Using Machine Learning Tools PG

Week 8 – Deep Learning & Neural Networks

COMP SCI 7317

Trimester 2, 2024

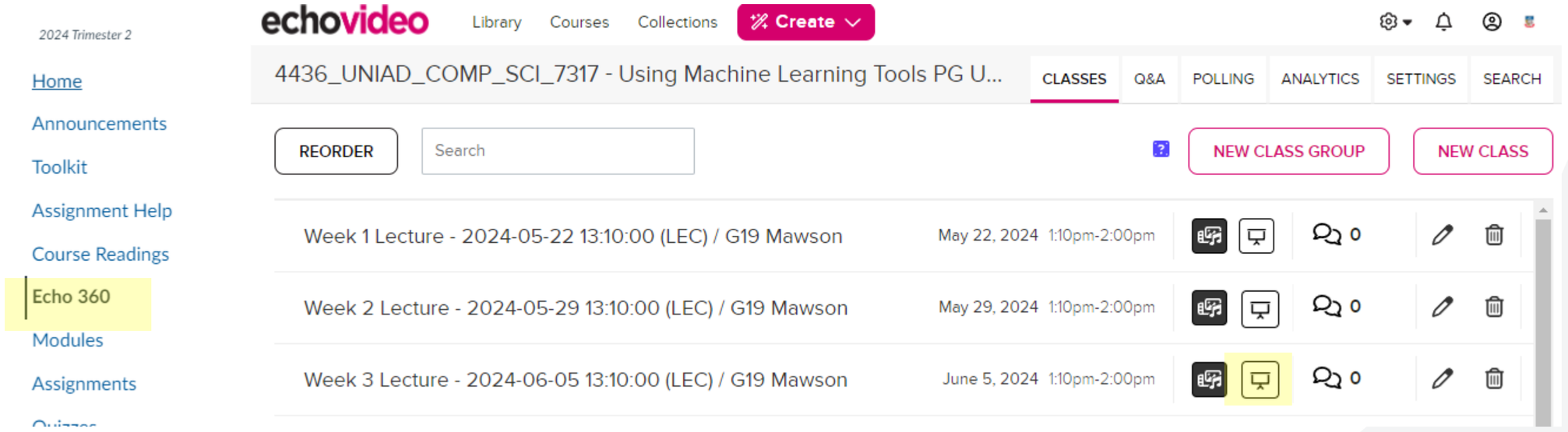
THE UNIVERSITY of ADELAIDE

150 YEARS

Before we start…

There will be some **interactive elements** throughout this lecture.

Please participate by heading to **MyUni** > **4436_COMP_SCI_7317** > **Echo360** > **Lecture Week 8** > 



THE UNIVERSITY of ADELAIDE
150 YEARS

Last week… Unsupervised learning

1. **Clustering**
   - K-Means
   - Gaussian Mixture Models
2. **Dimensionality Reduction**
   - PCA, tSNE
   - Other feature selection methods
3. **Data visualisations using unsupervised learning approaches**

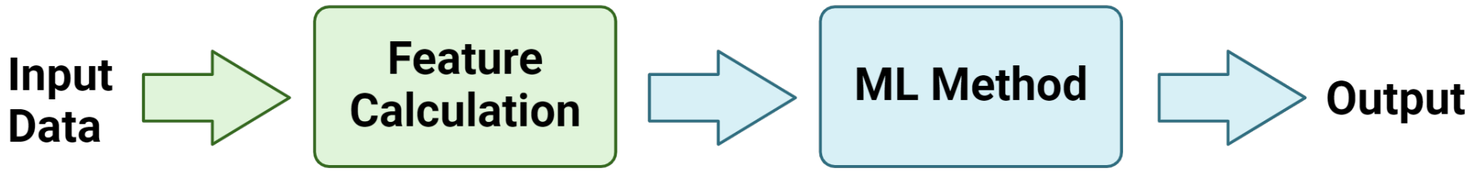
This week

1. **Components of a Neural Network**
   - Neurons: non-linearities and activation functions
   - Network architecture
   - Parameters: connections, weights and biases
   - Number of parameters
   - Deep Neural Networks
2. **Loss functions, epochs, batches, optimisers and training**
3. **Classification and regression variants**

Components of a Neural Network

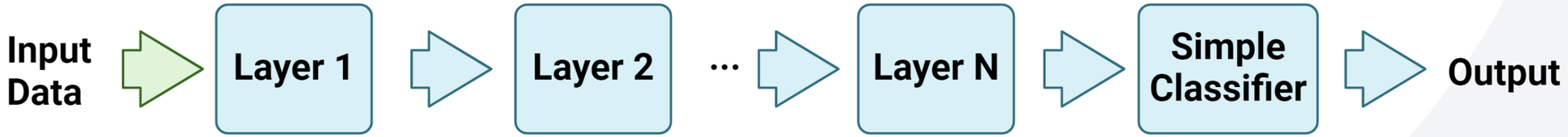
Introduction to Neural Networks

Traditional Machine Learning



- "Hand crafted features" i.e. pre-define the relevant features

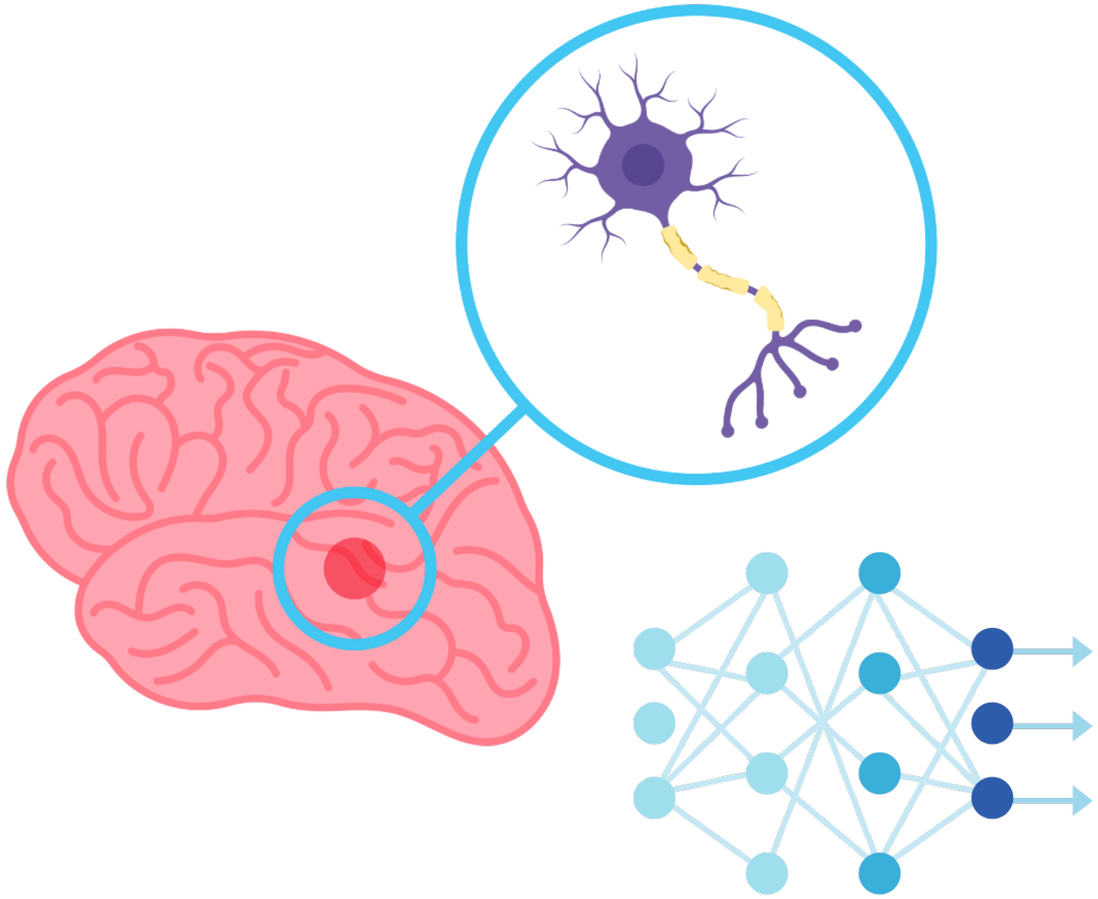
Deep Learning



- "Learned features": Network automatically learns relevant features
- Tend to have many more parameters + need much more data

Neurons

Neurons (also known as nodes or units) are the fundamental building blocks of the neural network.

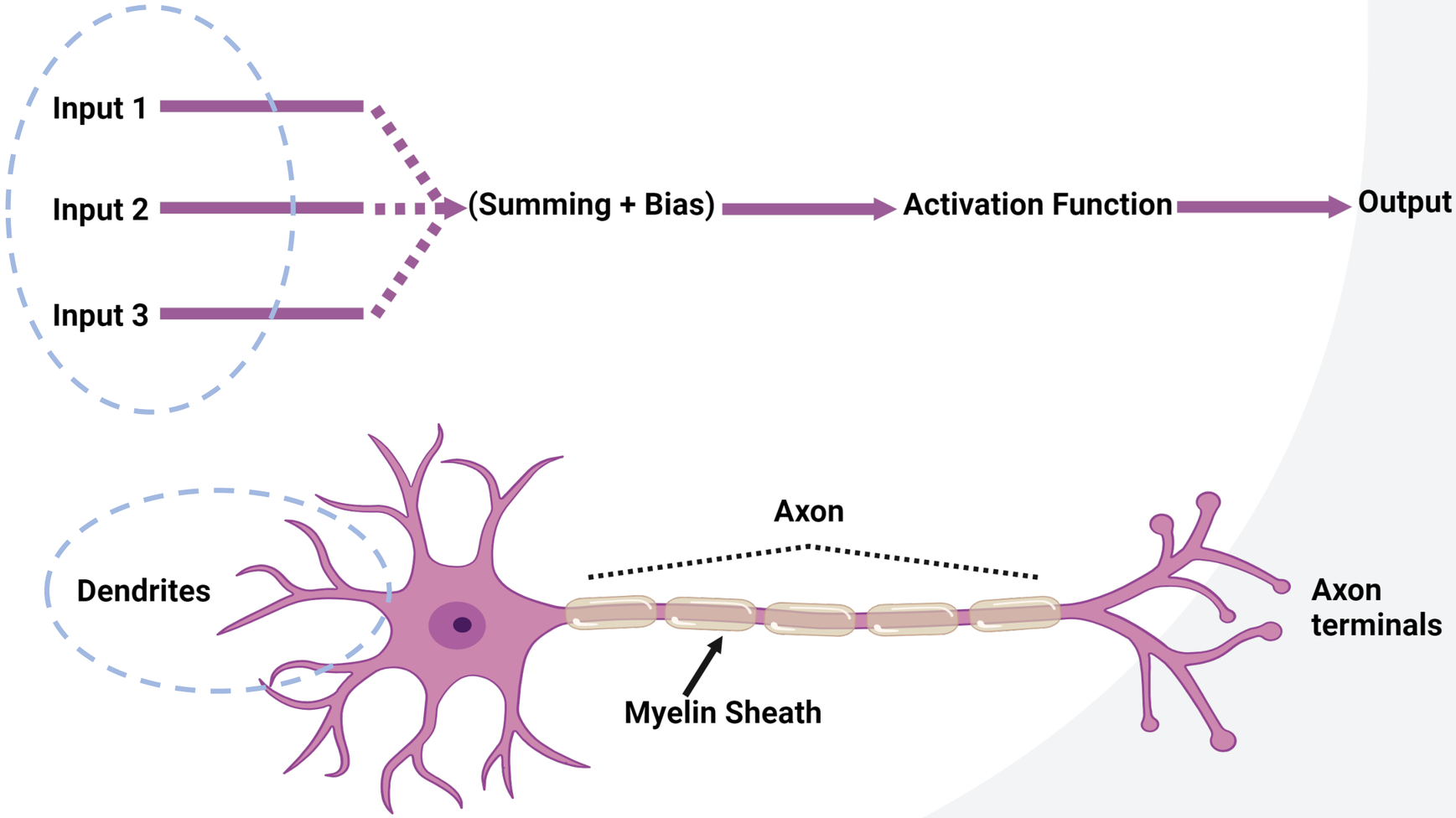


- Inspired by biological neurons in the brain but function in a simplified and mathematical manner.
- **Role:** to apply a set of weights to the inputs, sum them, pass the result through an activation function, and output the final value.
- **Training:** network adjusts weights and biases of the neurons to minimise the error in predictions (typically done using backpropagation and optimisation algorithms like gradient descent).

