Faculty of Engineering and Technology

School of Computer Science and Mathematics

# 7144COMP Deep Learning Concepts and Techniques

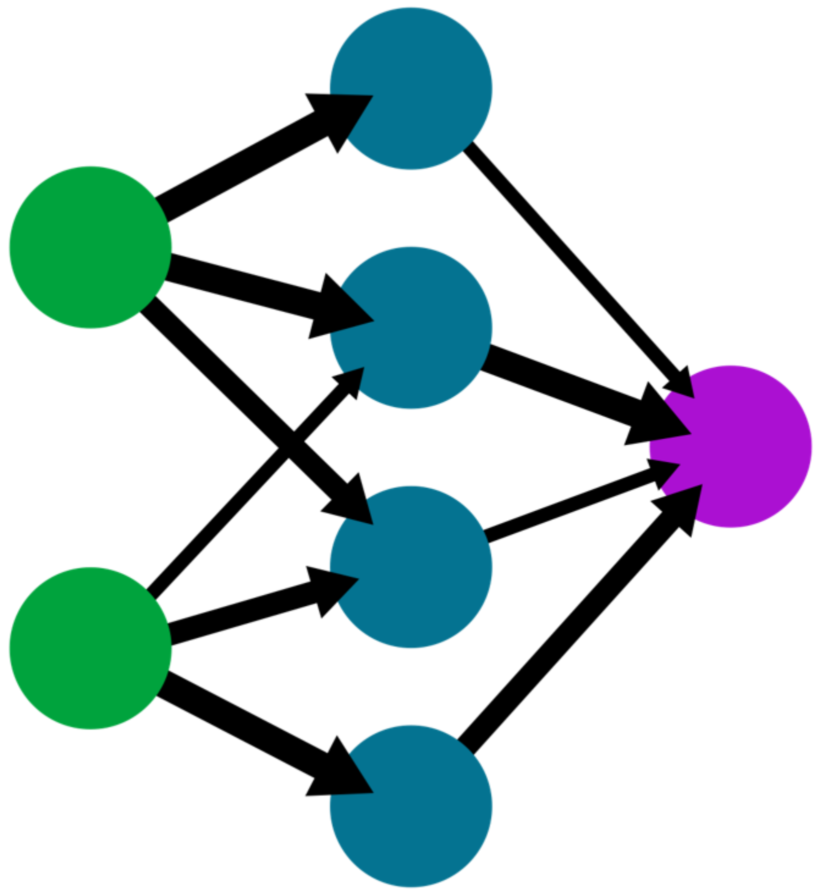
Dr Paul Fergus, Dr Carl Chalmers

**{p.fergus, c.chalmers}@ljmu.ac.uk**

Room 714, 629 Byrom Street

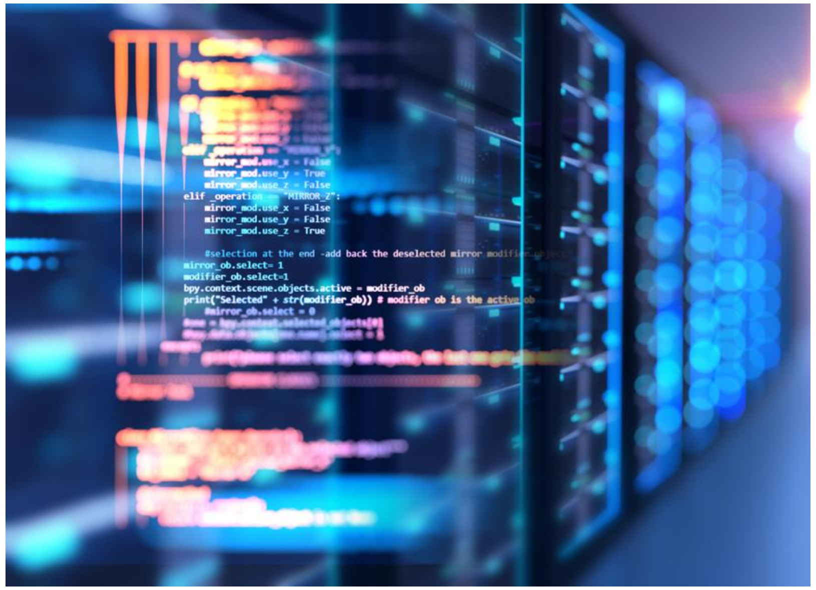
# Lecture 2

Artificial Neural Networks



## In this session…

- We will cover:
    - ANN’s and their biological comparison
    - Perceptions
    - Neural Networks (MLP)
    - ANN types and uses (CNN’S, Feed forward etc.)
    - Activation Functions
    - Multi-Class Classification Considerations

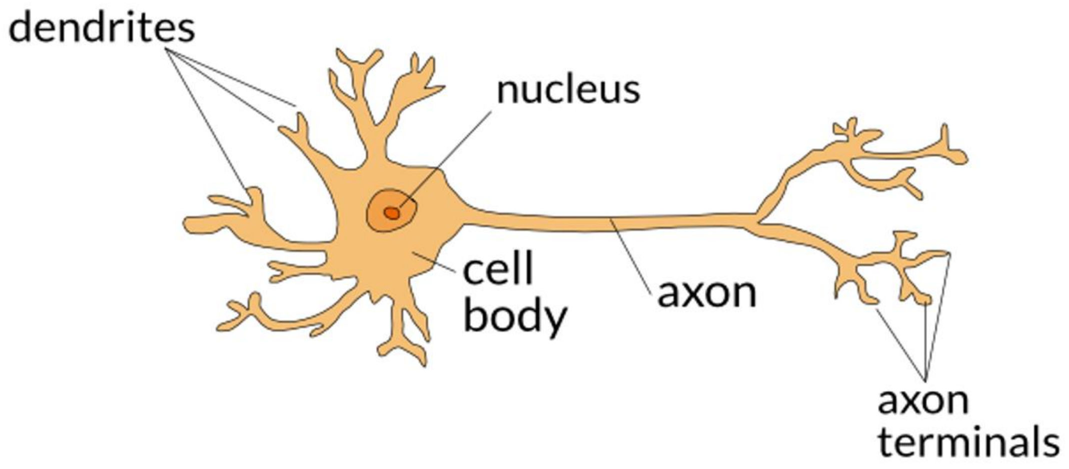


## Artificial Neural Networks

- Neural networks are modelled after biological neural networks and attempt to enable computers to learn in a similar manner to humans
- In human brain there are around 86 billion neurons with more than 1000 trillion synapsis connections
- In contrast the biggest ANN at the time of writing contained 16 million neurons (perceptron’s)
- However more typically the number you encounter will be in the hundreds and thousands

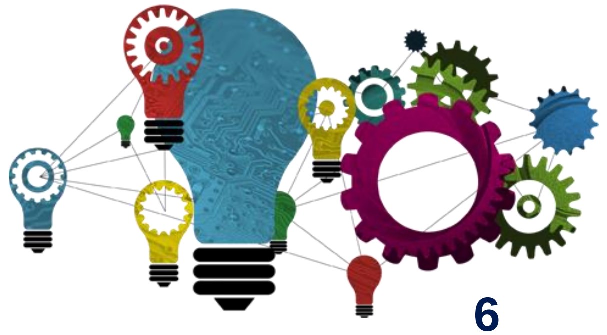
## Artificial Neural Networks

- Perceptron's which can be considered as artificial neurons can mimic certain parts of their biological counter parts
- These parts included dendrites, cell bodies and axons
- Signals are received from the dendrites and transmitted via the axon once enough of them have been received
- This signal is then used as an input for other neurons which continually repeats



