Deep Convolutional Neural Nets
Part I

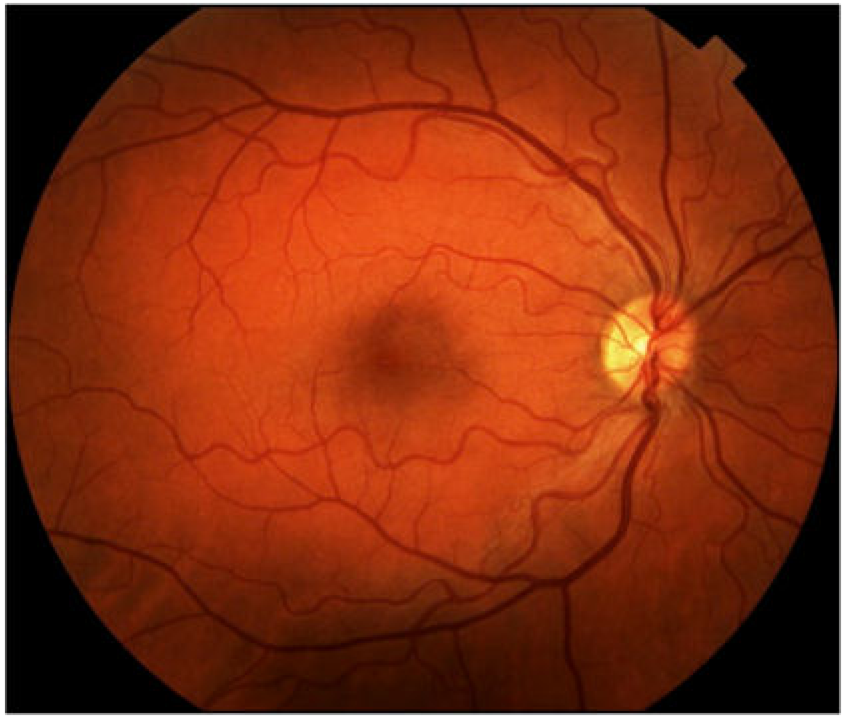
Tim Dunn

Duke MLSS 2018

Duke UNIVERSITY

Deep Learning for Image Analysis

Diabetic Retinopathy Classification

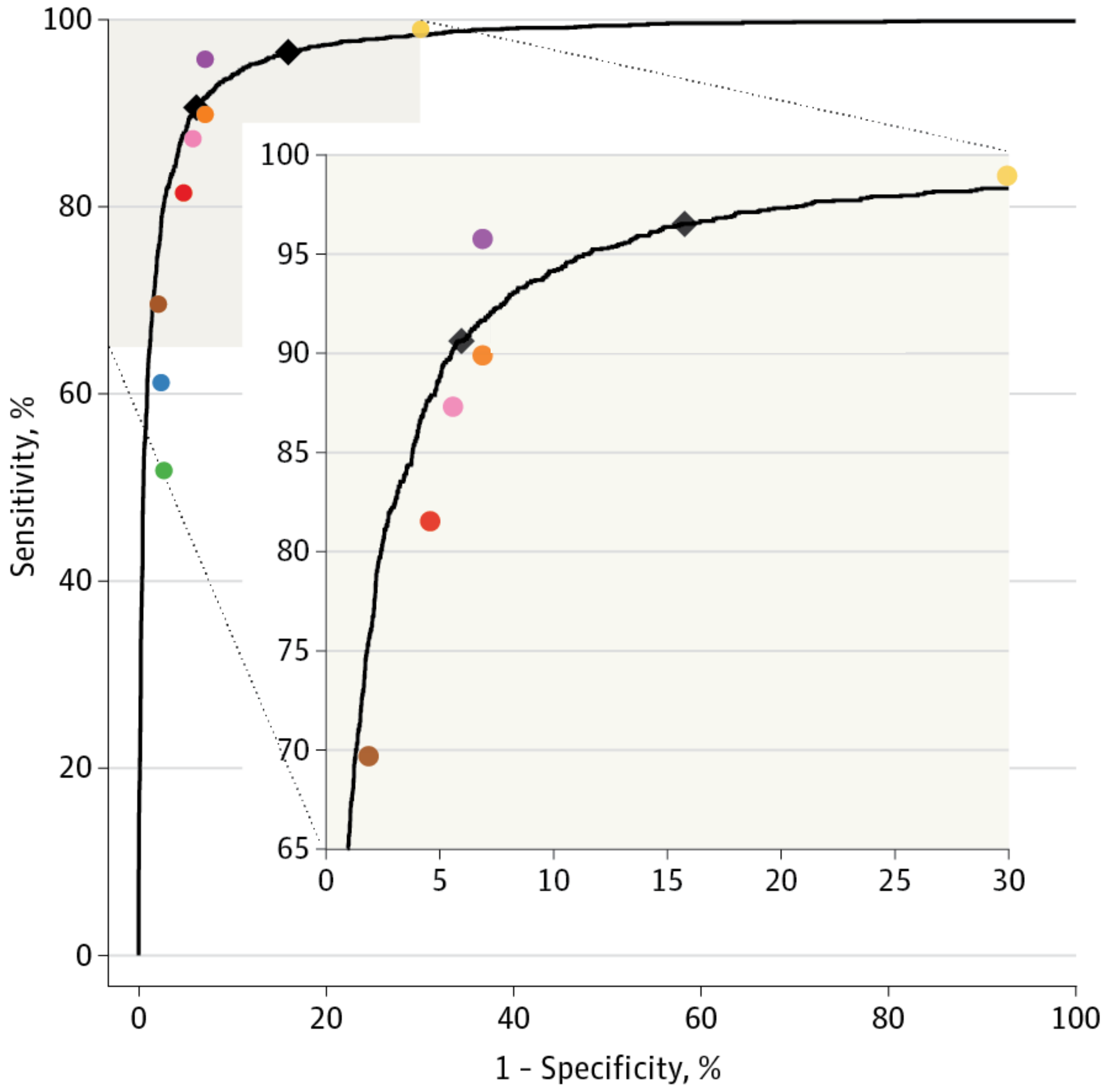


Normal Retina

Diabetic Retina

Deep Learning for Image Analysis

Diabetic Retinopathy Classification



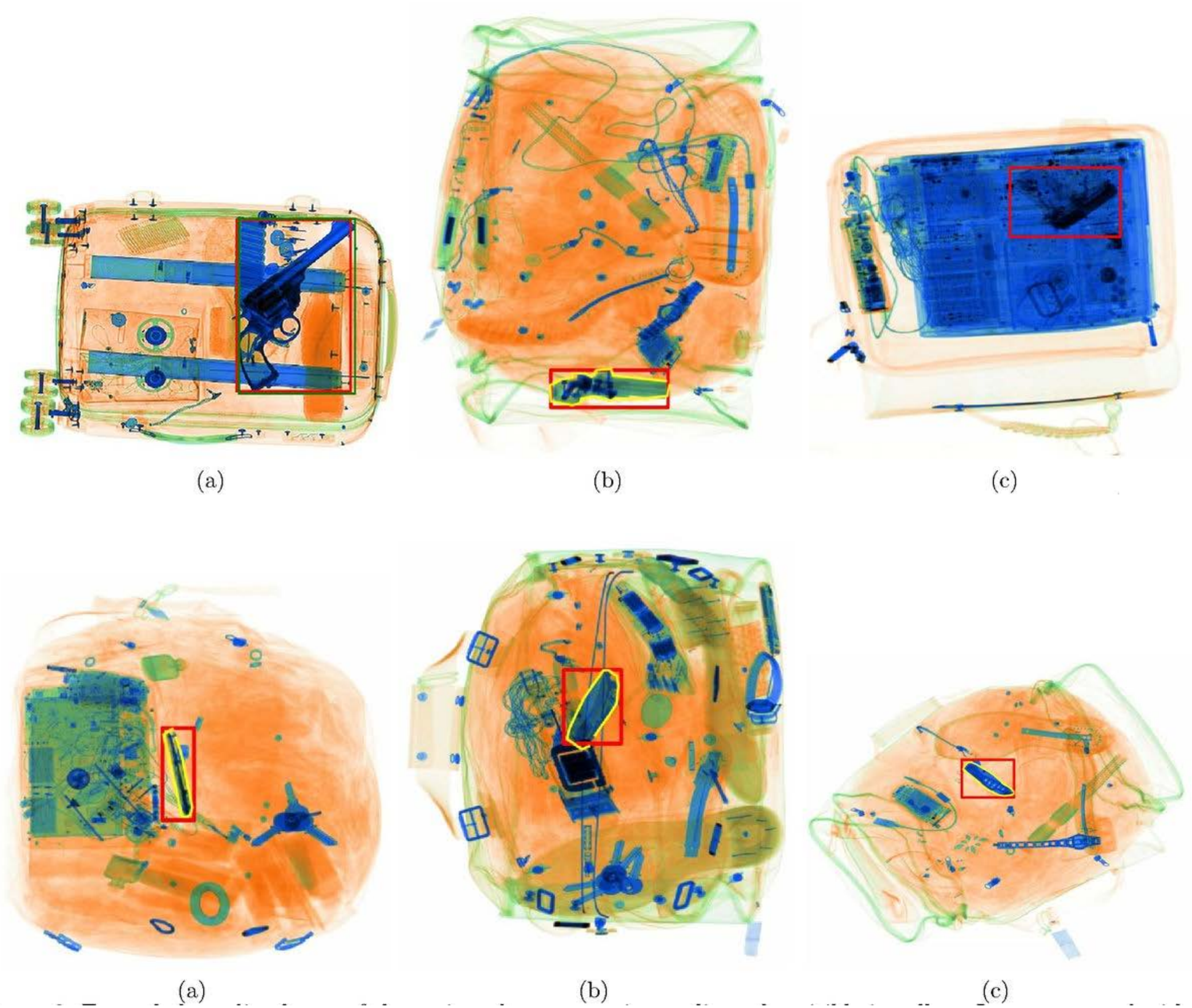
$$\text{sensitivity} = \frac{\text{number of true positives}}{\text{total number of positives in the dataset}}$$

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

Gulshan et al. *JAMA* (2016)

Deep Learning for Image Analysis

TSA Screening



Liang et al. *SPIE* (2018)

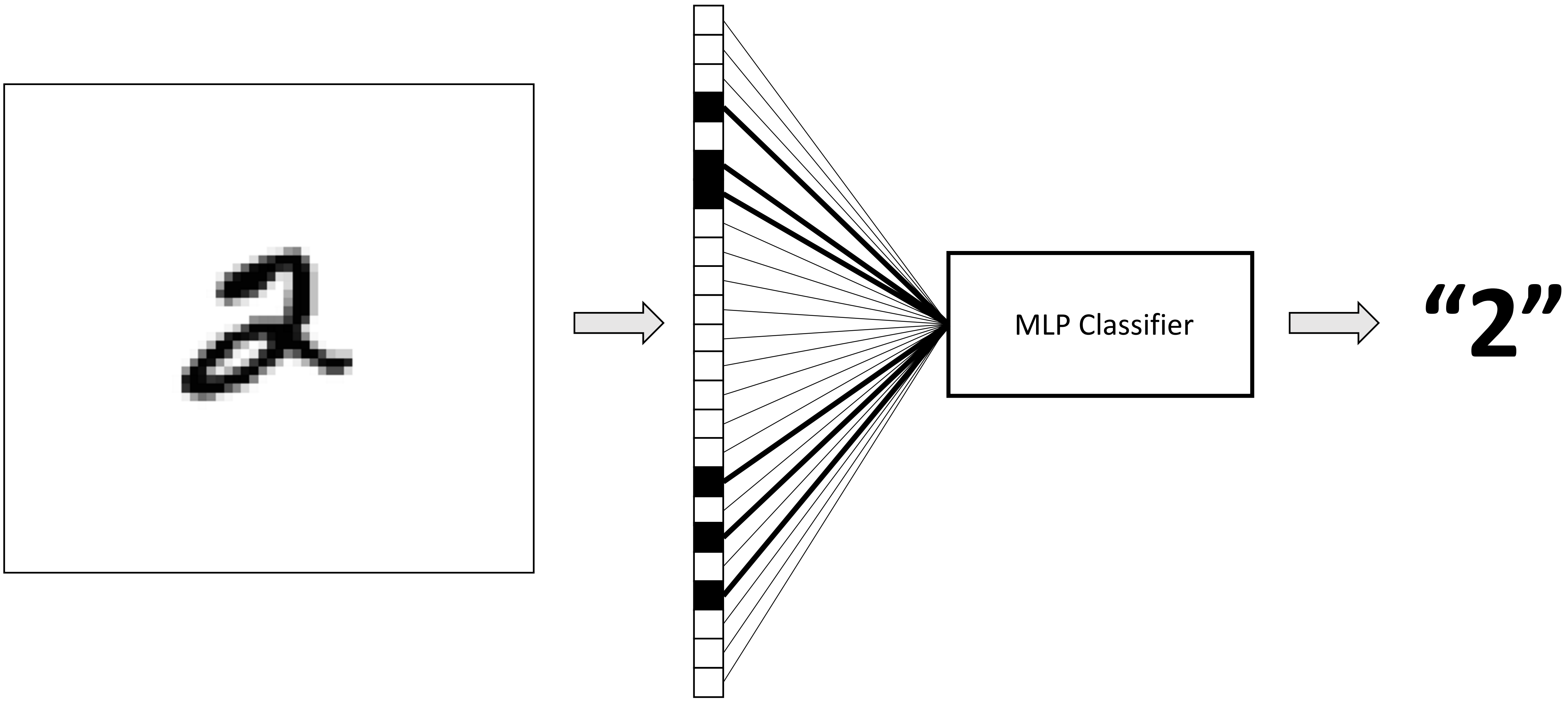
Deep Learning for Image Analysis

Automatic 3D Surface Meshes from Video

DensePose (Facebook)

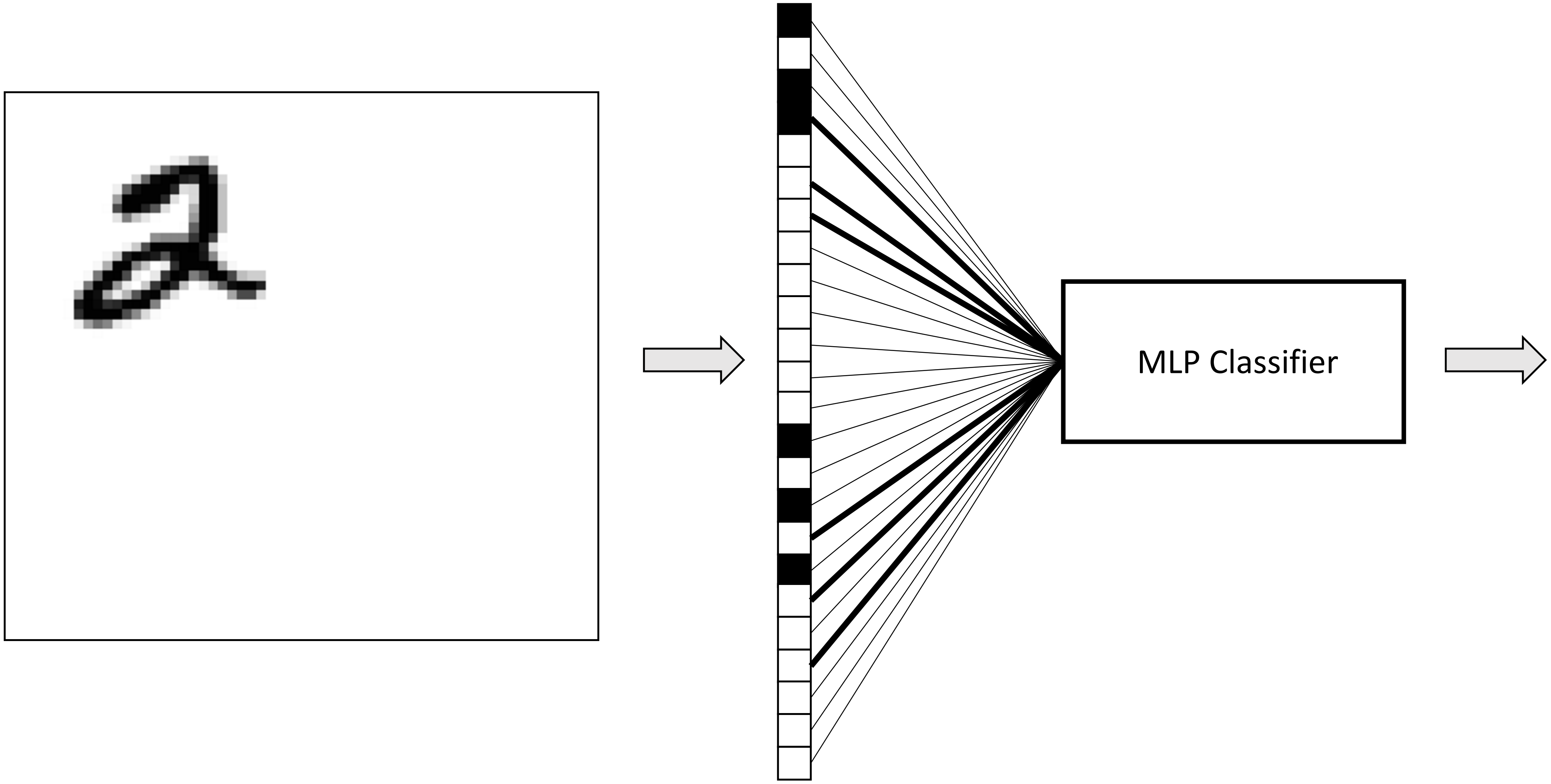
Convolutional Neural Networks Find Structure Anywhere In Images