Neurons

- **Inputs:** Neurons receive multiple inputs $(x_1, x_2, \ldots, x_n)$. These inputs represent features of the data, such as pixel values in an image or attributes in a dataset.

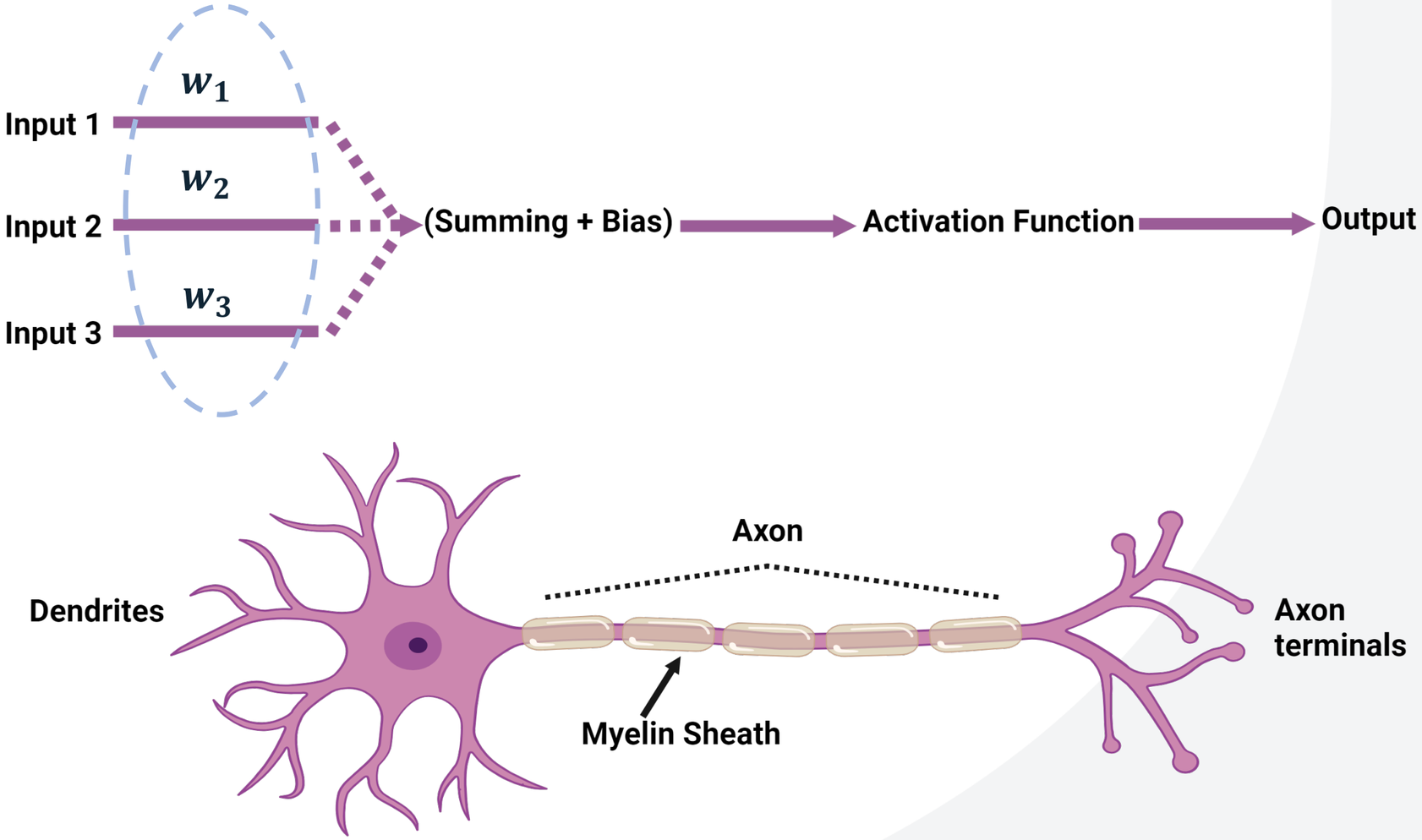
Dendrites receive signals. Similarly, inputs to an artificial neuron receive data from previous layers or features like pixel values or attributes.



Neurons

- **Weights:** Each input is associated with a weight ($w_1, w_2, \ldots, w_n$), which determines the significance of each input in the neuron's computation. These are optimised during training.

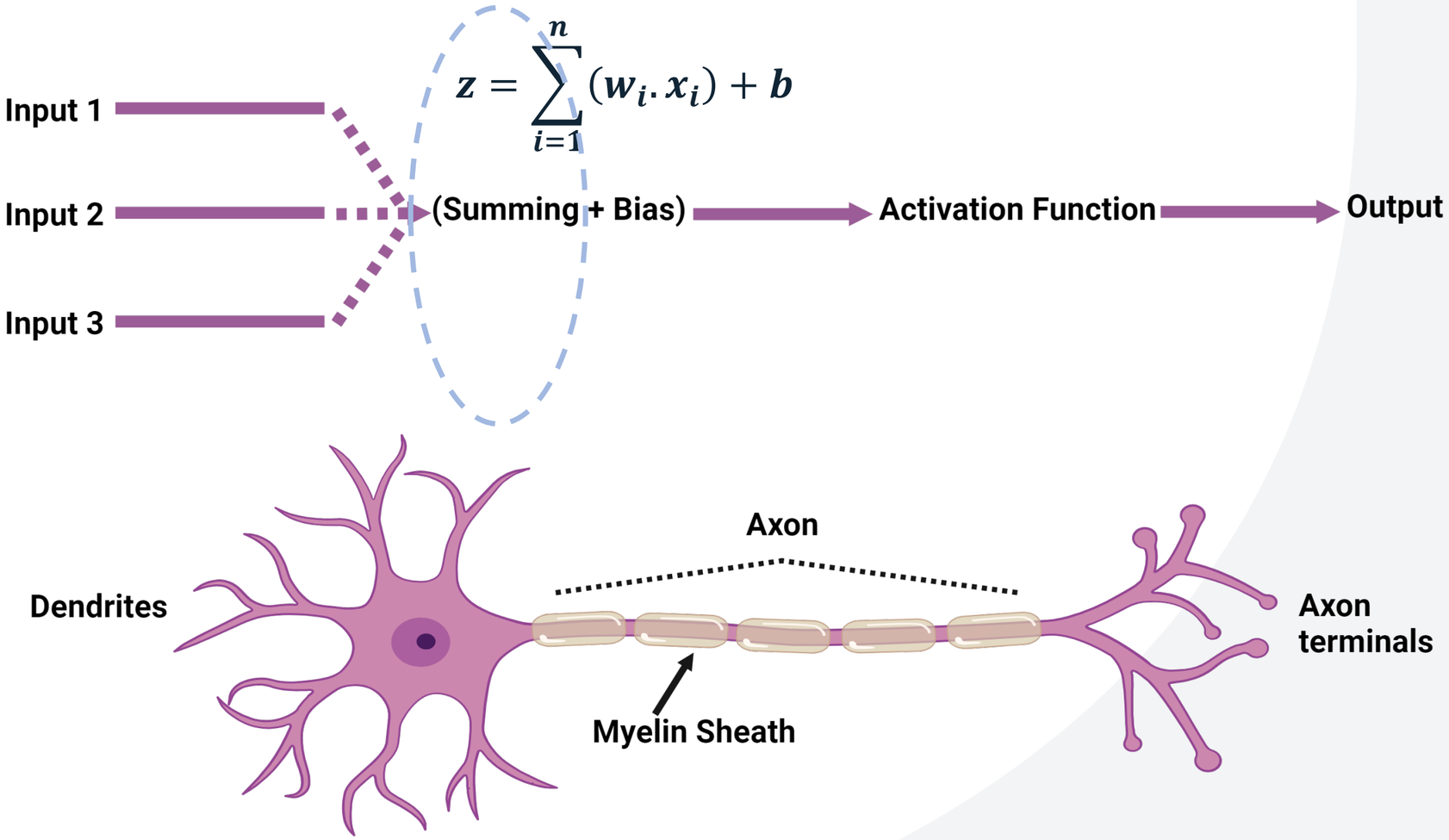
Synaptic strengths influence biological neurons. Similarly, weights in artificial neurons determine input importance and are optimized during training.



Neurons

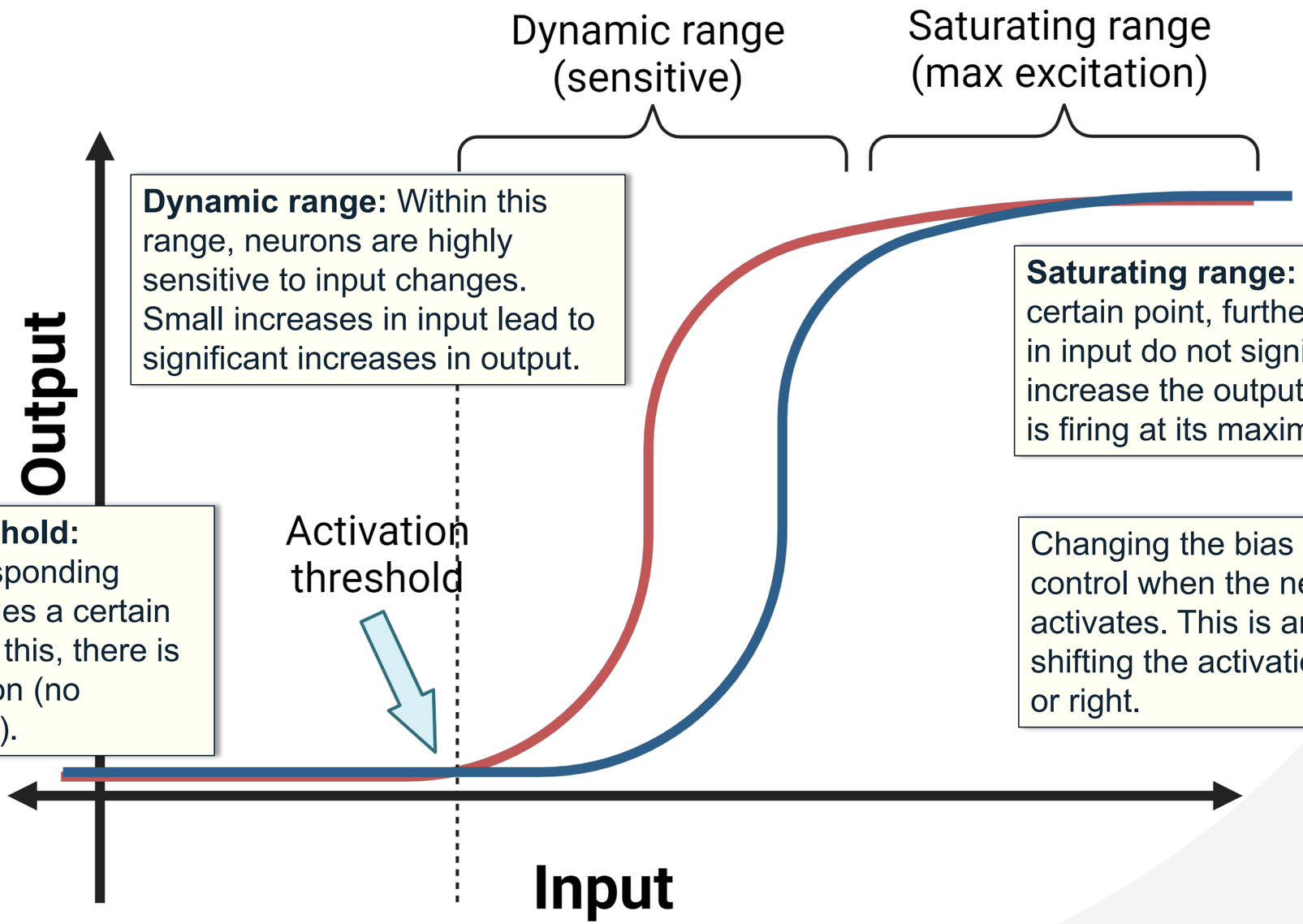
- **Summing:** The neuron calculates a weighted sum of the inputs and the bias, which effectively combines the inputs into a single value (output).

