UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Deep Learning

Session 10 and 11

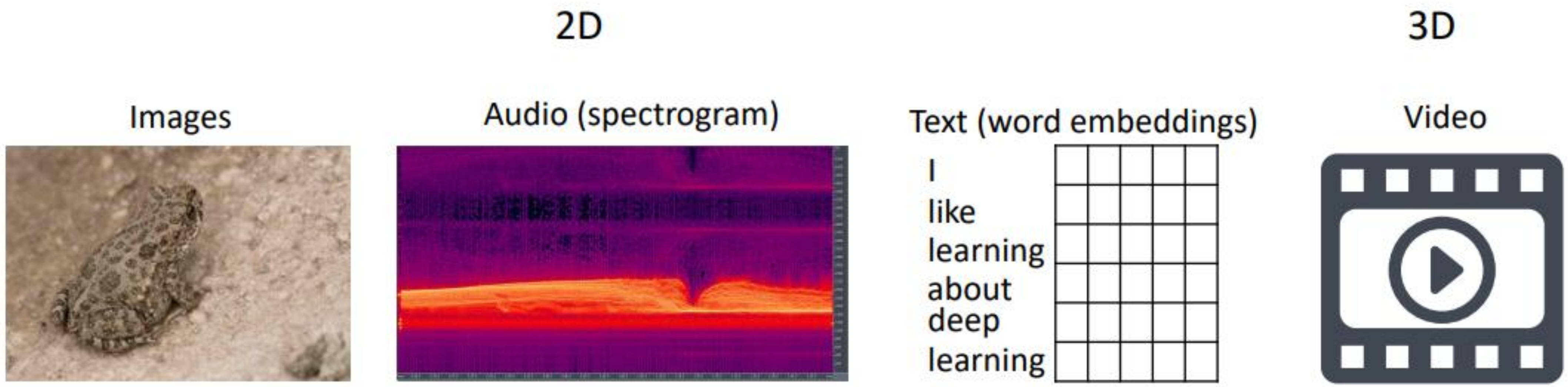
Convolutional Neural Networks (CNNs)

Applied Data Science

2024/2025

Spatial Data

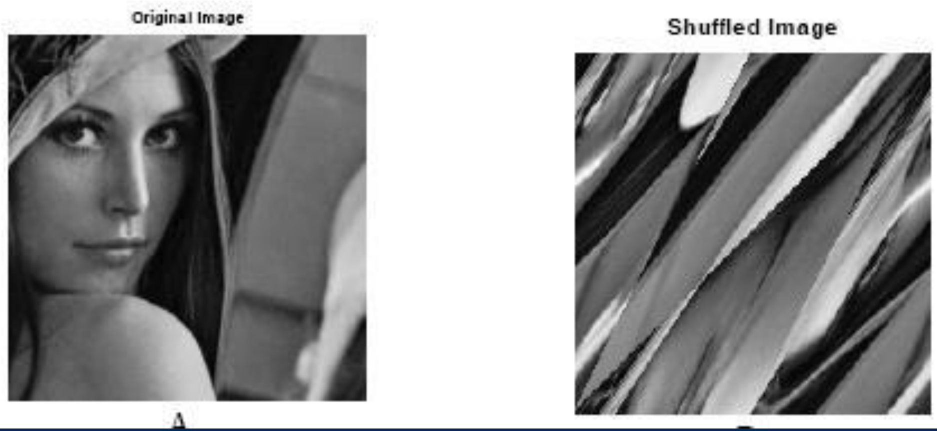
- **Definition:** Spatial data refers to data where the arrangement or location of elements is important.



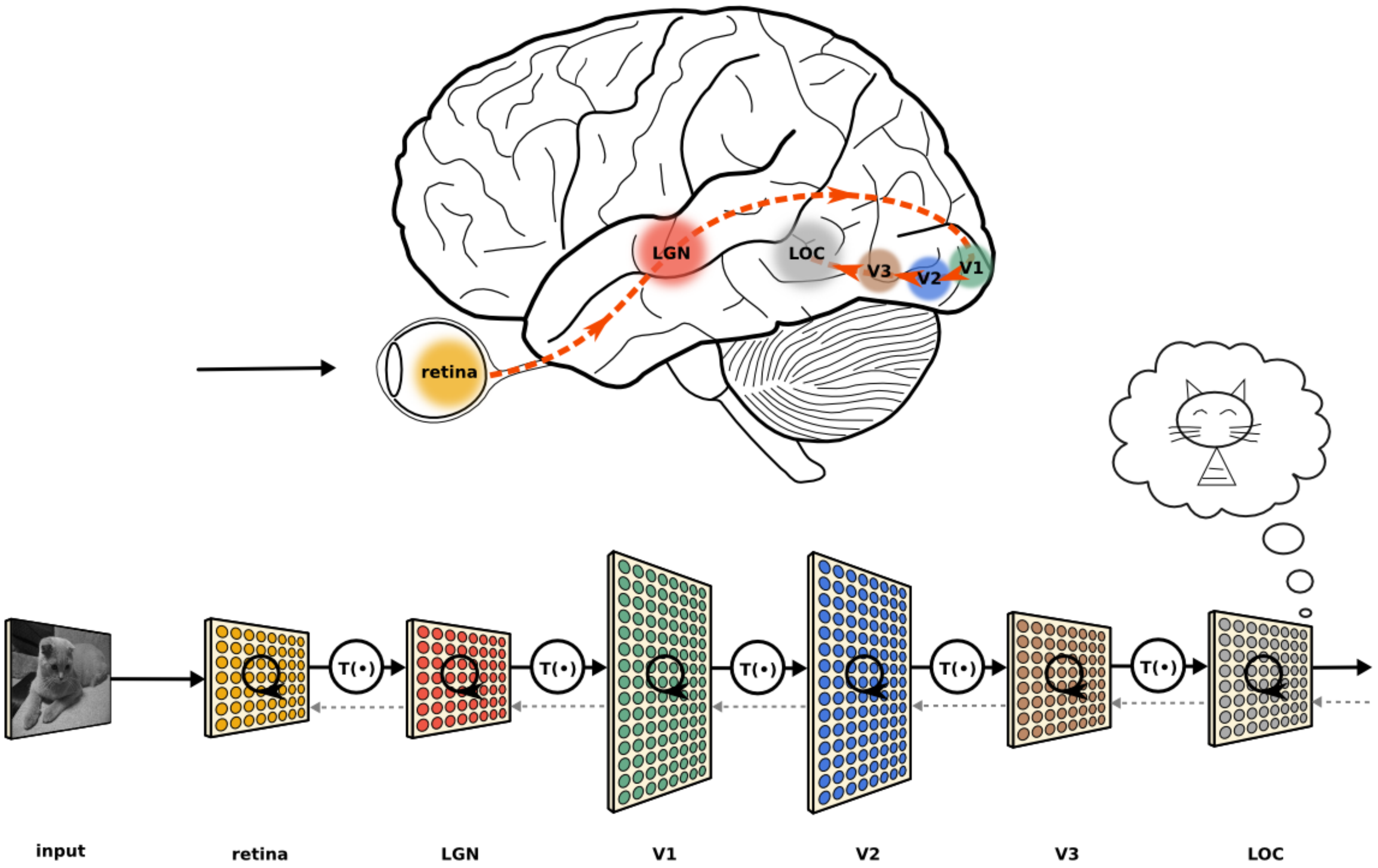
Why Spatial Structure Matters?

- **Local Patterns:** In spatial data, nearby elements (e.g., pixels) often have strong relationships.
  - Example: Edges, textures, and objects in an image are defined by neighboring pixel values.

- **Context Preservation:** Spatial arrangement helps capture context.
  - Example: In an image, nearby pixels form features like eyes, while distant pixels represent unrelated parts (e.g., background).



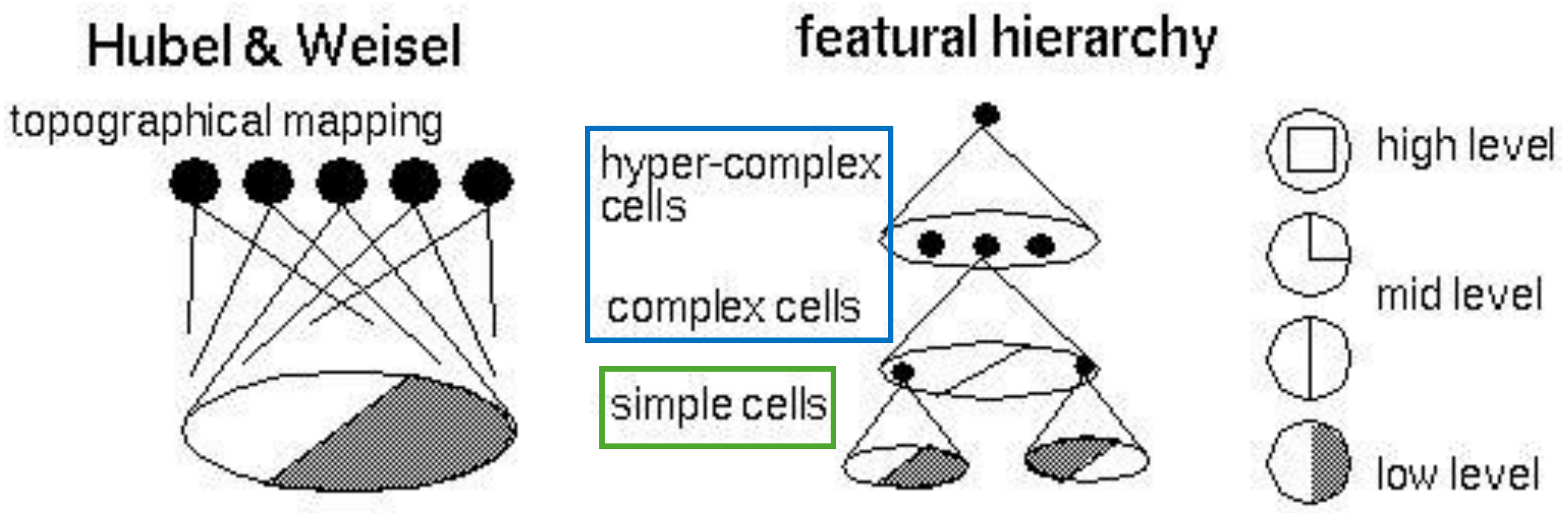
Motivation: How Vision Systems Work



de-Wit, L. H., Kubilius, J., de Beeck, H. P. O., & Wagemans, J. (2013). Configural Gestalts Remain Nothing More Than the Sum of Their Parts in Visual Agnosia. In i-Perception (Vol. 4, Issue 8, pp. 493–497). SAGE Publications. https://doi.org/10.1068/i0613rep

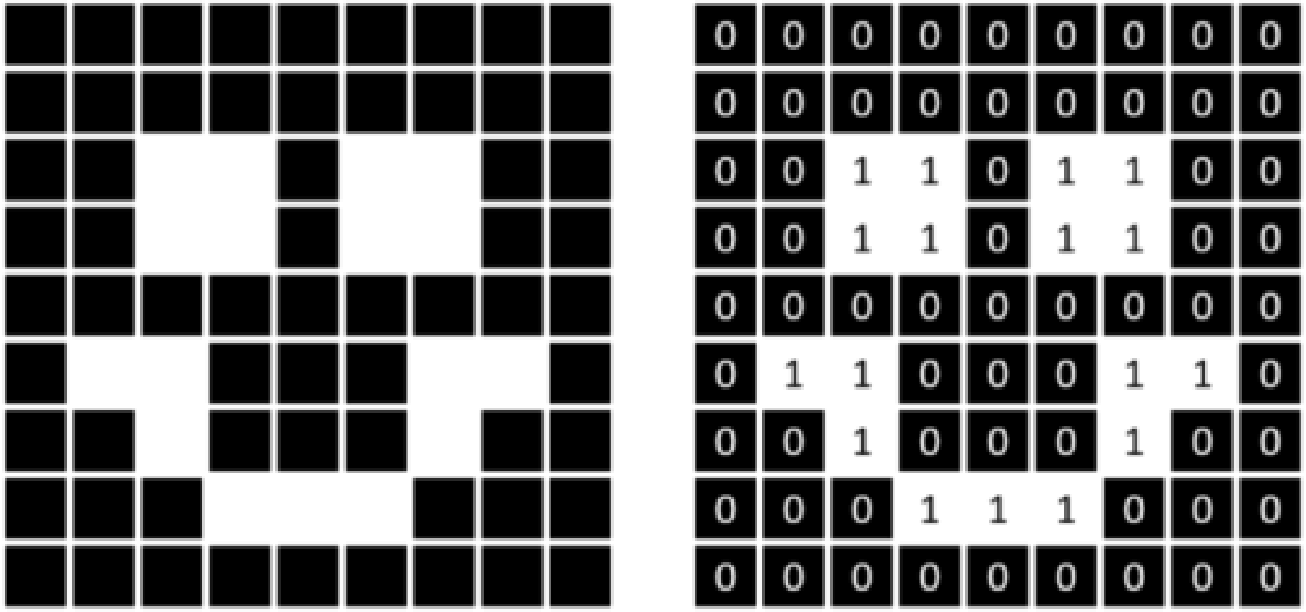
Motivation: How Vision Systems Work

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



What computers "see"?

- Images are Numbers



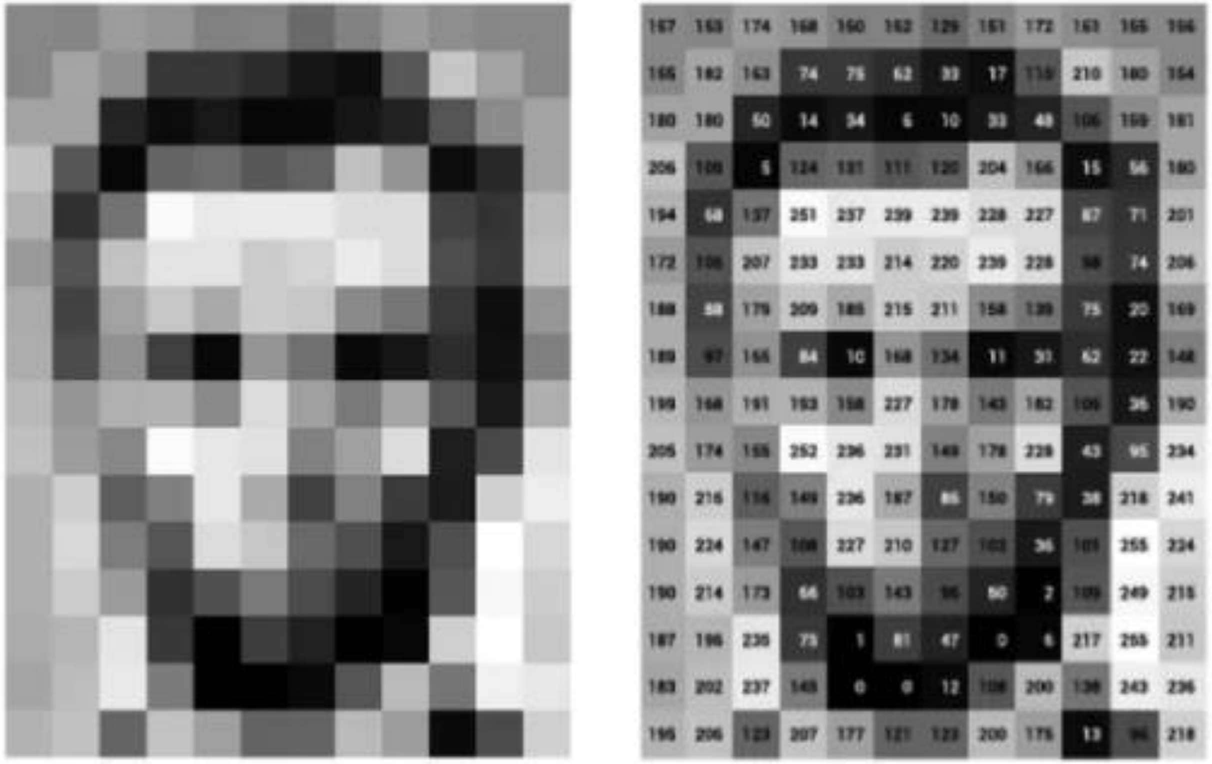
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	1	1	0	1	1	0	0
0	0	1	1	0	1	1	0	0
0	0	0	0	0	0	0	0	0
0	1	1	0	0	0	1	1	0
0	0	1	0	0	0	1	0	0
0	0	0	1	1	1	0	0	0
0	0	0	0	0	0	0	0	0

Binary Images

What computers "see"?

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Images are Numbers

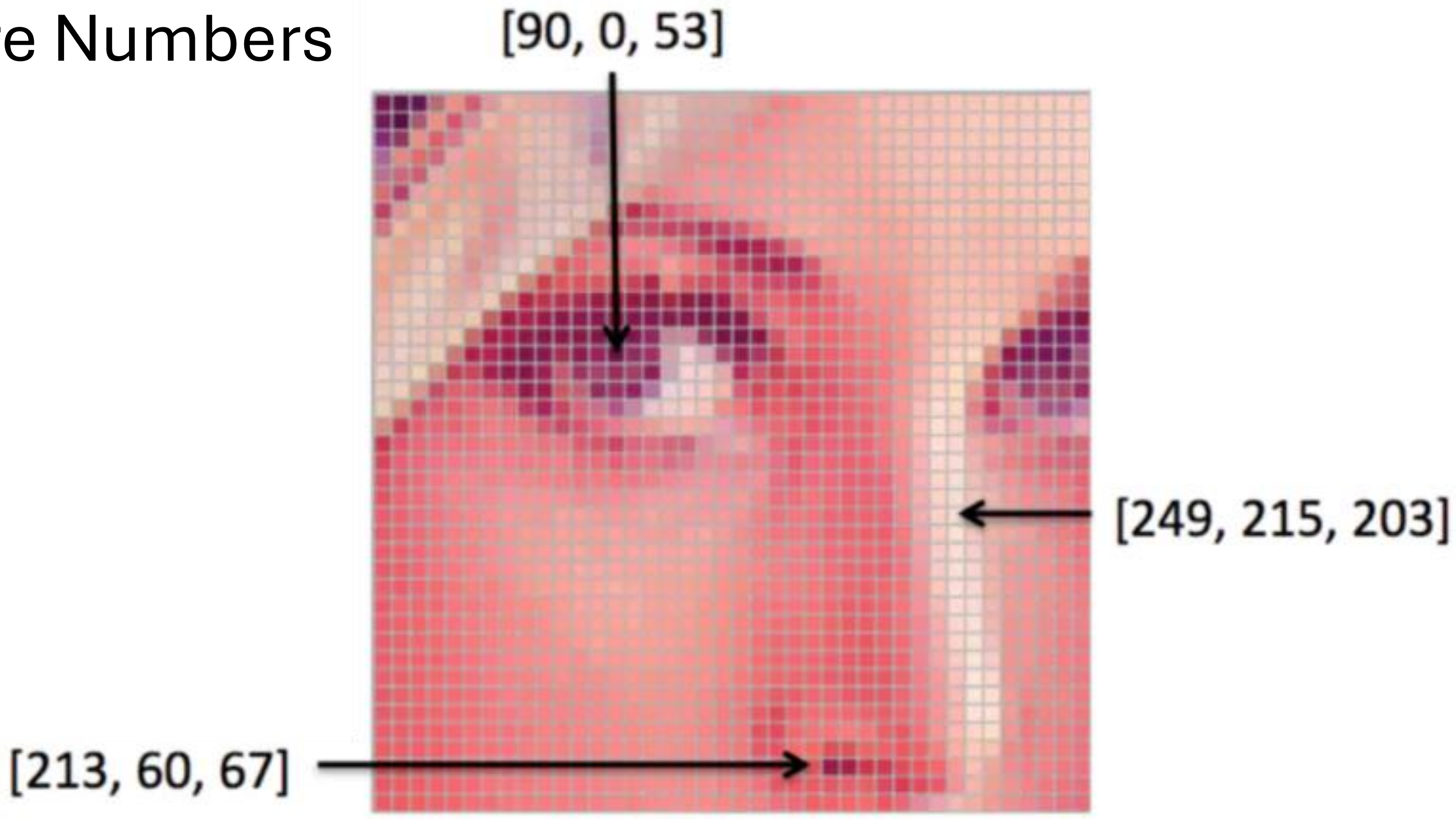


157	153	174	168	150	152	129	151	172	161	155	156
155	182	163	74	75	62	33	17	110	210	180	154
180	180	50	14	34	6	10	33	48	106	159	181
206	109	5	124	131	111	120	204	166	15	56	180
194	68	137	251	237	239	239	228	227	87	71	201
172	106	207	233	233	214	220	239	228	98	74	206
188	88	179	209	185	215	211	158	139	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	96	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

Greyscale Images

What computers "see"?

