Convolutional Neural Network

ISTD 50.035

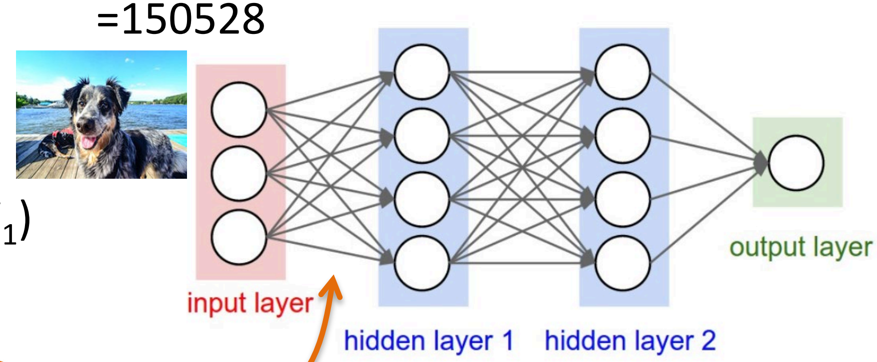
Computer Vision

Acknowledgement: Some images are from various sources: UCF, Stanford cs231n, etc.

Convolutional Neural Network

- CNN: similar to ordinary NN
- In most cases, the inputs are images
- Special network architecture for images
  - Less computation in the forward pass
  - Reduce the number of parameter
  - Better accuracy

224x224x3
=150528



If using hidden layer of similar size, approximately $1\text{x}10^{10}$ parameters (only W_1)

Revisit Image Filtering

- Correlation / convolution (precisely, there are subtle differences)

Correlation / convolution

$$f \otimes h = \sum_k \sum_l f(k,l)\,h(k,l)$$

f = Image

h = Kernel

$$\underset{f}{\begin{array}{|c|c|c|} \hline f_1 & f_2 & f_3 \\ \hline f_4 & f_5 & f_6 \\ \hline f_7 & f_8 & f_9 \\ \hline \end{array}} \otimes \underset{h}{\begin{array}{|c|c|c|} \hline h_1 & h_2 & h_3 \\ \hline h_4 & h_5 & h_6 \\ \hline h_7 & h_8 & h_9 \\ \hline \end{array}} \longrightarrow$$

$$\begin{aligned} f \otimes h = {} & f_1 h_1 + f_2 h_2 + f_3 h_3 \\ & + f_4 h_4 + f_5 h_5 + f_6 h_6 \\ & + f_7 h_7 + f_8 h_8 + f_9 h_9 \end{aligned}$$

Image Filtering

- Filtering (convolution) operation

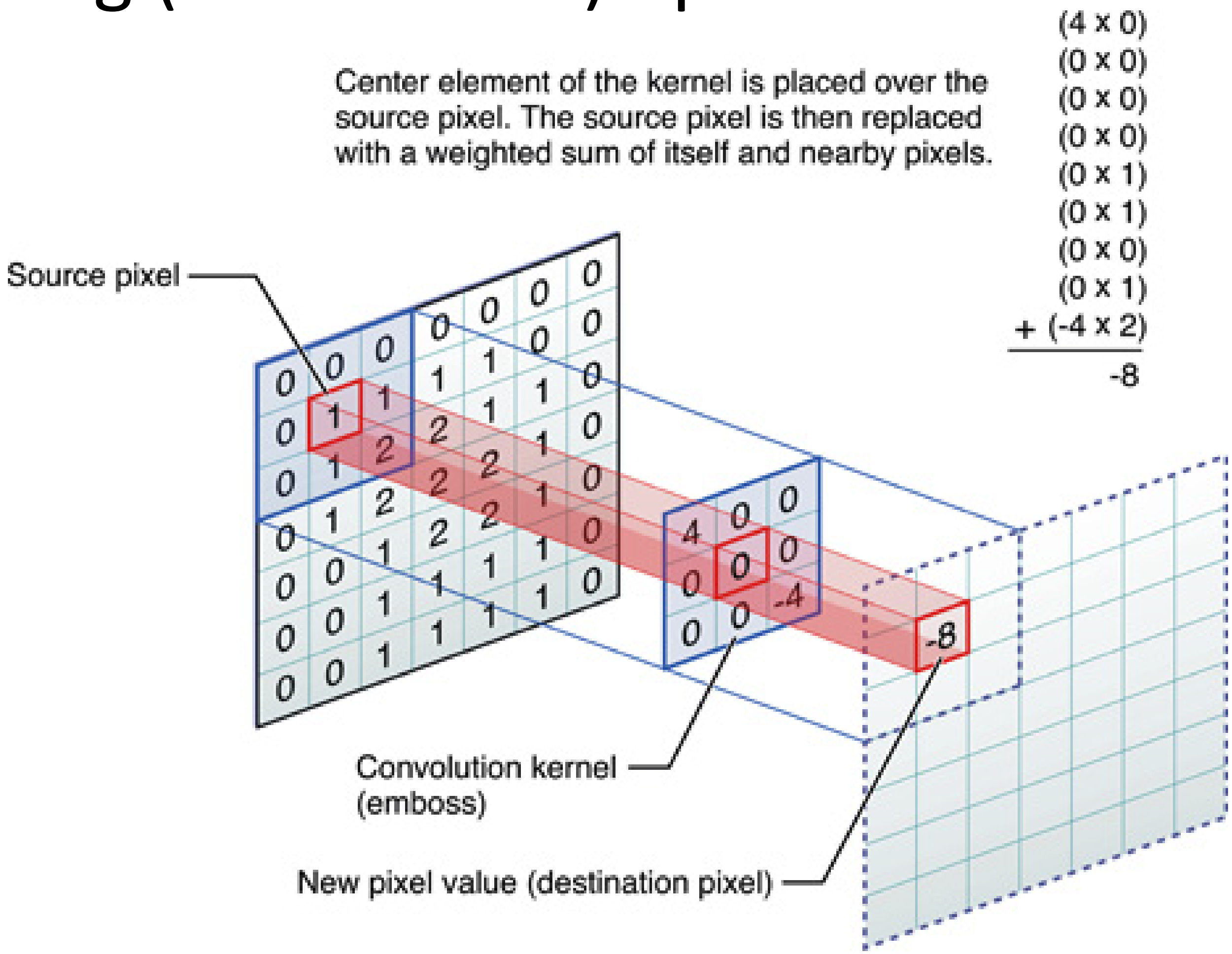


Image Filtering

- Filtering (convolution) operation
- Slide the filter kernel over the entire image to produce the output (image/activation)

