



deeplearning.ai

Case Studies

Why look at
case studies?

Outline

Classic networks:

- LeNet-5 ←
- AlexNet ←
- VGG ←

ResNet (152)

Inception

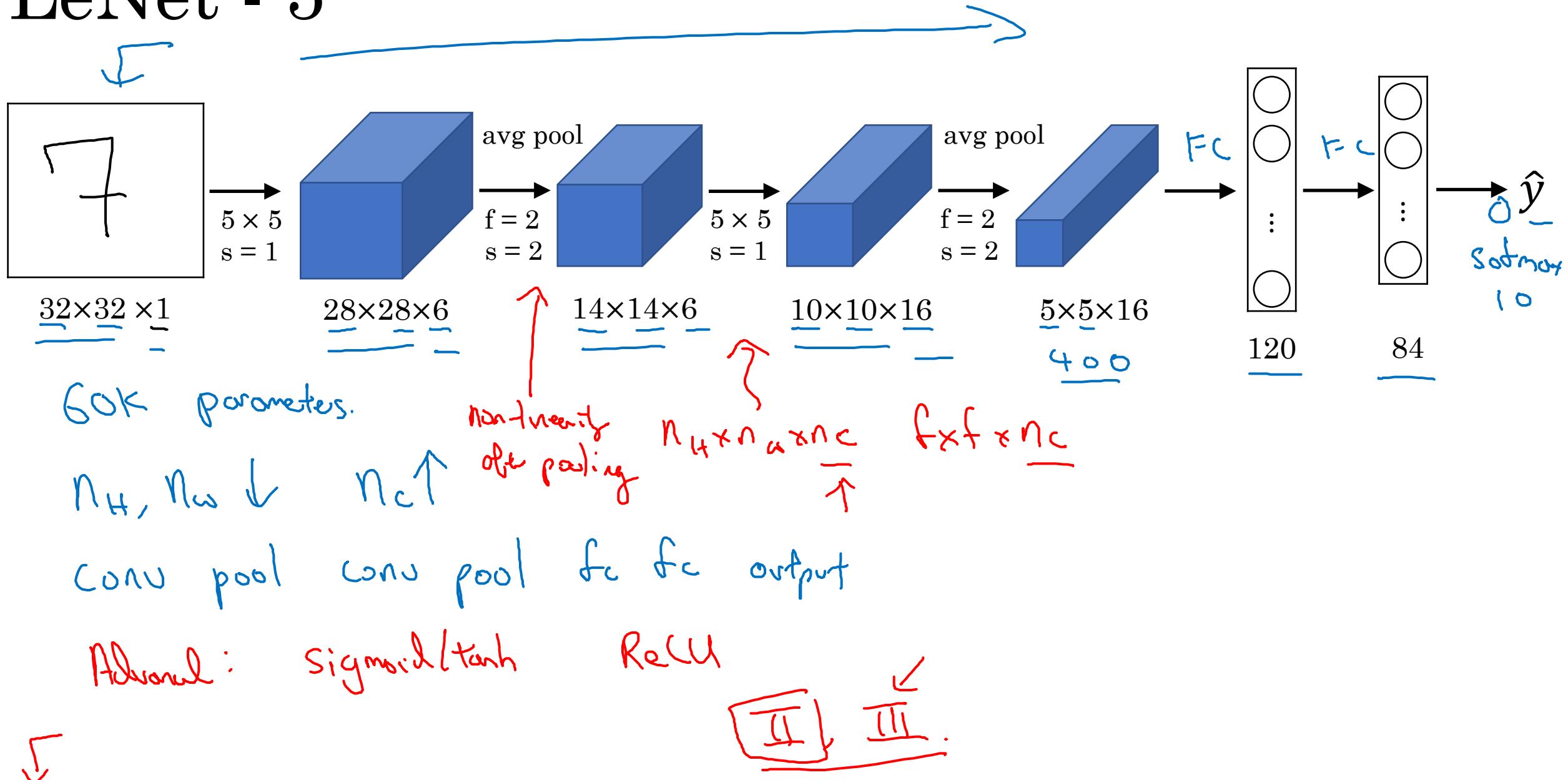


deeplearning.ai

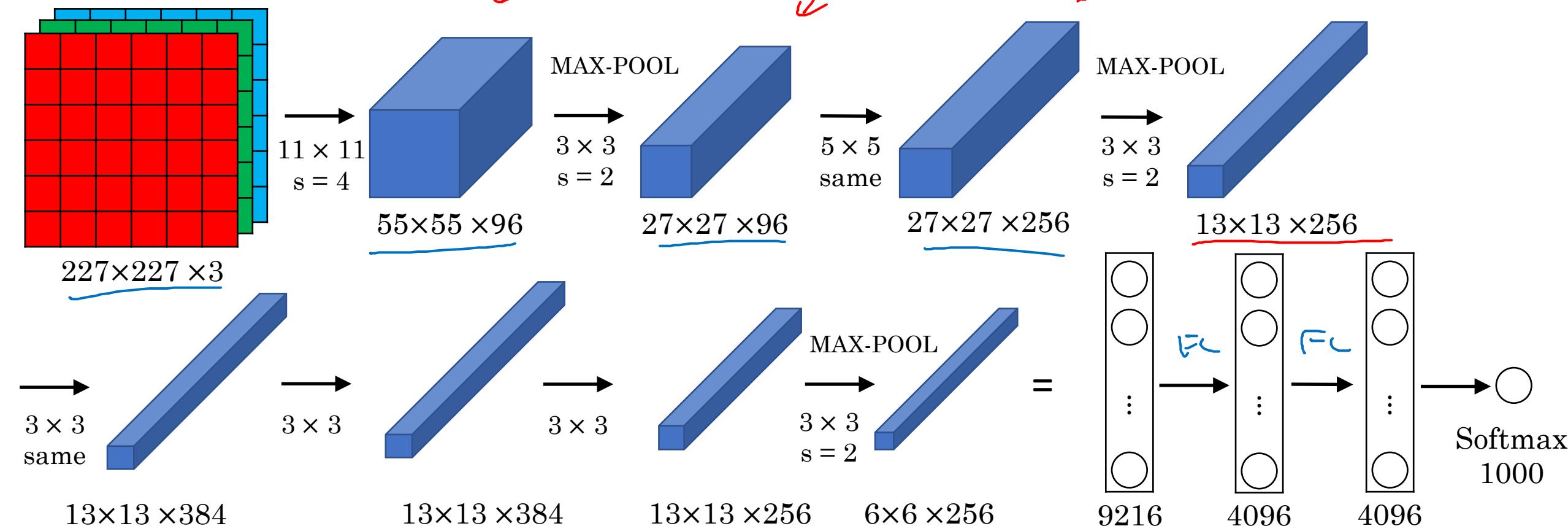
Case Studies

Classic networks

LeNet - 5



AlexNet

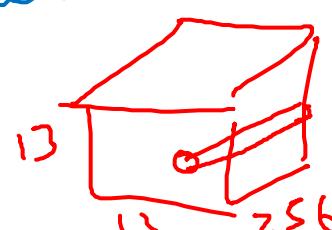


- Similar to LeNet, but much bigger.

- ReLU

- Multiple GPUs.

- Local Response Normalization (LRN)

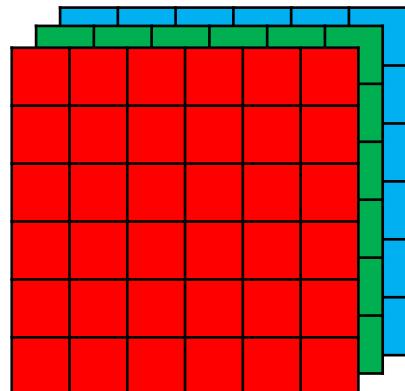


~60M parameters

Andrew Ng

VGG - 16

CONV = 3×3 filter, $s = 1$, same



224×224

[CONV 64]
 $\times 2$

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

$112 \times 112 \times 64$

[CONV 128]
 $\times 2$ → $112 \times 112 \times 128$

POOL → $56 \times 56 \times 128$

224×224

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

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

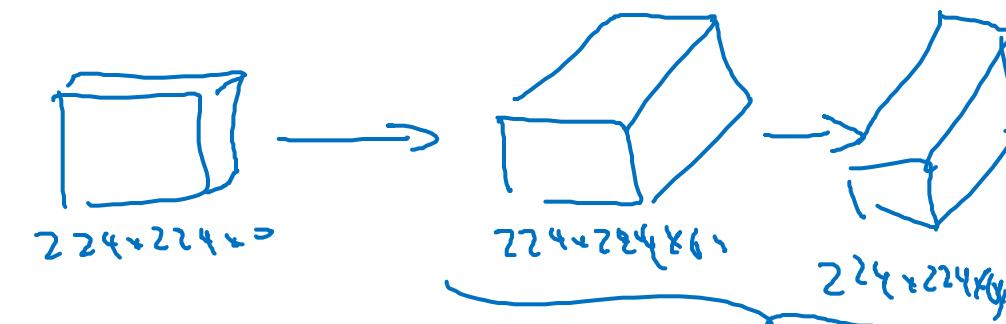
$n_H, n_W \downarrow$

$n_C \uparrow$

$\sim 38M$

VGG-19

MAX-POOL = 2×2 , $s = 2$



$224 \times 224 \times 64$ → POOL → $112 \times 112 \times 64$ → POOL → $56 \times 56 \times 128$

POOL

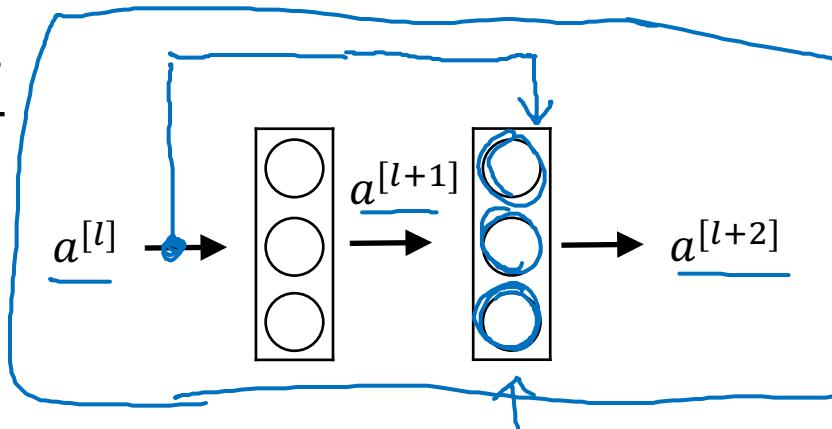


deeplearning.ai

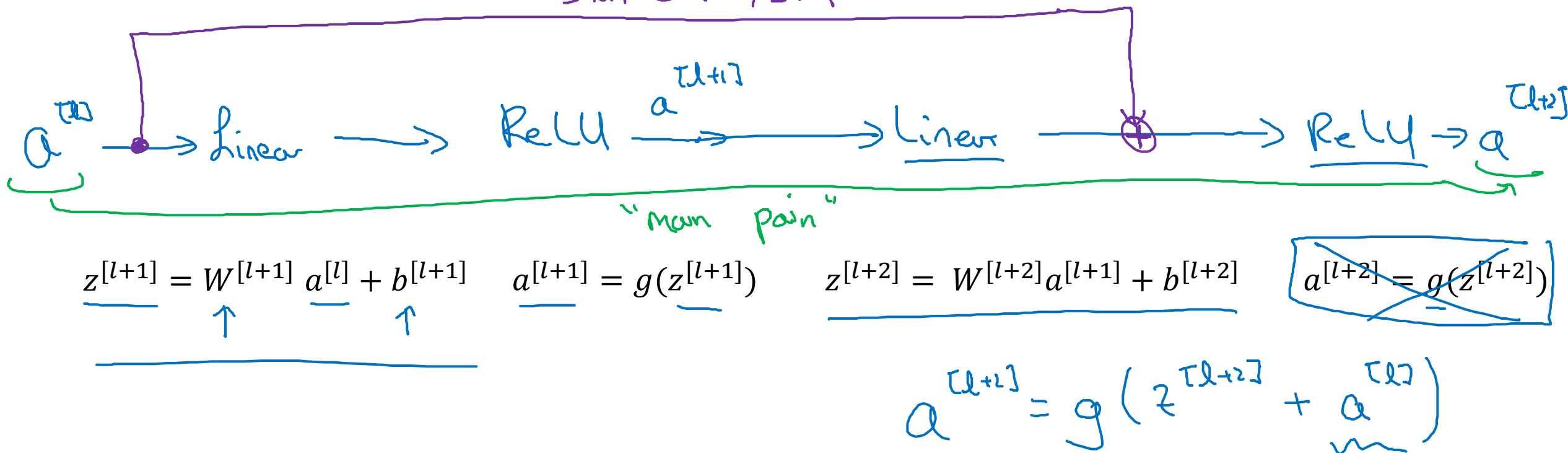
Case Studies

Residual Networks (ResNets)

