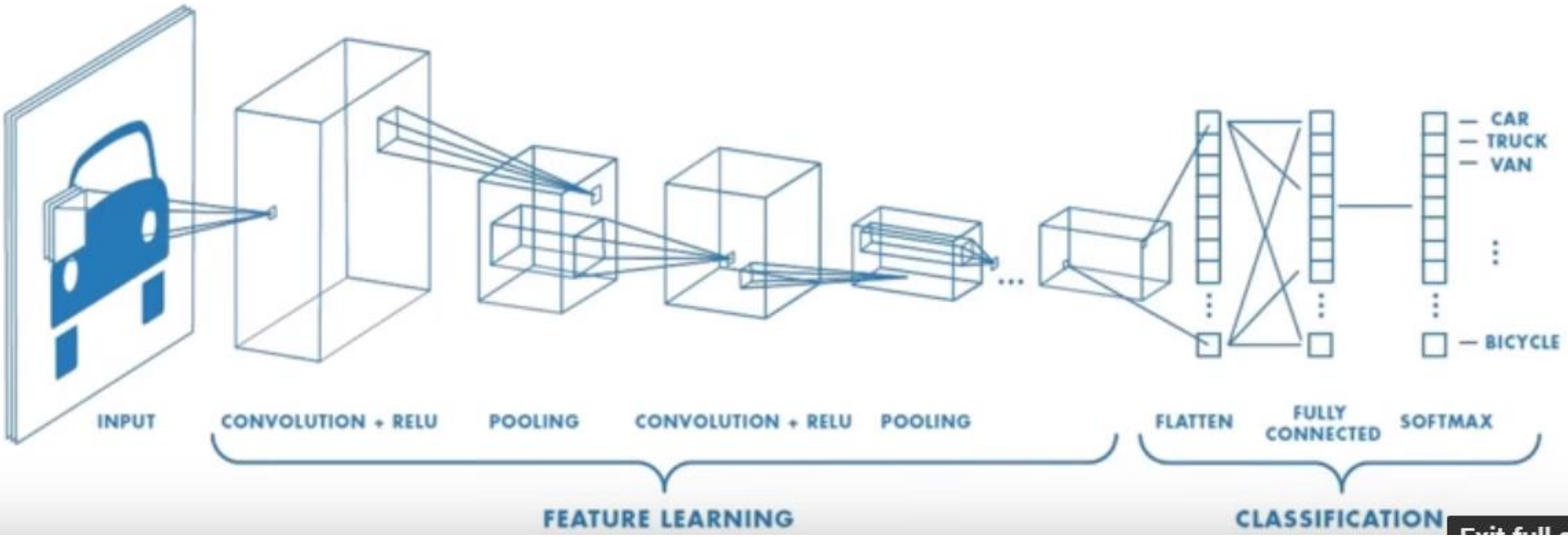
CNN

Dr. Kisor Kumar Sahu

SMMME, IITBBS

Convolutional Neural Network



Visual cortex

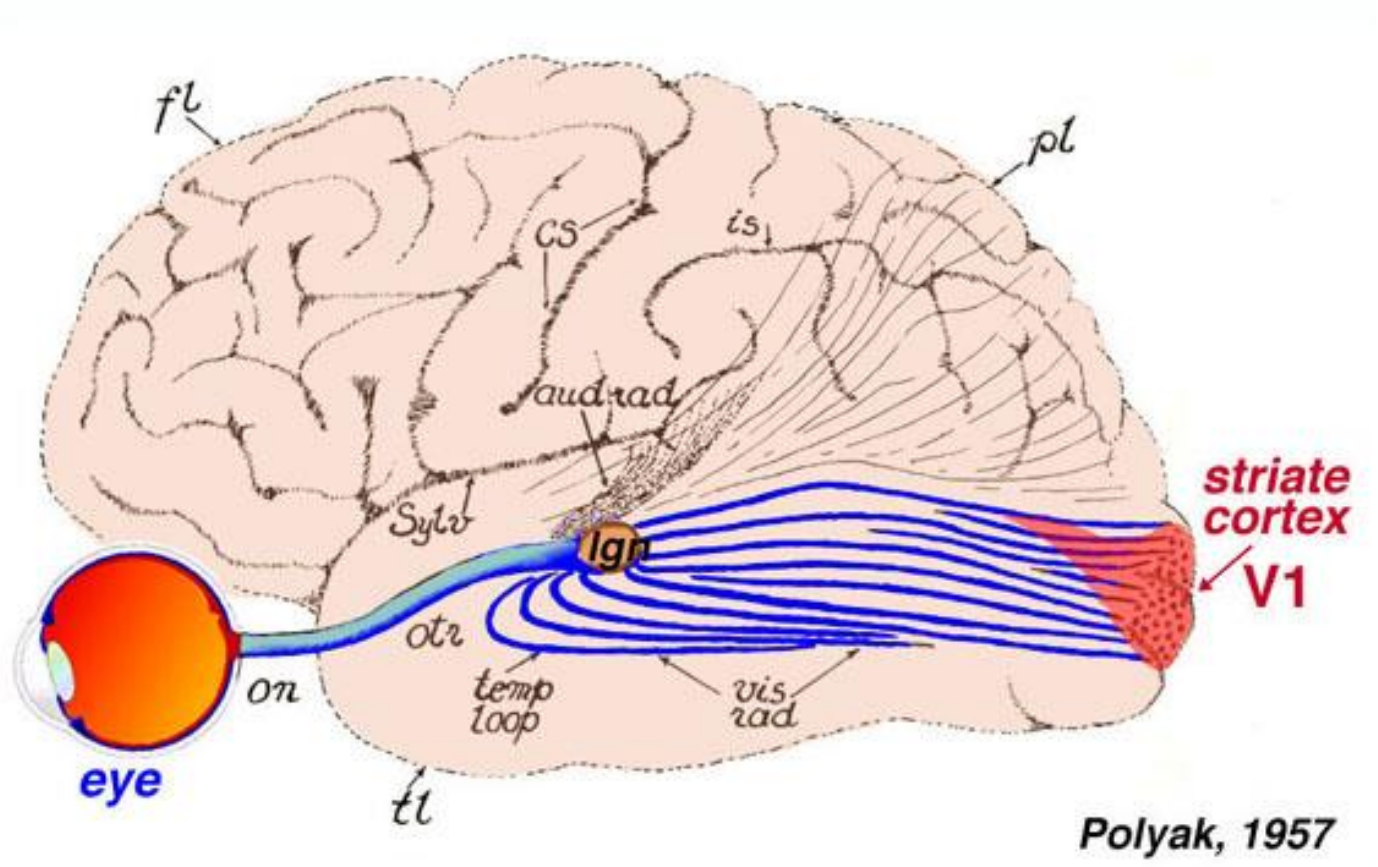
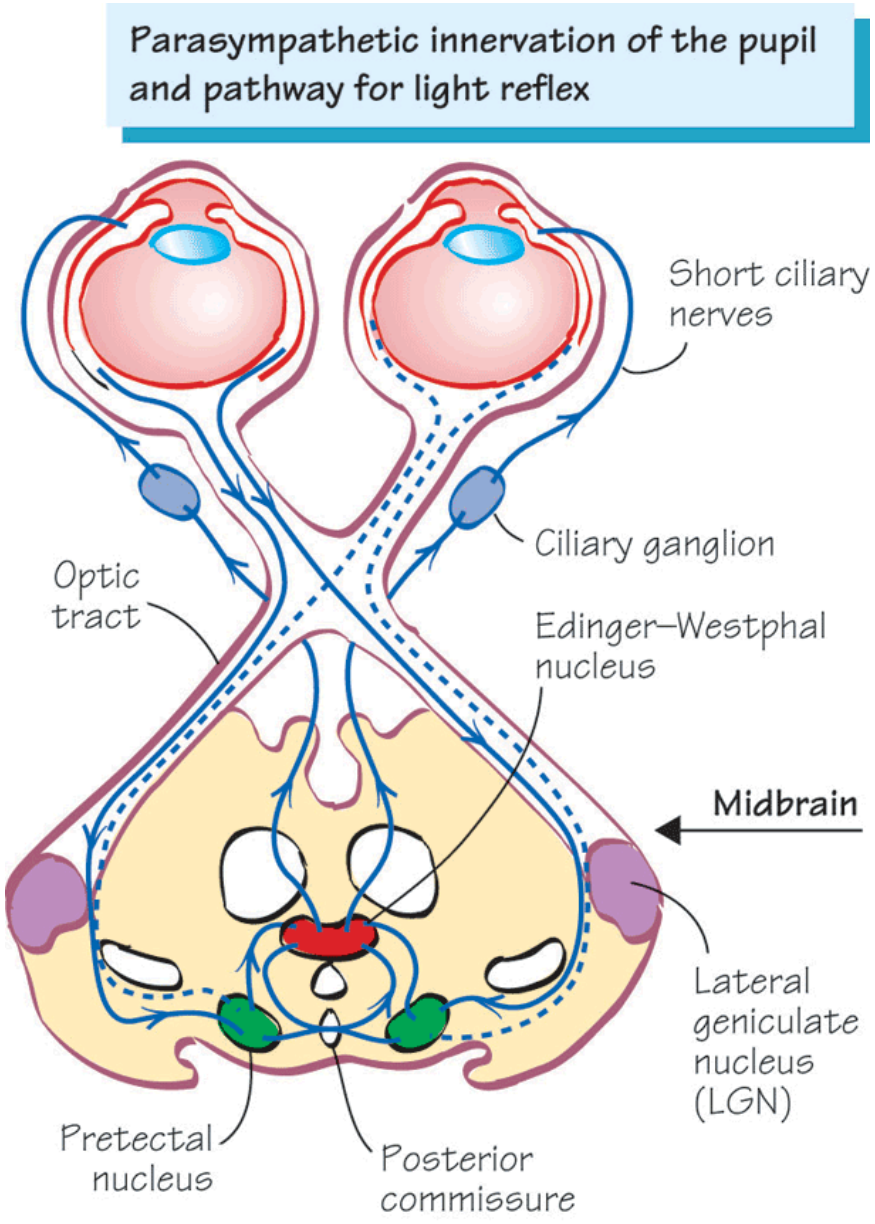
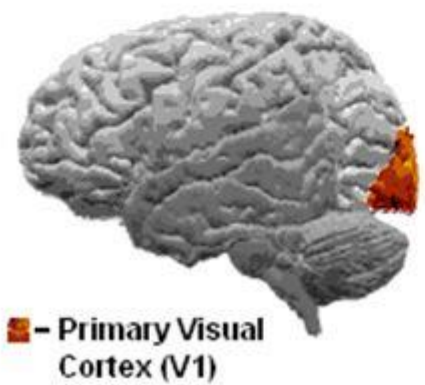


