Introduction to Deep Learning

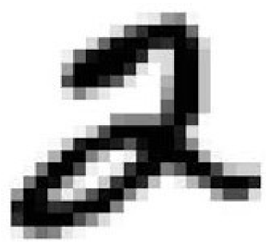
Presentation Title

H M Sabbir Ahmad

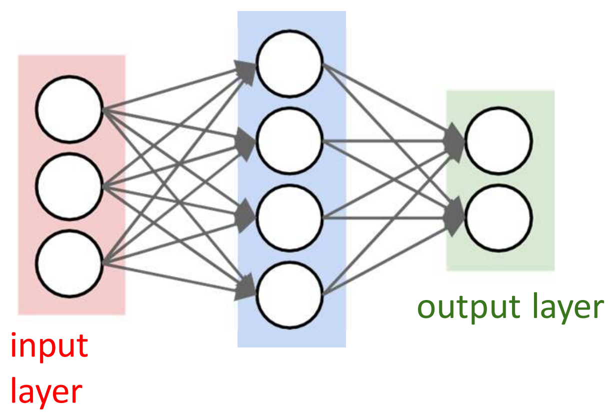
Phd, Systems Engineering.

09/12/2024

Multi-Layer Perception Issue



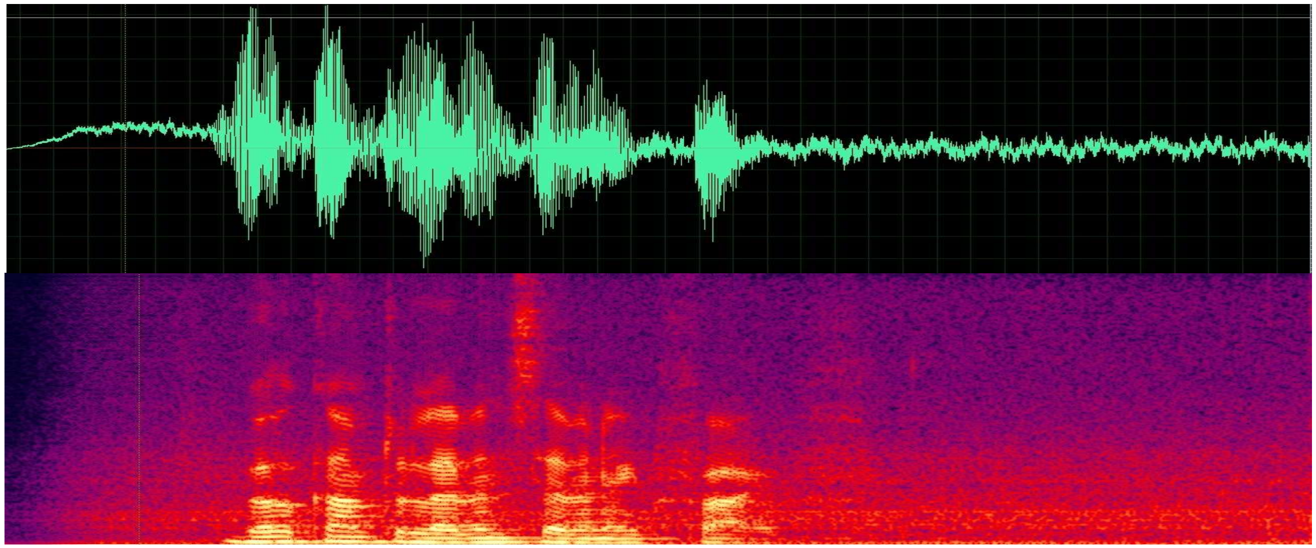
Or, more generally
a vector input



- Can recognize patterns in data
  - E.g. digits
  - Or any other vector data

Slide credit: Bhiksha Raj

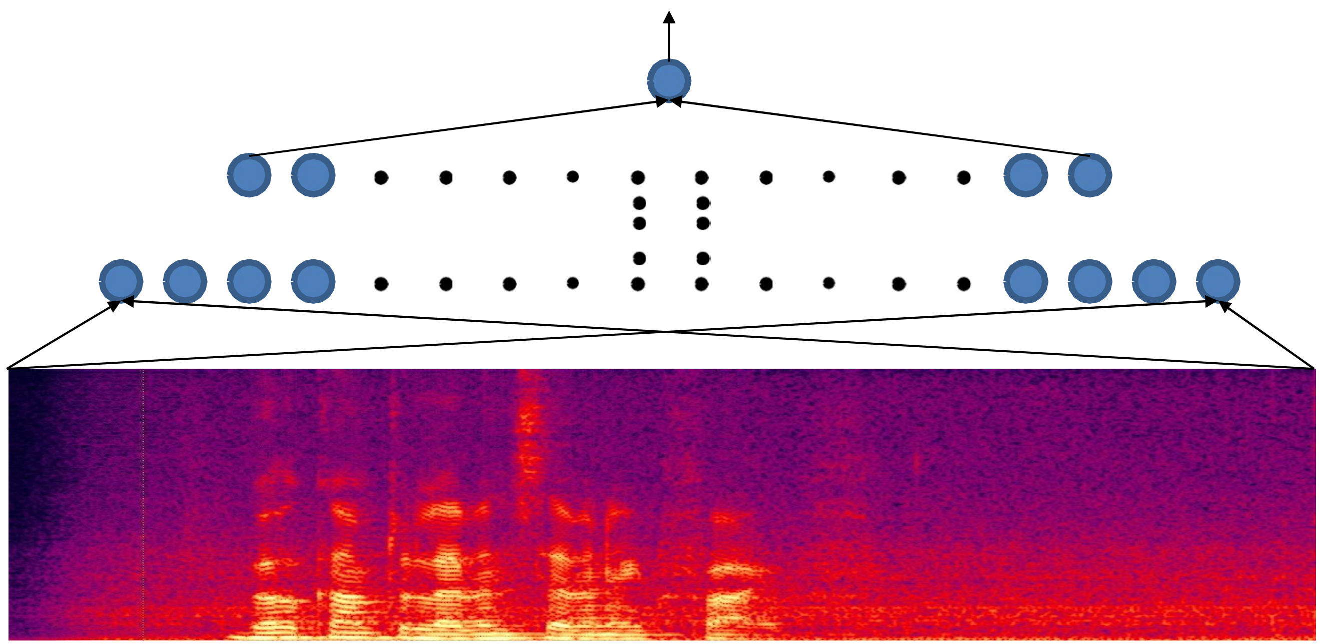
A new problem



- Does this signal contain the word “Welcome”?
- Compose an MLP for this problem.
  - Assuming all recordings are exactly the same length..

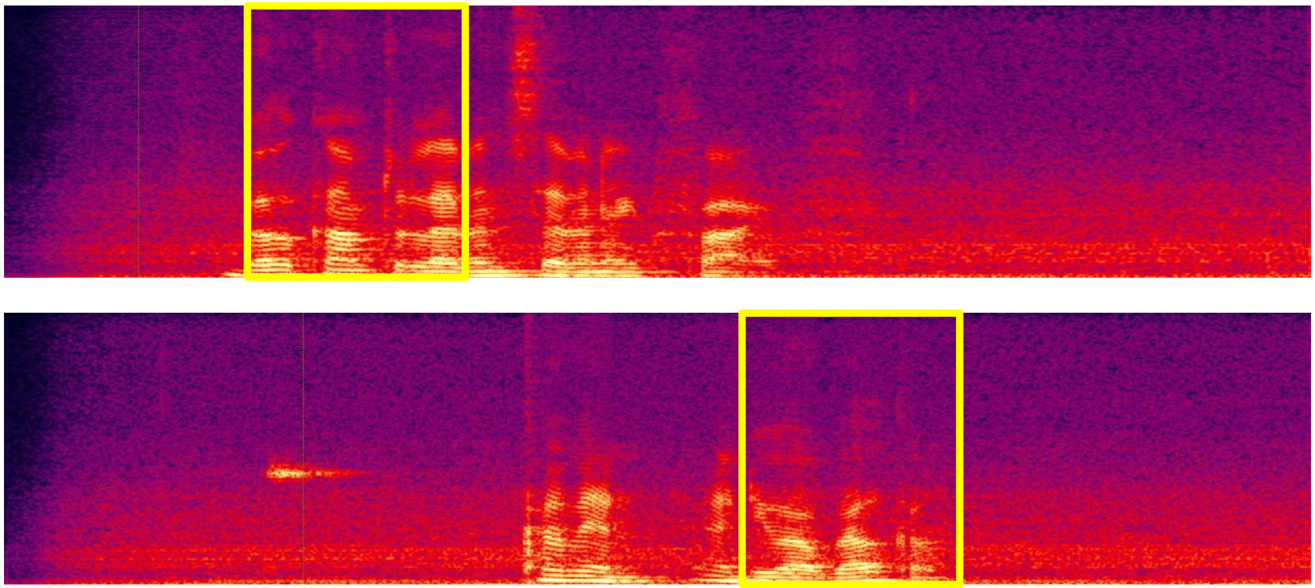
Slide credit: Bhiksha Raj

Finding a Welcome



- Trivial solution: Train an MLP for the entire recording

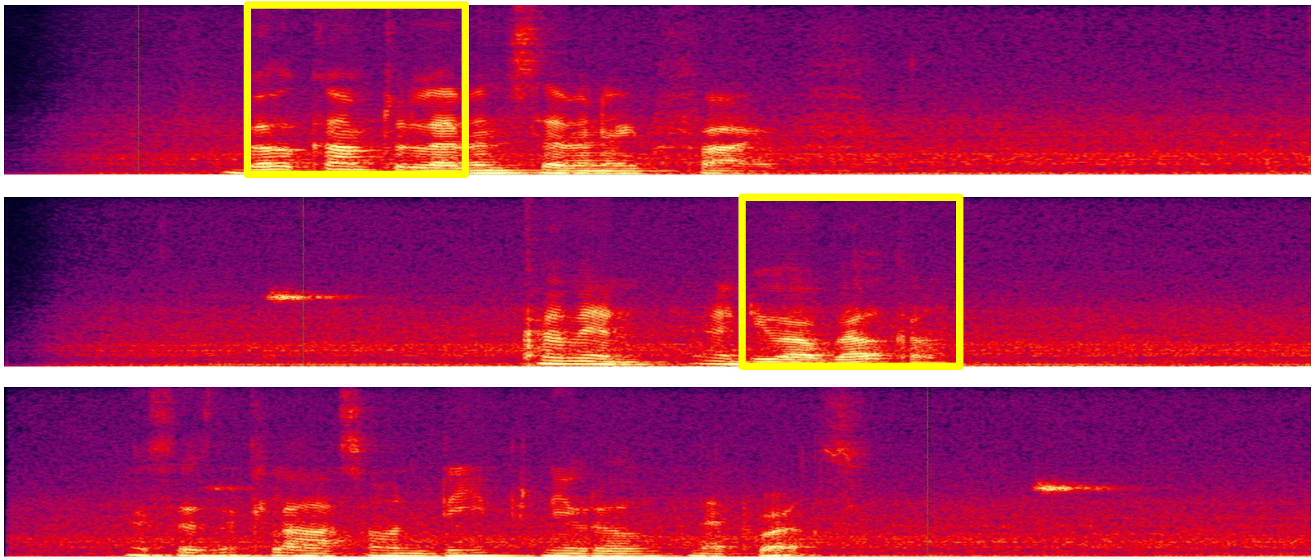
Finding a Welcome



- Problem with trivial solution: Network that finds a "welcome" in the top recording will not find it in the lower one
  - Unless trained with both
  - Will require a very large network and a large amount of training data to cover every case

