Lecture 12

Deep Learning

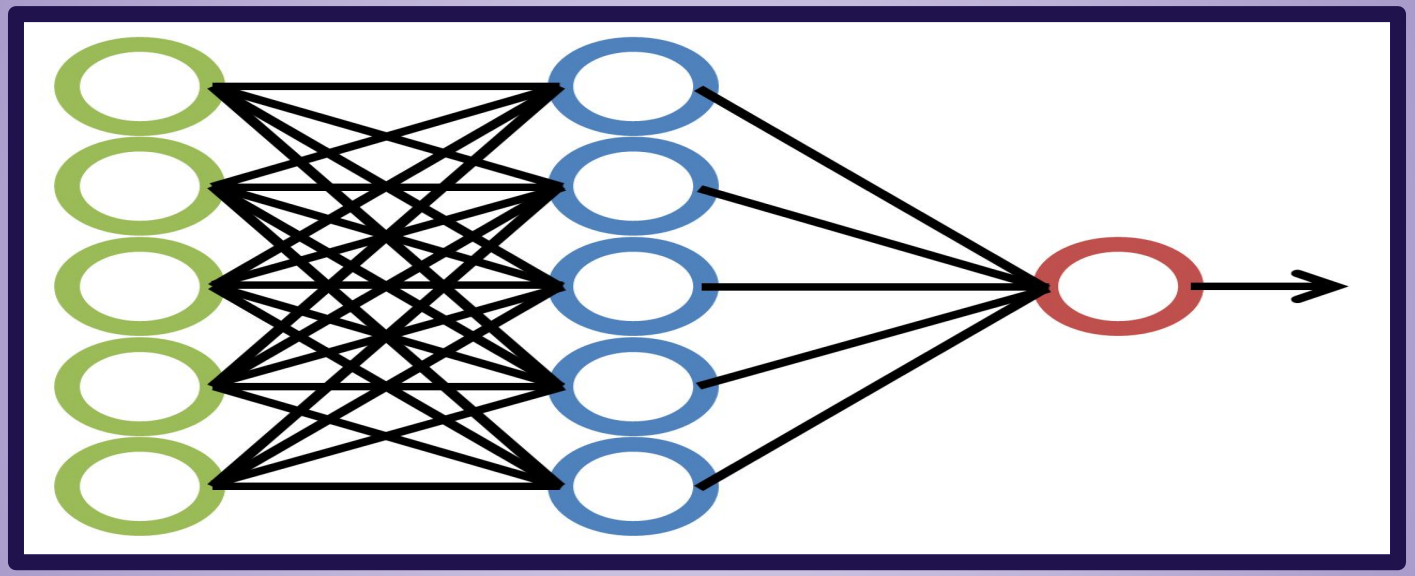
(Neural Networks)

AI/ML Foundation Course with Python

Introduction to Deep Learning

Deep Learning deals with artificial intelligence and machine learning.

Key : *Mimic Human Decision Making Capabilities*



Deep Neural Network

Data feeding takes place through neural networks.

Any Deep neural network consist of three types of layers:

- The Input Layer
- The Hidden Layer
- The Output Layer

Deep Learning

The input layer

It receives all the inputs and the last layer is the output layer which provides the desired output.

2. Hidden Layers

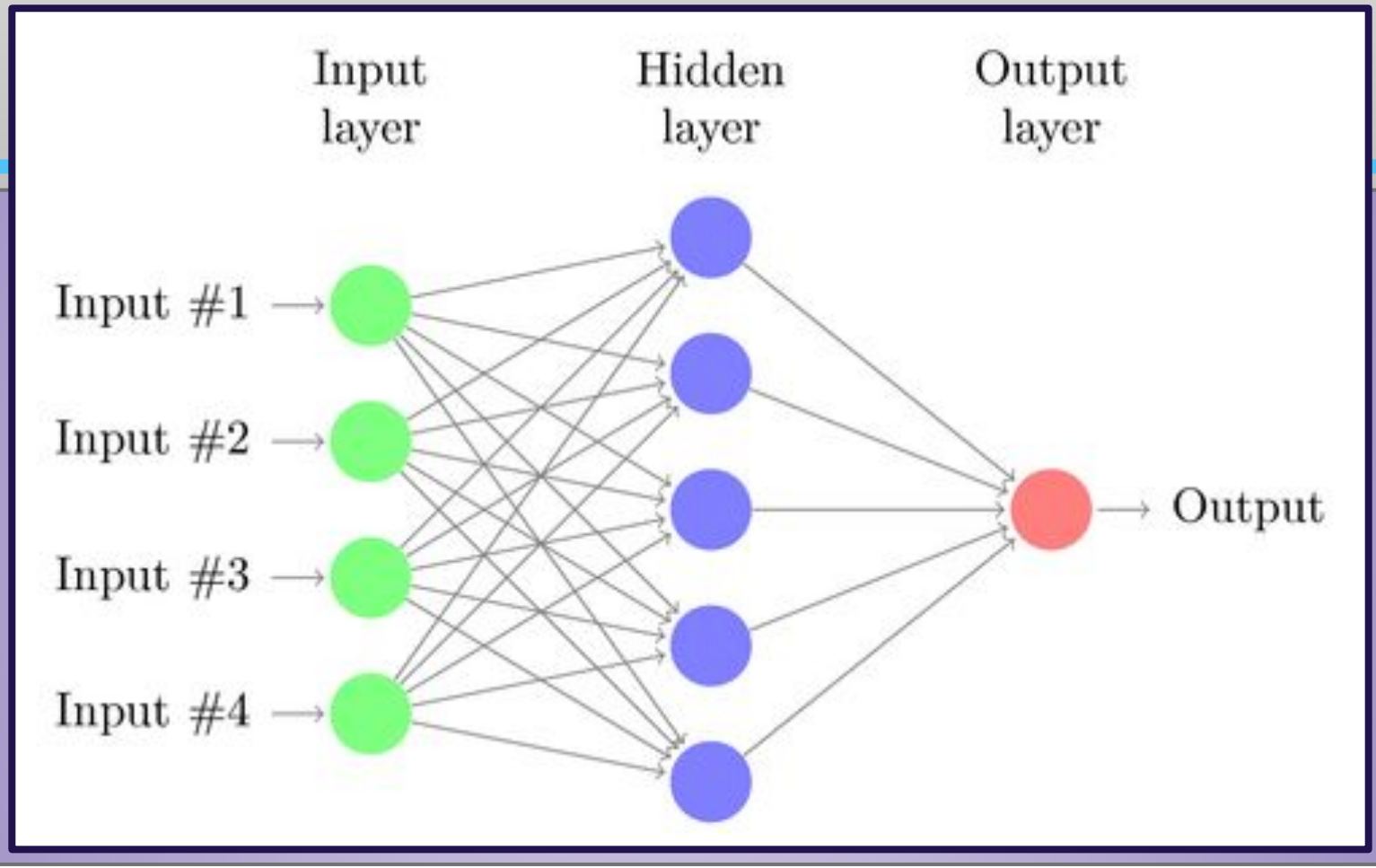
All the layers in between these layers are called hidden layers. There can be n number of hidden layers. The hidden layers and perceptrons in each layer will depend on the use-case you are trying to solve.

3. Output Layers

It provides the desired output.

Introduction to Neural Networks

- Produces best solutions to many problems like image recognition, speech recognition, and natural language processing.



Real Life Applications

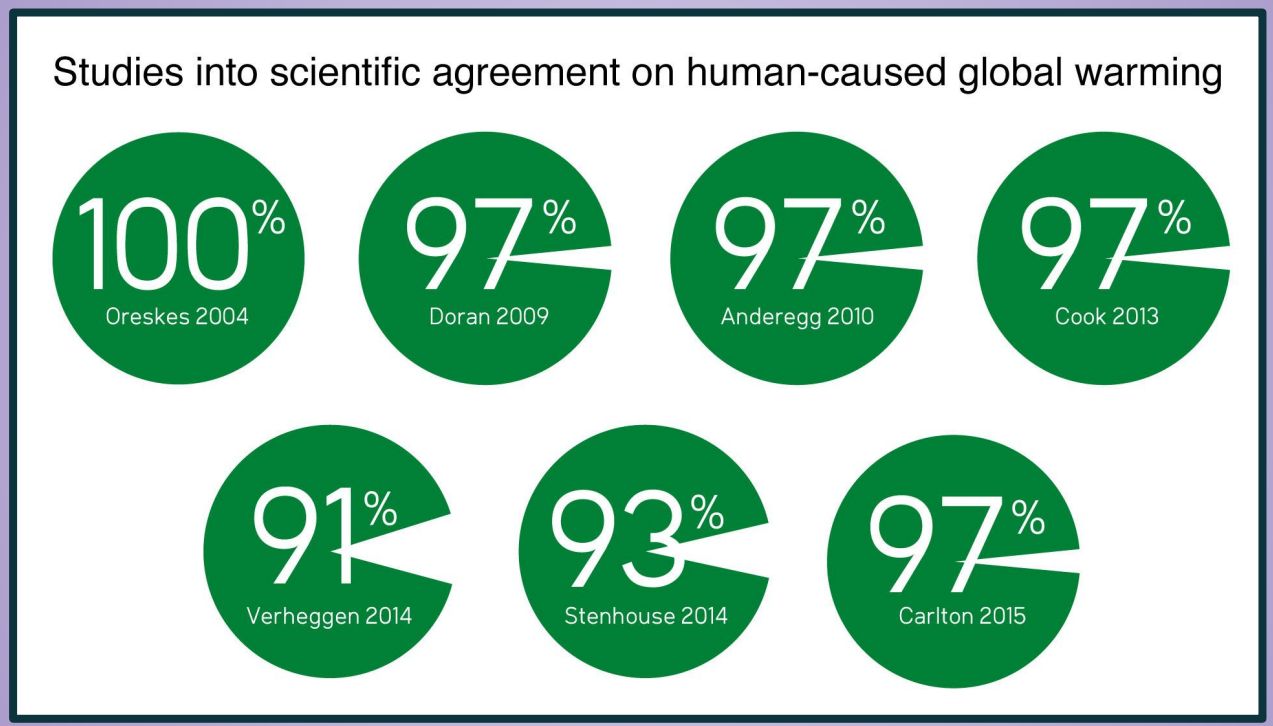
- Navigation of Self Driving Cars
  - Recognizing Obstacles
    - Use of Sensors, Onboard Analytics

Real Life Applications

- Recoloring Black and White Images
  - Recognizing Objects and what they should look like to humans.
    - Returning different colors
    - Recoloring B&W Movies, pics