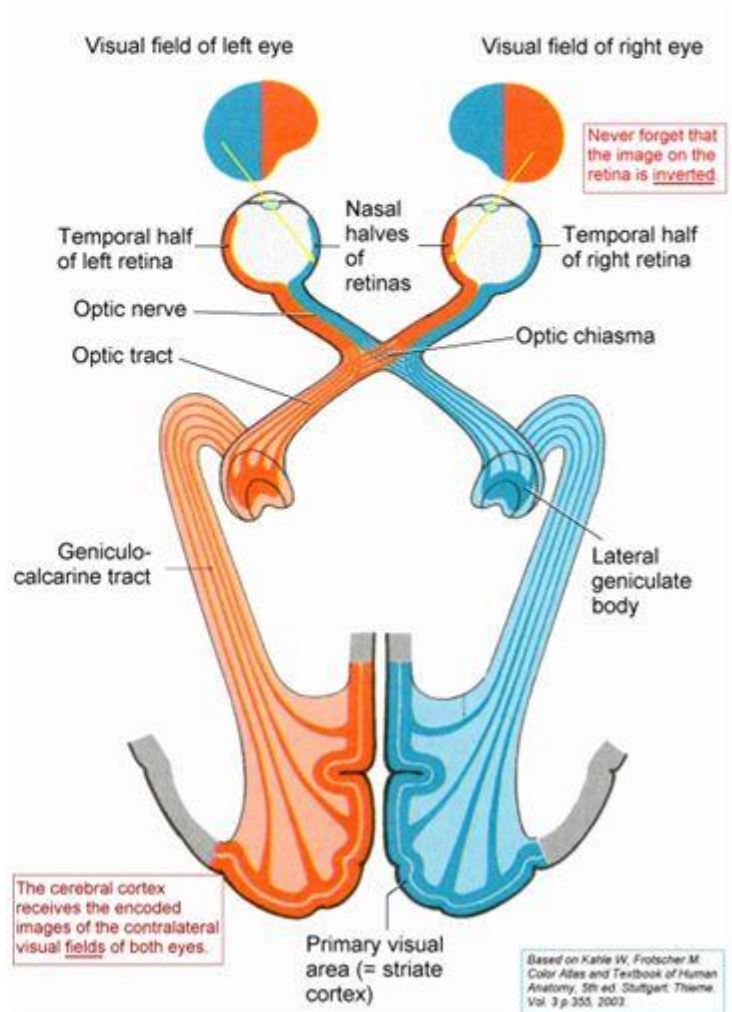
Figure 8. Visual input to the brain goes from eye to LGN and then to primary visual cortex, or area V1, which is located in the posterior of the occipital lobe. Adapted from Polyak (1957).

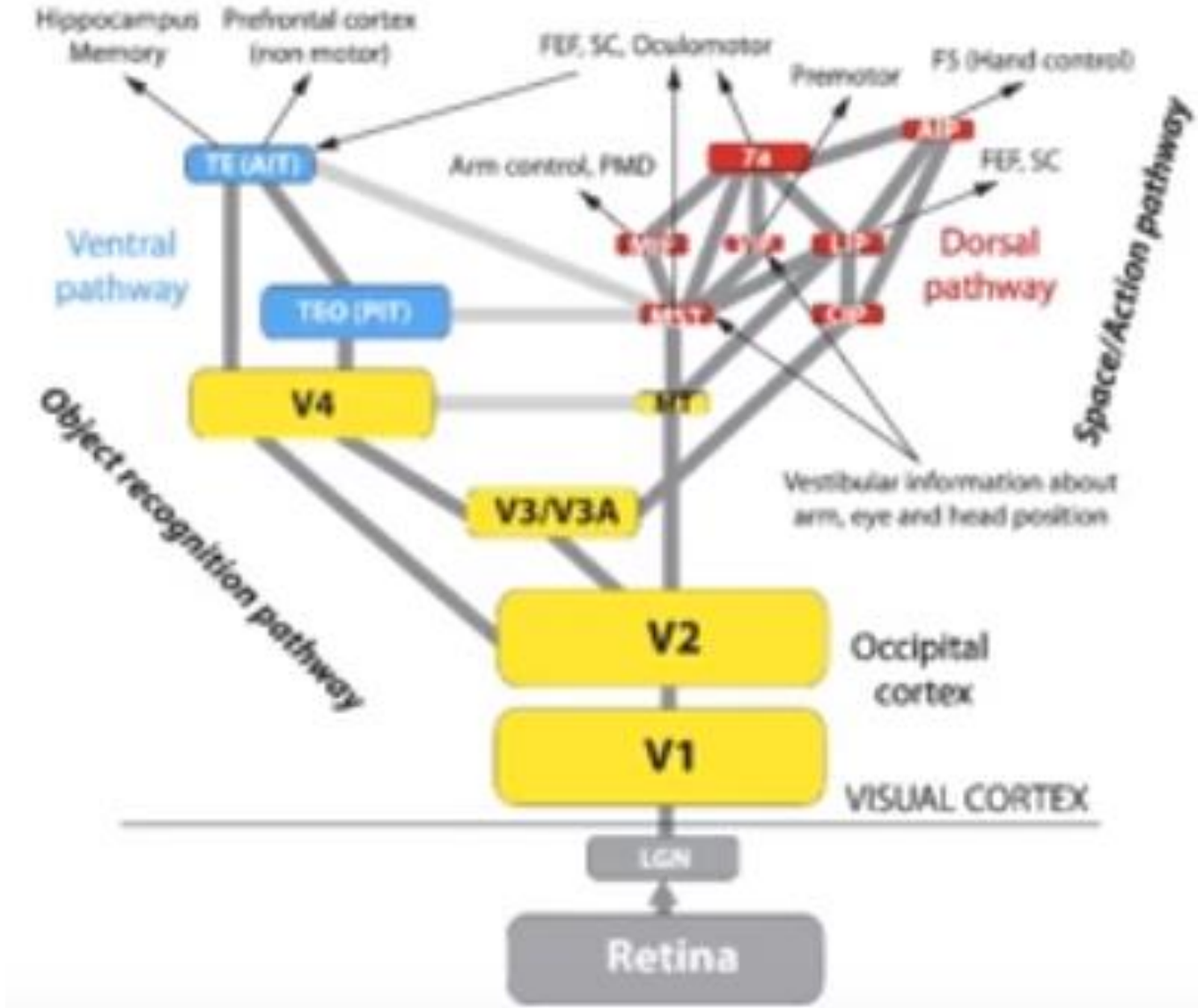


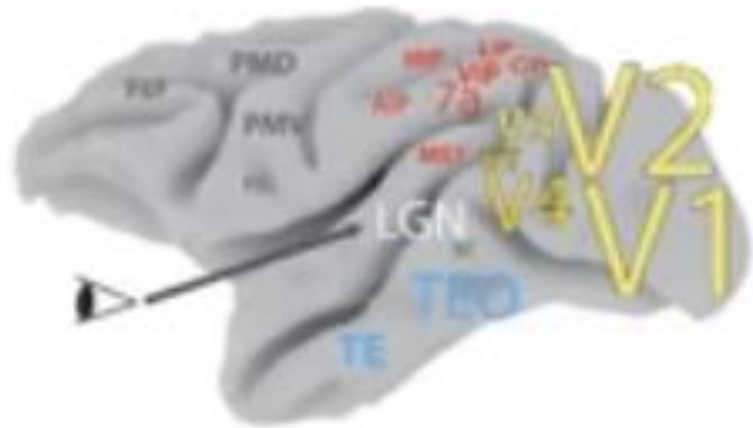
The Primary Visual Cortex

- A multi-layered structure located in the occipital lobe (AKA, Area 17, V1, Striate cortex).
- Receives axons from the LGN.
- Each hemisphere represents the contralateral visual field.

