Convolutional Neural Networks (CNNs)

Danna Gurari
University of Colorado Boulder
Spring 2025

https://dannagurari.colorado.edu/course/neural-networks-and-deep-learning-spring-2025/

Review

- Last class:
  - Model capacity: how it affects learning
  - Regularization: learning methods for improving model generalization
  - Hyperparameter selection: tuning to improve model performance
  - Programming tutorial

- Assignments (Canvas):
  - Problem set 1 grades are out
    - Review session will be held at 4pm
    - All regrade requests must be emailed to our TA, Nick Cooper (a comment in Canvas is not sufficient)
  - Problem set 2 due earlier today
  - Lab assignment 1 due a week from Thursday (in 9 days)

- Questions?

Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Pioneering CNN model: LeNet

Today's Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Pioneering CNN model: LeNet

Inspiration: Neural Networks for Spatial Data

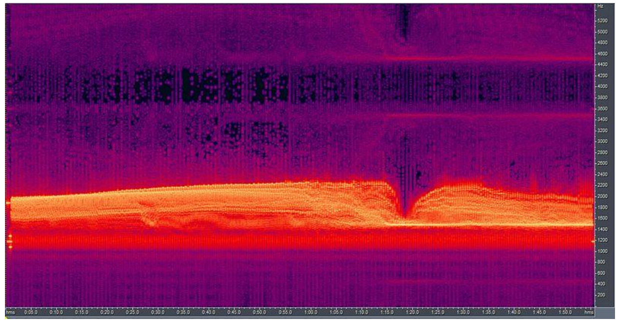
- Data where the order matters; e.g.,

2D

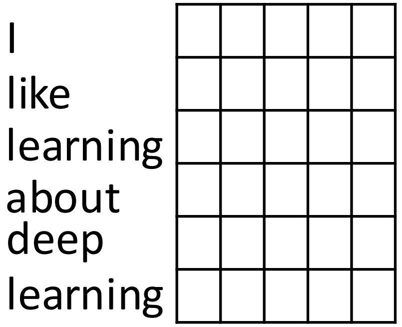
3D

Images

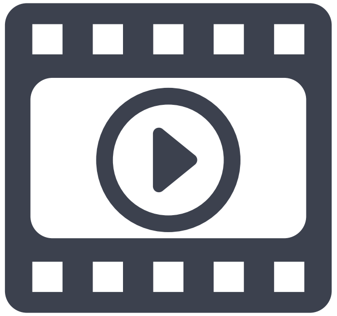
Audio (spectrogram)



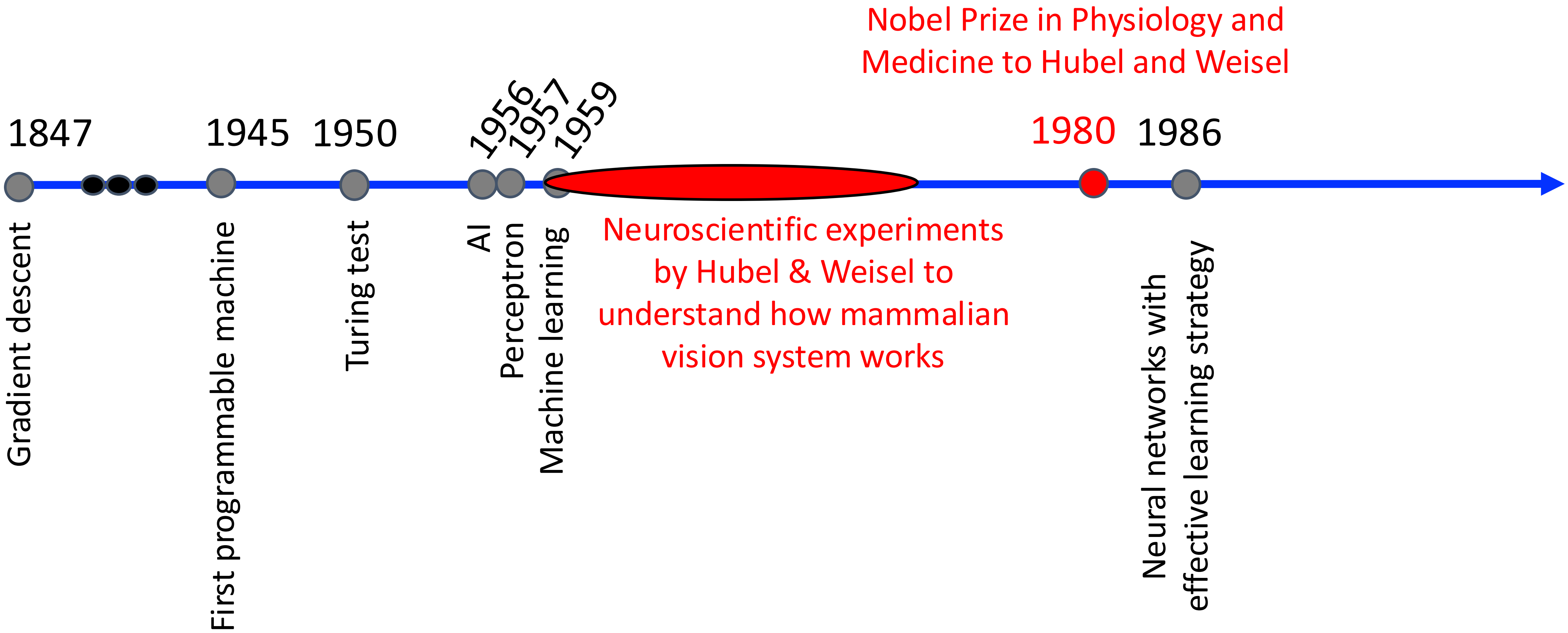
Text (word embeddings)



Video



Inspiration: Historical Context

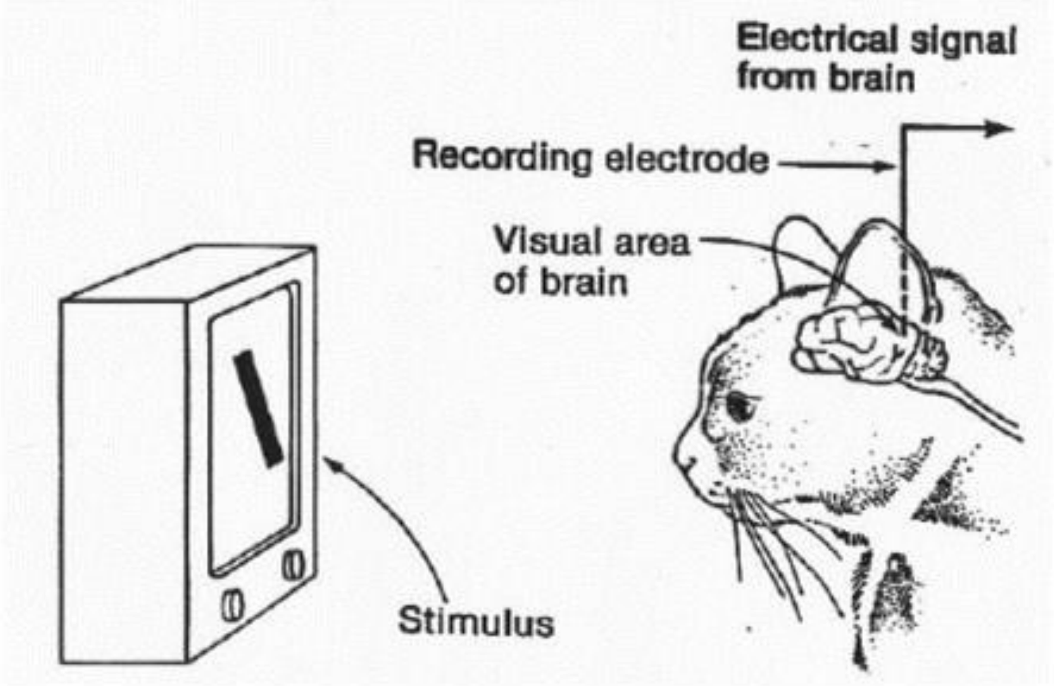


Inspiration: How Vision System Works

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

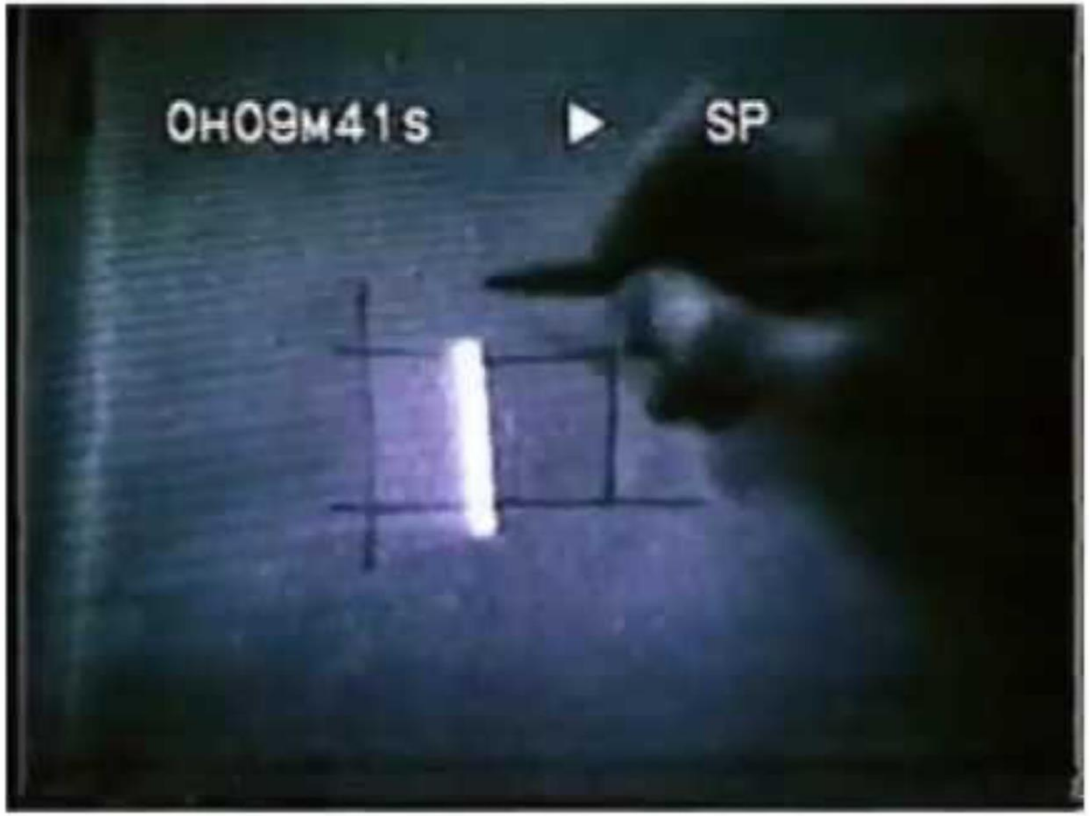
Inspiration: How Vision System Works

Experiment Set-up:



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

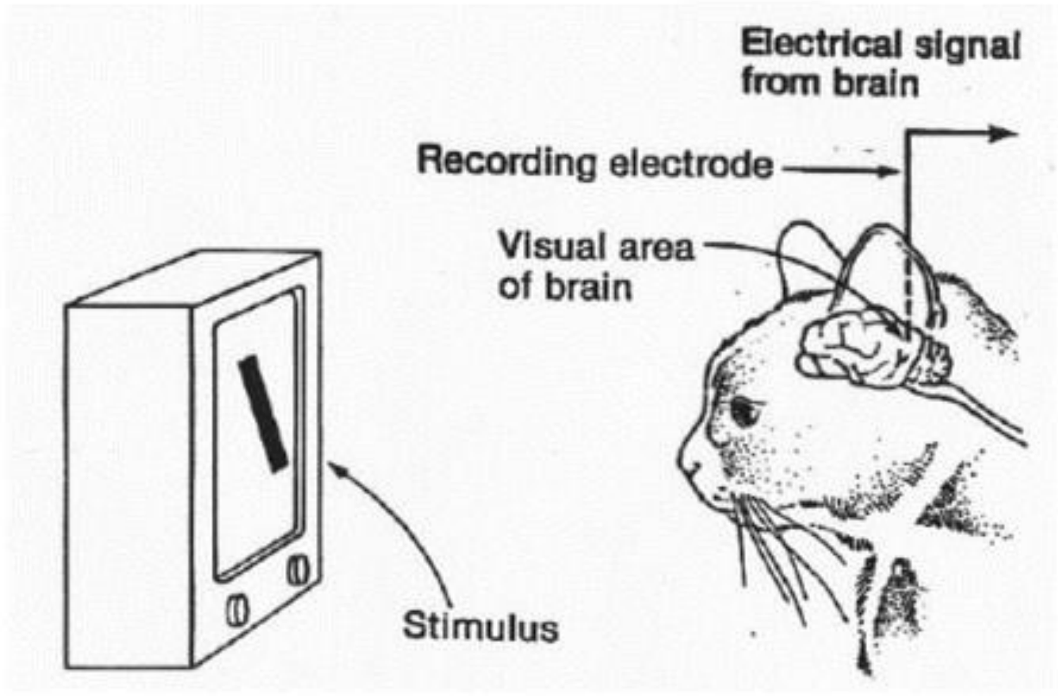
Key Finding: neurons respond strongly only when light is shown in certain orientations



https://www.youtube.com/watch?v=OGxVfKJqX5E&ab_channel=RyanAbbott

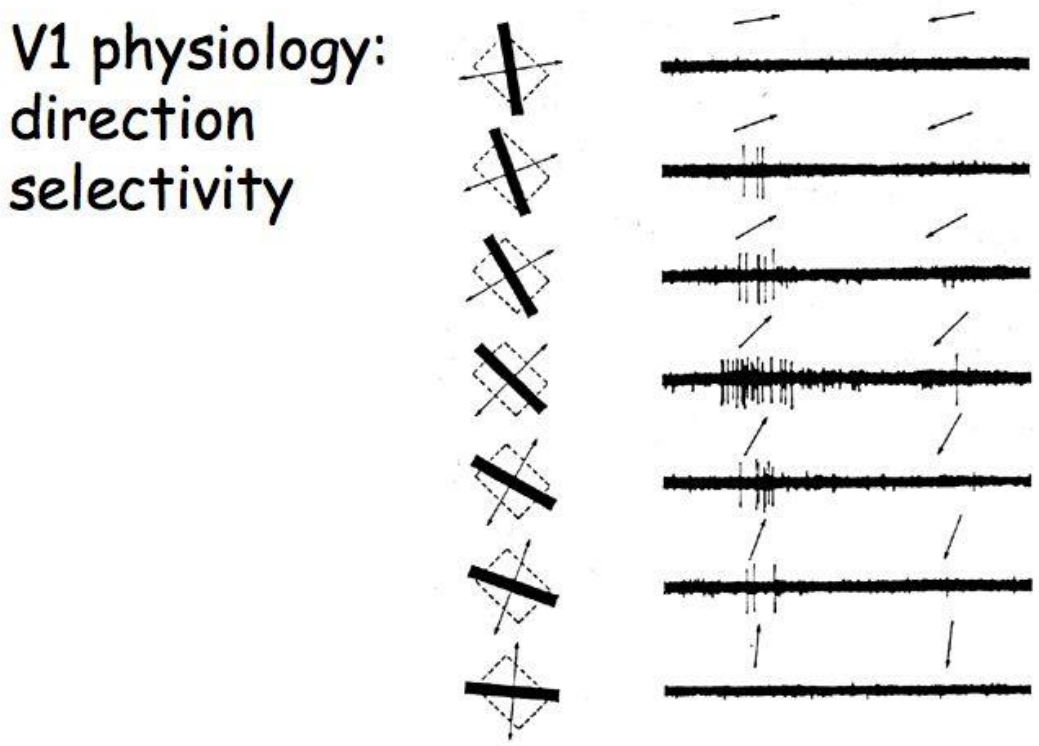
Inspiration: How Vision System Works

Experiment Set-up:



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

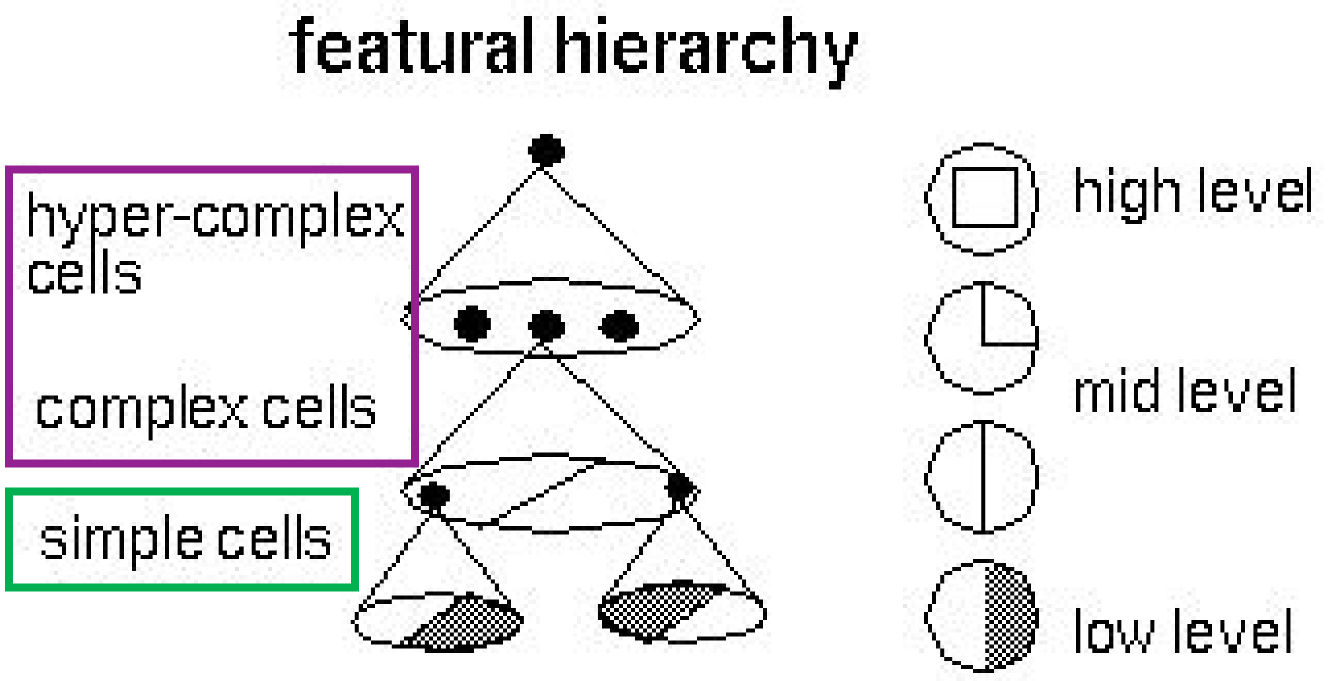
Key Finding: neurons respond strongly only when light is shown in certain orientations



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

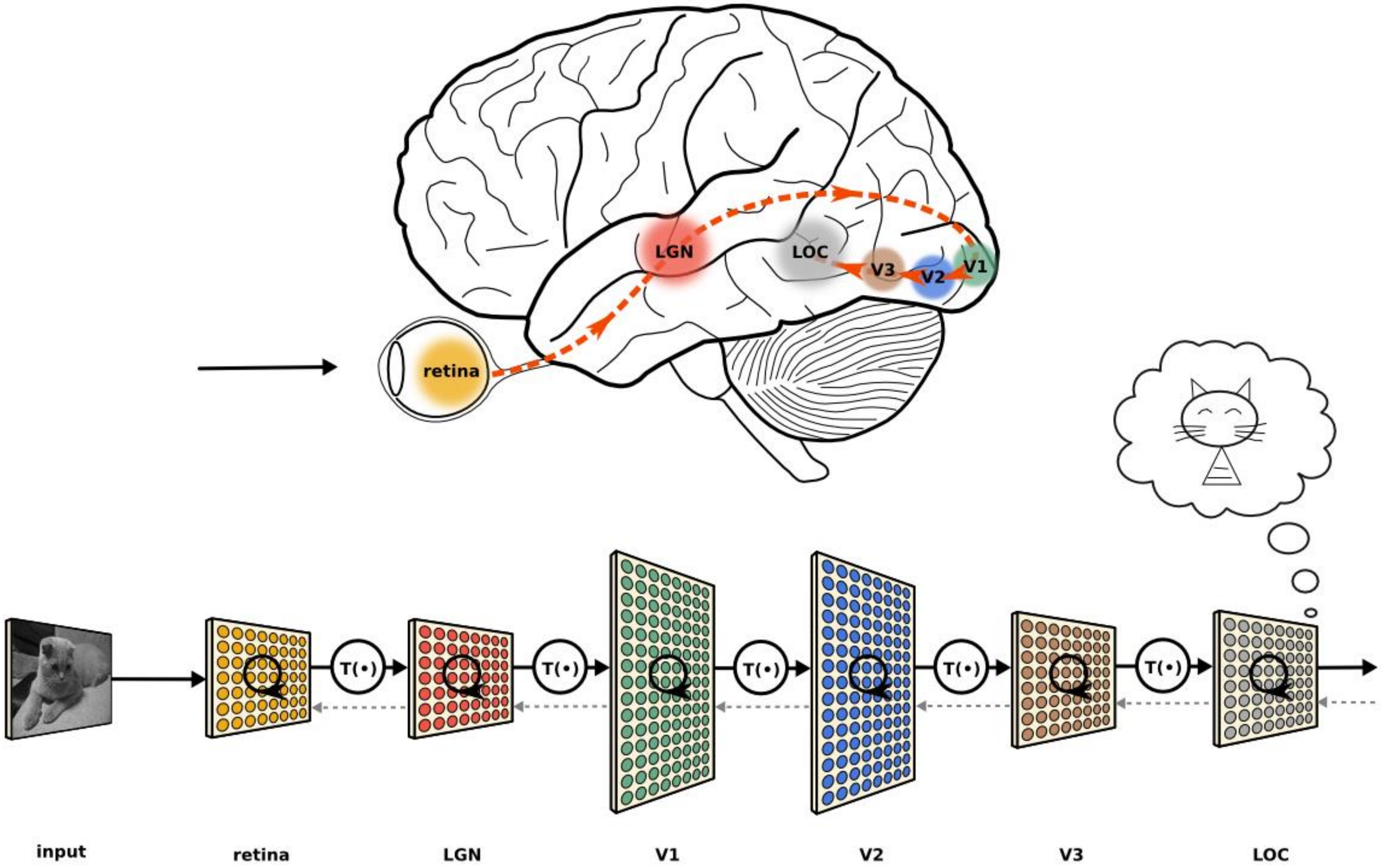
Inspiration: How Vision System Works

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



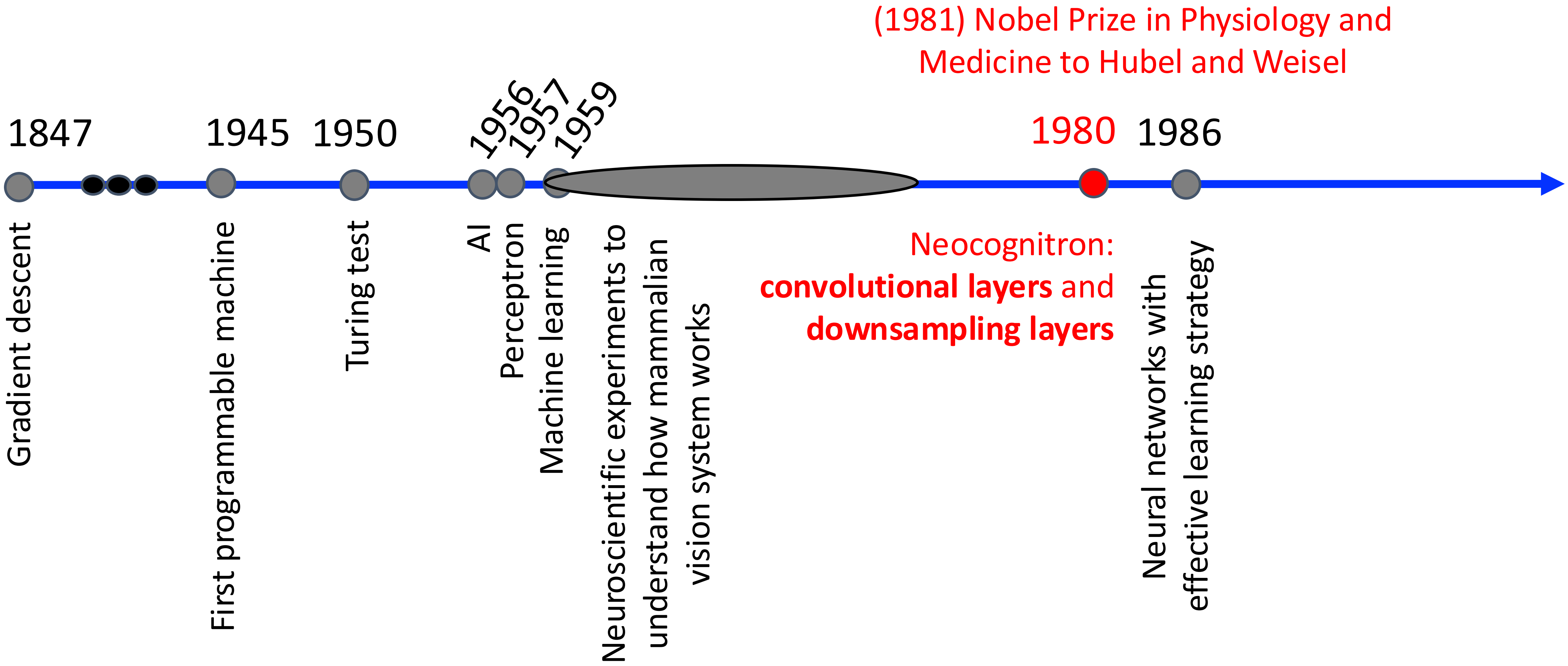
https://bruceoutdoors.files.wordpress.com/2017/08/hubel.jpg

