Introduction to Computer Vision

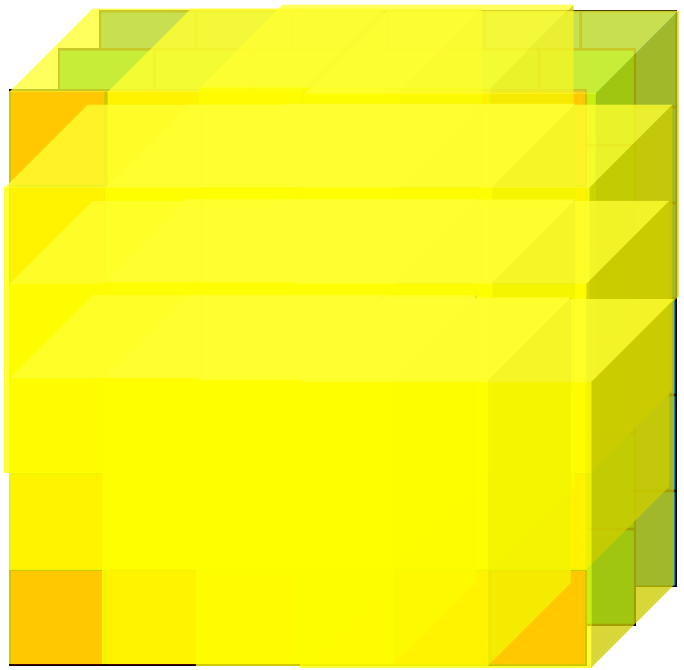
Continue on DL

Dr. Sedat Ozer

Announcements / Questions:

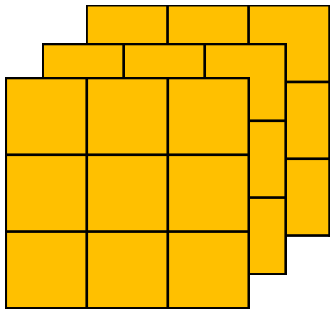
- HW1 - questions
- Class schedule - Fridays
- Midterm: Do we have a midterm?
- Paper presentations: **Due today**. Submit your presentation info at the link that was sent to you in my first email.
- Final project: Start working on your project early. Have frequent meetings with me especially if you are aiming

Convolutions on RGB image



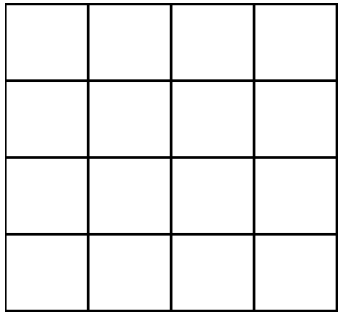
6x6x3

*



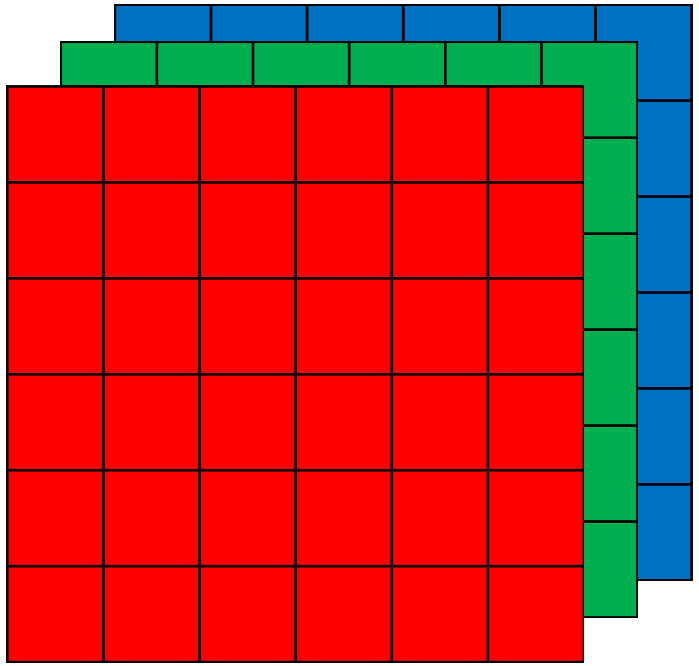
3x3x3

=



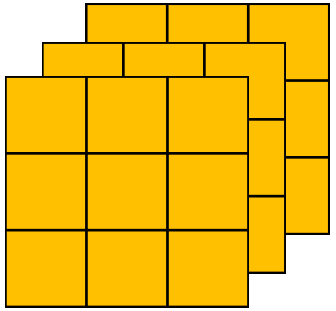
4 x 4

Multiple filters



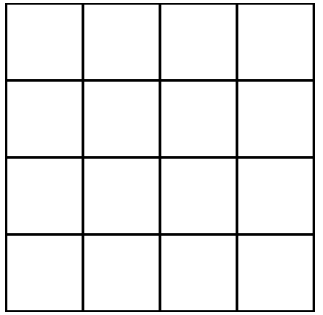
6 x 6 x 3

*



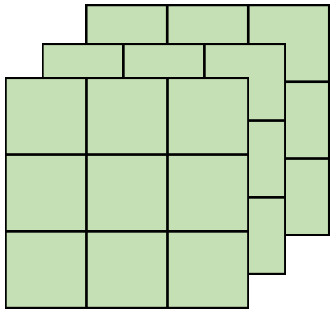
3 x 3 x 3

=



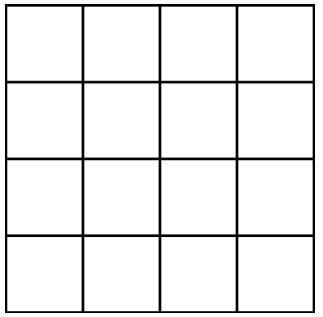
4 x 4

*



3 x 3 x 3

=



4 x 4

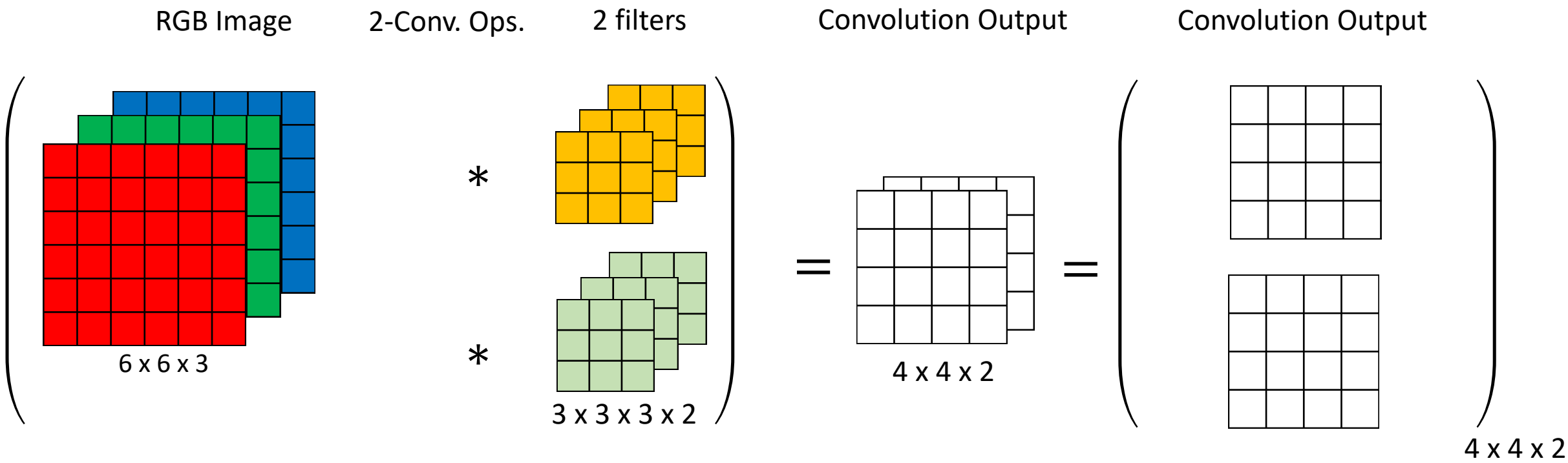
Number of filters

Number of channels

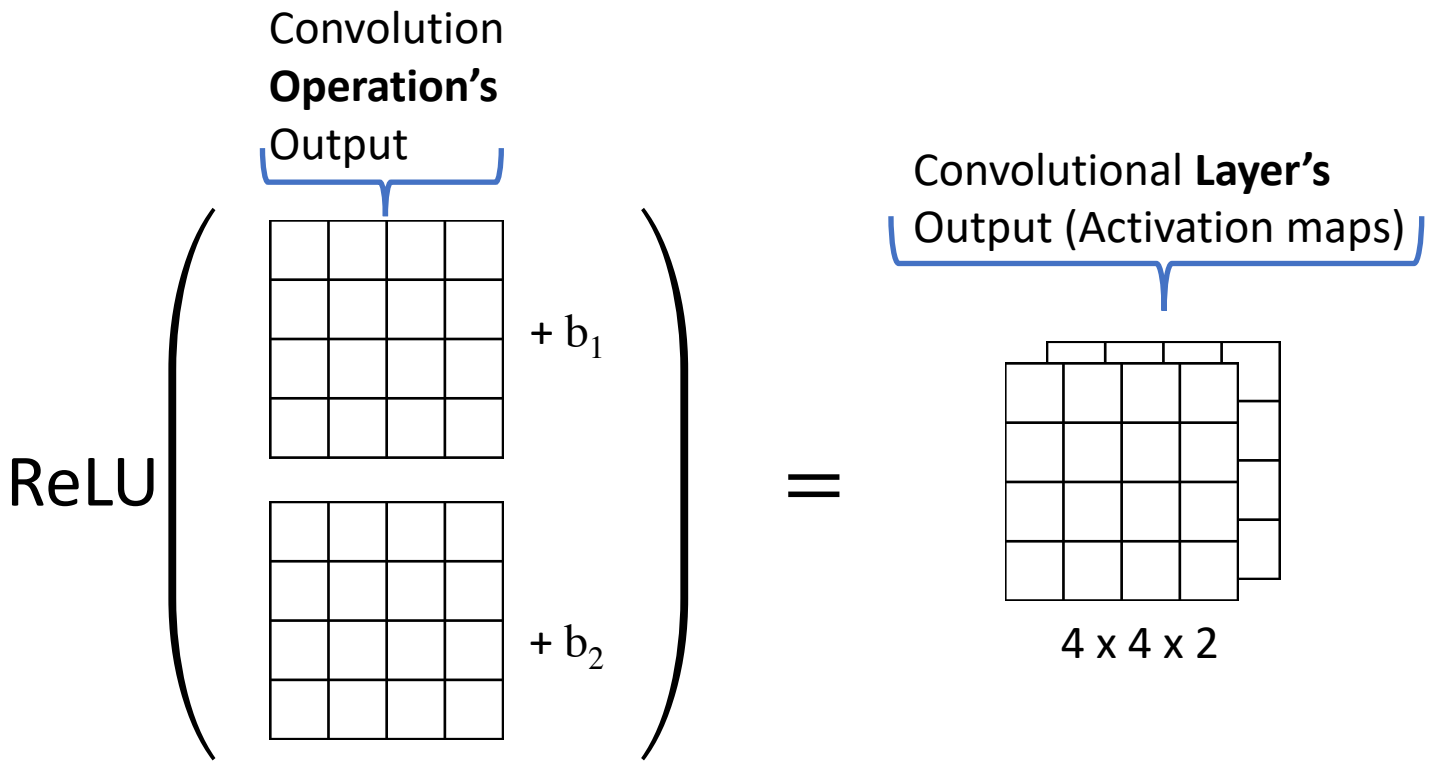
$n \times n \times n_c$ * $f \times f \times n_c \times 2$ = $(n- f +1) \times (n-f+1) \times 2$