The neuron calculates a weighted sum of inputs and bias, combining them into a single output.



Activation of biological neurons

Graph shows how real neurons respond to different levels of input.



Dynamic range: Within this range, neurons are highly sensitive to input changes. Small increases in input lead to significant increases in output.

Saturating range: Beyond a certain point, further increases in input do not significantly increase the output. The neuron is firing at its maximum rate.

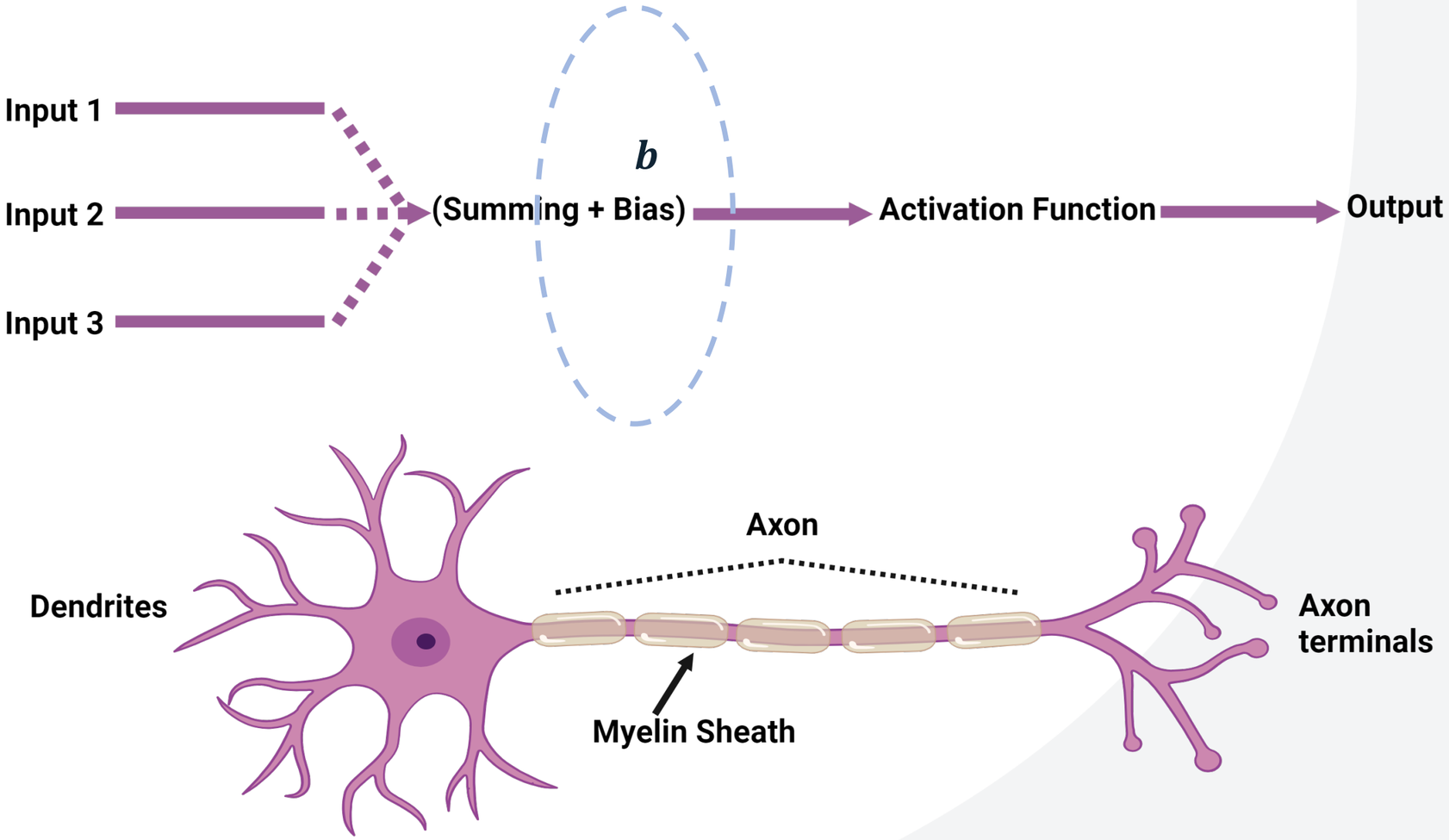
Activation threshold: Neurons start responding when input reaches a certain threshold. Below this, there is complete inhibition (no significant output).

Changing the bias allows us to control when the neuron activates. This is analogous to shifting the activation curve left or right.

Neurons

- **Bias:** In addition to inputs, each neuron has a bias term b that provides flexibility and helps shift the activation function, improving the model's fit to the data.

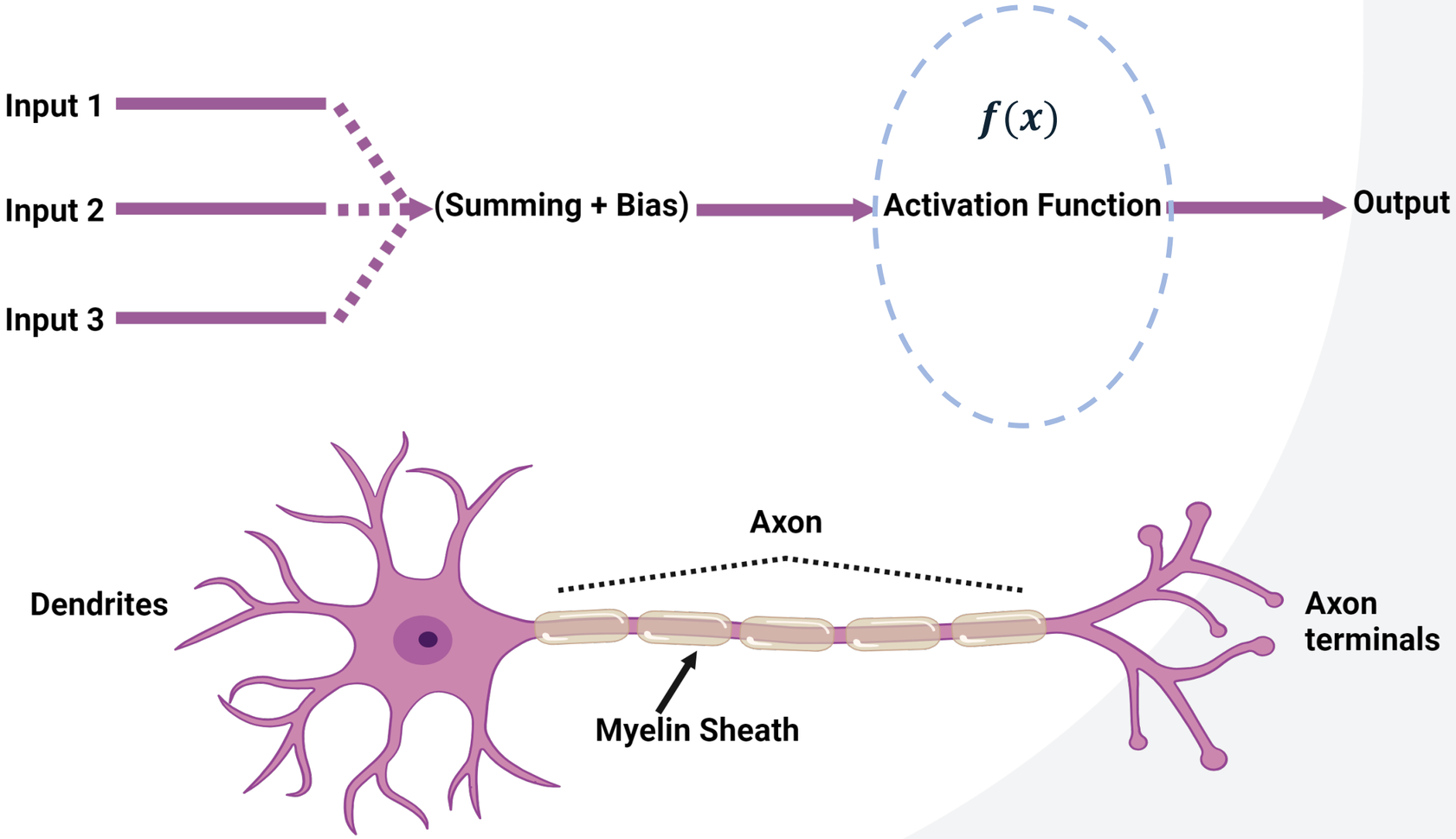
Bias, like resting potential, provides baseline activation, helping the neuron to activate more easily.



Neurons

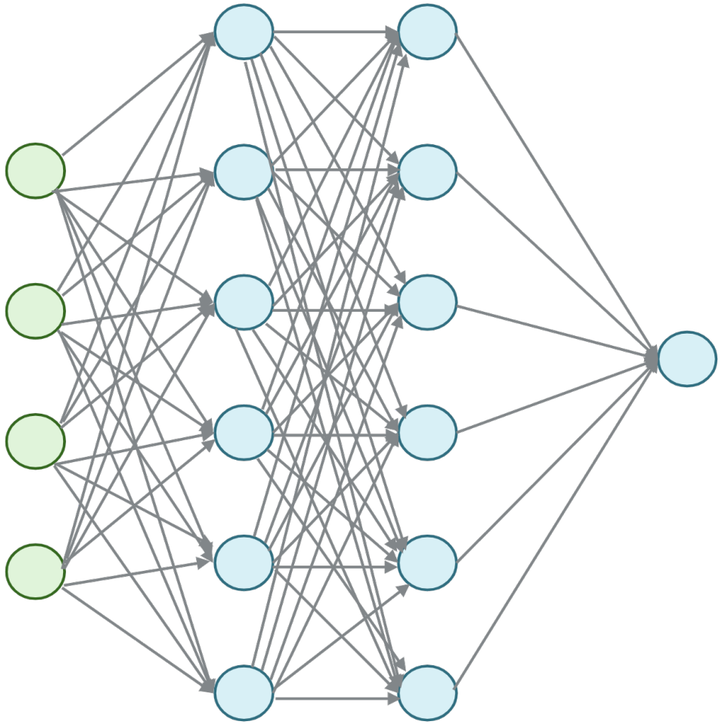
- **Activation Function:** Weighted sum z is then passed through an activation function $f(z)$, which introduces non-linearity into the model, enabling it to learn complex patterns.

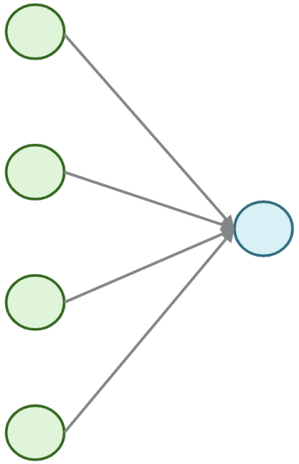
The activation function decides if the neuron activates, like an action potential in a biological neuron. This introduces non-linearity, allowing the network to learn complex patterns. Examples include Sigmoid, ReLU, and Tanh.

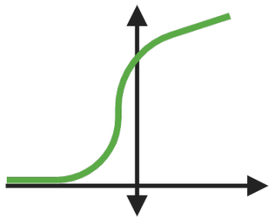


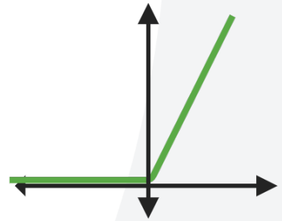
Neurons

- Without the non-linearity, the whole network just computes a linearly weighted sum and the below networks end up being equivalent.





