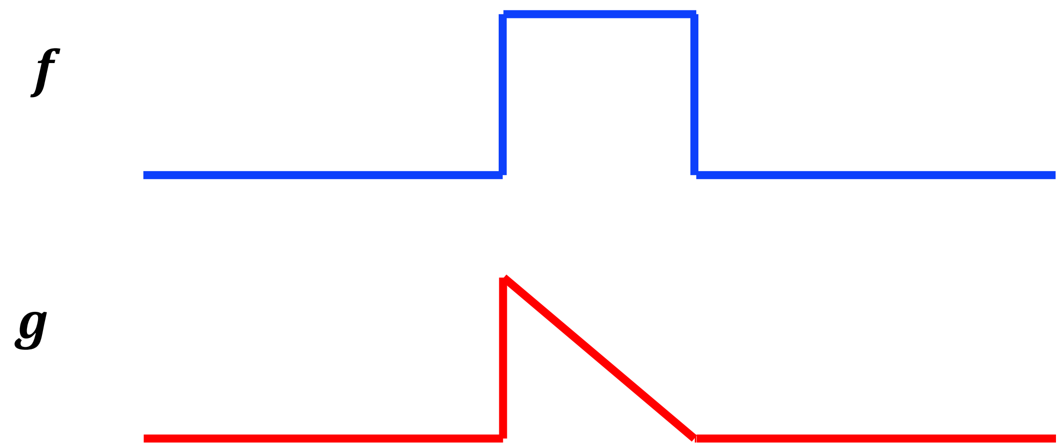
Deep Convolutional Neural Networks and Applications in Medical Imaging

AI in Medicine

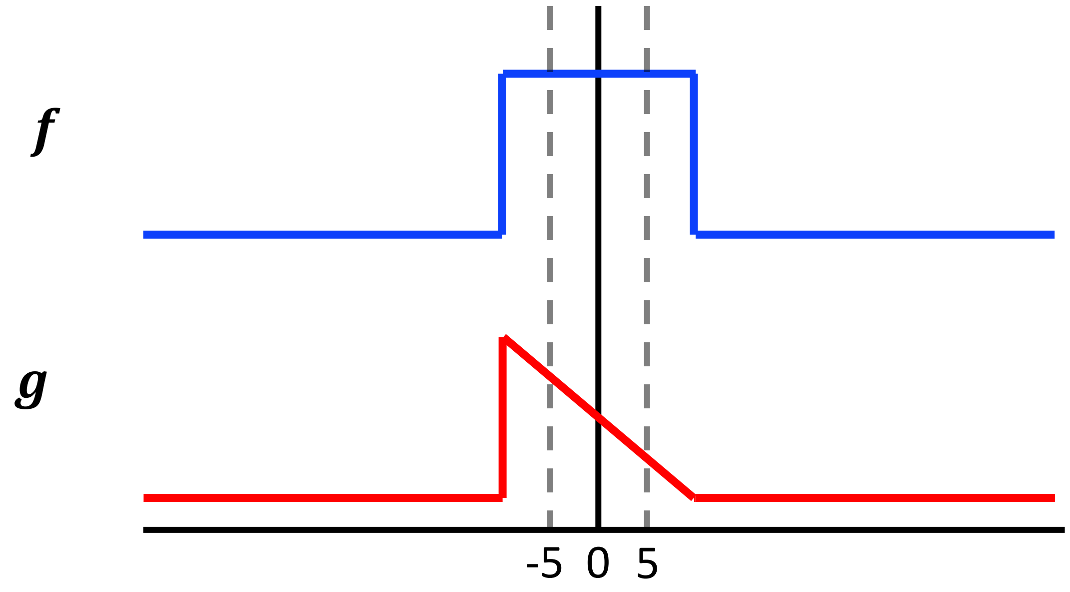
Ricardo Henao

Intro to the Convolution (1D)



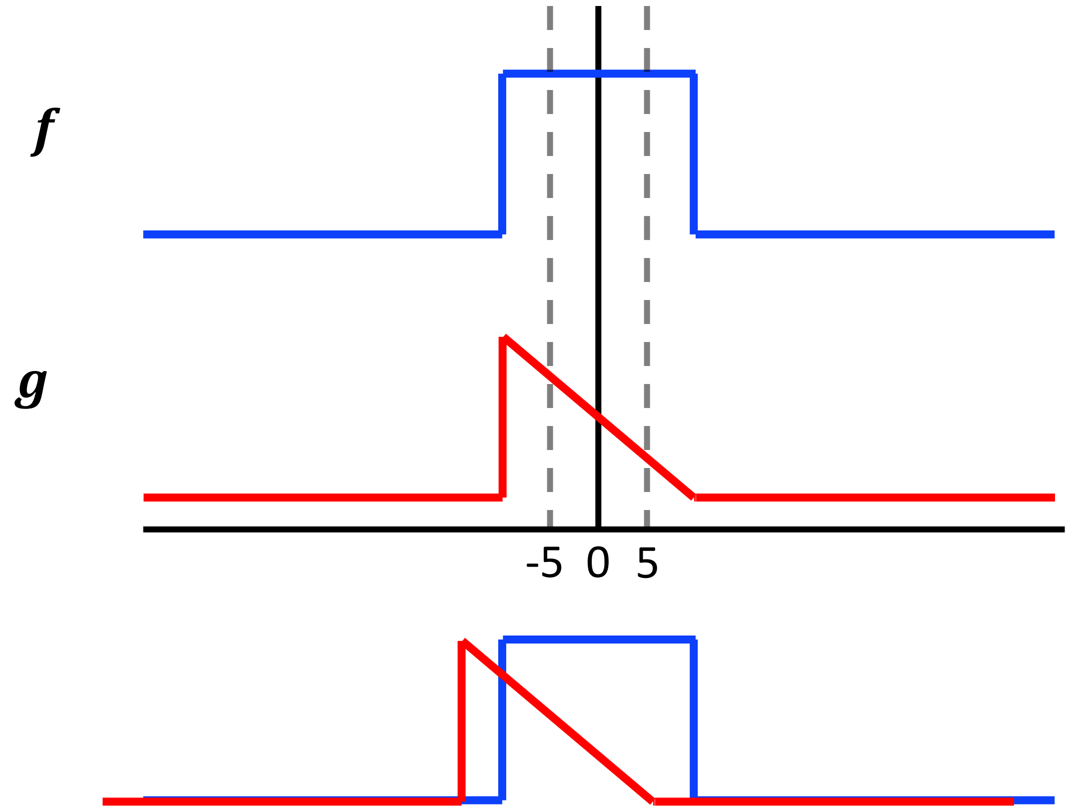
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Intro to the Convolution (1D)



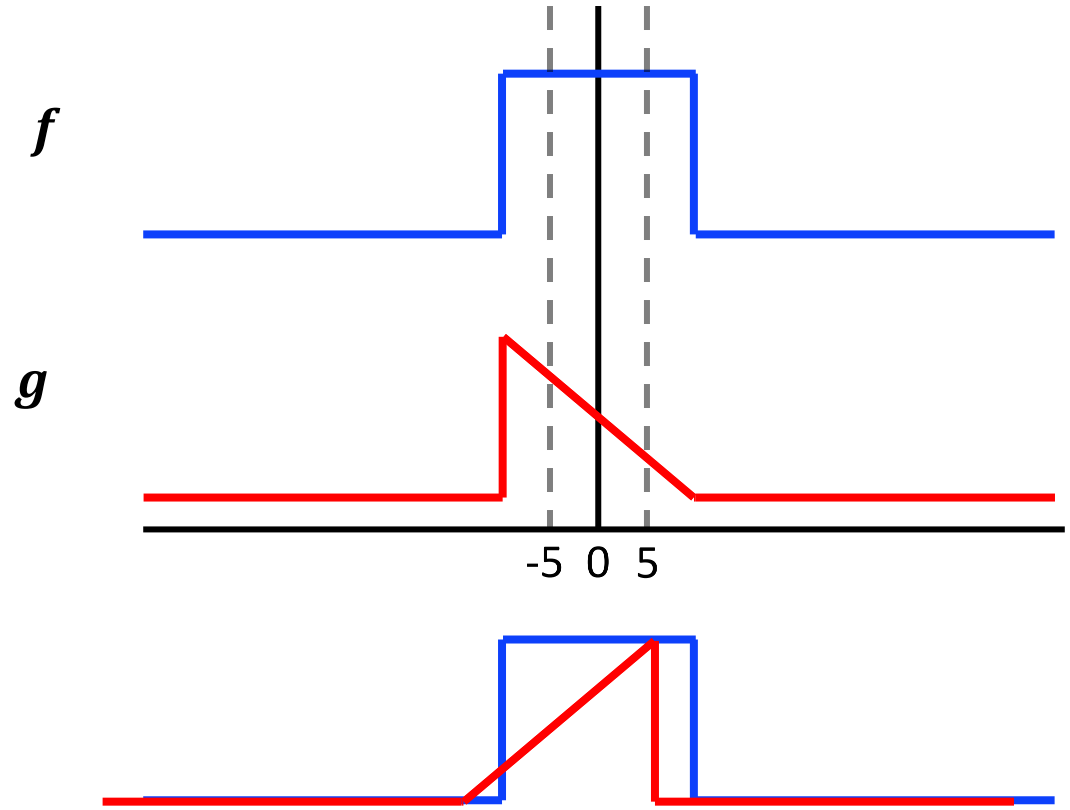
$$(f * g)[n] = \sum_{m=-\infty}^{\infty} f[m]g[n-m]$$

Intro to the Convolution (1D)



$$(f * g)[-5] = \sum_{m=-\infty}^{\infty} f[m]g[-(5+m)]$$

Intro to the Convolution (1D)



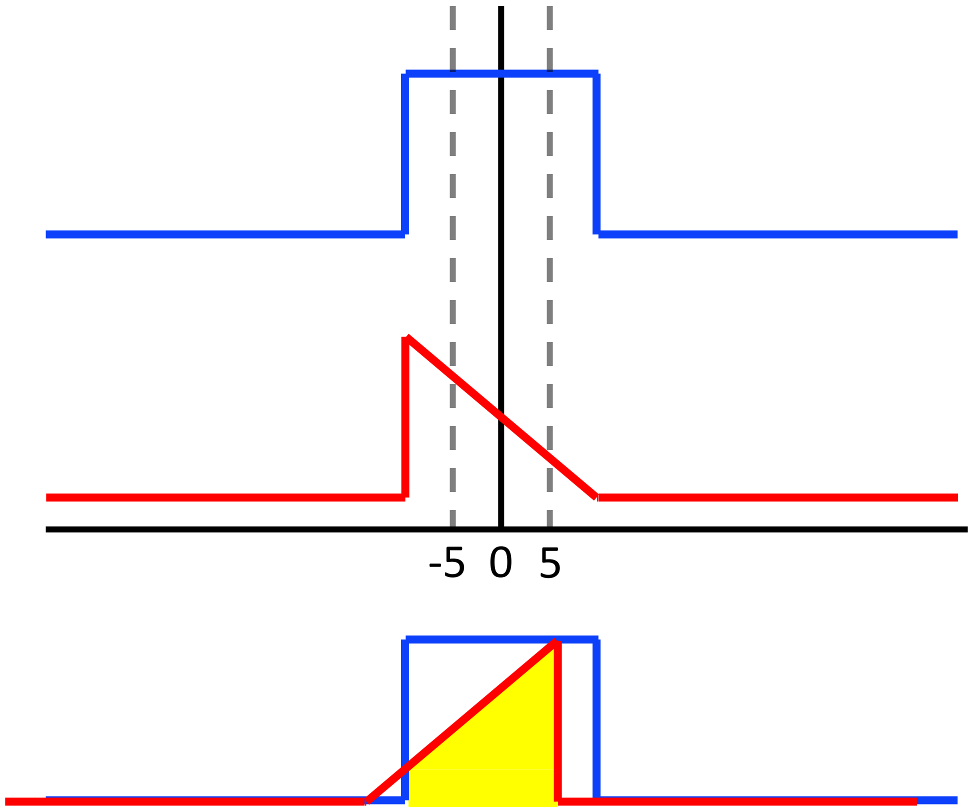
$$(f * g)[-5] = \sum_{m=-\infty}^{\infty} f[m]g[-(5+m)]$$

Intro to the Convolution (1D)

$\boldsymbol{f}$

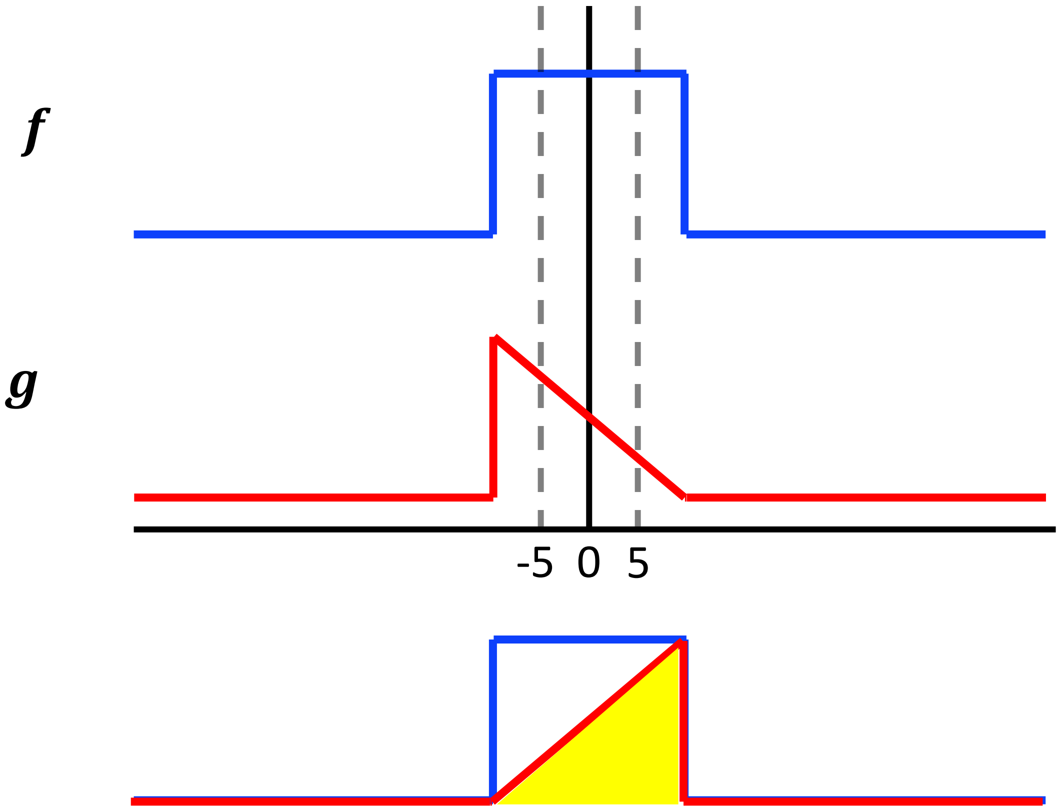
$\boldsymbol{g}$

$(f * g)[-5]$



$$(f * g)[-5] = \sum_{m=-\infty}^{\infty} f[m]g[-(5+m)] = 15$$

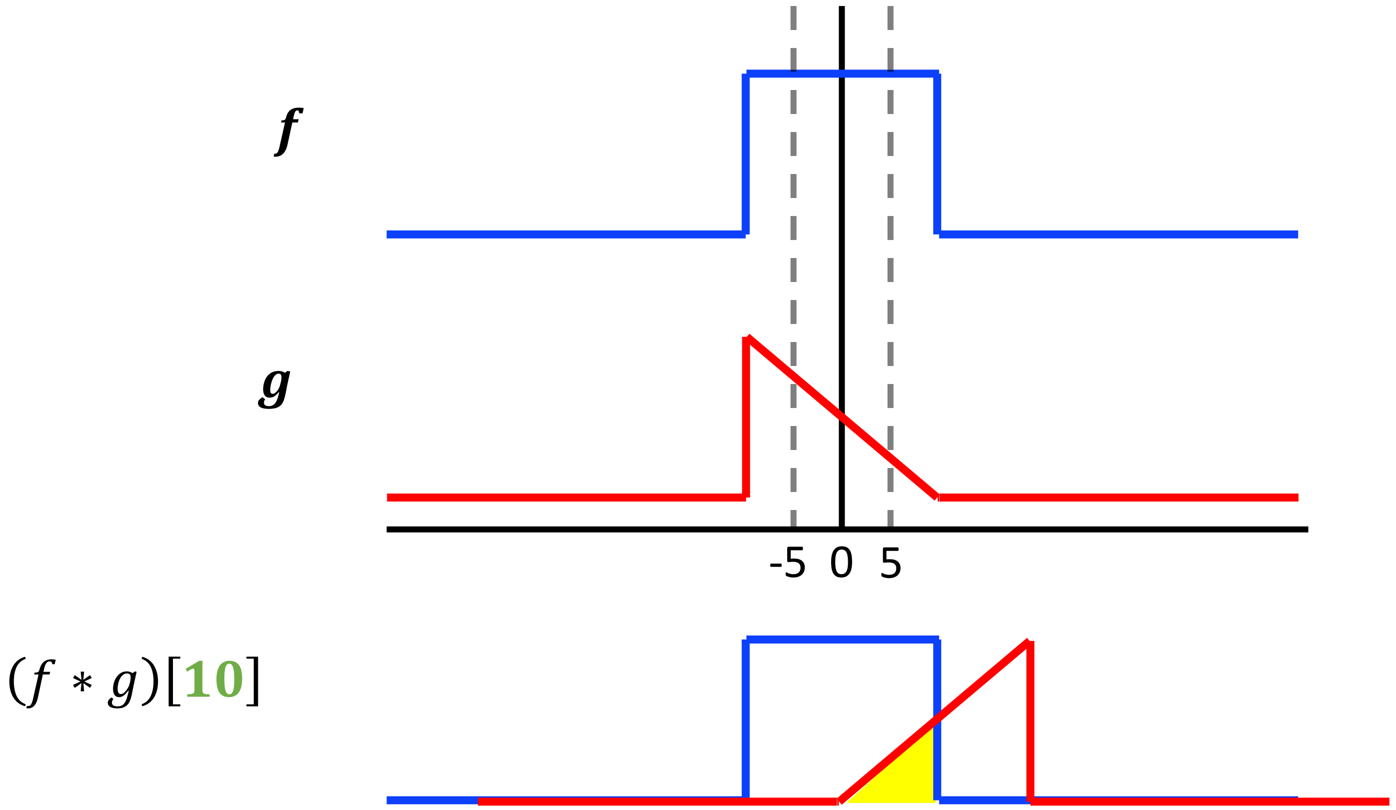
Intro to the Convolution (1D)



$(f * g)[\mathbf{0}]$

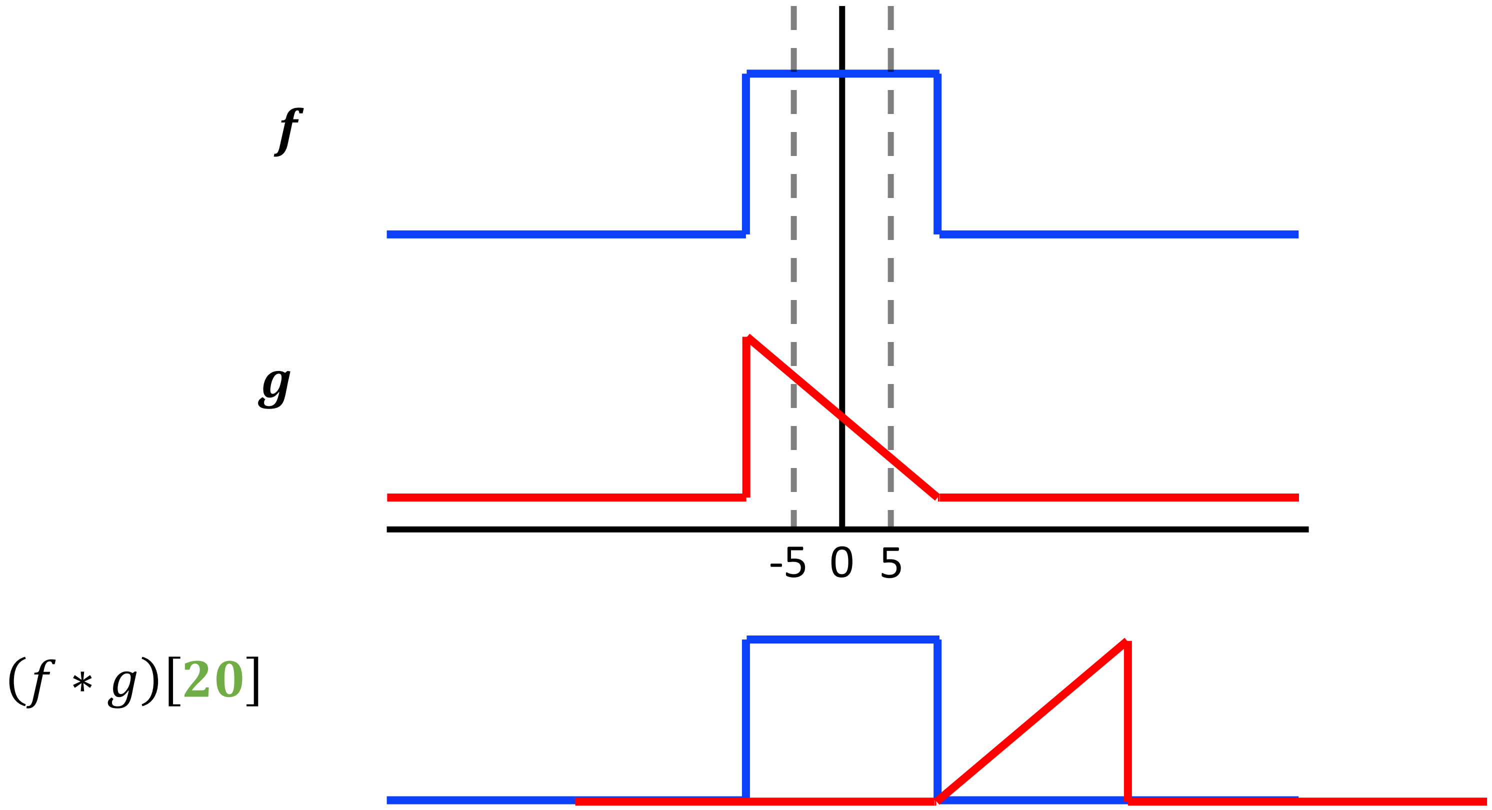
$$(f * g)[\mathbf{0}] = \sum_{m=-\infty}^{\infty} f[m]g[-(m)] = 20$$

Intro to the Convolution (1D)



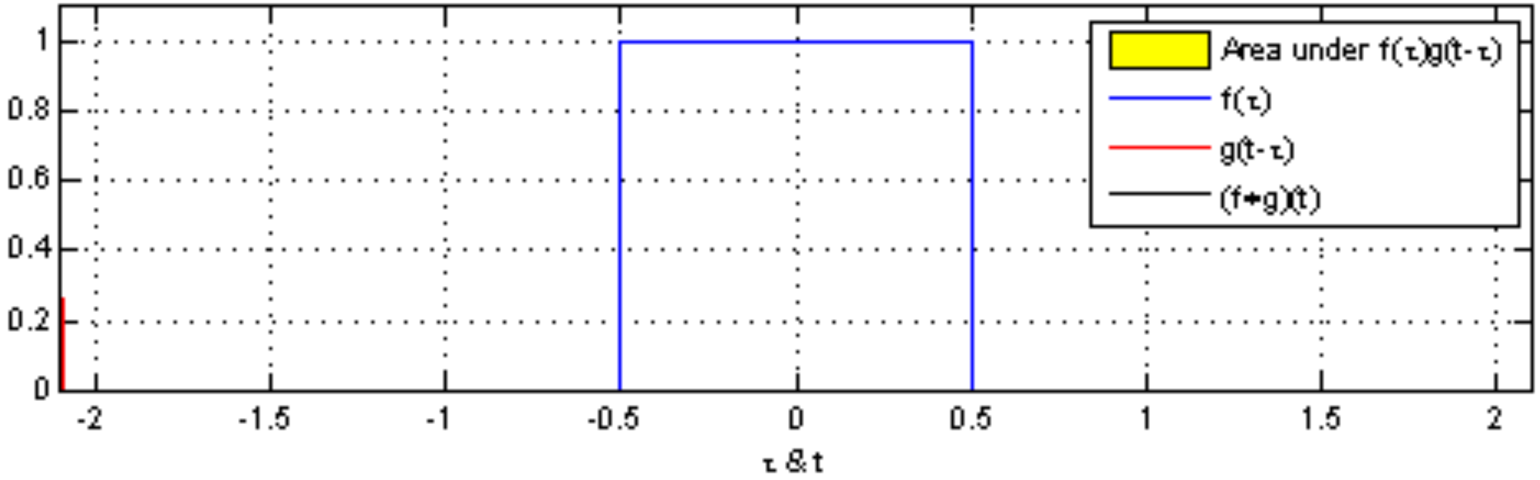
$$(f * g)[\mathbf{10}] = \sum_{m=-\infty}^{\infty} f[m]g[-(\mathbf{-10} + m)] = 10$$

Intro to the Convolution (1D)



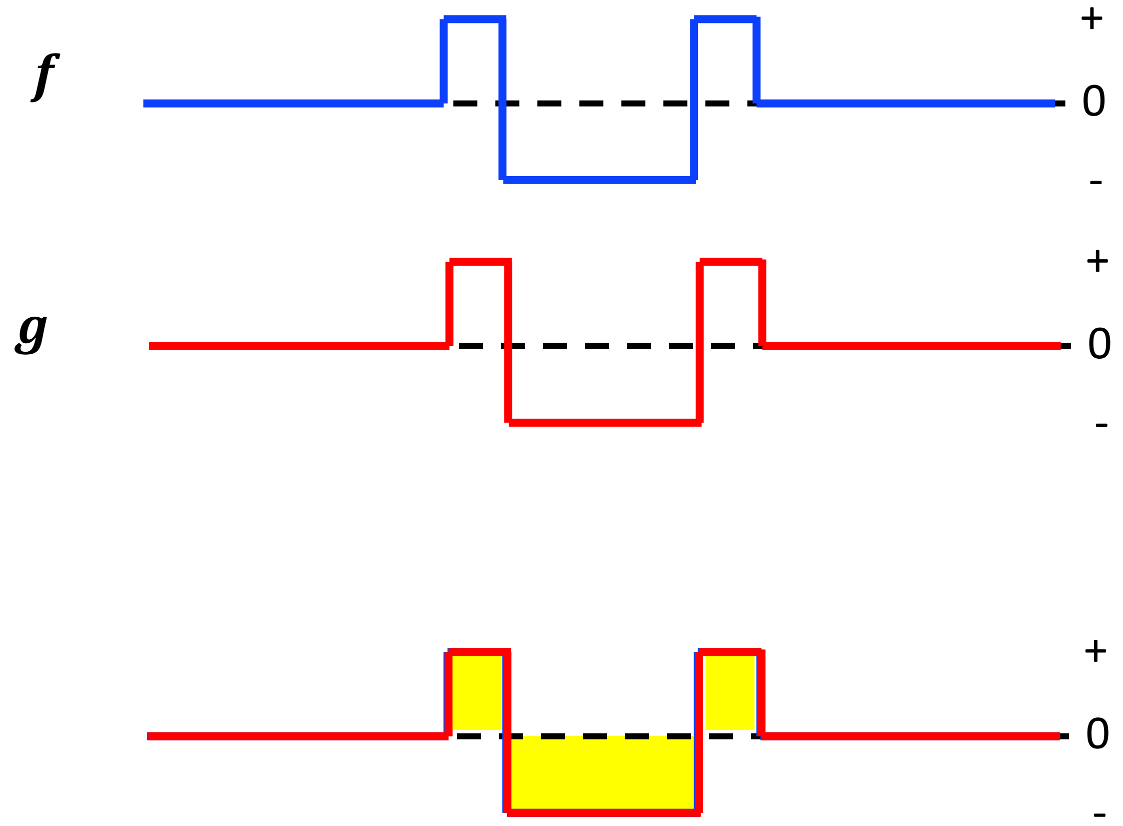
$$(f * g)[20] = \sum_{m=-\infty}^{\infty} f[m]g[-(-20 + m)] = 0$$

