# Lecture 17:
# Convolutional Neural Networks (CNNs)

COS 429: Computer Vision

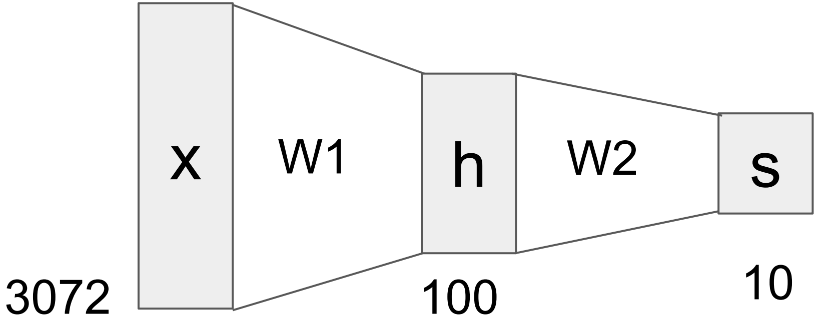
PRINCETON UNIVERSITY

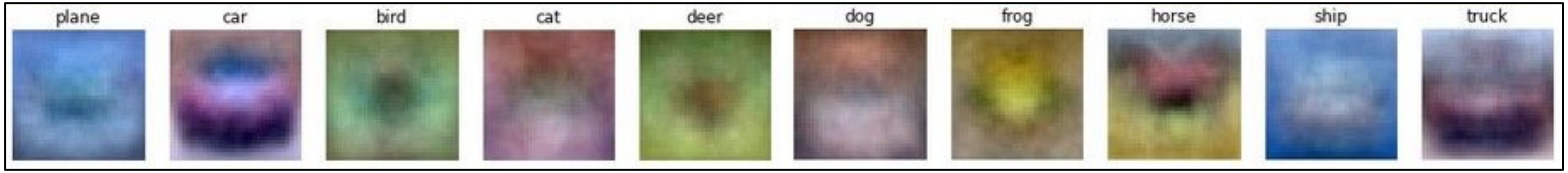
Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Last time: Neural Networks

Linear score function: $f = Wx$

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



# Next: Convolutional Neural Networks

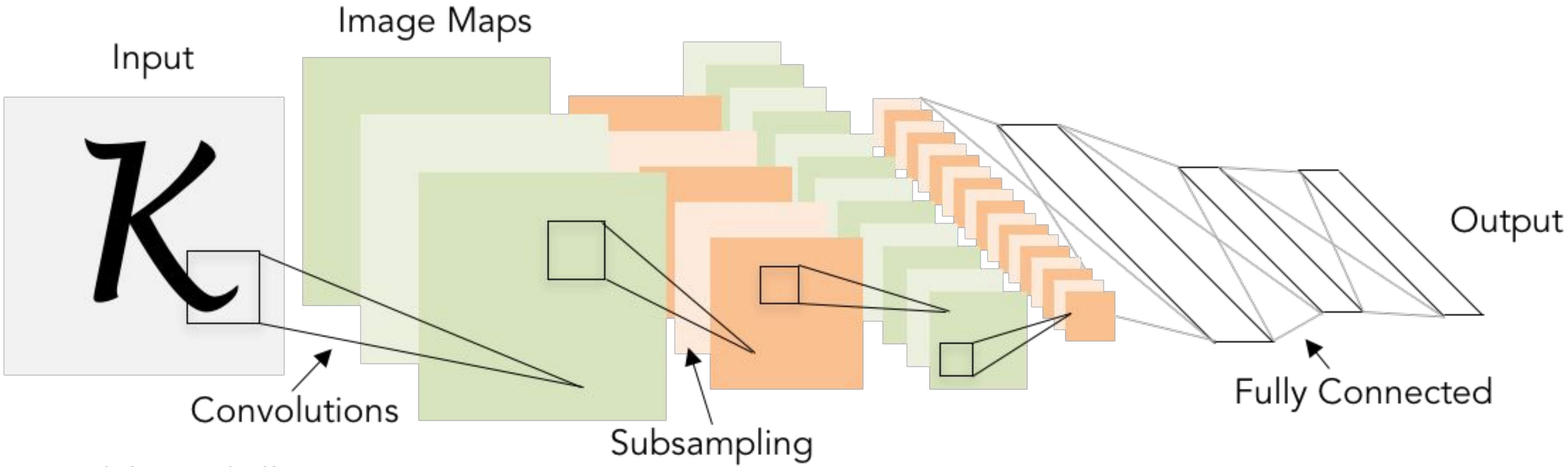



Illustration of LeCun et al. 1998 from CS231n 2017 Lecture 1

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A bit of history...

The **Mark I Perceptron** machine was the first implementation of the perceptron algorithm.

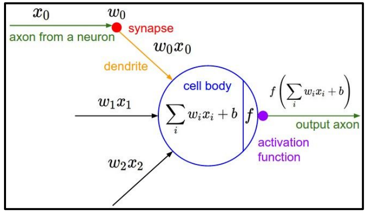
The machine was connected to a camera that used 20×20 cadmium sulfide photocells to produce a 400-pixel image.

recognized
letters of the alphabet

$$f(x) = \begin{cases} 1 & \text{if } w \cdot x + b > 0 \\ 0 & \text{otherwise} \end{cases}$$

update rule:

$$w_i(t+1) = w_i(t) + \alpha(d_j - y_j(t))x_{j,i},$$

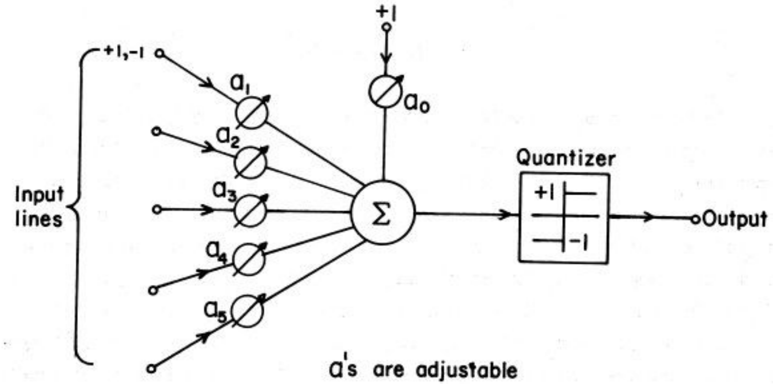


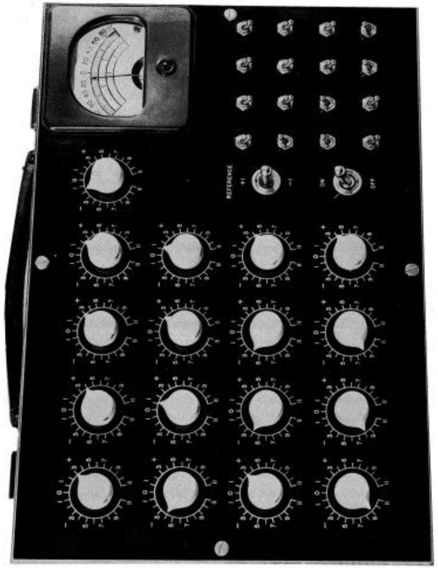

Frank Rosenblatt, ~1957: Perceptron

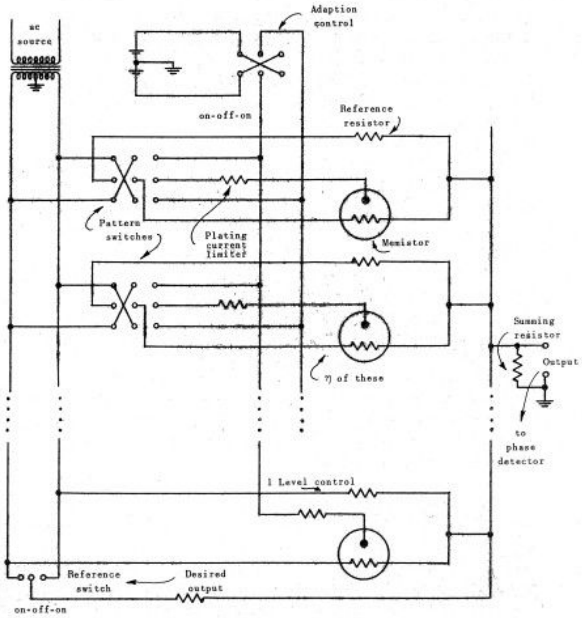



Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A bit of history...






Widrow and Hoff, ~1960: Adaline/Madaline

These figures are reproduced from [Widrow 1960, Stanford Electronics Laboratories Technical Report](#) with permission from [Stanford University Special Collections](#).

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A bit of history...

$$\frac{\partial E_p}{\partial w_{ji}} = \frac{\partial E_p}{\partial o_{pj}} \frac{\partial o_{pj}}{\partial w_{ji}}$$

recognizable math

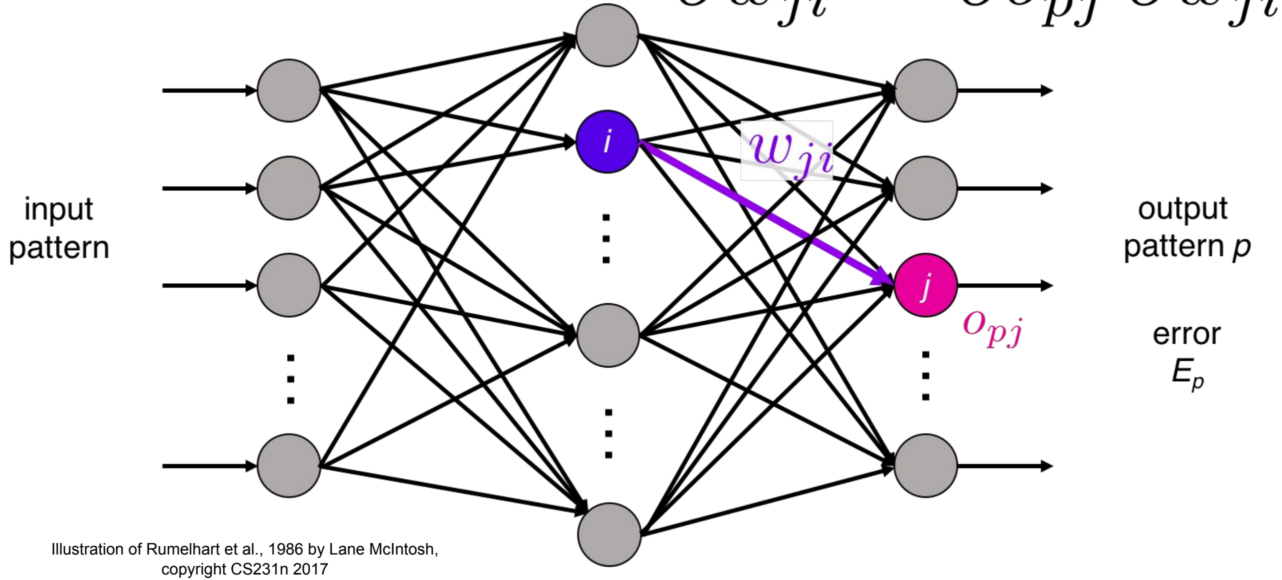



Illustration of Rumelhart et al., 1986 by Lane McIntosh, copyright CS231n 2017

Rumelhart et al., 1986: First time back-propagation became popular

# A bit of history...

[Hinton and Salakhutdinov 2006]

Reinvigorated research in Deep Learning

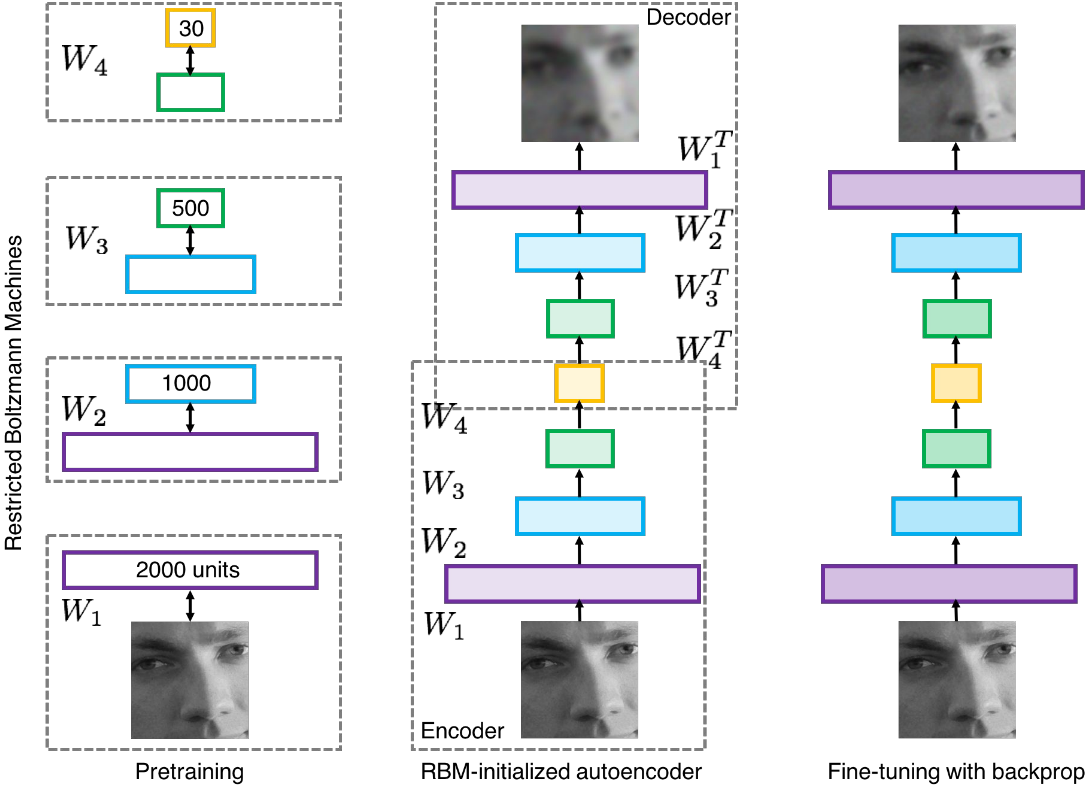



Illustration of Hinton and Salakhutdinov 2006 by Lane McIntosh, copyright CS231n 2017

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# First strong results

***Acoustic Modeling using Deep Belief Networks***
*Abdel-rahman Mohamed, George Dahl, Geoffrey Hinton, 2010*
***Context-Dependent Pre-trained Deep Neural Networks for Large Vocabulary Speech Recognition***
George Dahl, Dong Yu, Li Deng, Alex Acero, 2012

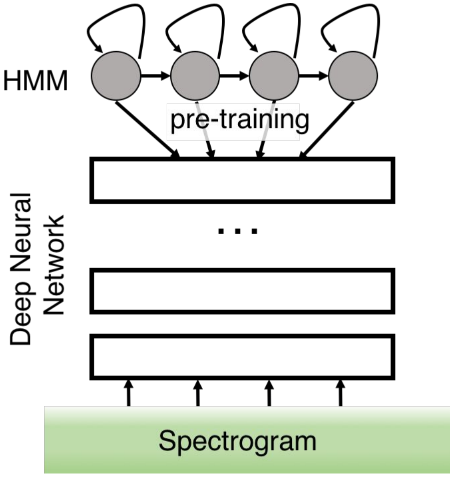
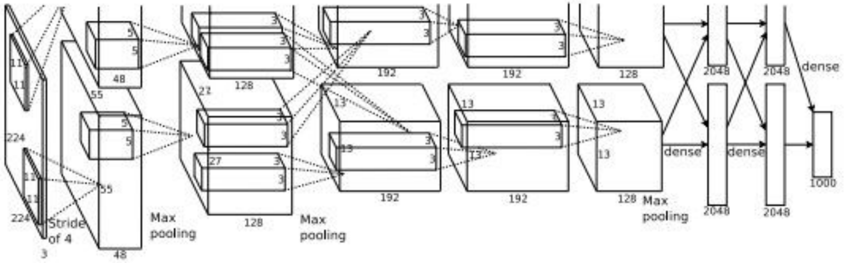
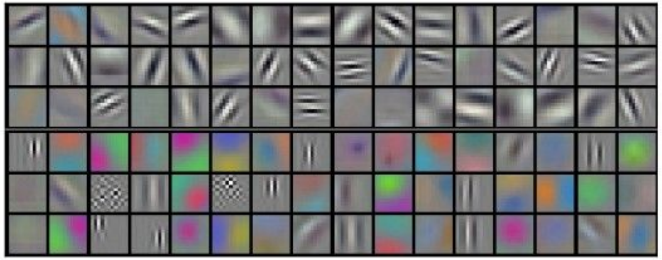


