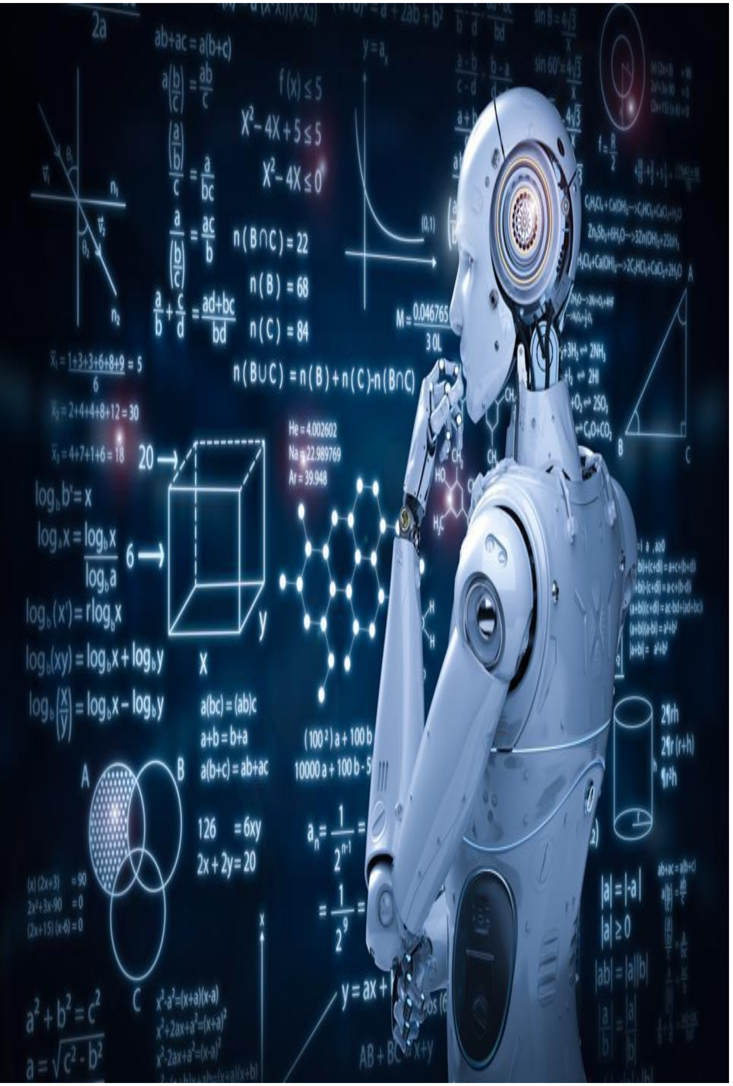


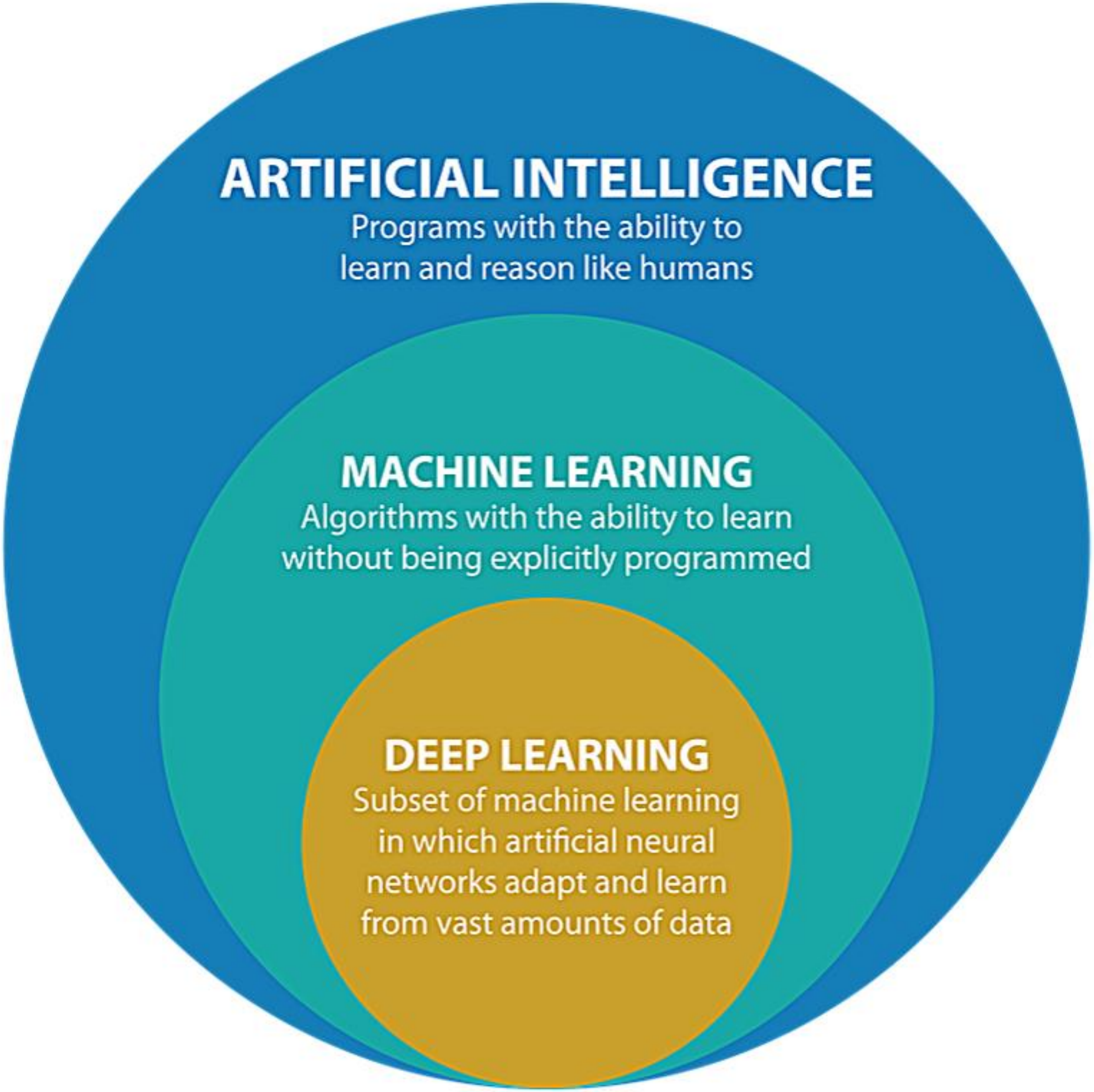
Practical Machine Learning

Day 15: SEP23 DBDA

Kiran Waghmare

Agenda

- Artificial Neural Network


ARTIFICIAL INTELLIGENCE
Programs with the ability to learn and reason like humans
MACHINE LEARNING
Algorithms with the ability to learn without being explicitly programmed
DEEP LEARNING
Subset of machine learning in which artificial neural networks adapt and learn from vast amounts of data

Why is Deep Learning Important?

- Causing a revolution in Artificial Intelligence
- Electrifying the computing industry
- Transforming corporate America
- Why? – because over the last five years we have experienced quantum leaps in the quality of many everyday technologies

Fortune, 2016

Why is Deep Learning Important?

- Major advances in Image Recognition
  - Search and automatically organize collections of photos
    - Apple, Amazon, Microsoft, Facebook
- Speech Technologies work much better
  - Speech recognition: Apple's Siri, Amazon's Alexa, Microsoft's Cortana, Chinese Baidu speech interfaces
  - Translation of spoken sentences: Google Translate
- Deep learning also improving medical applications, robotics, autonomous drones, self-driving cars, etc.

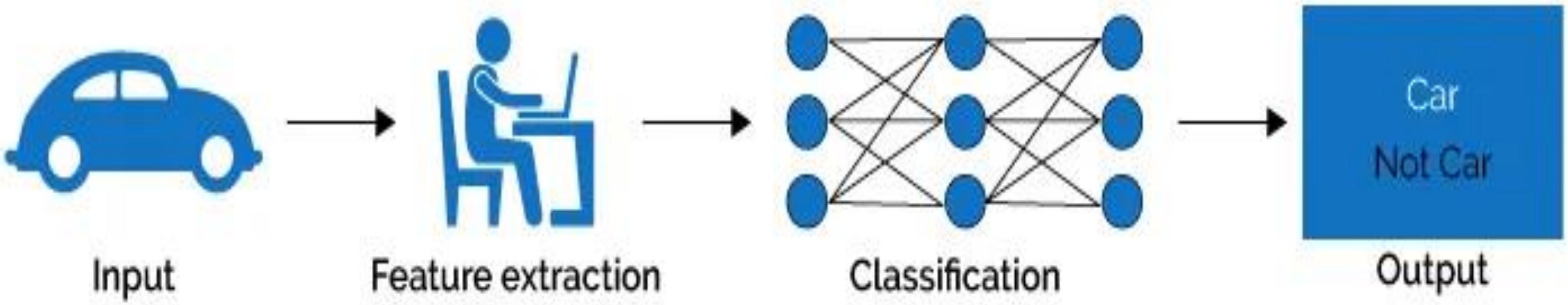
Frank Rosenblatt

Major Types of Deep Learning Systems

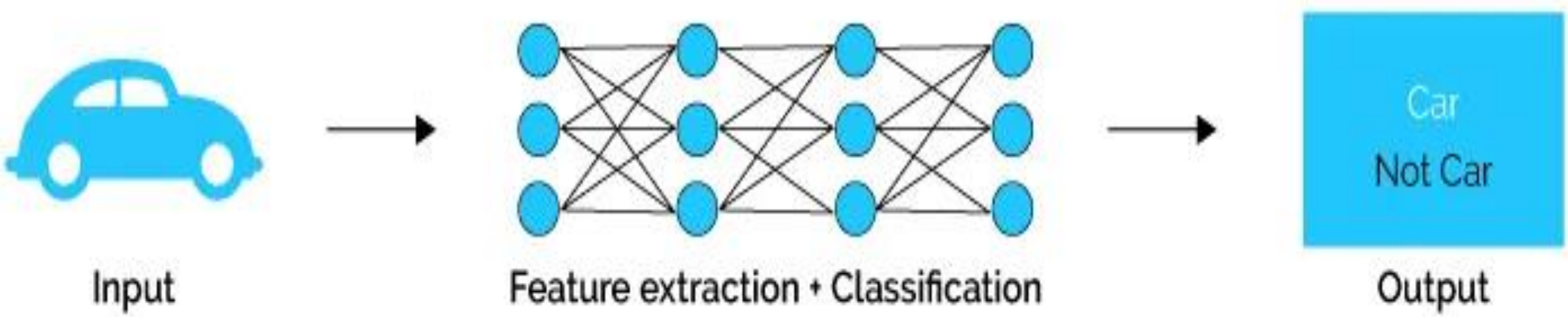
- Convolutional Neural Networks for matrix data
  - a type of feed-forward artificial neural network in which the connectivity pattern between its neurons is inspired by the organization of the animal visual cortex
  - often referred to as multilayer perceptrons
- Recurrent Neural Networks for sequential data
  - a class of artificial neural network where connections between units form a directed cycle (feedback)

