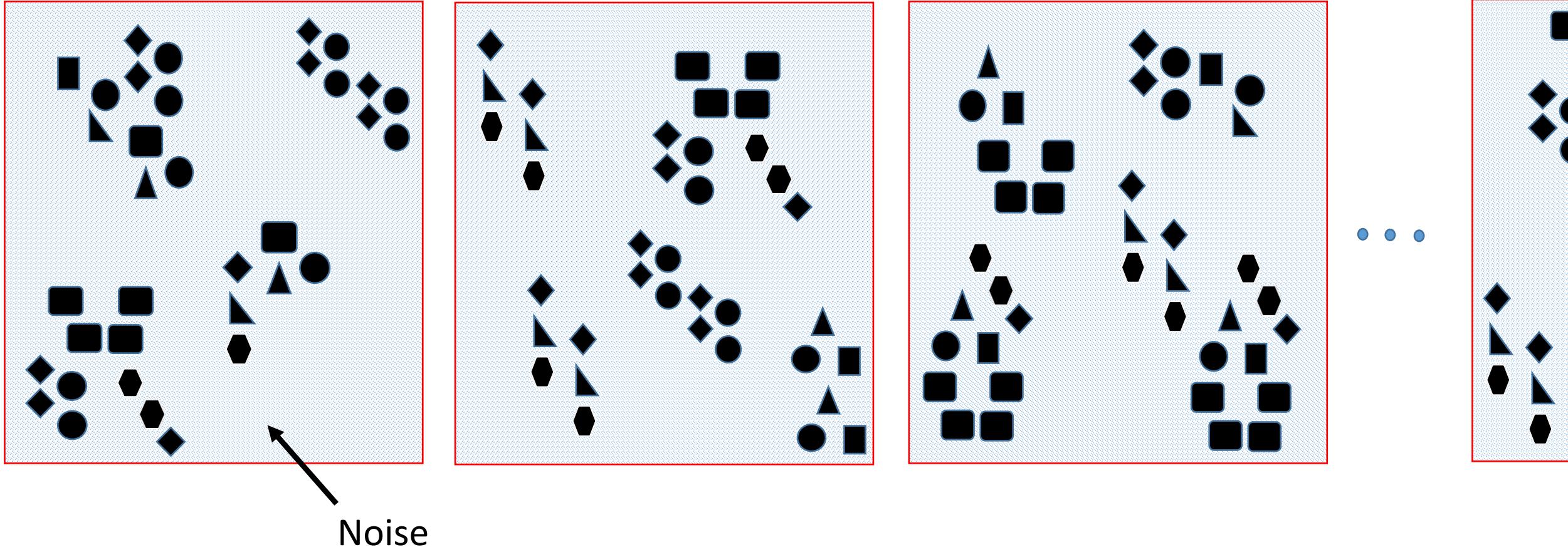


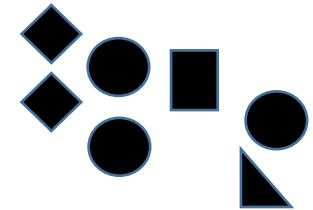
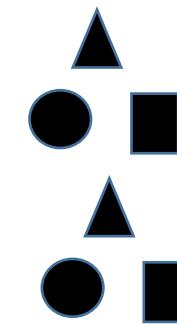
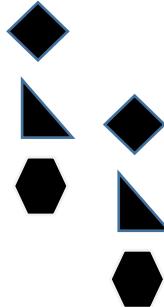
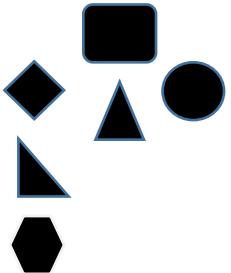
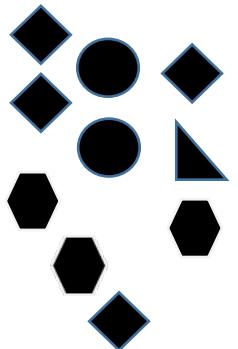
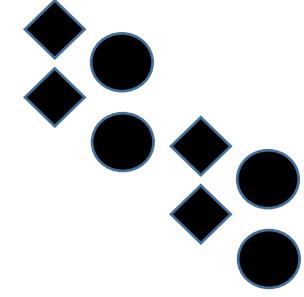
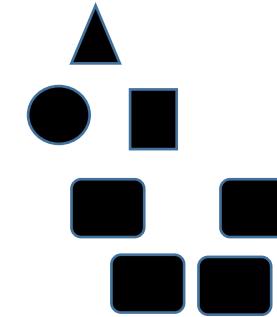
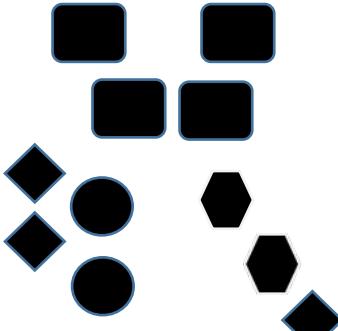
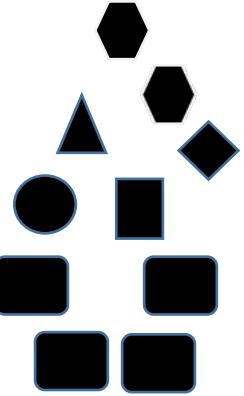
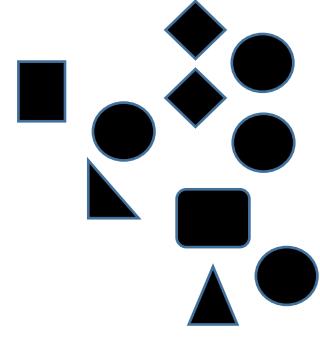
Deep Learning and Big Data

Lawrence Carin
Duke University

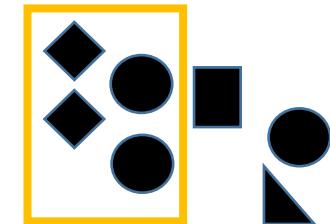
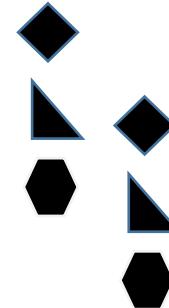
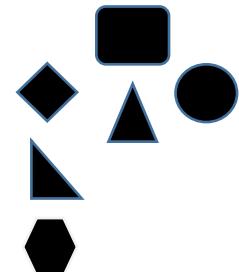
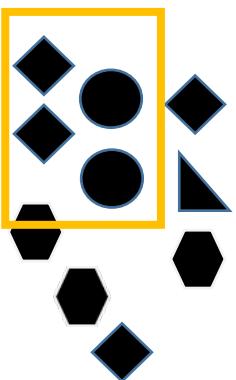
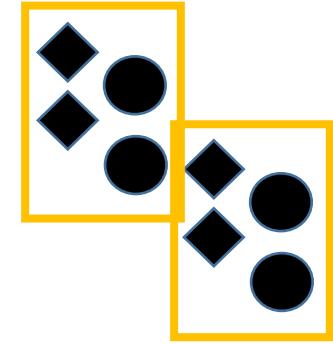
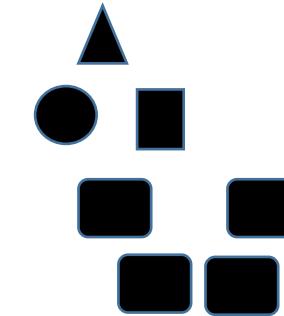
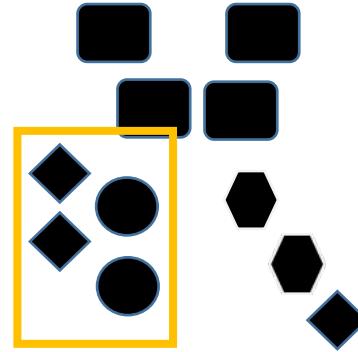
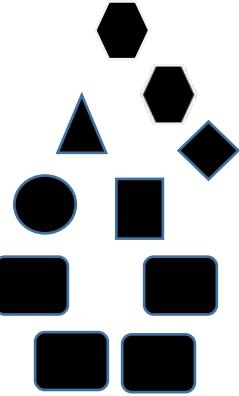
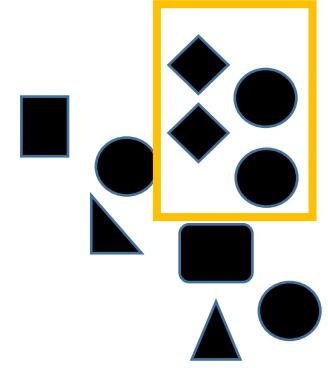
Consider a Set of “Toy” Images, for Illustration



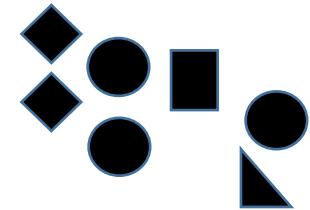
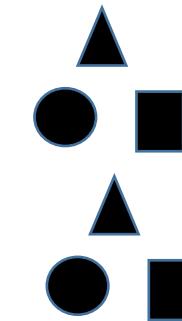
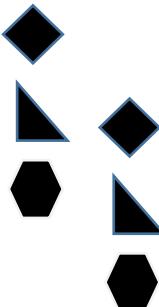
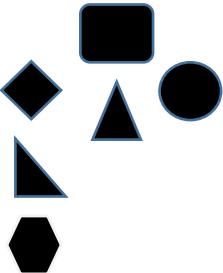
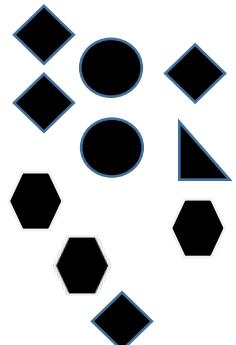
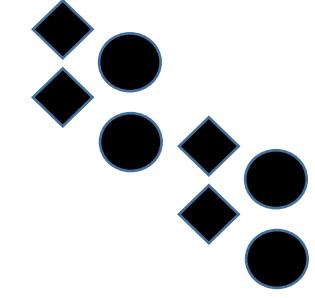
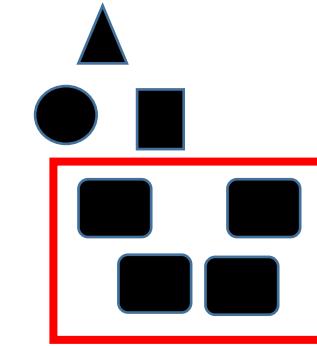
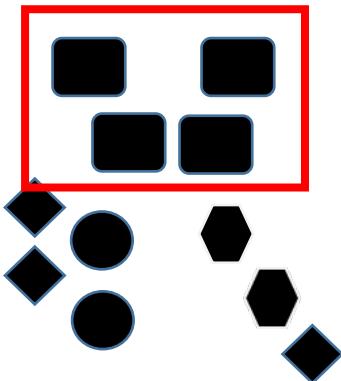
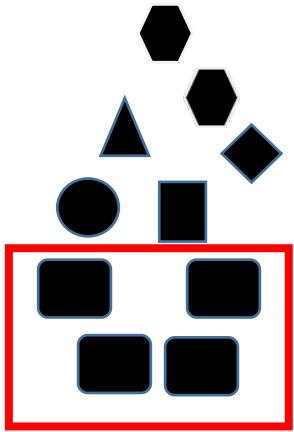
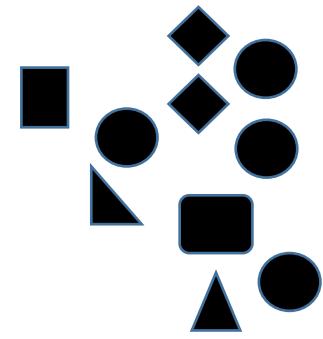
High-Level Motifs



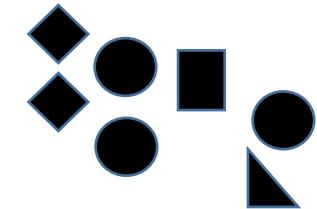
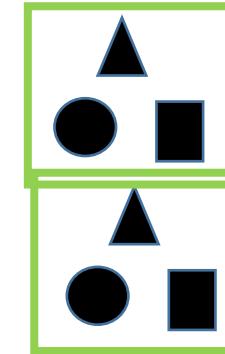
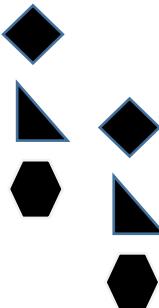
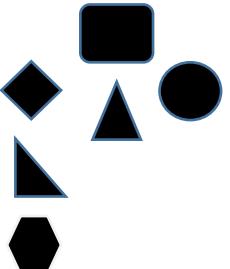
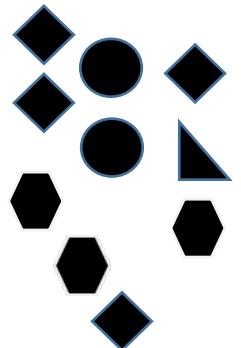
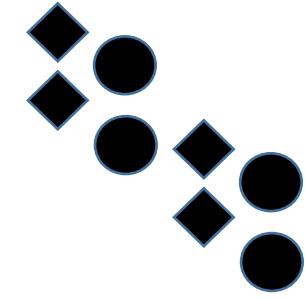
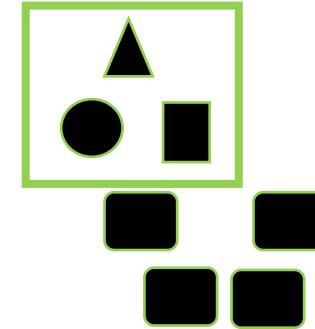
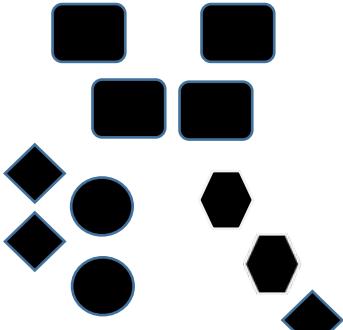
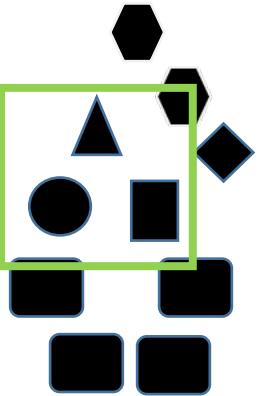
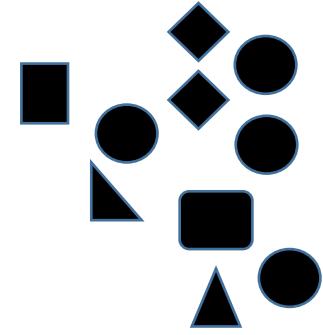
Shared Substructure Within Motifs



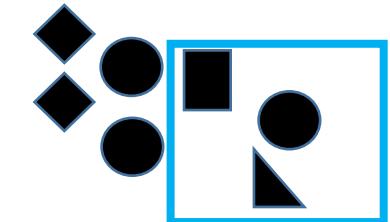
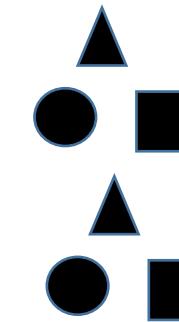
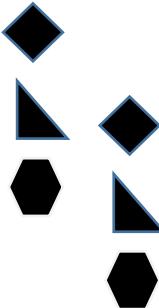
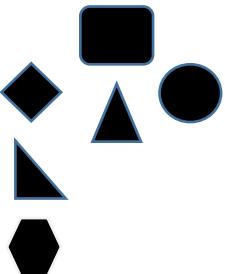
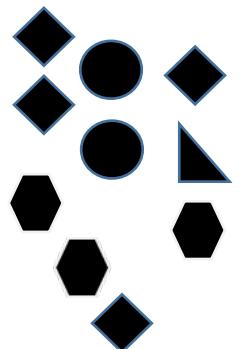
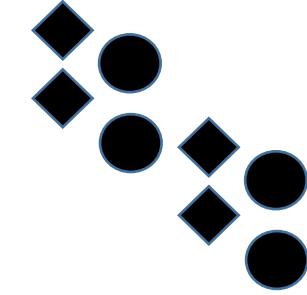
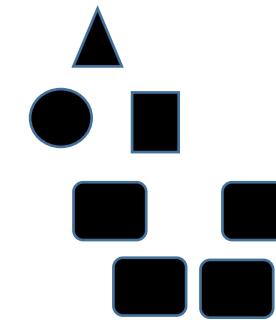
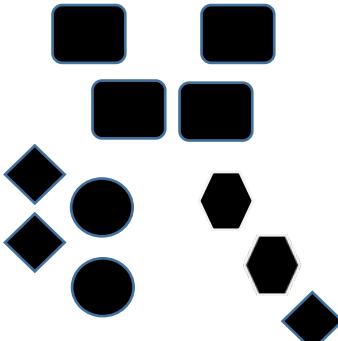
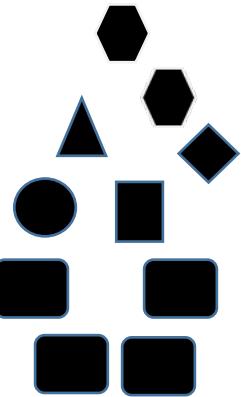
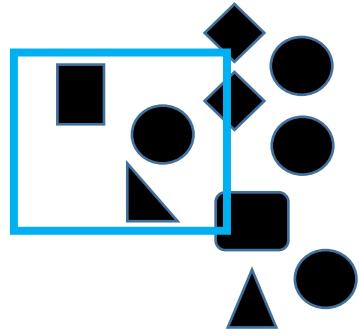
Shared Substructure Within Motifs



Shared Substructure Within Motifs

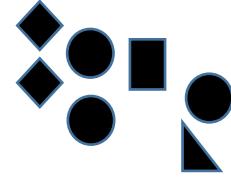
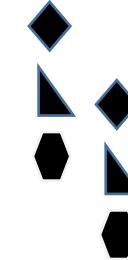
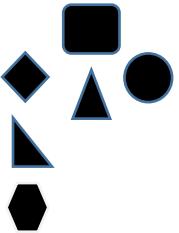
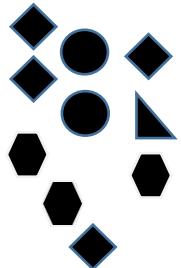
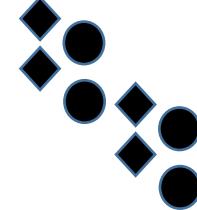
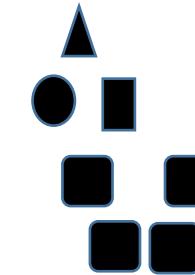
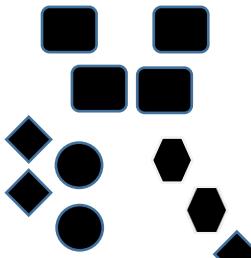
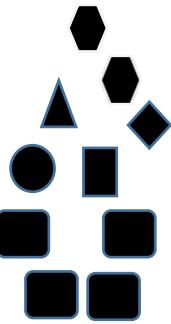
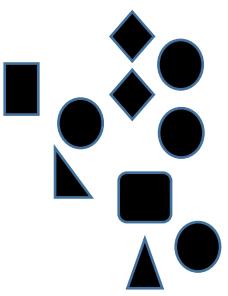


