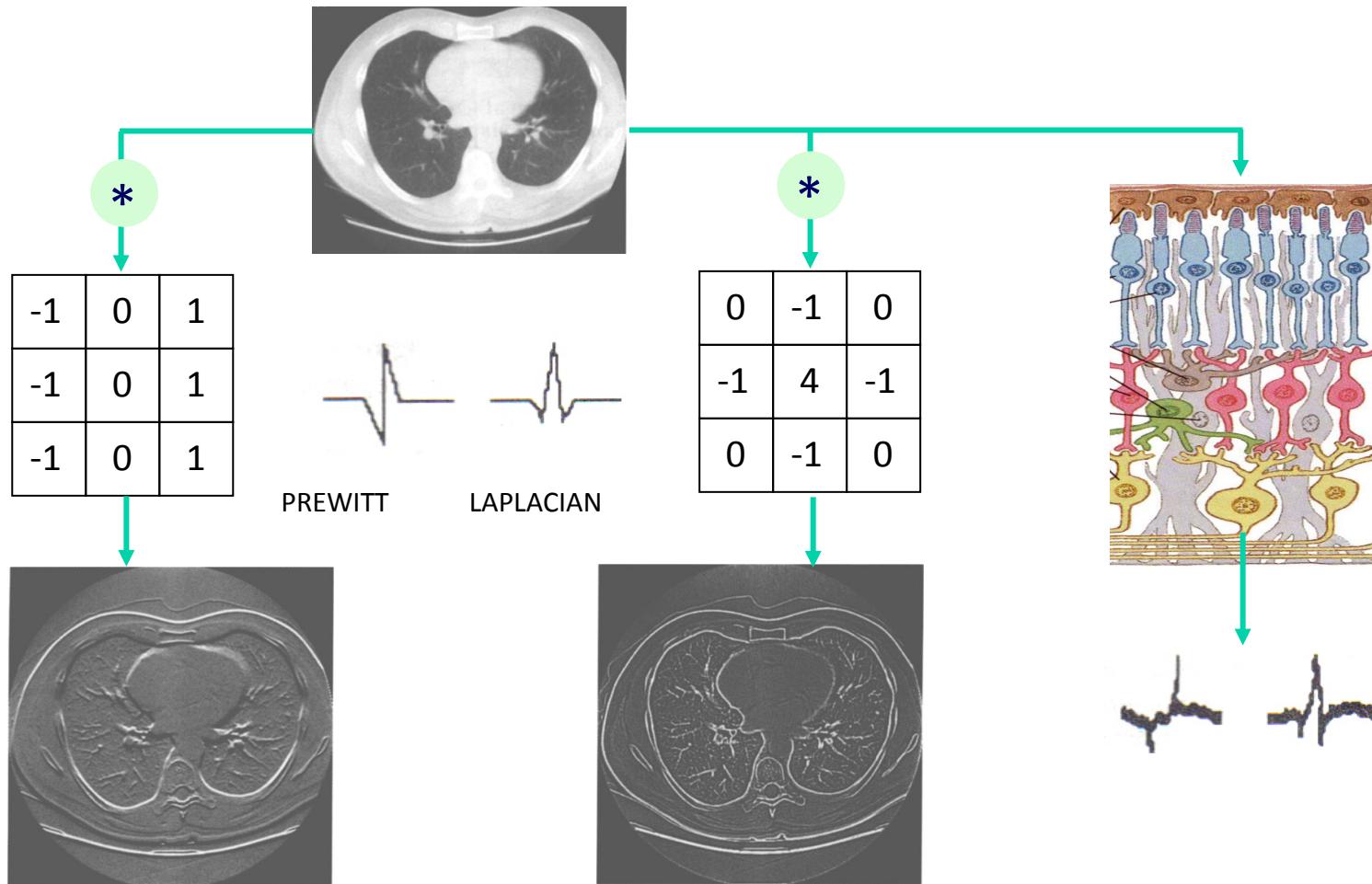


Processamento e Análise de Imagens

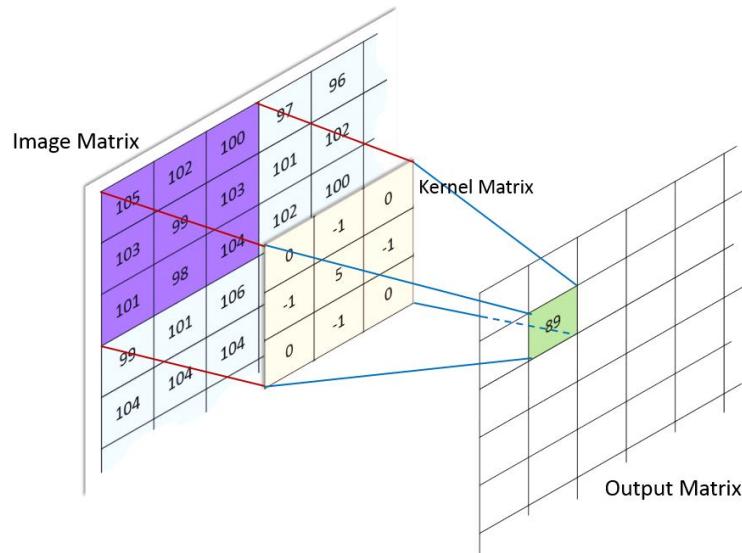
Convolutional Neural Networks

Prof. Alexei Machado
PUC Minas

Convolutional Neural Networks



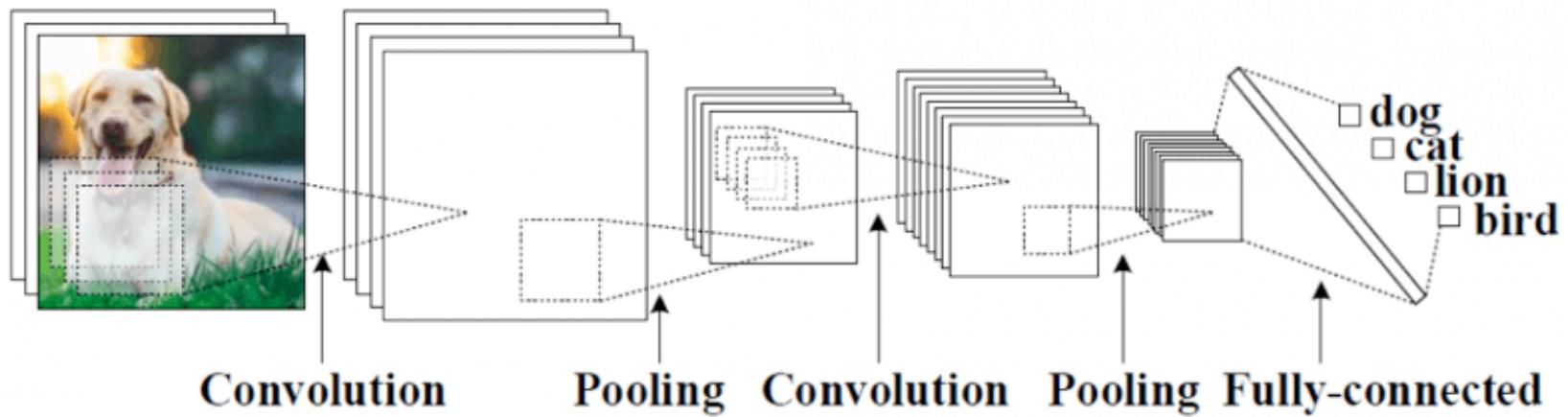
Convolutional Neural Networks



machinelearningguru.com

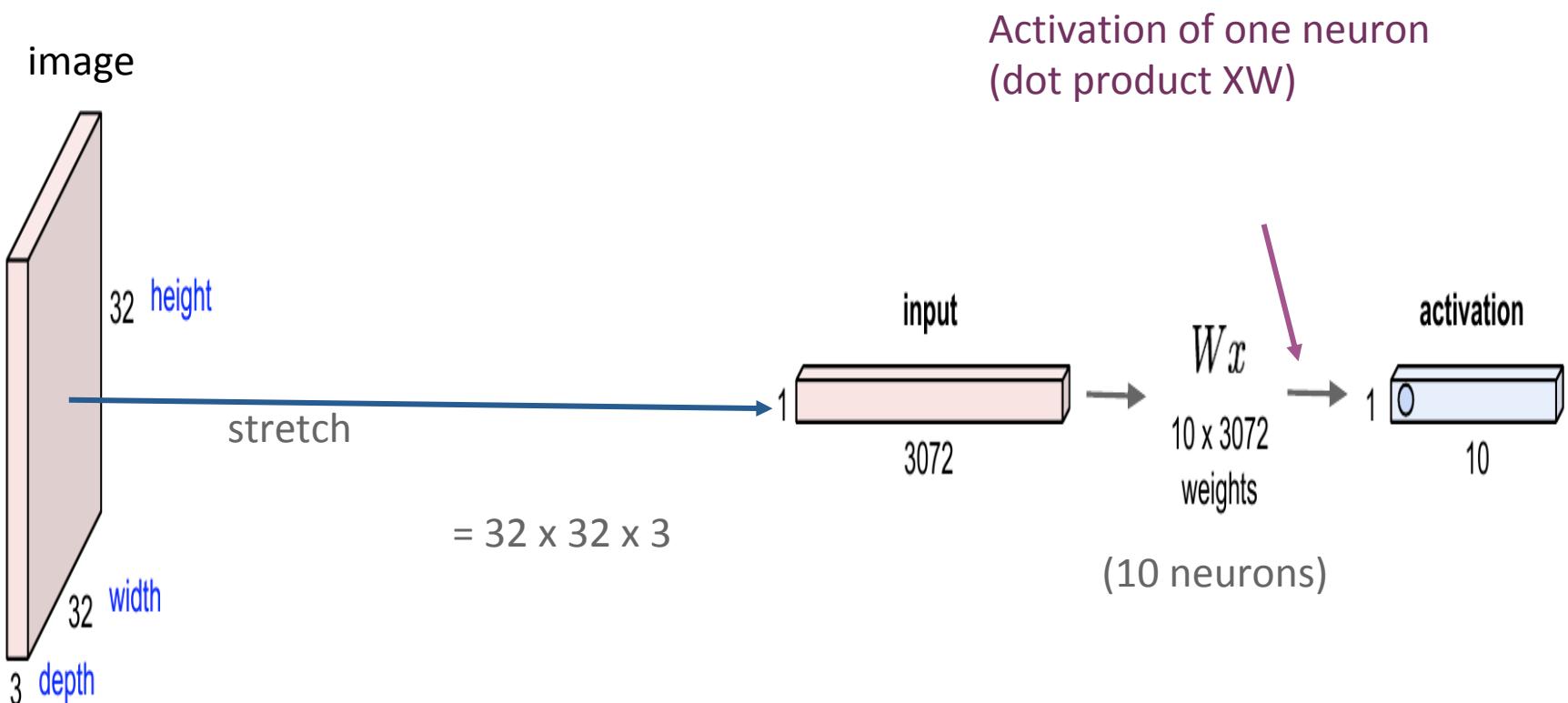
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1, n_2] \cdot g[x - n_1, y - n_2]$$