Intro to the Convolution (1D)

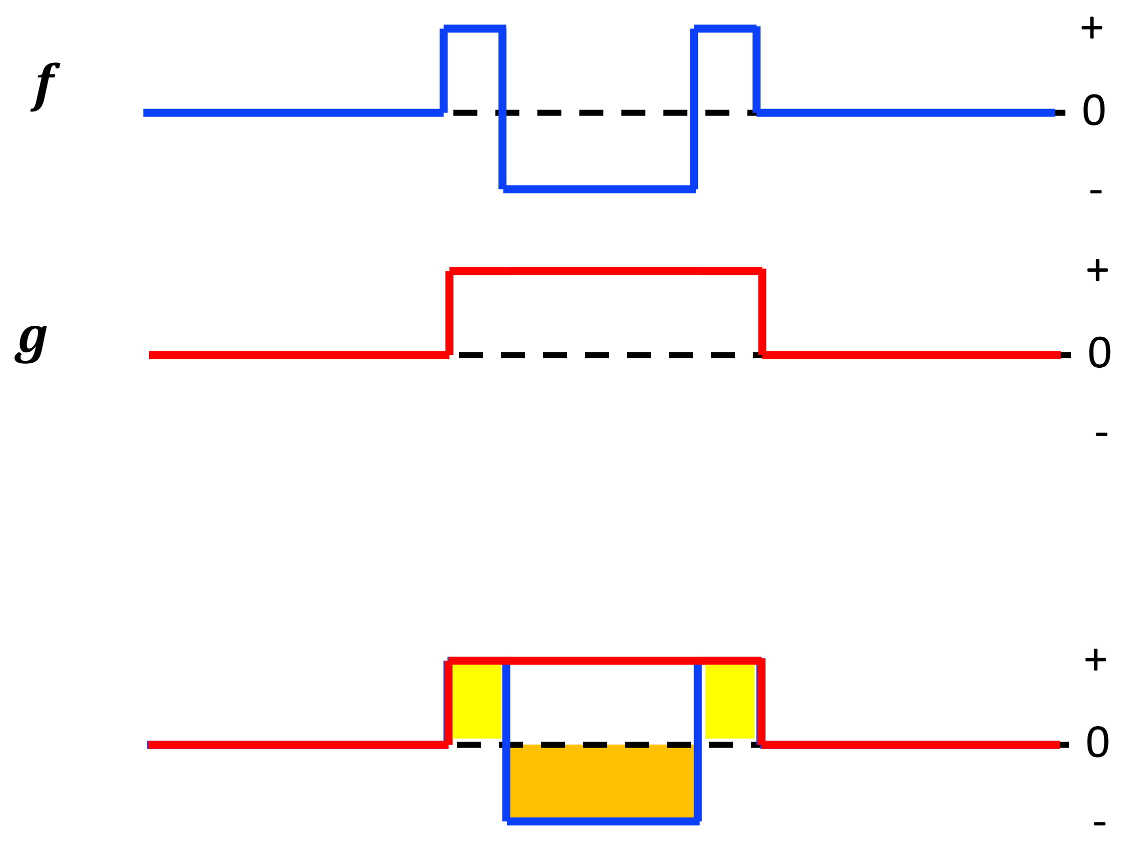


Wikipedia

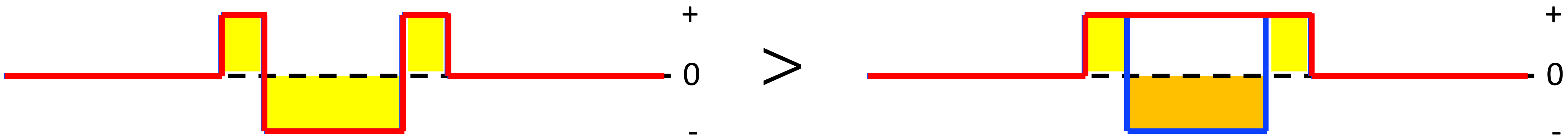
Convolutional Filters Are Feature Detectors



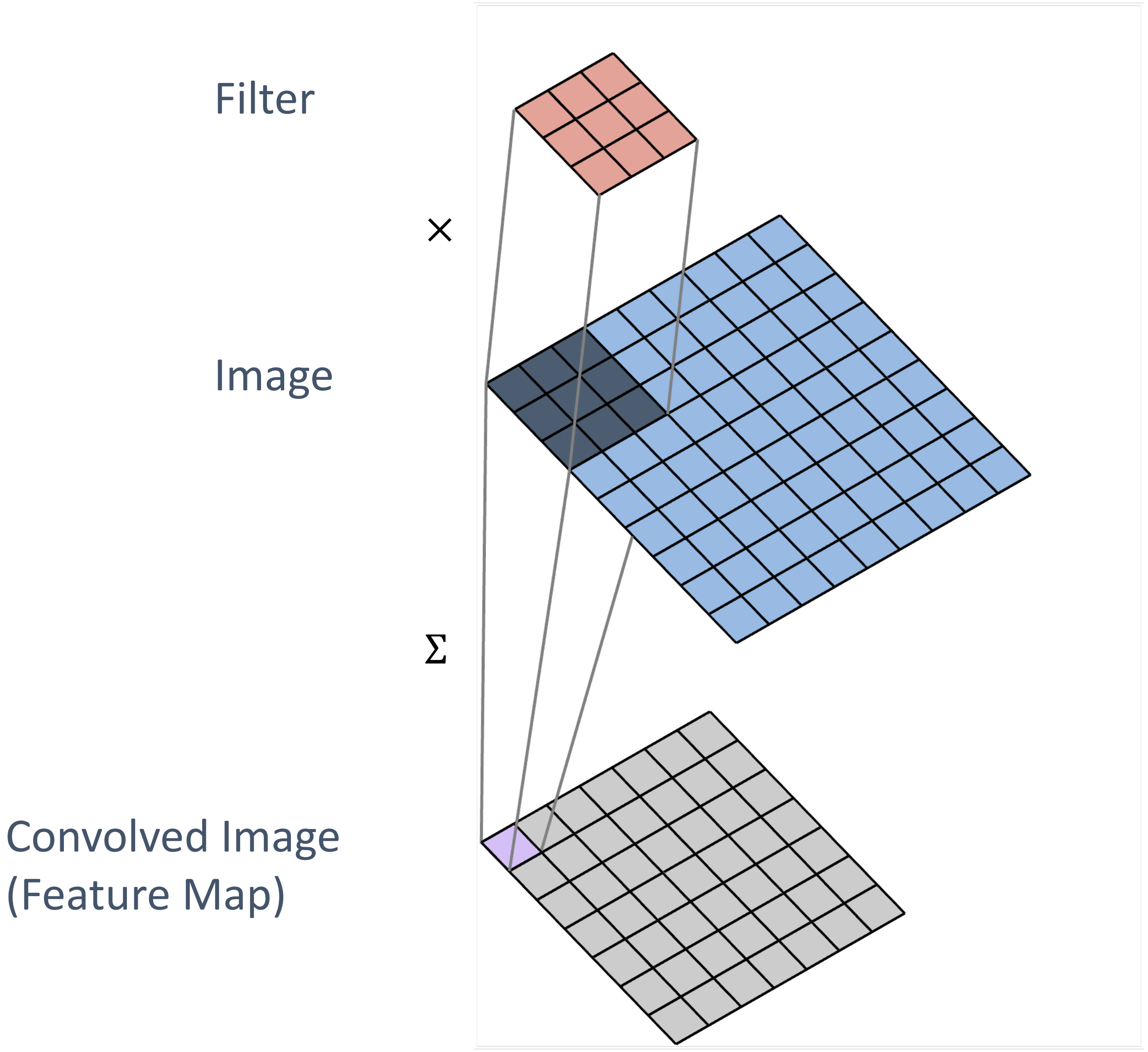
Convolutional Filters Are Feature Detectors



Convolutional Filters Are Feature Detectors



2D Spatial Convolution

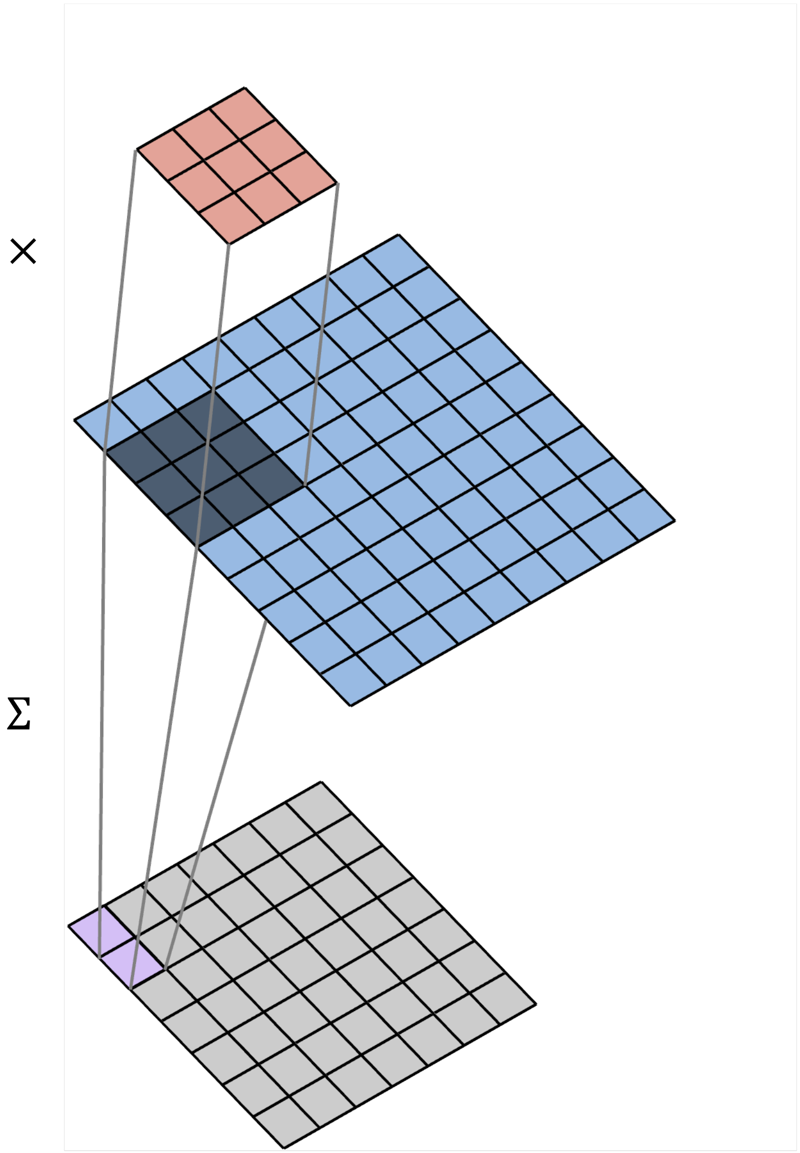


2D Spatial Convolution

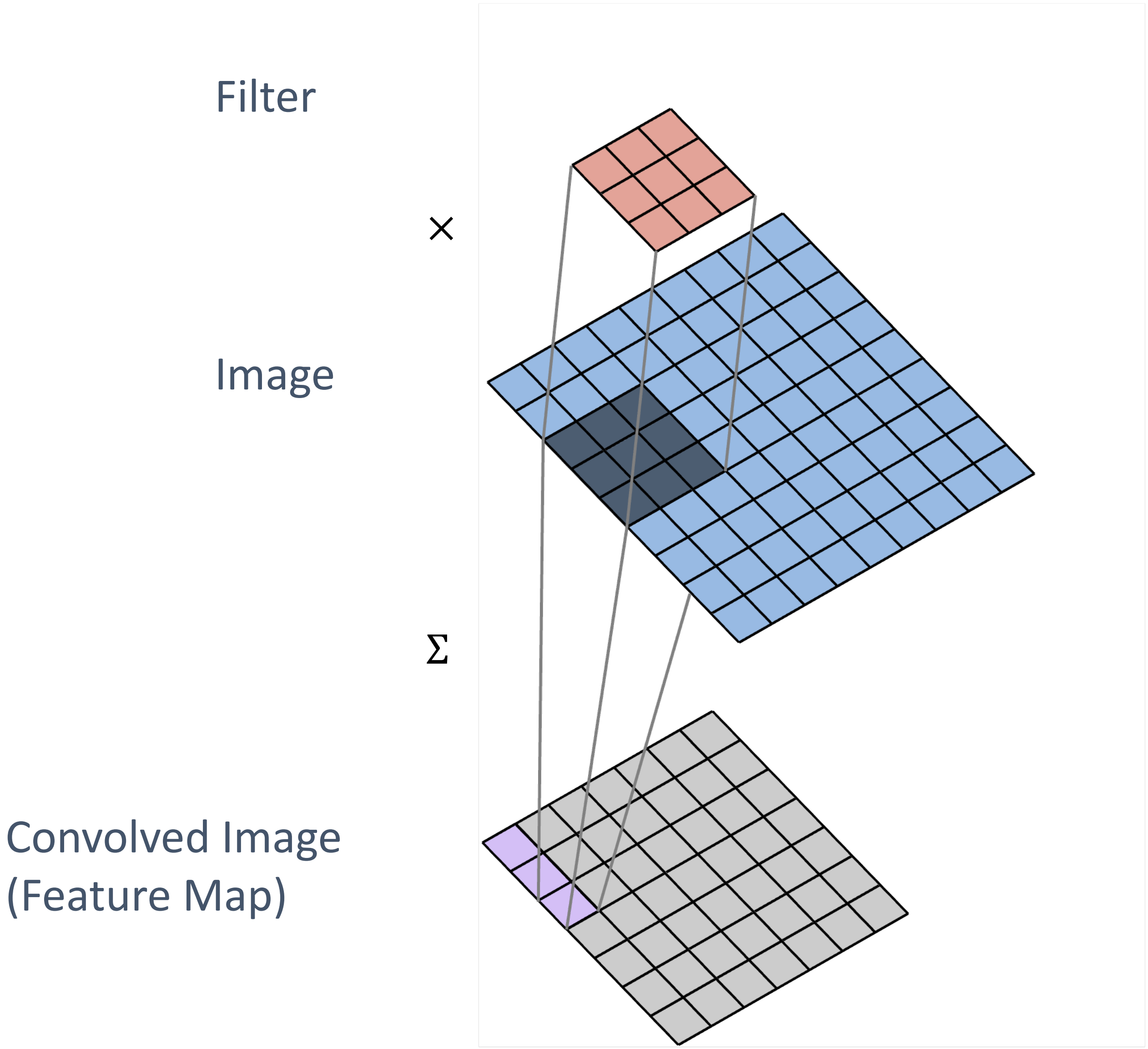
Filter

Image

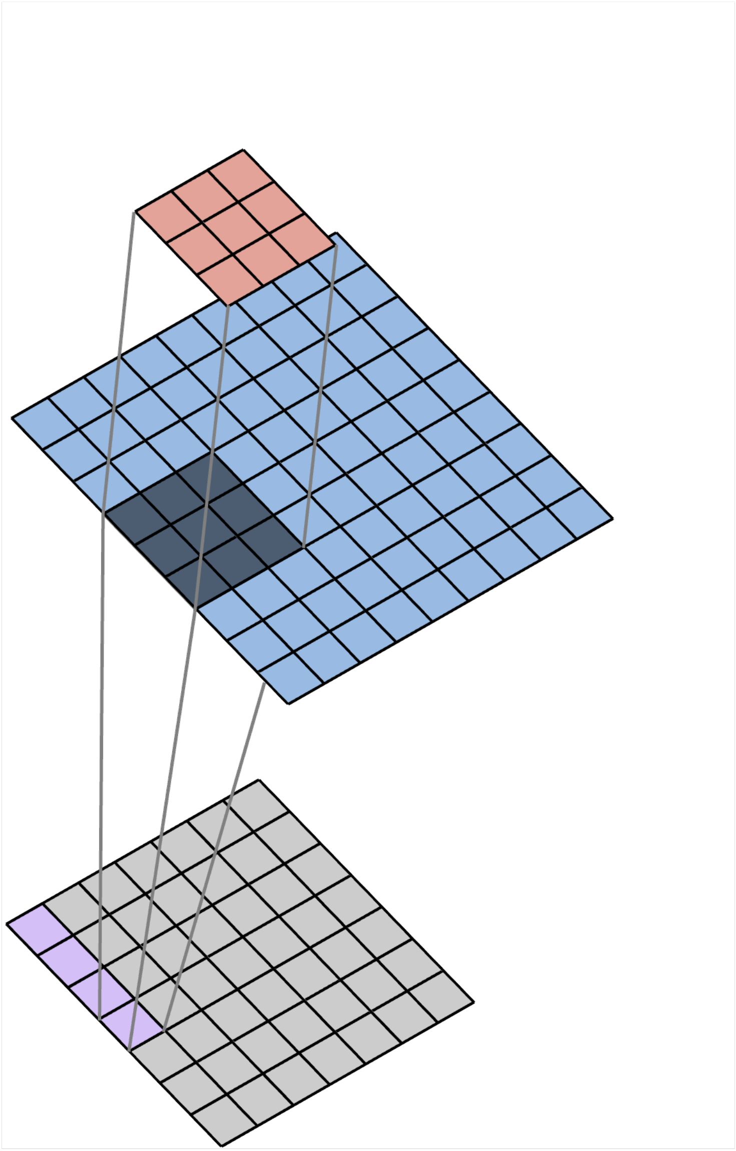
Convolved Image
(Feature Map)



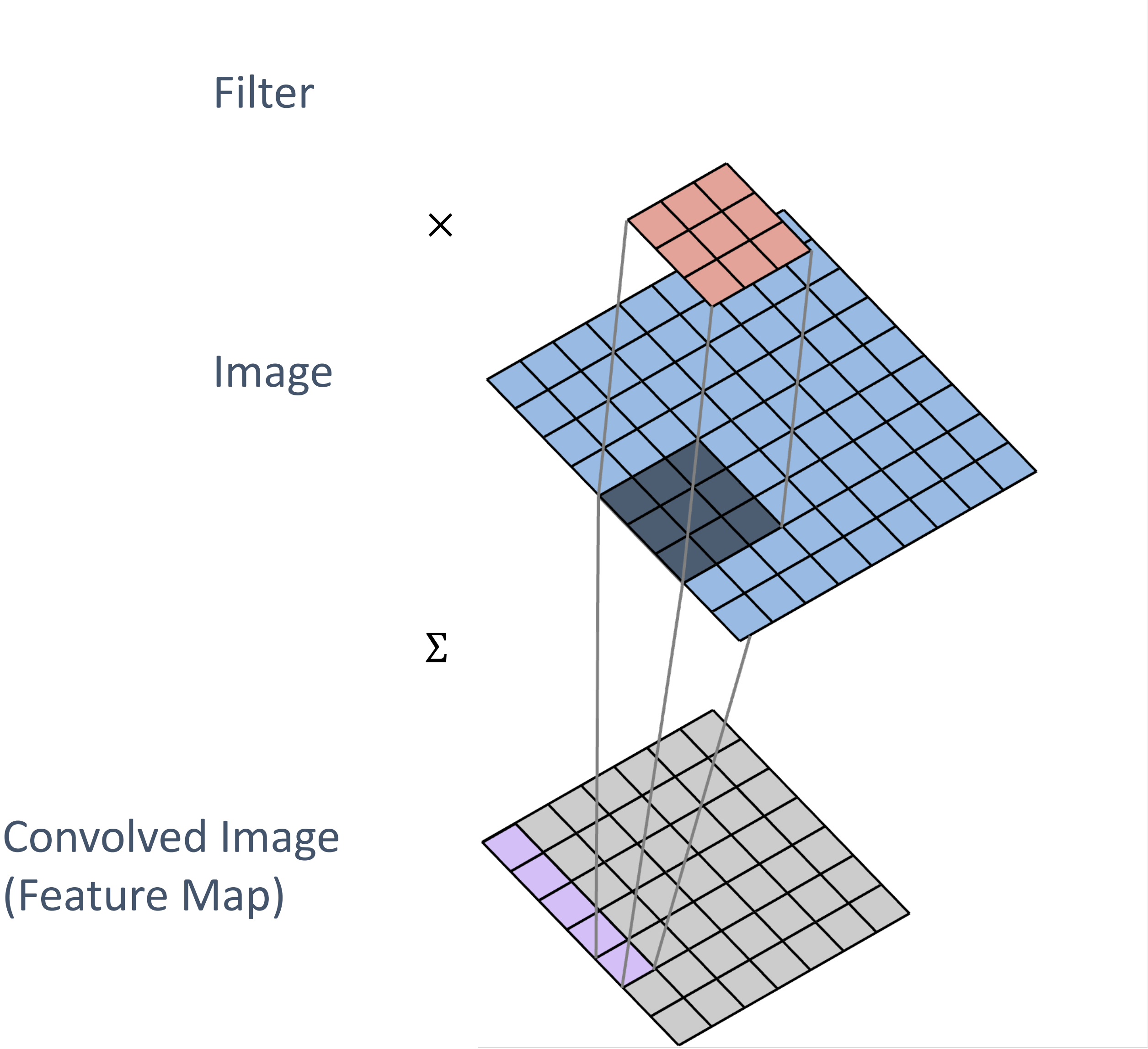
2D Spatial Convolution



2D Spatial Convolution



2D Spatial Convolution



2D Spatial Convolution

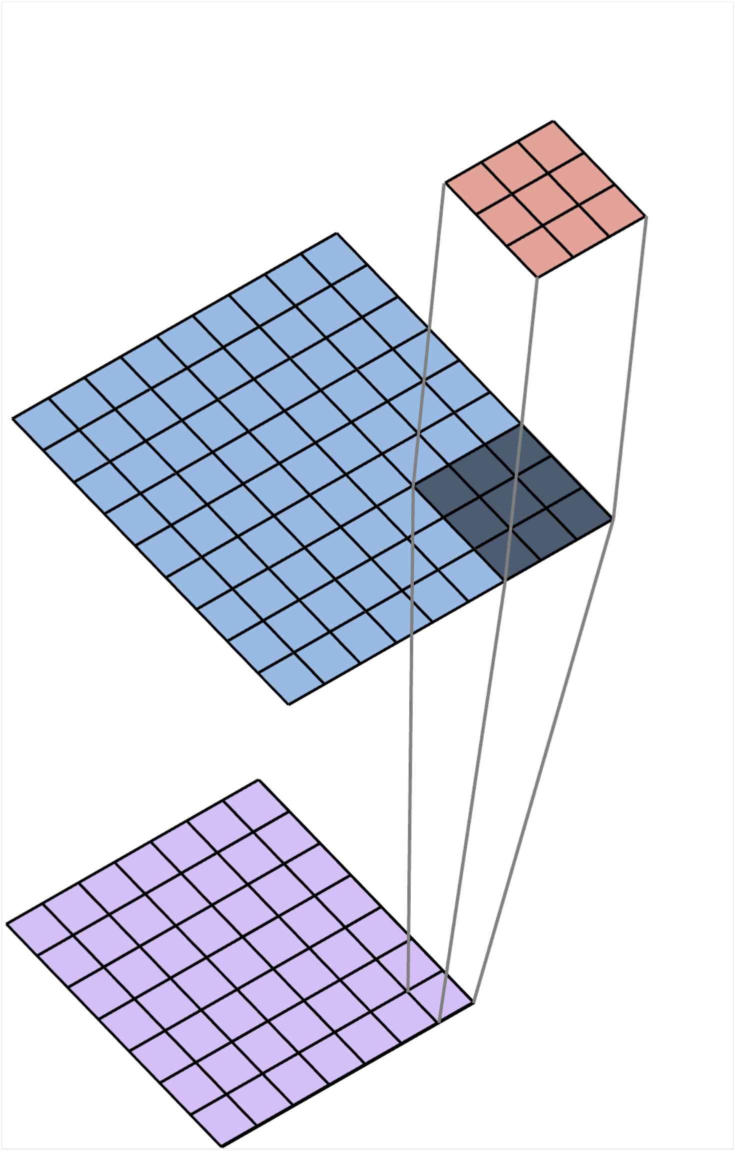
Filter

×

Image

Σ

Convolved Image
(Feature Map)