- Images are Numbers



Color Images

Neocognitron

“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

Cascade of simple and complex cells:

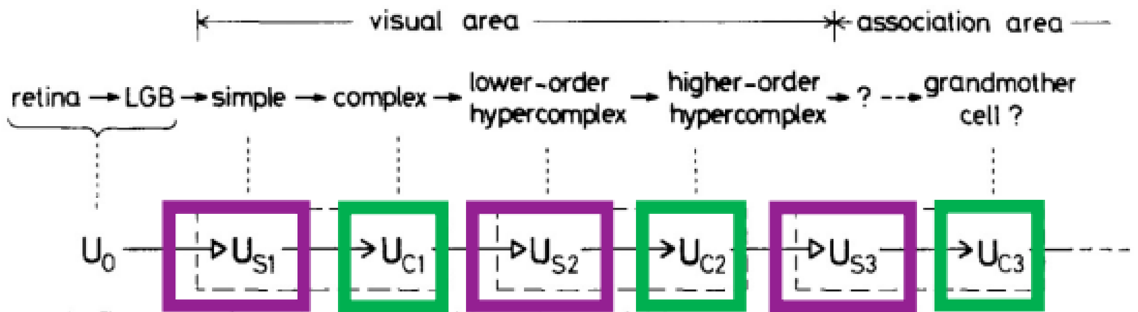


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

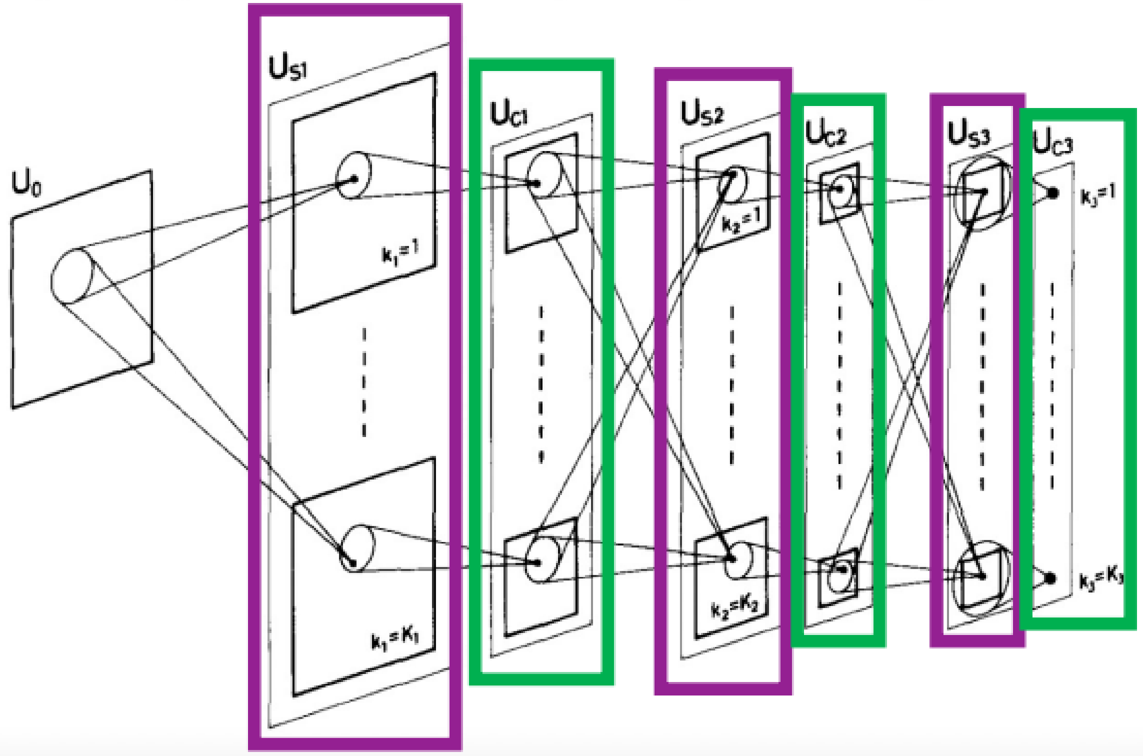


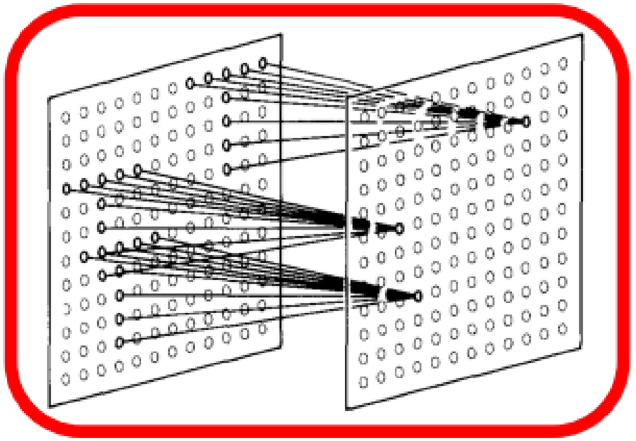
Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Neocognitron

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

Simple cells extract local features using a sliding filter:



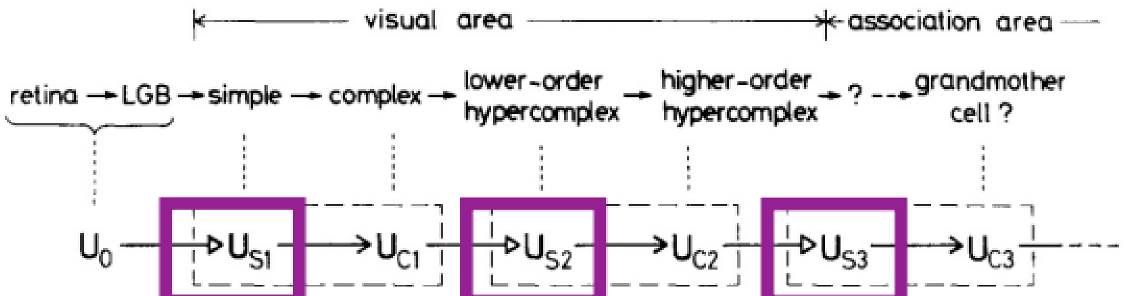


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

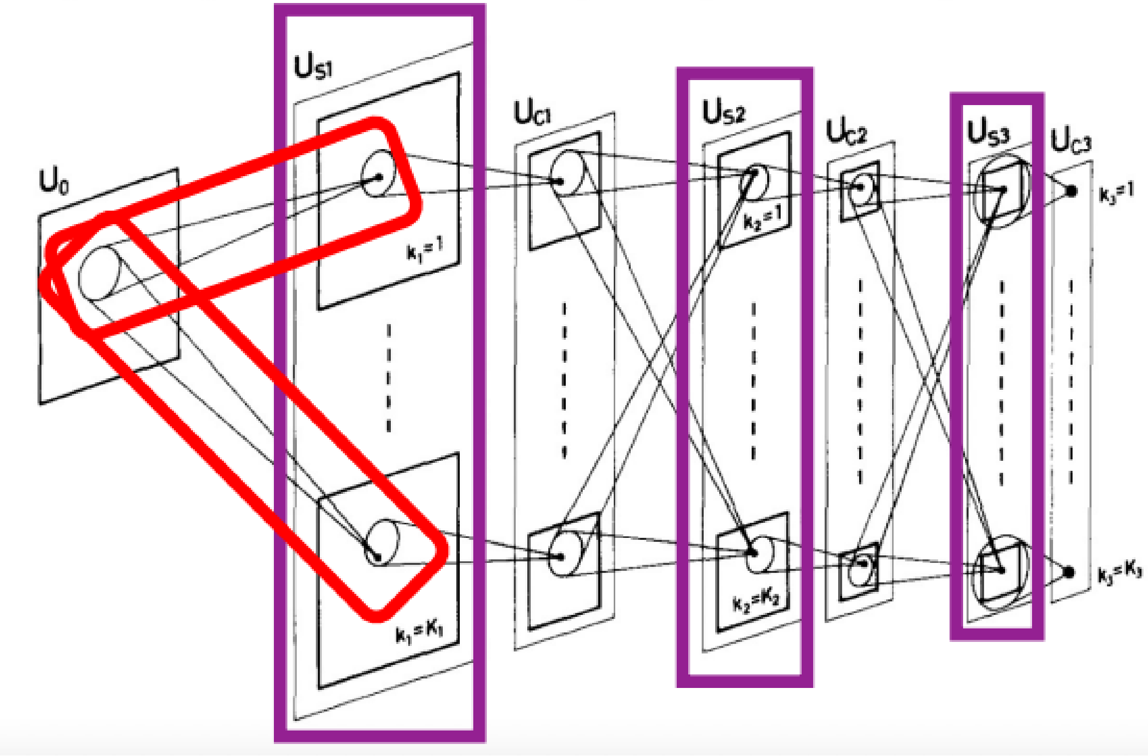


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Neocognitron

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

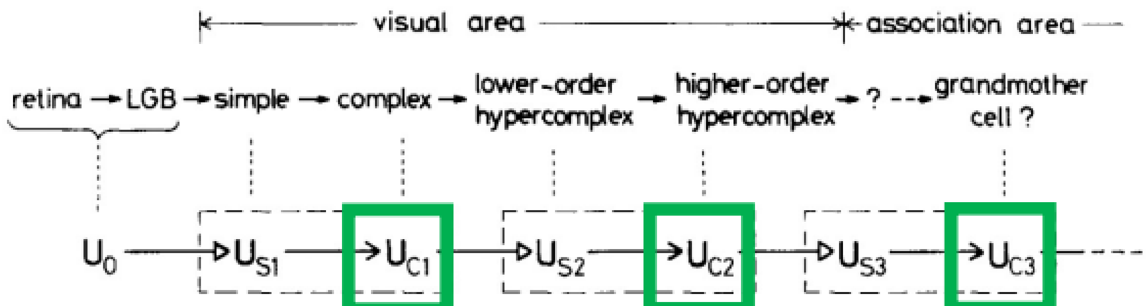


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

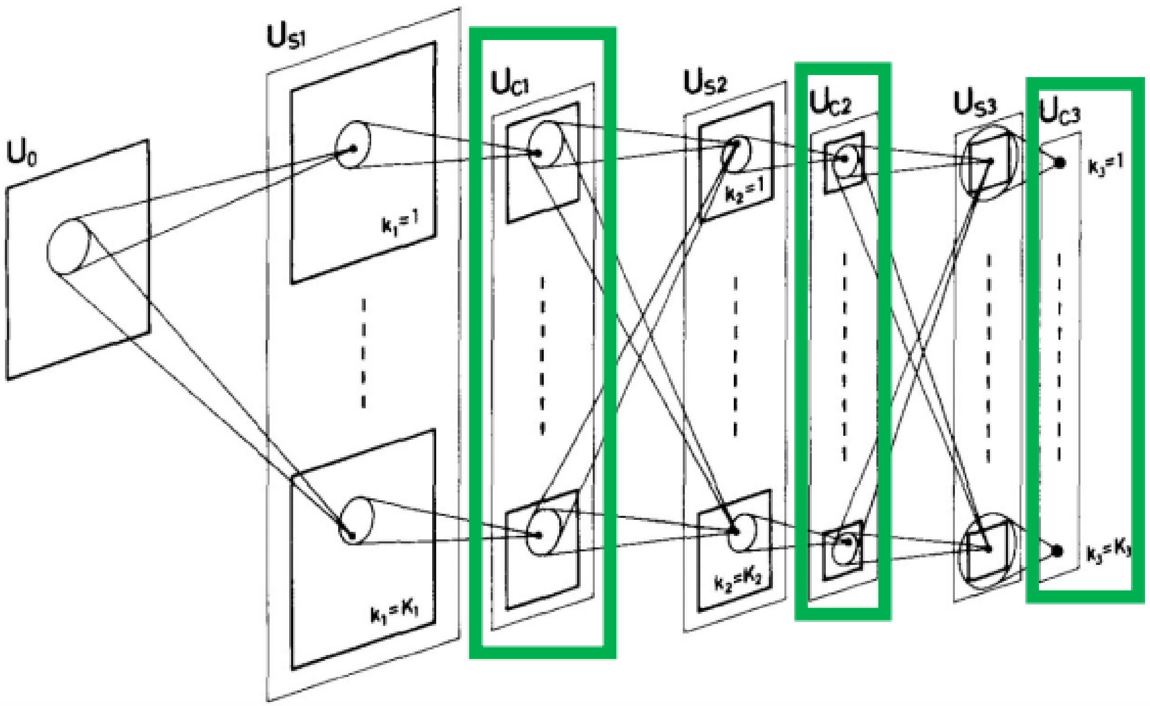
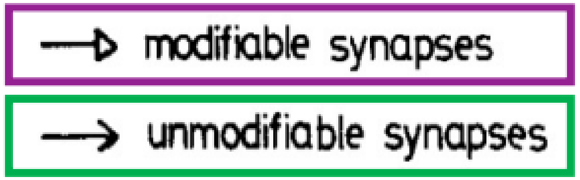


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Neocognitron

1. ~ Convolutional layers



2. ~ Pooling Layers

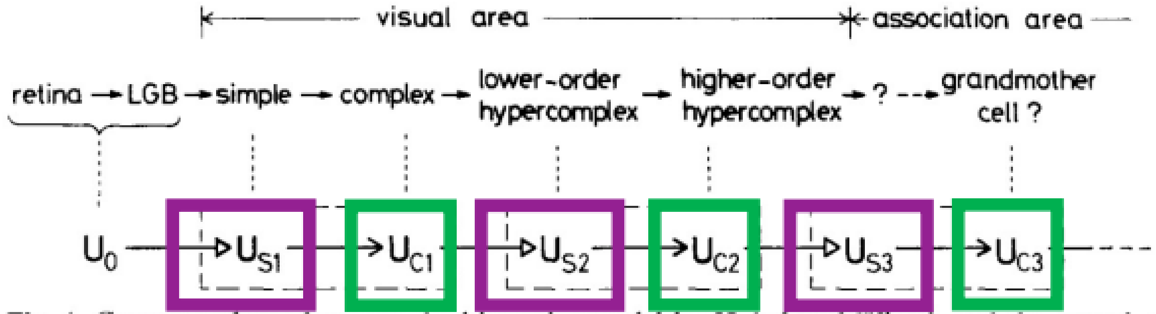


Fig. 1. Correspondence between the hierarchy model by Hubel and Wiesel, and the neural network of the neocognitron

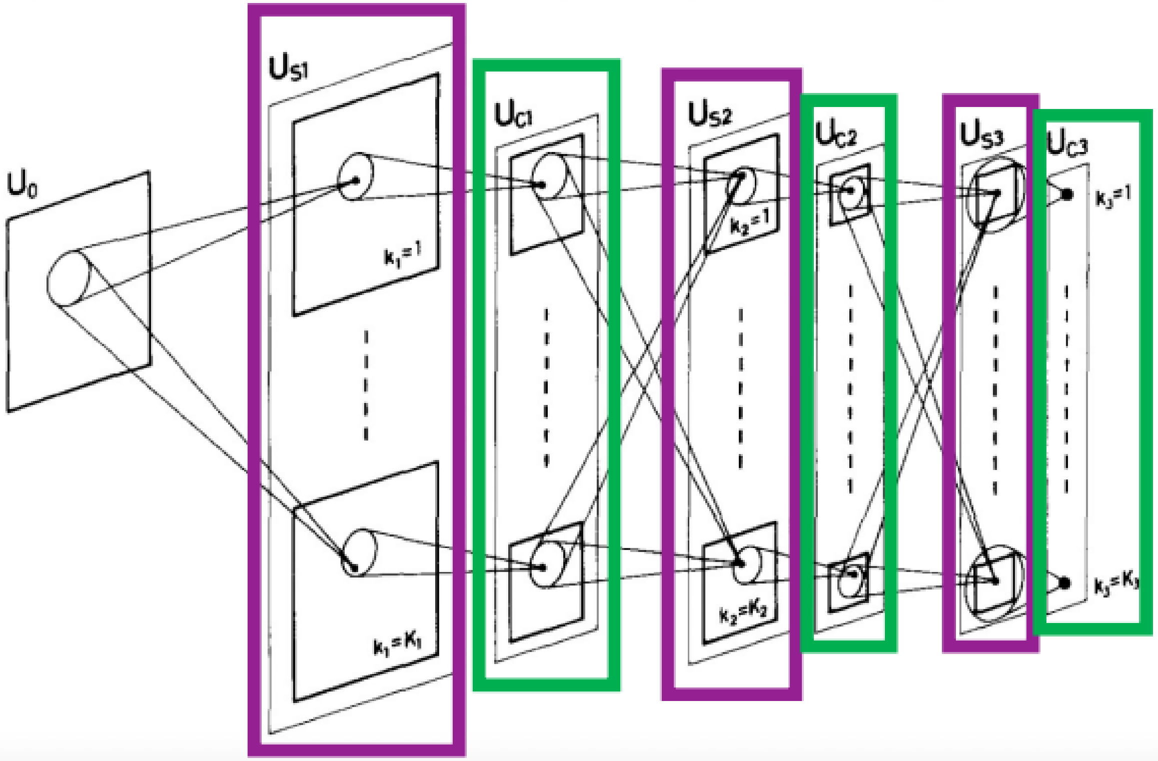
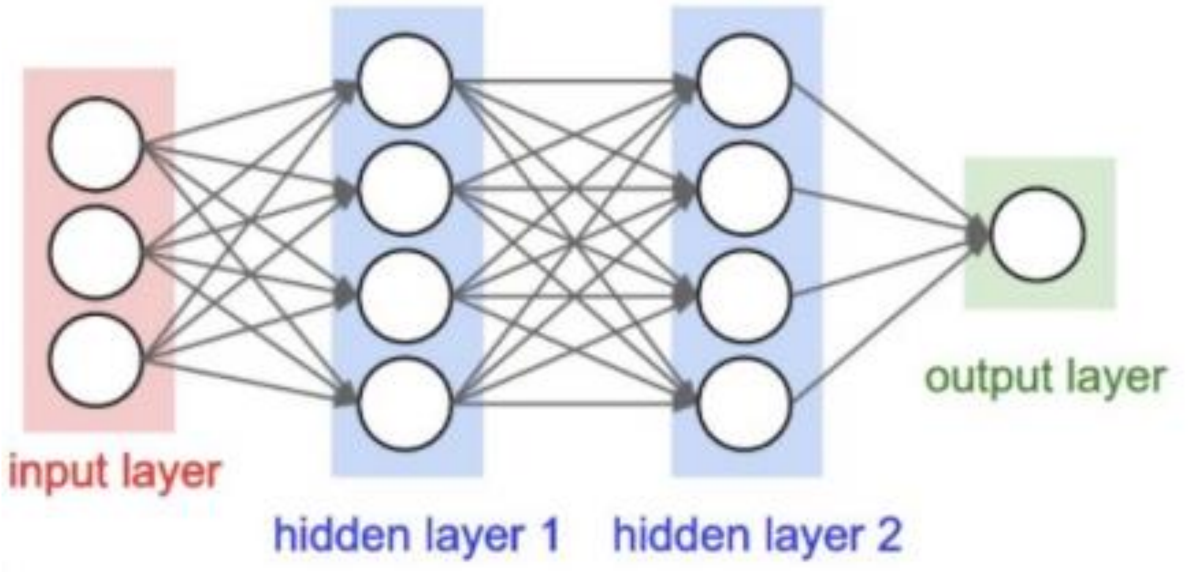


Fig. 2. Schematic diagram illustrating the interconnections between layers in the neocognitron

Fukushima, 1980.

Challenges of Fully Connected Networks

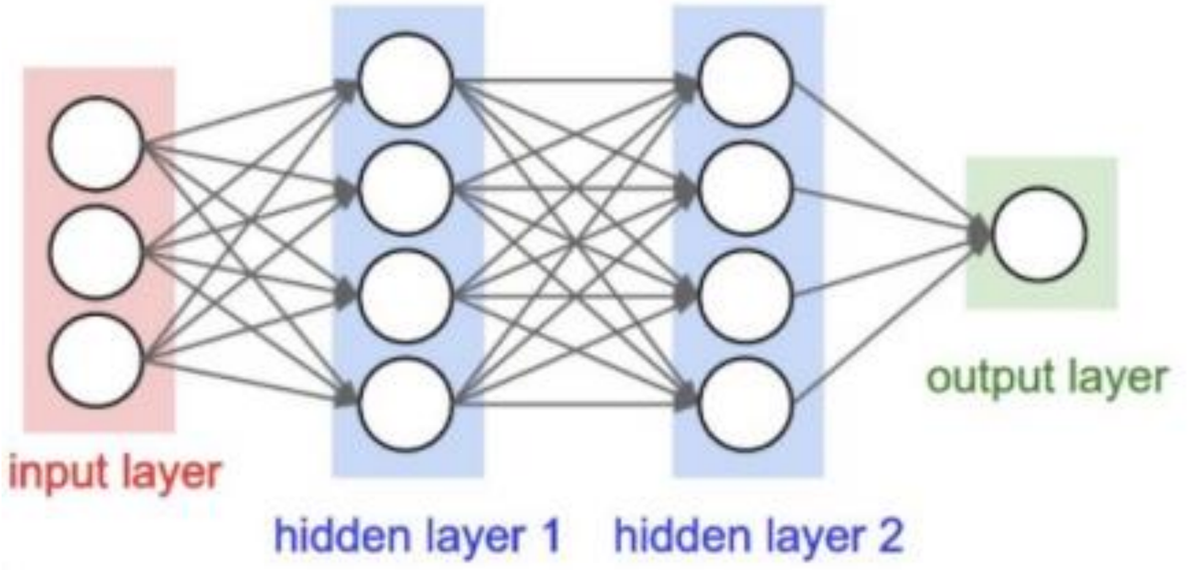
- Fully-Connected Layers are Limited



- Each node provides input to each node in the next layer.

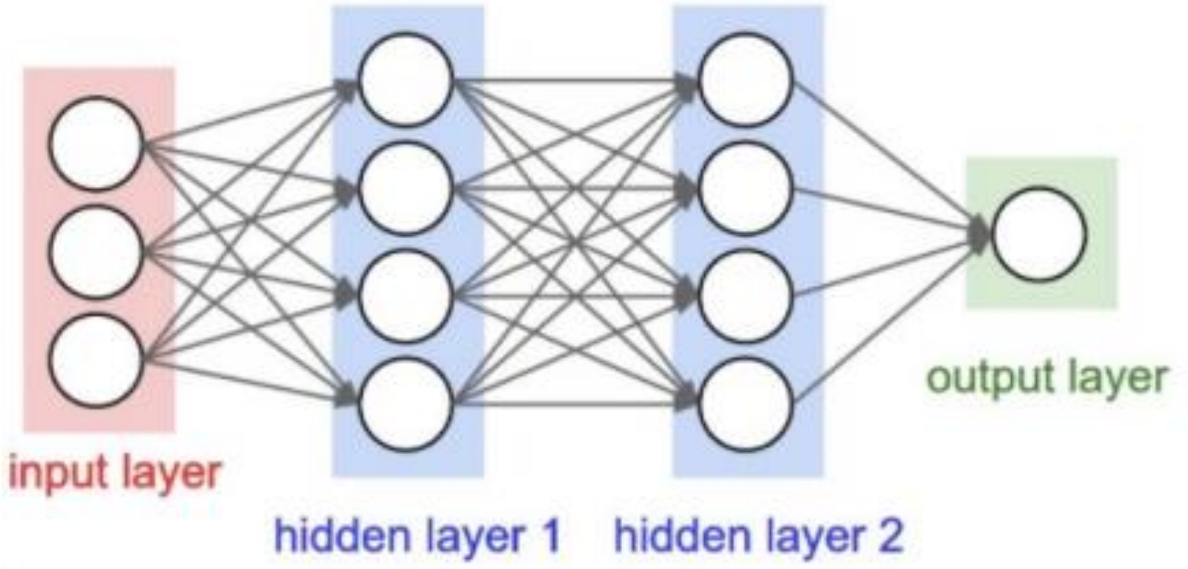
- No spatial information!

Challenges of Fully Connected Networks



- Assume 2 layer model with 100 nodes per layer
  - e.g., how many weights are in a colored 640x480 image?