Illustration of Dahl et al. 2012 by Lane McIntosh, copyright CS231n 2017

***Imagenet classification with deep convolutional neural networks***
Alex Krizhevsky, Ilya Sutskever, Geoffrey E Hinton, 2012

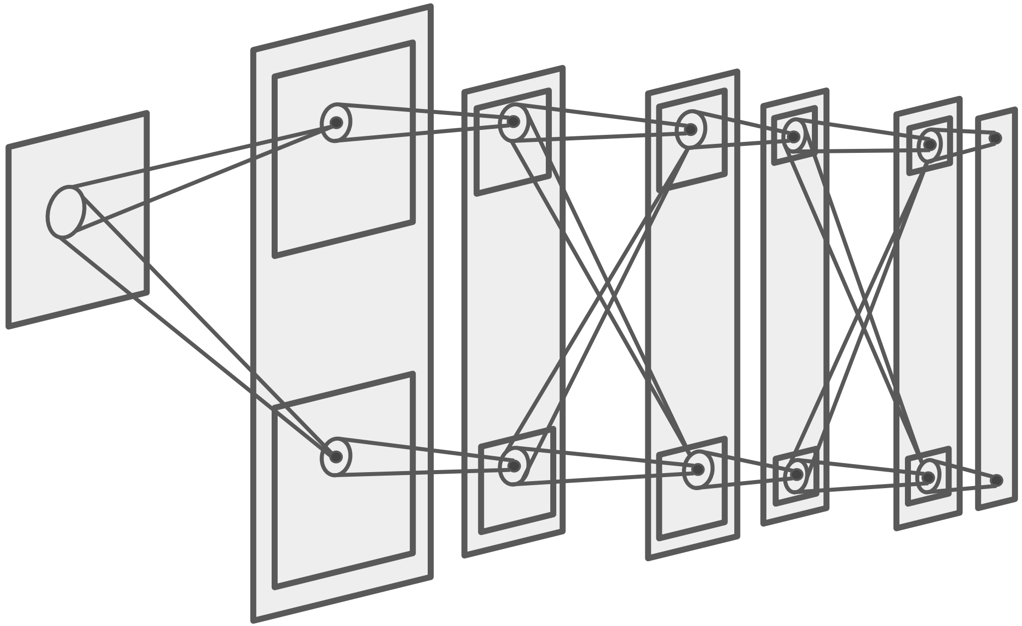


Figures copyright Alex Krizhevsky, Ilya Sutskever, and Geoffrey Hinton, 2012. Reproduced with permission.

# A bit of history:

## **Neocognitron**
*[Fukushima 1980]*



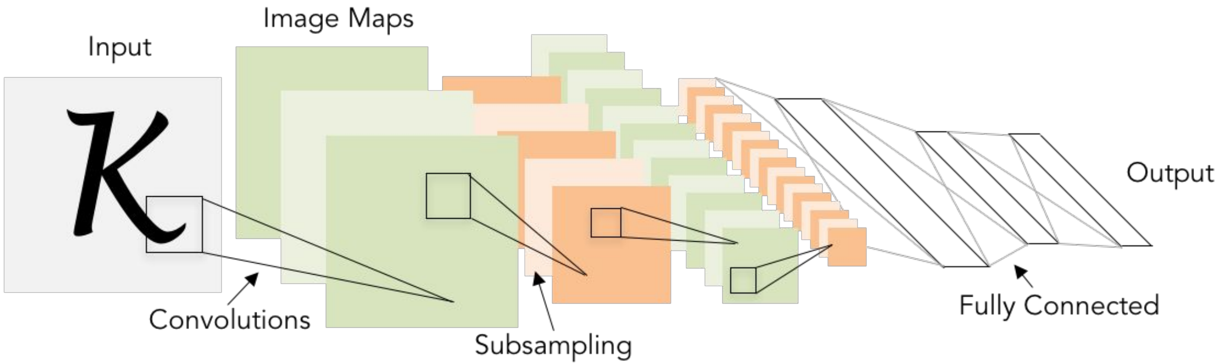
"sandwich" architecture (SCSCSC…)
simple cells: modifiable parameters
complex cells: perform pooling

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A bit of history:

**Gradient-based learning applied to document recognition**

*[LeCun, Bottou, Bengio, Haffner 1998]*



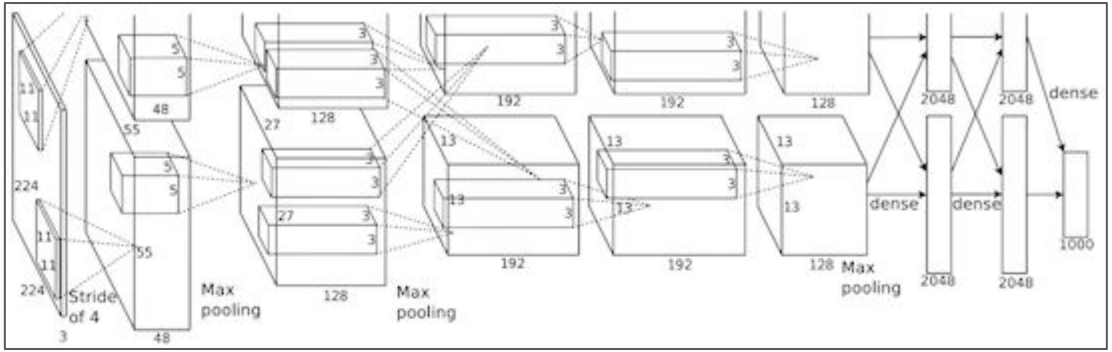

LeNet-5

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A bit of history:

## ImageNet Classification with Deep Convolutional Neural Networks

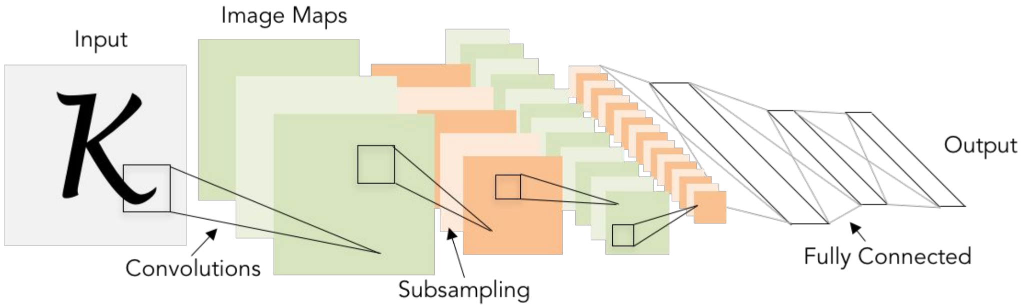
*[Krizhevsky, Sutskever, Hinton, 2012]*



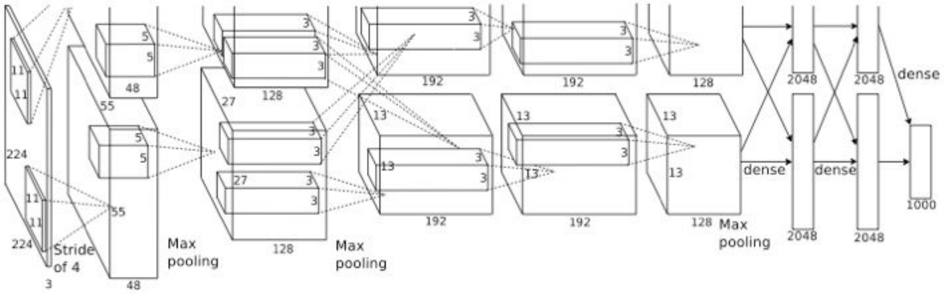



“AlexNet”