Number of filters

Multiple filters



A Convolutional Layer of CNN (ConvNet)



Convolutional Layer Output: Activation_map = ReLU ($(RGBImage * filter) + b$)

Output of a neuron in a FC NN: a = ReLU ($\mathbf{w}^{\mathrm{T}}\mathbf{x} + b$) in FC NN

Similar notation! (but not equal)

Input – Output Dimensions for l^{th} Convolutional Layer

Input (One image): $n_H^{[l-1]}$ x $n_W^{[l-1]}$ x $n_C^{[l-1]}$ ⟶ Output: $n_H^{[l]}$ x $n_W^{[l]}$ x $n_C^{[l]}$

Padding size: $p^{[l]}$ x $p^{[l]}$

Stride size: $s^{[l]}$ x $s^{[l]}$

Filter size: $f^{[l]} \times f^{[l]} \times n_C^{[l-1]} \times n_C^{[l]} = One\_Filter_{Dims} \times n_C^{[l]}$

(Number of Weights = Filter size, Bias size: $n_C^{[l]}$)

Output dims (for one image) : $\left\lfloor \frac{n_H^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \times \left\lfloor \frac{n_W^{[l-1]} + 2p^{[l]} - f^{[l]}}{s^{[l]}} + 1 \right\rfloor \times n_C^{[l]}$

$n_H^{[l]}$ x $n_W^{[l]}$ x $n_C^{[l]}$

The output dimensions for m input samples $A^{[l]}$: m x $n_H^{[l]}$ x $n_W^{[l]}$ x $n_C^{[l]}$

Common Layer Types in a ConvNet

- Convolutional Layer (CONV)
- Fully Connected (FC, Dense)
- Locally Connected (not so commonly used!)

These three layers have parameters to be learned!

- Pooling (POOL)
- Activation Layer
  - (Sometimes, activation function is considered as a separate layer)

…

No learned parameters for these layers!

We have already seen both CONV and FC layers!

For now, we will use only those two layer types mentioned above.

Pooling layer: Max pooling

Input

1	3	2	1
2	7	1	0
2	3	4	3
5	6	1	2

Filter size: f =2

Stride size: s=2

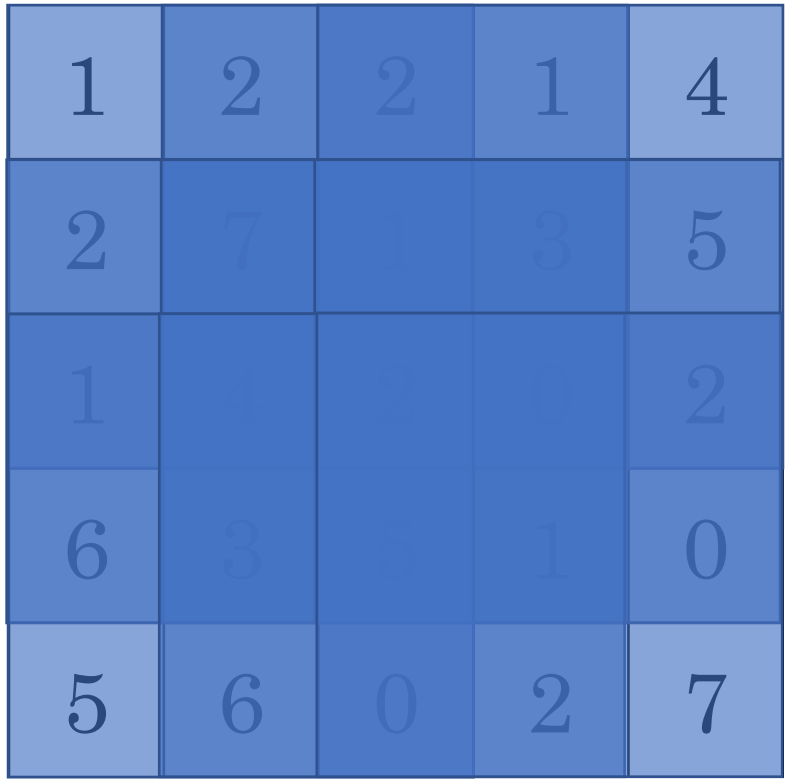
Output

7	2
6	4

Pooling is typically applied on each channel separately.

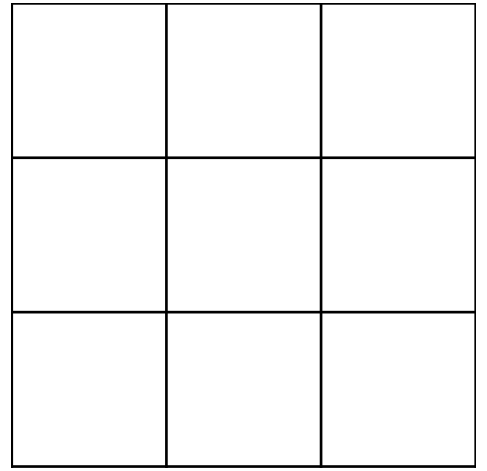
Pooling layer: Max pooling

Input



f=3
s=1

Output

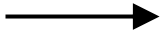


Pooling layer: Average pooling

Input

1	0	2	1
2	9	1	1
2	3	2	2
5	6	2	2

Filter size= 2 x 2
Stride: s = 2

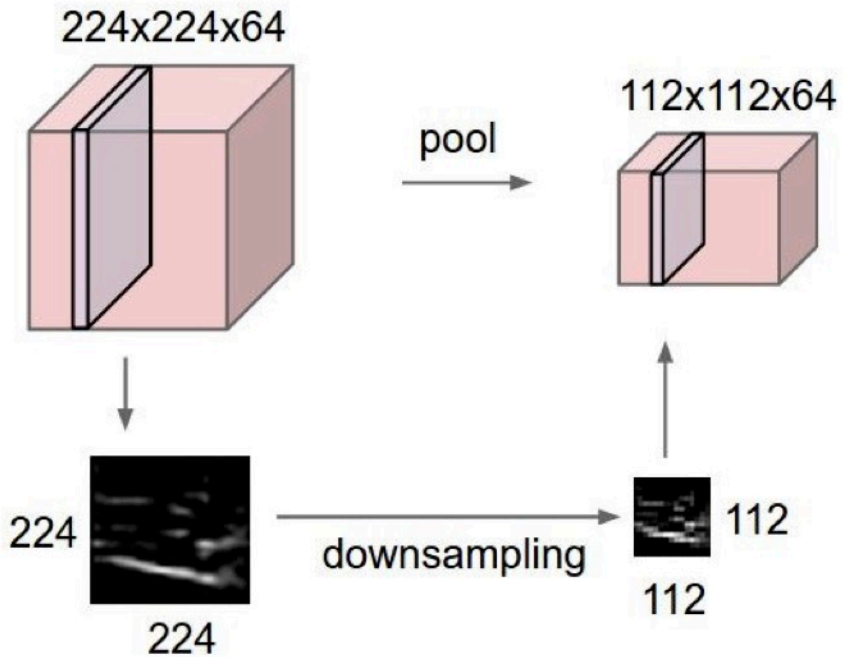


Output

3	1.25
4	2

Pooling layer

- makes the representations smaller and more manageable
- operates over each activation map independently:



Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Summary of pooling

Hyperparameters:

f : filter size

s : stride

Max or **Average** pooling

f=2, s=2

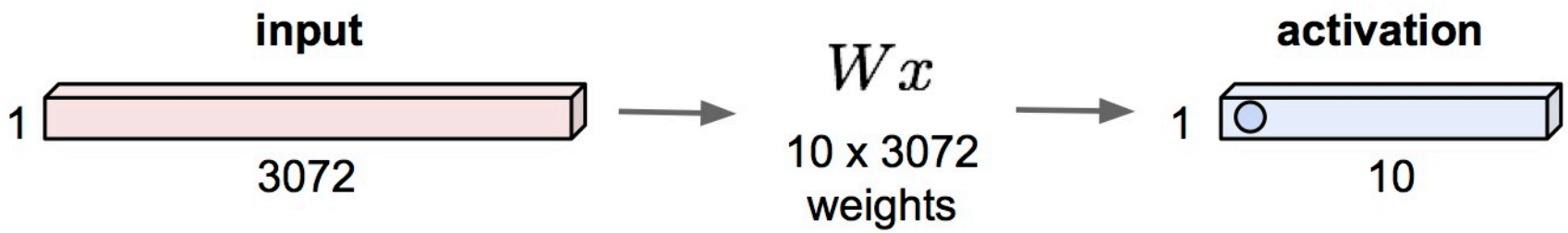
f=3, s=2

p = 0

Input: n_H x n_W x n_C Output: $\left\lfloor \frac{n_H - f}{s} + 1 \right\rfloor \times \left\lfloor \frac{n_W - f}{s} + 1 \right\rfloor \times n_C$

Fully Connected Layer

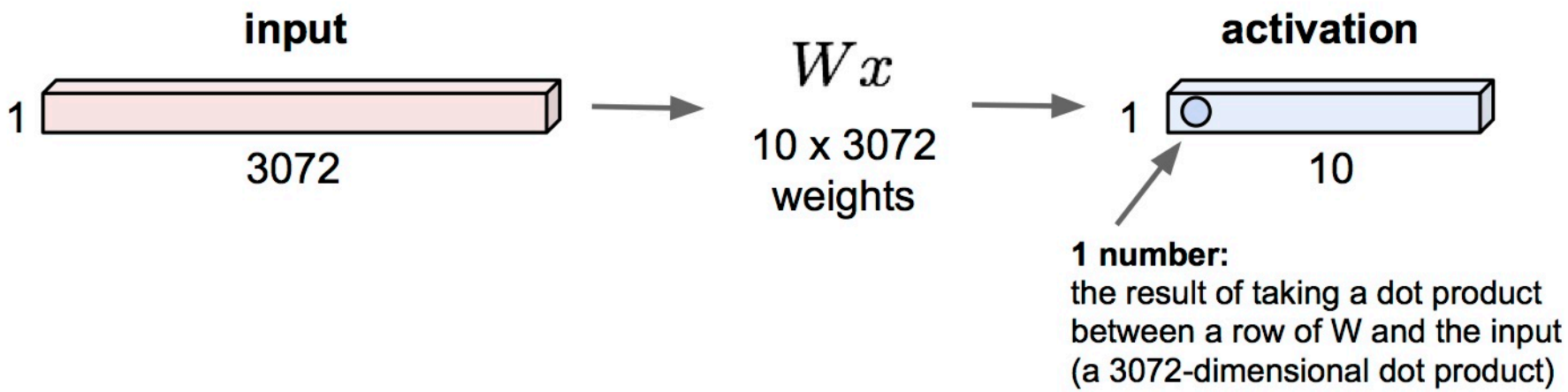
32x32x3 image -> stretch to 3072 x 1



Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Fully Connected Layer

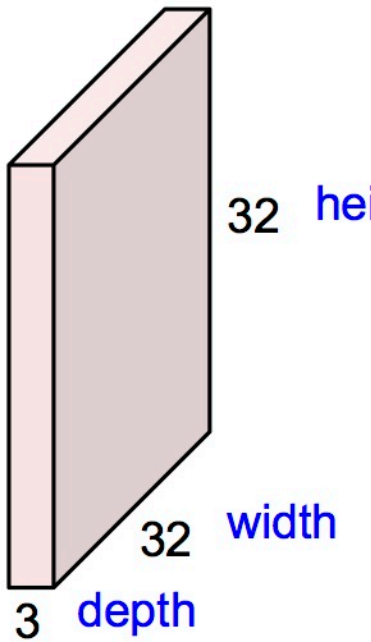
32x32x3 image -> stretch to 3072 x 1



Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Convolution Layer

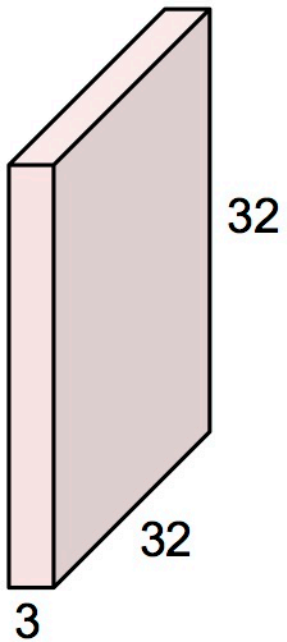
32x32x3 image -> preserve spatial structure