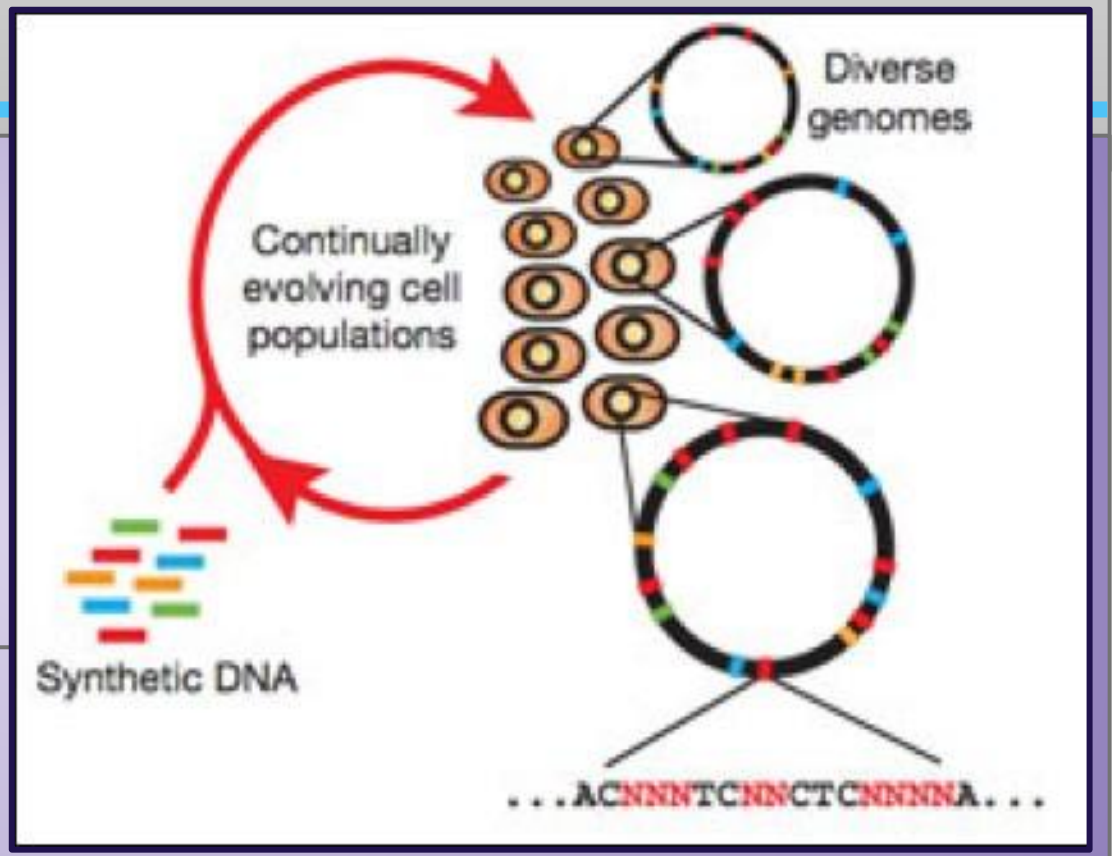
Real Life Applications

- Predicting the outcome of Legal Proceedings
  - British & Americans already use a system to predict court's decision beforehand.
  - Pattern of Proceedings and public surveys.



Real Life Applications

- Precision Medicine
  - Deep Learning can be used to develop medicines.
  - Genetically tailored treatments according to individual's genome
  - Genetic Engineering, Stem cell treatment for cancer etc.



Real Life Applications

- Automated Analysis and Reporting
  - Analysis and Reporting can be done automatically
  - Pollution monitoring stations, weather reports
  - Automated Air Pollution index reporting
  - Detecting Cyclone Activity

Real Life Applications

- Automatic Translation of TEXT, IMAGES
  - Convolution Neural Network to identify images and letters in the images.
  - Adjusting the alignment of letters with the input image

Artificial Neural Network

Neural for Neurons

Networks for a Graph like structure of neurons

Also k/s Artificial Neural Nets and Connectionist Systems

Parts of a Neuron Model

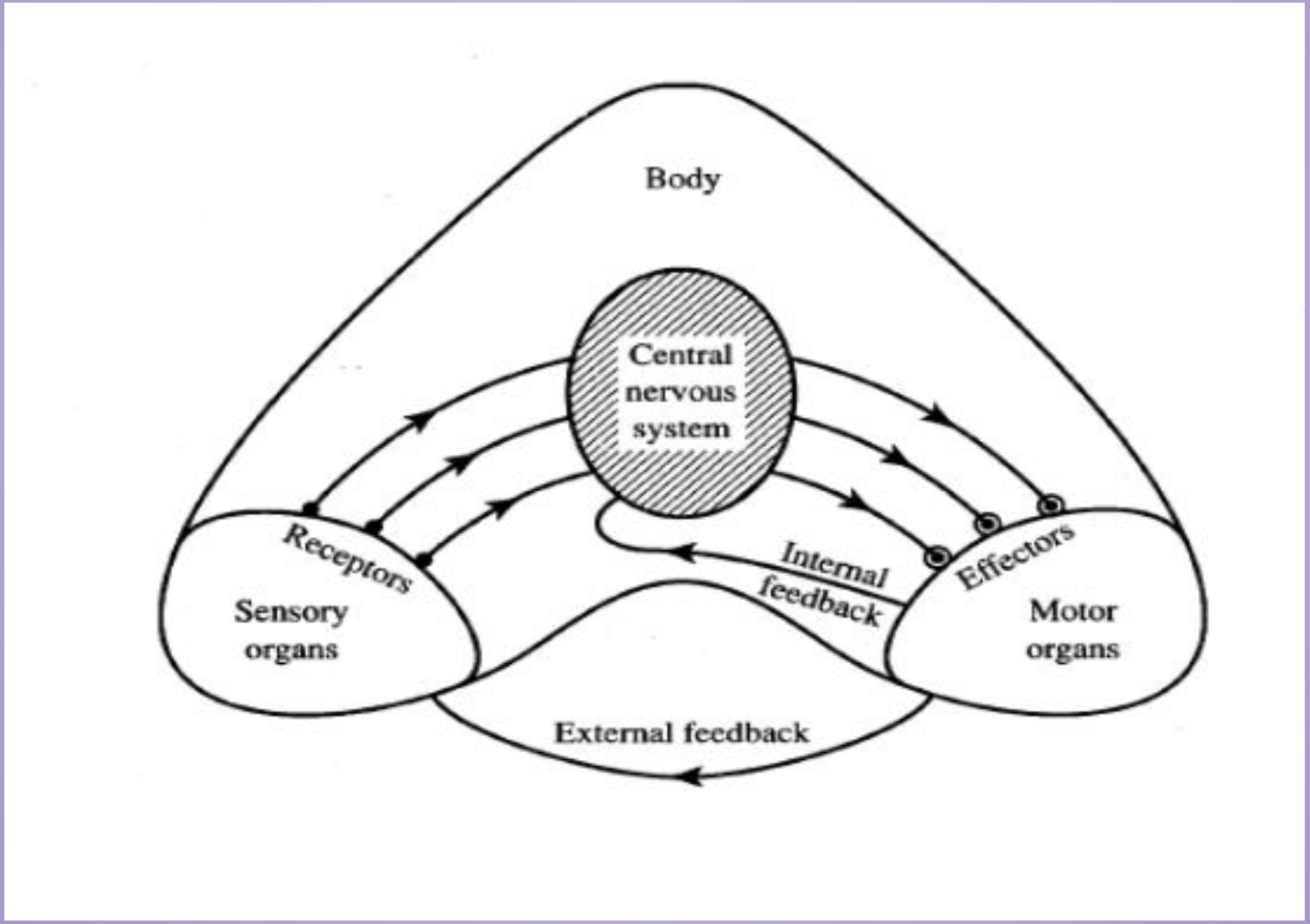
Dendrites : Accepts the inputs

SOMA : Processes the output

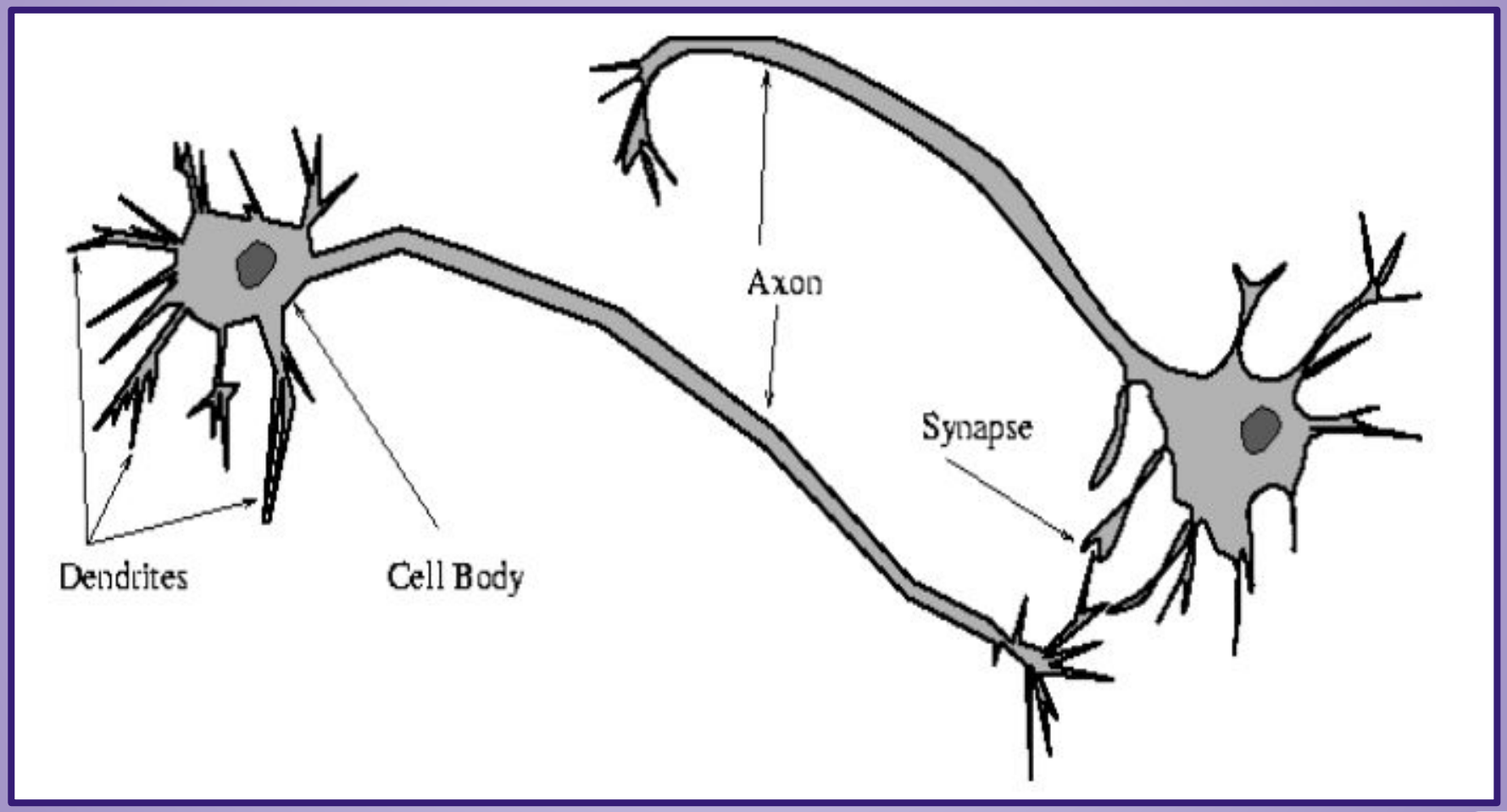
AXON : Turns the processed inputs to Outputs

