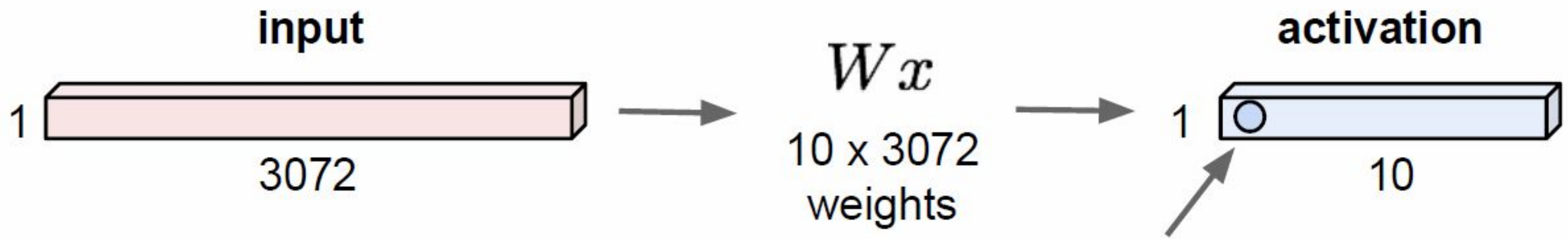
Convolutional Neural Nets

Fully Connected Layer

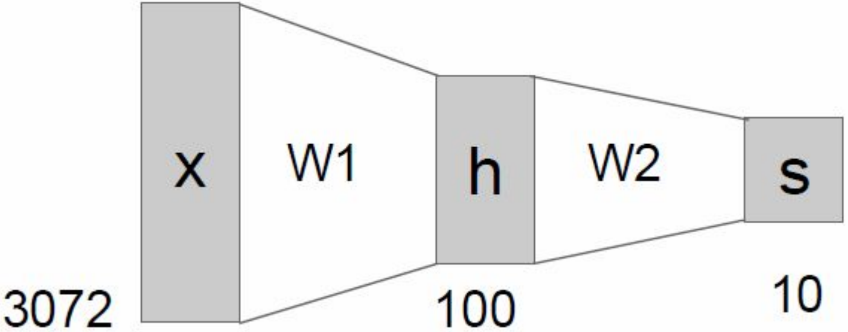
32x32x3 image -> stretch to 3072 x 1

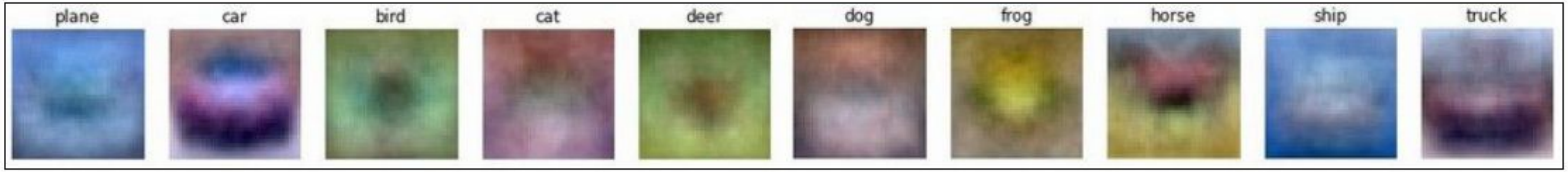


1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

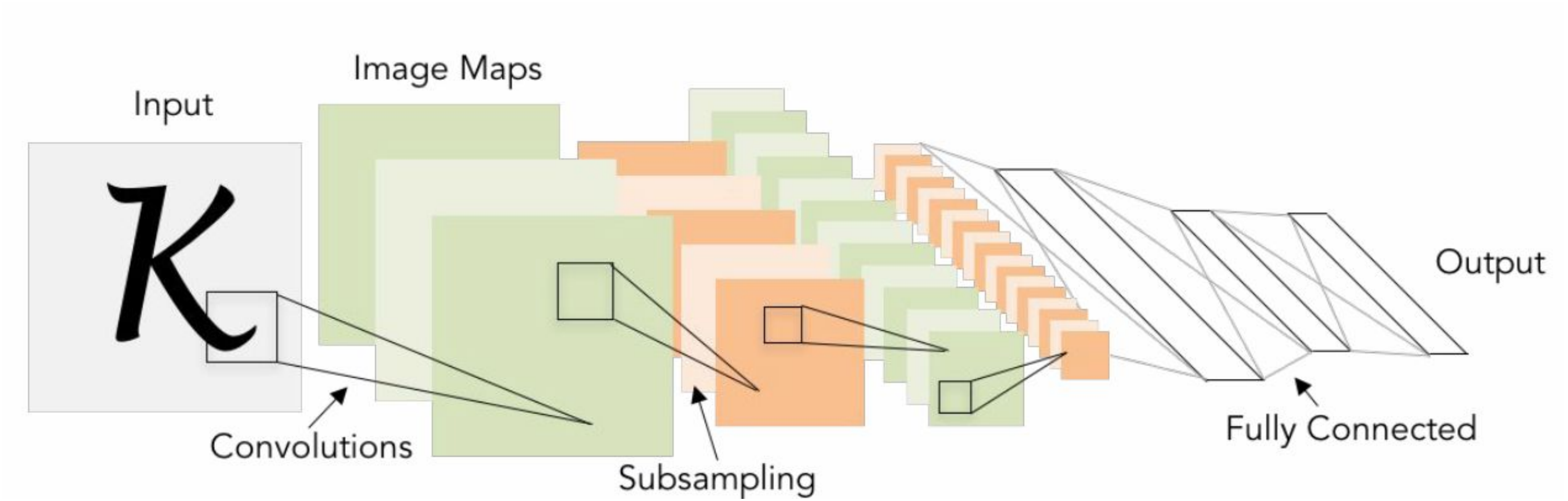
Linear score function: $f = Wx$

2-layer Neural Network $f = W_2 \max(0, W_1 x)$

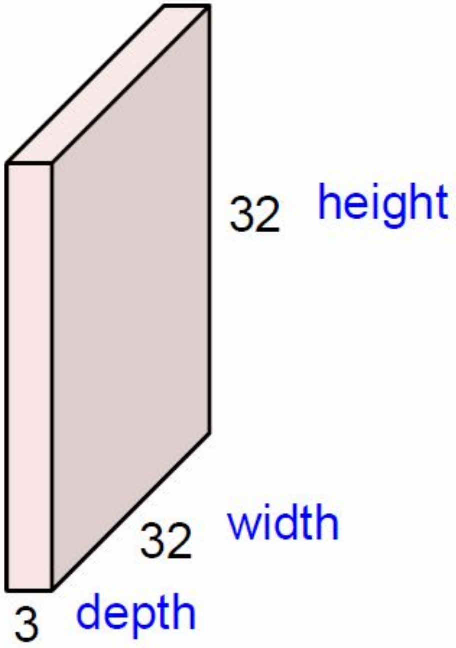




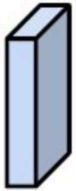
Convolutional Neural Net



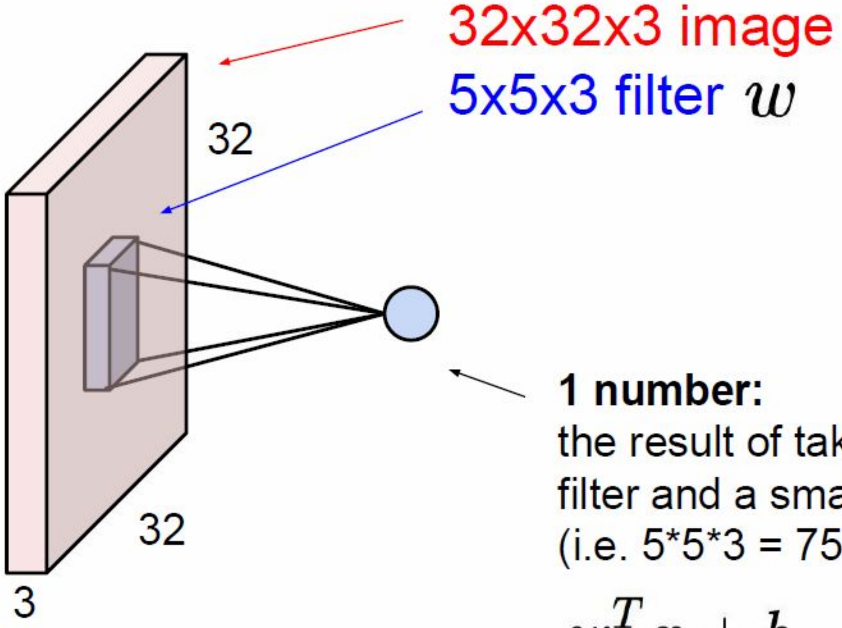
32x32x3 image -> preserve spatial structure



