UNIVERSITY OF NORTH TEXAS

CSCE 5218 & 4930 Deep Learning

Convolutional Neural Networks

Slides adapted from Adriana Kovashka

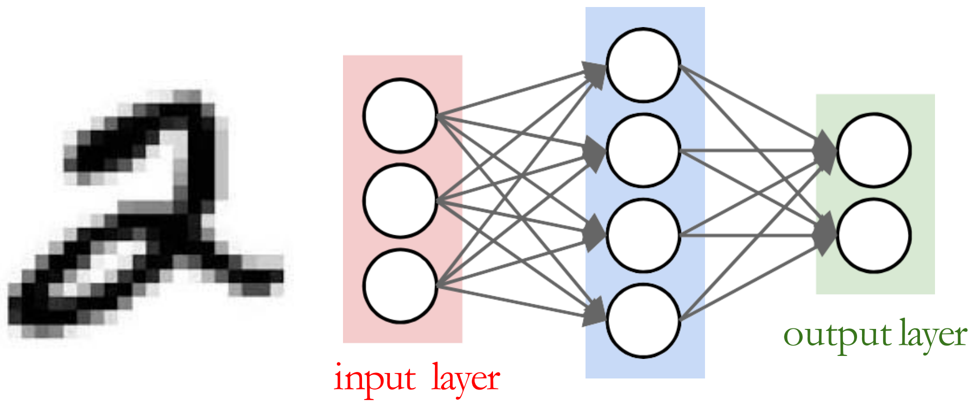
Plan for this lecture

- Motivation: Scanning for patterns
- Convolutional network operations
- Common architectures
- Visualizing convolutional networks
- Applications in computer vision

Plan for this lecture

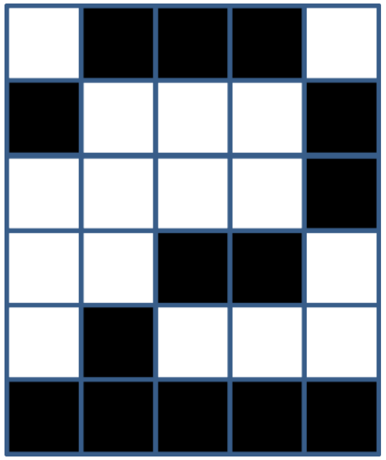
- Motivation: Scanning for patterns
- Convolutional network operations
- Common architectures
- Visualizing convolutional networks
- Applications in computer vision

Neural networks so far

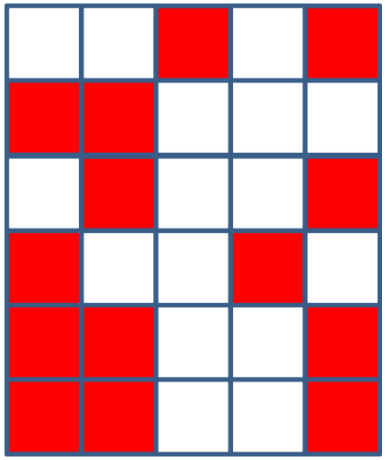


- Can recognize patterns in data (e.g. digits)

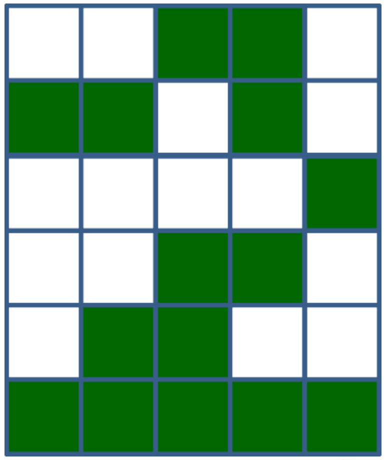
The weights look for patterns



$$y = \begin{cases} 1 \; \textit{if} \sum w_i x_i \geq T \\ 0 \; \textit{else} \end{cases}$$



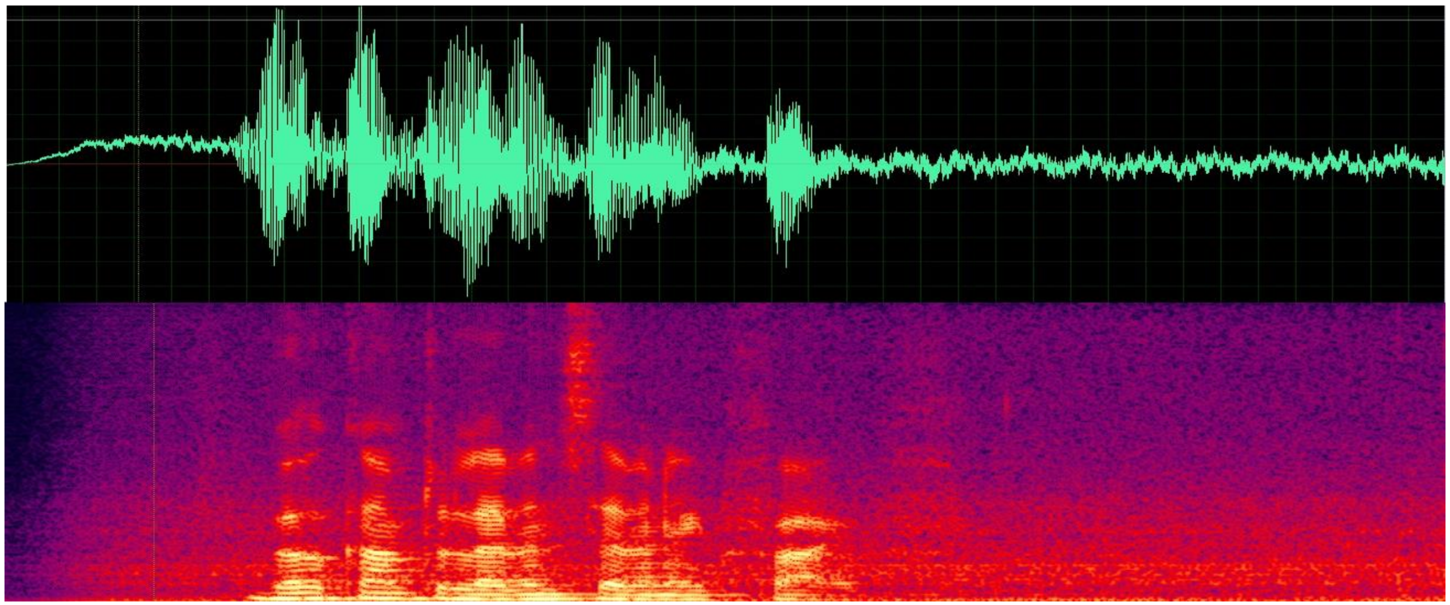
Correlation = 0.57



Correlation = 0.82

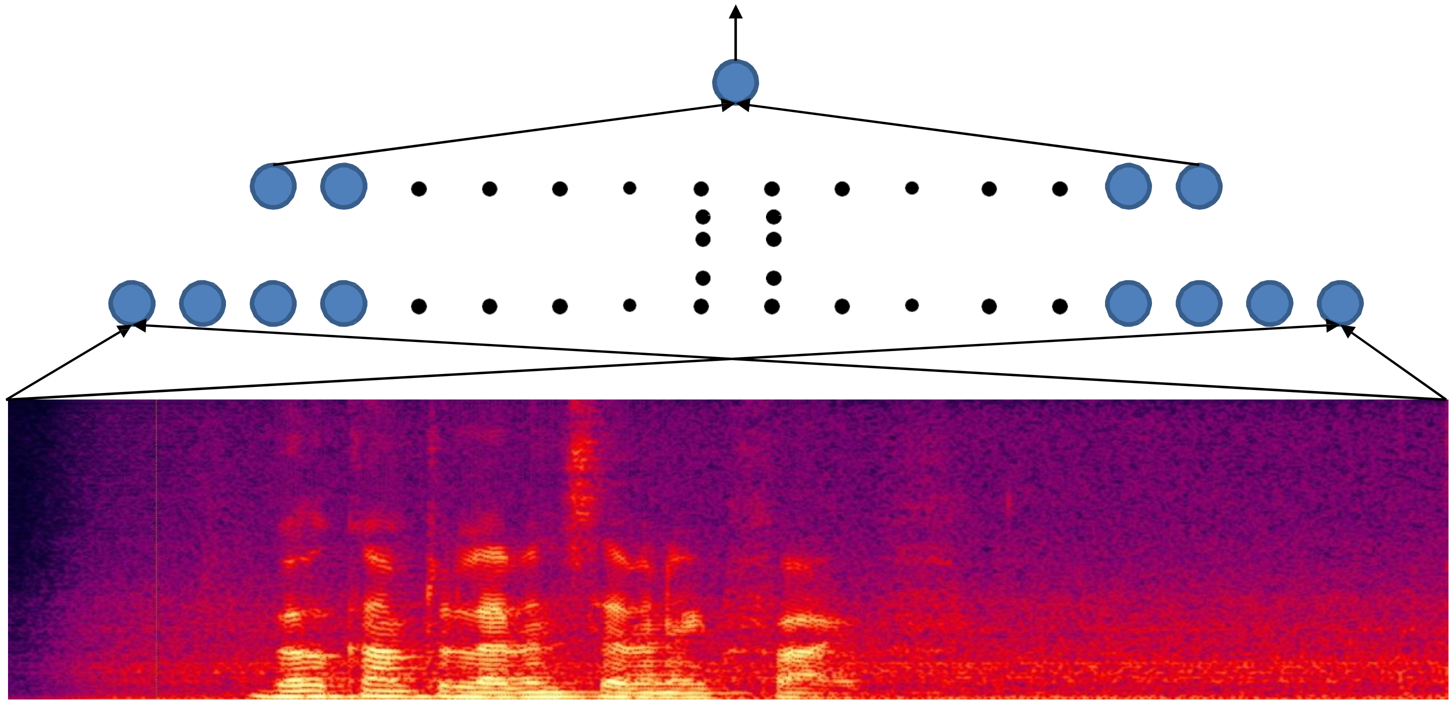
- The green pattern looks more like the weights pattern (black) than the red pattern
  - The green pattern is more ***correlated*** with the weights

A problem



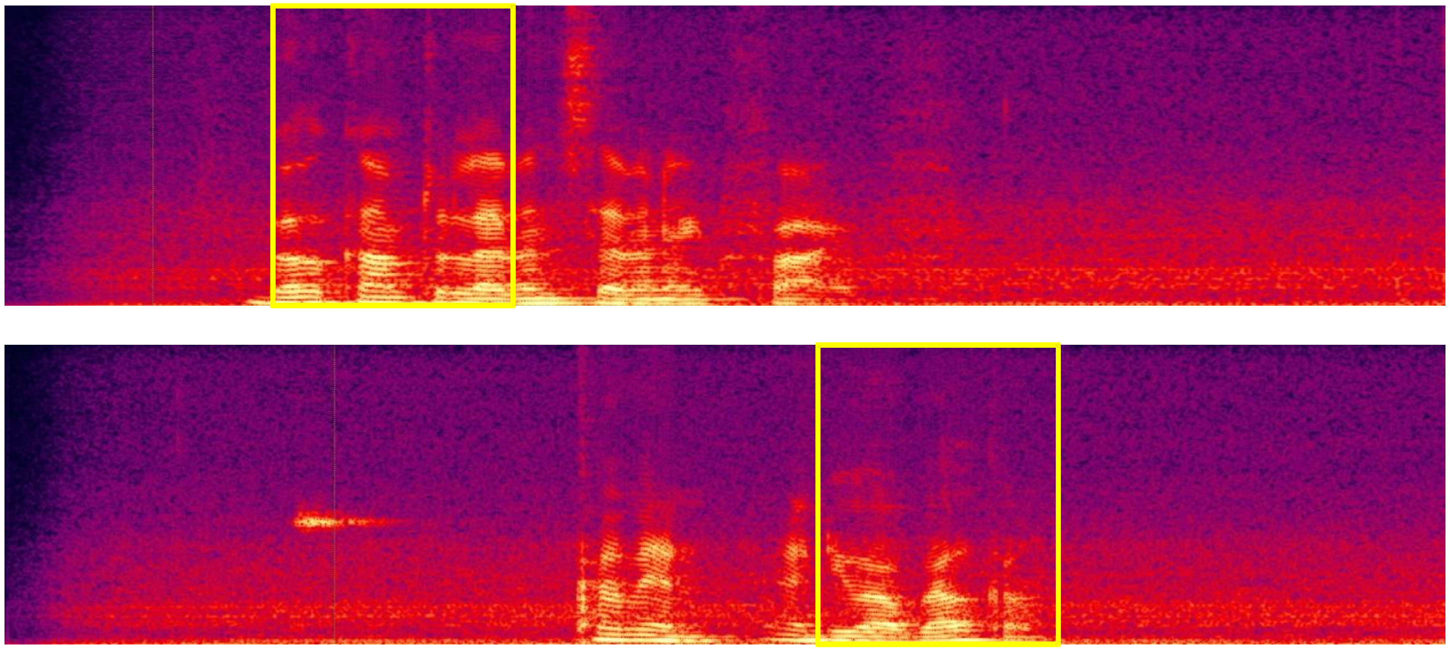
- Does this signal contain the word “Welcome”?
- Compose a NN for this problem
  - Assuming all recordings are exactly the same length

Finding a Welcome



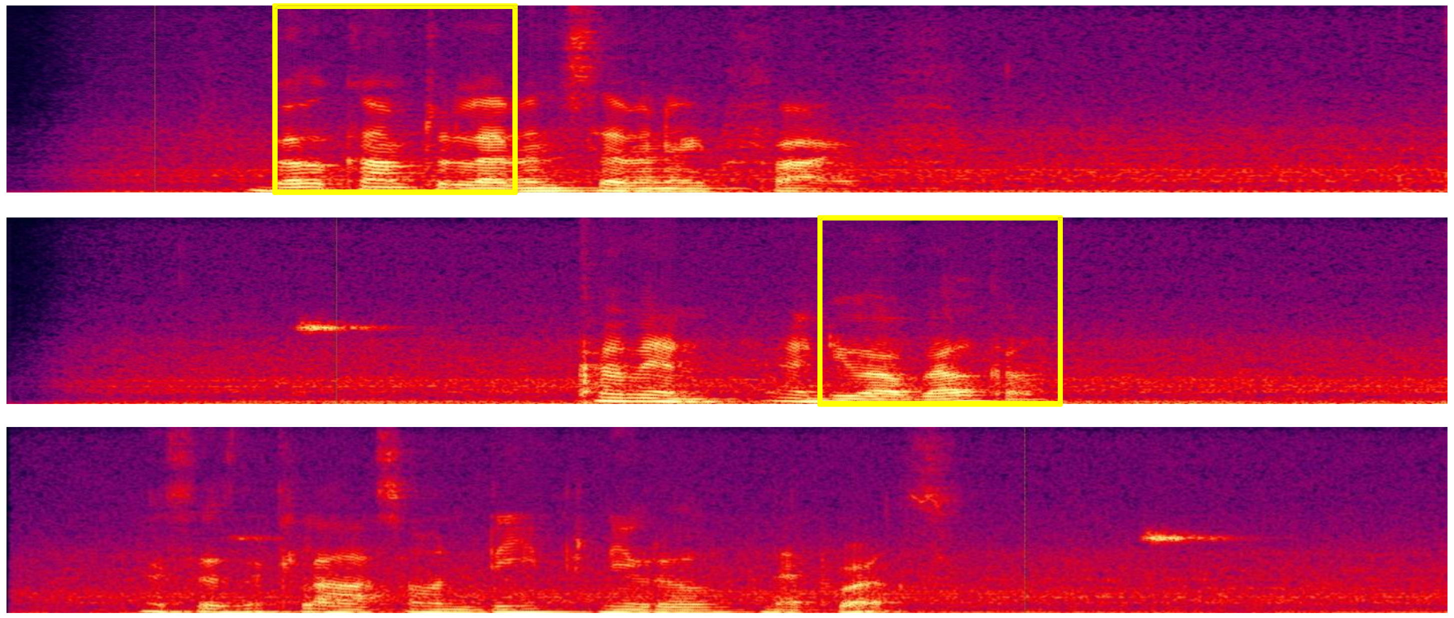
- Trivial solution: Train a NN for the entire recording