SYNAPSES : The electrochemical contact between the neurons

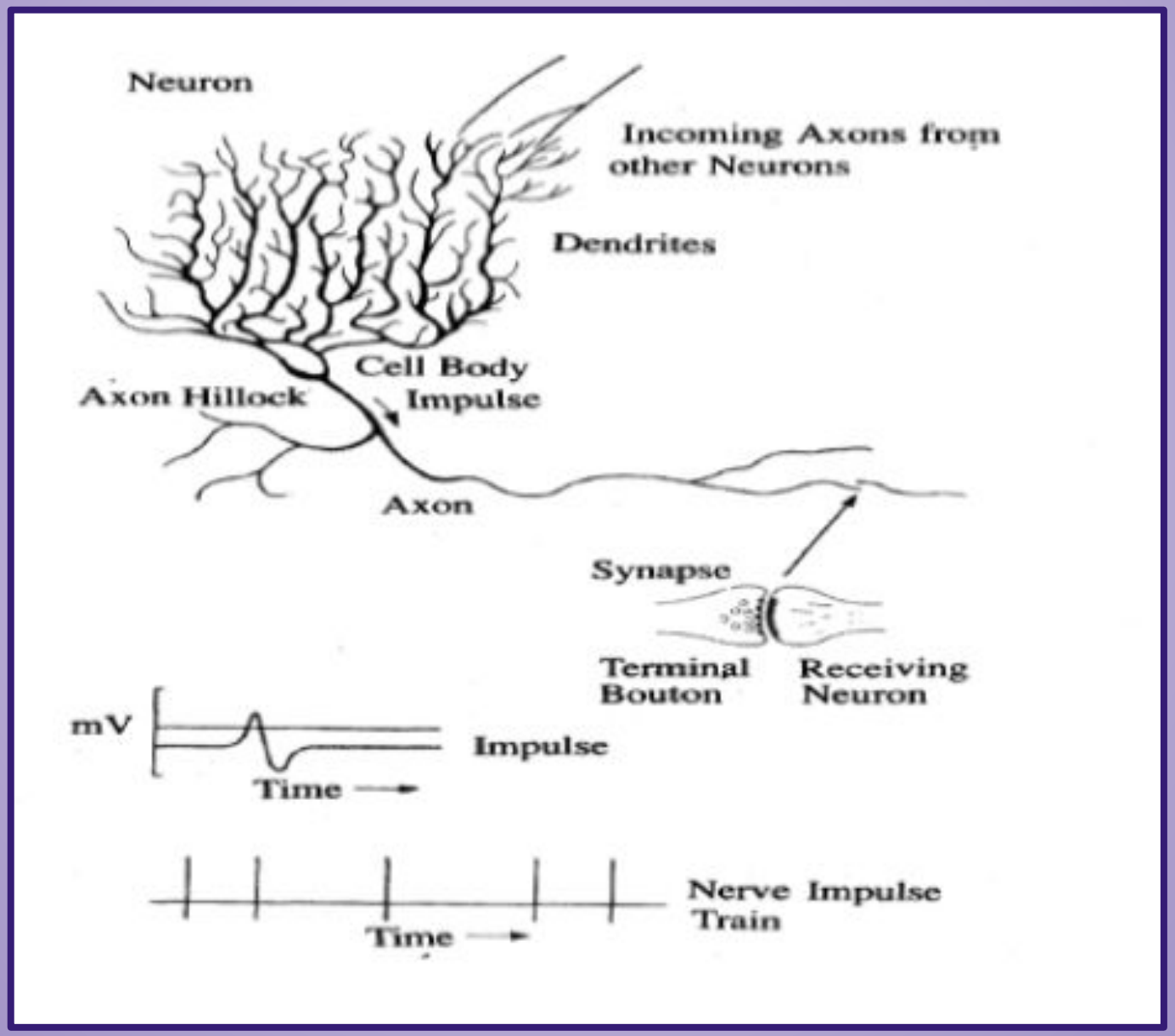
How Brain Works?



4 Parts of a typical nerve cell



Neuron Flow


Neuron
Incoming Axons from other Neurons
Dendrites
Cell Body
Axon Hillock
Impulse
Axon
Synapse
Terminal Bouton
Receiving Neuron
mV
Impulse
Time
Time
Nerve Impulse Train

ARTIFICIAL NEURON MODEL

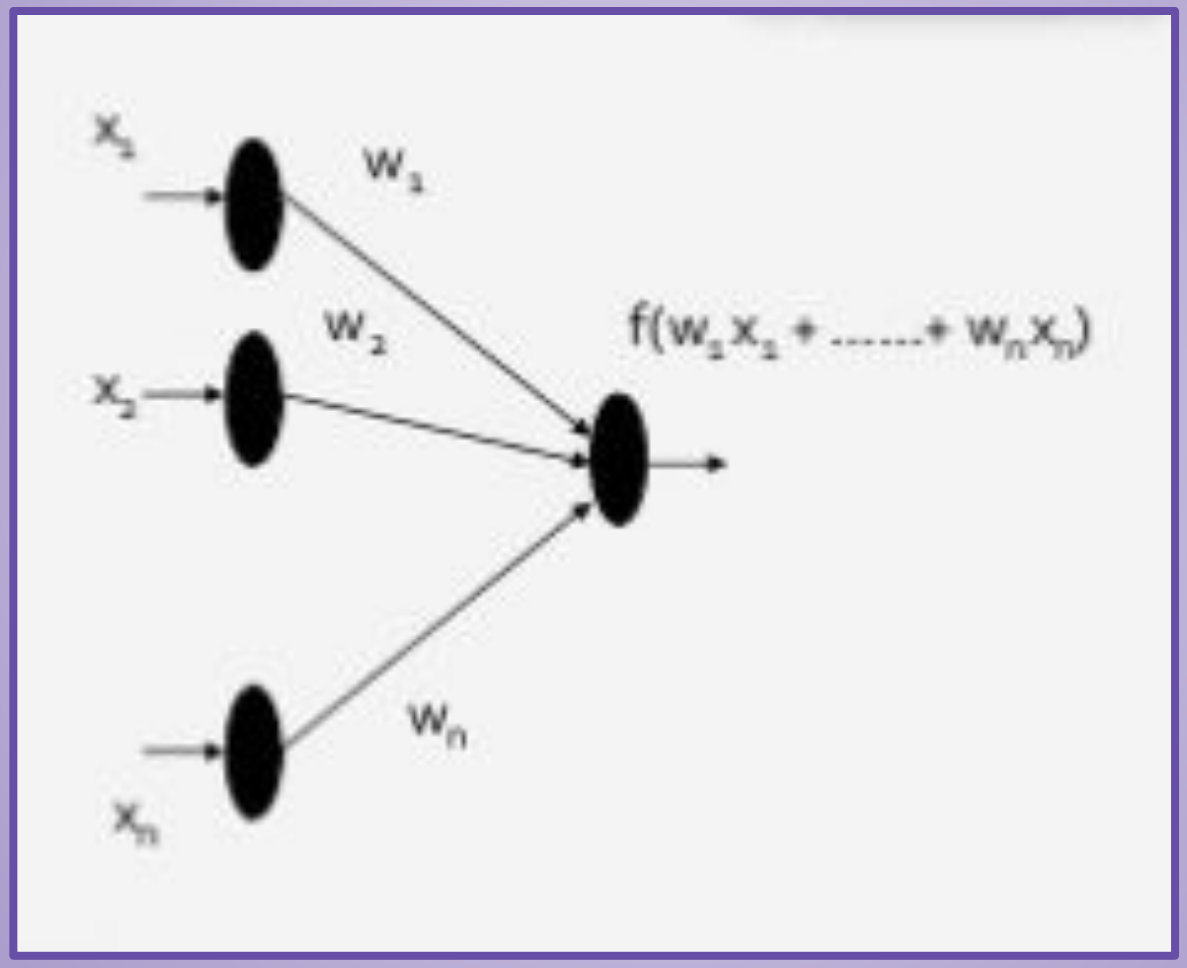
Inputs to the network are represented by symbol, x_n

Each of the inputs are multiplied by a connecting weight, w_n

$$\text{Sum} = w_1x_n + w_2x_n + \ldots\ldots + w_nx_n$$

The products of inputs and weights are summed and fed through a transfer function to generate output.

ARTIFICIAL NEURON MODEL



TERMINOLOGY

Neuron - Node/Cell/Unit/Neurode

Synapse - Connection/Edge/Link

Synaptic Efficiency - Connection Strength
Or, Weight

Firing Frequency - Node Output

ANN

ANN poses a large number of neurons which operate in parallel.

Neurons are connected with others by connection link.

Each link is associated with weights which contain information about the input signal.

Each neuron has an internal state of its own which is a fxn of inputs that neuron receives - Activation Level

ANNs

ANNs are programs designed to solve any problem by trying to mimic the structure and the function of our nervous system.

Neural networks are based on simulated neurons, which are joined together in a variety of ways to form networks.