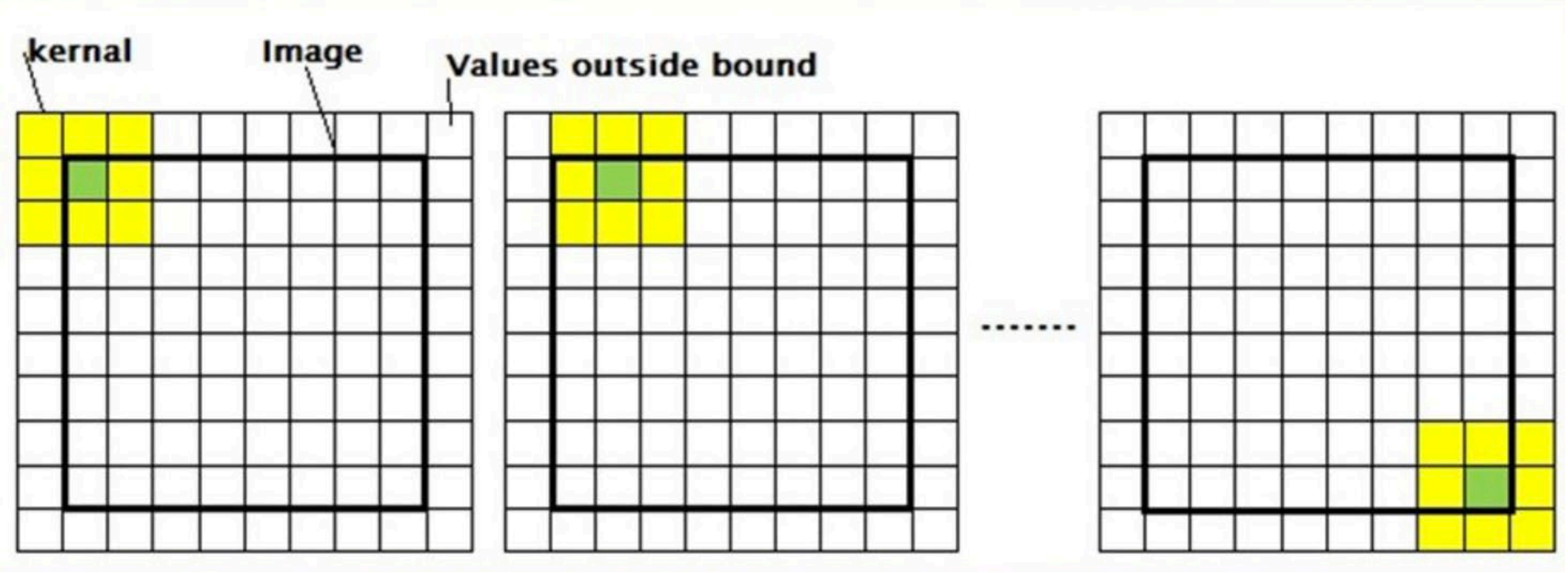
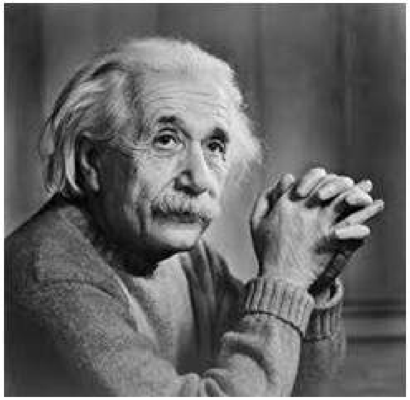


Image Filtering

- Filtering as feature detection / template matching



[[-1,0,1],
[-1,0,1],
[-1,0,1]]

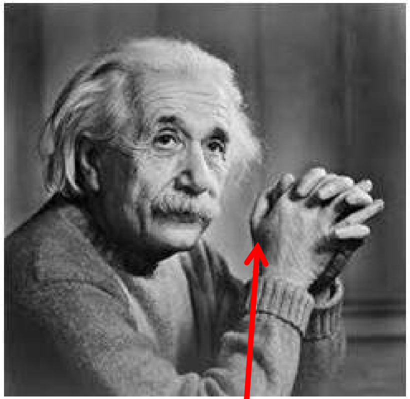
Detect vertical edge

[[-1,-1,-1],
[0, 0, 0],
[1, 1, 1]]

Detect horizontal edge

Image Filtering

- Filtering as feature detection / template matching



Activation / feature map

Vertical edge is detected here (large output)

[[-1,0,1],
[-1,0,1],
[-1,0,1]]

Detect vertical edge

[[-1,-1,-1],
[0, 0, 0],
[1, 1, 1]]

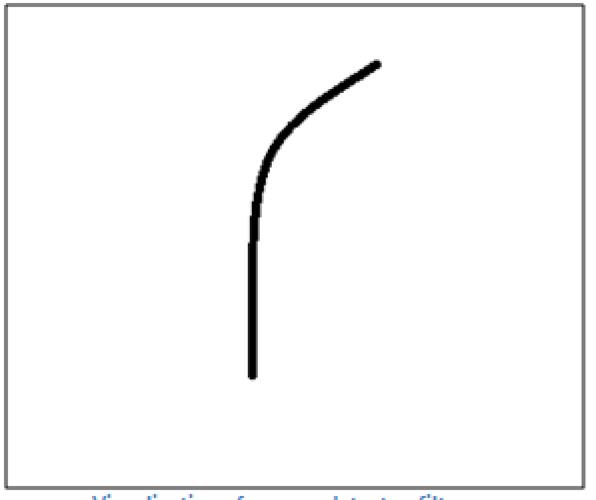
Detect horizontal edge

Image Filtering

- Filtering as feature detection / template matching

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

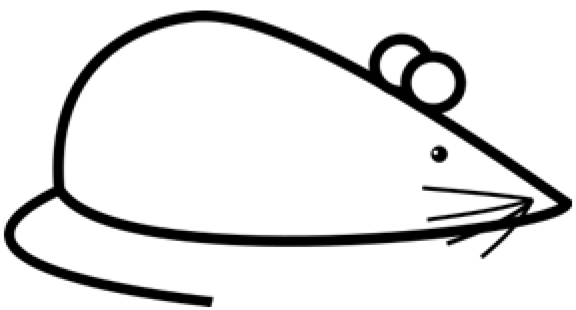


Visualization of a curve detector filter

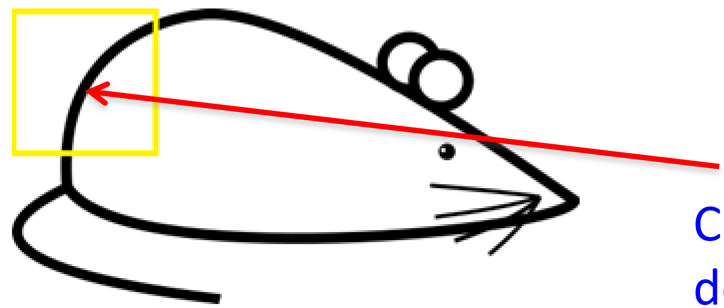
Generalize to curve detector

Image Filtering

- Filtering as feature detection / template matching

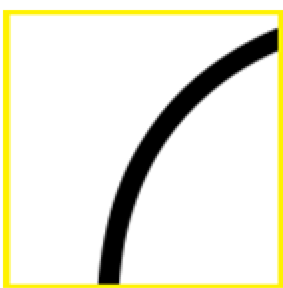


Original image



Visualization of the filter on the image

Curve is detected here (large output)



Visualization of the receptive field

0	0	0	0	0	0	30
0	0	0	0	50	50	50
0	0	0	20	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0
0	0	0	50	50	0	0

Pixel representation of the receptive field

*

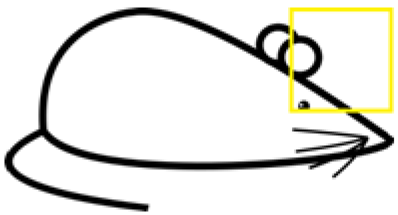
0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = (50*30)+(50*30)+(50*30)+(20*30)+(50*30) = 6600 (A large number!)

