Convolutional Neural Networks

Danna Gurari
University of Colorado Boulder
Spring 2024

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2024/

Review

- Last class:
  - Regularization
  - Parameter norm penalty
  - Early stopping
  - Dataset augmentation
  - Dropout
  - Batch normalization
  - Programming tutorial

- Assignments (Canvas):
  - Problem set 2 due Wednesday

- Questions?

Today’s Topics

- Neural Networks for Spatial Data
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Programming tutorial

Today’s Topics

- Neural Networks for Spatial Data
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Programming tutorial

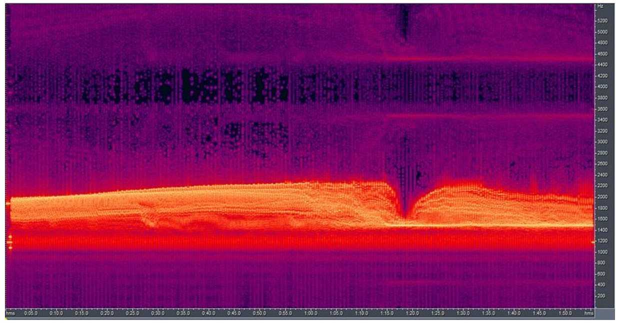
What is Spatial Data?

- Data where the order matters; e.g.,

2D

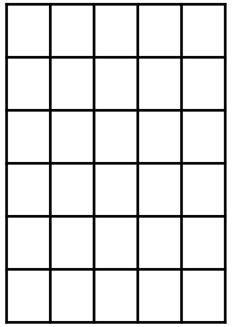
Images

Audio (spectrogram)



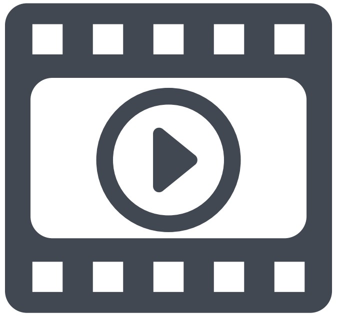
Text (word embeddings)

I
like
learning
about
deep
learning



3D

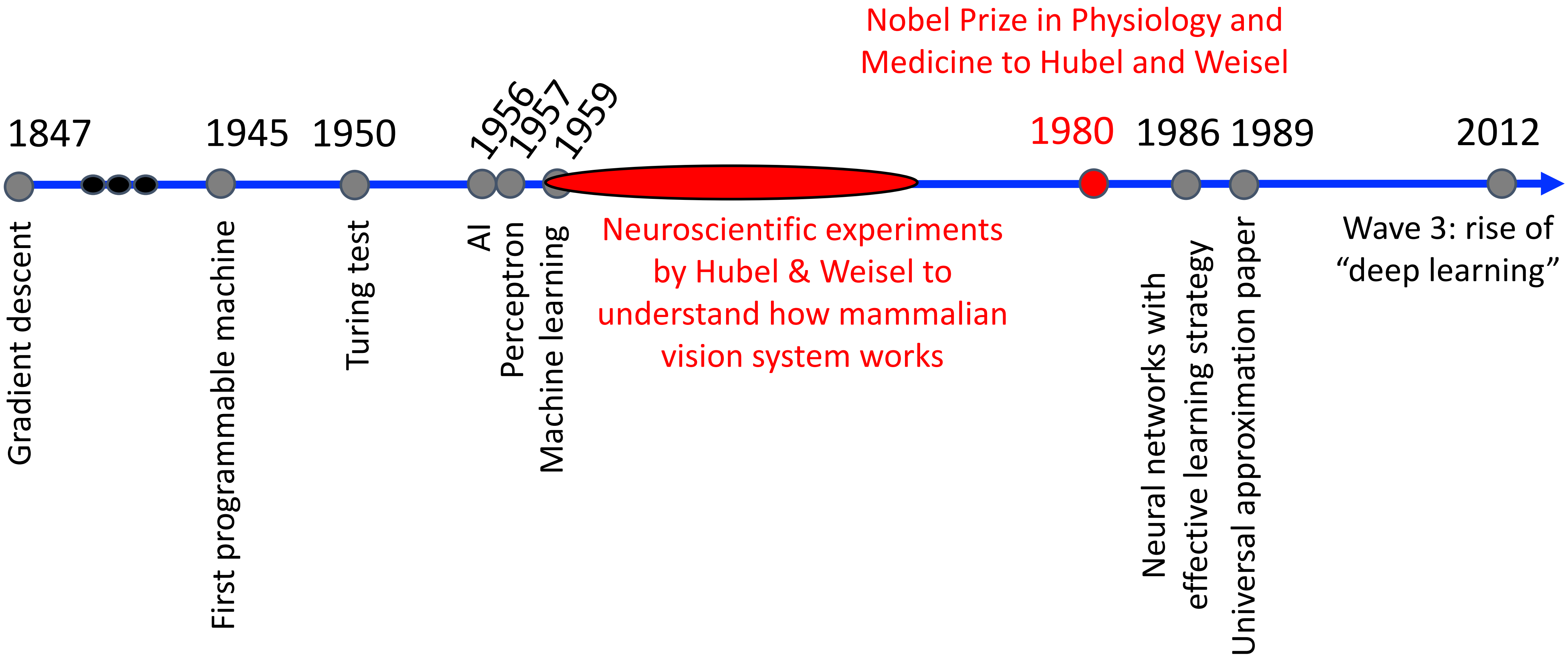
Video



Today’s Topics

- Neural Networks for Spatial Data
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Programming tutorial

Historical Context: Inspiration

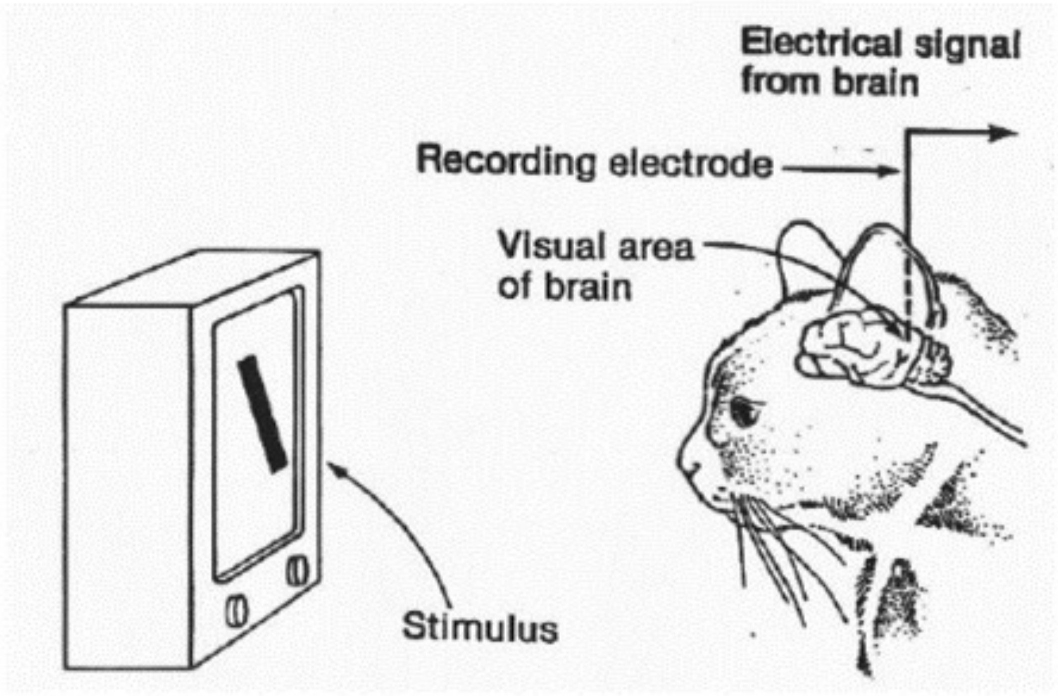


Motivation: How Vision System Works

Image Source: https://braintour.harvard.edu/archives/portfolio-items/hubel-and-wiesel

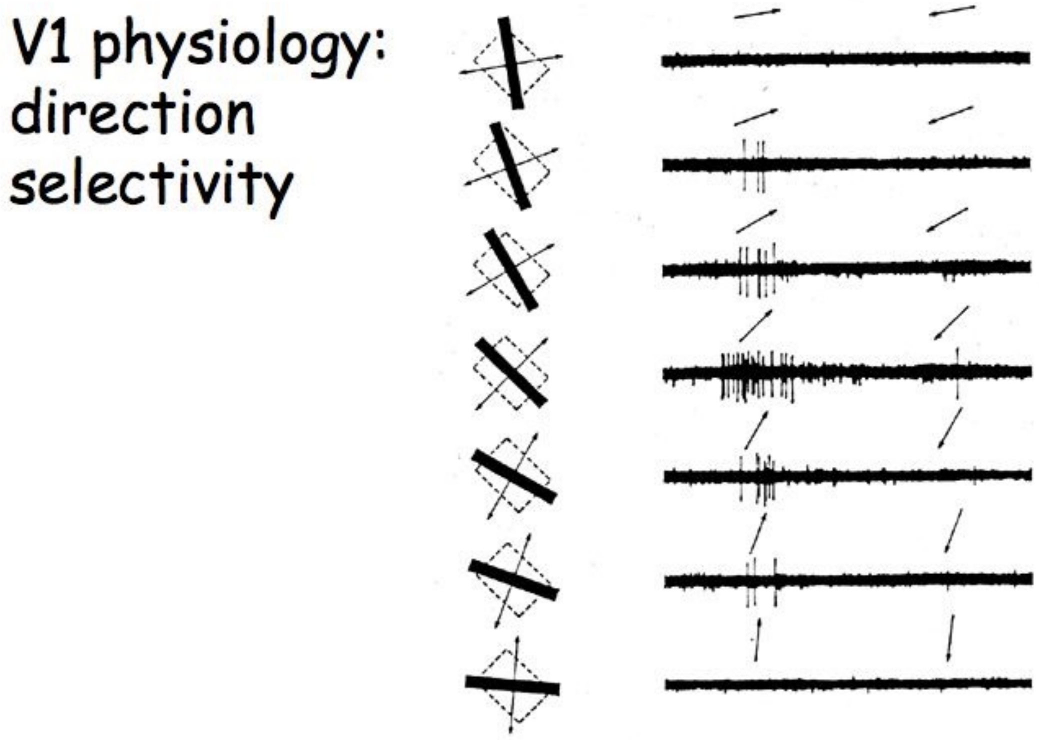
Motivation: How Vision System Works

Experiment Set-up:



https://www.esantus.com/blog/2019/1/31/convolutional-neural-networks-a-quick-guide-for-newbies

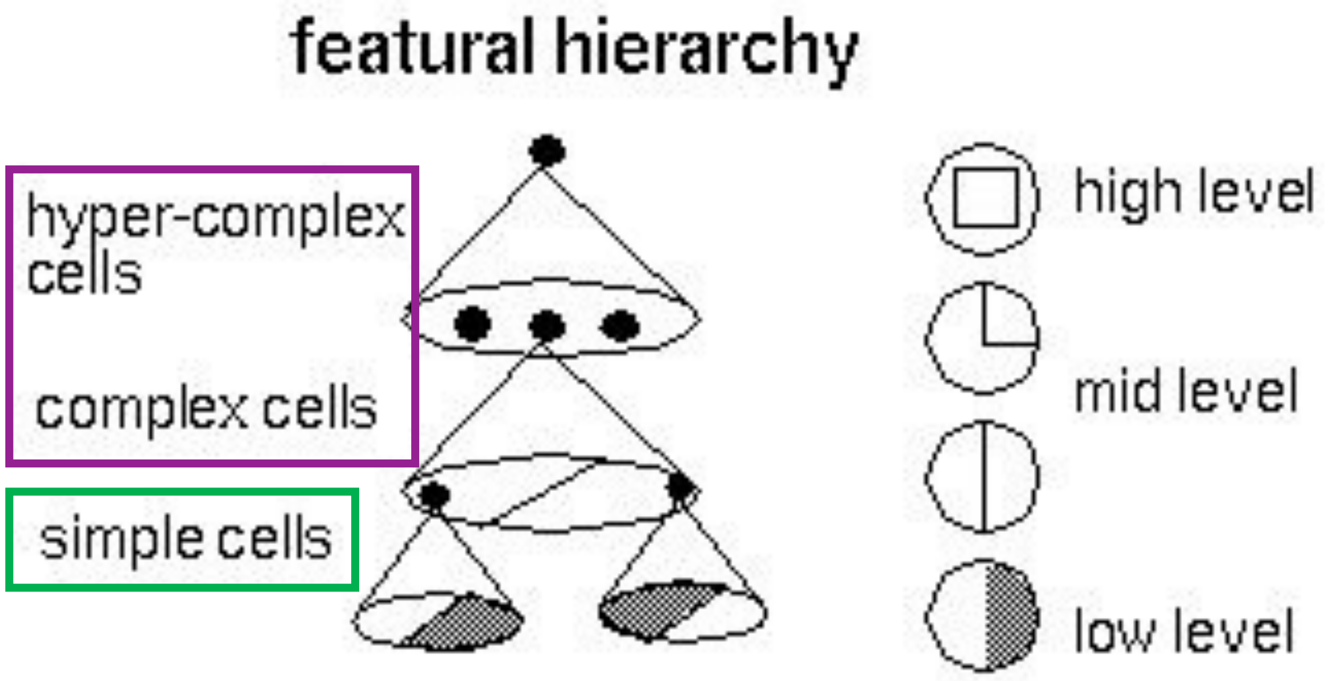
Key Finding: initial neurons responded strongly only when light was shown in certain orientations



https://www.cns.nyu.edu/~david/courses/perception/lecturenotes/V1/lgn-V1.html

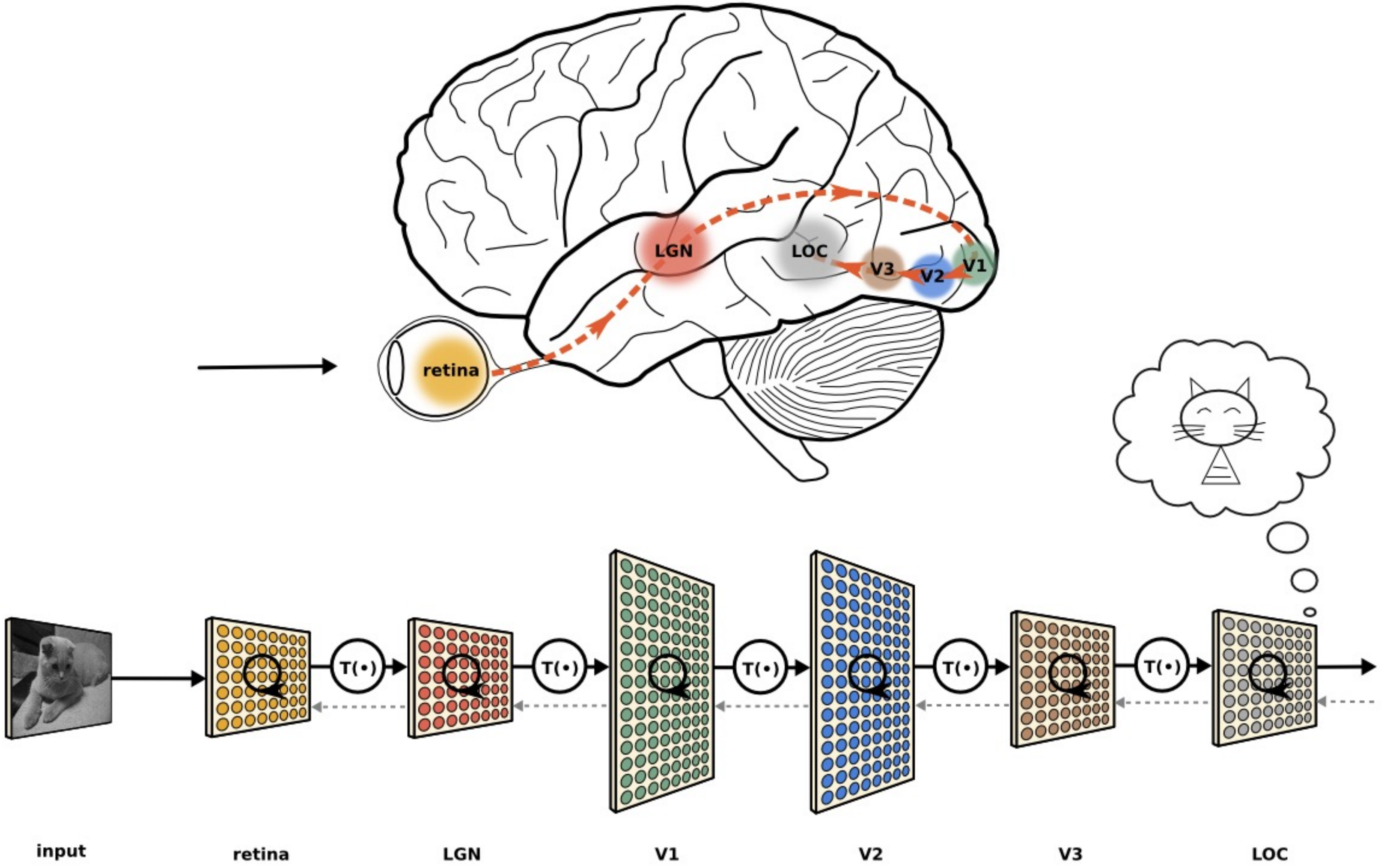
Motivation: How Vision System Works

Key Idea: cells are organized as a hierarchy of feature detectors, with higher level features responding to patterns of activation in lower level cells



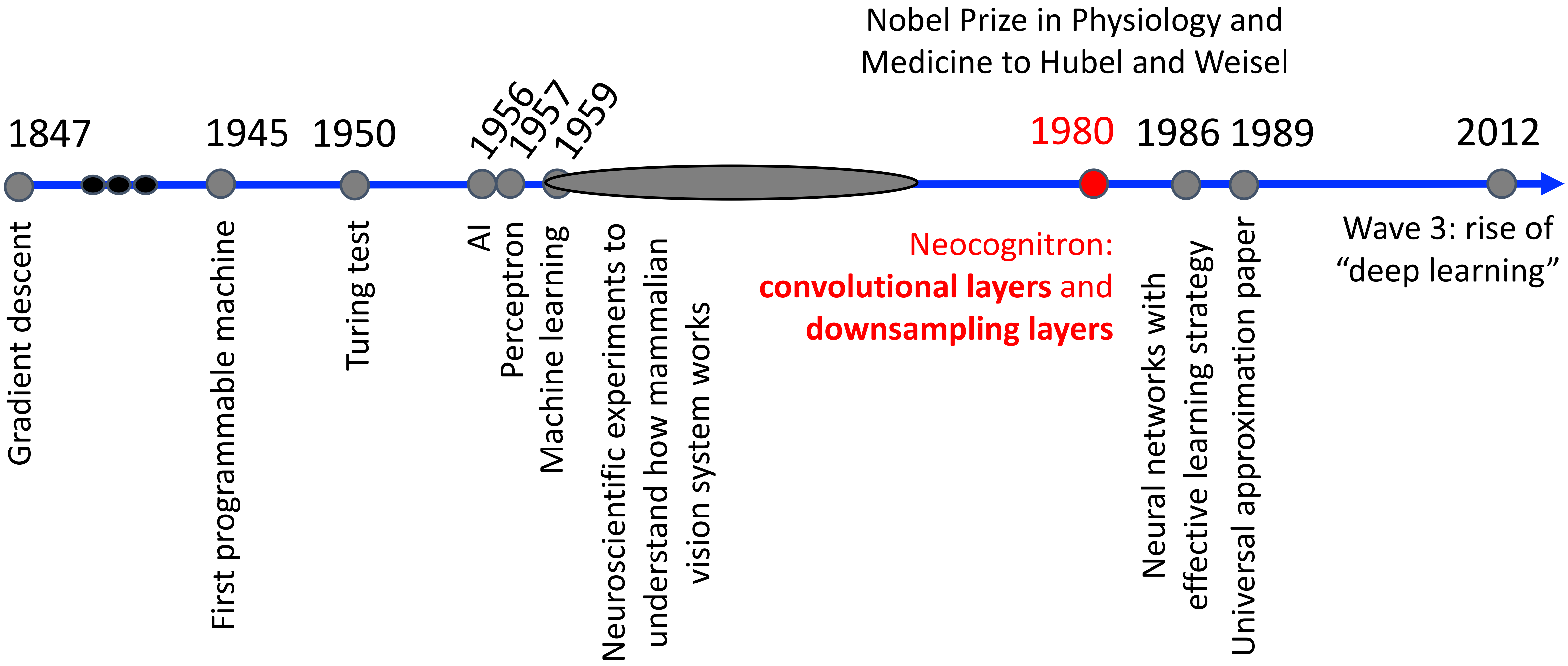
Source: https://bruceoutdoors.files.wordpress.com/2017/08/hubel.jpg

Motivation: How Vision System Works



https://neuwritesd.files.wordpress.com/2015/10/visual_stream_small.png

Historical Context: Key Ingredients



Neocognitron: Key Ingredients

http://personalpage.flsi.or.jp/fukushima/index-e.html

"In this paper, we discuss how to synthesize a neural network model in order to endow it an ability of pattern recognition like a human being… the network acquires a similar structure to the hierarchy model of the visual nervous system proposed by Hubel and Wiesel."

