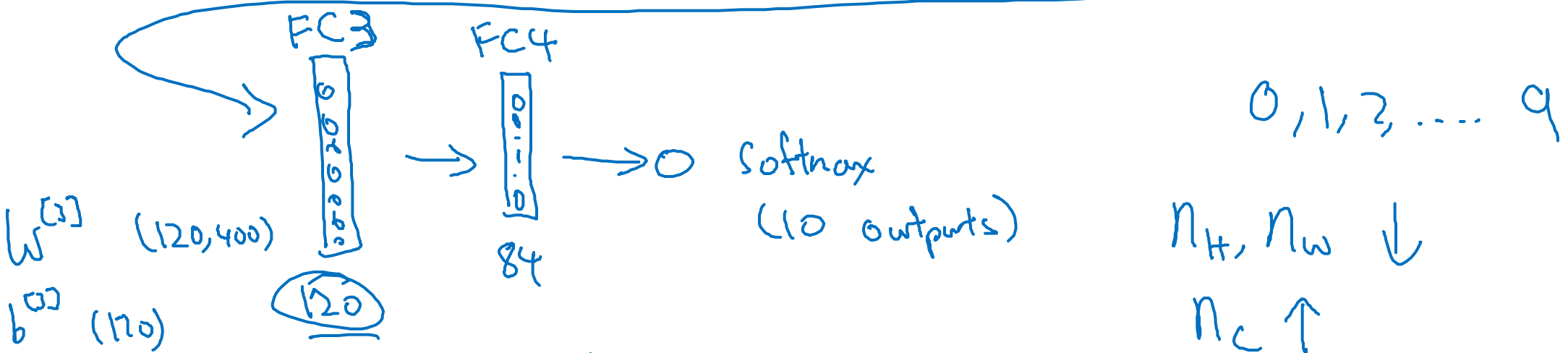
Convolutional Neural Networks

Convolutional neural network example

Neural network example (LeNet-5)



some refer to these as 2 layers, some as 1
(since only 1 has parameters to be trained)



Andrew Ng

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

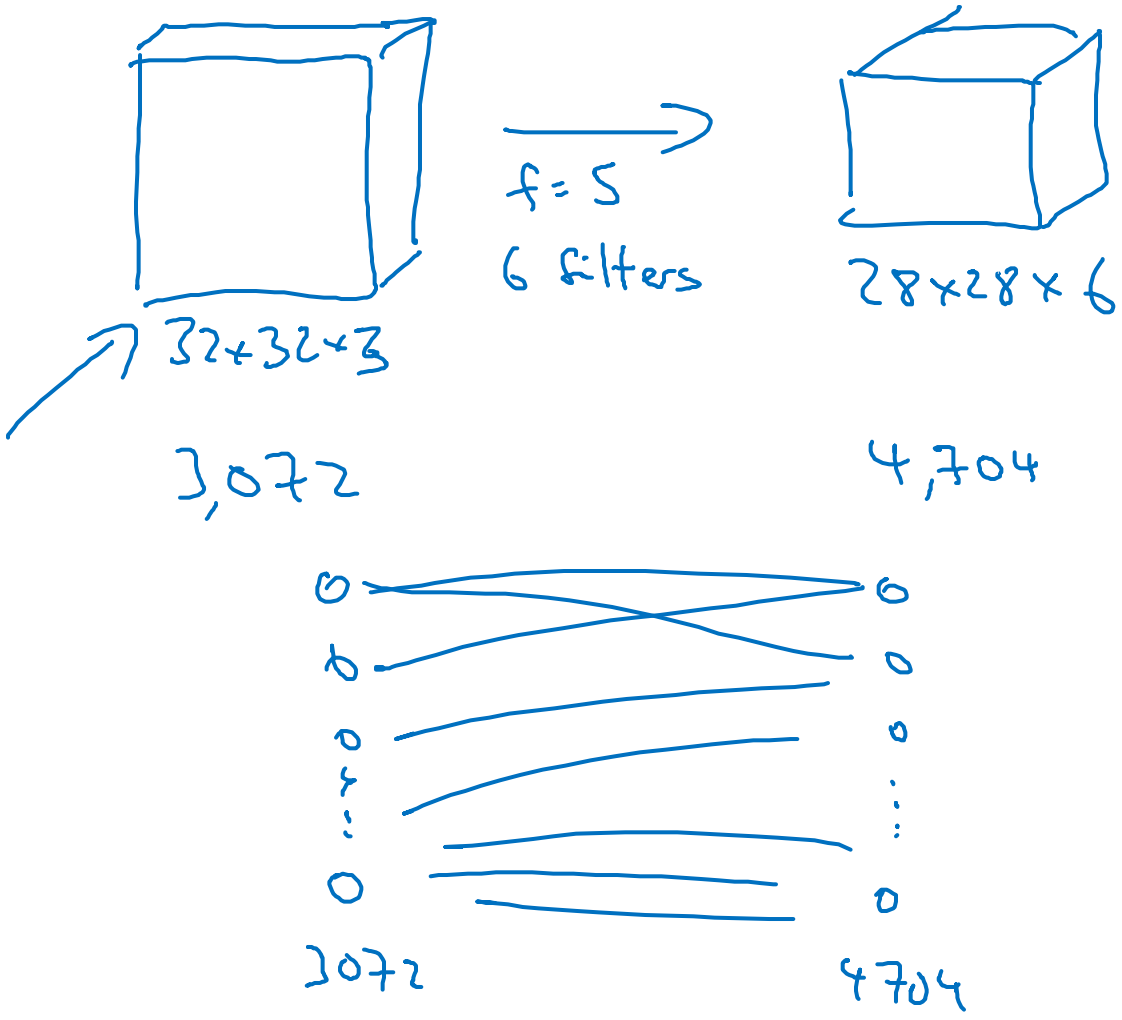
Andrew Ng

deeplearning.ai

Convolutional Neural Networks

Why convolutions?

Why convolutions



5×5 — 25

26

~~6 × 26~~ = ~~156~~ parameters

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

+ Parameter sharing
+ Sparcity of connections

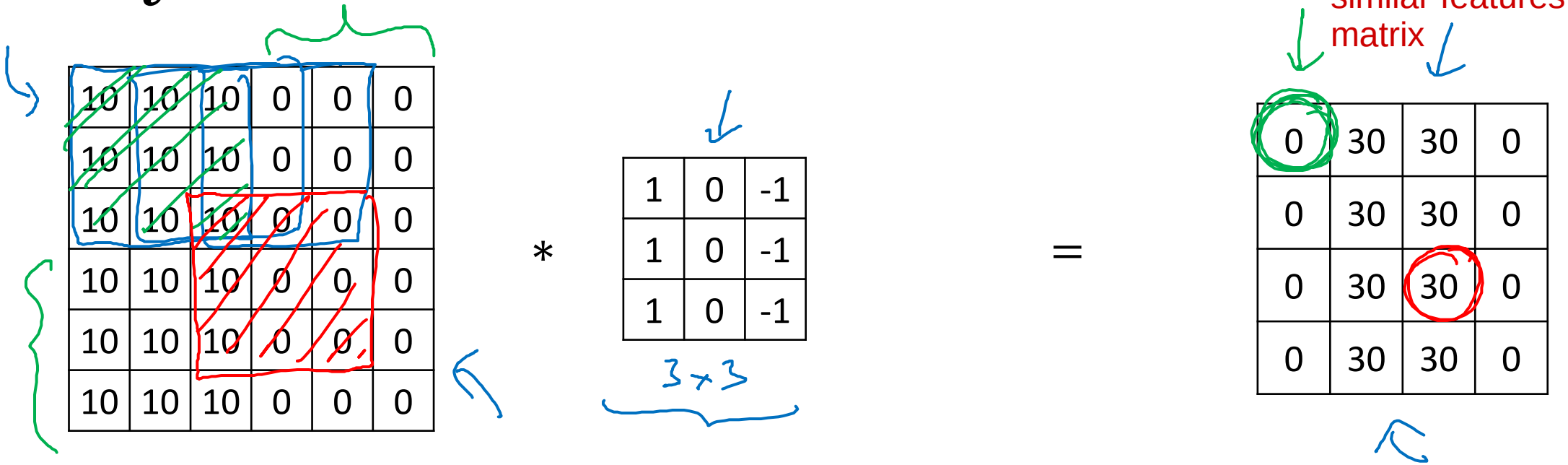
3,072 × 4,704 ≈ 14M

Andrew Ng

Why convolutions

translation invariance

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



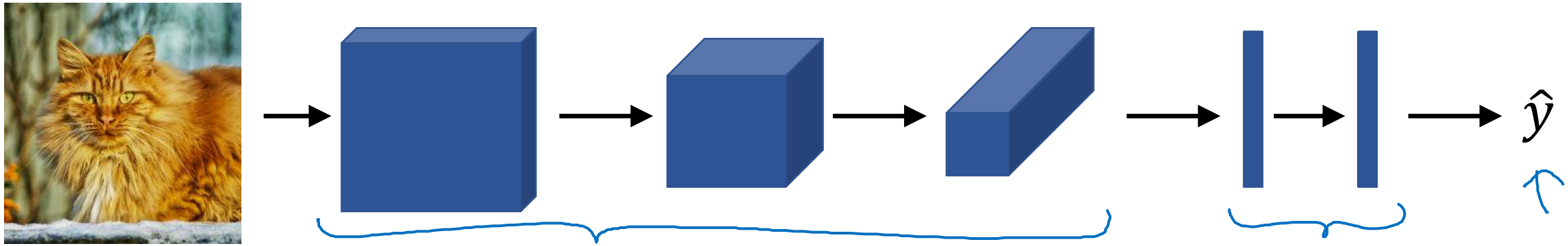
Parameter sharing: A feature detector (such as a vertical edge detector) that's useful in one part of the image is probably useful in another part of the image.

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)

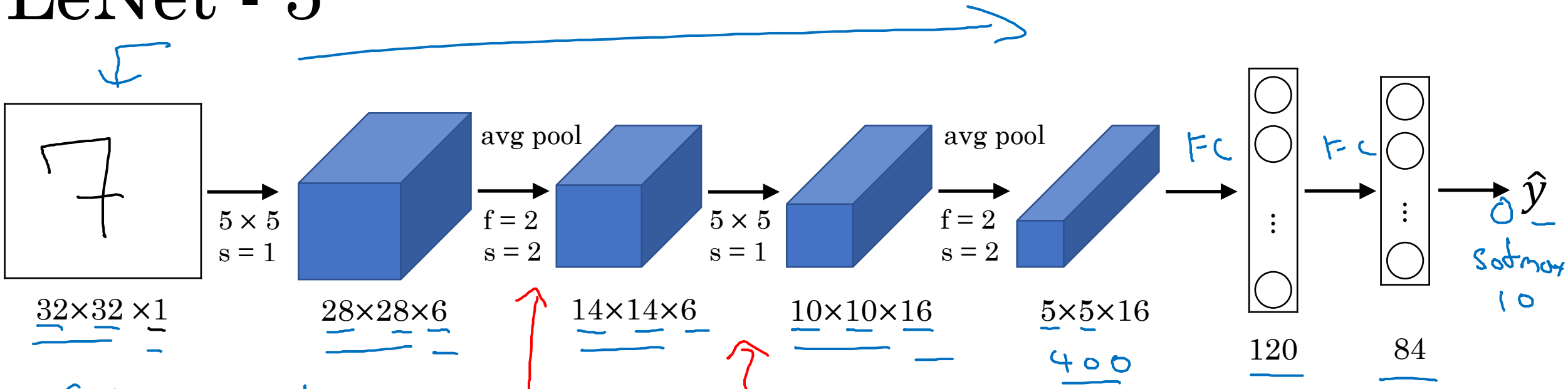
Inception

deeplearning.ai

Case Studies

Classic networks

LeNet - 5



60k parameters.

$n_H, n_w \downarrow$ $n_c \uparrow$

Non-linearity after pooling

$n_{H} \times n_{w} \times n_c$ $f \times f \times n_c$

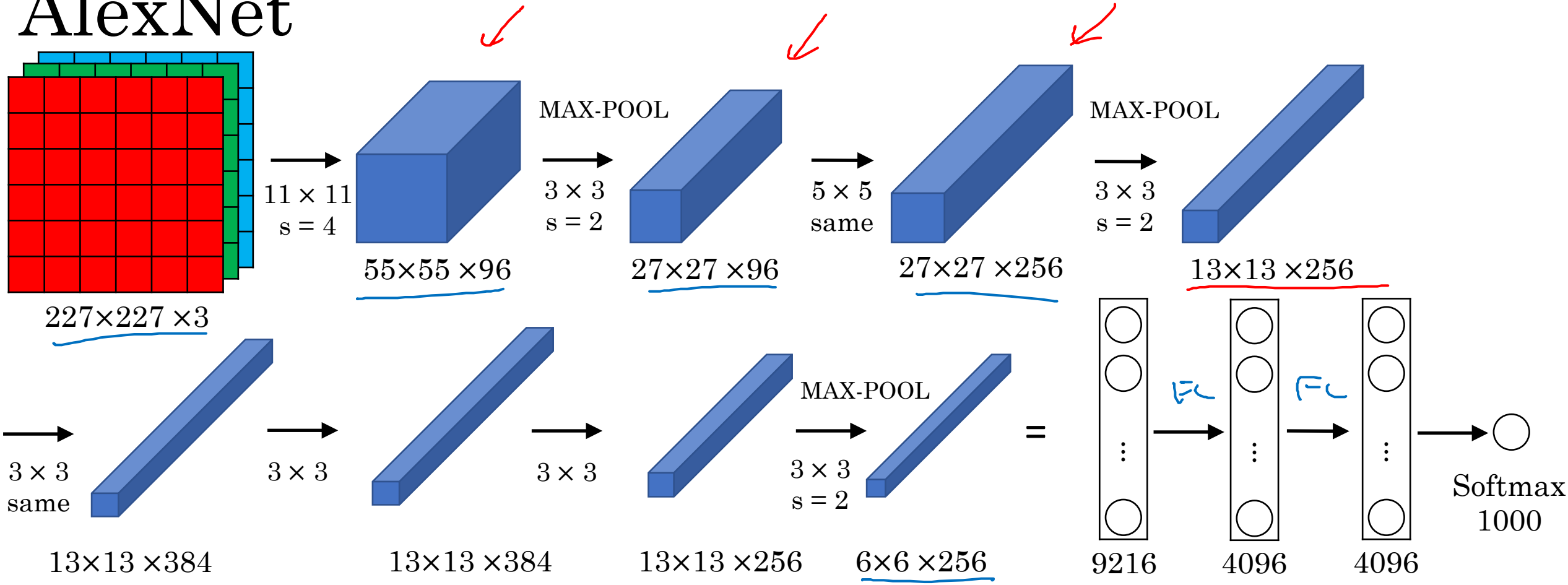
conv pool conv pool fc fc output

Activation: sigmoid/tanh ReLU

II III

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

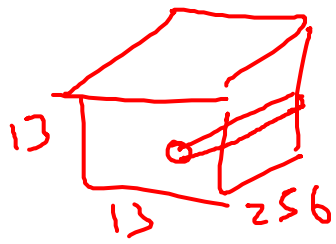
AlexNet



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

9216

60M parameters



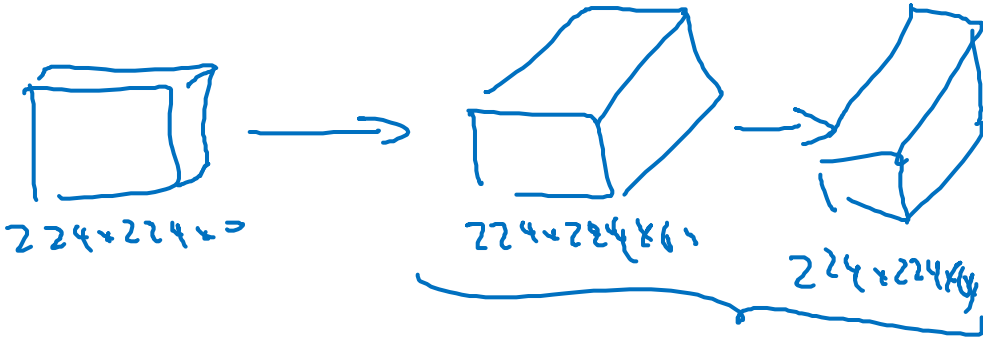
[Krizhevsky et al., 2012. ImageNet classification with deep convolutional neural networks]

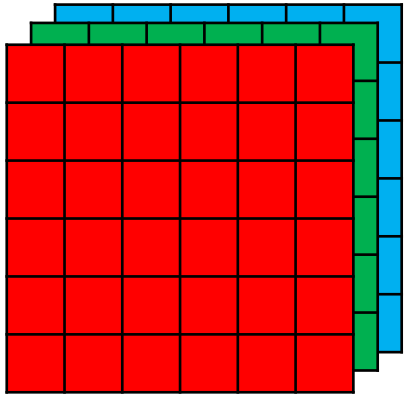
VGG - 16

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]

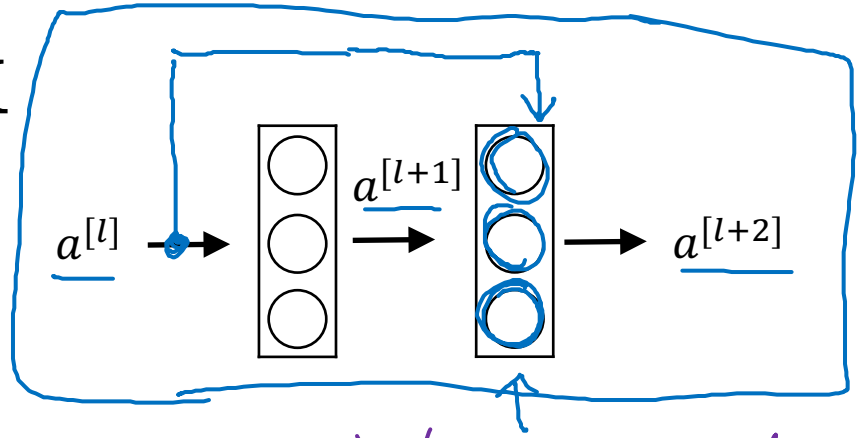
Andrew Ng

deeplearning.ai

Case Studies

Residual Networks (ResNets)

Residual block



"Short cut" / skip connection

$a^{[l]}$ → Linear → ReLU → $a^{[l+1]}$ → Linear → (+) → ReLU → $a^{[l+2]}$

"main path"

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

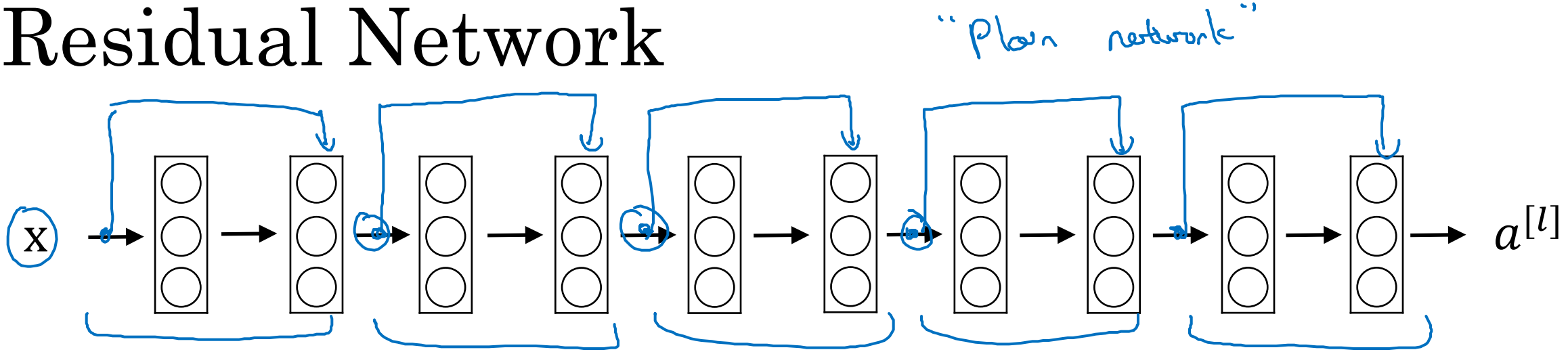
$$a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$$

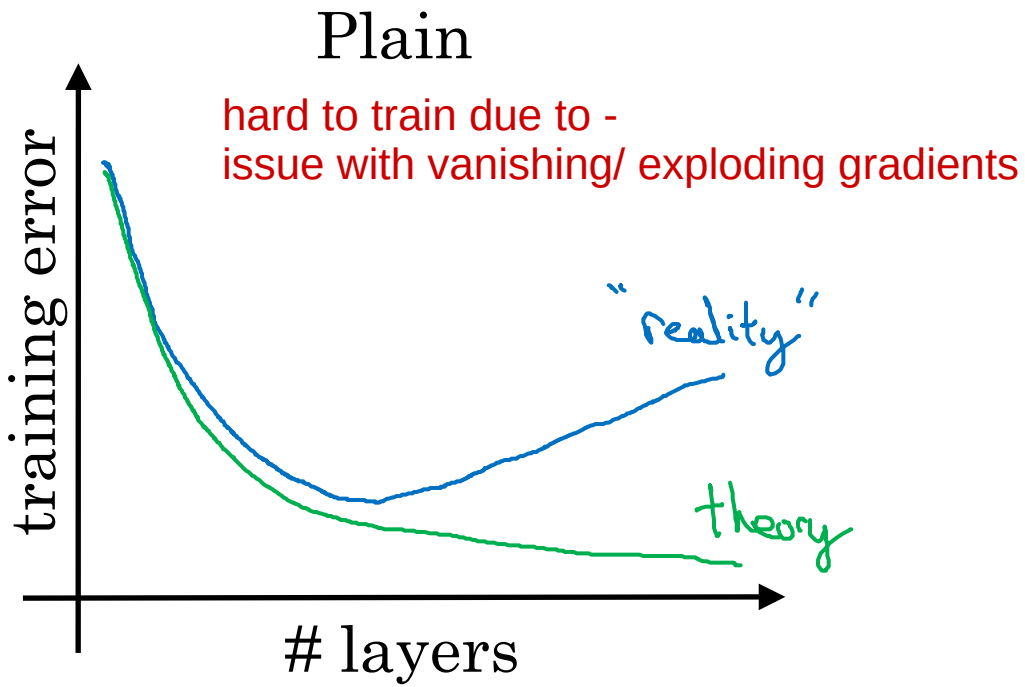
'a' added to 'z'
dimensions are kept same

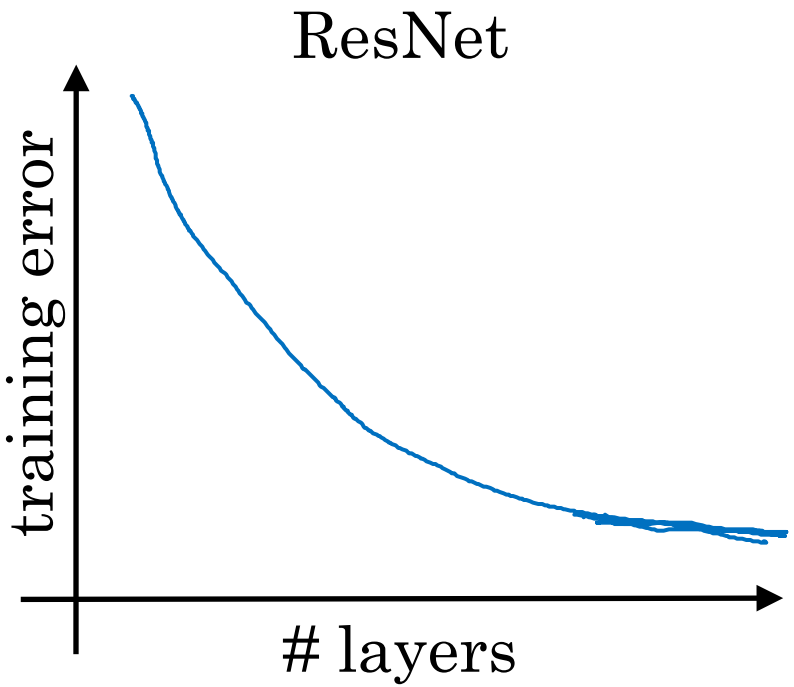
[He et al., 2015. Deep residual networks for image recognition]

Andrew Ng

Residual Network







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

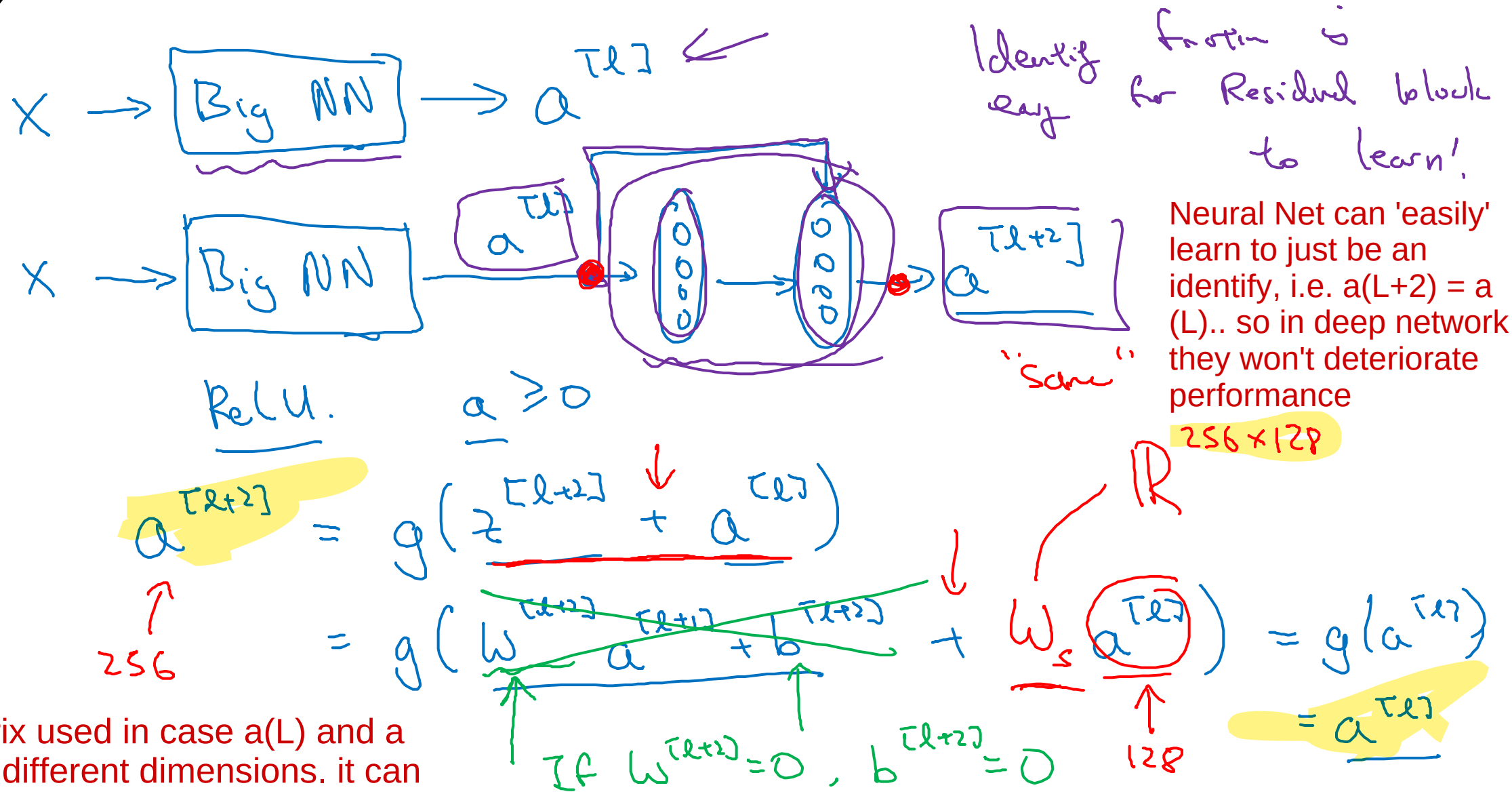
Andrew Ng

deeplearning.ai

Case Studies

Why ResNets work

Why do residual networks work?

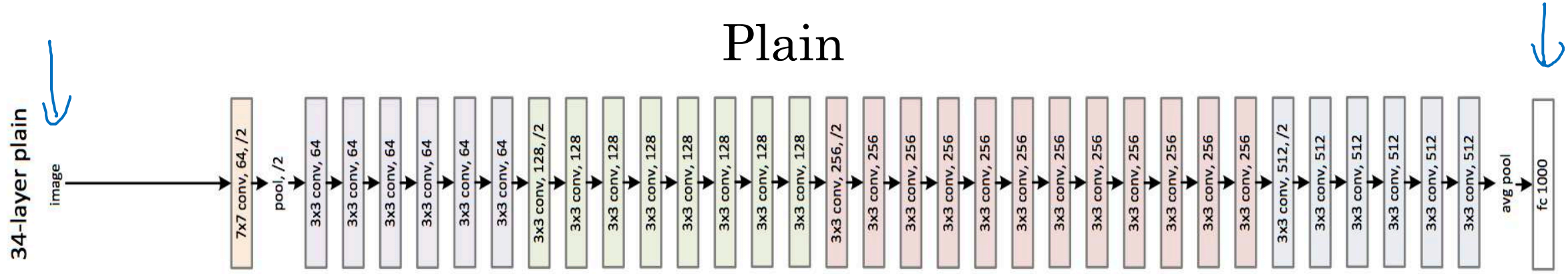


Ws - matrix used in case a(L) and a (L+2) are different dimensions. it can be parameter to be learnt, or padding matrix

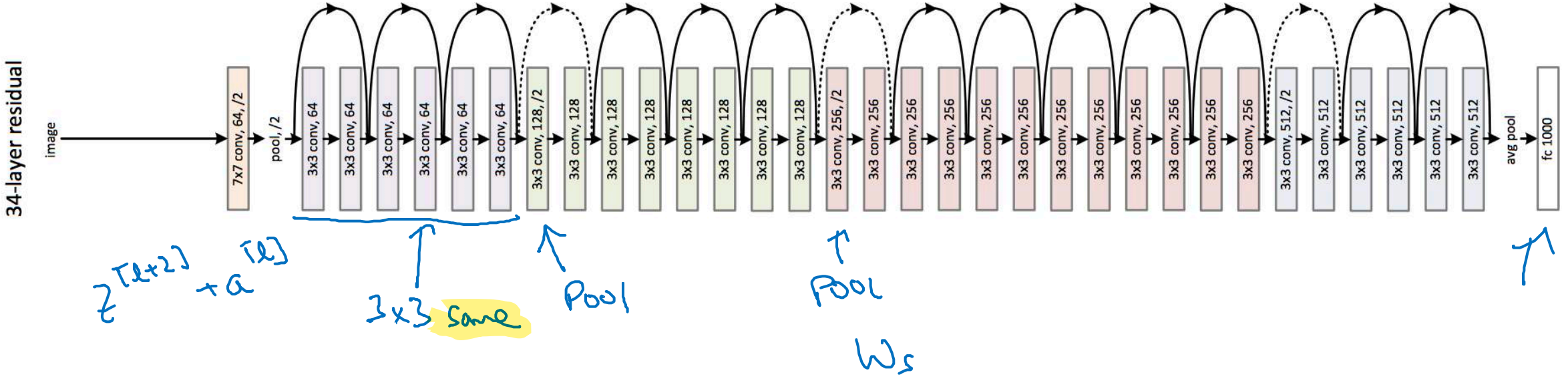
Andrew Ng

ResNet

Plain



ResNet



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

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

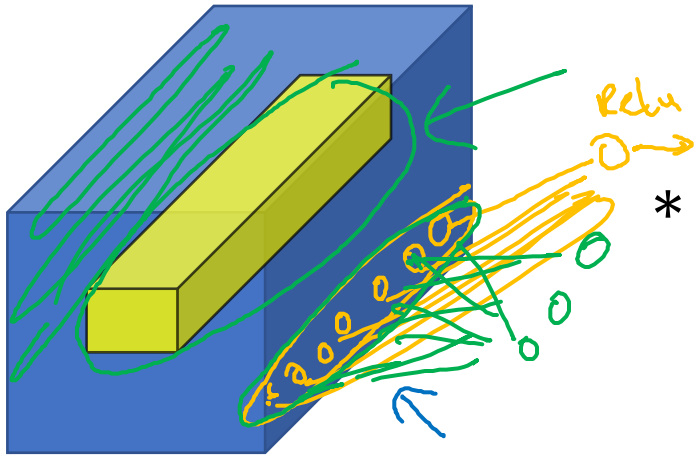
6 × 6 × 1

*

2

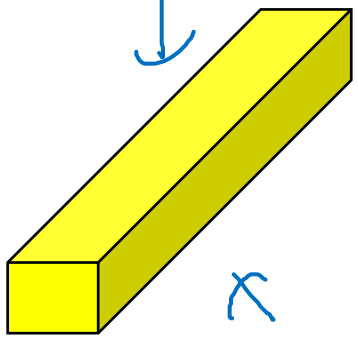
=

2	4	6	...		



