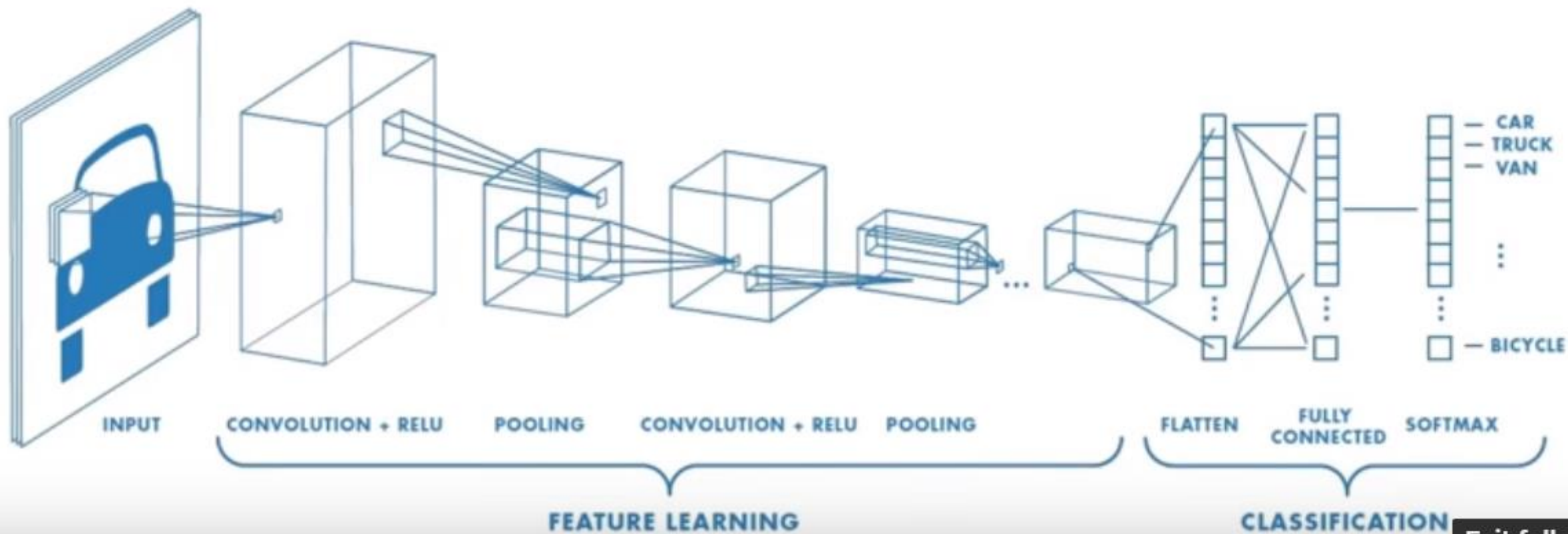


# CNN

Dr. Kisor Kumar Sahu

SMMME, IITBBS

# Convolutional Neural Network



# Visual cortex

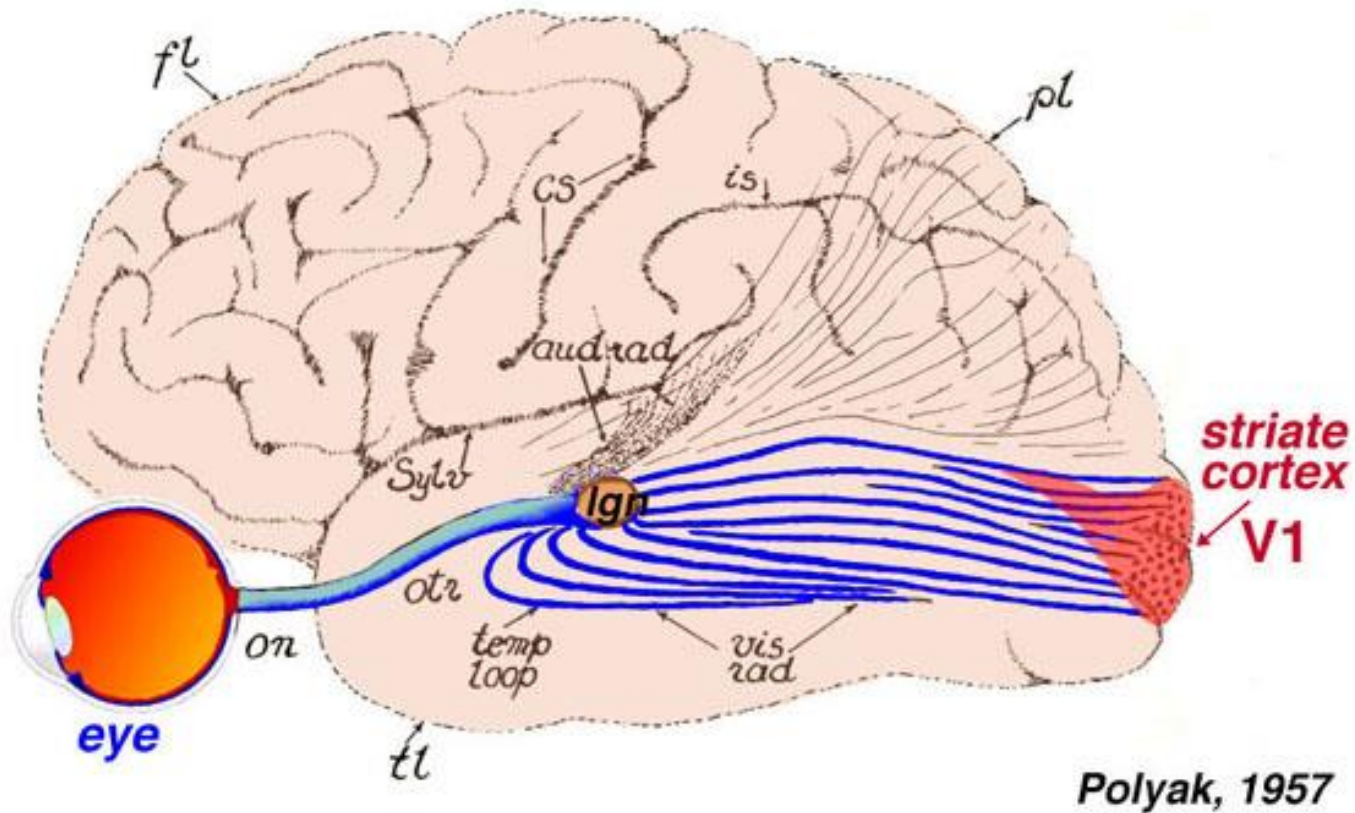
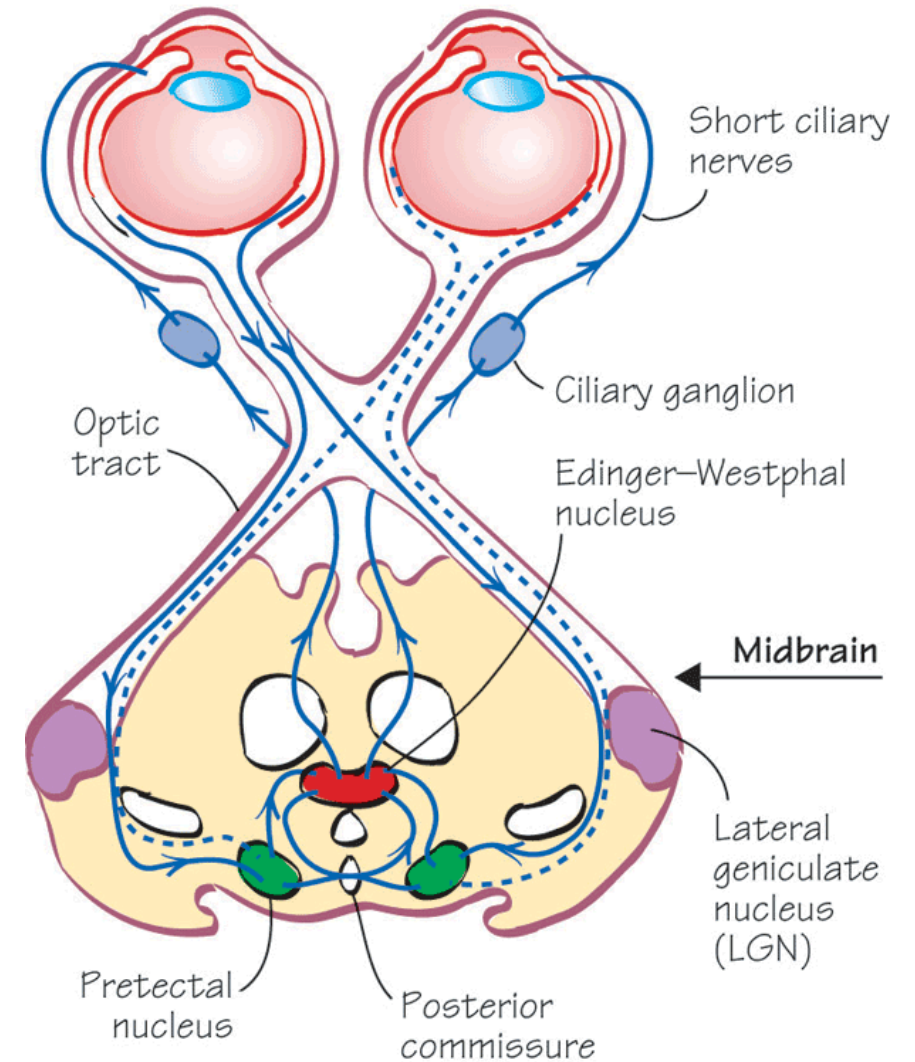
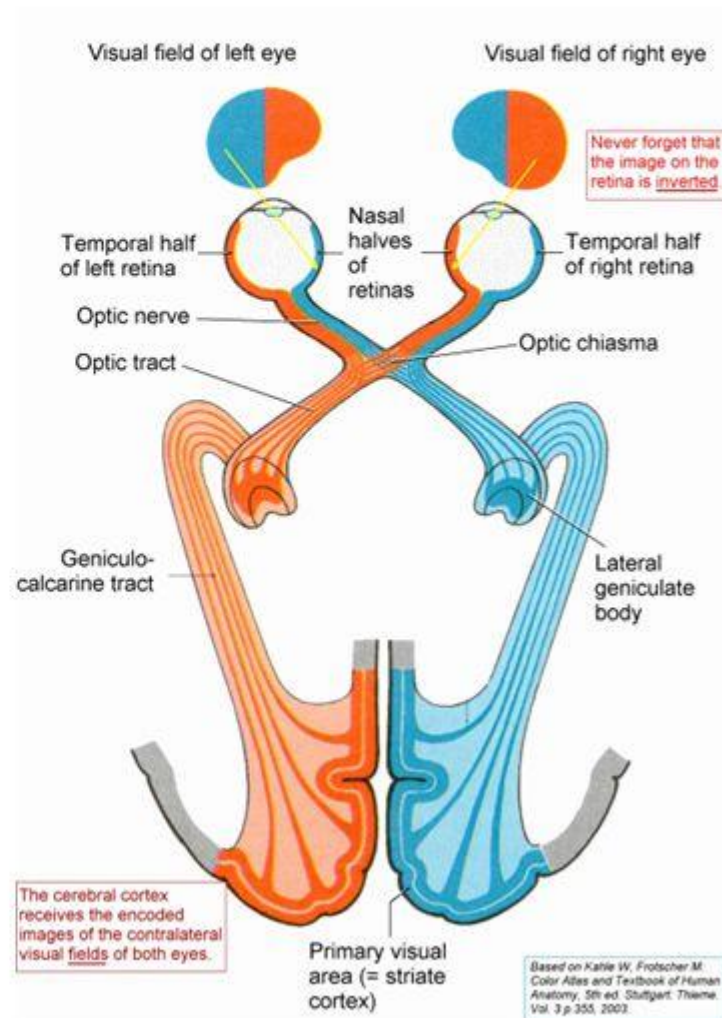
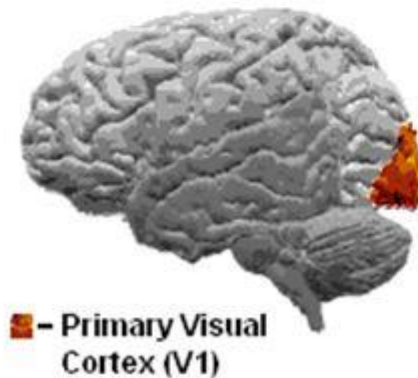