Shared Substructure Within Motifs

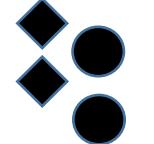


Hierarchical Representation of Images

 Layer 3:
Motifs



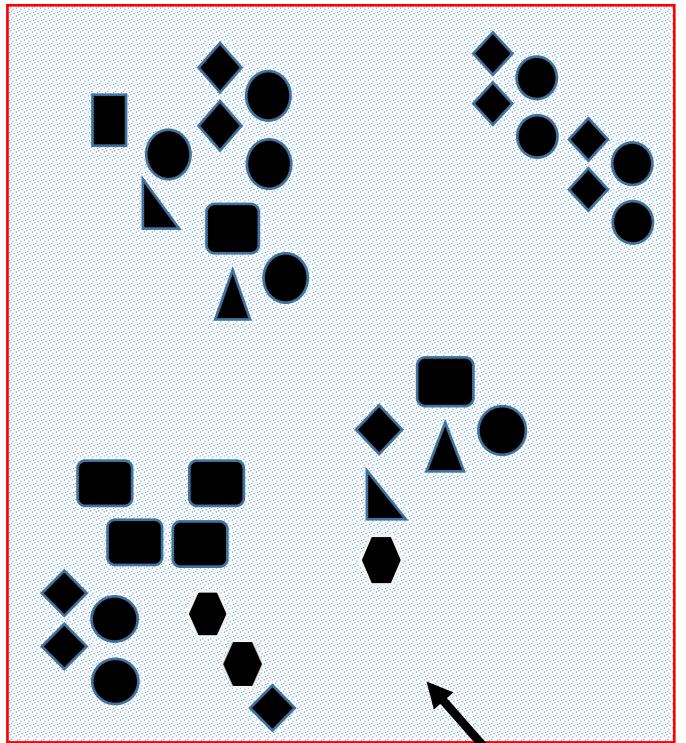
Layer 2:
Sub-Motifs



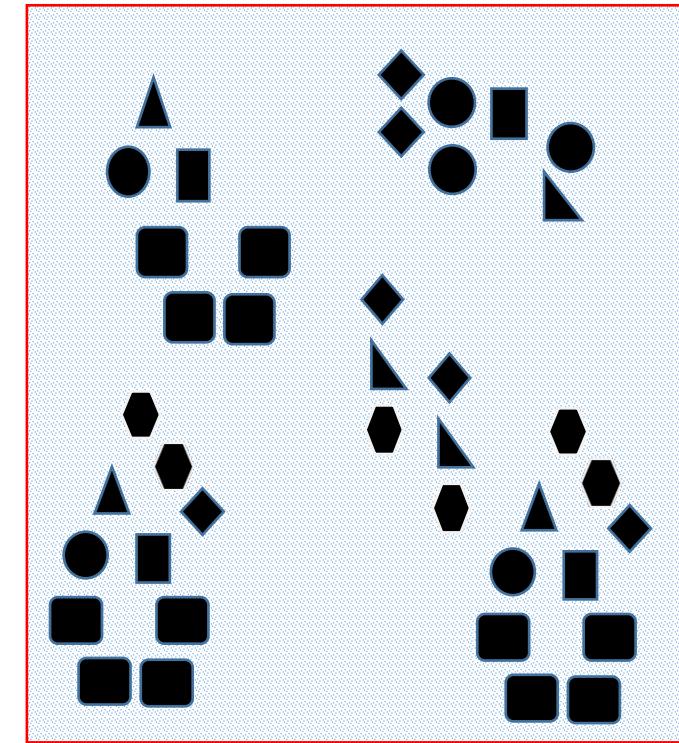
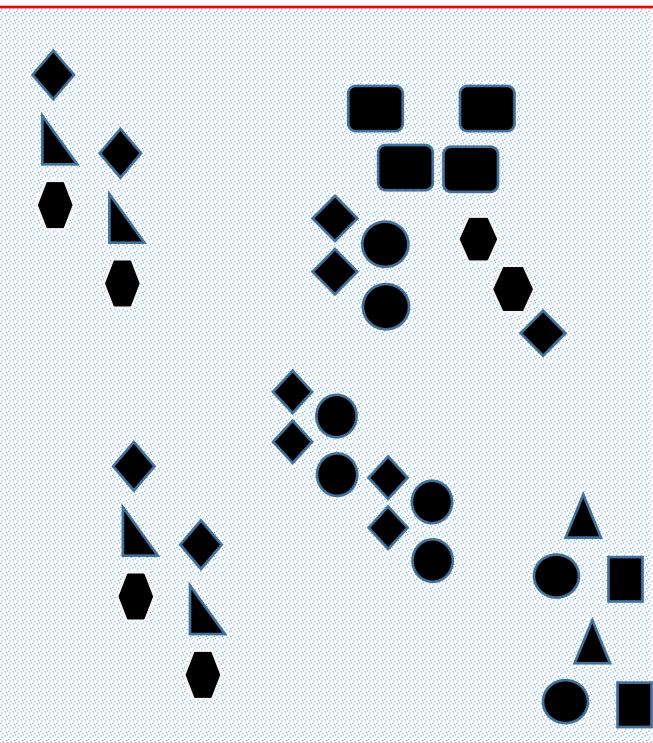
Layer 1:
Fundamental Building Blocks



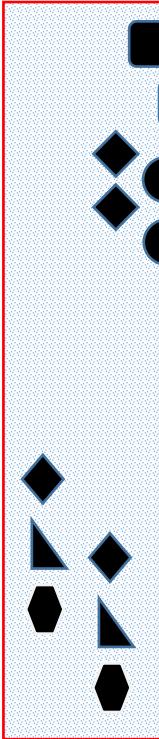
Recall the Data/Images



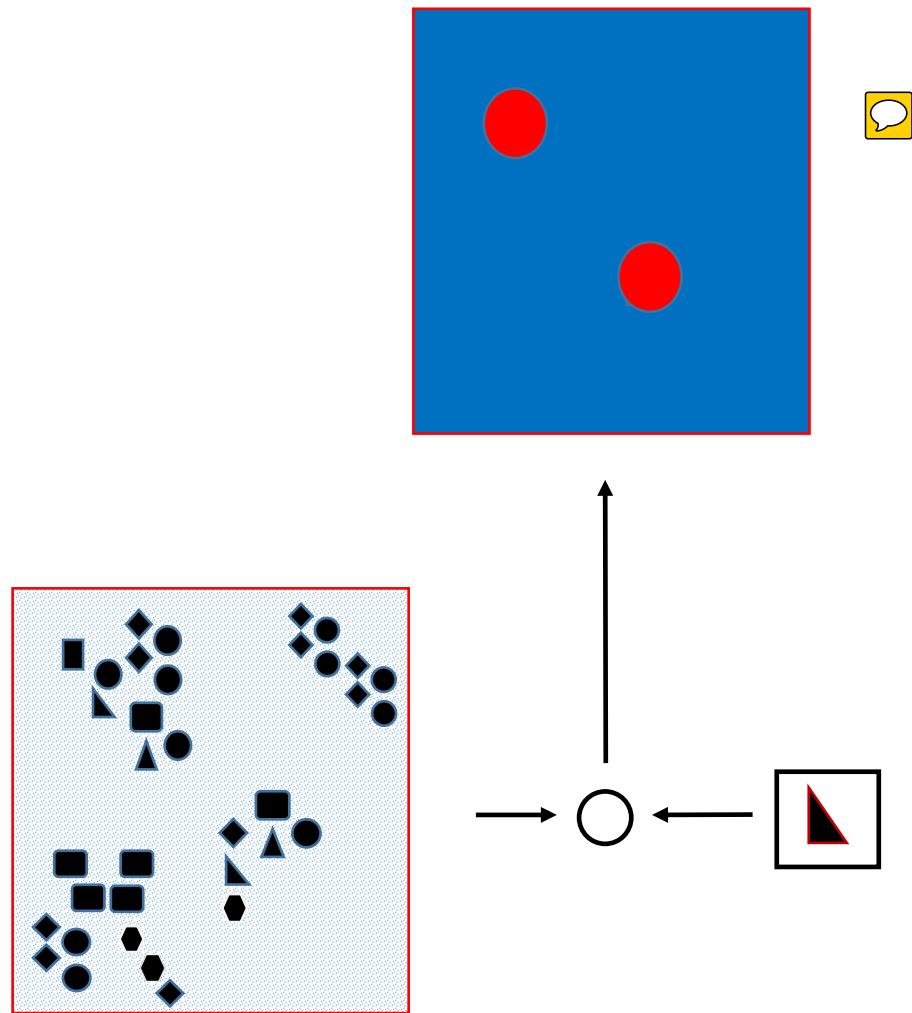
Noise



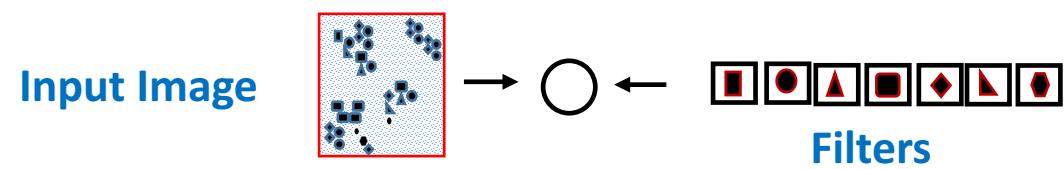
...

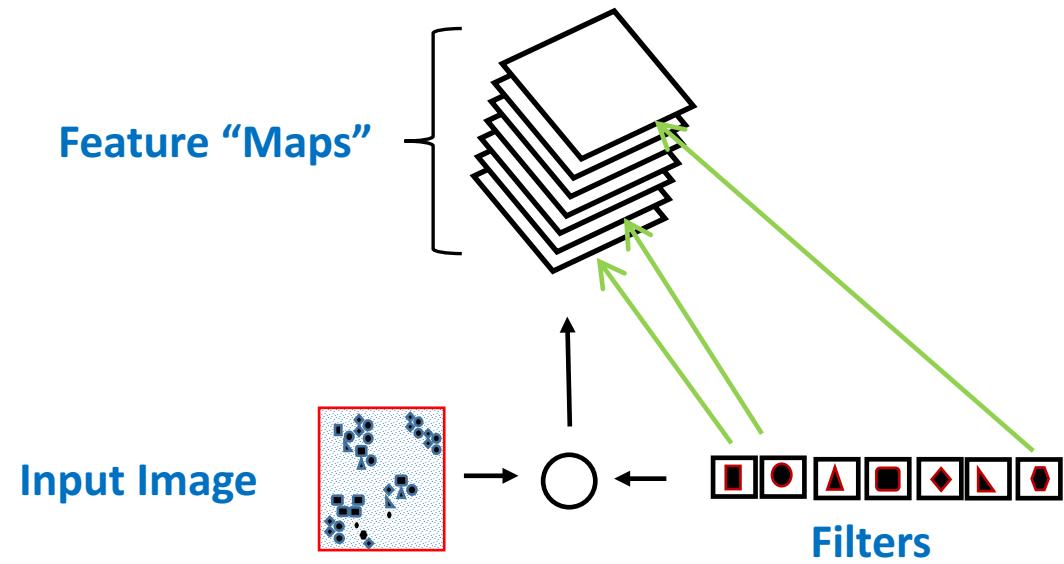


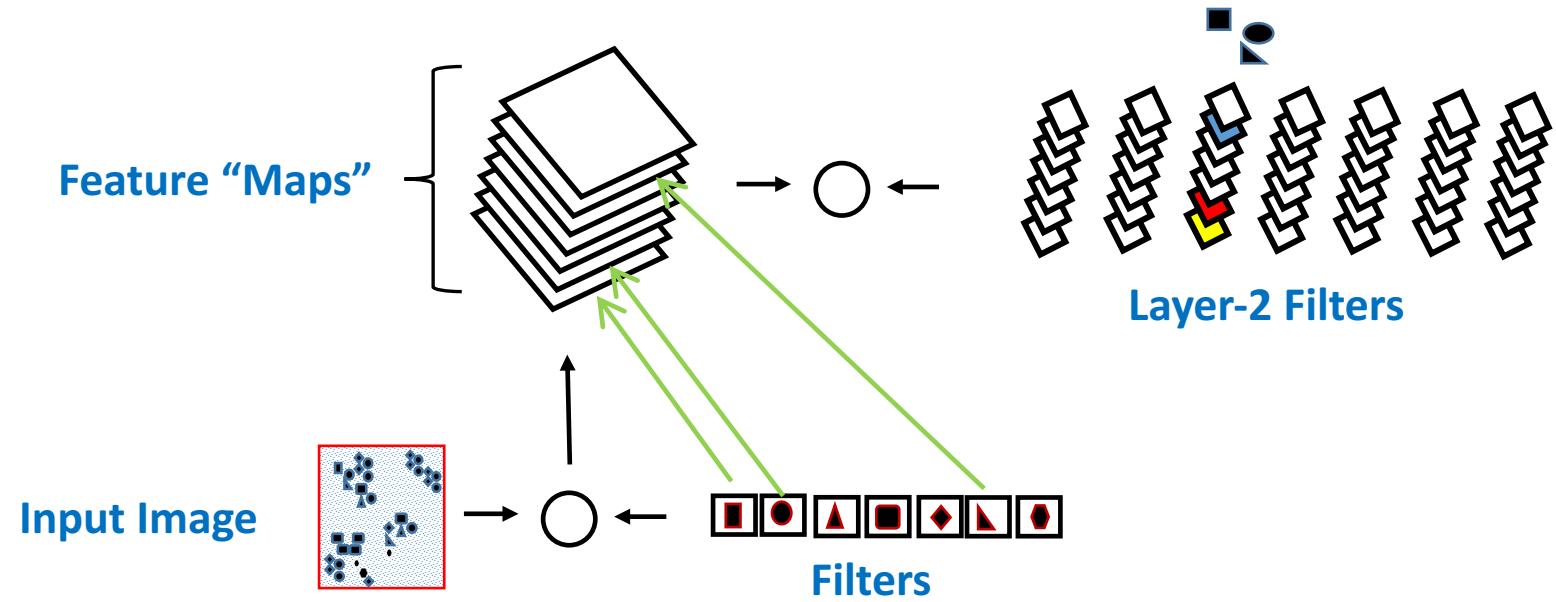
Convolutional Filter

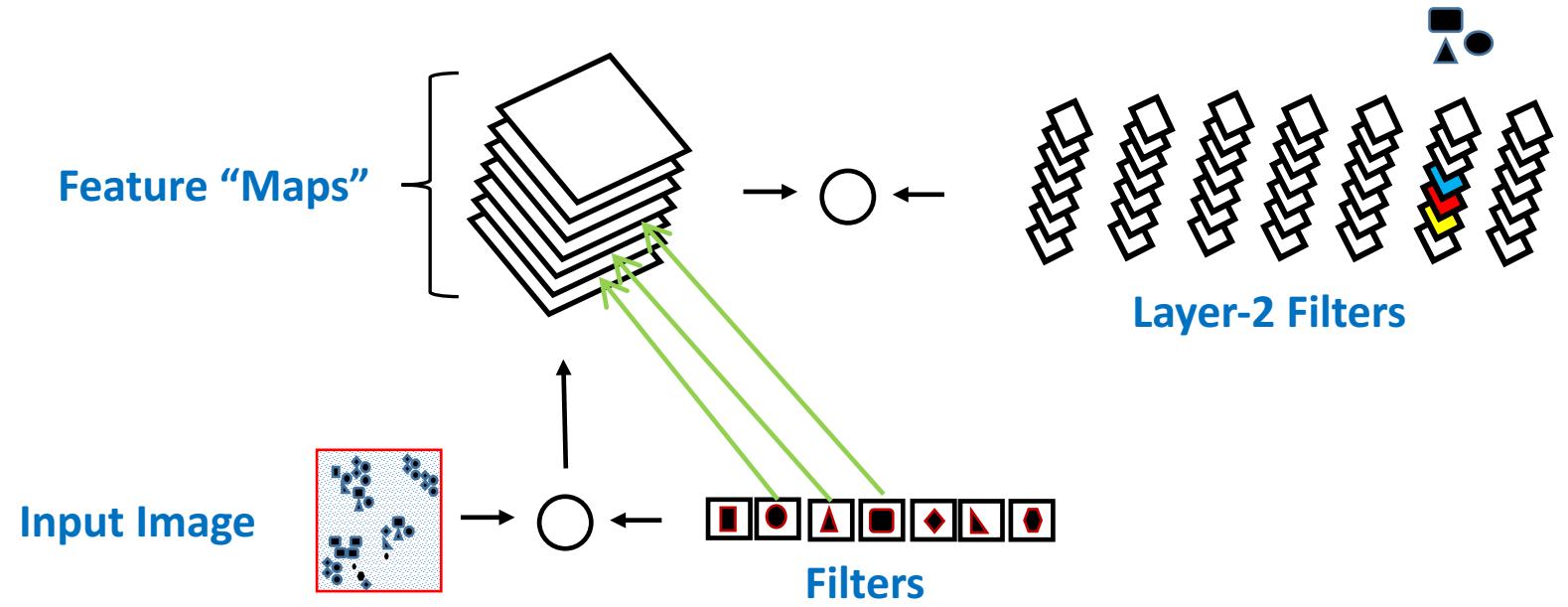


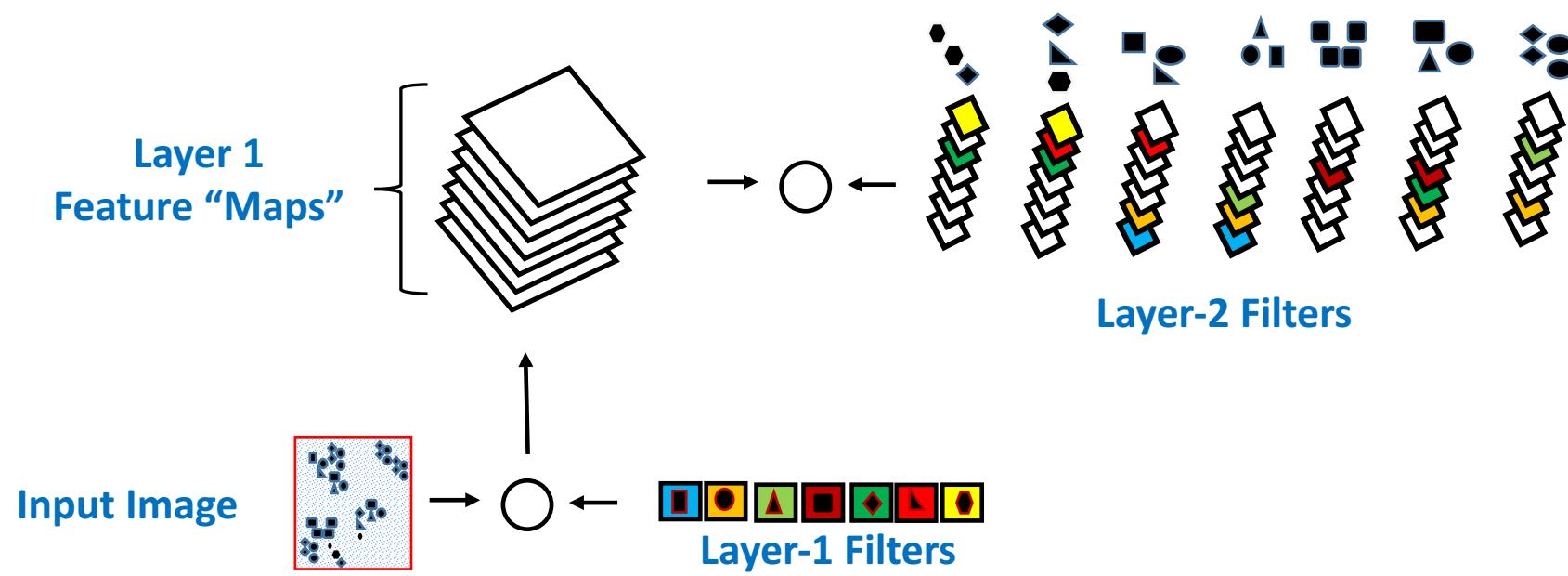
Multiple Filters, for Each Building Block

