AI Master class using Matlab

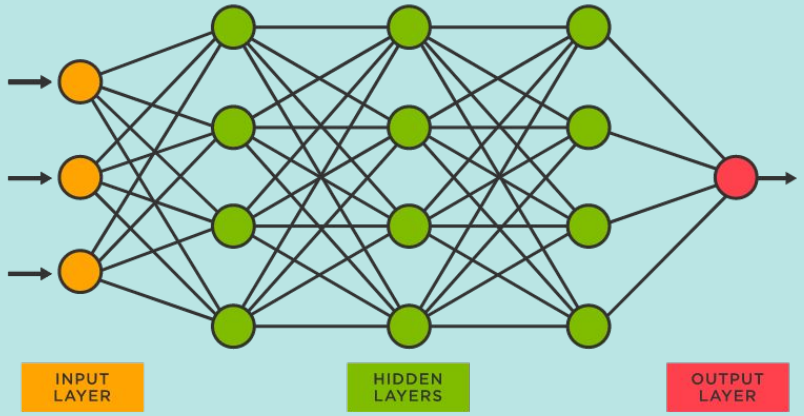
RESEARCH GROUP

Neural Network Design Using MATLAB

RESEARCH GROUP

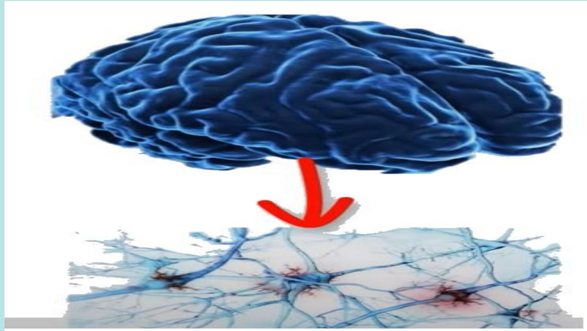
What is Neural network

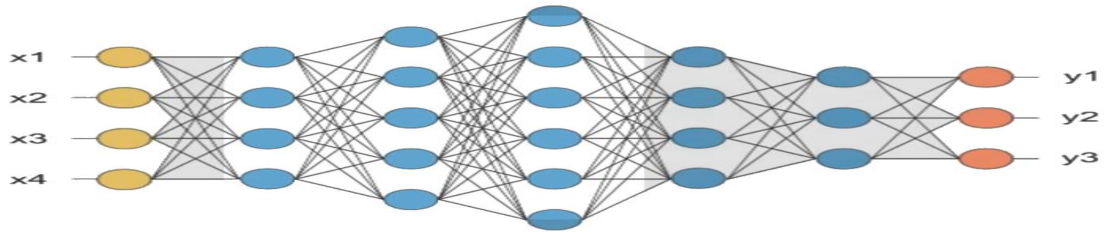
A **neural network** is a series of algorithms that endeavours to recognize underlying relationships in a set of data through a process that mimics the way the human brain operates

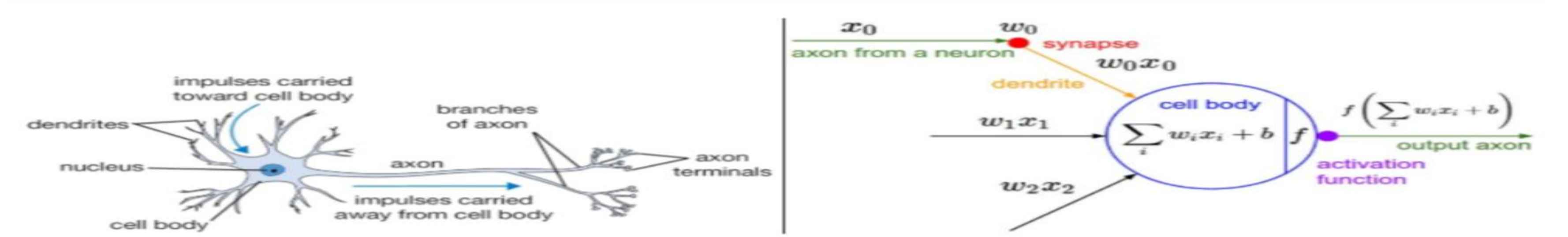


RESEARCH GROUP

Brain Vs Neural Network







A cartoon drawing of a biological neuron (left) and its mathematical model (right).

RESEARCH GROUP

Brain Vs Neural Network

- Neural networks are computer models inspired by the human brain.
- Both the human brain and neural networks process information using neurons and synapses.
- Neural networks can be used to simulate the behavior of the human brain in certain tasks.
- Advancements in neural networks can lead to a better understanding of how the human brain works.

RESEARCH GROUP

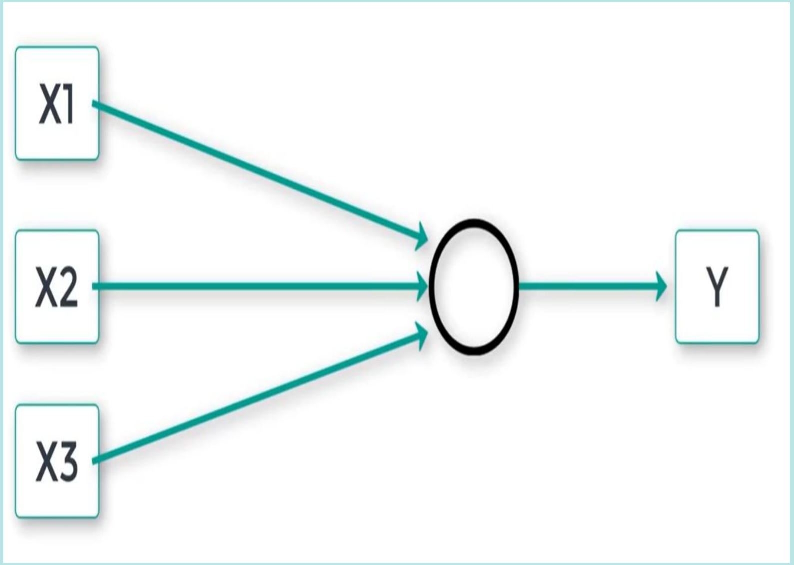
Brain Vs Neural Network

- ❖ The human brain is a biological organ, while neural networks are computer models.
- ❖ The human brain has billions of neurons, while neural networks typically have a much smaller number of artificial neurons.
- ❖ The human brain is capable of complex, adaptive behaviors, while neural networks are typically limited to specific tasks.
- ❖ The human brain has a degree of consciousness and self-awareness, which is not present in current neural network models.