Image Filtering

- Filtering as feature detection / template matching



Visualization of the filter on the image

0	0	0	0	0	0	0
0	40	0	0	0	0	0
40	0	40	0	0	0	0
40	20	0	0	0	0	0
0	50	0	0	0	0	0
0	0	50	0	0	0	0
25	25	0	50	0	0	0

Pixel representation of receptive field

$*$

0	0	0	0	0	30	0
0	0	0	0	30	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	30	0	0	0
0	0	0	0	0	0	0

Pixel representation of filter

Multiplication and Summation = 0

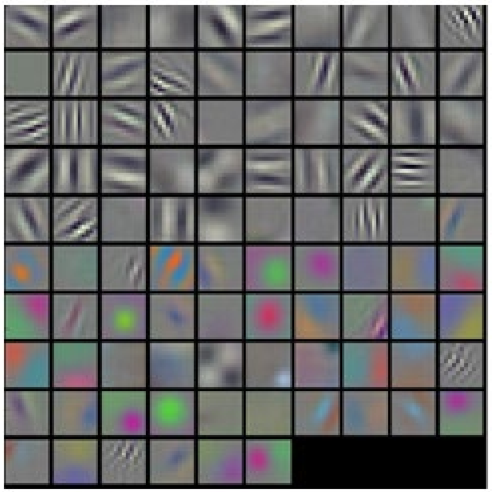
Small output: no curve is detected

Take away: Filtering (convolution) is an efficient mechanism for finding patterns

Filters respond most strongly to pattern elements that look like the filters

Image Filtering

- Filtering as feature detection / template matching

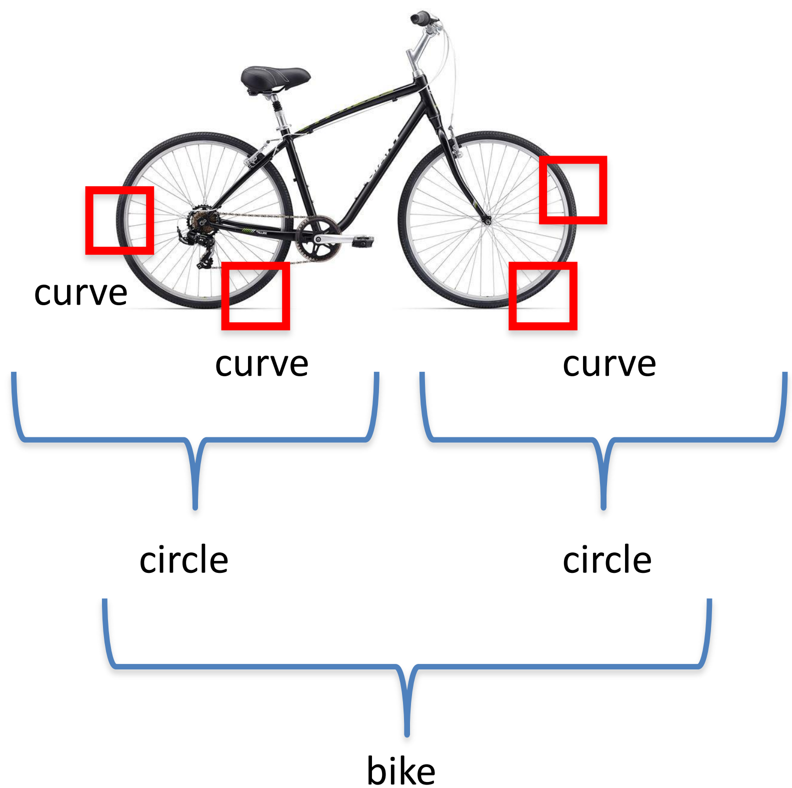


Visualizations of filters

-Need different filter kernels to detect different features
-Data driven approach: use training images to tell us what filter kernels are useful

How to recognize an object?