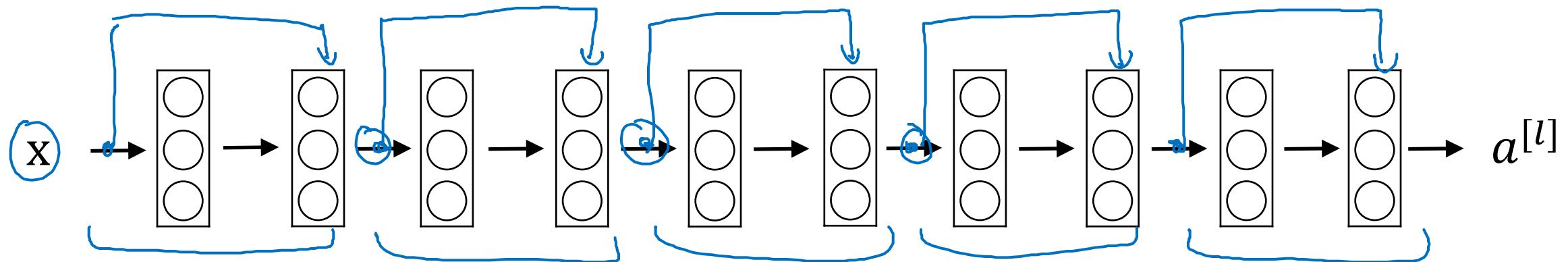
Residual block



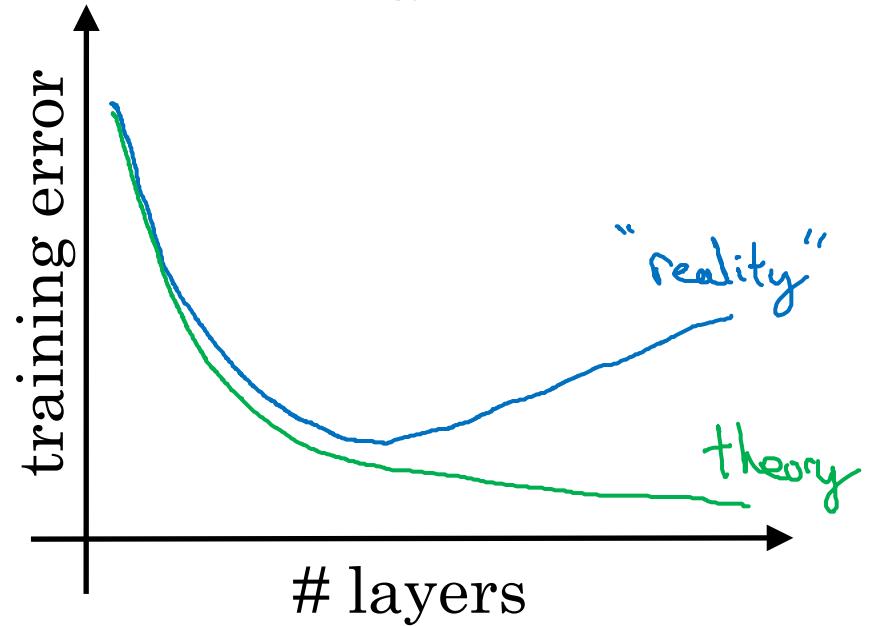
"Short cut" /skip connection



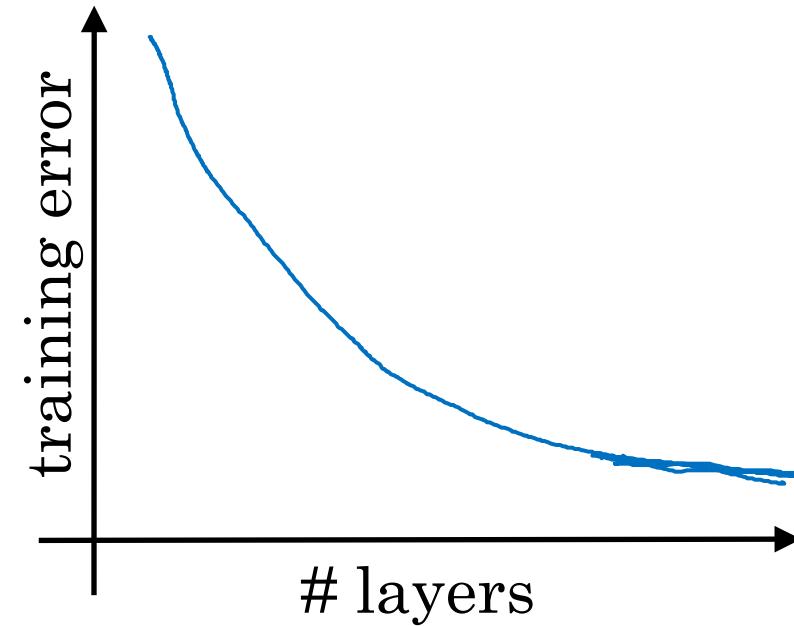
Residual Network



Plain



ResNet



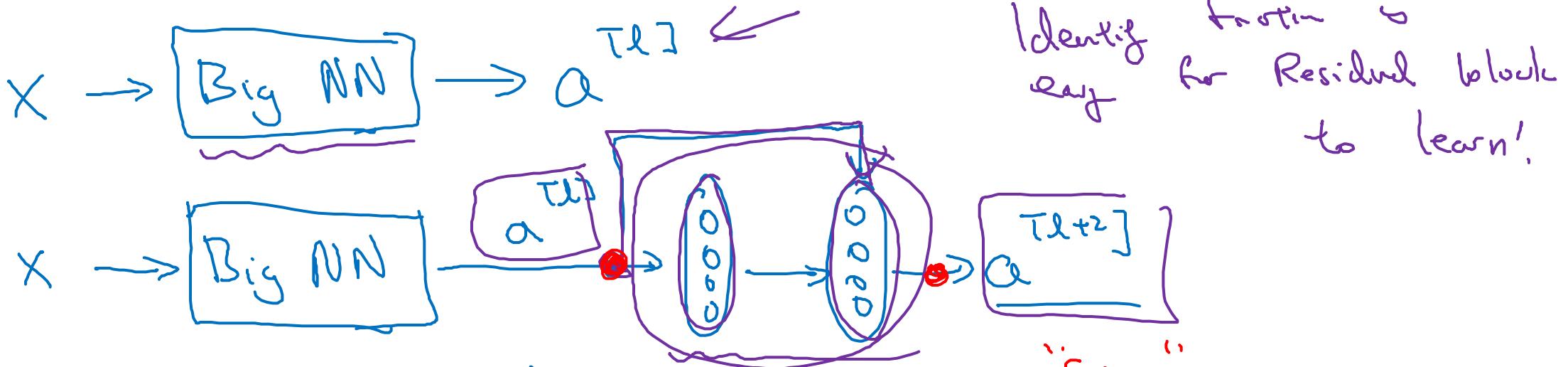


deeplearning.ai

Case Studies

Why ResNets work

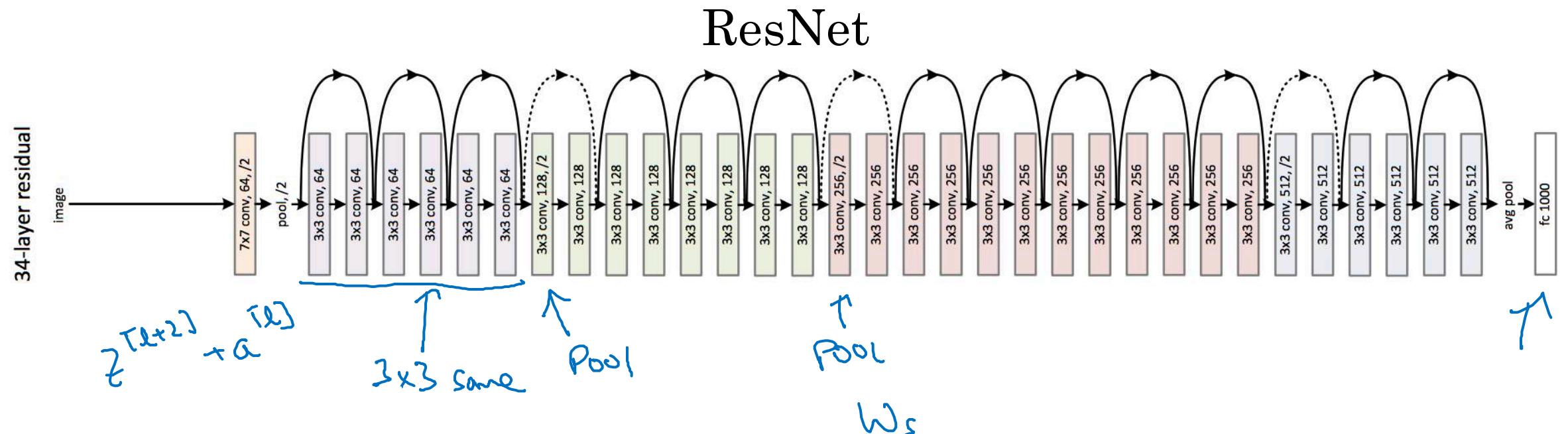
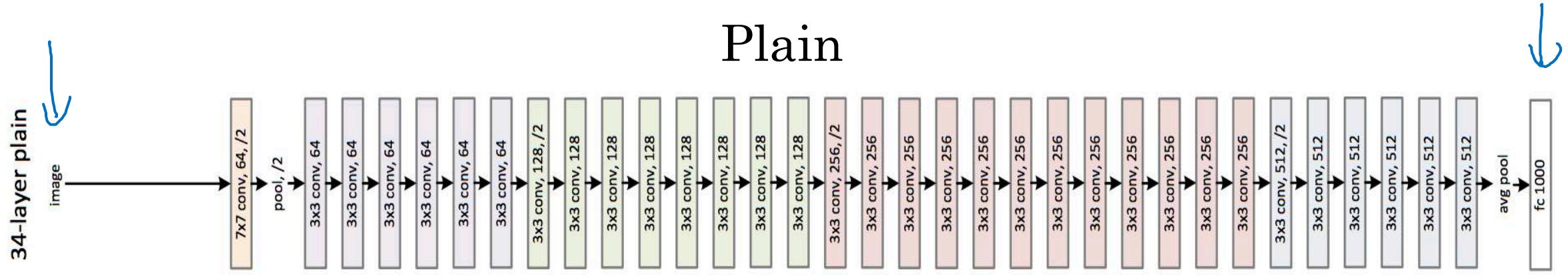
Why do residual networks work?



ReLU.

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

ResNet



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

Andrew Ng



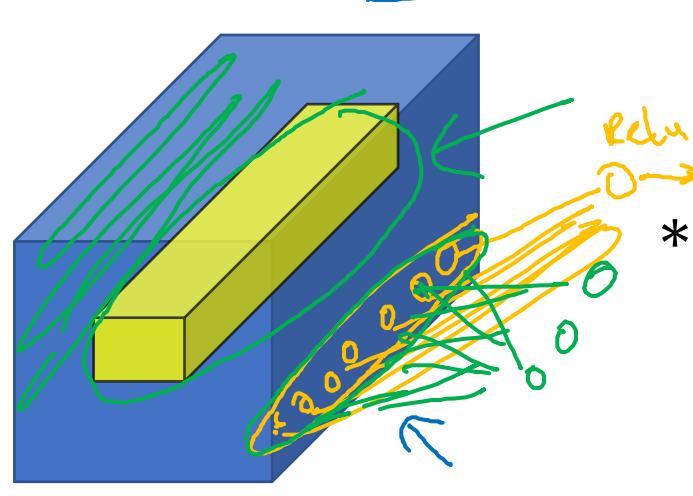
deeplearning.ai

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



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

*

2



=

32 \rightarrow # filters.