- Fukushima, Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics, 1980.

Neocognitron: Key Ingredients

Cascade of simple and complex cells:

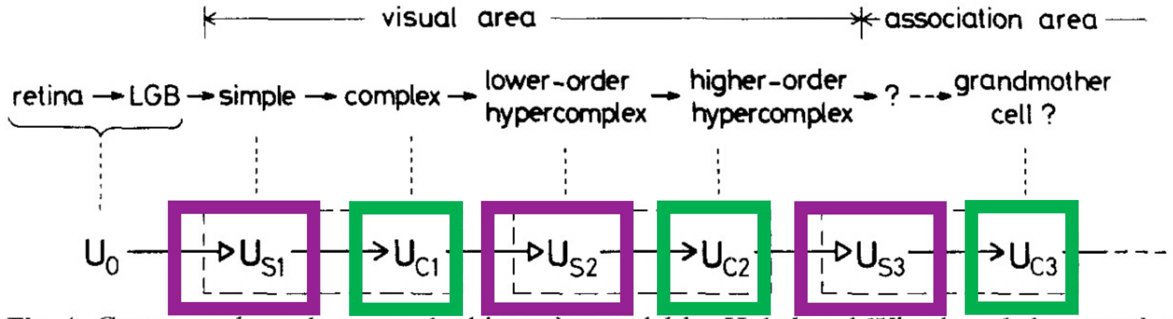


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

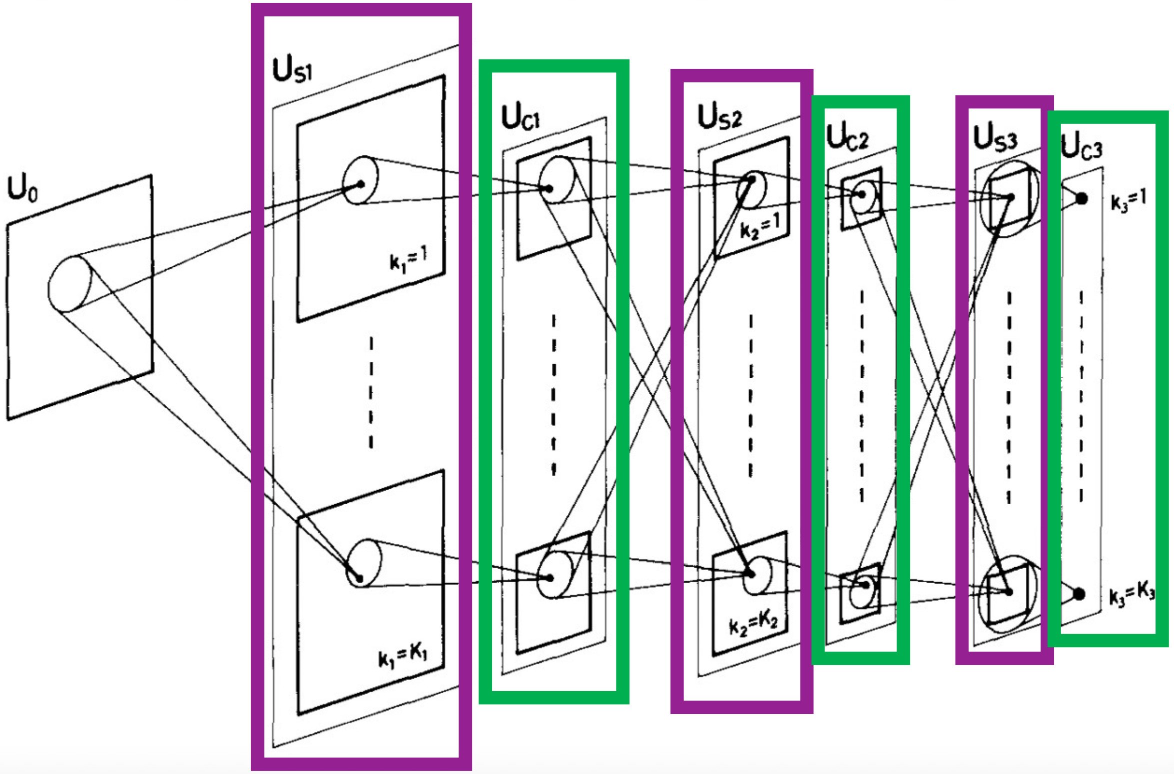
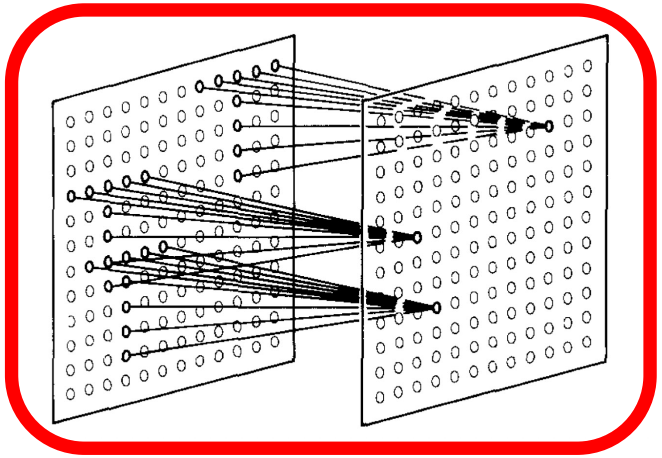


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Neocognitron: Key Ingredients

Simple cells extract local features using a sliding filter:



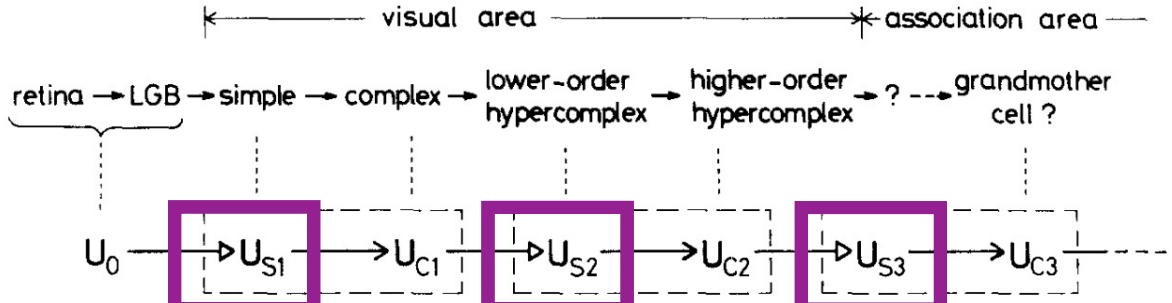


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

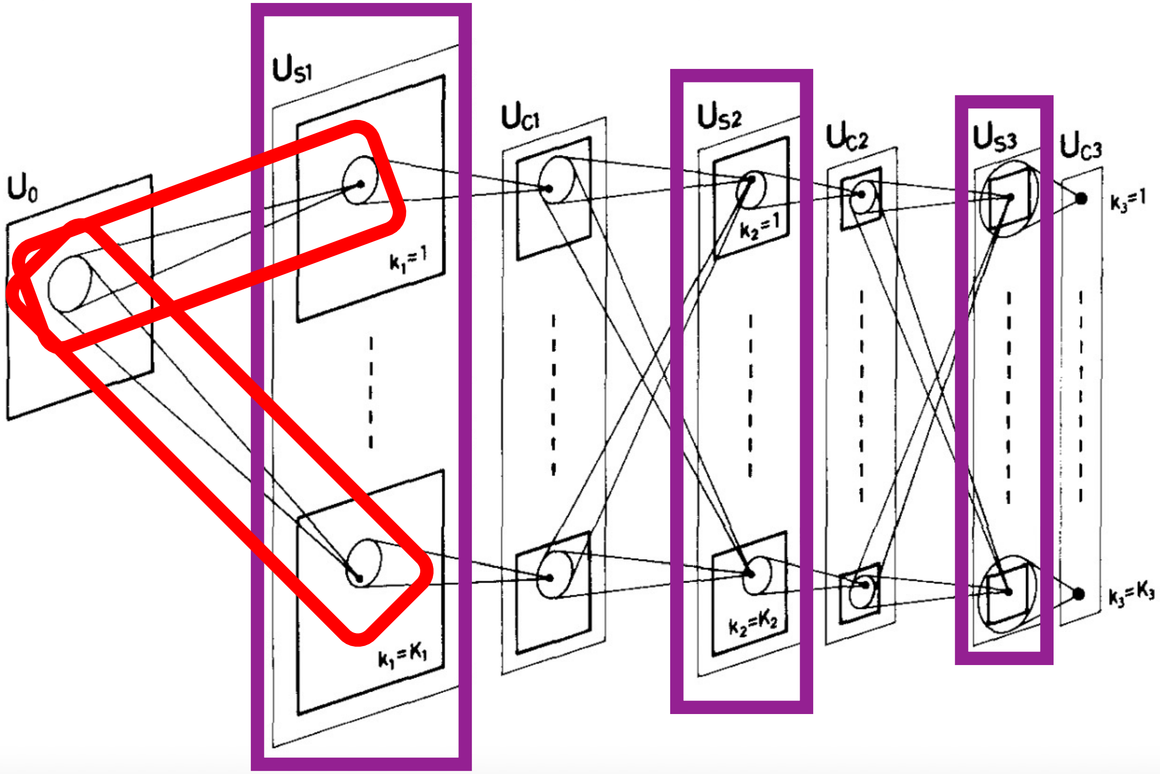


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Neocognitron: Key Ingredients

Complex cells fire when any part of the local region is the desired pattern

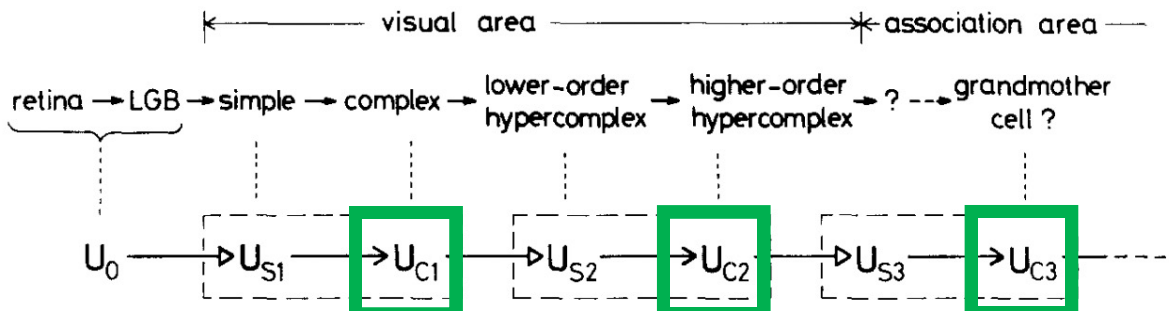


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

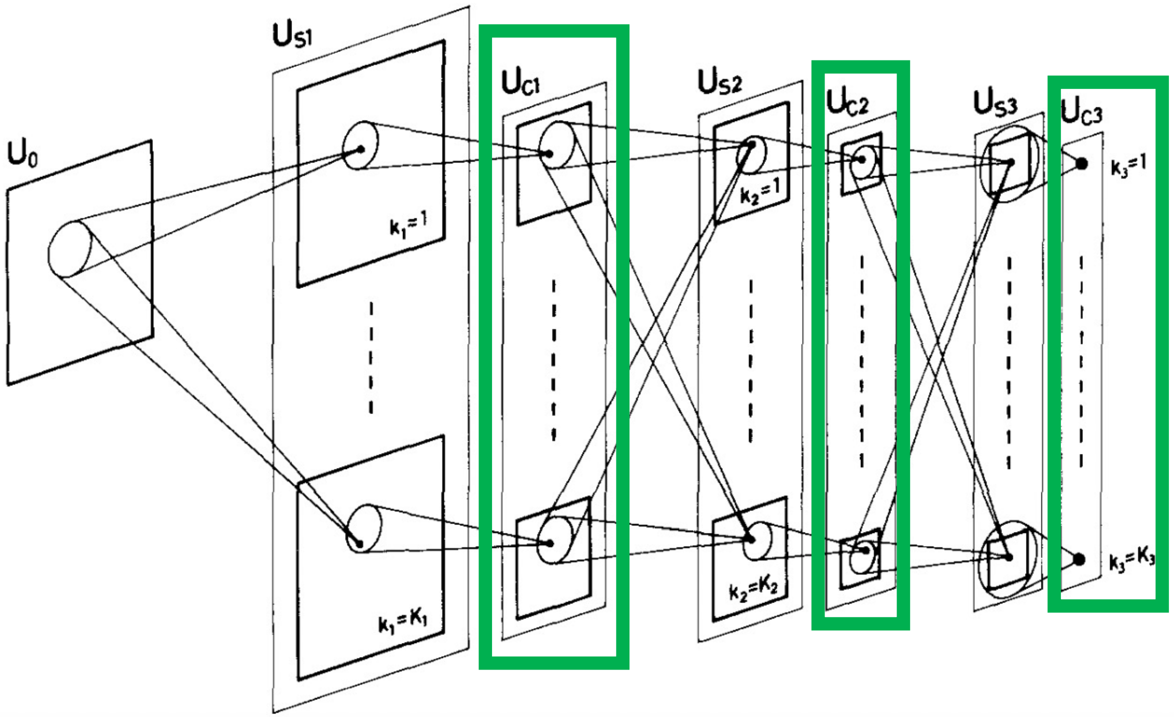
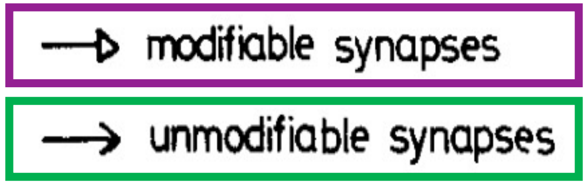


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Neocognitron: Key Ingredients

1. Convolutional layers



2. Pooling Layers

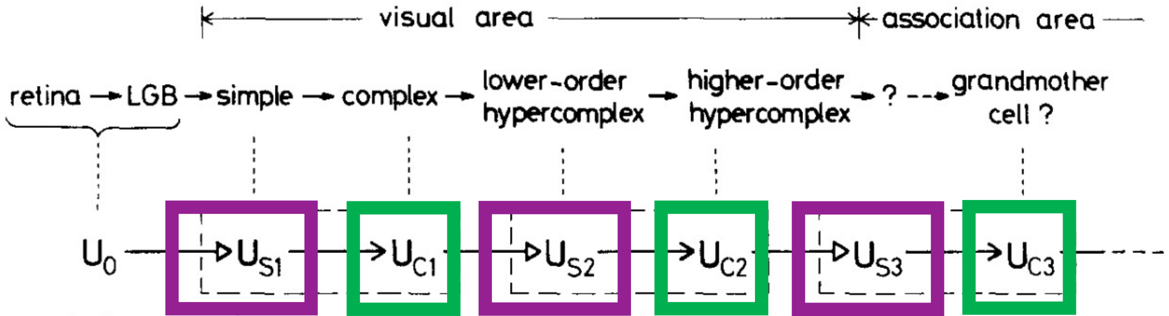
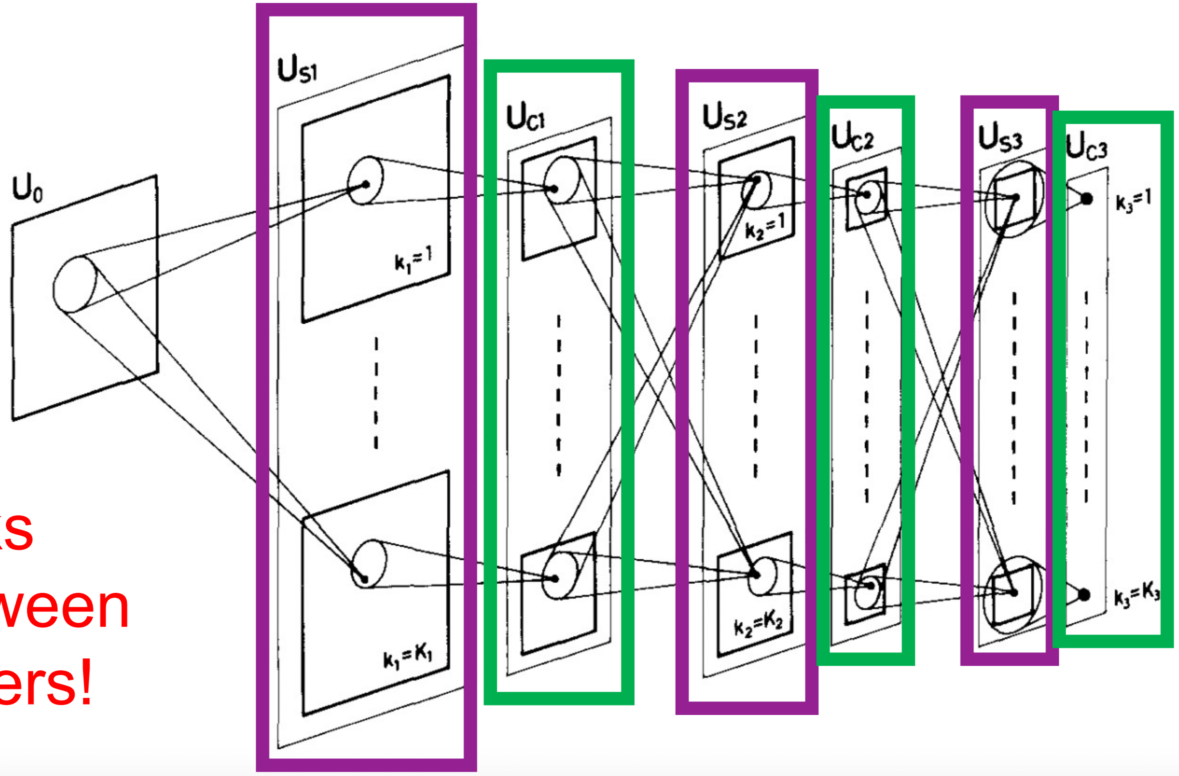


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron



Note: modern networks similarly alternate between these two types of layers!