- Use feature detection (image filtering) in a hierarchical manner



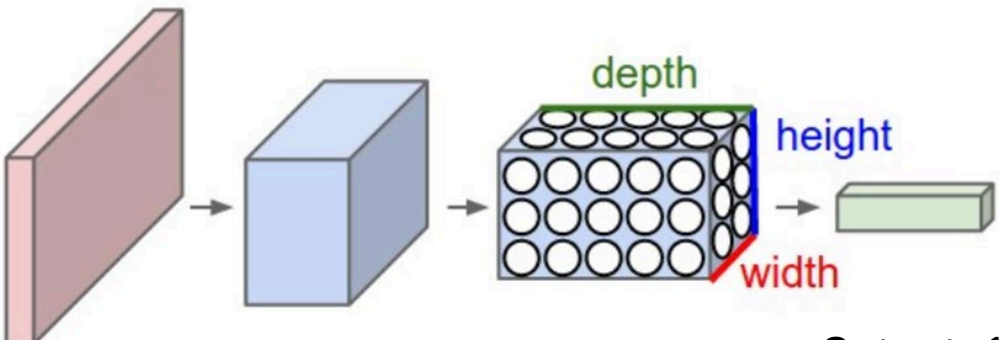
Low level feature

High level feature

Implement this approach using CNN

CNN

- 3D volumes of neurons



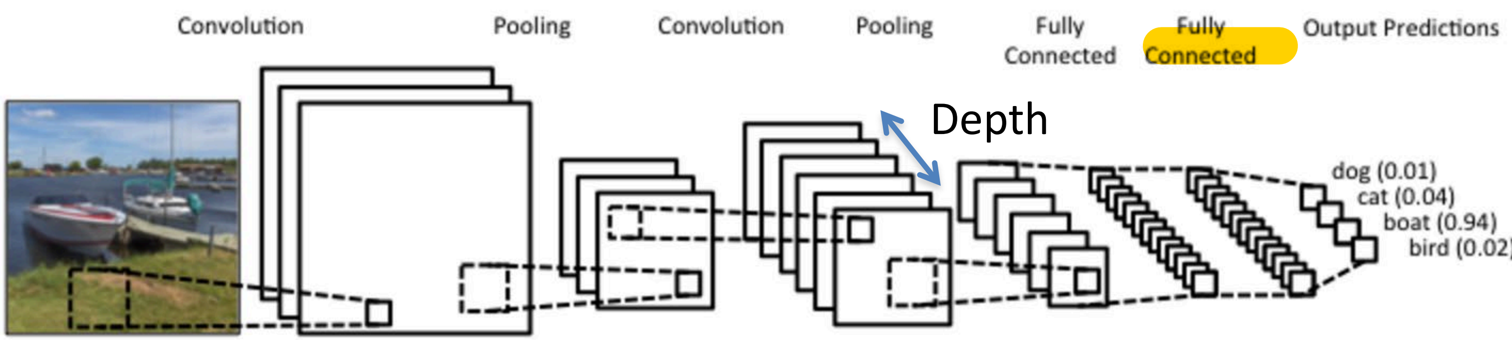
Input: width x height x numChannel

Output: 1 x 1 x numClass

- Stack of
  - Convolutional layer
  - Fully connected layer
  - Pooling layer

CNN

- One example



Input:
width x height x numChannel

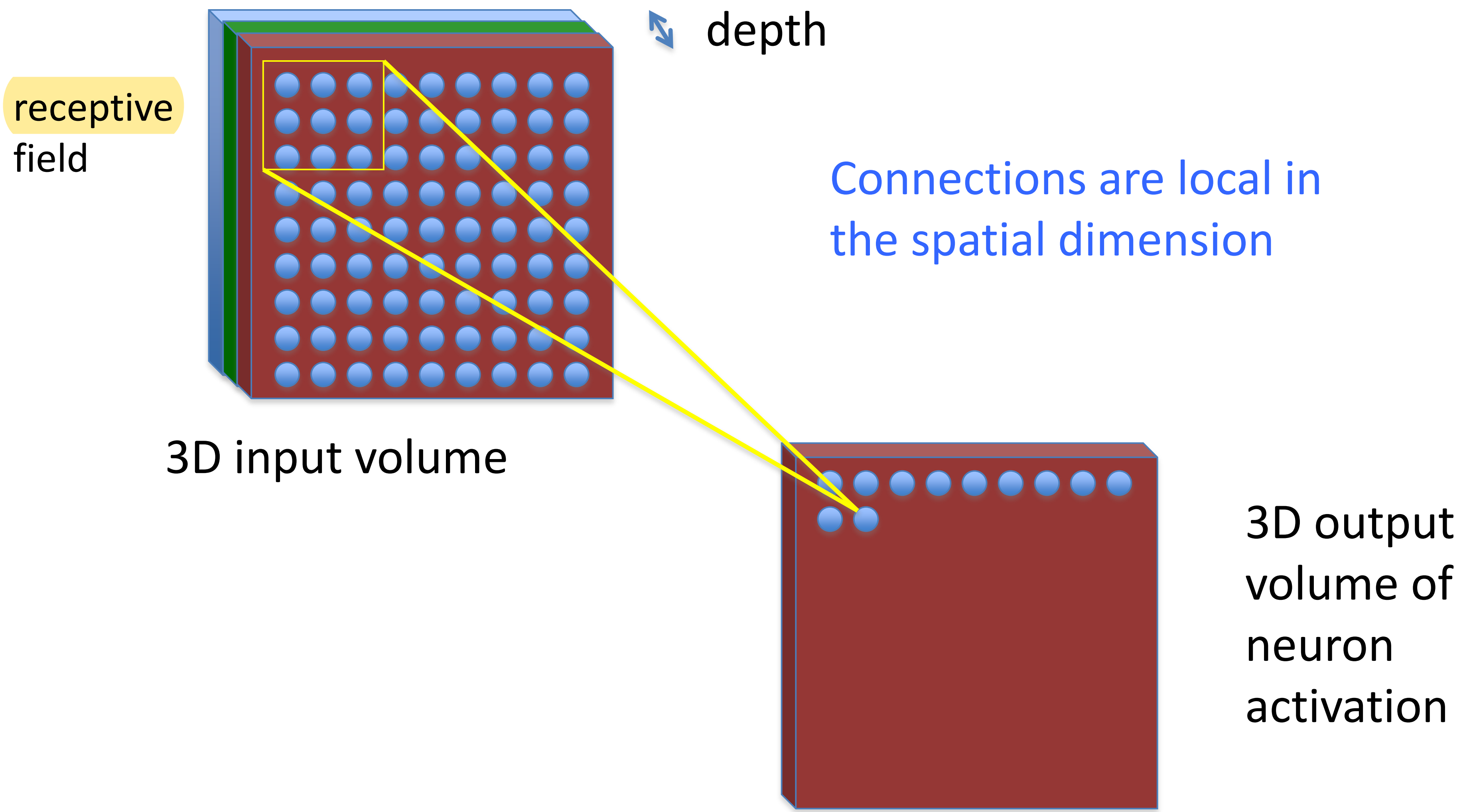
Output:
1 x 1 x numClass

- Stack of
  - Convolutional layer
  - Fully connected layer
  - Pooling layer

Convolutional Layer

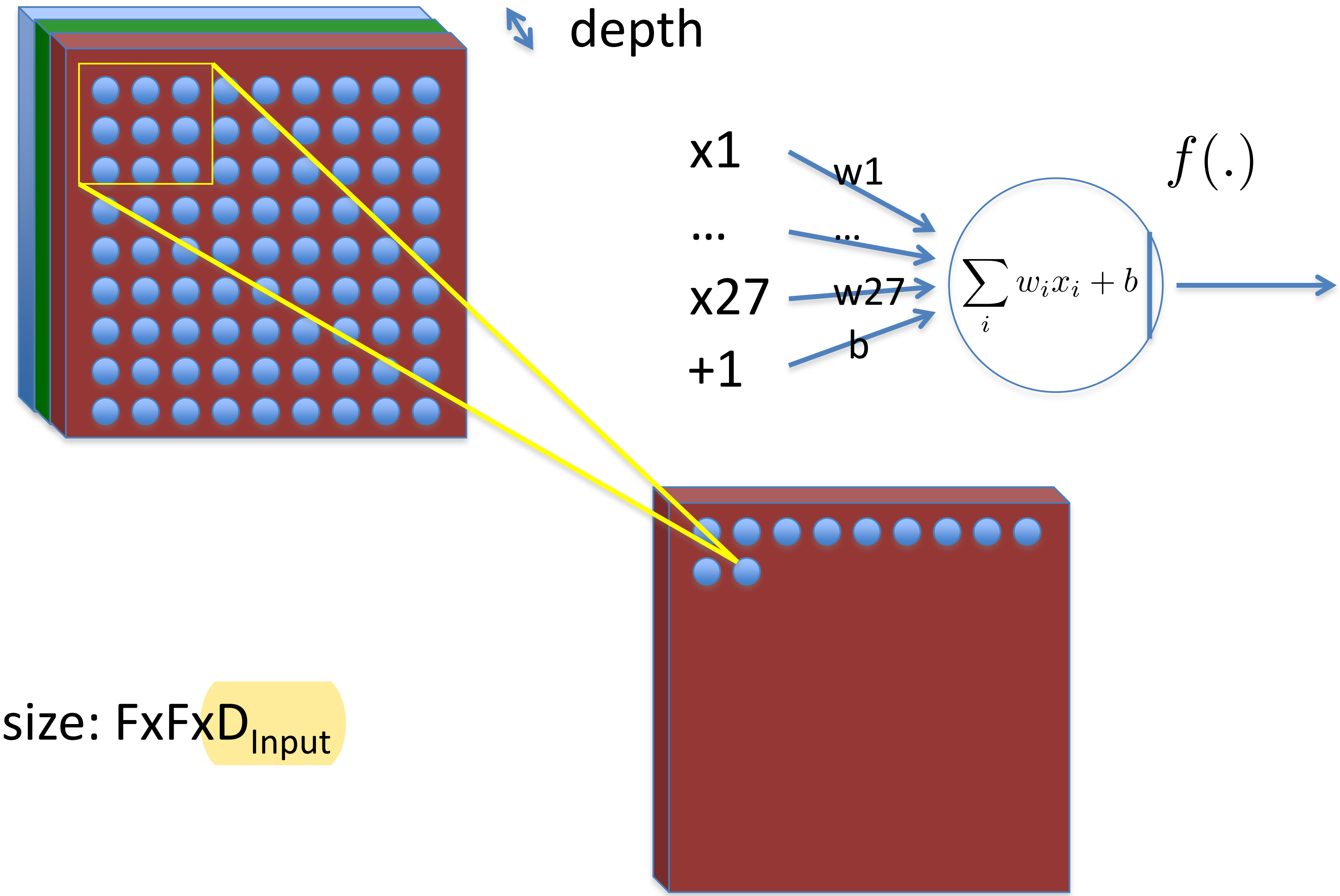
- Conv layer: core component
- Local connectivity
  - Spatial extent: receptive field
  - Extent of connectivity along the depth dimension = depth of the input
- Parameter sharing
- Filtering / convolution
  - Instead of matrix multiplication