Figure 8. Visual input to the brain goes from eye to LGN and then to primary visual cortex, or area V1, which is located in the posterior of the occipital lobe.  
Adapted from Polyak (1957).

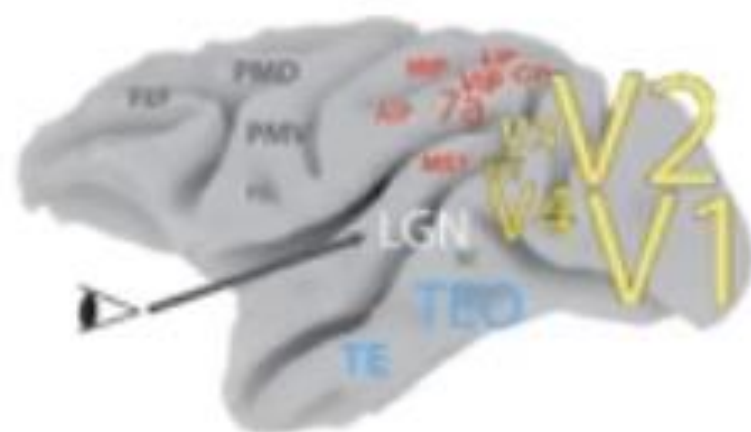
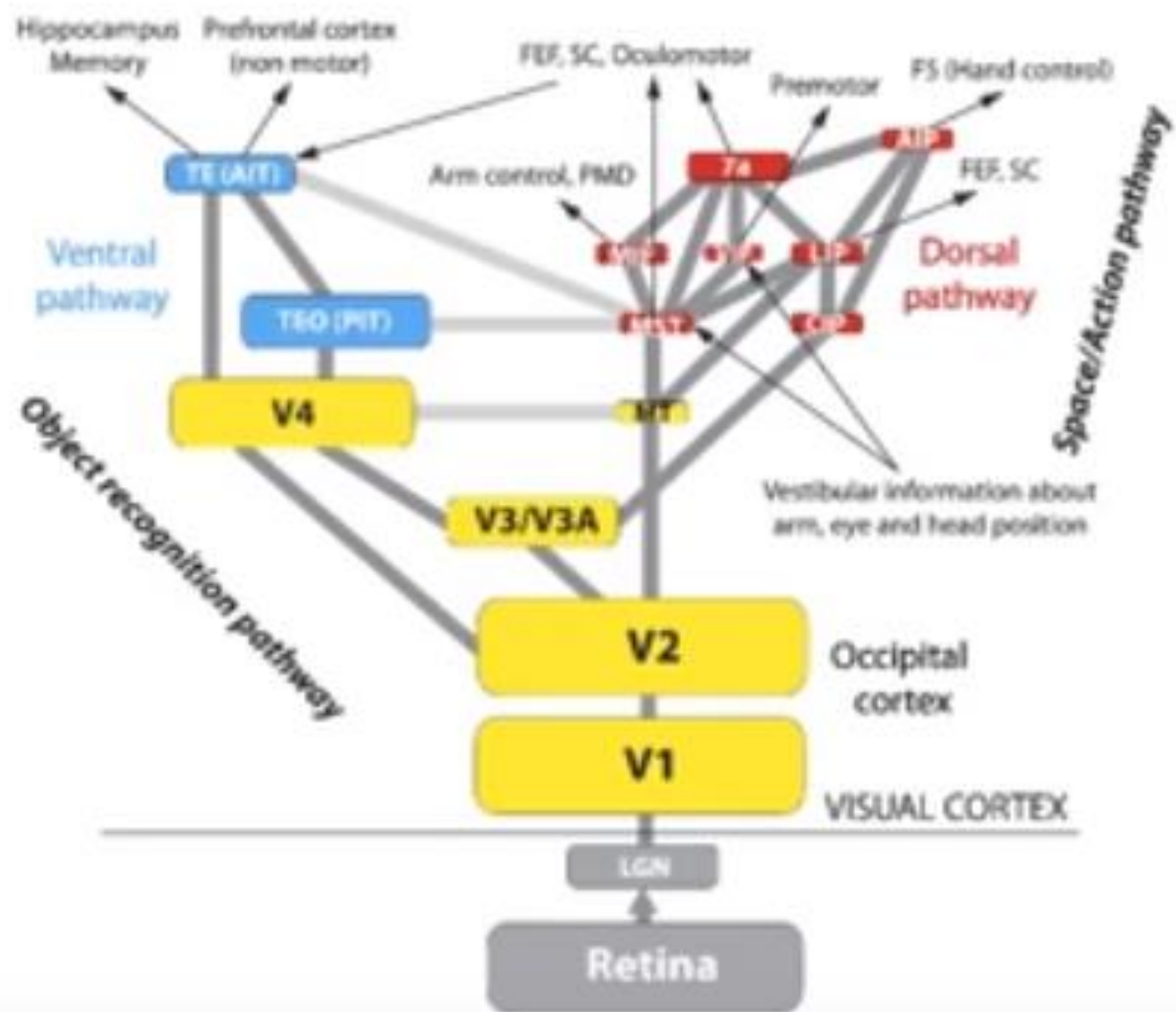