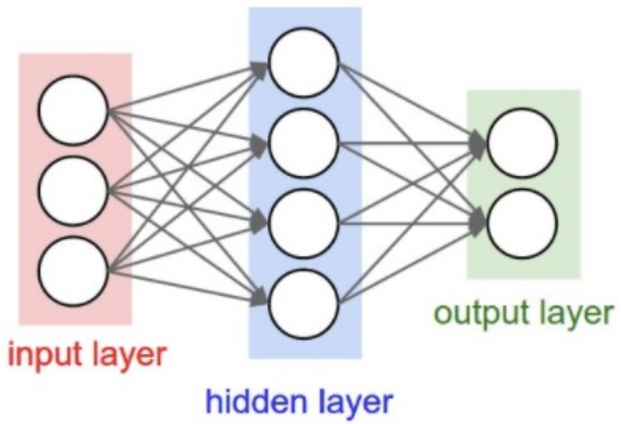
Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Today’s Topics

- Neural Networks for Spatial Data
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Programming Tutorial

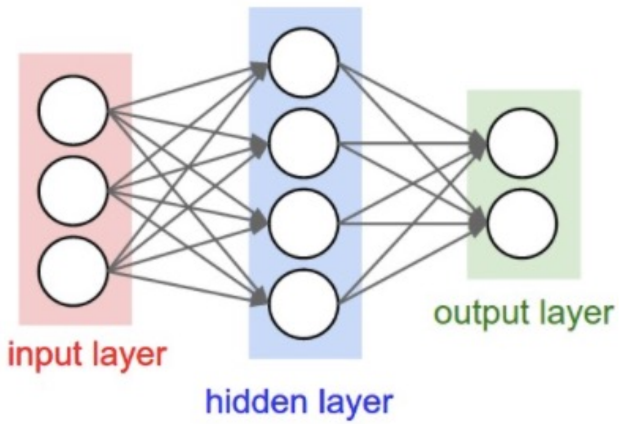
Motivation: Fully-Connected Layers Are Limited



Each node provides input to each node in the next layer

- Assume 3-layer model with 100 nodes, 100 nodes, and then 2 nodes
  - e.g., how many weights are in a 640x480 grayscale image?
    - 640x480x100 + 100x100 + 100x2 = 30,730,200
  - e.g., how many weights are in a 3.1 Megapixel grayscale image (2048X1536)?
    - 2048x1536x100 + 100x100 + 100x2 = 314,583,000

Motivation: Fully-Connected Layers Are Limited



Issue: many model parameters in fully connected networks

- Assume 3-layer model with 100 nodes, 100 nodes, and then 2 nodes
  - e.g., how many weights are in a 640x480 grayscale image?
    - 640x480x100 + 100x100 + 100x2 = 30,730,200
  - e.g., how many weights are in a 3.1 Megapixel grayscale image (2048X1536)?
    - 2048x1536x100 + 100x100 + 100x2 = 314,583,000

Motivation: Fully-Connected Layers Are Limited

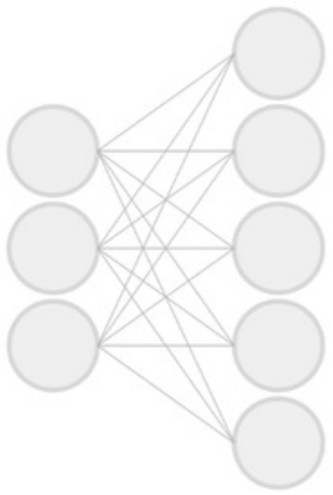
Many model parameters…

- increases chance of overfitting
- requires more training data
- increases memory/storage requirements

- Assume 3-layer model with 100 nodes, 100 nodes, and then 2 nodes
  - e.g., how many weights are in a 640x480 grayscale image?
    - 640x480x100 + 100x100 + 100x2 = 30,730,200
  - e.g., how many weights are in a 3.1 Megapixel grayscale image (2048X1536)?
    - 2048x1536x100 + 100x100 + 100x2 = 314,583,000

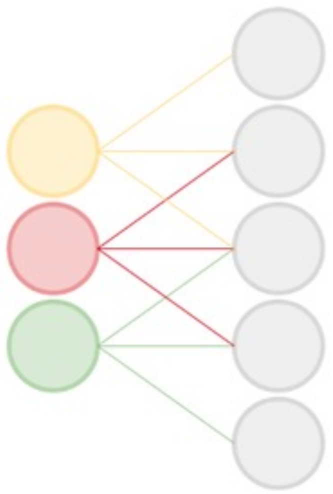
Key Ingredient 1: Convolutional Layers

Fully-connected:



Rather than have each node provide input to each node in the next layer…

Convolutional:

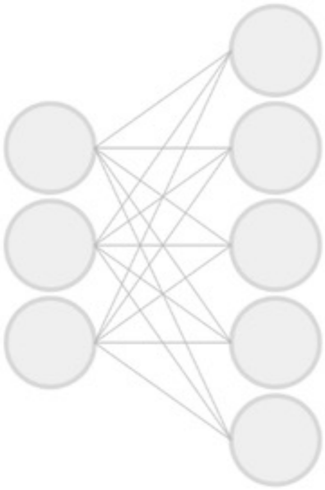


each node receives input only from a small neighborhood in previous layer (and there is parameter sharing)

Figure Source: https://qph.fs.quoracdn.net/main-qimg-2e1f0071ca9878f7719ed0ea8aeb386d

Fully-Connected vs Convolutional Layers

Fully-connected:



Convolutional layers dramatically reduce number of model parameters!

Convolutional:

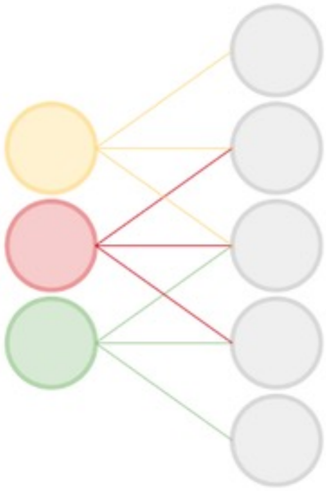
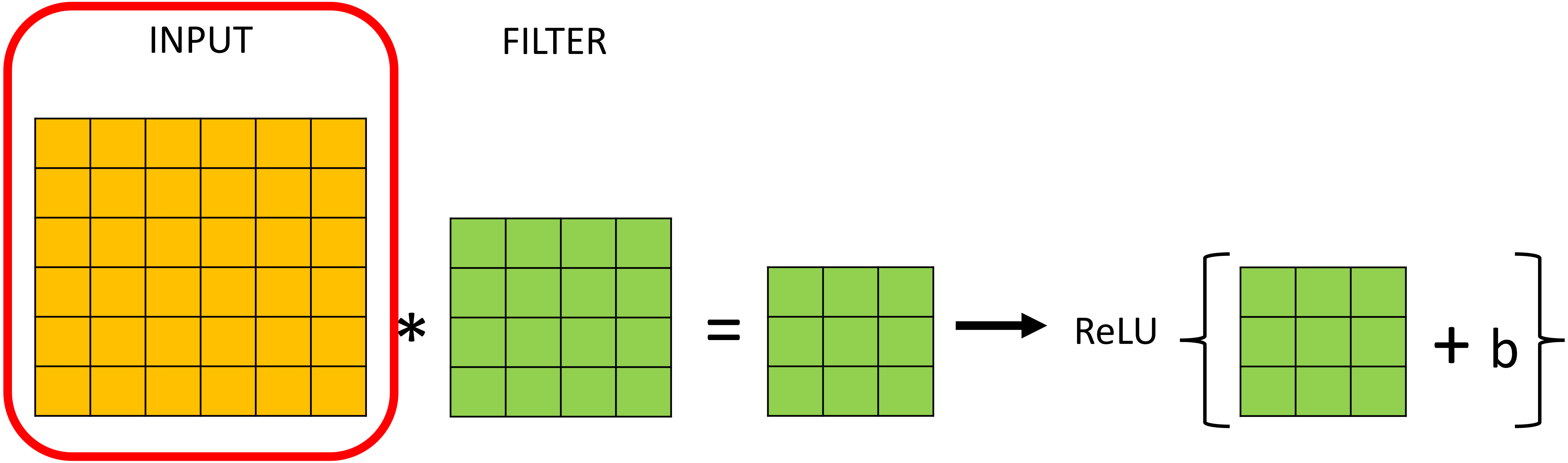


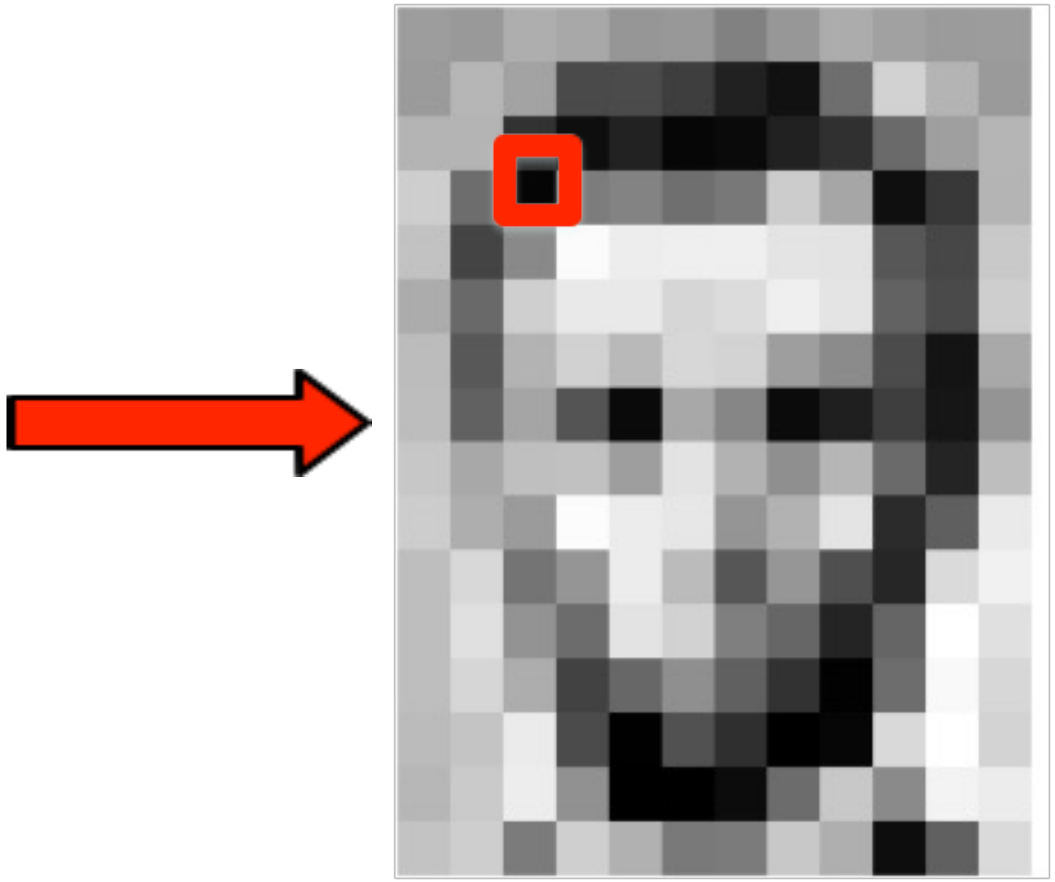
Figure Source: https://qph.fs.quoracdn.net/main-qimg-2e1f0071ca9878f7719ed0ea8aeb386d

Key Ingredient 1: Convolutional Layers



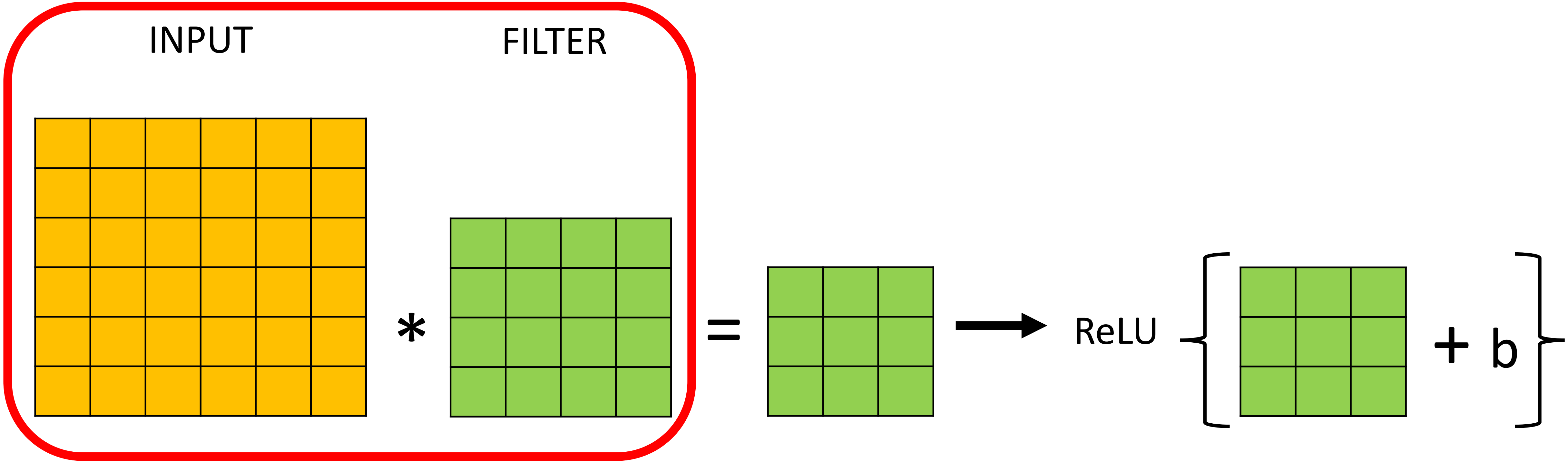
Recall: Image Representation (8-bit Grayscale)

157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	[illegible]	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	105	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	75	20	169
189	97	165	84	10	168	134	11	31	62	22	148
199	168	191	193	158	227	178	143	182	106	36	190
205	174	155	252	236	231	149	178	228	43	95	234
190	216	116	149	236	187	86	150	79	38	218	241
190	224	147	108	227	210	127	102	36	101	255	224
190	214	173	66	103	143	95	50	2	109	249	215
187	196	235	75	1	81	47	0	6	217	255	211
183	202	237	145	0	0	12	108	200	138	243	236
195	206	123	207	177	121	123	200	175	13	96	218

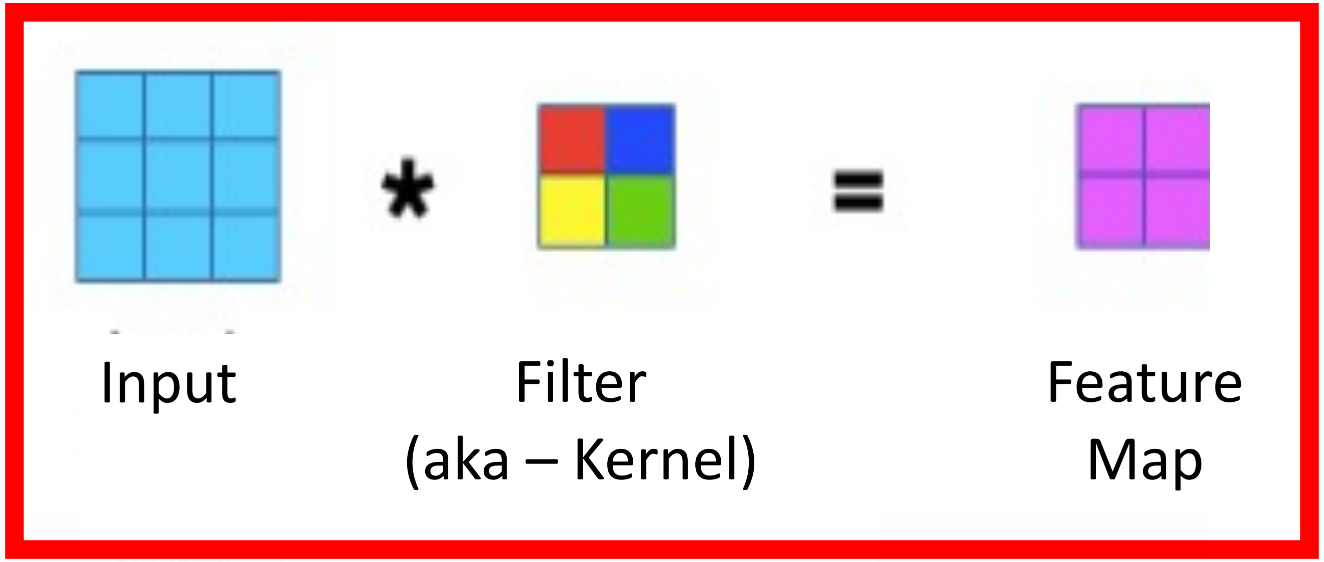


https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

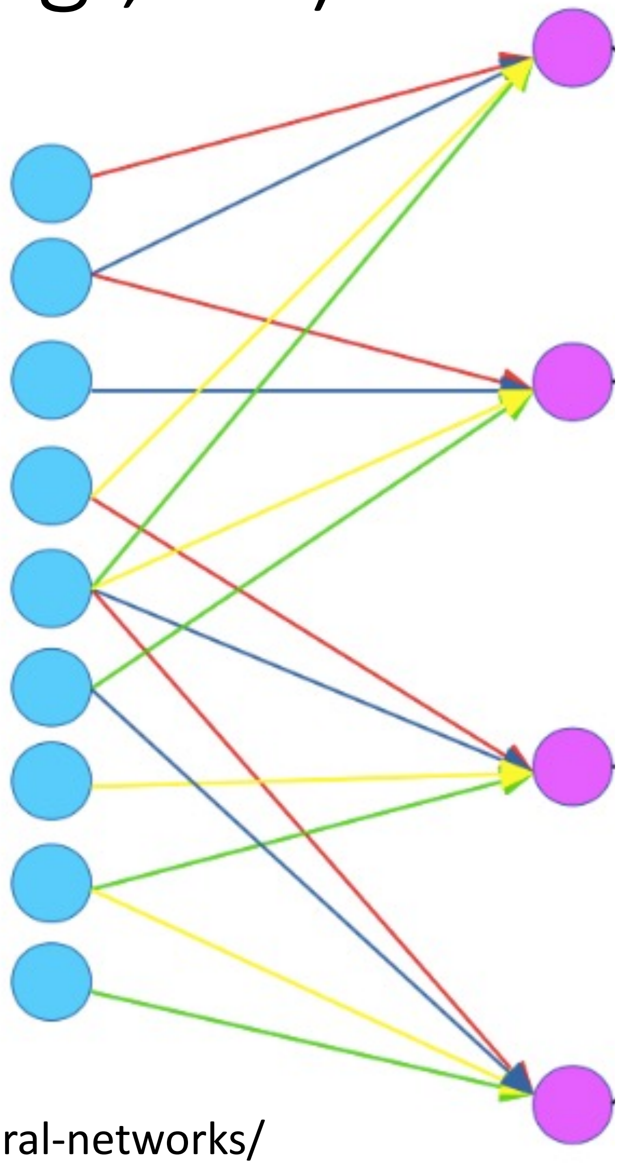
Key Ingredient 1: Convolutional Layers



Convolution: Applies Linear Filter (e.g., 2D)



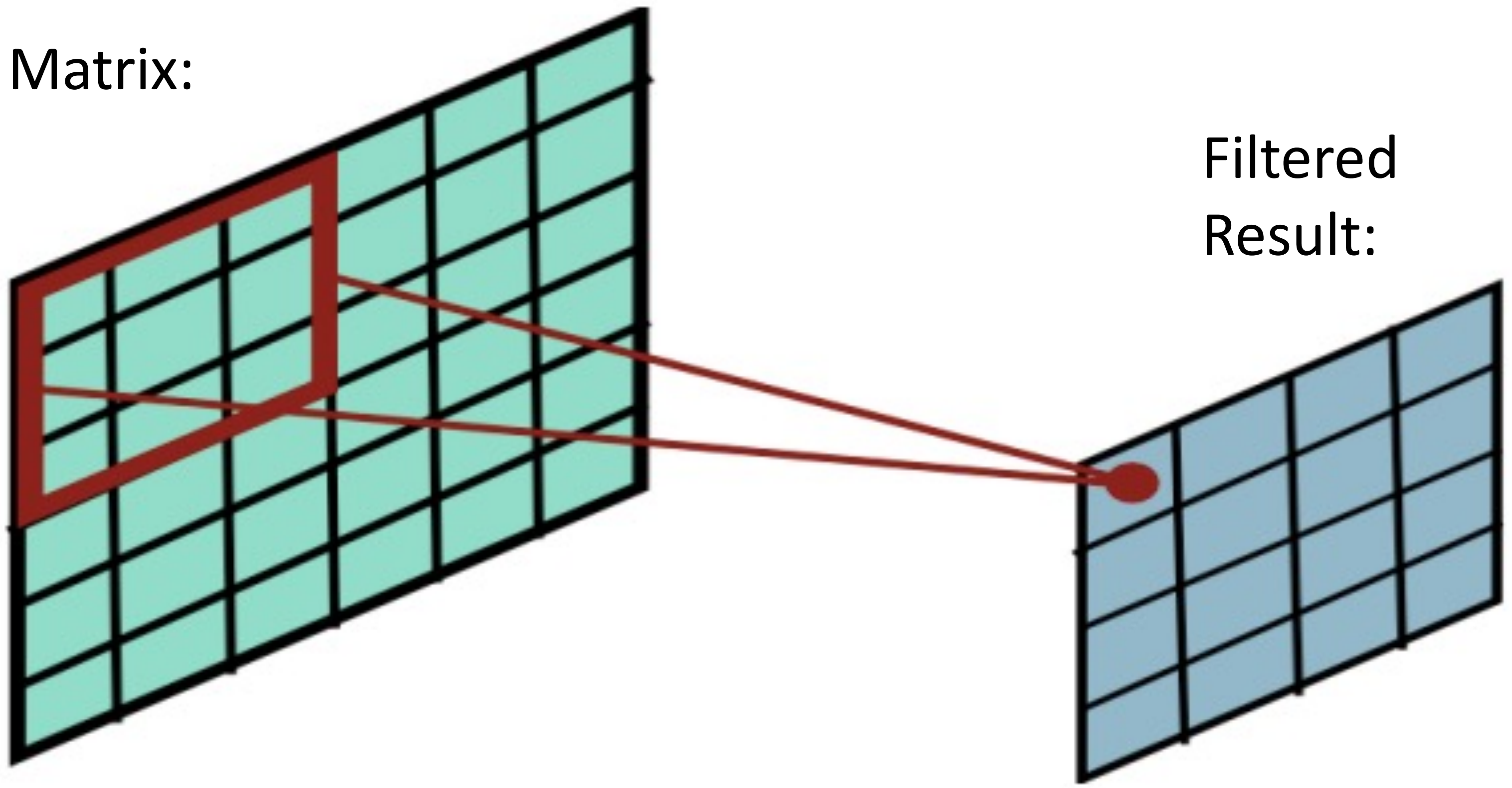
Way to Interpret
Neural Network



- Compute a function of local neighborhood for each location in matrix
- A filter specifies the function for how to combine neighbors' values

