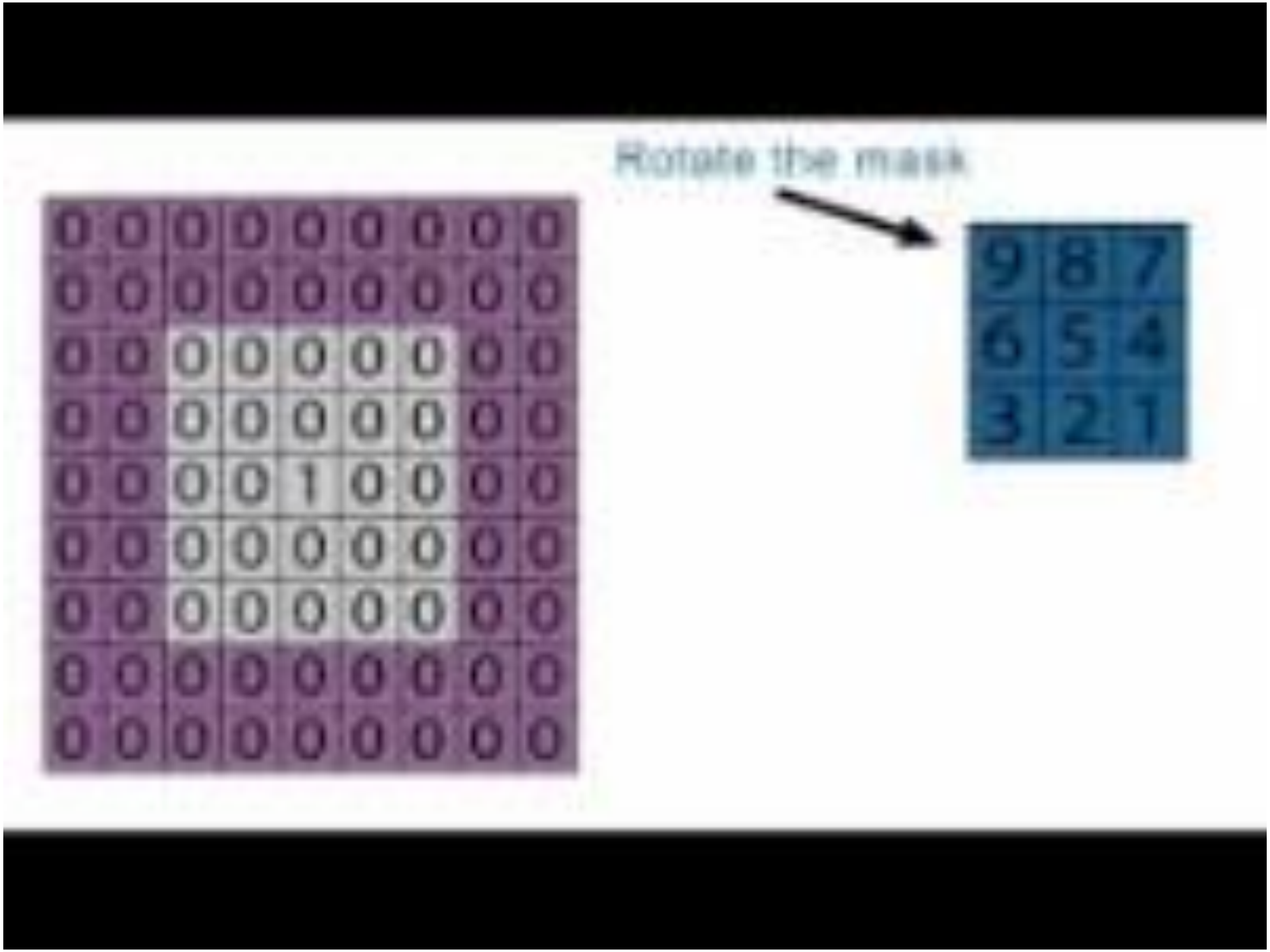
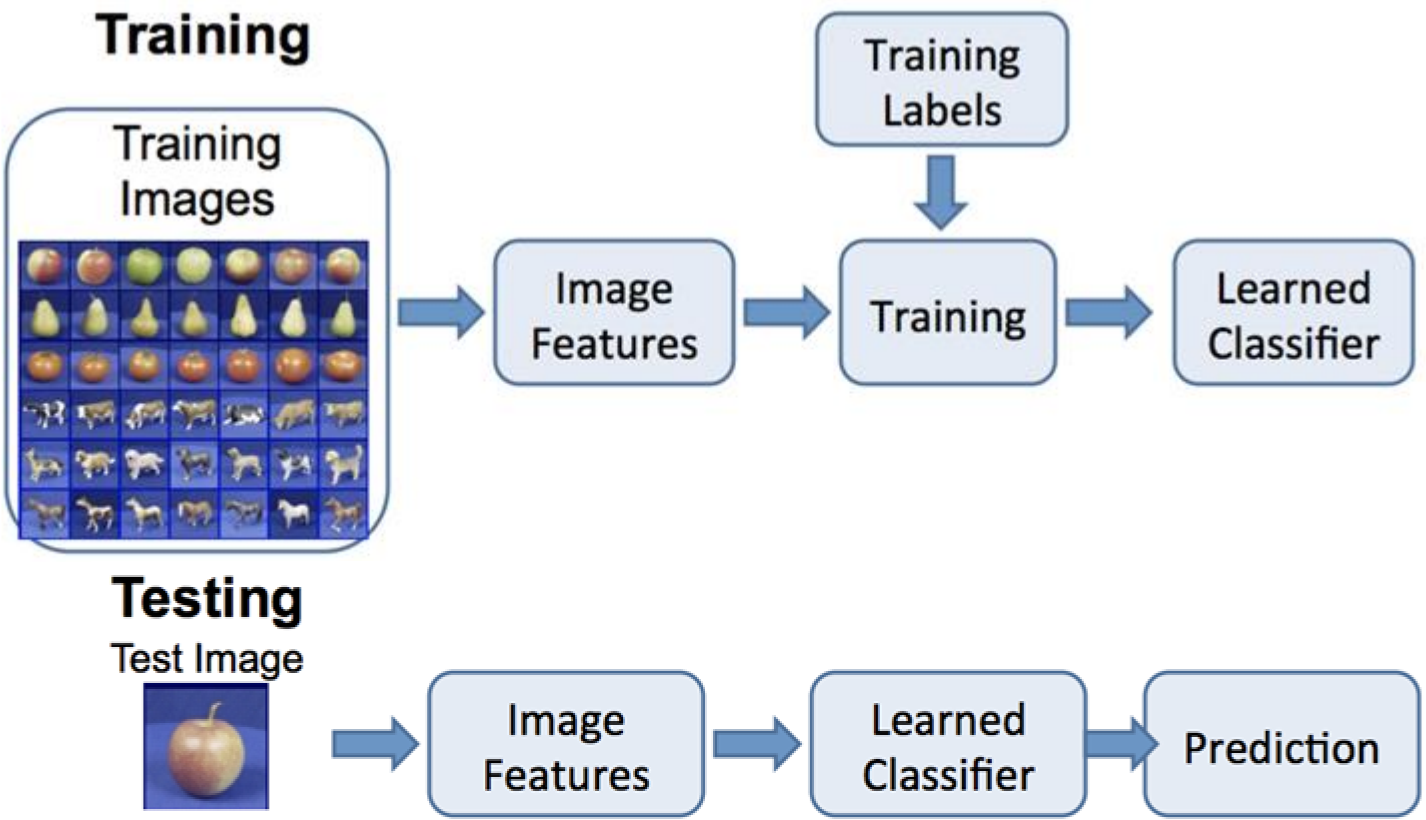
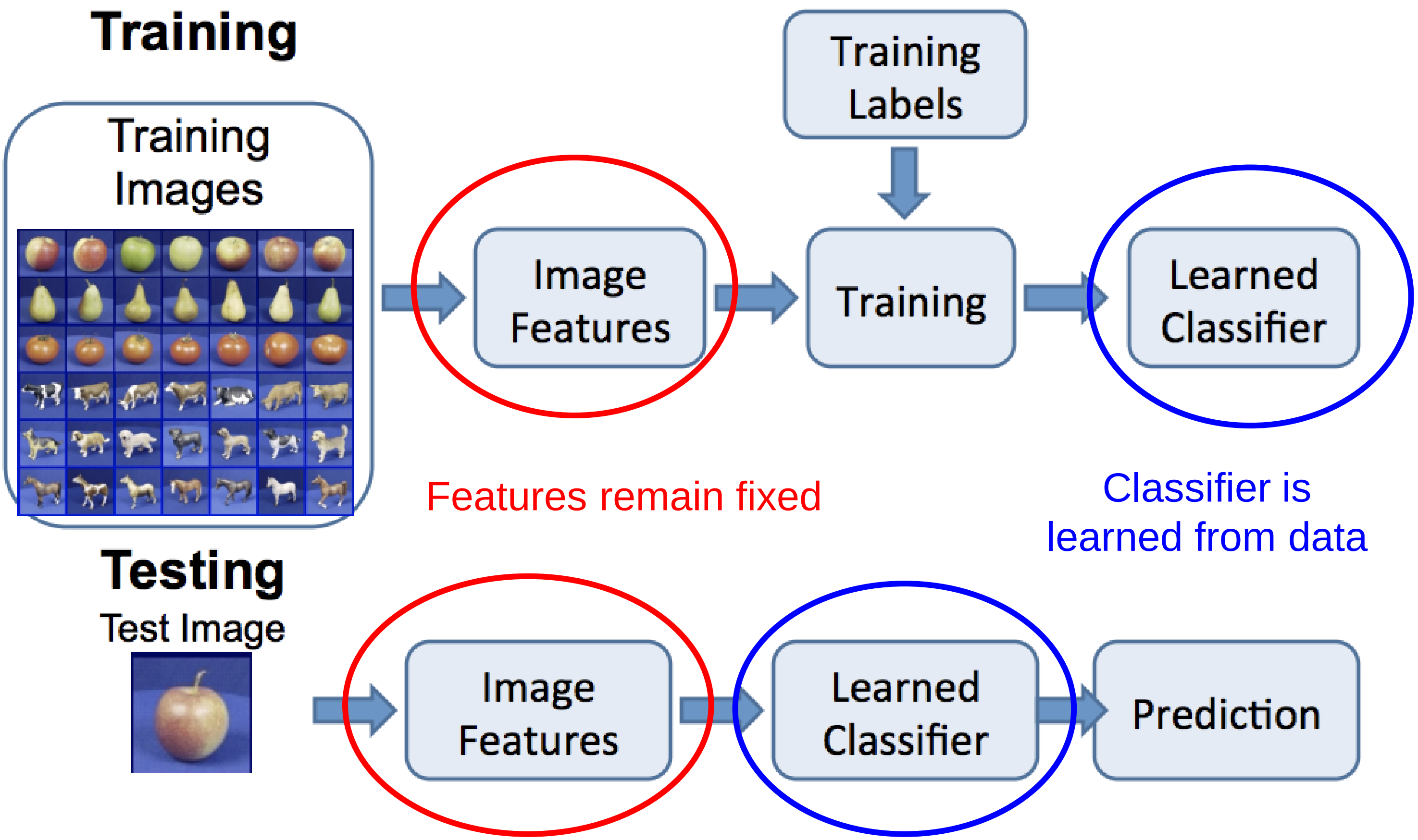
Convolution



