Convolutional Neural Networks

June 7, 2019

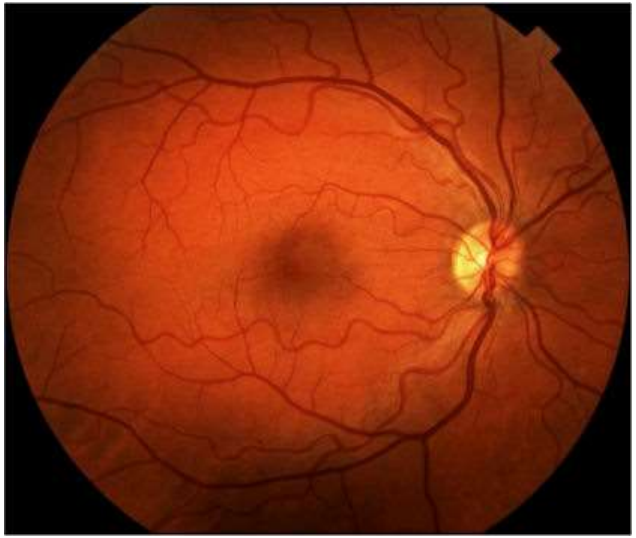
Block 2, Lecture 1
Applied Data Science
MMCi Term 4, 2019

Matthew Engelhard

Many slides created by Tim Dunn

Deep Learning for Image Analysis

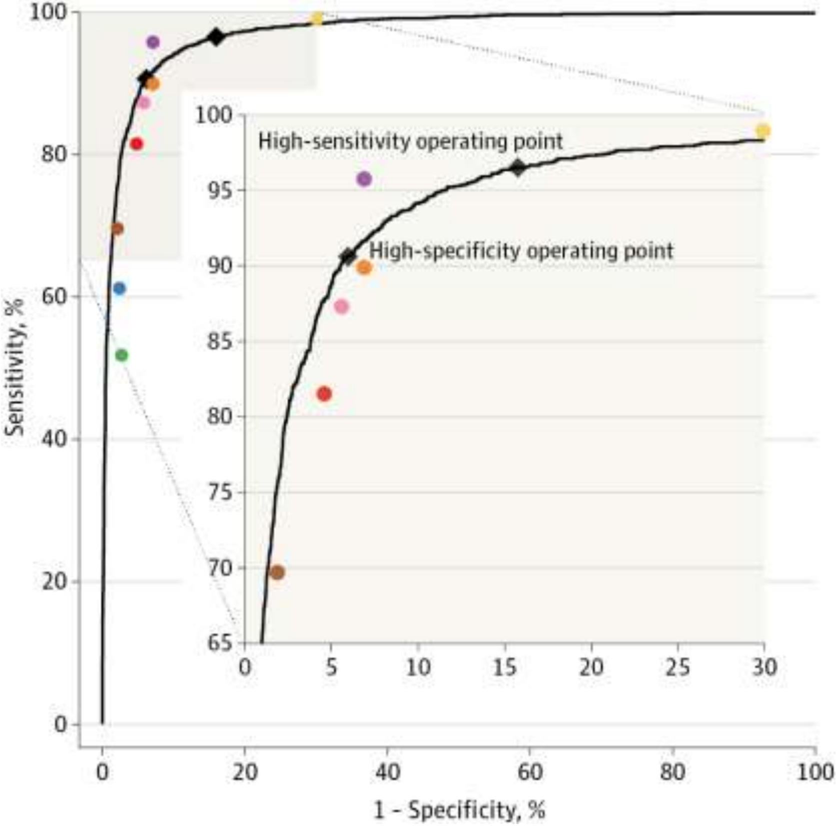
Diabetic Retinopathy
Classification



Healthy Retina

Unhealthy Retina

Deep Learning for Diabetic Retinopathy Classification



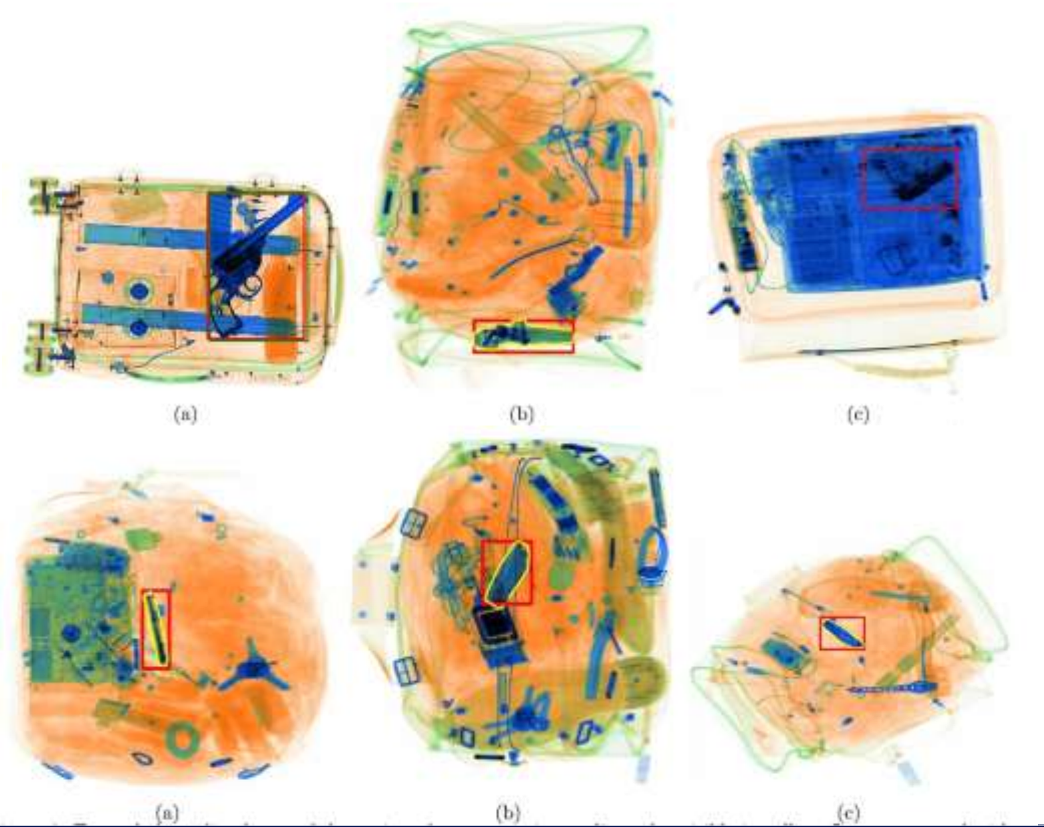
$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{total number of positives in the dataset}}$$

$$\text{specificity} = \frac{\text{number of true negatives}}{\text{total number of negatives in the dataset}}$$

Gulshan et al. *JAMA* (2016)

Deep Learning for Image Analysis

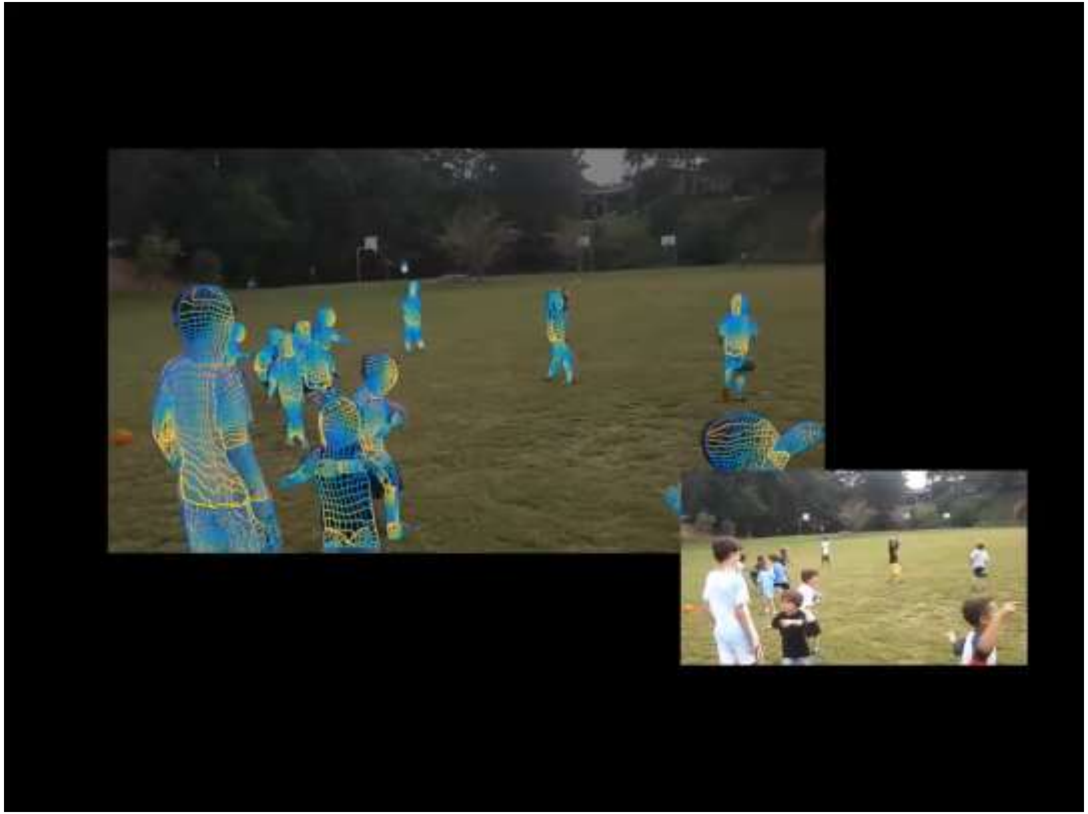
TSA
Screening



Liang et al. *SPIE* (2018)

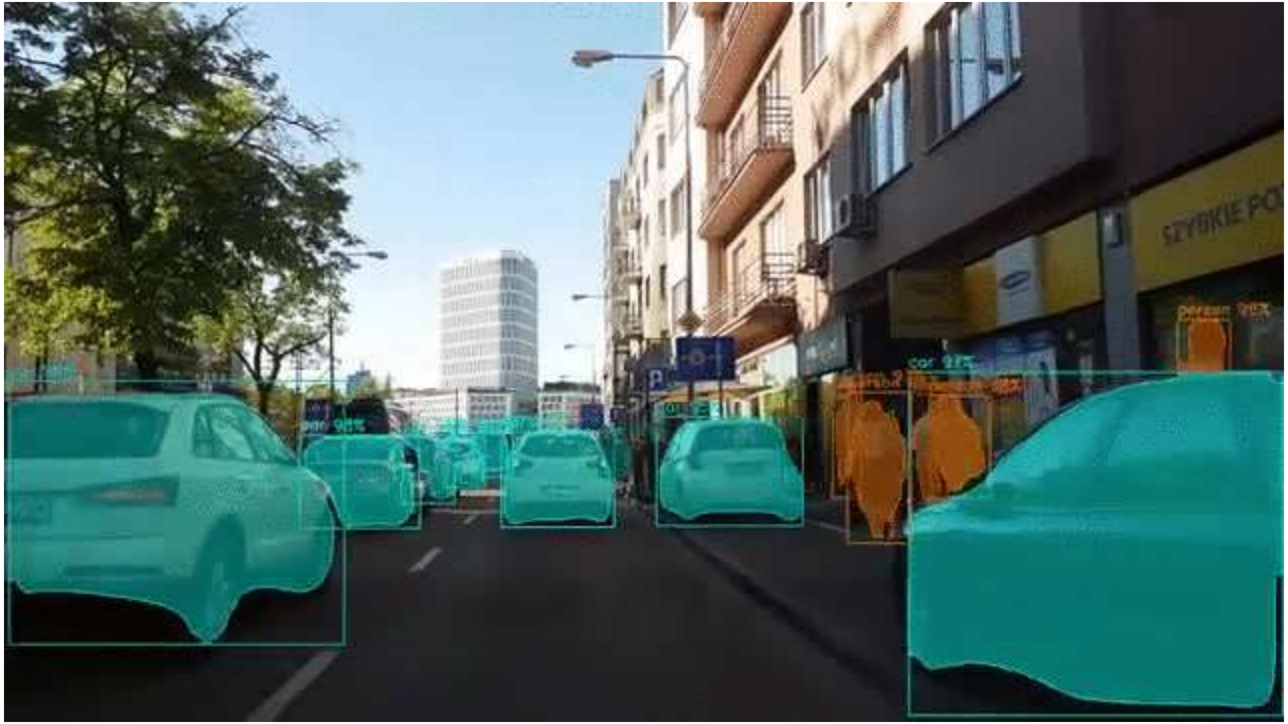
Deep Learning for Image Analysis

Markerless Motion Capture: Automatic 3D Surface Meshes from Video



DensePose
(Facebook)

Mask R-CNN



Deep Learning for Image Analysis

Style Transfer and Harmonization

Gatys et al. A Neural Algorithm of Artistic Style. *arXiv* (2015)

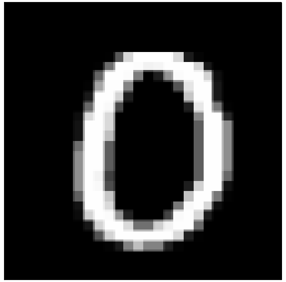
Deep Learning for Image Analysis

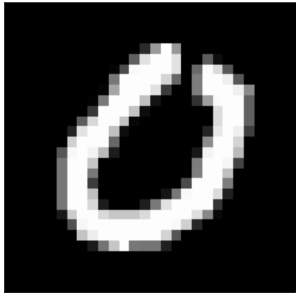
Style Transfer and Harmonization

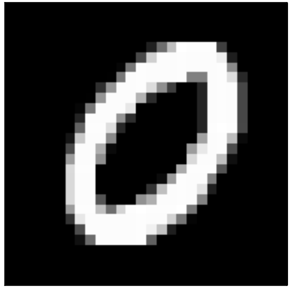
Luan et al. Deep Painterly Harmonization. *arXiv* (2018)

Motivating the CNN: Back to MNIST

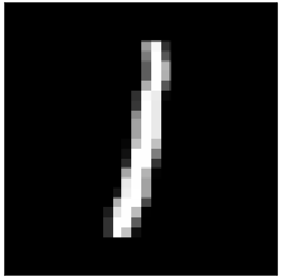
Zeros

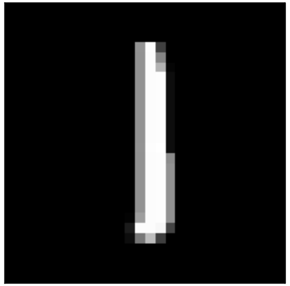


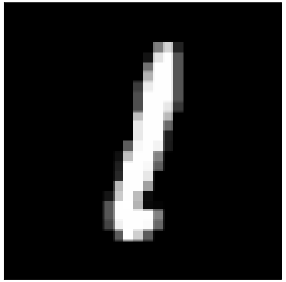




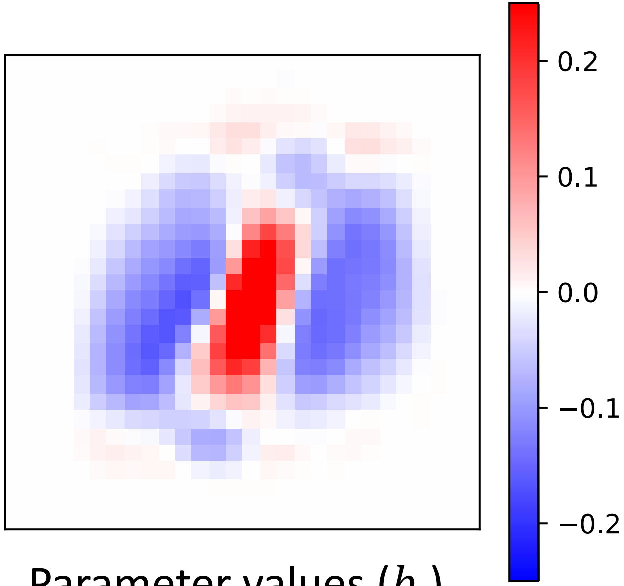
Ones





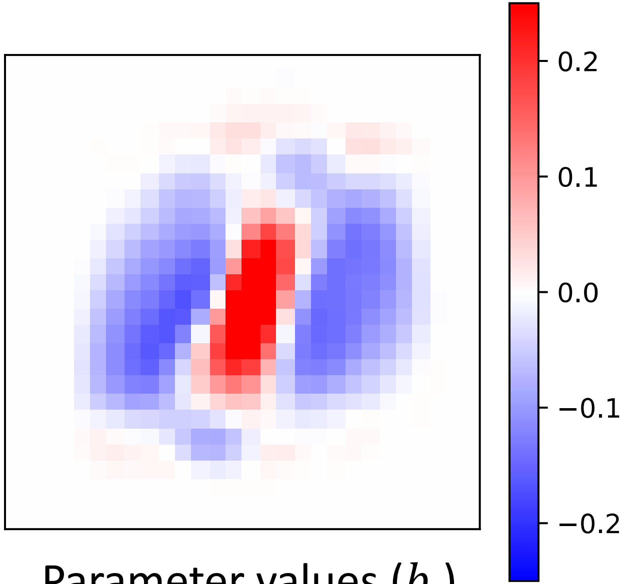


A “Filter” to Detect Ones

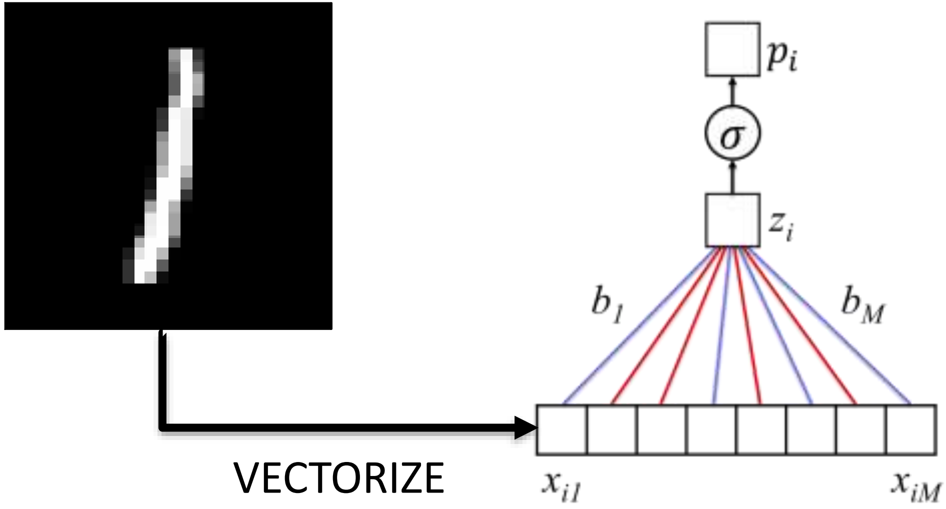


Parameter values (b_j)
for each pixel j

A “Filter” to Detect Ones

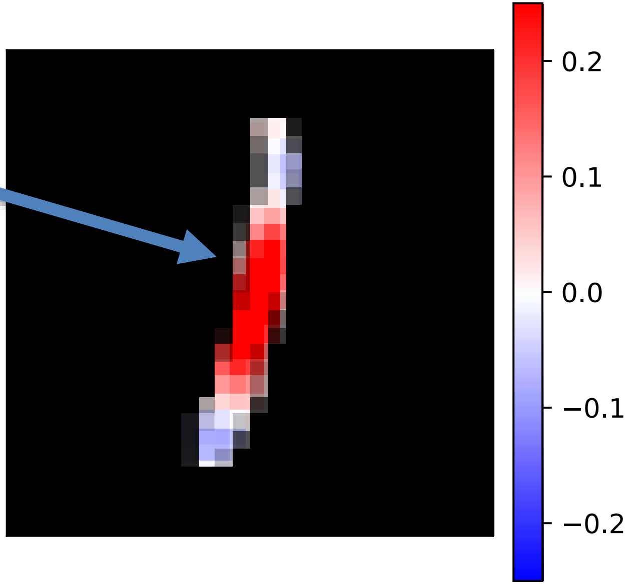


Parameter values (b_j)
for each pixel j



A "Filter" to Detect Ones

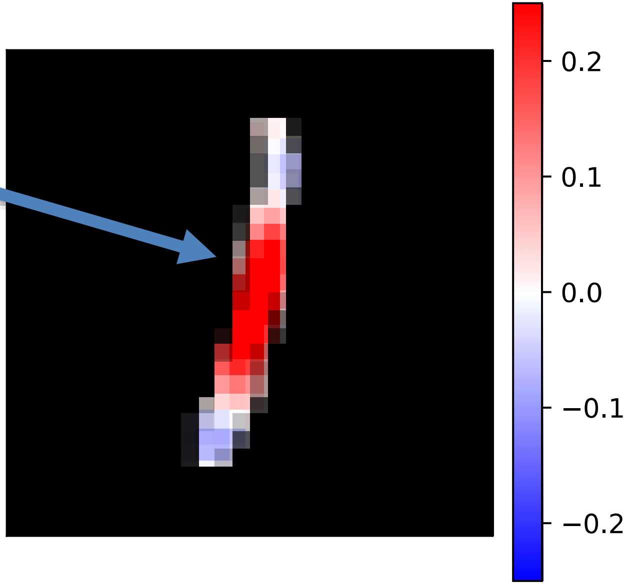
Positive Section



A “Filter” to Detect Ones

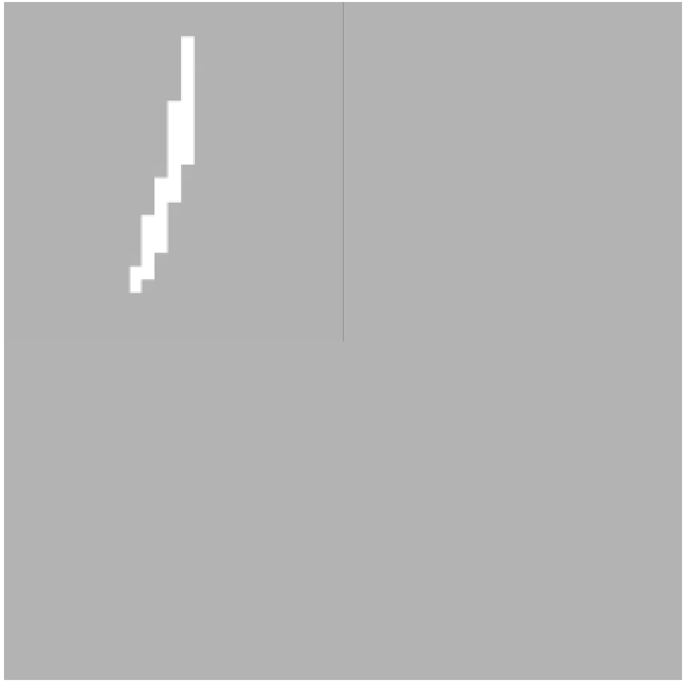
Positive Section

$\sigma($

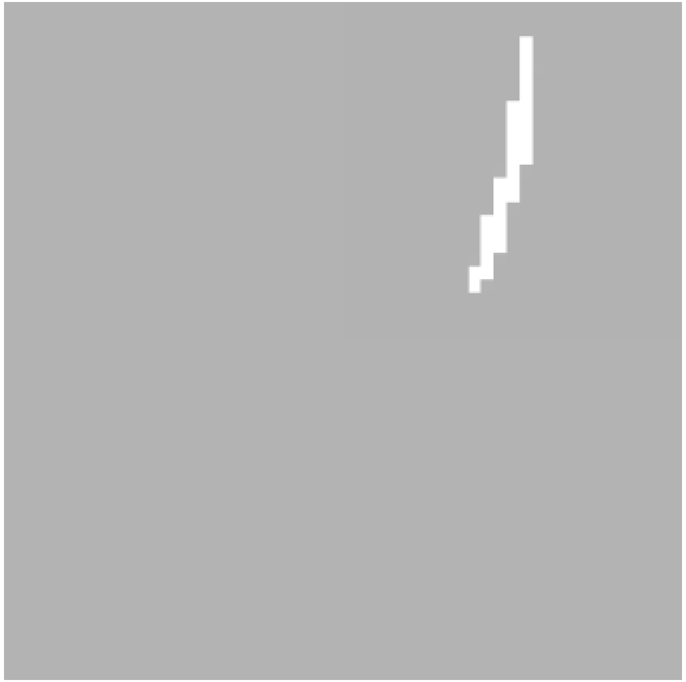


$) = .991$

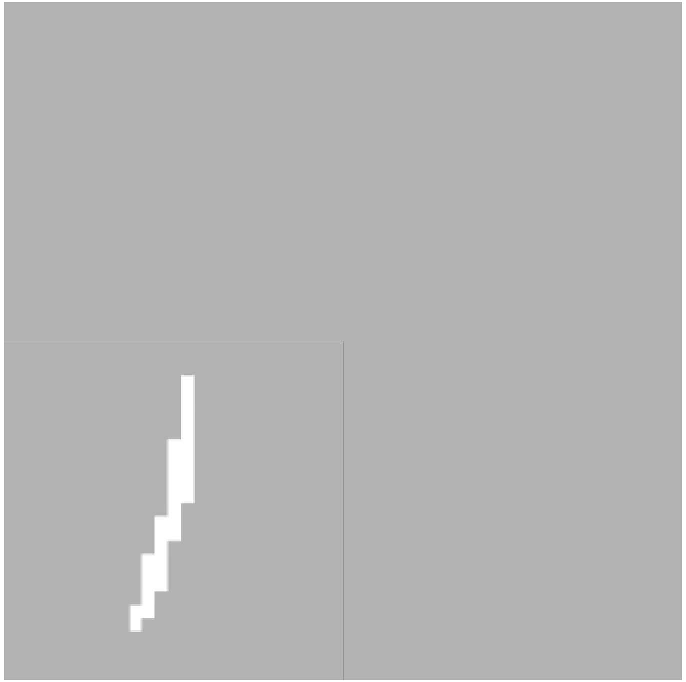
What if we'd like to find a 1 anywhere in a larger image?