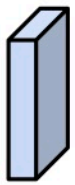
Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Convolution Layer

32x32x3 image



5x5x3 filter



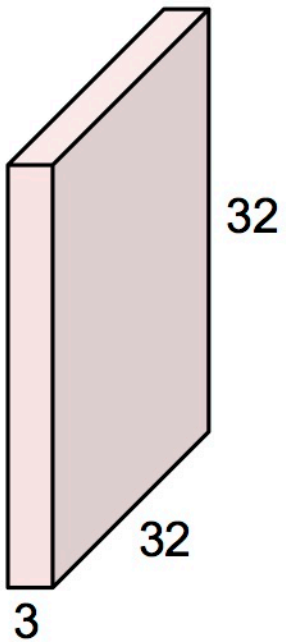
Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Source: Fei-Feli Li & Justin Johnson & Serena Yeung

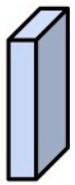
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



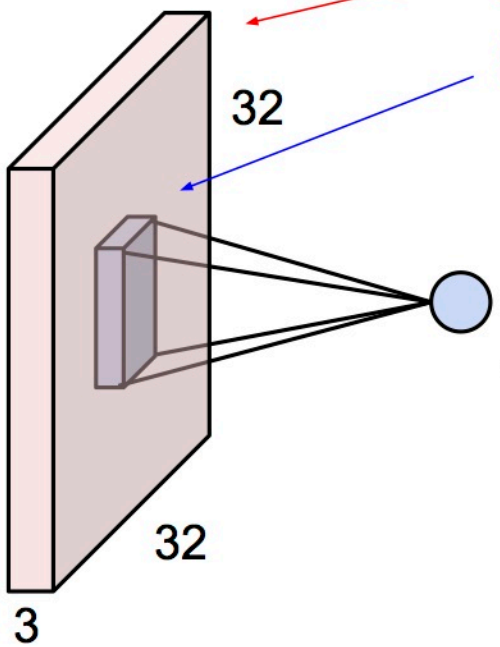
5x5x3 filter



Convolve the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Convolution Layer



32x32x3 image

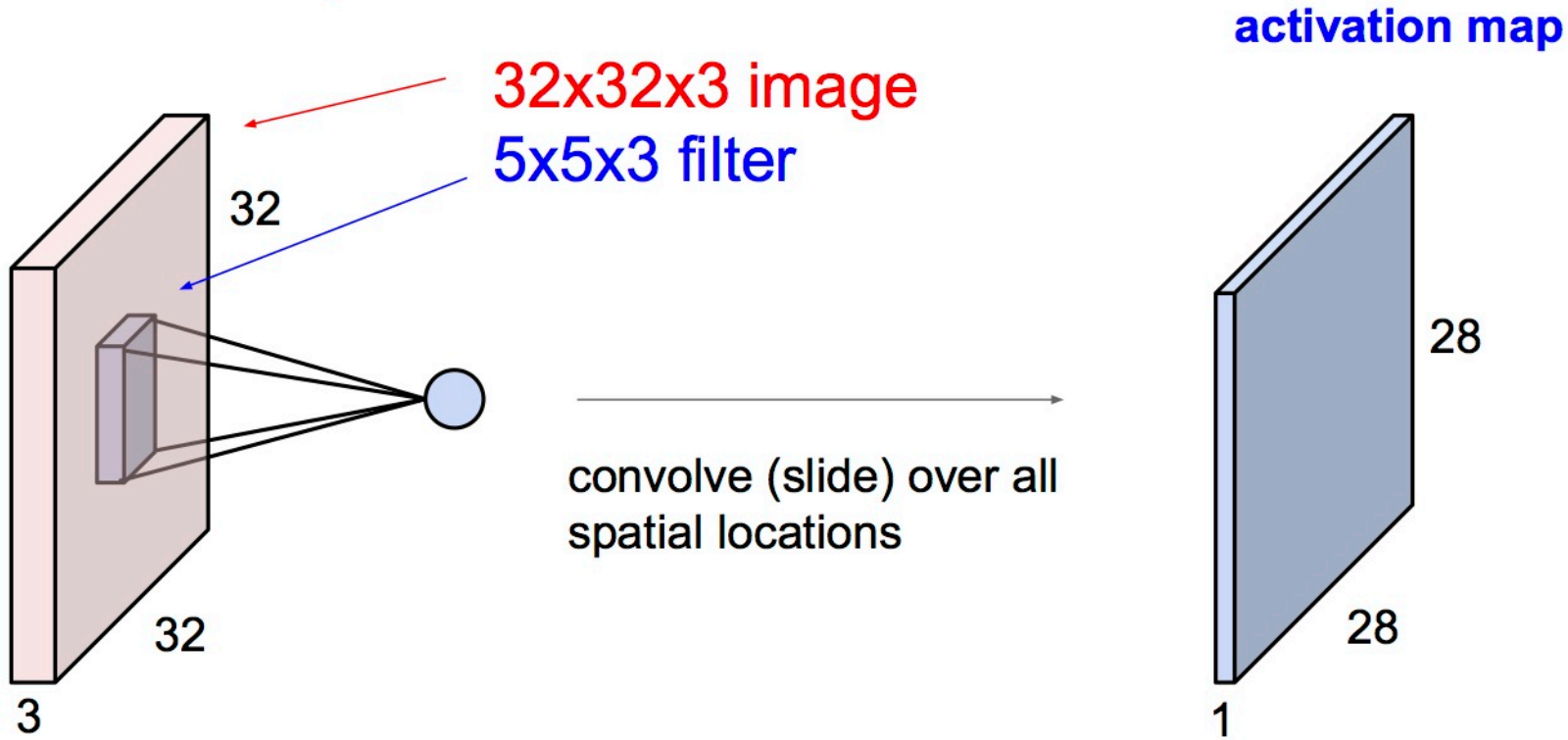
5x5x3 filter w

1 number:
the result of taking a dot product between the filter and a small 5x5x3 chunk of the image
(i.e. 5*5*3 = 75-dimensional dot product + bias)

