Intro to Numpy, Pandas, PyTorch and Keras

Sundharakumar KB

Department of Computer Science and Engineering
School of Engineering

Shiv Nadar University Chennai

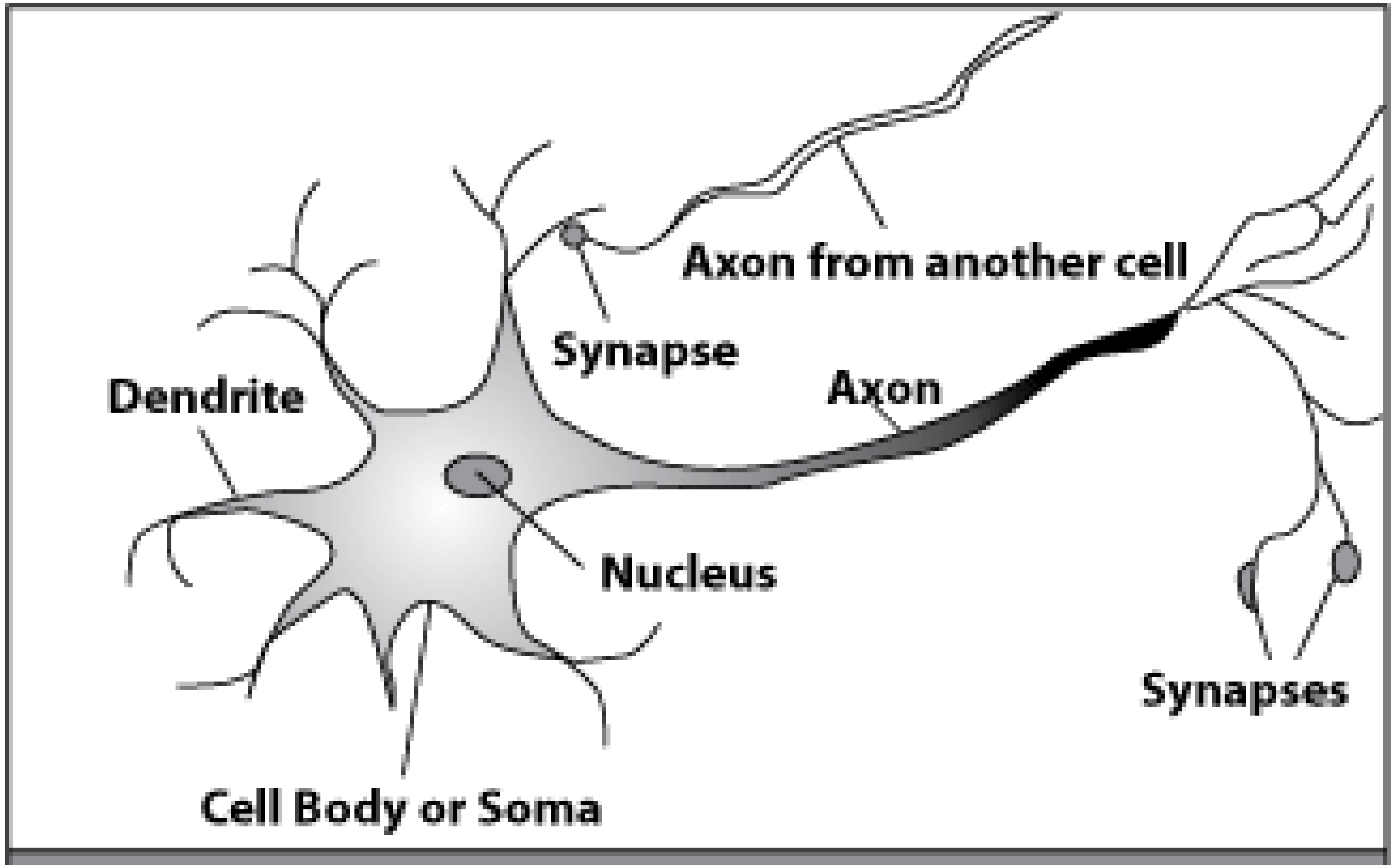
Agenda

- Numpy & Pandas intro
- Artificial Neural Network
- Implementation with Keras
- What is PyTorch?
- PyTorch and Numpy functions
- Perceptron using numpy/torch
- Autograd
- Multi layer perceptron for MNIST
- Performance difference in CNN

How Neural Networks work?

Neurons:

Biological Neuron

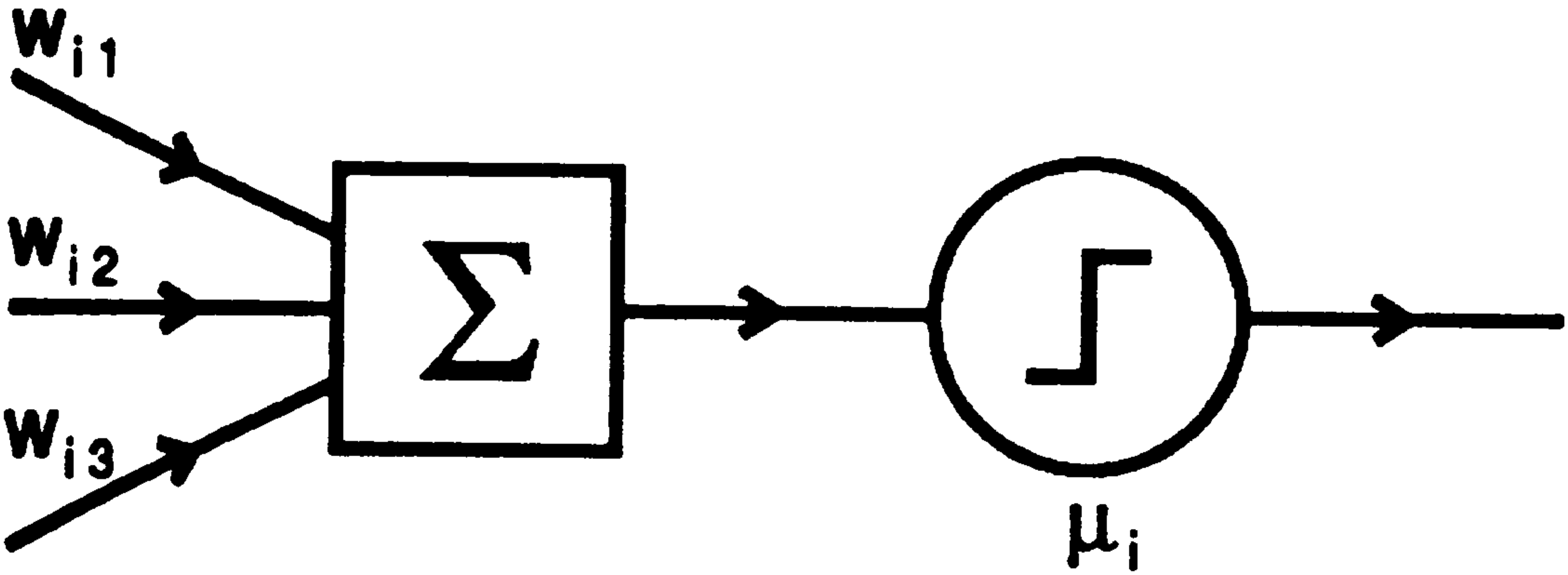


SHIV NADAR UNIVERSITY CHENNAI

Neurons in brain

- Although heterogeneous, at a low level, the brain is composed of neurons.
- A neuron receives input from other neurons (generally thousands) from its synapses.
- Inputs are summed up.
- When the input exceeds a threshold, the neuron sends a spike (electrical pulse) that travels to next neuron(s) via axons.