Convolutional Layer



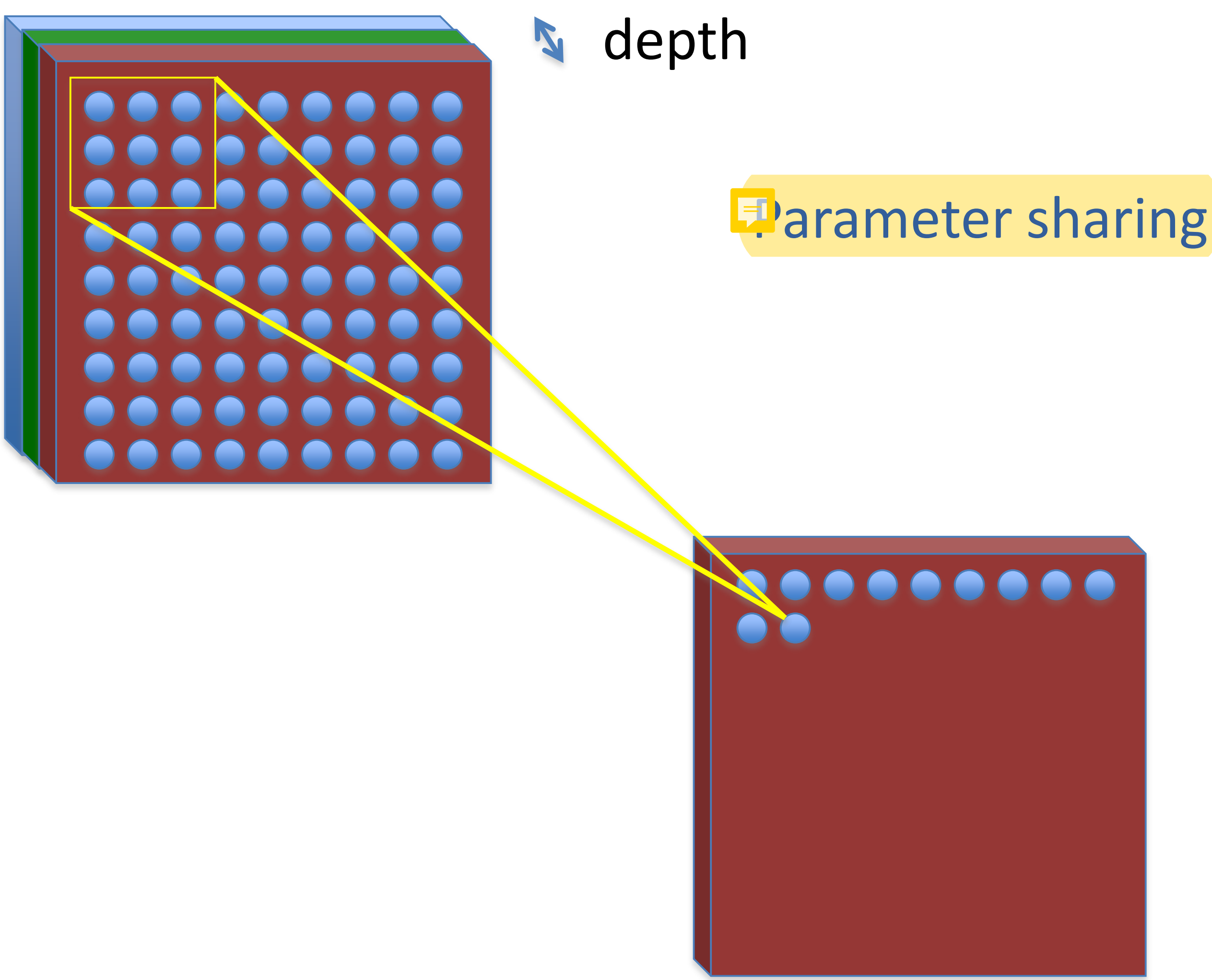
Connections are local in the spatial dimension