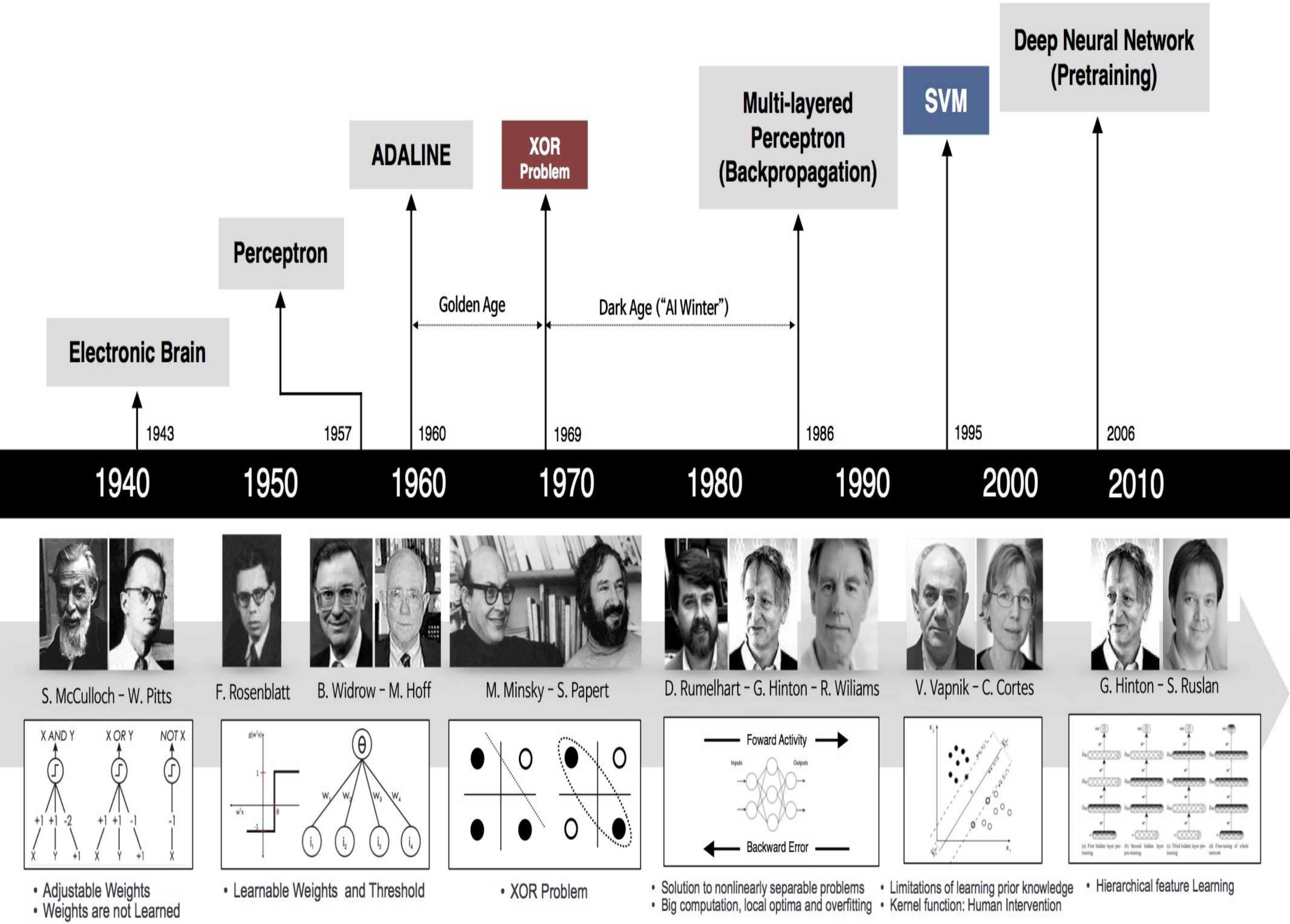
Frank Rosenblatt

Machine Learning

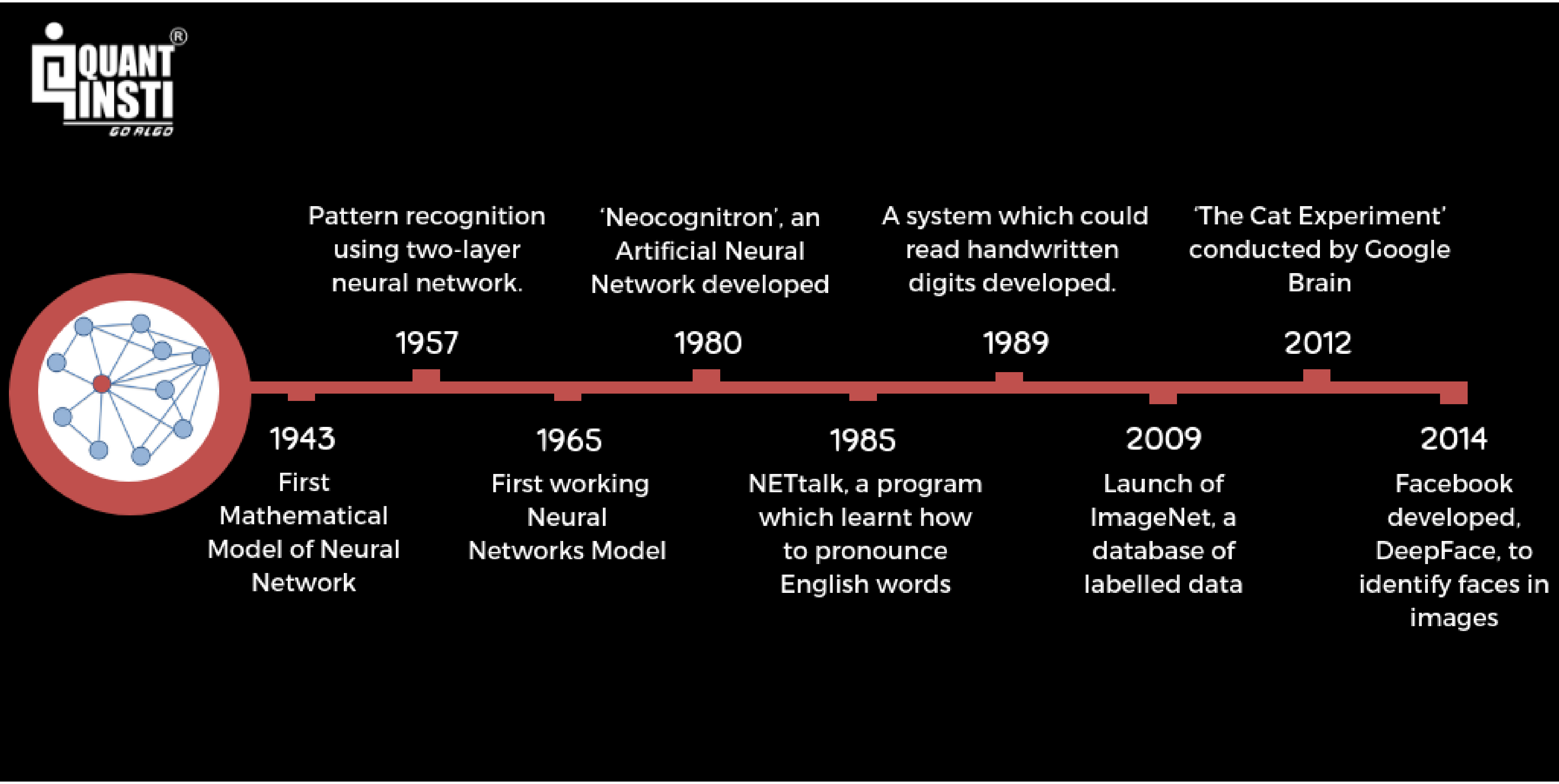


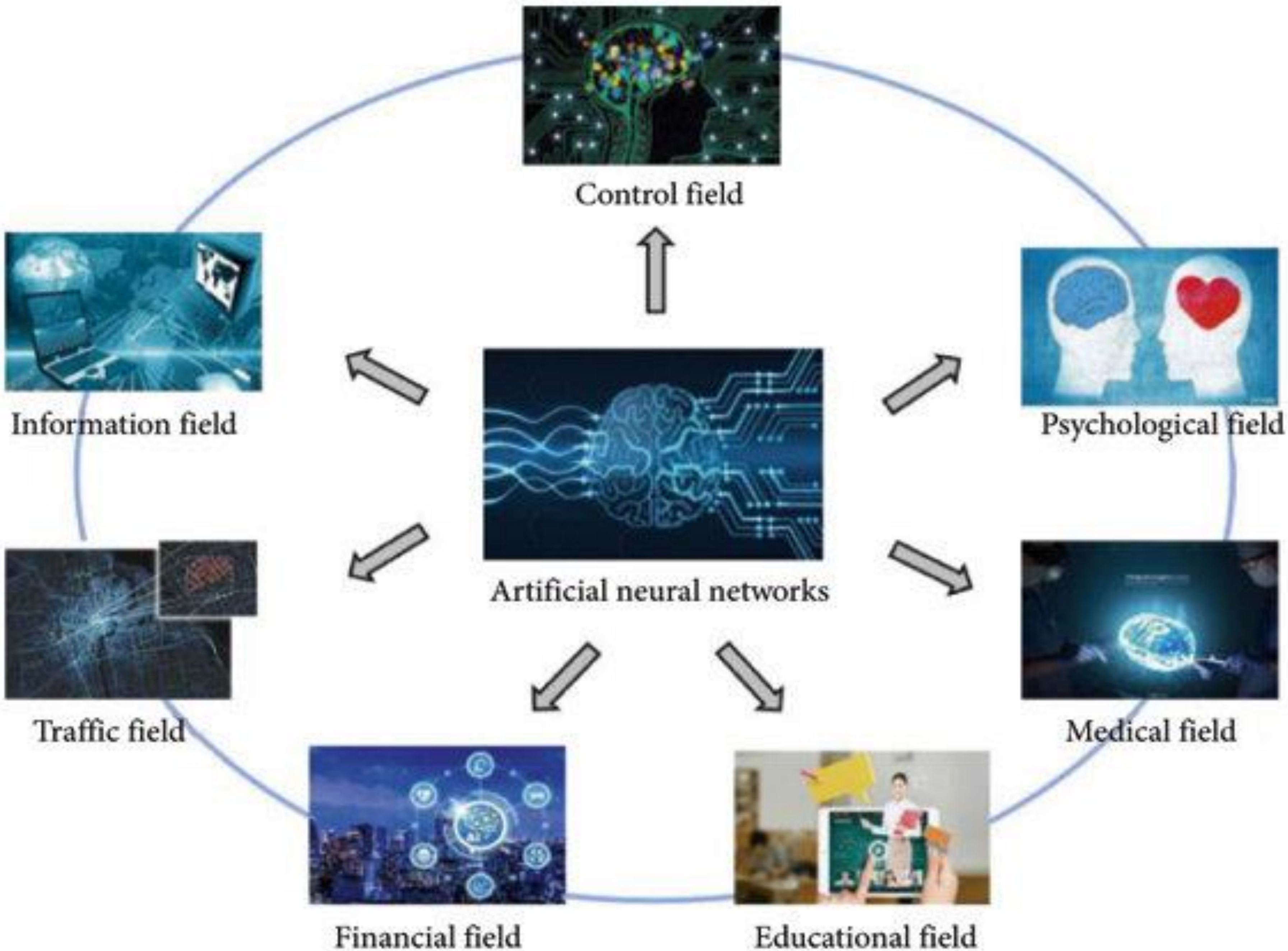
Deep Learning





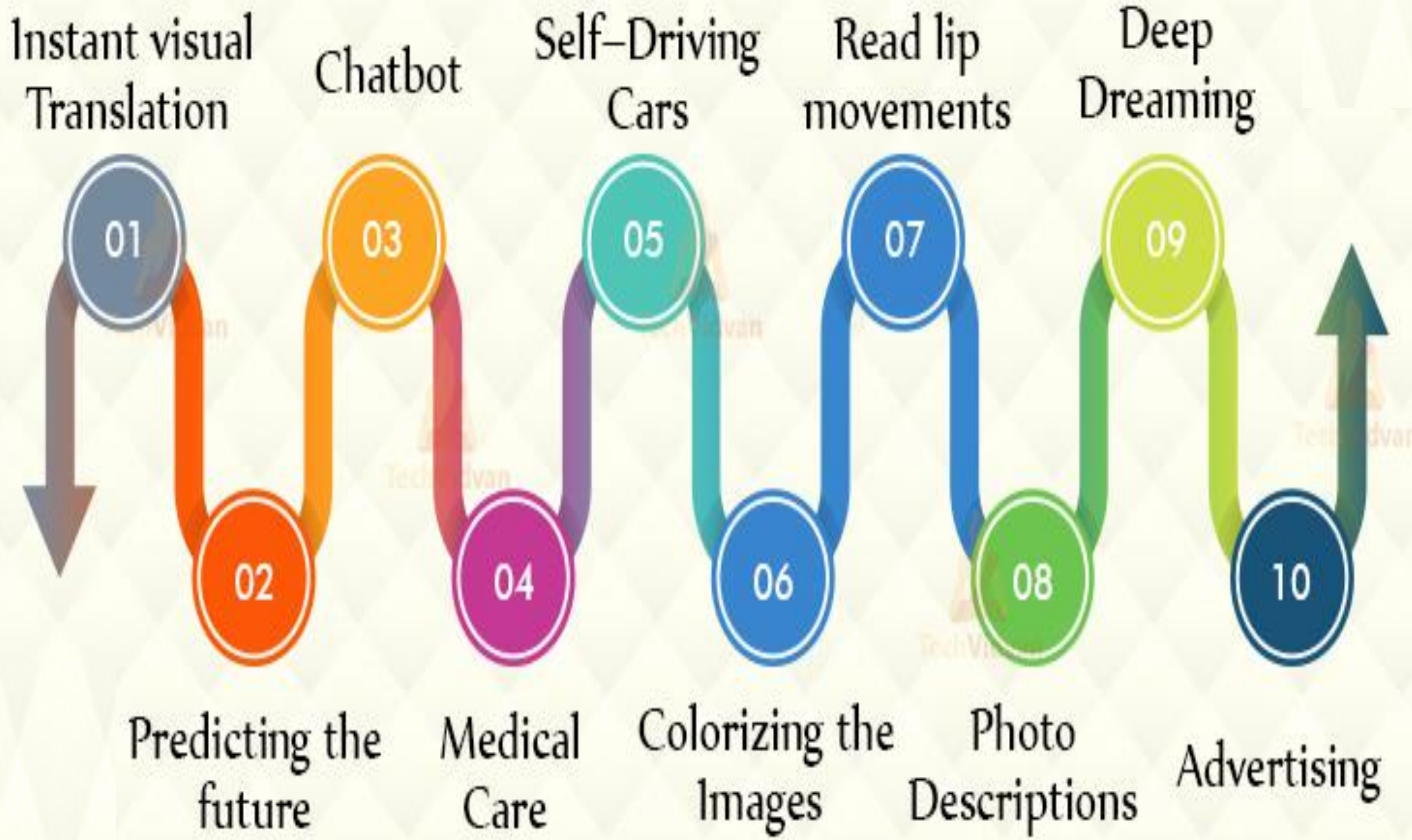
Deep Neural Network (Pretraining)
SVM
Multi-layered Perceptron (Backpropagation)
XOR Problem
ADALINE
Perceptron
Golden Age
Dark Age ("AI Winter")
Electronic Brain
1943
1957
1960
1969
1986
1995
2006
1940
1950
1960
1970
1980
1990
2000
2010
S. McCulloch - W. Pitts
F. Rosenblatt
B. Widrow - M. Hoff
M. Minsky - S. Papert
D. Rumelhart - G. Hinton - R. Wiliams
V. Vapnik - C. Cortes
G. Hinton - S. Ruslan
X AND Y
X OR Y
NOT X
Foward Activity
Backward Error
• Adjustable Weights
• Weights are not Learned
• Learnable Weights and Threshold
• XOR Problem
• Solution to nonlinearly separable problems
• Big computation, local optima and overfitting
• Limitations of learning prior knowledge
• Kernel function: Human Intervention
• Hierarchical feature Learning