McCulloh-Pitts Neuron



SHIV NADAR UNIVERSITY CHENNAI

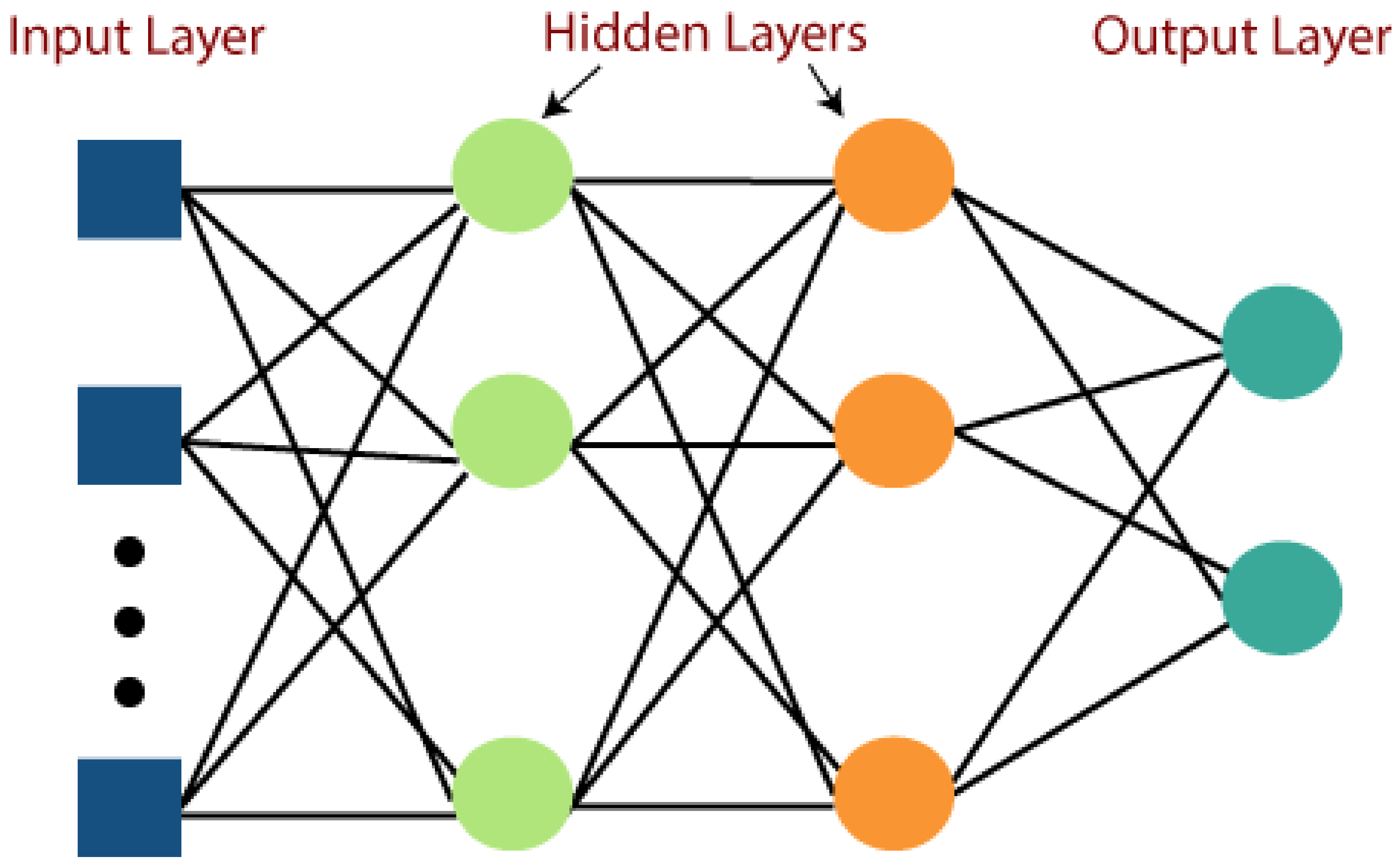
Single Layer perceptron

- Initialize w_{ij} with random values.

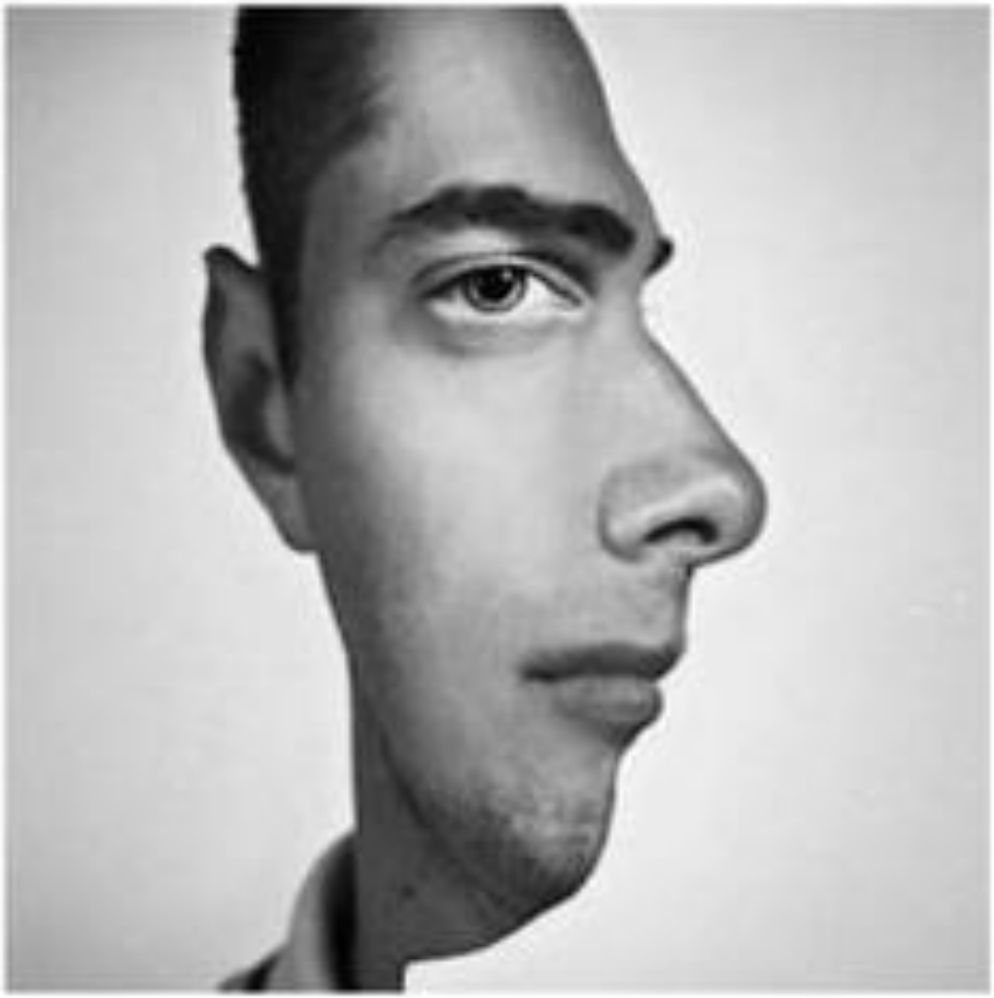
- Repeat until w_{ij} (t+1) ~= w_{ij} (t)