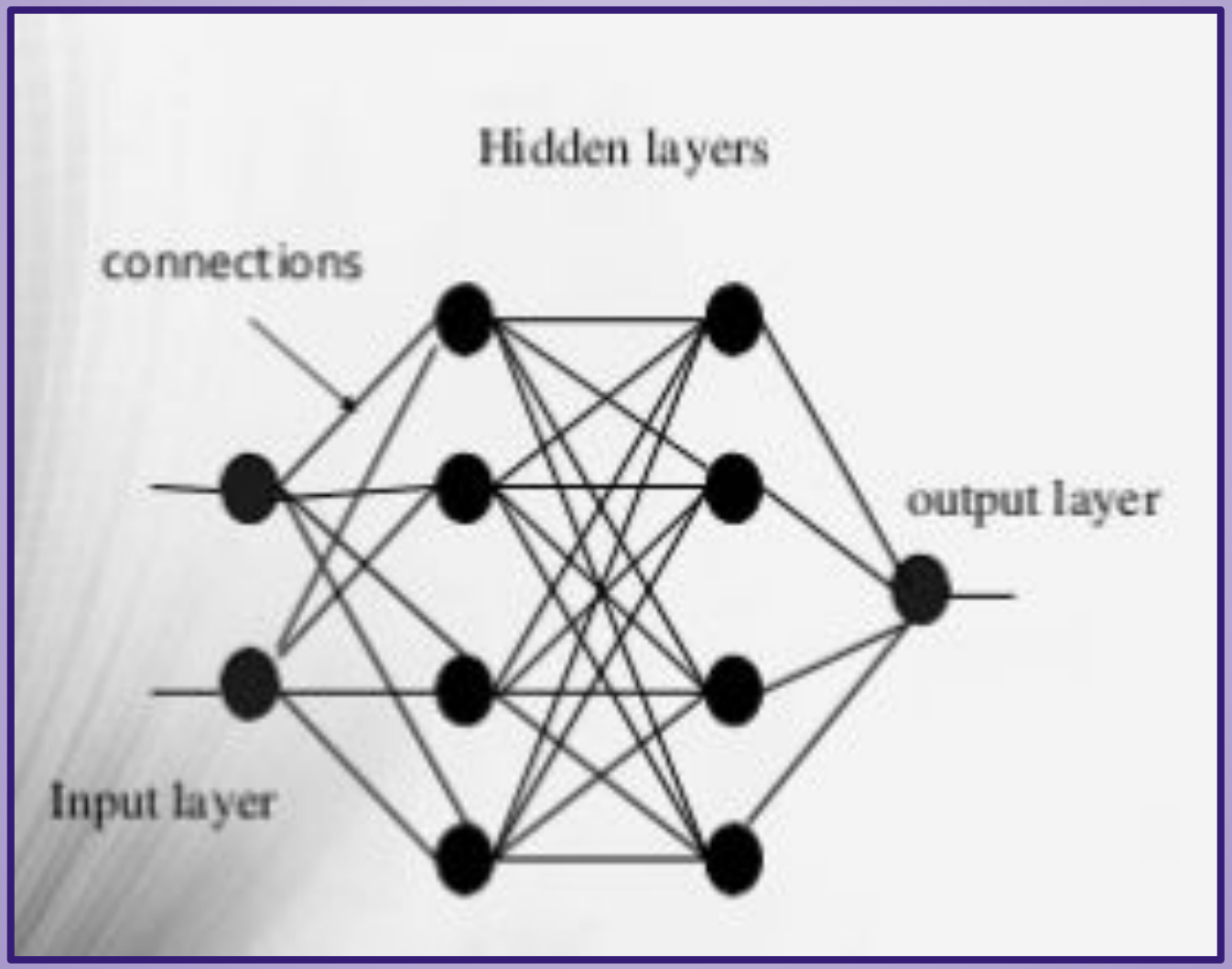
ANNs

Neural networks resemble the human brain in two ways

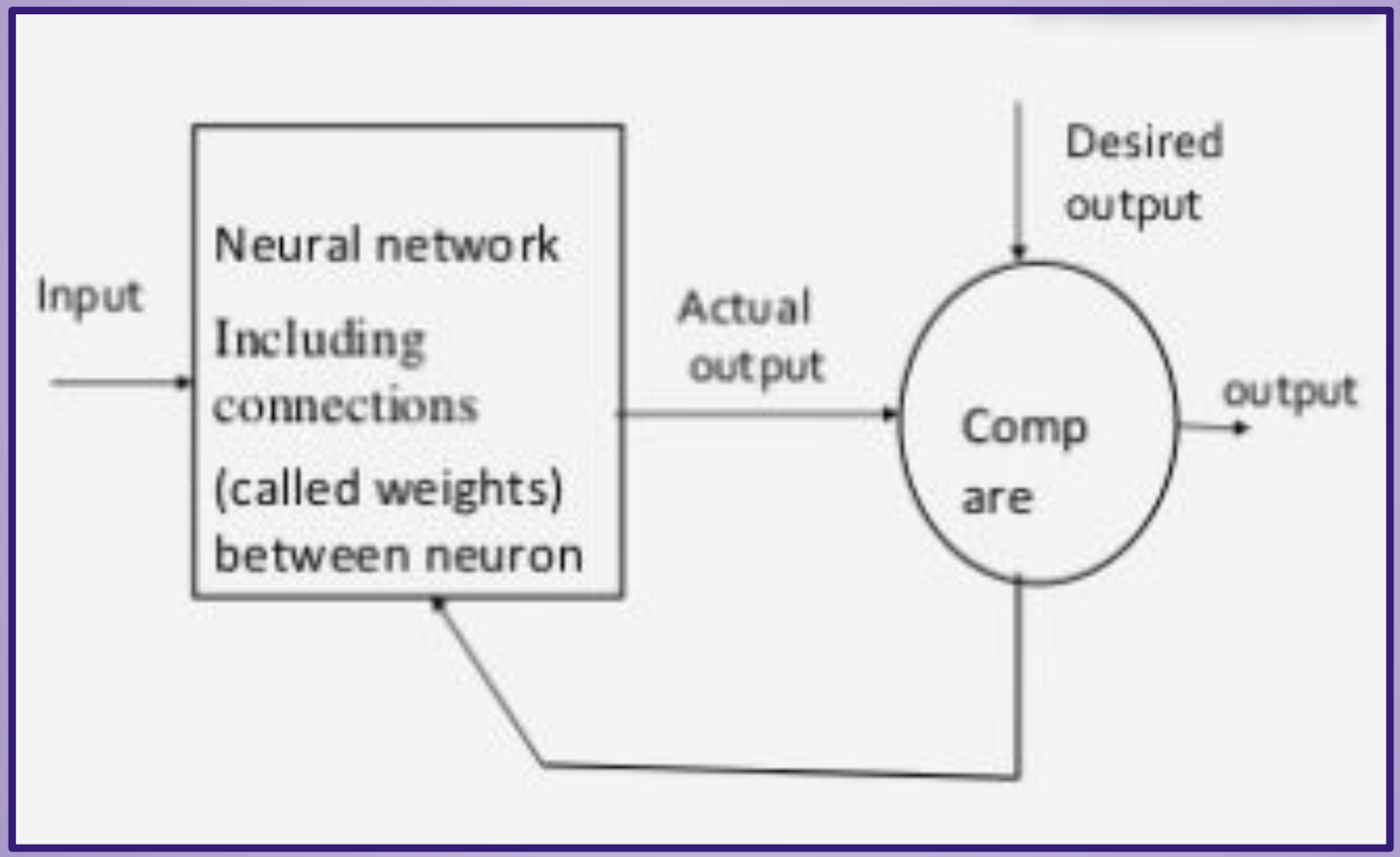
⇒ A neural network acquires knowledge through learning

⇒ A neural network's knowledge is stored within the interconnection strengths known as synaptic weight

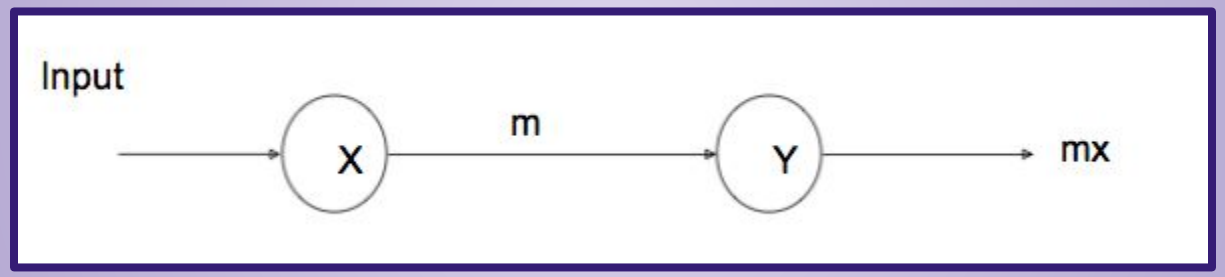
ANN Model



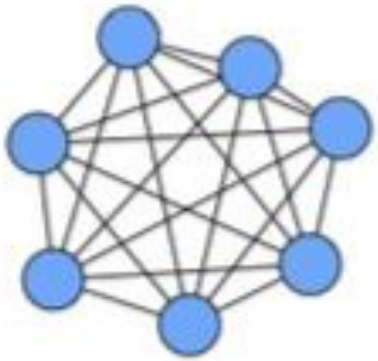
ANN Model


Input
Neural network
Including
connections
(called weights)
between neuron
Actual
output
Desired
output
Comp
are
output

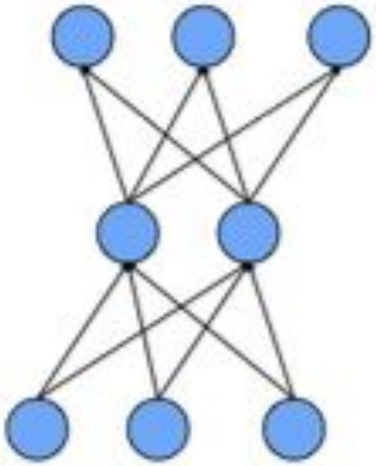
ANN Linear Model



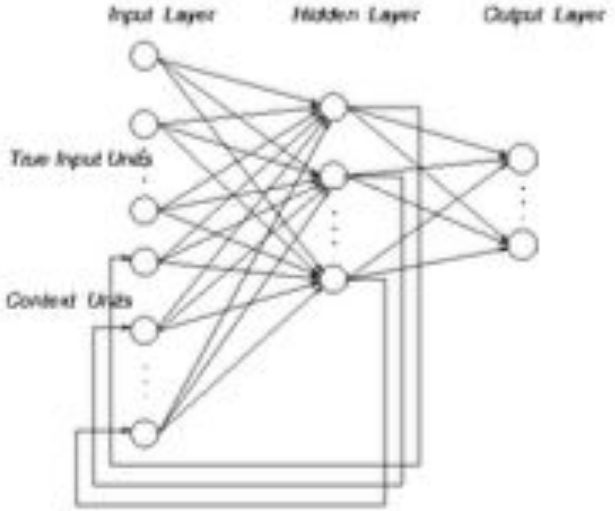
Types of Network Topology



completely connected



feedforward (directed, a-cyclic)



recurrent (feedback connections)

Types of Network Topology

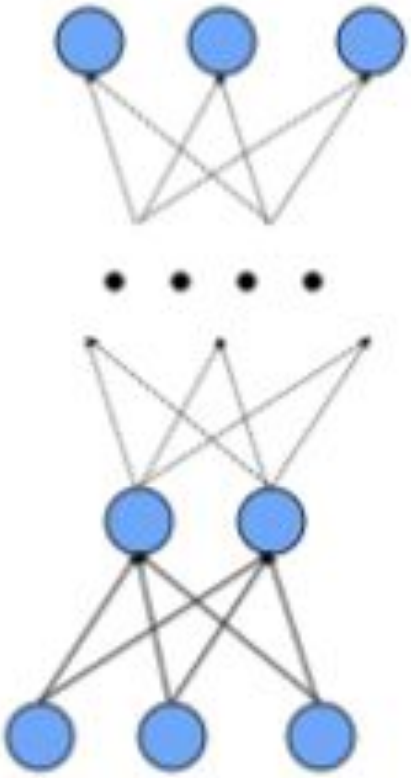
⇒ Feedforward

⇒ No loops, input→ hidden layer→ output

⇒ Recurrent

⇒ Uses feedback (positive or negative)

Multi Layer



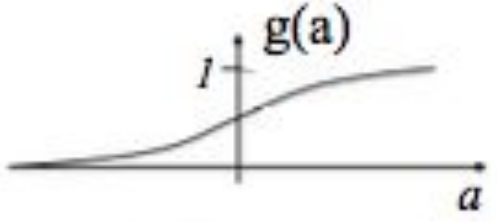
Output neurons

One or more layers of hidden units (hidden layers)

Input nodes

The most common output function (Sigmoid):

$$g(a) = \frac{1}{1 + e^{-\beta a}}$$



(non-linear squashing function)

Layers of Artificial Neural Network

⇒ ANNs are organized in layers.

⇒ Layers are being made of many interconnected 'nodes'

Input Layer

⇒ The purpose of the input layer is to receive the values of input variables for each observation.

⇒ The number of input nodes is equal lo the number of input variables.

⇒ The input layer communicates to the hidden layers.

Hidden Layer

⇒ The hidden layers apply given transformations to the input values inside the network

⇒ It connects with outgoing arcs to either the output nodes or to other hidden layers.

⇒ The value entering a hidden node is multiplied with its weights and then added to produce a single number.

Output Layer

⇒ The hidden layers link to an output layer.

⇒ Output layers receives connections from hidden layers.

⇒ This layer returns an output value which corresponds to the prediction of the response variable.

⇒ The efficiency of a neural network depends upon the proper selection of weights.