Convolutional Layer

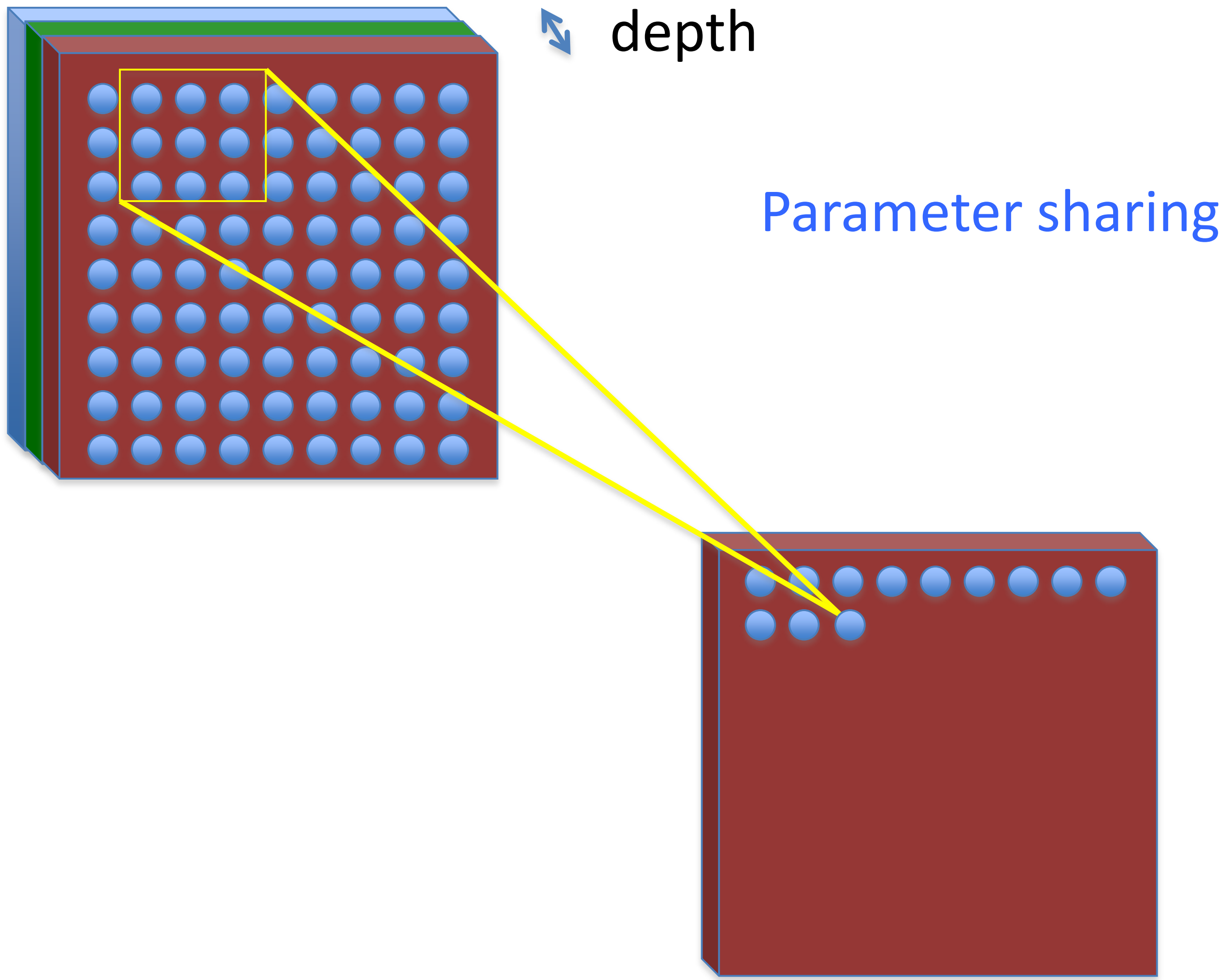


Filter size: $FxFxD_{Input}$

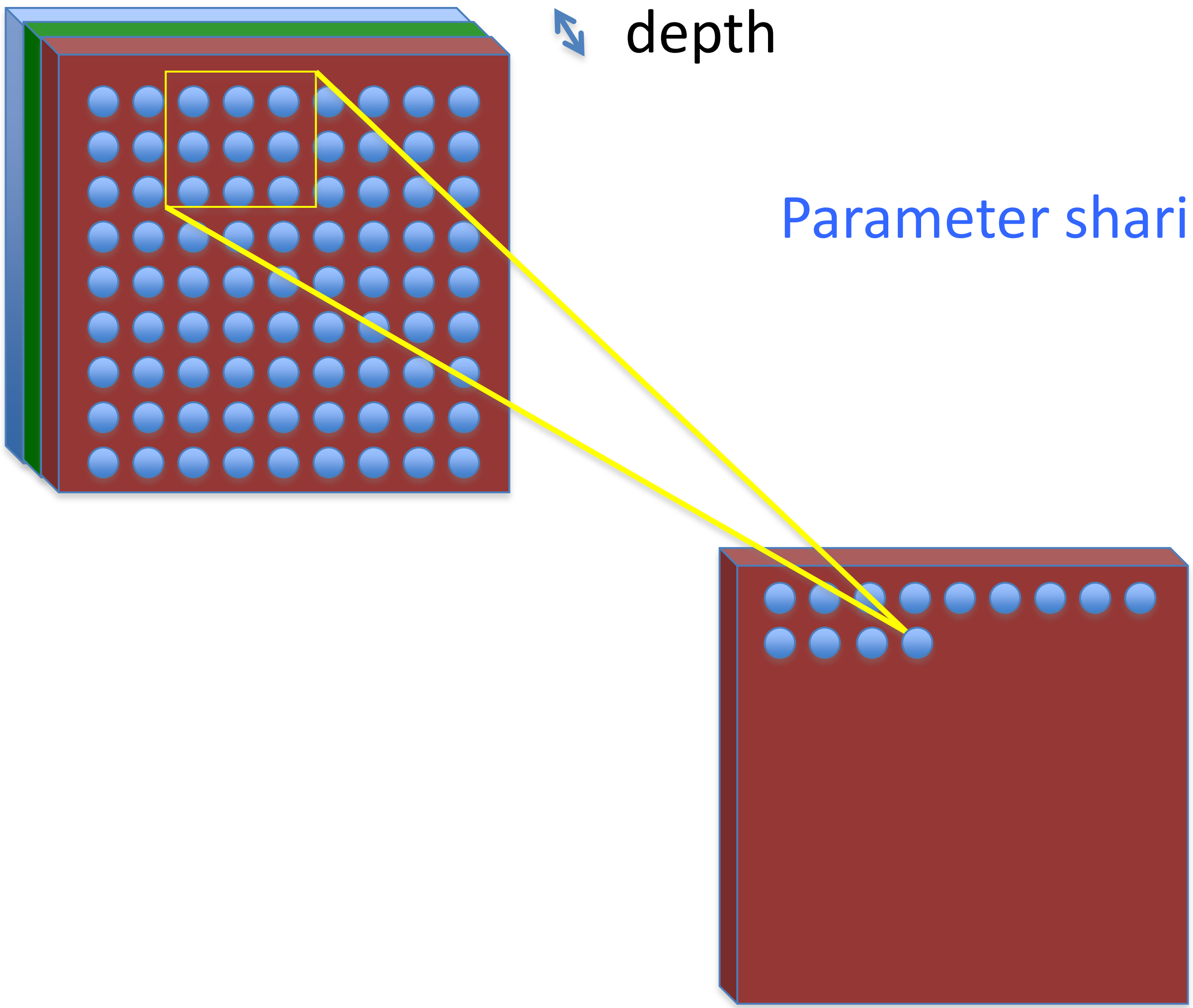
Convolutional Layer



Convolutional Layer

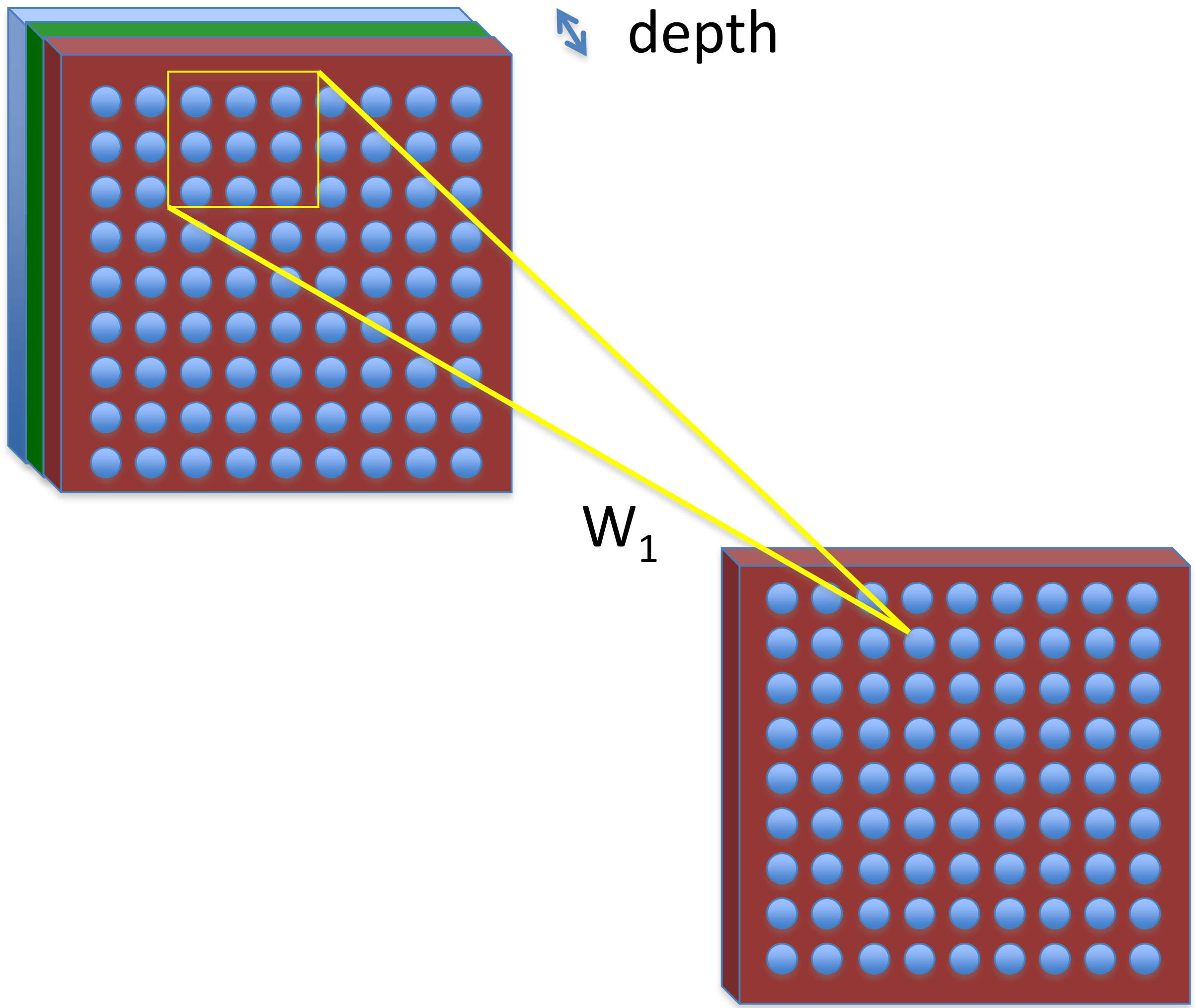


Convolutional Layer

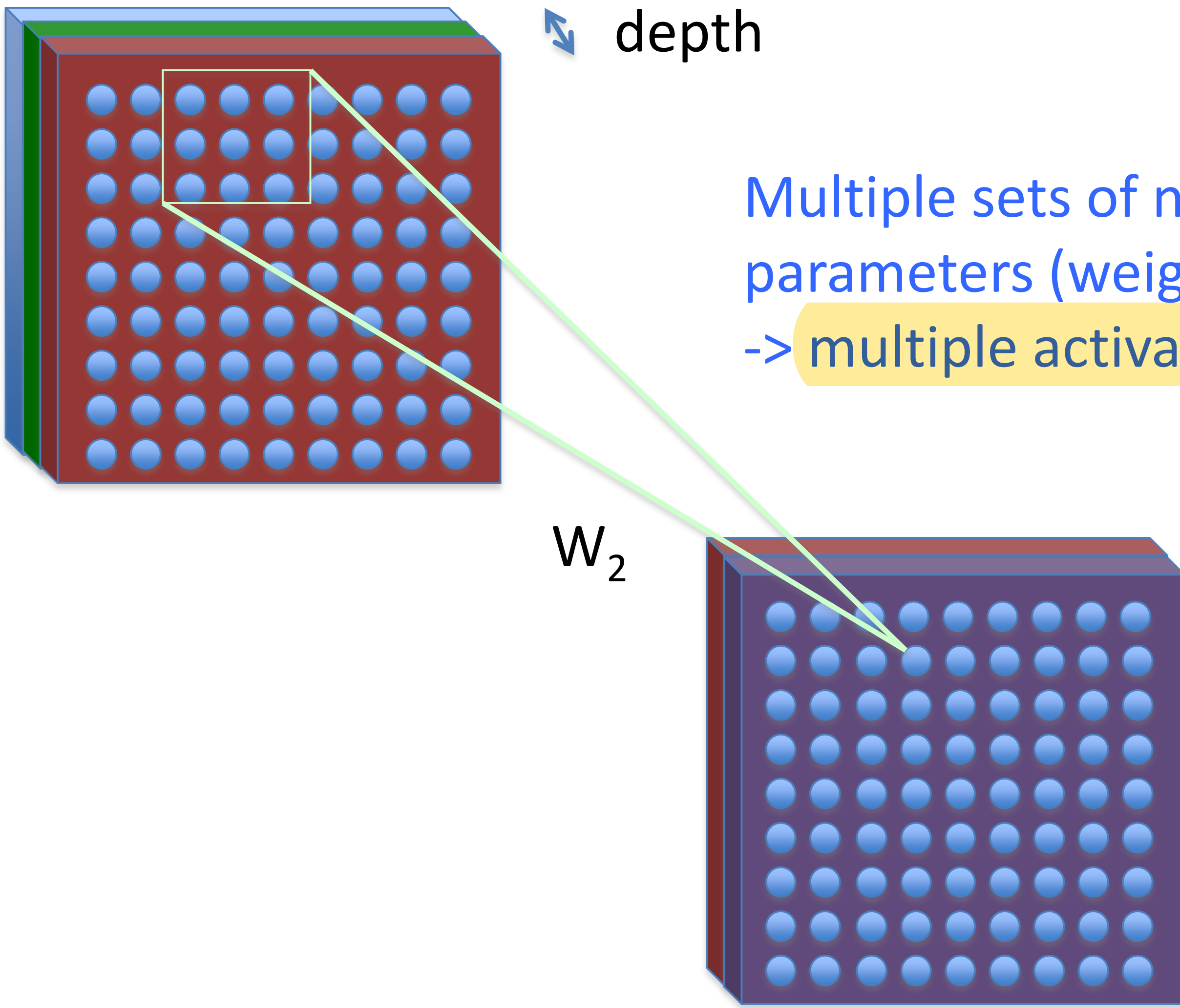


Parameter sharing