6 × 6 × 32

*

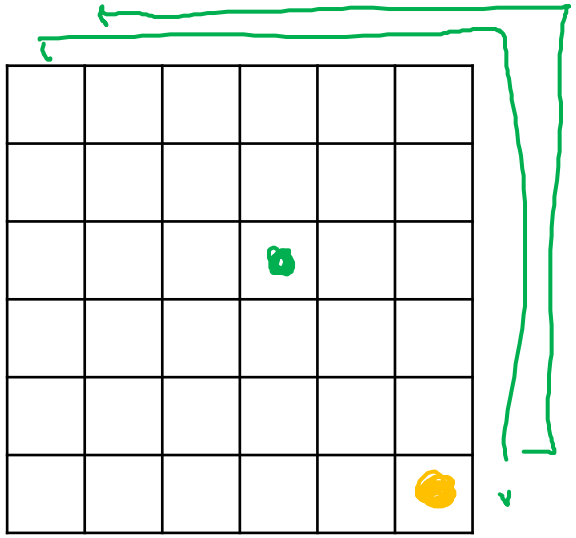


1 × 1 × 32

=

32 → # filters.

$n_c^{[l+1]}$



6 × 6 × # filters

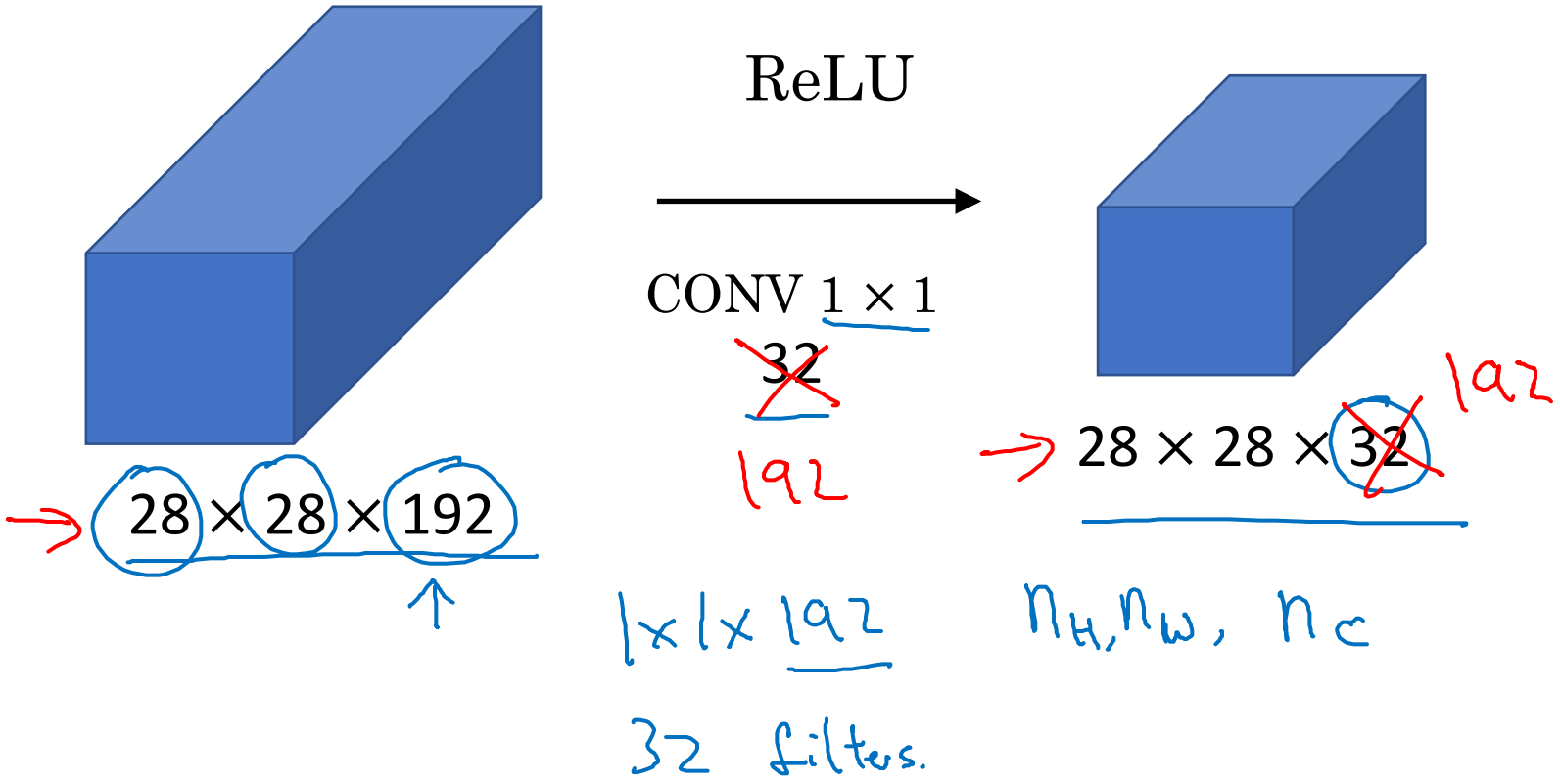
ReLU

Network in Network

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

Andrew Ng

Using 1×1 convolutions



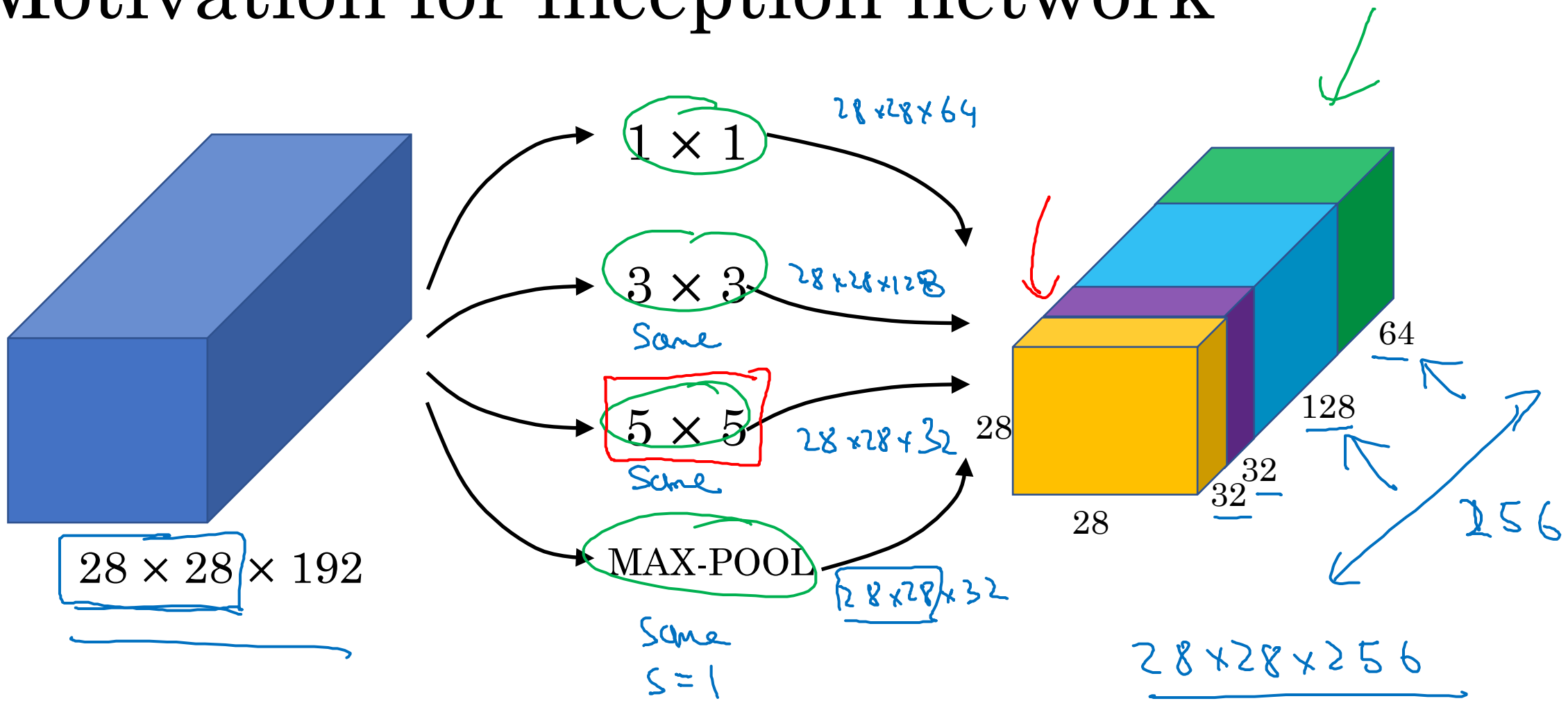
[Lin et al., 2013. Network in network]

deeplearning.ai

Case Studies

Inception network motivation

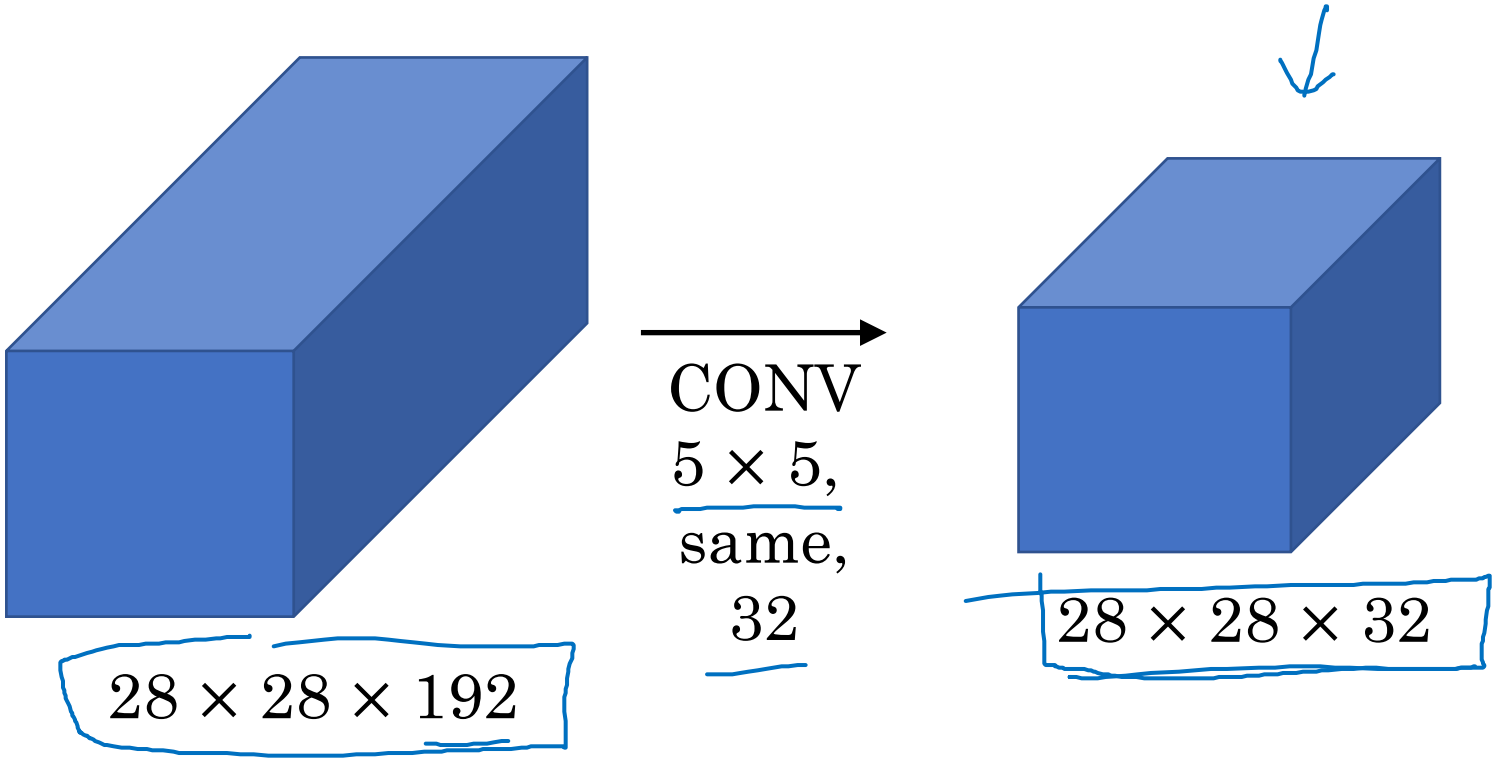
Motivation for inception network



Combine multiple approaches in a single step
Pooling will require padding

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

The problem of computational cost



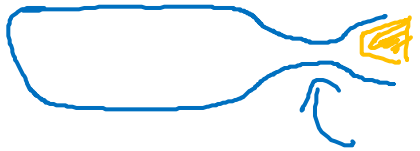
32 filters. filter one 5×5×192.

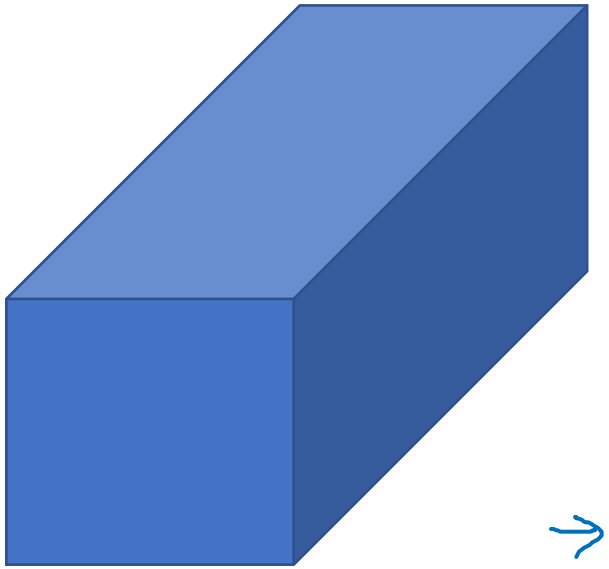
28×28×32 × 5×5×192 = 120M.

# of multiplications

Using 1×1 convolution

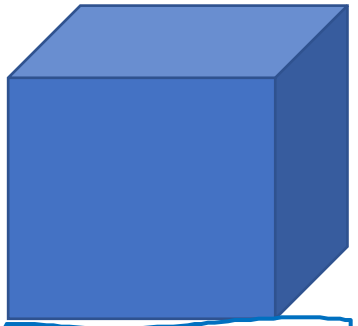
"bottleneck layer"





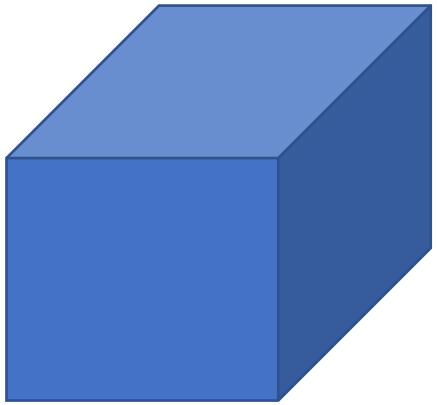
28 × 28 × 192

CONV
1 × 1,
16,
1 × 1 × 192



28 × 28 × 16

CONV
5 × 5,
32,
5 × 5 × 16



28 × 28 × 32

28×28×16 × 192 = 2.4M

28×28×32 × 5×5×16 = 10.0M

12.4M

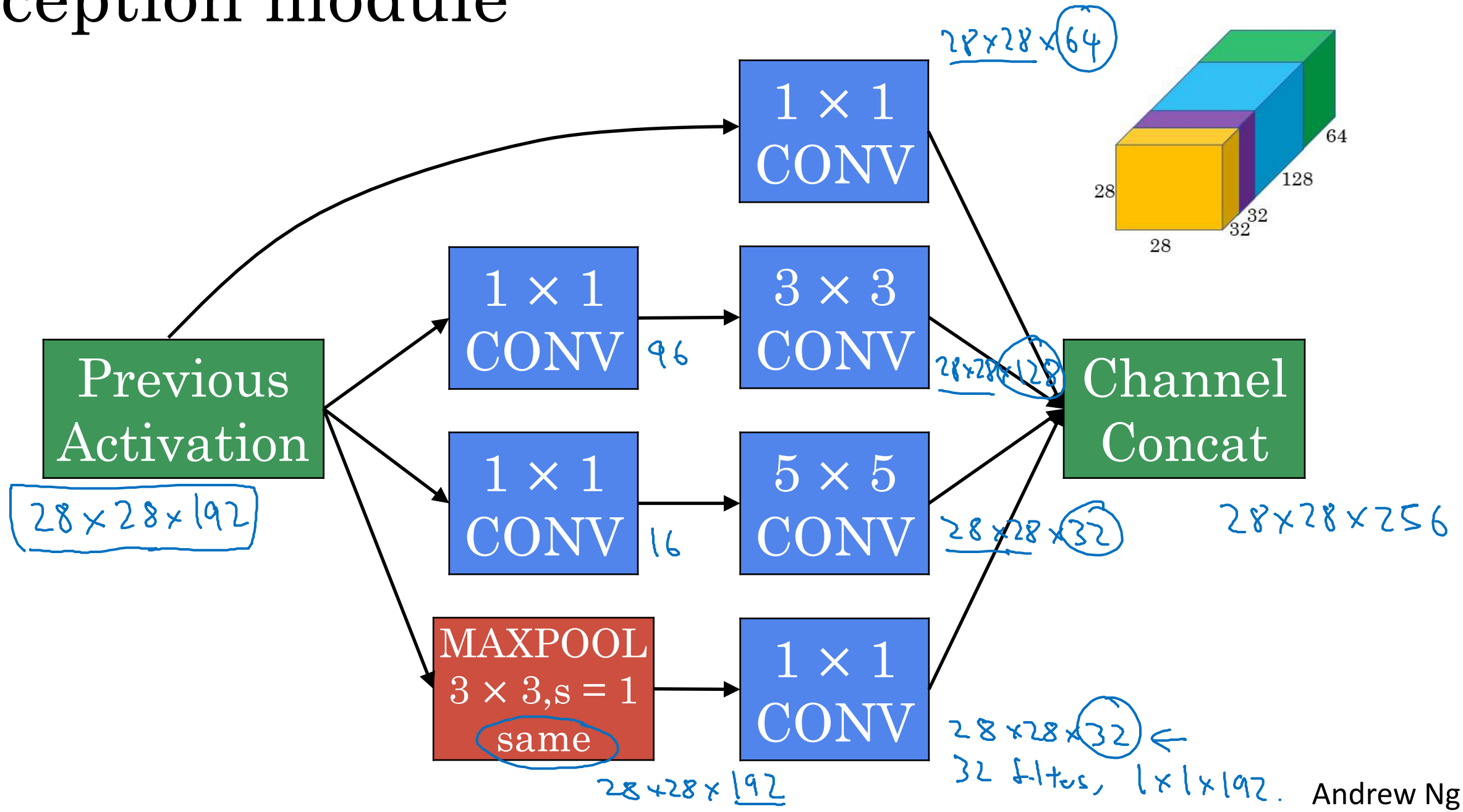
120M

deeplearning.ai

Case Studies

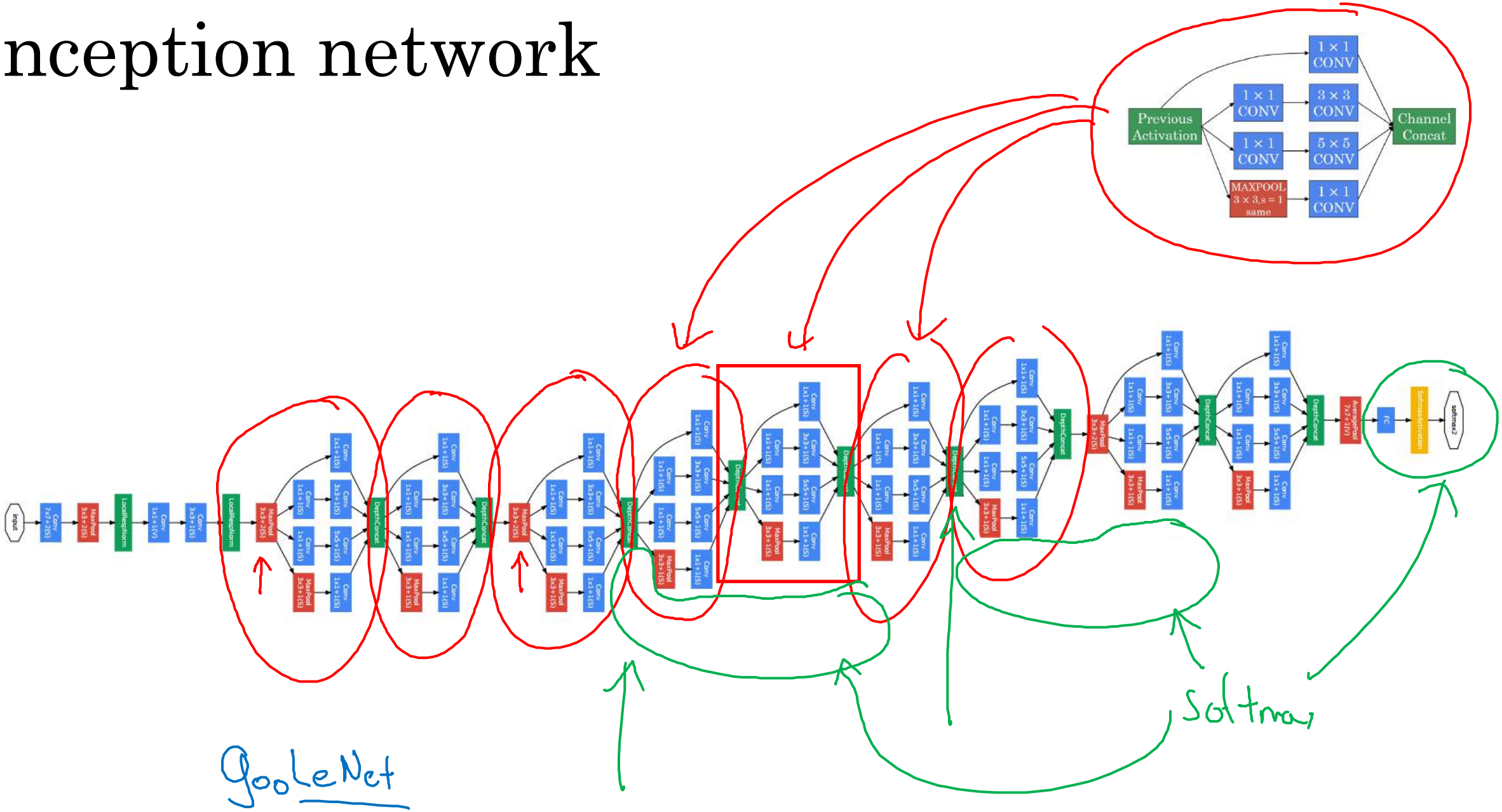
Inception network

Inception module



Andrew Ng

Inception network



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

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

Convolutional Neural Networks

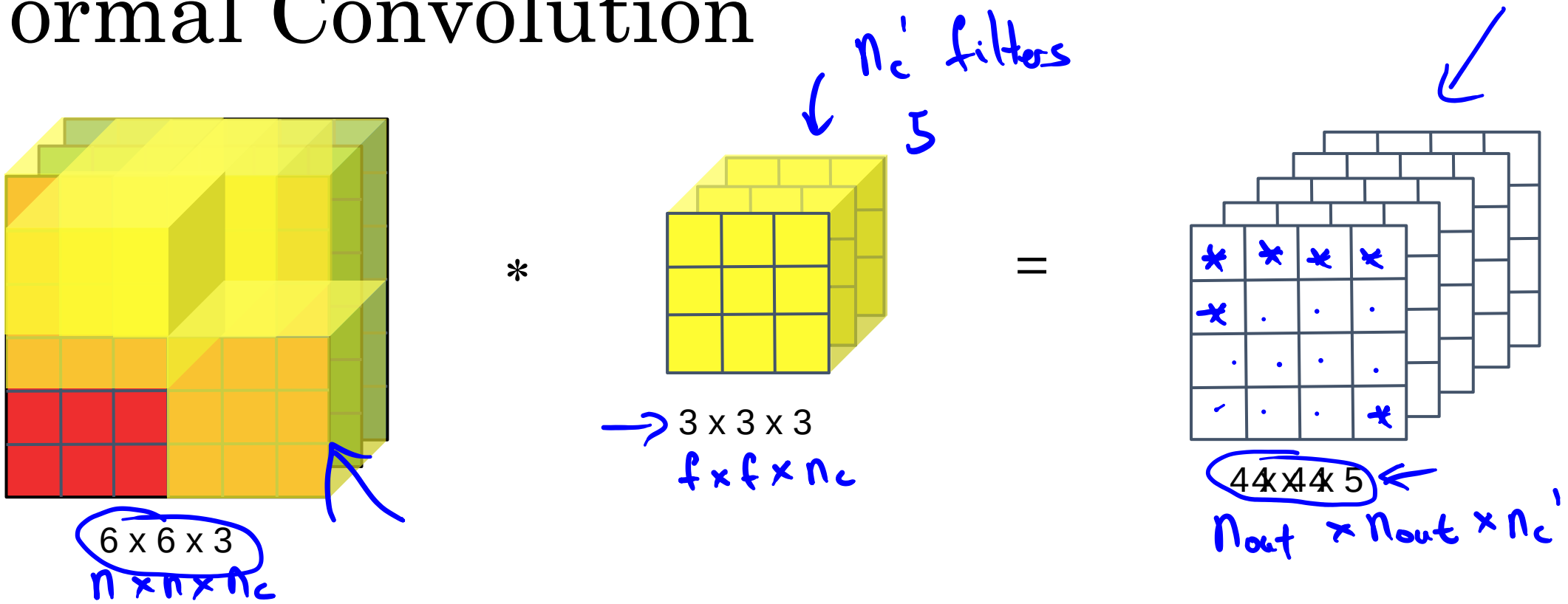
MobileNet

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]

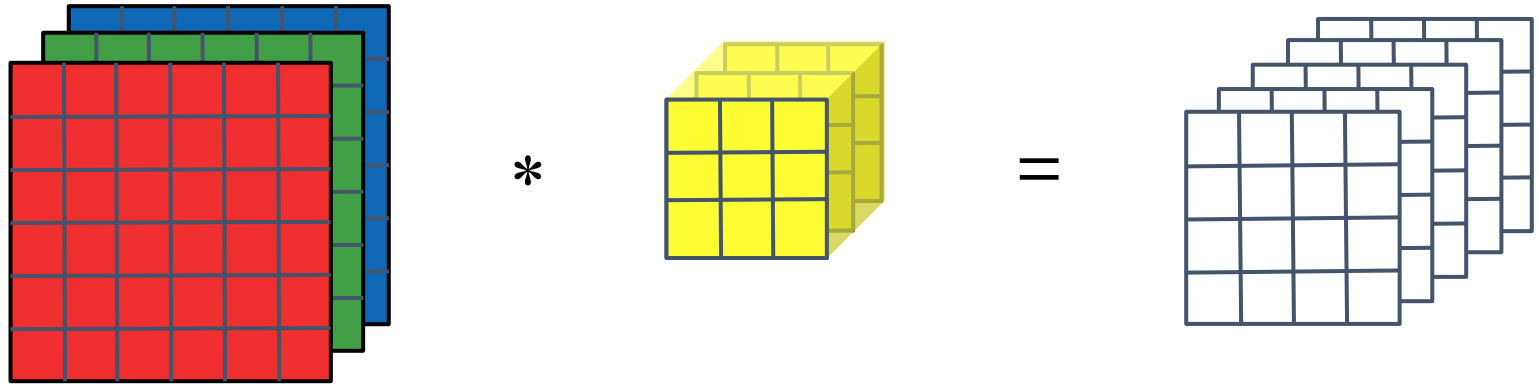
Normal Convolution



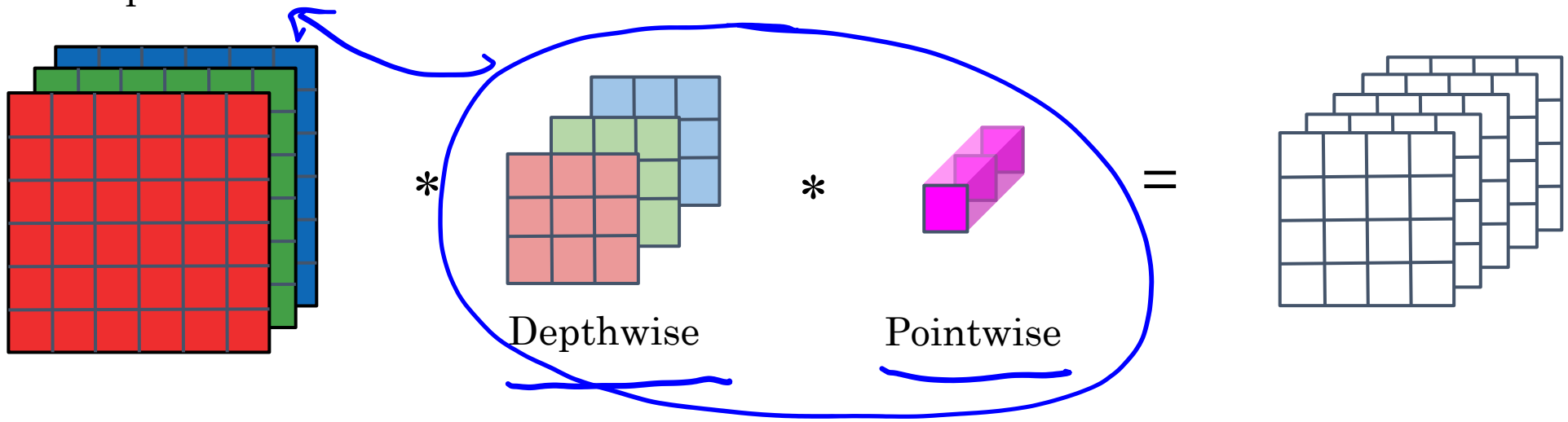
Computational cost	=	#filter params	x	# filter positions	x	# of filters
2160	=	3x3x3	x	4x4	x	5