$n_c^{[l+1]}$

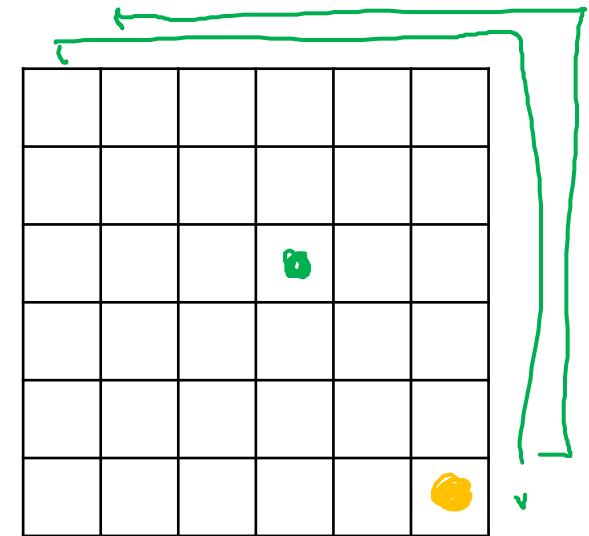
=

ReLU

Network \hookrightarrow Network

1 \times 1 \times 32

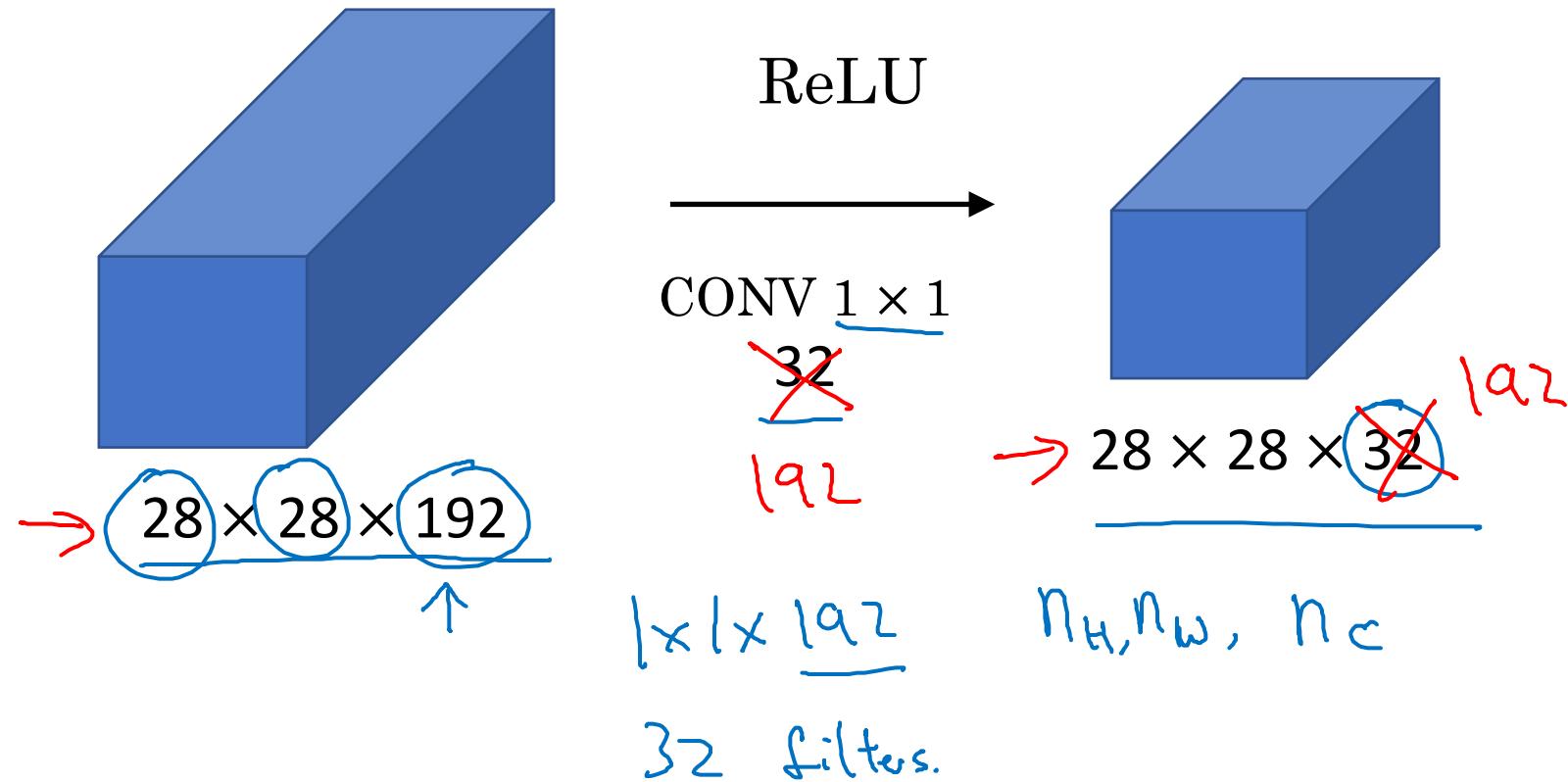
2	4	6	...