Convolution Neural Network

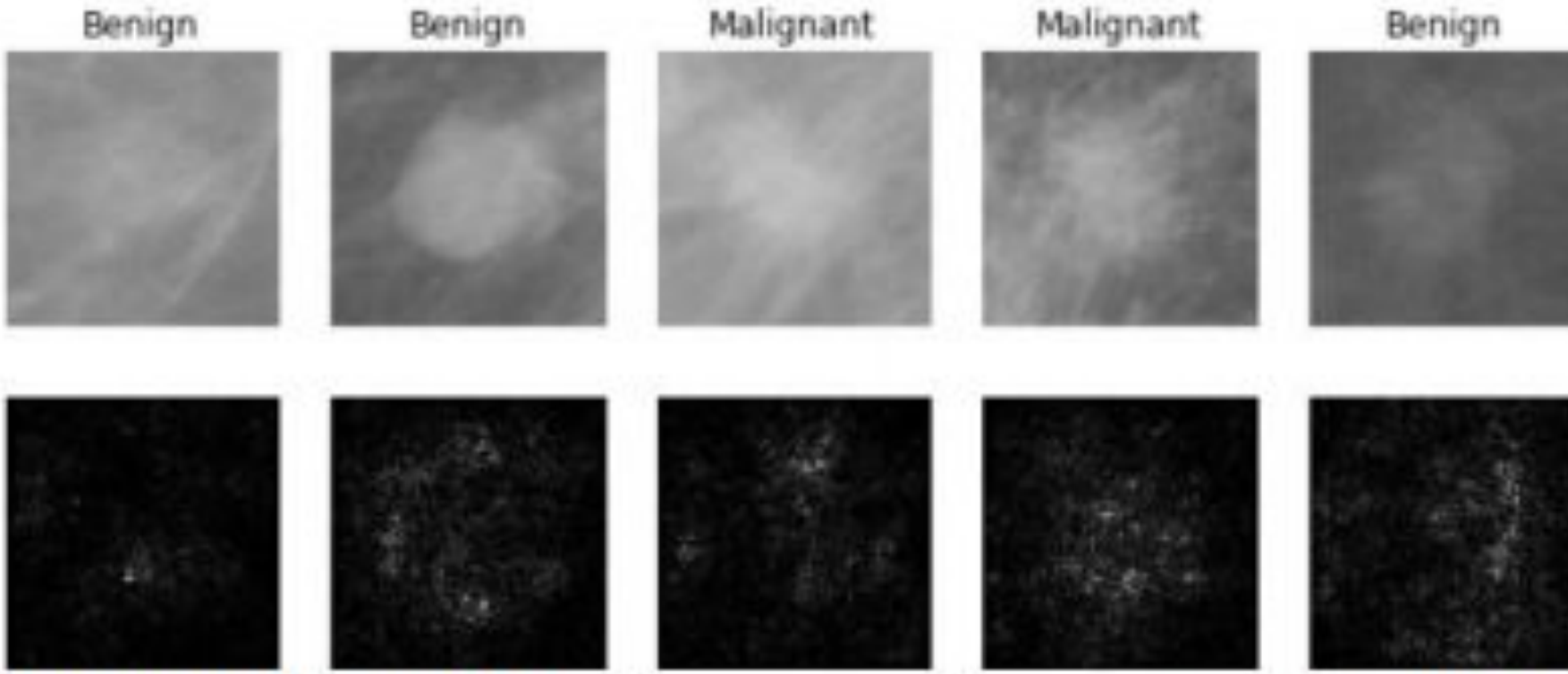
⇒ The hidden layers link to an output layer.

⇒ Output layers receives connections from hidden layers.

⇒ This layer returns an output value which corresponds to the prediction of the response variable.

⇒ The efficiency of a neural network depends upon the proper selection of weights.

Applications



[Levy et al. 2016]

Applications

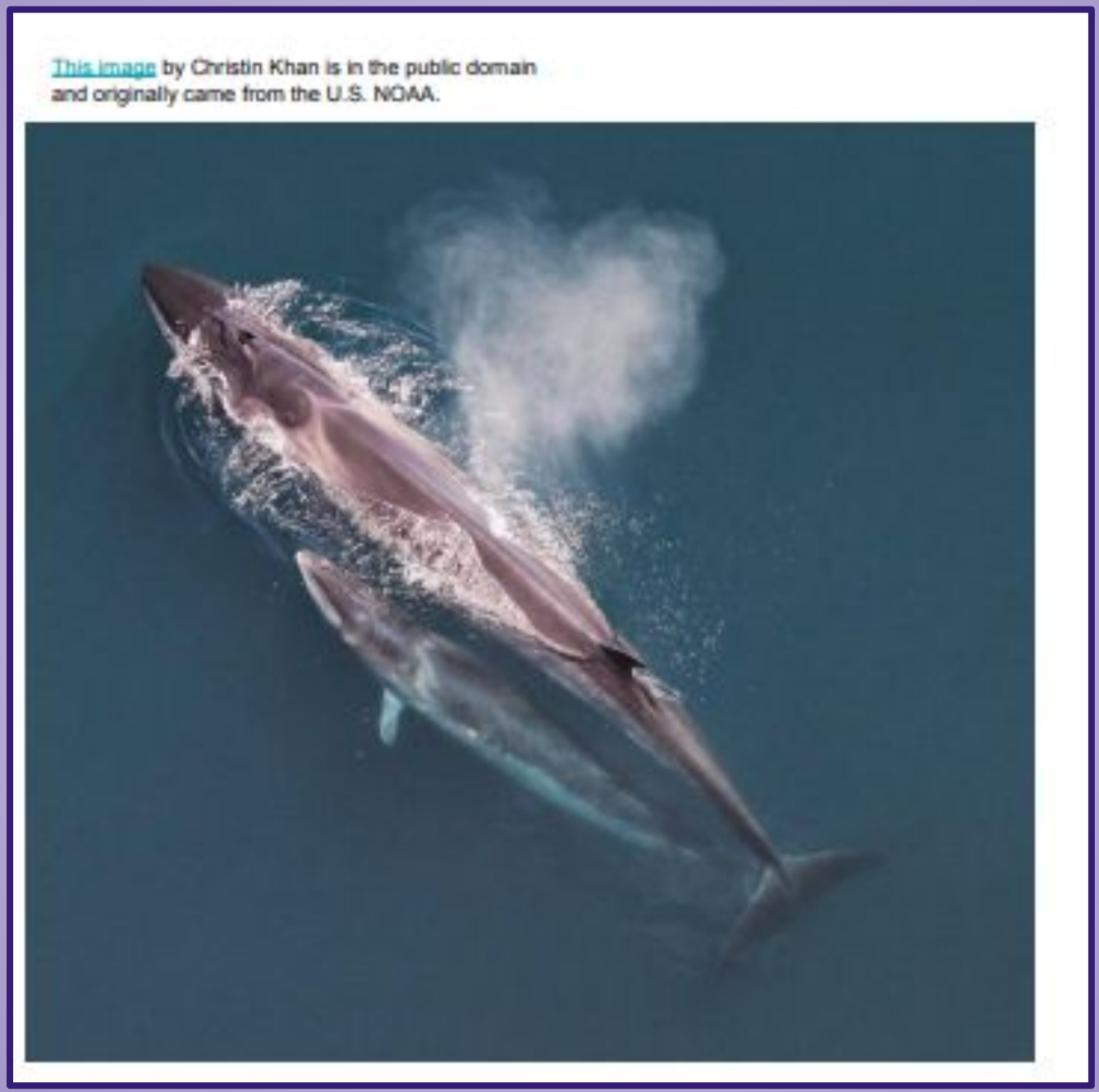
[Sermanet et al. 2011]
[Ciresan et al.]

Applications

[Dieleman et al. 2014]

From left to right: public domain by NASA, usage permitted by ESA/Hubble, public domain by NASA, and public domain.