Convolutional Layer

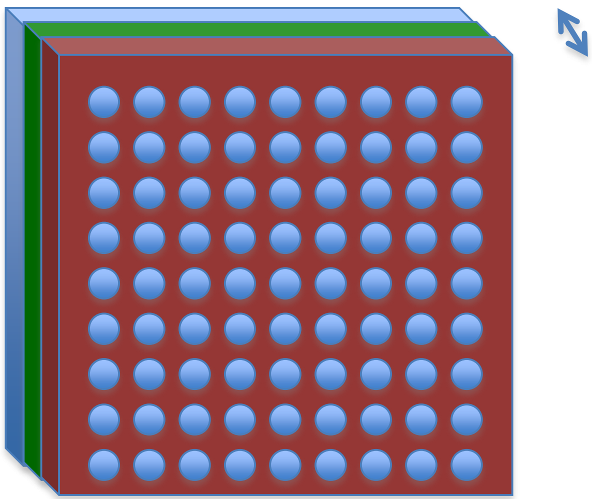


Convolutional Layer



Multiple sets of neuron parameters (weights and bias) -> multiple activation maps

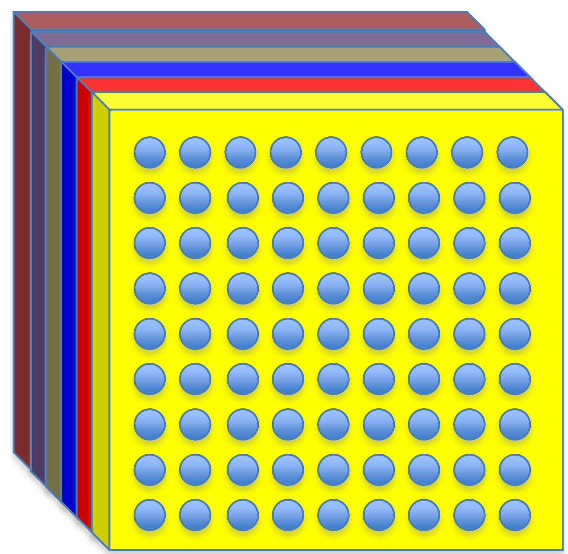
Convolutional Layer



depth

Multiple sets of neuron parameters (weights and bias) -> multiple activation maps