6 \times 6 \times # filters

Andrew Ng

Using 1×1 convolutions



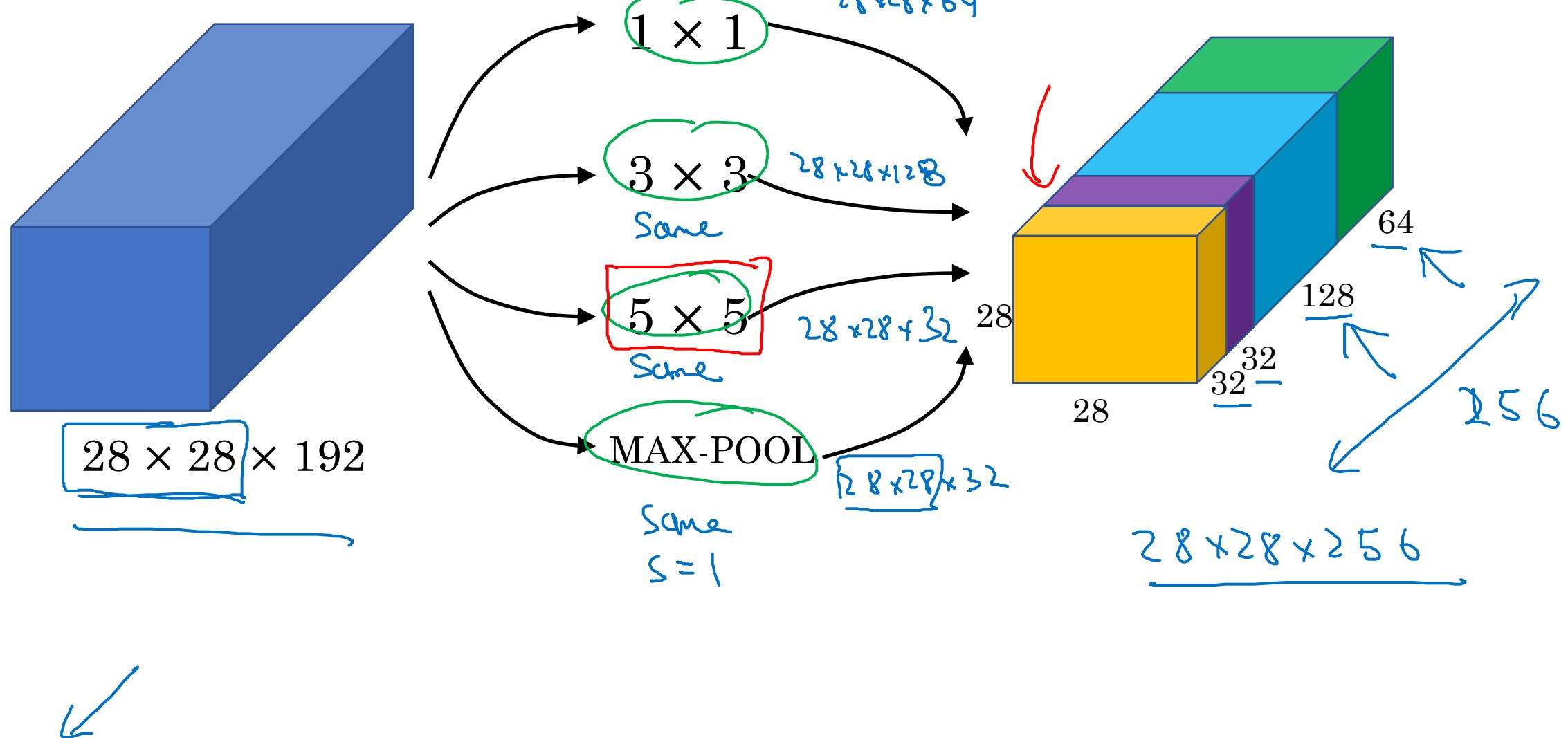


deeplearning.ai

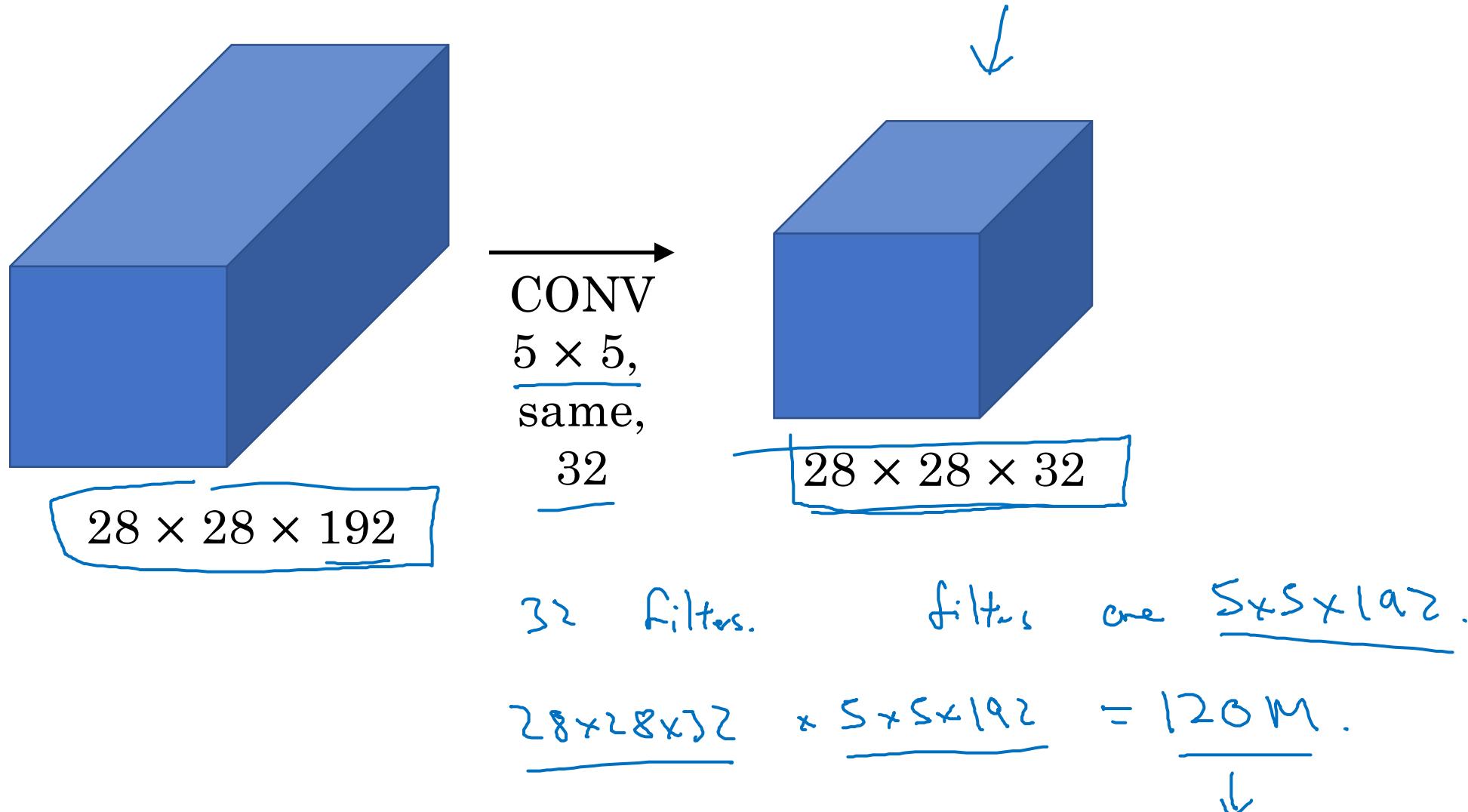
Case Studies

Inception network
motivation

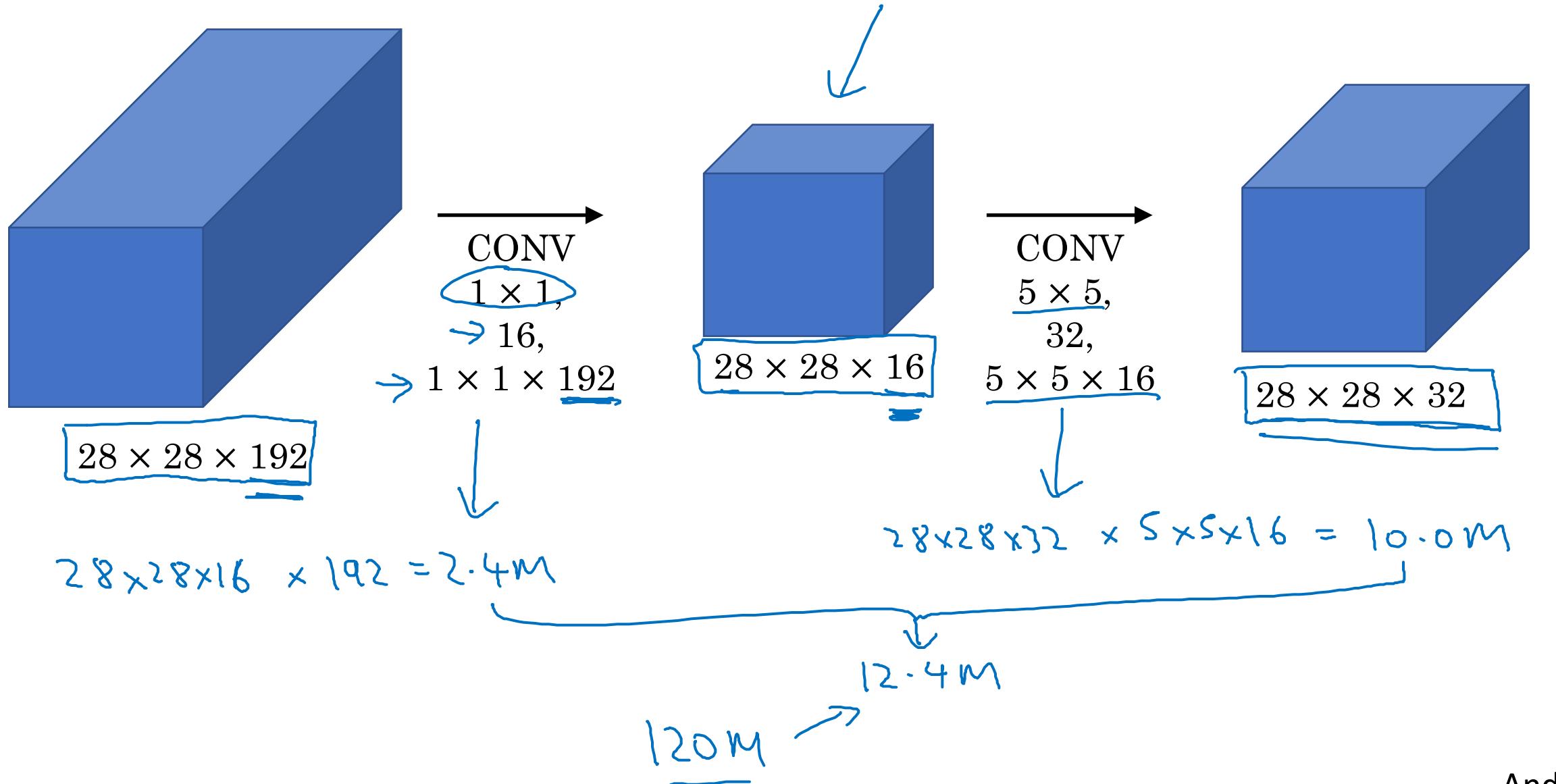
Motivation for inception network



The problem of computational cost



Using 1×1 convolution



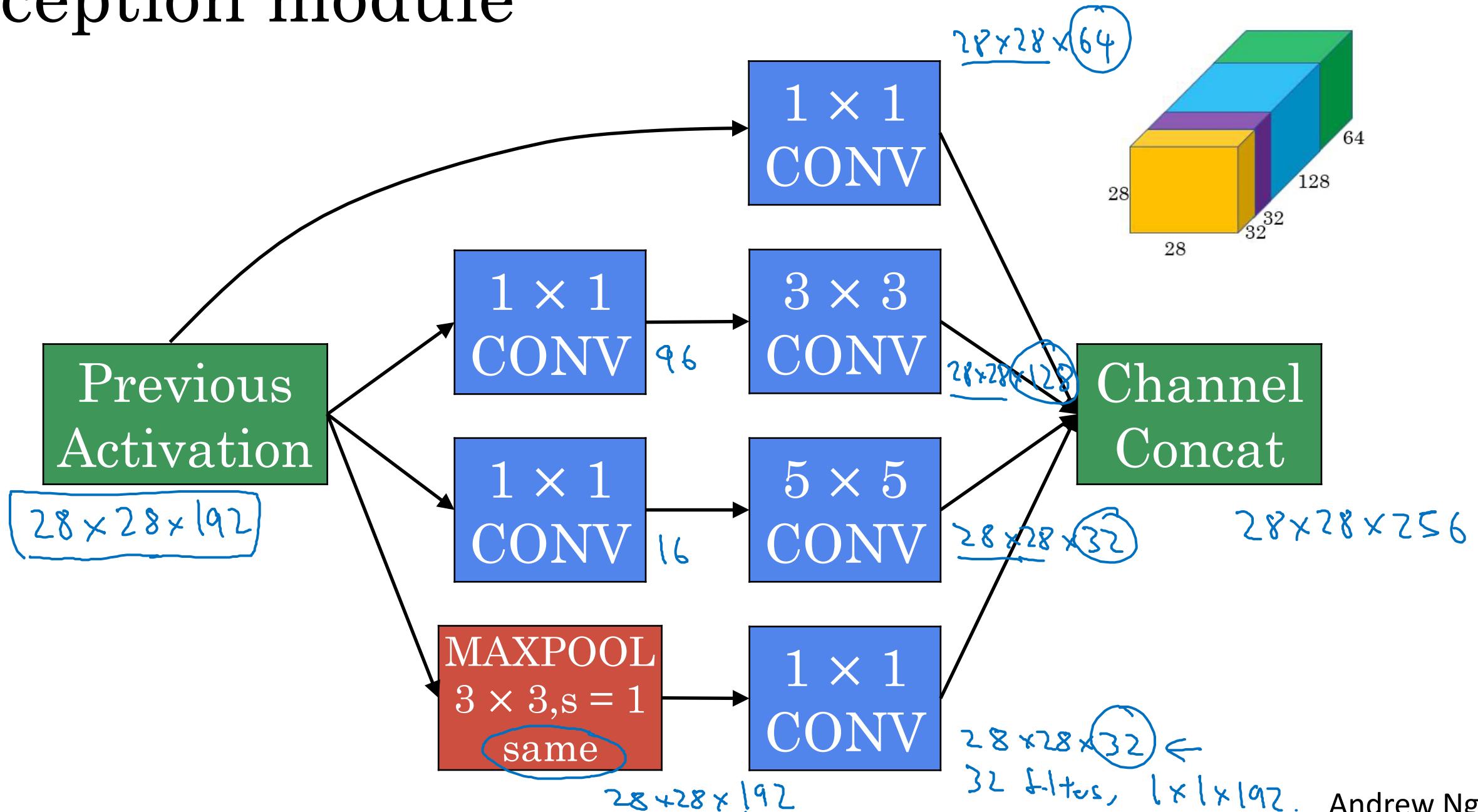


deeplearning.ai

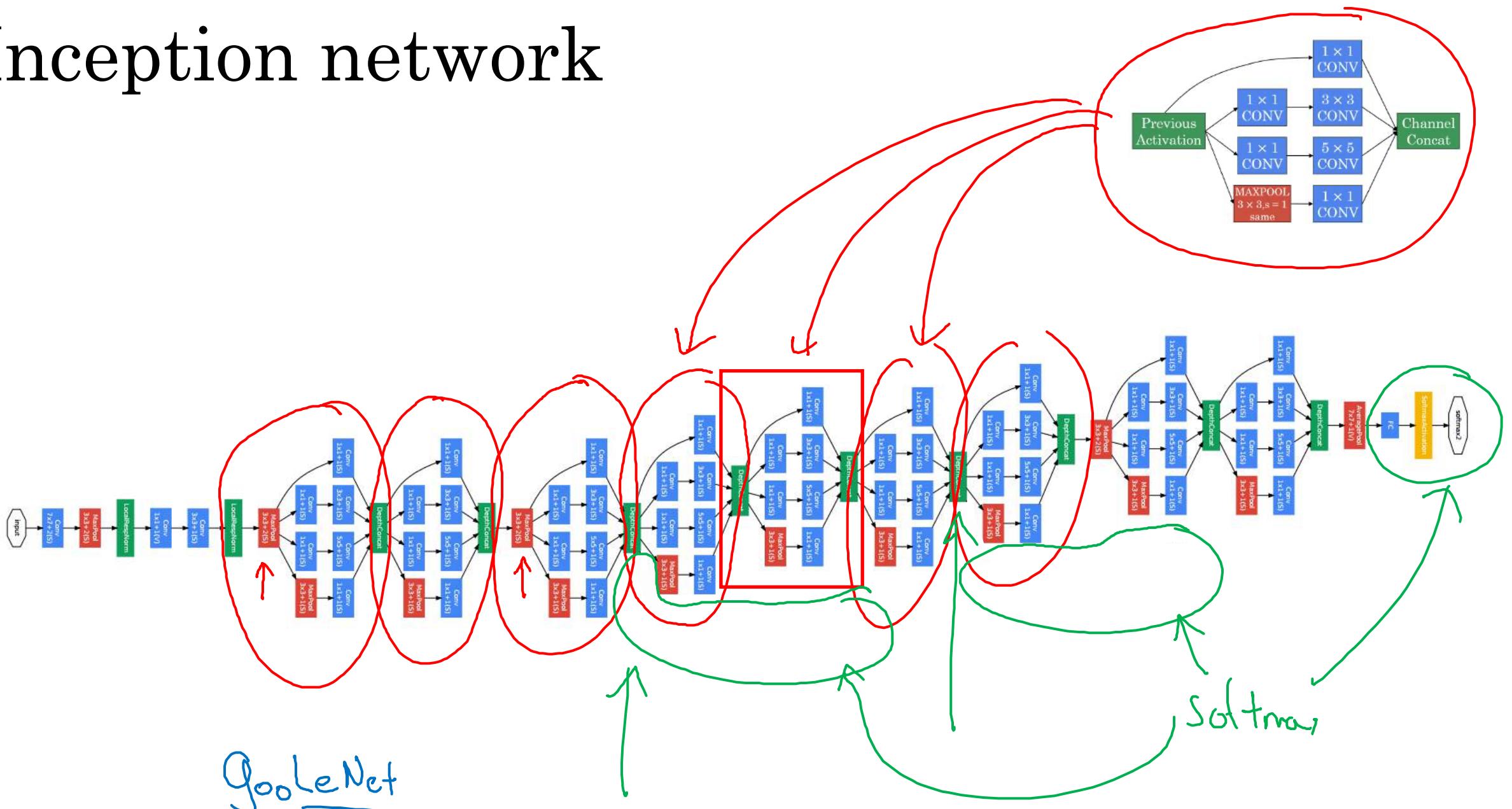
Case Studies

Inception network

Inception module



Inception network







deeplearning.ai

Convolutional Neural Networks

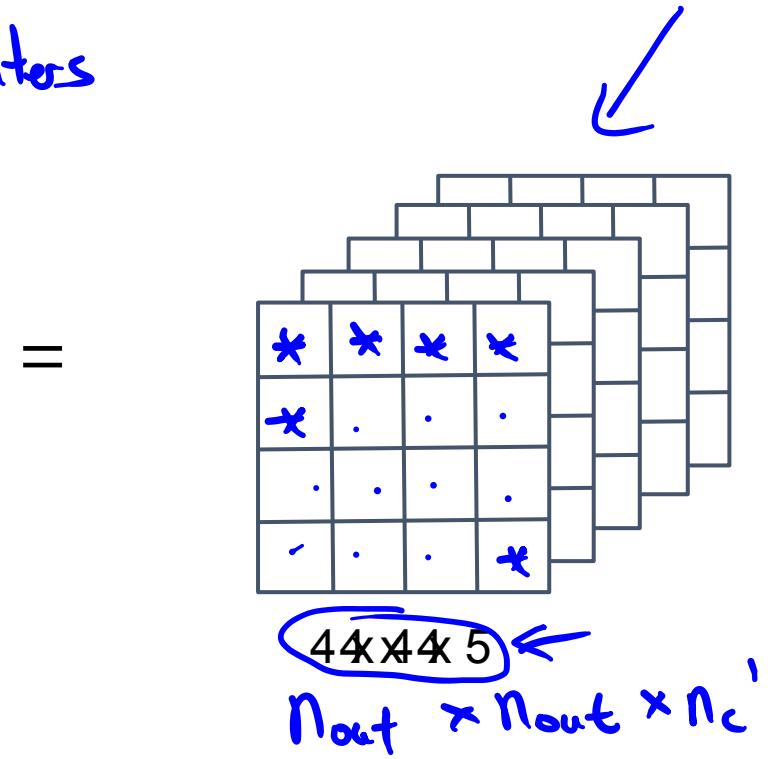
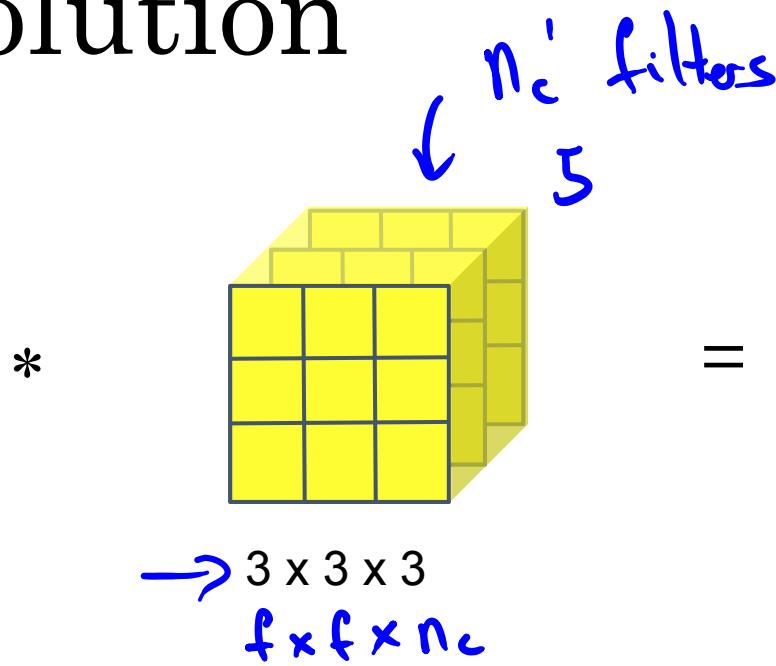