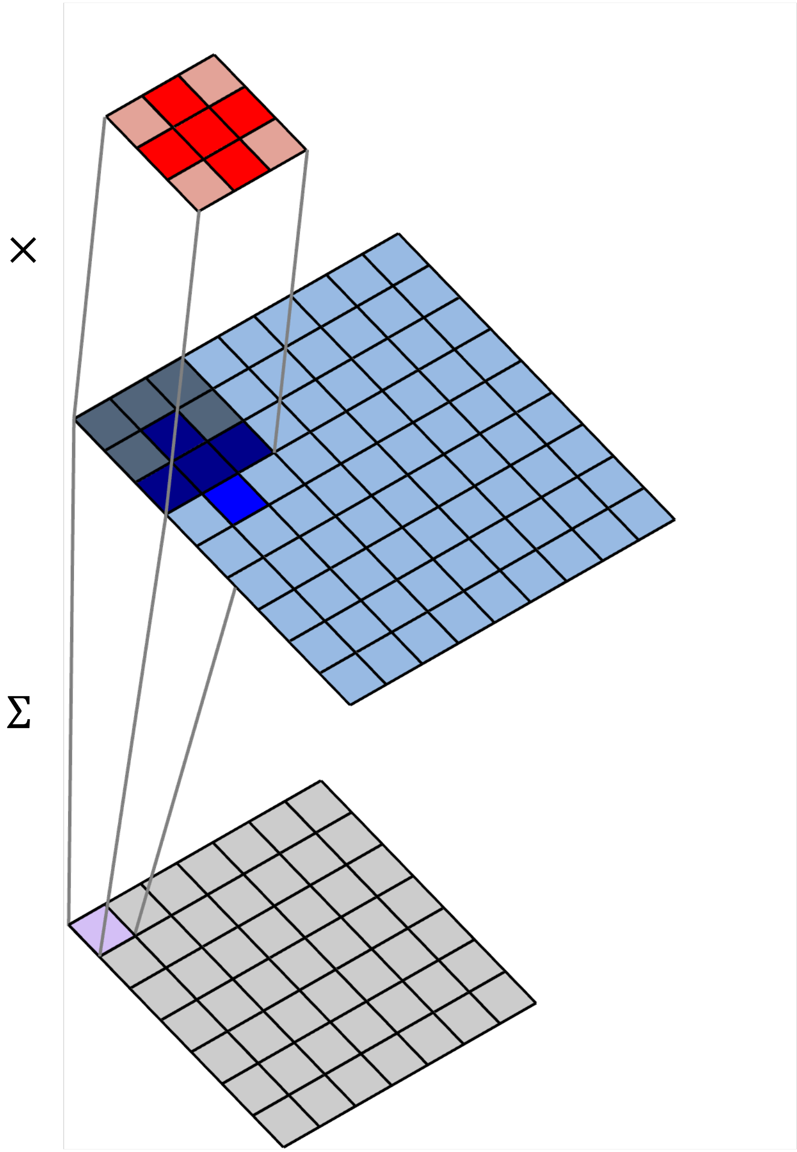


2D Spatial Convolution

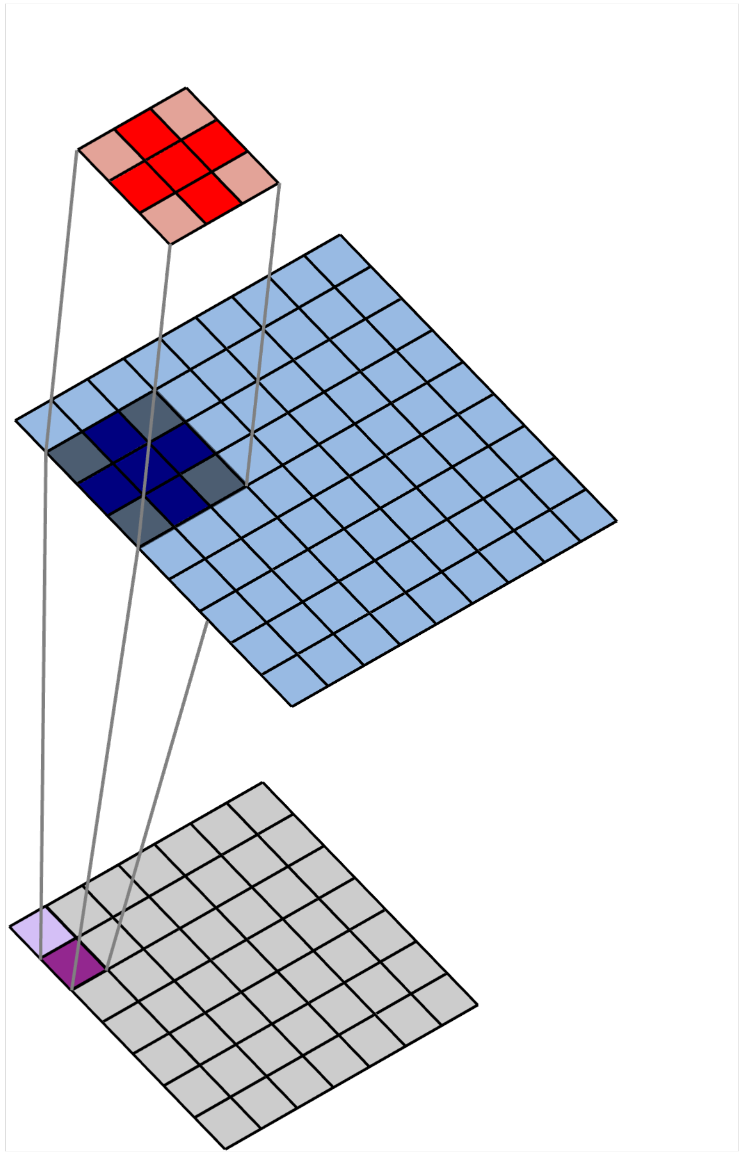
Filter

Image

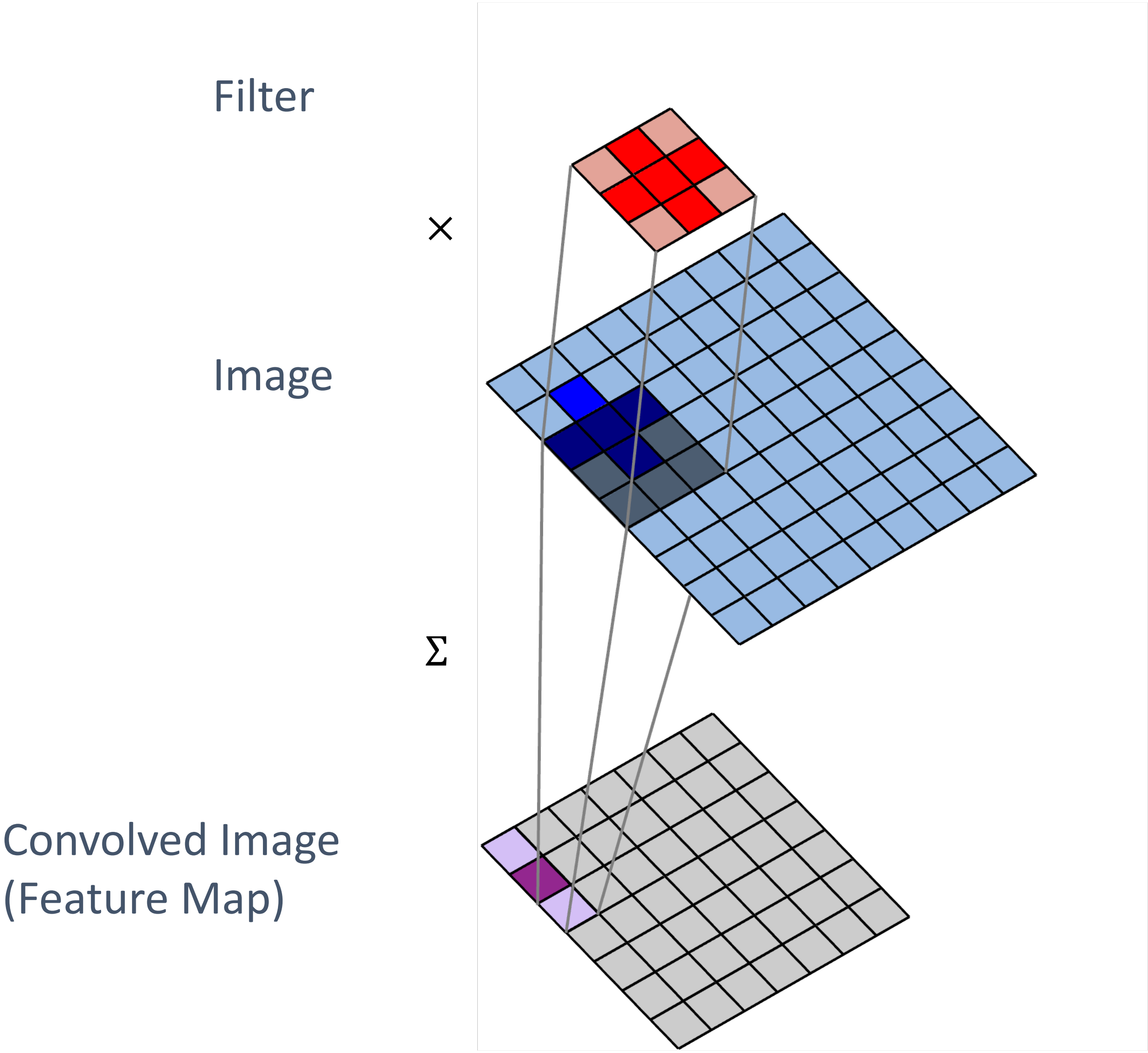
Convolved Image
(Feature Map)



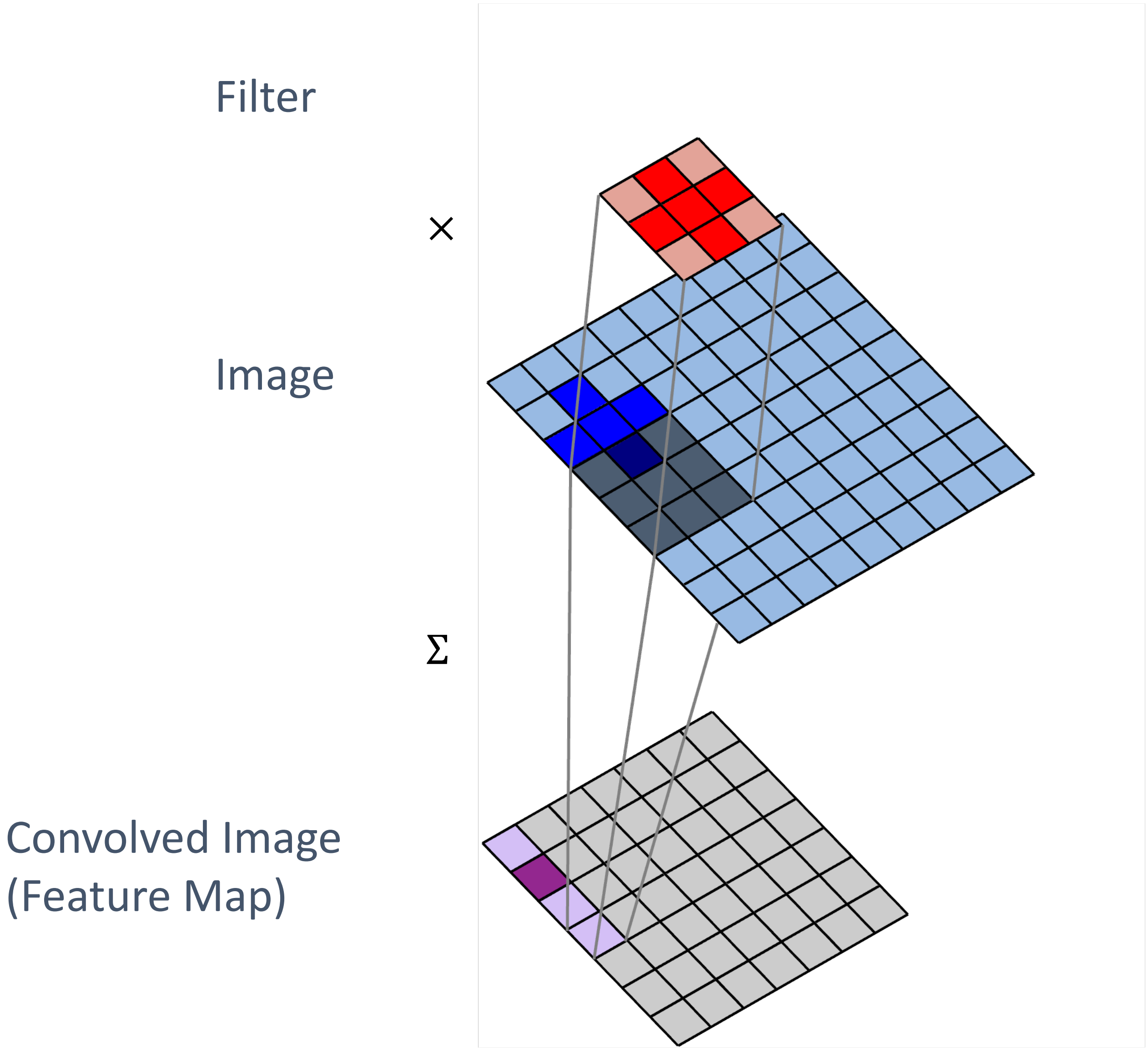
2D Spatial Convolution



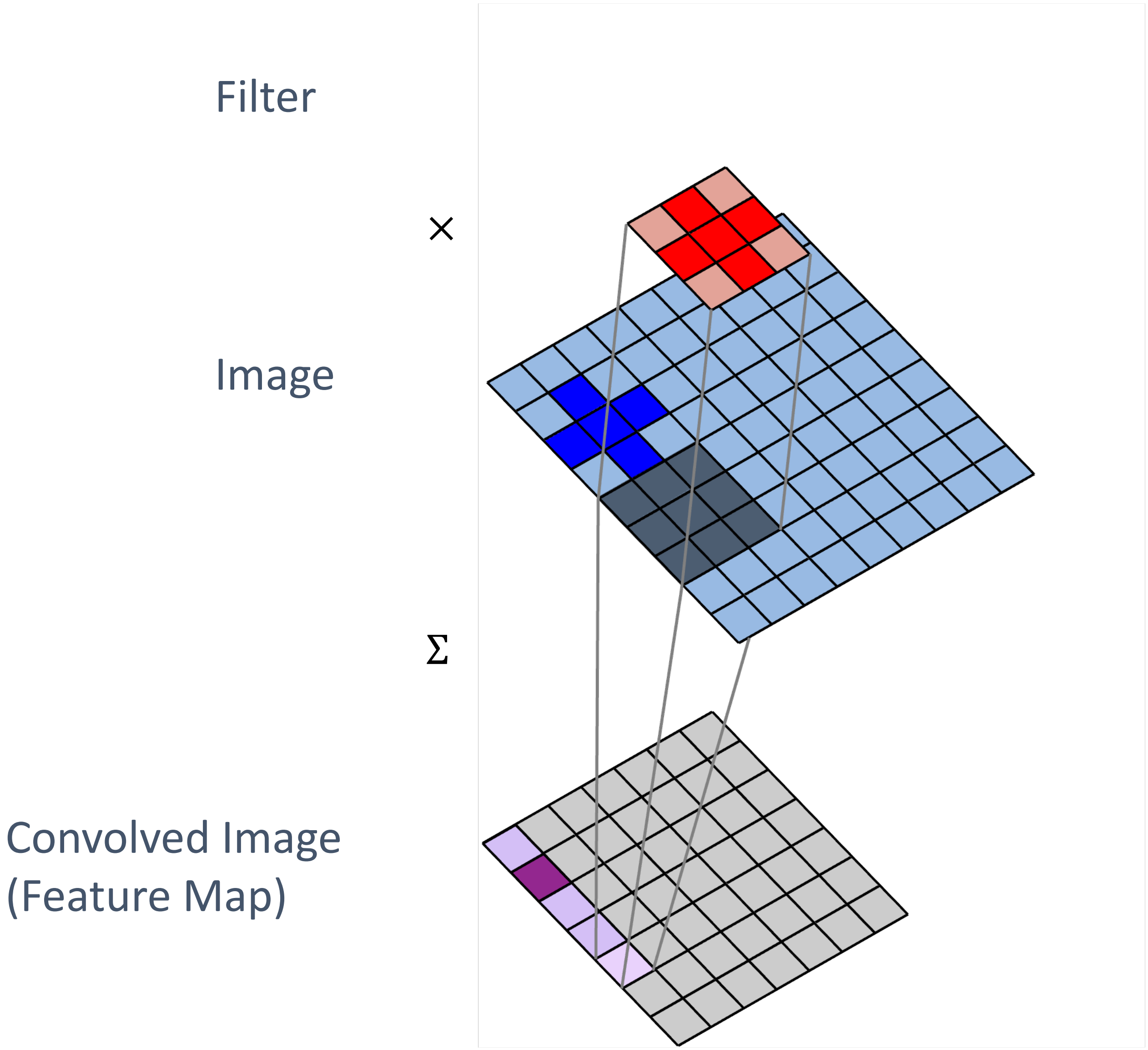
2D Spatial Convolution



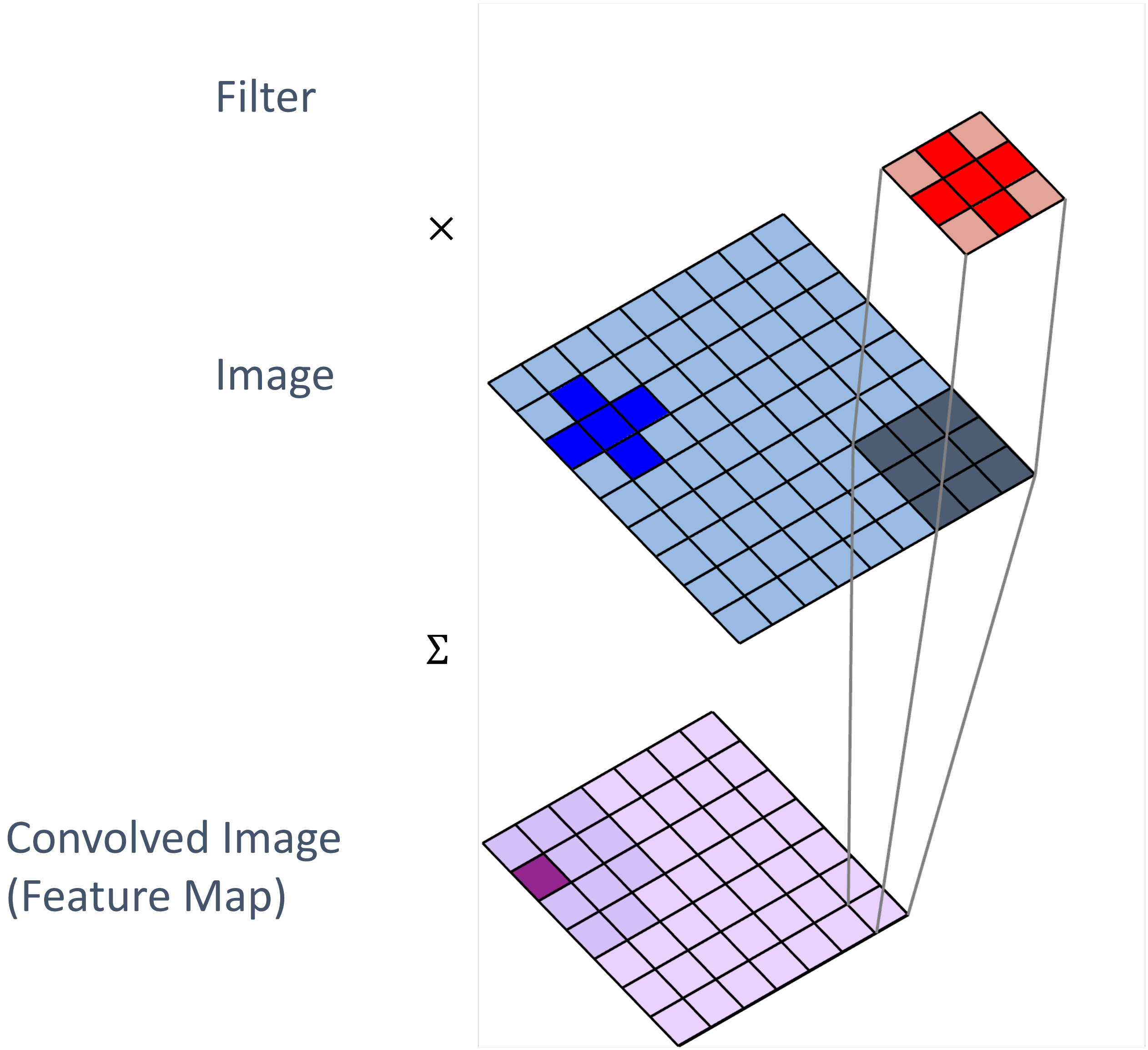
2D Spatial Convolution



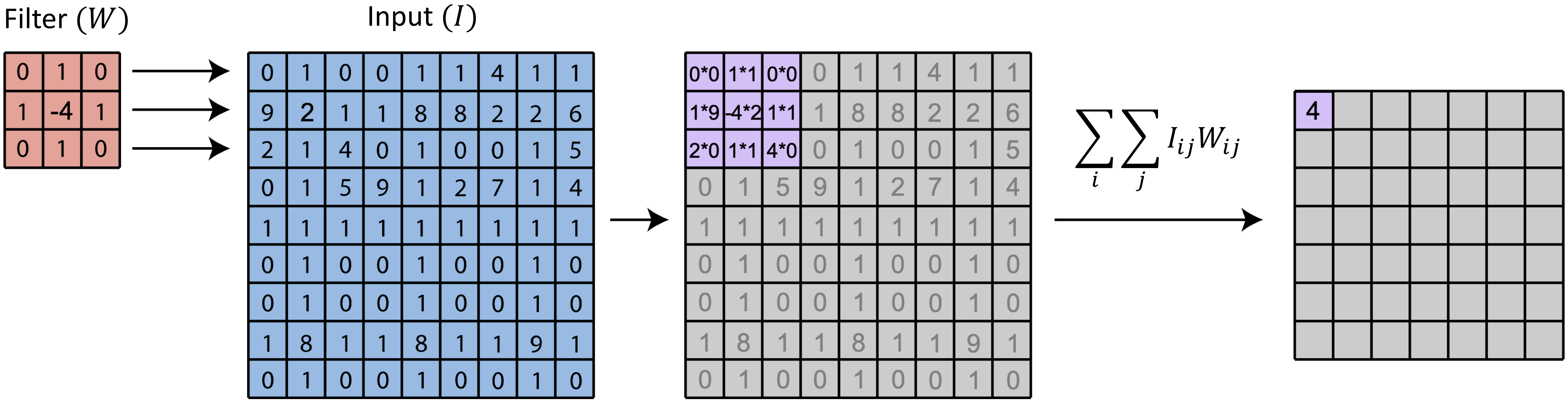
2D Spatial Convolution



2D Spatial Convolution



2D Spatial Convolution



2D Spatial Convolution

Filter (W)

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

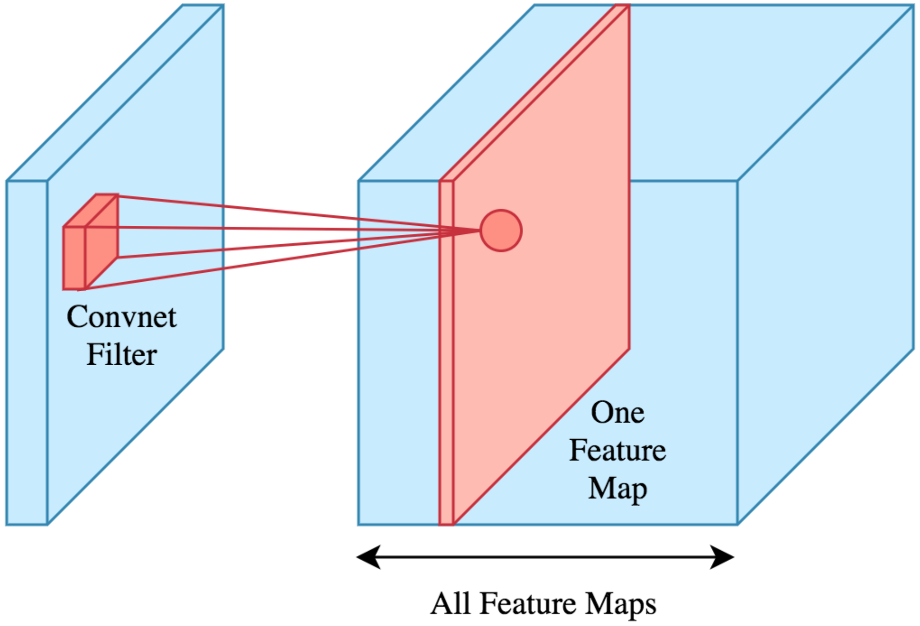
Input (I)

Convolutional Filters Are Feature Detectors

Core Elements of the Convolutional Neural Network

- **Convolutional Layers**
- **Activation Functions**
- **Pooling Layers**
- **Fully Connected Layers**

Convolutional Layer

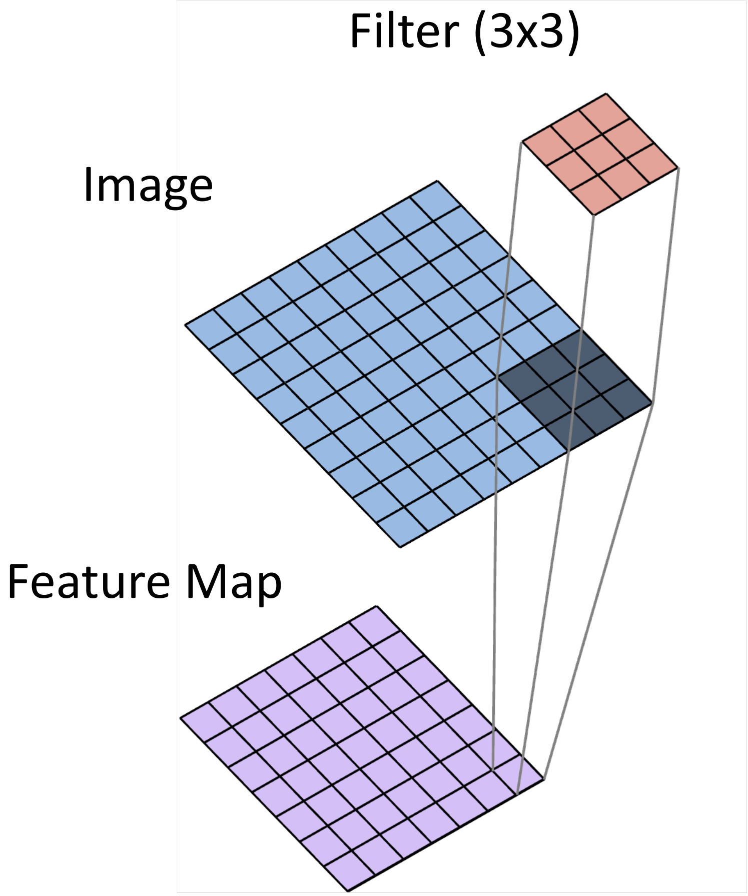


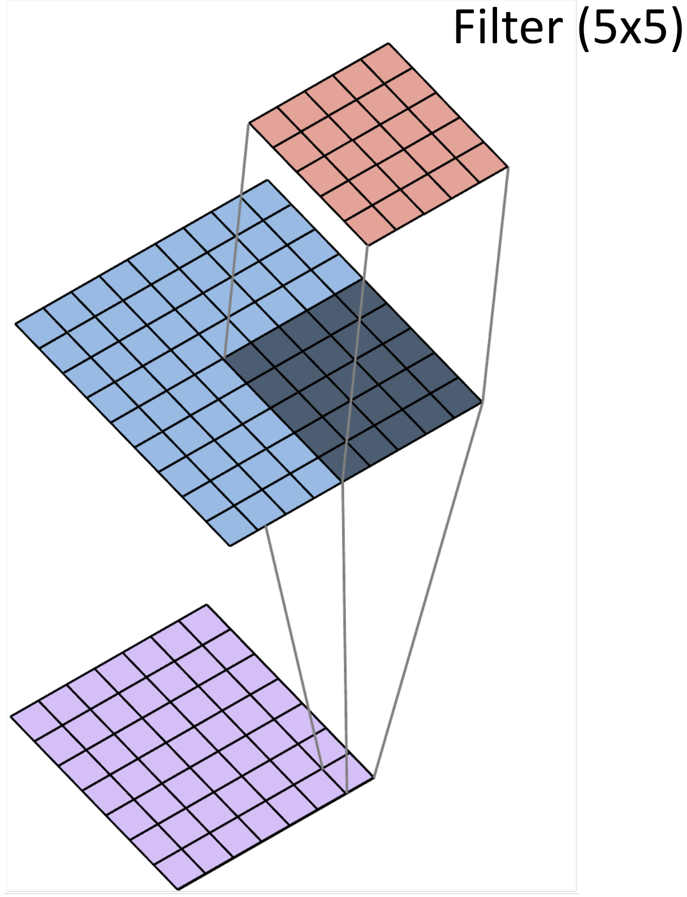
Elements of a convolutional layer:

Filter Size
Filter Stride
Filter (Feature) Number

Convolutional Layer

Filter Size

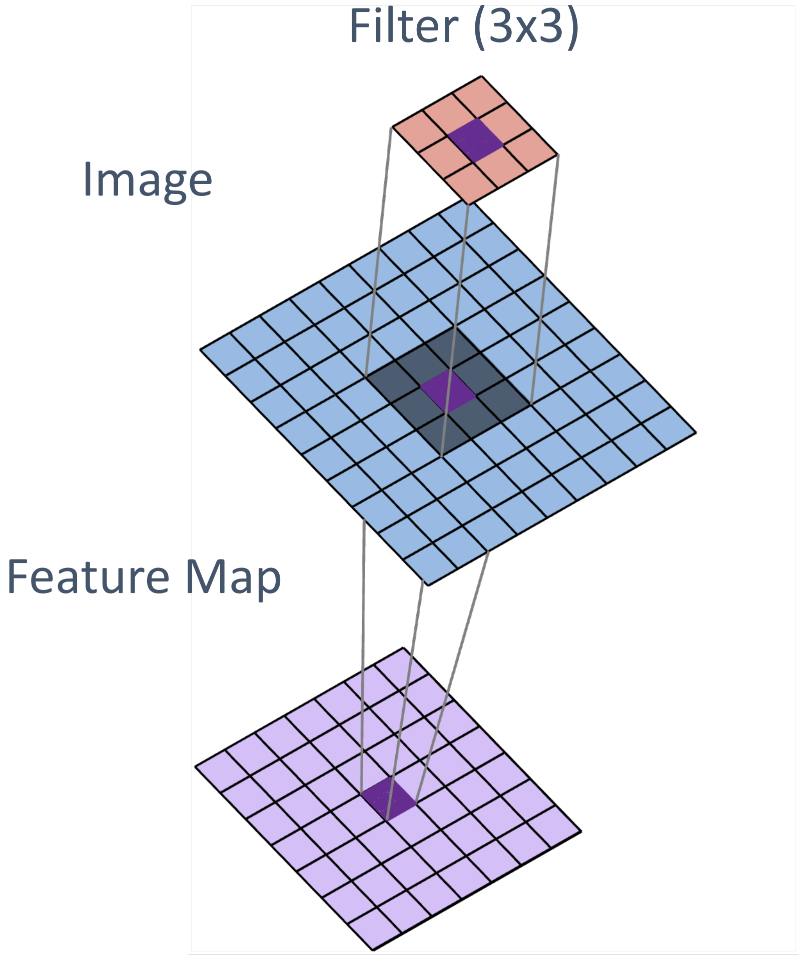




Filters should be just large enough to capture small local features (*e.g.*, edges) in space