# The Primary Visual Cortex

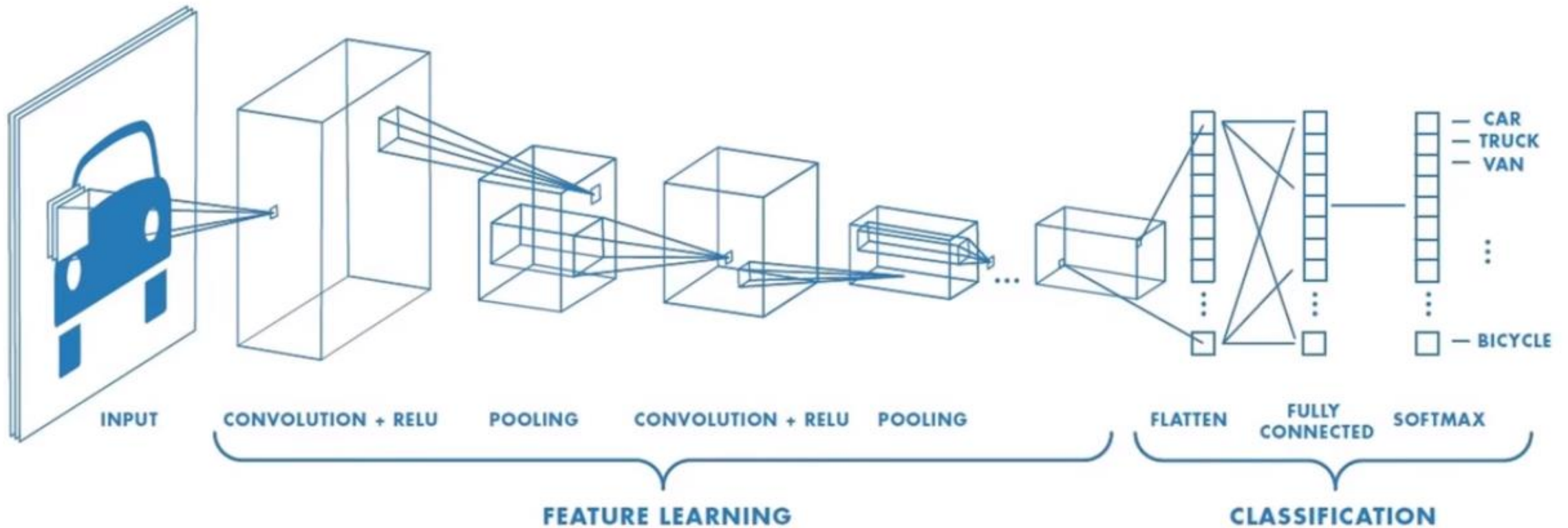
- A multi-layered structure located in the occipital lobe (AKA, Area 17, V1, Striate cortex).
- Receives axons from the LGN.
- Each hemisphere represents the contralateral visual field.





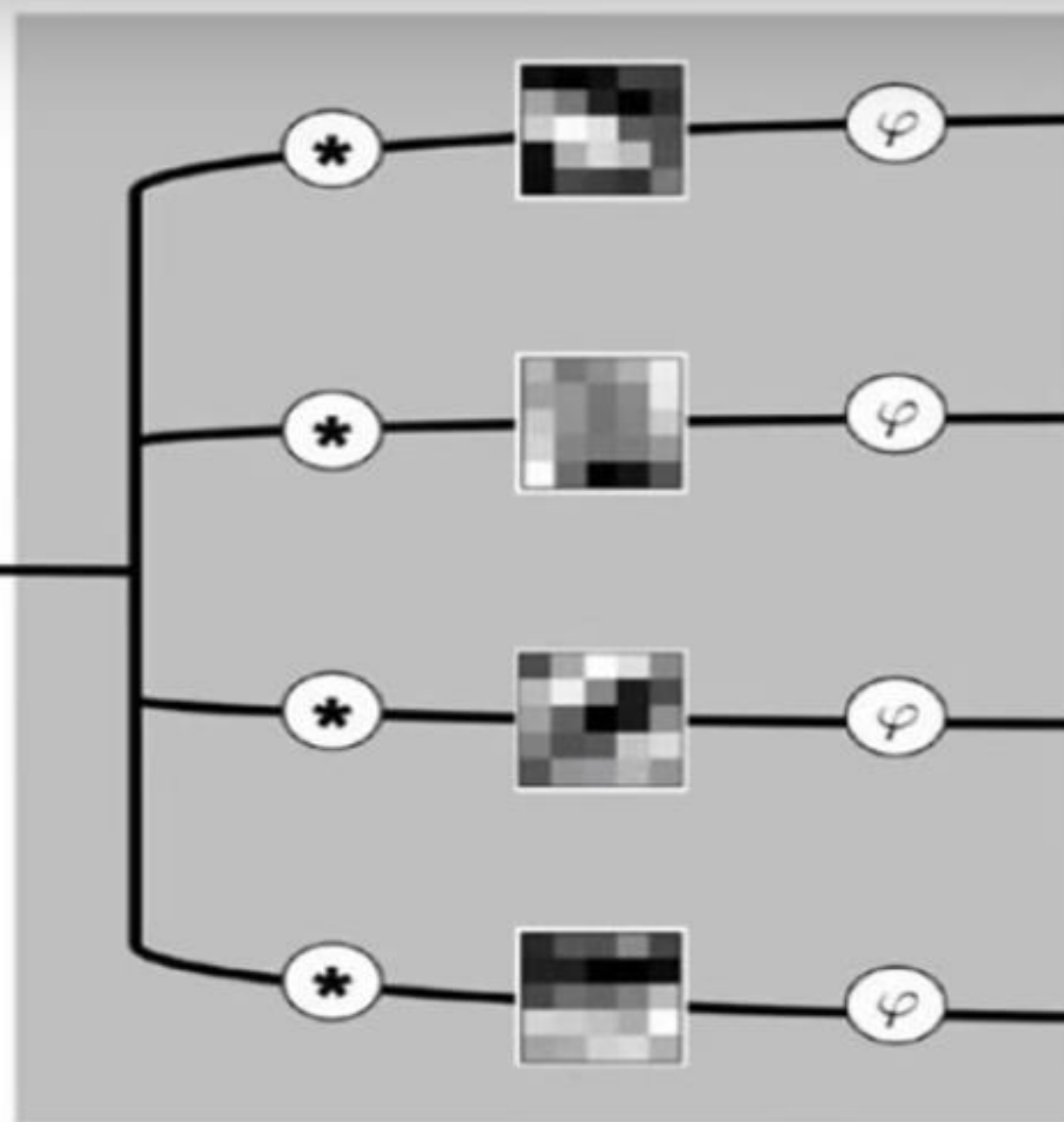


# Convolutional Neural Network





Input Image



Convolutional Layer



Feature Map

# Convolution operation

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

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

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	

$$1 \times 1 + 1 \times 0 + 6 \times 0 + 1 \times 4 = 5$$



# Convolution operation

$$\begin{bmatrix} 1 & 1 & 1 & 3 \\ 4 & 6 & 4 & 8 \\ 30 & 0 & 1 & 5 \\ 0 & 2 & 2 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 9 \\ 4 & 7 & 9 \\ 32 & 2 & 5 \end{bmatrix}$$

The diagram illustrates the first step of a 2D convolution operation. A 4x4 input matrix is multiplied by a 2x2 kernel matrix. The result is a 3x3 output matrix. Red circles highlight the 3x3 region of the input matrix (rows 2-4, columns 1-3) and the 2x2 kernel matrix, indicating the area being processed.

$$\begin{bmatrix} 1 & 1 & 1 & 3 \\ 4 & 6 & 4 & 8 \\ 30 & 0 & 1 & 5 \\ 0 & 2 & 2 & 4 \end{bmatrix} * \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} 7 & 5 & 9 \\ 4 & 7 & 9 \\ 32 & 2 & 5 \end{bmatrix}$$

The diagram illustrates the second step of the 2D convolution operation. The 4x4 input matrix is multiplied by the 2x2 kernel matrix. The result is a 3x3 output matrix. Red boxes highlight the 3x3 region of the input matrix (rows 2-4, columns 1-3) and the 2x2 kernel matrix, indicating the area being processed.