Finding a Welcome



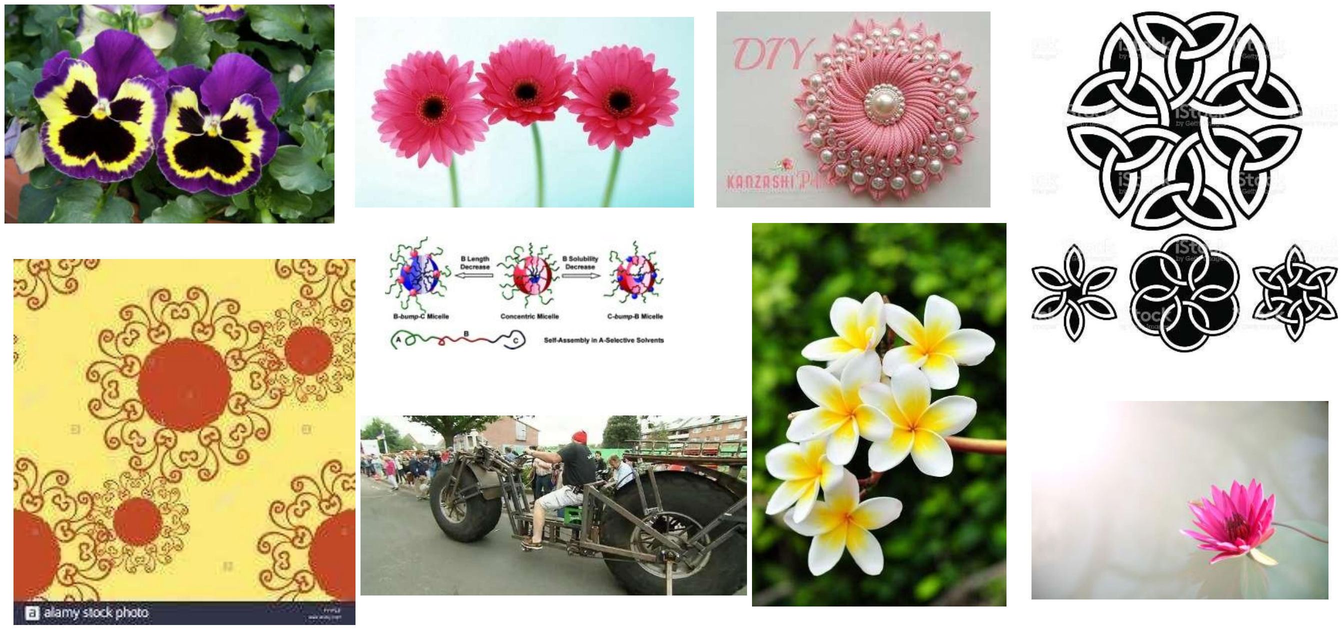
- Problem with trivial solution: Network that finds a "welcome" in the top recording will not find it in the lower one
  - Unless trained with both
  - Will require a very large network and a large amount of training data to cover every case

Finding a Welcome



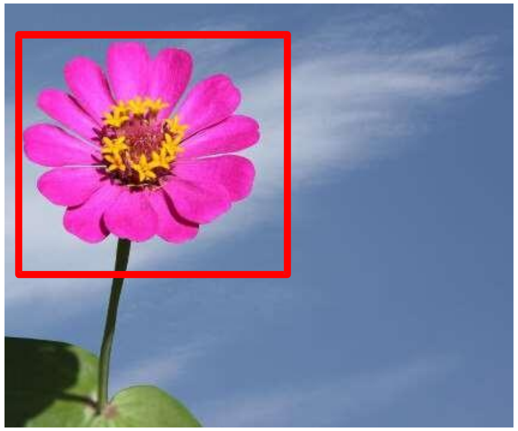
- Need a ***simple*** network that will fire regardless of the location of "Welcome"
  - – and not fire when there is none

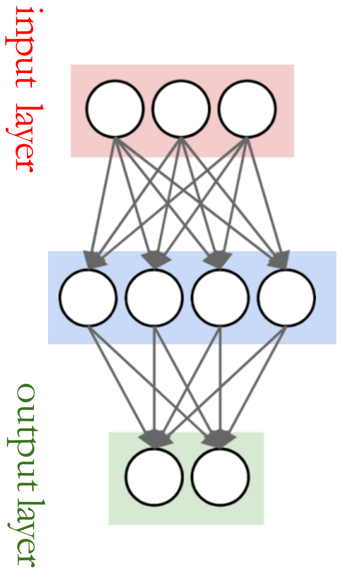
Flower

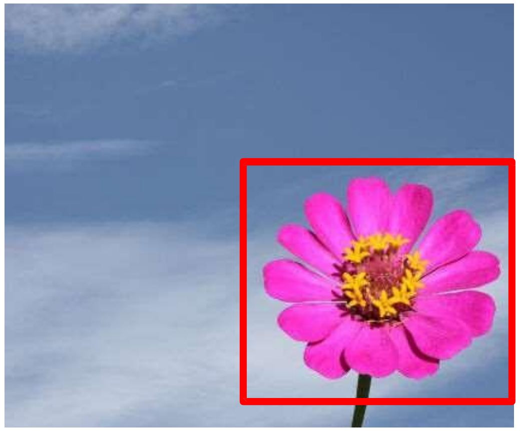


- Is there a flower in any of these images?

Flower

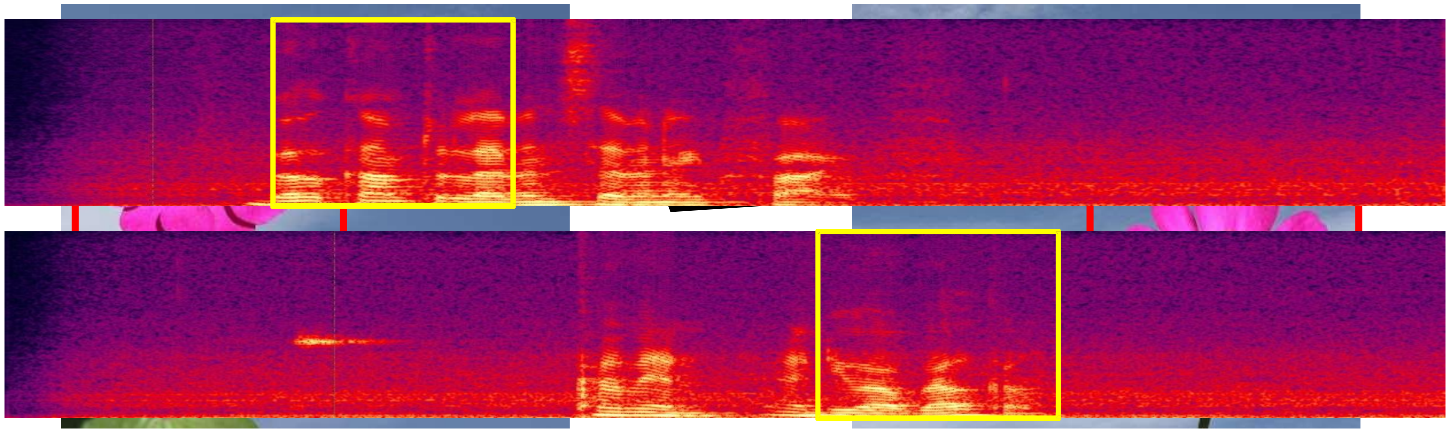






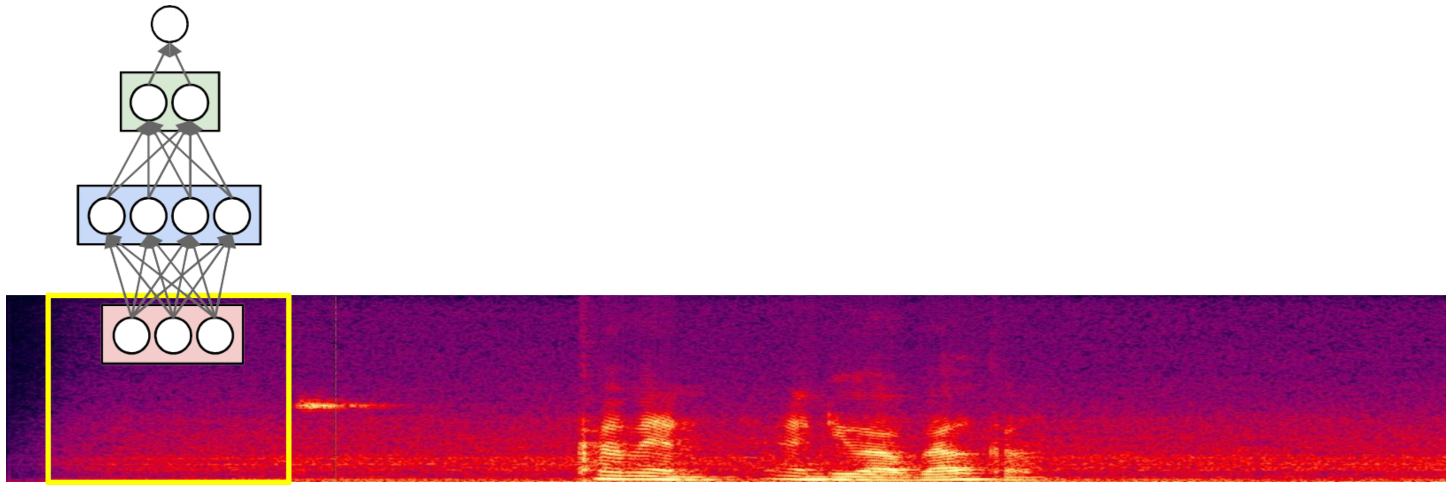
- Will a NN that recognizes the left image as a flower also recognize the one on the right as a flower?
- Need a network that will "fire" regardless of the precise location of the target object

The need for *shift invariance*



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

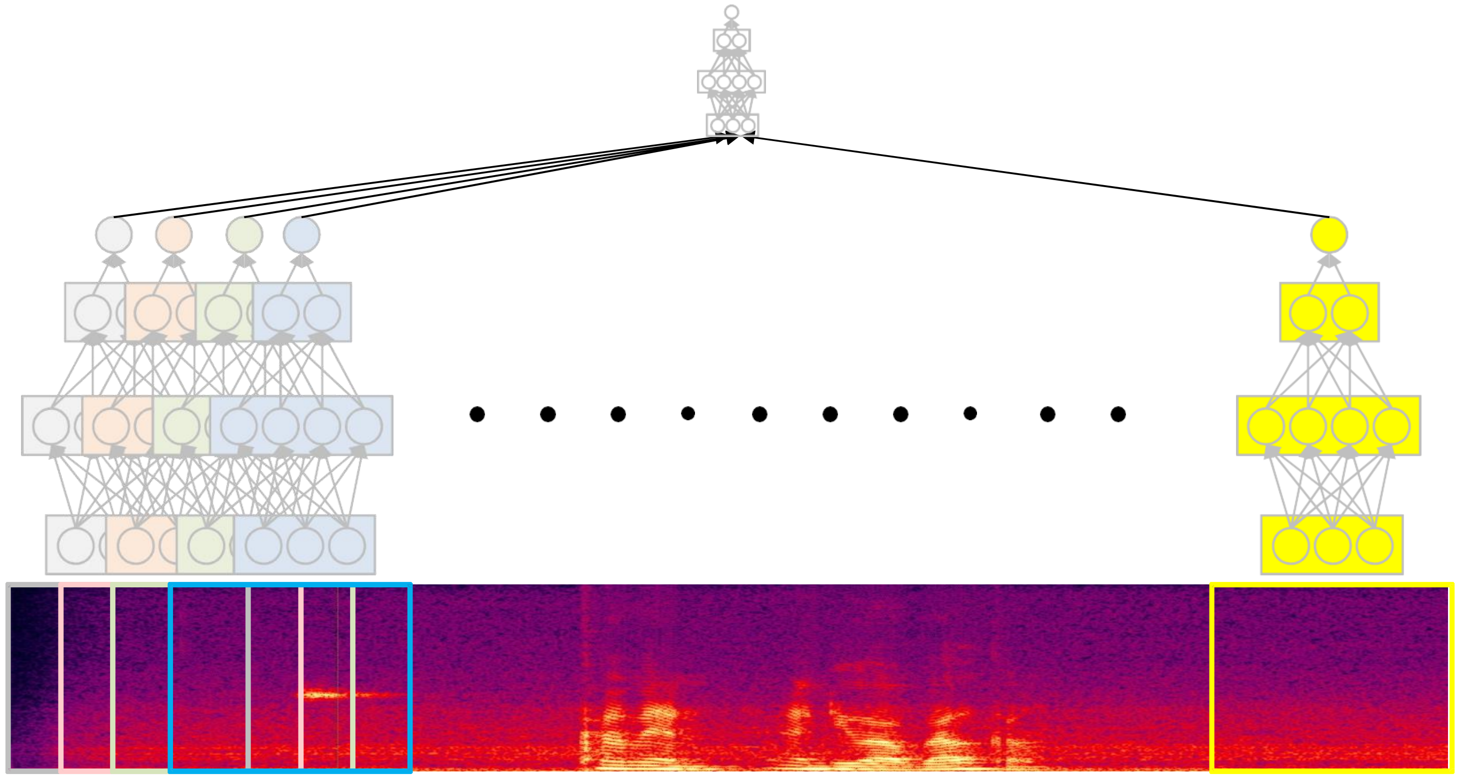
Solution: Scan



- Scan for the target word
  - The audio signals in a "window" are input to a "welcome-detector" NN

Adapted from Bhiksha Raj

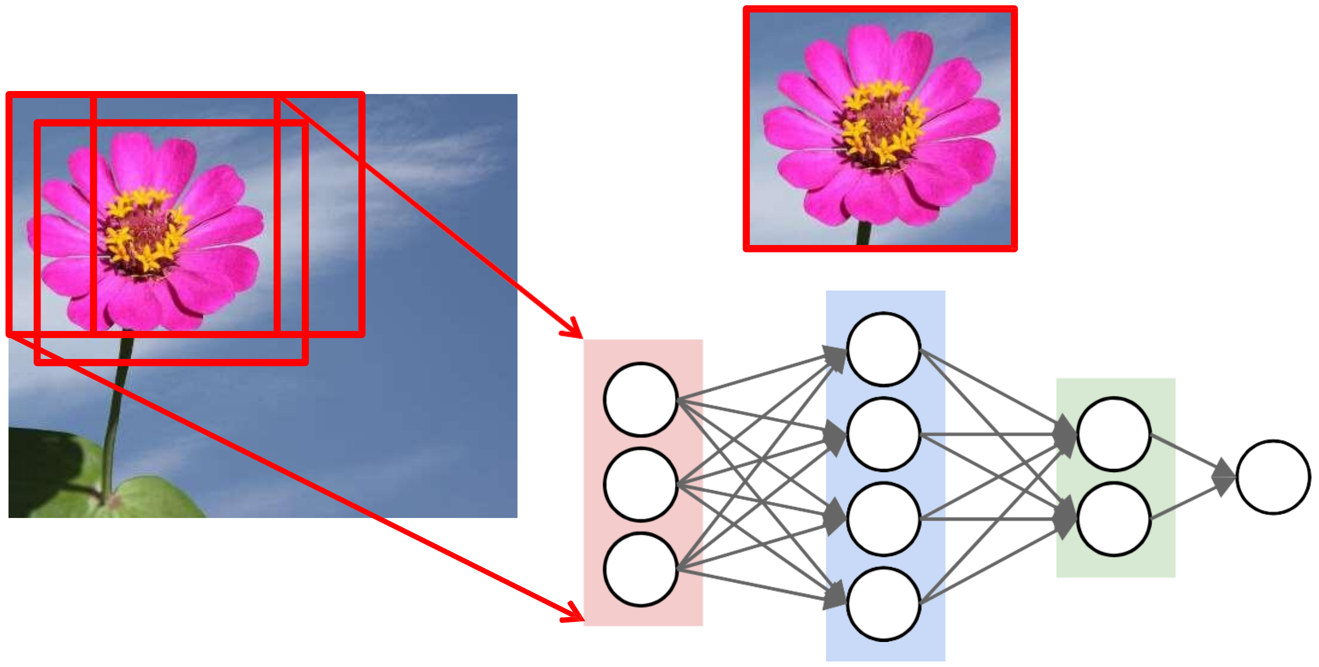
Solution: Scan



- "Does welcome occur in this recording?"
  - Maximum of all outputs (Equivalent of Boolean OR)
  - Or more complex function

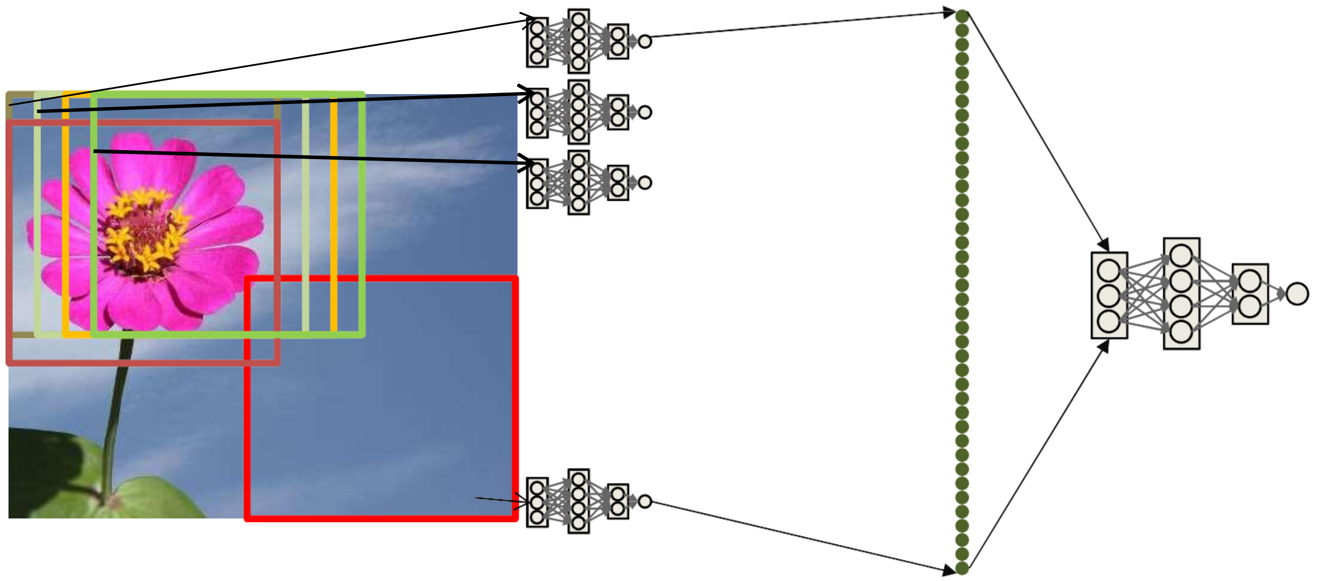
Adapted from Bhiksha Raj

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



- ***Scan*** for the desired object
  - "Look" for the target object at each position
  - At each location, entire region is sent through NN