Depthwise Separable Convolution

Normal Convolution

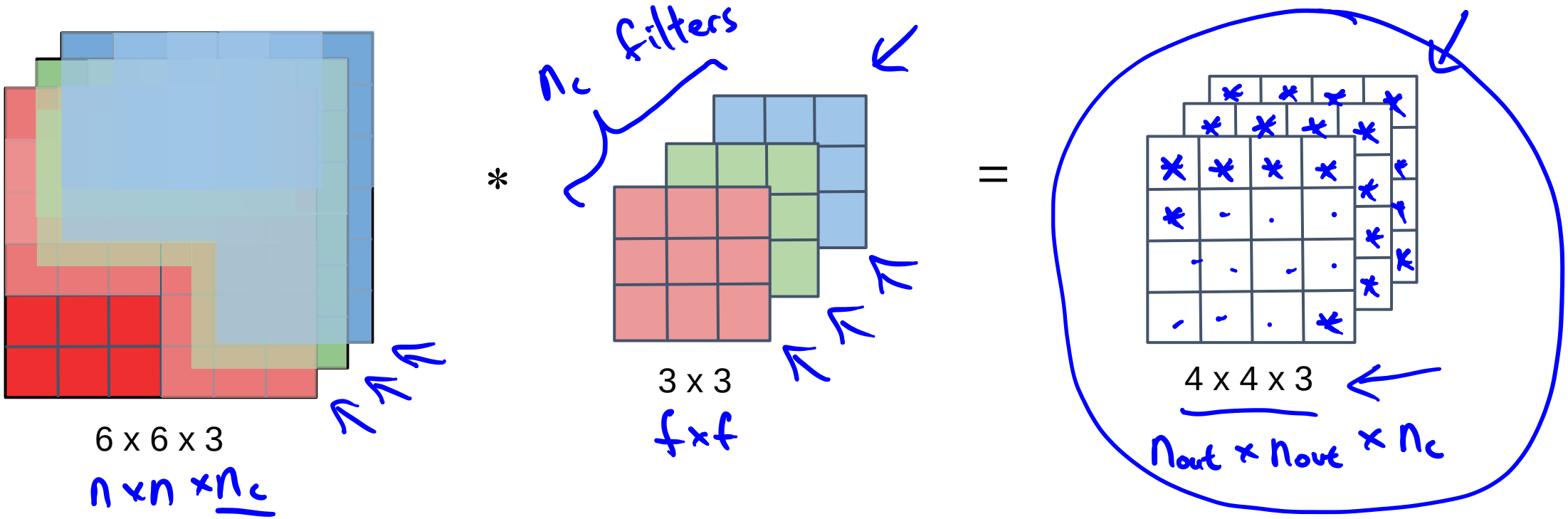


Depthwise Separable Convolution



Andrew Ng

Depthwise Convolution

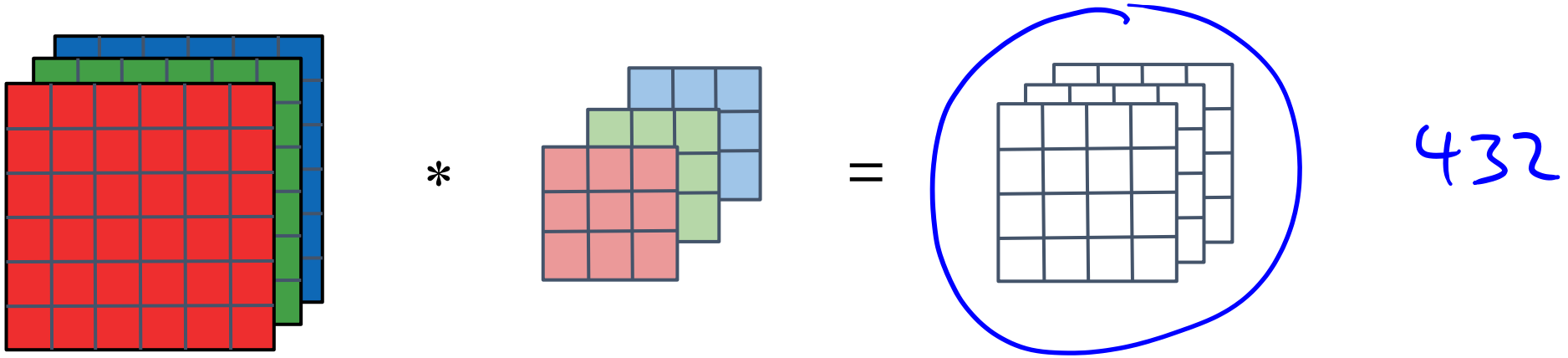


Computational cost	=	#filter params	x	# filter positions	x	# of filters
432	=	3x3	x	4x4	x	3

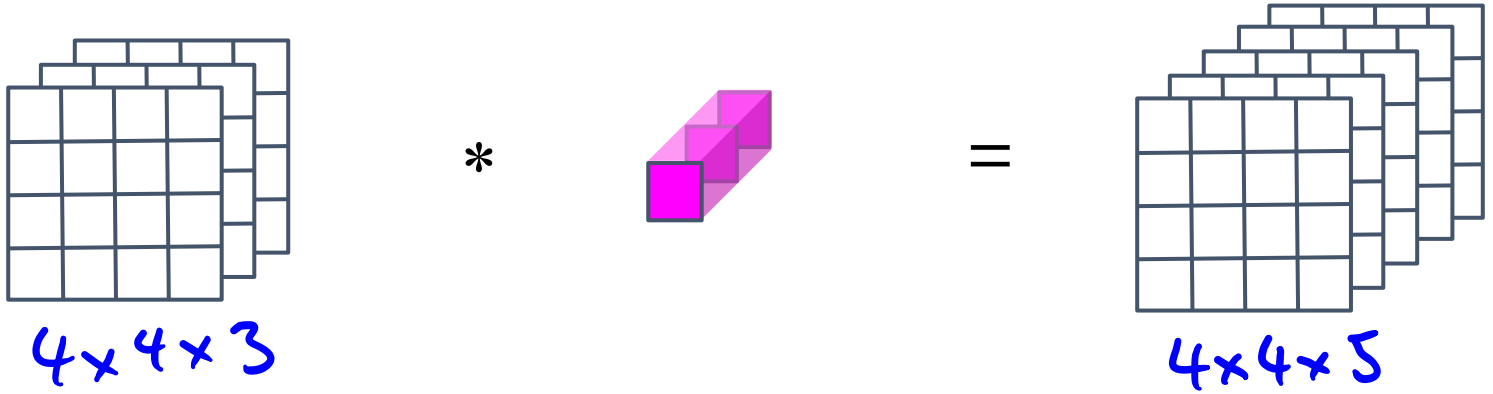
Andrew Ng

Depthwise Separable Convolution

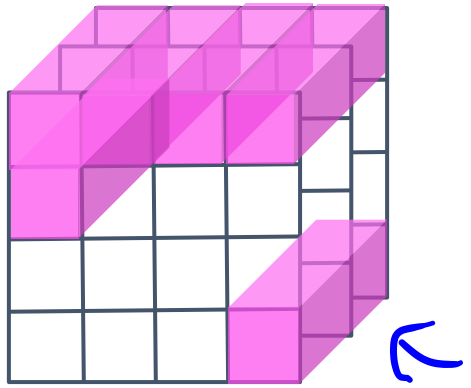
Depthwise Convolution



Pointwise Convolution



Pointwise Convolution



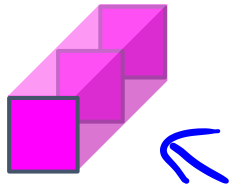
4 x 4 x 3

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

*

n_c' filters

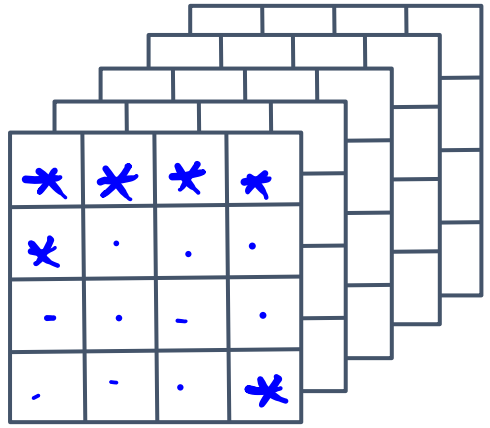
5



1 x 1 x 3

$1 \times 1 \times n_c$

=



4 x 4 x 5

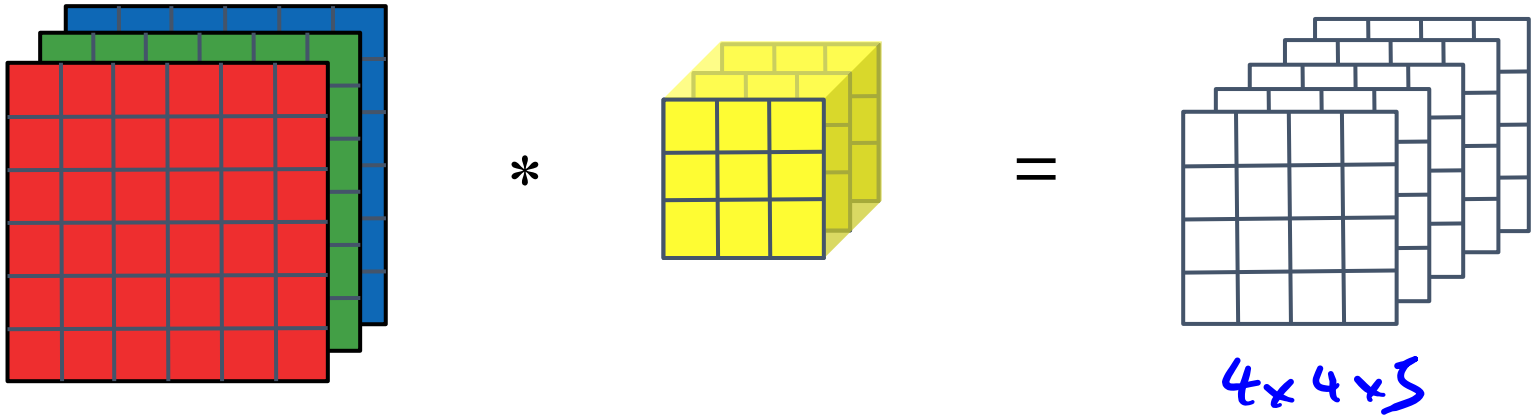
$n_{out} \times n_{out} \times n_c'$

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

Depthwise Separable Convolution

Normal Convolution

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

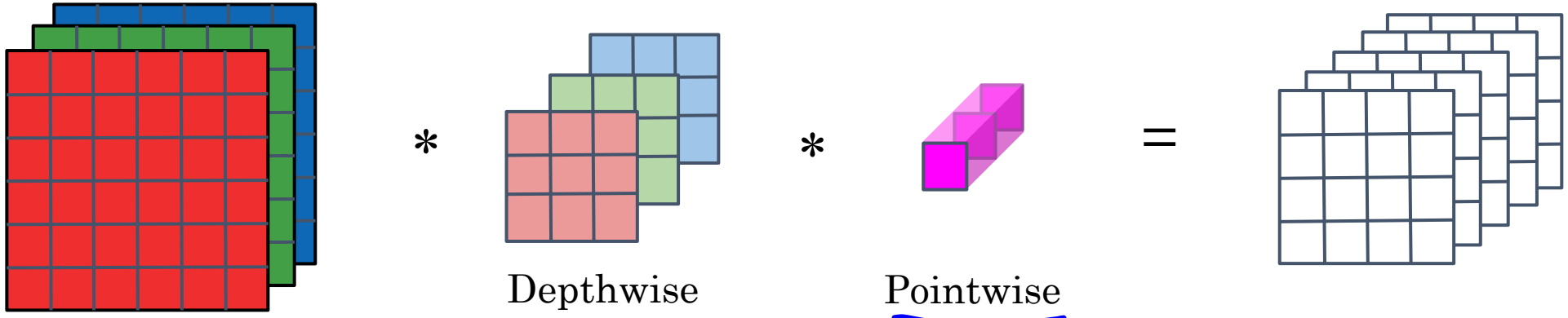


changed from x to +

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

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

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

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

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

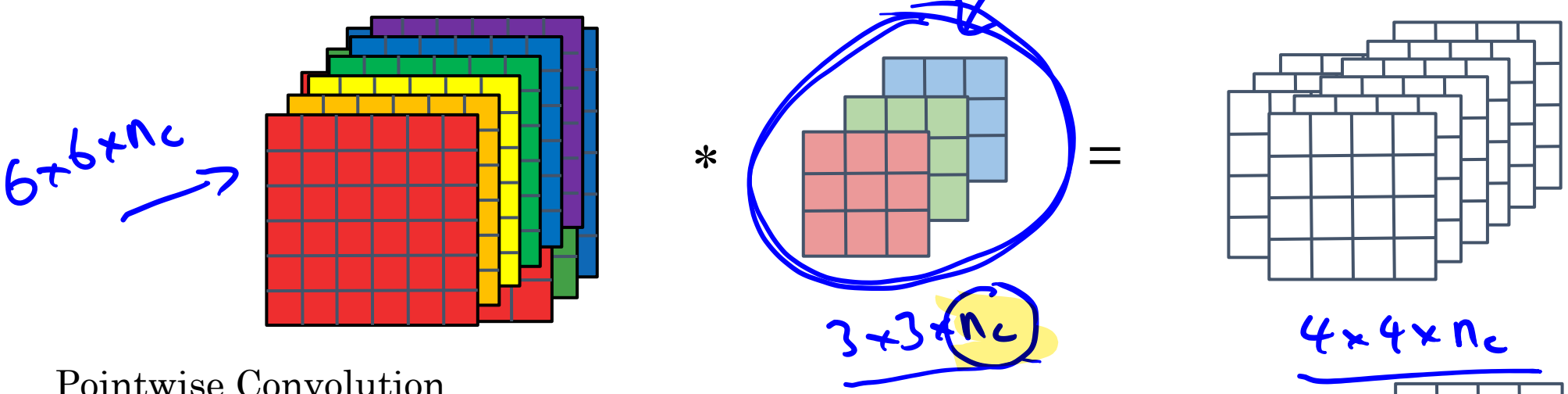
$\sim$10 times cheaper

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

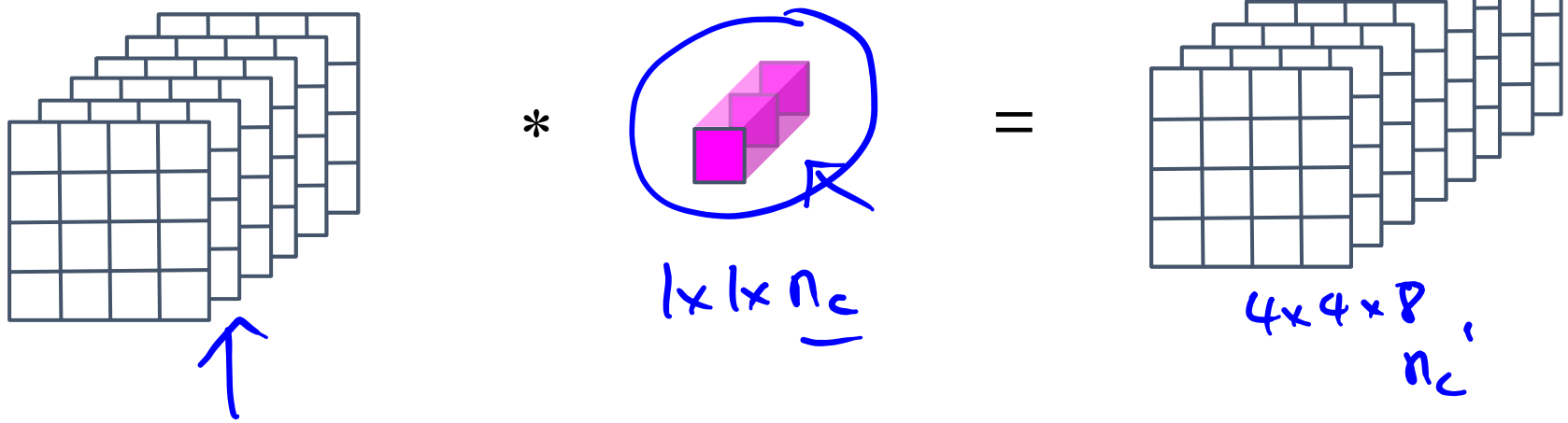
Depthwise Separable Convolution

Depthwise Convolution

icon to denote depth-wise convolution
w/ "appropriate" number of layers



Pointwise Convolution



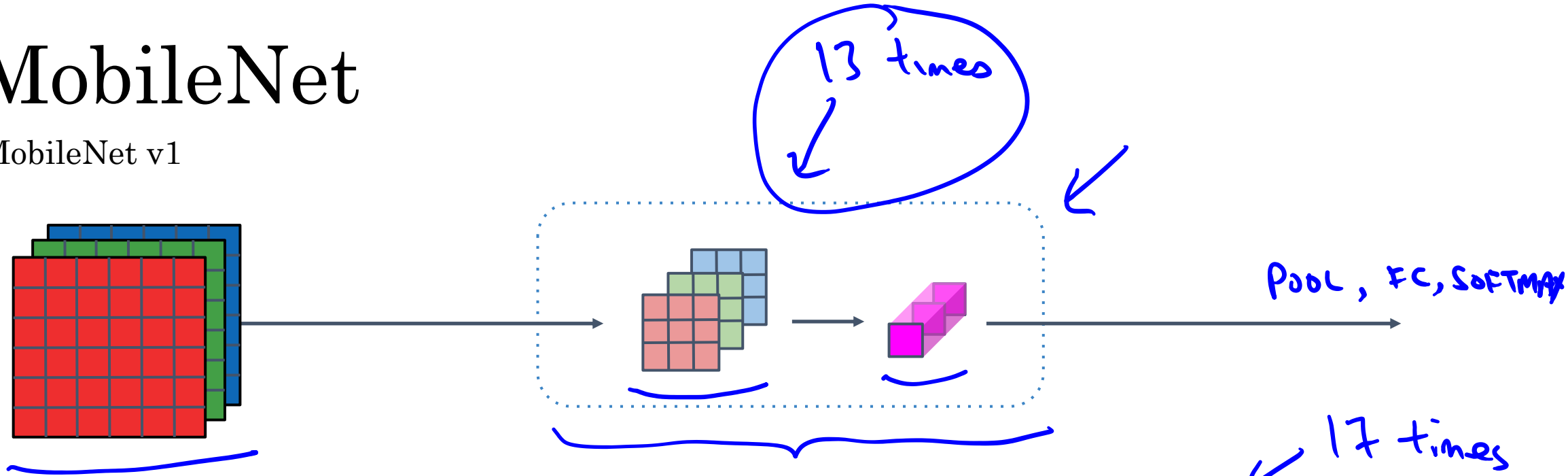
Andrew Ng

Convolutional Neural Networks

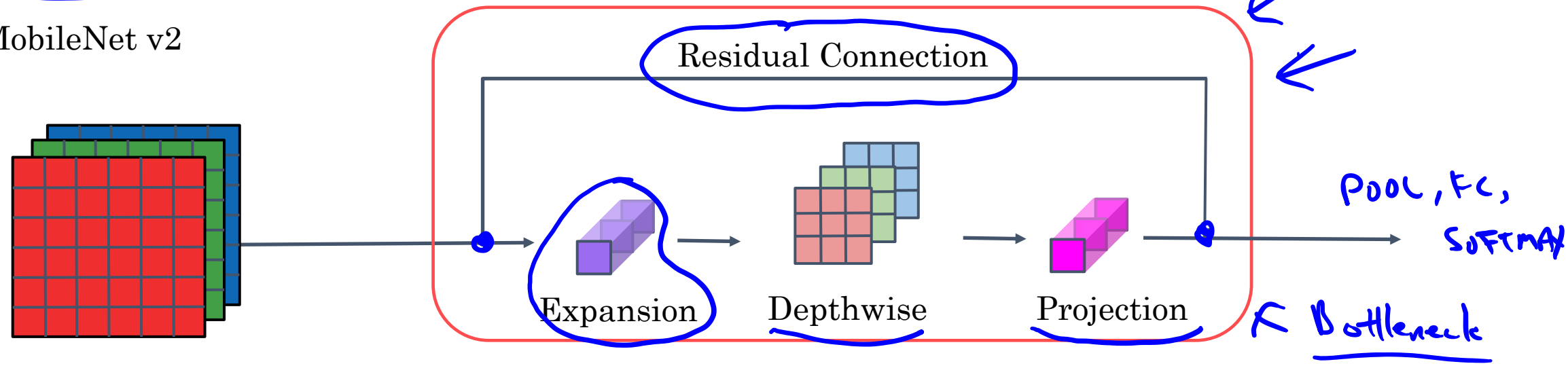
MobileNet Architecture

MobileNet

MobileNet v1



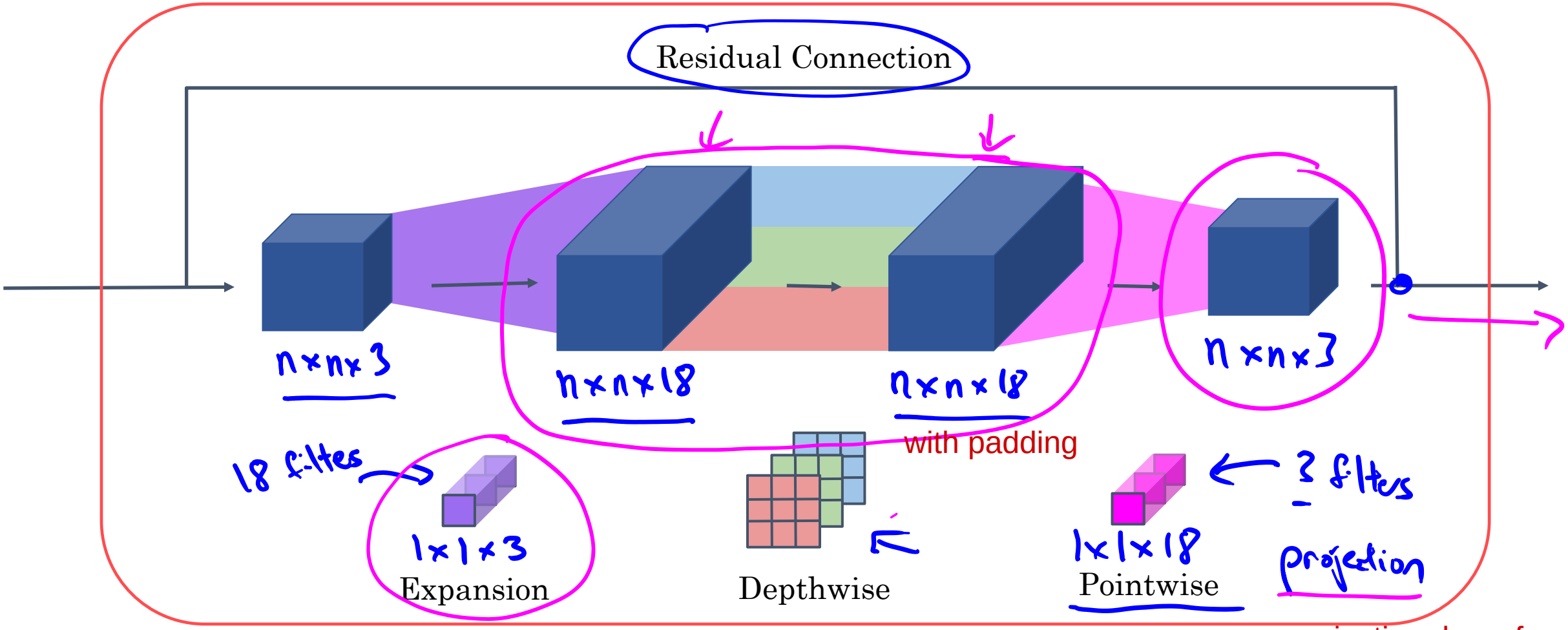
MobileNet v2



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

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

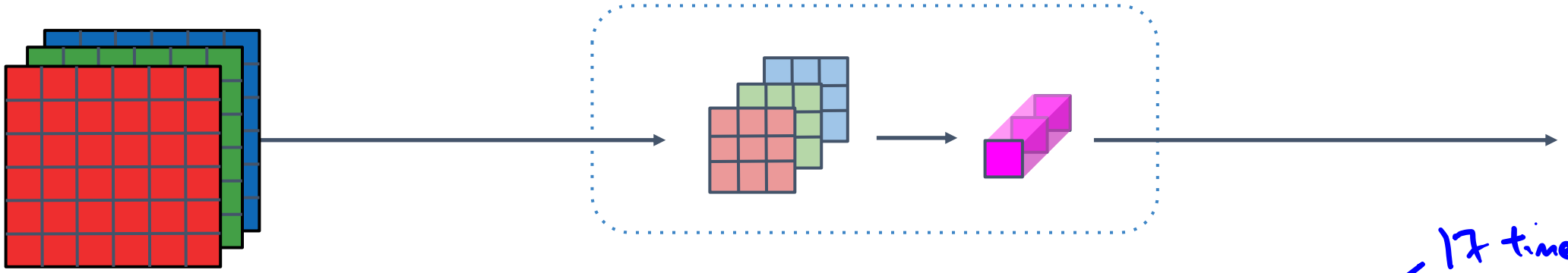


projecting down from nxnx18 to nxnx3

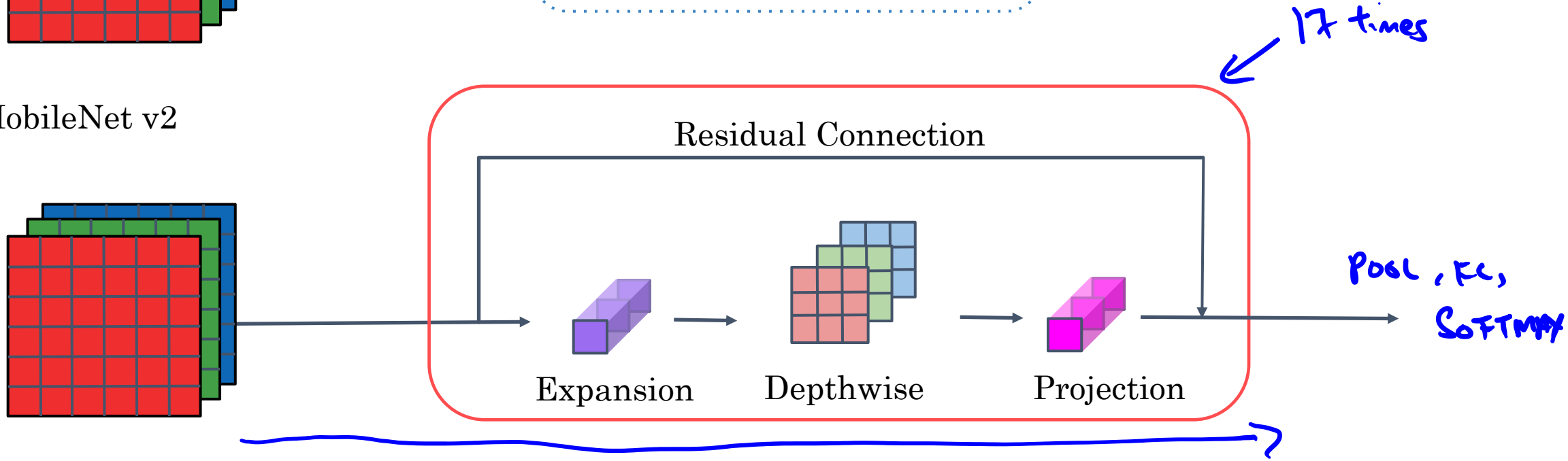
[Sandler et al. 2019, MobileNetV2: Inverted Residuals and Linear Bottlenecks]

MobileNet

MobileNet v1

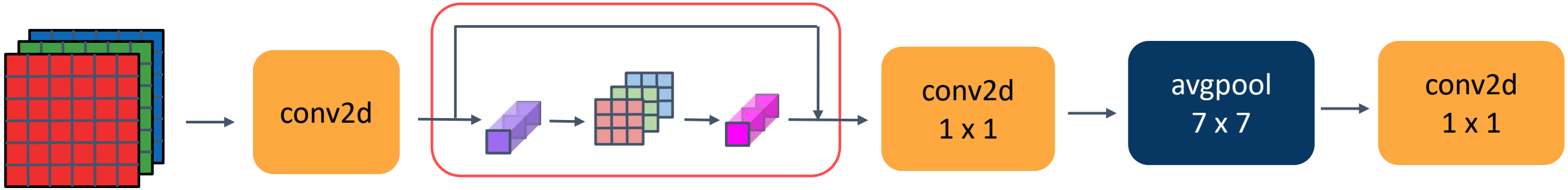


MobileNet v2



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

MobileNet v2 Full Architecture



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

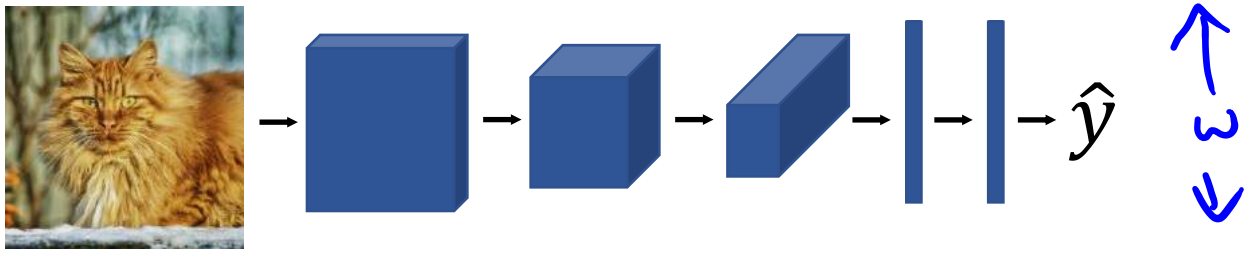
Convolutional Neural Networks

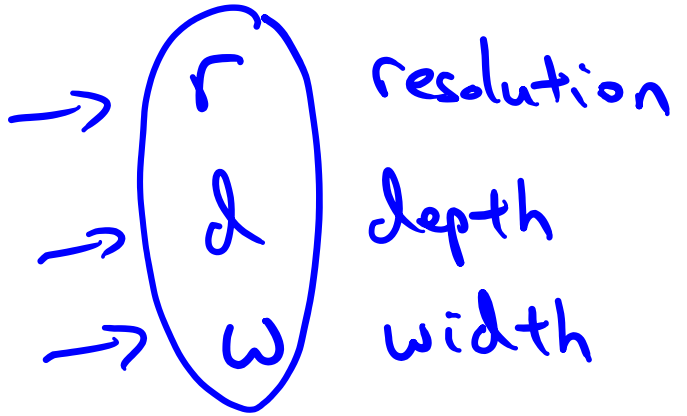
EfficientNet

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

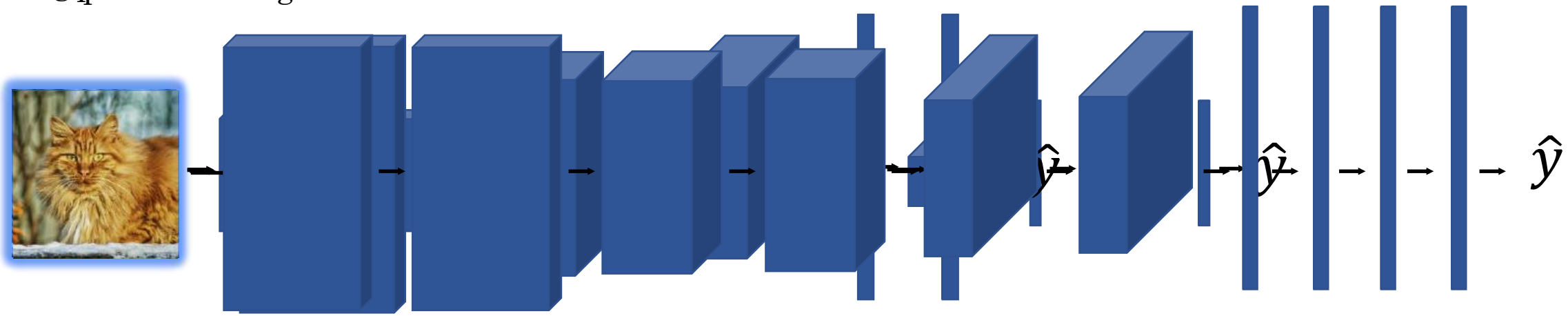
Baseline





Wider Resolution

Compound scaling

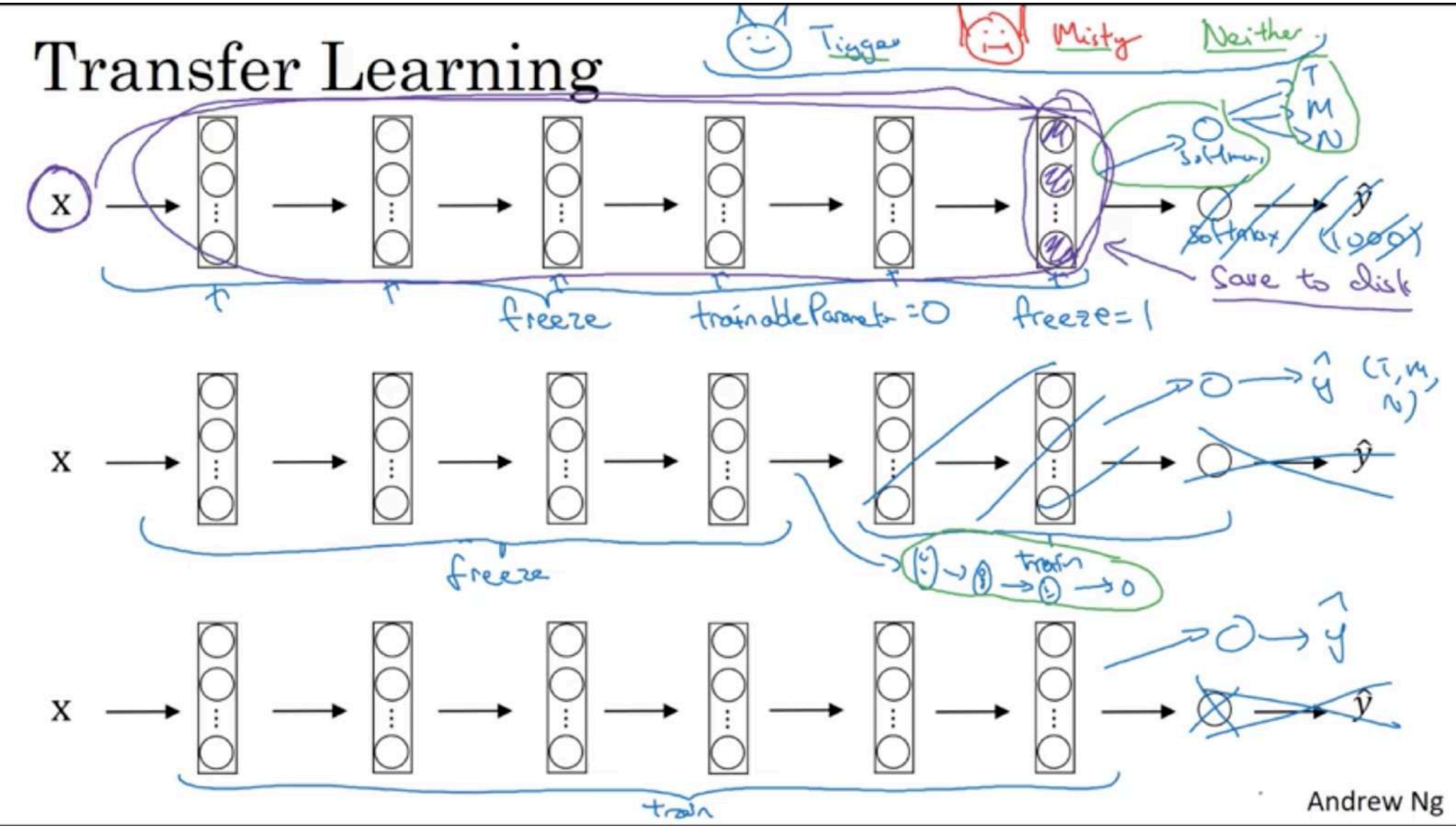


[Tan and Le, 2019, EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks]

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

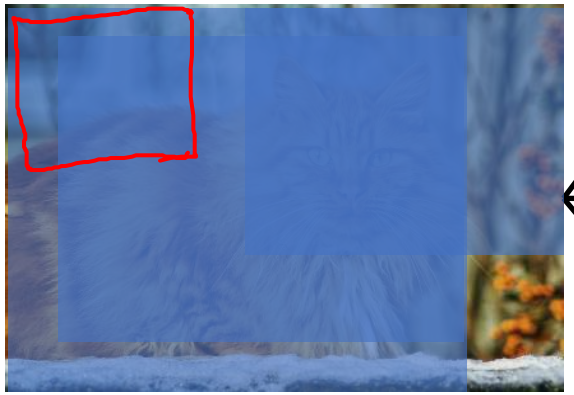
Data augmentation

Common augmentation method

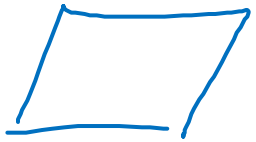
Mirroring

y

Random Cropping



- Rotation
- Shearing
- Local warping
- ...



