deeplearning.ai

Case Studies

Why look at case studies?

Outline

Classic networks:

- LeNet-5
- AlexNet
- VGG

ResNet (152)

Inception

NN هایی که قرار است بحث کند.

deeplearning.ai

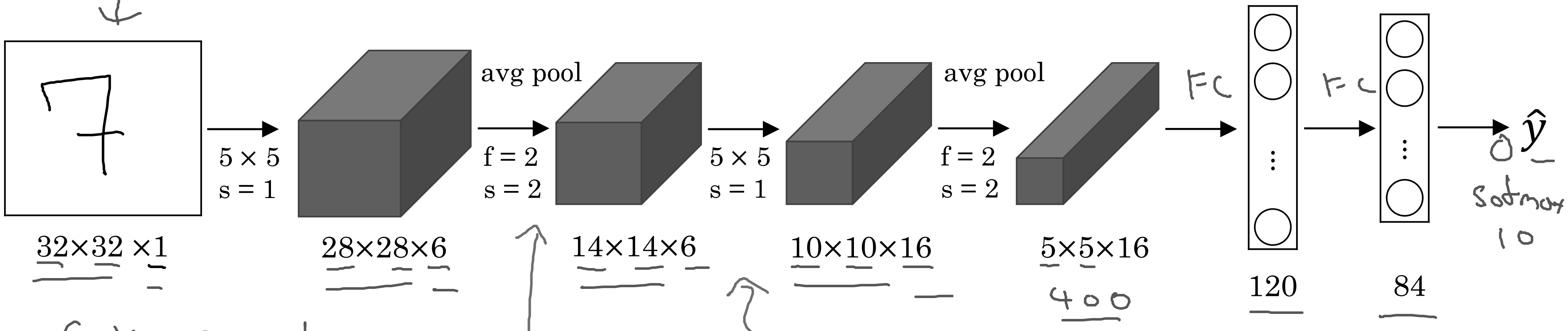
Case Studies

Classic networks

LeNet - 5

n_C ↑

n_H, n_W ↓



60K parameters.

n_H, n_W ↓ n_C ↑

non-linearity after pooling

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

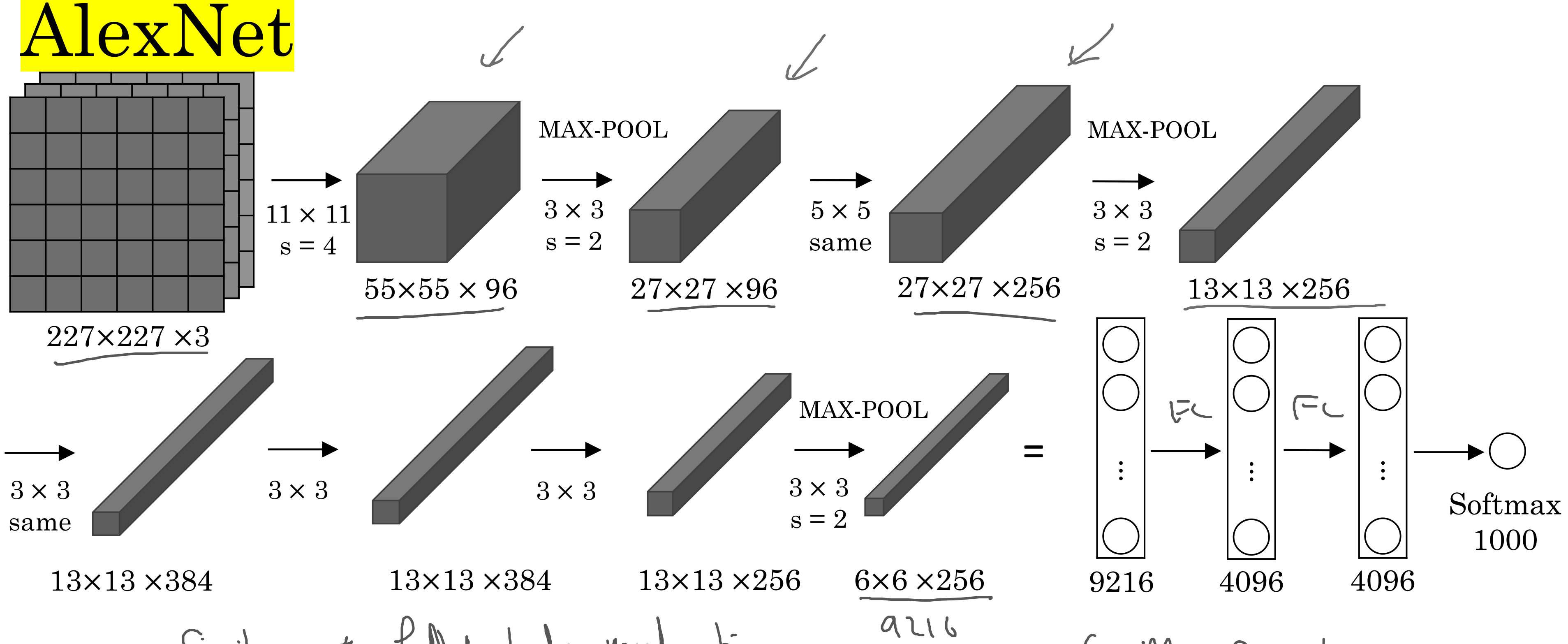
conv pool conv pool fc fc output

Advanced: sigmoid/tanh   ReLU

II III.