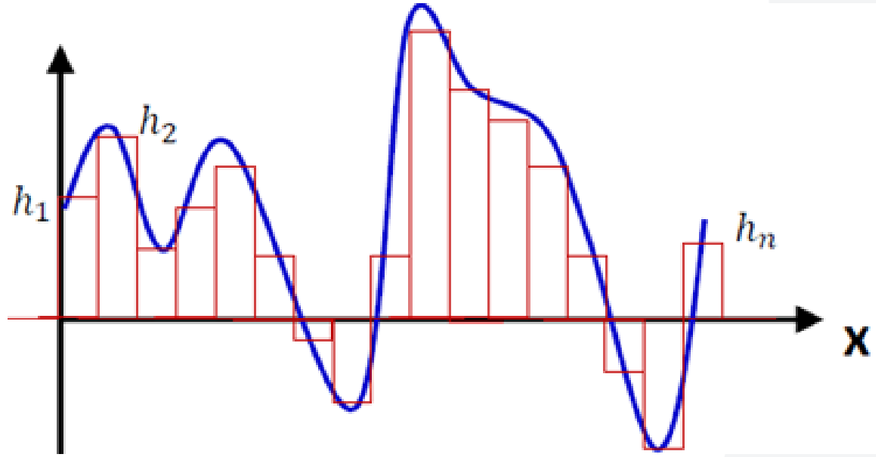




- Non-linearity is crucial, and a number of options exist, all similar to sigmoids.
- ReLU (Rectified Linear Unit) is most popular.

Neurons

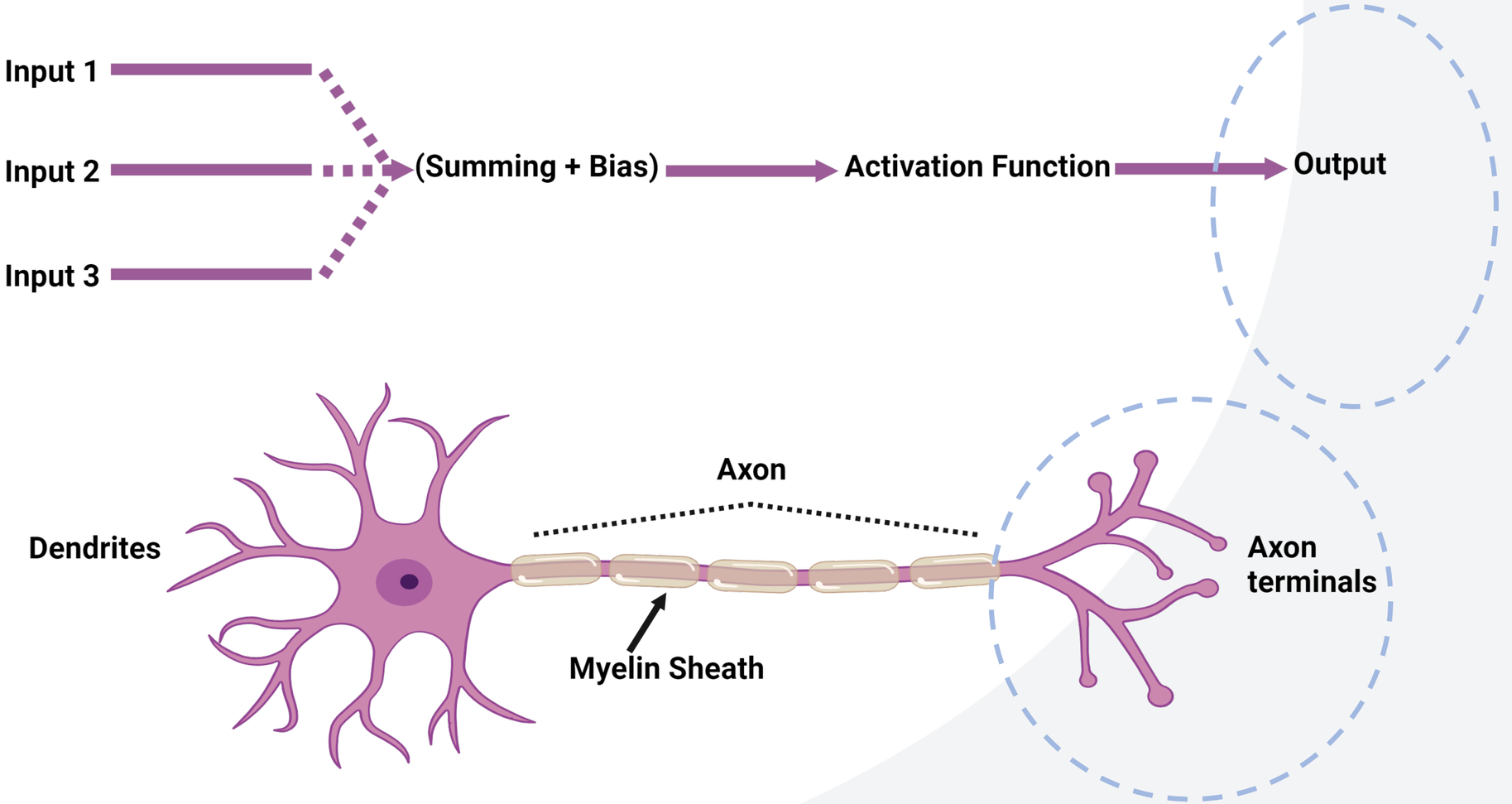
- Linear models can only capture linear relationships between inputs and outputs, which may not reflect complex real-world data.
- **Universal approximation theorem:** neural network with at least one hidden layer and non-linear activation functions can approximate any continuous function (given enough neurons).
  - Can model any type of data distribution or function → highly versatile
- Non-linearities allow network to learn interactions between features by combining basic features learned in earlier layers in more complex ways.
  - Without non-linearity, each layer would just perform linear transformation and stacking more layers would not add additional modelling power.



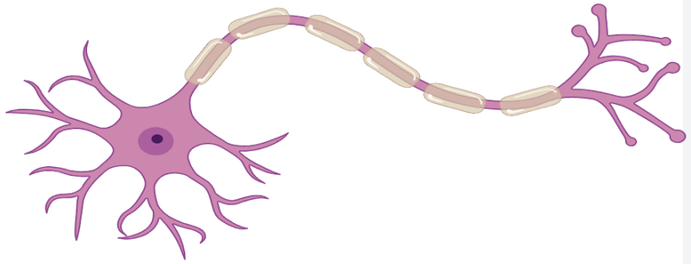
Neurons

- **Output:** Output of the activation function, $f(z)$, is the neuron's final output, which is typically transmitted to neurons in subsequent layers. This facilitating the propagation of information through the network..

Axon terminals transmit signals in biological neurons; similarly, the output of an artificial neuron is passed to the next layer or final output.



Neurons

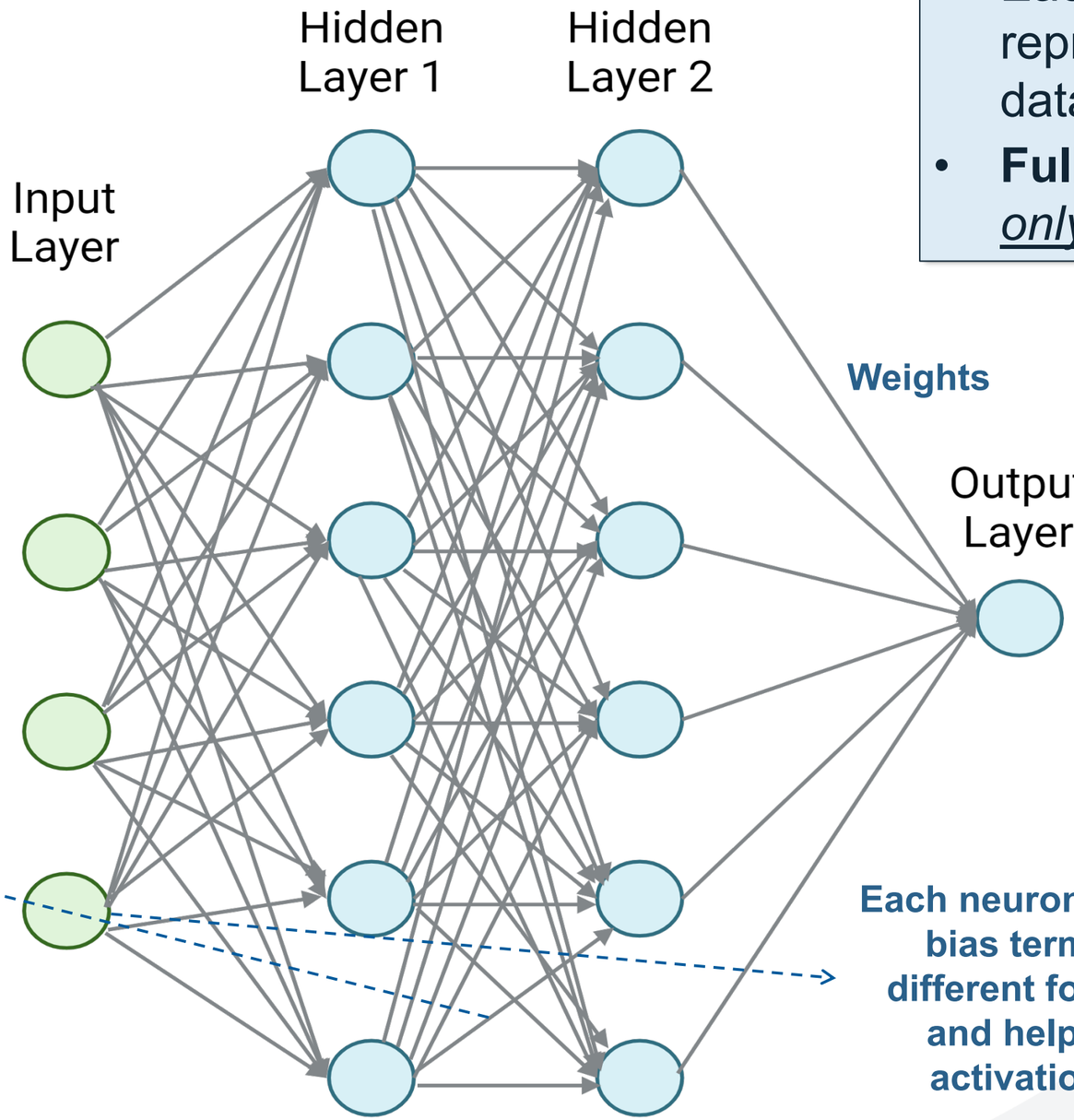


THE UNIVERSITY of ADELAIDE

150 YEARS

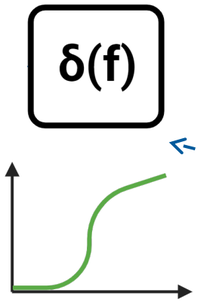
Neural Network

- Commonly used architecture.
- Each green circle/input node represents a single value (i.e. data sample, pixel).
- **Fully connected:** Each layer *only* connects to next layer.