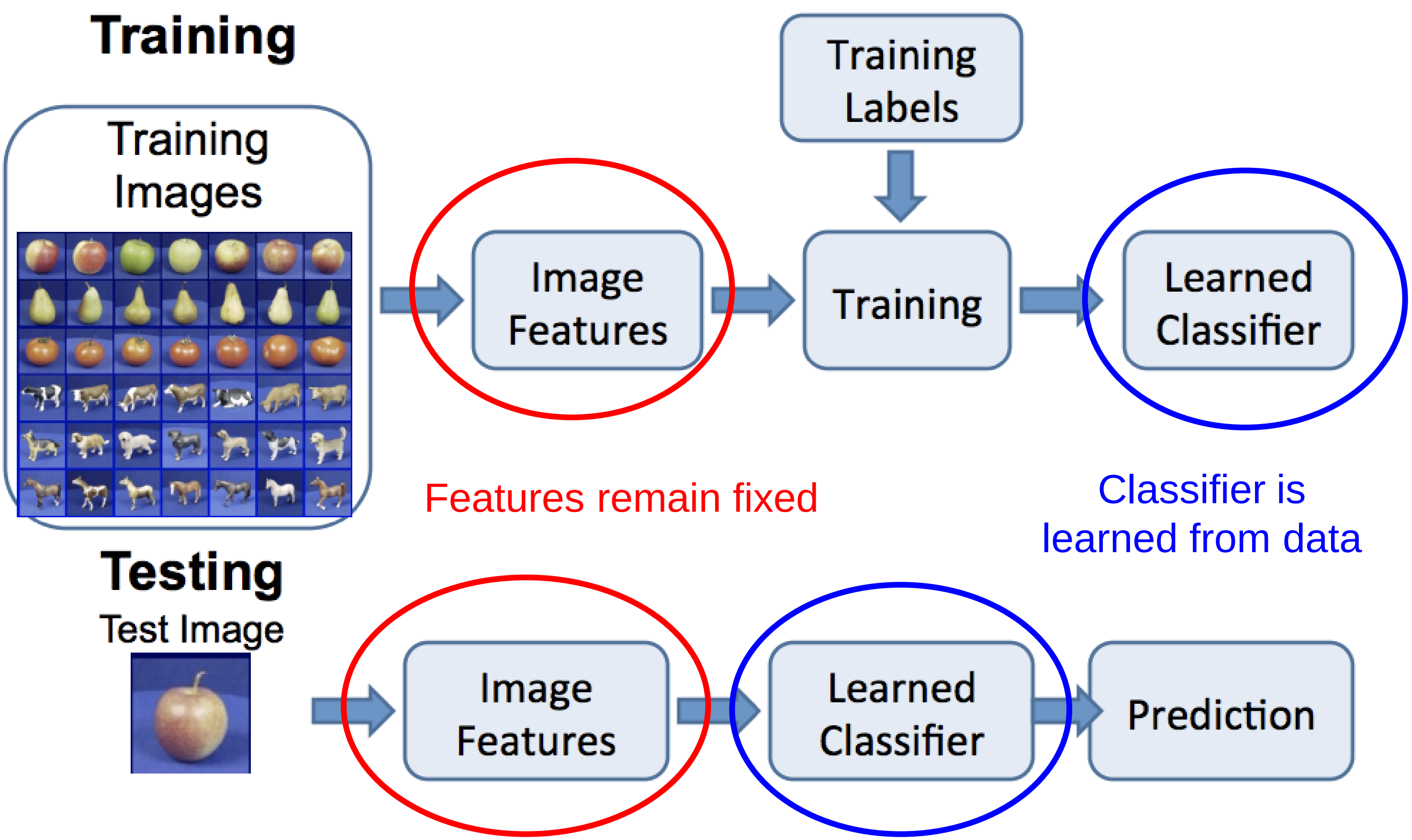
Recall: Image Classification



Recall: Image Classification

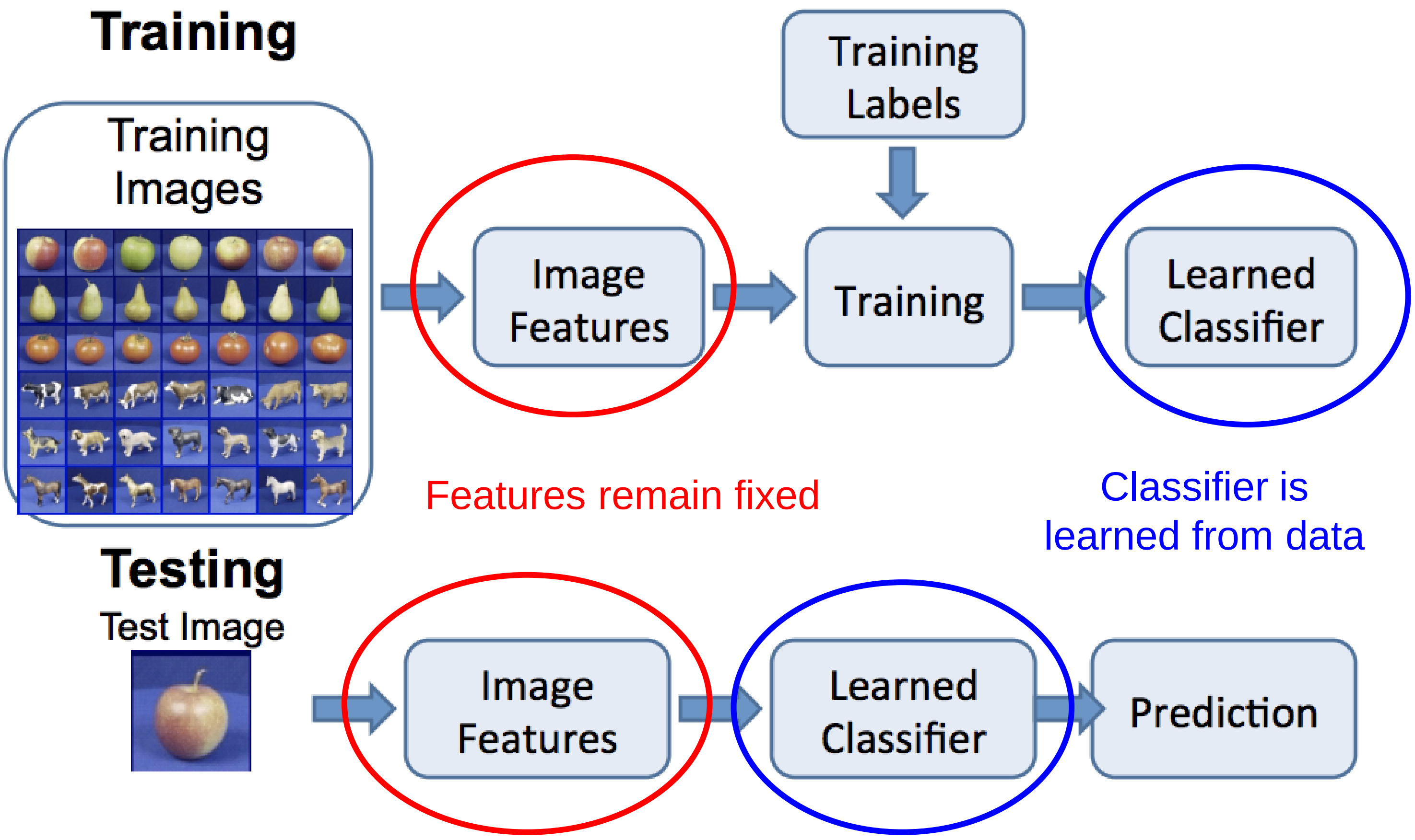


Recall: Image Classification



Problem:
How do we know which features to use? We may need different features for each problem!

Recall: Image Classification



Problem:
How do we know which features to use? We may need different features for each problem!

Solution:
Learn the features jointly with the classifier!

Image Classification: Feature Learning

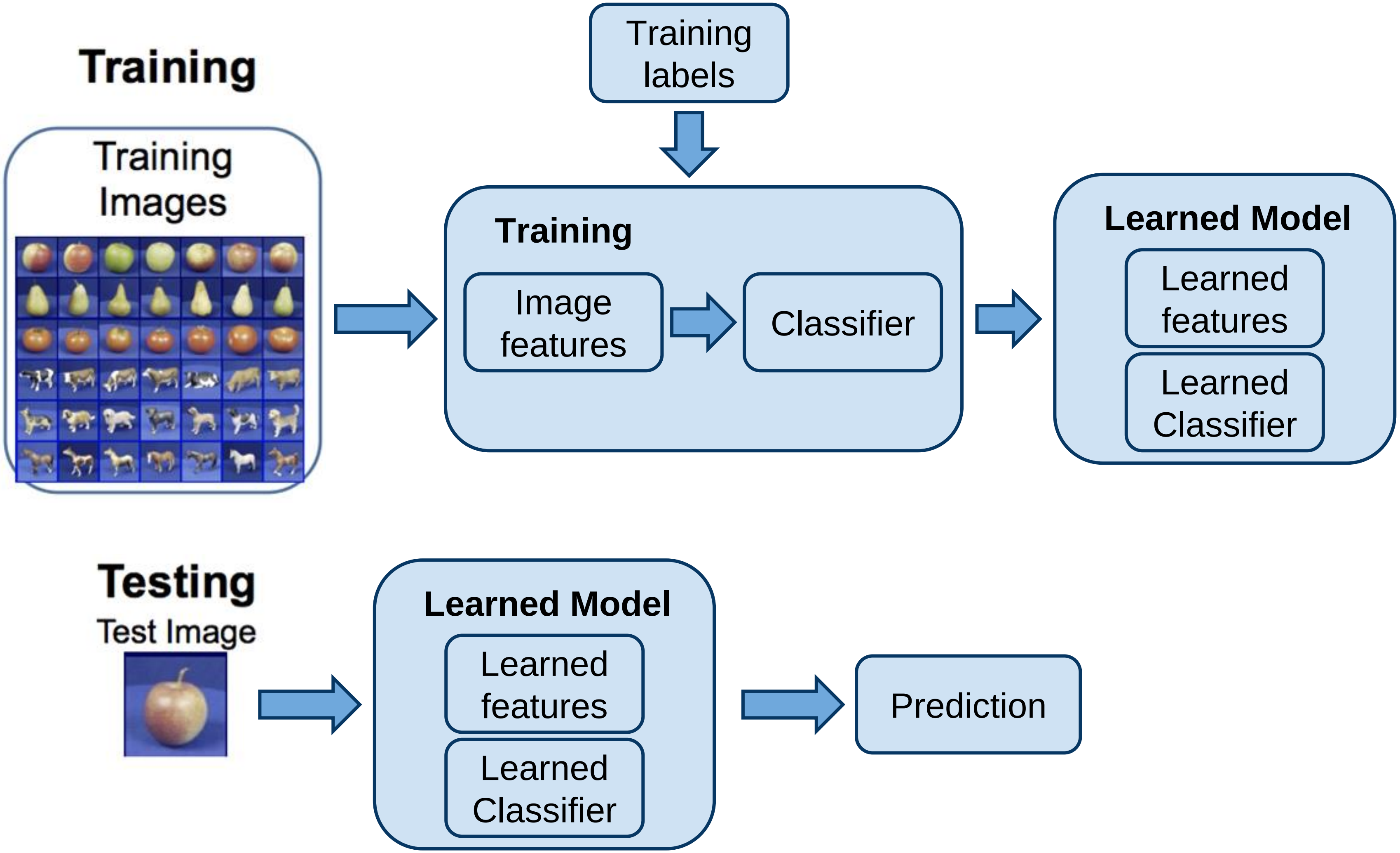
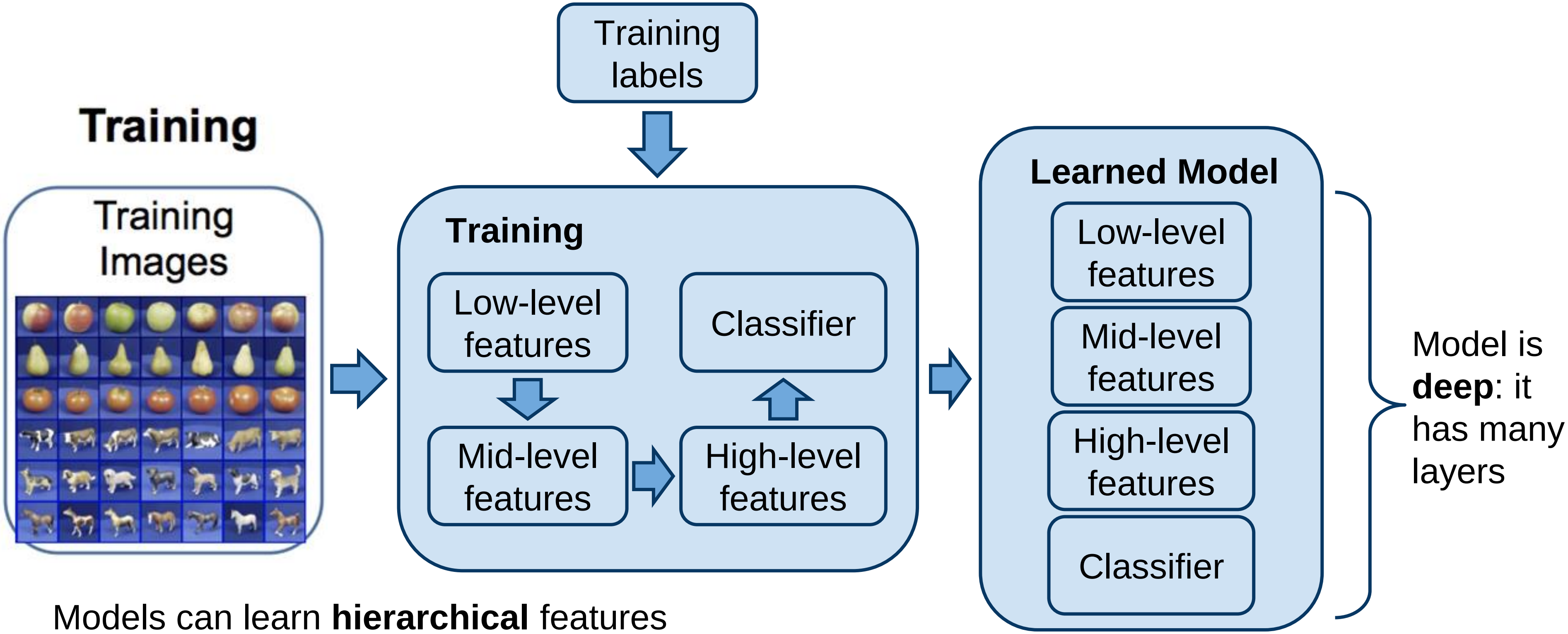
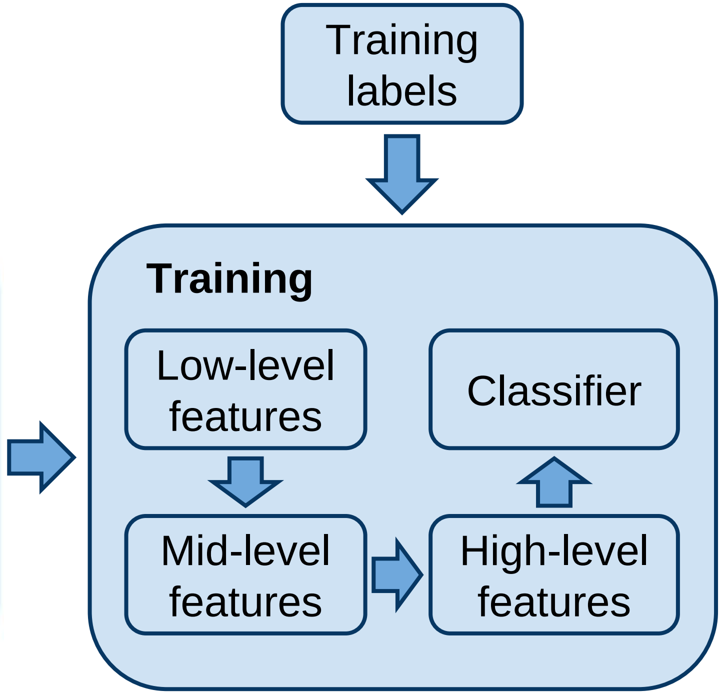


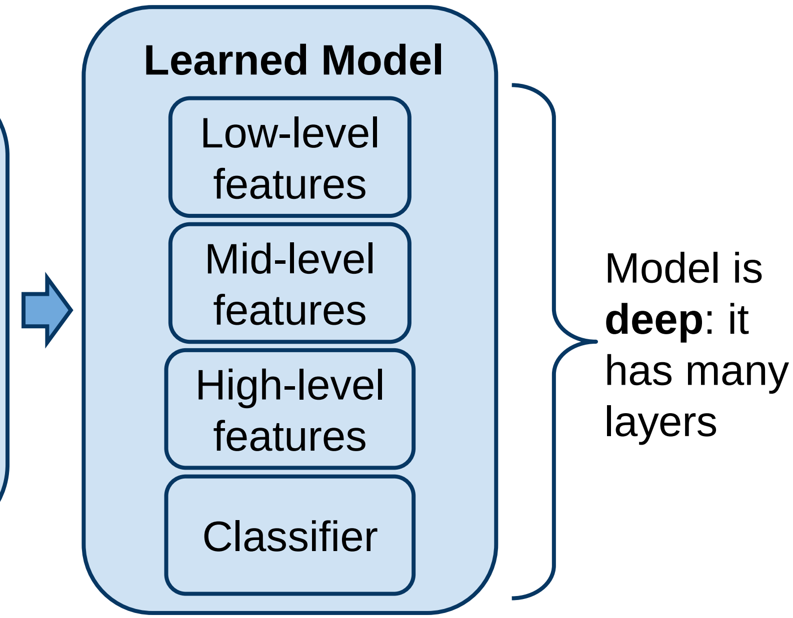
Image Classification: Deep Learning



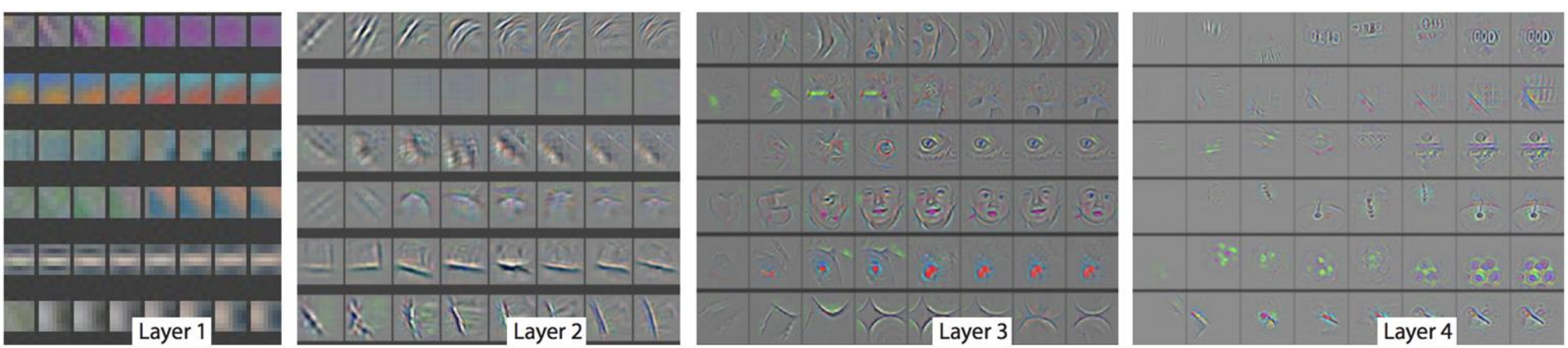
Models can learn **hierarchical** features

Image Classification: Deep Learning





Models can learn **hierarchical** features



Linear classifier

3072-dimensional dot-product

$$\underbrace{\boxed{f(x,W)}}_{10\times1} = \underbrace{\boxed{W}}_{10\times3072}\overbrace{\boxed{x}}^{3072\times1} + \boxed{b}\ \ 10\times1$$

Image

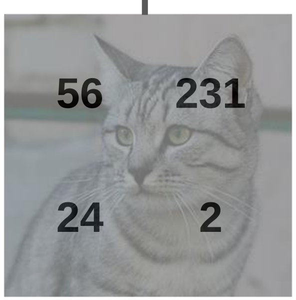
Array of **32x32x3** numbers
(3072 numbers total)

$$\longrightarrow f(\mathbf{x},\mathbf{W}) \longrightarrow$$

10 numbers giving
class scores

Spatial connection and
information is lost!
Spatially Variant!!