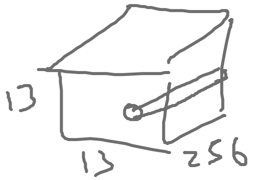
[LeCun et al., 1998. Gradient-based learning applied to document recognition]

AlexNet



~60M parameters

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



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

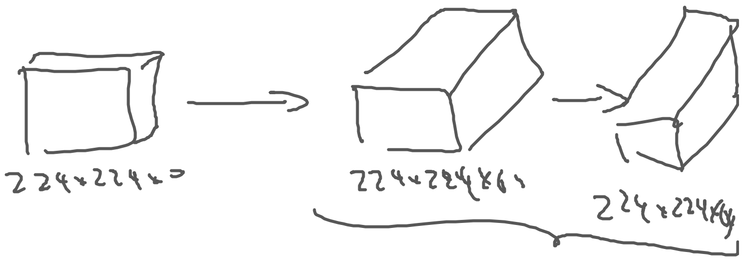
VGG - 16

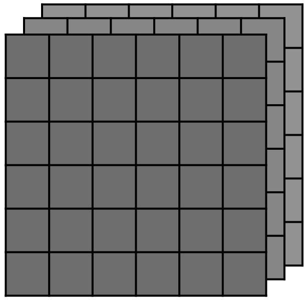
VGG-19

P=0

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

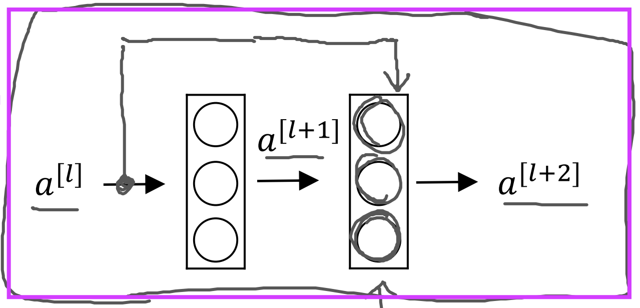
[Simonyan & Zisserman 2015. Very deep convolutional networks for large-scale image recognition]

deeplearning.ai

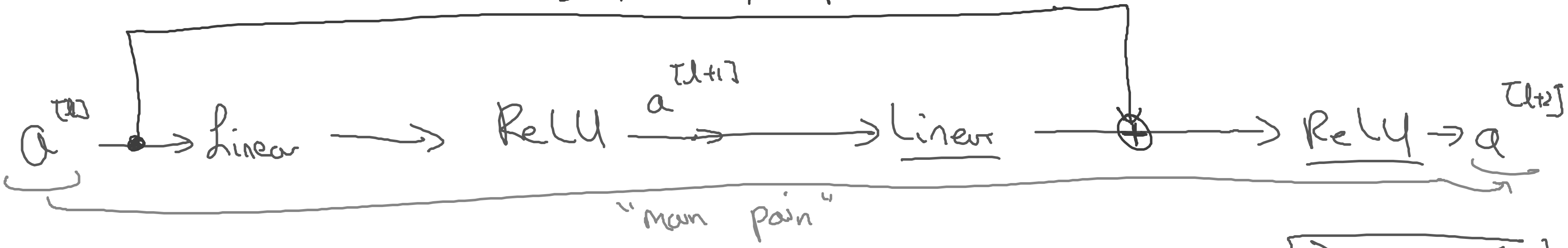
Case Studies

Residual Networks (ResNets)