Slide credit: Bhiksha Raj

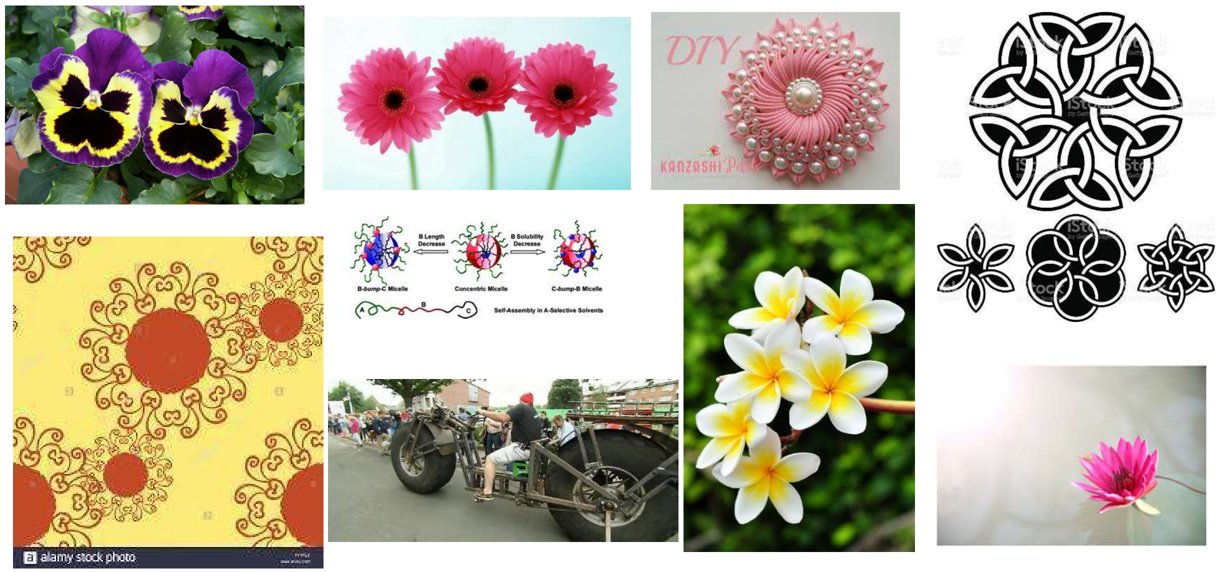
Finding a Welcome



- Need a *simple* network that will fire regardless of the location of "Welcome"
  - and not fire when there is none

Slide credit: Bhiksha Raj

Flowers

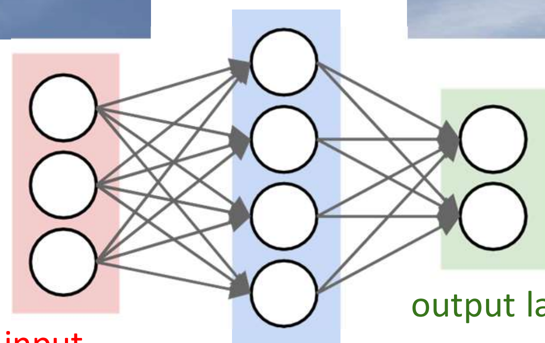


- Is there a flower in any of these images

Boston University School/college name here

Slide credit: Bhiksha Raj

A problem



- Will an MLP that recognizes the left image as a flower also recognize the one on the right as a flower?

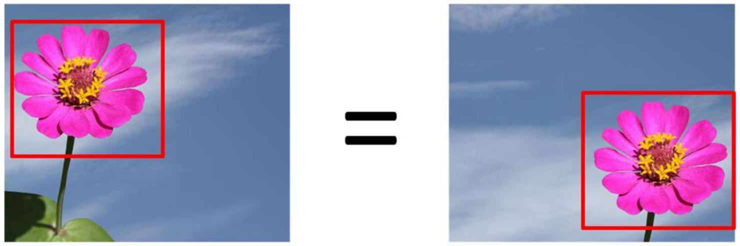
Slide credit: Bhiksha Raj

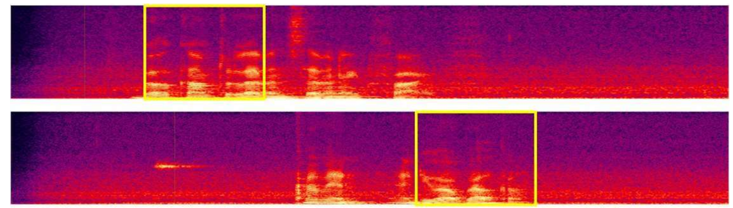
A problem

- Need a network that will “fire” regardless of the precise location of the target object

Slide credit: Bhiksha Raj

The need for *shift invariance*

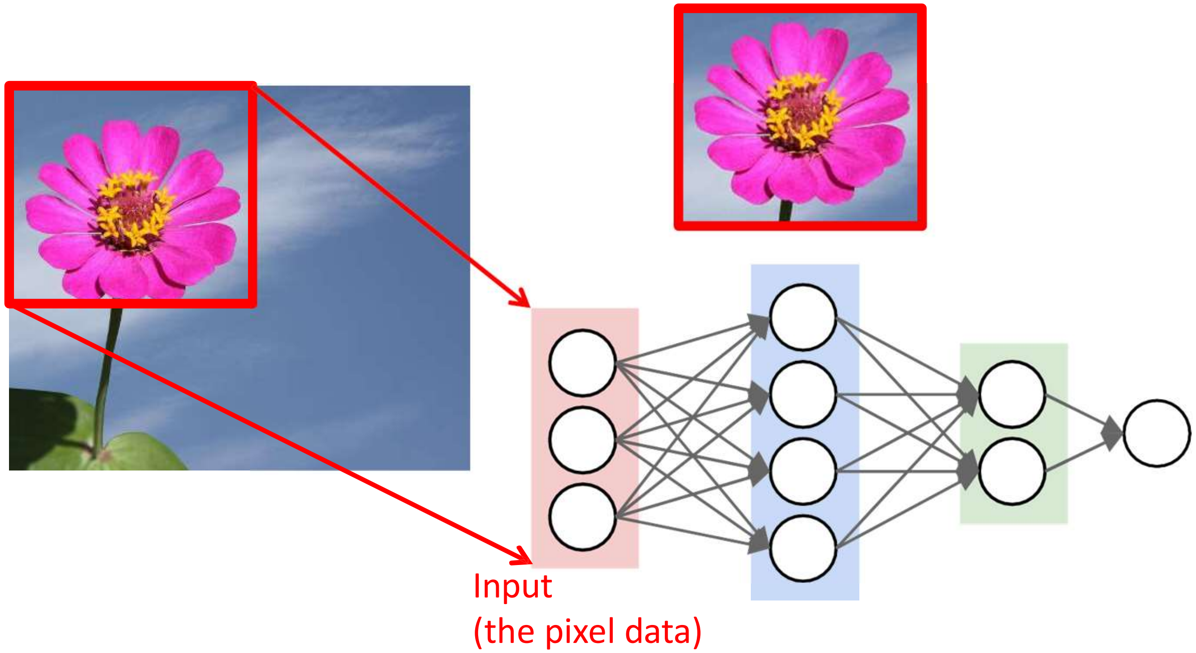




- In many problems the *location* of a pattern is not important
  - Only the presence of the pattern
- Conventional MLPs are sensitive to the location of the pattern
  - Moving it by one component results in an entirely different input that the MLP won't recognize
- Requirement: Network must be *shift invariant*

Slide credit: Bhiksha Raj

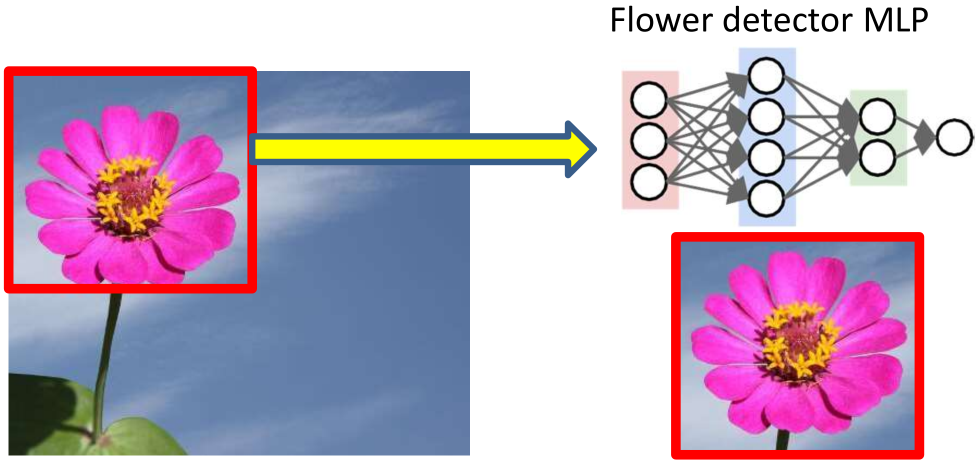
The 2-d analogue: Does this picture **have a flower?**



- *Scan* for the desired object
  - "Look" for the target object at each position