Inspiration: How Vision System Works



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

Key Ingredients of CNNs



Neocognitron: Key Ingredients

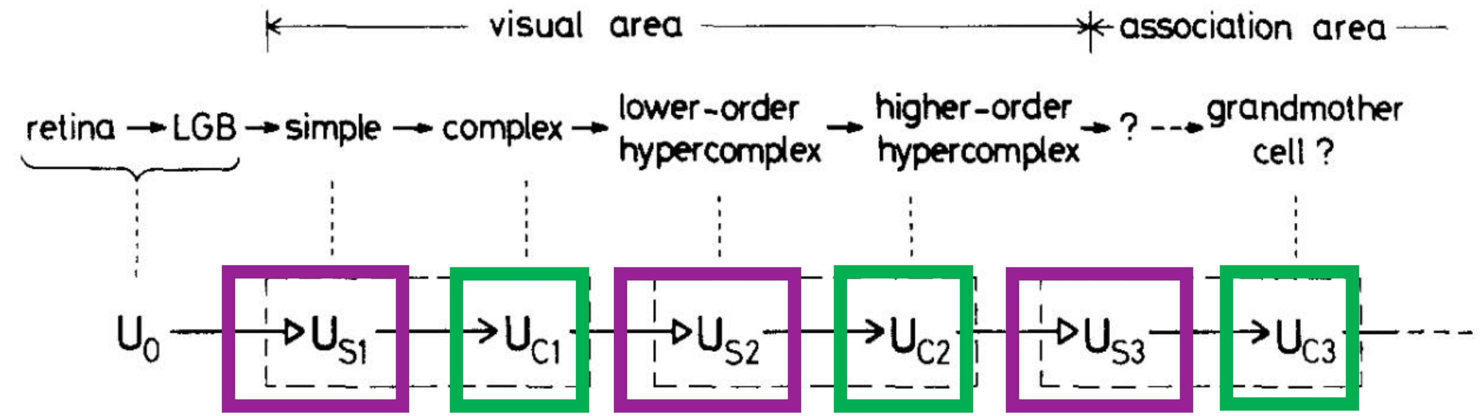
http://personalpage.flsi.or.jp/fukushima/index-e.html

"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."

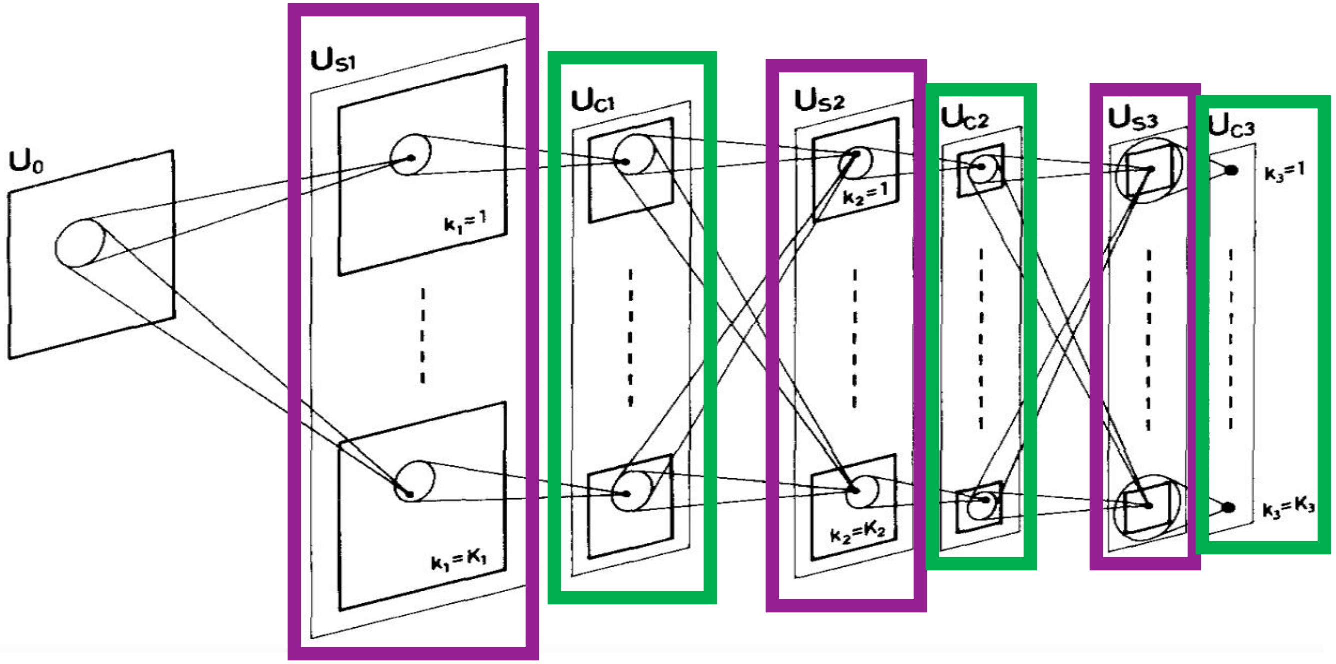
- Fukushima, Neocognitron: A Self-organizing Neural Network Model for a Mechanism of Pattern Recognition Unaffected by Shift in Position. Biological Cybernetics, 1980.

Neocognitron: Key Ingredients

Cascade of simple and complex cells identified by Hubel and Weisel:



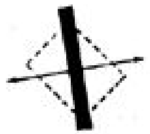
Model with alternating convolutional layers and pooling layers:

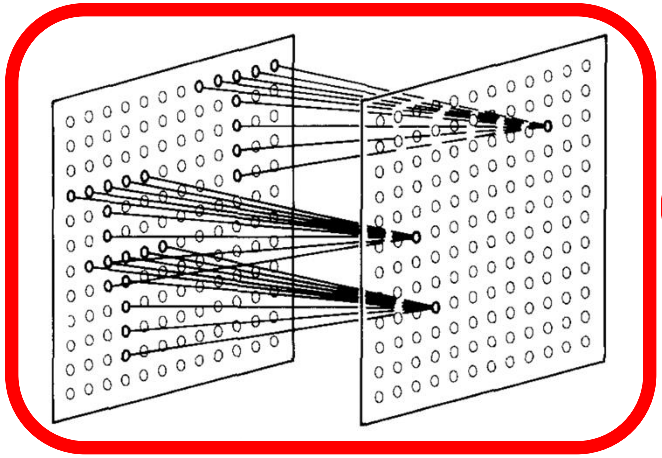


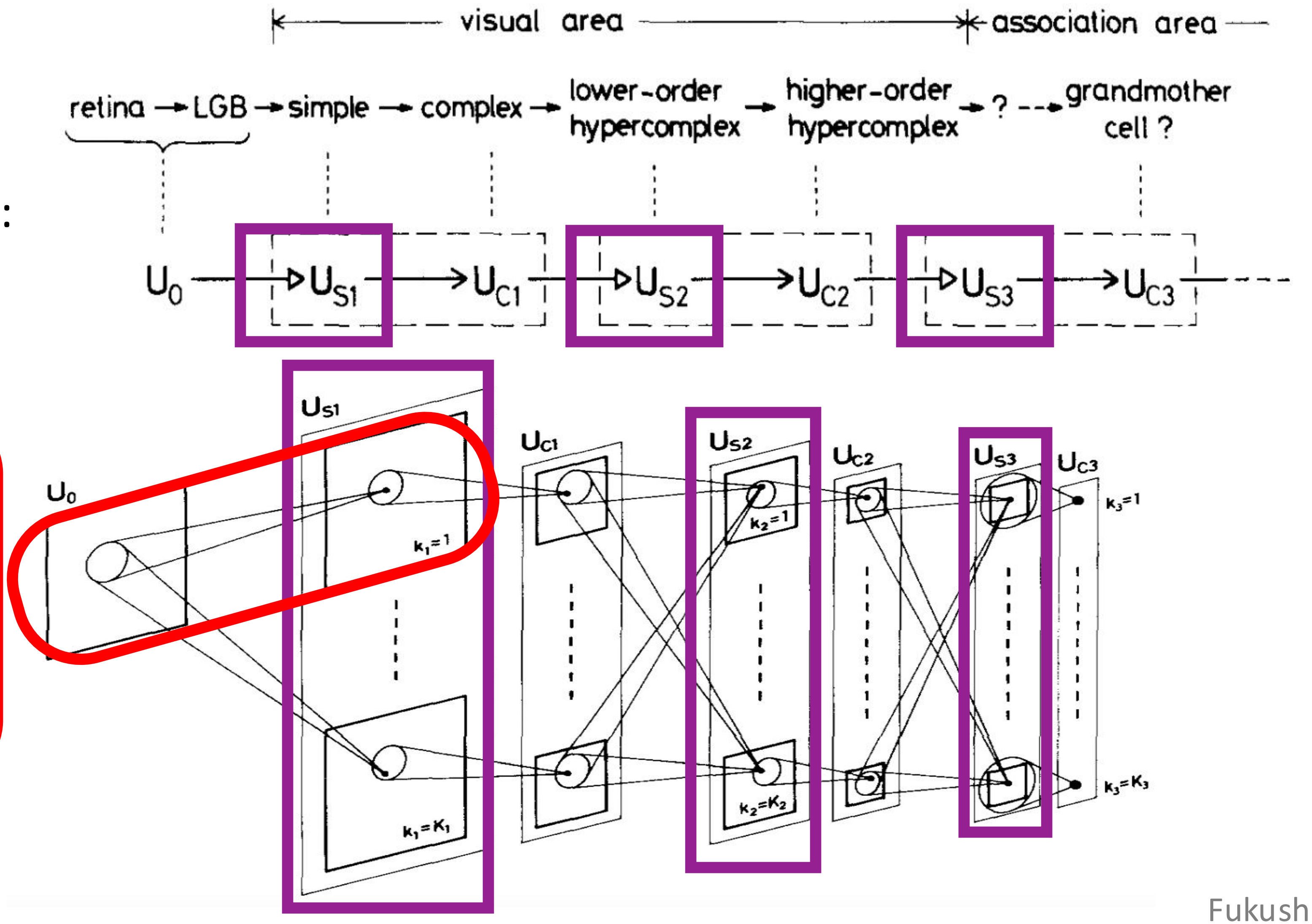
Fukushima, 1980

Neocognitron: Key Ingredients

Simple cells use a sliding filter to identify local features (e.g., orientations):



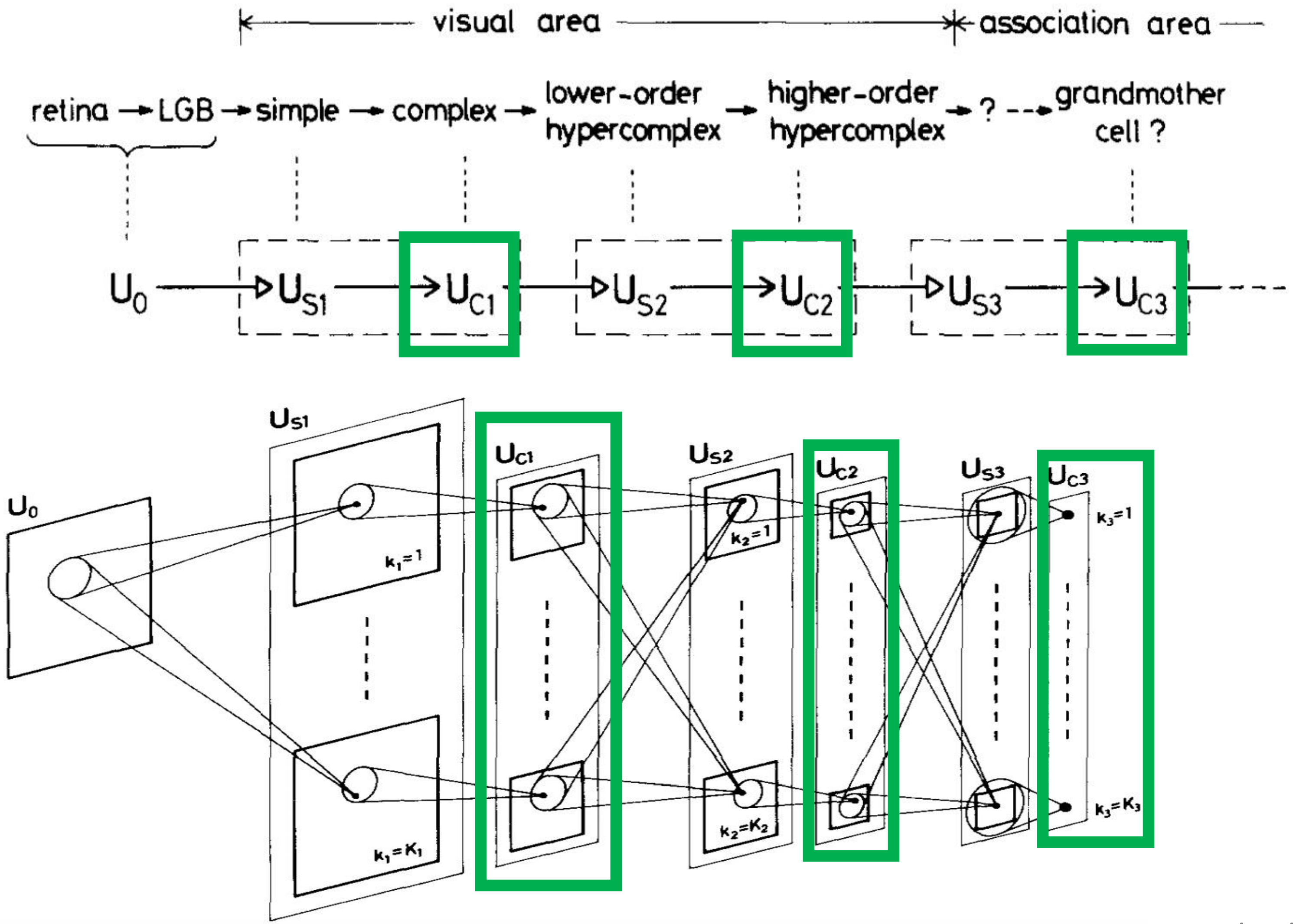




Fukushima, 1980

Neocognitron: Key Ingredients

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



Fukushima, 1980

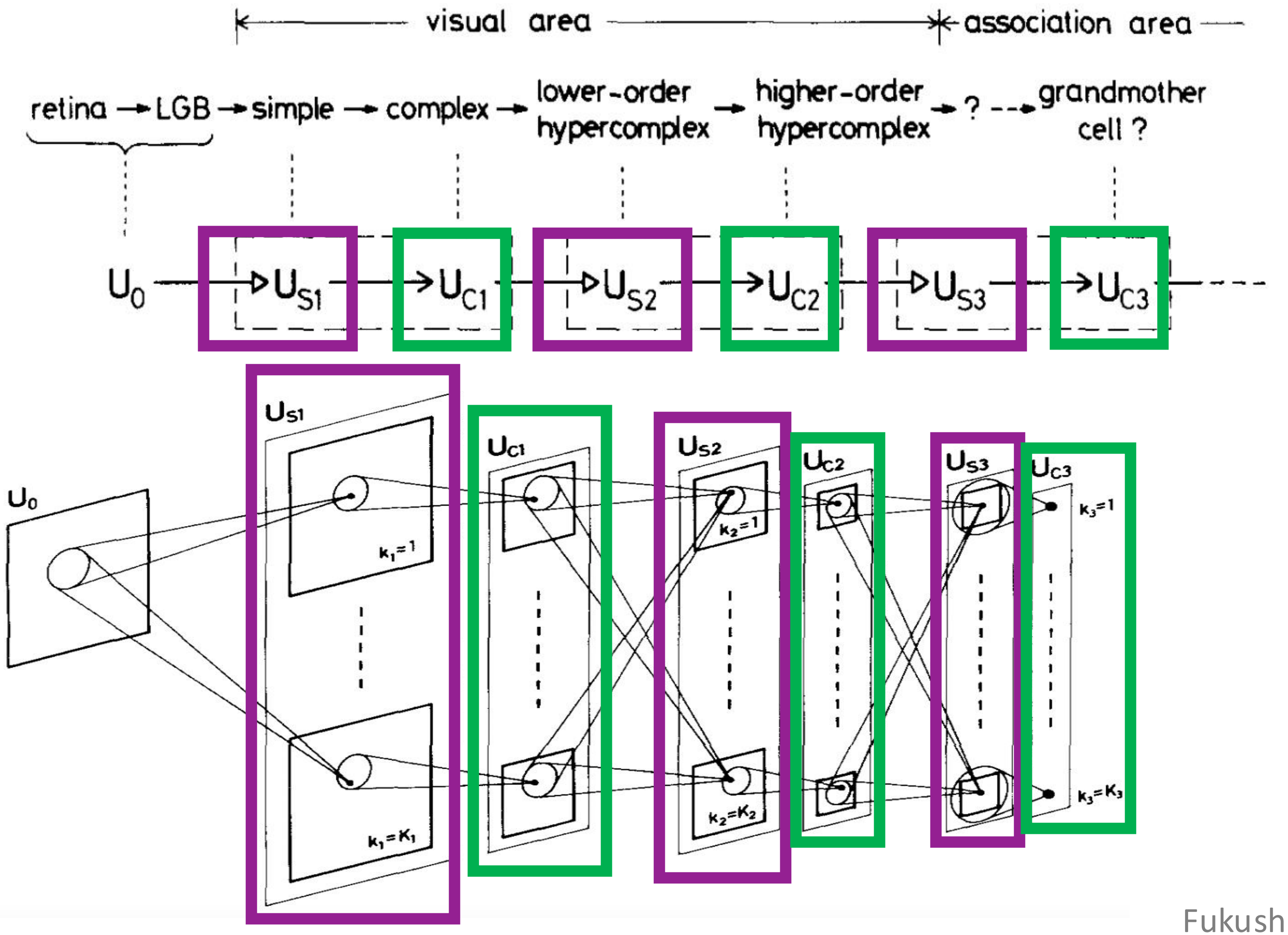
Neocognitron: Key Ingredients

1. Convolutional layers

modifiable synapses

unmodifiable synapses

2. Pooling Layers

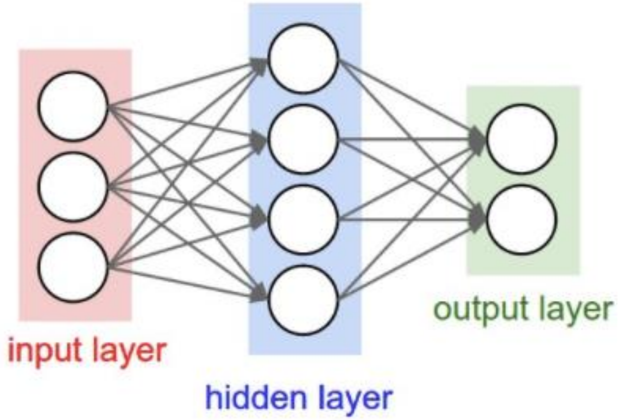


Fukushima, 1980

Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Pioneering CNN model: LeNet

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

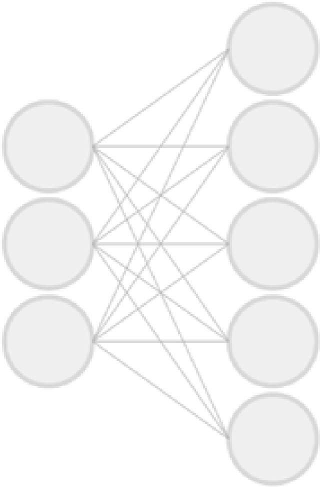
Motivation: Fully-Connected Layers Are Limited

Concern: many model parameters…

- increases chance of overfitting
- increases memory/storage requirements
- increases computational expense

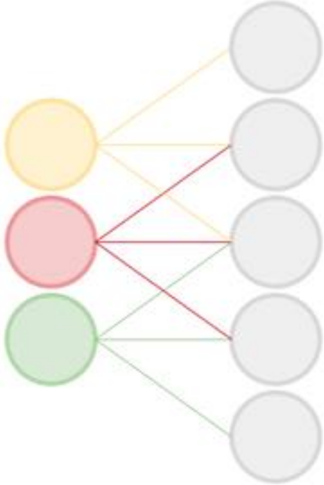
Idea: Convolutional Layers

Fully-connected:



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

Convolutional:

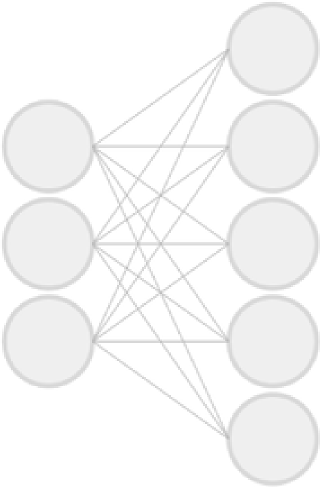


each node receives input only from a small neighborhood in previous layer (and there is parameter sharing)

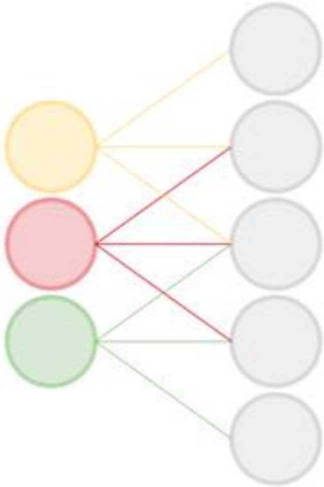
https://qph.fs.quoracdn.net/main-qimg-2e1f0071ca9878f7719ed0ea8aeb386d

Fully-Connected vs Convolutional Layers

Fully-connected:



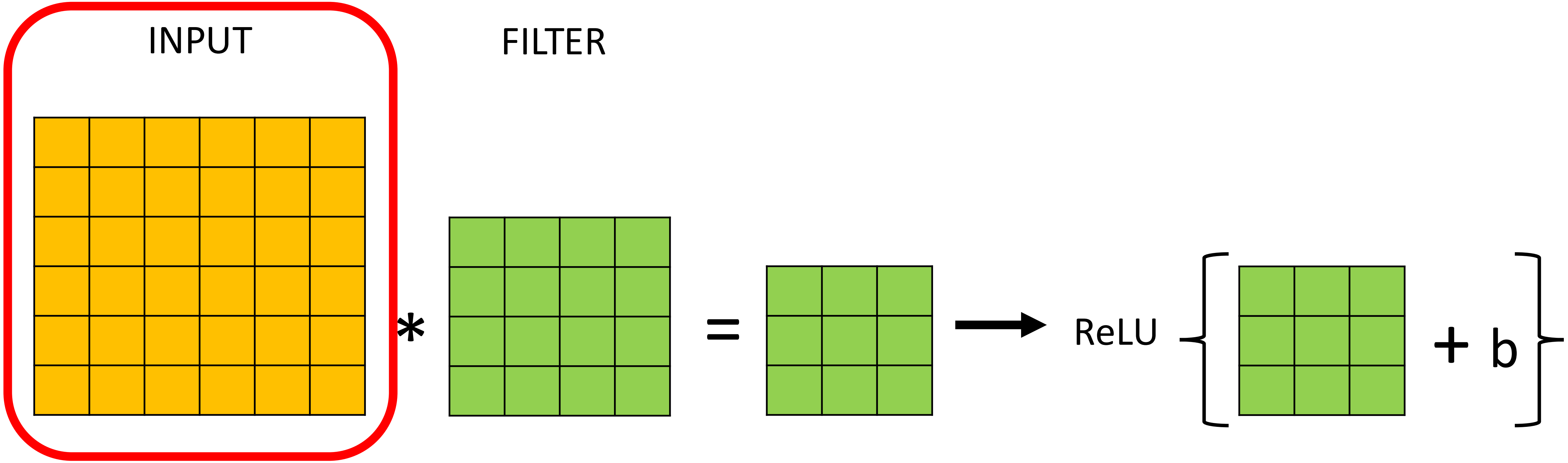
Convolutional:



Convolutional layers dramatically reduce number of model parameters!