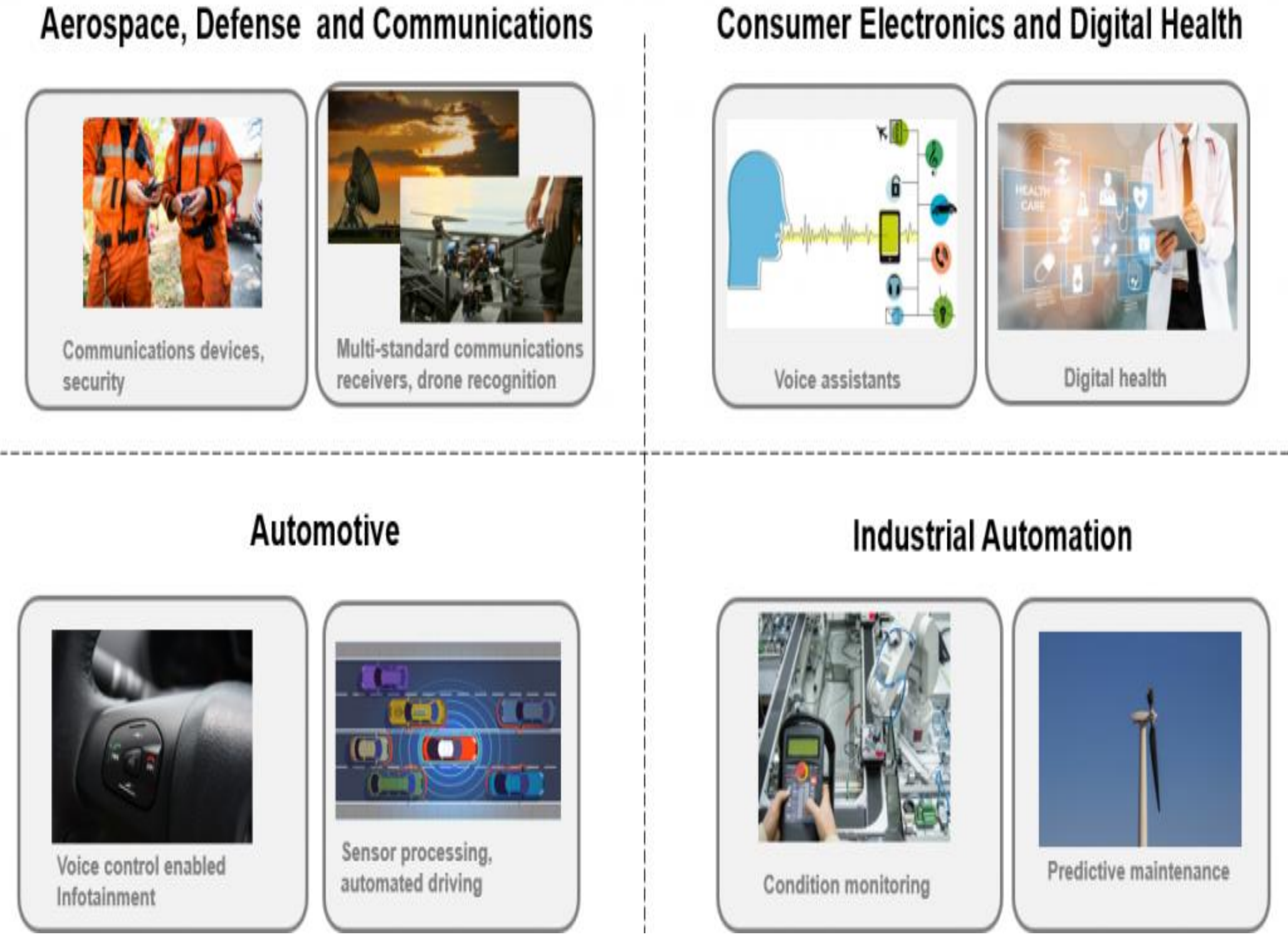

QUANT INSTI
GO ALGO
Pattern recognition using two-layer neural network.
1957
'Neocognitron', an Artificial Neural Network developed
1980
A system which could read handwritten digits developed.
1989
'The Cat Experiment' conducted by Google Brain
2012
1943
First Mathematical Model of Neural Network
1965
First working Neural Networks Model
1985
NETtalk, a program which learnt how to pronounce English words
2009
Launch of ImageNet, a database of labelled data
2014
Facebook developed, DeepFace, to identify faces in images



Control field
Information field
Psychological field
Artificial neural networks
Traffic field
Medical field
Financial field
Educational field

Python Deep Learning Applications




Aerospace, Defense and Communications
Communications devices, security
Multi-standard communications receivers, drone recognition
Consumer Electronics and Digital Health
Voice assistants
Digital health
Automotive
Voice control enabled Infotainment
Sensor processing, automated driving
Industrial Automation
Condition monitoring
Predictive maintenance