Slide credit: Bhiksha Raj

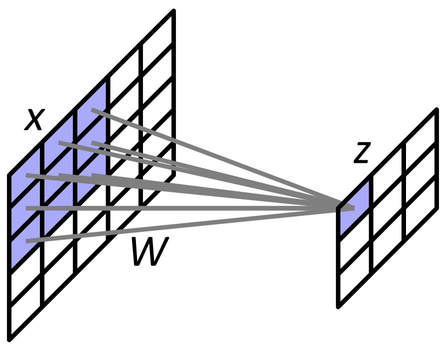
Solution: Scan

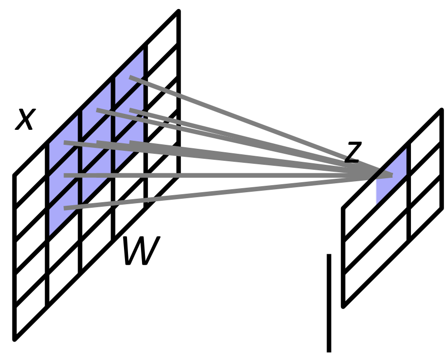


- *Scan* for the desired object
- At each location, the entire region is sent through the MLP

Slide credit: Bhiksha Raj

Advantages of convolutions





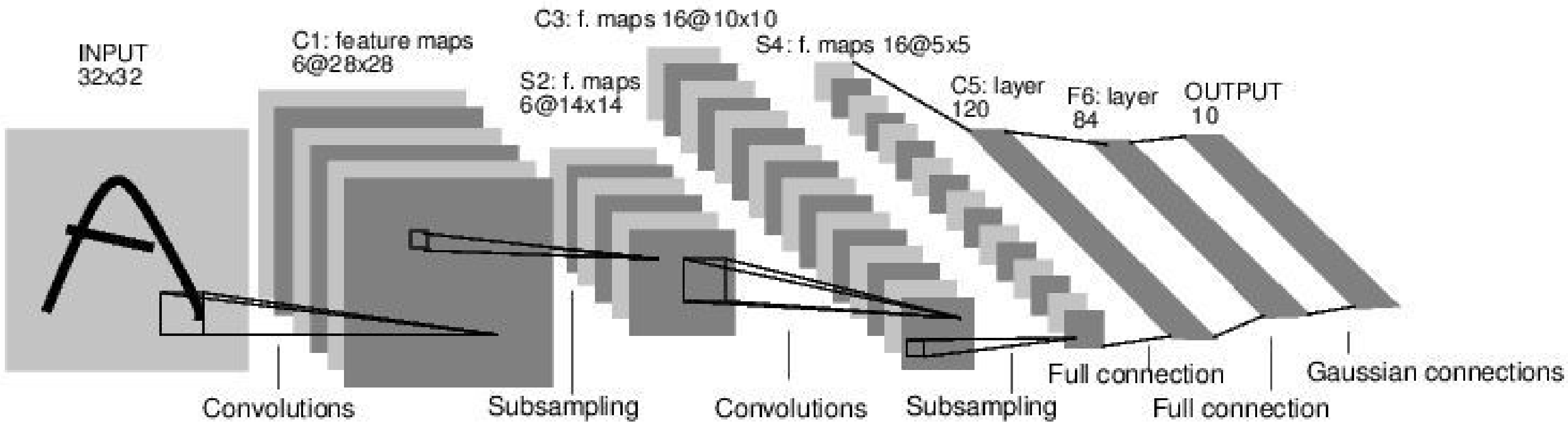
Drastically reduces the parameter count

- 256x256 grayscale image ⟹ 256x256 single-channel hidden layer: 4 *billion parameters* in fully connected network to *9 parameters* in 3x3 convolution

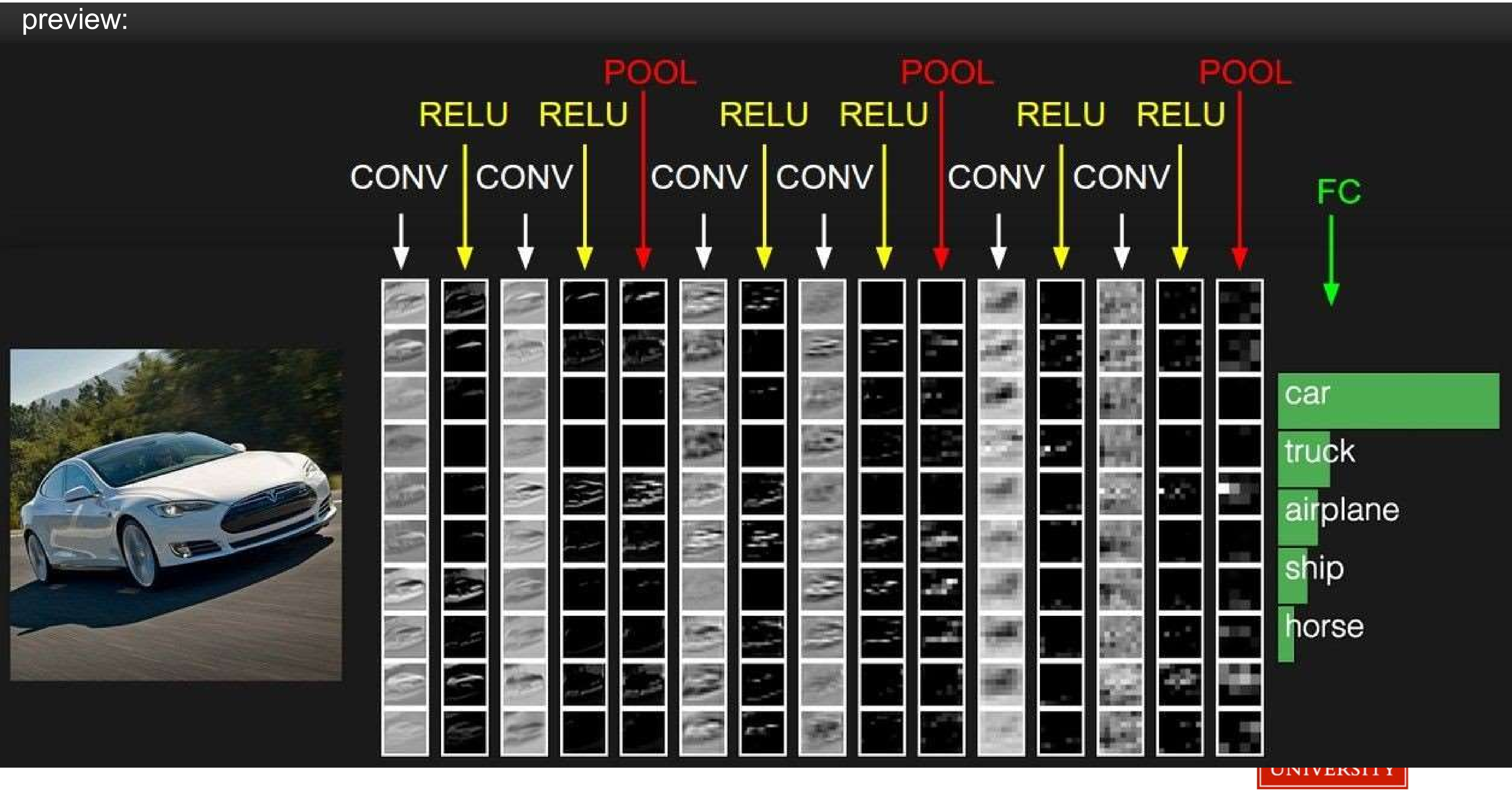
Captures (some) “natural” invariances

- Shifting input image one pixel to the right shifts creates a hidden shifts the hidden unit “image”

Convolutional Neural Networks

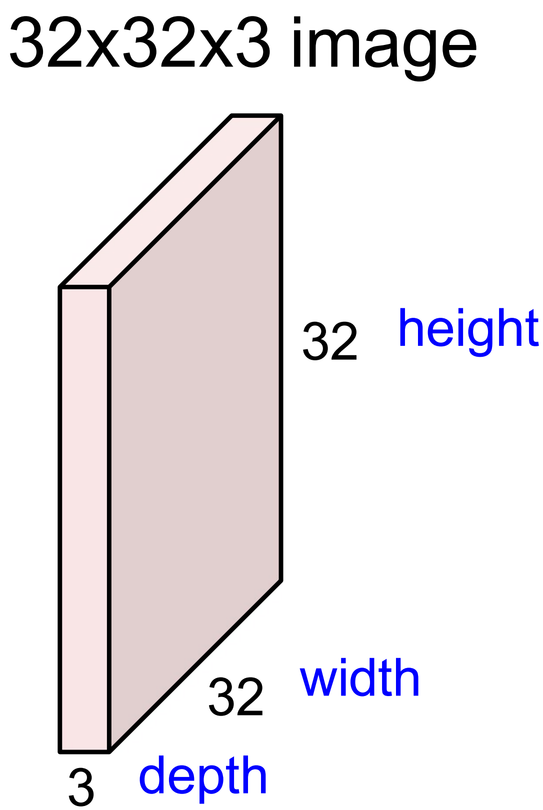


[LeNet-5, LeCun 1980]



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

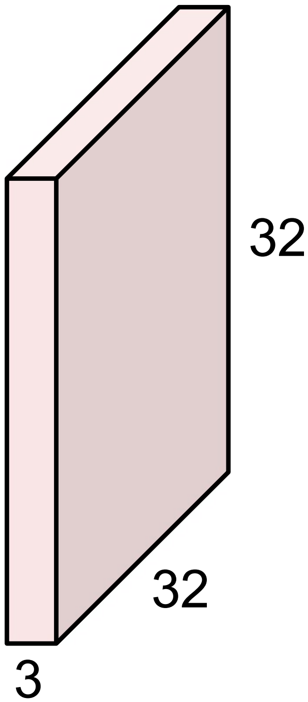
Convolution Layer



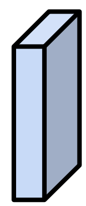
Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Convolution Layer

32x32x3 image



5x5x3 filter



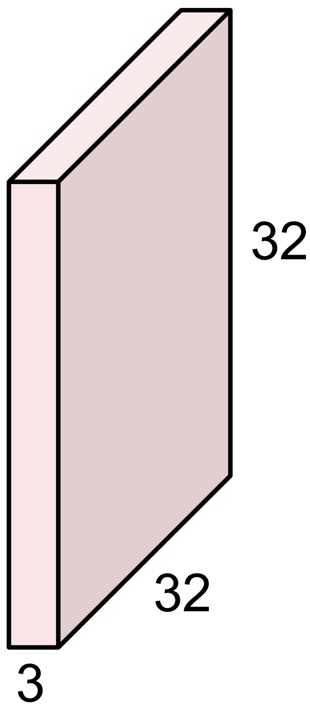
Convolve the filter with the image
i.e. "slide over the image spatially, computing dot products"

Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

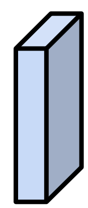
Convolution Layer

Filters always extend the full depth of the input volume

32x32x3 image



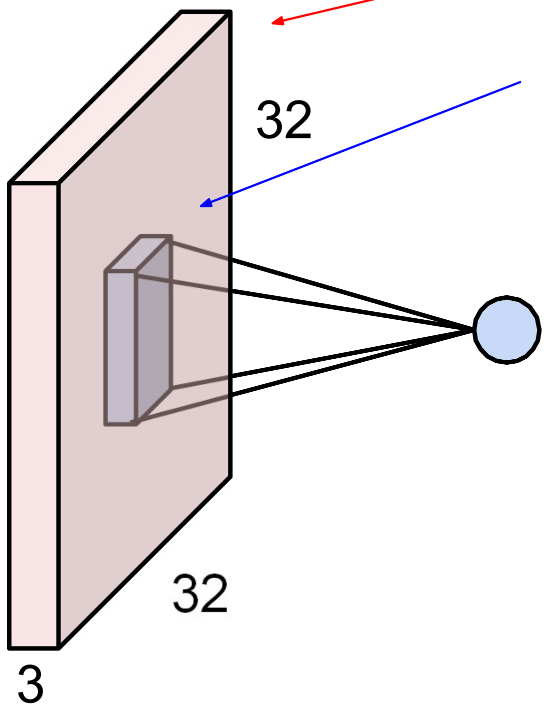
5x5x3 filter



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

Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Convolution Layer



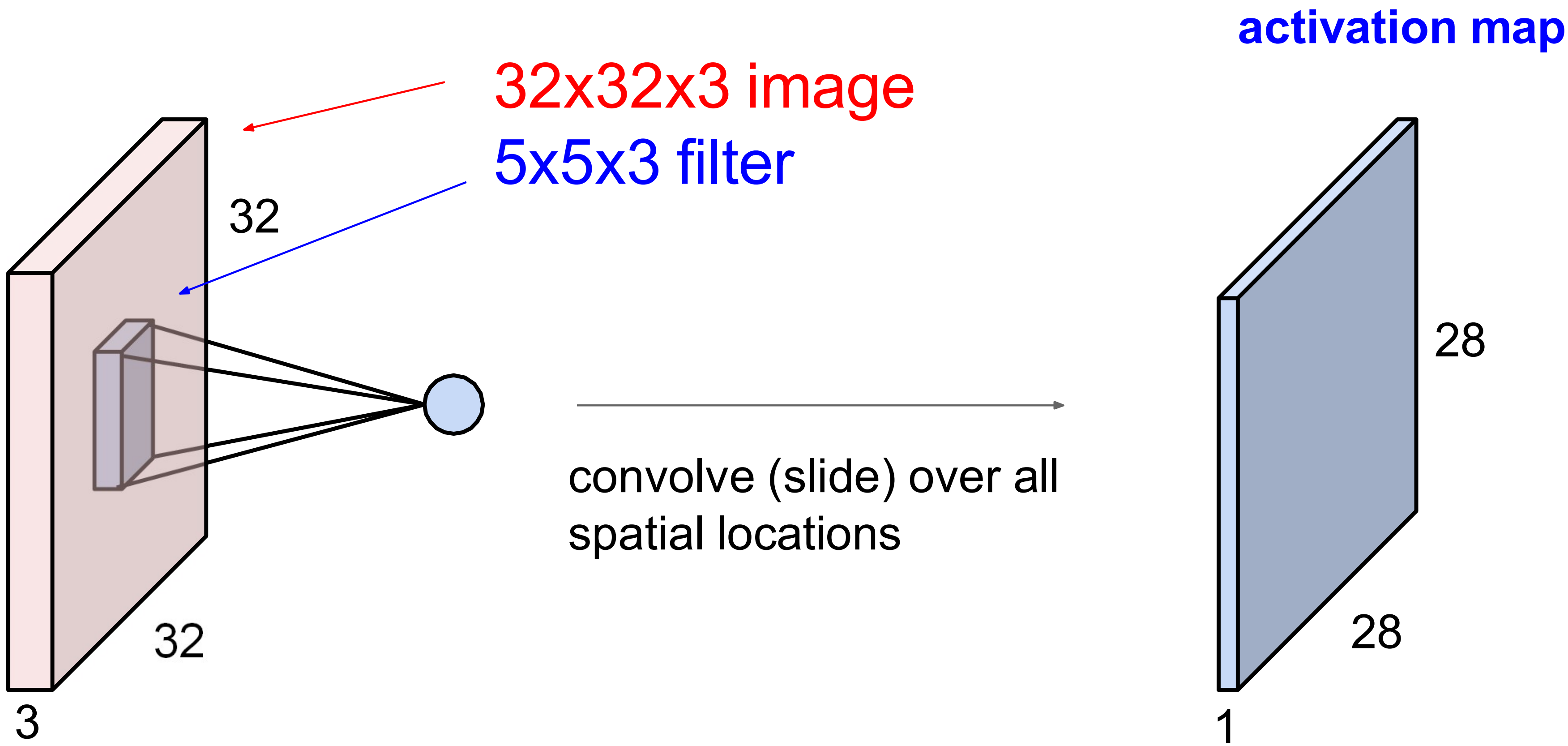
32x32x3 image

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)

