deeplearning.ai

Case Studies

Why look at case studies?

Outline

Classic networks:

- LeNet-5
- AlexNet
- VGG

ResNet (152)

Inception

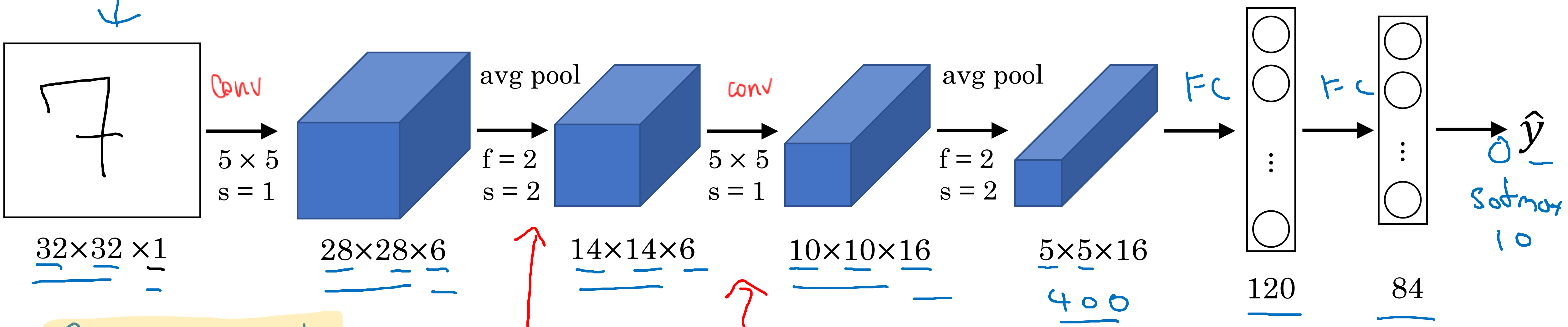
deeplearning.ai

Case Studies

Classic networks

LeNet - 5

- One of the first ones
- For hand-written digit recognition



60K parameters

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

non-linearity after pooling

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

conv pool conv pool fc fc output

Advanced: sigmoid/tanh $\quad$ ReLU

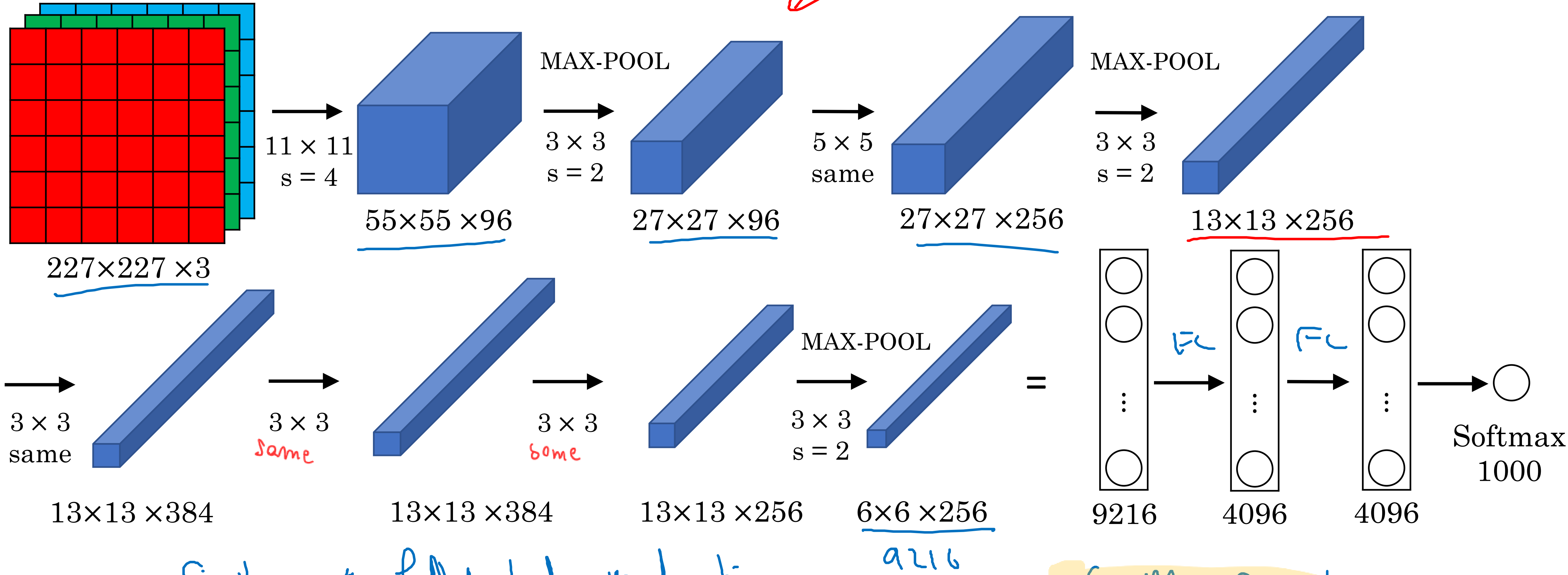
$\boxed{\text{II}}$ $\quad$ III.

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

Andrew Ng

AlexNet

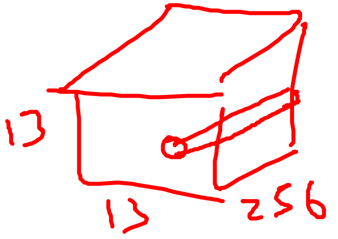
• Inspired by LeNet-5, with 1k-times more parameters



~60M parameters

- Similar to LeNet, but much bigger.
- ReLU
- Multiple GPUs.

(not used today) – Local Response Normalization (LRN)



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

Andrew Ng

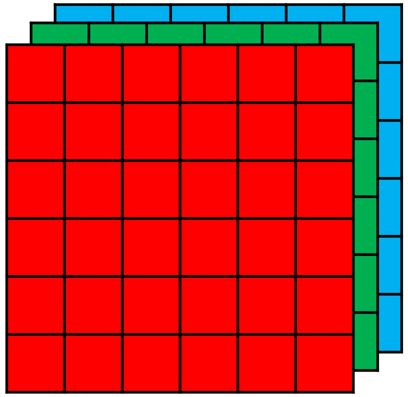
VGG - 16

VGG-19

- Relatively uniform architecture compared to the previous ones. Only uses CONV(3x3) and MAXPOOL(2x2) operations. But they do it many times.
- Activations increase in depth by a factor of 2 due to conv and shrink in size by a factor of 2 due to pooling

224×224×64 224×224×64 224×224×64

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

- Motivation: adding too many layers can worsen the training error. This is because redundant layers should learn an identity function, which is hard in general.
- This problem can be solved by adding "skip-connections" that forward the activation of one layer to the linear operation of a later layer. This solves the problem because redundant blocks can easily learn the identity function by setting their weights to zero.

$$\rightarrow a^{[l]} \rightarrow a^{[l+1]} \oplus, \; a^{[l+2]} \rightarrow$$

RESIDUAL BLOCK

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

The residual block can implement identity by setting $W^{[l+2]}, b^{[l+2]}$ to zero.

Case Studies

The group of layers included within the skip connection is defined "residual block".

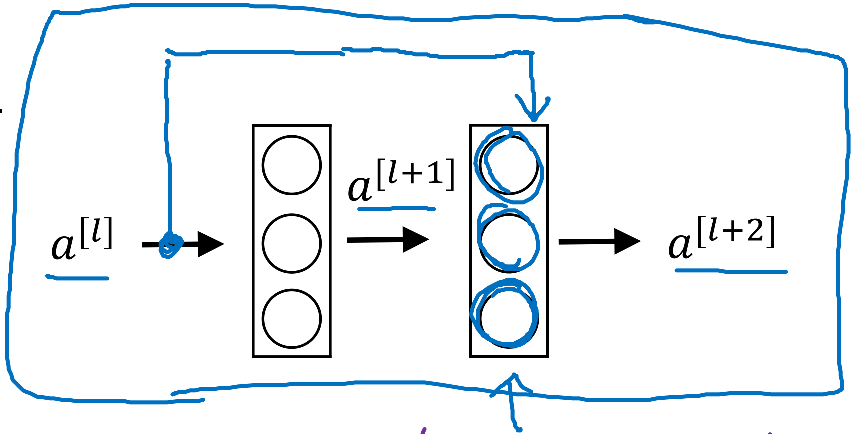
- ResNets are very deep networks containing residual blocks.