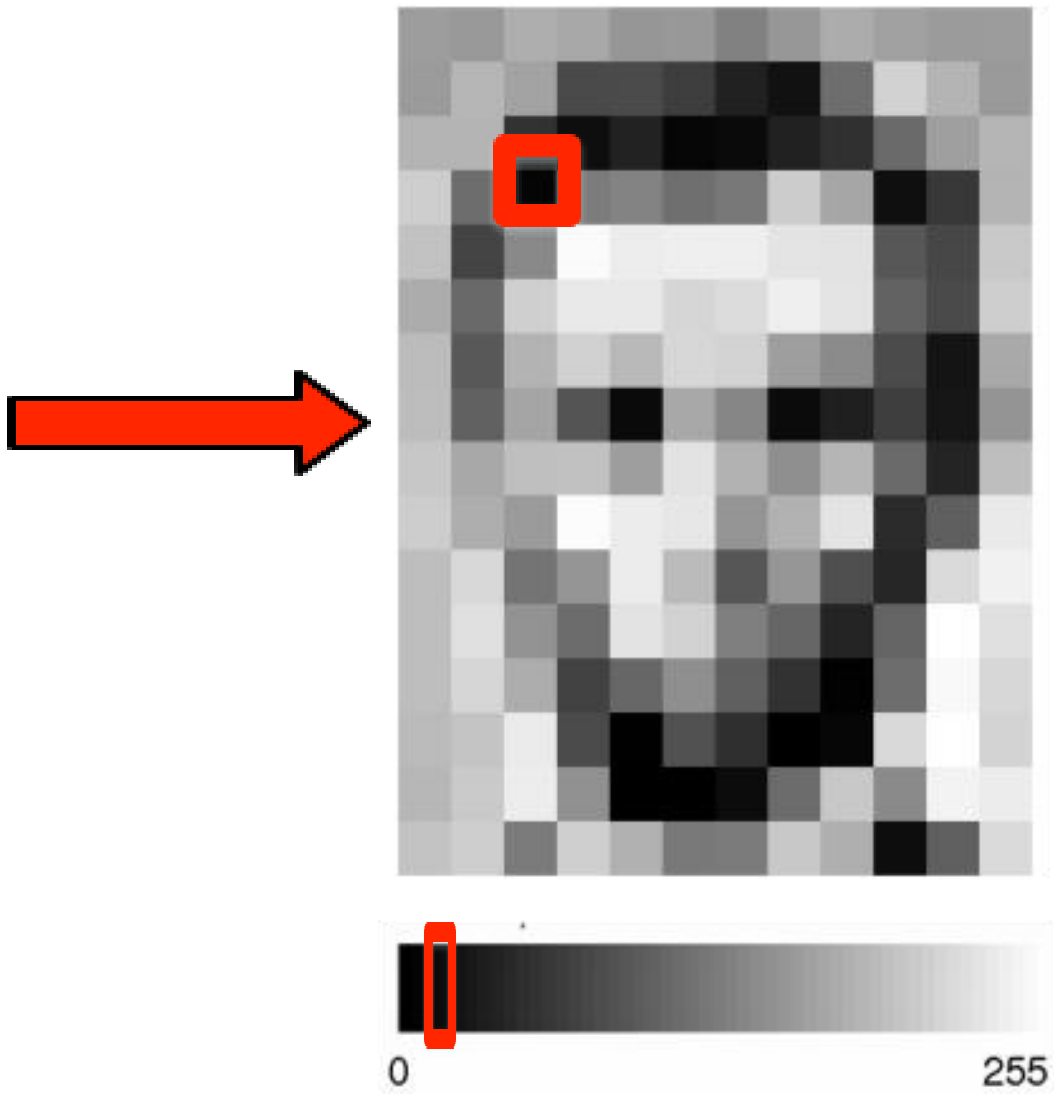
https://qph.fs.quoracdn.net/main-qimg-2e1f0071ca9878f7719ed0ea8aeb386d

Key Ingredient 1: Convolutional Layers



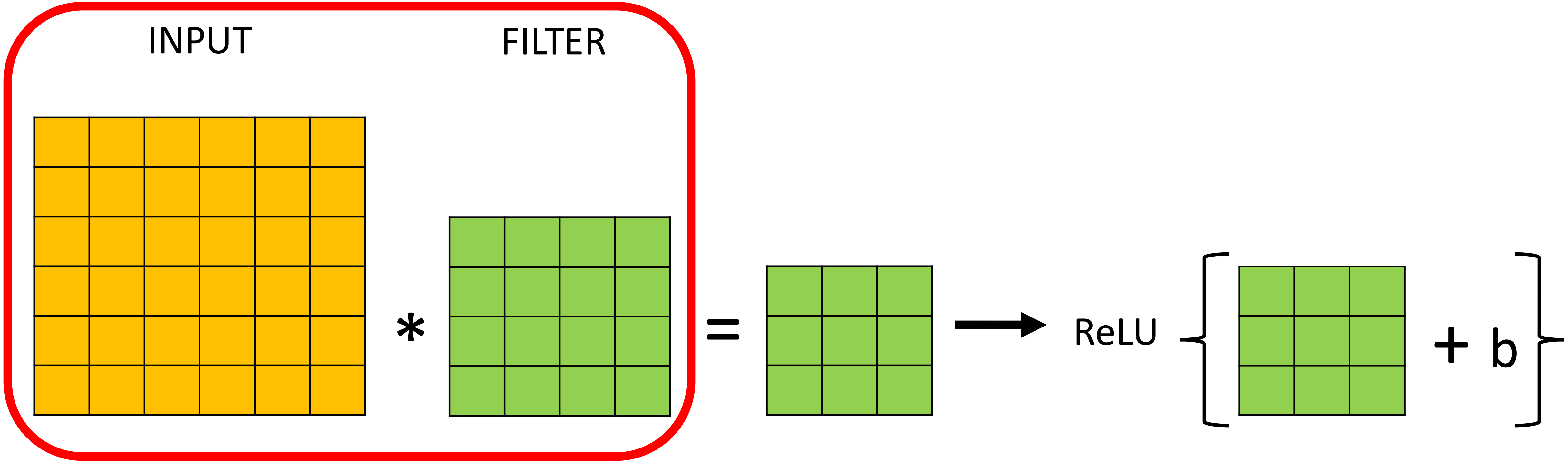
Recall: Image Representation (8-bit Grayscale)

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	10[illegible]	5	[illegible]24	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	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