Input values as 1D vector

Fully Connected Layout



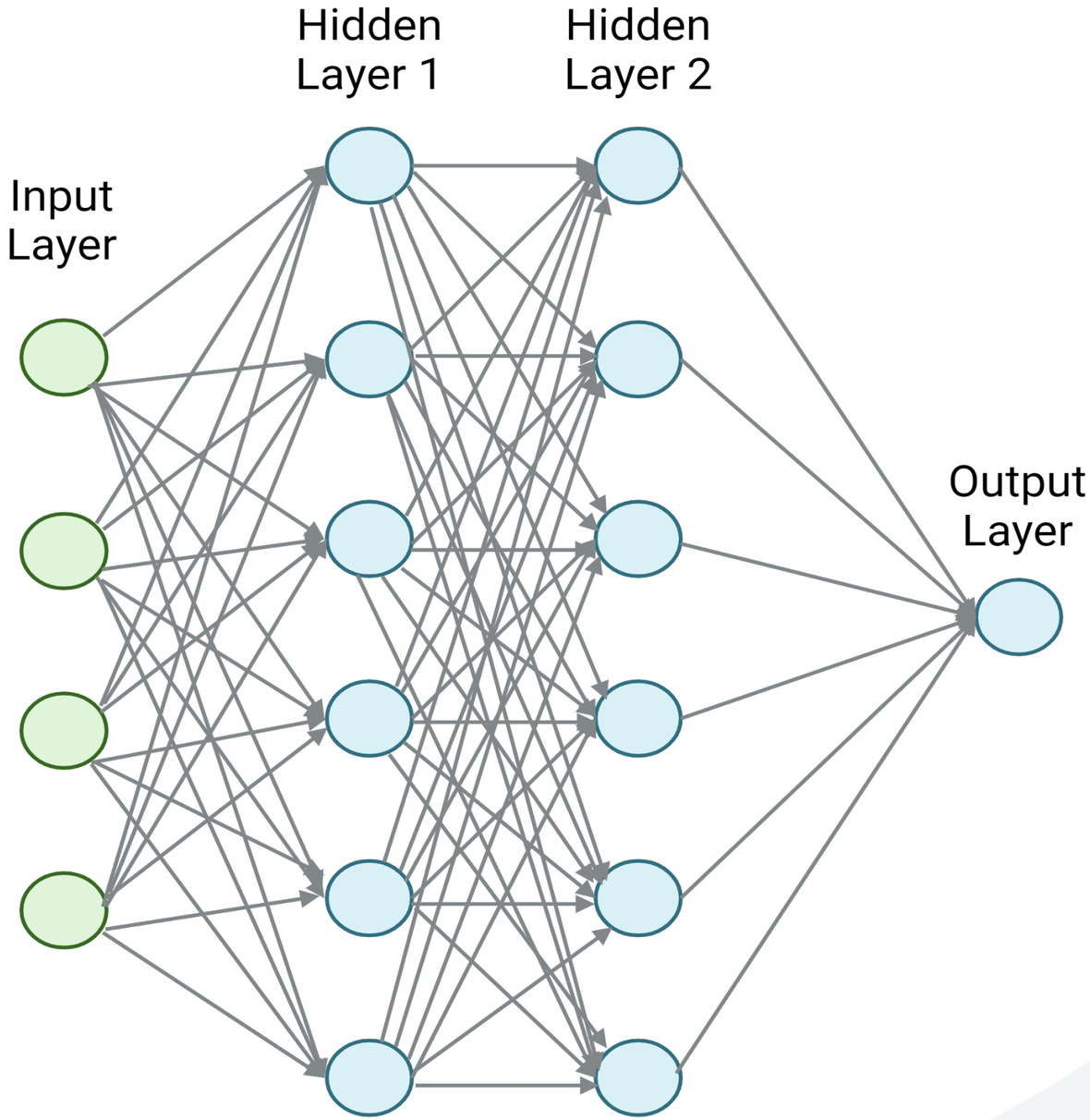
Activation function

Each neuron has an implicit bias term b, which is different for each neuron and helps adjust the activation threshold

THE UNIVERSITY of ADELAIDE
150 YEARS

Neural Network

#Parameters = #Weights + #Bias



Fully Connected Layout

Hidden layer 1:
4 inputs (weights) + 1 bias

Hidden layer 2:
6 inputs + 1 bias

Output layer:
6 inputs + 1 bias

Number of parameters:
5*6 + 6*7 + 1*7 = 79

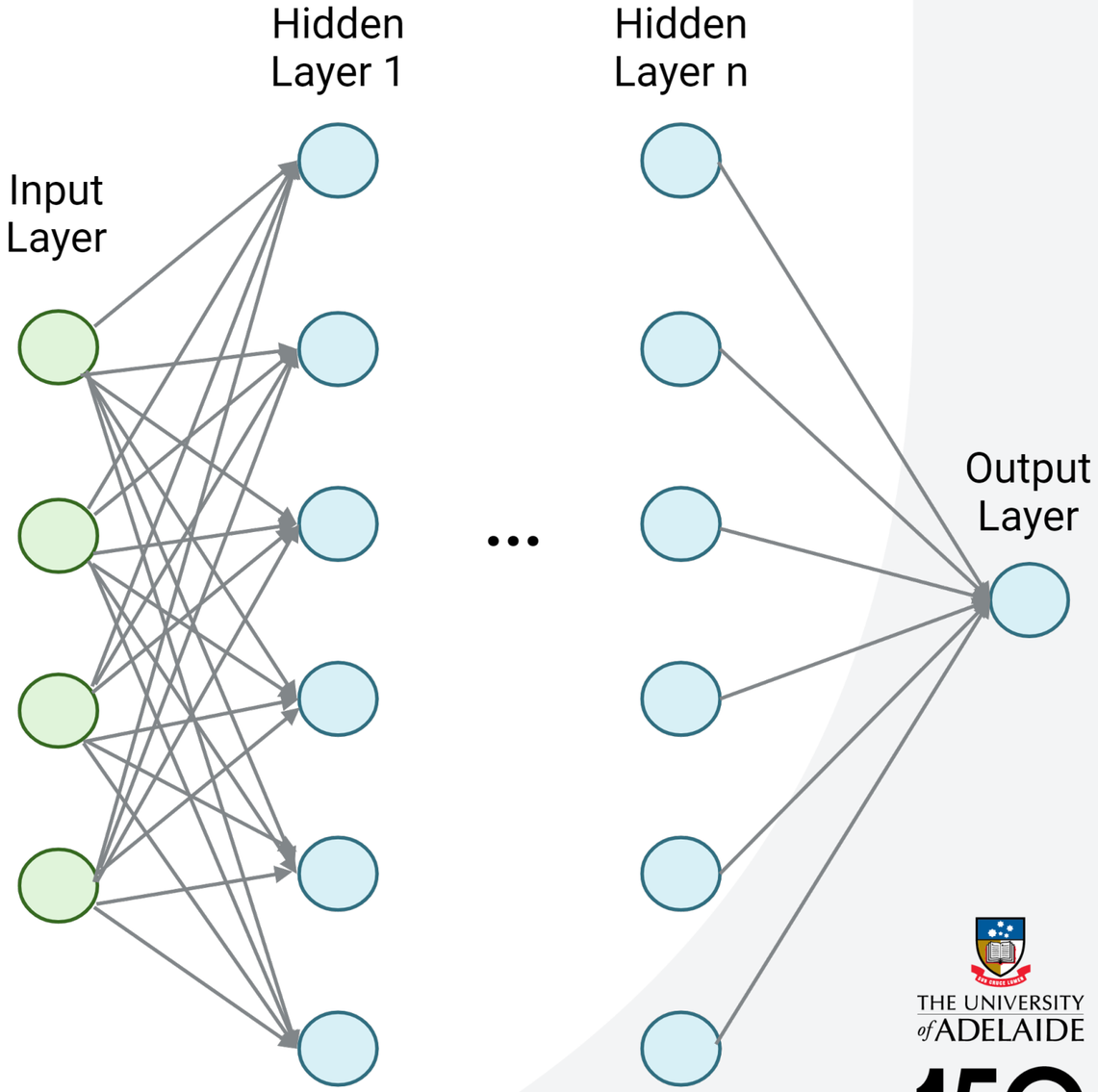
Already 79 parameters for this small network!

- More parameters enable modelling more complex patterns.
- Requires more data for effective training and careful management to avoid overfitting.

THE UNIVERSITY of ADELAIDE
150 YEARS

Deep Neural Network

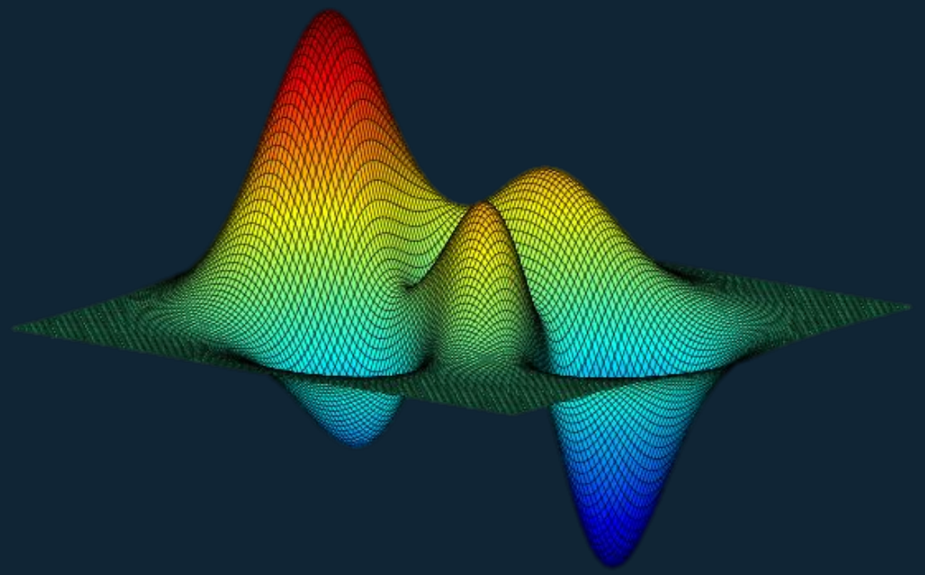
- **Deep neural networks (DNNs)** are typically much larger, consisting of an input layer, multiple hidden layers, and an output layer.
- Capable of modelling more complex relationships and patterns in data.
  - Suitable for tasks like image recognition, natural language processing, and other applications requiring the learning of intricate features.
- Slower to train, requires more data and computational resources.



Which answer is correct?

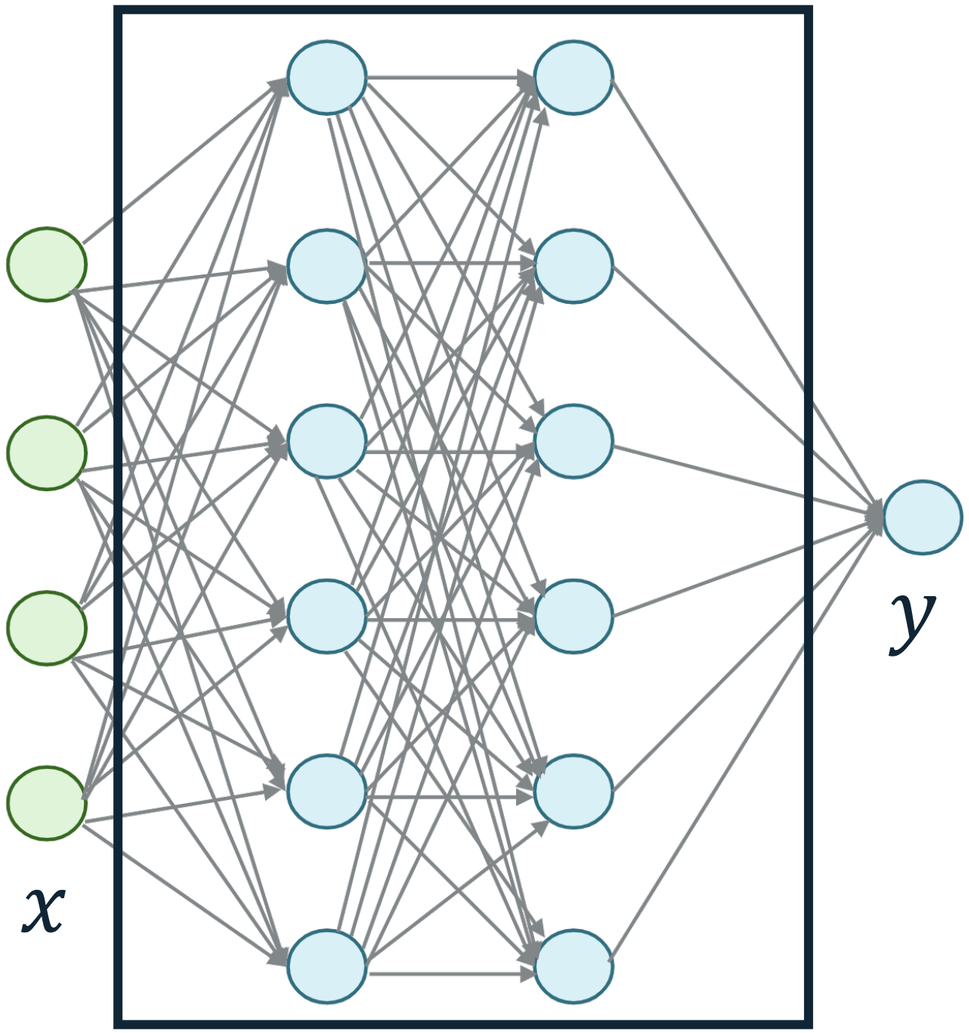
What is the main purpose of an activation function in a neural network?