Residual Networks (ResNets)

deeplearning.ai

In general the activations of all layers within a residual block have the same dimension. However, it is possible to have different dimensions if an intermediate "conjunction" matrix is used to sum $a^{[l]}$ to $z^{[l+2]}$.

Residual block



"short cut" / skip connection

$a^{[l]} \to$ Linear $\to$ ReLU $\xrightarrow{a^{[l+1]}}$ Linear $\to \oplus \to$ ReLU $\to a^{[l+2]}$

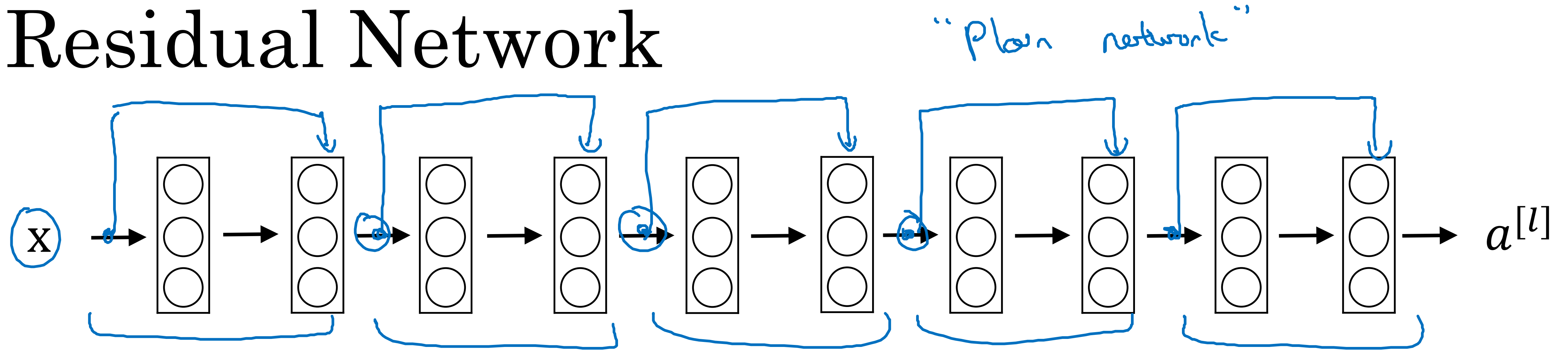
"main path"

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

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

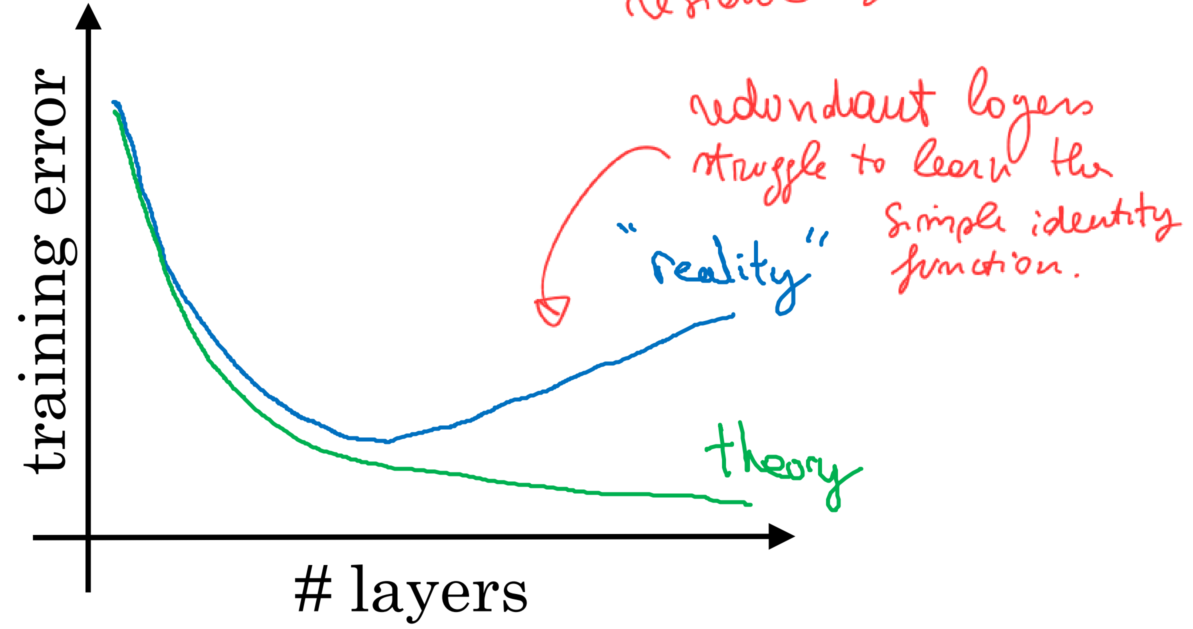
[He et al., 2015. Deep residual networks for image recognition]