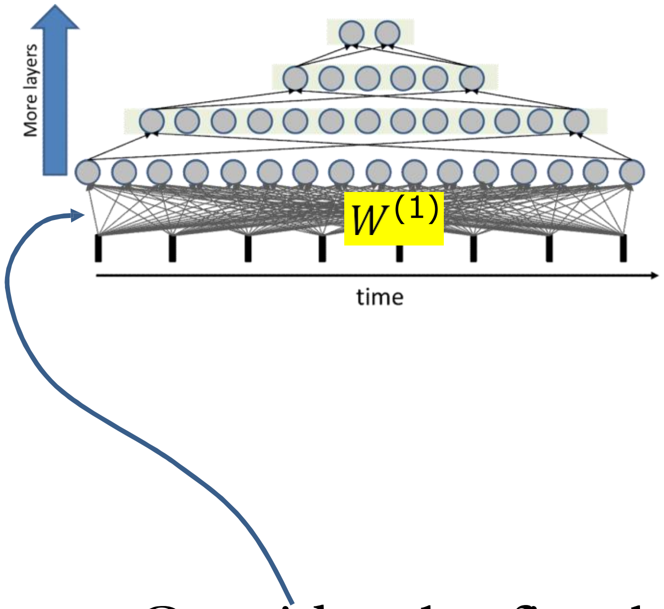
A giant net with common identical subnets



- Determine if any of the locations had a flower
  - Each dot in the right represents the output of the NN when it classifies one location in the input figure
  - Look at the maximum value
  - Or pass it through a simple NN (e.g. linear combination + softmax)

Adapted from Bhiksha Raj

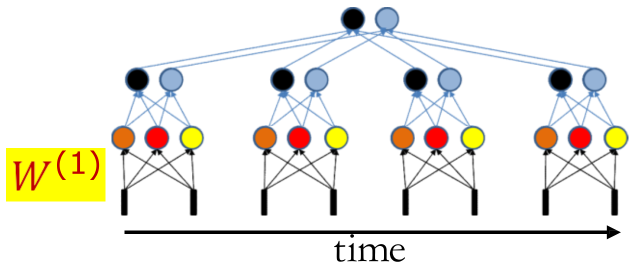
Regular network



$$W^{(1)} = \begin{bmatrix} w_{11} & w_{12} & w_{13} & w_{14} & \cdots & \cdots & \cdots & w_{1M} \\ w_{21} & w_{22} & w_{23} & w_{24} & \cdots & \cdots & \cdots & w_{2M} \\ w_{31} & w_{32} & w_{33} & w_{34} & \cdots & \cdots & \cdots & w_{3M} \\ w_{41} & w_{42} & w_{43} & w_{44} & \cdots & \cdots & \cdots & w_{4M} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ w_{N1} & w_{N2} & w_{N3} & w_{N4} & \cdots & \cdots & \cdots & w_{NM} \end{bmatrix}$$

- Consider the first layer
  - Assume N inputs and M outputs
- The weights matrix is a full $N \times M$ matrix
  - Requiring $N*M$ unique parameters

Scanning networks

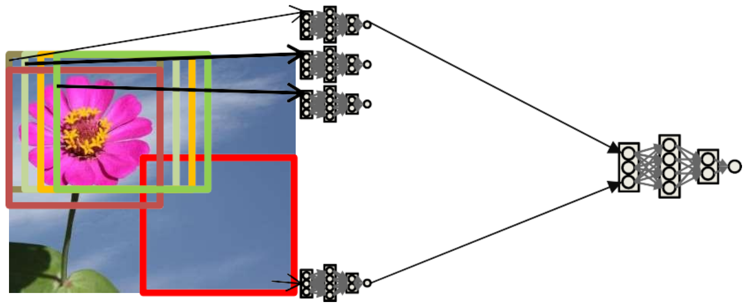


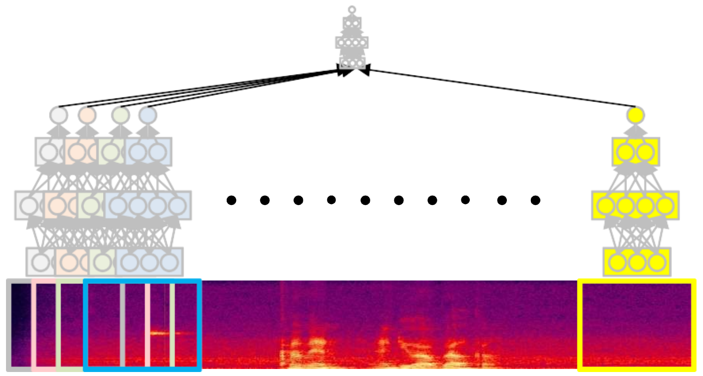
$$W^{(1)} = \begin{bmatrix}
w_{11} & w_{12} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\
w_{21} & w_{22} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\
w_{31} & w_{32} & 0 & 0 & 0 & \cdots & \cdots & \cdots & 0 \\
0 & 0 & w_{11} & w_{12} & 0 & 0 & 0 & \cdots & 0 \\
0 & 0 & w_{21} & w_{22} & 0 & 0 & 0 & \vdots & \vdots \\
0 & 0 & w_{31} & w_{32} & 0 & 0 & 0 & \vdots & \vdots \\
0 & 0 & 0 & 0 & w_{11} & w_{12} & 0 & 0 & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\
0 & 0 & 0 & 0 & \cdots & 0 & 0 & w_{31} & w_{32}
\end{bmatrix}$$

- In a ***scanning NN*** each neuron is connected to a subset of neurons in the previous layer
  - The weights matrix is sparse
  - The weights matrix is block structured ***with identical blocks***
  - **The network is a *shared parameter* model**

Adapted from Bhiksha Raj

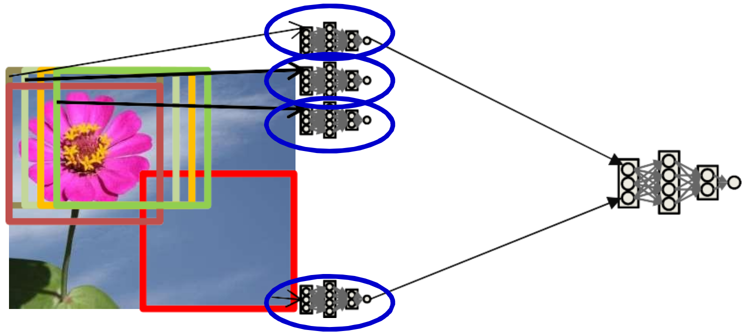
Training the network

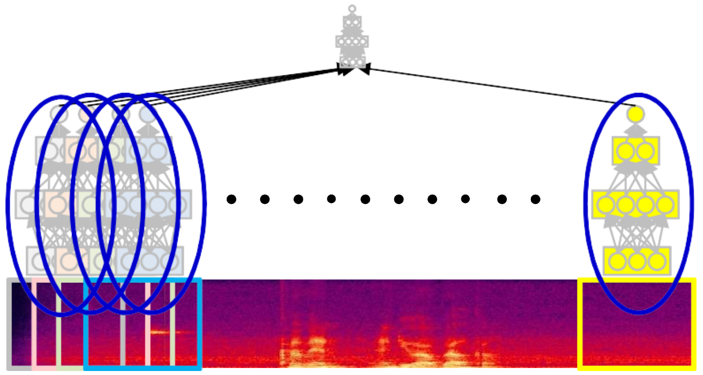




- These are really just large networks
  - Can use conventional backpropagation to learn parameters
  - Backprop learns a network that maps the training inputs to the target binary outputs

Training the network: constraint

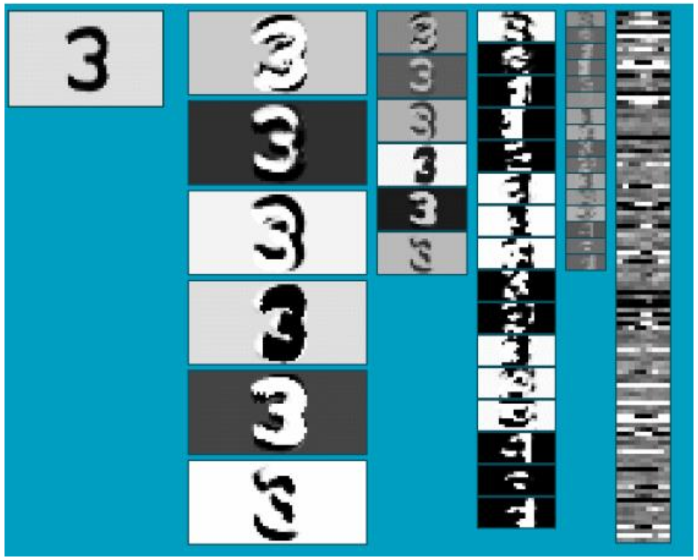


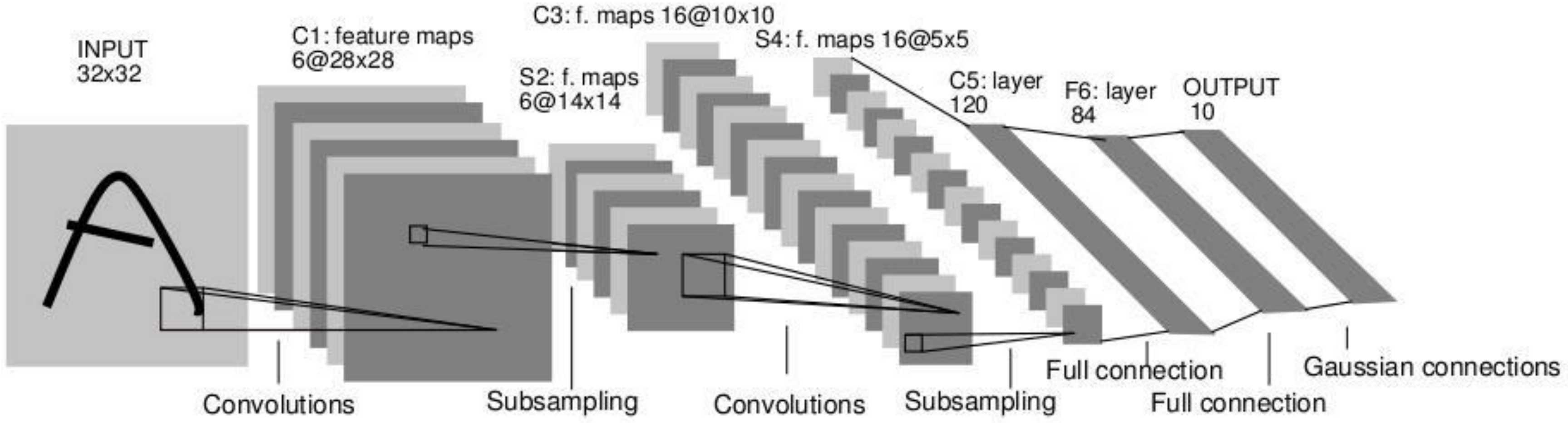


- These are ***shared parameter*** networks
  - All lower-level subnets are identical
    - Are all searching for the same pattern
  - Any update of the parameters of one copy of the subnet must equally update ***all*** copies

Convolutional Neural Networks (CNN)

- Neural network with specialized connectivity structure
- Stack multiple stages of feature extractors
- Higher stages compute more global, more invariant, *more abstract* features
- Classification layer at the end



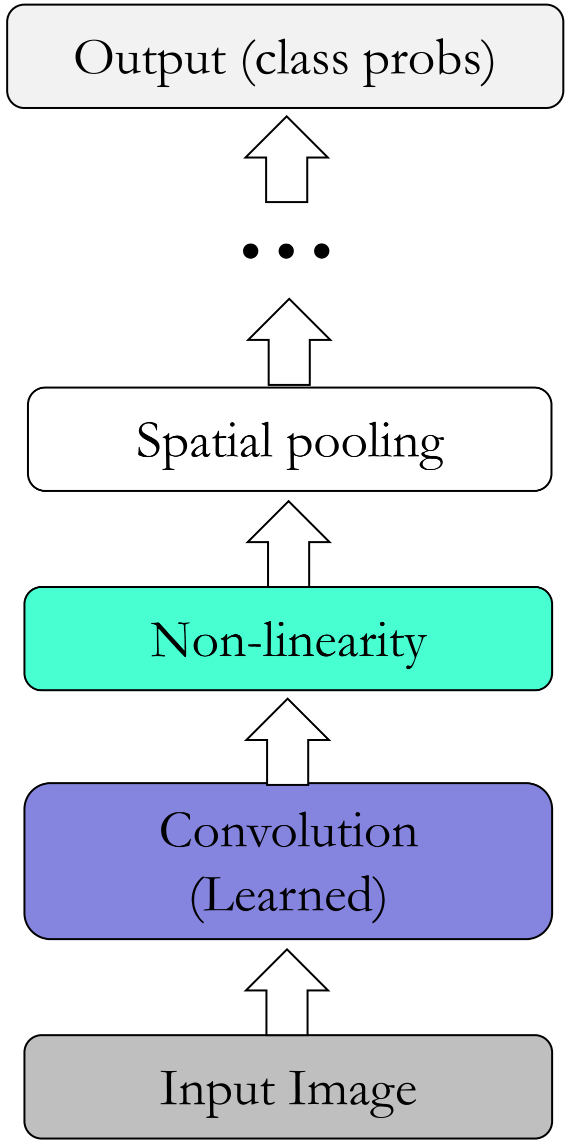


Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, Gradient-based learning applied to document recognition, Proceedings of the IEEE 86(11): 2278–2324, 1998.

Adapted from Rob Fergus

Convolutional Neural Networks (CNN)

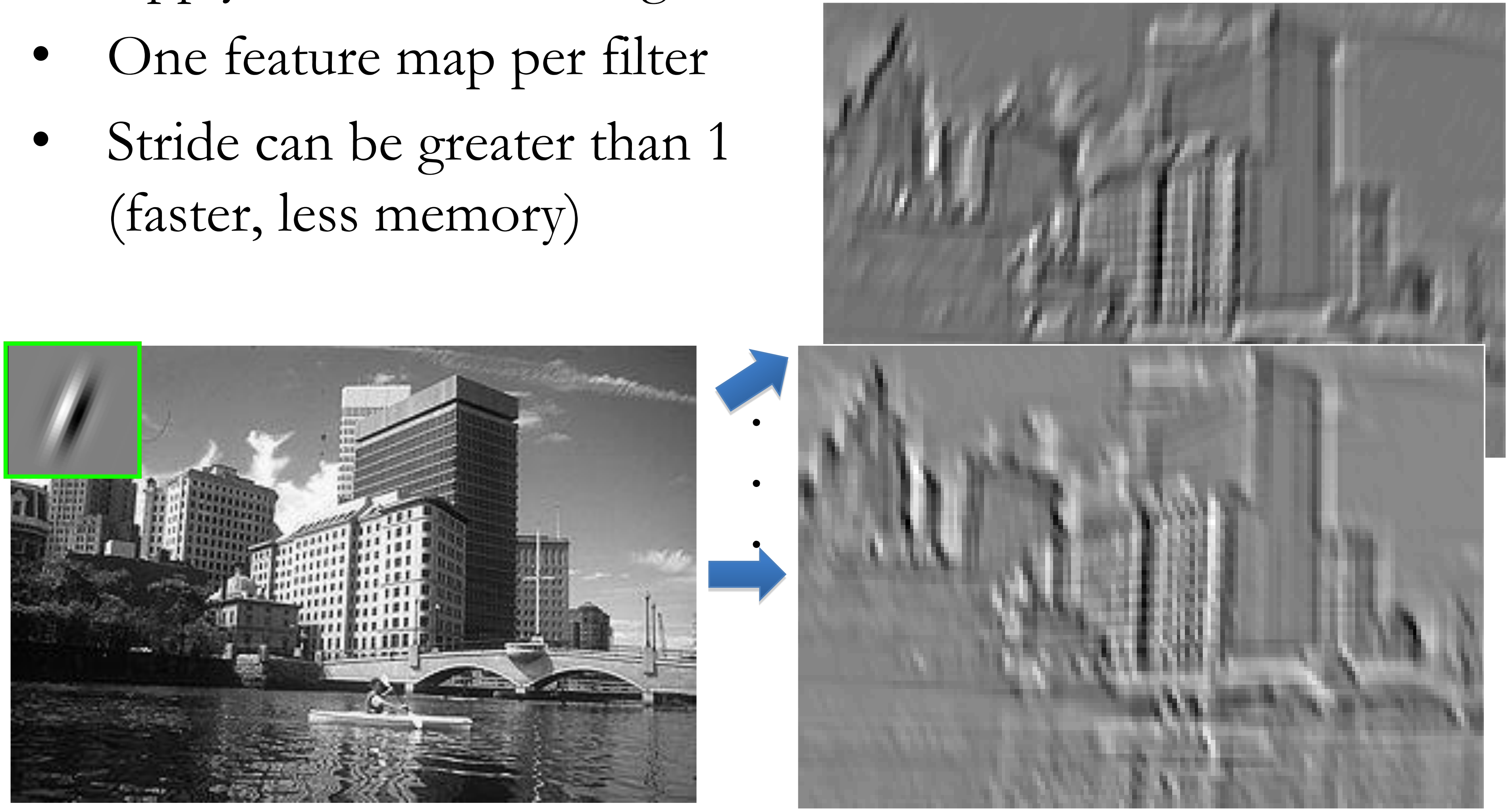
- Feed-forward feature extraction:
  1. *Convolve* input with learned filters
  2. Apply non-linearity
  3. Spatial pooling (downsample)
- Recent architectures have additional operations (to be discussed)
- Trained with some loss, backprop



Adapted from Lana Lazebnik

1. Convolution

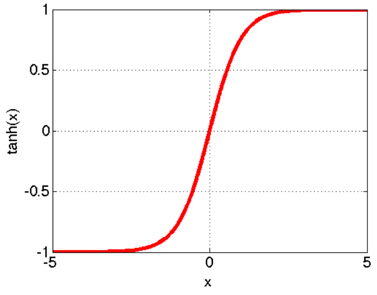
- Apply learned filter weights
- One feature map per filter
- Stride can be greater than 1 (faster, less memory)

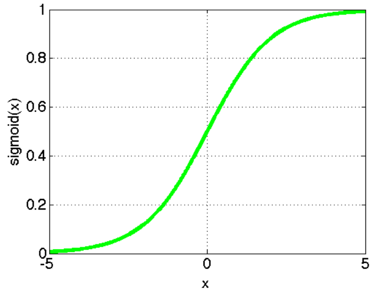


Adapted from Rob Fergus

2. Non-Linearity

- Per-element (independent)
- Some options:
  - Tanh
  - Sigmoid: 1/(1+exp(-x))
  - Rectified linear unit (ReLU)
    - Avoids saturation issues





Krizhevsky et al.