https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

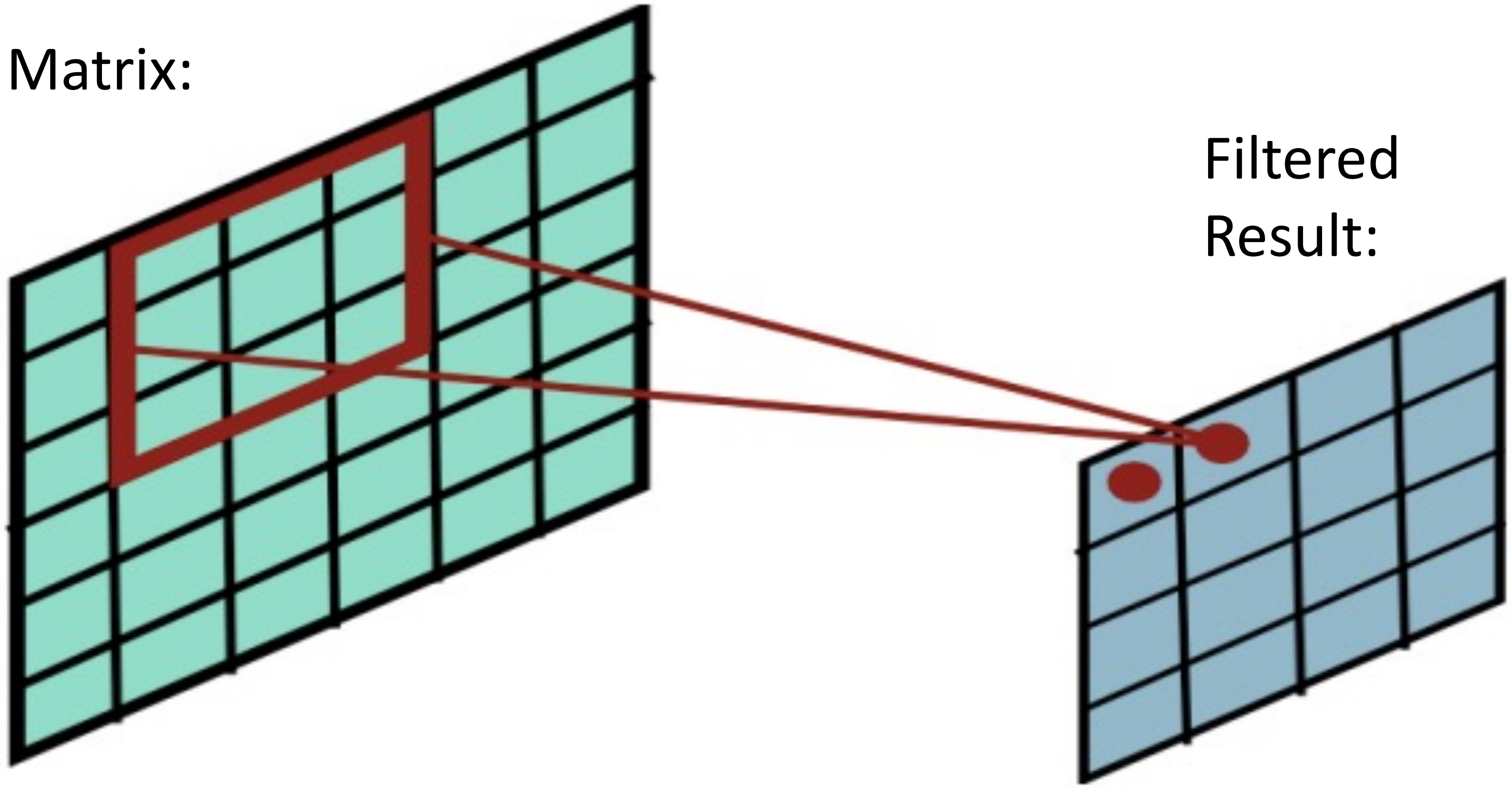
2D Filtering



Slides filter over the matrix and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

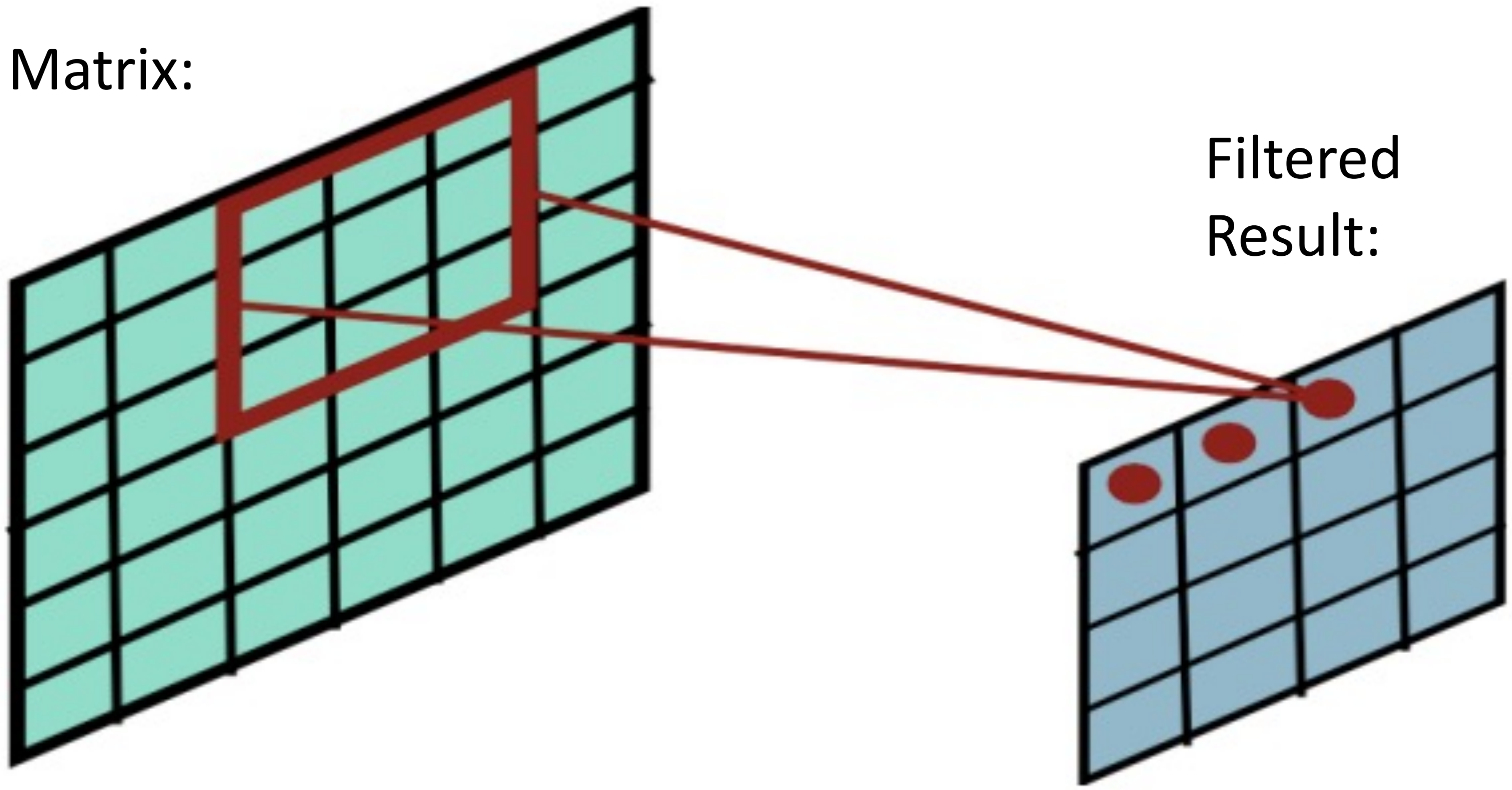
2D Filtering



Slides filter over the matrix and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

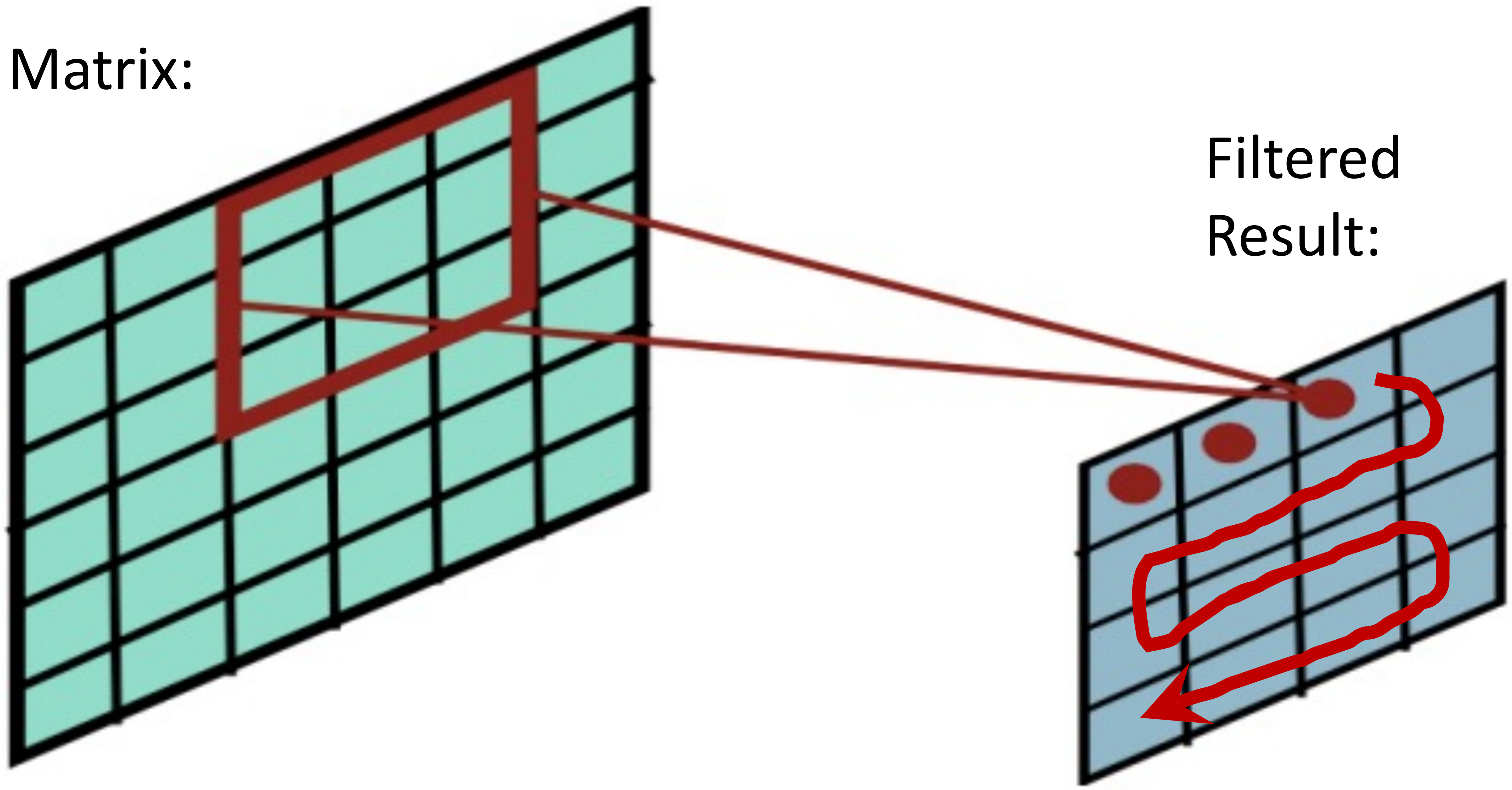
2D Filtering



Slides filter over the matrix and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

2D Filtering



Slides filter over the matrix and computes dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

?	?	?
?	?	?
?	?	?

Dot Product = 1*1 + 1*0 + 1*1 + 0*0 + 1*1 + 1*0 + 0*1 + 0*1 + 0*0 + 0*0 + 1*1

Dot Product = 4

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	?	?
?	?	?
?	?	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	3	?
?	?	?
?	?	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	3	4
?	?	?
?	?	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	3	4
2	?	?
?	?	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	3	4
2	4	?
?	?	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	3	4
2	4	3
?	?	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	3	4
2	4	3
2	?	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	3	4
2	4	3
2	3	?

2D Filtering: Toy Example

Input

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

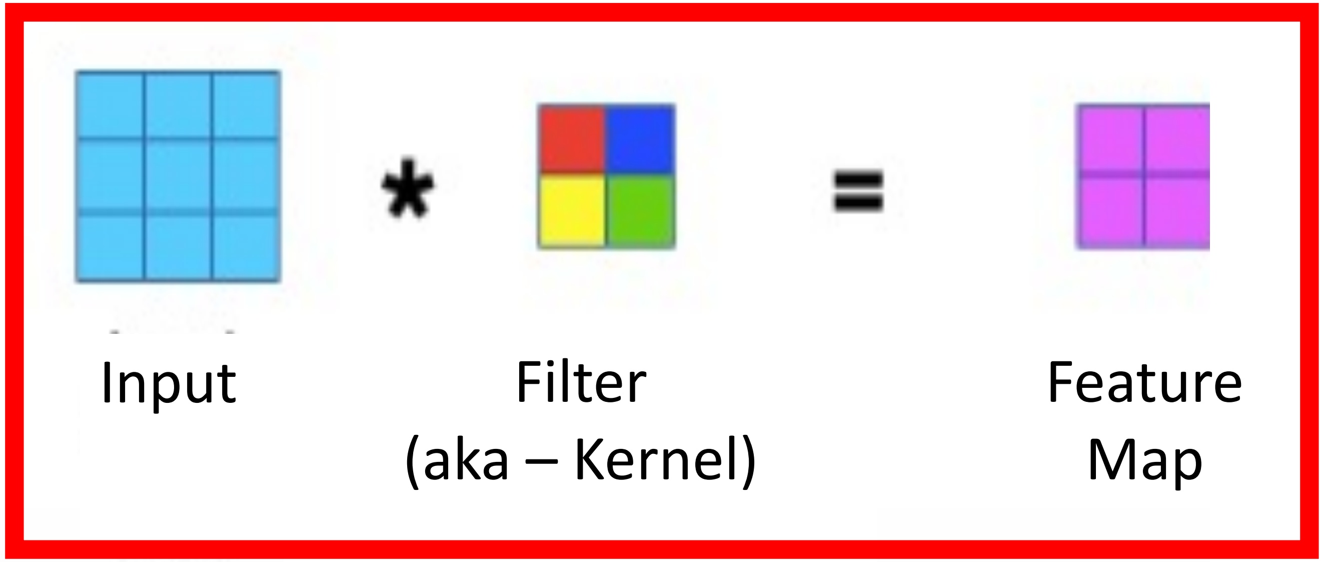
1	0	1
0	1	0
1	0	1

Feature Map

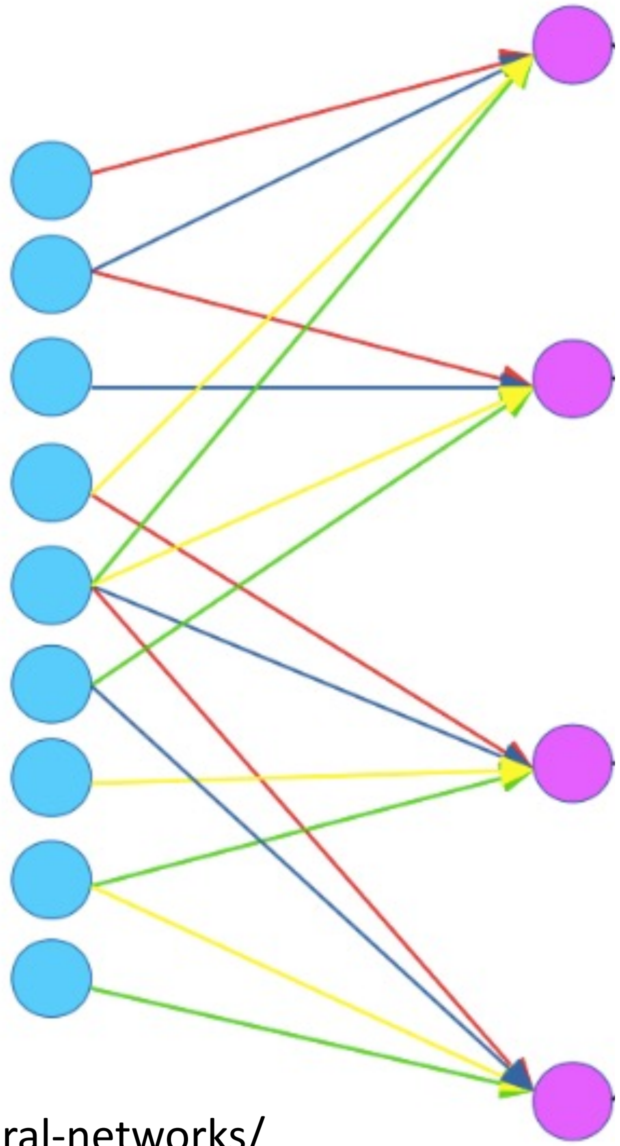
4	3	4
2	4	3
2	3	4

Convolutional Layer

- Many neural network libraries use “convolution” interchangeably with “cross correlation”; for mathematicians, these are technically different
- Examples in these slides show the “cross-correlation” function

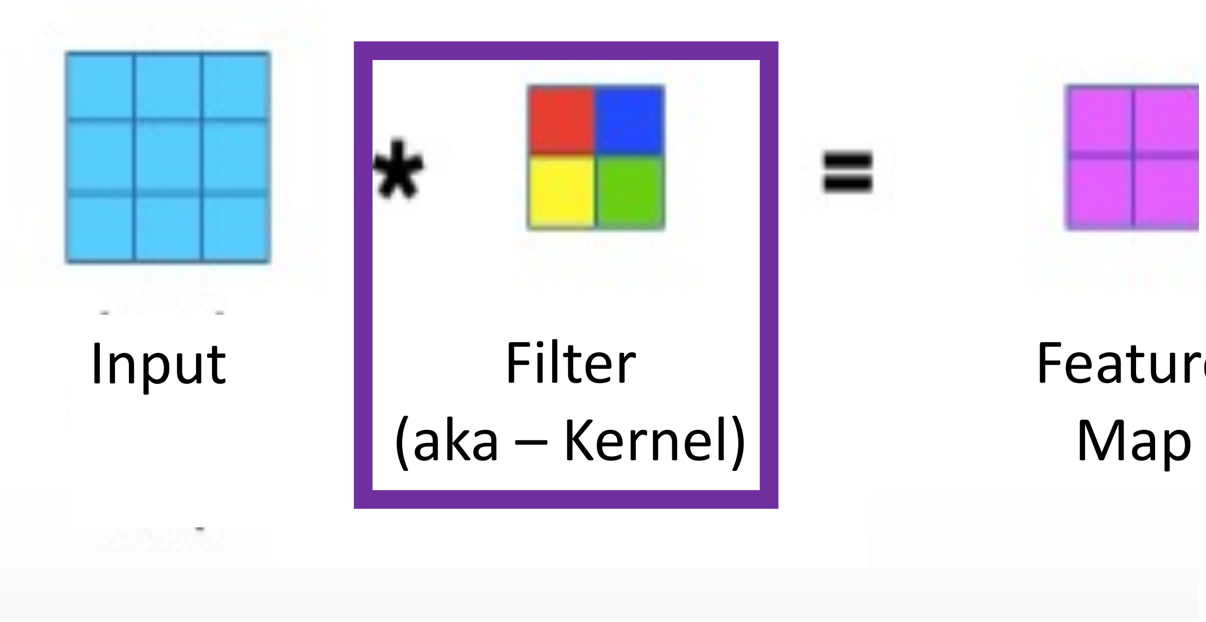


Way to Interpret
Neural Network

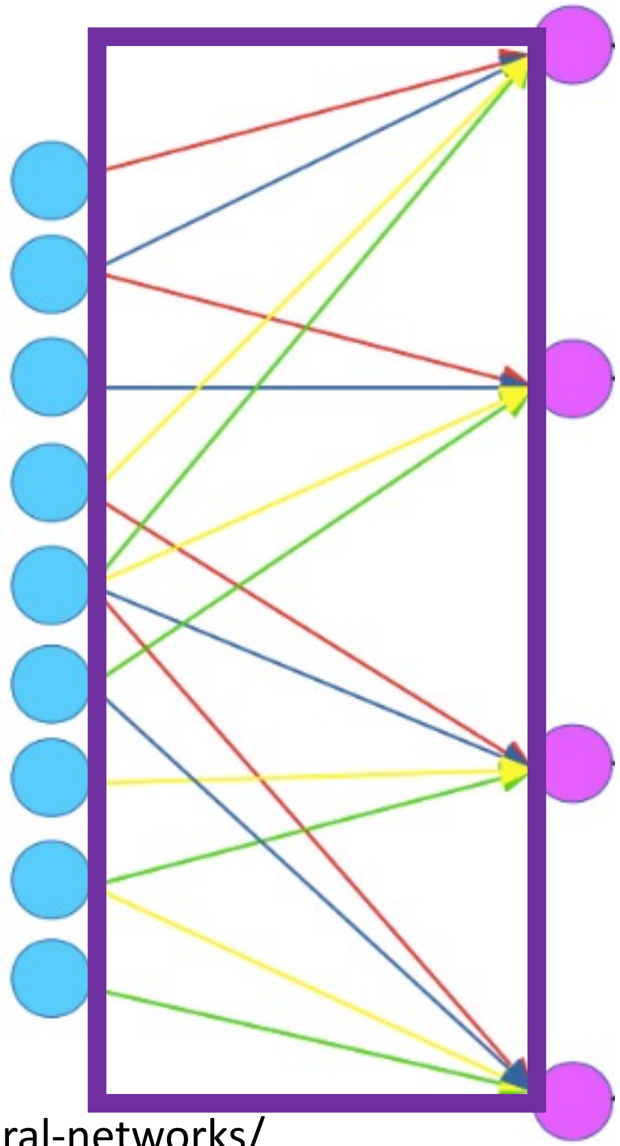


https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Convolutional Layer: Parameters to Learn



