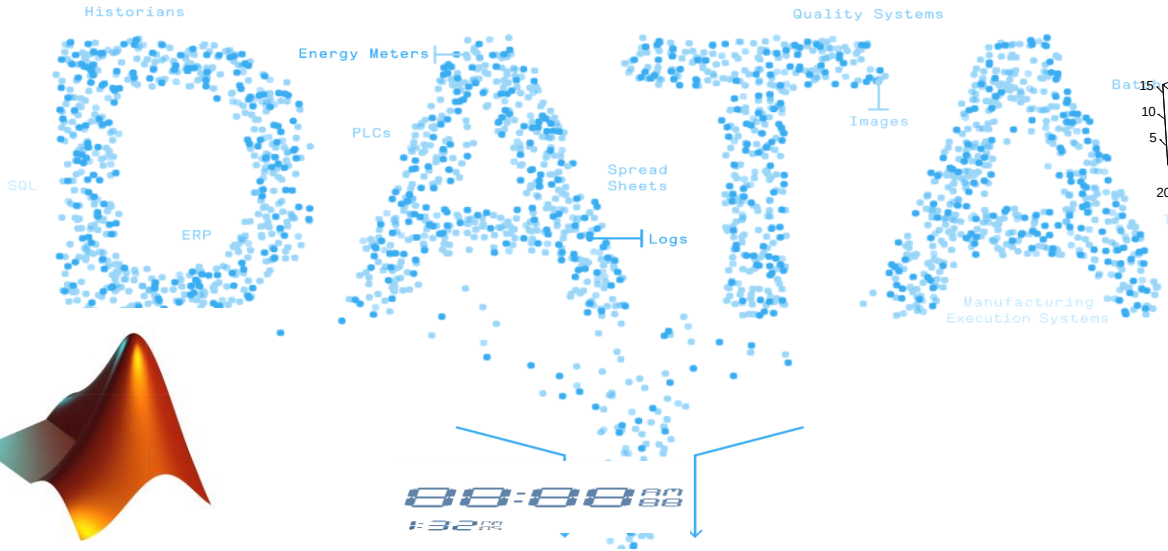
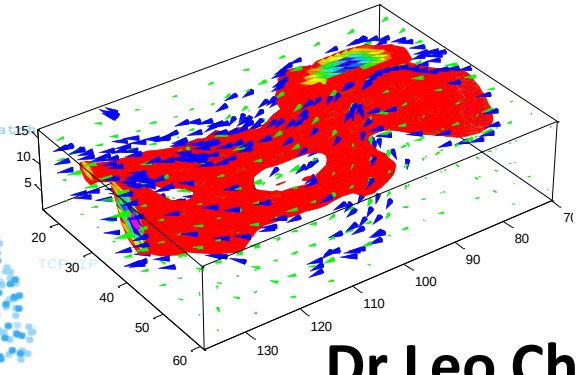
Introduction to Artificial Intelligence

- 03-01 Artificial Neural Networks



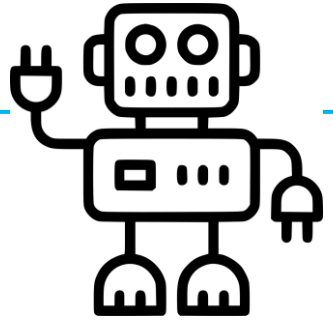


Dr Leo Chen

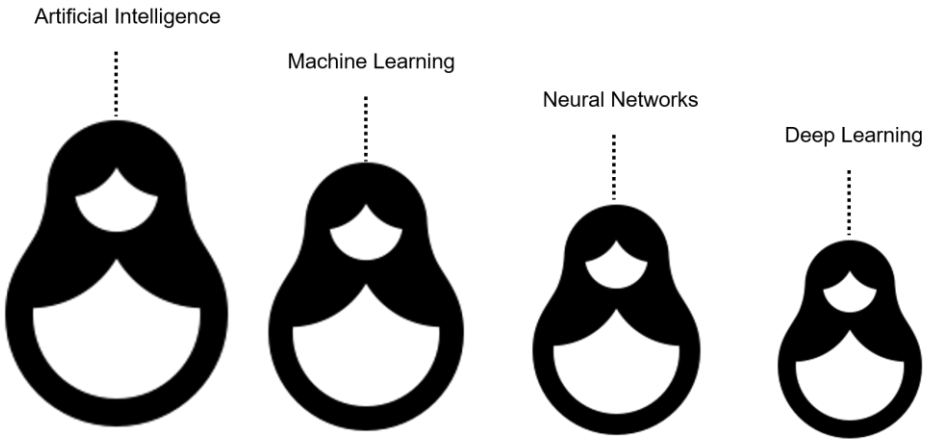
leo.chen@ieee.org

23/Feb/2024

Module Contents



1. Introduction
2. Evolutionary Computation
3. Artificial Neural Network
4. Fuzzy Logic and Fuzzy Systems
5. More AI Subsets
6. AI and Industry 4.0
7. AI Applications
8. Labs
9. Courseworks

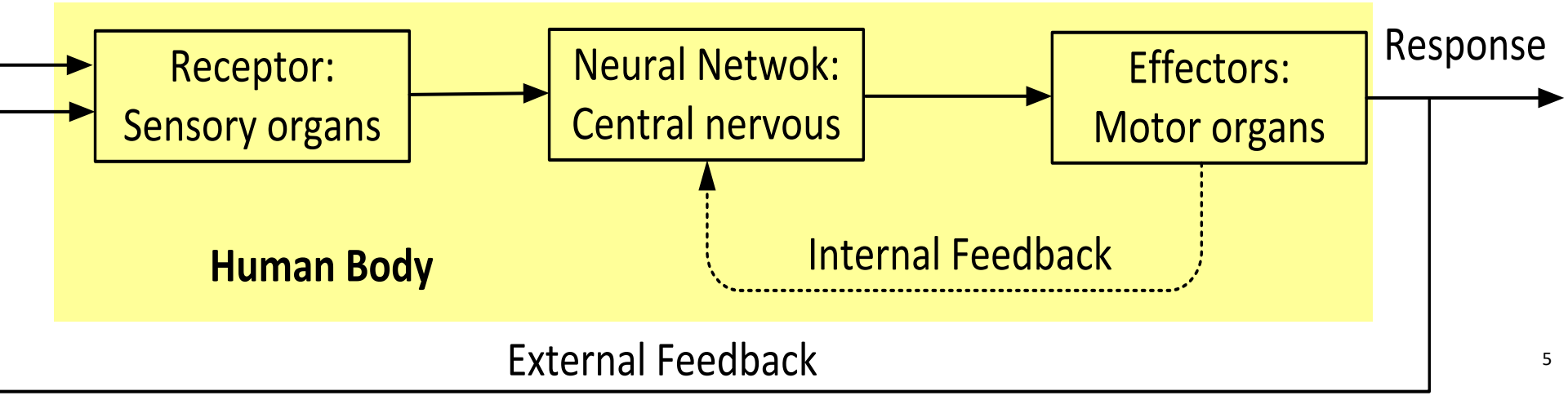


Chapter Contents

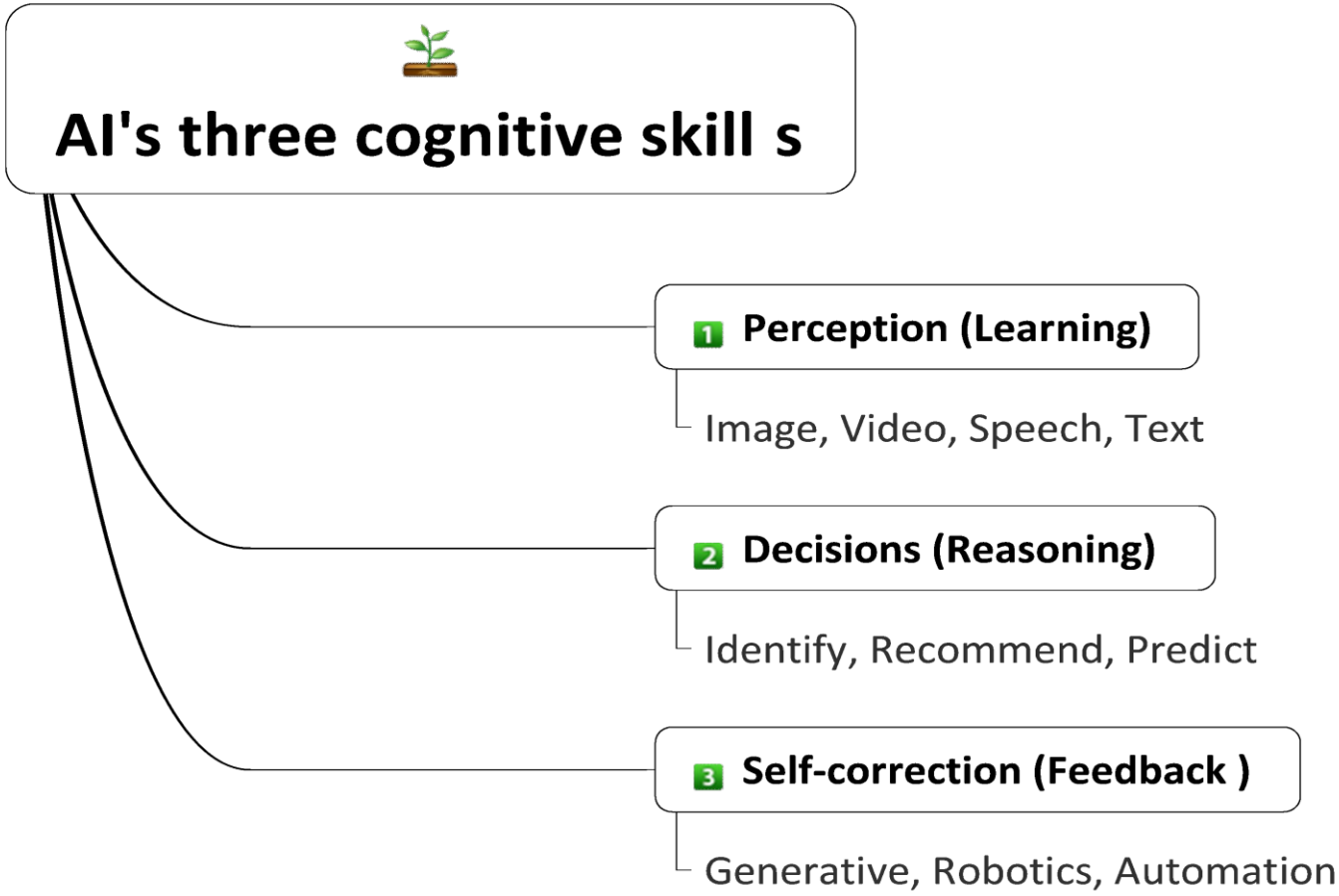
Introduction

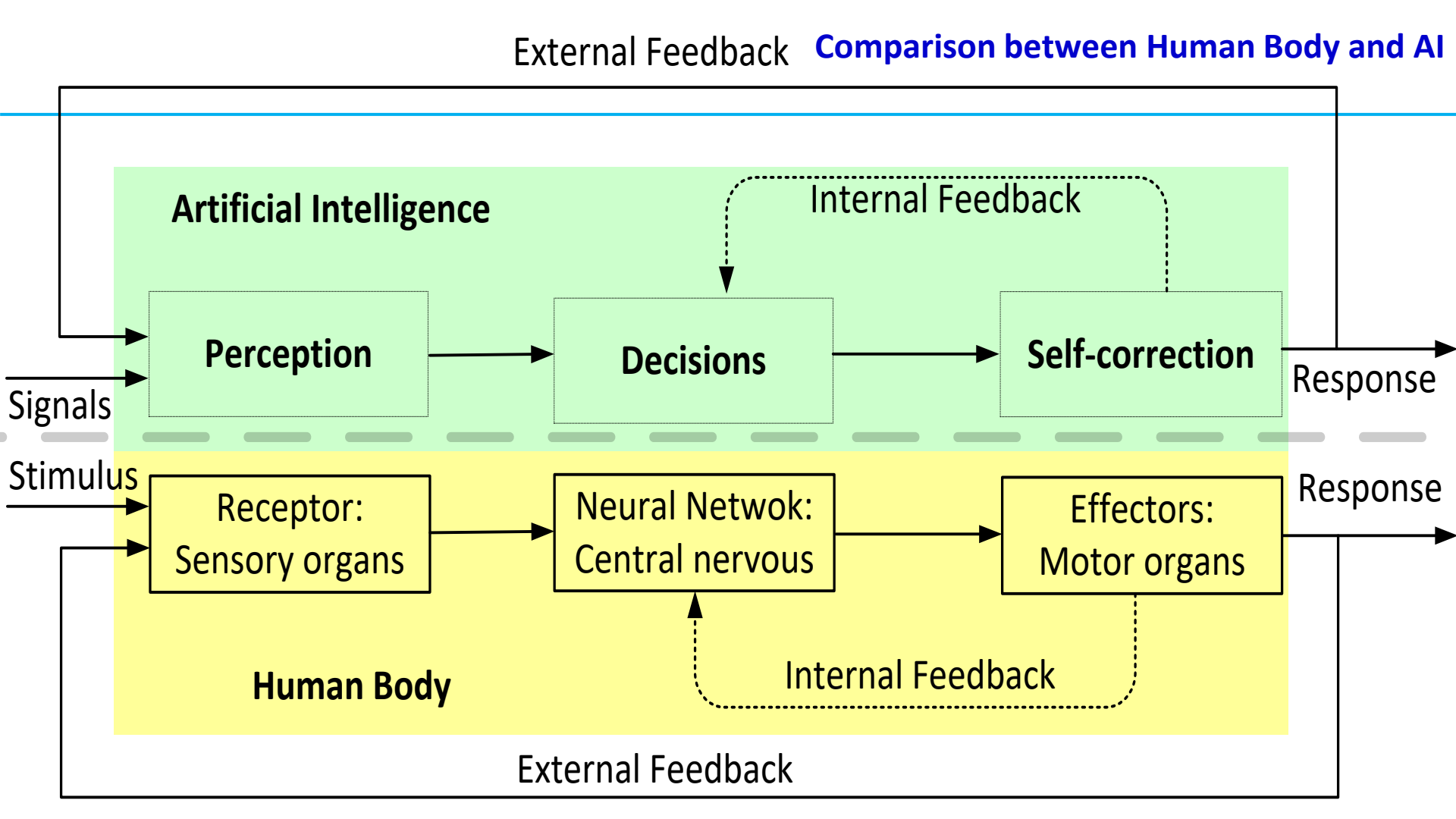
- Artificial Neural Networks (**ANN**) are algorithms based on **brain function** and are **used to** model complicated patterns and forecast issues.
- The ANN is a method that **arose from** the concept of the **human brain's Biological Neural Networks**.
- The development of ANN was the result of an attempt to **replicate** the **working modes** of the human brain.
- The **working modes** of **ANN** are extremely similar to those of biological neural networks, although they are **not** identical. ANN algorithm accepts only **numeric** and **structured data**.

How Does Human Body Perform



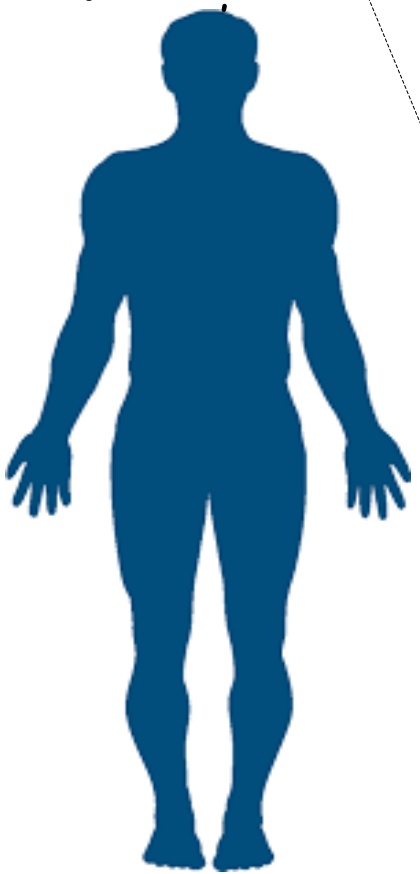
How Does AI Perform following Human Body



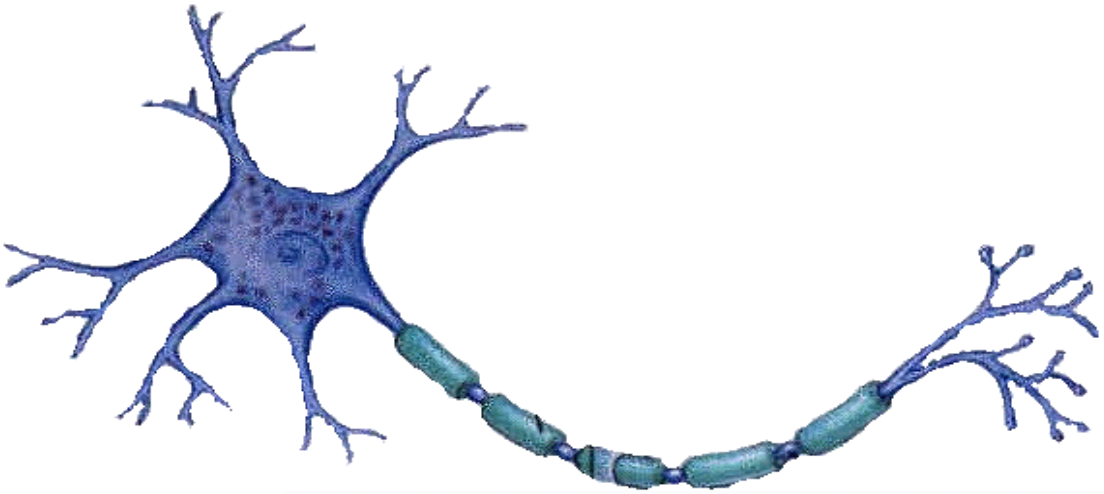

External Feedback
Comparison between Human Body and AI
Artificial Intelligence
Internal Feedback
Perception
Decisions
Self-correction
Response
Signals
Stimulus
Receptor:
Sensory organs
Neural Netwok:
Central nervous
Effectors:
Motor organs
Response
Human Body
Internal Feedback
External Feedback

Neuron

/ˈnjʊərɒn/

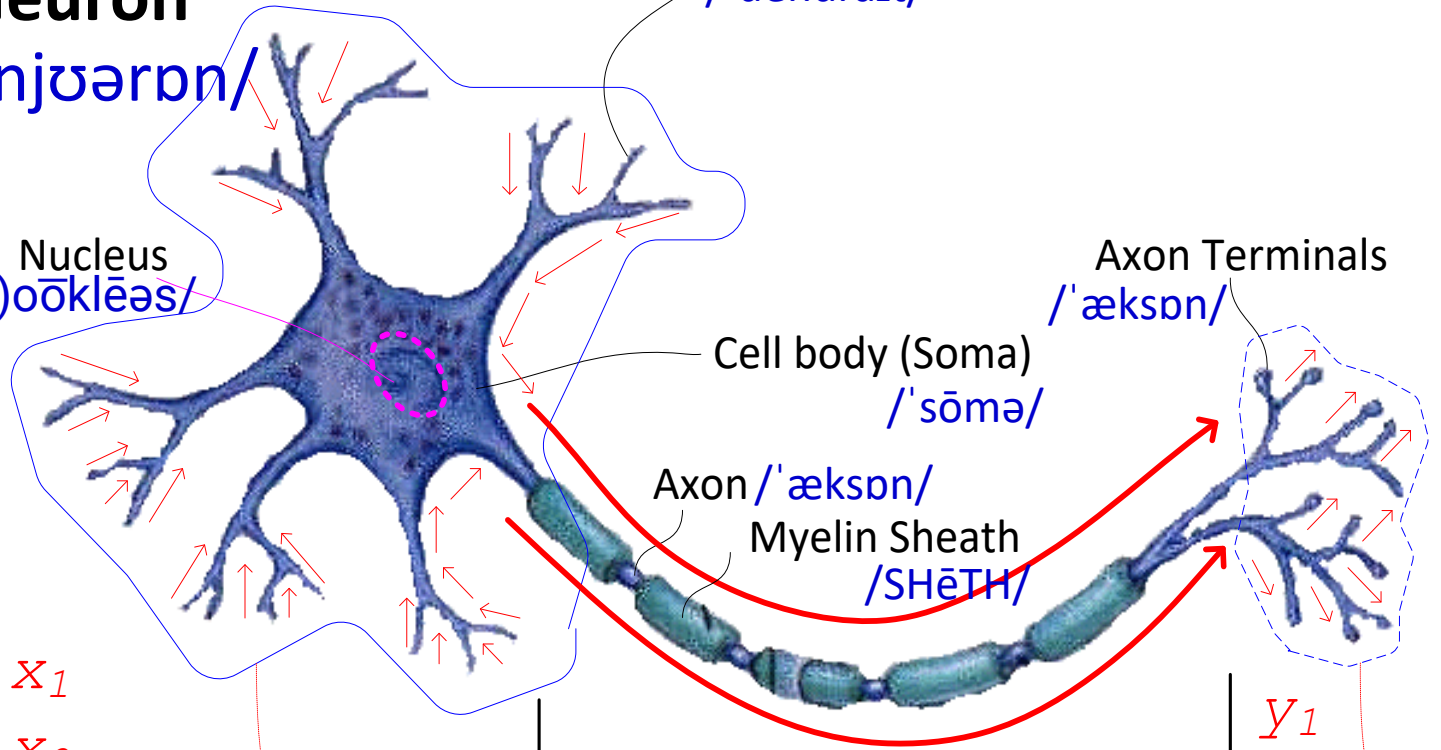


Neuron in 2mins (2mins)



Neuron
/ˈnjʊərɒn/

See FAQ 10 Terminology



Concepts [1,2,3]

- Biological **Neurons** (also called **nerve cells**) or simply **neurons** are the fundamental units of the brain and nervous system, the cells responsible for *receiving* **sensory input** from the external world via dendrites, *process* it and gives the *output* through **Axons**.
- **Axon** /ˈæksɒn/ **:** It is a *long*, *thin*, *tubular* structure that works like a *transmission line*.

The Neuron (5mins)

Concepts

- **Cell body (Soma):** The **body** of the **neuron cell** contains the **nucleus** and carries out biochemical transformation necessary to the life of neurons.
- **Dendrites** /ˈdendraɪt/ **:** Each neuron has fine, hair-like tubular structures (extensions) around it. They branch out into a tree around the cell body. They accept incoming signals.

Structure and function of Neuron - Animation (4mins)

Concepts

- **Synapse** /ˈsaɪnæps/ **:** Neurons are connected to one another in a *complex spatial arrangement*.

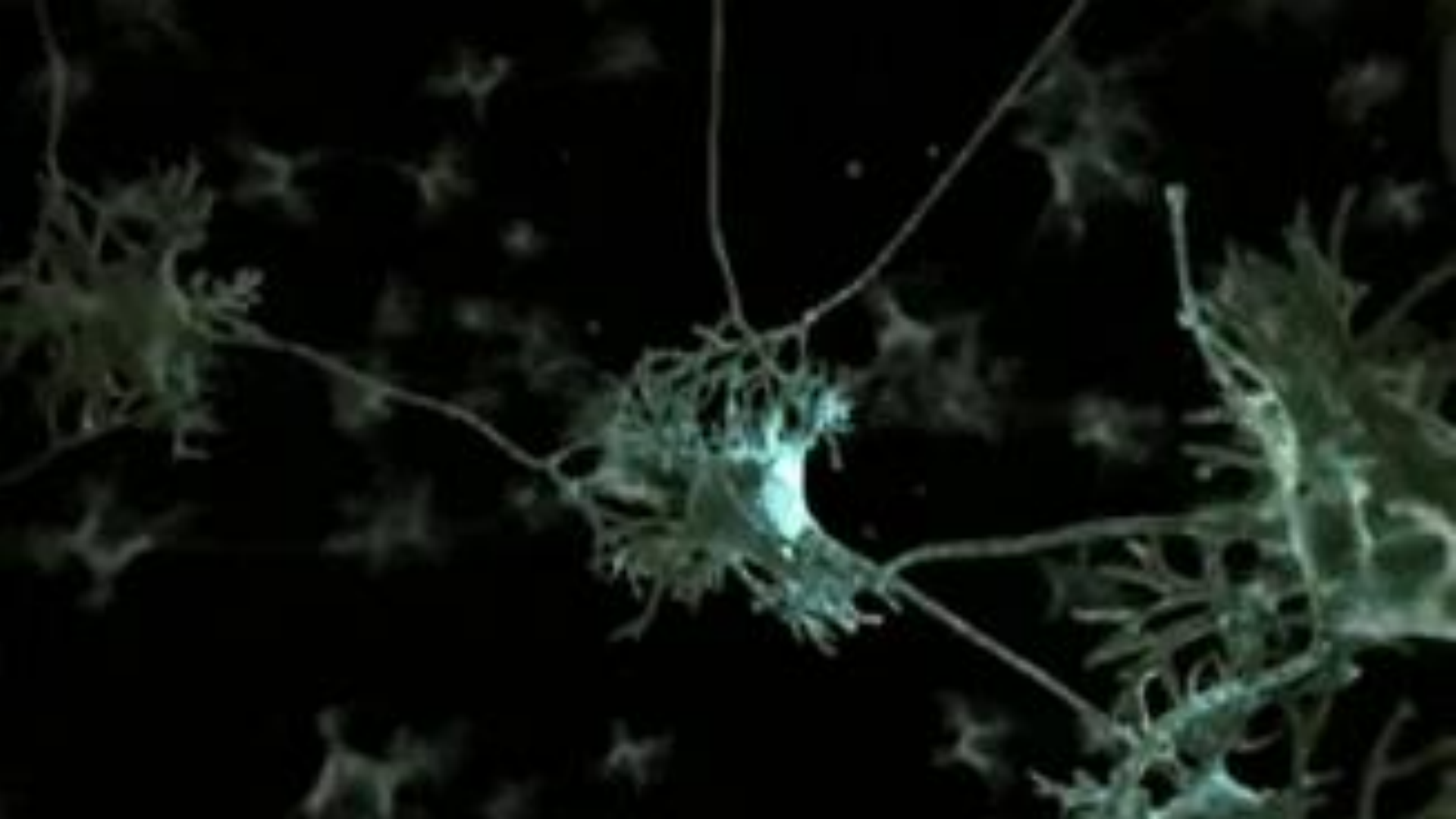
At the end of the axon are highly complex and specialized structures called **synapses**.

The *connection* between two neurons takes place at these synapses.

- **Terminal Arborisation** /ˌärb(ə)rəˈzāSHən/***:*** When axon reaches its final destination it branches again called **Terminal Arborisation.**

How does neuron work

- **Dendrites** receive input through the **synapses** of other neurons.
- The **soma** processes these *incoming signals* over time and converts that *processed value* into an **output**, which is sent out to other neurons through the **axon** and the **synapses**.
- The flow of electric signals through neurons as shown in the next slide.



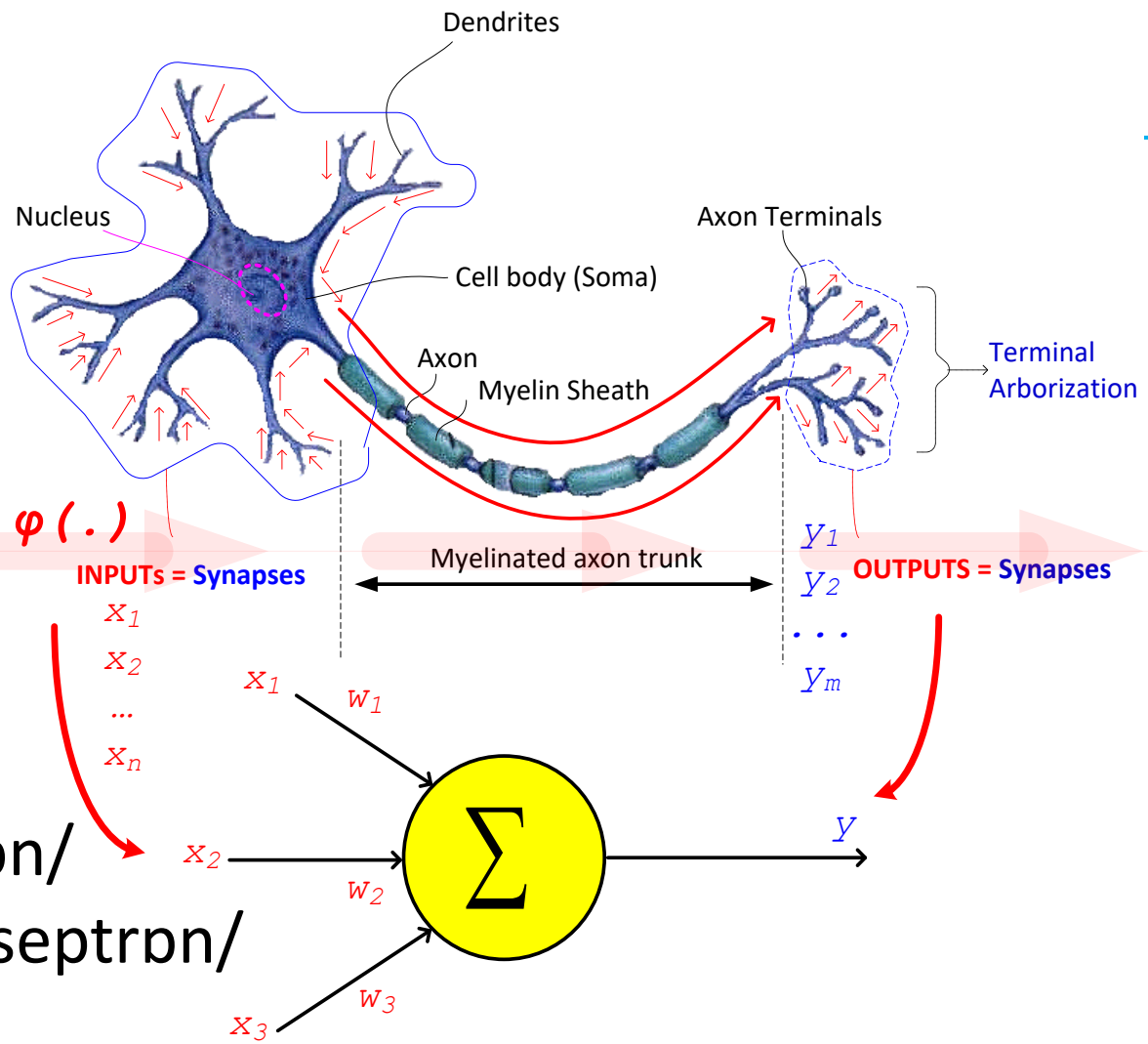
Chapter Contents

What is Artificial Neural Network

- Artificial Neural Networks (**ANN**) is an information processing paradigm that is **inspired** by the way the **biological nervous system** such as brain process information.
- It is composed of large number of highly interconnected processing elements (**neurons**) working in unison to solve a specific problem.
- ANN is considered **"Universal Function Approximators".** It means they can learn and compute any function at all.