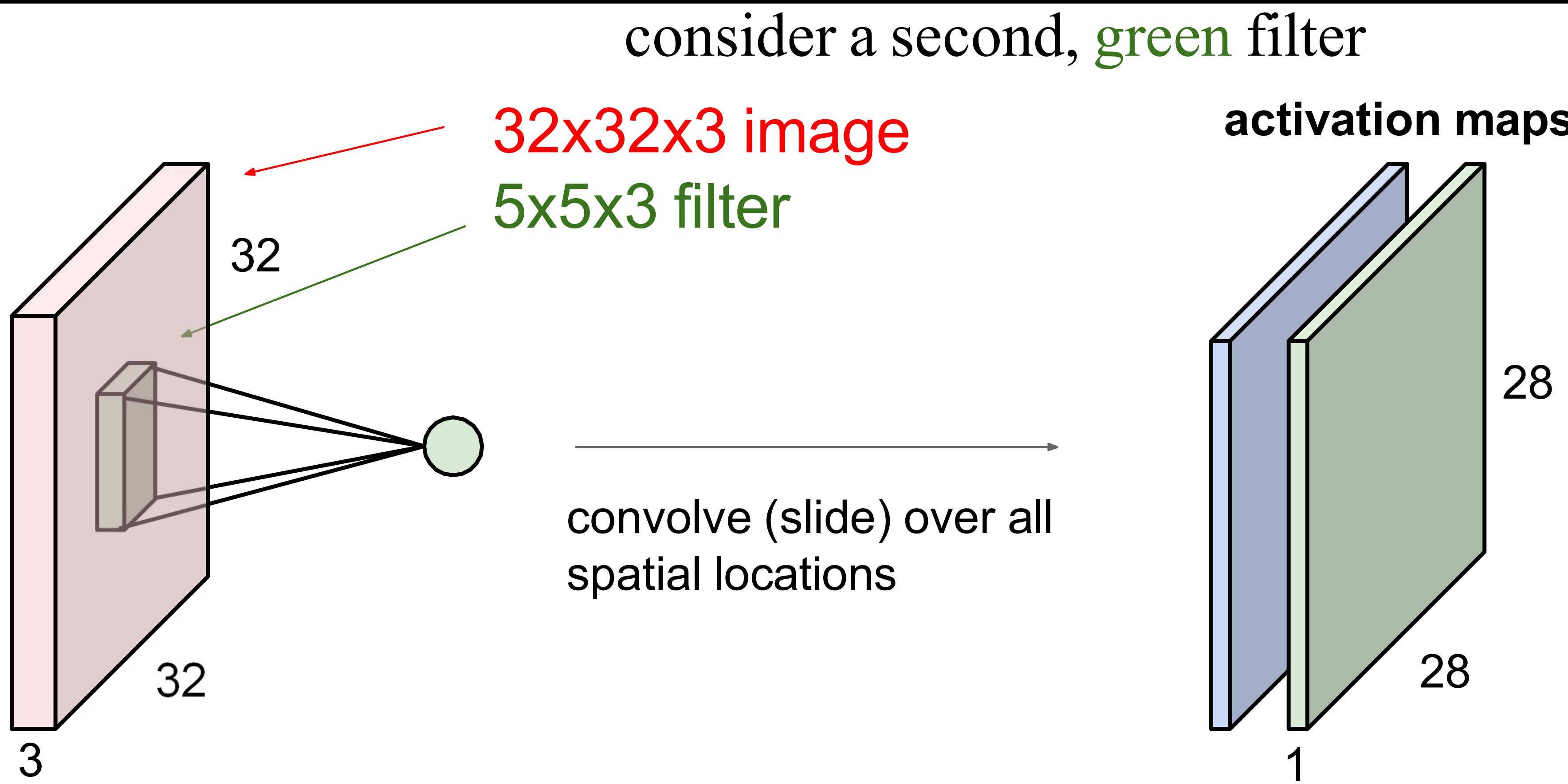
$$w^T x + b$$

Convolution Layer



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

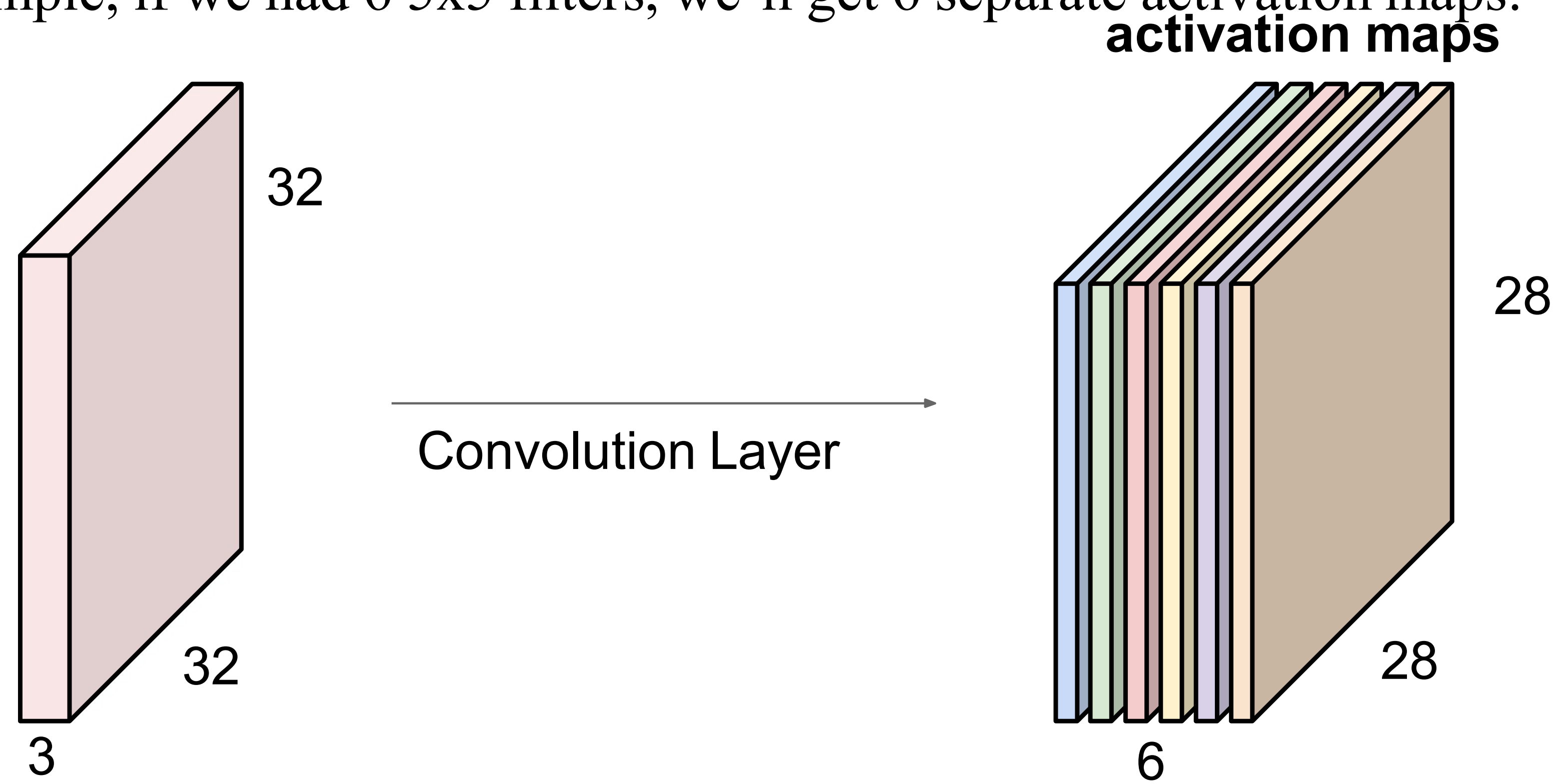
Convolution Layer



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Convolution Layer

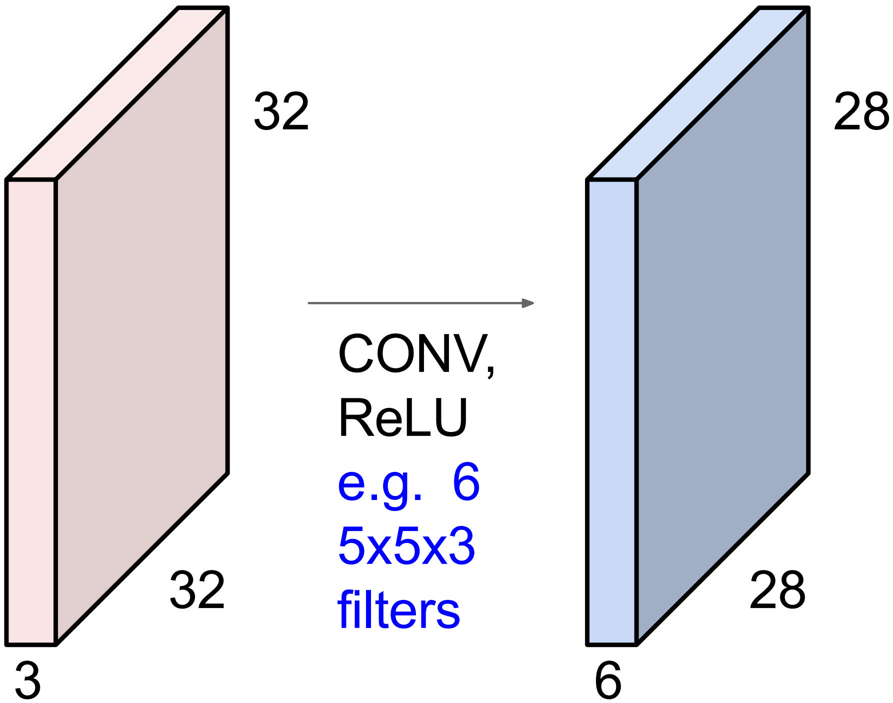
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

Convolution Layer

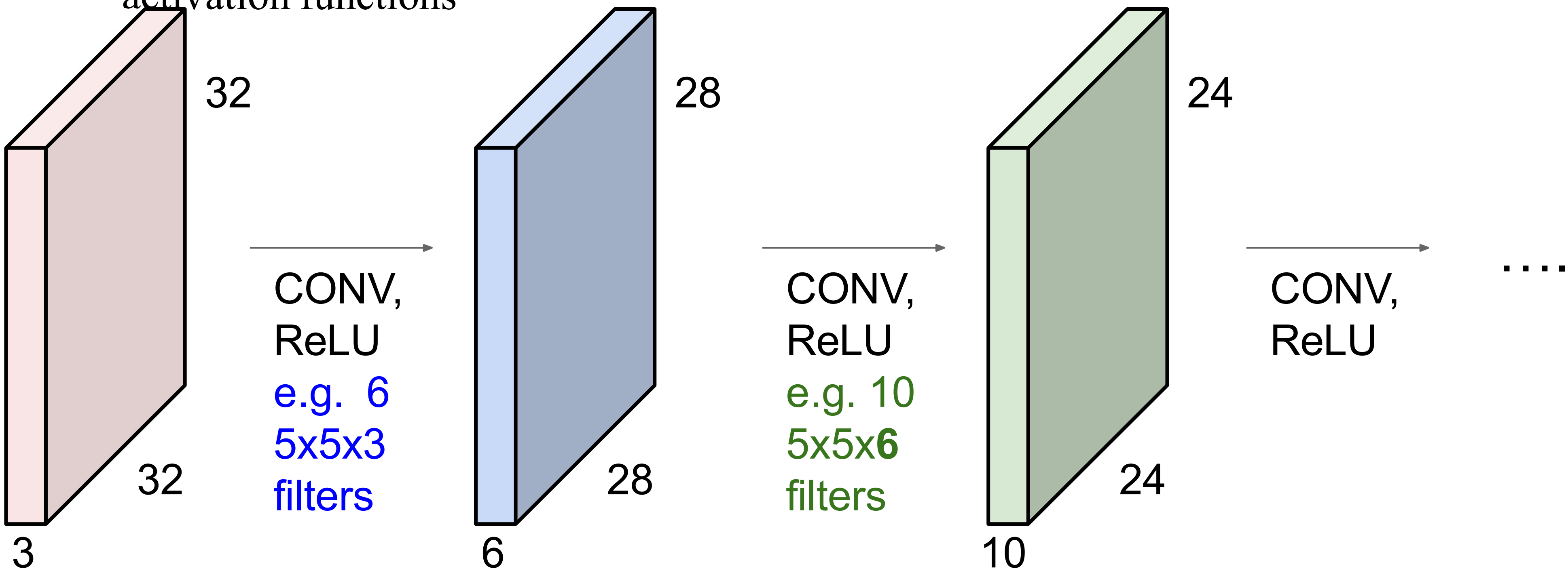
Preview: ConvNet is a sequence of Convolution Layers, interspersed with activation functions



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Convolution Layer

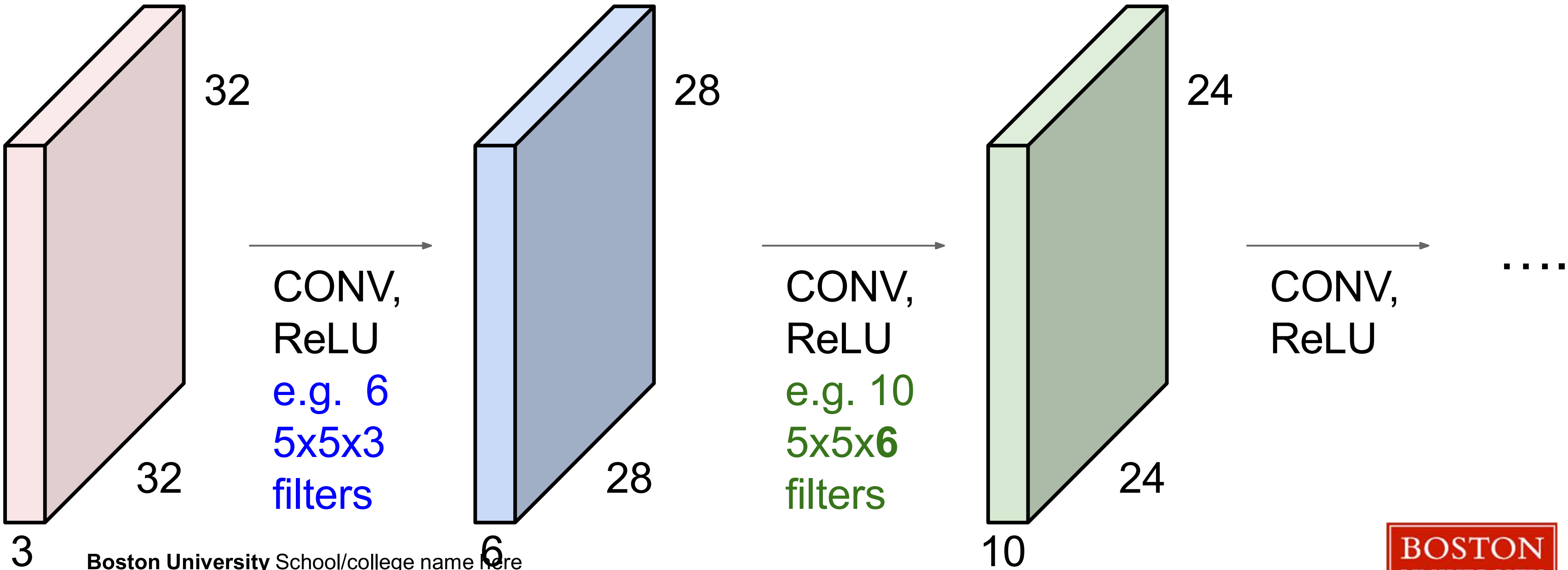
Preview: ConvNet is a sequence of Convolutional Layers, interspersed with activation functions



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

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.