$$w^T x + b$$

Source: Fei-Feli Li & Justin Johnson & Serena Yeung

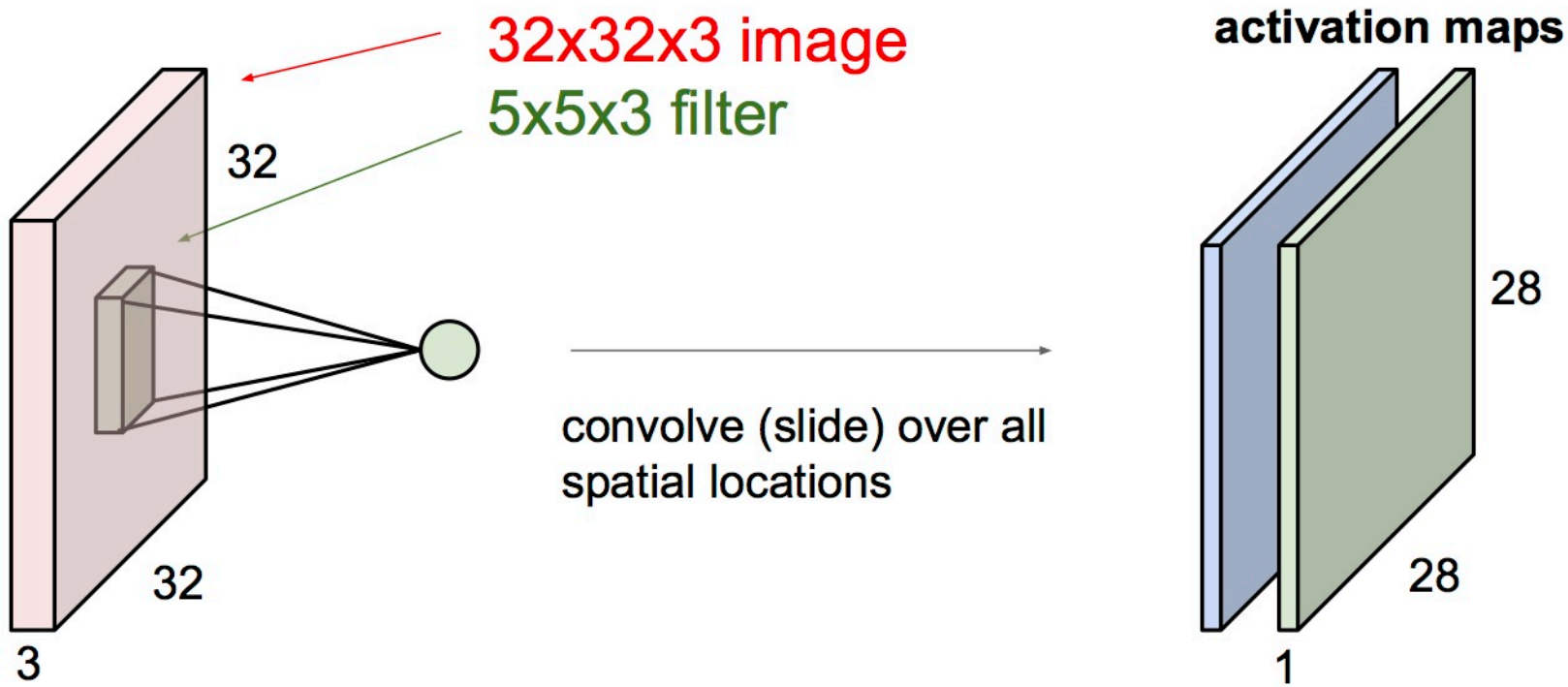
Convolution Layer



Source: Fei-Feli Li & Justin Johnson & Serena Yeung

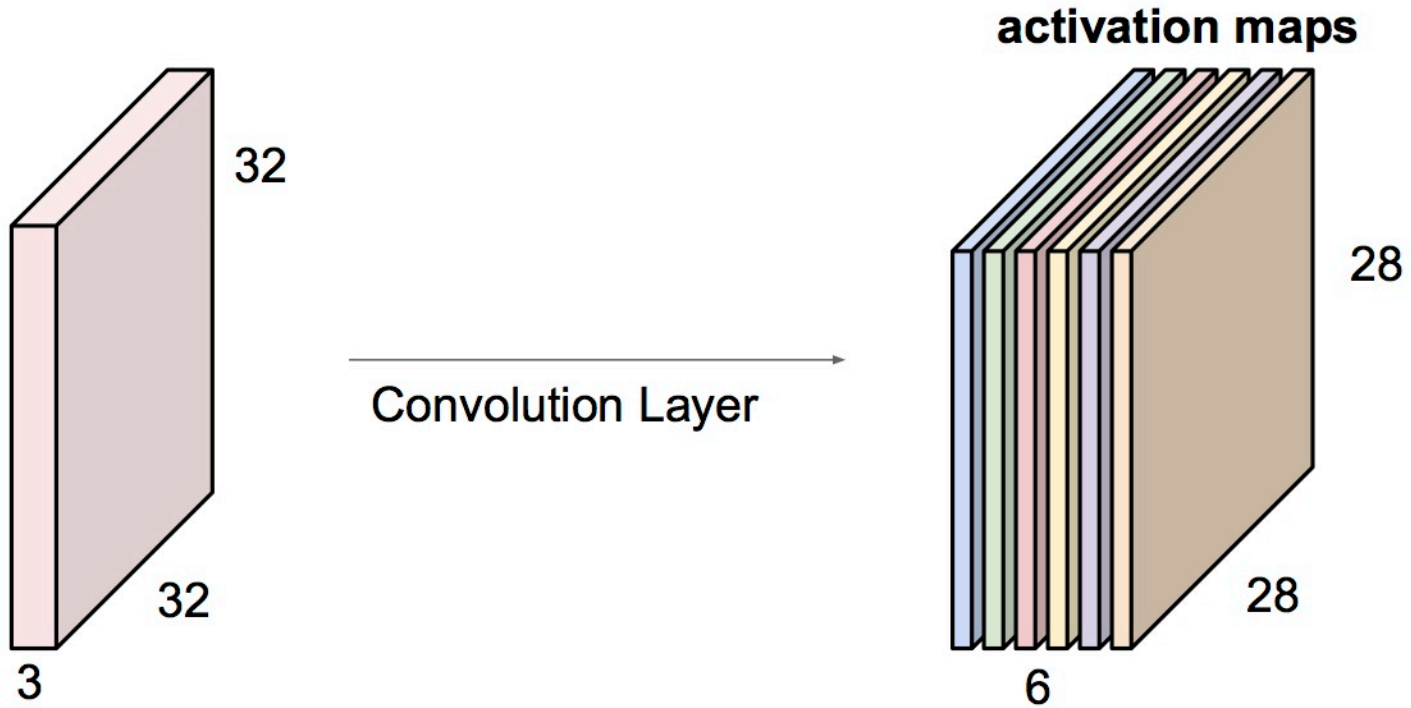
Convolution Layer

consider a second, green filter



Source: Fei-Feli Li & Justin Johnson & Serena Yeung

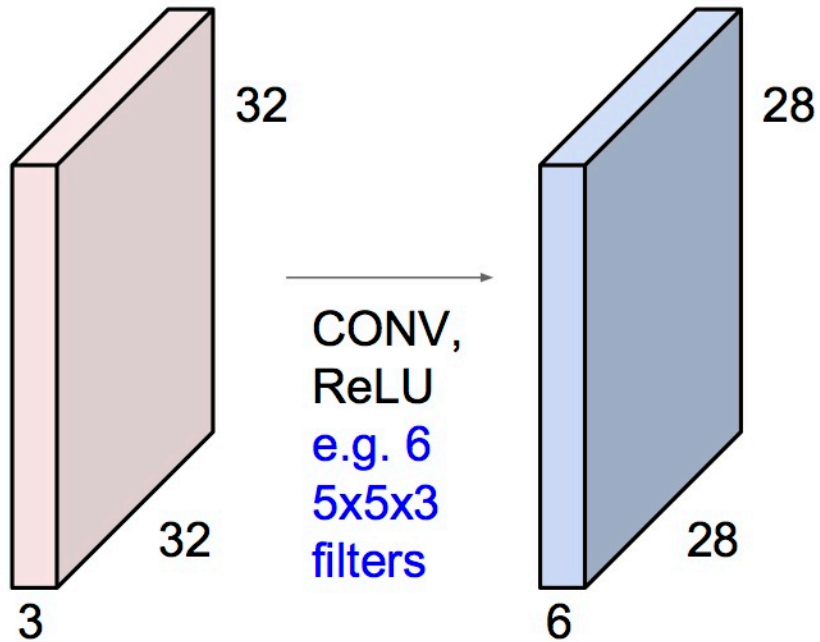
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



We stack these up to get a "new image" of size 28x28x6!

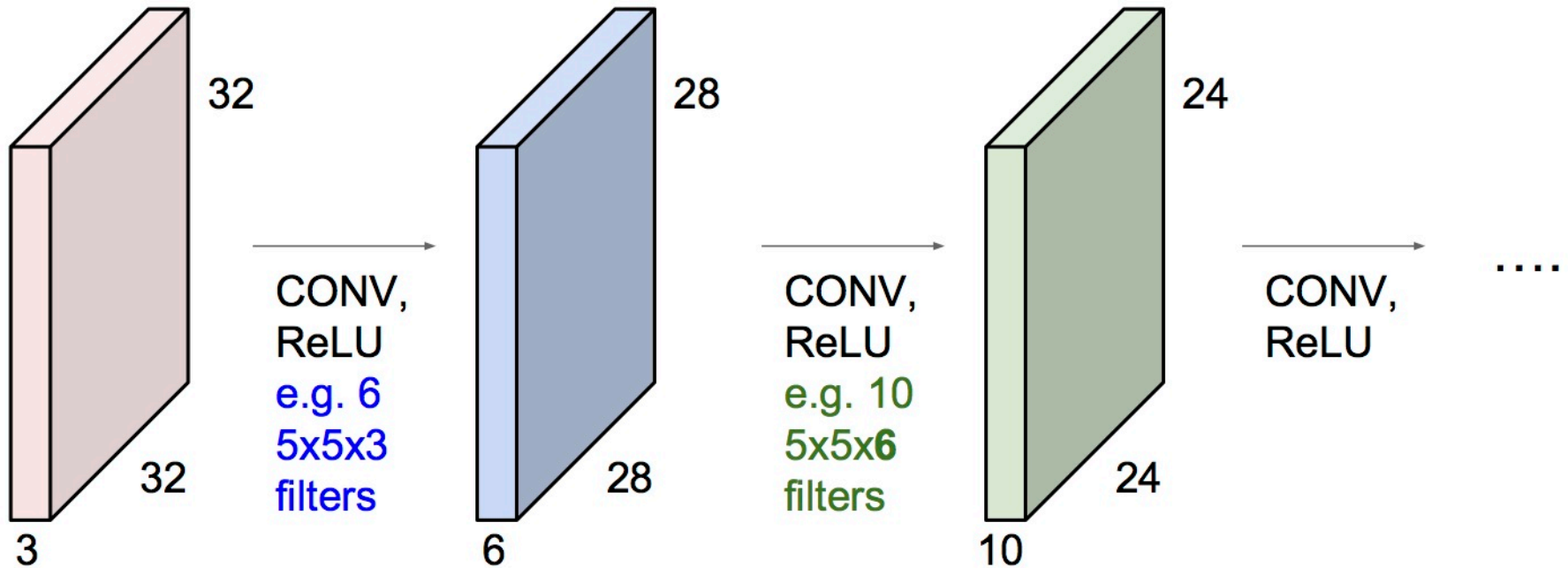
Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



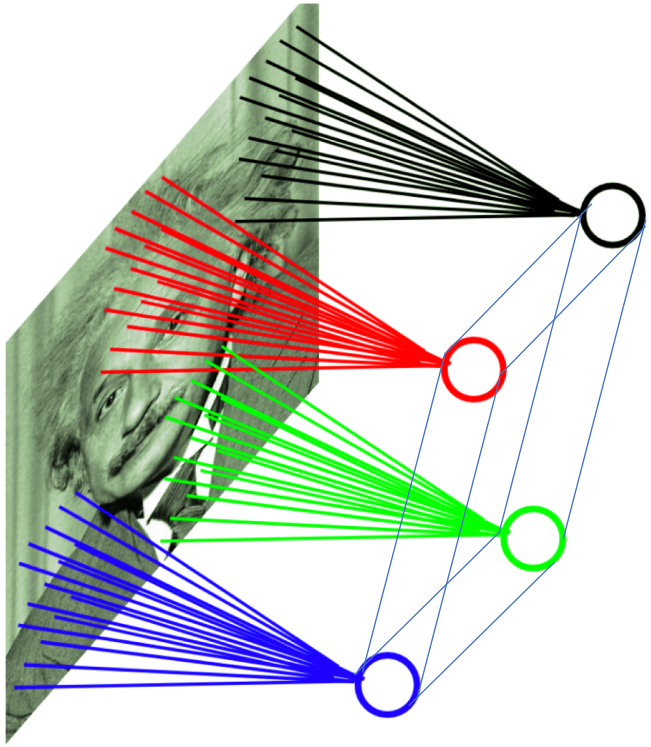
Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Source: Fei-Feli Li & Justin Johnson & Serena Yeung

Parameter Sharing



One filter to rule them all!

- Image Classification & Recognition: A main task in computer vision
  - PASCAL
  - IMAGENET

Pop Quiz

- How many parameters do we have in a conv layer with 100 filters that are 3x3x3 dimensional each?

- For one filter: we have 3x3x3 = 27 filter weights; and + 1 bias = **28** parameters.
- For all 100 filters: we have **28** x 100 = **2800 parameters** to learn for that layer!

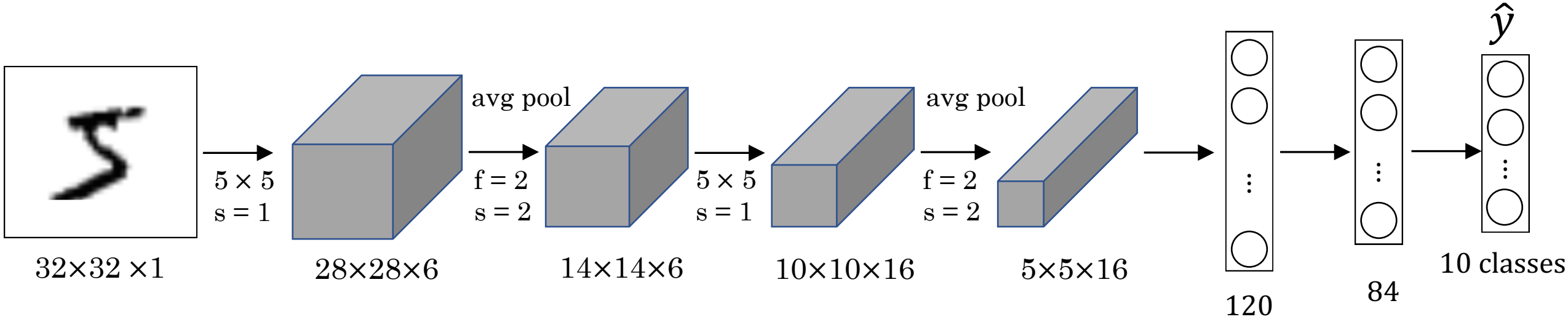
Pop Quiz

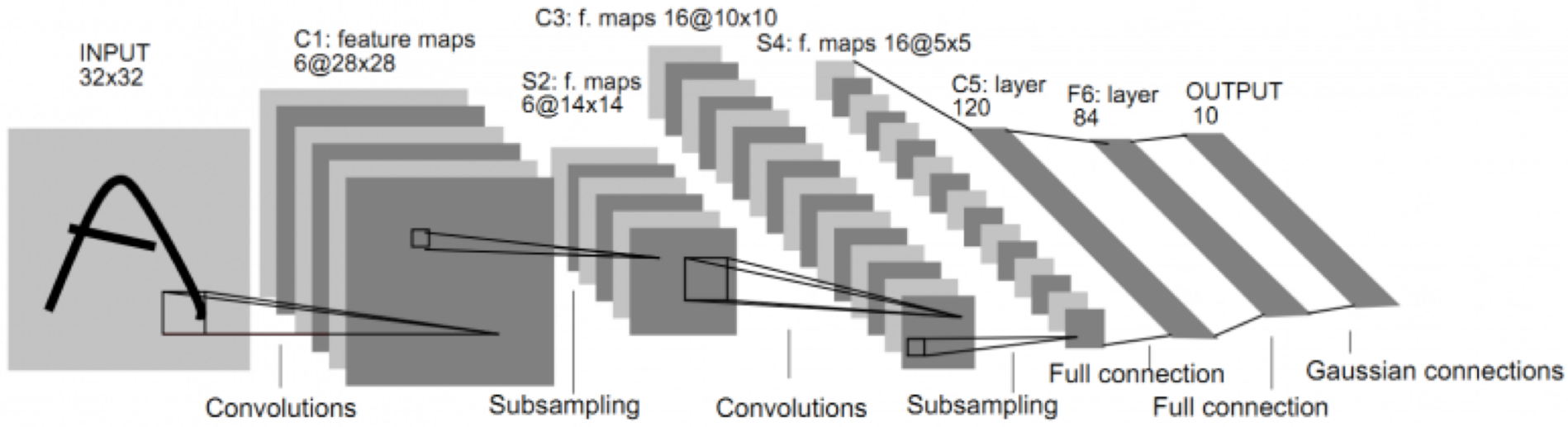
- Imagine that you are designing a CNN with 5 layers. The first 4 layers are convolutional and the last layer is FC layer with one neuron (using logistic reg).
  - The input dims are: 600x600x3.
  - In each of the 4 conv. layers:
    - We have 10 filters (for each, f = 3)
    - Stride is 2
    - Padding is “valid”.