Mathematical Model

1. Inputs x_i
2. Synaptic weights w_i
3. Bias b
4. Summing junction v
5. Activation Function $\varphi(.)$
6. Outputs y_i

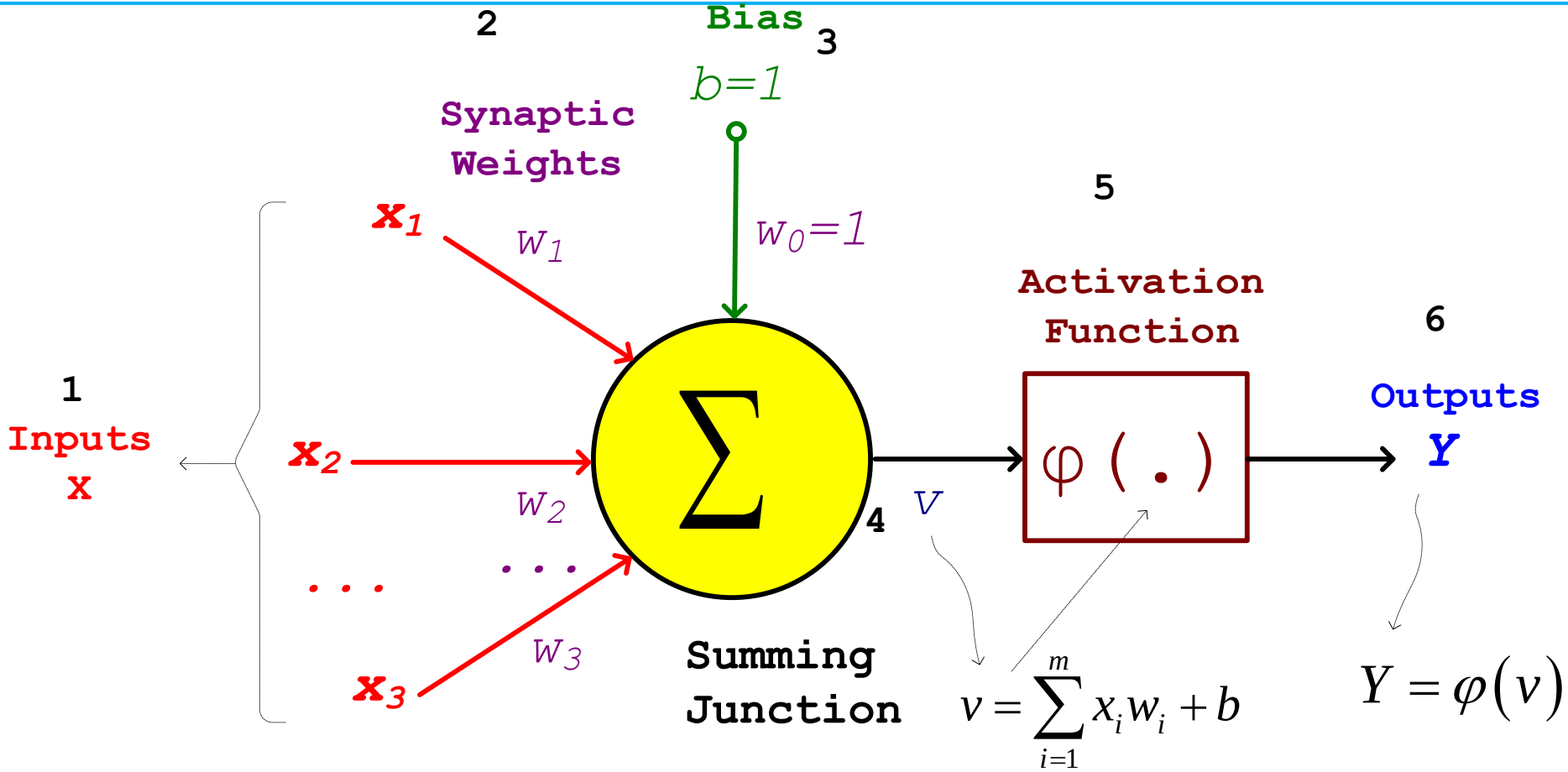


nodes

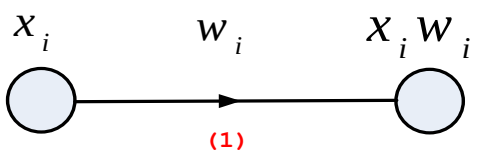
From Neurons /ˈnjʊərɒn/

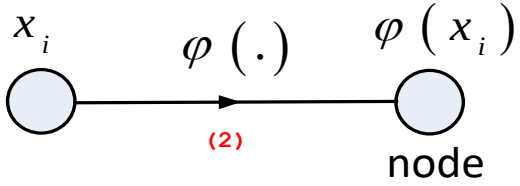
to Perceptron /pəˈseptrɒn/

Mathematical Model

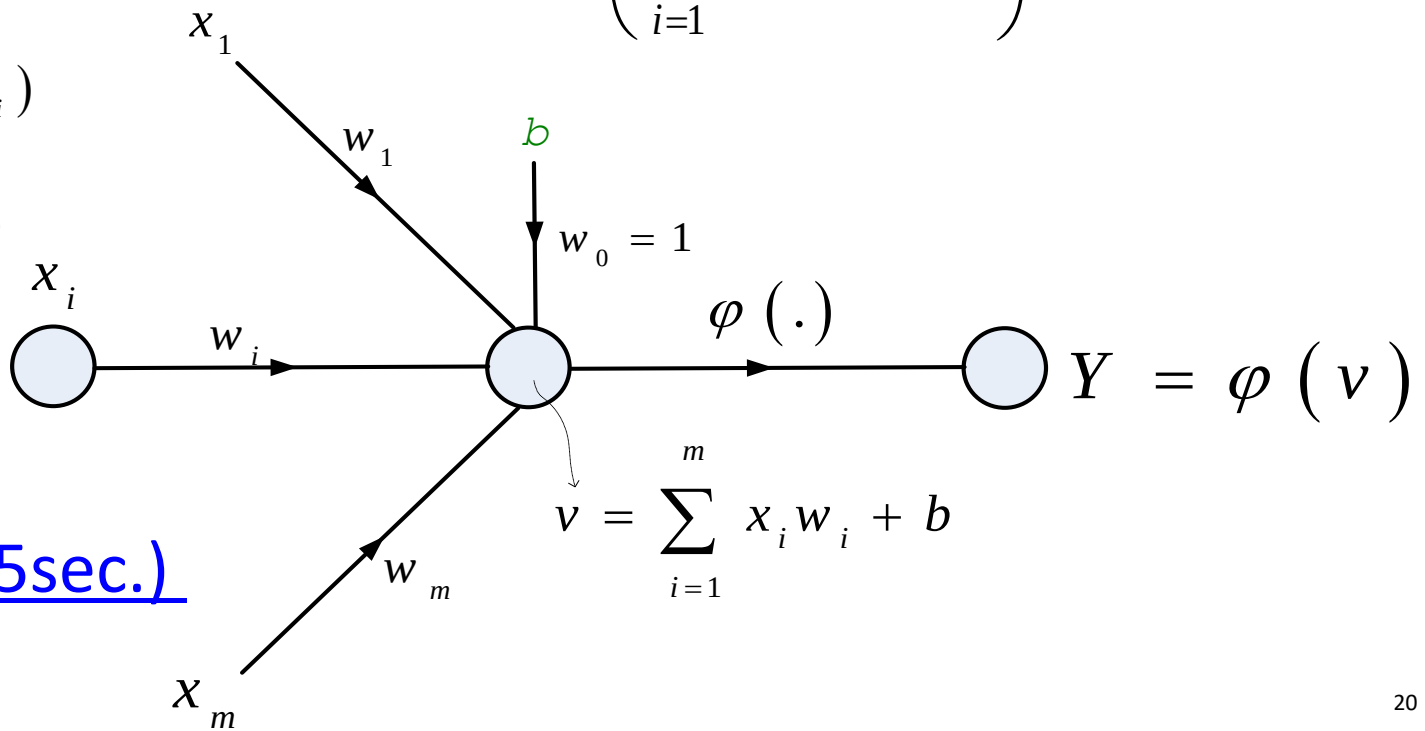


Mathematical Model - Perceptron /pəˈseptrɒn/





$$Y = \varphi\left(\sum_{i=1}^{m} x_i w_i + b w_0\right)$$



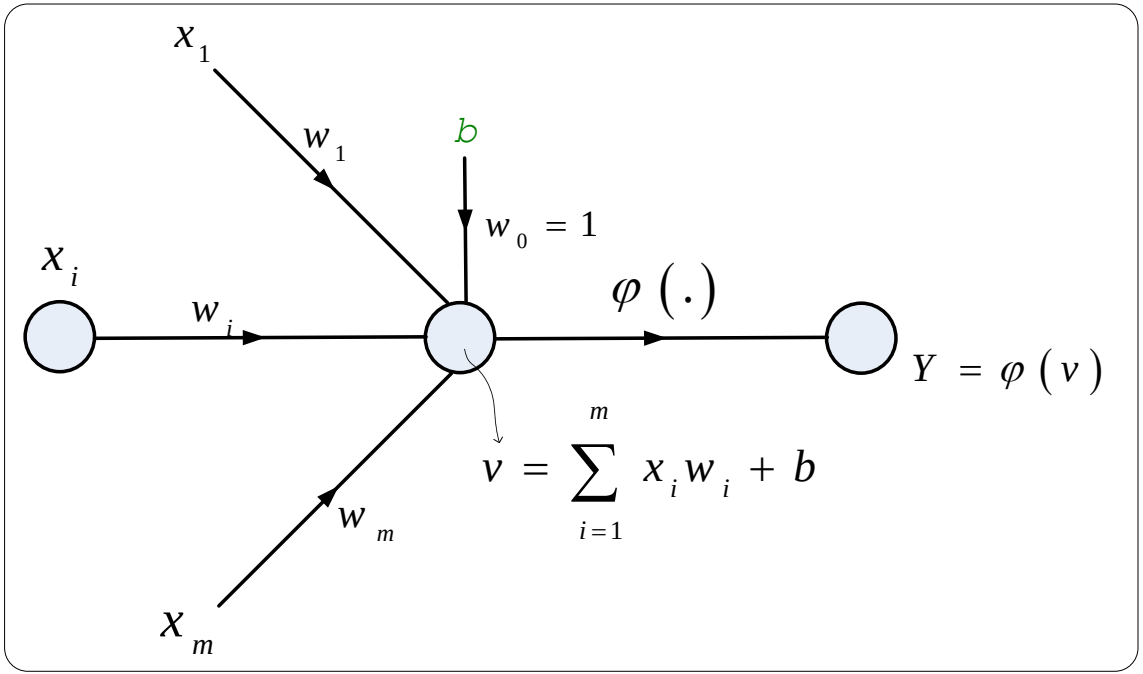
[Perceptron (45sec.)]

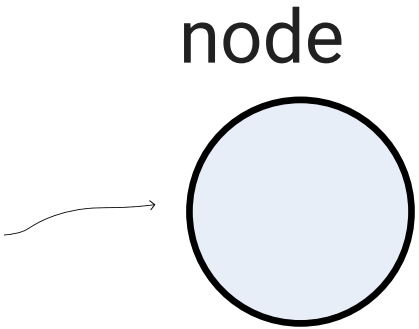
Perceptron /pəˈseptrɒn/

- A mathematical model represents the descriptions of the three functions: **weighting**, **summation** and **activation**;
- one single observation, x0, x1, x2, x3...x(n) represents various **inputs**(independent variables) to the network. Each of these inputs is multiplied by a connection **weight** or **synapse**;
- The weights are represented as w0, w1, w2, w3….w(n). **Weight** shows the **strength** of a particular node.
- b is a **bias** value. A bias value allows you to **shift** the activation function up or down.

Perceptron /pəˈseptrɒn/

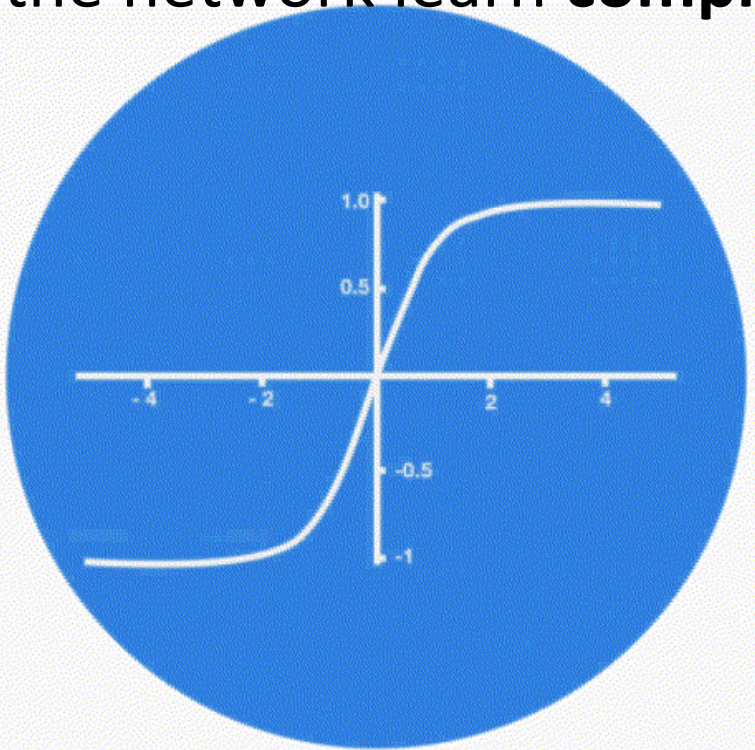
- A **single layer** neural network is called a **Perceptron**. It gives a **single** output;





Activation Functions

- An **activation function** is a **function** that is added to an **ANN** in order to help the network learn **complex patterns in the data**.



0.99
0.009
-0.46

Activation Functions[5]

- The Activation function is important for an ANN to **learn** and **make sense** of something really complicated.
- Their main **purpose** is to convert an input signal of a node in an ANN to an output signal. This output signal is used as input to the next layer in the stack.
- Activation function decides whether a neuron should be activated or not by calculating the weighted sum and further adding bias to it.
- The motive is to introduce non-linearity into the output of a neuron.

Types of Activation Functions

1 Threshold Function(Binary step function)

2 Signum Function

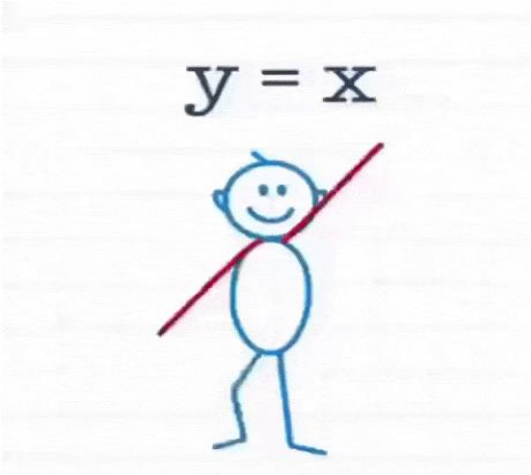
3 Piecewise Linear Function (Saturation function)

4 Sigmoid Function (Logistic function)

5 Hyperbolic Tangent Function (Tanh)

6 Rectified Linear Unit (ReLU) Function

7 Leaky ReLU Function



Types of Activation Functions

8 Exponential Linear Units (ELU) function

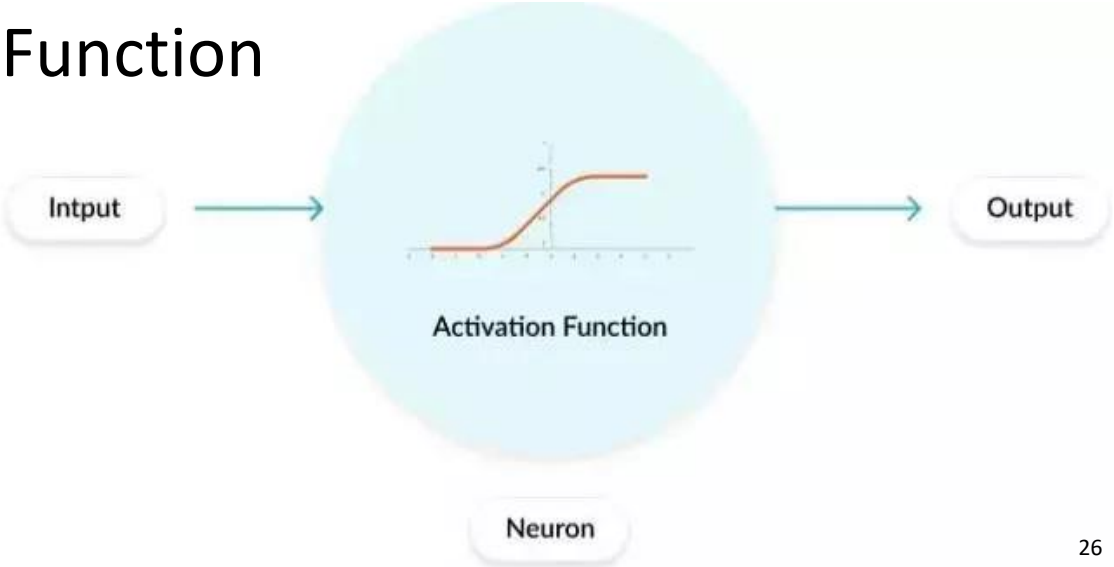
9 Parametric ReLU (PRelu)

10 Softmax

11 Swish (A Self-Gated) Function

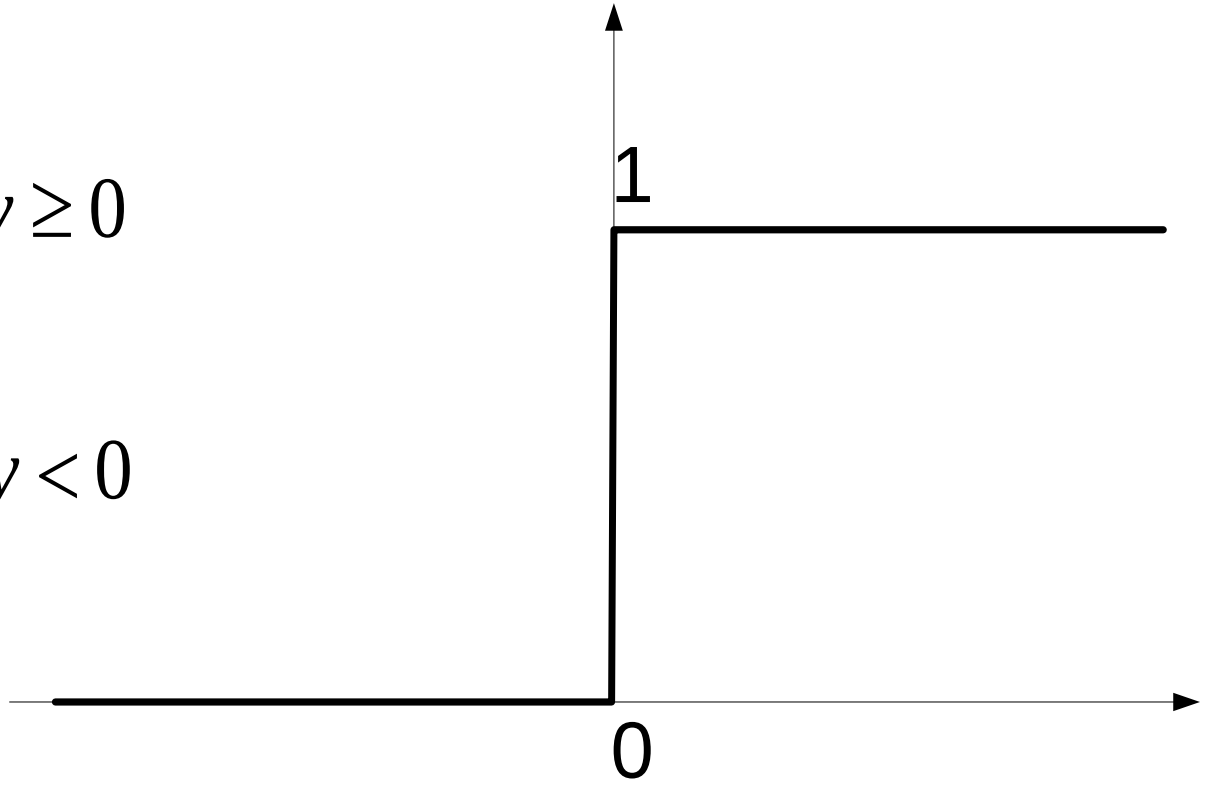
12 Maxout

13 Softplus



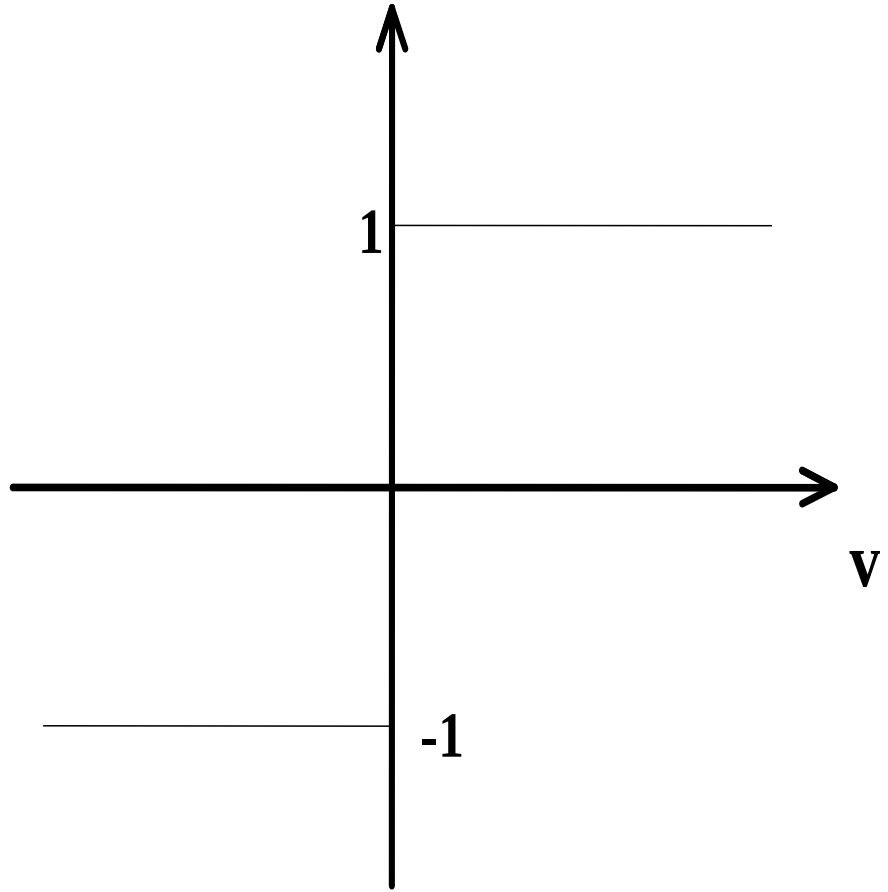
1 Threshold Function (Binary step function)

$$\varphi(v) = \begin{cases} 1, v \geq 0 \\ 0, v < 0 \end{cases}$$



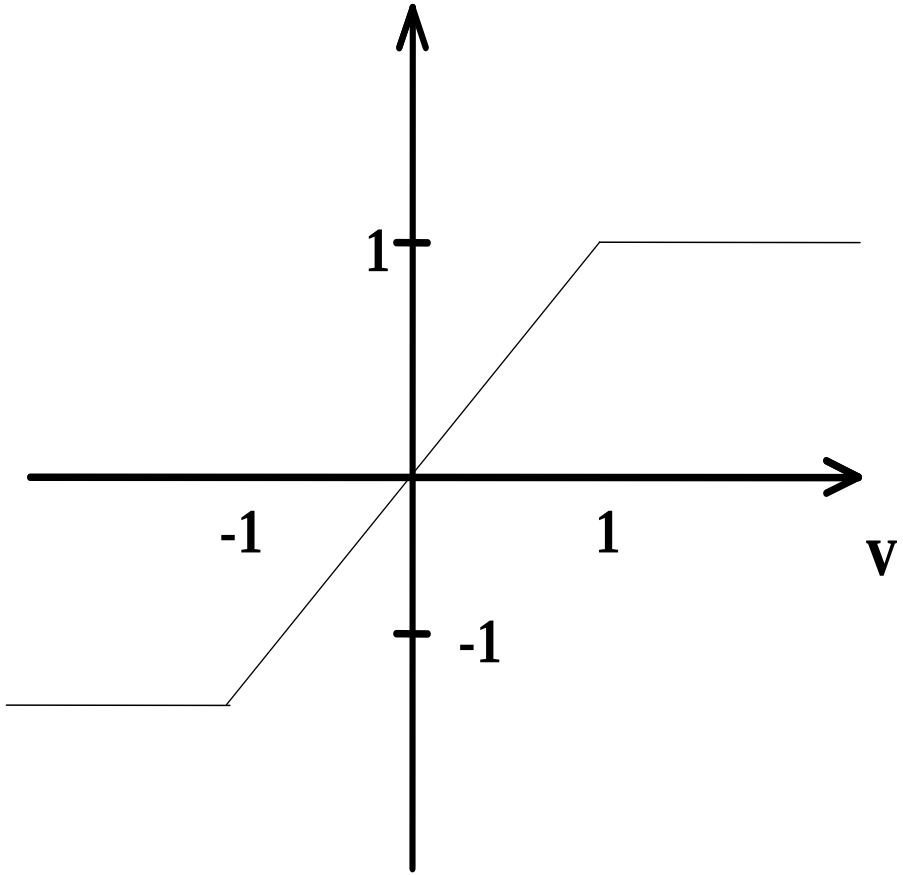
2 Signum Function

$$\varphi(v) = \begin{cases} 1, v > 0 \\ 0, v = 0 \\ -1, v < 0 \end{cases}$$



3 Piecewise Linear Function (Saturation function)[6]

$$\varphi(v)=\begin{cases}1, v\geq 1\\ v, -1<v<1\\ -1, v\leq -1\end{cases}$$

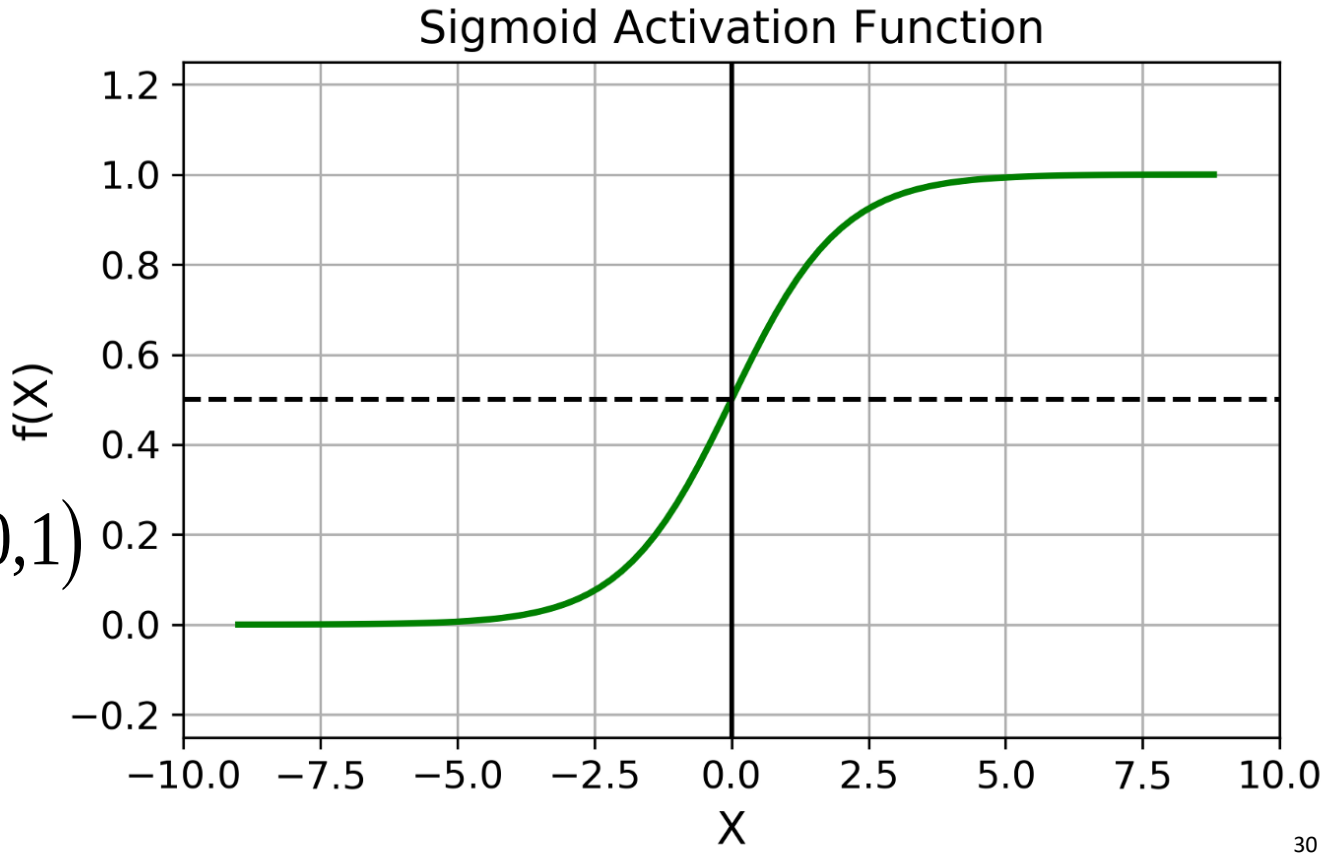


4 Sigmoid Function (Logistic function)

A smooth, **S-shaped** function that maps any input to a value between **0** and **1**