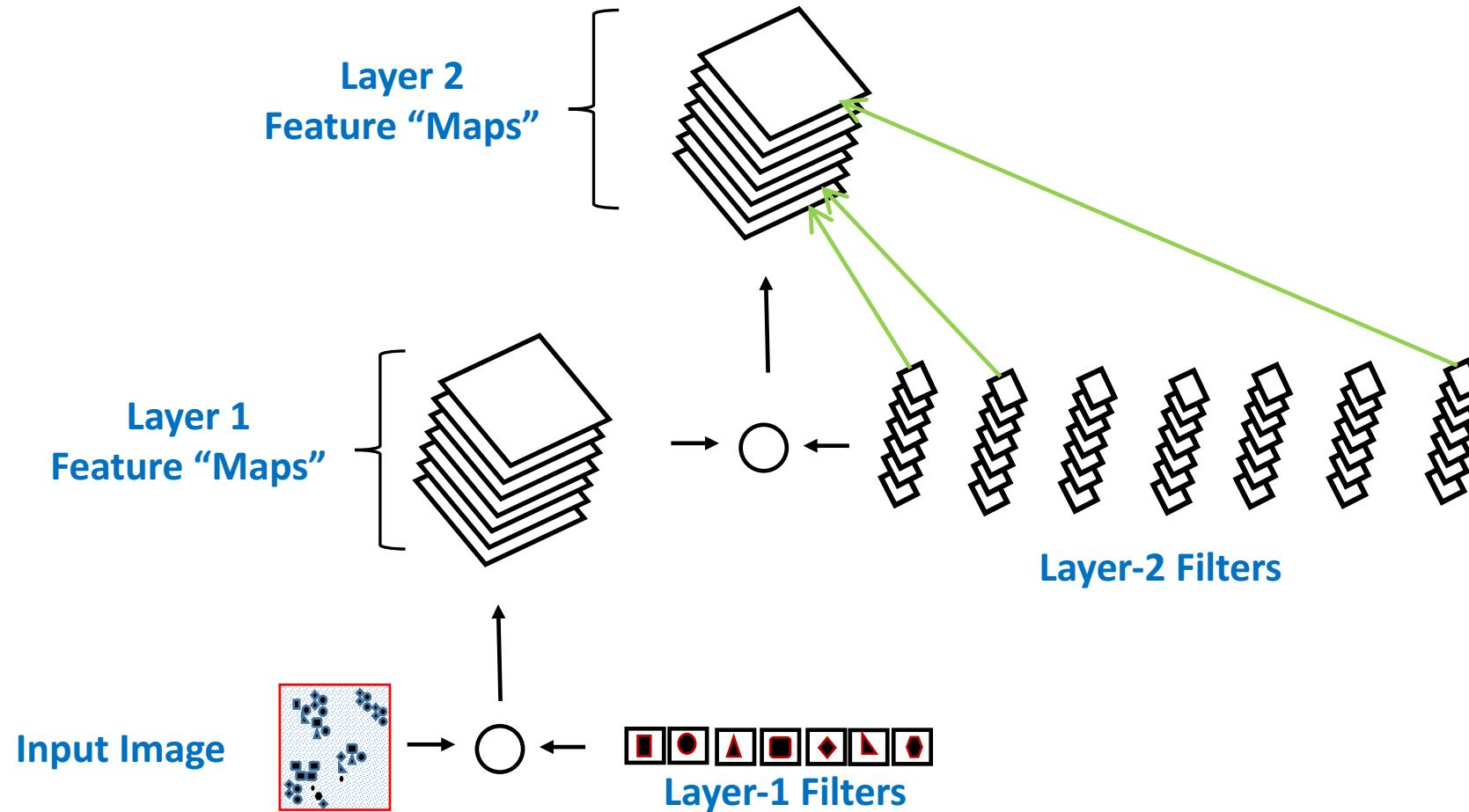


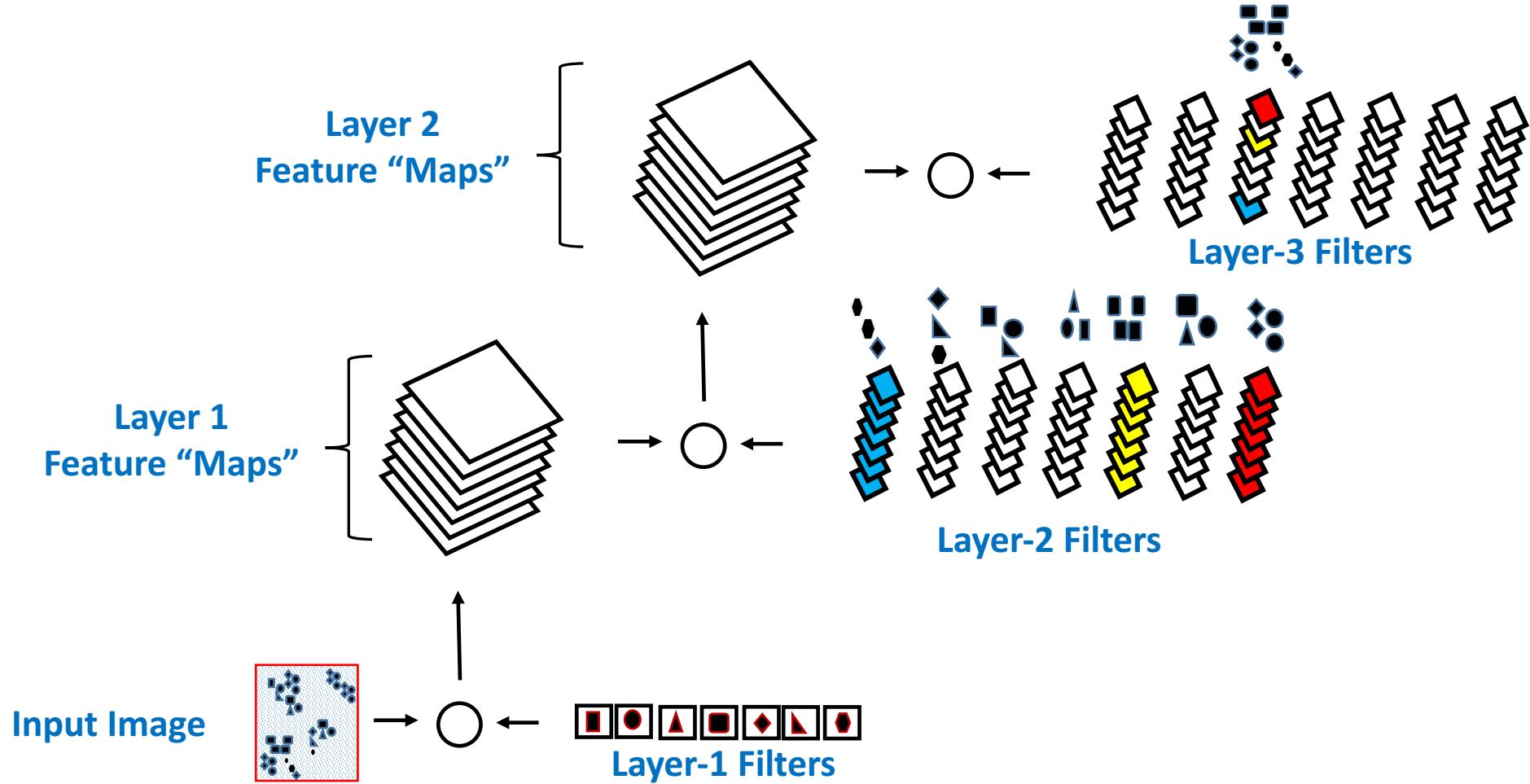


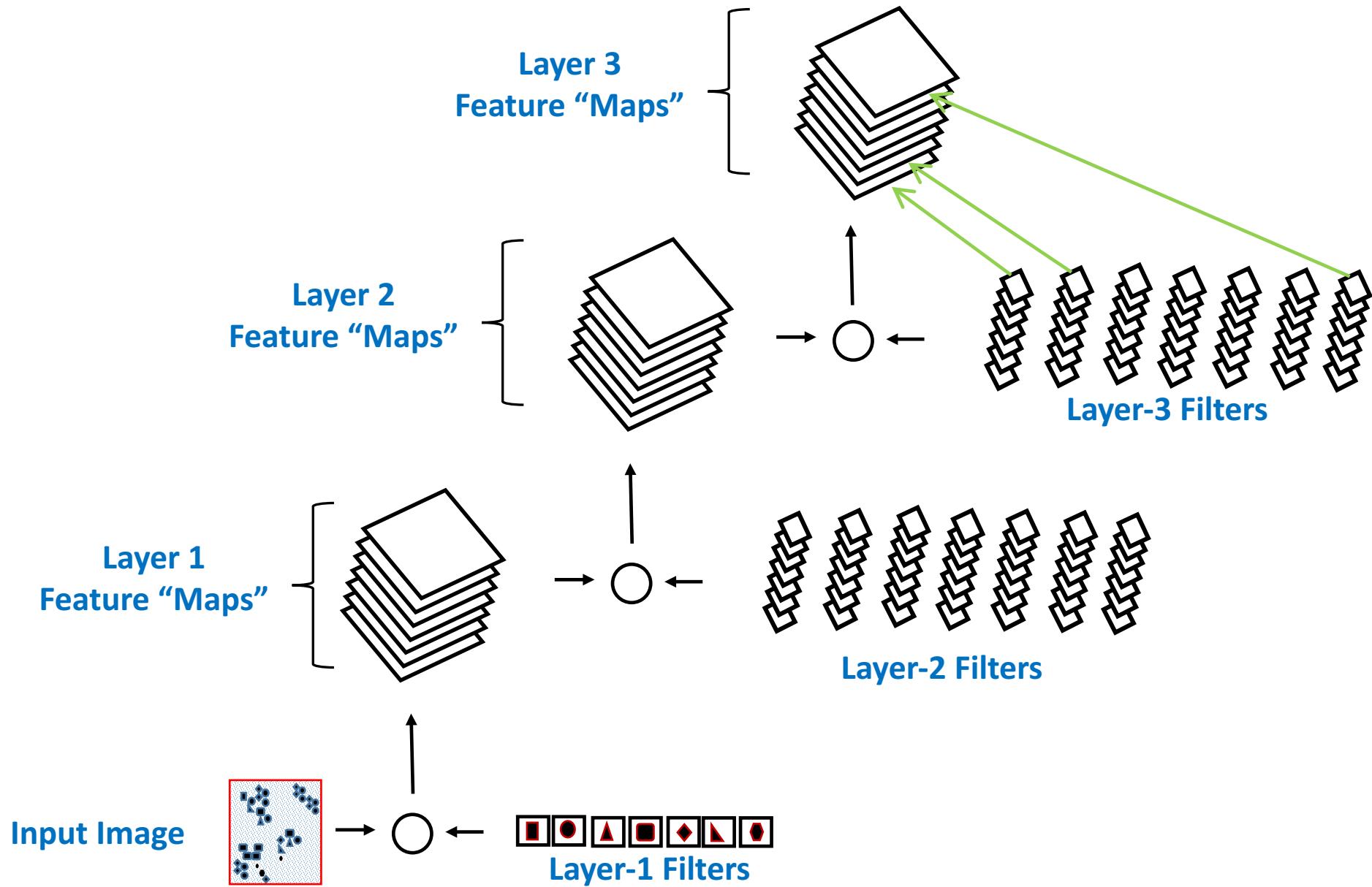




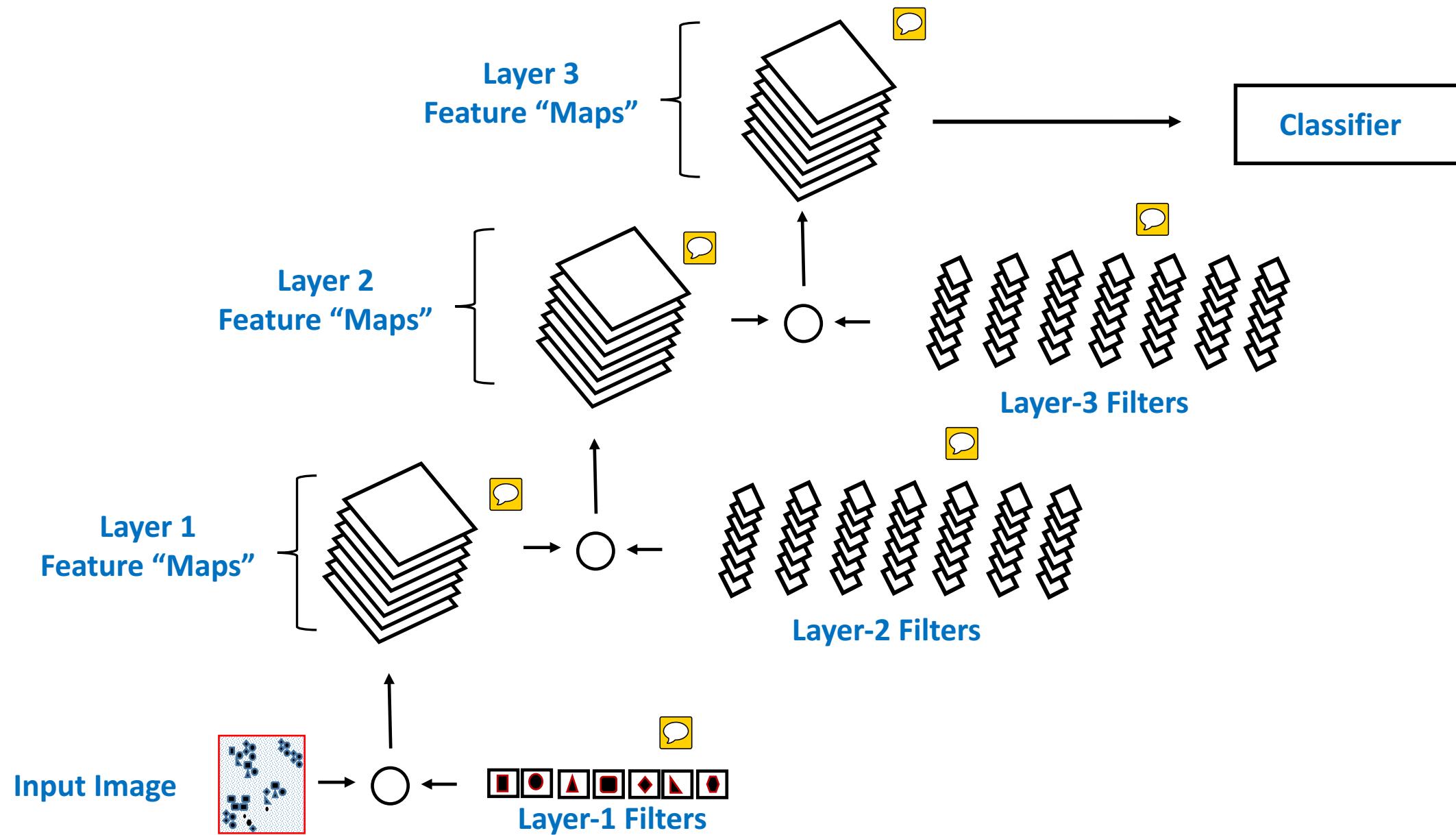








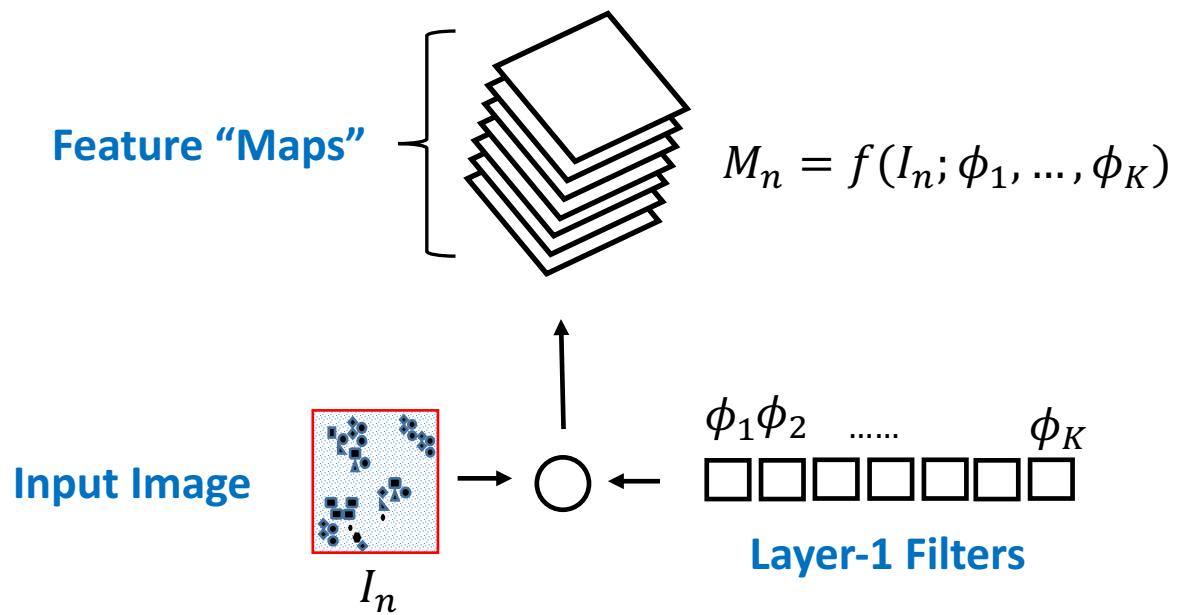
Deep Analysis Architecture

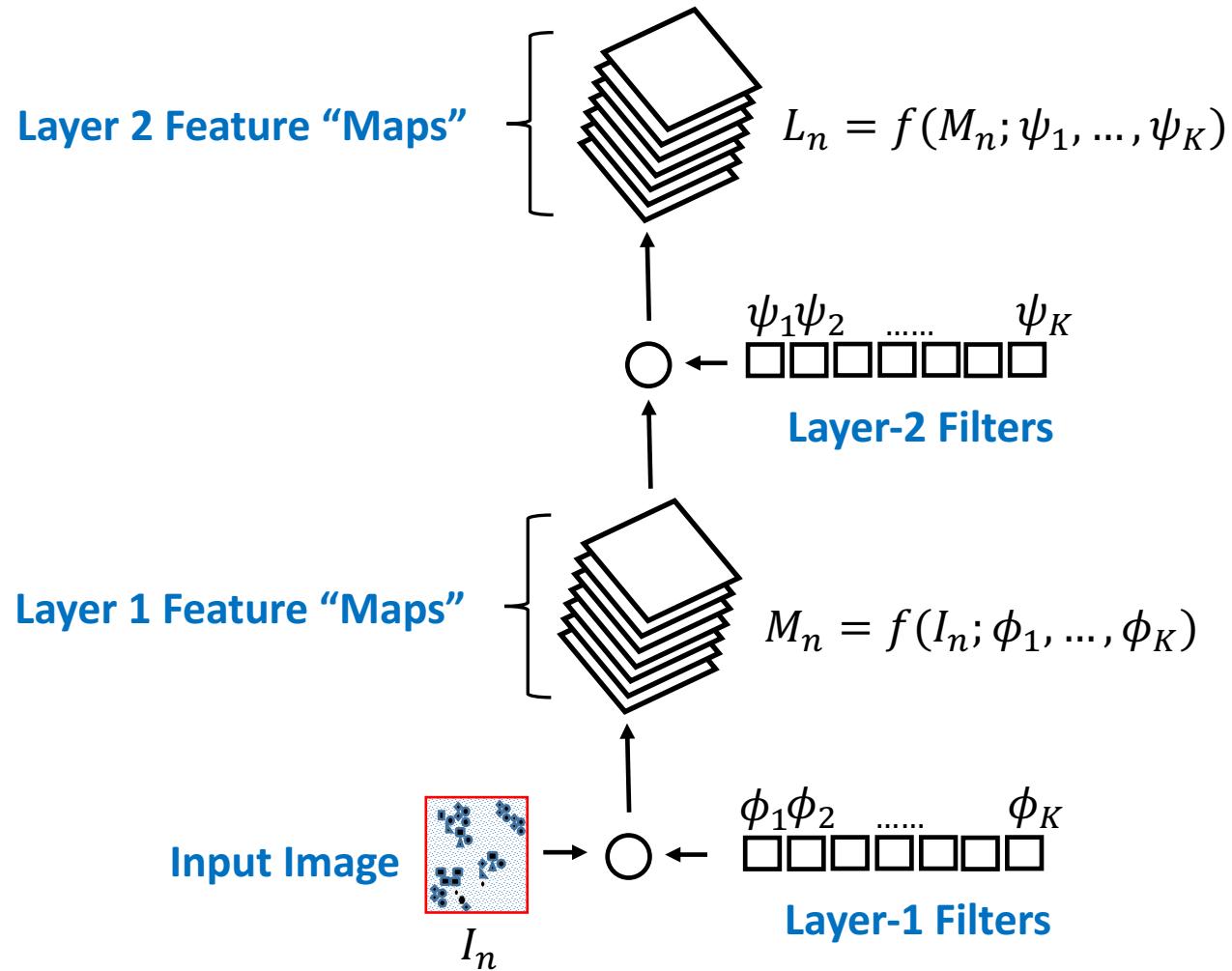


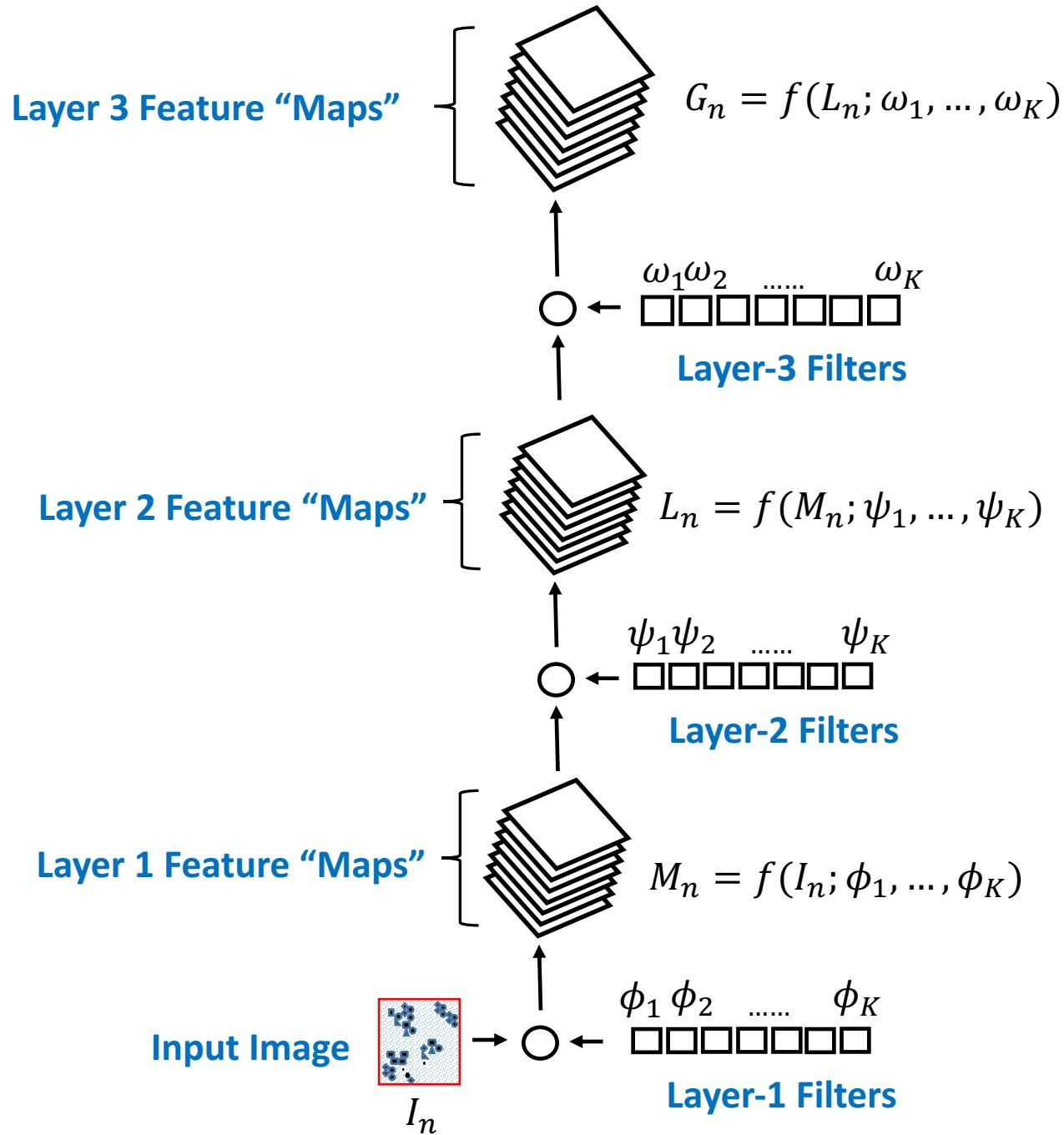
Given Images, How Do We Learn Model Parameters?

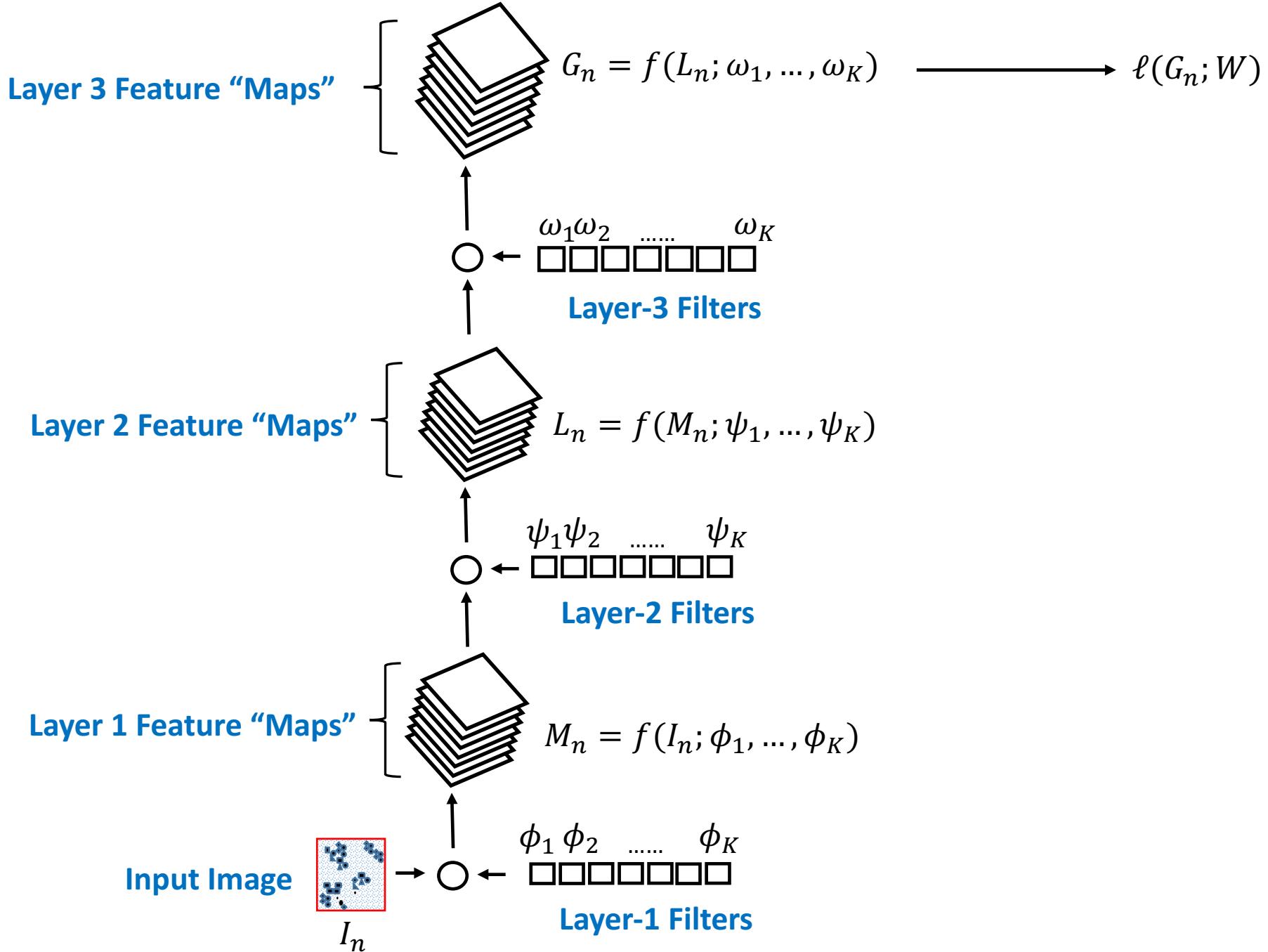