5x5x3 filter



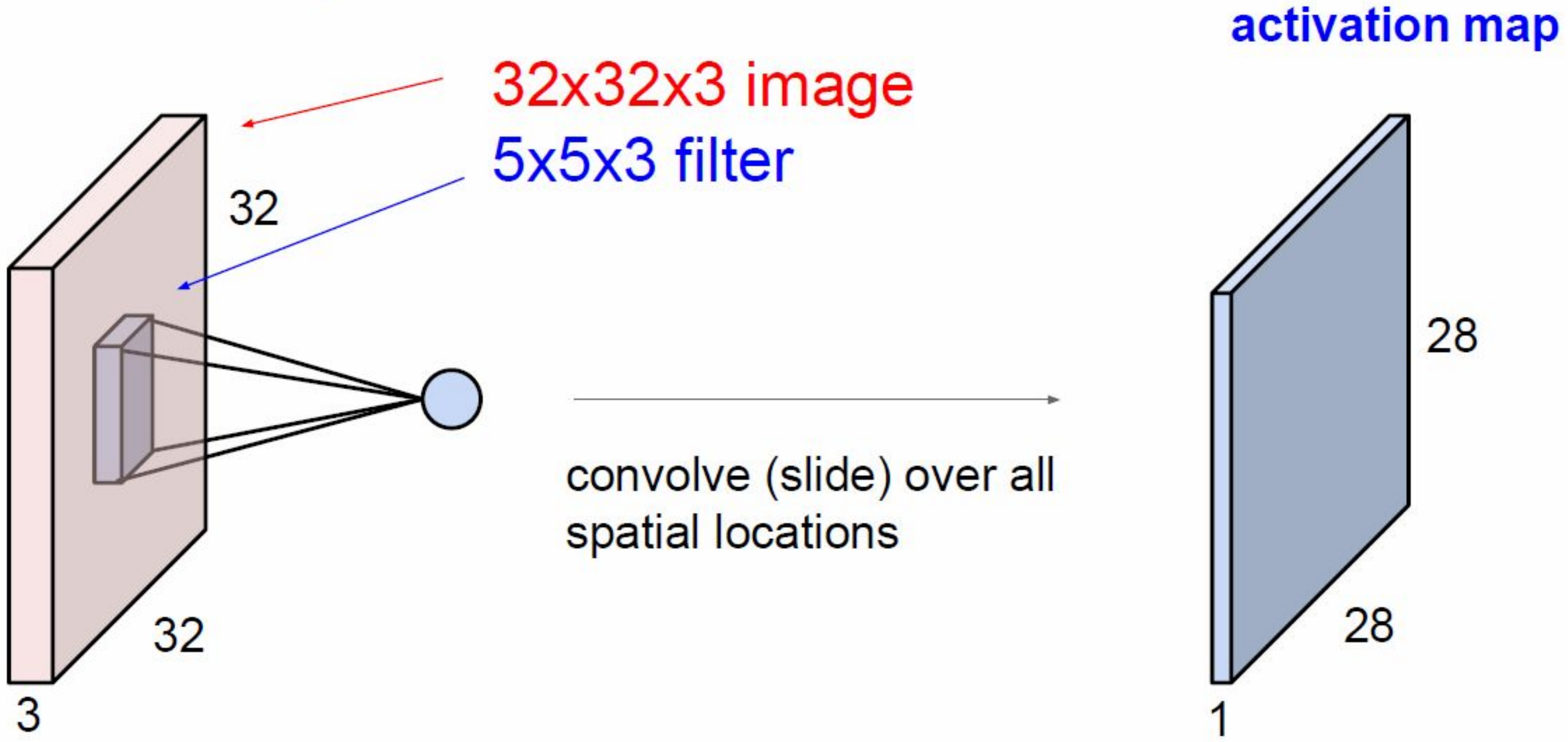
Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"



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)