Residual Network

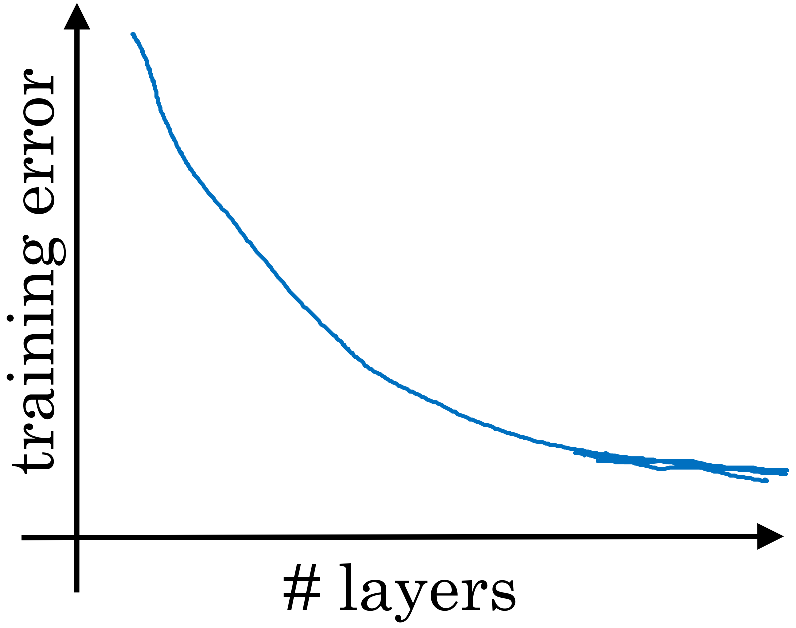


residual blocks

Plain



ResNet



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

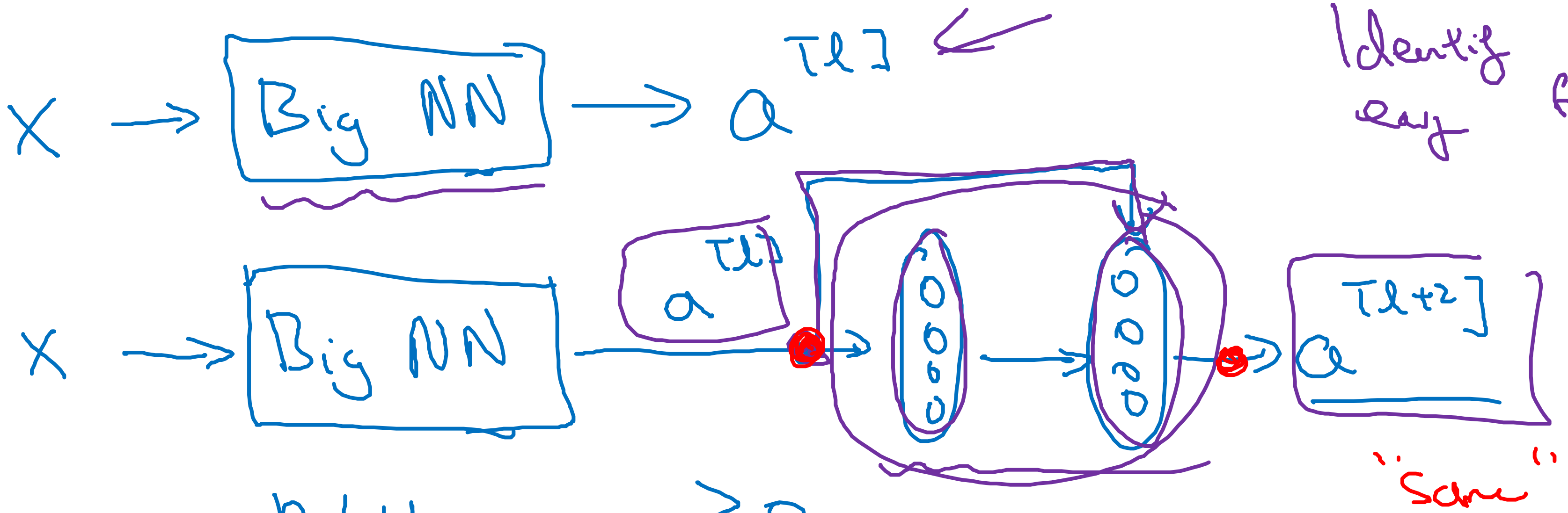
Andrew Ng

Case Studies

deeplearning.ai

Why ResNets work

Why do residual networks work?



Identity function is easy for Residual block to learn!

to join activations with different sizes.

ReLU. $a \geq 0$

$$\underset{256}{a^{[l+2]}} = g(z^{[l+2]} + a^{[l]})$$

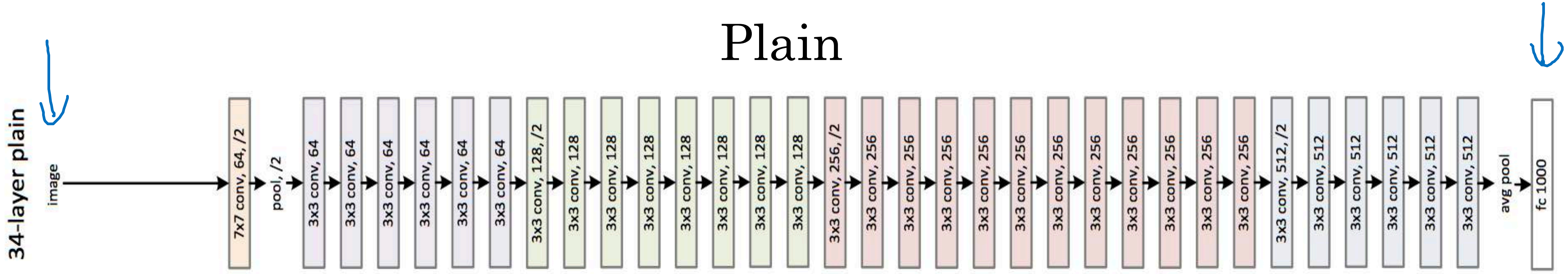
$$= g(\cancel{W^{[l+2]} a^{[l+1]} + b^{[l+2]}} + W_s \underset{128}{a^{[l]}}) = g(a^{[l]}) = a^{[l]}$$

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

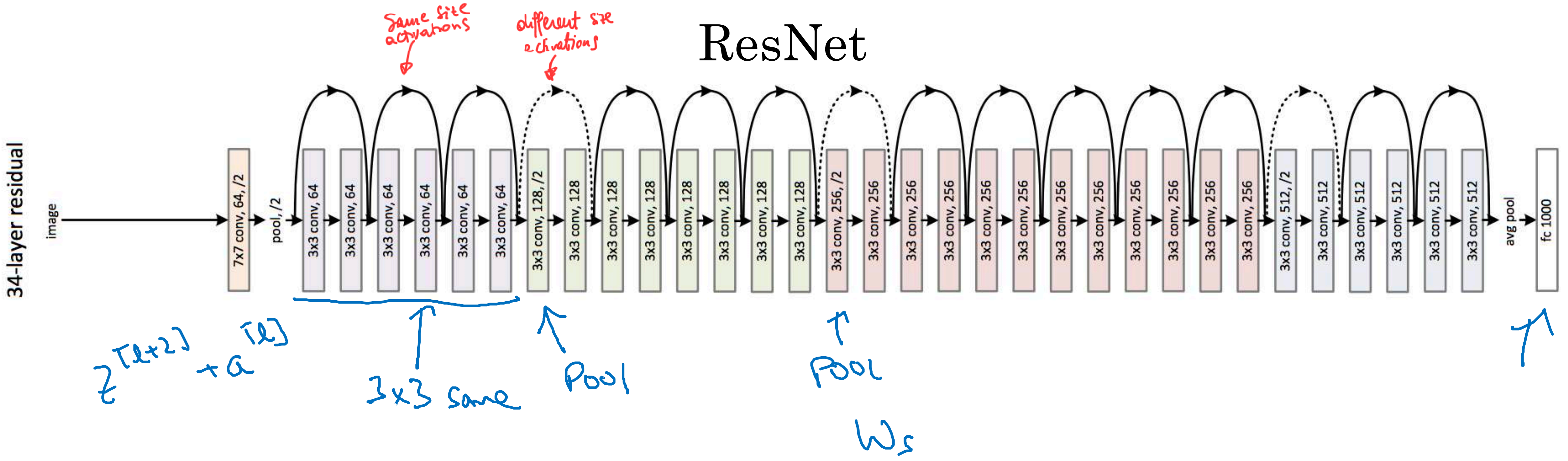
If $W^{[l+2]} = 0$, $b^{[l+2]} = 0$

ResNet

Plain



ResNet



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

Andrew Ng

• Applies a nontrivial function of the channels*

→ Used to shrink the number of channels in a nontrivial way (it's like a fancy version of pooling). This is especially useful for inception networks that tend to have extremely high number of channels.

Case Studies

deeplearning.ai

Network in Network and 1×1 convolutions

*• AKA: "Network in network" because convolving all the C channels of 1 neuron with a 1×1×C filter corresponds to a FC layer taking the C channels as inputs.

Why does a 1×1 convolution do?

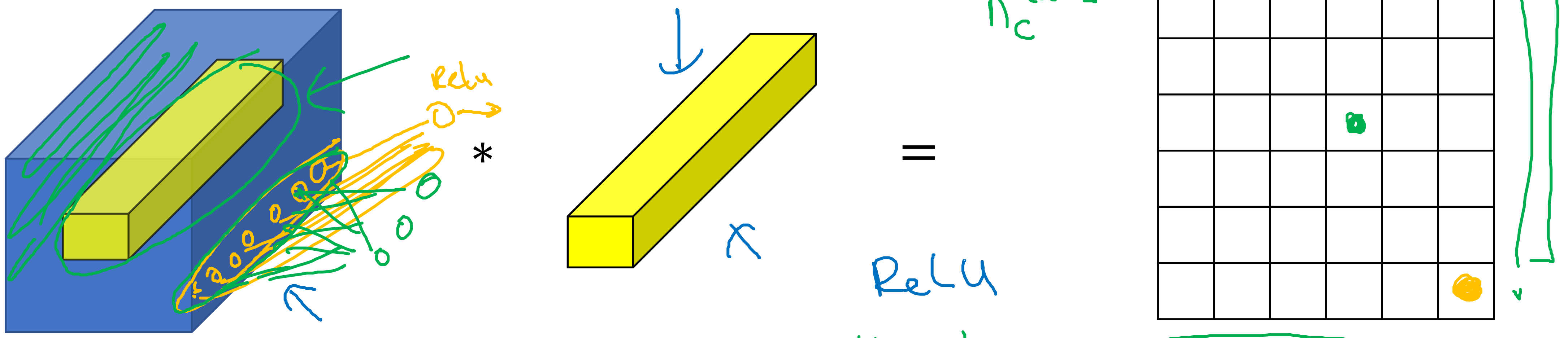
1	2	3	6	5	8
3	5	5	1	3	4
2	1	3	4	9	3
4	7	8	5	7	9
1	5	3	7	4	8
5	4	9	8	3	5

6×6 $\times 1$

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

2	4	6	...		