Convolutional Neural Networks

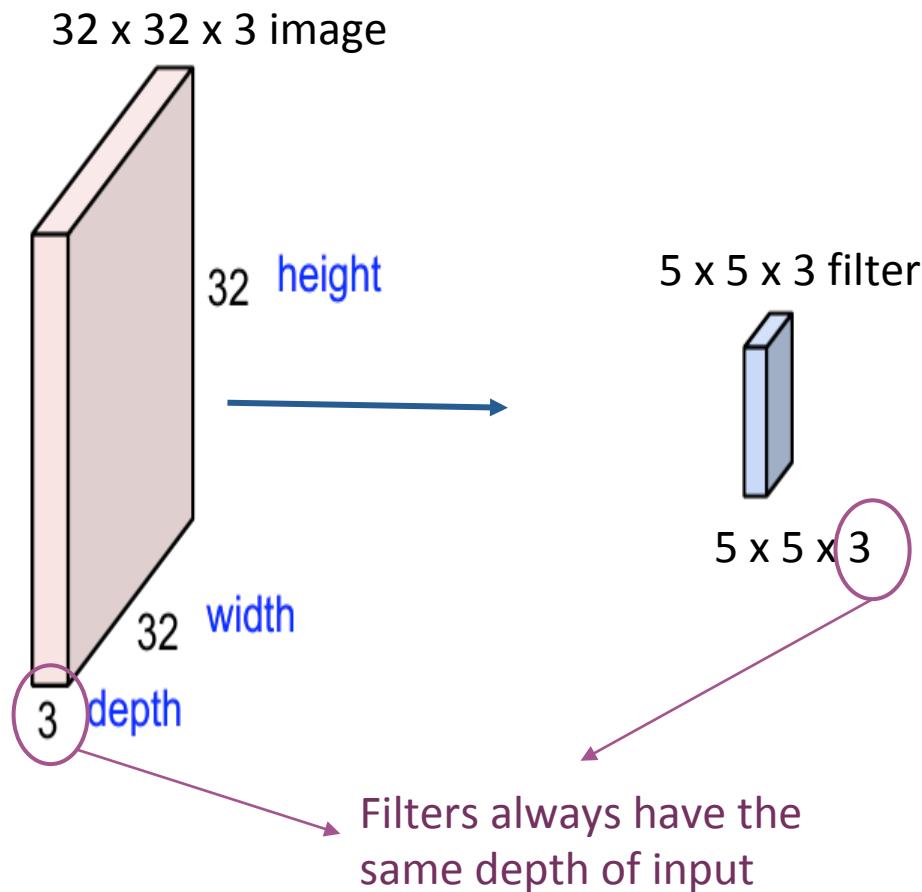


www.ayasdi.com

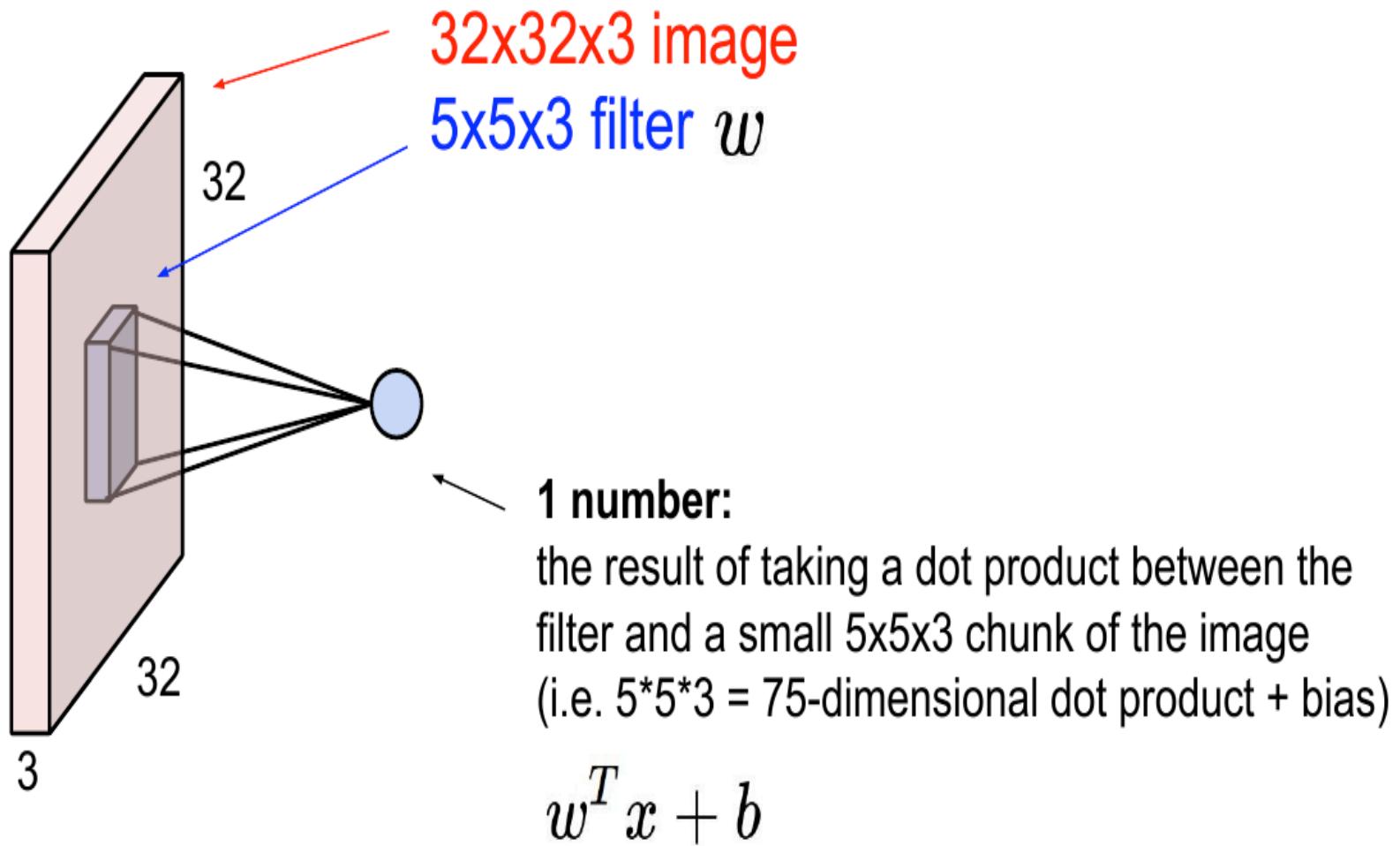
Fully-connected layers



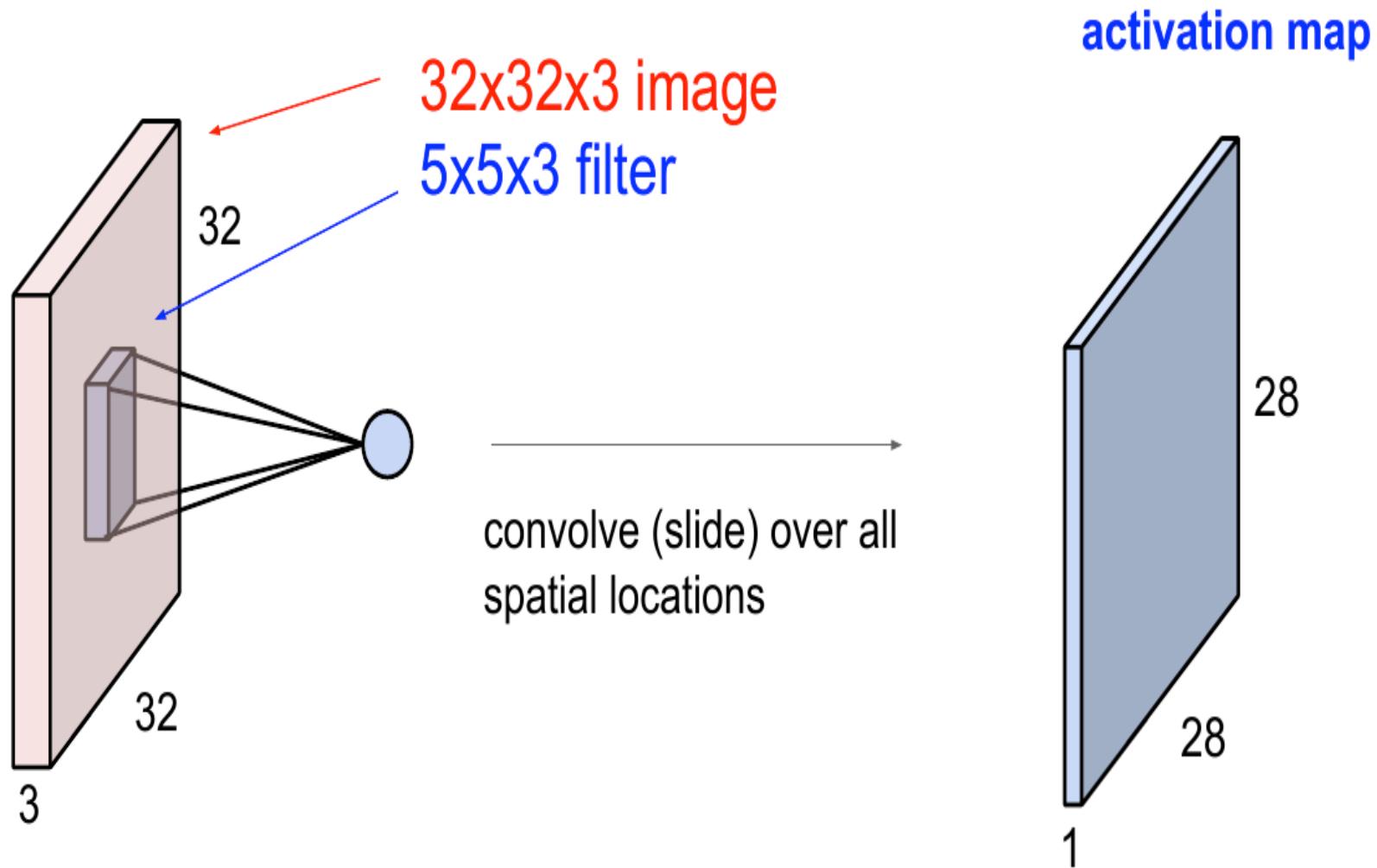
Convolutional layers



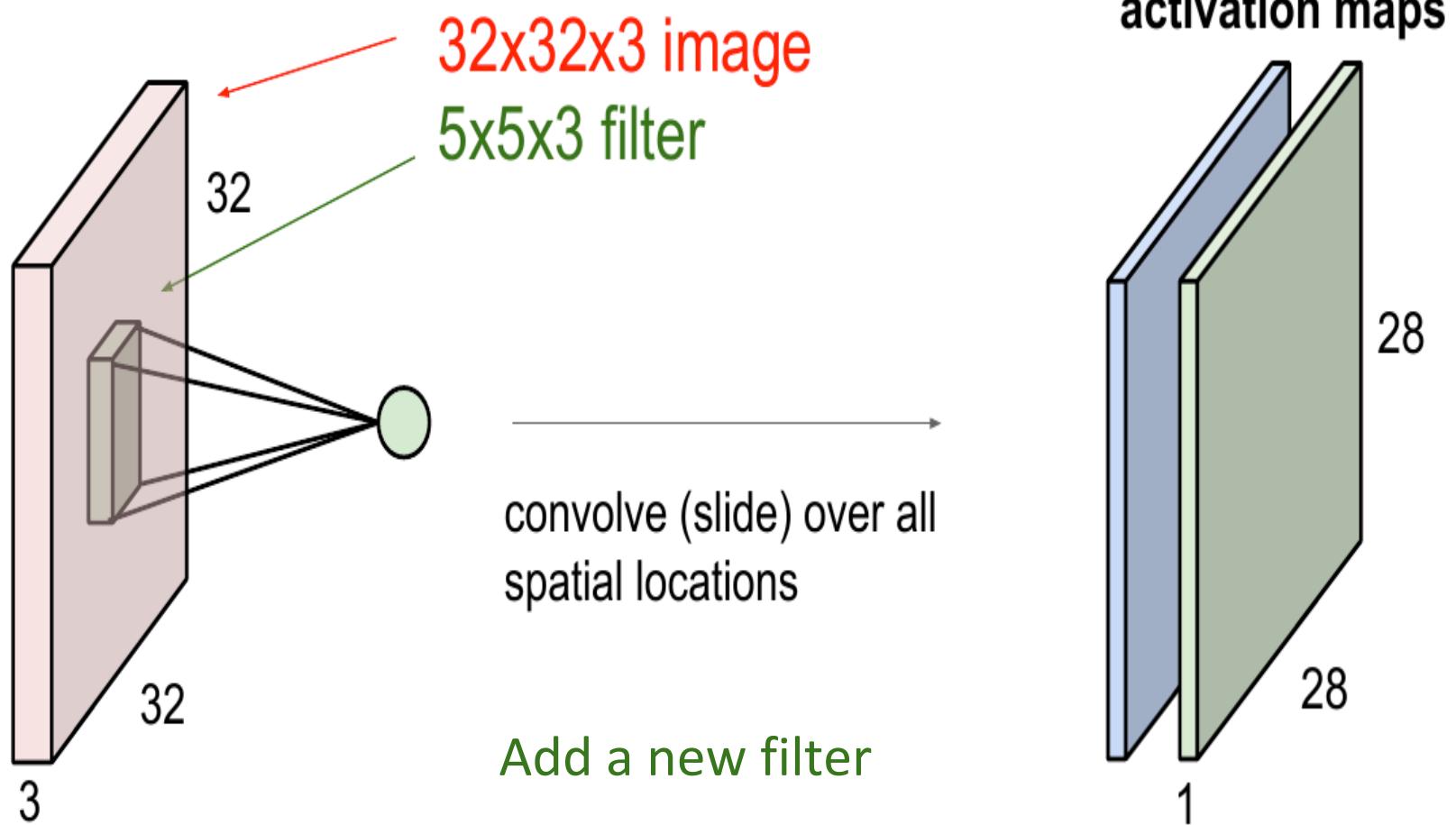