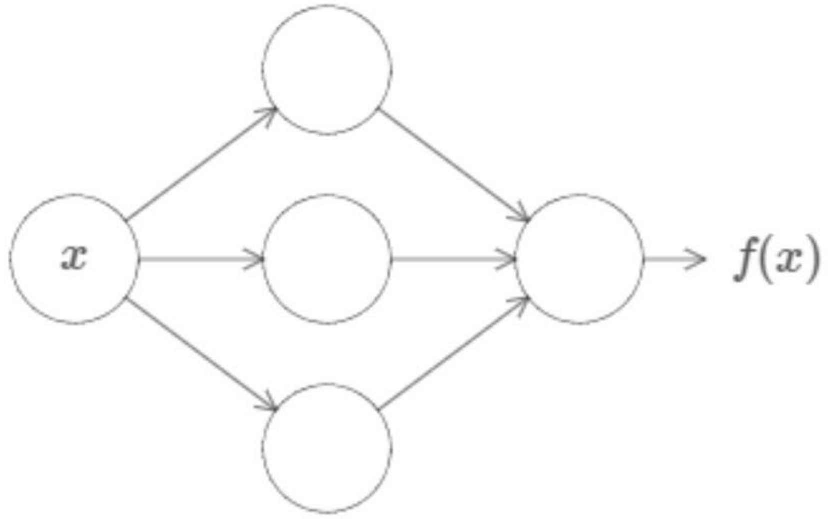
## Artificial Neural Networks

- Some signals are considered to be more important that others which causes other connected neurons to fire easier
- Connections can become stronger or weaker while new connections can be formed or disappear completely (this is also known as plasticity)
- An ANN can mimic a significant part of this process by modelling a function using as set of weighted inputs which triggers an activation in the neuron (perceptron) if the nodes activation threshold is reached (i.e. the neuron fires)
- Once it fires some of output is passed to the next neuron

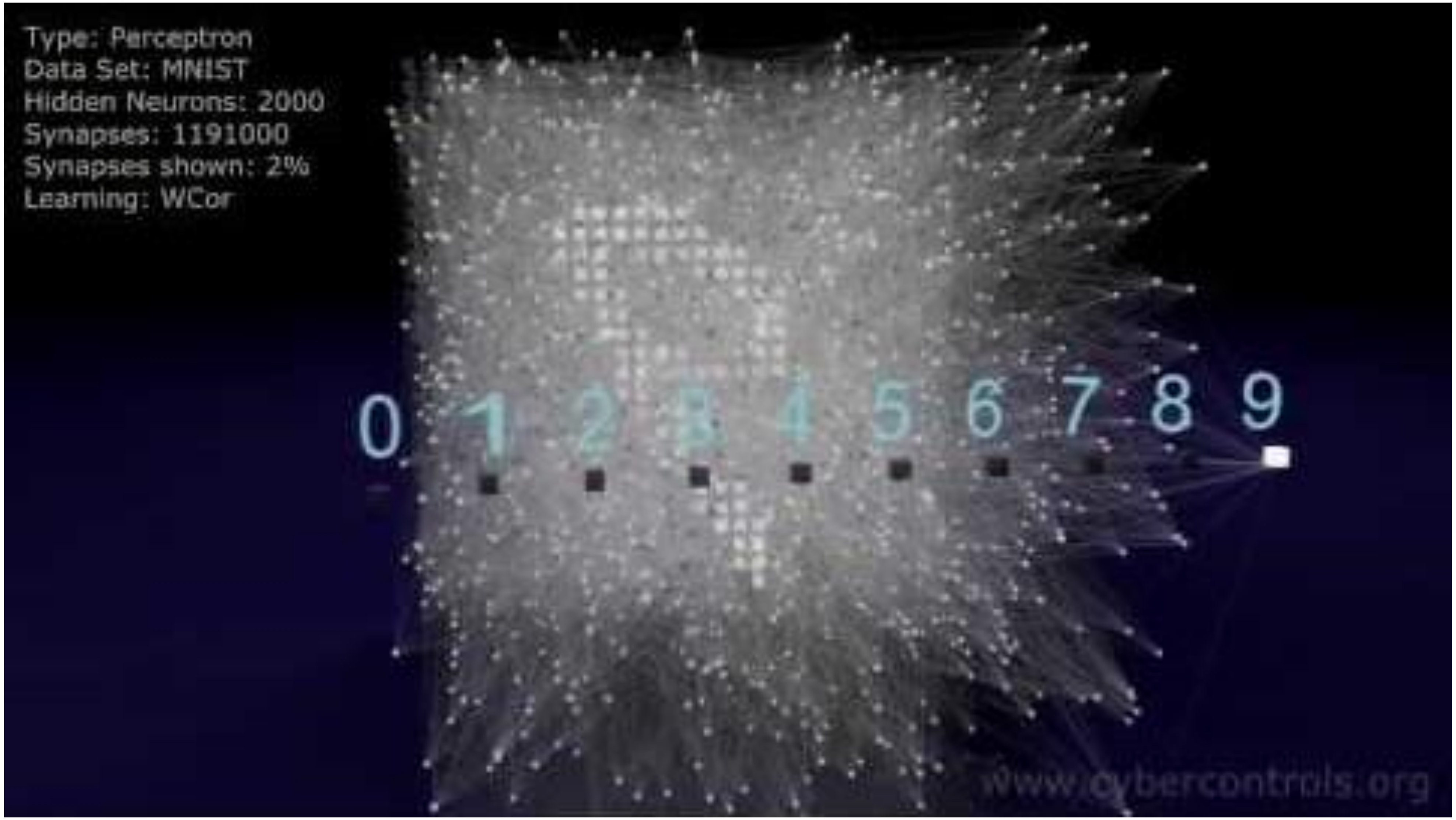


## Artificial Neural Networks

- As before this value is multiplied by the weight of the connection and added to any other connections to the neuron
- As a result the process continues causing activations throughout the network
- Given a set number of neurons ANN can map a set of inputs to a corresponding set of outputs (known as universal approximation)