$$\varphi(v) = \frac{1}{1+e^{-av}}, \in (0,1)$$

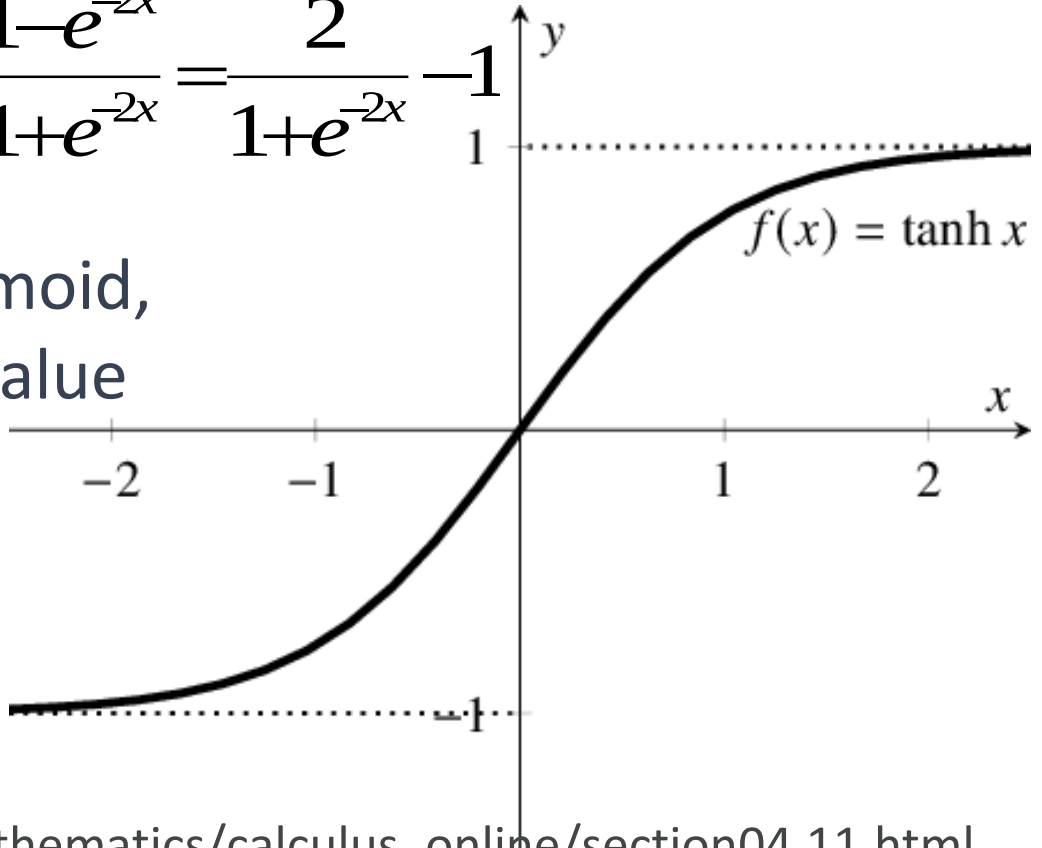
a is called the slope parameter



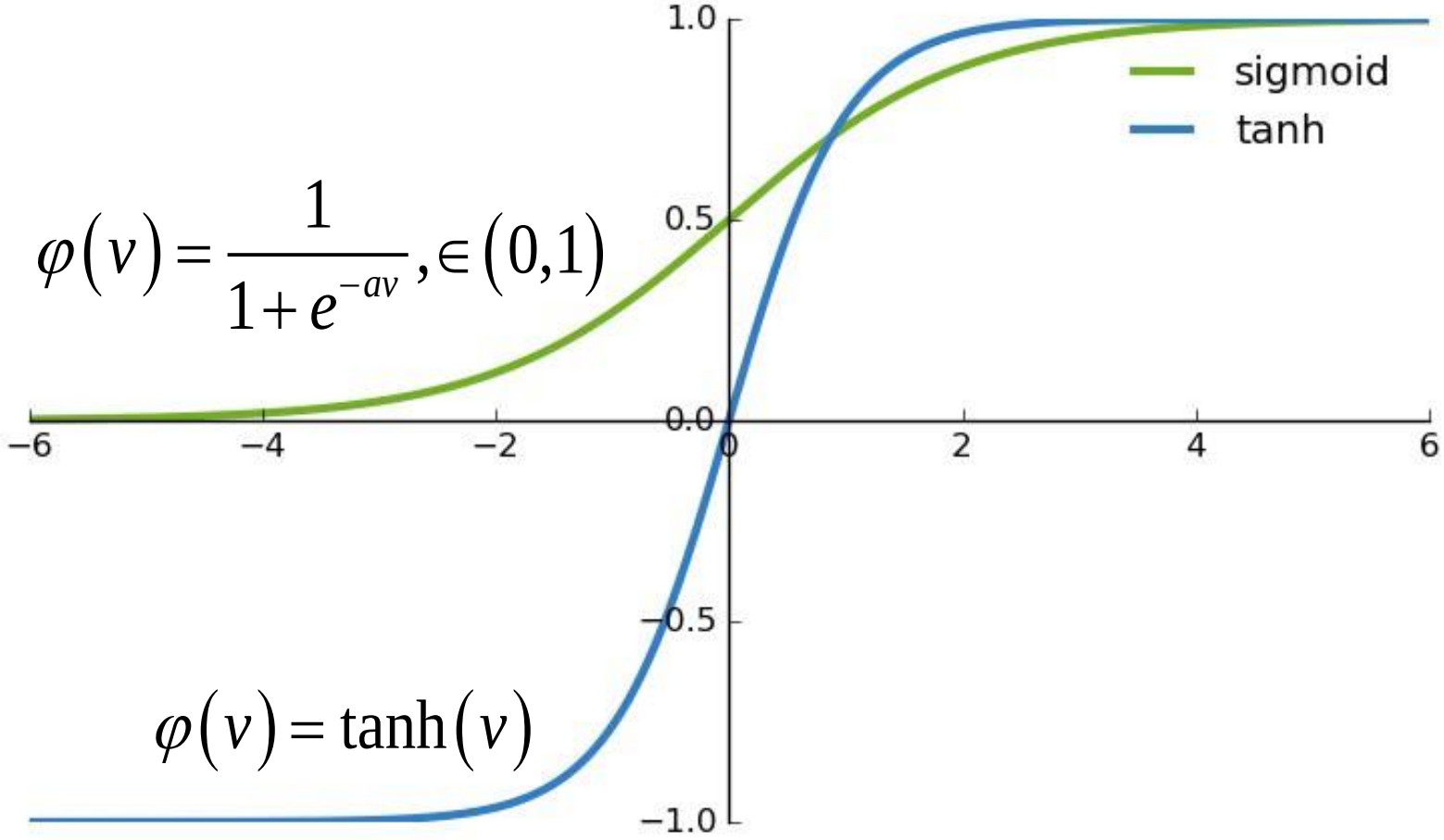
5 Hyperbolic Tangent Function (Tanh)

$$\varphi(x) = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}} = \frac{1 - e^{-2x}}{1 + e^{-2x}} = \frac{2}{1 + e^{-2x}} - 1$$

A function similar to sigmoid, but it maps inputs to a value between **-1** and **1.**

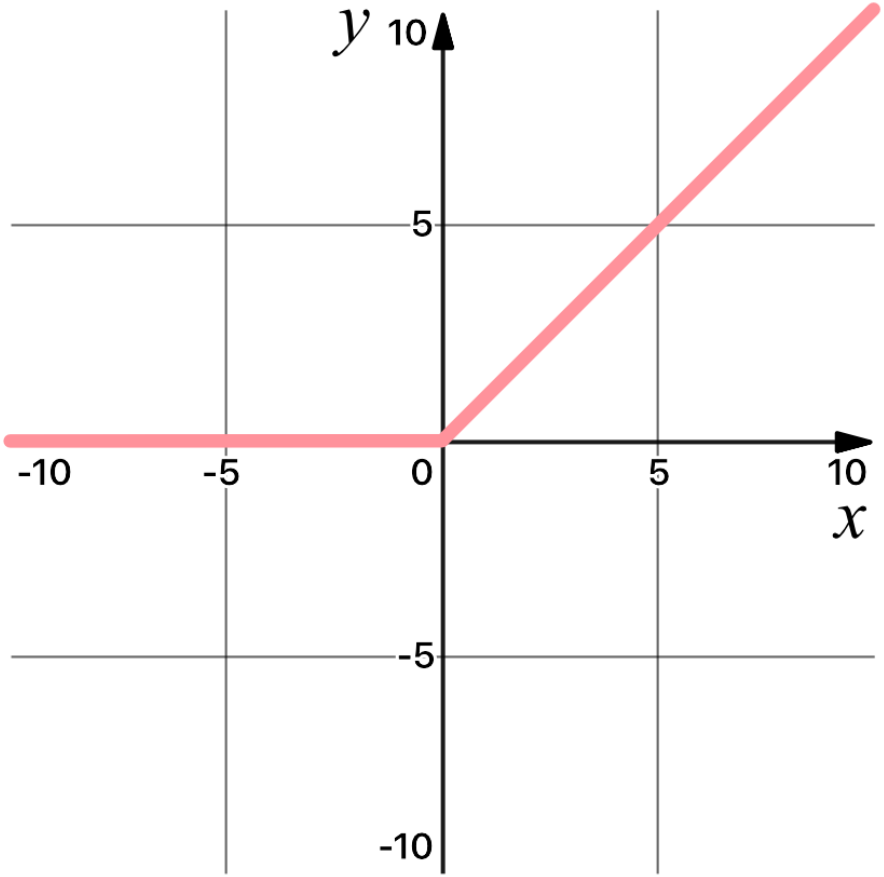


https://www.whitman.edu/mathematics/calculus_online/section04.11.html


$\varphi(v) = \frac{1}{1+e^{-av}}, \in (0,1)$
$\varphi(v) = \tanh(v)$
sigmoid
tanh
1.0
0.5
0.0
−0.5
−1.0
−6
−4
−2
0
2
4
6

6 Rectified Linear Unit (ReLU) Function

$$\mathrm{Re}LU(x) = \begin{cases} \max(0, x), x \geq 0 \\ 0, x < 0 \end{cases}$$

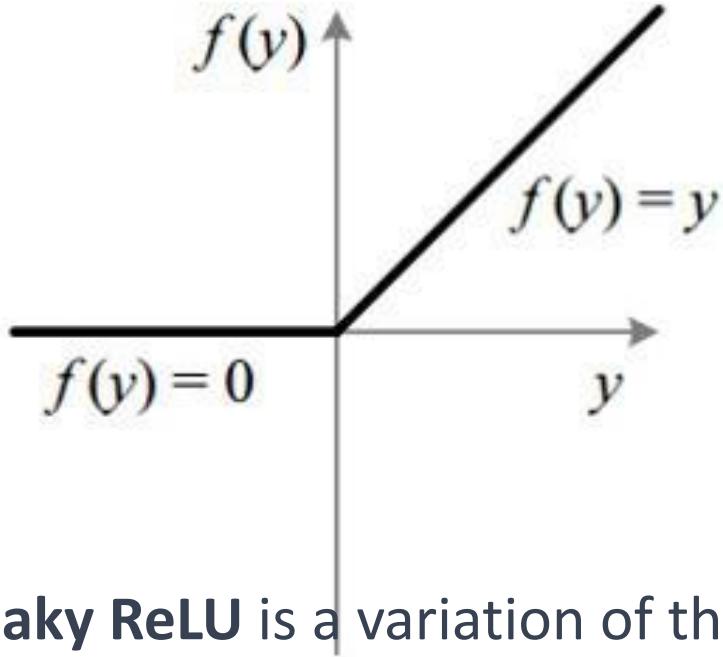


A simple function that returns the input if it's positive, and zero otherwise.

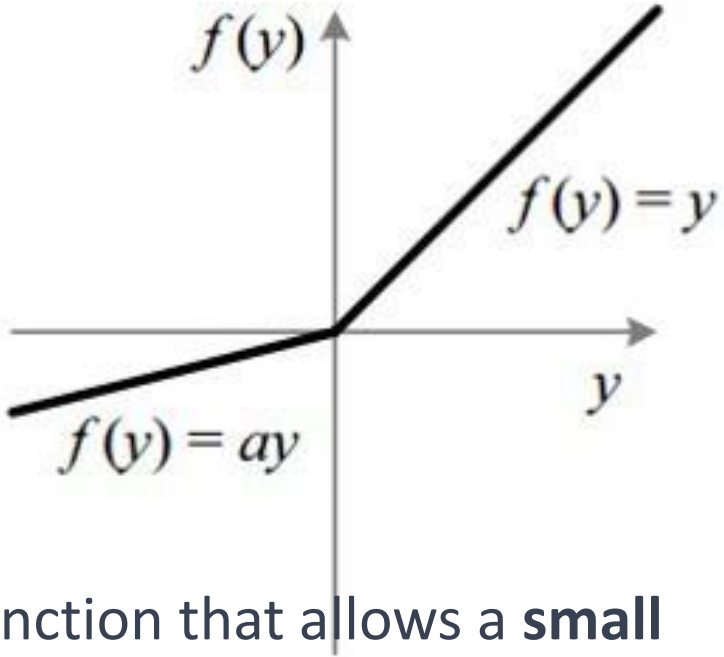
7 Leaky ReLU Function

$$\varphi(v) = \begin{cases} \max(0, x), x \geq 0 \\ ax, x < 0 \end{cases}$$

ReLU



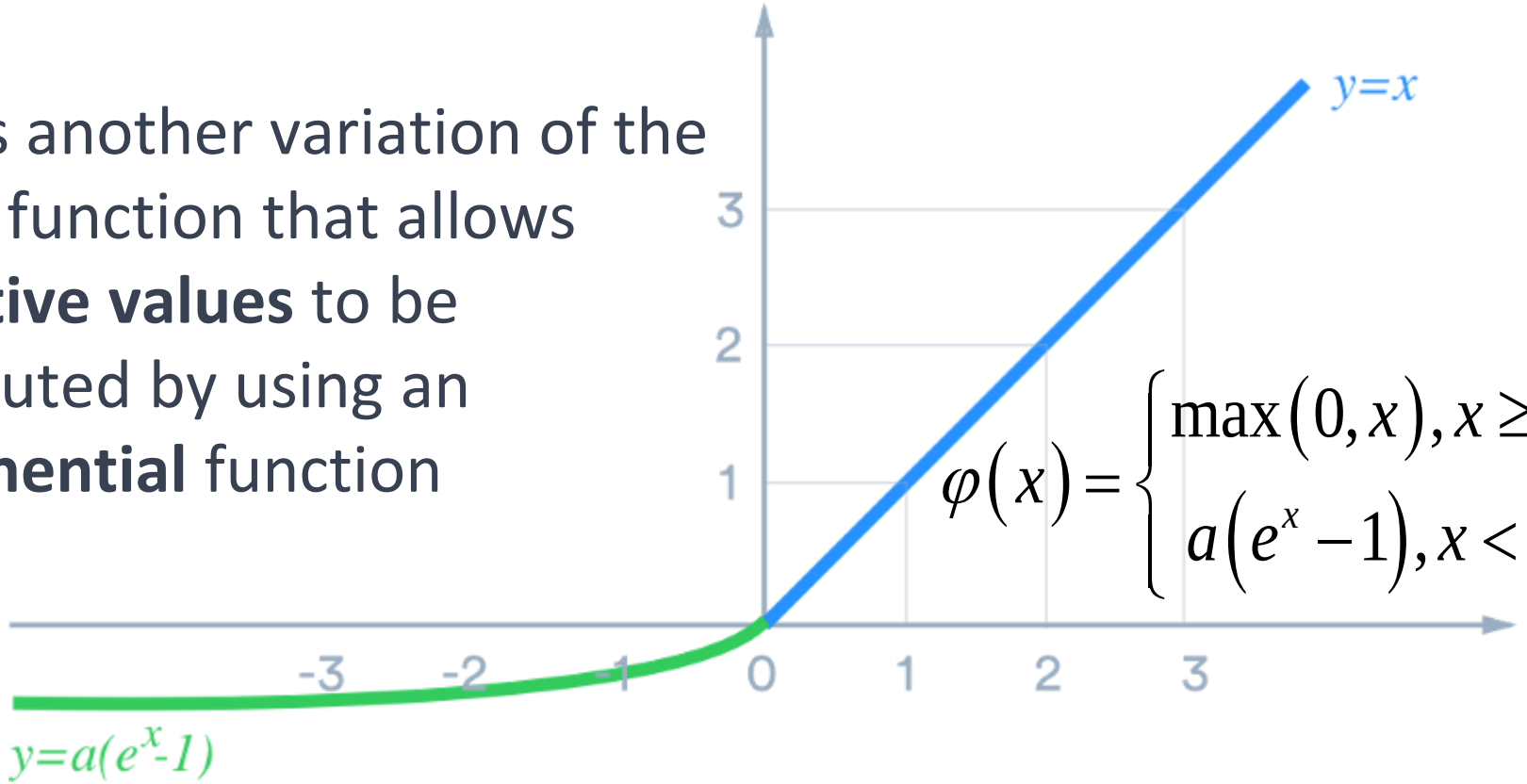
Leaky ReLU



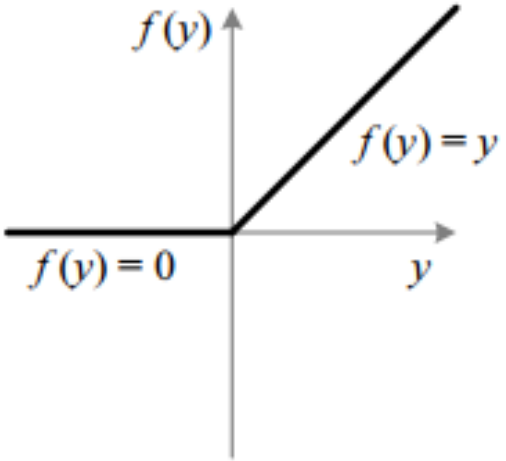
Leaky ReLU is a variation of the ReLU function that allows a **small negative** slope for negative input values

8 Exponential Linear Units (ELU) function [11]

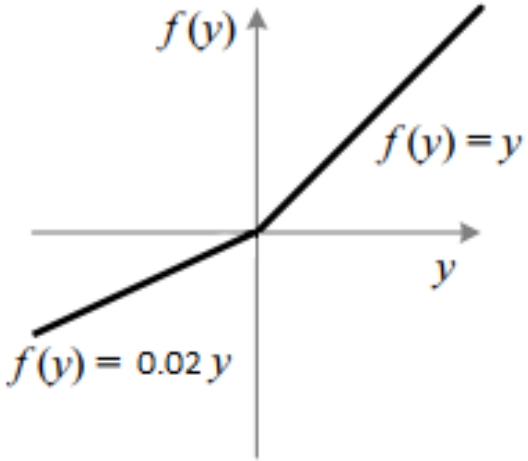
ELU is another variation of the **ReLU** function that allows **negative values** to be computed by using an **exponential** function



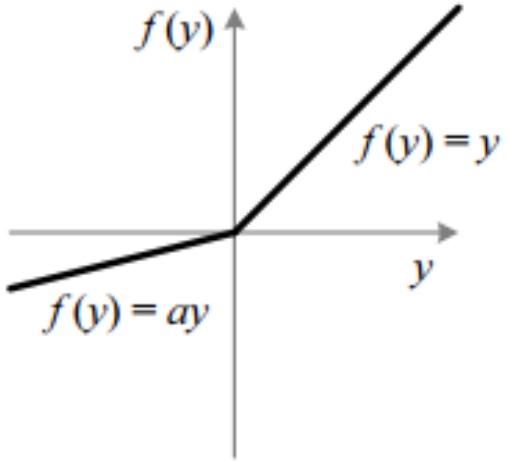
9 Parametric ReLU (PRelu)[12]



ReLU



LeakyReLU

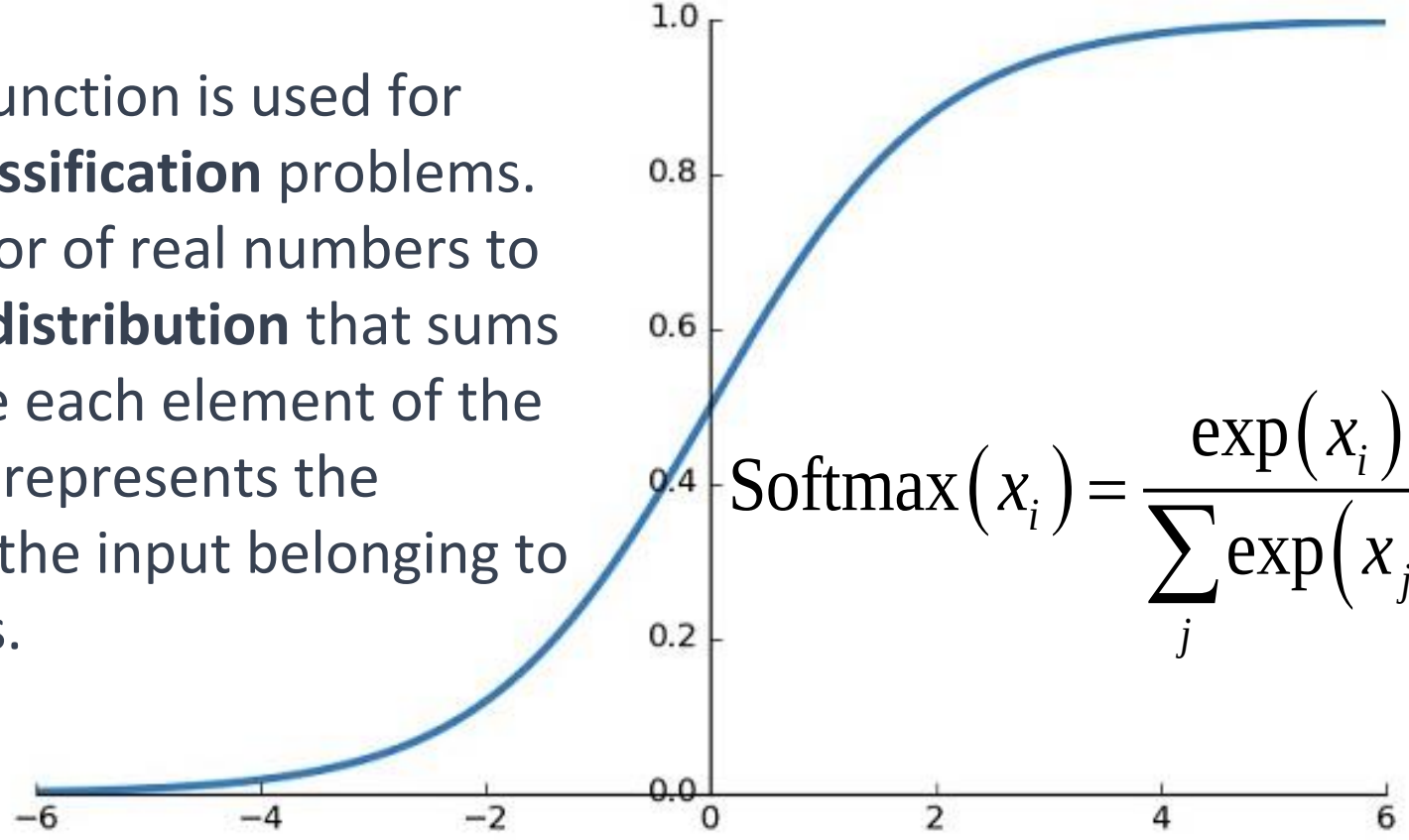


PReLU

$$f(y_i) = \begin{cases} y_i, & \text{if } y_i > 0 \\ a_i y_i, & \text{if } y_i \leq 0 \end{cases}$$

10 Softmax Function

- The softmax function is used for multi-class **classification** problems.
- It maps a vector of real numbers to a **probability distribution** that sums up to **1**, where each element of the output vector represents the probability of the input belonging to a specific class.