Challenges of Fully Connected Networks



- Assume 2 layer model with 100 nodes per layer
  - e.g., how many weights are in a colored 640x480 image?
    - 640x480x3x100 + 100x100 + 100x1 = 92,170,100
  - e.g., how many weights are in a 2048X1536 image (3.1 Megapixel image)?
    - 2048x1536x3x100 + 100x100 + 100x1 = 943,728,500

Challenges of Fully Connected Networks

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

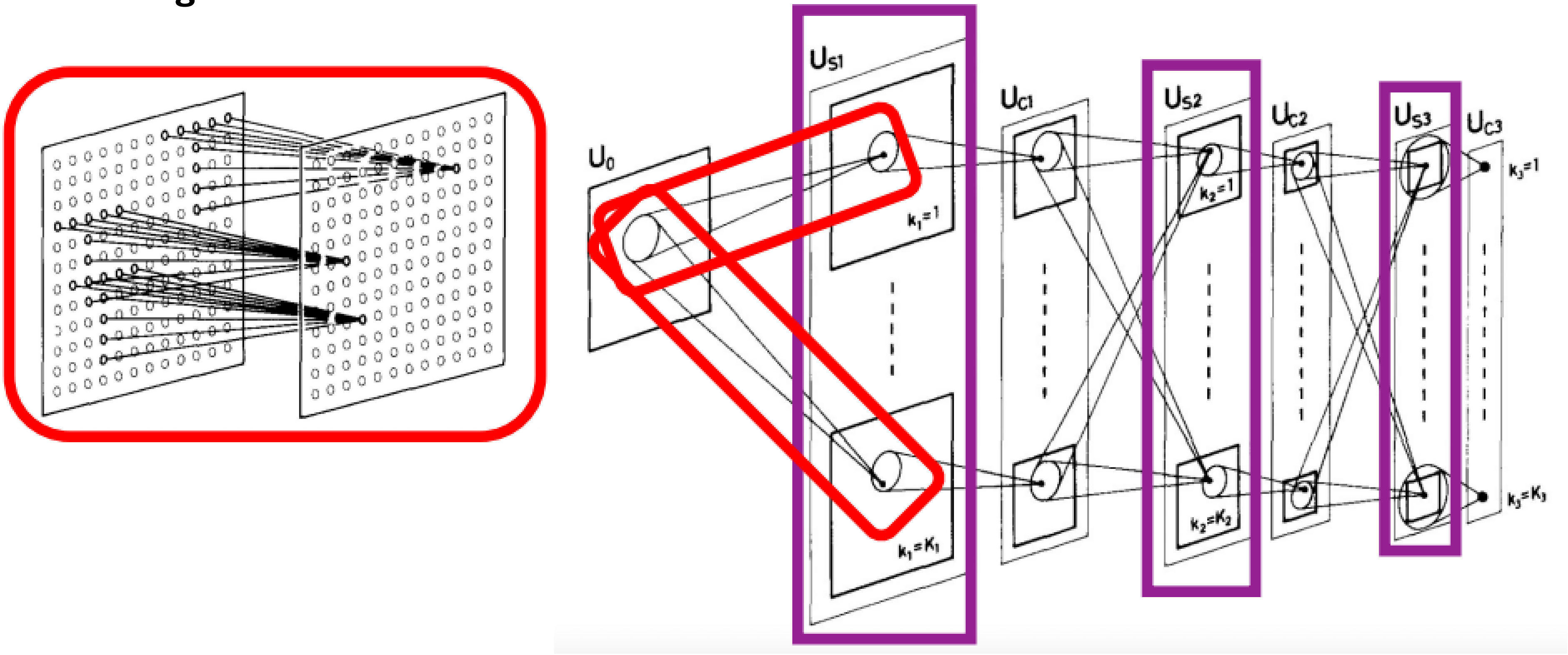
- **Issue:** many model parameters in fully connected networks

- **Many model parameters and so…**
  - **Greater chance to overfit**
  - **Increased training time**
  - **Needs more training data**

Convolutional Layers

- **Idea:** each node receives input only from a small neighborhood in previous layer and parameter sharing

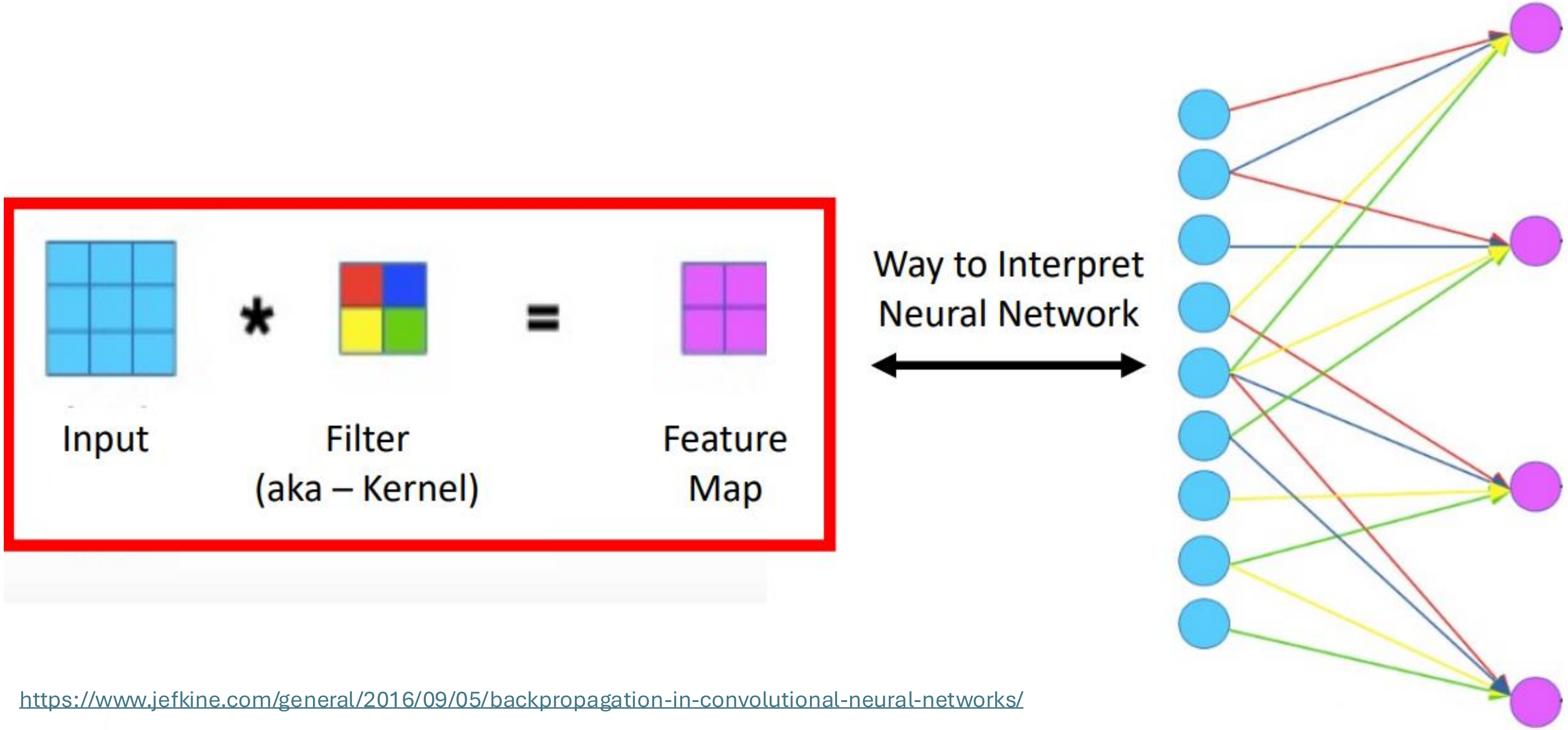
Neocognitron



Fukushima, 1980.

Convolution

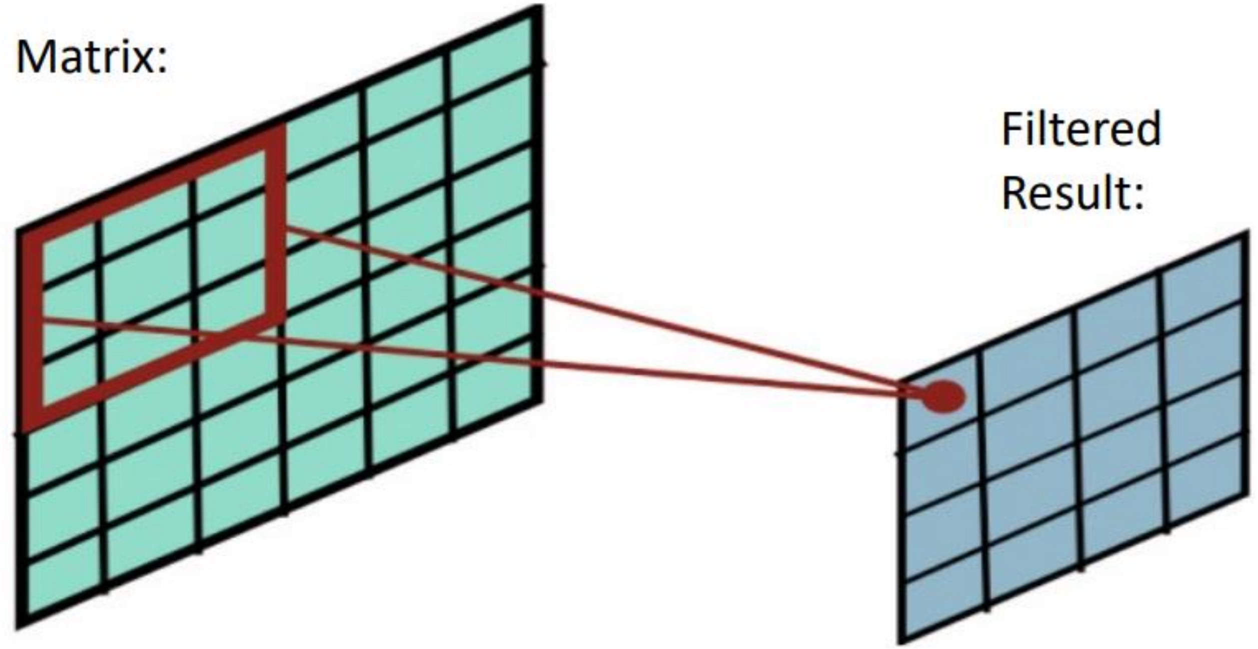
- Applies a linear **filter** (e.g., 2D)



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

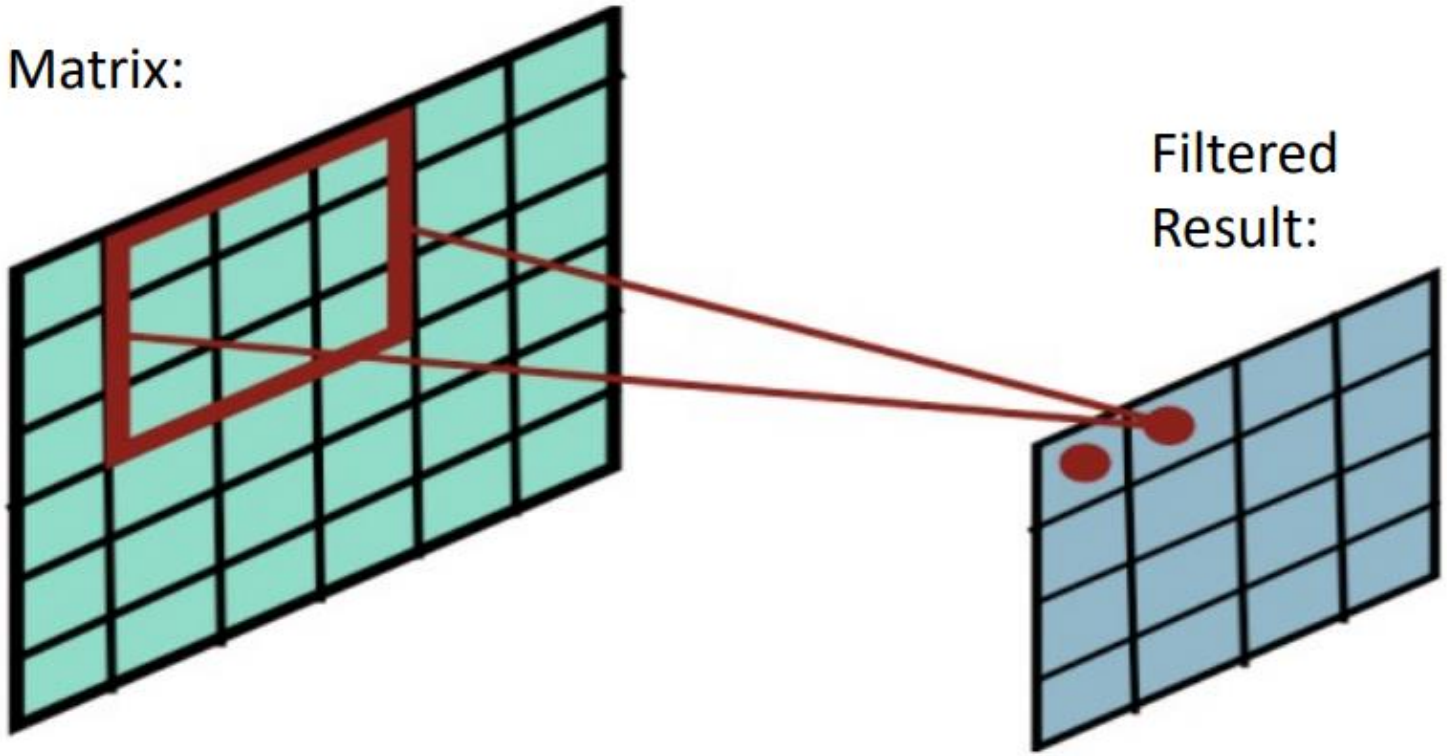
2D Filtering

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



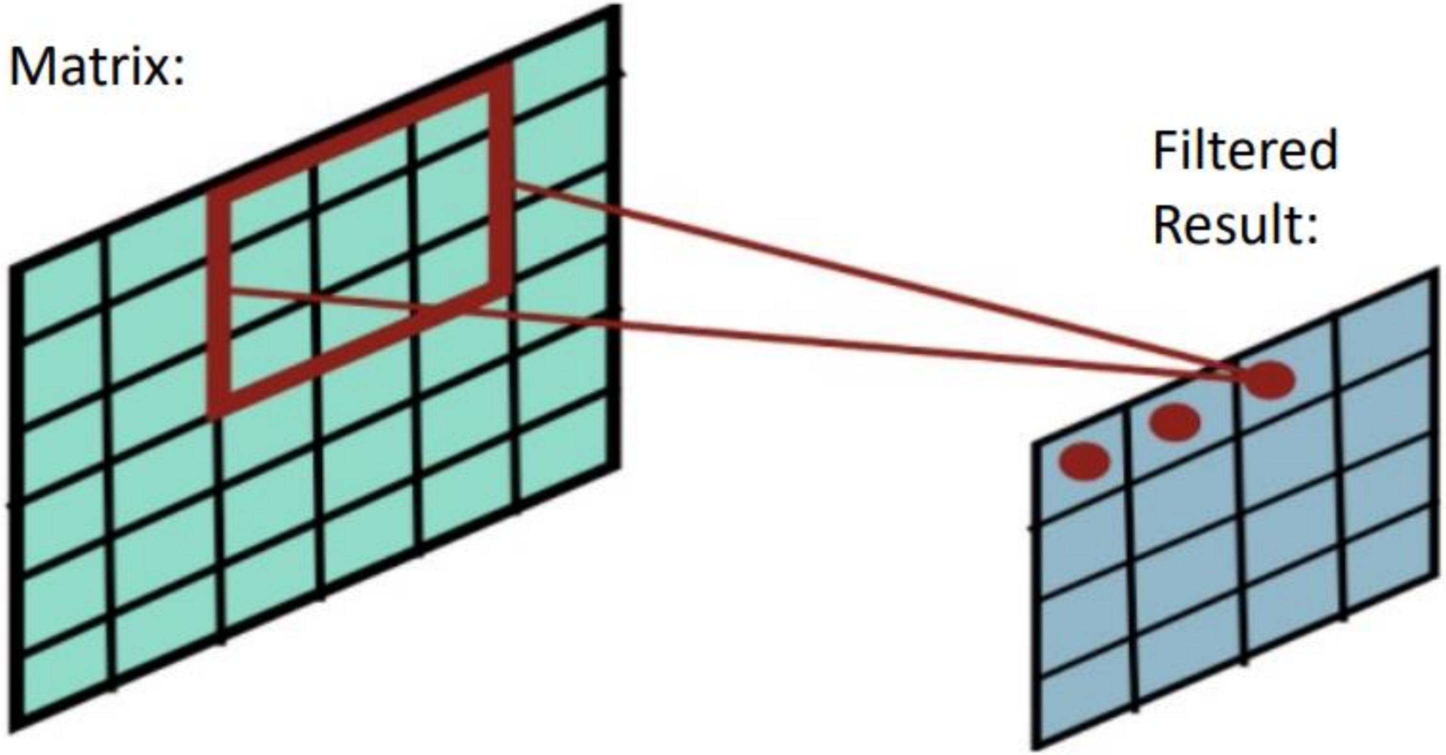
Slides filter over the matrix and computes dot products

2D Filtering



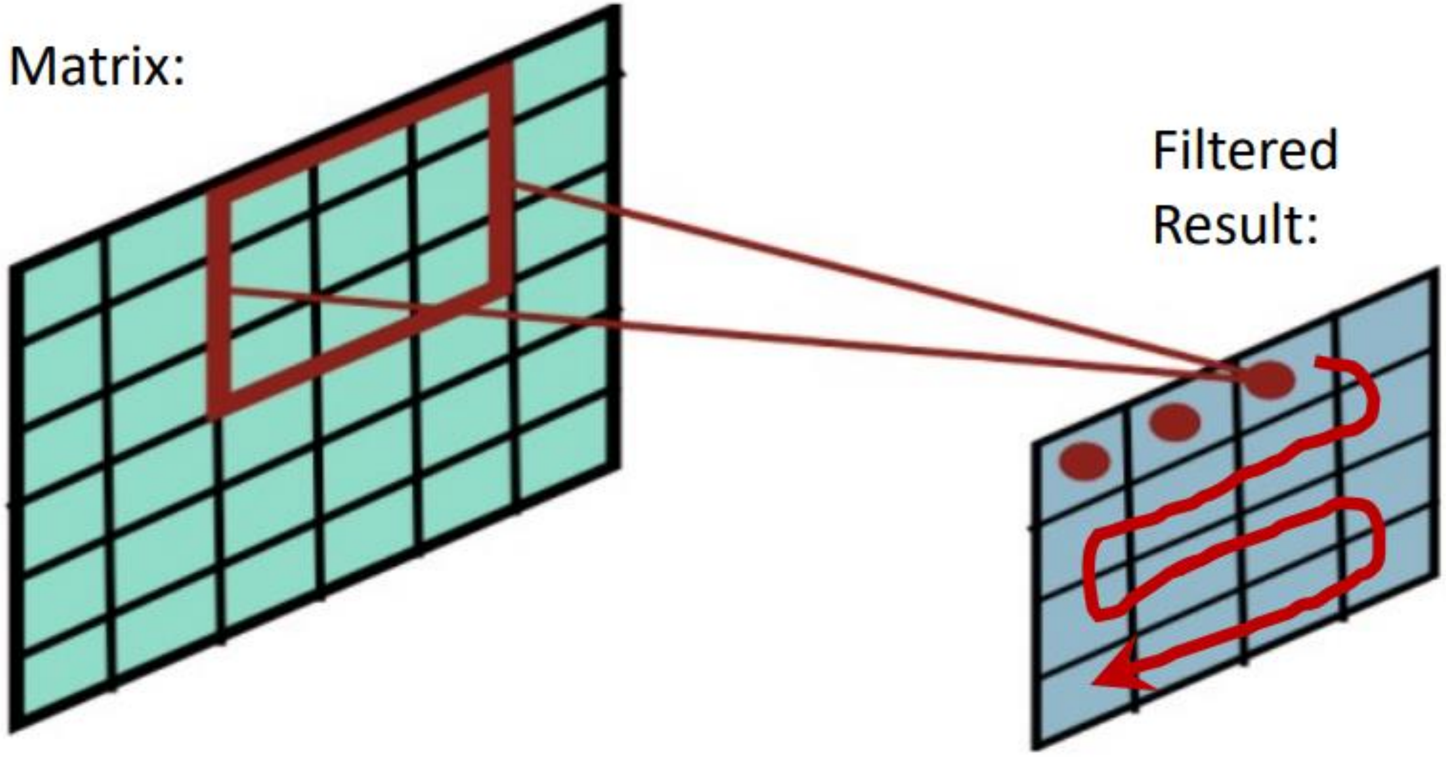
Slides filter over the matrix and computes dot products

2D Filtering



Slides filter over the matrix and computes dot products

2D Filtering



Slides filter over the matrix and computes dot products

2D Filtering: Example

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

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: 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: Example

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

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: 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: 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: 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: Example

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

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: Example

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

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: 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: Example