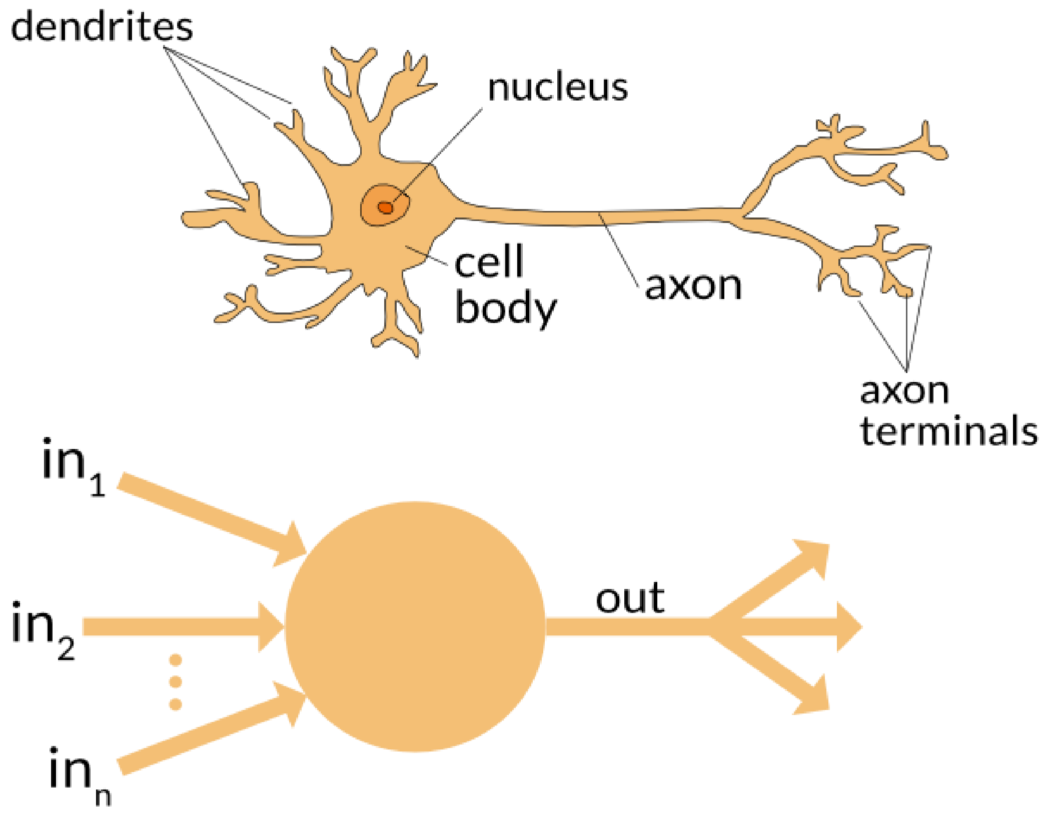


## Artificial Neural Networks

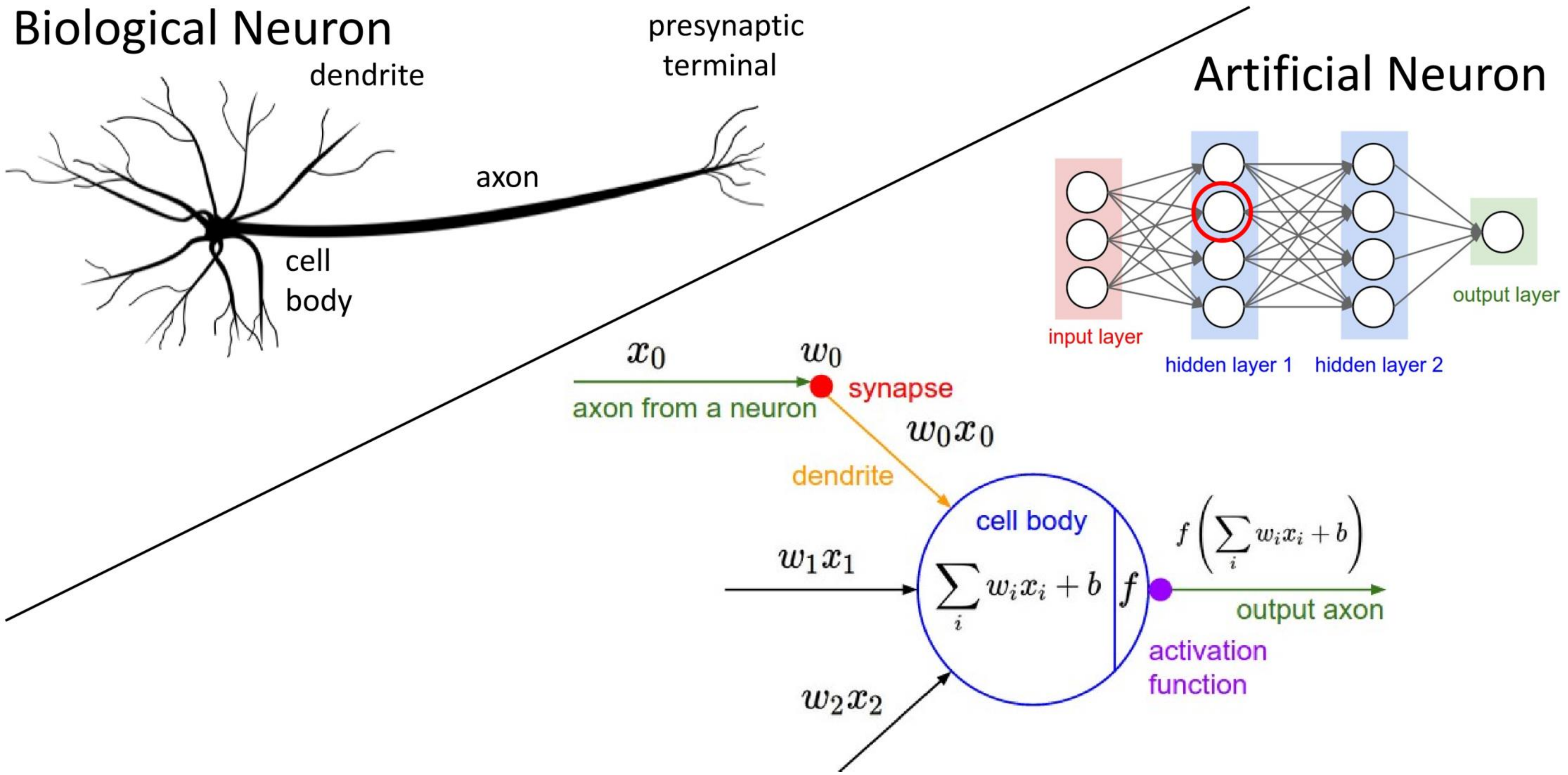


## Perceptron's

- Perceptions were inspired by how our biological neurons work
- A perceptron consists of one or more inputs, a processor, and a single output
- The inputs represent the input data you wish to model, for example, pixel intensities in an image, hot encodings describing words in a text, or signal information obtained from a speech recording or stream

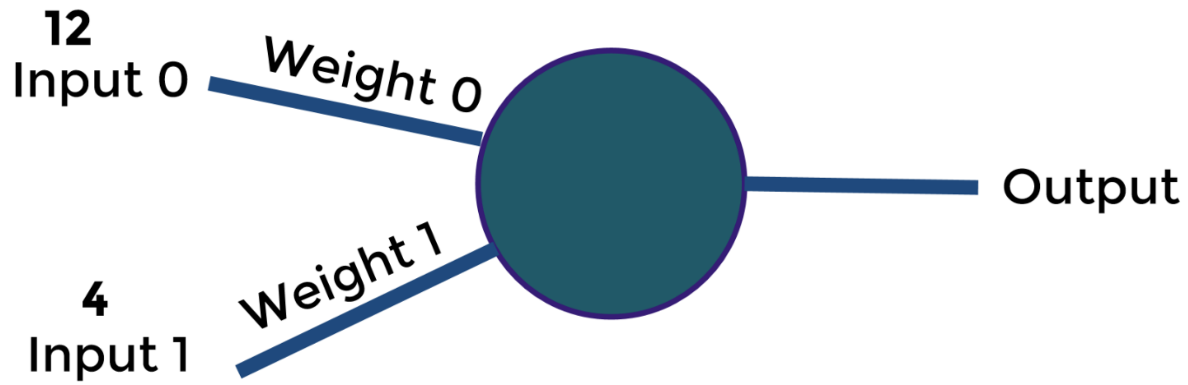


## Perceptron's



## Perceptron's

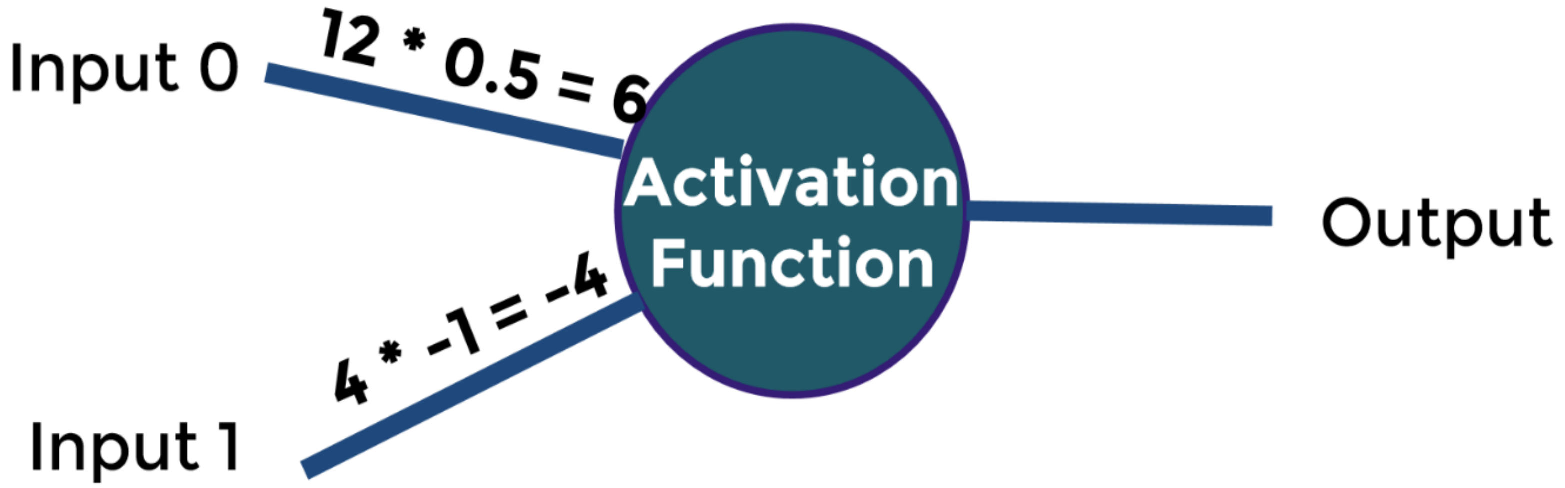
- The processor fires or activates if the sum of the inputs exceeds a given threshold which generates an associated output (dependent on the activation function)
- Therefore, there are four main steps to the process: receive inputs, weight inputs, sum inputs and generate output
- For example, consider below, where we have two inputs 12 and 4. Each input must be weighted (multiplied by a value) which is usually a number between -1 and 1 (initially weights are randomly assigned)