Consider the multi-layer perceptron for digit recognition:



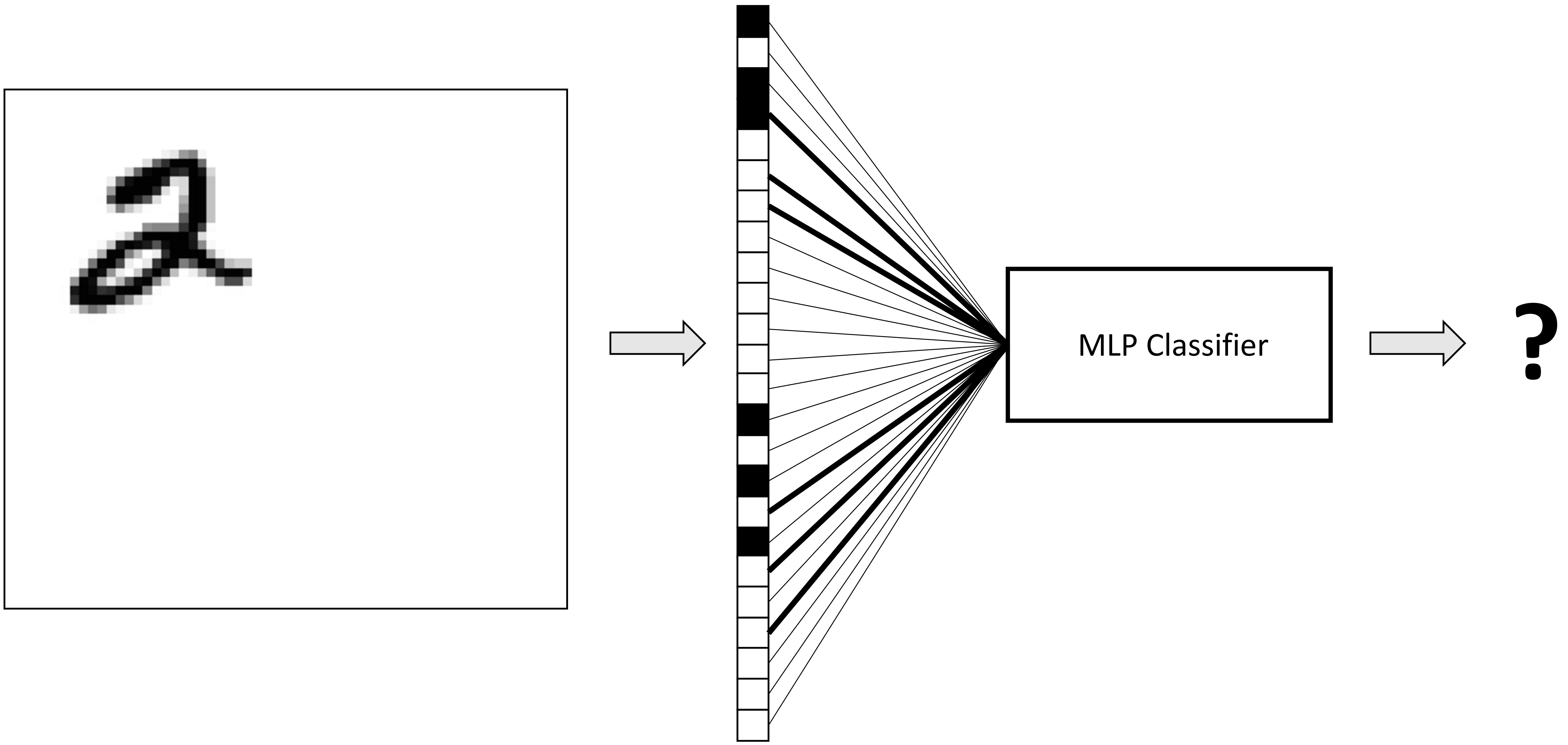
Convolutional Neural Networks Find Structure Anywhere In Images

Consider the multi-layer perceptron for digit recognition:

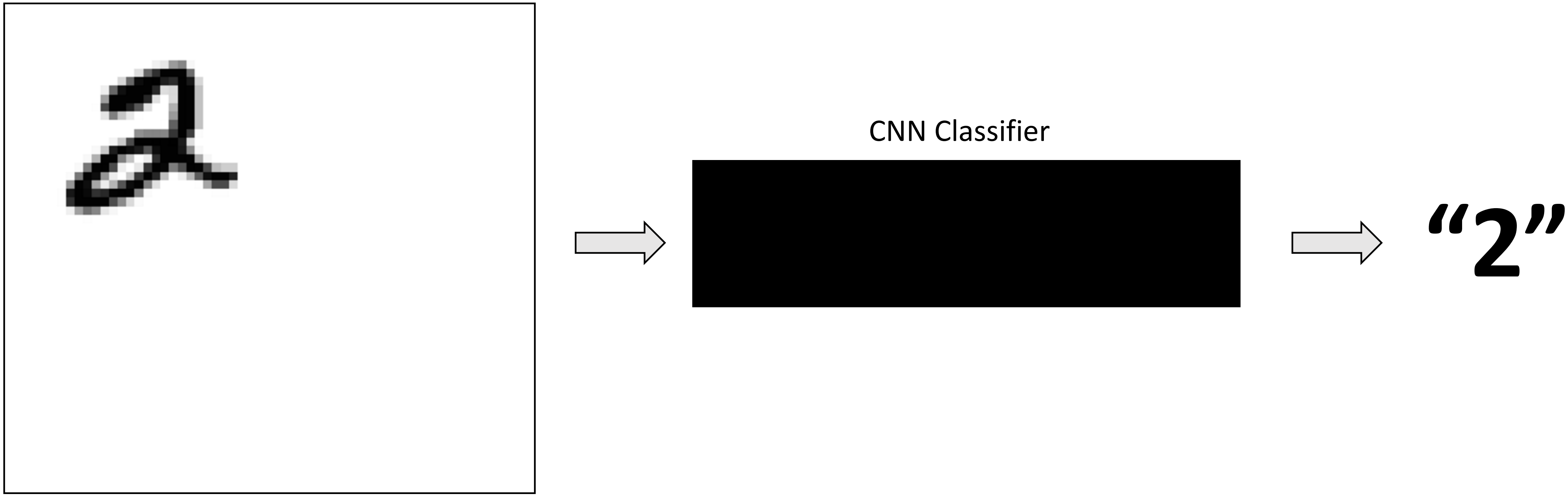


Convolutional Neural Networks Find Structure Anywhere In Images

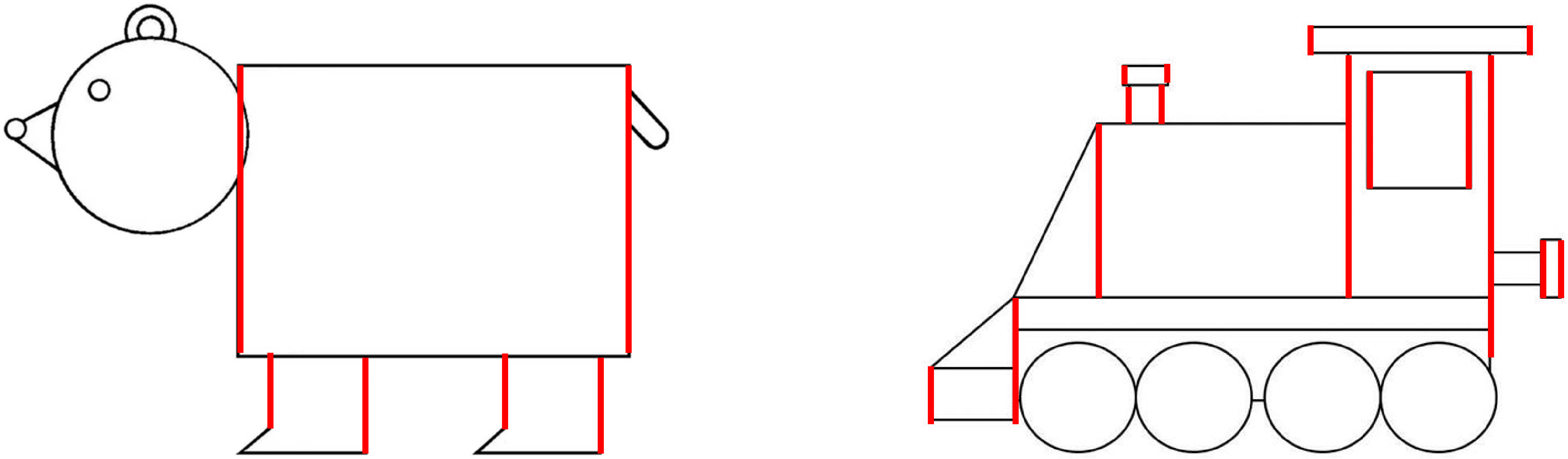
Consider the multi-layer perceptron for digit recognition:



Convolutional Neural Networks Find Structure Anywhere In Images

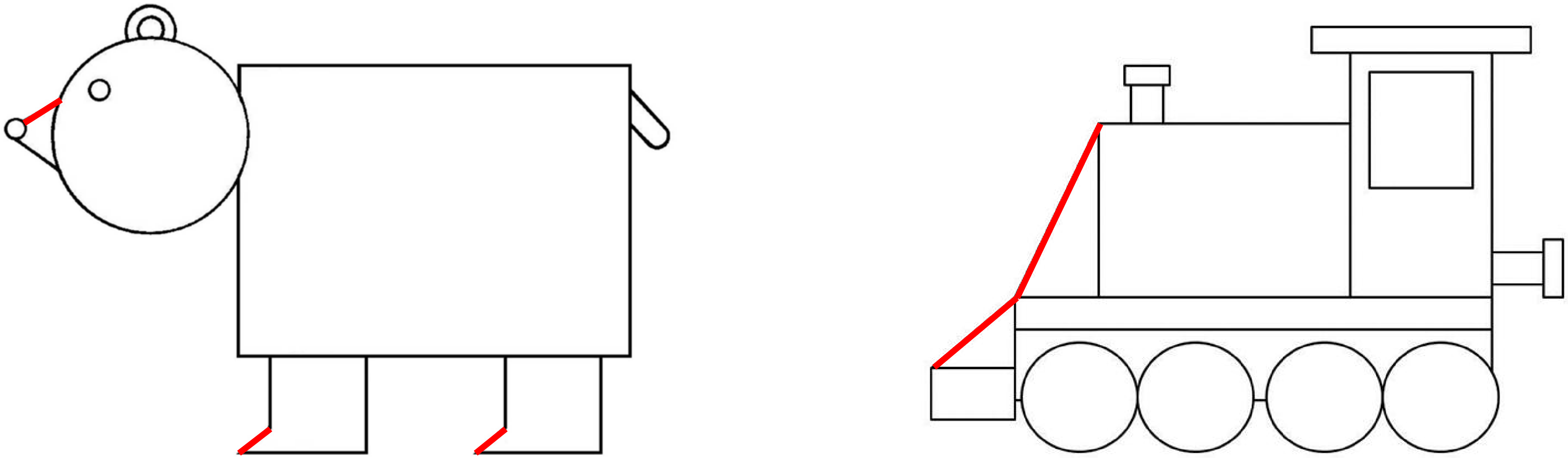


CNNs Take Advantage of Repeated, Hierarchical Structure in Images



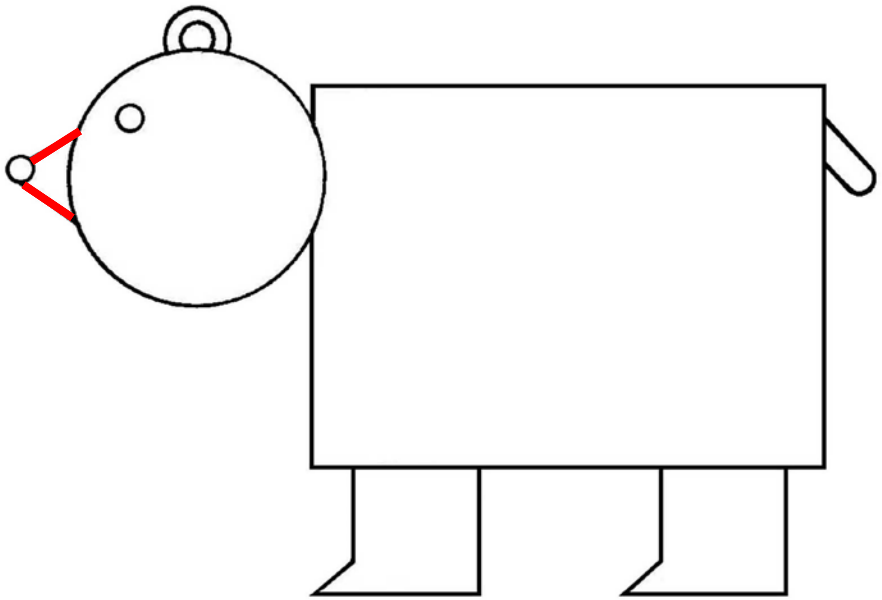
Low-level structure: lines, curves

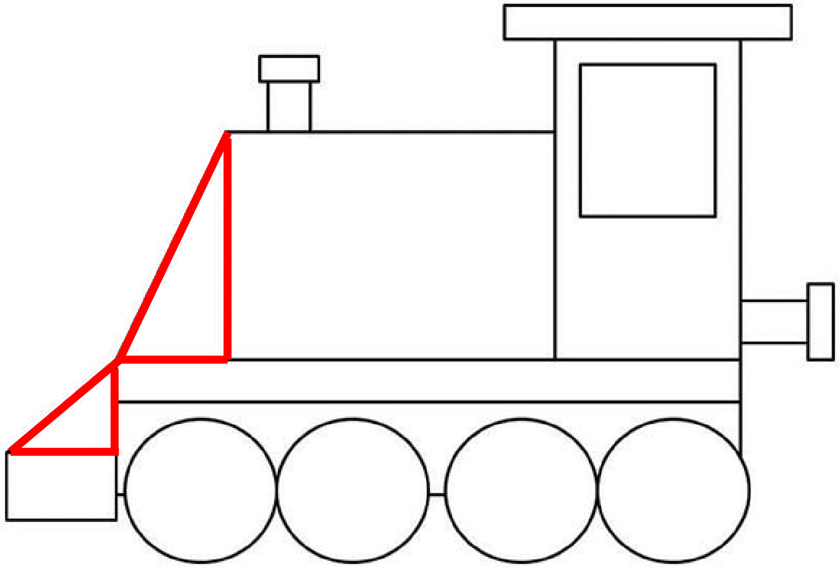
CNNs Take Advantage of Repeated, Hierarchical Structure in Images



Low-level structure: lines, curves

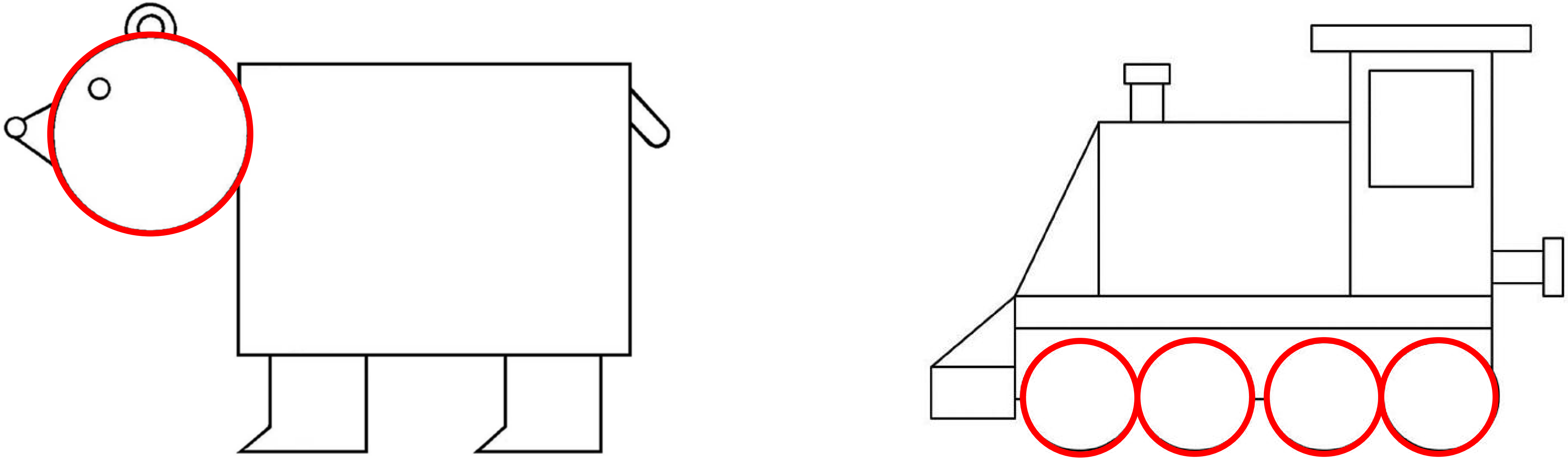
CNNs Take Advantage of Repeated, Hierarchical Structure in Images