Convolutional Layer

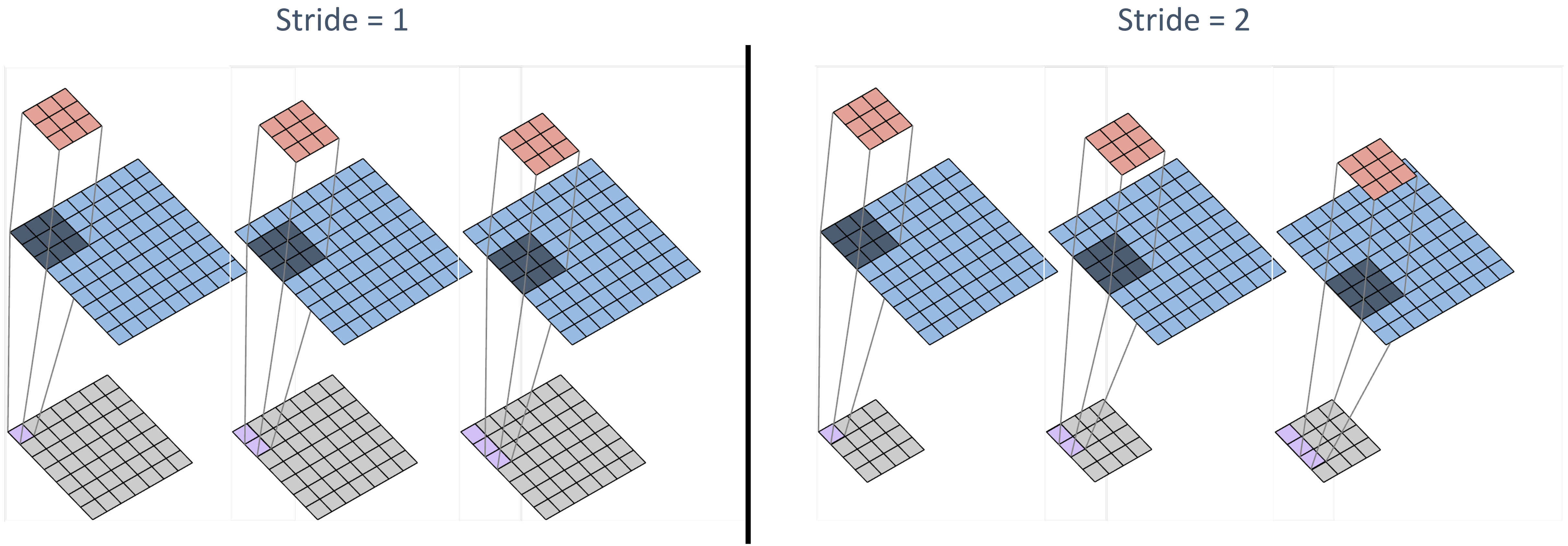
Filter Size



Filter size is typically odd to enforce symmetry

Convolutional Layer

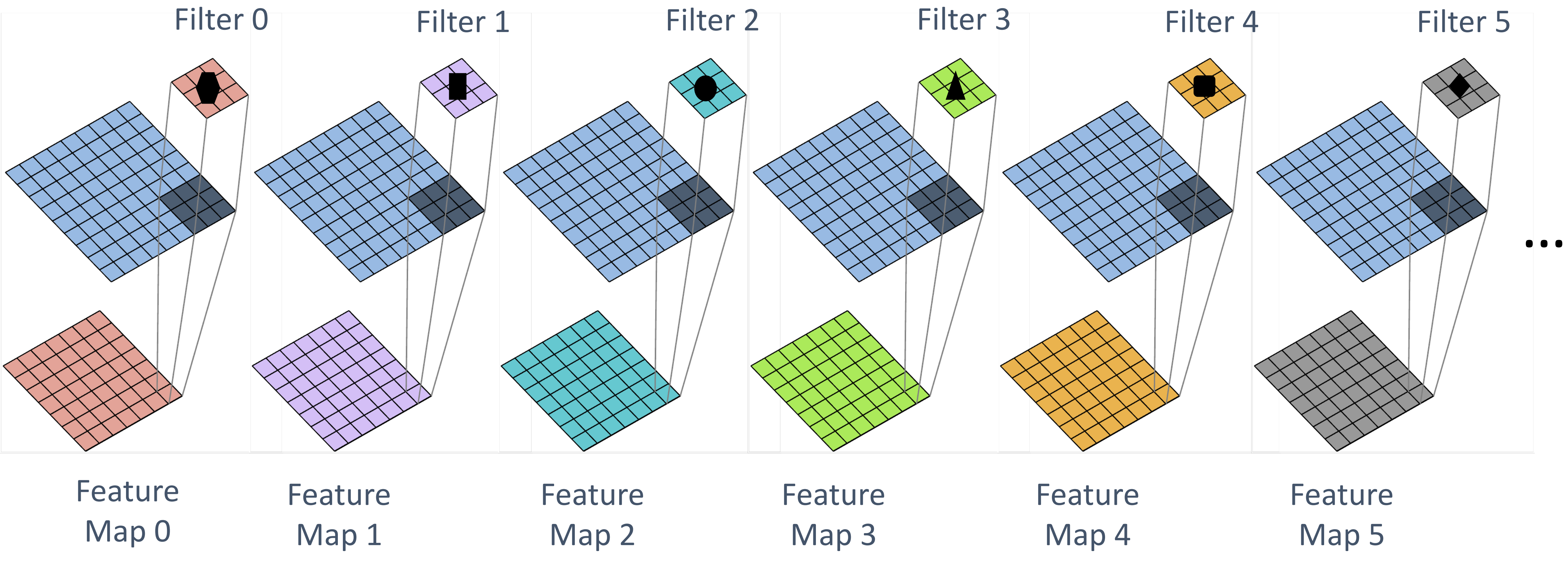
Filter Stride



Filter stride > 1 reduces computational load by downsampling the input

Convolutional Layer

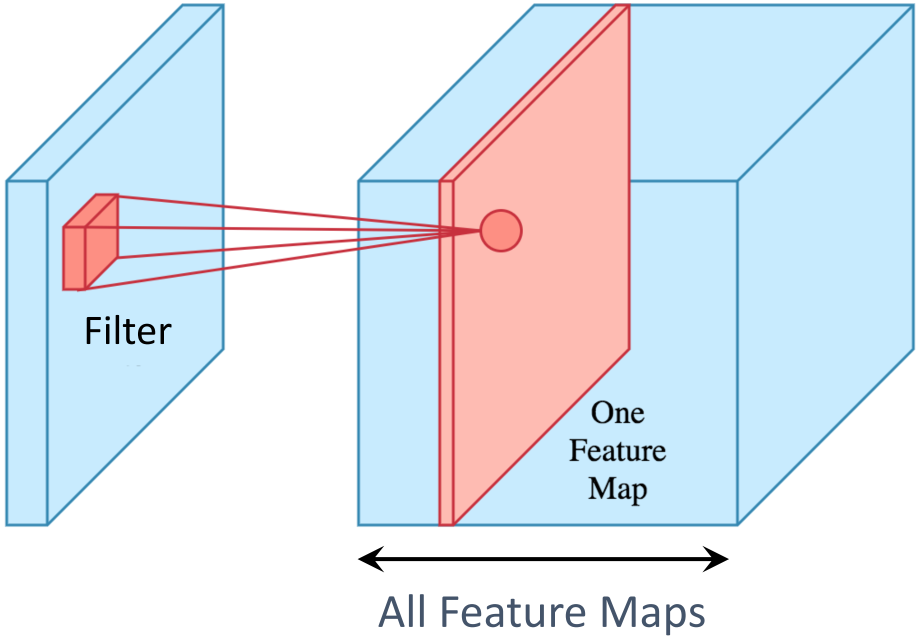
Filter (Feature) Number



Filter number determines the number of unique feature detectors that operate on inputs

