VICTORIA UNIVERSITY OF WELLINGTON TE HERENGA WAKA

AIML231/DATA302 —Techniques in Machine Learning

Week 8 Neural Networks (1)

Introduction to Neural Networks

Dr Qi Chen

School of Engineering and Computer Science

Victoria University of Wellington

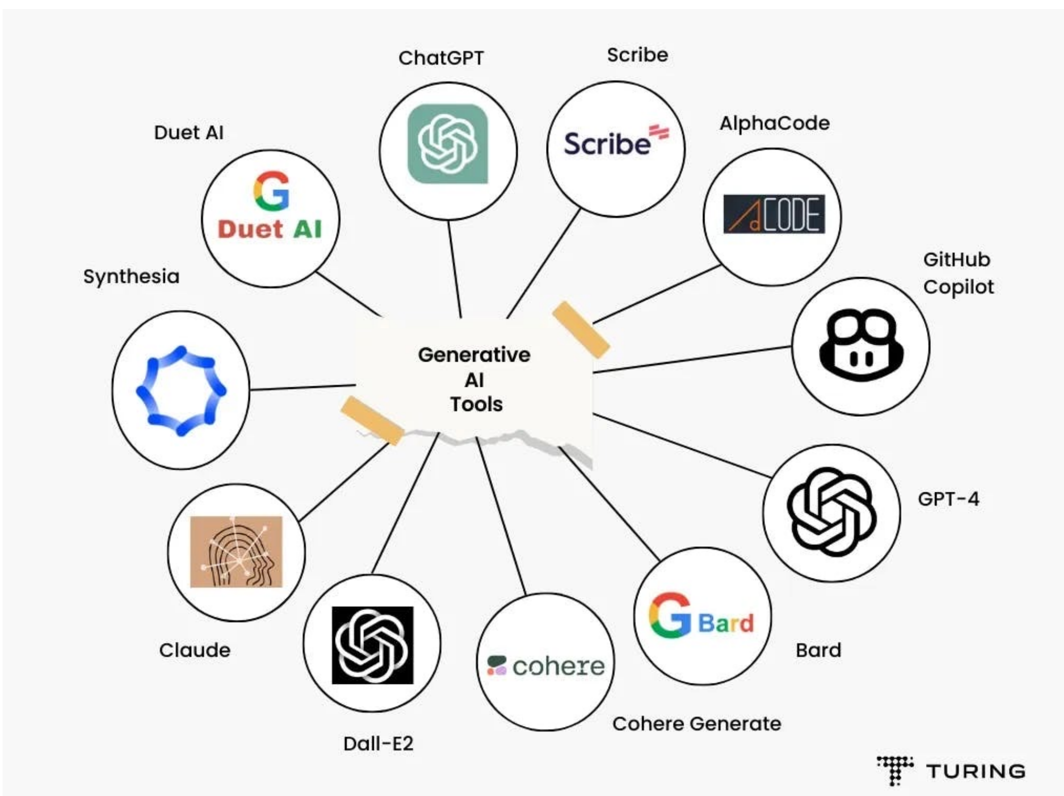
Qi.Chen@vuw.ac.nz

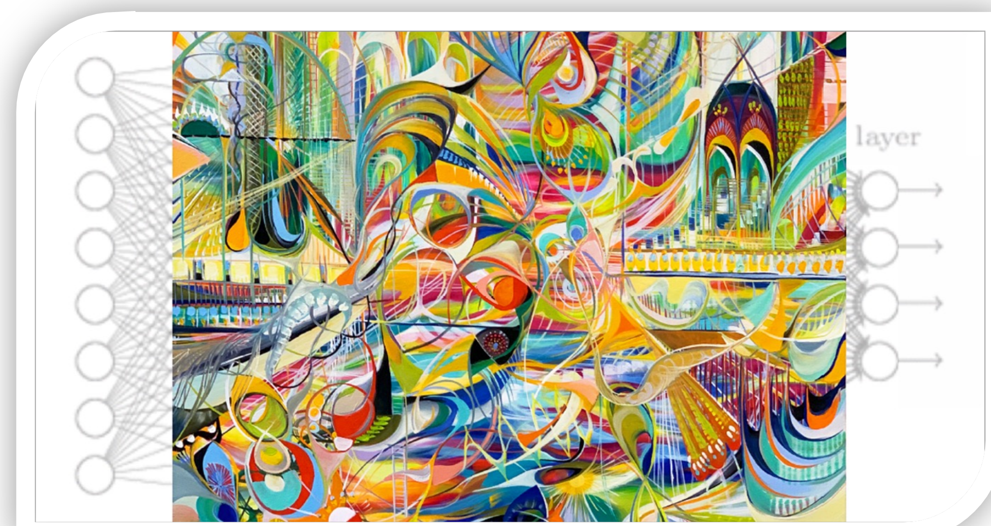
Outline

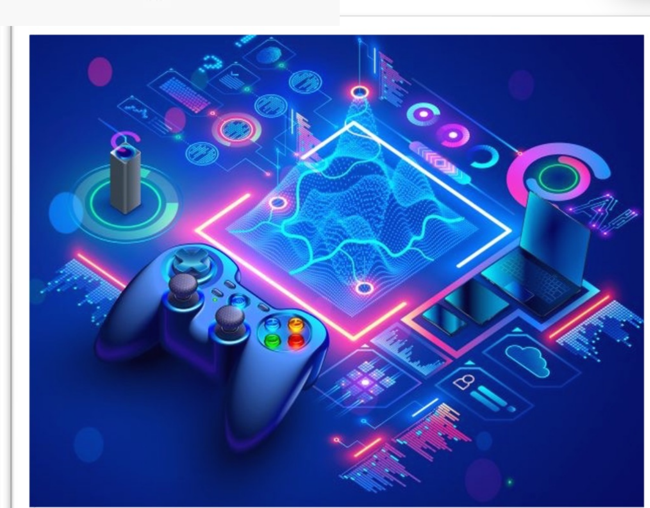
- Neural Network Introduction
- Basic Concepts in NNs
  - Perceptron
  - Neurons, Layers, Weights and Bias
  - NN Architectures
  - Activation functions
- How NNs work – Forward Propagation

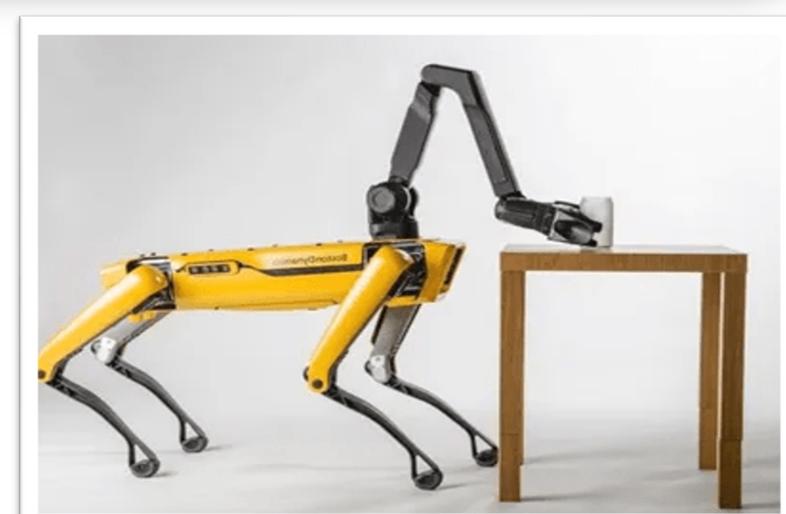
Why Neural Networks

- Many successful applications
  - Generative models: Large language models, Image/video generation
  - Image processing: medical/game/self-driving vehicles