Mid-level structure: shapes

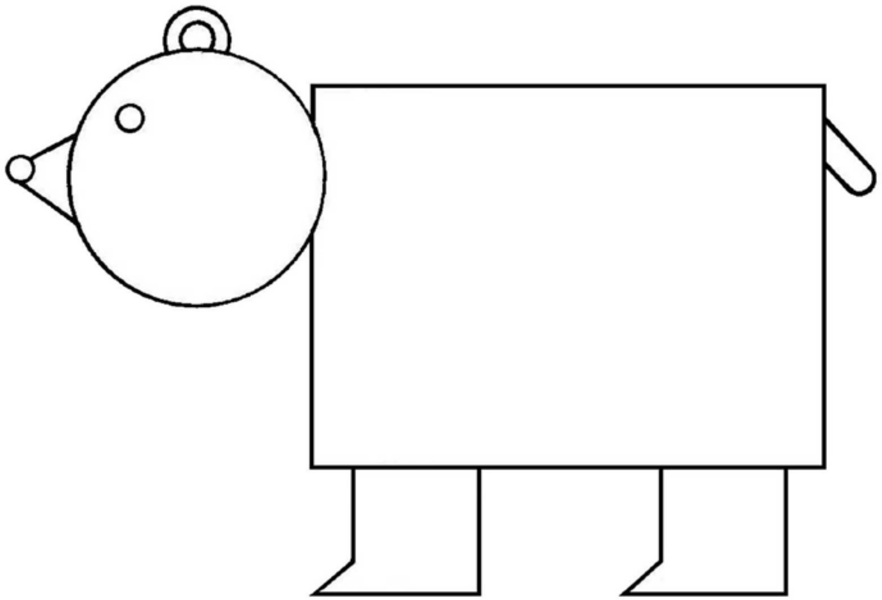
CNNs Take Advantage of Repeated, Hierarchical Structure in Images



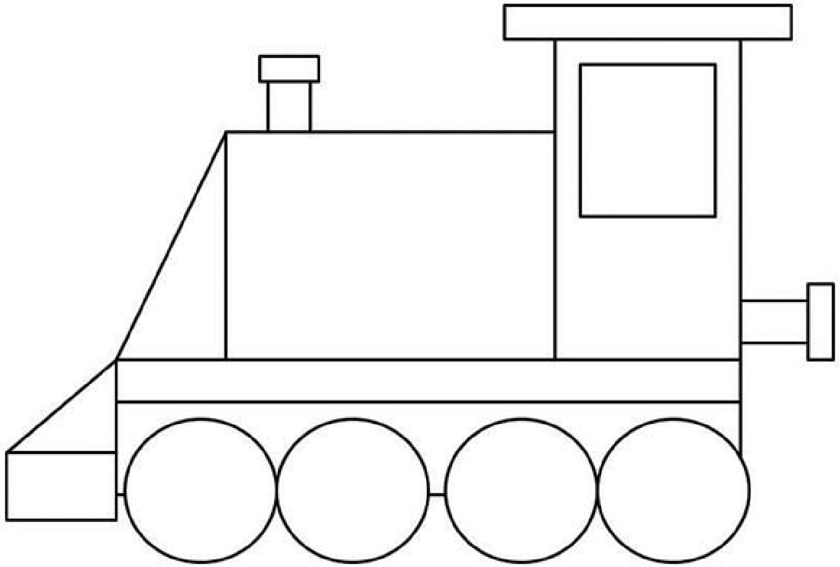
Mid-level structure: shapes

CNNs Take Advantage of **Repeated, Hierarchical Structure** in Images

Bear

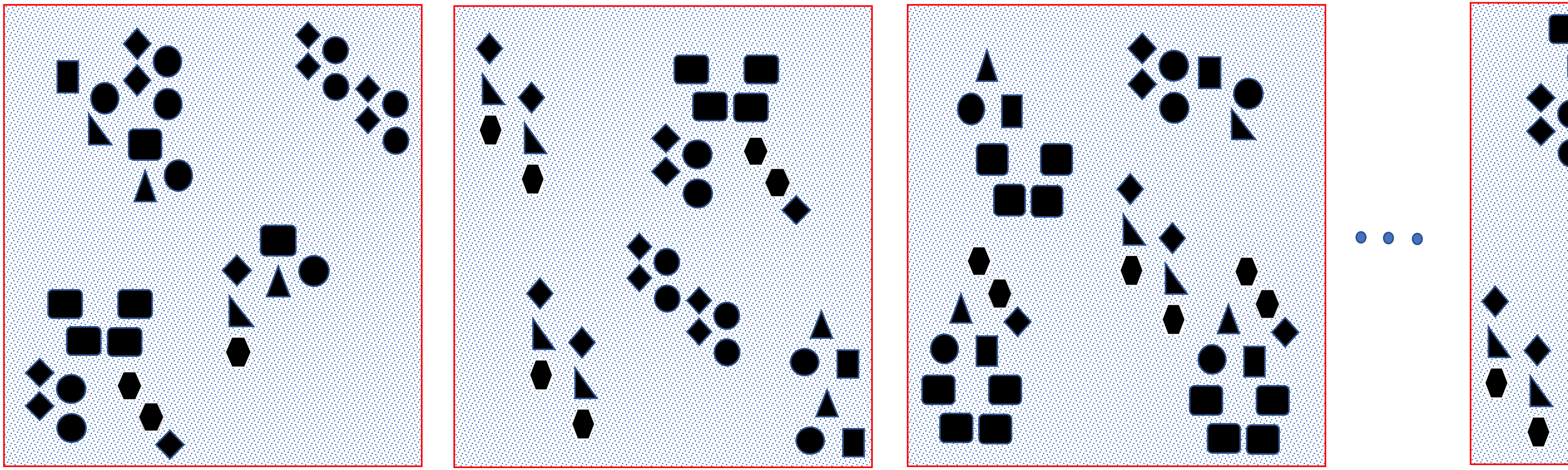


Train

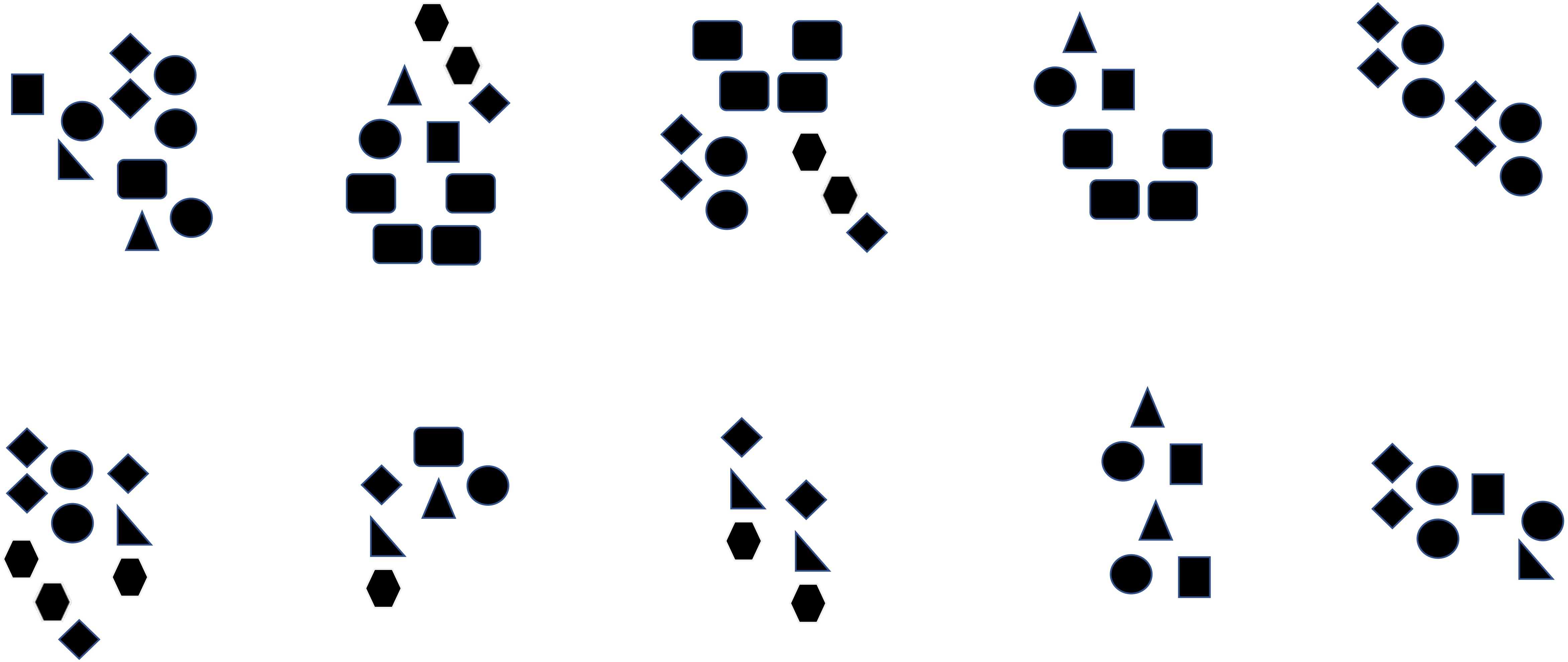


High-level structure: groups of shapes $\rightarrow$ objects

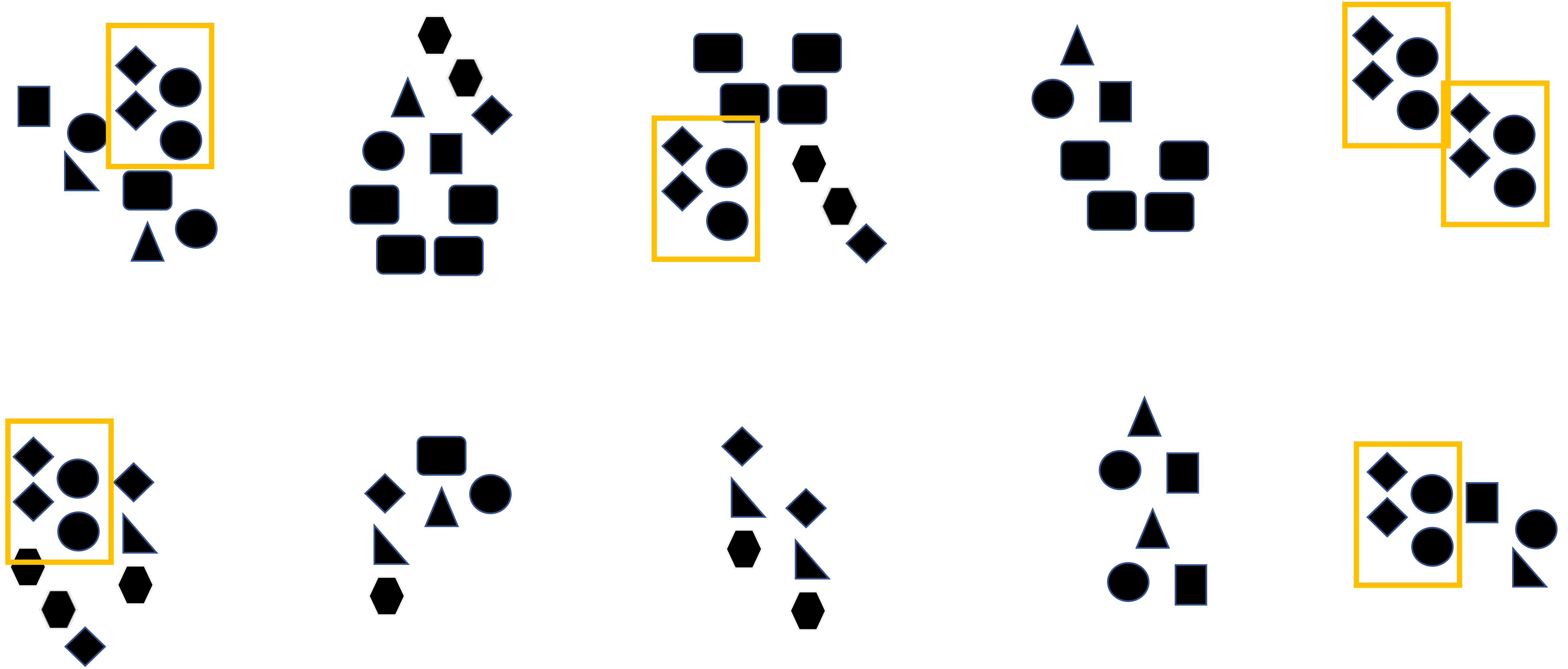
Consider a Set of “Toy” Images, for Illustration