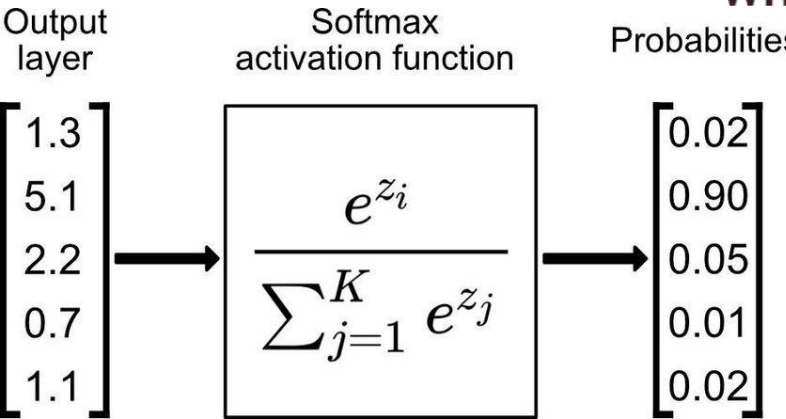
$$\text{Softmax}(x_i) = \frac{\exp(x_i)}{\sum_j \exp(x_j)}$$

https://deepai.org/machine-learning-glossary-and-terms/softmax-layer

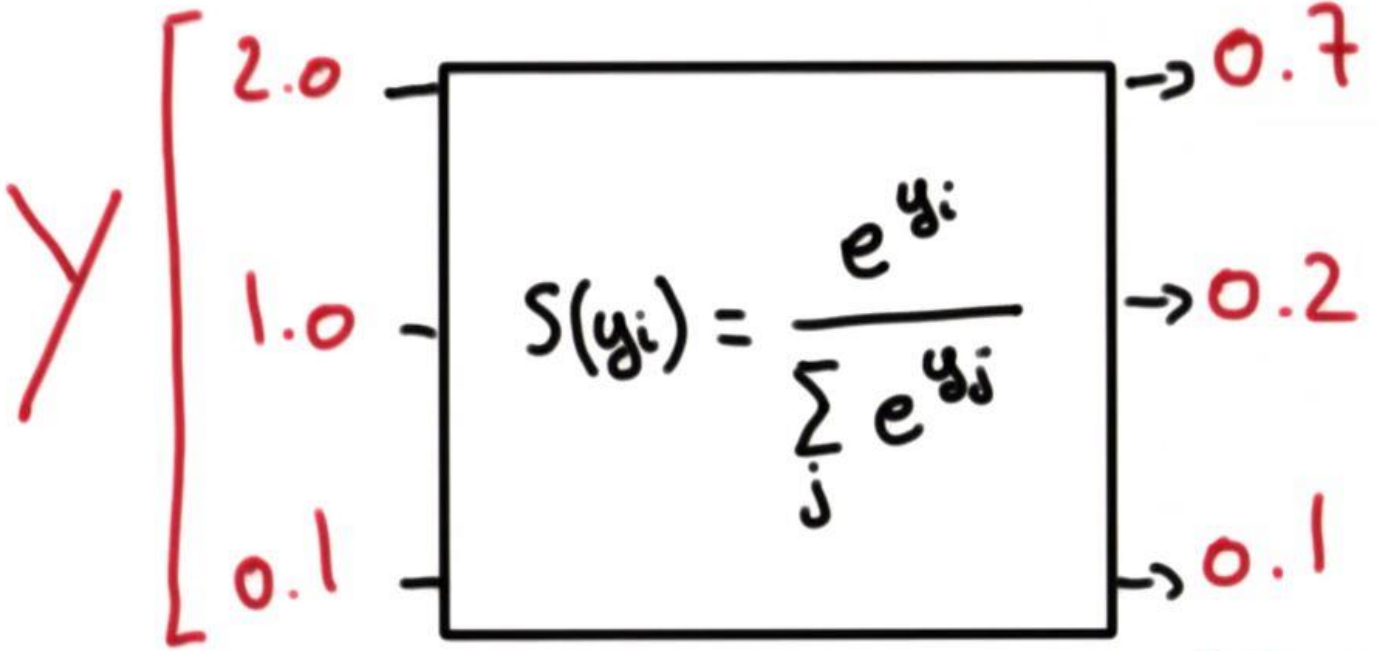
Softmax function

$$P(y=j \mid \Theta^{(i)}) = \frac{e^{\Theta^{(i)}}}{\sum_{j=0}^{k} e^{\Theta_k^{(i)}}}$$

where $\Theta = w_0x_0 + w_1x_1 + \ldots + w_kx_k = \sum_{i=0}^{k} w_ix_i = w^Tx$



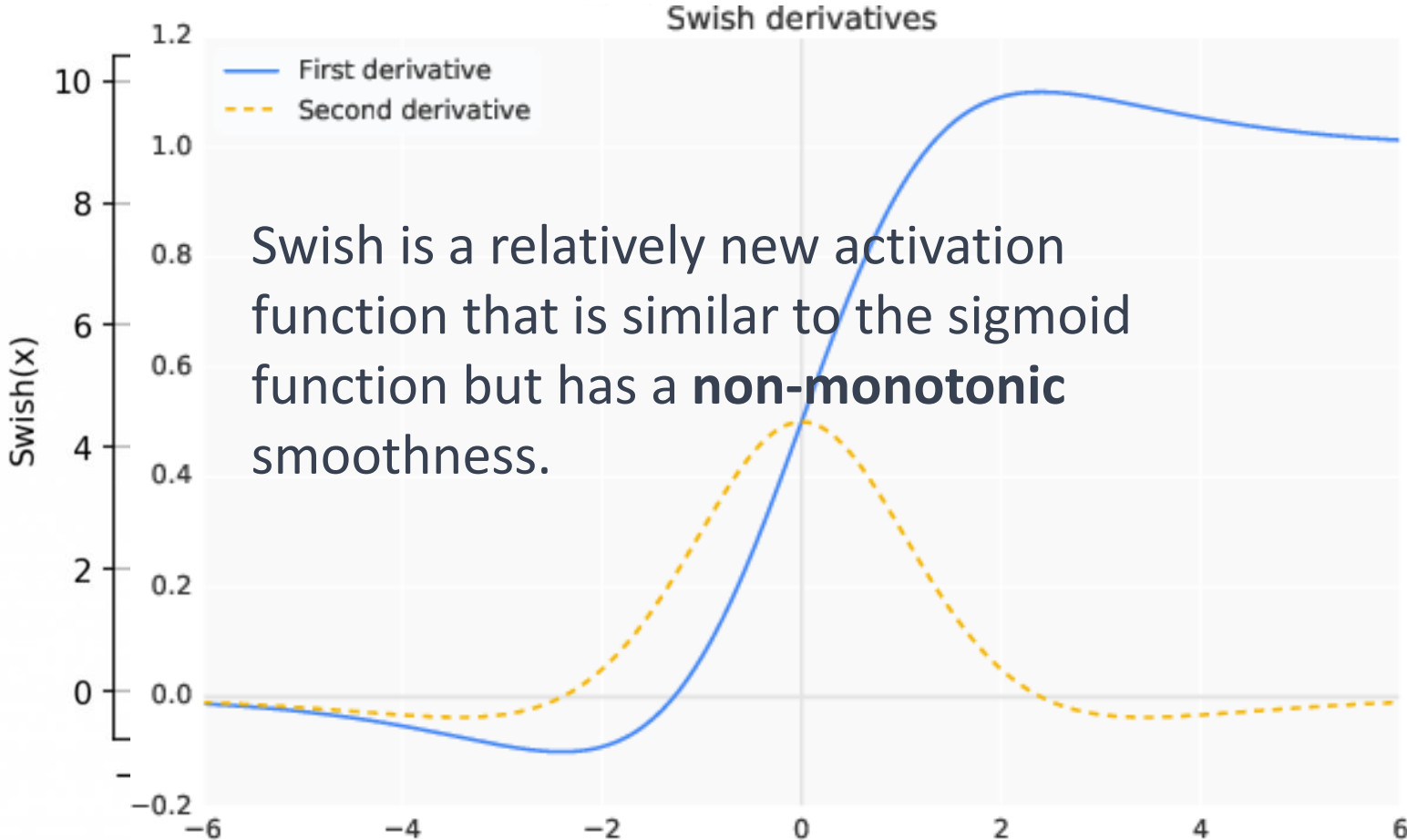
SOFTMAX



'LOGITS'

SCORES → PROBABILITIES

11 Swish (A Self-Gated) Function[13]



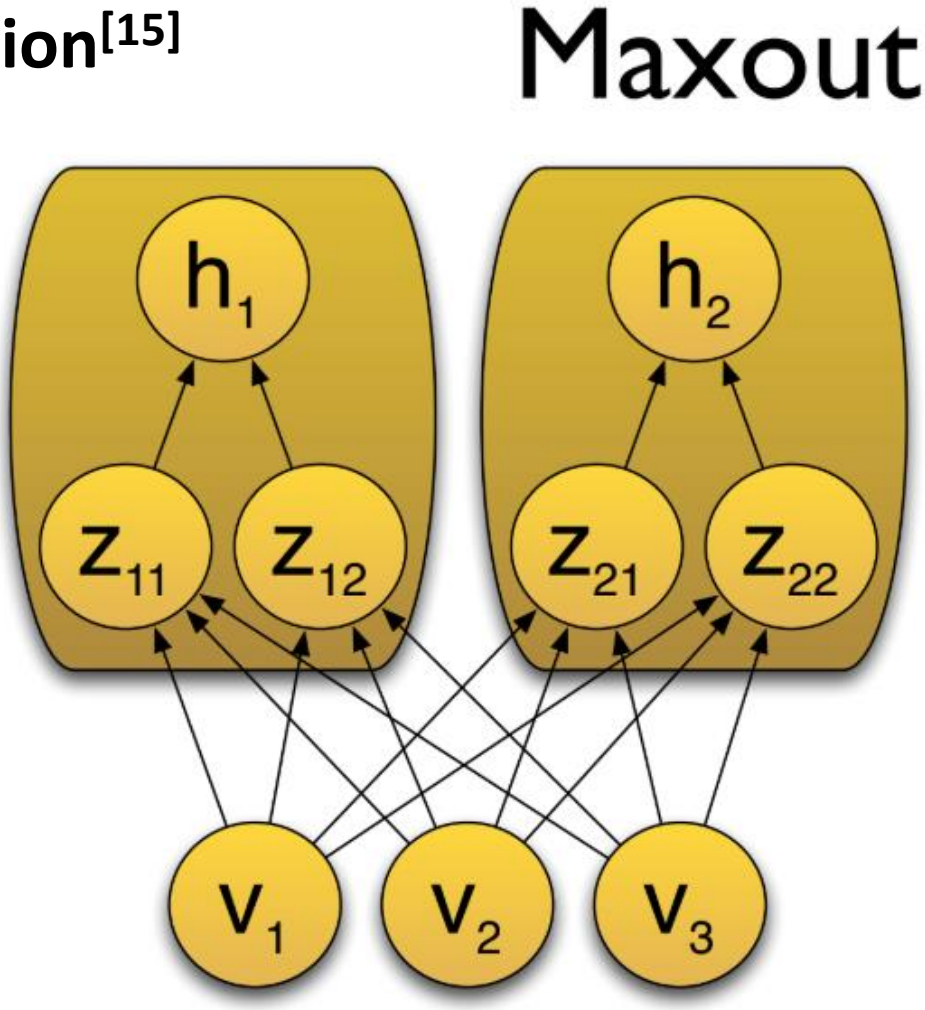
Swish is a relatively new activation function that is similar to the sigmoid function but has a **non-monotonic** smoothness.

11 Swish (A Self-Gated) Function[13]

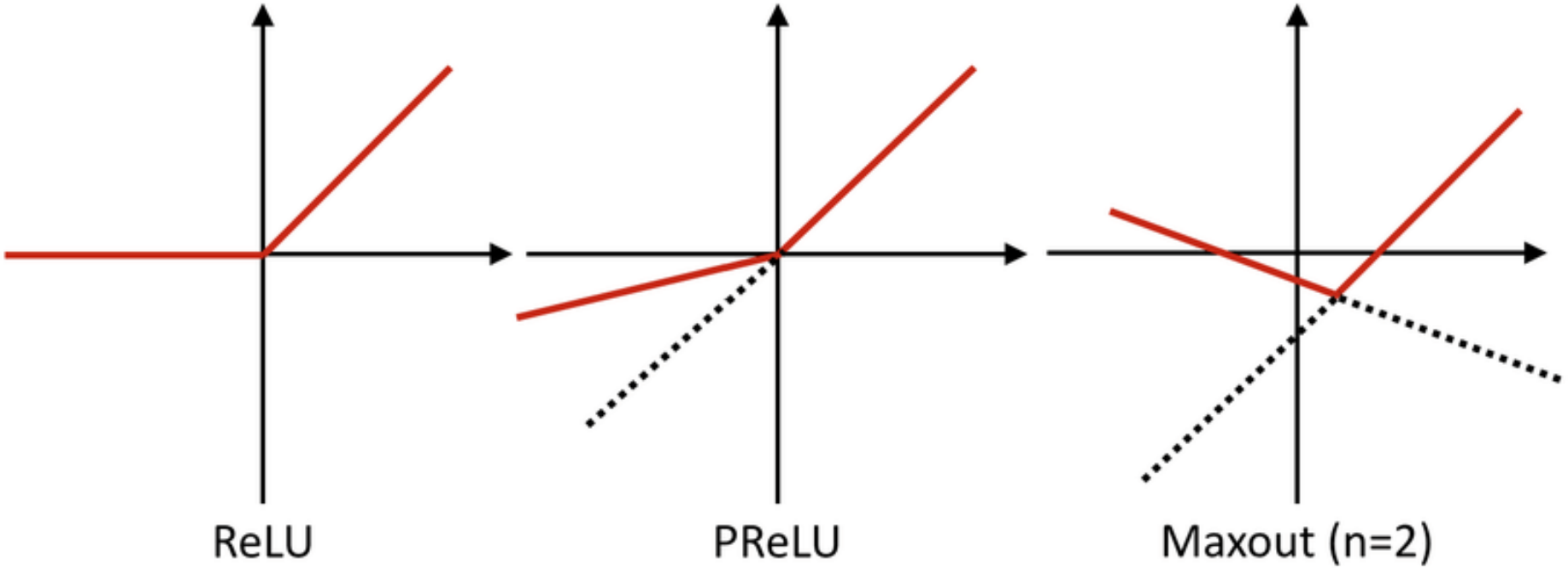
- It **combines** the best aspects of the **ReLU** and **sigmoid** functions, making it efficient to compute while also avoiding problems like vanishing gradients and dead neurons.
- Swish involves a **gating mechanism** that determines the output of the function based on the input.
- The Swish function is defined as **swish(x) = x * sigmoid(beta * x),** where **beta** is a hyperparameter that controls the shape of the function.
- Swish has been shown to perform well on a variety of deep learning tasks, including image classification and language modelling.

12 Maxout Activation Function[15]

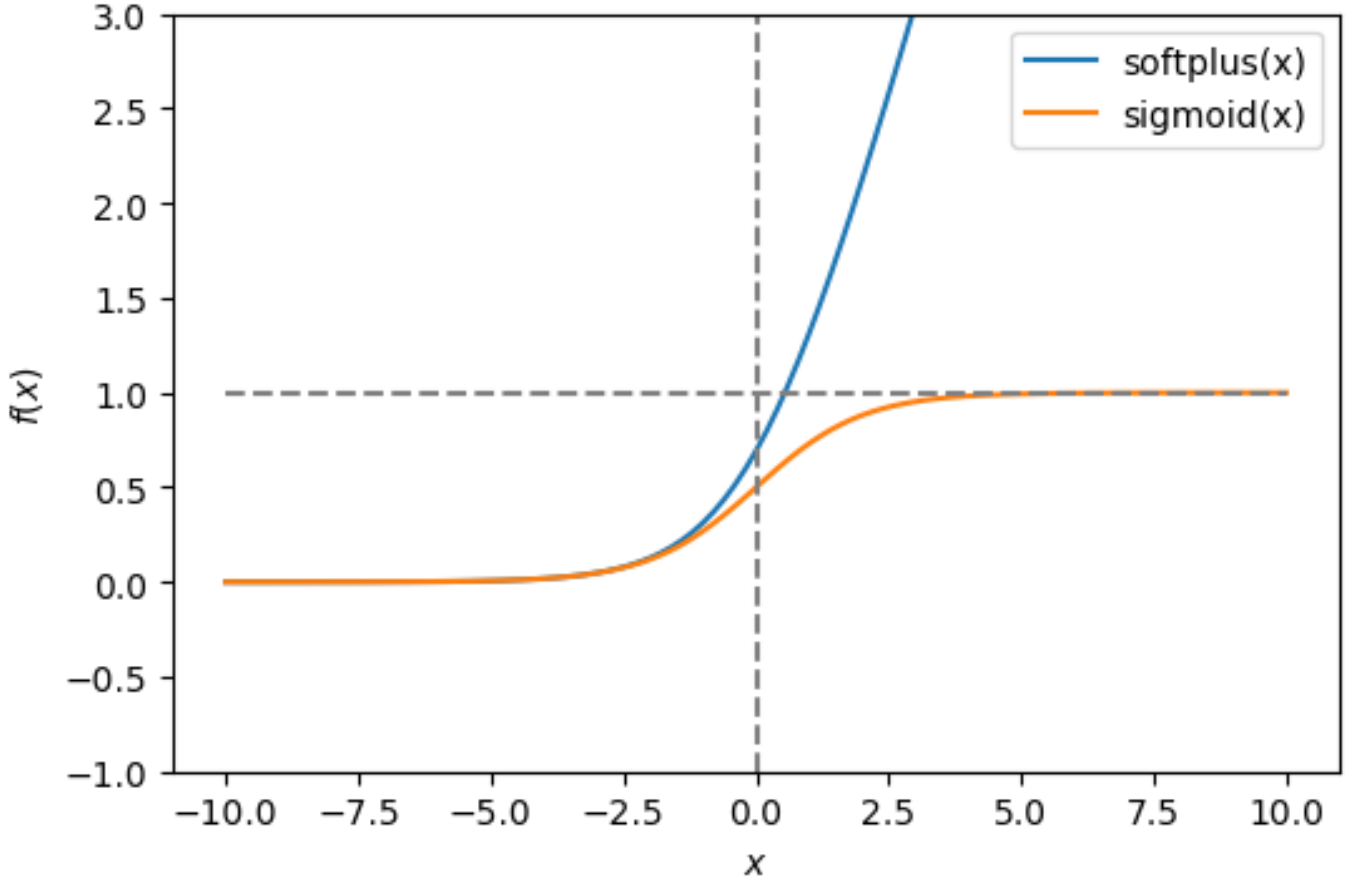
A **Maxout** layer is simply a layer where the activation function **is the max of the inputs.**



12 Maxout Activation Function



13 Softplus Activation Function [16]



13 Softplus Activation Function [16]

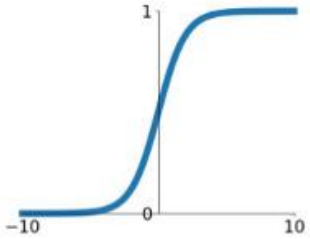
$$softplus(v) = \max(0, x) + \log\left(1 + e^{-|x|}\right)$$

$$soft\min us(v) = \min(0, x) - \log\left(1 + e^{-|x|}\right)$$

$$sigmoid(v) = \frac{1}{2}\left[1 + \tanh\left(\frac{v}{2}\right)\right]$$

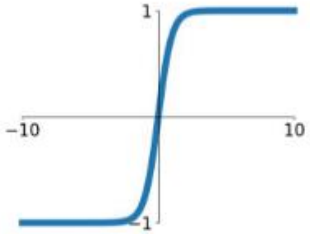
Sigmoid

$$\sigma(x) = \frac{1}{1+e^{-x}}$$



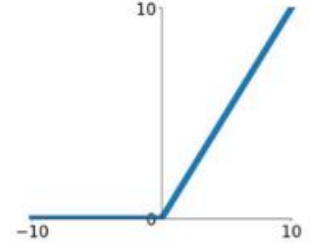
tanh

$$\tanh(x)$$



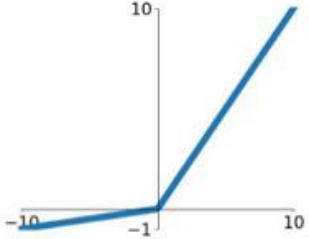
ReLU

$$\max(0, x)$$



Leaky ReLU

$$\max(0.1x, x)$$

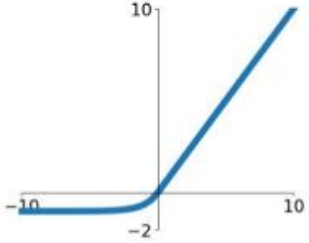


Maxout

$$\max(w_1^T x + b_1, w_2^T x + b_2)$$

ELU

$$\begin{cases} x & x \geq 0 \\ \alpha(e^x - 1) & x < 0 \end{cases}$$



Chapter Contents

Why Use Artificial Neural Network

1. ANN can **learn and model non-linear** and **complicated** interactions, which is critical since many of the relationships between inputs and outputs in **real life are non-linear** and complex.

2. ANNs can **generalise** – After learning from the original inputs and their associations, the model may **infer** unknown relationships from anonymous data, allowing it to generalise and predict unknown data.

Why Use Artificial Neural Network

3. ANN **does not impose** any **constraints** on the input variables, unlike many other prediction approaches (like how they should be distributed).

4. ANNs can better simulate **heteroskedasticity**, or data with **high volatility** and **non-constant variance**, because of their capacity to discover latent correlations in the data without imposing any preset associations. This is particularly helpful in financial time series forecasting (for example, stock prices) when significant data volatility.

Advantages of ANN

1 Attribute-value pairs are used to represent problems in ANN.

2 The output of ANNs can be **discrete-valued, real-valued**, or a **vector** of multiple **real** or **discrete-valued** characteristics, while the **target function can be** discrete-valued, real-valued, or a vector of numerous real or discrete-valued attributes.

Advantages of ANN

3 Noise in the training data is **not** a problem for ANN learning techniques. There may be **mistakes** in the training samples, but they will **not** affect the final result.

4 It's utilised when a quick assessment of the **taught target function** is necessary.

5 The **number of weights** in the network, the number of training instances evaluated, and the settings of different learning algorithm parameters can all contribute to **extended training periods** for ANNs.

Disadvantages of ANN

1. Hardware Dependence
2. Not clear to understand the network's operation
3. Assured network structure by'trial and error'
4. Difficulty in presenting the issue to the network
5. The network's lifetime is unknown

Disadvantages of ANN

1. Hardware Dependence:

- The construction of ANN necessitates the use of **parallel processors**.
- As a result, the equipment's realisation is contingent.

2. Not easy to understand the network's operation:

- This is the most serious issue with ANN.
- When ANN provides a probing answer, it does **not explain** why or how it was chosen.
- As a result, the network's **confidence** is eroded.

Disadvantages of ANN

3. Assured network structure:

- Any precise rule does **not** determine the **structure** of ANN.
- **Experience and trial and error** are used to develop a suitable network structure.

4. Difficulty in presenting the issue to the network:

- ANNs are capable of working with **numerical data**.
- Before being introduced to ANN, problems must **be converted into** numerical values.
- The display method that is chosen will have a **direct impact** on the network's performance. The user's skill is a **factor**.

Disadvantages of ANN

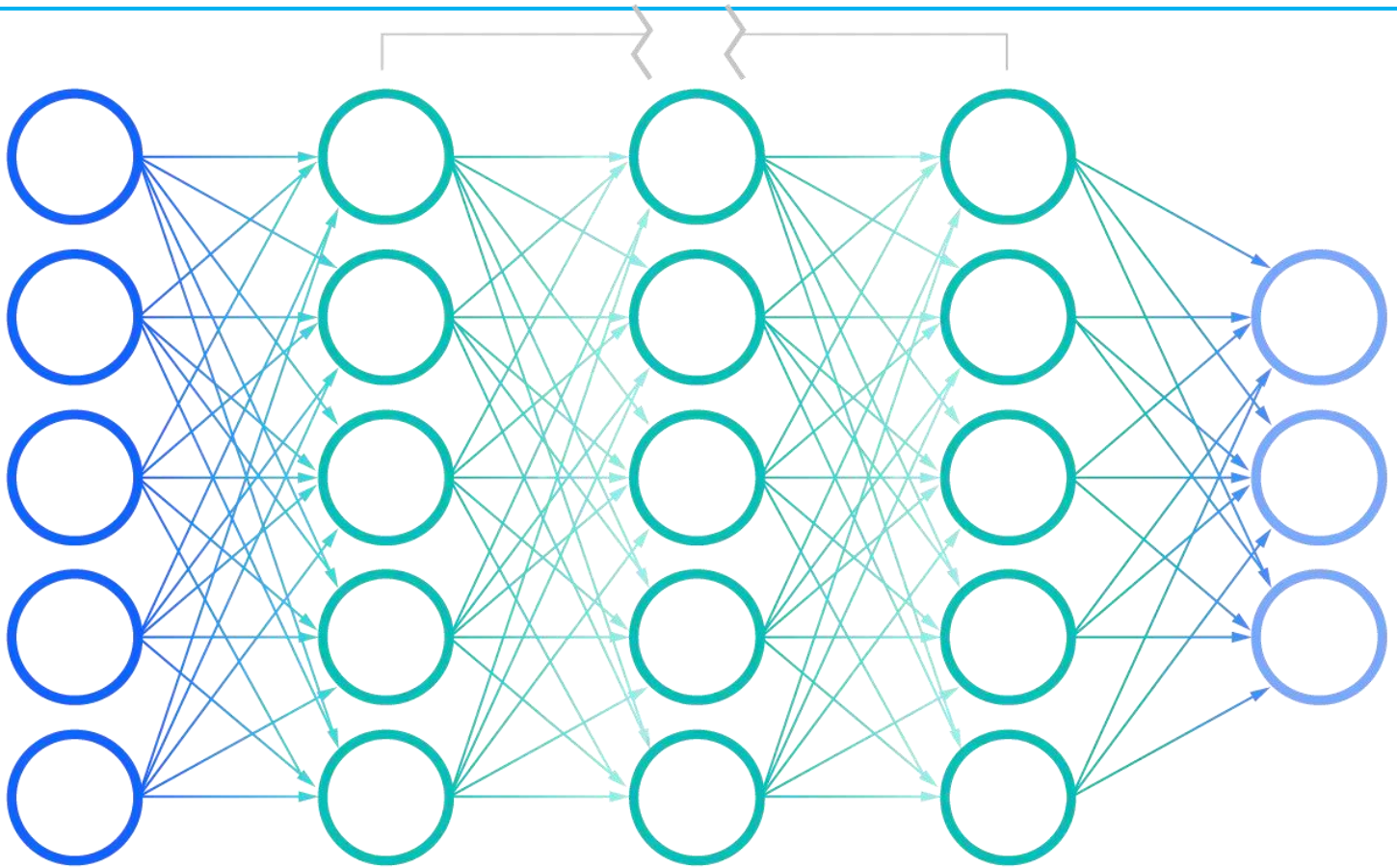
5. The network's lifetime is unknown:

- When the network's **error** on the sample is **decreased** to a specific amount, the training is complete.
- The value does **not** produce the **best** outcomes.

Chapter Contents

Deep neural network

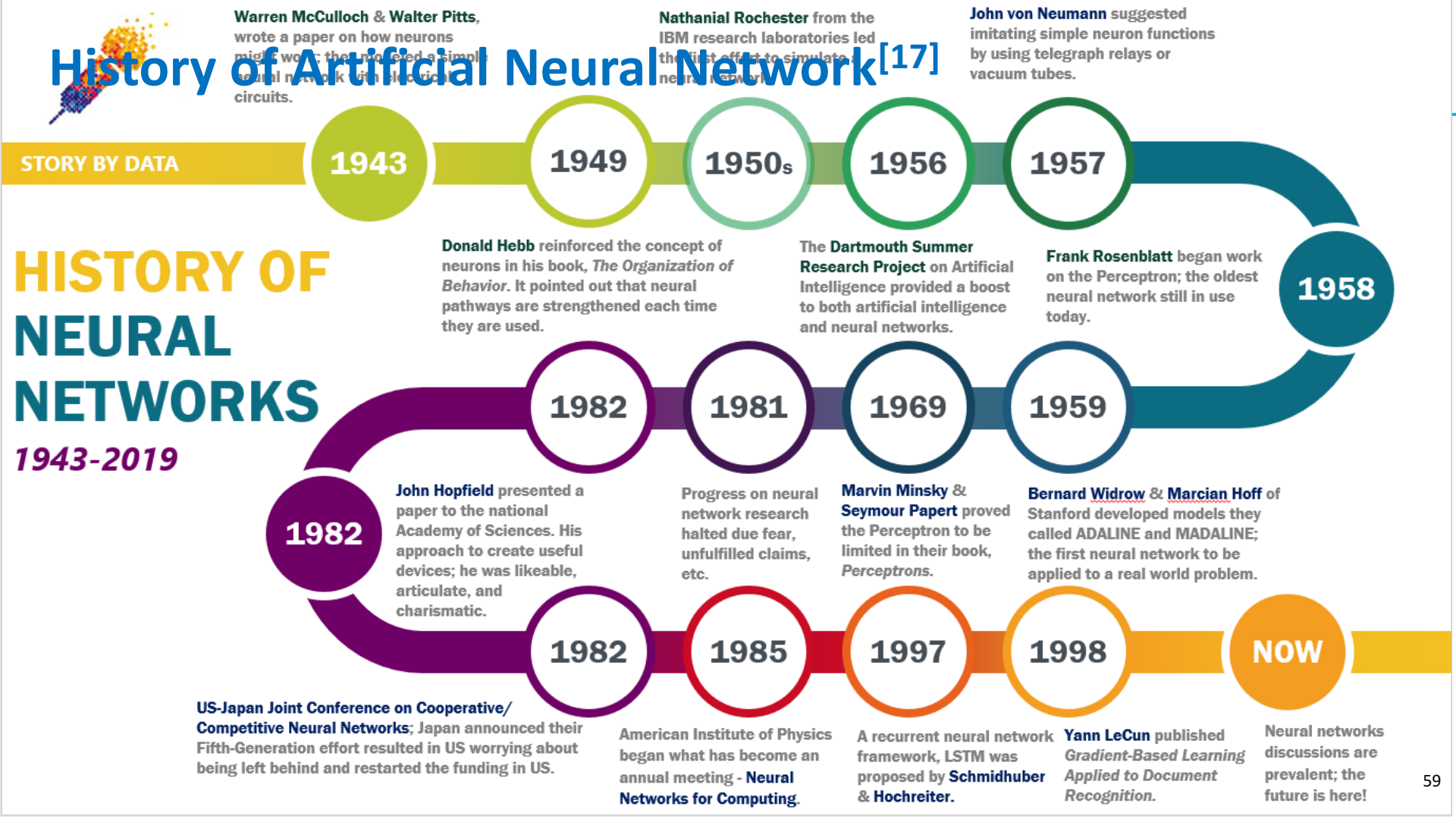
Input layer | Multiple hidden layers | Output layer



Fundamentals of Artificial Neural Network [4]

- History of Artificial Neural Network
- Artificial Neural Networks Architecture
- How Do ANNs Work
- Types of ANN
- Three Steps in ANN Design
- Learning Paradigms
- Learning Algorithms
- Knowledge Representation
- Loss Function and Cost Function

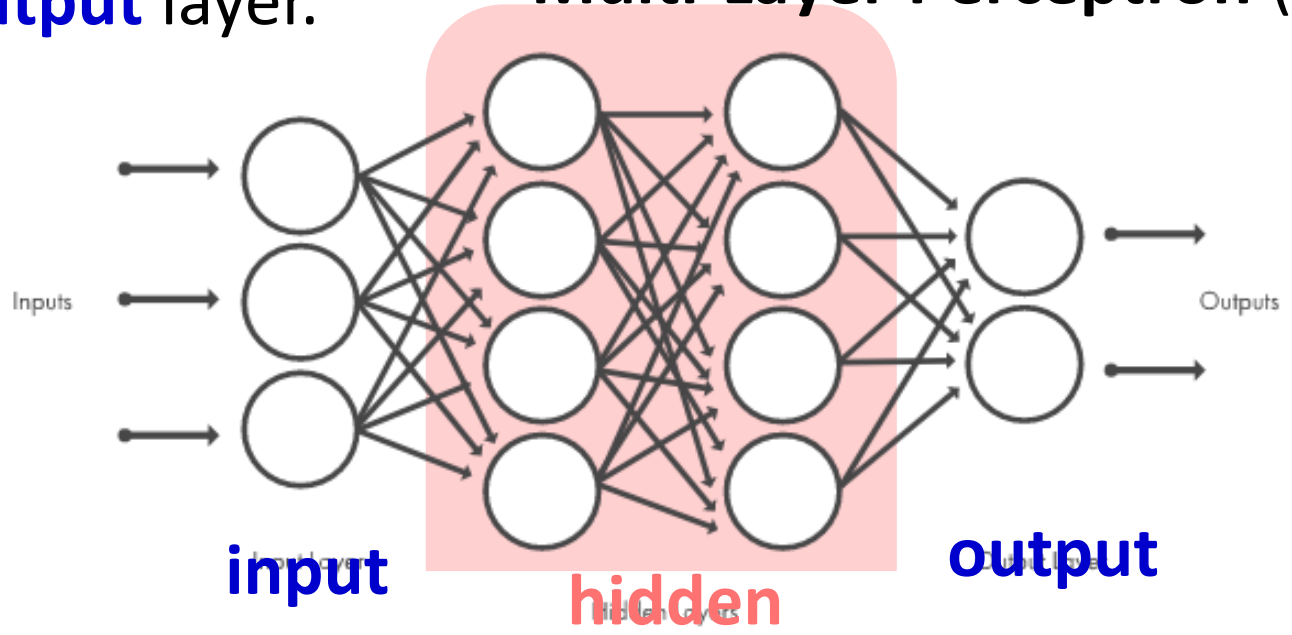
History of Artificial Neural Network[17]



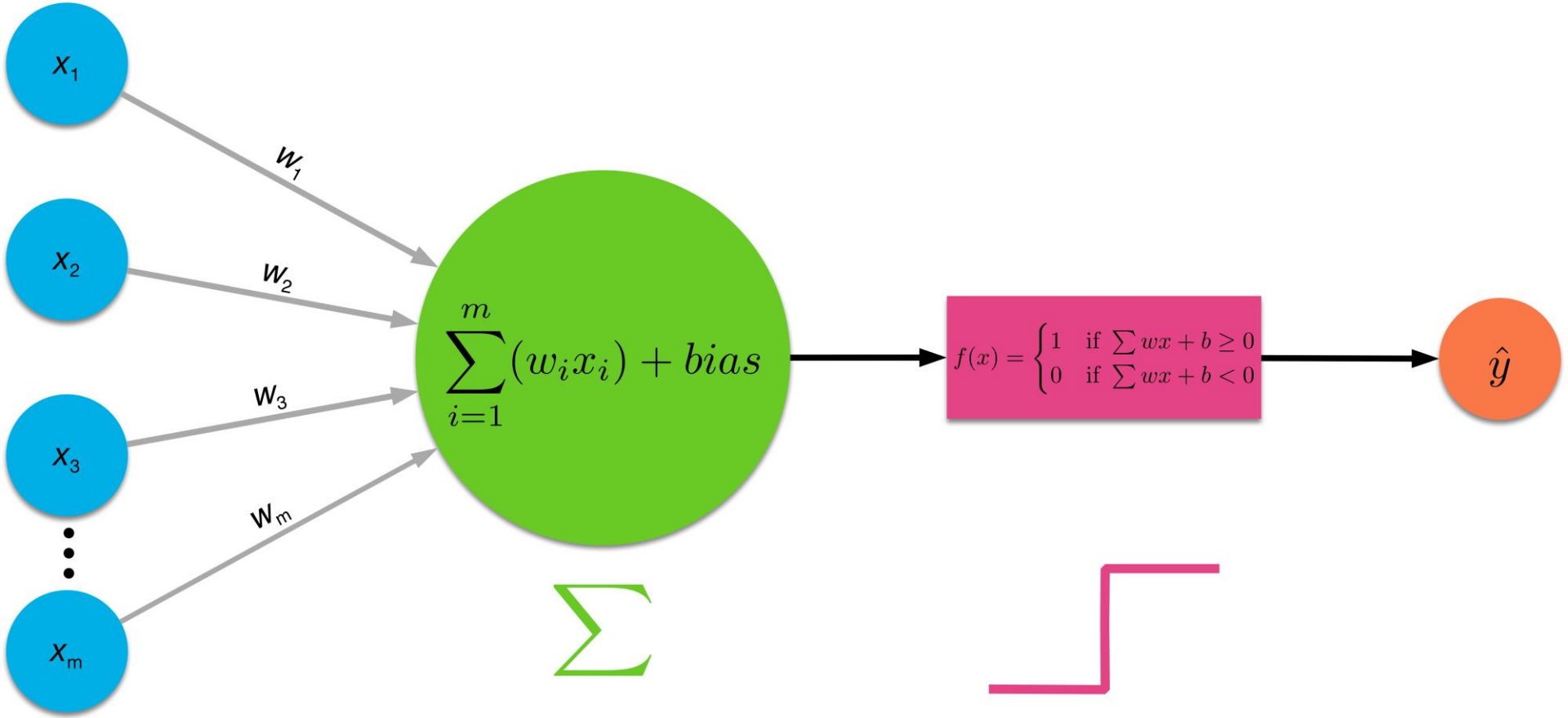
Artificial Neural Networks Architecture

- the **input** layer
- the **hidden** layer(s),
- the **output** layer.