Residual block



"short cut"/skip connection



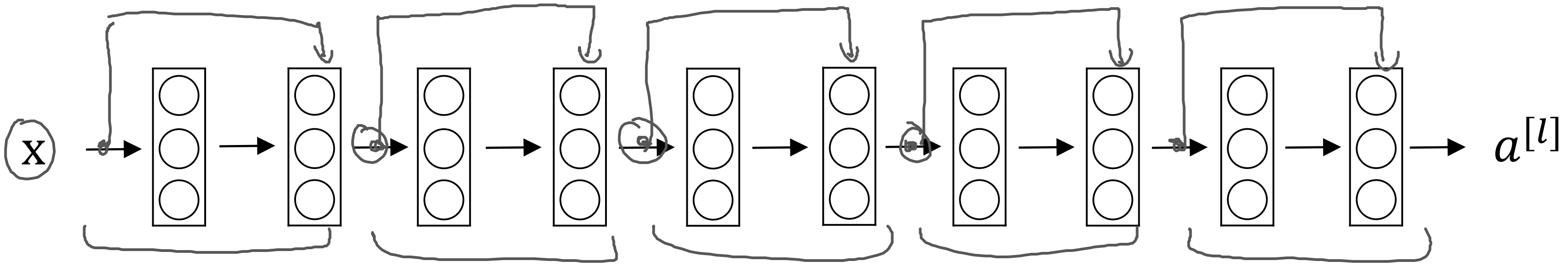
$$\underline{z^{[l+1]}} = \underset{\uparrow}{W^{[l+1]}}\ \underline{a^{[l]}} + \underset{\uparrow}{b^{[l+1]}} \qquad \underline{a^{[l+1]}} = g(\underline{z^{[l+1]}}) \qquad \underline{z^{[l+2]} = W^{[l+2]}a^{[l+1]} + b^{[l+2]}} \qquad \boxed{\cancel{a^{[l+2]} = g(z^{[l+2]})}}$$

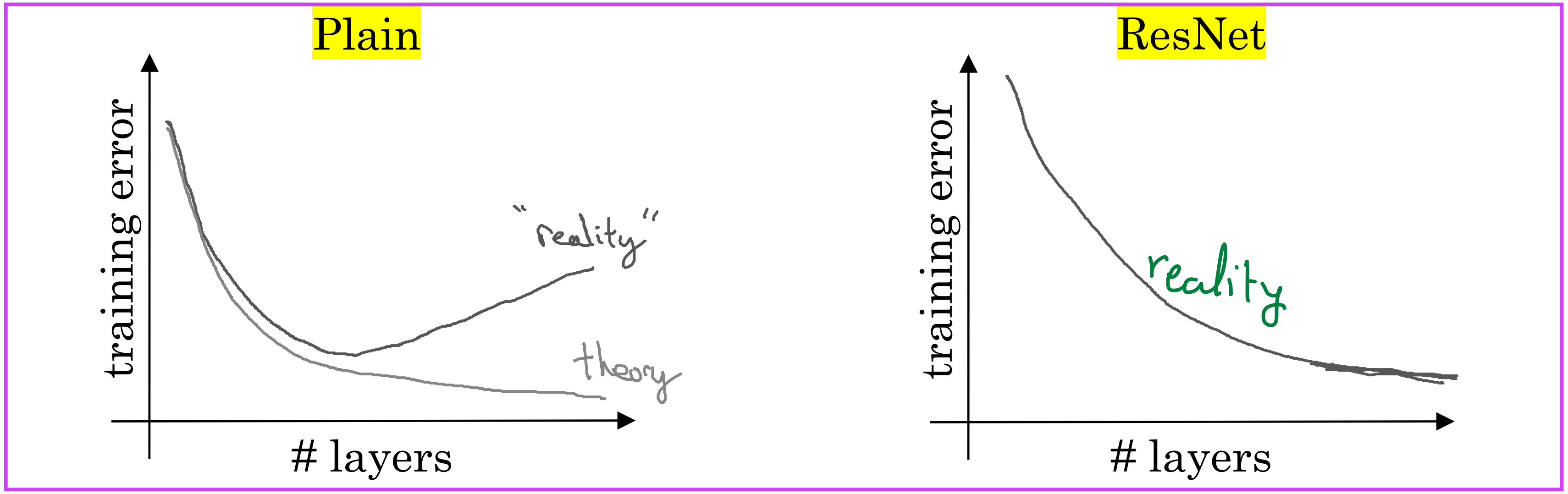
$$\boxed{a^{[l+2]} = g\left(z^{[l+2]} + \underbrace{a^{[l]}}_{}\right)}$$

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

Residual Network

"plain network"





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

deeplearning.ai

Case Studies

Network in Network and 1×1 convolutions

$$28\times28\times192 \xrightarrow[1\times1\times192]{\text{Conv }1\times1} 28\times28\times n_c$$

filters

Why does a 1 × 1 convolution do?