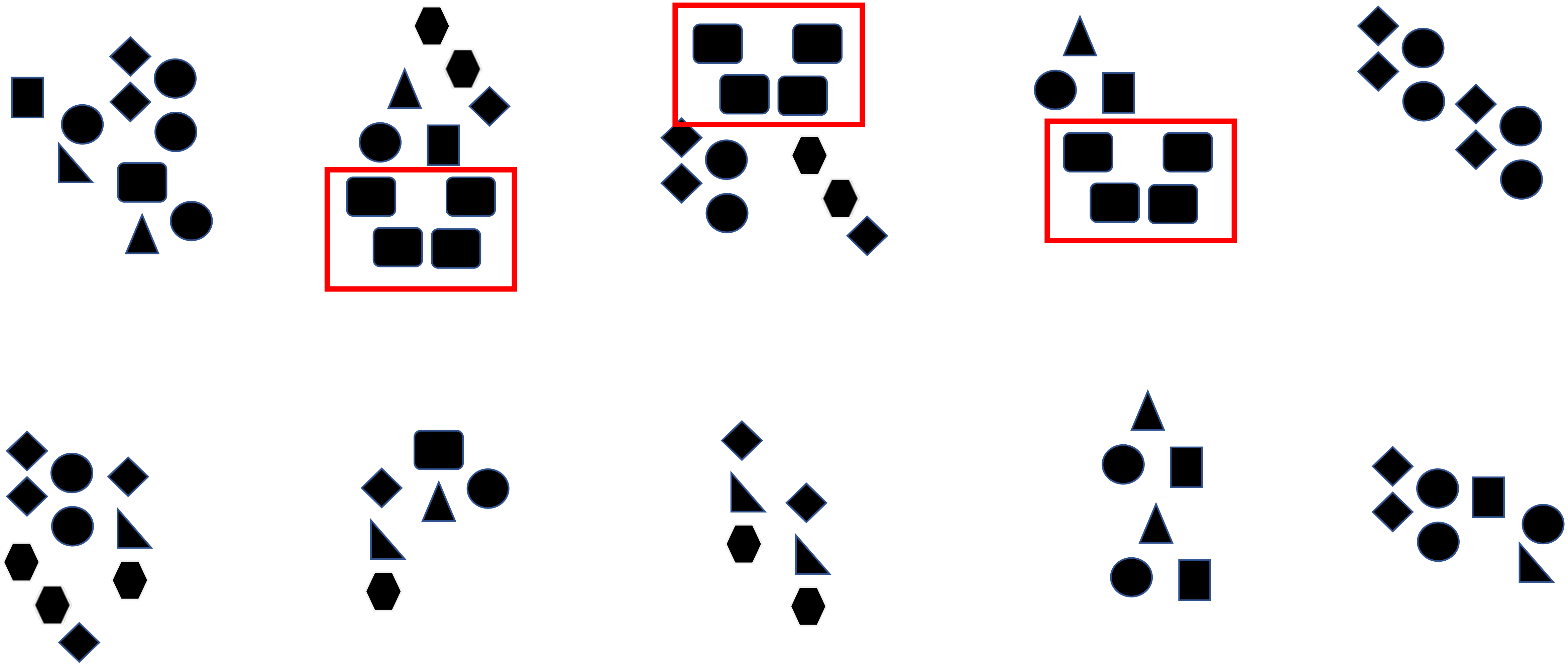
High-Level Motifs



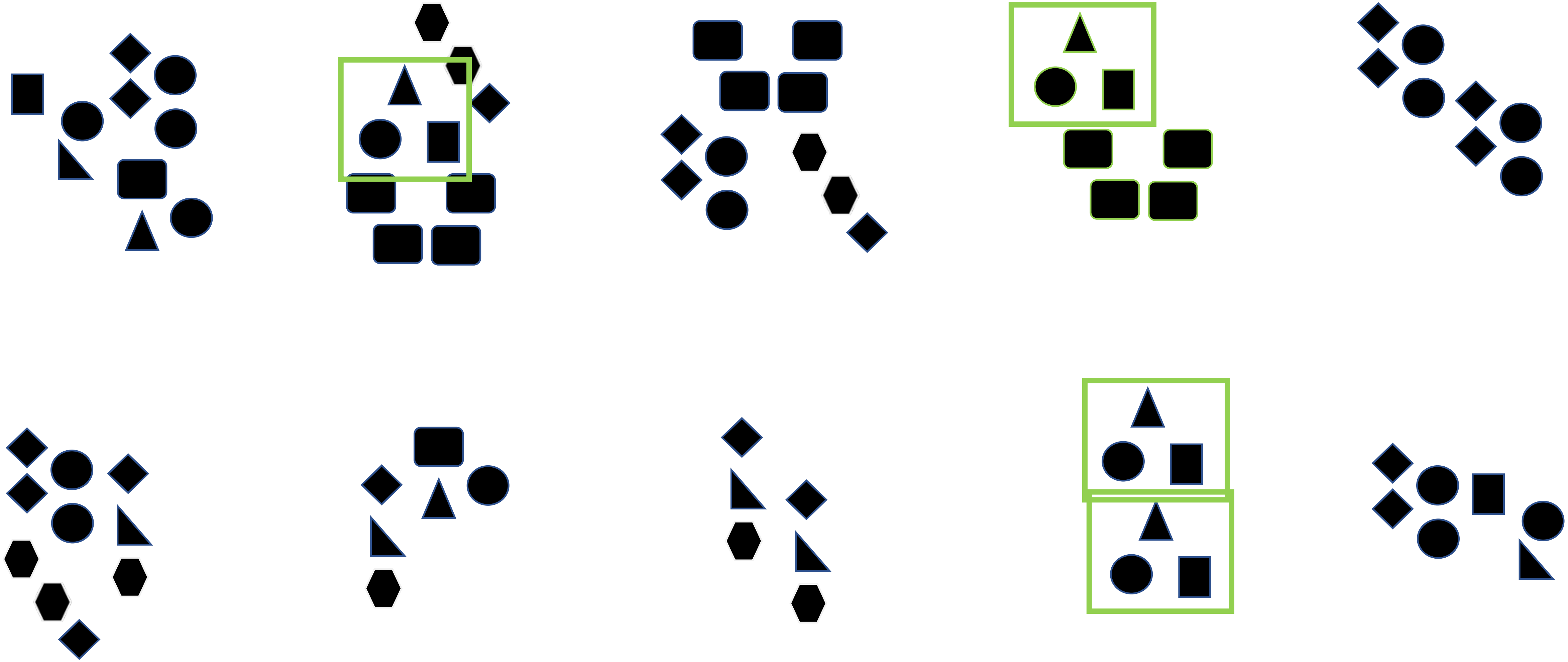
Shared Substructure Within Motifs



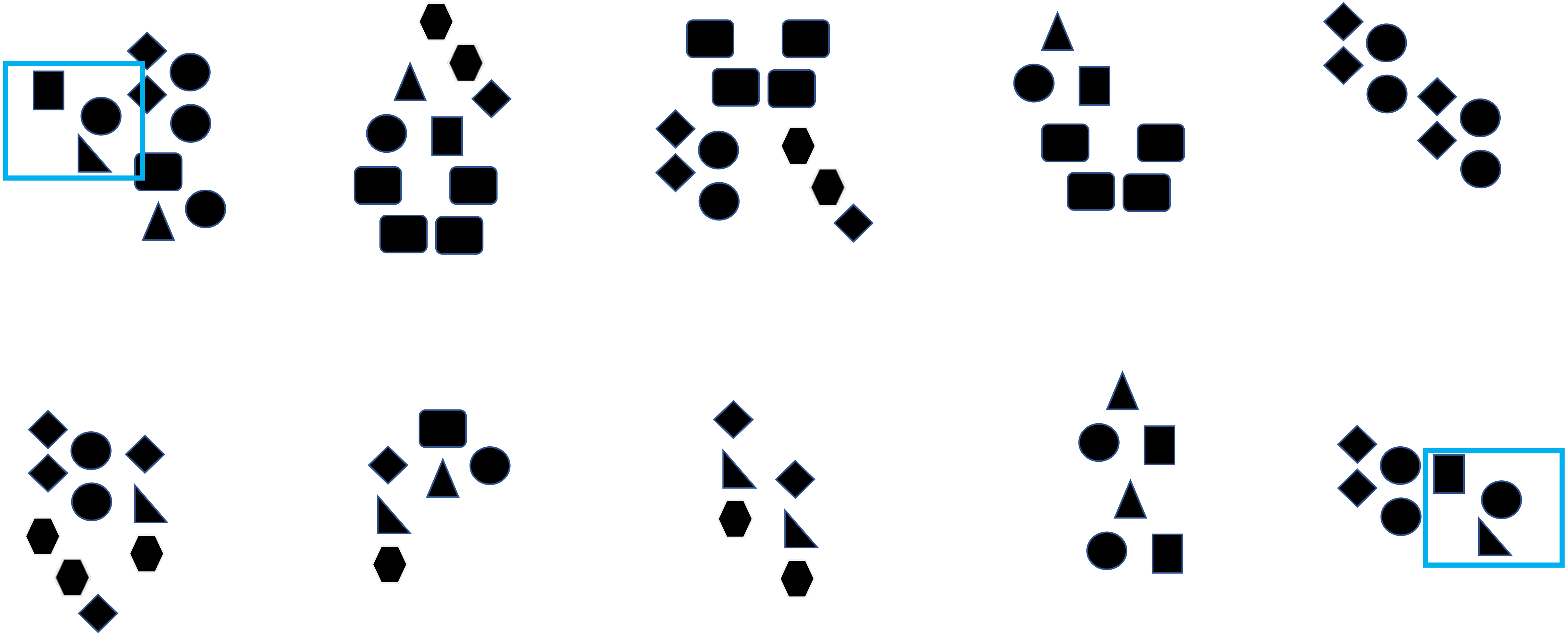
Shared Substructure Within Motifs



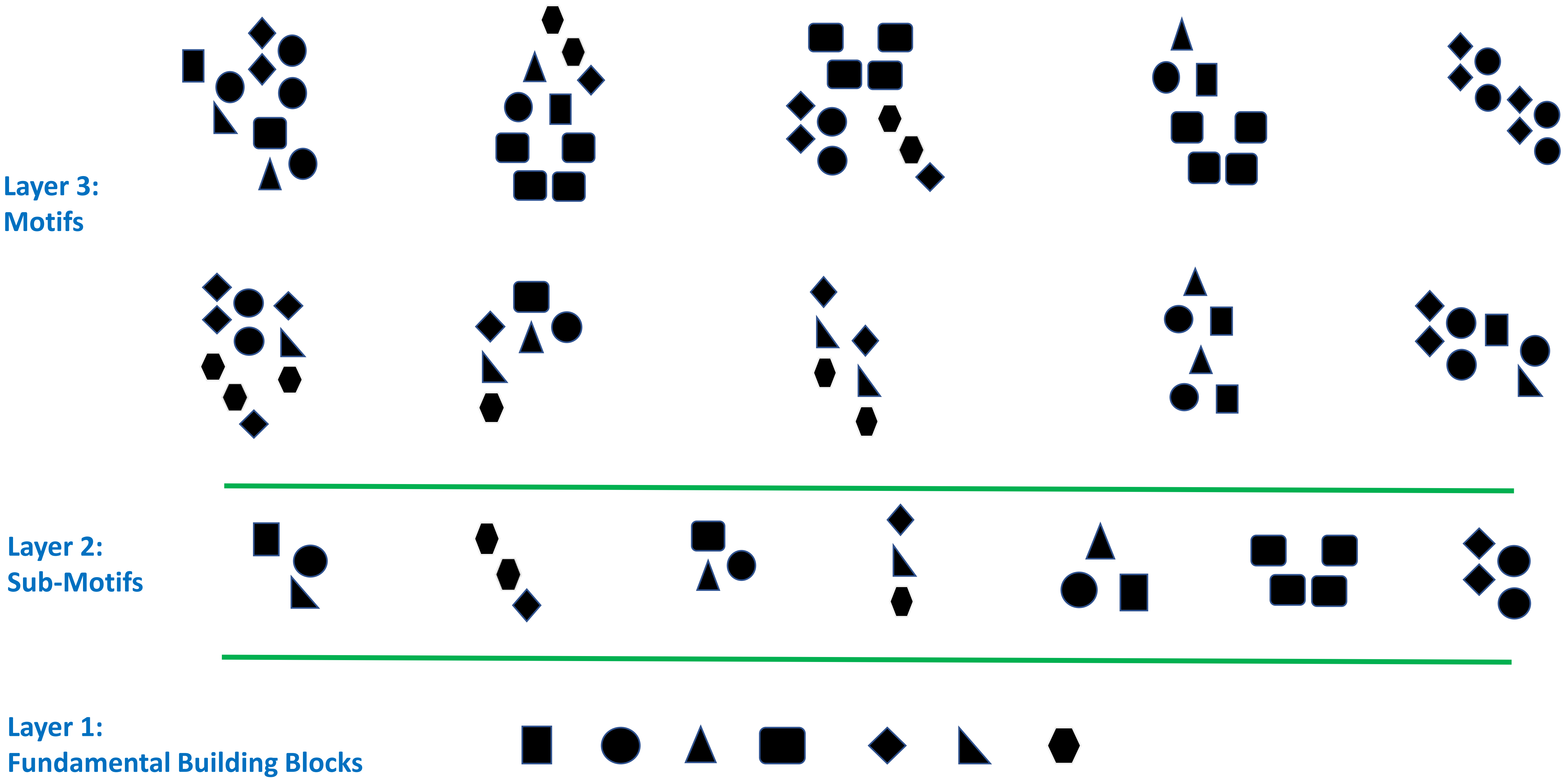
Shared Substructure Within Motifs



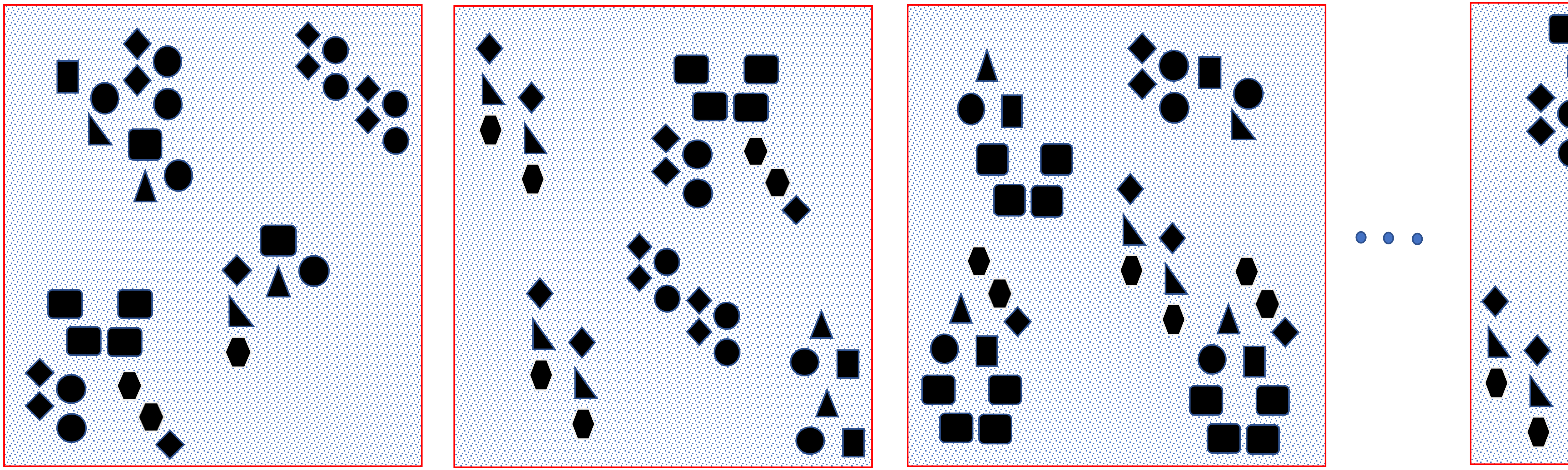
Shared Substructure Within Motifs



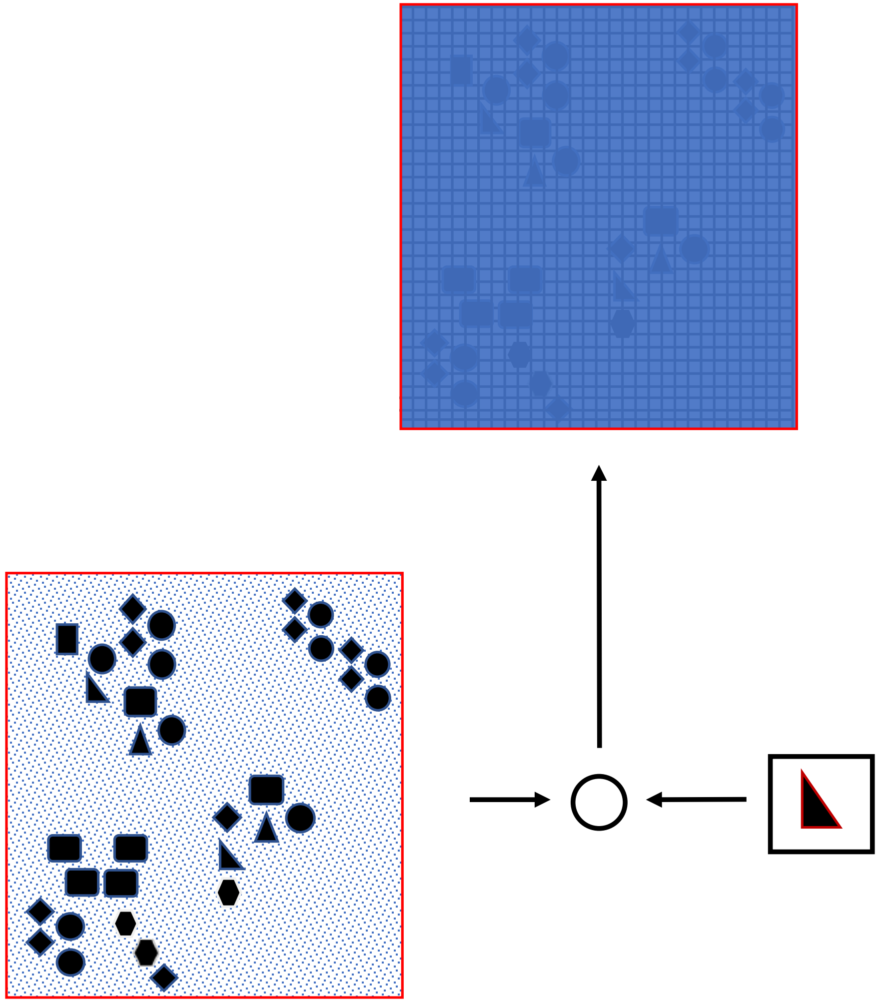
Hierarchical Representation of Images