Way to Interpret
Neural Network

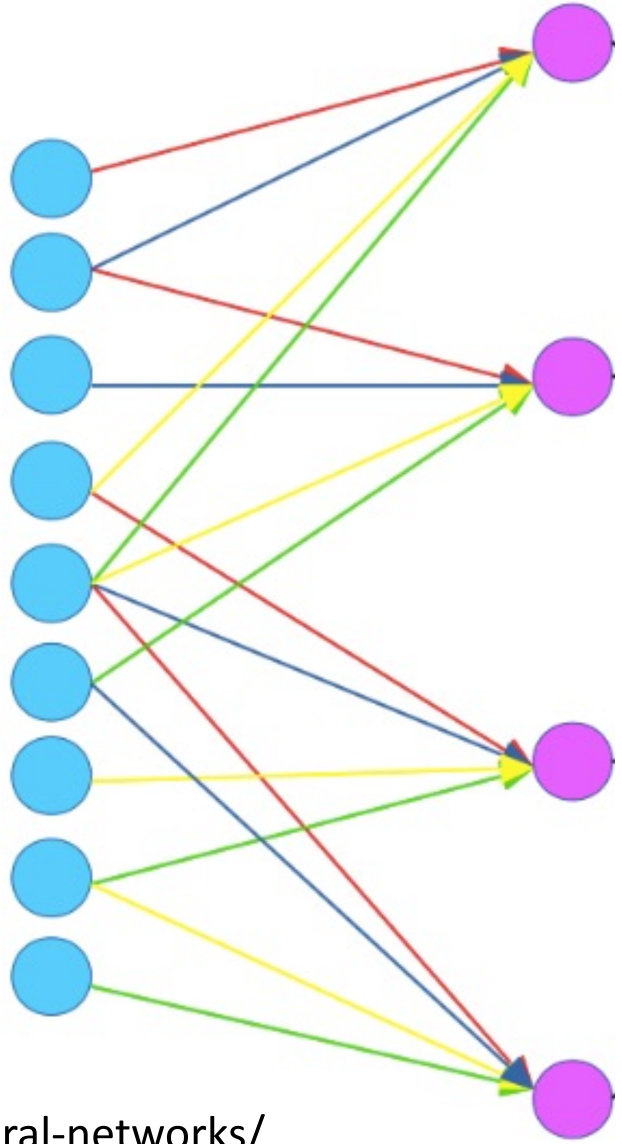


https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Convolutional Layer: Parameters to Learn

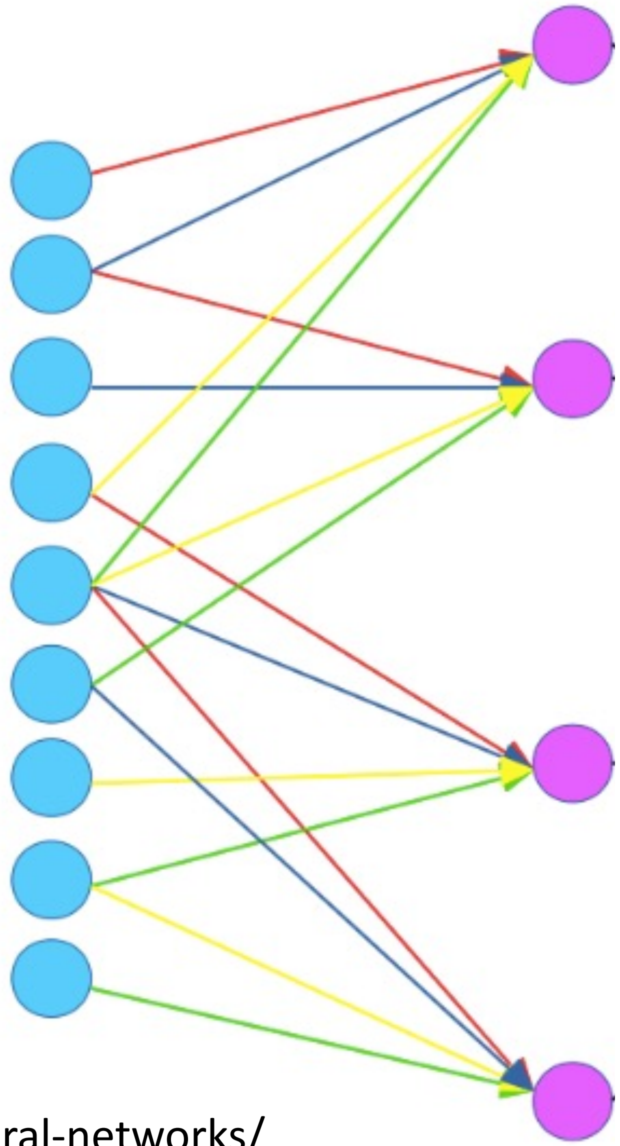
- For shown example, how many weights must be learned?
  - 4 (red, blue, yellow, and green values)

- If we instead used a fully connected layer, how many weights would need to be learned?
  - 36 (9 turquoise nodes x 4 magenta nodes)



https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

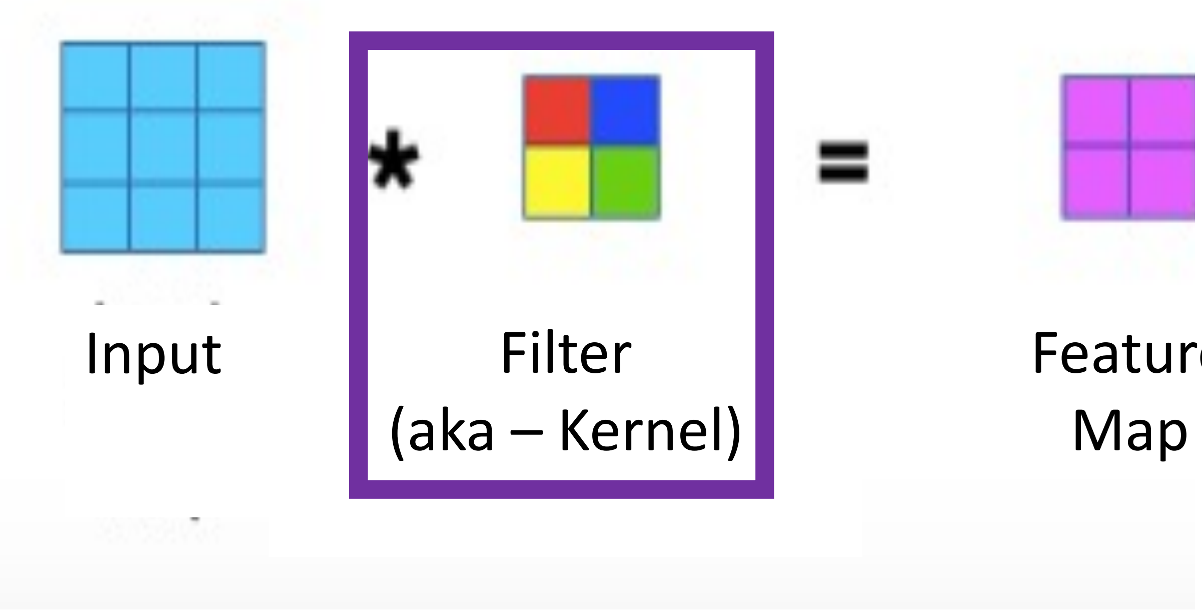
Convolutional Layer: Parameters to Learn



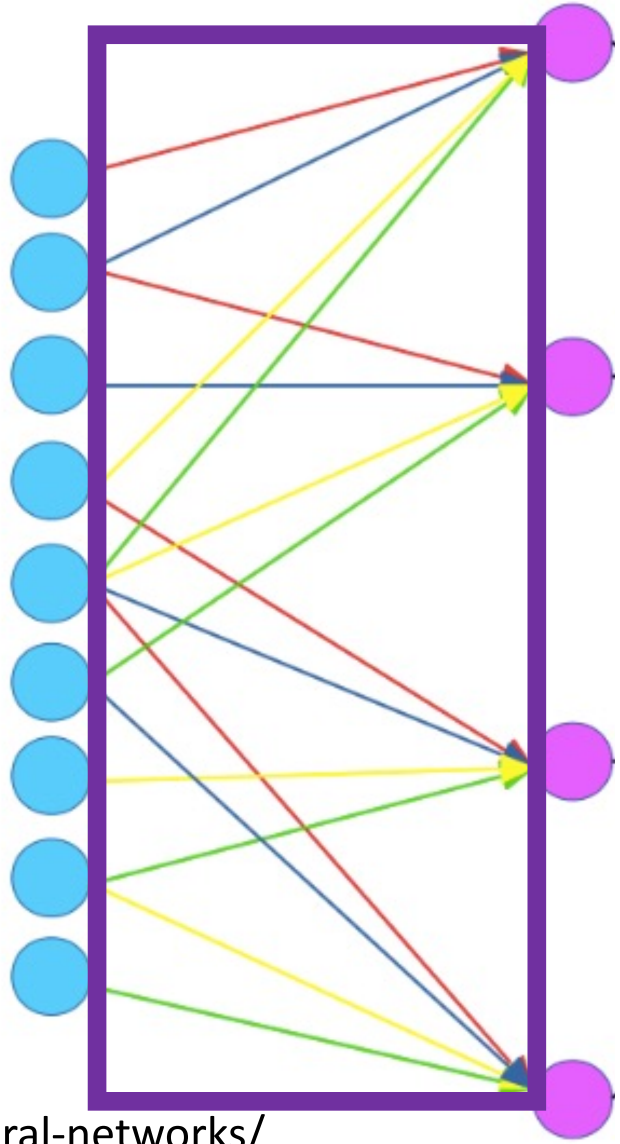
Neocognitron hard-coded filter values...
filter values are learned for CNNs

https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Convolutional Layer: What Can Filters Do?



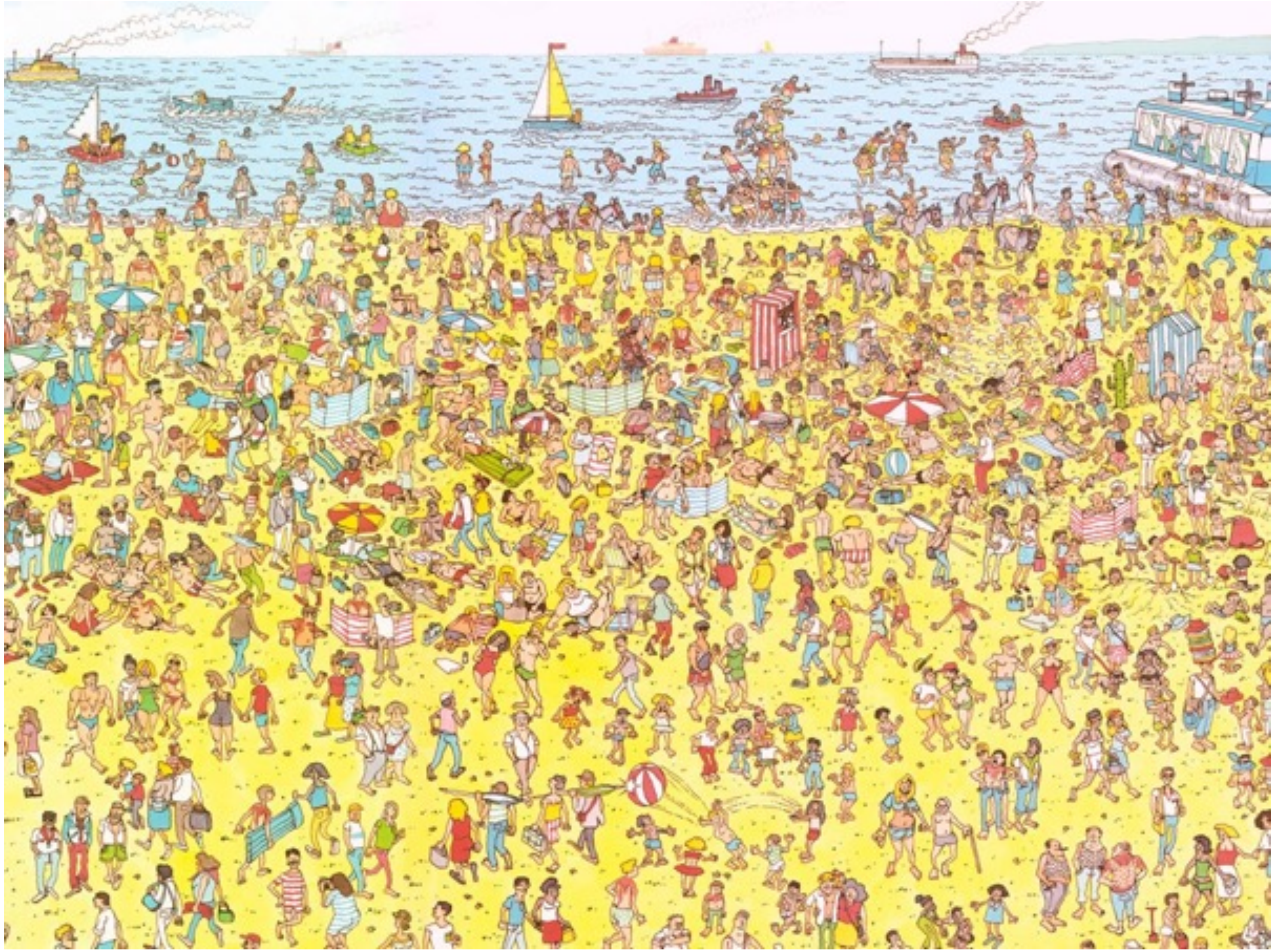
Way to Interpret
Neural Network



https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Convolutional Layer: What Can Filters Do?

Filter



Convolutional Layer: What Can Filters Do?

- e.g.,

Filter

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

Visualization of Filter

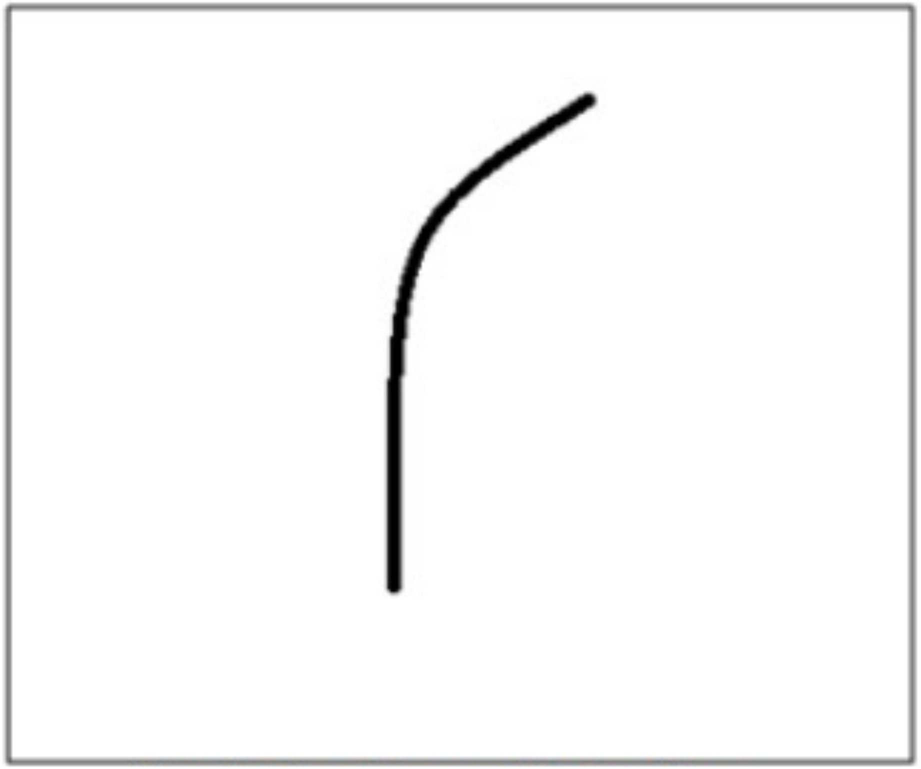
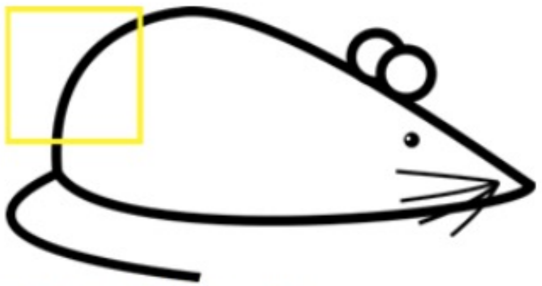


Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Convolutional Layer: What Can Filters Do?

- e.g.,

Filter Overlaid on Image



Image

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

*

Filter

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

Weighted Sum = ?

Weighted Sum = (50x30) + (20x30) + (50x30) + (50x3) + (50x30)

Weighted Sum = 6600 **(Large Number!!)**

Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Convolutional Layer: What Can Filters Do?

- e.g.,

Filter Overlaid on Image

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

$*$

Filter

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

Weighted Sum = ?

Weighted Sum = 0 **(Small Number!!)**

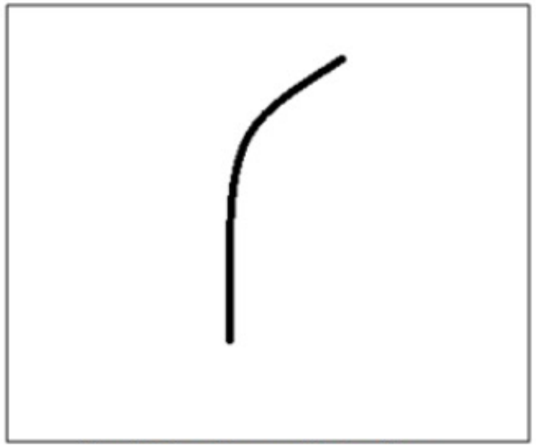
Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Convolutional Layer: What Can Filters Do?

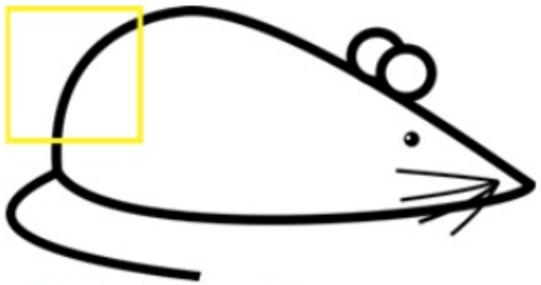
- e.g.,

This Filter is a Curve Detector!

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



Filter Overlaid on Image (Big Response!)



Filter Overlaid on Image (Small Response!)