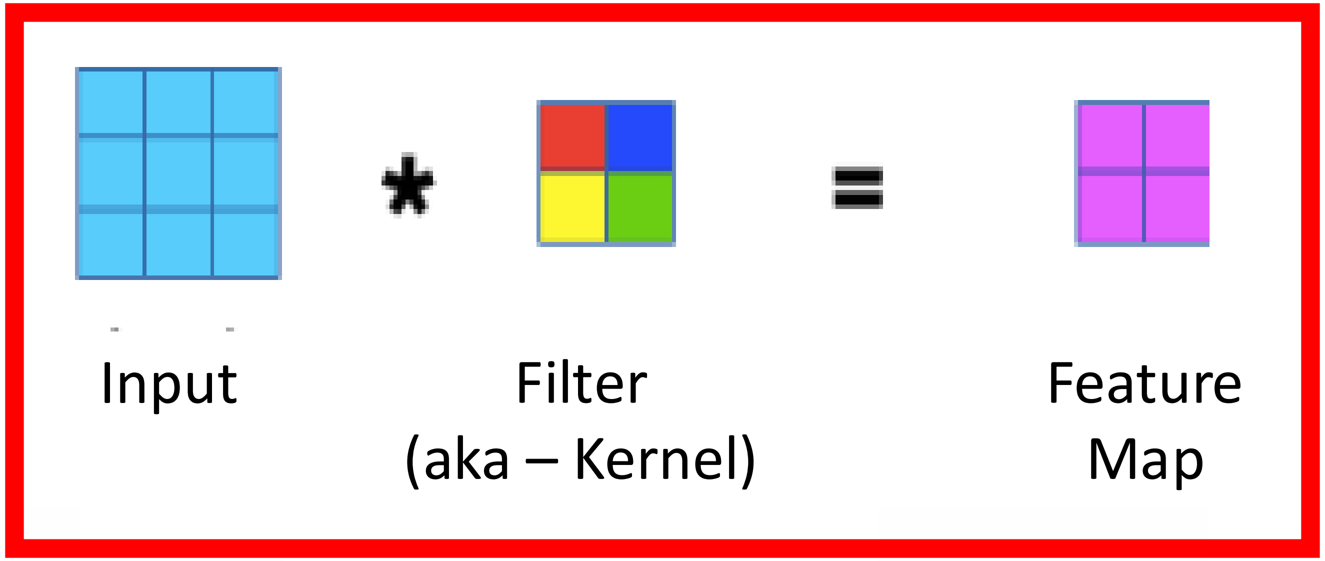


https://ai.stanford.edu/~syyeung/cvweb/tutorial1.html

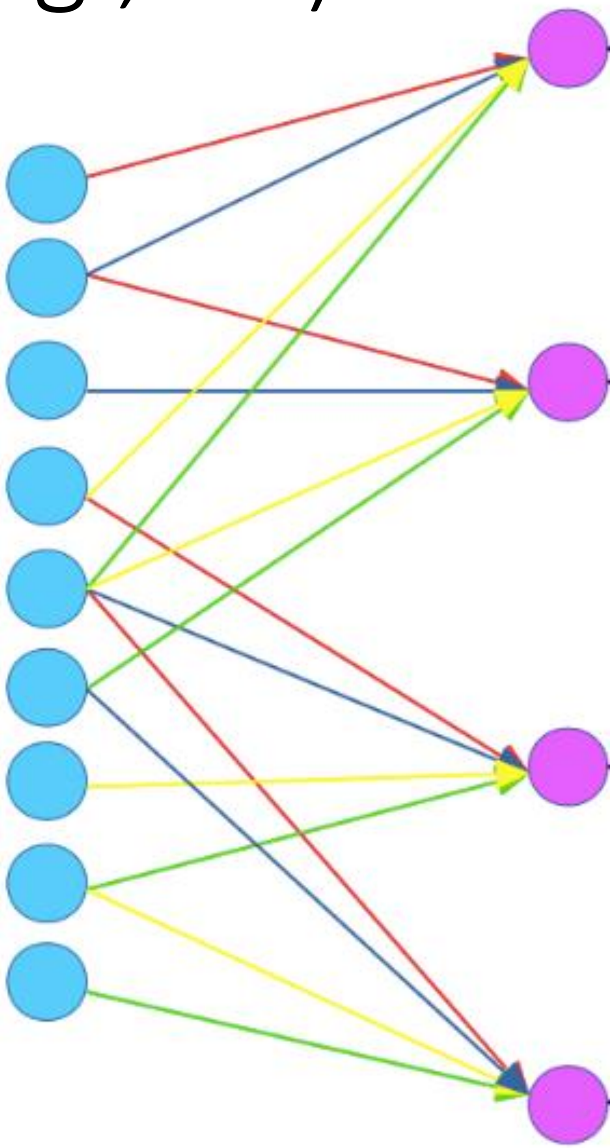
Key Ingredient 1: Convolutional Layers



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



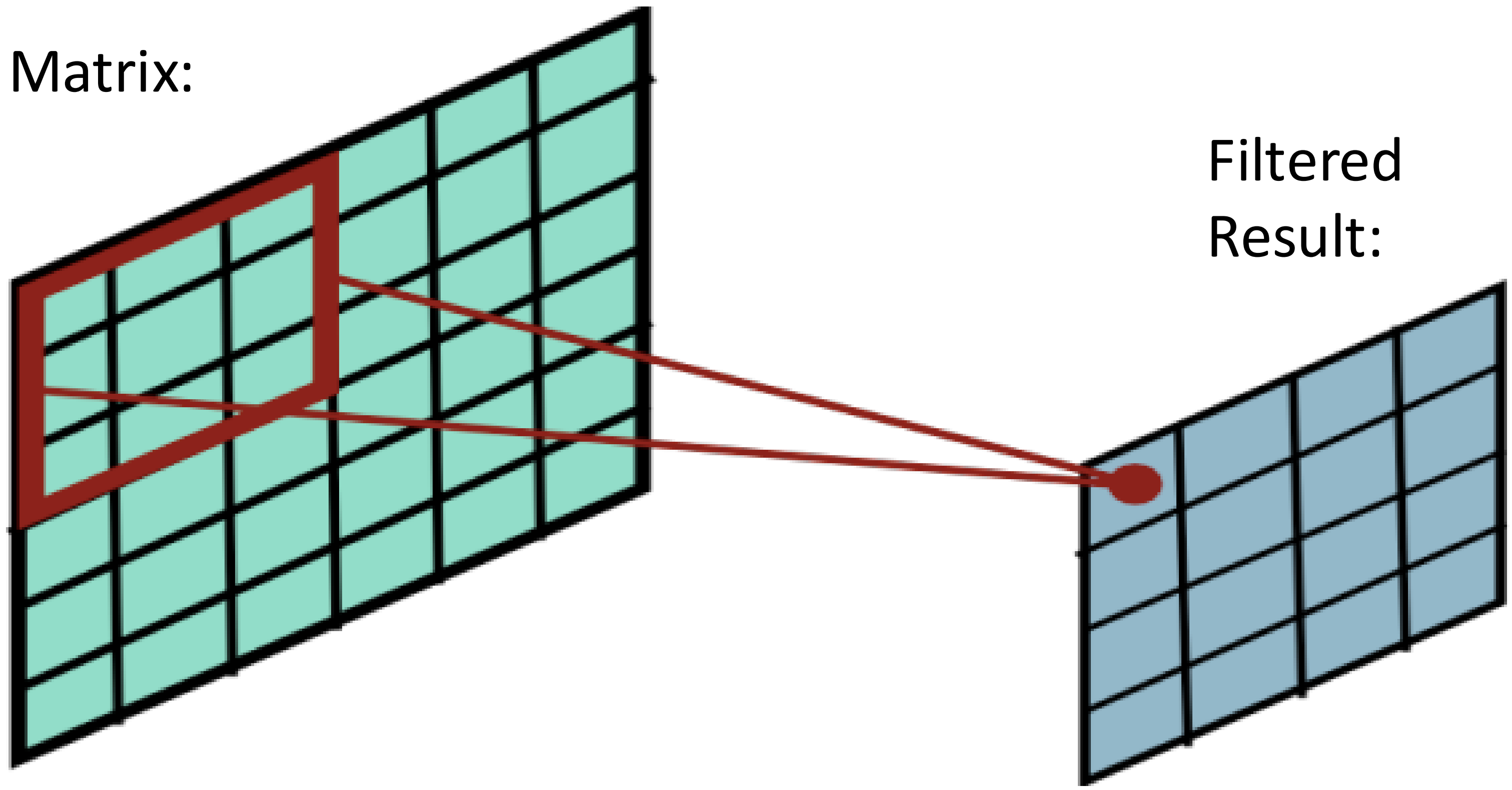
Way to Interpret
Neural Network



- 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/

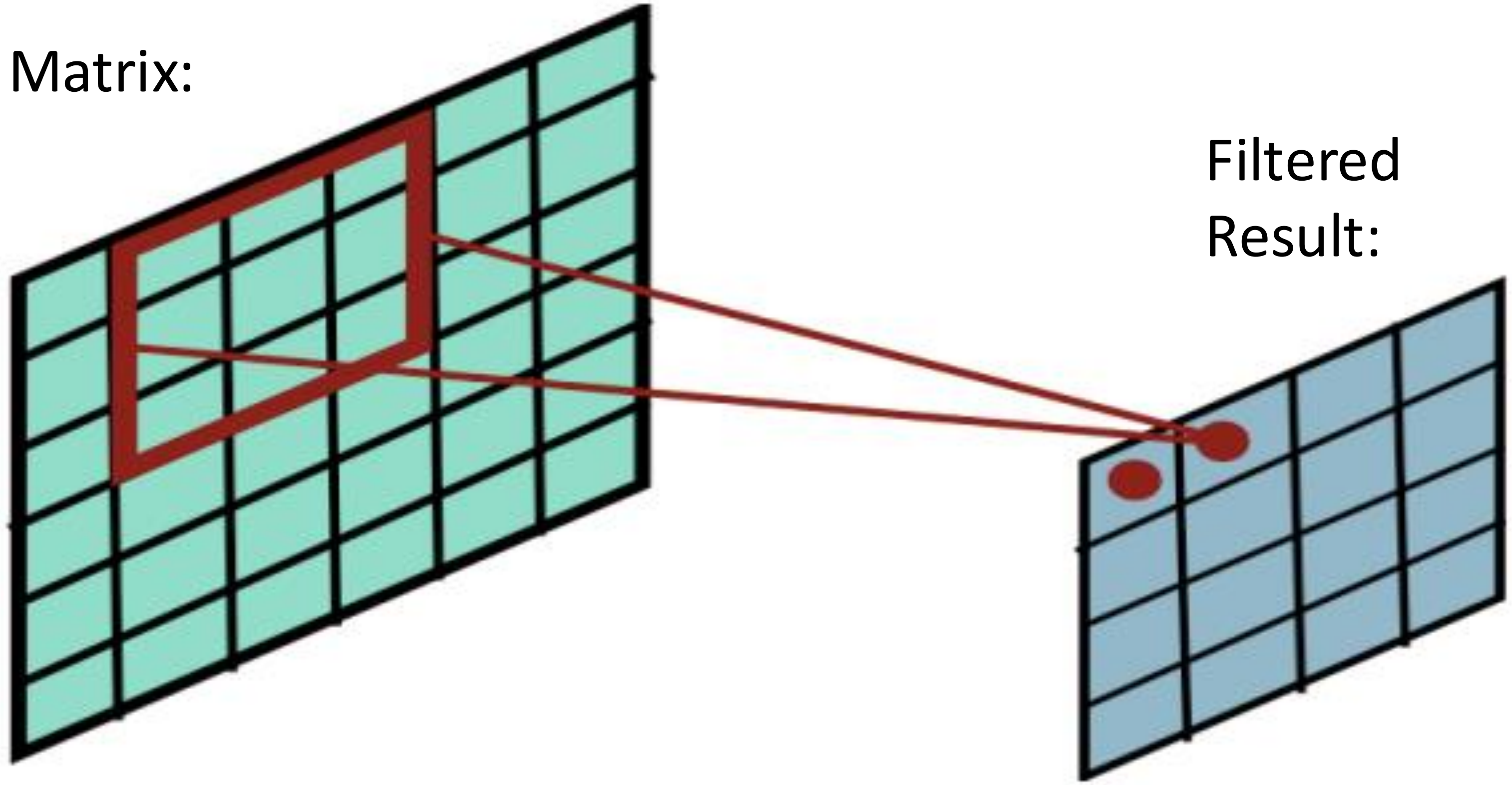
2D Filtering



Slides filter over the matrix and computes matrix multiplication

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

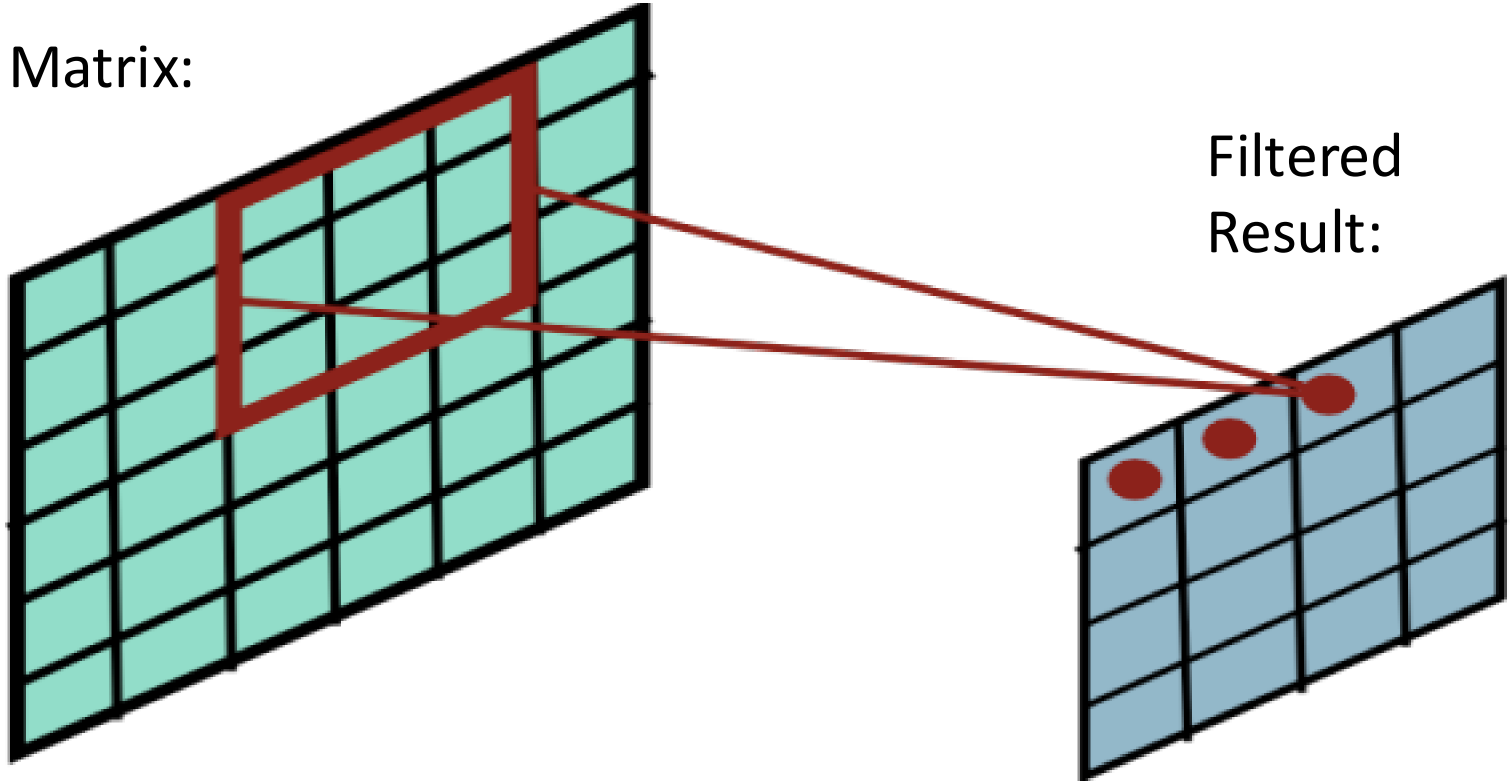
2D Filtering



Slides filter over the matrix and computes matrix multiplication

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

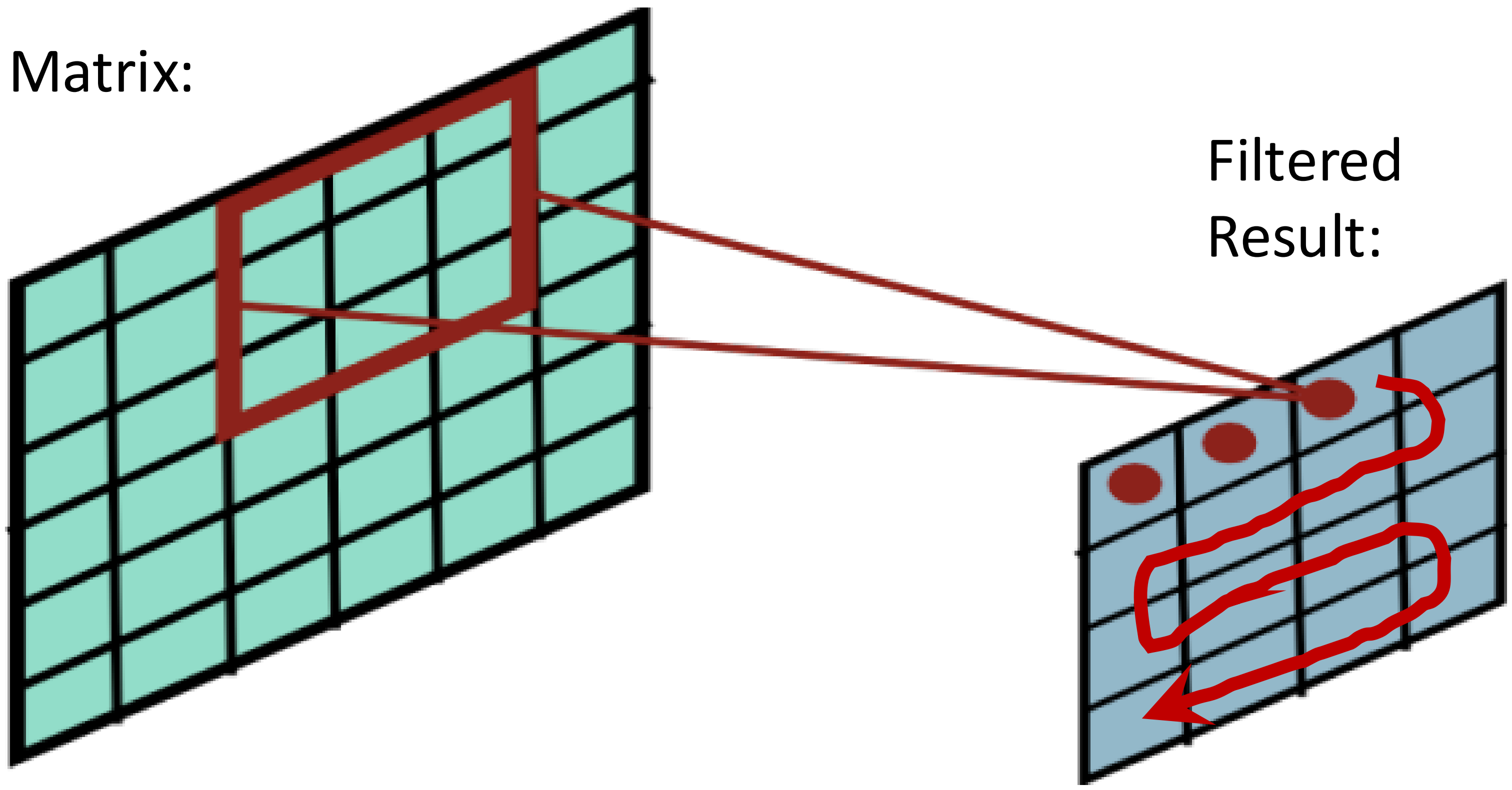
2D Filtering



Slides filter over the matrix and computes matrix multiplication

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

2D Filtering



Slides filter over the matrix and computes matrix multiplication

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

?	?	?
?	?	?
?	?	?

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

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

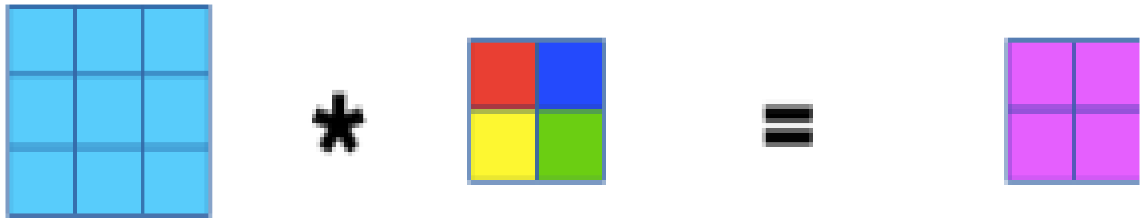
1	0	1
0	1	0
1	0	1

Feature Map

4	3	4
2	4	3
2	3	4

Convolutional Layer

- Many neural network libraries use "convolution" interchangeably with "cross correlation"; these are technically different
- Examples in these slides show the "cross-correlation" function

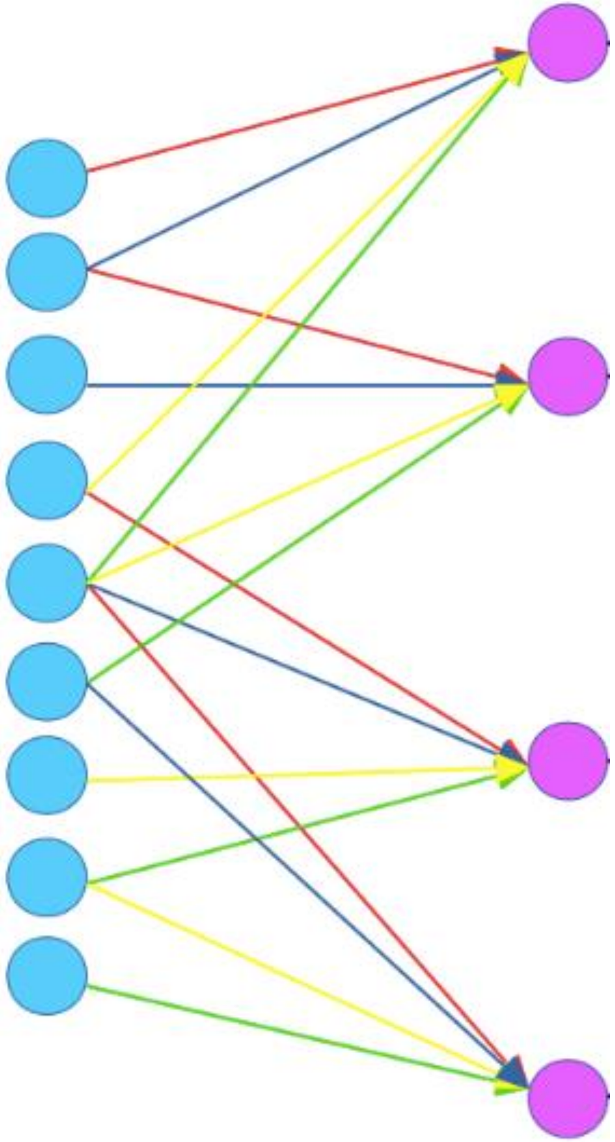


Input

Filter
(aka – Kernel)

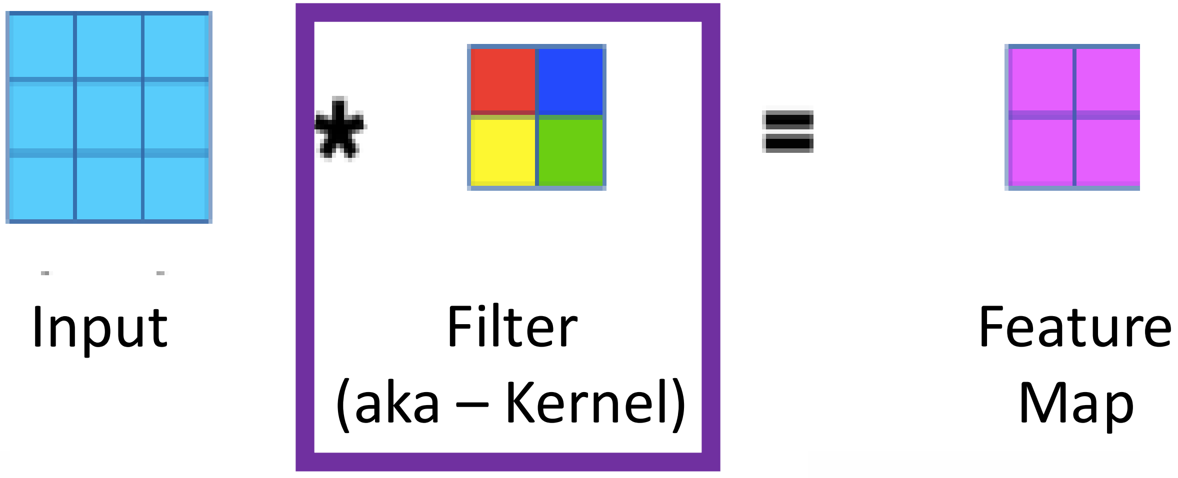
Feature
Map

Way to Interpret
Neural Network

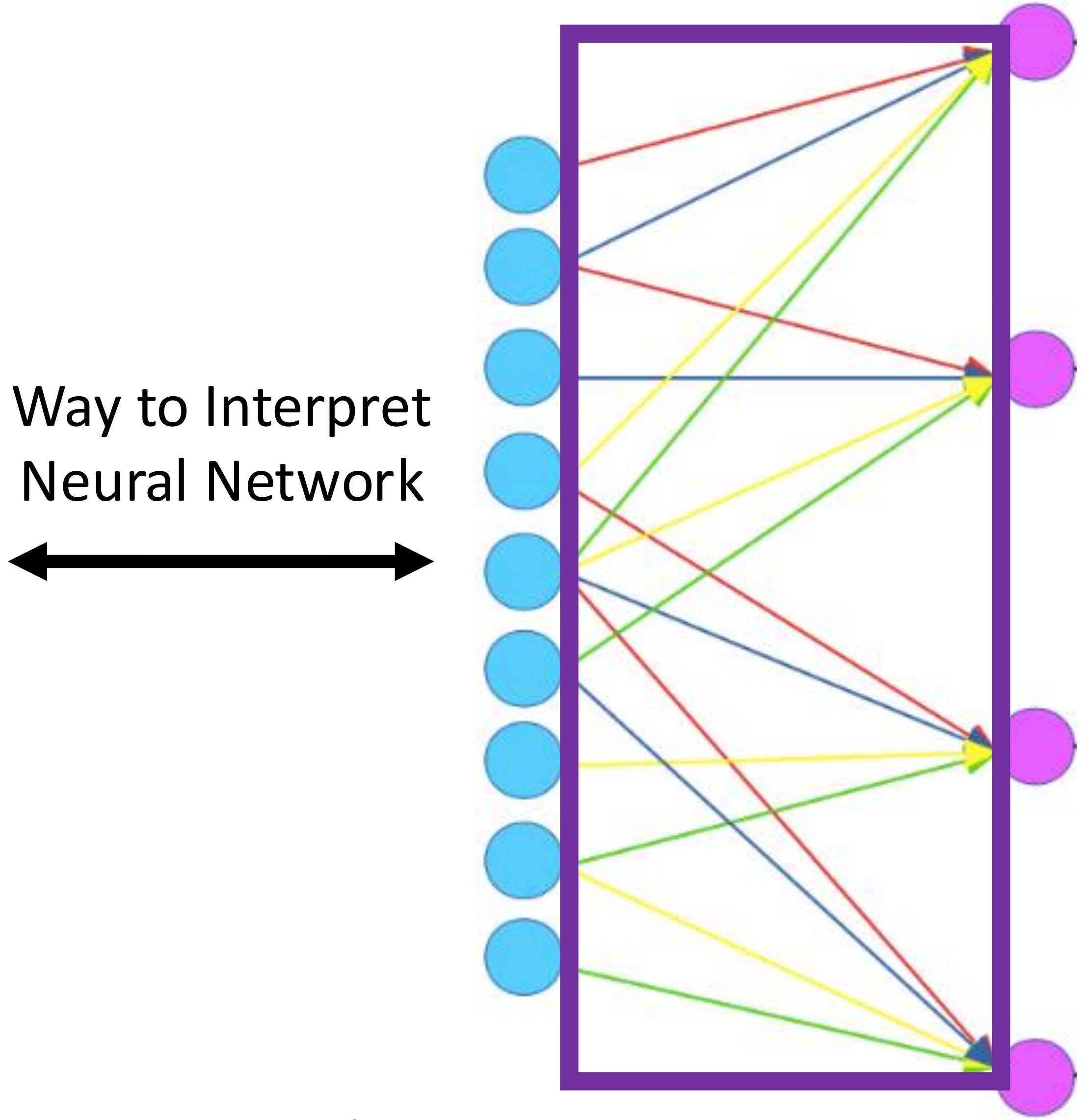


https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Convolutional Layer: Parameters to Learn



Way to Interpret Neural Network

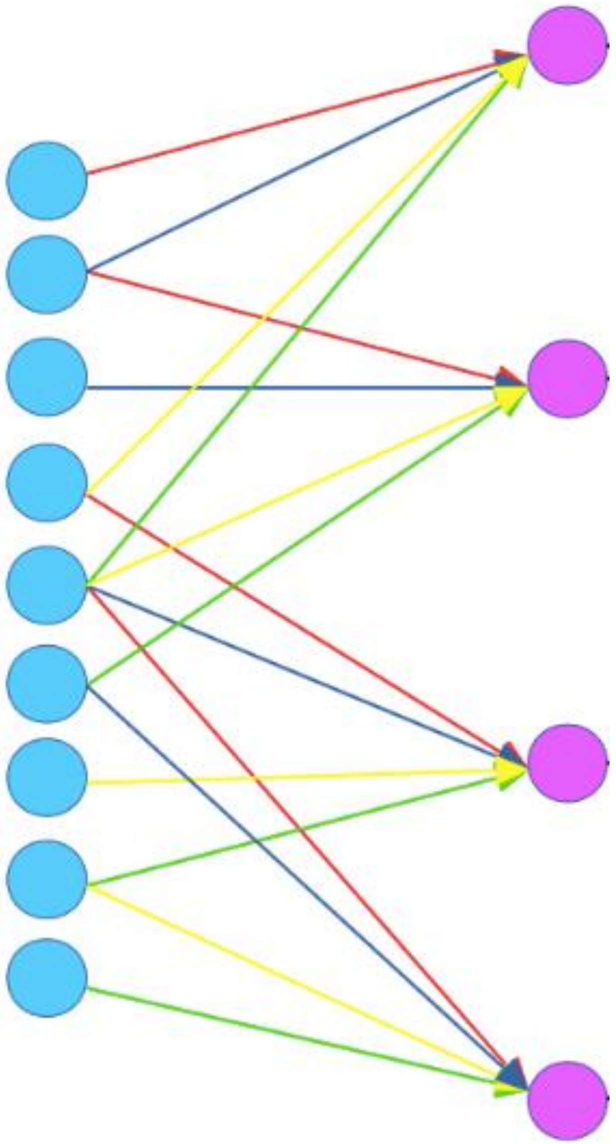


https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

Convolutional Layer: Parameters to Learn

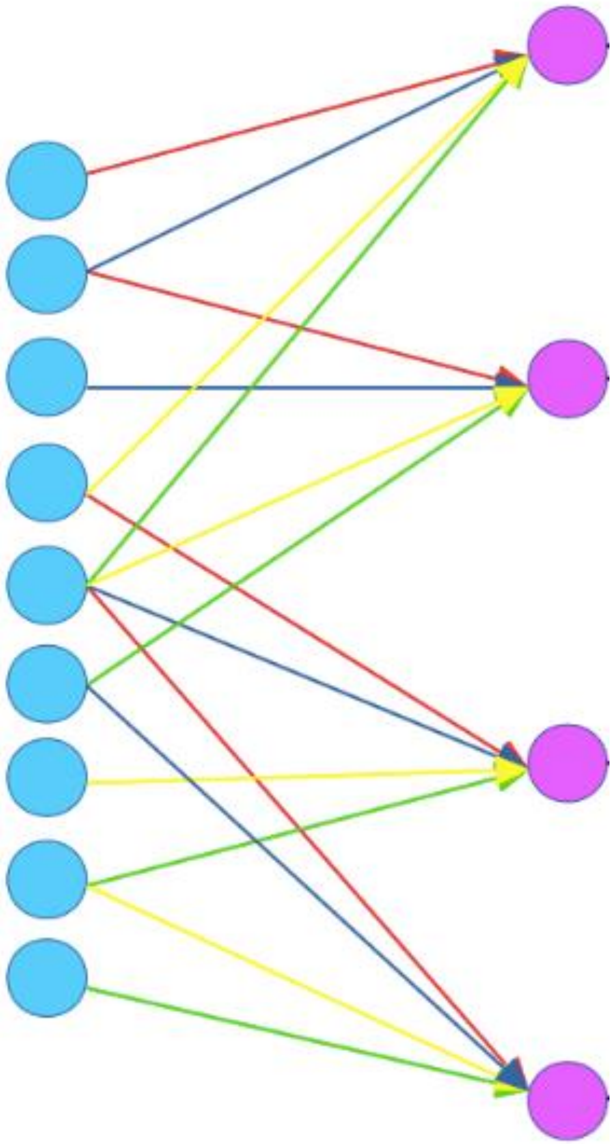
- For shown example, how many weights must be learned?
  - 4 (red, blue, yellow, and green values)

- 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/

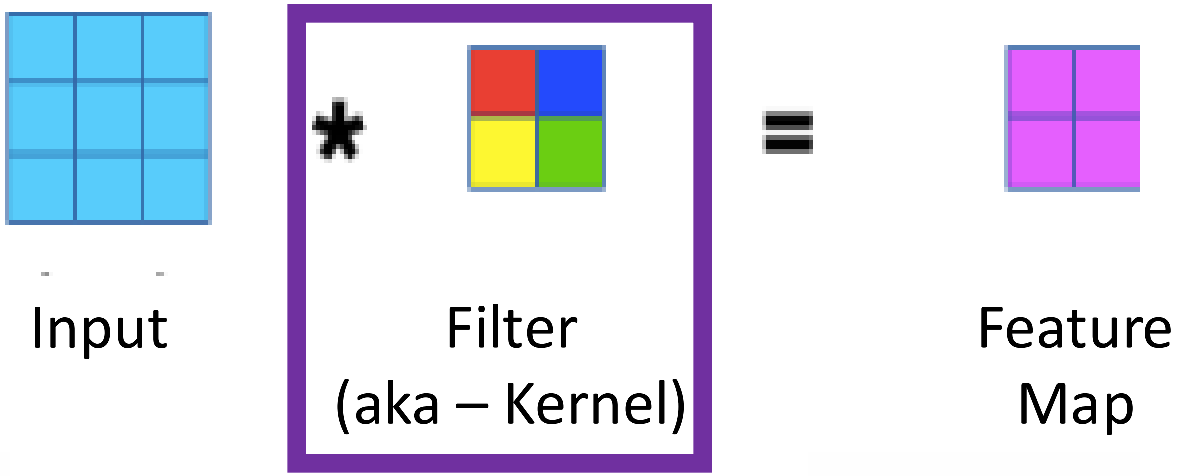
Convolutional Layer: Parameters to Learn



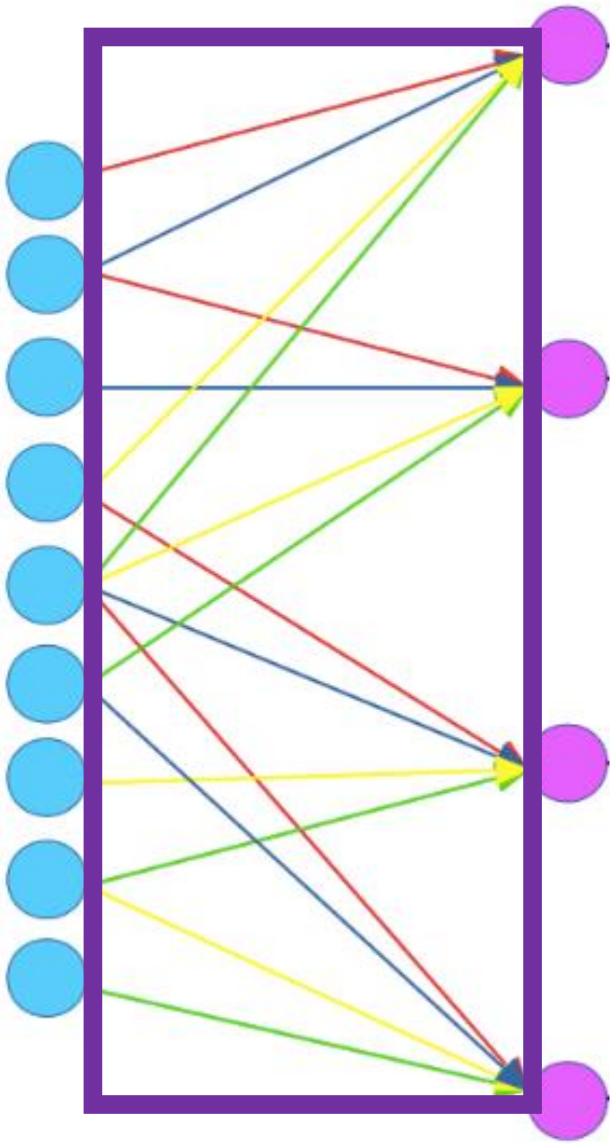
Neocognitron hard-coded filter values...
filter values are learned for CNNs

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

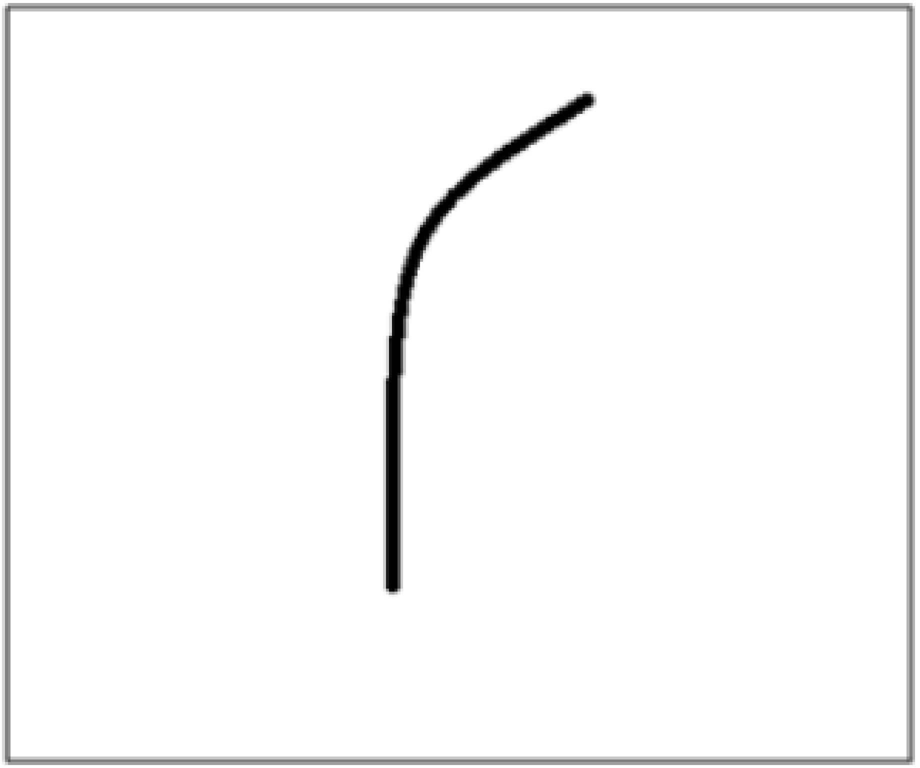
Convolutional Layer: What Can Filters Do?