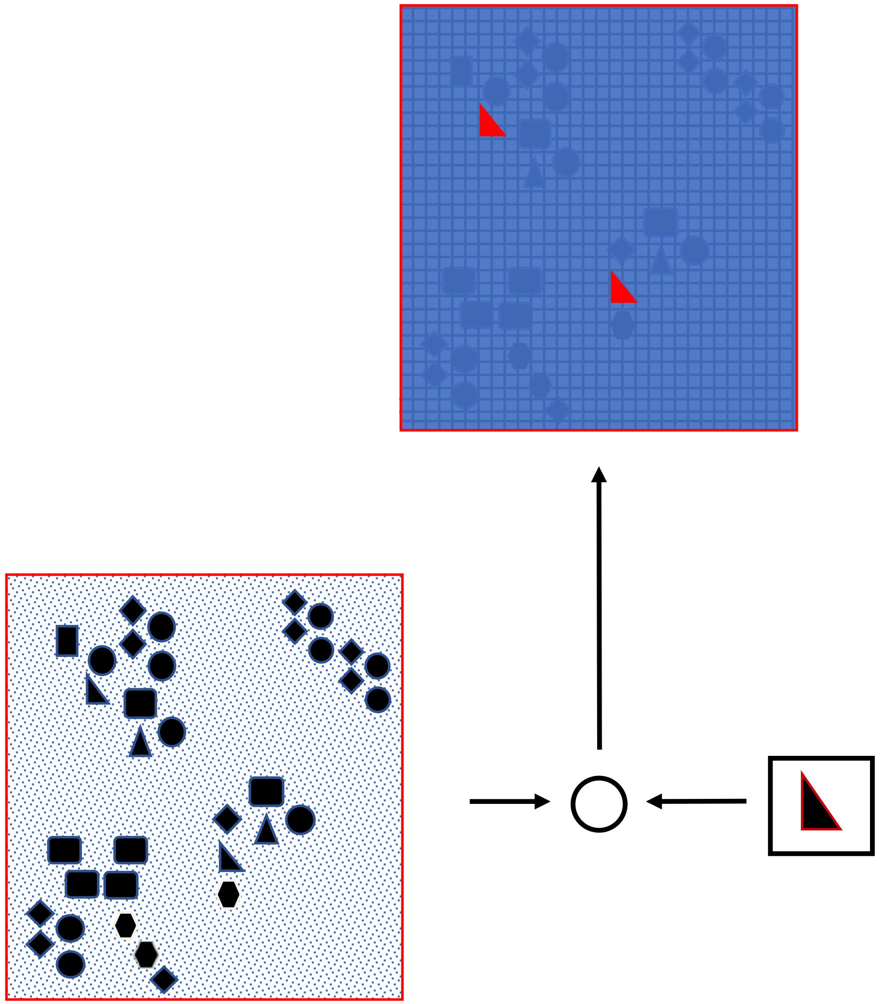
Recall the Data/Images



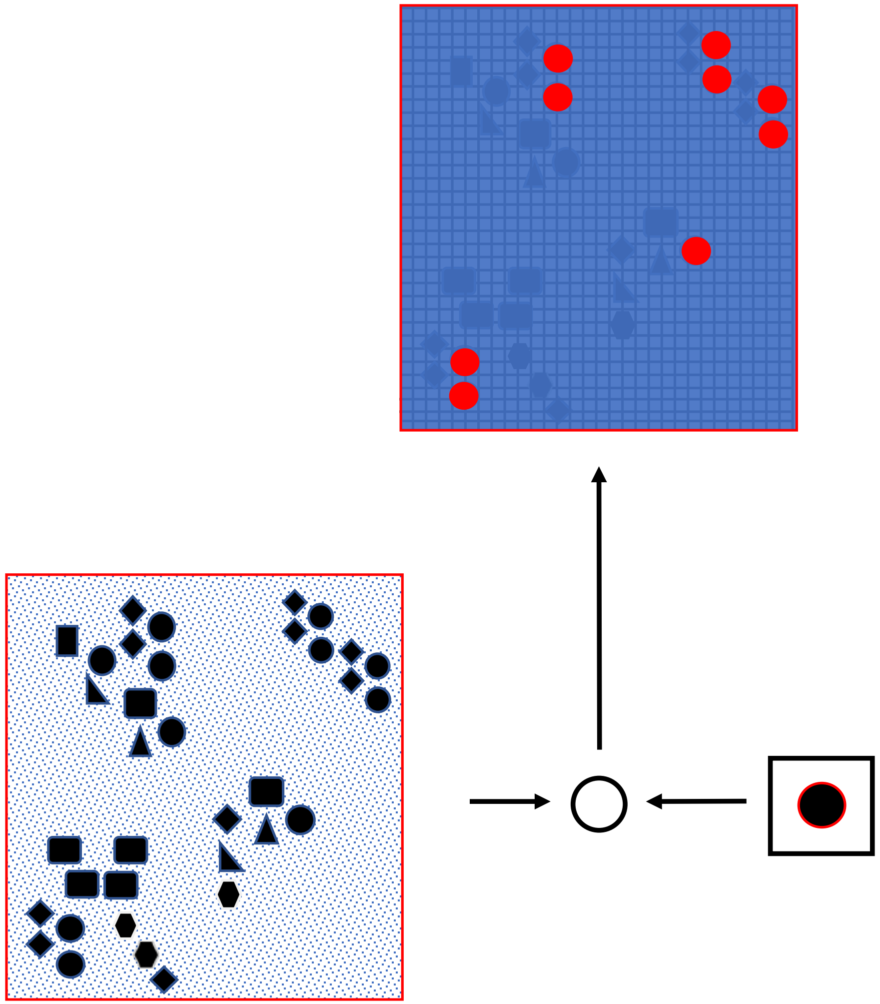
Convolutional Filter



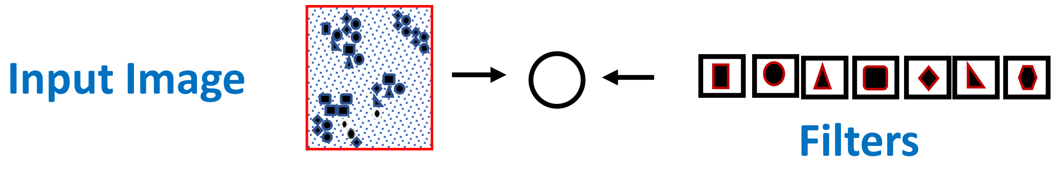
Convolutional Filter

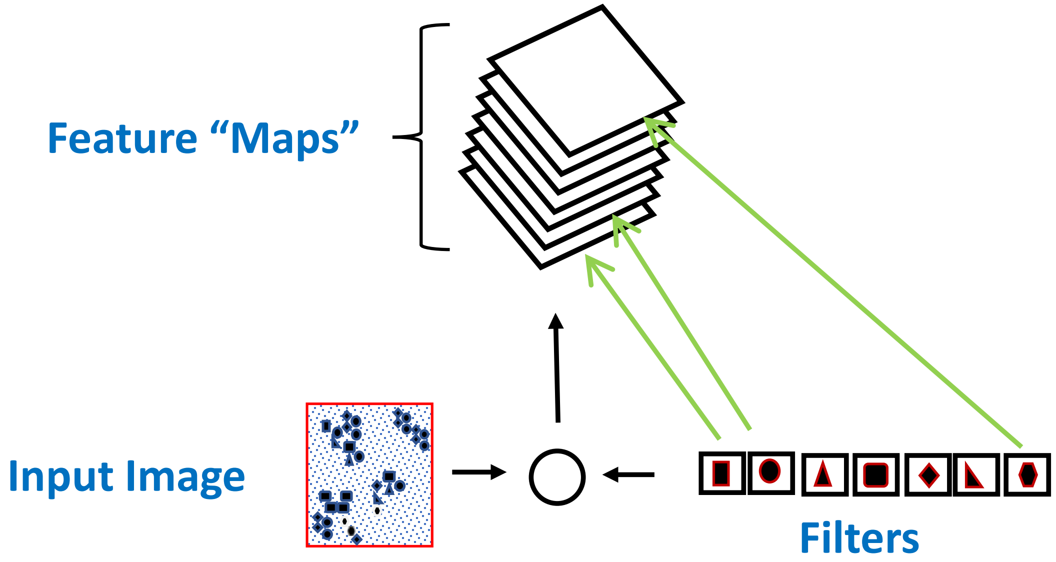


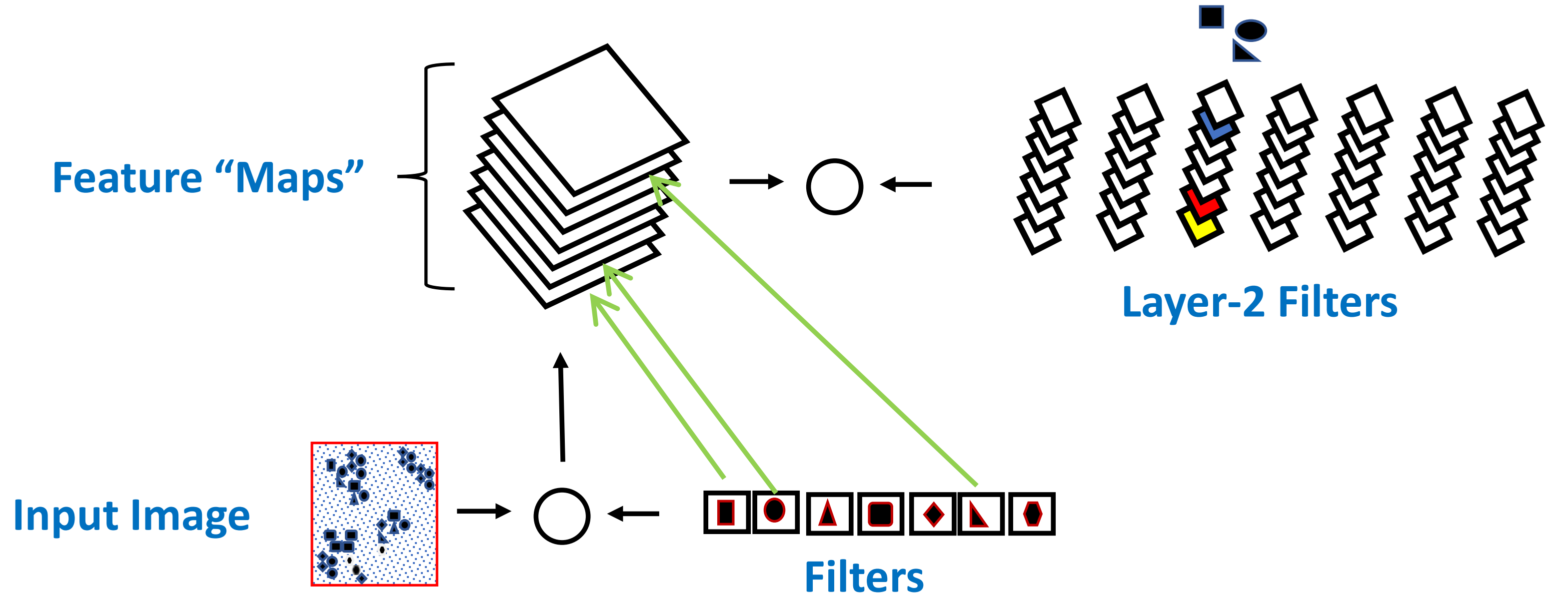
Convolutional Filter

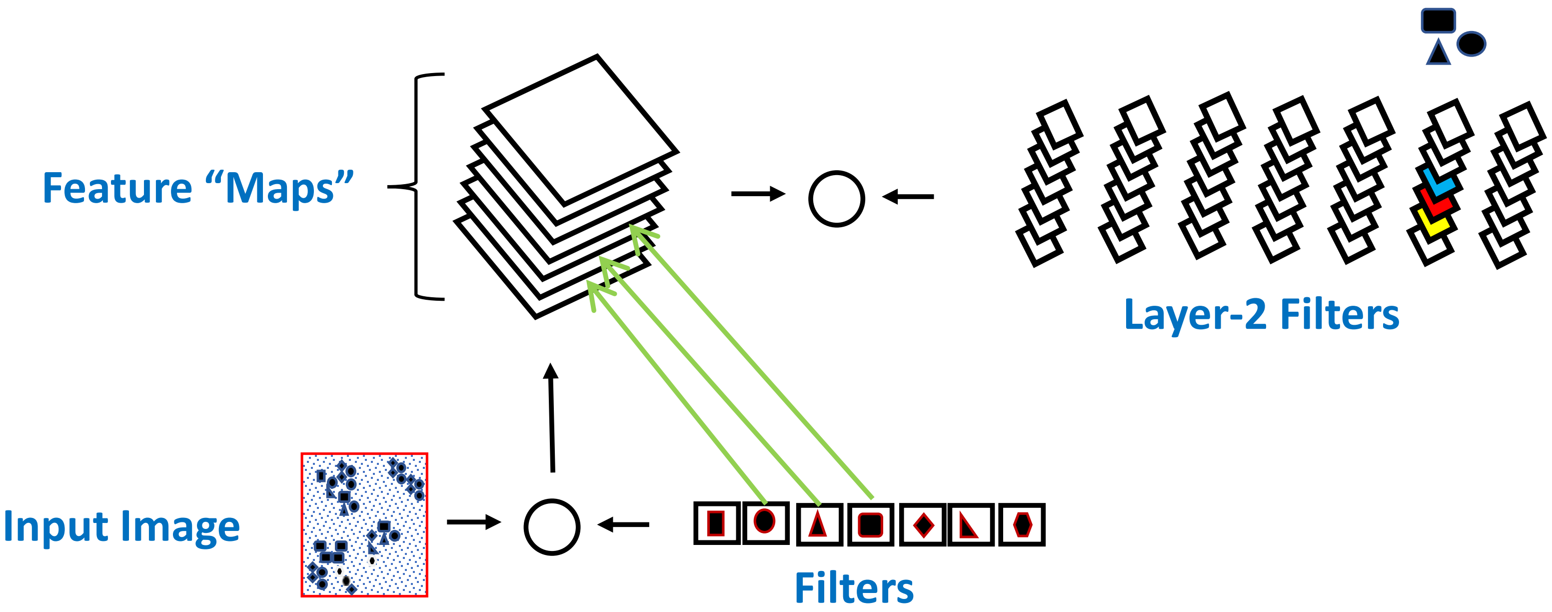


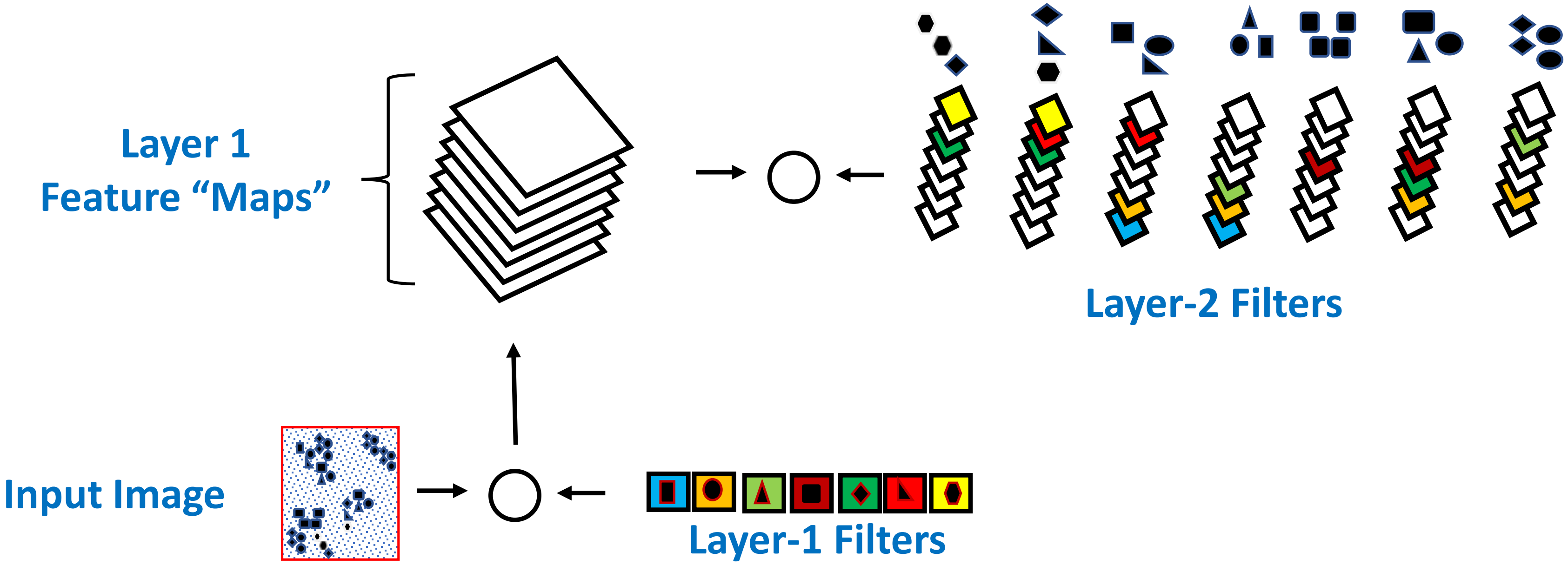
Multiple Filters, One for Each Building Block