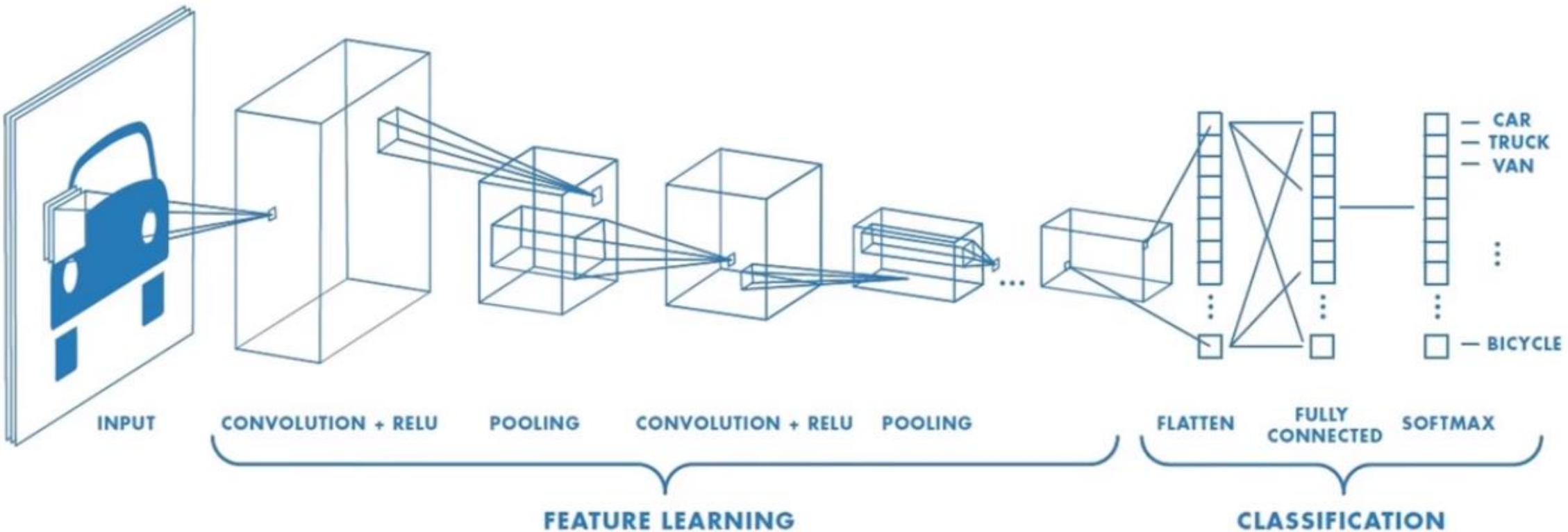


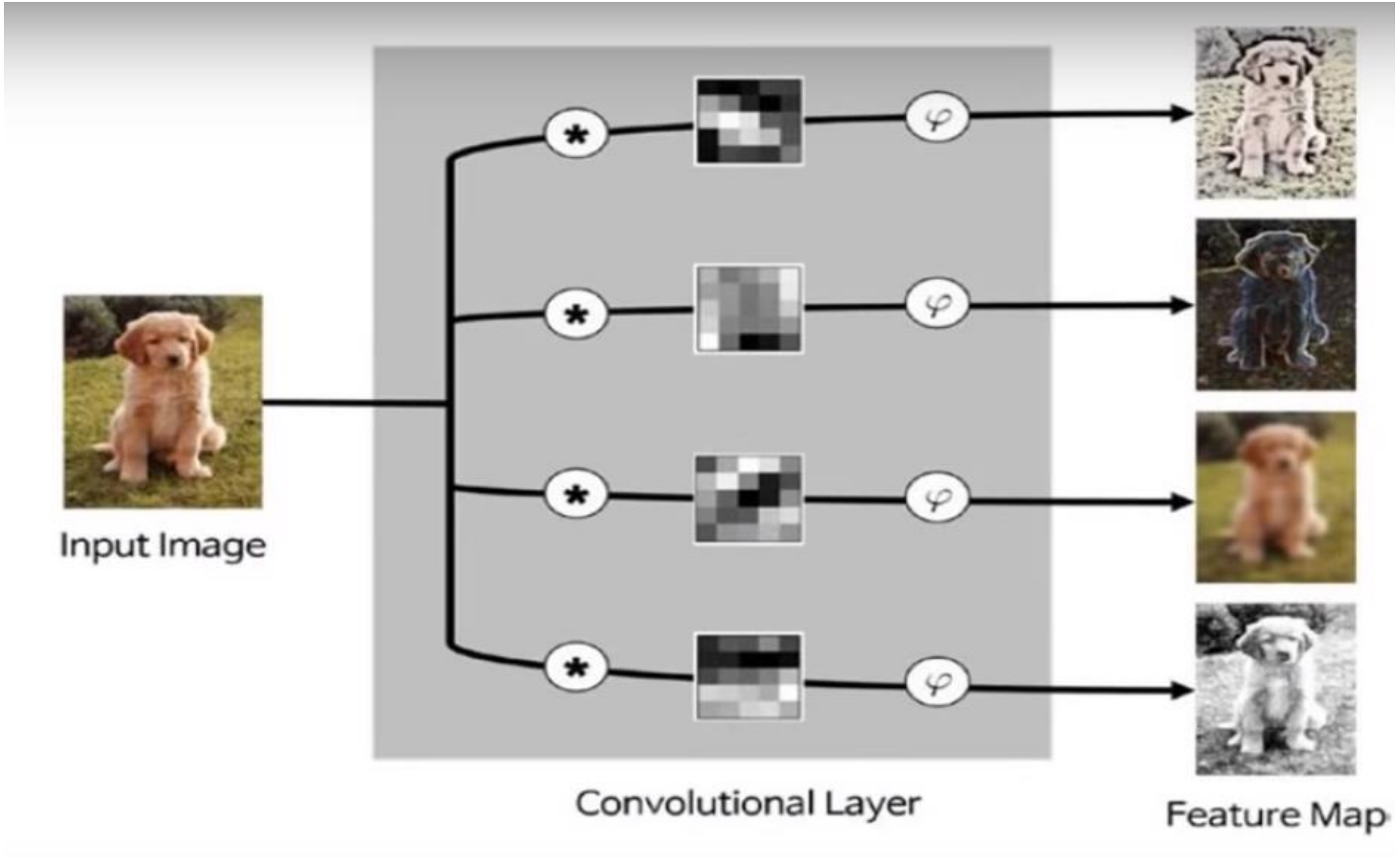



Hippocampus
Memory
Prefrontal cortex
(non motor)
FEF, SC, Oculomotor
Premotor
F5 (Hand control)
TE (AIT)
Arm control, PMD
7a
FEF, SC
Ventral
pathway
Dorsal
pathway
Space/Action pathway
TEO (PIT)
V4
MT
Object recognition pathway
Vestibular information about
arm, eye and head position
V3/V3A
V2
Occipital
cortex
V1
VISUAL CORTEX
LGN
Retina