# 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

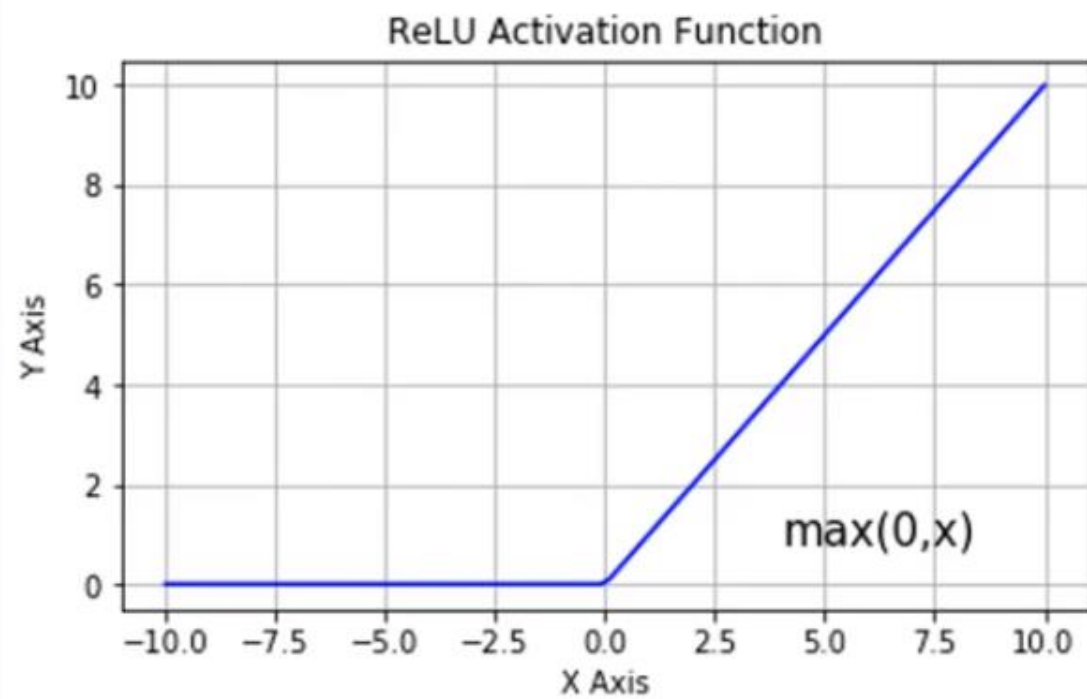
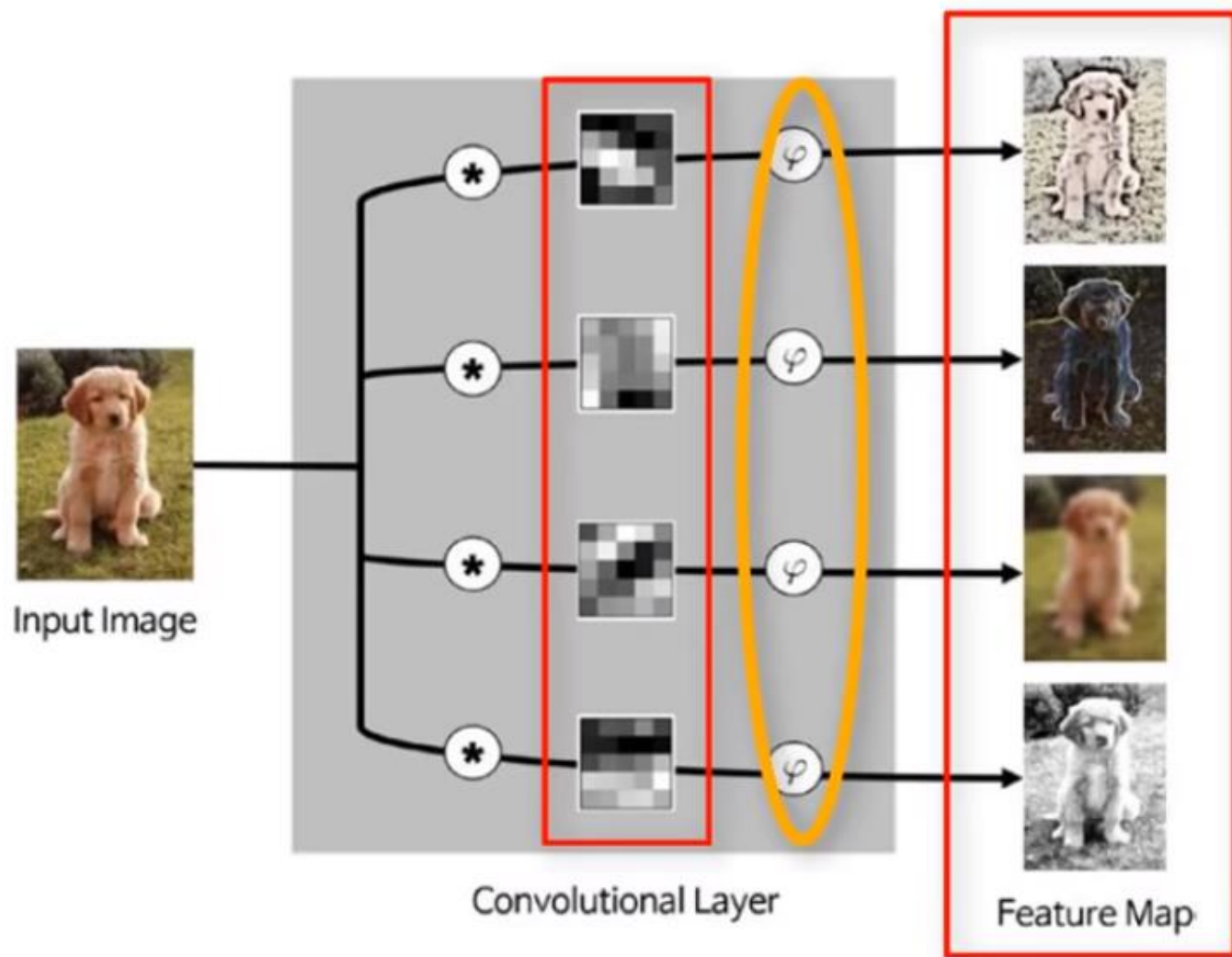
1	1	1	3
4	6	4	8
30	0	1	5
0	2	2	4

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

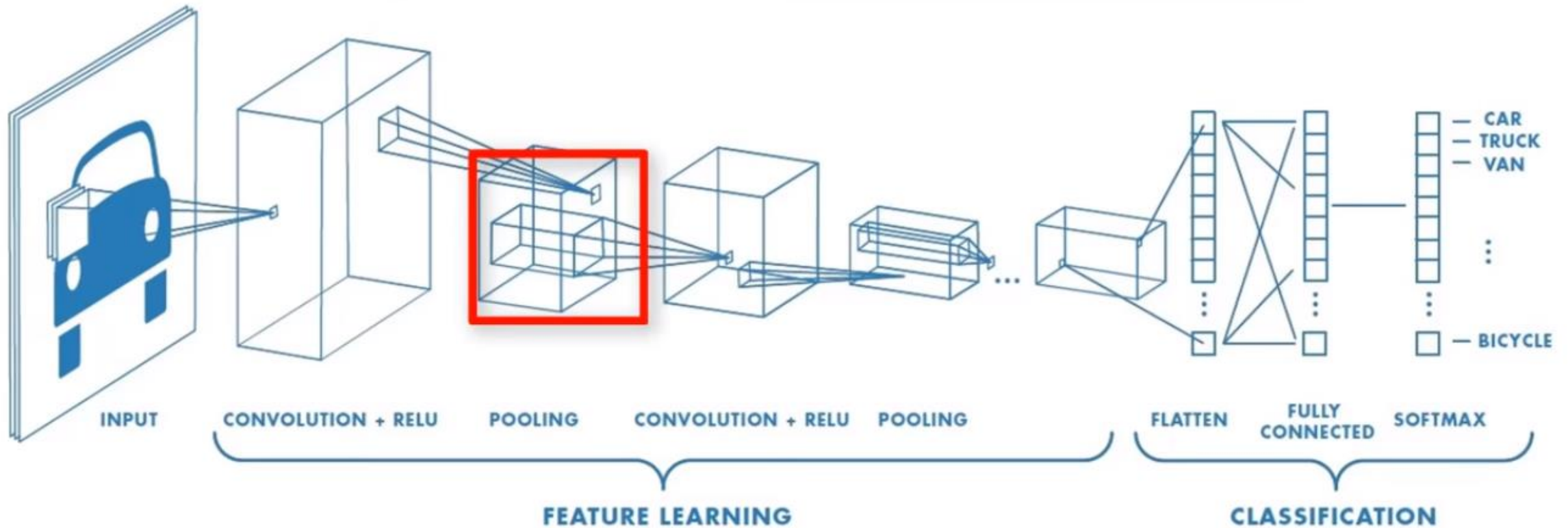
5	7	7
36	4	9
0	3	7

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

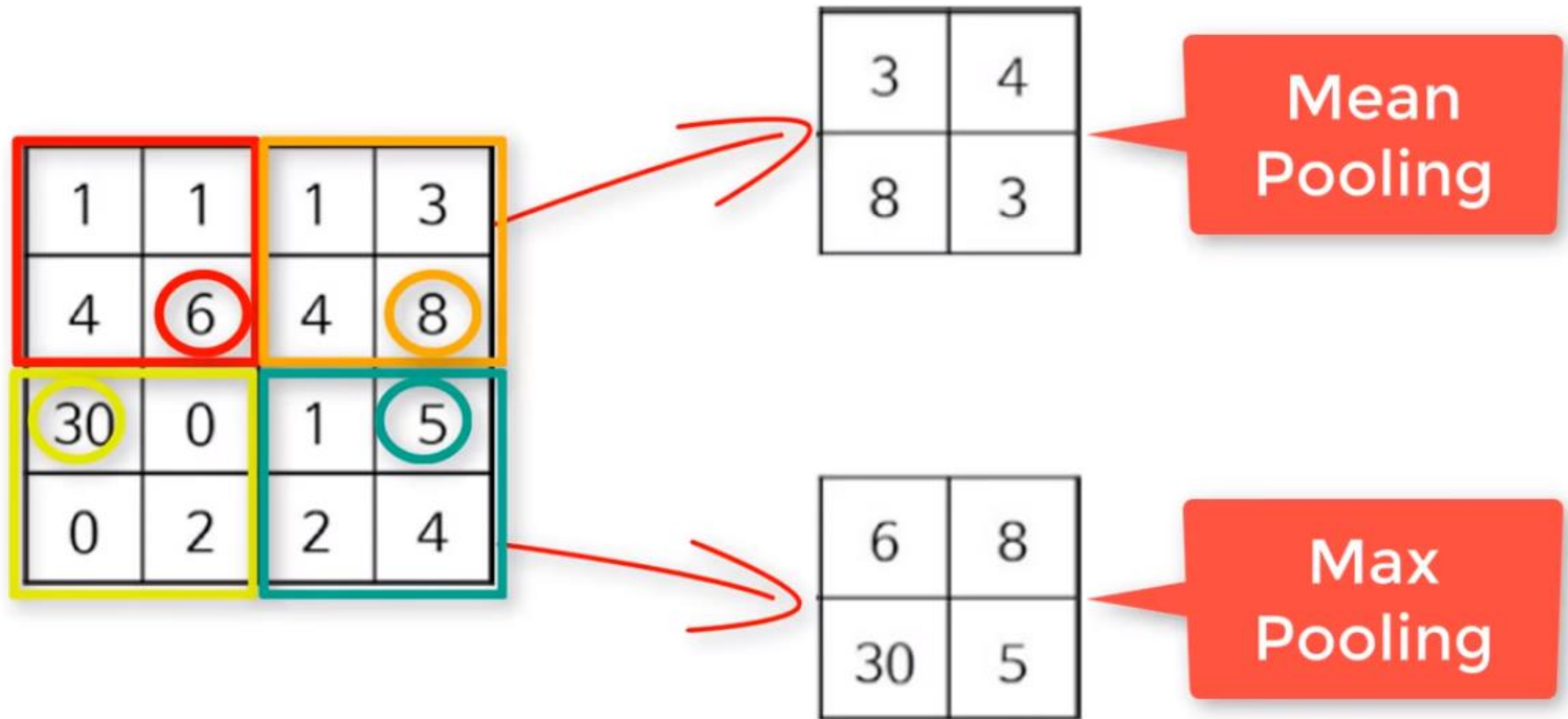
 $*$  $\begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$