- 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

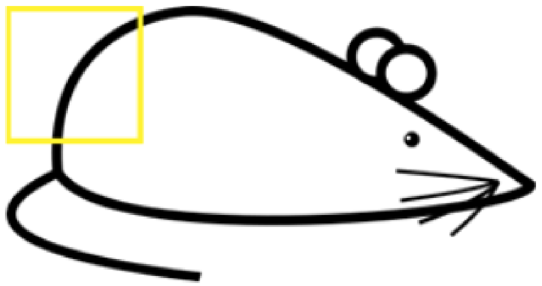


https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

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!!)**

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!!)**

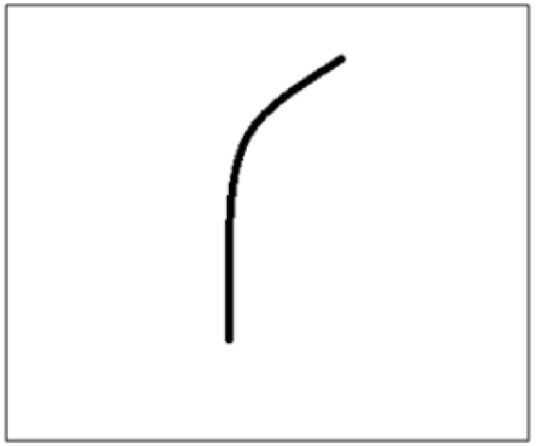
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Convolutional Layer: What Can Filters Do?

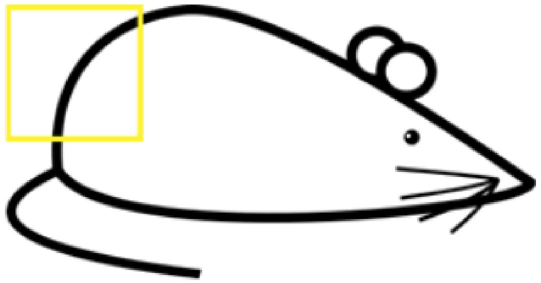
This Filter is a Curve Detector!

- e.g.,

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!)

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

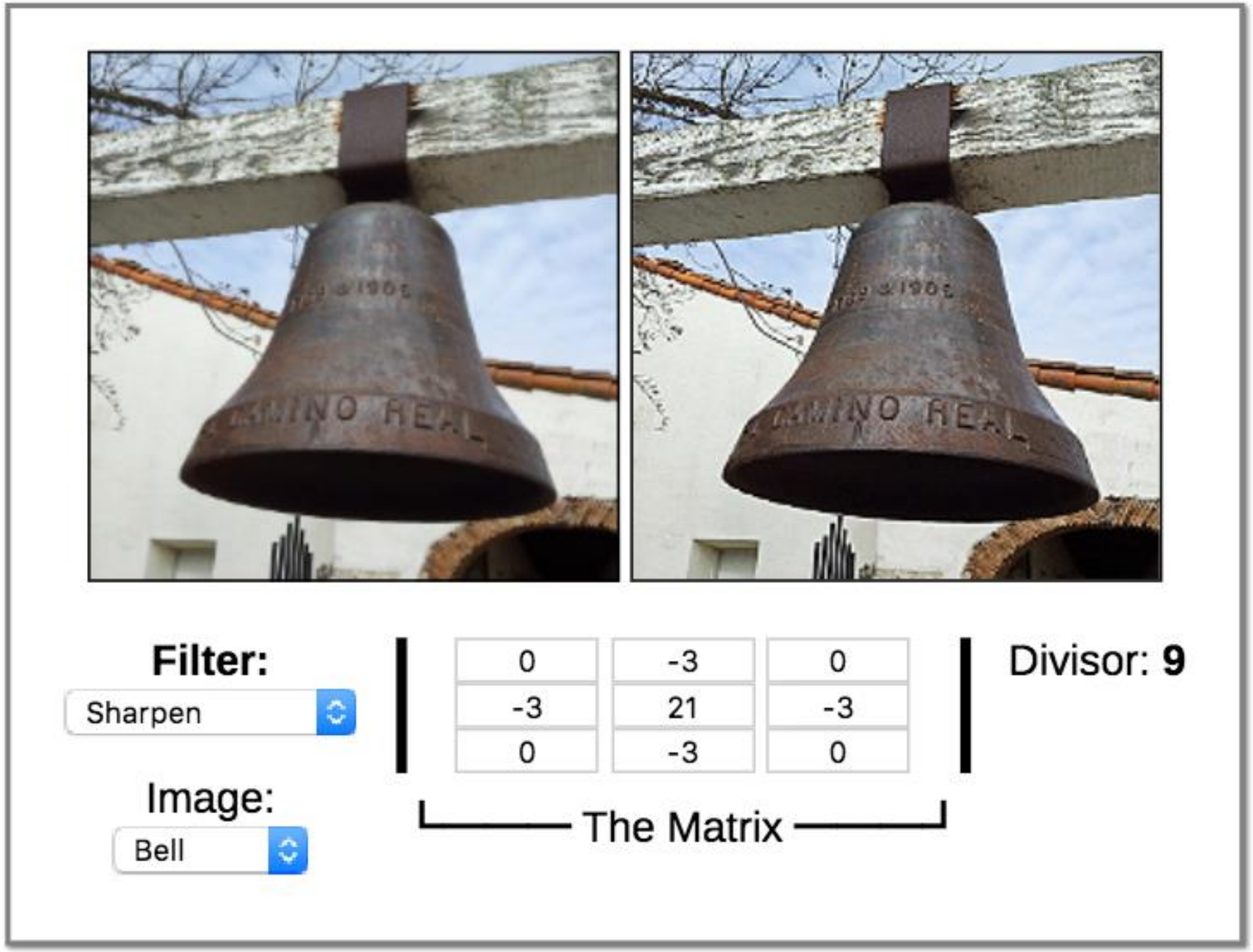
Convolutional Layer: What Can Filters Do?

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

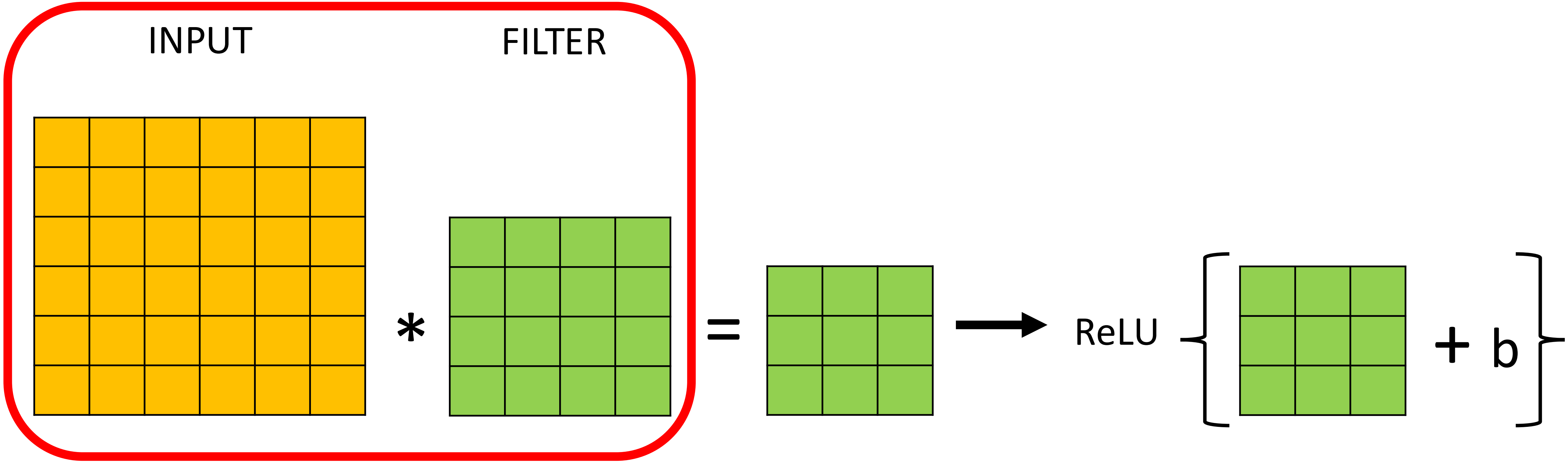
https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/

Convolutional Layer: What Can Filters Do?



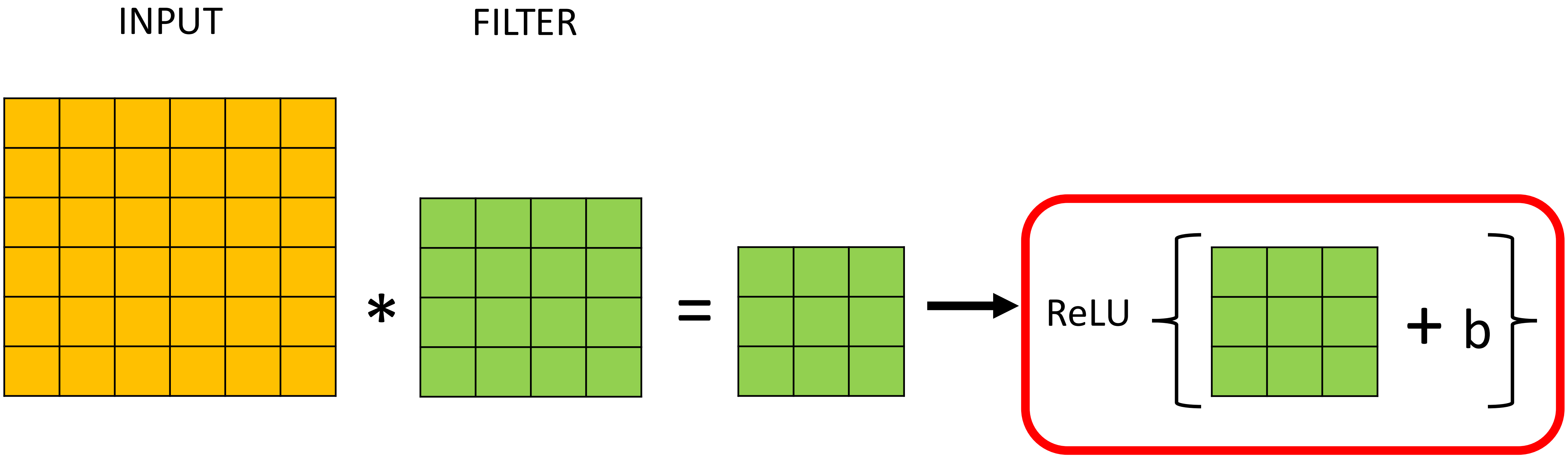
Demo: http://beej.us/blog/data/convolution-image-processing/

Key Ingredient 1: Convolutional Layers



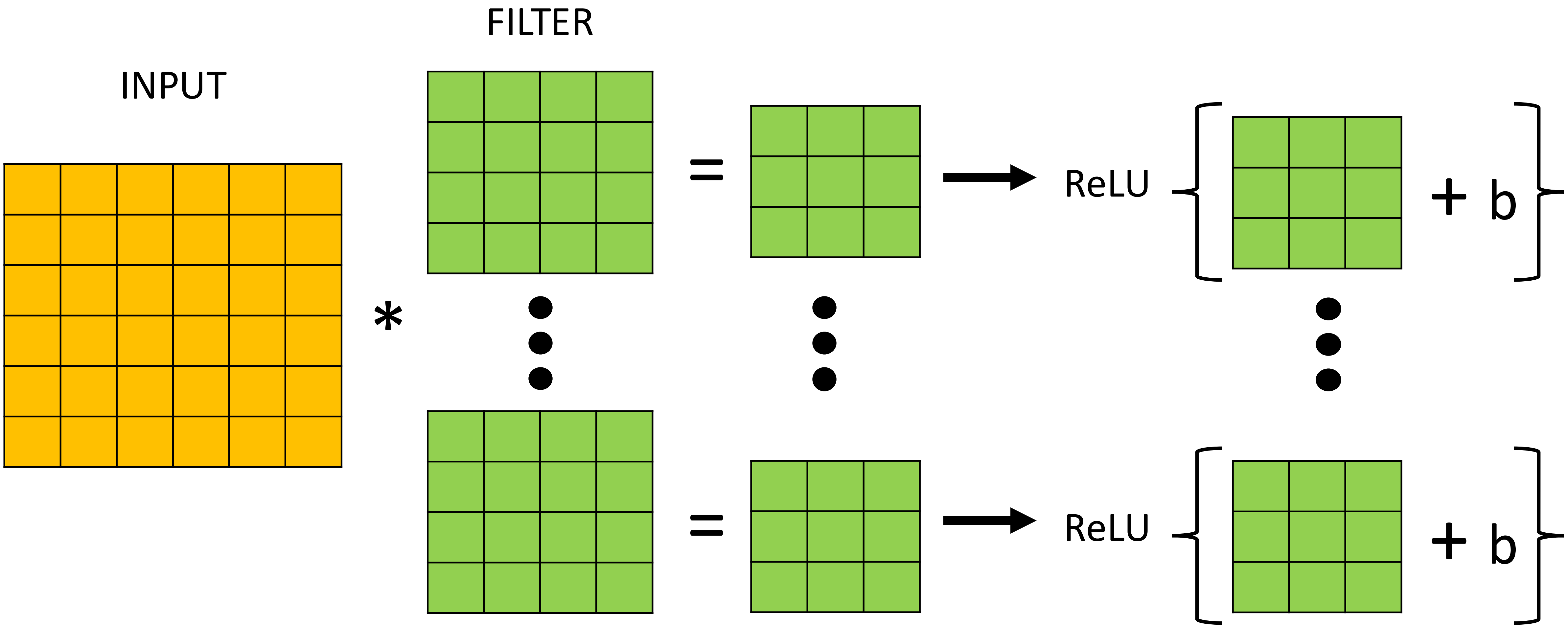
Can choose filters of any size to support feature learning!

Key Ingredient 1: Convolutional Layers



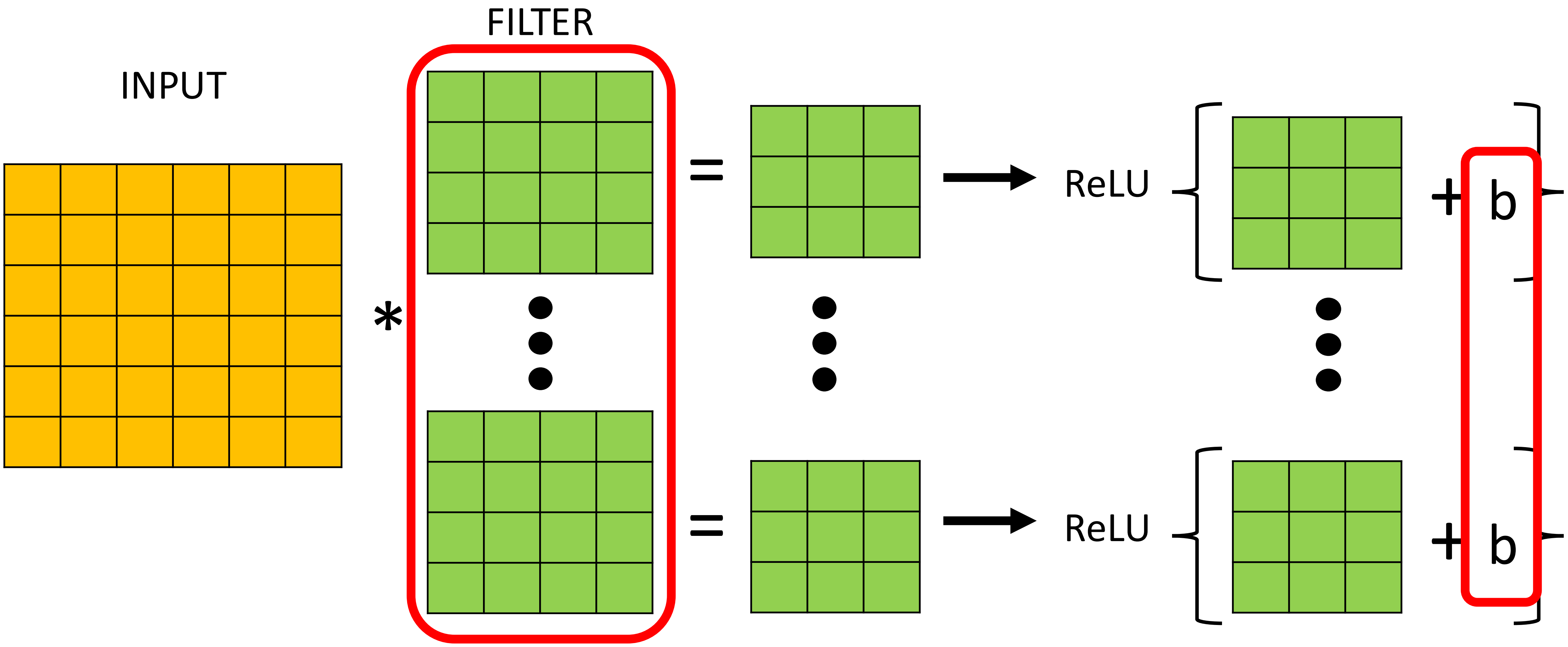
Filtered results are passed, with a bias term, through an activation function to create **activation/feature maps**

Key Ingredient 1: Convolutional Layers



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

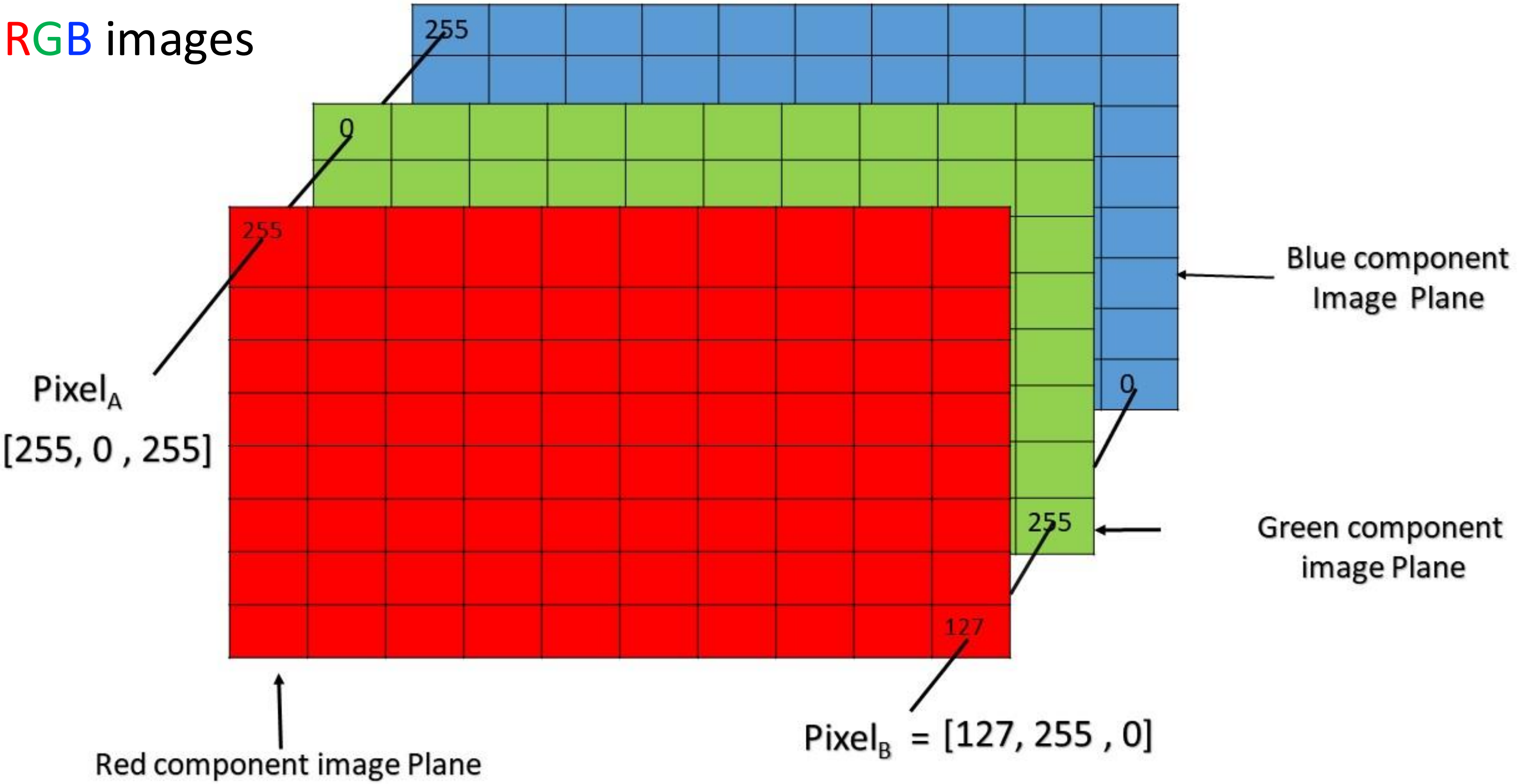
Key Ingredient 1: Convolutional Layer Summary