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

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

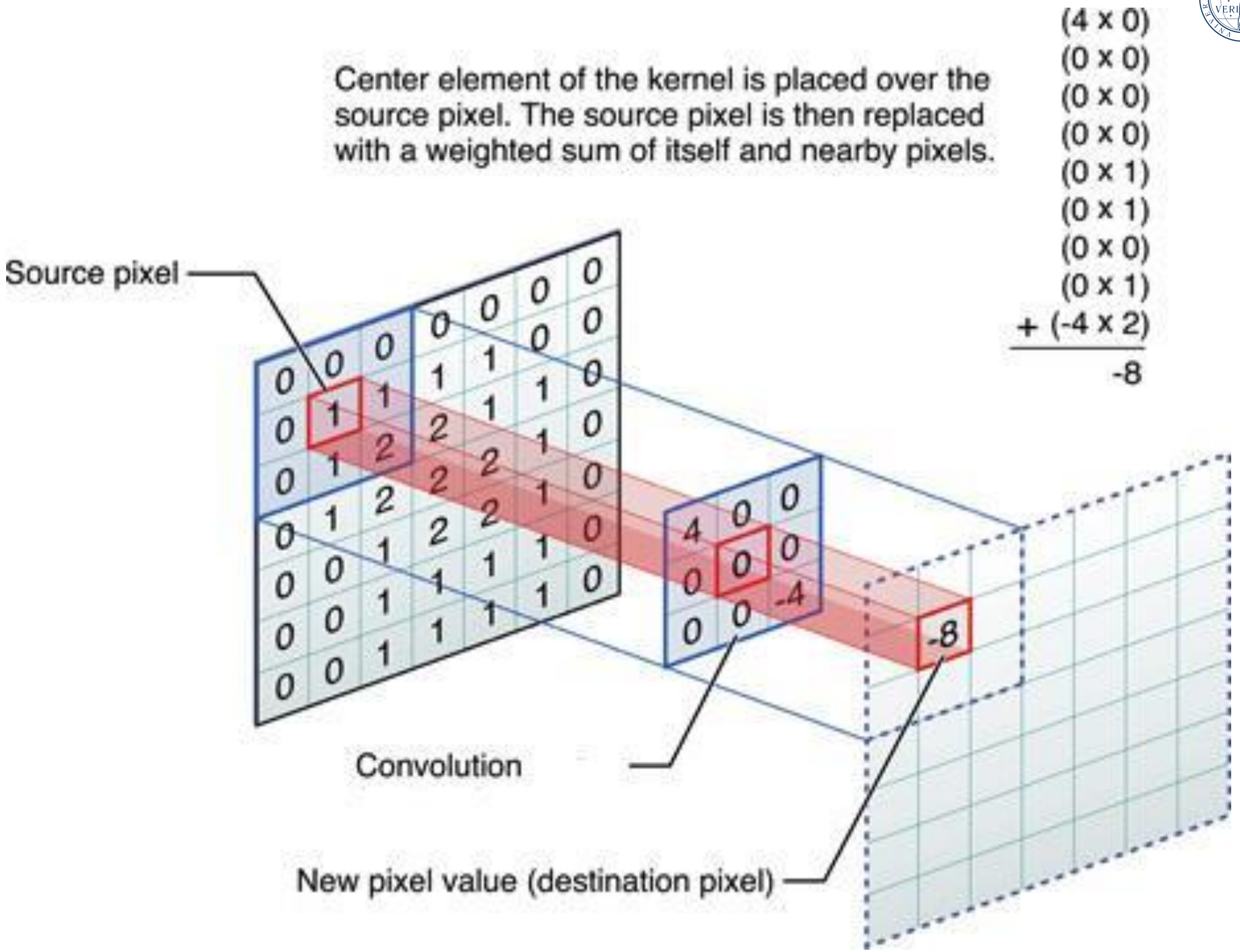
Filter

1	0	1
0	1	0
1	0	1

Feature Map

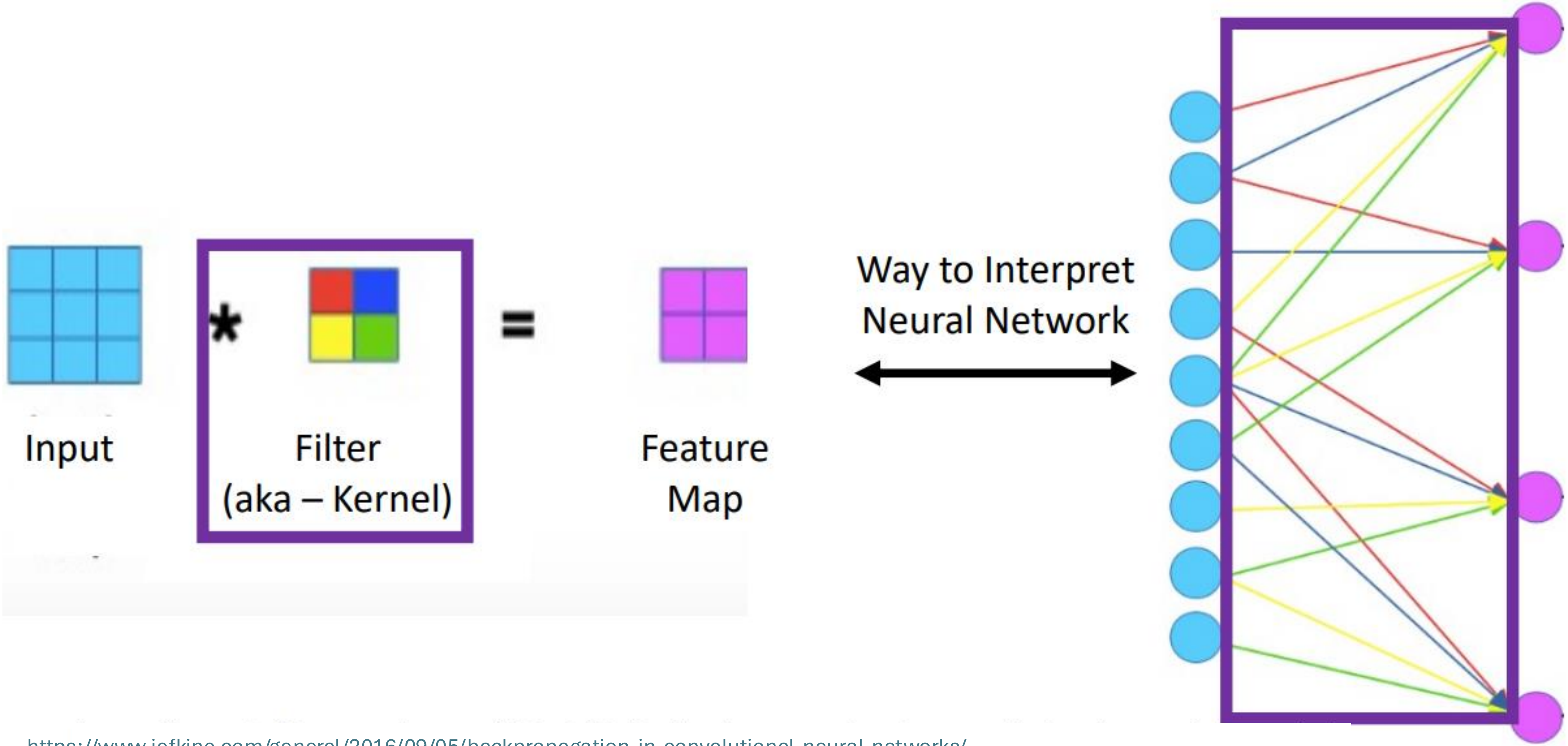
4	3	4
2	4	3
2	3	4

Convolution



https://medium.com/@bdhuma/6-basic-things-to-know-about-convolution-daef5e1bc411

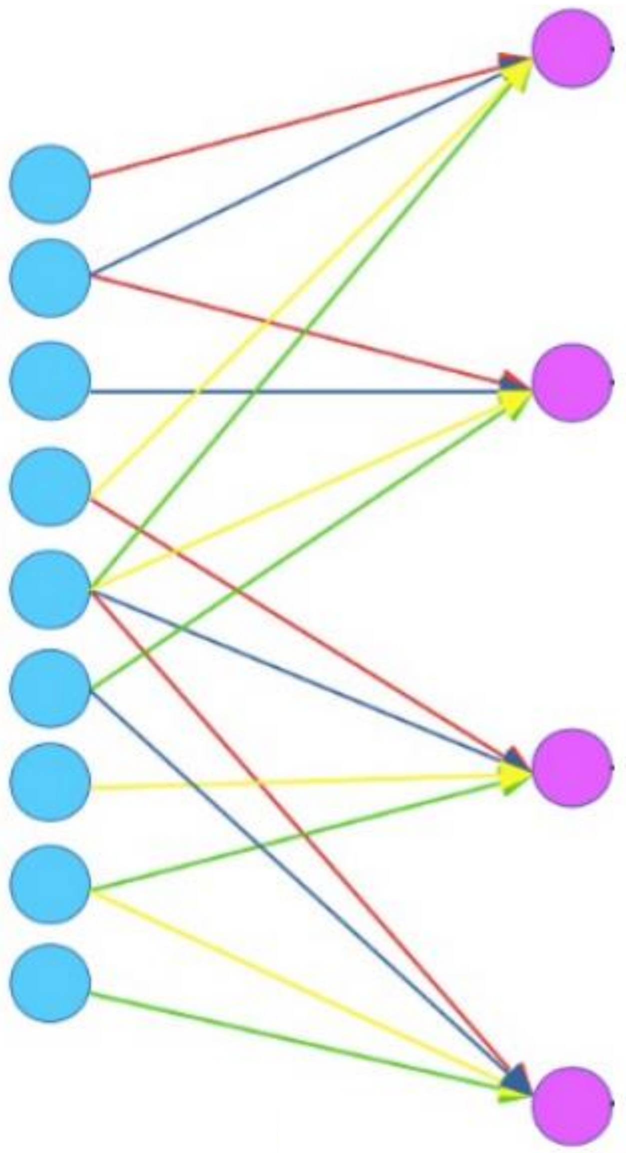
Convolutional Layer: Parameters to Learn



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

Convolutional Layer: Parameters to Learn

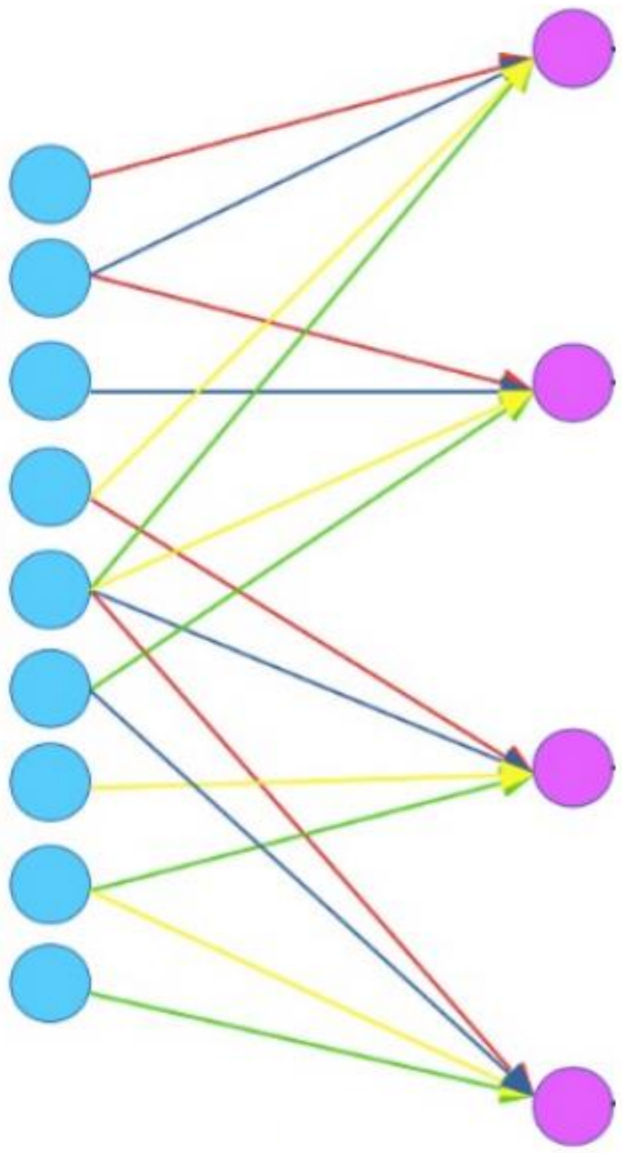
- For the shown example, how many weights must be learned?



Convolutional Layer: Parameters to Learn

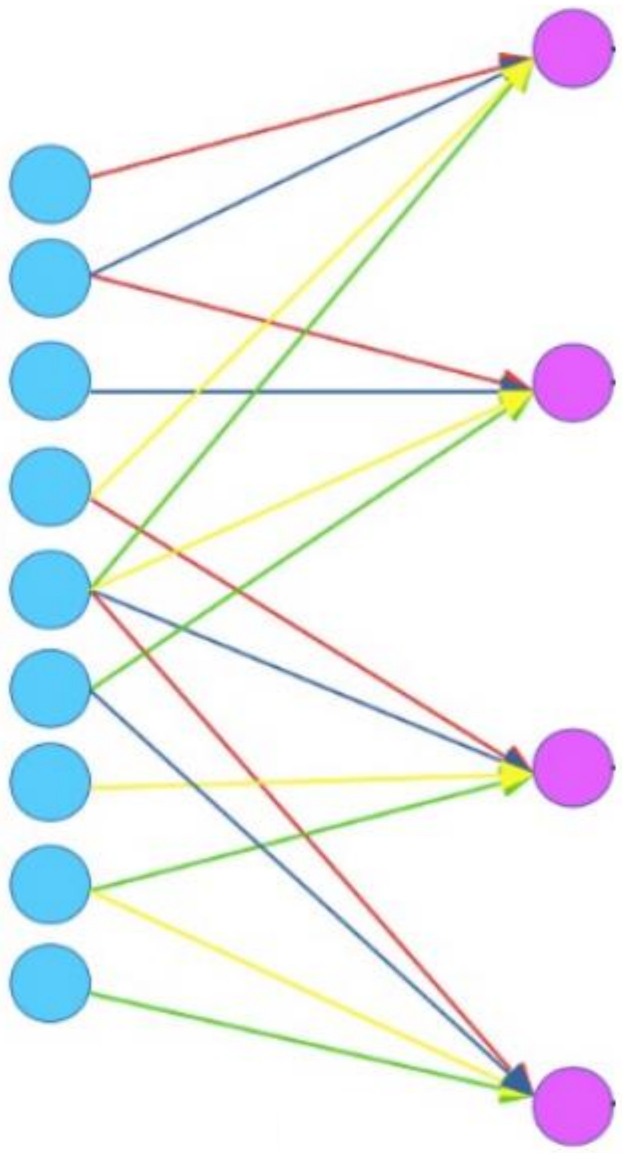
- For the shown example, how many weights must be learned?
  - **4 (red, blue, yellow, and green values)**

- If we instead used a fully connected layer, how many weights would need to be learned?



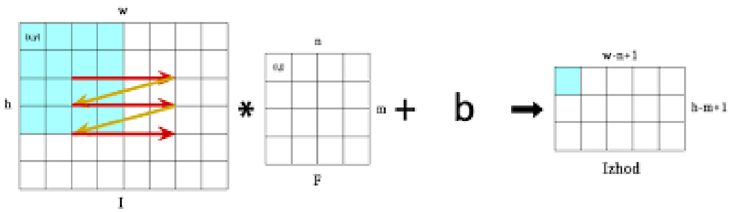
Convolutional Layer: Parameters to Learn

- For the 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)**
- For shown example, how many parameters must be learned?

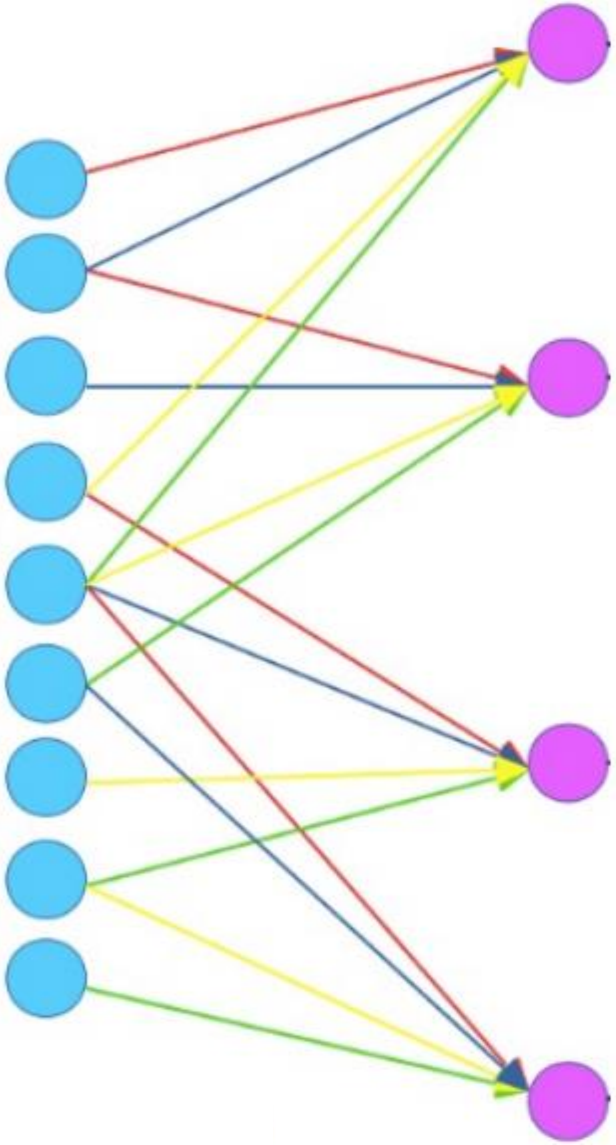


Convolutional Layer: Parameters to Learn

- For the 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)
- For the shown example, how many parameters must be learned?
  - **5 (4 weights + 1 bias)**

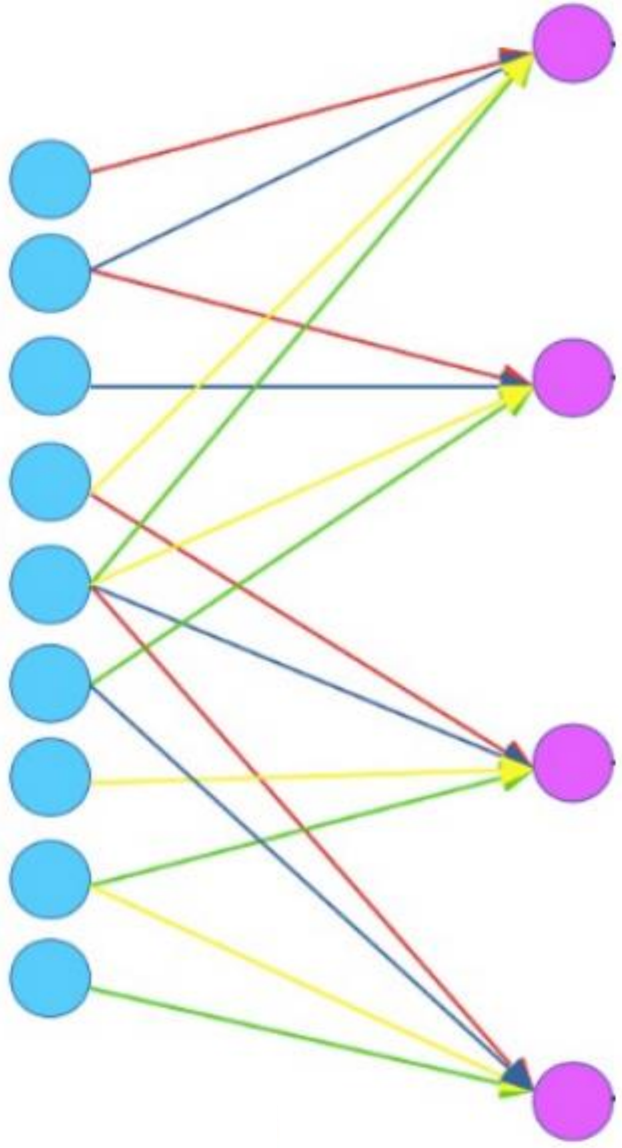


- If we instead used a fully connected layer, how many parameters would need to be learned?



Convolutional Layer: Parameters to Learn

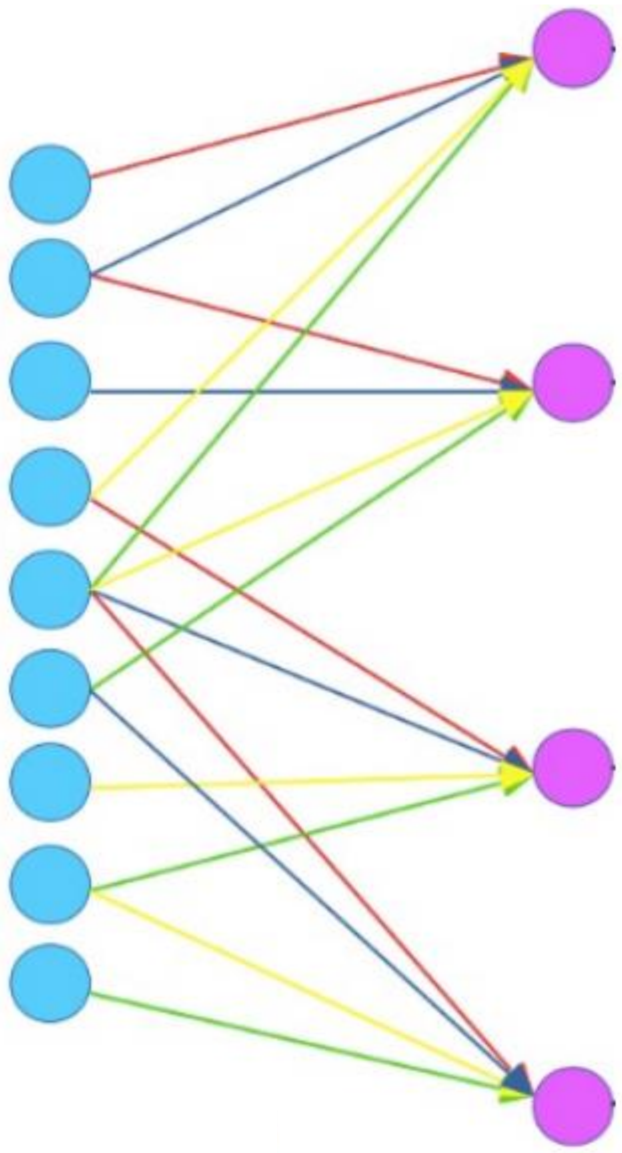
- For the shown example, how many weights must be learned?
  - 4 (red, blue, yellow, and green values)
- If we instead used a fully connected layer, how many weights would need to be learned?
  - 36 (9 turquoise nodes x 4 magenta nodes)
- For the shown example, how many parameters must be learned?
  - 5 (4 weights + 1 bias)
- If we instead used a fully connected layer, how many parameters would need to be learned?
  - **40 (36 weights + 4 bias)**



Convolutional Layer: Parameters to Learn

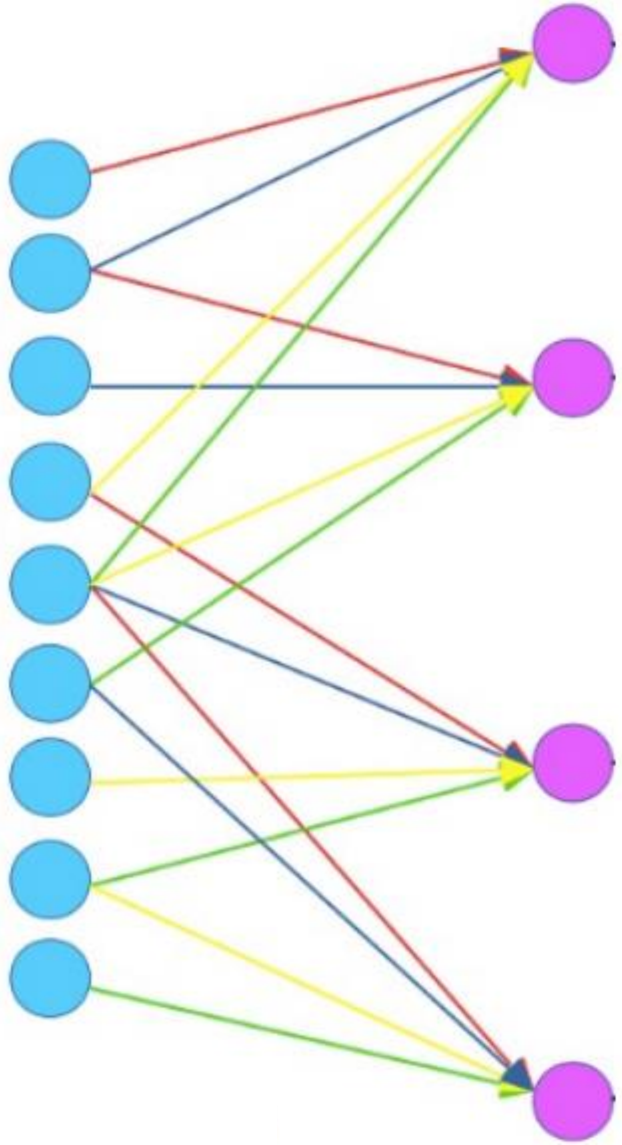
- **Parameter sharing** significantly reduces number of parameters to learn and so storage requirements

- **Sparse connectivity** (rather than full) also significantly reduces the number of computational operations required