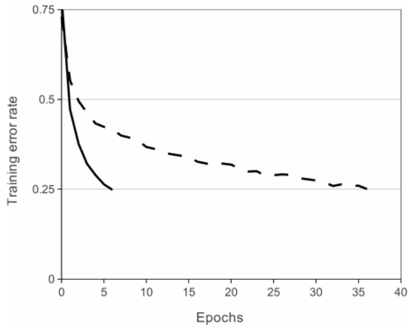
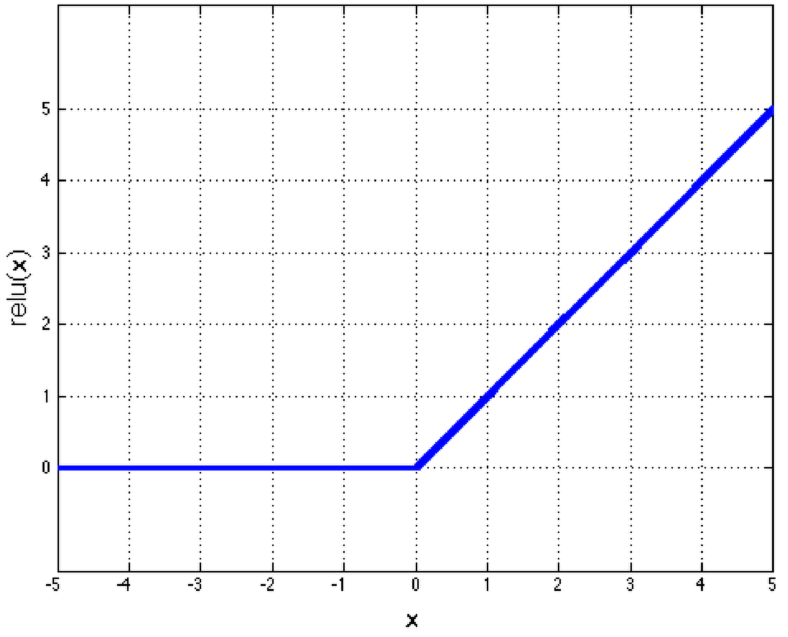


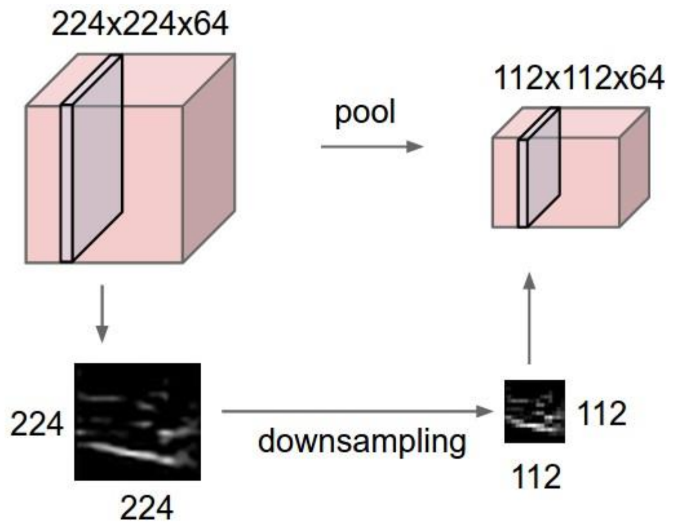
Figure 1: A four-layer convolutional neural network with ReLUs (**solid line**) reaches a 25% training error rate on CIFAR-10 six times faster than an equivalent network with tanh neurons (**dashed line**). The learning rates for each network were chosen independently to make training as fast as possible. No regularization of any kind was employed. The magnitude of the effect demonstrated here varies with network architecture, but networks with ReLUs consistently learn several times faster than equivalents with saturating neurons.

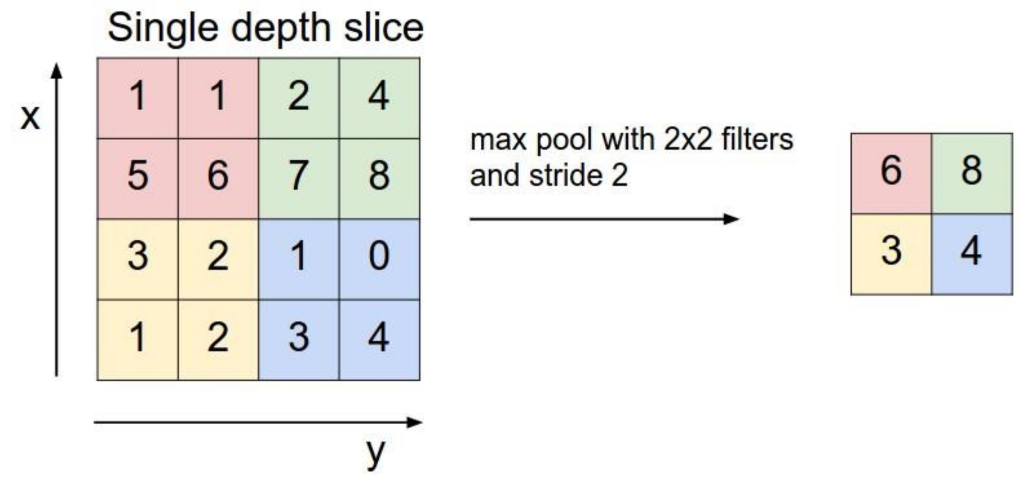


Adapted from Rob Fergus

3. Spatial Pooling

- Sum or max over non-overlapping / overlapping regions



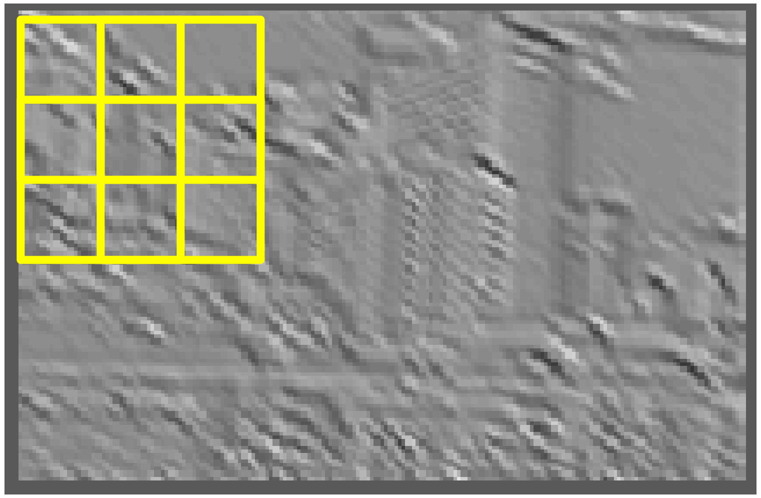


- Role of pooling:
  - Invariance to small transformations
  - Larger receptive fields (neurons see more of input)

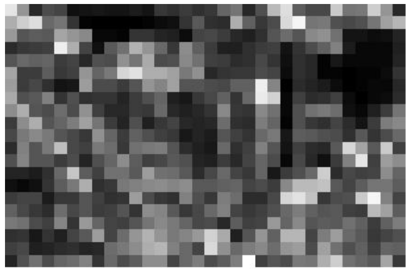
Rob Fergus, figure from Andrej Karpathy

3. Spatial Pooling

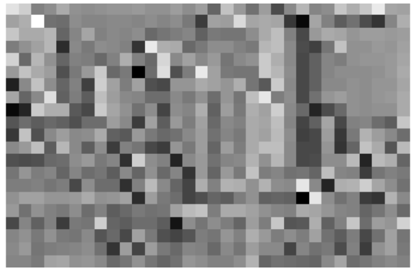
- Sum or max over non-overlapping / overlapping regions



Max



Sum



Adapted from Rob Fergus

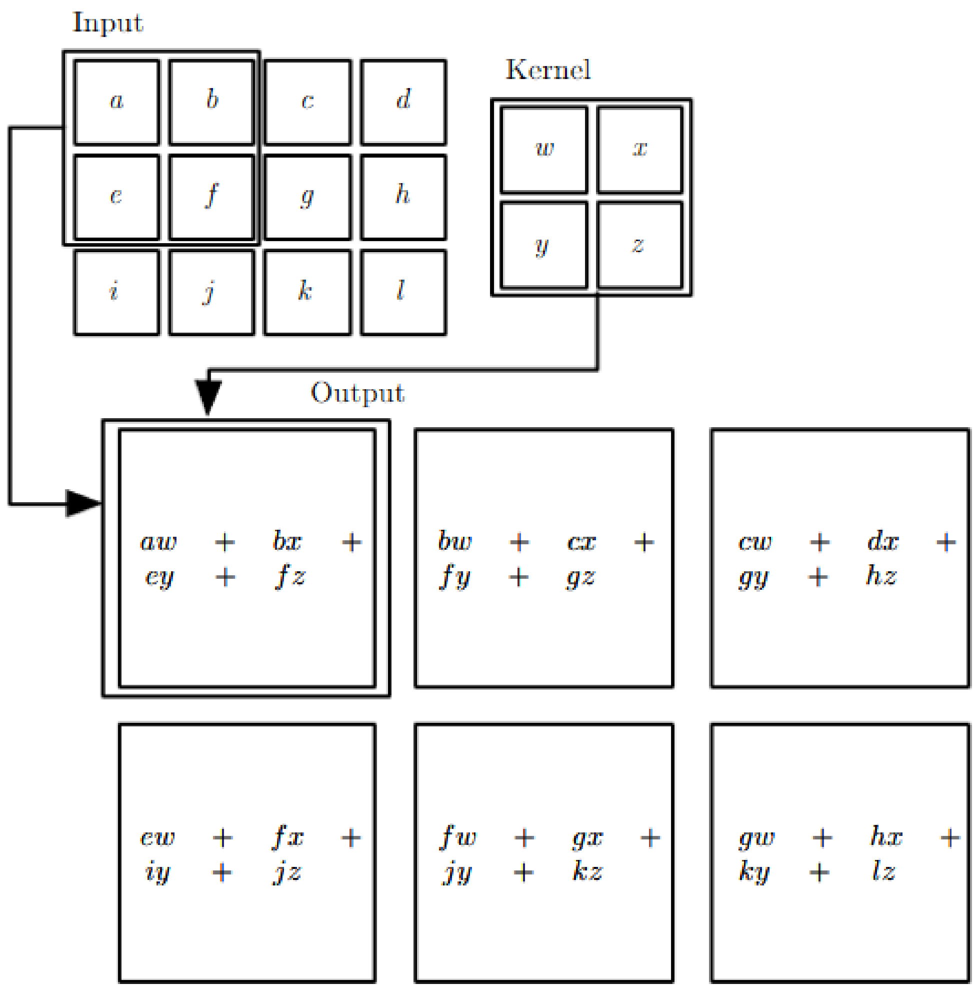
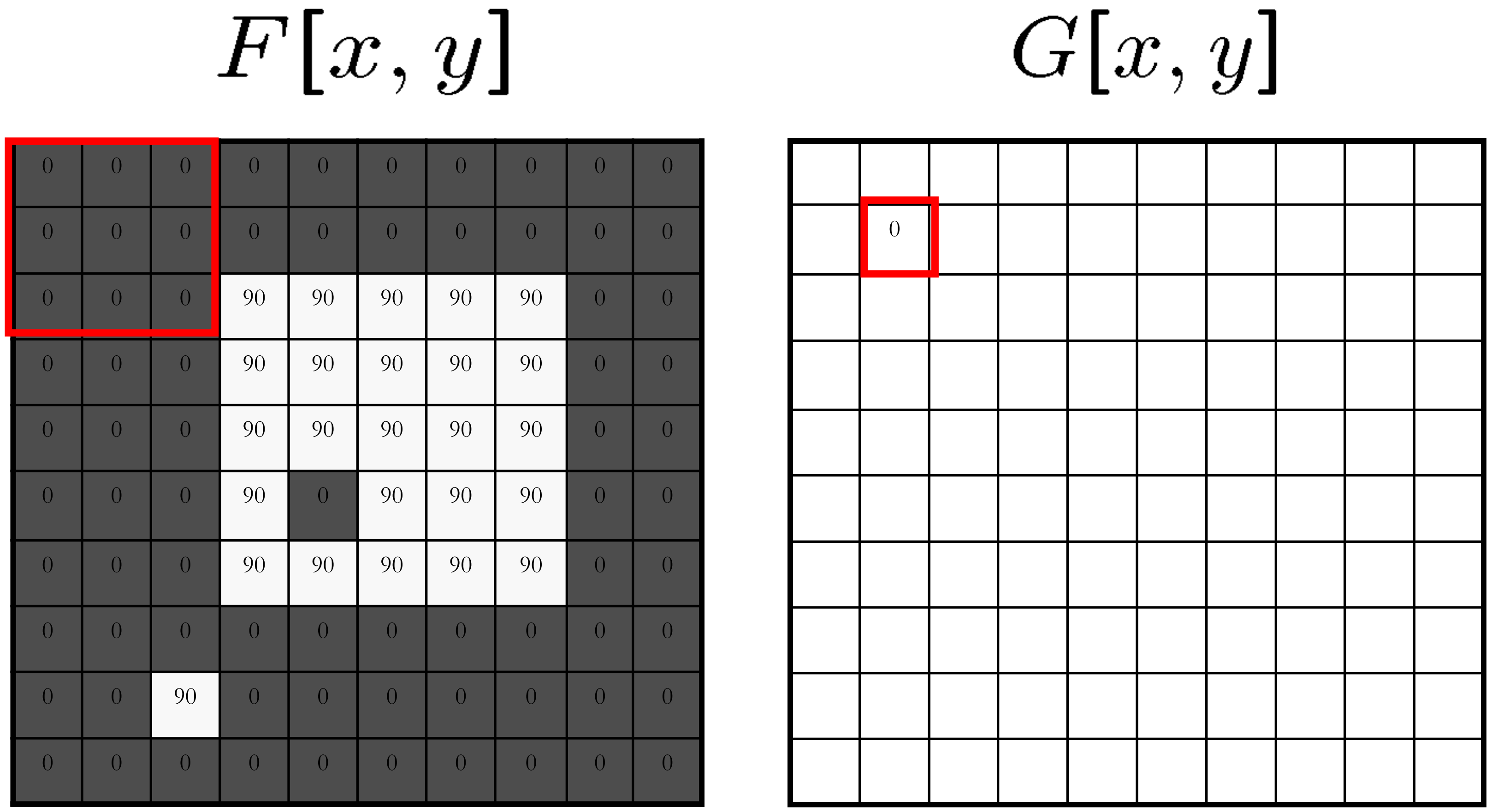


Figure 9.1: An example of 2-D convolution without kernel flipping. We restrict the output to only positions where the kernel lies entirely within the image, called "valid" convolution in some contexts. We draw boxes with arrows to indicate how the upper-left element of the output tensor is formed by applying the kernel to the corresponding upper-left region of the input tensor.