# Convolutional Neural Network

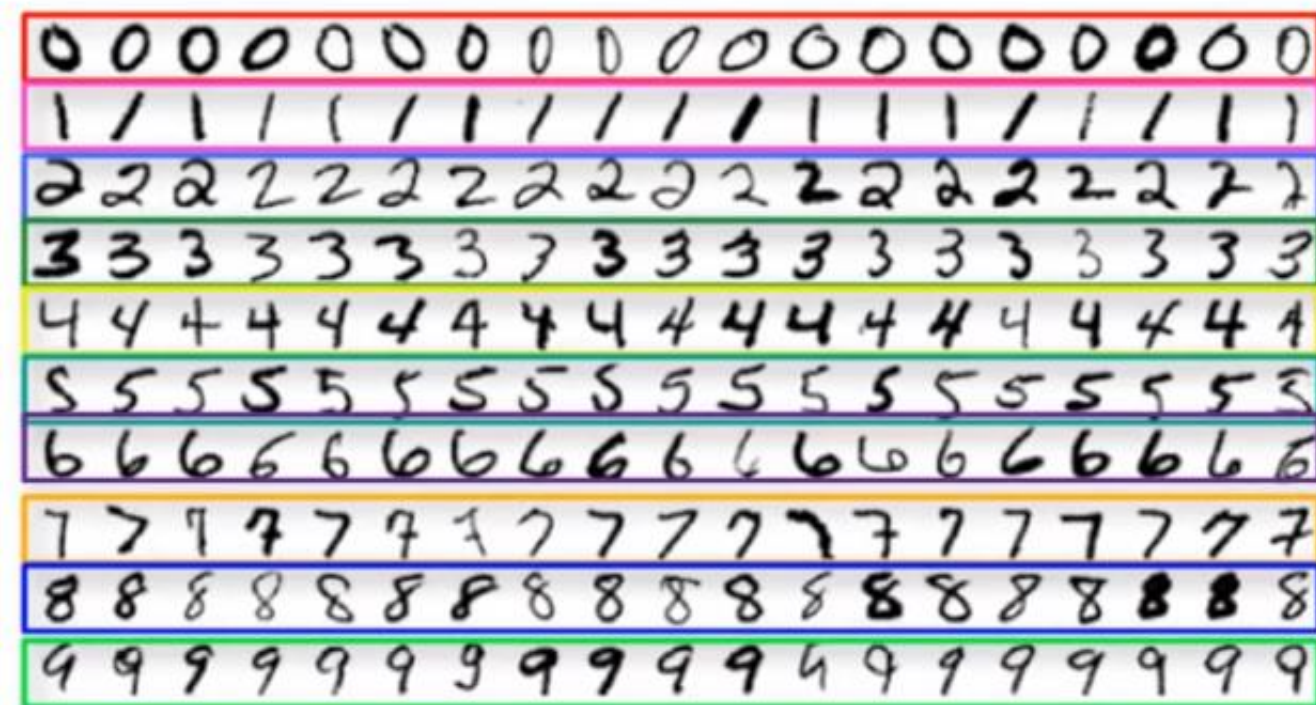


# Pooling





# MNIST dataset



0  
1  
2  
3  
4  
5  
6  
7  
8  
9

$$28 \times 28 = 784$$

# 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

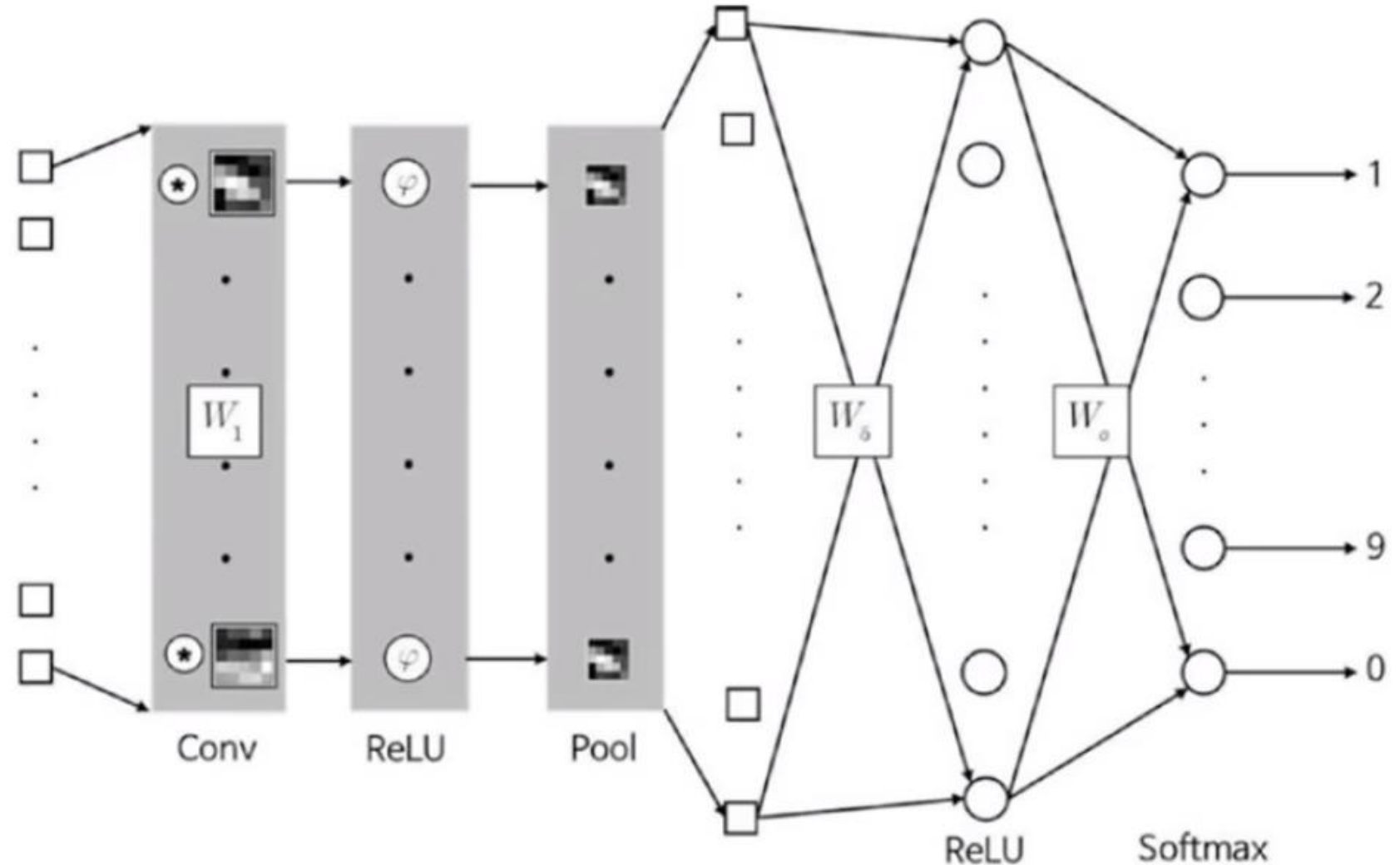
20 Convolution Filter (9x9) → ReLU

2x2 Submatrices for Pooling Layer

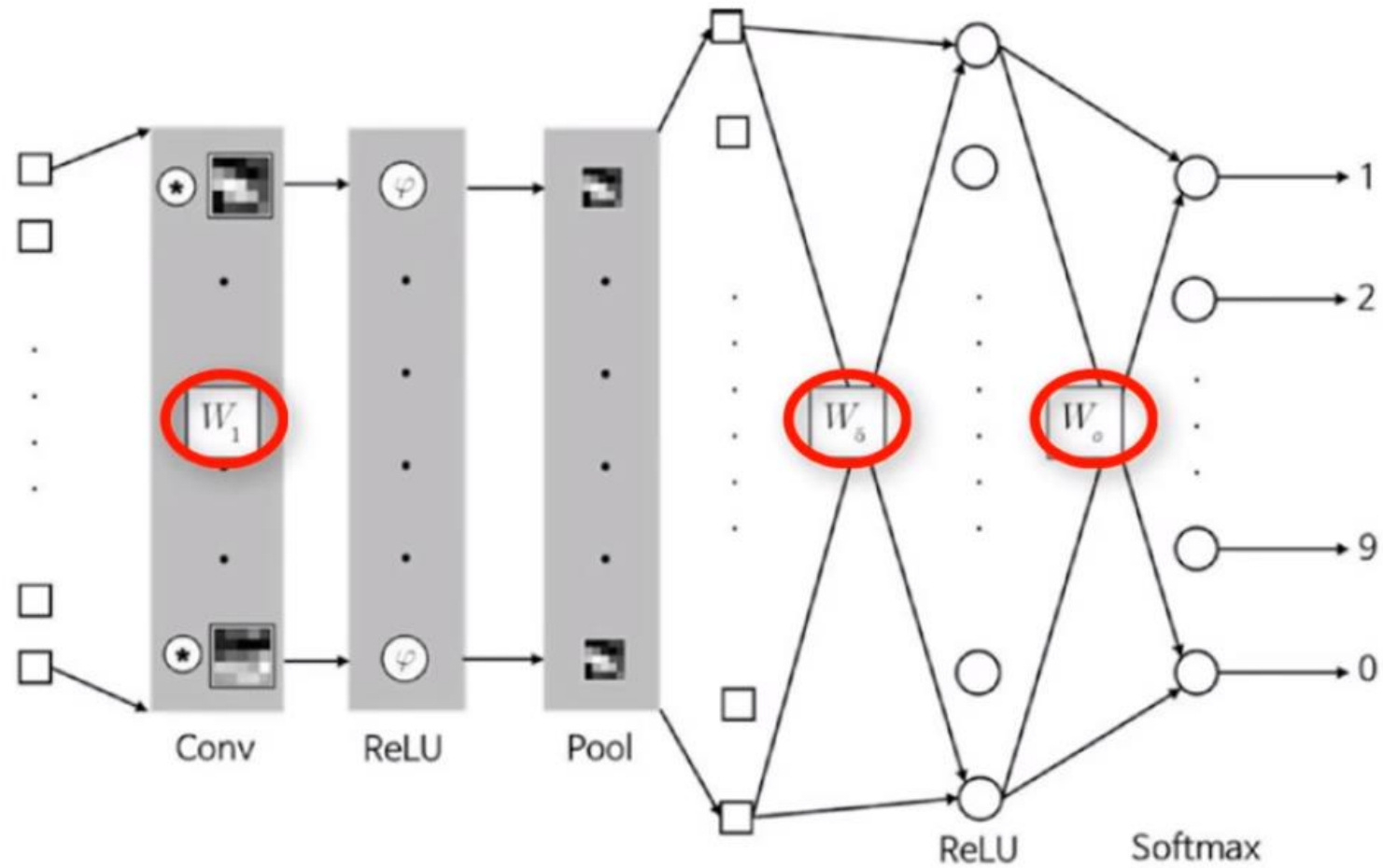


Single Hidden Layer: 100 Nodes → ReLU

10 output nodes → Softmax

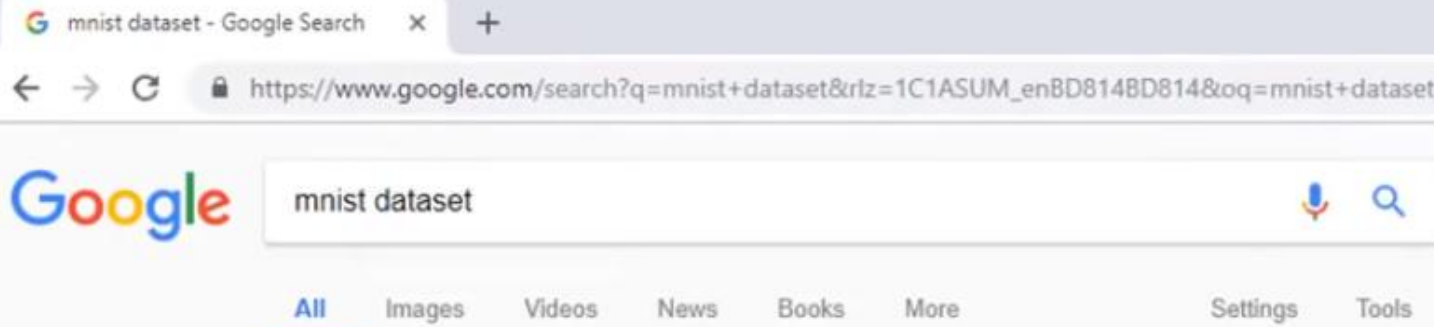


# CNN architecture: training





# Downloading MNIST dataset



About 312,000 results (0.35 seconds)

**MNIST handwritten digit database, Yann LeCun, Corinna Cortes and ...**

[yann.lecun.com/exdb/mnist/](http://yann.lecun.com/exdb/mnist/)

The MNIST database of handwritten digits, available from this page, has a ... Therefore it was necessary to build a new database by mixing NIST's datasets.

You've visited this page 2 times. Last visit: 9/14/18

**MNIST database - Wikipedia**

[https://en.wikipedia.org/wiki/MNIST\\_database](https://en.wikipedia.org/wiki/MNIST_database)

Jump to **Dataset** - The MNIST database is a large database of handwritten digits for training various image processing systems.