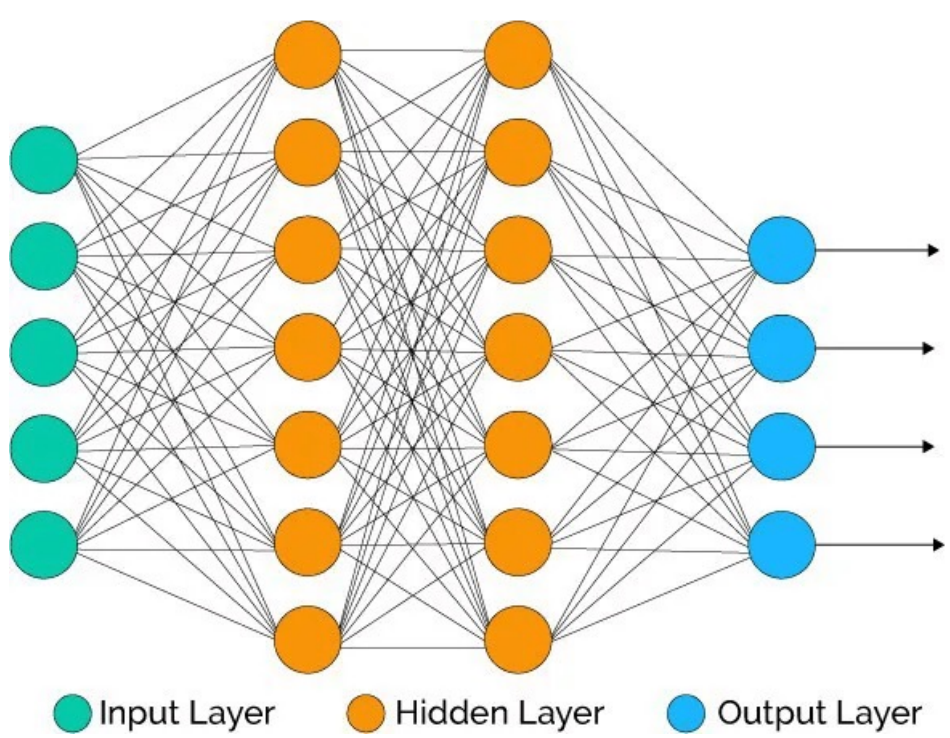


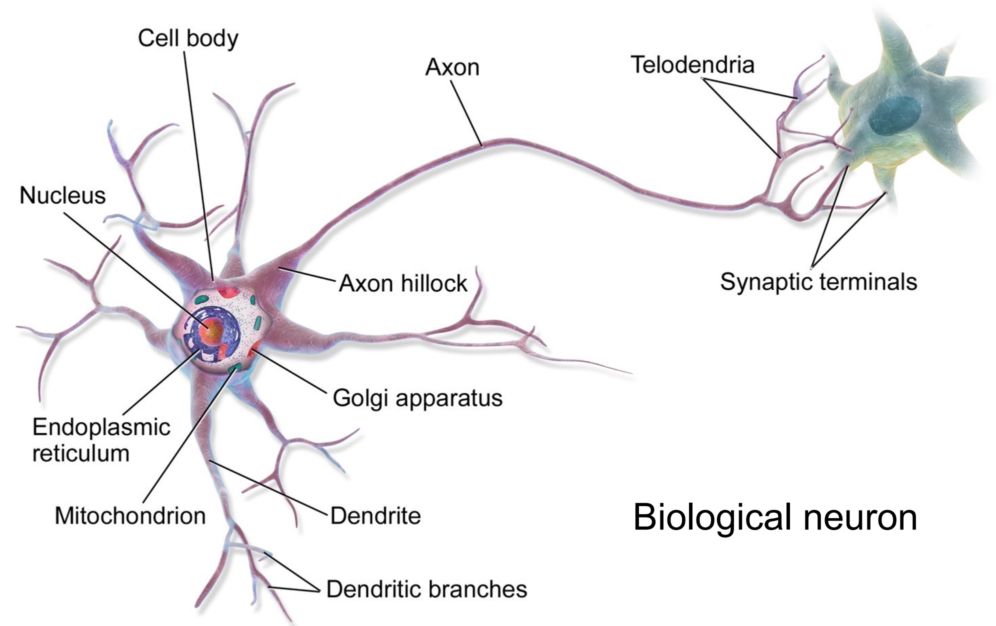


https://www.turing.com/resources/generative-ai-tools

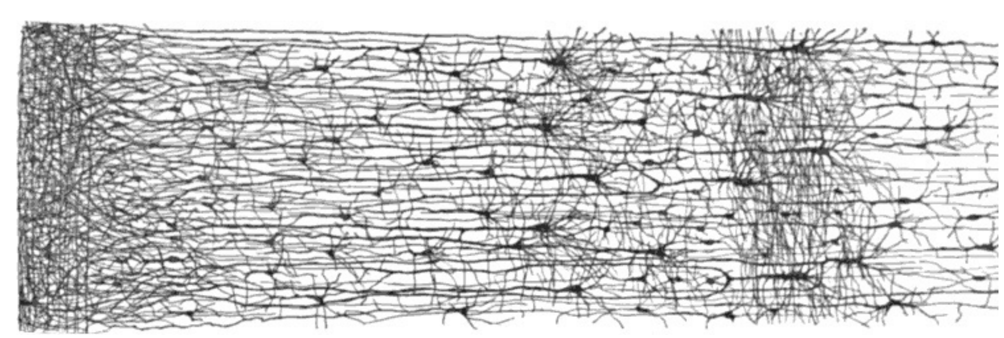
Artificial Neural Networks

- A Neural Network is a machine learning (ML) model designed to recognize patterns and solve complex problems in a manner that mimics human thought processes
  - inspired by the structure and function of the human brain
  - consists of layers of interconnected nodes, or "neurons," each of which performs simple computations