32 → # filters. $n_c^{[l+1]}$



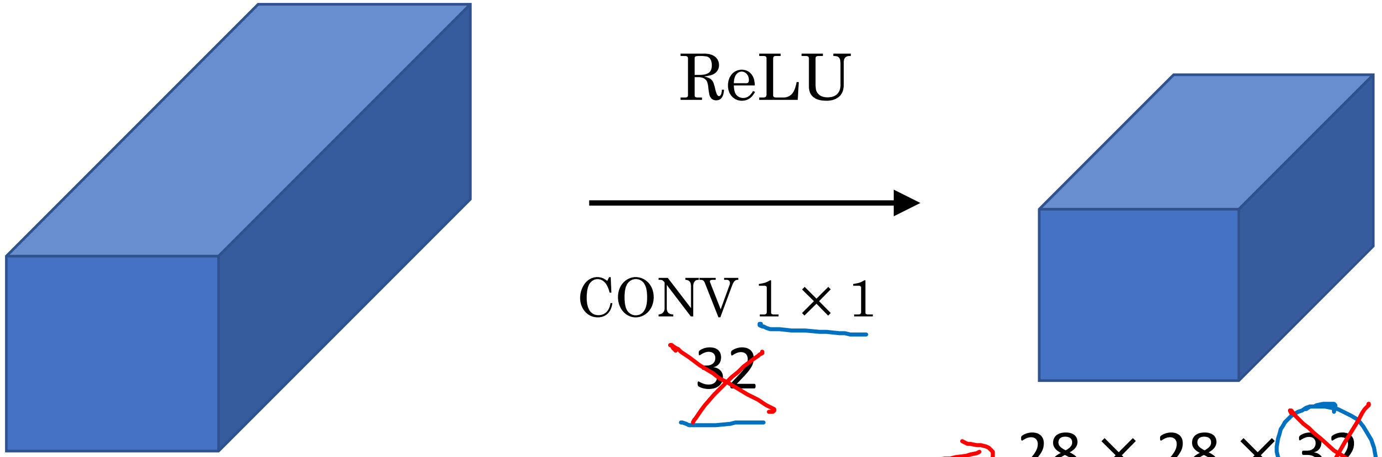
$6 \times 6 \times 32$ $\quad$ $1 \times 1 \times 32$ $\quad$ $6 \times 6 \times \#\ \text{filters}$

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

Andrew Ng

Using 1×1 convolutions

Shrinking channels
(fancy pooling).



→ $28 \times 28 \times 192$

→ $28 \times 28 \times \cancel{32}$ 192

$1 \times 1 \times 192$

32 filters.

n_H, n_W, n_C

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

- Idea: instead of choosing which filters to apply, apply them all (in parallel) and stack them.
- however this creates layers with many channels which increases enormously the complexity of convolutions.
- The inception module is designed to save computation.

The idea is to replace each convolution with a 2 step operation:

1- "compress the no. channels" by performing n CONV 1x1x no. channels where n is "small within reason".
=> RESULTS IN A "BOTTLENECK LAYER" where the information is compressed;

Case Studies

2- apply the convolution for the desired output dimension to the bottleneck layer.

Inception network motivation

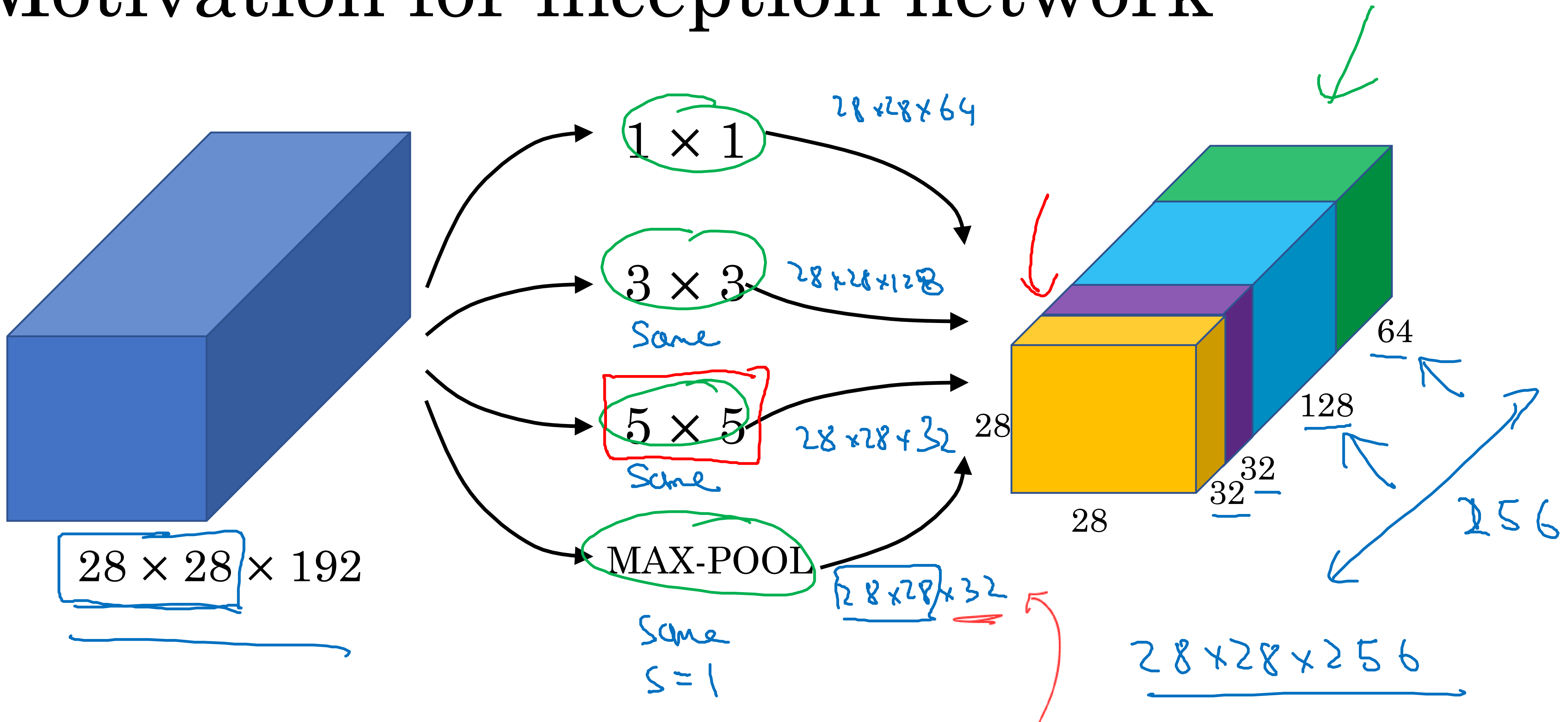
deeplearning.ai

This 2 step operation can reduce computations by a factor of 10 while obtaining similar results.