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

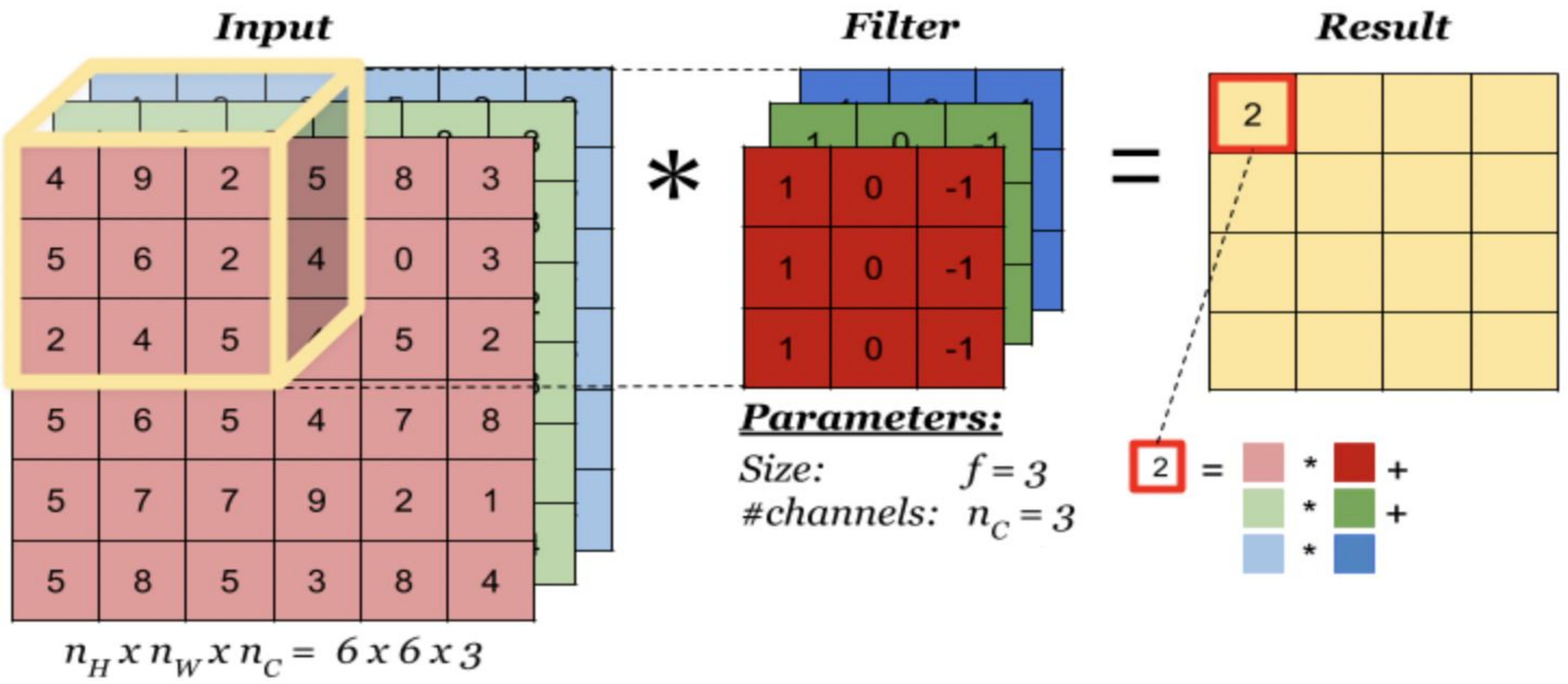
How Filters Are Applied to Multi-Channel Inputs

e.g., RGB images



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

How Filters Are Applied to Multi-Channel Inputs

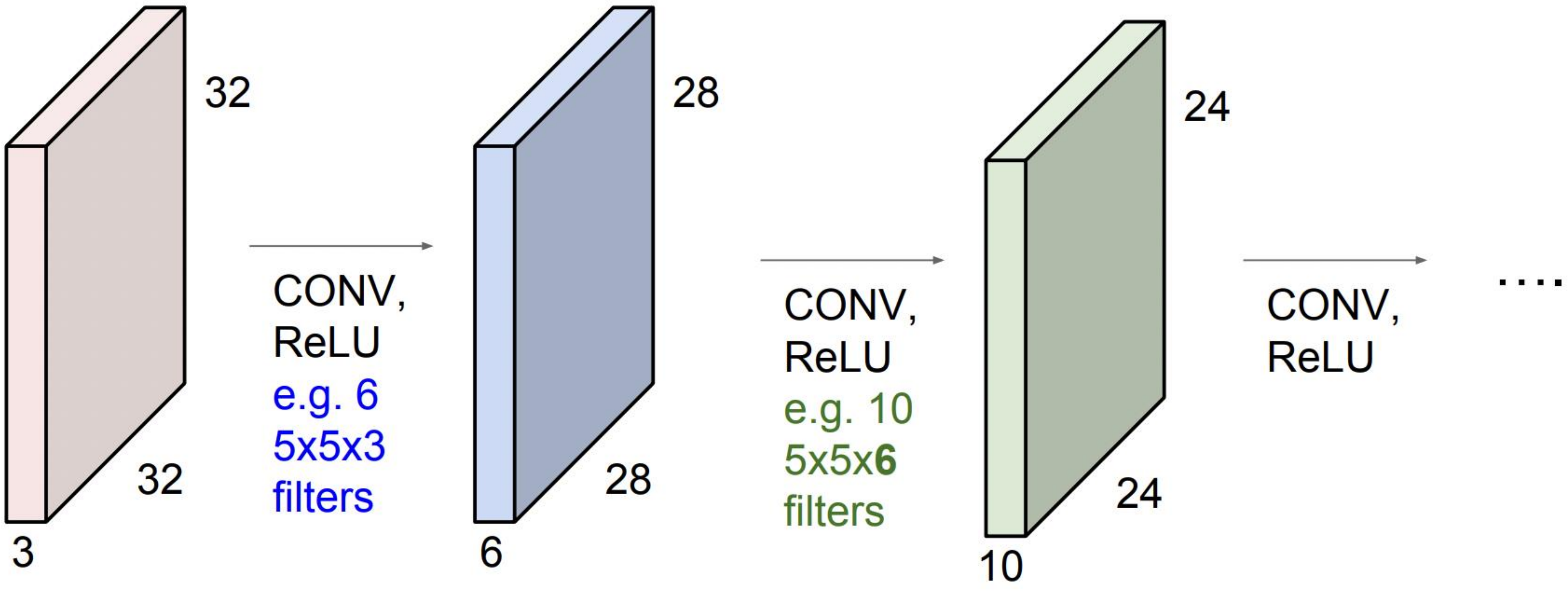


Number of channels in a filter matches that of the input

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

Convolutional Layers Stacked

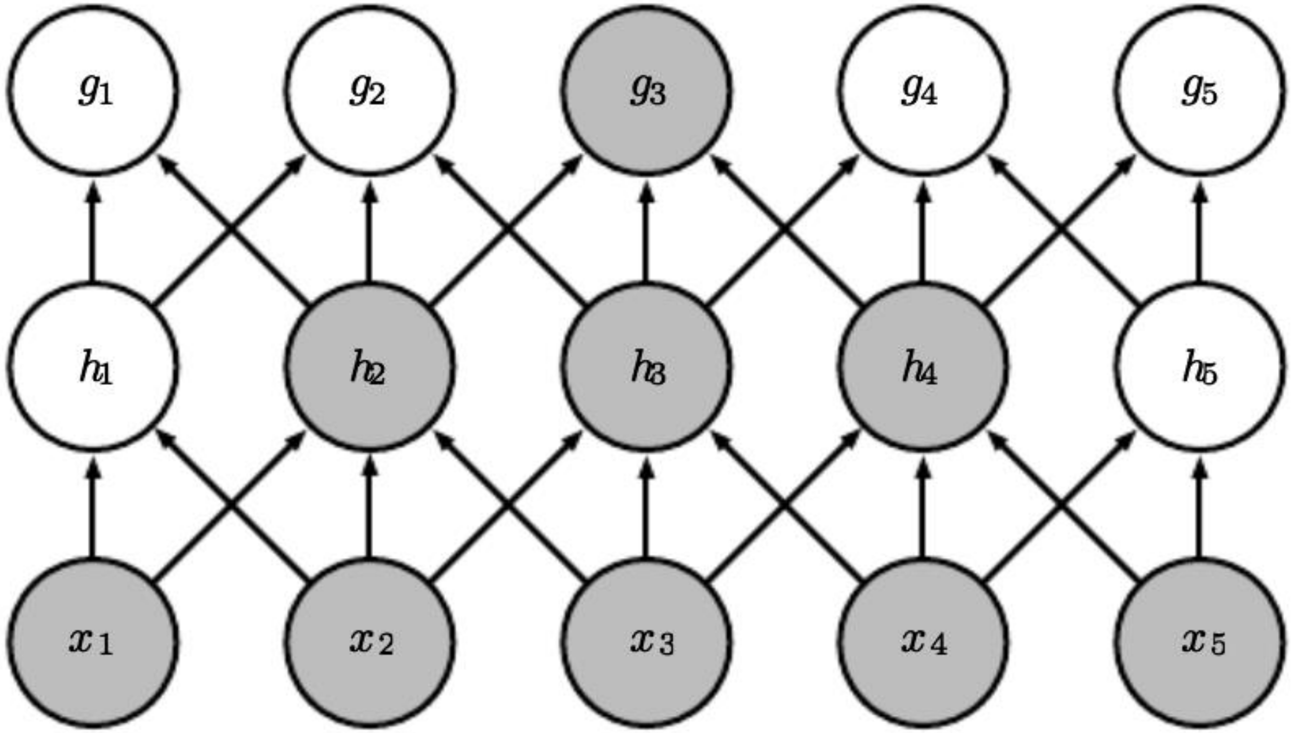
Can then stack a sequence of convolution layers; e.g.,



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:**

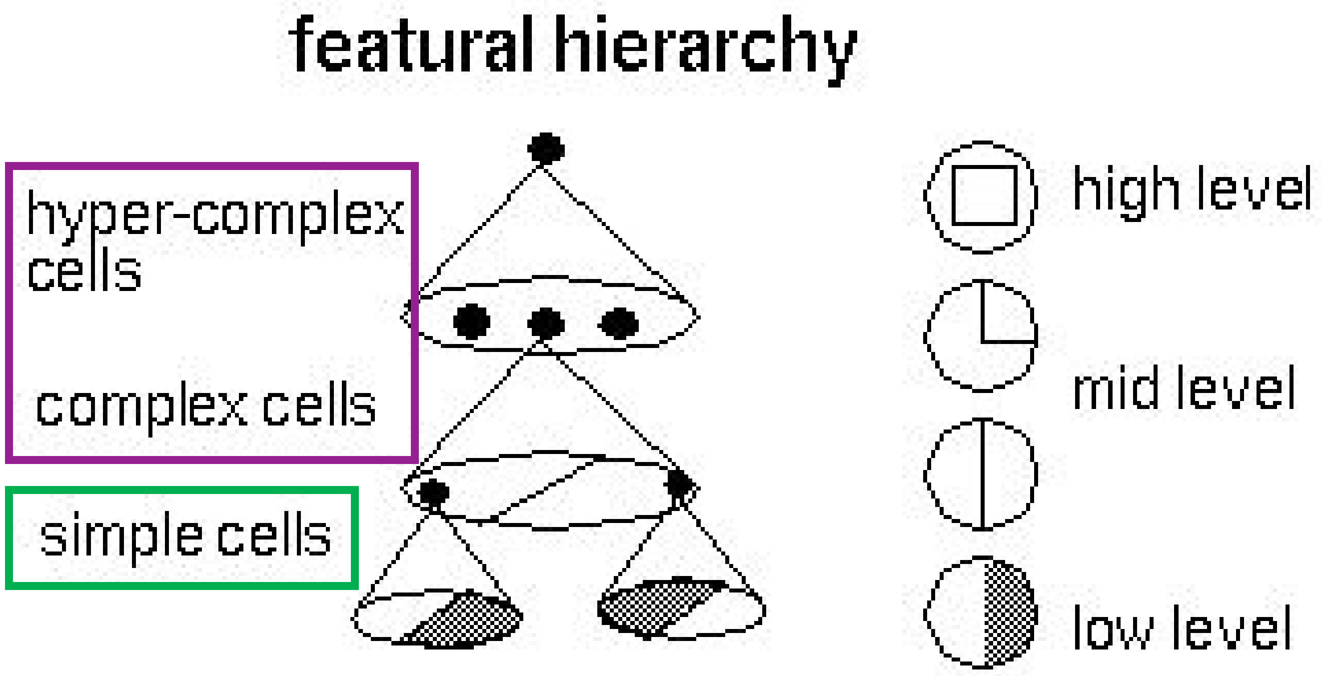


https://www.deeplearningbook.org/contents/convnets.html

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**:

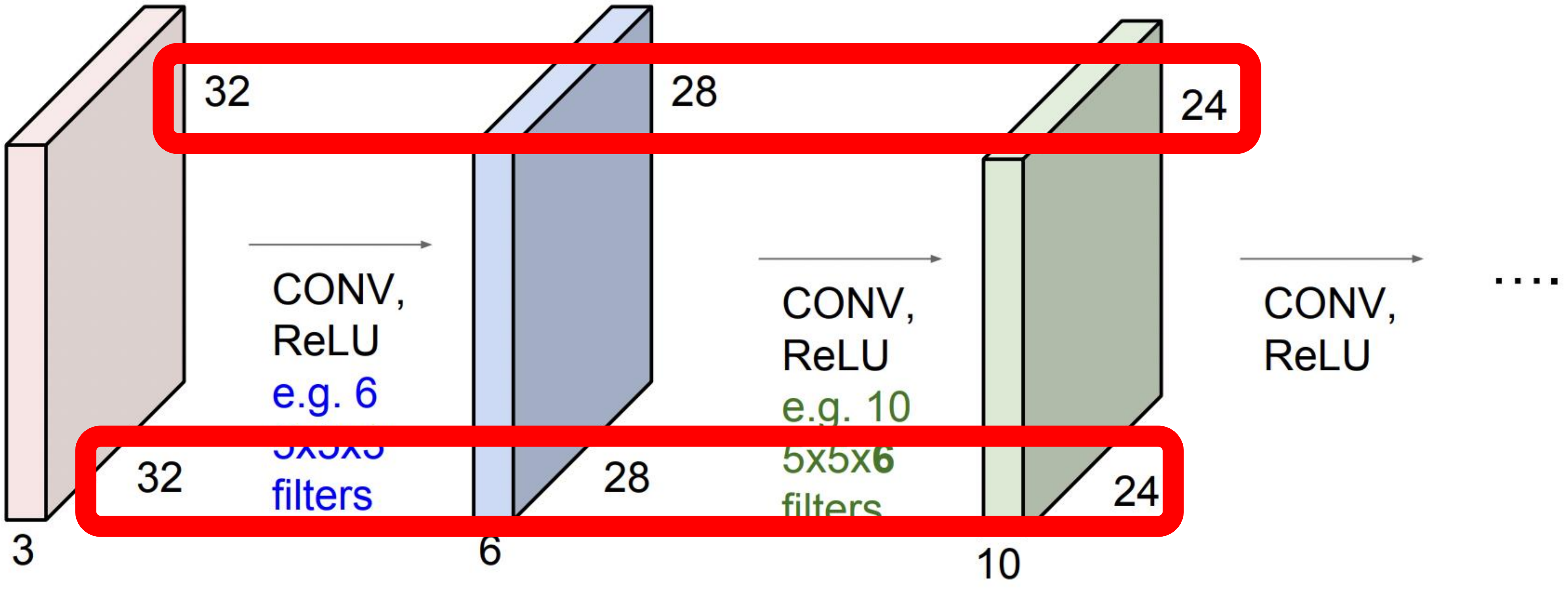
Higher level features are constructed by combining lower level features



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

Problem #1: Input Shrinks

Why do the dimensions shrink with each convolutional layer?

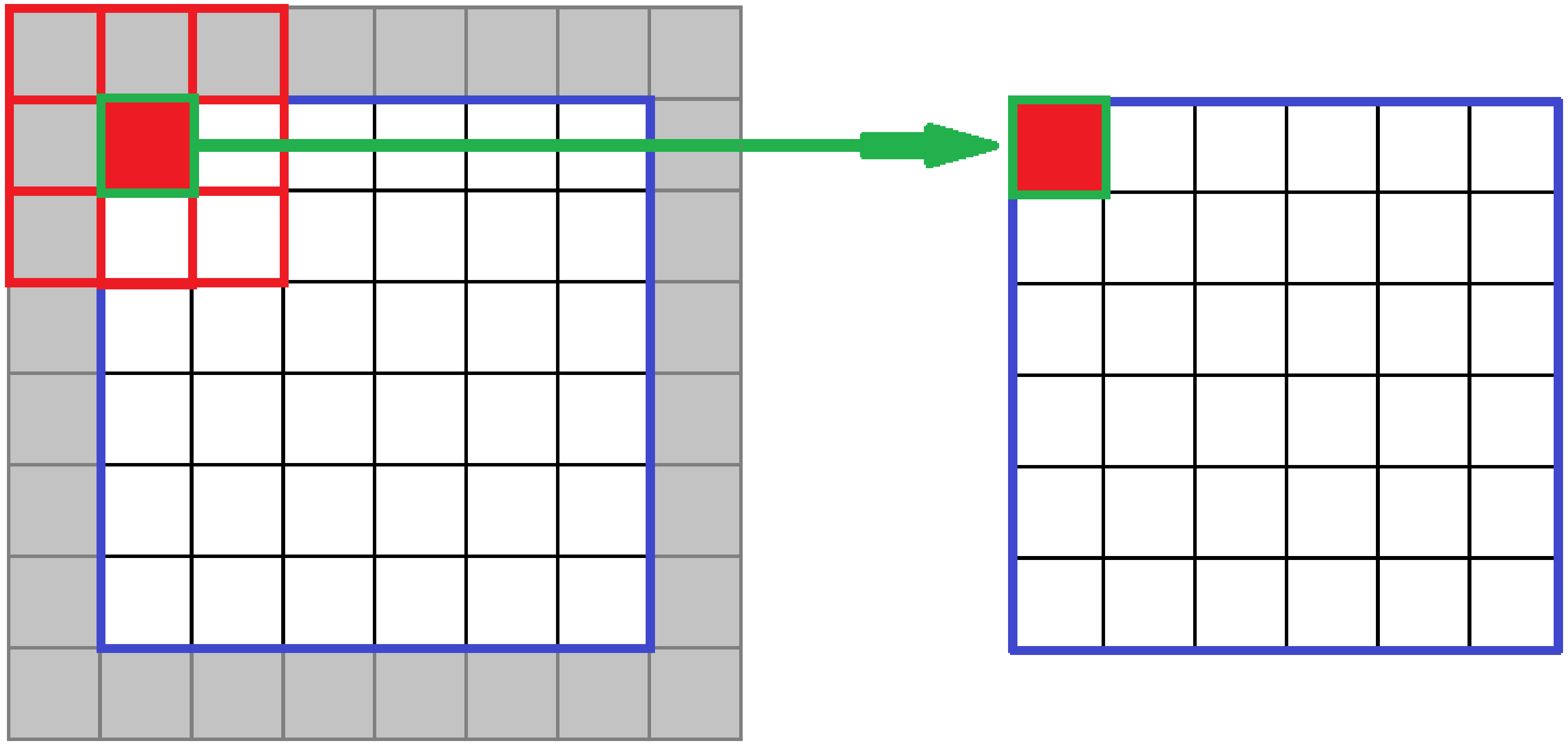


Information is lost around boundary of the input!

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

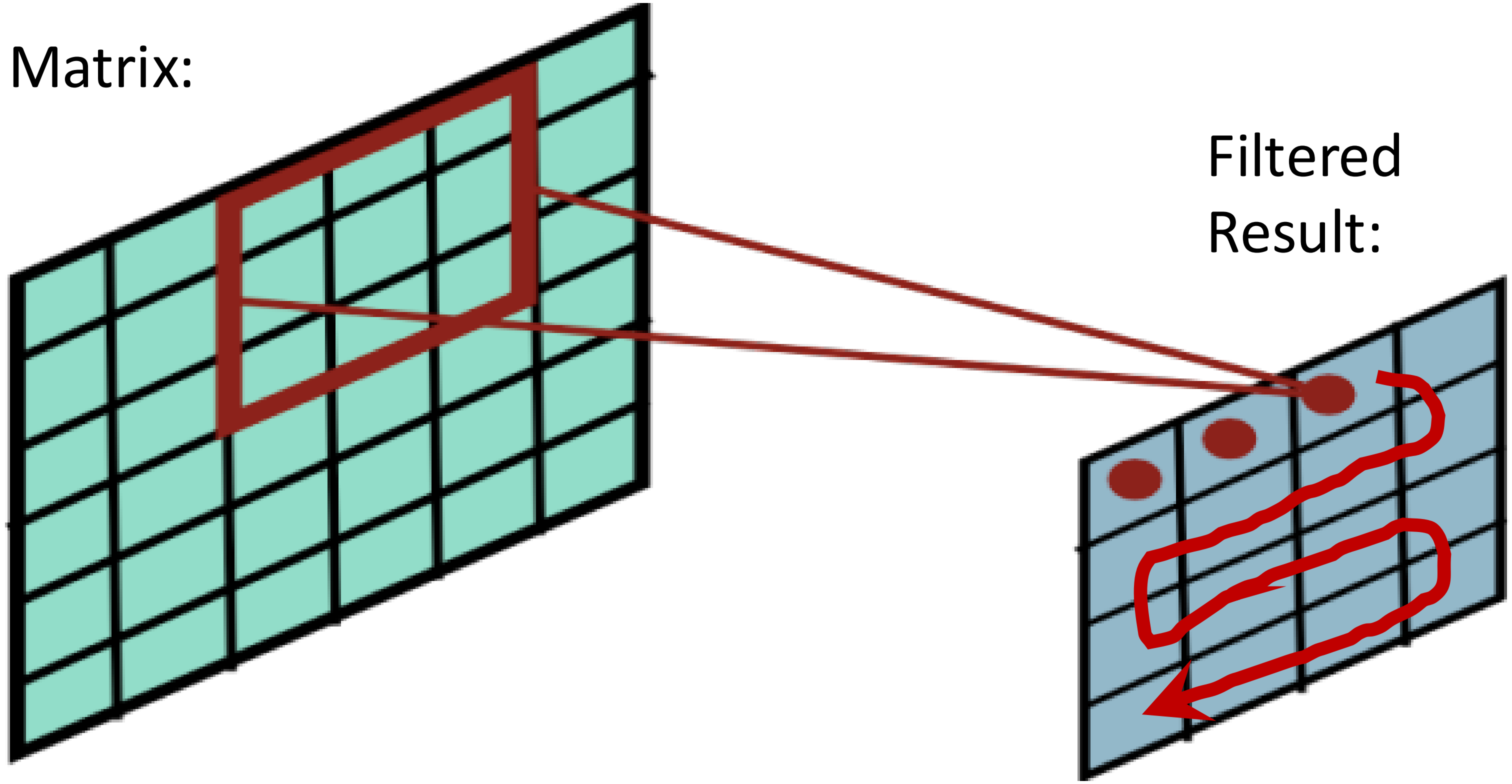
Solution: Control Output Size with **Padding**

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



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 multiplications

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 Layer Summary

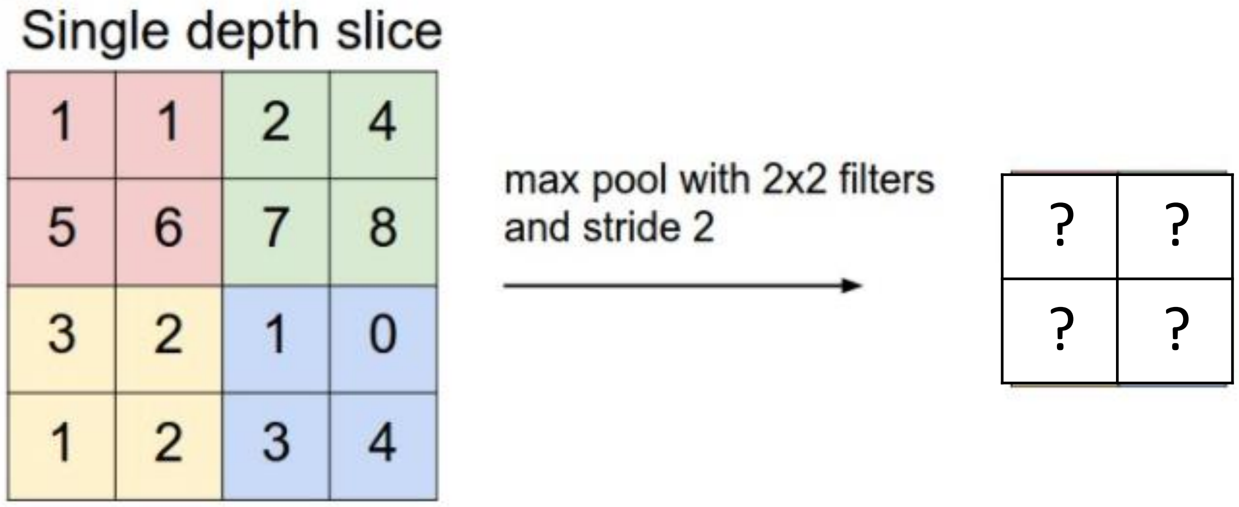
- Hyperparameters:
  - Number of convolutional layers
  - For each layer, number of filters and their dimensions, padding type, & stride
- Model will learn values for:
  - Weights
  - Biases

Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Pioneering CNN model: LeNet

Pooling Layer: Summarizes Neighborhood

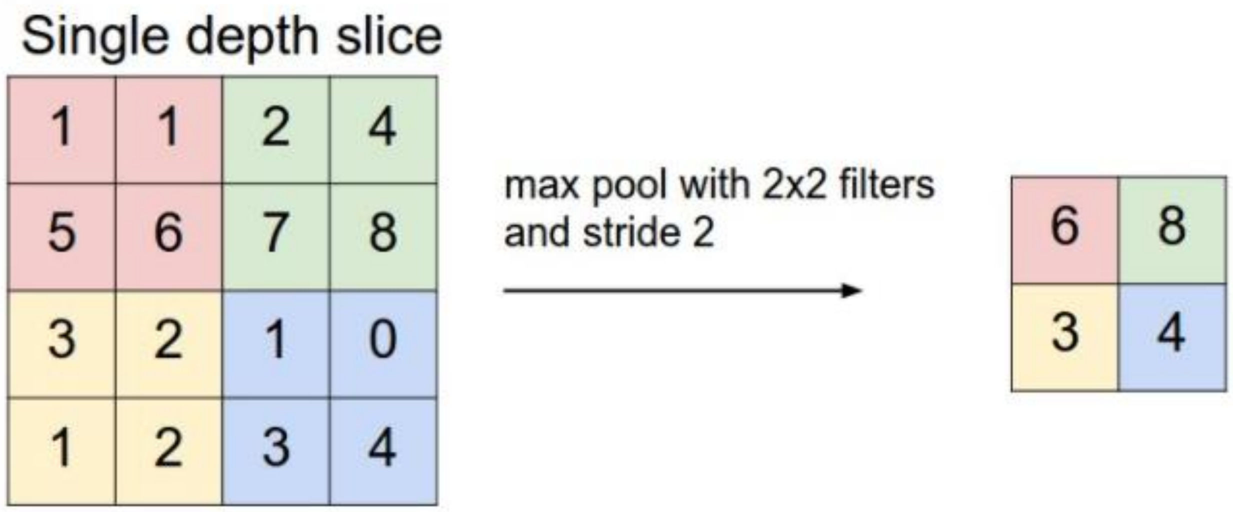
- **Max-pooling***:* partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk



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

Pooling Layer: Summarizes Neighborhood

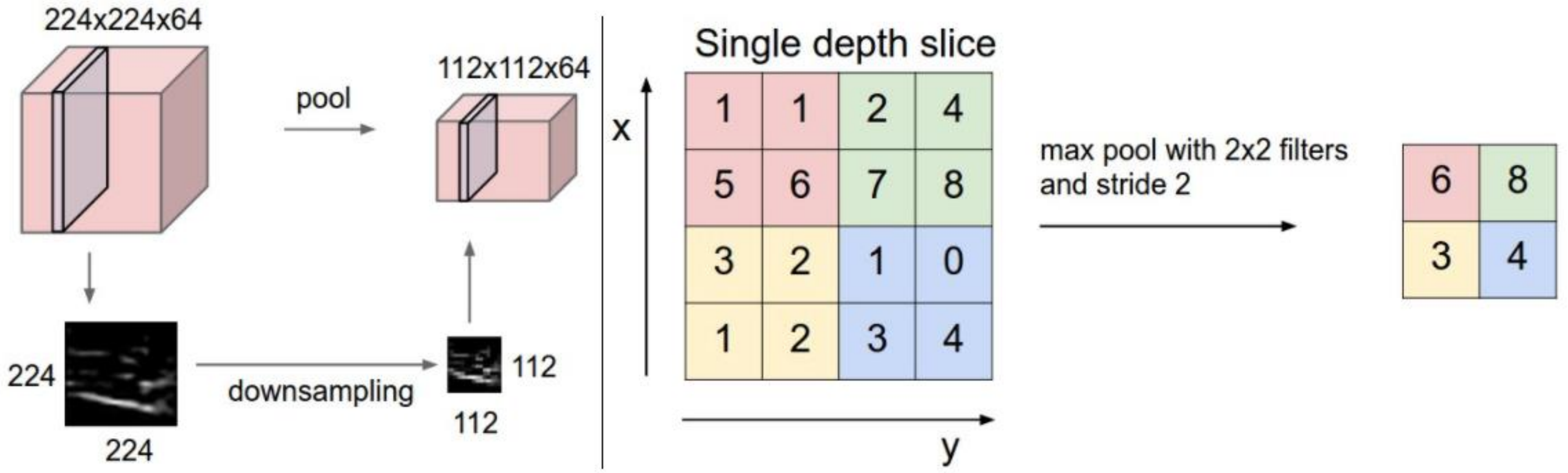
- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk



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

Pooling Layer: Summarizes Neighborhood

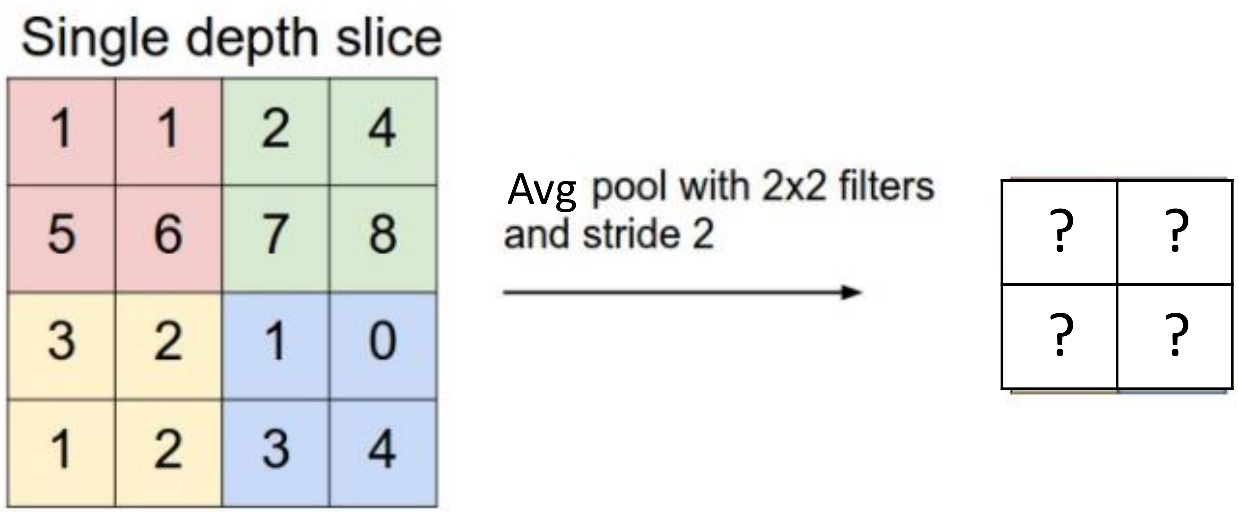
- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk



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

Pooling Layer: Summarizes Neighborhood

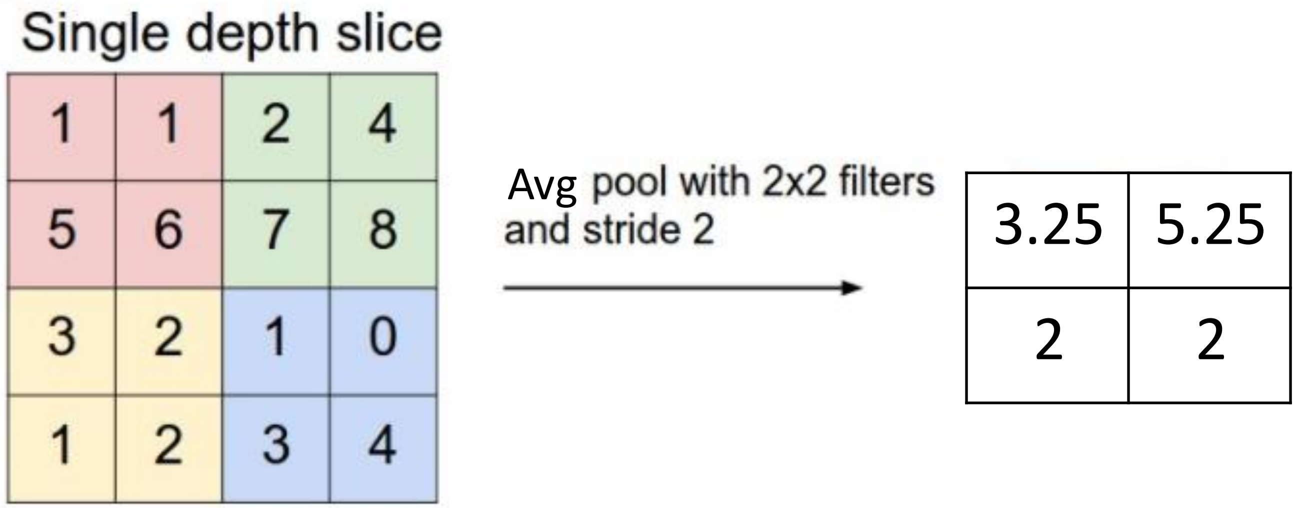
- **Max-pooling**: partitions input into a set of non-overlapping rectangles and outputs the maximum value for each chunk
- **Average-pooling**: partitions input into a set of non-overlapping rectangles and outputs the average value for each chunk



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



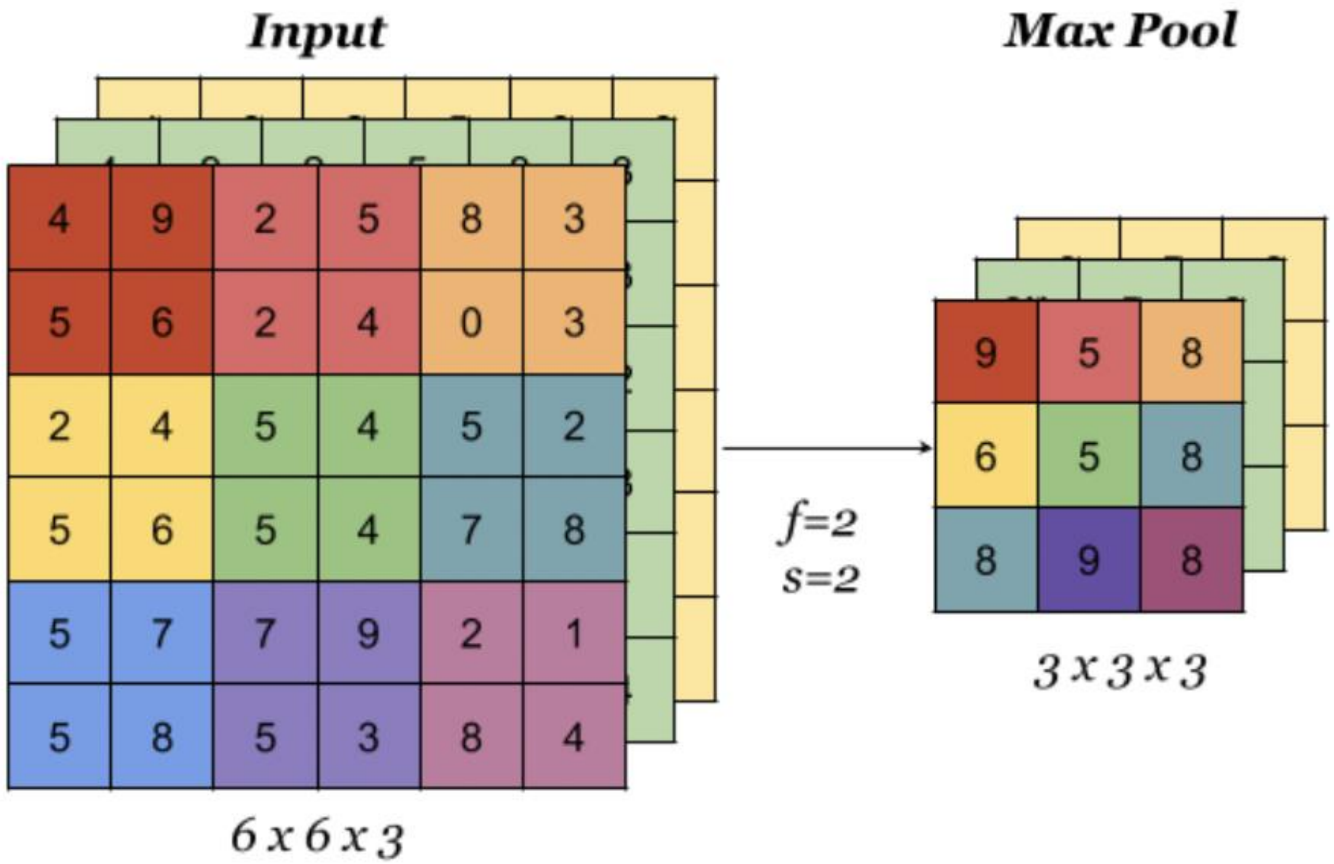
http://cs231n.github.io/convolutional-networks/#pool

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/nn.html#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/

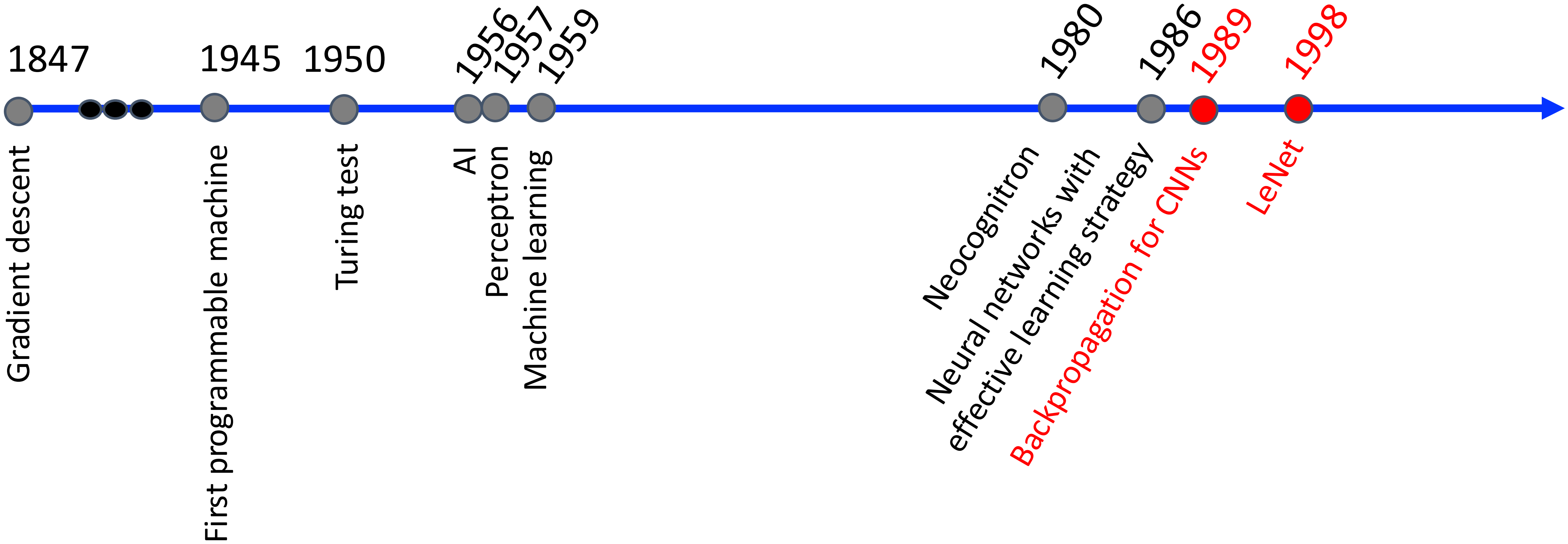
Pooling Layer: Benefits

- Reduces memory requirements
- Reduces computational requirements

Today’s Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Pioneering CNN model: LeNet

Historical Context: Inspiration

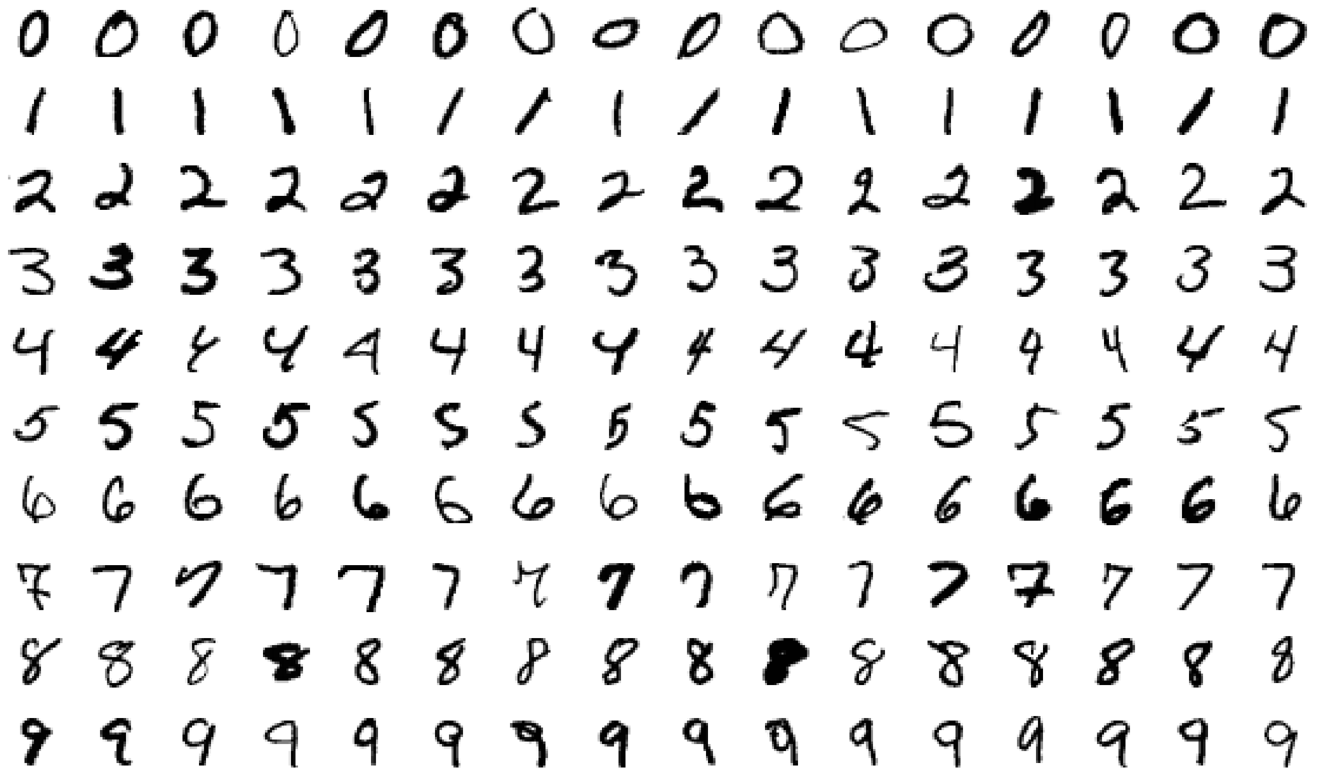


https://yann.lecun.com/exdb/lenet/index.html

By end of 1990s, LeNet read over 10% of checks in North America with millions every month

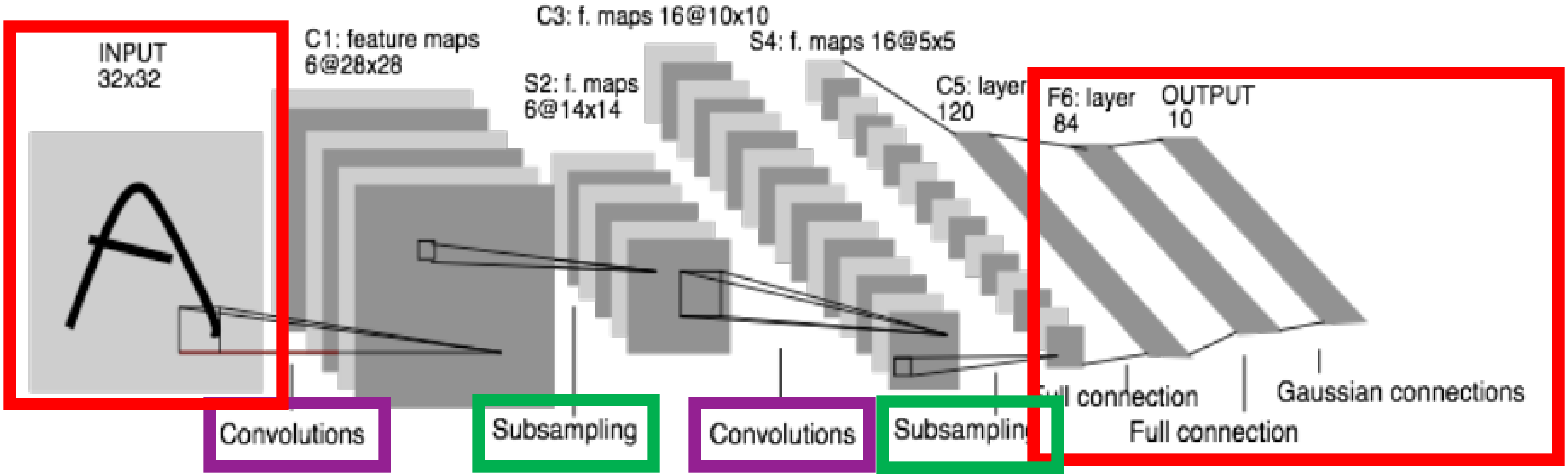
MNIST Dataset Challenge

- Goal: classify digit as 0, 1, …, or 9
- Source: images collected by NIST from a total of 500 Census Bureau employees and high school students
- Dataset: 60,000 training and 10,000 test examples, pre-processed to be centered and same dimension; writers were different in the two sets
- Evaluation metric: accuracy (% correct)



https://commons.wikimedia.org/w/index.php?curid=64810040

LeNet: Architecture (like Neocognitron, has alternating convolutional layers and pooling layers)