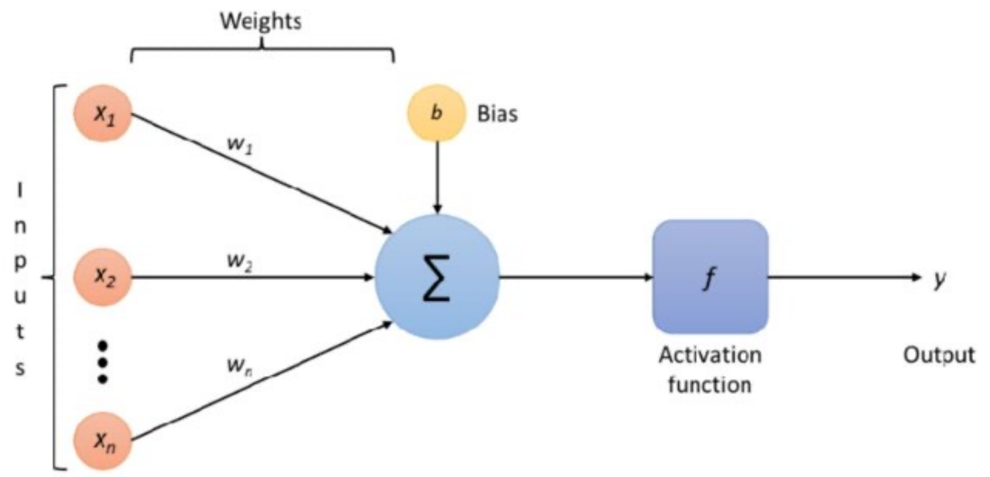
Biological neuron



Biological neural network

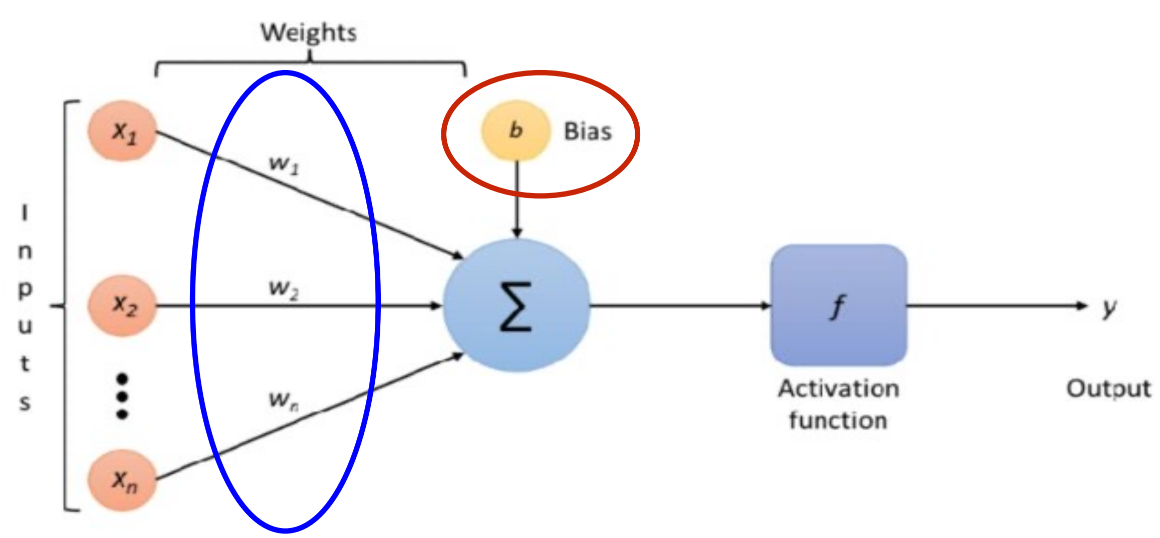
(Artificial) Neurons

- **Neurons** are the basic units of a neural network
  - also also referred to as a node or a unit,
  - the basic unit of computation, as a processing point for carrying out specific calculations
  - The primary function of a neuron in a neural network is to receive input, processes it, and generates output
  - the input x is transformed into an output y using weights W and a bias b as: $y = f(WX + b)$ where f is the activation function



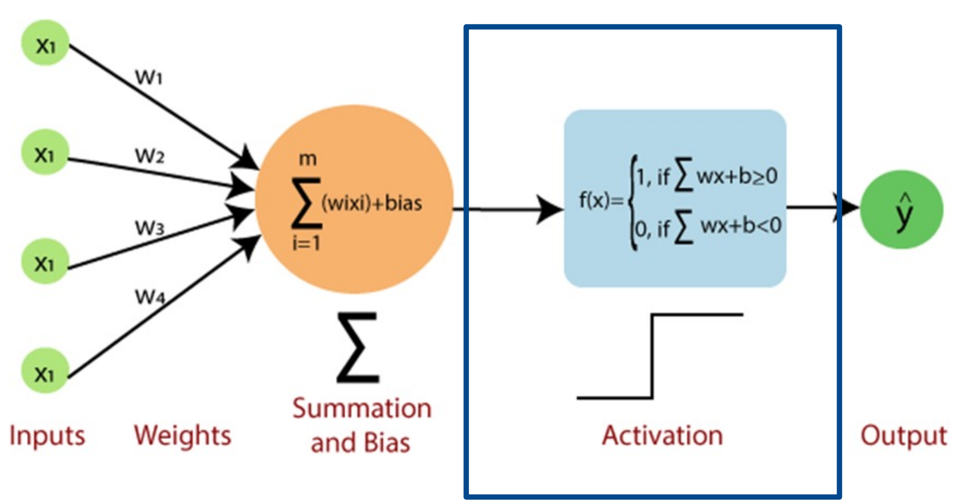
Weights and Bias

- Weights are the numbers assigned to each connection between one neuron in a layer and the neurons from the previous layer
  - each weight is an indication of the strength and influence of the input signals on the neuron's output
- Bias: an additional parameter associated with each neuron that allows the model to better fit the data.
  - provides neuron the flexibility to shift the activation function to the left or right
  - the bias tells you how big that weighted sum needs to be before the neuron gets meaningfully active

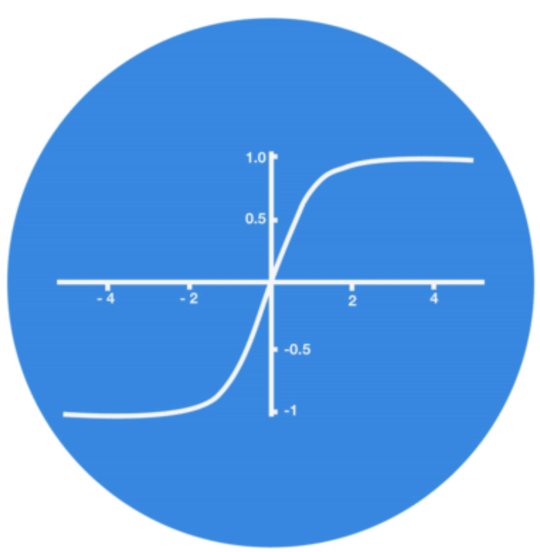


Activation Function

- Activation function is a function that takes the weighted input of a neuron and produces an output, which is used as the input for the next layer of neurons
  - primary role: introduce non-linearities into the network and enable complex learning
  - regulate the output and prevent numerical instability
  - affect the speed, accuracy and generalisation performance



5
0.1
-0.5

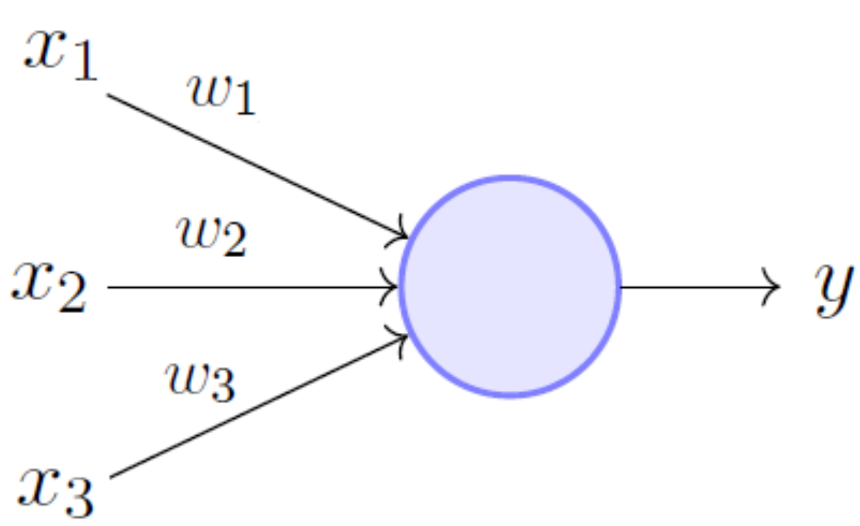


simplest neural network - Perceptron

- **Perceptron:** the simplest form of a neural network

  - it is a binary linear classifier, based on a linear threshold unit

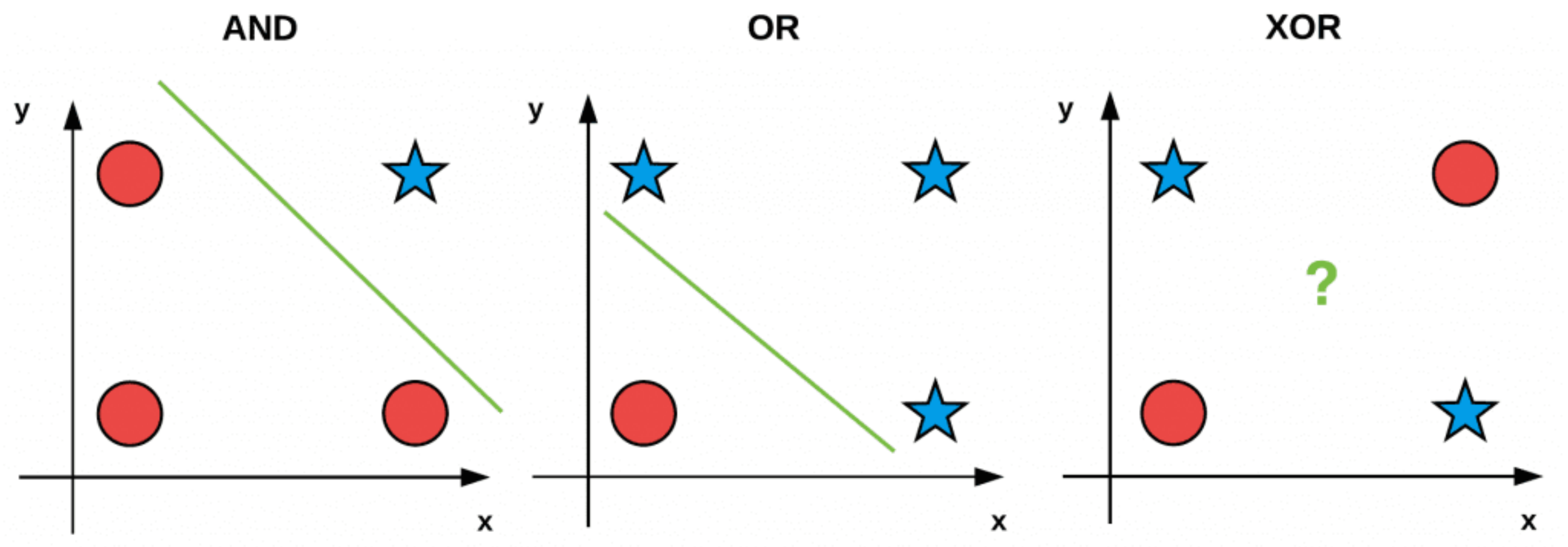
  - calculate a weighted sum of the input features, apply a step function, mathematically represented as $y = Step(WX)$ makes decisions by weighing up evidence