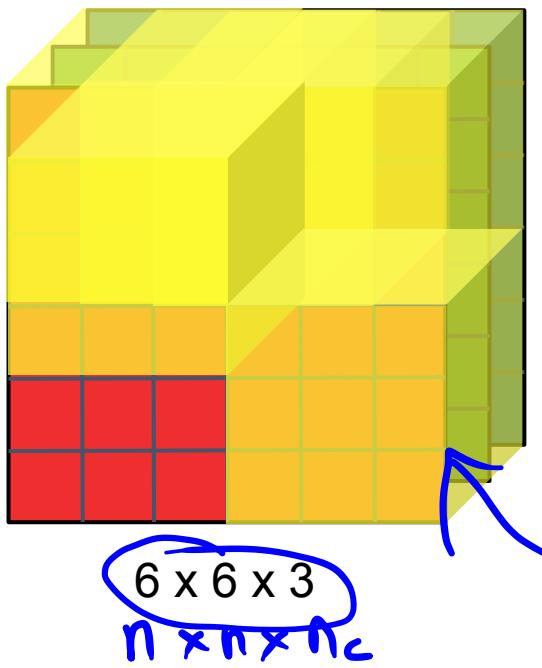
MobileNet

Motivation for MobileNets

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



Normal Convolution

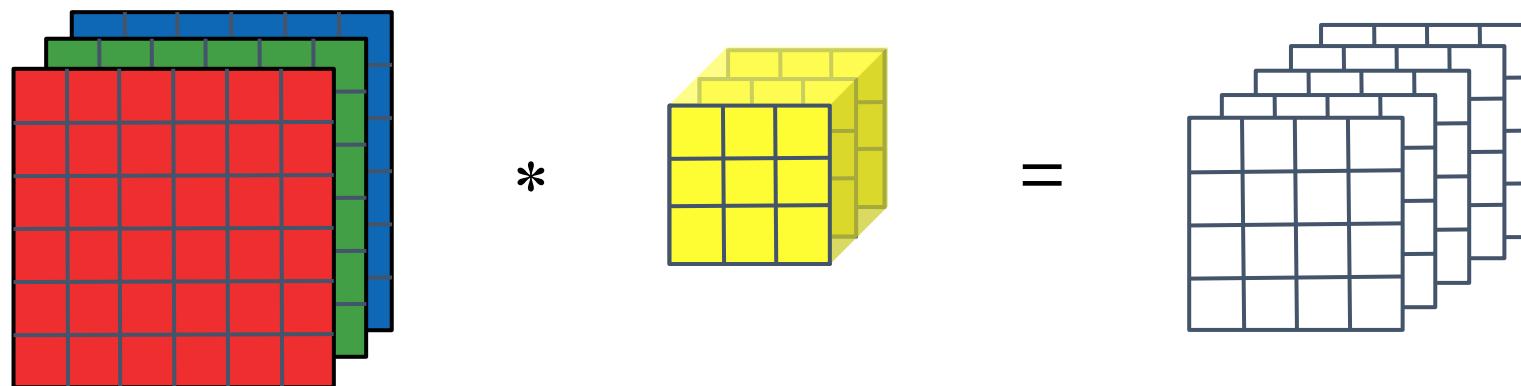


$$\text{Computational cost} = \frac{\# \text{filter params}}{3 \times 3 \times 3} \times \frac{\# \text{filter positions}}{4 \times 4} \times \# \text{of filters}$$

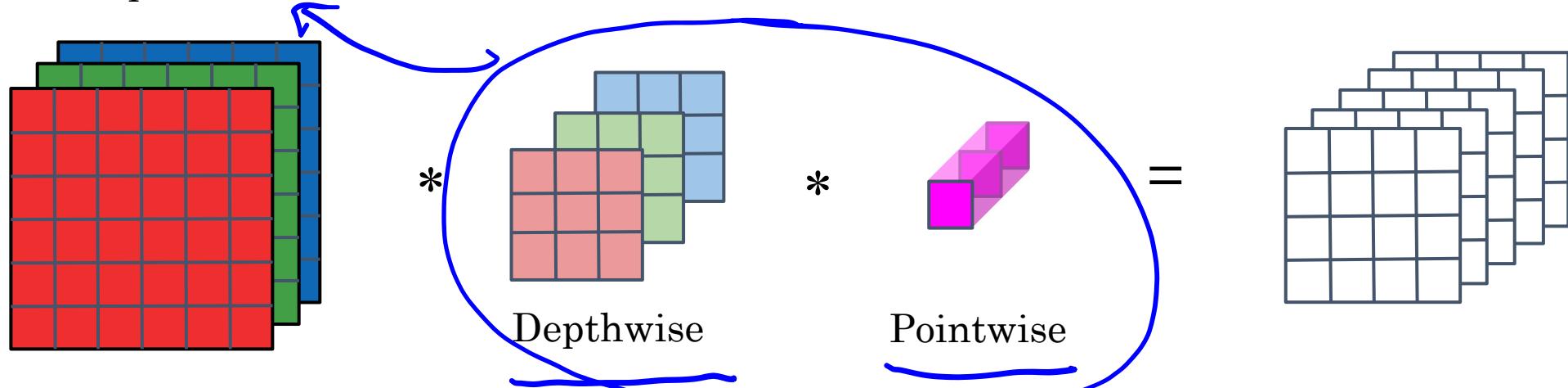
$$\underline{\rightarrow 2160} = \underline{\uparrow} \quad \underline{\uparrow} \quad \underline{\uparrow}$$

Depthwise Separable Convolution

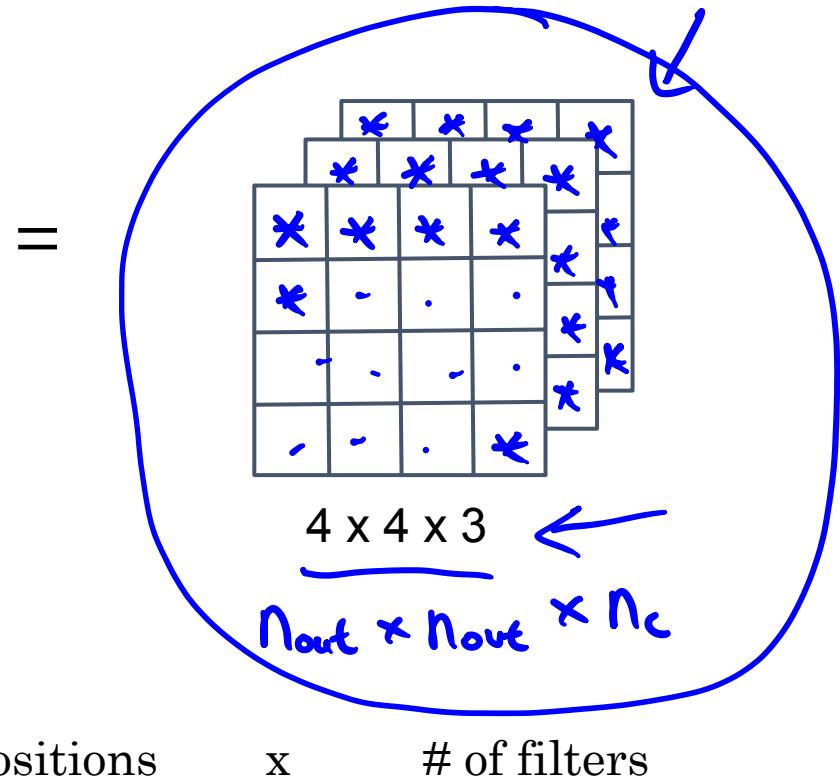
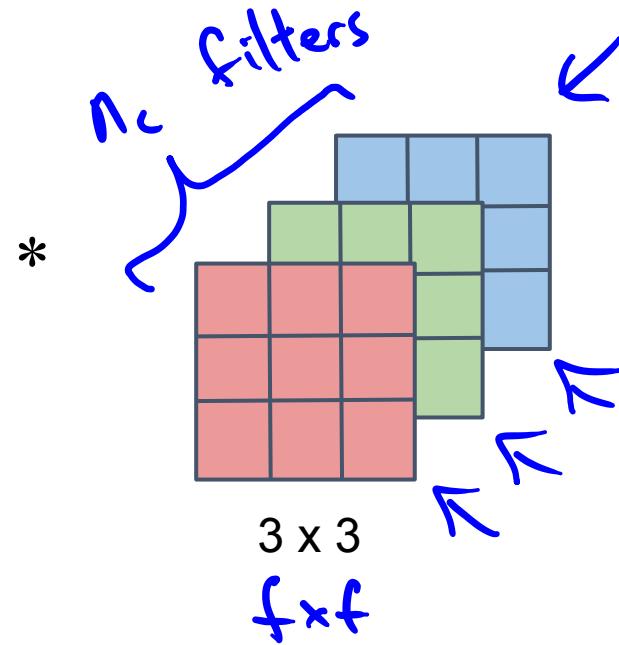
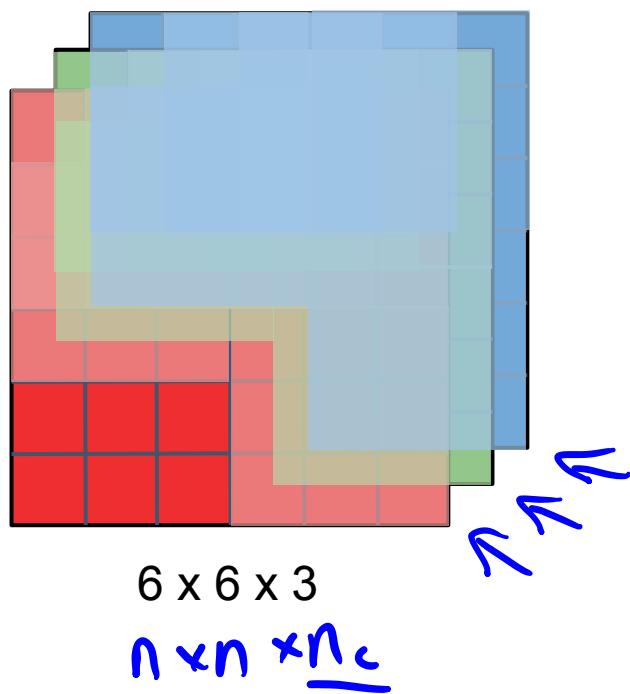
Normal Convolution