Background: Moving Average In 2D



$F[x, y]$

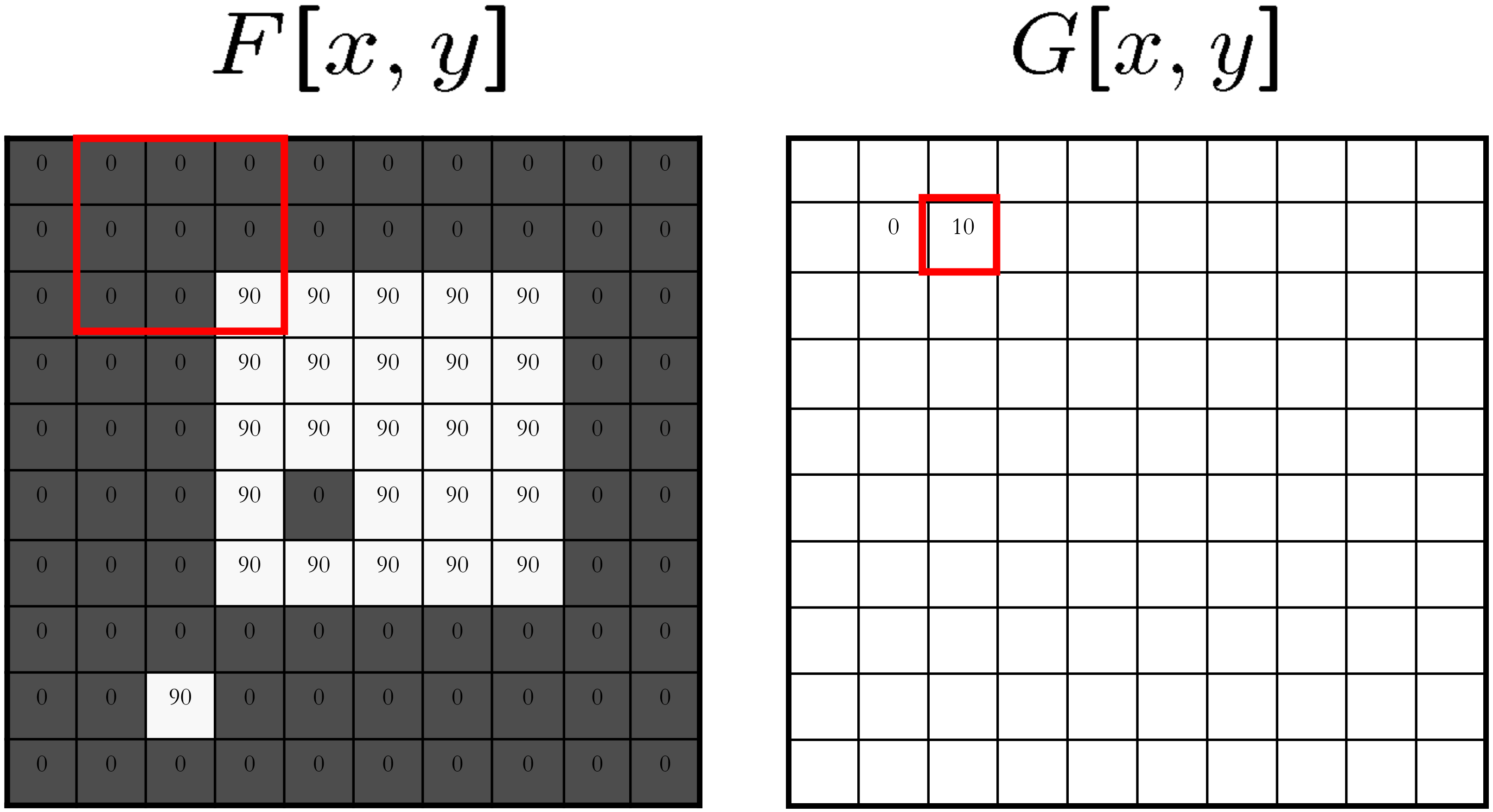
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0								

Source: S. Seitz

Background: Moving Average In 2D



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Source: S. Seitz

Background: Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20						

Source: S. Seitz

Background: Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30					

Source: S. Seitz

Background: Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30				

Source: S. Seitz

Background: Moving Average In 2D

$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$G[x, y]$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

Source: S. Seitz

Image filtering

- Compute a function of the local neighborhood at each pixel in the image
  - Function specified by a "filter" or mask saying how to combine values from neighbors.
  - Element-wise multiplication

- Uses of filtering:
  - Enhance an image (denoise, resize, etc)
  - Extract information (texture, edges, etc)
  - **Detect patterns (template matching)**

Adapted from Derek Hoiem

Correlation filtering

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\textit{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} F[i+u, j+v]}_{\textit{Loop over all pixels in neighborhood around image pixel } F[i,j]}$$

Now generalize to allow **different weights** depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H[u,v]}_{\textit{Non-uniform weights}} F[i+u, j+v]$$

Kristen Grauman

Correlation filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v] F[i+u, j+v]$$

This is called **cross-correlation**, denoted $G = H \otimes F$

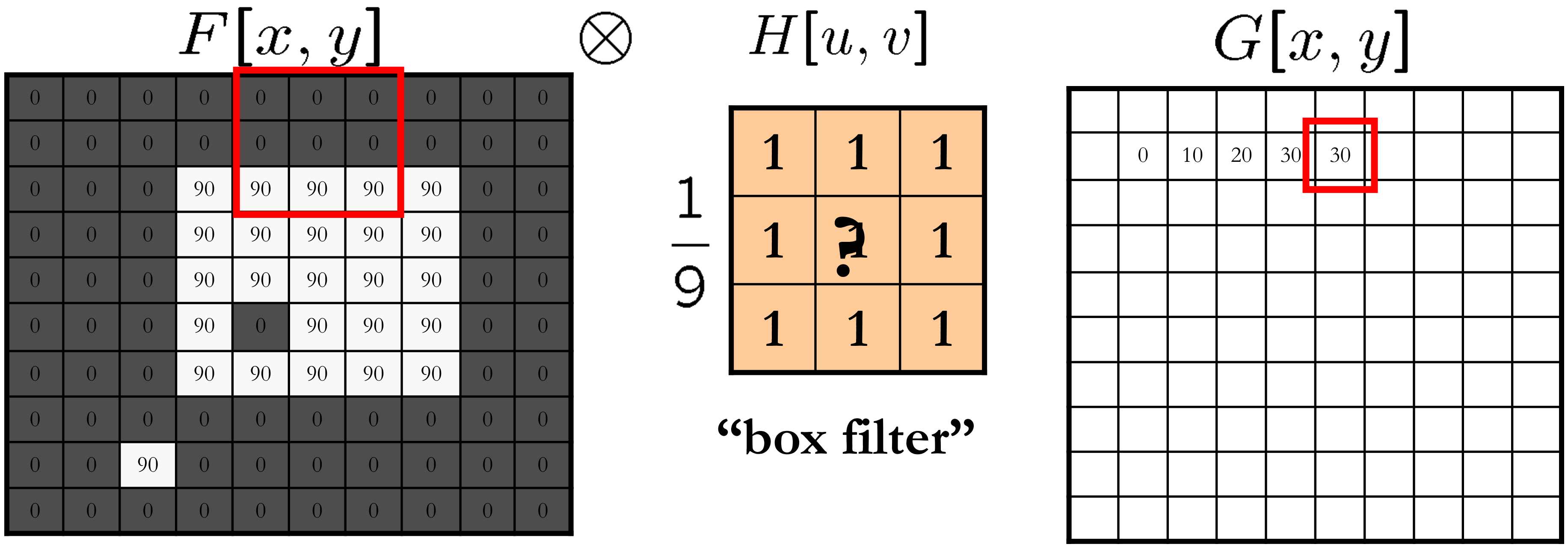
Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter a.k.a. kernel a.k.a. mask *H*[*u*,*v*] is the prescription for the weights in the linear combination.

Adapted from Kristen Grauman

Averaging filter

- What values belong in the kernel H for the moving average example?



$F[x, y]$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$$H[u, v] = \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & ? & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

"box filter"

$G[x, y]$

	0	10	20	30	30				

$$G = H \otimes F$$

Kristen Grauman

Gaussian filter

- What if we want nearest neighboring pixels to have the most influence on the output?

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

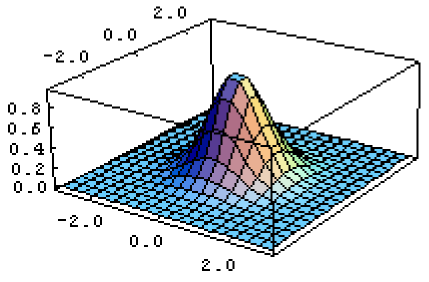
$F[x, y]$

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$H[u, v]$

This kernel is an approximation of a 2d Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$



Source: S. Seitz

Convolution

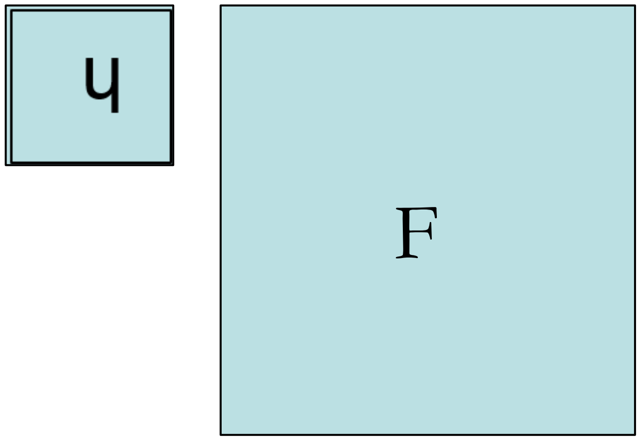
- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

$$G = H \star F$$

$$\uparrow$$

Notation for convolution operator



Kristen Grauman

Convolution vs. correlation

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

For a Gaussian or box filter, how will the outputs differ?

Kristen Grauman

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

F

5	2	5	4	4	
5	200	3	200	4	
1	5	5	4	4	(i, j)
5	5	1	1	2	
200	1	3	5	200	
1	200	200	200	1	

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

H

.06	.12	.06	
.12	.25	.12	(0, 0)
.06	.12	.06	

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

v = 0

F

5	2	5	4	4
5	200	3	200	4
1	5	5	4	4
5	5	1	1	2
200	1	3	5	200
1	200	200	200	1

(i, j)

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

H

.06	.12	.06
.12	.25	.12
.06	.12	.06

(0, 0)

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

v = 0

v = +1

F

5	2	5	4	4	
5	200	3	200	4	
1	5	5	4	4	(i, j)
5	5	1	1	2	
200	1	3	5	200	
1	200	200	200	1	

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

H

.06	.12	.06	
.12	.25	.12	(0, 0)
.06	.12	.06	

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

v = 0

v = +1

u = 0, v = -1

F

5	2	5	4	4
5	200	3	200	4
1	5	5	4	4
5	5	1	1	2
200	1	3	5	200
1	200	200	200	1

(i, j)

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

H

.06	.12	.06
.12	.25	.12
.06	.12	.06

(0, 0)

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

F

5	2	5	4	4	
5	200	3	200	4	
1	5	5	4	4	(i, j)
5	5	1	1	2	
200	1	3	5	200	
1	200	200	200	1	

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

H

.06	.12	.06	
.12	.25	.12	(0, 0)
.06	.12	.06	

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u,j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

v = 0

F

5	2	5	4	4
5	200	3	200	4
1	5	5	4	4
5	5	1	1	2
200	1	3	5	200
1	200	200	200	1

(i, j)

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u,j-v]$$

$$G = H \star F$$

H

.06	.12	.06
.12	.25	.12
.06	.12	.06

(0, 0)

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

v = 0

v = +1

F

5	2	5	4	4
5	200	3	200	4
1	5	5	4	4
5	5	1	1	2
200	1	3	5	200
1	200	200	200	1

(i, j)

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

H

.06	.12	.06
.12	.25	.12
.06	.12	.06

(0, 0)

Convolution vs. correlation

Cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

$$G = H \otimes F$$

u = -1, v = -1

v = 0

v = +1

u = 0, v = -1

F

5	2	5	4	4
5	200	3	200	4
1	5	5	4	4
5	5	1	1	2
200	1	3	5	200
1	200	200	200	1

(i, j)

Convolution

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

H

.06	.12	.06
.12	.25	.12
.06	.12	.06

(0, 0)

Predict the outputs using correlation filtering

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 0 & 0 \end{bmatrix} = ?$$

$$* \begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} = ?$$

Kristen Grauman

Practice with linear filters

Original

0	0	0
0	1	0
0	0	0

?

Source: D. Lowe

Practice with linear filters

Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline\end{array}$$

Filtered
(no change)

Source: D. Lowe

Practice with linear filters

Original

0	0	0
0	0	1
0	0	0

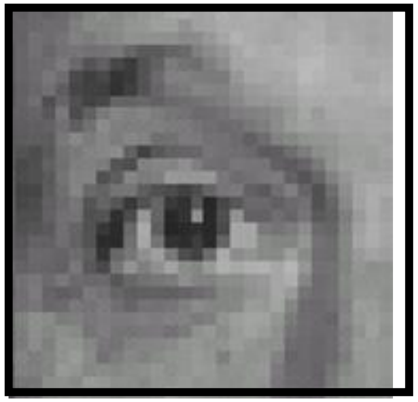
?

Source: D. Lowe

Practice with linear filters

Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$


Shifted left by 1 pixel with correlation

Source: D. Lowe

Practice with linear filters

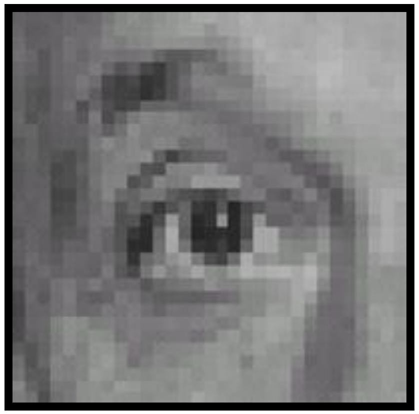
Original

$$\frac{1}{9}\boxed{\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline\end{array}}$$

?

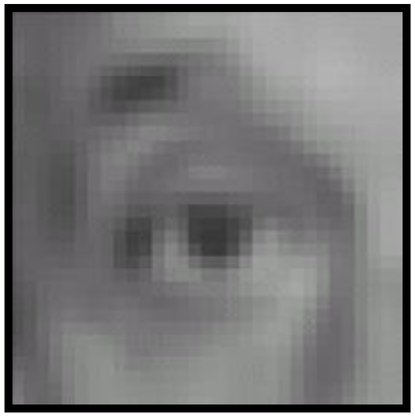
Source: D. Lowe

Practice with linear filters



Original

$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$



Blur (with a box filter)

Source: D. Lowe

Practice with linear filters

Original

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 2 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array} - \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

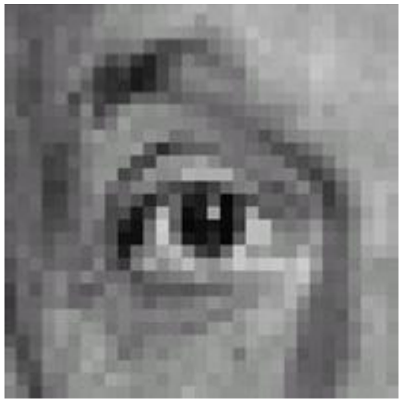
?

Source: D. Lowe

Practice with linear filters

Original

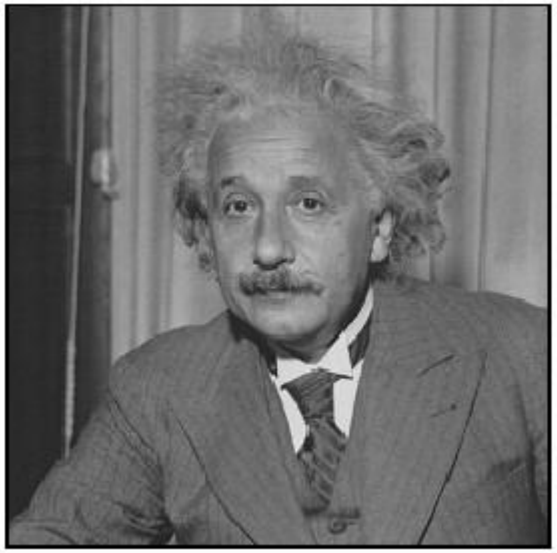
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

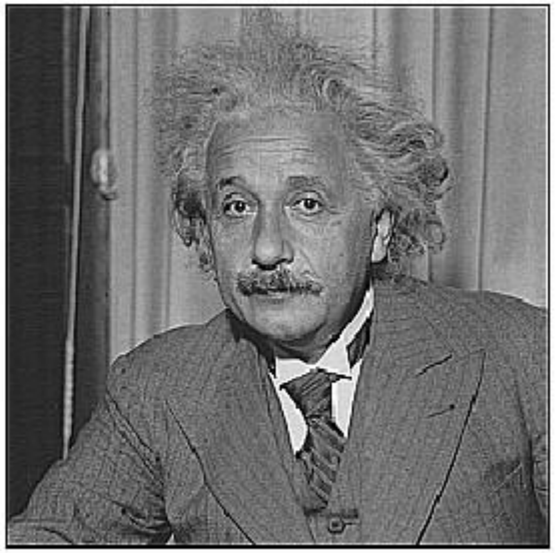


Sharpening filter:
accentuates differences with local average

Source: D. Lowe

Sharpening

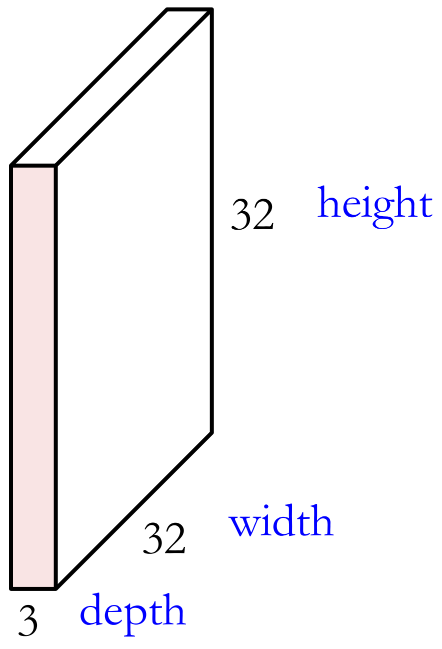

before


after

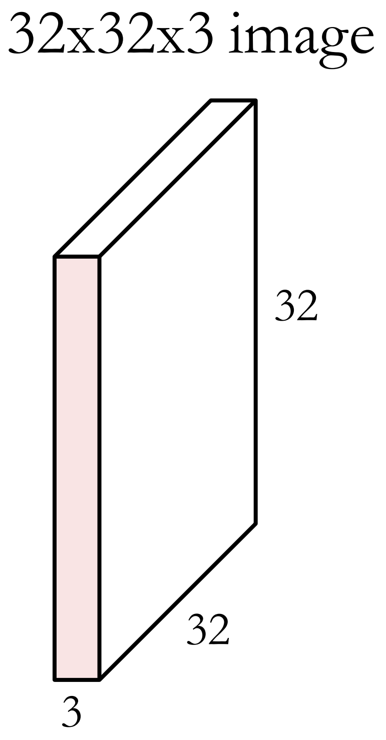
Kristen Grauman

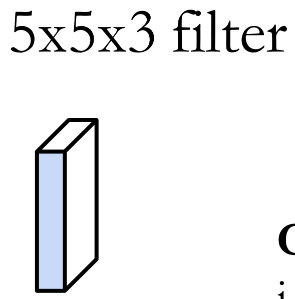
Convolutions: More detail

32x32x3 image



Convolutions: More detail

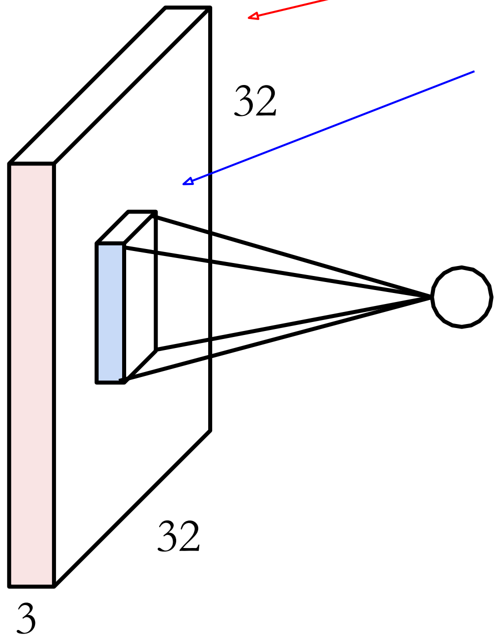




Convolve the filter with the image
i.e. "slide over the image spatially,
computing dot products"