Performance · Classifiers

**Digit Recognizer | Kaggle**

<https://www.kaggle.com/c/digit-recognizer>

Jul 25, 2012 - Learn computer vision fundamentals with the famous Digit Recognizer competition. Your goal is to correctly identify digits from a dataset of tens of thousands of handwritten digits.

large subset MNIST challenge Sep 26, 2017

mnist-classification Sep 26, 2017

MNIST Tutorial Machine Learning Challenge Sep 25, 2017

Best up to date result on MNIST dataset Feb 1, 2017

More results from [www.kaggle.com](http://www.kaggle.com)

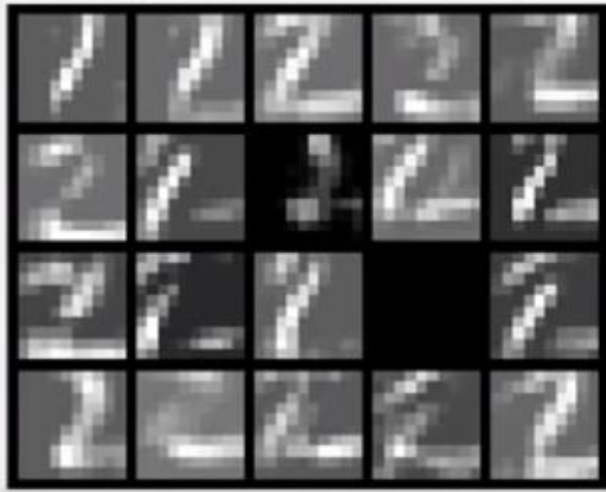
<a href="#"><u>train-images-idx3-ubyte.gz</u></a> :	training set images (9912422 bytes)
<a href="#"><u>train-labels-idx1-ubyte.gz</u></a> :	training set labels (28881 bytes)
<a href="#"><u>t10k-images-idx3-ubyte.gz</u></a> :	test set images (1648877 bytes)
<a href="#"><u>t10k-labels-idx1-ubyte.gz</u></a> :	test set labels (4542 bytes)

**please note that your browser may uncompress these files without telling them to remove the .gz extension. Some people have asked me "my application program to read them. The file format is described at the bottom of this page"**