Depthwise Separable Convolution



Depthwise Convolution

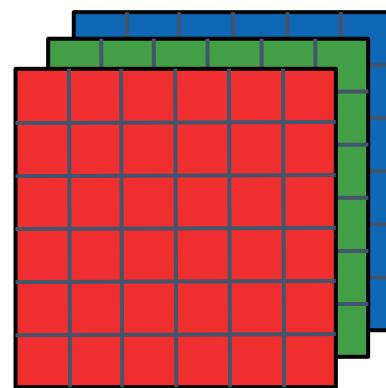


Computational cost = #filter params \times # filter positions \times # of filters

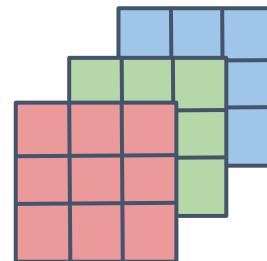
$$432 = \underbrace{3 \times 3}_{\# \text{filter params}} \times \underbrace{4 \times 4}_{\# \text{filter positions}} \times \underbrace{3}_{\# \text{of filters}}$$

Depthwise Separable Convolution

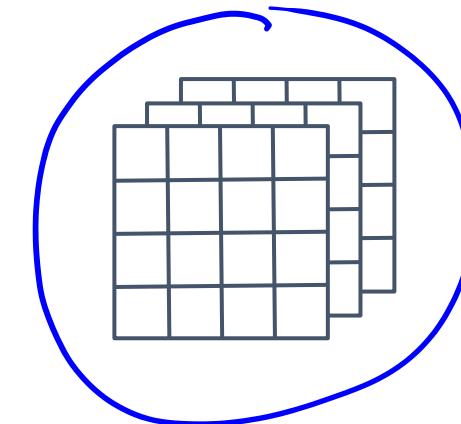
Depthwise Convolution



*

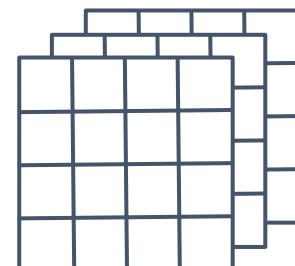


=



432

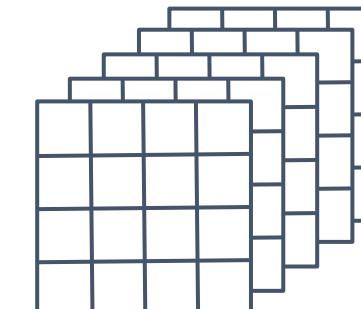
Pointwise Convolution



*

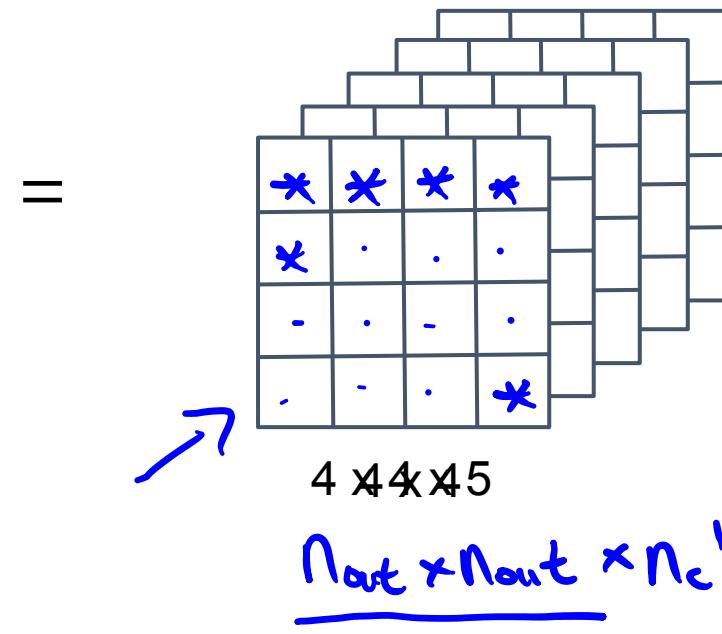
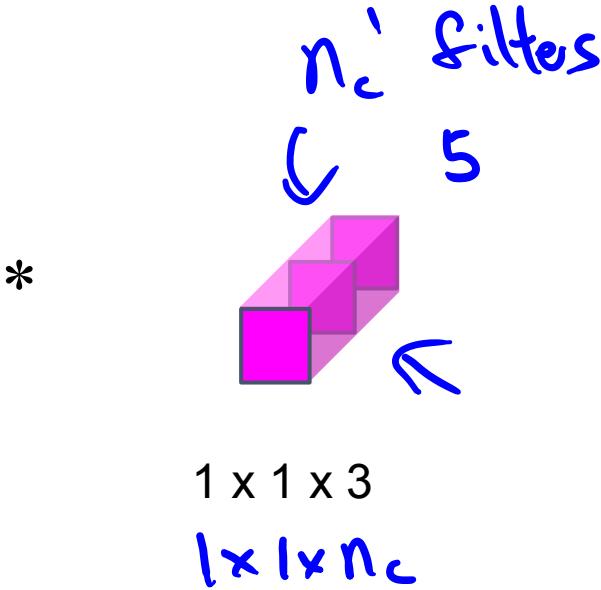
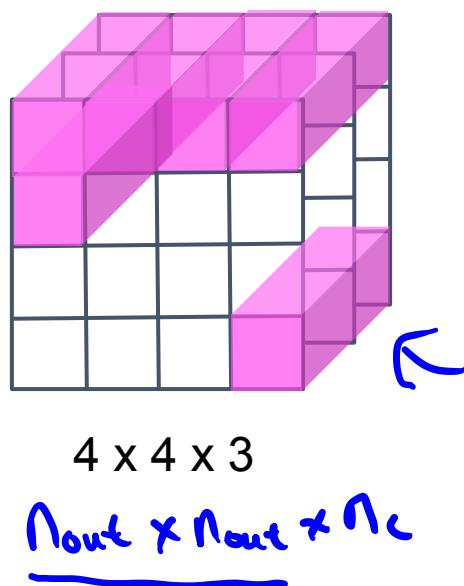


=



4x4x5

Pointwise Convolution

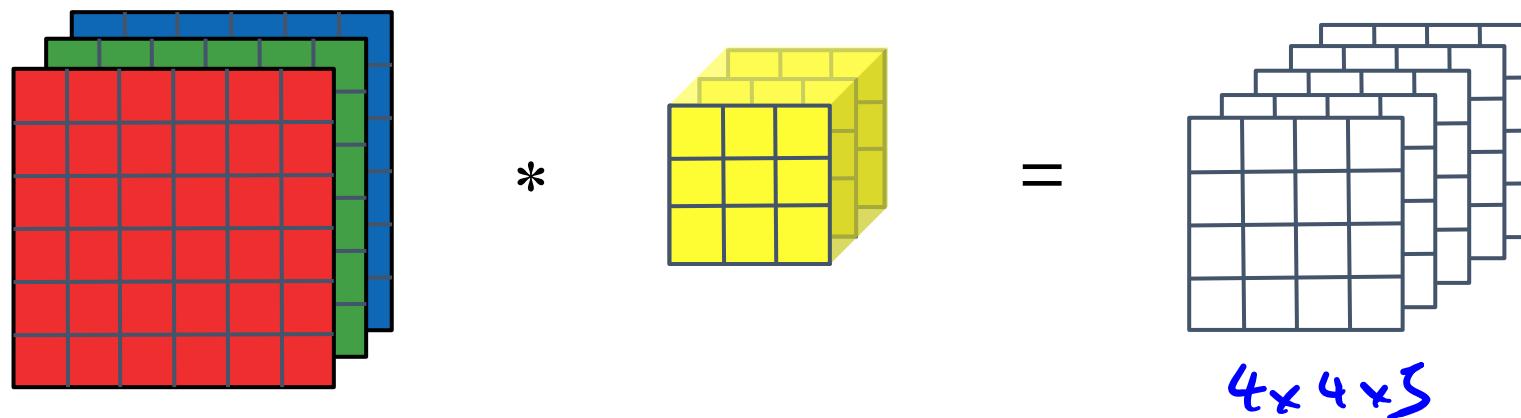


Computational cost = #filter params \times # filter positions \times # of filters

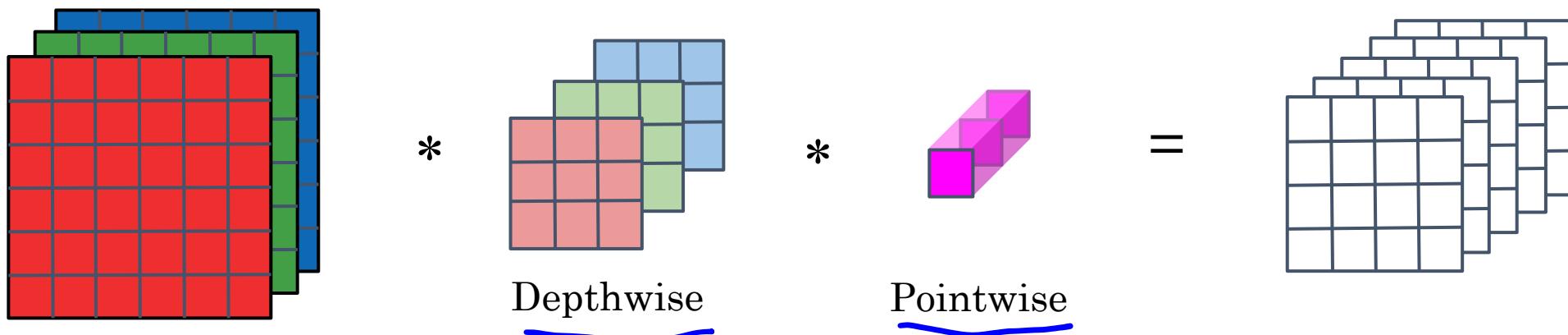
$$240 = 1 \times 1 \times 3 \times 4 \times 4 \times 5$$

Depthwise Separable Convolution

Normal Convolution



Depthwise Separable Convolution



Cost Summary

Cost of normal convolution

2160

Cost of depthwise separable convolution

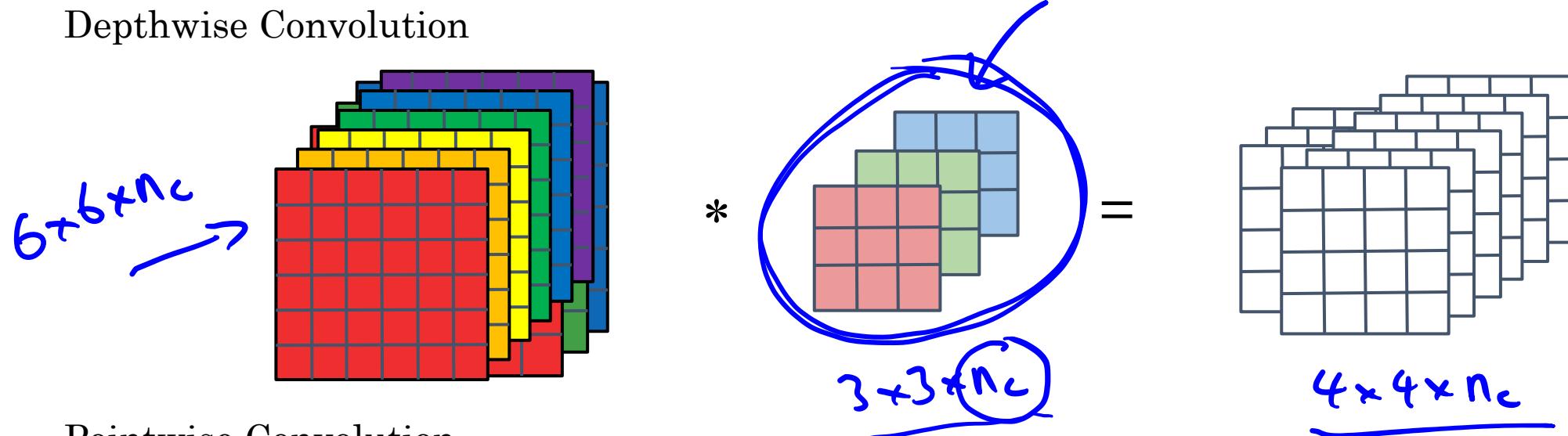
$$\begin{array}{l} \text{depthwise} + \text{pointwise} \\ 432 + 240 = 672 \end{array}$$