- To initialise weights
- To introduce non-linearity
- To normalise input data
- To optimise the loss function



Loss functions, epochs, batches, optimisers and training

Loss Function



$y = g(x; w,b)$

Loss: $loss(y_{pred}, y_{train})$

Predicted output: $y_{pred} = g(x_{train})$

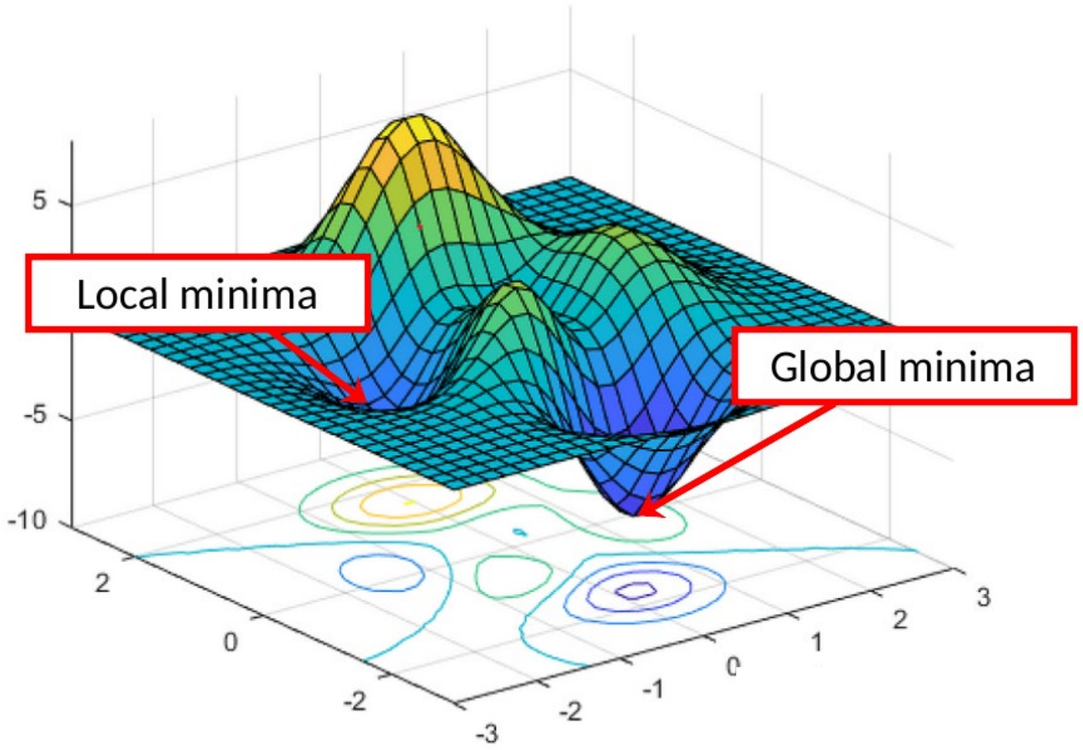
Process:

- Minimise loss by changing parameters (weights + biases)
- Sum loss over batches/epochs

g: general function the neural network is learning to approximate, using the given weights and biases to map inputs to outputs.

Parameters: weights (w), bias terms (b).

Optimisation



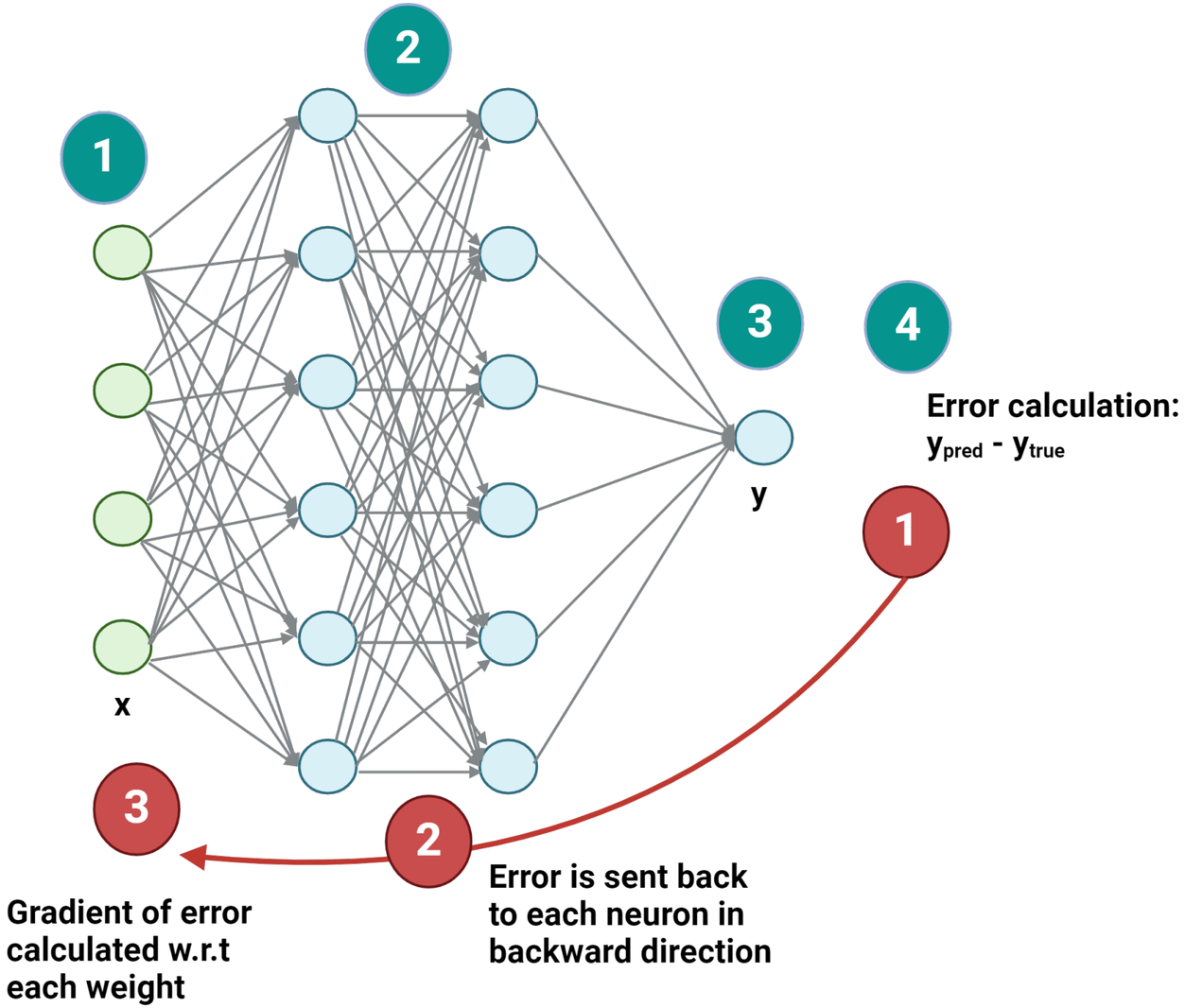
- Like other ML approaches, optimisation methods are used to find the minimum loss value in neural networks and deep neural networks.
- Gradient descent, including backpropagation, is commonly used for optimisation.

Key factors for effective optimisation:

- **Initialisation:** Proper initialisation of weights.
- **Learning Rate:** Appropriate learning rate.

Optimisation: Backpropagation

Used to compute the gradient of the loss function with respect to each weight by the chain rule, essential for gradient descent and to adjust weights to minimise error in predictions.



Forward pass: Compute the output of the network. Calculate the loss based on the difference between predicted and actual values.

Error calculation: Calculate the error (difference between predicted output and actual output).

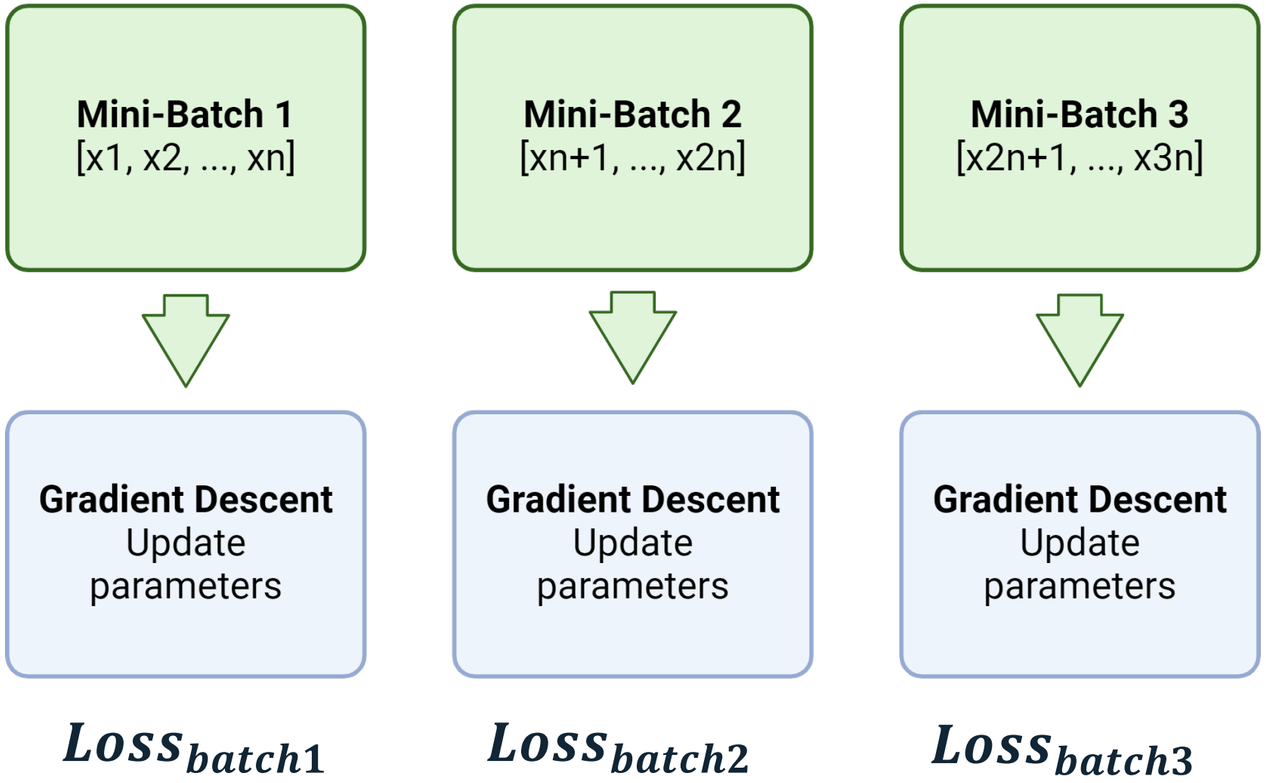
Backward Pass: Compute the gradient of the loss function with respect to each weight. Update the weights using gradient descent.

Optimisation: Backpropagation

- **Systematic Gradient Calculation:**
  - The chain rule allows the gradient of the error to be propagated backward through the network, calculating the contribution of each layer to the overall gradient (determines contribution of each weight/bias).
- **Error Propagation:**
  - The error is propagated backward from the output layer to the input layer.
  - Ensures accurate gradient computation for each weight and bias.
- **Weight Updates:**
  - Gradients obtained from backpropagation are used by optimisation algorithms (i.e.., gradient descent) to update weights and biases.
  - The goal is to adjust parameters to minimise the loss function, reducing prediction error.
- **Efficiency:**
  - This layer-by-layer approach avoids redundant calculations and ensures efficiency.
  - Makes the process more memory-efficient and leads to faster convergence.

Optimisation: Batches and Epochs

When dealing with a large amount of data in deep learning, the loss function is typically averaged over mini-batches rather than the entire training set.



- **Mini-Batch Averaging:** Each mini-batch has its own averaged loss for computing gradients and updating parameters.

$$Loss_{batch} = \frac{1}{n}\sum_{i=1}^{n} Loss(y_{pred}, y_{true})$$

- **Forward pass:** Calculate individual losses for each sample and compute the average loss over the mini-batch.
- **Backward pass:** Compute gradients of the averaged loss with respect to model parameters.
- Repeat until the loss across all mini-batches stabilises or does not significantly decrease further (converges).