What is the output dims at the end of the 5^{th} layer?

30 seconds to return your answers! ☺

(Slightly Modified) LeNet-5

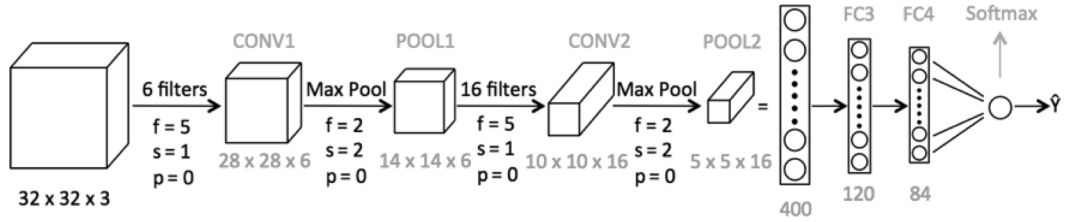




Original LeNet-5

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

(Slightly Modified) LeNet-5



For: Grayscale image (single channel)

	Activation shape	Activation Size	# parameters
Input:	(32,32,**1**)	1,024	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	150 + 6 bias = **156**
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	(5x5x6) x 16= 2400 2400 + 16 = **2416**
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,000 + 120 = **48,120**
FC4	(84,1)	84	10,080 + 84 = **10,164**
Softmax	(10,1)	10	84x10 + 10 = **850**

For: Colored (RGB) image

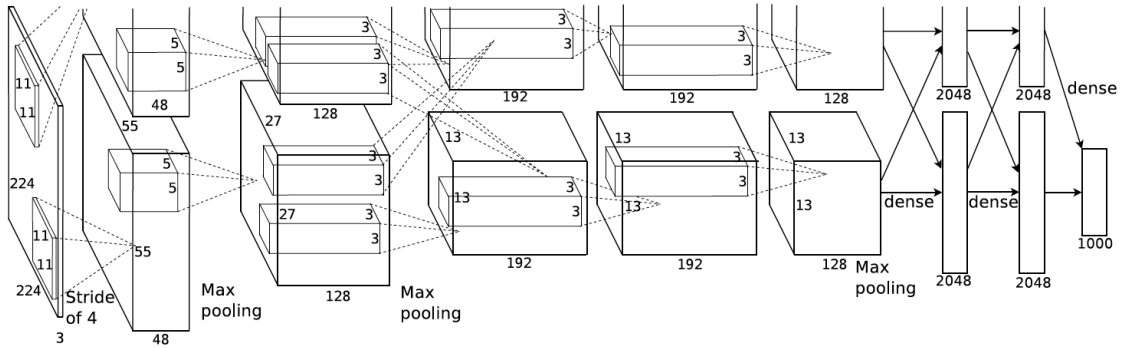
	Activation shape	Activation Size	# parameters
Input:	(32,32,**3**)	3,072	0
CONV1 (f=5, s=1)	(28,28,6)	4,704	450 + 6 bias = **456**
POOL1	(14,14,6)	1,176	0
CONV2 (f=5, s=1)	(10,10,16)	1,600	(5x5x6) x 16= 2400 2400 + 16 = **2416**
POOL2	(5,5,16)	400	0
FC3	(120,1)	120	48,000 + 120 = **48,120**
FC4	(84,1)	84	10,080 + 84 = **10,164**
Softmax	(10,1)	10	84x10 + 10 = **850**

Tensorflow output for the (grayscale) model summarizing the model parameters:

```
Variables: name (type shape) [size]
---------
Variable:0 (float32_ref 5x5x1x6) [150, bytes: 600]
conv1_biases:0 (float32_ref 6) [6, bytes: 24]
Variable_1:0 (float32_ref 5x5x6x16) [2400, bytes: 9600]
conv2_biases:0 (float32_ref 16) [16, bytes: 64]
Variable_2:0 (float32_ref 400x120) [48000, bytes: 192000]
fc1_biases:0 (float32_ref 120) [120, bytes: 480]
Variable_3:0 (float32_ref 120x84) [10080, bytes: 40320]
fc2_biases:0 (float32_ref 84) [84, bytes: 336]
Variable_4:0 (float32_ref 84x10) [840, bytes: 3360]
fc3_biases:0 (float32_ref 10) [10, bytes: 40]
Total size of variables: 61706
Total bytes of variables: 246824
sedat@sedat-ThinkPad-P50:~/
```

(Slightly modified) AlexNet

Original



227×227 ×3

11 × 11
s = 4

55×55 ×96

MAX-POOL

3 × 3
s = 2

27×27 ×96

5 × 5
same

27×27 ×256

MAX-POOL

3 × 3
s = 2

13×13 ×256

3 × 3
same

13×13 ×384

3 × 3
same

13×13 ×384

3 × 3
same

13×13 ×256

MAX-POOL

3 × 3
s = 2

6×6 ×256

=

9216

4096

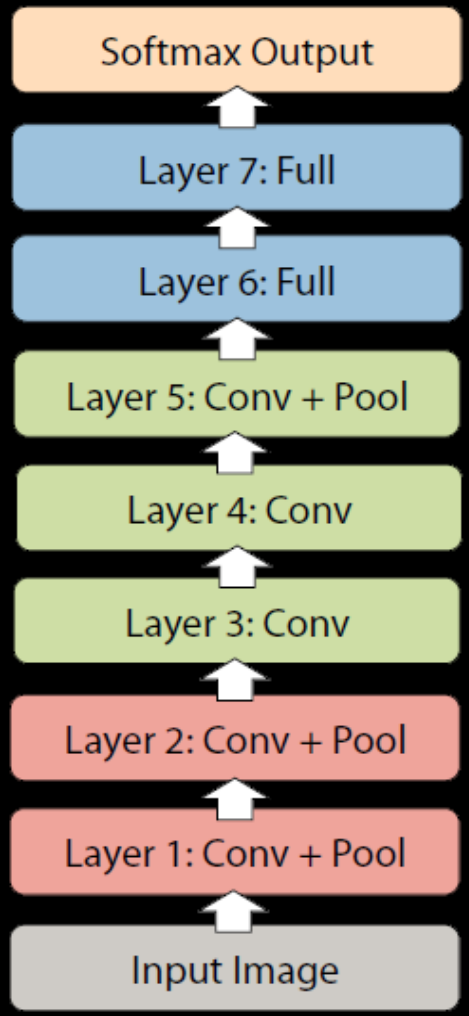
4096

Softmax
1000

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

Why ConvNet should be Deep?

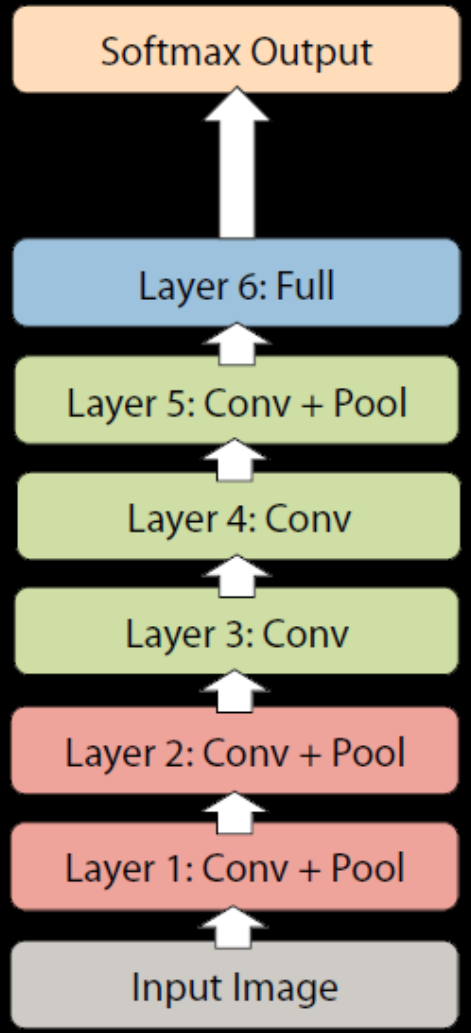
- 8 layers total
- Trained on Imagenet dataset [Deng et al. CVPR'09]
- 18.2% top-5 error
- Our reimplementation:
  18.1% top-5 error



Rob Fergus, NIPS 2013

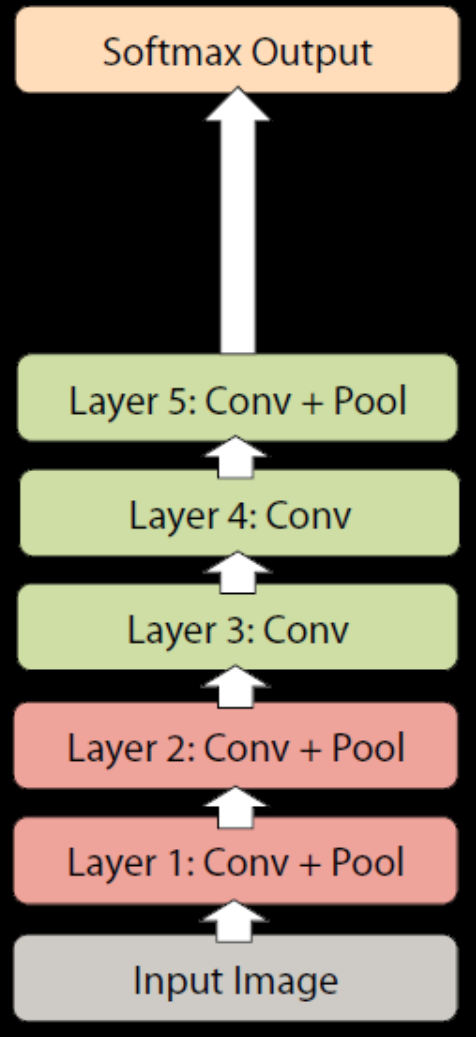
Why ConvNet should be Deep?

- Remove top fully connected layer
  - Layer 7
- Drop 16 million parameters
- Only 1.1% drop in performance!