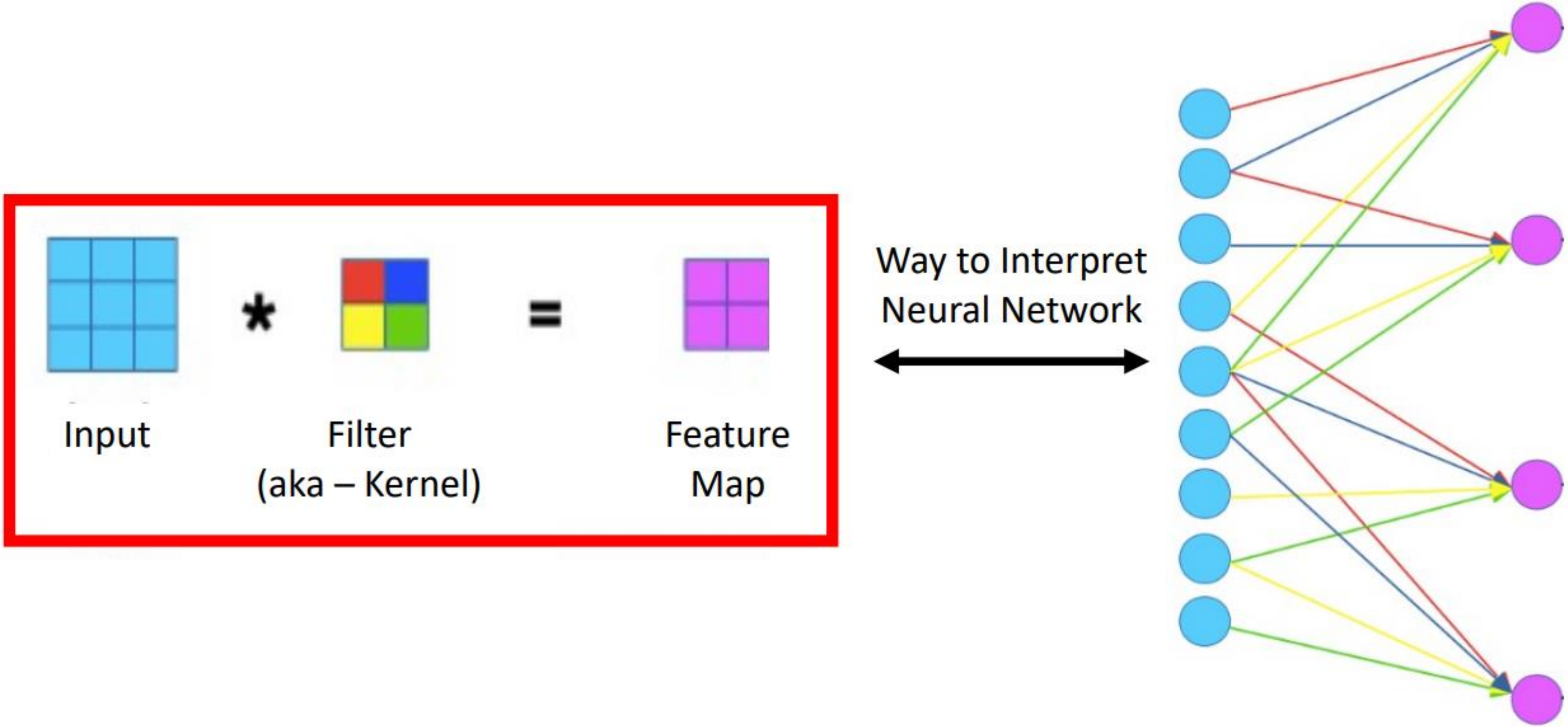


Convolutional Layer: Parameters to Learn

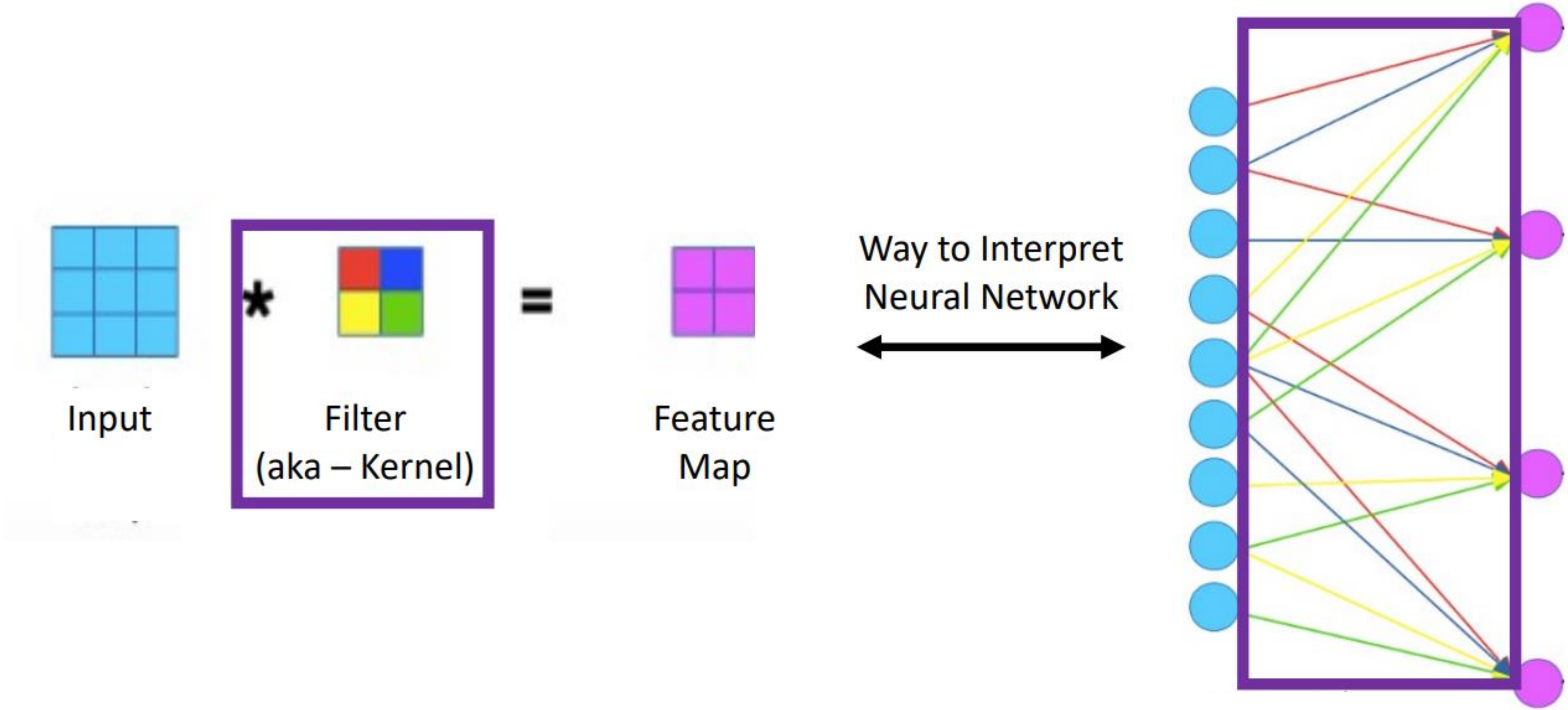
- **Neocognitron has hard-coded filter values… we will cover models that learn the filter values**



Convolutional Layer

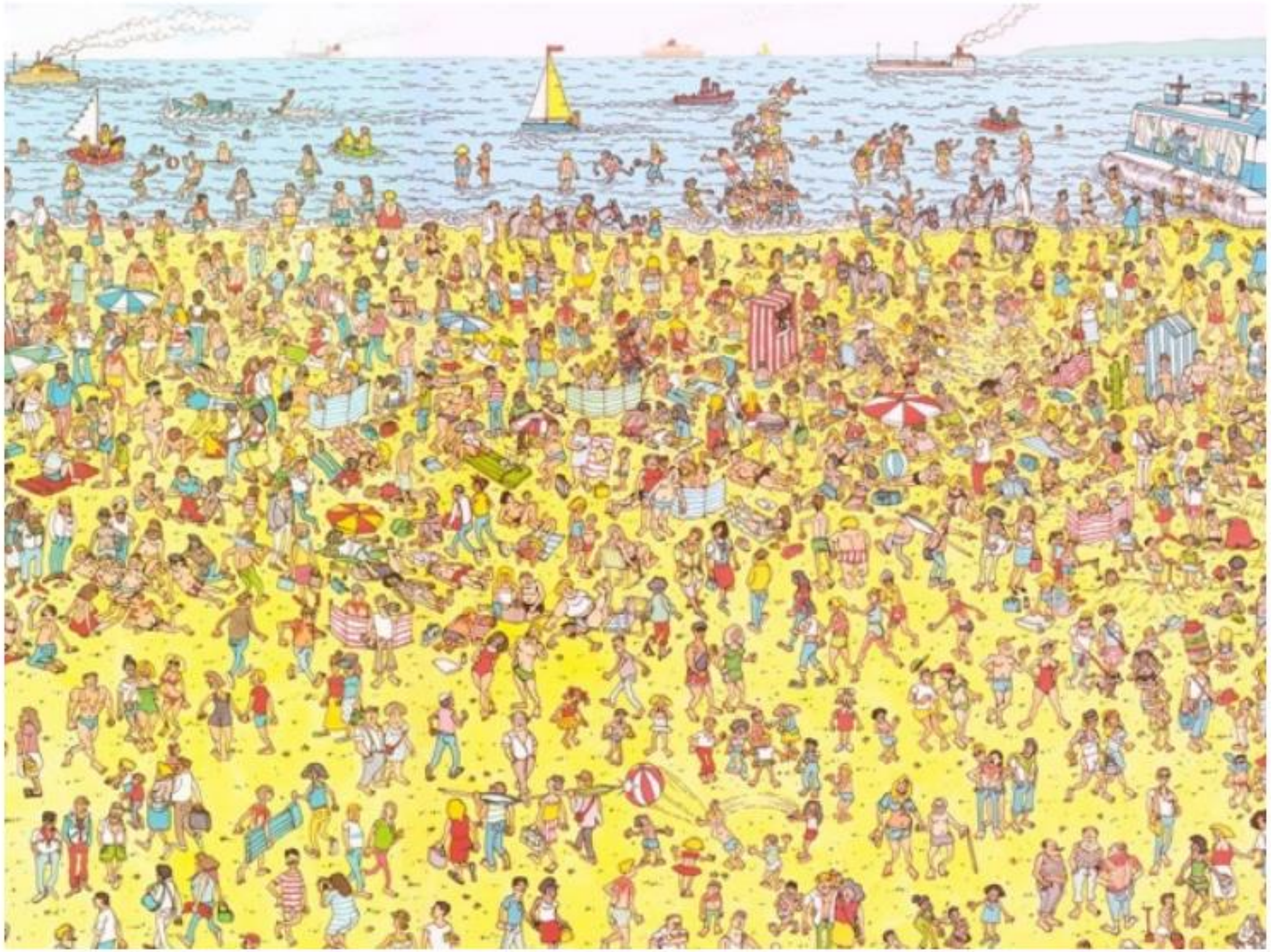


Convolutional Layer



Filter: What Does It Do?

Filter



Filter: What Does It Do?

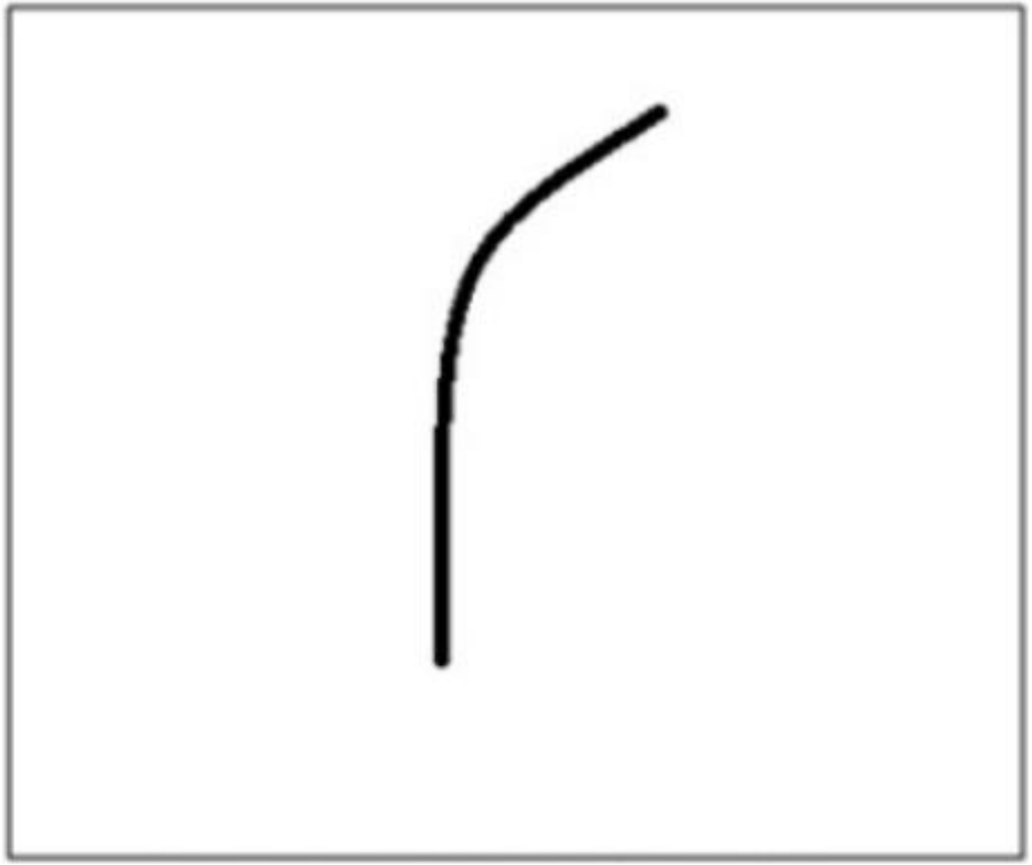
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- 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

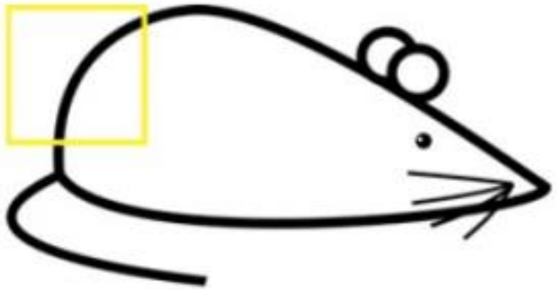


https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Filter: What Does It 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/

Filter: What Does It 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!!)**

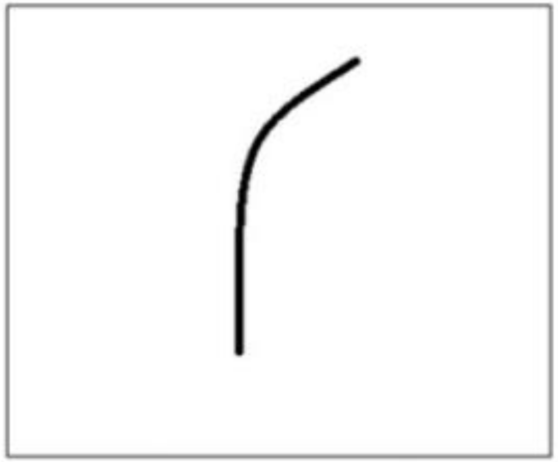
https://adeshpande3.github.io/A-Beginner%27s-Guide-To-Understanding-Convolutional-Neural-Networks/

Filter: What Does It Do?

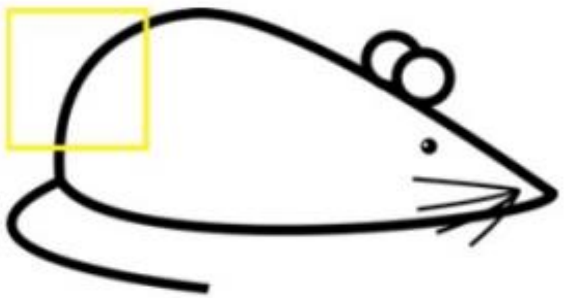
- e.g.,

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!)

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

Filters Detect Different Features

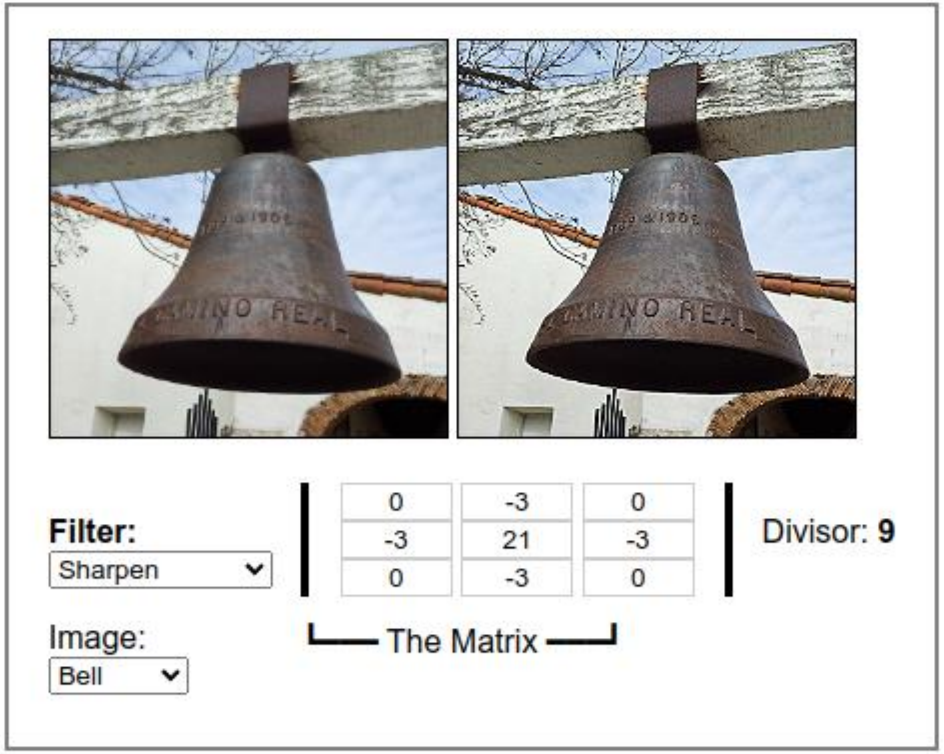
Operation	Filter	Convolved Image
Identity	$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$	
Edge detection	$\begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$	
	$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	
	$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$	

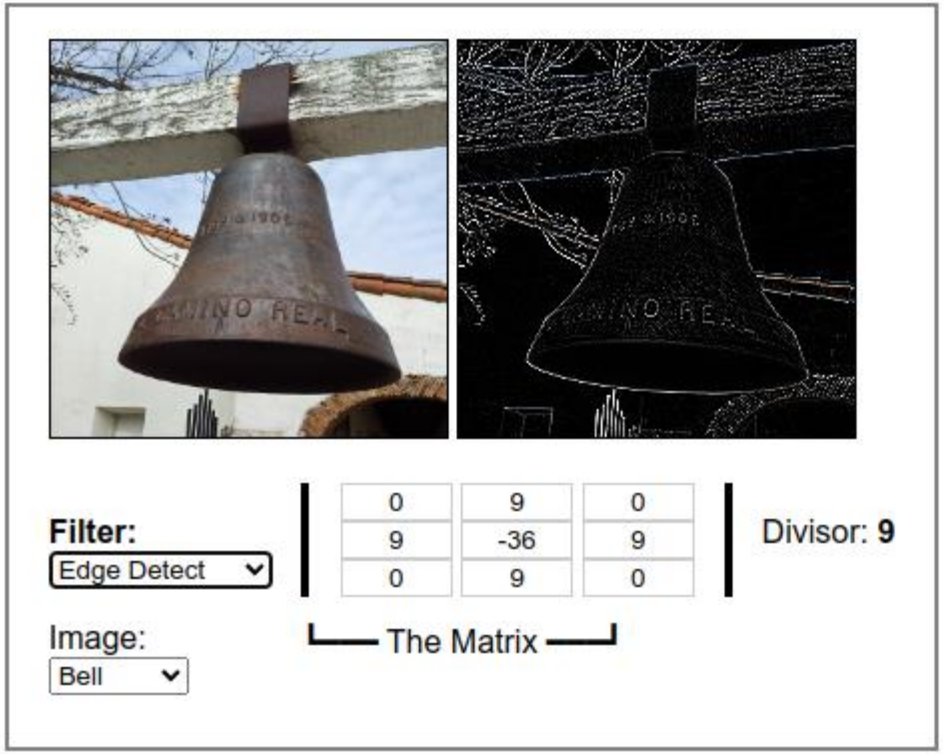
Operation	Filter	Convolved Image
Sharpen	$\begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$	
Box blur (normalized)	$\frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$	
Gaussian blur (approximation)	$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\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