- $w_{new} = w_{old} - \eta \frac{\partial E}{\partial w}$

Multi Layer perceptron

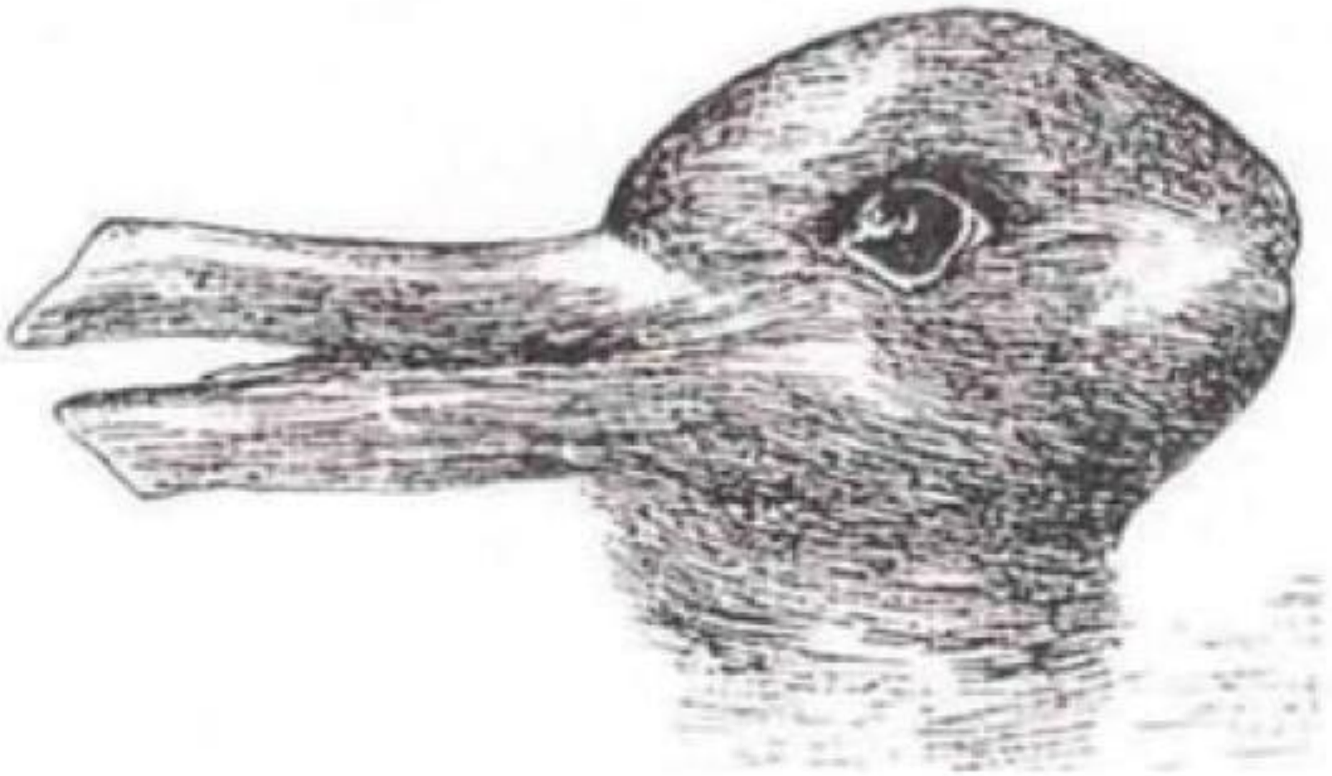


Convolutional Neural Network



Convolutional Neural Network

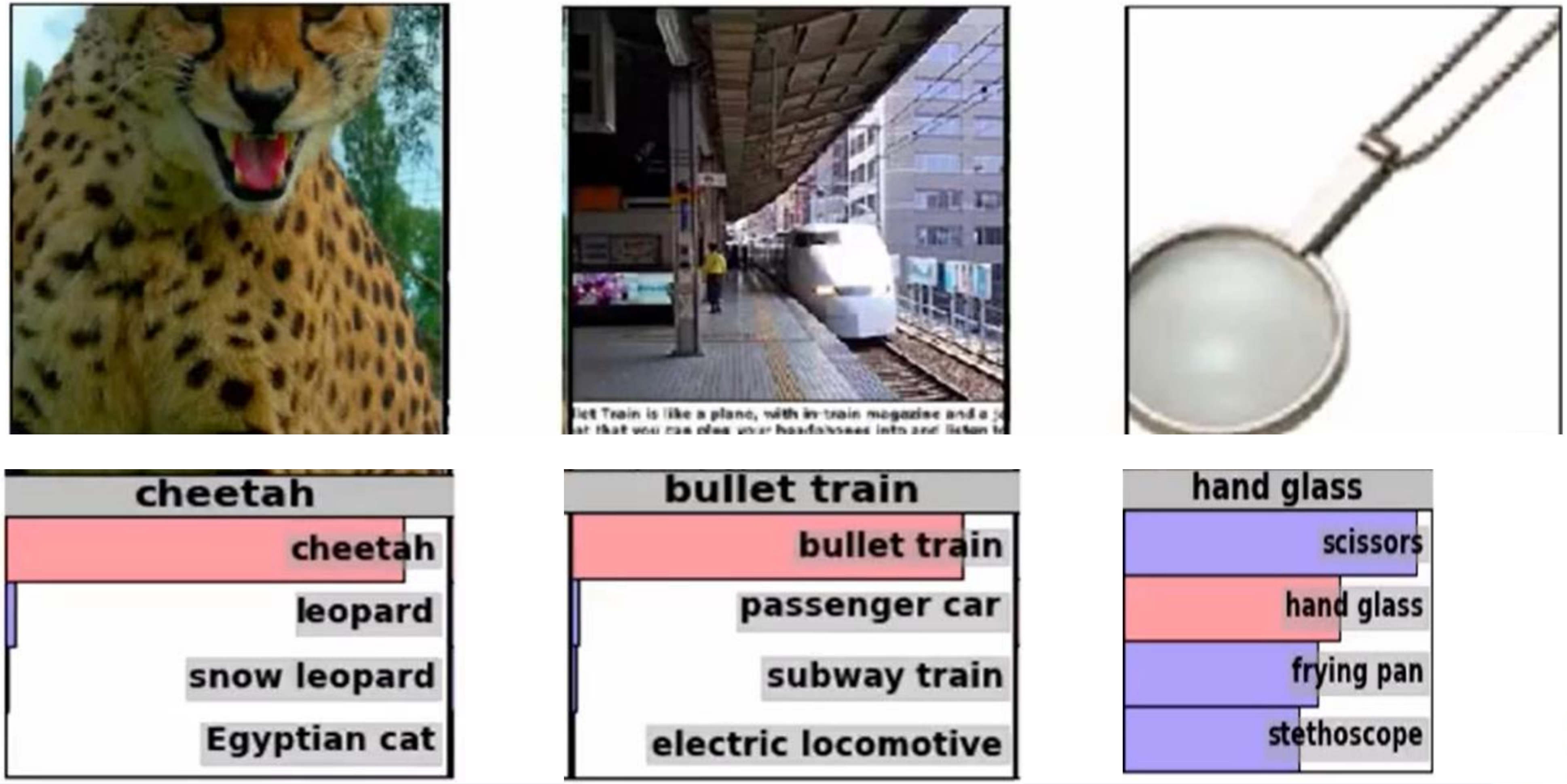
Convolutional Neural Network



SHIV NADAR UNIVERSITY CHENNAI

Convolutional Neural Network

Examples from the test set (with the network's guesses)

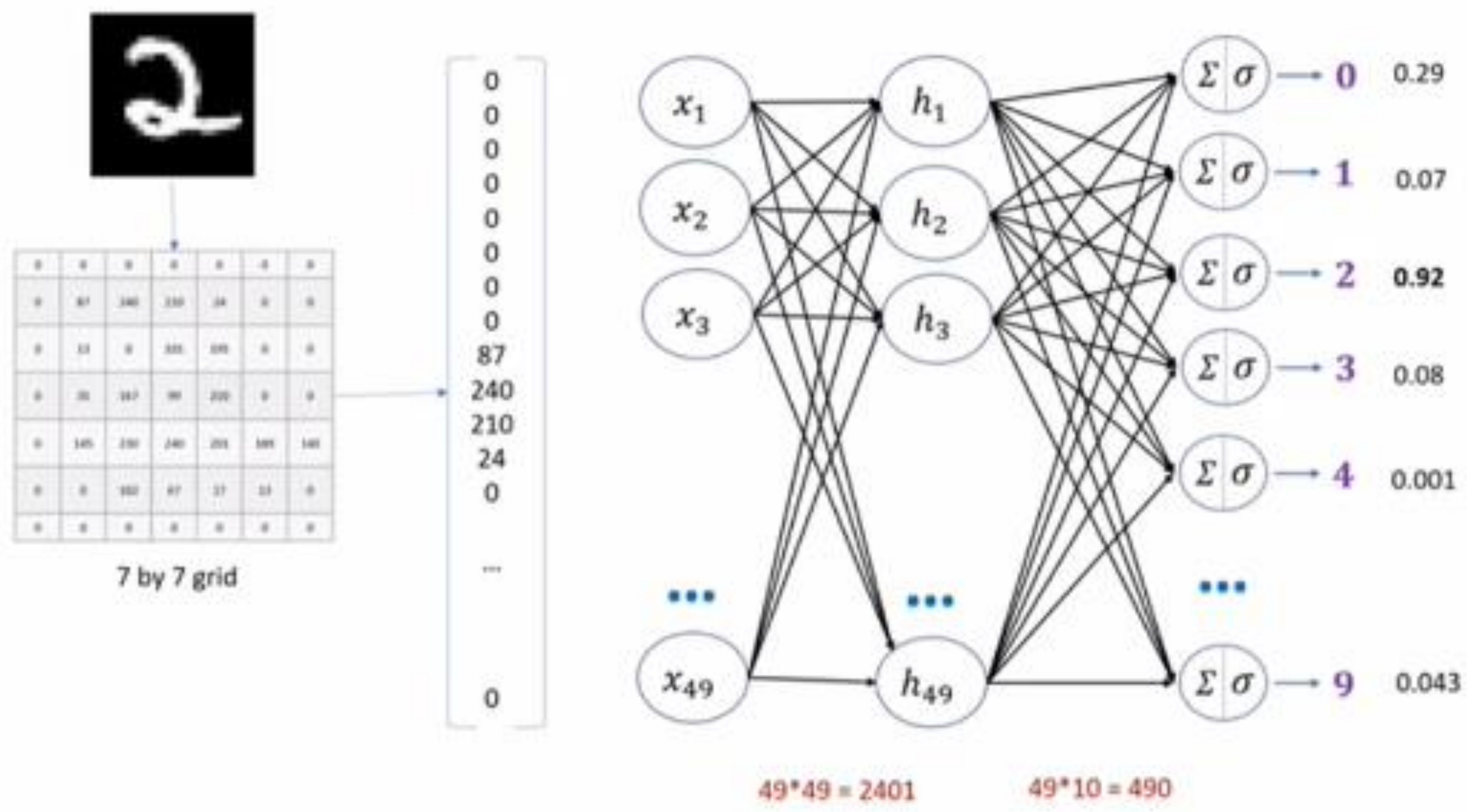


Img source: Geoffrey Hinton

Convolutional Neural Network

Convolutional Neural Network

Digit recognition using ANN



What about color images?