Color shifting

R G B

+20,-20,+20

-20,+20,+20

+5,0,+50

Advanced:

PCA

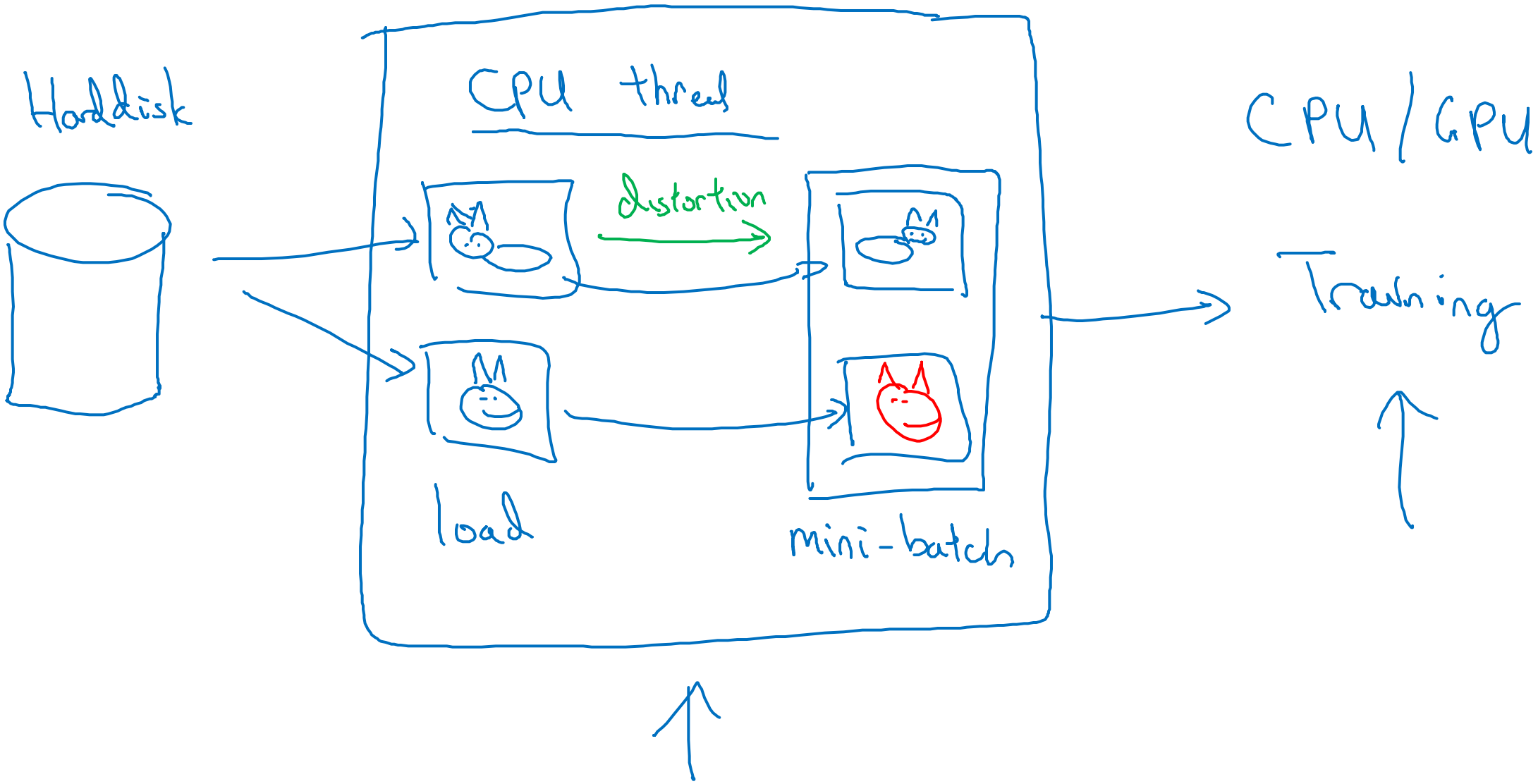
ml-class.org

[AlexNet paper

["PCA color augmentation."

R B G

Implementing distortions during training

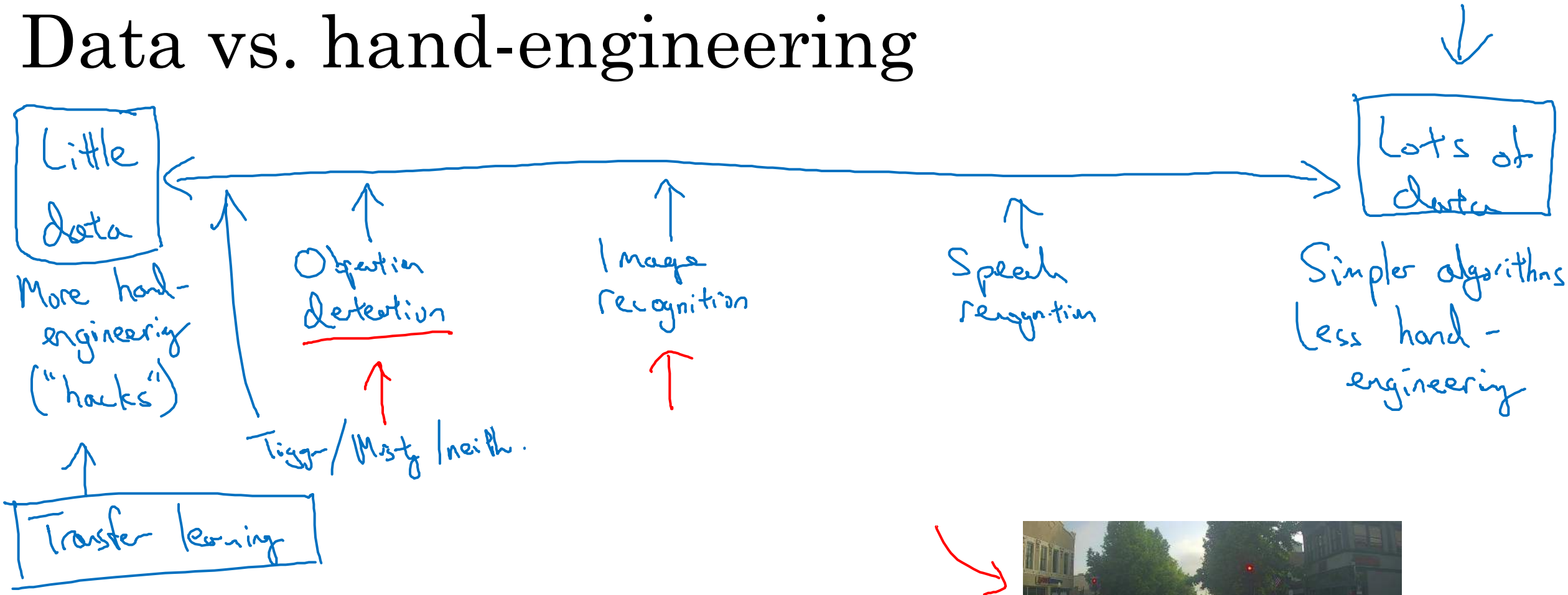


deeplearning.ai

Practical advice for using ConvNets

The state of computer vision

Data vs. hand-engineering



Two sources of knowledge

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

Andrew Ng

Tips for doing well on benchmarks/winning competitions

Ensembling

3-15 networks → $\hat{y}$

- 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

Copyright Notice

deeplearning.ai

Object Detection

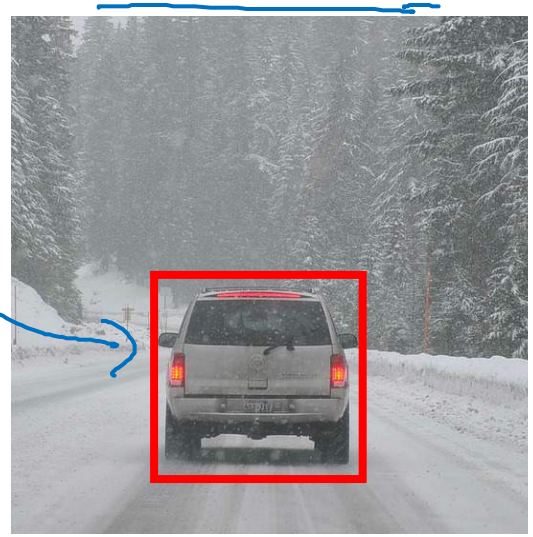
Object localization

What are localization and detection?

Image classification

"Car"

Classification with localization



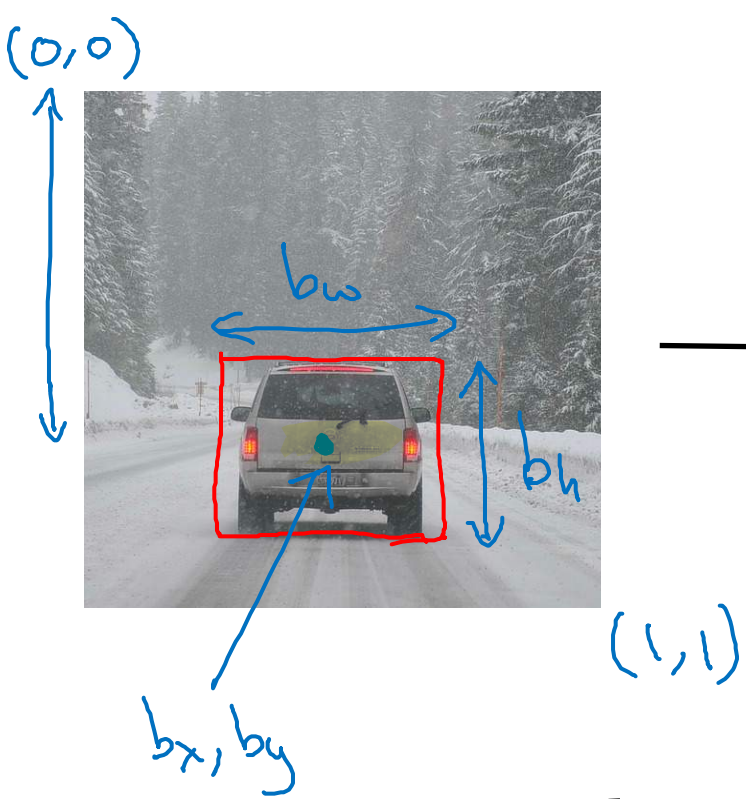
"Car"

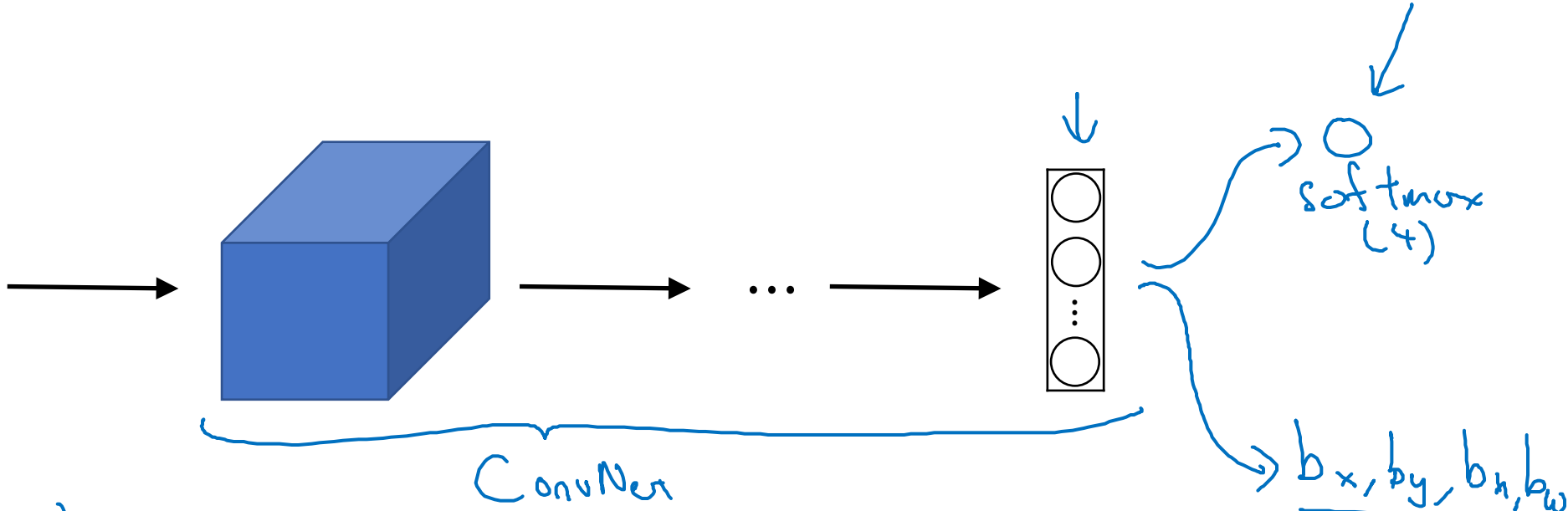
1 object

Detection

multiple objects

Classification with localization





1 - pedestrian ←

2 - car ←

3 - motorcycle ←

4 - background

$b_x = 0.5$

$b_y = 0.7$

$b_h = 0.3$

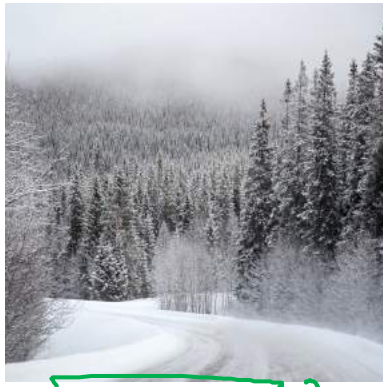
$b_w = 0.4$

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)

$x =$



$y =$ p_c (is there an object?), b_x, b_y, b_h, b_w, c_1, c_2, c_3

(x, y): $p_c = 1$, b_x, b_y, b_h, b_w, 0, 1, 0

$p_c = 0$, ?, ?, ?, ?, ?, ?, ?, ? ← "don't care"

$\mathcal{L}(\hat{y}, y) =$ RMSE (ideally use RMSE + LogLoss)

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

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

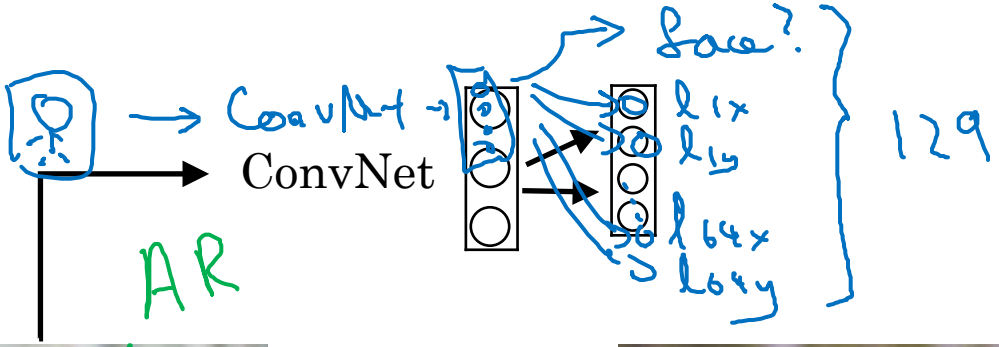
Andrew Ng

deeplearning.ai

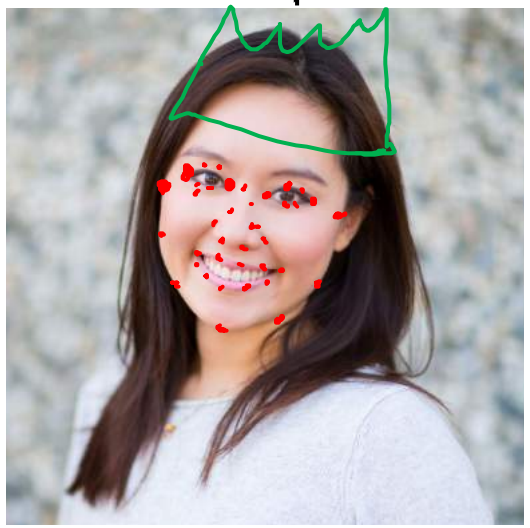
Object Detection

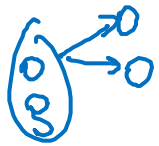
Landmark detection

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_{64}, l_{64y}

X, y

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

Andrew Ng

deeplearning.ai

Object Detection

Object detection

Car detection example

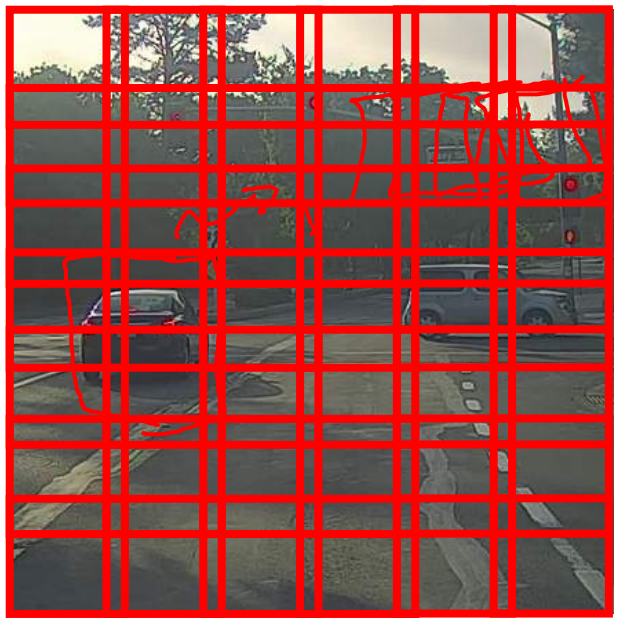
Training set:

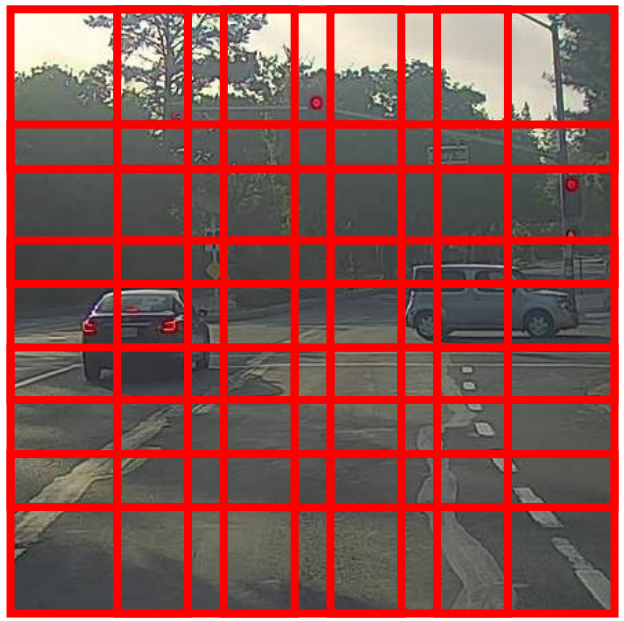
x	y
	1
	1
	1
	0
	0

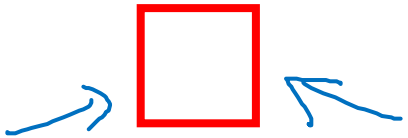
→ ConvNet → y

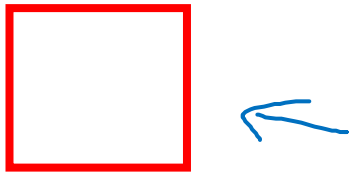
Sliding windows detection

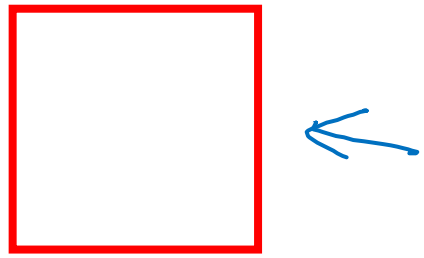
Disadvantage - computational cost of running multiple sliding windows. Originally (pre-CNN), classifiers were simple linear classifiers, so sliding window was feasible. Not so with CNNs.











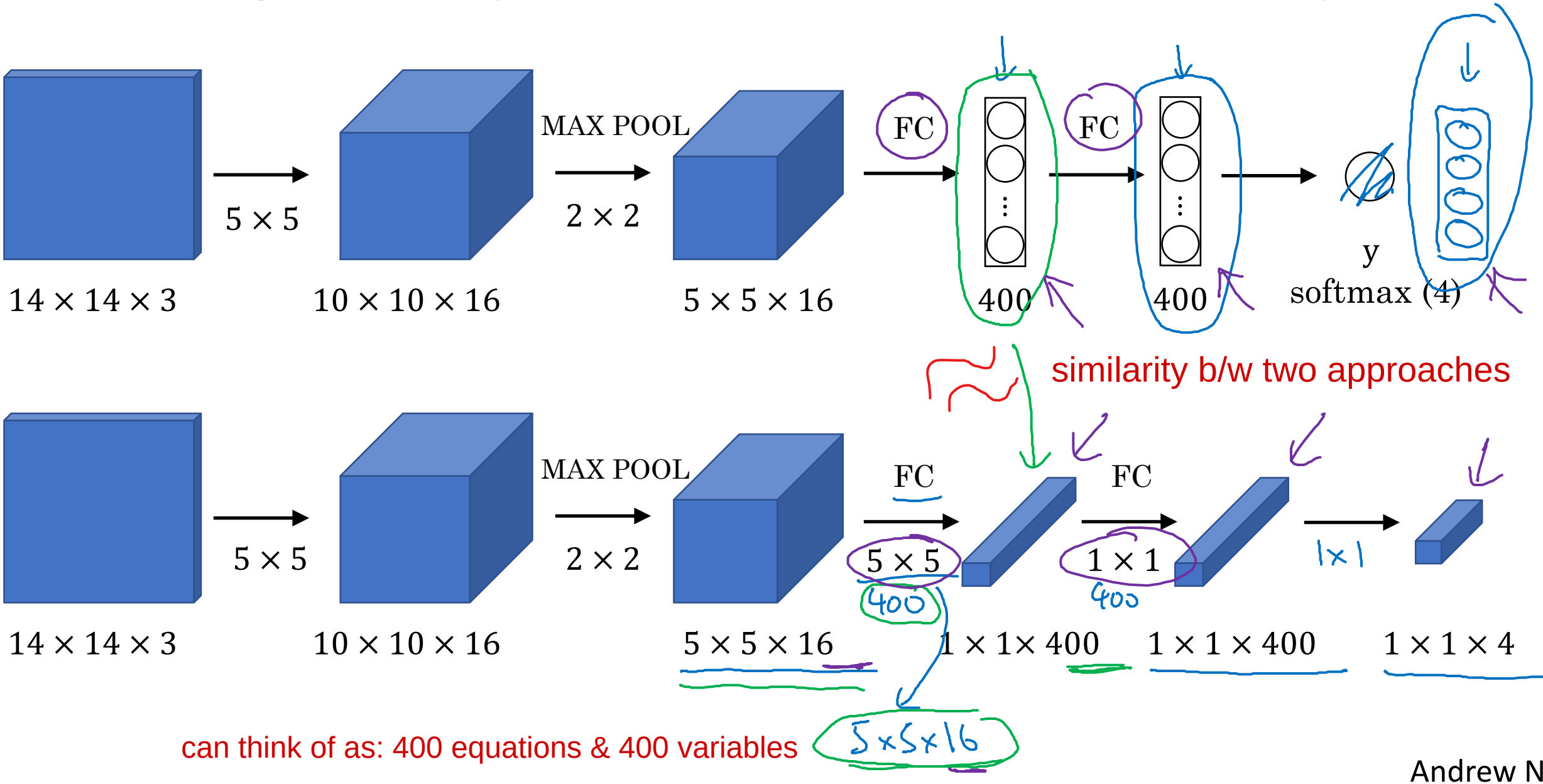
Andrew Ng

deeplearning.ai

Object Detection

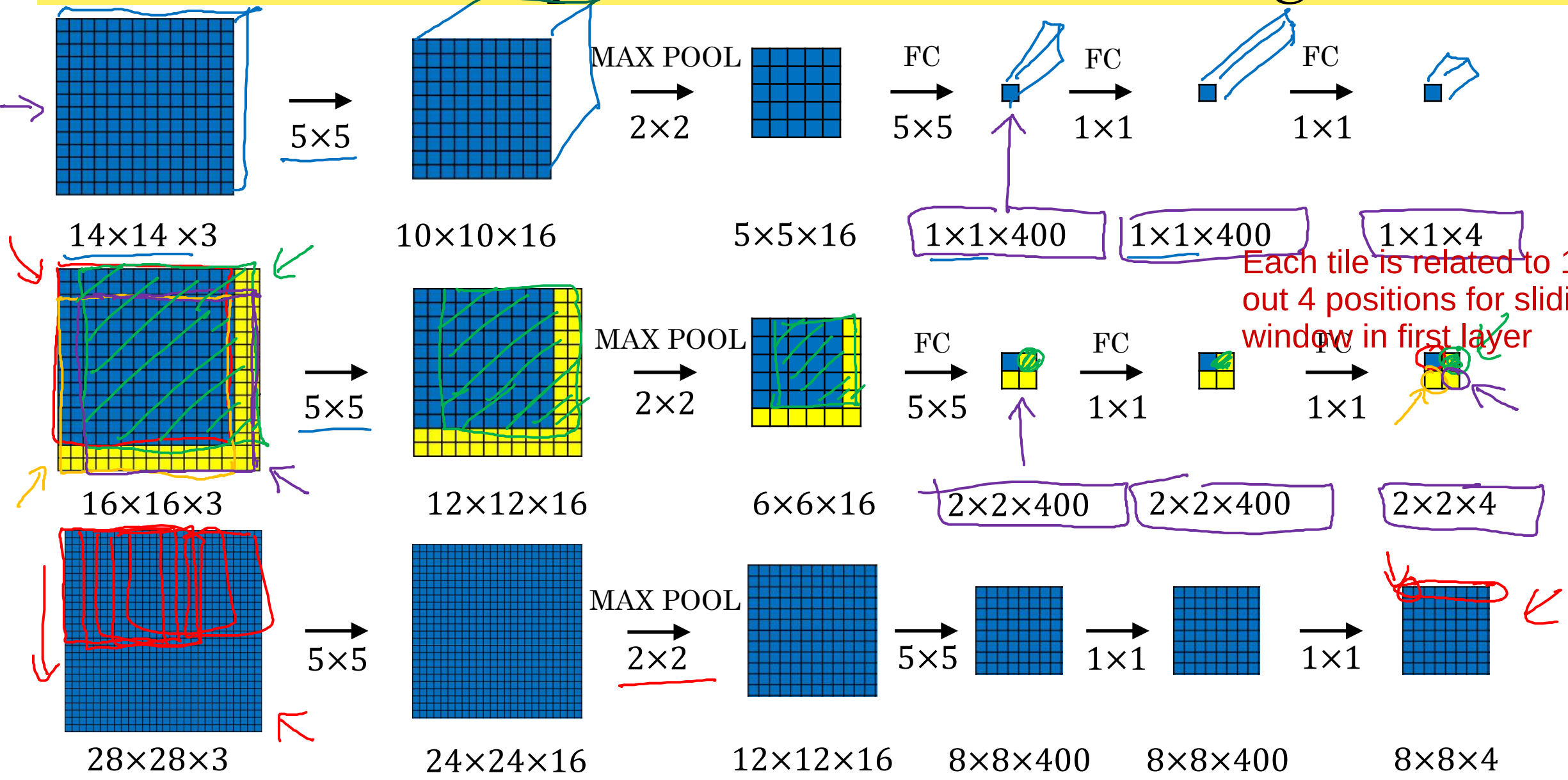
Convolutional implementation of sliding windows