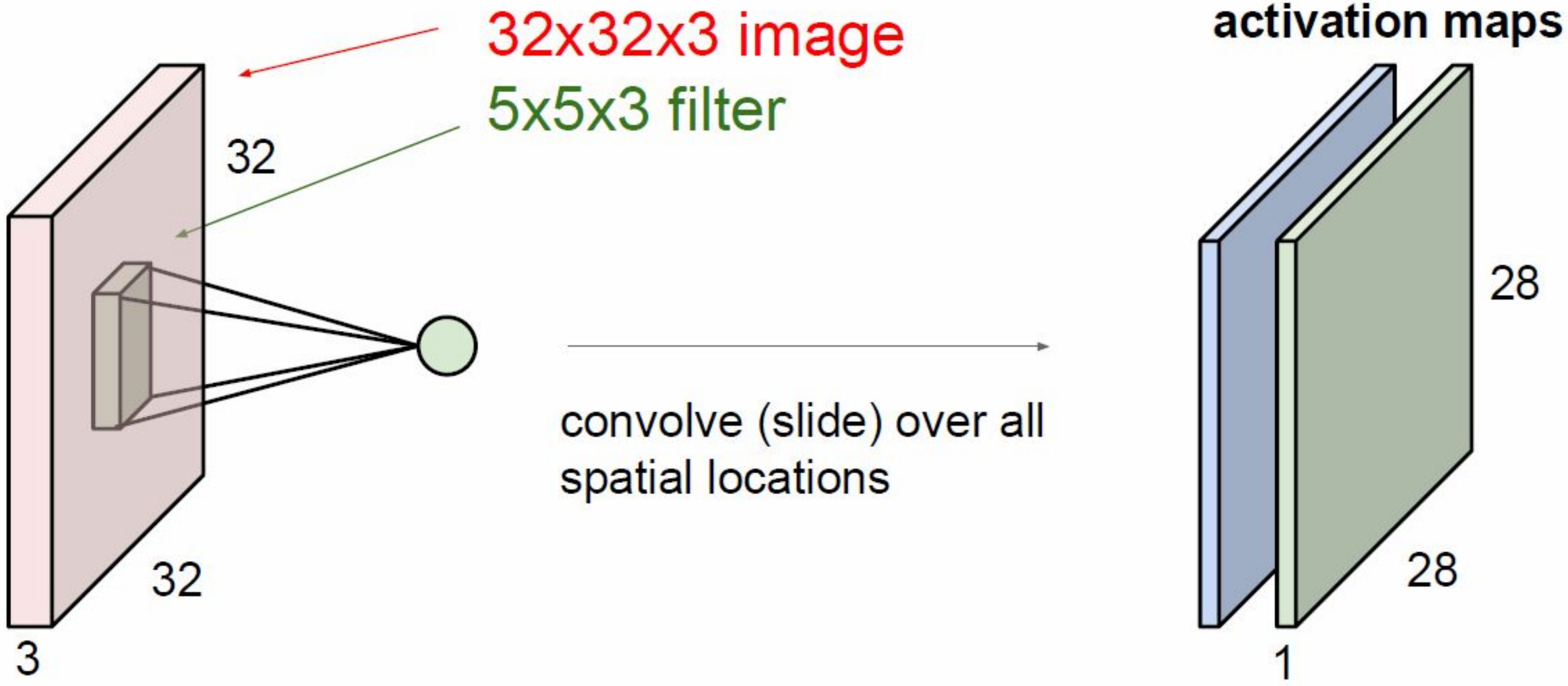
$$w^T x + b$$

Convolution Layer

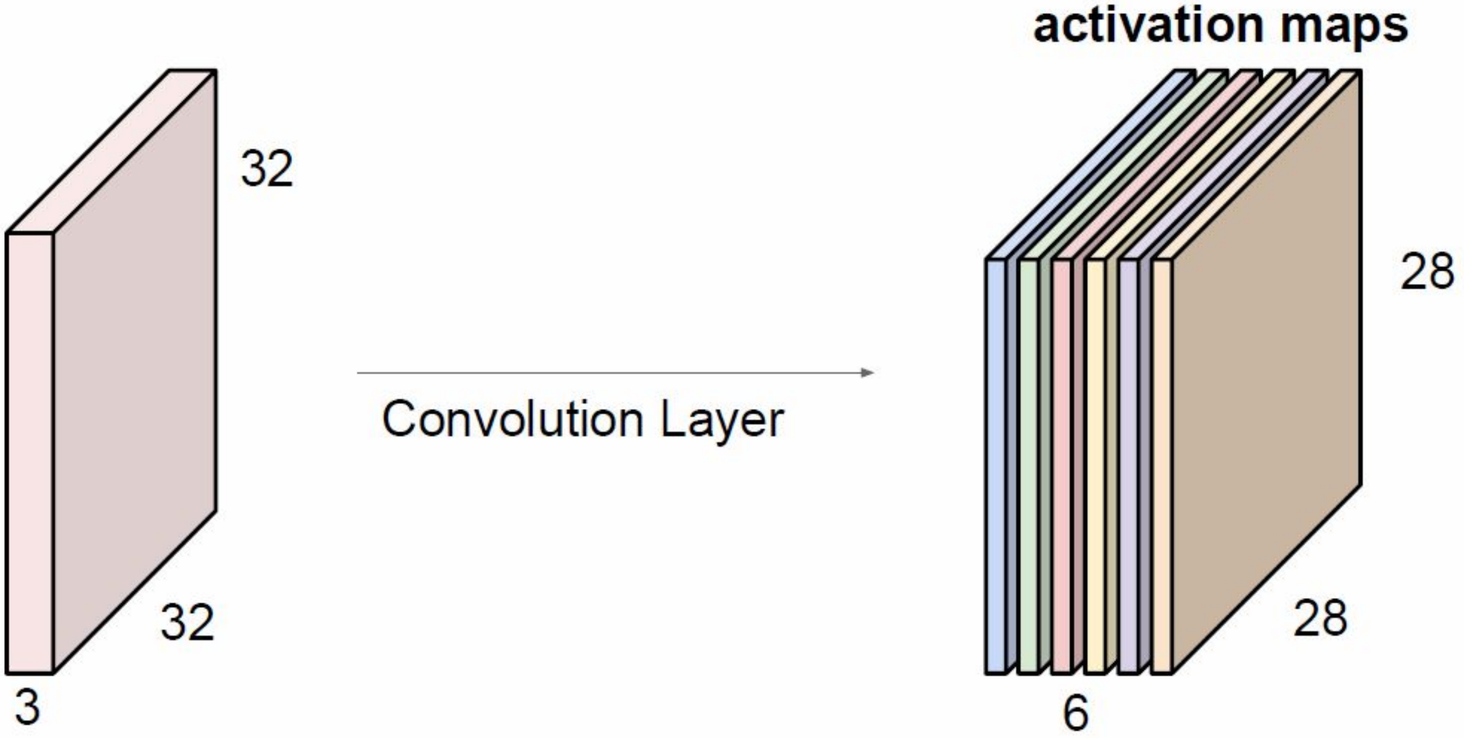


Convolution Layer

consider a second, green filter

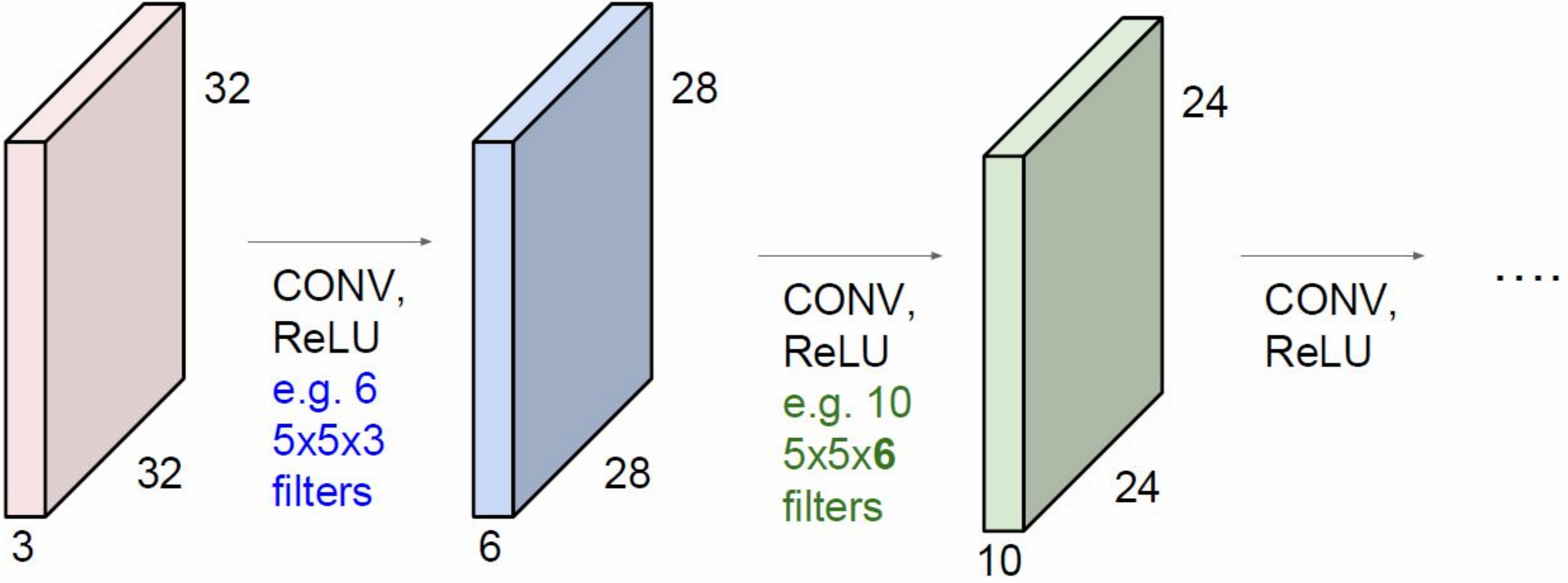


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

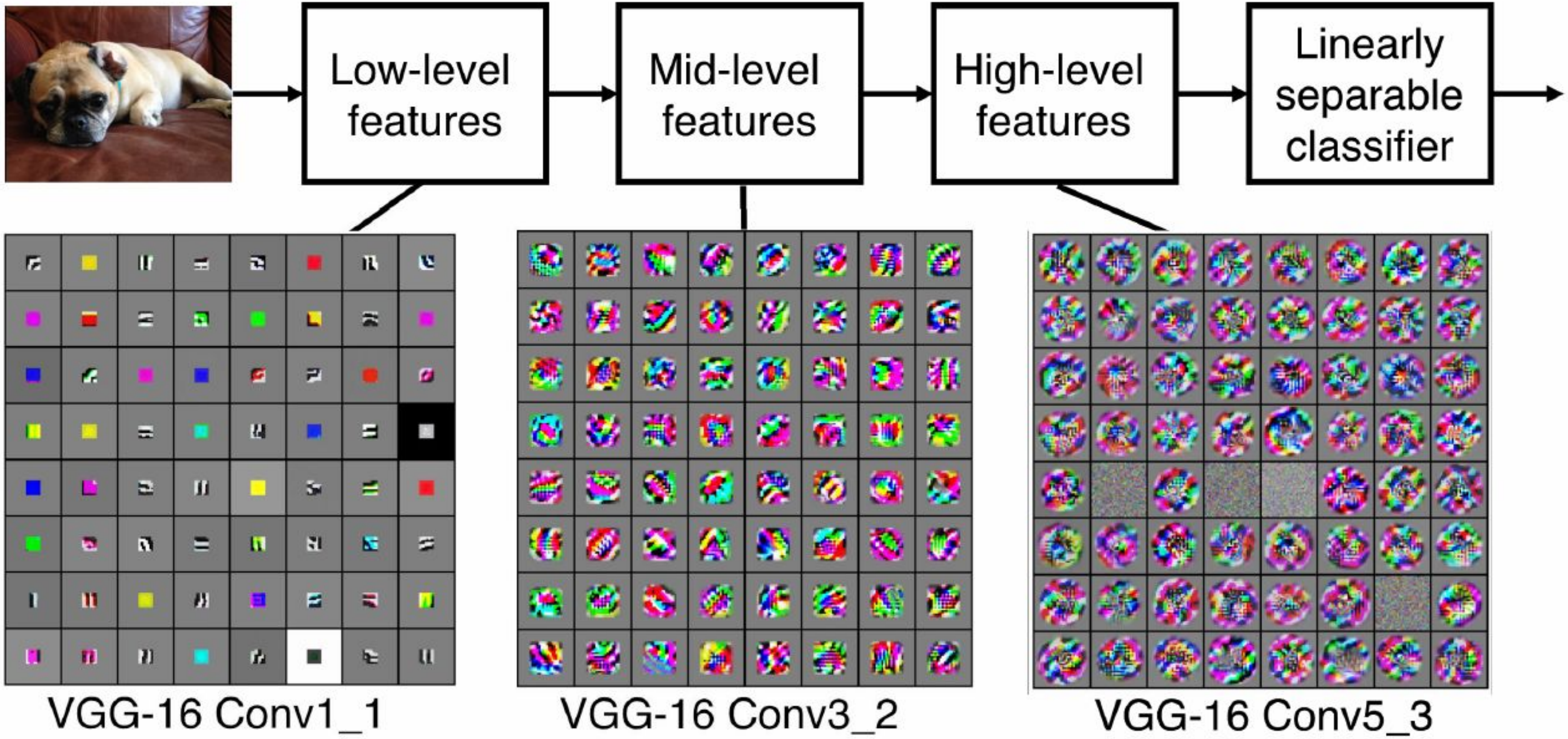
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



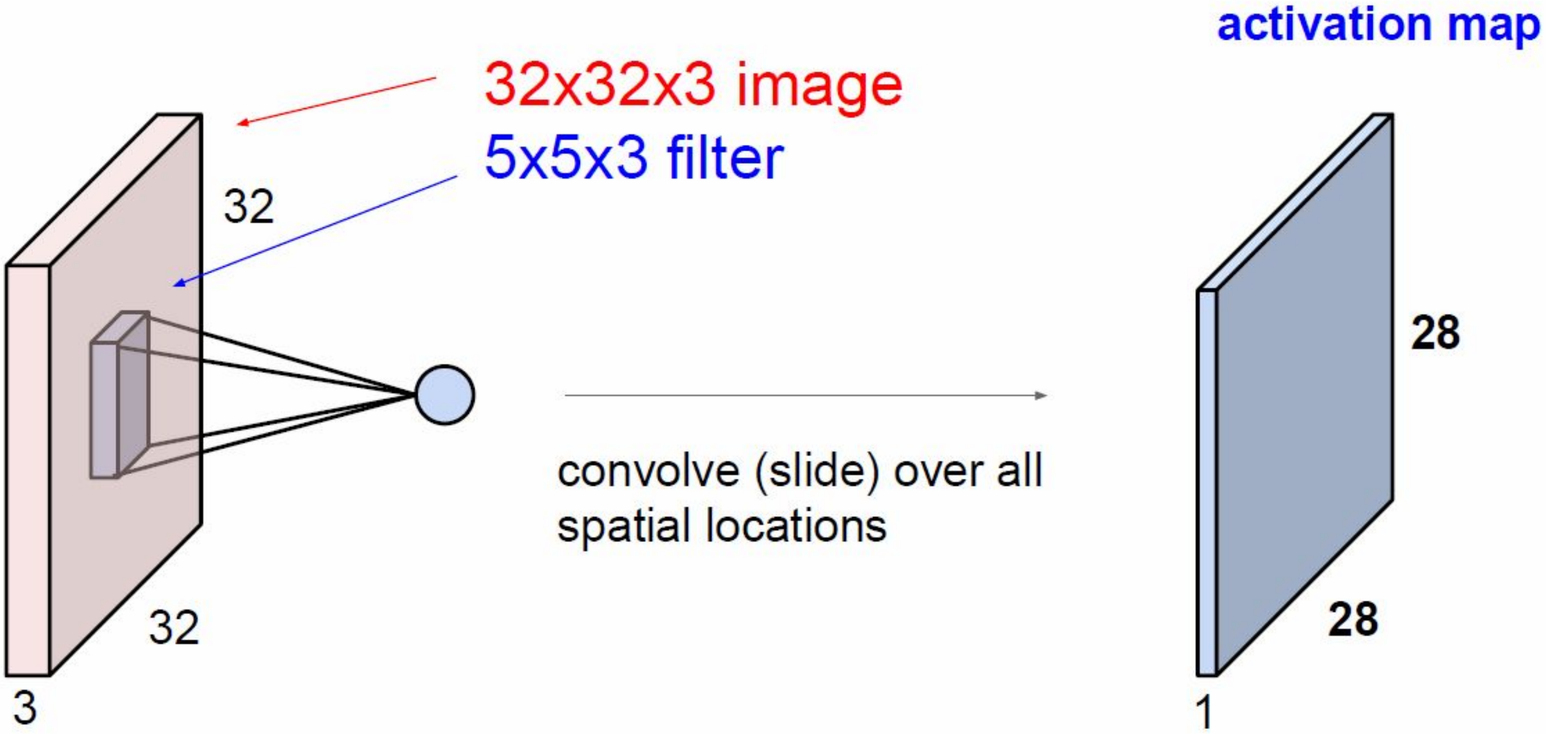
Preview

[Zeiler and Fergus 2013]

Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



A closer look at spatial dimensions:



Convolution Layer - Stride=1, Pad=0, Channel=1

Input An image with height n_h and width n_w, represented by an $n_h \times n_w$ matrix X, e.g., $n_h = n_w = 32$.

Filter Represented by an $F \times F$ matrix W, e.g., $F = 3$.

Output $Y = X * W$, where

$$Y_{ij} = \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} W_{m,n} X_{i+m,j+n}$$

for $i = 0, \cdots, n_h - F + 1$ and $j = 0, \cdots, n_w - F + 1$.

For example, if the input is 32×32 and $F = 3$, then the output will be 30×30.

Convolution Layer - Stride=1, Pad=0, Channel=3

Input An image with height n_h and width n_w with n_c channels, represented by an $n_h \times n_w \times n_c$ tensor X, e.g., $n_h = n_w = 32, n_c = 3$.

Filter Represented by an $F \times F \times n_c$ tensor W, e.g., $F = 3$.