Stretch pixels into column



Input image

$$\underbrace{\begin{bmatrix} 0.2 & -0.5 & 0.1 & 2.0 \\ 1.5 & 1.3 & 2.1 & 0.0 \\ 0 & 0.25 & 0.2 & -0.3 \end{bmatrix}}_{W} \underbrace{\begin{bmatrix} 56 \\ 231 \\ 24 \\ 2 \end{bmatrix}}_{x} + \underbrace{\begin{bmatrix} 1.1 \\ 3.2 \\ -1.2 \end{bmatrix}}_{b} = \begin{bmatrix} -96.8 \\ 437.9 \\ 61.95 \end{bmatrix} \begin{array}{l} \text{Cat score} \\ \text{Dog score} \\ \text{Ship score} \end{array}$$

How to get spatial invariance?

- 2D Convolution is the process which is spatially invariant because for every pixel we also take into account its neighborhood pixels
- So the *convolutional layer* would preserve the topology of the input.
- We can learn the weights of convolution filter just as we could learn weights for neural network.

Input image

*

0	-½	0
0	0	0
0	½	0

2D convolution filter

=

output

Impact of Spatial Information

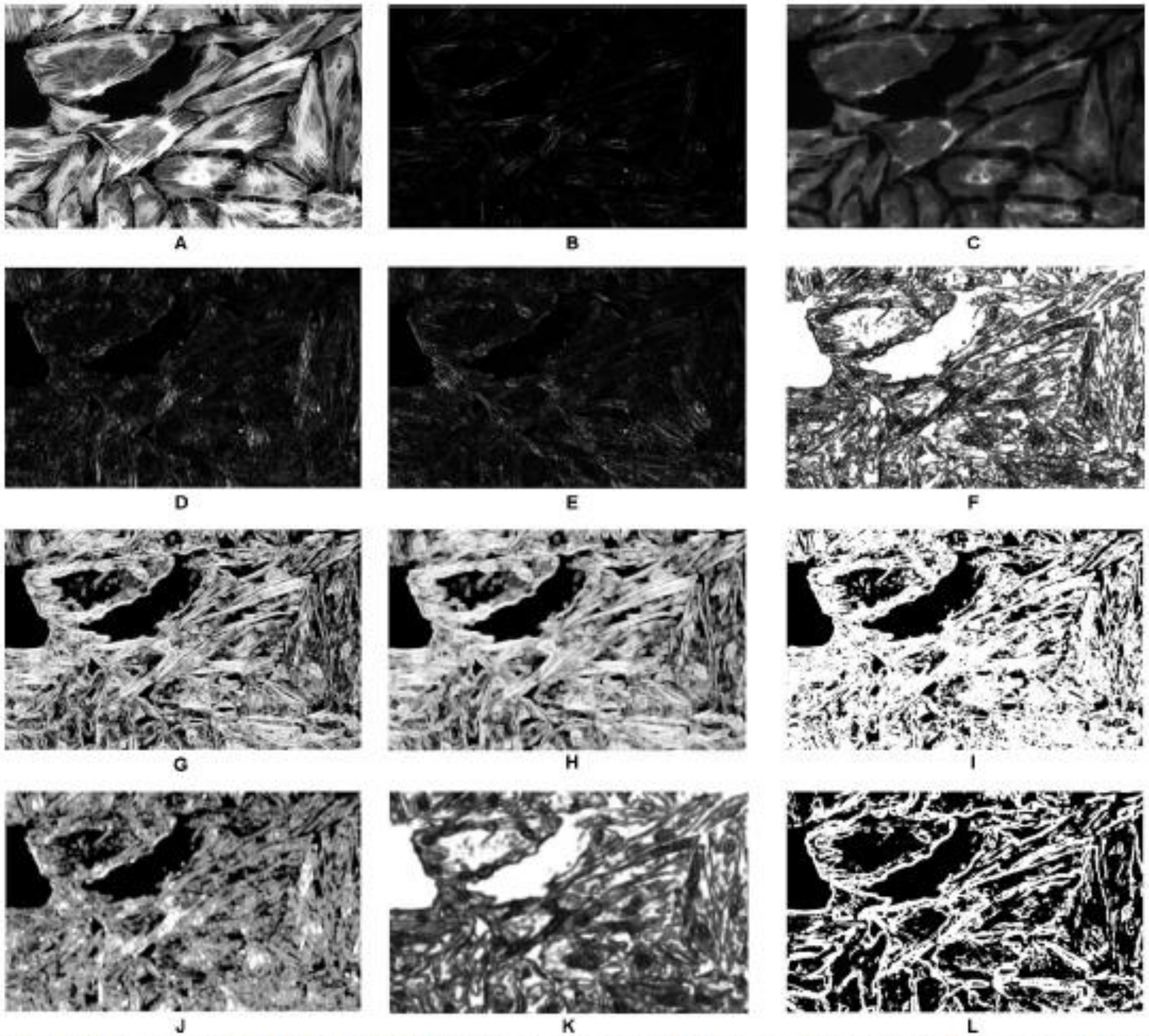


Figure 4 Visual representation of features used by classifiers. Visual representation of features used by classifiers. (a) A pre-processed image, (b) VAR_{3x3}, (c) MIN_{7x7}, (d) $f1/4th0_{5x5}$, (e) $f1/4th3pi/4_{5x5}$, (f) ASM_{5x5}, (g) $IMOC2_{7x7}$, (h) $IMOC2_{9x9}$, (i) ENT_{5x5}, (j) DOE_{7x7}, (k) ASM_{9x9} and (l) outlines obtained from thresholding the output of classifier. The size of the images is 700×430 pixels.

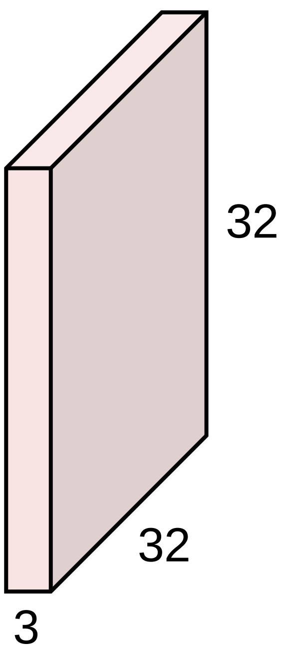
Convolution Layer

What about color image?

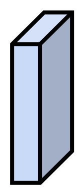
The color image is 3D (x,y,RGB channels) so we need 3D filter too, one for each color channel

Filters always extend the full depth of the input volume

32x32x3 image



5x5x3 filter

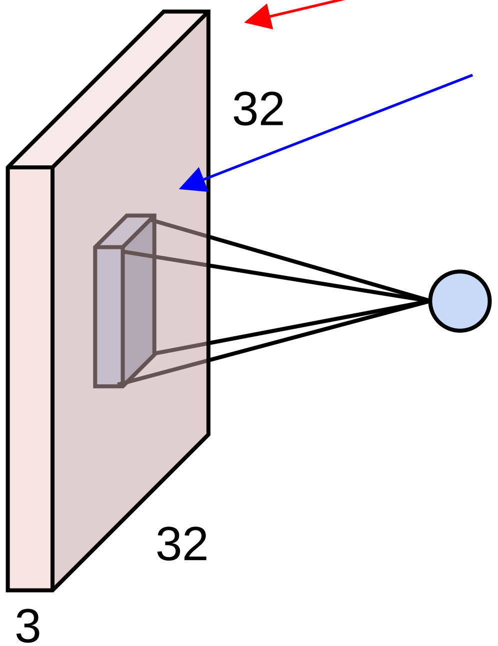


Convolve the filter with the image i.e. “slide over the image spatially, computing dot products”

Convolution Layer

32x32x3 image

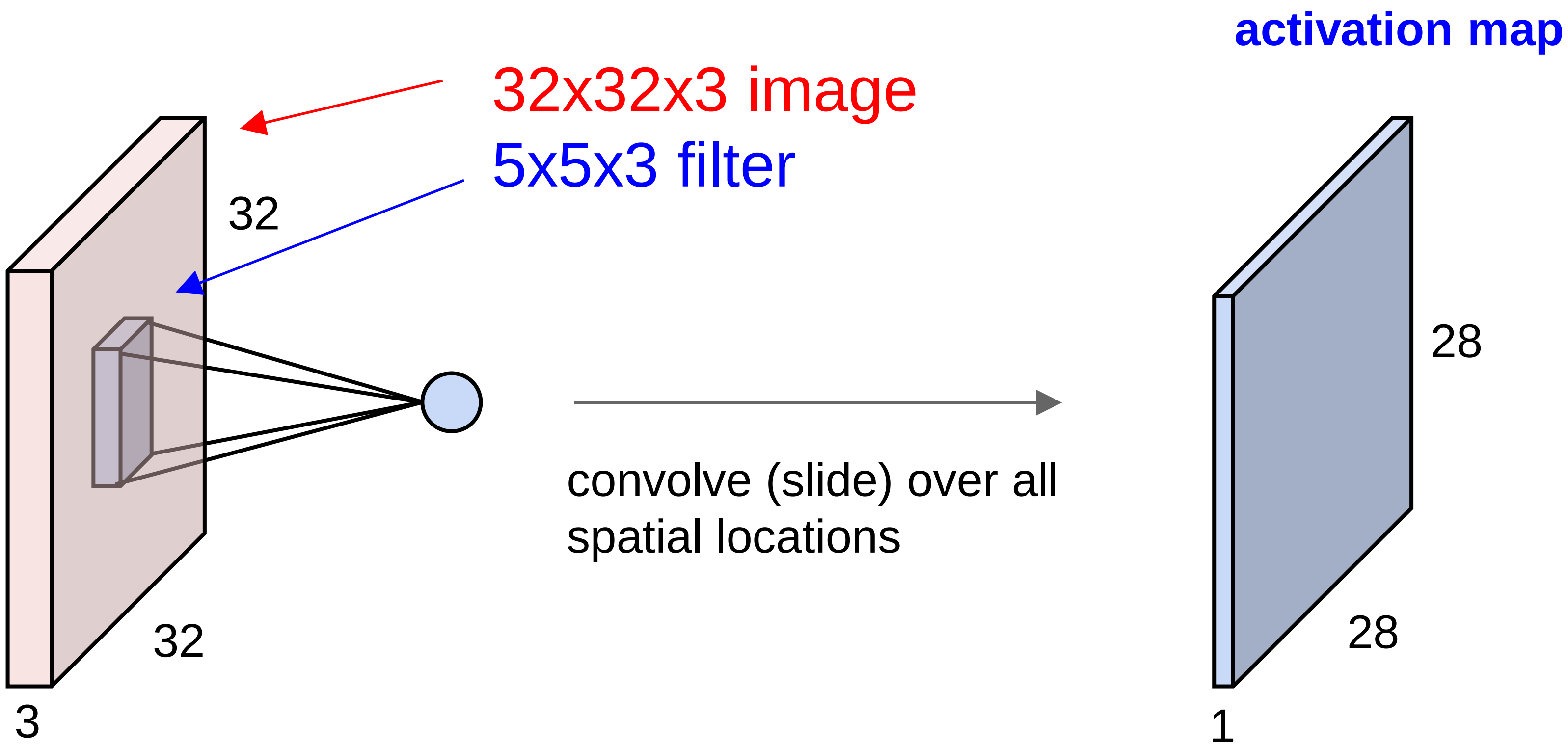
5x5x3 filter w



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)

$$w^T x + b$$

Convolution Layer



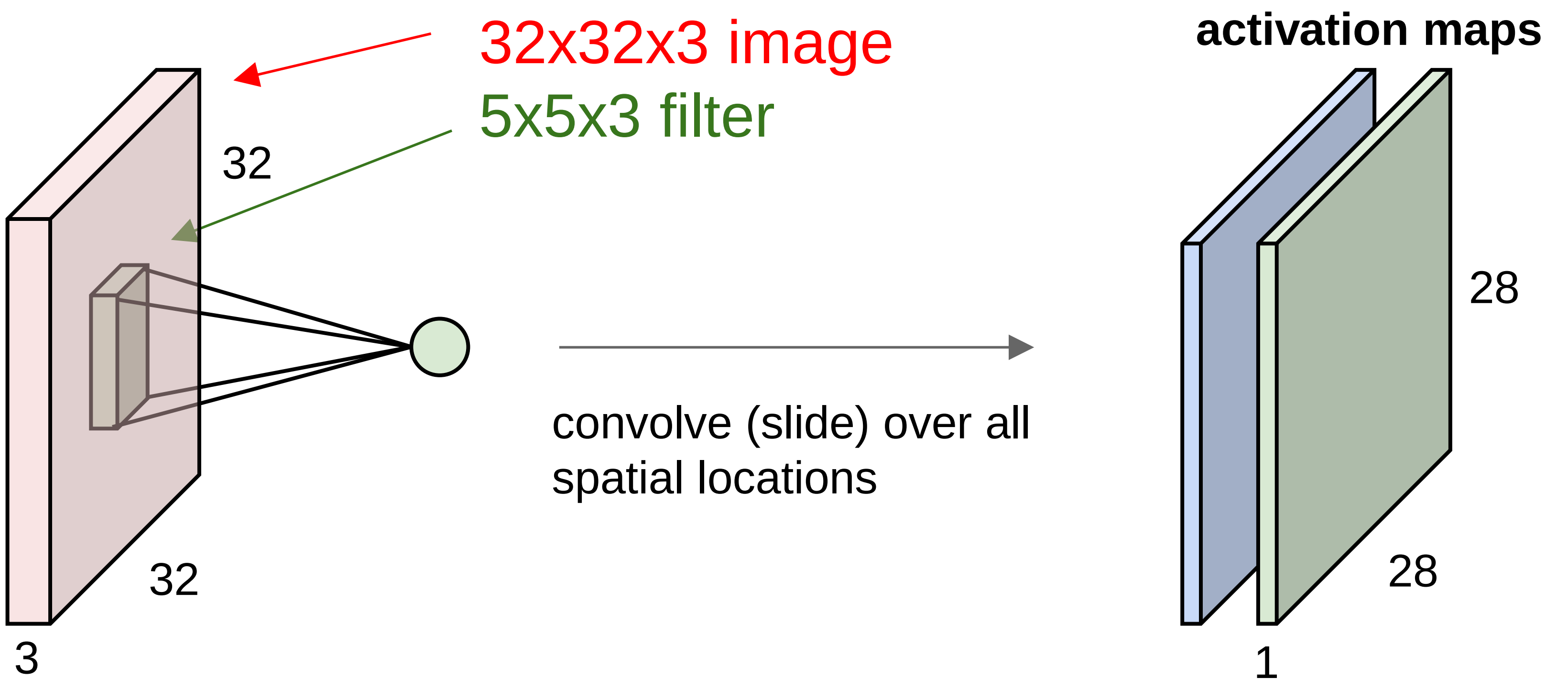
The output size gets reduced if we do not pad the input image border

Q.How many weights involved?