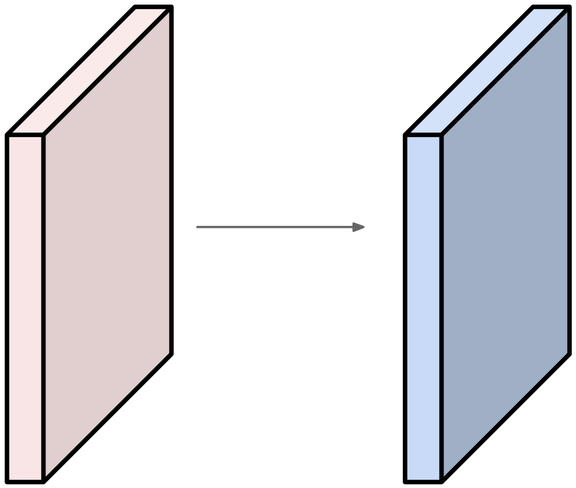


Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Examples time:

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

Output volume size: ?

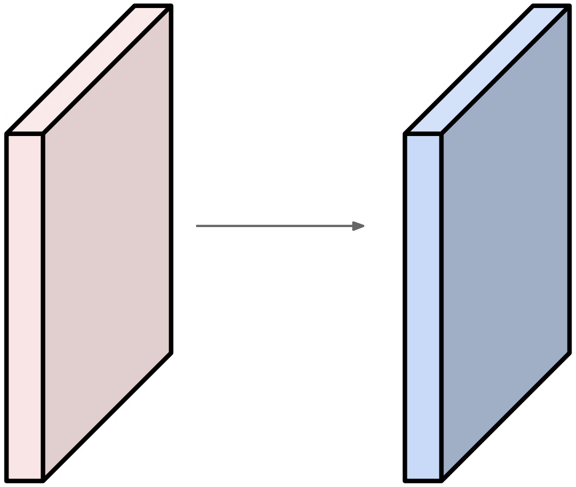


Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Examples time:

Input volume: **32x32x3**

10 5x5 filters with stride 1, pad 2



Output volume size:

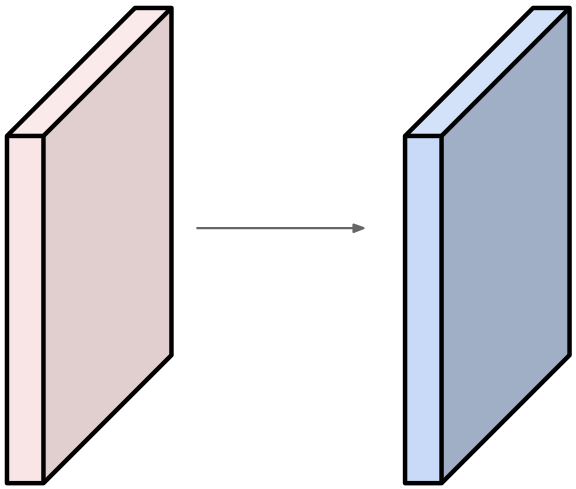
(32+2*2-5)/1+1 = 32 spatially, so

32x32x10

Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Examples time:

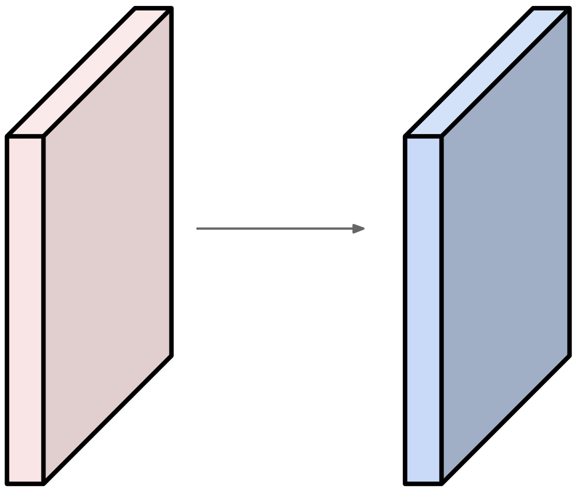
Input volume: **32x32x3**
10 5x5 filters with stride 1, pad 2



Number of parameters in this layer?

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**

Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Convolutions in image processing

Convolutions (typically with *prespecified* filters) are a common operation in many computer vision applications: convolution networks just move to *learned* filters

Original image x

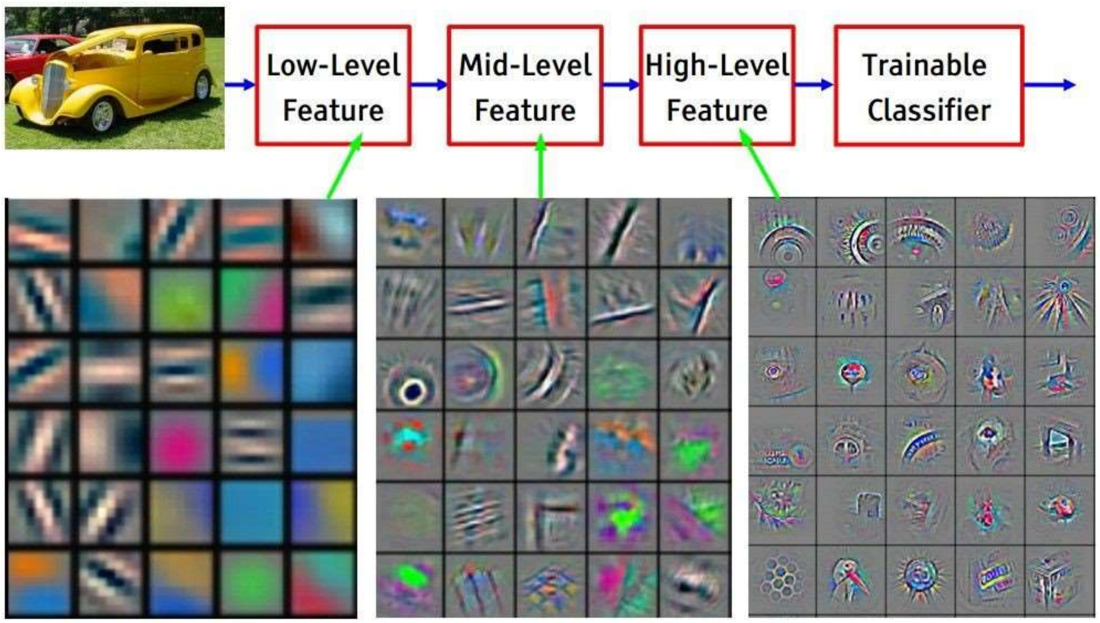
Gaussian blur

Image gradient

$$x * \begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 4 & 4 & 1 \end{bmatrix} / 273$$

$$\left(\left(x * \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \right)^2 + \left(x * \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \right)^2 \right)^{\frac{1}{2}}$$

Preview



Feature visualization of convolutional net trained on ImageNet from [Zeiler & Fergus 2013]

[From recent Yann LeCun slides]

Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Convolutions in detail

- Convolutions are a basic primitive in many computer vision and image processing algorithms

- Idea is to "slide" the weights $h \times h$ weight m (called a filter, with kernel size h) over the image to produce a new image, written $y = x * m$

$$\begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & y_{33} \end{bmatrix}$$

Slide credit: J. Zico Kolter and Tianqi Chen

Convolutions in detail

- Convolutions are a basic primitive in many computer vision and image processing algorithms

- Idea is to "slide" the weights $h \times h$ weight m (called a filter, with kernel size h) over the image to produce a new image, written $y = x * m$

$$\begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & 33 \end{bmatrix}$$

$$y_{11} = z_{11}w_{11} + z_{12}w_{12} + z_{13}w_{13} + z_{21}w_{21} + \ldots$$

Slide credit: J. Zico Kolter and Tianqi Chen

Convolutions in detail

- Convolutions are a basic primitive in many computer vision and image processing algorithms

- Idea is to "slide" the weights $h \times h$ weight m (called a filter, with kernel size h) over the image to produce a new image, written $y = x * m$

$$\begin{array}{|c|c|c|c|c|}\hline z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ \hline z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ \hline z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\ \hline z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\ \hline z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \\ \hline \end{array} * \begin{array}{|c|c|c|}\hline w_{11} & w_{12} & w_{13} \\ \hline w_{21} & w_{22} & w_{23} \\ \hline w_{31} & w_{32} & w_{33} \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline y_{11} & y_{12} & y_{13} \\ \hline y_{21} & y_{22} & y_{23} \\ \hline y_{31} & y_{32} & 33 \\ \hline \end{array}$$

$$y_{12} = z_{12}w_{11} + z_{13}w_{12} + z_{14}w_{13} + z_{22}w_{21} + \ldots$$

BOSTON UNIVERSITY

Slide credit: J. Zico Kolter and Tianqi Chen

Convolutions in detail

- Convolutions are a basic primitive in many computer vision and image processing algorithms

- Idea is to "slide" the weights $h \times h$ weight m (called a filter, with kernel size h) over the image to produce a new image, written $y = x * m$

$$\begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & 33 \end{bmatrix}$$

$$y_{13} = z_{13}w_{11} + z_{14}w_{12} + z_{15}w_{13} + z_{23}w_{21} + \ldots$$

Slide credit: J. Zico Kolter and Tianqi Chen

Convolutions in detail

- Convolutions are a basic primitive in many computer vision and image processing algorithms

- Idea is to “slide” the weights $h \times h$ weight m (called a filter, with kernel size h) over the image to produce a new image, written $y = x * m$

$$
\begin{array}{|c|c|c|c|c|}
\hline
z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\
\hline
z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\
\hline
z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\
\hline
z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\
\hline
z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \\
\hline
\end{array}
\; * \;
\begin{array}{|c|c|c|}
\hline
w_{11} & w_{12} & w_{13} \\
\hline
w_{21} & w_{22} & w_{23} \\
\hline
w_{31} & w_{32} & w_{33} \\
\hline
\end{array}
\; = \;
\begin{array}{|c|c|c|}
\hline
y_{11} & y_{12} & y_{13} \\
\hline
y_{21} & y_{22} & y_{23} \\
\hline
y_{31} & y_{32} & 33 \\
\hline
\end{array}
$$

$$y_{21} = z_{21}w_{11} + z_{22}w_{12} + z_{23}w_{13} + z_{31}w_{21} + \ldots$$

Slide credit: J. Zico Kolter and Tianqi Chen

Convolutions in detail

- Convolutions are a basic primitive in many computer vision and image processing algorithms