Convolutional layers



Convolutional layers

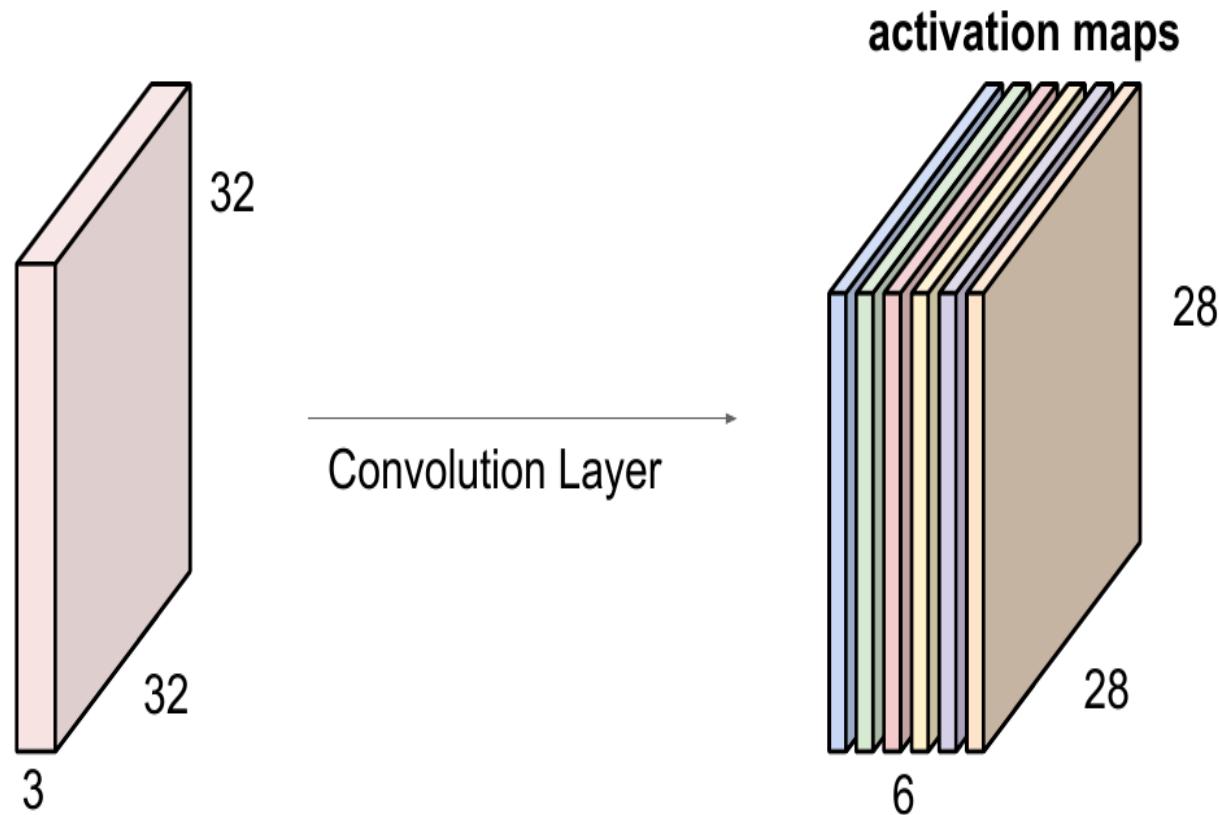


Convolutional layers



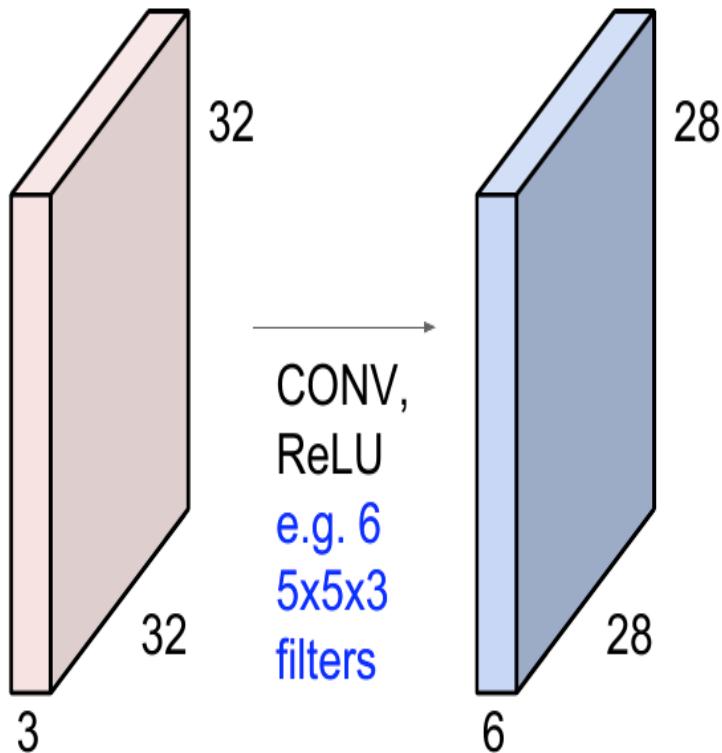
Convolutional layers

If we have 6 filters ($5 \times 5 \times 3$), there will be 6 activation maps:



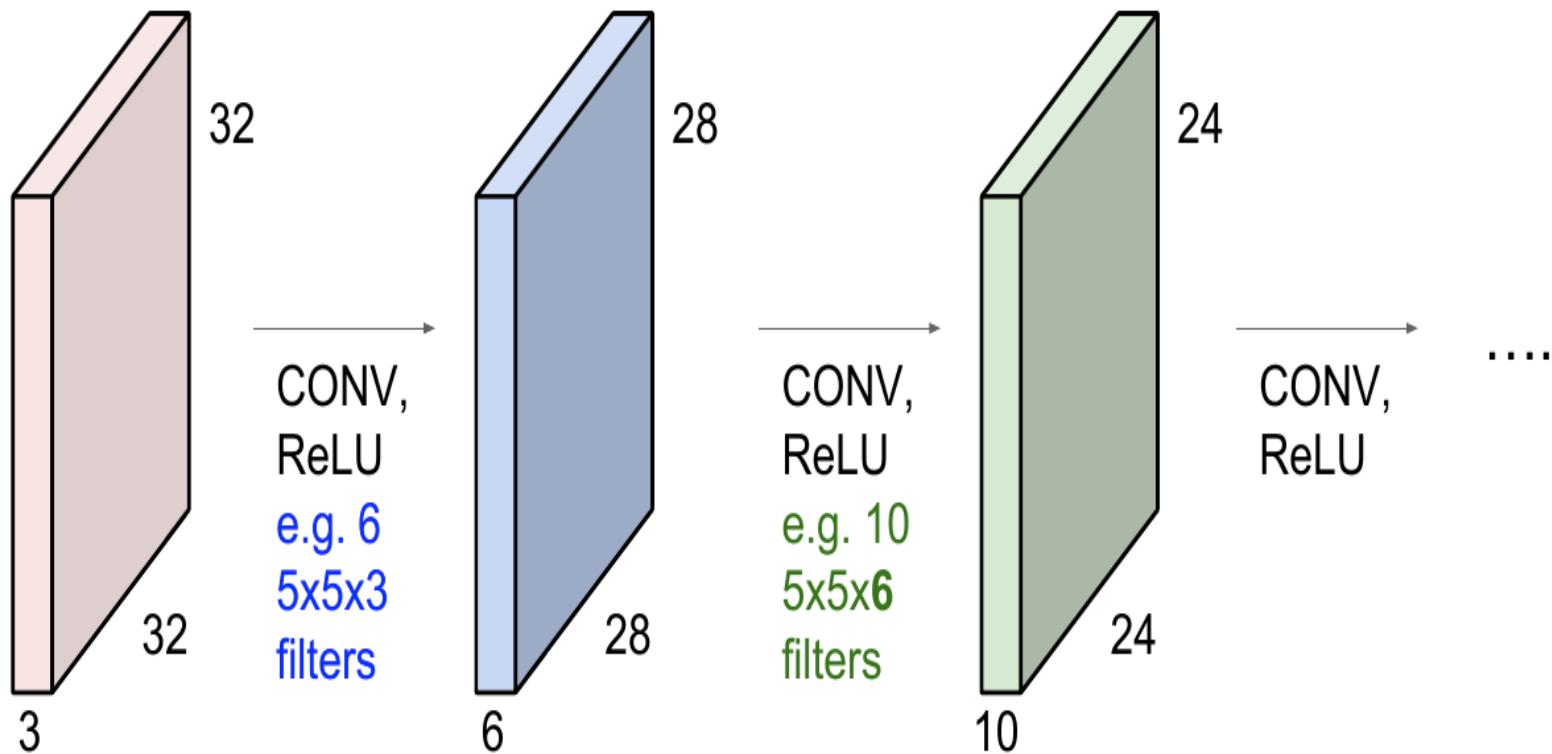
Convolutional layers

A ConvNet is a sequence of convolutional layers, each followed by an activation function:

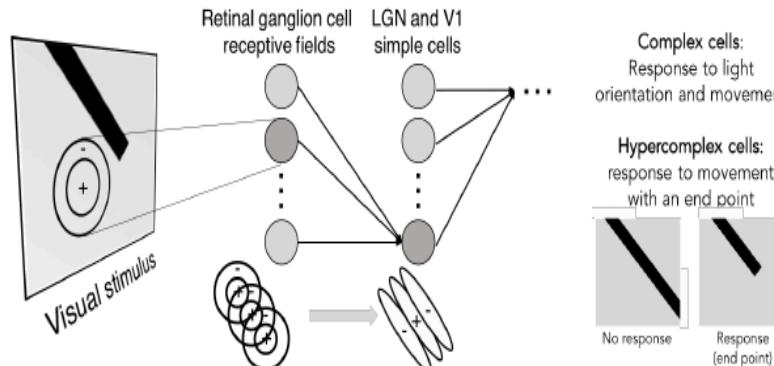
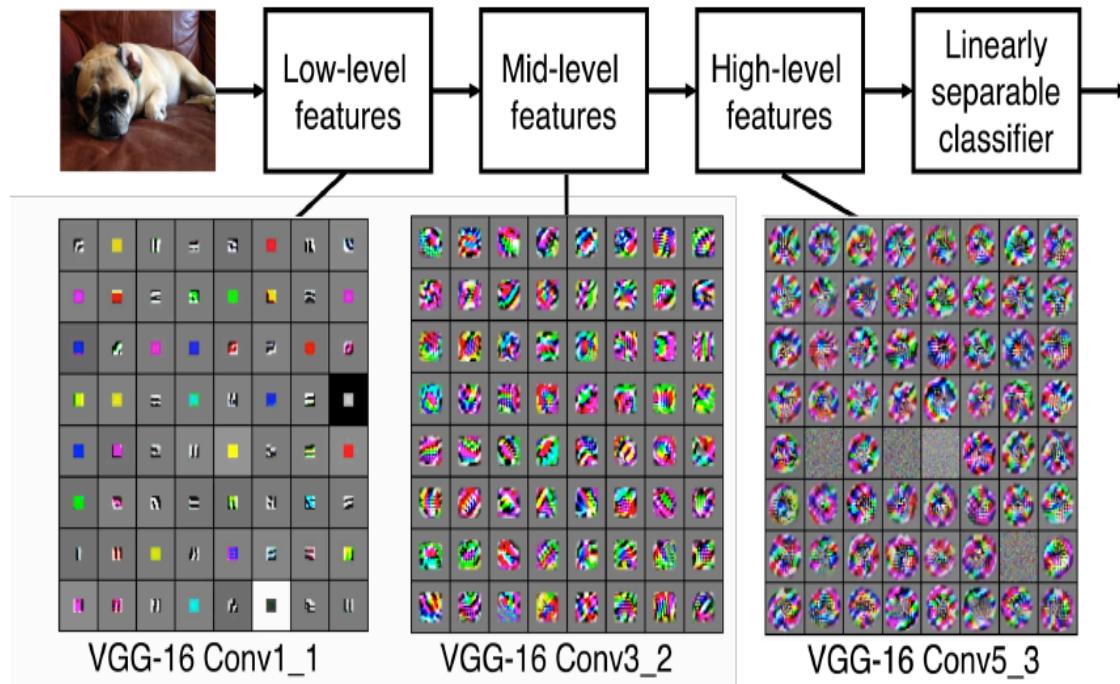


Convolutional layers

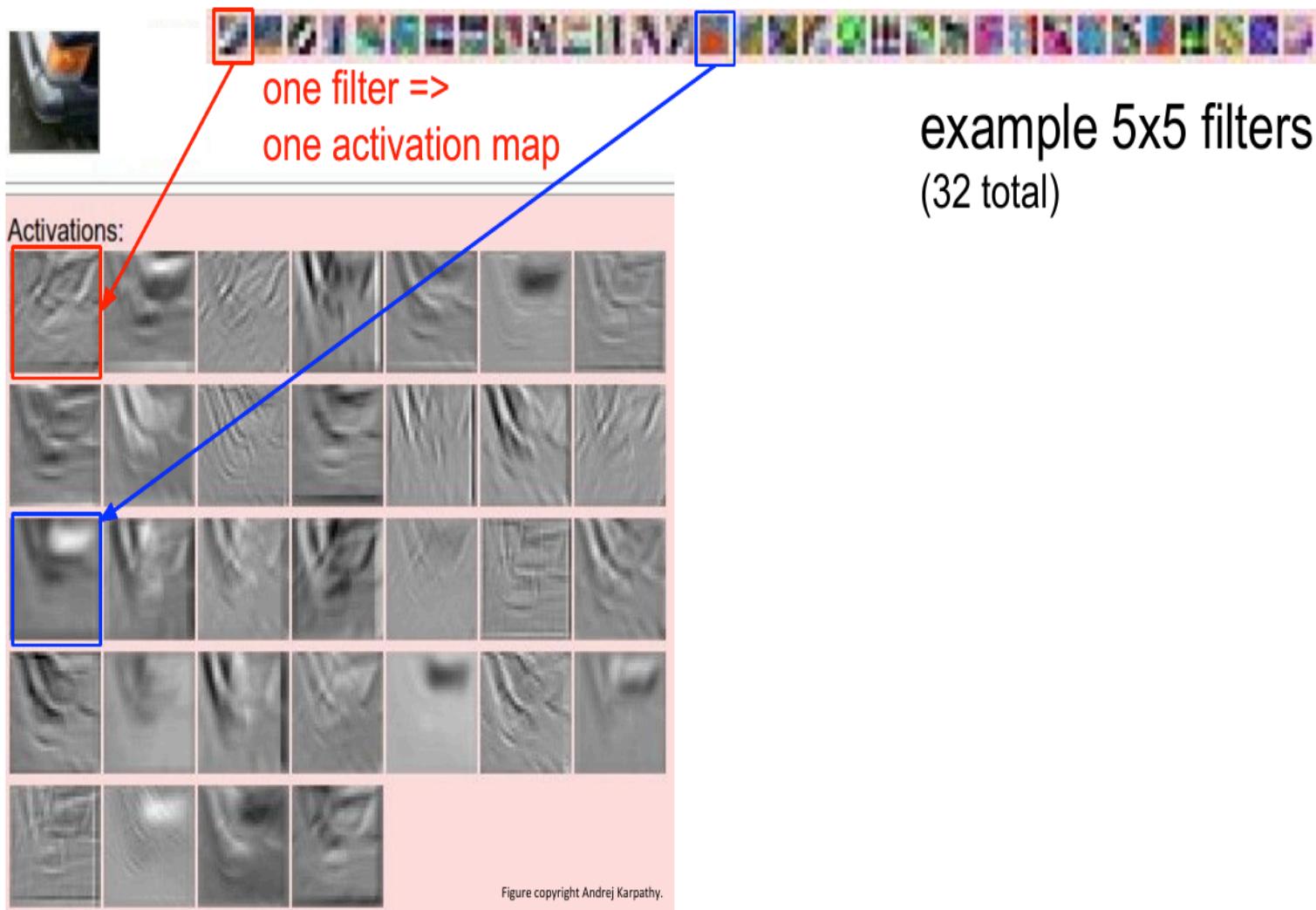
A ConvNet is a sequence of convolutional layers, each followed by an activation function:



Filter hierarchy

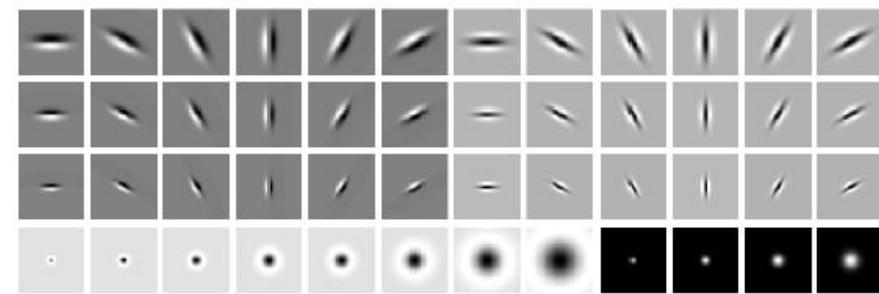


Activation maps

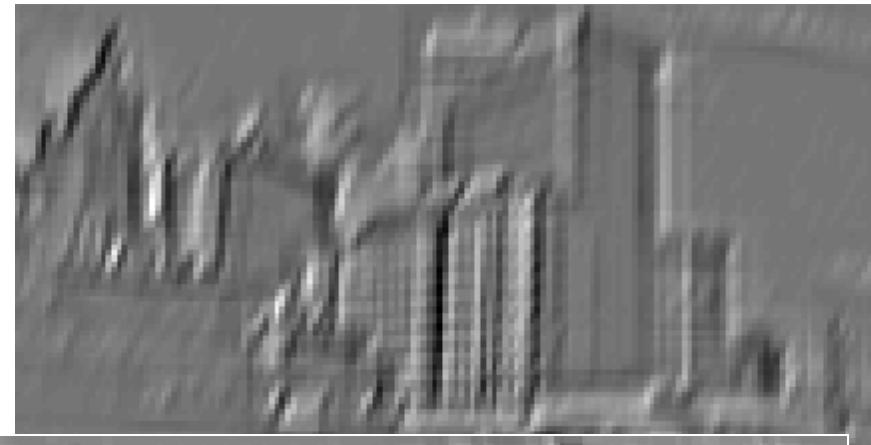


Convolution as feature extraction

bank of K filters K feature maps

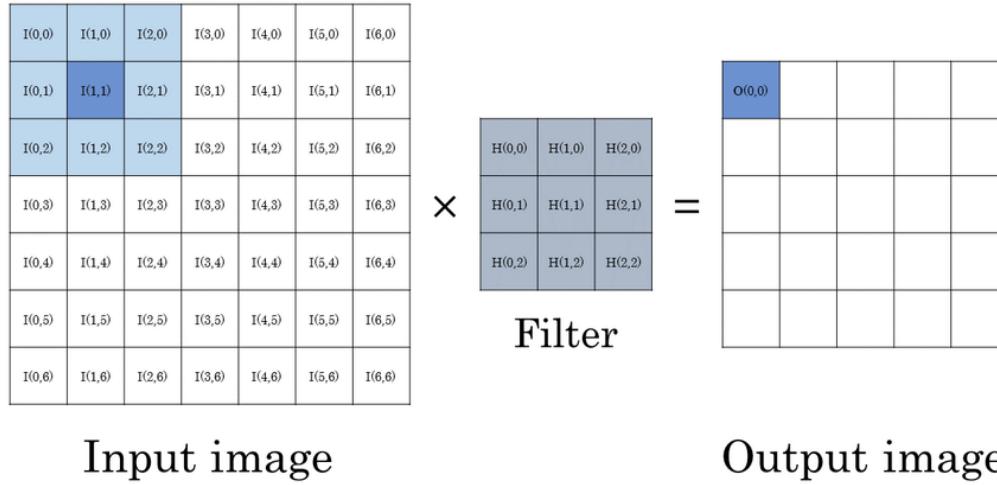


image



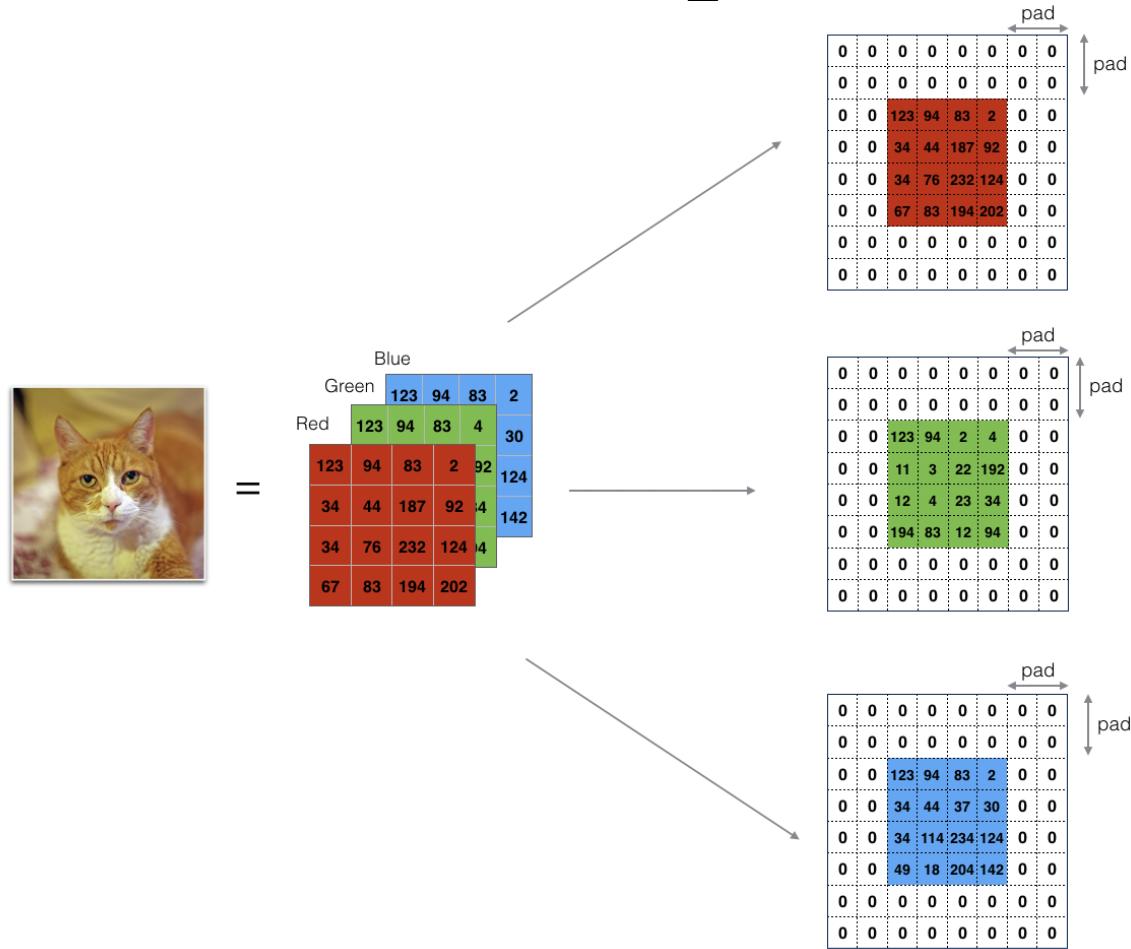
feature map

Padding



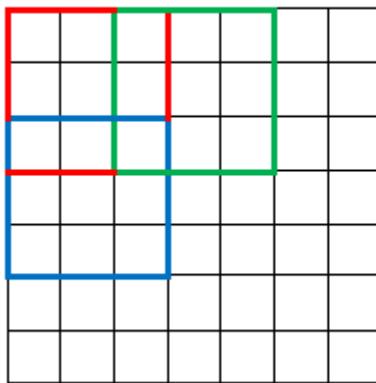
- When an image $n \times n$ is convolved by a filter $f \times f$, it loses rows and columns, resulting in a $n-f+1 \times n-f+1$ image
- In order for the image to keep the same dimensions, p columns and rows must be added prior to convolution, resulting in a $n+2p-f+1 \times n+2p-f+1$ image

Padding

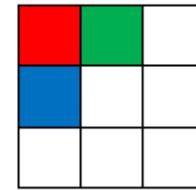


Striding

7 x 7 Input Volume



3 x 3 Output Volume



- When an image $n \times n$ is convolved by a filter $f \times f$ using stride s , it loses rows and columns, resulting in a $(n-f)/s+1 \times (n-f)/s+1$ image

Padding and striding

- When an image $n \times n$ is convolved by a filter $f \times f$ using stride s and pad p the resulting image is

$$(n+2p-f)/s+1 \times (n+2p-f)/s+1$$

- In order for the image to keep original dimensions (“same” padding):

$$p = (ns - n + f - s) / 2$$

Convolution over volumes

- A convolutional layer may receive an image with multiple channels to be convolved by multiple filters
- The number of channels of the image and the filters must be the same
- Example: Input image of **6x6x3** and **10** filters **3x3x3** resulting in a **4x4x10** output (with $p = 0$, $s = 1$)

Convolution over volumes

- Each layer receives a volume of $W_1 \times H_1 \times D_1$
- Requires 4 hyperparameters:
 - Number of filters n_c
 - Size of filter f
 - *Stride* s
 - *Padding* p
- Outputs a volume of $W_2 \times H_2 \times D_2$, where:
 - $W_2 = (W_1 - f + 2p)/s + 1$
 - $H_2 = (H_1 - f + 2p)/s + 1$
 - $D_2 = n_c$
- Must compute $f \times f \times D_1$ weights per filter, a total of $f \times f \times D_1 \times n_c$ weights and n_c biases
- In the output volume, the d -th slice of size $W_2 \times H_2$ is the result of the convolution between the d -th filter and the image with *stride* s , and applying an offset related to the d -th *bias*

Summary of notation

$f[l]$ = filter size

$p[l]$ = padding

$s[l]$ = stride

$nc[l]$ = number of filters

Input: $n[l-1] \times n[l-1] \times nc[l-1]$ **Or** $nH[l-1] \times nW[l-1] \times nc[l-1]$