## Perceptron's

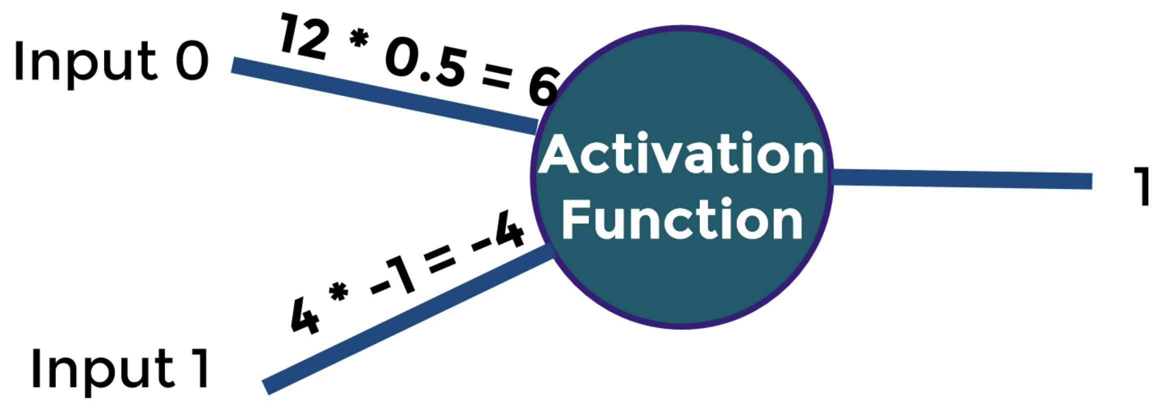
- In a similar way to how we learn, this simple network will adjust these weights automatically through some form of feedback mechanism until a mapping between the inputs and output is found
- These weights are used to describe how important that particular input is
- Once we have our randomly assigned weights, we multiply the inputs by the weights (input * weight)
- The results of the calculation are passed to an activation function (processor)
- In a simple binary output the activation function determines if the perceptron (neuron) activates or not I.e. generates a 0 if the threshold in not reached and a 1 if it is
- There are a wide variety of different activation functions such as Sigmoid, Hyperbolic Tangent etc which will be discussed later in this module

## Perceptron's



## Perceptron's

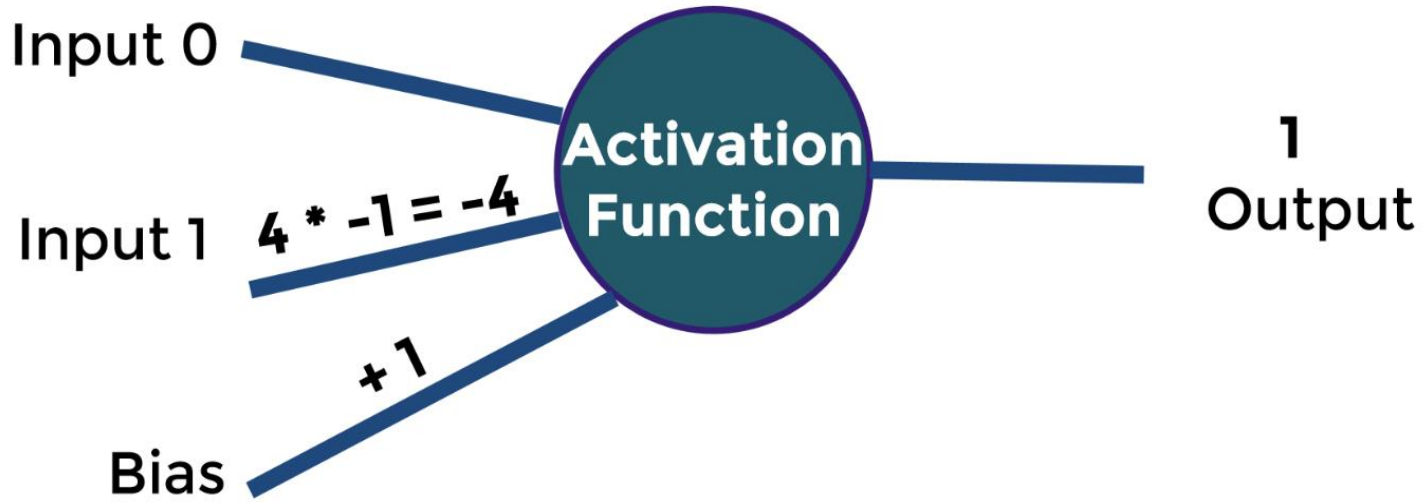
- The most basic activation function works the following way: – If the sum is a positive number the output will be 1 – If the sum is a negative number the output will be 0. So, in our example 6 + -4 = 2 so our activation function would return 1



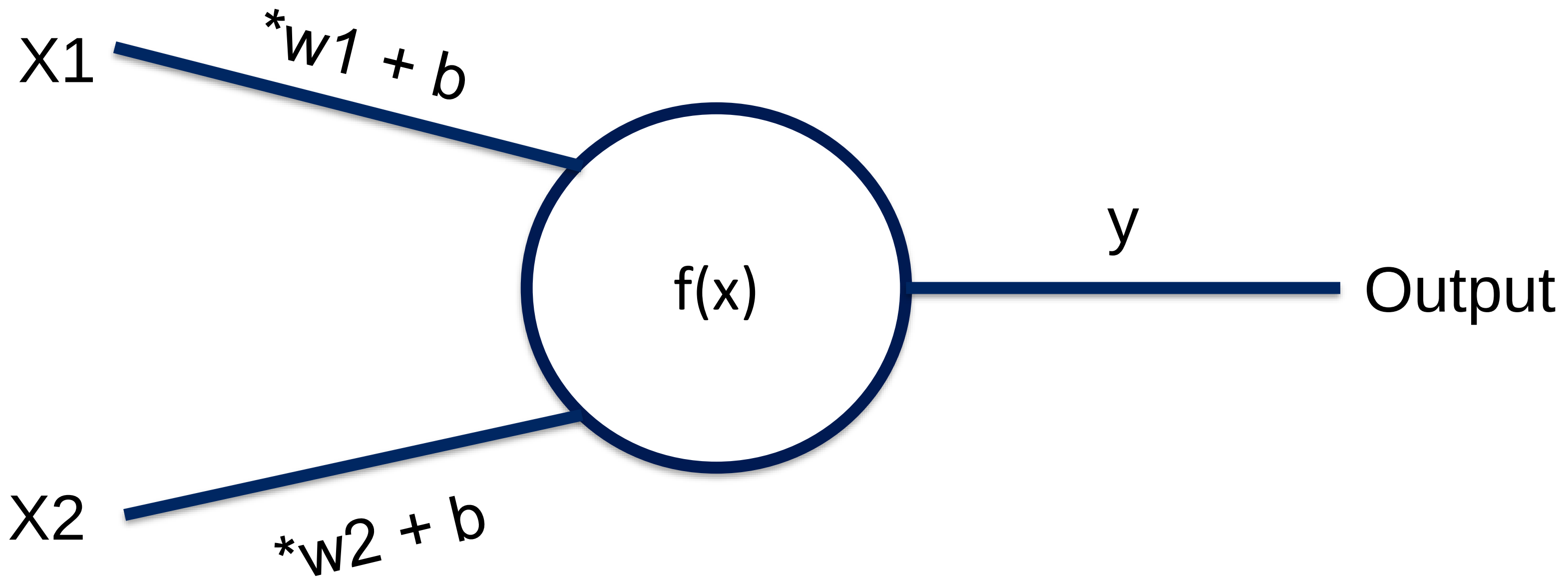
## Perceptron's

- There is an issue with this. What if the initial inputs where both 0? The activation function would never activate
- To solve this, we add what's known as a bias term to each input
- It is important to note that the weight won't start to have an effect until it has surpasses the bias value
- In this sense the bias can act as a suppressant when the weighting is low
- When the weighting surpasses the bias the its effect on the network is correlated with the size of the weight itself (the smaller the weight the smaller the effect, the larger the weight the larger the effect)