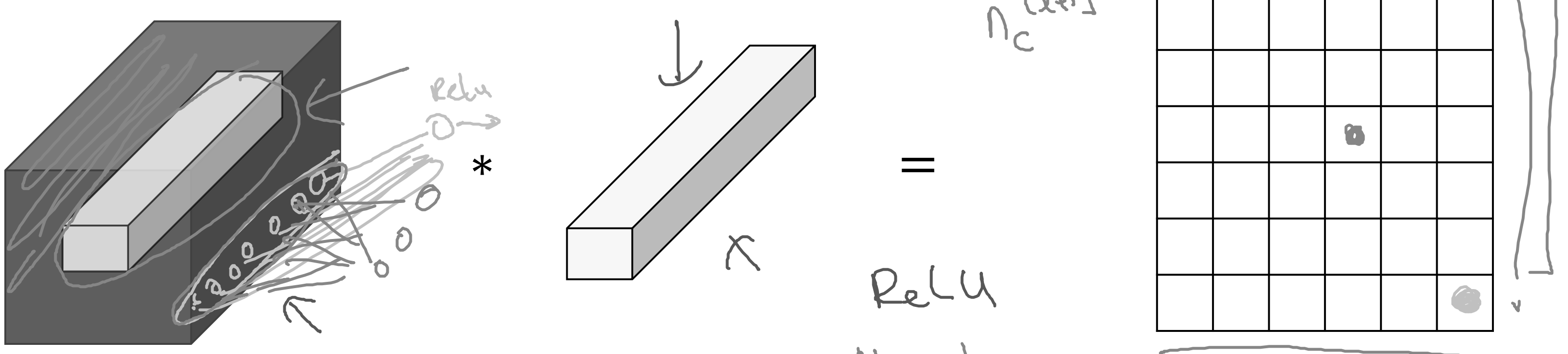
changes n_c, fixes n_H, n_W

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

6 × 6 × 1

* $\boxed{2}$ =

2	4	6	...		



32 → # filters. $n_c^{[l+1]}$

6 × 6 × 32

1 × 1 × 32

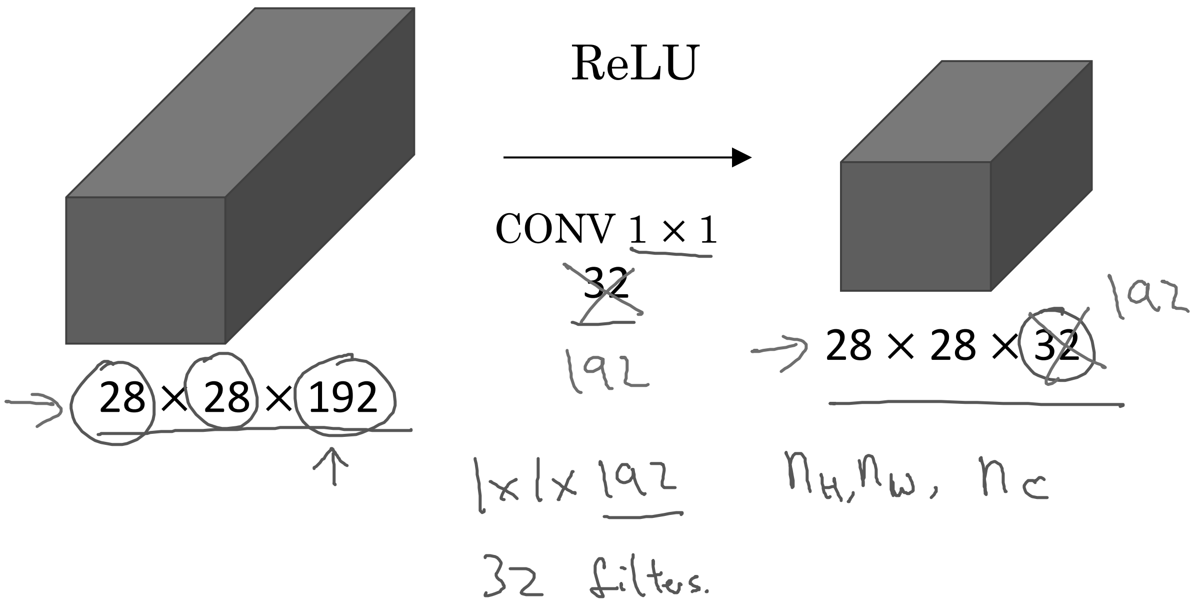
Network in Network

6 × 6 × # filters

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

حتی می‌توان تبدیل 28x28x192 → 28x28x192 داشت؛ در این حالت نقش 1x1 conv اضافه کردن non-linearity است.

Using 1×1 convolutions



28 × 28 × 192 → 28 × 28 × 32

برای آن به 32 تا conv 1x1 که هرکدام 1x1x192 هستند احتیاج داریم.