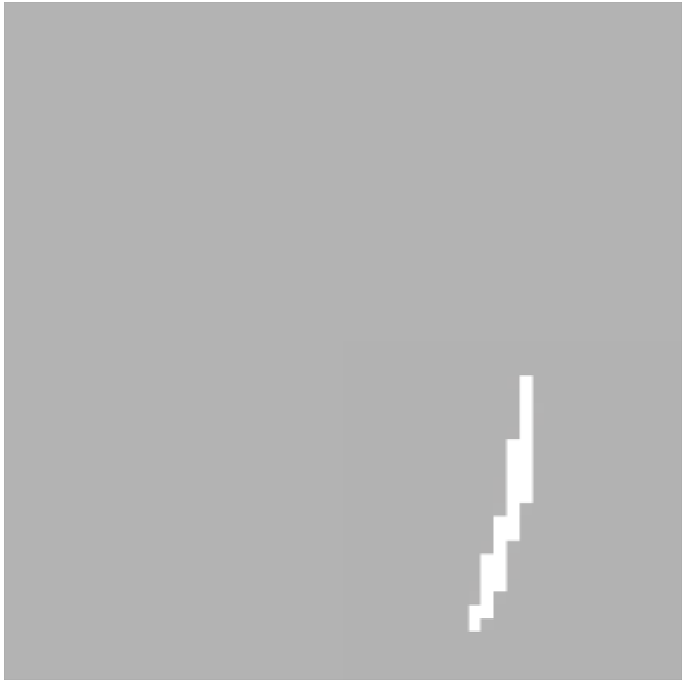
Searching for a 1…



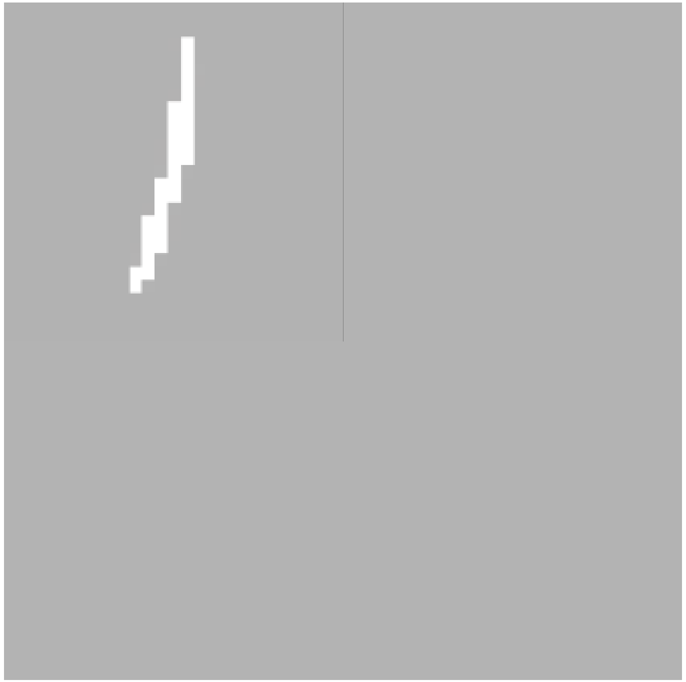
Searching for a 1…

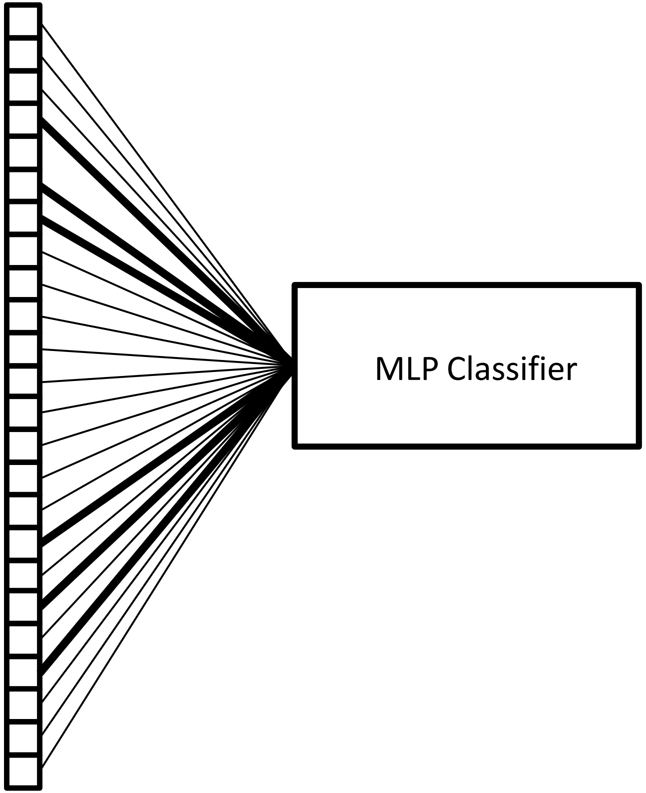


Searching for a 1…

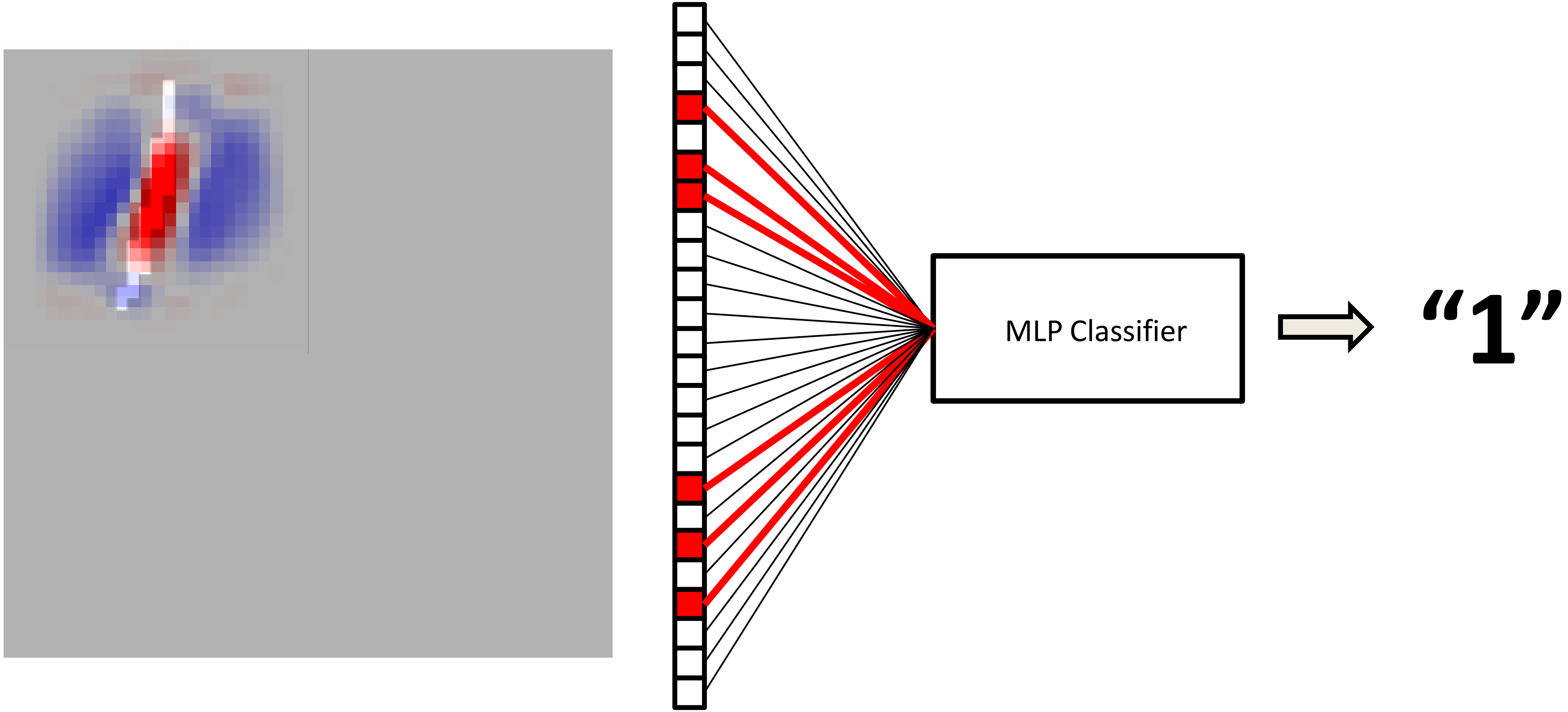


Our previous approach looks for a 1 at a specific location.

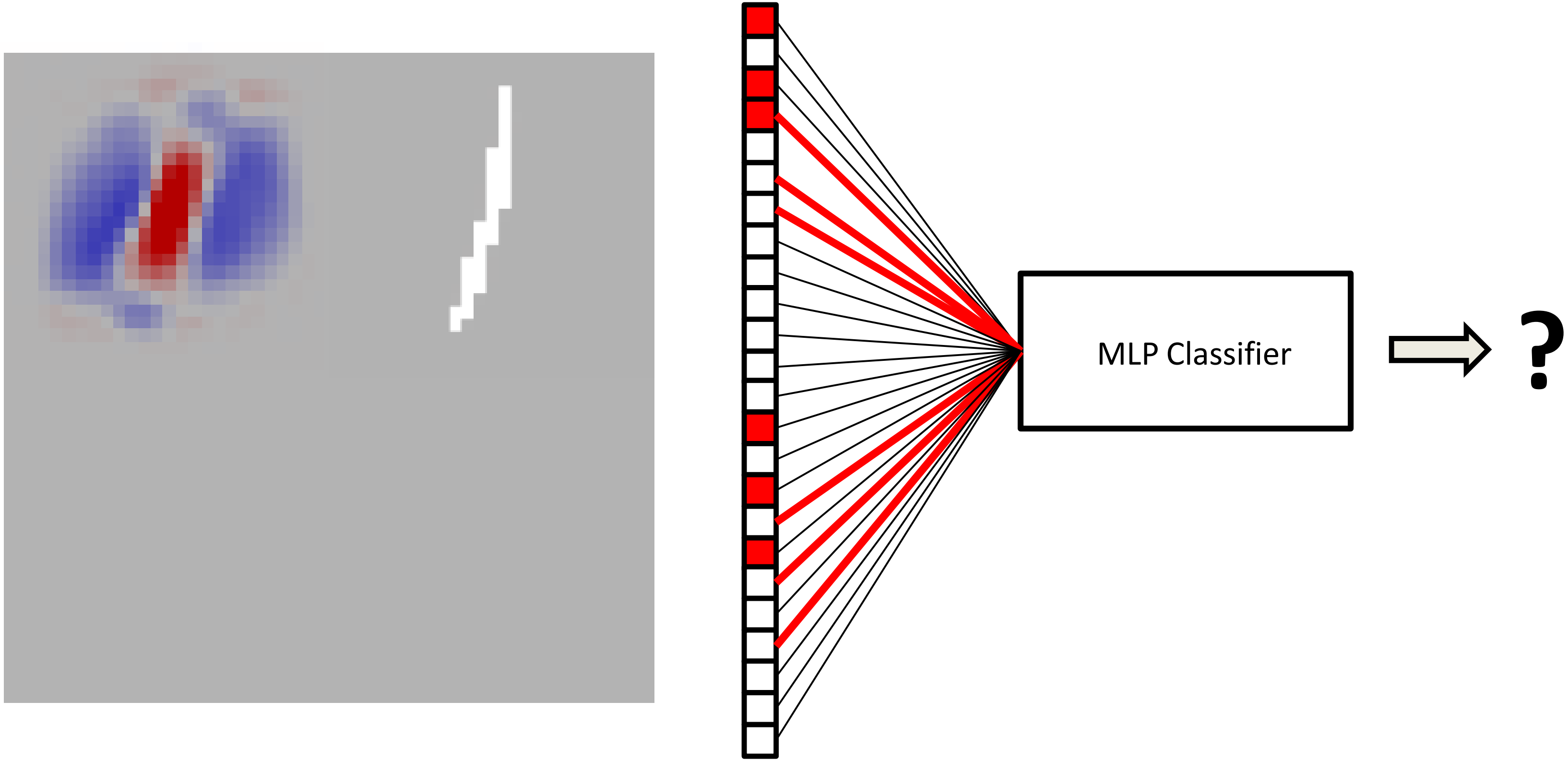




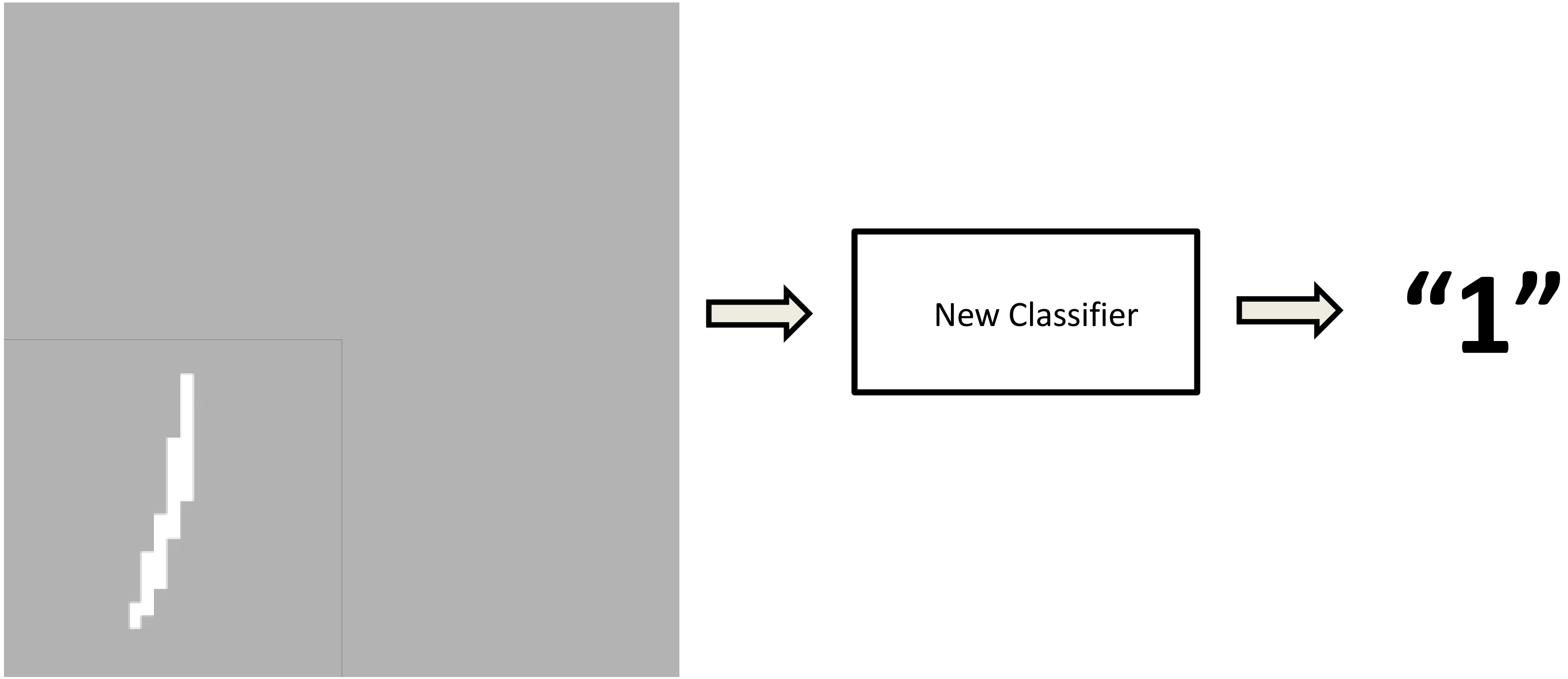
Our previous approach looks for a 1 at a specific location.



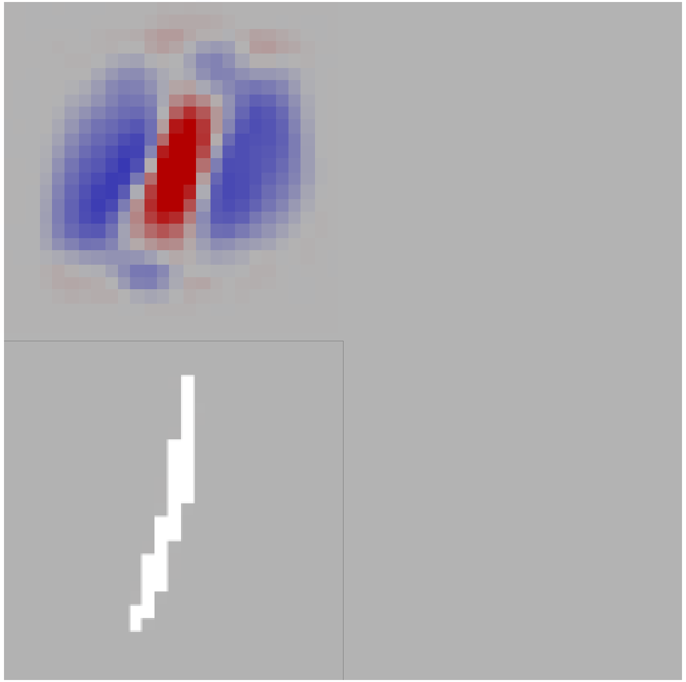
If we move the position of the 1, it no longer works.



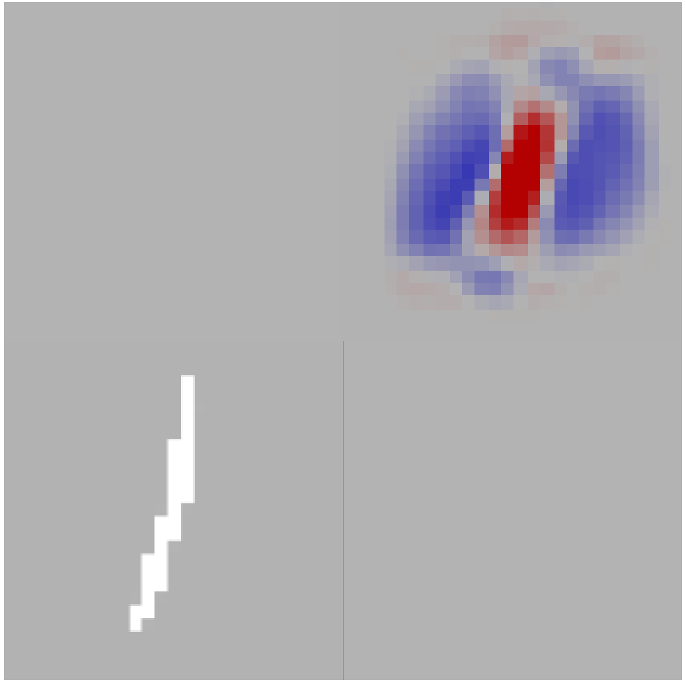
Instead of using logistic regression or an MLP, let's look for a new kind of model, one with more flexible filters



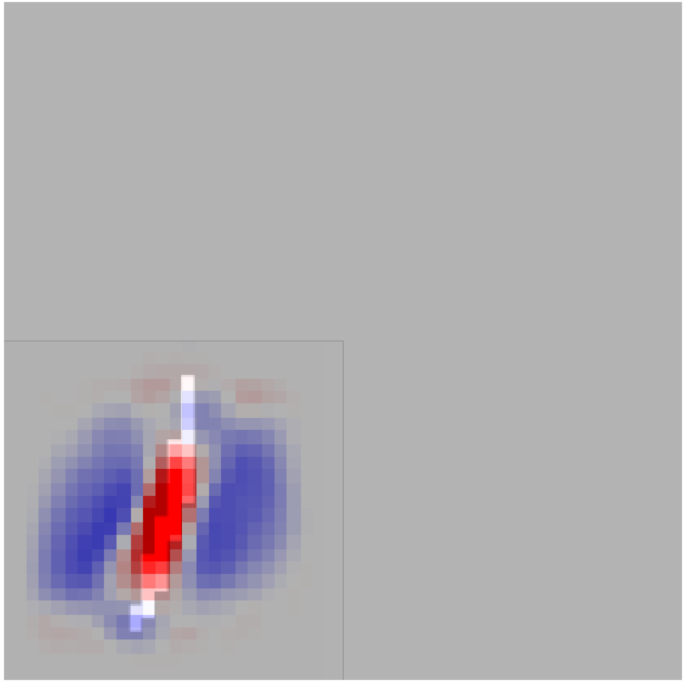
Instead of a filter that's the size of the whole image, we'd like a <u>smaller filter that we can move around</u>



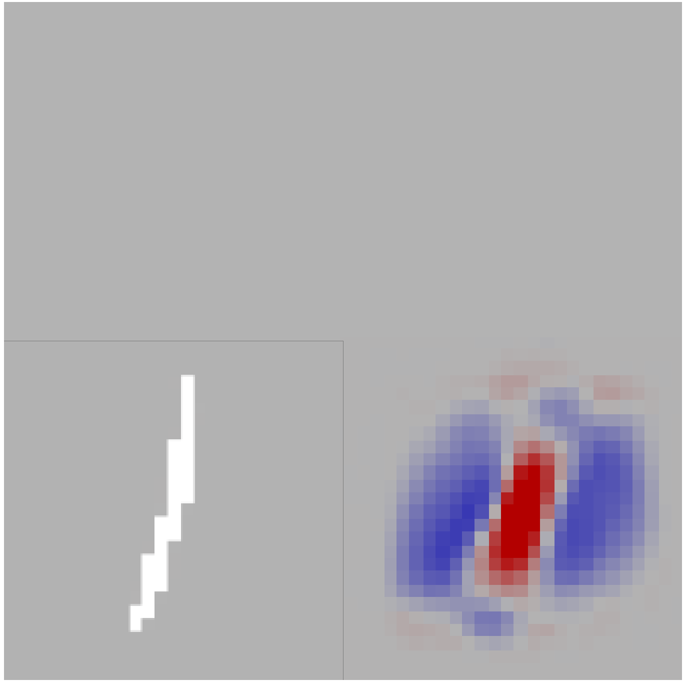
As we move this filter, we calculate the inner product between the filter itself and the portion of the image that's underneath it.



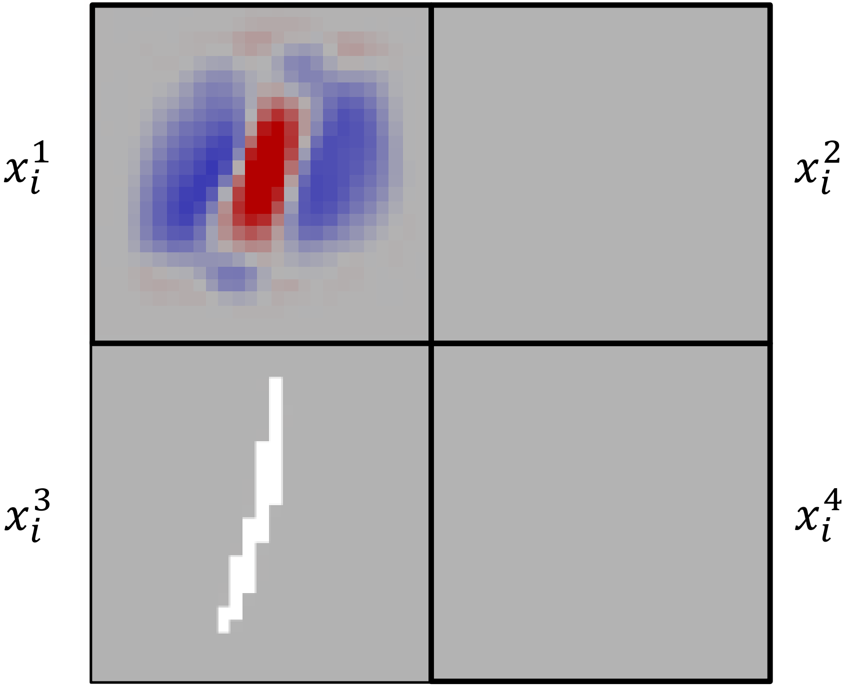
When the filter is placed over a region that looks like the filter, the inner product (i.e. filter output) will be large.



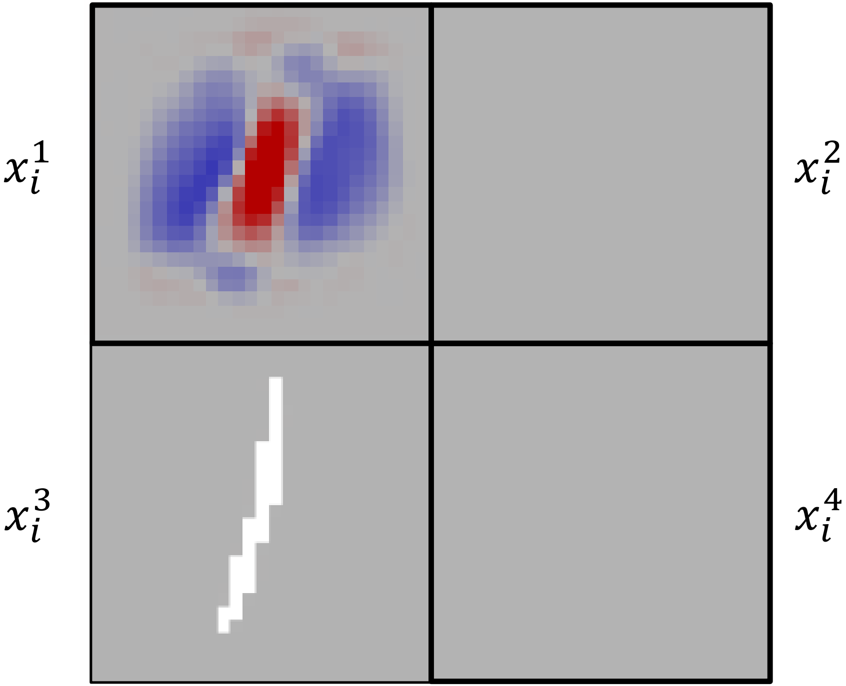
When it's placed over a region that does not look like the filter, the inner product (i.e. filter output) will be small

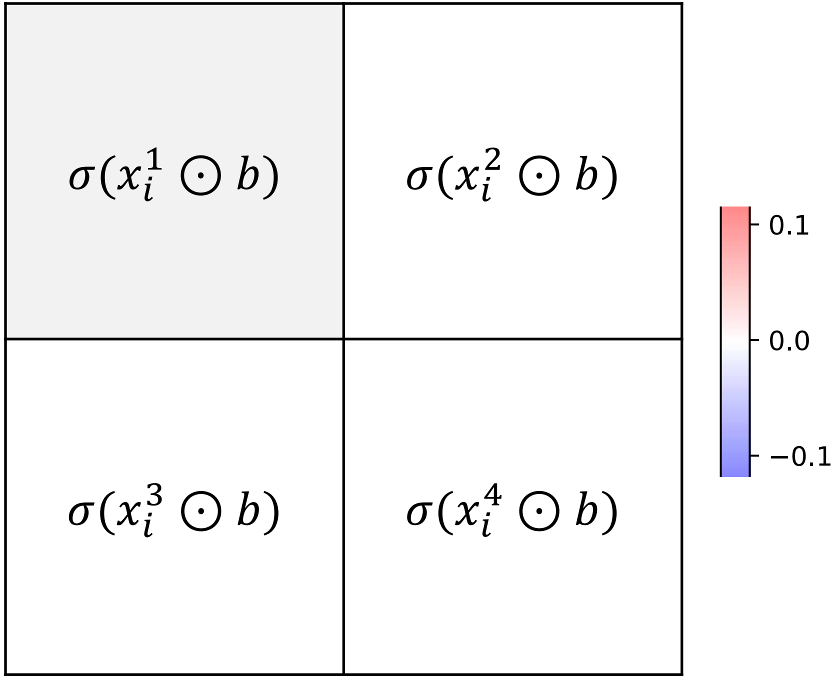


Examining filter output $\sigma(x_i^R \odot b)$, where x_i^R is the portion of image i where the filter is placed.