A.5x5x3

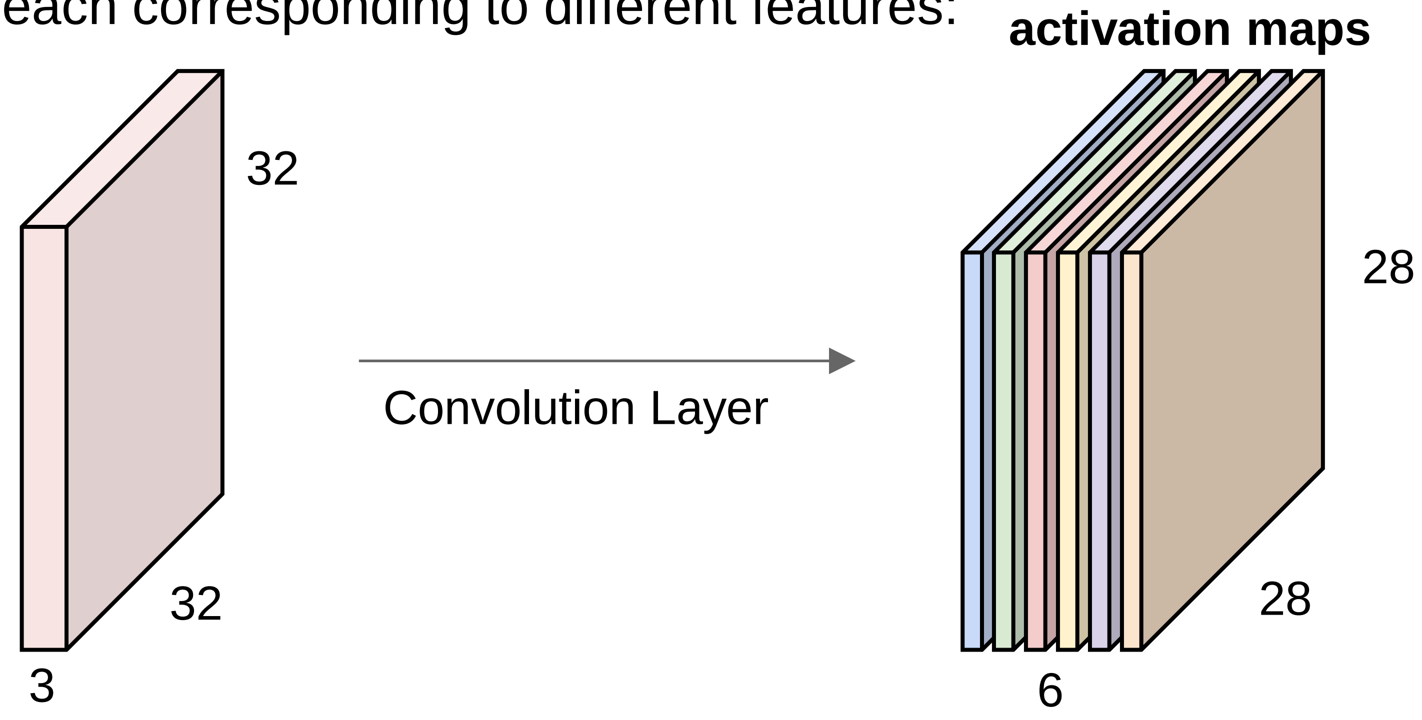
Convolution Layer

consider a second, green filter for some other feature



Convolution Layer

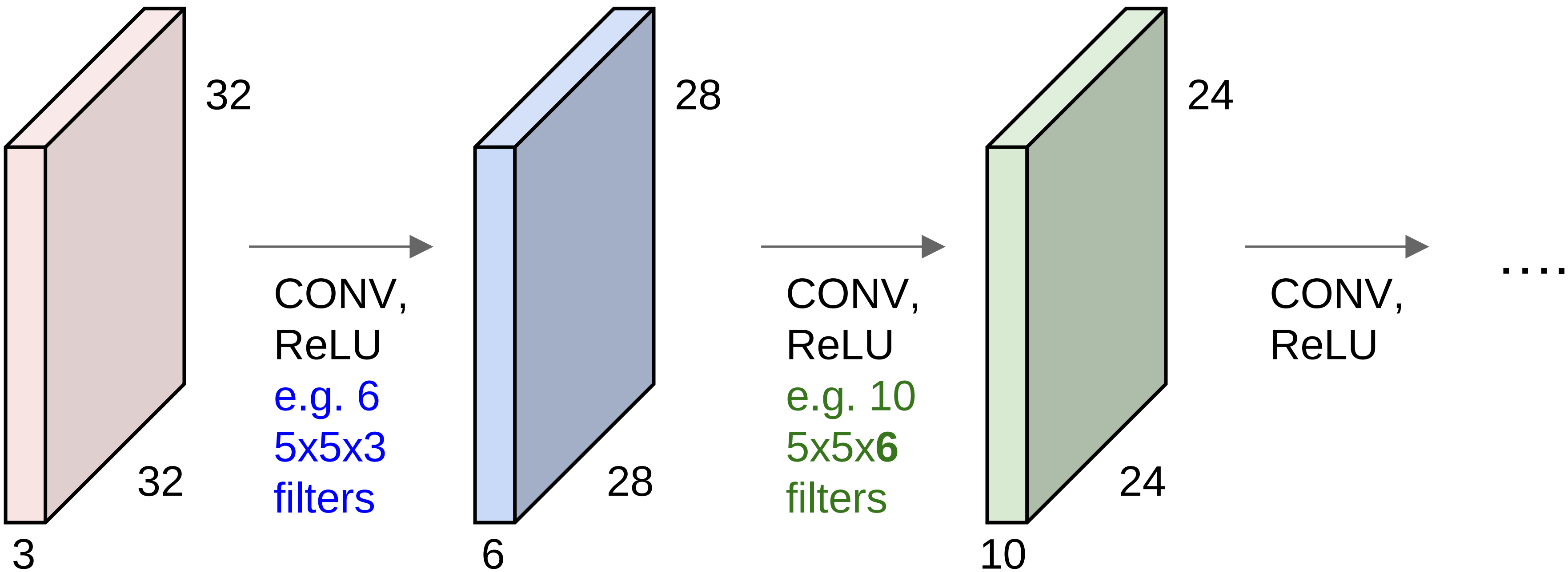
For example, if we had six 5x5x3 filters, we'll get 6 separate activation maps, each corresponding to different features:



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

Convolutional Neural Networks (CNN)

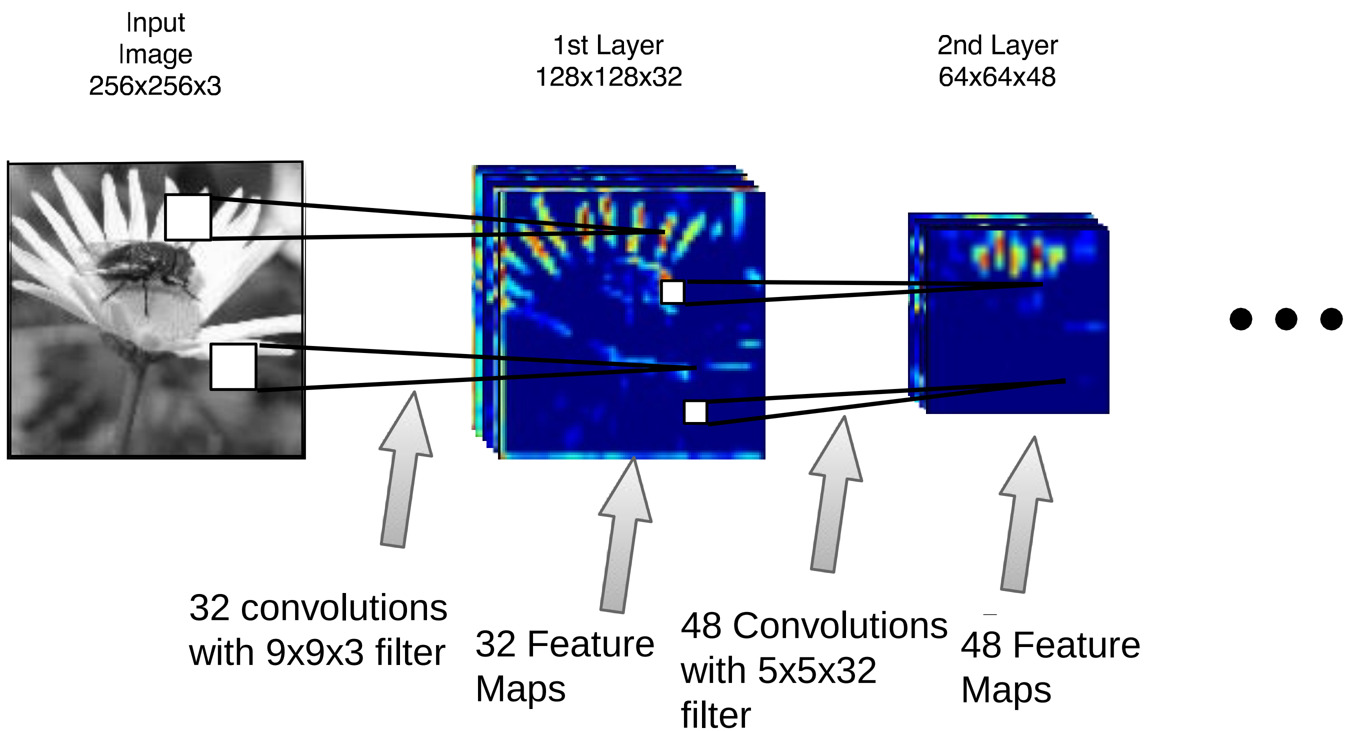
A CNN consists of sequence of convolution layers and nonlinearities



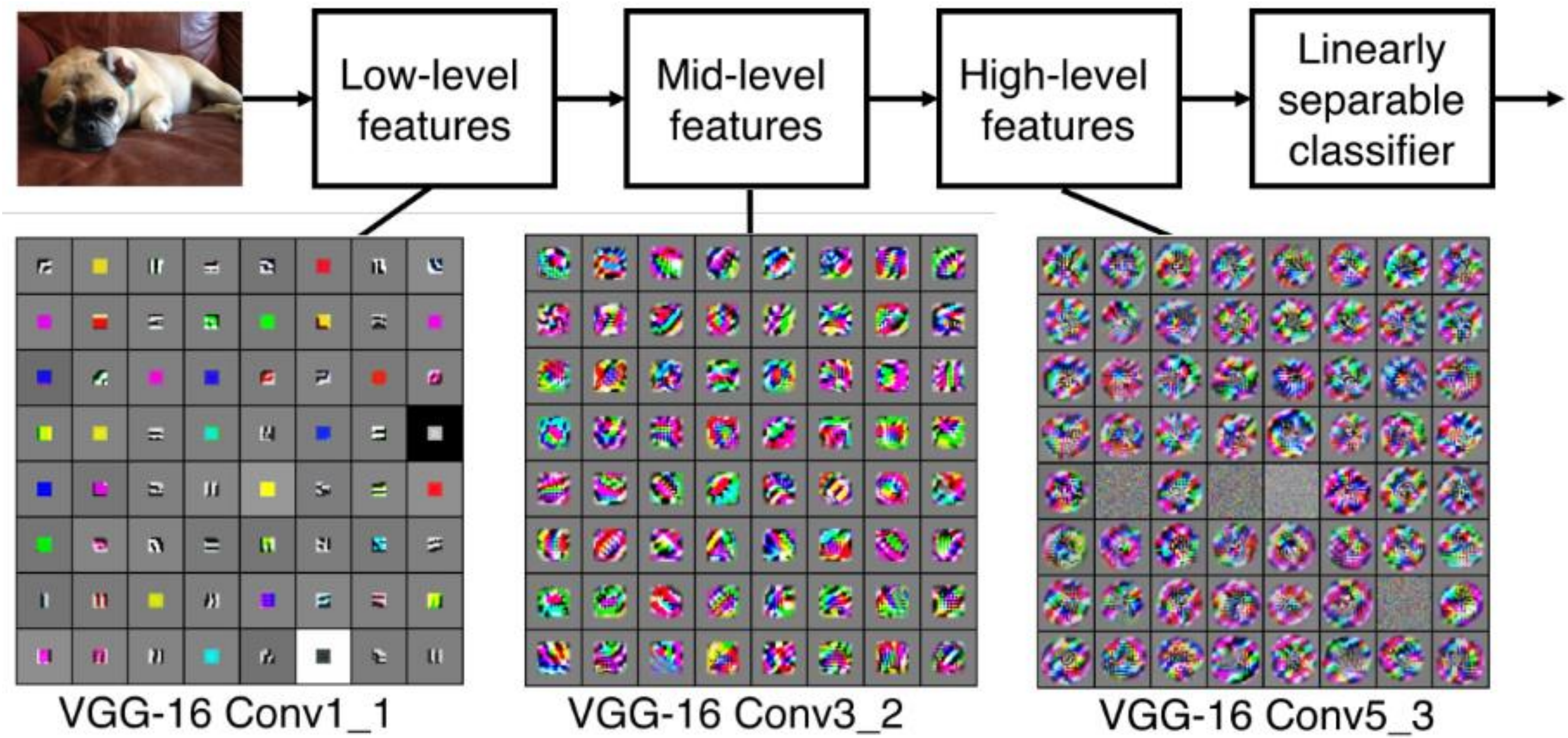
32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially!

(32 -> 28 -> 24 ...)

