

Copyright Notice

These slides are distributed under the Creative Commons License.

[DeepLearning.AI](#) makes these slides available for educational purposes. You may not use or distribute these slides for commercial purposes. You may make copies of these slides and use or distribute them for educational purposes as long as you cite [DeepLearning.AI](#) as the source of the slides.

For the rest of the details of the license, see <https://creativecommons.org/licenses/by-sa/2.0/legalcode>



deeplearning.ai

Case Studies

Why look at
case studies?

Outline

Classic networks:

- LeNet-5 ←
- AlexNet ←
- VGG ←

ResNet (152)

Inception

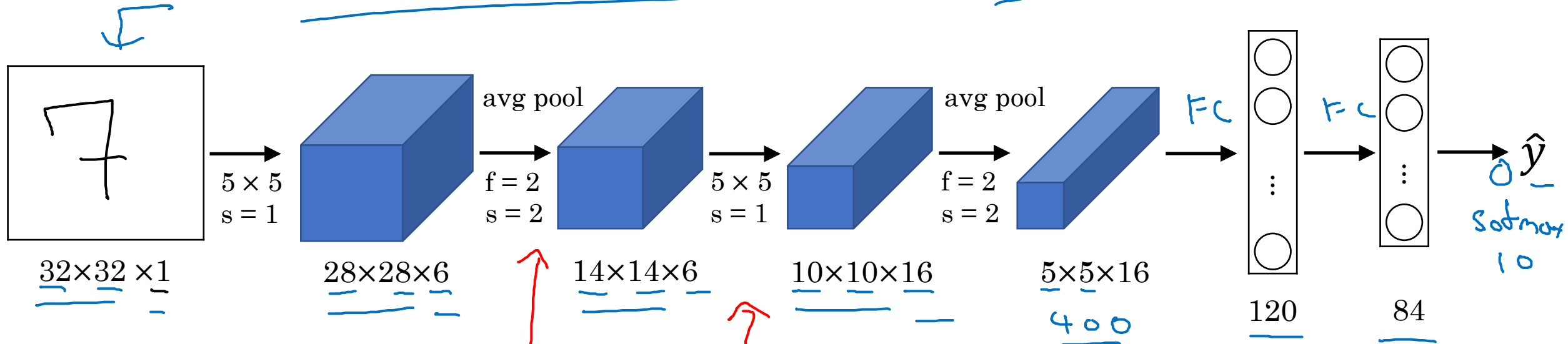


deeplearning.ai

Case Studies

Classic networks

LeNet - 5



60K parameters.

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

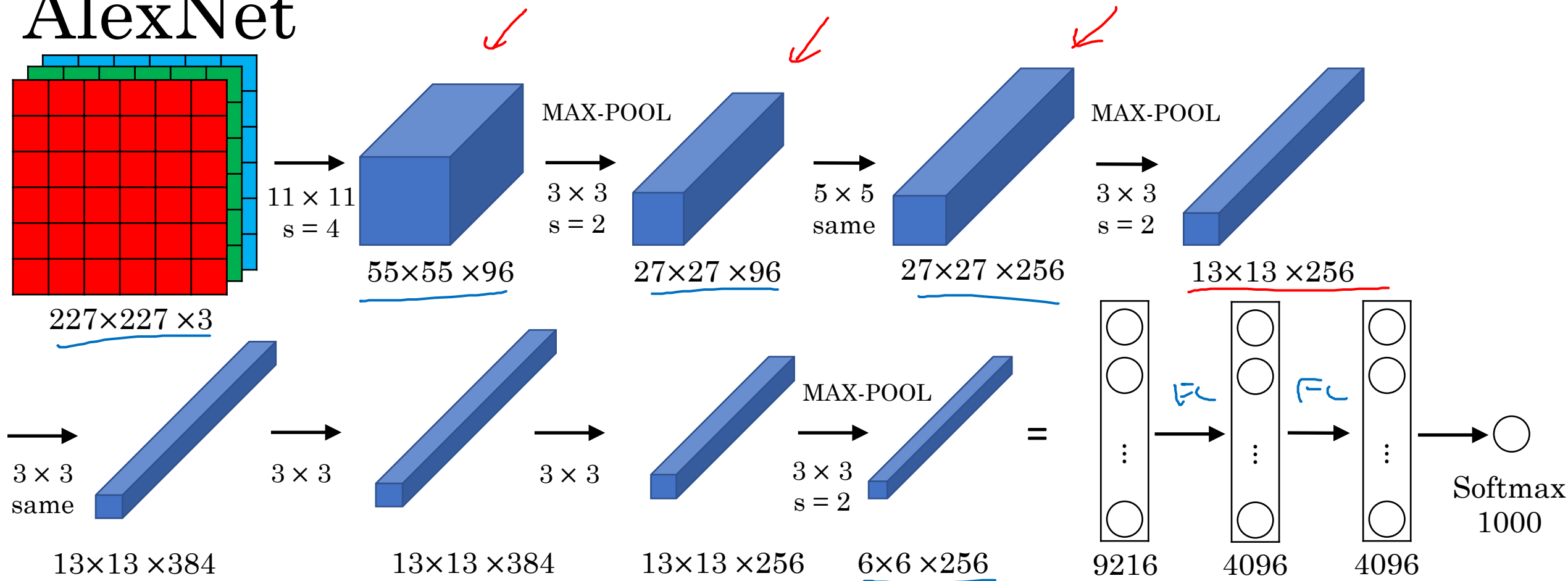
conv pool conv pool fc fc output

Advanced: sigmoid/tanh ReLU

II, III.

↓

AlexNet

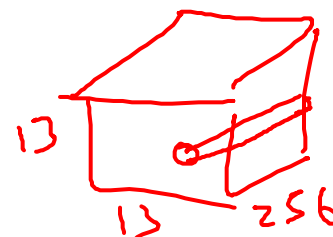


- Similar to LeNet, but much bigger.

- ReLU

- Multiple GPUs.

- Local Response Normalization (LRN)

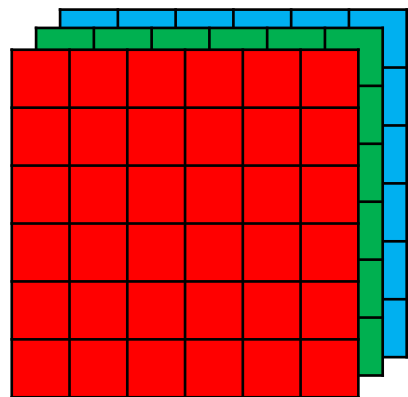


60M parameters

VGG - 16

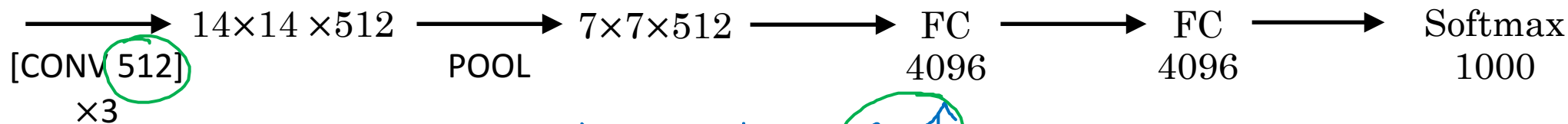
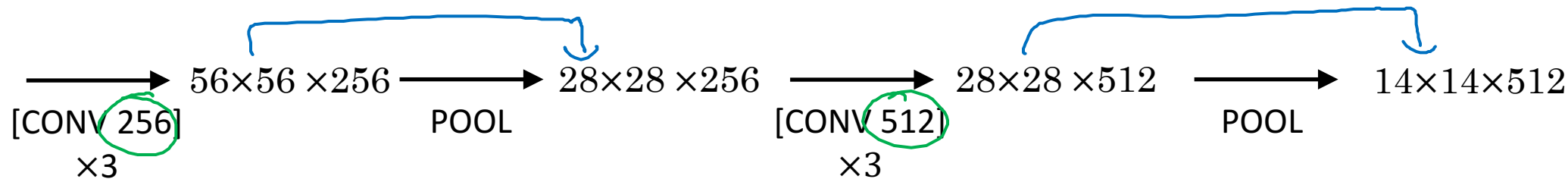
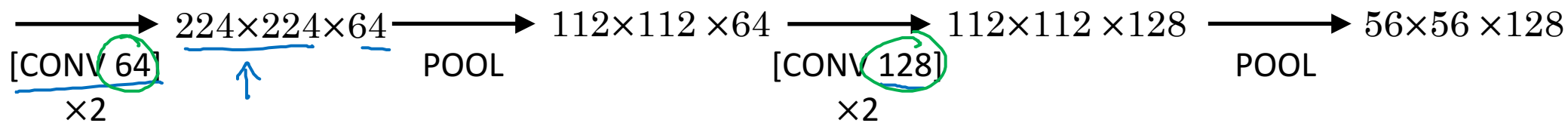
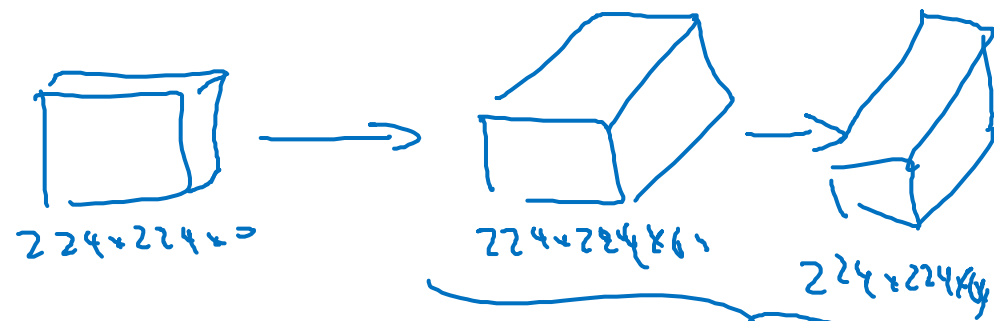
CONV = 3x3 filter, s = 1, same

MAX-POOL = 2x2, s = 2



224x224x3

VGG-19



$n_h, n_w \downarrow$

$n_c \uparrow$

$\sim 38M$

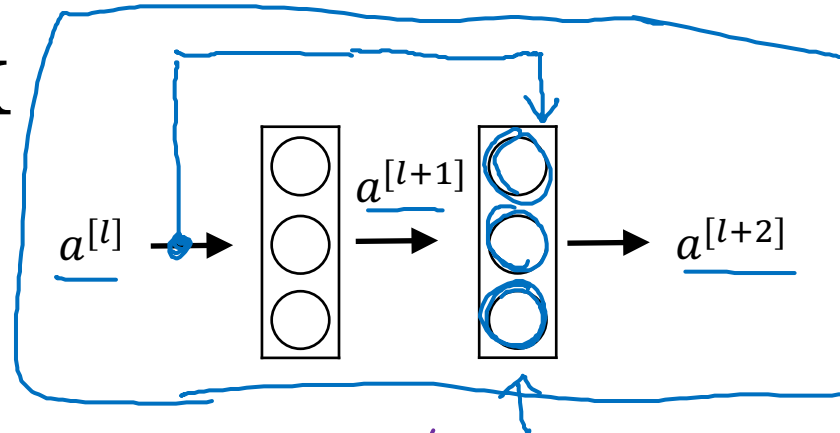


deeplearning.ai

Case Studies

Residual Networks (ResNets)

Residual block



"short cut" / skip connection



$$\underline{z^{[l+1]}} = W^{[l+1]} \underline{a^{[l]}} + b^{[l+1]}$$

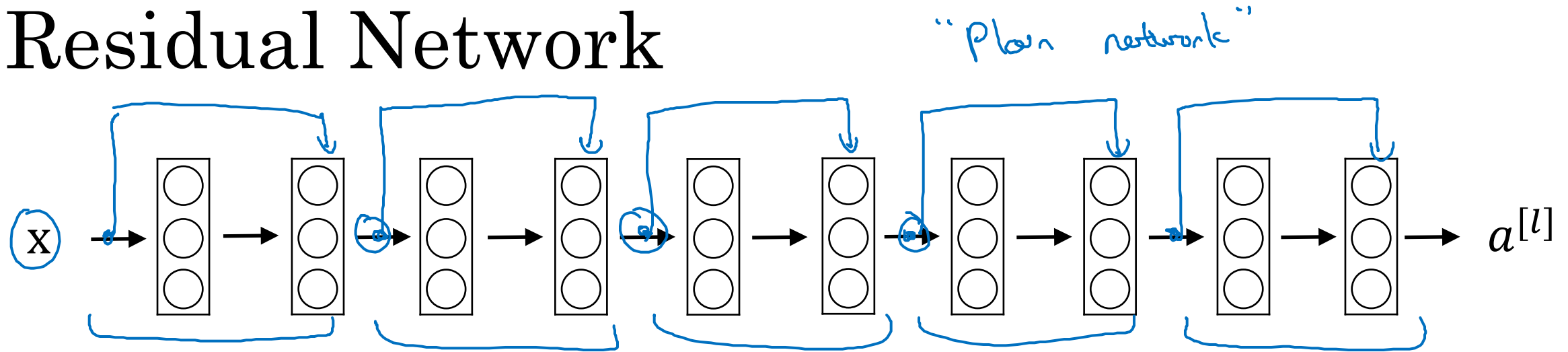
$$\underline{a^{[l+1]}} = g(\underline{z^{[l+1]}})$$