Size of the bottleneck layer must be "small within reason."

e.g.: instead of $28 \times 28 \times 192 \xrightarrow{32\ \text{CONV}\ 5 \times 5} 28 \times 28 \times 32$ (about 120M operations), do $28 \times 28 \times 192 \xrightarrow{16\ \text{CONV}\ 1 \times 1} 28 \times 28 \times 16 \xrightarrow{32\ \text{CONV}\ 5 \times 5} 28 \times 28 \times 32$ (about 12M operations)

Motivation for inception network

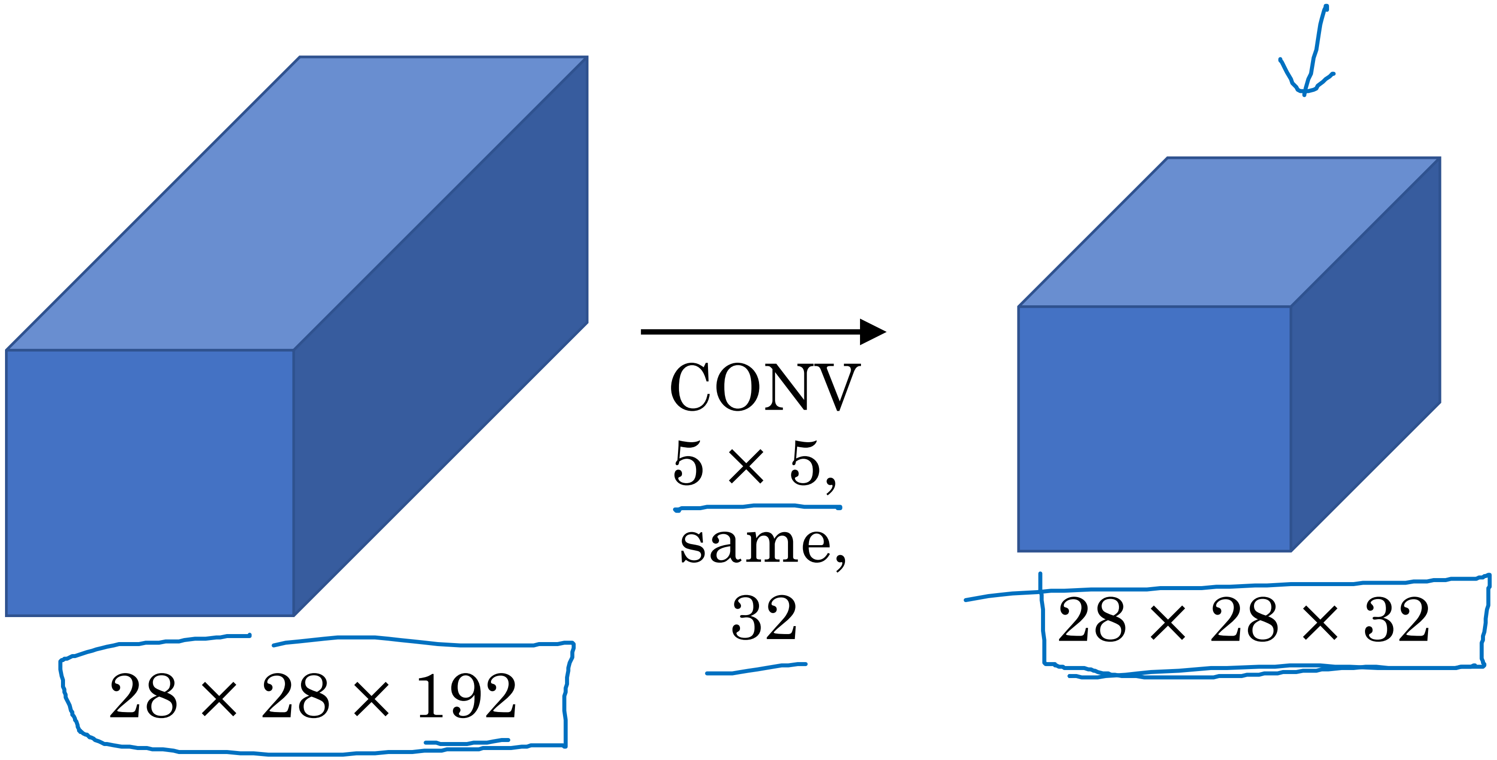


cryptic: actually
maxpool returns same no. channels (192)
but typically this is followed by a 1x1 conv
to shrink the no. channel (in the inception architecture).