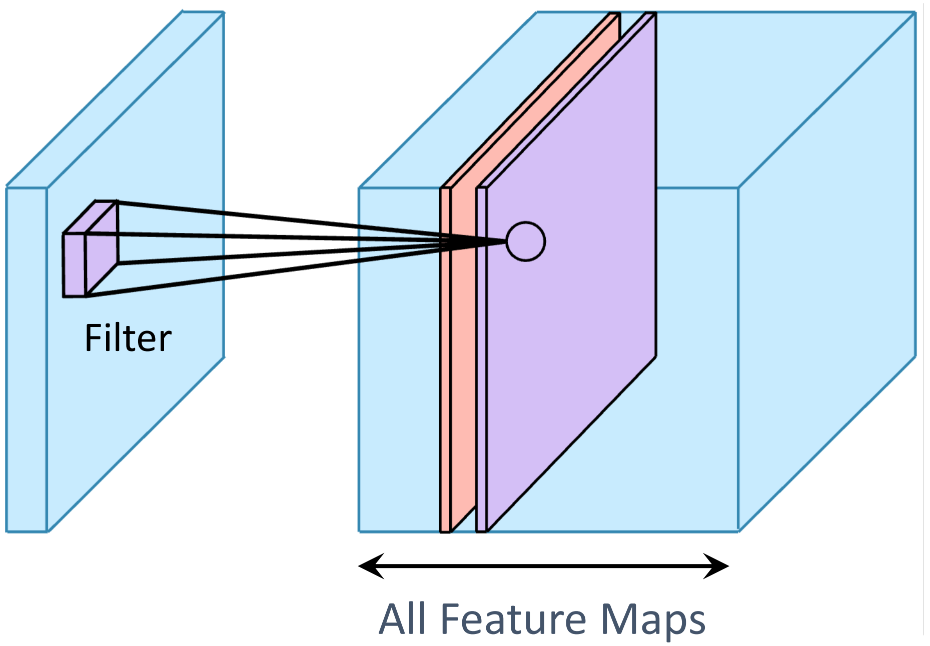
Convolutional Layer

Filter (Feature) Number



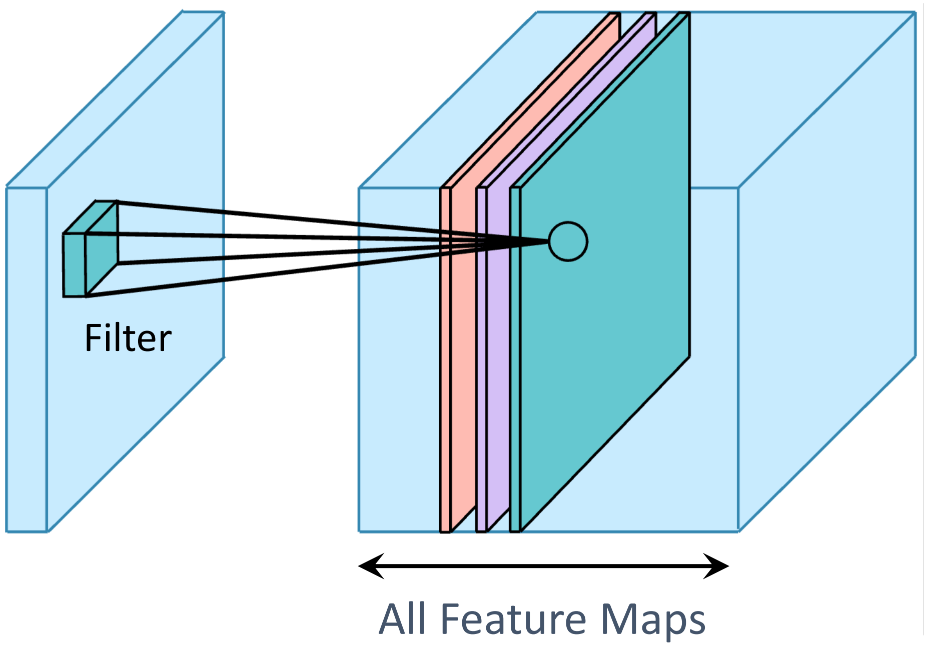
Convolutional Layer

Filter (Feature) Number



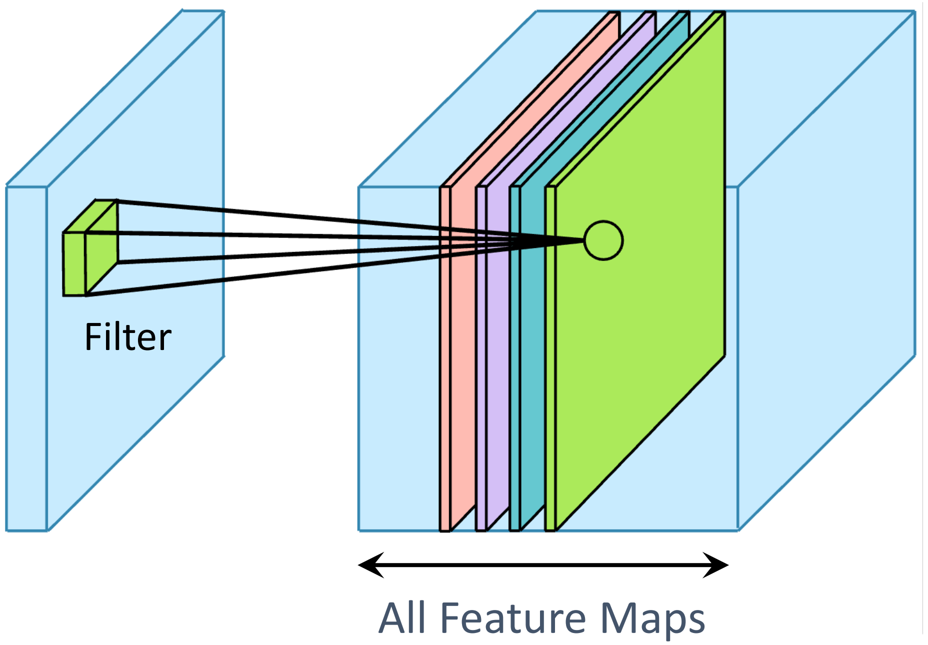
Convolutional Layer

Filter (Feature) Number



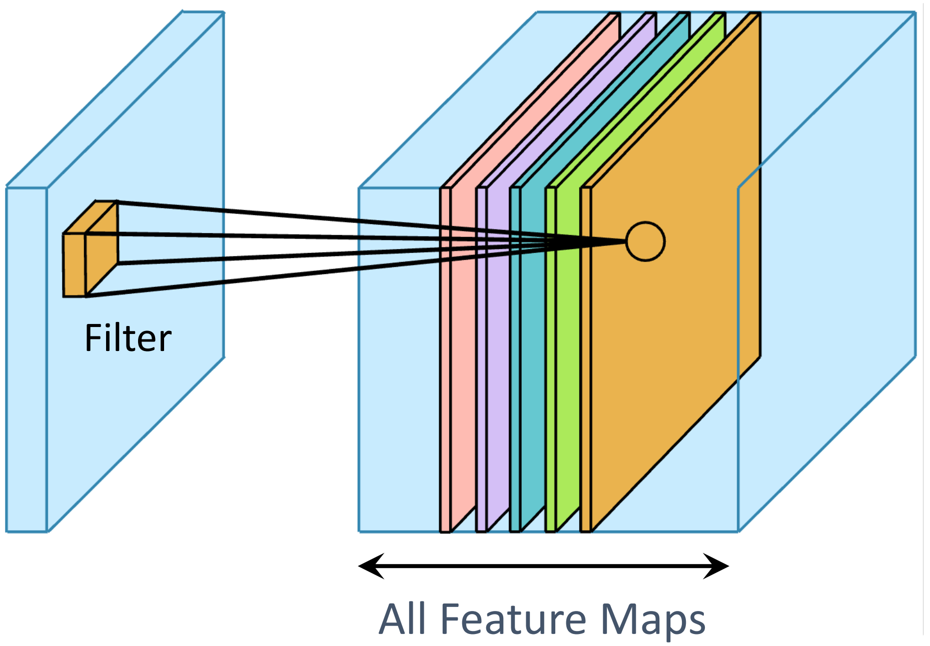
Convolutional Layer

Filter (Feature) Number



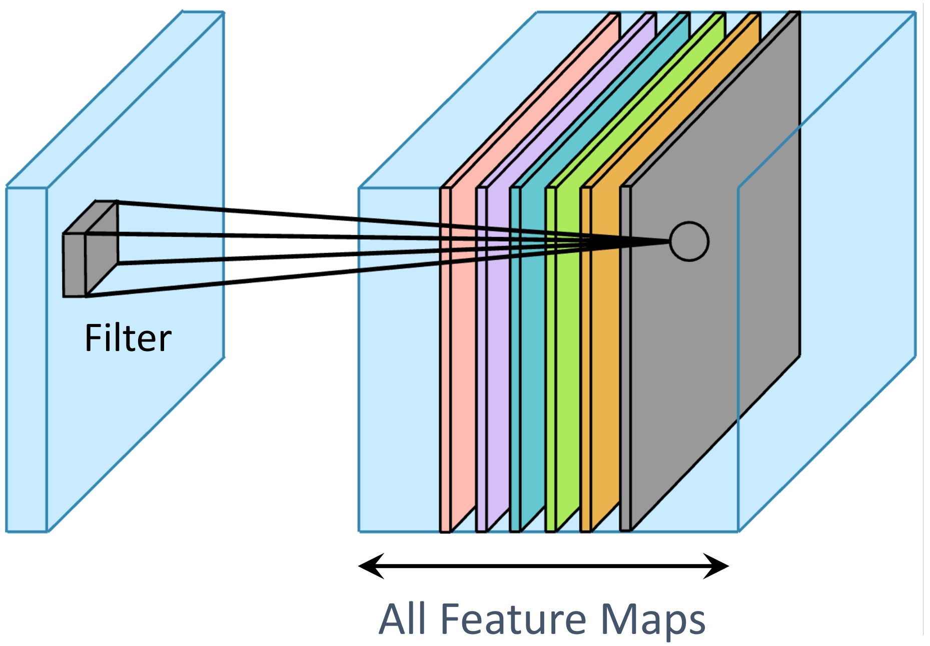
Convolutional Layer

Filter (Feature) Number

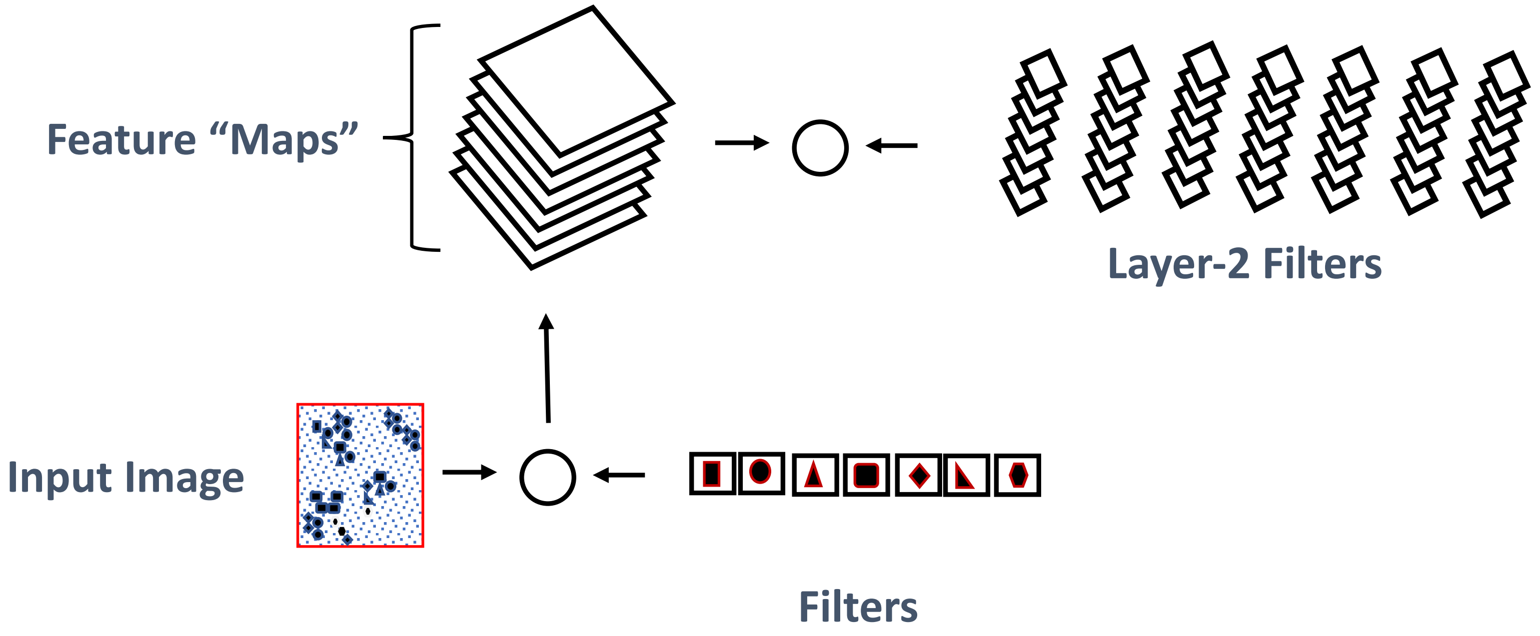


Convolutional Layer

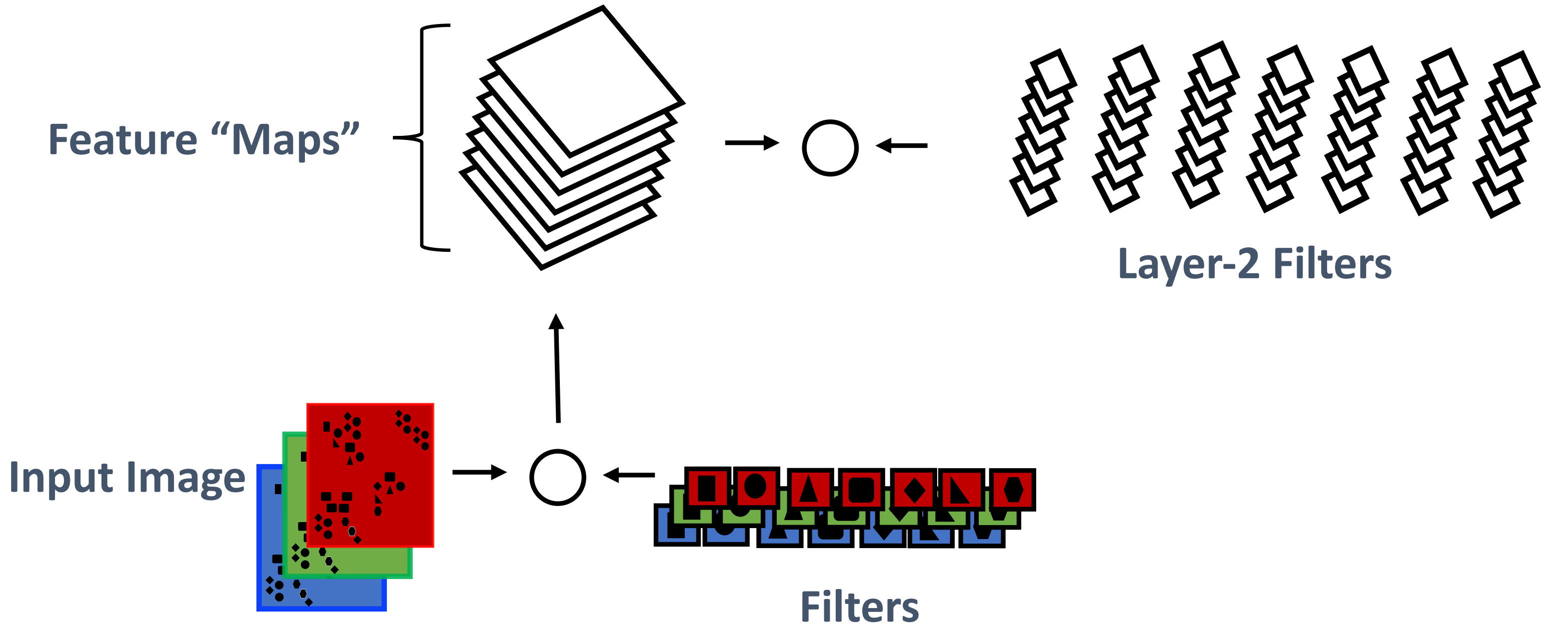
Filter (Feature) Number



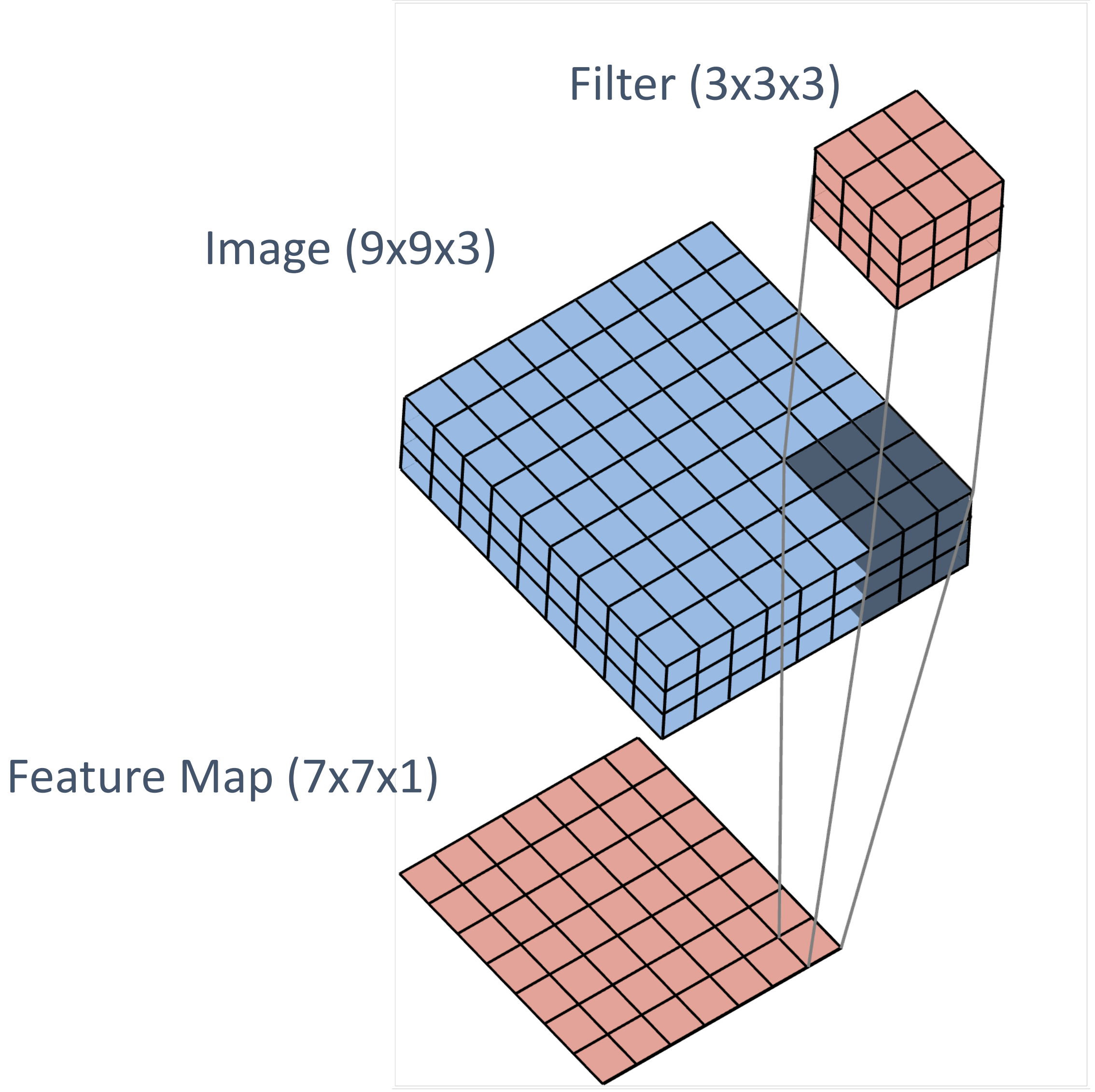
Filters Operate Over Input Volumes



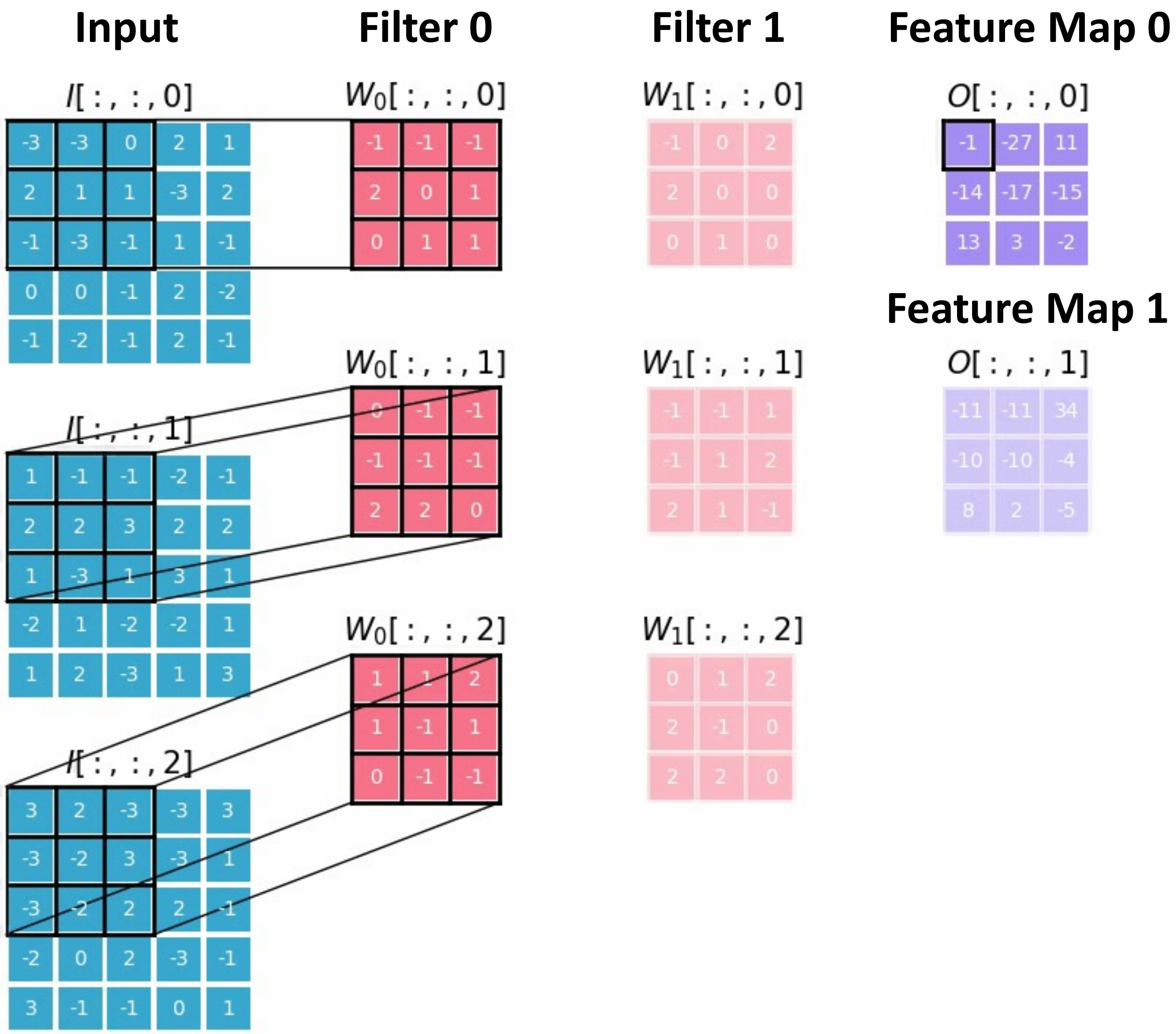
Filters Operate Over Input Volumes



Filters Operate Over Input Volumes



Convolutional Layer



Inspired by Stanford 231n

Activation Functions

Filter (W)

0	1	0
1	-4	1
0	1	0

Input (I)

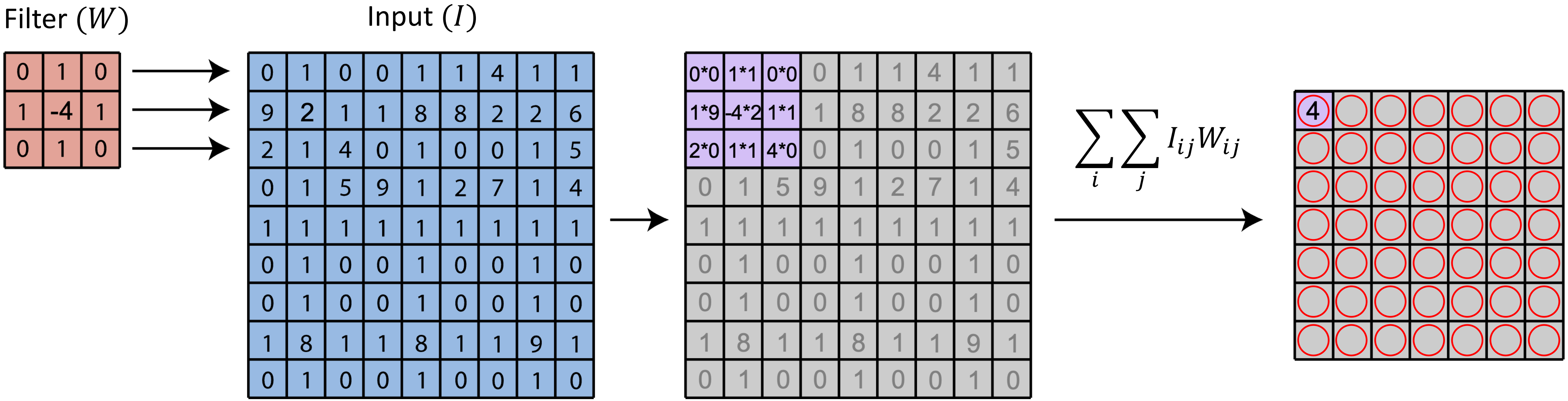
0	1	0	0	1	1	4	1	1
9	2	1	1	8	8	2	2	6
2	1	4	0	1	0	0	1	5
0	1	5	9	1	2	7	1	4
1	1	1	1	1	1	1	1	1
0	1	0	0	1	0	0	1	0
0	1	0	0	1	0	0	1	0
1	8	1	1	8	1	1	9	1
0	1	0	0	1	0	0	1	0

0*0	1*1	0*0	0	1	1	4	1	1
1*9	-4*2	1*1	1	8	8	2	2	6
2*0	1*1	4*0	0	1	0	0	1	5
0	1	5	9	1	2	7	1	4
1	1	1	1	1	1	1	1	1
0	1	0	0	1	0	0	1	0
0	1	0	0	1	0	0	1	0
1	8	1	1	8	1	1	9	1
0	1	0	0	1	0	0	1	0

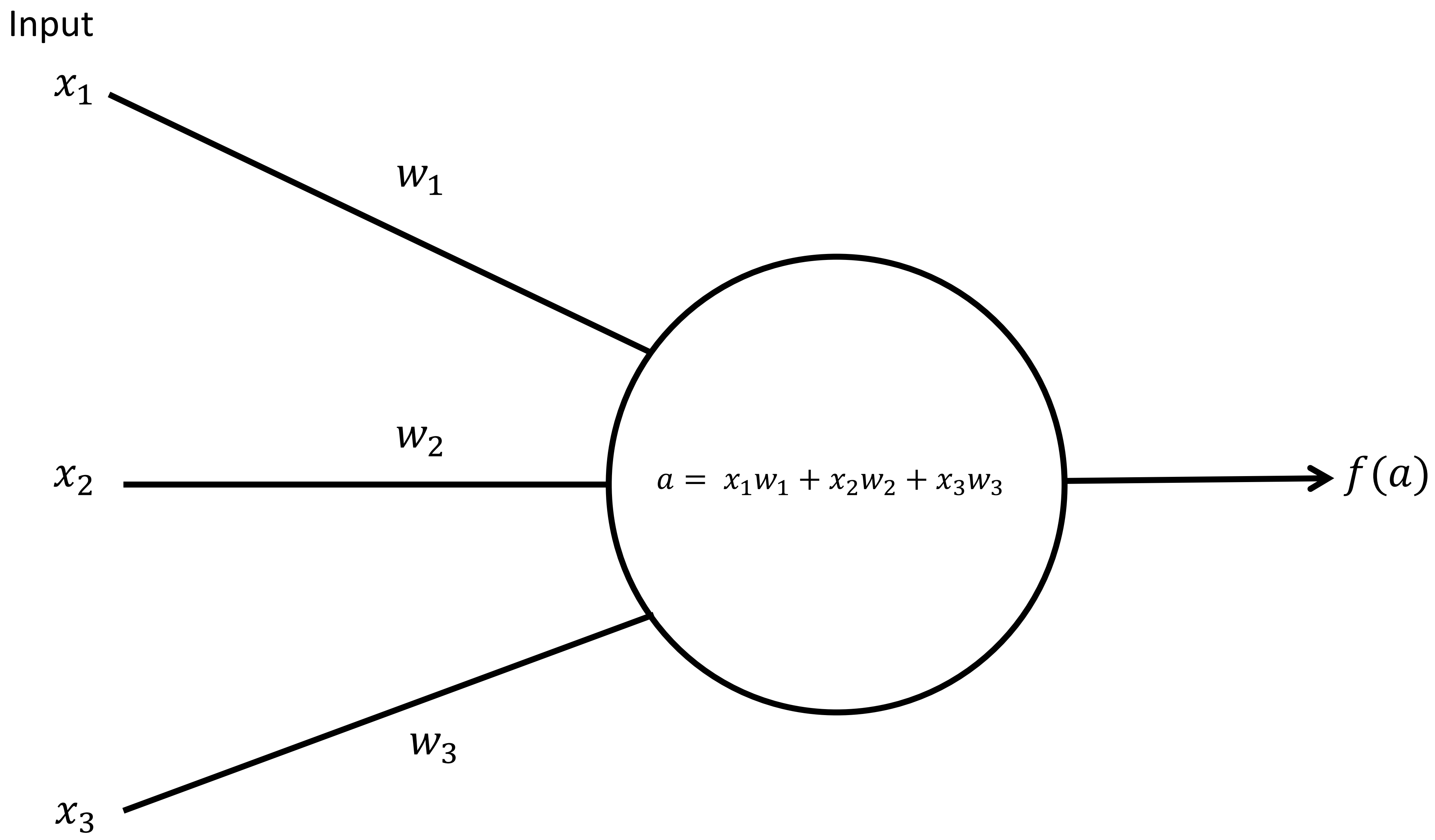
$$\sum_i \sum_j I_{ij} W_{ij}$$

4						

Activation Functions

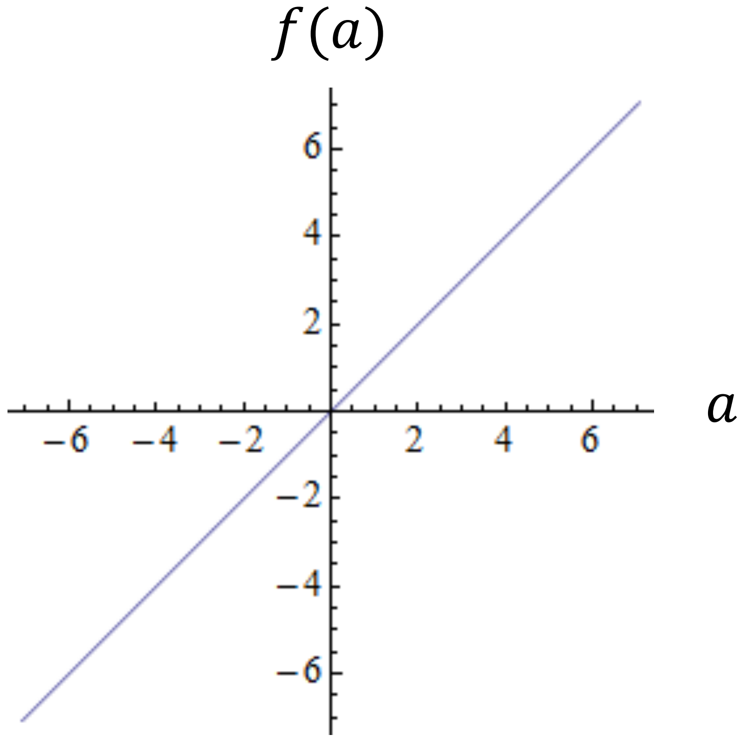


Activation Functions



Activation Functions

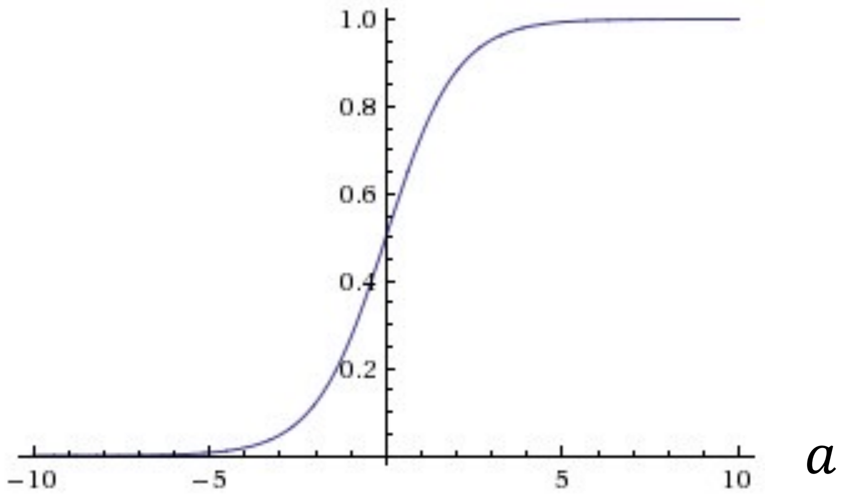
Linear Activation



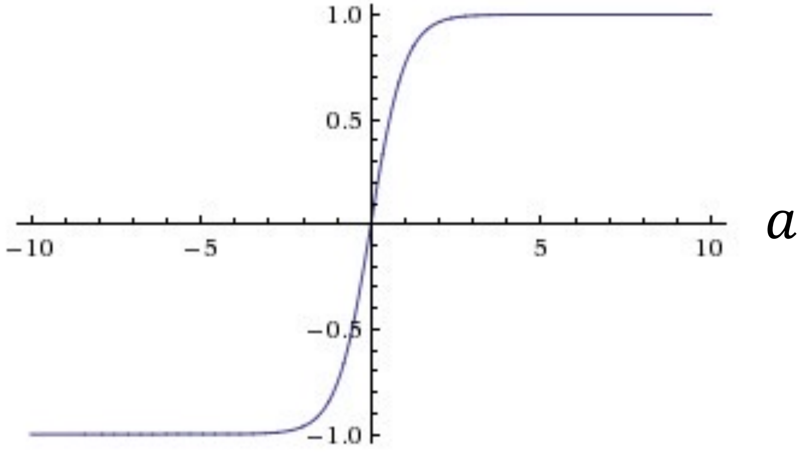
Activation Functions

Non-Linear Activations

$$f(a) = \sigma(a)$$



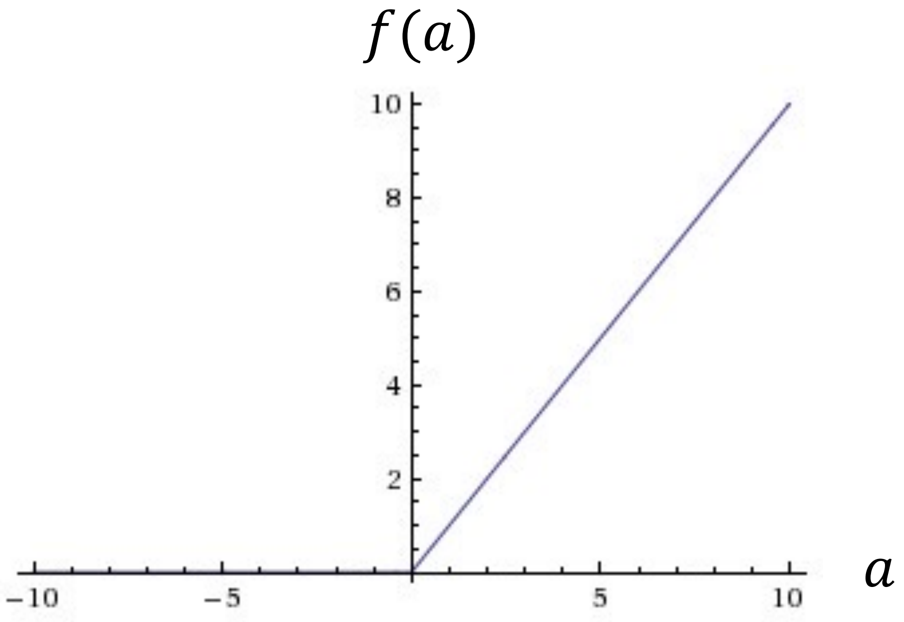
$$f(a) = \tanh(a)$$



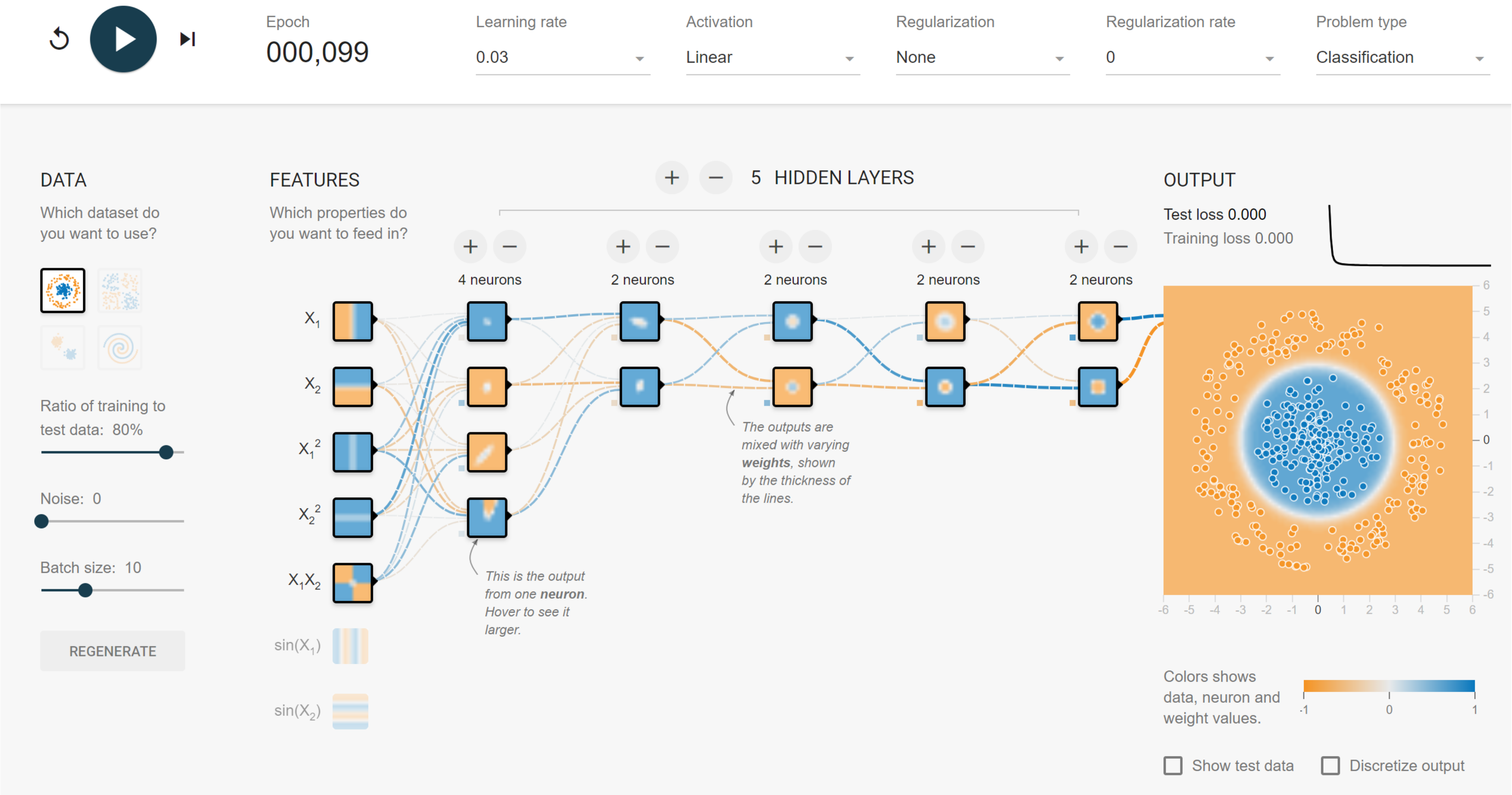
Non-linear activations increase the functional capacity of the neural network

Activation Functions

Non-Linear Activation:
Rectified Linear Unit (ReLU)

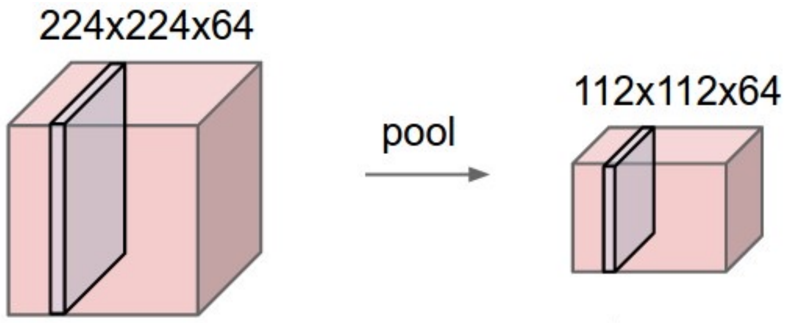


Activation Functions



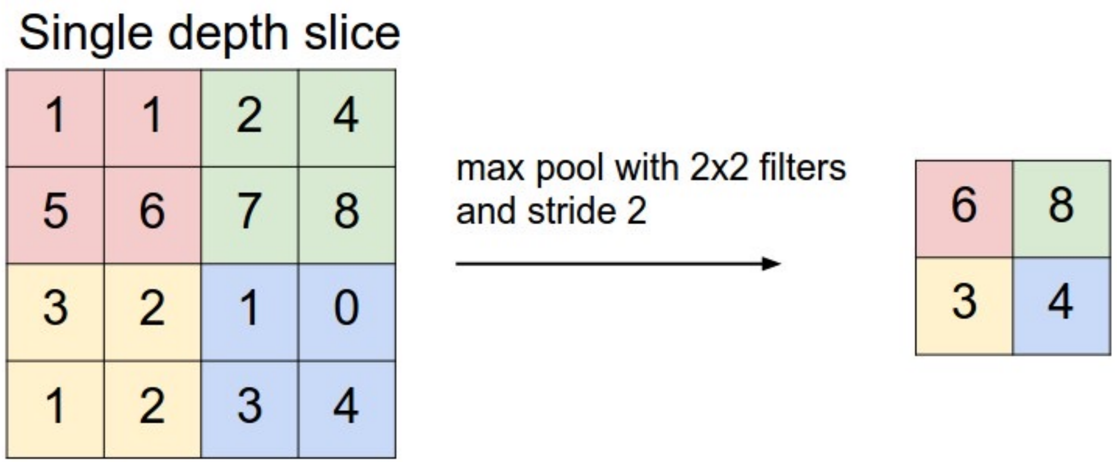
playground.tensorflow.org

Pooling Layer



- Reduces computational complexity
- Combats overfitting
- Encourages translational invariance

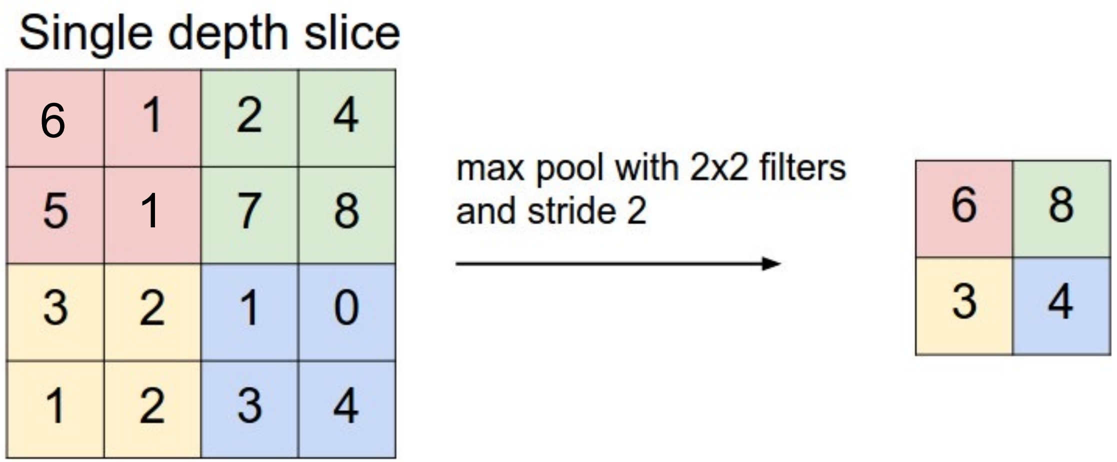
Pooling Layer



Pooling layers also have **width** and **stride**.