Applications



Applications

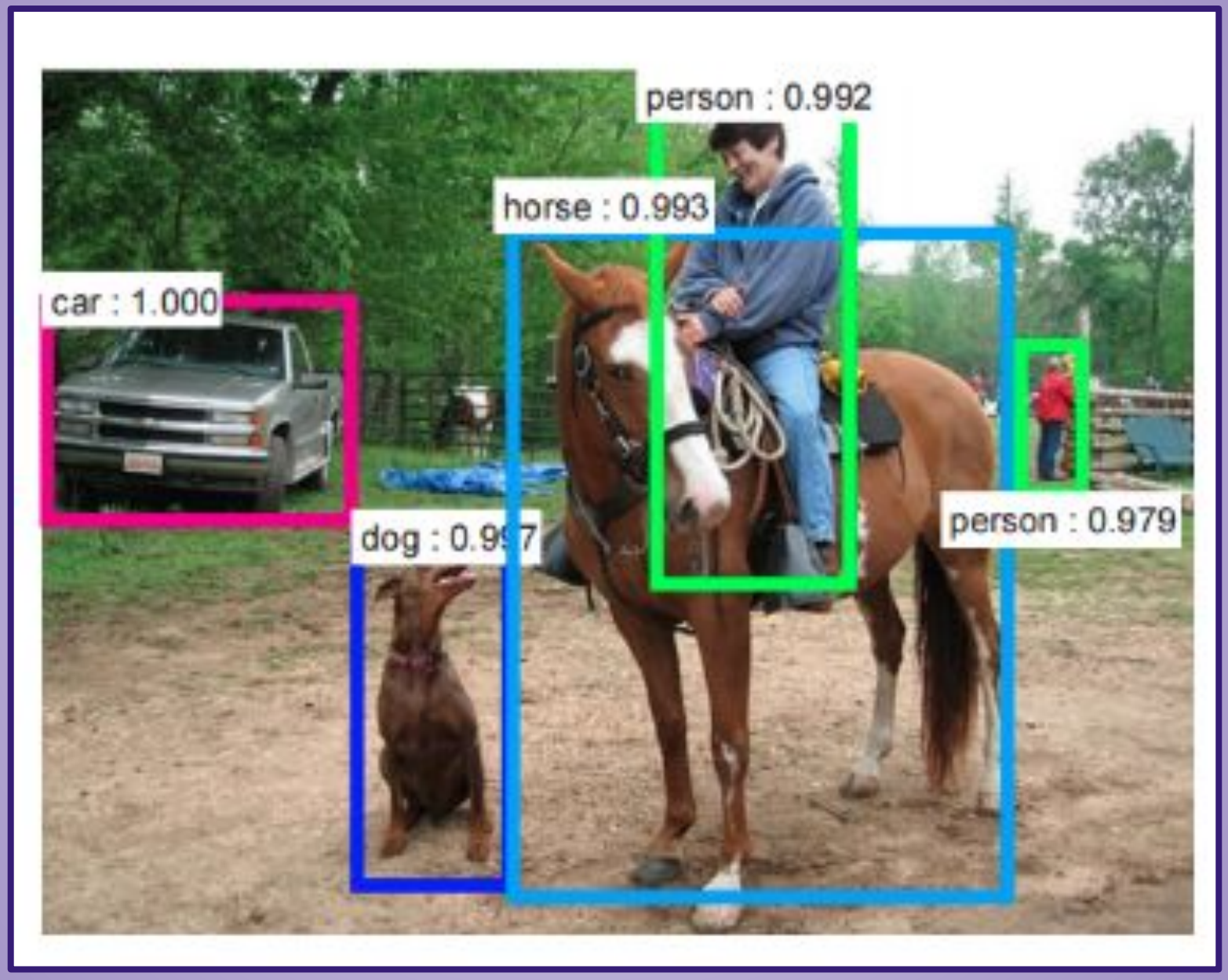
A white teddy bear sitting in the grass

A man in a baseball uniform throwing a ball

A man riding a wave on top of a surfboard

A cat sitting on a suitcase on the floor

Applications



Applications

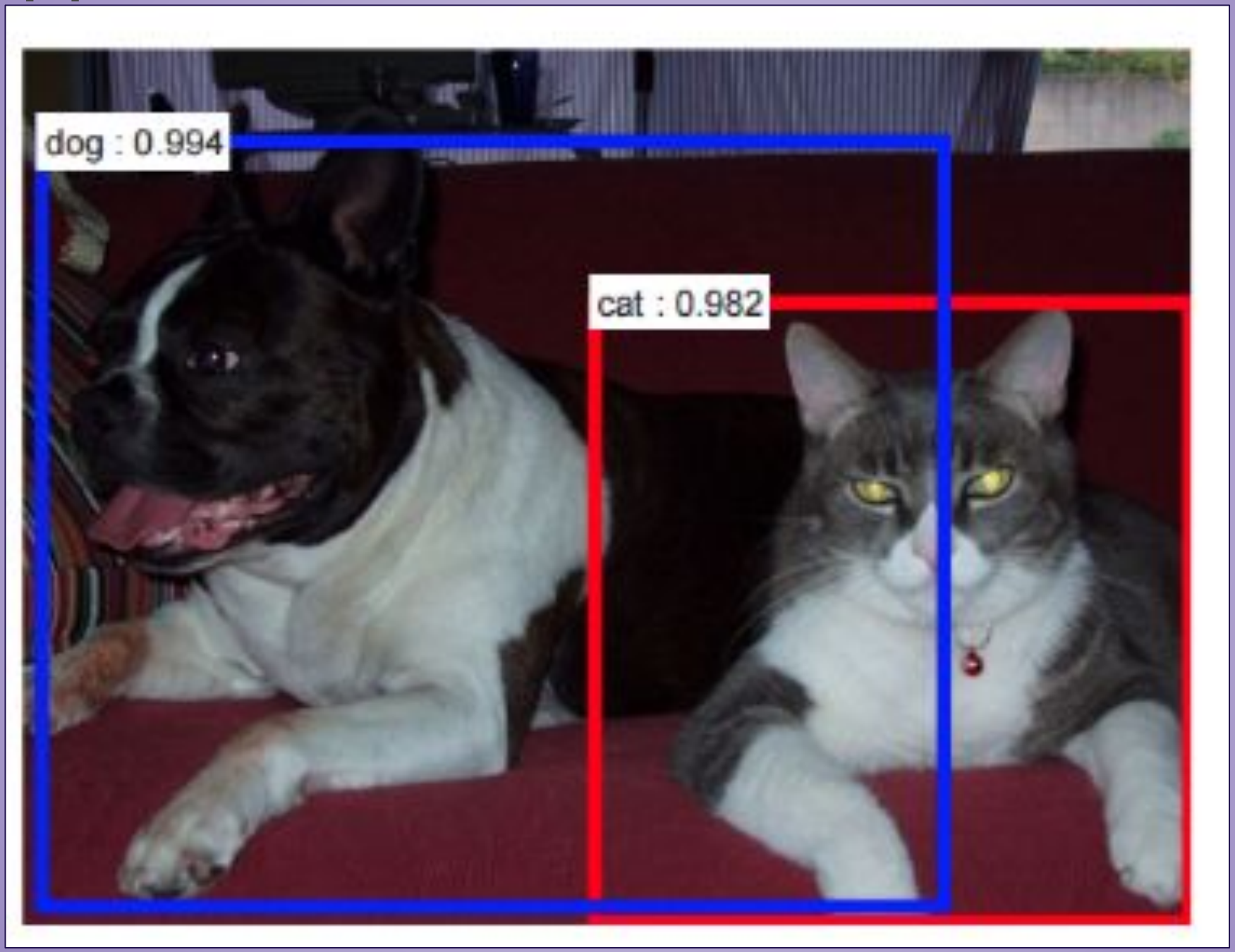
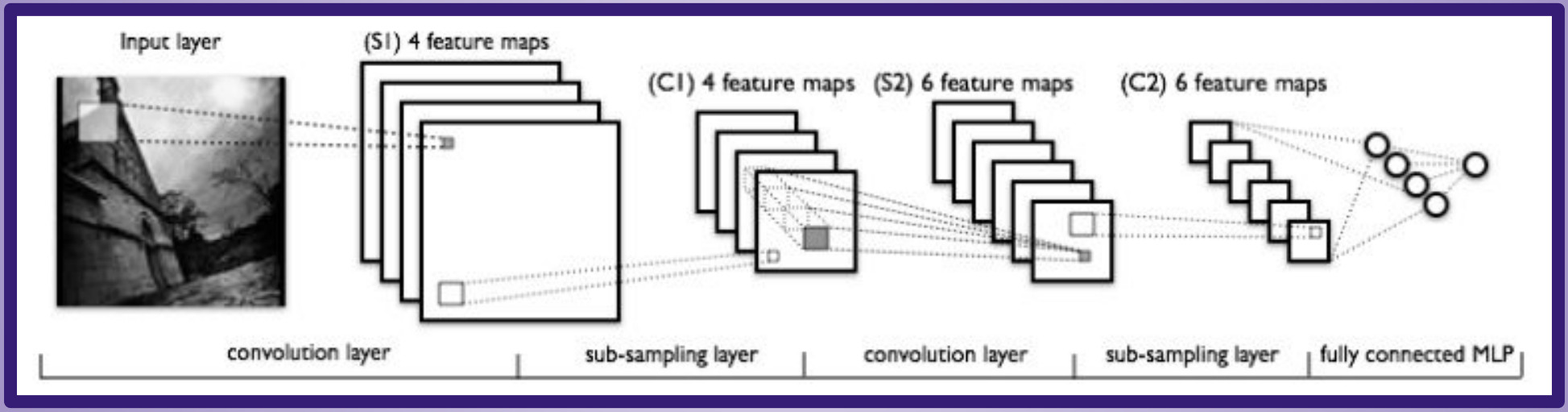


Image net classification

(from Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton, ImageNet Classification with Deep Convolutional Neural Networks, NIPS, 2012.)

Sequence of transformations in CNN



Sequence of transformations in CNN

- The actual input image that is scanned for features.
- The light rectangle is the filter that passes over it.
- Activation maps stacked atop one another, one for each filter you employ.
- The larger rectangle is one patch to be downsampled.
- The activation maps condensed through downsampling.
- A new set of activation maps created by passing filters over the first downsampled stacks
- The second downsampling, which condenses the second set of activation maps.
- A fully connected layer that classifies output with one label per node.

CNNs

- Convolutional neural networks ingest and process images as tensors, and tensors are matrices of numbers with additional dimensions.
- A scalar is just a number, such as 7;
- a vector is a list of numbers (e.g., [7,8,9]);
- a matrix is a rectangular grid of numbers occupying several rows and columns like a spreadsheet.
- f a scalar is a zero-dimensional point, then a vector is a one-dimensional line, a matrix is a two-dimensional plane,
- a stack of matrices is a three-dimensional cube,
- when each element of those matrices has a stack of feature maps attached to it, you enter the fourth dimension. Known as 4D Tensors

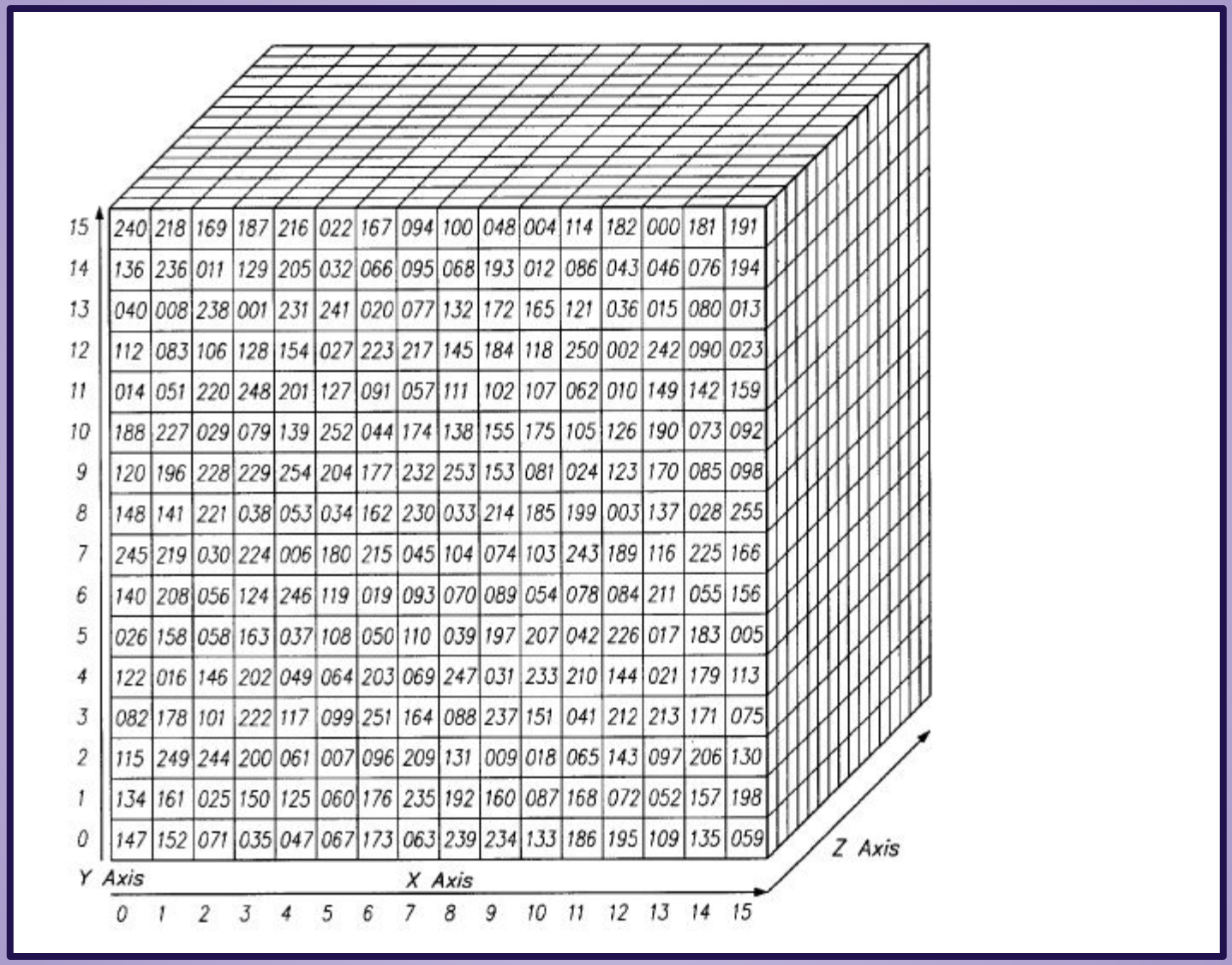
CNNs

$$\begin{bmatrix} 1, & 2 \\ 5, & 8 \end{bmatrix}$$

$$\begin{bmatrix} \begin{pmatrix} 2 \\ 3 \end{pmatrix} & \begin{pmatrix} 3 \\ 5 \end{pmatrix} & \begin{pmatrix} 4 \\ 7 \end{pmatrix} \\ \begin{pmatrix} 3 \\ 4 \end{pmatrix} & \begin{pmatrix} 4 \\ 6 \end{pmatrix} & \begin{pmatrix} 5 \\ 8 \end{pmatrix} \end{bmatrix}$$