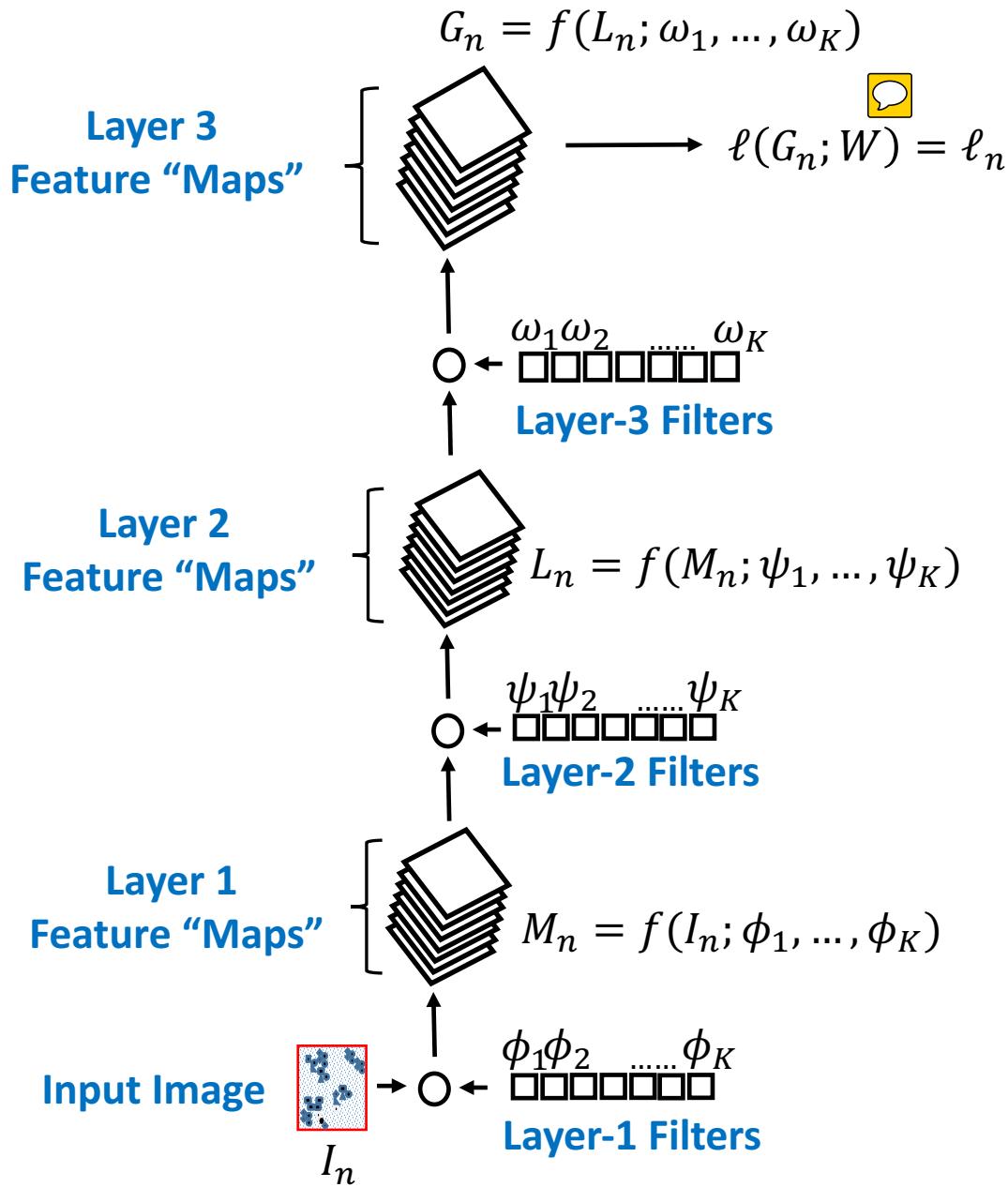
- The previous discussion was an illustration for motivating the “deep” algorithm concept
- Demonstrated using “toy” images
- How do we build such an algorithm in practice, given a large set of training images?









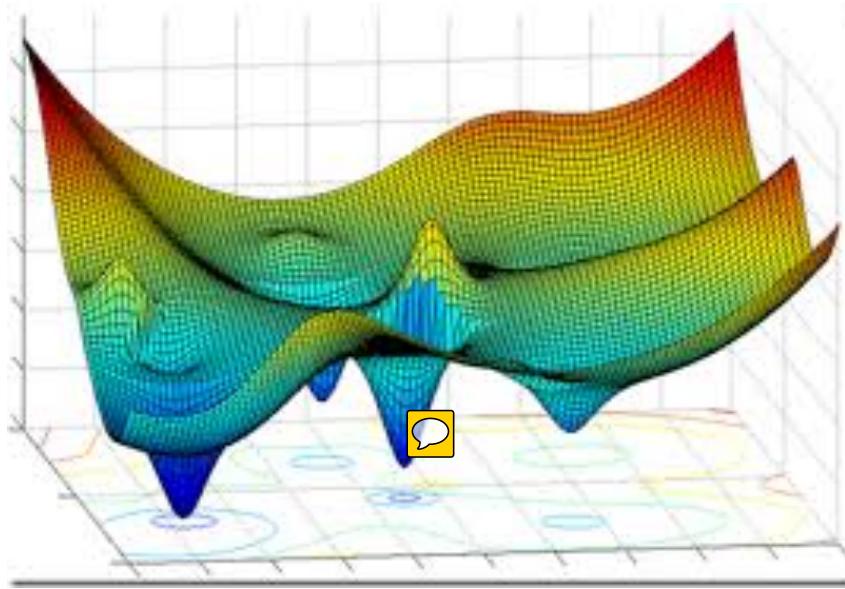


- Assume we have labeled images $\{I_n, y_n\}_{n=1,N}$
- I_n is image n , $y_n \in \{+1, -1\}$ is associated label
- Risk function of model parameters:

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

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

Cost Function vs. Model Parameters

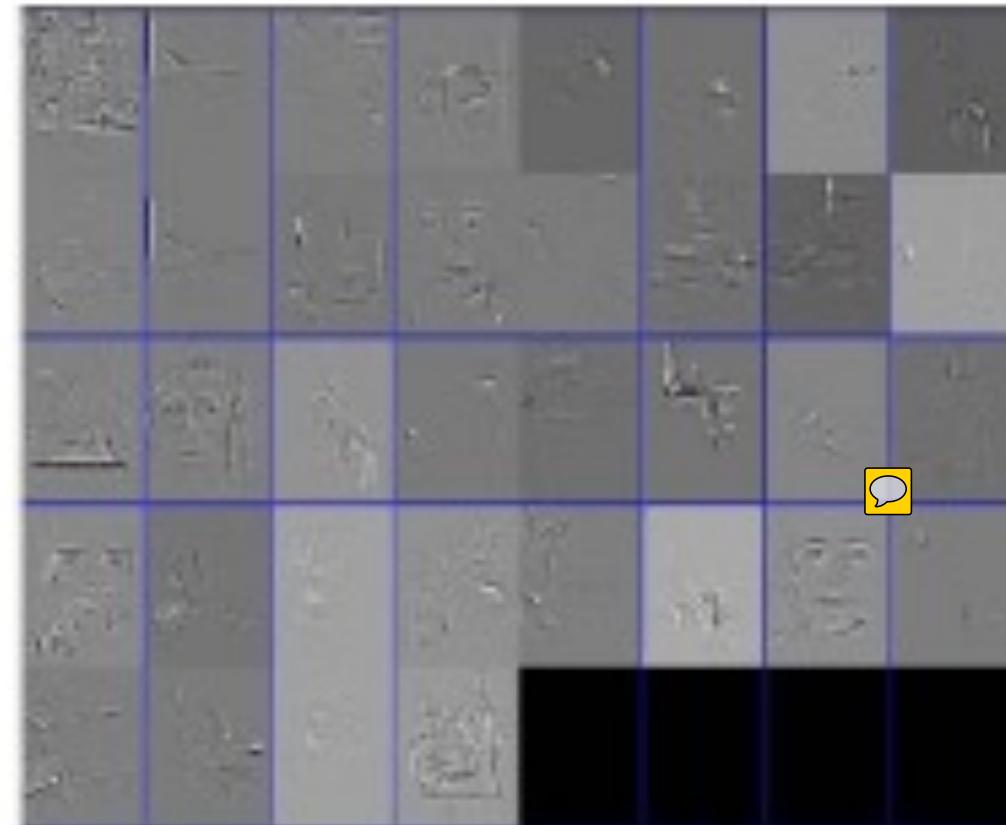
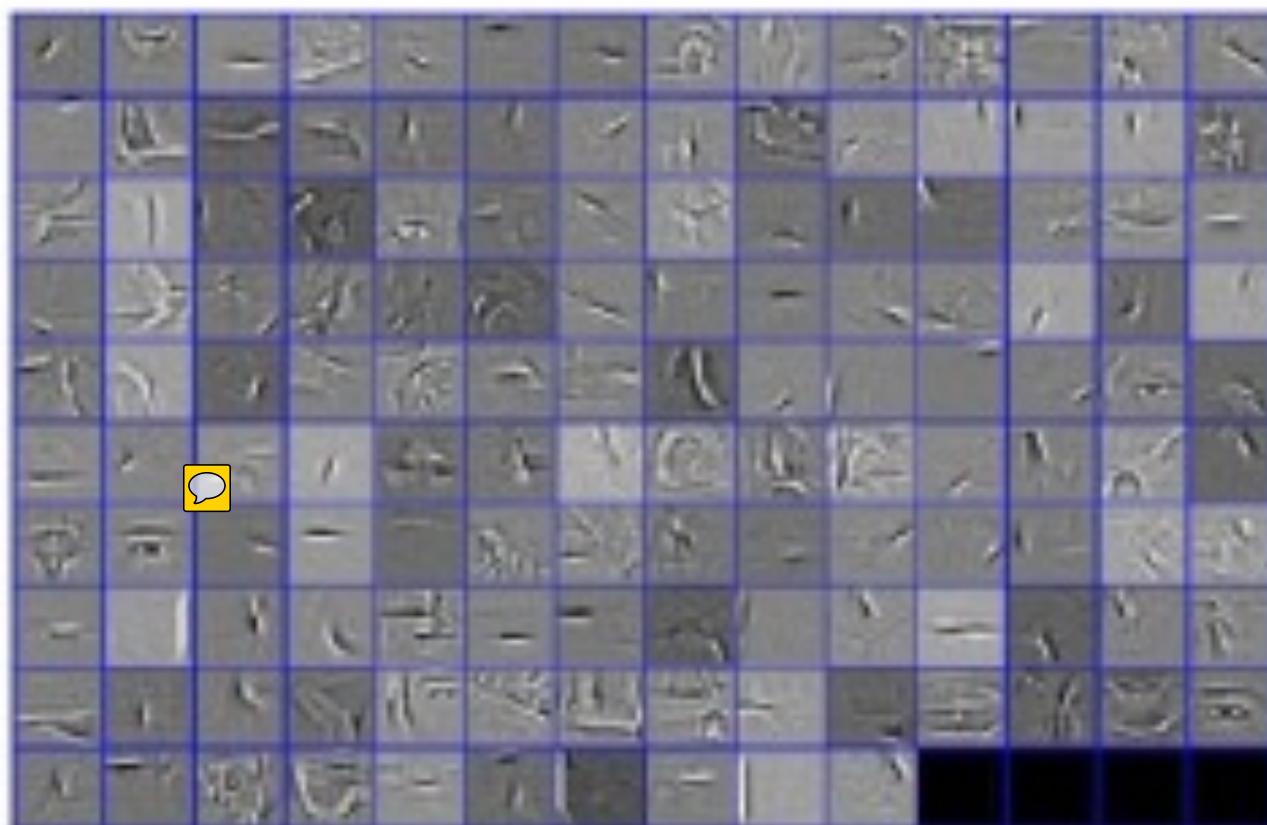


- High-dimensional function, as a consequence of a large number of model parameters
- Typically many local minima
- May be expensive to compute, for sophisticated models & large quantity of training images

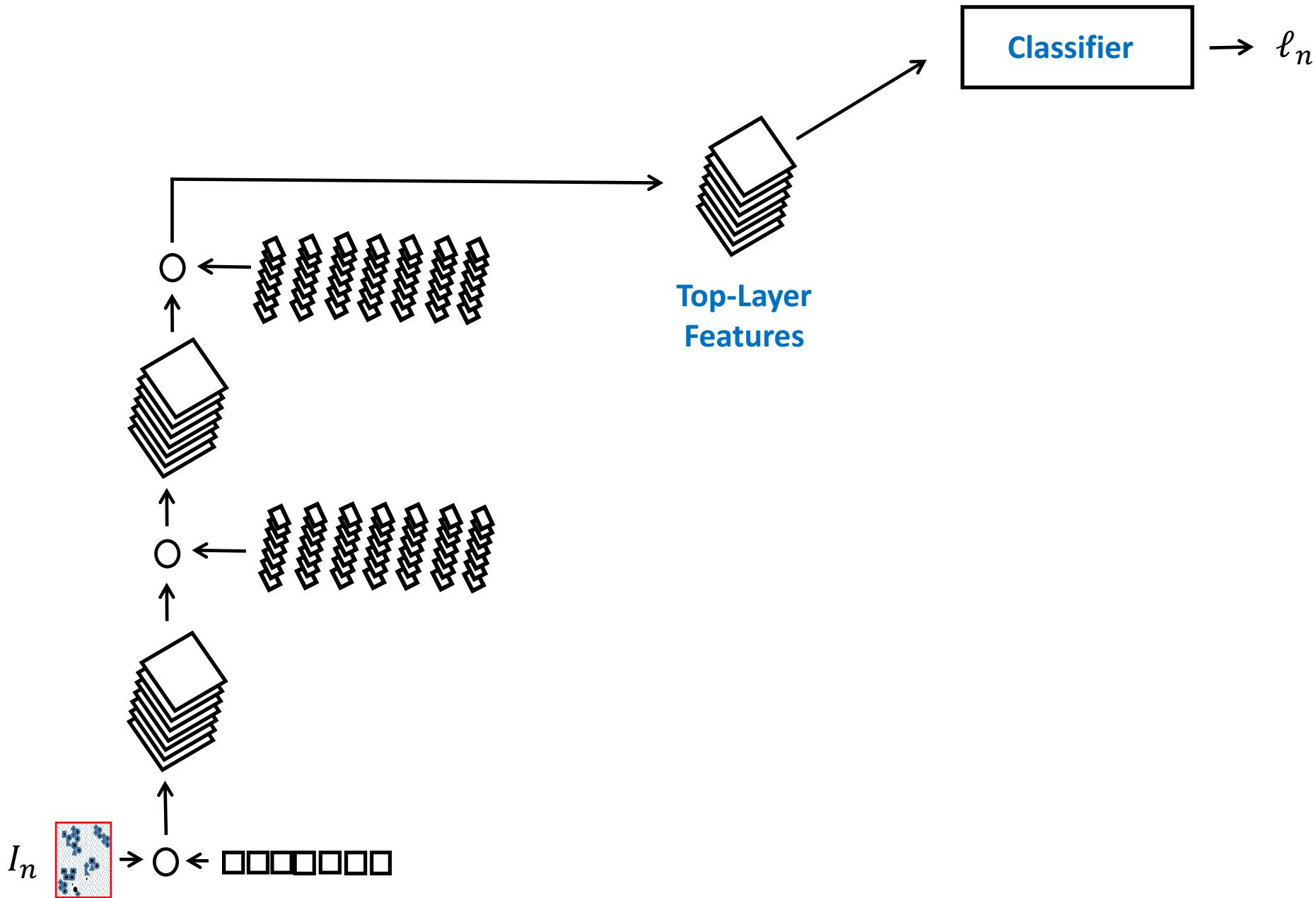
- Layer 1 dictionary elements:



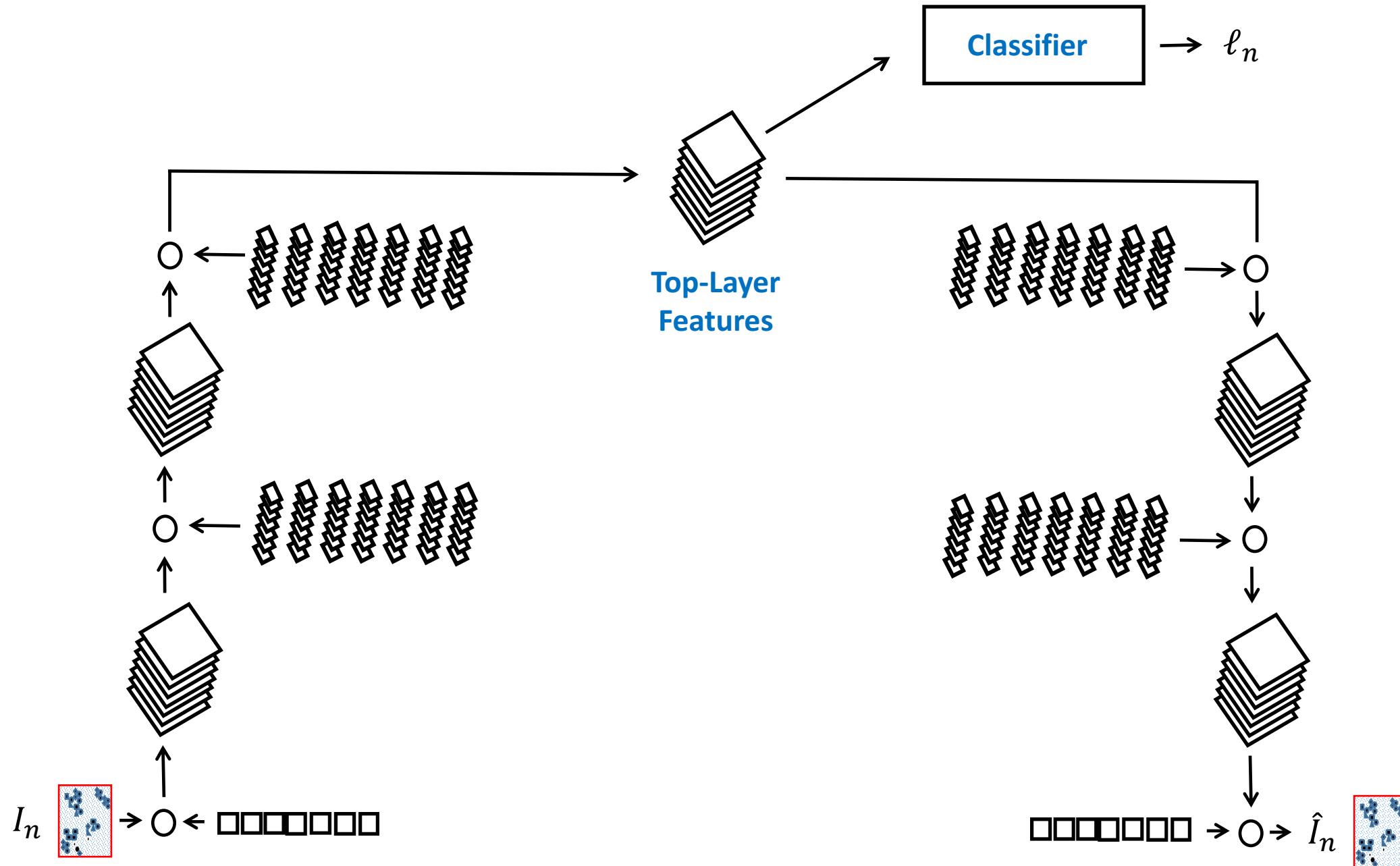
- Second (left) and third (right) layer dictionary elements:



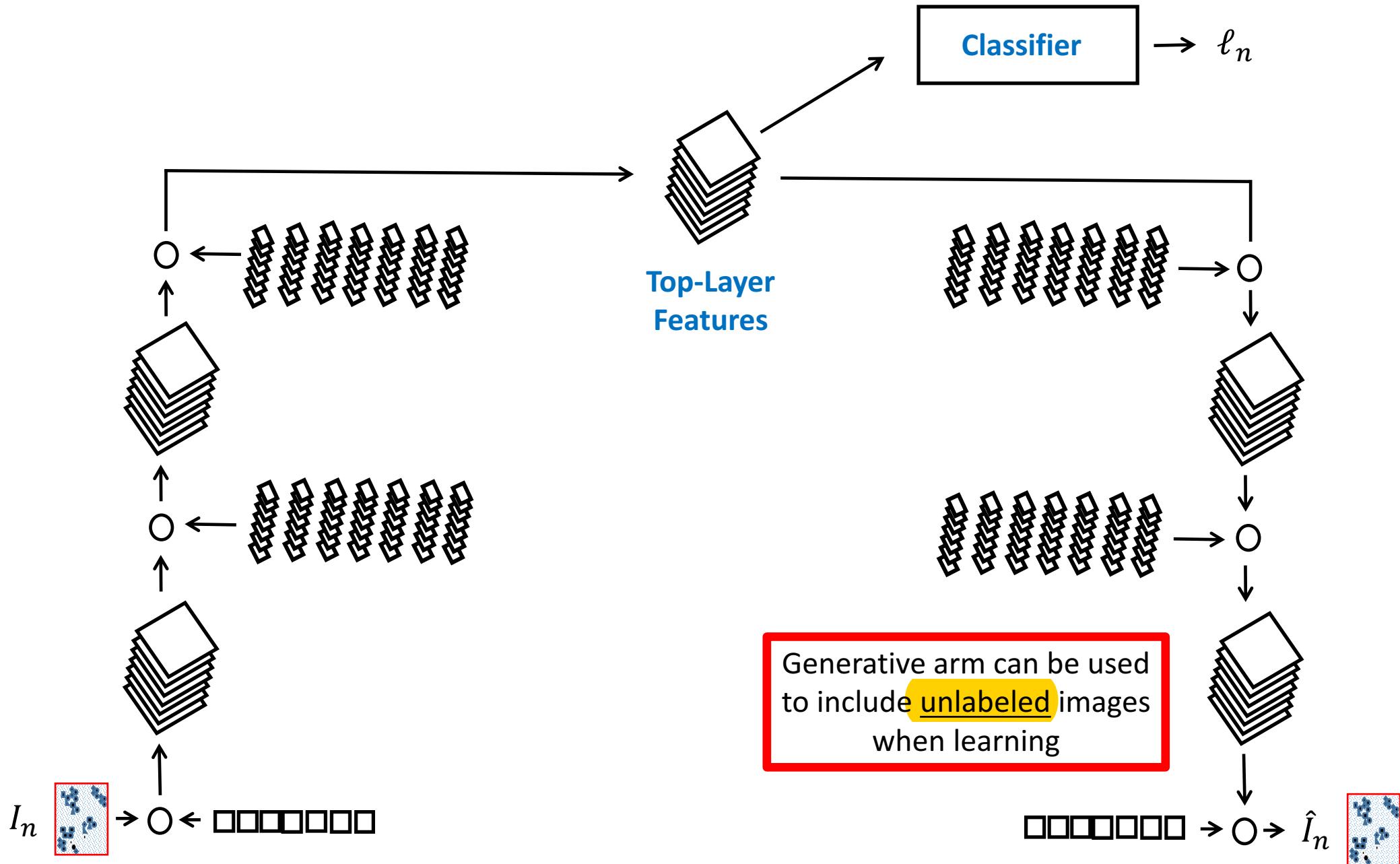
Deep Architecture



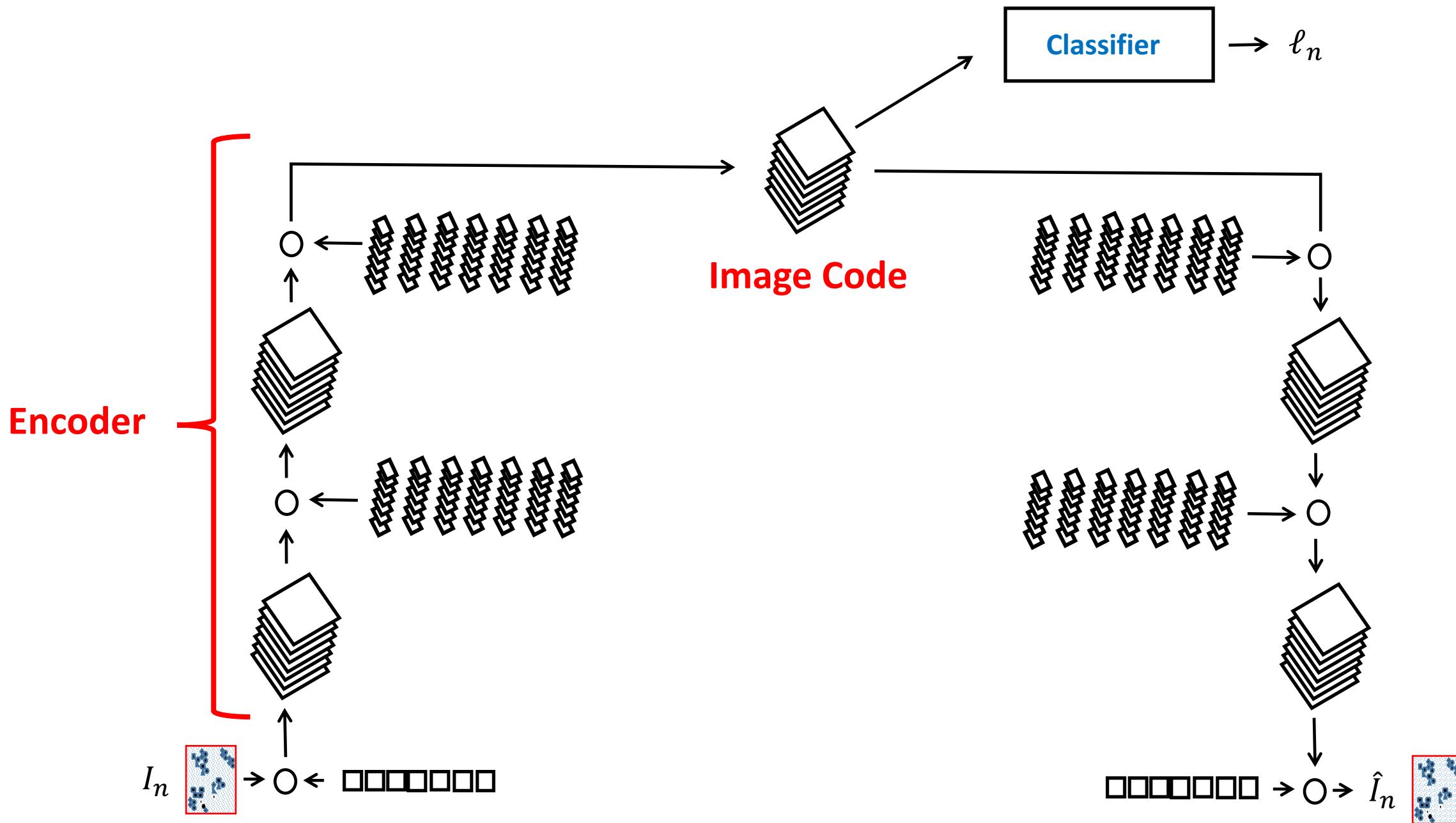
Add Generative Arm



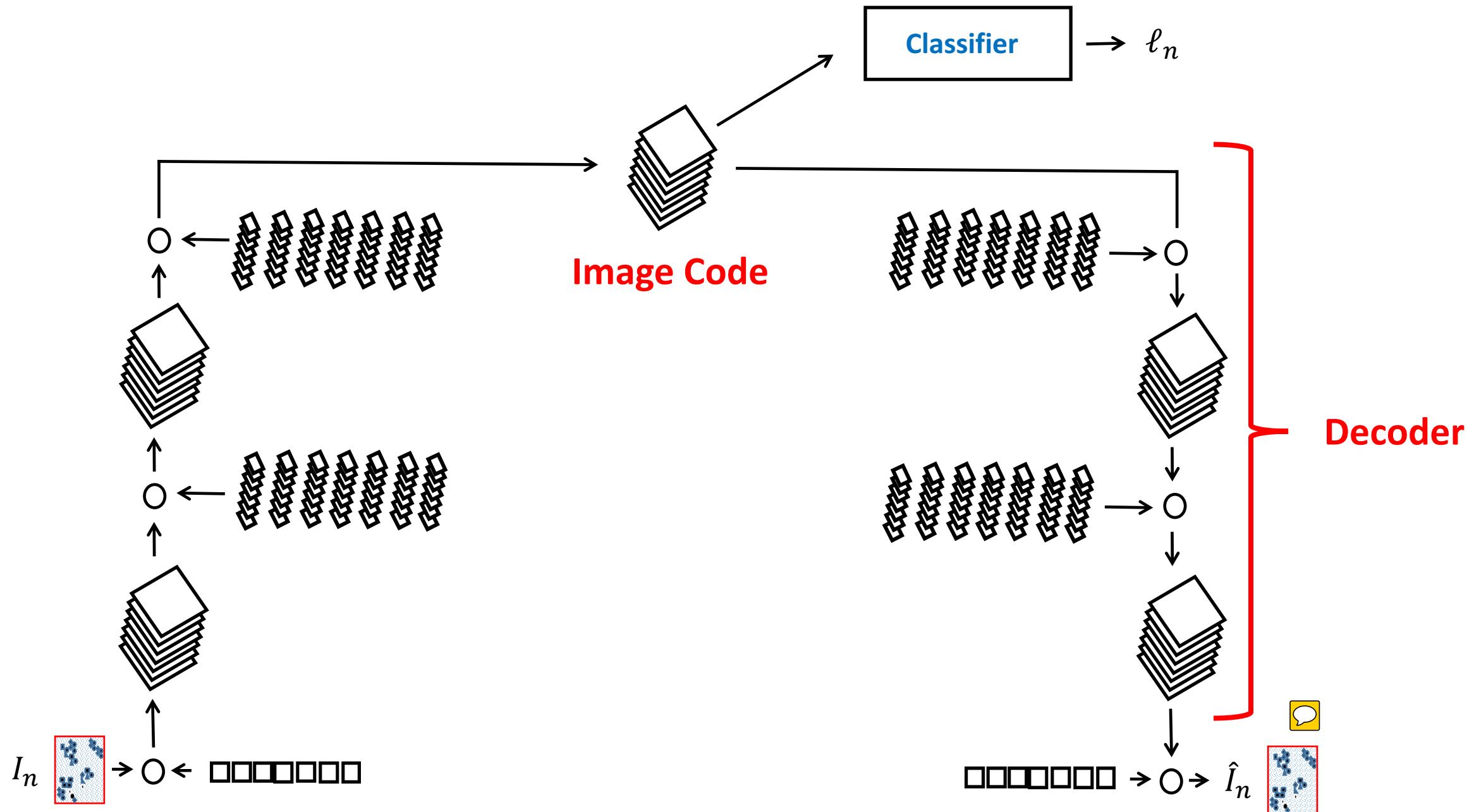
Add Generative Arm



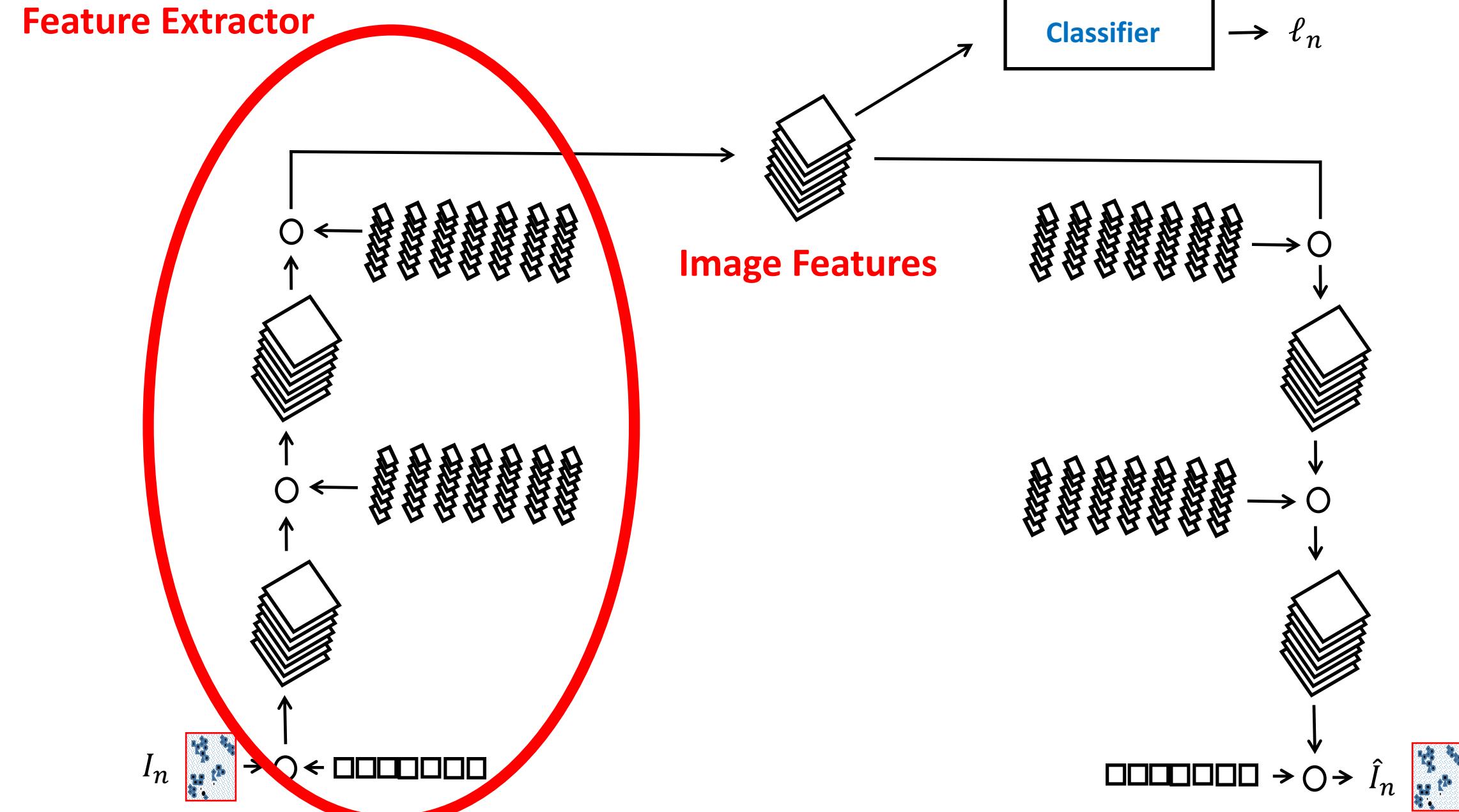
Deep Analysis Architecture



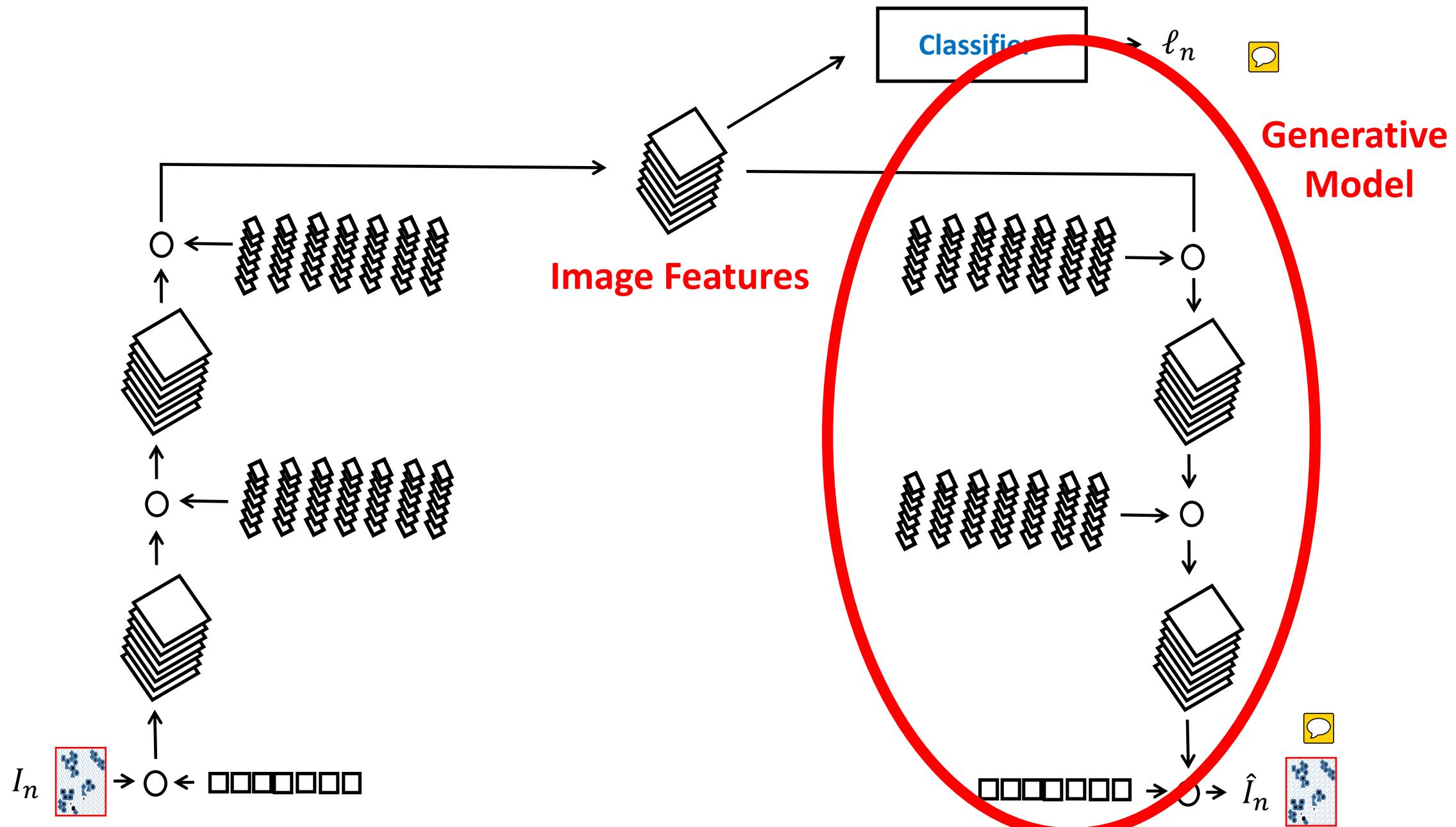
Deep Analysis Architecture



Deep Analysis Architecture



Deep Analysis Architecture



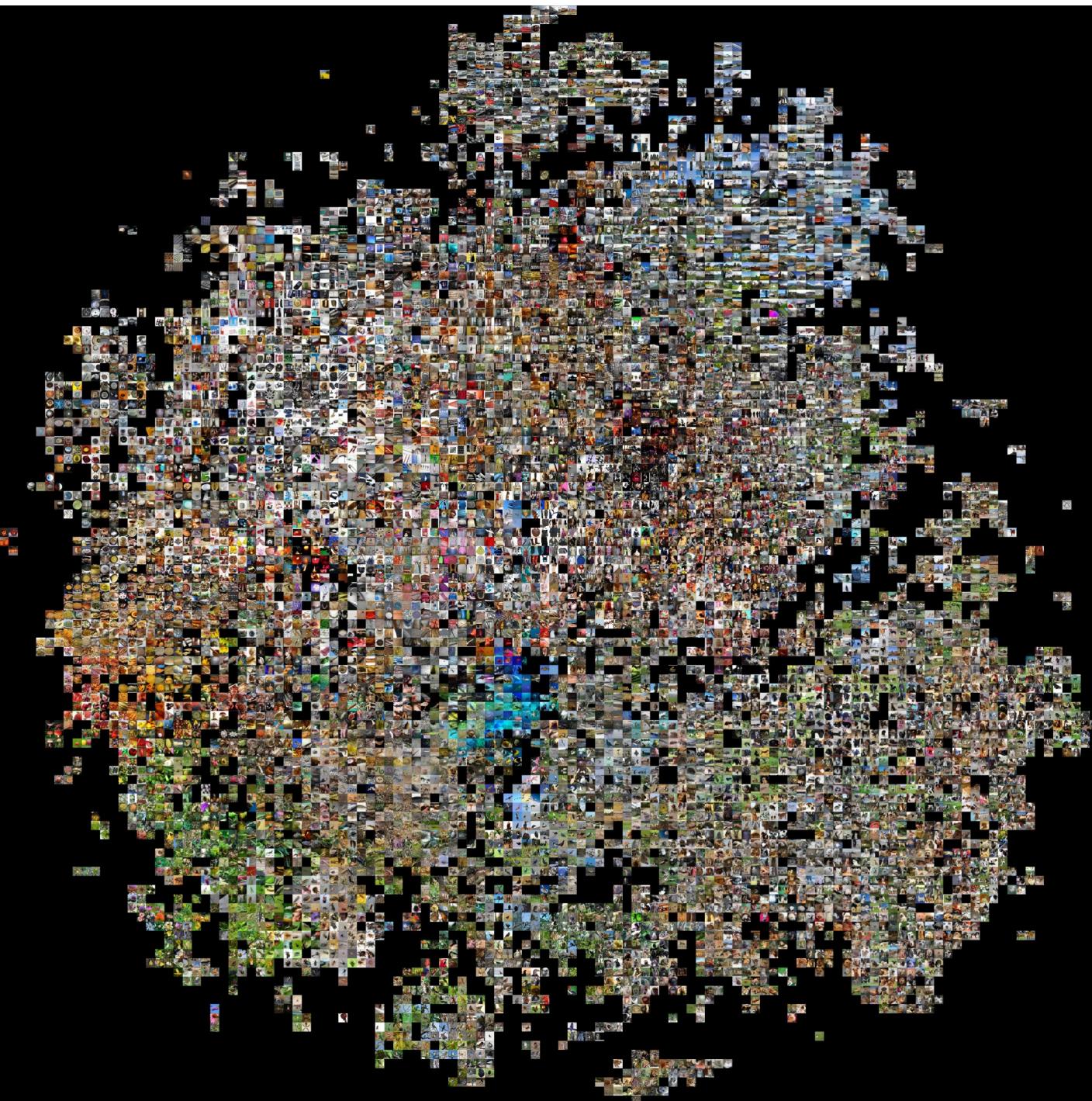
ImageNet Challenge



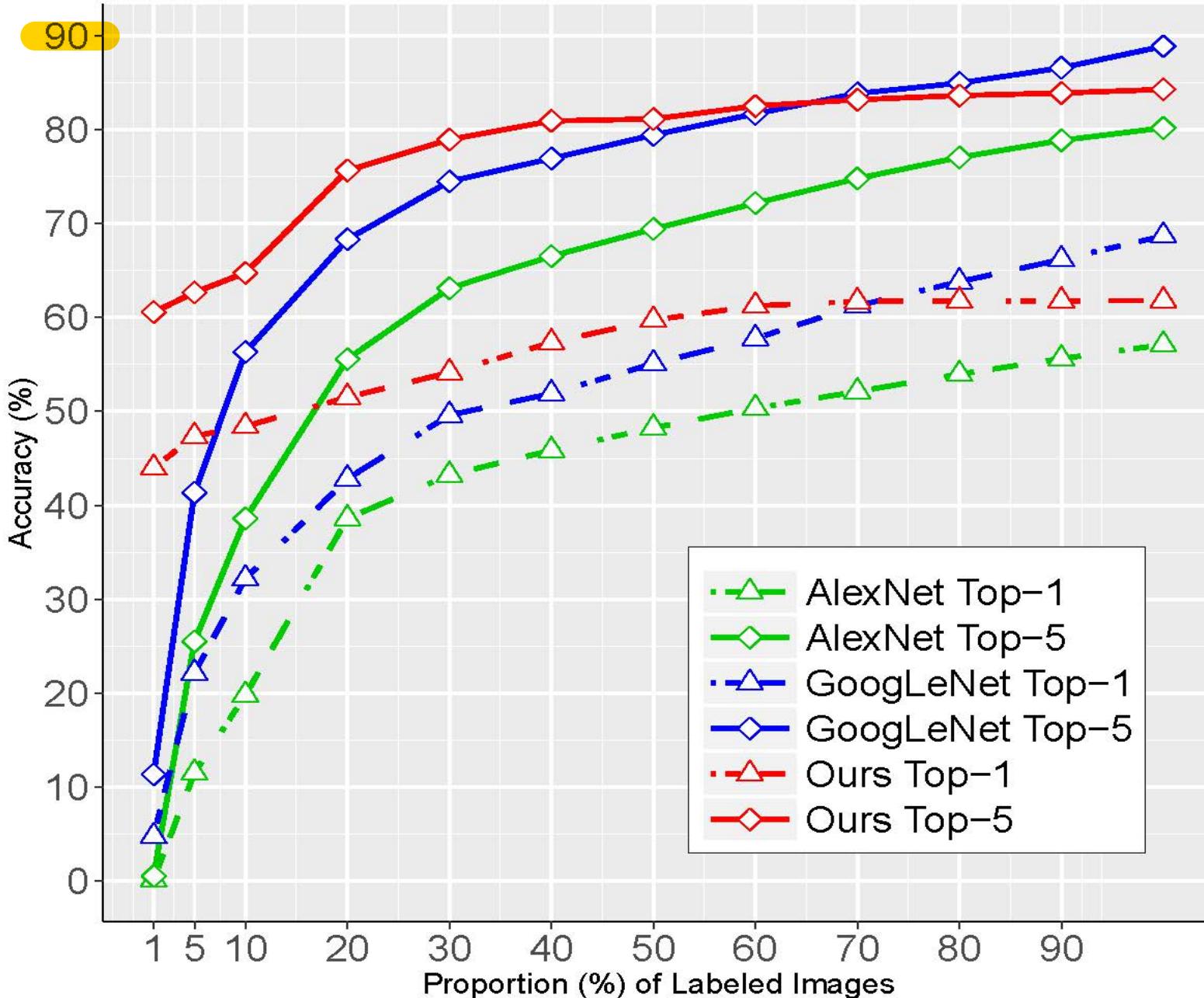
- 1000 image classes
- 1000 training images per class
- 1 million training images
- Real-life RGB images