Output: $n[l] \times n[l] \times nc[l]$ **Or** $nH[l] \times nW[l] \times nc[l]$

Where $n[l] = (n[l-1] + 2p[l] - f[l]) / s[l] + 1$

Each filter is: $f[l] \times f[l] \times nc[l-1]$

Activations: $a[l]$ is $nH[l] \times nW[l] \times nc[l]$

$A[l]$ is $m \times nH[l] \times nW[l] \times nc[l]$

Weights: $f[l] * f[l] * nc[l-1] * nc[l]$

bias: $(1, 1, 1, nc[l])$

Forward propagation

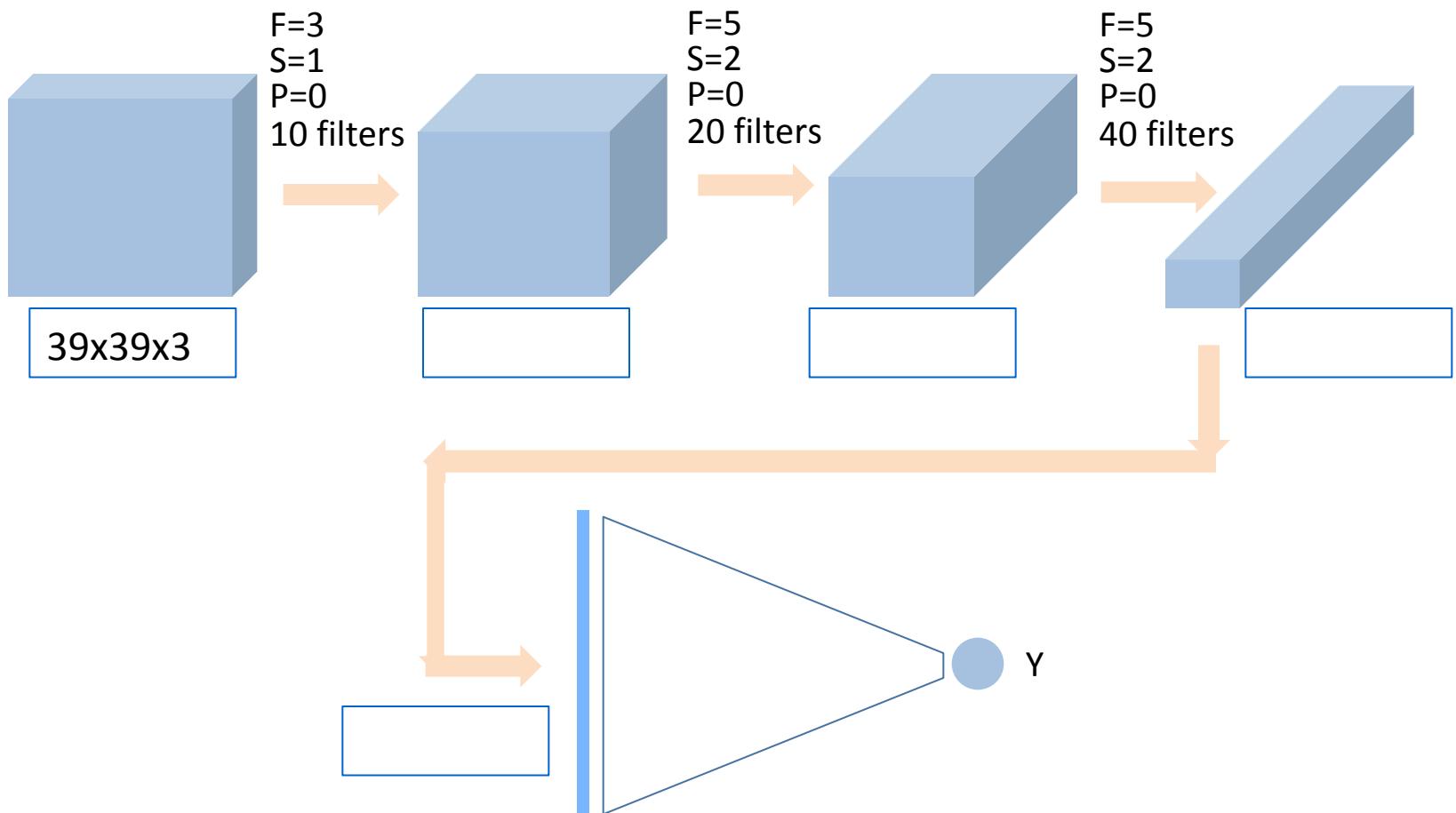
- Convolve the filters with the image
- Add a bias to the output of each filter
- Apply ReLU to the result

Forward propagation

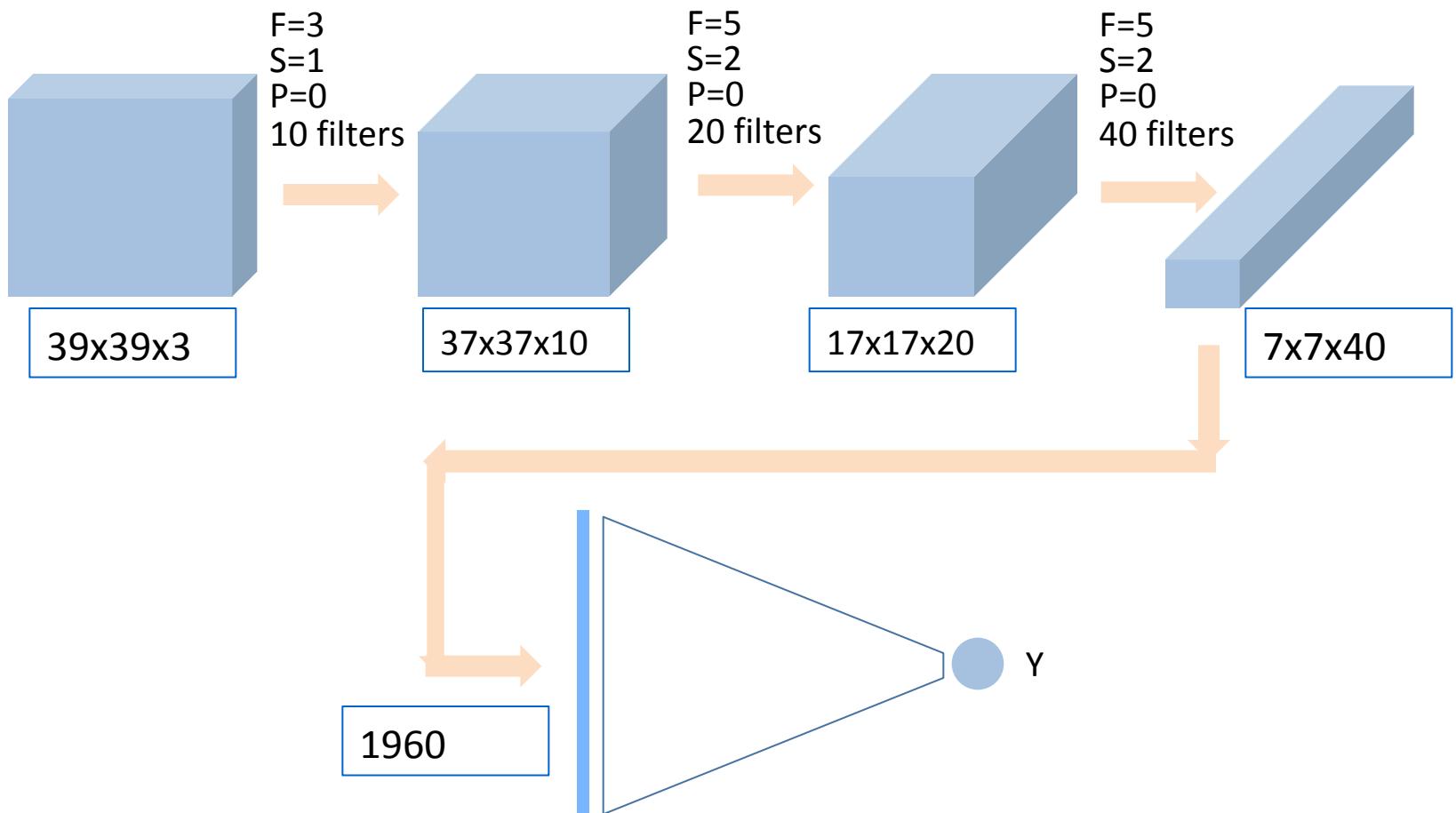
Example:

- Convolve the filters with the image:
 - Input image: **6x6x3** (a_0)
 - 10 filters: **3x3x3** (W_1)
 - Result: **4x4x10** (W_1a_0)
- Add a bias to the output of each filter
 - **b, 10x1**: image **4x4x10** ($W_1a_0 + b$)
- Apply ReLU to the result
 - imagem **4x4x10** ($A_1 = \text{RELU}(W_1a_0 + b)$)
- Number of parameters: **(3x3x3x10) + 10 = 280**

Example of ConvNet



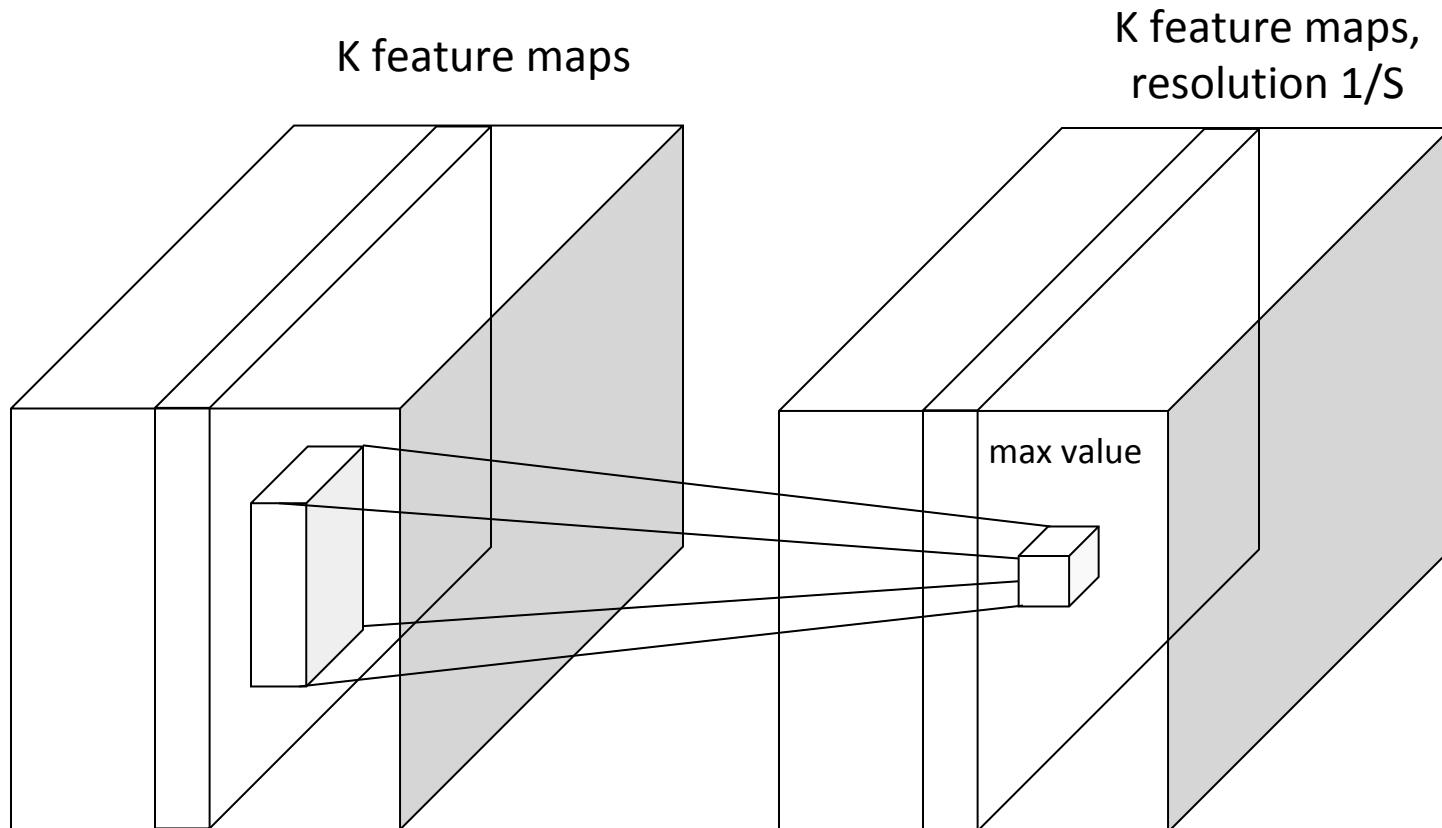
Example of ConvNet



Pooling layers

- In addition to the convolutional layers, CNNs usually use pooling layers to
 - Reduce the size of entries
 - Speed up computation
 - Make some detected features more robust
- Pooling layers have no parameters to learn!