$$\text{output} = \begin{cases} 0 & \text{if } \sum_j w_j x_j \leq \text{threshold} \\ 1 & \text{if } \sum_j w_j x_j > \text{threshold} \end{cases}$$

Problem with Perceptron

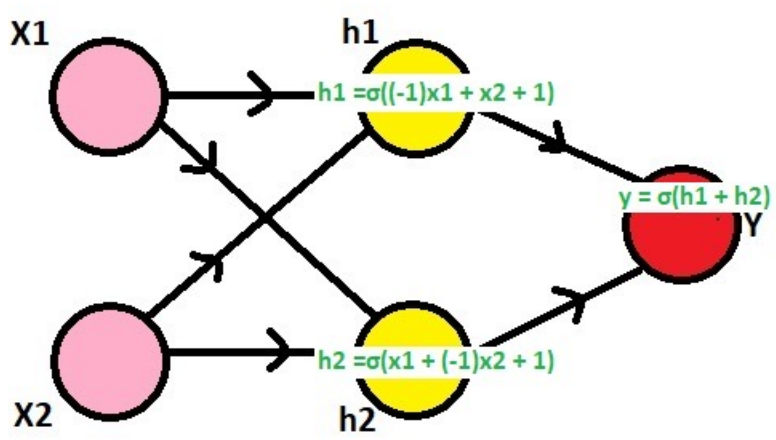
- Famous example of cannot solve the XOR problem (Minsky 1969)



- XOR is not linearly separable, you cannot draw a single straight line that separates the inputs that produce 1 from those that produce 0

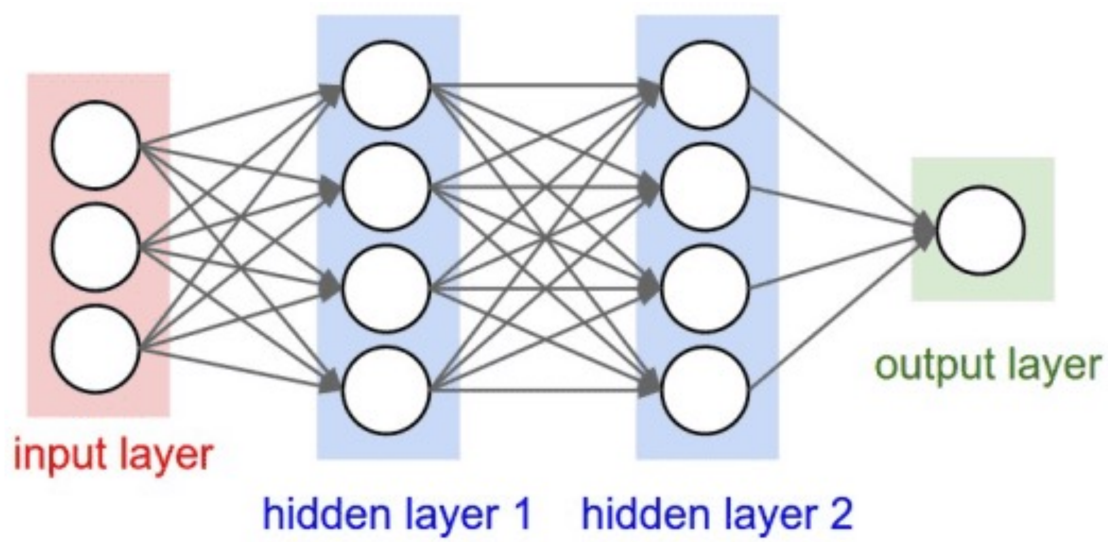
Multi-layer Perceptron

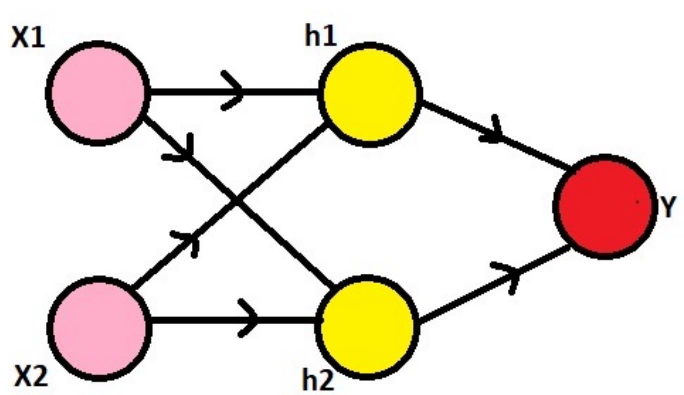
- MLP: a class of feedforward artificial neural network (ANN) that includes multiple layers of nodes
  - all outputs of previous layer connected to all neurons of a layer or the output of a neuron in one layer is fed as input to neurons in the subsequent layers
  - each node typically using a non-linear activation function
  - solve problems that the single-layer perceptron cannot, such as the XOR problem
  - "universal approximation theorem"- can approximate virtually any continuous function



Layers of Neural Networks

- **Input Layer:** The first layer that receives the input signal to be processed, neurons simply pass the data on to the subsequent hidden layers without applying any transformation or computation
- **Hidden Layers:** One or more layers that perform computations through neurons and are not exposed to the input or output directly
- **Output Layer:** The final layer that produces the output of the model