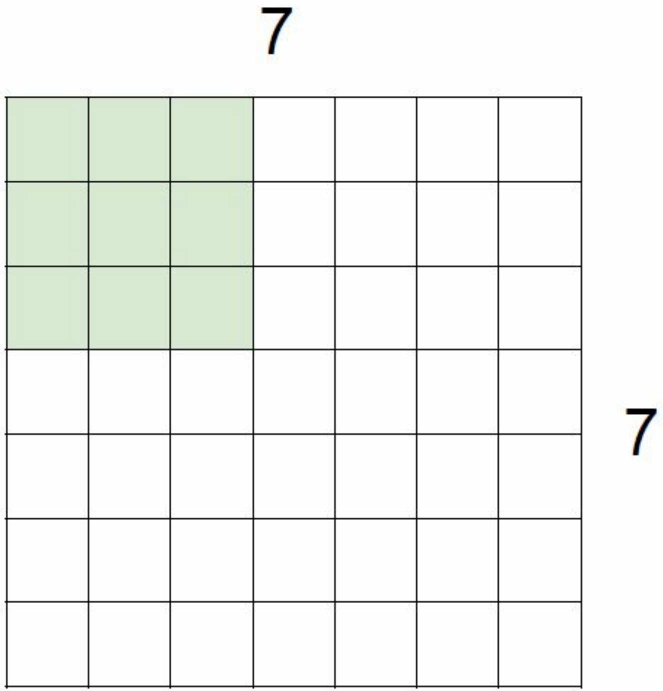
Output $Y = X * W$, where

$$Y_{ij} = \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} \sum_{p=0}^{n_c} W_{m,n,p} X_{i+m,j+n,p}$$

for $i = 0, \cdots, n_h - F + 1$ and $j = 0, \cdots, n_w - F + 1$.

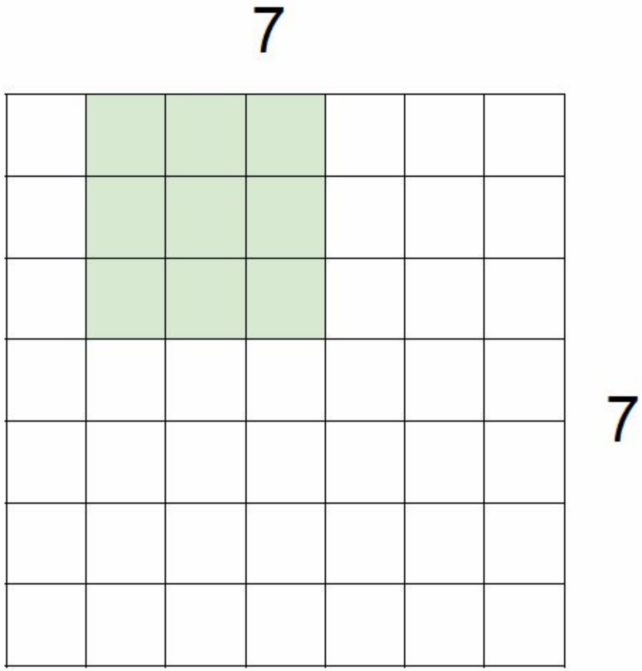
For example, if the input is $32 \times 32 \times 3$ and $F = 3$, then the output will be 30×30.

A closer look at spatial dimensions:



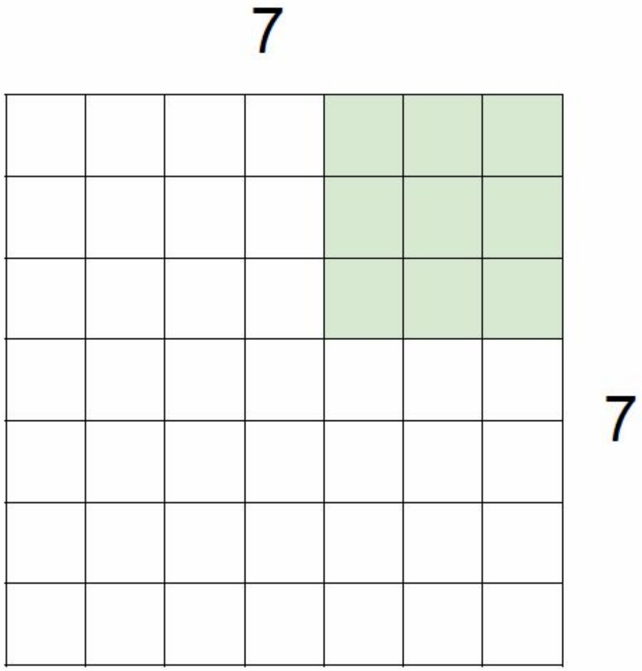
7x7 input (spatially)
assume 3x3 filter

A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

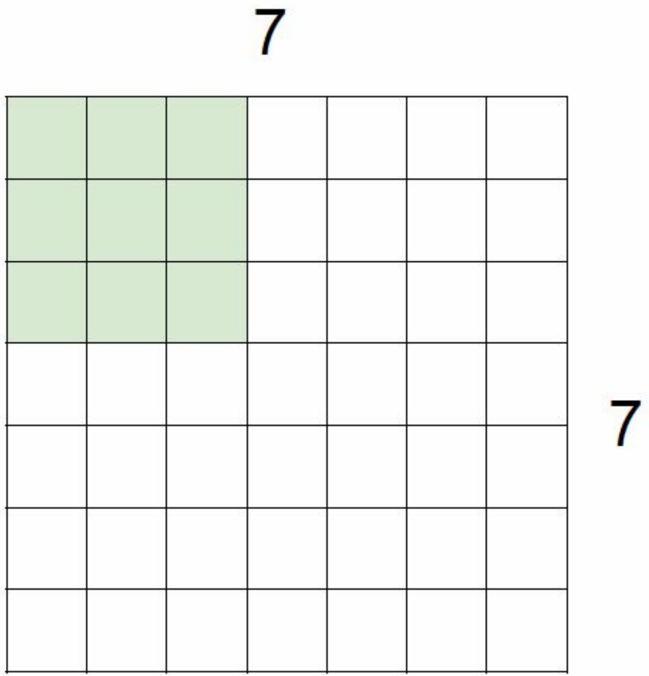
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

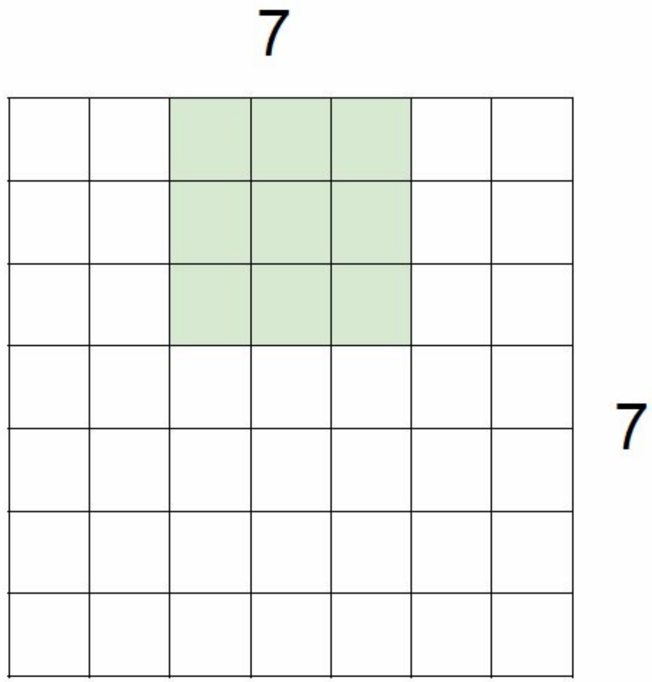
=> 5x5 output

A closer look at spatial dimensions:



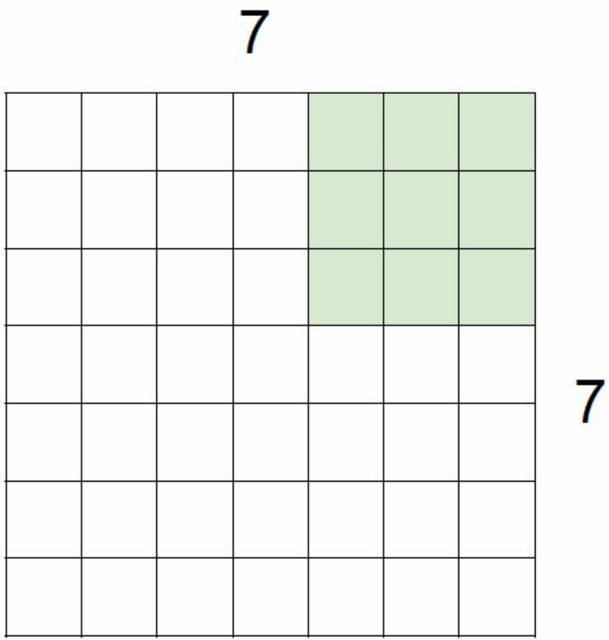
7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:

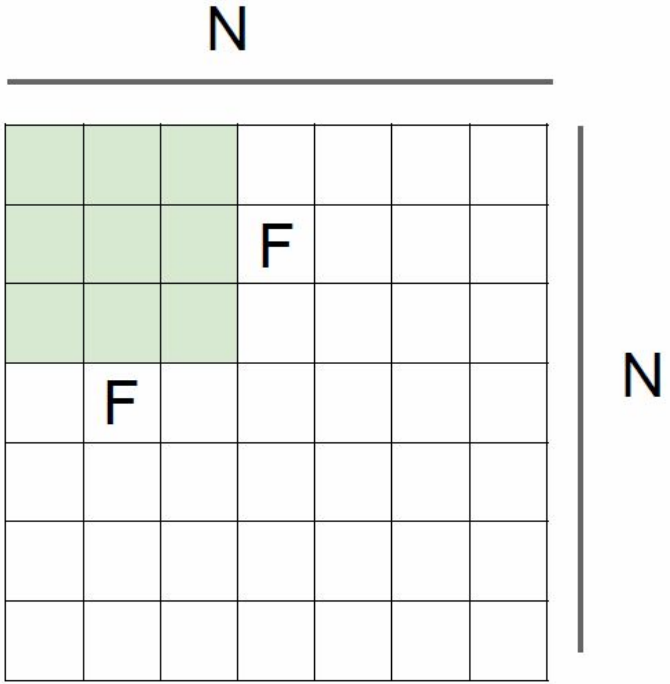


7x7 input (spatially)
assume 3x3 filter
applied **with stride 2**

A closer look at spatial dimensions:



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



Output size:

(N - F) / stride + 1

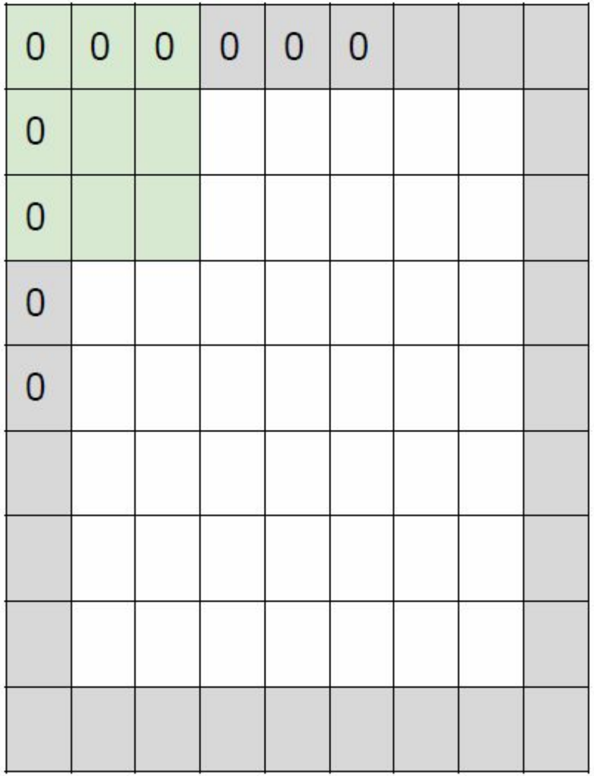
e.g. N = 7, F = 3:

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?

7x7 output!

Convolution Layer - Stride=1, Pad=1, Channel=1

Input An image with height n_h and width n_w, represented by an $n_h \times n_w$ matrix X, e.g., $n_h = n_w = 32$. Zero pad the border so that X will be 34×34.

Filter Represented by an $F \times F$ matrix W, e.g., $F = 3$.

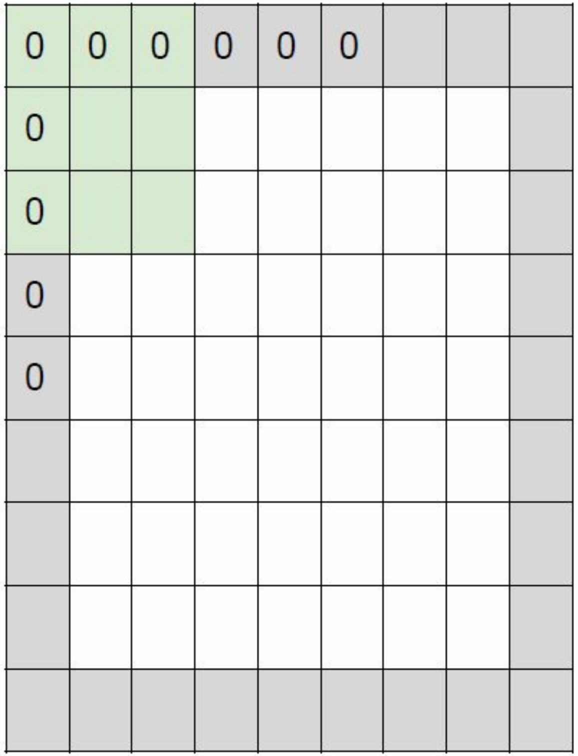
Output $Y = X * W$, where

$$Y_{ij} = \sum_{m=0}^{F-1} \sum_{n=0}^{F-1} W_{m,n} X_{i+m,j+n}$$

for $i = 0, \cdots, n_h$ and $j = 0, \cdots, n_w$.

For example, if the input is 32×32 and $F = 3$, then the output will be 32×32.