Pooling layers



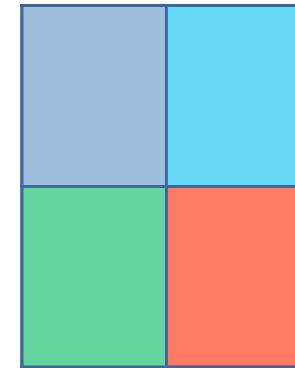
$F \times F$ pooling filter, stride S

Usually: $F=2$ or 3, $S=2$

Backward pass: gradient from next layer is passed back only to the unit with max value

Max pooling layer

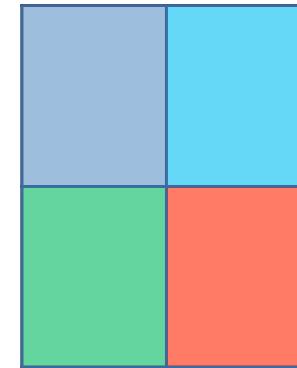
1	3	2	1
2	9	1	1
1	3	2	3
5	6	1	2



Max pooling says that if a feature is detected anywhere in the filter, keep it high to the next layer

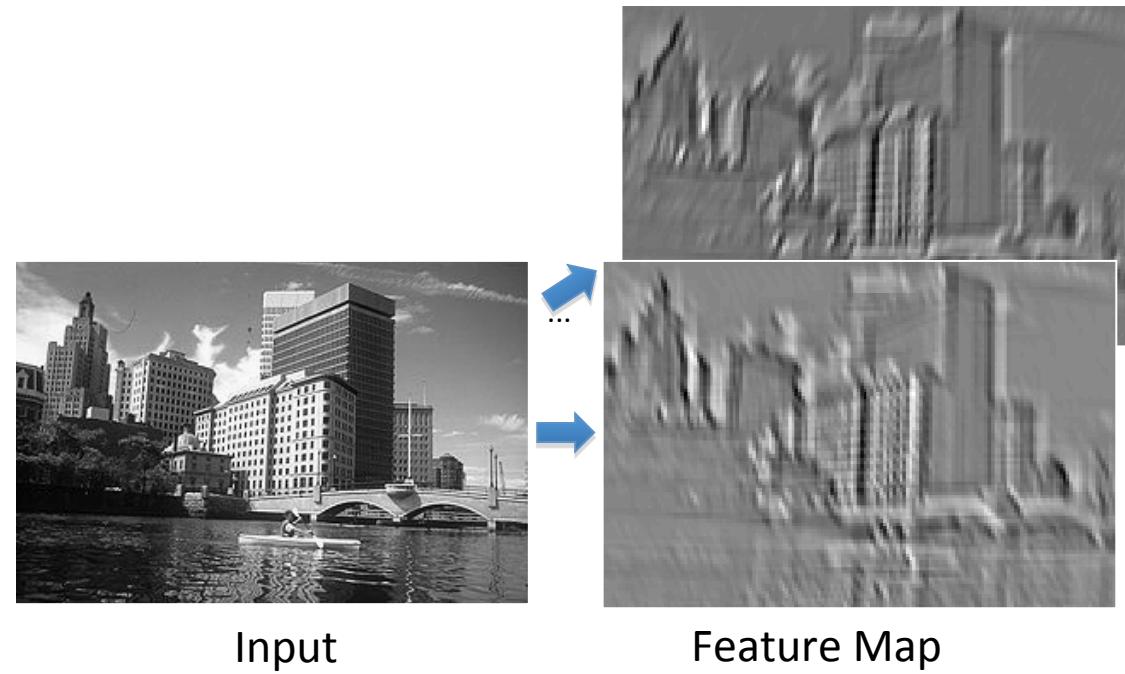
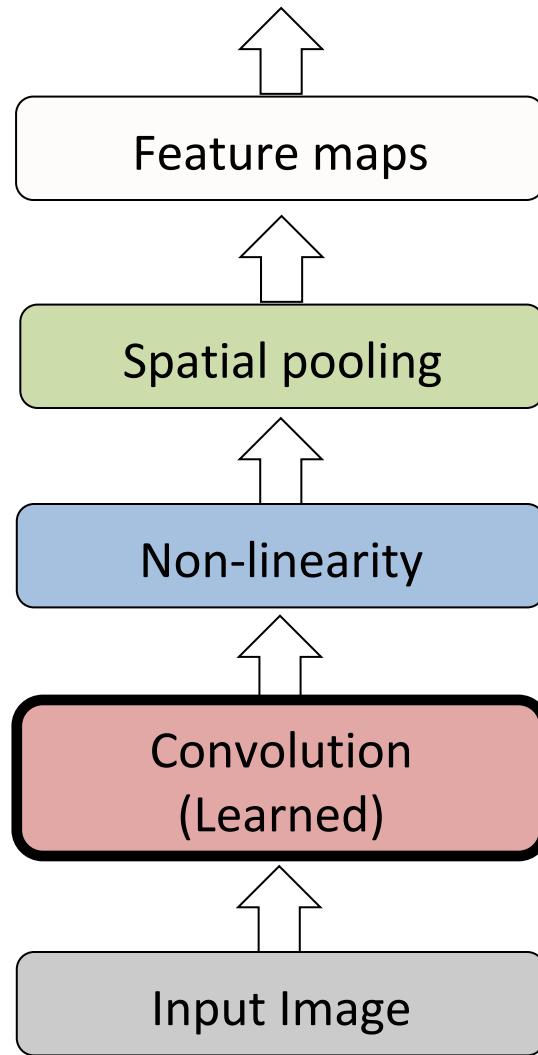
Average pooling layer

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



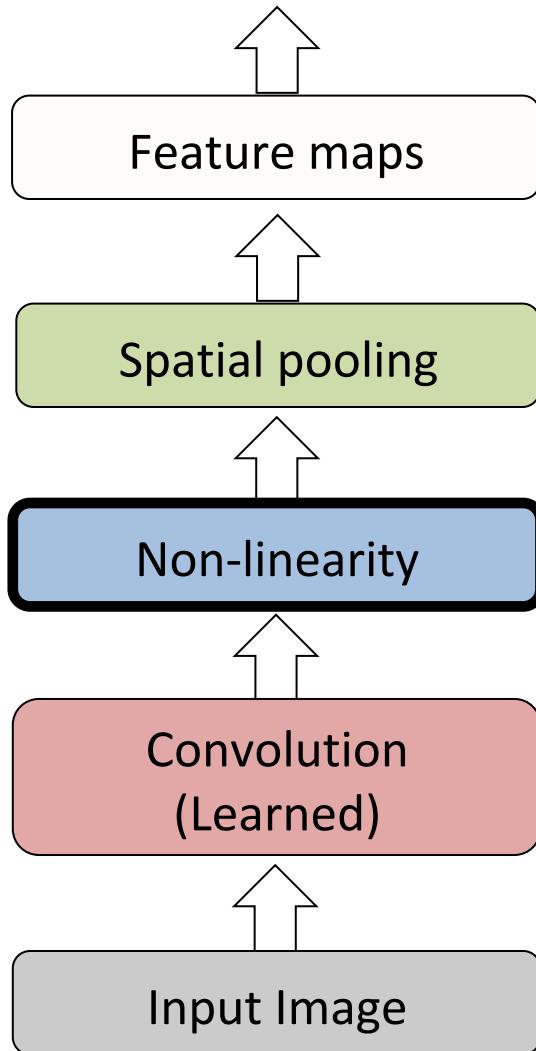
Average pooling takes the average of values and is less used than max pooling