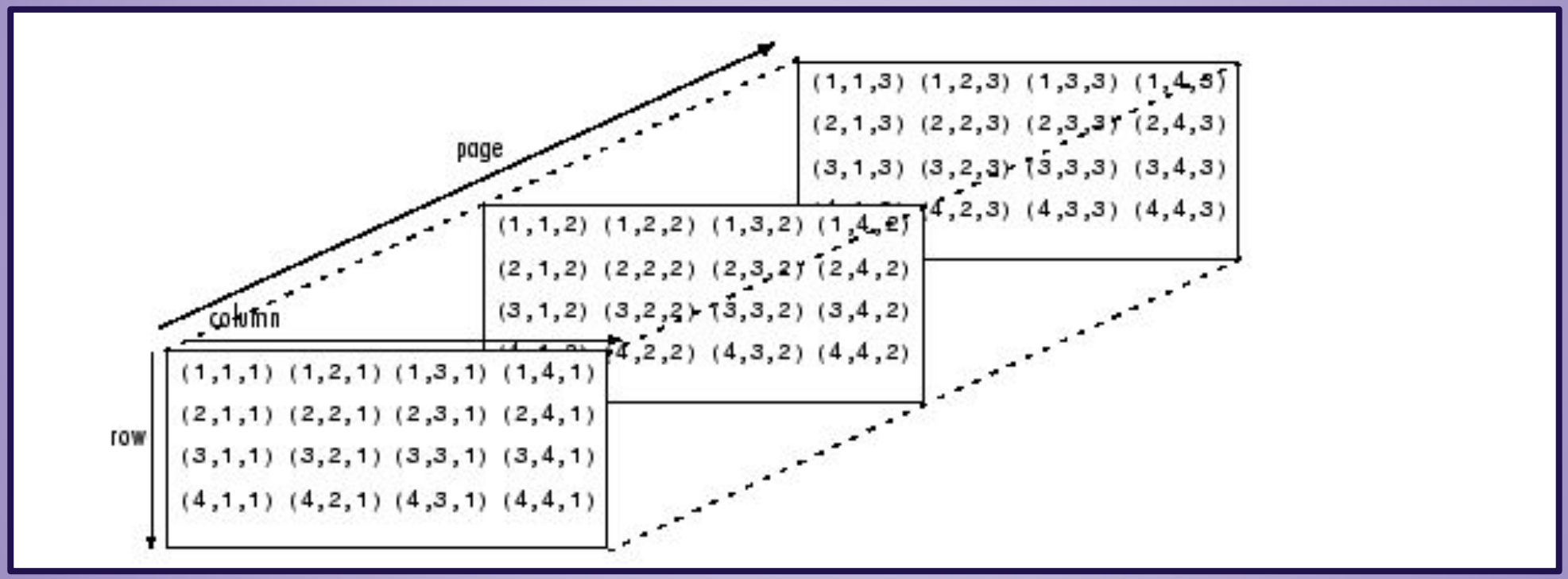
```
[[[2,3],[3,5],[4,7]],[[3,4],[4,6],[5,8]]].
```

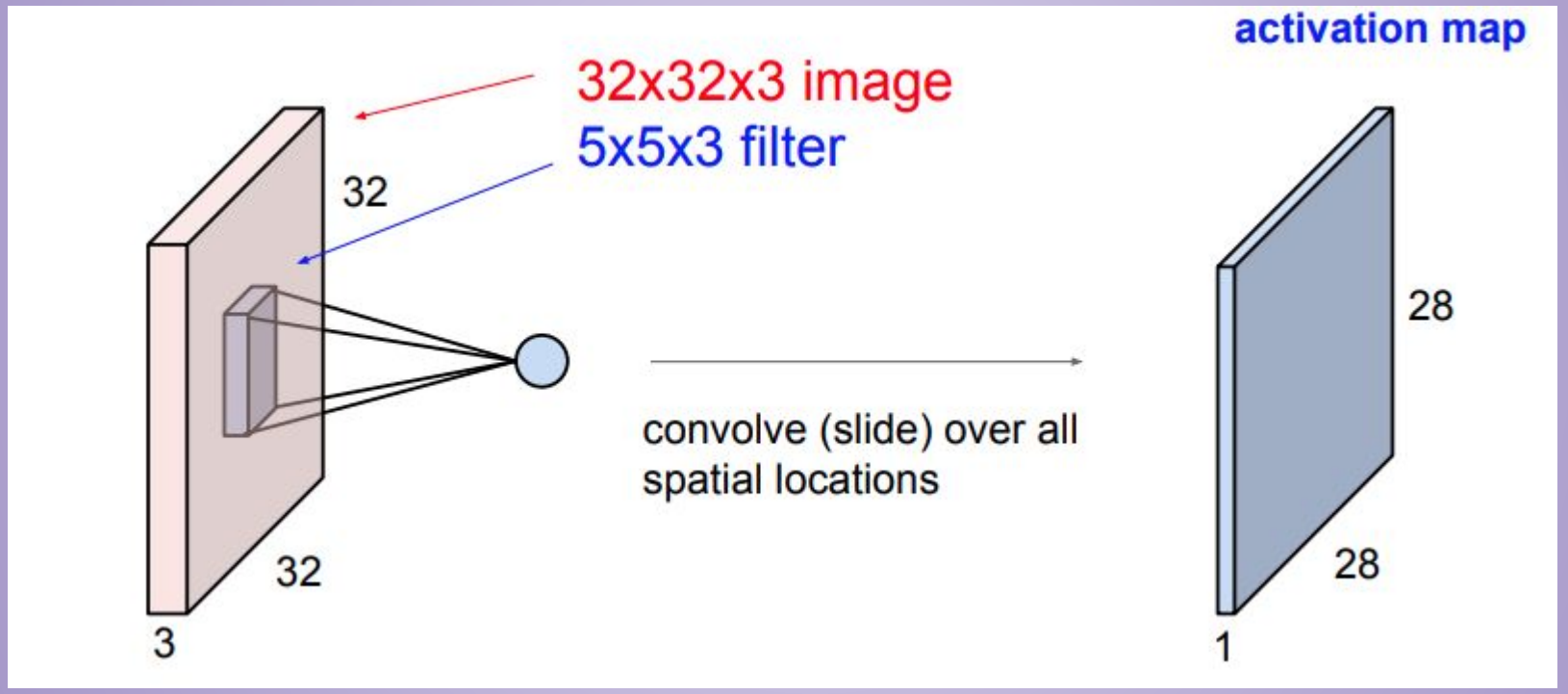
CNNs


15 240 218 169 187 216 022 167 094 100 048 004 114 182 000 181 191
14 136 236 011 129 205 032 066 095 068 193 012 086 043 046 076 194
13 040 008 238 001 231 241 020 077 132 172 165 121 036 015 080 013
12 112 083 106 128 154 027 223 217 145 184 118 250 002 242 090 023
11 014 051 220 248 201 127 091 057 111 102 107 062 010 149 142 159
10 188 227 029 079 139 252 044 174 138 155 175 105 126 190 073 092
9 120 196 228 229 254 204 177 232 253 153 081 024 123 170 085 098
8 148 141 221 038 053 034 162 230 033 214 185 199 003 137 028 255
7 245 219 030 224 006 180 215 045 104 074 103 243 189 116 225 166
6 140 208 056 124 246 119 019 093 070 089 054 078 084 211 055 156
5 026 158 058 163 037 108 050 110 039 197 207 042 226 017 183 005
4 122 016 146 202 049 064 203 069 247 031 233 210 144 021 179 113
3 082 178 101 222 117 099 251 164 088 237 151 041 212 213 171 075
2 115 249 244 200 061 007 096 209 131 009 018 065 143 097 206 130
1 134 161 025 150 125 060 176 235 192 160 087 168 072 052 157 198
0 147 152 071 035 047 067 173 063 239 234 133 186 195 109 135 059
Y Axis
X Axis
Z Axis
0 1 2 3 4 5 6 7 8 9 10 11 12 13 14 15

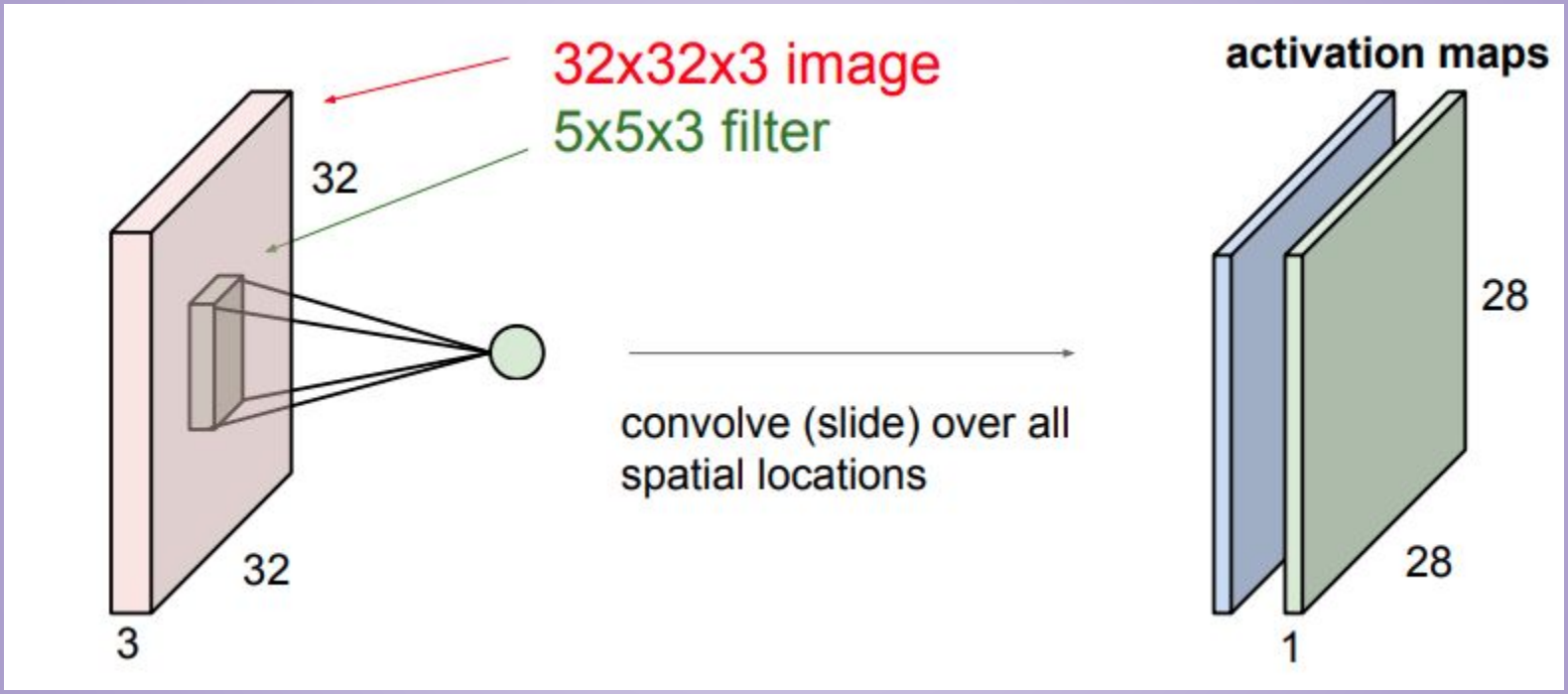
CNNs

A 4-D tensor would simply replace each of the scalars with an array nested one level deeper.