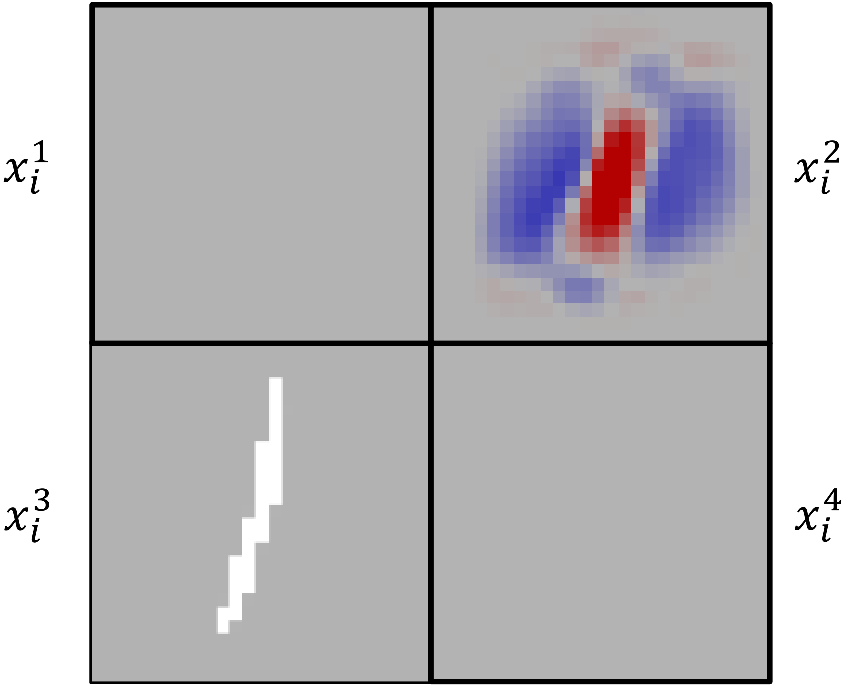


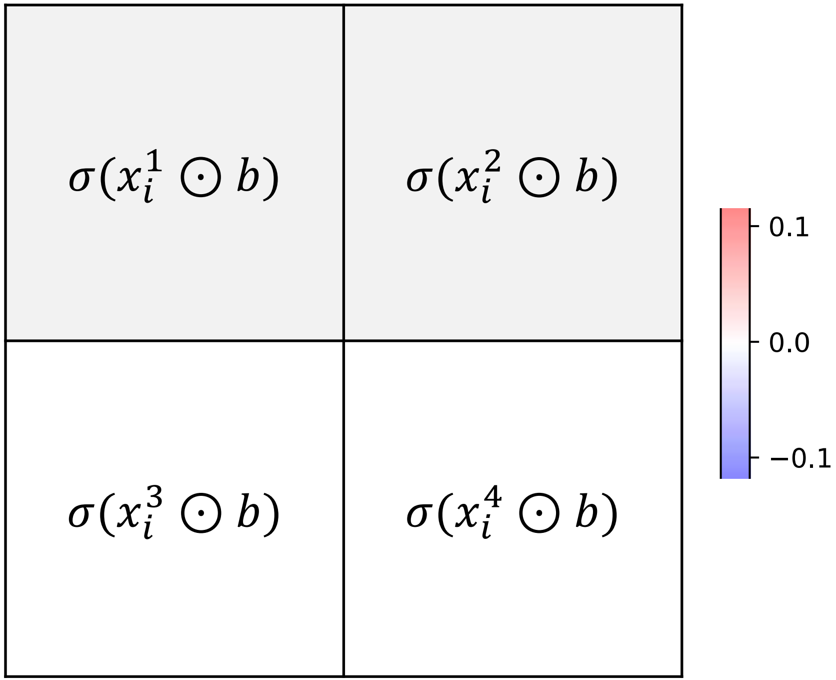
Examining filter output $\sigma(x_i^R \odot b)$, where x_i^R is the portion of image i where the filter is placed.



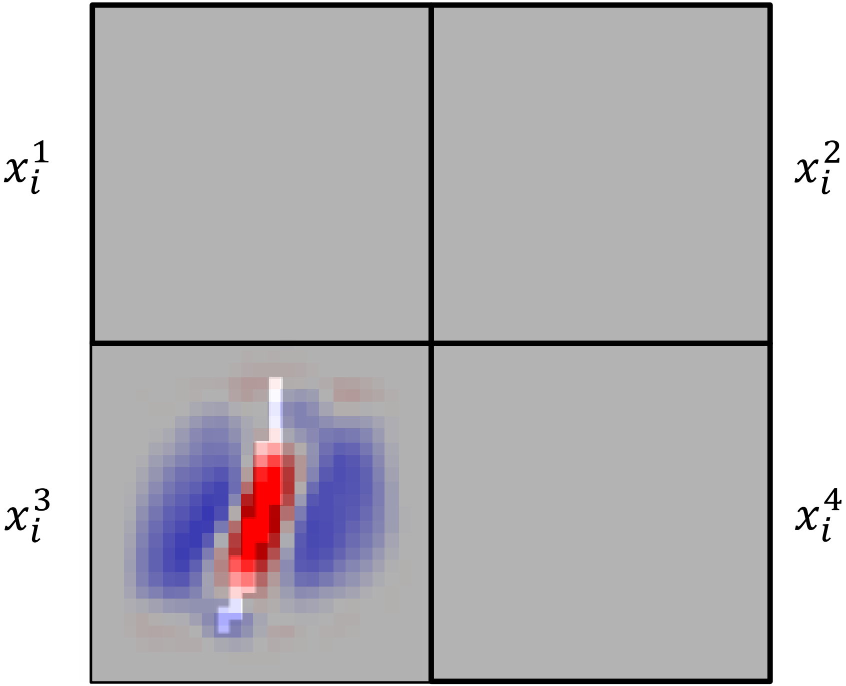


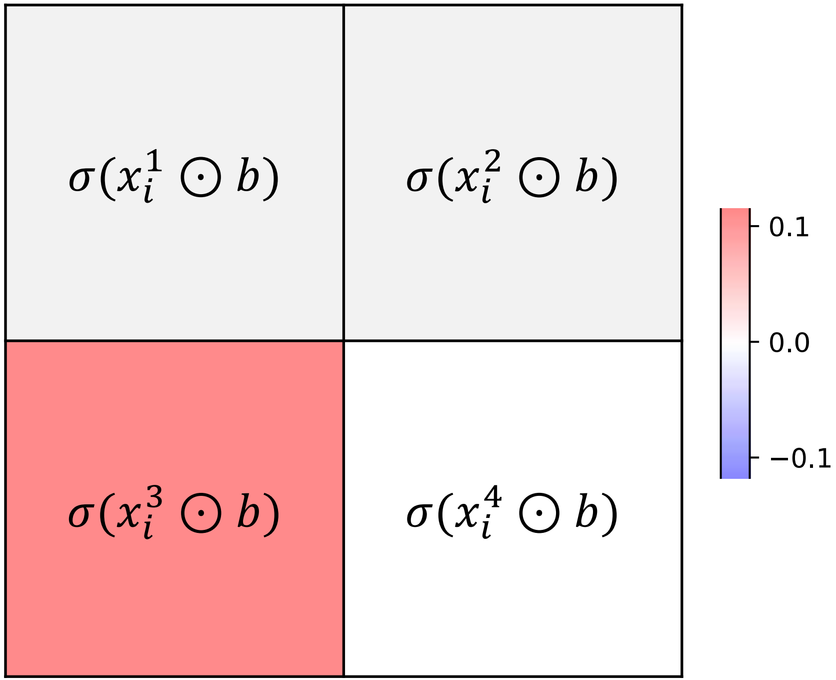
Examining filter output $\sigma(x_i^R \odot b)$, where x_i^R is the portion of image i where the filter is placed.



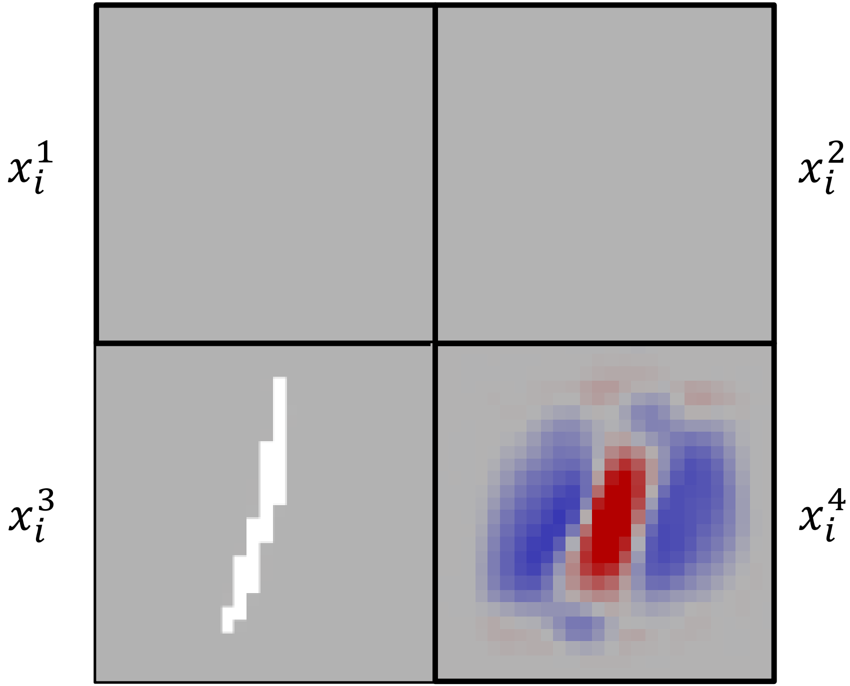


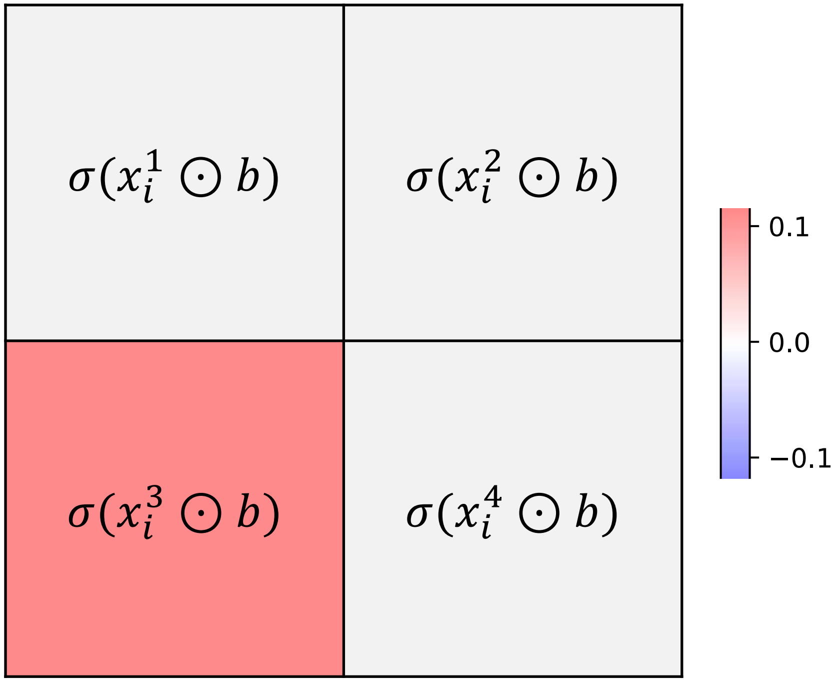
Examining filter output $\sigma(x_i^R \odot b)$, where x_i^R is the portion of image i where the filter is placed.



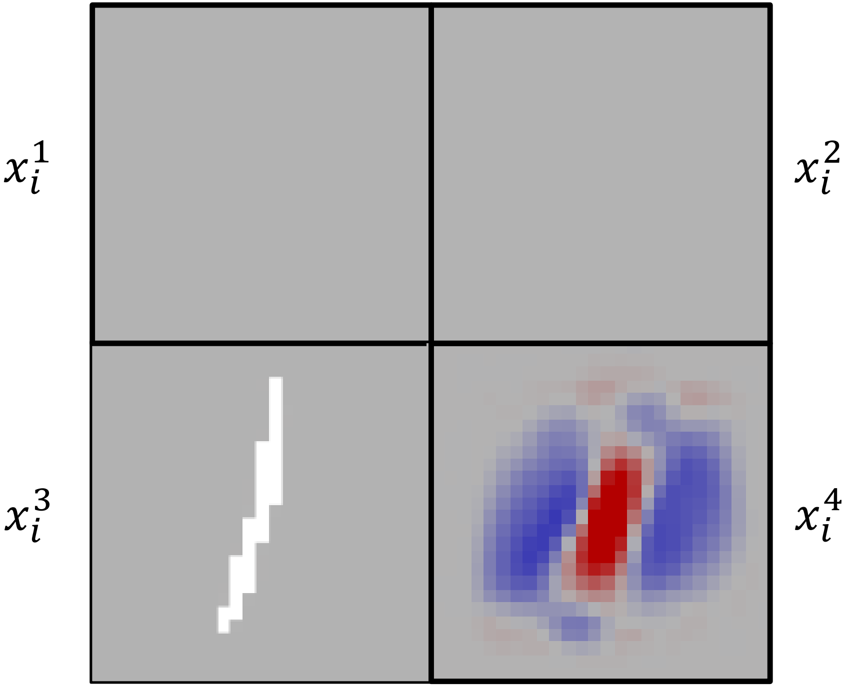


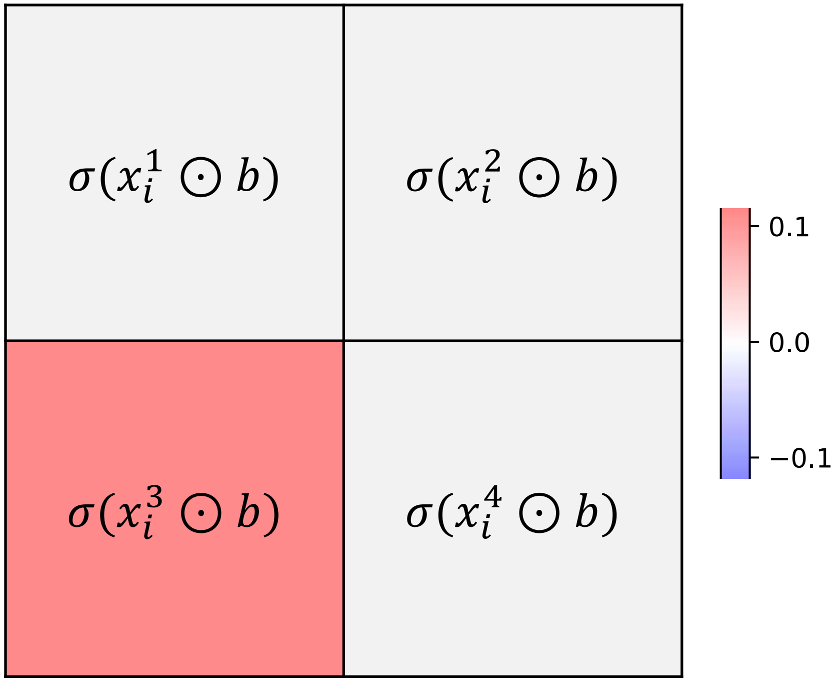
Examining filter output $\sigma(x_i^R \odot b)$, where x_i^R is the portion of image i where the filter is placed.



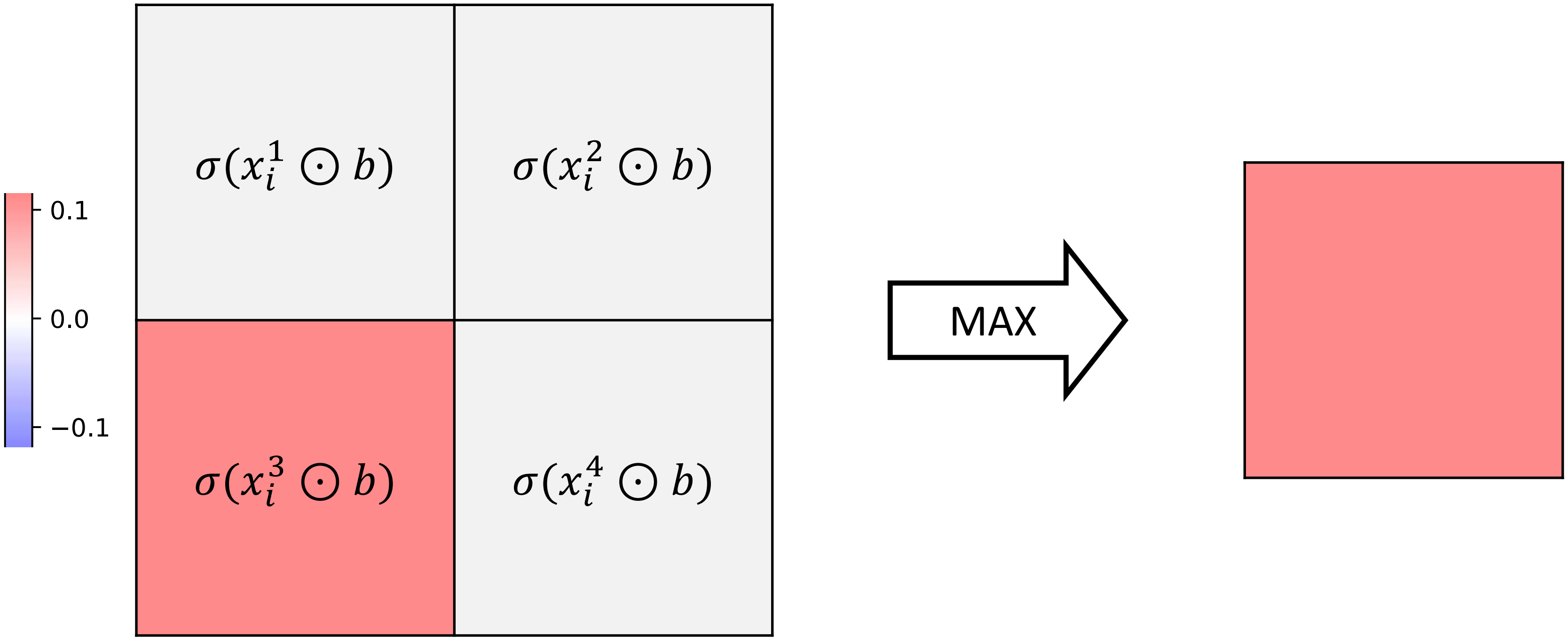


What if we want to know if a 1 is present *anywhere* in the image?

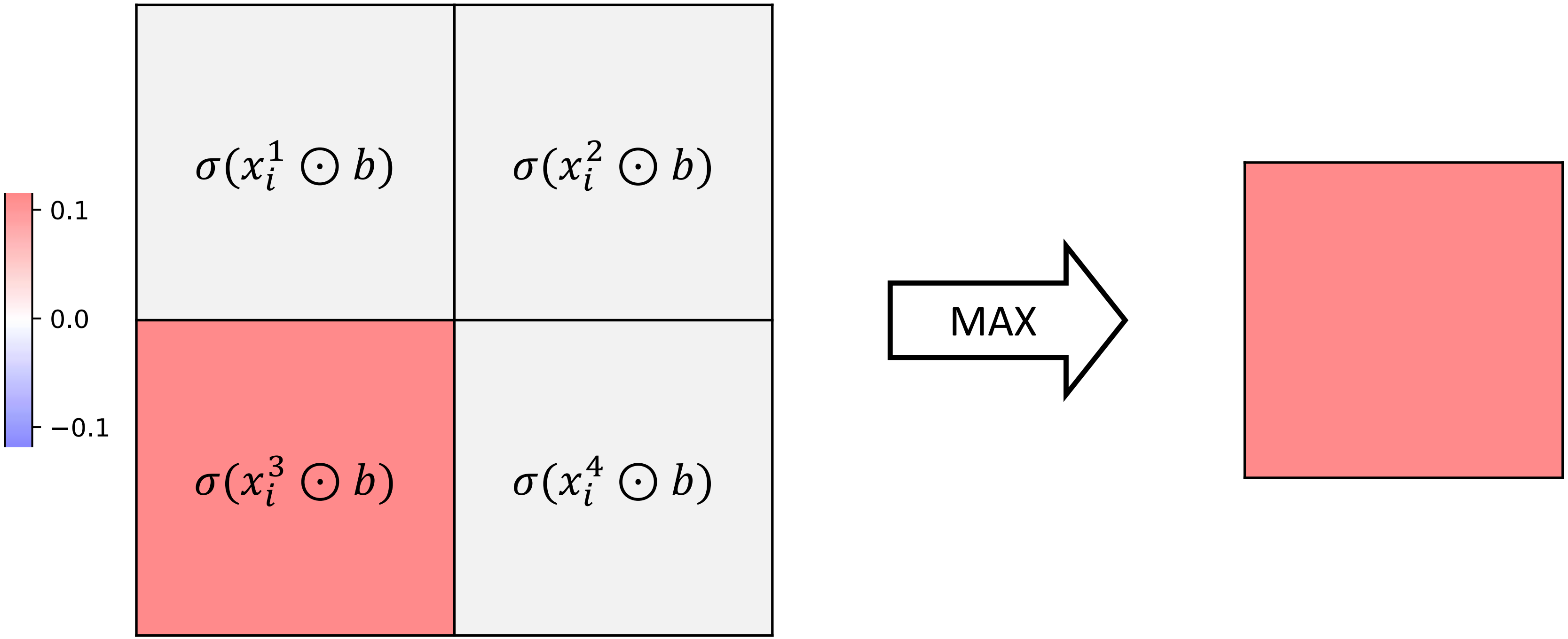




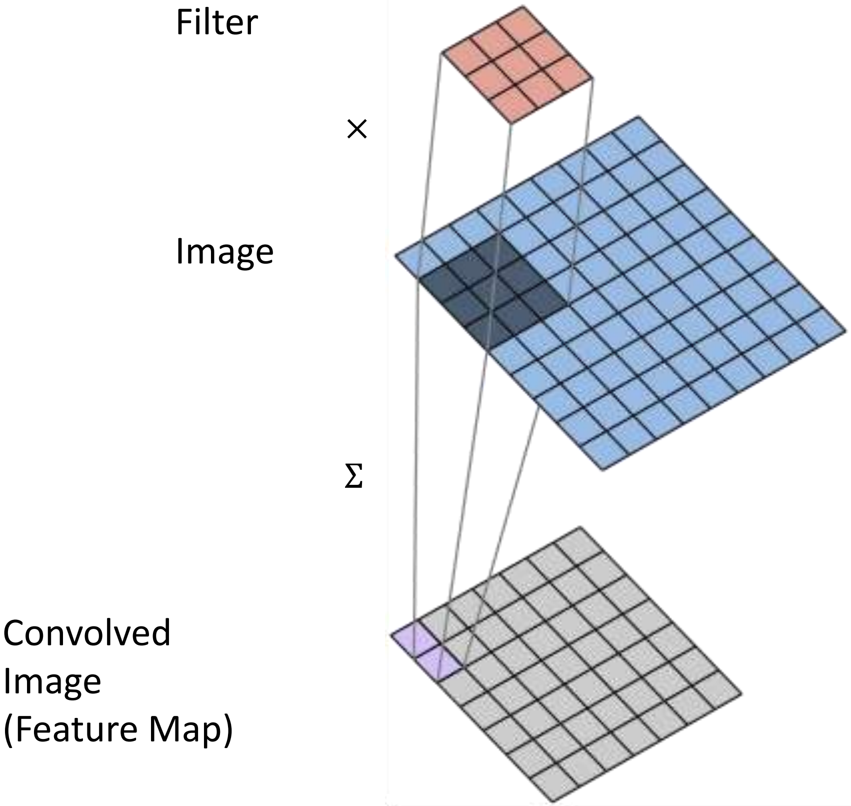
What if we want to know if a 1 is present *anywhere* in the image?



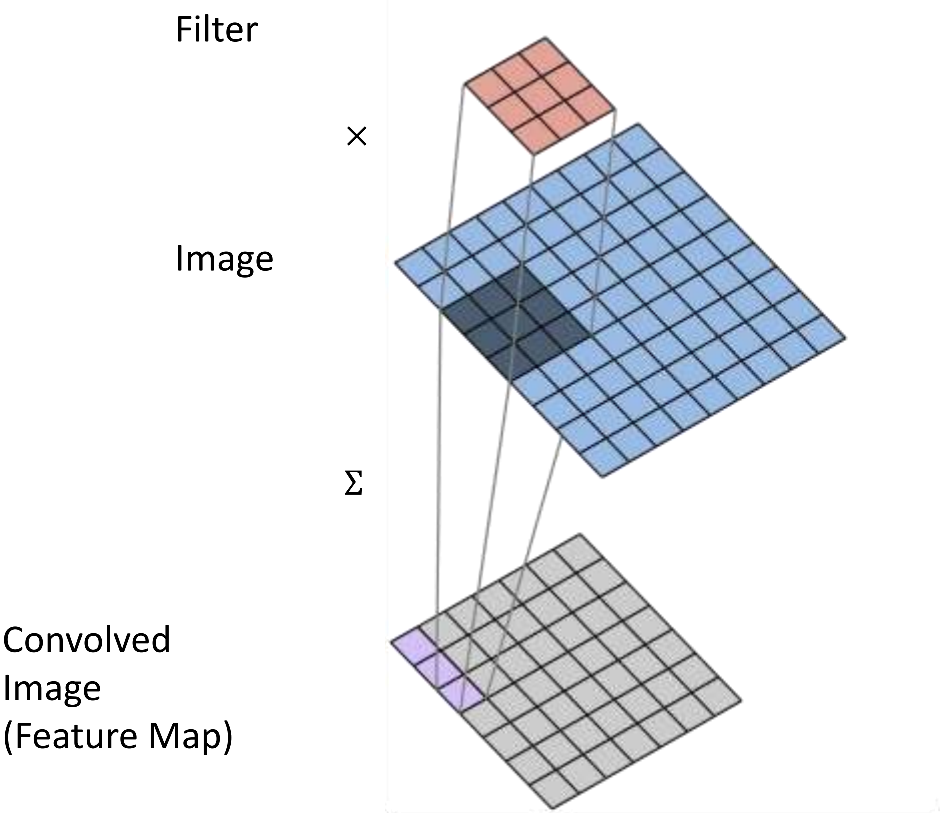
This idea is called "max pooling", and is widely used in CNNs to determine whether features are present in a given region



We perform “2D Spatial Convolution” as we move the filter across the image.