$$\underline{z^{[l+2]}} = W^{[l+2]} a^{[l+1]} + b^{[l+2]}$$

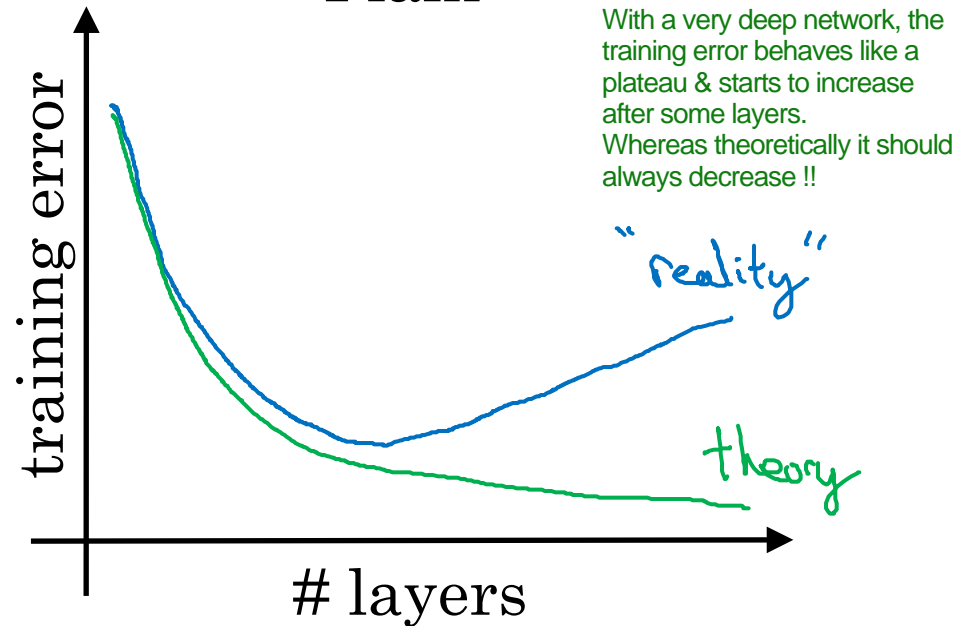
~~$$a^{[l+2]} = g(z^{[l+2]})$$~~

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

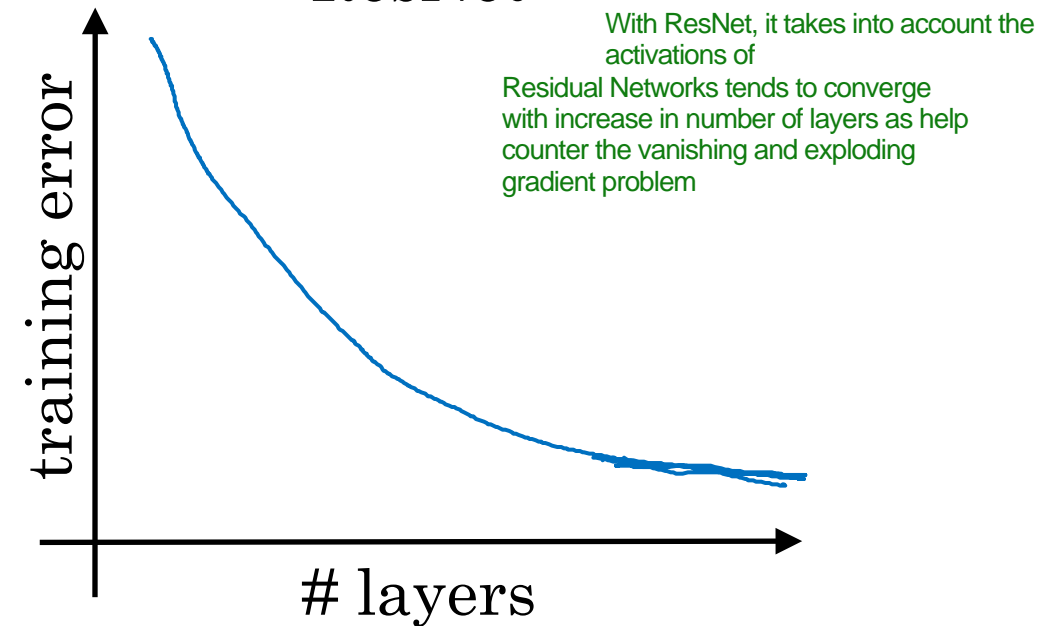
Residual Network



Plain



ResNet



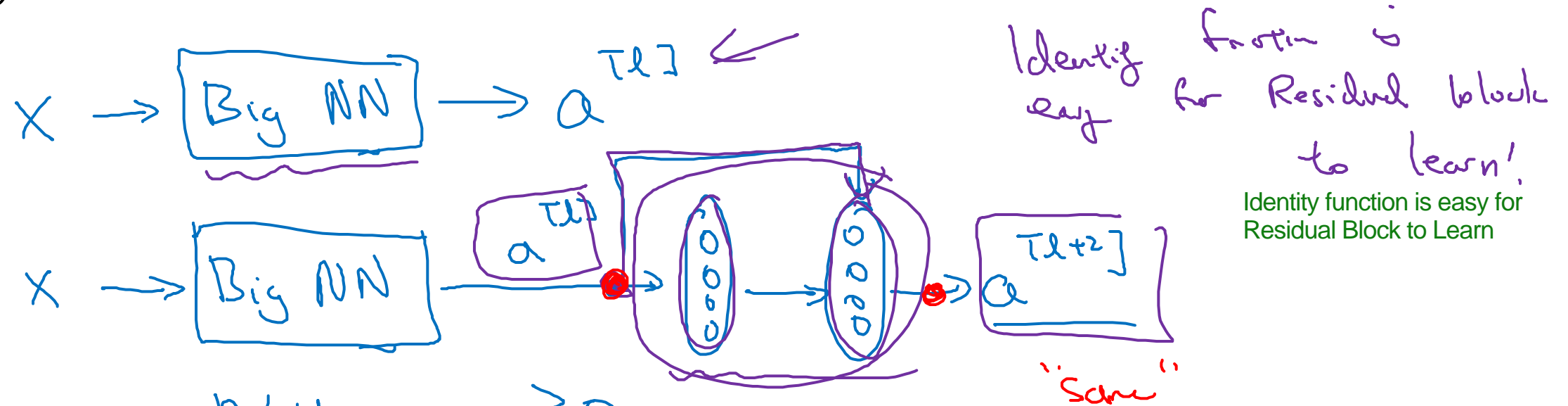


deeplearning.ai

Case Studies

Why ResNets work

Why do residual networks work?



ReLU. $a \geq 0$

$$a^{[l+2]} = g(\underbrace{z^{[l+2]} + a^{[l]}}_{\text{ReLU}})$$

$$= g(\underbrace{W^{[l+2]} a^{[l]} + b^{[l+2]}}_{\text{If } W^{[l+2]}=0, b^{[l+2]}=0} + \underbrace{W_s a^{[l]}}_{\text{ReLU}})$$

256×128 (red text) R (red text) 256×128 (red text)

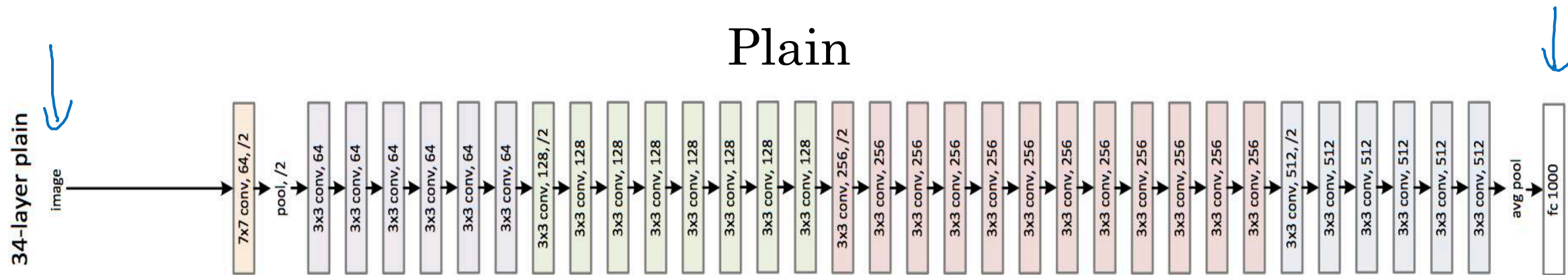
We will add this W_s only when the output activation layer has some other shape and activation from shortcut connection is different.