RESEARCH GROUP

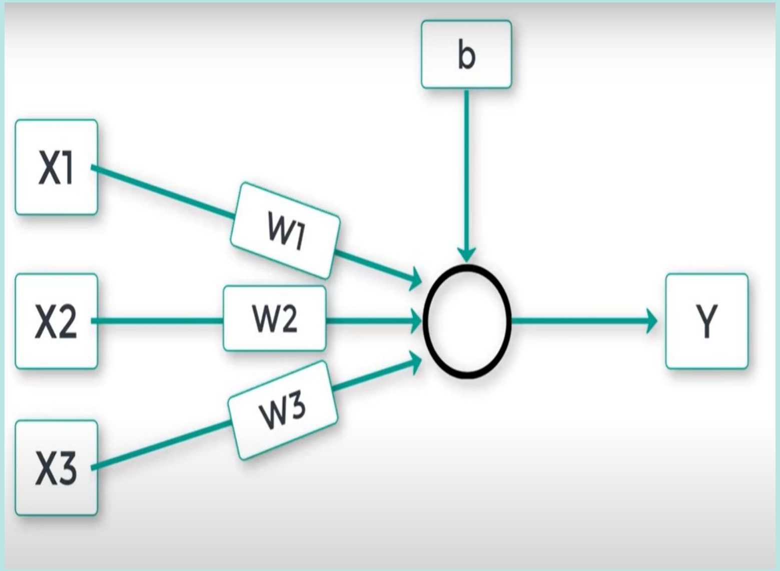
Single Layer Neural Network

- A single layer neural network is a simple type of neural network consisting of a single layer of artificial neurons.
- It takes in input values, applies weights, and passes the weighted sum through an activation function to produce an output.
- Single layer neural networks are commonly used for simple classification problems such as binary classification or linear regression, but have limitations in solving complex problems as they can only model linear decision boundaries.



RESEARCH GROUP

Single Layer Neural Network

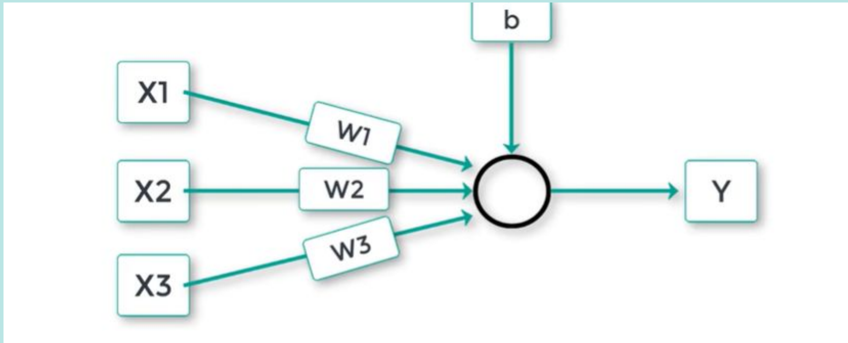


Single layer Neural Network

- ➢ Bias is an adjustable constant in a neural network that allows for the shifting of the activation function and accounts for errors and inaccuracies in the data.

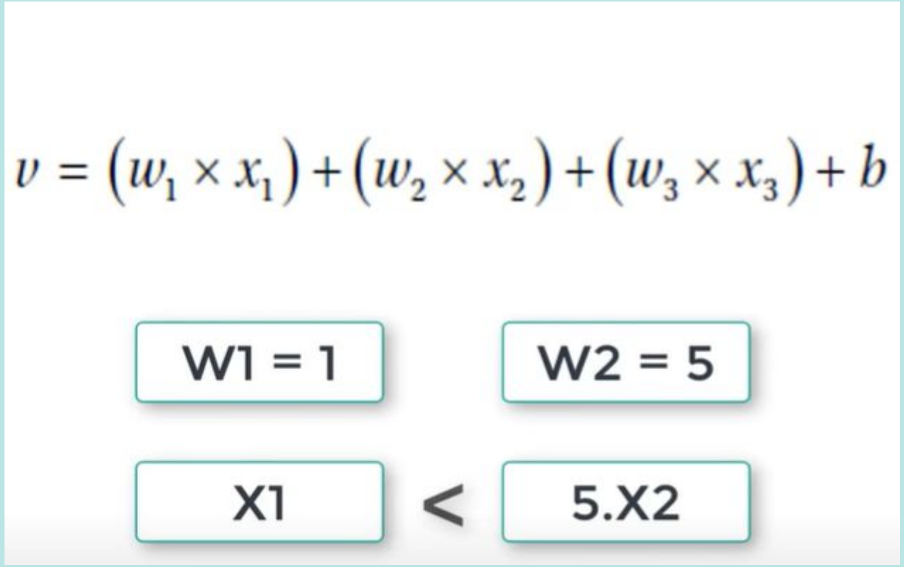
- ➢ Weight determines the strength of the connection between two neurons and represents the importance of a particular input to the output of the neural network.

- ➢ The weights are adjusted during training to optimize the model's ability to accurately predict outputs based on inputs.



$$v = \boxed{(w_1 \times x_1)} + \boxed{(w_2 \times x_2)} + \boxed{(w_3 \times x_3)} + b$$