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

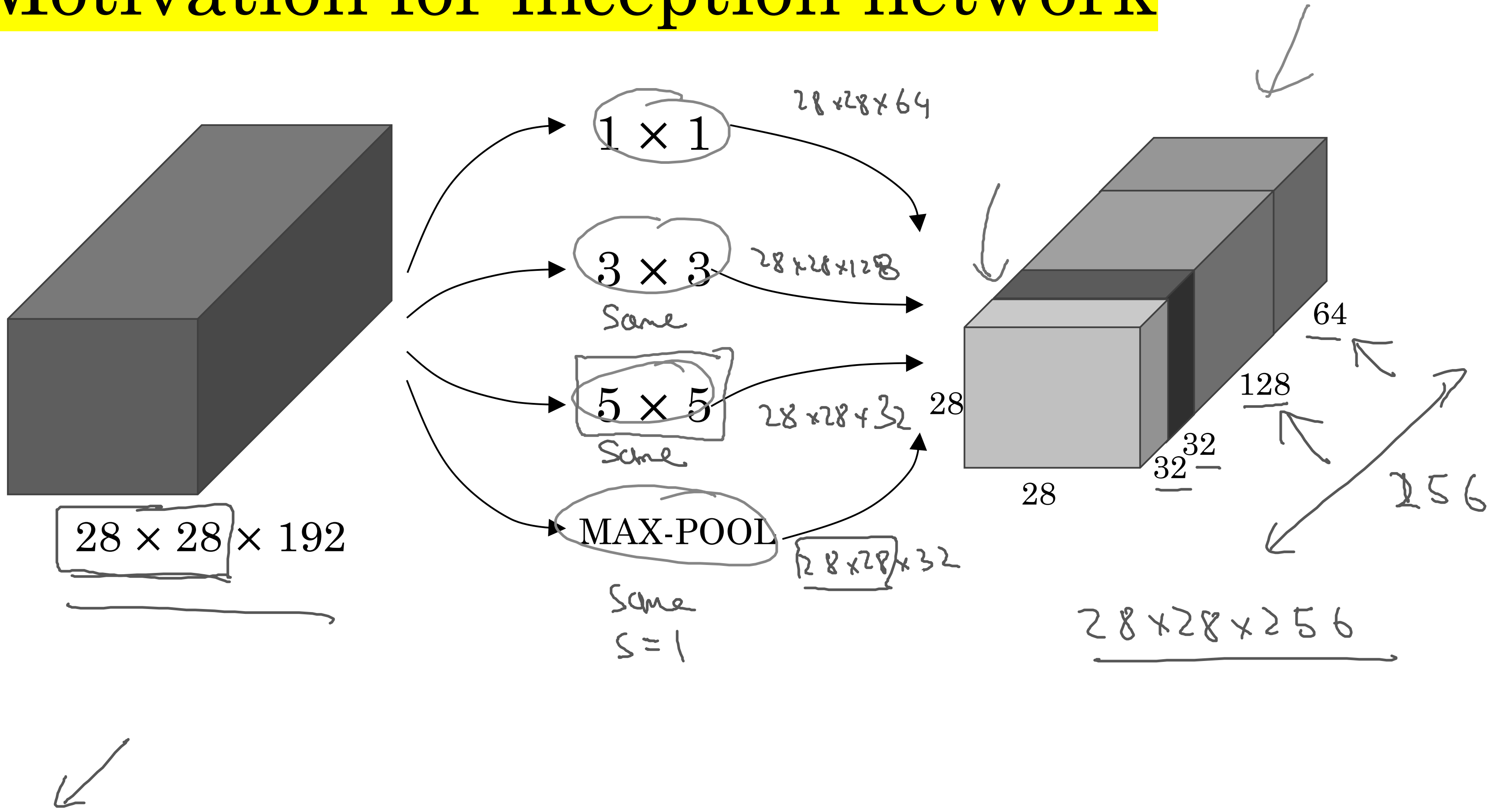
deeplearning.ai

Case Studies

Inception network motivation

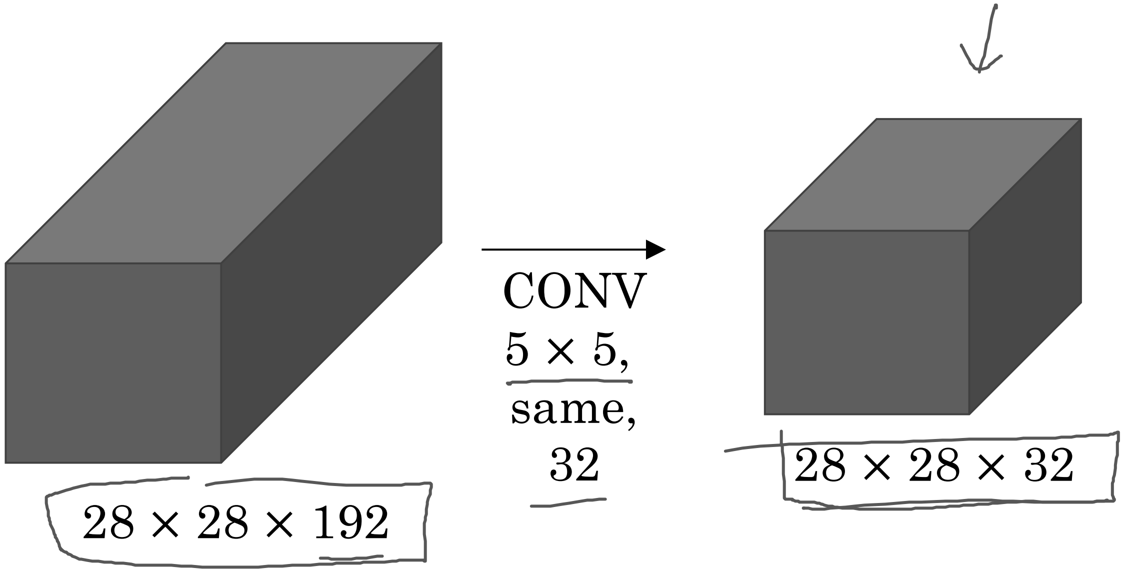
28×28×192 → 28×28×256

Motivation for inception network