Because of the numerous layers are sometimes referred to as the **Multi-Layer Perceptron** (MLP).

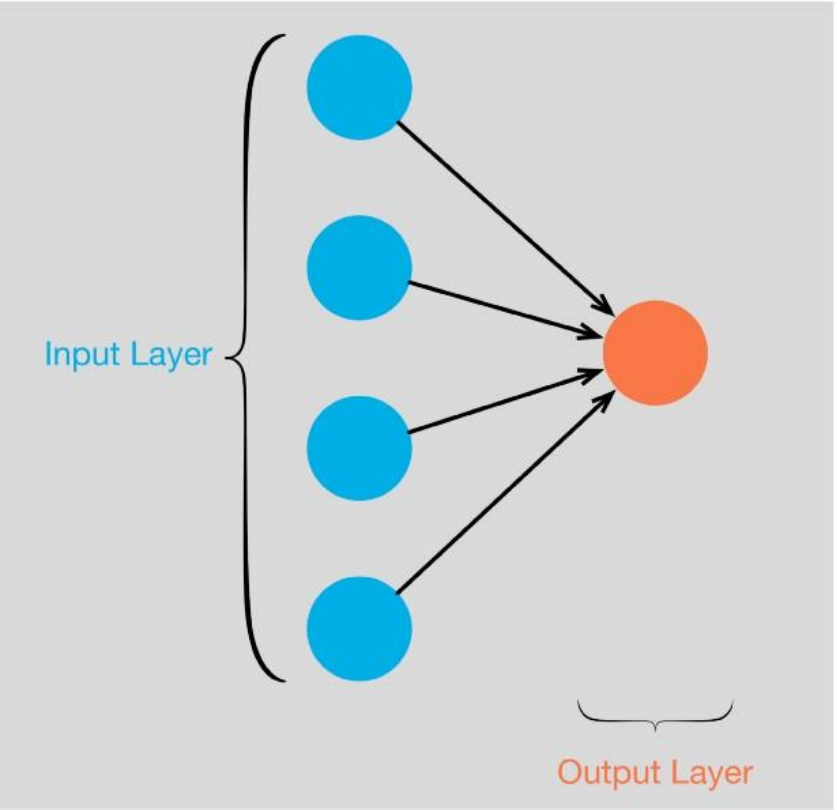


Single-layer Perceptron

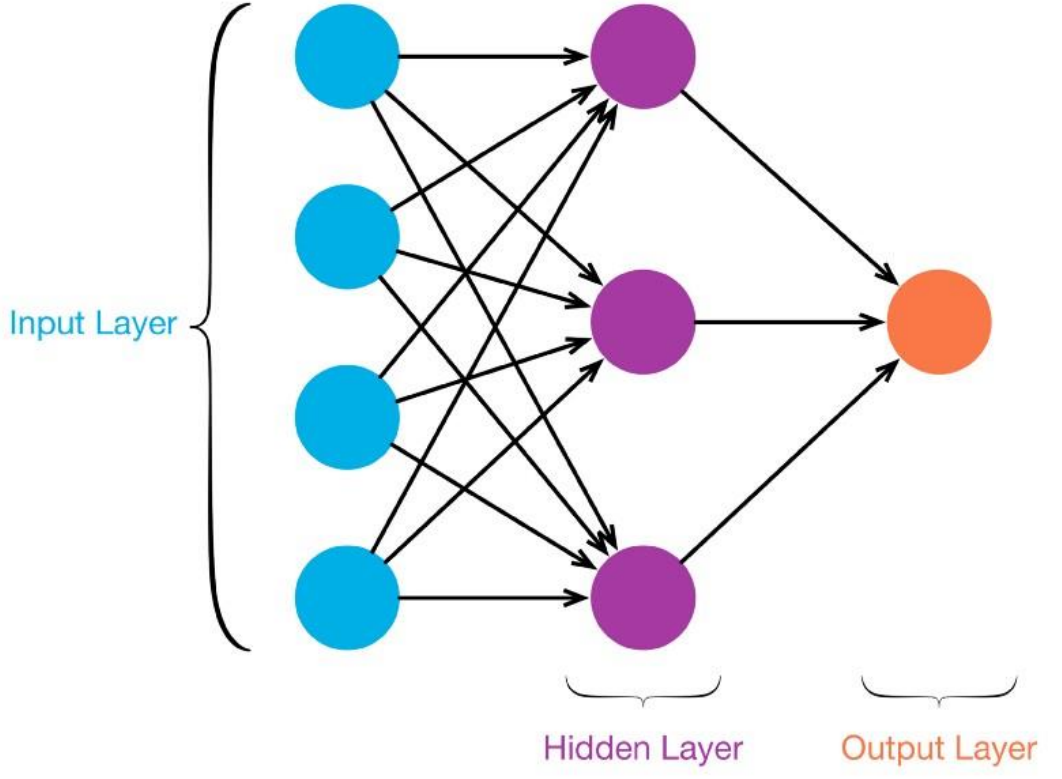


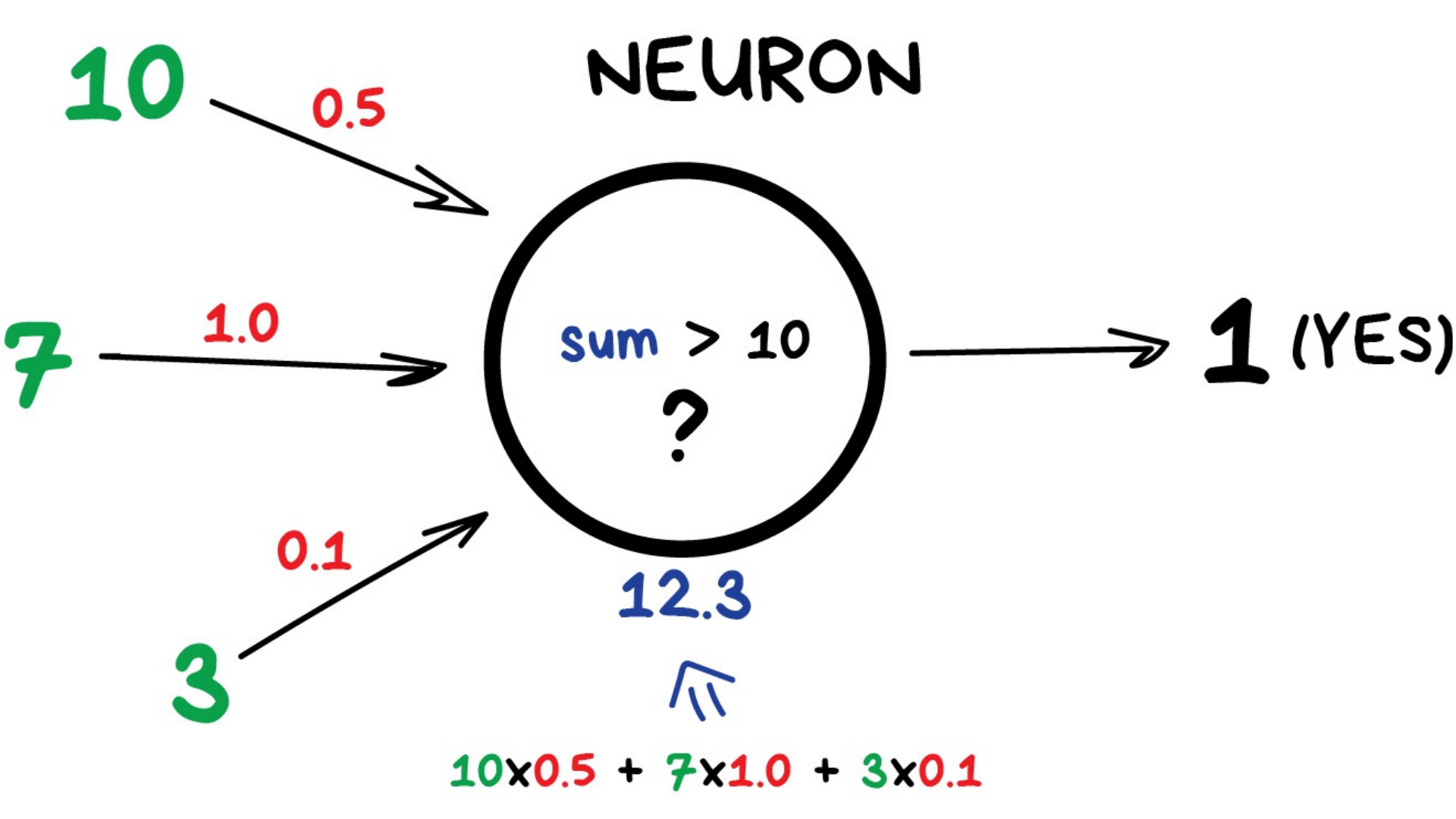
Single-layer vs. Single-hidden layer

Single-Layer Perceptron

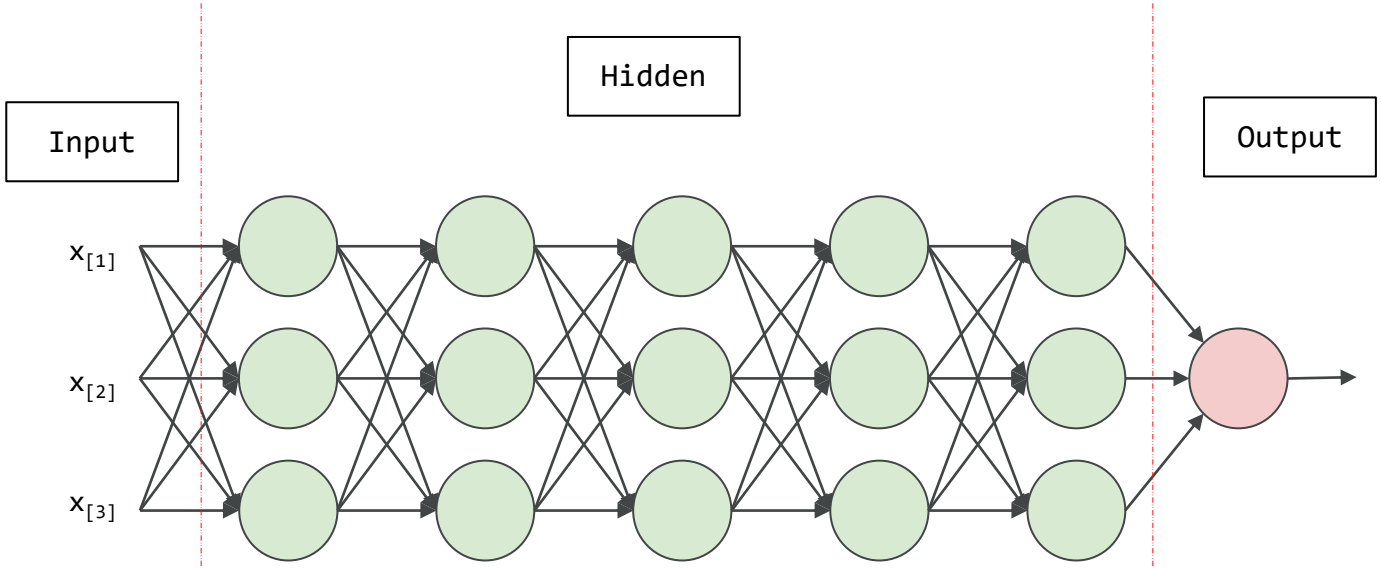


Perceptron with Hidden Layer

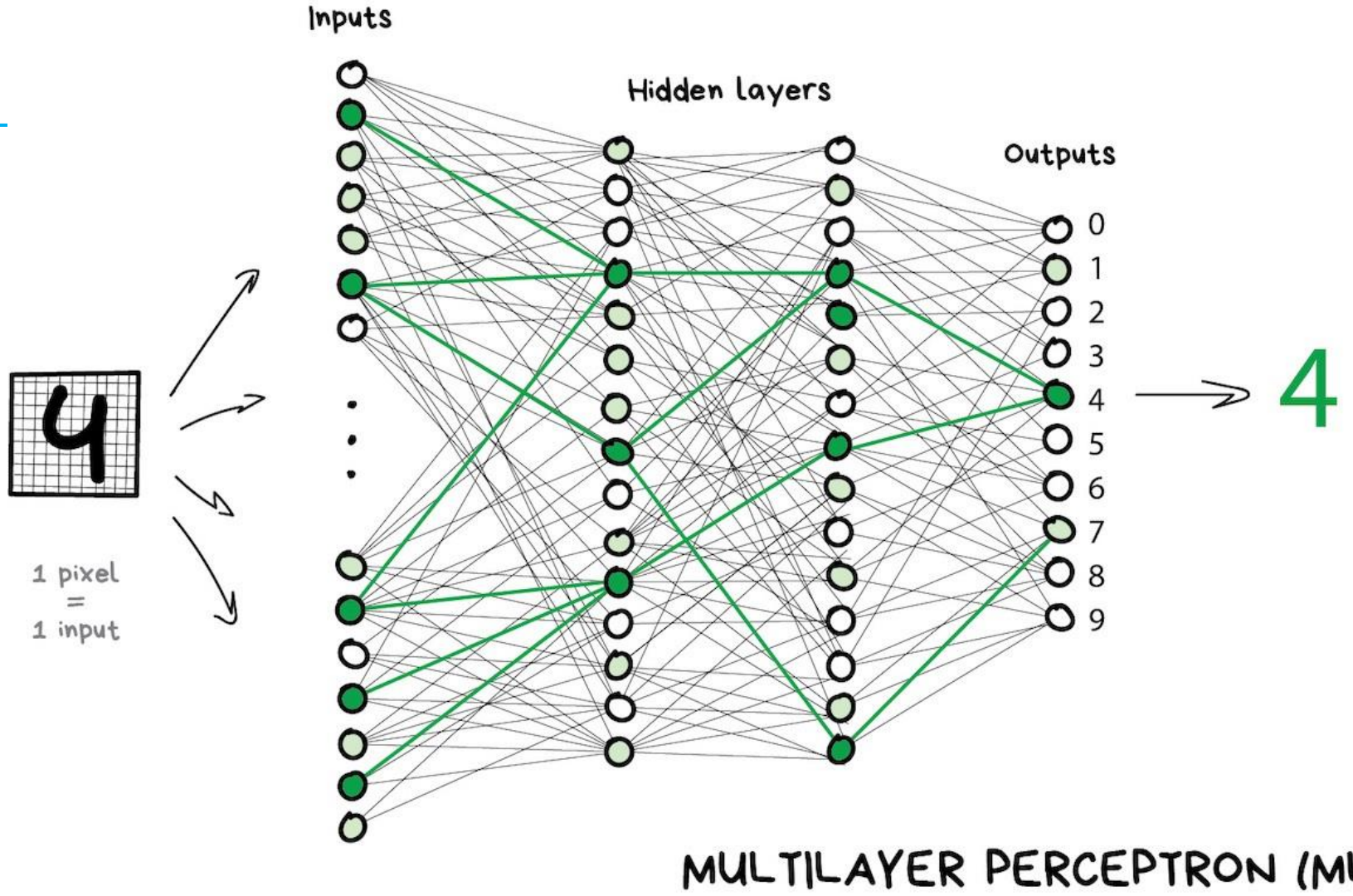



NEURON
10
0.5
7
1.0
3
0.1
sum > 10
?
12.3
1 (YES)
10x0.5 + 7x1.0 + 3x0.1

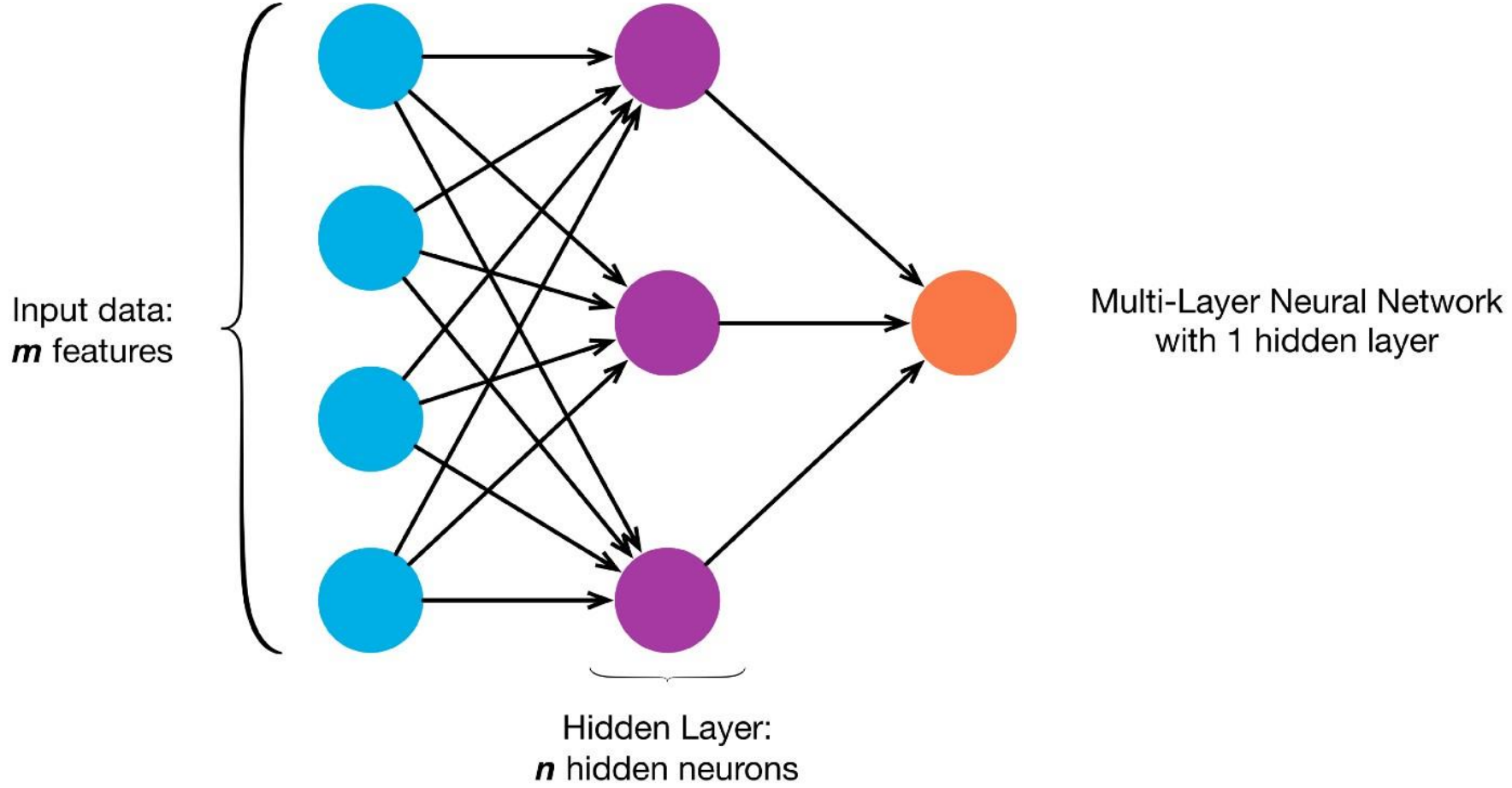
Multiple hidden layer ANN



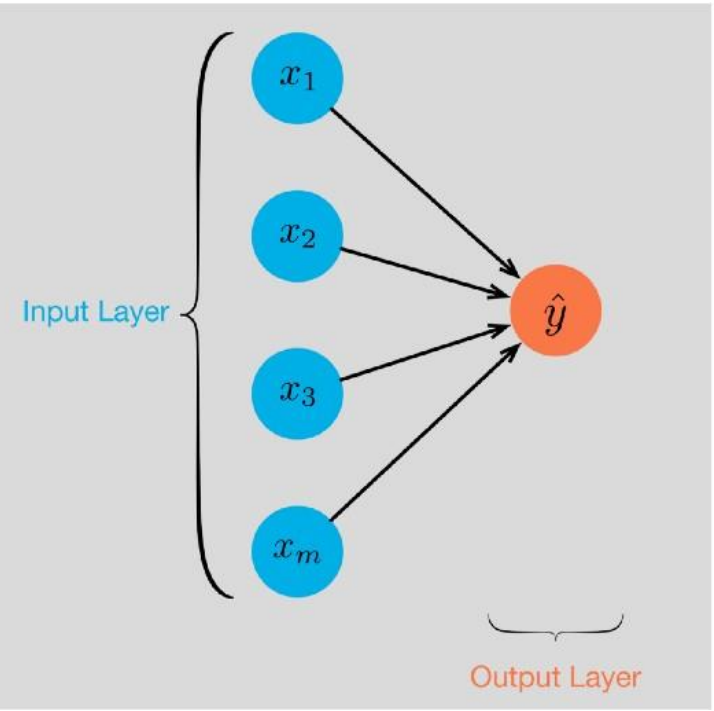
Number of hidden layers >= 3 -- Deep Learning

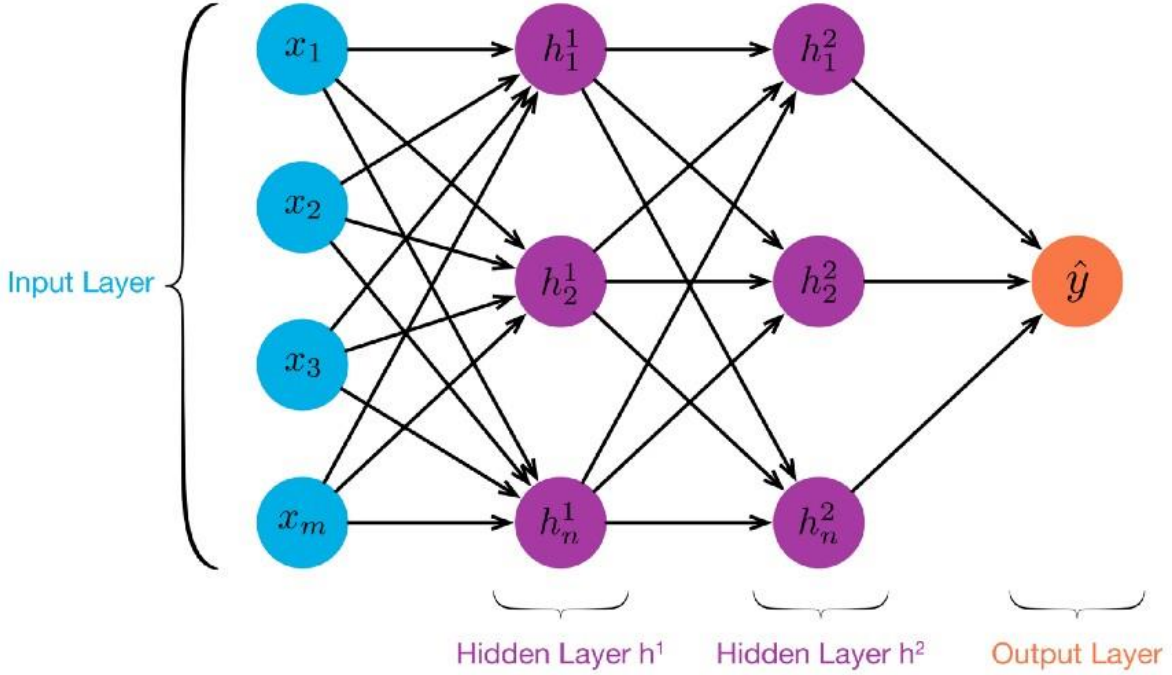

Inputs
Hidden layers
Outputs
0
1
2
3
4
5
6
7
8
9
4
1 pixel
=
1 input
MULTILAYER PERCEPTRON (MLP)

Multi-layer ANN



Single-layer vs. Multi-layer





How Do ANNs Work

- ANNs are inspired from the biological **neurons** within the human body which activate under certain circumstances resulting in a related action performed by the body in response.
- ANNs consist of various **layers** of interconnected artificial **neurons** powered by **activation functions** which help in switching them **ON/OFF**.

How Do ANNs Work

- Inspired by biological nervous systems, an ANN combines several processing layers, using **simple elements** operating **in parallel**.
- The ANN consists of an input layer, one or more hidden layers, and an output layer.
- In **each layer** there are **several** nodes, or neurons, and the nodes in each layer use the outputs of all nodes in the previous layer as inputs, such that all neurons interconnect with each other through the different layers.

How Do ANNs Work

- Each **neuron** typically is assigned a **weight** that is adjusted during the learning process and decreases or increases in the weight change the strength of that neuron's signal.