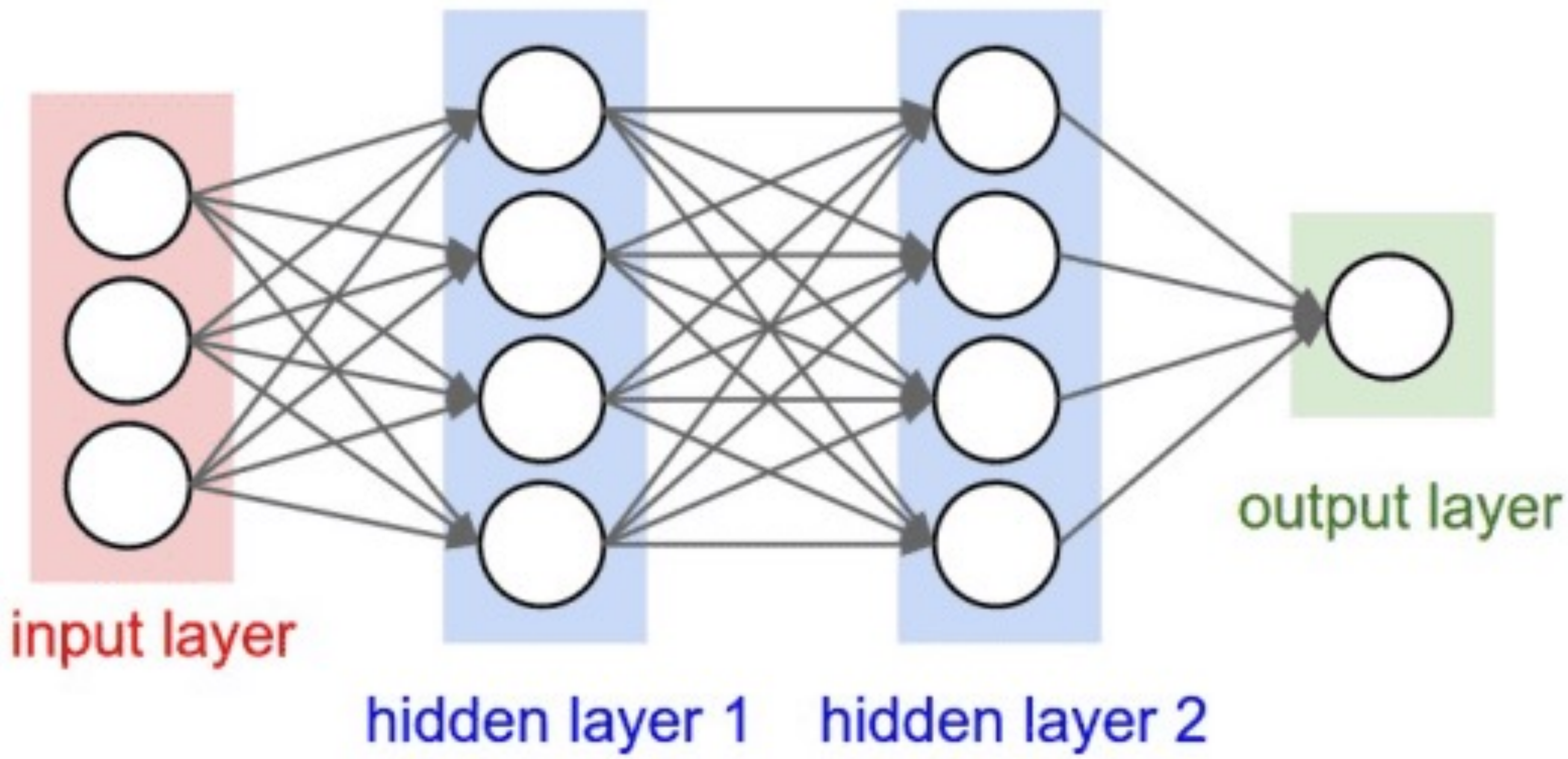
X1 **XOr** X2 = (X1 **AND NOT** X2) **OR** (X2**AND NOT**X1)

- Why layers: layers break problems into bite-sized pieces

Designing layers

- Input layer mirrors the format and structure of input data
  - number of neurons in the input layer is typically equal to the number of features in the input data
  - e.g., with images of size 28x28 pixels, need 784 (28x28) neurons, each representing one pixel value.
- the output layer is closely tied to the specific task
  - for classification tasks, the number of neurons typically corresponds to the number of classes, e.g., in a task to identify digits from 0 to 9, the output layer would have 10 neurons,
  - For regression tasks, the output layer usually contains a single neuron
- it can be quite an art to the design of the hidden layers
  - determining #hidden layers and #neuron in each layer
  - increasing the number of hidden layers can enable the network to learn more complex patterns and features in the data
  - use heuristics-how to trade-off the number of hidden layers against the time required to train the network, trade-off between underfitting and overfitting

Numbers of Weights and Bias in MLP



In a MLP,

- each neuron in a given layer has as many weights as there are neurons in the preceding layer
- each neuron typically has one bias

How many weights and bias in this feedforward NN?