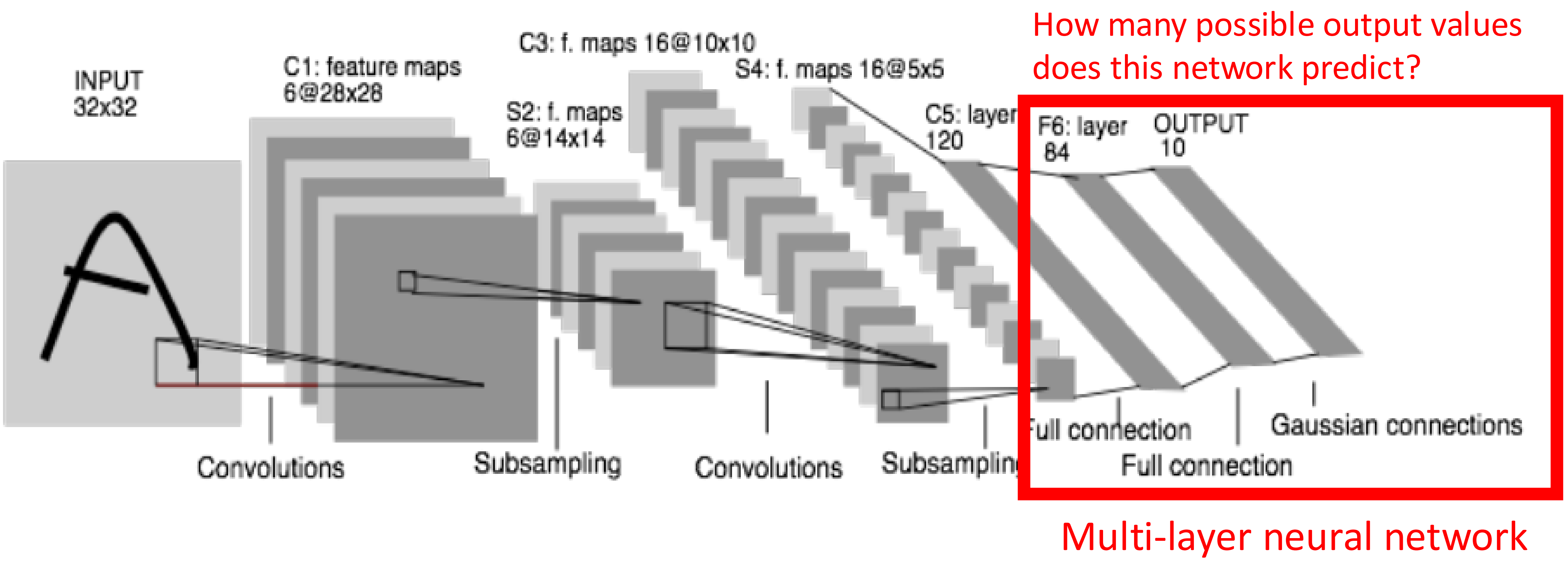


tanh is used as the activation function

Multi-layer neural network

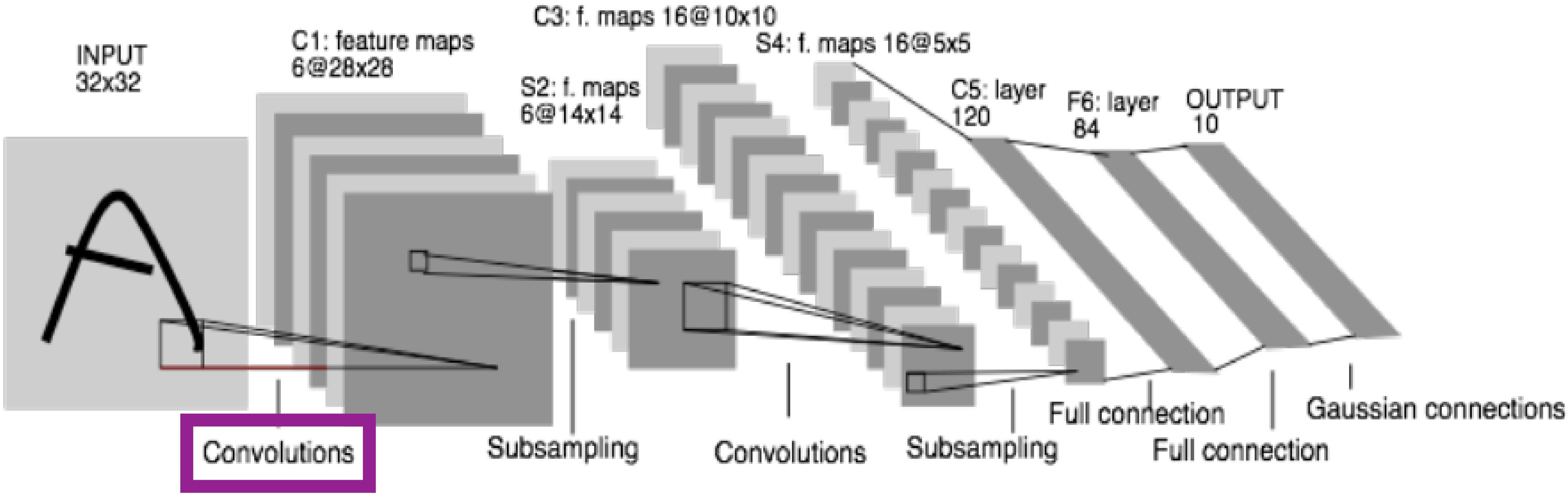
Lecun, Bottou, Bengio, and Haffner. Gradient-based learning applied to document recognition. 1998

LeNet: Architecture (like Neocognitron, has alternating convolutional layers and pooling layers)



Lecun, Bottou, Bengio, and Haffner. Gradient-based learning applied to document recognition. 1998

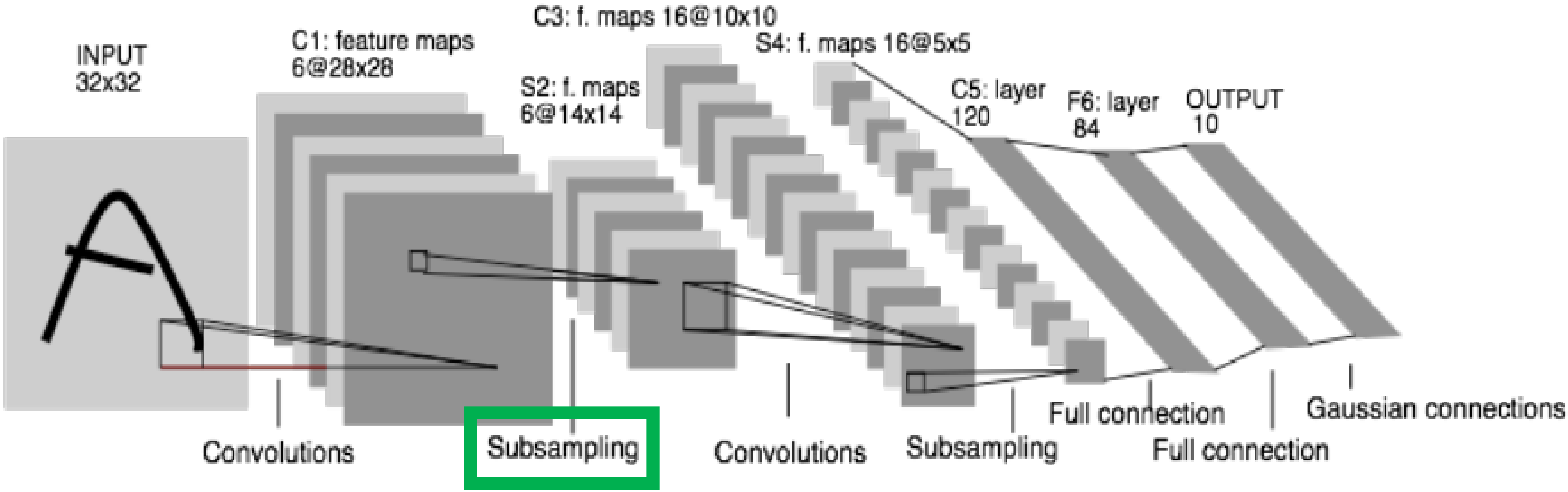
LeNet: Architecture (like Neocognitron, has alternating convolutional layers and pooling layers)



How many filters are between the input and hidden layer 1?

Lecun, Bottou, Bengio, and Haffner. Gradient-based learning applied to document recognition. 1998

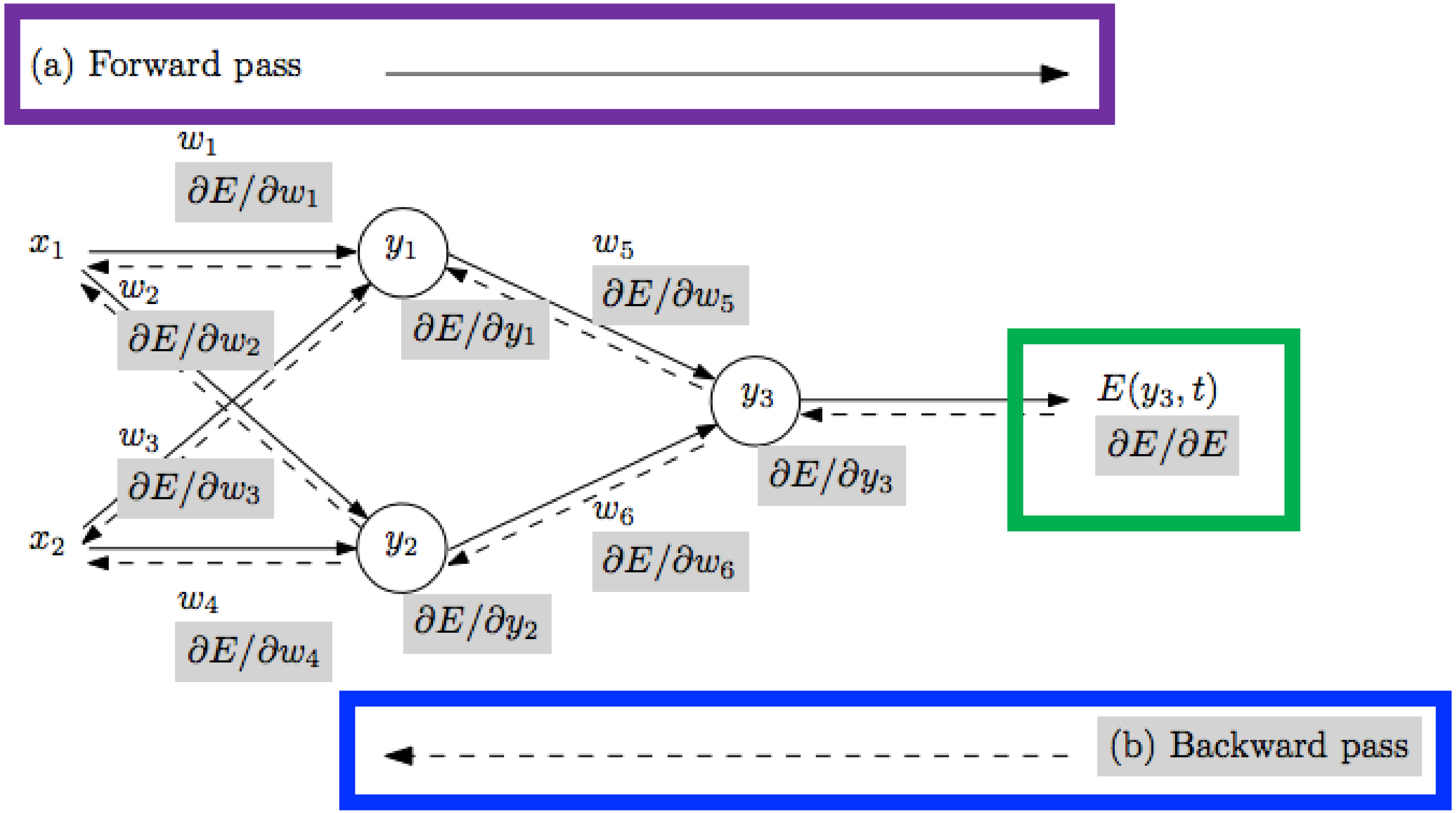
LeNet: Architecture (like Neocognitron, has alternating convolutional layers and pooling layers)



What size of a neighborhood is used for this pooling layer?

Lecun, Bottou, Bengio, and Haffner. Gradient-based learning applied to document recognition. 1998

Training Procedure



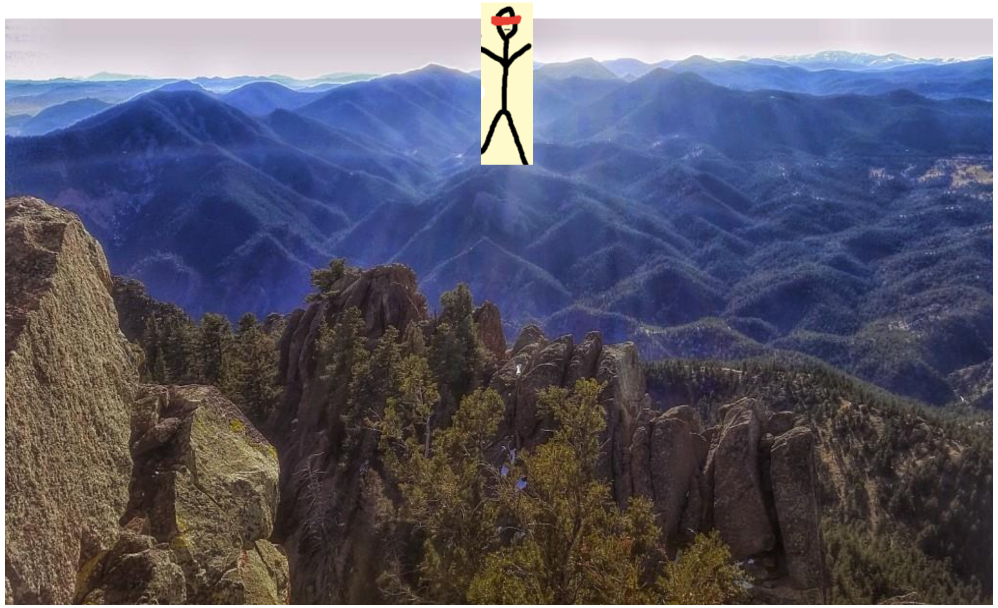
Repeat until stopping criterion met:

1. **Forward pass**: propagate training data through model to make predictions
2. **Error quantification**: measure error of the model's predictions on training data using a loss function
3. **Backward pass**: calculate gradients to determine how each model parameter contributed to model error
4. Account for weight sharing by using average of all connections for a parameter
5. Update each parameter using calculated gradients

Baydin et al. Automatic Differentiation in Machine Learning: a Survey. 2018

Training Procedure

Still descend an error surface, E, based on the chosen objective function (cross entropy loss)



Repeat until stopping criterion met:

1. **Forward pass**: propagate training data through model to make predictions
2. **Error quantification**: measure error of the model's predictions on training data using a loss function
3. **Backward pass**: calculate gradients to determine how each model parameter contributed to model error
4. Account for weight sharing by using average of all connections for a parameter
5. Update each parameter using calculated gradients

Training Procedure

Gradient computed for all values in all convolutional filters (i.e., model weights) as well as all bias terms

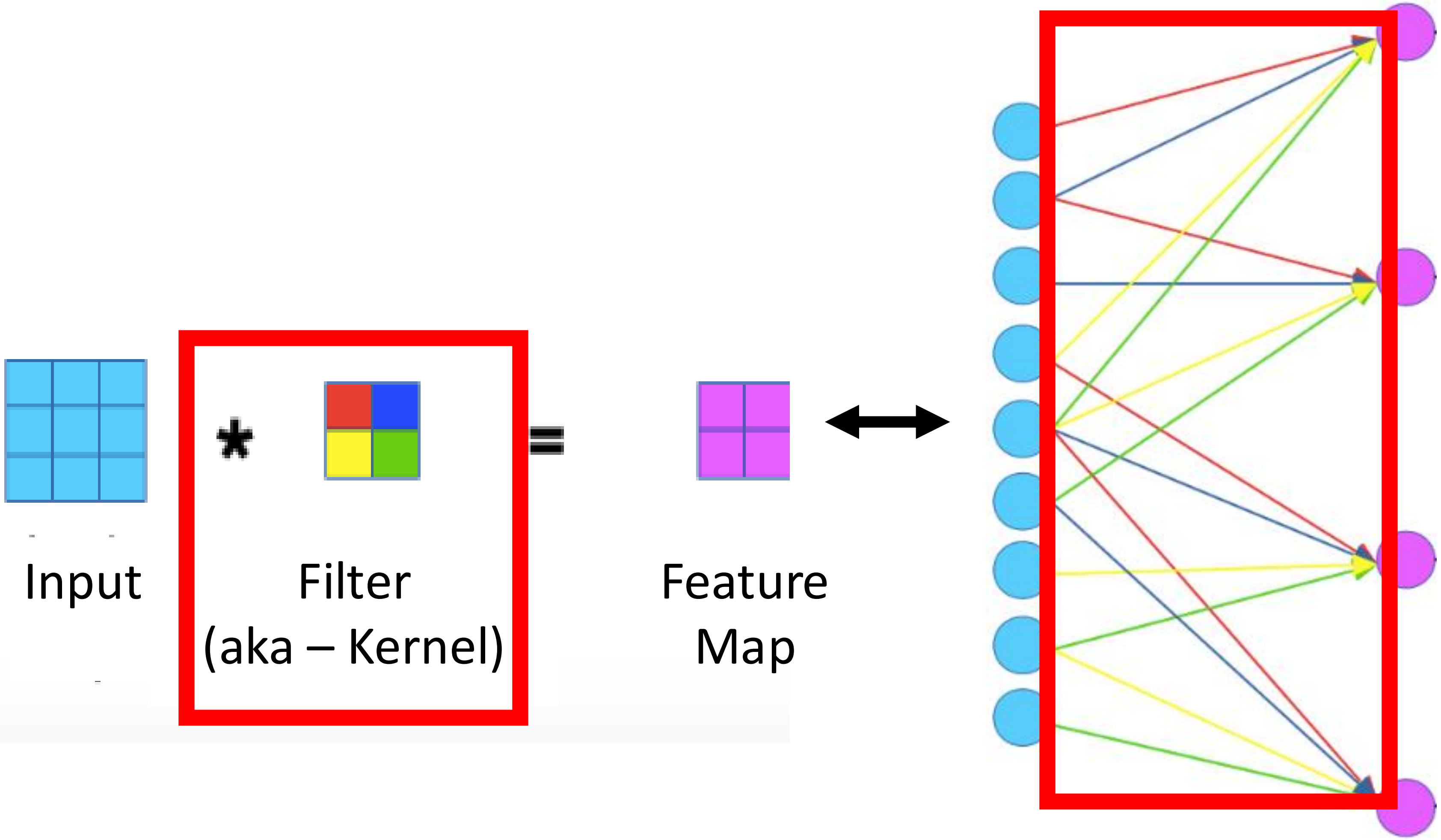
(covered in Section 6.3 of Kamath book and https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/)

Repeat until stopping criterion met:

1. **Forward pass**: propagate training data through model to make predictions
2. **Error quantification**: measure error of the model's predictions on training data using a loss function
3. **Backward pass**: calculate gradients to determine how each model parameter contributed to model error
4. Account for weight sharing by using average of all connections for a parameter
5. Update each parameter using calculated gradients

Yann Lecun. Generalization and network design strategies. Technical Report, 1989

Training Procedure (Key Novelty)



Repeat until stopping criterion met:

1. **Forward pass**: propagate training data through model to make predictions
2. **Error quantification**: measure error of the model's predictions on training data using a loss function
3. **Backward pass**: calculate gradients to determine how each model parameter contributed to model error
4. Account for weight sharing by using average of all connections for a parameter
5. Update each parameter using calculated gradients

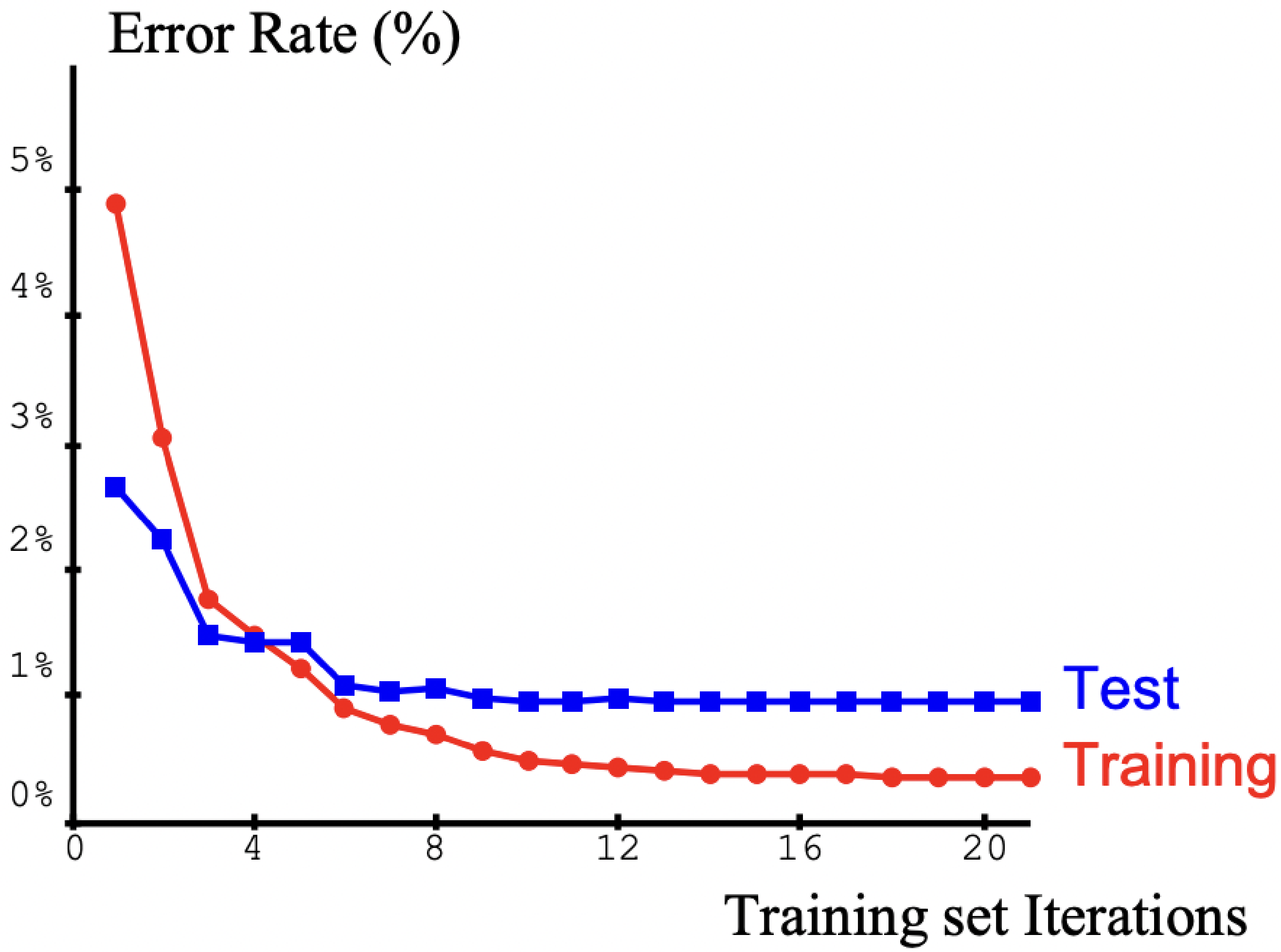
https://www.jefkine.com/general/2016/09/05/backpropagation-in-convolutional-neural-networks/

LeNet Analysis

How many epochs are needed for training to converge?

Why might overfitting not arise with more training?

- Learning rate too large for the model to settle in a local minimum (instead oscillated randomly)



Y. Lecun ; L. Bottou ; Y. Bengio ; P. Haffner; Gradient-based learning applied to document recognition; 1998

LeNet Analysis

All 82 mislabeled examples (correct answer on left, predicted answer on right):

Why might the model be making mistakes?

- Insufficient representation in the training data
- Ambiguity

Y. Lecun ; L. Bottou ; Y. Bengio ; P. Haffner; Gradient-based learning applied to document recognition; 1998

Today's Topics

- History of Convolutional Neural Networks (CNNs)
- CNNs – Convolutional Layers
- CNNs – Pooling Layers
- Pioneering CNN model: LeNet

The End