We perform "2D Spatial Convolution" as we move the filter across the image.



We perform "2D Spatial Convolution" as we move the filter across the image.

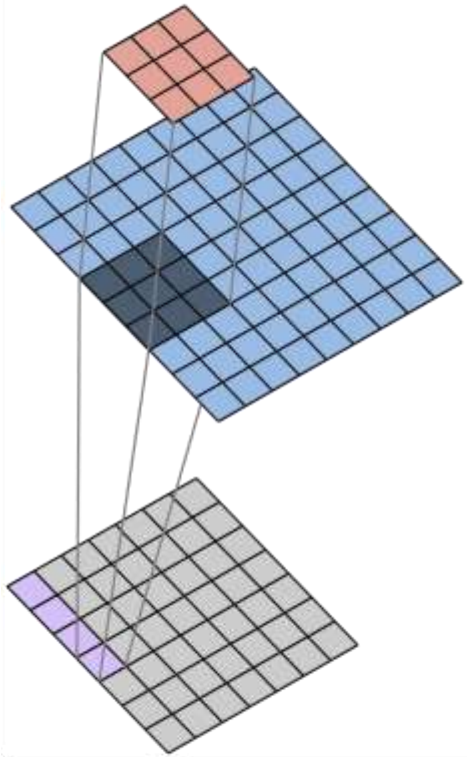
Filter

×

Image

Σ

Convolved Image (Feature Map)

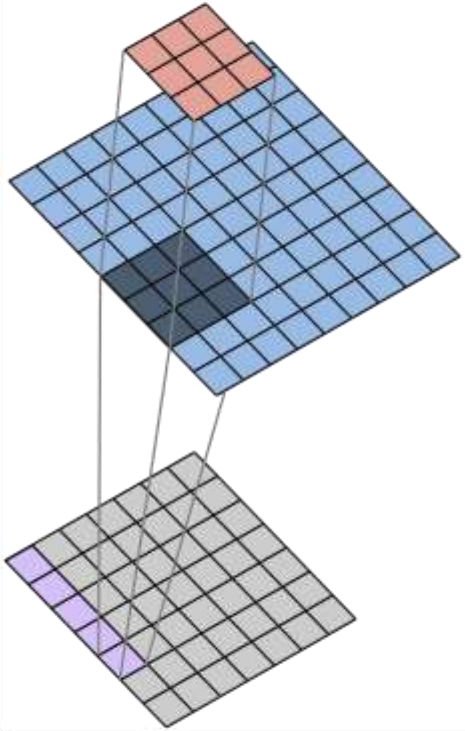


We perform "2D Spatial Convolution" as we move the filter across the image.

Filter

×

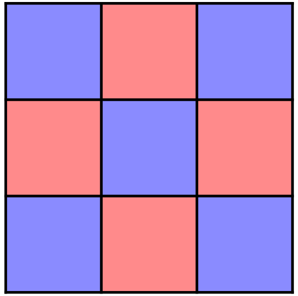
Image



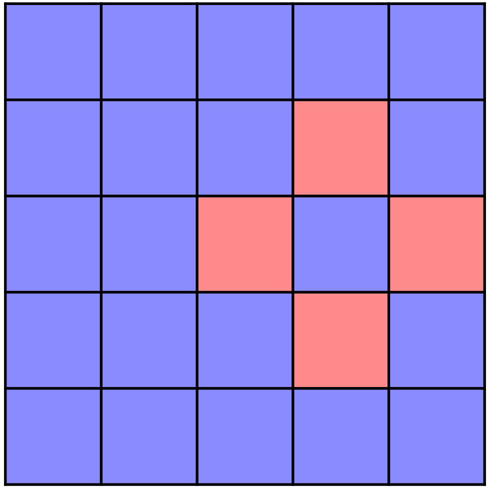
Σ

Convolved
Image
(Feature Map)

An Example…



filter



image

An Example…

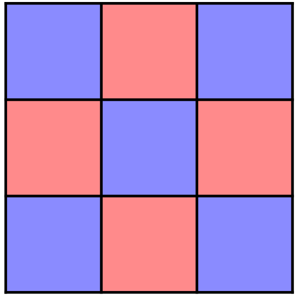
-1	1	-1
1	-1	1
-1	1	-1

filter

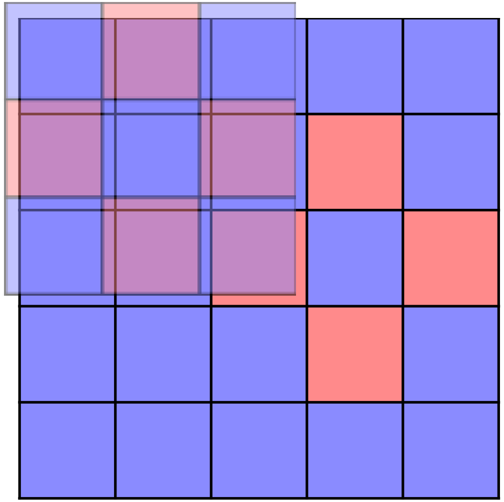
-1	-1	-1	-1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	1
-1	-1	-1	1	-1
-1	-1	-1	-1	-1

image

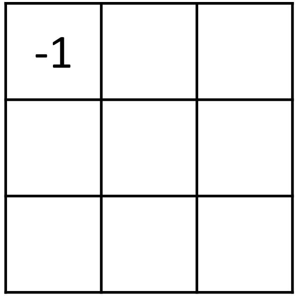
An Example…



filter

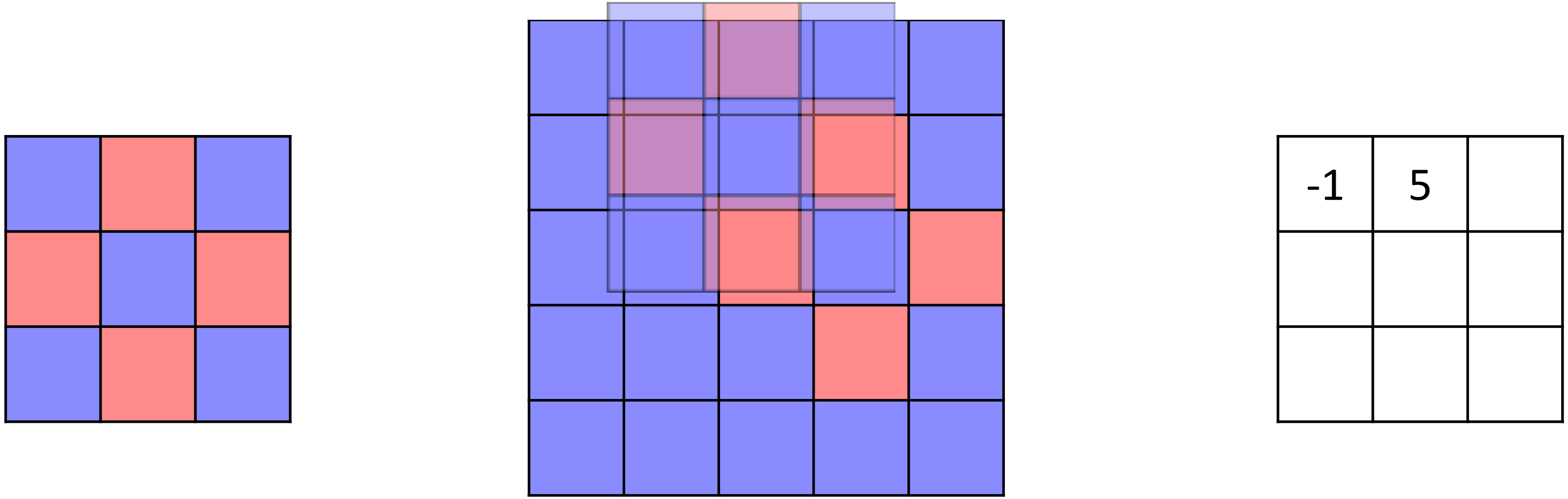


image



$x_i^R \odot b$

An Example…

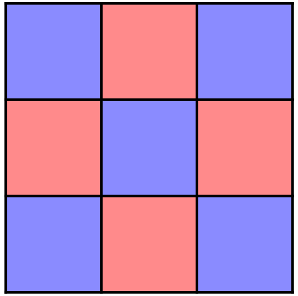


filter

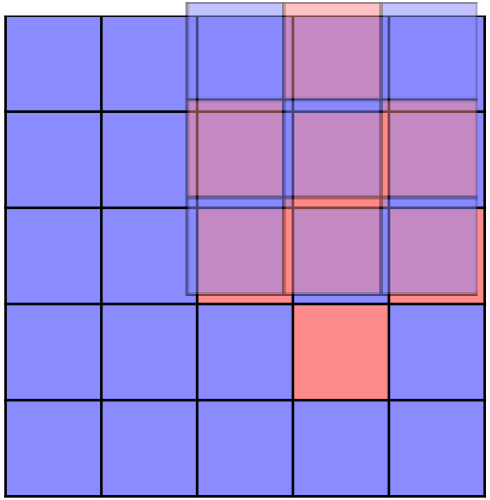
image

$x_i^R \odot b$

An Example…



filter

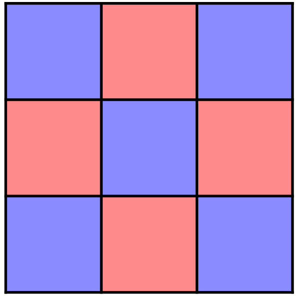


image

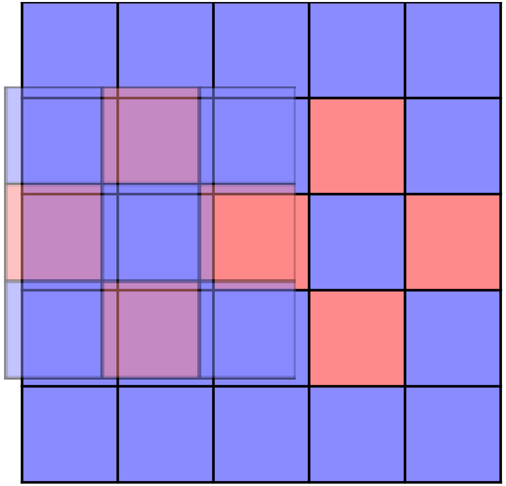
-1	5	-5

$x_i^R \odot b$

An Example…



filter

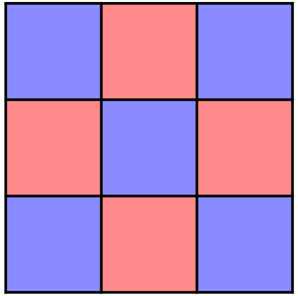


image

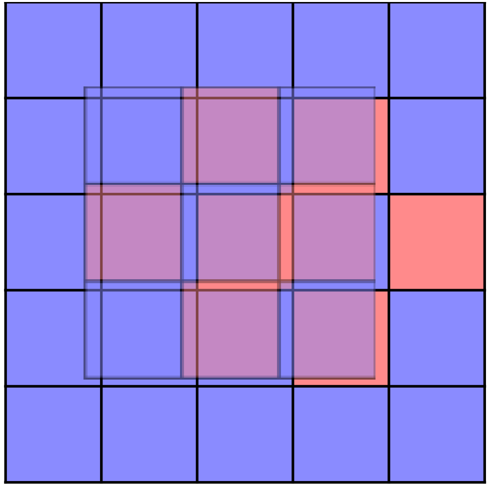
-1	5	-5
3		

$x_i^R \odot b$

An Example…



filter

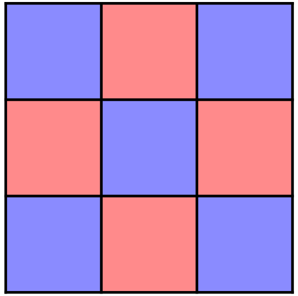


image

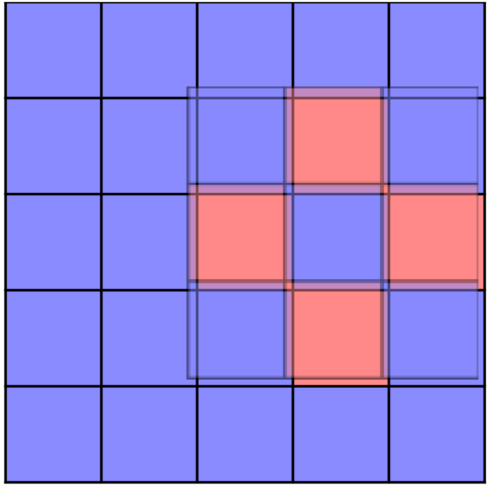
-1	5	-5
3	-5	

$x_i^R \odot b$

An Example…



filter

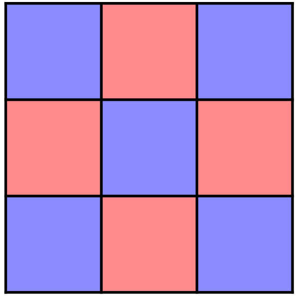


image

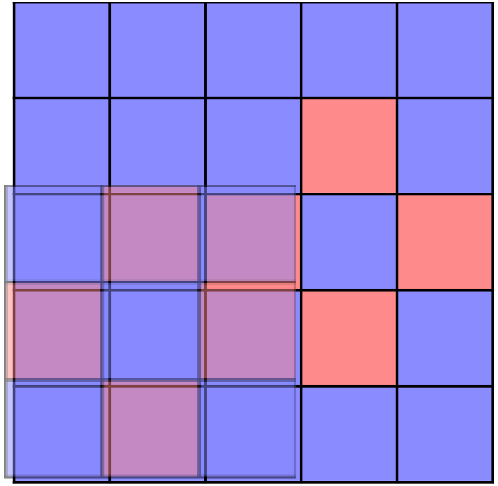
-1	5	-5
3	-5	9

$x_i^R \odot b$

An Example…



filter

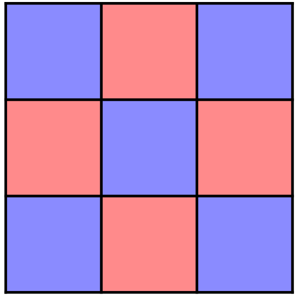


image

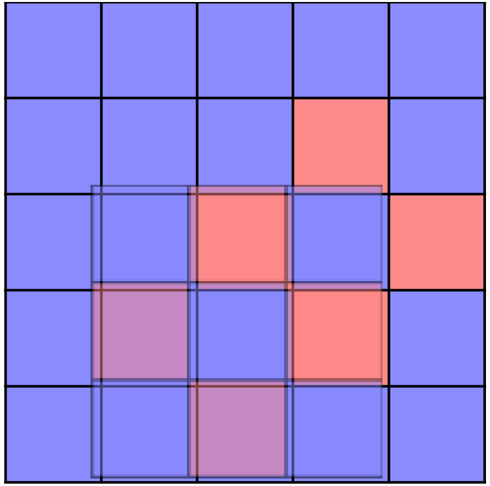
-1	5	-5
3	-5	9
-1		

$x_i^R \odot b$

An Example…



filter

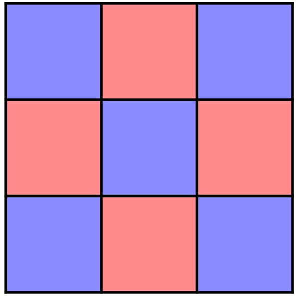


image

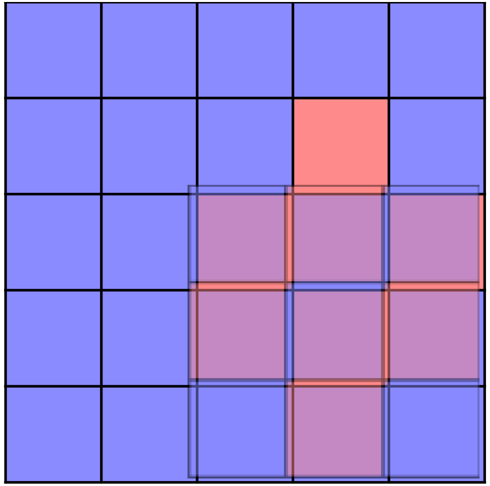
-1	5	-5
3	-5	9
-1	5	

$x_i^R \odot b$

An Example…



filter

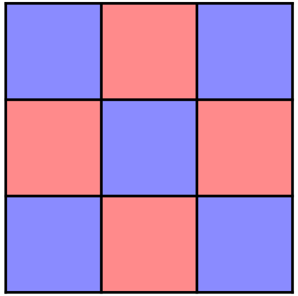


image

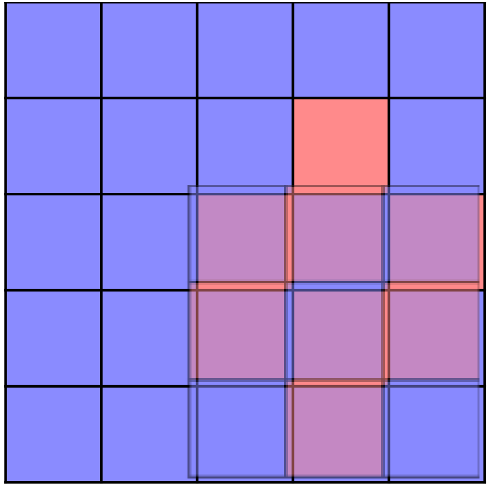
-1	5	-5
3	-5	9
-1	5	-5

$x_i^R \odot b$

An Example…



filter

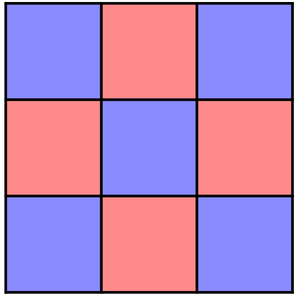


image

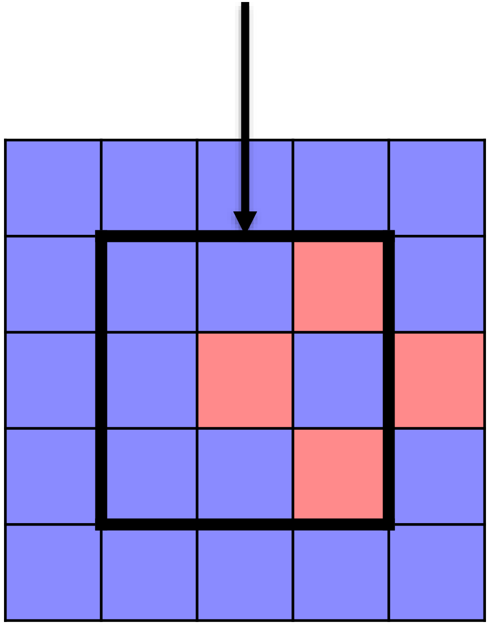
-1	5	-5
3	-5	9
-1	5	-5

$x_i^R \odot b$

Each location where the filter was centered has been evaluated: “how similar is this location to the filter”?



filter



image

-1	5	-5
3	-5	9
-1	5	-5

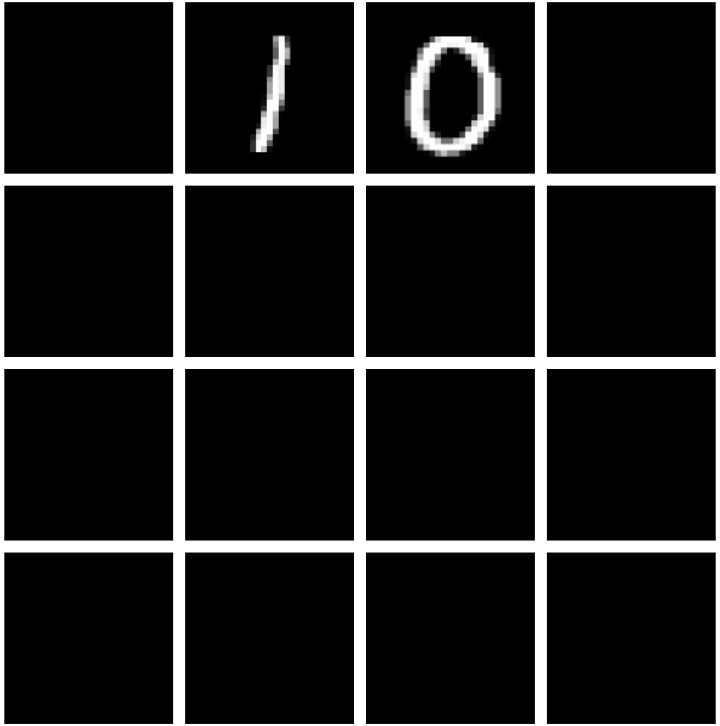
$x_i^R \odot b$

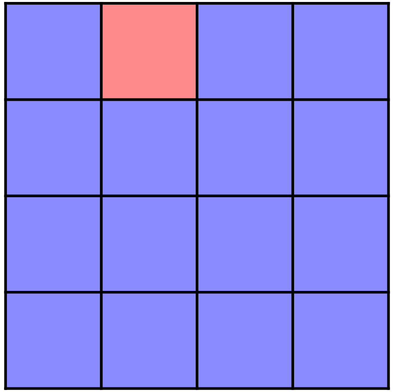
Convolutional Filters Are Feature Detectors

- Now we know how to identify a “1” or a “0” anywhere in an image.

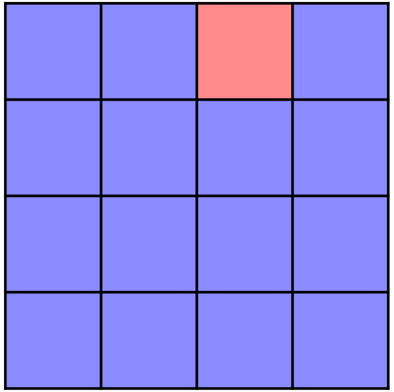
- What if we want to identify a “10”?