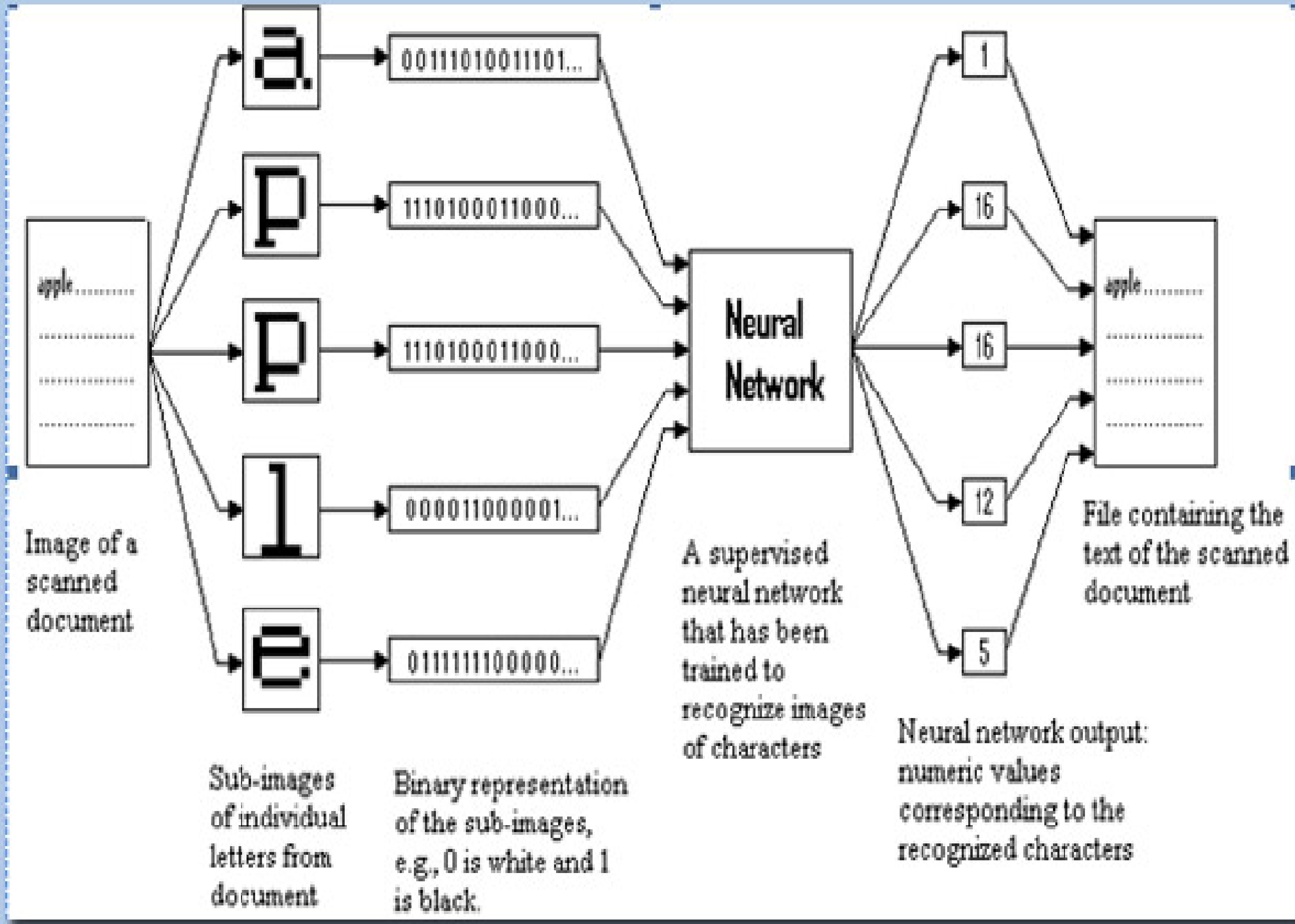
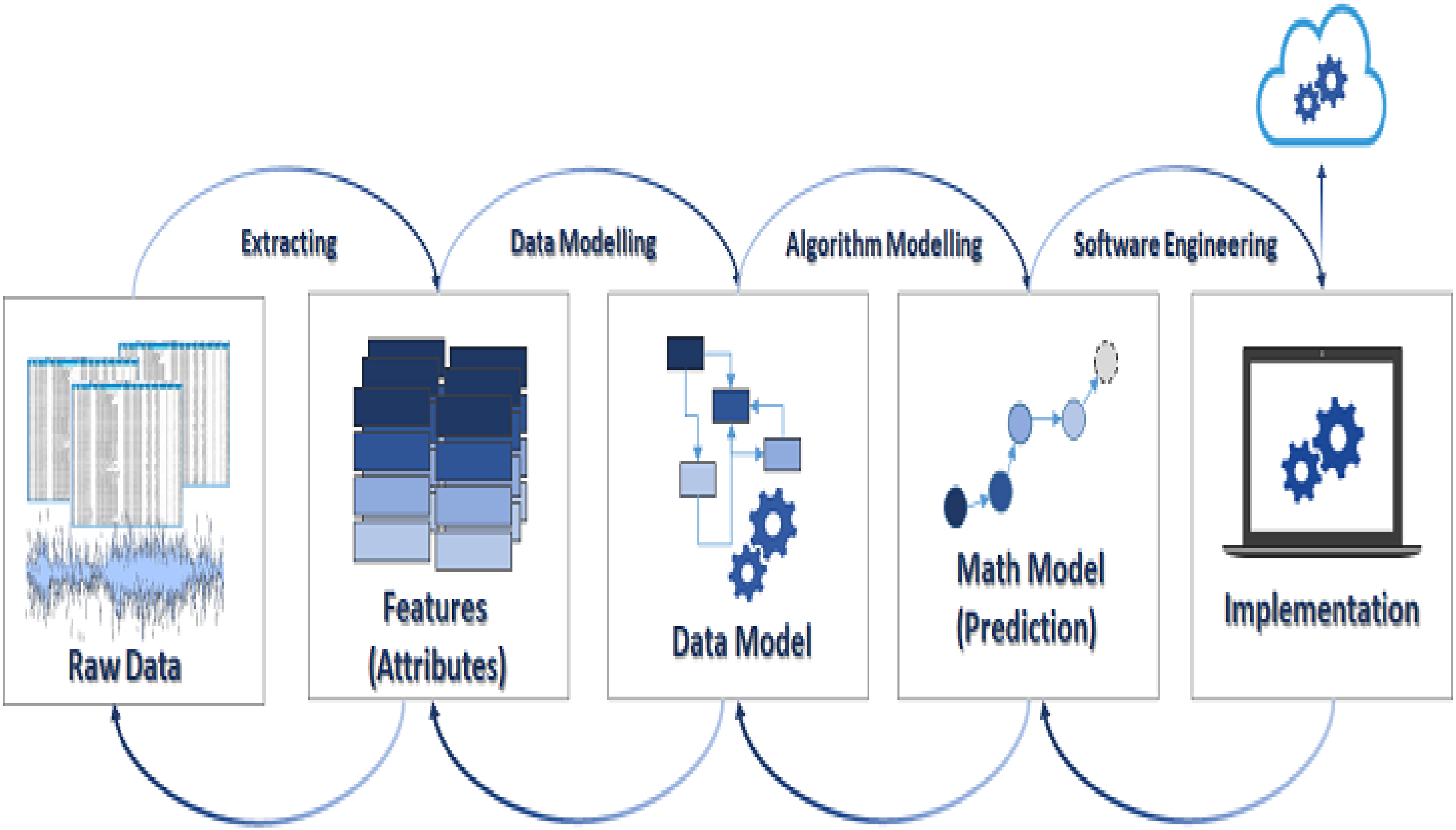
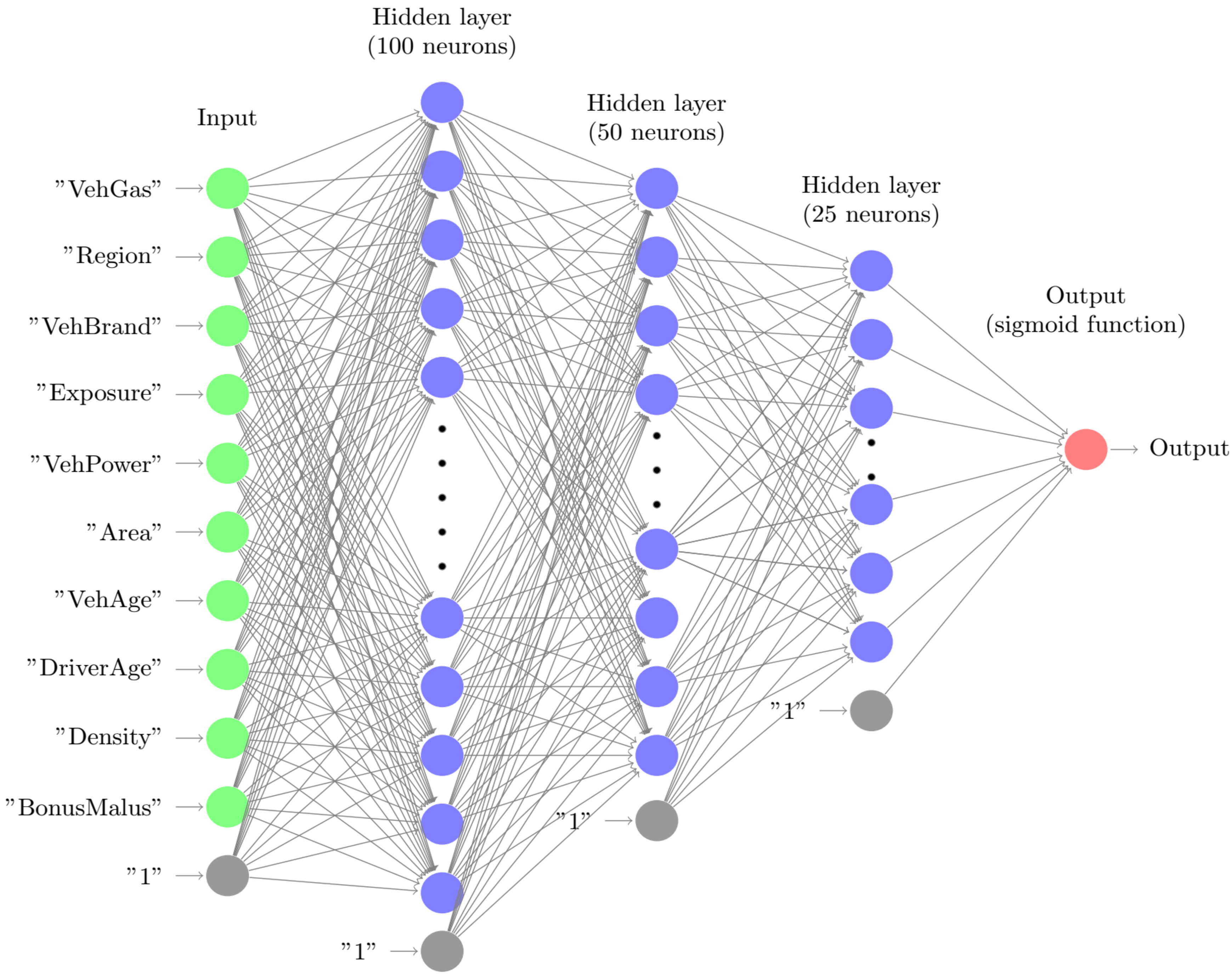
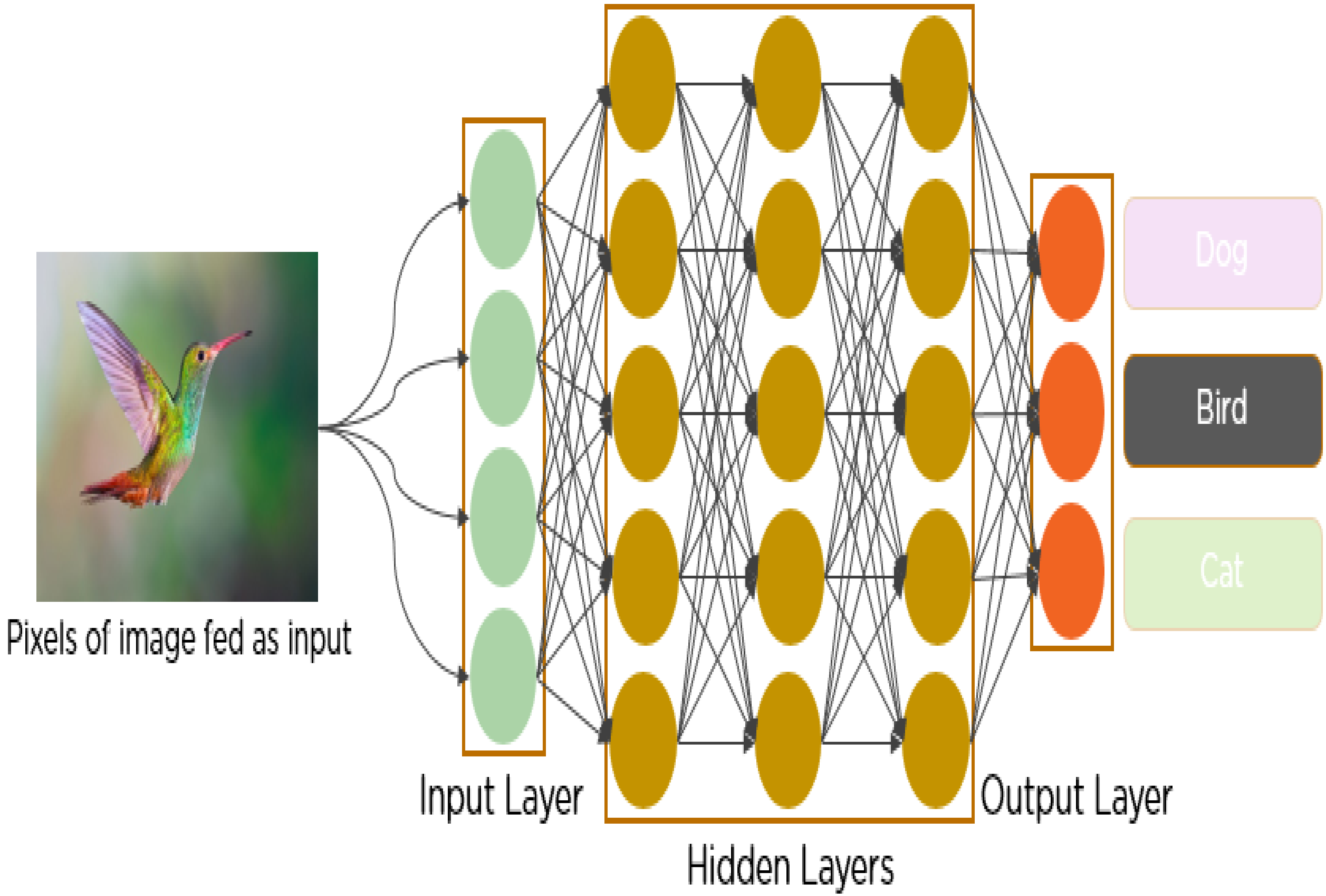


Figure 2: Example of a neural network for OCR.

Deep Learning Modelling

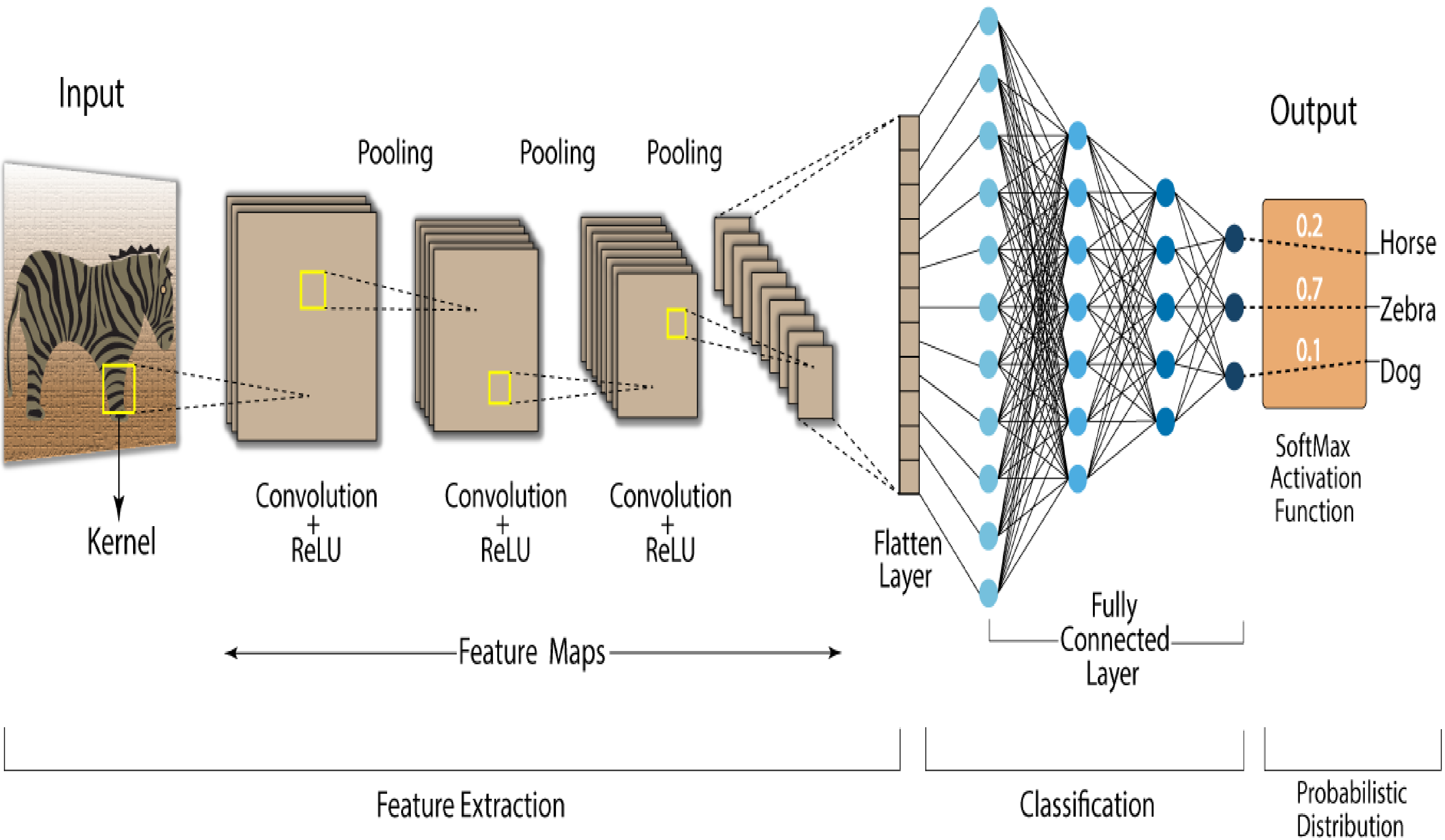



Hidden layer
(100 neurons)
Input
Hidden layer
(50 neurons)
Hidden layer
(25 neurons)
Output
(sigmoid function)
"VehGas"
"Region"
"VehBrand"
"Exposure"
"VehPower"
"Area"
"VehAge"
"DriverAge"
"Density"
"BonusMalus"
"1"
"1"
"1"
"1"
Output