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

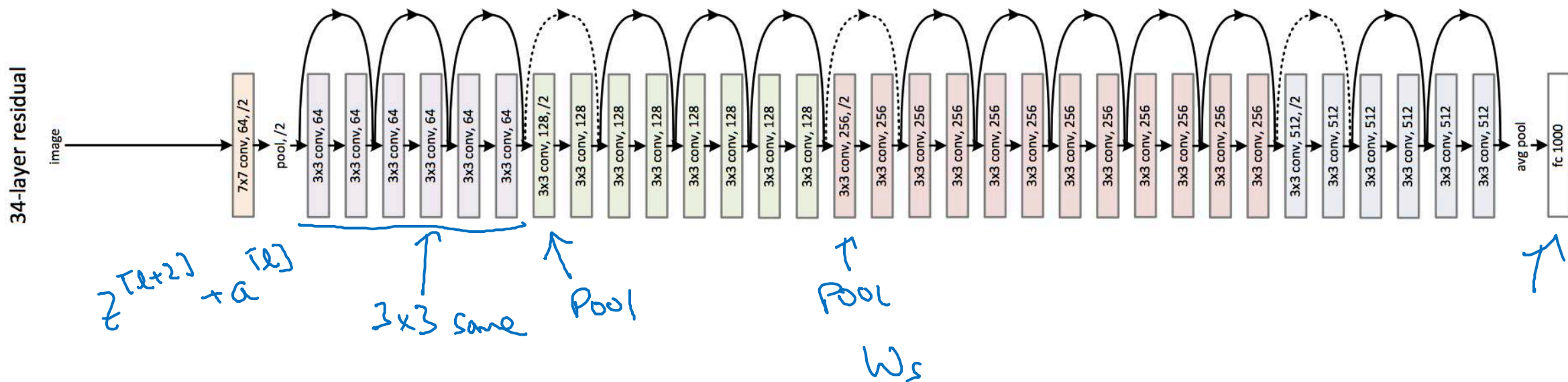
$$= \underline{a^{[l]}}$$

ResNet

Plain



ResNet





deeplearning.ai

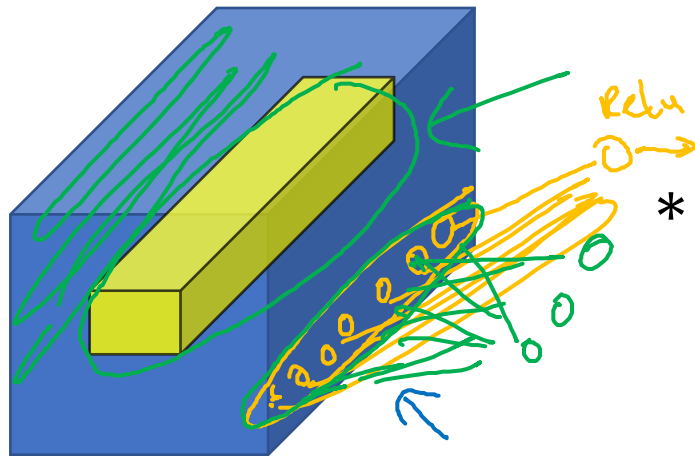
Case Studies

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

$6 \times 6 \times 1$



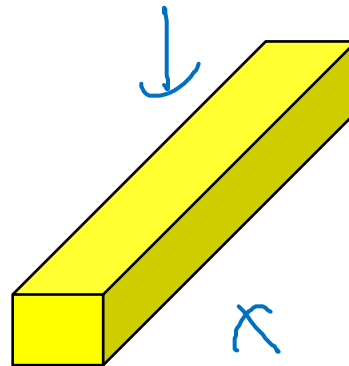
$6 \times 6 \times 32$

*

2

=

32 \rightarrow # filters.
 $n_c^{[l+1]}$



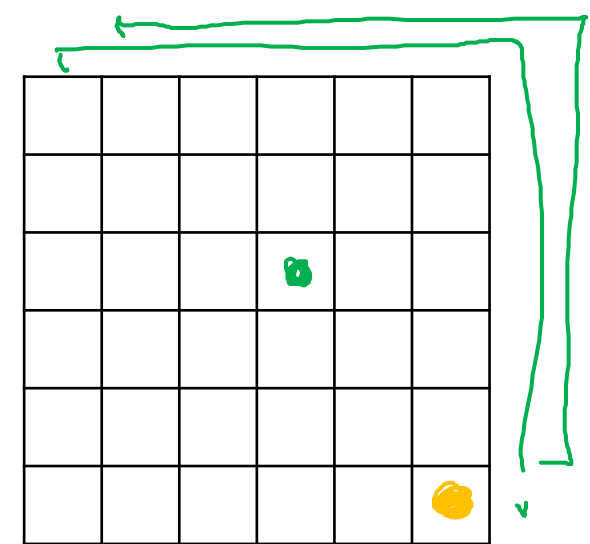
$1 \times 1 \times 32$

=

ReLU

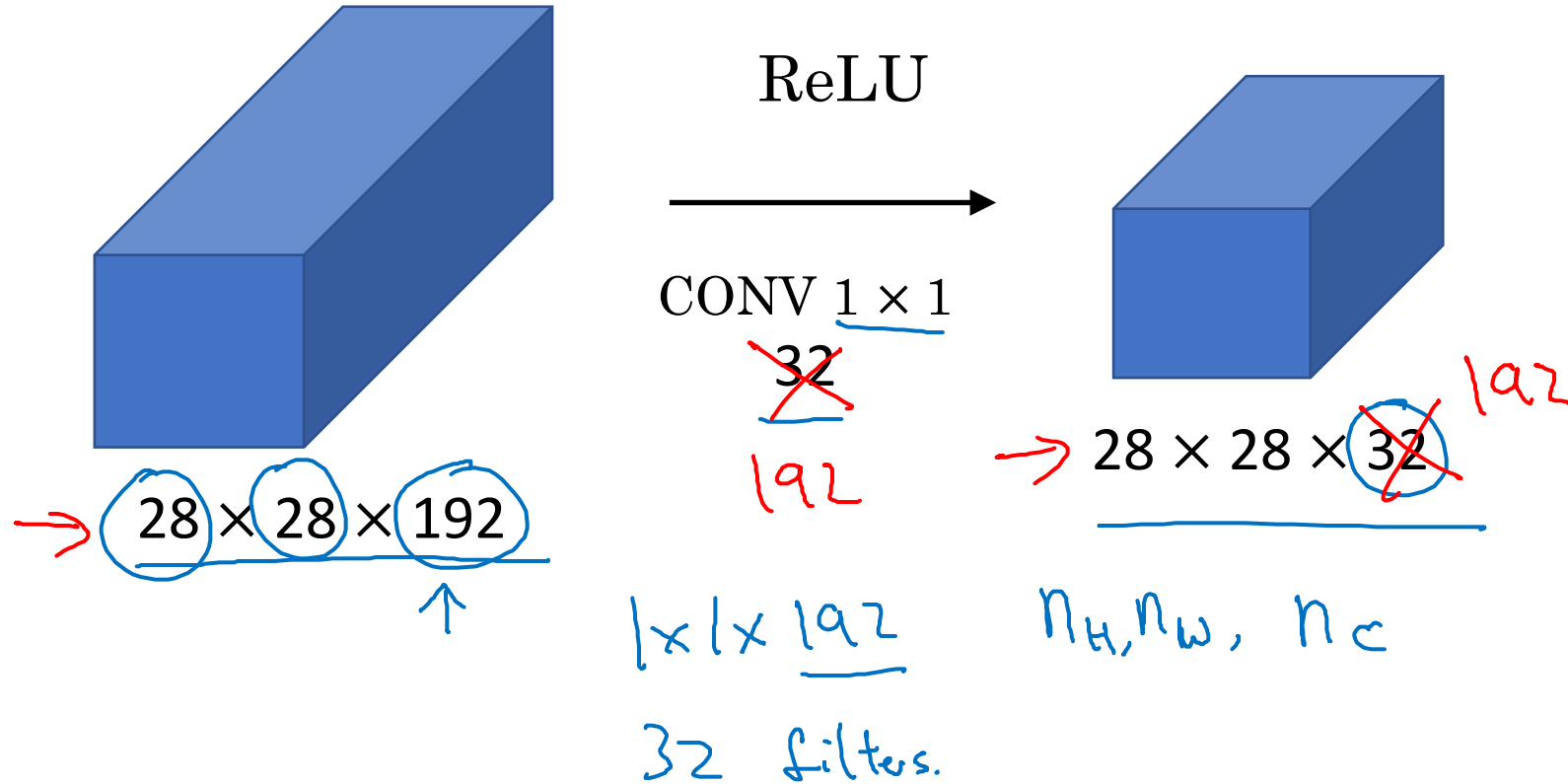
Network in
Network

2	4	6	...		



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

Using 1×1 convolutions



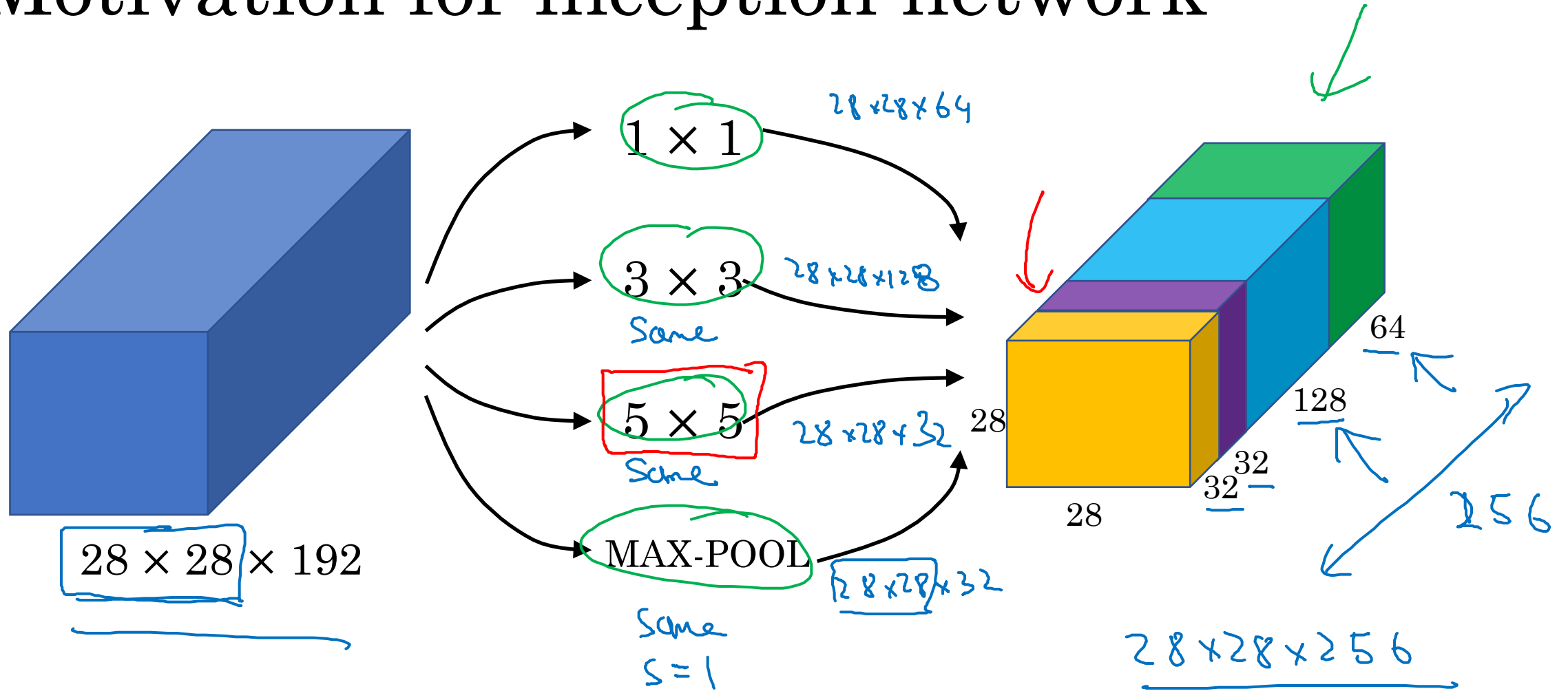


deeplearning.ai

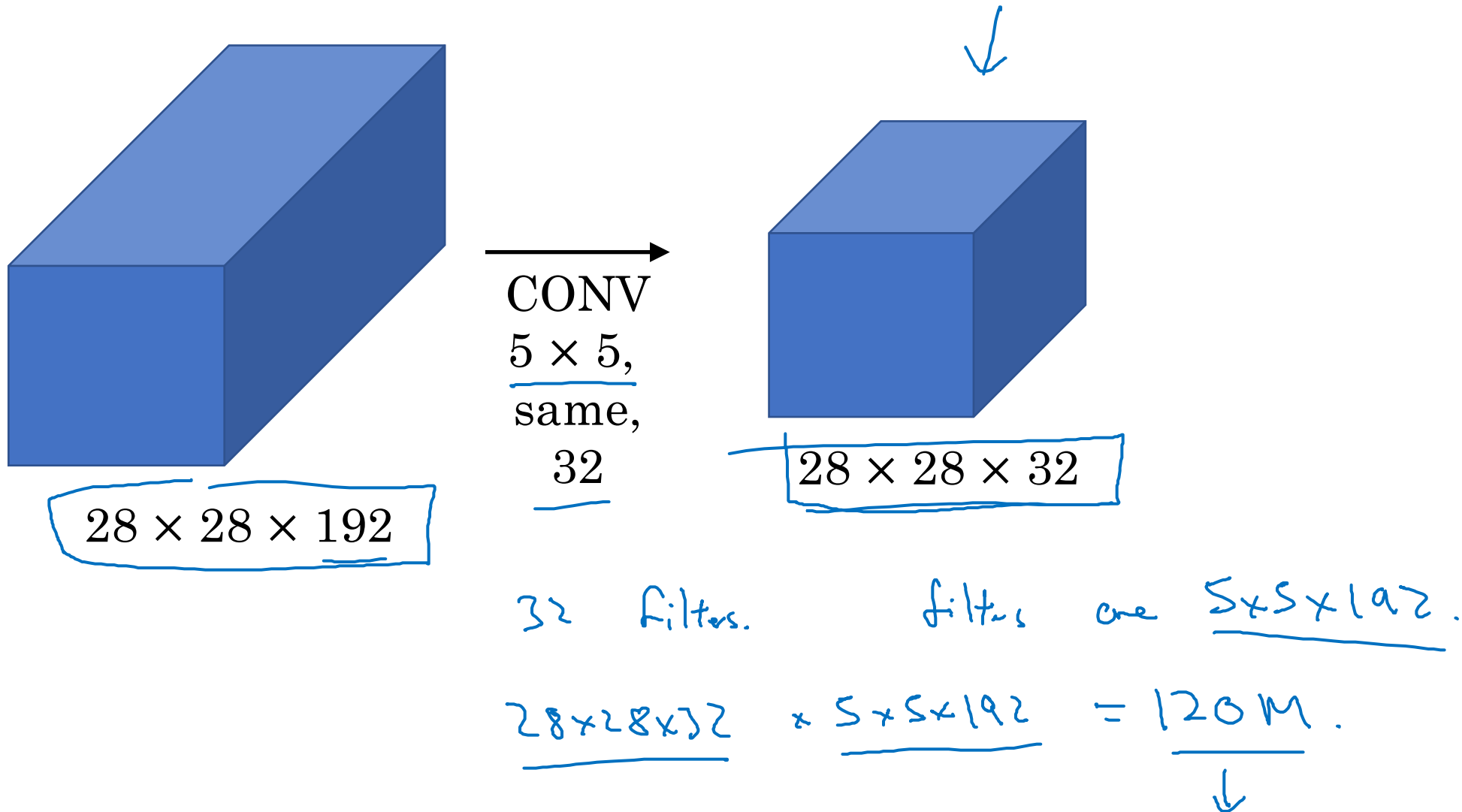
Case Studies

Inception network motivation

Motivation for inception network

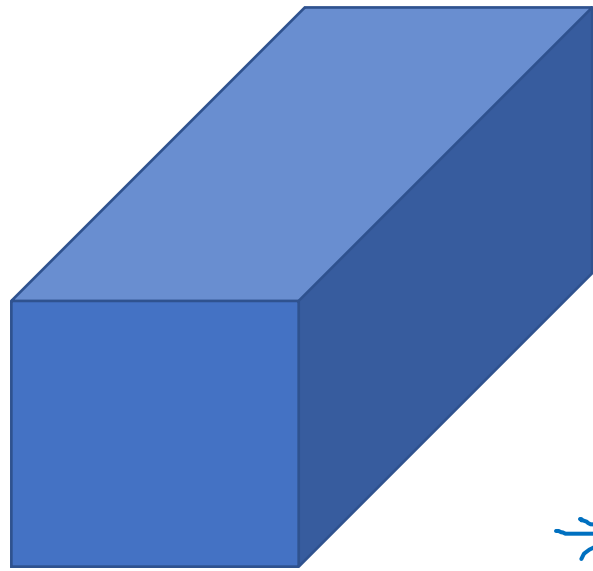
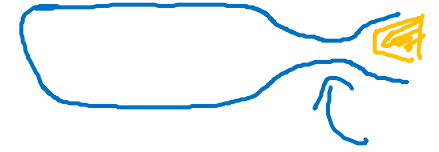


The problem of computational cost



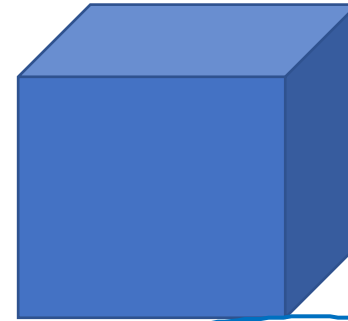
Using 1×1 convolution

"bottleneck layer"