Example Images



ImageNet Challenge

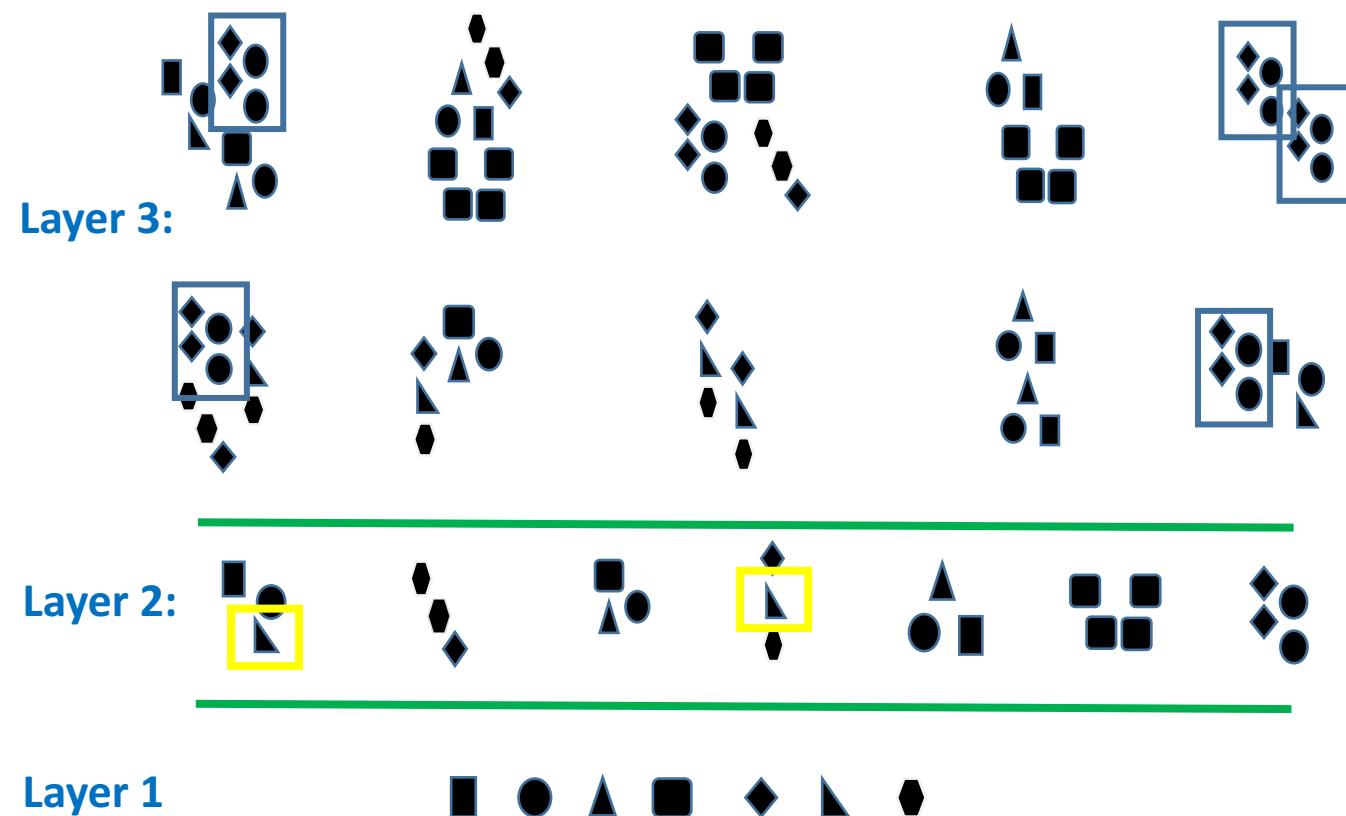




Semi-supervised classification error (%) on MNIST. N is the number of labeled images per class.

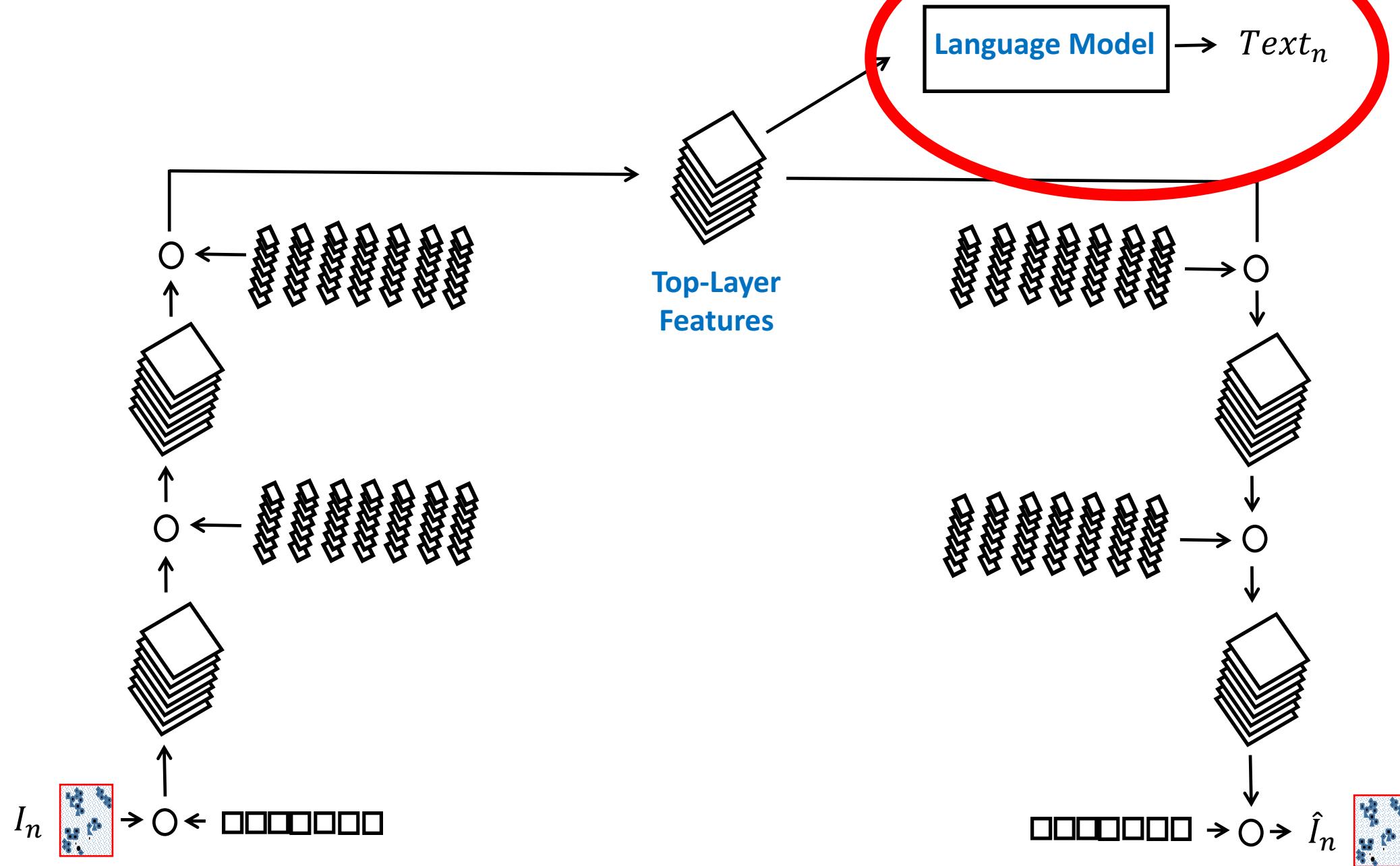
N	TSVM	Deep generative model [27]		Ladder network [28]		Our model	
		MI+TSVM	MI+M2	Γ -full	Γ -conv	$\xi = 0$	$\xi = N_x/(C\rho)$
10	16.81	11.82 \pm 0.25	3.33 \pm 0.14	3.06 \pm 1.44	0.89 \pm 0.50	5.83 \pm 0.97	1.49 \pm 0.36
60	6.16	5.72 \pm 0.05	2.59 \pm 0.05	-	-	2.19 \pm 0.19	0.77 \pm 0.09
100	5.38	4.24 \pm 0.07	2.40 \pm 0.02	1.53 \pm 0.10	-	1.75 \pm 0.14	0.63 \pm 0.06
300	3.45	3.49 \pm 0.04	2.18 \pm 0.04	-	-	1.42 \pm 0.08	0.51 \pm 0.04

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

Deep Analysis Architecture





a man with a snowboard
next to a man with glasses



a big black dog standing on
the grass



a player is holding a
hockey stick



a desk with a keyboard

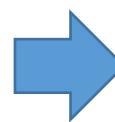
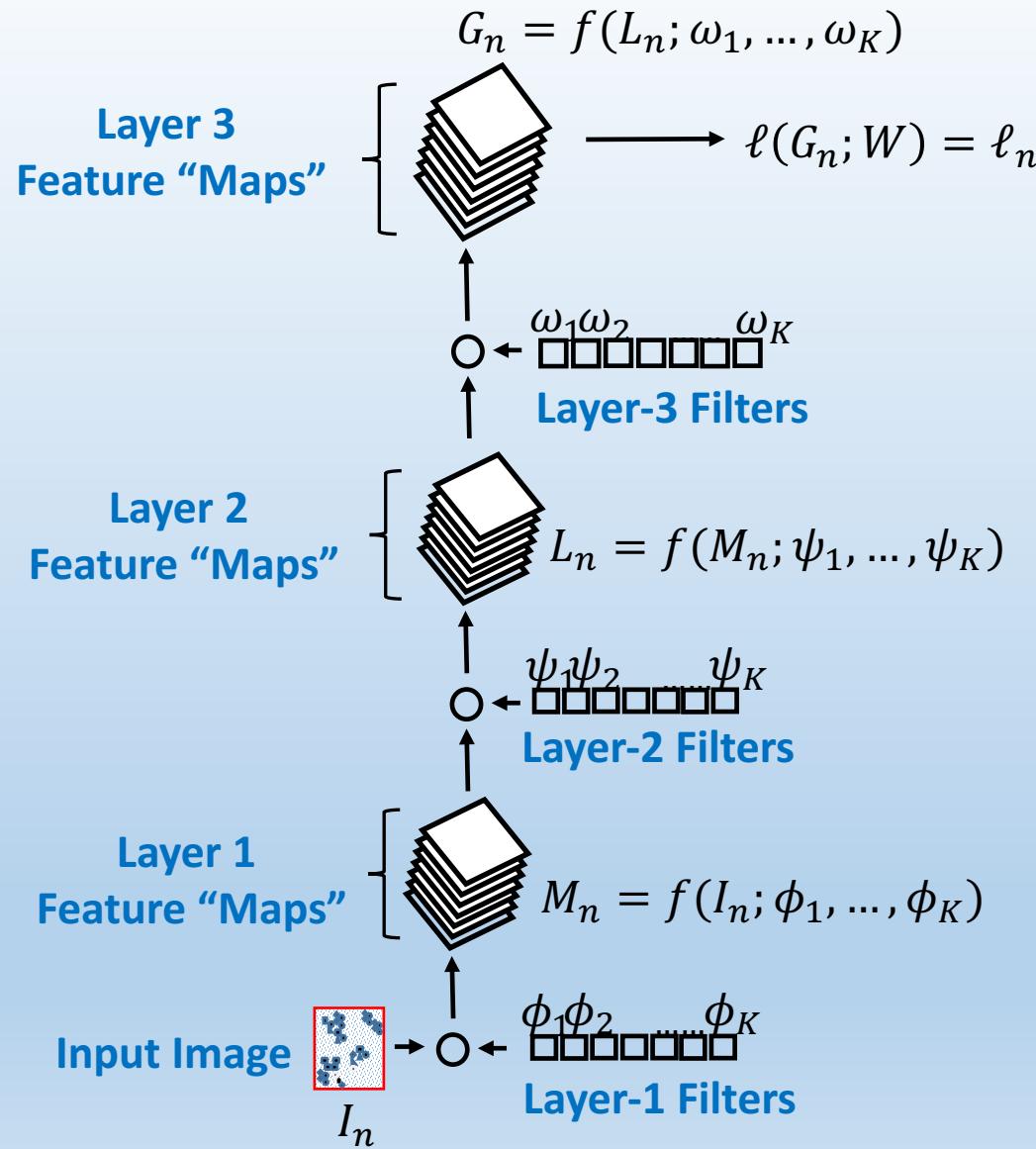


a man is standing next to a
brown horse

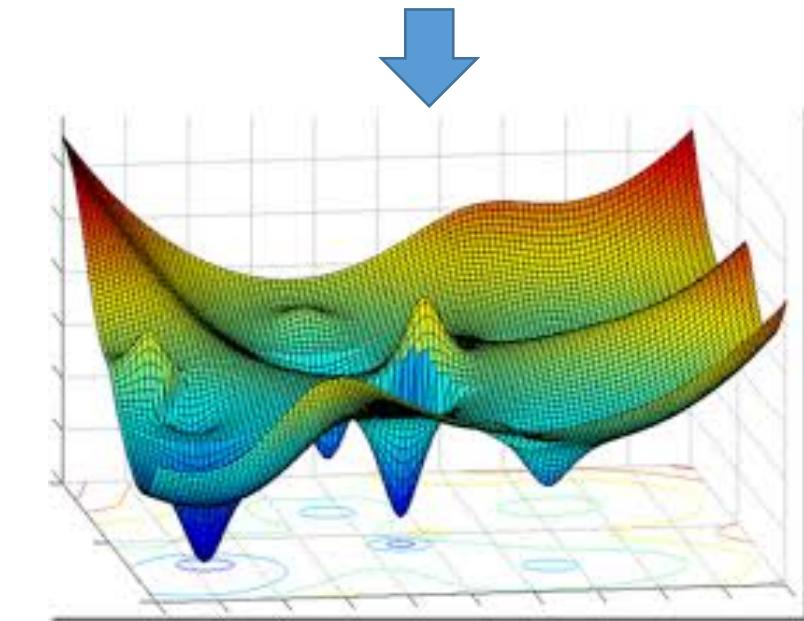


a box full of apples and
oranges

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:
$$E(\Phi, \Psi, \Omega, W) = 1/N \sum_{n=1}^N \text{loss}(y_n, \ell_n)$$
- Find model parameters $\hat{\Phi}, \hat{\Psi}, \hat{\Omega}, \hat{W}$ that minimize $E(\Phi, \Psi, \Omega, W)$

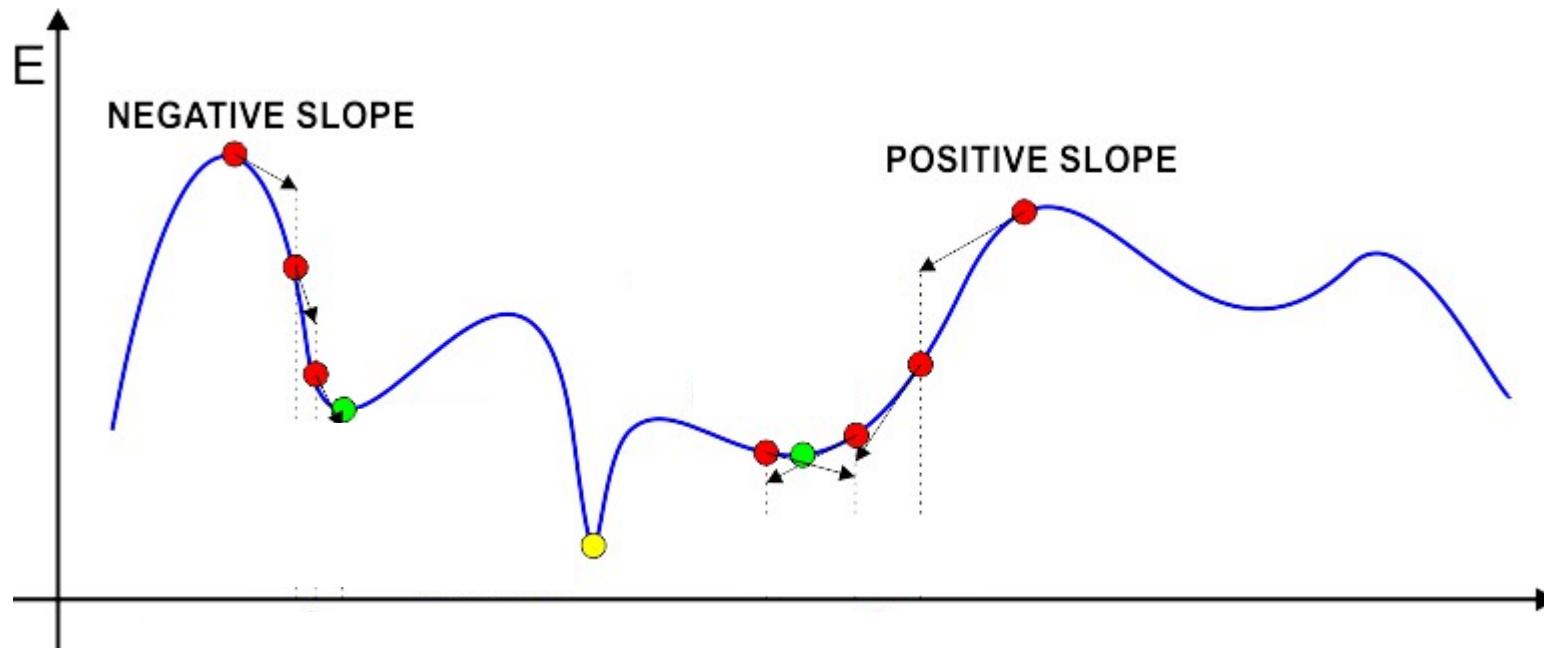


Technical Challenges

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

- For large number of training pairs N , computation of risk $E(\cdot)$ could be prohibitive
- How do we optimize for Φ, Ψ, Ω, W ?

Optimization for $\Theta = \{\Phi, \Psi, \Omega, W\}$?



Gradient Descent

$$\Theta_t = \Theta_{t-1} - \alpha \nabla_{\Theta} E(\Theta_t)$$



Multi-dimensional
“slope”

Massive N ?