## Perceptron's



## Perceptron's

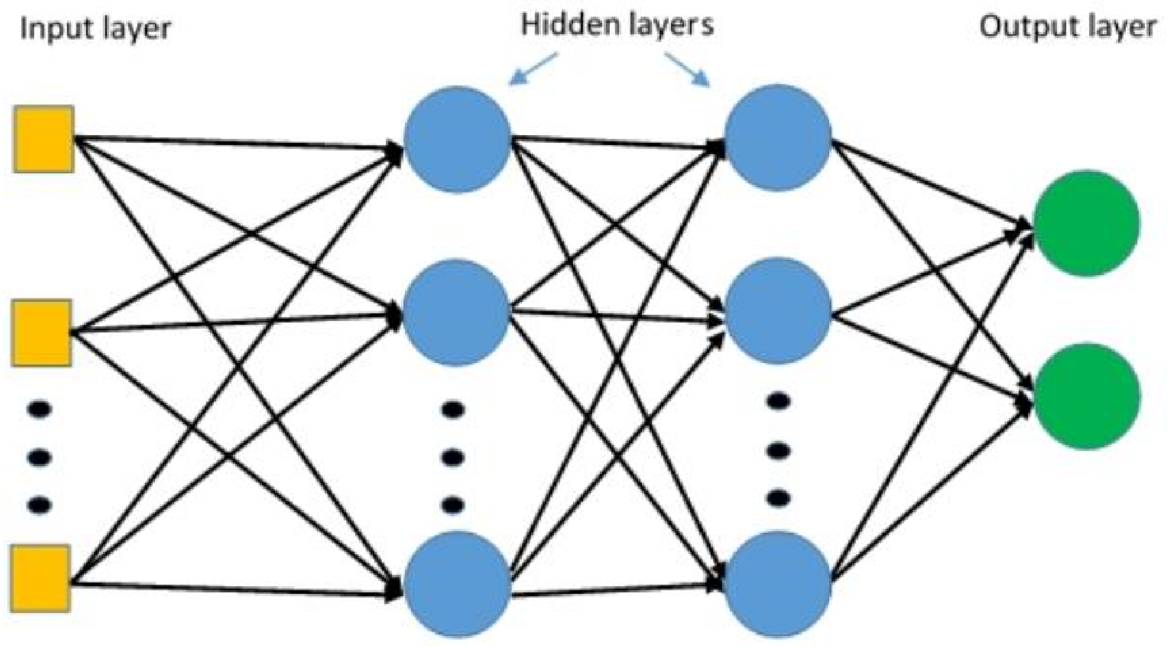


## Neural Networks

- Obviously, a single perceptron wouldn’t be very useful as such these can be expanded to form more complex network structures just like the human brain, ANN consist three primary layer types which includes:
  - **Input Layer** – Real values from the data
  - **Hidden Layers** – Layers in between input and output – 3 or more layers is classified as a “deep network”
  - **Output Layer** – Final estimate of the output
- As you go forward through more layers, the level of abstraction increases, and this is where model interpretation becomes harder

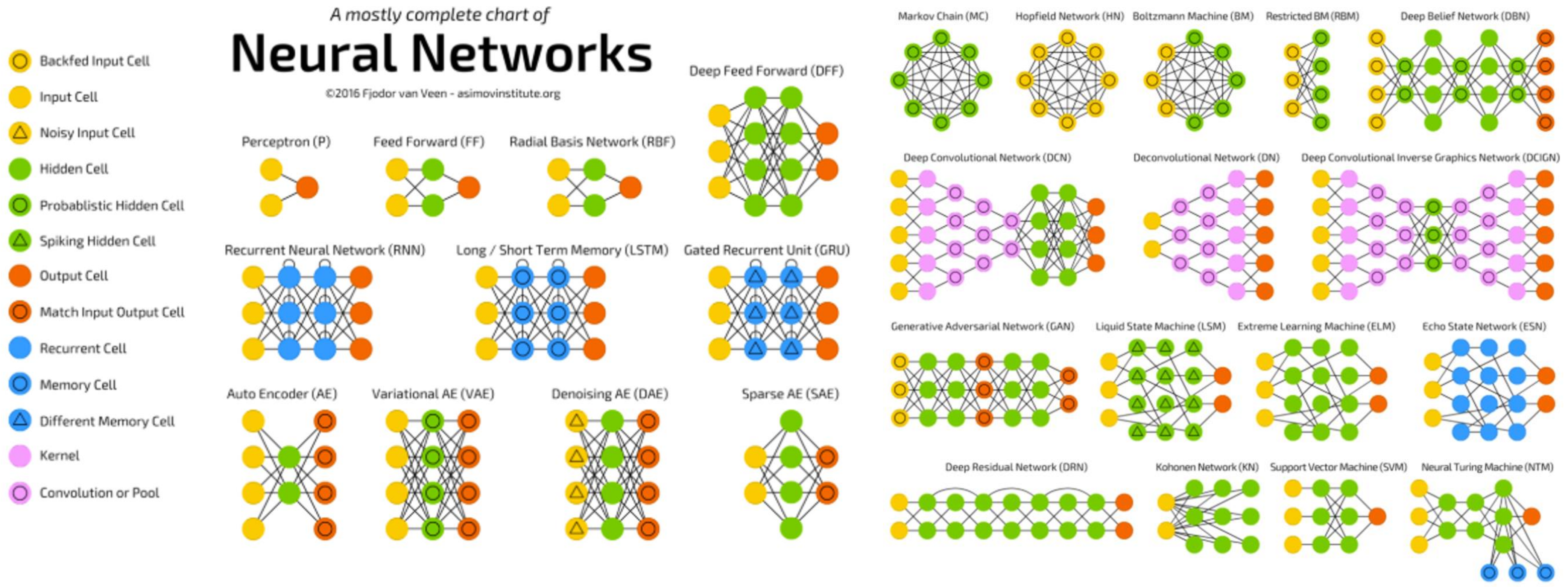
## Multi Layer Perceptron's (MLP)

- These more complex configurations are known as Multilayer Perceptron Model (MLP)
- Each layer provided input to the next layer of preceptors until on output value is received
- An MLP is a fully connected network as every perceptron in a layer is connected to every perceptron in the next layer



# Multi Layer Perceptron's (MLP)

- There are many different variants of an ANN each capable of addressing a particular problem


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

## Neural Networks

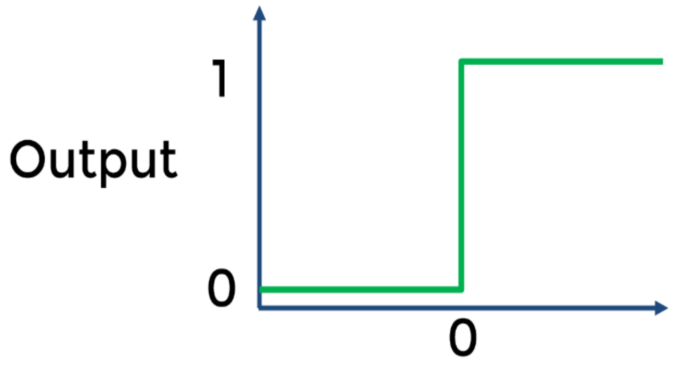
- Obviously, a single perceptron wouldn’t be very useful as such these can be expanded to form more complex network structures just like the human brain, ANN consist three primary layer types which includes:
  - **Input Layer** – Real values from the data
  - **Hidden Layers** – Layers in between input and output – 3 or more layers is classified as a “deep network”
  - **Output Layer** – Final estimate of the output
- As you go forward through more layers, the level of abstraction increases, and this is where model interpretation becomes harder

## Activation Functions

- We have seen how inputs can be fed into an ANN and how these inputs, weights and bias can be calculated and adjusted to allow the network to learn
- In this section we will focus on the processor (perceptron) and understand how a neuron (perceptron) fires
- This is achieved using activation functions and a significant amount of research has been undertaken to understand their effectiveness
- The input, weigh and bias are summed and passed to the activation function. Depending on the value the activation function will either fire or not

## Step Function

- The simplest activation function which is known as the step function
- If the summed value for our inputs, weights and biases produces a negative value the output is 0
- A value of 0 and above will return an output value of 1
- The problem with this function is that it is often too drastic and doesn't take into account small changes