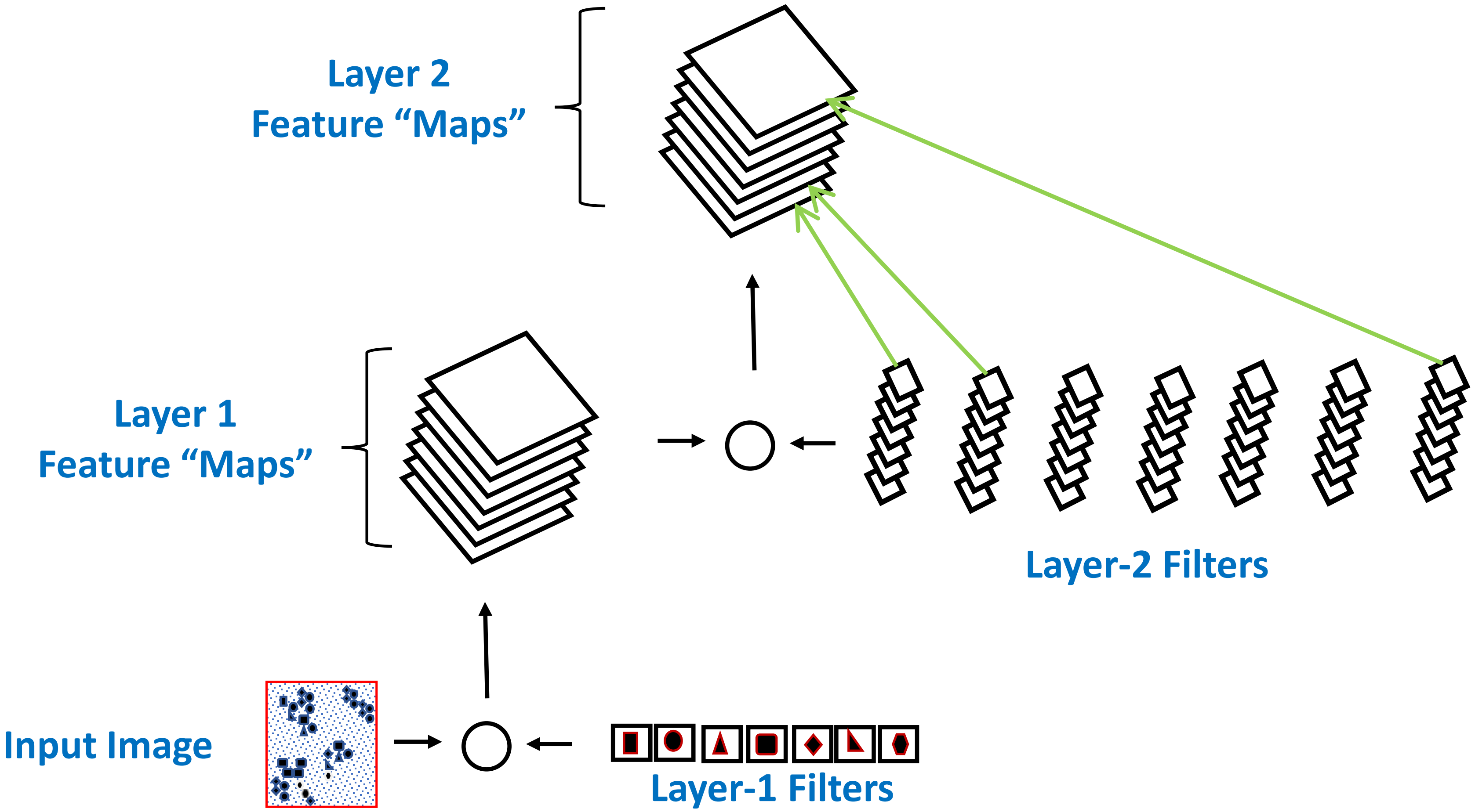


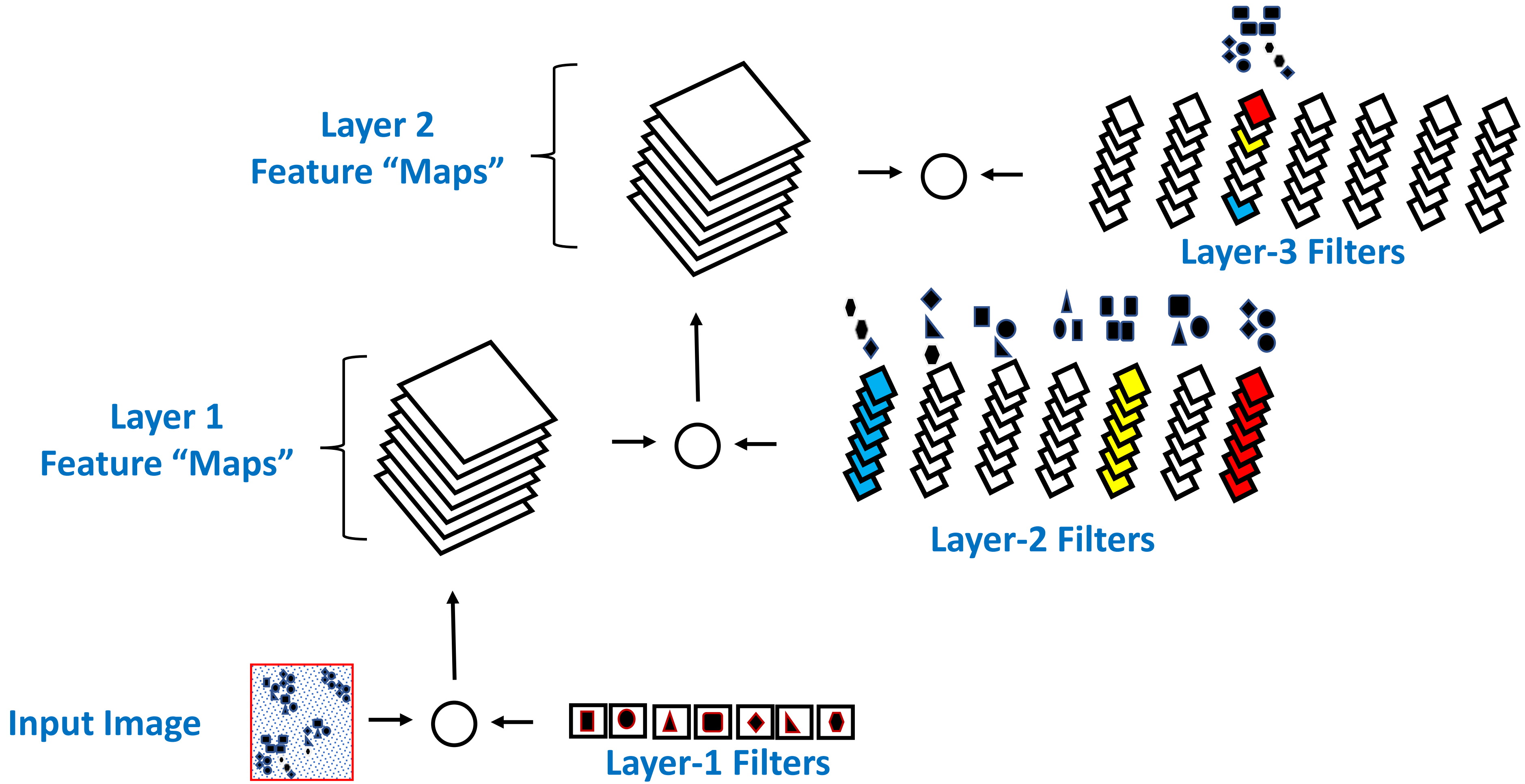

Feature “Maps”
Input Image
Filters

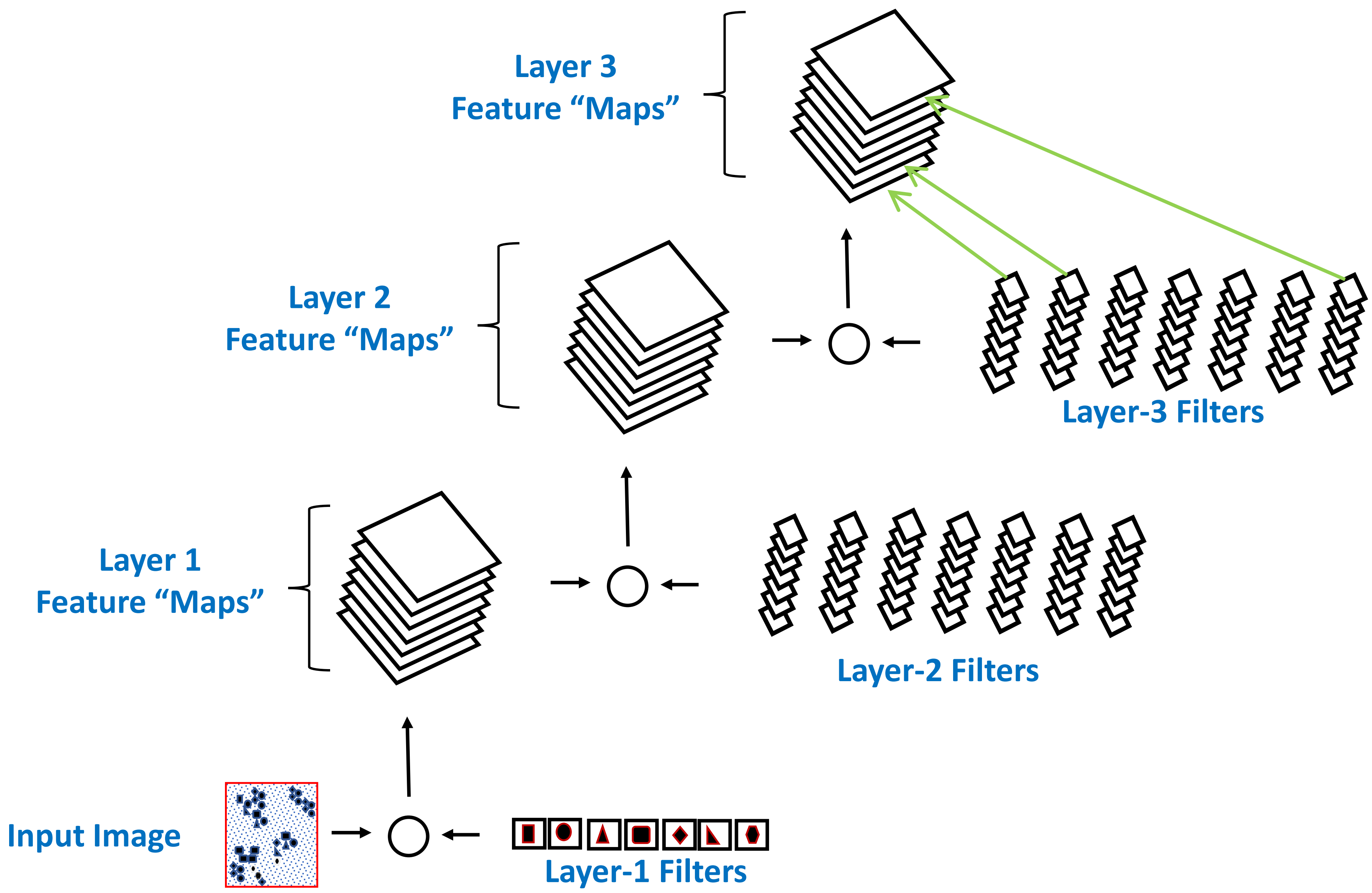

Feature "Maps"
Layer-2 Filters
Input Image
Filters


Feature "Maps"
Layer-2 Filters
Input Image
Filters

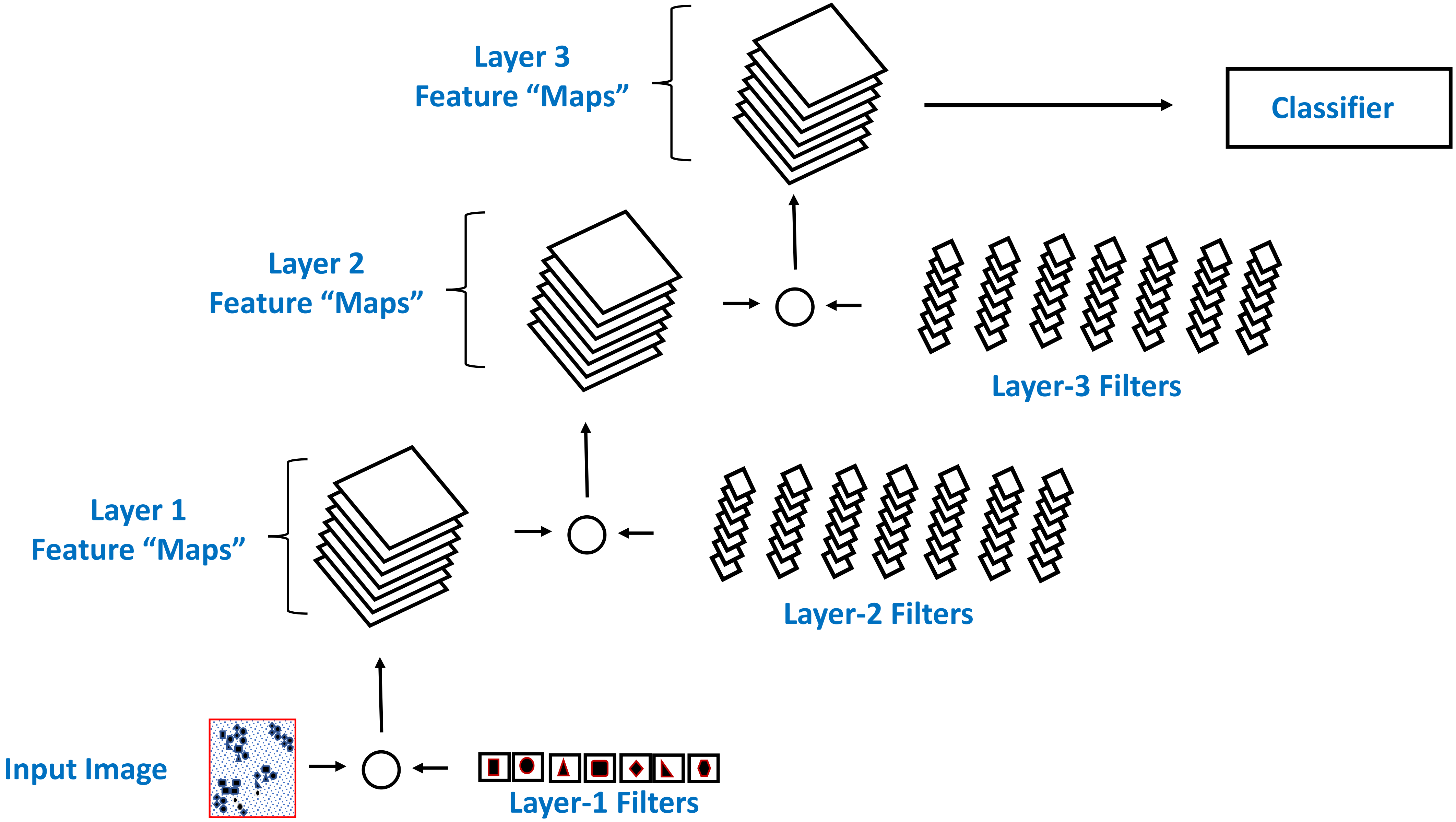

Layer 1
Feature "Maps"
Layer-2 Filters
Input Image
Layer-1 Filters


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

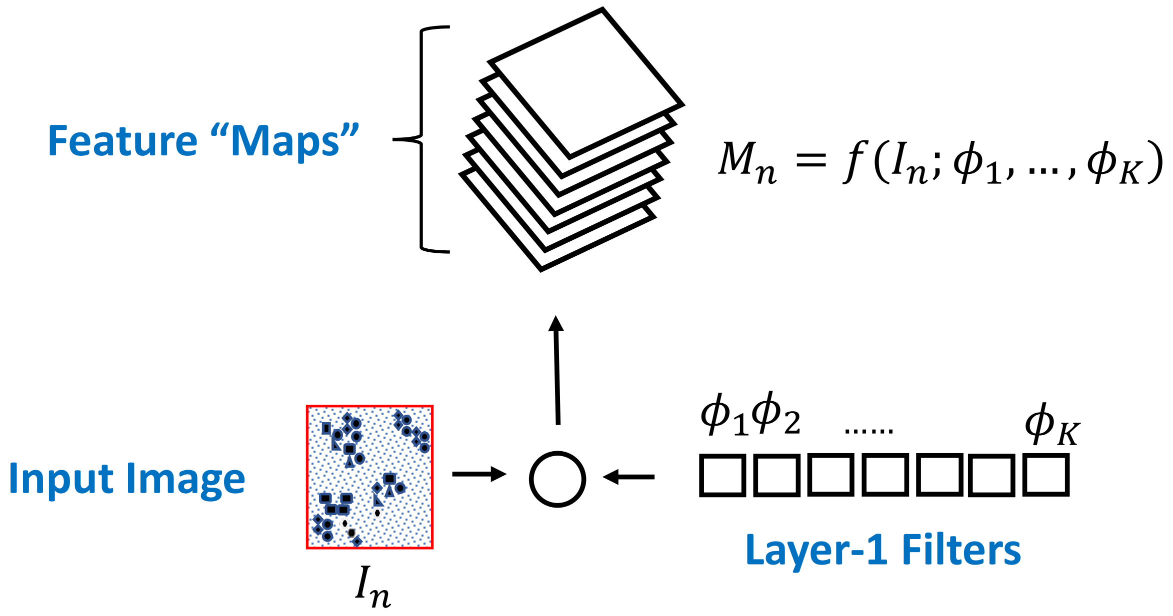

Layer 3
Feature "Maps"
Layer 2
Feature "Maps"
Layer-3 Filters
Layer 1
Feature "Maps"
Layer-2 Filters
Input Image
Layer-1 Filters