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

Different Filters Detect Different Features

UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA





DEMO: https://beej.us/blog/data/convolution-image-processing/

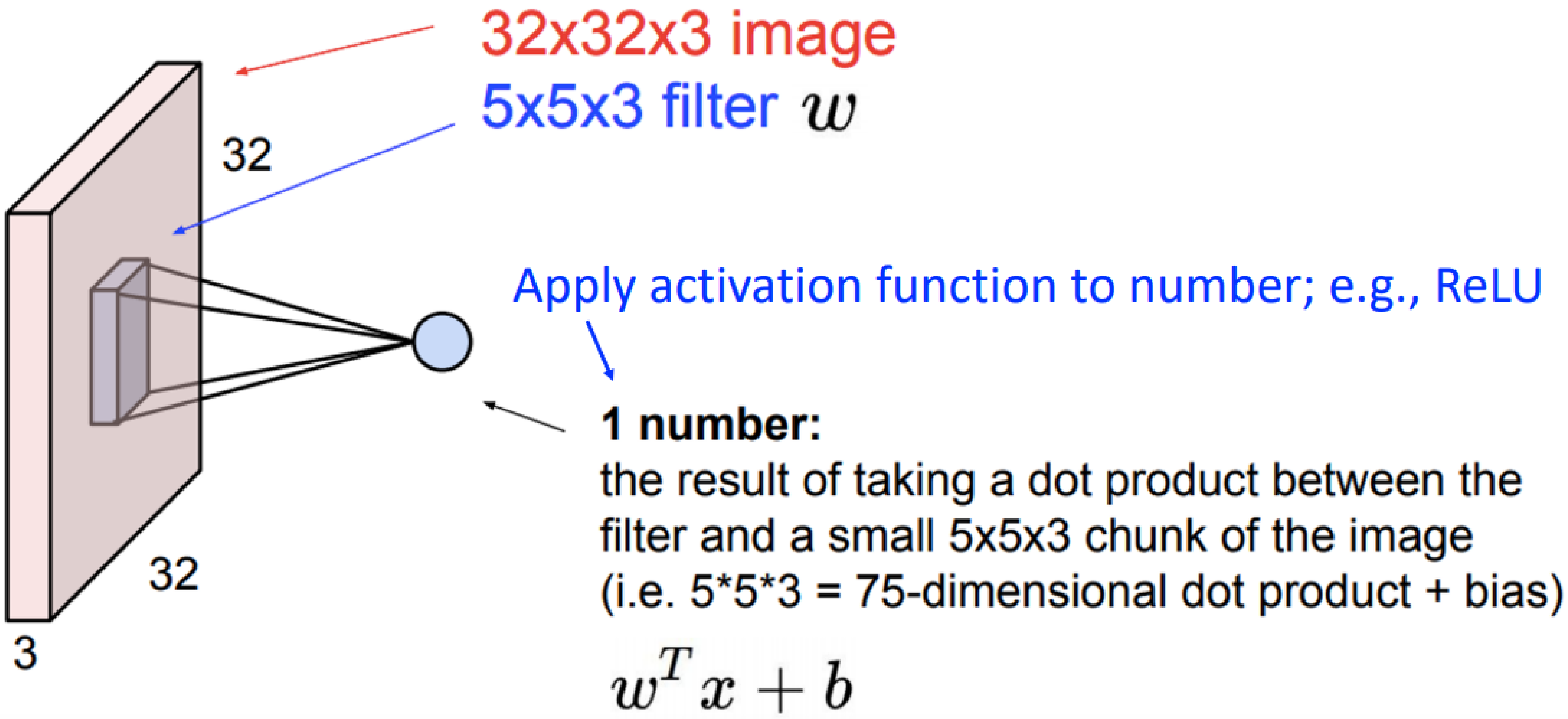
Group Discussion

1. How would you design a filter to “brighten” an image?

2. How would you design a filter to remove wrinkles/blemishes?

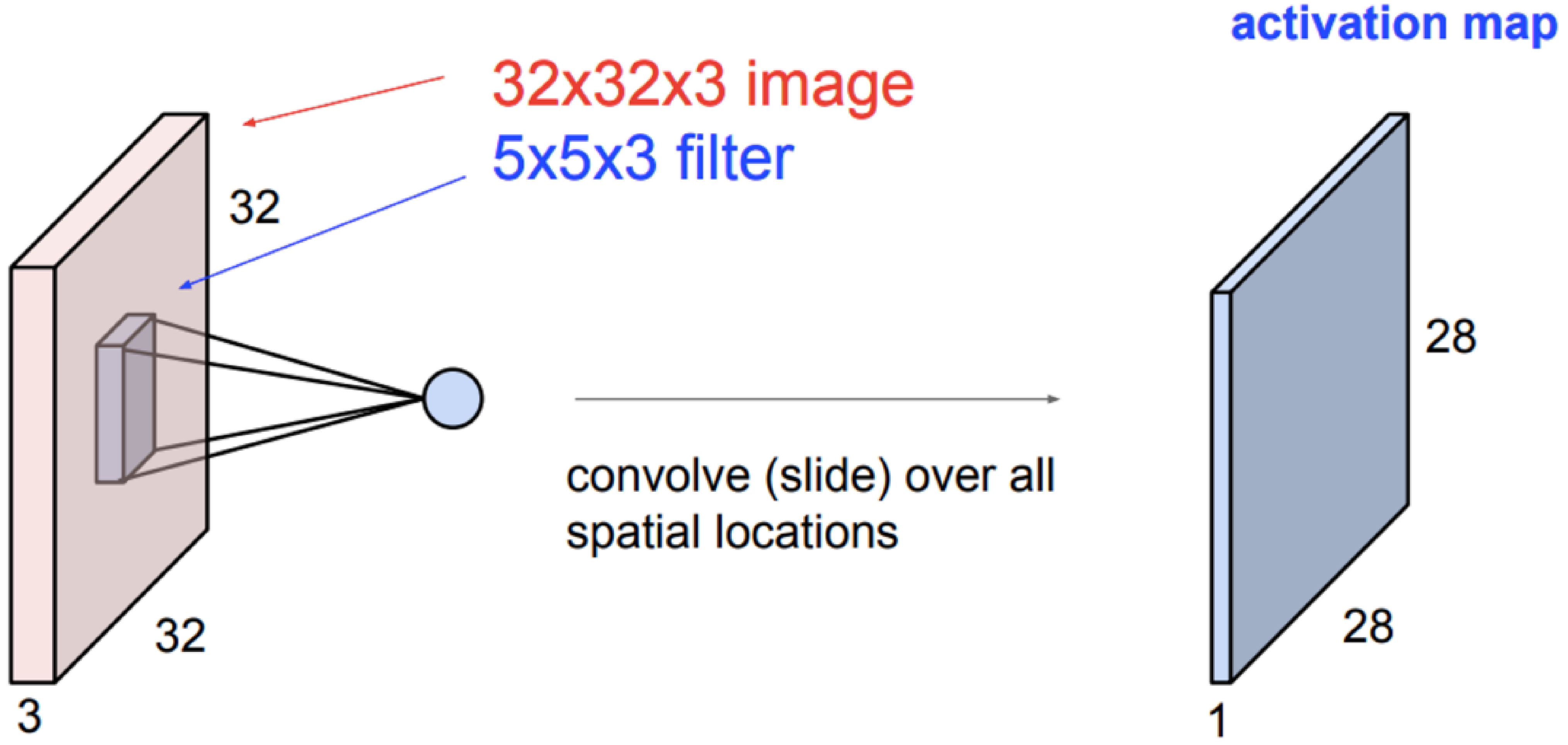
Convolutional Layer

- After applying the filter, introduce non-linearity



Convolutional Layer

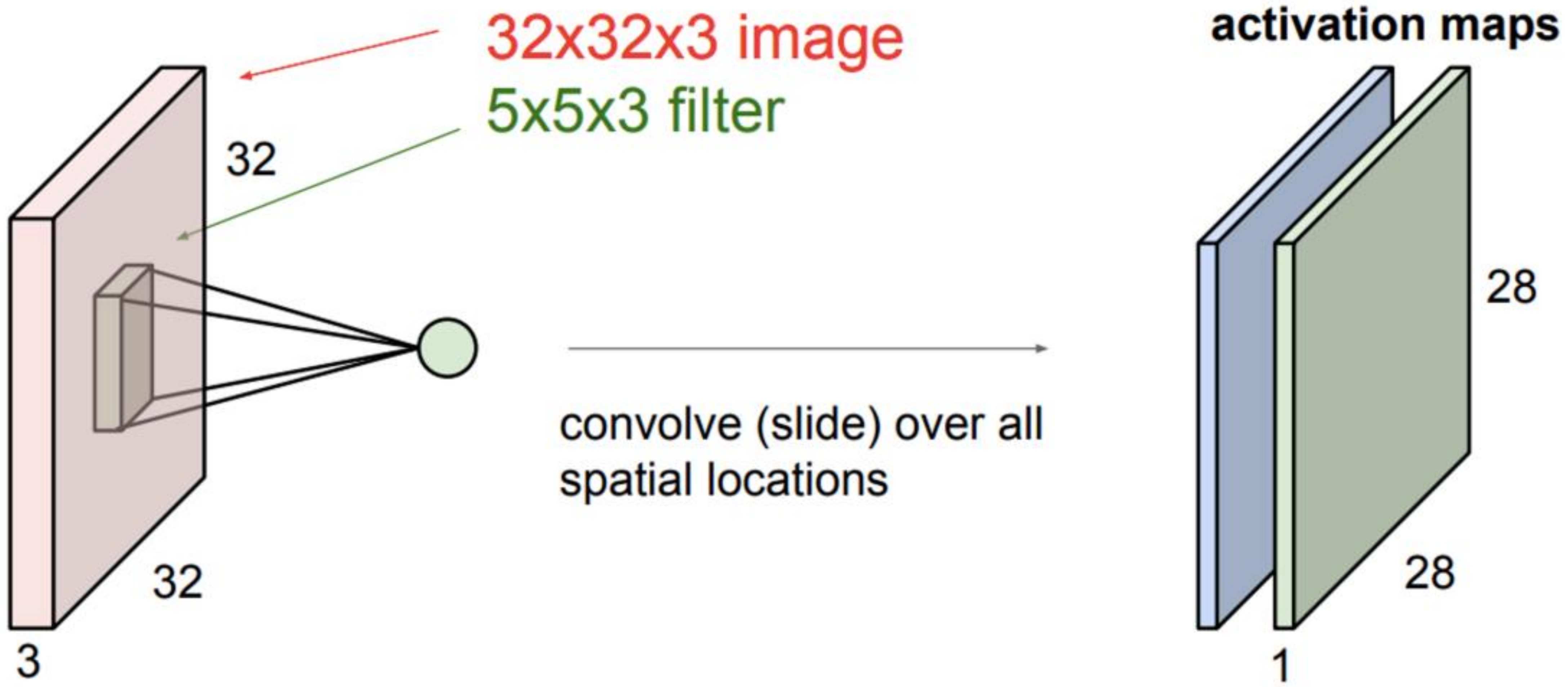
- Slide filter across input



Convolutional Layer

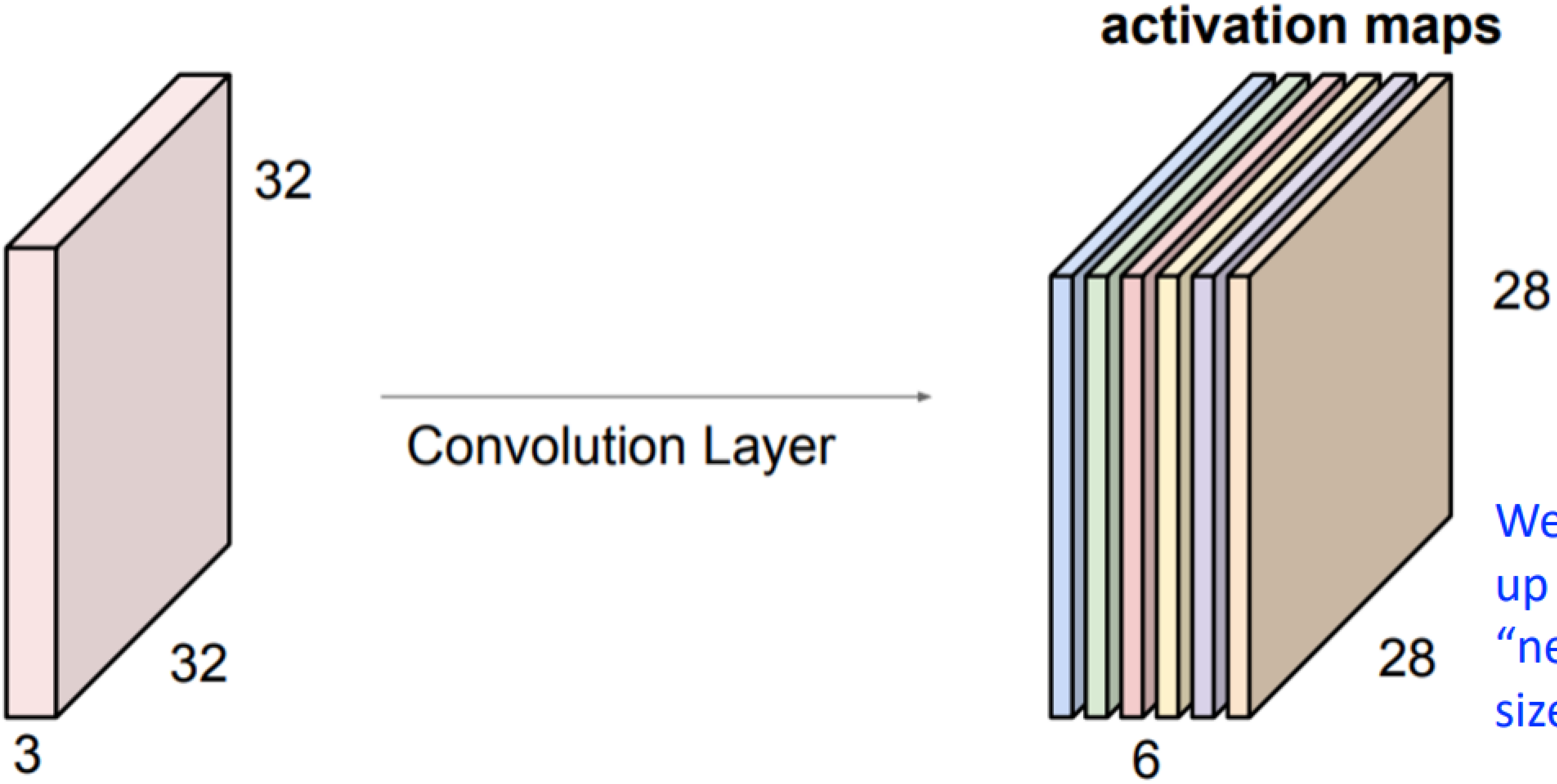
- Slide filter across input

Consider a second, green filter.



Convolutional Layer

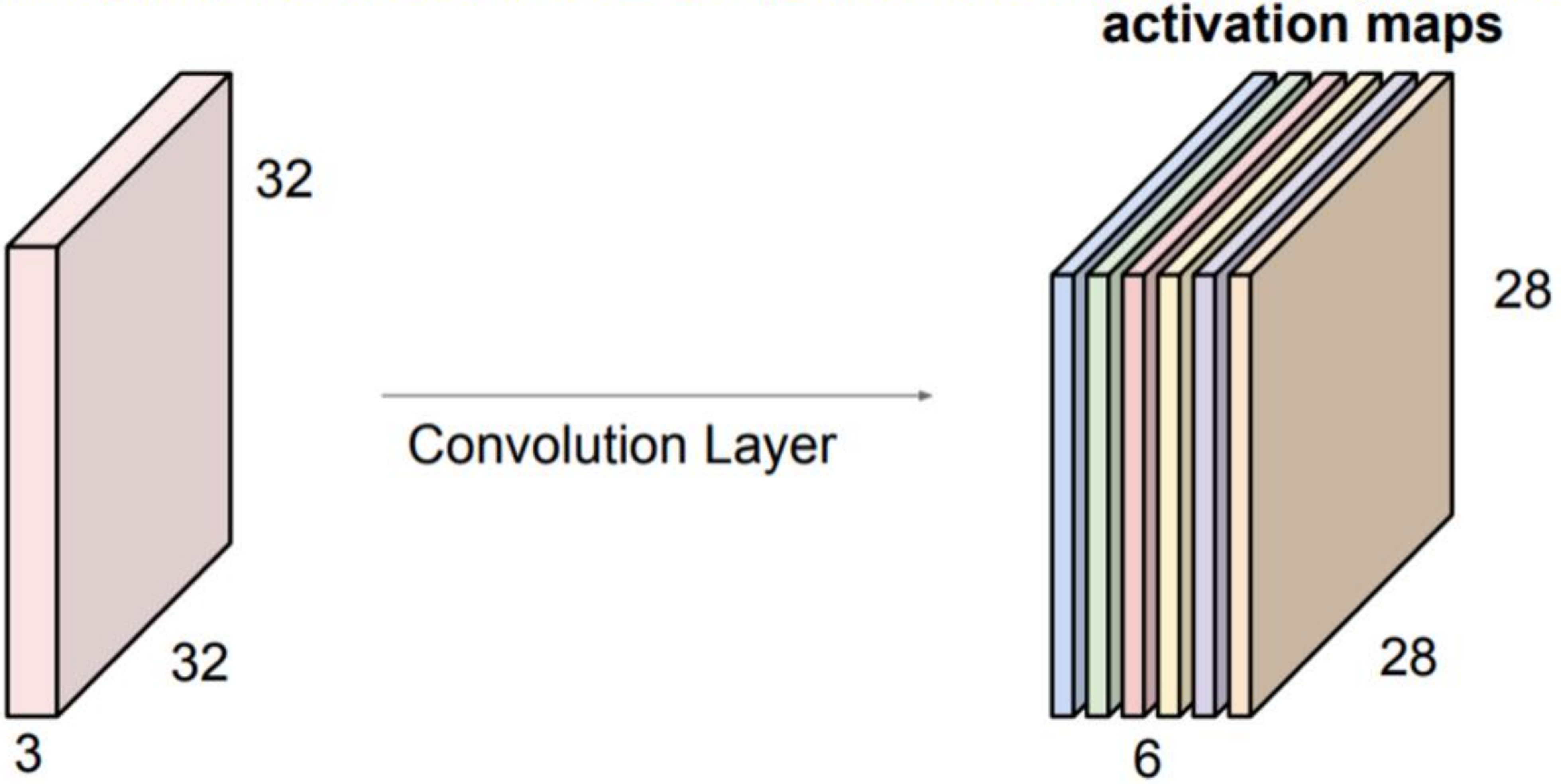
if we had 6 5x5 filters, we'll get 6 separate activation maps:



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

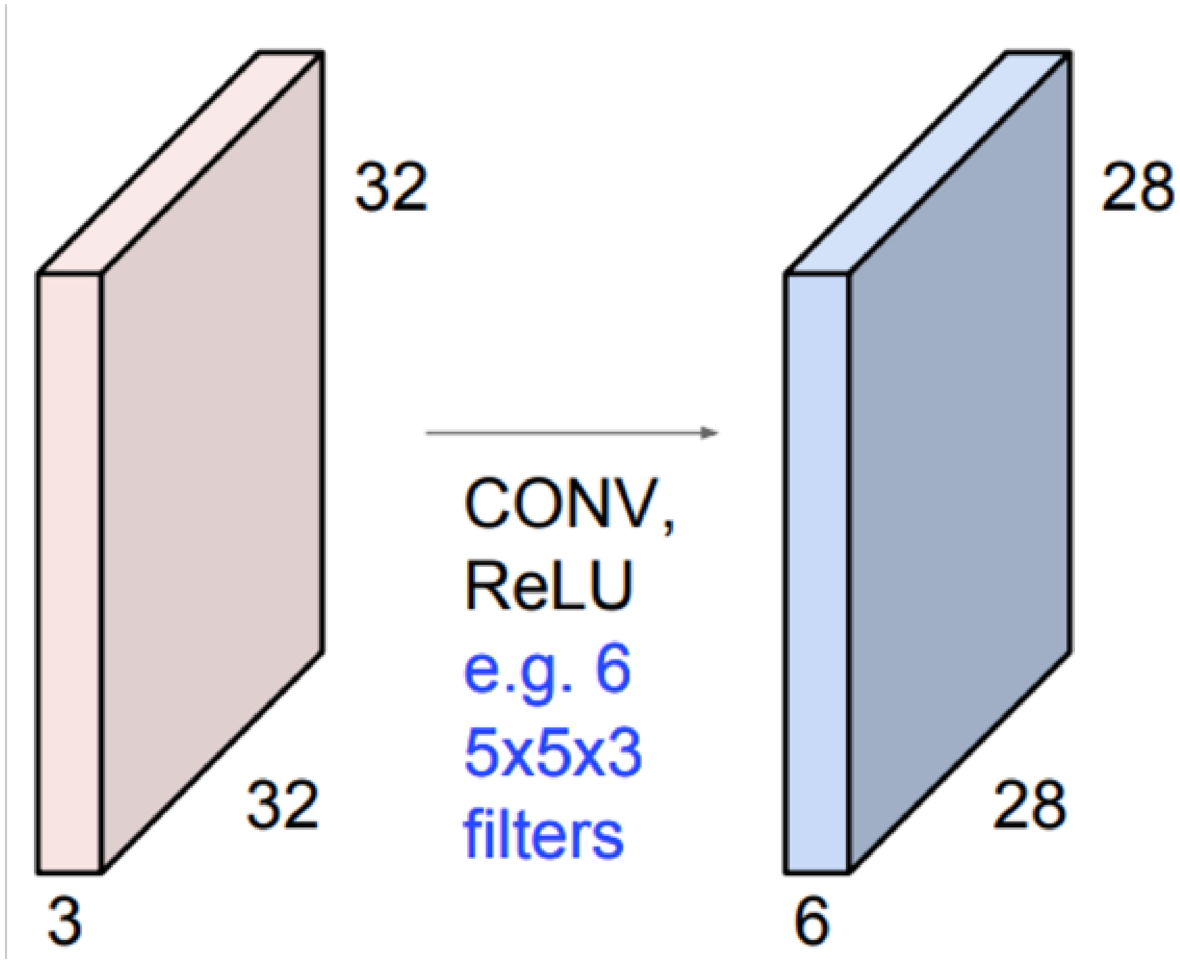
Convolutional Layer: Parameters to Learn

Parameters: bank of filters and biases used to create the activation maps (aka – feature maps)



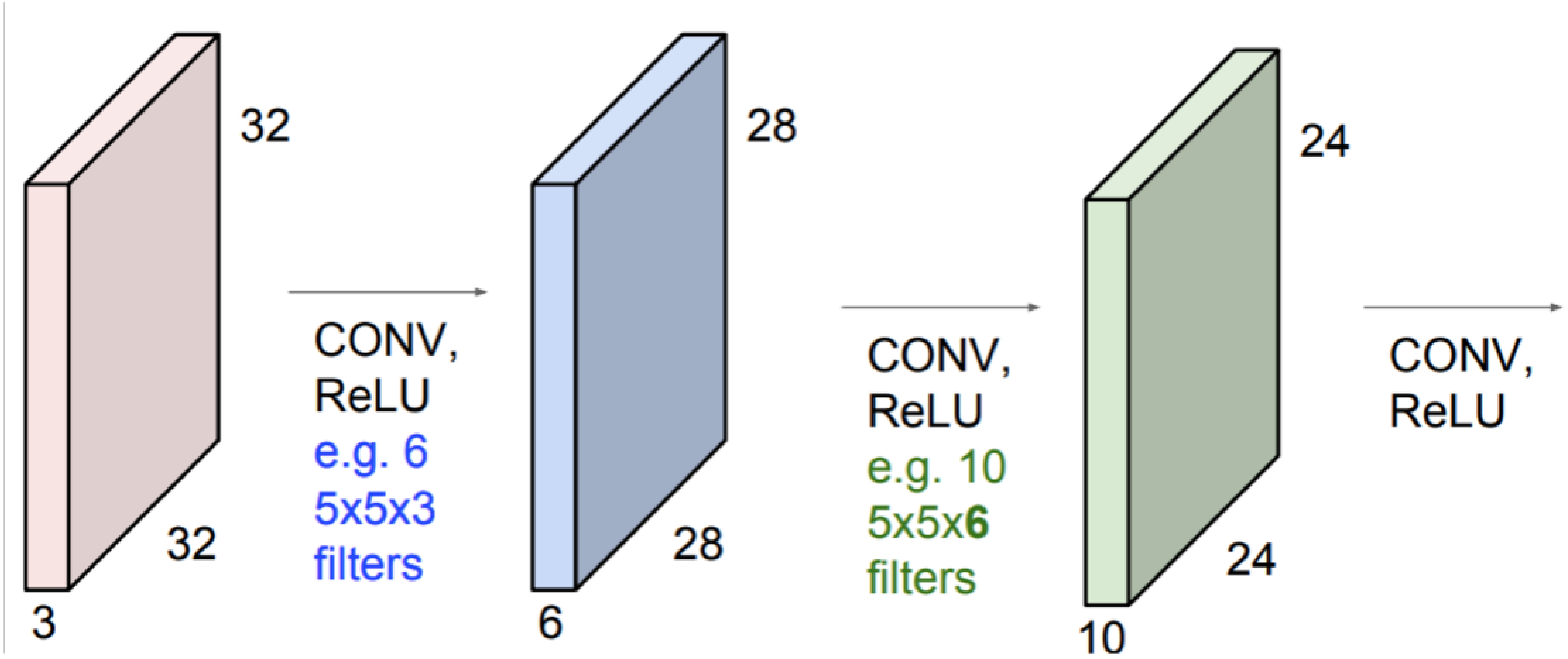
Convolutional Layers Stacked

Can then stack a sequence of convolution layers, interspersed with activation functions:



Convolutional Layers Stacked

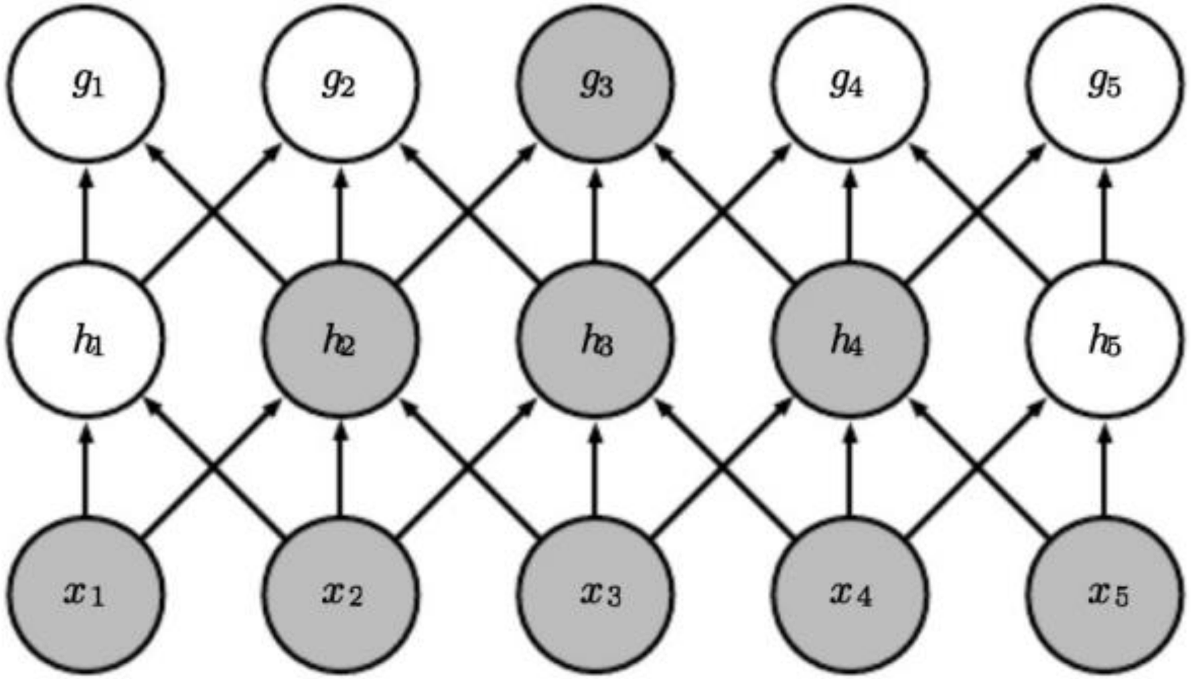
Can then stack a sequence of convolution layers, interspersed with activation functions:



Convolutional Layers Stacked

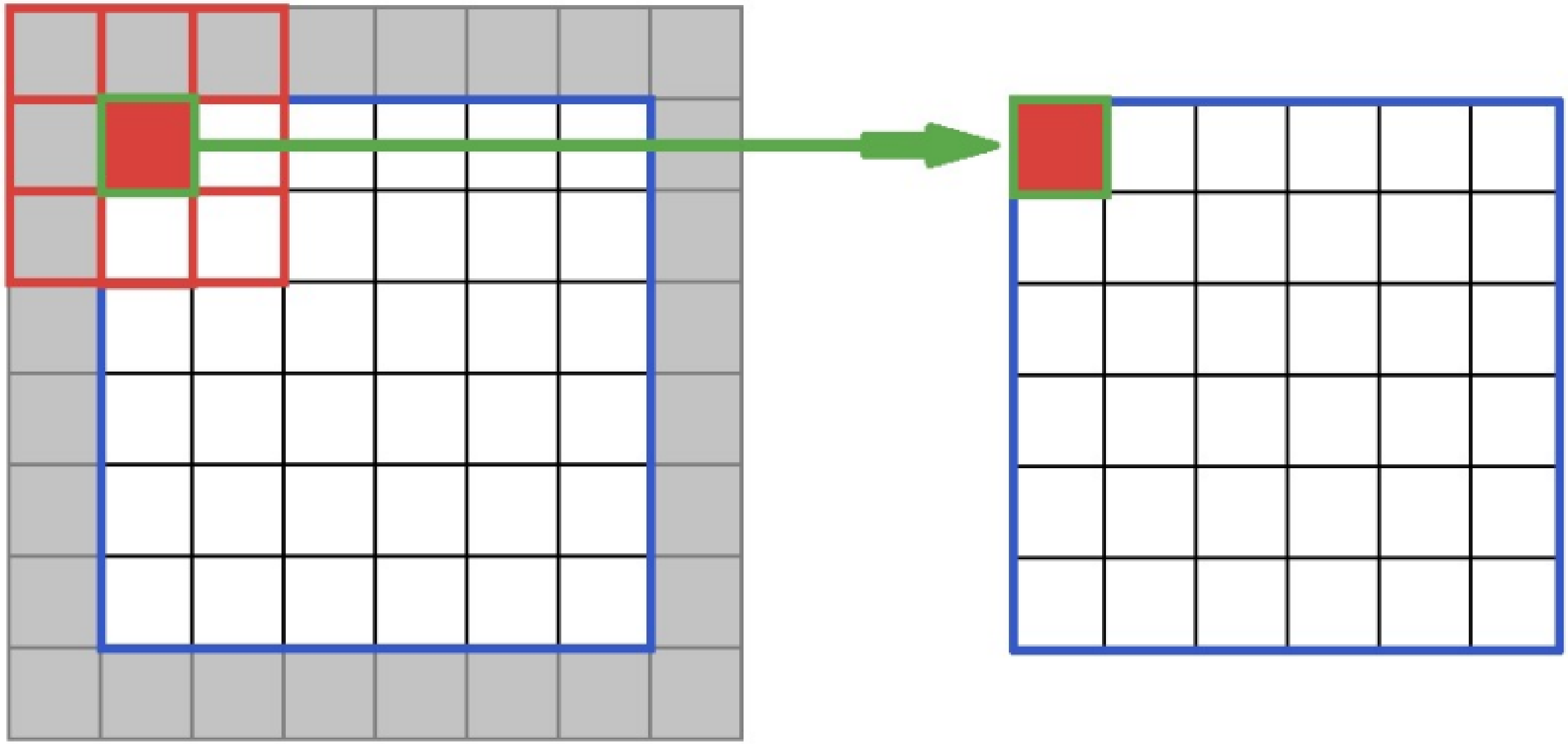
Can then stack a sequence of convolution layers, interspersed with activation functions:

Stacking many convolutional layers leads to identifying patterns in increasingly **larger regions of the input (e.g., pixel) space.**



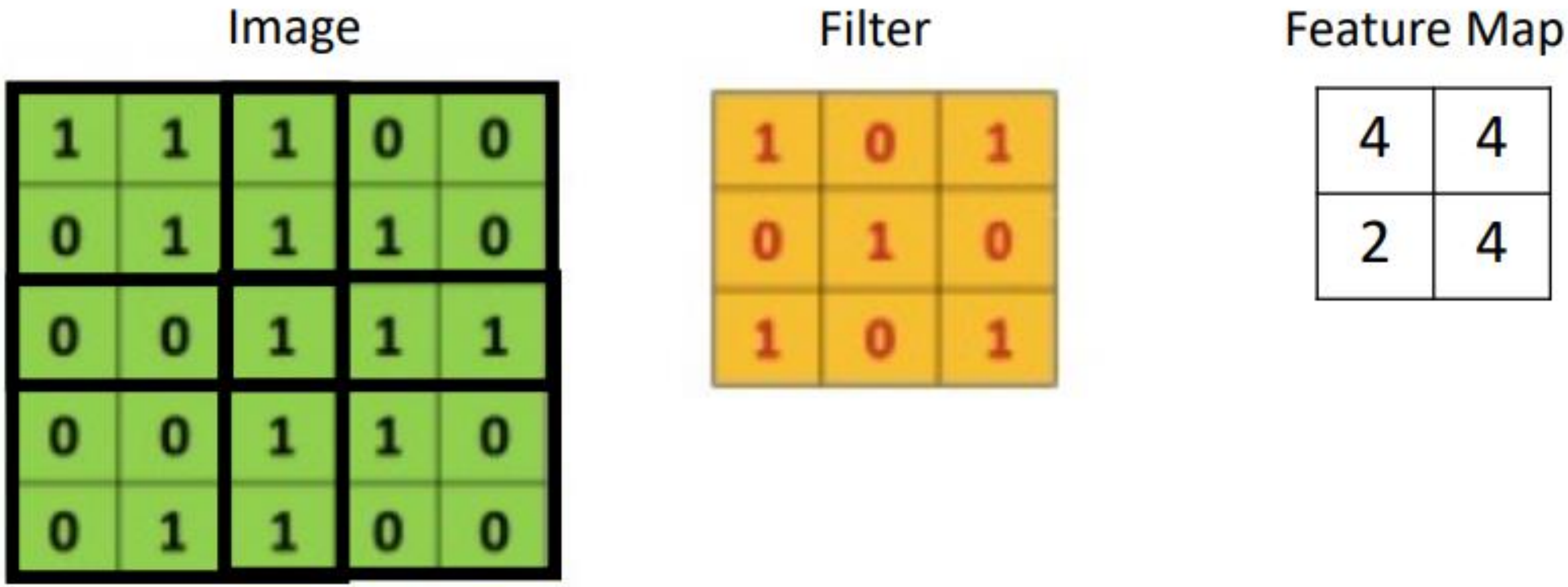
Convolution: Implementation Details

- **Padding:** add values at the boundaries to control output size



Convolution: Implementation Details

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



Parameters vs Hyperparameters in Convolutional Layers

- Hyperparameters:
  - ?
  - ?
  - ?

- Parameters:
  - ?
  - ?

Parameters vs Hyperparameters in Convolutional Layers

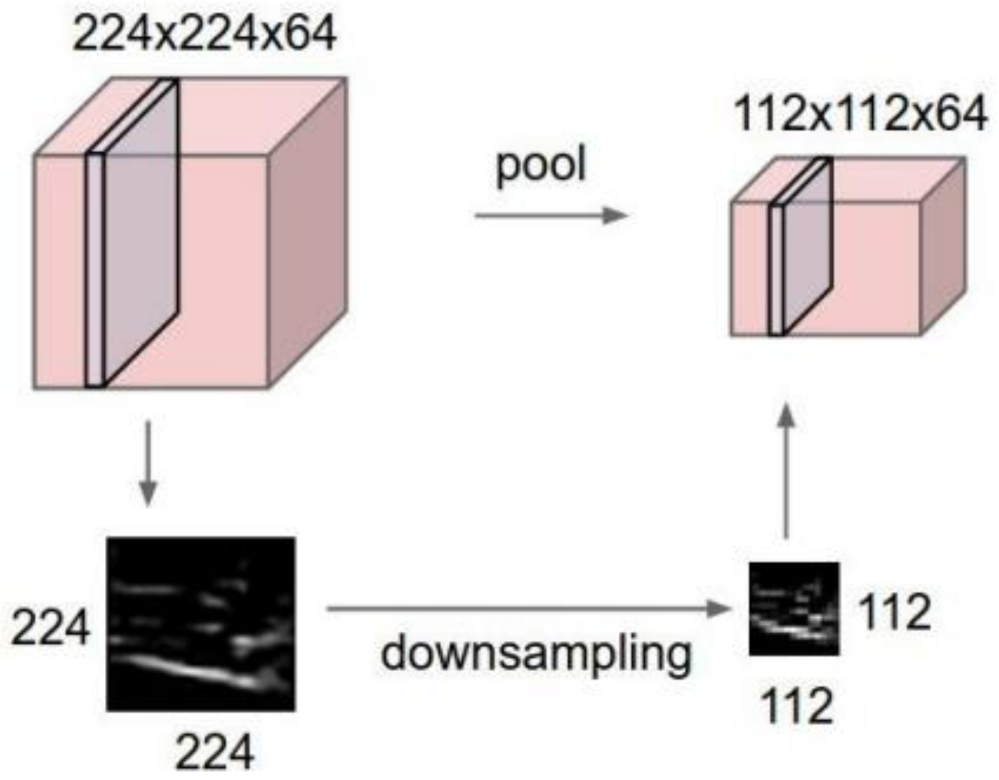
UNIVERSIDADE CATÓLICA PORTUGUESA
BRAGA

- Hyperparameters:
  - **Number of filters** and their dimensions (**height and width**)
  - **Stride**
  - **Padding type**

- Parameters:
  - **Weights**
  - **Biases**

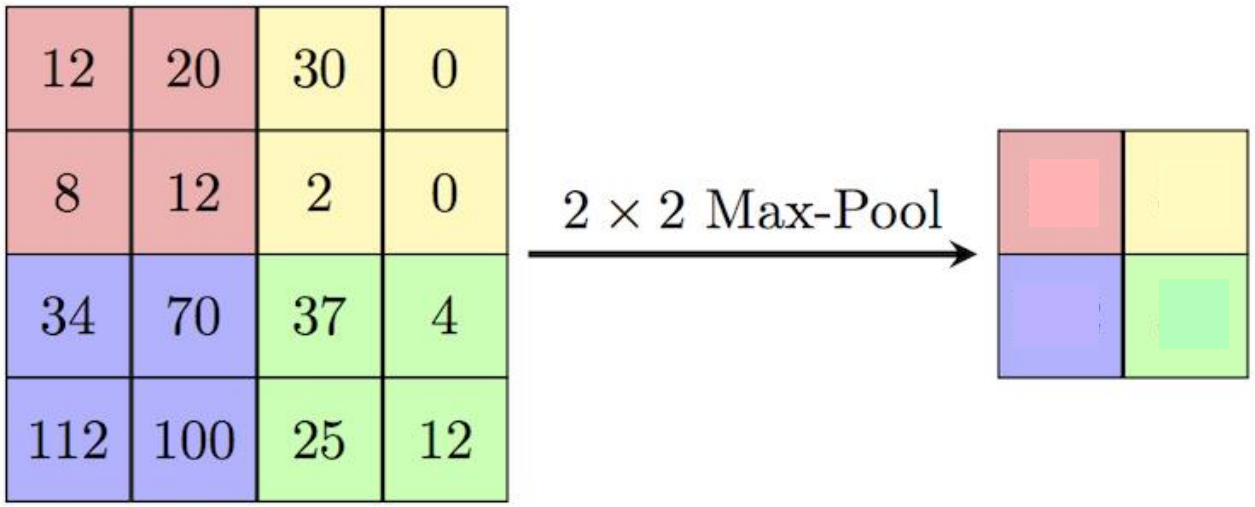
Pooling Layer

- Makes the representations **smaller and more manageable**
- Helps **retain important information** while discarding unnecessary details.
- Introduces some **invariance to small translations or distortions** (e.g., slight shifts in the image).
- Operates over each activation map independently:



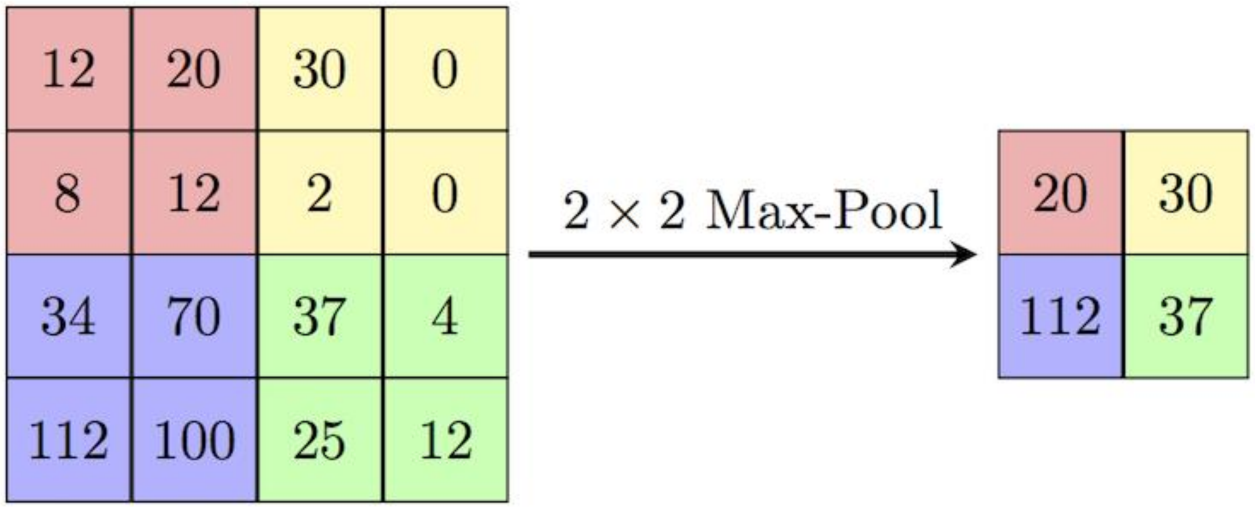
Types of Pooling

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



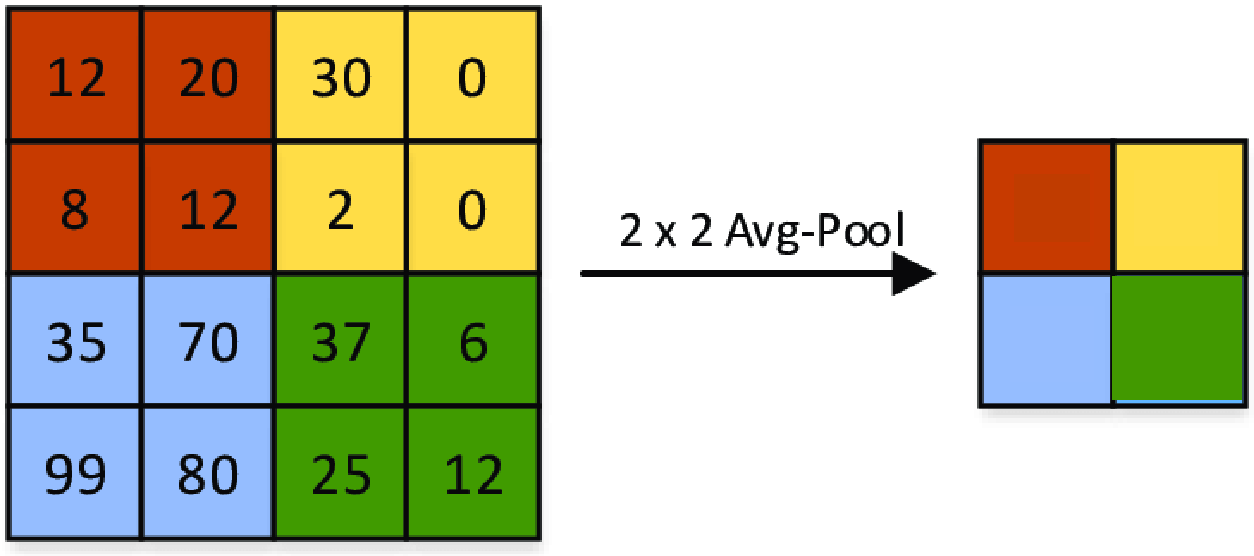
Types of Pooling

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



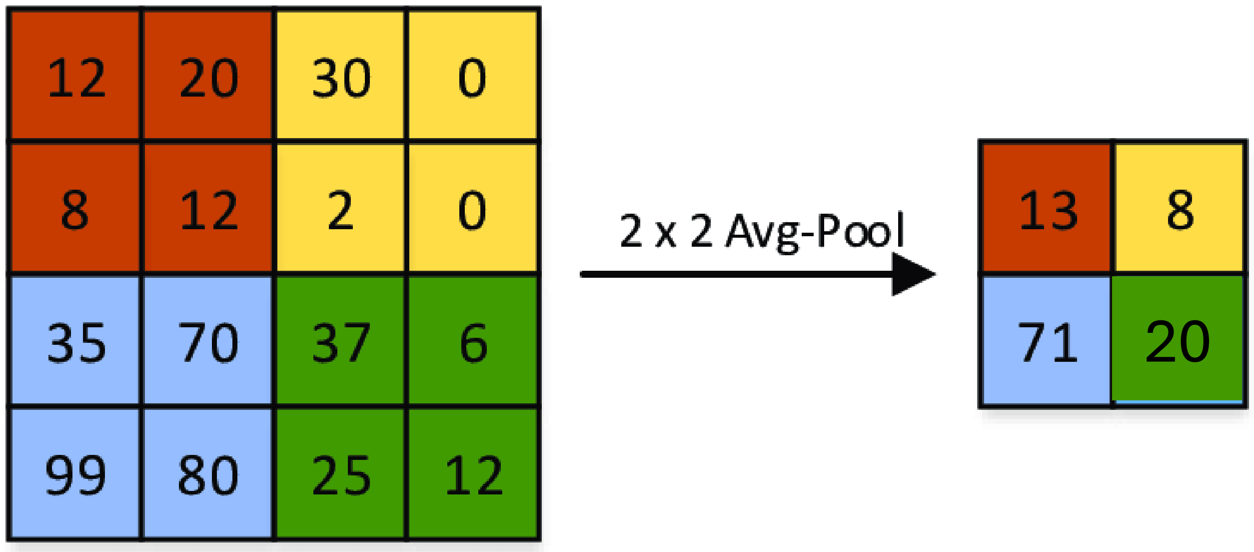
Types of Pooling

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



Types of Pooling

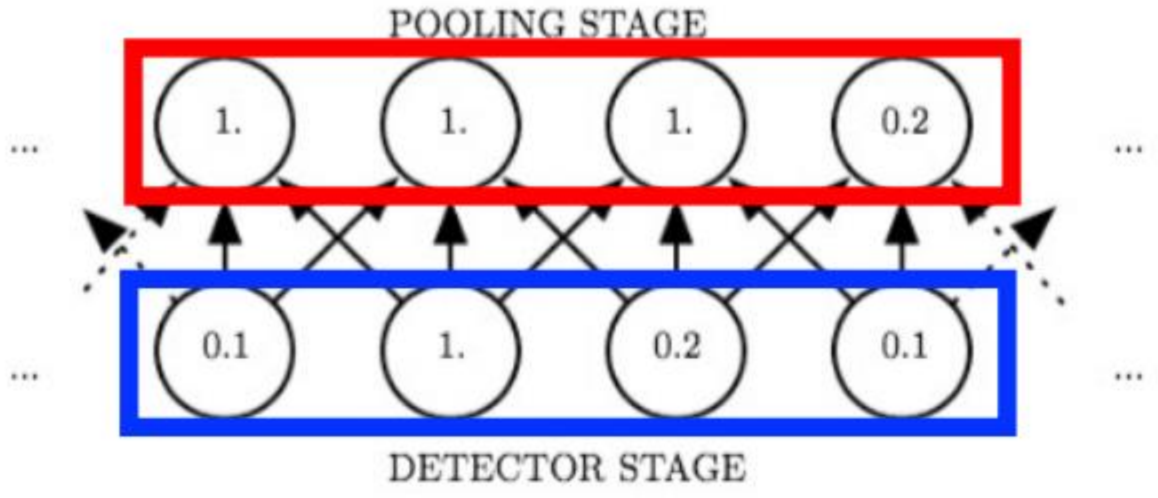
- **Average-pooling:** partitions input into a set of non-overlapping (generaly) rectangles and outputs the average value for each chunk