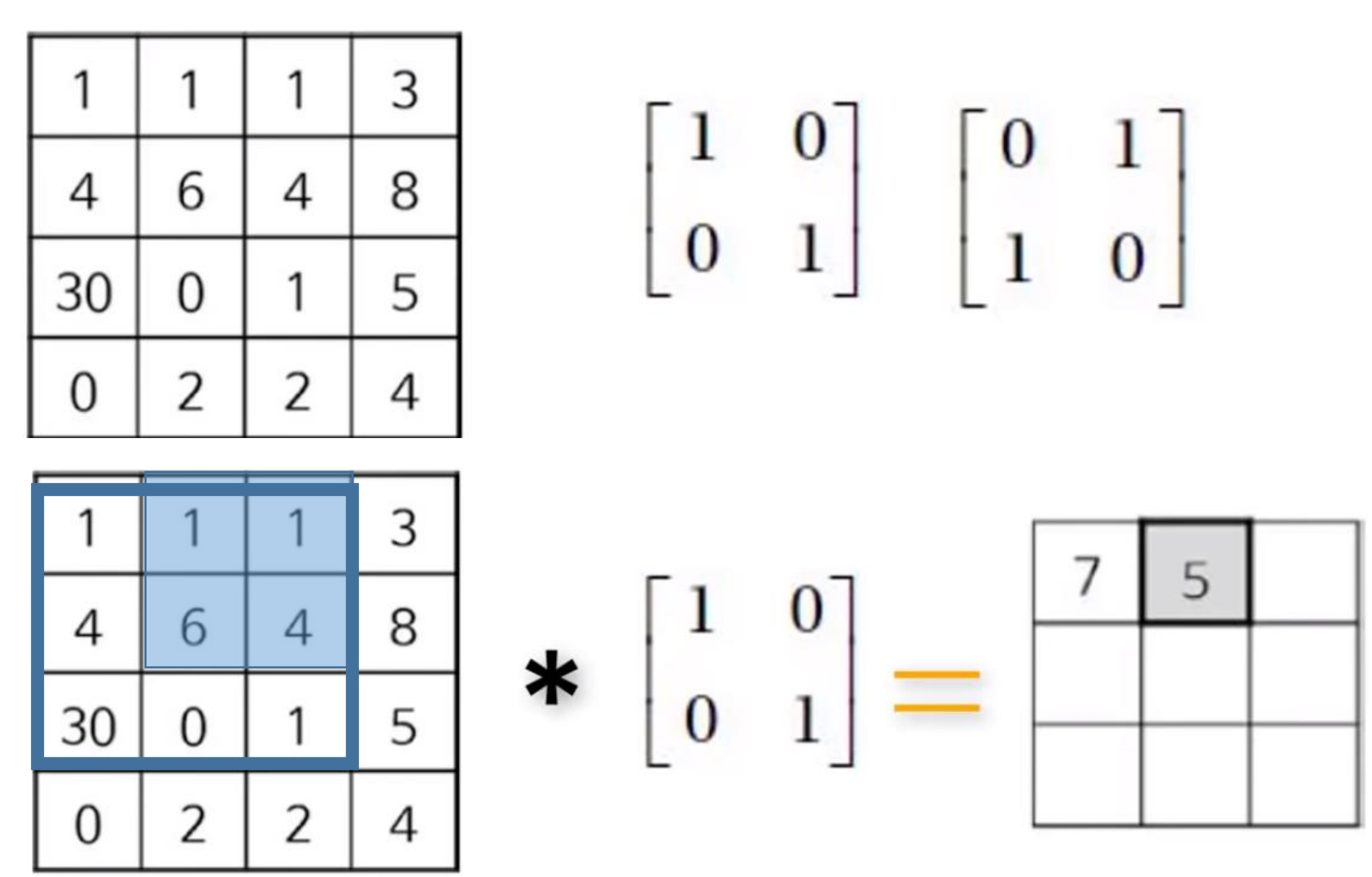

PMD
PMV
LGN
V2
V4
V1
TEO
TE

Convolutional Neural Network

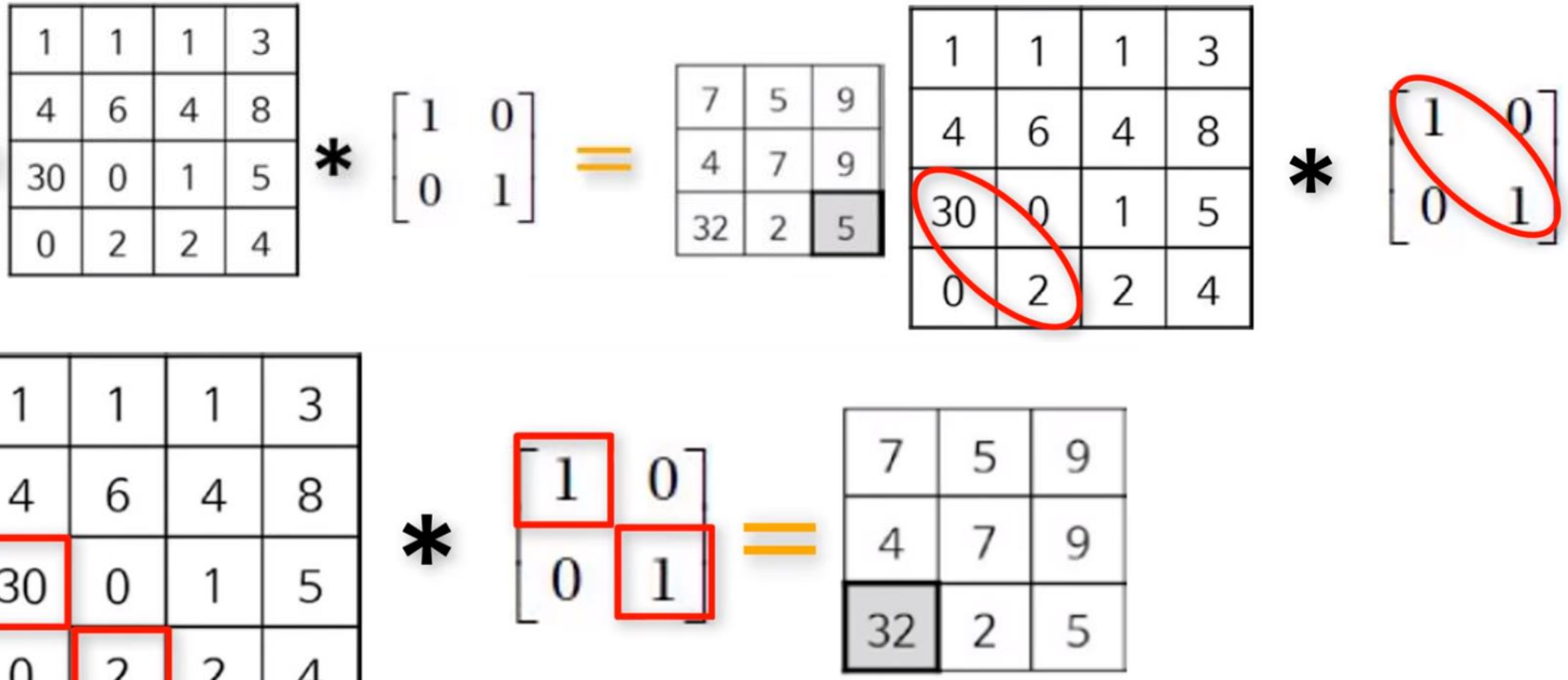



Input Image
Convolutional Layer
Feature Map

Convolution operation



Convolution operation



Convolution operation

1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

$$*\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} =$$

7	5	9
4	7	9
32	2	5

Convolution operation using second filter

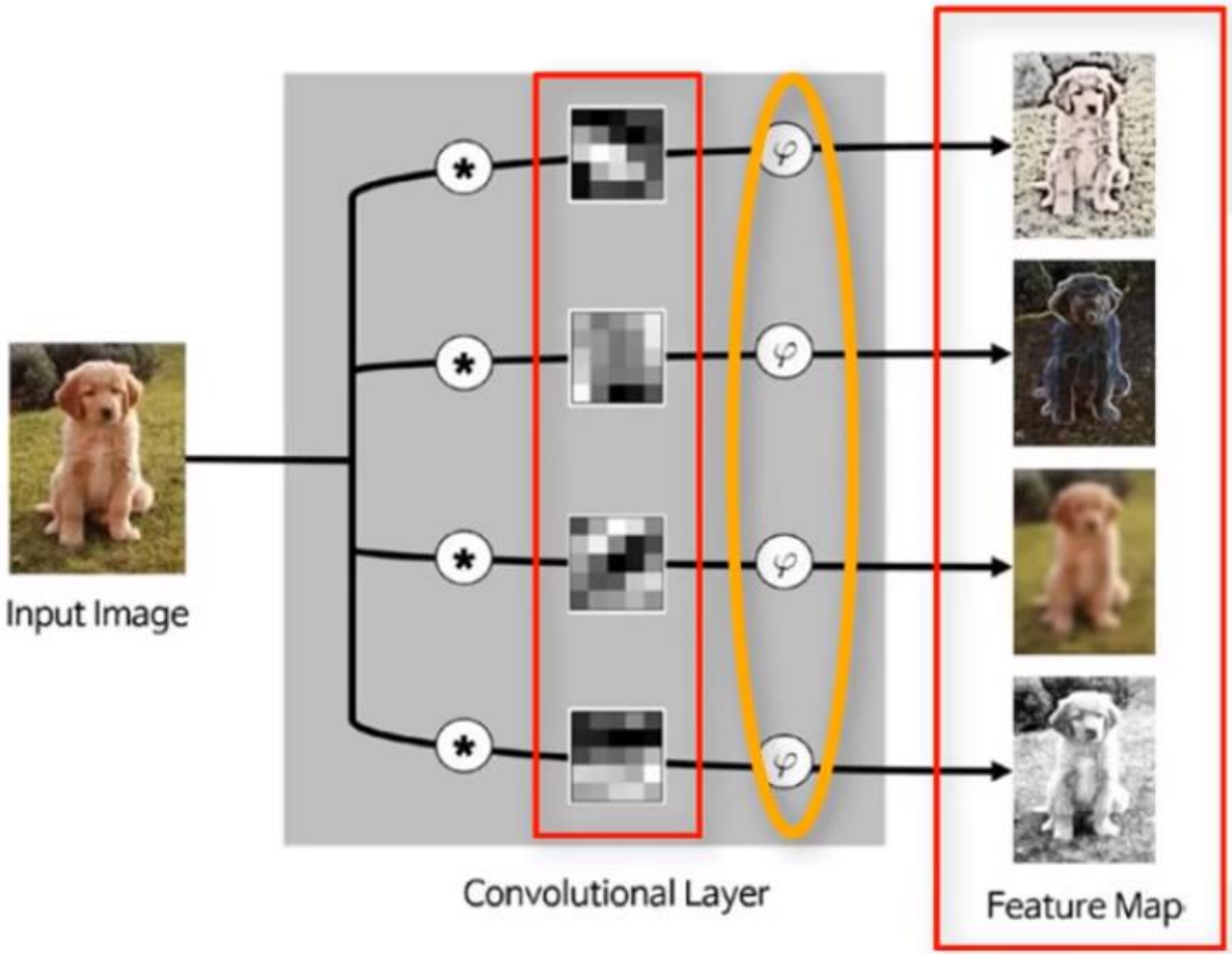
1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

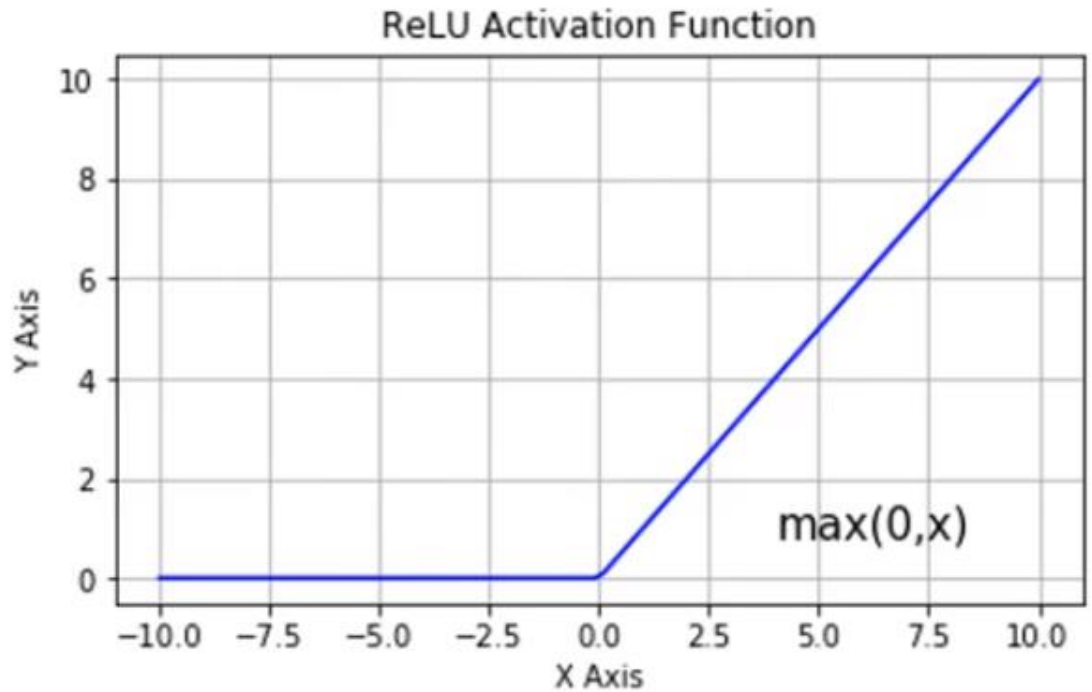
$$*\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} =$$