Turning FC layer into convolutional layers



Andrew Ng

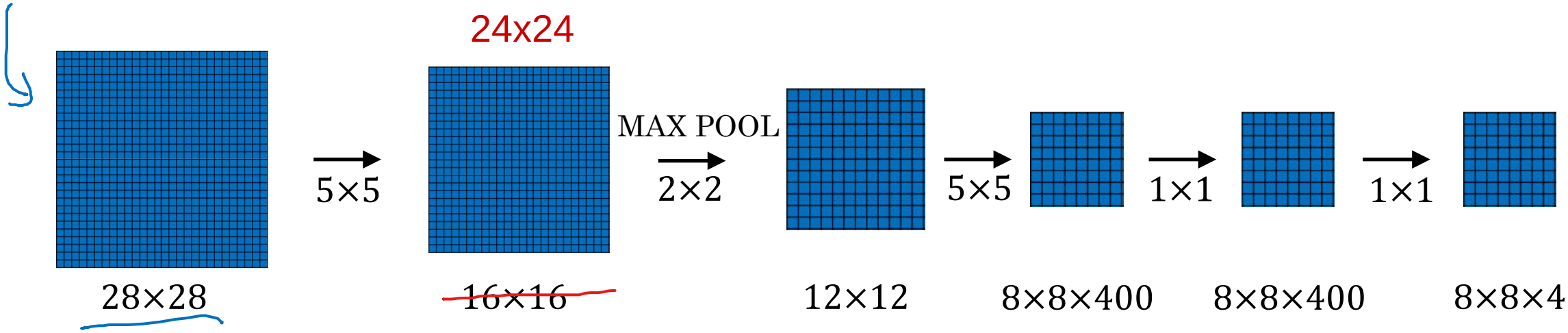
Convolution implementation of sliding windows

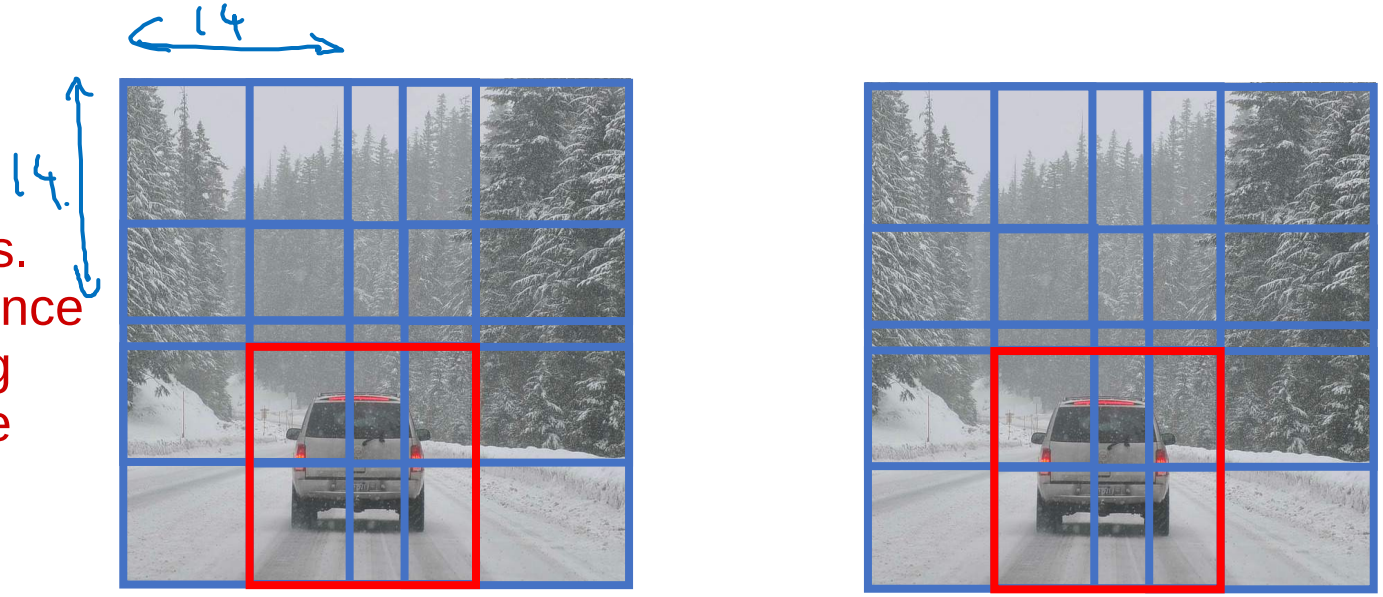


[Sermanet et al., 2014, OverFeat: Integrated recognition, localization and detection using convolutional networks]

Andrew Ng

Convolution implementation of sliding windows





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

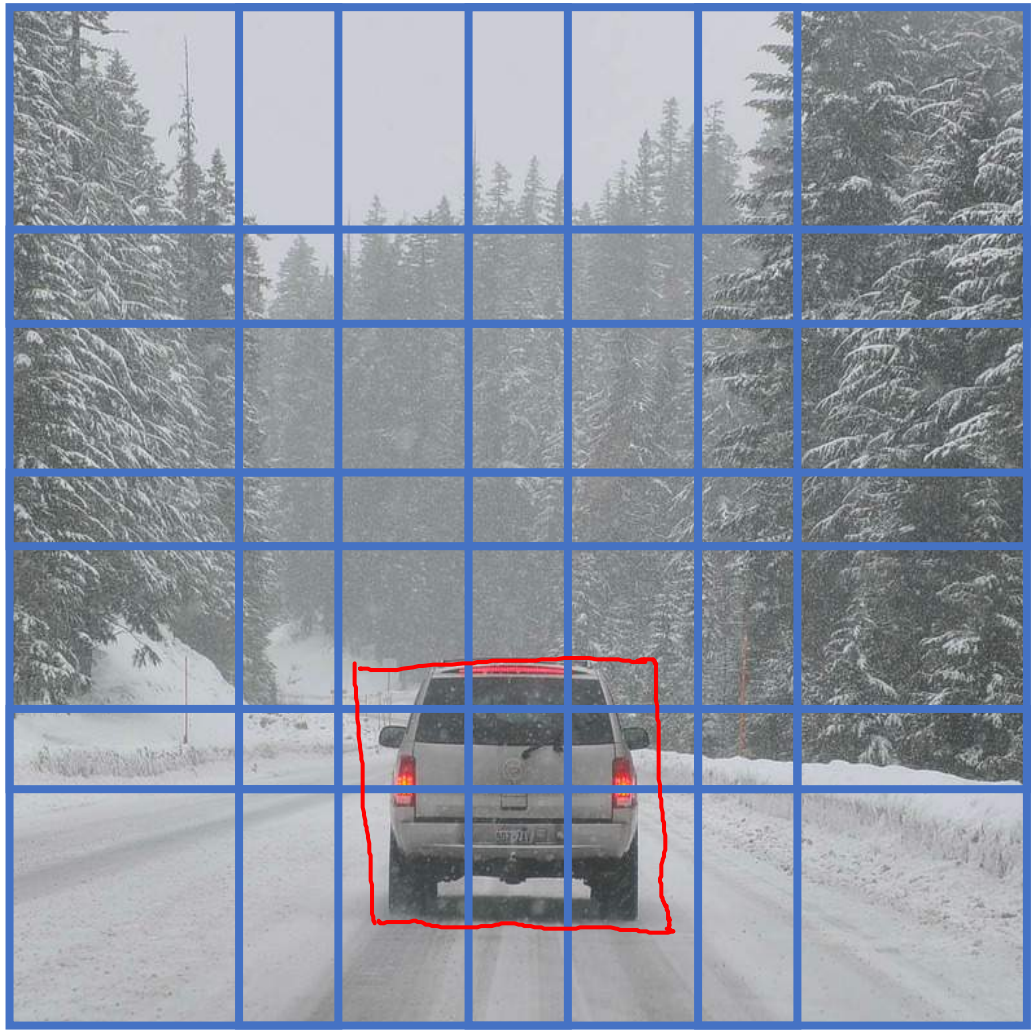
Andrew Ng

deeplearning.ai

Object Detection

Bounding box predictions

Output accurate bounding boxes



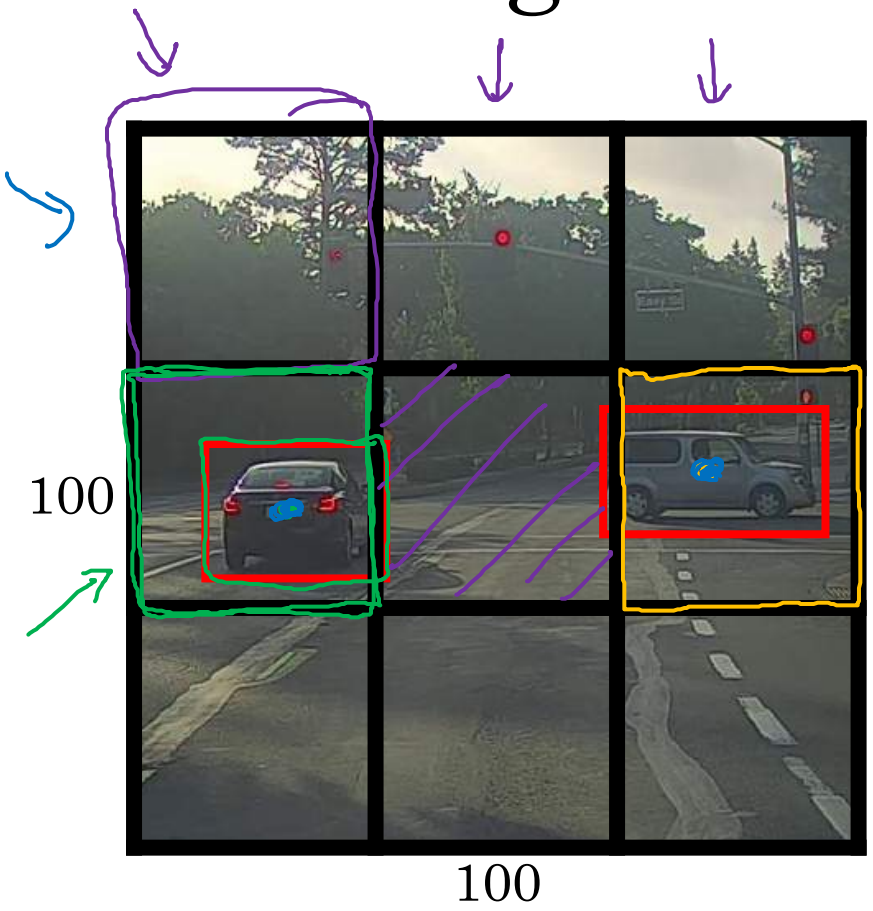
Andrew Ng

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

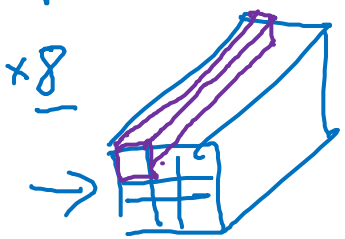
Labels for training

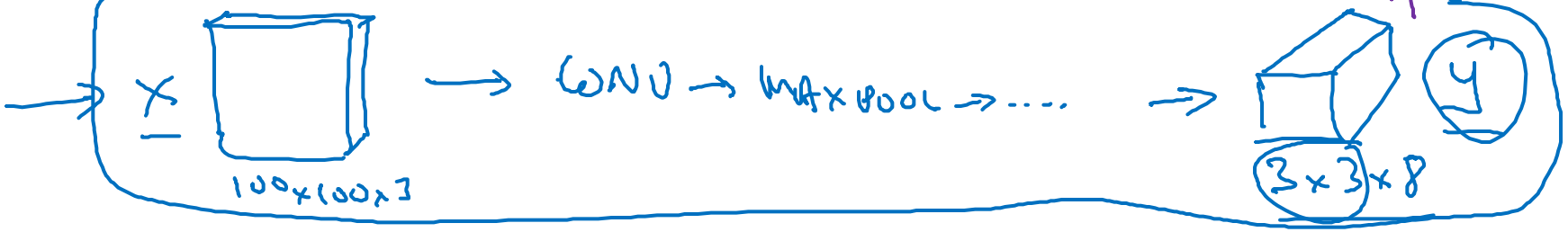
For each grid cell:

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

Target output:

3x3x8





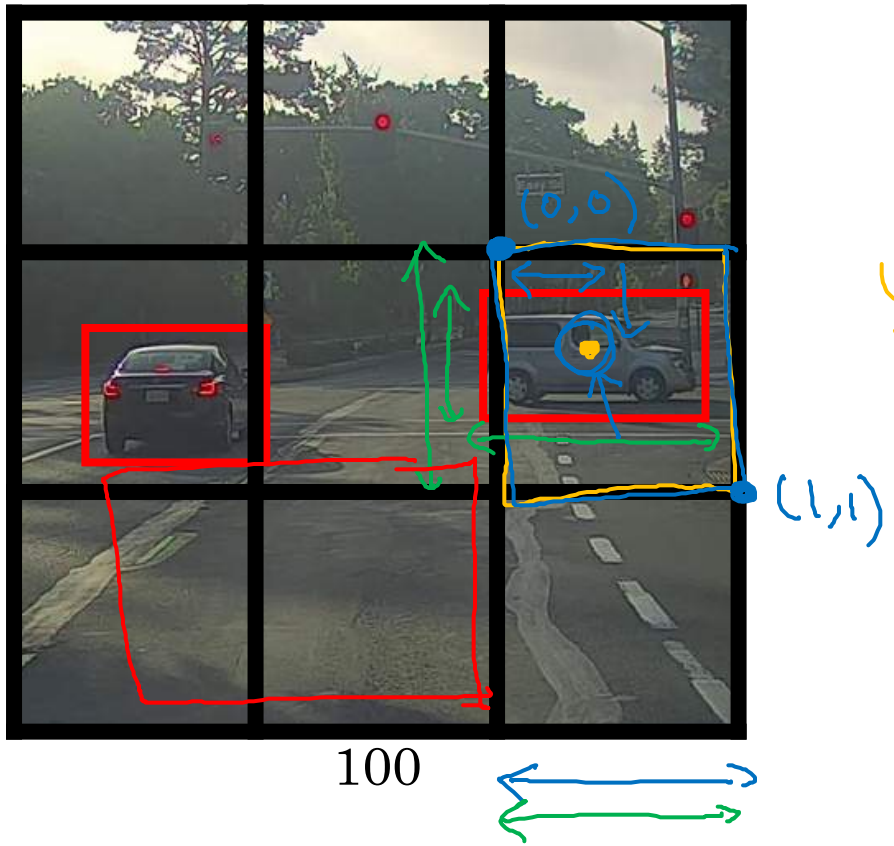
361

→ 19x19x8

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

Andrew Ng

Specify the bounding boxes



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

0.4, 0.3, 0.9, 0.5 } between 0 and 1

could be > 1

bottom two values can be >1

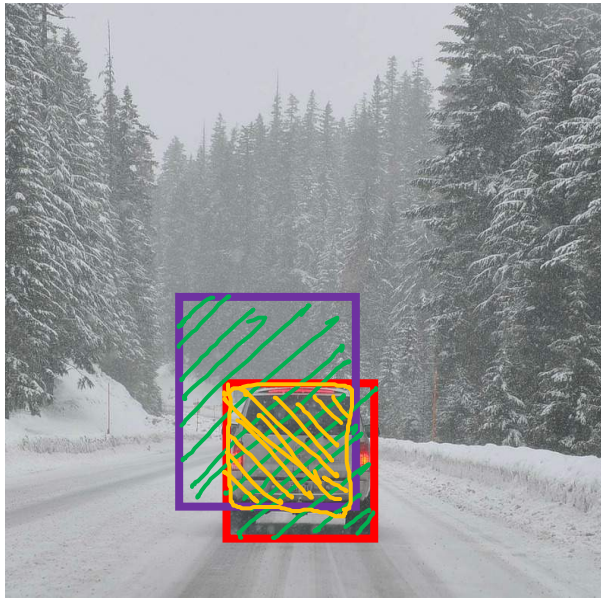
[Redmon et al., 2015, You Only Look Once: Unified real-time object detection]

deeplearning.ai

Object Detection

Intersection over union

Evaluating object localization



Intersection over Union (IoU)

$$= \frac{\text{Size of } \square}{\text{Size of } \square}$$

"Correct" if IoU $\geq$ 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

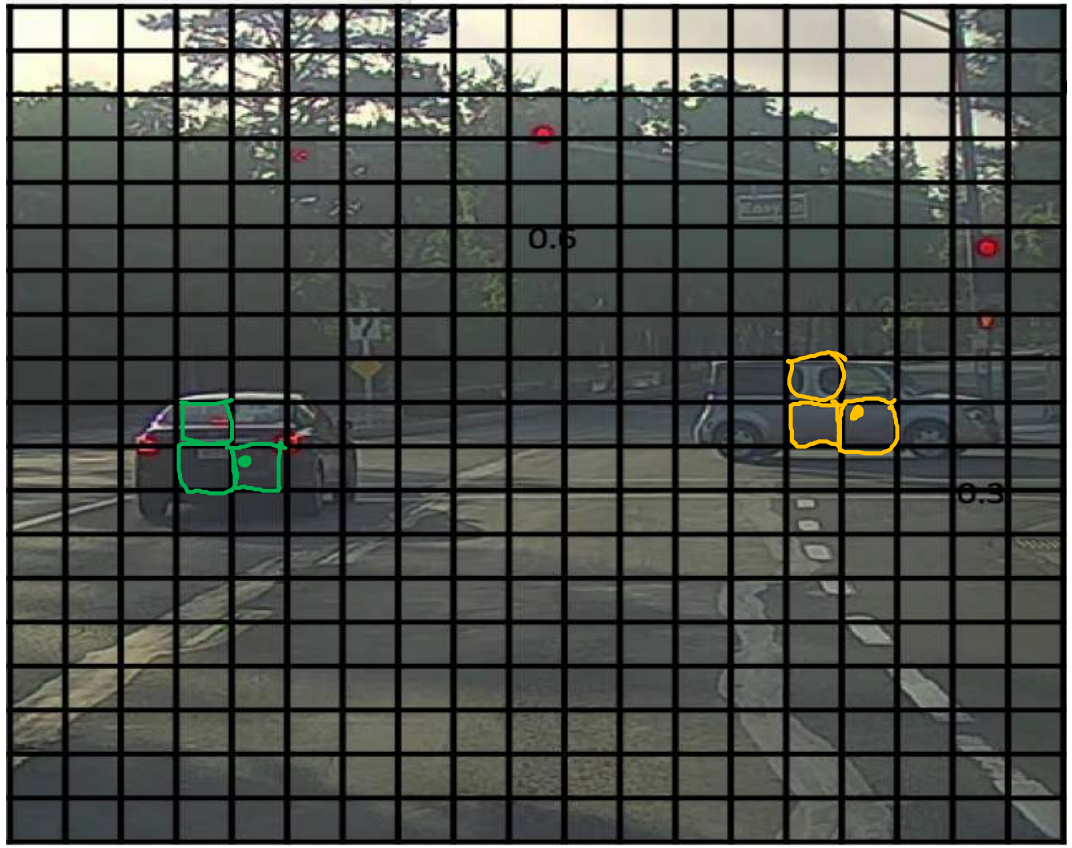
deeplearning.ai

Object Detection

Non-max suppression

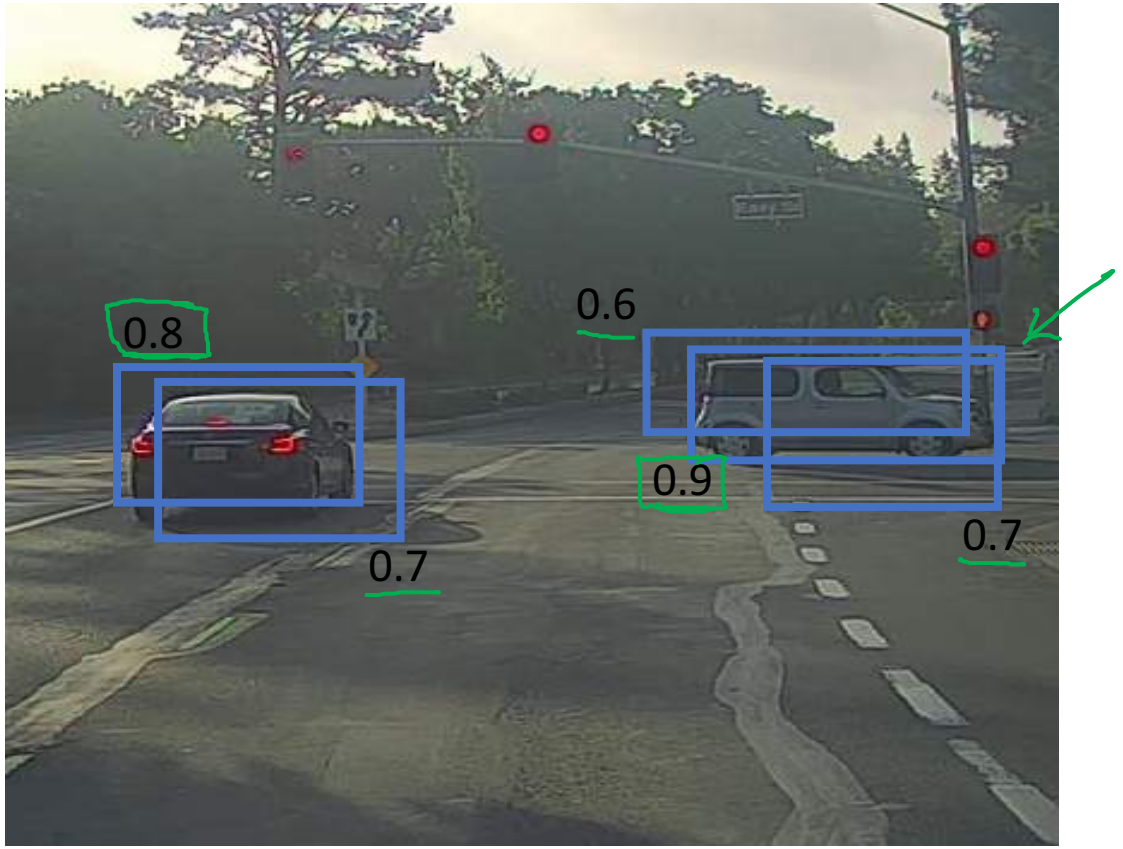
Non-max suppression example

Non-max suppression example



19×19

Non-max suppression example



Pc

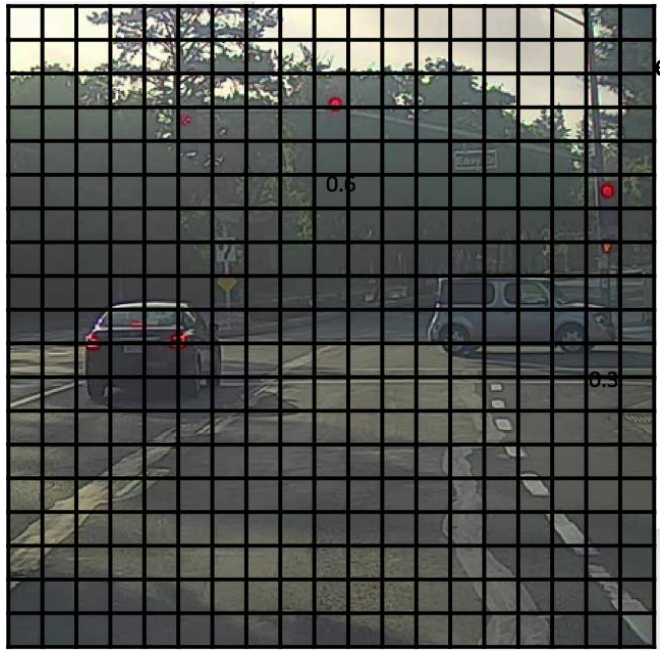
Issue - many boxes will detect any single car and generate Pc > 0

- 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



19×19

Each output prediction is:

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

361 outputs

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

Andrew Ng

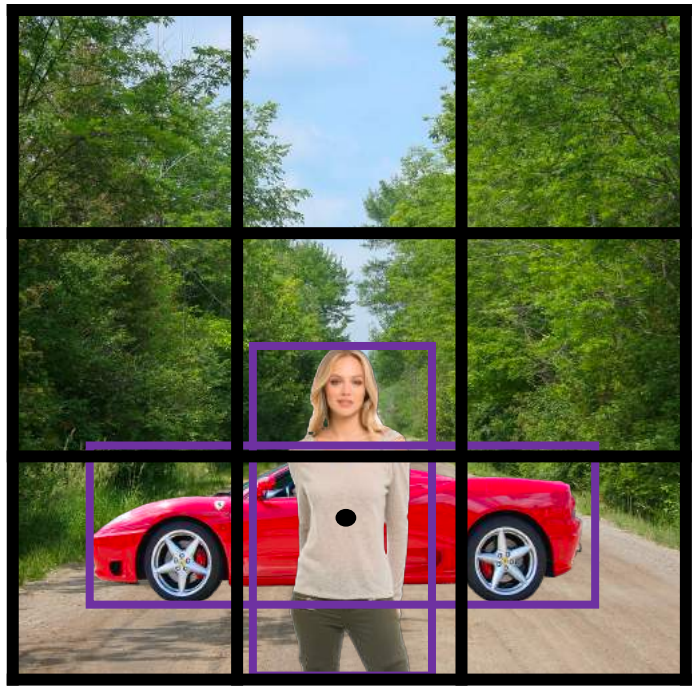
deeplearning.ai

Object Detection

Anchor boxes

Overlapping objects:

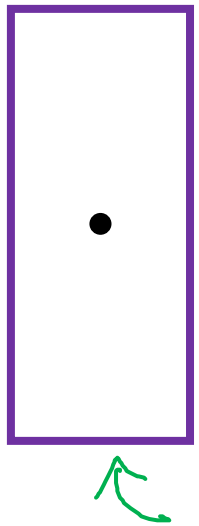
SOLUTION: suppose there can be two kinds of bounding boxes, include both as part the output vector



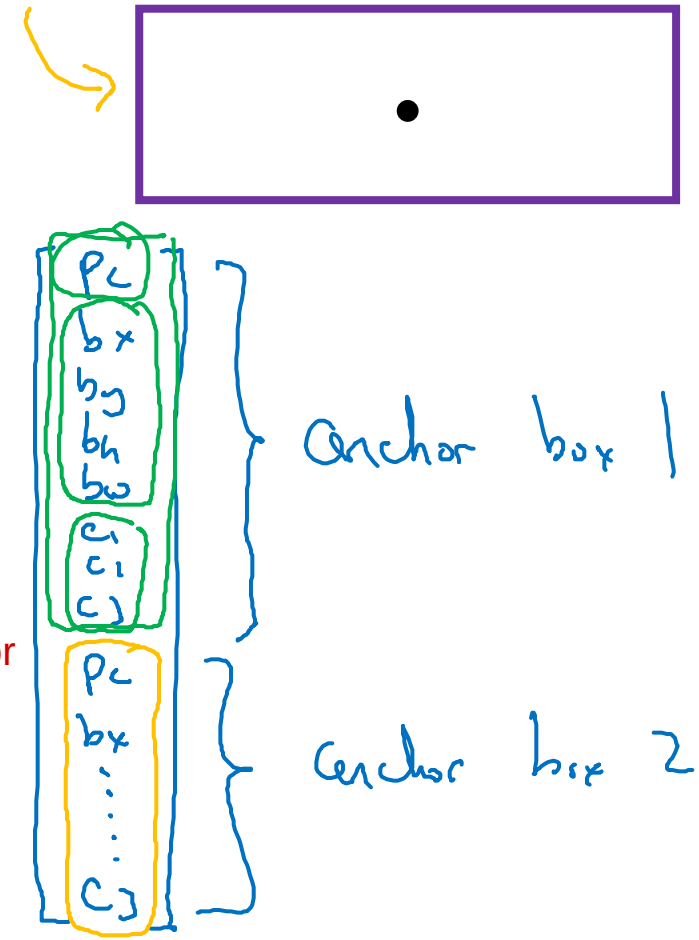
$$y = \begin{bmatrix} p_c \\ b_x \\ b_y \\ b_h \\ b_w \\ c_1 \\ c_2 \\ c_3 \end{bmatrix}$$

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:



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

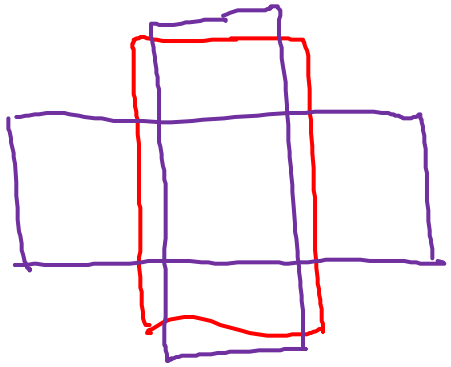
Anchor box algorithm

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.