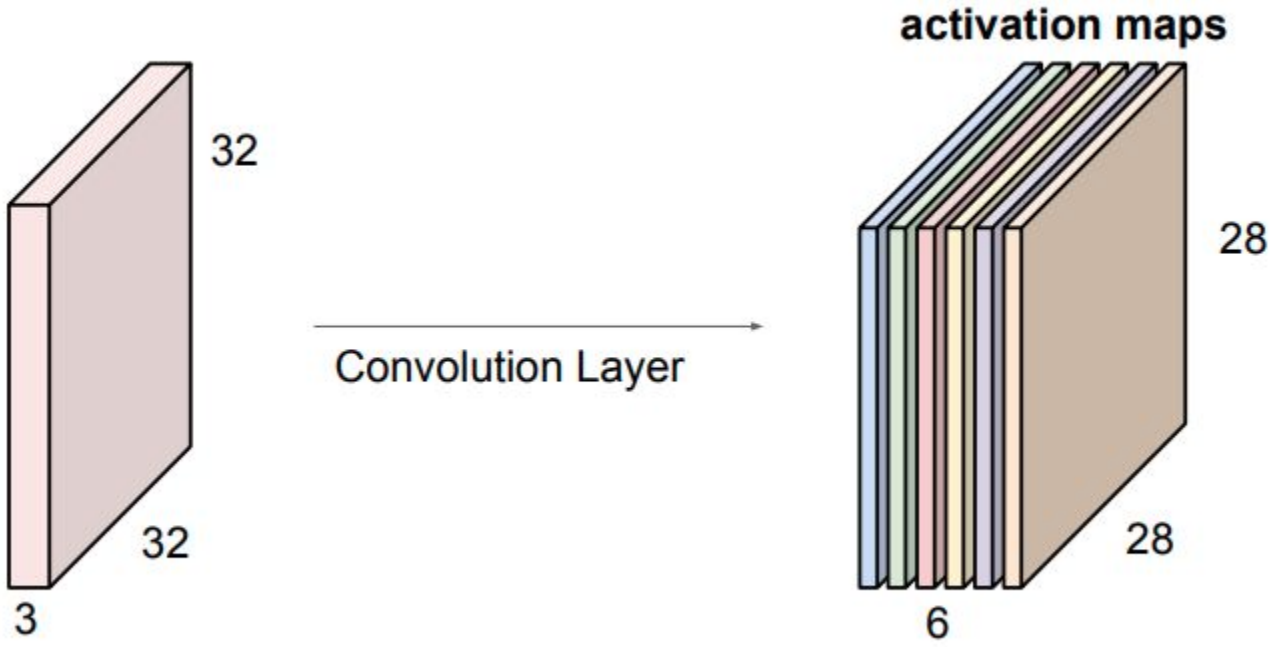



activation map
32x32x3 image
5x5x3 filter
32
32
3
convolve (slide) over all
spatial locations
28
28
1

Filter and Feature Maps

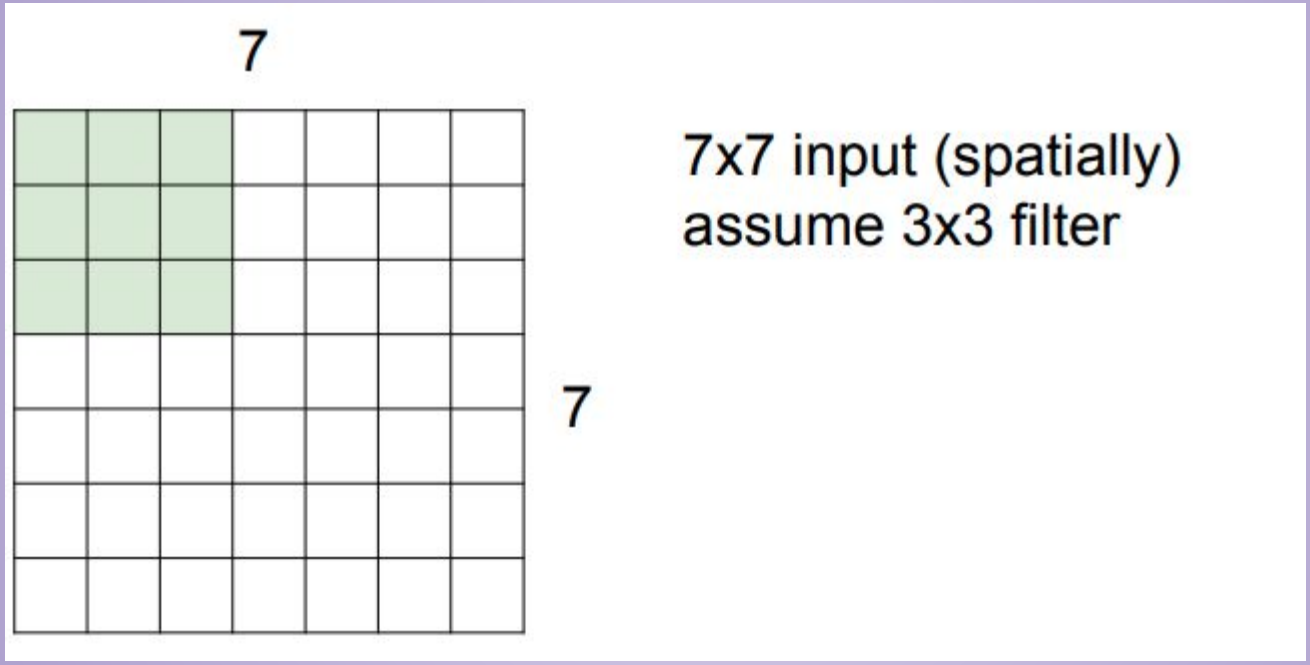


For example, if we had 6 5x5 filters, we'll get 6 separate activation maps:

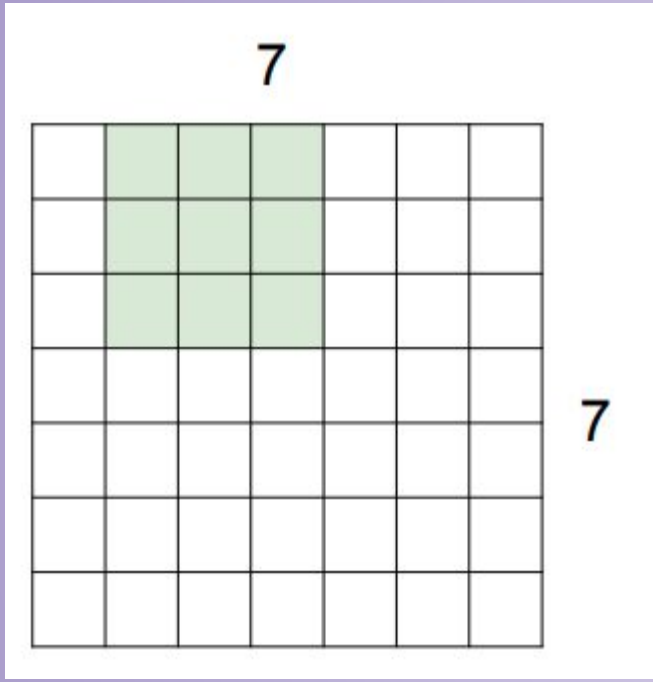


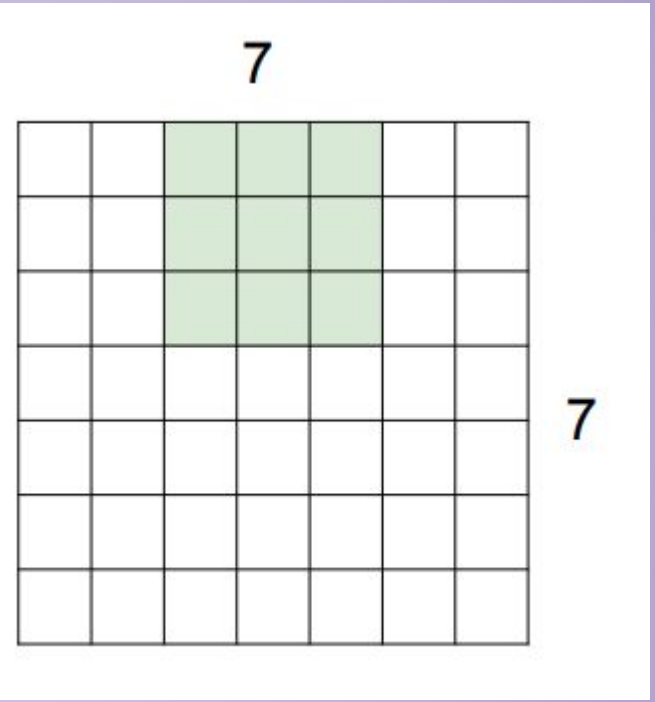
We stack these up to get a "new image" of size 28x28x6!

Filters



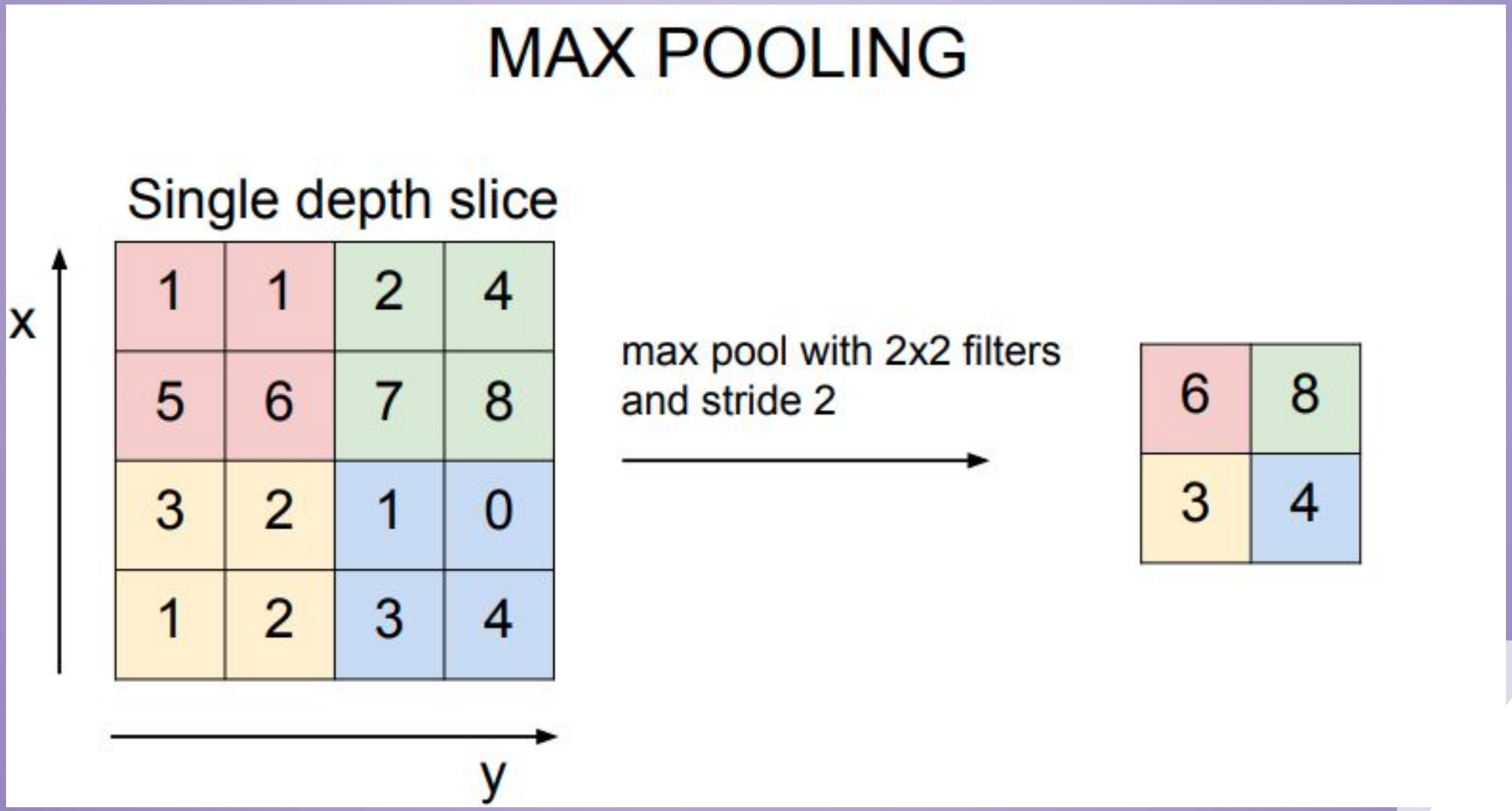
Filters





Max Pooling/Downsampling/ Subsampling

Max pooling takes the largest value from one patch of an image at a time and places into a new matrix.
Much information about lesser values is lost.




thanks