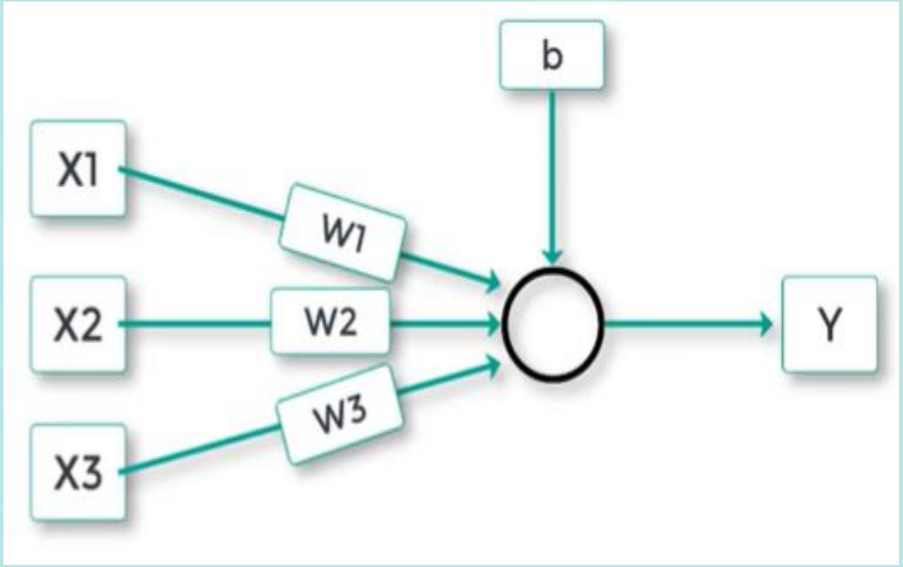
Single Layer Neural Network



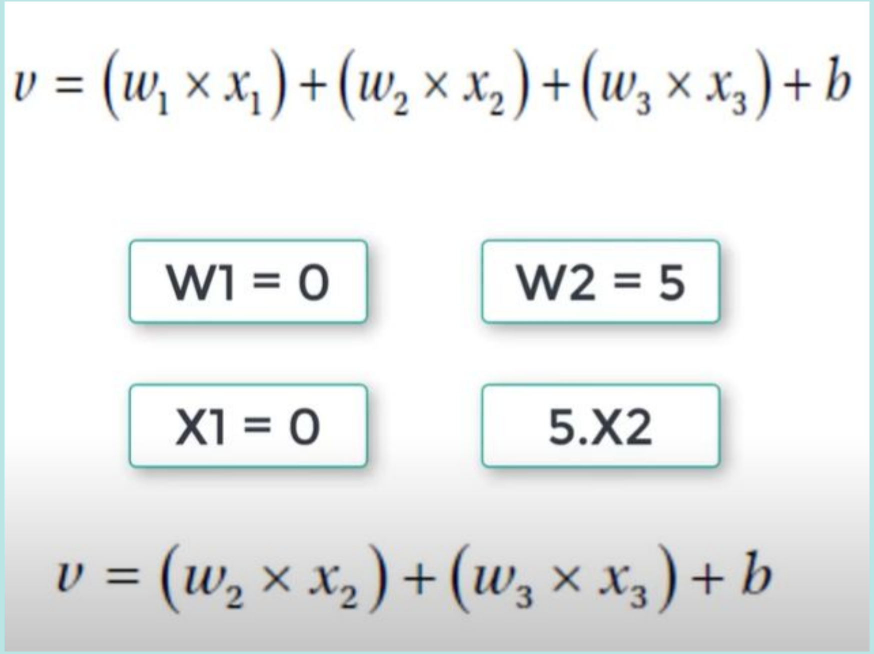
$$v = (w_1 \times x_1) + (w_2 \times x_2) + (w_3 \times x_3) + b$$

$\boxed{W1 = 1}$ $\boxed{W2 = 5}$

$\boxed{X1} < \boxed{5.X2}$



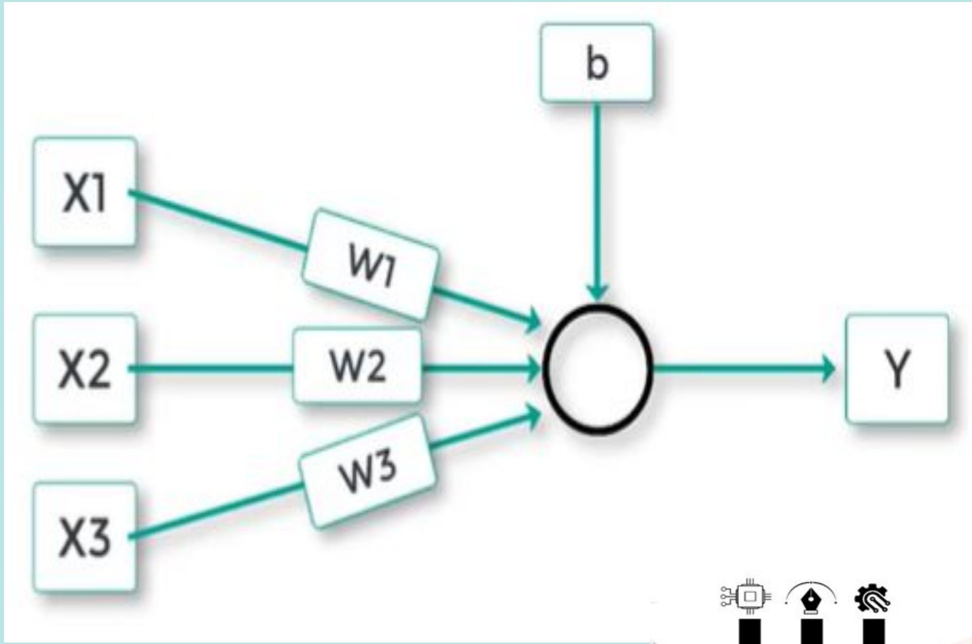
Single layer Neural Network



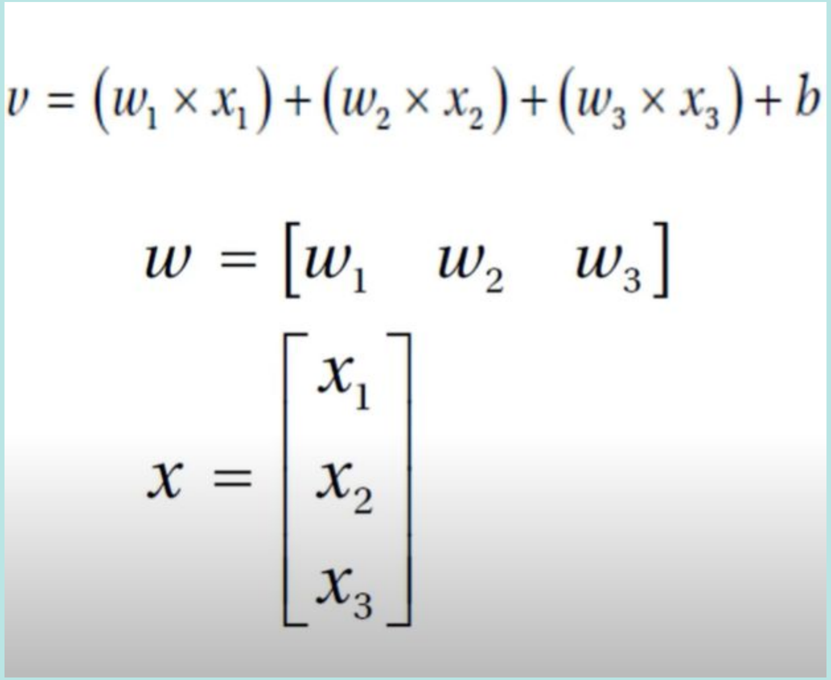
$$v = (w_1 \times x_1) + (w_2 \times x_2) + (w_3 \times x_3) + b$$

W1 = 0	W2 = 5
X1 = 0	5.X2

$$v = (w_2 \times x_2) + (w_3 \times x_3) + b$$



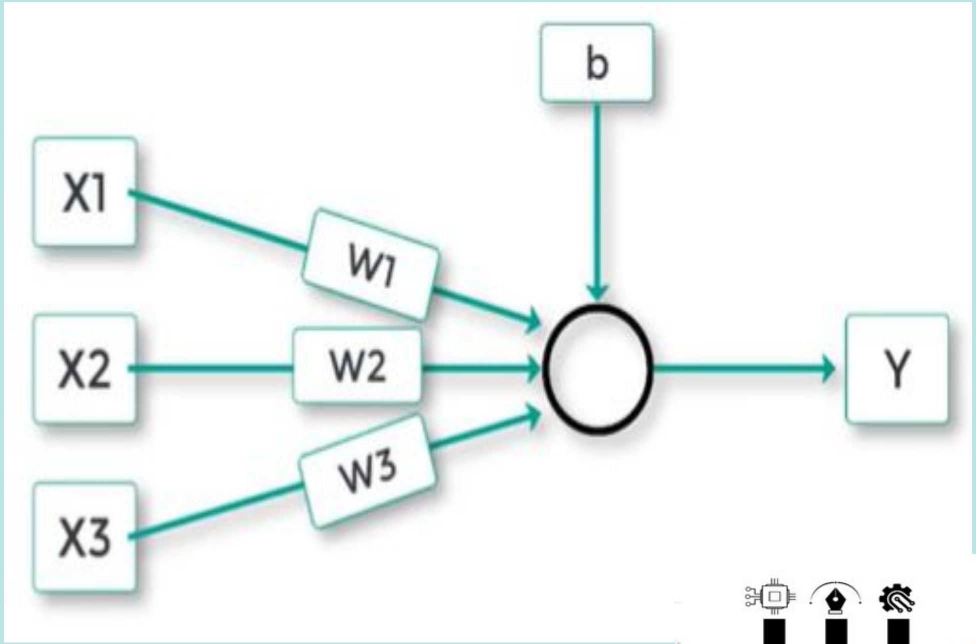
Single Layer Neural Network



$$v = (w_1 \times x_1) + (w_2 \times x_2) + (w_3 \times x_3) + b$$

$$w = \begin{bmatrix} w_1 & w_2 & w_3 \end{bmatrix}$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$