# 1998

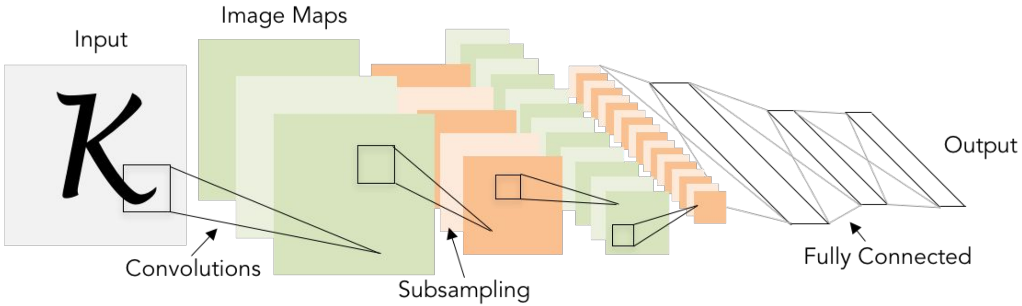



# 2010



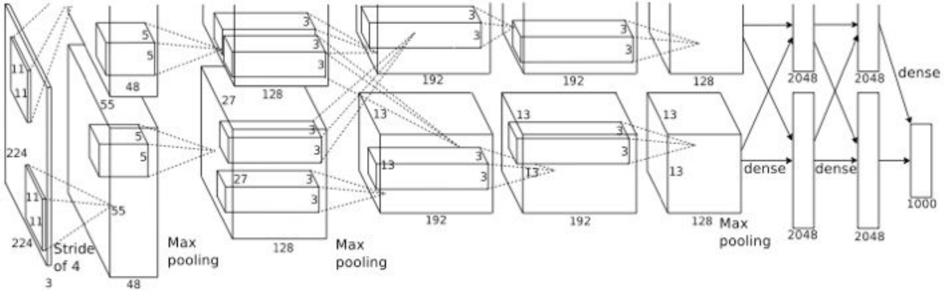

Credit: Fei-Fei Li

## 1998




# transistors $10^6$


## 2010

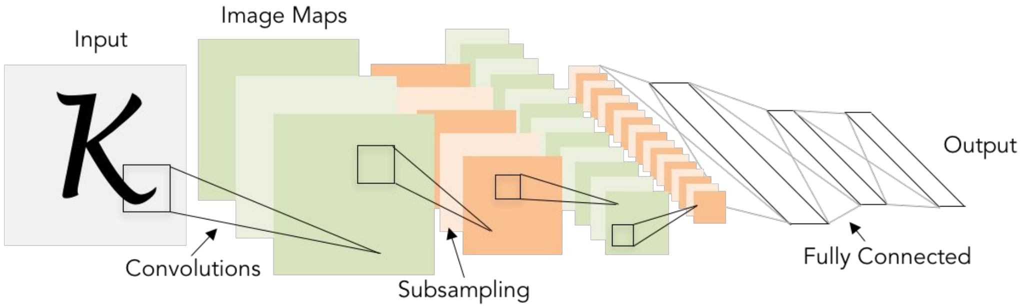



# transistors $10^9$

GPUs


Credit: Fei-Fei Li

## 1998

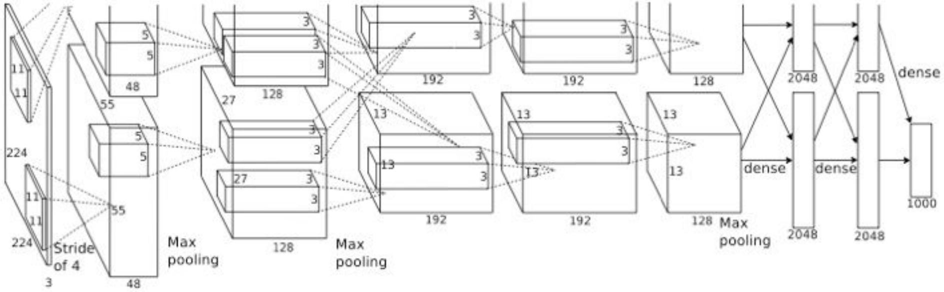



# transistors $10^6$


# pixels used in training $10^7$

NIST

## 2010



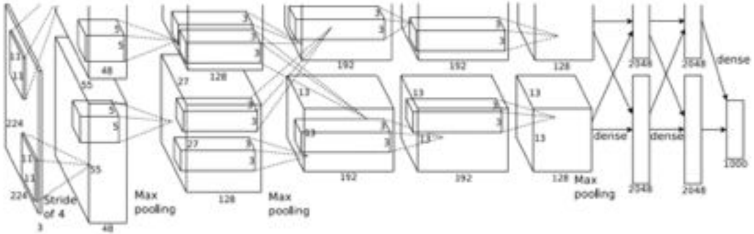

# transistors $10^9$
GPUs

# pixels used in training $10^{14}$

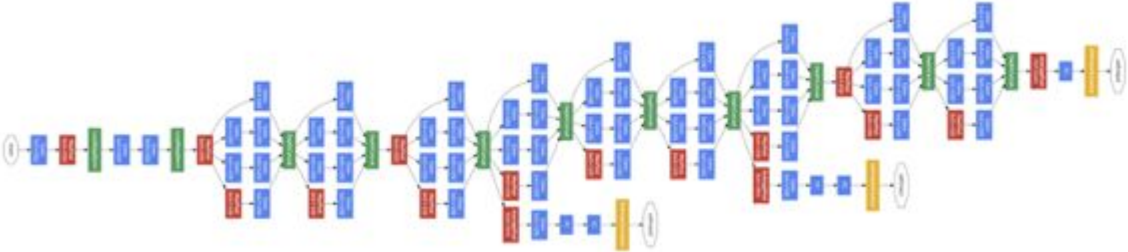
IMAGENET

Credit: Fei-Fei Li

**2012**




**2014**



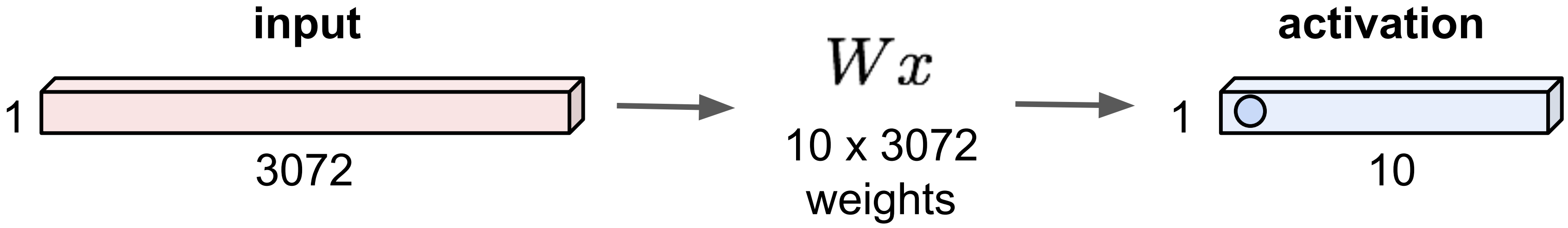

**2015**

# So, what are CNNs?

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Fully Connected Layer

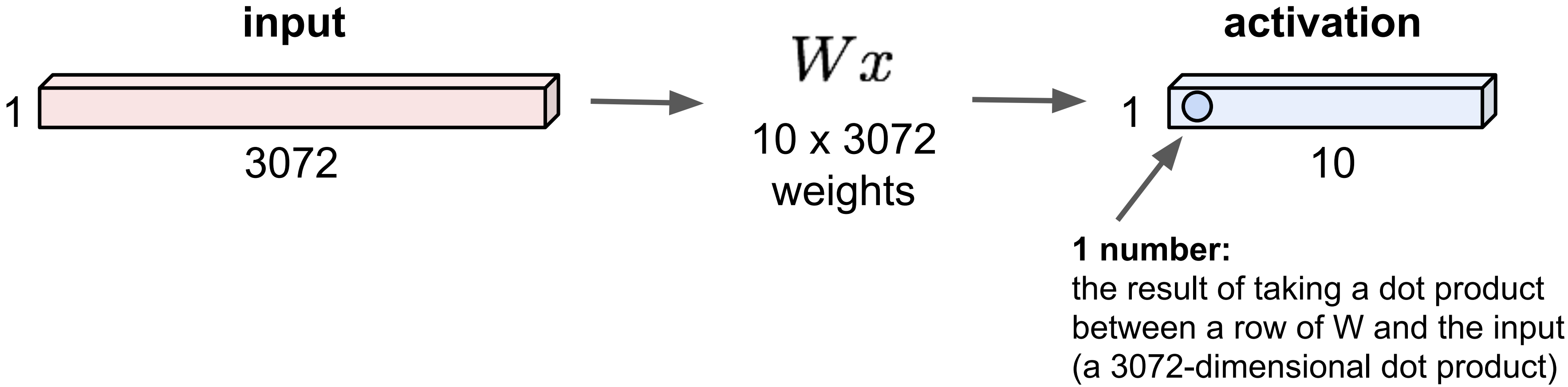
32x32x3 image -> stretch to 3072 x 1




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

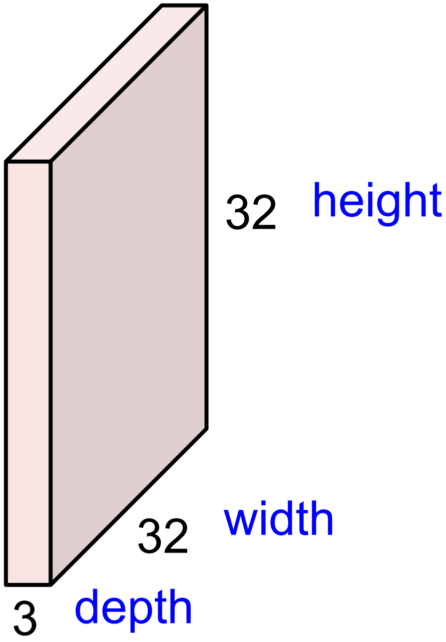
# Fully Connected Layer

32x32x3 image -> stretch to 3072 x 1



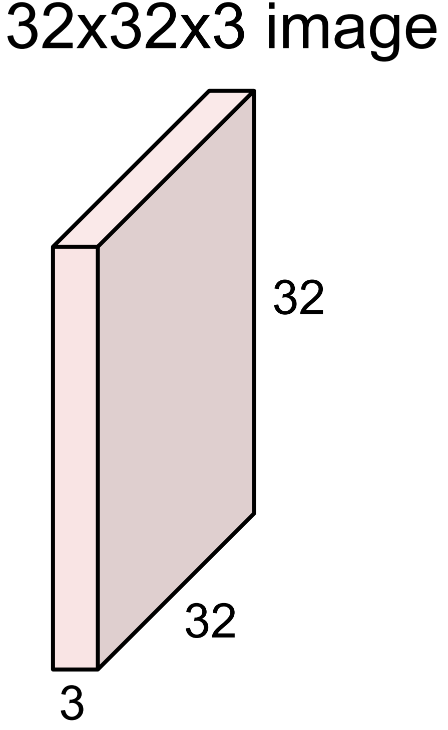
# Convolution Layer

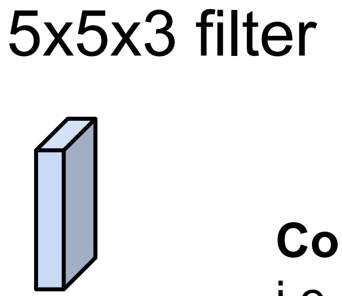
32x32x3 image -> preserve spatial structure




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Convolution Layer



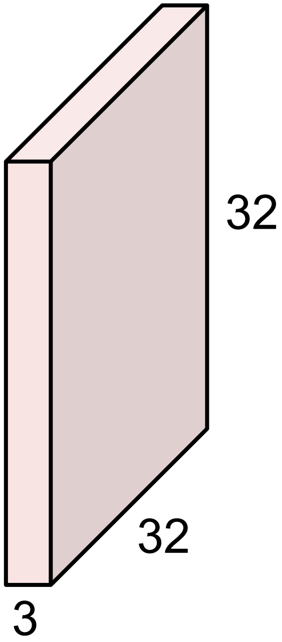

**Convolve** the filter with the image
i.e. “slide over the image spatially,
computing dot products”

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

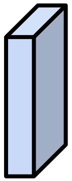
# Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



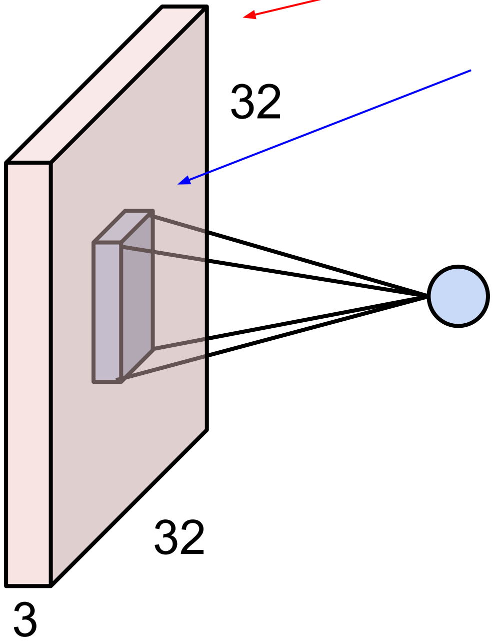

5x5x3 filter



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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Convolution Layer




32x32x3 image

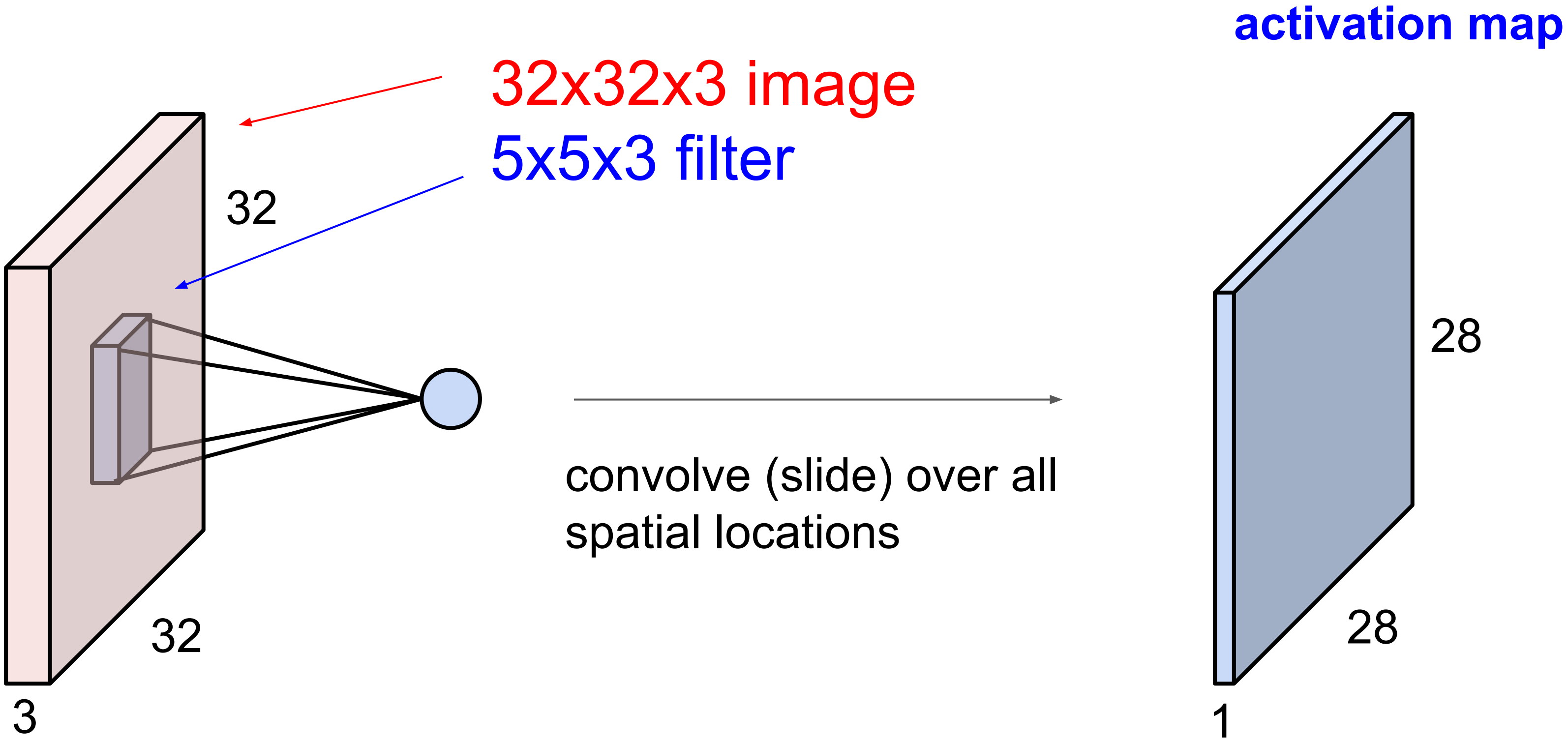
5x5x3 filter $w$

**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)

$$w^T x + b$$

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

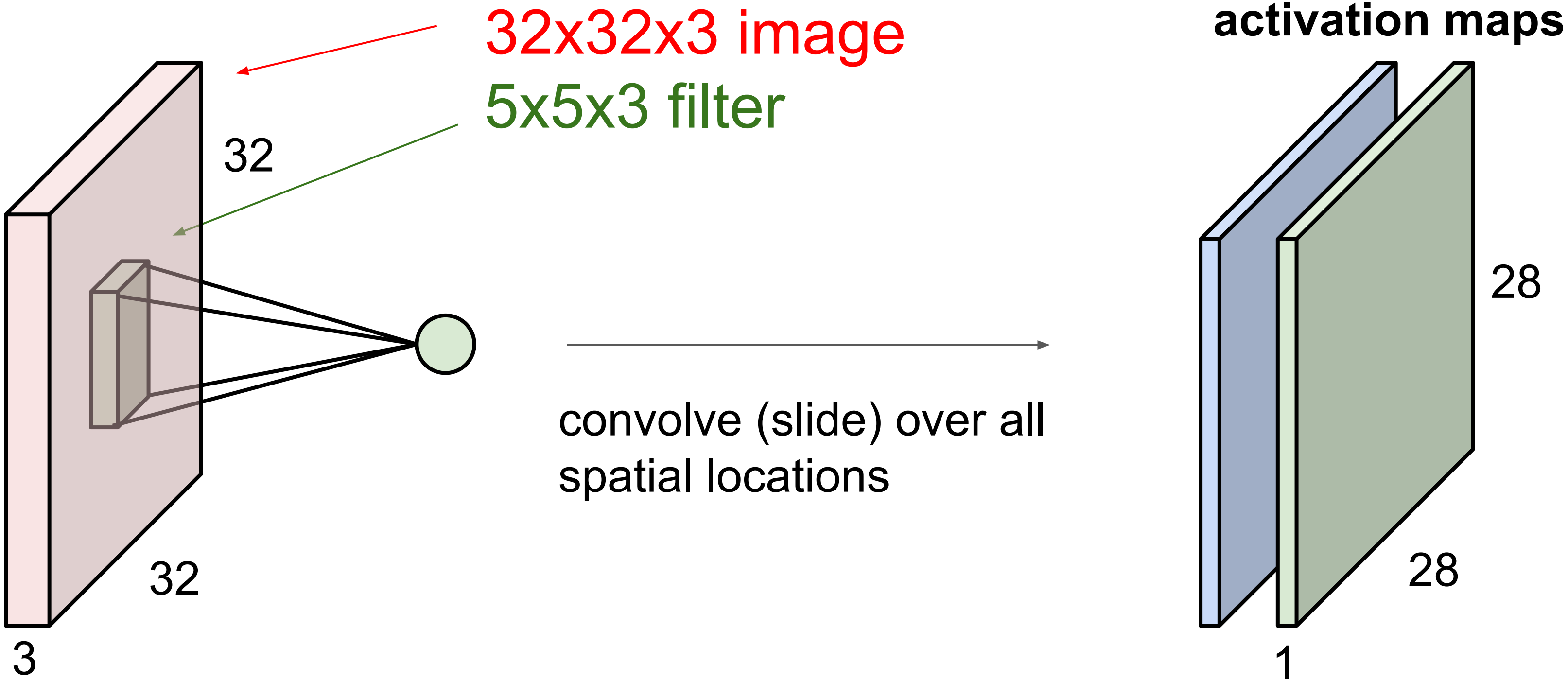
# Convolution Layer



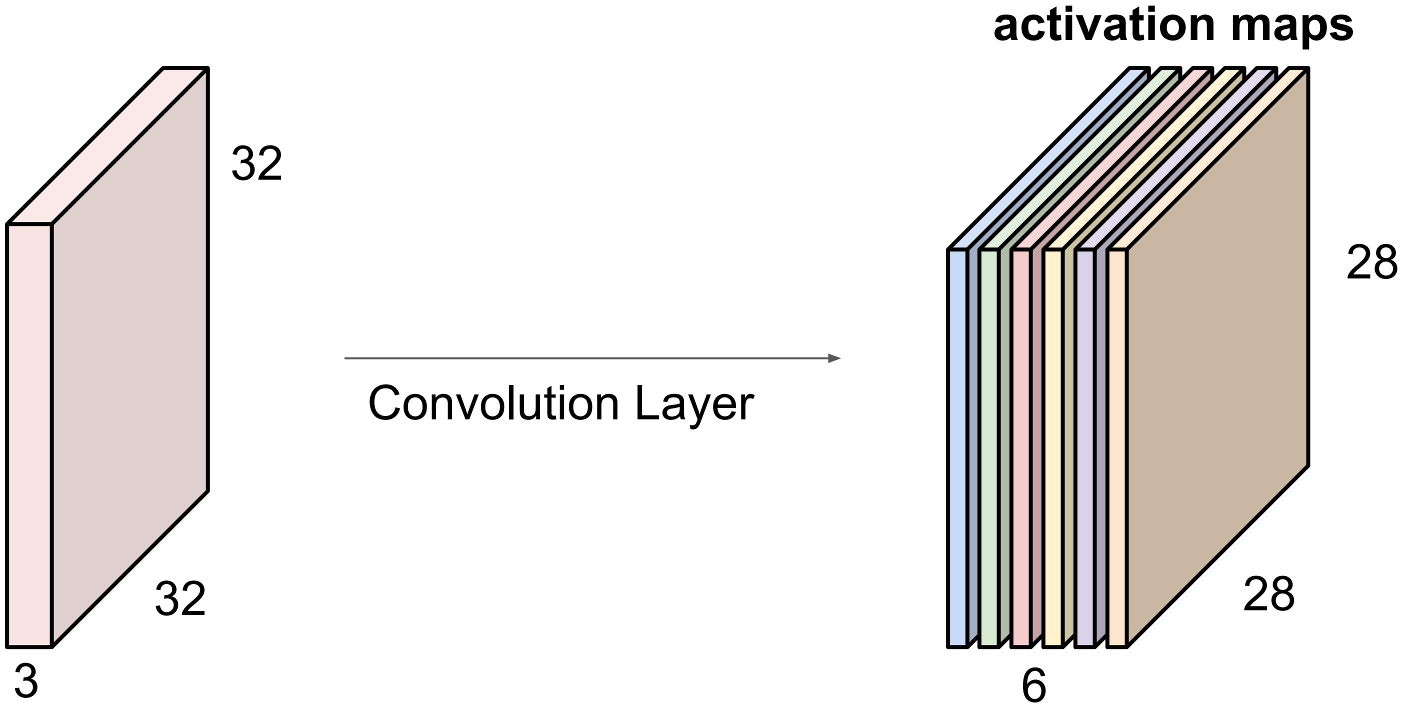

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Convolution Layer

consider a second, green filter



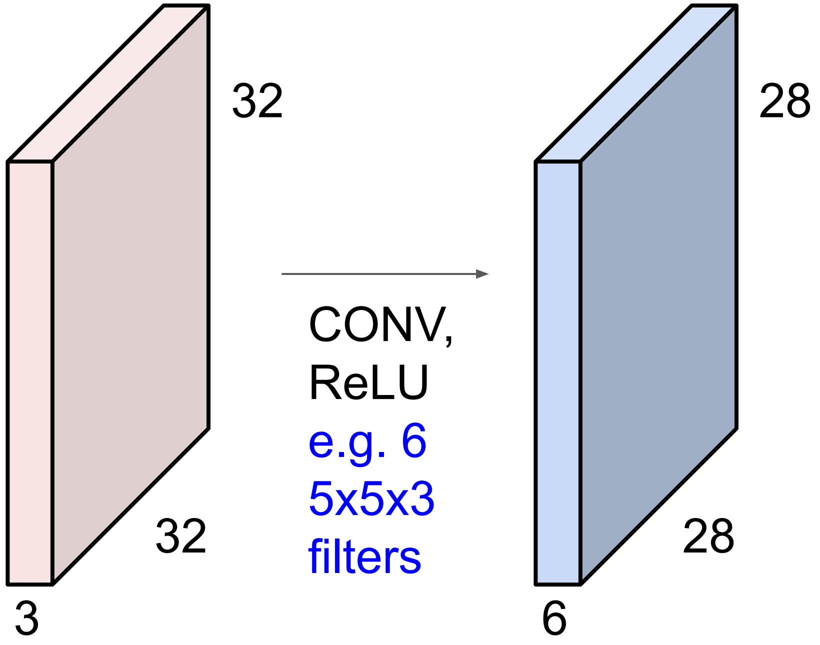
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:




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

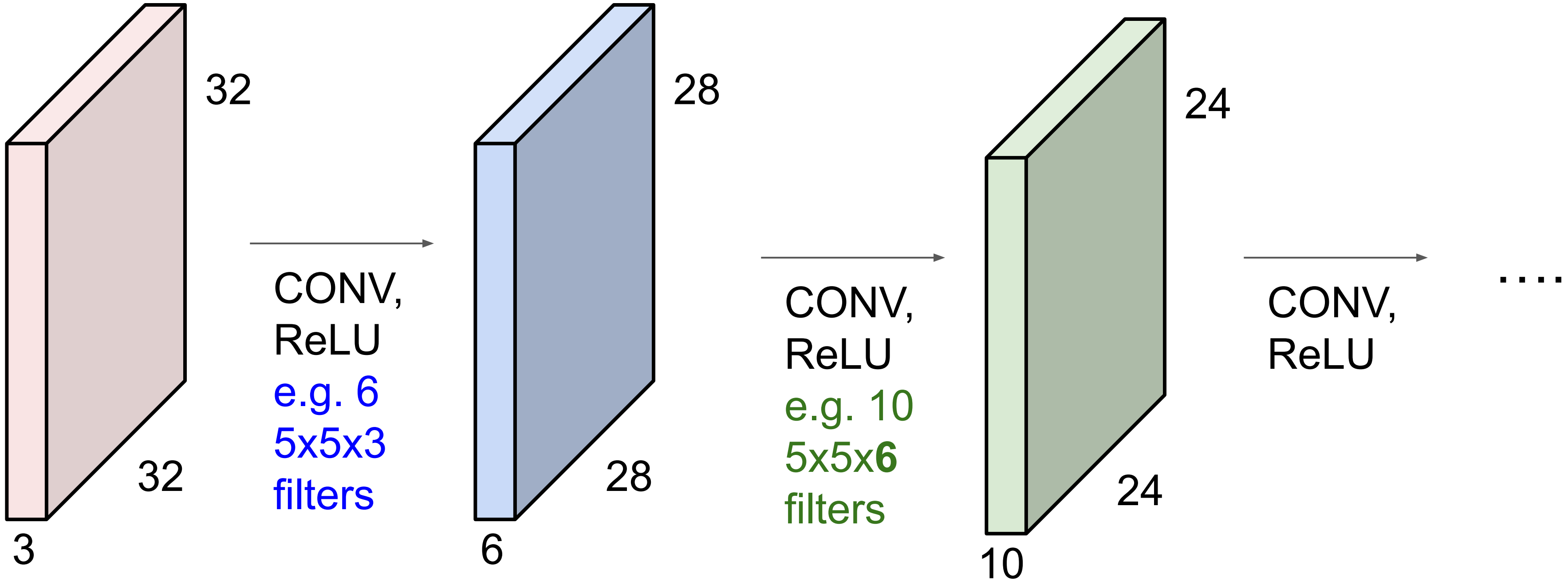
Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

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




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

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

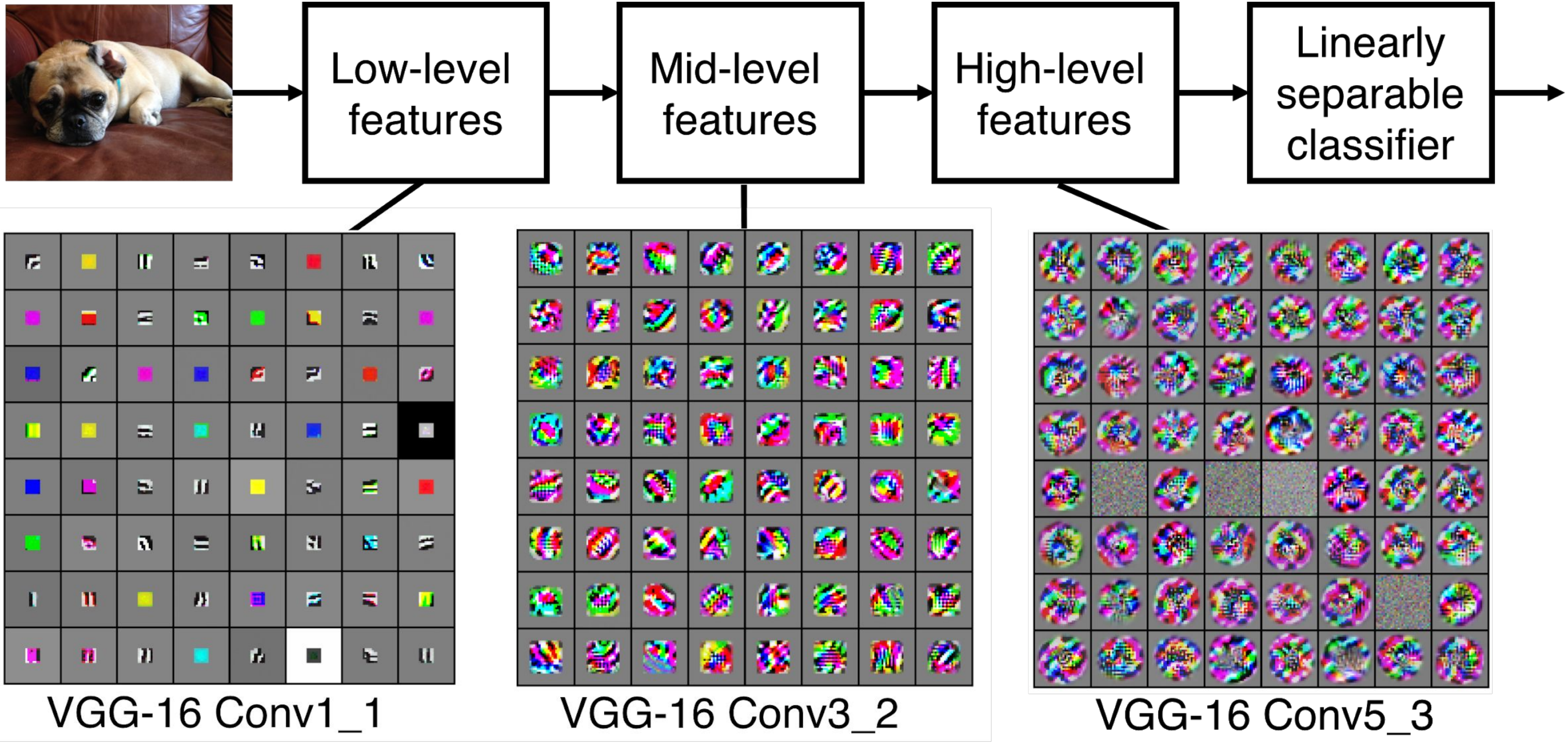



Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

## Preview

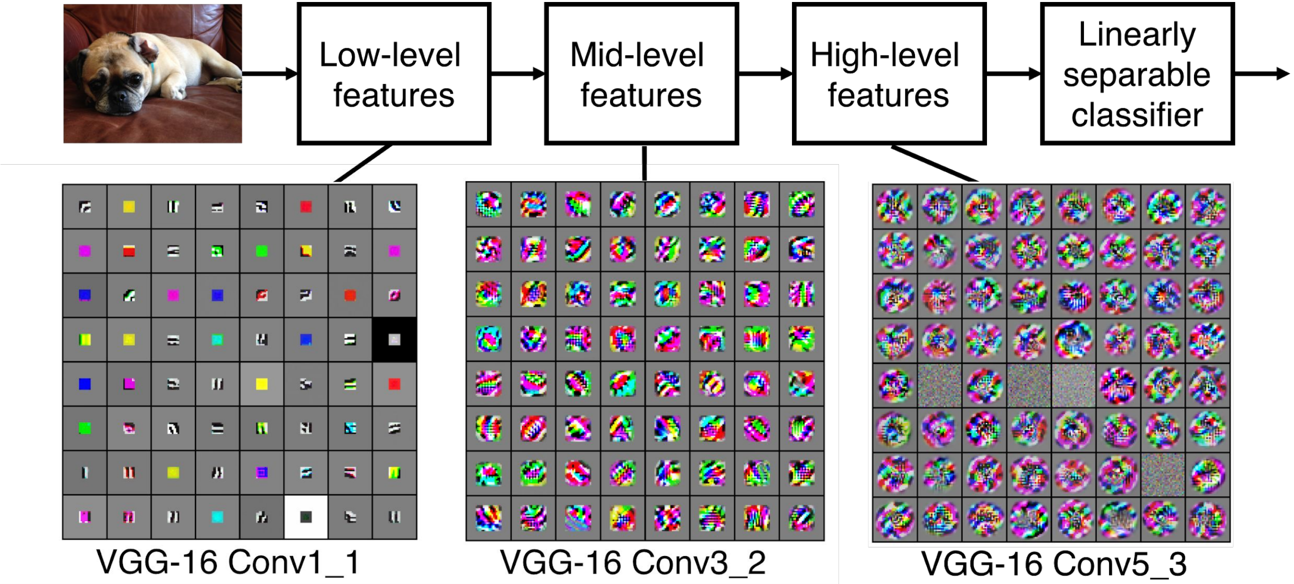
[Zeiler and Fergus 2013]

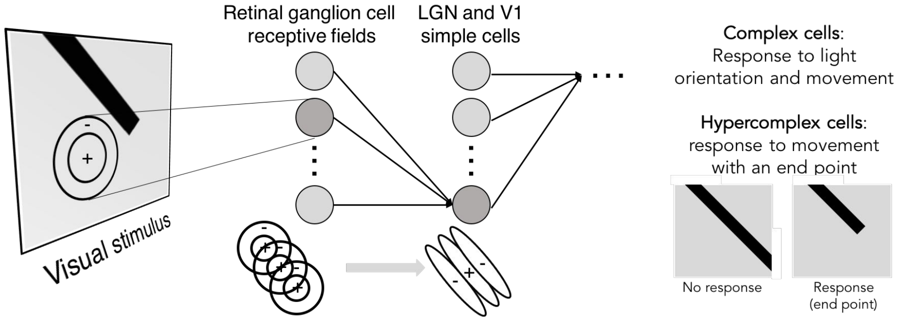
Visualization of VGG-16 by Lane McIntosh. VGG-16 architecture from [Simonyan and Zisserman 2014].



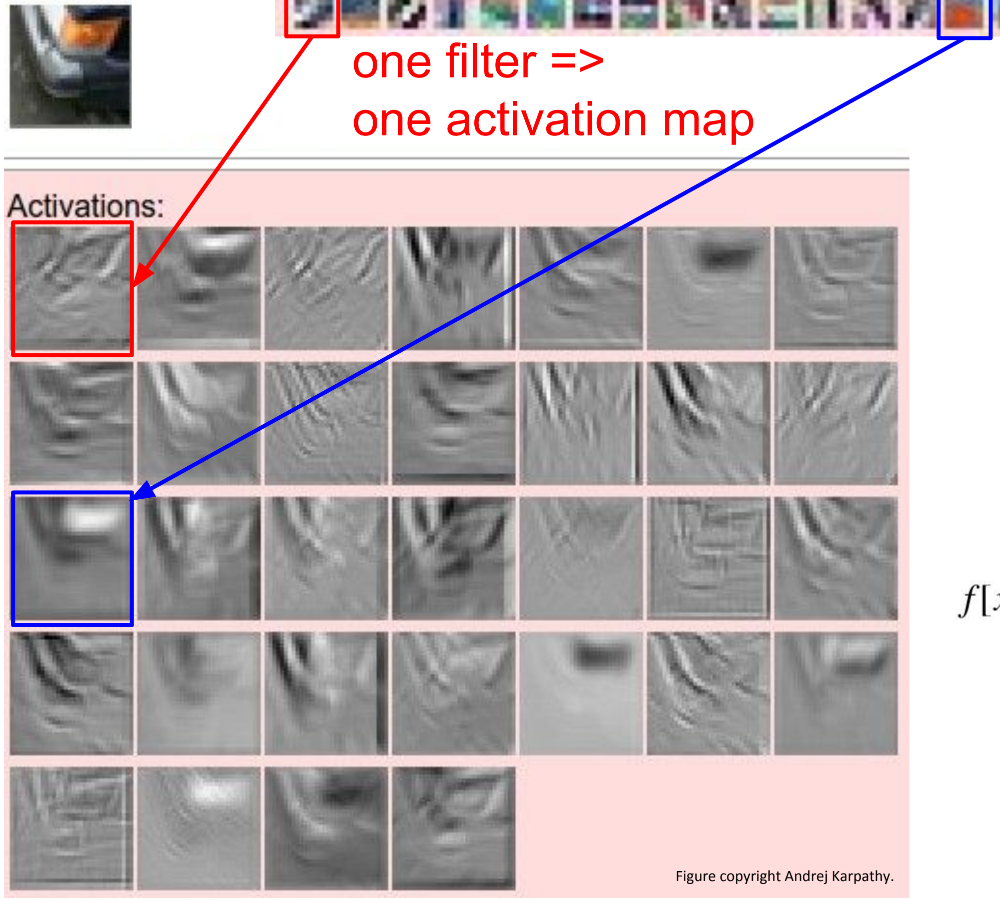

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Preview




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung



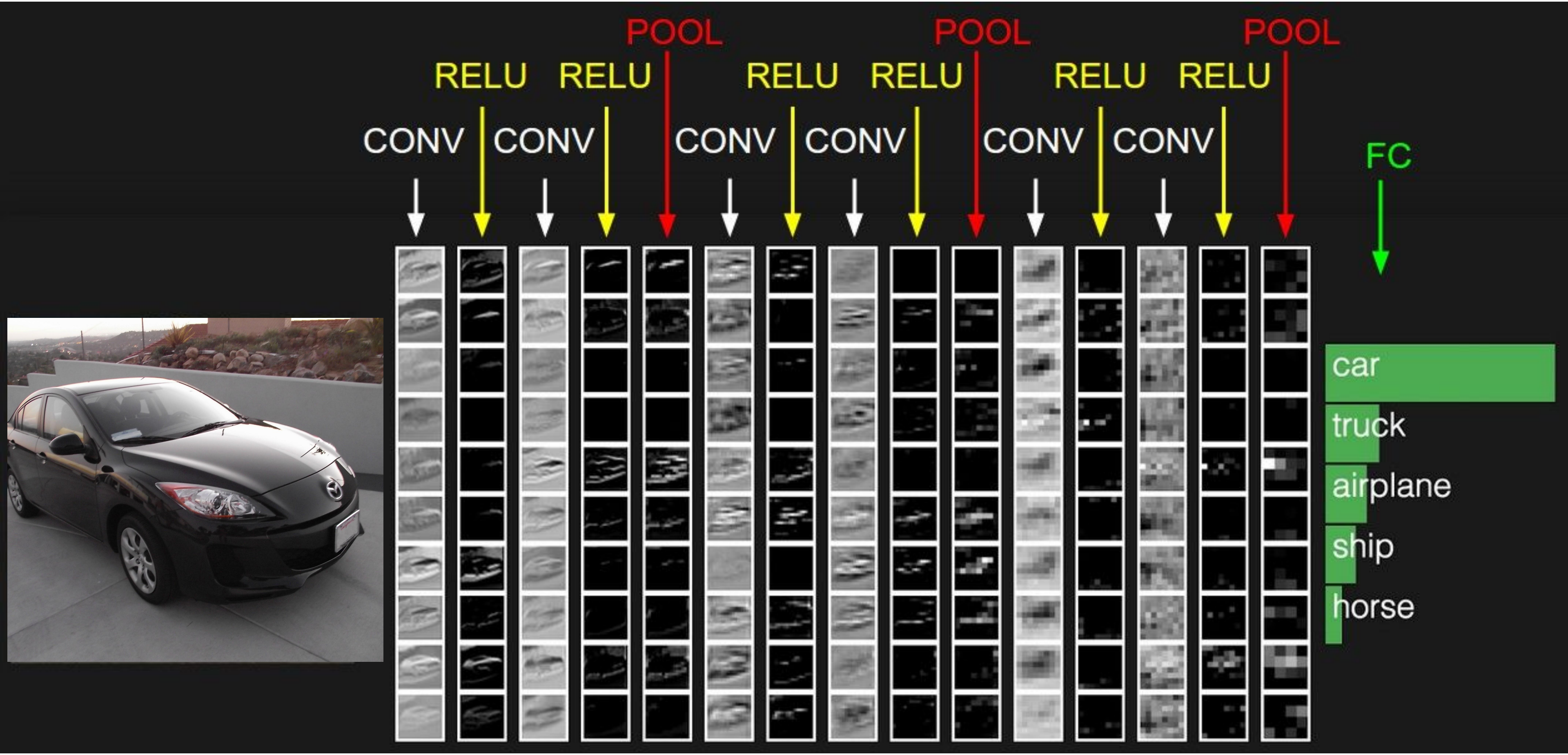



example 5x5 filters
(32 total)