Like other machine learning algorithms:

ANNs can be used for:

- **supervised learning** (classification, regression)
- **unsupervised learning** (pattern recognition, clustering)

Model parameters are set by **weighting** the neural network through "**learning**" on **training data**, typically by optimising weights to minimize prediction error

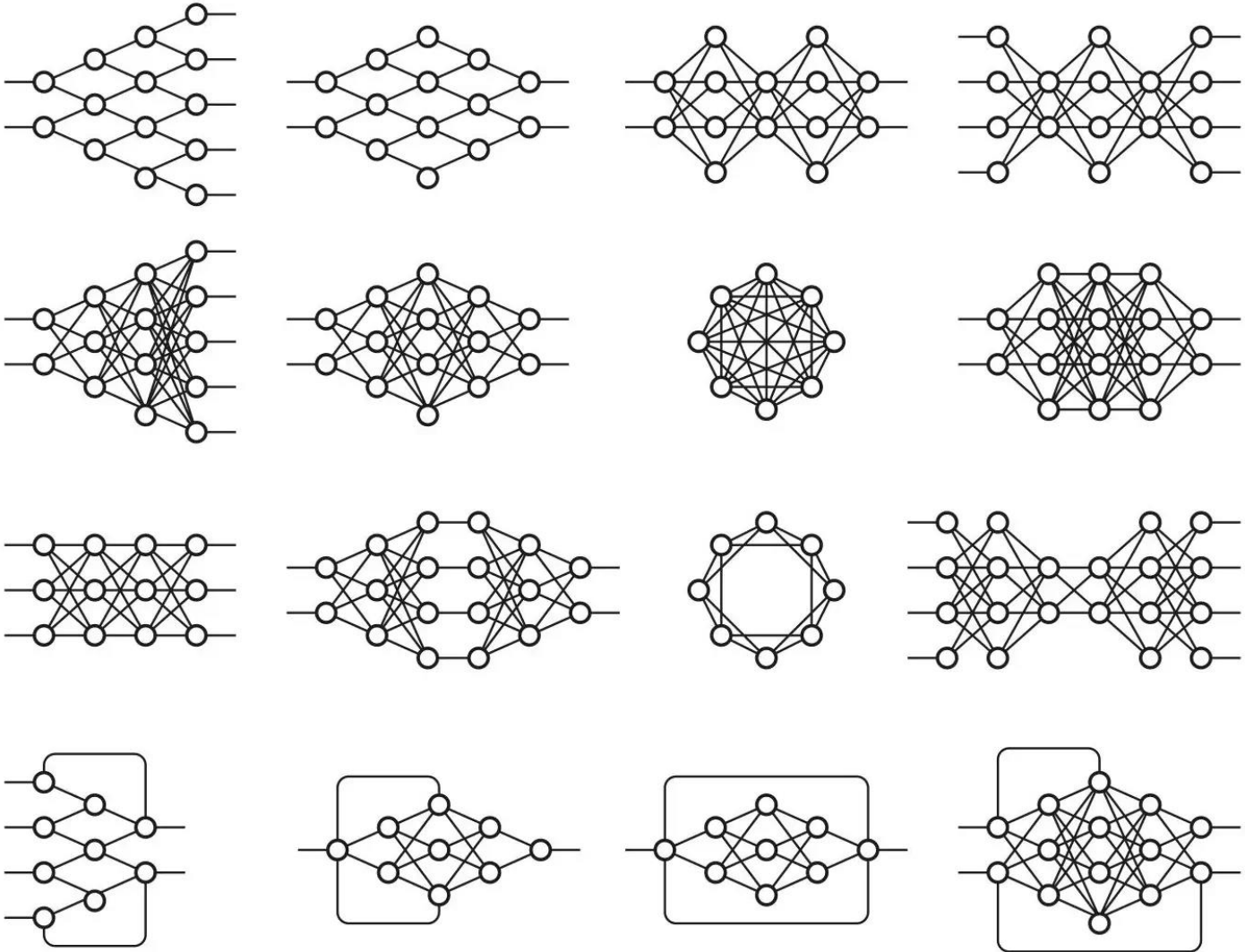
Types of ANN

There are many types of neural networks available or that might be in the development stage.

They can be classified depending on their:

- Structure,
- Data flow,
- Neurons and their density,
- Layers and their depth activation filters etc.

what is a neural network (19:12mins)



Types of ANN

- Perceptron
- Feed Forward Neural Network (FFNN)
- Multilayer Perceptron (MLP)
- Convolutional Neural Network (CNN)
- Radial Basis Functional Neural Network (RBFNN)
- Recurrent Neural Network (RNN)
- Long Short-Term Memory (LSTM)
- Sequence to Sequence Models (SSM)
- Modular Neural Network (MNN)

Perceptron

The first and simplest neural network was the perceptron, introduced by Frank Rosenblatt in 1958.

Perceptron is also known as TLU (threshold logic unit)

Perceptron is a **supervised** learning algorithm that classifies the data into two categories, thus it is a binary classifier.

A perceptron **separates** the input space into **two** categories by a hyperplane represented by the following equation

Perceptron

✓ **Advantages of Perceptron**

Perceptrons can implement **Logic Gates** like AND, OR, or NAND

✓ **Disadvantages of Perceptron**

Perceptrons can only learn **linearly** separable problems such as boolean AND problem.

For **non-linear** problems such as boolean XOR problem, it **does not work**.

Feedforward neural network

- Consists of an input layer, one or a few hidden layers, and an output layer (a typical shallow neural network, <=3)
- The simplest form of neural networks where **input data** travels **in one direction**, passing **through** artificial neural nodes and **exiting** through **output** nodes.
- Where **hidden layers may or may not** be present, input and output layers are present there. They can be further classified as **a single-layered** or **multi-layered** feed-forward neural network

Feedforward neural network

Applications on Feedforward Neural Networks

- Simple classification (where traditional Machine-learning based classification algorithms have limitations)
- Face recognition (Simple straight forward image processing)
- Computer vision (Where target classes are difficult to classify)
- Speech Recognition

Feedforward neural network

- **Number** of layers depends on the **complexity** of the function. It has uni-directional forward propagation but **no backward propagation**.
- Weights are **static** here.
- An activation function is fed by **inputs** which are **multiplied** by **weights**.
- To do so, **classifying** activation function or **step** activation function is used. For example: The neuron is activated if it is **above** threshold (usually **0**) and the neuron produces **1** as an output.
- The neuron is **not** activated if it is **below** threshold (usually 0) which is considered as -1. They are fairly simple to maintain and are equipped with to deal with data which contains a lot of **noise.**

Feedforward neural network

✓**Advantages of Feed Forward Neural Networks**

- Less complex, easy to design & maintain
- Fast and speedy [One-way propagation]
- Highly responsive to noisy data

✓**Disadvantages of Feed Forward Neural Networks**

- **Cannot** be used for deep learning (due to absence of dense layers and back propagation)

Multilayer Perceptron (MLP)[9,10]

- An entry point towards complex neural nets where **input** data **travels through** various layers of artificial **neurons**.
- Every single node is connected to **all neurons** in the next layer which makes it a **fully connected** neural network.
- Input and output layers are present having multiple hidden Layers i.e. **at least three** or more layers in total.
- It has a bi-directional propagation i.e. **forward** propagation and **backward** propagation.

Multilayer Perceptron

- **In forward propagation**, **Inputs** are **multiplied** with **weights** and fed to the **activation function**
- In **backpropagation**, they are **modified to reduce the loss**.
- In simple words, **weights** are machine **learnt** values **from** Neural Networks.
- They self-adjust depending on the **difference** between predicted outputs vs training inputs.
- **Nonlinear** activation functions are used followed by **softmax** as an output layer activation function.

Multilayer Perceptron

✓**Advantages on Multi-Layer Perceptron**

- Used for deep learning [due to the presence of dense **fully connected** layers and **back propagation**]

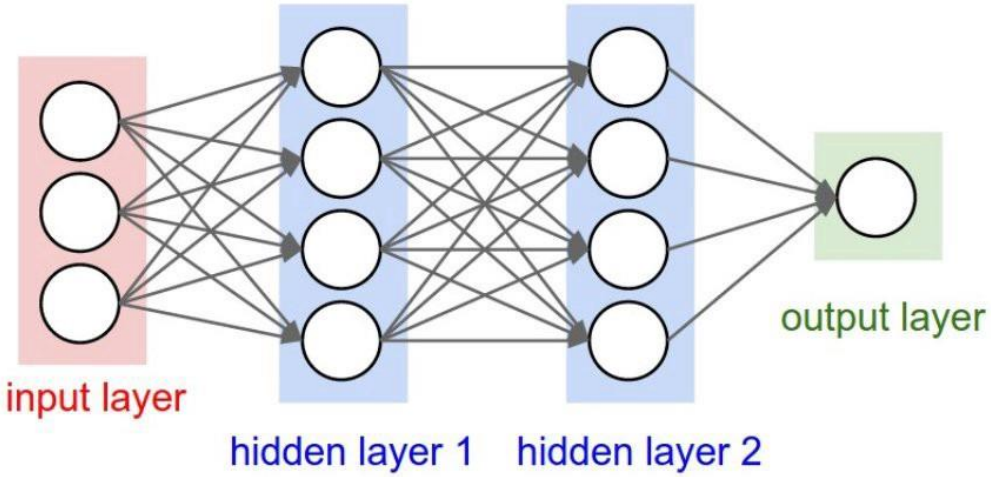
✓**Disadvantages on Multi-Layer Perceptron**

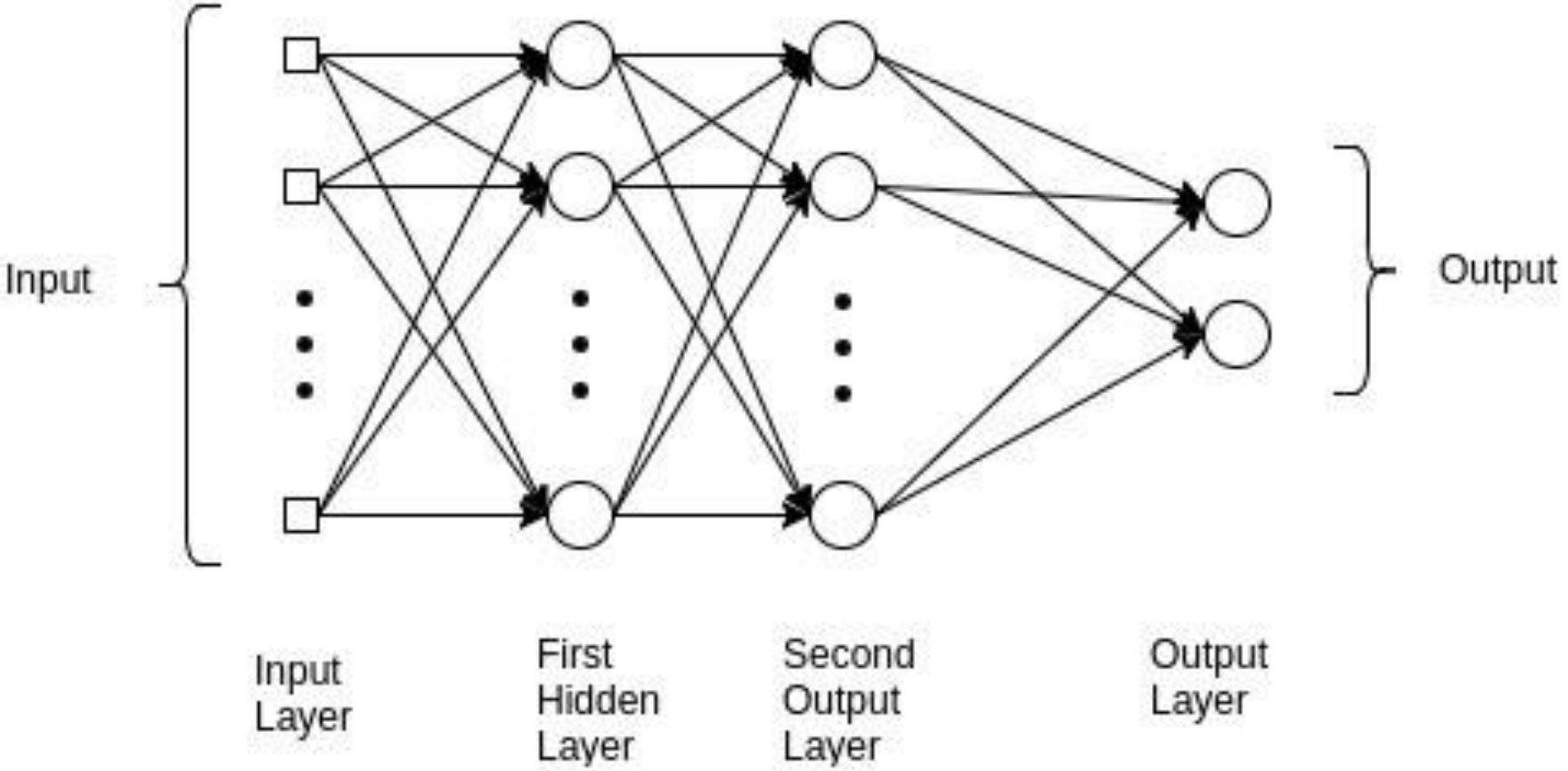
- Comparatively **complex** to design and maintain
- Comparatively **slow** (depends on number of hidden layers)

Multilayer Perceptron

Applications on Multi-Layer Perceptron

- Speech Recognition
- Machine Translation
- Complex Classification




Input
Output
Input Layer
First Hidden Layer
Second Output Layer
Output Layer

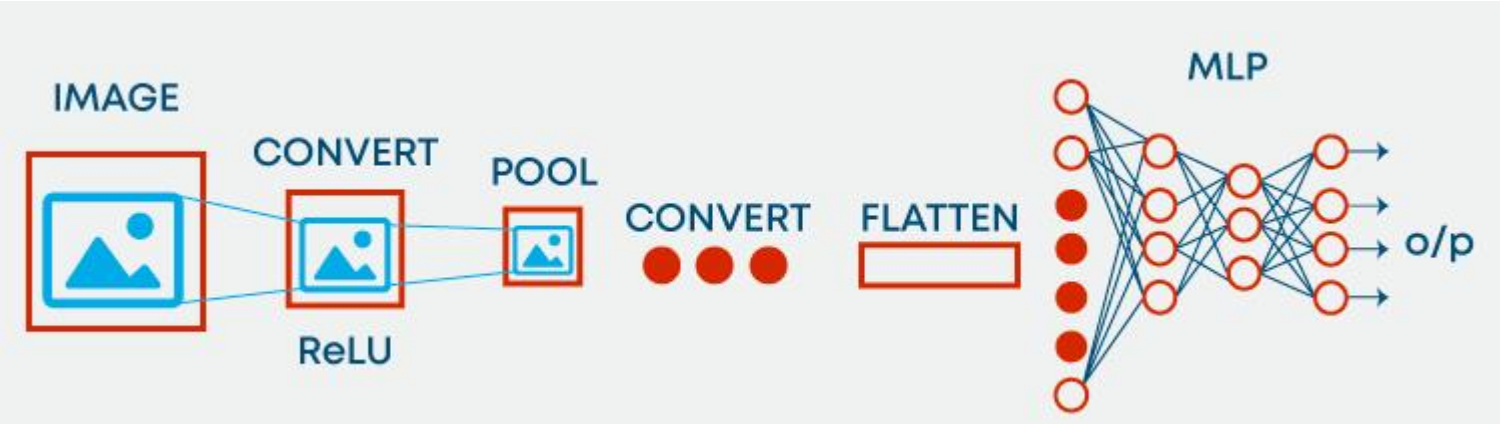
Backpropagation Algorithm

- Backpropagation (**BP**) is an algorithm for training multilayer neural networks that was developed by **Geoffrey Hinton** with **David Rumelhart** and **Ronald Williams** in 1986.
- It is based on the idea of:
  - *propagating the errors from the **output layer** back to the **input layer***
  - *adjusting the **weights** of the connections accordingly*

Convolutional neural network (CNN)

- **Convolutional** neural network (**CNN**):

Deep neural network architecture widely applied to image processing and characterized by convolutional layers that shift windows across the input with nodes that **share** weights, abstracting the (typically image) input to feature maps



Convolutional neural network (CNN)

Applications on Convolution Neural Network

- Image processing
- Computer Vision
- Speech Recognition
- Machine translation

Convolutional neural network (CNN)

- CNN contains a **three-dimensional** arrangement of neurons, **instead of** the standard **two-dimensional** array.
- The **1st layer** is called a **convolutional layer**. Each neuron in the convolutional layer only processes the information from a **small part** of the visual field.
- **Input features** are taken in batch-wise like a filter. The network understands the **images in parts** and can compute these operations multiple times to complete the full image processing.

Convolutional neural network (CNN)

- Processing involves **conversion** of the **image** *from RGB or HSI scale* to grey-scale. Furthering the changes in the pixel value **will help to** detect the edges and images can be classified into different categories.
- Propagation is uni-directional (single direction) where CNN contains one or more convolutional layers followed by pooling and bi-directional (mutual direction) where the output of convolution layer goes to a **fully connected** neural network for classifying the images as shown in the above diagram.

Convolutional neural network (CNN)

- **Filters** are used to **extract certain parts** of the **image**. In MLP the inputs are **multiplied** with weights and **fed** to the activation function.
- Convolution uses **RELU** and **MLP** uses **nonlinear** activation function followed by **softmax**. CNN shows very **effective results** in **image** and **video** recognition, semantic parsing and paraphrase detection.

Convolutional neural network (CNN)

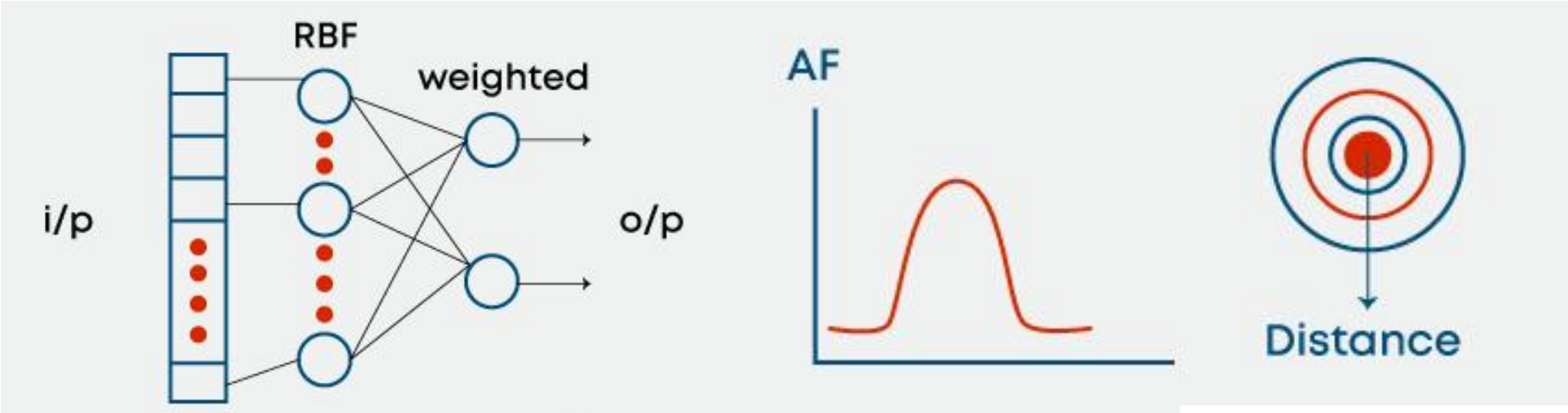
Advantages of Convolution Neural Network

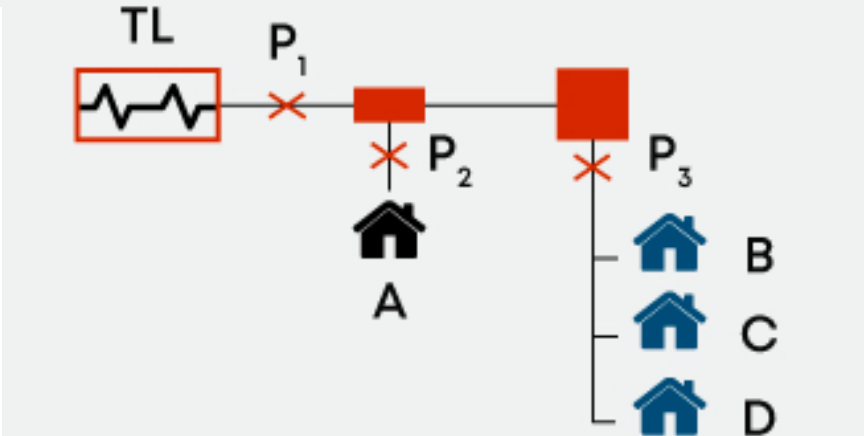
- Used for deep learning with **few** parameters
- **Less** parameters to learn as compared to fully connected layer

Disadvantages of Convolution Neural Network

- Comparatively **complex** to design and maintain
- Comparatively **slow** [depends on the number of hidden layers]

Radial Basis Function Neural Networks (RBFNN)





Radial Basis Function Neural Networks (RBFNN)

- RBFNN consists of an input vector followed by a layer of **RBF neurons** and an output layer with one node per category.
- Classification is performed by **measuring** the input's **similarity** to **data** points from the **training** set where each neuron stores a prototype.
- When a new input vector [the n-dimensional vector that you are trying to classify] needs to be classified, each neuron calculates the Euclidean distance between the input and its prototype.

One Example

- if we have two classes i.e. class A and class B,
- the new input to be classified is more close to class A prototype than the class B prototype. Hence, it could be tagged or classified as class A.
- Each RBF neuron compares the input vector to its class and outputs a value ranging which is a measure of similarity from 0 to 1.

One Example

- As the input equals to the prototype, the output of that RBF neuron will be 1 and with the distance grows between the input and prototype the response falls off exponentially towards 0.
- The curve generated out of neuron's response tends towards a typical bell curve. The output layer consists of a set of neurons [one per category].

Recurrent neural network (RNN)

Neural network architecture with feedback loops that model sequential dependencies in the input, as in time series, sensor, and text data; the most popular type of RNN is a long short-term memory network (LSTM)

Recurrent neural network (RNN)

Applications of Recurrent Neural Networks

- Text processing like auto suggest, grammar checks, etc.
- Text to speech processing
- Image tagger
- Sentiment Analysis
- Translation

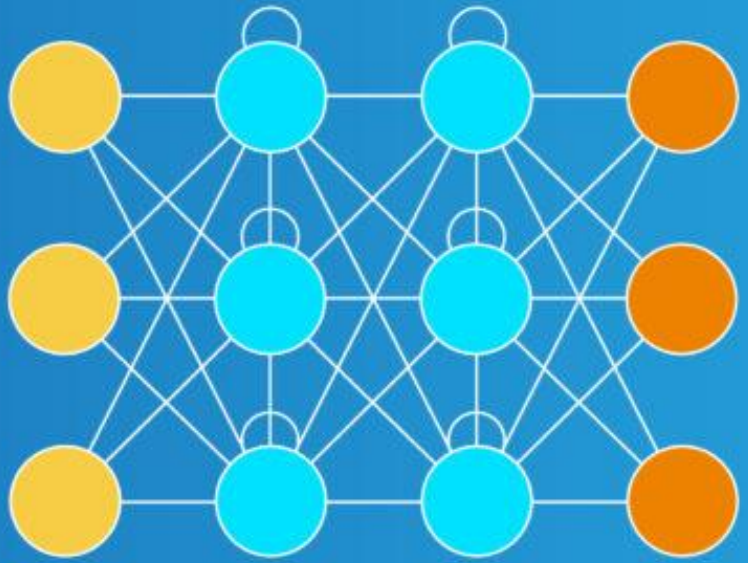
Recurrent neural network (RNN)

- Designed to **save** the output of a layer, RNN is fed back to the input to help in **predicting** the outcome of the layer.
- The **first layer** is typically a **feed forward** neural network followed by **recurrent** neural network layer where some information it had in the previous time-step is remembered by a memory function.

Recurrent neural network (RNN)

- **Forward** propagation is implemented in this case. It stores information required for it's future use.
- If the prediction is **wrong**, the learning rate is employed to make **small changes**. Hence, making it gradually increase towards making the right prediction during the backpropagation.

Recurrent Neural Network (RNN)



Input cell Output cell Recurrent cell

Recurrent neural network (RNN)

Advantages of Recurrent Neural Networks

- Model **sequential data** where each sample can be **assumed to be dependent** on historical ones is one of the advantage.
- Used with **convolution** layers to extend the pixel effectiveness.

Disadvantages of Recurrent Neural Networks

- **Gradient** vanishing and exploding problems
- **Training recurrent** neural nets could be a **difficult** task
- Difficult to process long sequential data using **ReLU** as an activation function Improveme

Long Short-Term Memory (LSTM) Networks

- LSTM networks are a **type of RNN** that uses special units in addition to standard units.
- LSTM units include a '**memory cell**' that can maintain information in memory for long periods of time.

Long Short-Term Memory (LSTM) Networks

- **A set of gates** is used to **control** when information **enters** the memory when it's **output**, and when it's **forgotten**.
- There are three types of gates: Input gate, output gate and forget gate:

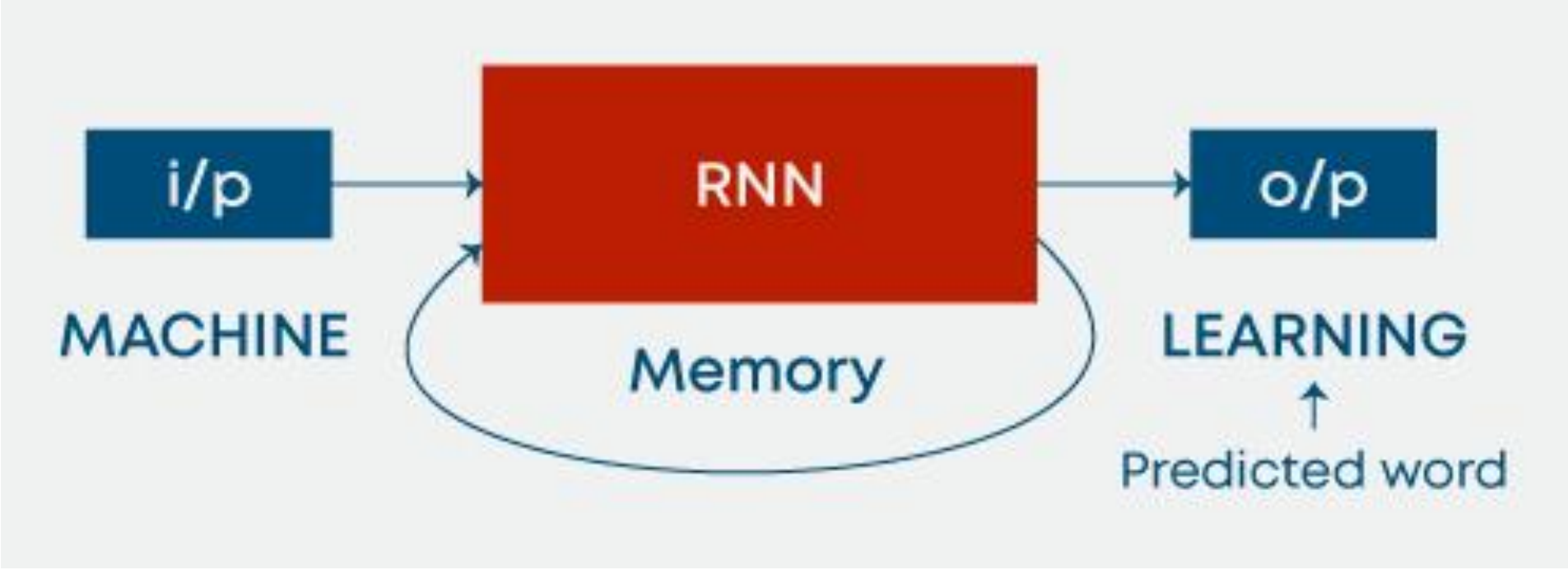
✓the **input** gate decides how many information from the last sample will be kept in memory;

✓the **output** gate regulates the amount of data passed to the next layer;

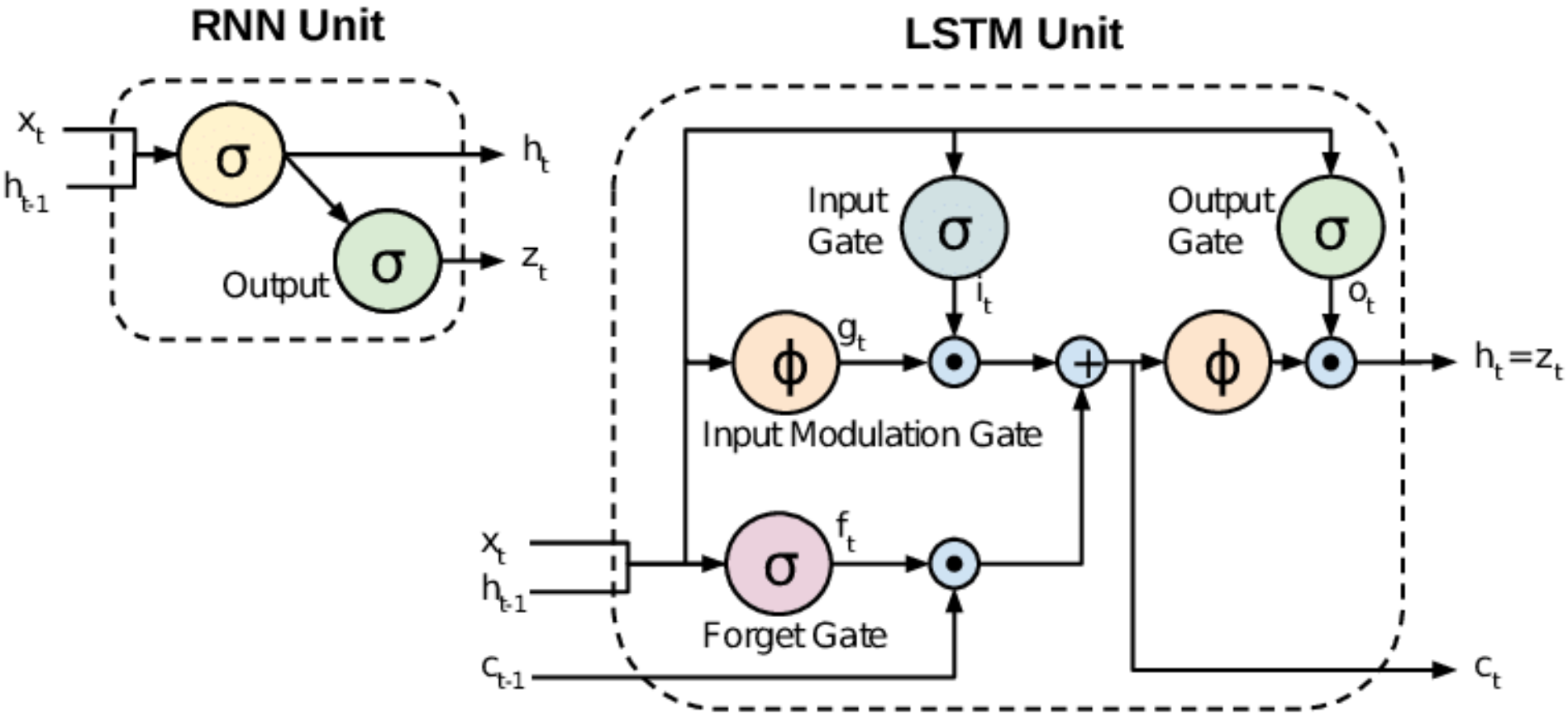
✓the **forget** gates control the tearing rate of memory