Deep Analysis Architecture

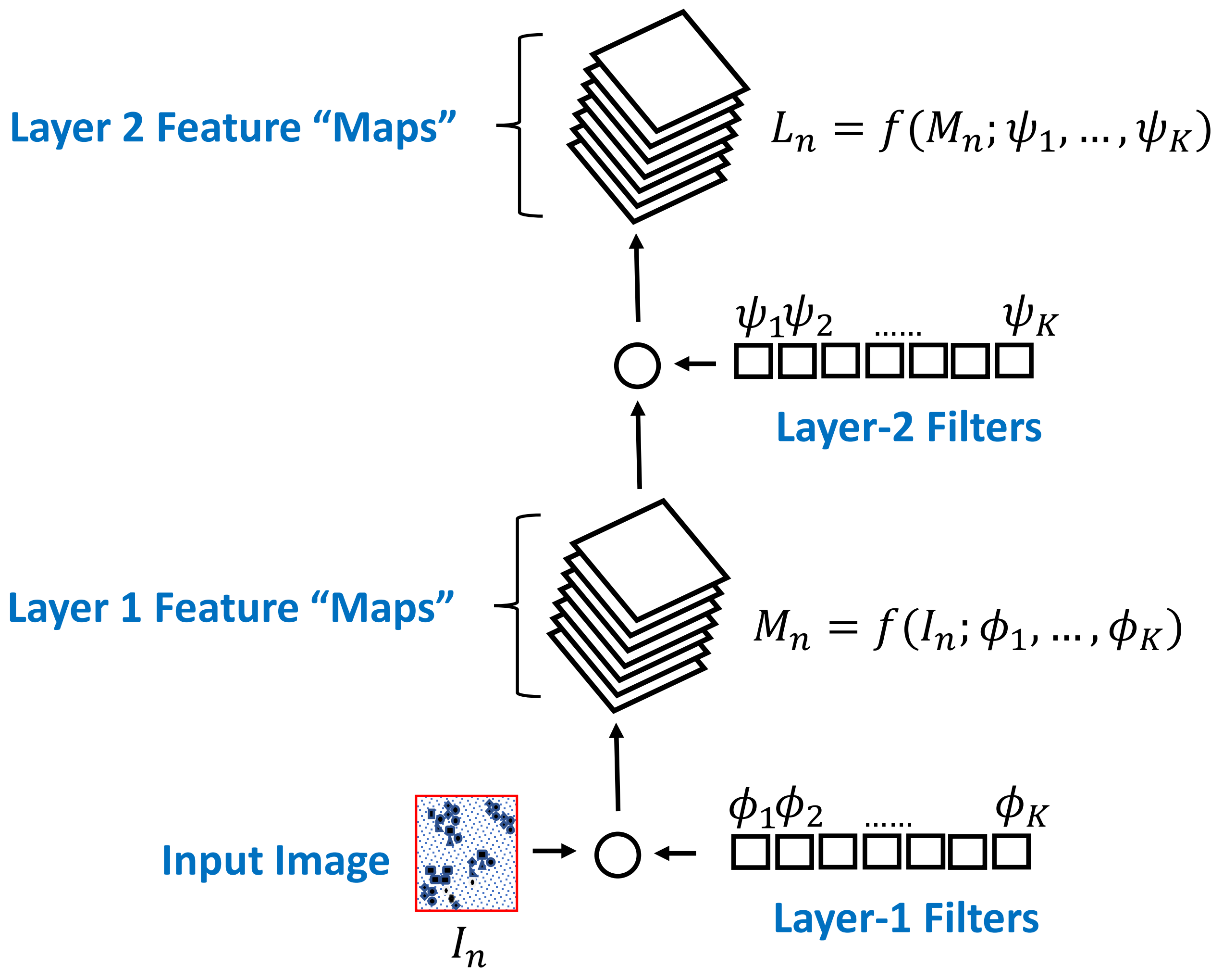


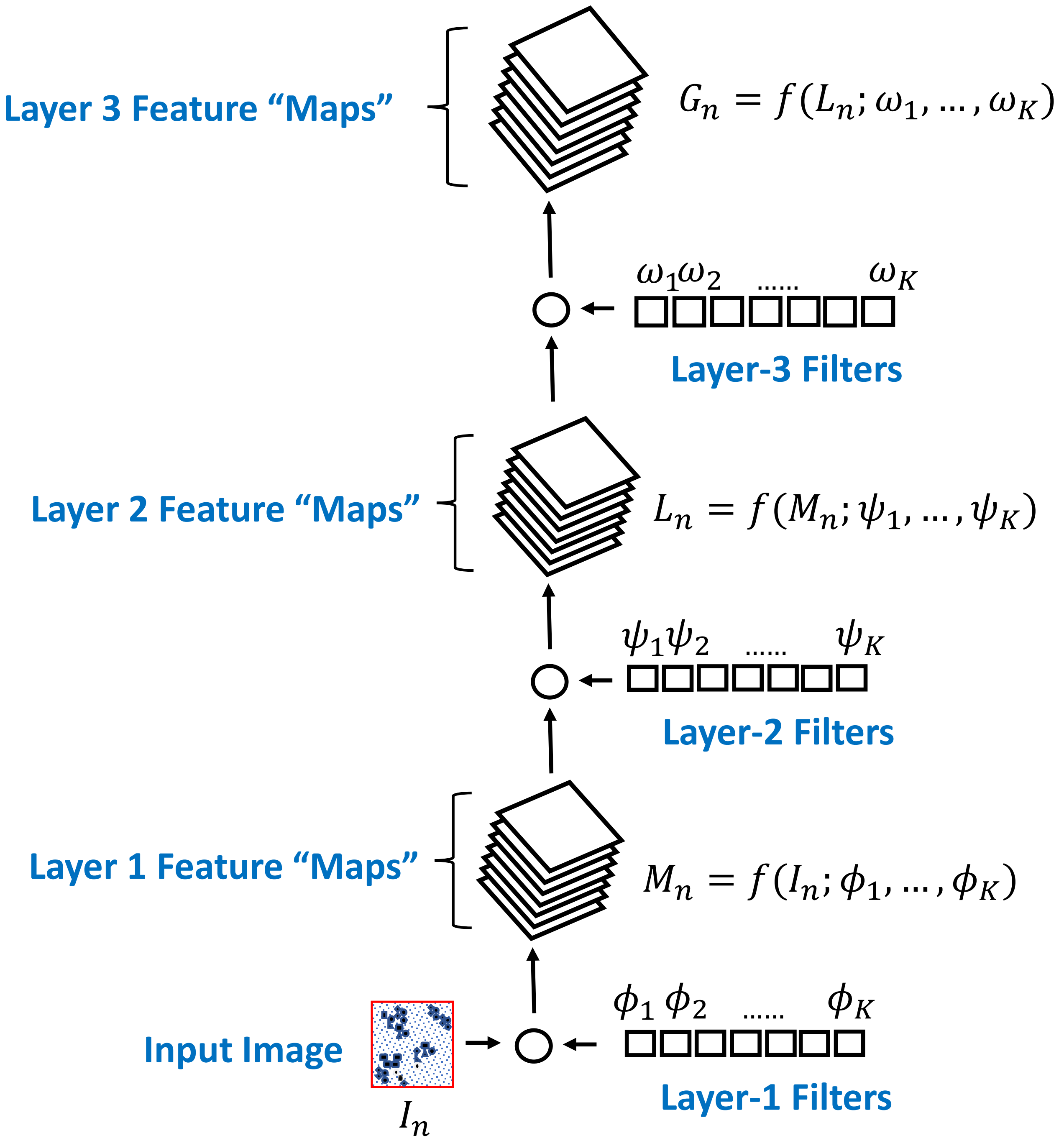
Given Images, How Do We Learn Model Parameters?

- The previous discussion was an illustration for motivating the "deep" algorithm concept
- Demonstrated using "toy" images
- How do we build such an algorithm in practice, given a large set of training images?

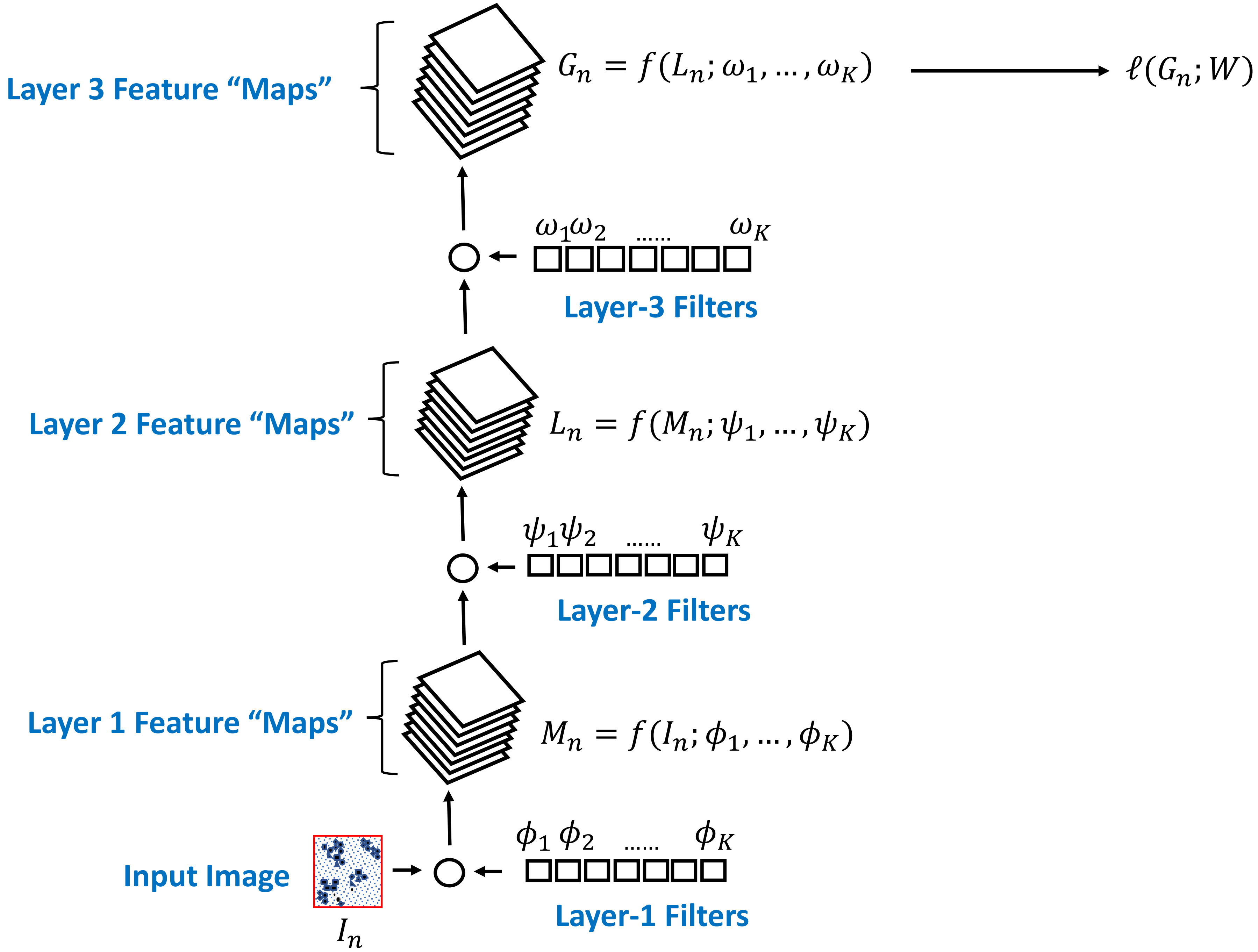


Feature “Maps”
$M_n = f(I_n; \phi_1, \dots, \phi_K)$
Input Image
I_n
$\phi_1 \phi_2$ …… ϕ_K
Layer-1 Filters

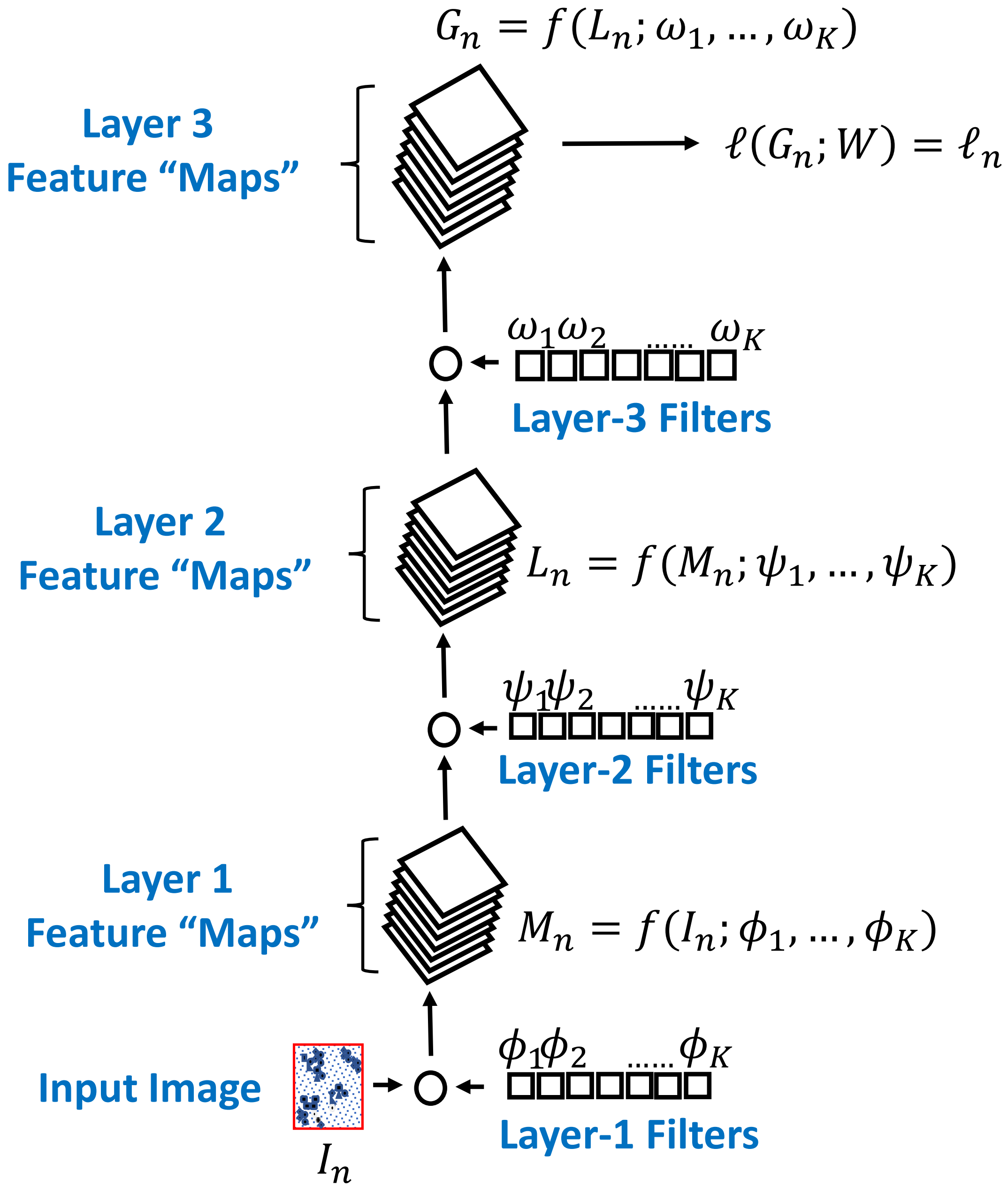

Layer 2 Feature “Maps”
$L_n = f(M_n; \psi_1, \dots, \psi_K)$
$\psi_1 \psi_2$ ...... ψ_K
Layer-2 Filters
Layer 1 Feature “Maps”
$M_n = f(I_n; \phi_1, \dots, \phi_K)$
$\phi_1 \phi_2$ ...... ϕ_K
Input Image
I_n
Layer-1 Filters



Layer 3 Feature "Maps"
$G_n = f(L_n; \omega_1, \ldots, \omega_K)$
$\omega_1 \omega_2$ ...... 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



Layer 3 Feature "Maps"
$G_n = f(L_n; \omega_1, \dots, \omega_K)$
$\ell(G_n; W)$
$\omega_1 \omega_2$ ...... ω_K
Layer-3 Filters
Layer 2 Feature "Maps"
$L_n = f(M_n; \psi_1, \dots, \psi_K)$
$\psi_1 \psi_2$ ...... 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