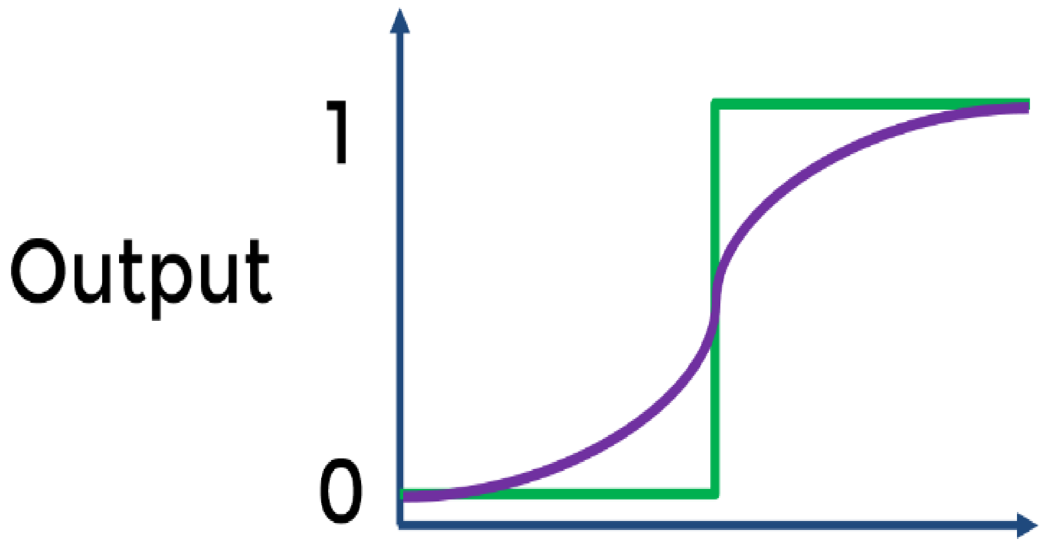


- Step function
- Small changes are not taken into account
- Output 0 or 1

## Sigmoid Function

- This function takes in the summed inputs, weights and bias and like before it returns a value of either 0 or 1
- With a sigmoid activation it allows us be more granular with the cut off were by smaller alterations can change the output of the activation function (so rather than activation being dependent on a negative/non-negative value, activation occurs between 0 and 1 with a threshold of 0.5)
- Therefore, the sigmoid activation function is more granular and sensitive to small changes (I.e. where an input is equal to 0.5 it would result in an out of 1 where an input 0.4 would generate a 0 output

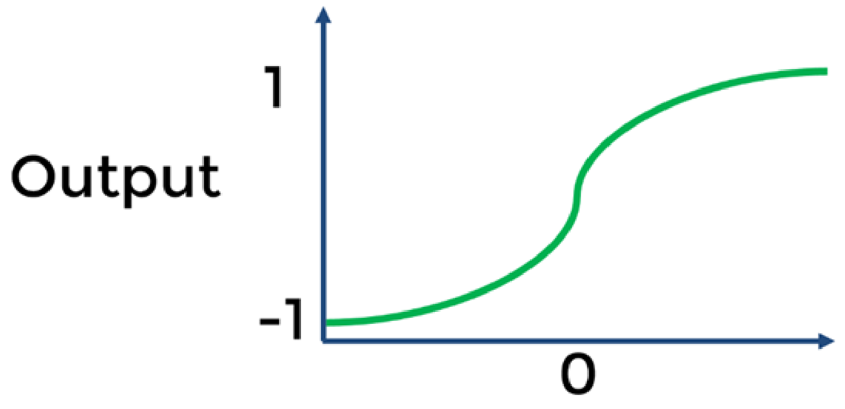
## Sigmoid Function



- Sigmoid function enables us to be more granular

## Hyperbolic Tangent

- Unlike the sigmoid activation function the output can be either 1 or –1
- Having a range between 1 and –1 allows for larger updates to the weights and bias therefore speeding up the training time of the network (discussed later in the module)
- With a sigmoid activation function only small adjustments can be made to the weights and bias during learning without causing a dramatic effect

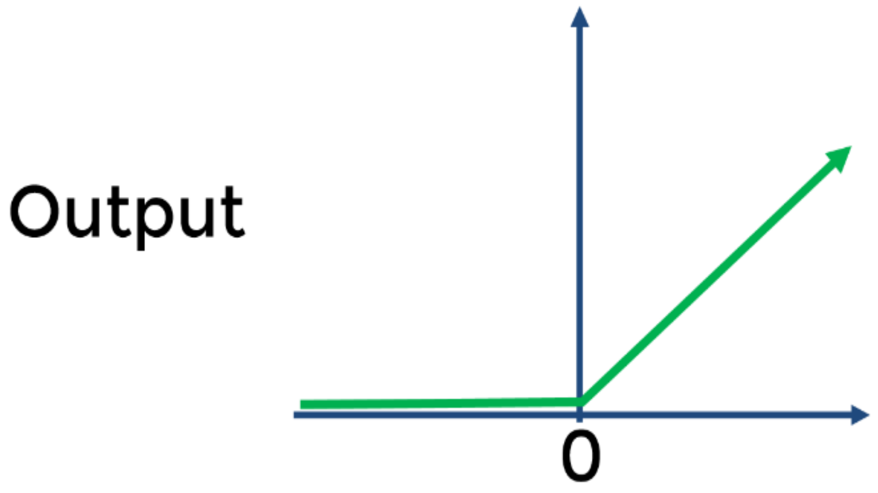


- Hyperbolic Tangent
- Output can be -1

## Rectified Linear Unit (ReLU)

- Rectified Linear Unit (ReLU) is one of the most common activation functions in ANN
- With ReLU the summed inputs, weights and bias are feed into the activation function
- If the value is below zero, the activation function returns a value of 0. If an input of 0 or more is fed in the true value of the input is returned
- The issue with both sigmoid and hyperbolic tangent they suffer from what is known as the vanishing gradient affect which will be discussed later

## Rectified Linear Unit (ReLU)



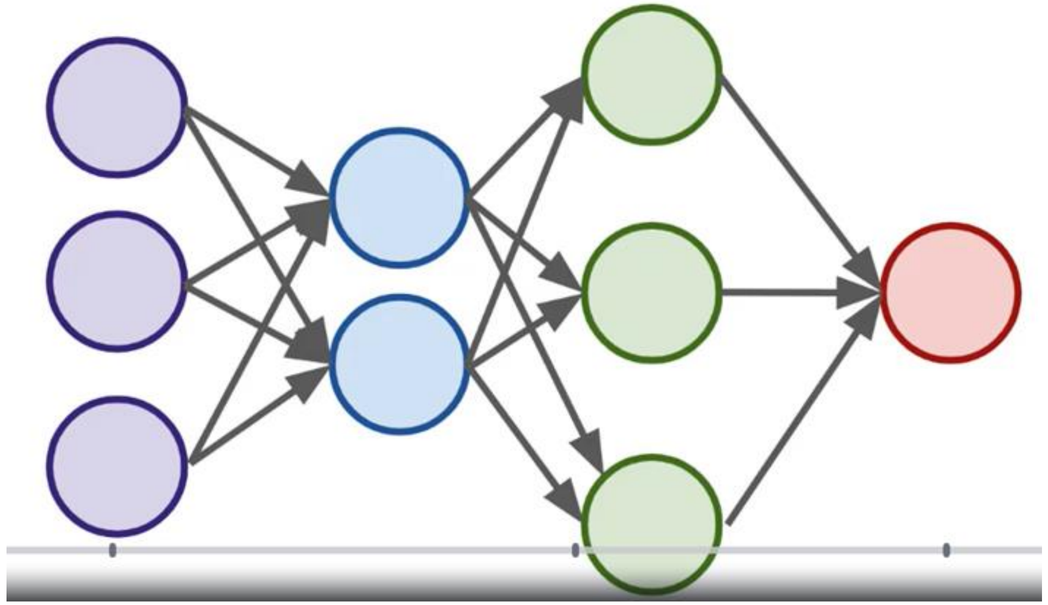
- Rectified Linear Unit (ReLU)
- Very common activation function

## Multi-Class Classification Considerations

- There are two main types of multi-class scenarios
  - **Non-exclusive classes**
    - A data point can have multiple classes/categories assigned to it. For example, a picture that can contain multiple objects in to (e.g. sandcastle, person, kite and a bird).
  - **Mutually exclusive classes**
    - A data point can have one class/category assigned to it. It is when something cannot both occur at the same time (More common in ML e.g. test for a disease and have both negative and positive).