5	7	7
36	4	9
0	3	7

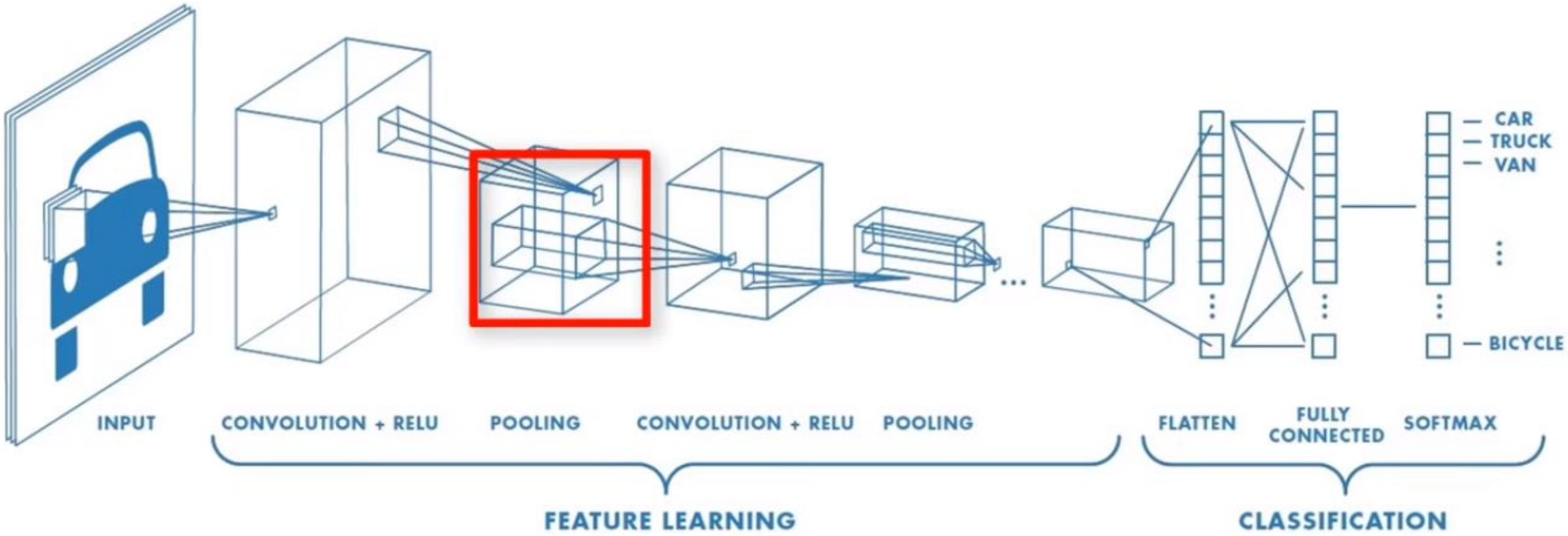
1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

$$*\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$$

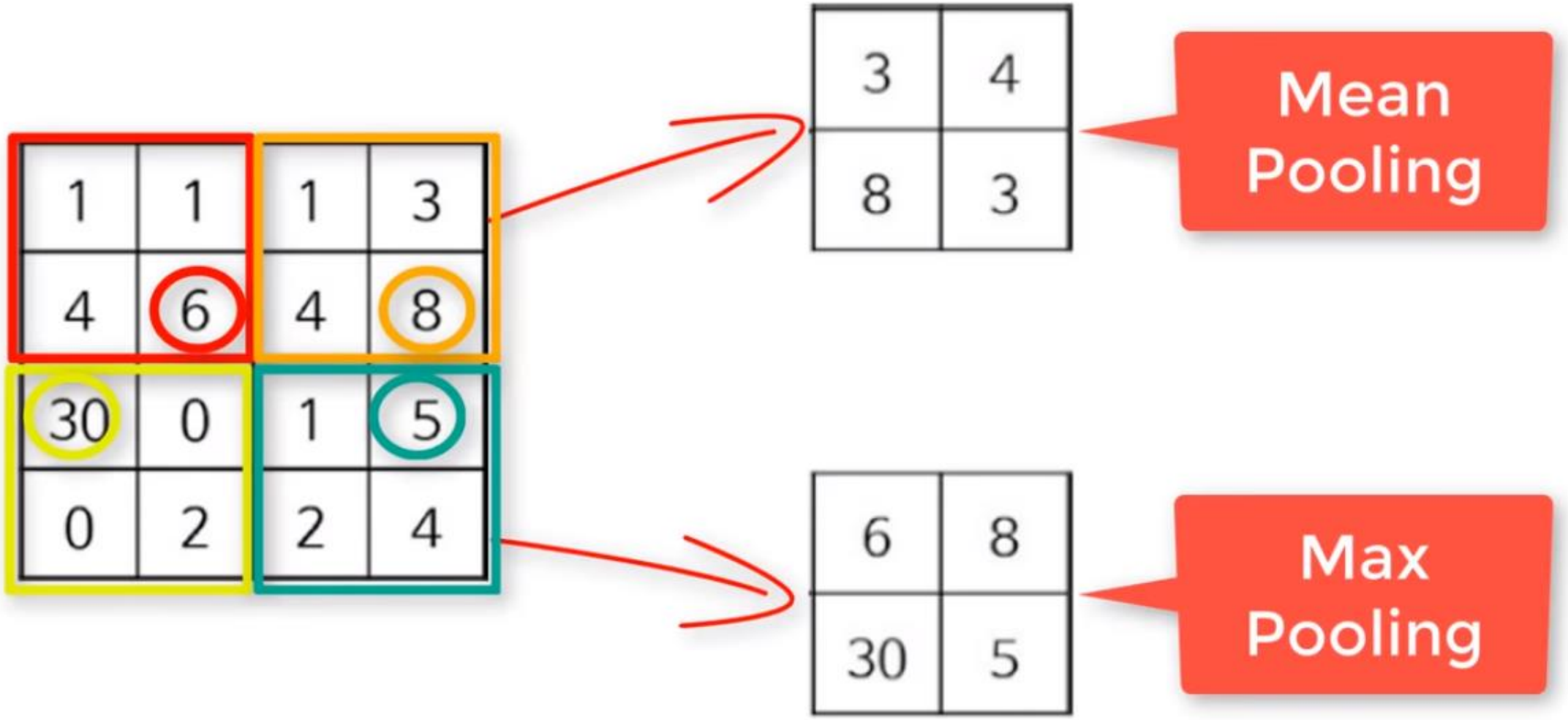

Input Image
Convolutional Layer
Feature Map


ReLU Activation Function
Y Axis
X Axis
max(0,x)
10
8
6
4
2
0
−10.0
−7.5
−5.0
−2.5
0.0
2.5
5.0
7.5
10.0