RESEARCH GROUP

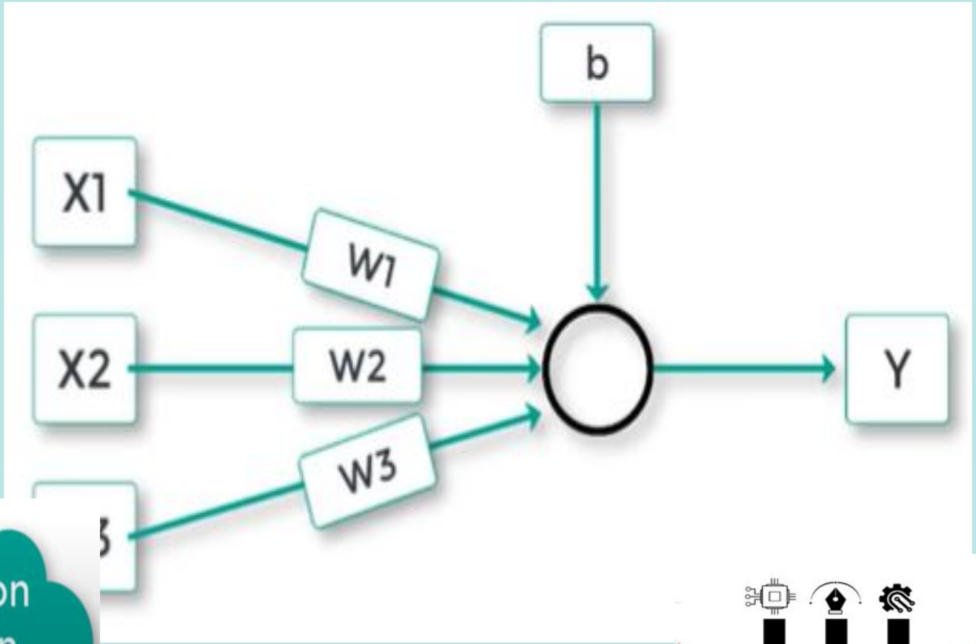
Single layer Neural Network

$$v = (w_1 \times x_1) + (w_2 \times x_2) + (w_3 \times x_3) + b$$

$$w = [w_1 \quad w_2 \quad w_3]$$

$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix}$$

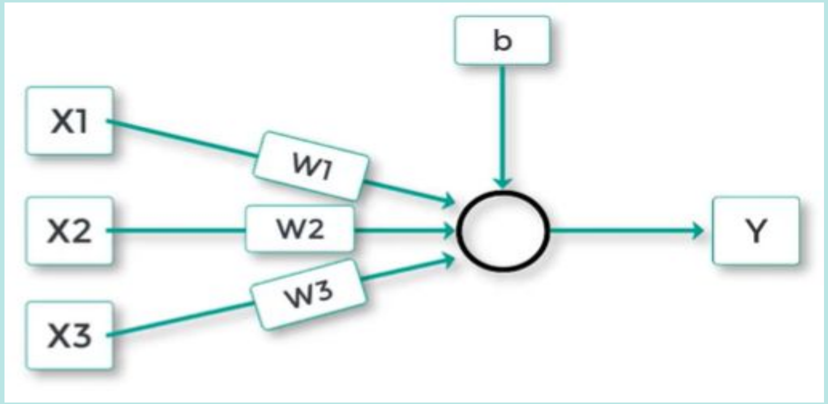
$$y = \varphi(v)$$



Single layer Neural Network

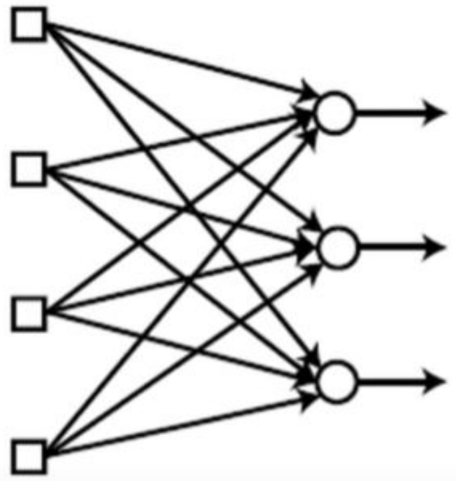
$$\begin{aligned} v &= w_1x_1 + w_2x_2 + w_3x_3 + b \\ &= wx + b \end{aligned}$$

$$w = [w_1 \quad w_2 \quad w_3]$$



$$x = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} \qquad y = \varphi(v) = \varphi(wx + b)$$

Single Layer Network


Single-layer Neural Network

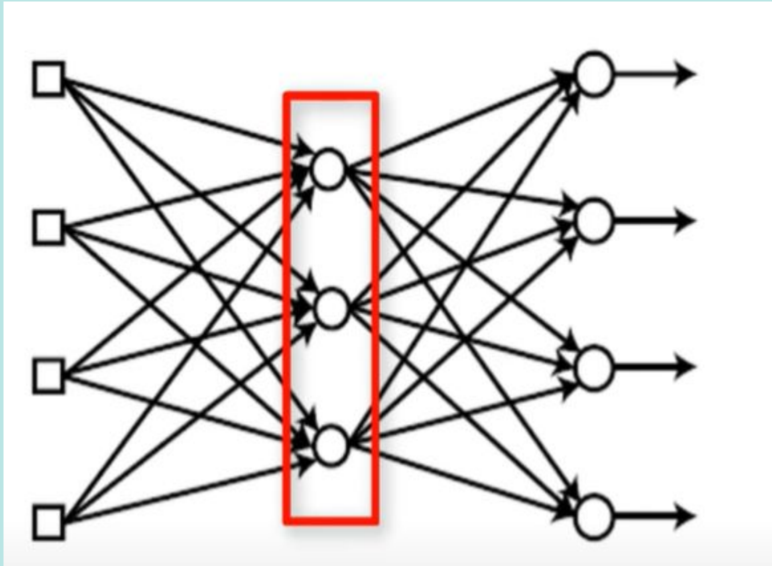
Shallow or Multilayer Neural network

- A shallow neural network has only one layer of neurons, while a multilayer neural network has two or more layers of neurons

- Shallow neural networks are used for simpler classification tasks, while multilayer neural networks are used for more complex tasks such as image recognition or natural language processing

- Shallow neural networks can be implemented using the perceptron algorithm, while multilayer neural networks are typically implemented using feedforward neural networks or convolutional neural networks