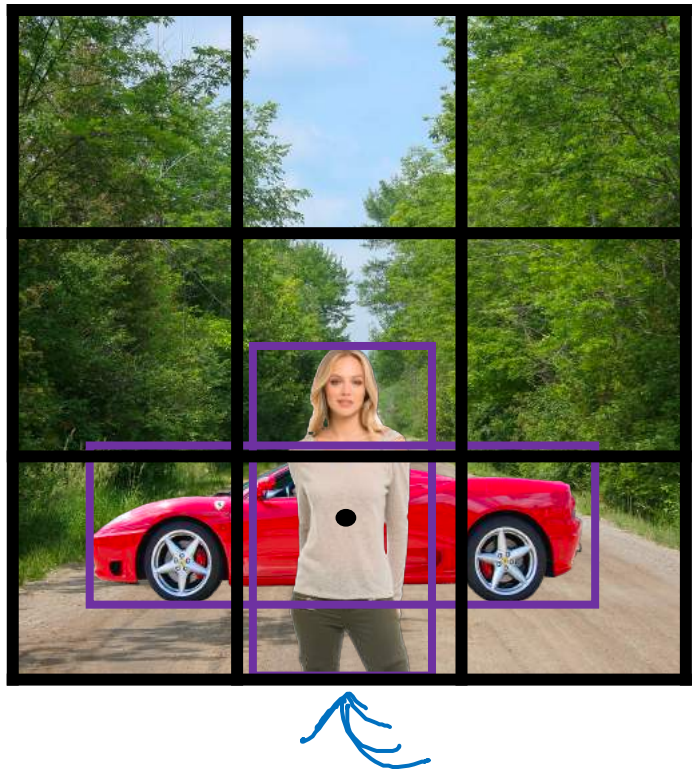
(grid cell, anchor box)

Output y:

$3 \times 3 \times 16$

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

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

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

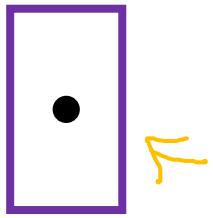
person

car

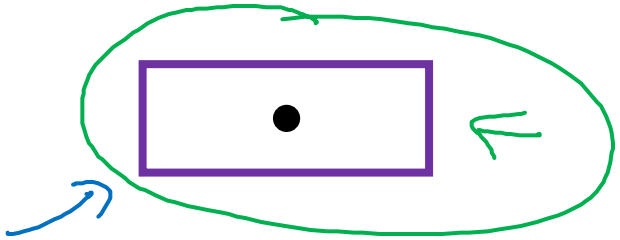
anchor box 1

anchor box 2

Anchor box 1:



Anchor box 2:



deeplearning.ai

Object Detection

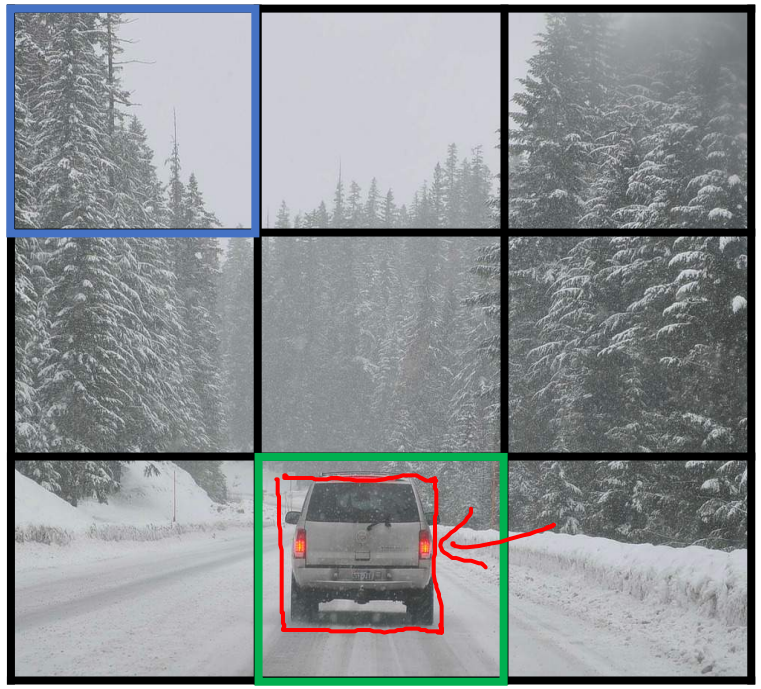
Putting it together: YOLO algorithm

Training

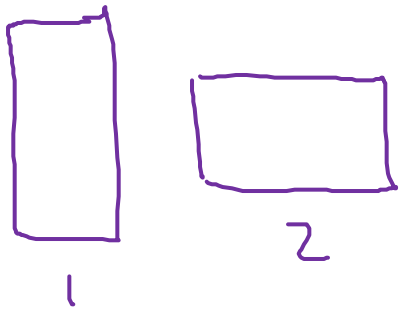
1 - pedestrian

2 - car

3 - motorcycle



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



more overlap with 2nd anchor box

3×3×16

y is 3×3×2×8

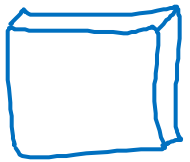
training set for each image

19×19×16

19×19×40

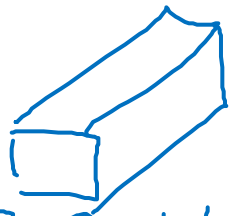
#anchors

5 + #classes



100×100×3

→ ConvNet →

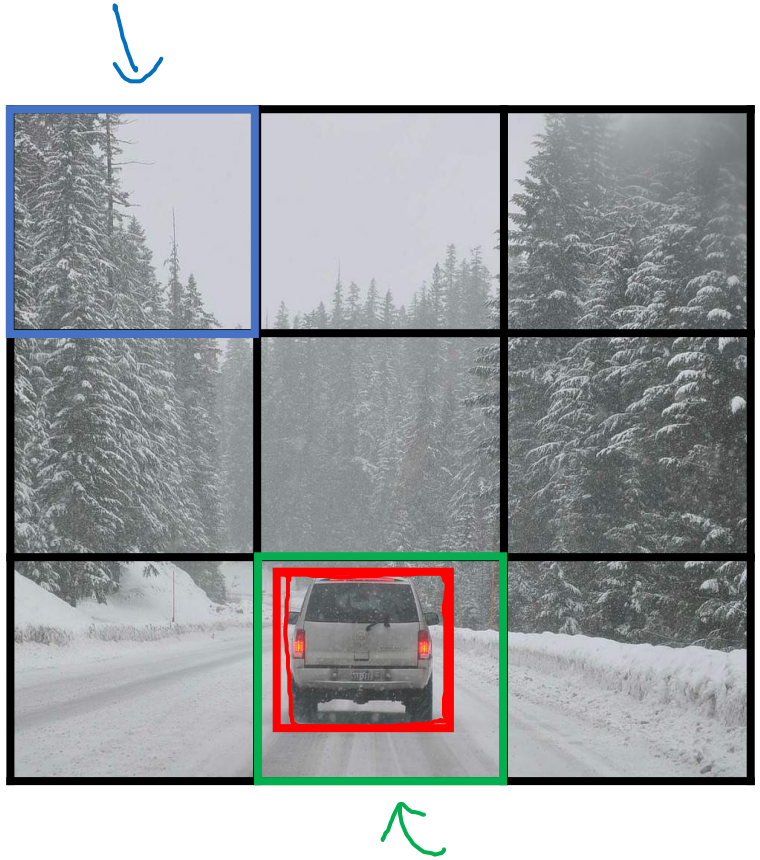


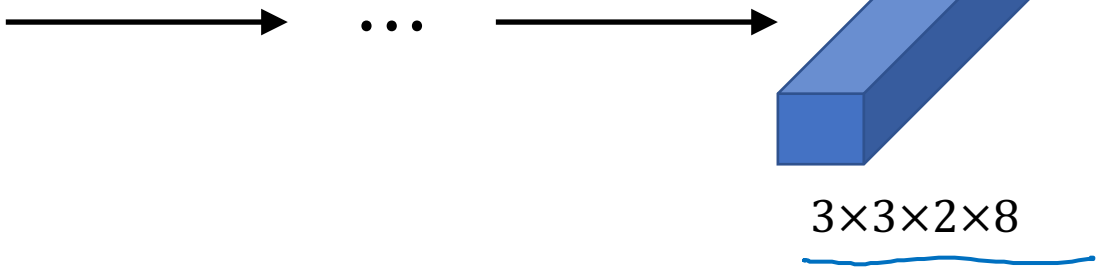
convnet will learn the magic trick

3×3×16

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

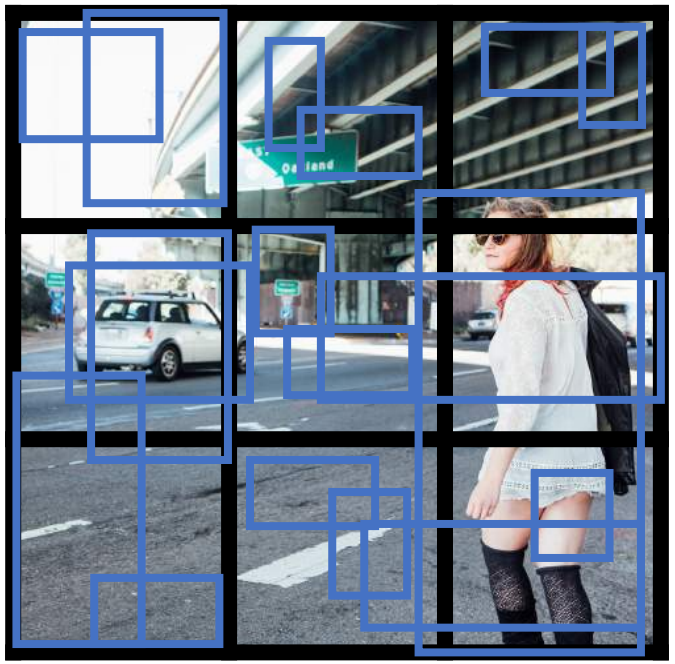
Making predictions





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

Outputting the non-max supressed outputs



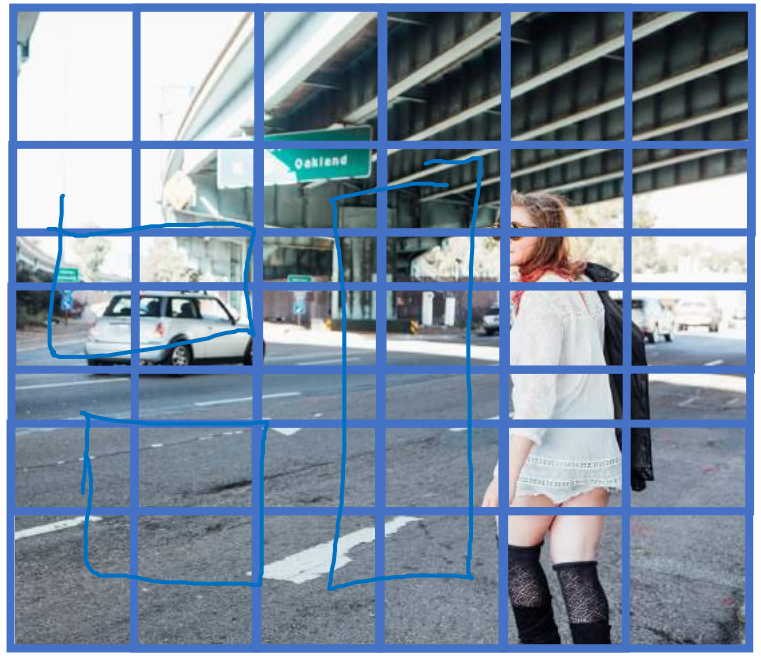
- For each grid call, get 2 predicted bounding boxes.
- Get rid of low probability predictions. many of these boxes will go away
- For each class (pedestrian, car, motorcycle) use non-max suppression to generate final predictions.

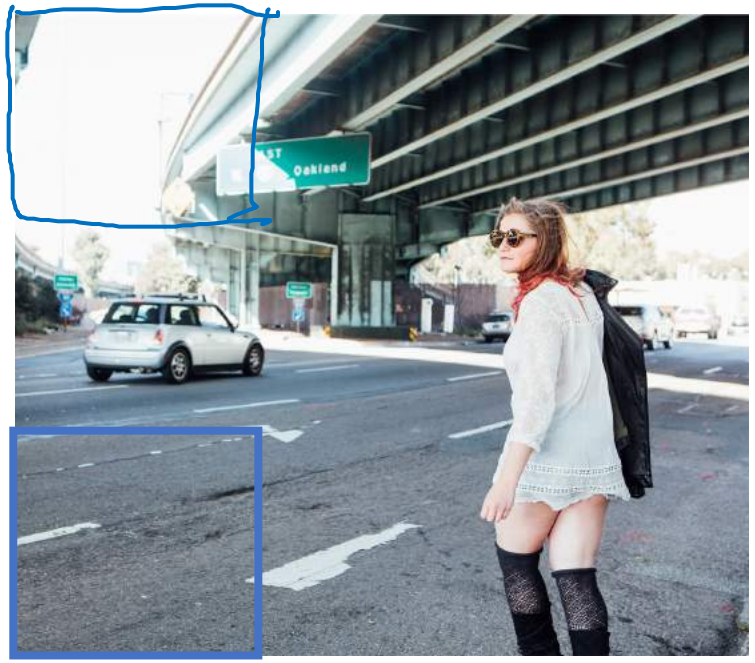
deeplearning.ai

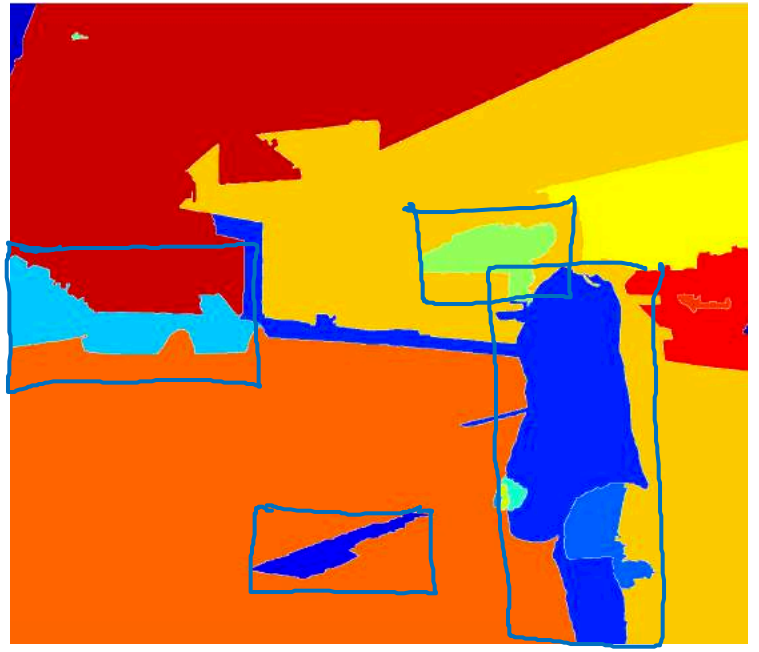
Object Detection

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]

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.

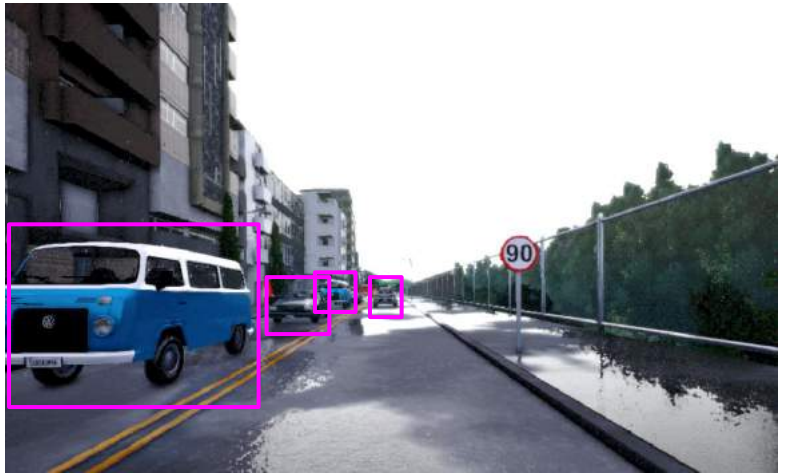
[Girshik et. al, 2013. Rich feature hierarchies for accurate object detection and semantic segmentation]
[Girshik, 2015. Fast R-CNN]
[Ren et. al, 2016. Faster R-CNN: Towards real-time object detection with region proposal networks]

Convolutional Neural Networks

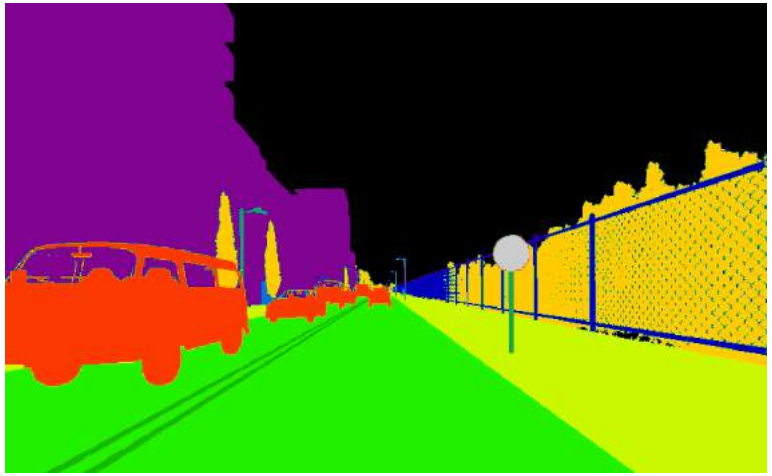
Semantic segmentation with U-Net

Object Detection vs. Semantic Segmentation

Input image



Object Detection



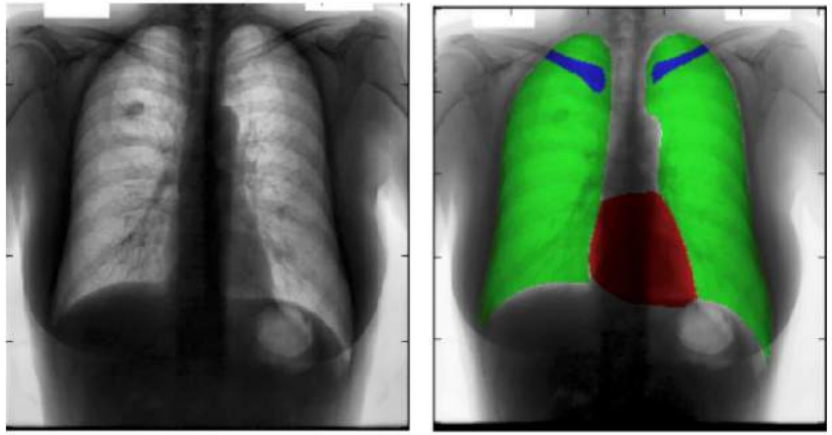
Semantic Segmentation

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

Andrew Ng

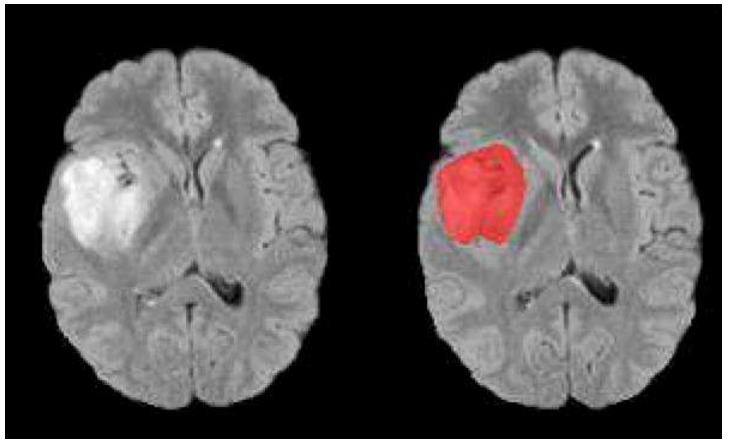
Motivation for U-Net

algorithm highlights different parts - heart, lungs, clavicle



Chest X-Ray

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



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]

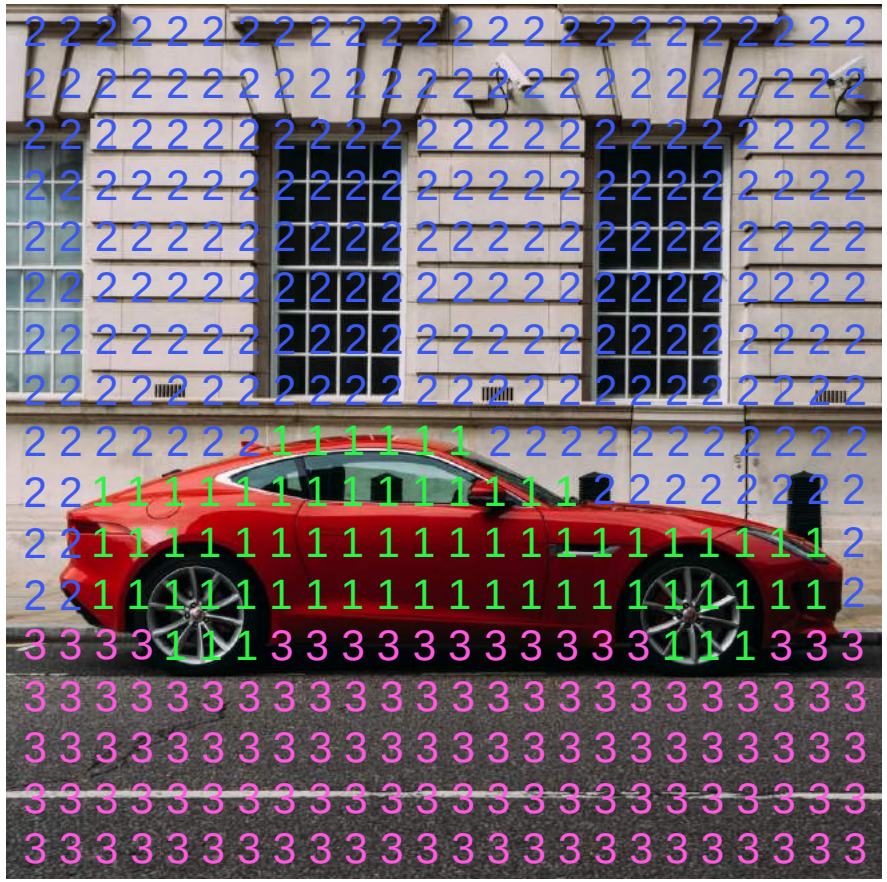
Per-pixel class labels

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

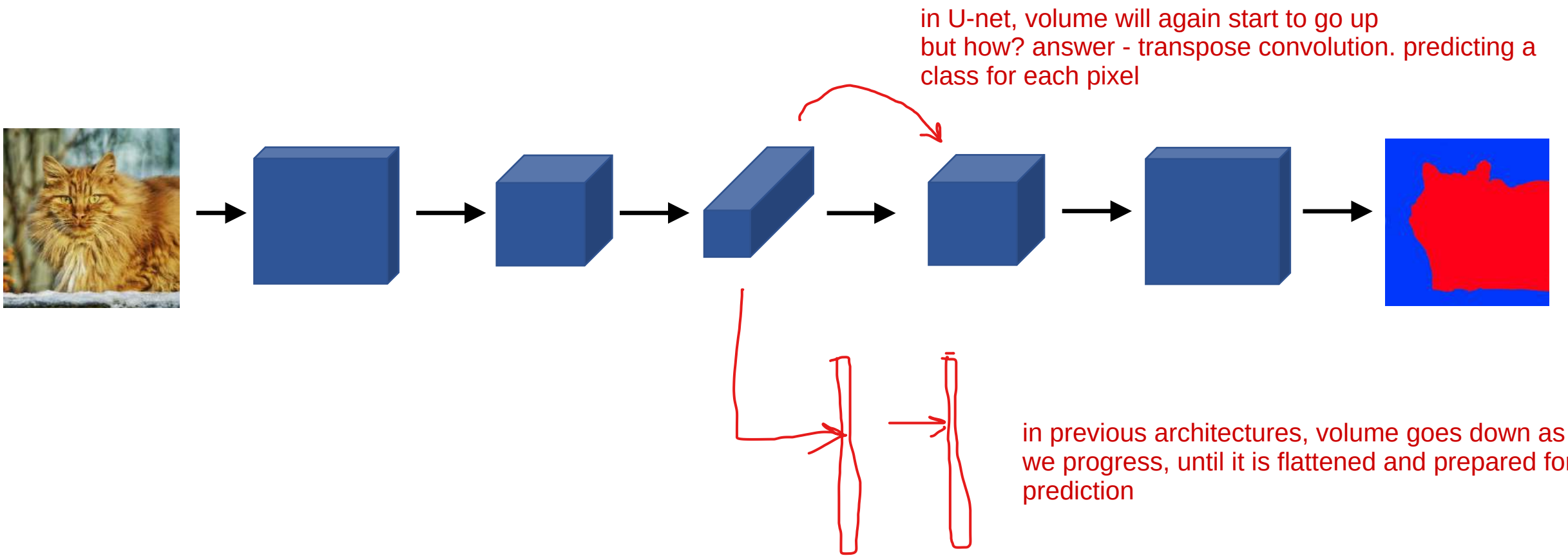


1. Car
2. Building
3. Road

Segmentation Map

Andrew Ng

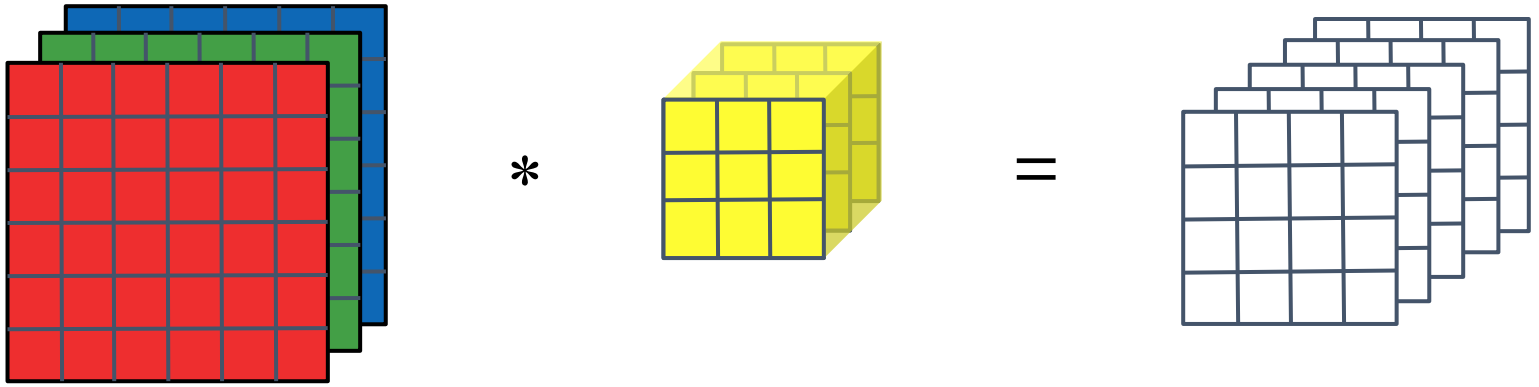
Deep Learning for Semantic Segmentation



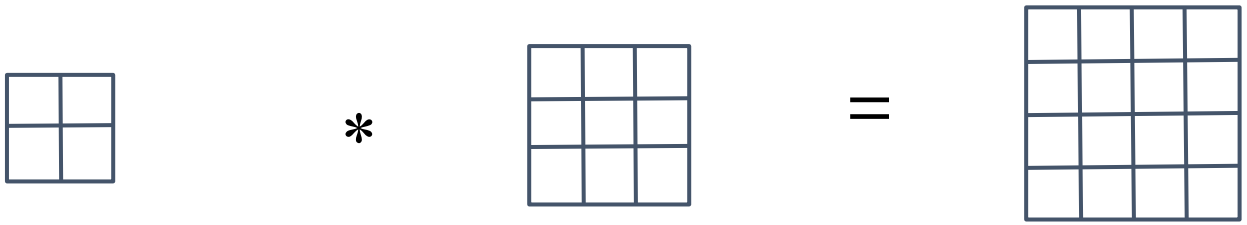
Andrew Ng

Transpose Convolution

Normal Convolution

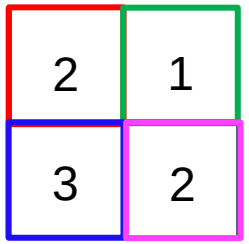


Transpose Convolution

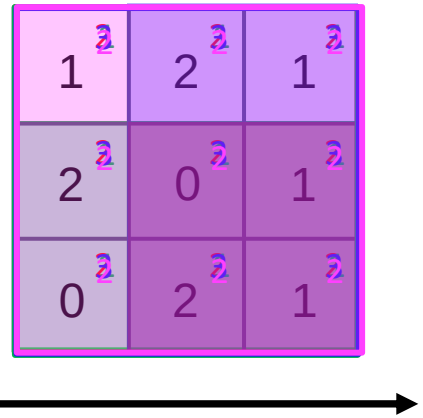


Transpose Convolution

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

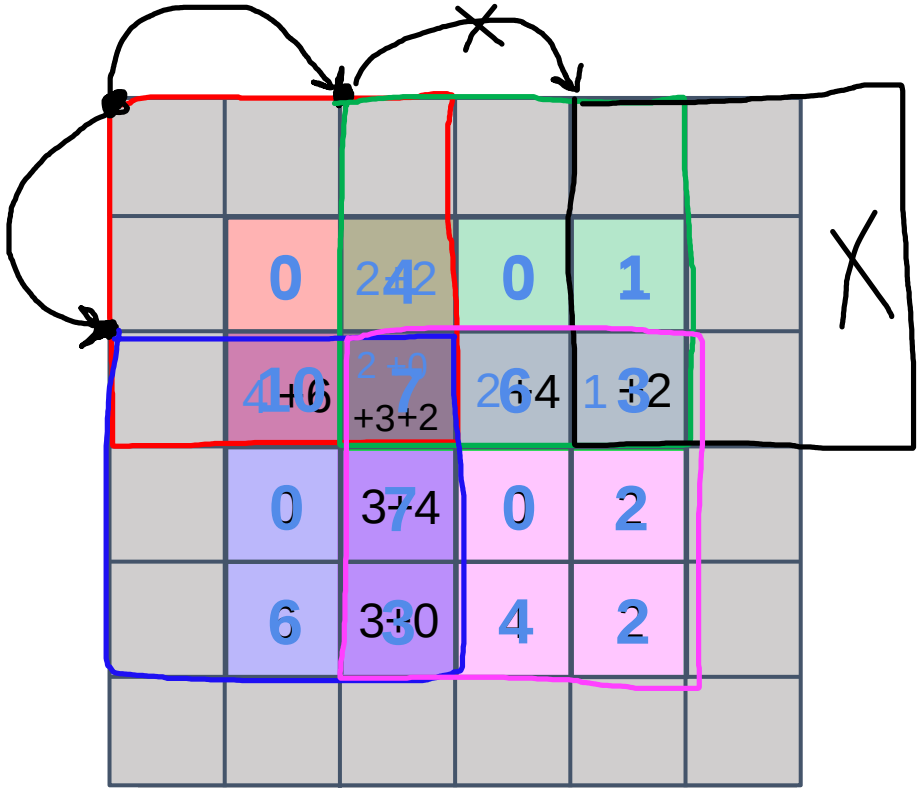


2 x 2



weight filter

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

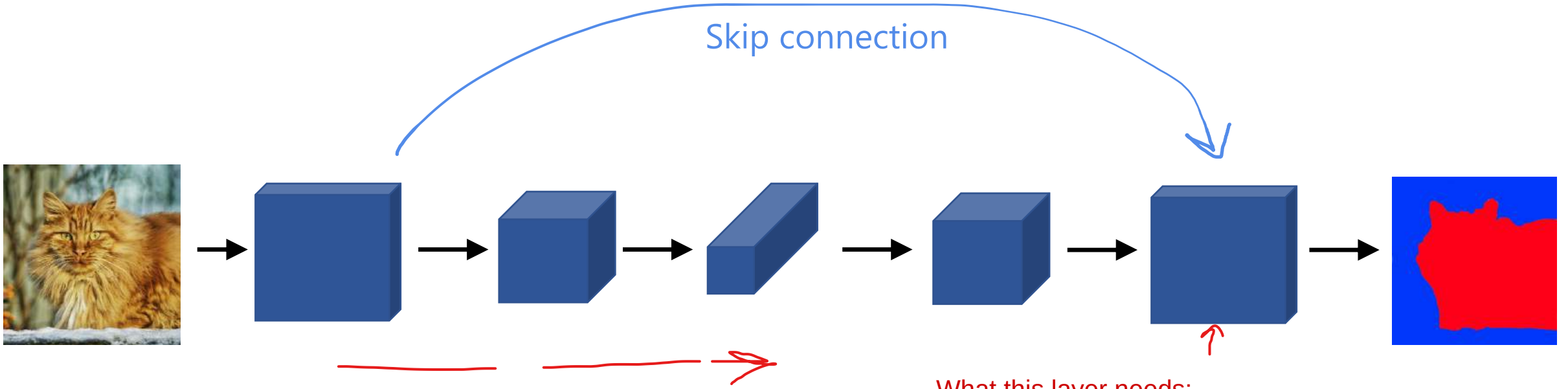


4 x 4

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

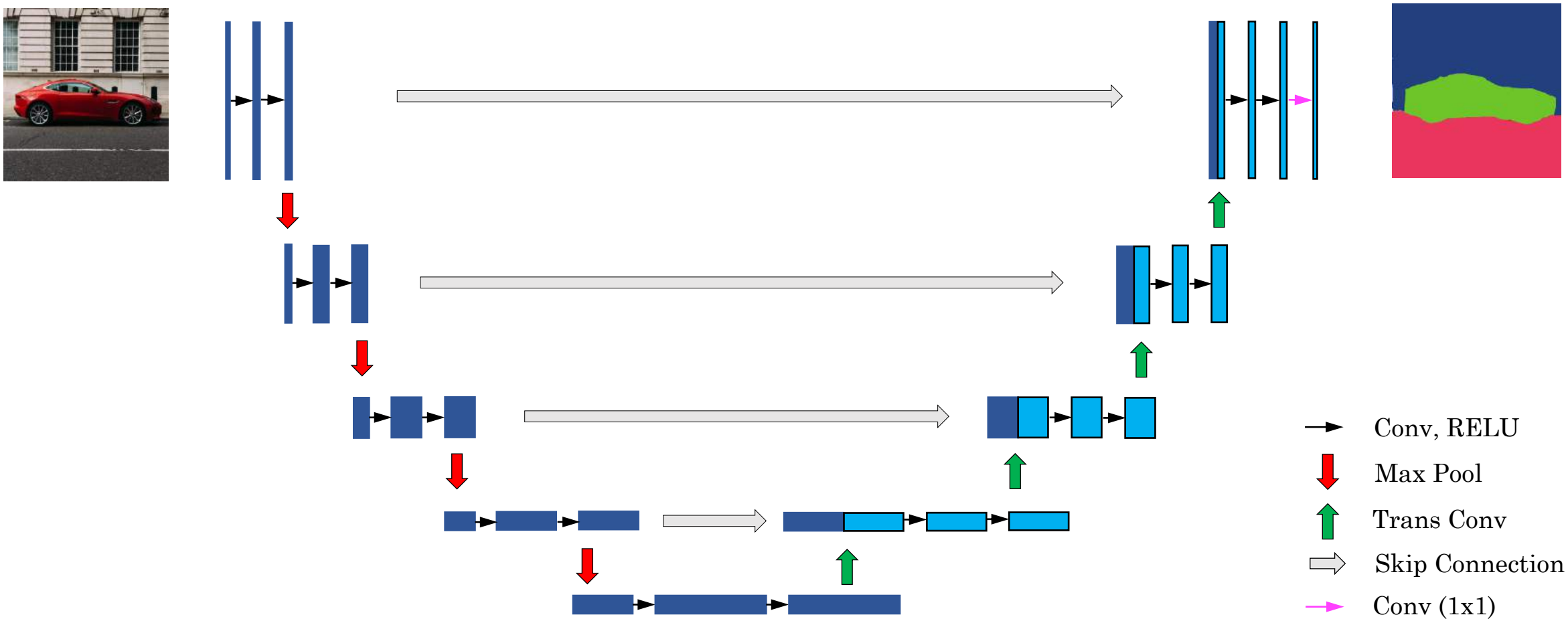
Andrew Ng

Deep Learning for Semantic Segmentation



Andrew Ng

U-Net



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

U-Net

h x w x 3

width of tiles is # of channels

1x1 convolution

h x w x # classes

increased h/w

increased h/w

we didn't increase h/w but reduced channels

Conv, RELU

Max Pool

Trans Conv

Skip Connection

Conv (1x1)