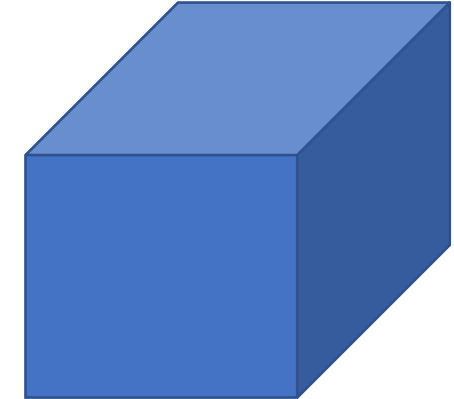
$$28 \times 28 \times 192$$

CONV
 1×1 ,
 $\rightarrow 16$,
 $\rightarrow 1 \times 1 \times 192$



$$28 \times 28 \times 16$$

CONV
 5×5 ,
32,
 $5 \times 5 \times 16$



$$28 \times 28 \times 32$$

$$28 \times 28 \times 16 \times 192 = 2.4M$$

$$28 \times 28 \times 32 \times 5 \times 5 \times 16 = 10.0M$$

$$12.4M$$

$$120M \rightarrow$$



deeplearning.ai

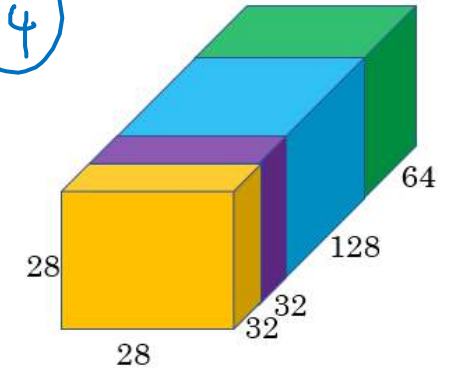
Case Studies

Inception network

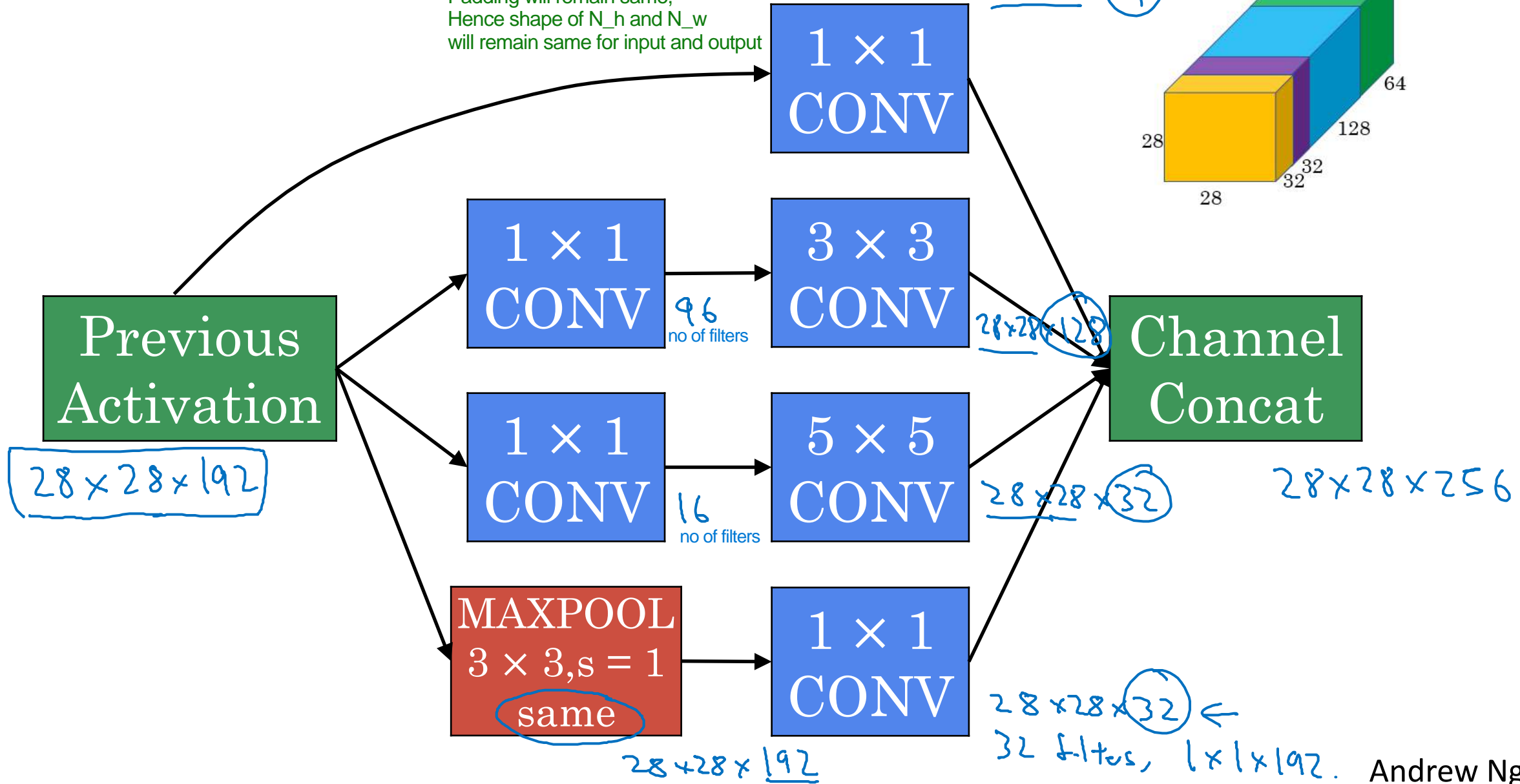
Inception module

SHAPE OF
OUTPUT BLOCKS

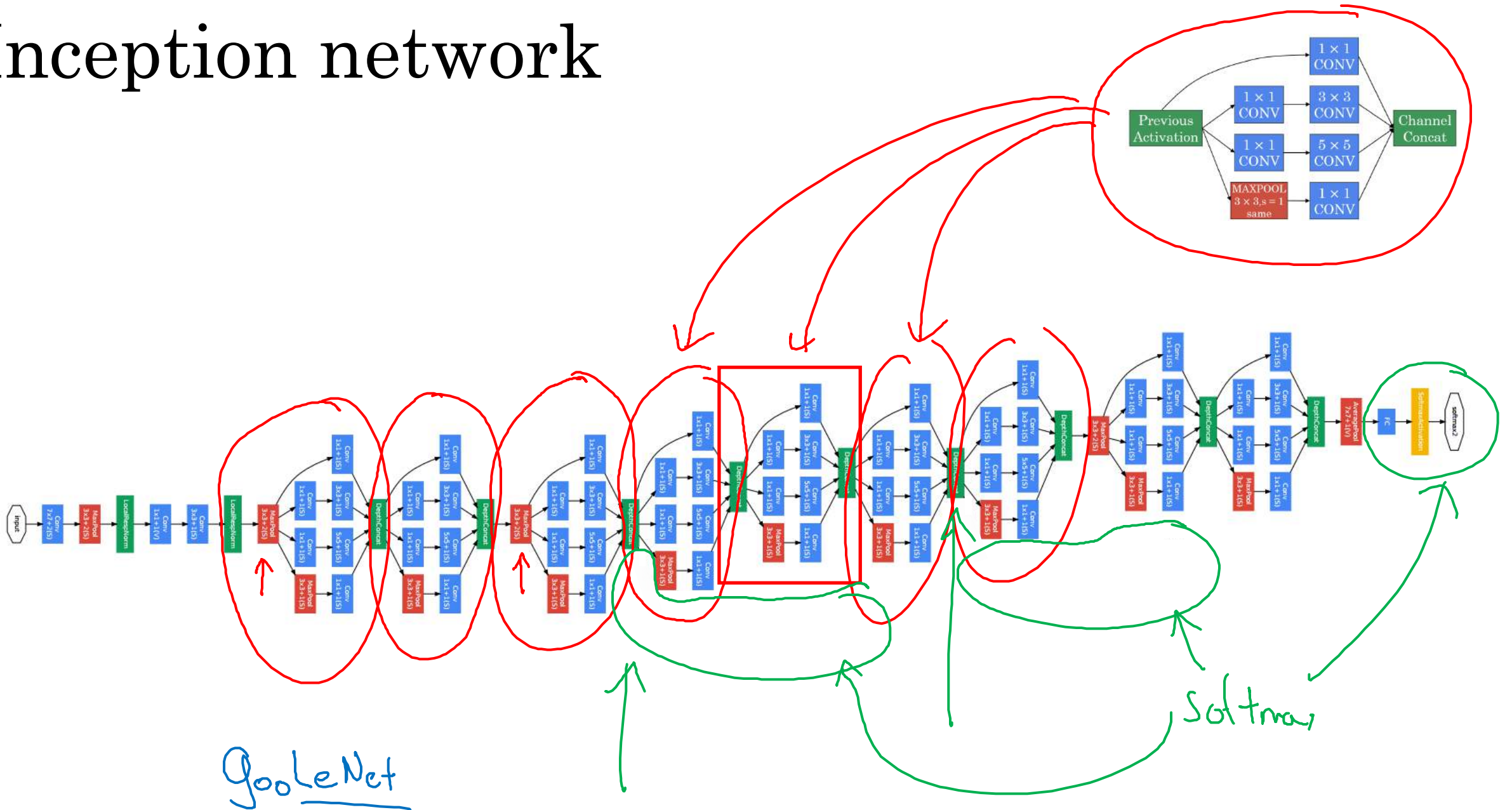
$$28 \times 28 \times 64$$



Padding will remain same,
Hence shape of N_h and N_w
will remain same for input and output



Inception network







deeplearning.ai

Convolutional Neural Networks

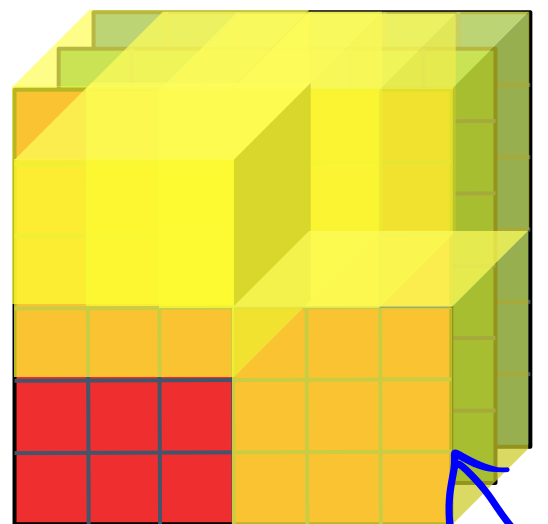
MobileNet

Motivation for MobileNets

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

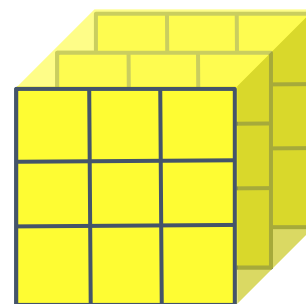


Normal Convolution



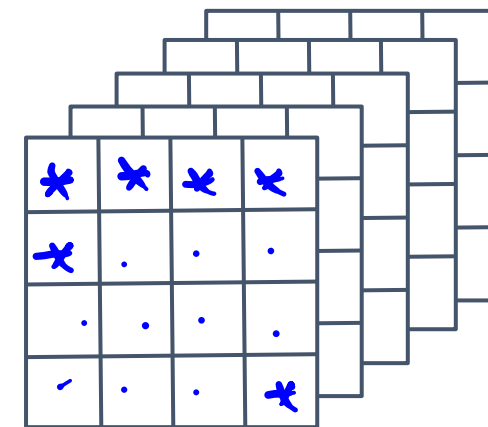
$6 \times 6 \times 3$
 $n \times n \times n_c$