#Weights =3X4+4X4+4=32

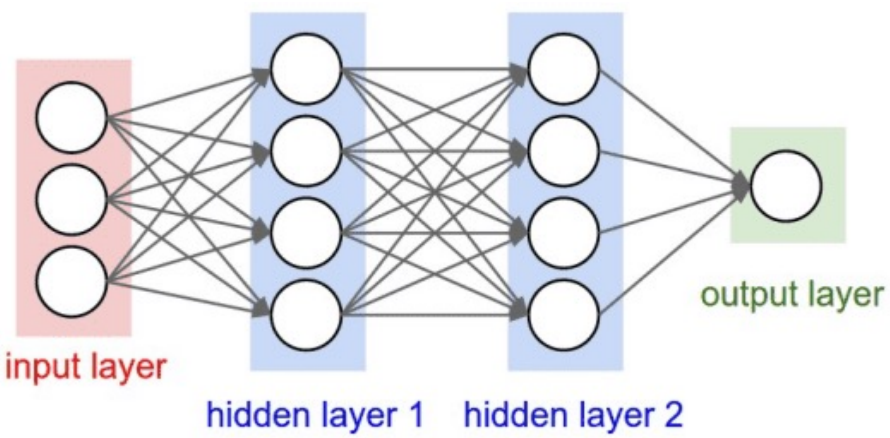
#Bias = 4+4+1=9

In total =32 + 9 = 41

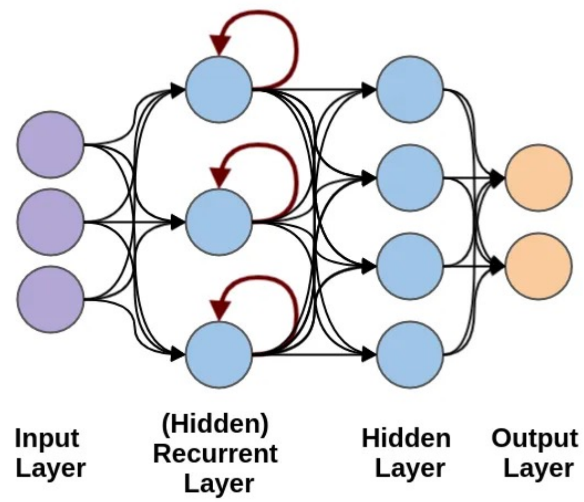
Neural network architectures

- Various NN architectures designed for specific types of tasks

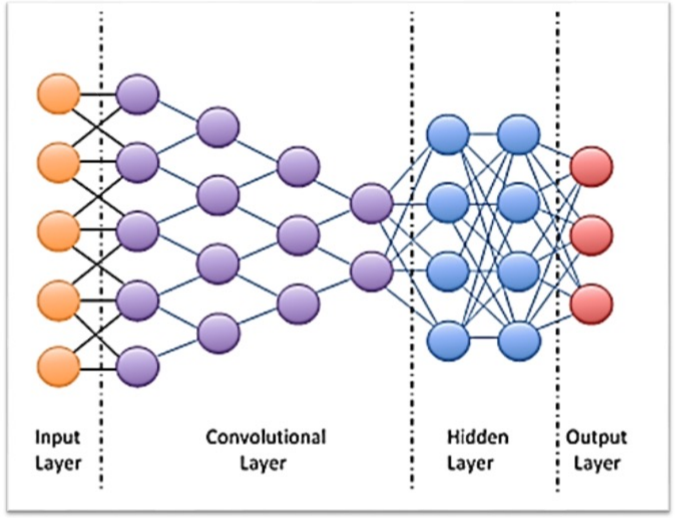
MLP

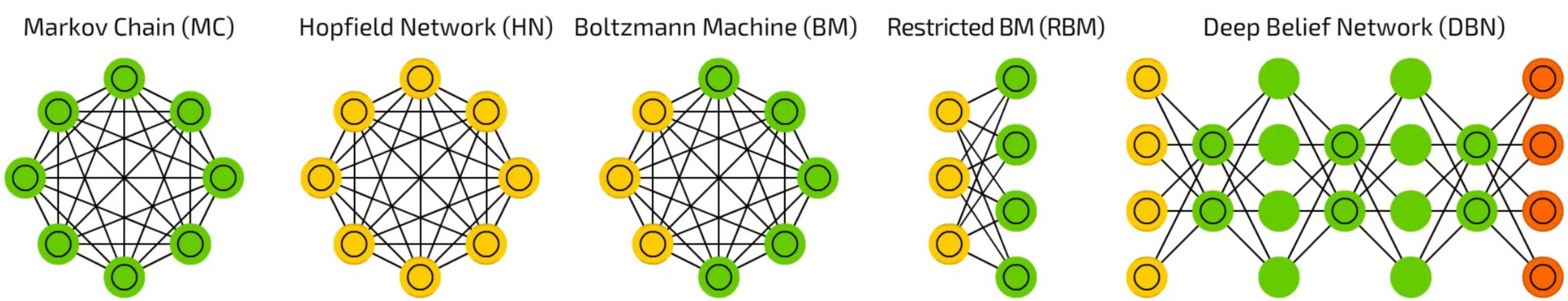


Recurrent NN



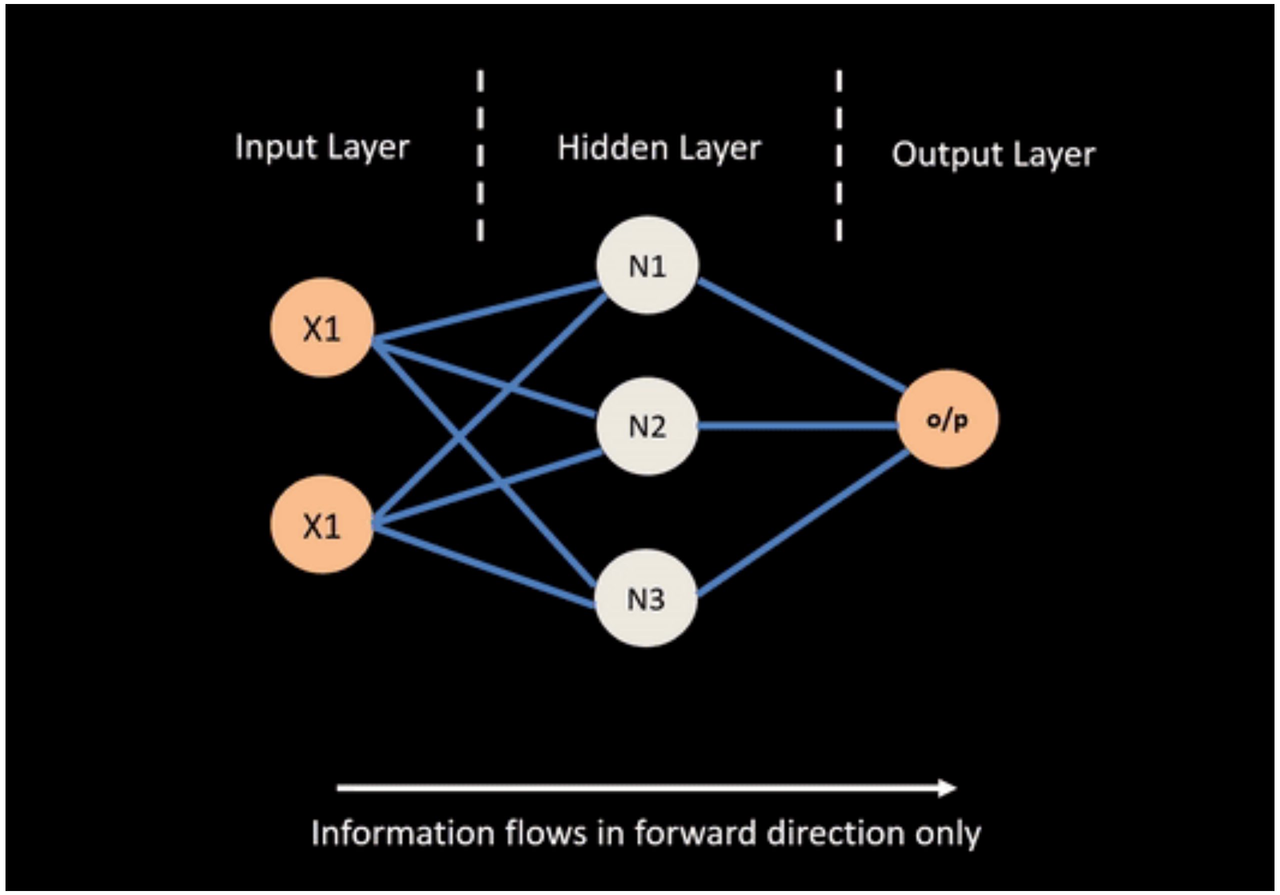
Convolutional NN





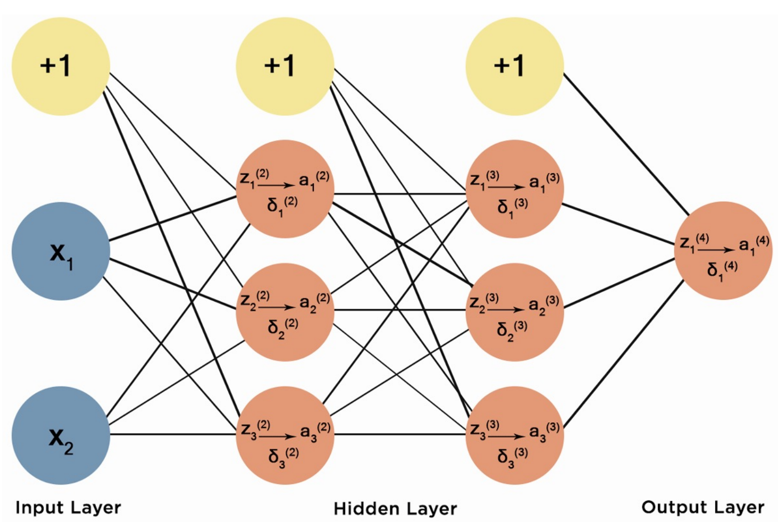
Forward Propagation

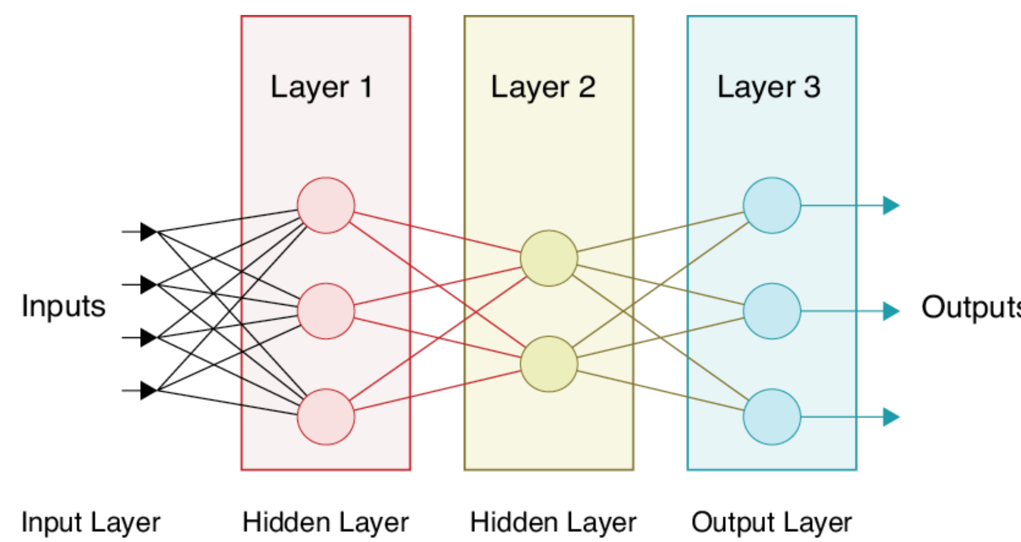
- input data is passed forward through the network to generate an output/prediction
  - This sequence of operations is crucial for both training the network and making predictions/inference



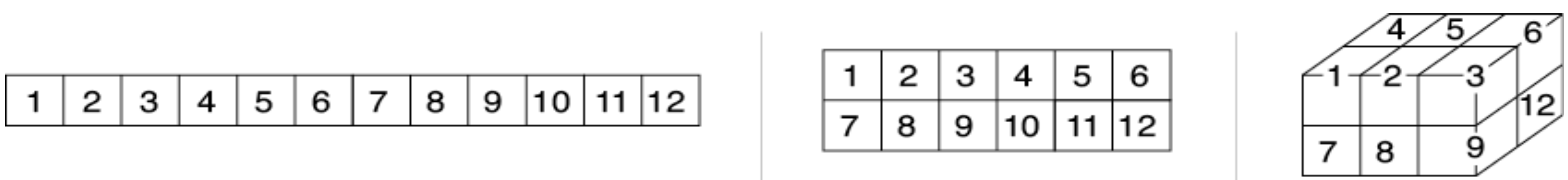
Tensor

- A tensor is a multidimensional array that represents a collection of numerical values
  - a block of numbers with a given number of dimensions and a size in each dimension
  - the fundamental data structure used to store and operate on data in various machine learning frameworks and libraries, such as TensorFlow, PyTorch, and NumPy.