Long Short-Term Memory (LSTM) Networks

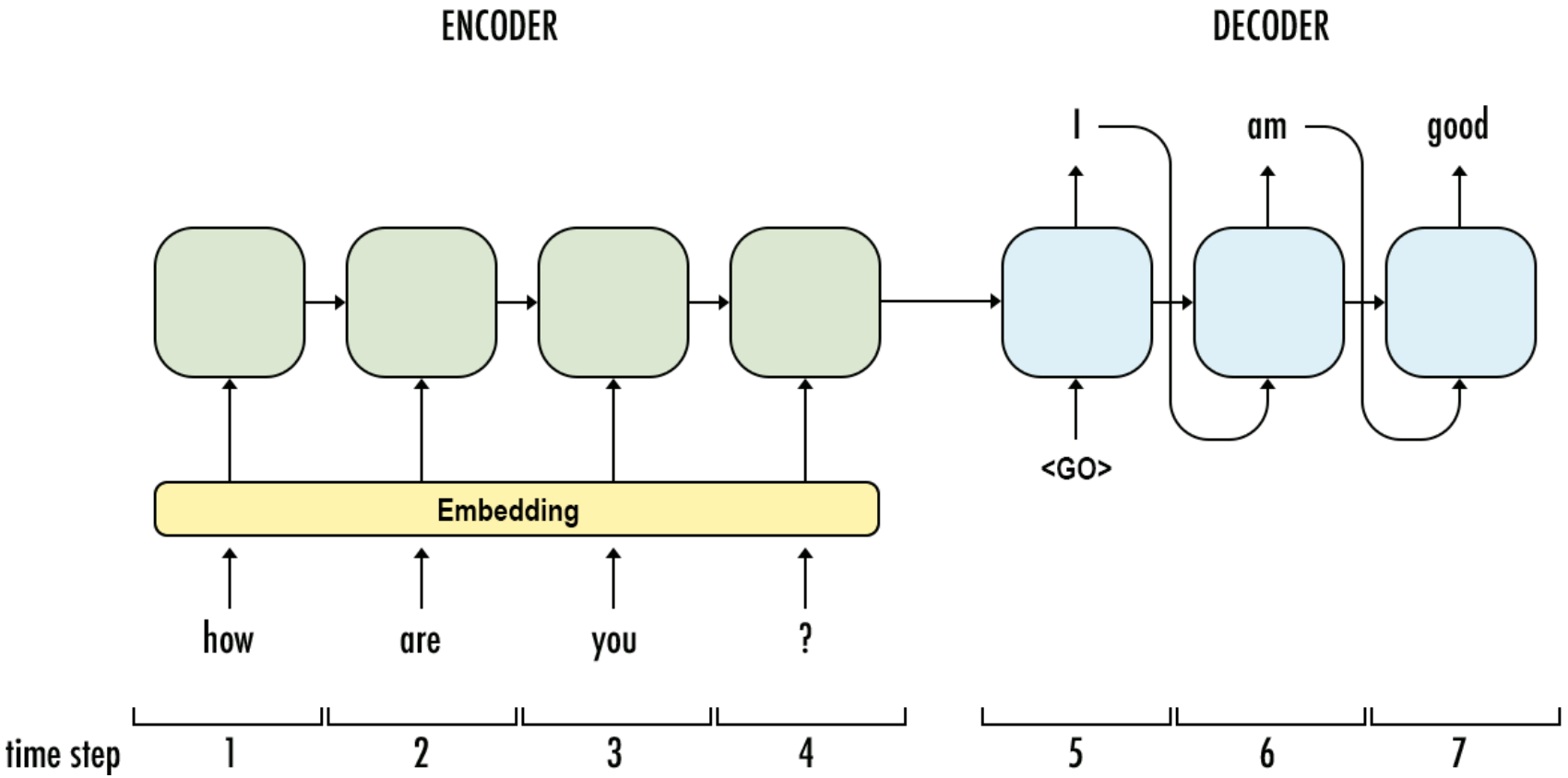
- This architecture lets them learn **longer-term dependencies**



RNN simple cell versus LSTM cell



Sequence to Sequence Models (SSM)



Sequence to Sequence Models (SSM)

- A sequence to sequence model consists **of two RNNs**.
- There exists an **encoder** that processes the **input** and a **decoder** that processes the **output**.
- The **encoder and decoder** work simultaneously – **either** using the **same parameter** or **different** ones. This model, on contrary to the actual RNN, is particularly applicable in those cases where the length of the input data is equal to the length of the output data.

Sequence to Sequence Models (SSM)

- While they possess **similar benefits** and limitations of the RNN, these models are usually applied mainly in **chatbots, machine translations, and question answering systems**.

Modular Neural Network (MNN)

- A MNN has a number of **different** networks that function independently and perform sub-tasks.
- The different networks do **not** really interact with or signal each other during the computation process. They work **independently** towards achieving the output.
- As a result, a large and complex computational process are done significantly faster by **breaking** it down into **independent components**.

Modular Neural Network (MNN)

The computation speed increases because the networks are **not** interacting with or even connected to each other.

Applications of Modular Neural Network

- Stock market prediction systems
- Adaptive MNN for character recognitions
- Compression of high level input data

Modular Neural Network (MNN)

Advantages of Modular Neural Network

- Efficient
- Independent training
- Robustness

Disadvantages of Modular Neural Network

- Moving target Problems

greatlearning
Learning for Life

A mostly complete chart of

Neural Networks

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)

Markov Chain (MC)

Hopfield Network (HN)

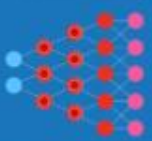
Boltzmann Machine (BM)

Restricted BM (RBM)

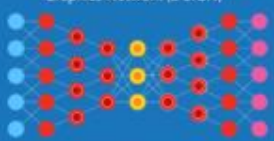
Deep Belief Network (DBN)

Deep Convolutional Network (DCN)

Deconvolutional Network (DN)



Deep Convolutional Inverse Graphics Network (DCIGN)



Generative Adversarial Network (GAN)

Liquid State Machine (LSM)

Extreme Learning Machine (ELM)

Echo State Network (ESN)

Deep Residual Network (DRN)

Kohonen Network (KN)

Support Vector Machine (SVM)

Neural Turing Machine (NTM)

Three Steps in ANN Design[18]

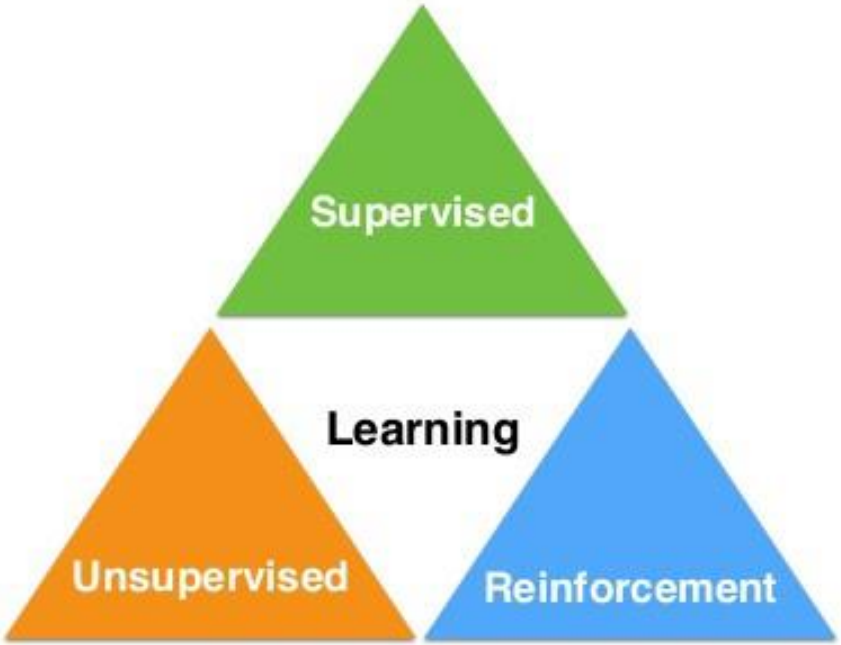
1. **Select** an appropriate **architecture** or **topology**, with **explicit** input and output nodes
2. **Gain correct network functions** through learning or training **weight** parameters
3. **Validate** the trained network in a generalisation process by testing its recognition performance with **un-used data** (that have **never** been seen before).

Learning Paradigms

- Supervised Learning
- Unsupervised Learning
- Reinforcement Learning

See more in 05-01 Machine Learning.pptx

- Labeled data
- Direct feedback
- Predict outcome/future



- No labels
- No feedback
- "Find hidden structure"

- Decision process
- Reward system
- Learn series of actions

Knowledge Representation

- Prior information (a priori knowledge)
- Observations (measurements)

General common sense rules for knowledge representation

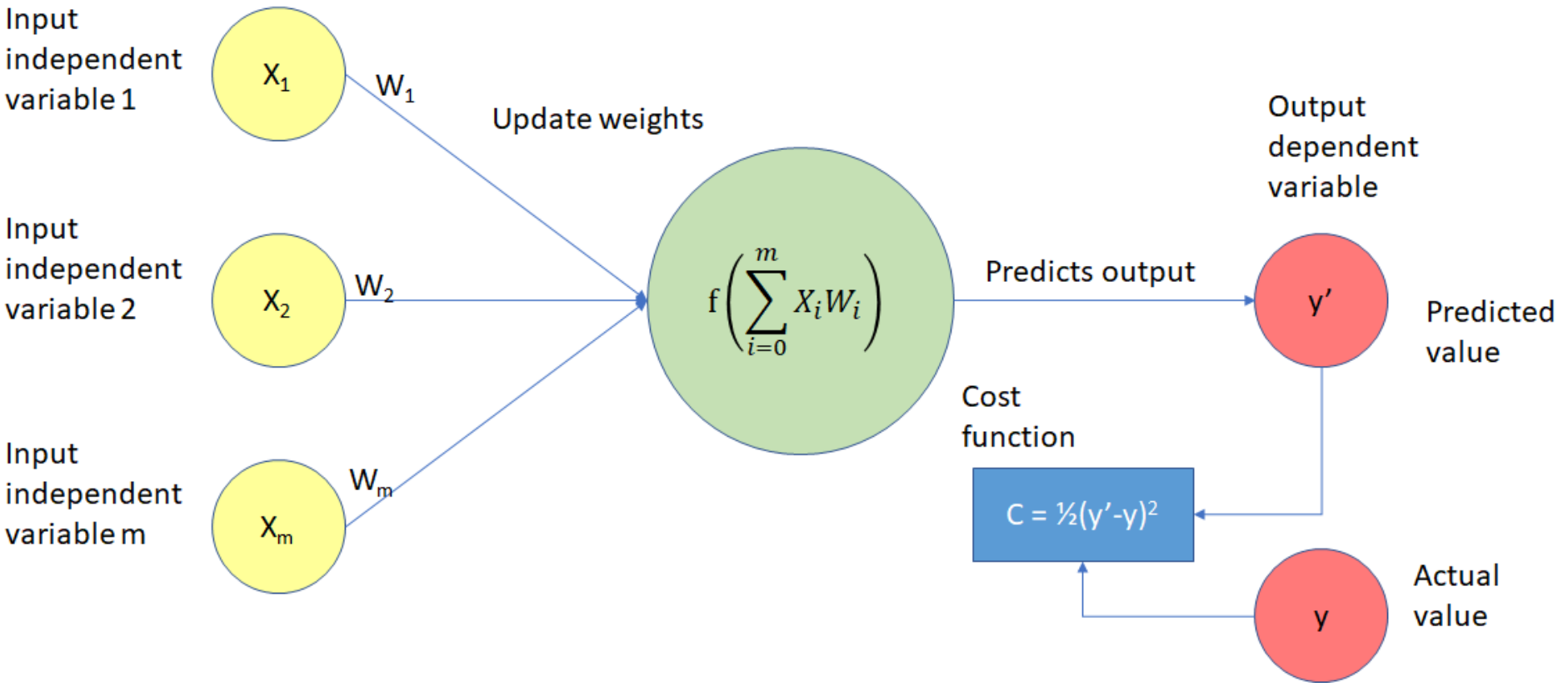
- **Rule 1:** Similar inputs from similar classes should usually produce similar representations inside the network and should therefore be classified as belonging to the same category
- **Rule 2:** Items to be categorised as separate classes should be given widely different representations in the network. (This rule is the exact opposite of Rule 1.)
- **Rule 3:** If a particular feature is important, then there should be a large number of neurons involved in the representation of that item in the network

General common sense rules for knowledge representation

- **Rule 4:** Prior information on the network and invariances should be built into the design of a neural network, thereby simplifying the network design by not having to learn them

For instance, a specialized/restricted structure, with a smaller number of independent (necessary connecting) parameters to train and thus a lower cost requires a small data set for training, learns faster and often generalizes better.

Loss Function and Cost Function



Loss Function and Cost Function

- A loss function is a **measure of error** between what value your model **predicts** and what the **value actually** is.
- **For example**, we **predict** the real value y_i for data point x_i *using the predicted data* $y_i^{'}$ *or* $\hat{y}_i$.

$$y_i^{'} = f_\theta(x_i) \quad (1)$$

- represent the prediction or output of some arbitrary model for the point x_i with parameters θ, one of the many loss functions could be

$$L_2 = \sum_{i=1}^{m} \left[y_i - y_i^{'} \right]^2 \quad (2)$$

Loss Function and Cost Function

- This function is known as the **L2 loss**. Training the hypothetical model we stated above would be the process of finding the θ that **minimises** this sum.
- The loss function (or error) is for a **single** training example
- The cost function is over the **entire** **training set** (or mini-batch for mini-batch gradient descent)

$$C_{L2} = \frac{1}{2m} \sum_{i=1}^{m} \left\| y_i - y_i^{'} \right\|^2 \qquad (3)$$

m is the total number of training examples

Loss Function and Cost Function

Generally cost and loss functions are synonymous, but cost function can contain regularisation terms in addition to **loss function**. Although it is not always necessary.

$$w^* = \arg\min_w \sum_j \left(t(x_j) - \sum_i w_i h_i(x_j) \right)^2 + \lambda \sum_{i=1}^{k} |w_i| \quad (4)$$

Objective, Loss and Cost Functions

- In the context of an optimisation algorithm, the function used to evaluate a candidate solution (i.e. a set of weights) is referred to as the **objective function**.
- We may seek to **maximise** or **minimise** the **objective function**, meaning that we are searching for a candidate solution that has the **highest** or **lowest** score respectively.
- In ANN, we seek to **minimise** the **error**. As such, the objective function is often referred to as a cost function or a loss function and the value calculated by the loss function is referred to as simply "*loss*."

Objective, Loss and Cost Functions

- *The function we want to **minimise** or **maximise** is called the **objective function** or criterion.*
- *When we are **minimising** it, we may also call it the **cost** function = **loss** function = error function.*
- *The cost function reduces all the various good and bad aspects of a possibly complex system down to a single number, a scalar value, which allows candidate solutions to be ranked and compared.*

Loss and Loss Function [21]

In calculating the error of the model during the optimisation process, ***a loss function must be chosen:***

- ***Maximum Likelihood***

  See more in FAQ08 and 09

  - ✓ *Mean Squared Error (MSE)*
  - ✓ ***Regression Problem*** *(linear activation)*
- ***Cross-Entropy***
  - ✓ *Referred to as logarithmic loss, logistic loss, or log loss*
  - ✓ ***Binary Classification Problem*** *(sigmoid activation)*
  - ✓ ***Multi-Class Classification Problem*** *(softmax activation)*

Lost function as Objective function

- ***maximize*** *the posterior probabilities (e.g., naive Bayes)*
- ***maximize*** *a fitness function (genetic programming)*
- ***maximize*** *the total reward/value function (reinforcement learning)*
- ***maximize*** *information gain/minimize child node impurities (CART decision tree classification)*
- ***minimize*** *a mean squared error cost (or loss) function (CART, decision tree regression, linear regression, adaptive linear neurons,* ***maximize*** *log-likelihood or* ***minimize*** *cross-entropy loss (or cost) function*
- ***minimize*** *hinge loss (support vector machine)*

Maximum likelihood estimation (MLE)[21]

- There are **many** functions that could be used to estimate the error of a set of weights in a neural network.
- We **prefer** a function where the space of candidate solutions **maps** onto a **smooth** (but high-dimensional) landscape that the optimisation algorithm can reasonably navigate via iterative updates to the model weights.
- **Maximum likelihood estimation, or MLE**, is a framework for inference for finding the **best** statistical estimates of parameters from **historical** training data: exactly what we are trying to do with the neural network.

Maximum likelihood estimation (MLE)

- Maximum likelihood seeks to find the **optimum** values for the parameters by **maximizing** a likelihood function derived from the training data.

Cross-Entropy

- Under the framework maximum likelihood, the error between two probability distributions is measured using cross-entropy.

Learning Rules (Learning Processes, Learning Algorithms)

1. **Hebbian Learning**
2. **Error-Correction Learning**
3. **Competitive Learning**
4. **Darwinian Selective Learning and Darwin Machine**

Training Neural Networks (12:28)

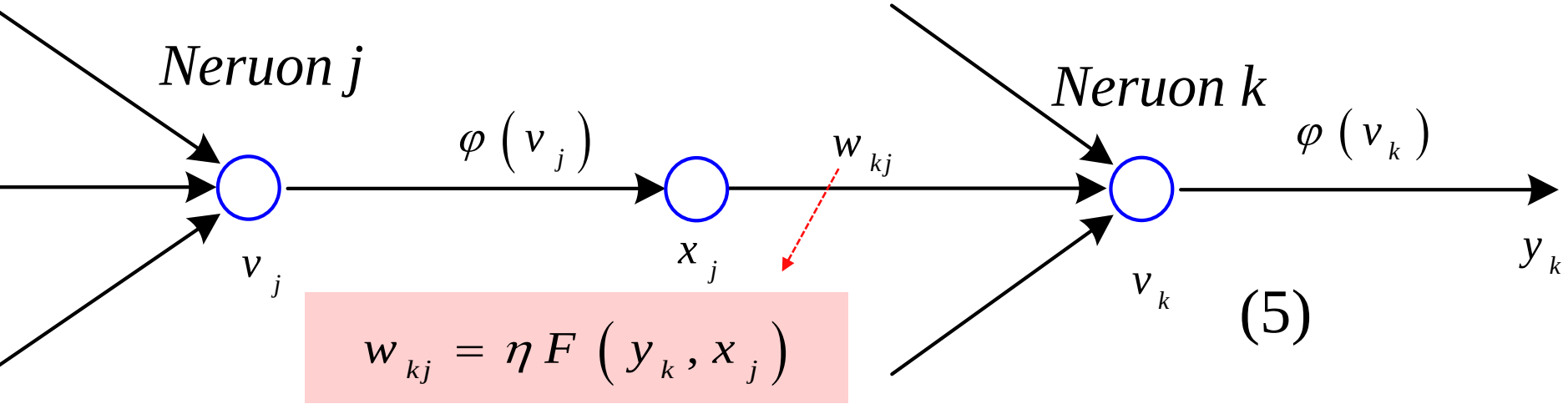
1. Hebbian Learning

This was inspired by **neuro-biological** phenomenon. The basic rules are:

- *If two neurons on **both** sides of a synapse are **activated synchronously**, then the strength of that synapse is **selectively increased**;*

- *If two neurons on **either** side of a synapse are **activated asynchronously**, then that synapse is selectively **weakened** or **eliminated**;*

Hebbian Learning



$$w_{kj} = \eta F(y_k, x_j) \quad (5)$$

- x_j = *presynaptic activity (i.e., input)*
- y_k = *postsynaptic activity (i.e., output)*
- *F(.) = function of both postsynaptic and presynaptic activities (i.e., both inputs and outputs)*
- η *is the learning rate*

Hebbian Learning

(6)

$$w_{kj} = \eta F\left(y_k, x_j\right) \xrightarrow{ab = F(a,b)} w_{kj} = \eta y_k x_j \qquad (7)$$

$$w_{kj}(n+1) = w_{kj}(n) + \Delta w_{kj} \qquad (8)$$

Where, *n* is the n^{th} data. With this learning rule, through repeated application of the input signal will lead to an **exponential growth** that finally drives the synapse's weight into **saturation**.

Hebbian Learning

To avoid this, a limit on the growth can be imposed by, for example, a nonlinear **forgetting factor** α as follows:

$$\Delta w_{kj} = \eta y_k x_j - \alpha y_k w_{kj}$$

$$= \eta y_k \left[x_j - \frac{\alpha}{\eta} w_{kj} \right] \qquad (9)$$

α is a new positive constant, representing the forgetting rate. This implies that the weights at time n + 1 will decrease if $\alpha w_{kj}(n)$ is greater than $\eta x_j(n)$.

Hebbian Learning

In **Hebbs** postulate, it is also sensible to use **statistical terms** to determine the amount of **adjustment**. One way of doing this is the activity **covariance rule**, as given by:

$$ab = F(a,b) \longrightarrow \mathrm{cov}[a,b] = F(a,b) \quad (10)$$

Hebbian Learning

$$\begin{aligned}\Delta w_{kj} &= \eta\, \mathrm{cov}[y_k, x_j] \\ &= \eta\, E[(y_k - \overline{y}_k)(x_j - \overline{x}_j)] \\ &= \eta\, E[y_k x_j] - \overline{y}_k \overline{x}_j\end{aligned} \tag{11}$$

- where, **E** is the statistical **expectation** operator and '**cov**' **stands** for the **covariance** function.
- Here $\bar{x}$ and $\bar{y}$ are the mean values of the pre-synaptic and post-synaptic activities, respectively.

2. Error-Correction Learning

① **Least-Mean-Square (LMS)** Algorithm-Based Error-Correction

② **Backpropagation (BP)** for Multilayer Perceptrons

Error Signal Definition

*The actual response of neuron **k** differs from the desired response before and during training. The difference between the **target response** d_k and the **actual response** y_k is defined as an **error signal** e_k:*

$$e_k = d_k - y_k \qquad (12)$$

This signal may be used to **guide the learning process** by **minimising** a **cost** function based on the error.

Error Energy

- This signal may be used to **guide** the learning process by **minimising** a cost function based on the error.
- A commonly used **cost function** is the means-square-error:

$$J = E\left\{\frac{1}{2}\sum_{k} e_k^2(n)\right\} \qquad (13)$$

Where, ***J*** stands for the (statistical average) effective **error energy** and the **summation** is over **all** the **neurons** in the output layer of the network.

Error Energy

- The factor $\frac{1}{2}$ is used to represent bipolar fluctuations (e.g., sine wave type of errors) and to simplify the derivative expressions.
- Note that **minimization** of the **cost function J** with respect to the network parameters leads to the so-called method of **gradient (descent) guidance**.
- A plot of J against weight in the multidimensional space is termed the error (**performance**) **surface**.

① Least-Mean-Square Algorithm-Based Error-Correction

- The **smaller** the learning rate **η**, the more **accurate** the tracking.
- The **small** η will usually take a **longer** convergence time, but a **large** η may result in **oscillations** (unstable learning).

$$\begin{aligned}\Delta w_k(n) &= \eta\left[x_k(n)d(n) - x_k(n)\sum_{j=1}^{p} w_j x_j(n)\right] \\ &= \eta x_k(n)\left[d(n) - y(n)\right] \\ &= \eta x_k(n)e(n)\end{aligned} \tag{14}$$

② Backpropagation for Multilayer Perceptrons

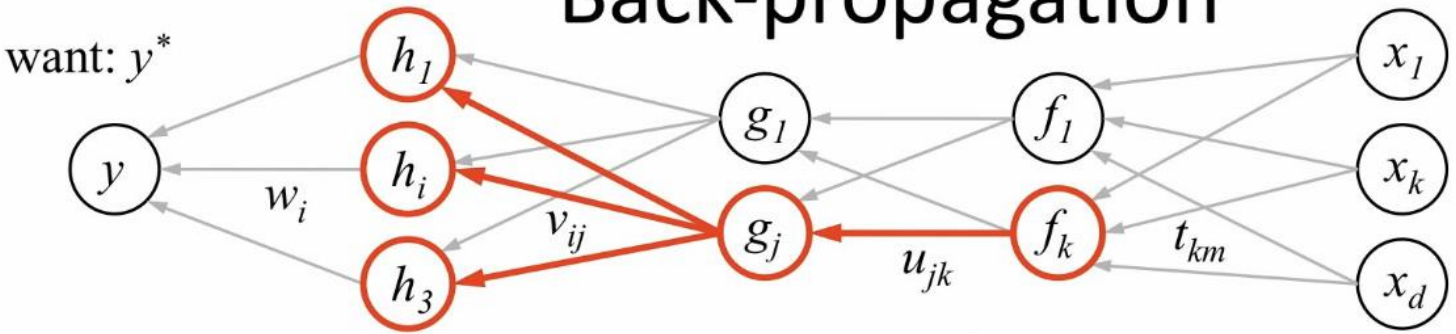
- **Backpropagation** is a popular method for training artificial neural networks, especially deep neural networks.
- **Error** is calculated between the **expected** outputs and the **outputs forward** propagated from the network.
- These **errors** are **then** propagated **backward** through the network from the output layer to the hidden layer, assigning blame for the error and updating weights as they go.
- See more in *03-02-DeepLearning.pptx*

$$J = \frac{1}{2}\sum_{j \in O} e_j^2(n) \qquad v_j(n) = u_j(n) - \theta_j(n) = \sum_{i=0}^{p} w_{ji}(n) y_j(n)$$

$$\begin{aligned} \Delta w_{ji}(n) &= -\eta \frac{\partial J(n)}{\partial w_{ji}} \\ &= -\eta \frac{\partial J(n)}{\partial y_j(n)} \frac{\partial y_j(n)}{\partial w_{ji}} \\ &= -\eta \frac{\partial J(n)}{\partial y_j(n)} \varphi'(v_j(n)) y_j(n) \end{aligned} \tag{14}$$

[Backpropagation in 5 Minutes]

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

$$\frac{\partial E}{\partial g_j} = \sum_i \sigma'(h_i) v_{ij} \frac{\partial E}{\partial h_i}$$

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

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

(i) compute error on u_{jk}

$$\frac{\partial E}{\partial u_{jk}} = \frac{\partial E}{\partial g_j} \sigma'(g_j) f_k$$

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

(ii) update the weight

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

3. Competitive Learning

A limit is imposed on the ‘strength’ of each neuron, such as:

$$\sum_{i} w_{ji} = 1, \quad \forall j \qquad (14)$$

***Winner takes-all**: A mechanism that permits the neurons to compete for **the right to respond** to a given subset of inputs, such that only one output neuron, or only one neuron per group, is active (i.e., ‘on’) at a time.*

Competitive Learning

The standard competitive learning rule adjusts the synaptic weights by:

$$\Delta w_{ji} = \begin{cases} \eta \left(x_i - w_{ji} \right), & j+ \\ 0, & j- \end{cases} \qquad (15)$$

in which,

j+, if neuron j **wins** the competition;

j–, if neuron j **loses** the competition.