Convolutions: More detail

Convolution Layer



32x32x3 image

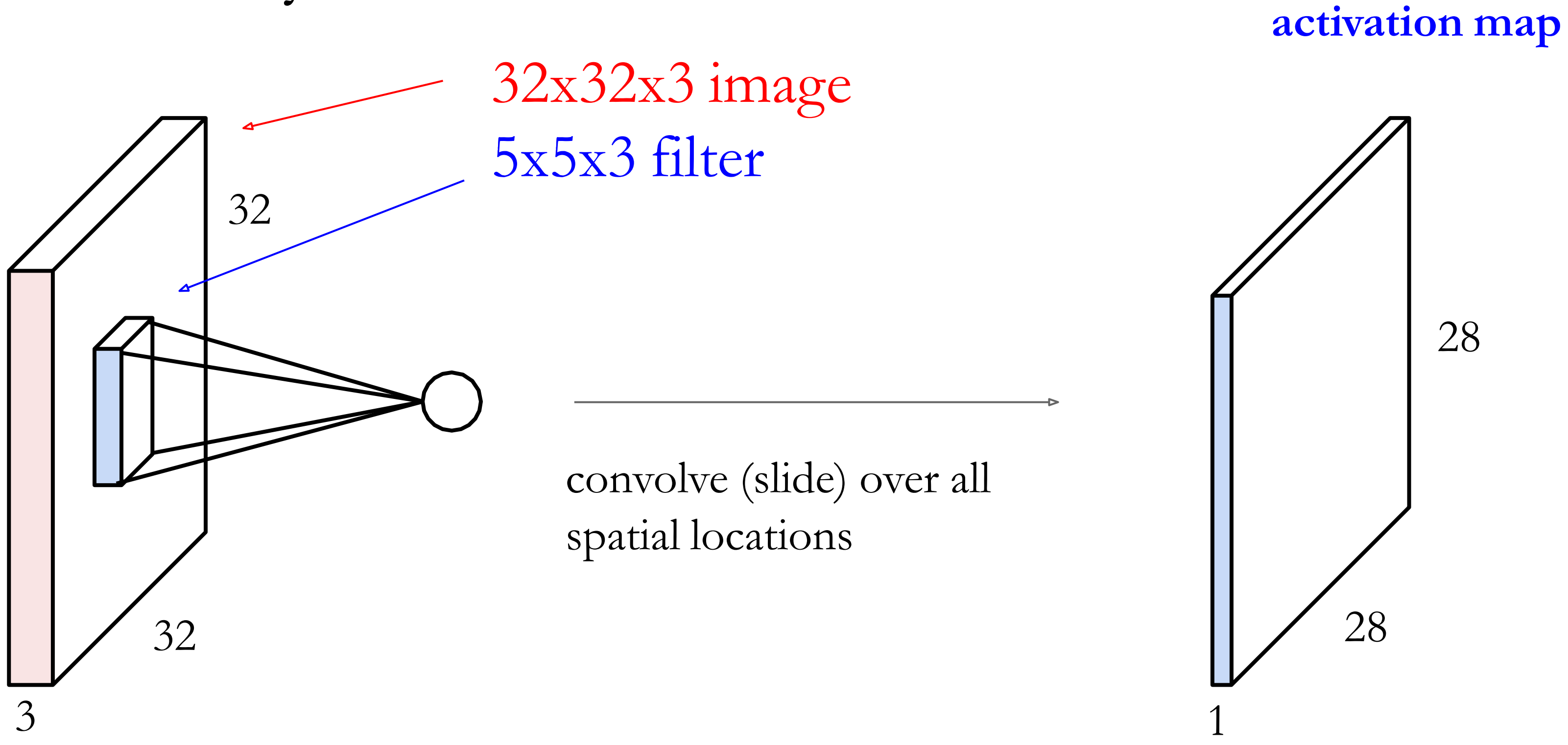
5x5x3 filter w

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)

$$w^T x + b$$

Convolutions: More detail

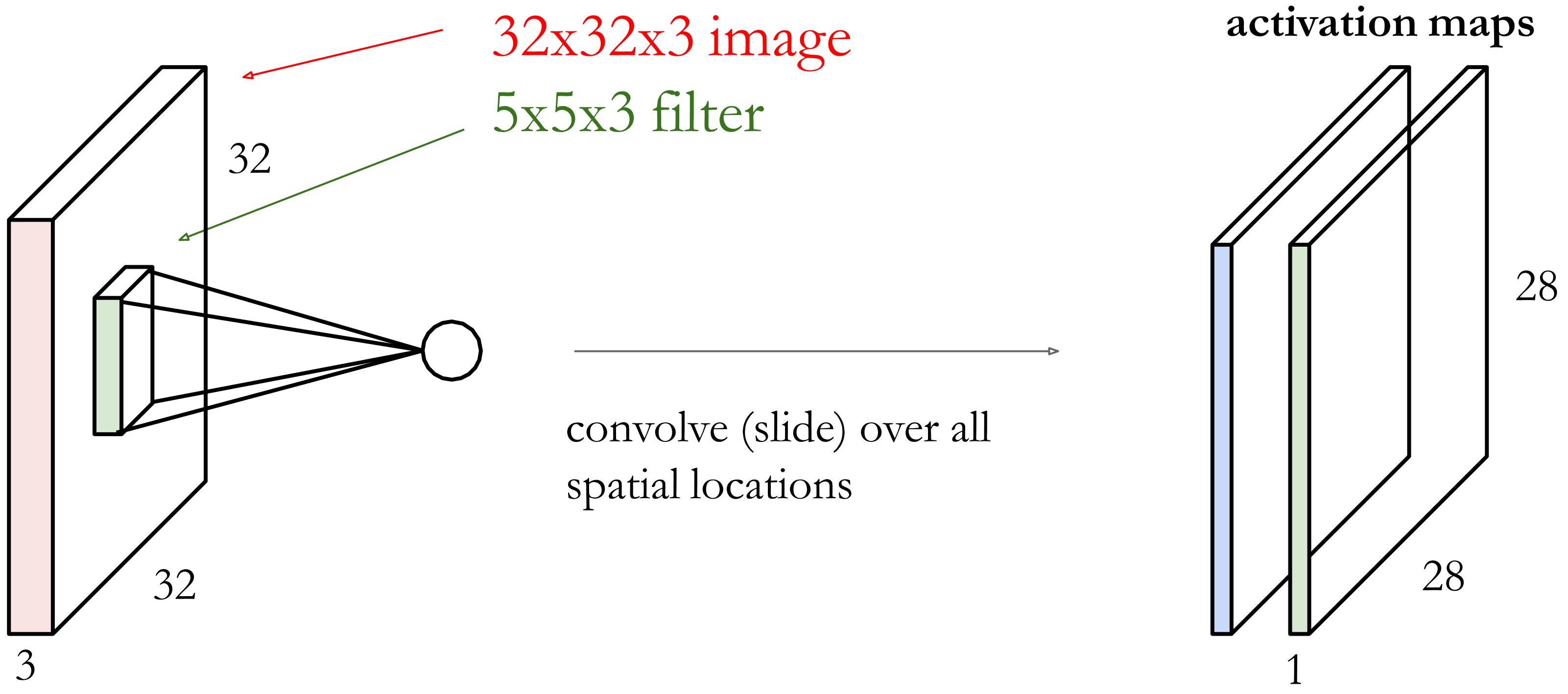
Convolution Layer



Convolutions: More detail

Convolution Layer

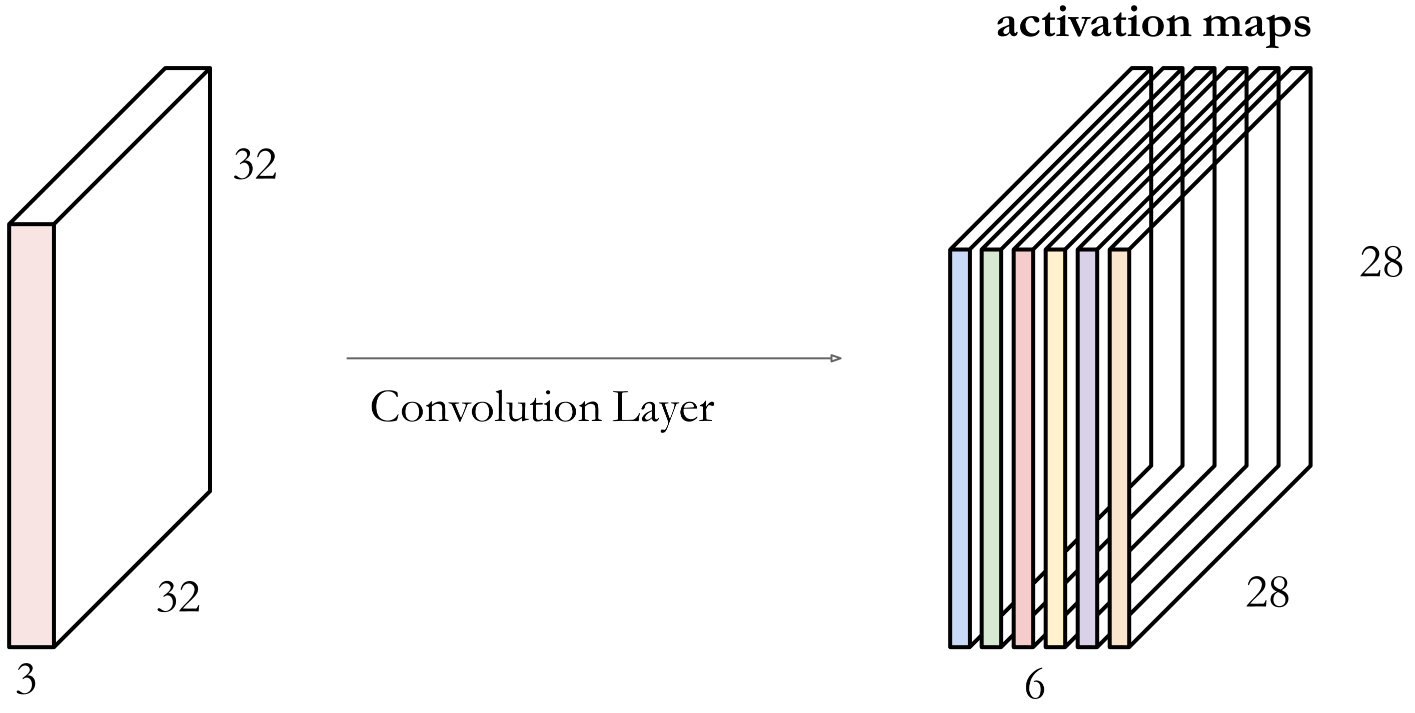
consider a second, green filter



Andrej Karpathy

Convolutions: More detail

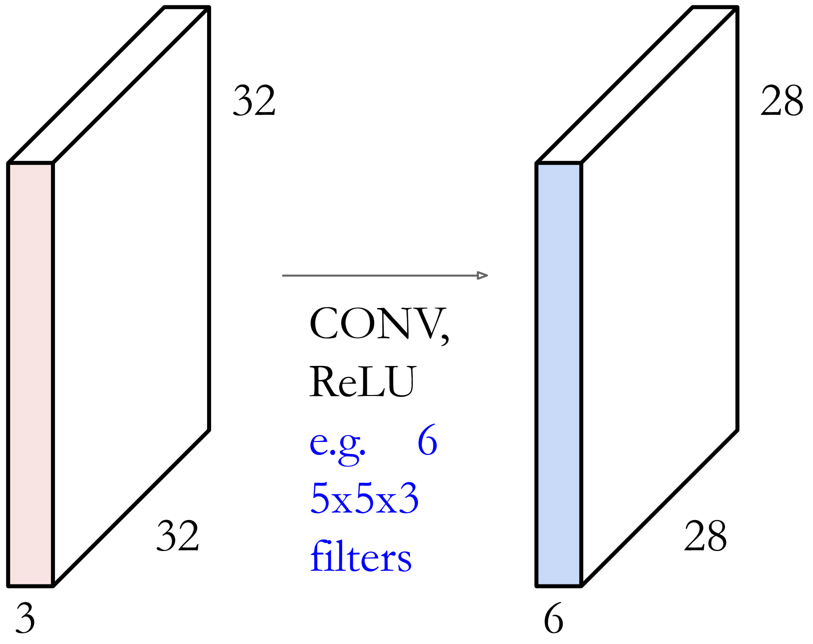
For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