In practice: Common to zero pad the border

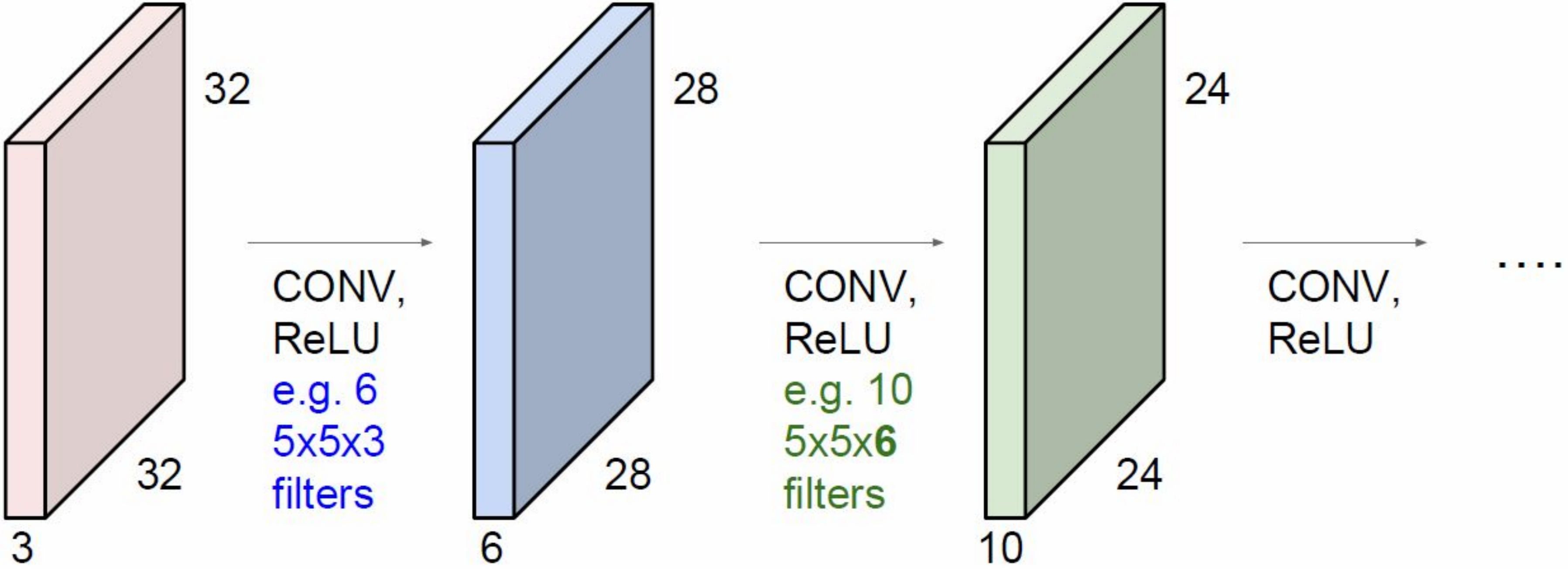


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (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

Remember back to…

E.g. 32x32 input convolved repeatedly with 5x5 filters shrinks volumes spatially!
(32 -> 28 -> 24 ...). Shrinking too fast is not good, doesn't work well.



Stride + Padding

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size: ?

(32+2*2-5)/1+1 = 32 spatially, so
32x32x10

Stride + Padding

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

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.

Common settings:

Summary. To summarize, the Conv Layer:

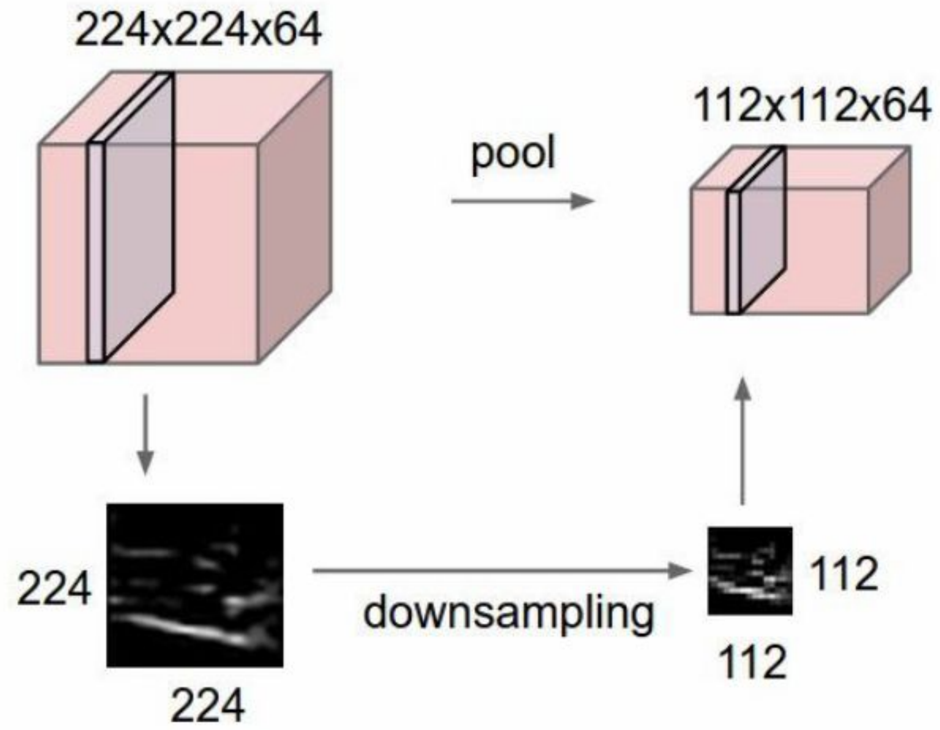
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters K,
  - their spatial extent F,
  - the stride S,
  - the amount of zero padding P.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  - $H_2 = (H_1 - F + 2P)/S + 1$ (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)

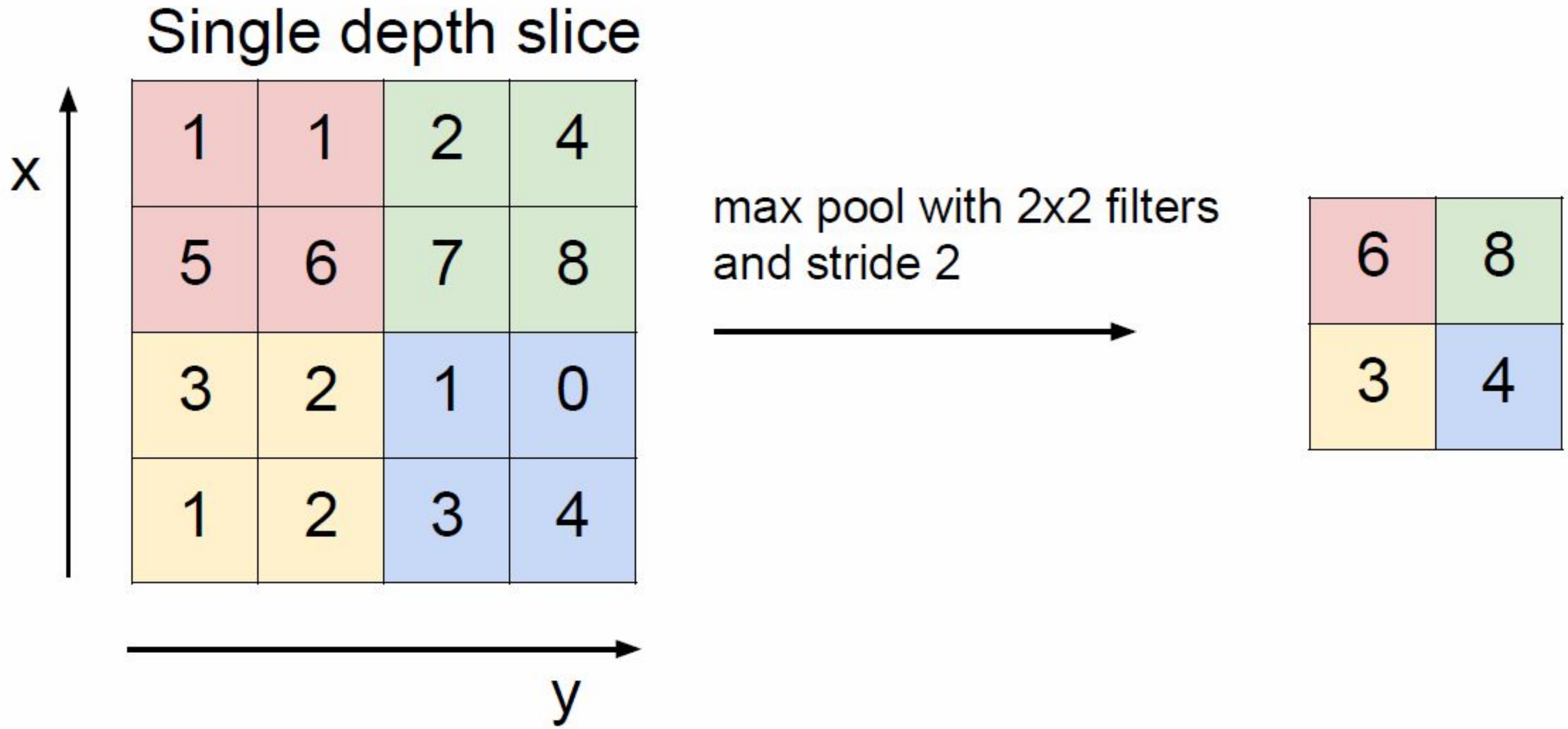
- F = 3, S = 1, P = 1
- F = 5, S = 1, P = 2
- F = 5, S = 2, P = ? (whatever fits)
- F = 1, S = 1, P = 0

Pooling layer

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



MAX POOLING


Single depth slice
x
y
1
1
2
4
5
6
7
8
3
2
1
0
1
2
3
4
max pool with 2x2 filters
and stride 2
6
8
3
4

Gradient of Max Pooling - Stride=2

Forward $Y = \mathrm{POOL(X)}$, where $X \in \mathbb{R}^{n_h \times n_w}$

$$Y_{ij} = \max\{X_{2i,2j}, X_{2i+1,2j}, X_{2i,2j+1}, X_{2i+1,2j+1}\}$$

for $i = 0, \cdots, n_h/2$ and $j = 0, \cdots, n_w/2$.