Pooling is typically done by taking the **maximum** across the pooling area

Pooling Layer

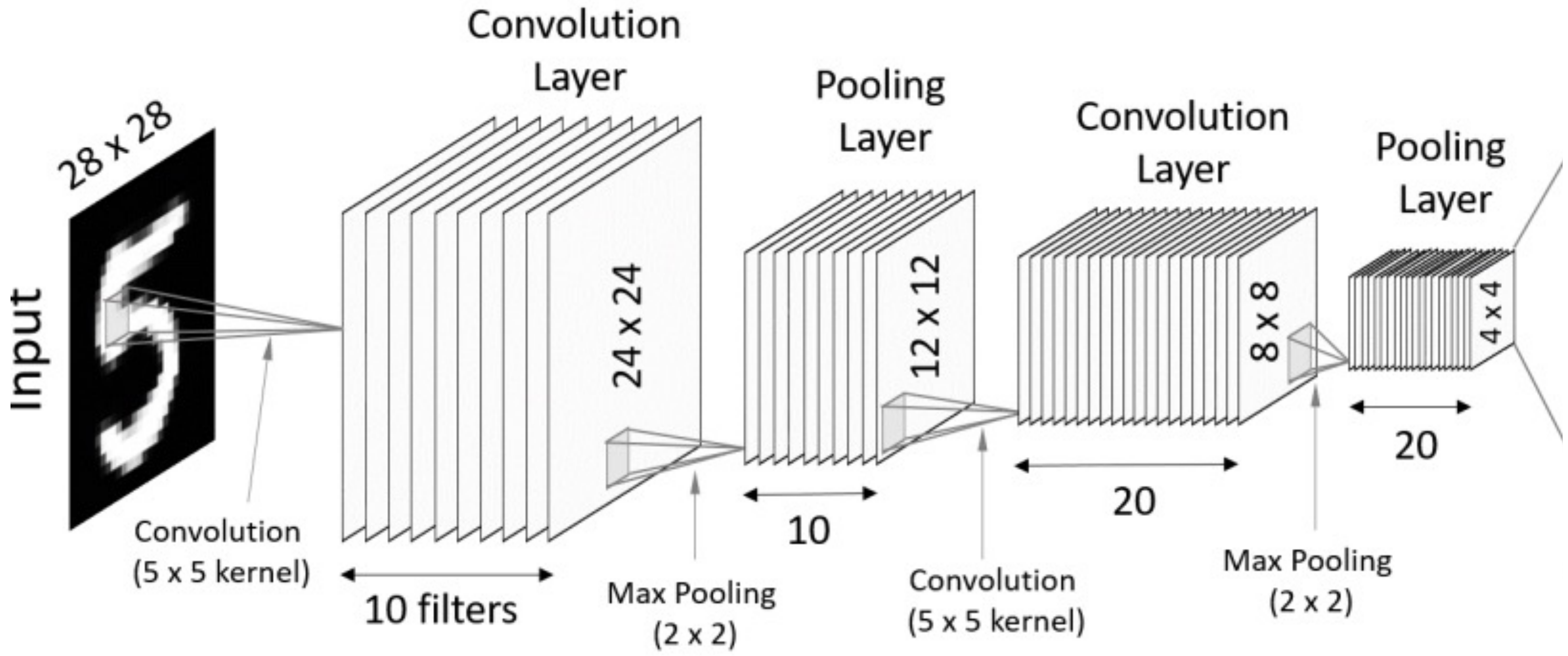


Max Pooling picks out strong activations with some position independence

Fully Connected Layer

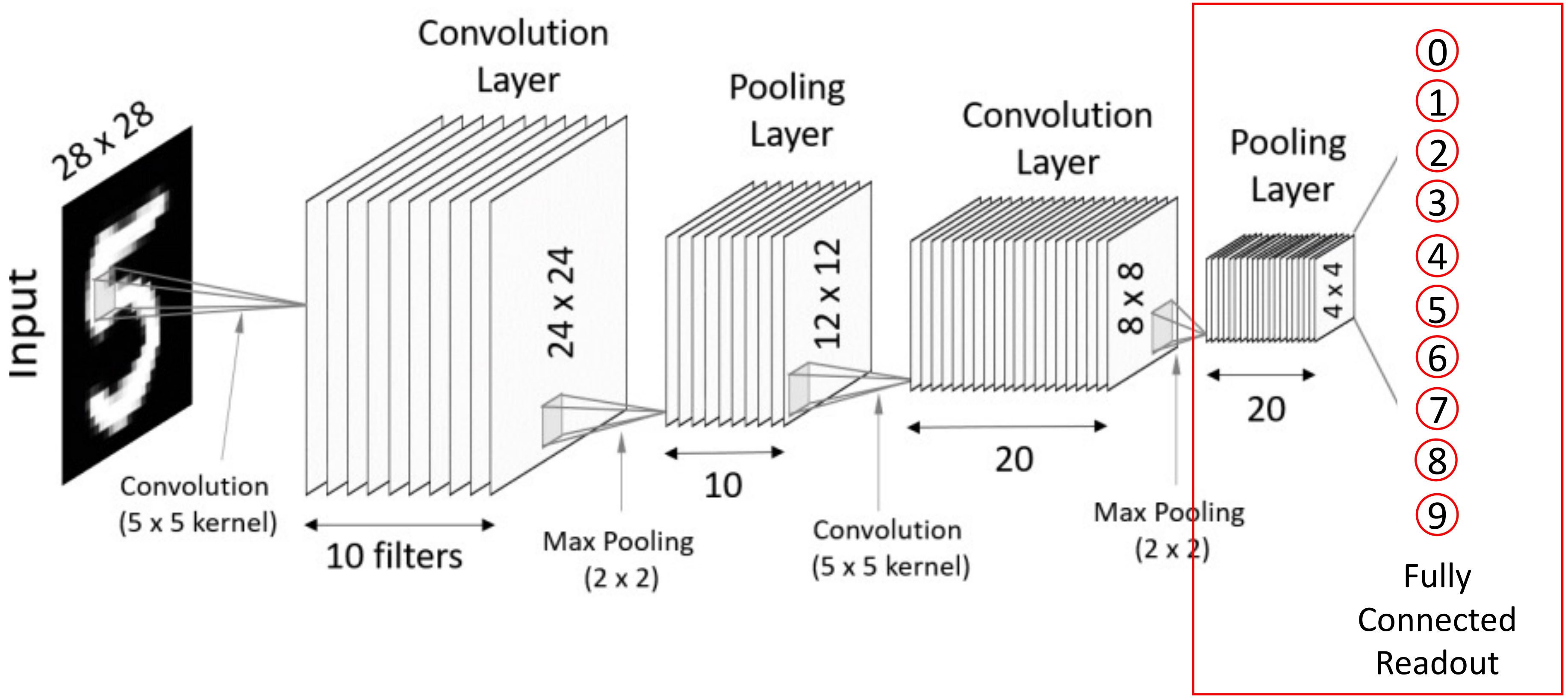
Convolutional and pooling layers are stacked to build up high-level feature representations

How are these high-level features processed to arrive at a final classification?



Fully Connected Layer

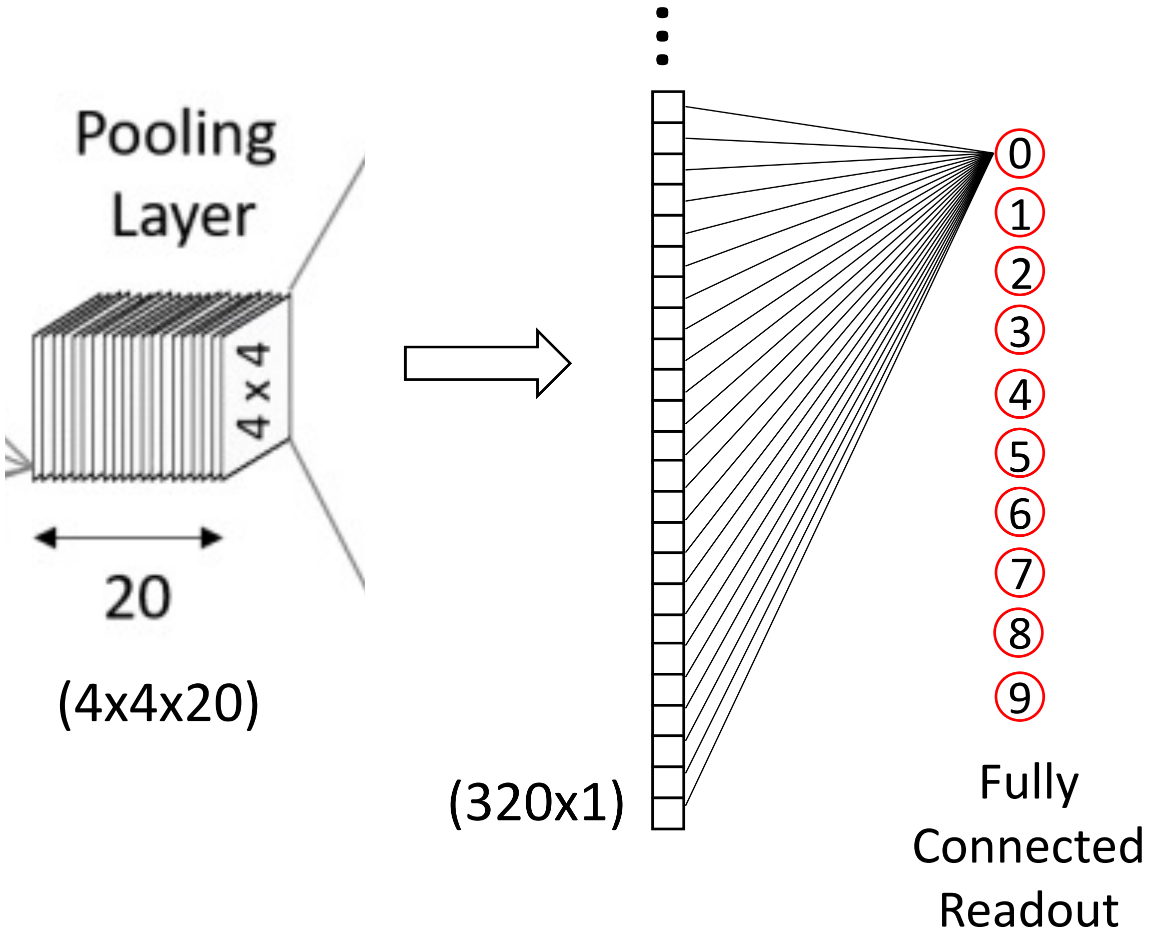
The most basic way is to have a final **fully connected** readout layer with as many neurons as there are classes



Fully Connected Layer

Fully connected means each neuron takes input from all neurons in the final set of feature maps

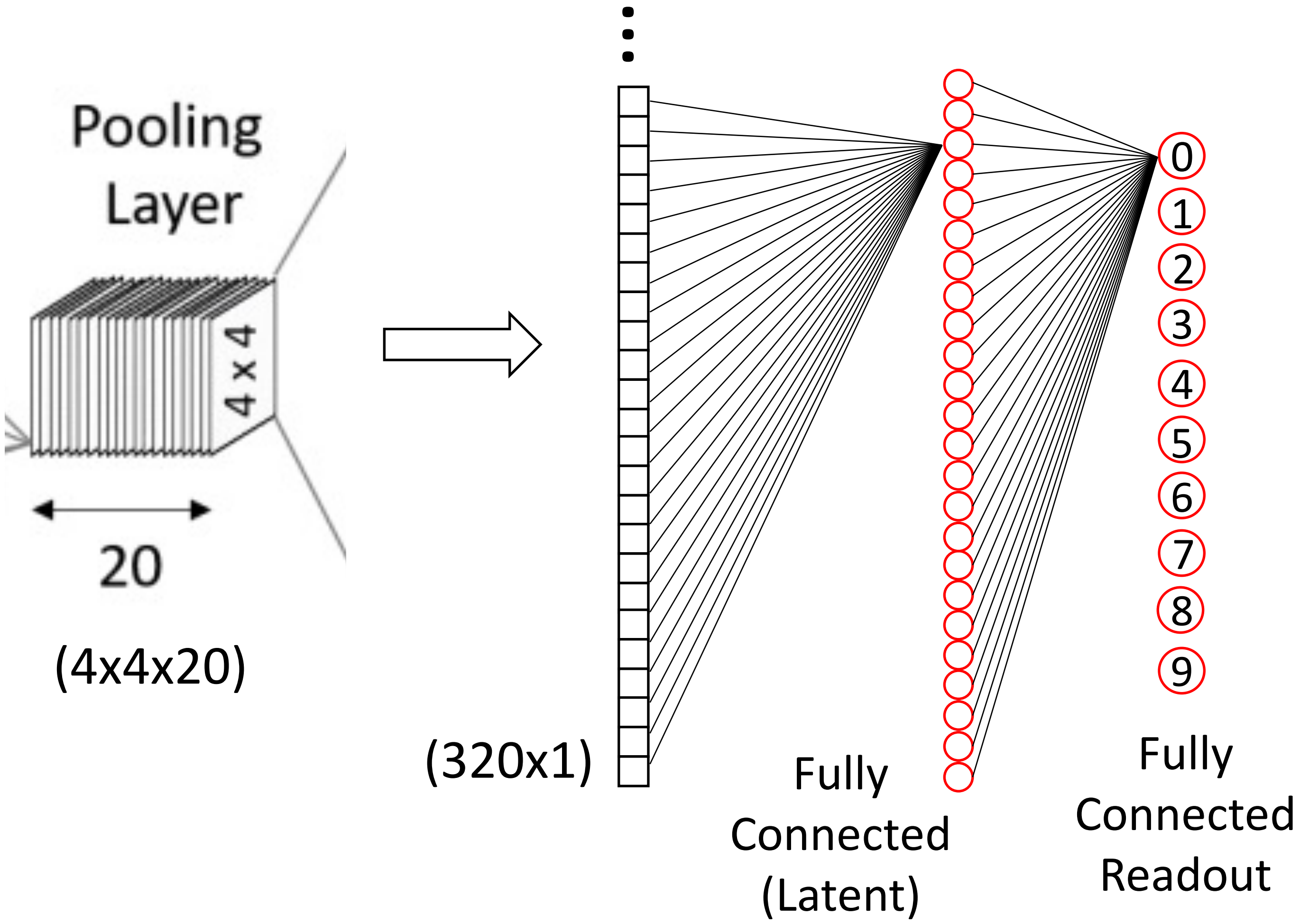
The final set of feature maps are vectorized to create an MLP-like configuration



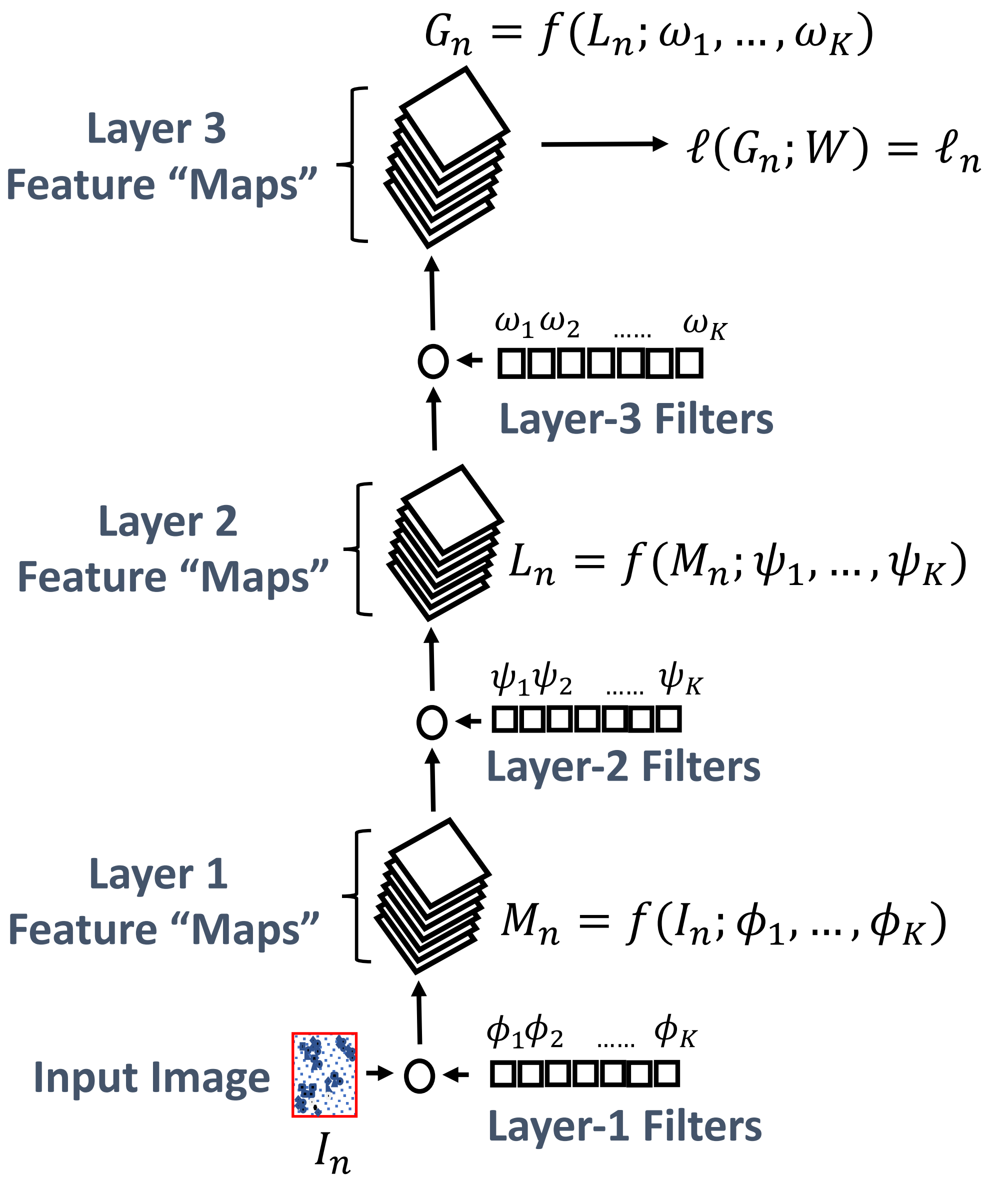
Fully Connected Layer

But it is also common to stack multiple fully connected layers before the final readout layer

Neurons in these intermediate layers can be considered **latent** classes



Review: Training A Deep Convolutional Neural Network



- 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:

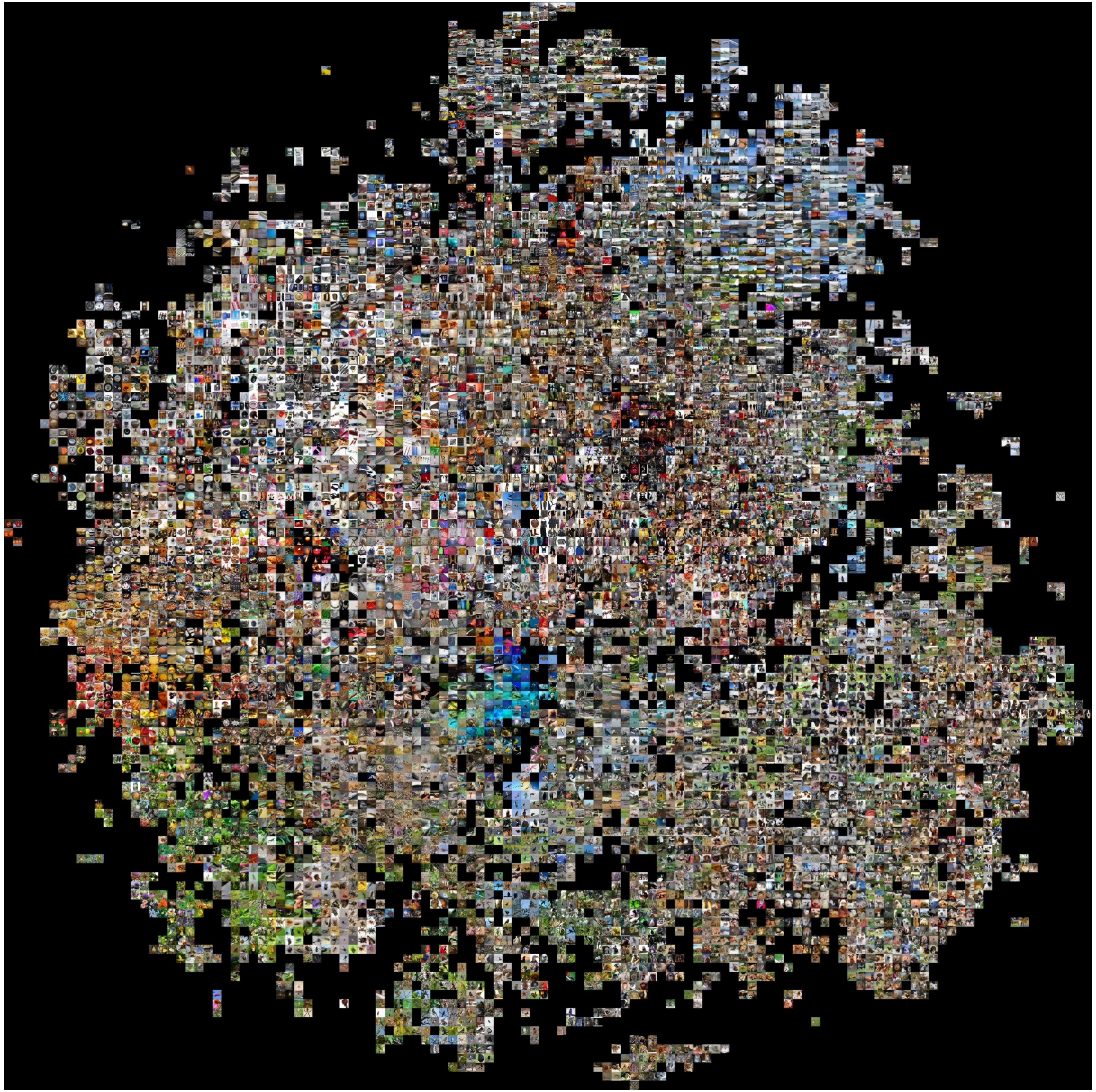
$$E(\Phi, \Psi, \Omega, W) = 1/N \sum_{n=1}^{N} loss(y_n, \ell_n)$$

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

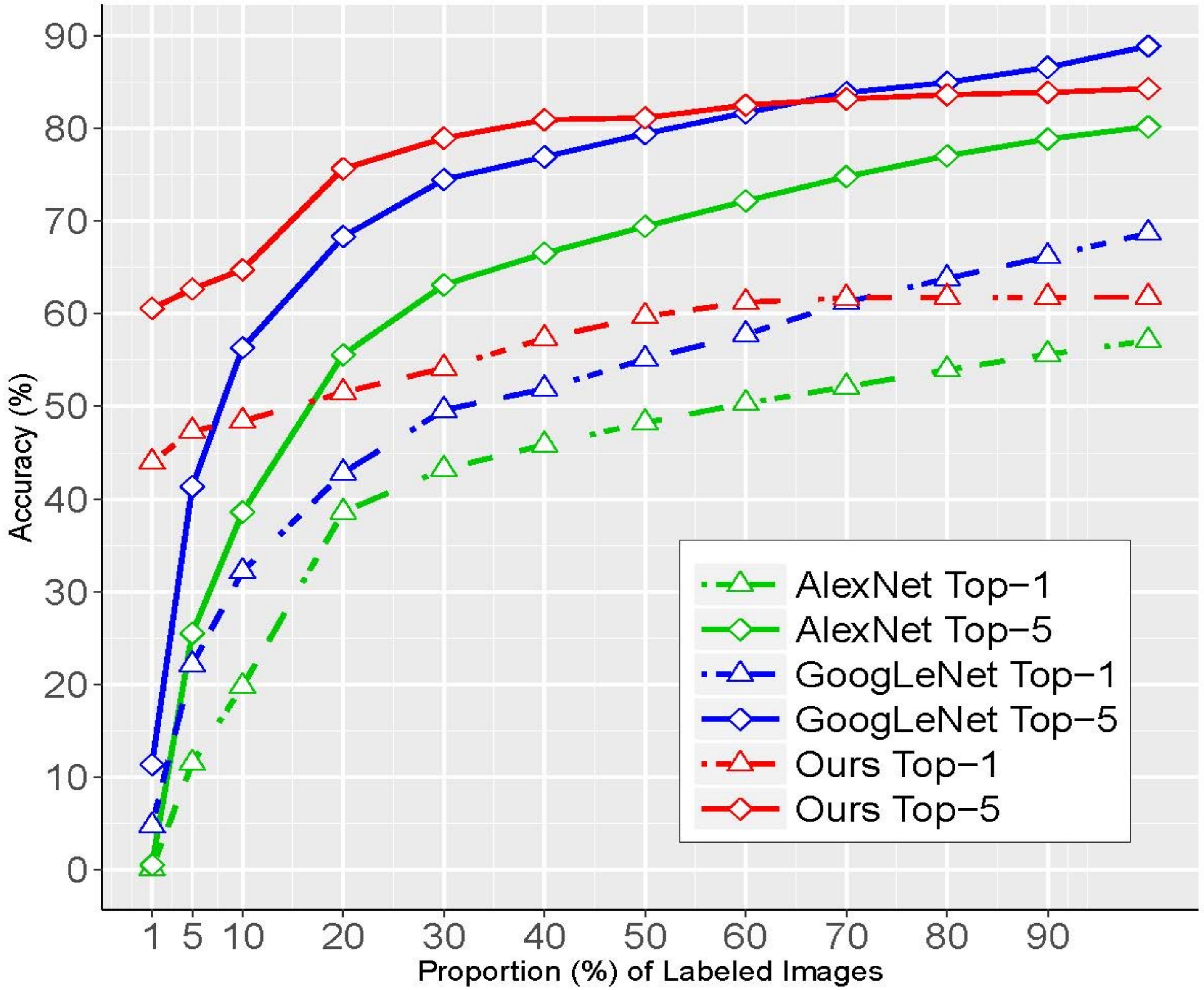
ImageNet Challenge

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

Example Images



ImageNet Challenge



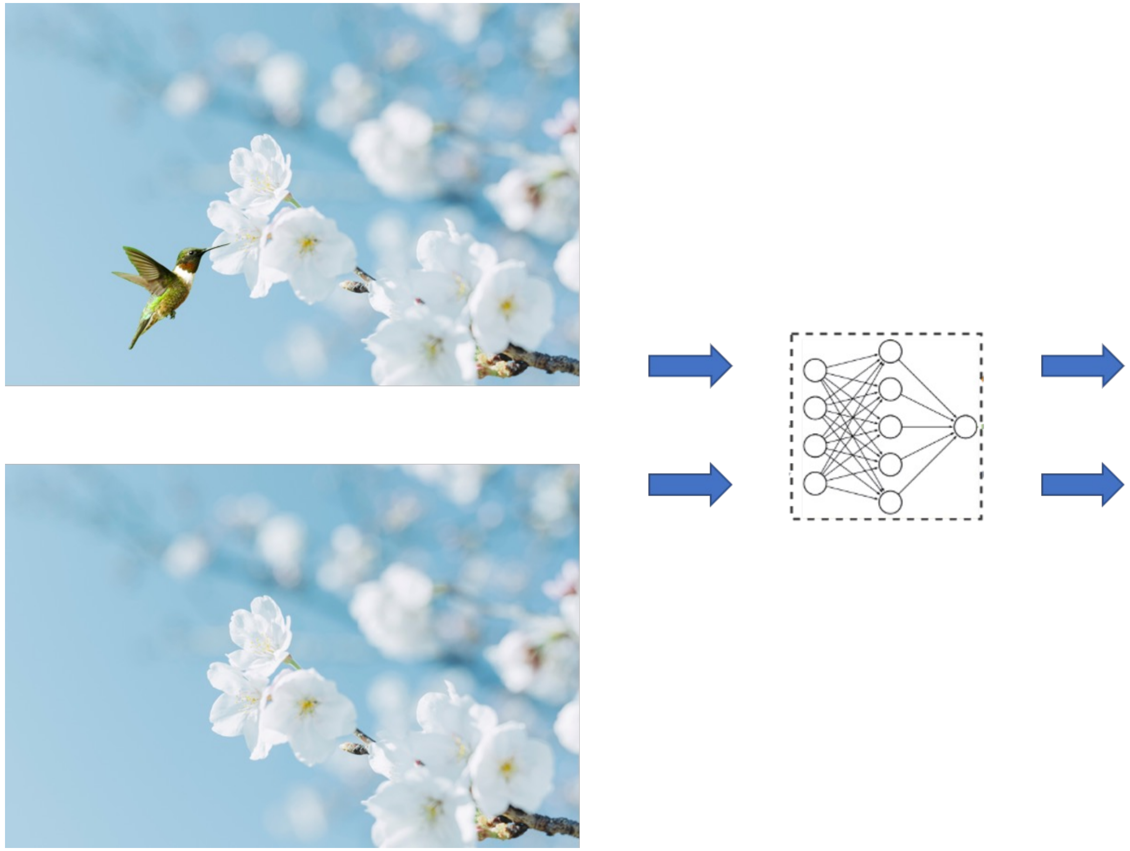
Explainability in ML Models

Goal
Vision models in many (*e.g.*, healthcare) applications will benefit from explainable predictions

Motivation
Explainable predictions help build trust in ML systems and may yield improved model performance

Approach
Leverage contrastive representation learning and architecture-agnostic saliency maps to identify input features relevant for prediction