4. Darwinian Selective Learning and Darwin Machine

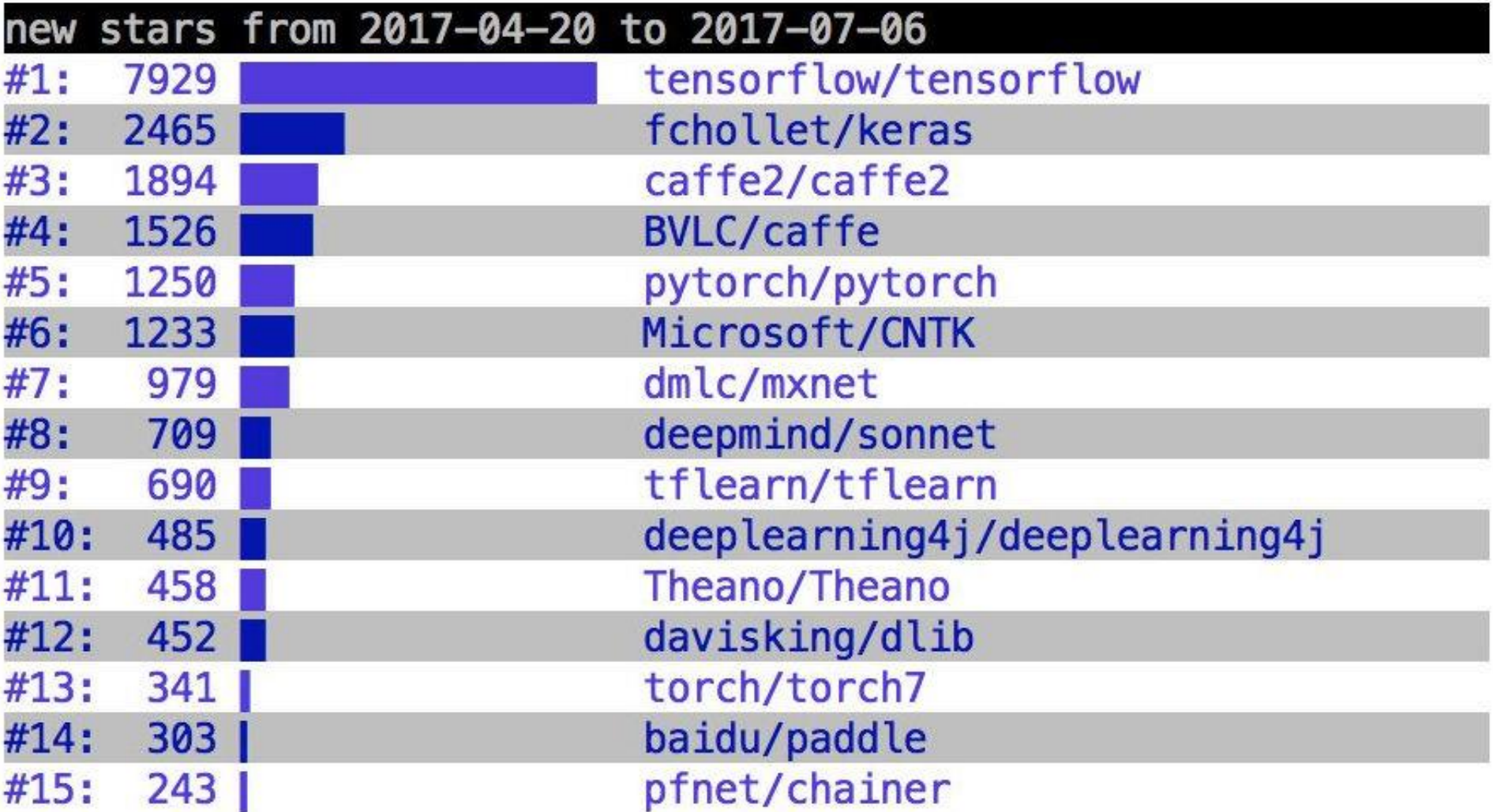
- Both learning mechanism are fundamentally related by the '**survival-of-the-fittest**'Darwinian principle.
- Through chromosome coding and genetic algorithms, the architecture of an ANN may be pruned (tailored to an appropriate sub-network) from a parent network,
- i.e., the architecture and weights can be trained simultaneously (for details, see [Li and Haußler (1996)]) [22]

Chapter Contents

Artificial Neural Network Platforms

- See from FAQ 11 Deep Learning Frameworks in

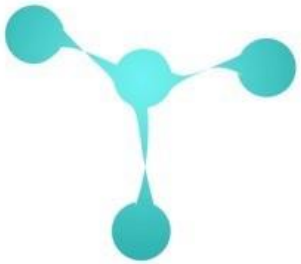
01-08-03 introduction-EN-FAQ.pptx


new stars from 2017-04-20 to 2017-07-06
#1: 7929 tensorflow/tensorflow
#2: 2465 fchollet/keras
#3: 1894 caffe2/caffe2
#4: 1526 BVLC/caffe
#5: 1250 pytorch/pytorch
#6: 1233 Microsoft/CNTK
#7: 979 dmlc/mxnet
#8: 709 deepmind/sonnet
#9: 690 tflearn/tflearn
#10: 485 deeplearning4j/deeplearning4j
#11: 458 Theano/Theano
#12: 452 davisking/dlib
#13: 341 torch/torch7
#14: 303 baidu/paddle
#15: 243 pfnet/chainer


Microsoft
CNTK


Caffe


Caffe2




PYTORCH


Chainer


K
Keras


TensorFlow


theano


∂y/net


mxnet


GLUON

Chapter Contents

Applications

1. Computer Vision (Facial Recognition)
2. Natural Language Processing[8]:
    1. *Text Classification and Categorization*
    2. *Named Entity Recognition (NER)*
    3. *Part-of-Speech Tagging*
    4. *Semantic Parsing and Question Answering*
    5. *Paraphrase Detection*
    6. *Language Generation and Multi-document Summarization*
    7. *Machine Translation*
    8. *Speech Recognition*
    9. *Character Recognition*
    10. ***Spell Checking***

Applications

3. Social Media
4. Aerospace, Automotive, Electronics, Manufacturing, Mechanics, Robotics and Telecommunications, Defence
5. Healthcare
6. Signature Verification and Handwriting Analysis
7. Weather Forecasting
8. Handwriting Recognition
9. Travelling Salesman Problem

Applications

10. Image Compression
11. Stock Exchange Prediction

- Neural Network Visualization (7:35mins)
- A Visual Introduction to The Neural Network (13:50mins)
- Visualizing Neural Network 3D Simulation (2:43mins)

FAQs

FAQ01 What's ANN?

FAQ02 How does ANN work?

FAQ03 What are 3 major categories of neural networks?

FAQ04 What is CNN and DNN?

FAQ05 How does CNN differ from ANN?

FAQ06 Why is CNN better than MLP?

FAQs

FAQ 07 Informative Chat

FAQ 08 Loss Functions

FAQ 09 loss function, cost function and objective function

FAQ 10 Terminology

FAQ 11 Videos for ANN

FAQ 12 ANN Diagram and Graph

FAQ01 What's ANN

- The ANN or NN is an adaptive system that learns by using interconnected **nodes** or **neurons** in a **layered structure** that resembles a human brain.
- ANN can learn from data - so it can be trained to recognise patterns, classify data, and forecast future events.
- ANN can break down the input into layers of abstraction. It can be trained using many examples to recognize patterns in speech or images, just as the human brain does.

FAQ02 How does ANN work

Its behaviour is defined by:

- the way its individual elements are connected
- the strength, or weights, of those connections.

These weights are automatically adjusted during training according to a specified learning rule until the ANN performs the desired task correctly.

FAQ03 What are 3 major categories of neural networks?

The three most important types of neural networks are:

- Artificial Neural Networks (ANN)
- Convolution Neural Networks (CNN)
- Recurrent Neural Networks (RNN)

FAQ04 What is CNN and DNN?

- A deep neural network (DNN) is an artificial neural network (ANN) with multiple layers between the input and output layers. They can model complex non-linear relationships.
- Convolutional Neural Networks (CNN) are an alternative type of DNN that allow modelling both time and space correlations in multivariate signals.

FAQ05 How does CNN differ from ANN?

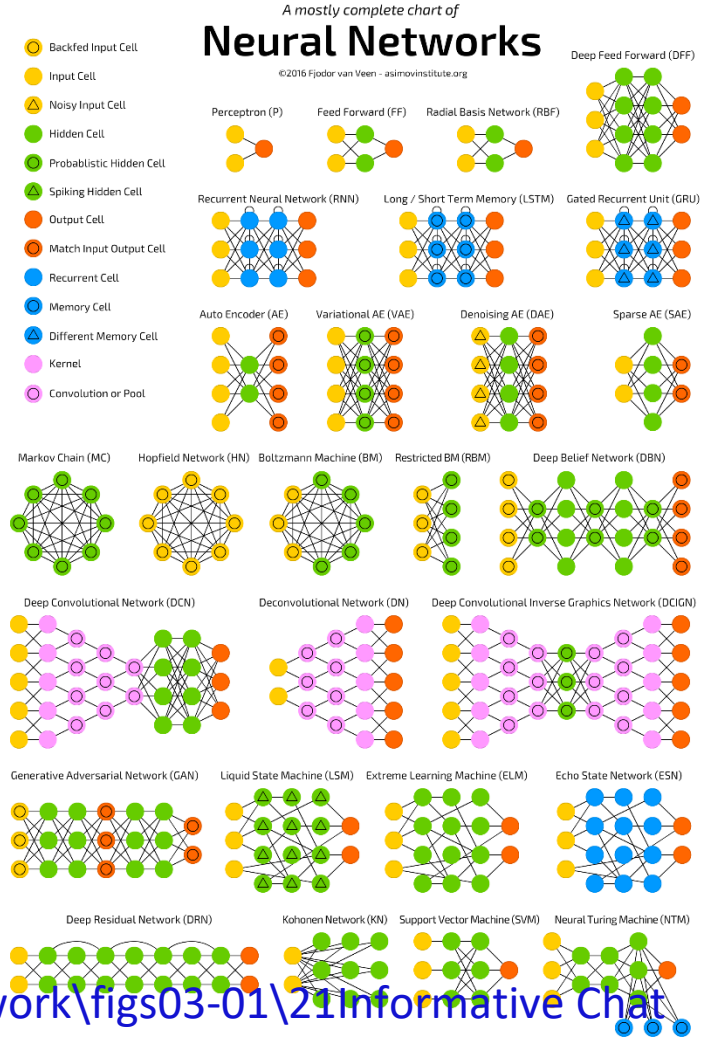
- CNN is a specific kind of ANN that has one or more layers of convolutional units.
- The class of ANN covers several architectures including Convolutional Neural Networks (CNN), Recurrent Neural Networks (RNN), e.g. LSTM and GRU, Autoencoders, and Deep Belief Networks.

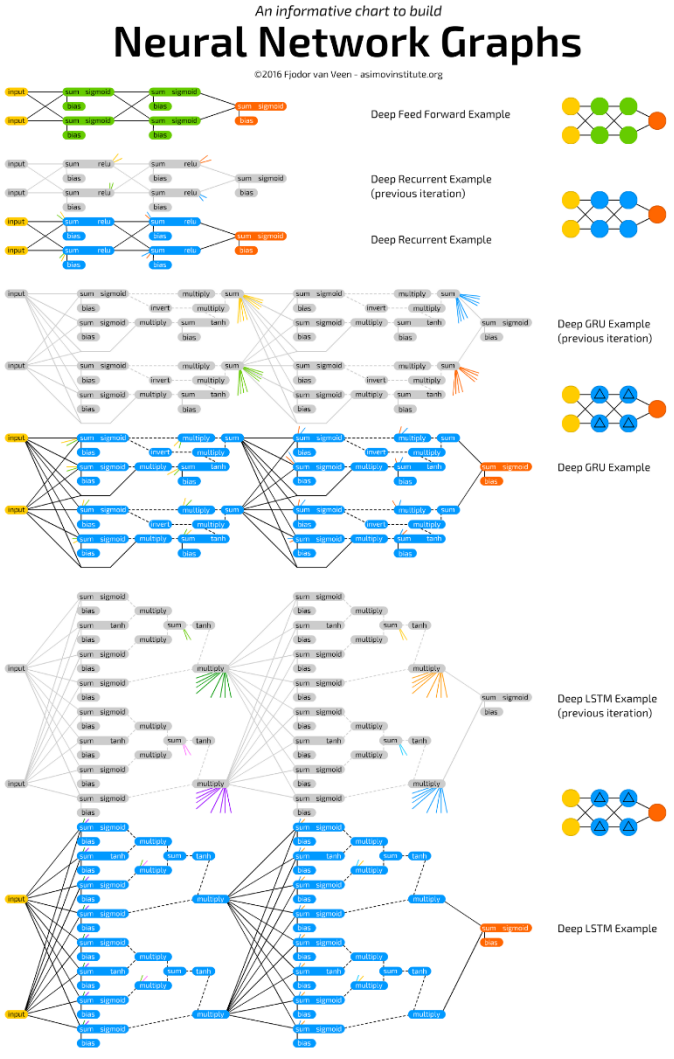
FAQ06 Why is CNN better than MLP?

- Multilayer Perceptron (MLP) is great for MNIST dataset[9] as it is a simpler and more straight forward dataset, but it lags when it comes to real-world application in computer vision, specifically image classification as compared to CNN which is great.

FAQ 07

Informative Chat





03-Artificial Neural Network\figs03-01\21Informative Chat

FAQ 08 Loss Functions

- L1 loss, smooth L1 loss
- Mean Squared Error (MSE) Loss = L2 Loss
- Mean Absolute Error Loss
- Huber Loss
- Quantile Loss
- Cross Entropy Loss
- Hinge Loss
- Exponential Loss
- 0-1 Loss
- IoU/GIoU/DIoU/CIoU Loss

L1, Smooth L1 and L2

$$L_1(x) = |x|$$

$$L_2(x) = x^2$$

x

$$smoothL_1(x) = \begin{cases} 0.5x^2, if\ |x| < 1 \\ |x| - 0.5, otherwise \end{cases}$$

Mean Squared Error (MSE) loss as L2 loss

$$J_{MSE} = \frac{1}{N} \sum_{i=1}^{N} (y_i - \hat{y}_i)^2$$

Mean Absolute Error (MAE) Loss as L1 loss

$$J_{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i|$$

Huber Loss = MSE + MAE

$$J_{huber} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{|y_i - \hat{y_i}| \leq \delta} \frac{(y_i - \hat{y_i})^2}{2} + \mathbb{I}_{|y_i - \hat{y_i}| > \delta}(\delta|y_i - \hat{y_i}| - \frac{1}{2}\delta^2)$$

Quantile Loss

$$J_{quant} = \frac{1}{N}\sum_{i=1}^{N} \mathbb{I}_{\hat{y}_i \geq y_i}(1-r)|y_i - \hat{y}_i| + \mathbb{I}_{\hat{y}_i < y_i} r|y_i - \hat{y}_i|$$

$$J_{quant}^{r=0.5} = \frac{1}{N}\sum_{i=1}^{N} |y_i - \hat{y}_i|$$

Cross Entropy Loss

$$NLL(x, y) = J_{CE} = -\sum_{i=1}^{N}(y_i \mathbf{log}(\hat{y_i}) + (1 - y_i)\mathbf{log}(1 - \hat{y_i}))$$

$$NLL(x, y) = J_{CE} = -\sum_{i=1}^{N}\sum_{k=1}^{K} y_i^k \mathbf{log}(\hat{y_i^k})$$

Hinge Loss

$$J_{hinge} = \sum_{i=1}^{N} \max(0, 1 - \mathbf{sgn}(y_i)\hat{y}_i)$$

widely used in Support Vector Machine (SVM)

FAQ 09 loss function, cost function and objective function

- The loss function (or error) is for a **single** training example
- The cost function is over the **entire** **training set** (or mini-batch for mini-batch gradient descent)
- The objective function in the optimisation problem being solved
- A **loss** function is a part of a **cost** function which is a type of an **objective** function.

FAQ 10 Terminology - Chinese

• Neuron	/ˈnjʊərɒn/	神经元
• Nucleus	/ˈn(y)o͞oklēəs/	细胞核
• Myelin	/ˈmīələn/ Sheath /SHēTH/	髓鞘
• Myelinated	/ˈmīələˌnādəd/ axon trunk	有髓鞘的轴突主干
• Axons	/ˈakˌsän/	轴突
• Soma	/ˈsōmə/	体细胞
• Dendrites	/ˈdendrīt/	树突
• Synapse	/ˈsinˌaps/	突触
• Synaptic	/səˈnaptik/ Weights	突触权重
• Perceptron	/pərˌsepträn/	感知器
• Arborisation	/ˌärb(ə)rəˈzāSHən/	树枝化
• Terminal Arborisation		神经纤维的终端树枝化/分支化

FAQ 11 – Videos for Artificial Neural Network

- What is a neural network (5:44)
- What is a neural network (19:00)
- Neural Network Tutorial For Beginners (3:17:00)

FAQ 12 ANN Diagram and Graph

1 PlotNeuralNet

Latex
https://github.com/HarisIqbal88/PlotNeuralNet

FCN-8:
https://www.overleaf.com/read/kkqntfxnvbsk

FCN-32：
https://www.overleaf.com/read/wsxpmkqvjnbs

Holistically-Nested Edge Detection：
https://www.overleaf.com/read/jxhnkcnwhfxp

FAQ 12 ANN Diagram and Graph

2 MATLAB
https://www.mathworks.com/help/deeplearning/ref/view.html;jsessionid=bd77484ba149c98d4d410abed983

3 NN-SVG
http://alexlenail.me/NN-SVG/LeNet.html

4 graphcore
https://www.graphcore.ai/posts/what-does-machine-learning-look-like

5 graphviz
http://www.graphviz.org/

6 Keras
https://keras.io/api/utils/model_plotting_utils/Image

FAQ 12 ANN Diagram and Graph

7 neataptic
https://github.com/wagenaartje/neataptic

8 Quiver
https://github.com/keplr-io/quiverImage

9 Keras.js
https://transcranial.github.io/keras-js/#/inception-v3ImageImage

10 Netscope CNN Analyzer
http://dgschwend.github.io/netscope/quickstart.htmlImage

11 keras-sequential-ascii
https://github.com/stared/keras-sequential-ascii/

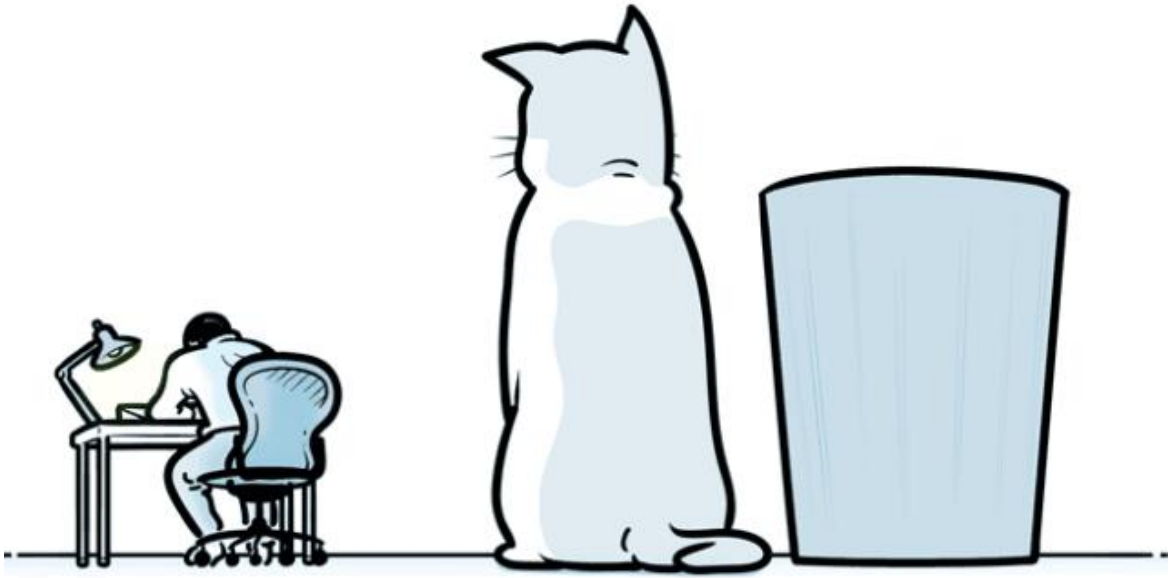
FAQ 12 ANN Diagram and Graph

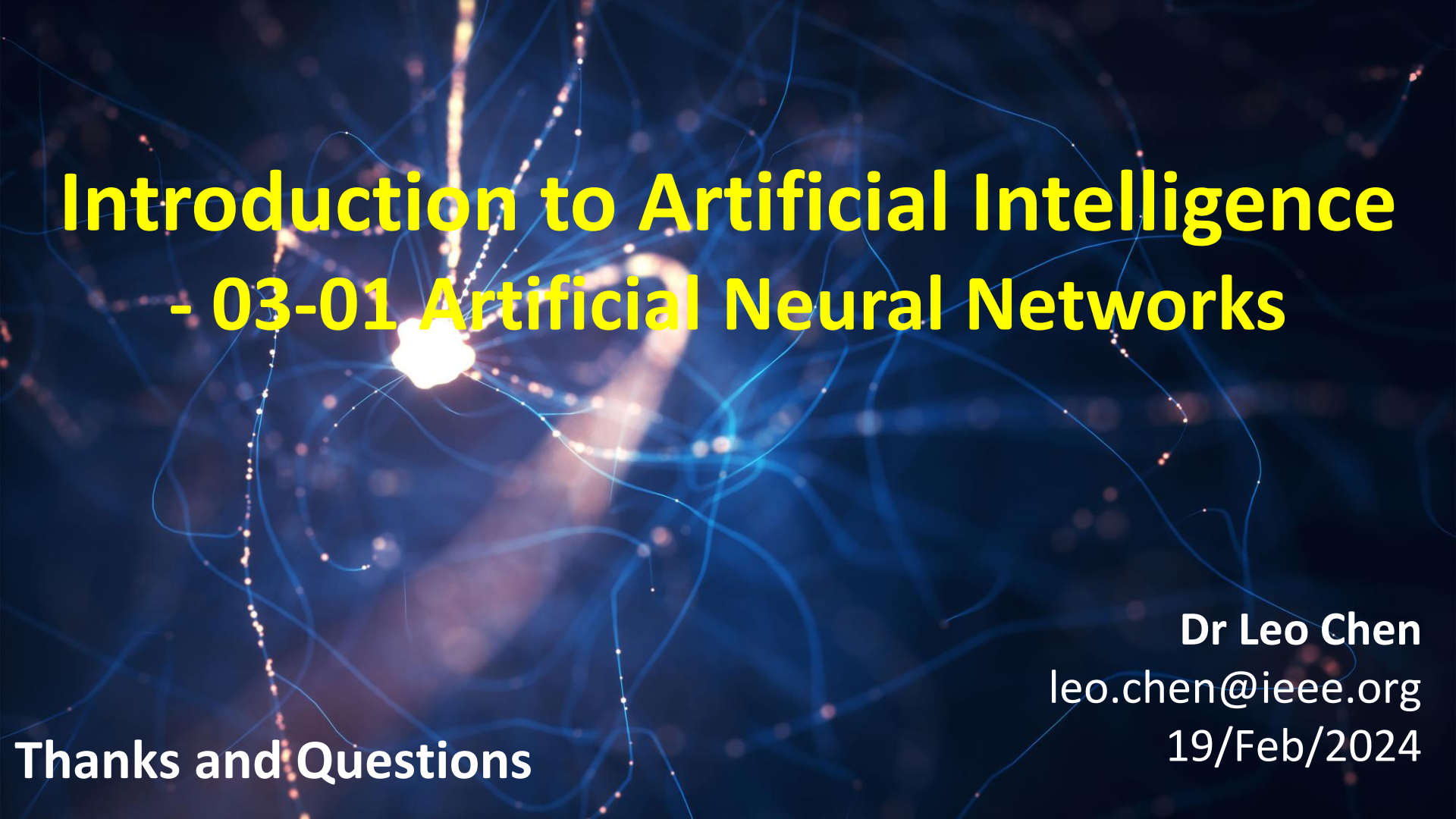
12 TensorBoard
https://www.tensorflow.org/tensorboard/graphs

13 Caffe
https://github.com/BVLC/caffe/blob/master/python/caffe/draw.pyImage

14 TensorSpace
https://tensorspace.org/

Thanks and Questions



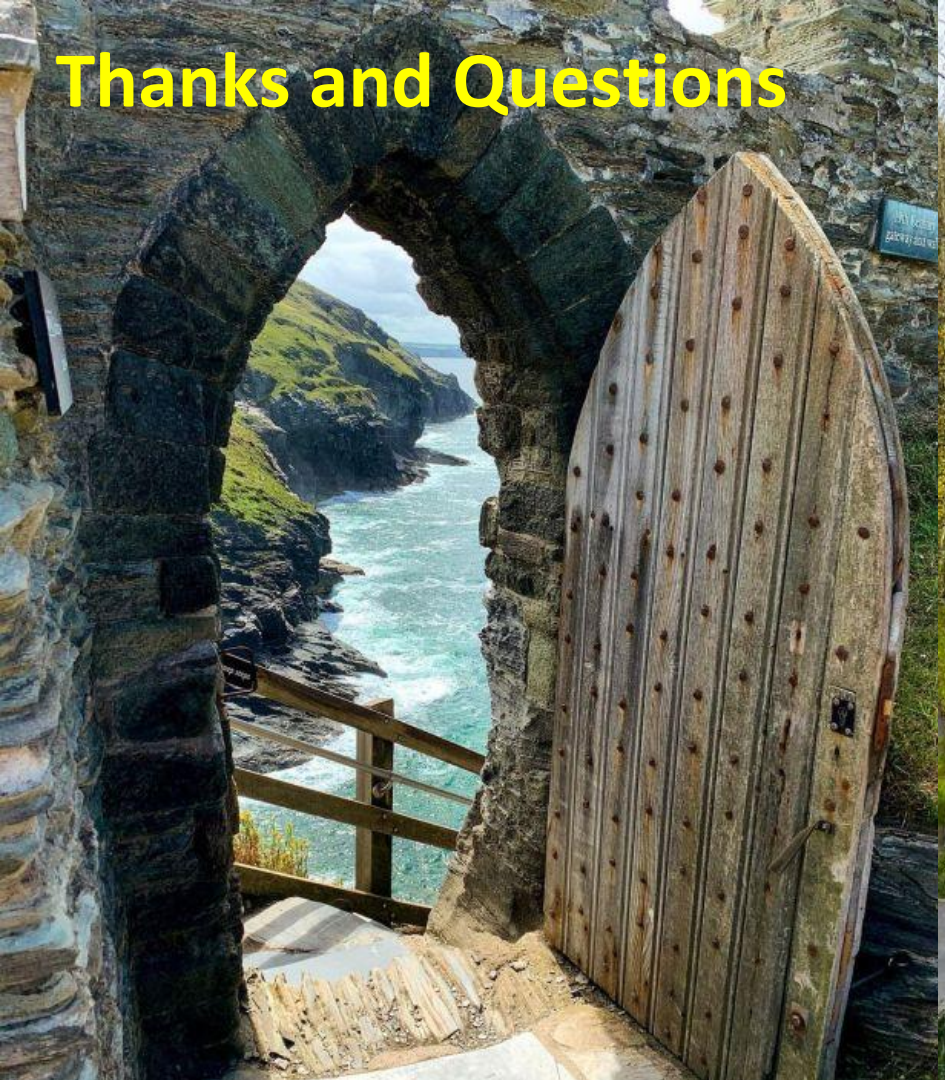


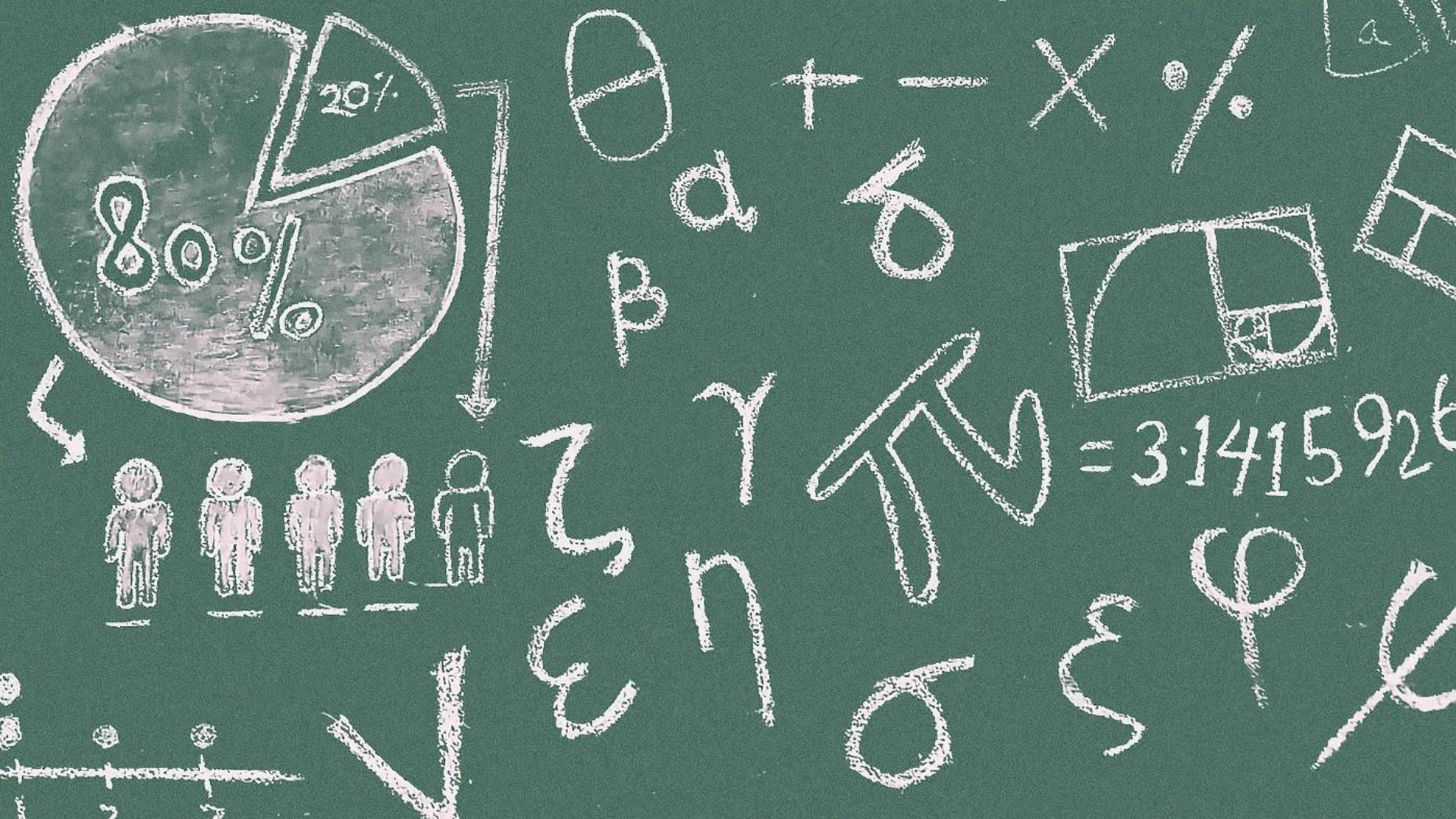
Introduction to Artificial Intelligence
- 03-01 Artificial Neural Networks

Dr Leo Chen
leo.chen@ieee.org
19/Feb/2024

Thanks and Questions

Thanks and Questions




20%
80%
= 3·1415926