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

Andrew Ng

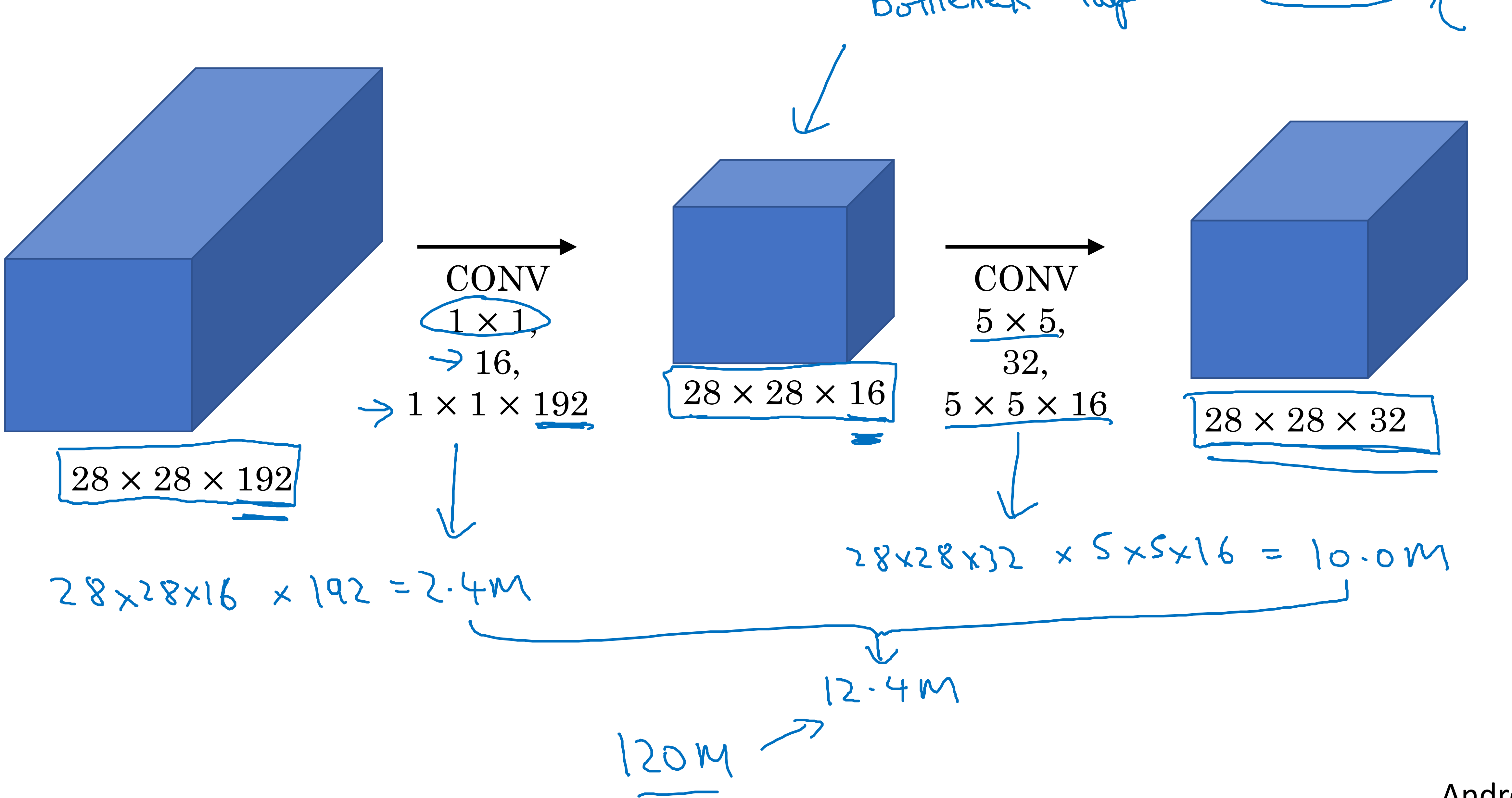
The problem of computational cost



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

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

Using 1×1 convolution



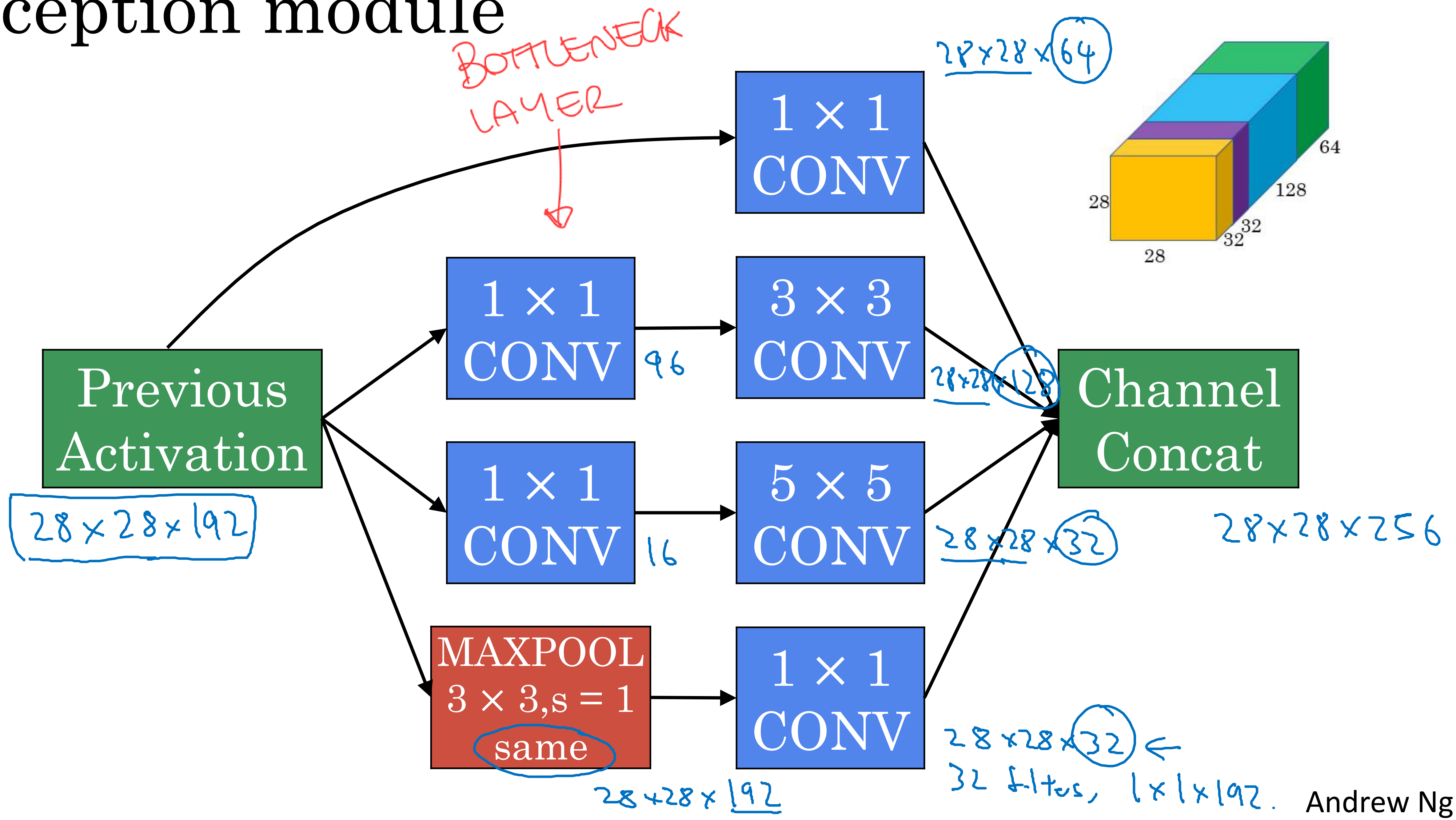
Andrew Ng

deeplearning.ai

Case Studies

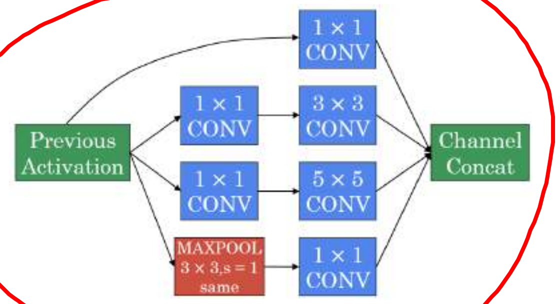
Inception network

Inception module

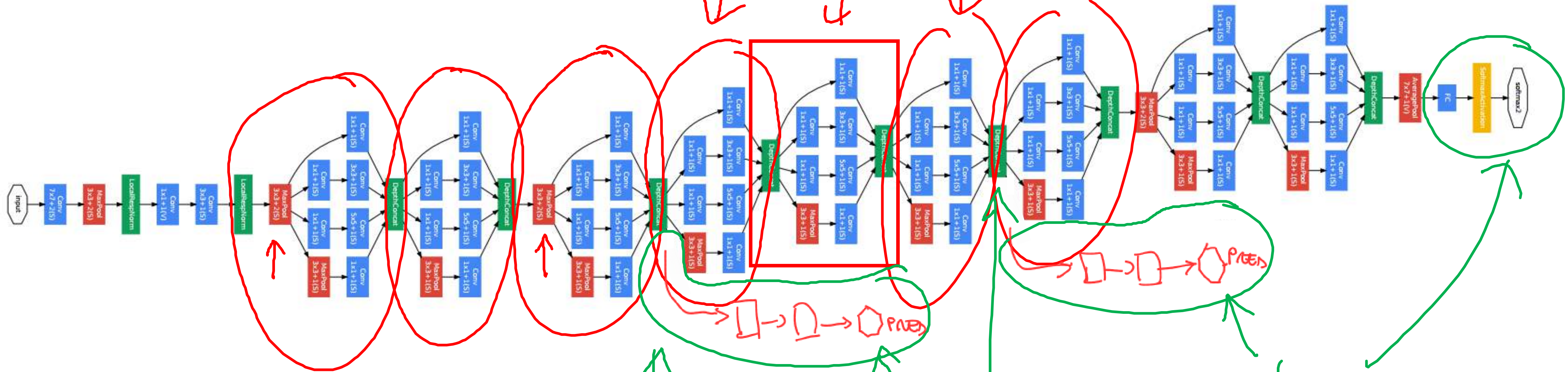


Andrew Ng

Inception network



- An inception network is just the concatenation of inception modules



Softmax

GoogLeNet

A regularization of this network is obtained by introducing some prediction branches in several intermediate places of the network to ensure that intermediate features are not too bad.

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

Andrew Ng

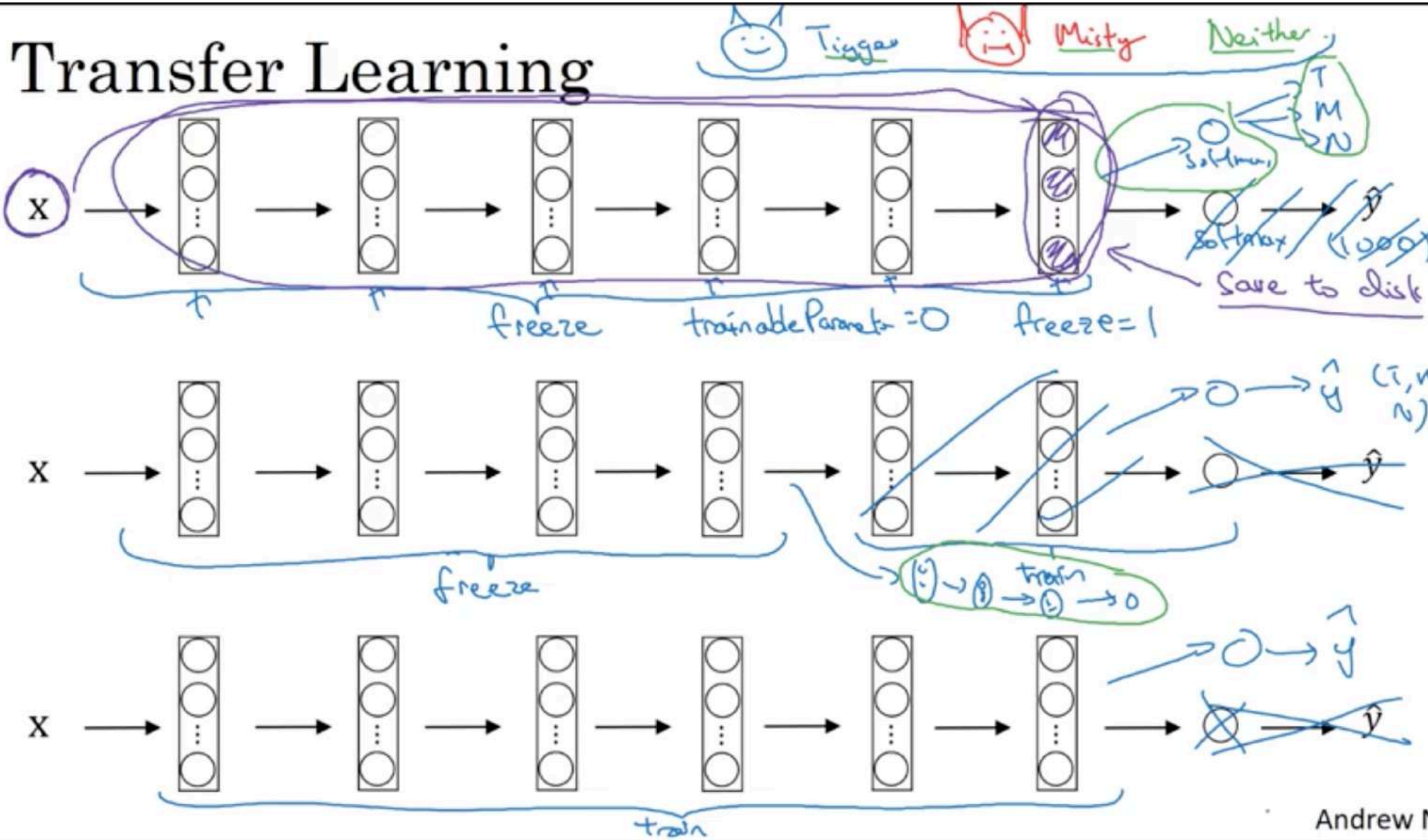
http://knowyourmeme.com/memes/we-need-to-go-deeper