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

The problem of computational cost



CONV
5 × 5,
same,
32

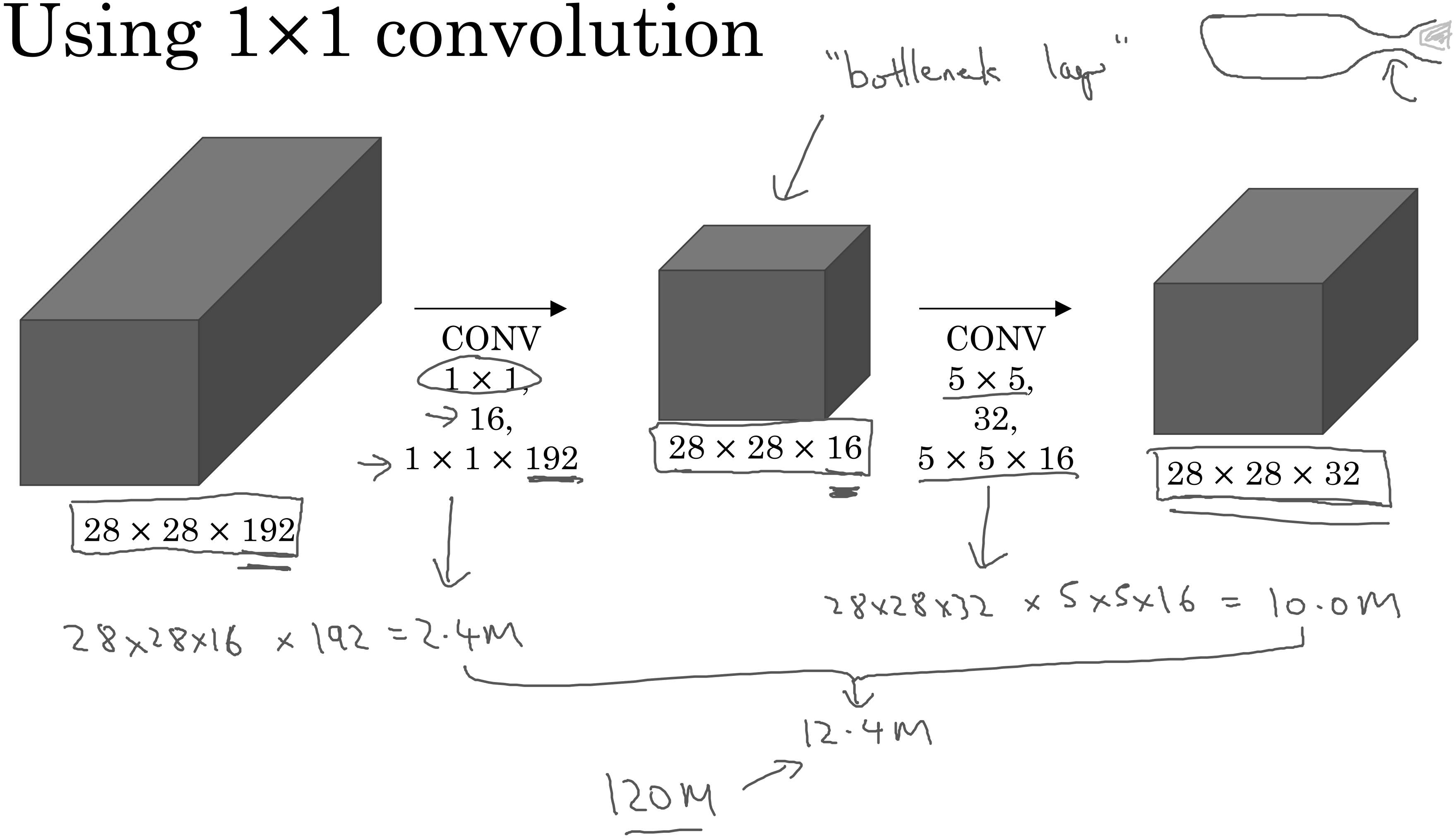
28 × 28 × 192

28 × 28 × 32

32 filters.   filters are 5×5×192.

28×28×32 × 5×5×192 = 120M.