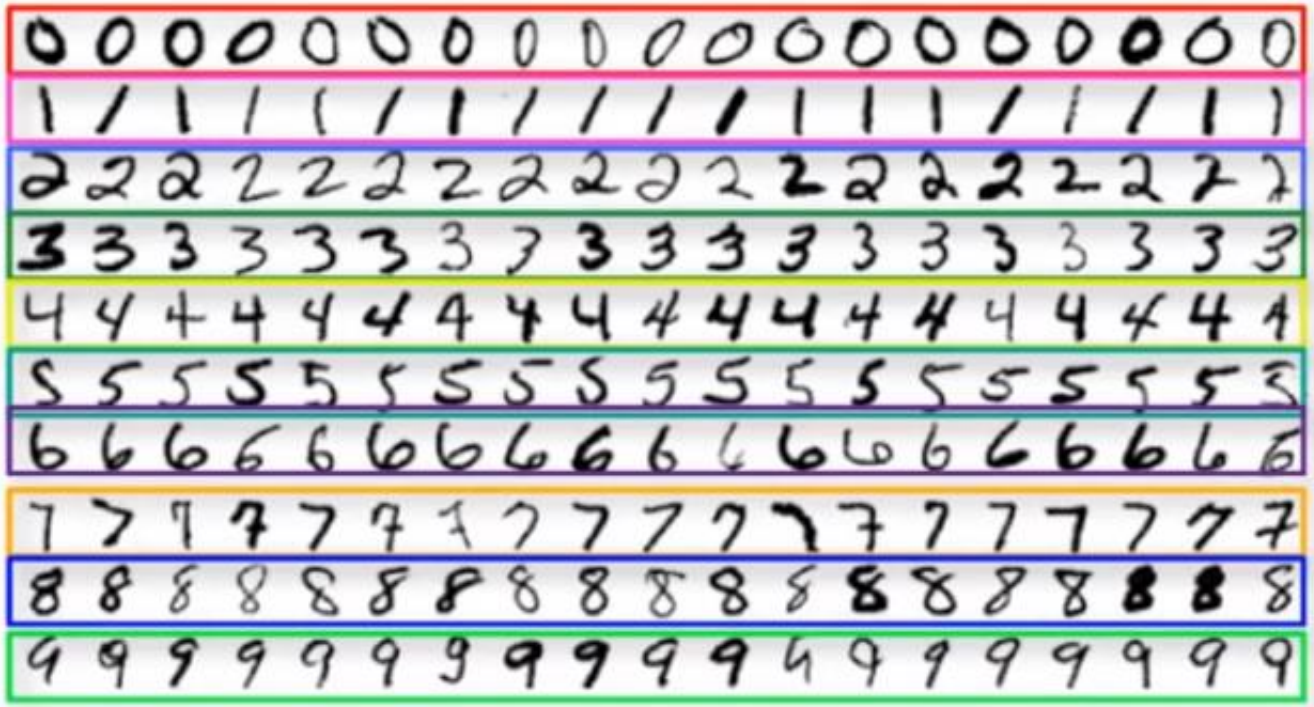
Convolutional Neural Network



Pooling



MNIST dataset

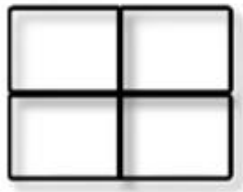


28 x 28 = 784

CNN architecture

784 input nodes

20 Convolution Filter (9x9) → ReLU

2x2 Submatrices for Pooling Layer 

Single Hidden Layer: 100 Nodes → ReLU

10 output nodes → Softmax

CNN architecture

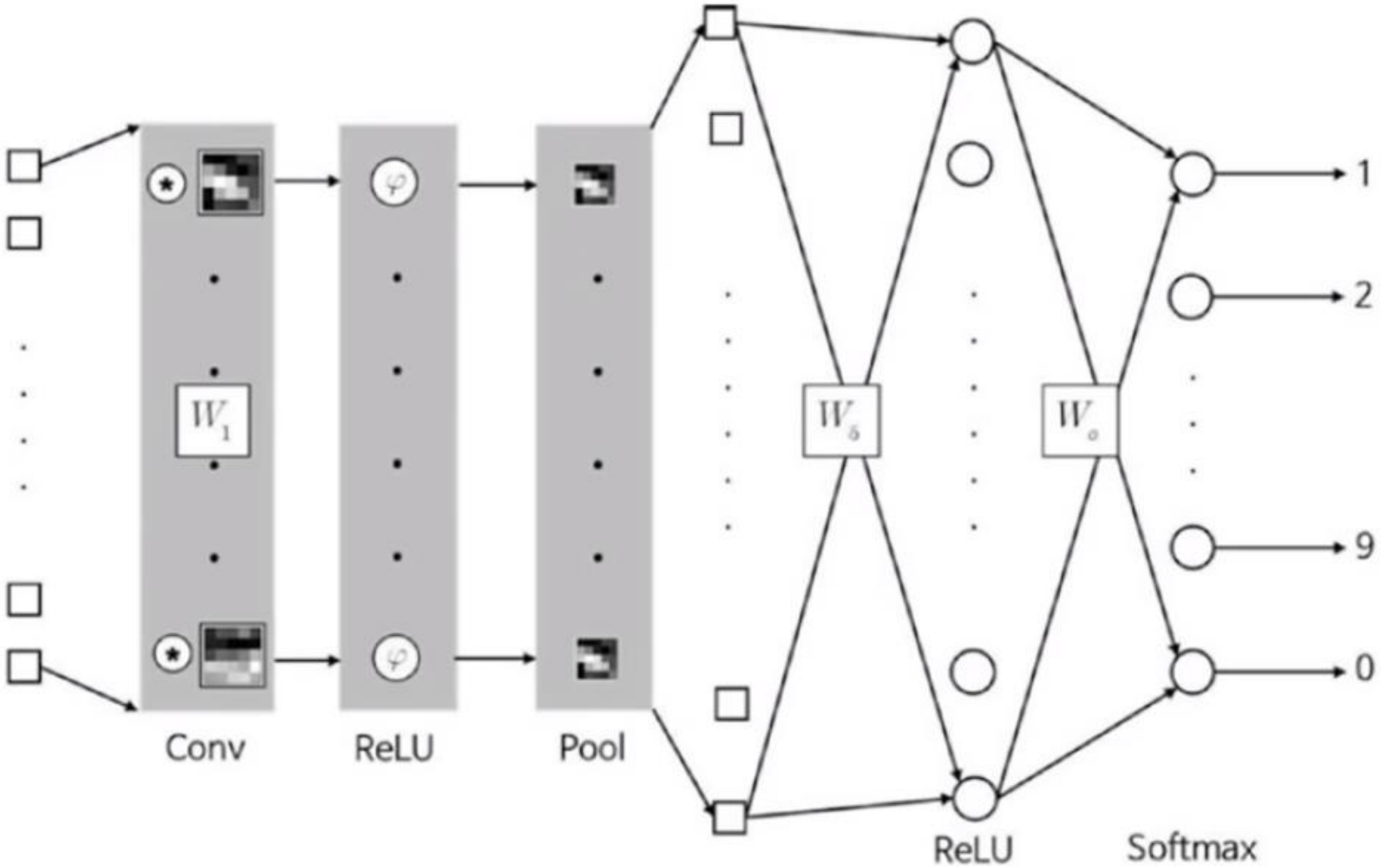
784 input nodes

20 Convolution Filter (9x9) → ReLU

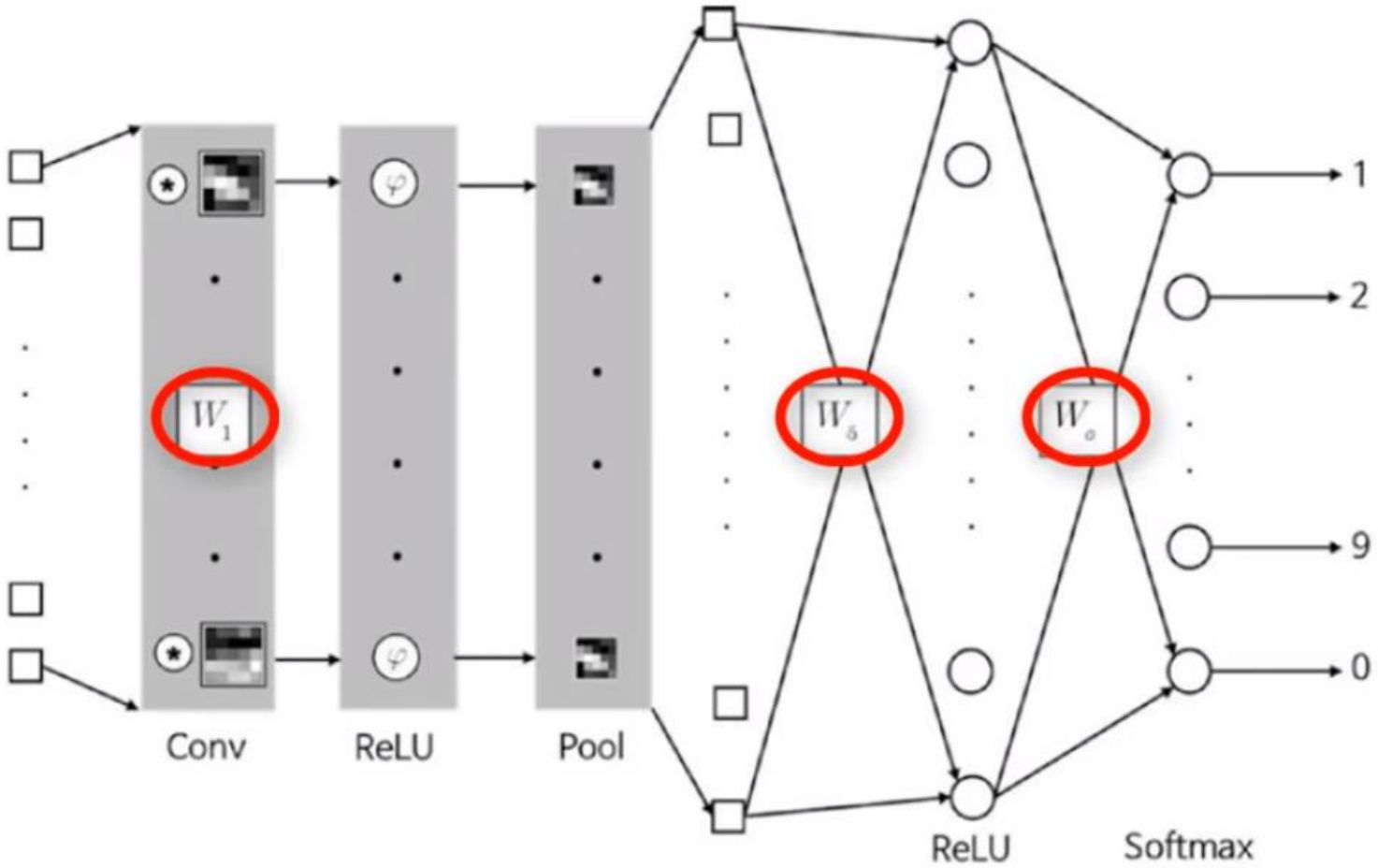
2x2 Submatrices for Pooling Layer

Single Hidden Layer: 100 Nodes → ReLU

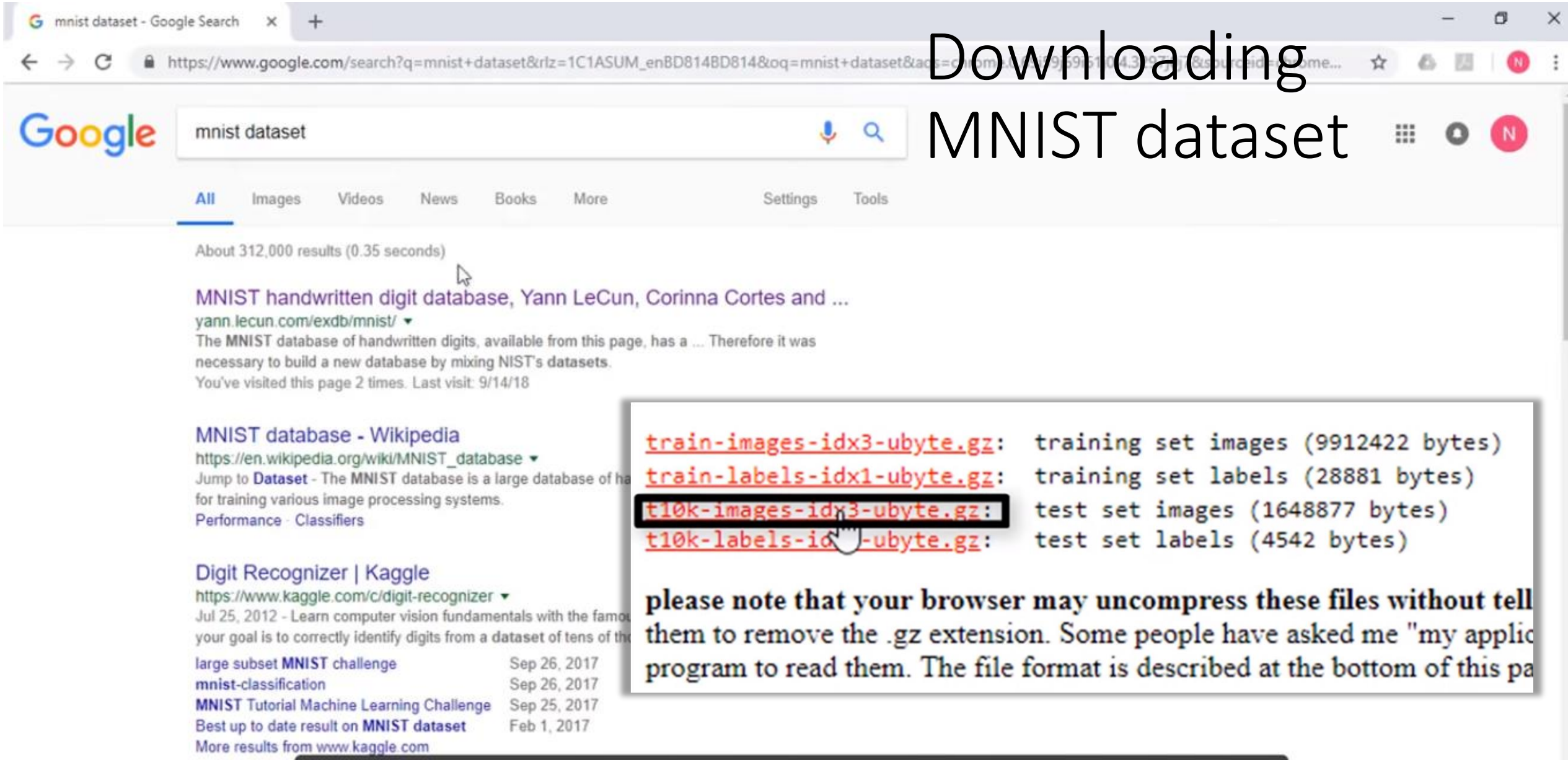
10 output nodes → Softmax



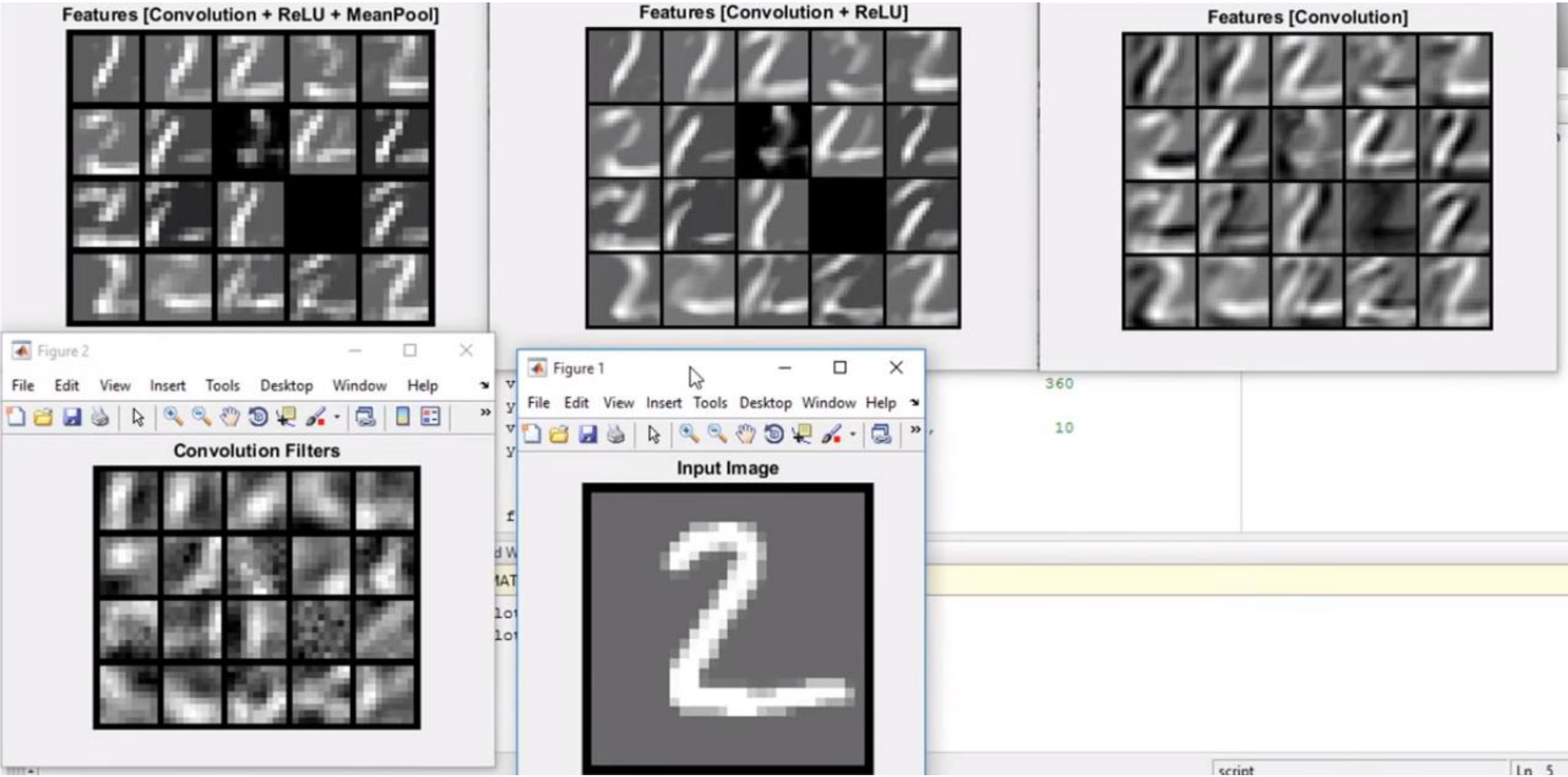
CNN architecture: training



Downloading MNIST dataset



Extracted features



Matlab Implementation

- Data in Matlab:
- C:\Program Files\MATLAB\R2020b\toolbox\nnet\nndemos\nndatasets\DigitDataset
- Training: 750 images from each digits; 7500 total images
- Validation: 250 images from each digits; 2500 total images

- imageInputLayer([Row Cols Laer],'Name', 'Input')
- imageInputLayer([28 28 1],'Name', 'Input')

- convolution2dLayer(FilterSize, numFilters, ‘Strides’, n, 'Padding',‘ same','Name','Conv_1')
- convolution2dLayer(3, 8, 'Padding', 'same', 'Name', 'Conv_1')

Batch Normalization Layer: It normalizes the activations and gradients, making network training an easier optimization problem and speeds up network training and reduces the sensitivity to network initialization.

batchNormalizationLayer('Name','BN_1')

- reluLayer('Name','Relu_1')