- Option 1: Design a new filter for “10”
- Option 2: Utilize our “1” and “0” filters…

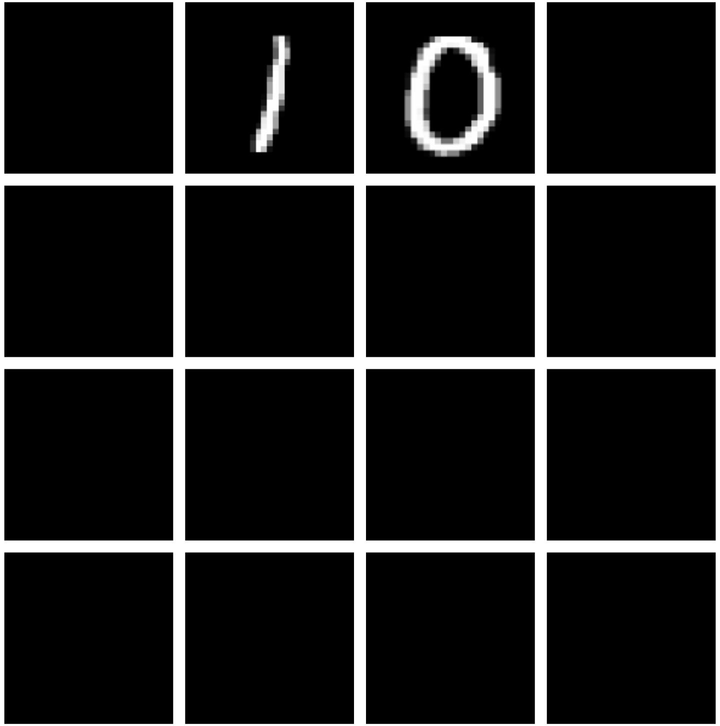


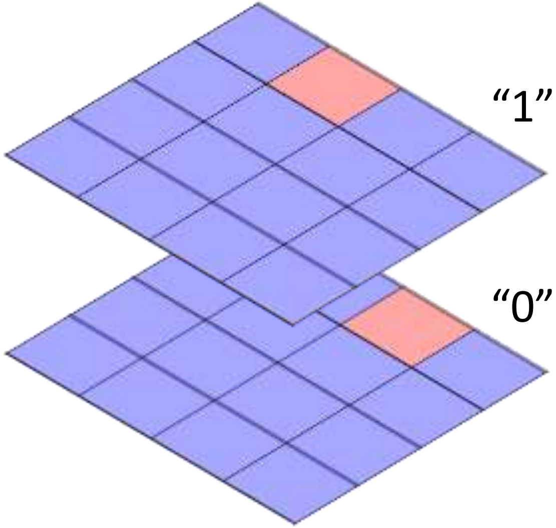


“1” filter output

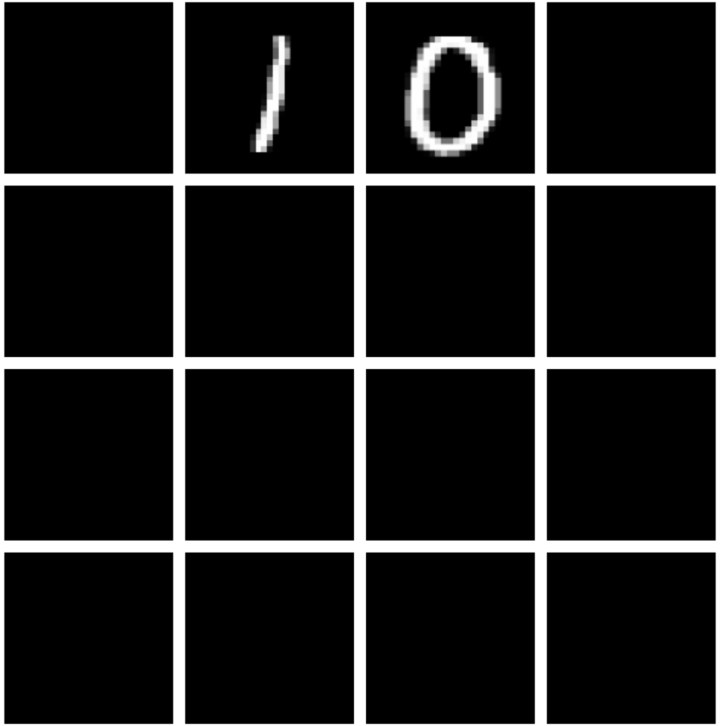


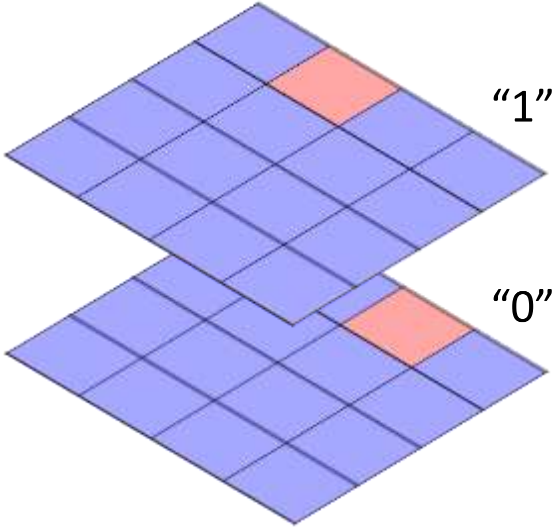
“0” filter output





Our “10” filter looks for a match from the “1” filter to the left of a match from the “0” filter

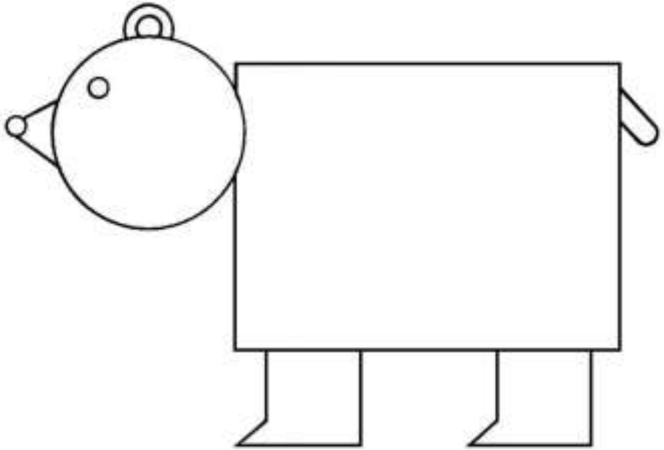


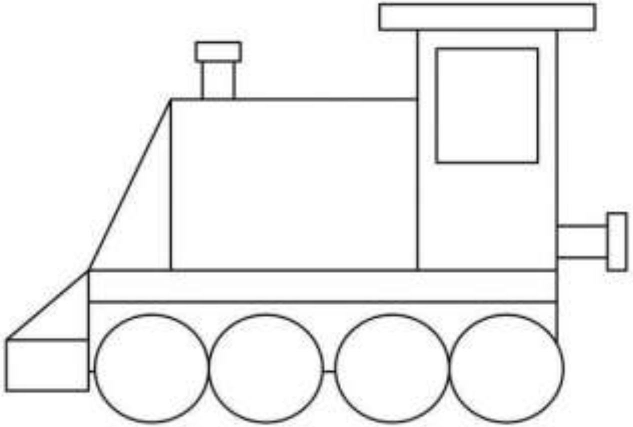


In this way, we learn to identify a hierarchy of features rather than a huge number of complex features

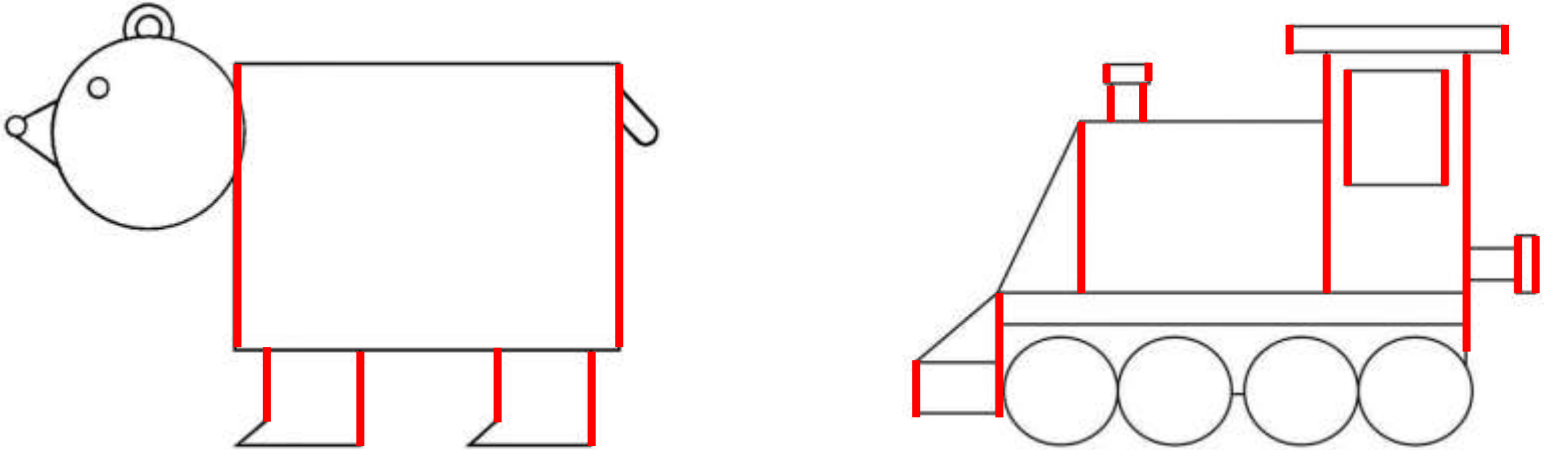
CNNs Take Advantage of Repeated, Hierarchical Structure in Images

CNNs Take Advantage of Repeated, Hierarchical Structure in Images



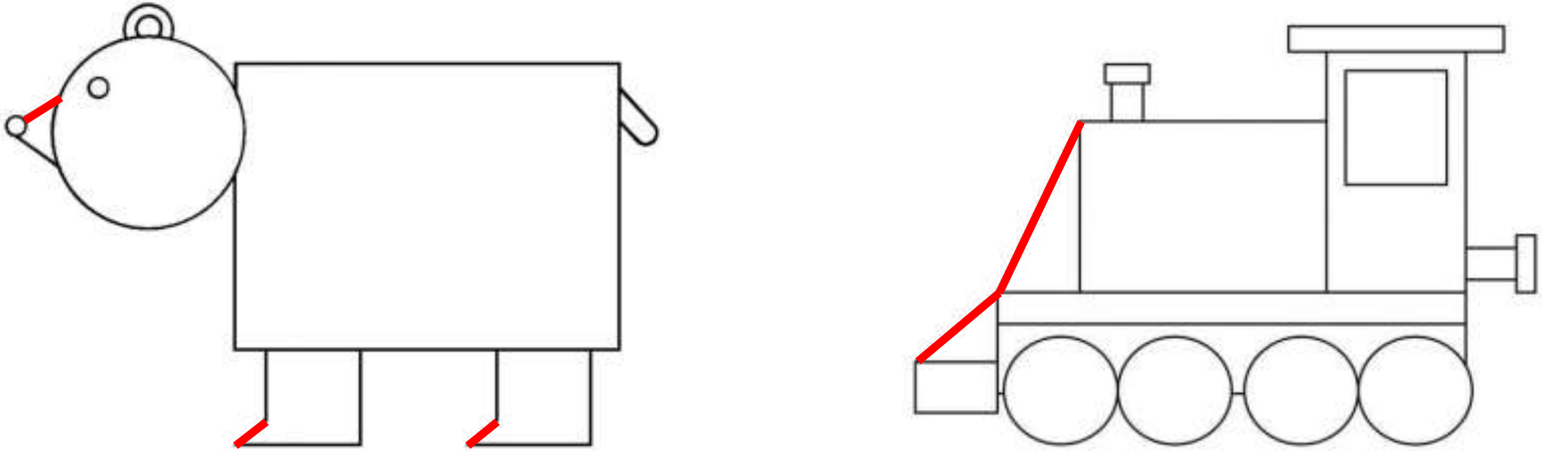


CNNs Take Advantage of Repeated, Hierarchical Structure in Images



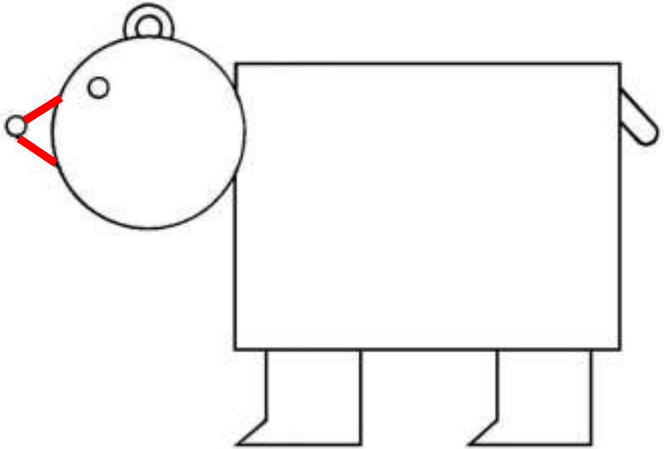
Low-level structure: lines, curves

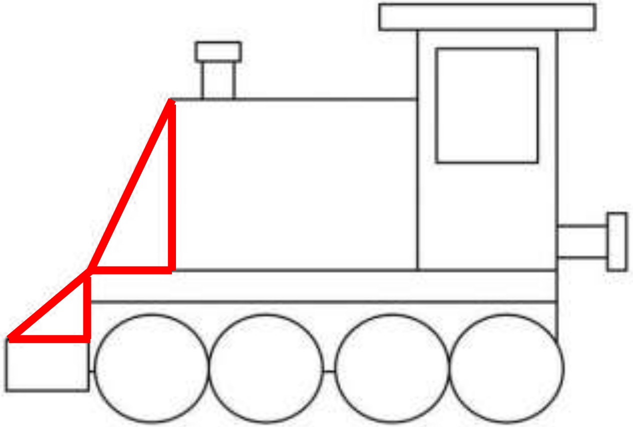
CNNs Take Advantage of Repeated, Hierarchical Structure in Images



Low-level structure: lines, curves

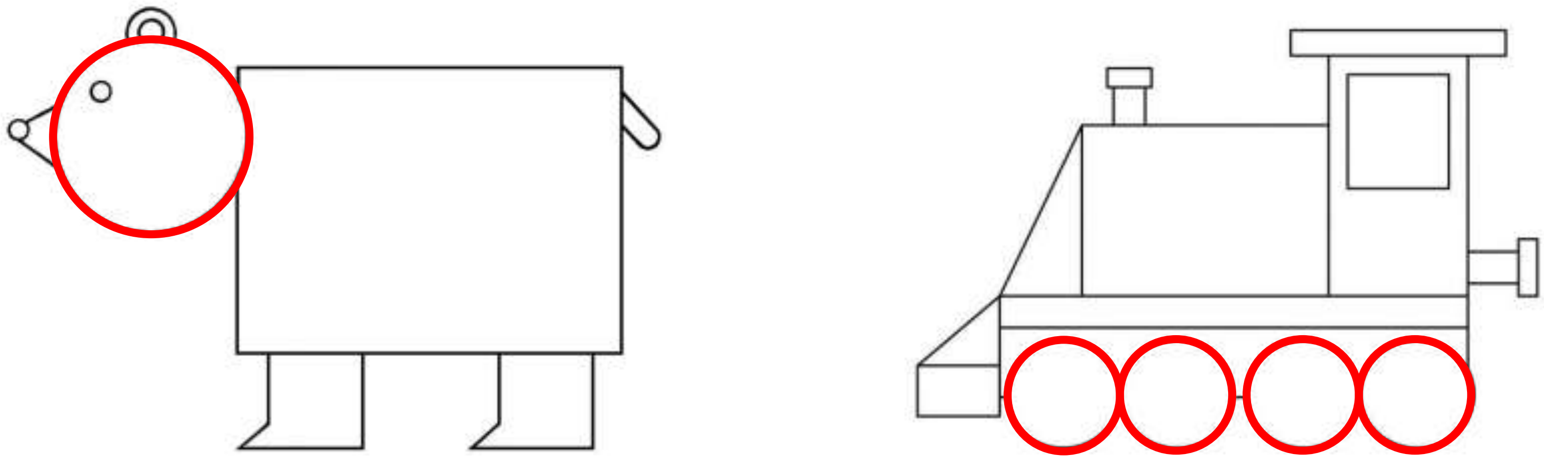
CNNs Take Advantage of Repeated, Hierarchical Structure in Images





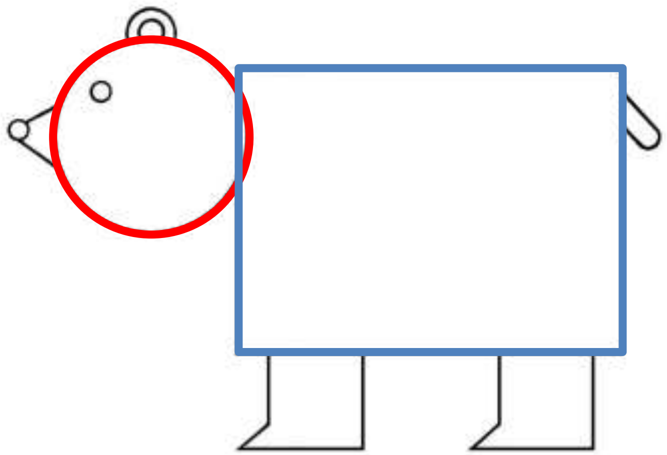
Mid-level structure: shapes

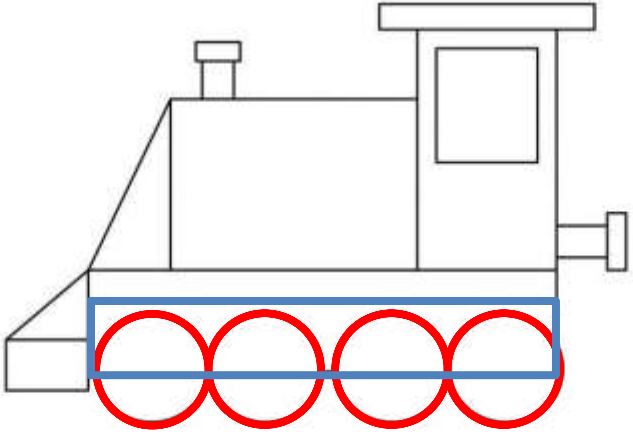
CNNs Take Advantage of Repeated, Hierarchical Structure in Images



Mid-level structure: shapes

CNNs Take Advantage of Repeated, Hierarchical Structure in Images

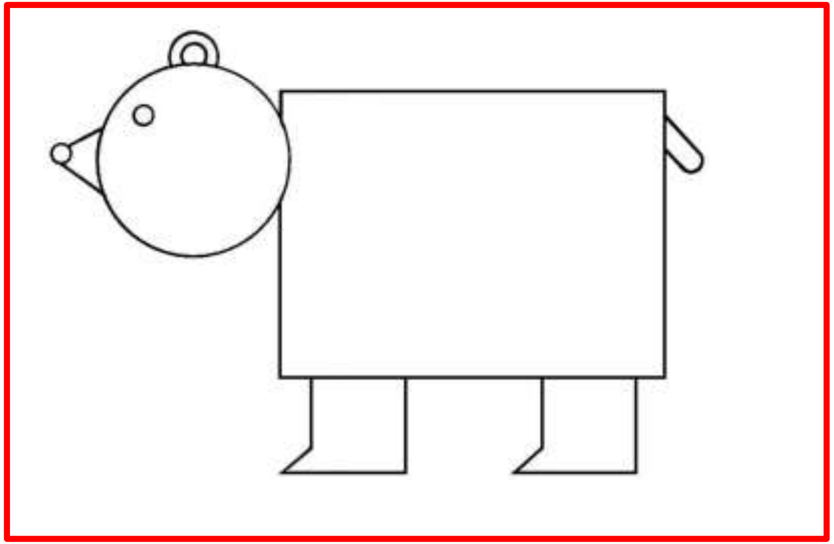




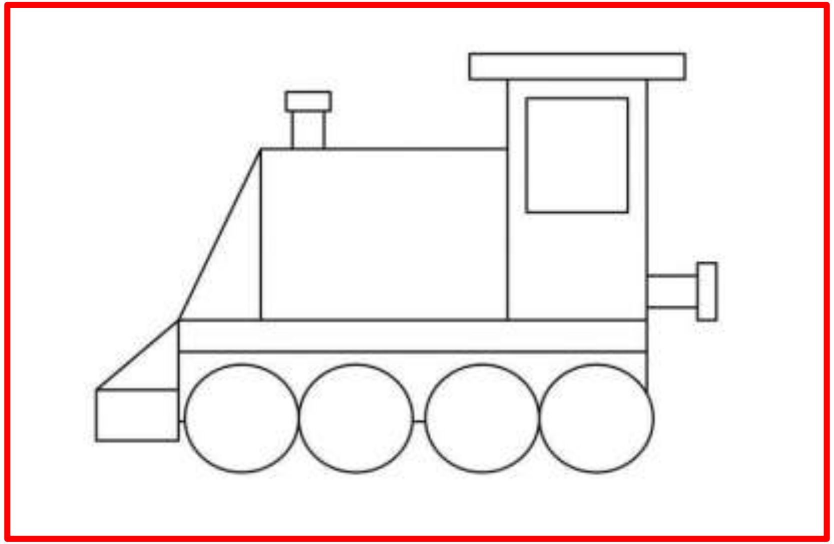
High-level structure: groups of shapes

CNNs Take Advantage of Repeated, Hierarchical Structure in Images

Bear

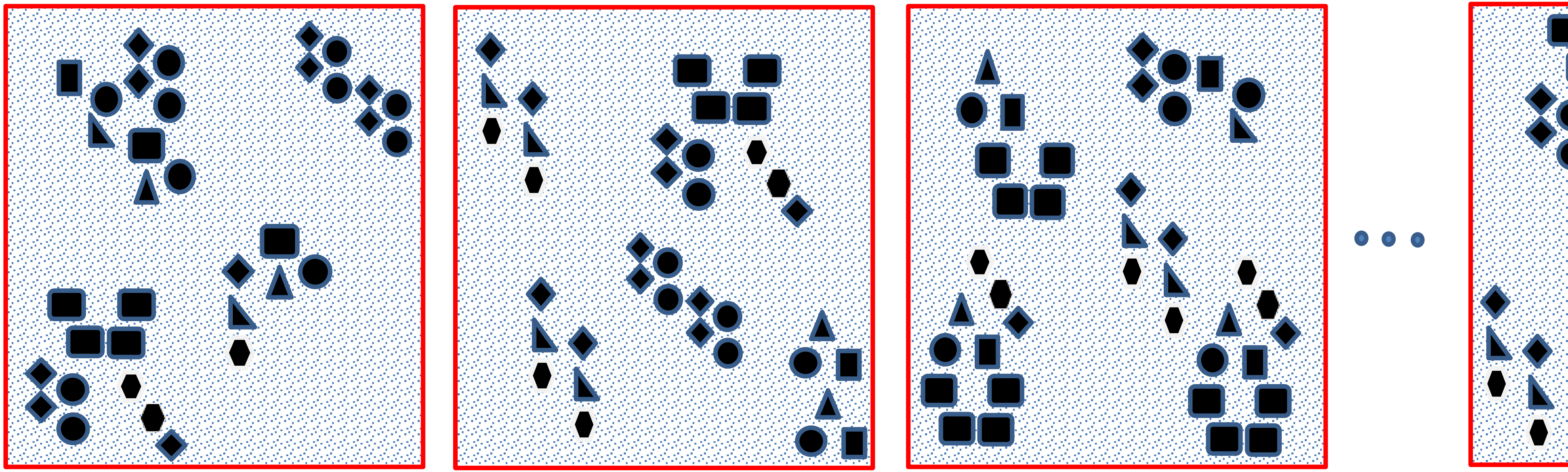


Train

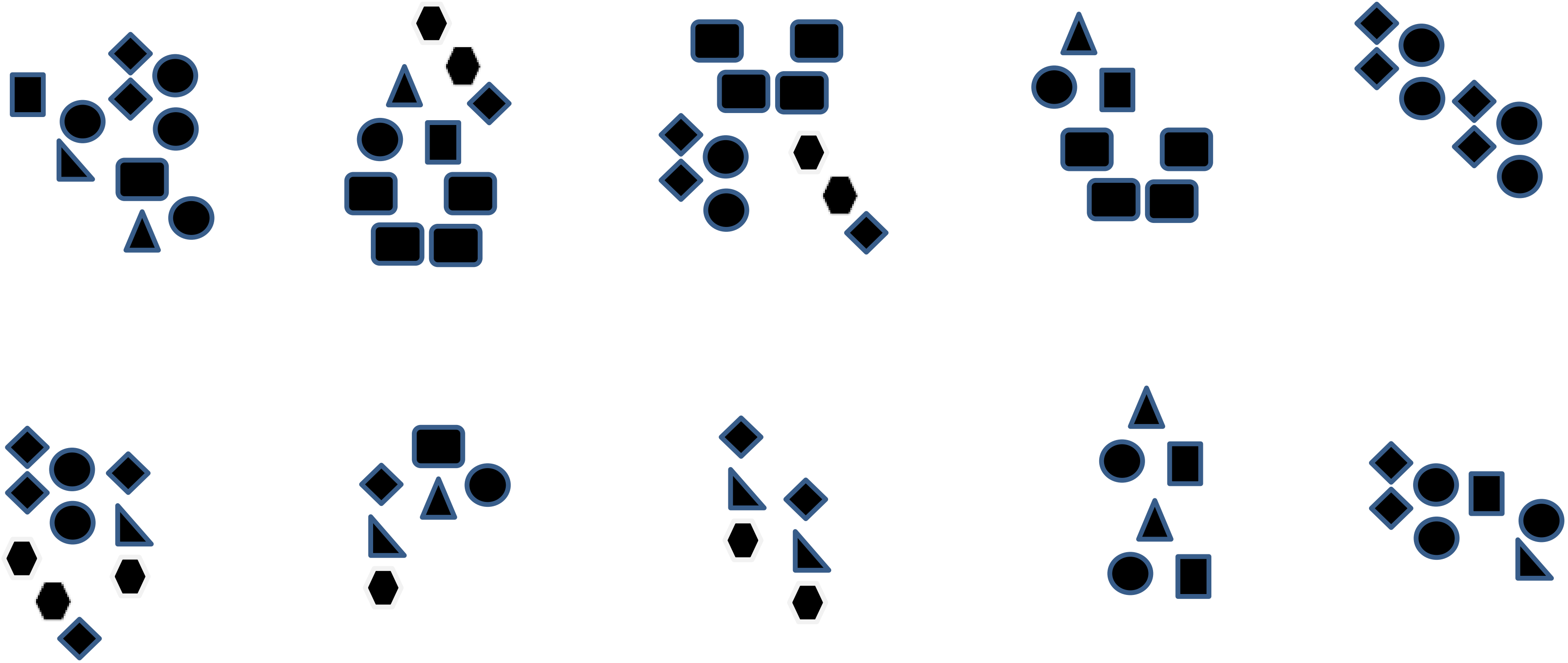


High-level structure: groups of shapes → objects

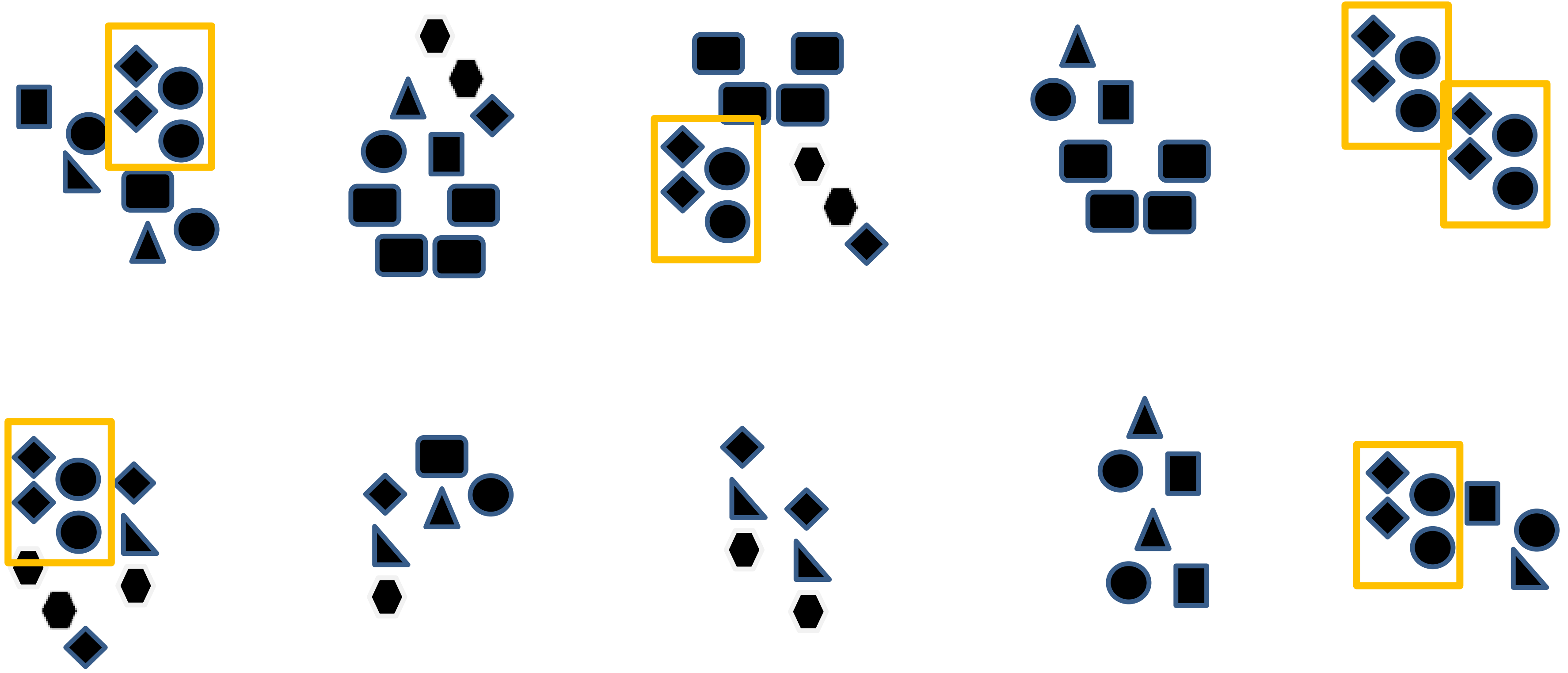
Consider a Set of "Toy" Images,
for illustration of how this structure can be extracted by an algorithm