- Image size = 1920 * 1080 * 3
- First layer neurons = 6 million (approx.)
- Hidden layer neurons = 4 million (approx.)
- Weights between first layer and hidden layer = 6M * 4M = 24 millions.

Image size = 1920 X 1080 X 3

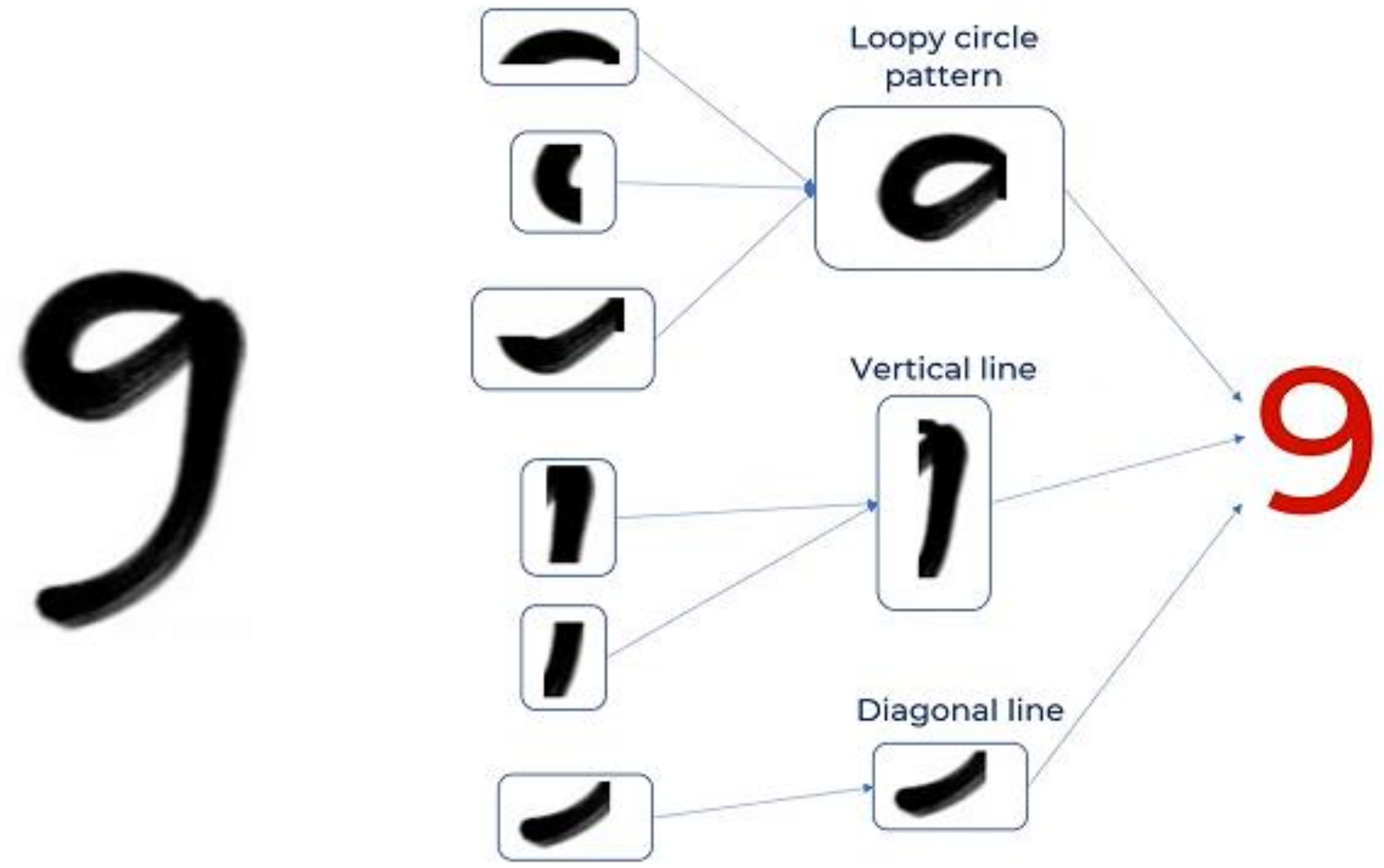
Drawbacks of ANN for Image classification

- Enormous computations
- Treats local pixels same as pixels far apart
- Sensitive to location of an object in an image.

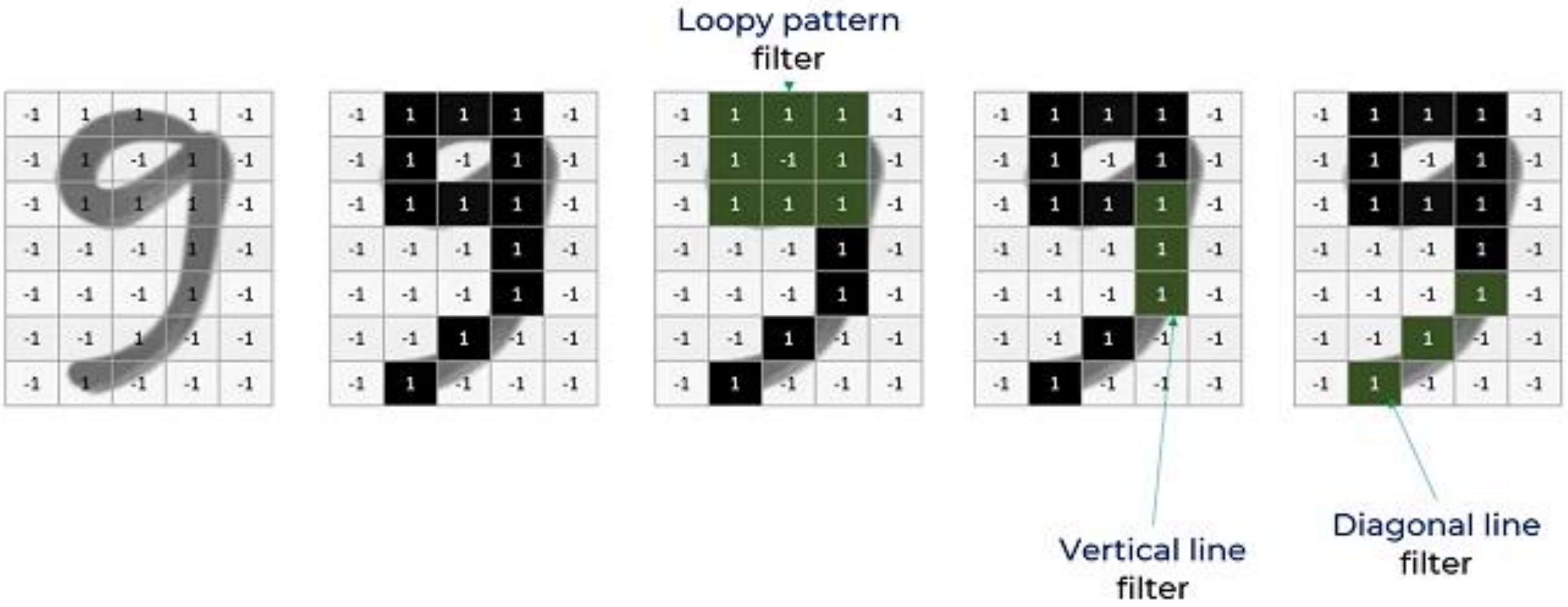
CNN steps

- Convolution operation
  - Generating feature maps
- Pooling layer
- Flattening
- Full connection

CNN advantage



Feature detection



Convolution operation

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

1	1	1
1	-1	1
1	1	1

Convolution operation

$$-1+1+1-1-1-1-1+1+1 = -1 \rightarrow -1/9 = -0.11$$

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

-0.11		

Convolution operation

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

*

1	1	1
1	-1	1
1	1	1

-0.11	1	-0.11
-0.55	0.11	-0.33

Convolution operation

-1	1	1	1	-1
-1	1	-1	1	-1
-1	1	1	1	-1
-1	-1	-1	1	-1
-1	-1	-1	1	-1
-1	-1	1	-1	-1
-1	1	-1	-1	-1