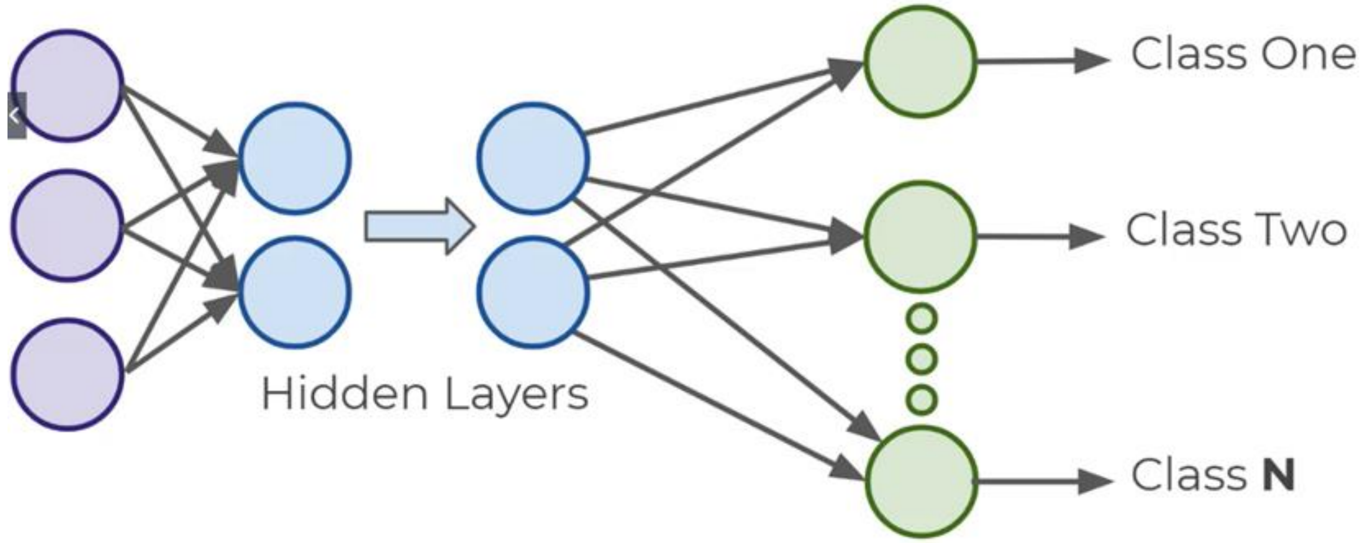
## Multi-Class Classification Considerations

- Previously we thought of the last output layer as a single node. This single node could output a continuous regression value or binary classification 0 or 1



## Multi-Class Classification Considerations

- So, what happens if you require more than one output, I.e. 2, 3 or 4 outputs
- In this instance you would need to add additional nodes to the output layer (one per class)

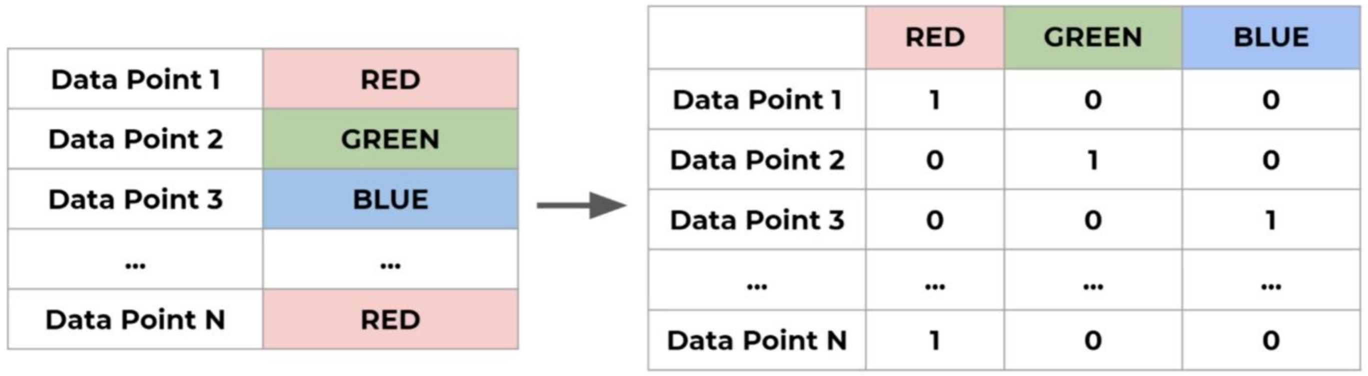


## Multi-Class Classification Considerations

- So, what happens if you require more than one output, I.e. 2, 3 or 4 outputs
- In this instance you would need to add additional nodes to the output layer (one per class)
- These outputs need to be number, I.e. you cannot use words such as “Red”, “Cat” or “Dog”
- Remember ANN’s always process numbers, you cant multiply words

## Multi-Class Classification Considerations

- These will need to be a numerical value in a correctly ordered sequence (the column positions)
- Therefore, you will need to perform a mapping between the text-based label and the corresponding output numerical value. This is commonly referred to as one hot encoding
- This technique is used for mutually exclusive classes where there can only ever be one value at a time

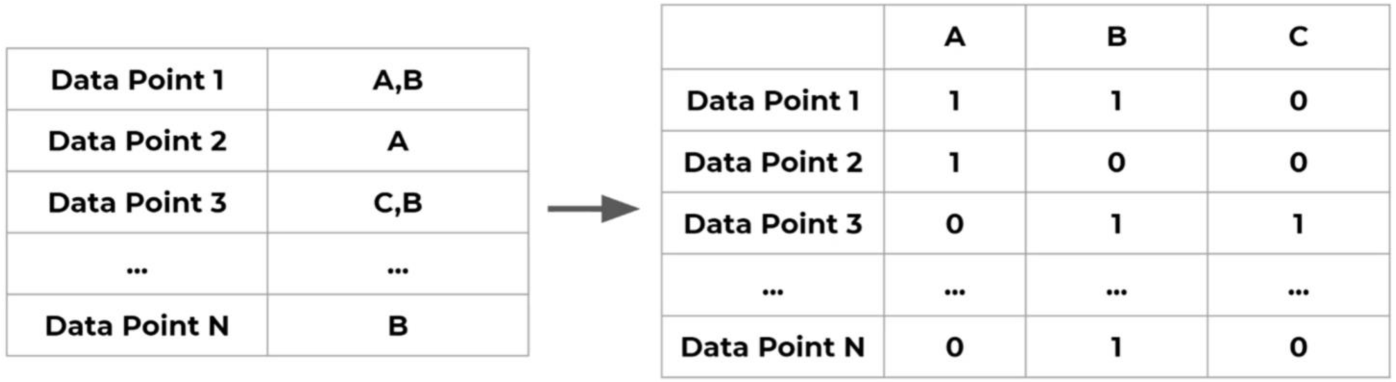


| | |
|---|---|
| Data Point 1 | RED |
| Data Point 2 | GREEN |
| Data Point 3 | BLUE |
| ... | ... |
| Data Point N | RED |

| | RED | GREEN | BLUE |
|---|---|---|---|
| Data Point 1 | 1 | 0 | 0 |
| Data Point 2 | 0 | 1 | 0 |
| Data Point 3 | 0 | 0 | 1 |
| ... | ... | ... | ... |
| Data Point N | 1 | 0 | 0 |

## Multi-Class Classification Considerations

- For non-exclusive classes this process changed to accommodate multiple outputs for example an image can have more than one example in it