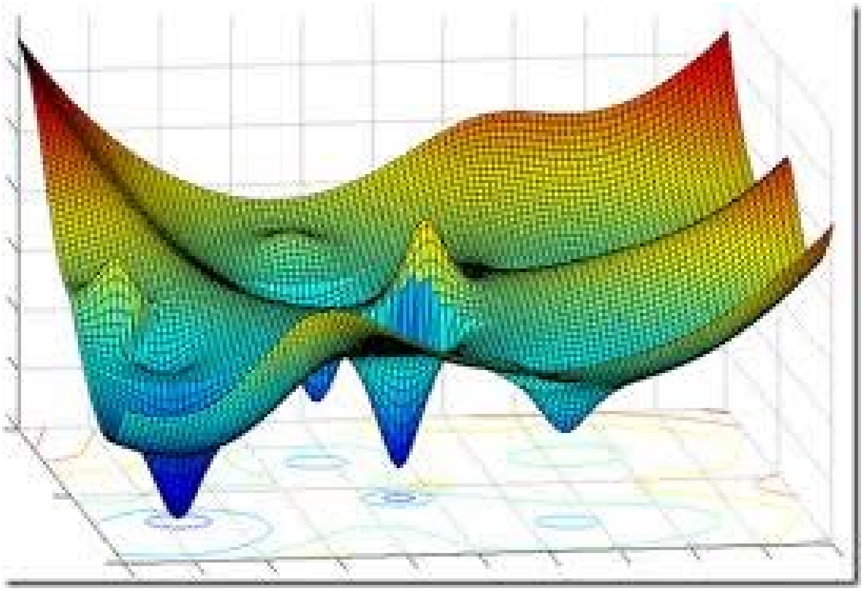


- Assume we have labeled images $\{I_n, y_n\}_{n=1,N}$
- I_n is image n, $y_n \in \{+1, -1\}$ is associated label
- Risk function of model parameters:

$$E(\Phi, \Psi, \Omega, W) = 1/N \sum_{n=1}^{N} loss(y_n, \ell_n)$$

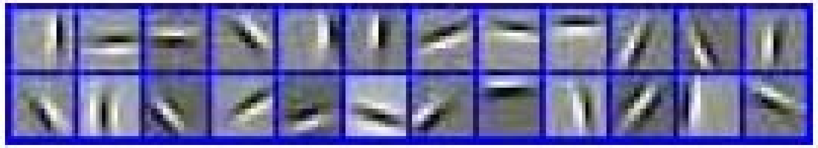
- Find model parameters $\hat{\Phi}, \hat{\Psi}, \hat{\Omega}, \hat{W}$ that minimize $E(\Phi, \Psi, \Omega, \mathrm{W})$

Cost Function vs. Model Parameters

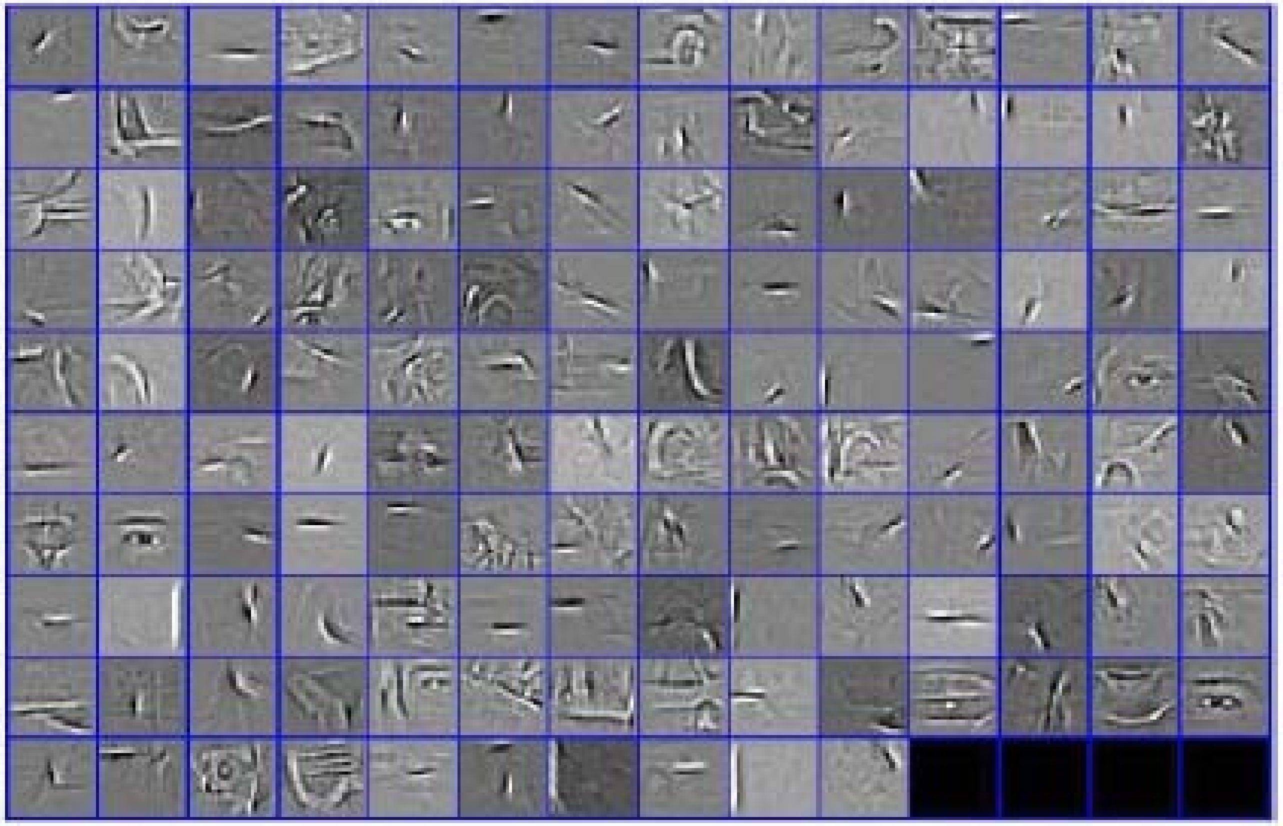


- High-dimensional function, as a consequence of a large number of model parameters
- Typically many local minima
- May be expensive to compute, for sophisticated models & large quantity of training images

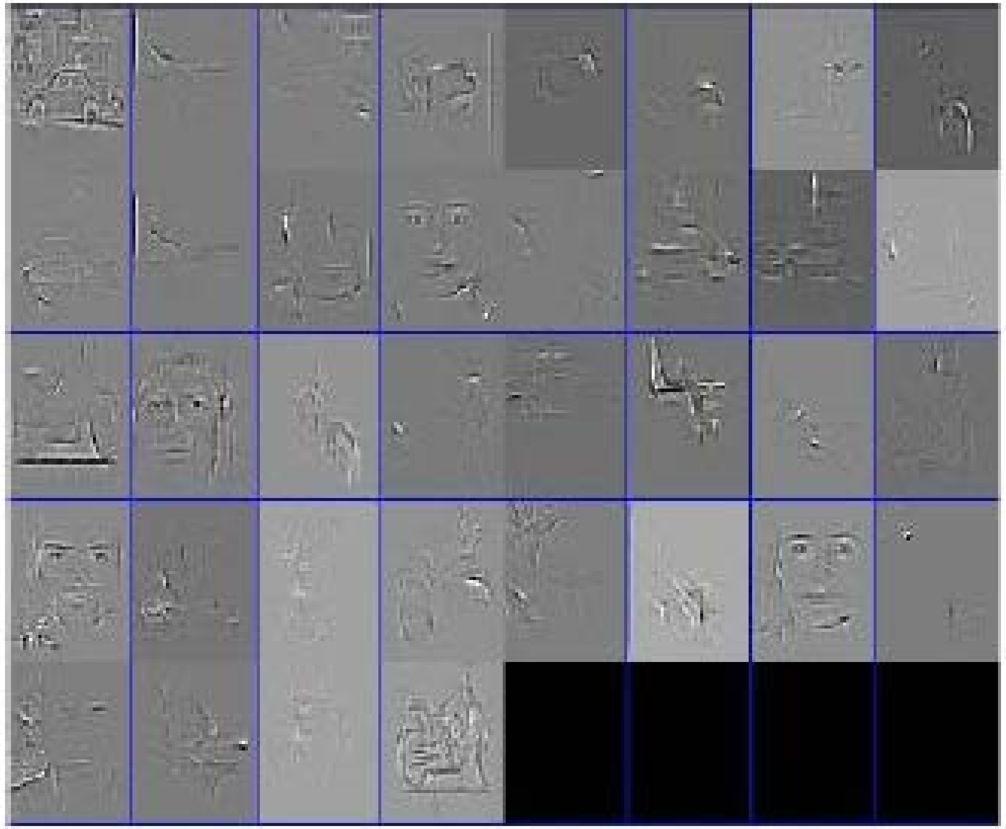
Layer 1



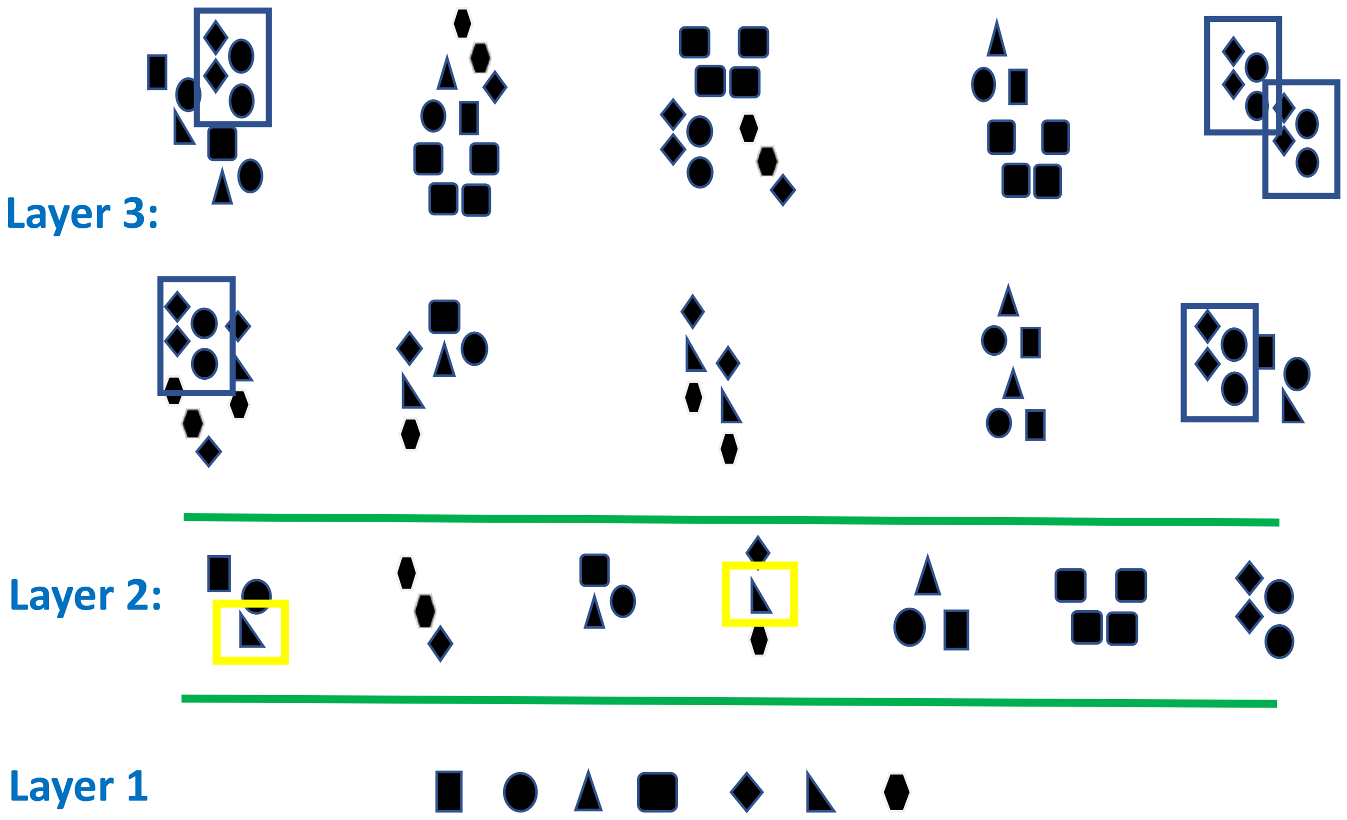
Layer 2



Layer 3



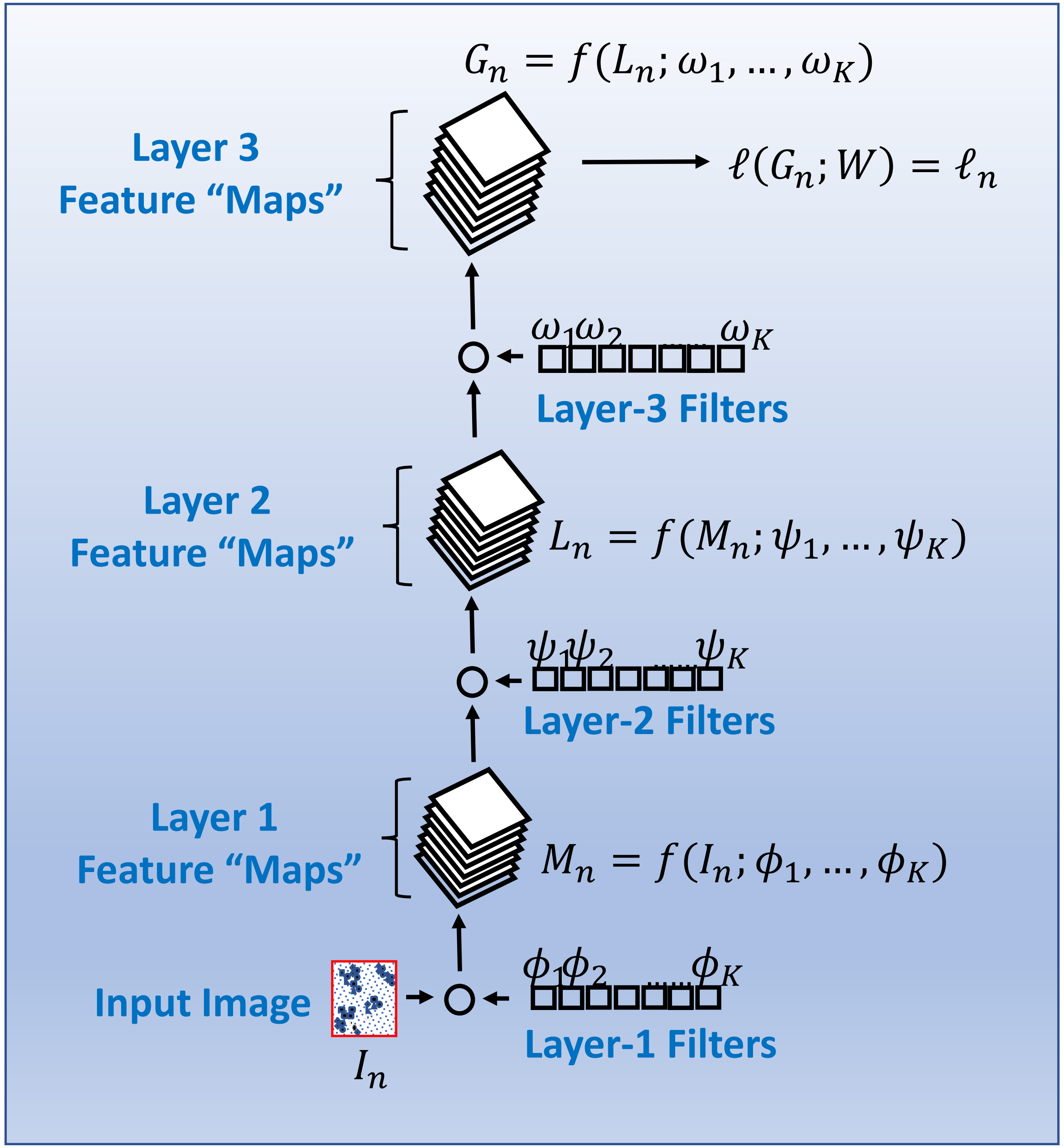
Advantage of Hierarchical Features?



- 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

Big Picture



- 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
- Risk function of model parameters:

$$E(\Phi, \Psi, \Omega, W) = 1/N \sum_{n=1}^{N} loss(y_n, \ell_n)$$

- Find model parameters $\widehat{\Phi}, \widehat{\Psi}, \widehat{\Omega}, \widehat{W}$ that minimize $E(\Phi, \Psi, \Omega, \mathrm{W})$