- Idea is to “slide” the weights $h \times h$ weight m (called a filter, with kernel size h) over the image to produce a new image, written $y = x * m$

$$
\begin{bmatrix}
z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\
z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\
z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\
z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\
z_{51} & z_{52} & z_{53} & z_{54} & z_{55}
\end{bmatrix}
*
\begin{bmatrix}
w_{11} & w_{12} & w_{13} \\
w_{21} & w_{22} & w_{23} \\
w_{31} & w_{32} & w_{33}
\end{bmatrix}
=
\begin{bmatrix}
y_{11} & y_{12} & y_{13} \\
y_{21} & y_{22} & y_{23} \\
y_{31} & y_{32} & 33
\end{bmatrix}
$$

$$y_{22} = z_{22}w_{11} + z_{23}w_{12} + z_{24}w_{13} + z_{32}w_{21} + \ldots$$

Slide credit: J. Zico Kolter and Tianqi Chen

Convolutions in detail

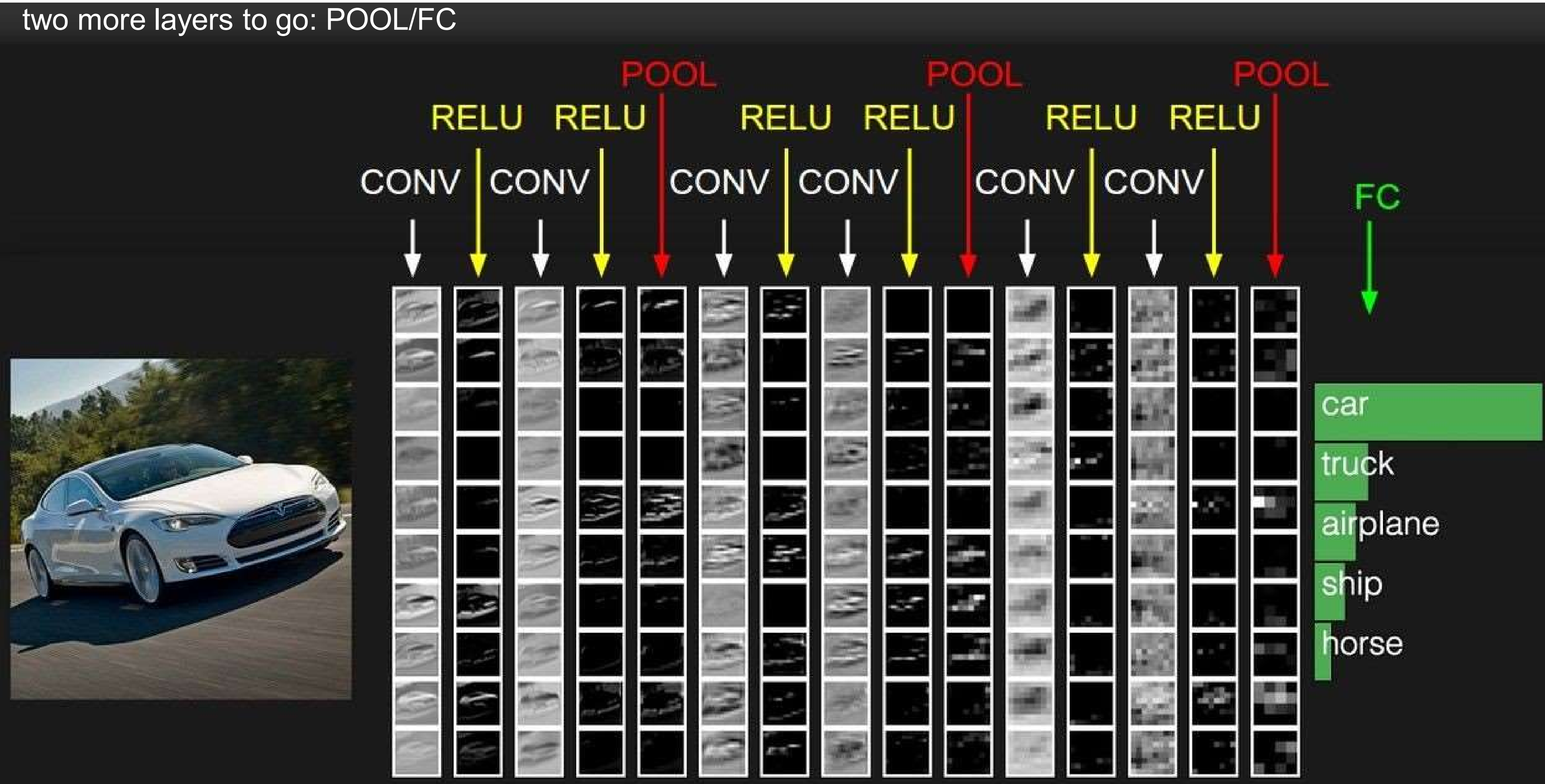
- Convolutions are a basic primitive in many computer vision and image processing algorithms

- Idea is to “slide” the weights $h \times h$ weight m (called a filter, with kernel size h) over the image to produce a new image, written $y = x * m$

$$
\begin{bmatrix} z_{11} & z_{12} & z_{13} & z_{14} & z_{15} \\ z_{21} & z_{22} & z_{23} & z_{24} & z_{25} \\ z_{31} & z_{32} & z_{33} & z_{34} & z_{35} \\ z_{41} & z_{42} & z_{43} & z_{44} & z_{45} \\ z_{51} & z_{52} & z_{53} & z_{54} & z_{55} \end{bmatrix} * \begin{bmatrix} w_{11} & w_{12} & w_{13} \\ w_{21} & w_{22} & w_{23} \\ w_{31} & w_{32} & w_{33} \end{bmatrix} = \begin{bmatrix} y_{11} & y_{12} & y_{13} \\ y_{21} & y_{22} & y_{23} \\ y_{31} & y_{32} & 33 \end{bmatrix}
$$

$$y_{23} = z_{23}w_{11} + z_{24}w_{12} + z_{25}w_{13} + z_{33}w_{21} + \ldots$$

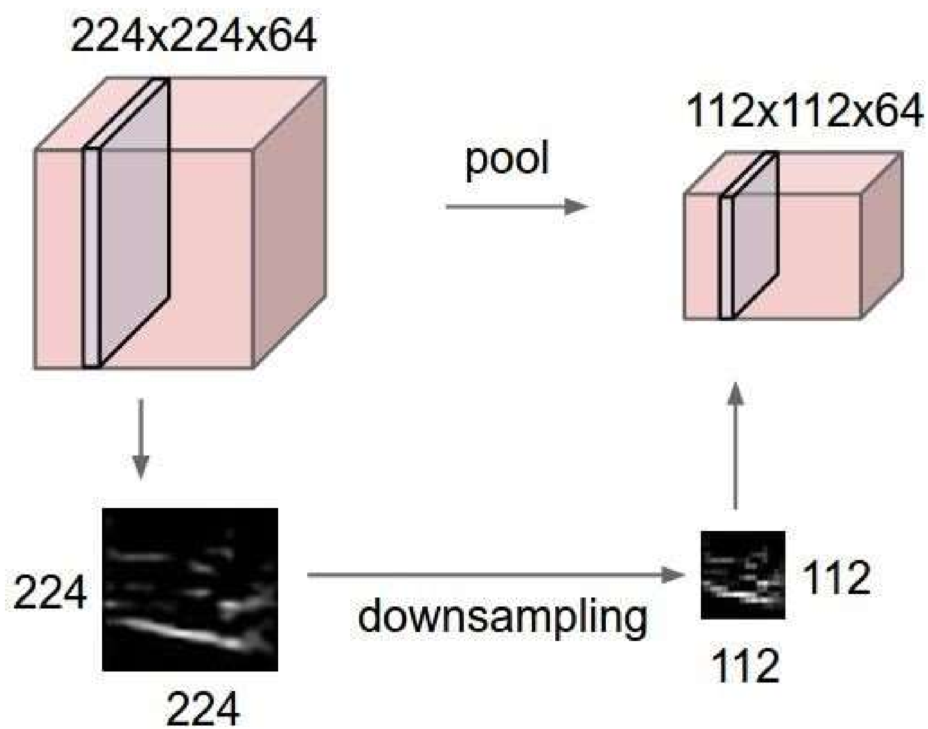
Slide credit: J. Zico Kolter and Tianqi Chen