Causality and Interventions

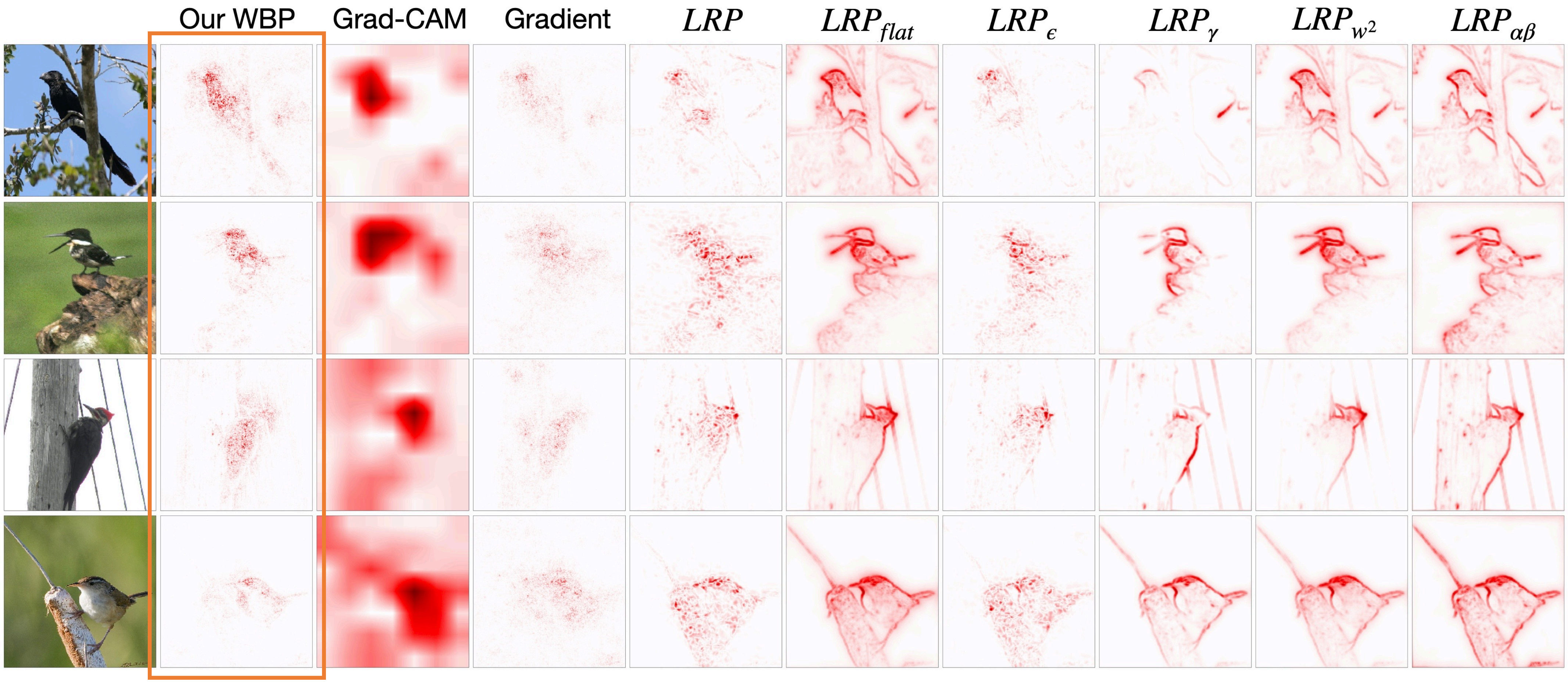


$p(\text{Bird}|\text{Image}) \rightarrow 1$

$p(\text{Bird}|\text{Image}) \rightarrow 0$

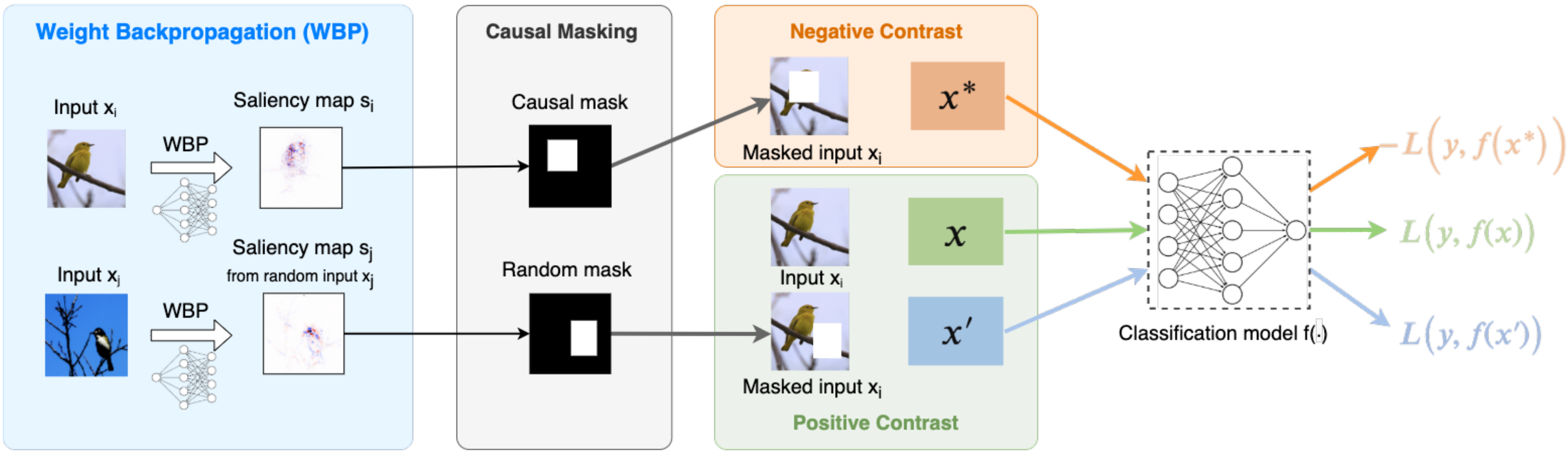
Creating datasets with real interventions is very difficult!

Pseduo Interventions



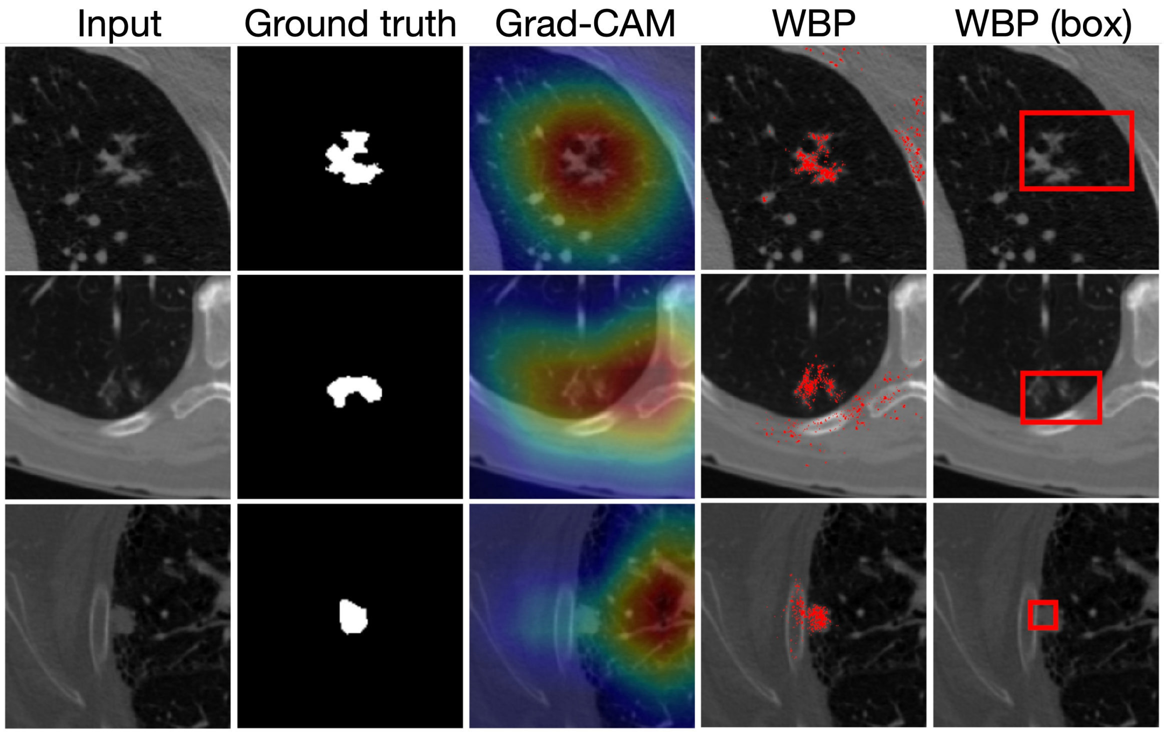
We can use the model (saliency maps) to generate pseudo interventions

Proactive Pseudo Interventions



Algorithm: saliency maps → pseudo interventions → better saliency maps → repeat

Lung Lesion (Nodule) Classification



Models	AUC
Tensor Net-X [15]	0.823
DenseNet [27]	0.829
LoTeNet [50]	0.874
Inception_v3 [58]	0.921
$+\mathrm{PPI}_{GradCAM}$	0.933
$+\mathrm{PPI}_{Gradient}$	0.930
$+\mathrm{PPI}_{LRP}$	0.931
$+\mathrm{PPI}_{WBP}$	0.935
$+\mathrm{PPI}_{WBP(box)}$	**0.941**

Inception_v3 + PPI outperforms Inception_v3 and produces meaningful saliency maps

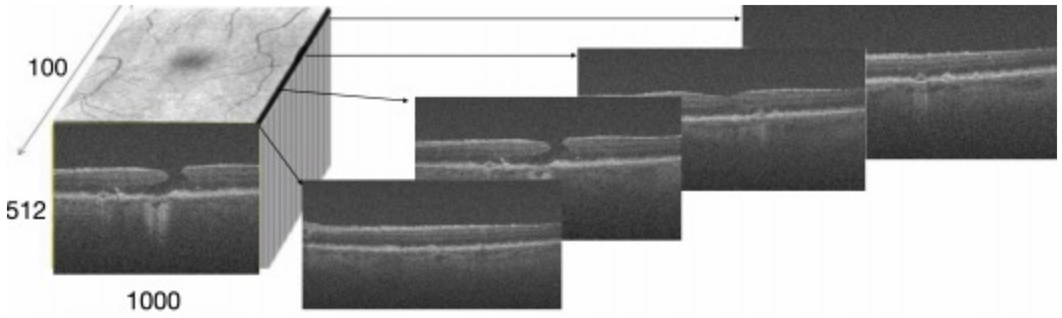
1-Year Geographic Athropy Conversion

Problem

- Geographic atrophy (GA) is the severe nonexudative form of age-related macular de-generation (AMD)
- GA results in irreversible loss of visual function and affects 5M people worldwide
- No treatments have been approved due to the lack of understanding of the progression of AMD

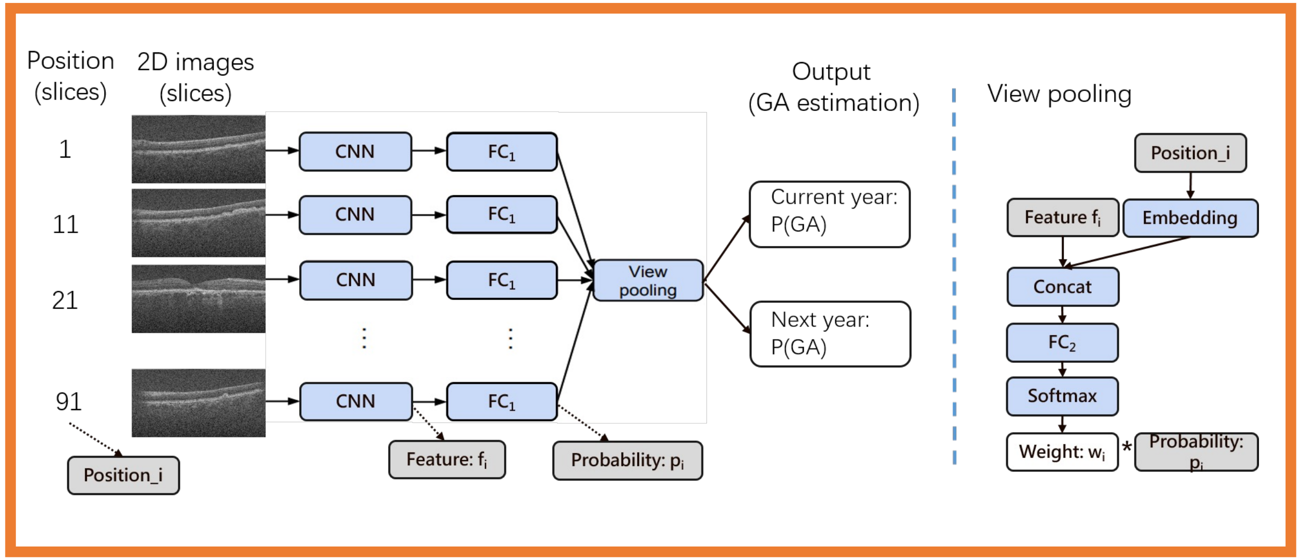
Cohort

- A2A SD-Study (NCT00734487): 316 participants
- SD-OCT (6.7mm x 6.7mm) volumes, 275 participants with an average of 4 yearly volumes were used for the analysis

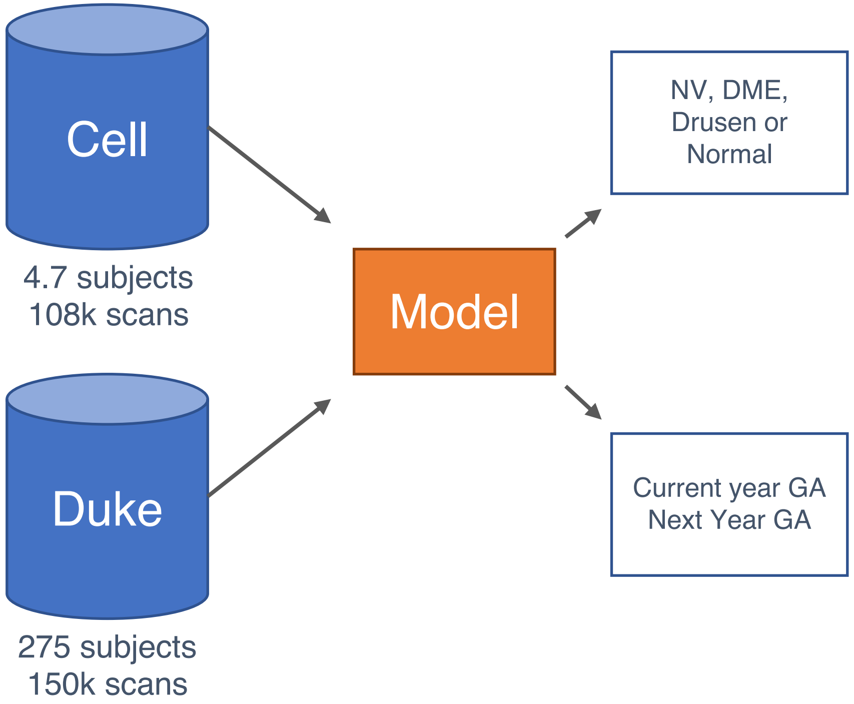


Volumetric OCT (Optical Coherent Tomography):
512 pixels (height)
1000 pixels (width)
100 (scans)

Model Architecture



Model

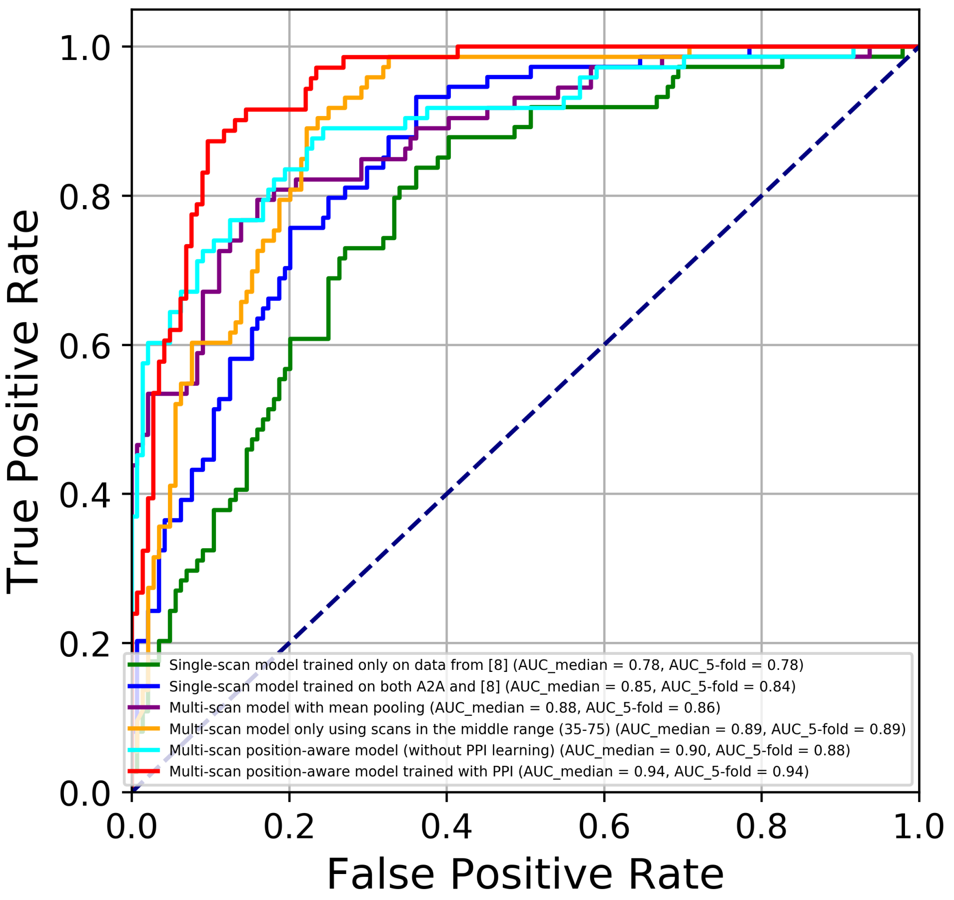


Cell: Kermany *et al.* Cell, 2018

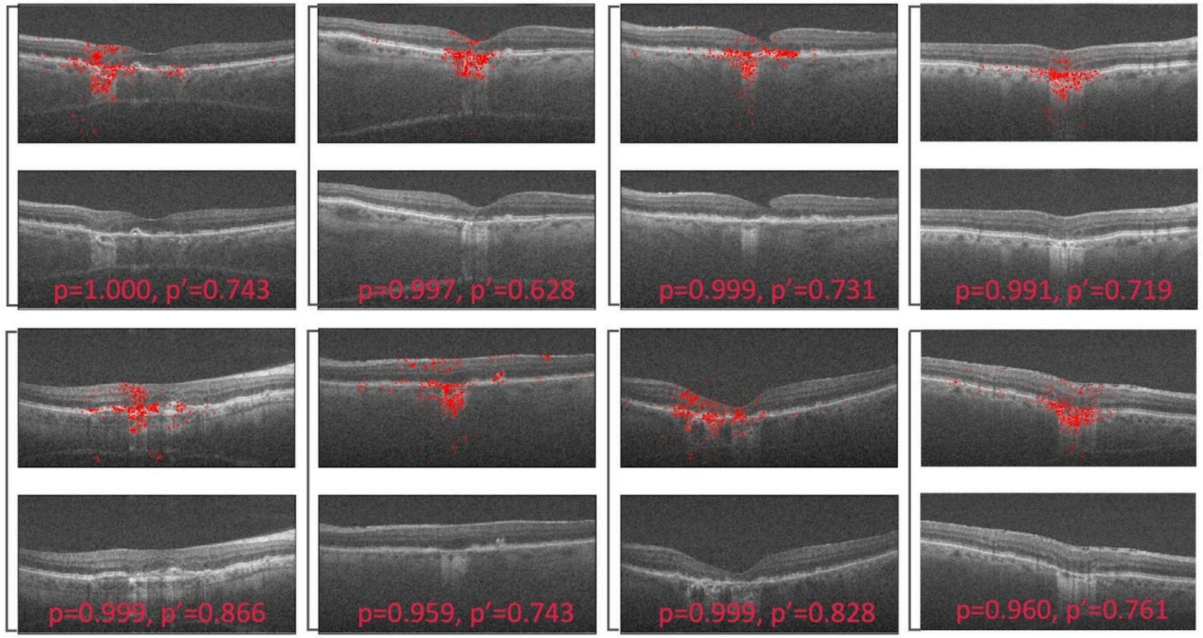
Performance Characteristics

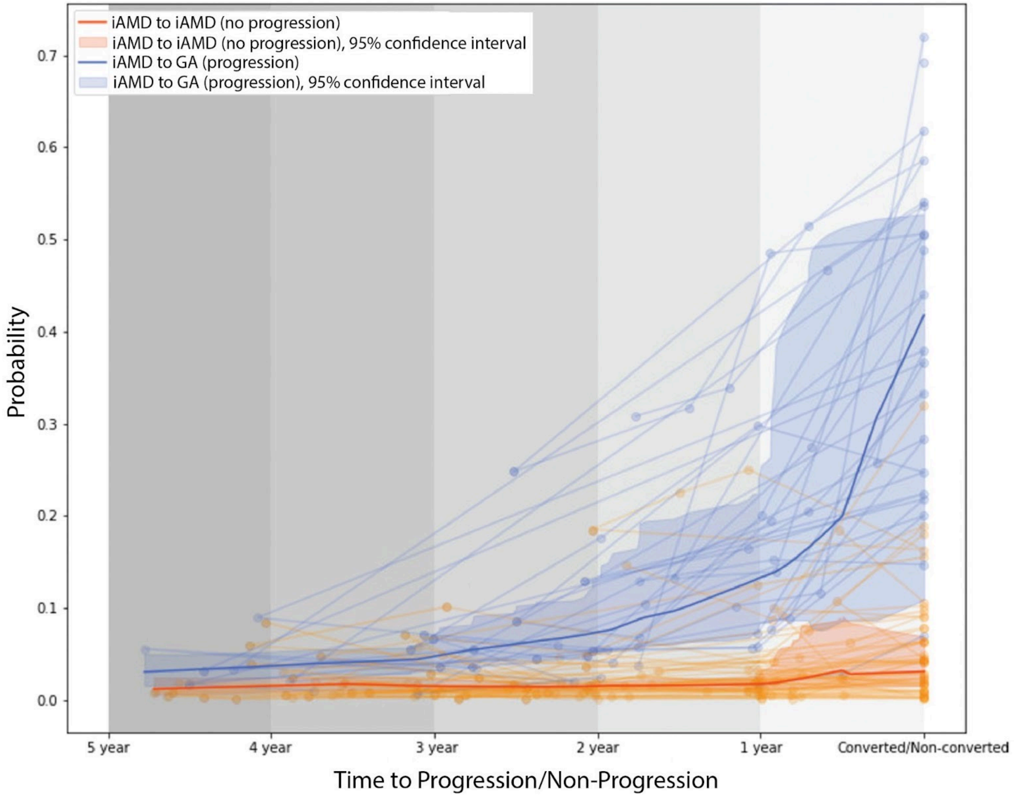
Model	AUC (SD)
Multi-scan position-aware model trained with PPI and quantitative/qualitative measurements	**0.945 (0.018)**
Multi-scan position-aware model trained with PPI	0.937 (0.017)
Multi-scan position-aware model (without PPI learning)	0.877 (0.040)
Multi-scan model only using scans in the middle range (35-75)	0.890 (0.047)
Multi-scan model with mean pooling	0.862 (0.030)
Single-scan model trained on both A2A and [Cell]	0.840 (0.030)
Single-scan model trained only on data from [Cell]	0.781 (0.026)
Single-scan model trained on ImageNet	0.650 (0.024)
Logistic regression model on quantitative/qualitative measurements	0.856 (0.033)

[Cell] Kermany *et al.* Cell, 2018



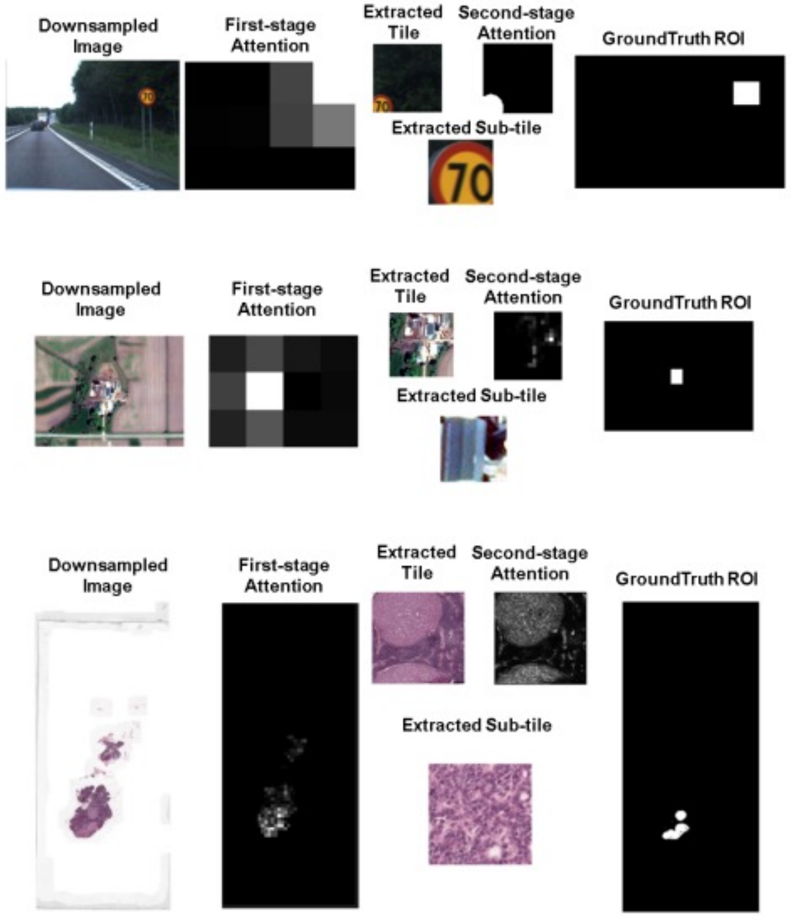
Generalization and Explainability



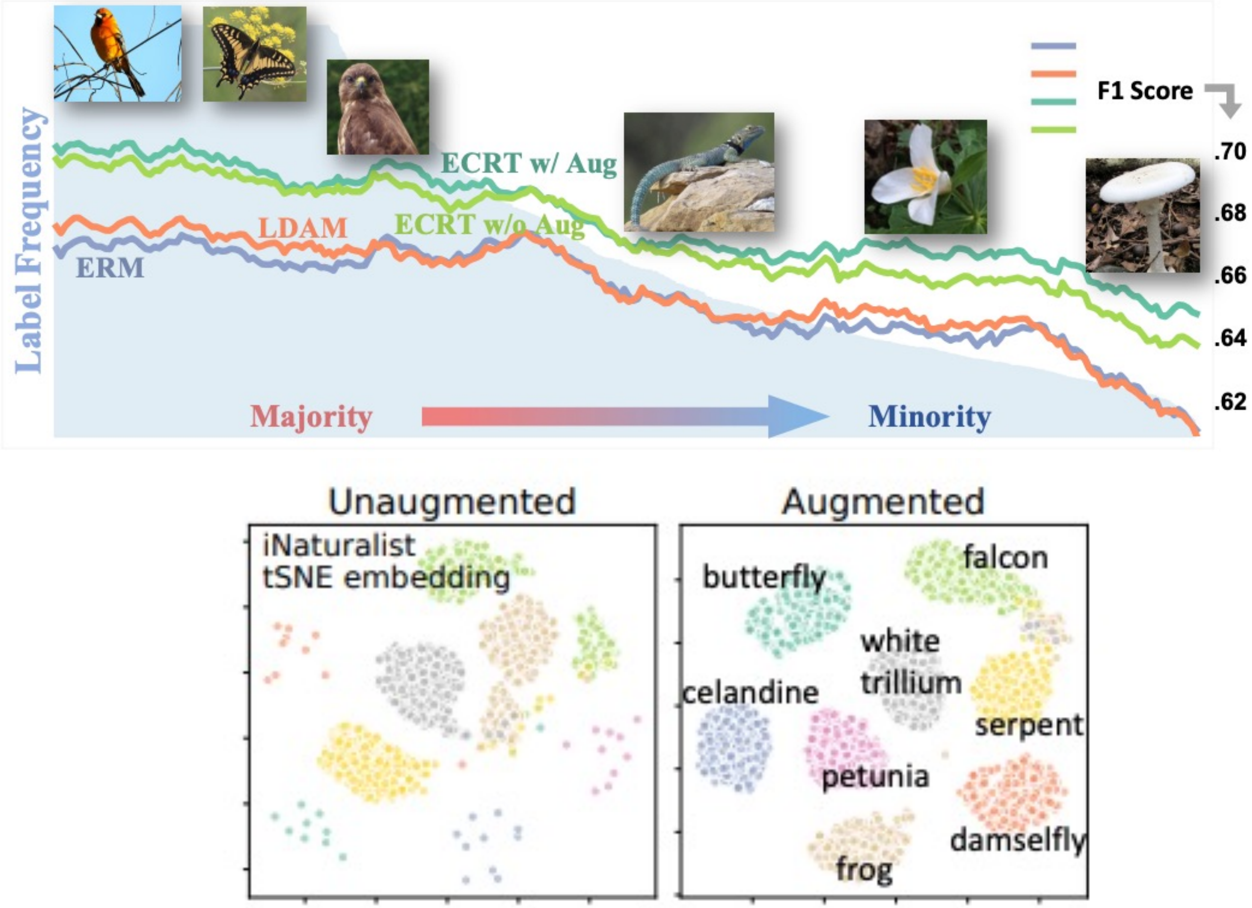


The model generalizes to standard of care data (years 2012 to 2022)