Using 1×1 convolution



"bottleneck layer"

$28 \times 28 \times 192$

CONV
1×1,
16,
$1 \times 1 \times 192$

$28 \times 28 \times 16$

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

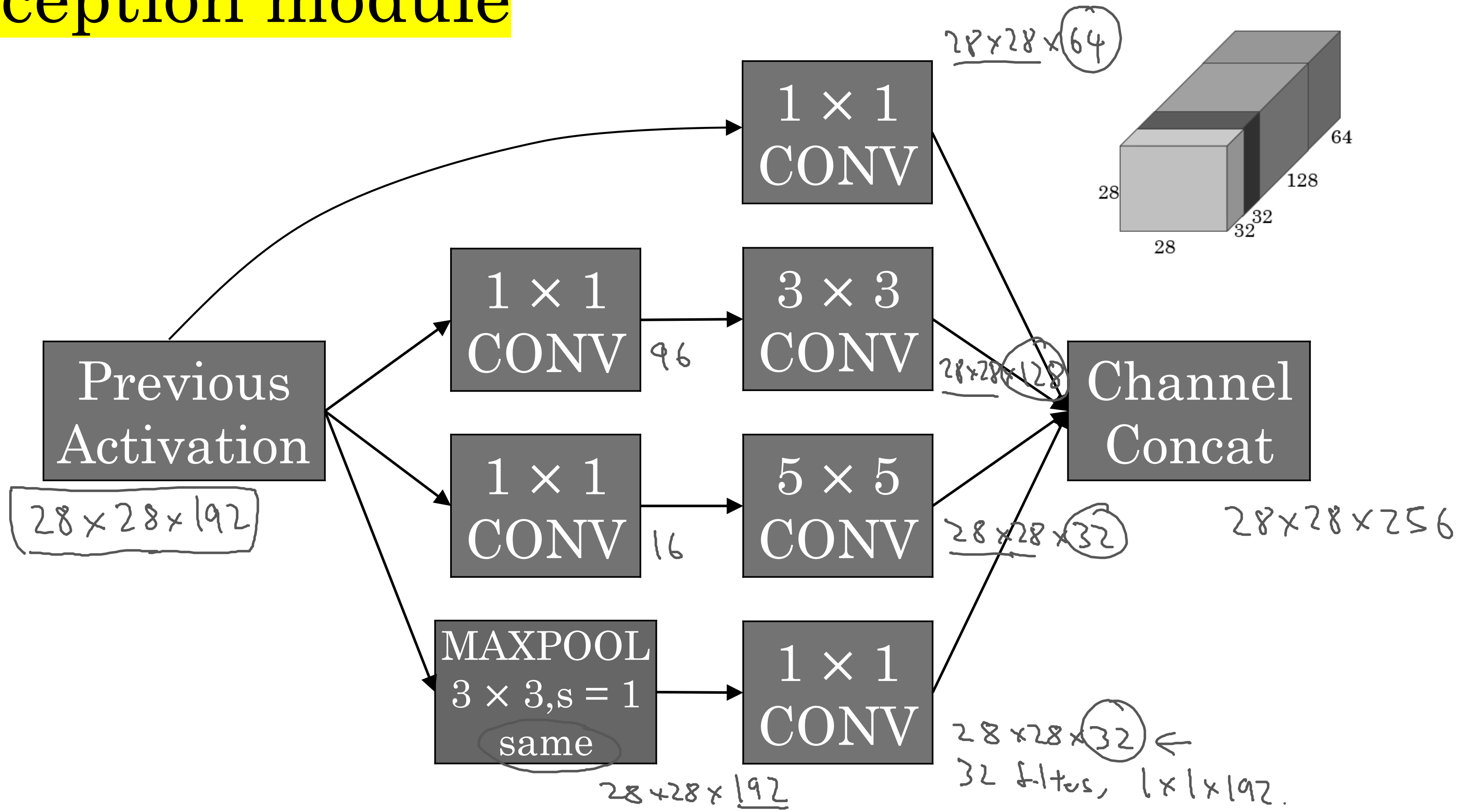
120M

deeplearning.ai

Case Studies

Inception network

Inception module



Previous Activation

28×28×192

1 × 1 CONV

1 × 1 CONV

96

3 × 3 CONV

28×28×128

1 × 1 CONV

16

5 × 5 CONV

28×28×32

MAXPOOL 3 × 3,s = 1 same

28×28×192

1 × 1 CONV

28×28×32 ←