[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
- 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")

1:1 99% 99.9

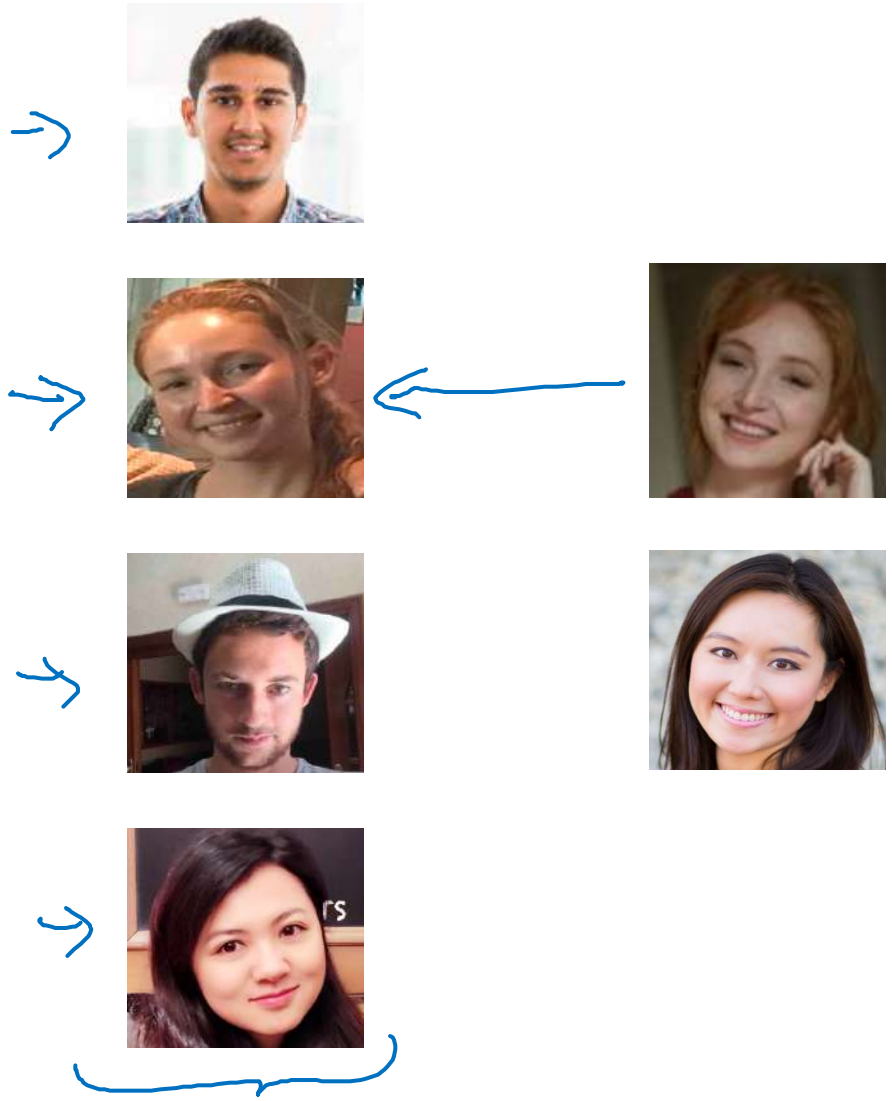
1:K K=100

deeplearning.ai

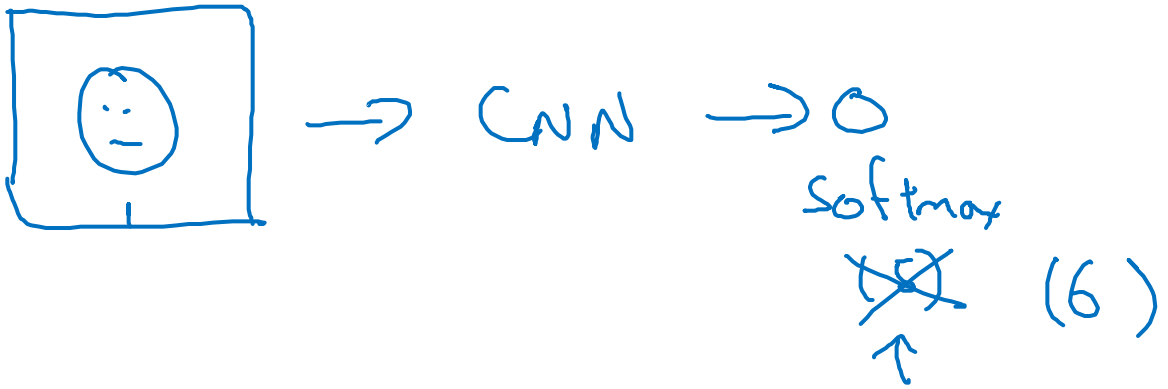
Face recognition

One-shot learning

One-shot learning



Learning from one example to recognize the person again



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

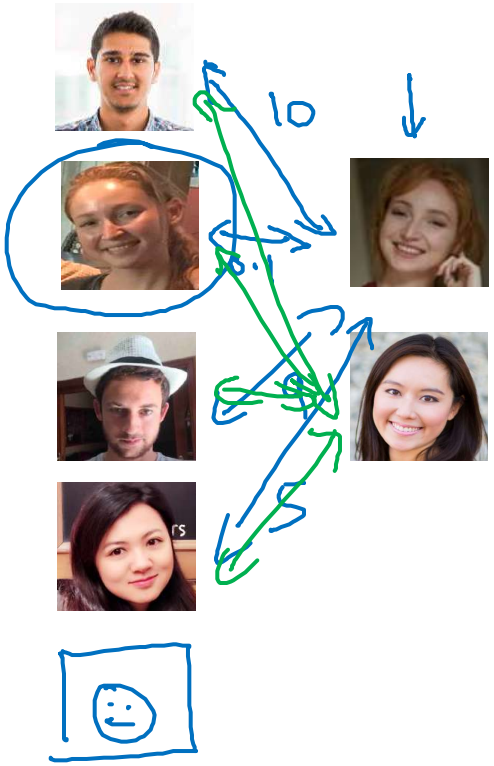
Learning a “similarity” function

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

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

$> \tau$ "different"

} Verification.



d(img1, img2)

Key idea - learn a "similarity function"

How to train a NN to learn this function "d" - use a Siamese network

Training images for Siamese network can be a different set of images (with >1 for each person)

Andrew Ng

deeplearning.ai

Face recognition

Siamese network

Siamese network

$x^{(1)}$

"encoding of $x^{(1)}$"

128

$f(x^{(1)})$

128

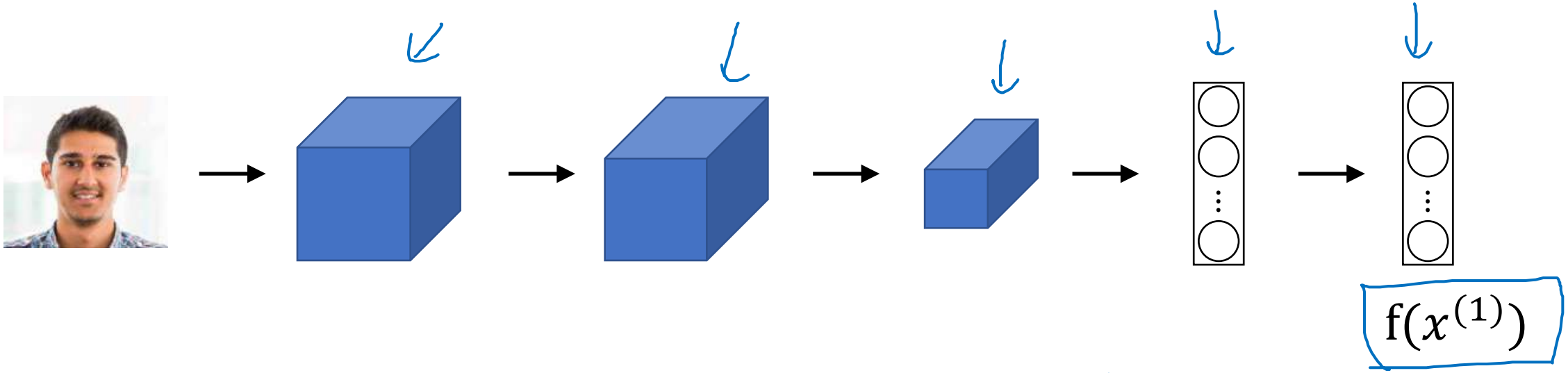
$x^{(2)}$

$f(x^{(2)})$

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

[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.

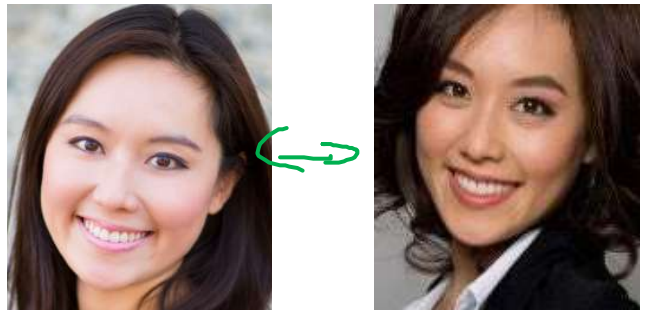
deeplearning.ai

Face recognition

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

Anchor

Negative

$d(A,P) = 0.5$

$d(A,N) = 0.7$

Want: $\|f(A) - f(P)\|^2 + \alpha \leq \|f(A) - f(N)\|^2$

$\alpha = 0.2$

$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha \leq 0$

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

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

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:

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

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

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.

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

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

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

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

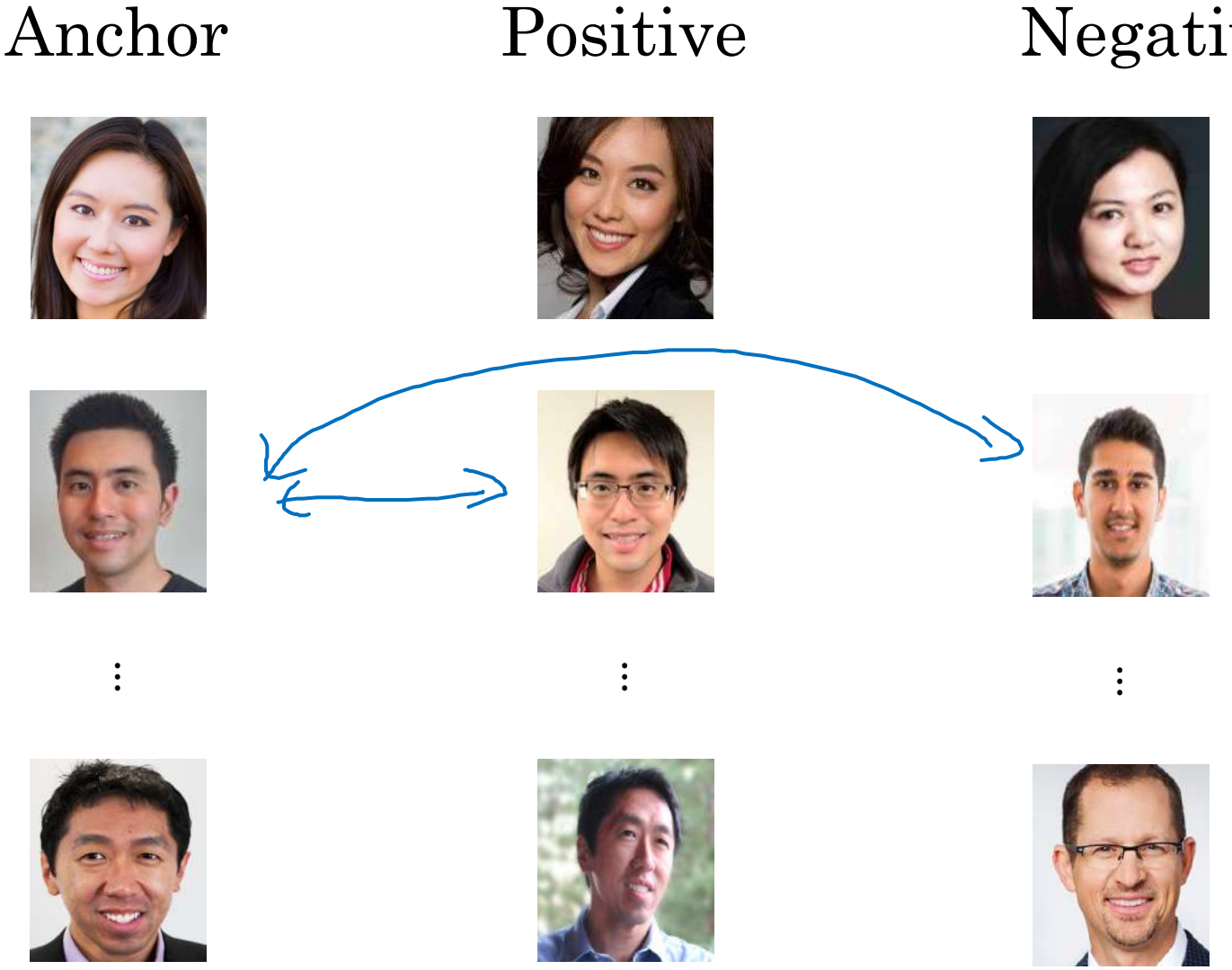
↓ ↑

Face Net

Deep Face

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

Training set using triplet loss



J

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

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

$x^{(i)}$

$x^{(j)}$

$f(x^{(i)})$

$f(x^{(j)})$

$\hat{y}$

y

Precompute

new

Database x

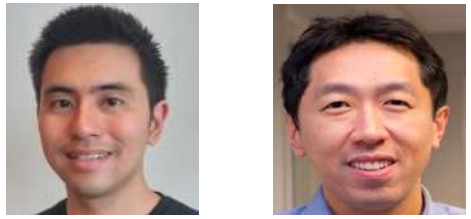
$$\hat{y} = \sigma\left(\sum_{k=1}^{128} w_k \left| f(x^{(i)})_k - f(x^{(j)})_k \right| + b\right)$$

$$\frac{\left(f(x^{(i)})_k - f(x^{(j)})_k\right)^2}{f(x^{(i)})_k + f(x^{(j)})_k}$$

χ^2

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

Face verification supervised learning

x	y	
	1	"Same"
	0	"Albert"
	0	
	1	

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

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)

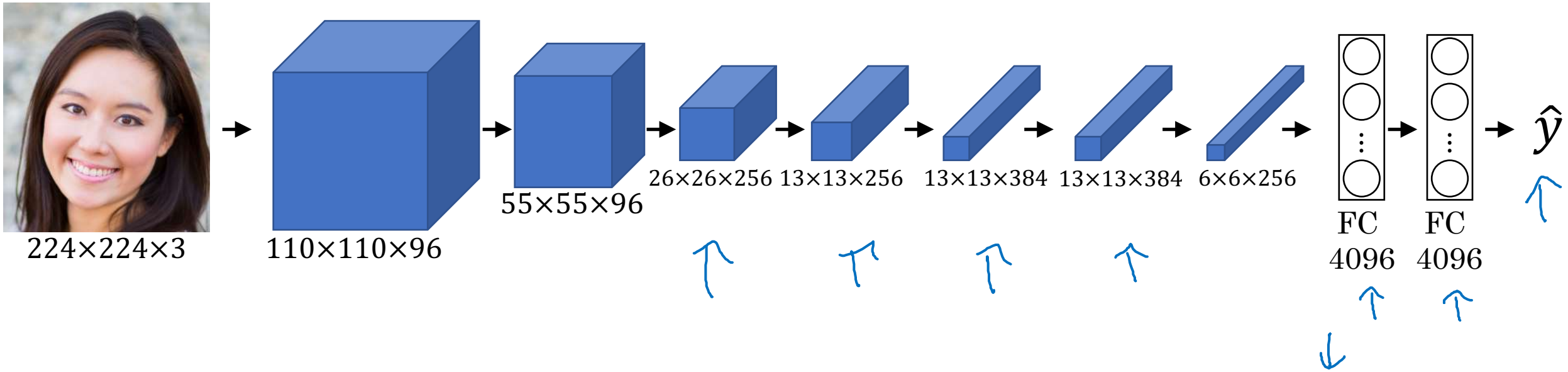
[Images generated by Justin Johnson]

deeplearning.ai

Neural Style Transfer

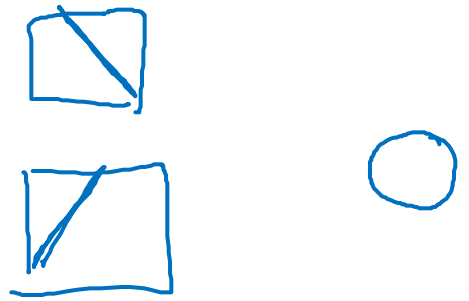
What are deep ConvNets learning?

Visualizing what a deep network is learning



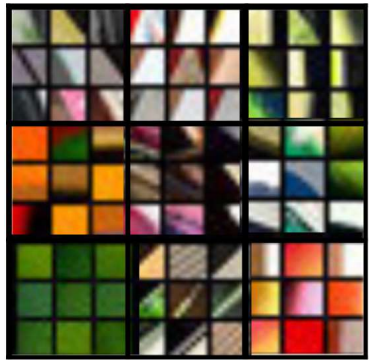
Pick a unit in layer 1. Find the nine image patches that maximize the unit's activation.

Repeat for other units.

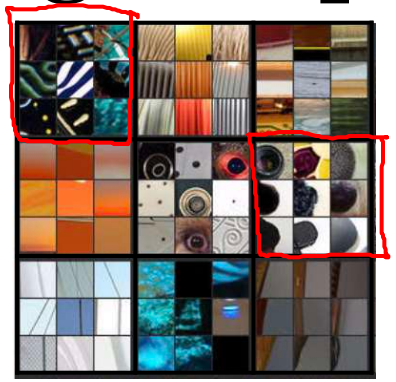


[Zeiler and Fergus., 2013, Visualizing and understanding convolutional networks]