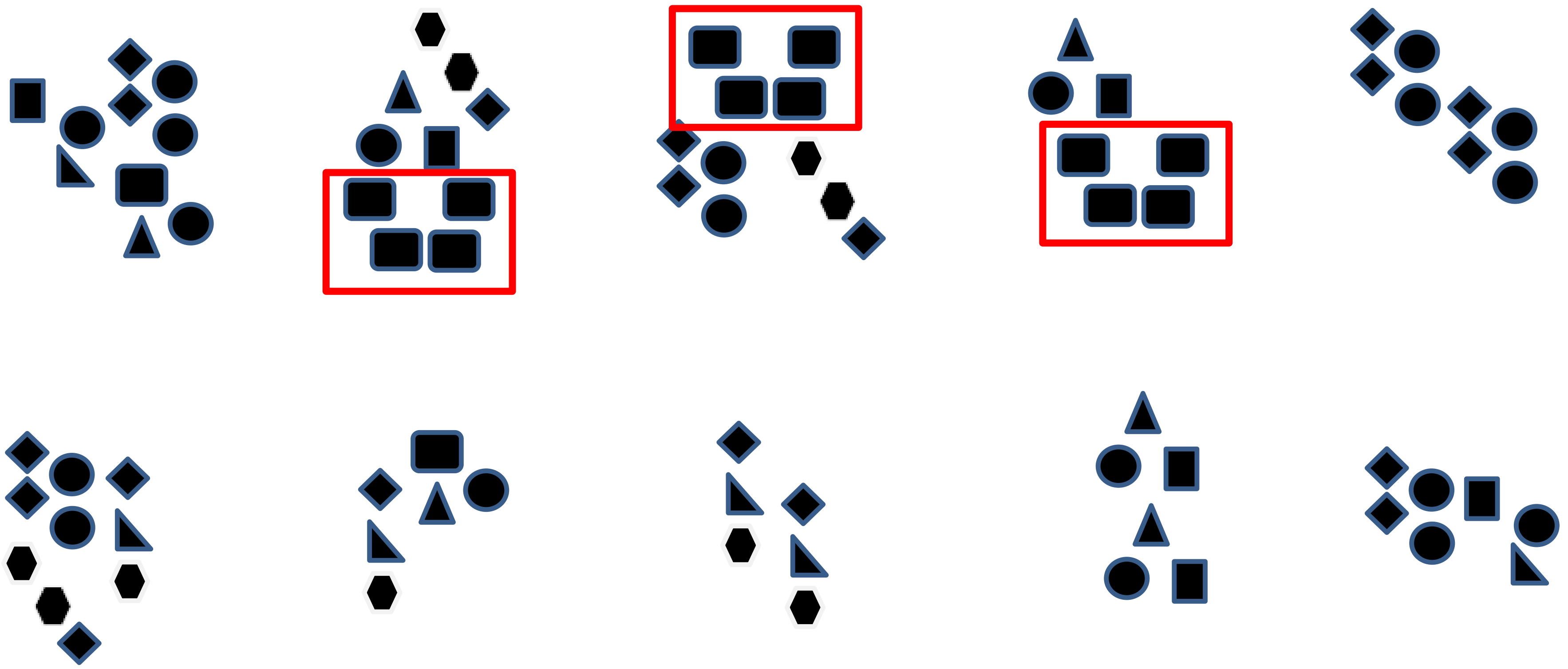
High-Level Motifs/Structure



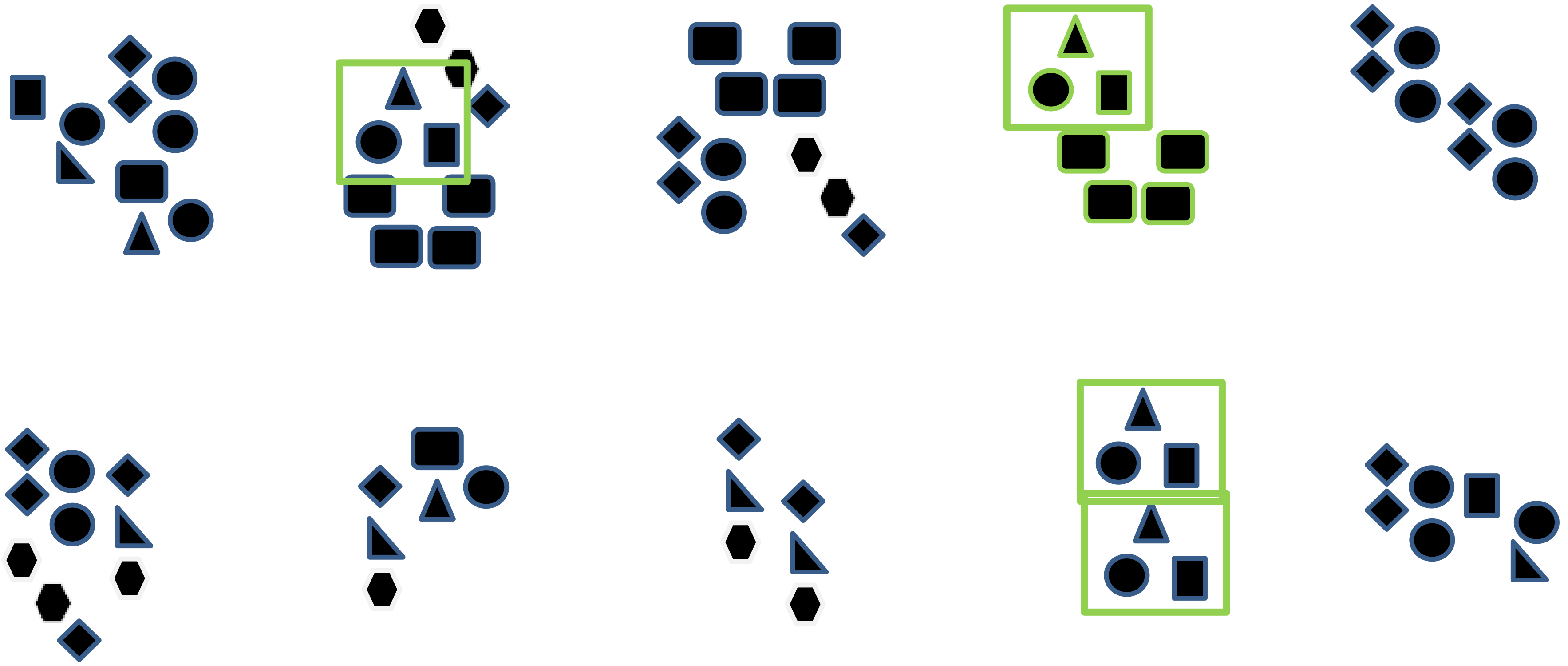
Shared Substructure Within Motifs



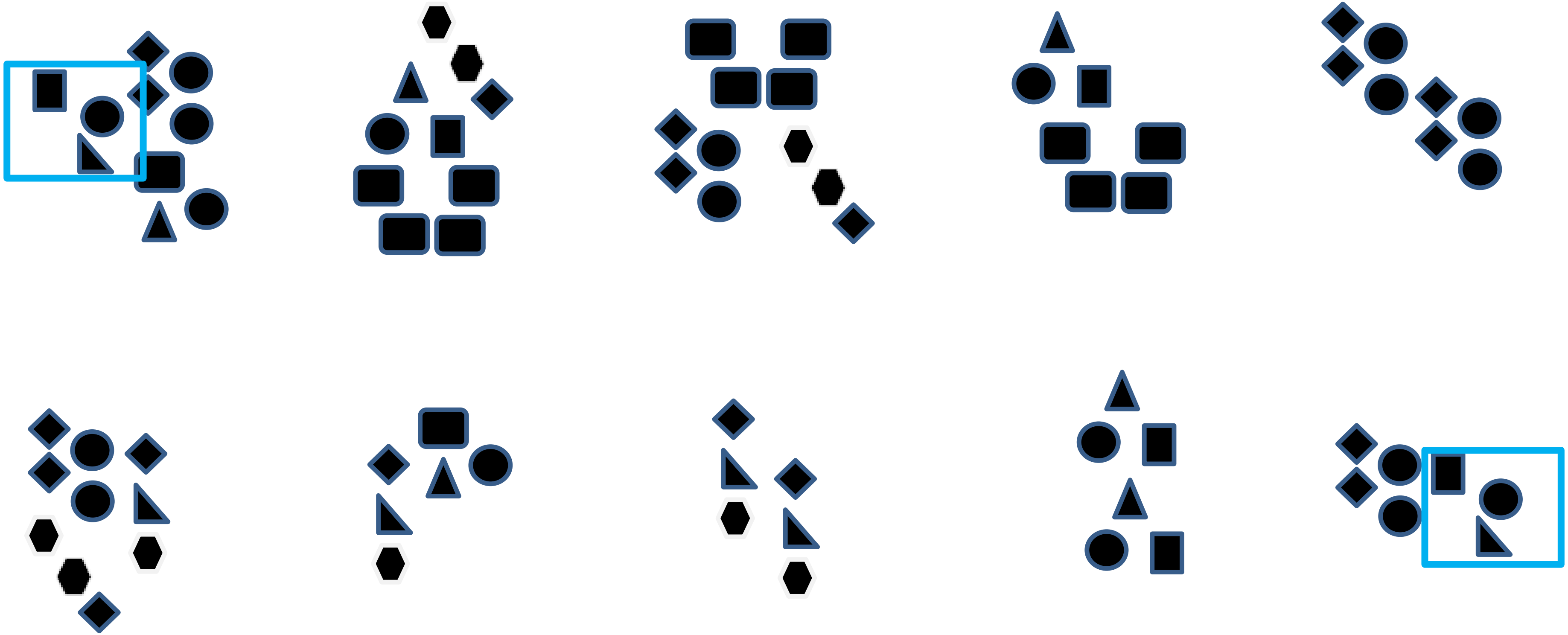
Shared Substructure Within Motifs



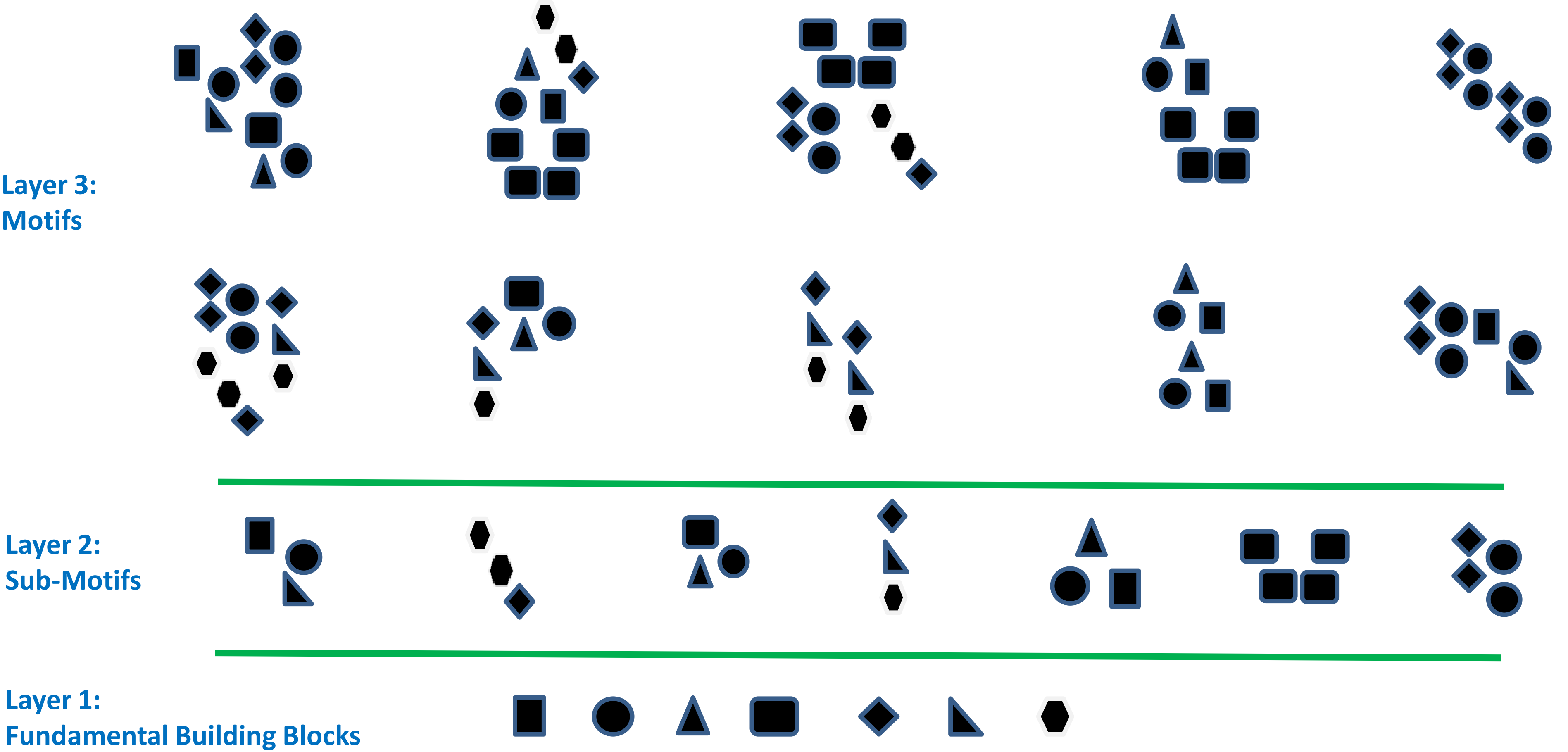
Shared Substructure Within Motifs



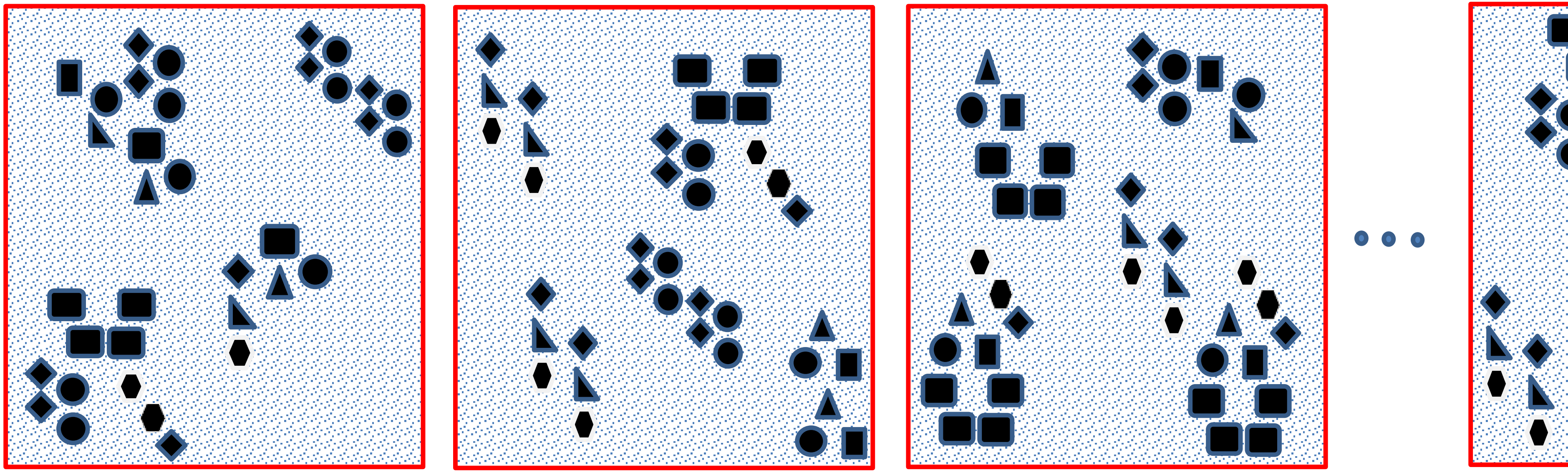
Shared Substructure Within Motifs



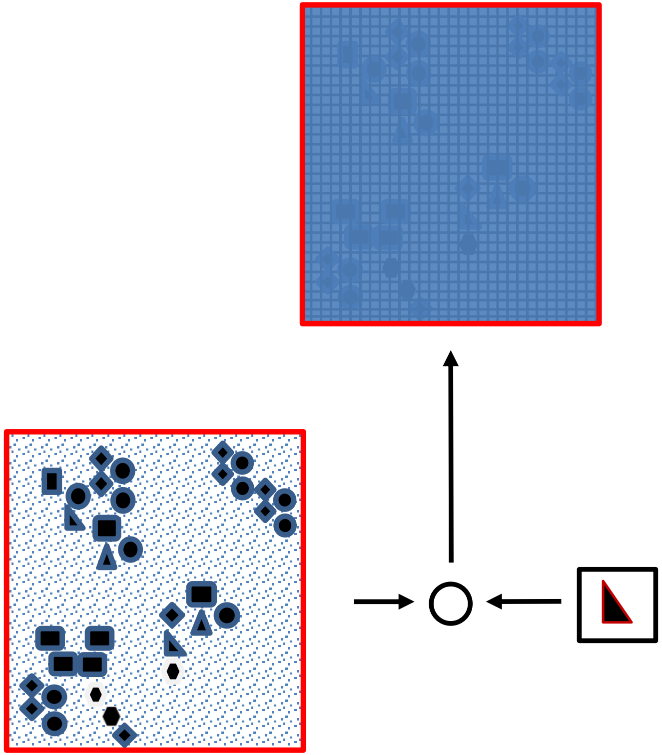
Hierarchical Representation of Images



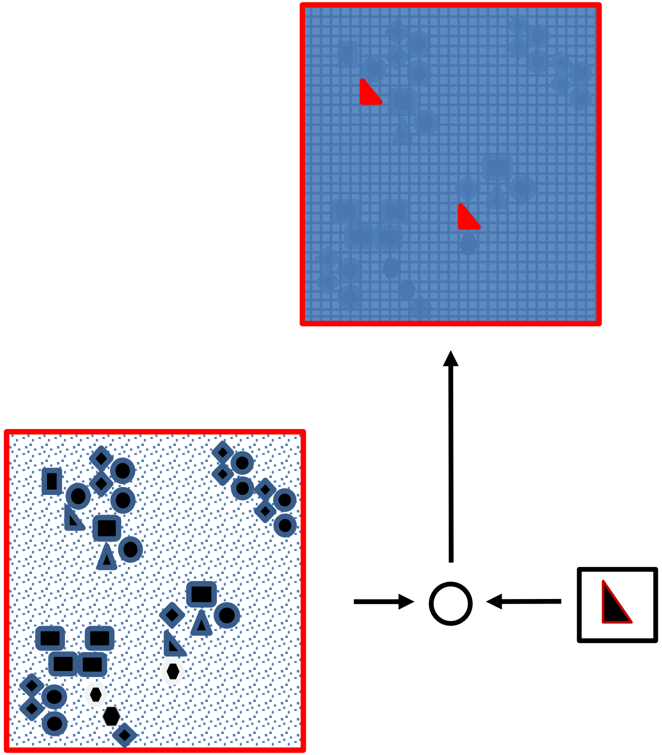
Recall the Data/Images



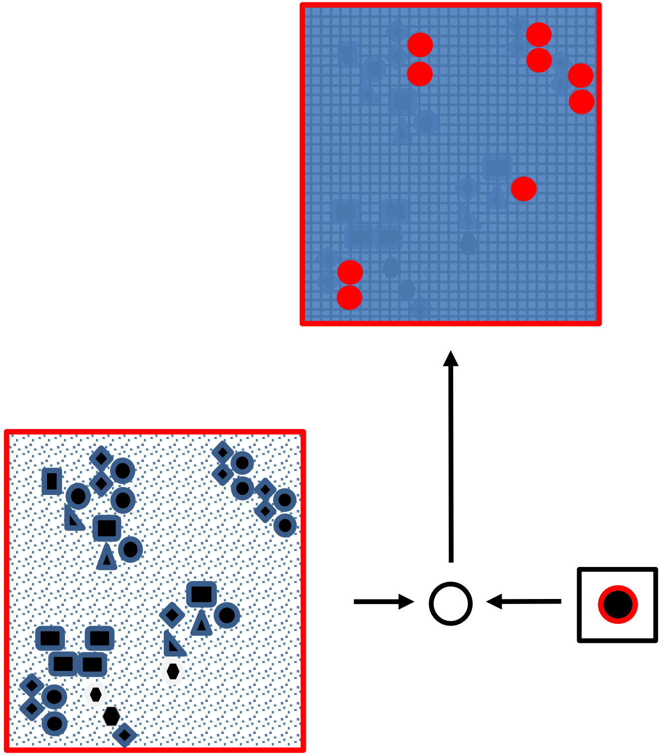
Convolutional Filter



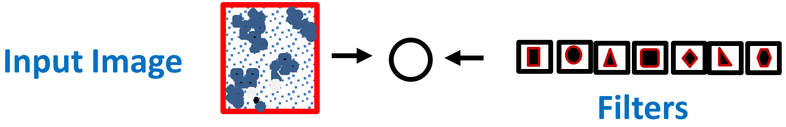
Convolutional Filter

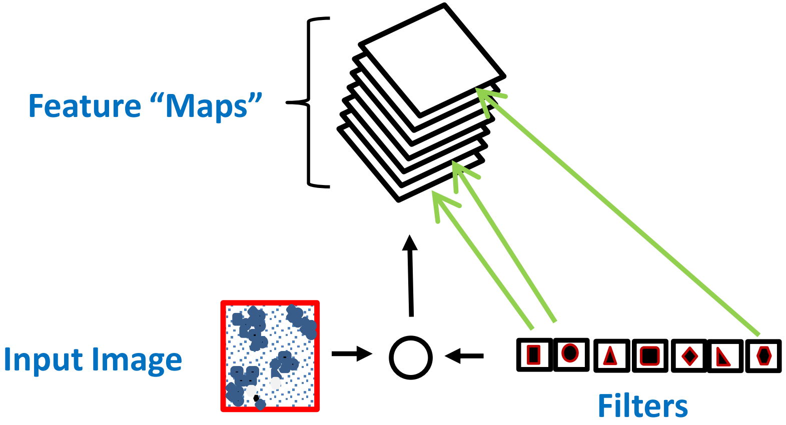


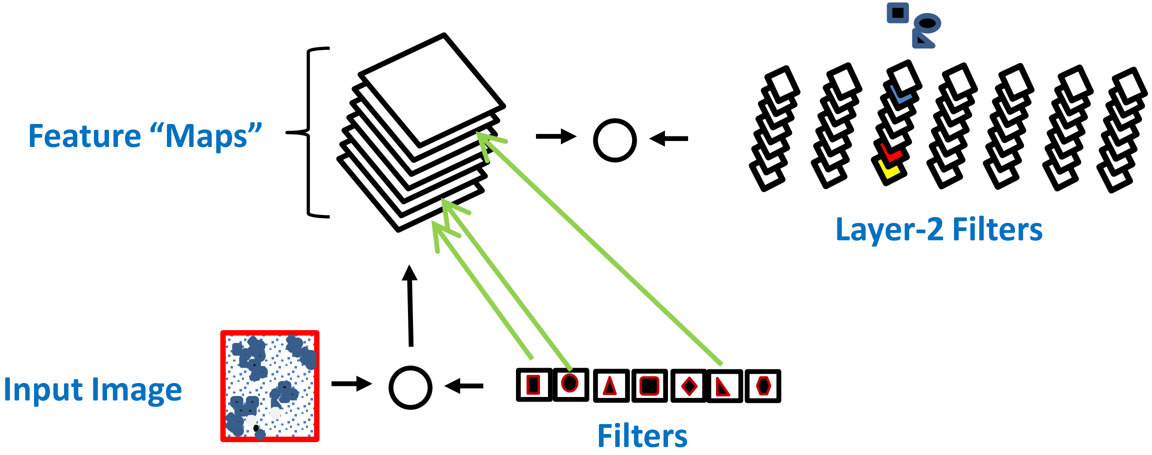
Convolutional Filter

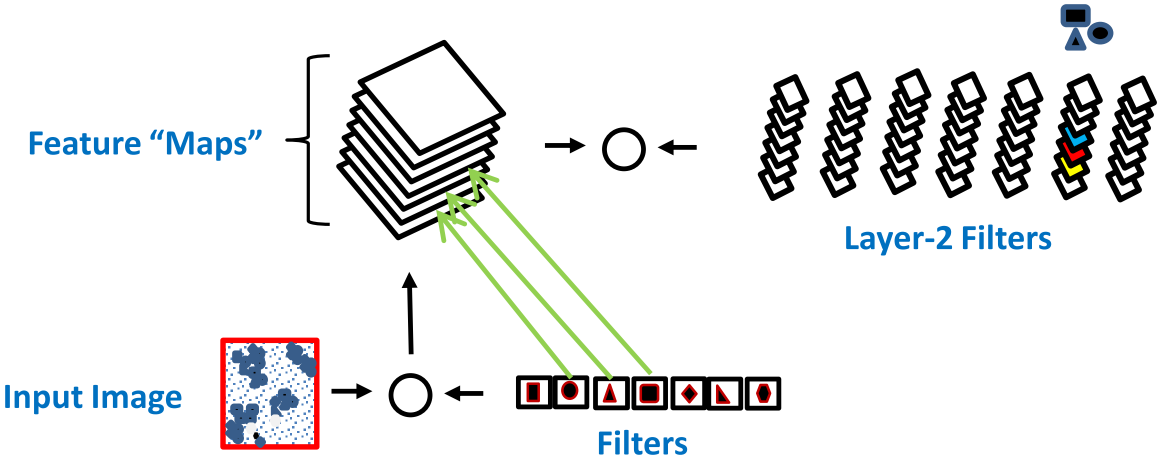


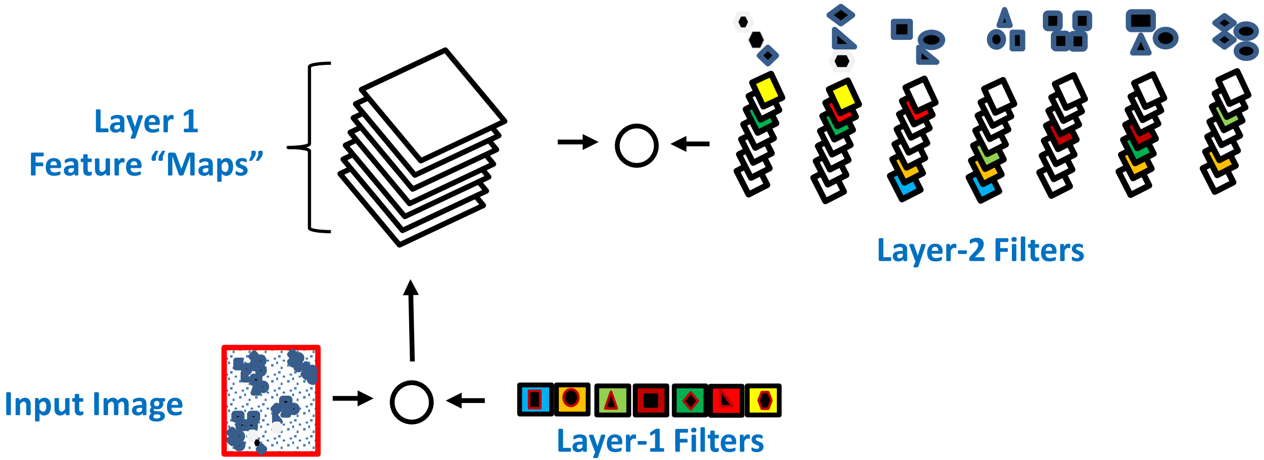
Multiple Filters, One for Each Building Block