Optimisation: Batches and Epochs

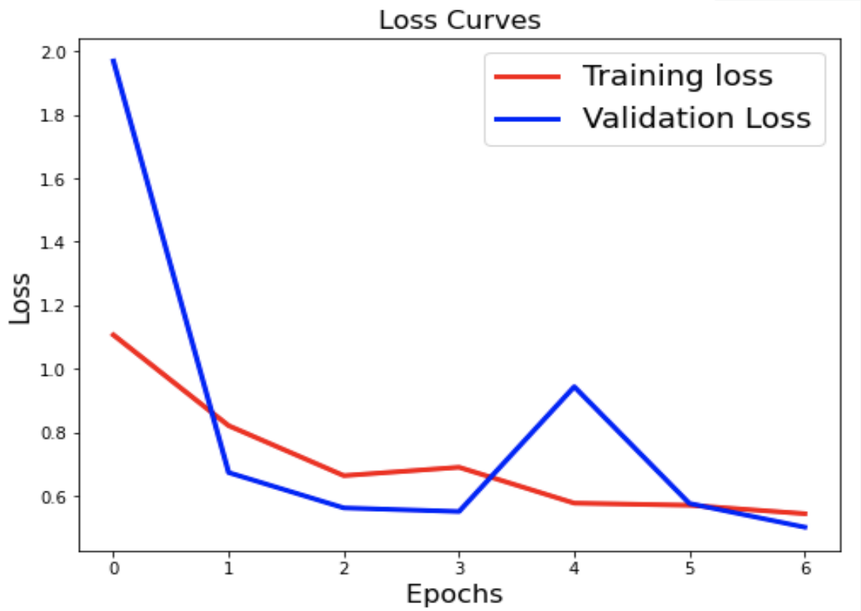
Optimiser: Calculates the gradients of the loss function and updates the model parameters once per batch (i.e. mini-batch gradient descent).

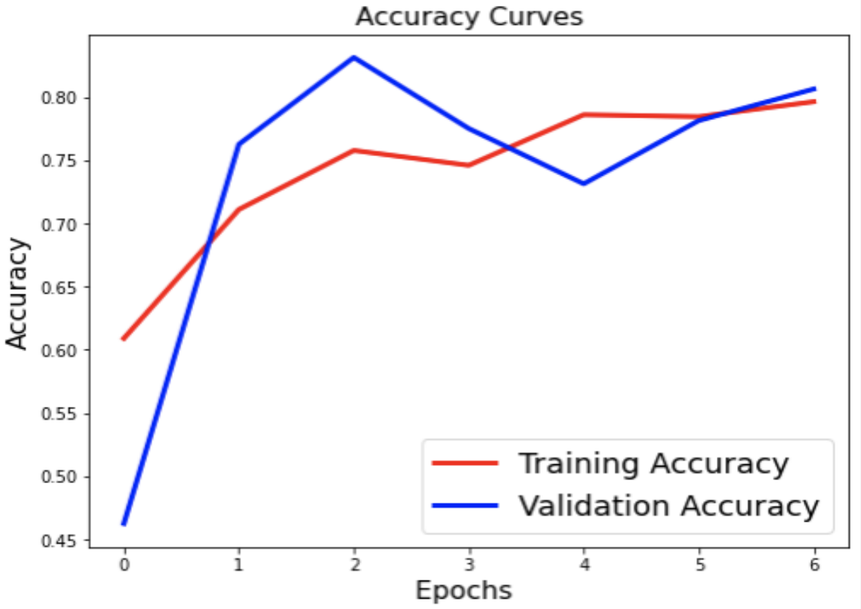
- **Batch Size:** The number of sample. Impacts memory and execution speed.

Repeat Batches: Training set is divided into multiple batches. The optimiser processes each batch sequentially.

- **One Epoch:** A single complete pass through the entire training dataset. After one epoch, the optimiser has processed all batches.

Repetition: The training process involves multiple epochs, repeating the process until the model converges.

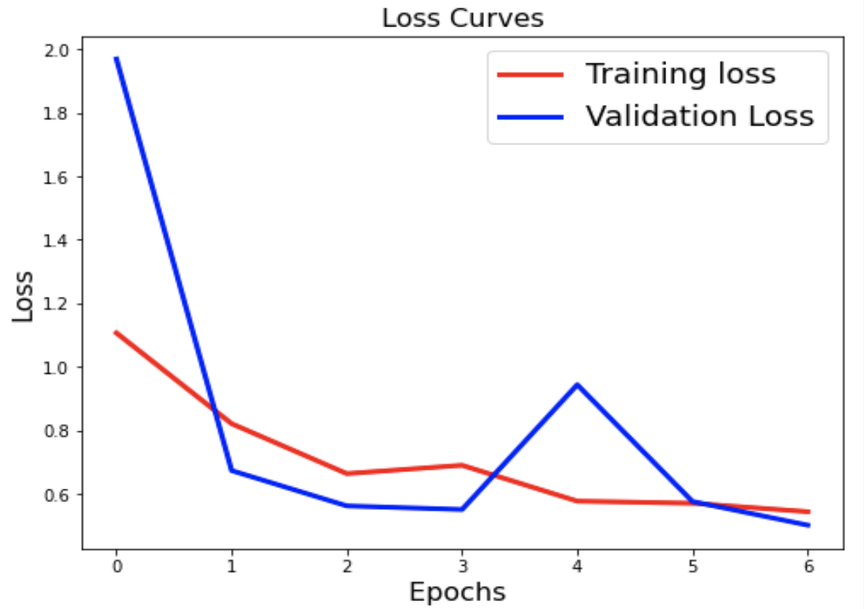


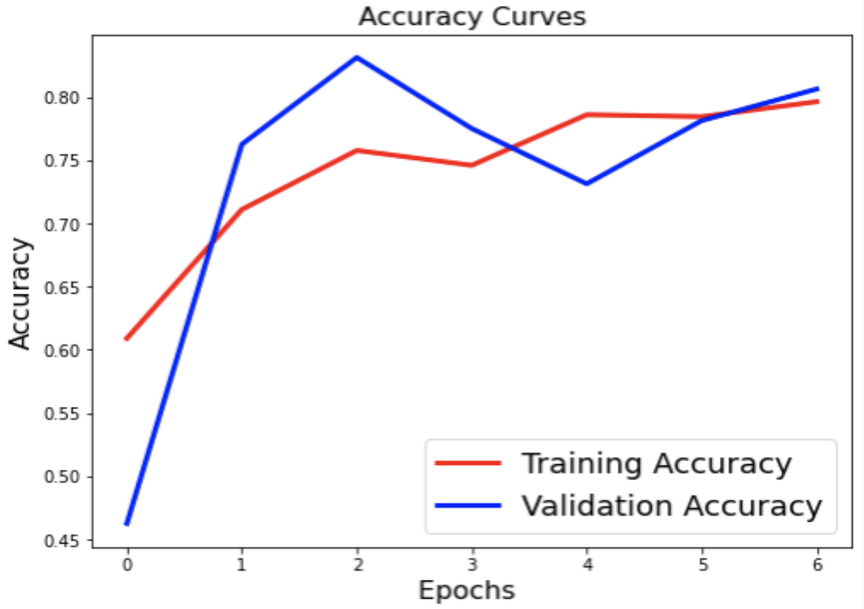


Optimisation: Batches and Epochs

- **Training Set:**
  - *Purpose:* Used to optimise the parameters (weights and biases) of the neural network.
  - *Process*: The model learns by adjusting its parameters to minimise the loss function on the training data through techniques like gradient descent and backpropagation.
- **Validation Set:**
  - *Purpose:* Provides an unbiased measurement of the model's performance.
  - *Process:* After training on the training set, the model's performance is evaluated on the validation set to tune hyperparameters, detect overfitting, and guide model selection.

converges.

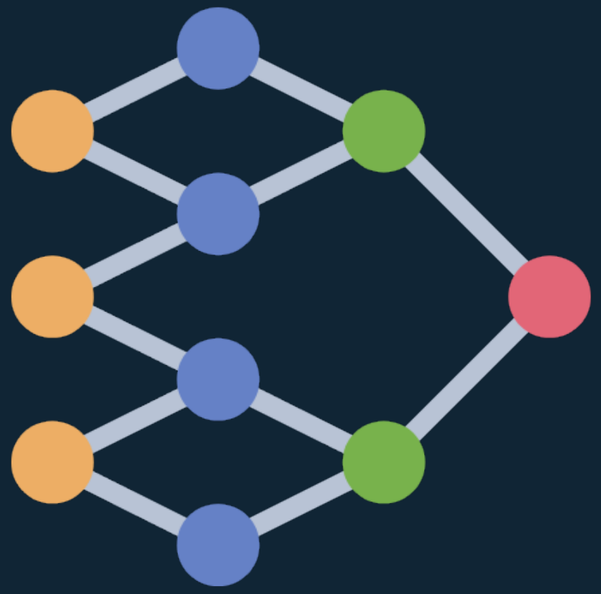




Which answer is correct?

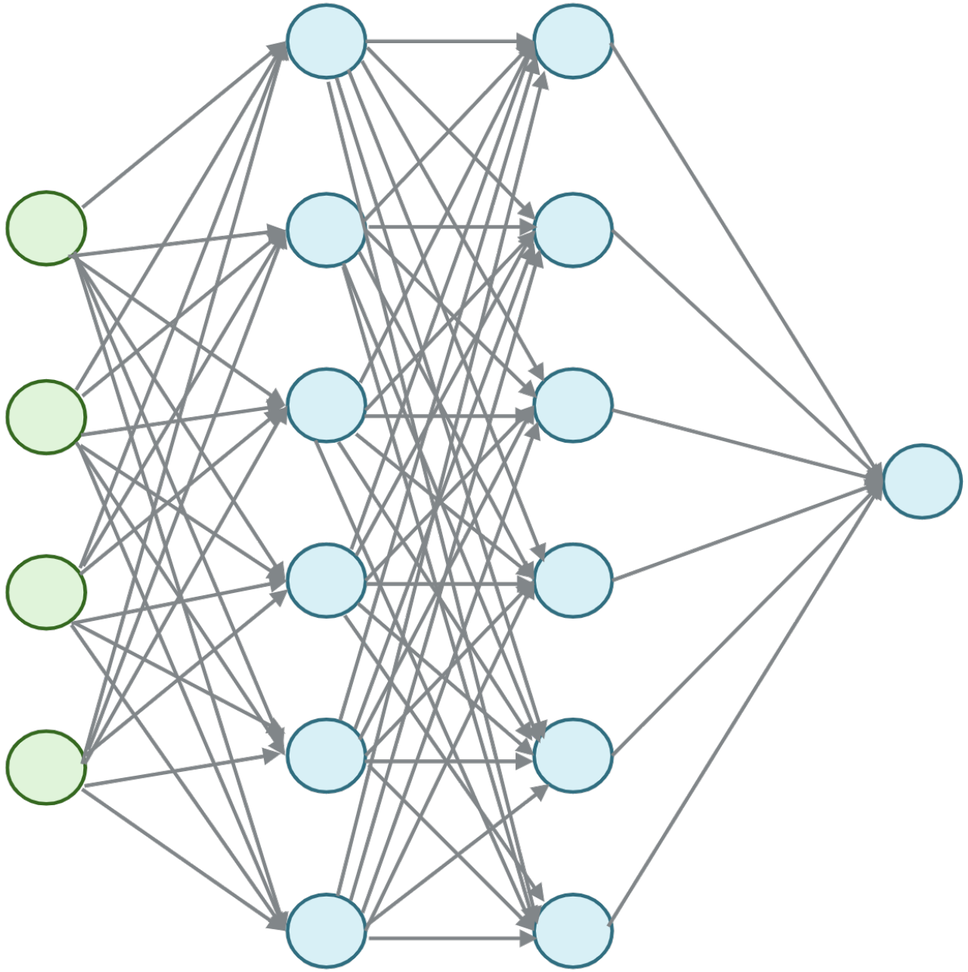
What is the primary goal of the backward pass in backpropagation?

- To calculate the output of the network
- To update weights using gradients
- To calculate the initial weights
- To normalise the input data



Classification & Regression Variants

Loss and Activation Functions: Regression



Neural Networks for Regression: Predicts continuous outcomes based on input features.

→ outputs a continuous value instead of a class label.