Image Credit: https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

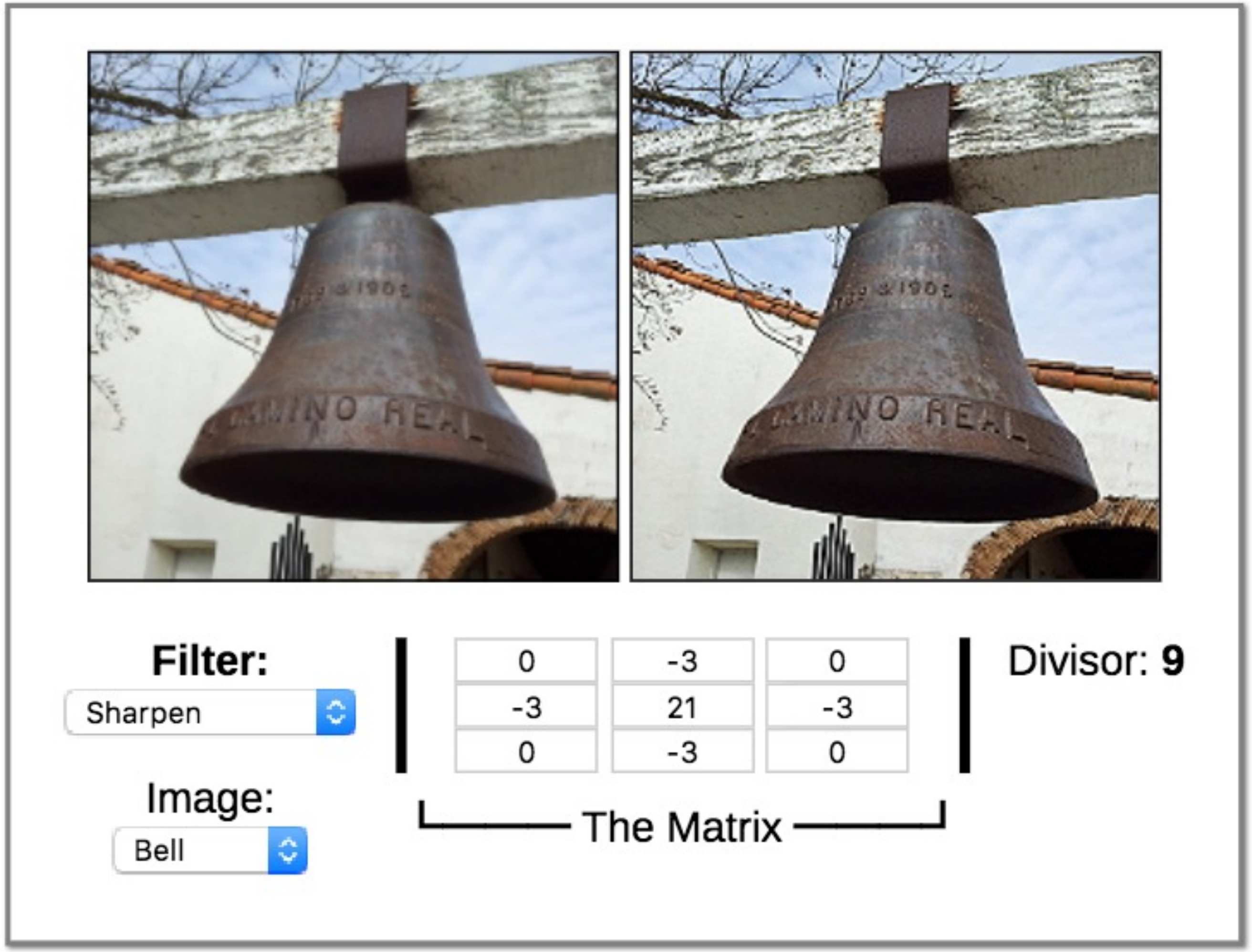
Convolutional Layer: What Can Filters Do?

	Filter	Feature Map
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

	Filter	Feature Map
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$	

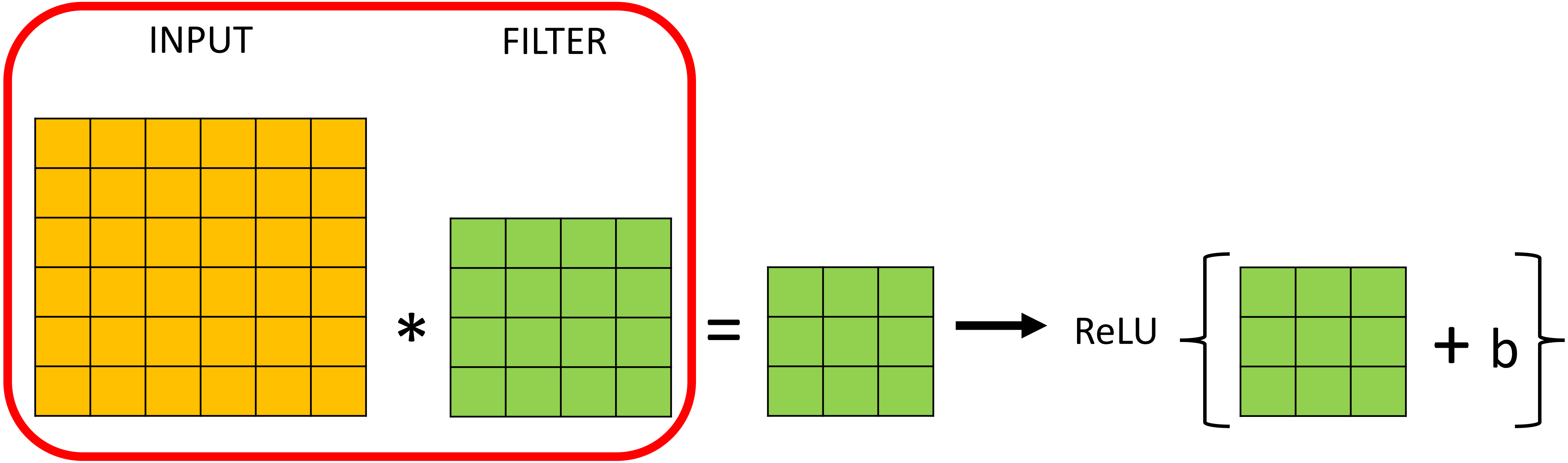
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Convolutional Layer: What Can Filters Do?



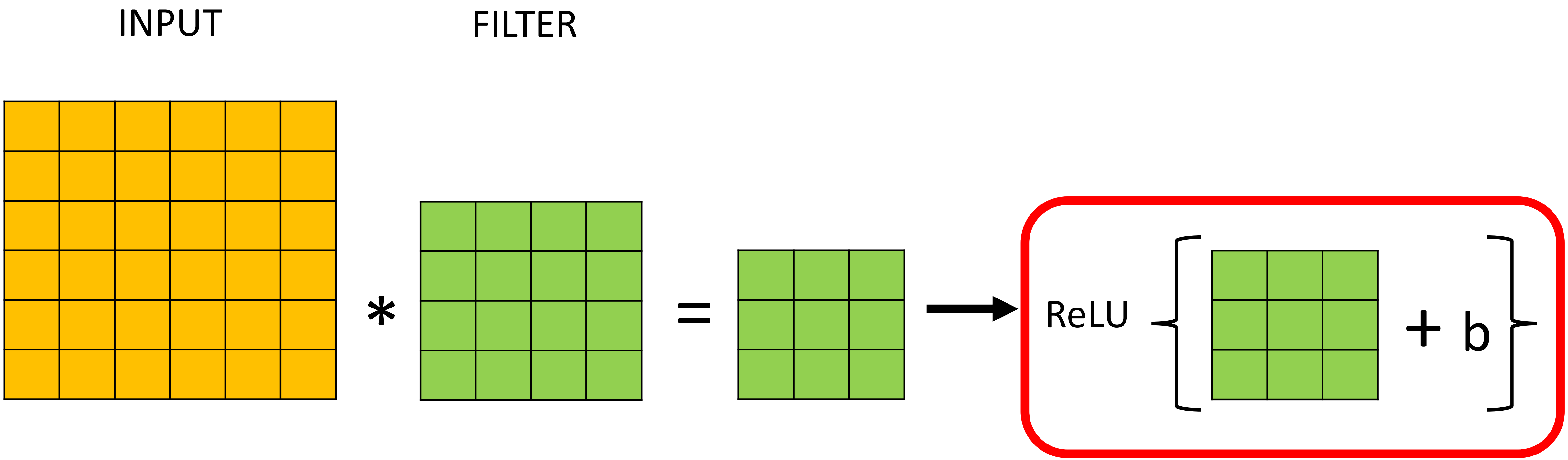
Demo: http://beej.us/blog/data/convolution-image-processing/

Key Ingredient 1: Convolutional Layers



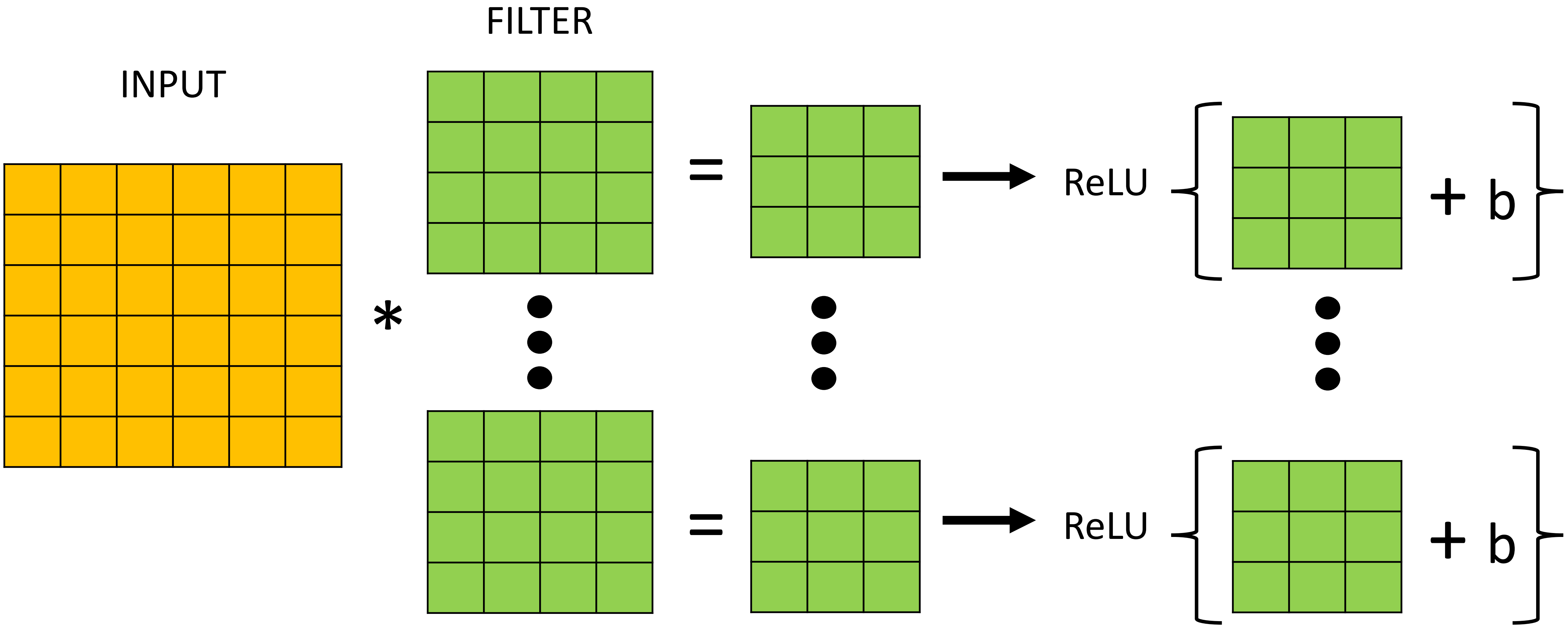
Can choose filters of any size to support feature learning!

Key Ingredient 1: Convolutional Layers



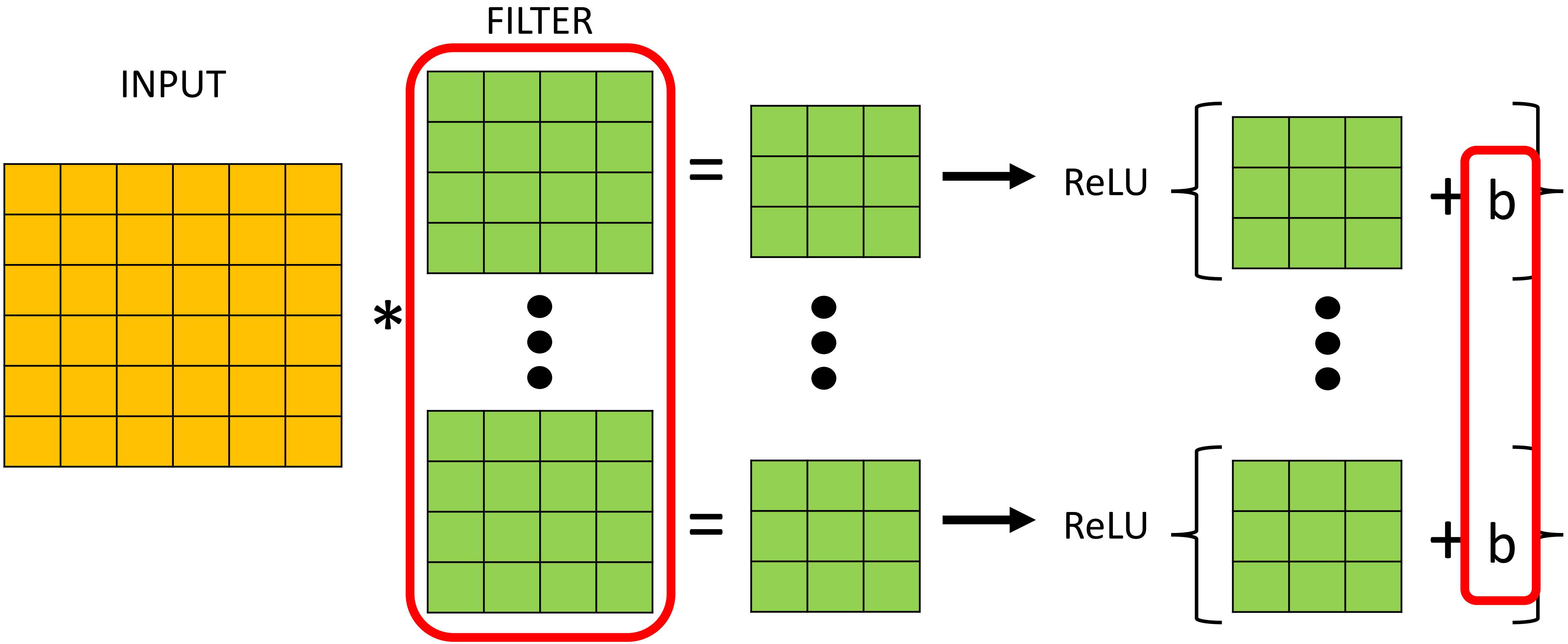
Filtered results are passed, with a bias term, through an activation function to create **activation/feature maps**

Key Ingredient 1: Convolutional Layers



Can have multiple filters (with a unique bias parameter per filter)

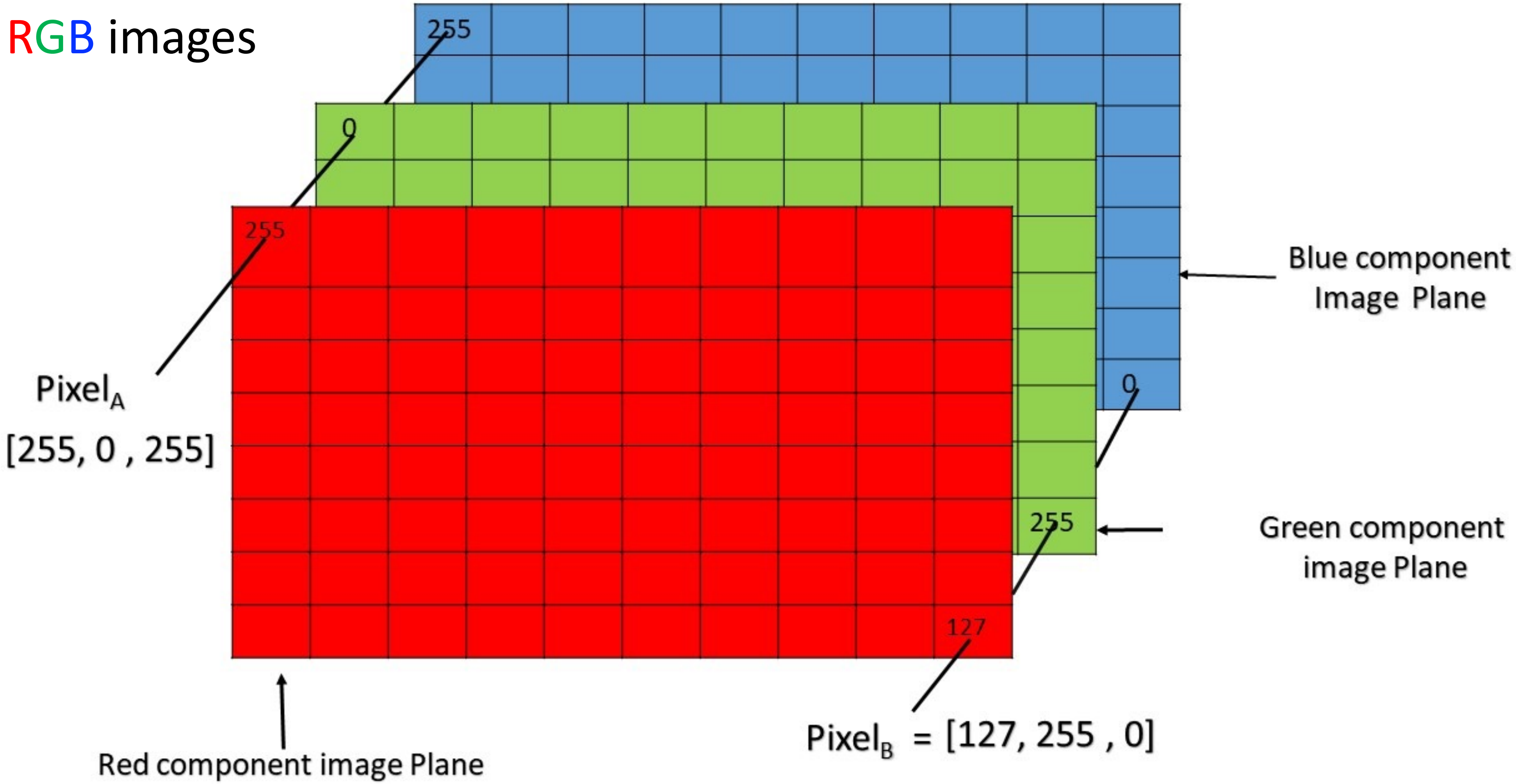
Key Ingredient 1: Convolutional Layer Summary