*



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

=

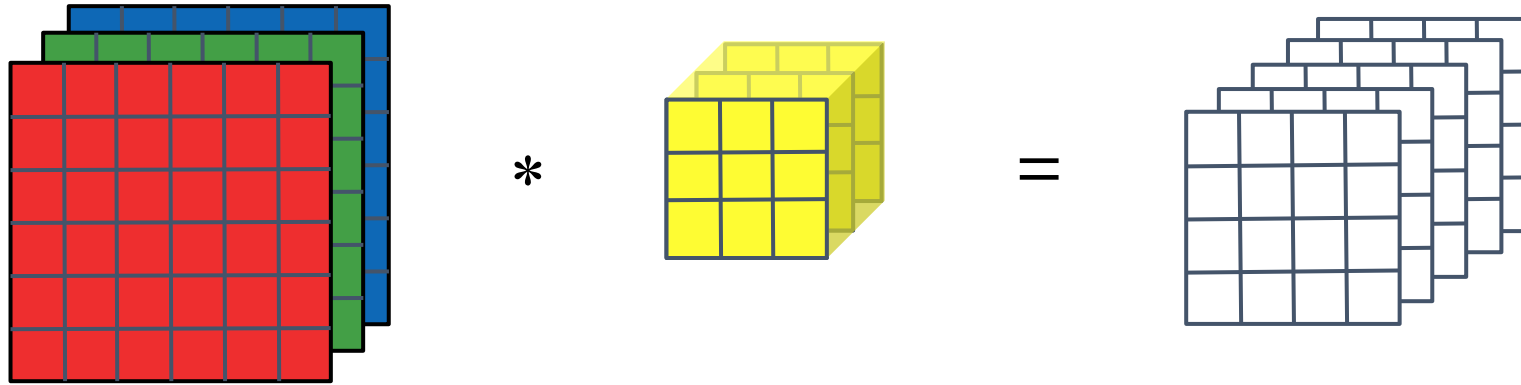


$4 \times 4 \times 5$
 $n_{out} \times n_{out} \times n_c'$

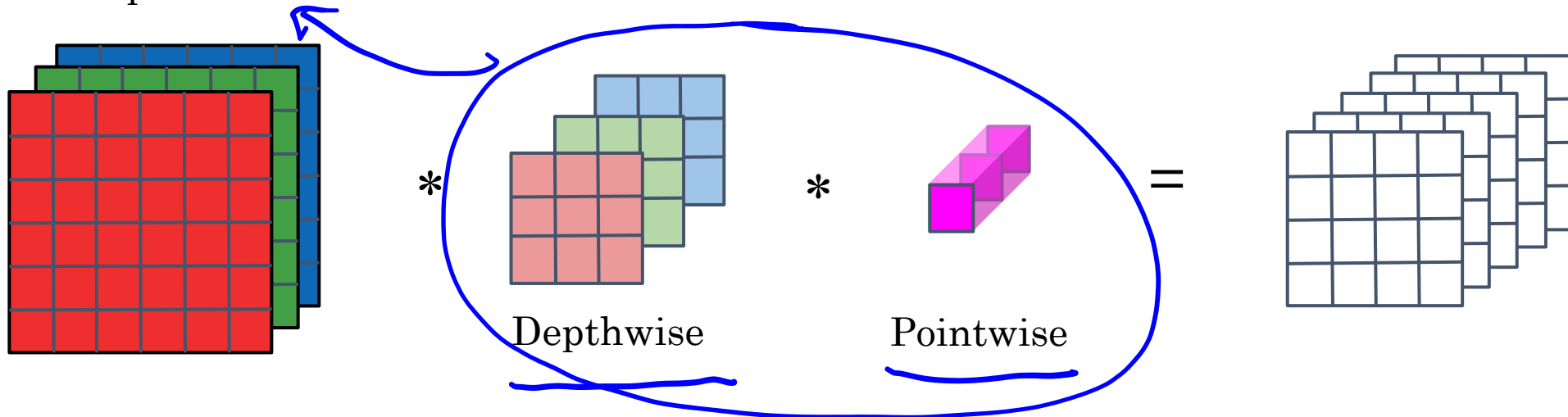
$$\begin{array}{ccccccc}
 \text{Computational cost} & = & \frac{\text{\#filter params}}{3 \times 3 \times 3} & \times & \frac{\text{\# filter positions}}{4 \times 4} & \times & \text{\# of filters} \\
 \rightarrow \underline{2160} & = & \uparrow & & \uparrow & & \uparrow \\
 & & & & & & 5
 \end{array}$$

Depthwise Separable Convolution

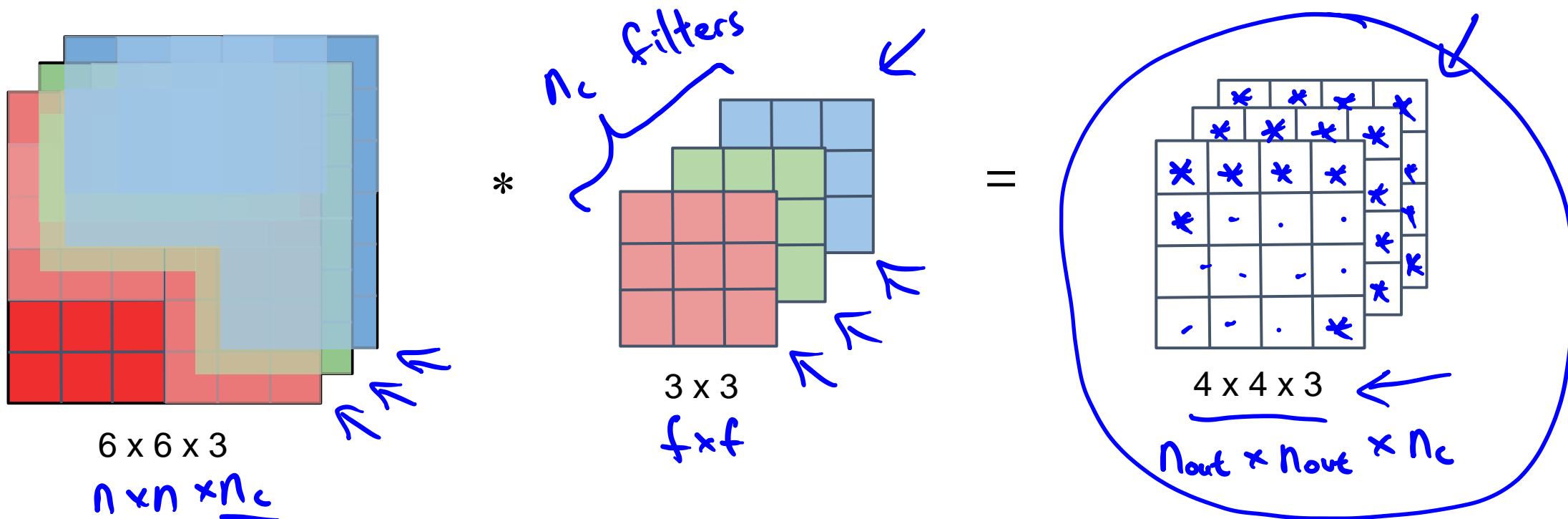
Normal Convolution



Depthwise Separable Convolution



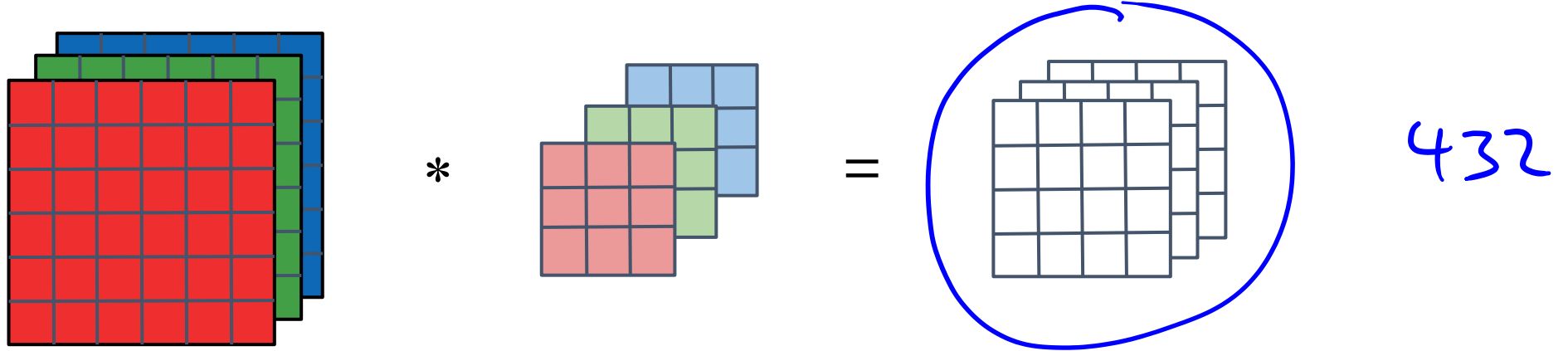
Depthwise Convolution



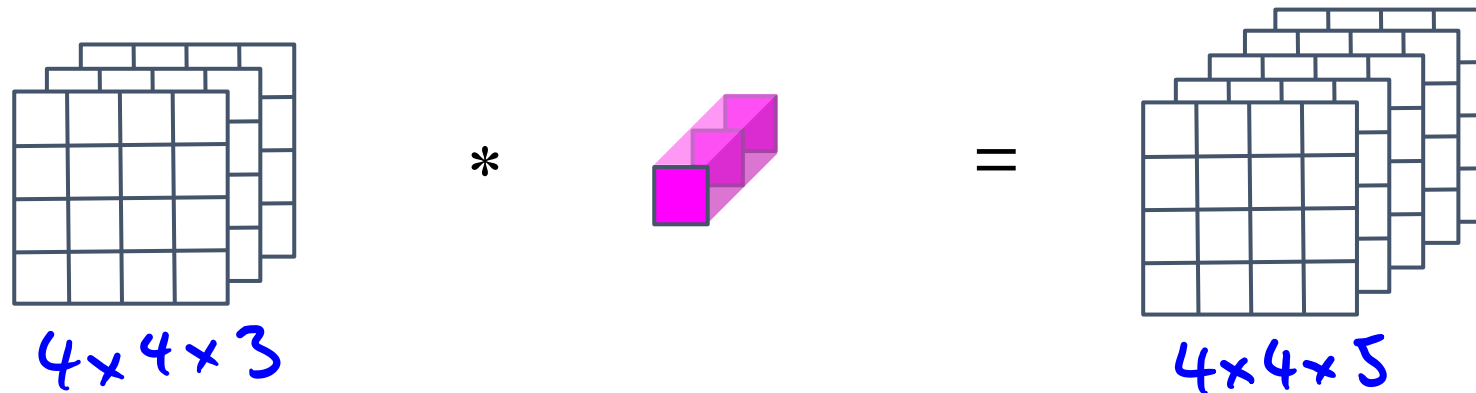
$$\begin{aligned}
 \text{Computational cost} &= \text{\#filter params} \times \text{\# filter positions} \times \text{\# of filters} \\
 432 &= \underbrace{3 \times 3} \times \underbrace{4 \times 4 \times 3}
 \end{aligned}$$