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

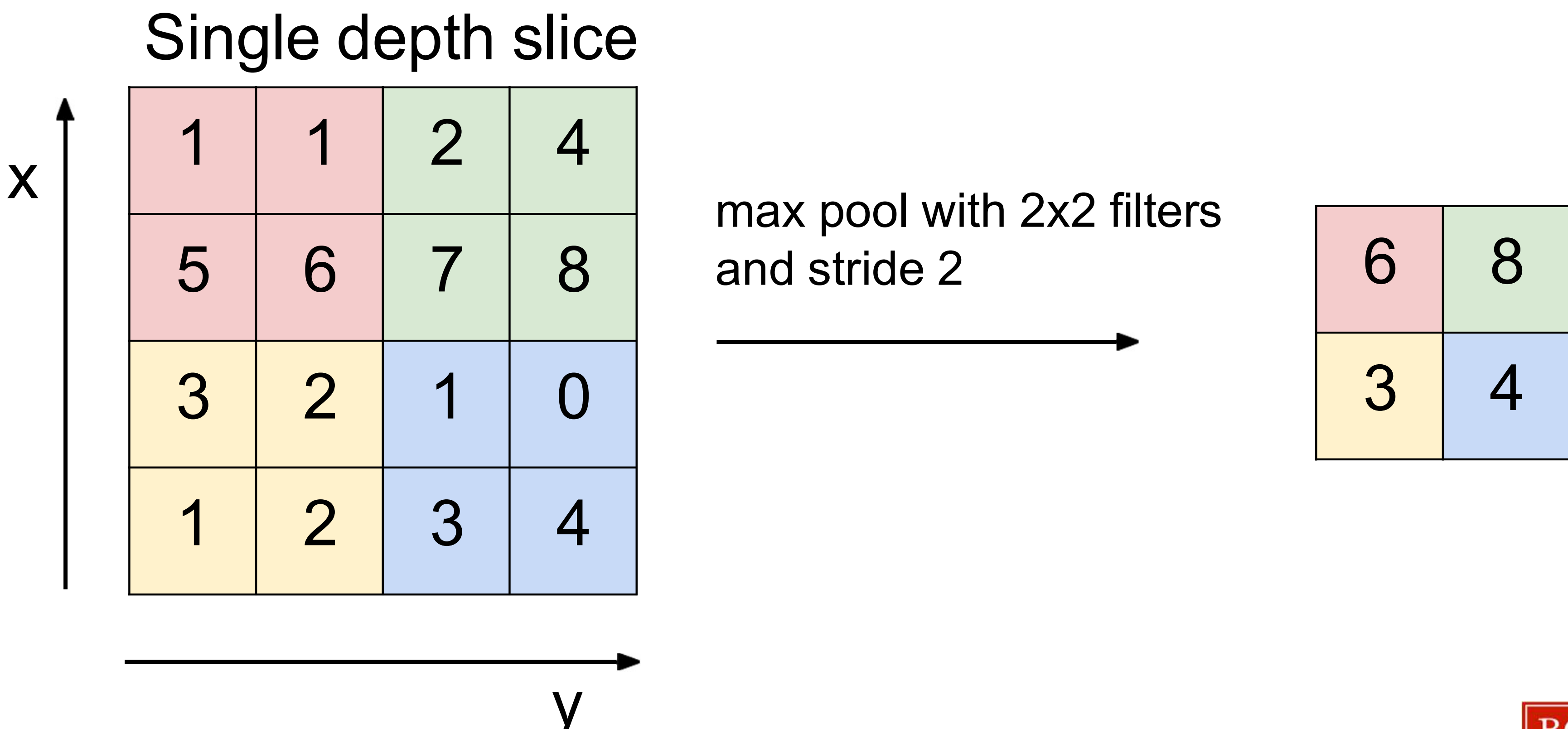
Pooling layer

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



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

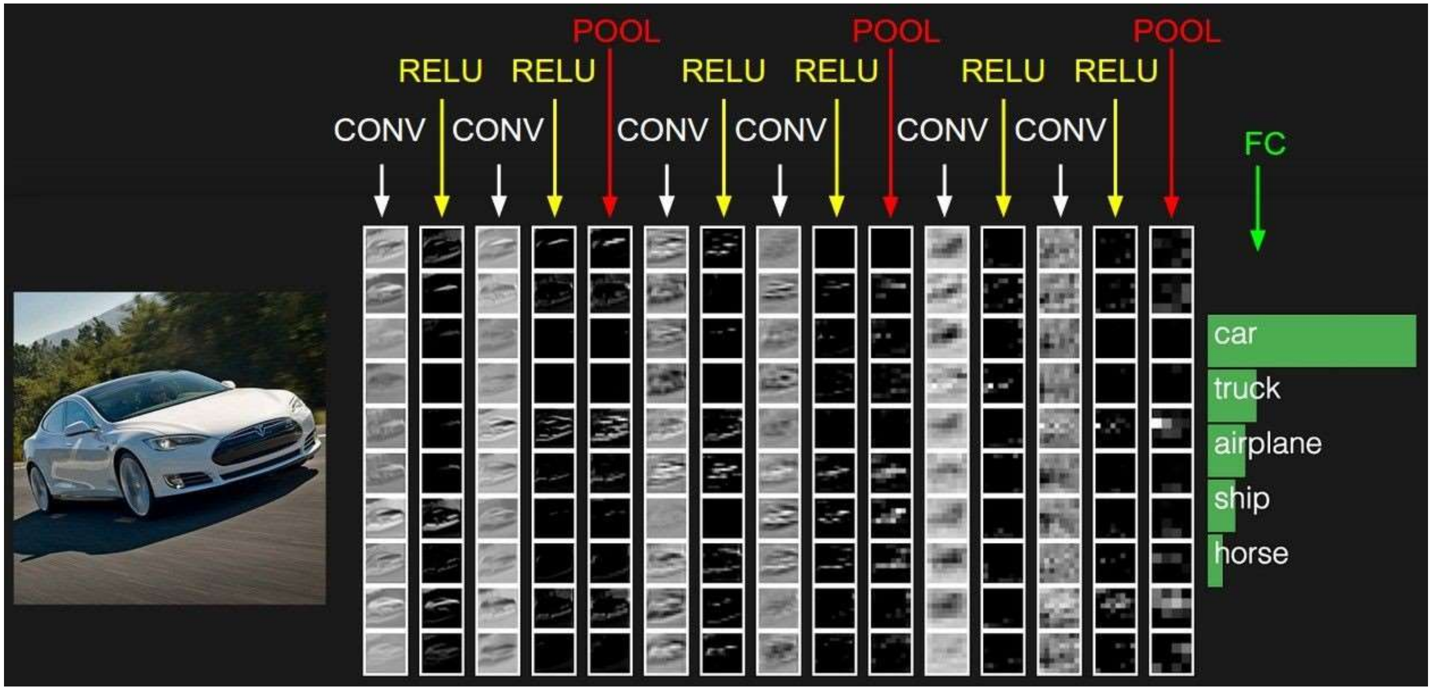
MAX POOLING



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Fully Connected Layer (FC layer)

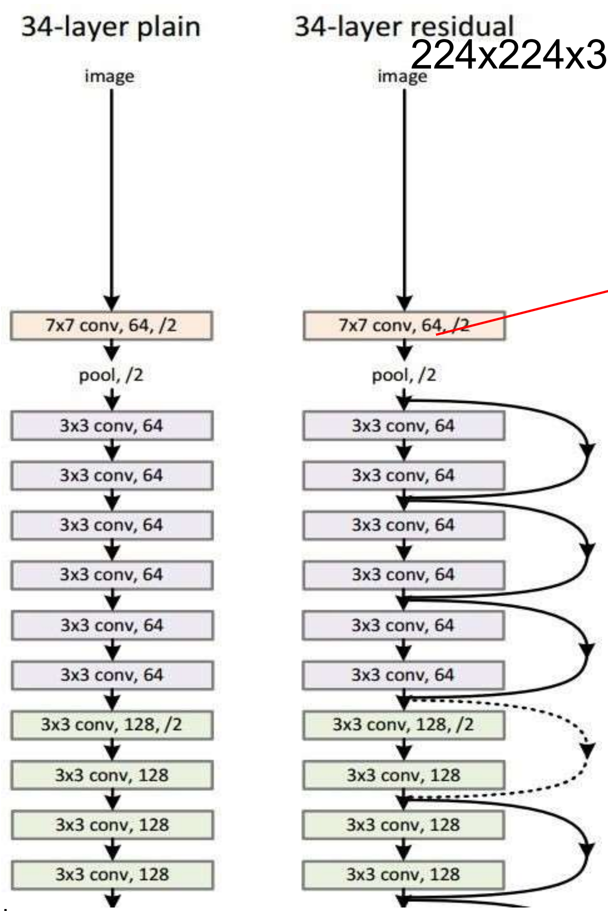
Contains neurons that connect to the entire input volume, as in ordinary Neural Networks



Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

Case Study: ResNet

[He et al., 2015]

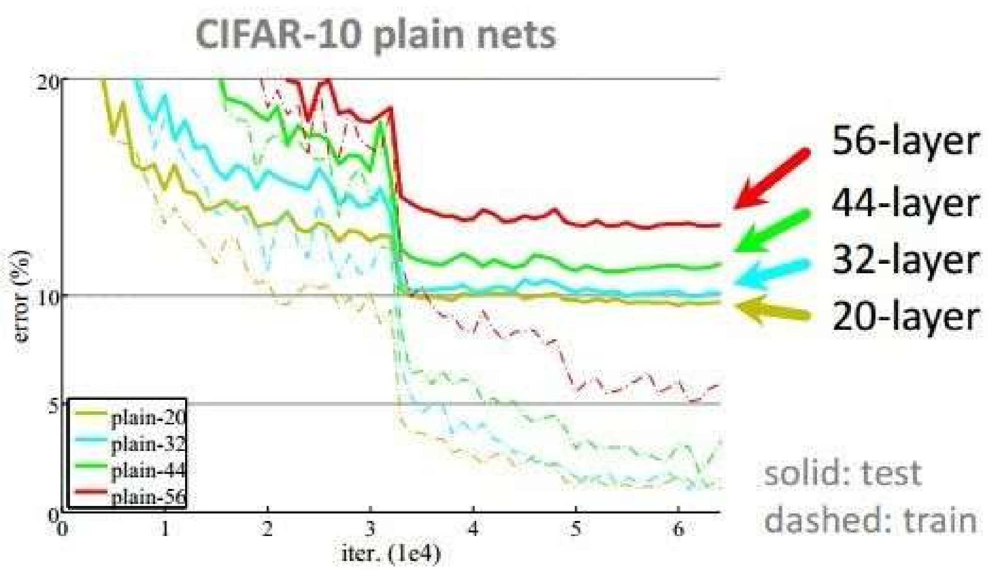


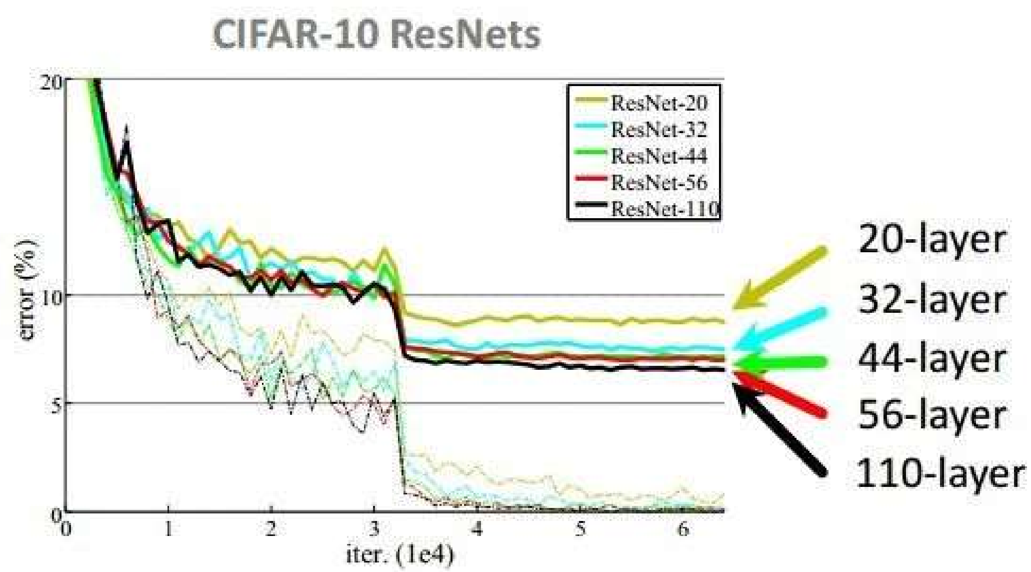
spatial dimension
only 56x56!

BOSTON UNIVERSITY

Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

CIFAR-10 experiments





Slide credit: Fei-Fei Li, Andrej Karpathy and Justin Johnson

The End

Thanks for your attention.

I would be glad if you have any question.

BOSTON UNIVERSITY