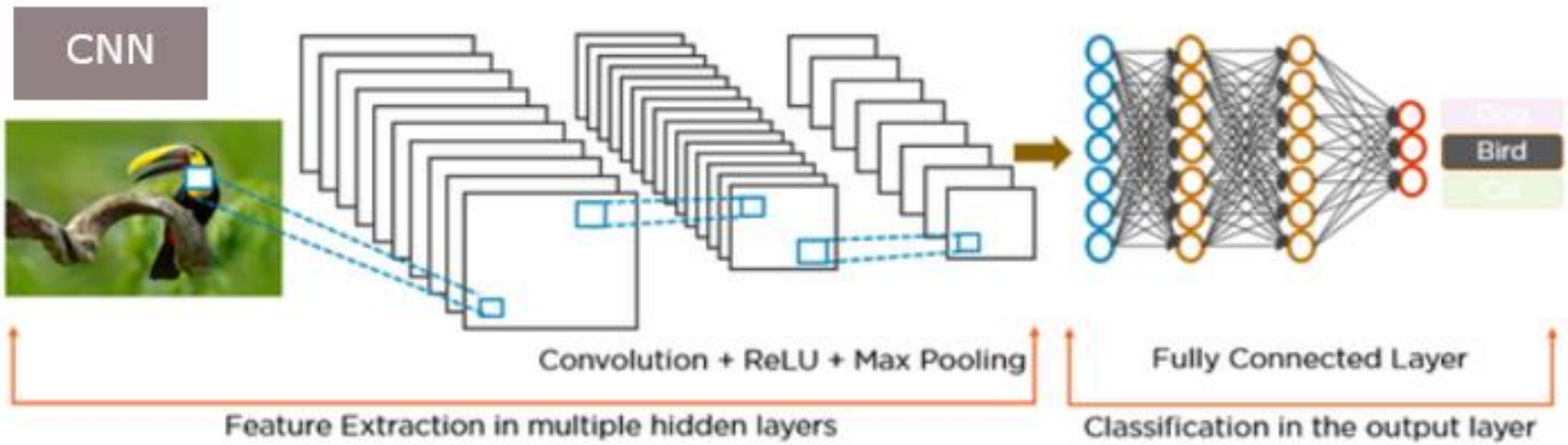


Pixels of image fed as input
Input Layer
Hidden Layers
Output Layer
Dog
Bird
Cat

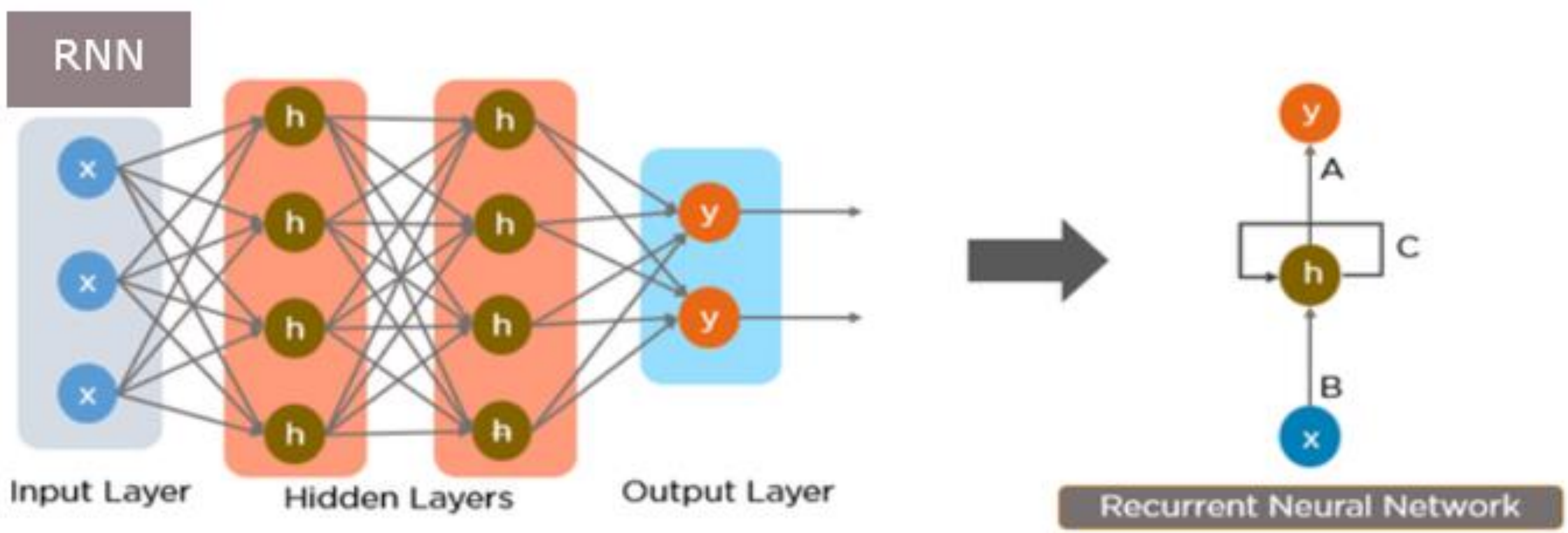
Convolution Neural Network (CNN)



Convolutional Neural Network

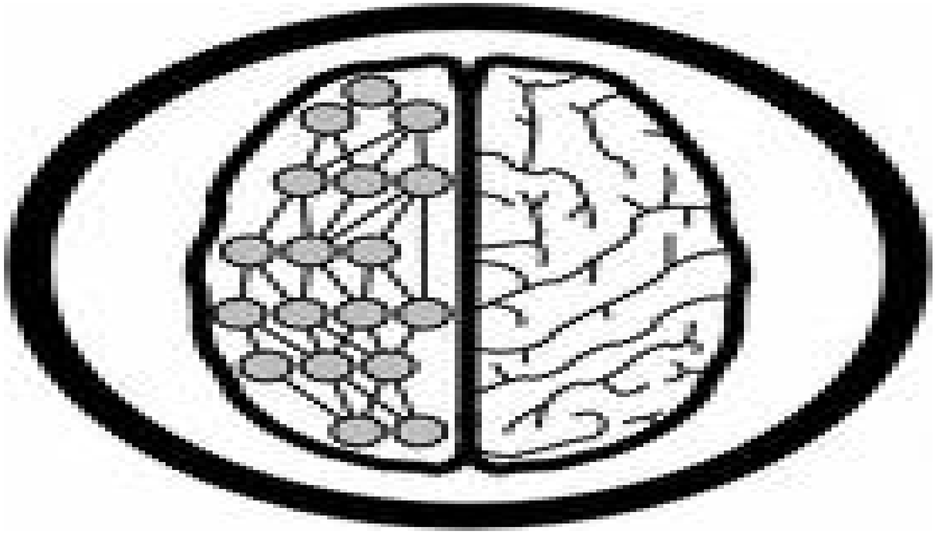


Recurrent Neural Network



How do ANNs work?

- An artificial neural network (ANN) is either a hardware implementation or a computer program which strives to simulate the information processing capabilities of its biological exemplar. ANNs are typically composed of a great number of interconnected artificial neurons. The artificial neurons are simplified models of their biological counterparts.
- ANN is a technique for solving problems by constructing software that works like our brains.

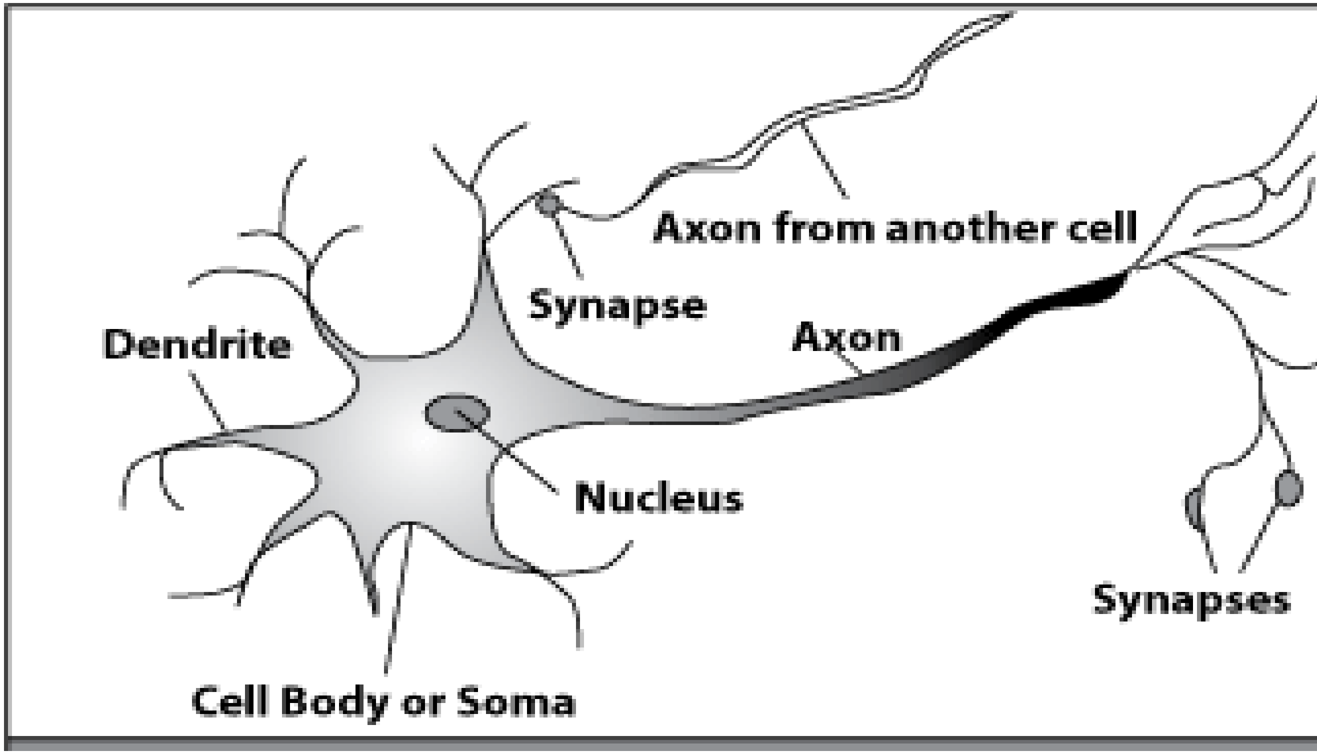


Characteristics of BNN and ANN

Characteristics	Biological Neural Network	Artificial Neural Network
Speed	Processes information at a slower rate. Response time is measured in milliseconds.	Information is processed at a faster rate. The response time is measured in nanoseconds.
Processing	Massively parallel processing.	Serial processing.
Size & Complexity	An extremely intricate and dense network of linked neurons of the order of 1011 neurons and 1015 interconnections.	Size and complexity are reduced. It is incapable of performing sophisticated pattern recognition tasks.
Storage	An extremely intricate and dense network of linked neurons with 1015 interconnections, including neurons on the order of 1011.	The term "replaceable information storage" refers to the practice of replacing fresh data with old data.
Fault tolerance	The fact that information storage is flexible means that new information may be added by altering the connectivity strengths without deleting existing information.	Intolerant of faults. In the event of a system failure, corrupt data cannot be recovered.
Control Mechanism	There is no unique control mechanism outside of the computational task.	Controlling computer activity is handled by a control unit.

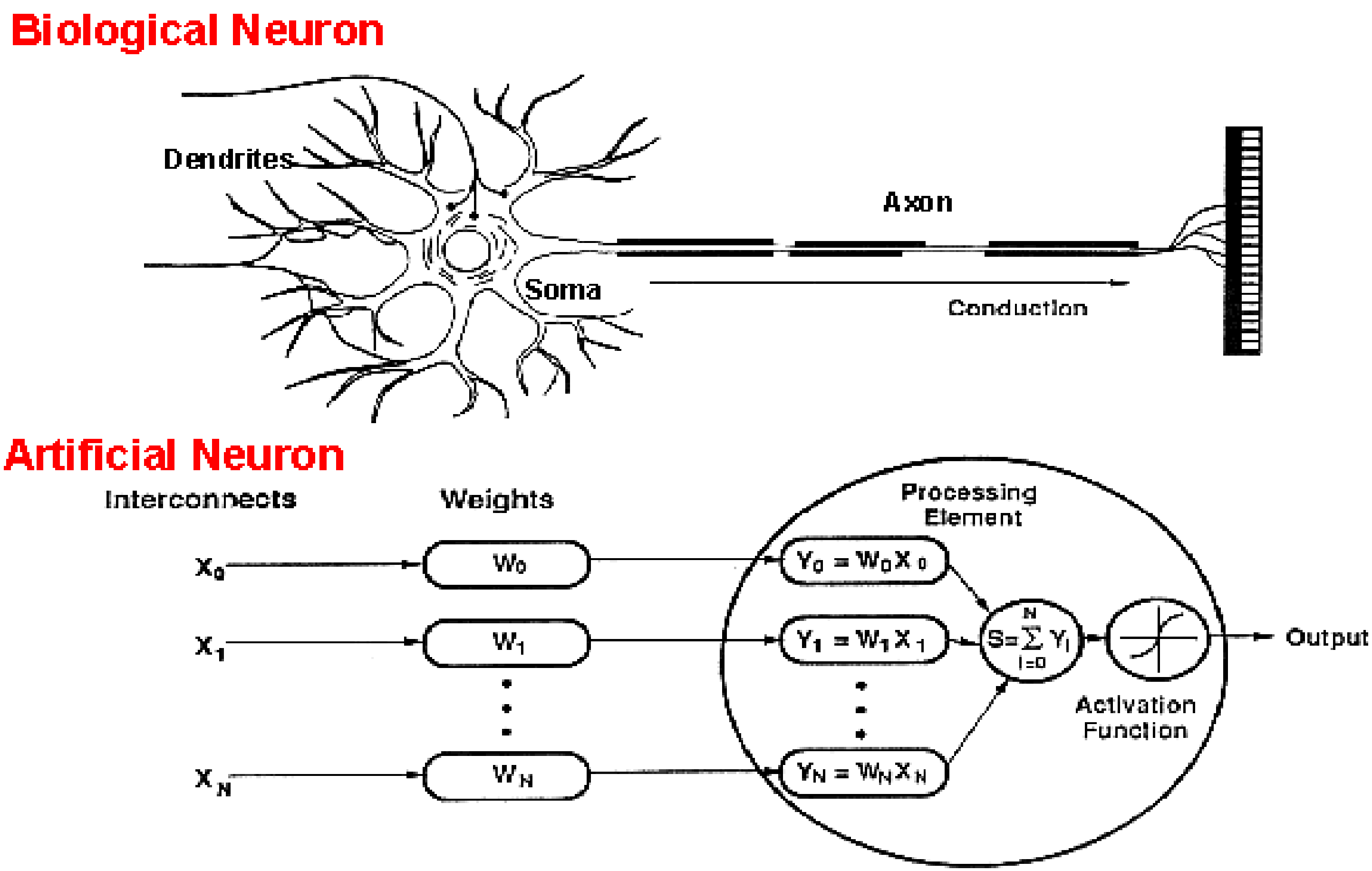
How do our brains work?