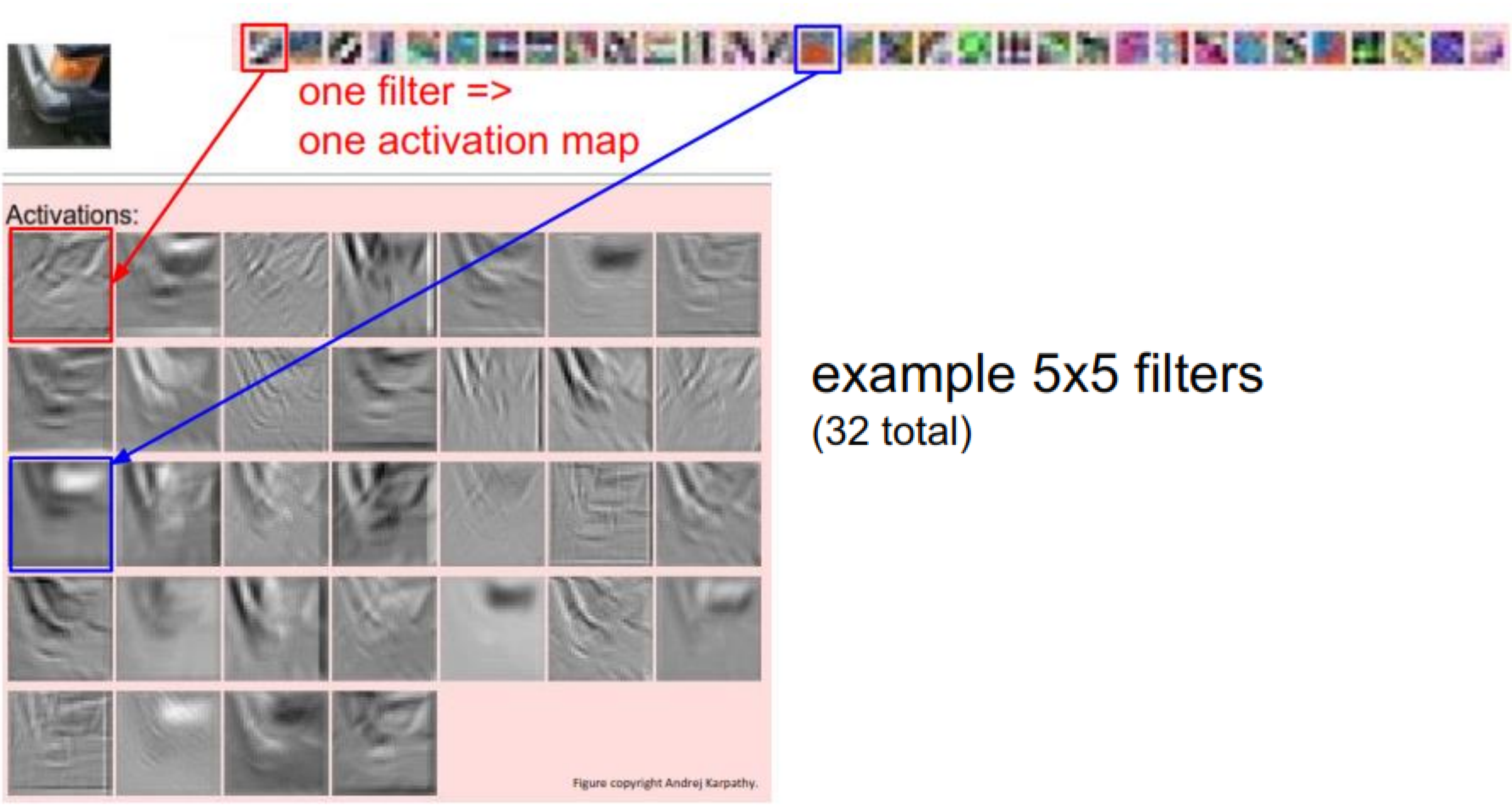
Convolutional Neural Networks (CNN)



Convolutional Neural Networks (CNN)

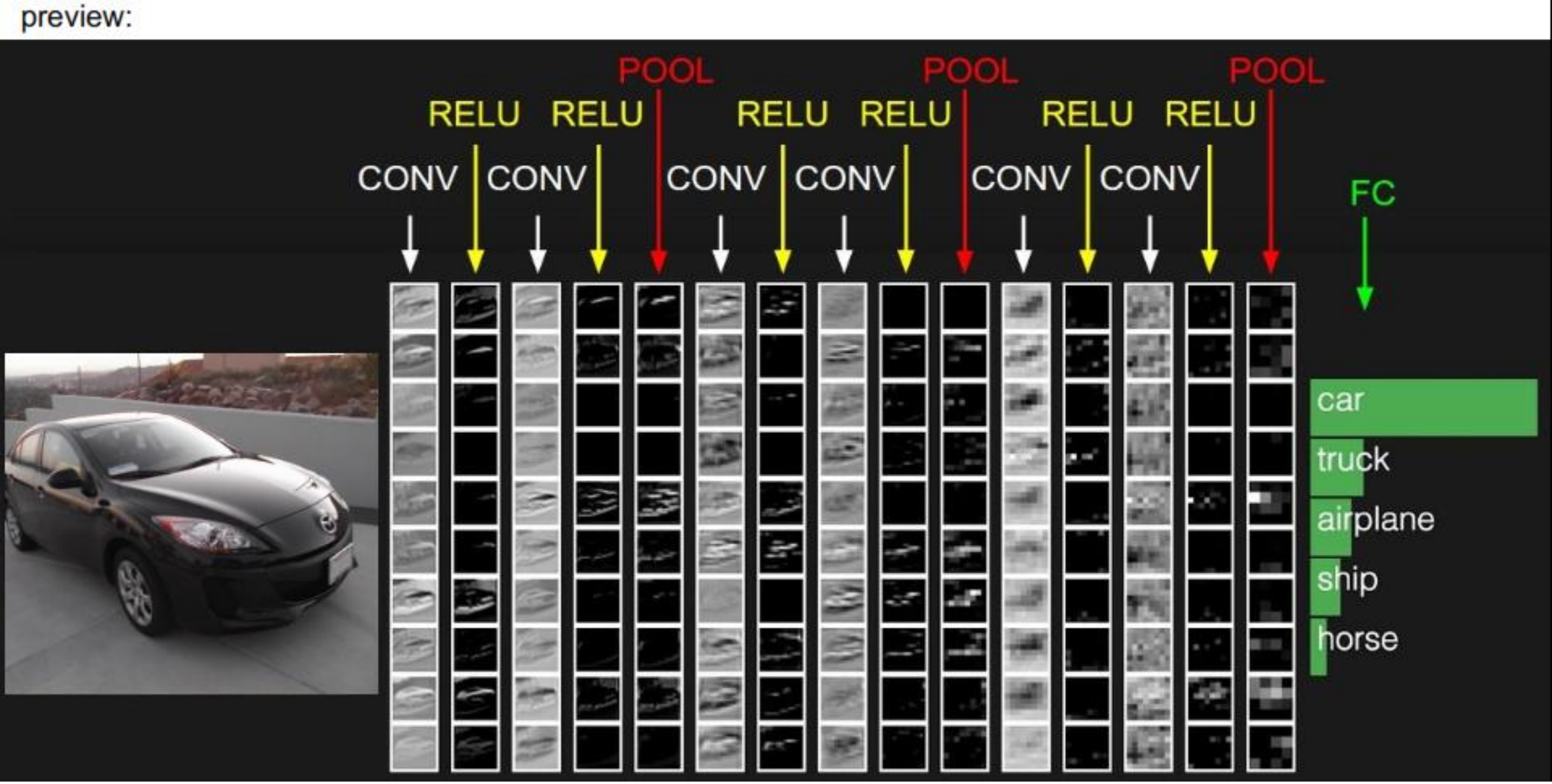


Convolutional Neural Networks (CNN)



example 5x5 filters
(32 total)

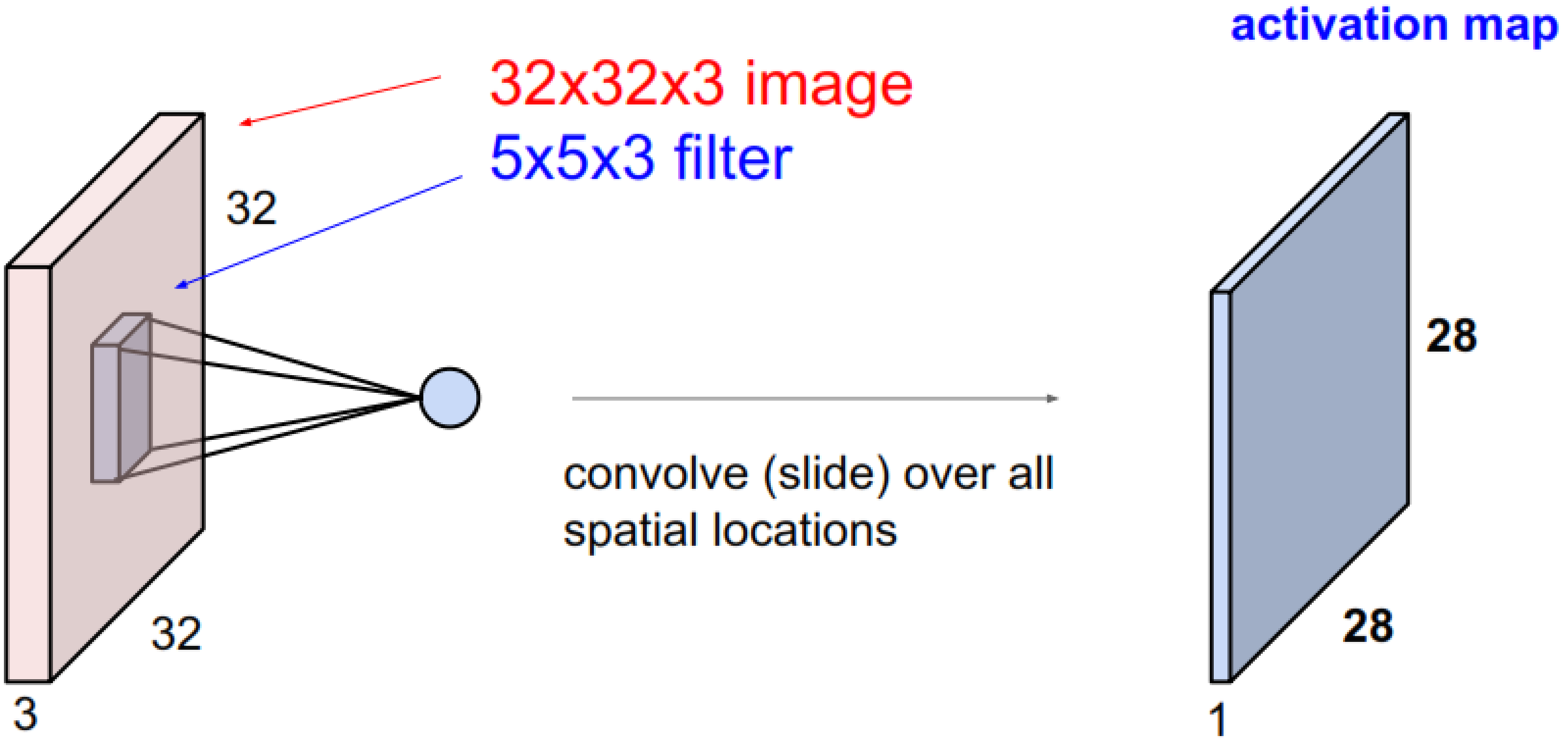
Convolutional Neural Networks (CNN)



The typical structure of a convolutional network repeats the above three elements: Convolution, RELU, and Pooling layers

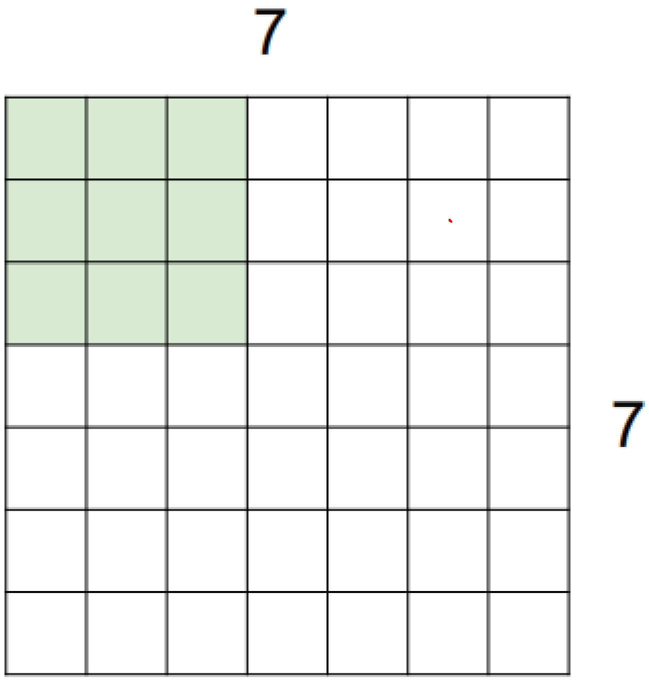
Convolutional Neural Networks (CNN)

A closer look at spatial dimensions:



Convolutional Neural Networks (CNN)

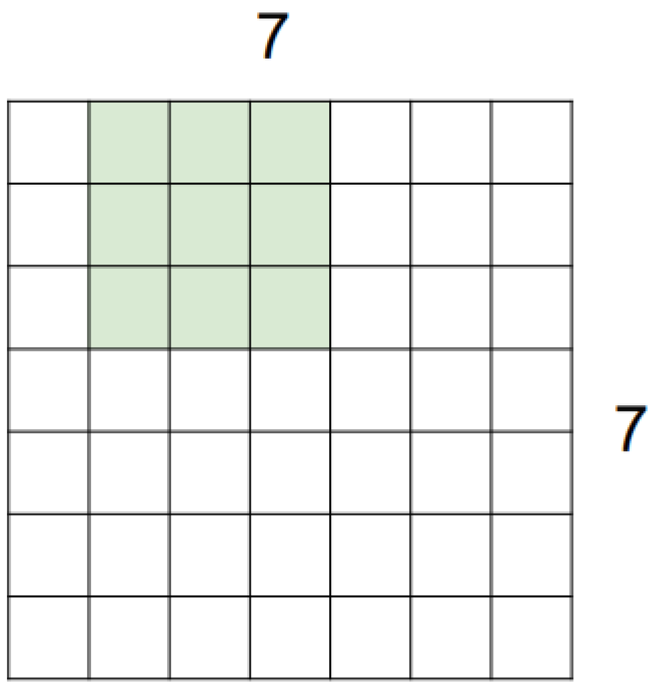
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutional Neural Networks (CNN)

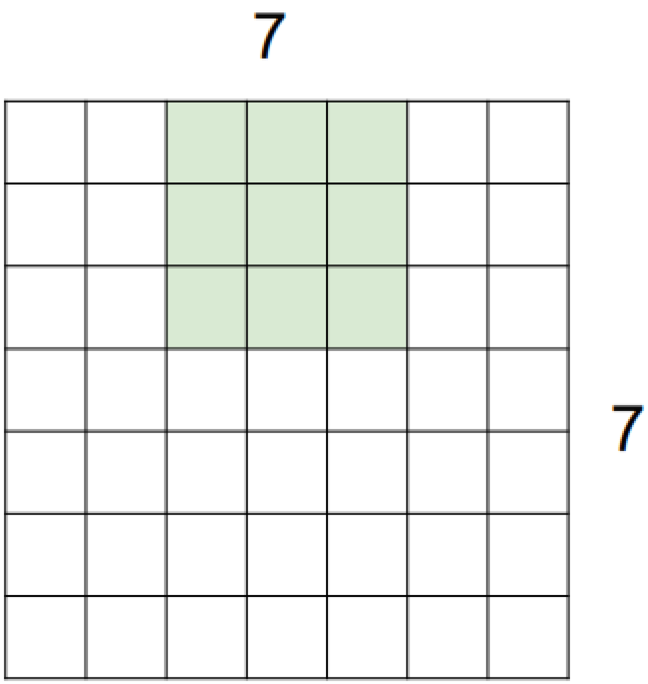
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutional Neural Networks (CNN)