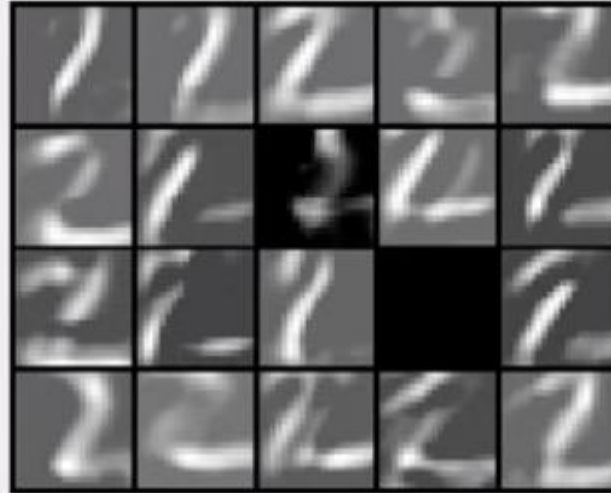


# Extracted features

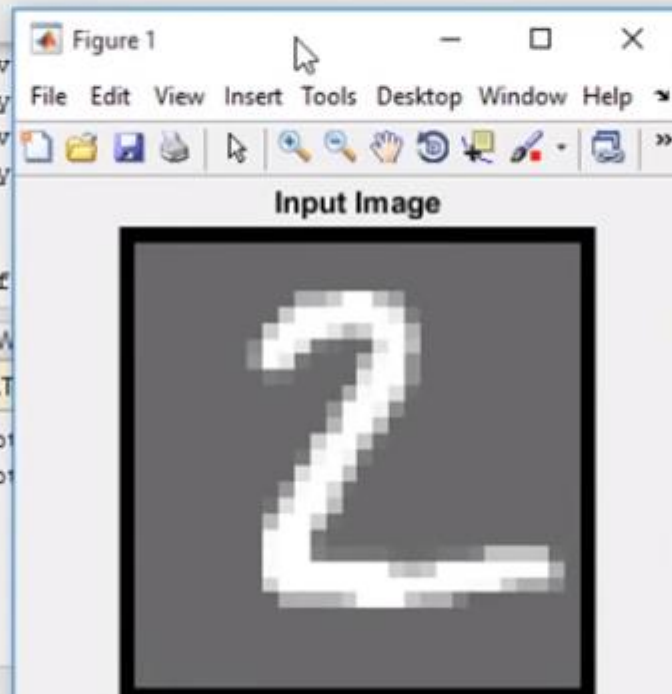
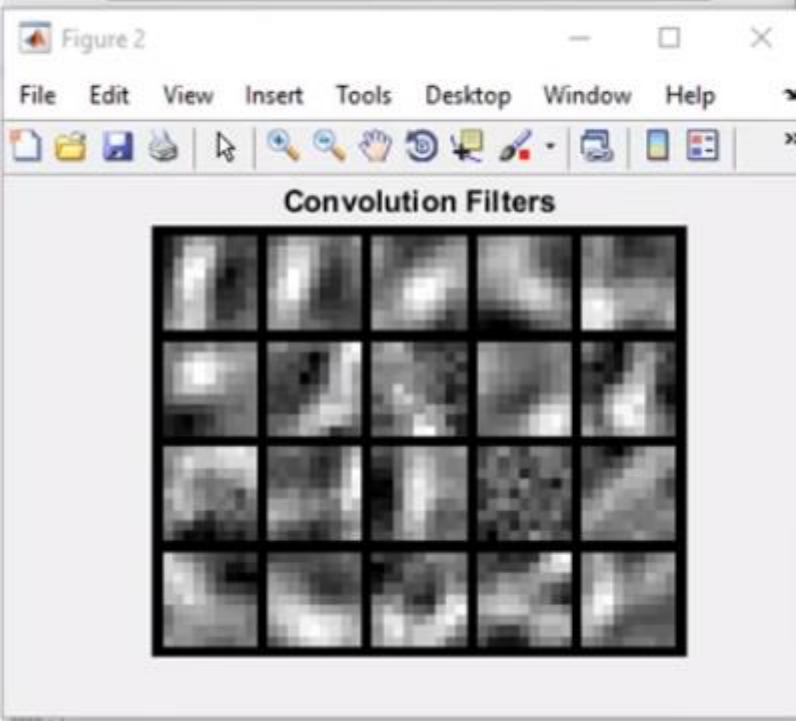
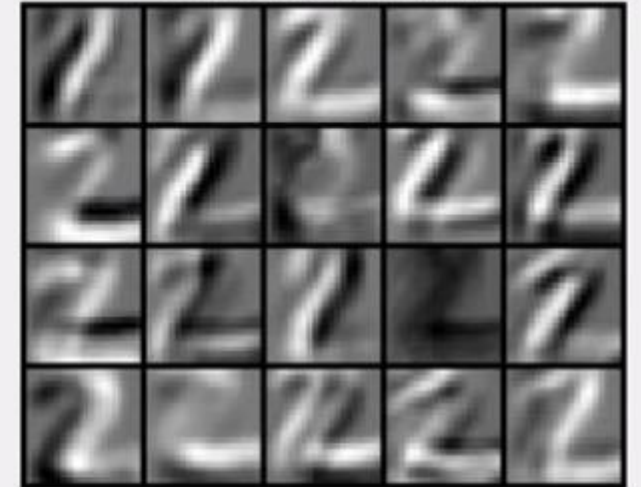
Features [Convolution + ReLU + MeanPool]



Features [Convolution + ReLU]



Features [Convolution]



360

10

# Matlab Implementation

- Data in Matlab:
  - C:\Program Files\MATLAB\R2020b\toolbox\nnet\nndemos\ndata\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,'Lab  
elsource','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')
```

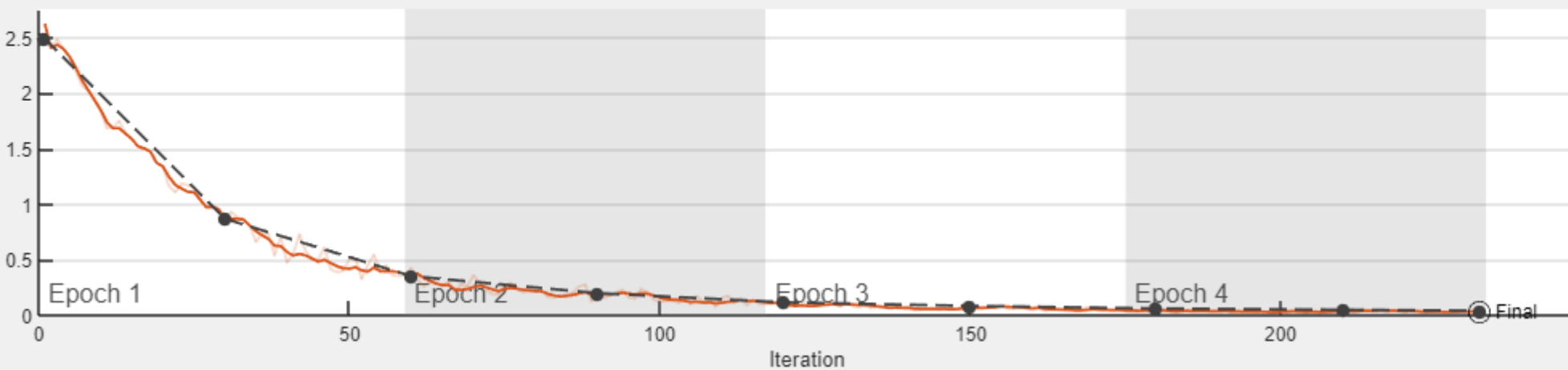
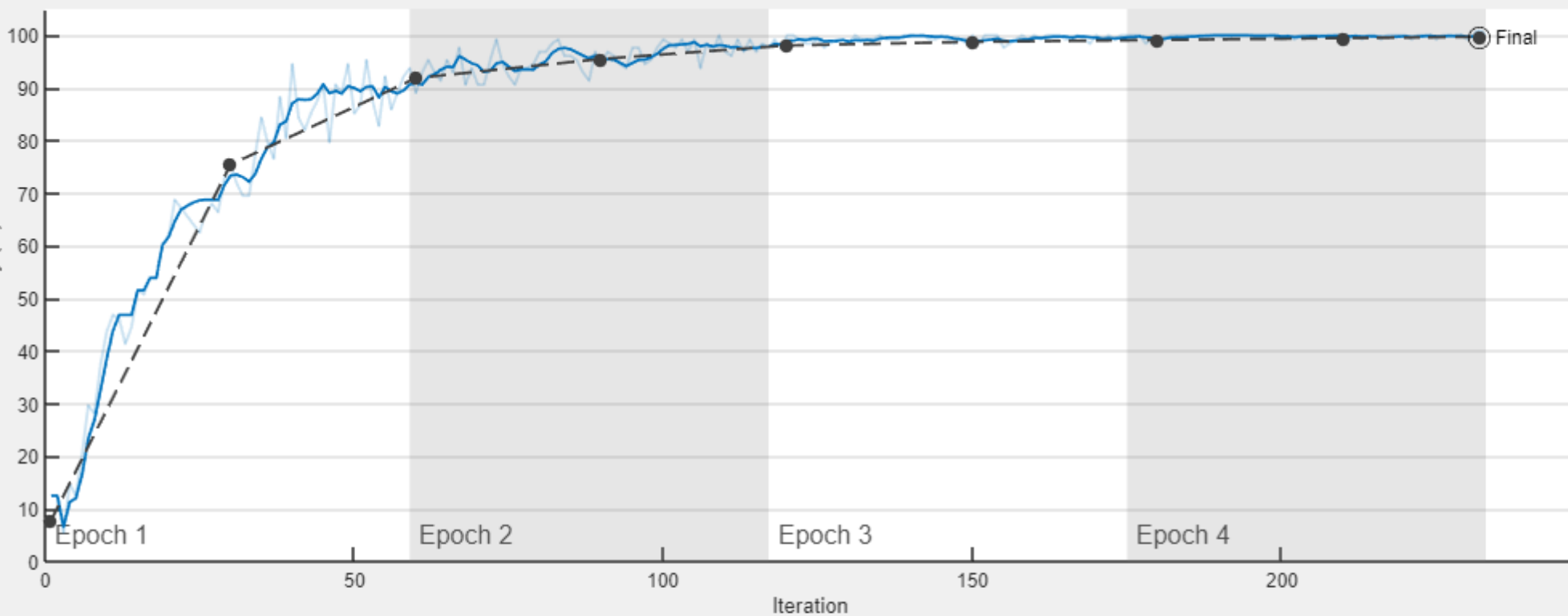
```
convolution2dLayer(3,16,'Padding','same','Name','Conv_2')  
batchNormalizationLayer('Name','BN_2')  
reluLayer('Name','Relu_2')  
maxPooling2dLayer(2,'Stride',2,'Name','Maxpool_2')
```