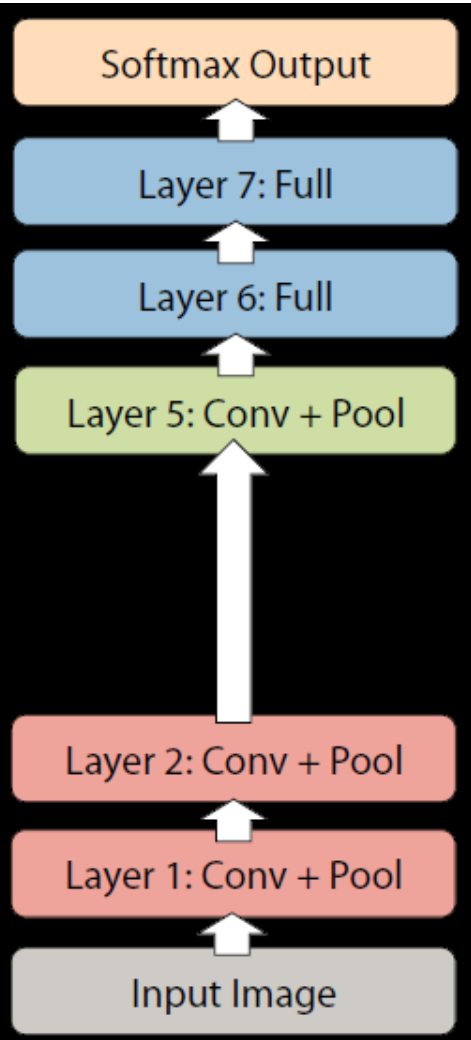
Why ConvNet should be Deep?

- Remove both fully connected layers
  - Layer 6 & 7
- Drop ~50 million parameters
- 5.7% drop in performance



Why ConvNet should be Deep?

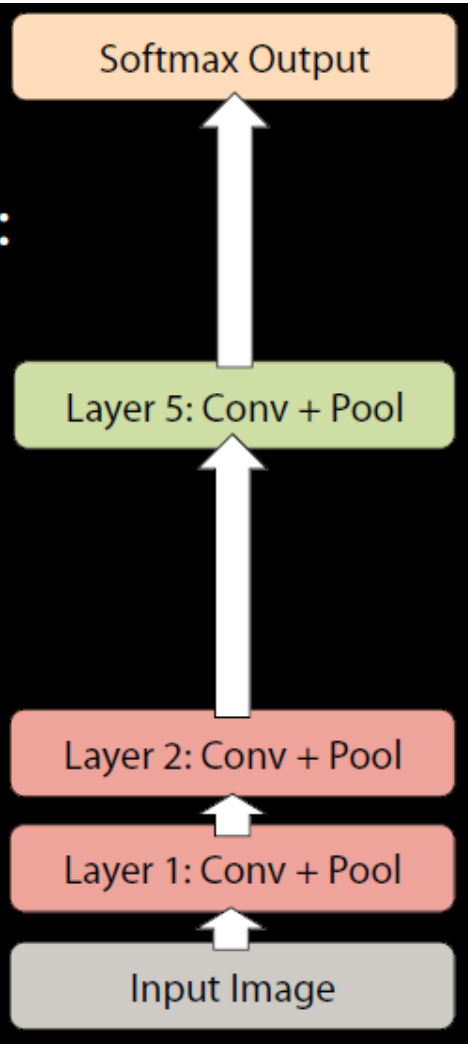
- Now try removing upper feature extractor layers:
  - Layers 3 & 4
- Drop ~1 million parameters
- 3.0% drop in performance



Why ConvNet should be Deep?

- Now try removing upper feature extractor layers & fully connected:
  - Layers 3, 4, 6 ,7
- Now only 4 layers
- 33.5% drop in performance

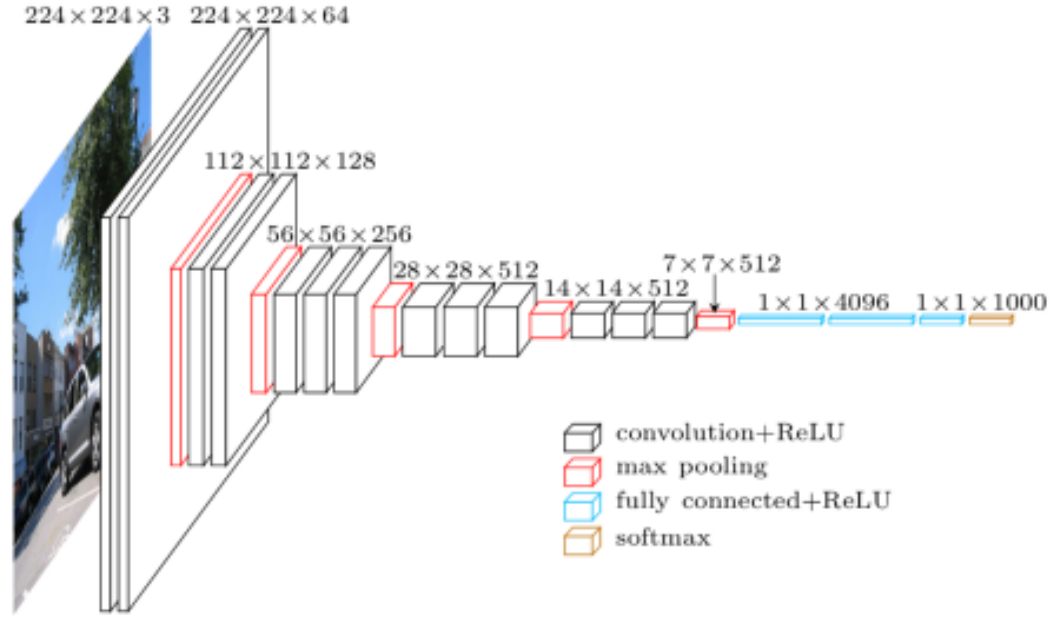
→Depth of network is key

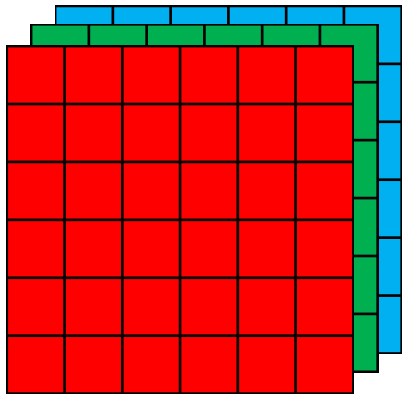


VGG – 16 (16 layers)

CONV = 3×3 filter, s = 1, “**same” padding**

MAX-POOL = 2×2 , s = 2





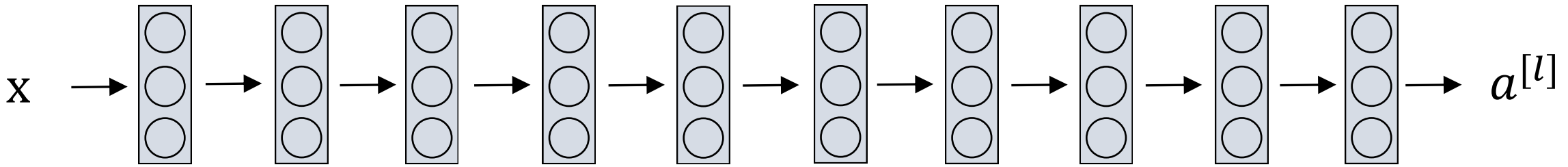
224×224 ×3 → [CONV: 64] ×2 layers → 224×224×64 → POOL → 112×112 ×64 → [CONV: 128] ×2 layers → 112×112 ×128 → POOL → 56×56 ×128

→ [CONV: 256] ×3 layers → 56×56 ×256 → POOL → 28×28 ×256 → [CONV: 512] ×3 layers → 28×28 ×512 → POOL → 14×14×512

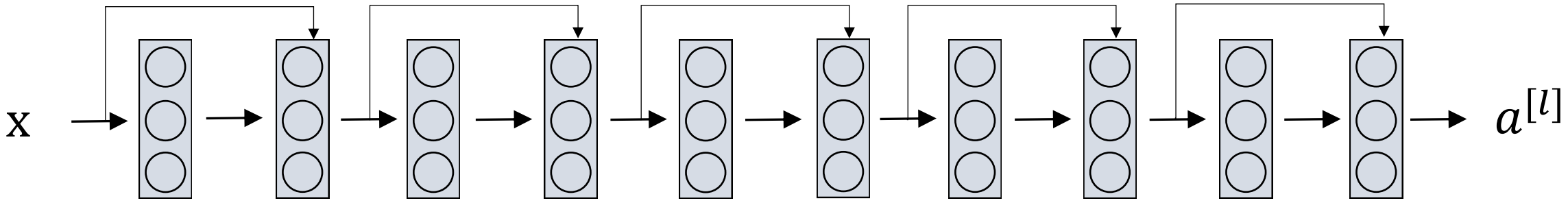
→ [CONV: 512] ×3 layers → 14×14 ×512 → POOL → 7×7×512 → FC 4096 → FC 4096 → Softmax 1000

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

Residual Network - ResNet (Microsoft)



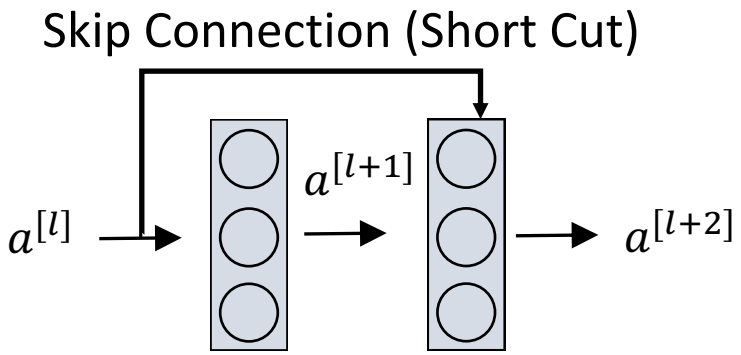
A “plain” (deep) network

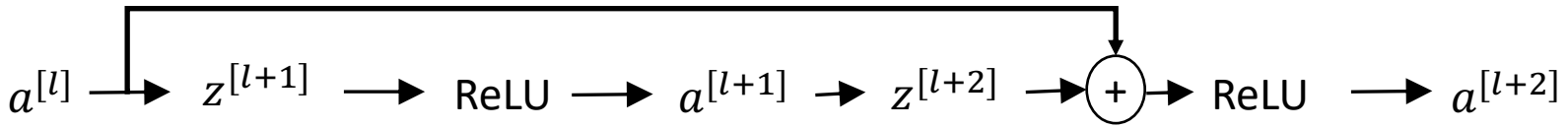


A residual network (with 5 residual blocks)