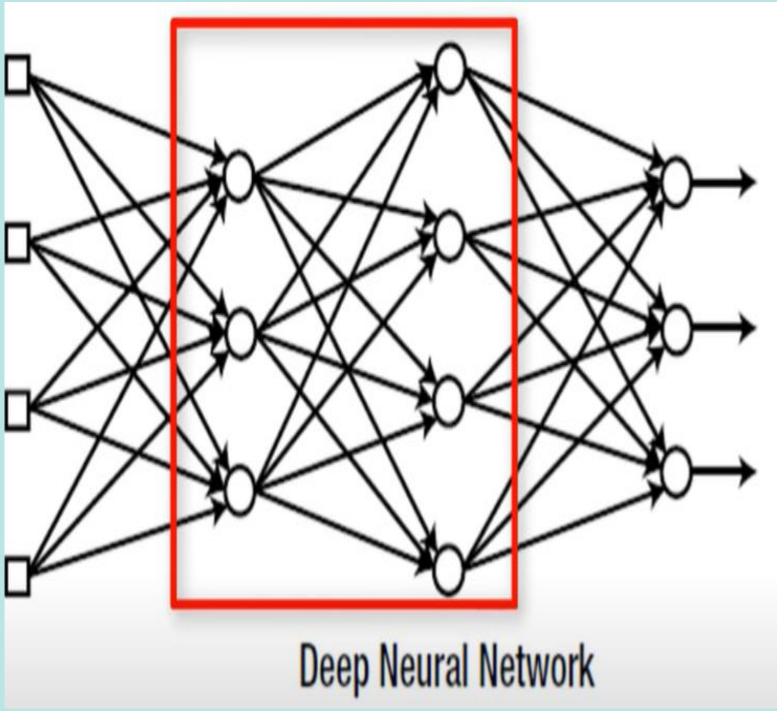
- In both types of neural networks, the input layer receives input data, the output layer produces the final result, and the hidden layers (in multilayer neural networks) process the data through a series of transformations


(Shallow) Multi-layer Neural Network

RESEARCH GROUP

Deep Neural Networks

- Deep neural networks are a type of artificial neural network with many layers, typically ranging from tens to hundreds.
- Deep neural networks are used in a variety of applications such as machine vision, natural language processing, and speech recognition.
- Convolutional neural networks (CNNs) are used for image classification and recognition tasks.
- Recurrent neural networks (RNNs) are used for processing sequential data, such as natural language text or time-series data.
- Deep reinforcement learning is a combination of deep learning and reinforcement learning and is used to train agents to perform complex tasks in environments.



RESEARCH GROUP

Deep neural networks

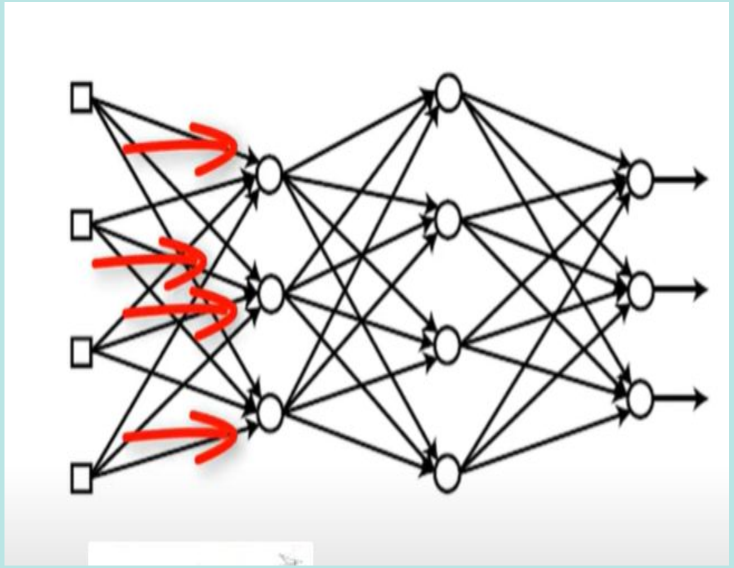
- Deep neural networks use multiple layers of interconnected neurons to model complex relationships between input and output data.

- Each layer in the network learns to extract increasingly abstract features from the input data.

- During training, the weights and biases of the network are adjusted using an optimization algorithm such as backpropagation.

- Deep neural networks are currently being used in a wide range of applications, including image and speech recognition, natural language processing, and robotics.

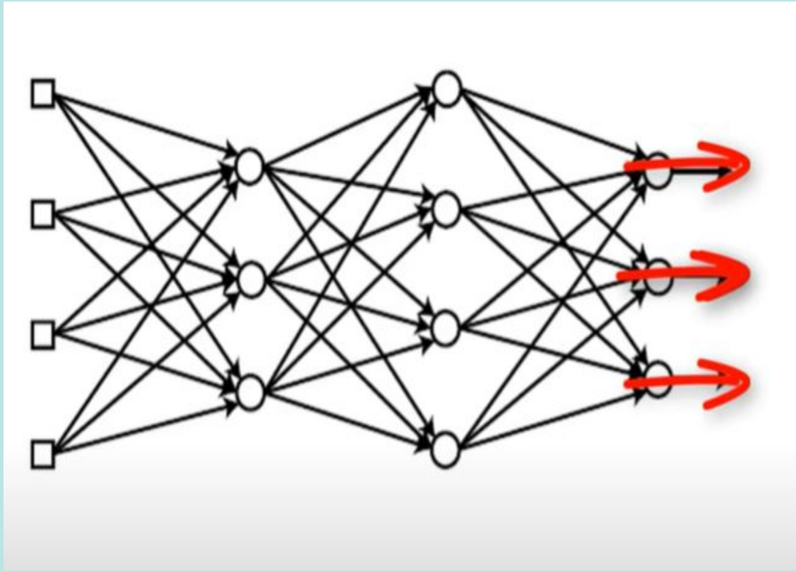
- While powerful, deep neural networks can be challenging to train and require large amounts of labeled data and computational resources.