```
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')`

## Training Progress (21-Jul-2021 19:27:12)



## 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

[i Learn more](#)

## Accuracy

- Training (smoothed)
- Training
- Validation

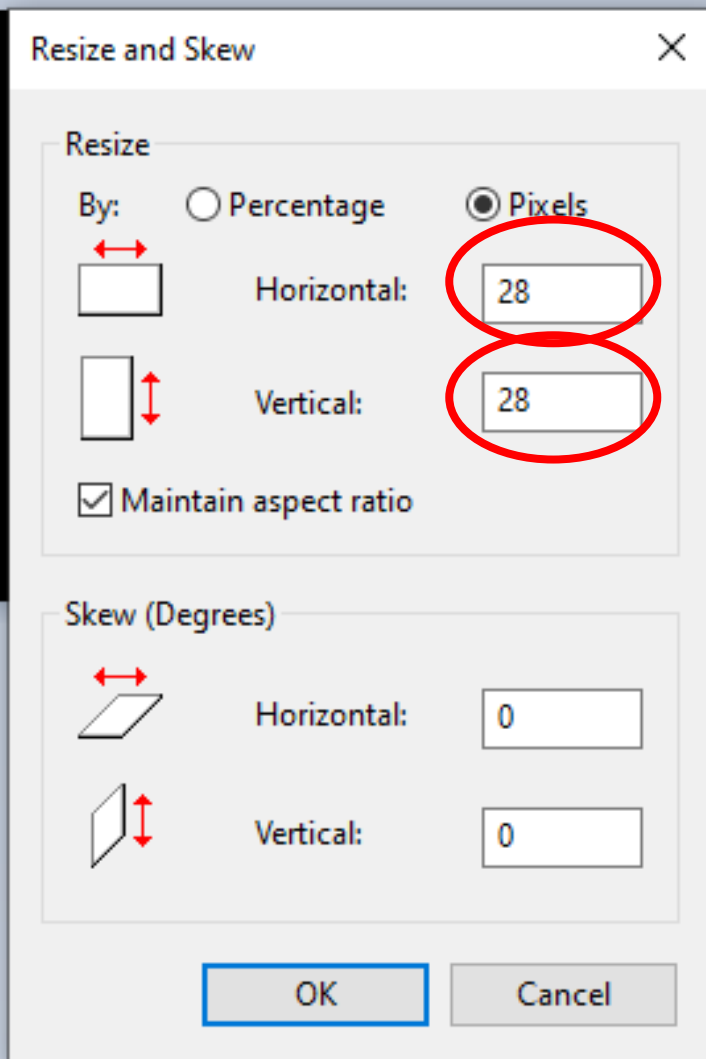
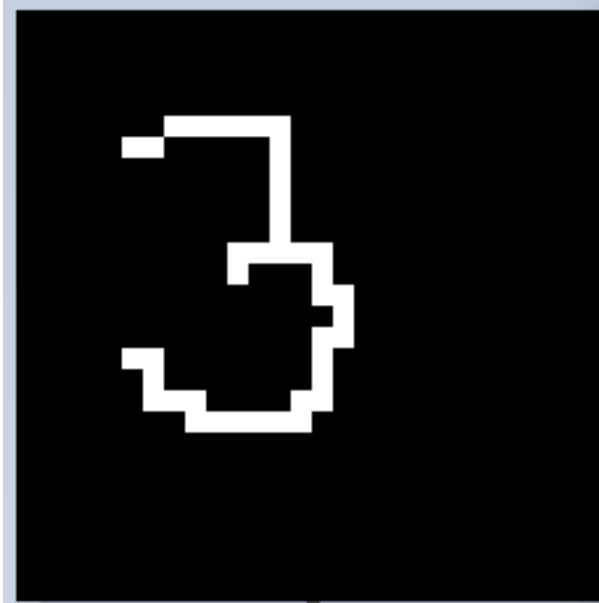
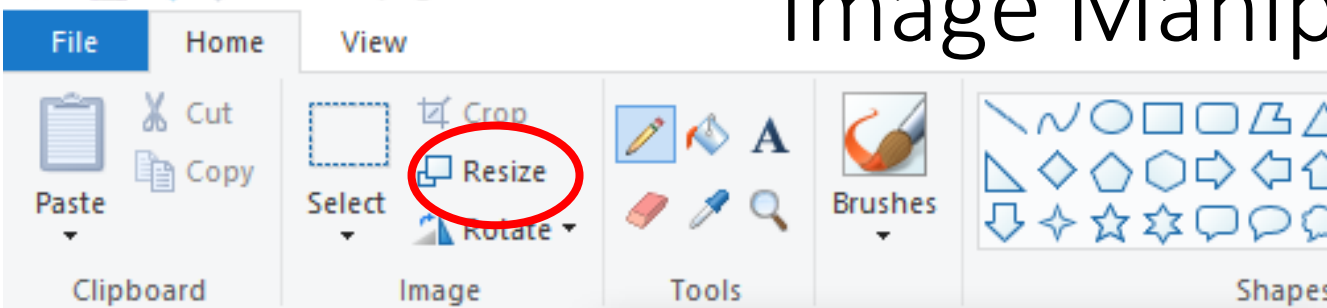
## Loss

- Training (smoothed)
- Training
- Validation

- `YPred=classify(net,TestImages)`
- `YValidation=TestImages.Labels`
- `accuracy=sum(YPred == YValidation)/numel(YValidation)`

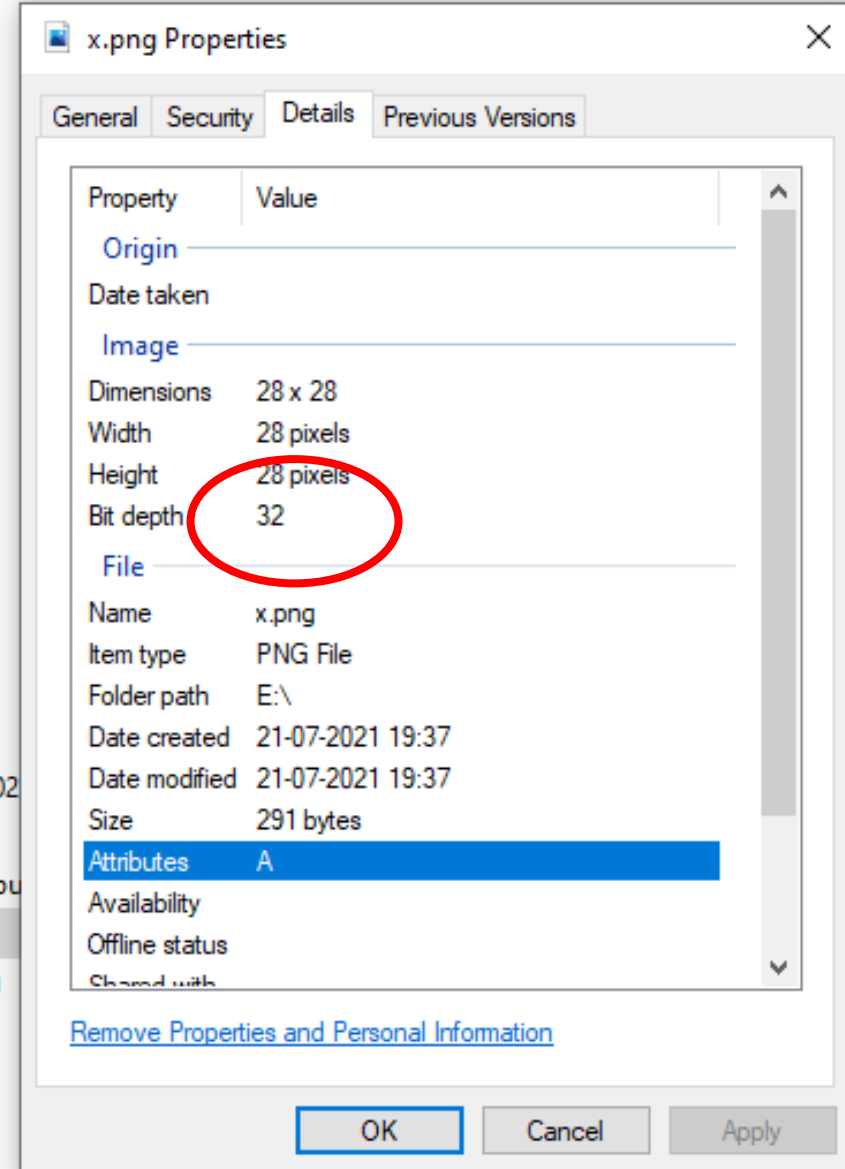
# Image Manipulation

kks.png - Paint



Name

- 2012
- 2013
- 2014
- 2017
- 2018
- 2019
- 2020
- 2021
- EBM\_SR
- GIAN IIT 2019
- IMP\_MISC
- Kis\_Docs
- MSL
- OTHERS\_Kis
- Papers\_from\_202
- Personal
- SMMME\_new\_bu
- x.png
- image2030.png
- yyyx.png
- kks.png
- kks\_8.png



# 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)`