Neural networks learn values for all filters and biases in all layers

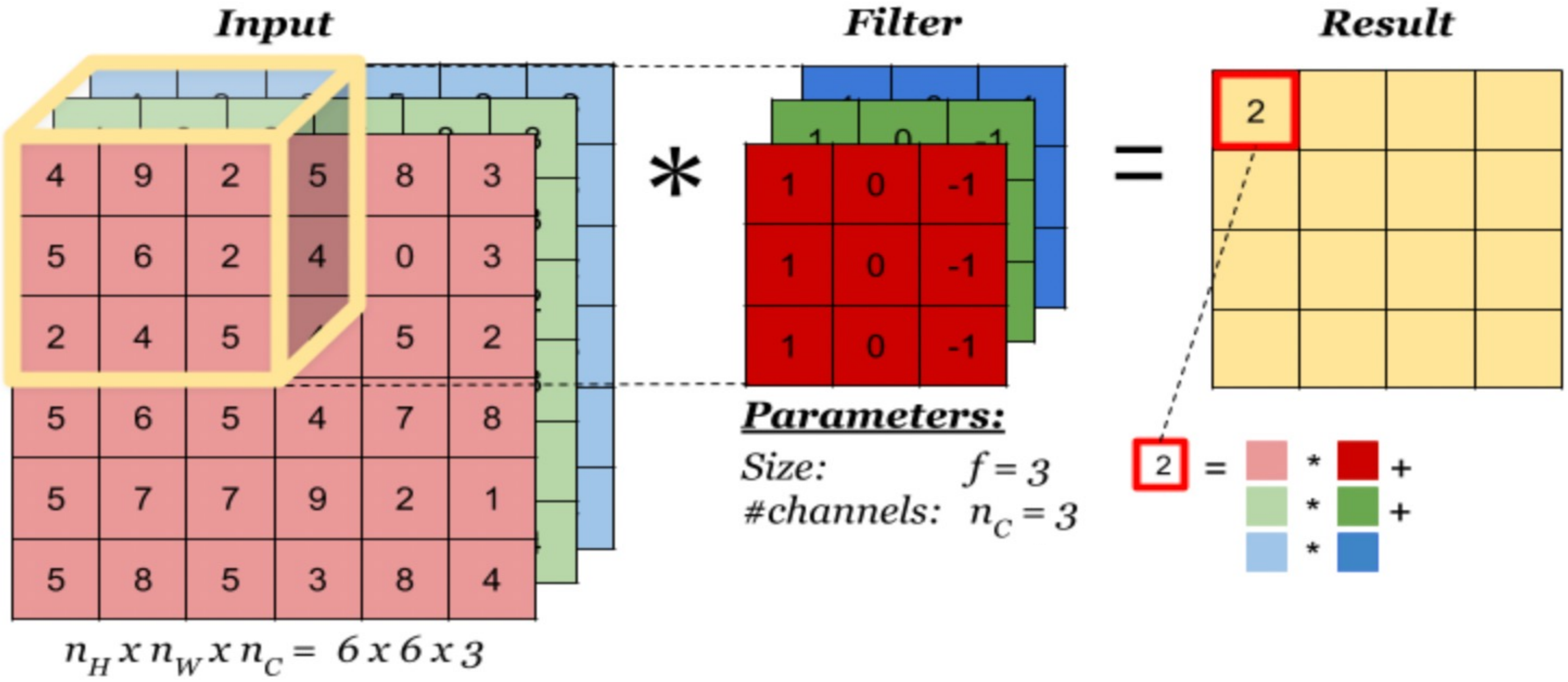
How Filters Are Applied to Multi-Channel Inputs

e.g., RGB images



https://www.geeksforgeeks.org/matlab-rgb-image-representation/

How Filters Are Applied to Multi-Channel Inputs

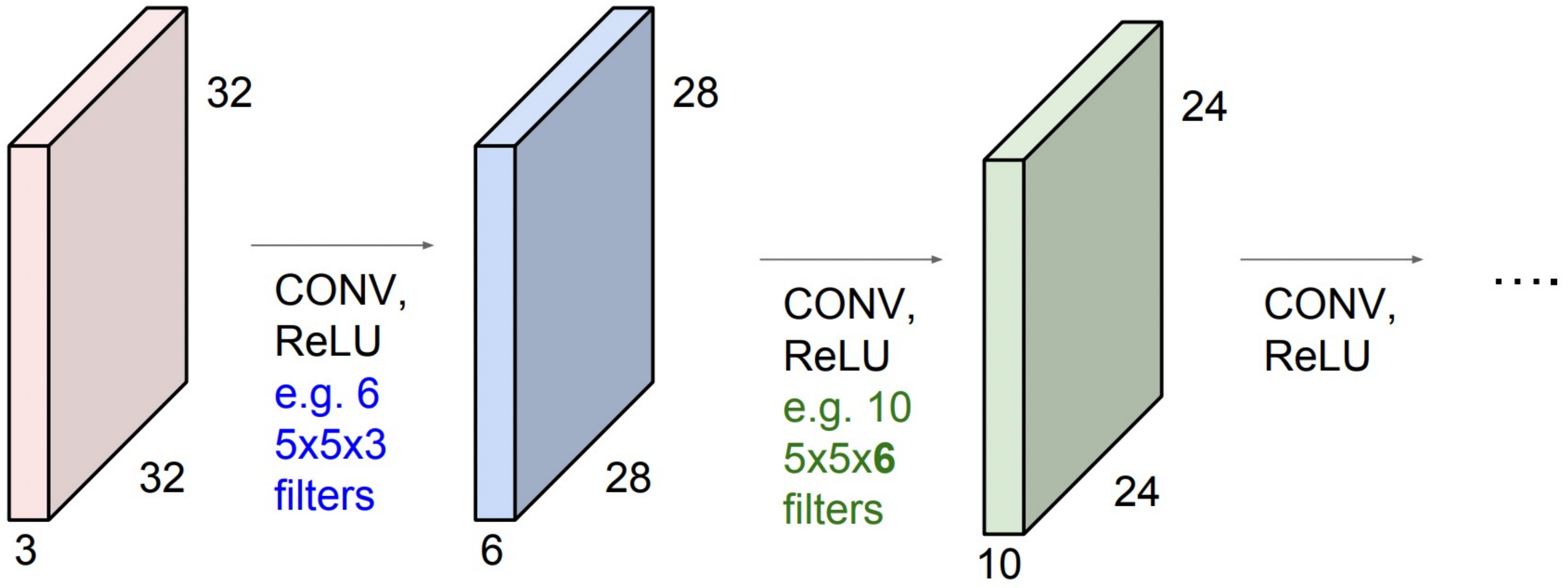


Number of channels in a filter matches that of the input

https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Convolutional Layers Stacked

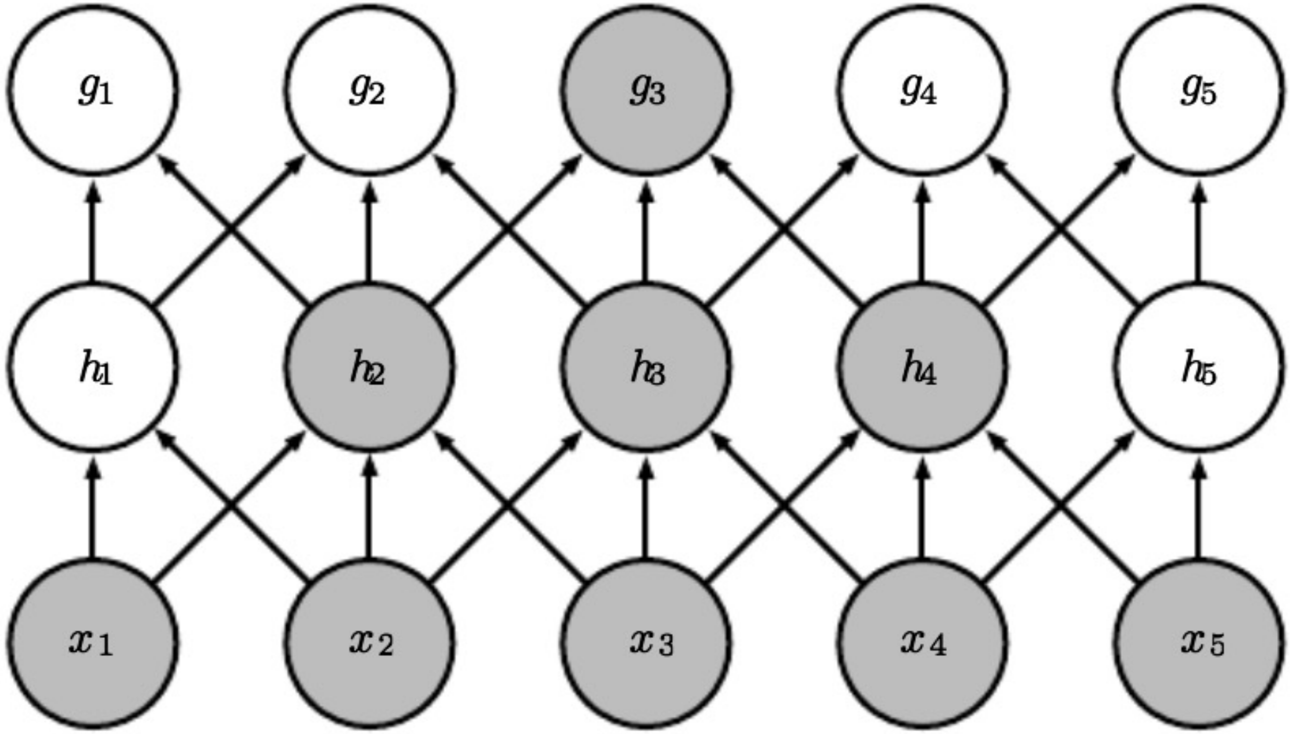
Can then stack a sequence of convolution layers; e.g.,



http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf

Convolutional Layers Stacked

Can then stack a sequence of convolution layers, which leads to identifying patterns in increasingly **larger regions of the input (e.g., pixel) space:**

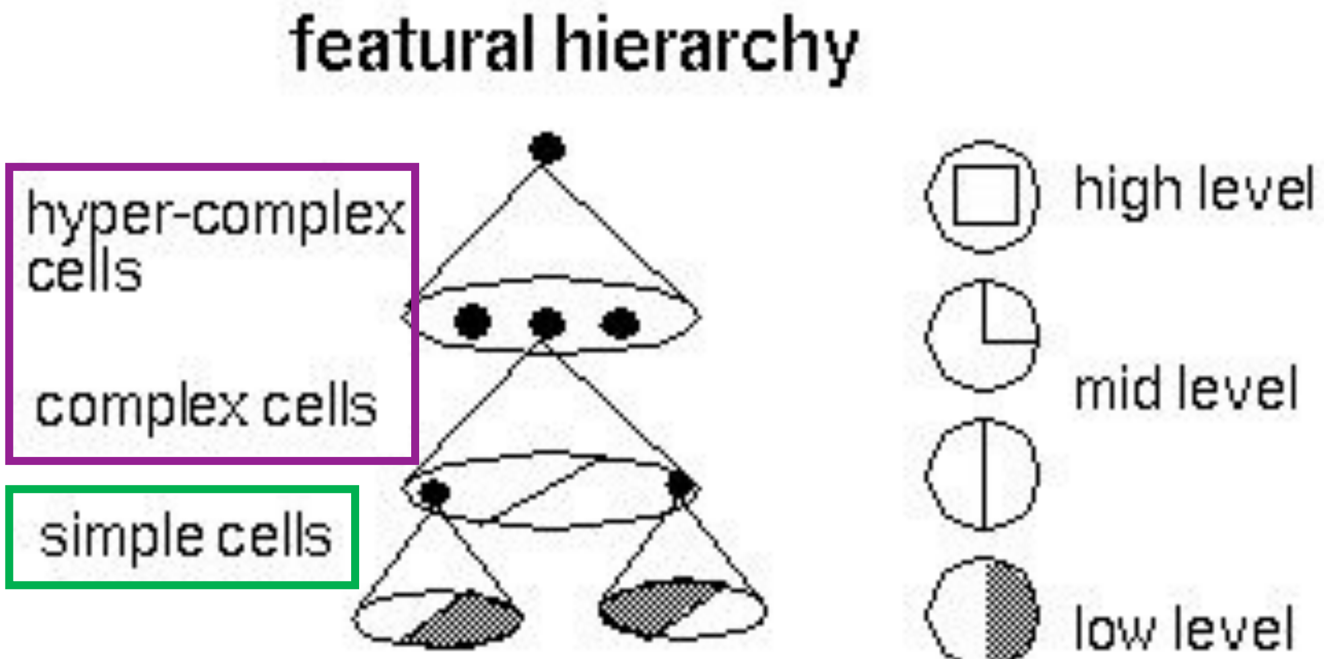


https://www.deeplearningbook.org/contents/convnets.html

Convolutional Layers Stacked

Can then stack a sequence of convolution layers, which leads to identifying patterns in increasingly **larger regions of the input (e.g., pixel) space** and **mimicking vision system**:

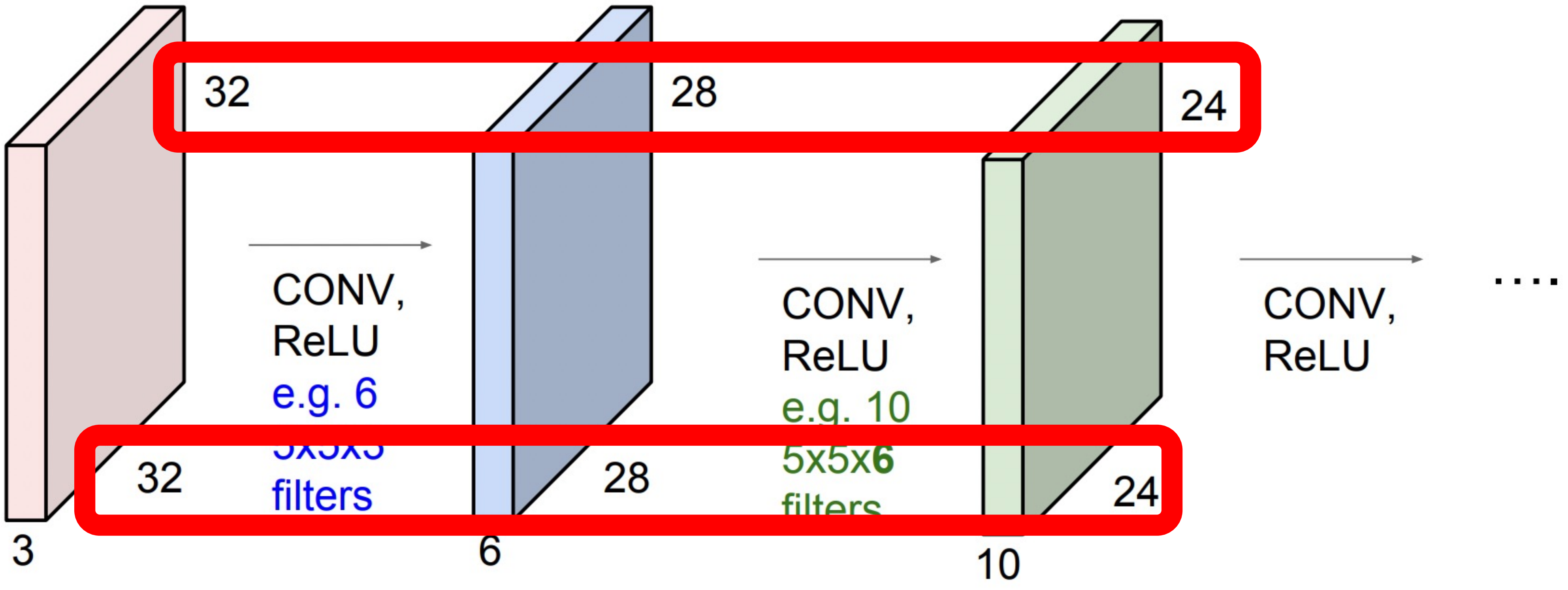
Higher level features are constructed by combining lower level features



Source: https://bruceoutdoors.files.wordpress.com/2017/08/hubel.jpg

Problem #1: Input Shrinks

Why do the dimensions shrink with each convolutional layer?



Information is lost around boundary of the input!

http://cs231n.stanford.edu/slides/2019/cs231n_2019_lecture05.pdf

Solution: Control Output Size with **Padding**

- **Padding**: add values at the boundaries

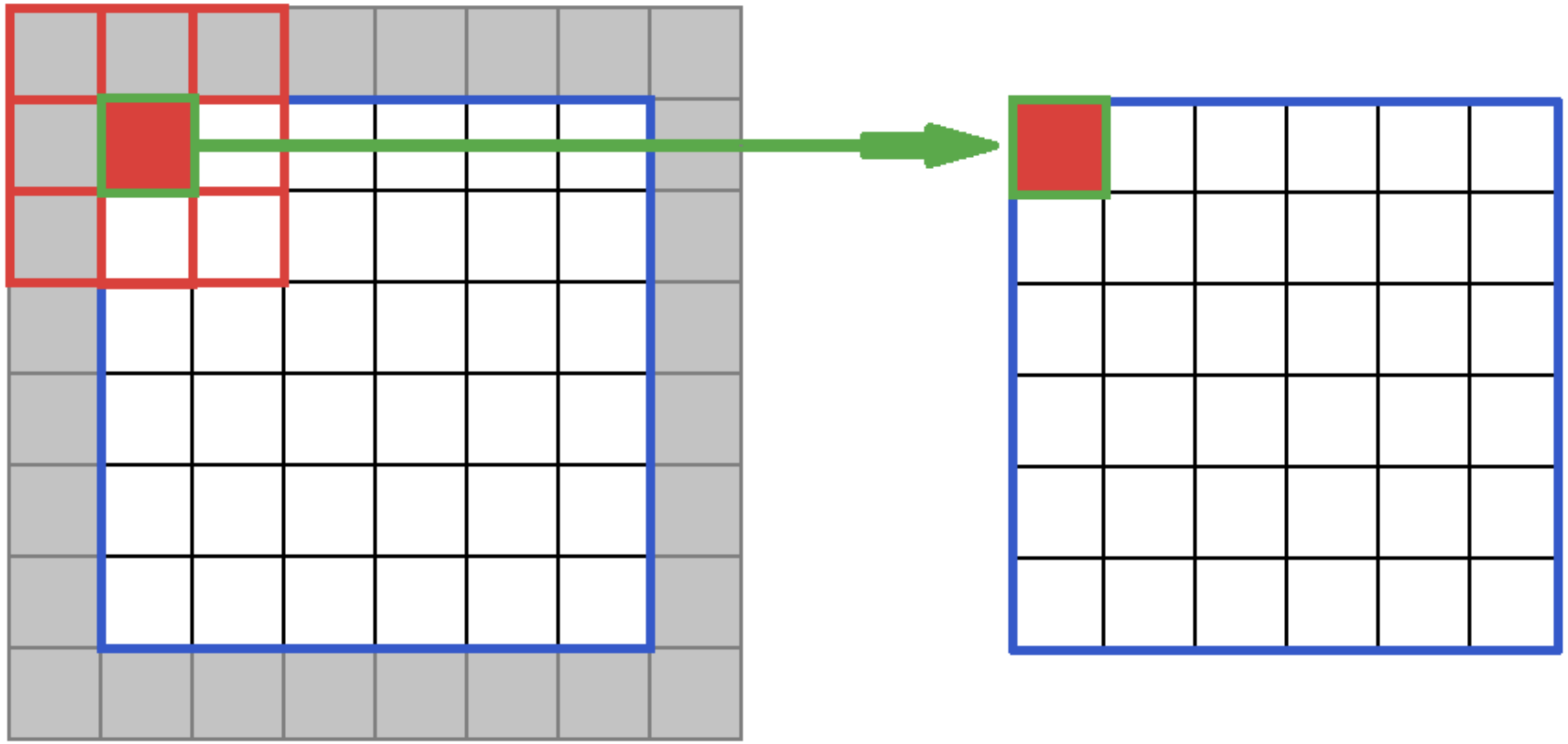
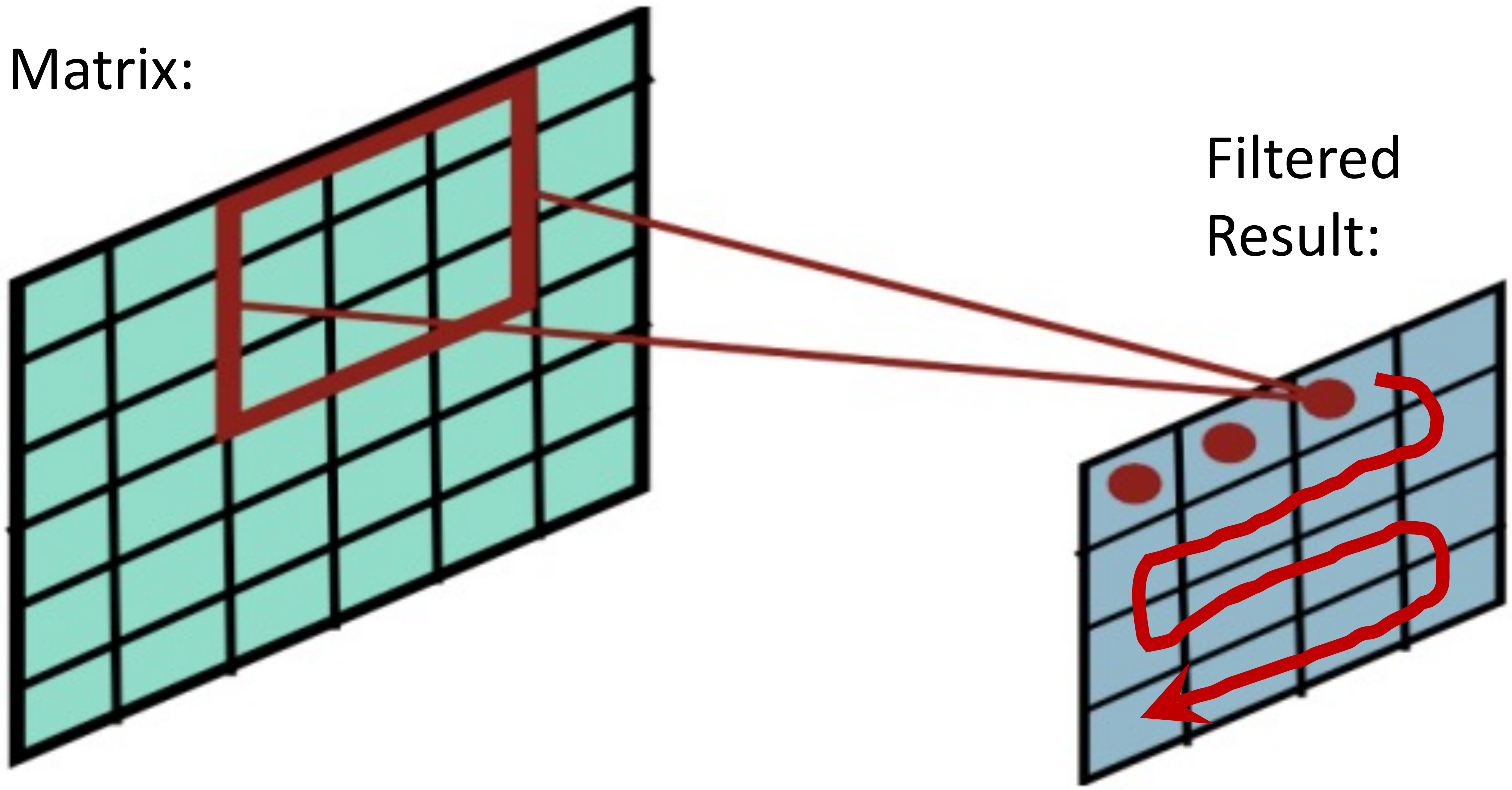