RESEARCH GROUP

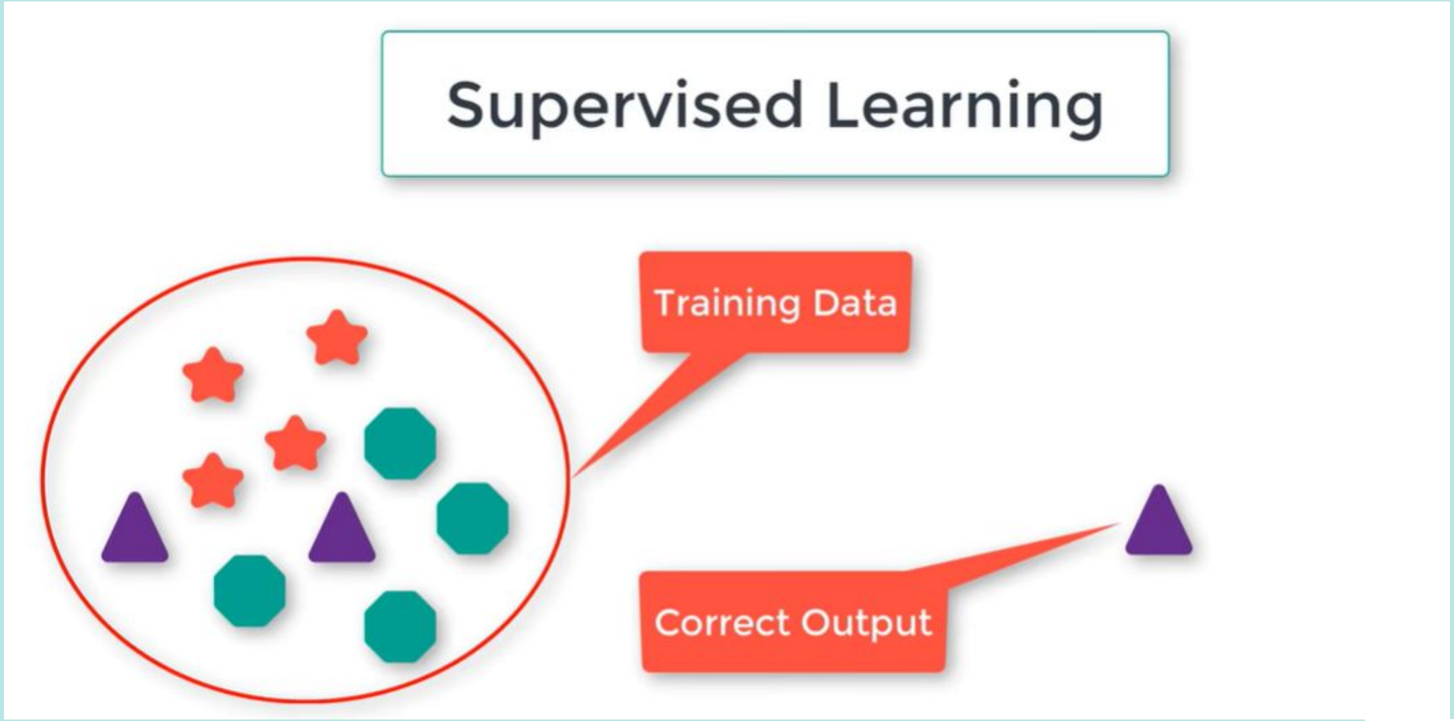
Deep neural networks

- The network is first trained on a large dataset of labeled images of cats and dogs.
- To classify a new image, the preprocessed image is passed through the network, with each neuron in the hidden layers extracting increasingly abstract features.
- During training, the weights and biases of the network are adjusted using an optimization algorithm such as backpropagation.
- To classify a new image, the network outputs the label corresponding to the class with the highest output probability.
- Deep neural networks can be very effective at image classification tasks such as distinguishing between cats and dogs, but require a large amount of labeled data and computational resources.