Depthwise Separable Convolution

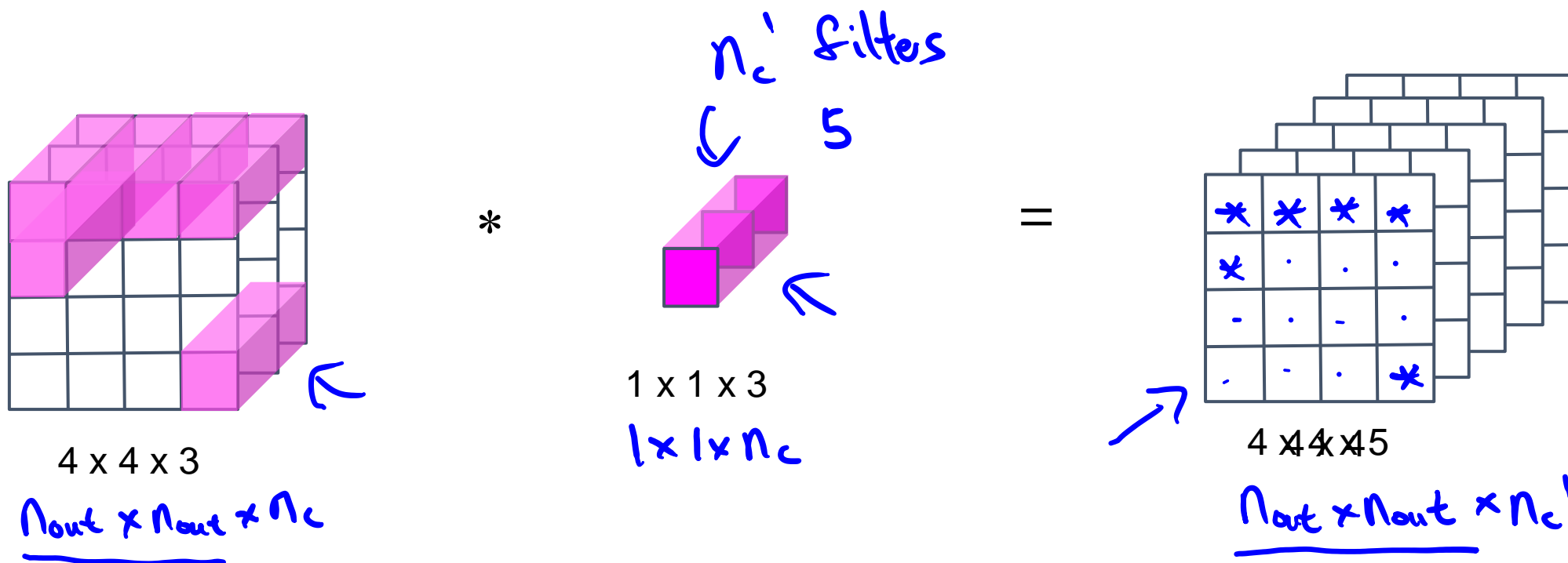
Depthwise Convolution



Pointwise Convolution



Pointwise Convolution

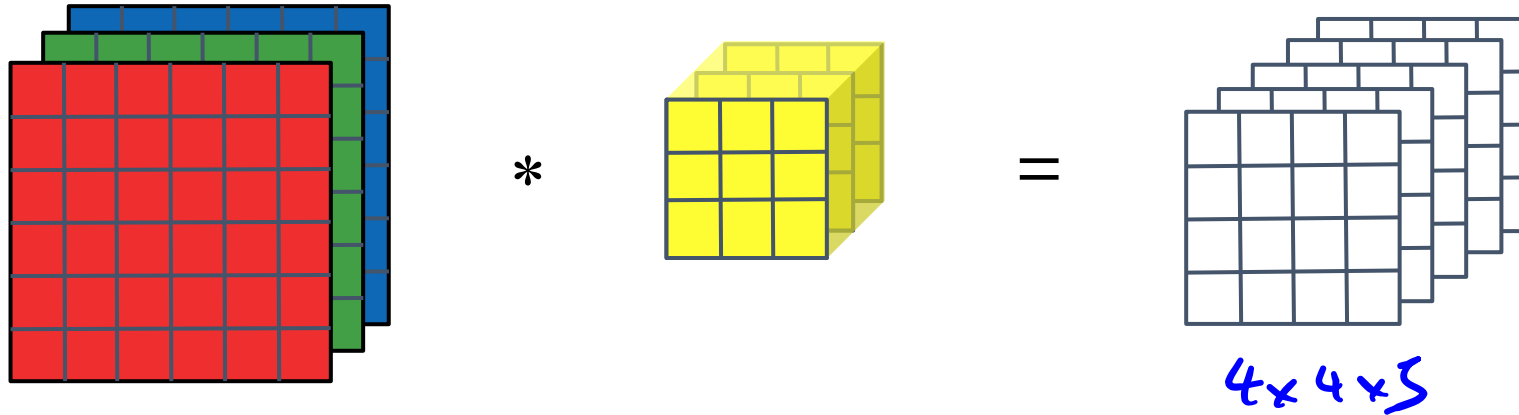


Computational cost = #filter params x # filter positions x # of filters

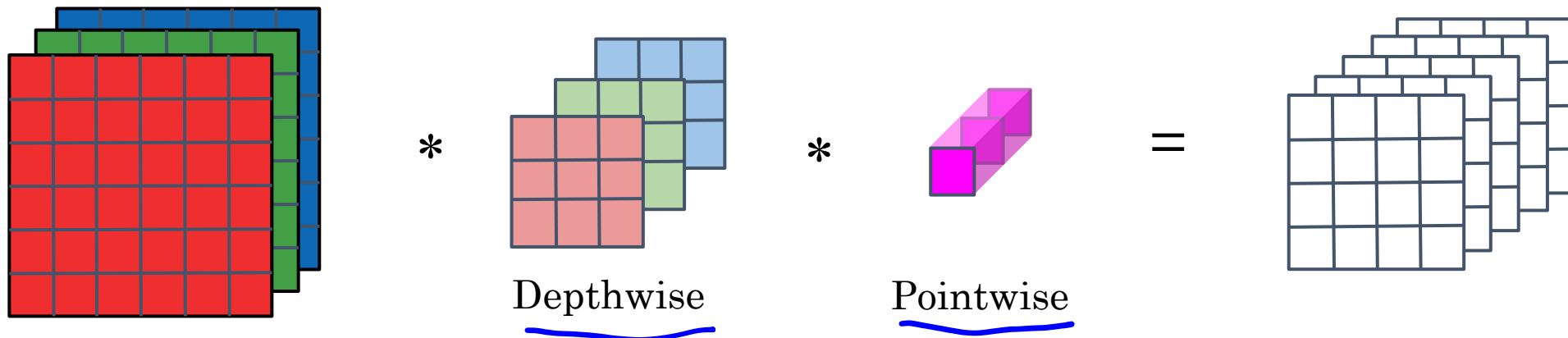
240 = $1 \times 1 \times 3$ x 4×4 x 5

Depthwise Separable Convolution

Normal Convolution



Depthwise Separable Convolution



Cost Summary

Cost of normal convolution

2160

Cost of depthwise separable convolution

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

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

$$= \frac{1}{\textcircled{n_c}} + \frac{1}{f^2}$$
$$\frac{1}{5} + \frac{1}{9}$$

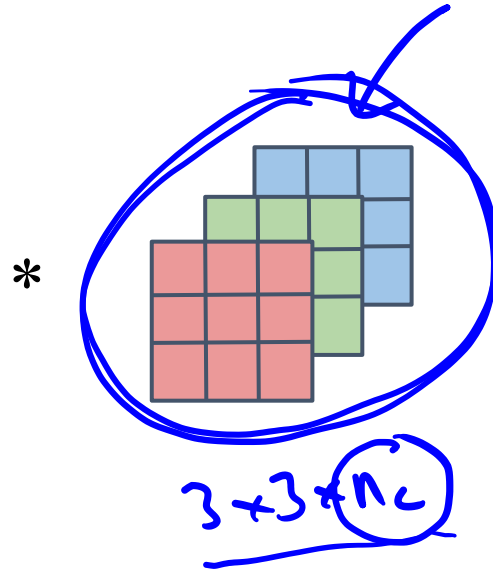
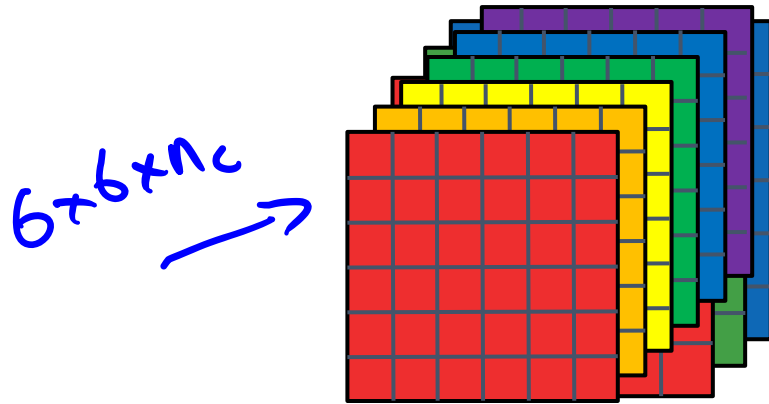
$$= \frac{1}{512} + \frac{1}{3^2}$$

↑ ↑

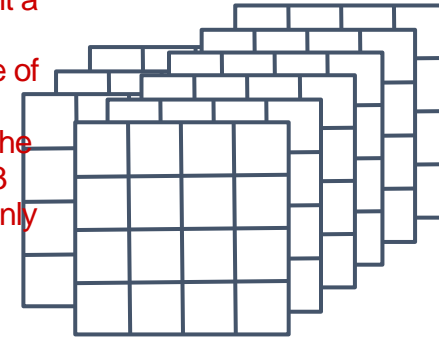
~10 times cheaper

Depthwise Separable Convolution

Depthwise Convolution

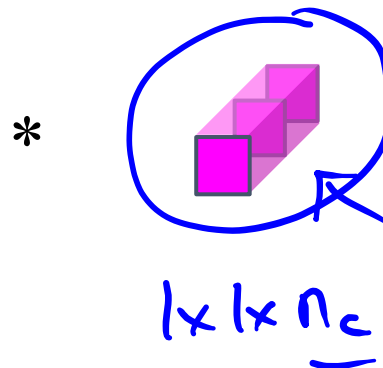
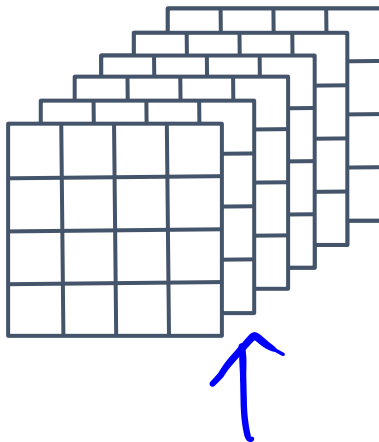


Sir has used this filter just to represent a filter here, irrespective of the input channels, the filter have 3 channels only

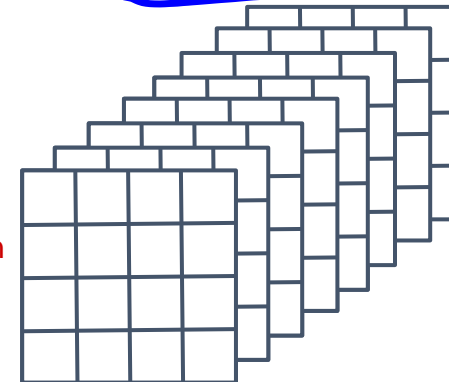


$4 \times 4 \times n_c$

Pointwise Convolution



Similarly for this 1 X 1 filter that is used for representation purpose only irrespective of the input layer