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

Residual Building Block



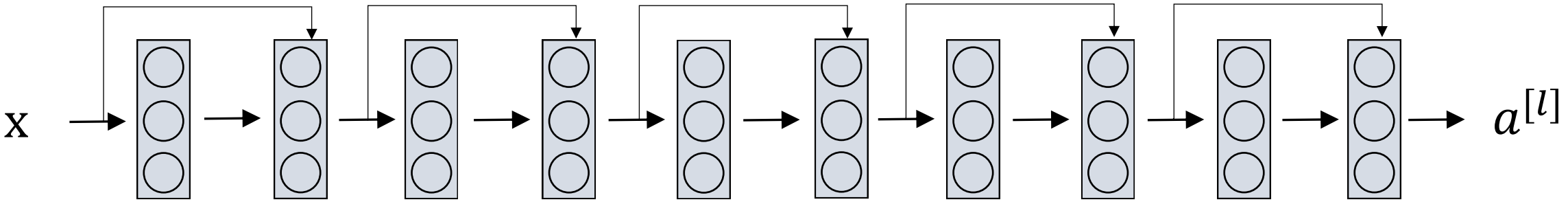


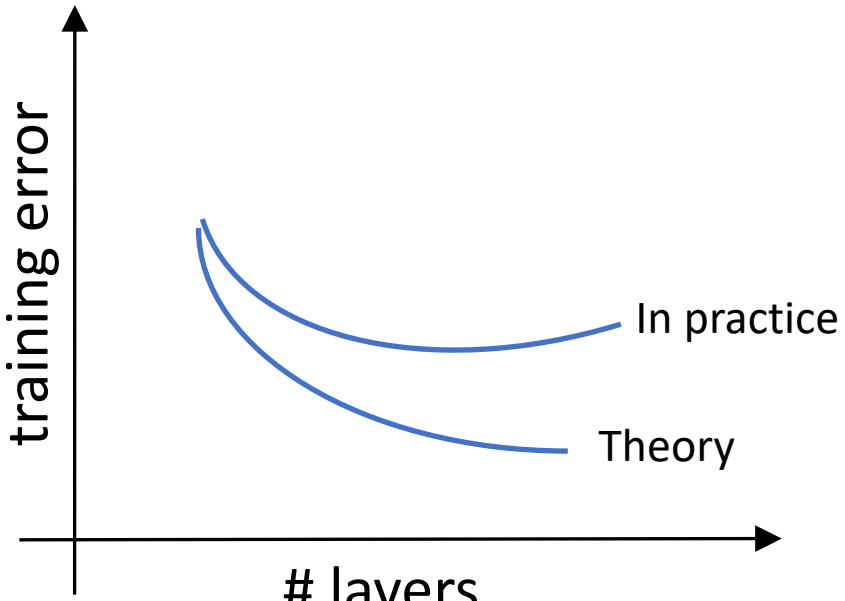
$$z^{[l+1]} = W^{[l+1]}\, a^{[l]} + b^{[l+1]} \qquad a^{[l+1]} = g(z^{[l+1]}) \qquad z^{[l+2]} = W^{[l+2]} a^{[l+1]} + b^{[l+2]} \qquad a^{[l+2]} = g(z^{[l+2]})$$

With skip connection: $a^{[l+2]} = g(z^{[l+2]} + a^{[l]})$

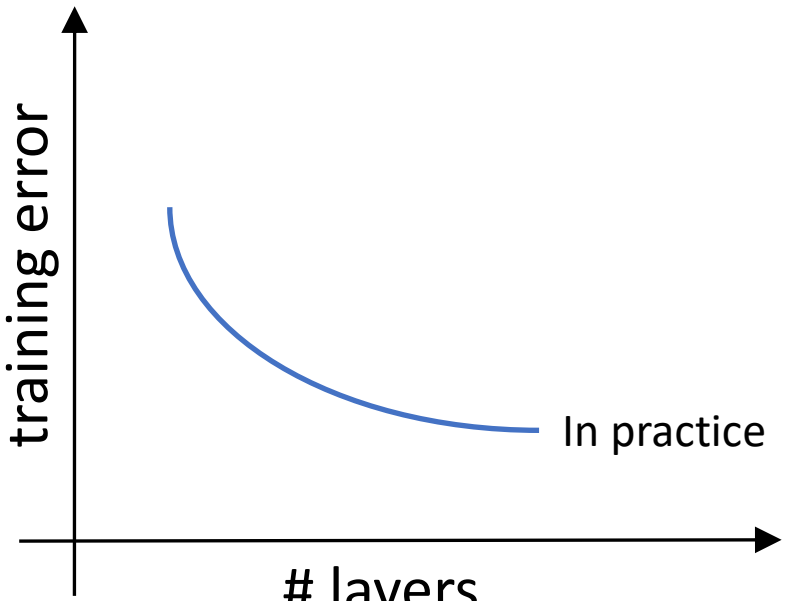
[He et al., 2015. Deep residual networks for image recognition]

Residual Network





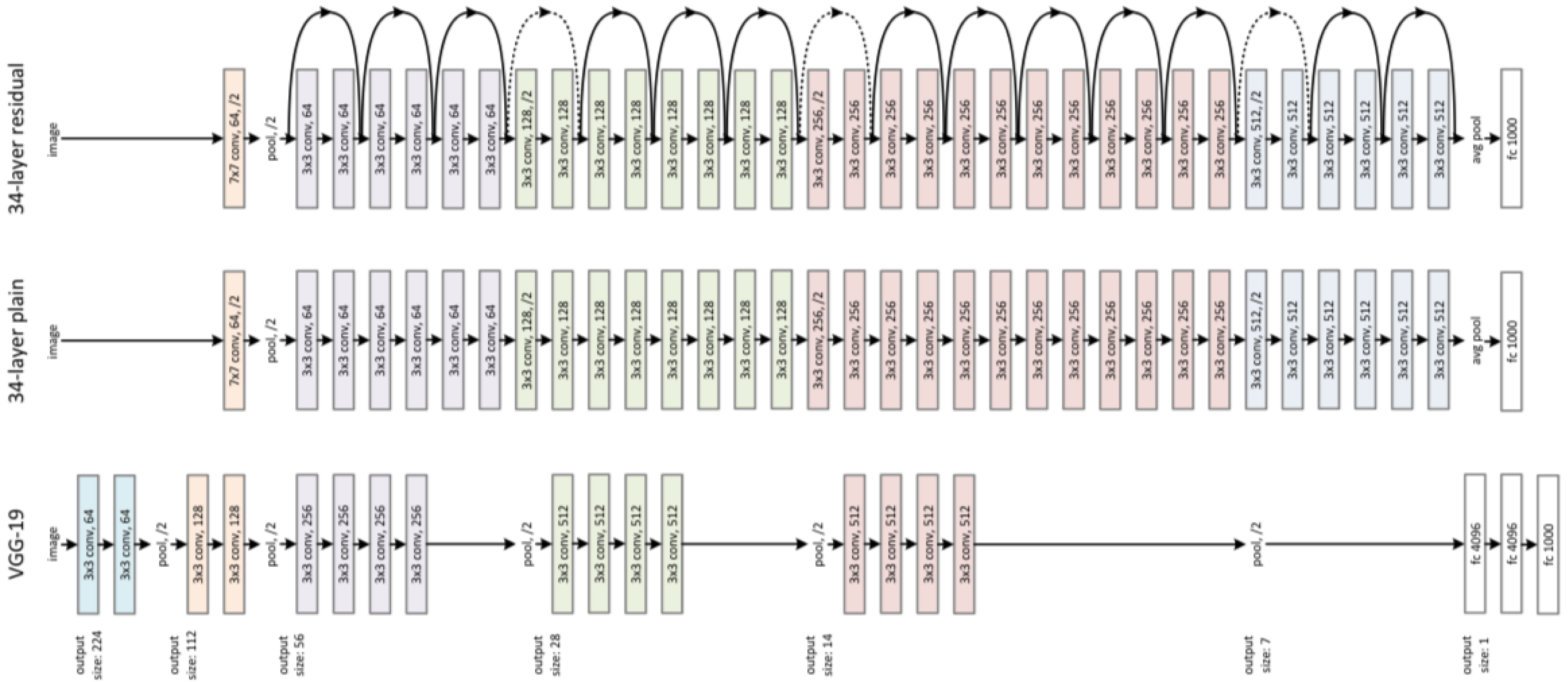
Plain



ResNet

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

ResNet



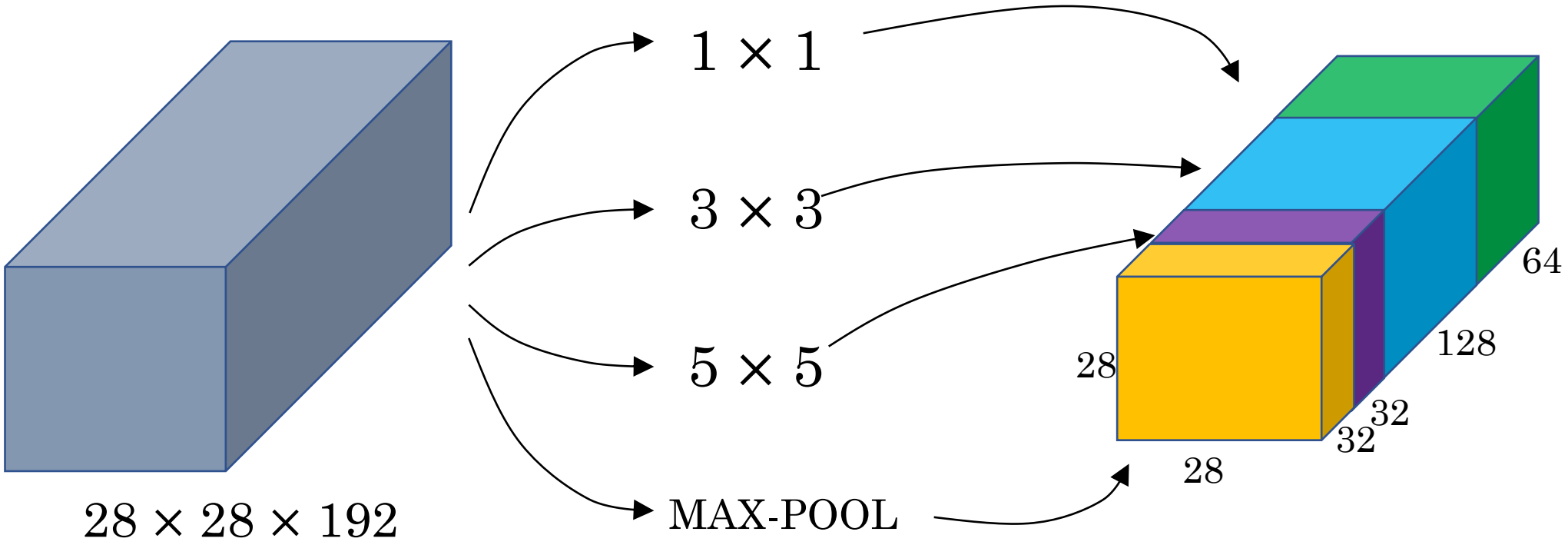
[Source: He et al., 2015. Deep residual networks for image recognition]

Inception Network

(GoogleNet)

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

Main motivation for inception network



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

What does a 1 × 1 convolution do?

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

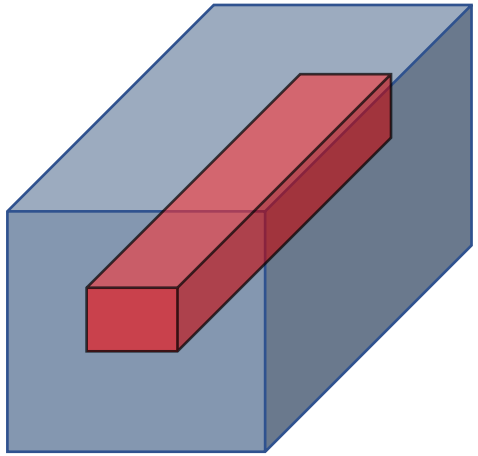
6 × 6

*

2

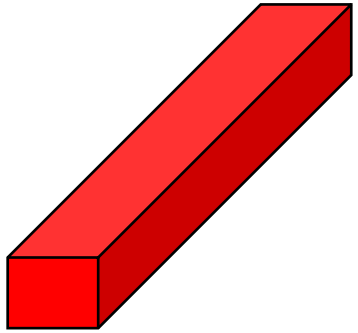
=

6	2	6	.	.	.
6	4	8	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.
.	.	.	.	.	.