- A processing element



Dendrites: Input
Cell body: Processor
Synaptic: Link
Axon: Output

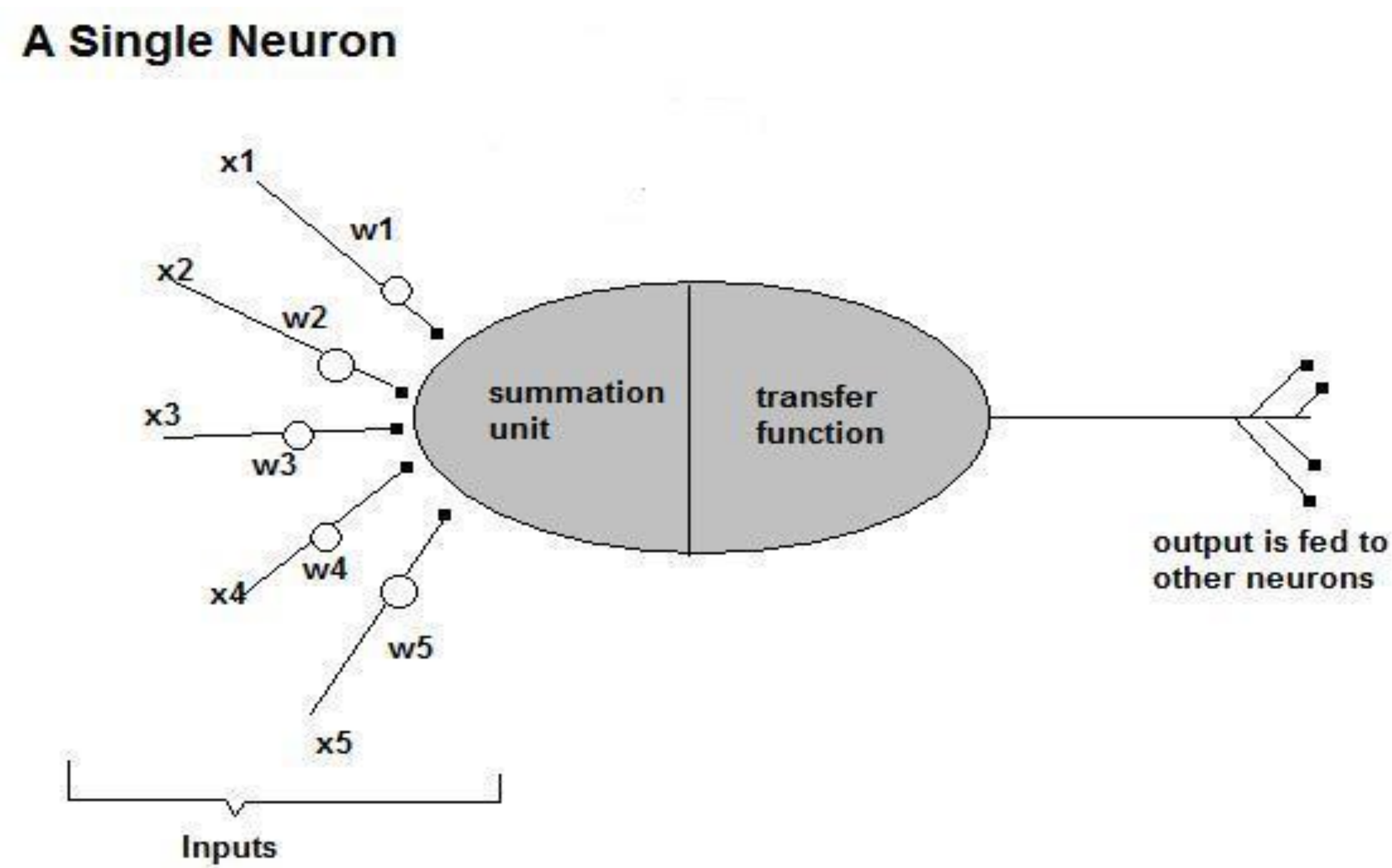
How do ANNs work?



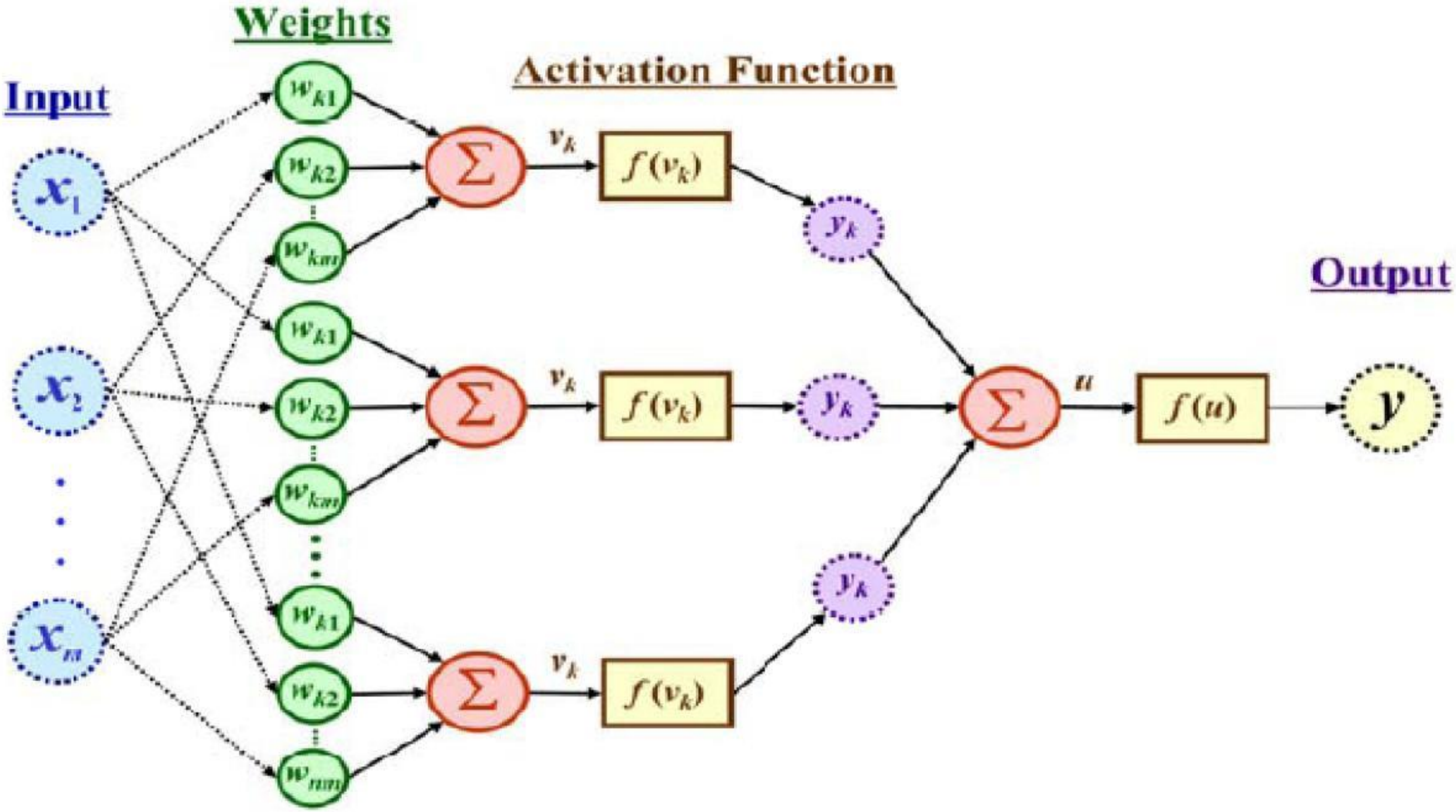
An artificial neuron is an imitation of a human neuron

How do ANNs work?

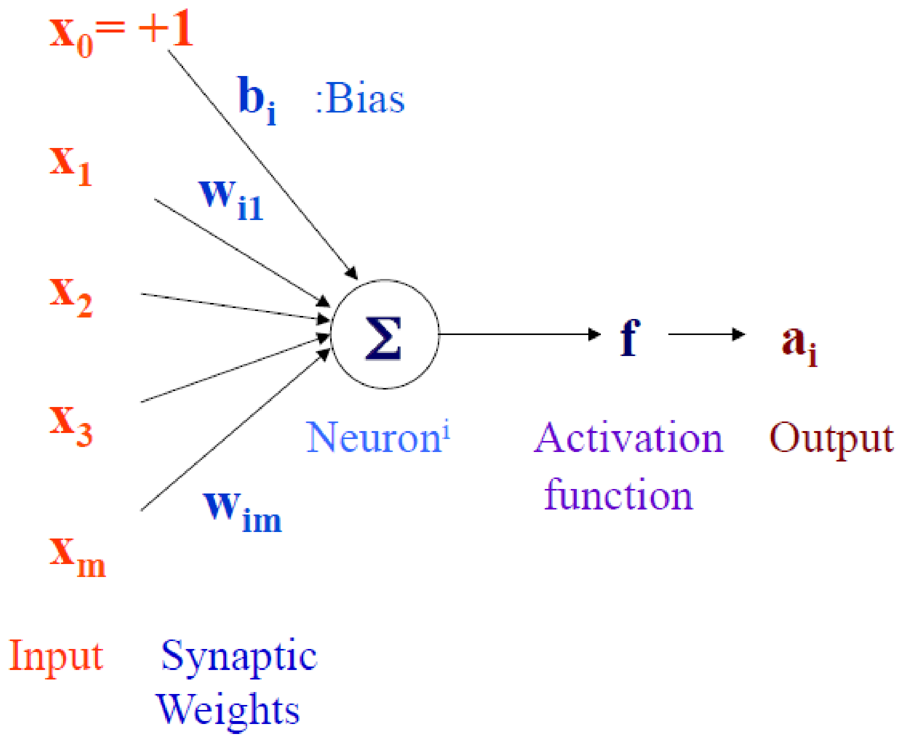
- Now, let us have a look at the model of an artificial neuron.



The output is a function of the input, that is affected by the weights, and the transfer functions



Artificial Neuron Model



Bias

$$a_i = f\,(n_i) = f\,\left(\sum_{j=1}^{n} w_{ij}x_j + b_i\right)$$

An artificial neuron:

- computes the weighted sum of its input (called its **net input**)
- adds its bias
- passes this value through an activation function

We say that the neuron “fires” (i.e. becomes active) if its output is above zero.

Bias

Bias can be incorporated as another weight clamped to a fixed input of +1.0

This extra free variable (bias) makes the neuron more powerful.

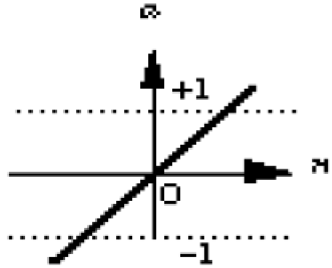
$$a_i = f\,(n_i) = f\,\left(\sum_{j=0}^{n} w_{ij} x_j\right) = f(\mathbf{w}_i . \mathbf{x}_j)$$

Activation functions

Also called the squashing function as it limits the amplitude of the output of the neuron.