Visualizing deep layers

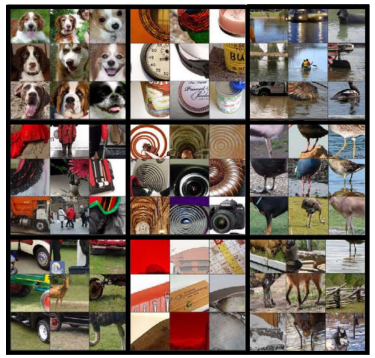


Layer 1

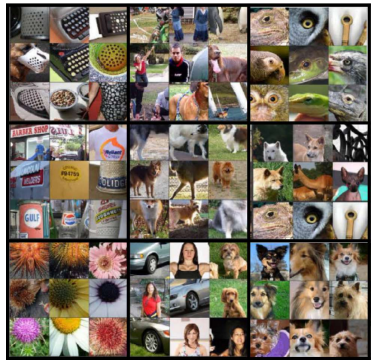


Layer 2

Layer 3

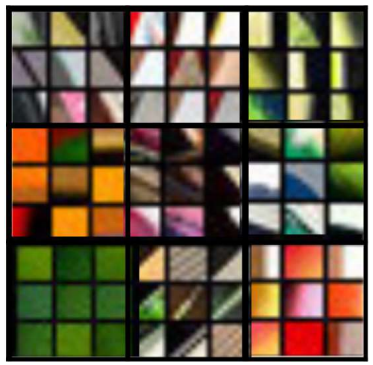


Layer 4



Layer 5

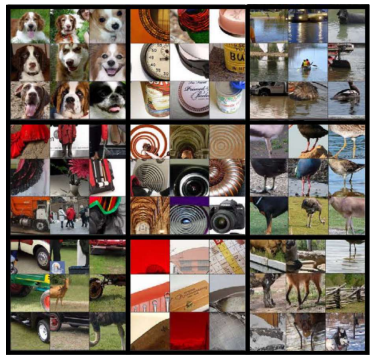
Visualizing deep layers: Layer 1



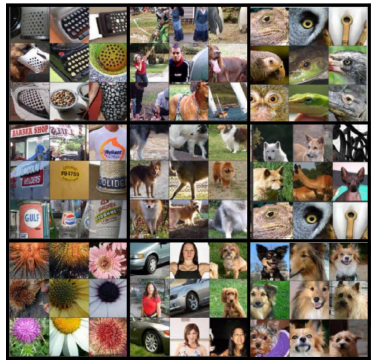
Layer 1

Layer 2

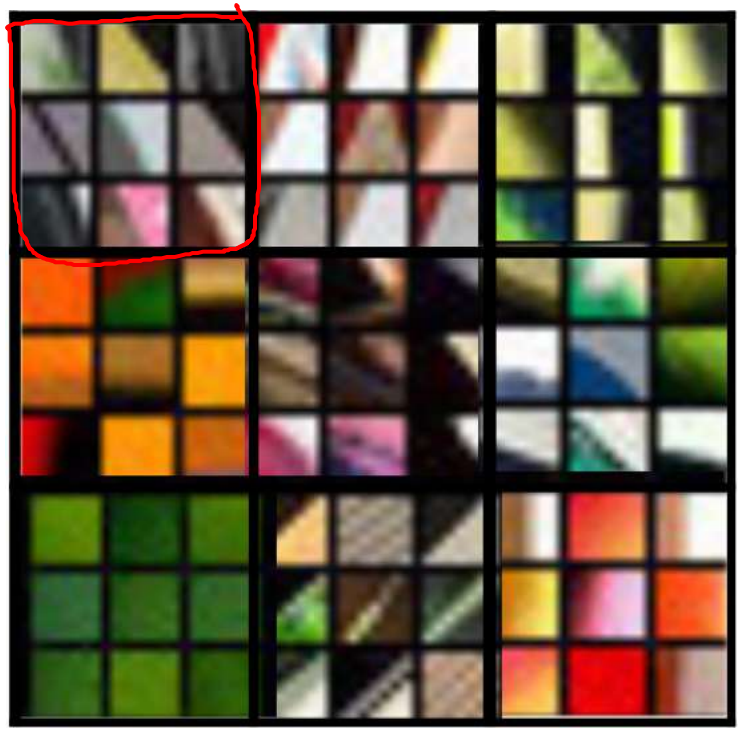
Layer 3



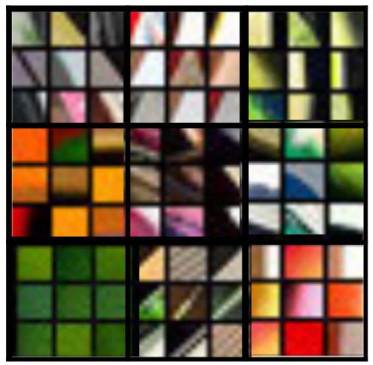
Layer 4



Layer 5



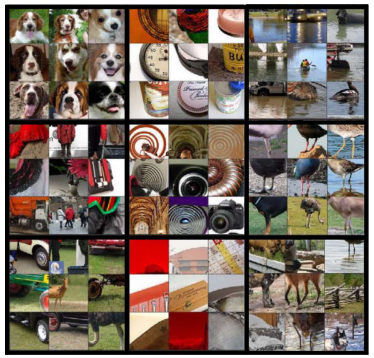
Visualizing deep layers: Layer 2



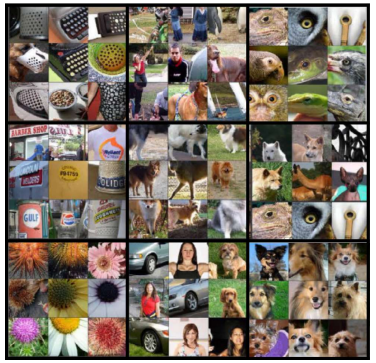
Layer 1

Layer 2

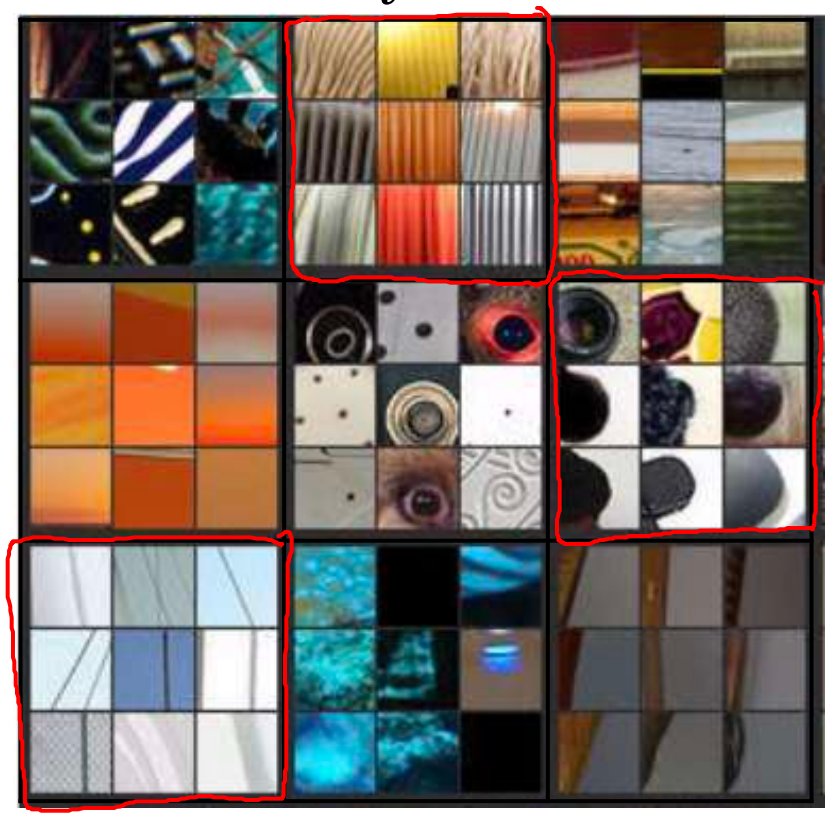
Layer 3



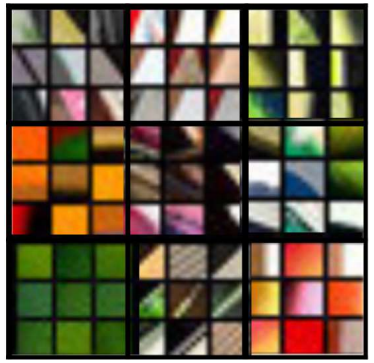
Layer 4



Layer 5



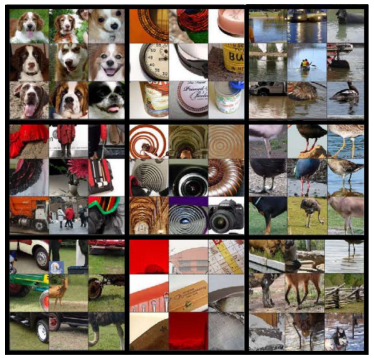
Visualizing deep layers: Layer 3



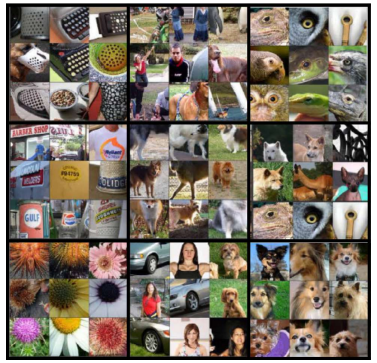
Layer 1

Layer 2

Layer 3

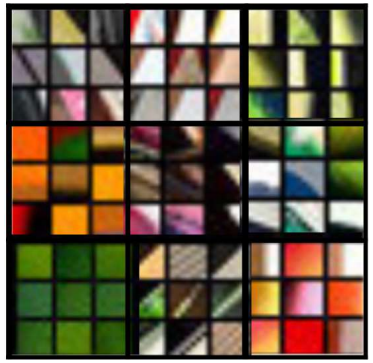


Layer 4

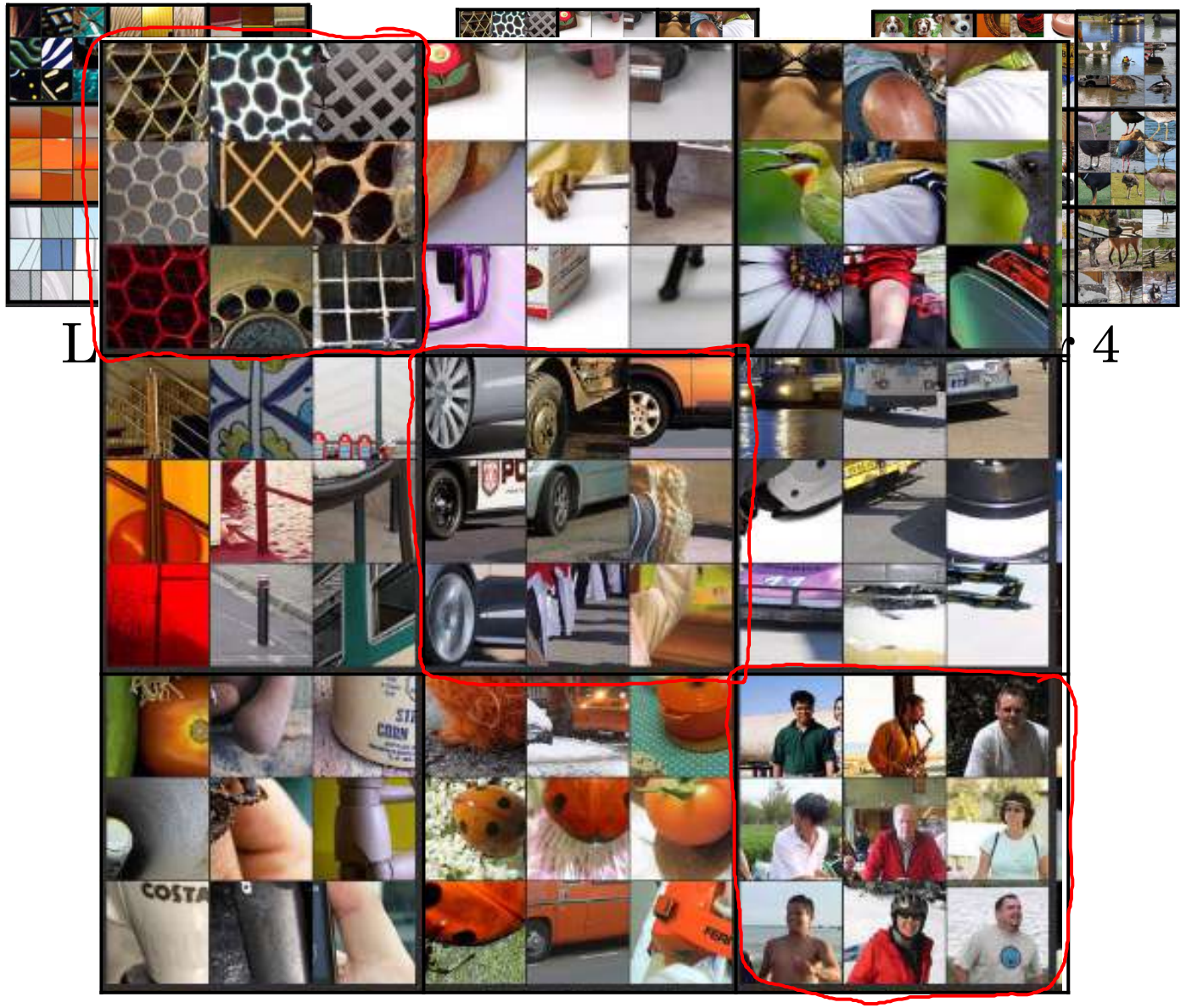


Layer 5

Visualizing deep layers: Layer 3

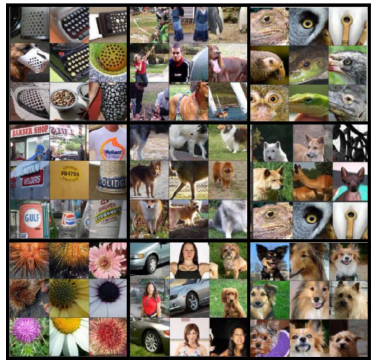


Layer 1



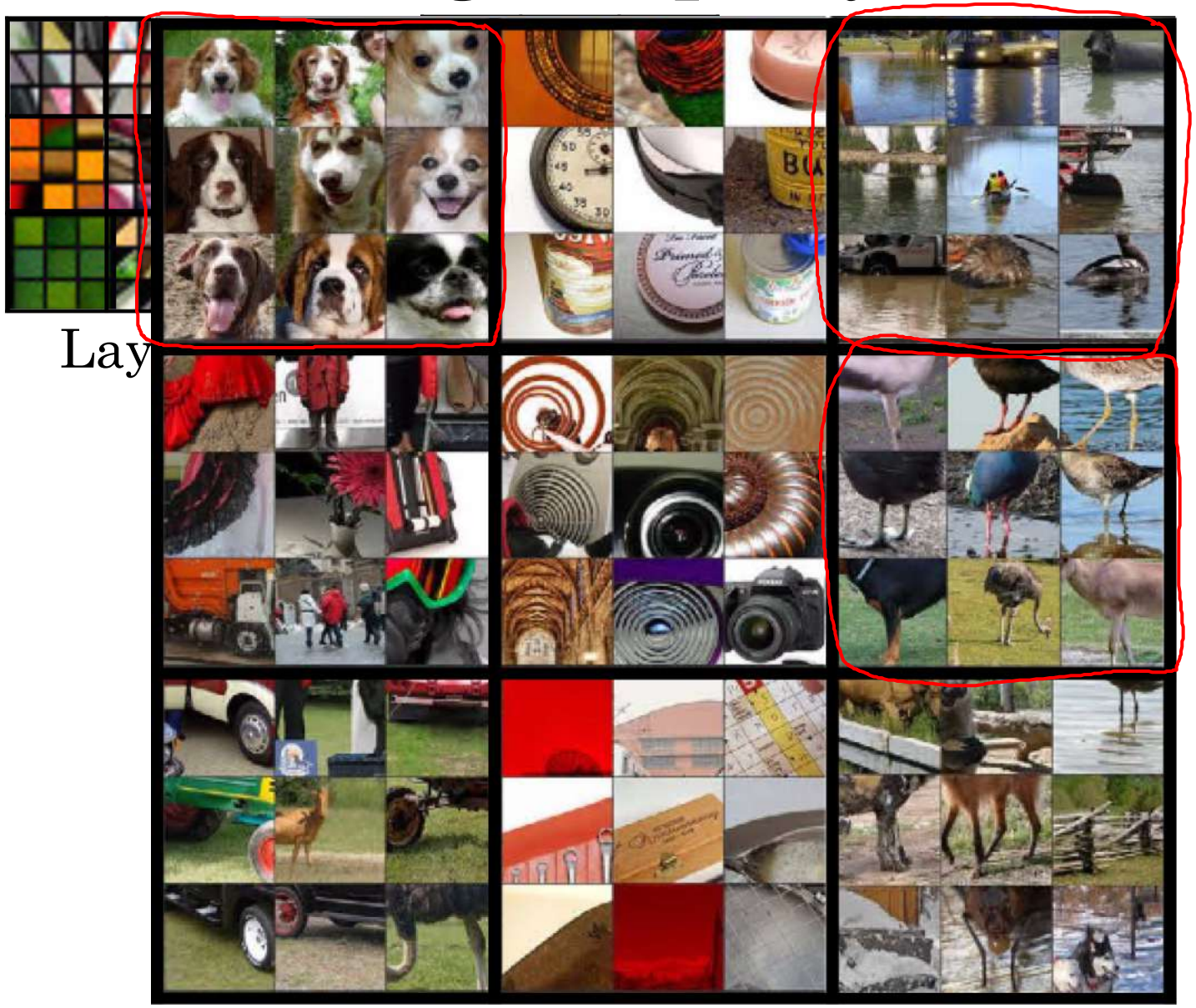
L

4

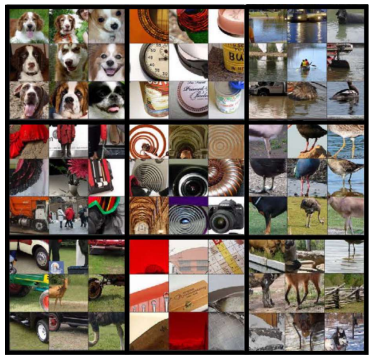


Layer 5

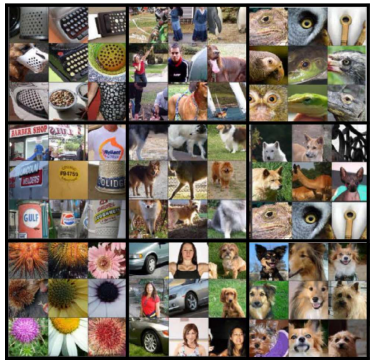
Visualizing deep layers: Layer 4



Lay

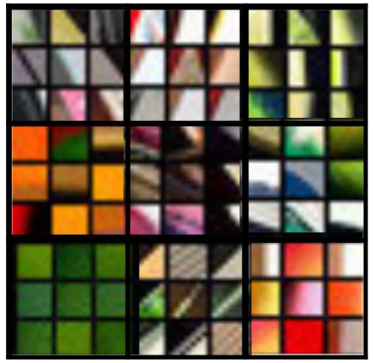


Layer 4

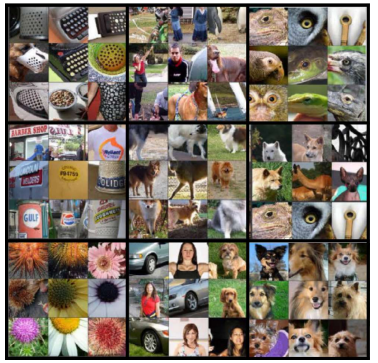


Layer 5

Visualizing deep layers: Layer 5



Layer 1



Layer 5

Andrew Ng

deeplearning.ai

Neural Style Transfer

Cost function

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

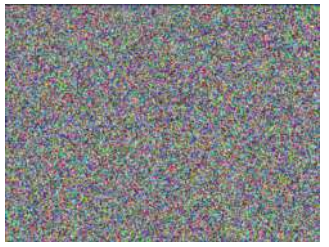
   start with white-noise image

   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]

deeplearning.ai

Neural Style Transfer

Content cost function

Content cost function

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

'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

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

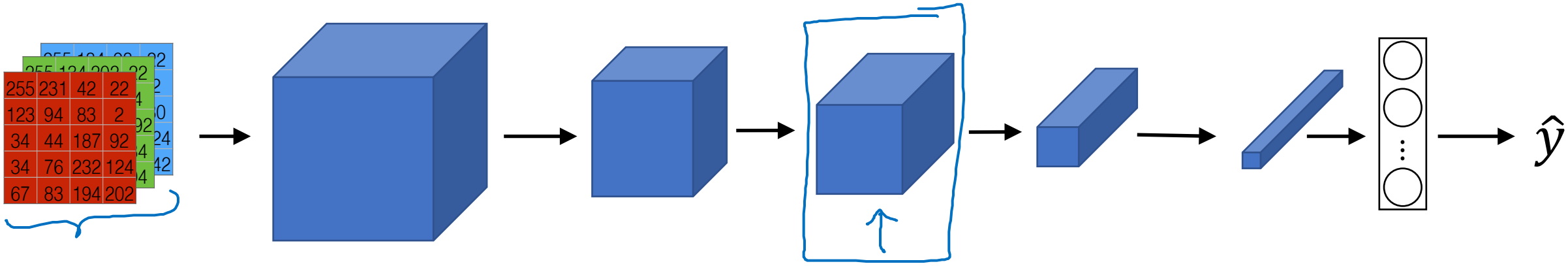
[Gatys et al., 2015. A neural algorithm of artistic style]

deeplearning.ai

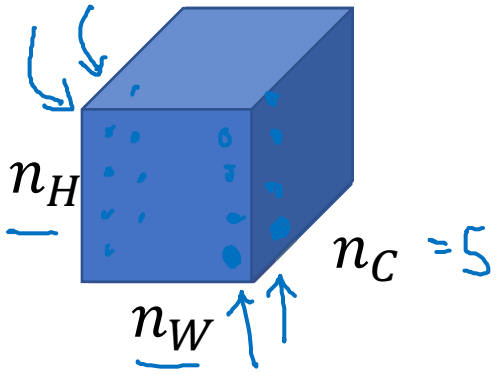
Neural Style Transfer

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.

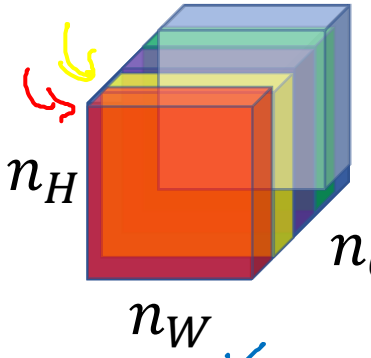


How correlated are the activations across different channels?

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

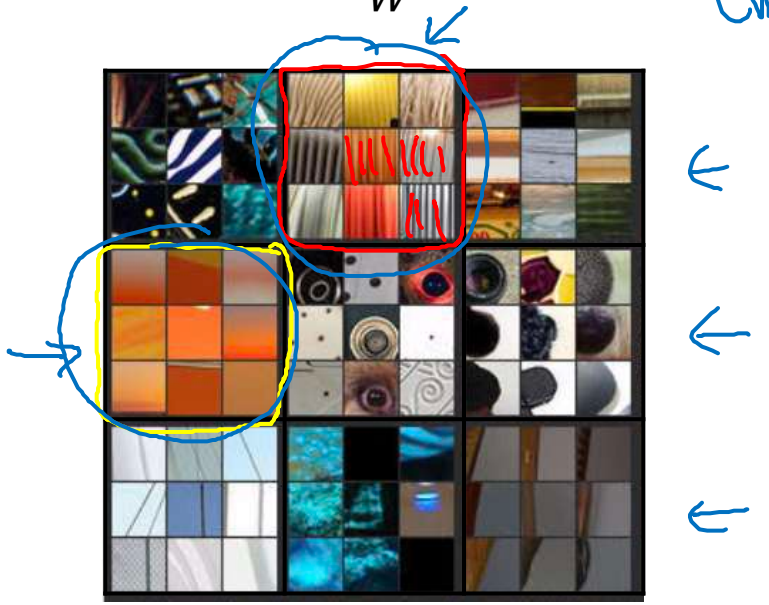
Intuition about style of an image

Style image

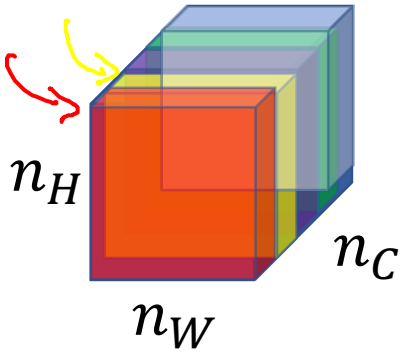


Correlated?

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

$n_c^{[l]}$

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

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

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

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

"Gram matrix"

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

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

β

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

Style cost function

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

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

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

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

G

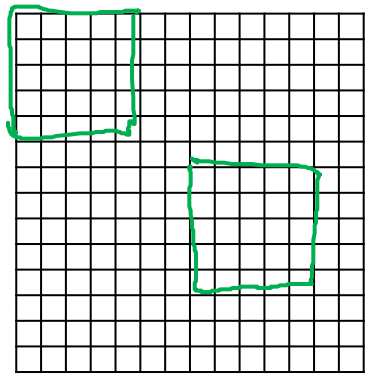
[Gatys et al., 2015. A neural algorithm of artistic style]

deeplearning.ai

Convolutional Networks in 1D or 3D

1D and 3D generalizations of models

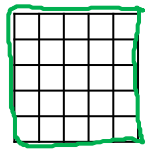
Convolutions in 2D and 1D



2D input image

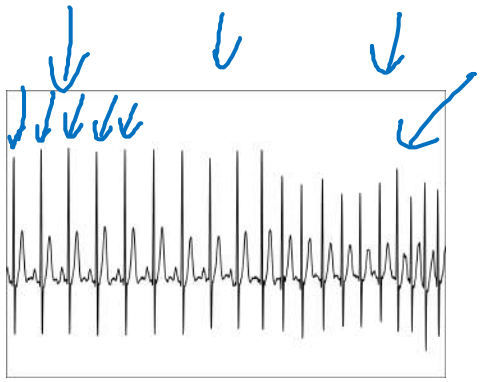
14×14

*

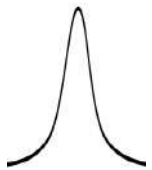


2D filter

5×5



*

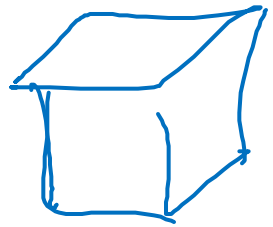


1	20	15	3	18	12	4	17

14

1	3	10	3	1

5



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

$\rightarrow 10 \times 10 \times 16$

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

$\rightarrow 6 \times 6 \times 32$

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

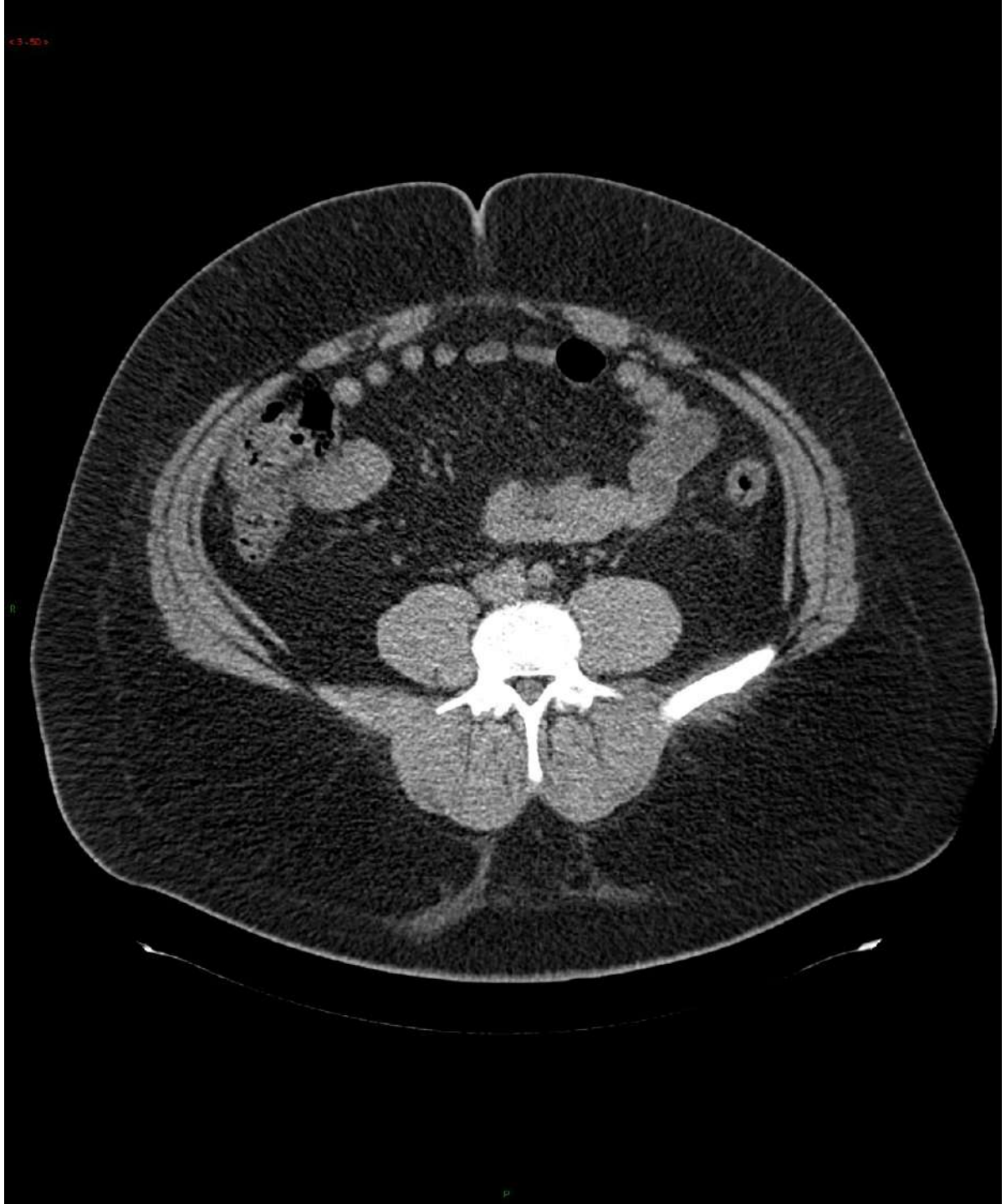
$\rightarrow 10 \times 16$

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

$\rightarrow 6 \times 32$

Andrew Ng

3D data

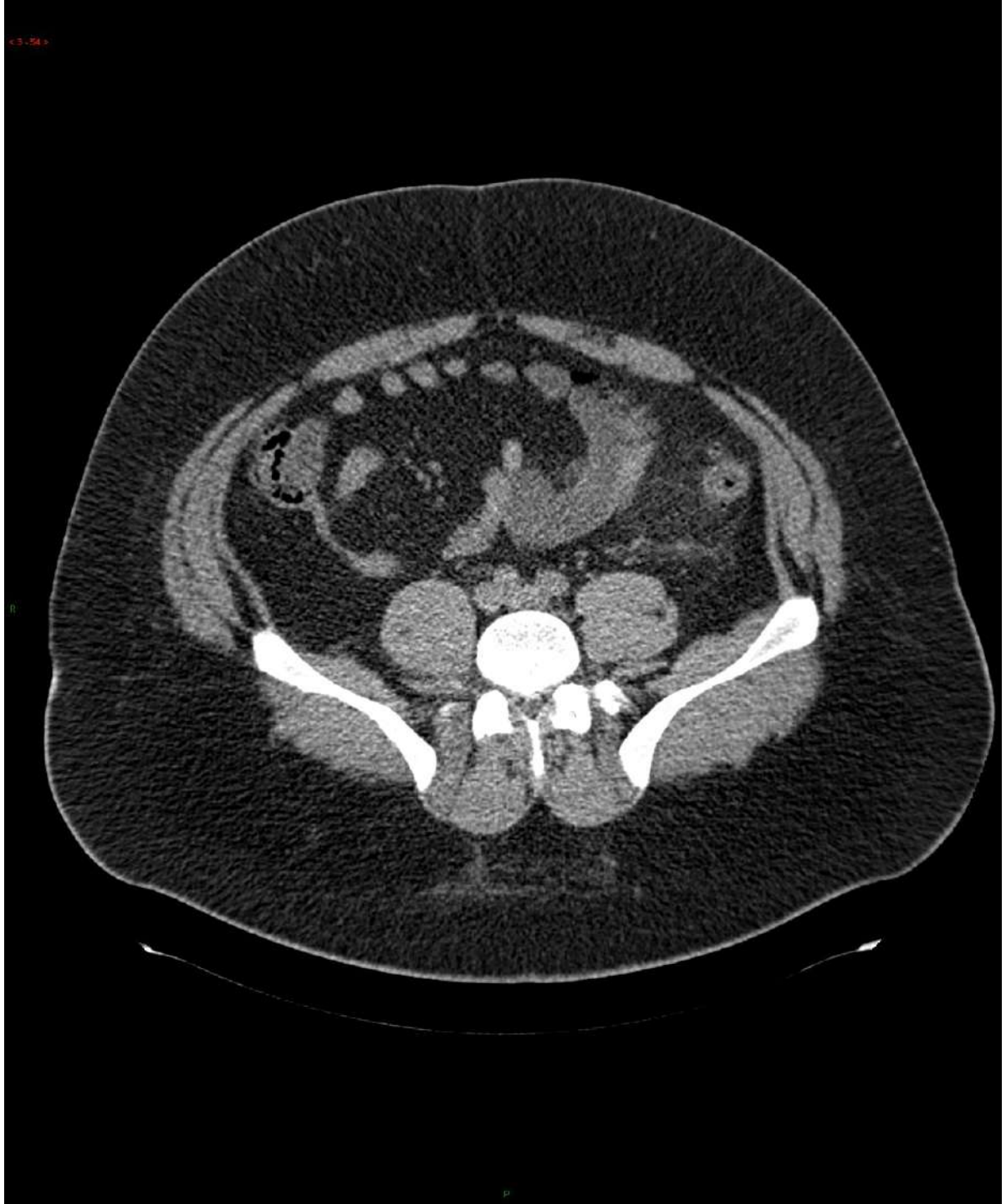


Andrew Ng

3D data

Andrew Ng

3D data

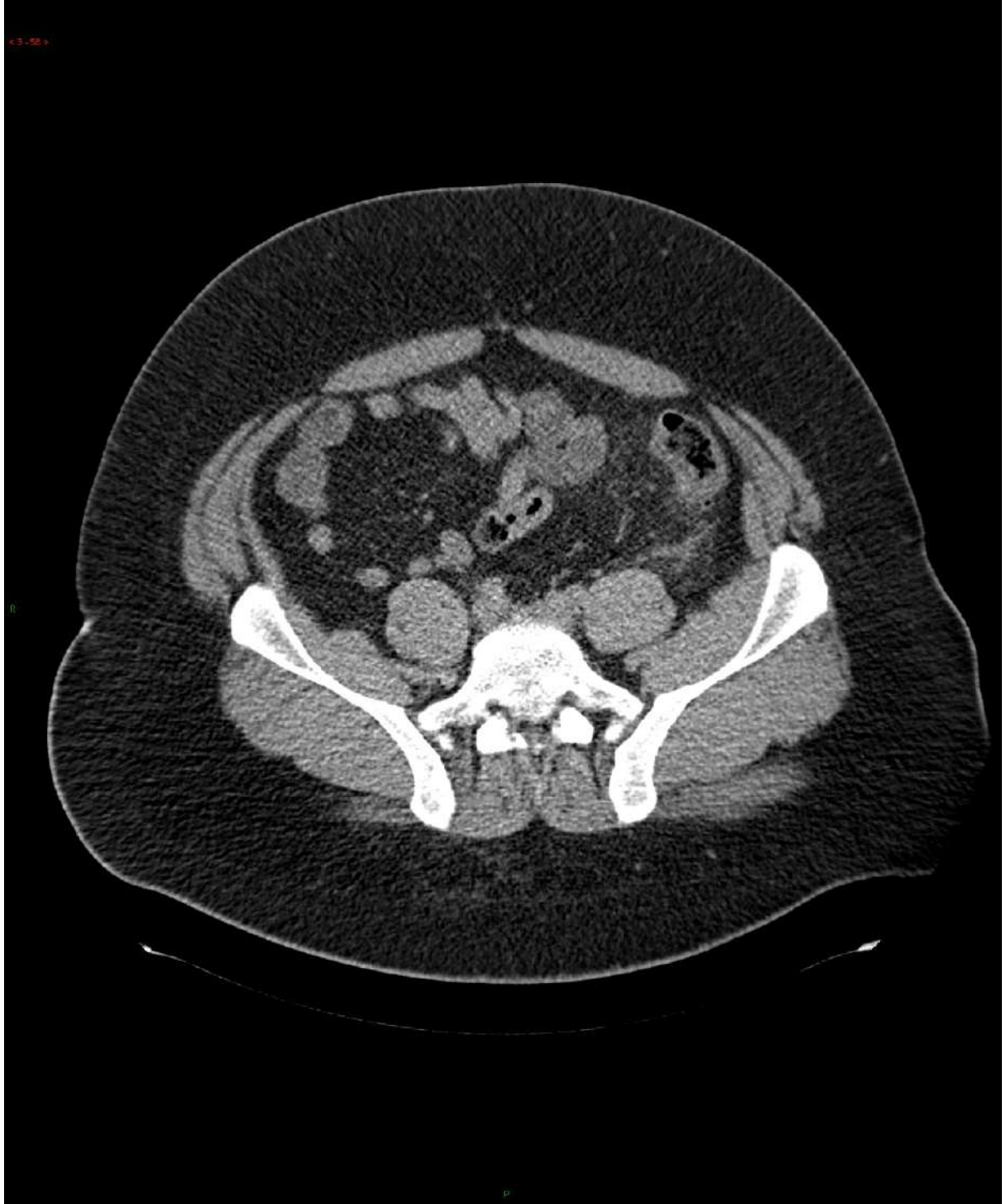


Andrew Ng

3D data

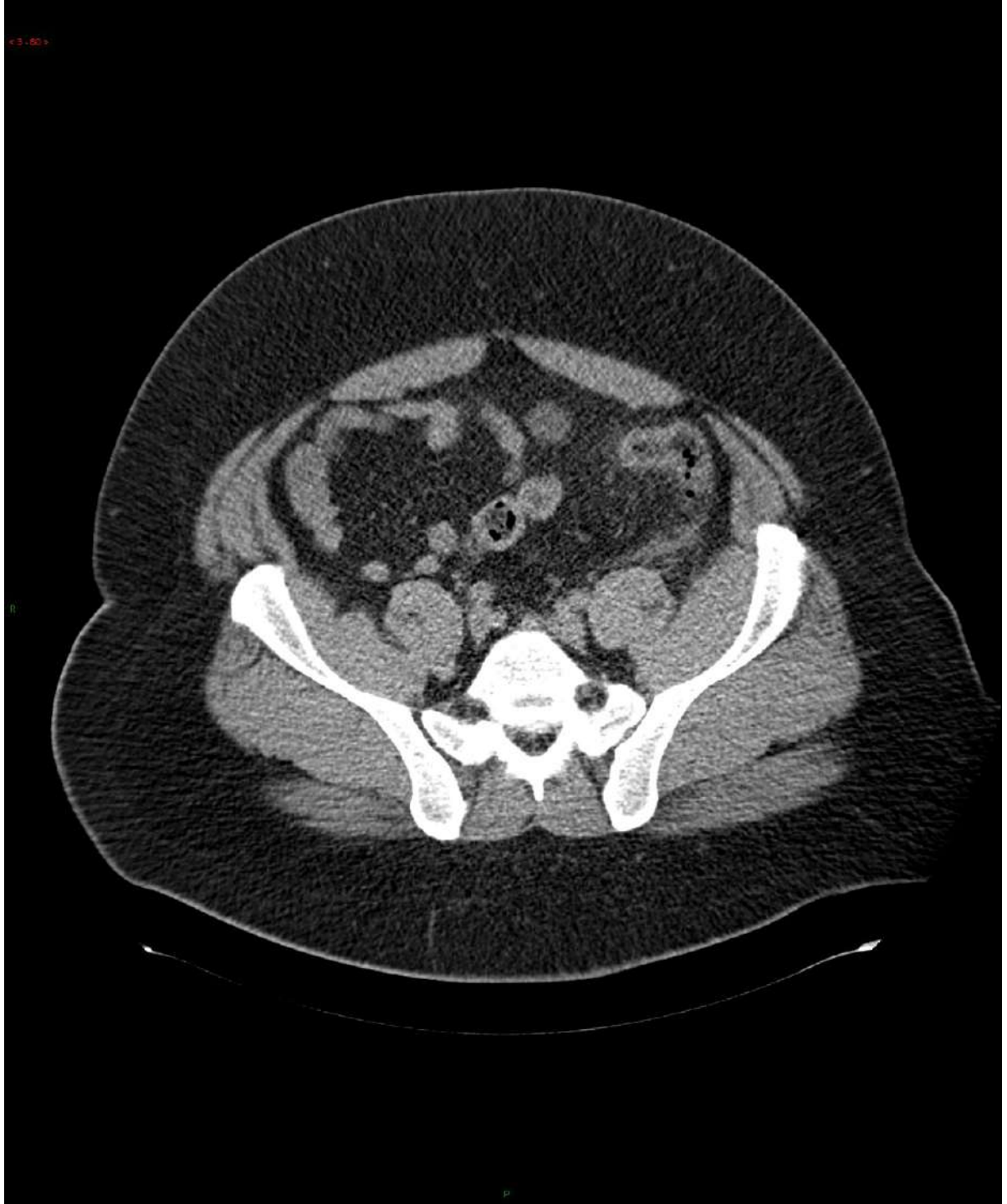
Andrew Ng

3D data



Andrew Ng

3D data



Andrew Ng

3D data

Andrew Ng

3D data

Andrew Ng

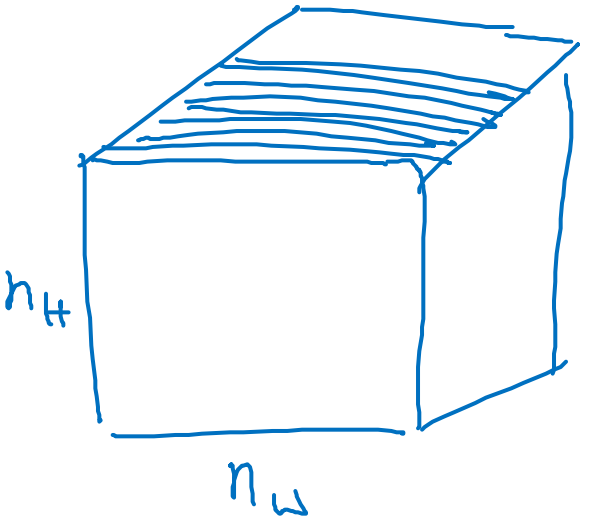
3D data

Andrew Ng

3D data

Andrew Ng

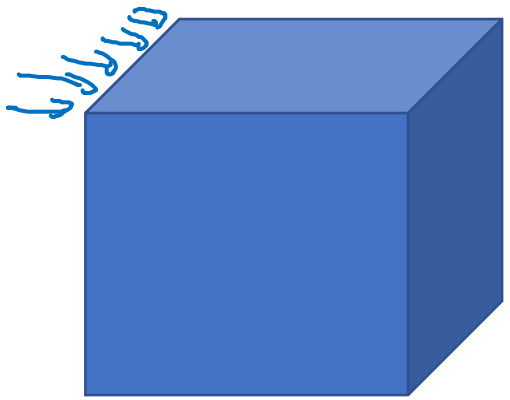
3D data



Andrew Ng

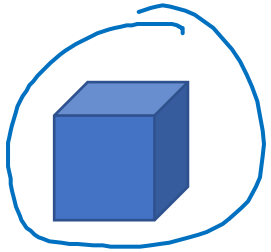
3D convolution

of channels can be considered 4th dimension



3D volume

*



3D filter

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

$14 \times 14 \times 14 \times 1$ ($n_c = 1$)

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

16 filters.

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

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

32 filters

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