$4 \times 4 \times n_c$



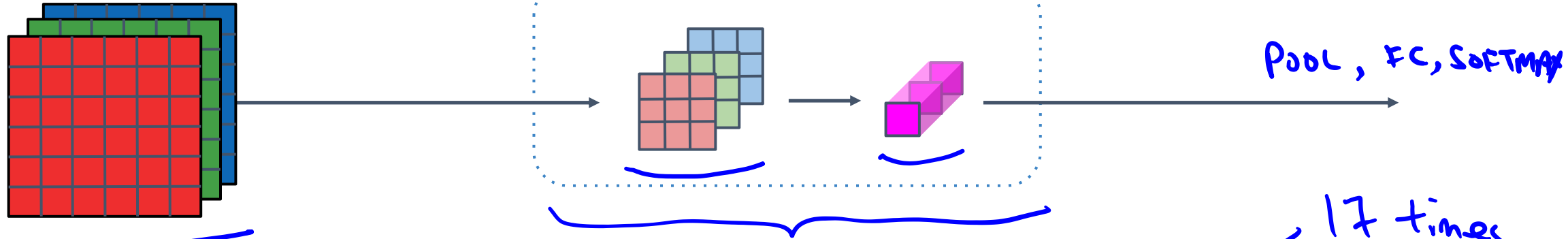
deeplearning.ai

Convolutional Neural Networks

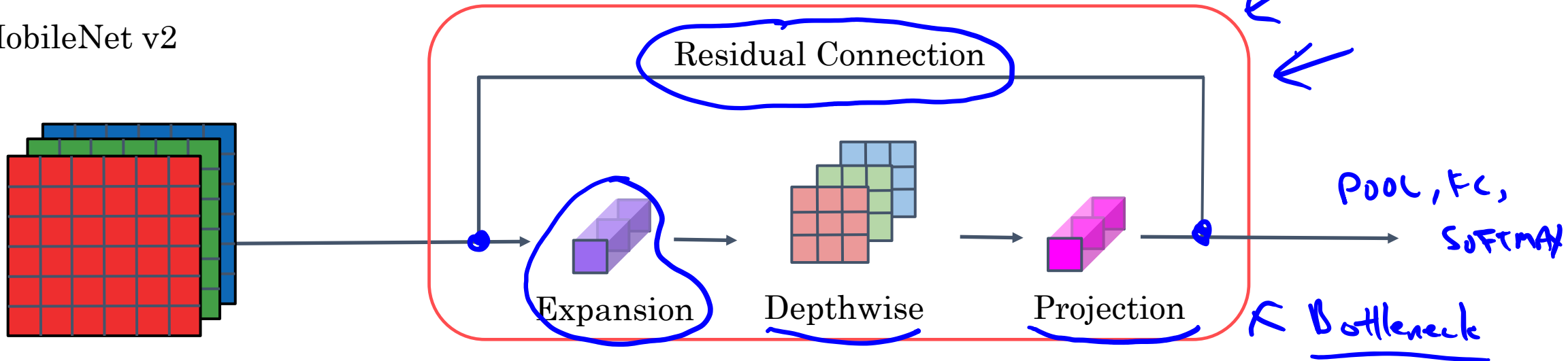
MobileNet Architecture

MobileNet

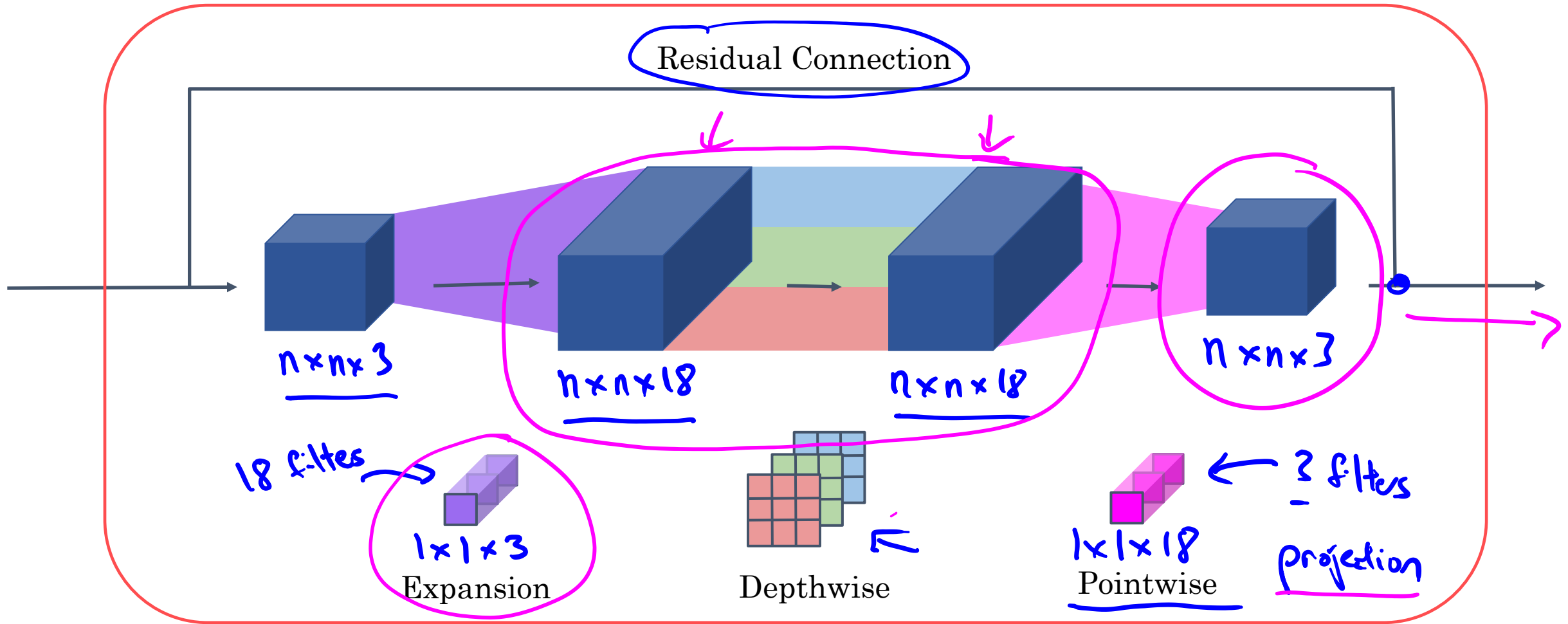
MobileNet v1



MobileNet v2

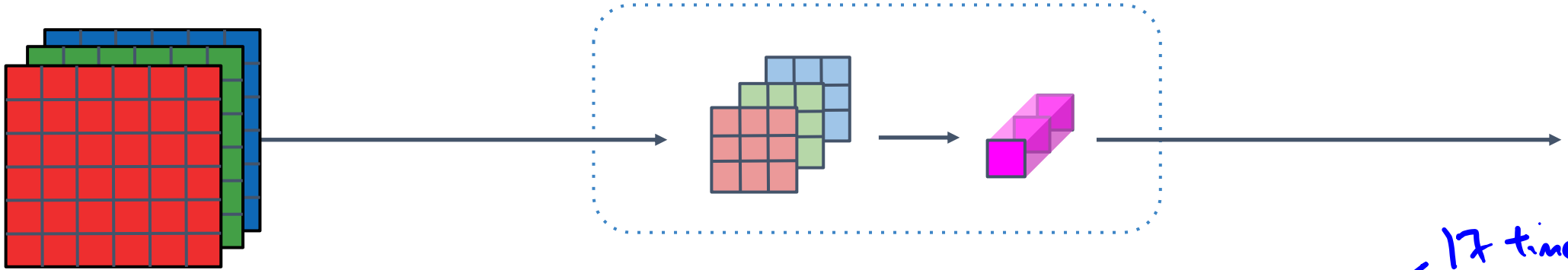


MobileNet v2 Bottleneck

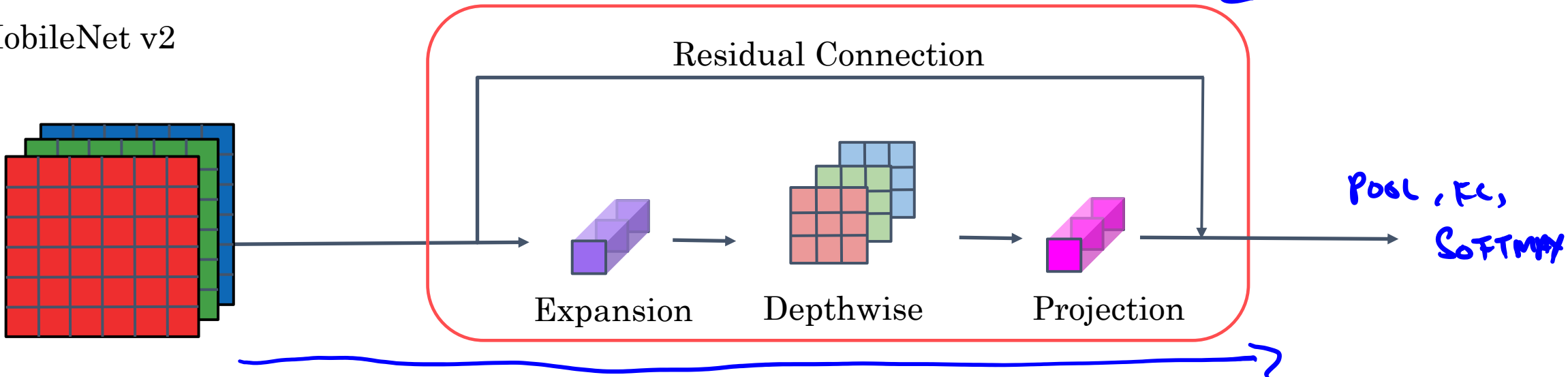


MobileNet

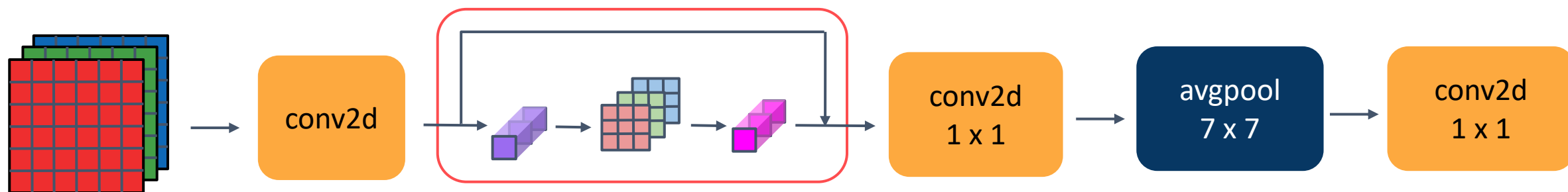
MobileNet v1



MobileNet v2



MobileNet v2 Full Architecture






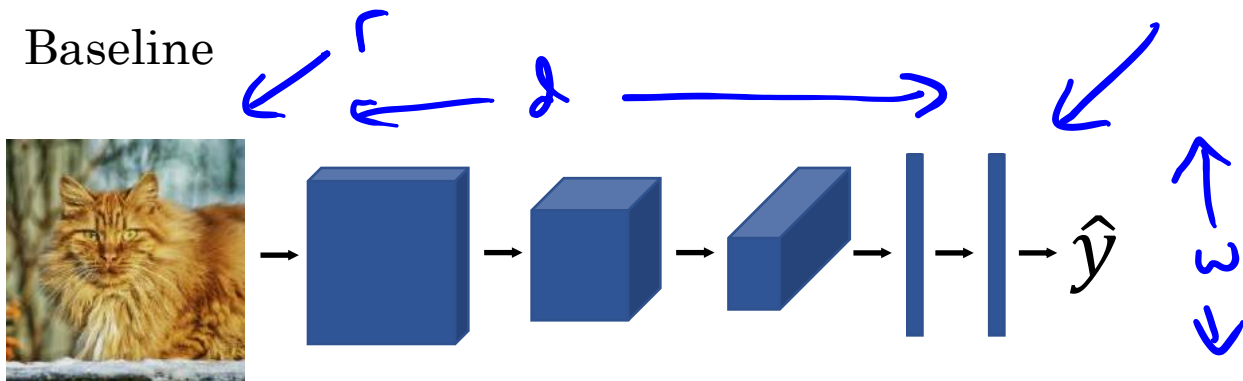
deeplearning.ai

Convolutional Neural Networks

EfficientNet

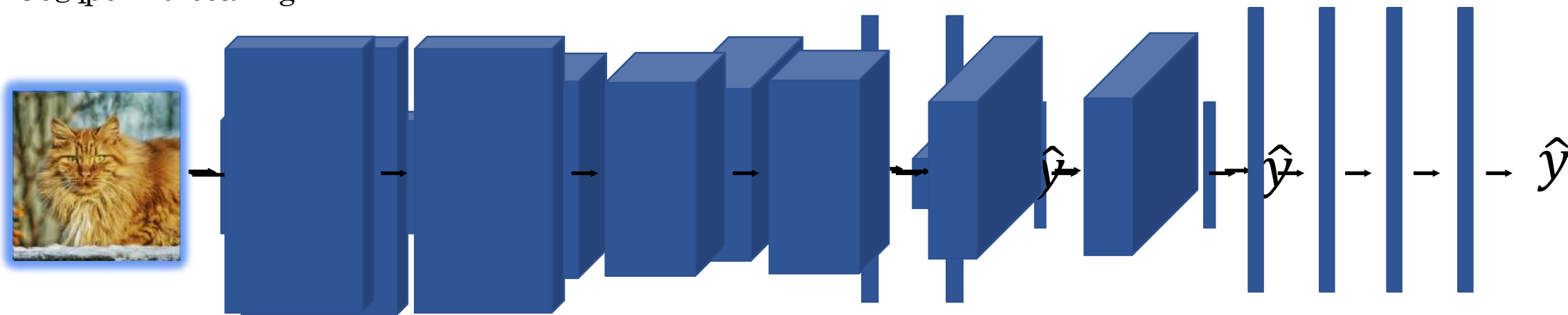
EfficientNet

Baseline



A diagram showing a vertical oval representing a 3D volume. Three arrows point from the left towards the oval, each aligned with a label inside: 'r' at the top, 'd' in the middle, and 'w' at the bottom. To the right of the oval, the words 'resolution', 'depth', and 'width' are written, corresponding to 'r', 'd', and 'w' respectively.

Widener Rescuing



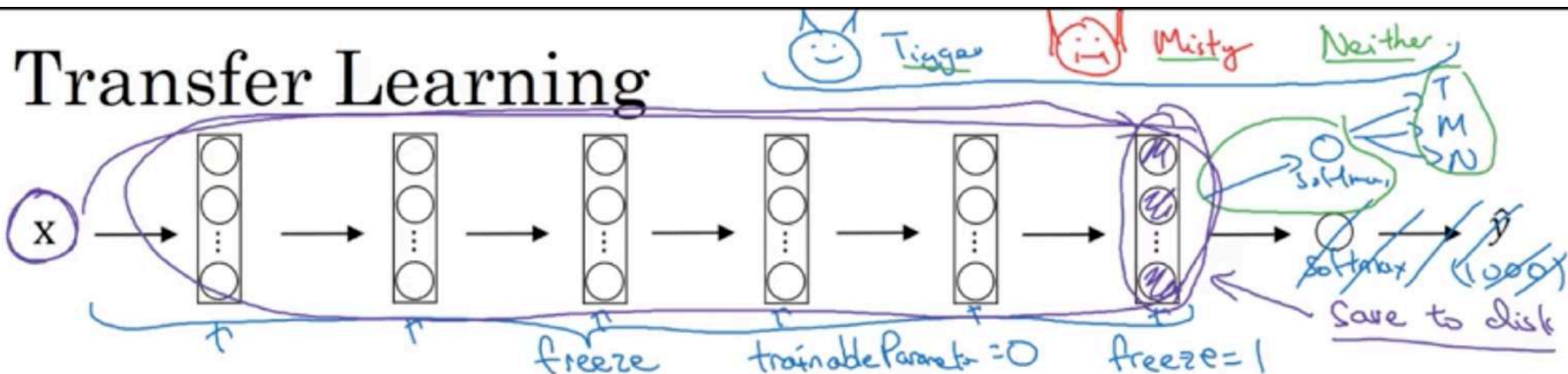


deeplearning.ai

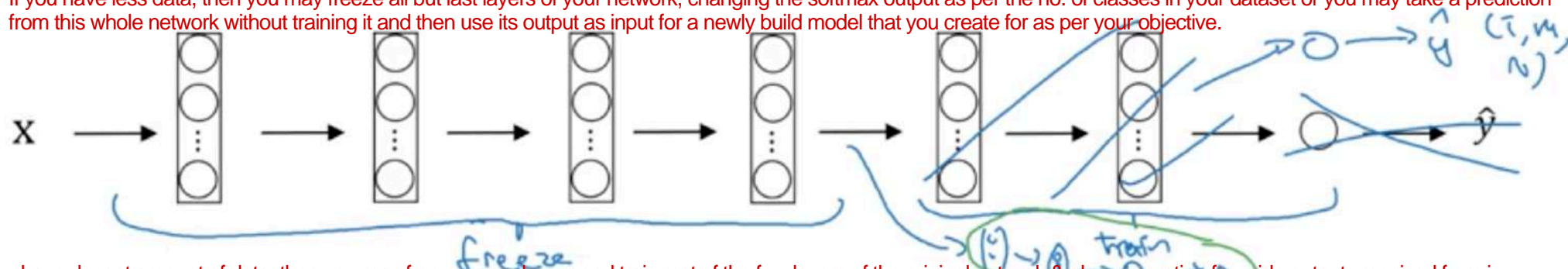
Practical advice for using ConvNets

Transfer Learning

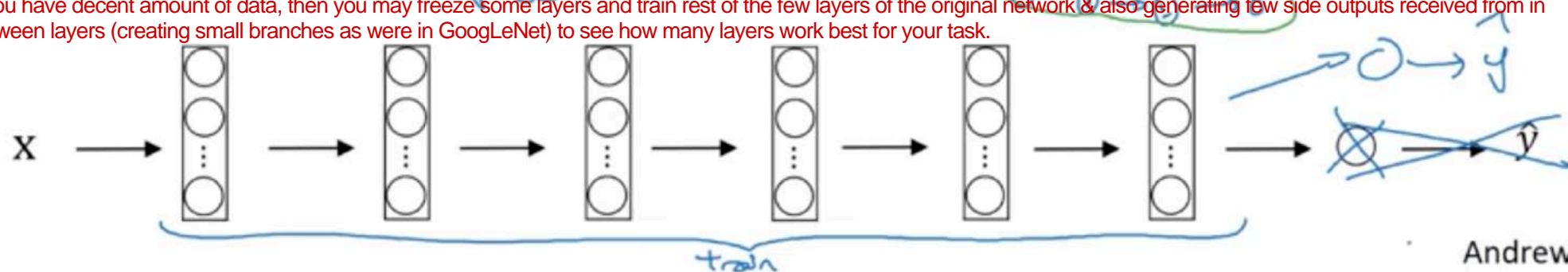
Transfer Learning



If you have less data, then you may freeze all but last layers of your network, changing the softmax output as per the no. of classes in your dataset or you may take a prediction from this whole network without training it and then use its output as input for a newly build model that you create for as per your objective.