Loss function options: mean_squared_error

mean_absolute_error

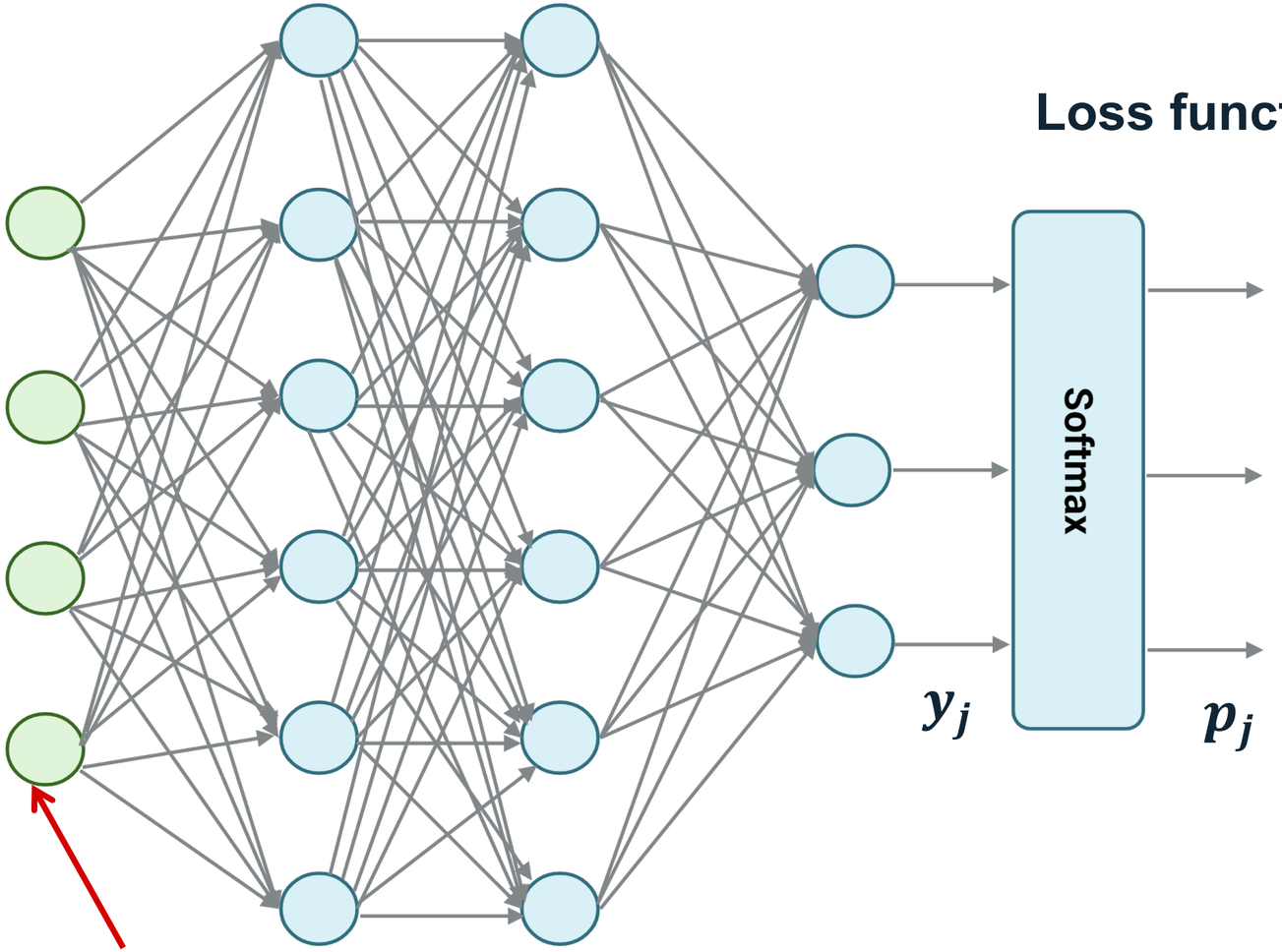
Activation function for final node:

Can be different to the rest of the nodes, and depends out form we want output to take:

- None (unconstrained): Output can take any continuous value.
- ReLU (positive range): Ensures output is non-negative.
- Sigmoid (limited range): onstrains the output to a range between 0 and 1.

Loss and Activation Functions: Classification

Neural Networks for Classification: Predicts class labels based on input features.



Loss function options: binary_crossentropy

Measures performance on 2 classes.

categorical_crossentropy

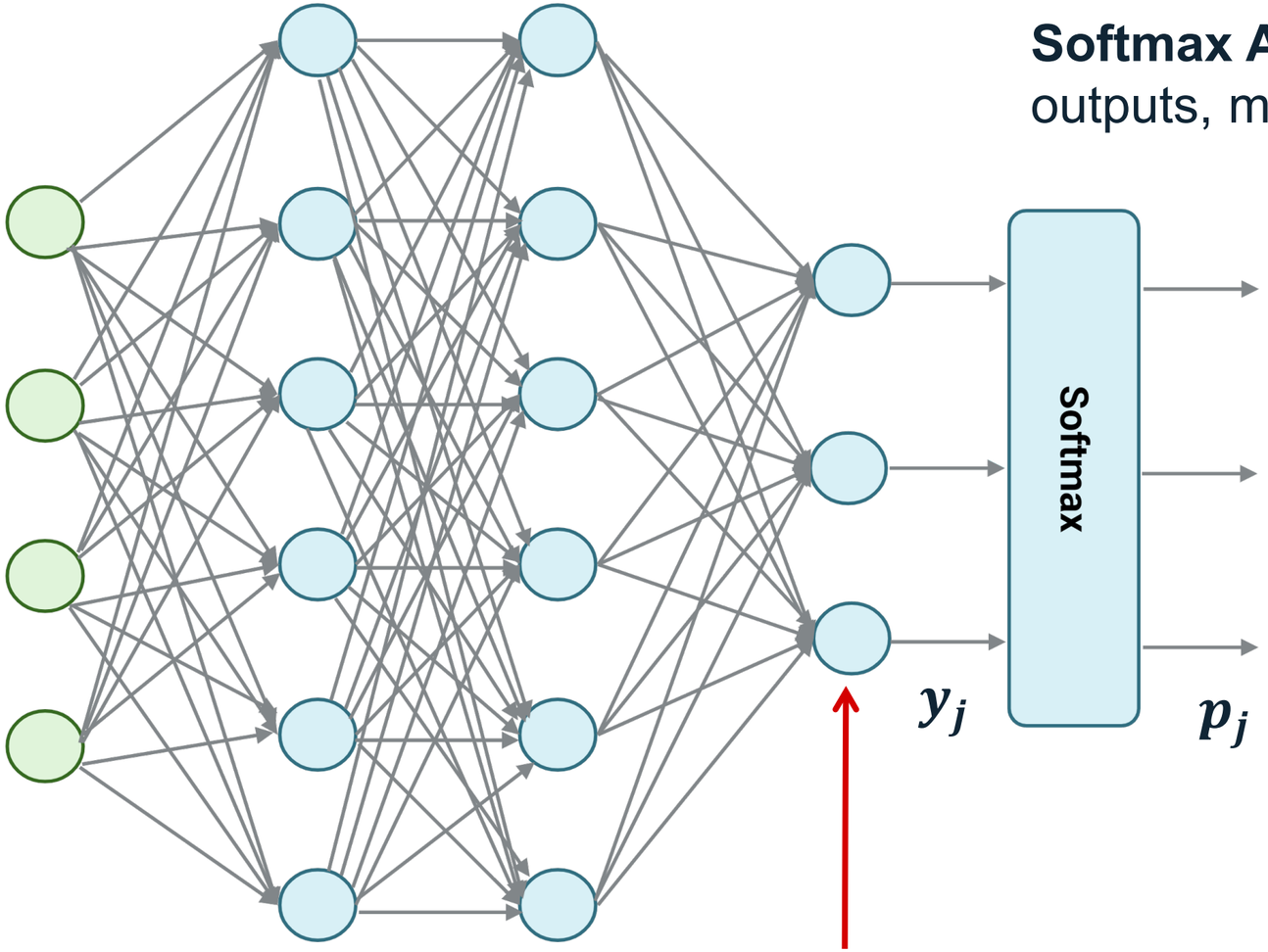
Measures performance for multi-class classification when we have one-hot encoded data.

sparse_categorical_crossentropy

Measures performance for multi-class classification when we have integer labels.

One-hot representation: Transforms categorical labels into a binary, suitable for neural network.

Loss and Activation Functions: Classification



Output layer: Equal to the number of classes in the classification problem.

Softmax Activation: Converts logits into probability-like outputs, making it easy to interpret the classification results:

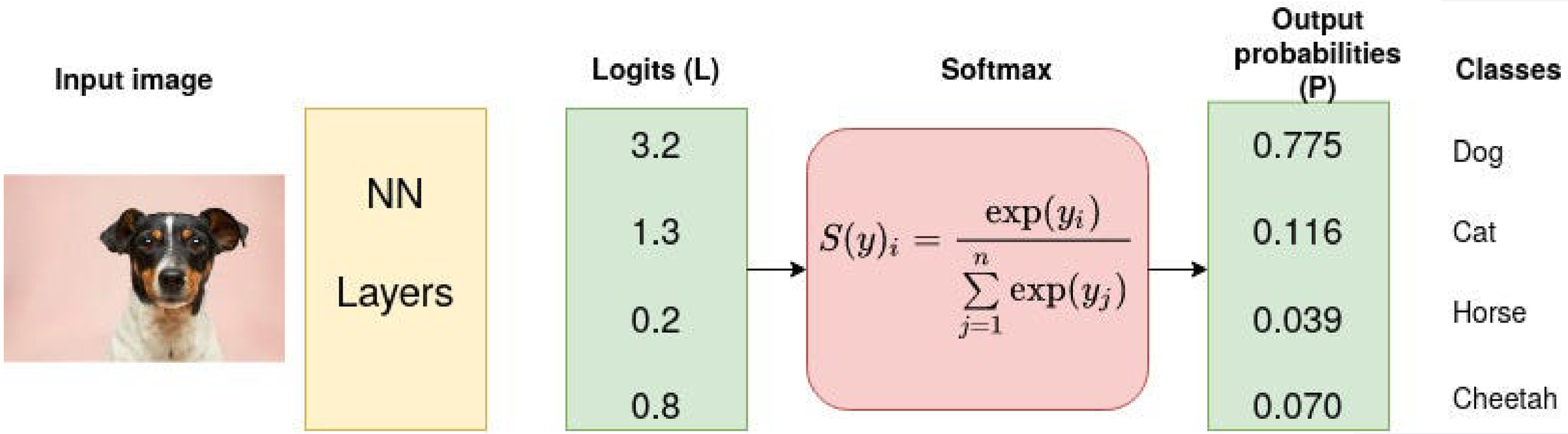
- *Close to 0:* Indicates a low probability of belonging to a particular class.
- *Close to 1:* Indicates a high probability of belonging to a particular class.

Activation function for final node:

$$Softmax\text{: } p_j = \frac{exp(y_j)}{\sum_k exp(y_k)}$$

One-hot representation: Transforms output into one-hot representation/binary vector.

Loss and Activation Functions: Classification



Loss and Activation Functions: Classification

- In some cases, it is beneficial to have probability-like outputs instead of strict class assignments.
- Particularly useful when a data point may belong to multiple classes to some extent.
  - Example: In medical imaging, such as brain scans, each voxel (a 3D pixel) in the image might not clearly belong to a single tissue type.
  - A voxel could represent a mixture of different tissue types, such as white matter and gray matter.
  - In this case, it is useful to have the model output probabilities indicating the likelihood of each tissue type.
  - This allows for more nuanced predictions and better understanding of the data.

Which answer is correct?

Which of the following is a common loss function used in regression tasks?

- Binary Crossentropy
- Categorical Crossentropy
- Mean Squared Error
- Sparse Categorical Crossentropy

Summary

1. **Components of a Neural Network**
   - Neurons: non-linearities and activation functions
   - Network architecture
   - Parameters: connections, weights and biases
   - Number of parameters
   - Deep Neural Networks
2. **Loss functions, epochs, batches, optimisers and training**
3. **Classification and regression variants**

Join at menti.com | use code 52 47 56 0

Instructions

Go to

www.menti.com

Enter the code

52 47 56 0

Or use QR code

Questions?

dhani.dharmaprani@adelaide.edu.au