*

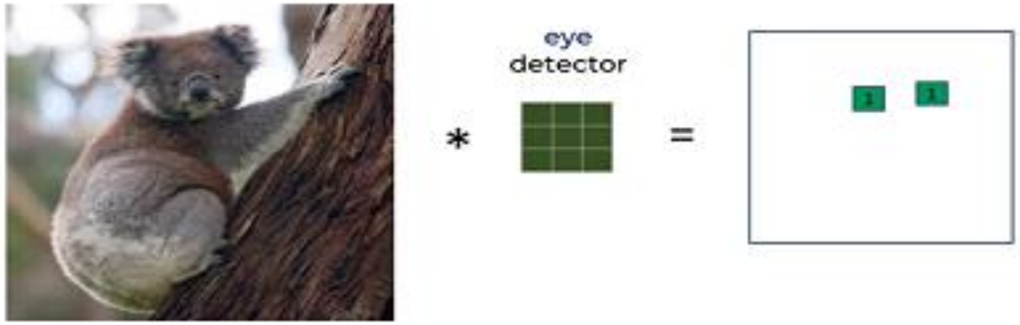
1	1	1
1	-1	1
1	1	1

-0.11	1	-0.11
-0.55	0.11	-0.33
-0.33	0.33	-0.33
-0.22	-0.11	-0.22
-0.33	-0.33	-0.33

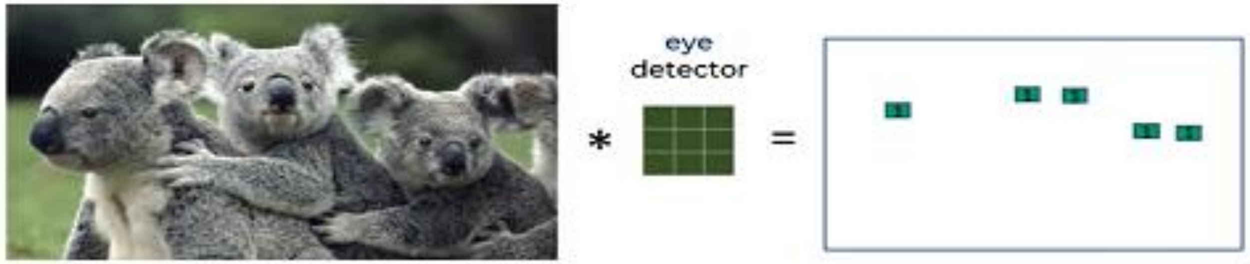
Feature Map

Convolution operation

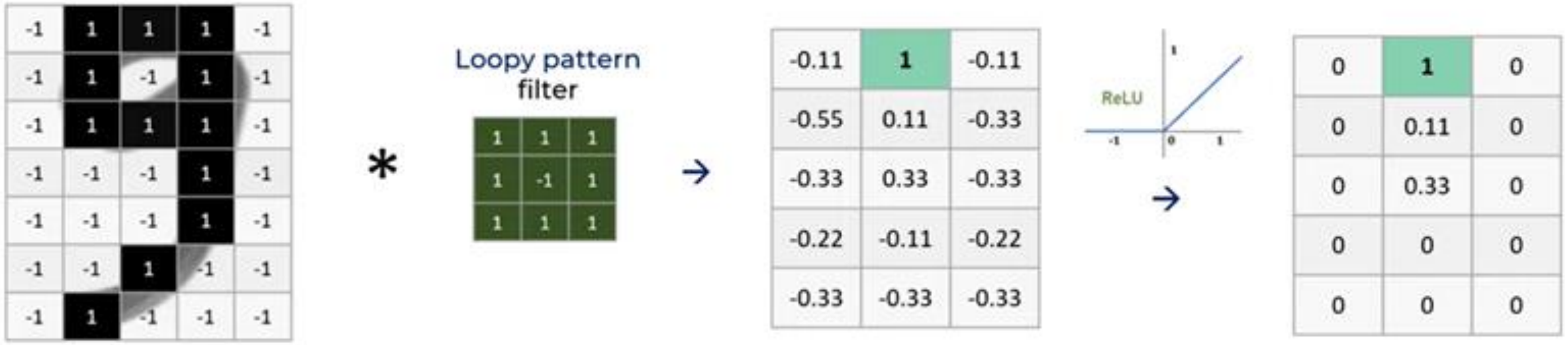
Convolution operation



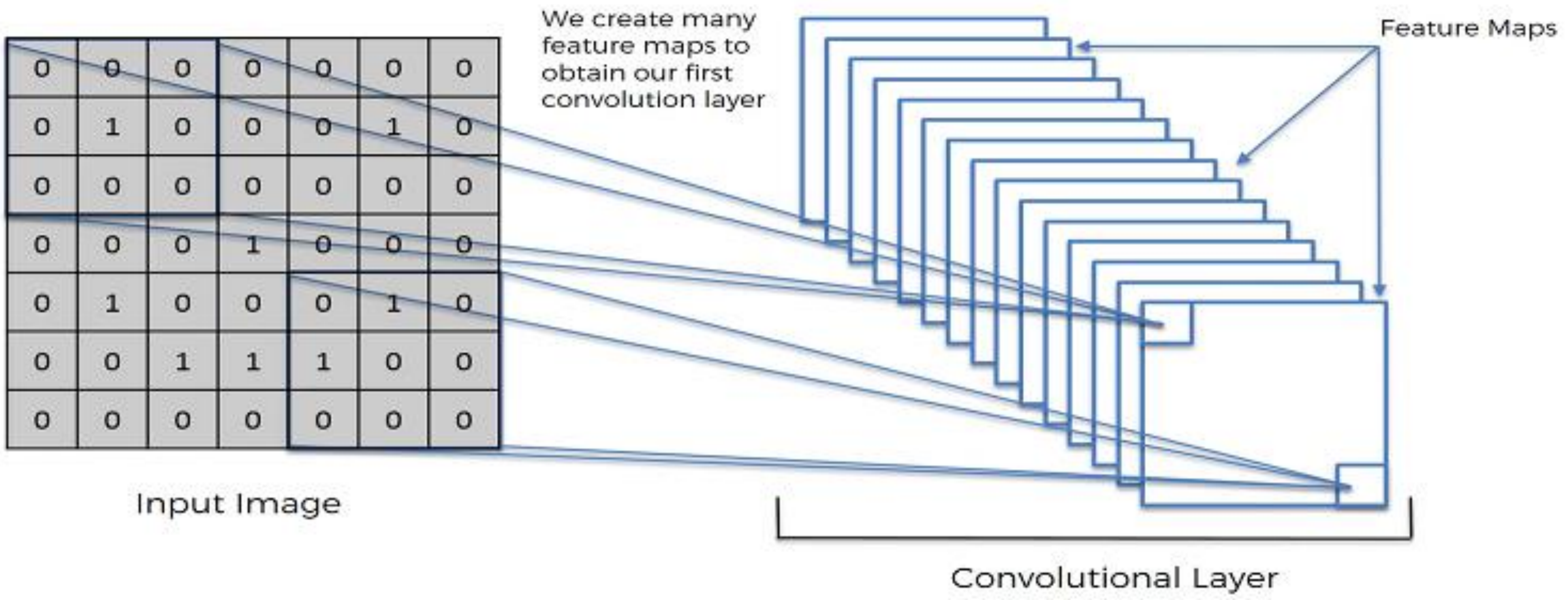
Location invariant: It can detect eyes in any location of the image