If you have decent amount of data, then you may freeze some layers and train rest of the few layers of the original network & also generating few side outputs received from in between layers (creating small branches as were in GoogLeNet) to see how many layers work best for your task.



Andrew Ng

If you have huge amount of data then you can just use the pretrained model architecture, use it as an initializer and train your dataset using all the layers of the original architecture model.



deeplearning.ai

Practical advice for using ConvNets

Data augmentation

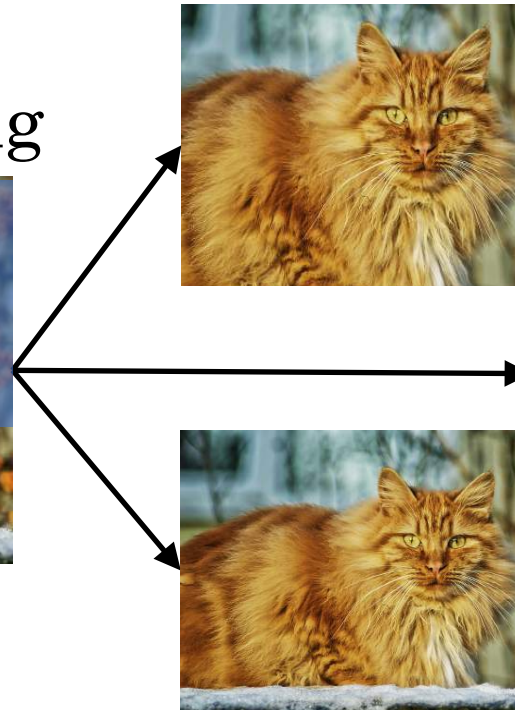
Common augmentation method

Mirroring



y

Random Cropping

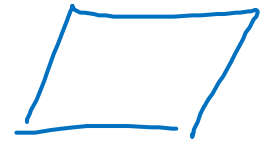


Rotation

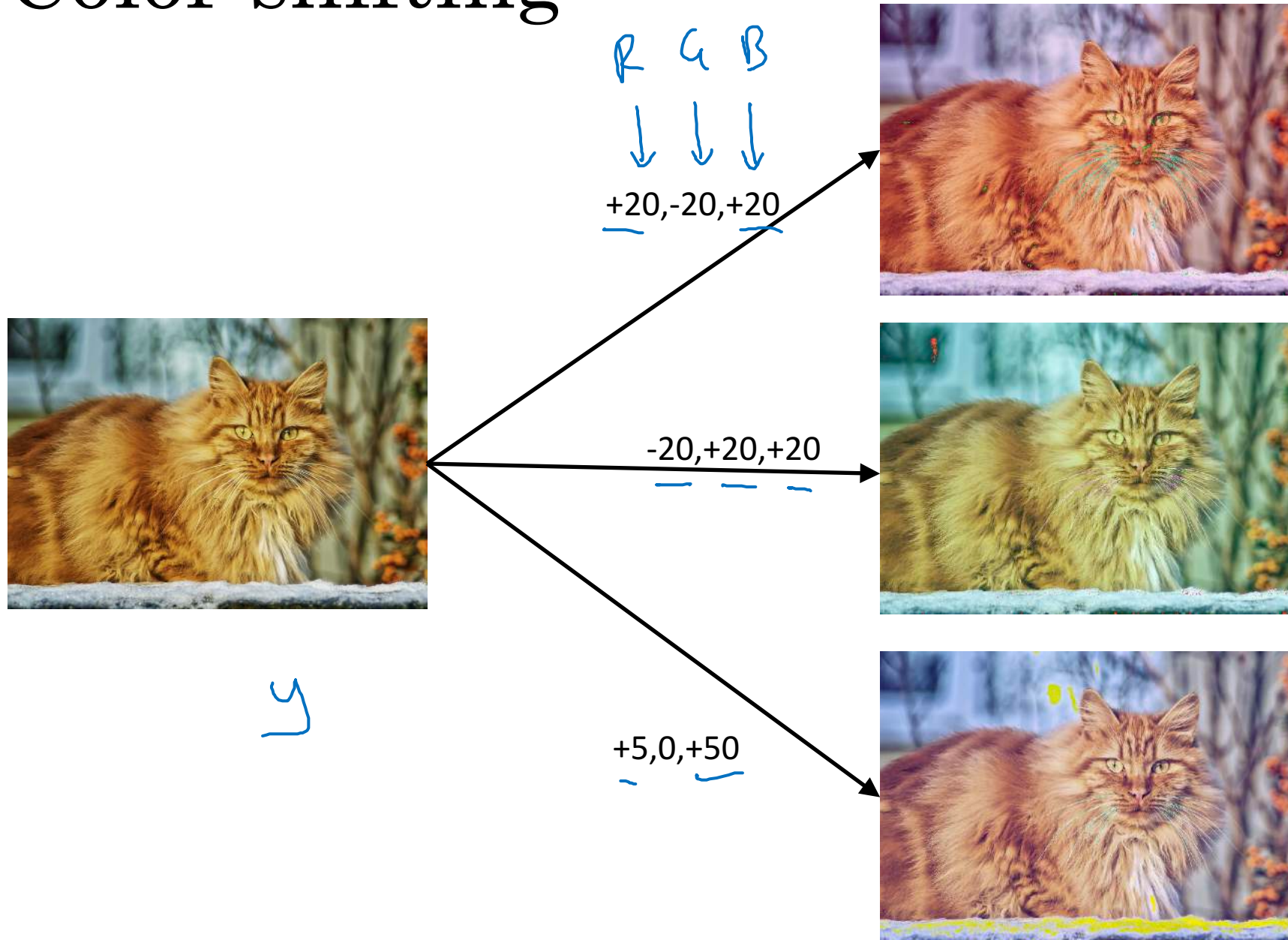
Shearing

Local warping

...



Color shifting



Advanced:

PCA

ml-class.org

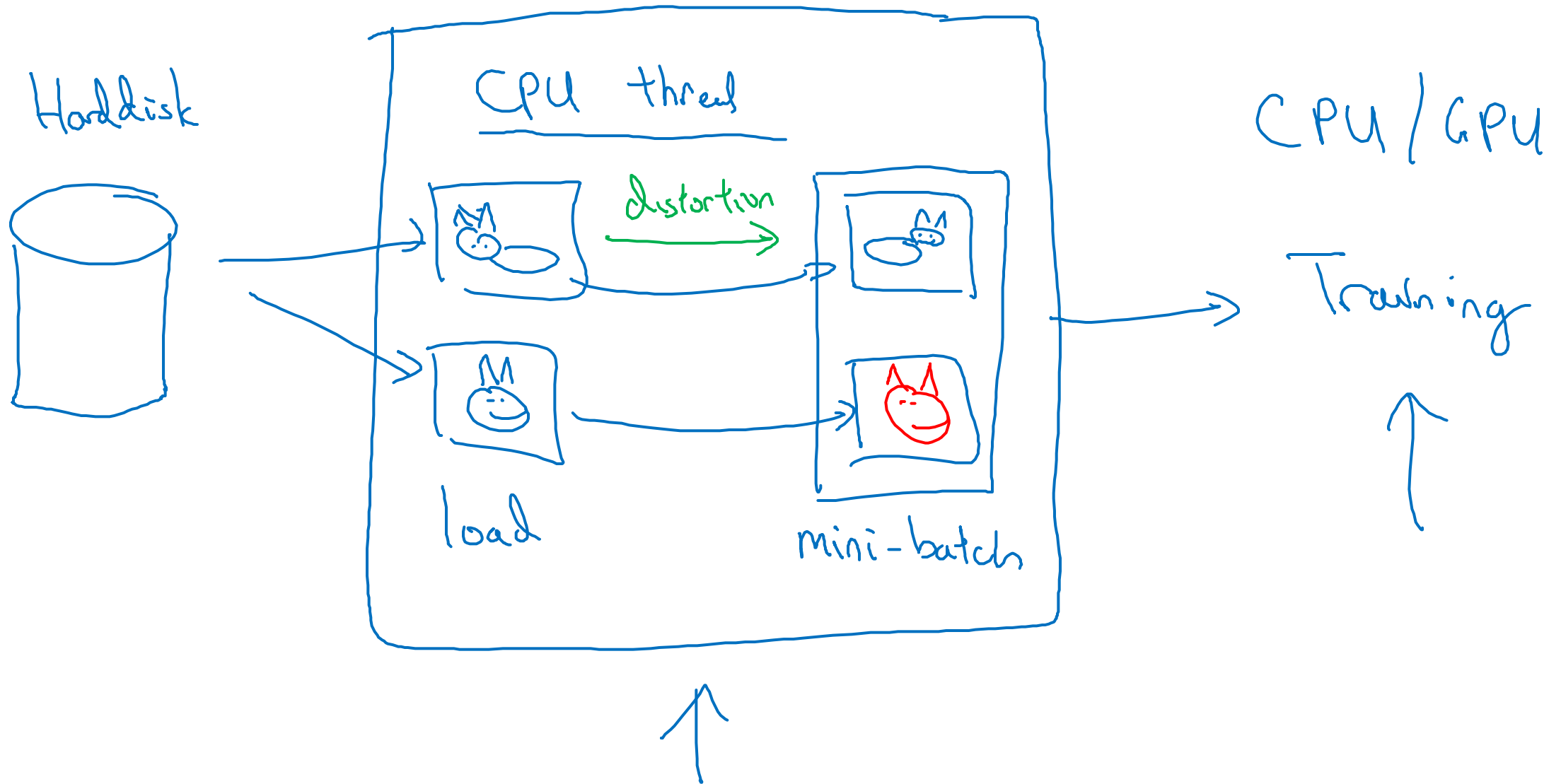
[AlexNet paper

["PCA color augmentation."

R B

G

Implementing distortions during training



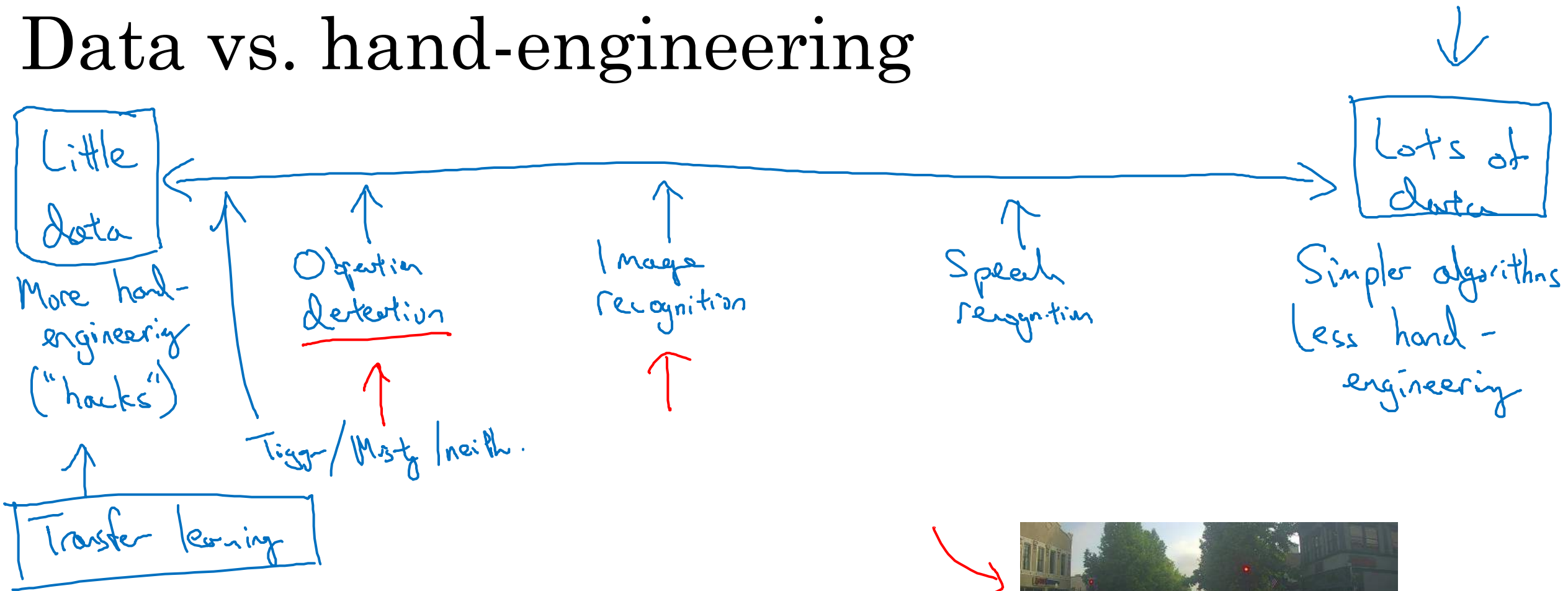


deeplearning.ai

Practical advice for using ConvNets

The state of computer vision

Data vs. hand-engineering



Two sources of knowledge

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



Tips for doing well on benchmarks/winning competitions

3-15 networks

→ \hat{y}

Ensembling

- 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