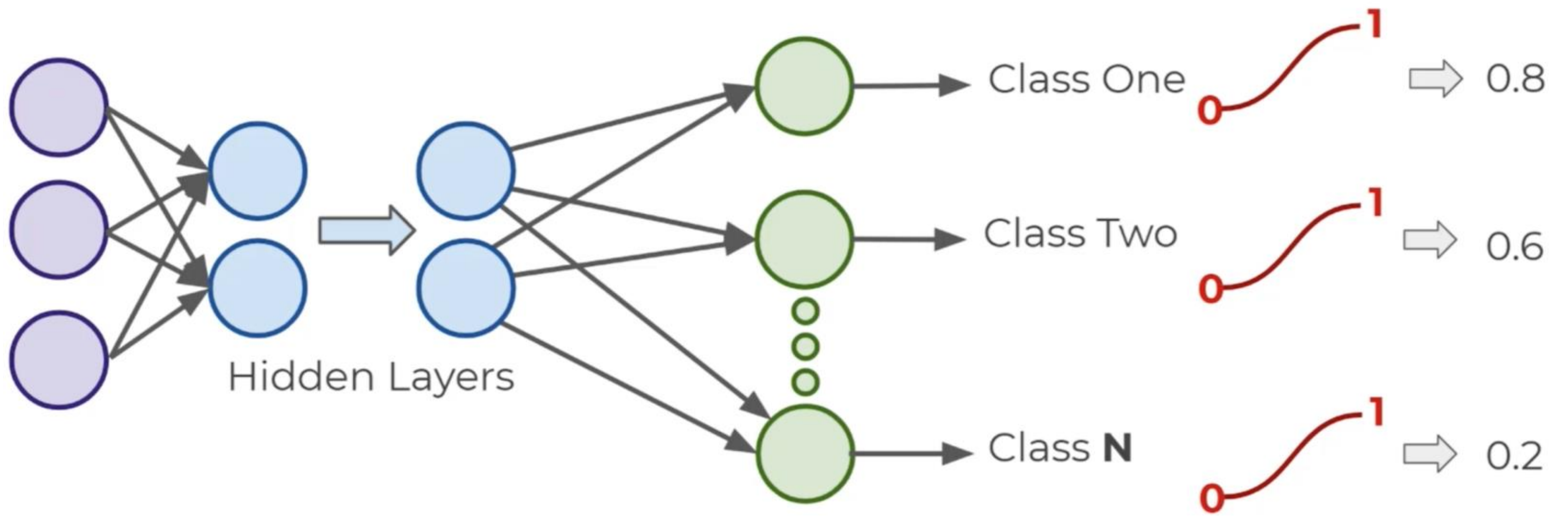
| | |
|---|---|
| Data Point 1 | A,B |
| Data Point 2 | A |
| Data Point 3 | C,B |
| ... | ... |
| Data Point N | B |

| | A | B | C |
|---|---|---|---|
| Data Point 1 | 1 | 1 | 0 |
| Data Point 2 | 1 | 0 | 0 |
| Data Point 3 | 0 | 1 | 1 |
| ... | ... | ... | ... |
| Data Point N | 0 | 1 | 0 |

- Once we have encoded our data, we need to choose the correct architecture for our output layer and the appropriate activation function

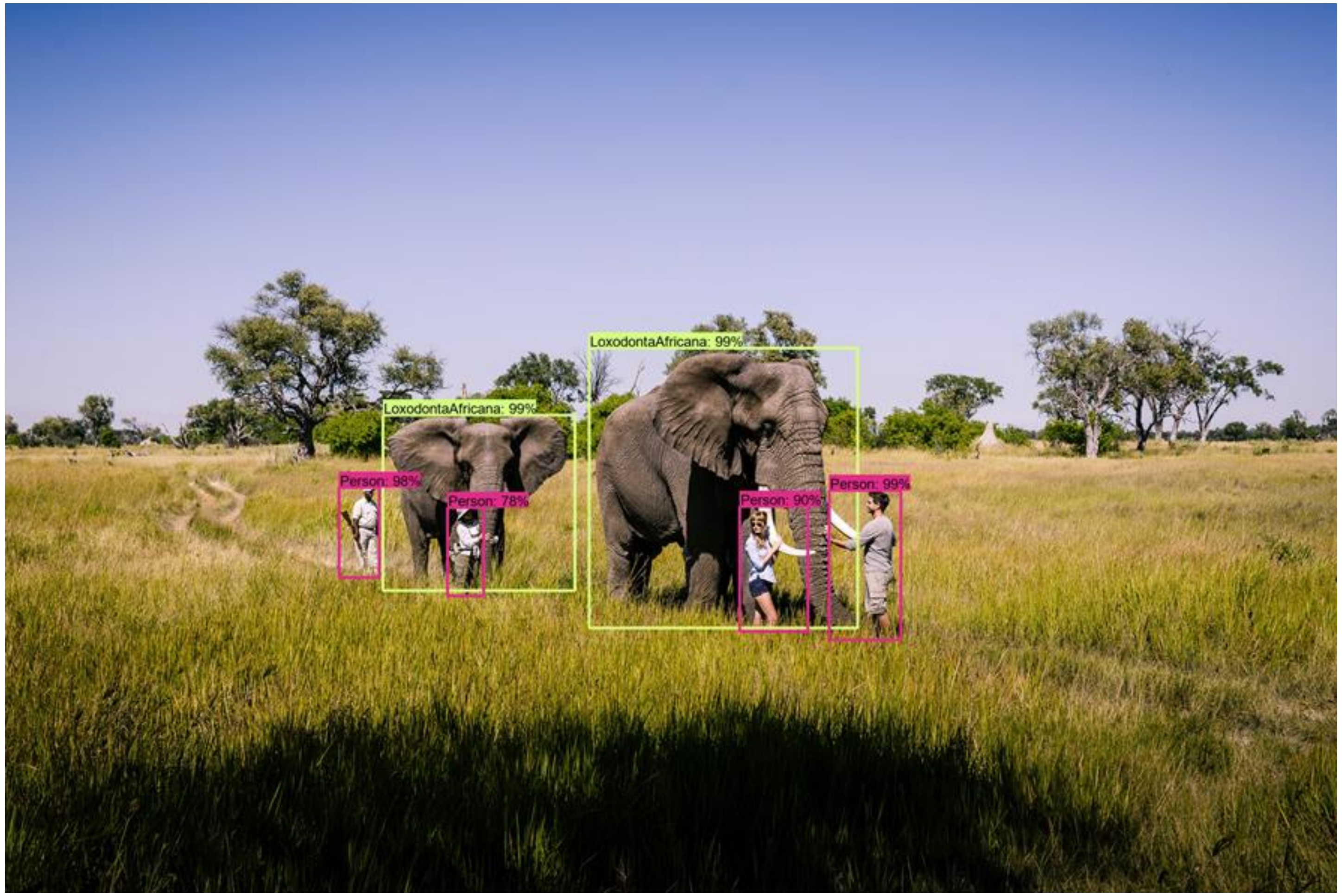
# Multi-Class Classification Considerations

- **Non-exclusive**
  - Sigmoid activation - each neuron will output a value between 0 and 1, indicating the probability of that class assigned to it



  - As its Non-exclusive we can set a threshold to include more classes e.g. a threshold of 0.5 would resuturn both class one and class two

## Multi-Class Classification Considerations

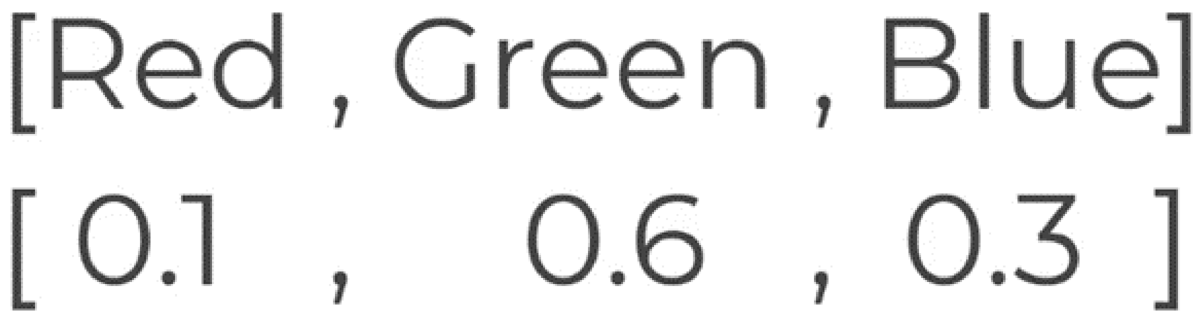


## Multi-Class Classification Considerations

- **Mutually exclusive classes**
  - It allows each neuron to output a value independently of other classes
  - This means that it allows a single datapoint to be input to the network and allow multiple outputs generated
- But what do we do when each data point can only have a single class assigned to?
- We can address this issue using the SoftMax function. The SoftMax function calculates the probability distribution over k-different events
- As a result, this function will calculate the probabilities of each target class over all target classes

## Multi-Class Classification Considerations

- The class with the highest probability will be the correct class. Note that the range of probabilities across all outputs will be between 0 and 1 and the sum of all probabilities will be equal to one
- The model returns the probability of each class and the class with the highest probability value is chosen



[Red , Green , Blue]

[ 0.1 , 0.6 , 0.3 ]

# Next Session

- Loss Functions
- Optimisation Algorithms (SGD, ADAM etc.)
- Local and Global Minima, local Maxima, Global Maxima
- Learning Rate
- Configuring Hyperparameters