Summary: CNN pipeline

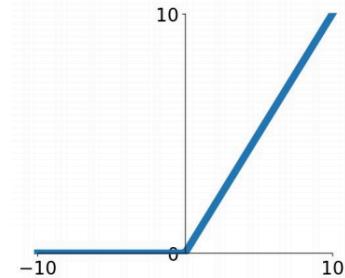


Source: R. Fergus, Y. LeCun

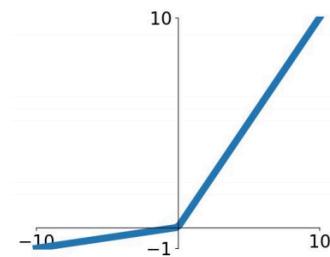
Summary: CNN pipeline



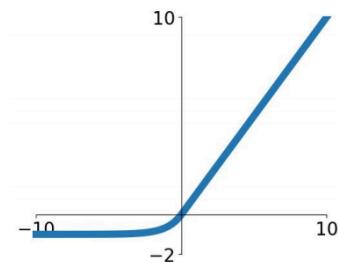
ReLU
 $\max(0, x)$



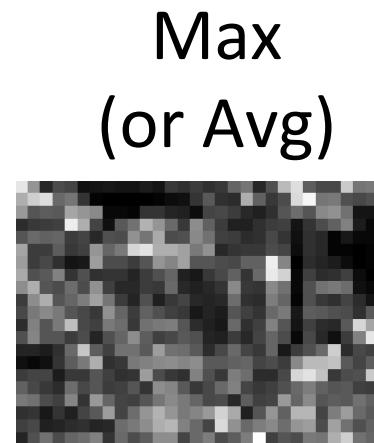
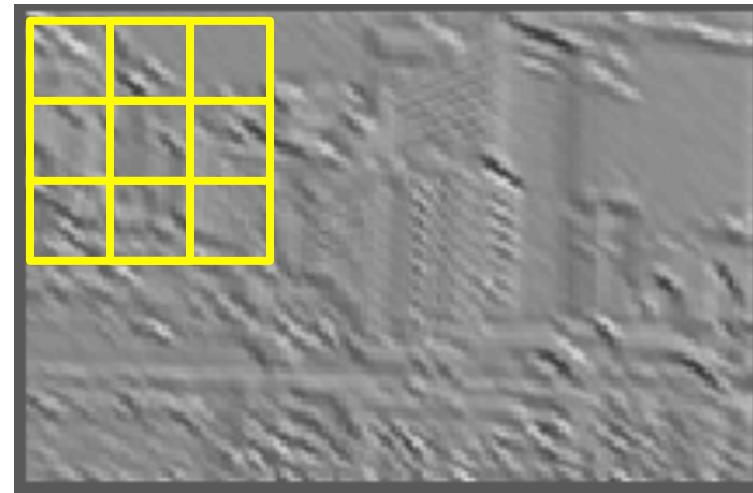
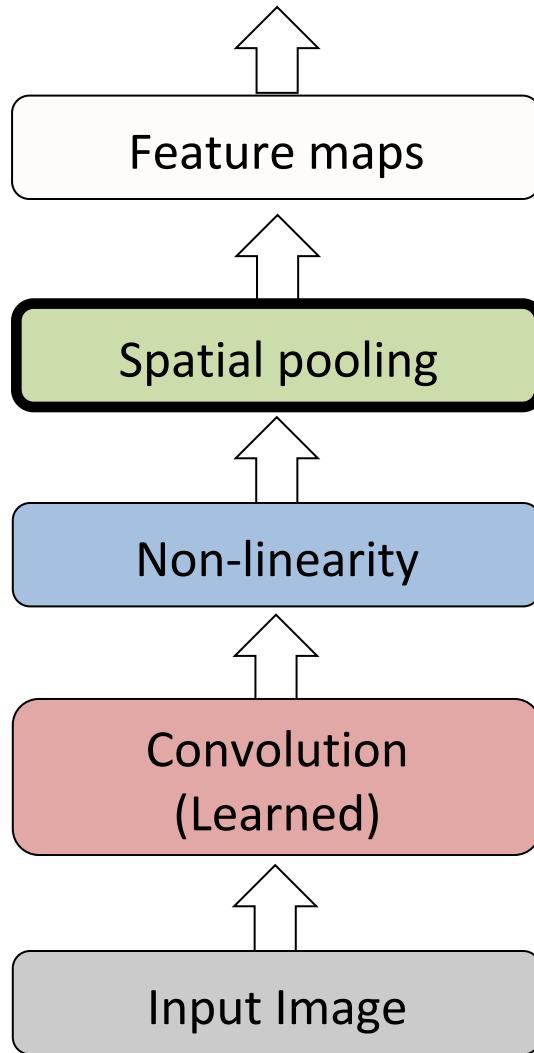
Leaky ReLU
 $\max(0.1x, x)$



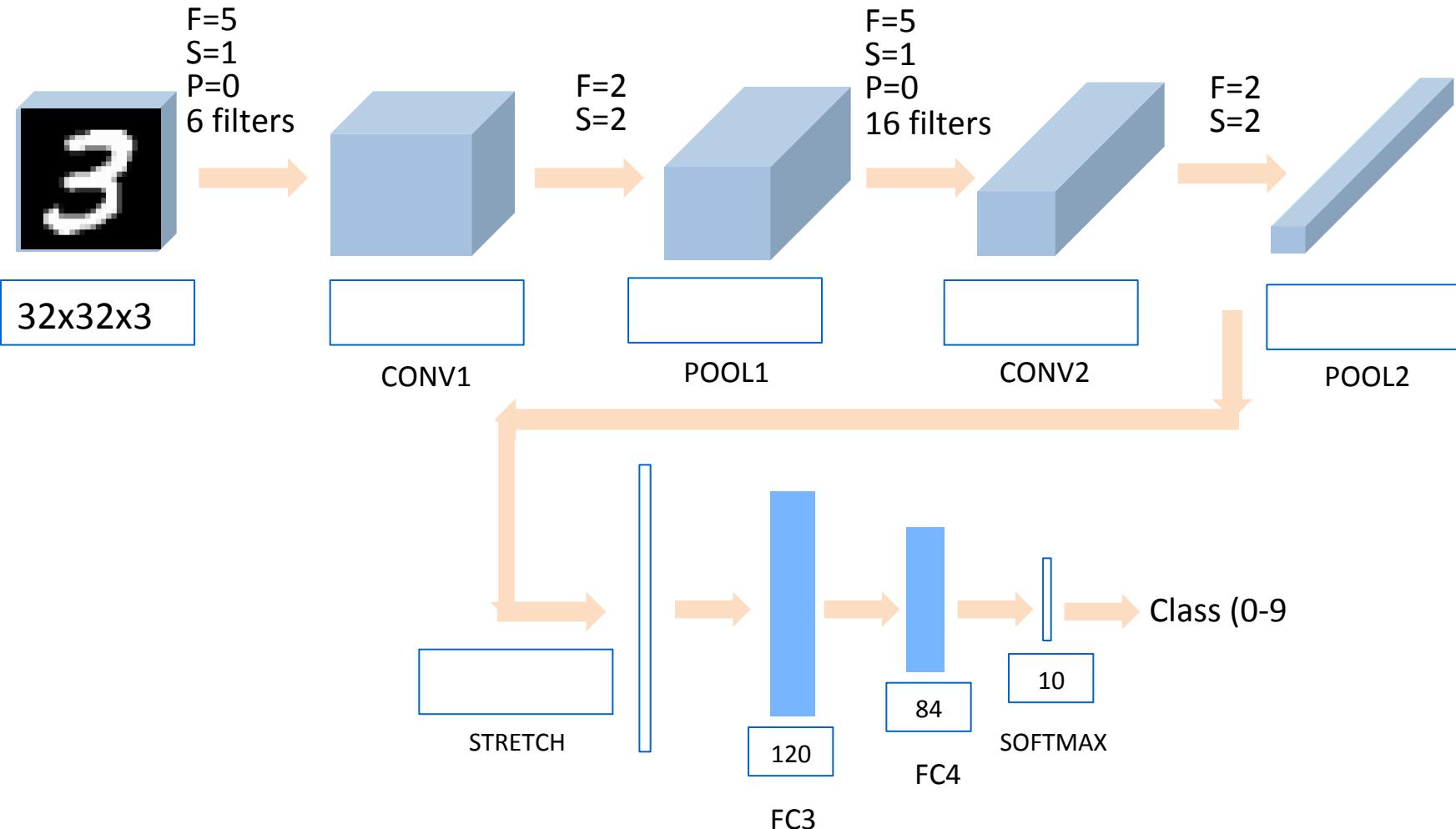
ELU
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



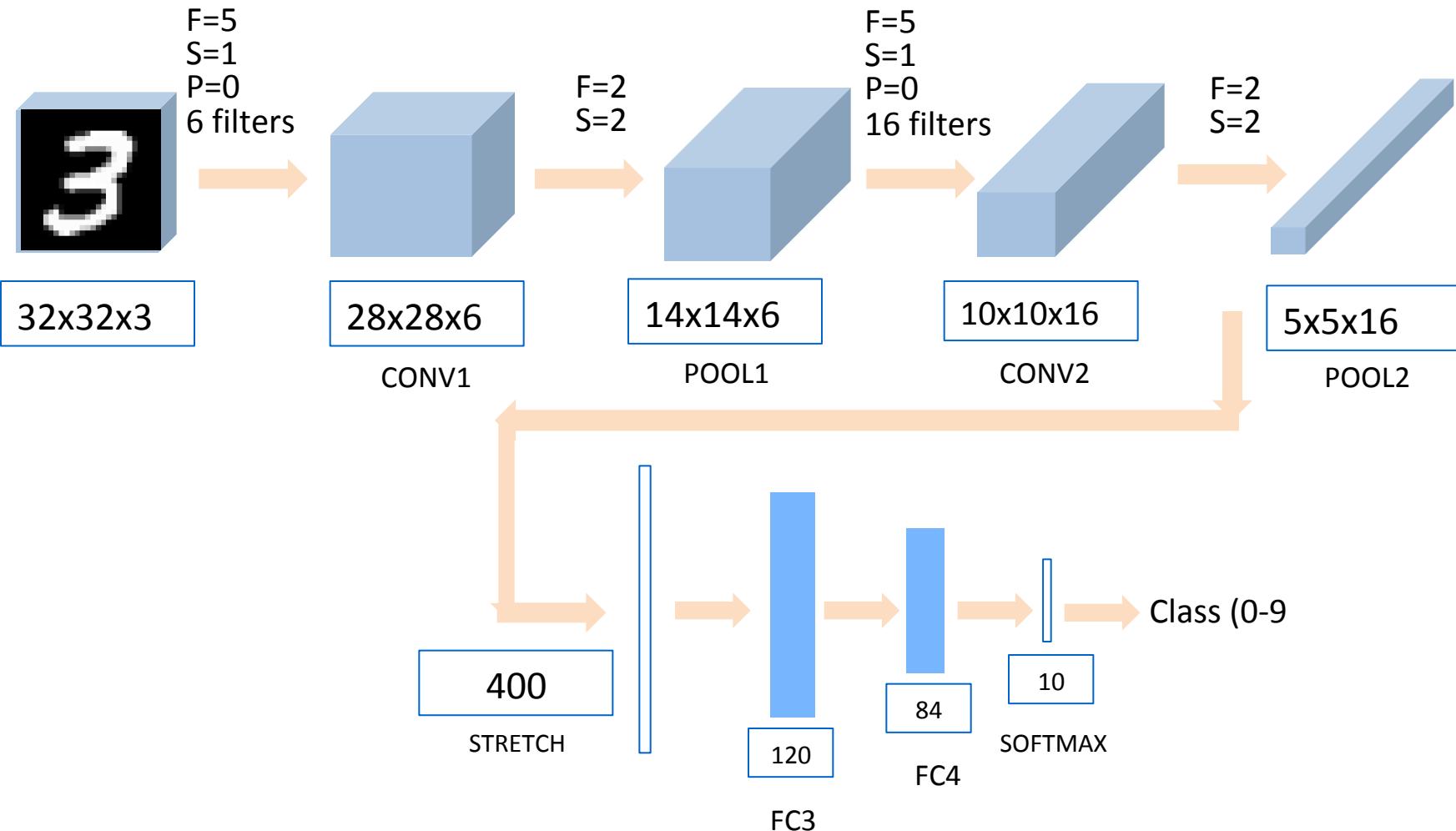
Summary: CNN pipeline



Example of CNN (LeNet-5)



Example of CNN (LeNet-5)



References and acknowledgements

Some of these slides were inspired or adapted from courses and presentations given by Andrew Ng, Camila Laranjeira, Fei-Fei Li, Flávio Figueiredo, Hugo Oliveira, Jefersson dos Santos, Justin Johnson, Keiller Nogueira, Pedro Olmo, Renato Assunção, Serena Yeung.

Reference courses include *Machine Learning* and *Deep Learning* CS230 and CS231 from Stanford University, *Deep Learning* and *Hands-on Deep Learning* from UFMG, *Deep Learning* CS498 from Un. Of Illinois.