Image Credit: https://software.intel.com/en-us/node/586159

Problem #2: Computation Expensive



Many computations to slide filter over every point in the matrix and compute dot products

https://people.eecs.berkeley.edu/~jrs/189/lec/cnn.pdf

Idea: Reduce Computations with Stride

- **Stride**: how many steps taken spatially before applying a filter
  - e.g., 2x2

Image

1	1	1	0	0
0	1	1	1	0
0	0	1	1	1
0	0	1	1	0
0	1	1	0	0

Filter

1	0	1
0	1	0
1	0	1

Feature Map

4	4
2	4

http://deeplearning.net/software/theano/tutorial/conv_arithmetic.html

Convolutional Layers: Parameters vs Hyperparameters

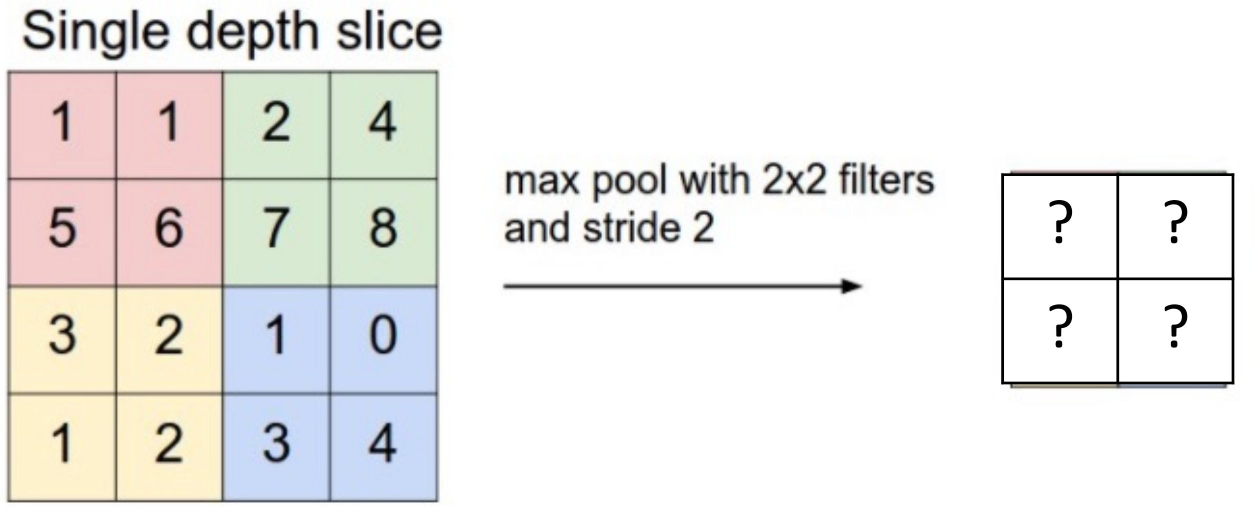
- Parameters
  - Weights
  - Biases
- Hyperparameters:
  - Number of filters, including height and width of each
  - Padding type
  - Strides

Today's Topics

- Neural Networks for Spatial Data
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Programming Tutorial

Pooling Layer: Summarizes Neighborhood

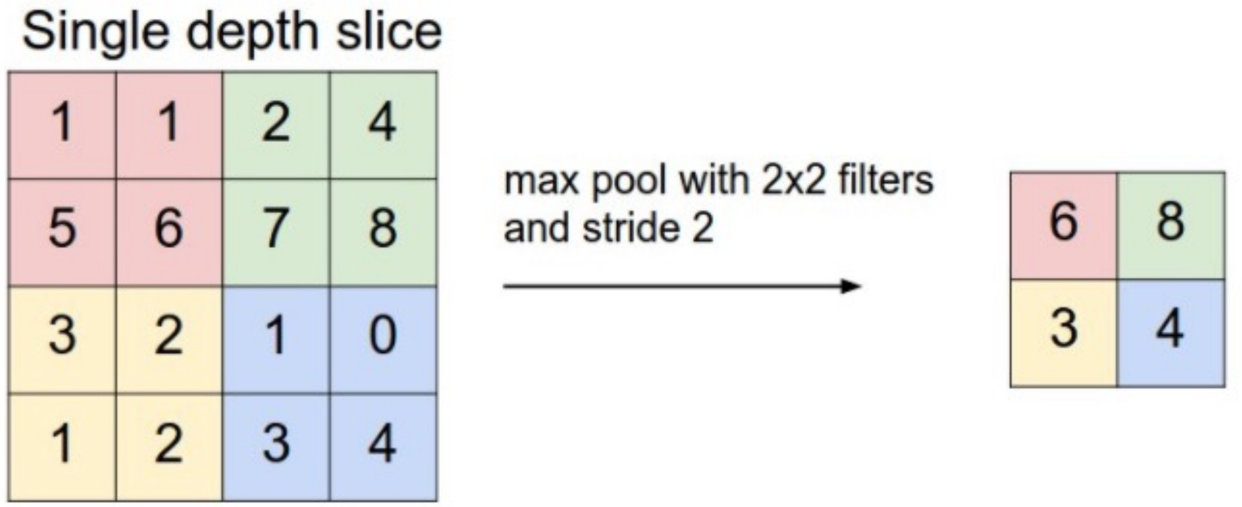
- **Max-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk



http://cs231n.github.io/convolutional-networks/#pool

Pooling Layer: Summarizes Neighborhood

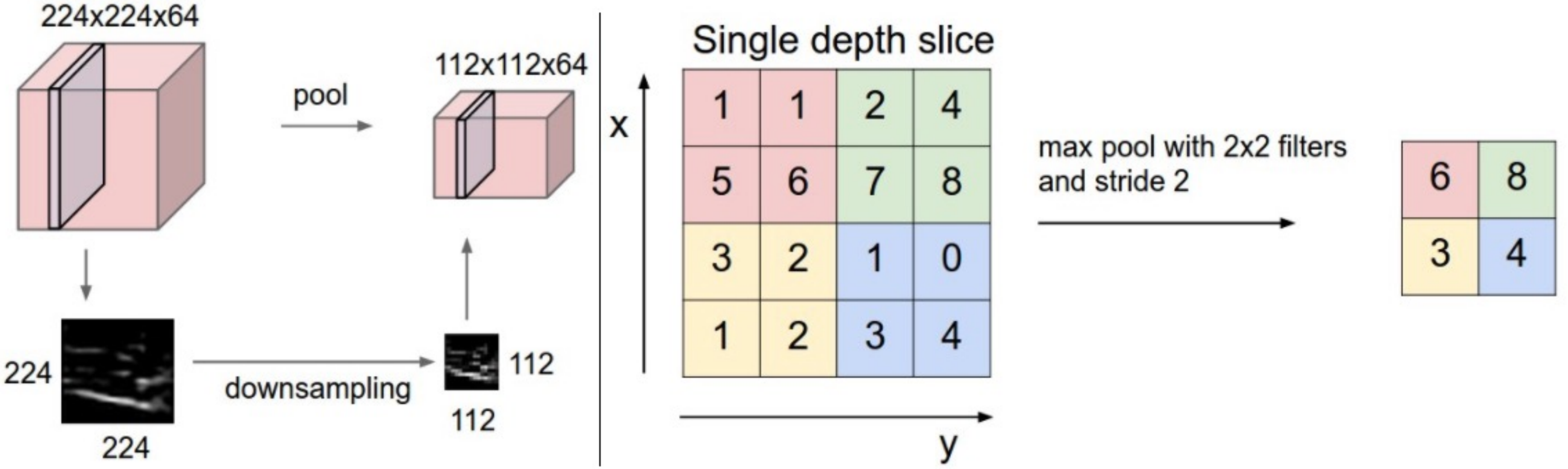
- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk



http://cs231n.github.io/convolutional-networks/#pool

Pooling Layer: Summarizes Neighborhood

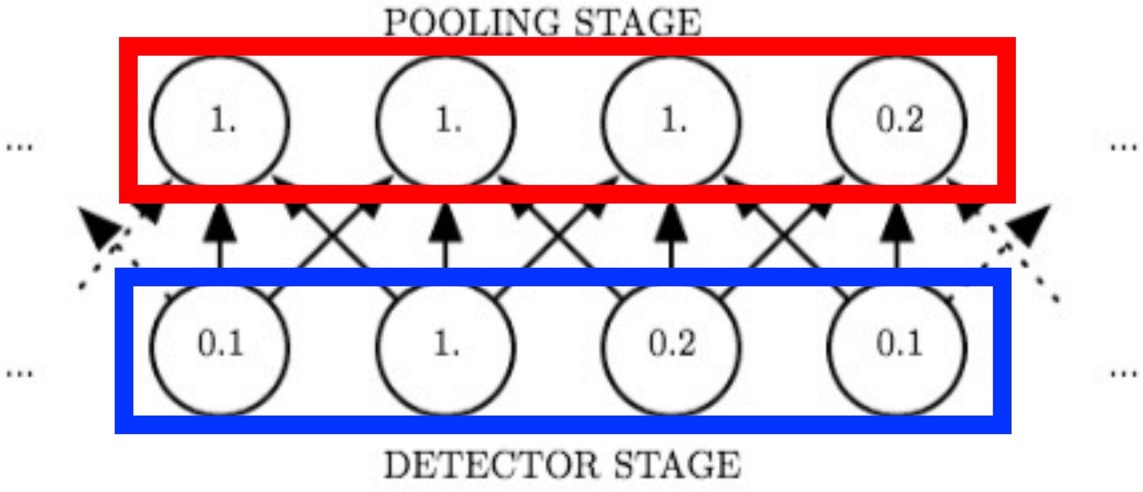
- **Max-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk



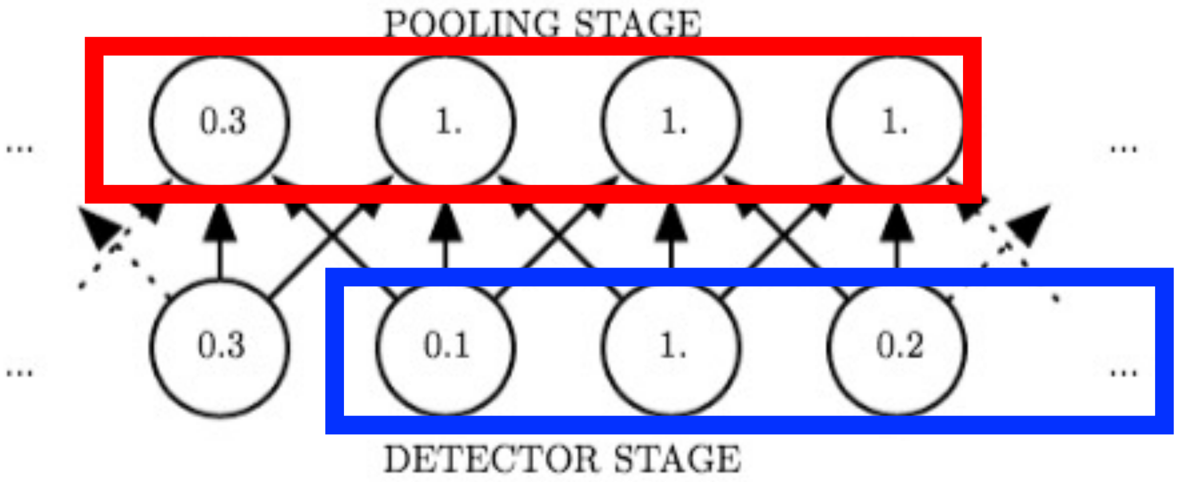
http://cs231n.github.io/convolutional-networks/#pool

Pooling Layer

- Resilient to small translations



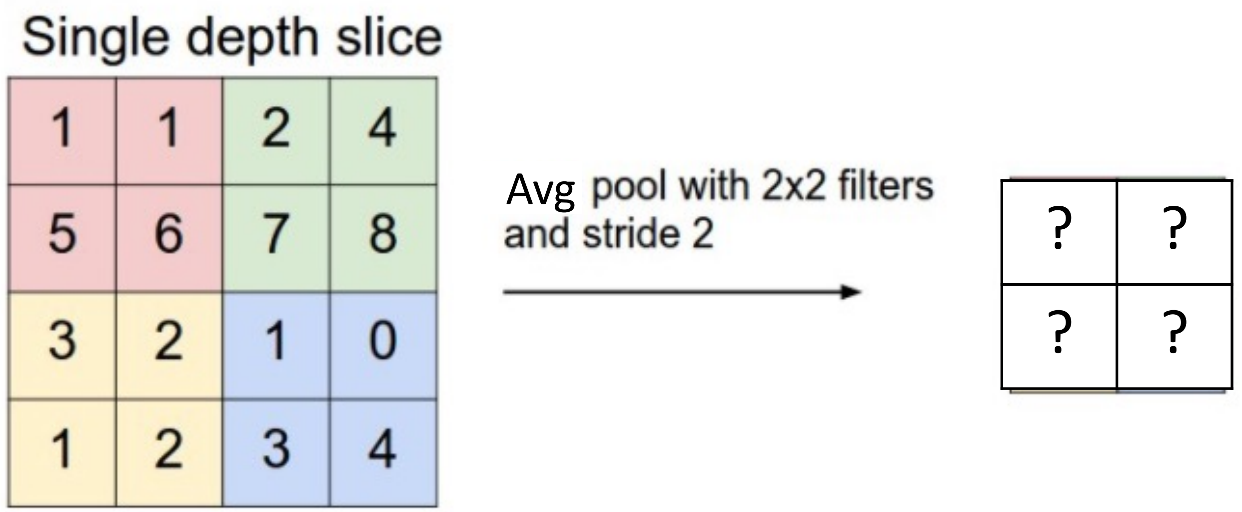
- e.g.,
  - Input: all values change (shift right)
  - Output: only half the values change



https://www.deeplearningbook.org/contents/convnets.html

Pooling Layer: Summarizes Neighborhood

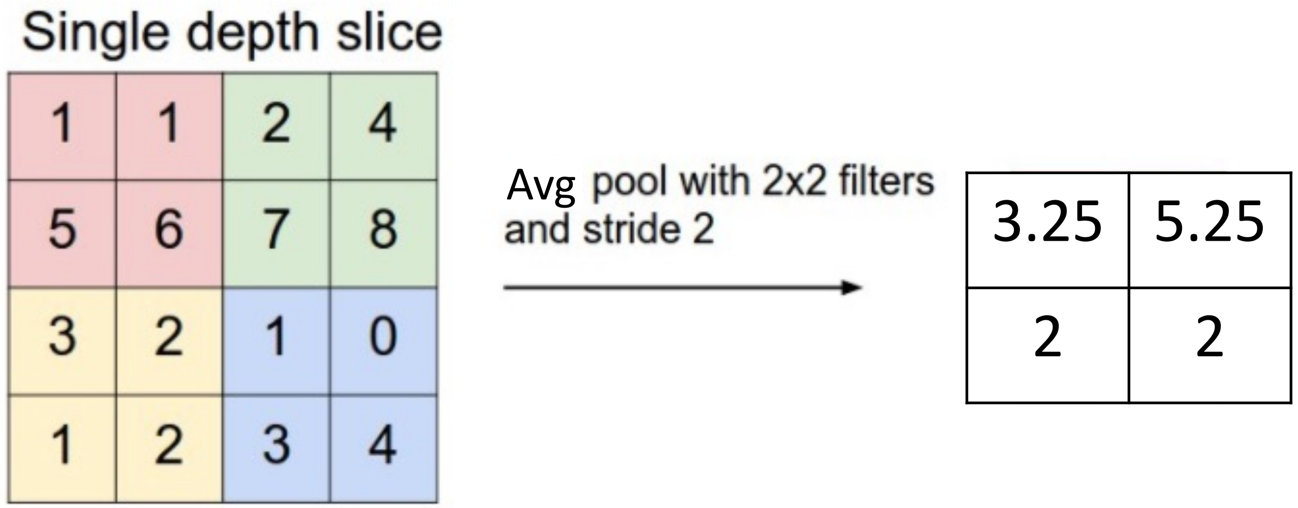
- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk
- **Average-pooling**: partitions input into a set of non-overlapping rectangles and outputs the average value for each chunk



http://cs231n.github.io/convolutional-networks/#pool

Pooling Layer: Summarizes Neighborhood

- **Max-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk
- **Average-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the average value for each chunk



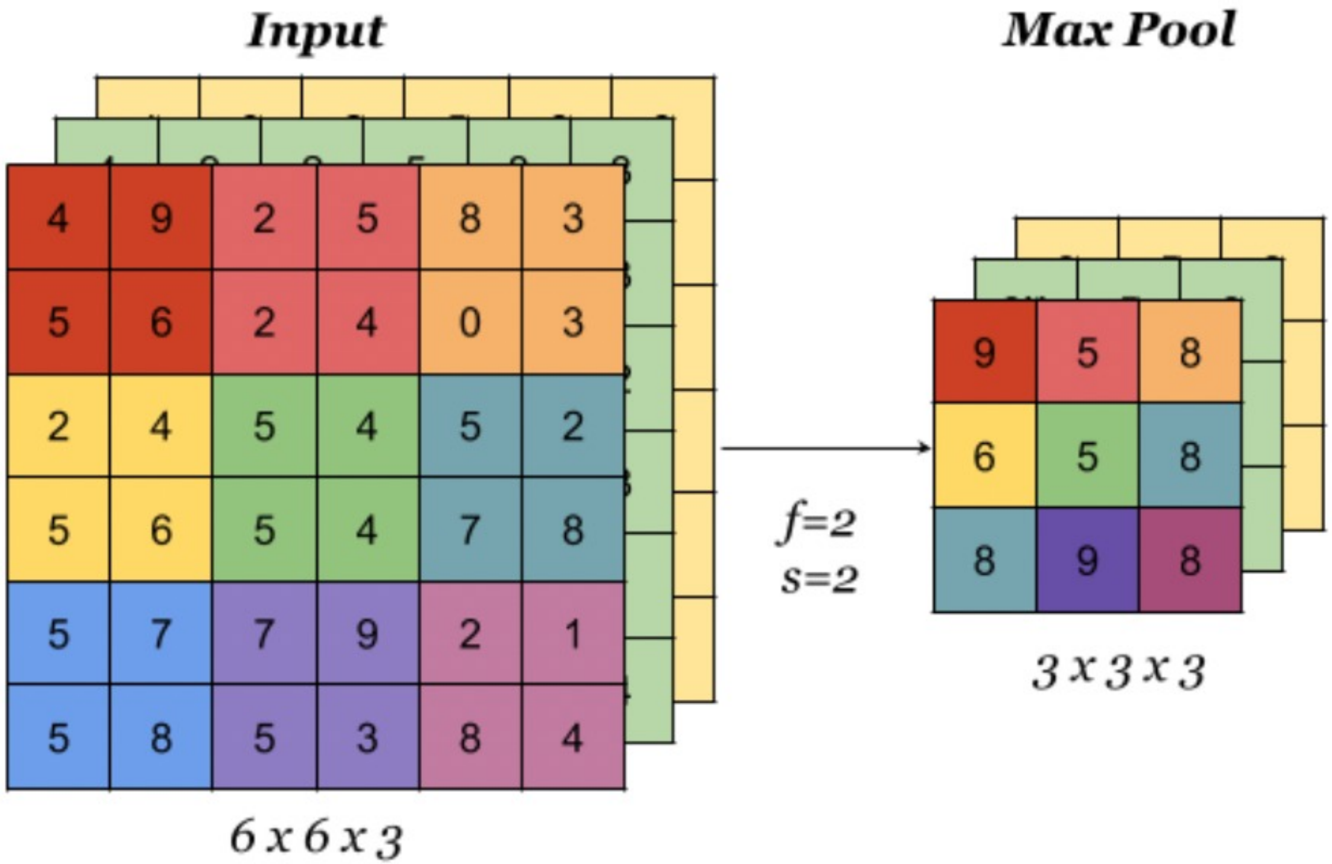
http://cs231n.github.io/convolutional-networks/#pool

Pooling Layer: Summarizes Neighborhood

- **Max-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk
- **Average-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the average value for each chunk

- And many more pooling options
  - E.g., listed here https://pytorch.org/docs/stable/…….tml#pooling-layers

Pooling for Multi-Channel Input



Pooling is applied to each input channel separately

https://indoml.com/2018/03/07/student-notes-convolutional-neural-networks-cnn-introduction/

Pooling Layer: Benefits

- Builds in invariance to translations of the input
- Reduces memory requirements
- Reduces computational requirements

Today’s Topics

- Neural Networks for Spatial Data
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Programming Tutorial

Today's Topics

- Neural Networks for Spatial Data
- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Programming Tutorial

The End