Convolution operation



Convolution operation



SHIV NADAR UNIVERSITY CHENNAI

Sharpen feature

0	0	0	0	0
0	0	-1	0	0
0	-1	5	-1	0
0	0	-1	0	0
0	0	0	0	0

SHIV NADAR UNIVERSITY CHENNAI

Blur feature

0	0	0	0	0
0	1	1	1	0
0	1	1	1	0
0	1	1	1	0
0	0	0	0	0

Edge Detect

	0	1	0	
	1	-4	1	
	0	1	0	

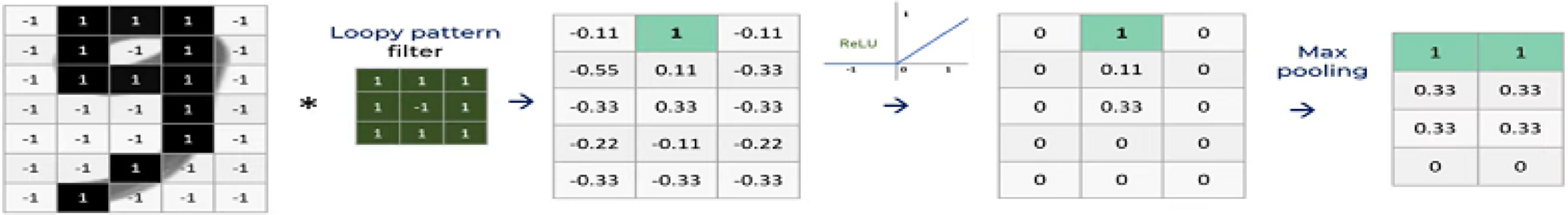
Pooling

5	1	3	4
8	2	9	2
1	3	0	1
2	2	2	0

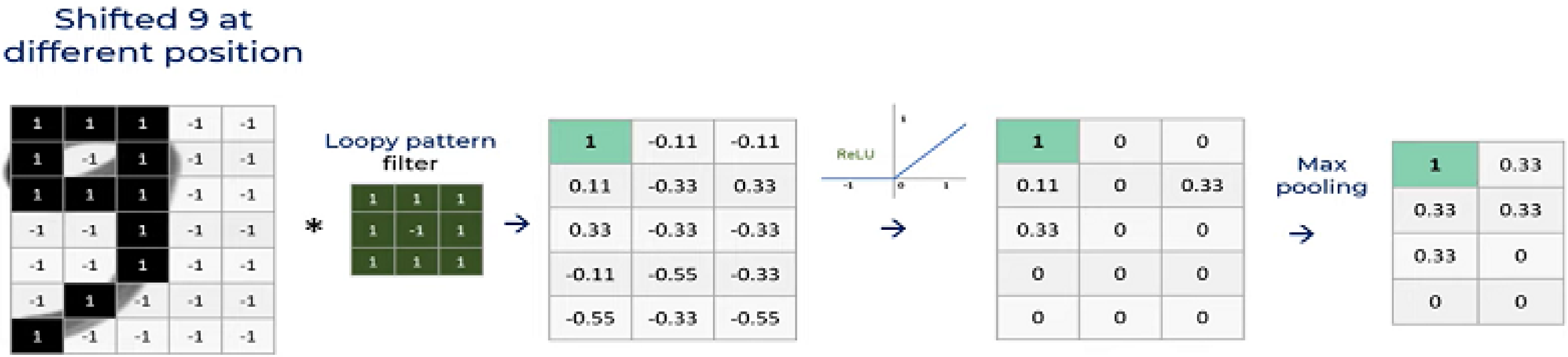
8	9
3	2

2 by 2 filter with stride = 2

Max Pooling



Max Pooling


Shifted 9 at
different position
1 1 1 -1 -1
1 -1 1 -1 -1
1 1 1 -1 -1
-1 -1 1 -1 -1
-1 -1 1 -1 -1
-1 1 -1 -1 -1
1 -1 -1 -1 -1
*
Loopy pattern
filter
1 1 1
1 -1 1
1 1 1
1 -0.11 -0.11
0.11 -0.33 0.33
0.33 -0.33 -0.33
-0.11 -0.55 -0.33
-0.55 -0.33 -0.55
ReLU
1 0 0
0.11 0 0.33
0.33 0 0
0 0 0
0 0 0
Max
pooling
1 0.33
0.33 0.33
0.33 0
0 0