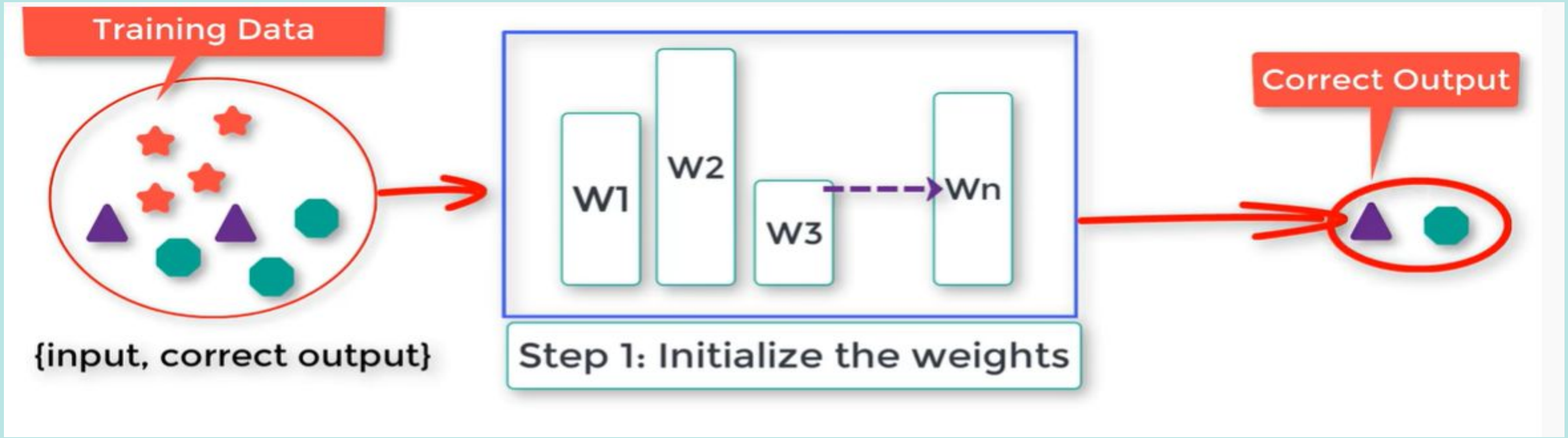


RESEARCH GROUP

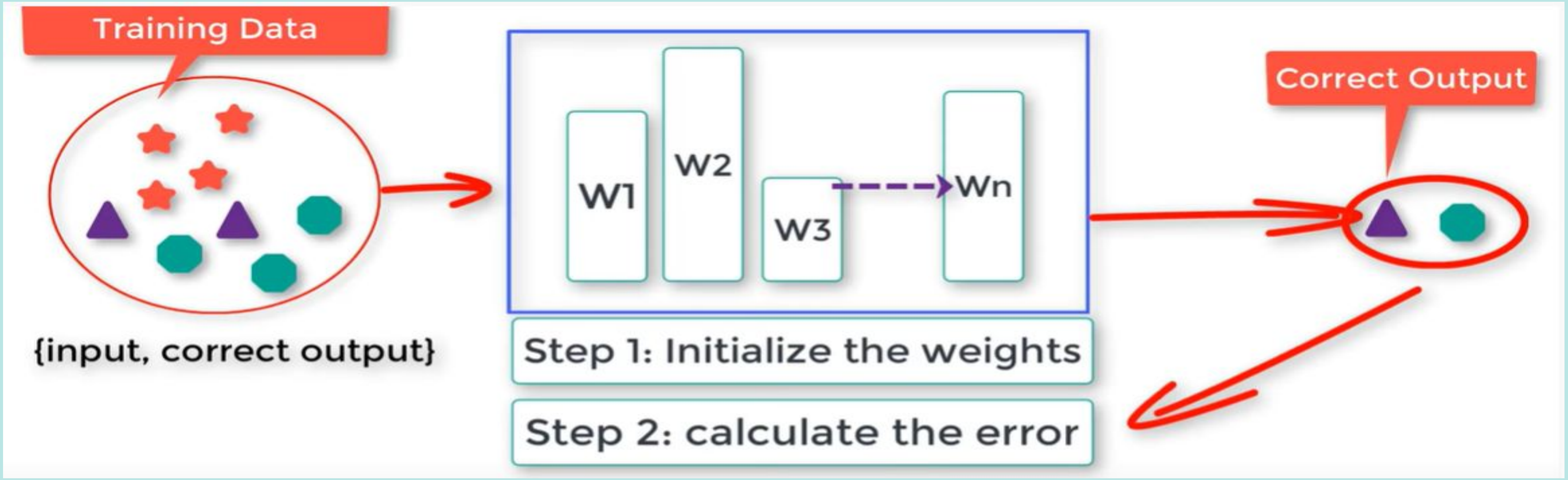
Supervised Learning



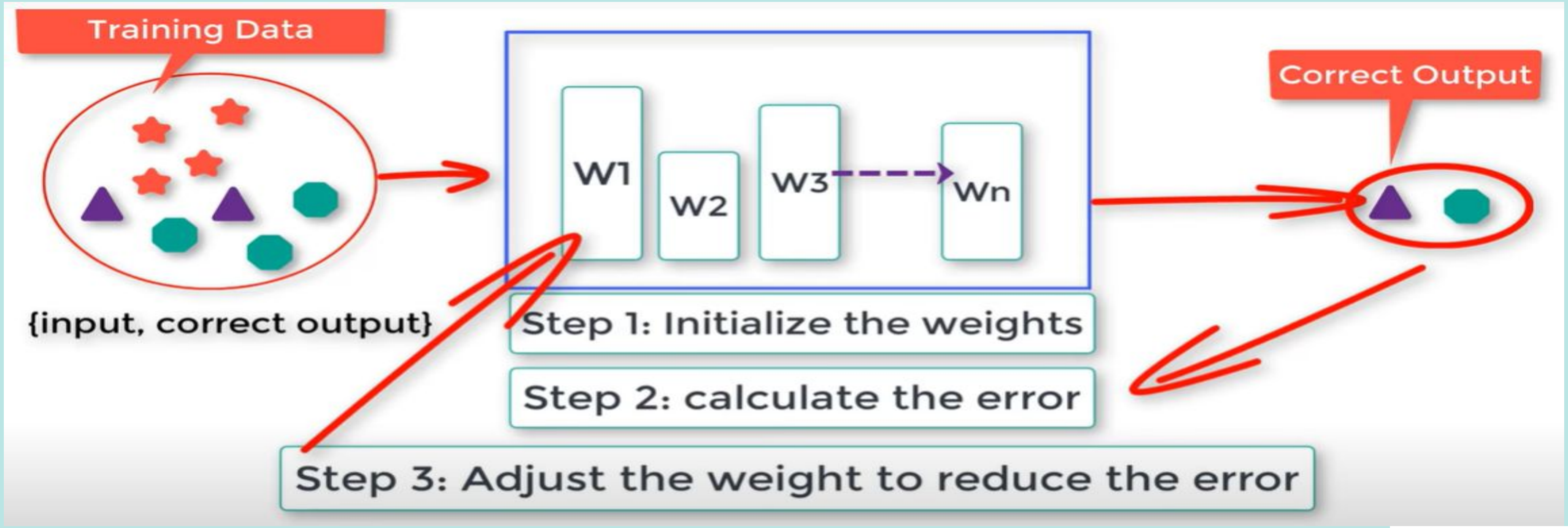
Step 1 Initialize the weights



Step 2 – Calculate the error



Step 3- Adjust the weight to reduce the error


Training Data
{input, correct output}
W1
W2
W3
Wn
Correct Output
Step 1: Initialize the weights
Step 2: calculate the error
Step 3: Adjust the weight to reduce the error

RESEARCH GROUP

Step 1: Initialize the weights

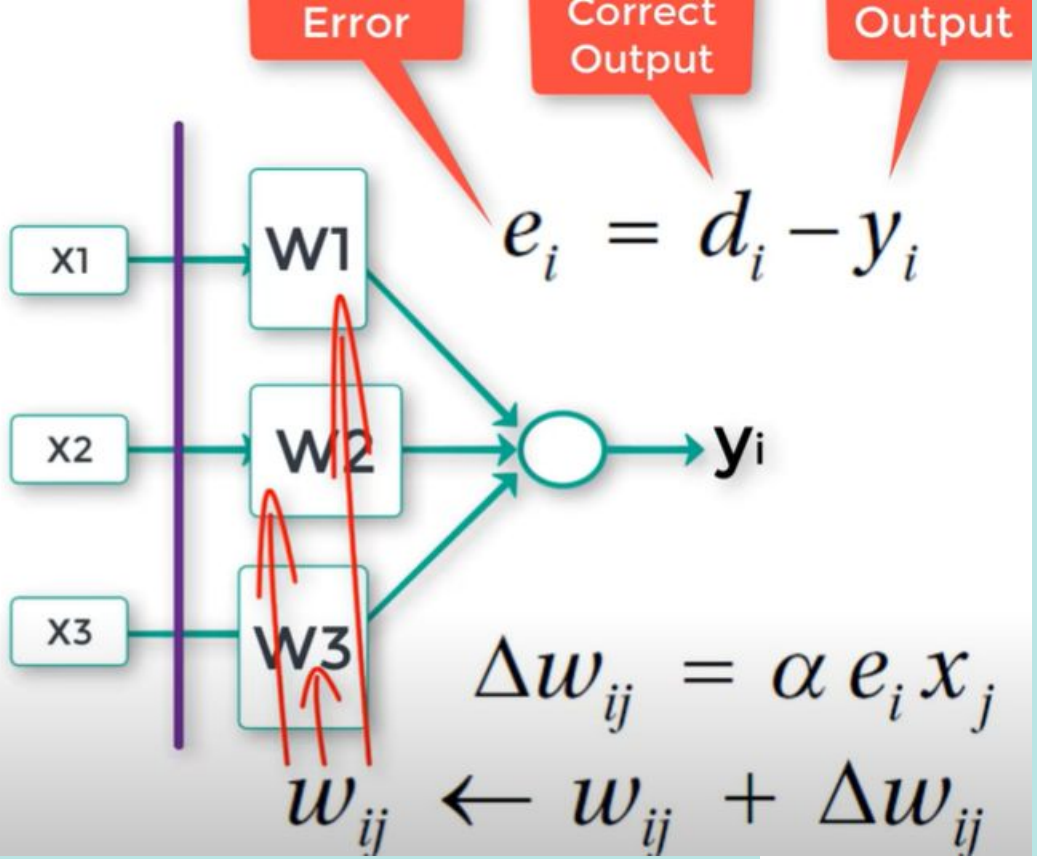
Step 2: Calculate the error from the difference between output and correct output