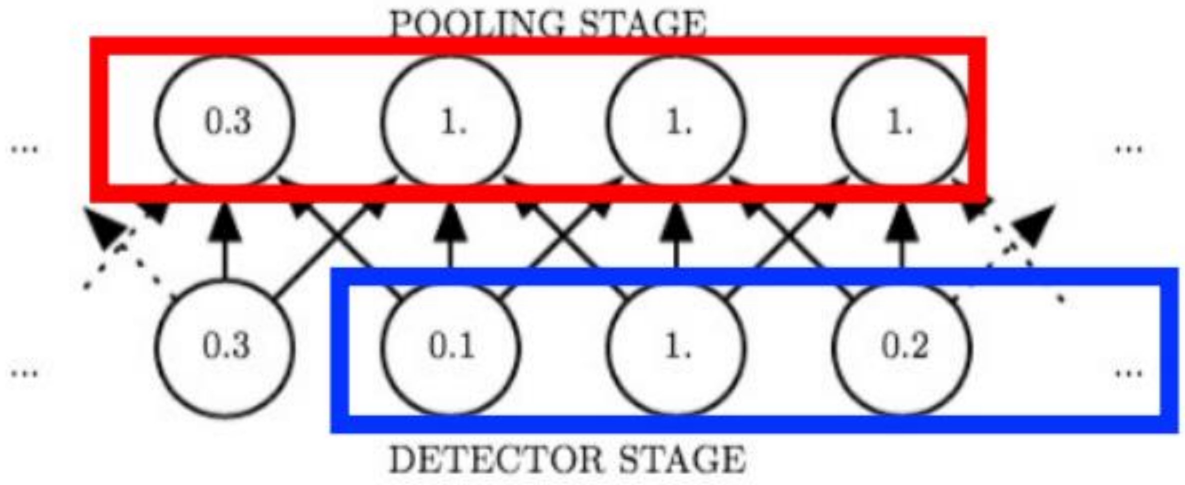


Pooling Layer

- Resilient to small translations

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



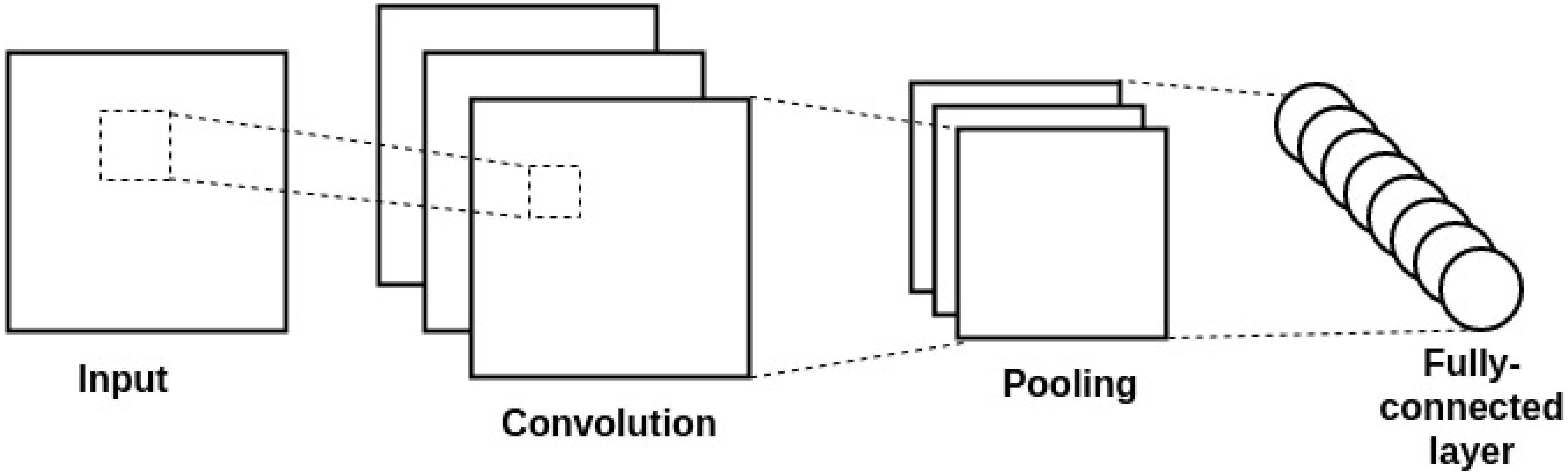


Pooling Layer: Benefits

- How many parameters must be learned?
  - None

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

Convolutional Neural Networks

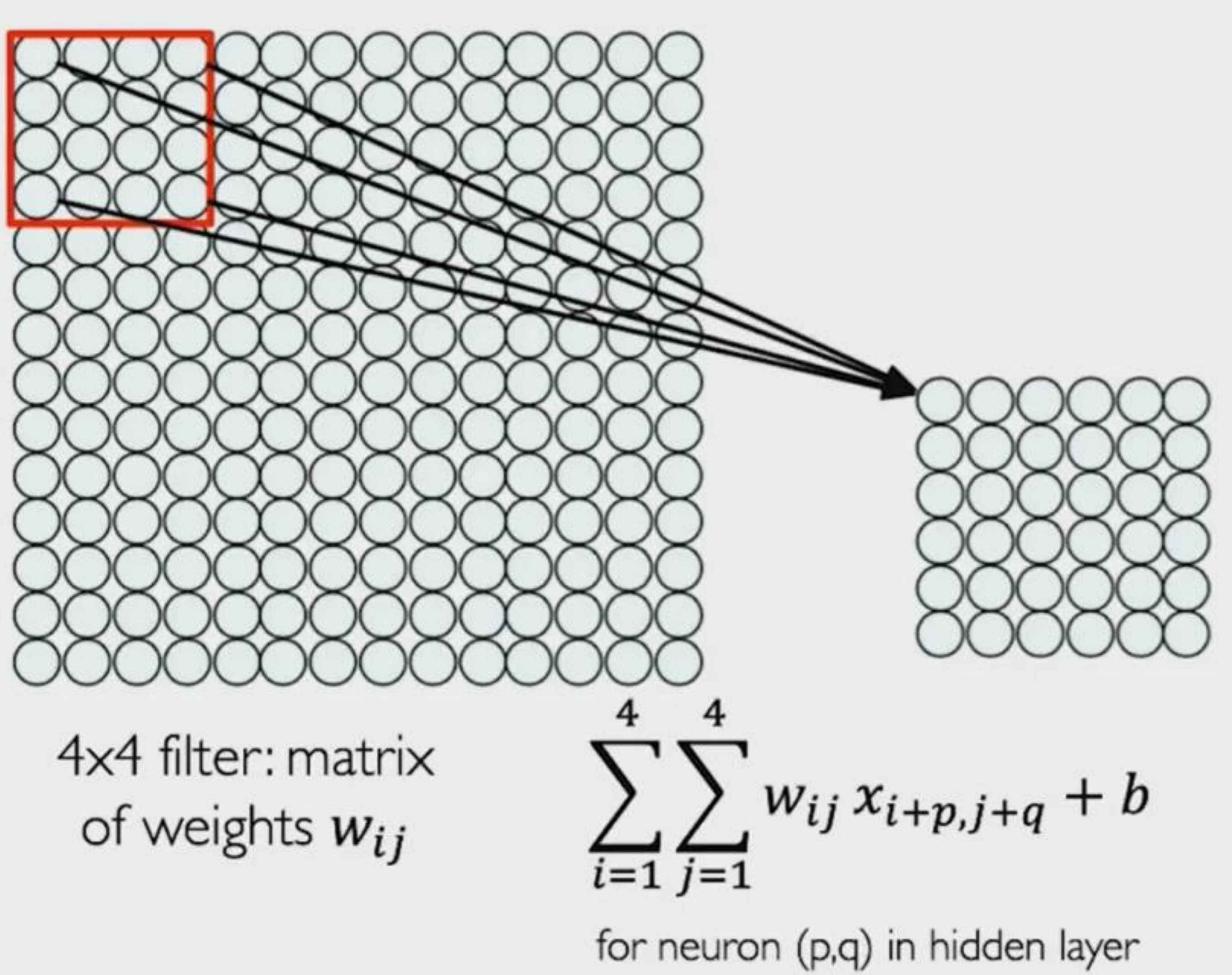


1. **Convolution:** apply filters to generate feature maps;
2. **Non-linearity:** Often ReLU;
3. **Pooling:** Downsampling operation on each feature map.

Train model with image data.
Learn weights of filters in convolutional layers.

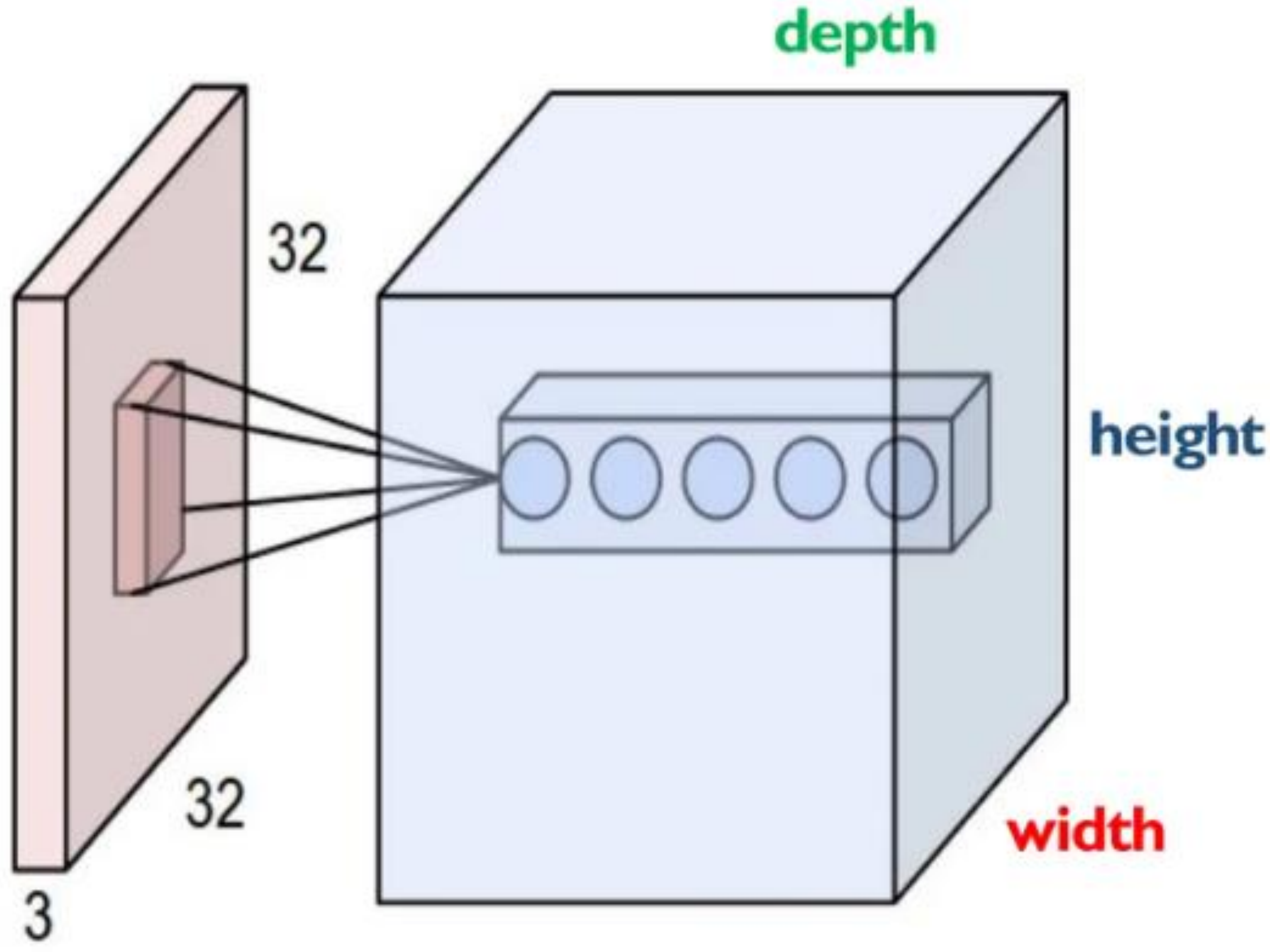
Convolutional Neural Networks

- For a neuron in hidden layer:
  - Take inputs from patch
  - Compute weighted sum
  - Apply bias



- Applying a window of weights
- Computing linear combinations
- Activating with non-linear function

Convolutional Neural Networks



Layer Dimensions:

$$h \times w \times d$$

where h and w are spatial dimensions d (depth) = number of filters

Stride:

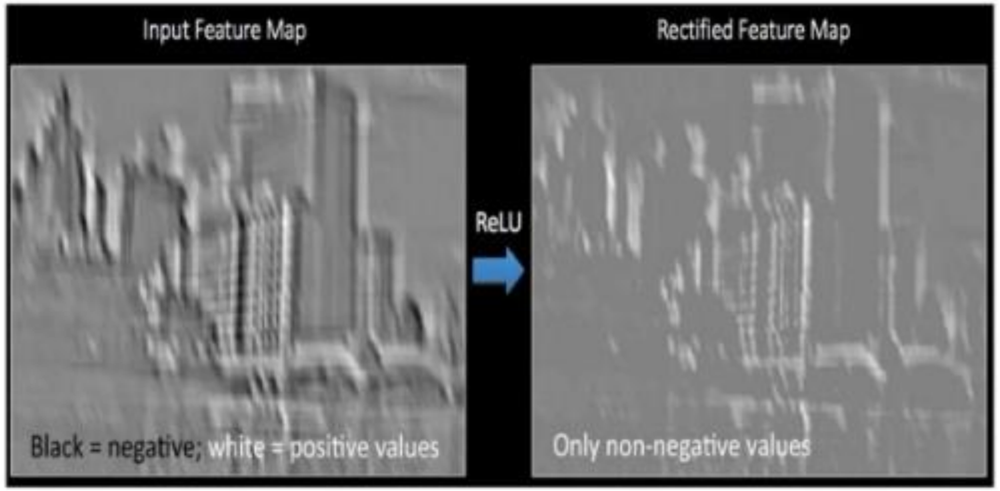
Filter step size

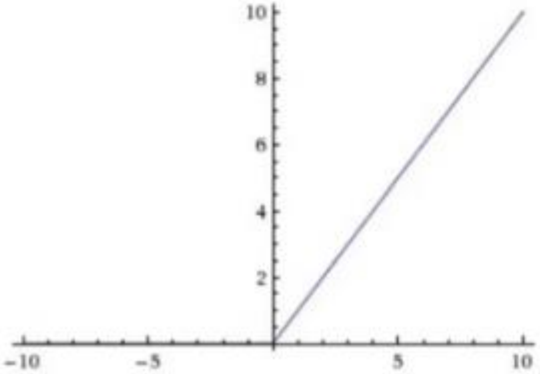
Receptive Field:

Locations in input image that a node is path connected to

Convolutional Neural Networks

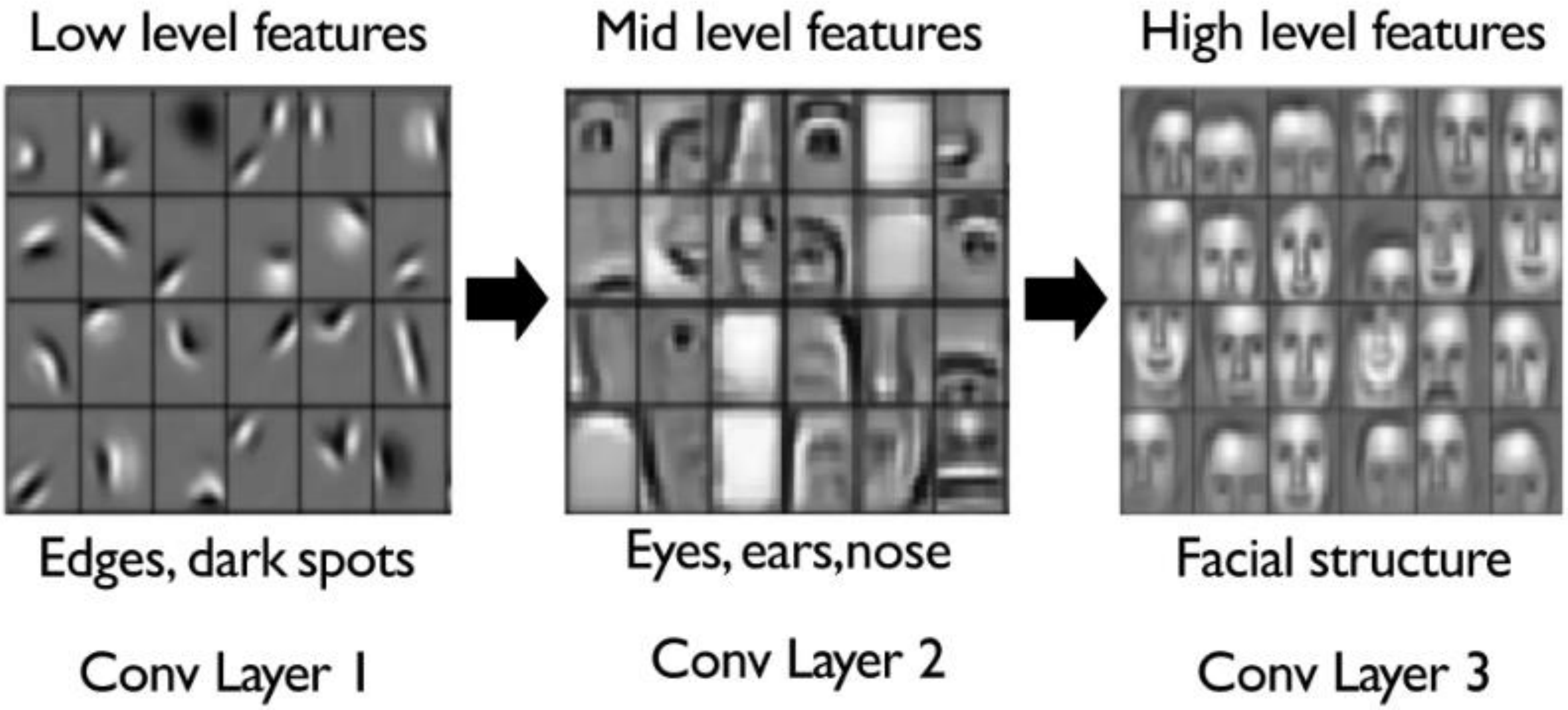
- Introducing non-linearity.
  - Apply after each convolutional layer
  - ReLU: pixel-by-pixel operation that replaces all negative values by zero.
    - Non-linear operation!



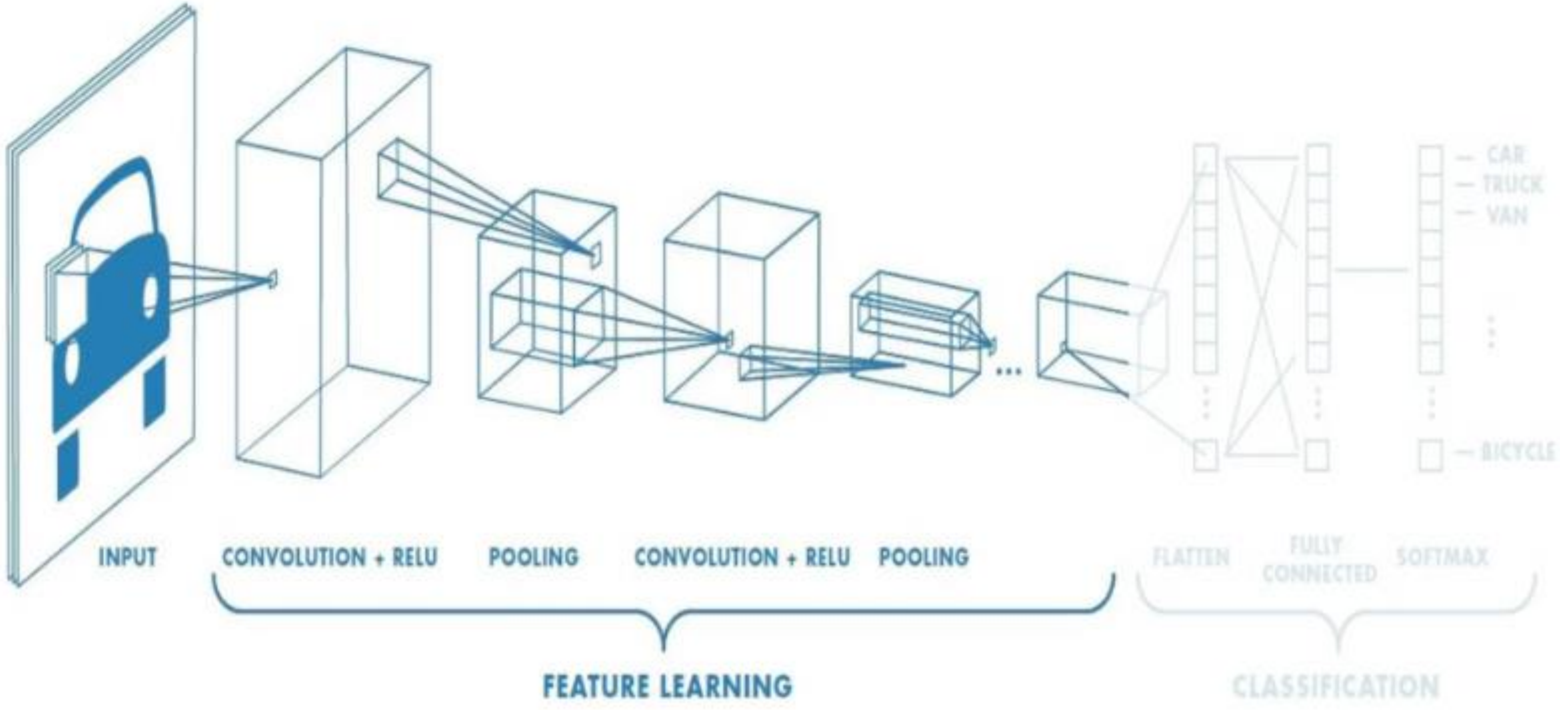


$$g(z) = \max(0, z)$$

Representation Learning in Deep CNNs

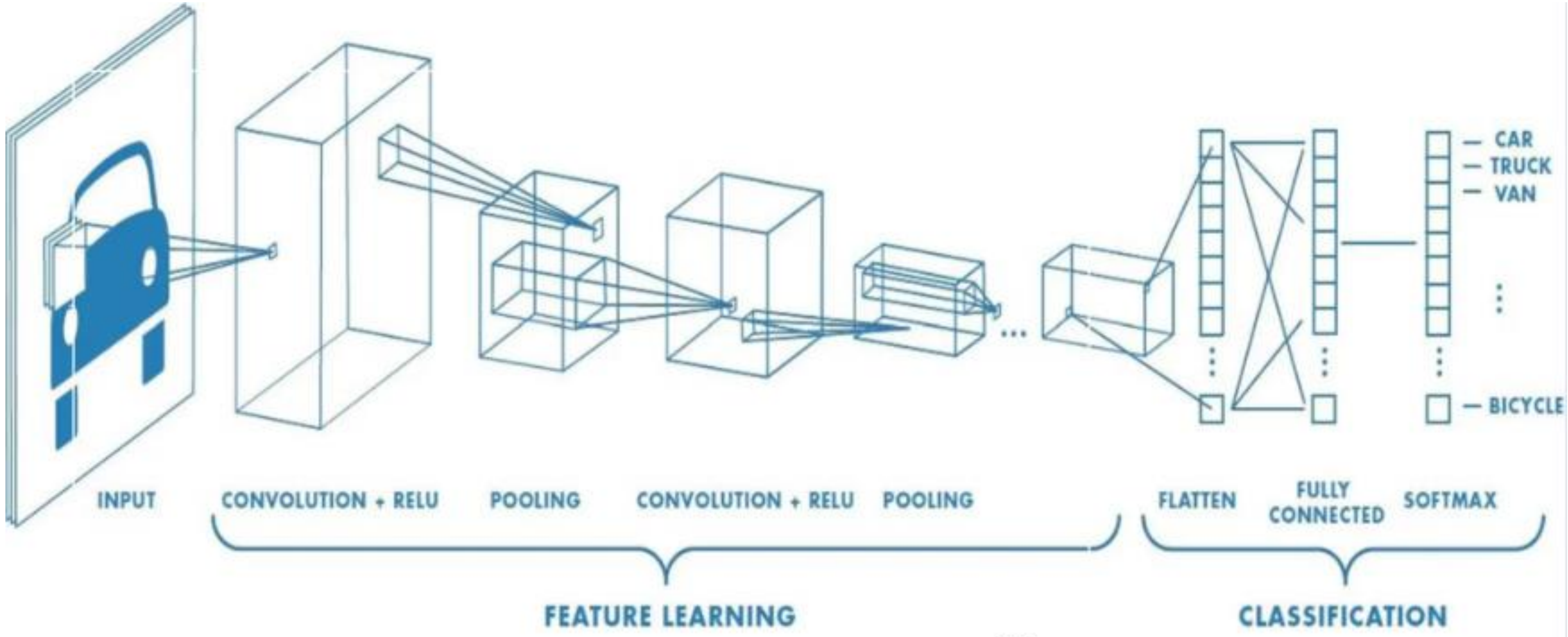


CNNs for Classification: Feature Learning



1. Learn features in input image through convolution
2. Introduce non-linearity through activation function (real worl data is non-linear!)
3. Reduce dimensionality and preserve spatial invariance with pooling

CNNs for Classification: Predictions



$$\mathrm{softmax}(y_i) = \frac{e^{y_i}}{\sum_j e^{y_j}}$$

- CONV and POOL layers output high-level features of input
- Fully connected layers uses these features for classifying input image
- Express output as probability of image belonging to a particular class