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

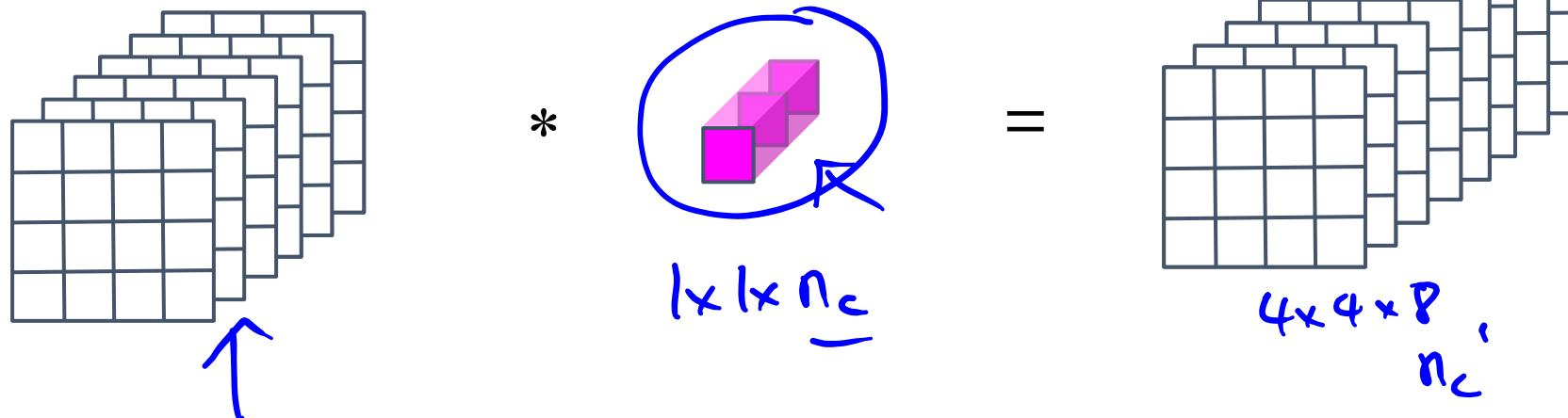
$$\begin{aligned} &= \frac{1}{n_c} + \frac{1}{f^2} \\ &\quad \frac{1}{s} + \frac{1}{q} \\ &= \frac{1}{512} + \frac{1}{3^2} \\ &\quad \nwarrow \quad \swarrow \\ &\text{n10 times cheaper} \end{aligned}$$