Backward

$$\left(\frac{\partial \mathcal{L}}{\partial X}\right)_{2i+m,2j+n} = I(X_{2i+m,2j+n} = Y_{ij}) \left(\frac{\partial \mathcal{L}}{\partial Y}\right)_{ij}$$

for $m = 0, 1$, $n = 0, 1$, $i = 0, \cdots, n_h/2$ and $j = 0, \cdots, n_w/2$.

- 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

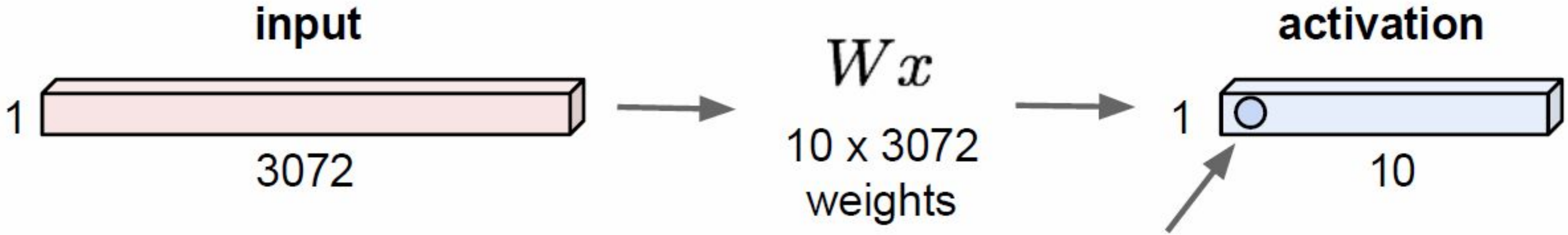
Common settings:

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

Reminder: Fully Connected Layer

Each neuron looks at the full input volume

32x32x3 image -> stretch to 3072 x 1



1 number:
the result of taking a dot product between a row of W and the input (a 3072-dimensional dot product)

Example ConvNet for CIFAR-10 classification

Architecture [INPUT - CONV - RELU - POOL - FC]:

INPUT [32x32x3] width 32, height 32, and with three color channels R,G,B.

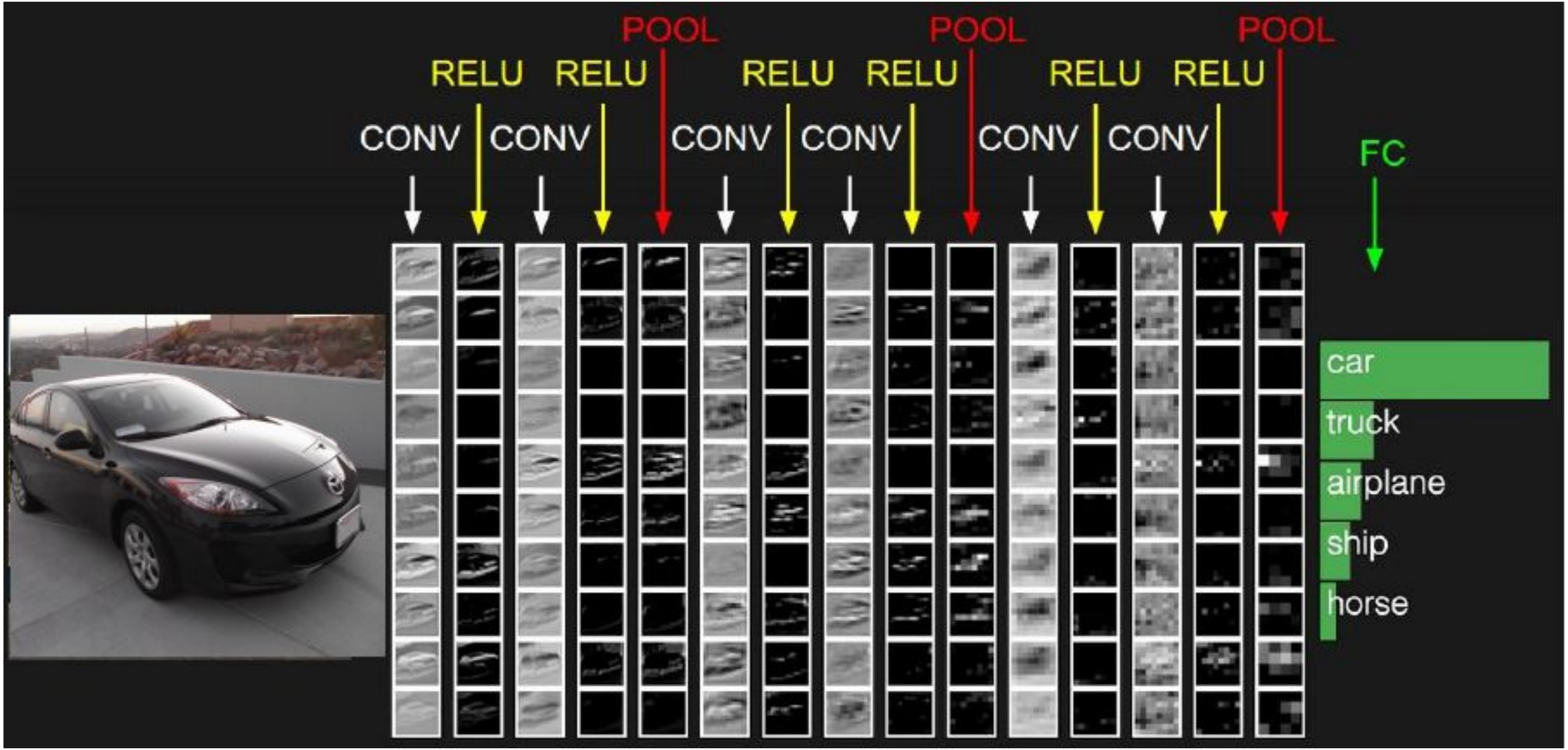
- CONV layer will compute the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32x32x12] if we decided to use 12 filters.

- RELU layer will apply an elementwise activation function, such as the max(0,x). This leaves the size of the volume unchanged ([32x32x12]).

- POOL layer will perform a downsampling operation along the spatial dimensions (width, height), resulting in volume such as [16x16x12].

- FC (i.e. fully-connected) layer will compute the class scores, resulting in volume of size [1x1x10], where each of the 10 numbers correspond to a class score, such as among the 10 categories of CIFAR-10. As with ordinary Neural Networks and as the name implies, each neuron in this layer will be connected to all the numbers in the previous volume.

Put layers together



Summary

- ConvNets stack CONV,POOL,FC layers
- Trend towards smaller filters and deeper architectures
- Trend towards getting rid of POOL/FC layers (just CONV)
- Typical architectures look like
  - **[(CONV-RELU)*N-POOL?]*M-(FC-RELU)*K,SOFTMAX**
  - where N is usually up to ~5, M is large, 0 <= K <= 2.
  - - but recent advances such as ResNet/GoogLeNet challenge this paradigm