Gradient Descent

$$\Theta_t = \Theta_{t-1} - \alpha \nabla_{\Theta} E(\Theta_t)$$



Stochastic Gradient Descent

$$\Theta_t = \Theta_{t-1} - \alpha \nabla_{\Theta} \hat{E}_t(\Theta_t)$$

$$\hat{E}_t(\Phi, \Psi, \Omega, W) = 1/|S_t| \sum_{n \in S_t} loss(y_n, \ell_n)$$

S_t a *random* subset of data, at iteration t



a cow is standing in front of a store



a group of elephants standing next to each other



a table that has wooden pieces on it



a cat is eating some kind of food

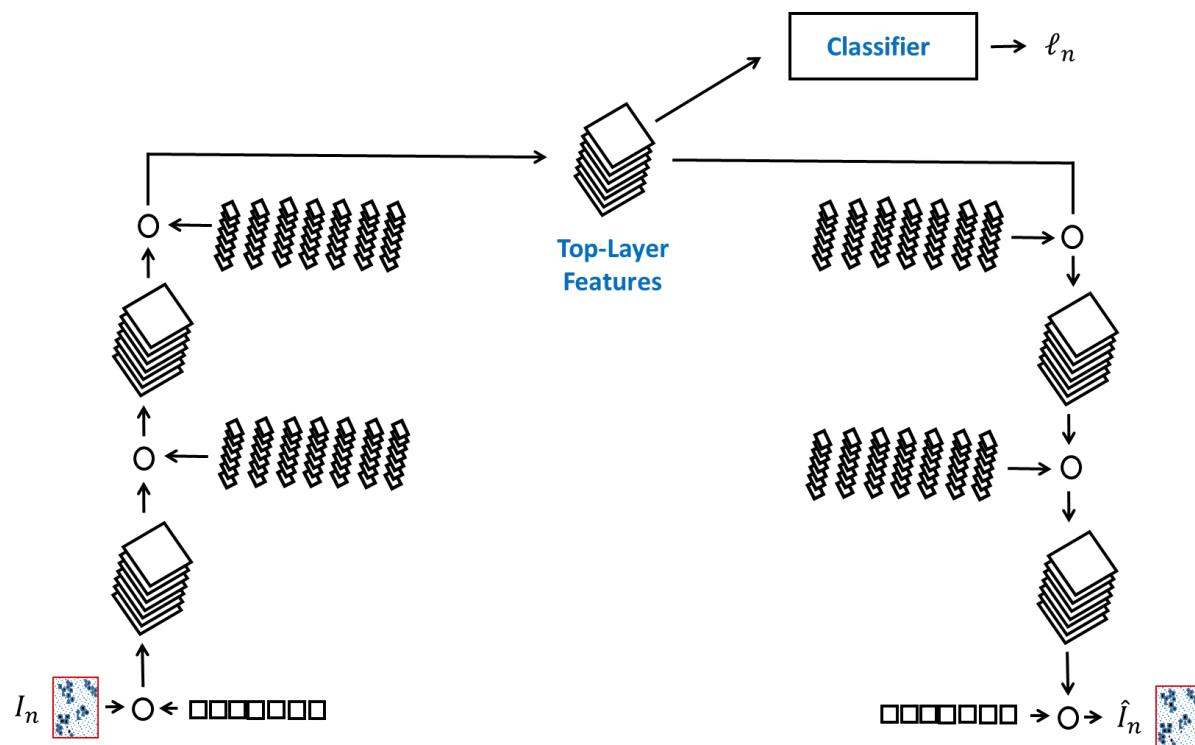


a bunch of bananas are sitting on a table



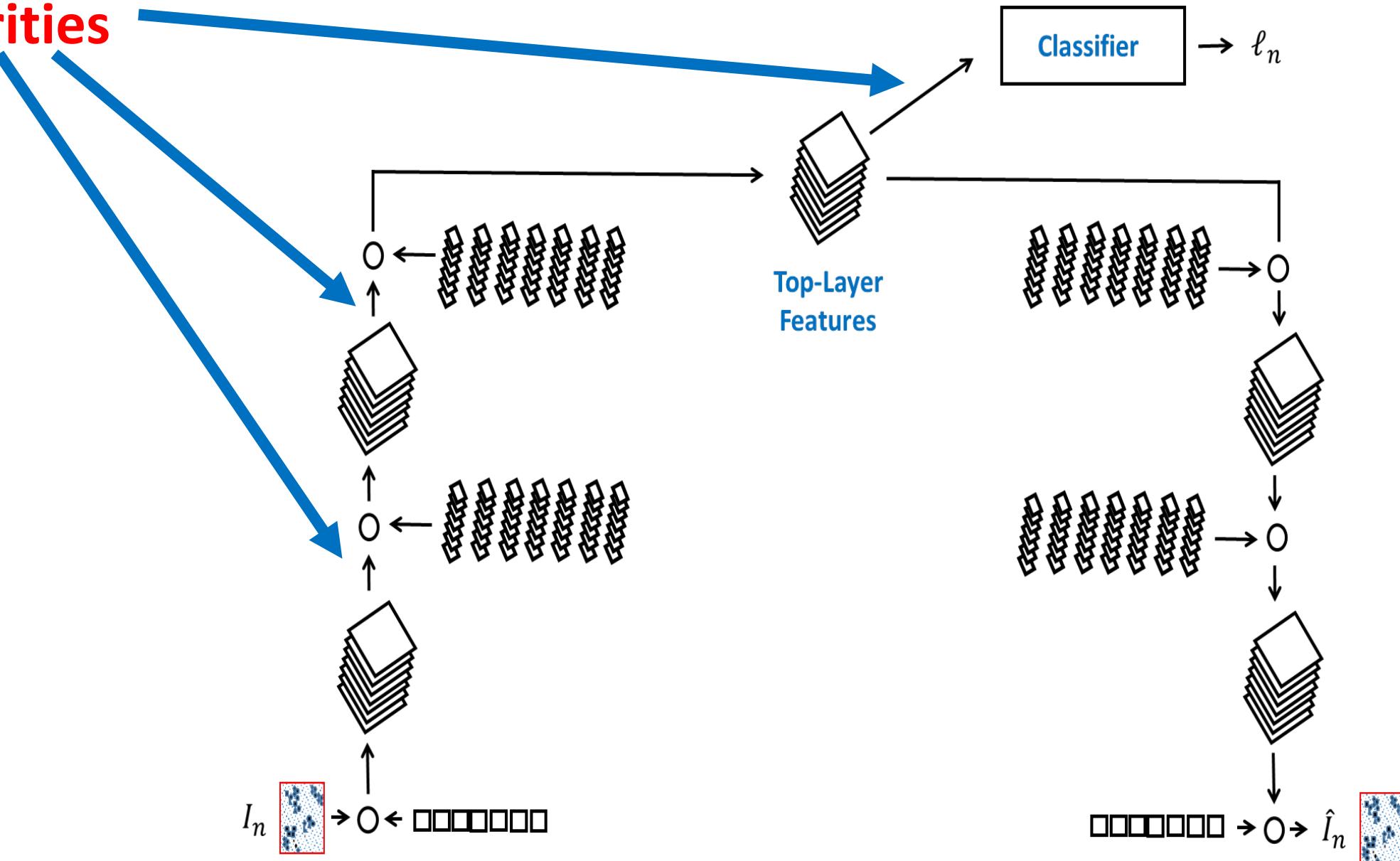
a motorcycle is parked next to a window

Important Technical Detail: Need for Nonlinearities



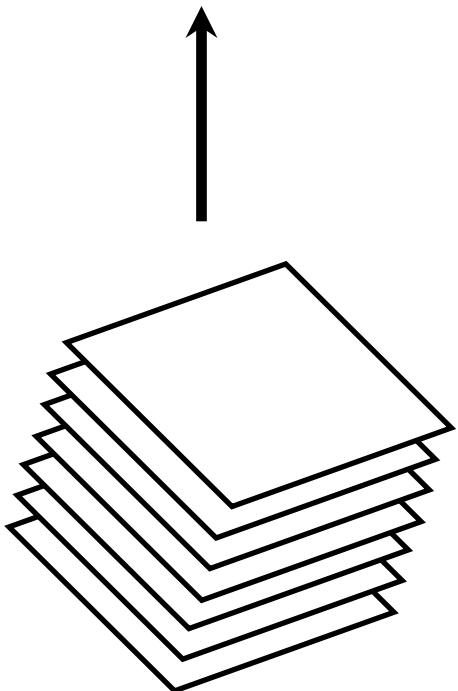
- As expressed, the model is linear
- If we consider image $I = I_n + I_m$ the top-layer features are the sum of the features from I_n and I_m separately
- Similarly, consider top-layer features f_n and f_m , that generate images I_n and I_m
- Top-layer features $f_n + f_m$ generate image $I_n + I_m$
- Such a linear model is overly restrictive

Add Nonlinearities



Pointwise Nonlinearity

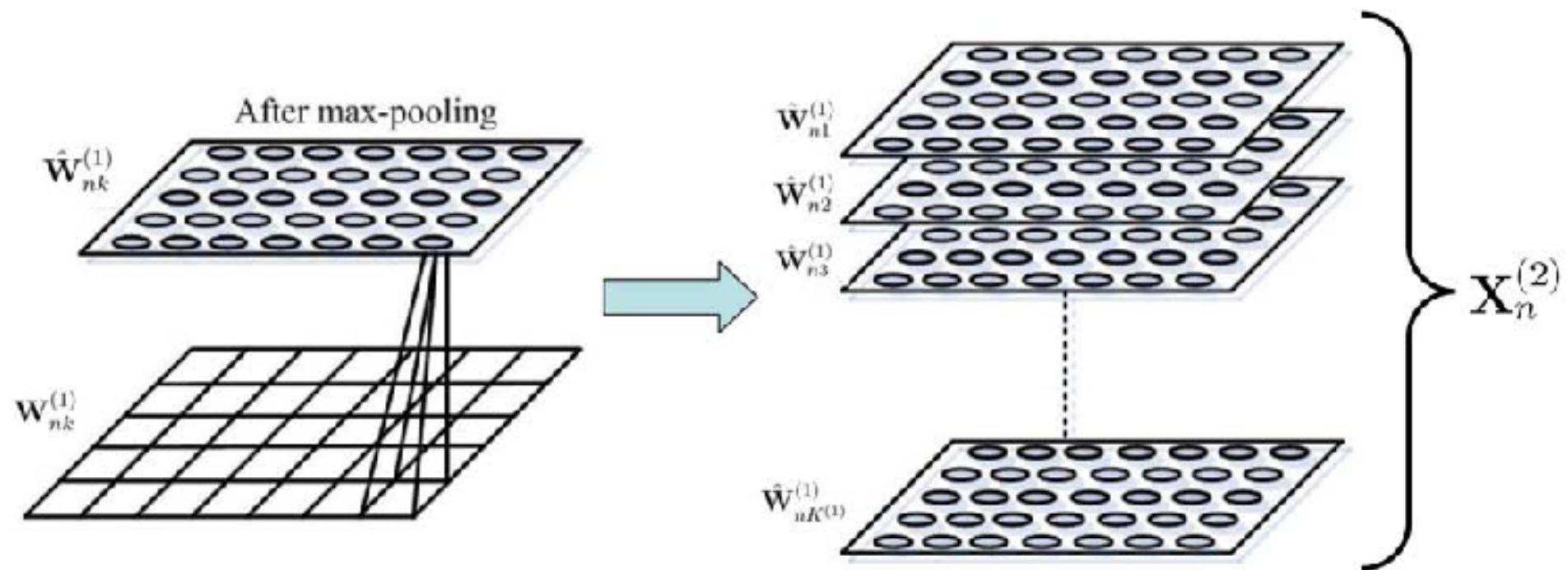
Pointwise Nonlinearity



Convolutional filtering

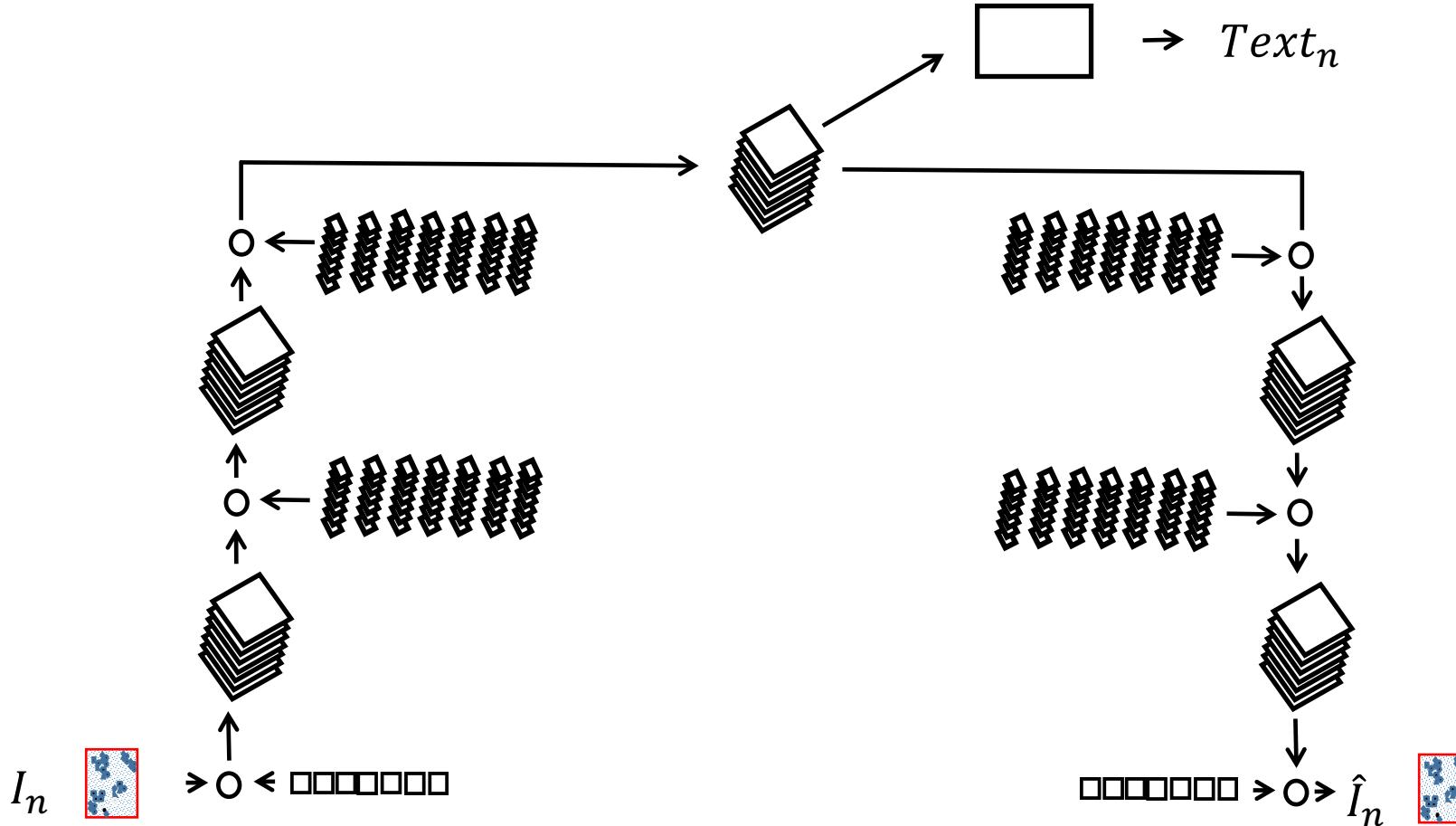
- The feature map after the convolutional filters is sent through a pointwise nonlinear function
- Common example: $f(x) = \tanh(x)$
- Recent interest in $f(x) = 0 \text{ if } x \leq 0, f(x) = x \text{ otherwise}$

Pooling of Activation Maps



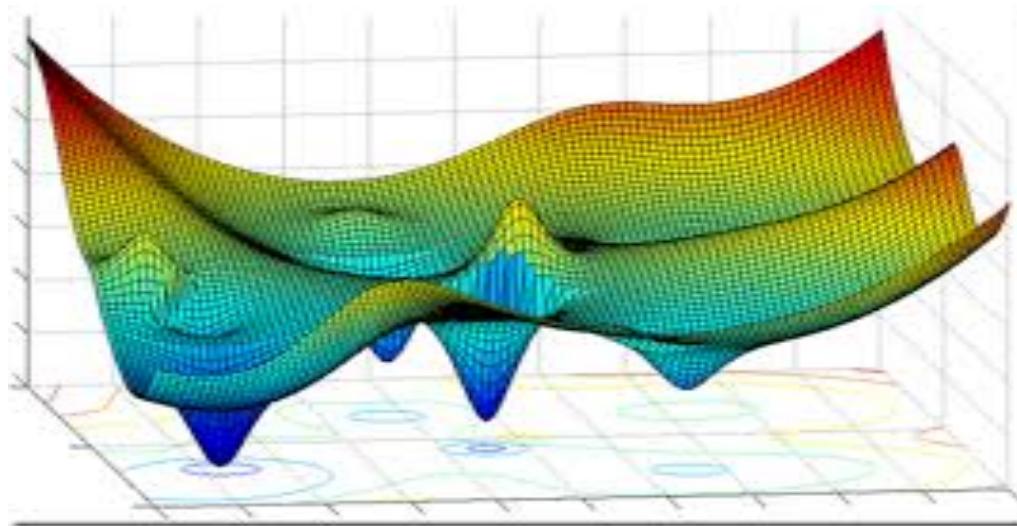


Summary



1. Deep architectures, data-driven feature learning

Summary



Stochastic Gradient Descent

$$\Theta_t = \Theta_{t-1} - \alpha \nabla_{\Theta} \hat{E}_t(\Theta_t)$$

2. Scale to “Big Data”

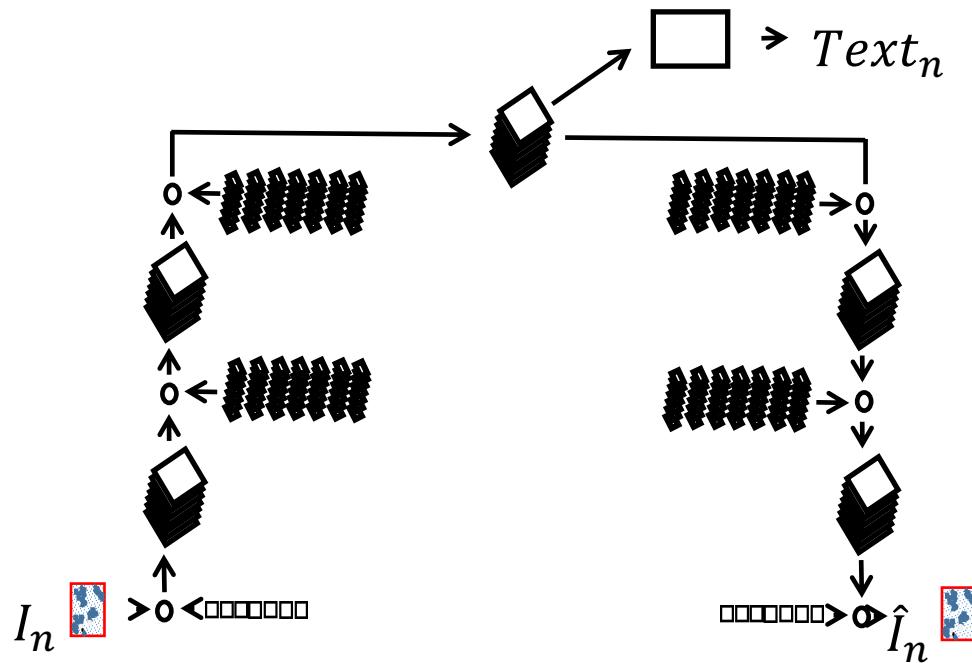
Summary



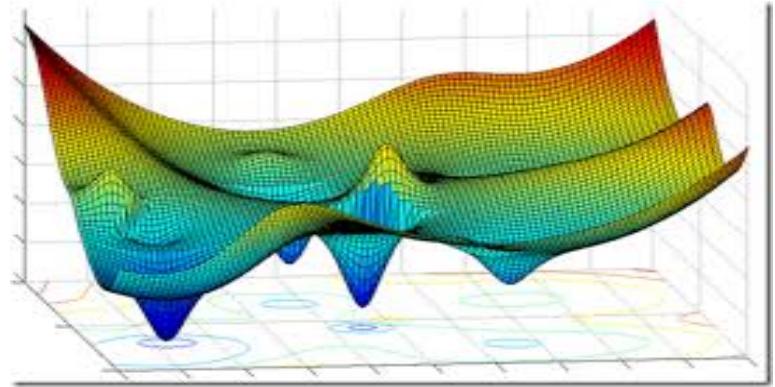
a man is standing next to a brown horse

3. Intriguing results on complex, large-scale data

Summary



1. Deep architectures, feature learning



Stochastic Gradient Descent

$$\Theta_t = \Theta_{t-1} - \alpha \nabla_{\Theta} \hat{E}_t(\Theta_t)$$

2. Scale to “Big Data”



a man is standing next to a brown horse

3. Intriguing results