Tensors:

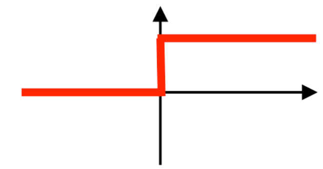
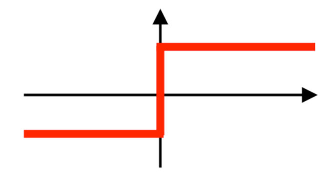
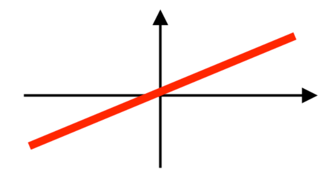
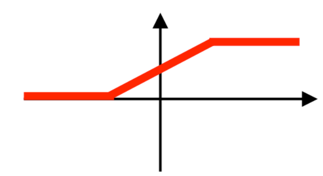
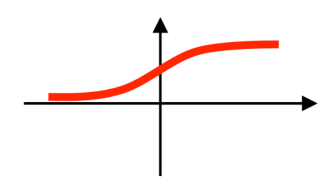
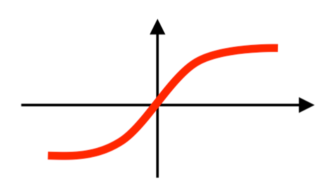
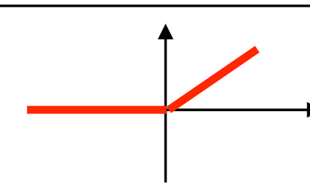
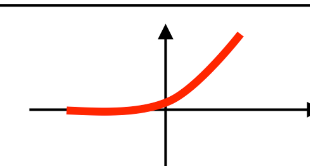


(a) **Array/list/vector**

(b) **Grid/Matrix**

(c) **Volume/Block**

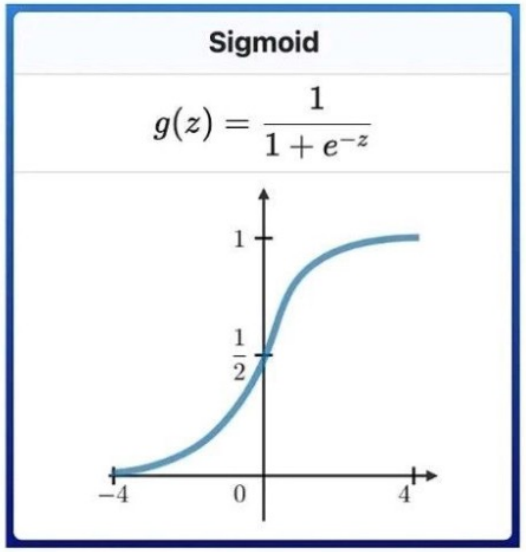
Activation Functions

Unit step (Heaviside)	$\phi(z) = \begin{cases} 0, & z < 0, \\ 0.5, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Sign (Signum)	$\phi(z) = \begin{cases} -1, & z < 0, \\ 0, & z = 0, \\ 1, & z > 0, \end{cases}$	Perceptron variant	
Linear	$\phi(z) = z$	Adaline, linear regression	
Piece-wise linear	$\phi(z) = \begin{cases} 1, & z \geq \frac{1}{2}, \\ z + \frac{1}{2}, & -\frac{1}{2} < z < \frac{1}{2}, \\ 0, & z \leq -\frac{1}{2}, \end{cases}$	Support vector machine	
Logistic (sigmoid)	$\phi(z) = \frac{1}{1 + e^{-z}}$	Logistic regression, Multi-layer NN	
Hyperbolic tangent	$\phi(z) = \frac{e^{z} - e^{-z}}{e^{z} + e^{-z}}$	Multi-layer Neural Networks	
Rectifier, ReLU (Rectified Linear Unit)	$\phi(z) = max(0, z)$	Multi-layer Neural Networks	
Rectifier, softplus	$\phi(z) = \ln(1 + e^{z})$	Multi-layer Neural Networks	

Activation Functions

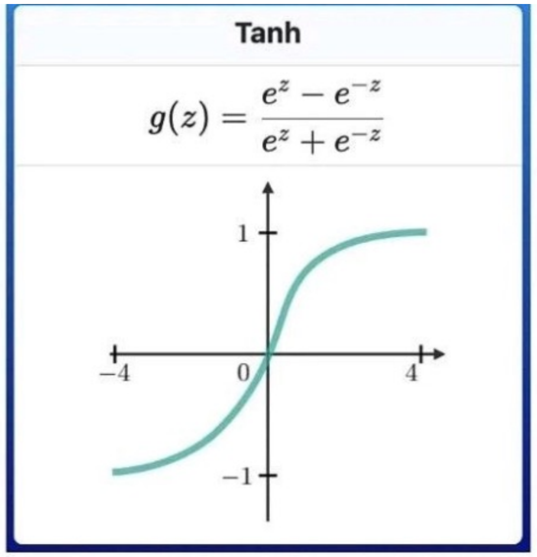
- **Sigmoid Function**

  AKA the Logistic function, maps input values to the range (0, 1) using a smooth S-shaped curve, making it suitable for binary classification tasks
  - output can be interpreted as a probability
  - nonlinear activation function



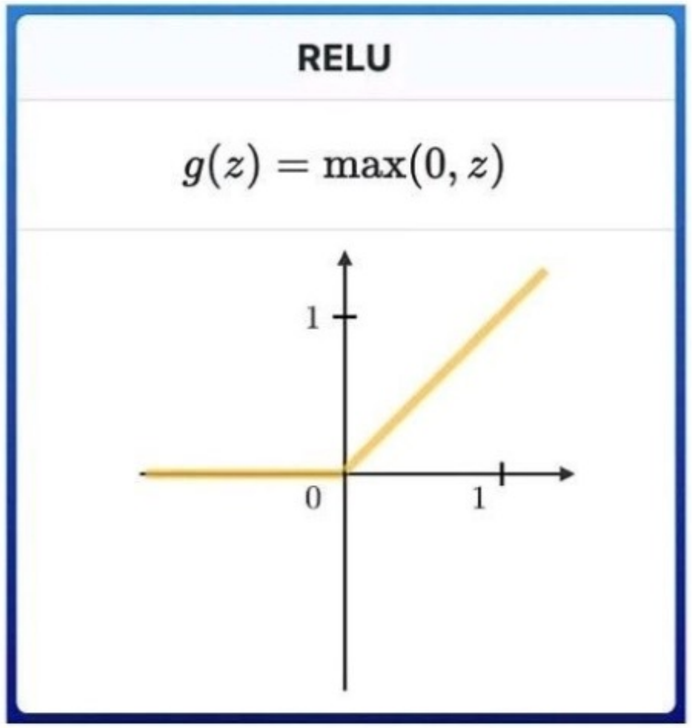
- **Tanh (Hyperbolic Tangent) Function**

  similar to the sigmoid but maps input values to (-1, 1)
  - providing a better symmetry around zero