deeplearning.ai

Practical advice for using ConvNets

Transfer Learning

Transfer Learning



Andrew Ng

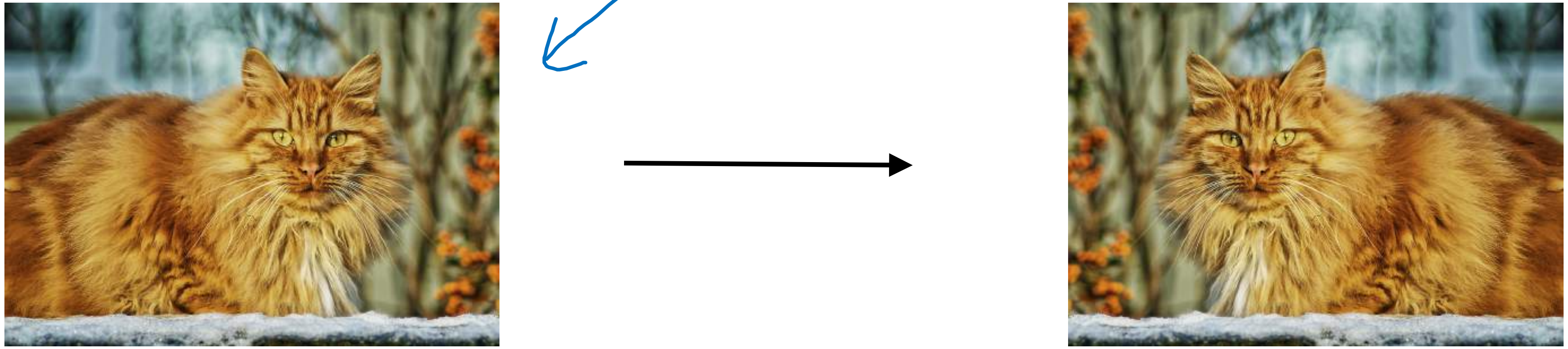
Practical advice for using ConvNets

deeplearning.ai

Data augmentation

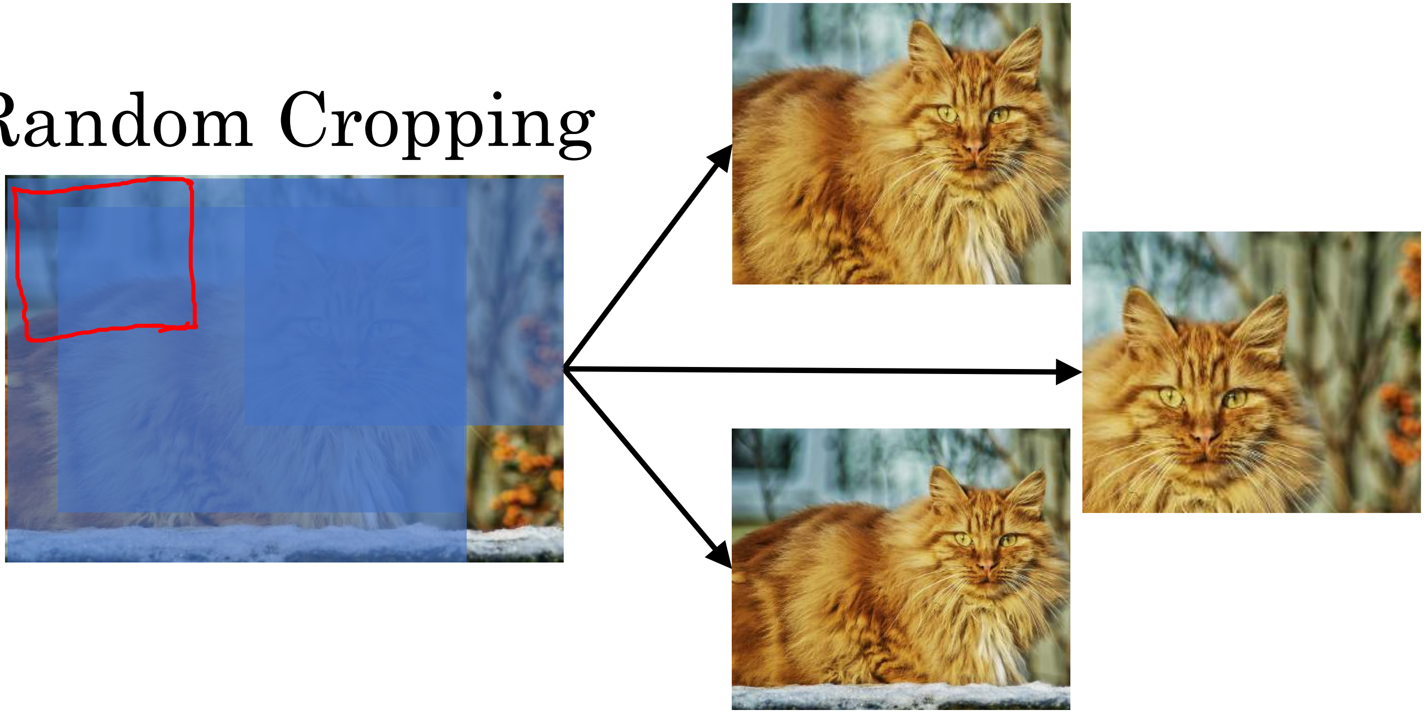
Common augmentation method

Mirroring



y

Random Cropping



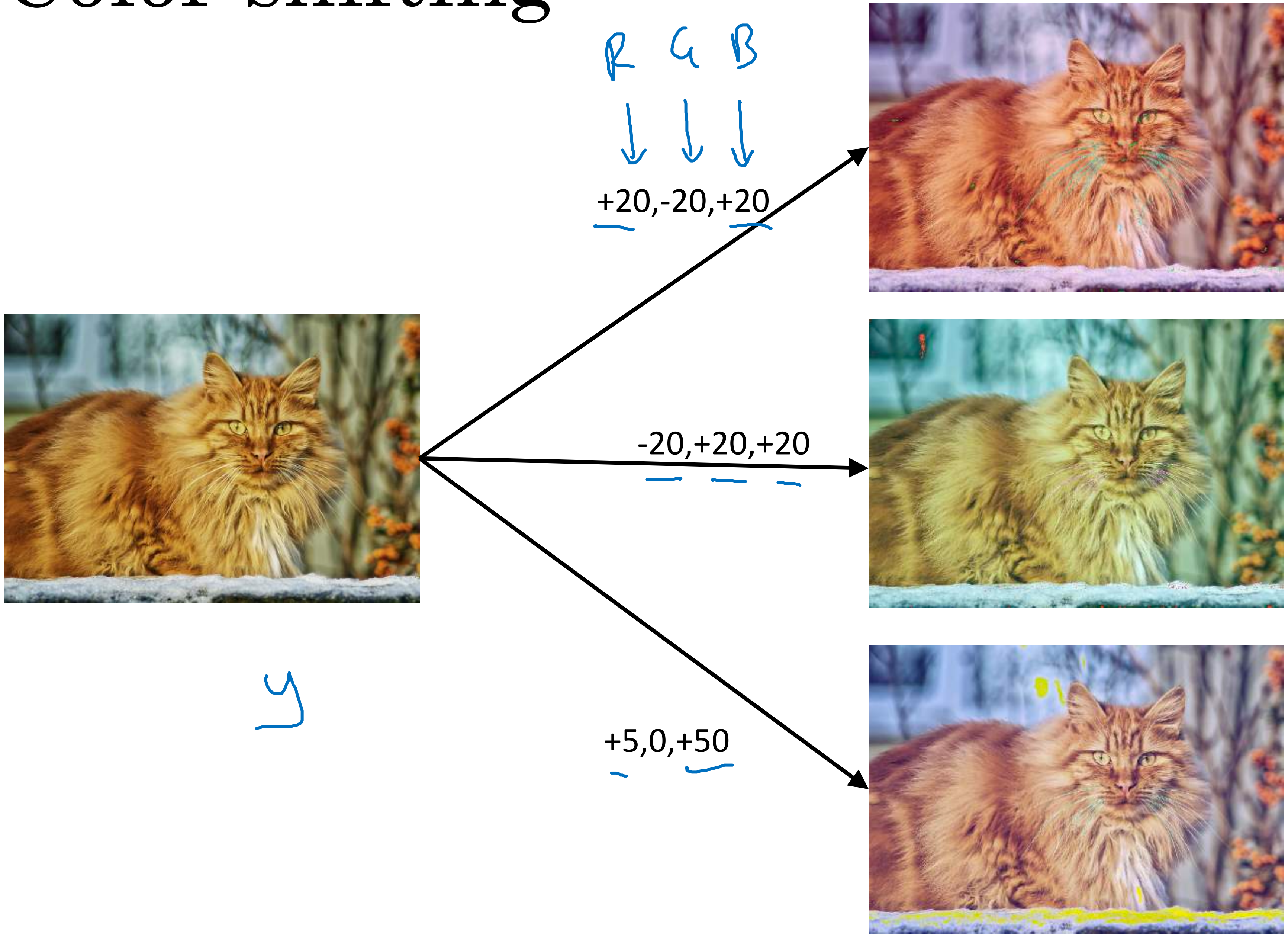
Rotation

Shearing

Local warping

...

Color shifting



Advanced:

PCA

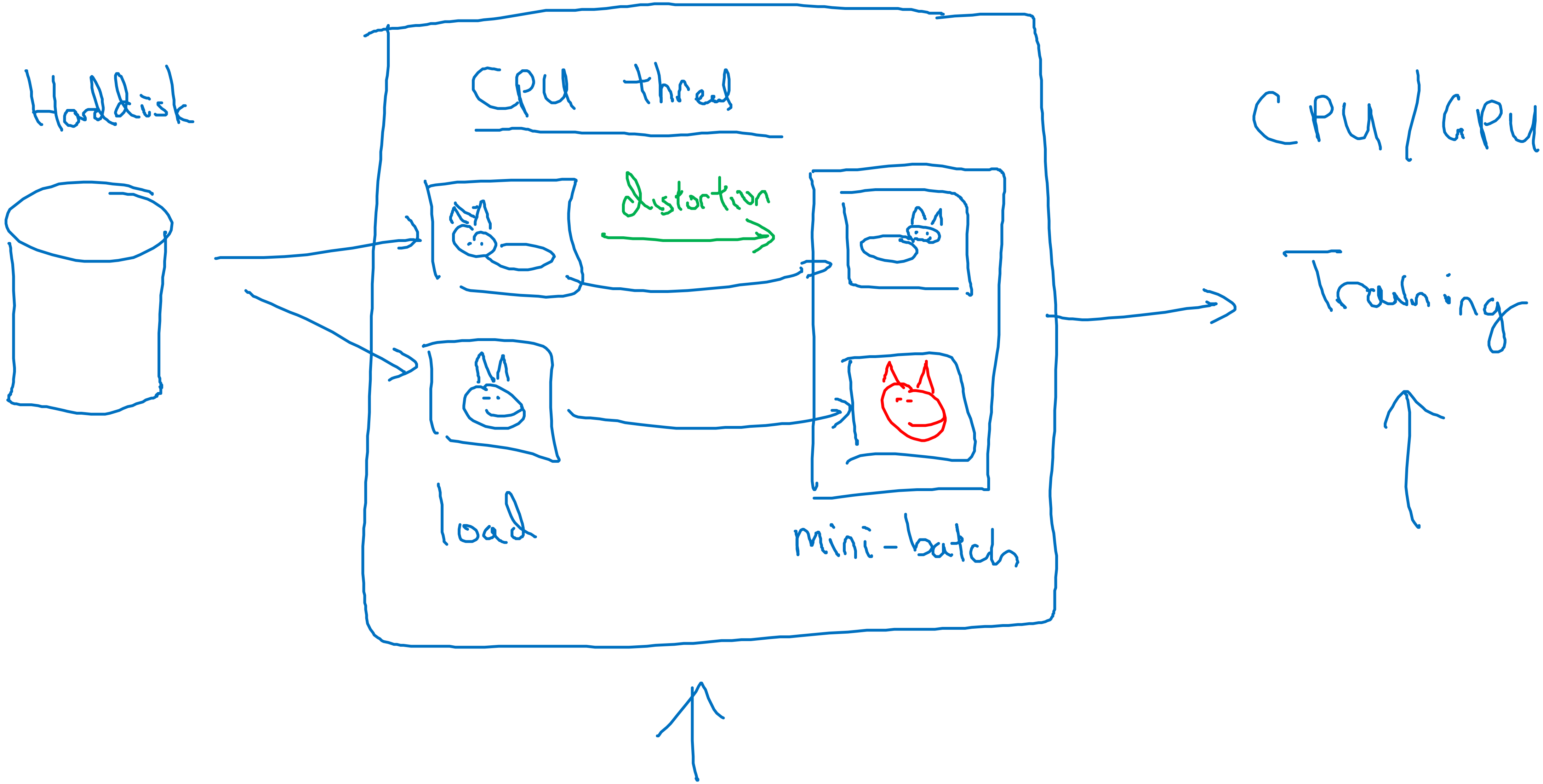
ml-class.org

AlexNet paper

"PCA color augmentation."

R B G

Implementing distortions during training



Andrew Ng

deeplearning.ai

Practical advice for using ConvNets

The state of computer vision

Data vs. hand-engineering

Little data

More hand-engineering ("hacks")

Transfer learning

Object detection

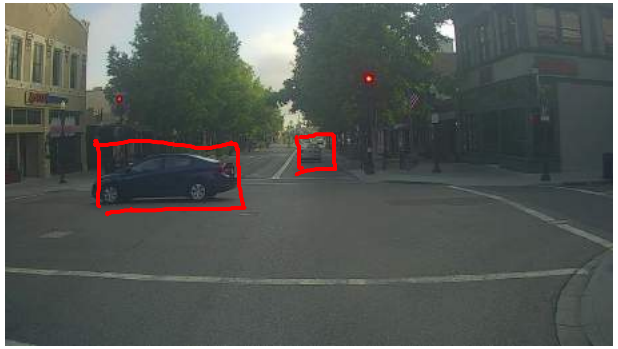
Tigger/Misty/neither.

Image recognition

Speech recognition

Lots of data

Simpler algorithms less hand-engineering



Two sources of knowledge

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

Andrew Ng

Tips for doing well on benchmarks/winning competitions

Ensembling

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

- Train several networks independently and average their outputs

Multi-crop at test time

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

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