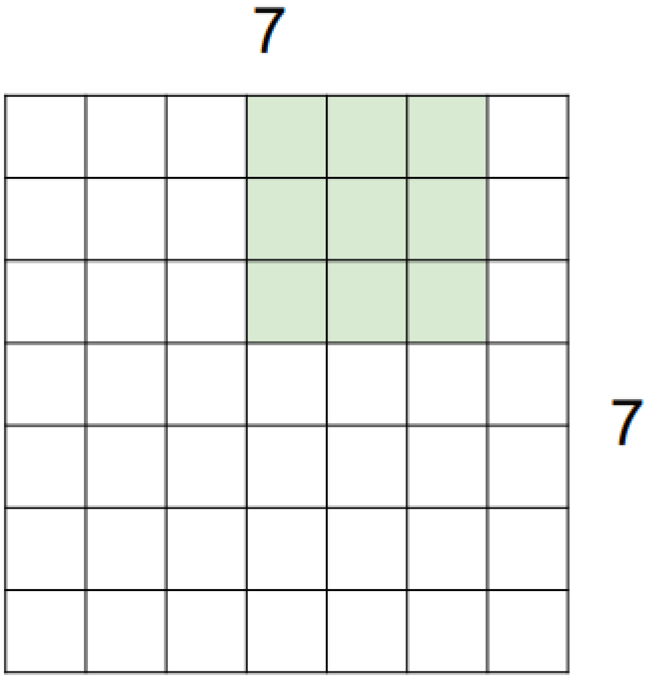
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutional Neural Networks (CNN)

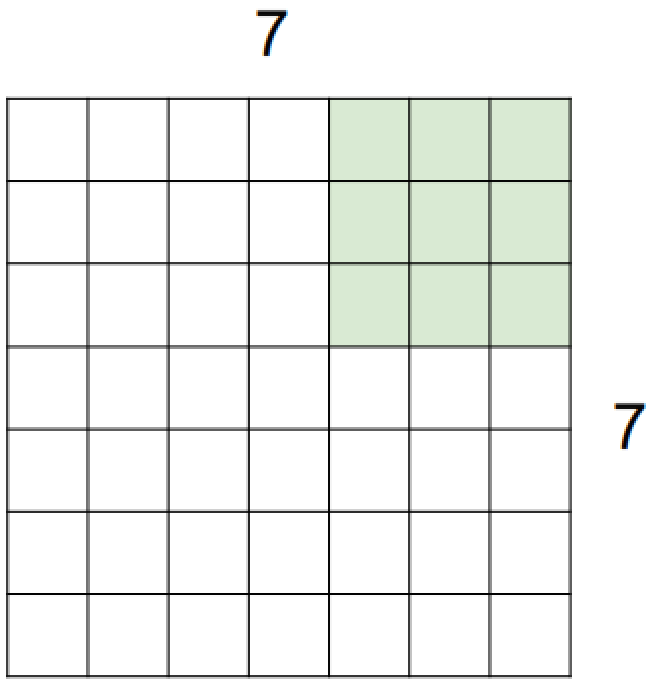
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutional Neural Networks (CNN)

A closer look at spatial dimensions:

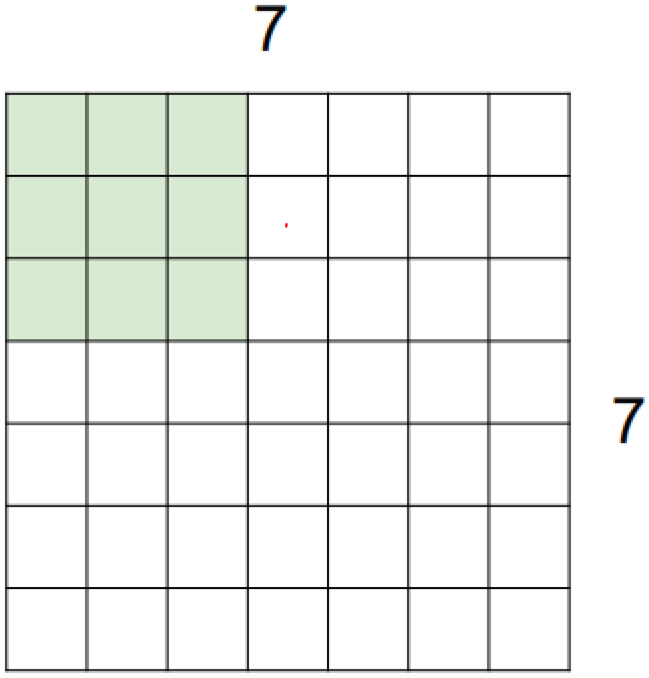


7x7 input (spatially)
assume 3x3 filter

=> 5x5 output

Convolutional Neural Networks (CNN)

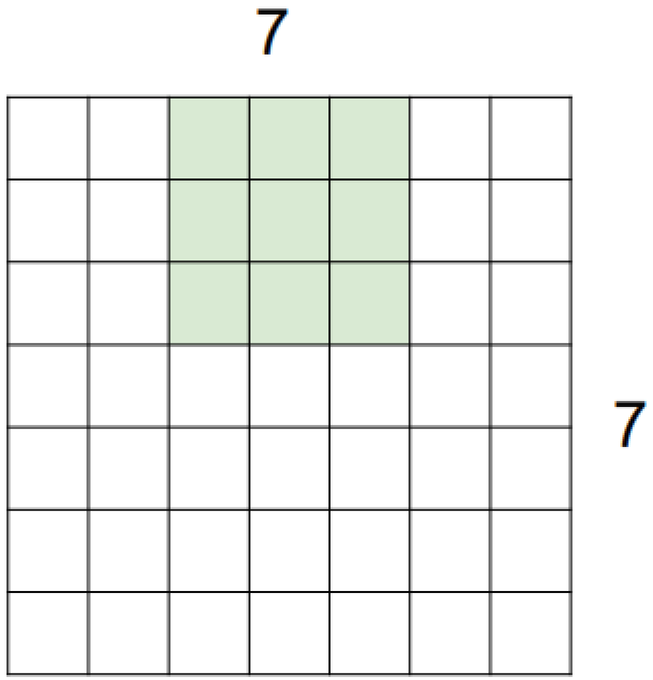
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolutional Neural Networks (CNN)

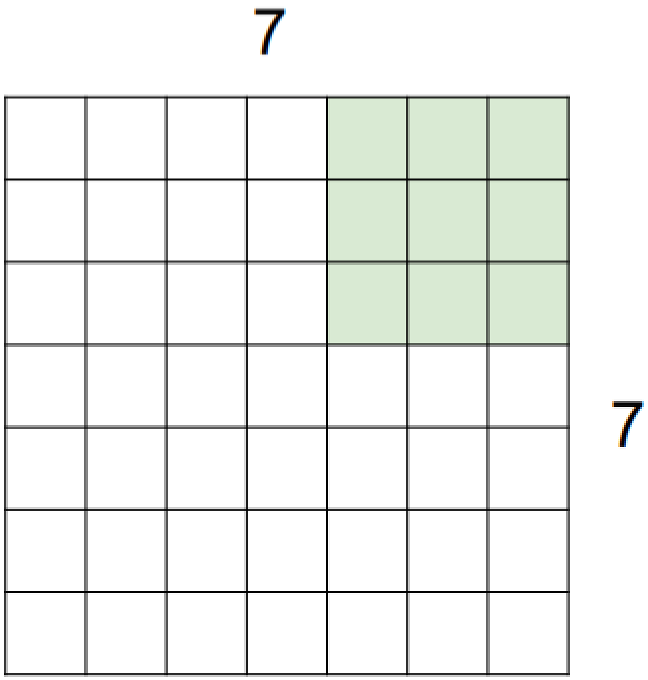
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

Convolutional Neural Networks (CNN)

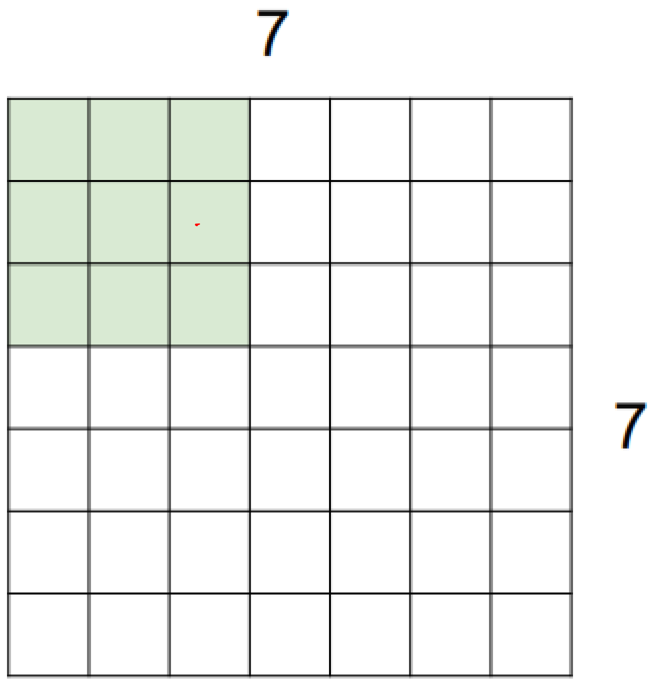
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**
=> 3x3 output!

Convolutional Neural Networks (CNN)

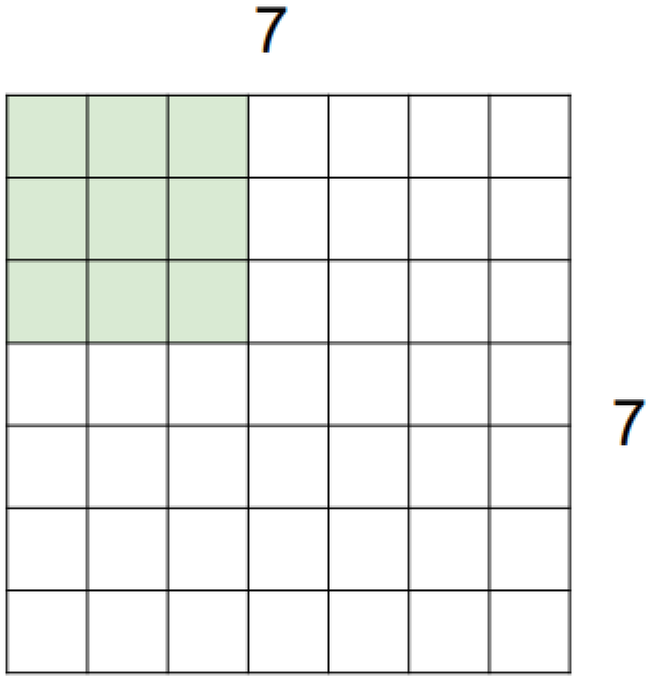
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

Convolutional Neural Networks (CNN)

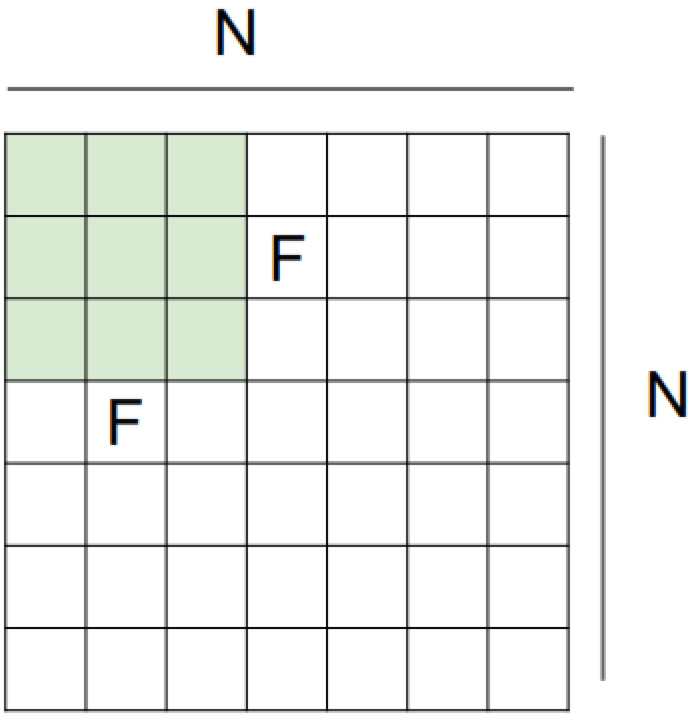
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
applied **with stride 3?**

doesn't fit!
cannot apply 3x3 filter on
7x7 input with stride 3.

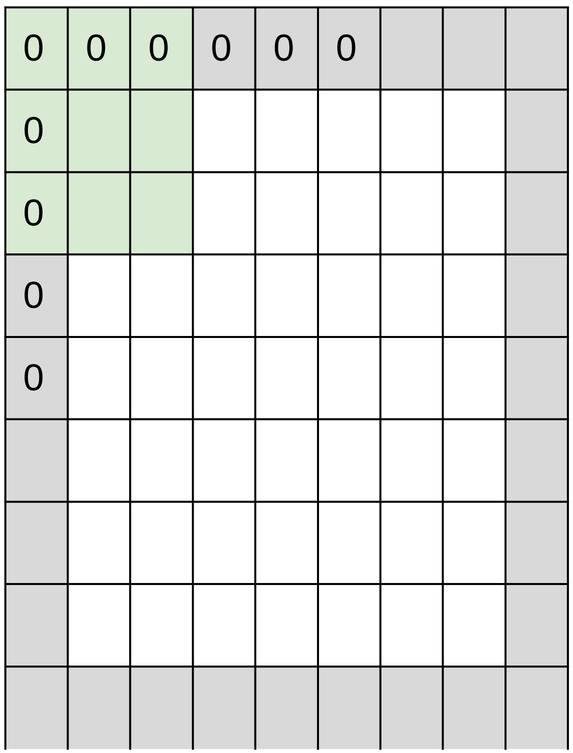
Convolutional Neural Networks (CNN)



Output size:
(N - F) / stride + 1

e.g. N = 7, F = 3:
stride 1 => (7 - 3)/1 + 1 = 5
stride 2 => (7 - 3)/2 + 1 = 3
stride 3 => (7 - 3)/3 + 1 = 2.33 :\

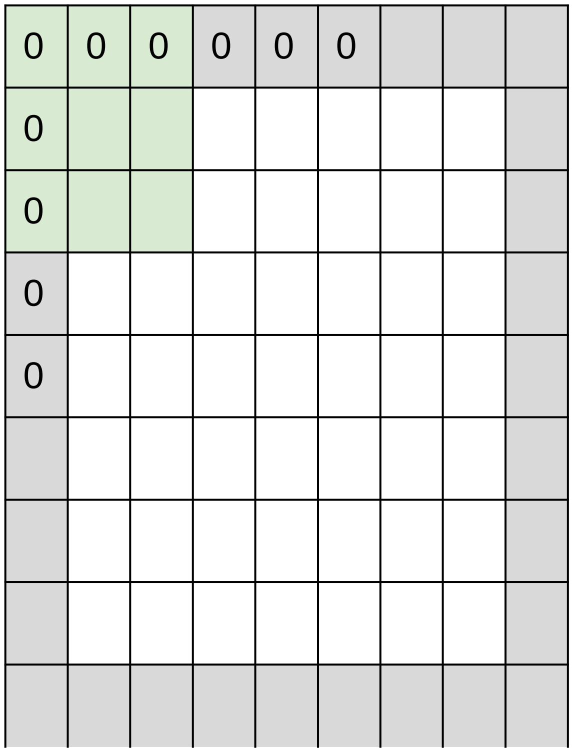
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