- Down-sampling with maxpooling:
- maxPooling2dLayer(2, 'Stride', 2, 'Name', 'MaxPool_1')

- fullyConnectedLayer(# of categories, 'Name', 'FC')
- fullyConnectedLayer(10, 'Name', 'FC')

- softmaxLayer('Name', 'SoftMax')
- classificationLayer('Name', 'OutputClassification')

Training parameters

- trainingOptions(solverName, Name, Value)
- trainingOptions('sgdm', 'LearnRateSchedule', 'piecewise', 'LearnRateDropFactor', 0.2, 'LearnRateDropPeriod', 5, 'MaxEpochs', 20, 'MiniBatchSize', 4, 'Plots', 'training-progress')

- digitDatasetPath='C:\Program Files\MATLAB\R2020b\toolbox\nnet\nndemos\nndatasets\DigitDataset‘
- Reading Digit images from the folder :
- digitimages=imageDatastore(digitDatasetPath,'IncludeSubfolders',true,'Labelsource','foldernames')

Training parameters …cntd…

- numTrainFiles=750 % 75% files for training

- [TrainImages, TestImages]=splitEachLabel(digitimages, numTrainFiles, 'Randomize')

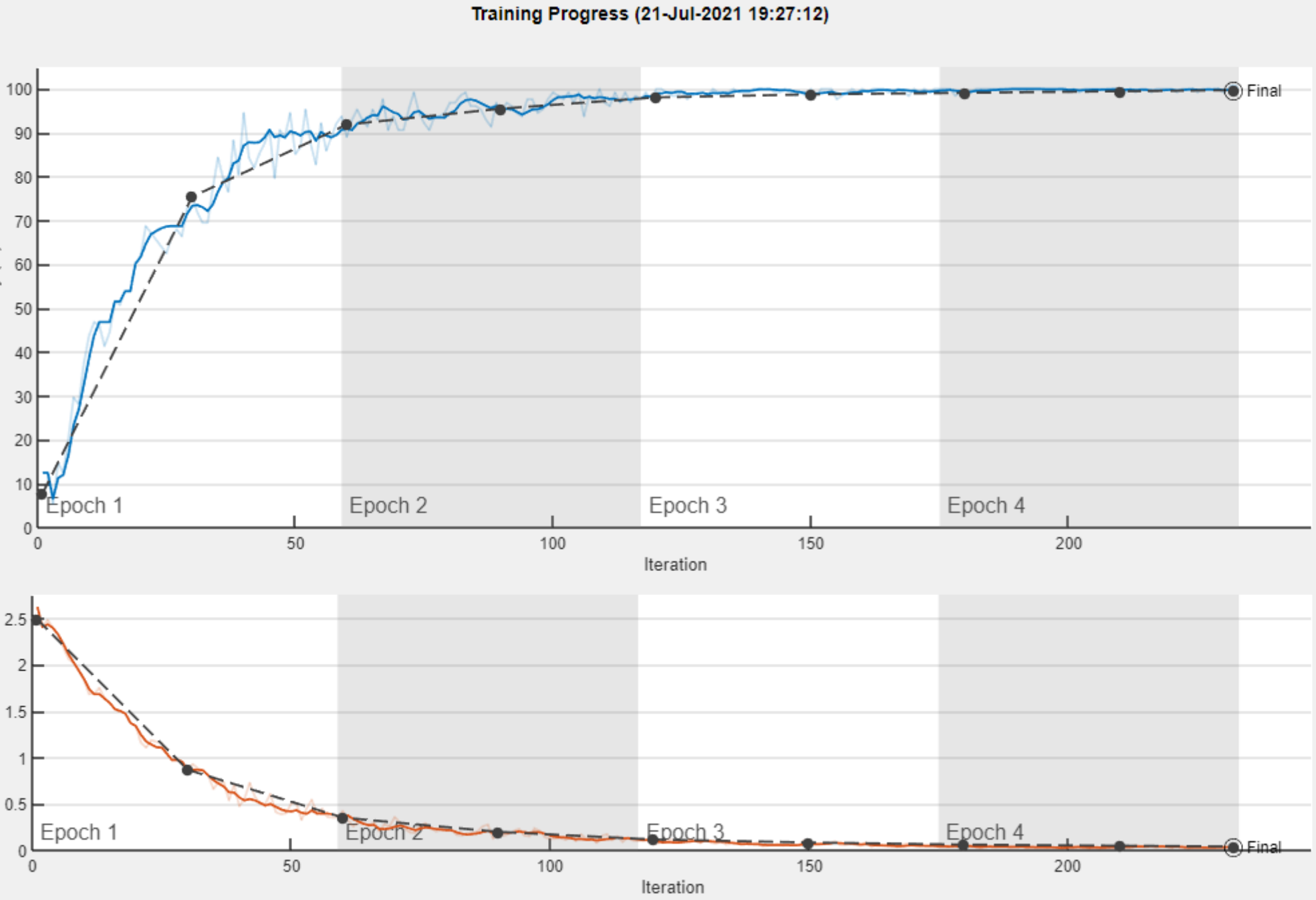
Building CNN

```
layers=[imageInputLayer([28,28,1],'Name','Input')
convolution2dLayer(3,8,'Padding','same','Name','Conv_1')
batchNormalizationLayer('Name','BN_1')
reluLayer('Name','Relu_1')
maxPooling2dLayer(2,'Stride',2,'Name','Maxpool_1')

convolution2dLayer(3,16,'Padding','same','Name','Conv_2')
batchNormalizationLayer('Name','BN_2')
reluLayer('Name','Relu_2')
maxPooling2dLayer(2,'Stride',2,'Name','Maxpool_2')

convolution2dLayer(3,32,'Padding','same','Name','Conv_3')
batchNormalizationLayer('Name','BN_3')
reluLayer('Name','Relu_3')
maxPooling2dLayer(2,'Stride',2,'Name','Maxpool_3')
```