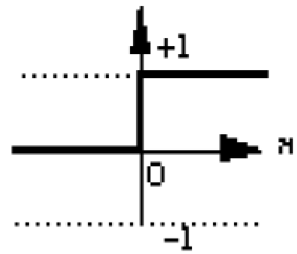
Many types of activations functions are used:

- linear: $a = f(n) = n$

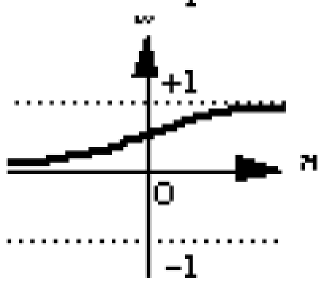


- threshold: (hardlimiting) $a = \{1$ if $n >= 0$

  0 if $n < 0$

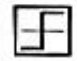


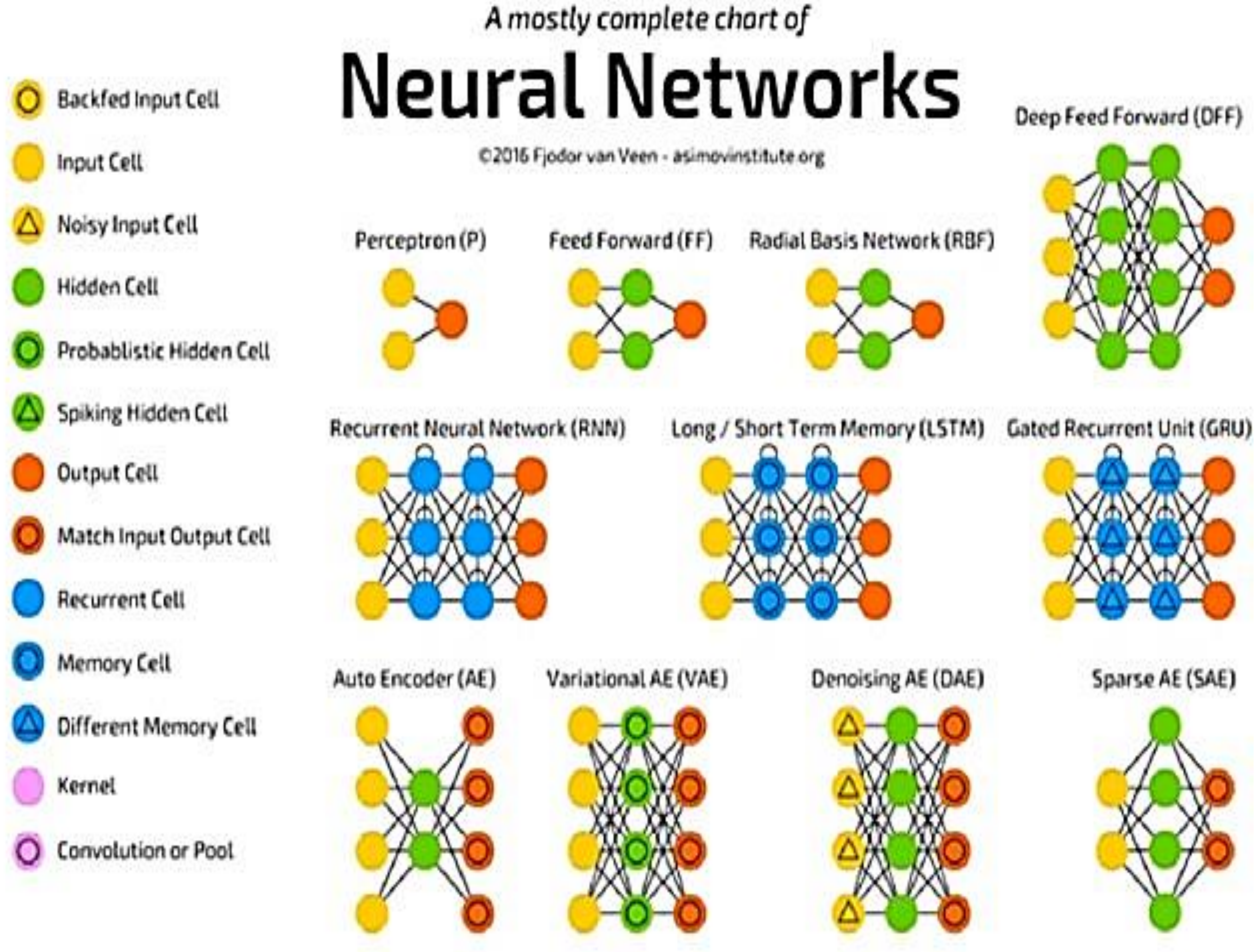
- sigmoid: $a = 1/(1+e^{-n})$



- ...

Activation Functions

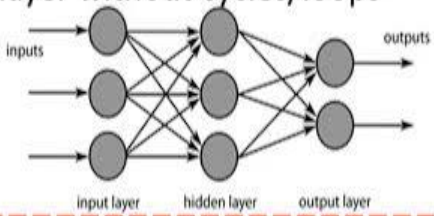
Name	Input/Output Relation	Icon	MATLAB Function
Hard Limit	$a = 0 \quad n < 0$ $a = 1 \quad n \geq 0$		hardlim
Symmetrical Hard Limit	$a = -1 \quad n < 0$ $a = +1 \quad n \geq 0$		hardlims
Linear	$a = n$		purelin
Saturating Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n \leq 1$ $a = 1 \quad n > 1$		satlin
Symmetric Saturating Linear	$a = -1 \quad n < -1$ $a = n \quad -1 \leq n \leq 1$ $a = 1 \quad n > 1$		satlins
Log-Sigmoid	$a = \frac{1}{1 + e^{-n}}$		logsig
Hyperbolic Tangent Sigmoid	$a = \frac{e^{n} - e^{-n}}{e^{n} + e^{-n}}$		tansig
Positive Linear	$a = 0 \quad n < 0$ $a = n \quad 0 \leq n$		poslin
Competitive	$a = 1$ neuron with max n $a = 0$ all other neurons	$\boxed{\mathrm{C}}$	compet



A mostly complete chart of
Neural Networks
©2016 Fjodor van Veen - asimovinstitute.org
Backfed Input Cell
Input Cell
Noisy Input Cell
Hidden Cell
Probablistic Hidden Cell
Spiking Hidden Cell
Output Cell
Match Input Output Cell
Recurrent Cell
Memory Cell
Different Memory Cell
Kernel
Convolution or Pool
Perceptron (P)
Feed Forward (FF)
Radial Basis Network (RBF)
Deep Feed Forward (DFF)
Recurrent Neural Network (RNN)
Long / Short Term Memory (LSTM)
Gated Recurrent Unit (GRU)
Auto Encoder (AE)
Variational AE (VAE)
Denoising AE (DAE)
Sparse AE (SAE)

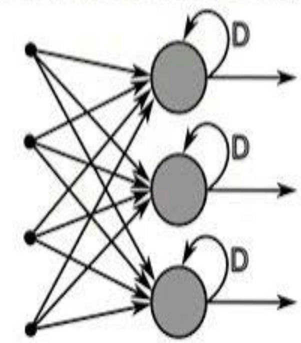
TYPES OF ARTIFICIAL NEURAL NETWORKS

- **Feedforward Neural Networks**
  - One to one mapping
  - Classification or regression
  - information flows from input layer directly through hidden layers then to the output layer without cycles/loops



LET'S VISUALIZE THESE TWO TYPES OF NETWORKS USING GOOGLE TENSORFLOW PLAYGROUND AND TENSORSPACE.JS

- **Recurrent Neural Networks**
  - Considers temporal (time) information
  - Used for sequences prediction (future stock prices, translation, text.
  - Long short term memory (LSTM) and GRU (Gated Recurrent Unit)



- **Convolutional Neural Networks**
  - Image classification

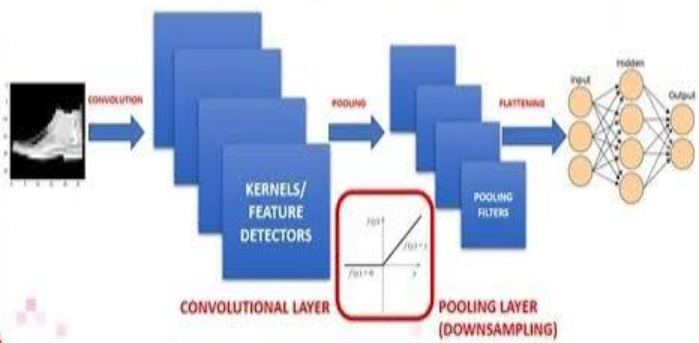
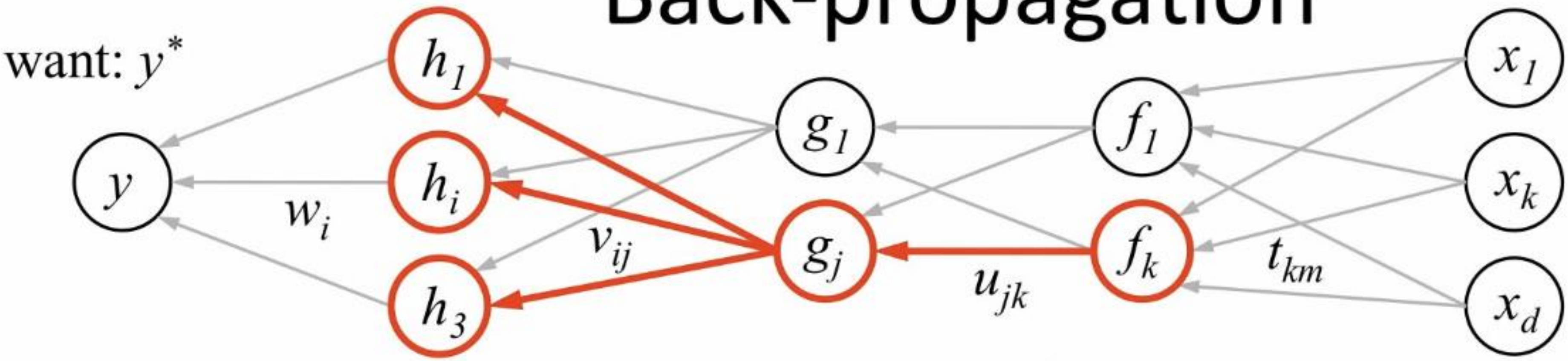


Photo Credit: https://commons.wikimedia.org/wiki/File:RecurrentLayerNeuralNetwork_english.png
Photo Credit: https://commons.wikimedia.org/wiki/File:Artificial_neural_network.svg

Back-propagation



1. receive new observation $\boldsymbol{x} = [x_1 \ldots x_d]$ and target y^*
2. **feed forward:** for each unit g_j in each layer 1…L
   compute g_j based on units f_k from previous layer: $g_j = \sigma\left(u_{j0} + \sum_k u_{jk} f_k\right)$
3. get prediction y and error $(y\text{-}y^*)$
4. **back-propagate error:** for each unit g_j in each layer L…1

(a) compute error on g_j

$$\underbrace{\frac{\partial E}{\partial g_j}}_{\text{should } g_j \text{ be higher or lower?}} = \sum_i \underbrace{\sigma'(h_i) v_{ij}}_{\text{how } h_i \text{ will change as } g_j \text{ changes}} \underbrace{\frac{\partial E}{\partial h_i}}_{\text{was } h_i \text{ too high or too low?}}$$