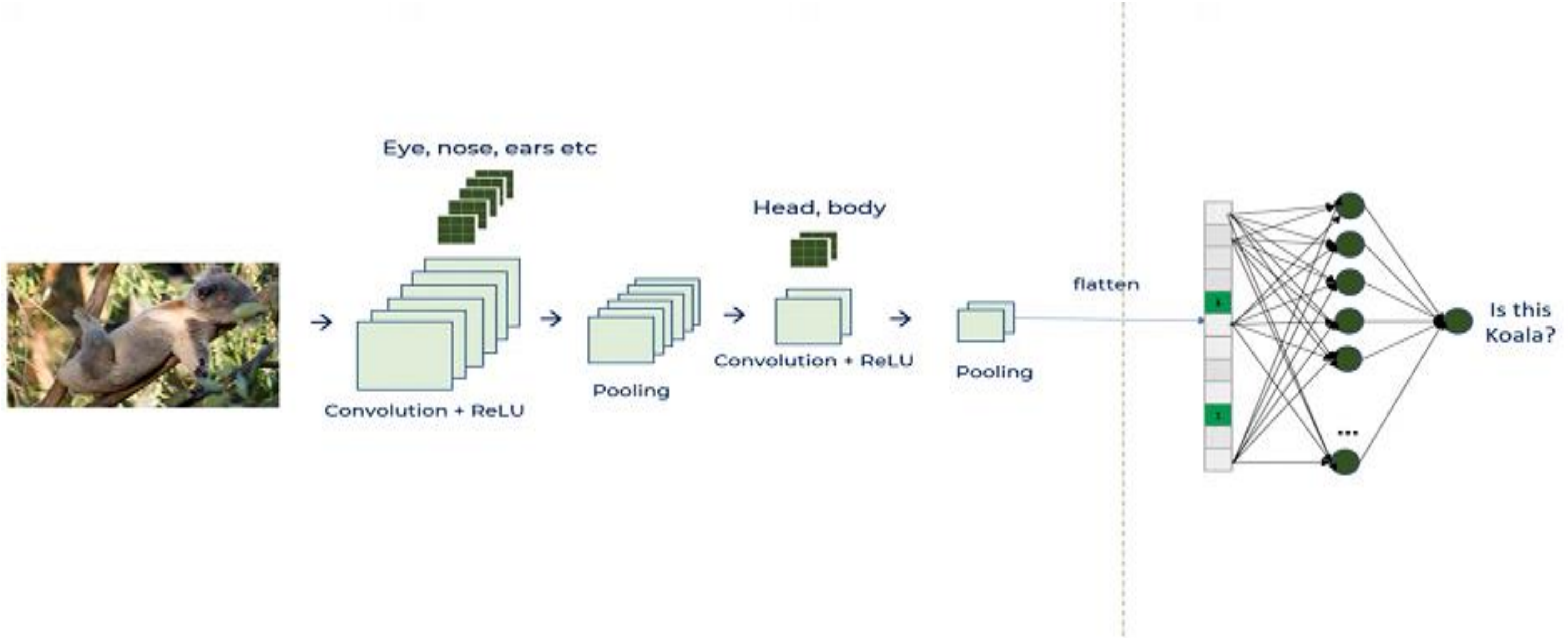
Benefits of Pooling

- Reduces dimensions & computation

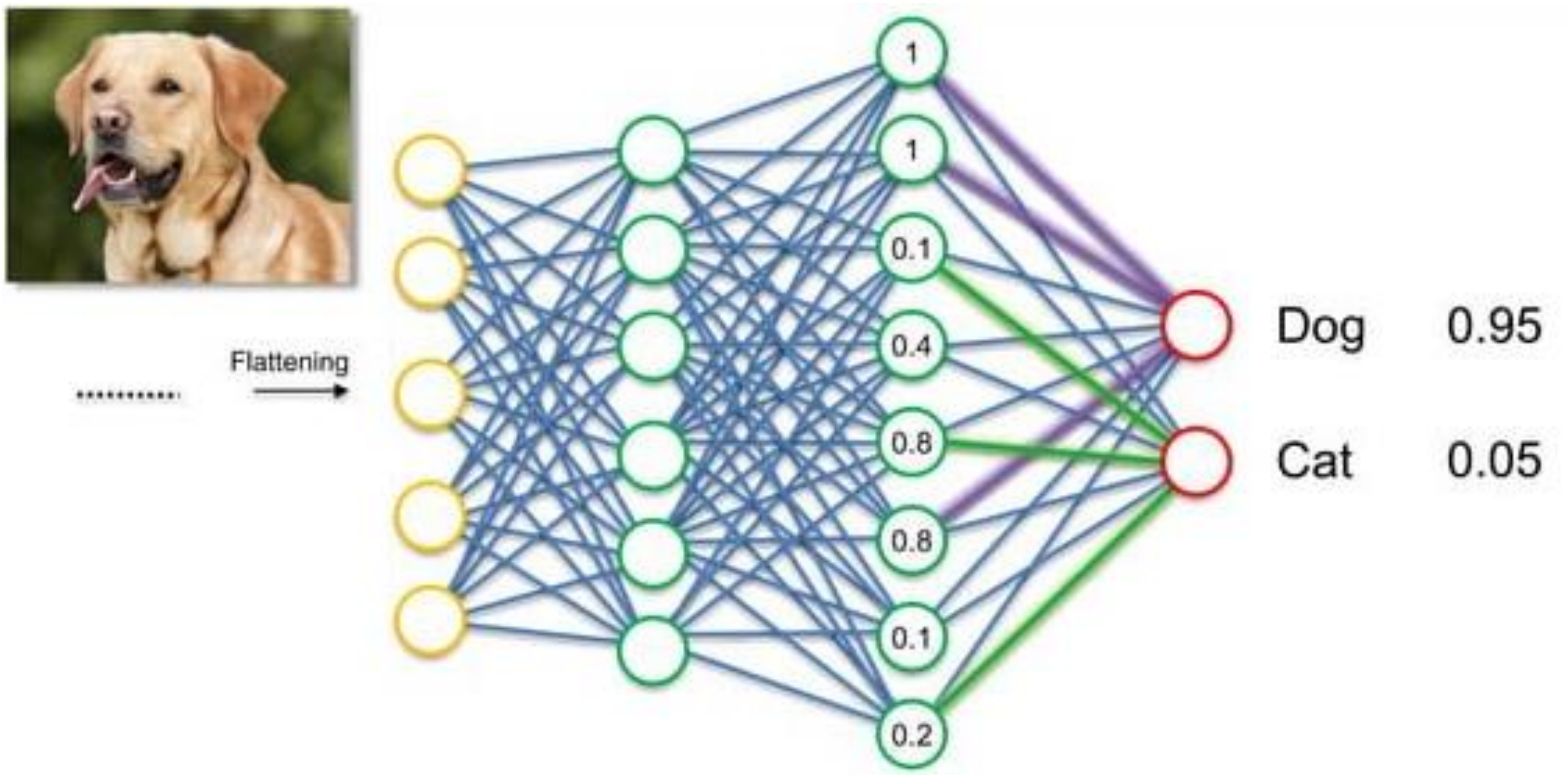
- Reduce overfitting as there are less params

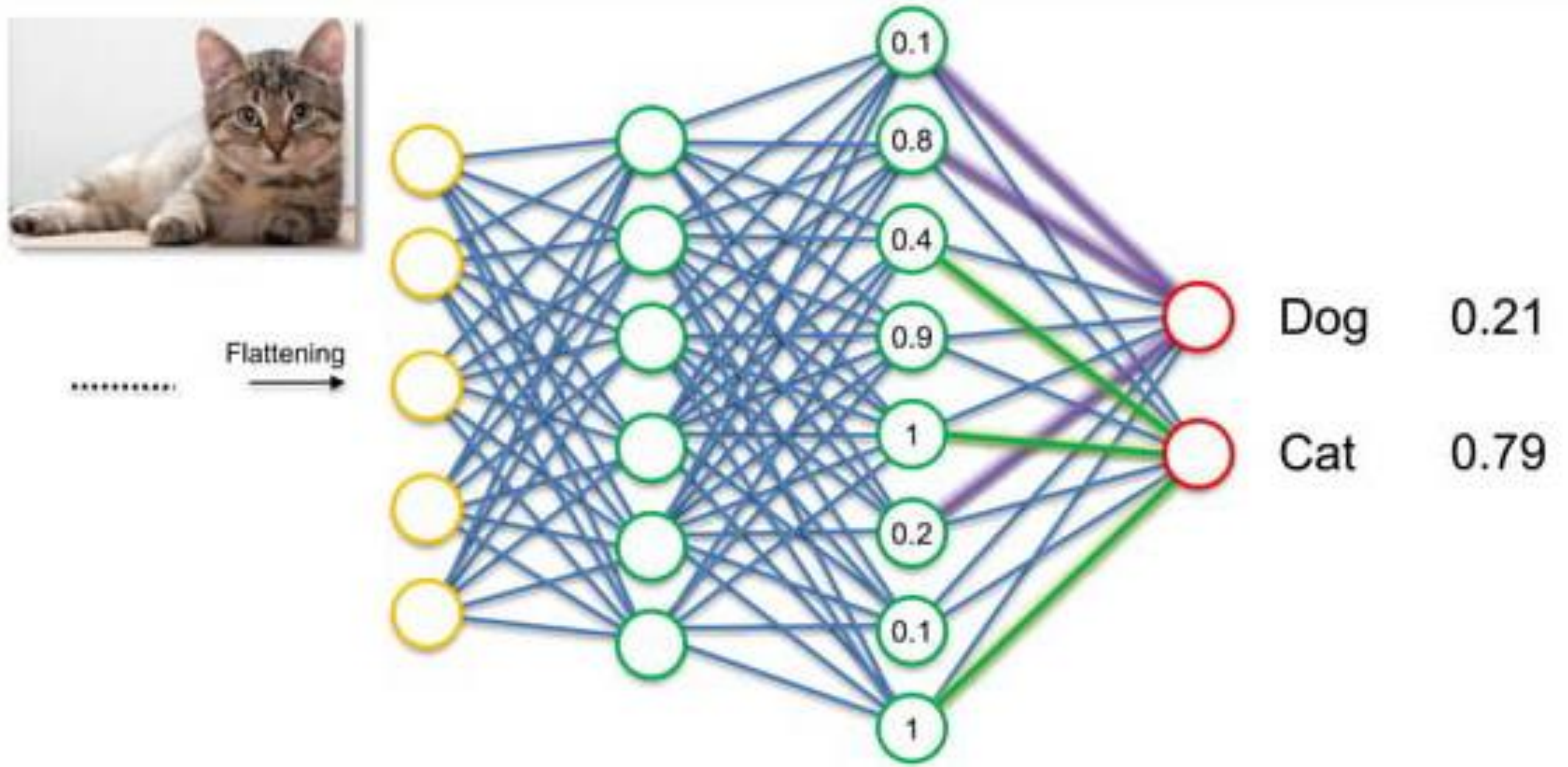
- Model is tolerant towards variations and distortions.

Fully connected CNN



SHIV NADAR UNIVERSITY CHENNAI


Flattening
1
1
0.1
0.4
0.8
0.8
0.1
0.2
Dog
0.95
Cat
0.05

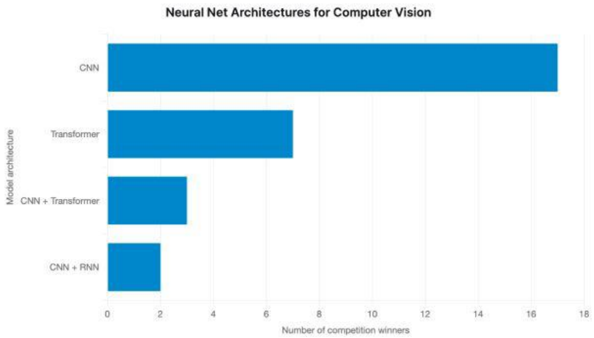

Flattening
0.1
0.8
0.4
0.9
1
0.2
0.1
1
Dog 0.21
Cat 0.79

PyTorch

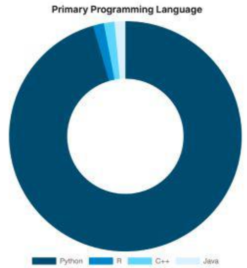
- PyTorch is a deep learning library
- Built by facebook and later made open-source
- Built over torch library (that uses **Lua**).
- Similar to numpy manipulations, exept that it is called as **Tensors** and has implicit **GPU** computational abilities.