Misc

- lgraph=layerGraph(layers)
- plot(lgraph)

- options=trainingOptions('sgdm','InitialLearnRate',0.01,'MaxEpochs',4,'Shuffle','every-epoch','ValidationData',TestImages,'ValidationFrequency',30,'Verbose',false,'Plots','training-progress')



Results

Validation accuracy:	99.72%
Training finished:	Reached final iteration

Training Time

Start time:	21-Jul-2021 19:27:12
Elapsed time:	54 sec

Training Cycle

Epoch:	4 of 4
Iteration:	232 of 232
Iterations per epoch:	58
Maximum iterations:	232

Validation

Frequency:	30 iterations

Other Information

Hardware resource:	Single CPU
Learning rate schedule:	Constant
Learning rate:	0.01

Learn more

Accuracy

Training (smoothed)
Training
Validation

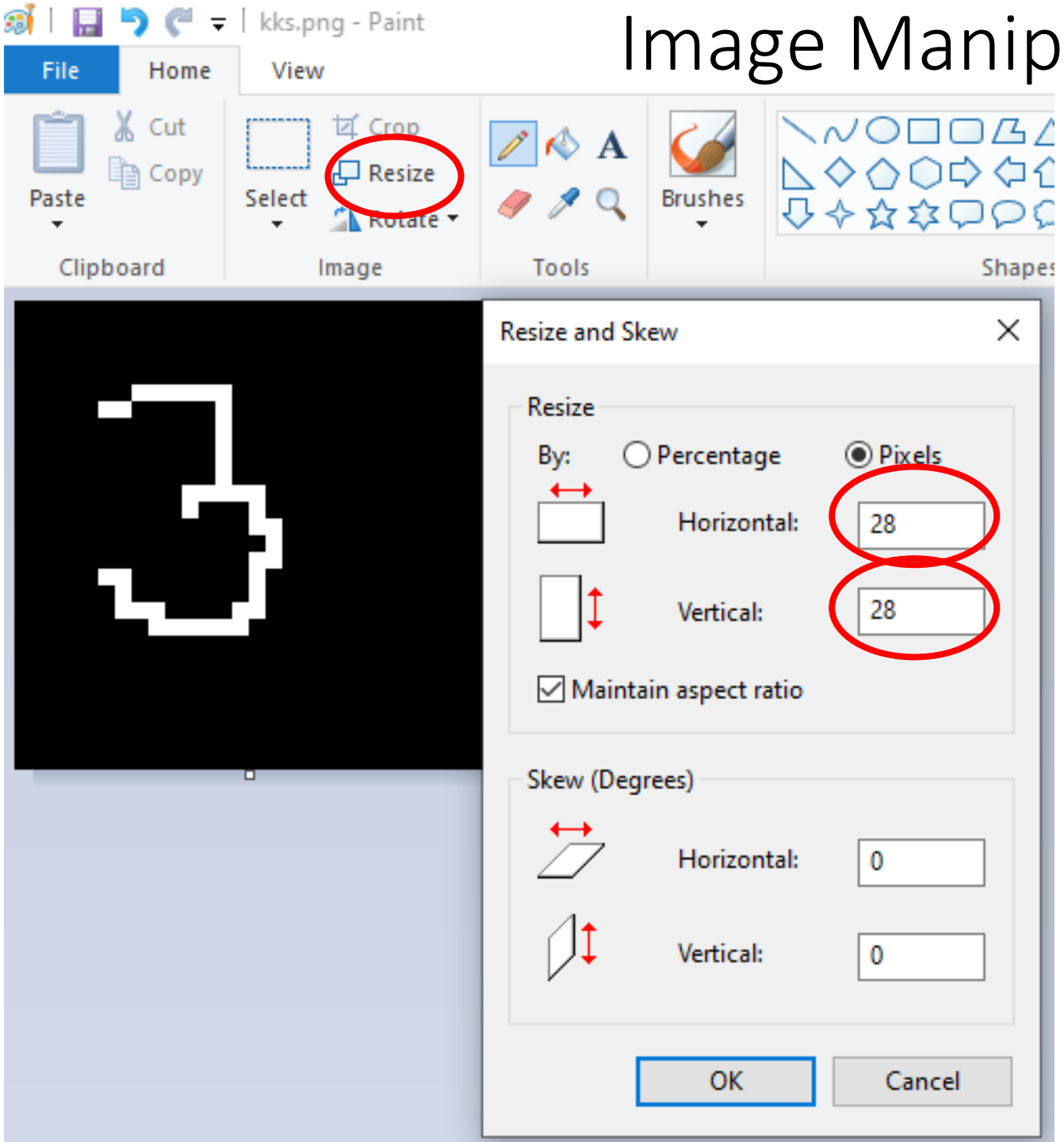
Loss

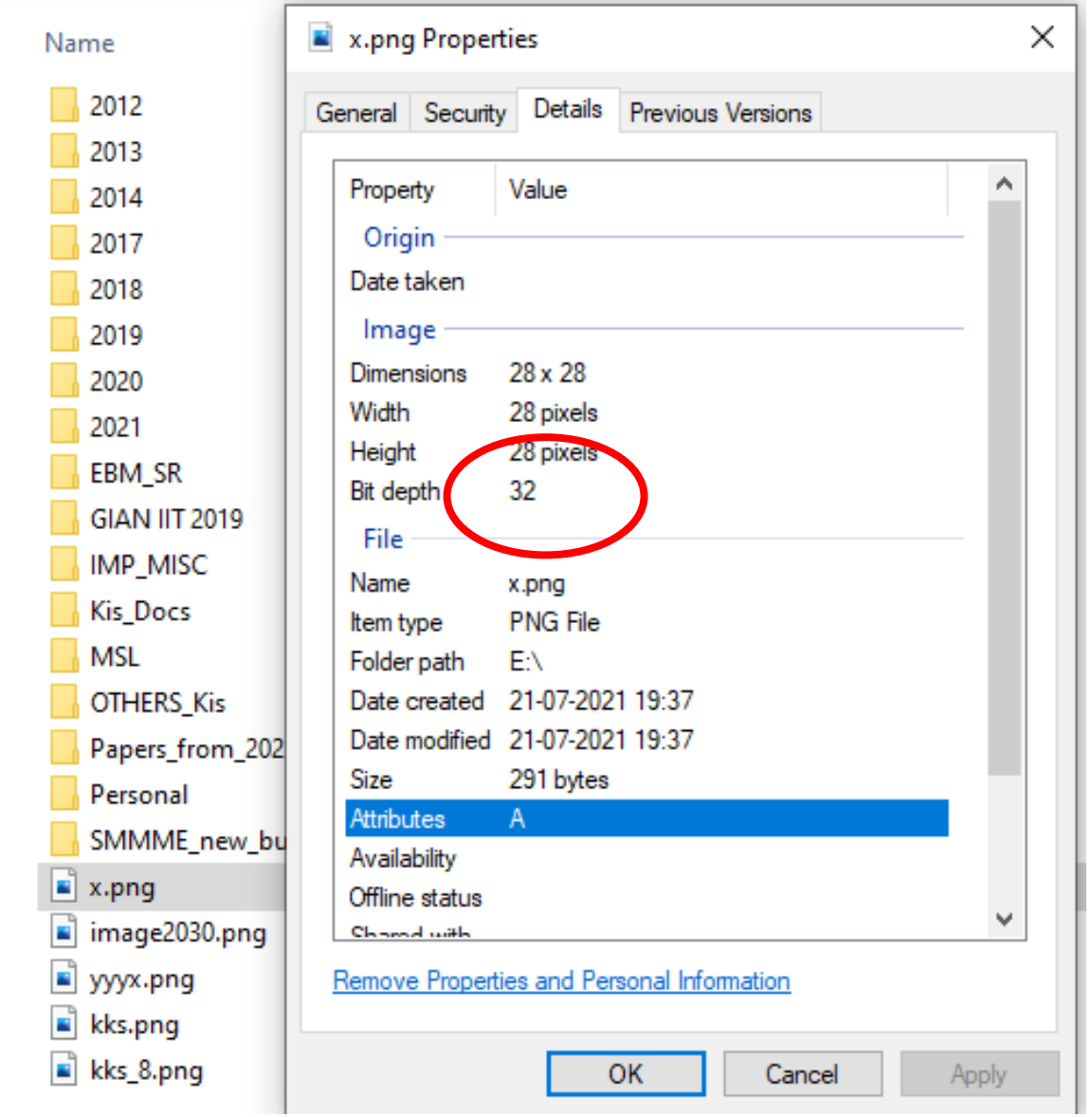
Training (smoothed)
Training
Validation

- YPred=classify(net,TestImages)
- YValidation=TestImages.Labels

- accuracy=sum(YPred == YValidation)/numel(YValidation)

Image Manipulation





im_test

- [filename,pathname]=uigetfile('*.*','Select the input greyscale image');
- filewithpath=strcat(pathname,filename);
- I=imread(filewithpath);
- figure;
- imshow(I)
-
- label=classify(net,I);
- title(['The recognised digit is ', char(label)])

Image testing

- kk=imread('kks.png');
- kk=rgb2gray(kk);
- imwrite(kk,'kks_8.png','BitDepth',8);
- kkxx=imageDatastore('kks_8.png');
- classify(net,kkxx)