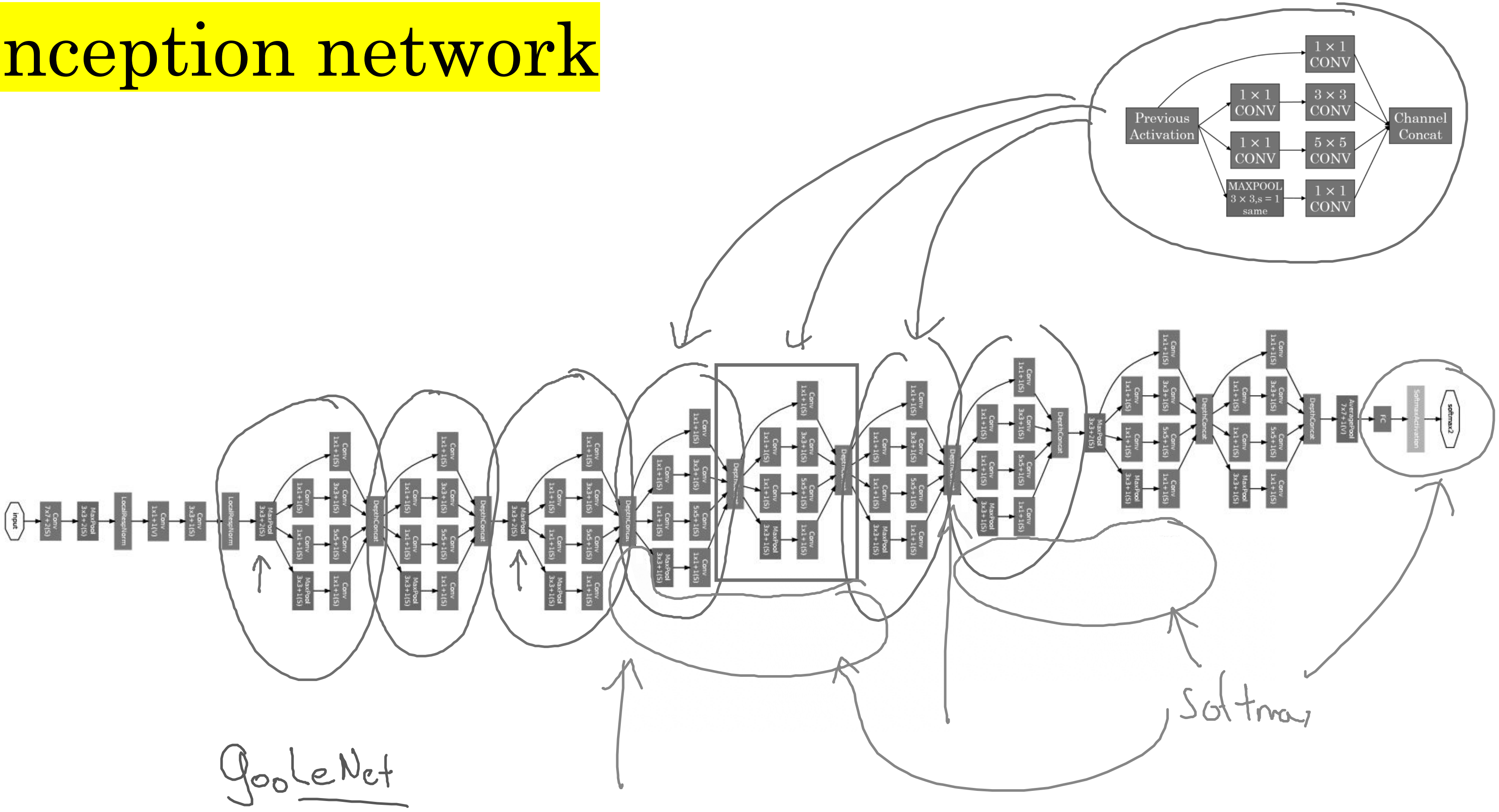
32 filters, 1×1×192.

28×28×64

Channel Concat

28×28×256

Inception network



Softmax

GooLeNet

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

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

Tips for doing well on benchmarks/winning competitions

Ensembling

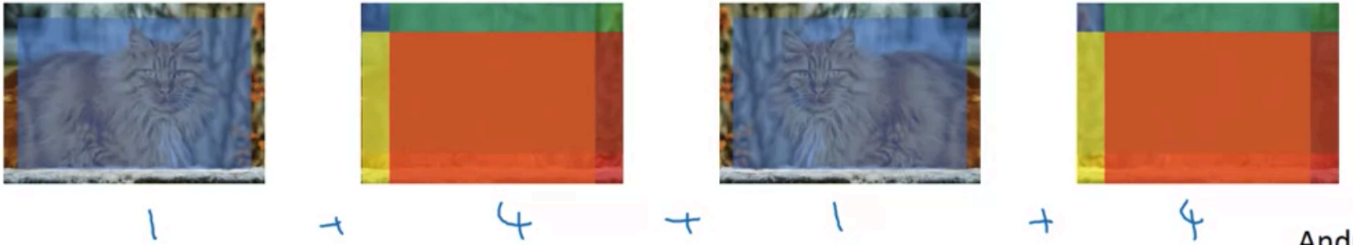
3-15 networks $\to \hat{y}$

- 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



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