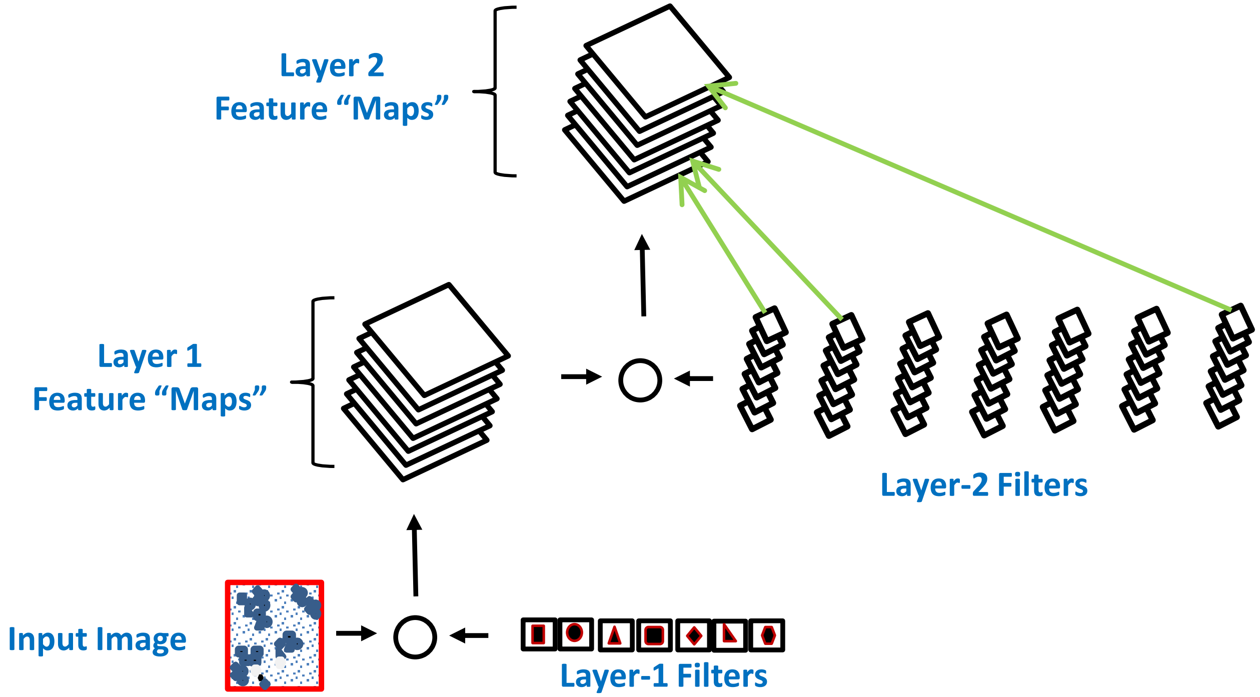


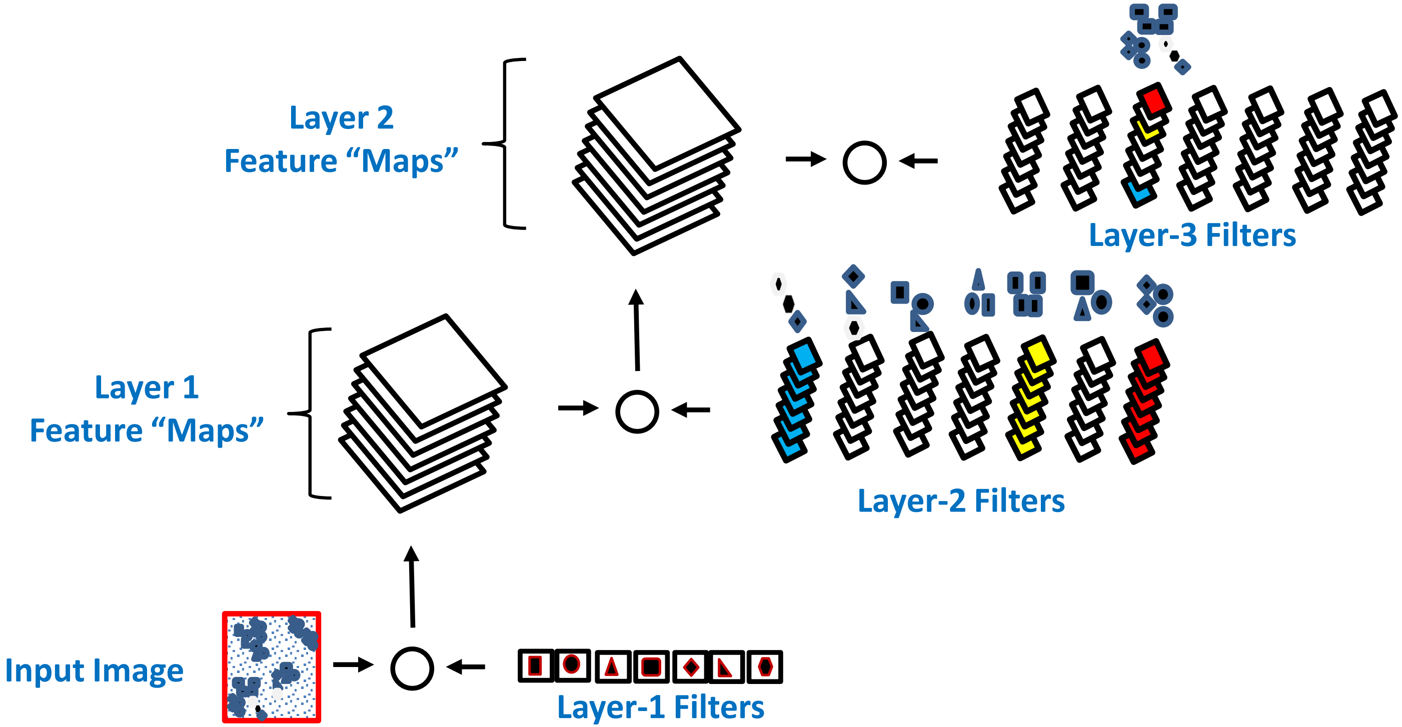

Feature "Maps"
Input Image
Filters

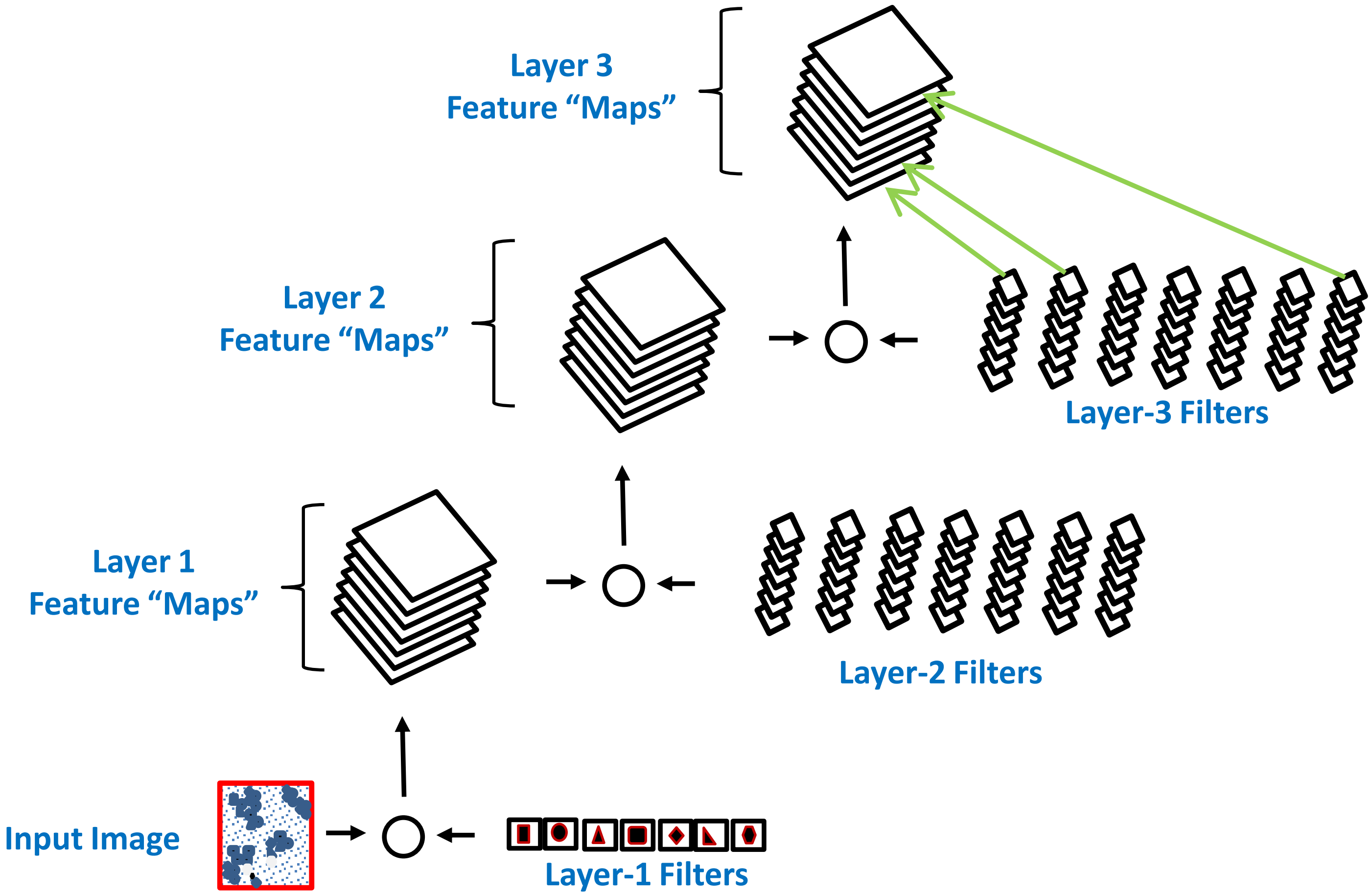

Feature “Maps”
Layer-2 Filters
Input Image
Filters


Feature "Maps"
Layer-2 Filters
Input Image
Filters

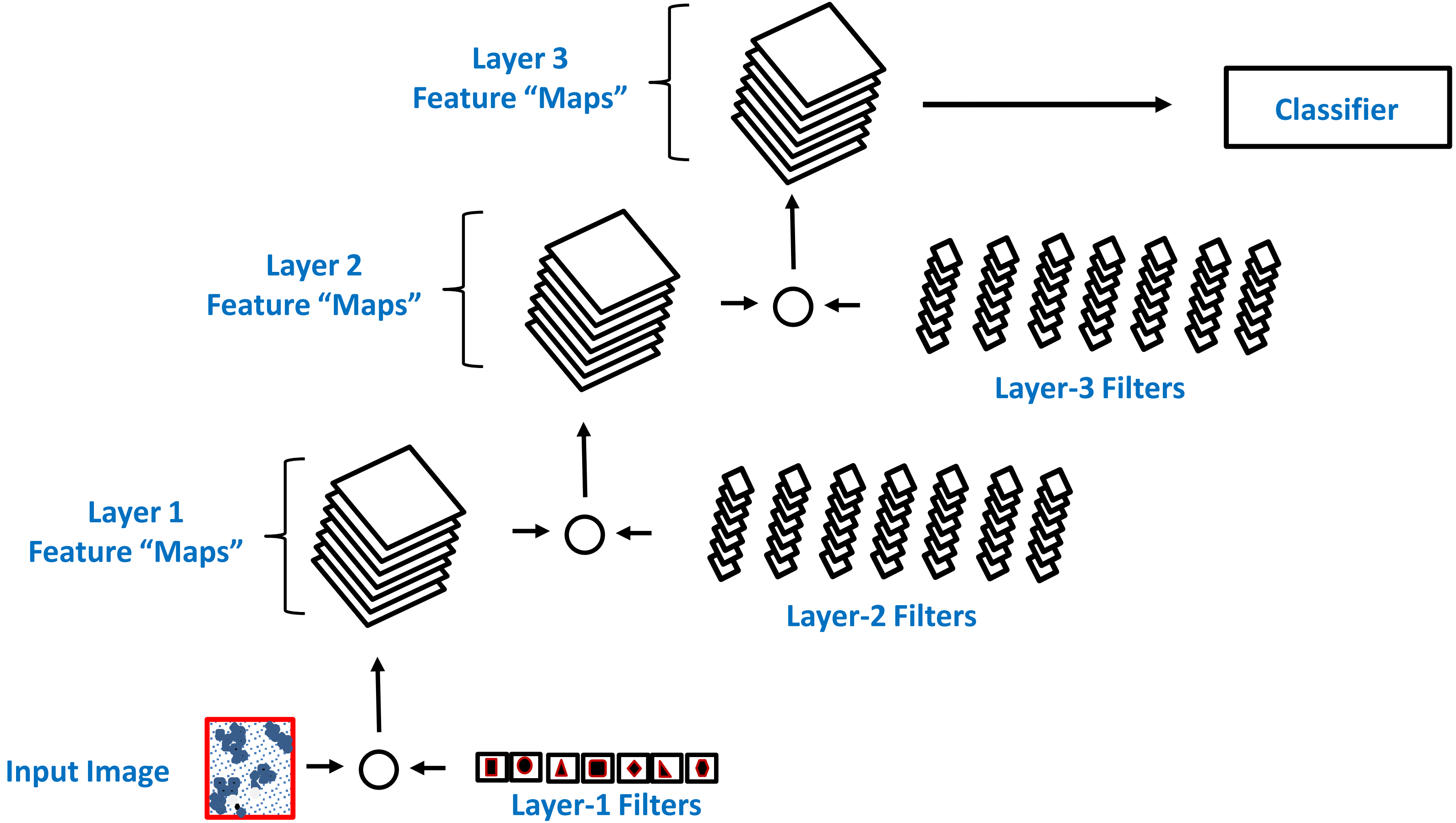

Layer 1
Feature “Maps”
Layer-2 Filters
Input Image
Layer-1 Filters


Layer 2
Feature "Maps"
Layer 1
Feature "Maps"
Layer-2 Filters
Input Image
Layer-1 Filters

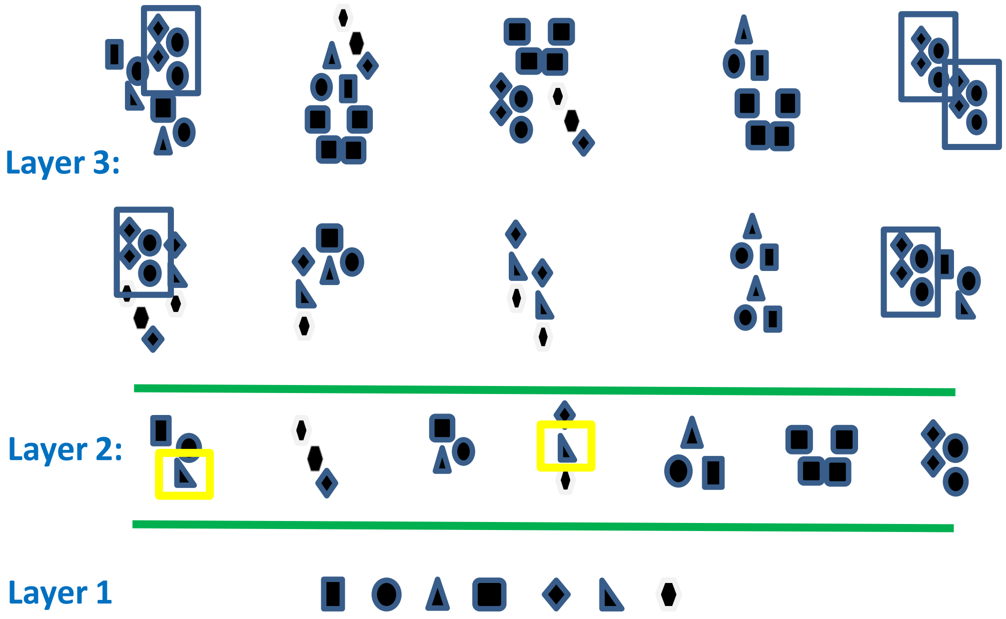

Layer 2
Feature "Maps"
Layer-3 Filters
Layer 1
Feature "Maps"
Layer-2 Filters
Input Image
Layer-1 Filters


Layer 3
Feature “Maps”
Layer 2
Feature “Maps”
Layer-3 Filters
Layer 1
Feature “Maps”
Layer-2 Filters
Input Image
Layer-1 Filters

Deep CNN Architecture



Advantage of Hierarchical Features?



- By learning and sharing statistical similarities within high-level motifs, we better leverage all training data

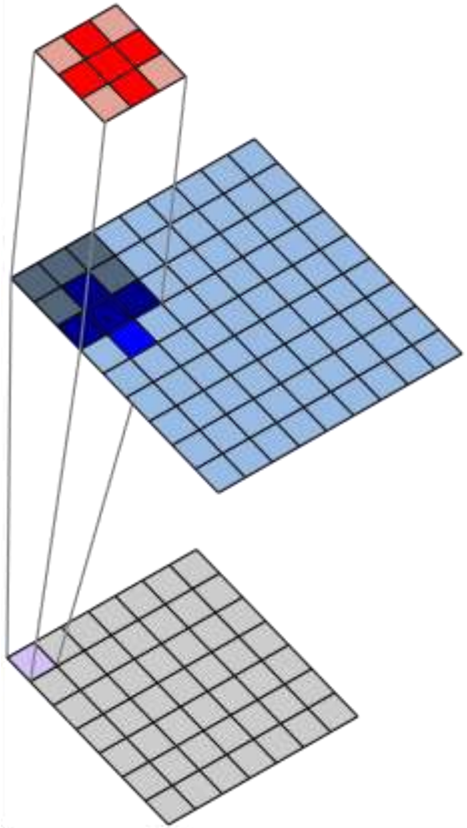
- If we do not use such a hierarchy, top-level motifs would be learned in isolation of each other

2D Spatial Convolution

Filter

×

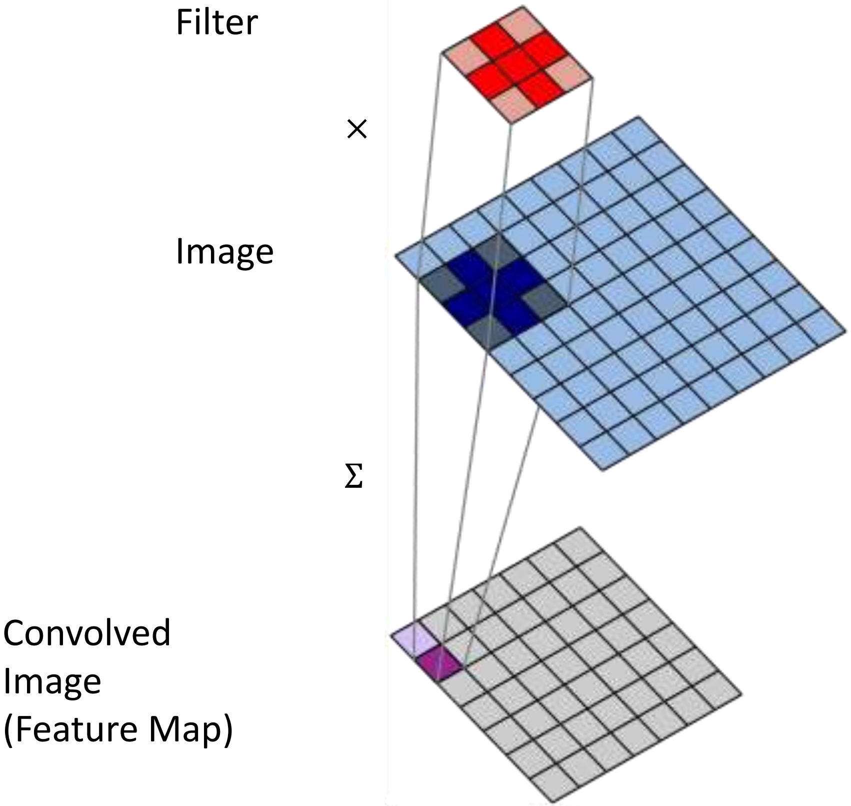
Image



Σ

Convolved
Image
(Feature Map)

2D Spatial Convolution



2D Spatial Convolution

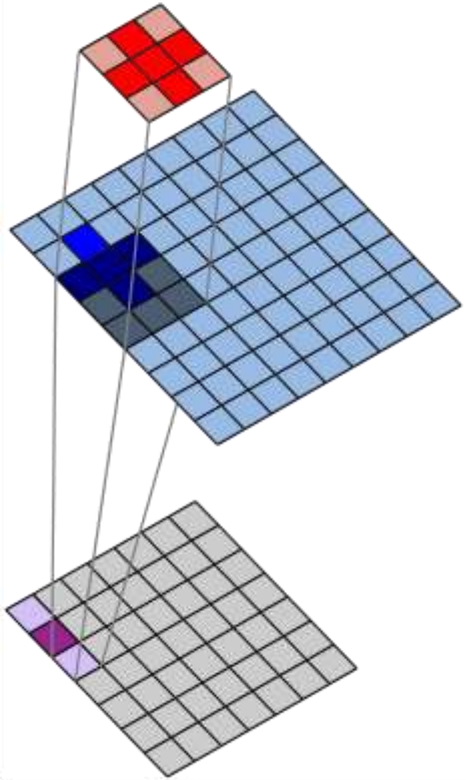
Filter

$\times$

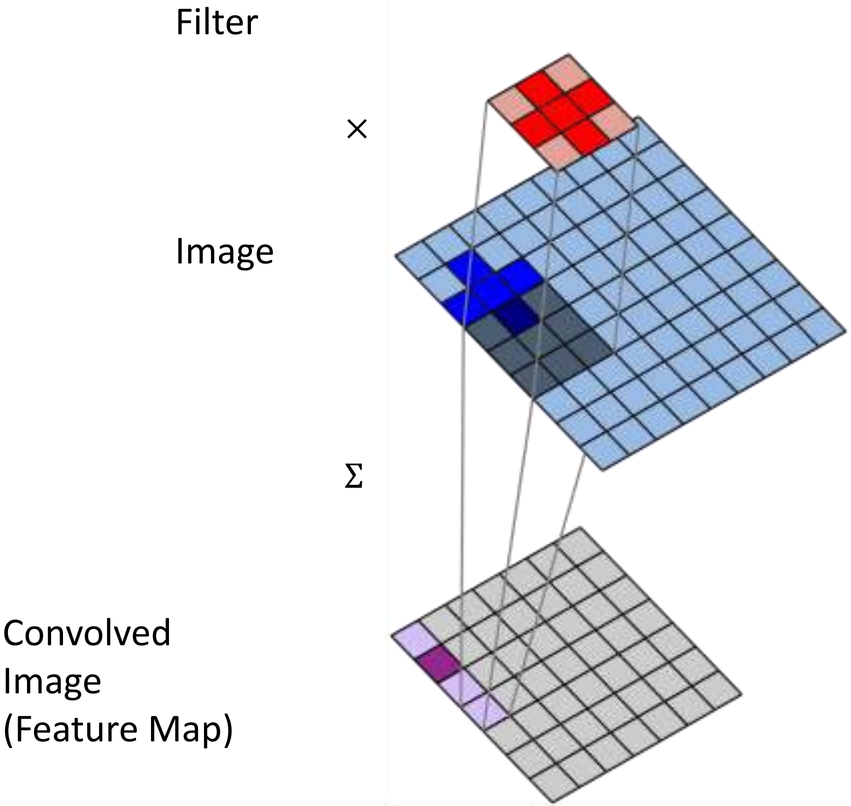
Image

Σ

Convolved
Image
(Feature Map)