3D input volume



3D output volume of neuron activation

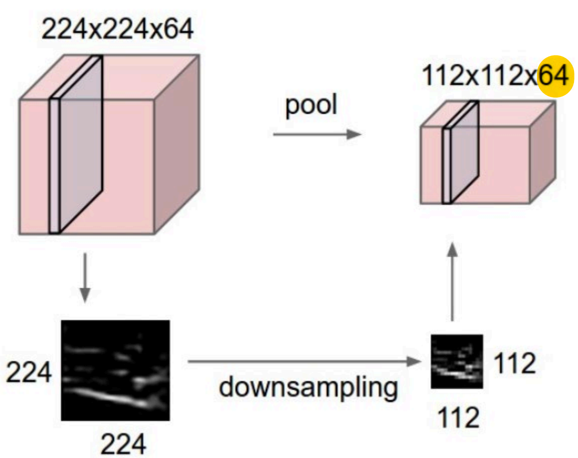
Num of filter kernels = num of output activation maps (depth)

Convolutional Layer

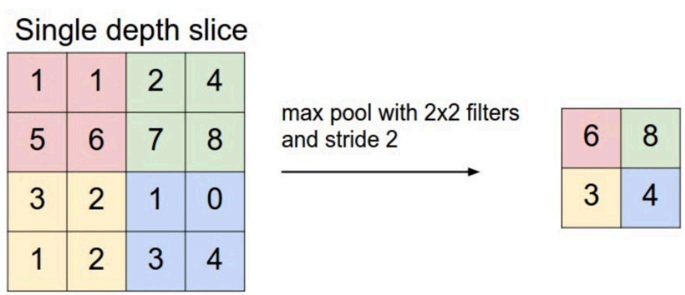
- Filtering (Convolution)
- Matched Filter to identify certain image features
  - Edges or corners (low level layers)
  - Faces or cars (high level layers)
- Assumption of image
  - Locality of pixel dependencies
  - Stationary of image statistics
  - Translation invariance
  - Use the same set of filters for the whole image

Pooling layer

- Progressively reduce the spatial size of the feature map
- Reduce model parameters
- Operate independently on every feature map of the input
- Overlap or non-overlap
- Average or max pooling
- Translation invariant: same pooled feature even when the image undergoes small translations
  - Same label even when the image is translated

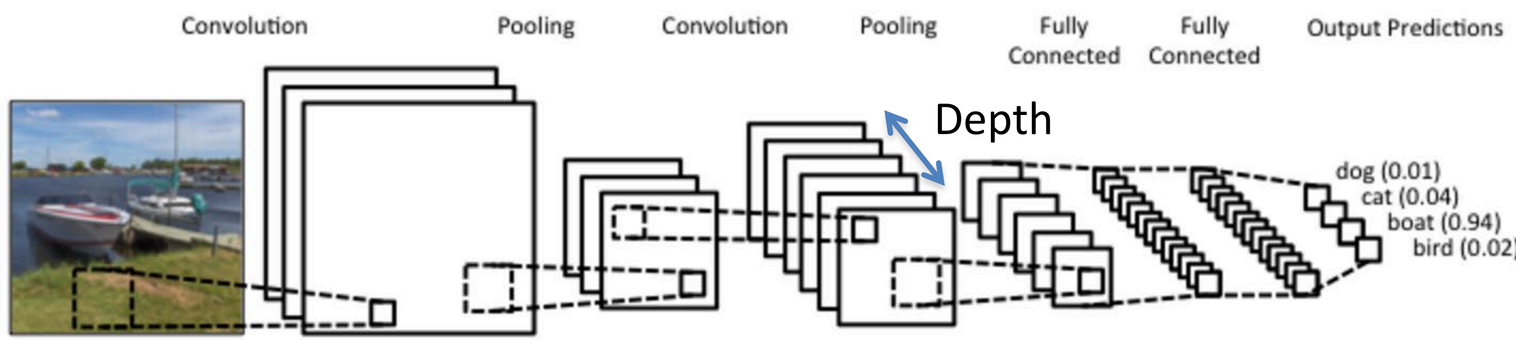


max pooling



CNN

- Stack the layers



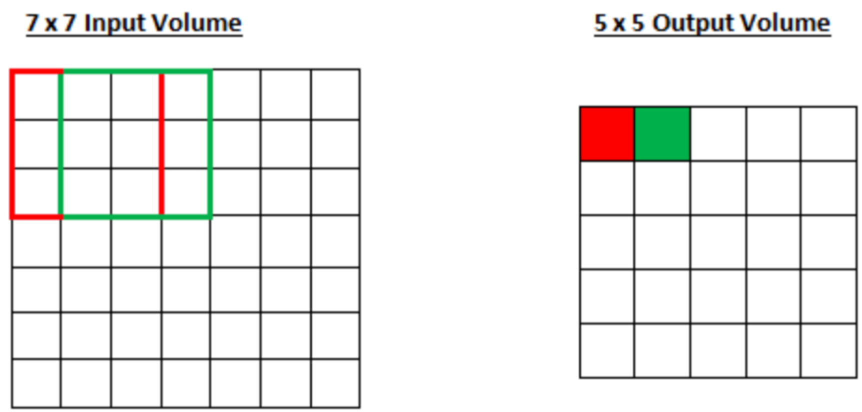
Input:
width x height x numChannel

Output:
1 x 1 x numClass

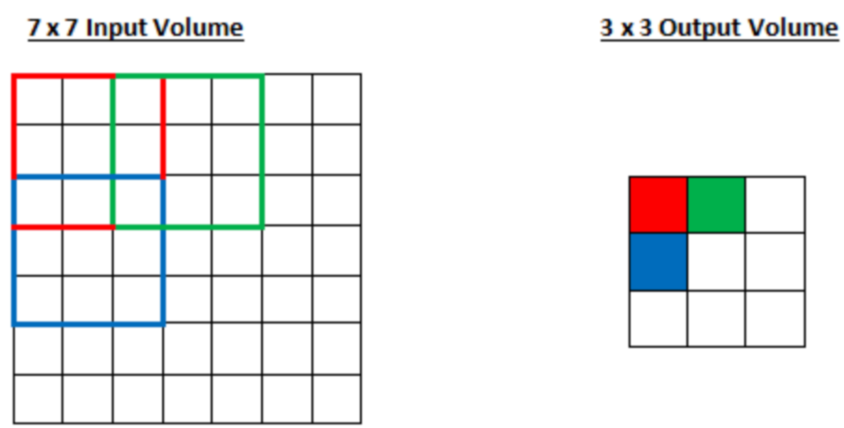
Stride

- Stride = the number of pixels (input units) by which the filter shifts

Stride = 1



Stride = 2



Receptive field

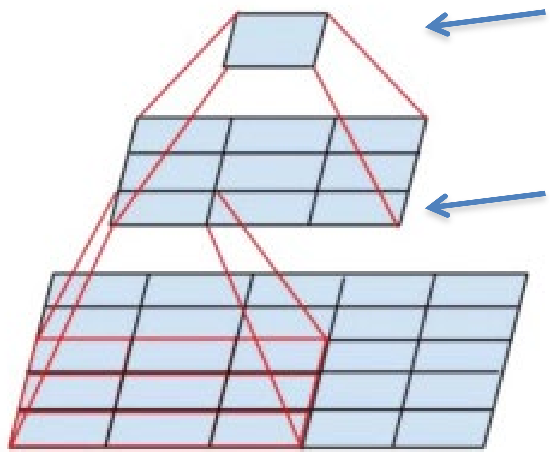
Receptive field: part of the image that is visible to a neuron
Inspired by visual cortex architecture

Stride = 1

Conv 2

Conv 1

Input layer (image)



- 3x3 receptive field size w.r.t. conv1
- 5x5 receptive field size w.r.t. input image

3x3 receptive field size (filter size)

Although connections are local, neurons in the higher layers could see large portions of the image (able to recognize object)