We call the layer convolutional because it is related to convolution of two signals:

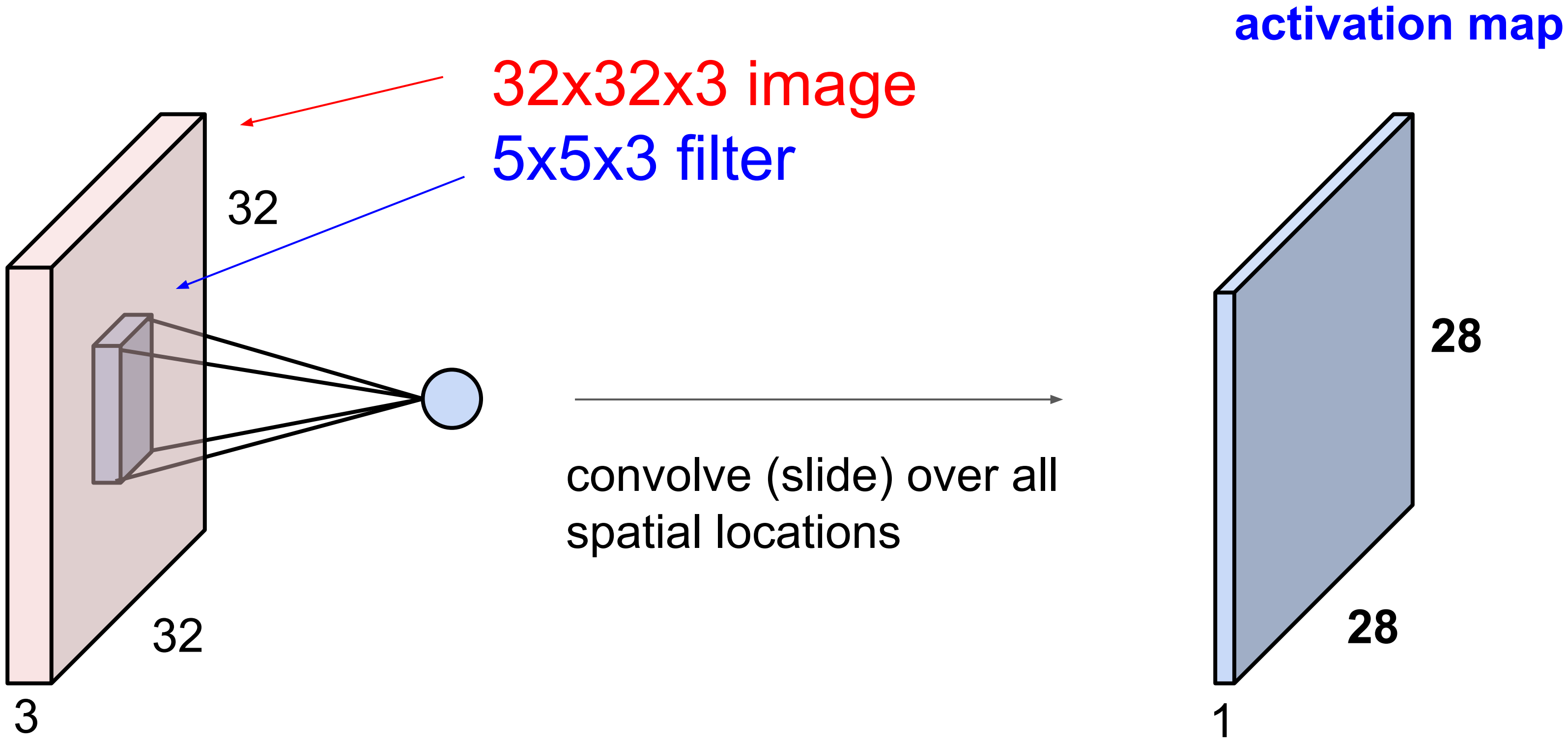
$$f[x,y] * g[x,y] = \sum_{n_1=-\infty}^{\infty} \sum_{n_2=-\infty}^{\infty} f[n_1,n_2] \cdot g[x-n_1, y-n_2]$$

elementwise multiplication and sum of a filter and the signal (image) preview:



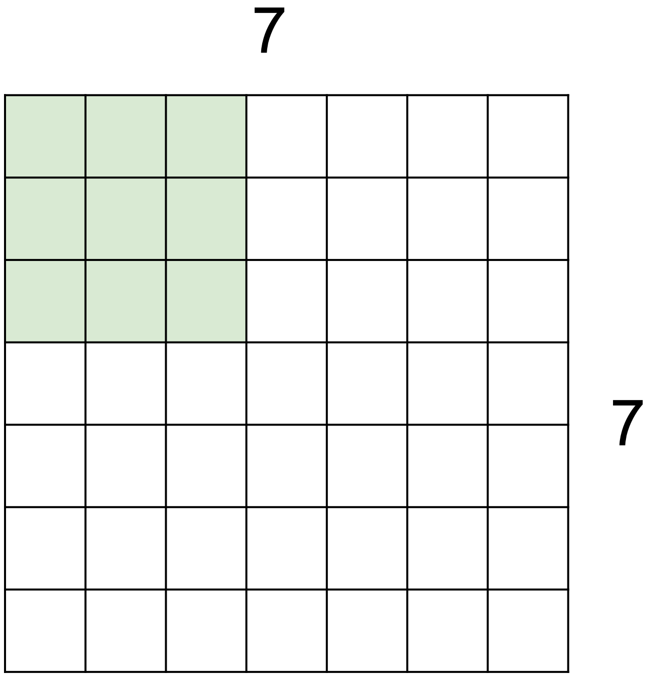

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A closer look at spatial dimensions:



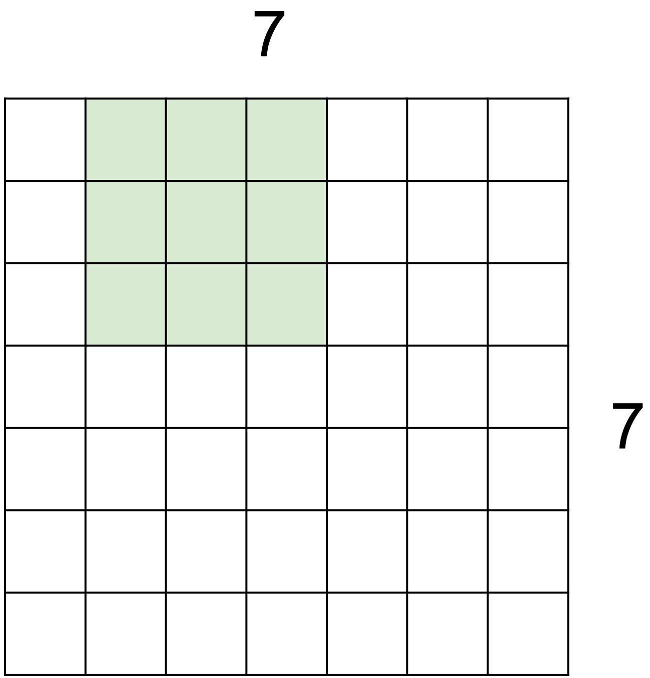

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A closer look at spatial dimensions:




7x7 input (spatially)
assume 3x3 filter

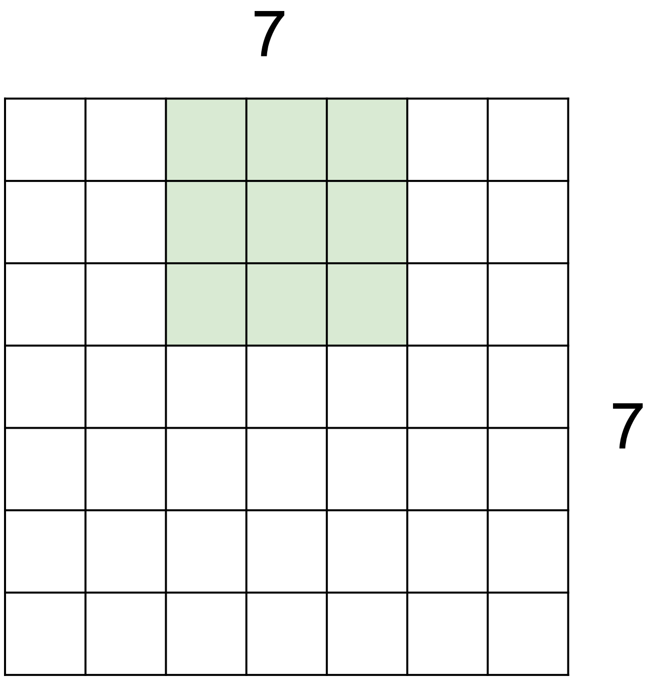
# A closer look at spatial dimensions:




7x7 input (spatially)
assume 3x3 filter

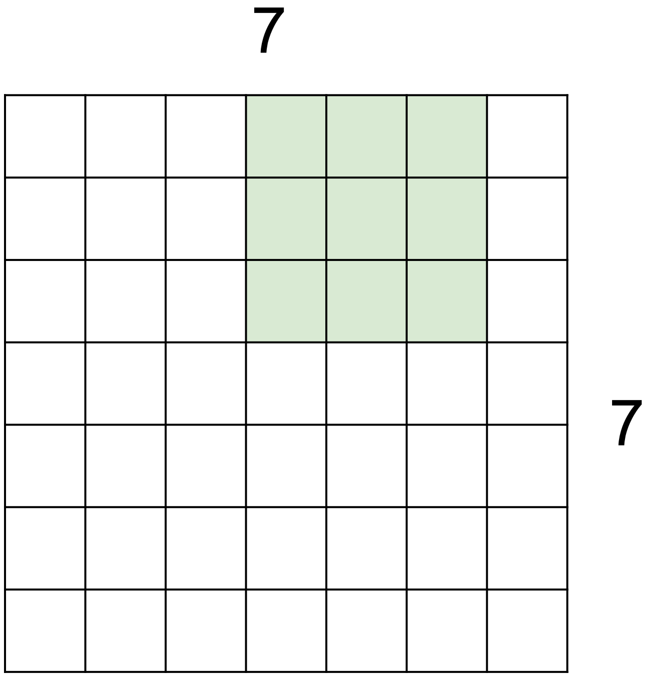
Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A closer look at spatial dimensions:



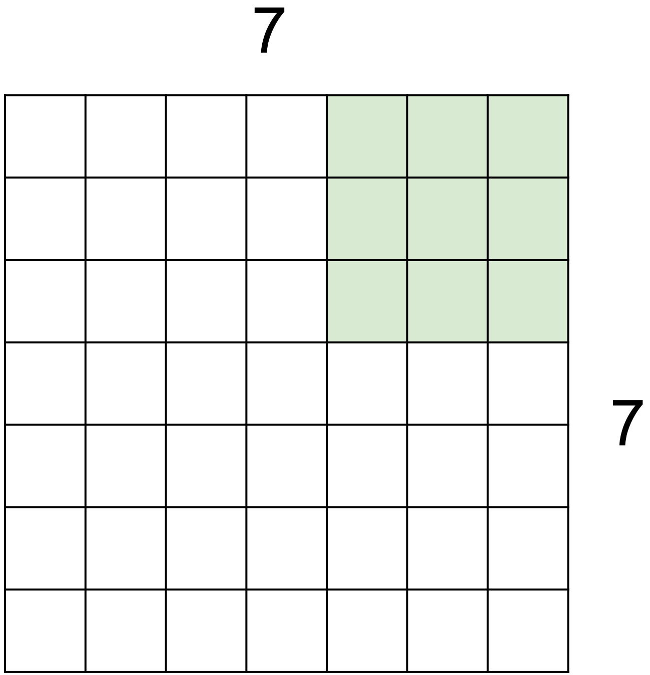

7x7 input (spatially)
assume 3x3 filter

# A closer look at spatial dimensions:




7x7 input (spatially)
assume 3x3 filter

# A closer look at spatial dimensions:

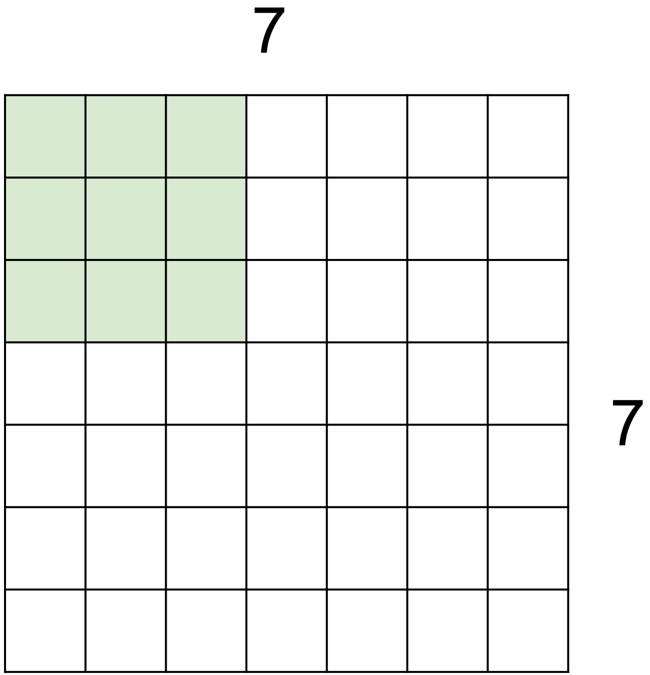



7x7 input (spatially)
assume 3x3 filter

**=> 5x5 output**

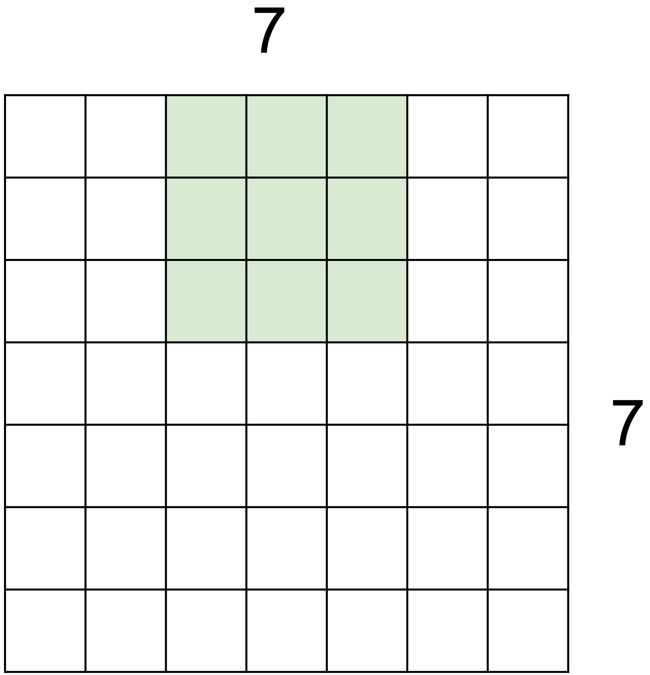
Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# A closer look at spatial dimensions:




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

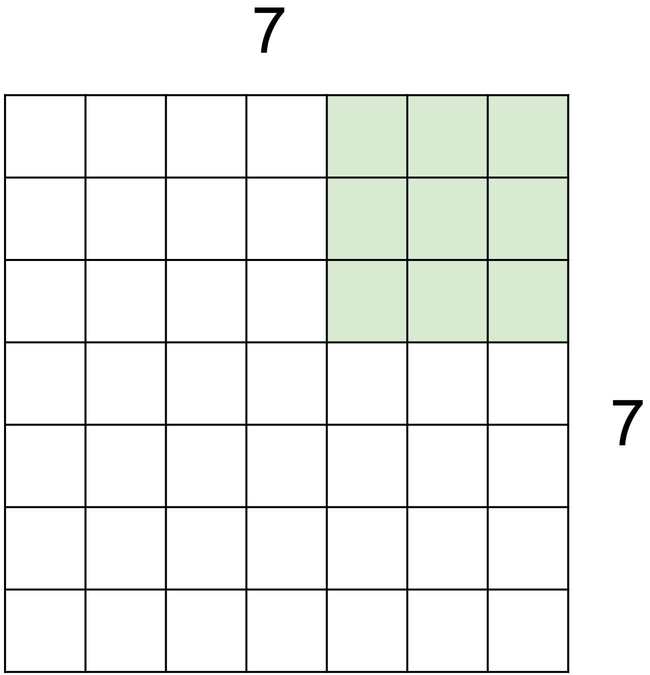
# A closer look at spatial dimensions:




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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

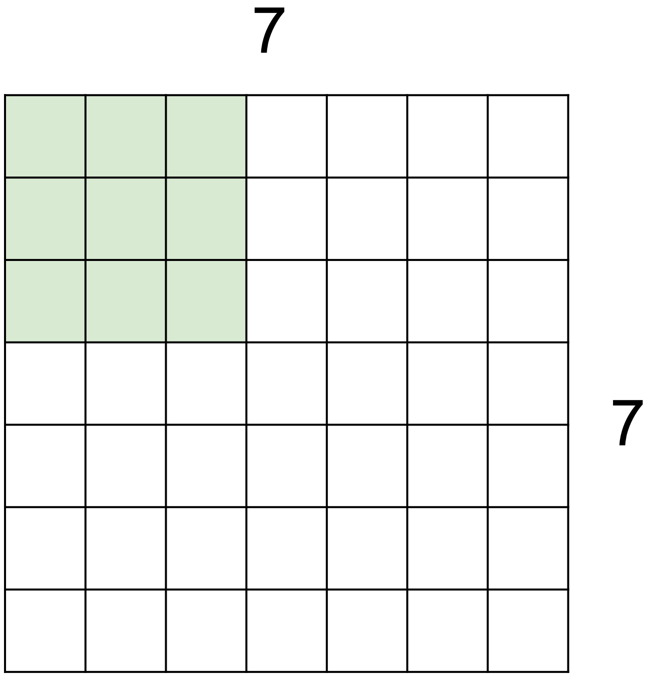
# A closer look at spatial dimensions:




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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

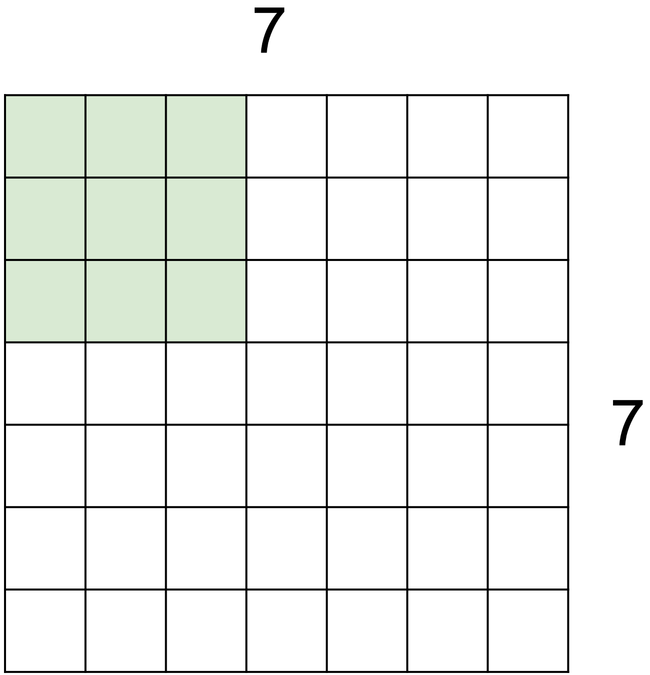
# A closer look at spatial dimensions:




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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

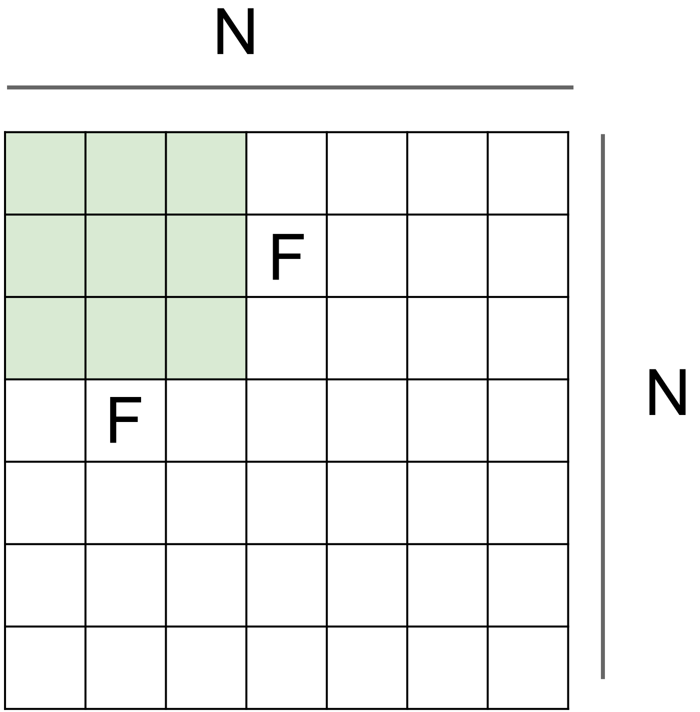
# A closer look at spatial dimensions:




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

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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

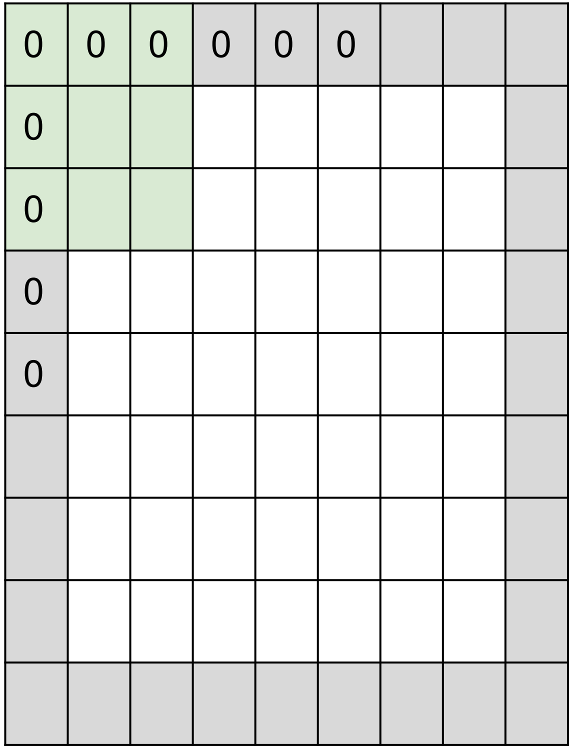



Output size:
**(N - F) / stride + 1**

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 :\

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

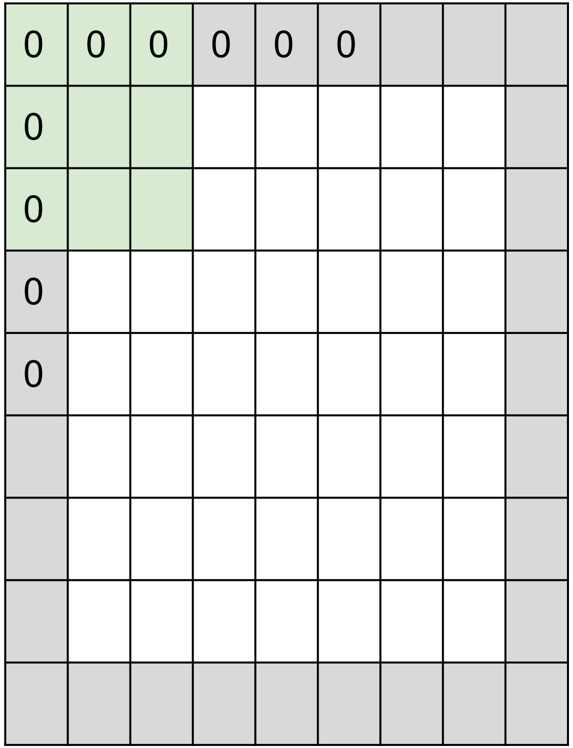
# In practice: Common to zero pad the border




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

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

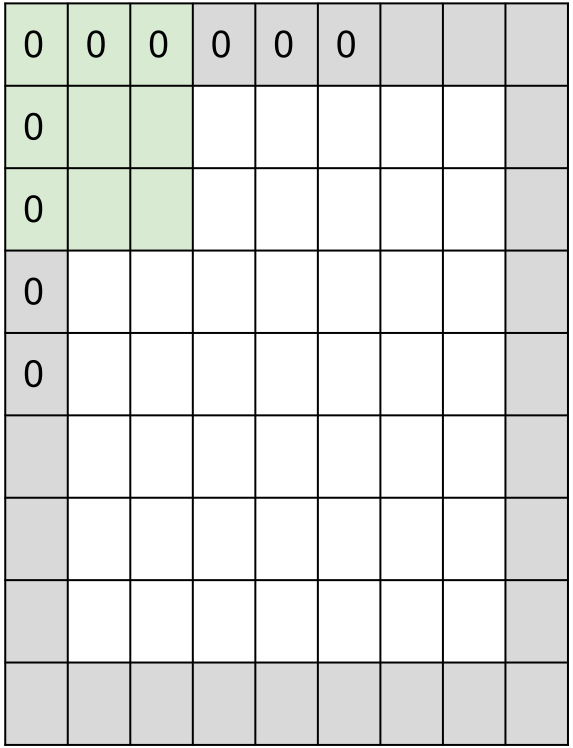
# In practice: Common to zero pad the border




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

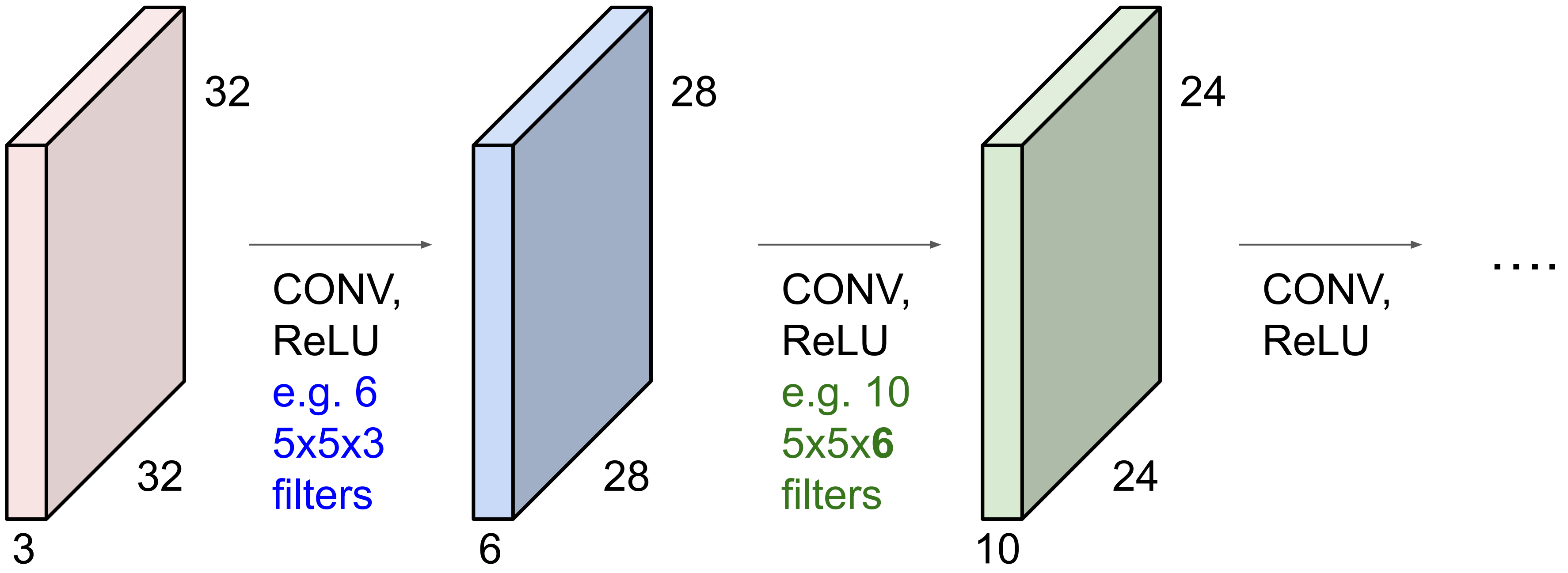



e.g. input 7x7
**3x3** filter, applied with **stride 1**
**pad with 1 pixel** border => what is the output?

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

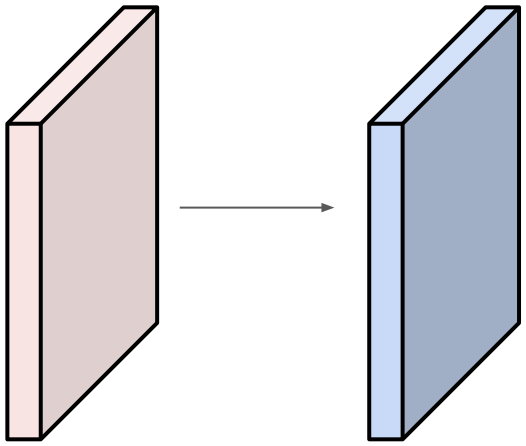
**Remember back to…**

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.




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

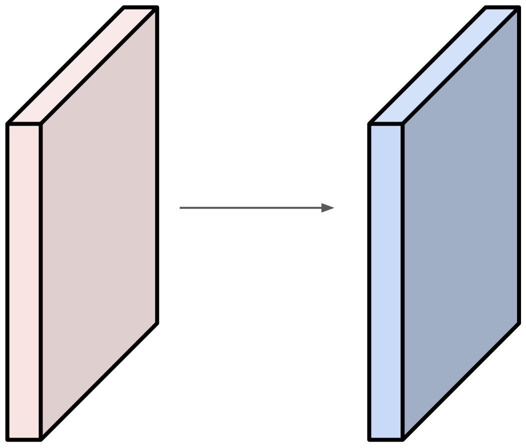
# Examples time:



Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Output volume size: ?

# Examples time:



Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2

Output volume size:

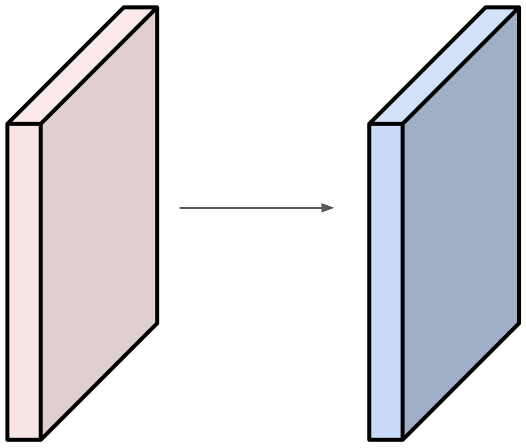
(32+2*2-5)/1+1 = 32 spatially, so

**32x32x10**

# Examples time:

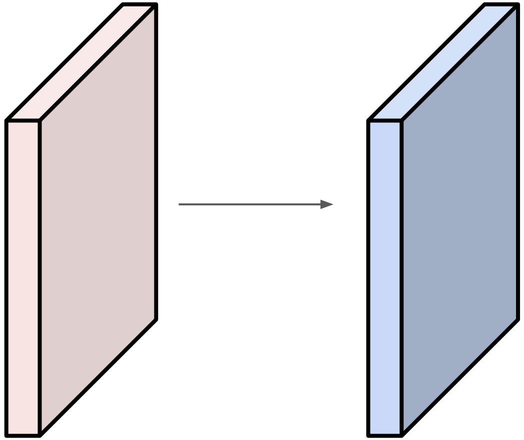
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2

Number of parameters in this layer?



Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Examples time:



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**

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

**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.

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

**Summary**. To summarize, the Conv Layer:

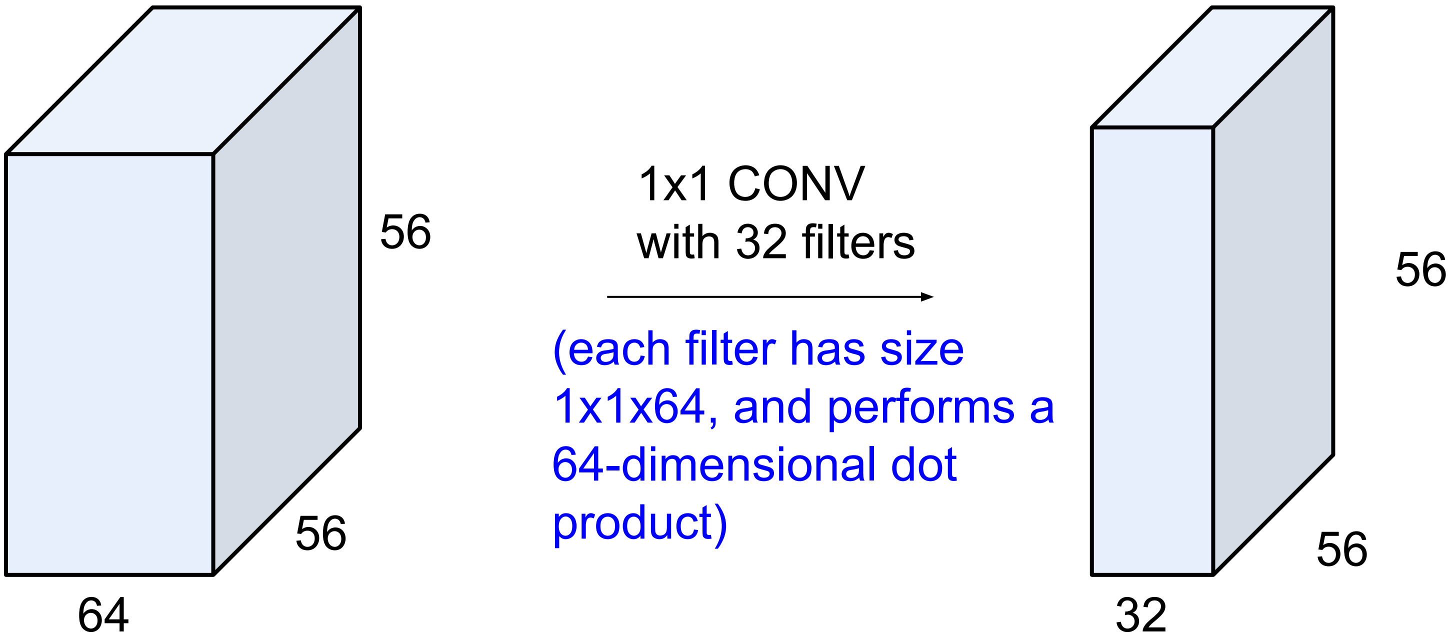
- Accepts a volume of size $W_1 \times H_1 \times D_1$
- Requires four hyperparameters:
  - Number of filters $K$,
  - their spatial extent $F$,
  - the stride $S$,
  - the amount of zero padding $P$.
- Produces a volume of size $W_2 \times H_2 \times D_2$ where:
  - $W_2 = (W_1 - F + 2P)/S + 1$
  -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

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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# (btw, 1x1 convolution layers make perfect sense)



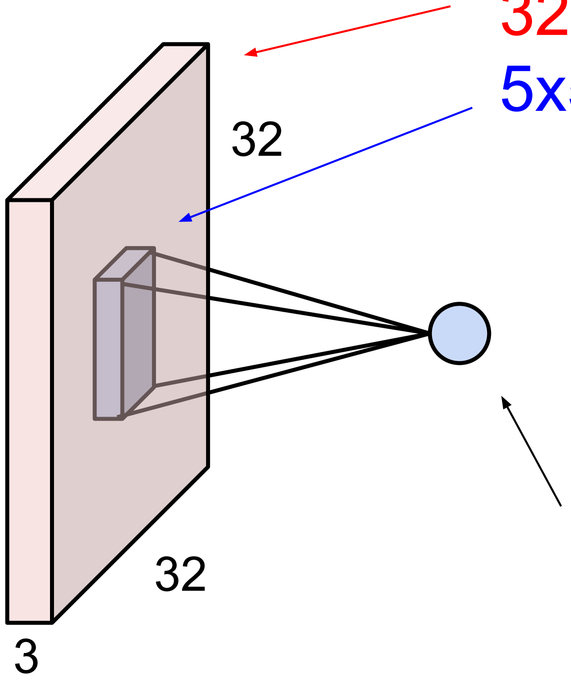

(each filter has size 1x1x64, and performs a 64-dimensional dot product)

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# The brain/neuron view of CONV Layer

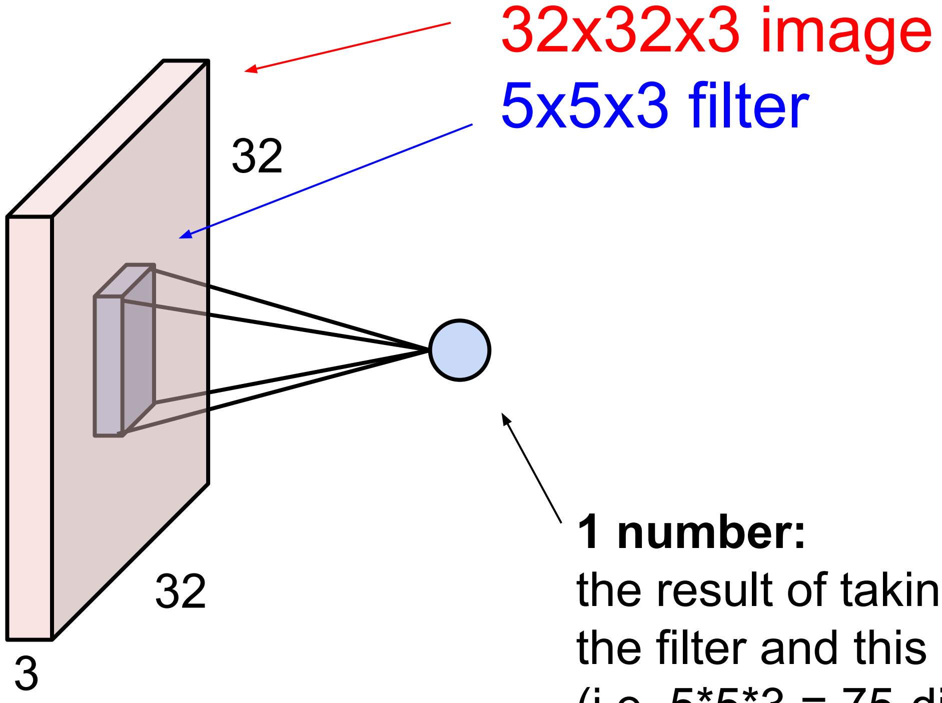
32x32x3 image

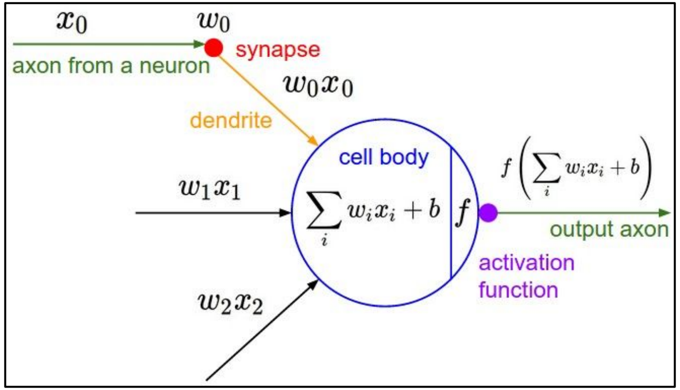
5x5x3 filter




**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)

# The brain/neuron view of CONV Layer



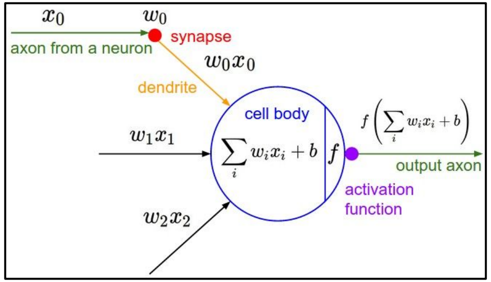



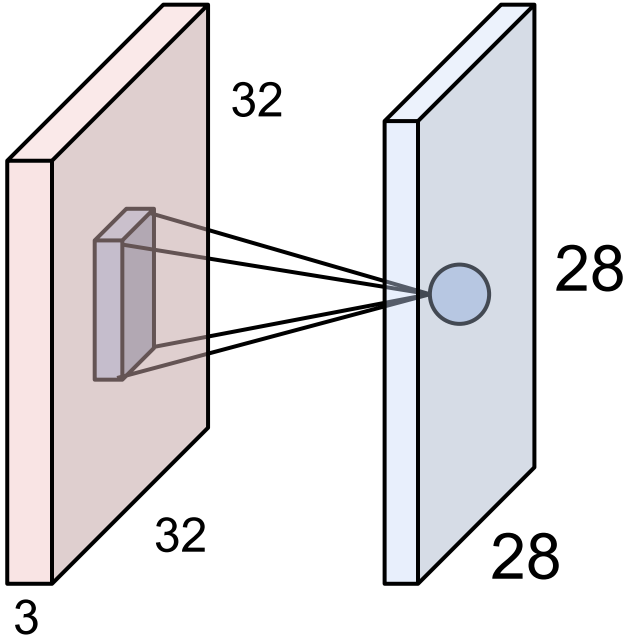

It's just a neuron with local connectivity...

**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)

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# The brain/neuron view of CONV Layer



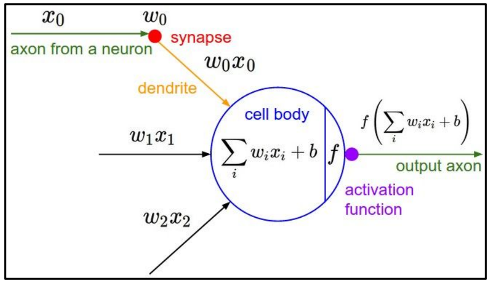



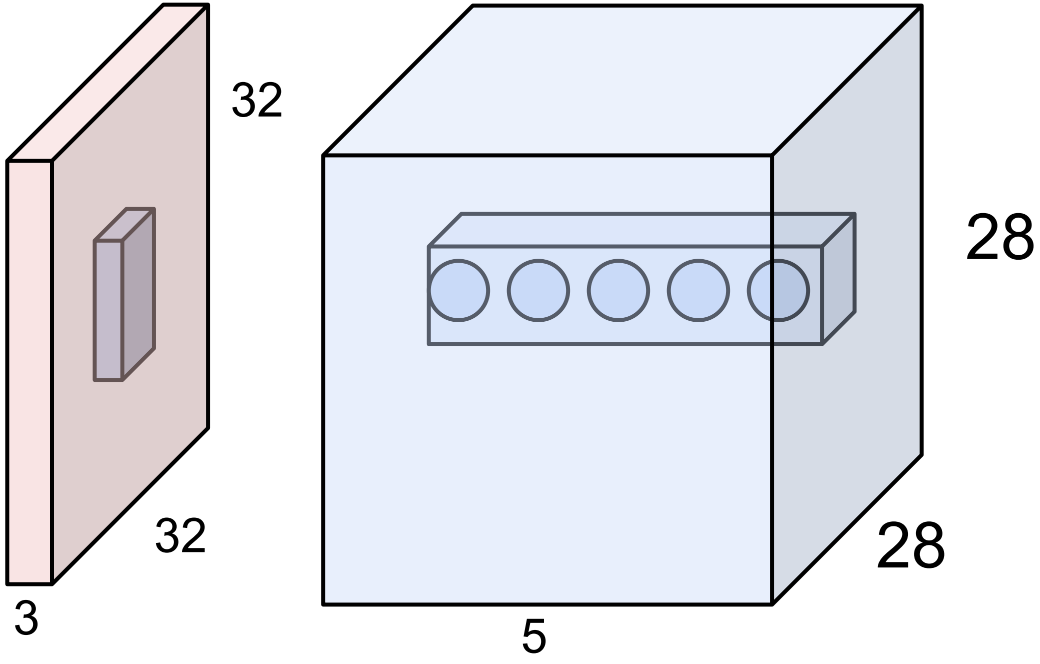

An activation map is a 28x28 sheet of neuron outputs:

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”

# The brain/neuron view of CONV Layer




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

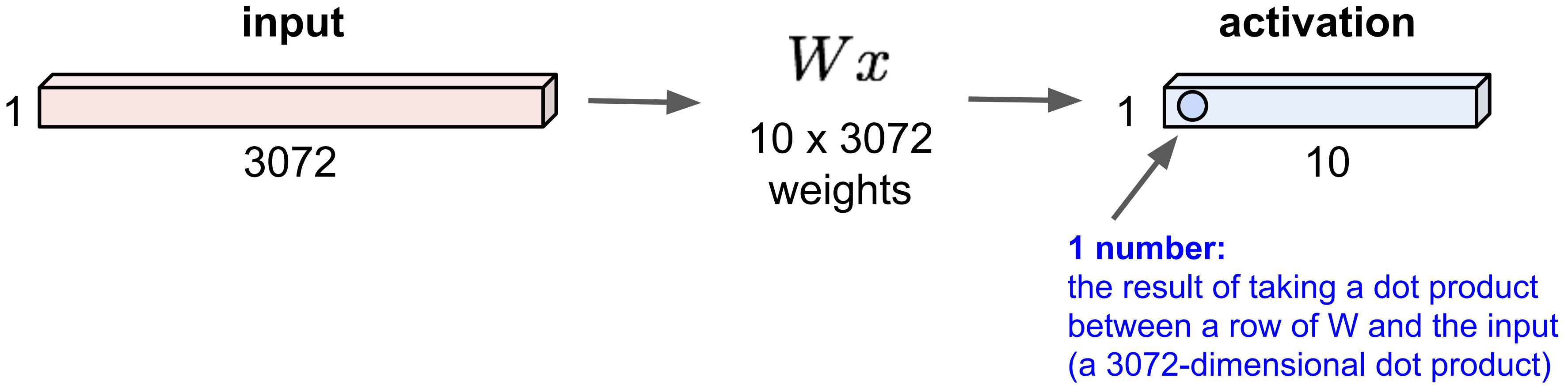
There will be 5 different neurons all looking at the same region in the input volume

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Reminder: Fully Connected Layer