(recall:)
(N - F) / stride + 1

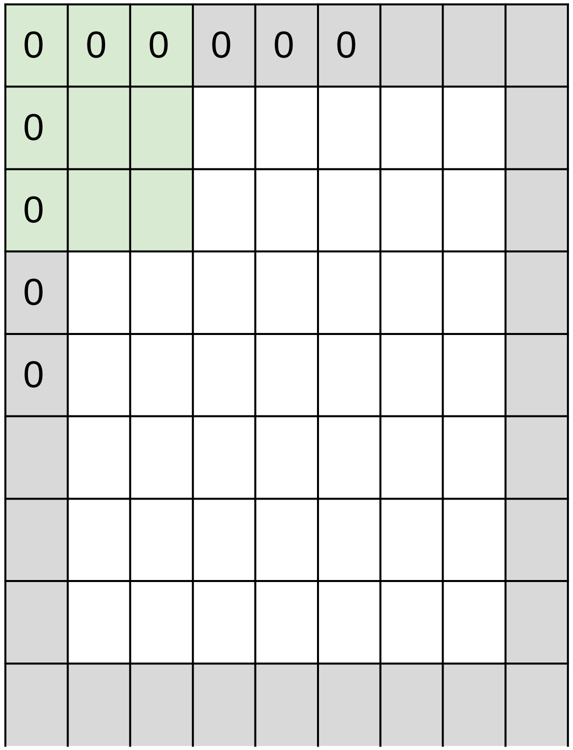
In practice: Common to zero pad the border



e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

In practice: Common to zero pad the border

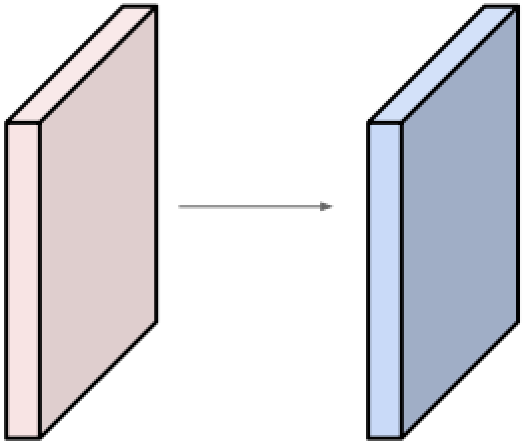


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1
F = 5 => zero pad with 2
F = 7 => zero pad with 3

Convolutional Neural Networks (CNN)

Examples time:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

Number of parameters in this layer?

each filter has 5*5*3 + 1 = 76 params (+1 for bias)

=> 76*10 = **760**

Summary. To summarize, the Conv Layer:

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

Summary. To summarize, the Conv Layer:

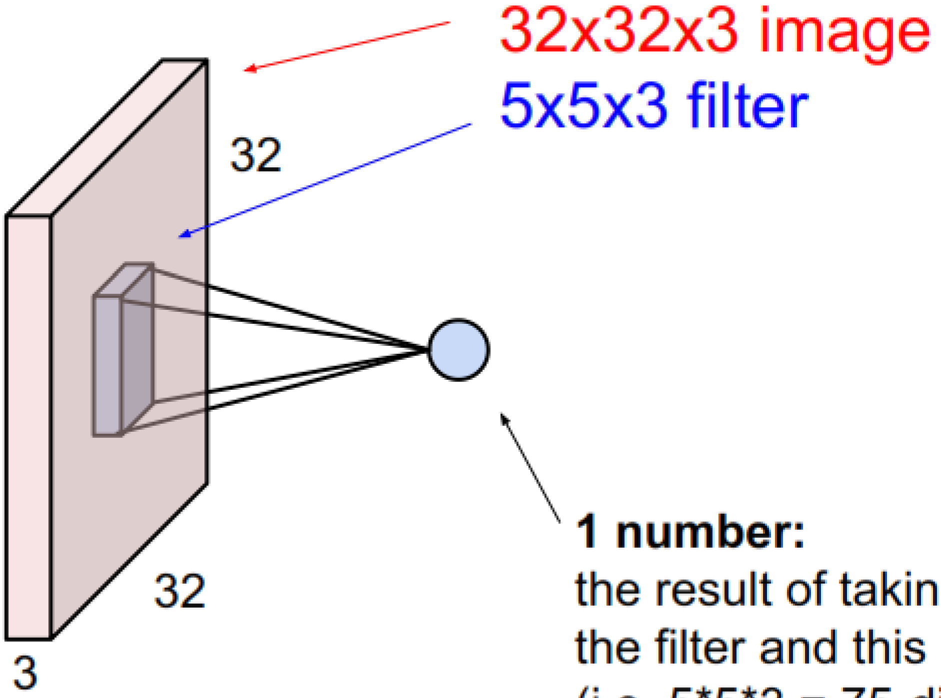
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (i.e. width and height are computed equally by symmetry)
  - $D_2 = K$
- With parameter sharing, it introduces $F \cdot F \cdot D_1$ weights per filter, for a total of $(F \cdot F \cdot D_1) \cdot K$ weights and K biases.
- In the output volume, the d-th depth slice (of size $W_2 \times H_2$) is the result of performing a valid convolution of the d-th filter over the input volume with a stride of S, and then offset by d-th bias.

K = (powers of 2, e.g. 32, 64, 128, 512)

- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Convolutional Neural Networks (CNN)

The brain/neuron view of CONV Layer



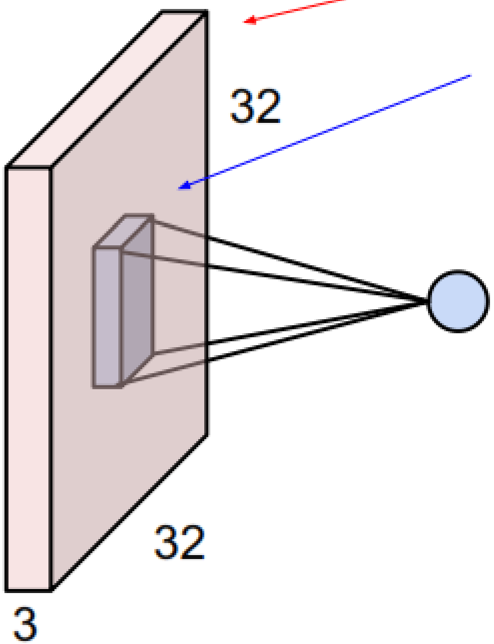
1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

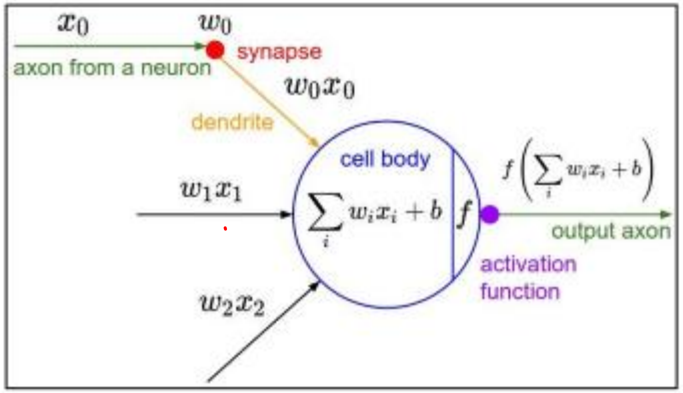
Convolutional Neural Networks (CNN)

The brain/neuron view of CONV Layer

32x32x3 image

5x5x3 filter



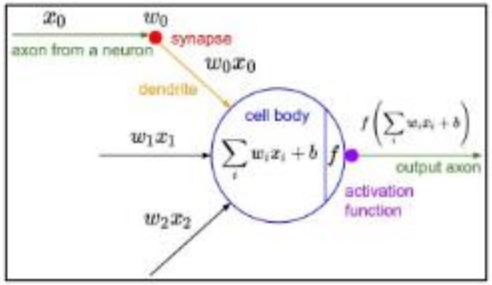


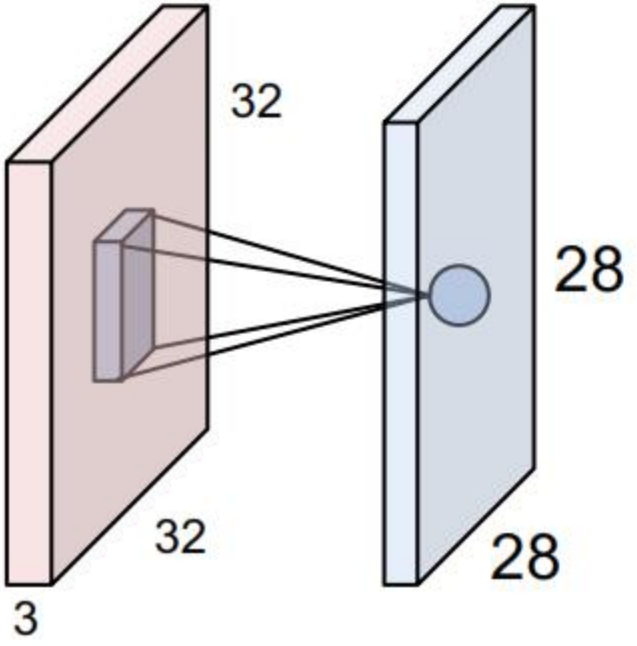
It's just a neuron with local connectivity...

1 number:
the result of taking a dot product between the filter and this part of the image
(i.e. 5*5*3 = 75-dimensional dot product)

Convolutional Neural Networks (CNN)

The brain/neuron view of CONV Layer





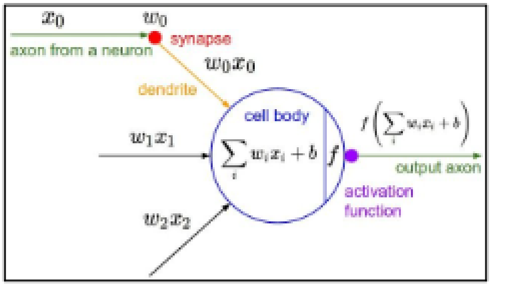
An activation map is a 28x28 sheet of neuron outputs:

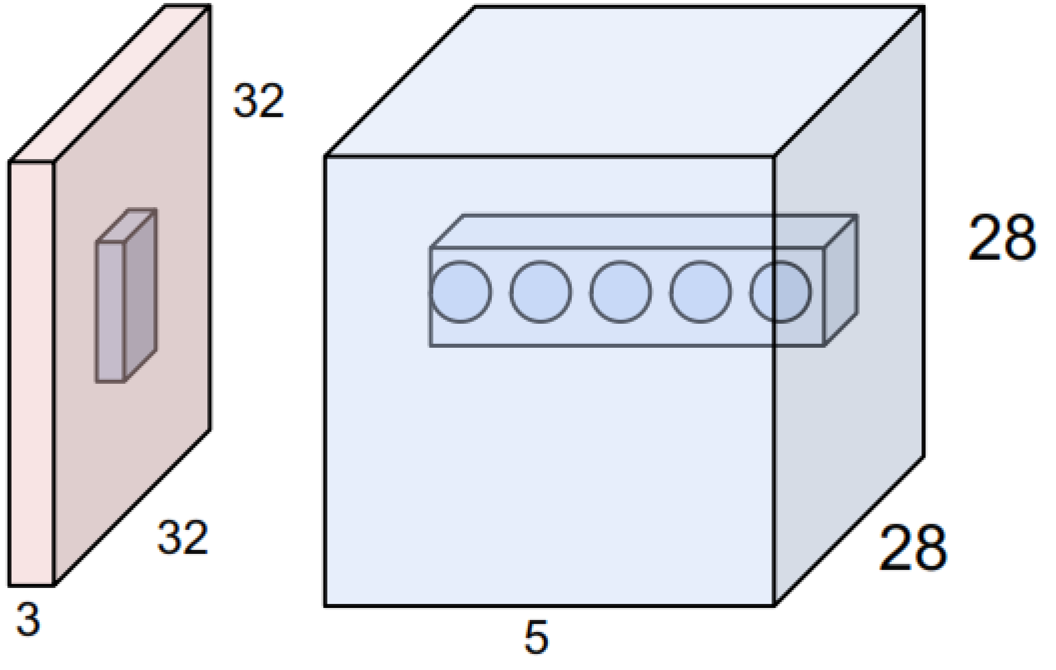
1. Each is connected to a small region in the input
2. All of them share parameters

“5x5 filter” -> “5x5 receptive field for each neuron”

Convolutional Neural Networks (CNN)

The brain/neuron view of CONV Layer





E.g. with 5 filters, CONV layer consists of neurons arranged in a 3D grid (28x28x5)

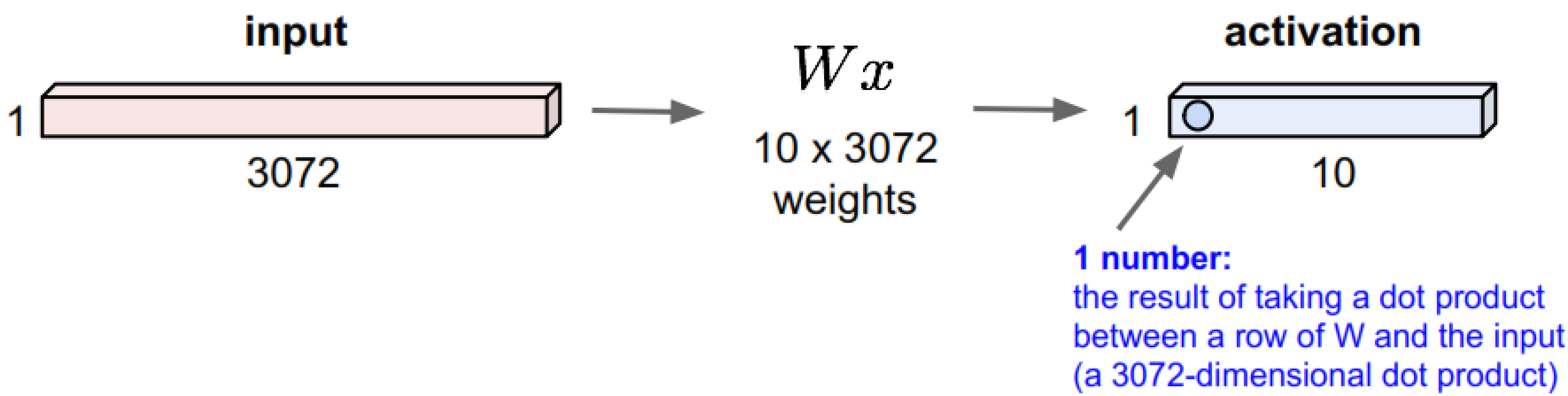
There will be 5 different neurons all looking at the same region in the input volume

Convolutional Neural Networks (CNN)

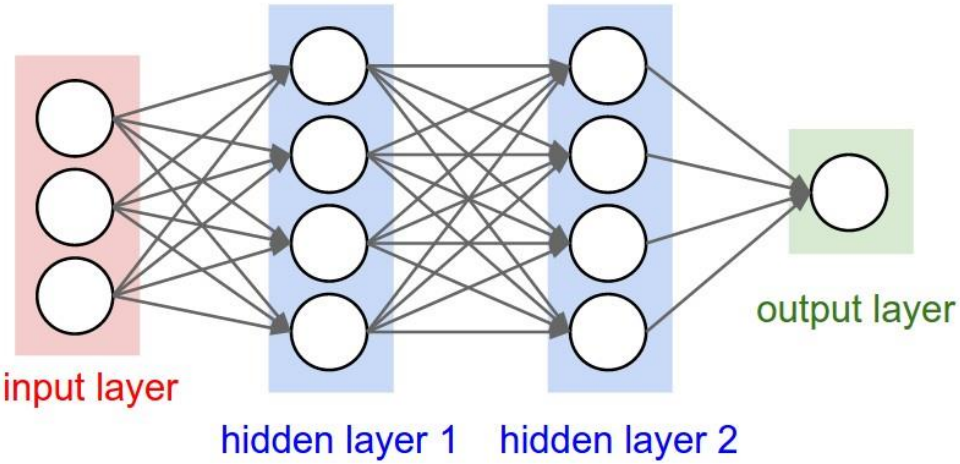
Reminder: Fully Connected Layer

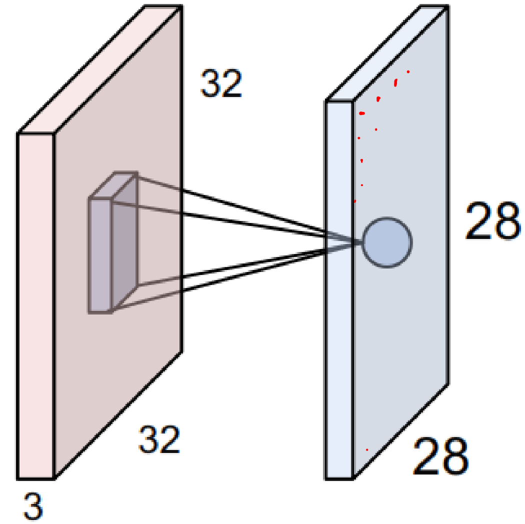
32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume



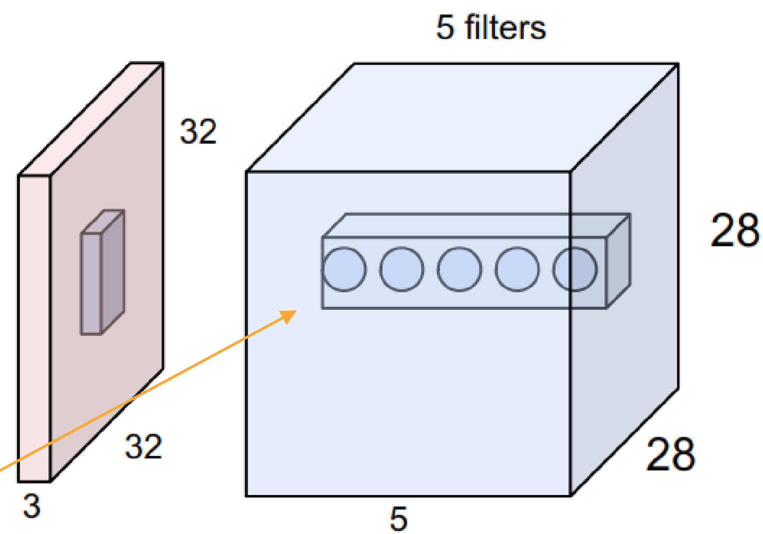
Fully Connected Feedforward Network Vs CNN





Fully connected or Dense

Each neuron looks at the full input volume



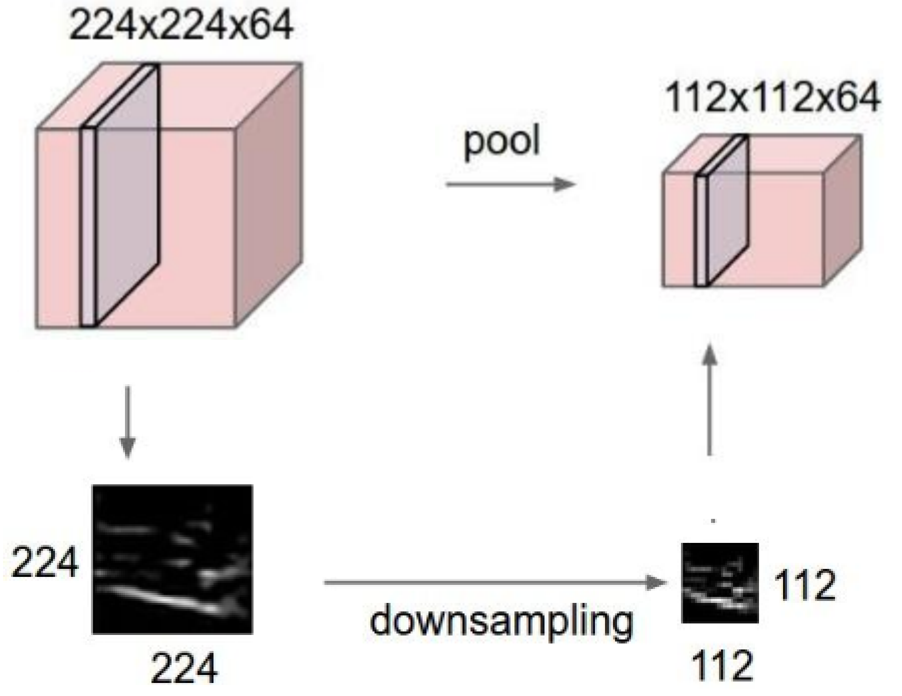
5 Different Neurons all looking at the same region in the input volume

Sparse

Convolutional Neural Networks (CNN)

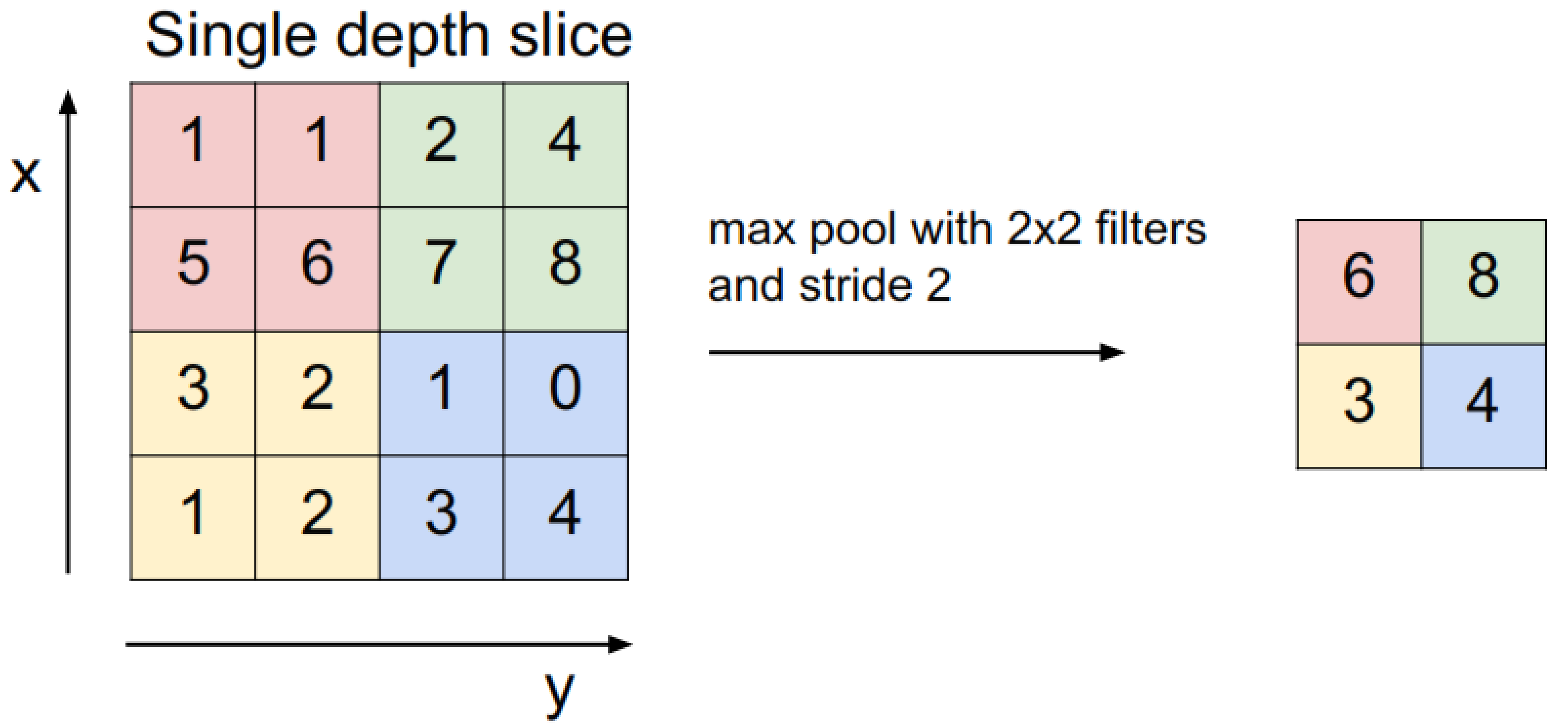
Pooling layer

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



Convolutional Neural Networks (CNN)

MAX POOLING



Convolutional Neural Networks (CNN)

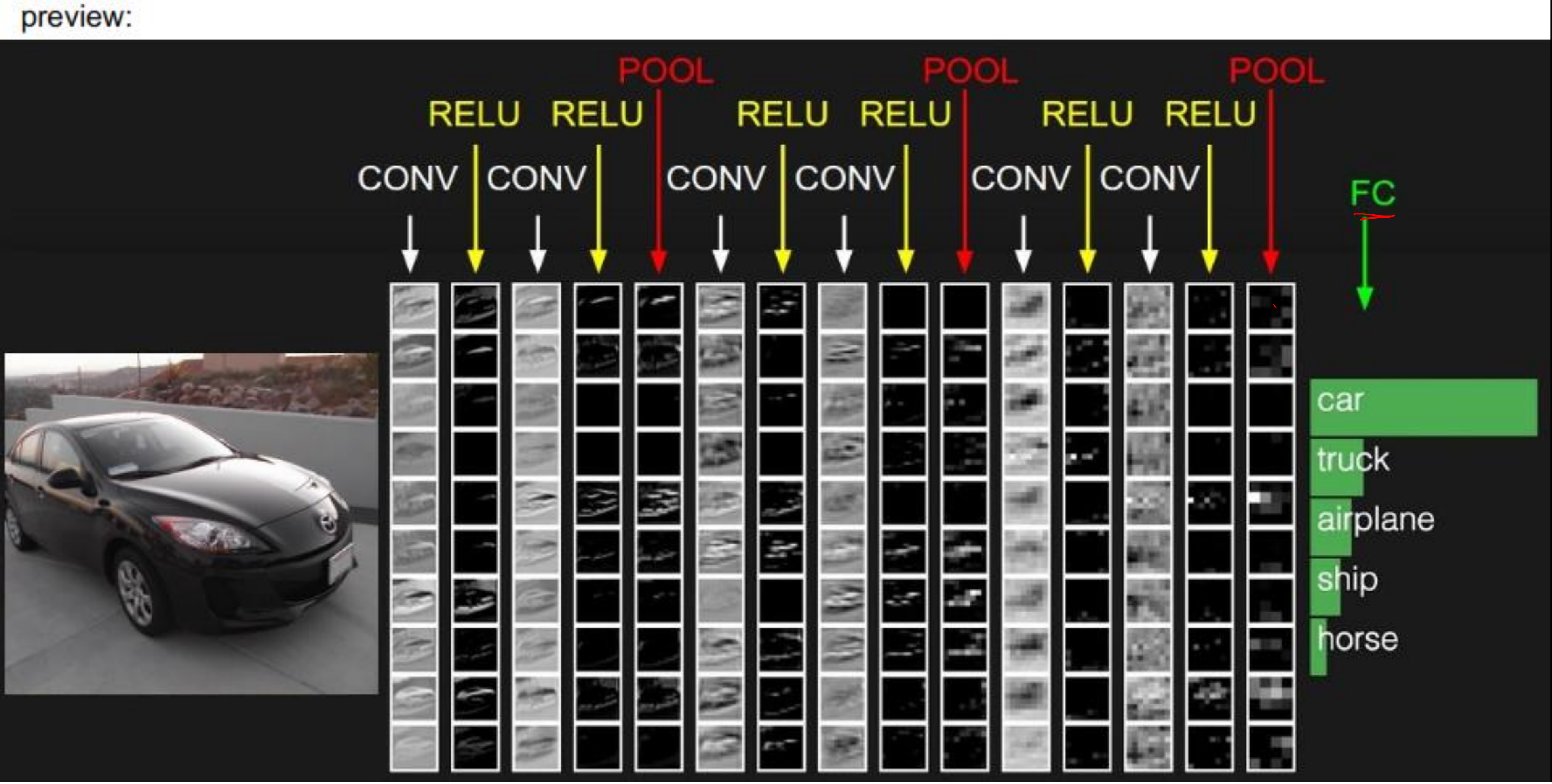
MAX POOLING

- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires three hyperparameters:
  - their spatial extent F,
  - the stride S,
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F)/S + 1$
  - $H_2 = (H_1 - F)/S + 1$
  - $D_2 = D_1$
- Introduces zero parameters since it computes a fixed function of the input
- Note that it is not common to use zero-padding for Pooling layers

Common settings:

F = 2, S = 2
F = 3, S = 2

Recap: Convolutional Neural Networks (CNN)

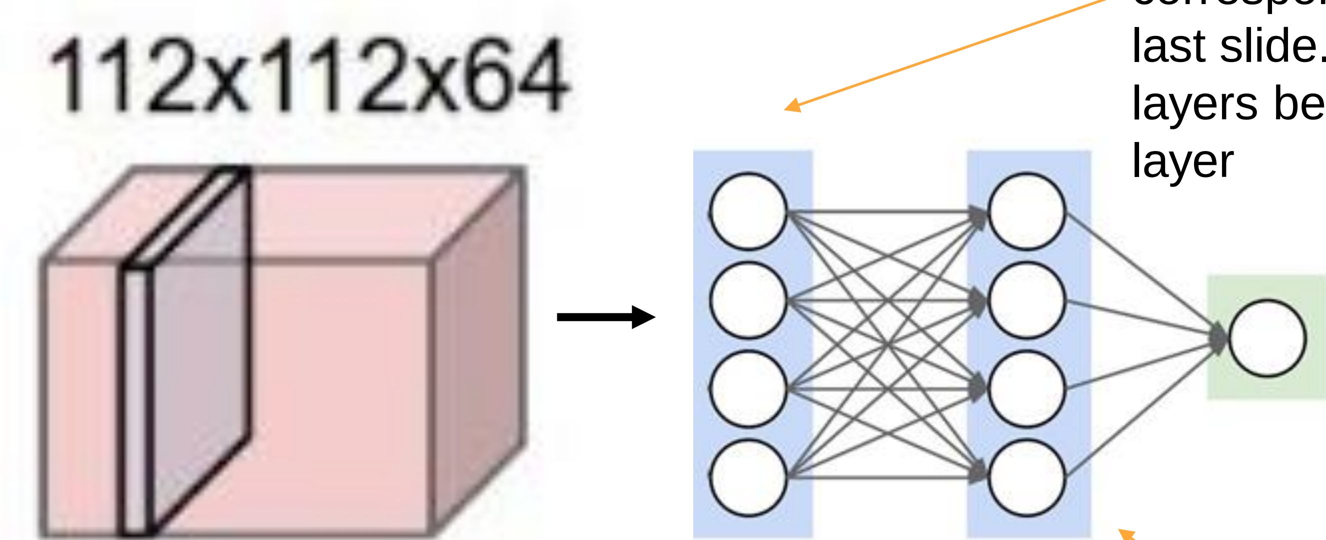


The typical structure of a convolutional network repeats the above three elements: Convolution, RELU, and Pooling layers

Convolutional Neural Networks (CNN)

Fully Connected (FC) Layer

This is not a layer but showing a flattened pooling output corresponding to architecture in last slide. There may be more FC layers between this and output layer



Output of pooling layer

FC layer

The number of nodes here is equal to number of classes

Only 4 nodes are shown but they will be 112x112x64 nodes obtained by flattening the output of pooling layer