2D Spatial Convolution



2D Spatial Convolution

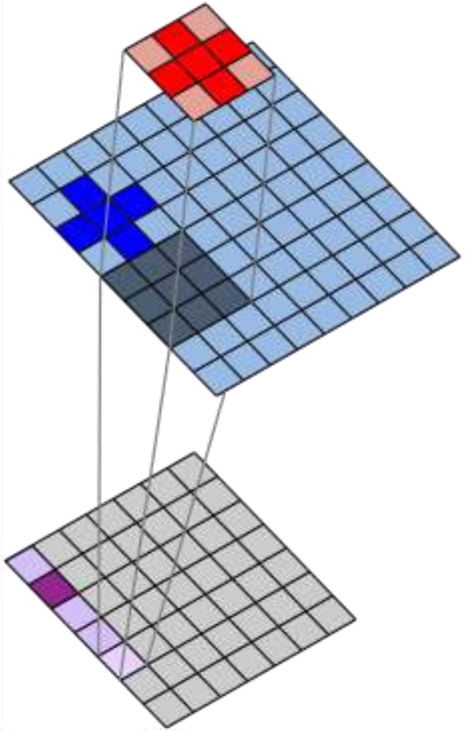
Filter

×

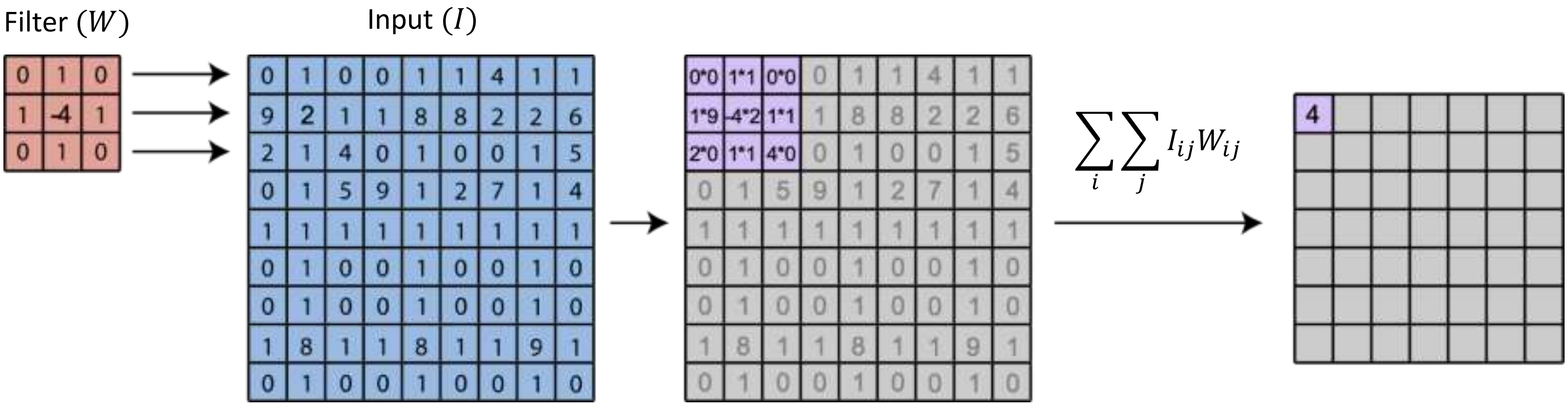
Image

Σ

Convolved Image (Feature Map)



2D Spatial Convolution

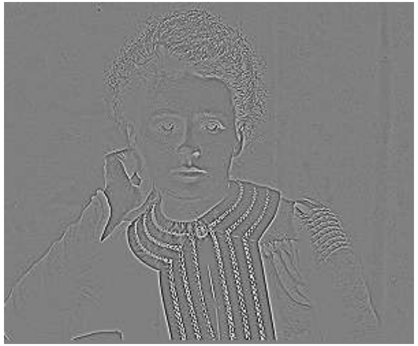


2D Spatial Convolution

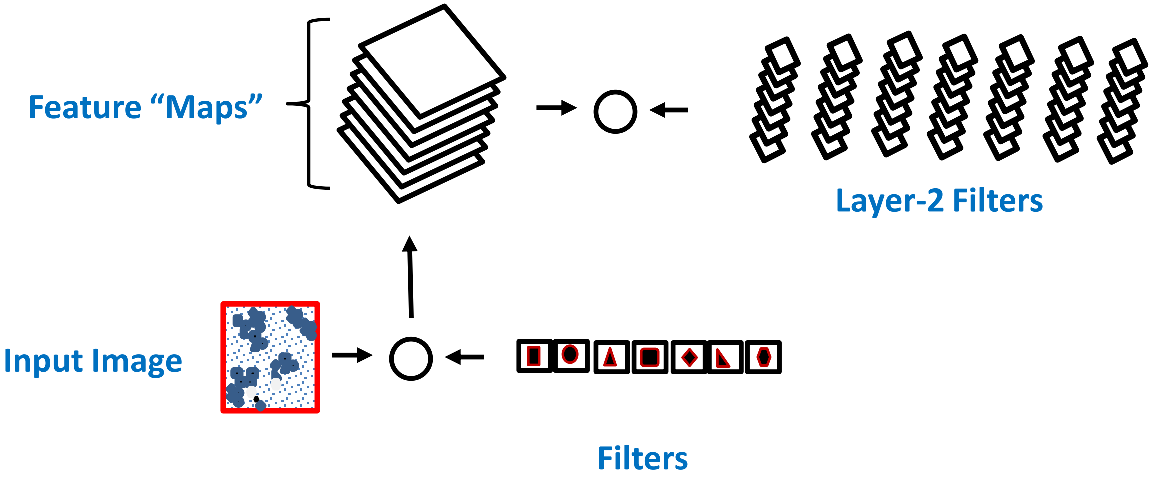
Filter (W)

$$\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

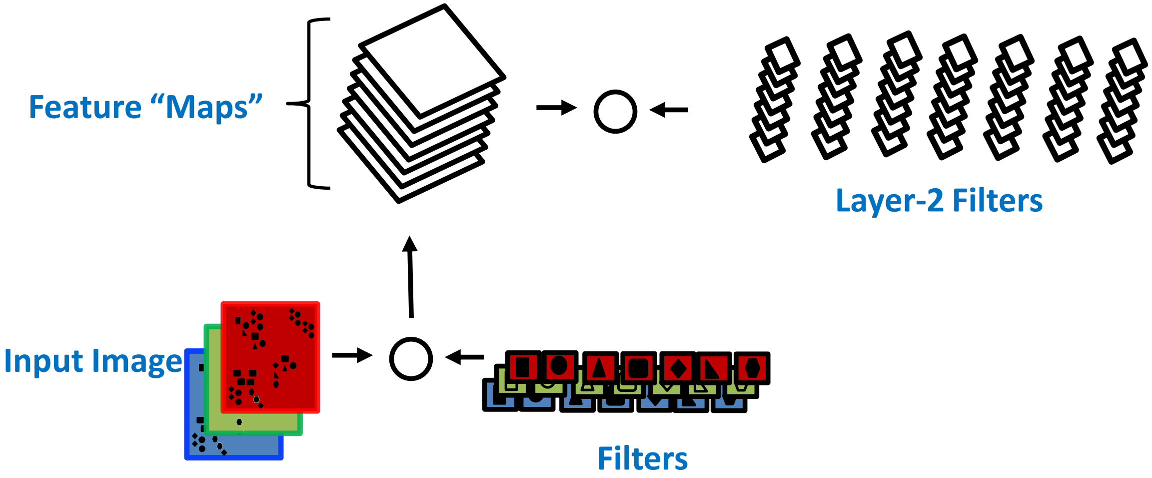
Input (I)



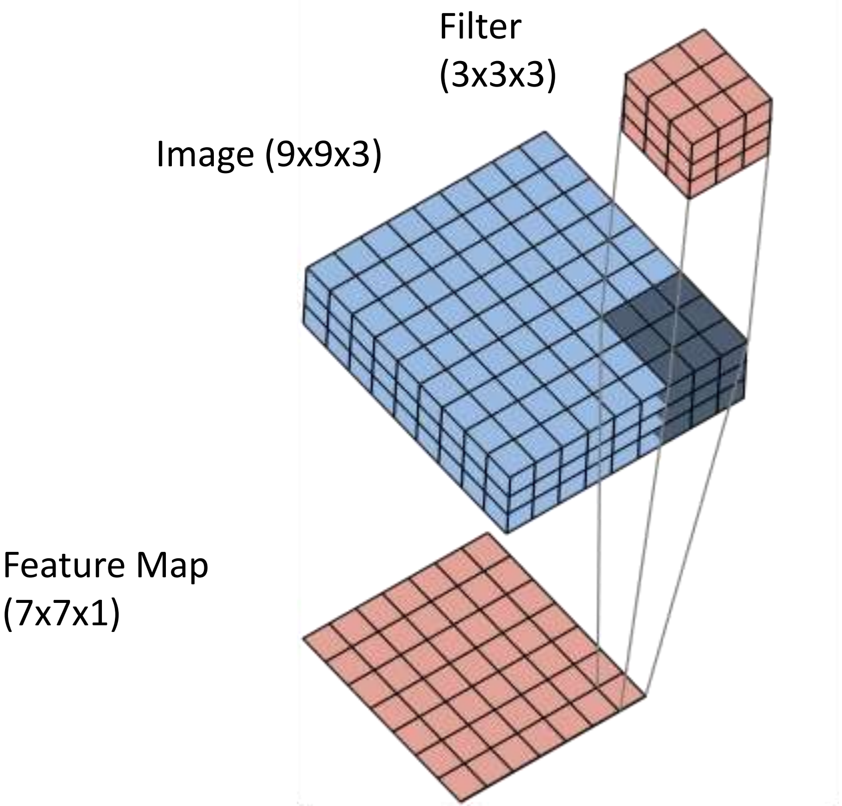
Filters Operate Over Input Volumes



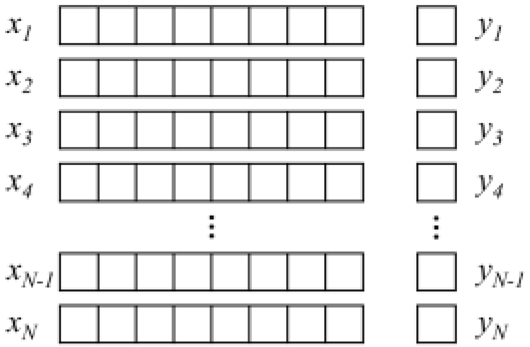
Filters Operate Over Input Volumes



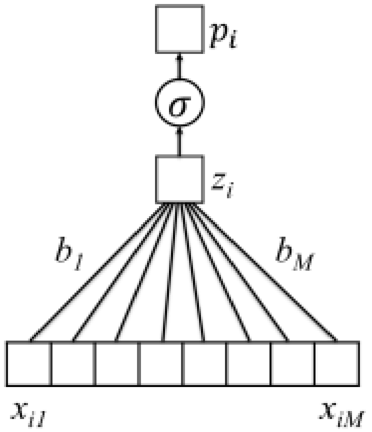
Filters Operate Over Input Volumes



Given Labeled Training Images, How do we Learn the Parameters of the CNN?

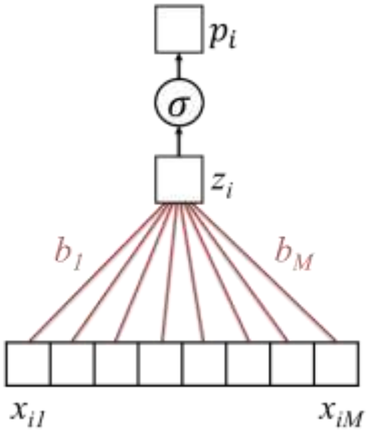


Training Set



$$p_i = \sigma(b_0 + b_1 x_{i1} + b_2 x_{i2} + \cdots + b_M x_{iM})$$

Untrained Logistic Regression Model (or “Network”)



$$b = (b_0, \ldots b_M)$$

Trained Model (with learned parameters)

Given Labeled Training Images, How do we Learn the Parameters of the CNN?

+ CNN Architecture → Trained CNN

Architecture (specified) vs Parameters (learned)

Architecture:

- Number of layers
- Layer types (e.g. convolutional, pooling, fully connected)
- Number of filters in each layer
- Shape and size of filters

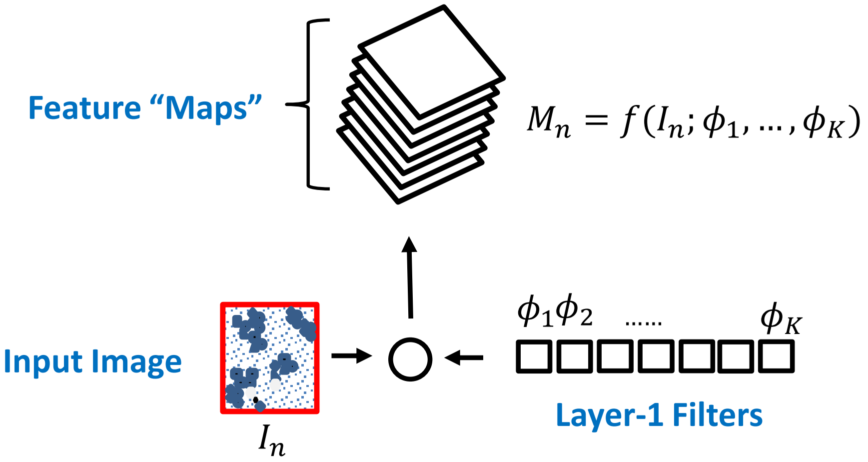
Use 3x3 filters
In layer 1

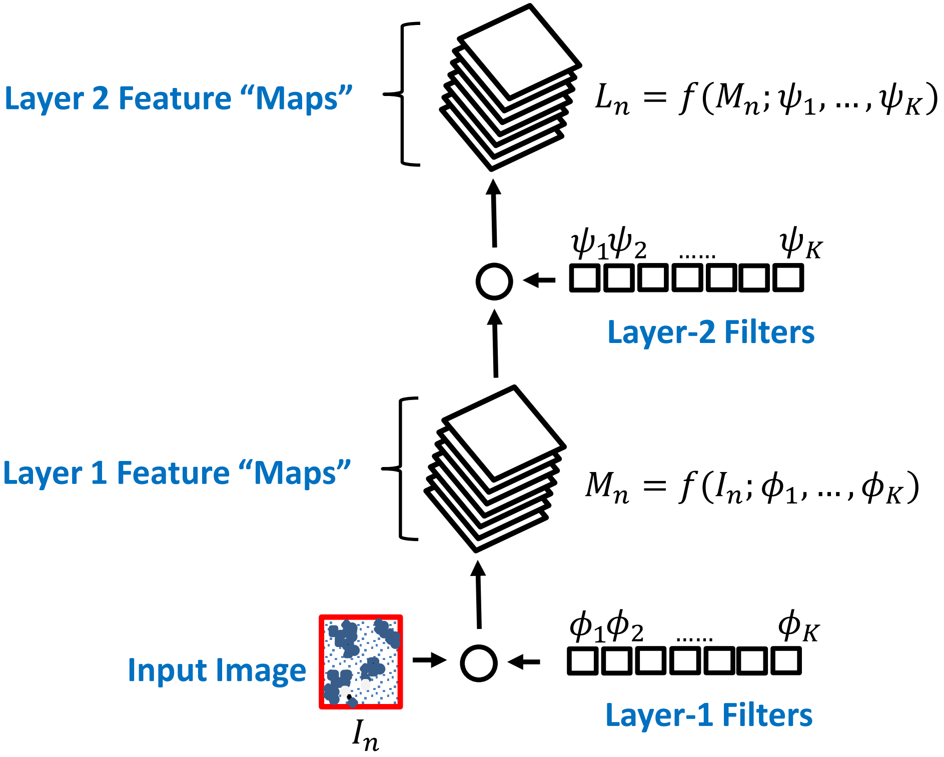
Parameters:

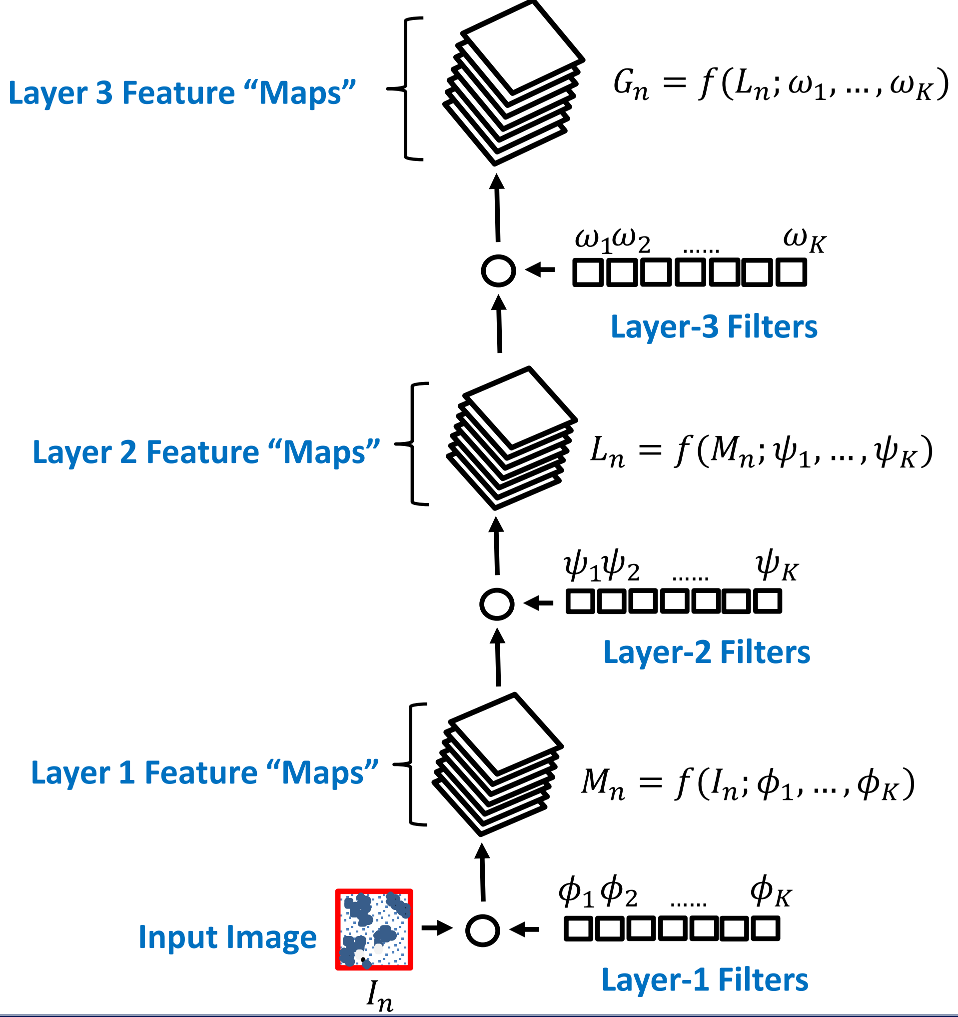
- Individual Elements of each filter
- Parameters of other layers

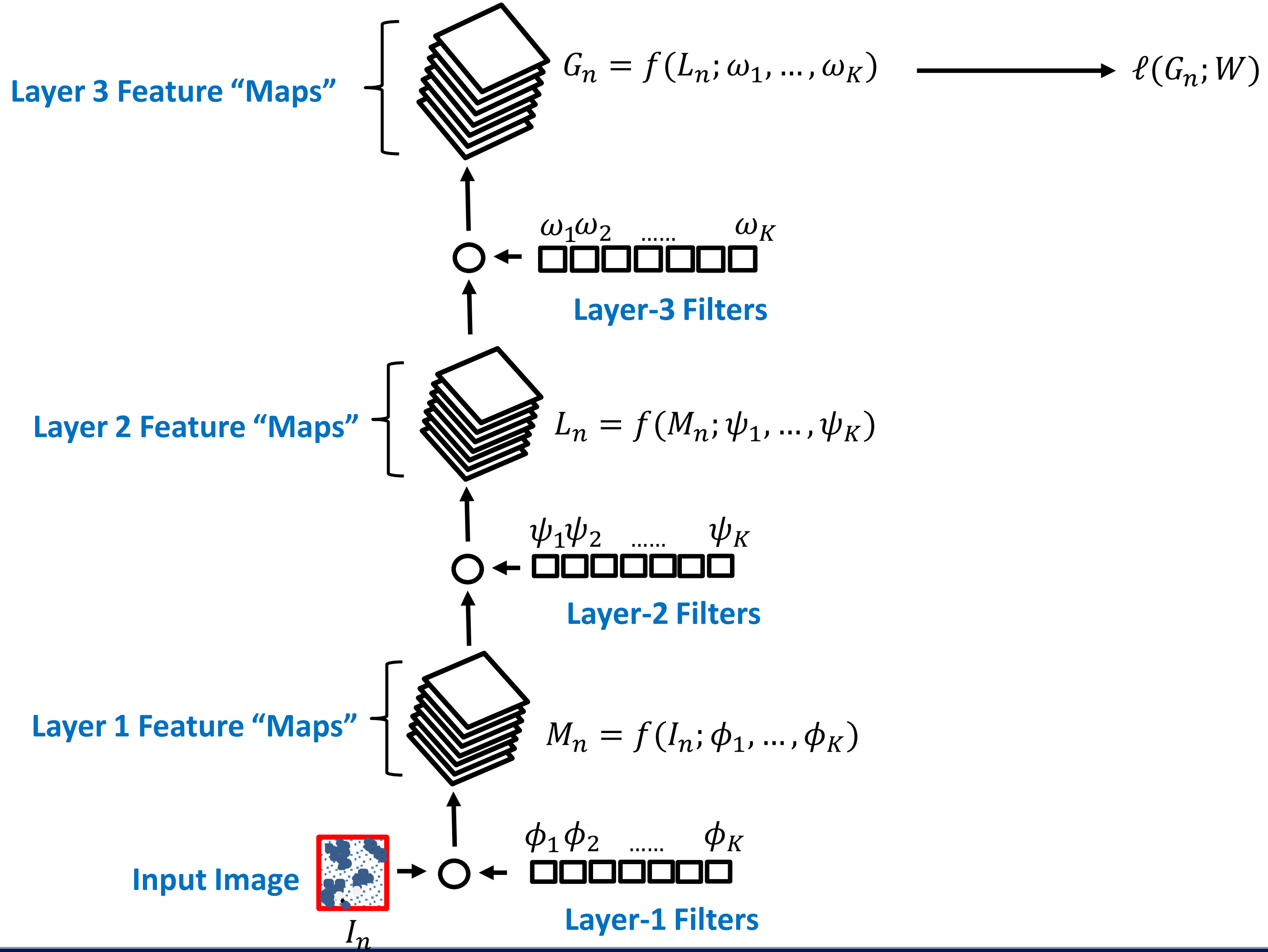
Learn values of
Each layer 1 filter

-1	1	-1
1	-1	1
-1	1	-1

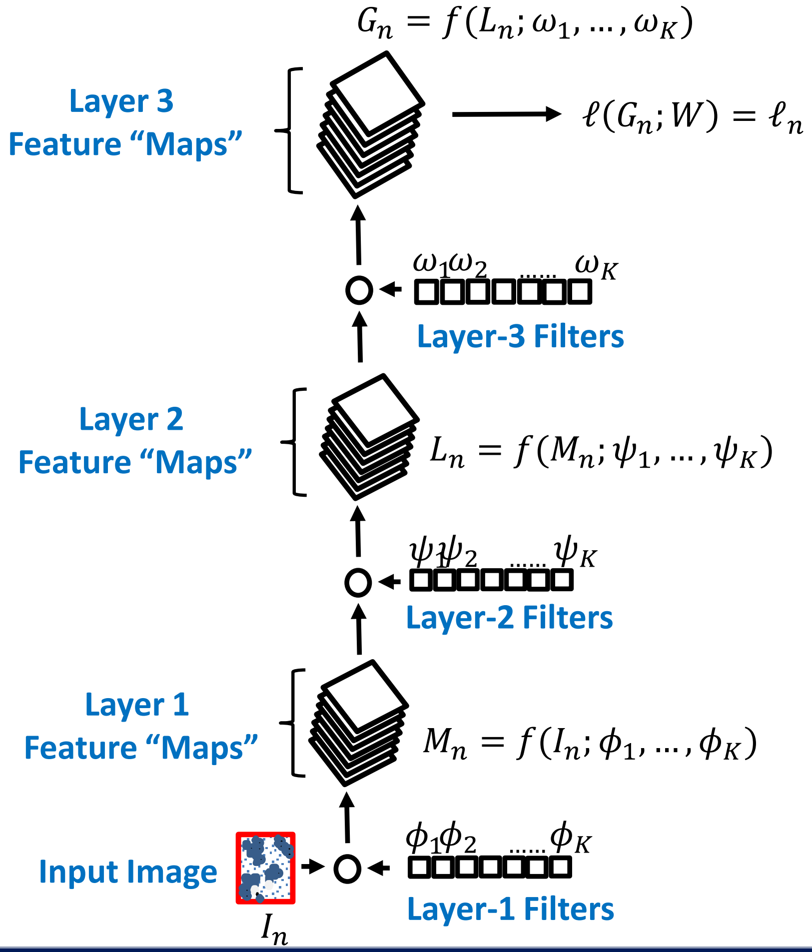

Feature "Maps"
$M_n = f(I_n; \phi_1, \ldots, \phi_K)$
Input Image
I_n
$\phi_1 \phi_2$ ...... ϕ_K
Layer-1 Filters


Layer 2 Feature "Maps"
$L_n = f(M_n; \psi_1, \ldots, \psi_K)$
$\psi_1 \psi_2$ ...... ψ_K
Layer-2 Filters
Layer 1 Feature "Maps"
$M_n = f(I_n; \phi_1, \ldots, \phi_K)$
$\phi_1 \phi_2$ ...... ϕ_K
Input Image
I_n
Layer-1 Filters


Layer 3 Feature "Maps"
$G_n = f(L_n; \omega_1, \dots, \omega_K)$
$\omega_1 \omega_2$ ...... ω_K
Layer-3 Filters
Layer 2 Feature "Maps"
$L_n = f(M_n; \psi_1, \dots, \psi_K)$
$\psi_1 \psi_2$ ...... ψ_K
Layer-2 Filters
Layer 1 Feature "Maps"
$M_n = f(I_n; \phi_1, \dots, \phi_K)$
$\phi_1 \phi_2$ ...... ϕ_K
Input Image
I_n
Layer-1 Filters



Layer 3 Feature "Maps"
$G_n = f(L_n; \omega_1, \ldots, \omega_K)$
$\ell(G_n; W)$
$\omega_1 \omega_2$ ...... ω_K
Layer-3 Filters
Layer 2 Feature "Maps"
$L_n = f(M_n; \psi_1, \ldots, \psi_K)$
$\psi_1 \psi_2$ ...... ψ_K
Layer-2 Filters
Layer 1 Feature "Maps"
$M_n = f(I_n; \phi_1, \ldots, \phi_K)$
$\phi_1 \phi_2$ ...... ϕ_K
Input Image
I_n
Layer-1 Filters



- Assume we have labeled images $\{I_n, y_n\}_{n=1,N}$
- I_n is image *n*, $y_n \in \{+1, -1\}$ is associated label
- Average loss, which depends on model parameters:

$$1/N \sum_{n=1}^{N} loss(y_n, \ell_n)$$

- Find specific parameters that minimize the average loss

Summary

- Convolutional neural networks learn to recognize **high-level structure** in images by building **hierarchical representations of features**
- Features are extracted via spatial convolutions with **filters**
- Filters are learned via iterative minimization of a loss function
- Convolutional neural networks have shown capabilities beyond human performance for image analysis