PyTorch

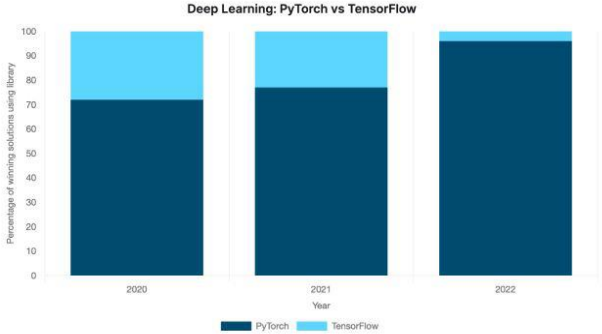
Convolutional neural networks still dominate computer vision



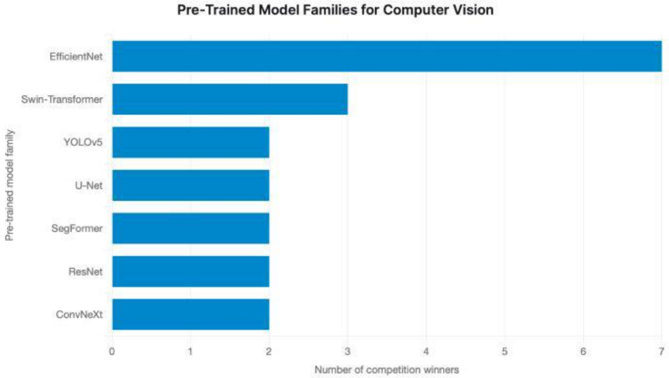
Almost everyone uses Python



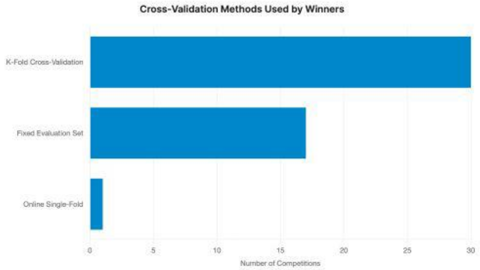
Among deep learning practitioners, almost everyone uses PyTorch



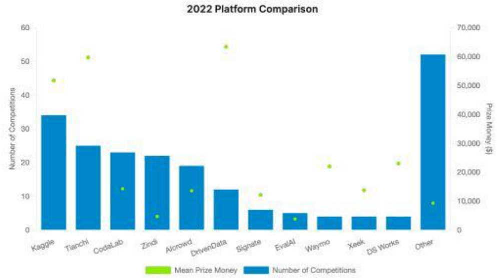
EfficientNet is the most popular pretrained architecture for computer vision



Almost twice as many winning solutions used k-fold CV instead of a fixed validation set



Kaggle remains the most popular competition platform



SHIV NADAR UNIVERSITY CHENNAI

PyTorch vs Numpy

```python
import torch

# Creating tensors
t = torch.tensor([1,2,3])
# Retrieving shape
print(t.shape)
# Accessing type of Data
print(t.dype)
# Changing datatype
t = t.to(torch.float32)
# Random number generation
m = torch.rand(1)
# Tensor with ones zeros
A1 = torch.ones(2,3)
```

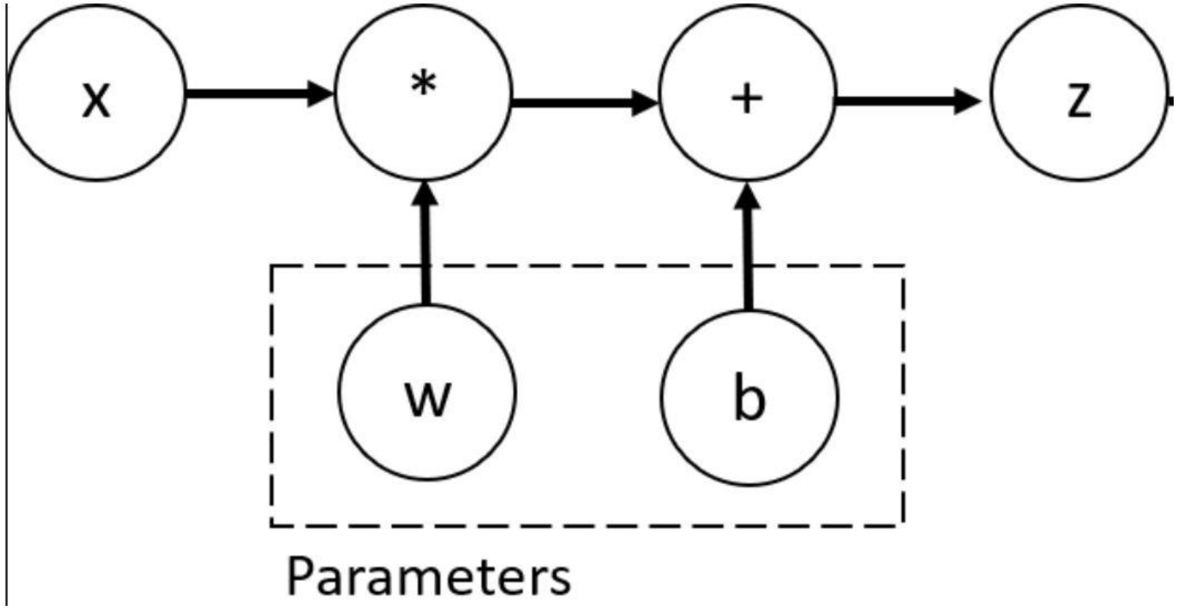
```python
import numpy as np

# Creating Arrays
a = np.array([4,5,6])
# Retrieving shape
print(a.shape)
# Accessing type of Data
print(a.dype)
# Changing datatype
a = a.astype('float32')
# Random number generation
n = np.rand(1)
# Array with ones
A2 = np.ones(2,3)
```

SHIV NADAR UNIVERSITY CHENNAI

AutoGrad

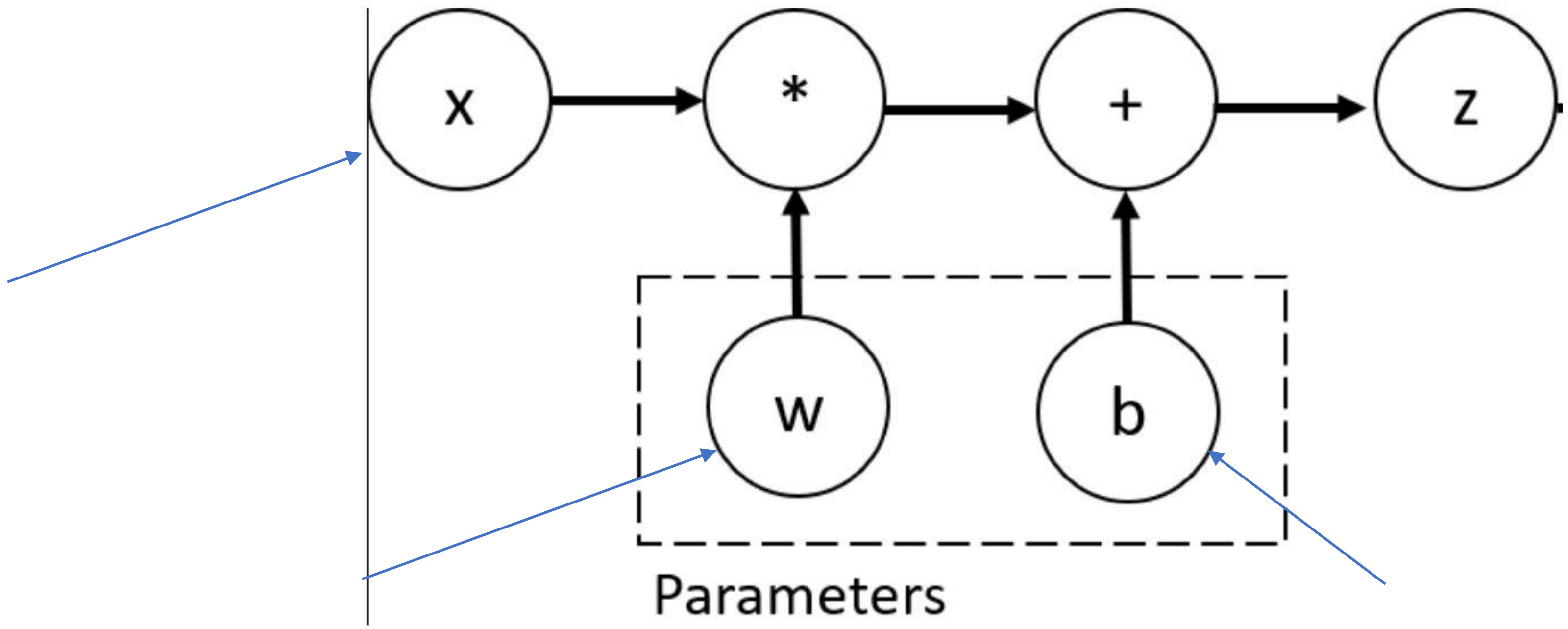
- Autograd is PyTorch’s differentiation engine

- It achieves the calculation of gradients of each weights by constructing a **Dynamic Directed Acyclic graph**



AutoGrad

- Gradients can be obtained only for the leaf nodes of the DAG using **.grad** attribute



AutoGrad

- After the forward propagation, i.e calculating the loss, gradients can be calculated with **.backward()** function.

- During forward propagation
  - Computes output
  - Keeps track of **grad_fn** at each node

- During back propagation
  - Gradients are calculated from leaf to root node based on the **.grad_fn** at that node
  - Accumulates gradient in **.grad** attribute of each node