Activation Functions

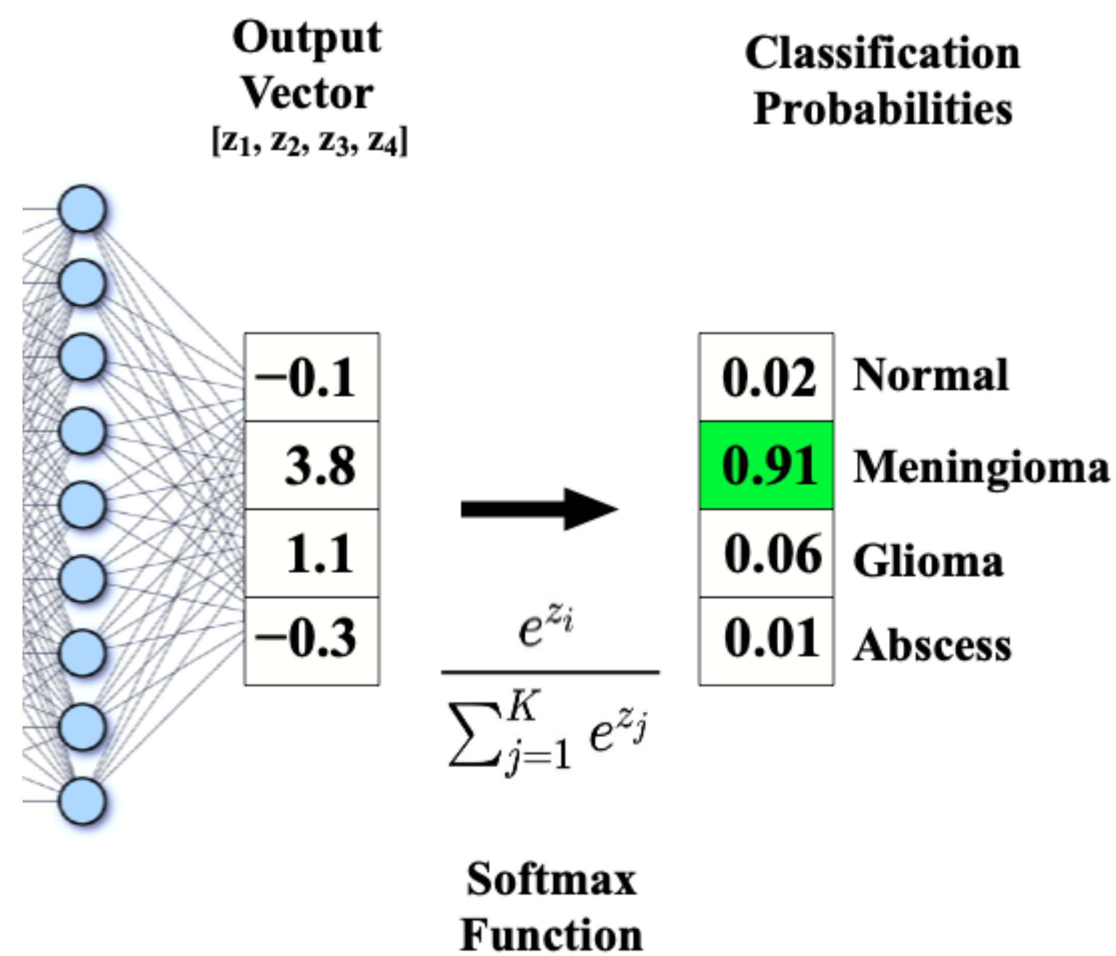
- **ReLU (Rectified Linear Unit) Function:** a nonlinear function that returns the input if it is positive, and zero otherwise, $\max(0, z)$
  - one of the most widely used activation functions due to simplicity and effectiveness
  - encourages sparse representations by activating only a subset of neurons for any given input



Activation Functions

- **Softmax Function:** maps the input to a probability distribution over a set of possible outcomes, using an exponential function
  - convert a vector of numbers into a new vector that reflects the probability distribution of the original vector's values
  - commonly used in the output layer of a neural network for multi-class classification tasks
  - computationally expensive

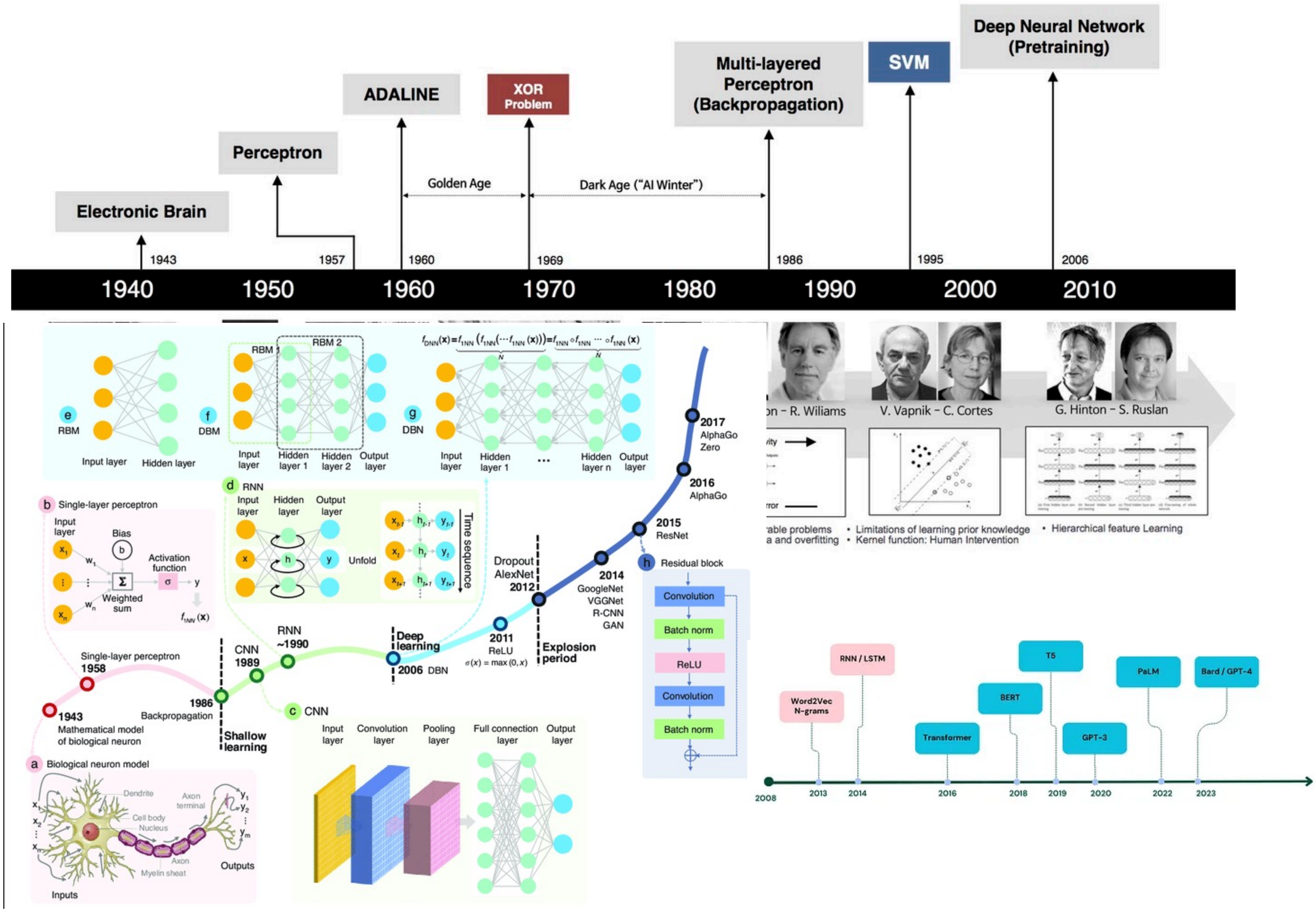
$$s\left(x_i\right) = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$



Which Activation function to use

- different activation functions may work better or worse for different problems, architectures, and datasets
- no definitive answer or rule for selecting an activation function
- some general guidelines
  - **type of problem**: binary classification uses a sigmoid activation function, multi-class classification softmax activation function
  - **architecture of the neural network:** may need a different activation function for the hidden layers of your neural network, deep neural network-> a ReLU activation function, while a recurrent neural network-> a tanh activation function
  - **properties of the activation function:** depending on the properties of the activation function, such as the range, the slope, the smoothness, the sparsity, etc.

History of Neural Networks



https://www.linkedin.com/pulse/history-neural-networks-datta-dharanikota/

Summary

- What is Neural Networks
- Why Neural Networks
- Neural network basic components
  - Neurons, Weights and bias, Activation Function
- Types of Neural Networks
- Neural Network Working Mechanisms
  - Various Activation functions
  - Forward Propagation