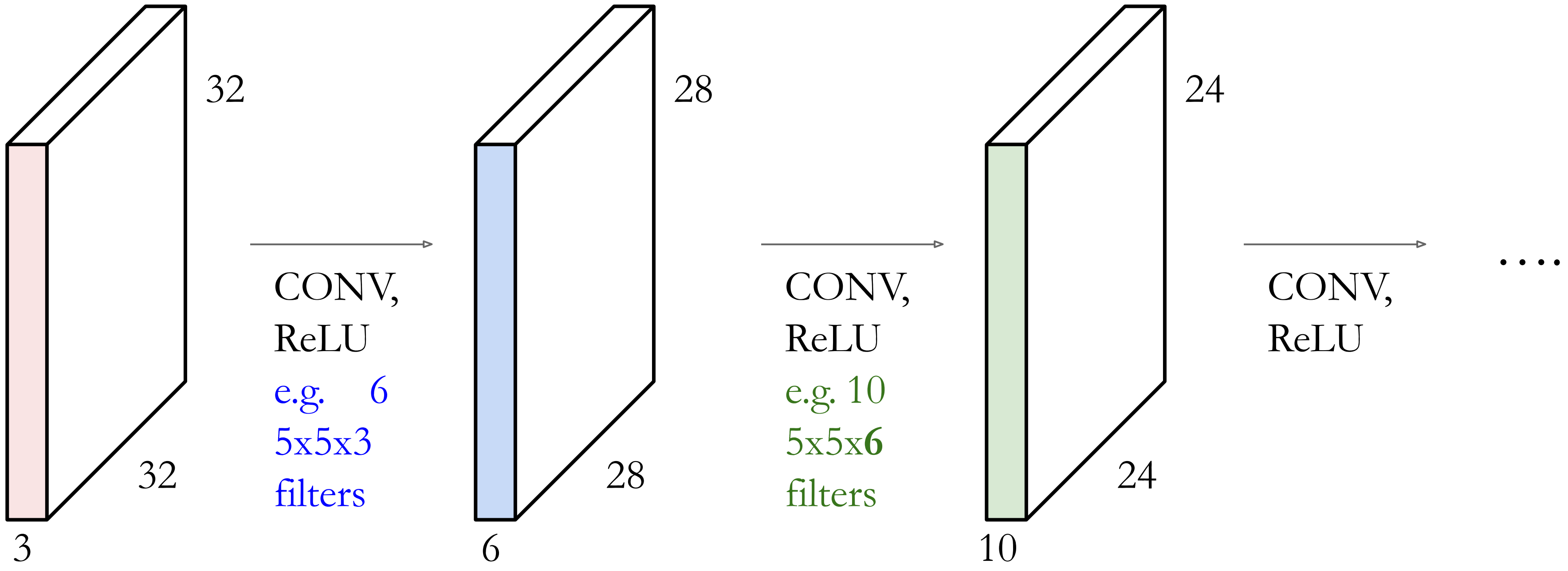
Convolutions: More detail

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



Convolutions: More detail

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

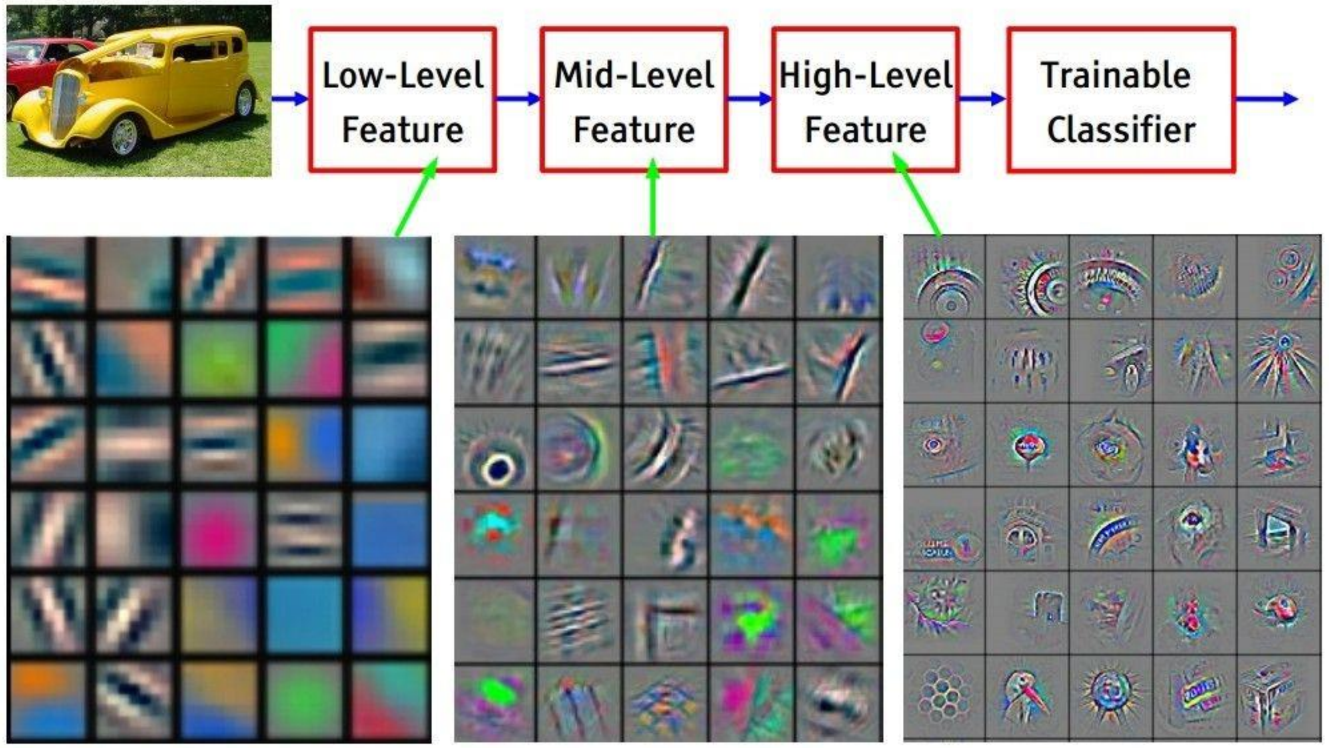


Andrej Karpathy

Convolutions: More detail

Preview

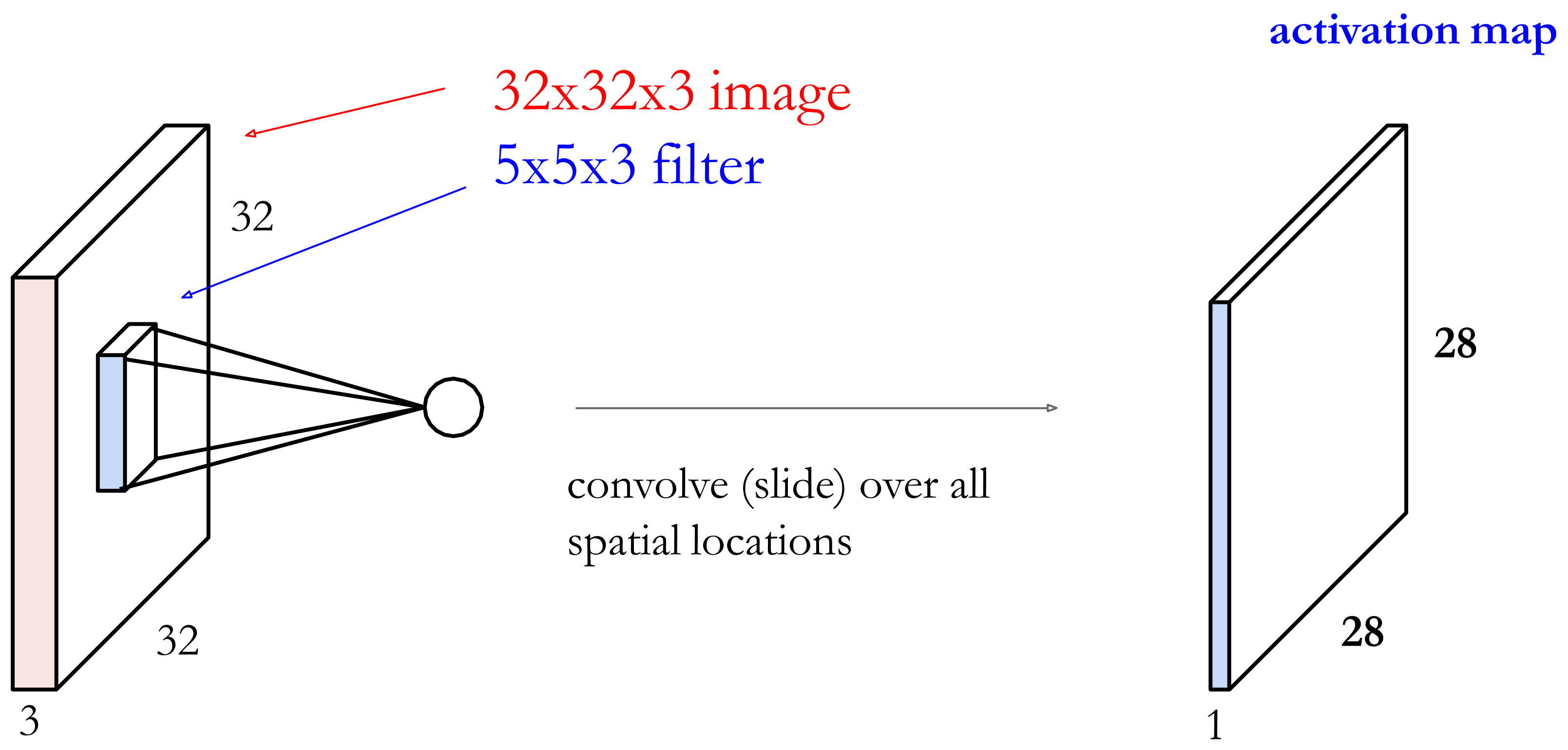
[From recent Yann LeCun slides]



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

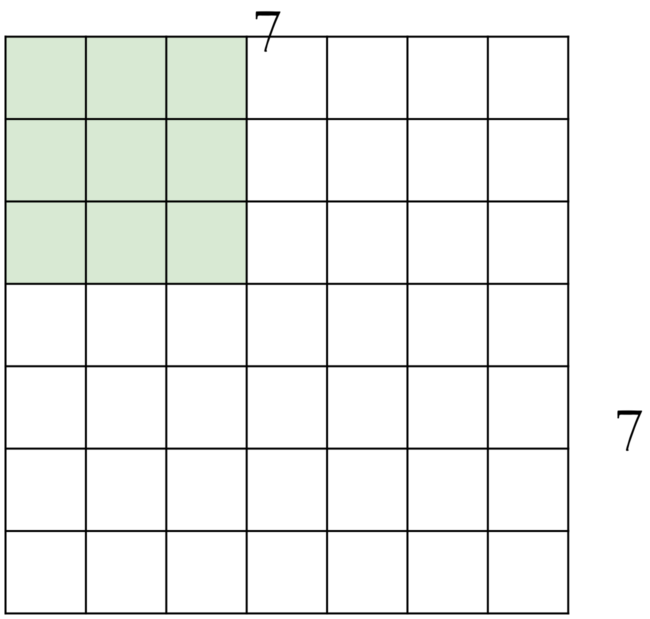
Convolutions: More detail

A closer look at spatial dimensions:



Convolutions: More detail

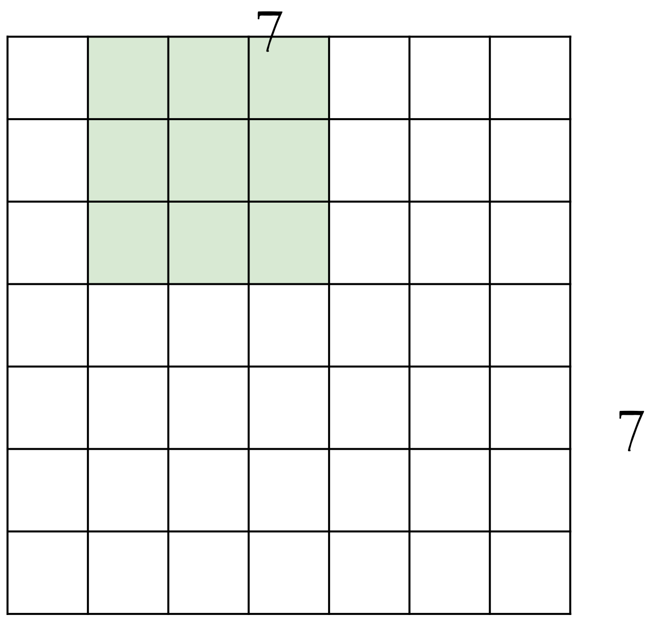
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutions: More detail

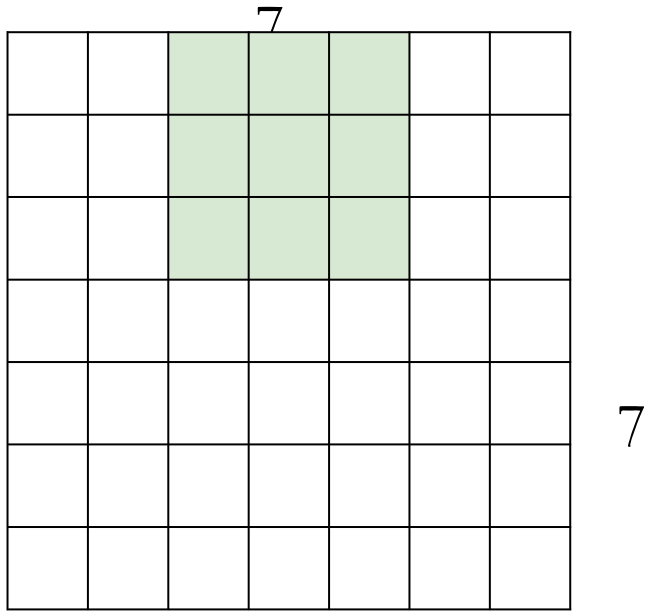
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutions: More detail

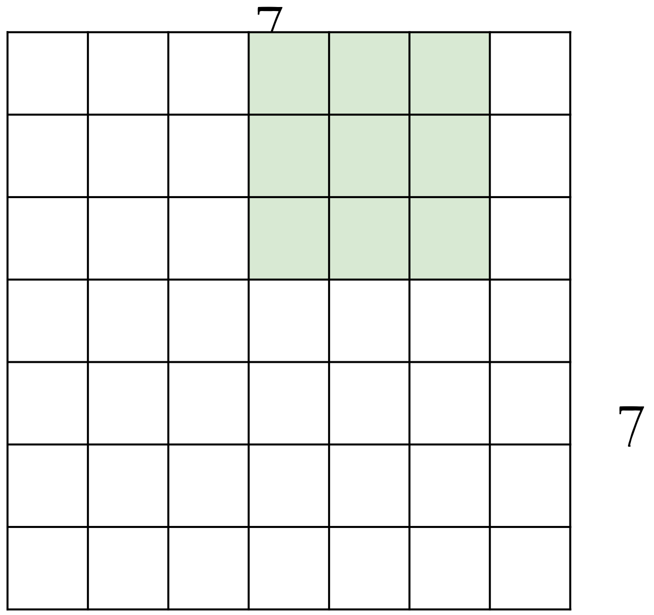
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutions: More detail

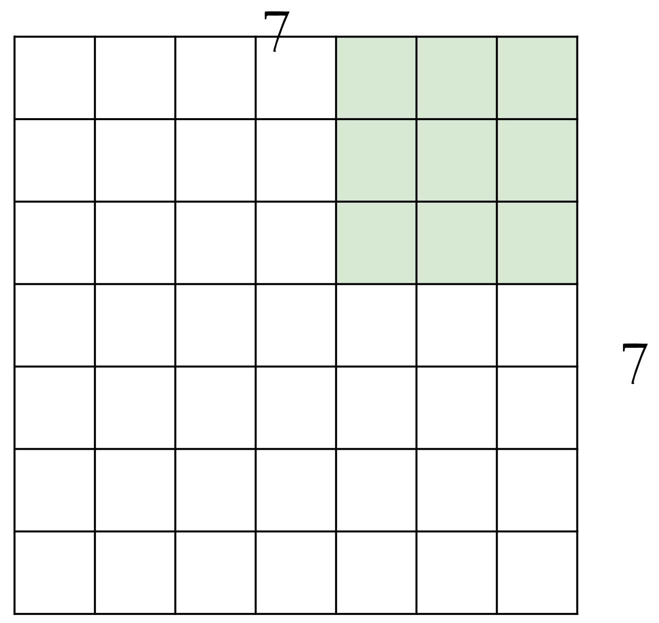
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter

Convolutions: More detail

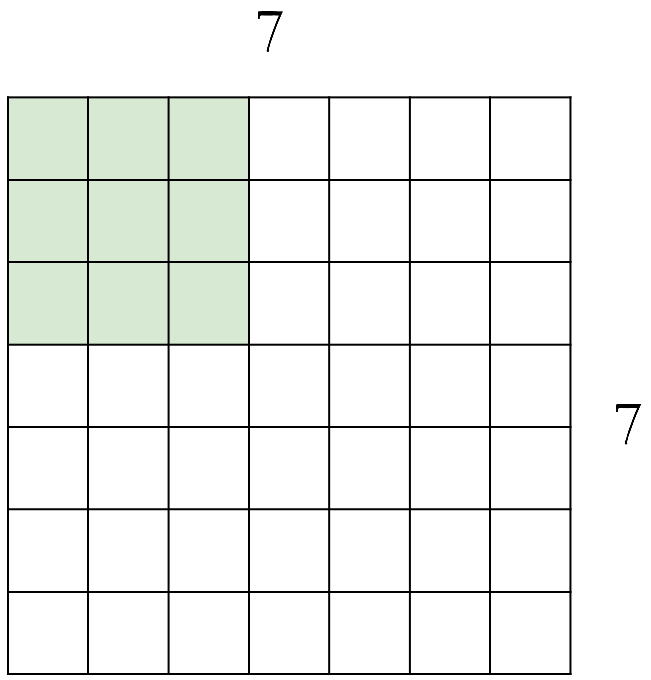
A closer look at spatial dimensions:



7x7 input (spatially)
assume 3x3 filter
=> 5x5 output

Convolutions: More detail

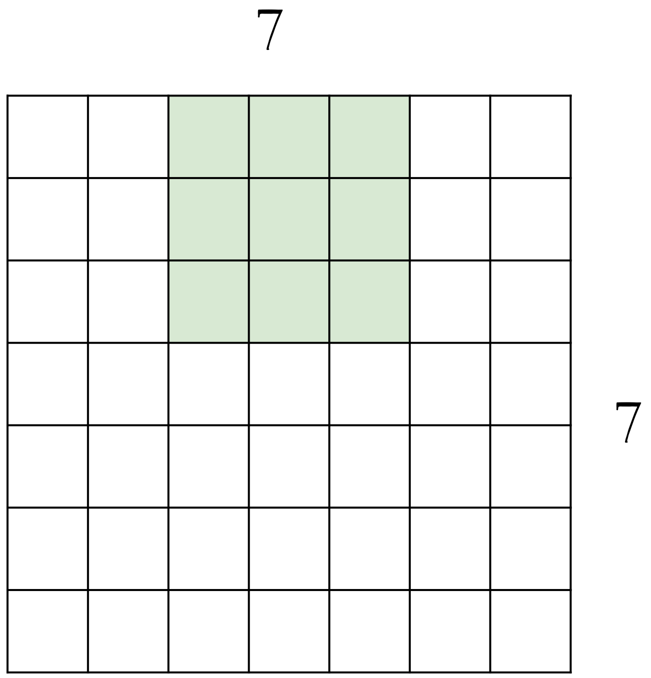
A closer look at spatial dimensions:



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

Convolutions: More detail

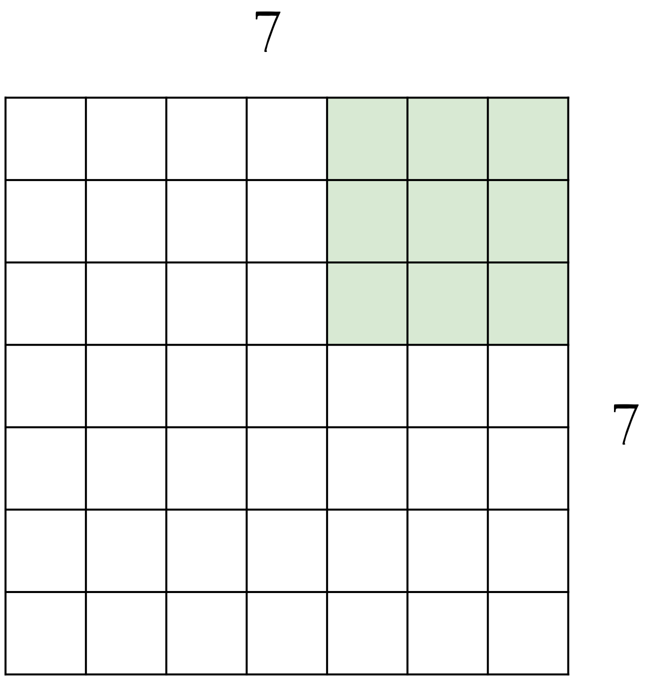
A closer look at spatial dimensions:



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

Convolutions: More detail

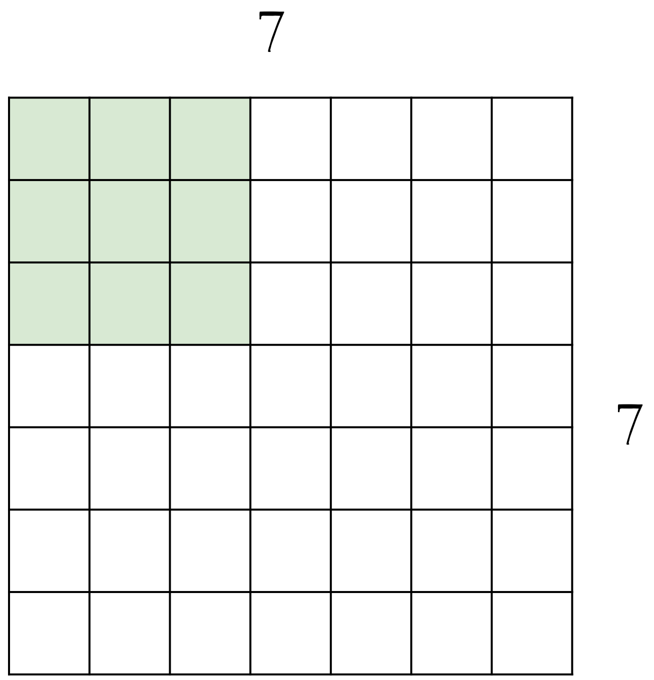
A closer look at spatial dimensions:



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

Convolutions: More detail

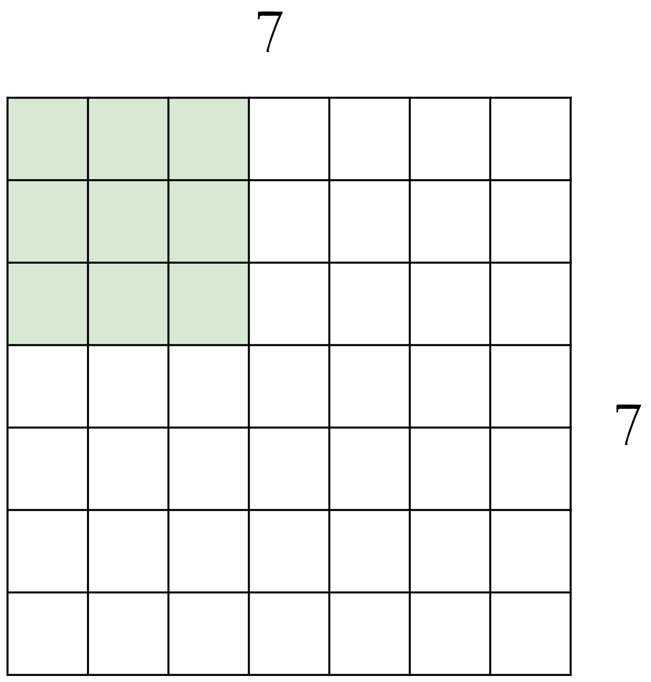
A closer look at spatial dimensions:



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

Convolutions: More detail

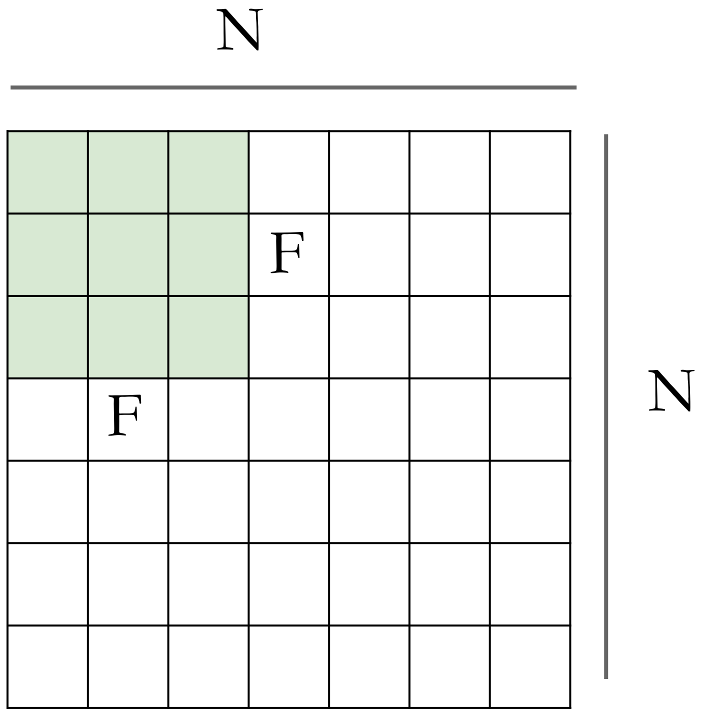
A closer look at spatial dimensions:



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

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

Convolutions: More detail

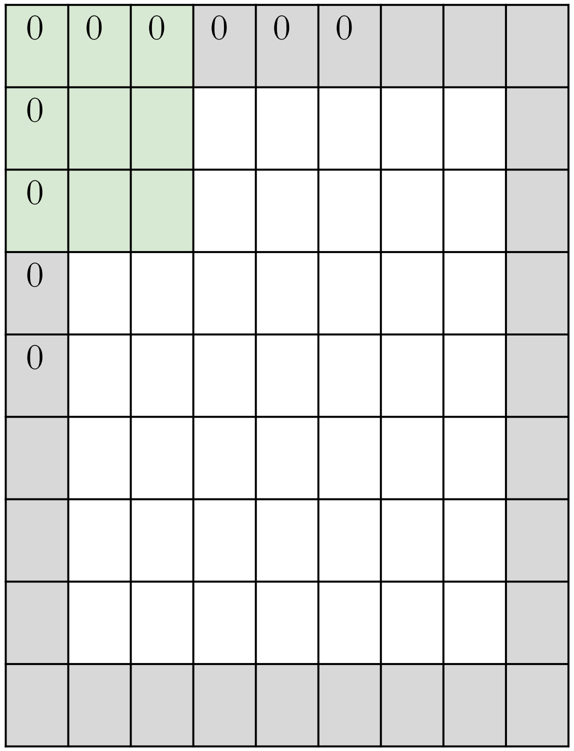


Output size:
(N - F) / stride + 1

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 :\

Convolutions: More detail

In practice: Common to zero pad the border



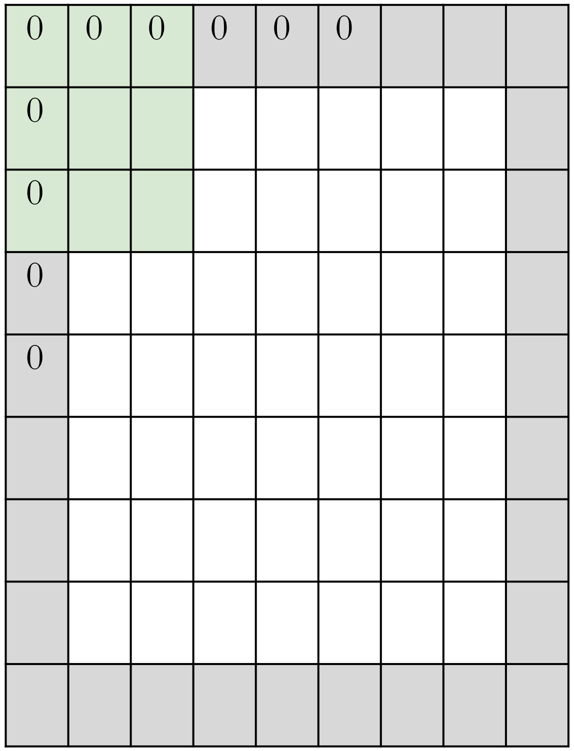
e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

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

Andrej Karpathy

Convolutions: More detail

In practice: Common to zero pad the border

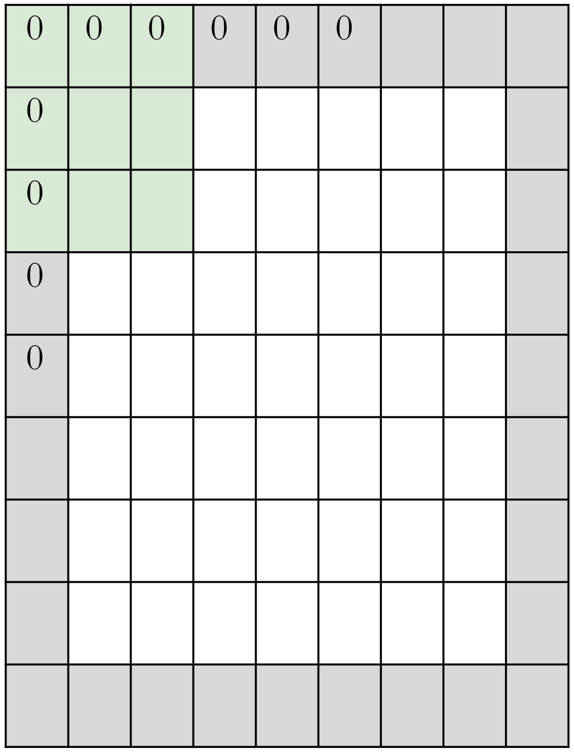


e.g. input 7x7
3x3 filter, applied with **stride 1**
pad with 1 pixel border => what is the output?

7x7 output!

Convolutions: More detail

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?

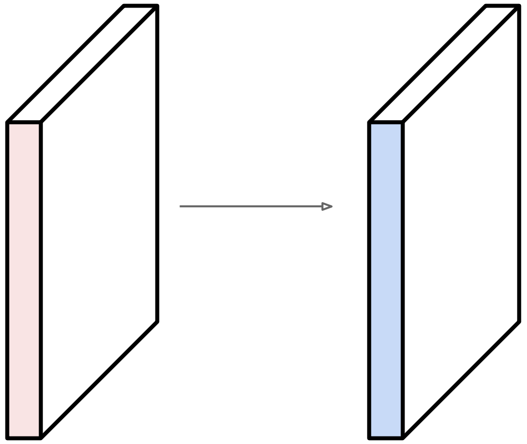
7x7 output!
in general, common to see CONV layers with stride 1, filters of size FxF, and zero-padding with (F-1)/2. (will preserve size spatially)
e.g. F = 3 => zero pad with 1 F
= 5 => zero pad with 2 F
= 7 => zero pad with 3

(N + 2*padding - F) / stride + 1

Andrej Karpathy

Convolutions: More detail

Examples time:



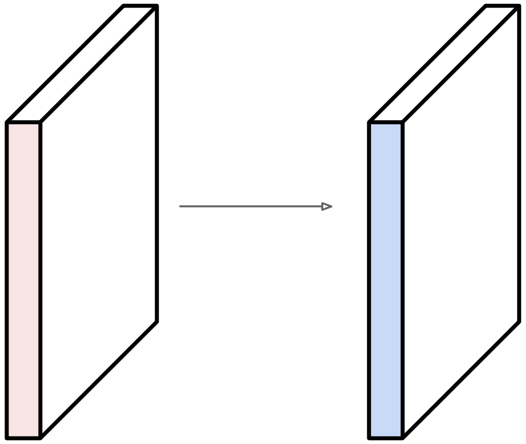
Input volume: **32x32x3**

10 5x5x3 filters with stride 1, pad 2

Output volume size: ?

Convolutions: More detail

Examples time:



Input volume: **32x32x3**

10 5x5x3 filters with stride 1, pad 2

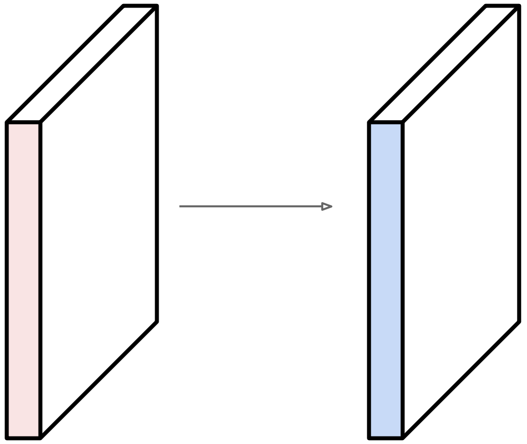
Output volume size:

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

32x32x10

Convolutions: More detail

Examples time:



Input volume: **32x32x3**
10 5x5x3 filters with stride 1, pad 2

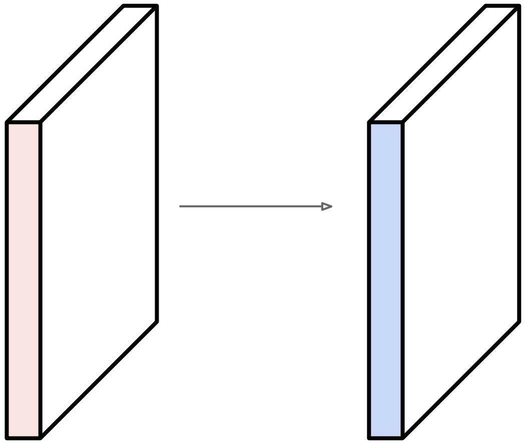
Number of parameters in this layer?

Convolutions: More detail

Examples time:

Input volume: **32x32x3**

10 5x5x3 filters with stride 1, pad 2

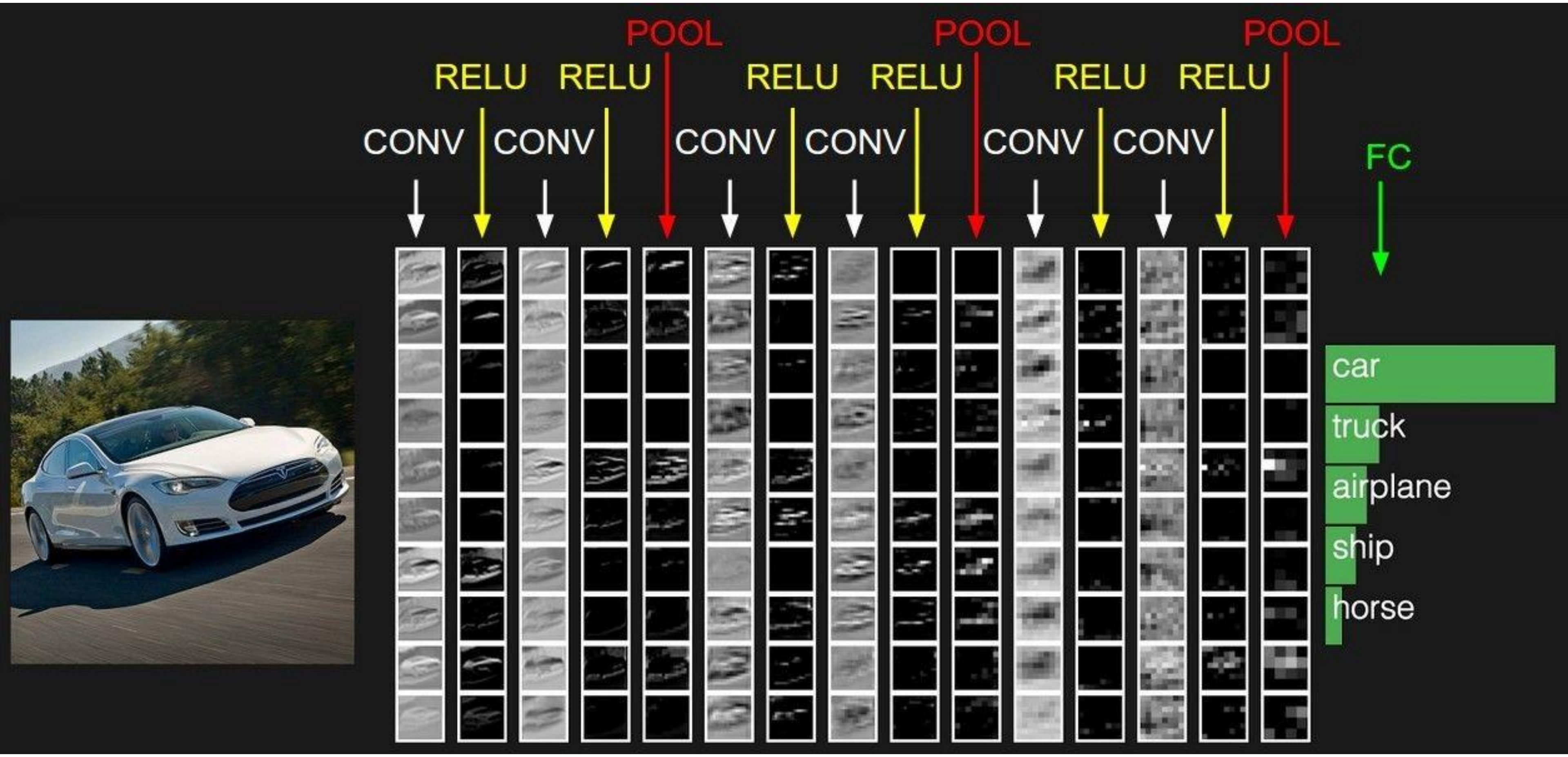


Number of parameters in this layer? each filter has 5*5*3 + 1 = 76 params

(+1 for bias)

=> 76*10 = **760**

Putting it all together



Andrej Karpathy