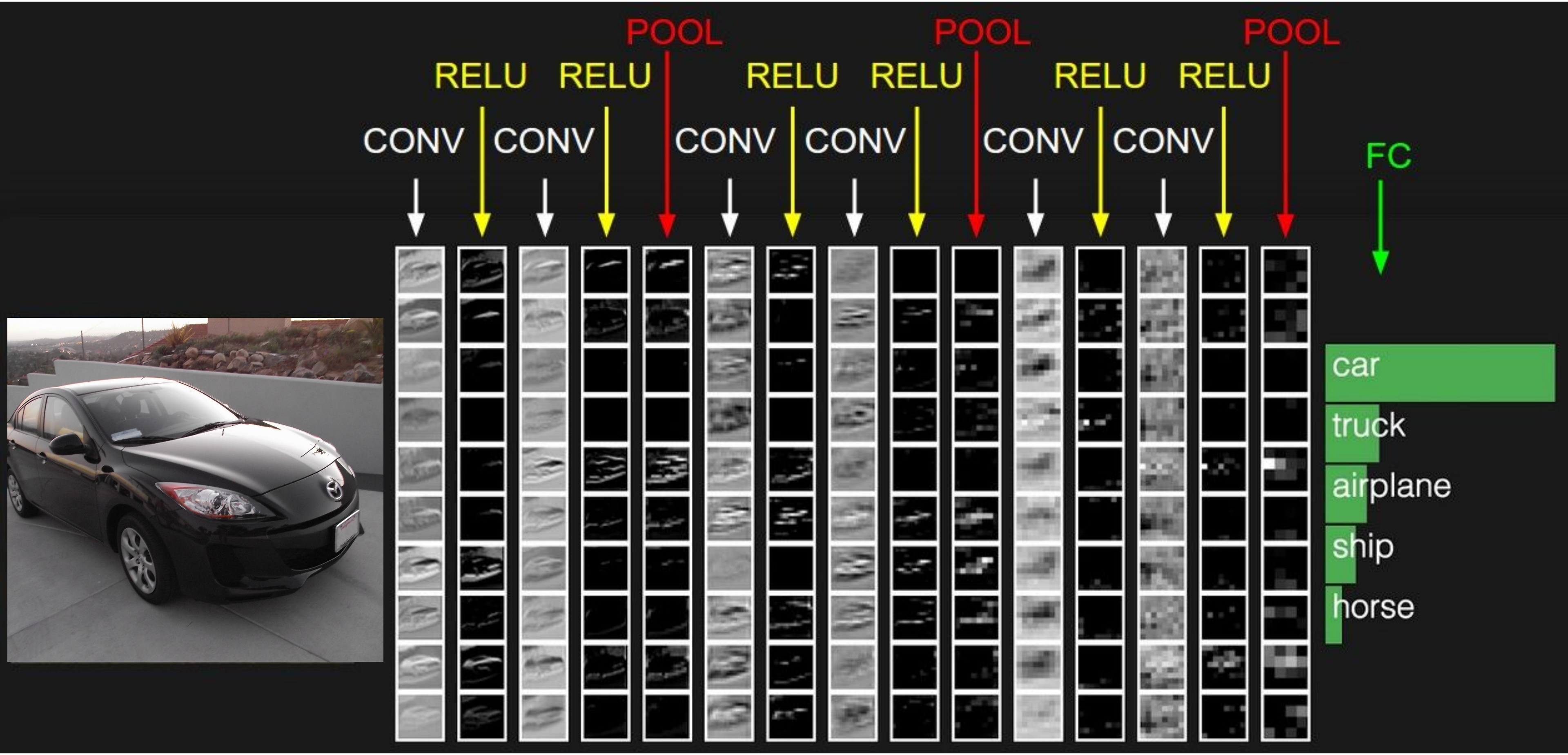
32x32x3 image -> stretch to 3072 x 1

Each neuron looks at the full input volume



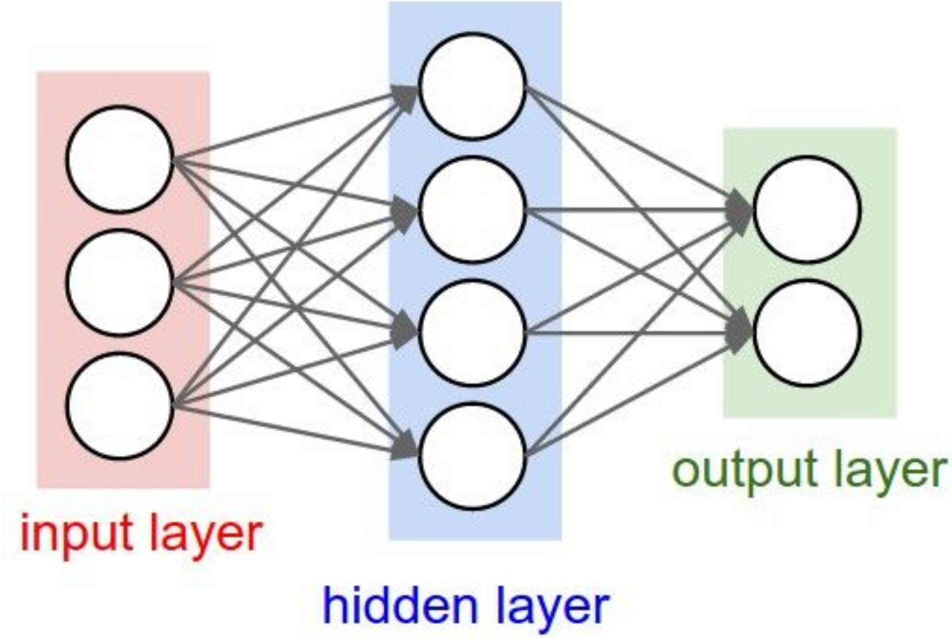

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

three more layers to go: RELU/POOL/FC




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Reminder: need non-linearity

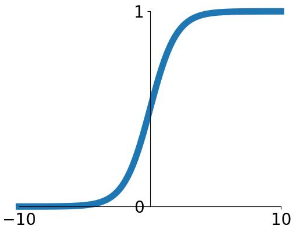



Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Reminder: Activation functions

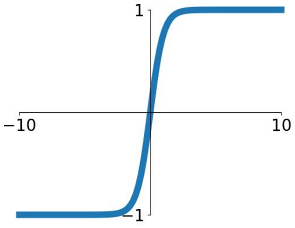
**Sigmoid**

$\sigma(x) = \frac{1}{1+e^{-x}}$



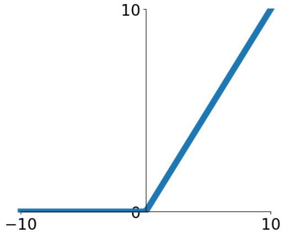

**tanh**

$\tanh(x)$



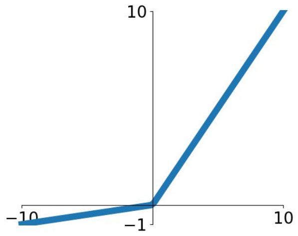

**ReLU**

$\max(0, x)$




**Leaky ReLU**

$\max(0.1x, x)$

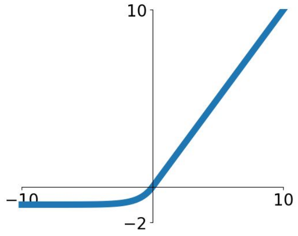



**Maxout**

$\max(w_1^T x + b_1, w_2^T x + b_2)$

**ELU**

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

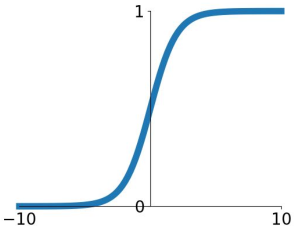



Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Reminder: Activation functions

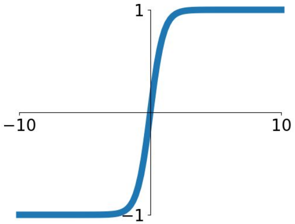
## Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



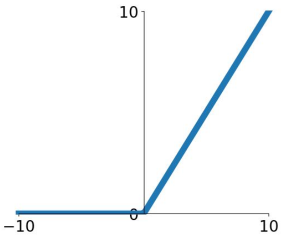

## tanh

$$\tanh(x)$$




## ReLU

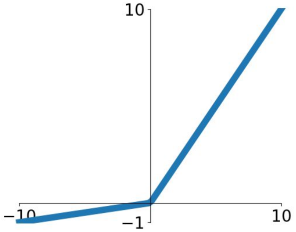
$$\max(0, x)$$




Good default choice

## Leaky ReLU

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

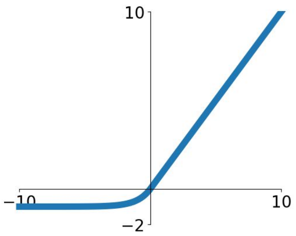



## Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

## ELU

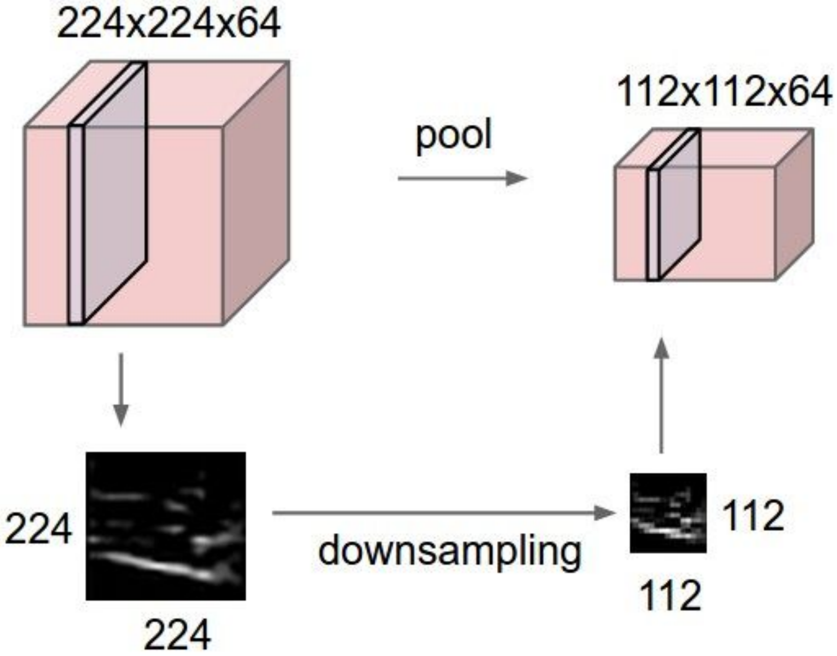
$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

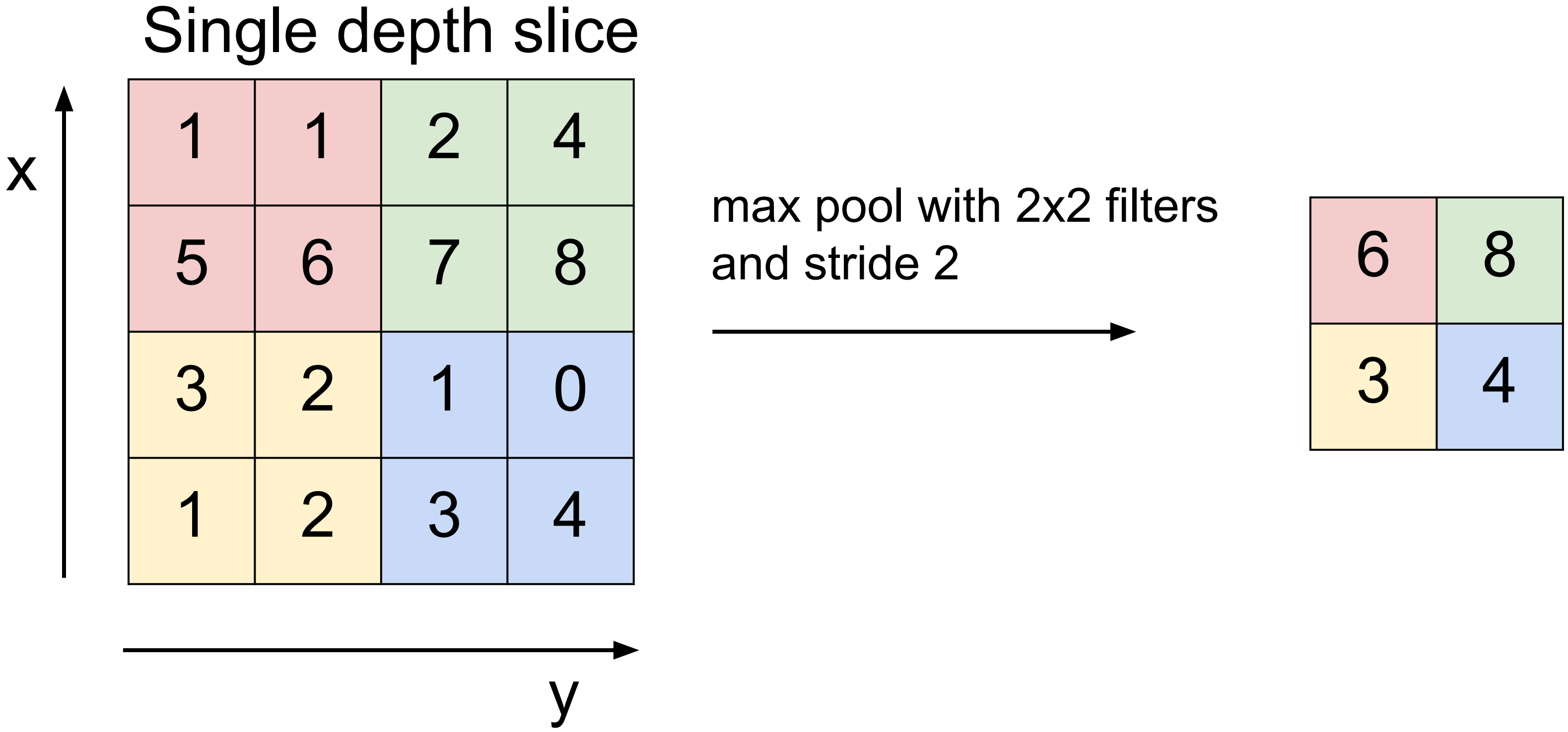
# Pooling layer

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




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# MAX POOLING




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

- 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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

- 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

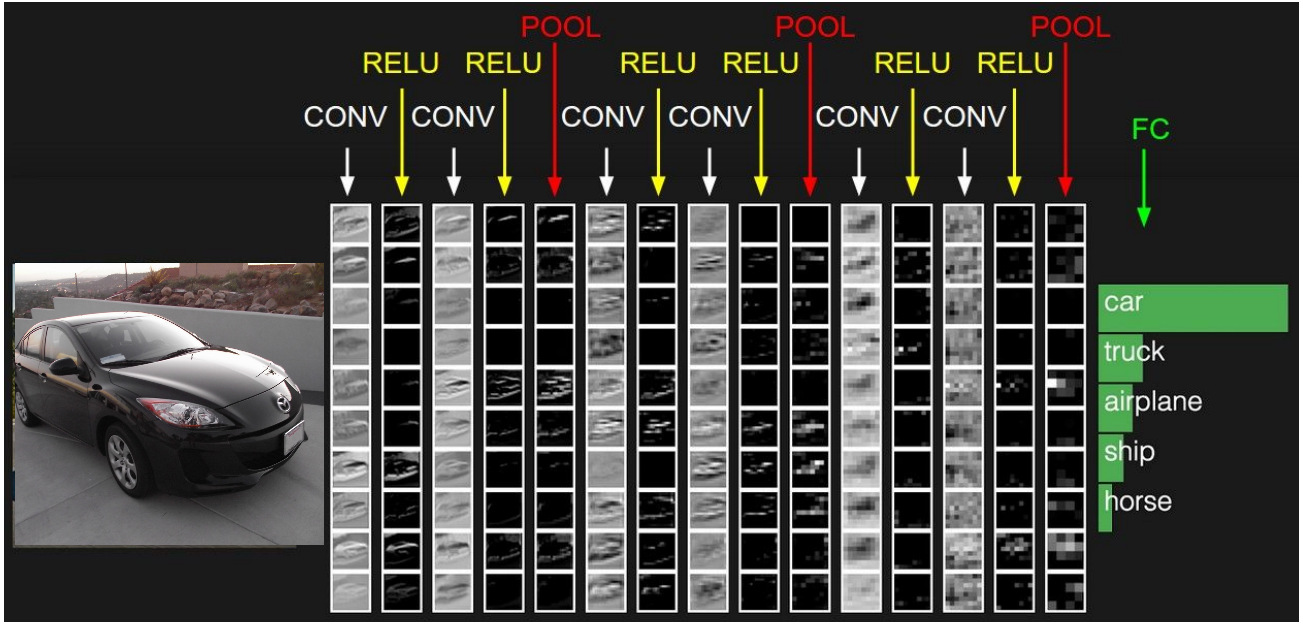
Common settings:

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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# Fully Connected Layer (FC layer)

- Contains neurons that connect to the entire input volume, as in ordinary Neural Networks




Credit: Fei-Fei Li & Justin Johnson & Serena Yeung

# 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

Credit: Fei-Fei Li & Justin Johnson & Serena Yeung