Extensions



Explanations for gigapixel images

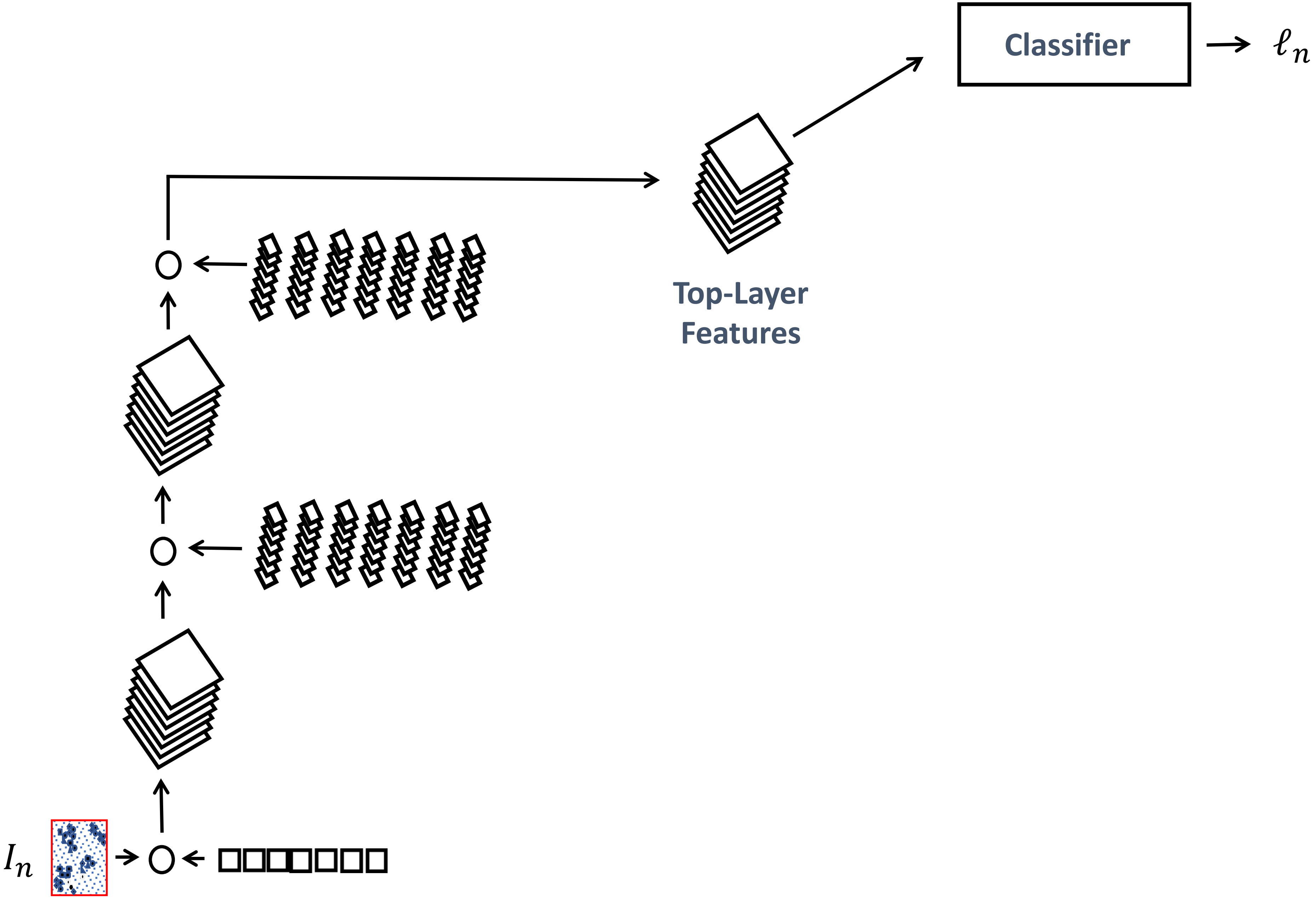


Augmentations for imbalanced data

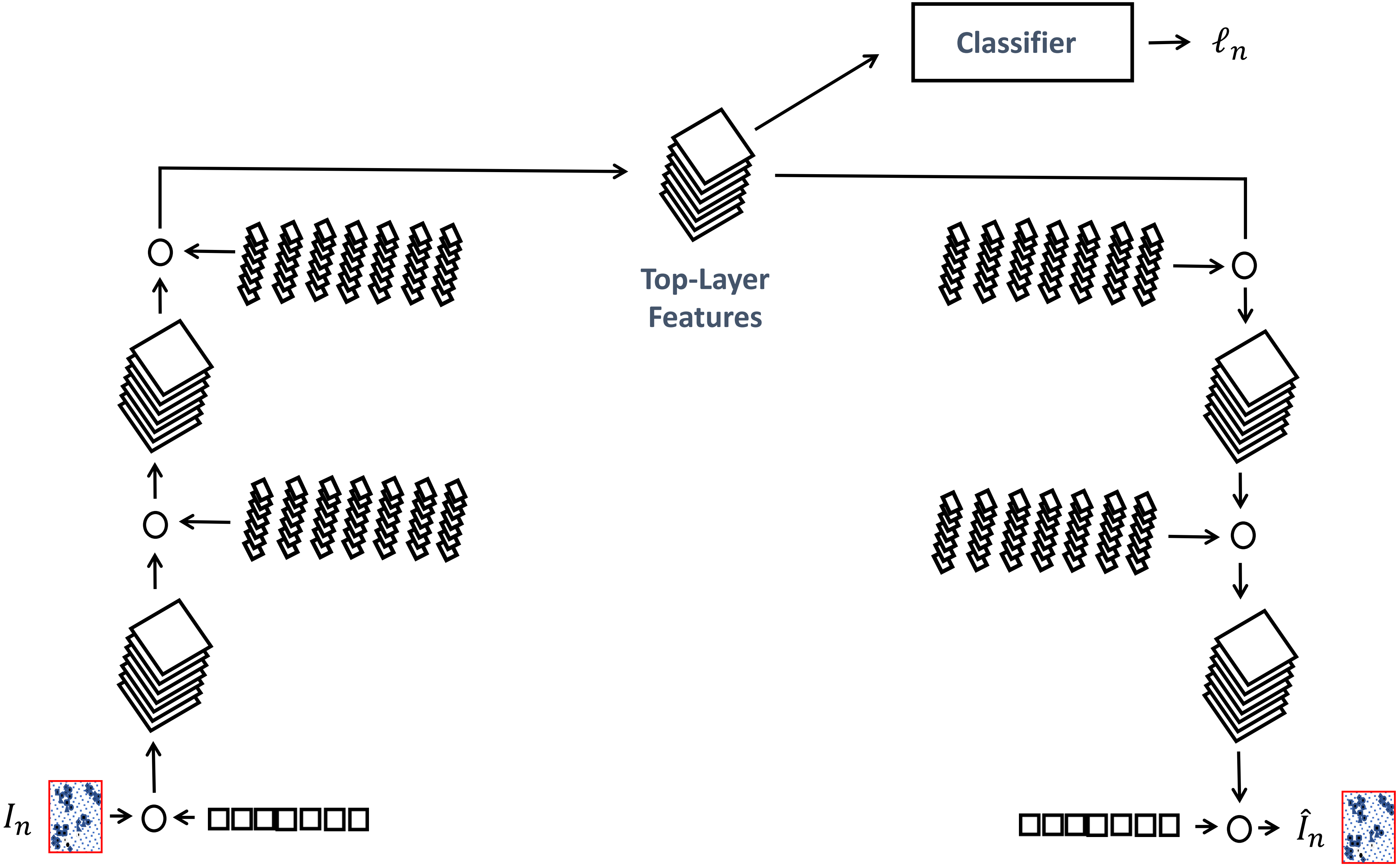
Summary

- 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 risk function
- Convolutional neural networks have shown capabilities beyond human performance for image analysis

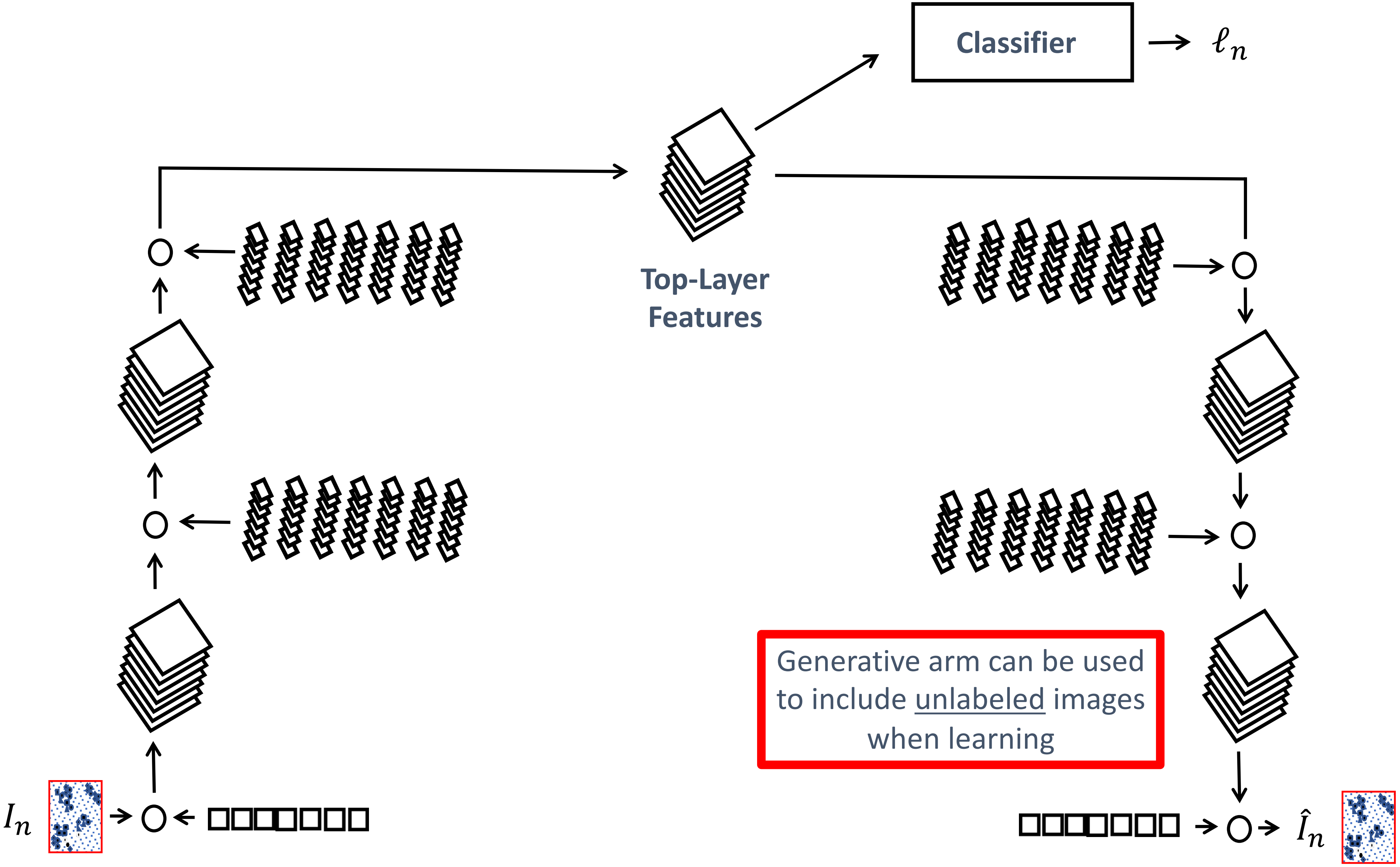
Deep Architecture



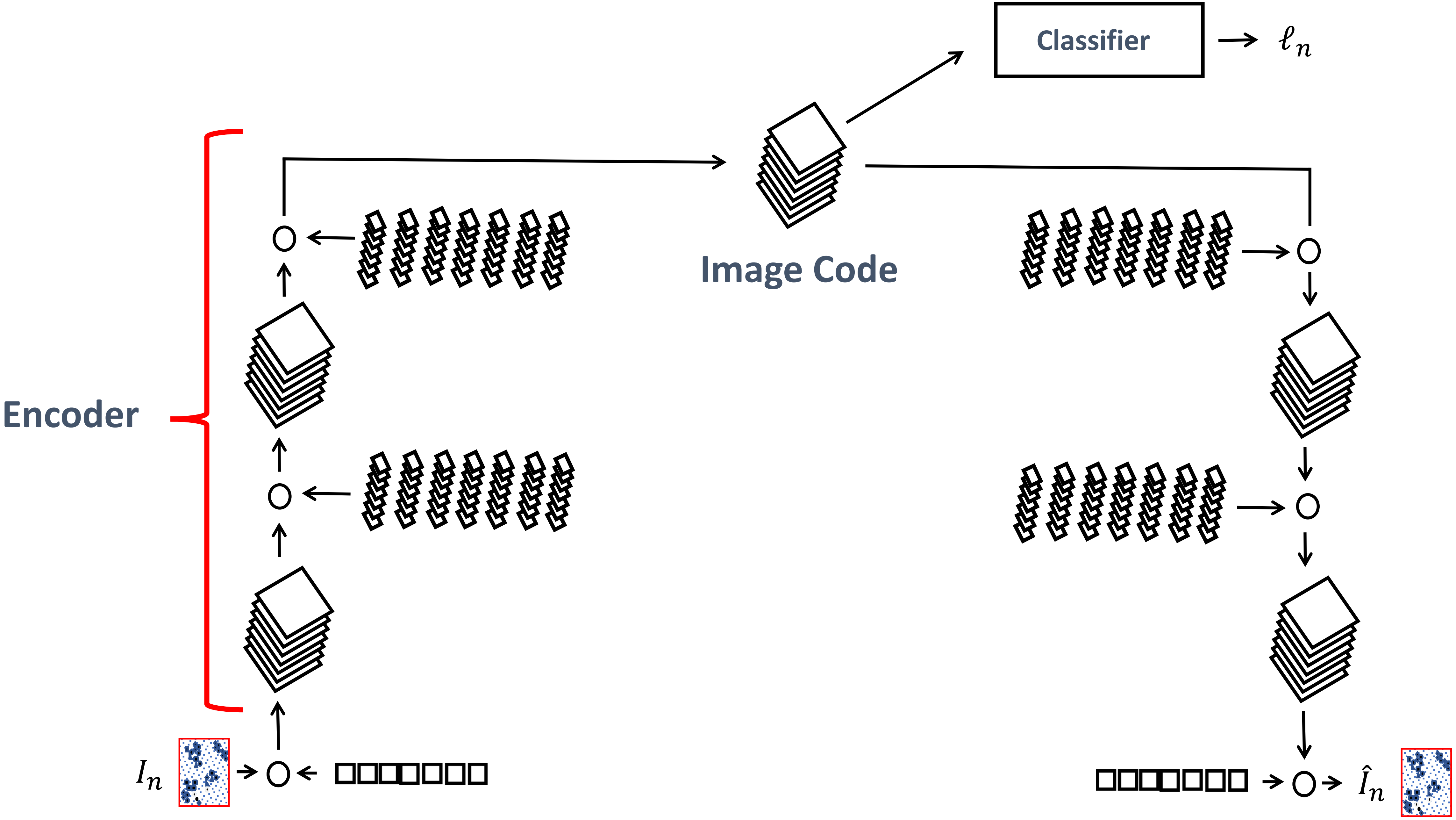
Add Generative Arm



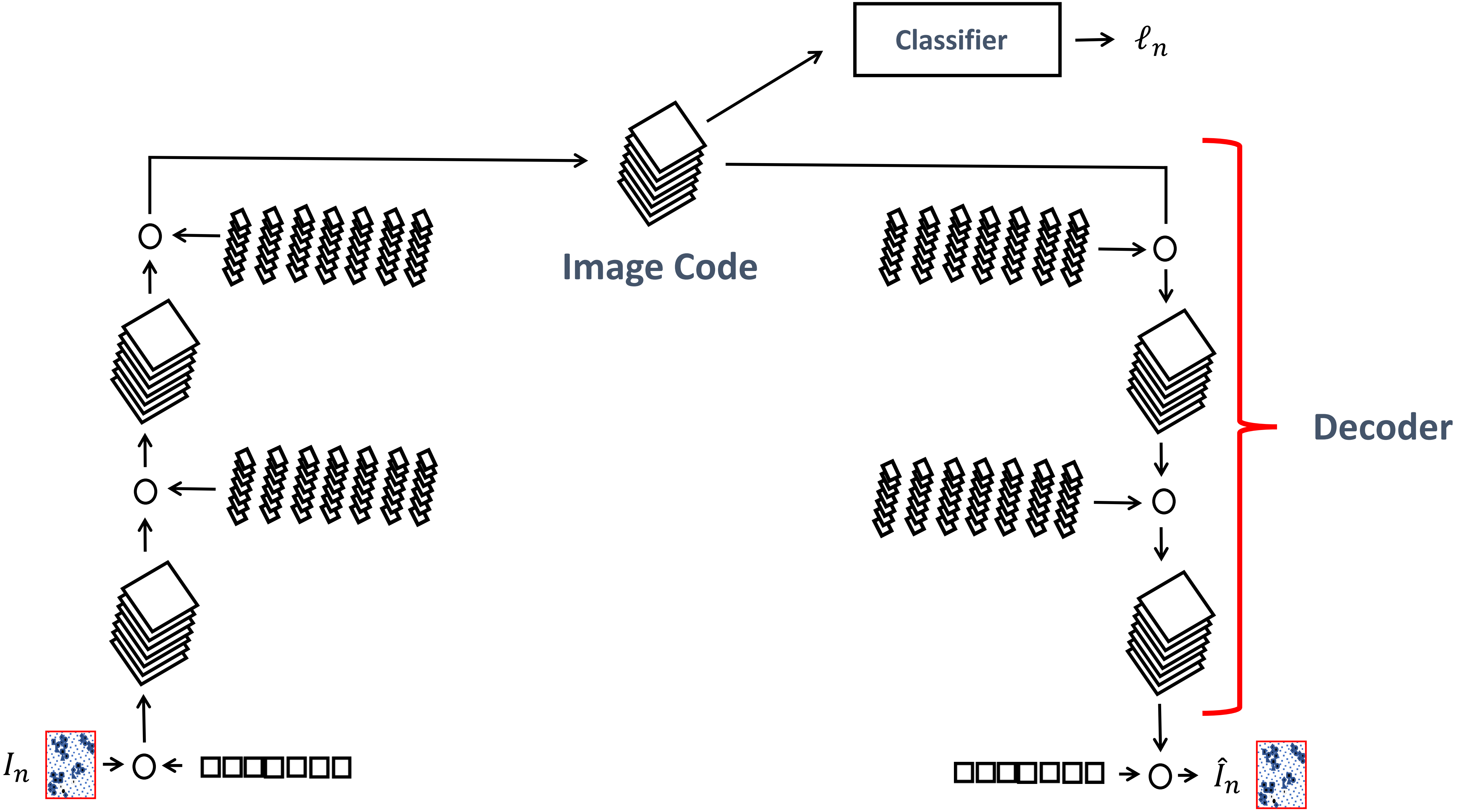
Add Generative Arm



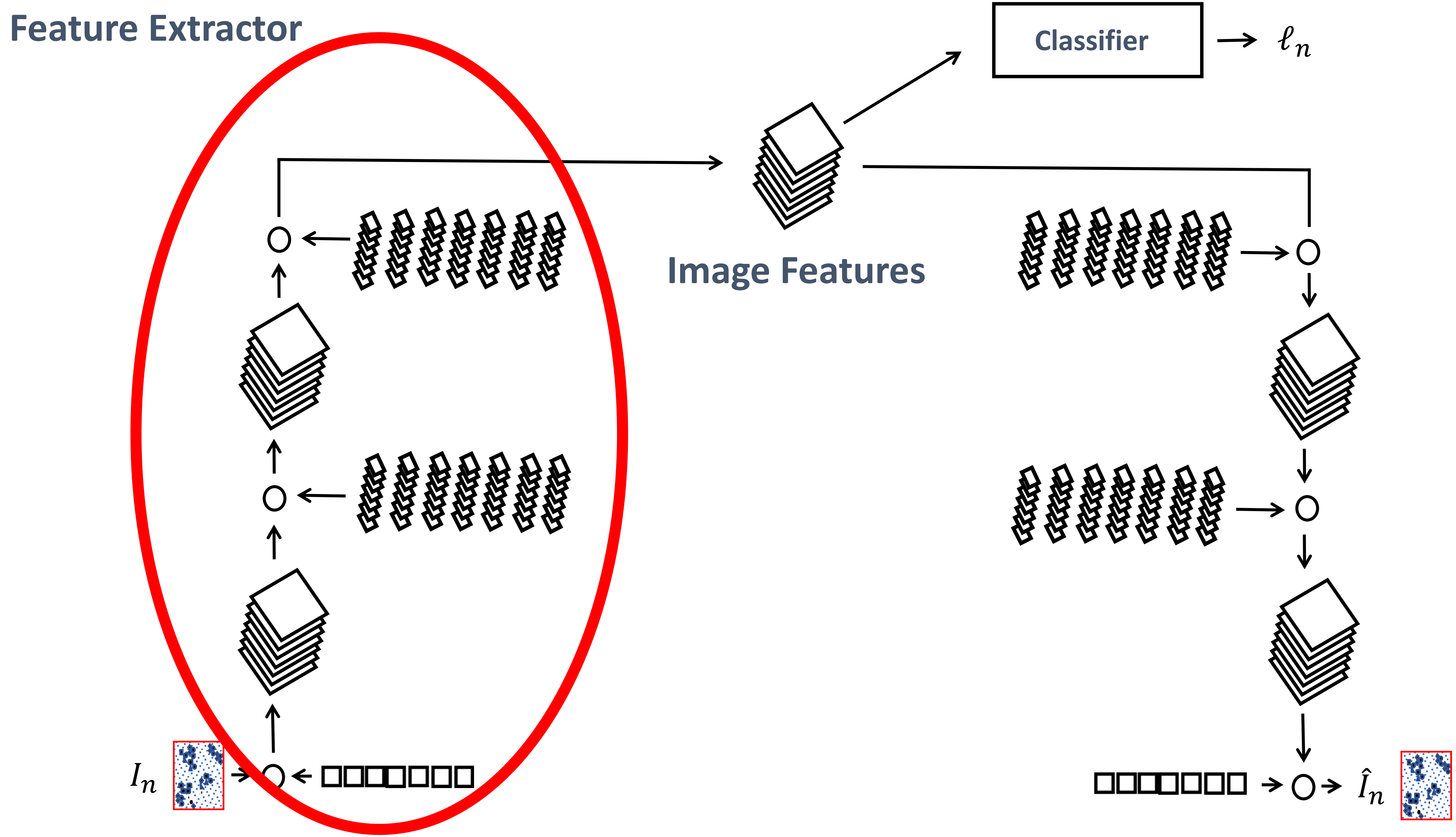
Deep Analysis Architecture



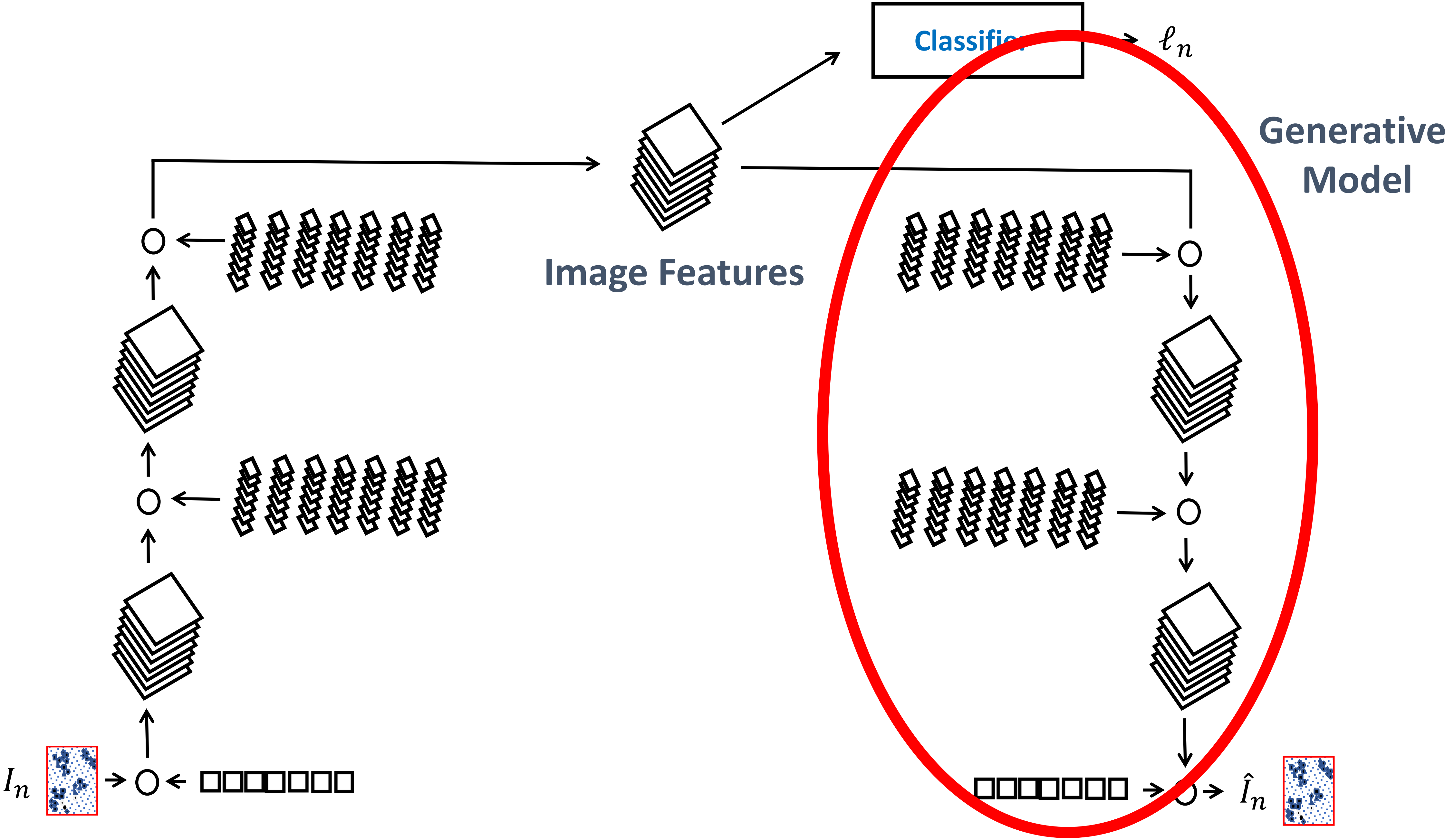
Deep Analysis Architecture



Deep Analysis Architecture



Deep Analysis Architecture



Deep Analysis Architecture