6 × 6 × 32

*



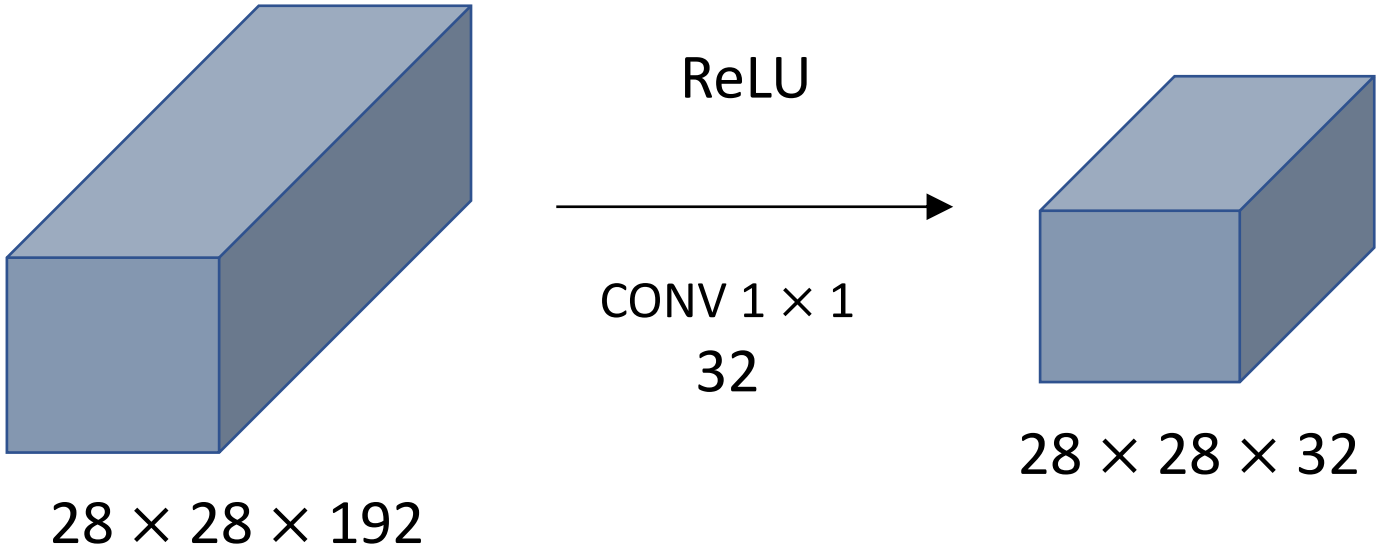
1 × 1 × 32

=

6 × 6 × # filters

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

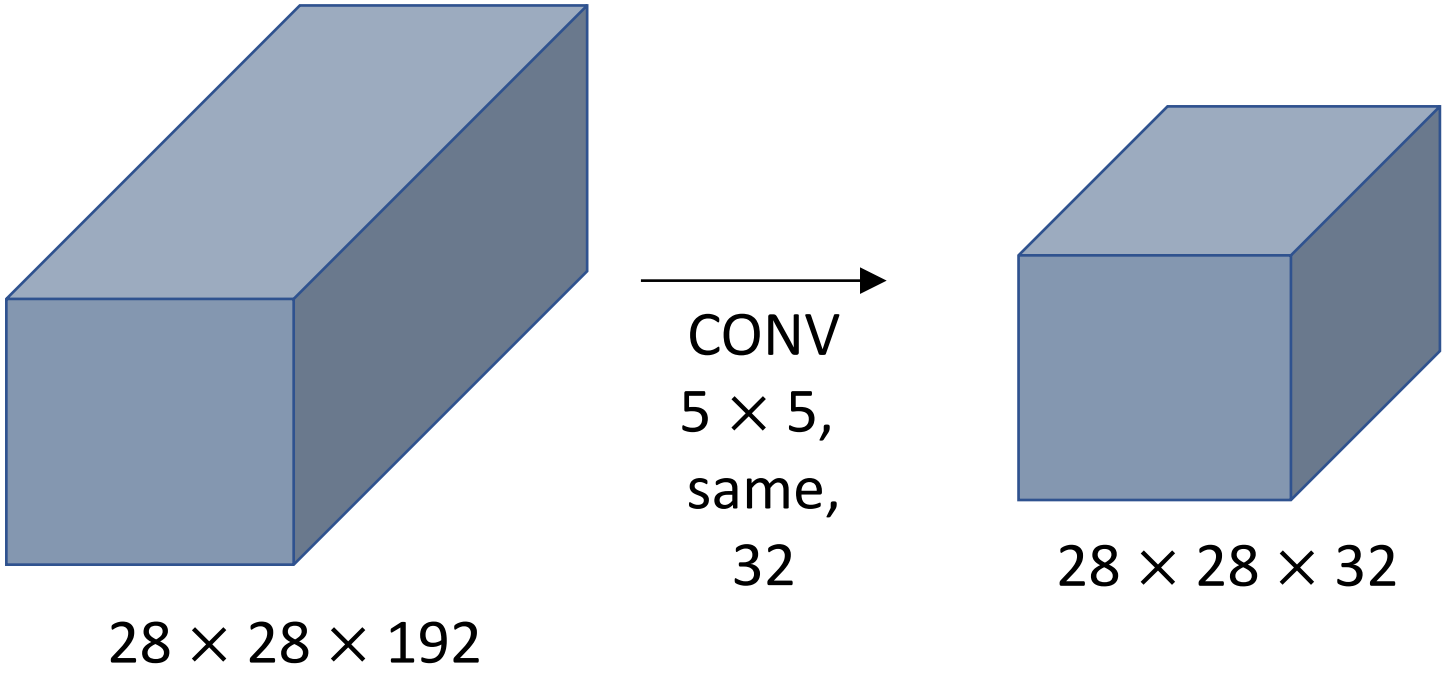
1×1 convolutions can shrink the layer dims



Helps reducing the number of parameters!

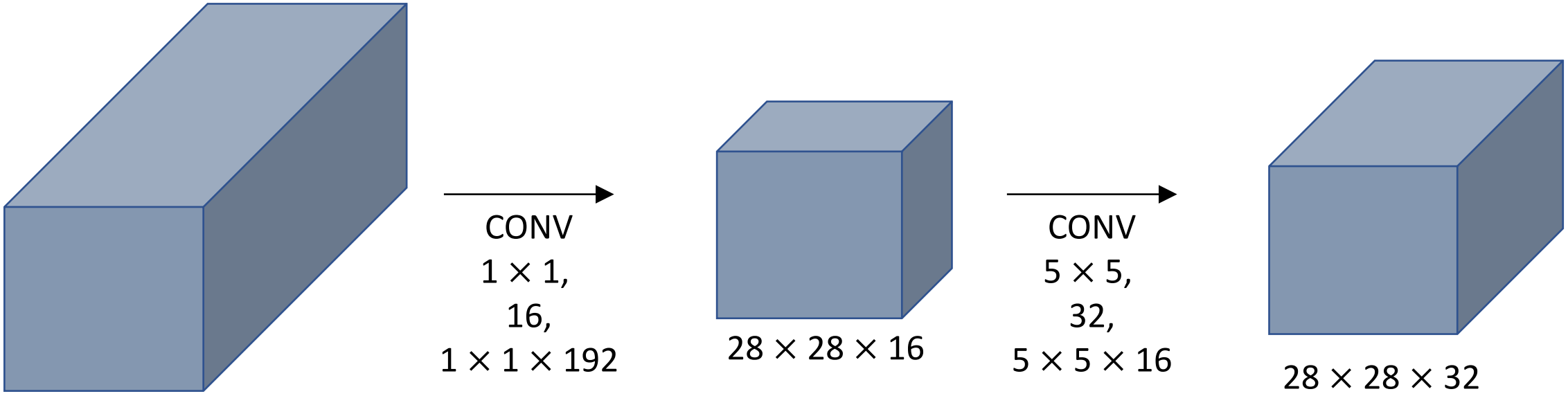
[Lin et al., 2013. Network in network]

The problem of computational cost



- One filter volume: 5x5x192 for one voxel in the output volume (in the activation map).
- Since we have 28x28x32 dimensional volume at the output, we need:
  (5x5x192) x (28x28x32) multiplications! (~120Million)

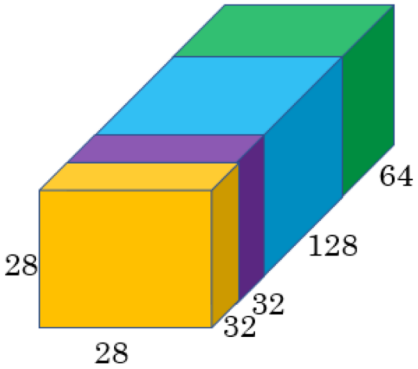
Using 1×1 convolution for reducing the computation



Multiplications needed: (28x28x16) x (1x1x192) = ~ 2.4 Million (28x28x32) x (5x5x16) = ~ 10 Million

Total: 12.4 Million << 120Million

Inception module



Previous Activation

1 × 1 CONV

1 × 1 CONV

3 × 3 CONV

1 × 1 CONV

5 × 5 CONV

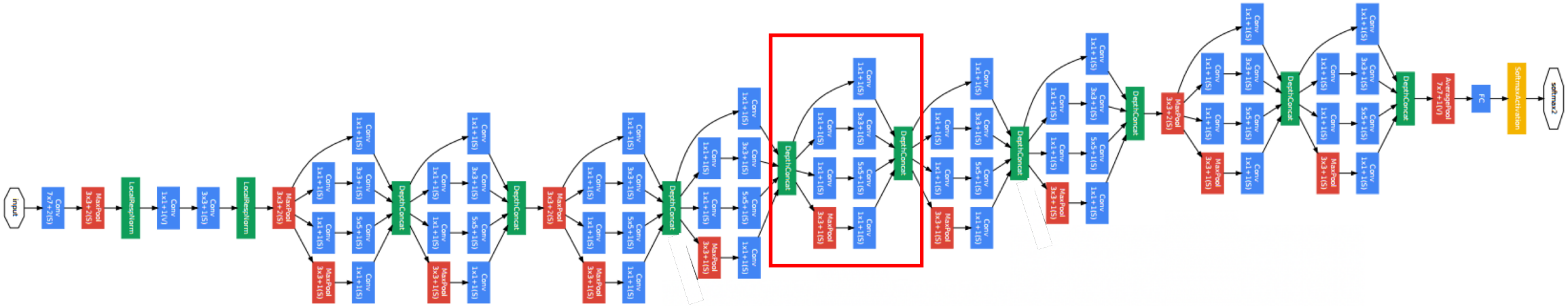
MAXPOOL 3 × 3,s = 1 same

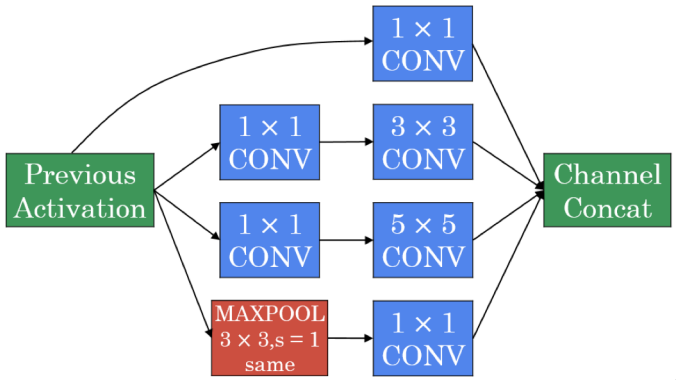
1 × 1 CONV

Channel Concat

Inception Network (Google)

GooLeNet or GoogLeNet





Local Response Normalization (LRN)

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

- This lecture has materials from CS231n (Stanford), Andrew Ng and Ulas Bagci.