Depthwise Separable Convolution

Depthwise Convolution



Pointwise Convolution





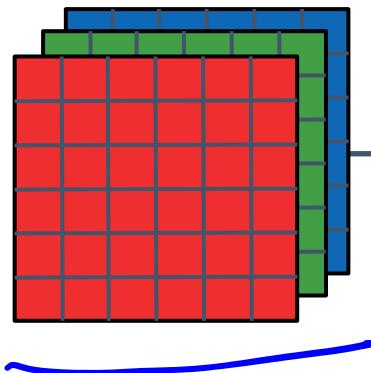
deeplearning.ai

Convolutional Neural Networks

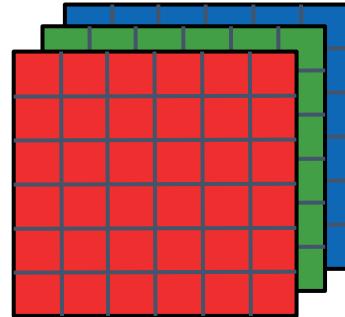
MobileNet Architecture

MobileNet

MobileNet v1



MobileNet v2



13 times

POOL, FC, SOFTMAX

17 times

Residual Connection

POOL, FC,
SOFTMAX

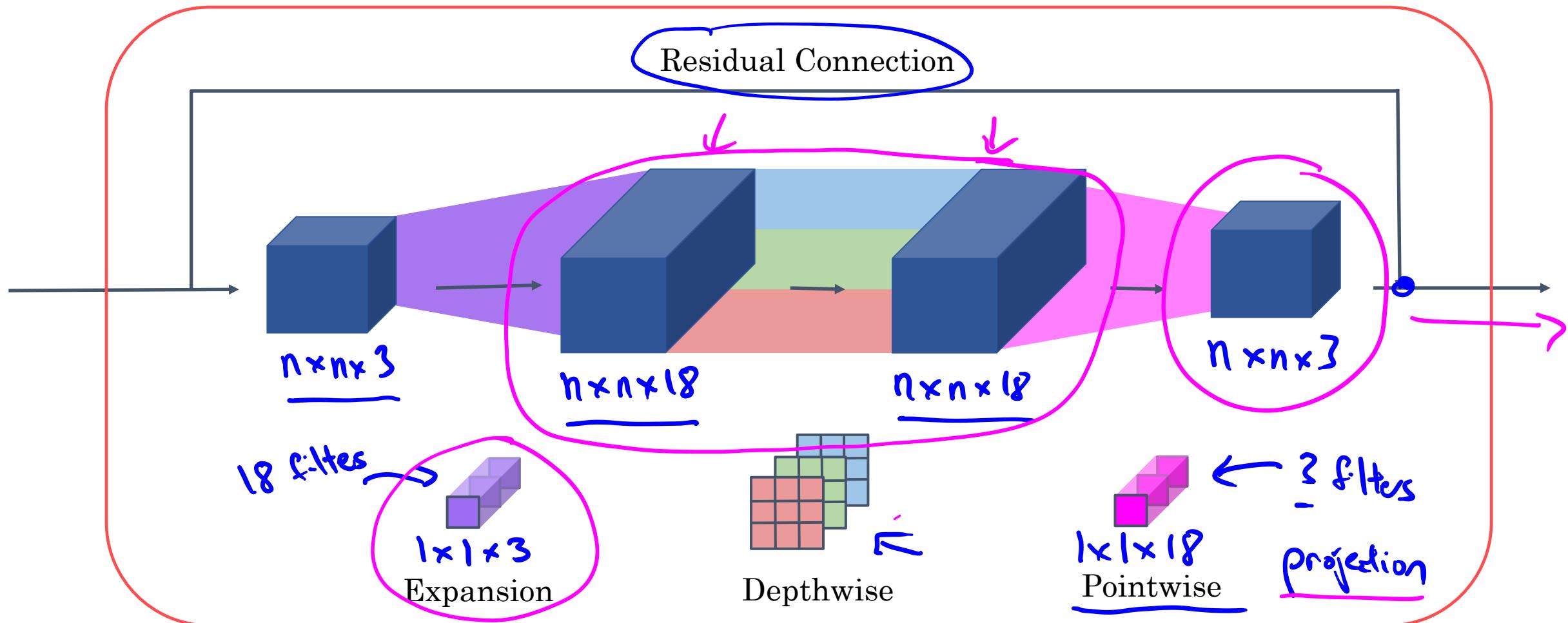
Expansion

Depthwise

Projection

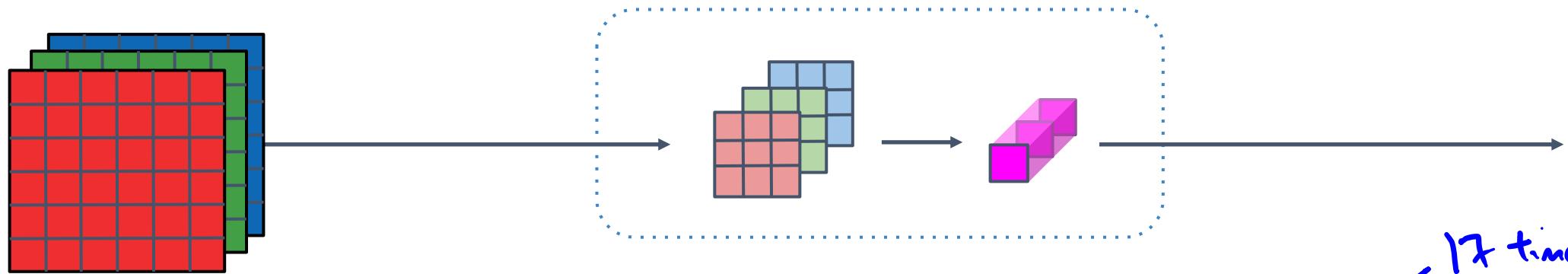
Bottleneck

MobileNet v2 Bottleneck

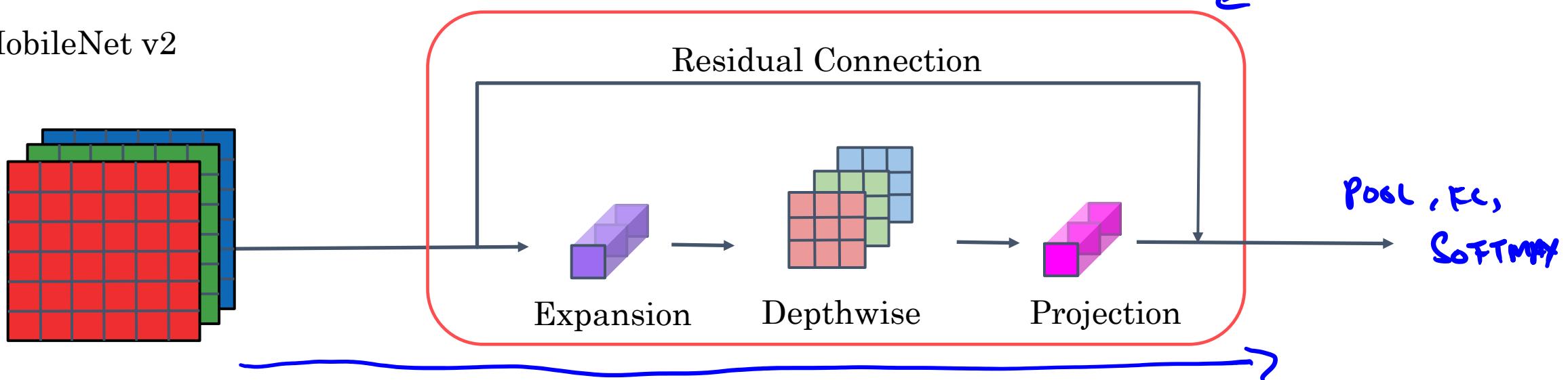


MobileNet

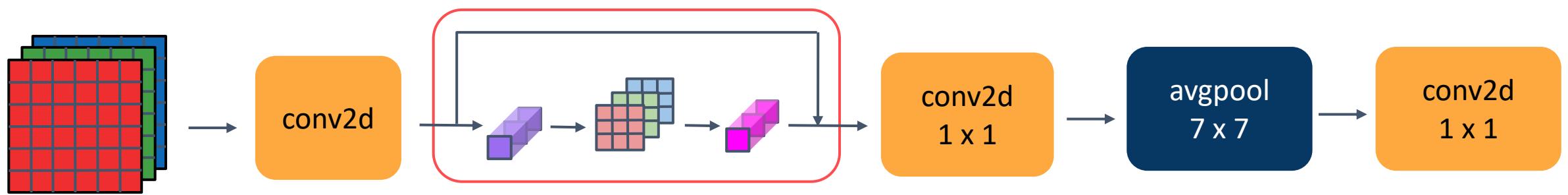
MobileNet v1



MobileNet v2



MobileNet v2 Full Architecture



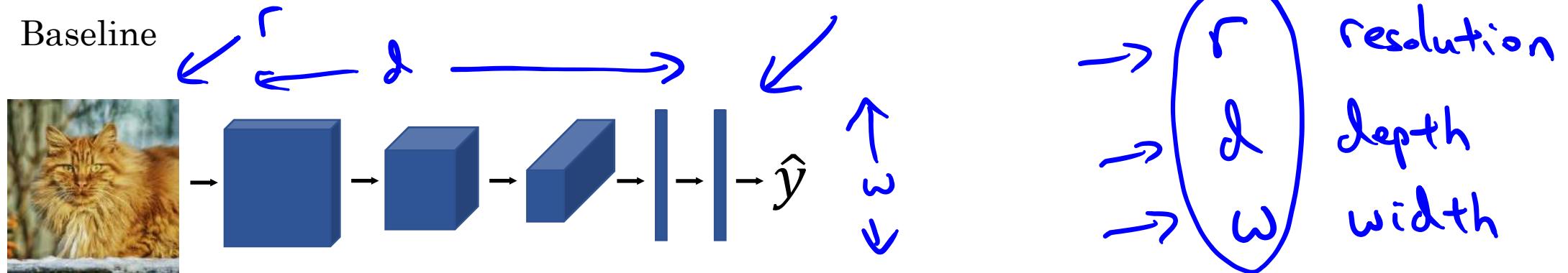


deeplearning.ai

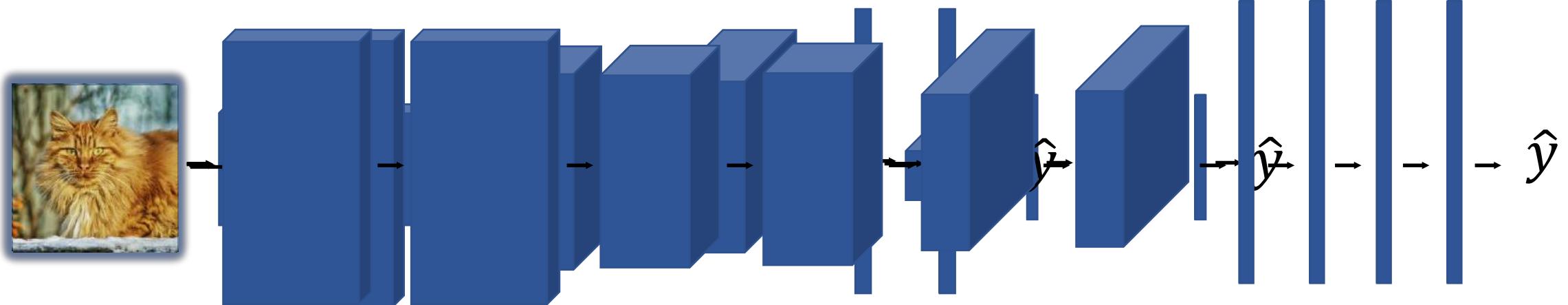
Convolutional Neural Networks

EfficientNet

EfficientNet



Without Rescaling



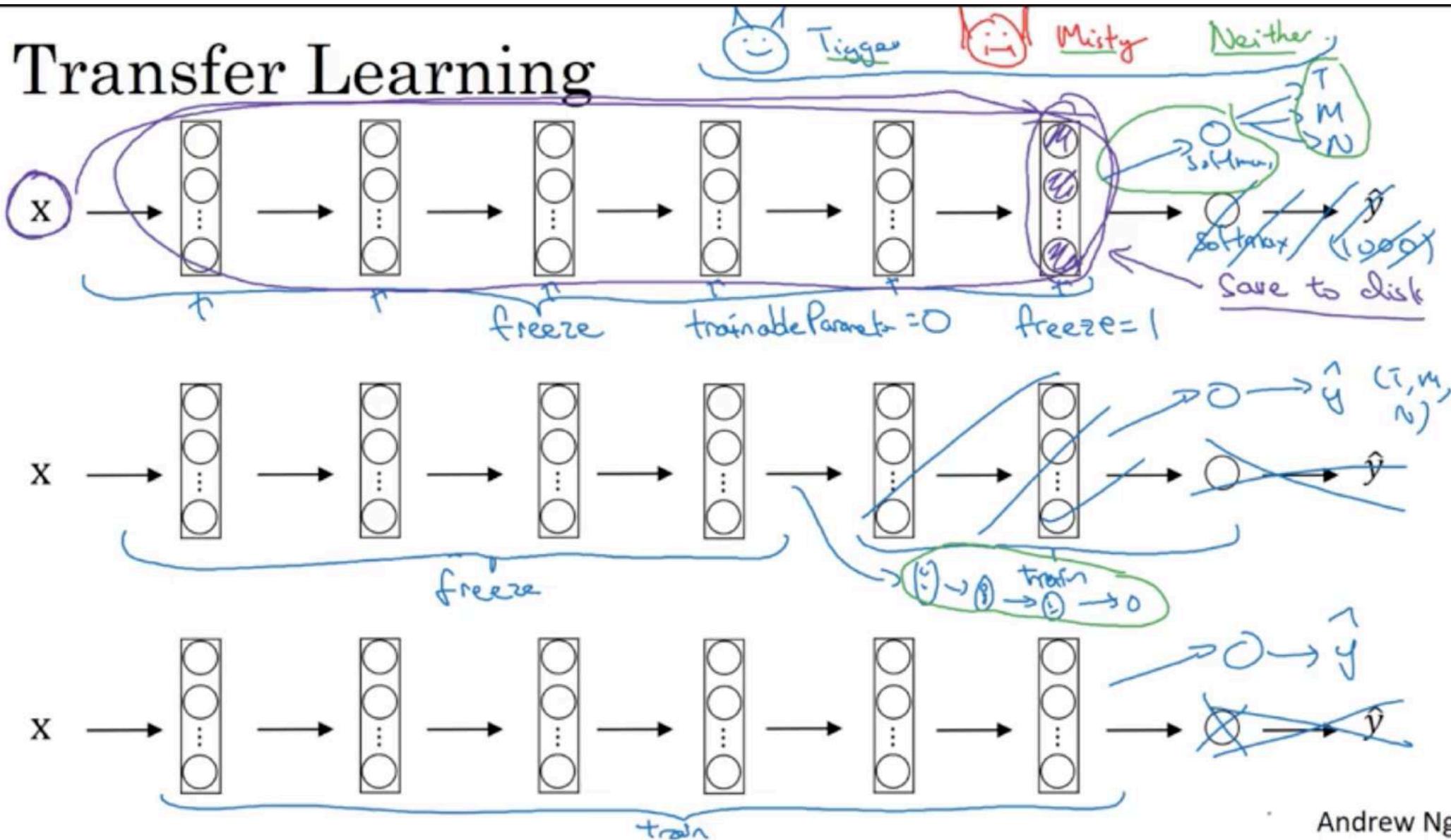


deeplearning.ai

Practical advice for
using ConvNets

Transfer Learning

Transfer Learning



Andrew Ng



deeplearning.ai

Practical advice for
using ConvNets

Data augmentation

Common augmentation method

Mirroring



yc

Random Cropping

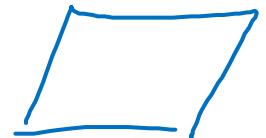


Rotation

Shearing

Local warping

...



Color shifting



R G B
↓ ↓ ↓
+20,-20,+20



-20,+20,+20



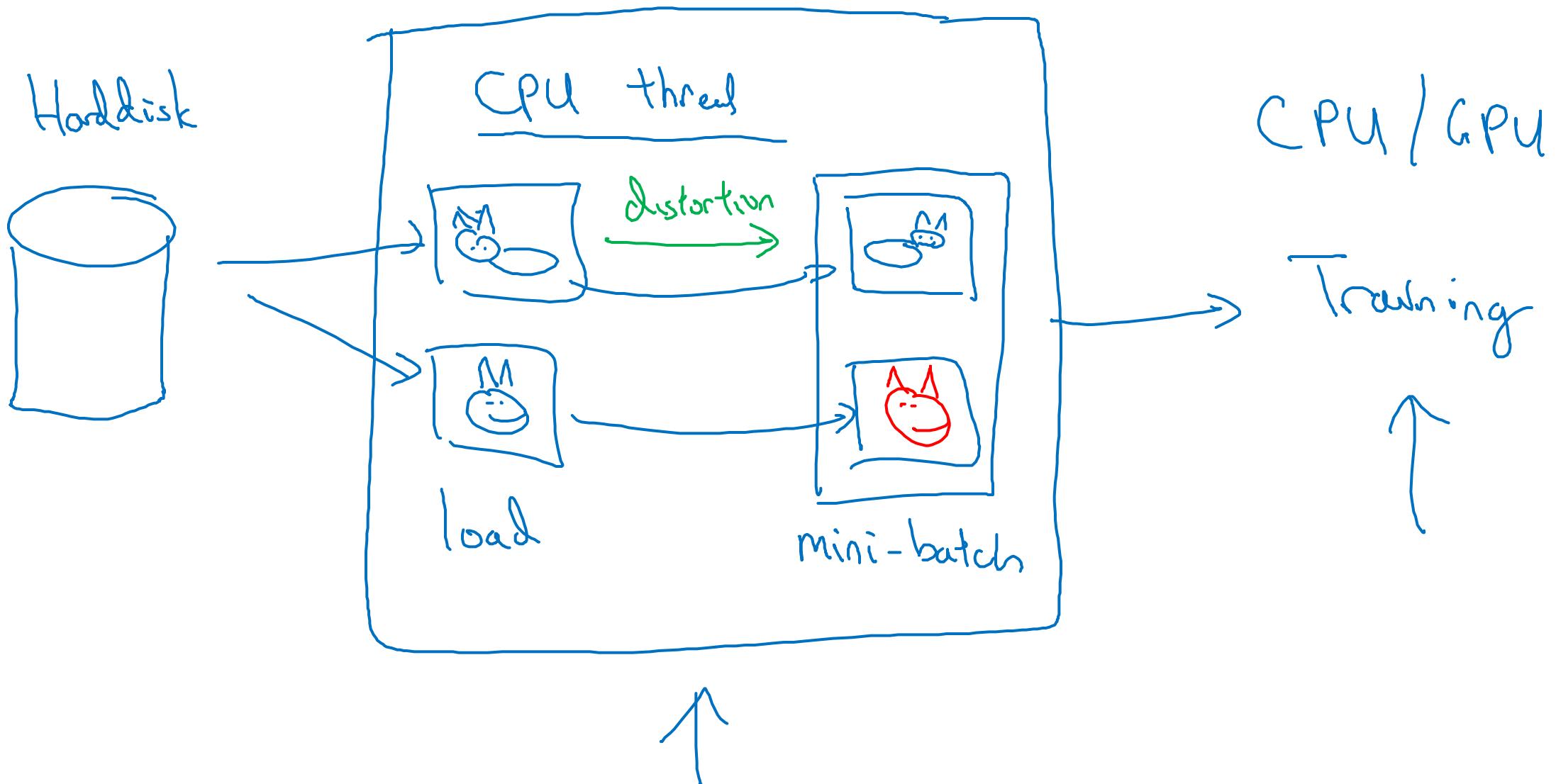
+5,0,+50



y

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

Implementing distortions during training



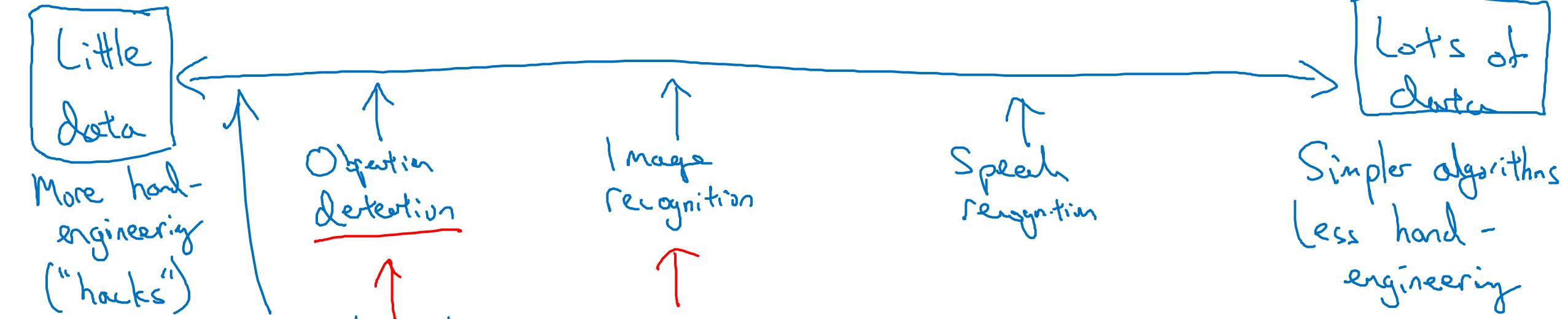


deeplearning.ai

Practical advice for
using ConvNets

The state of
computer vision

Data vs. hand-engineering



Two sources of knowledge

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



Tips for doing well on benchmarks/winning competitions

Ensembling

3 - 15 networks

$\rightarrow \hat{y}$

- Train several networks independently and average their outputs

Multi-crop at test time

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

10-crop



1



+

4



1



+

4

Use open source code

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