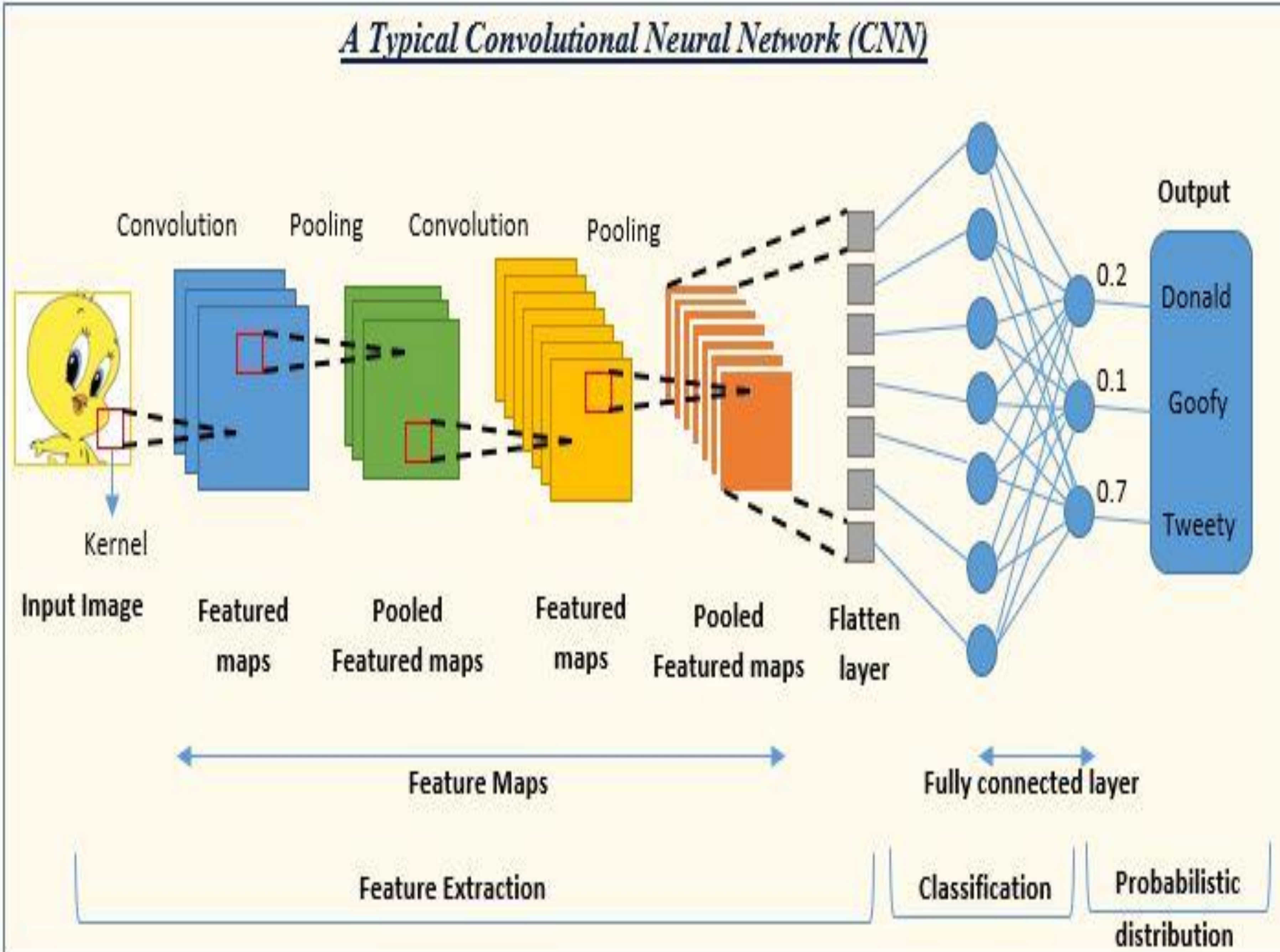
(b) for each u_{jk} that affects g_j

(i) compute error on u_{jk}

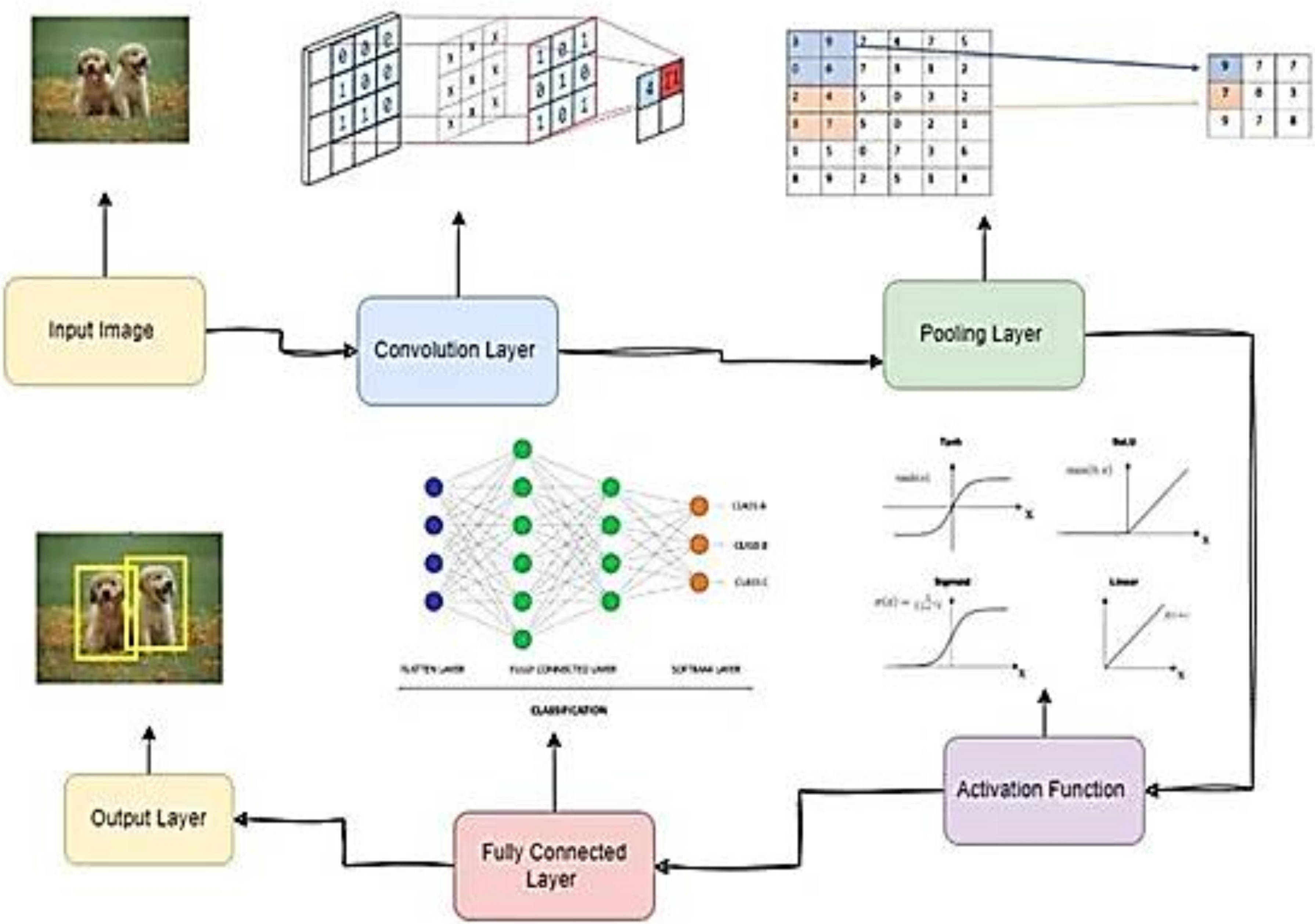
$$\frac{\partial E}{\partial u_{jk}} = \underbrace{\frac{\partial E}{\partial g_j}}_{\text{do we want } g_j \text{ to be higher/lower}} \underbrace{\sigma'(g_j) f_k}_{\text{how } g_j \text{ will change if } u_{jk} \text{ is higher/lower}}$$

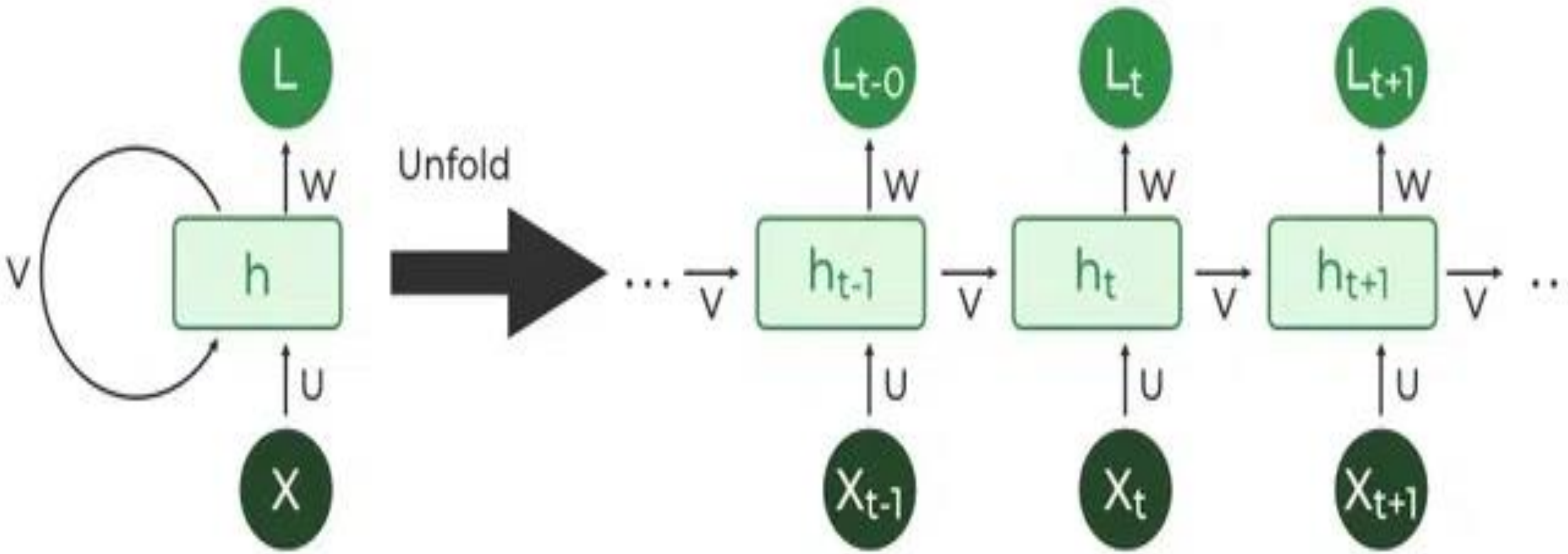
(ii) update the weight

$$u_{jk} \leftarrow u_{jk} - \eta \frac{\partial E}{\partial u_{jk}}$$

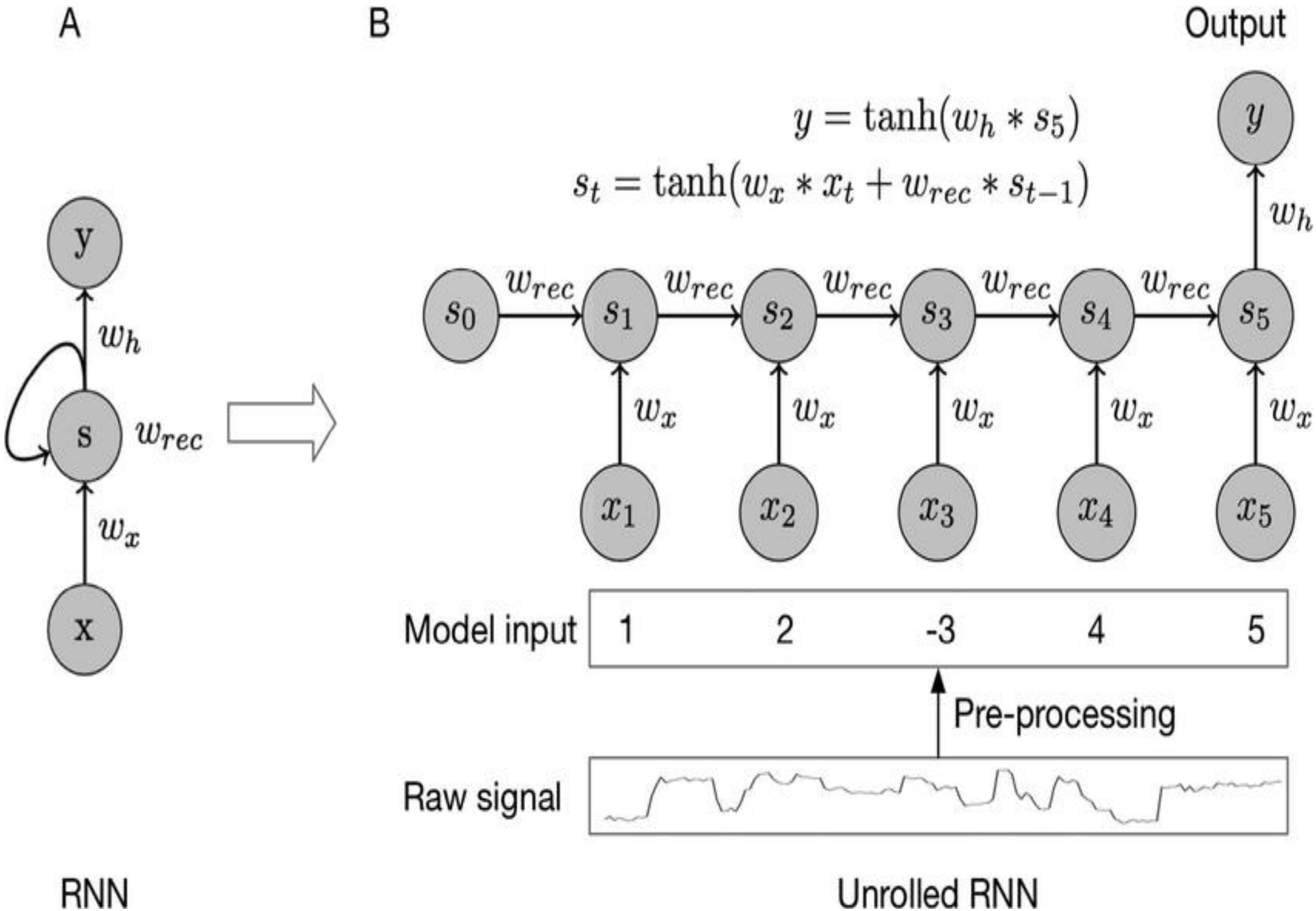

A Typical Convolutional Neural Network (CNN)
Convolution
Pooling
Convolution
Pooling
Output
0.2
Donald
0.1
Goofy
0.7
Tweety
Kernel
Input Image
Featured maps
Pooled Featured maps
Featured maps
Pooled Featured maps
Flatten layer
Feature Maps
Fully connected layer
Feature Extraction
Classification
Probabilistic distribution

CNN Components





L
W
Unfold
V
h
U
X
L_{t-0}
L_t
L_{t+1}
W
W
W
...
V
h_{t-1}
V
h_t
V
h_{t+1}
V
...
U
U
U
X_{t-1}
X_t
X_{t+1}



A
B
Output
$y = \tanh(w_h * s_5)$
$s_t = \tanh(w_x * x_t + w_{rec} * s_{t-1})$
y
w_h
s
w_{rec}
w_x
x
s_0
s_1
s_2
s_3
s_4
s_5
x_1
x_2
x_3
x_4
x_5
y
Model input
1
2
-3
4
5
Pre-processing
Raw signal
RNN
Unrolled RNN

Which framework should you use?

Keras

PyTorch

TensorFlow has implemented various levels of abstraction to make implementation easy. This also makes debugging easy

It is simple and easy, but not as fast as TensorFlow.
It is more user-friendly than any other deep learning API

It is the preferred deep learning API for teachers but is not as widely used in production as TensorFlow. Faster, but lower GPU utilization