Step 3: Calculate the weights updates

Step 4: Adjust the weight updates

Step 5: Repeat step 2 to 4 for all training data

Step 6: Repeat step 2 to 5 until the error reaches an acceptable level

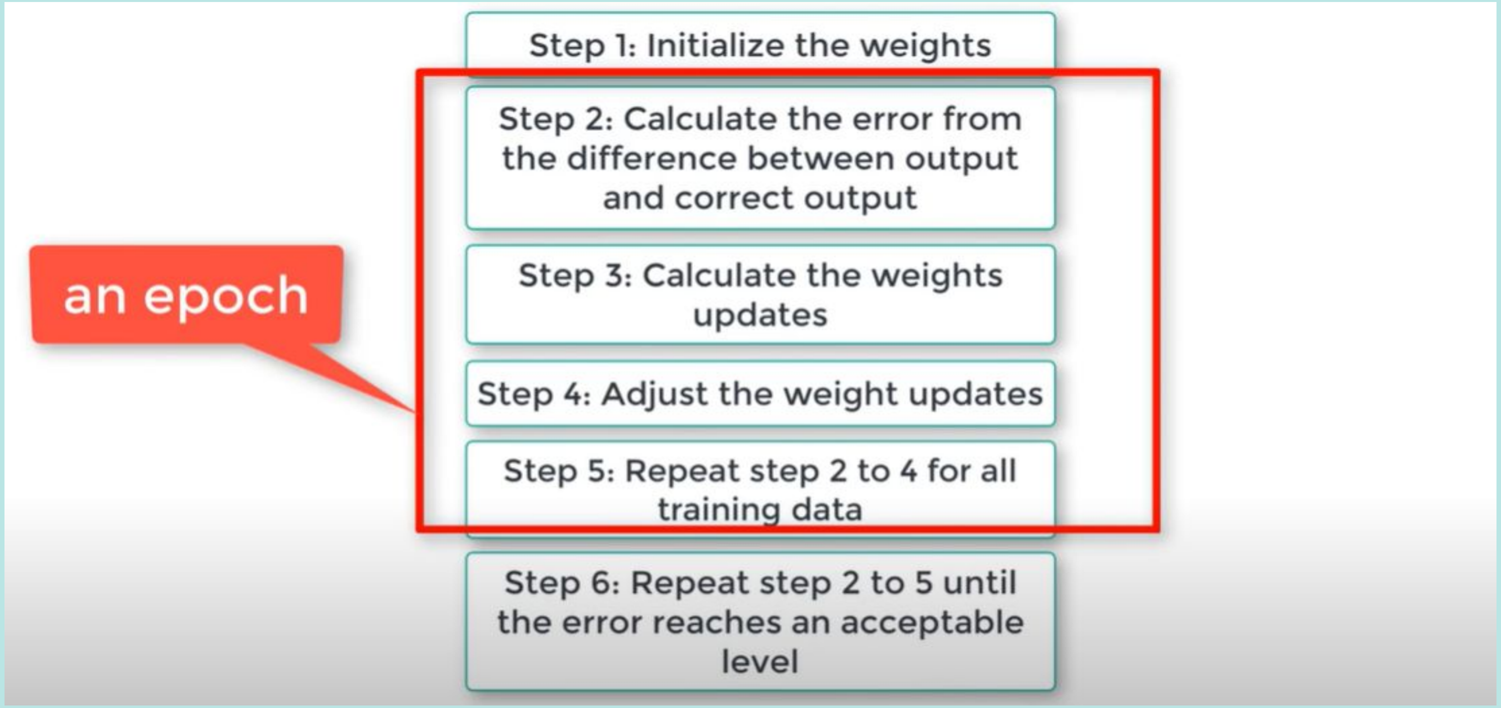


$$e_i = d_i - y_i$$

$$\Delta w_{ij} = \alpha e_i x_j$$

$$w_{ij} \leftarrow w_{ij} + \Delta w_{ij}$$

Epoch



Step 1: Initialize the weights

Step 2: Calculate the error from the difference between output and correct output

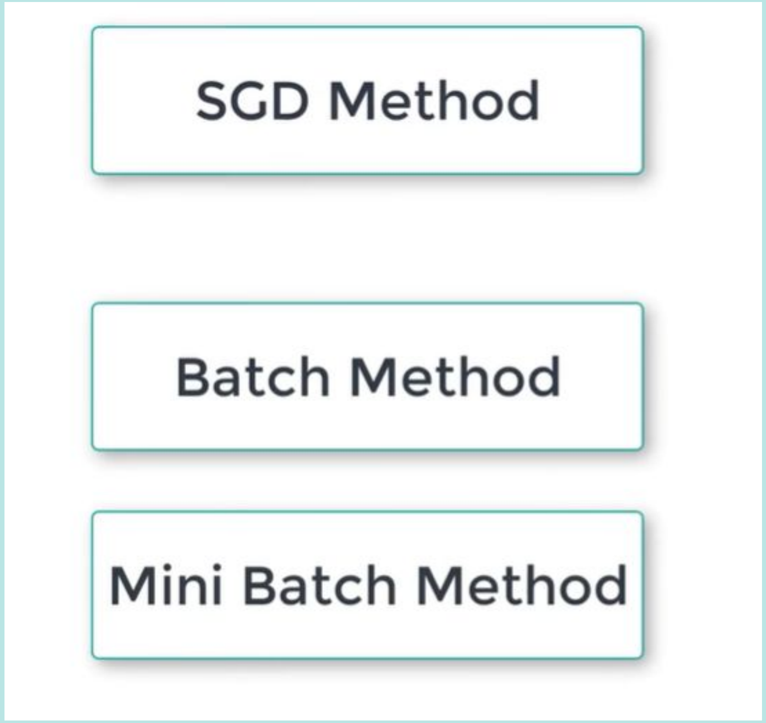
Step 3: Calculate the weights updates

Step 4: Adjust the weight updates

Step 5: Repeat step 2 to 4 for all training data

Step 6: Repeat step 2 to 5 until the error reaches an acceptable level

Updating The Weights



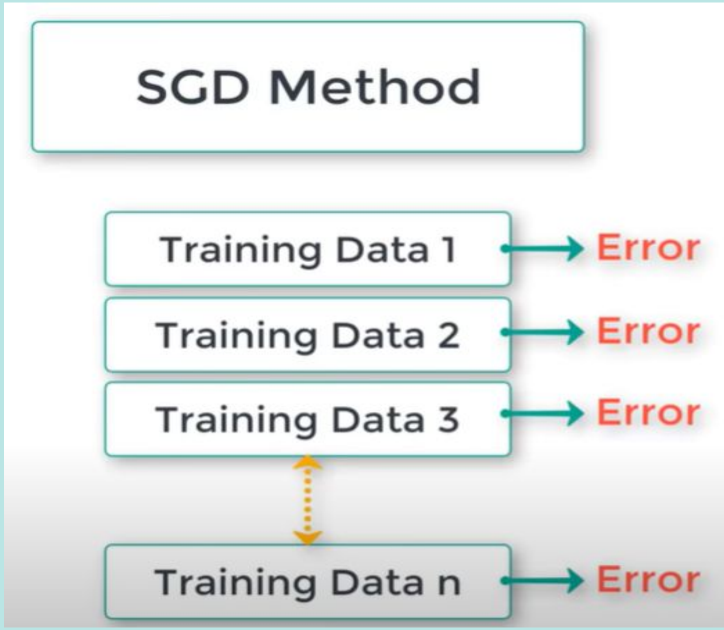
SGD Method

Batch Method

Mini Batch Method

Stochastic Gradient Descent Method

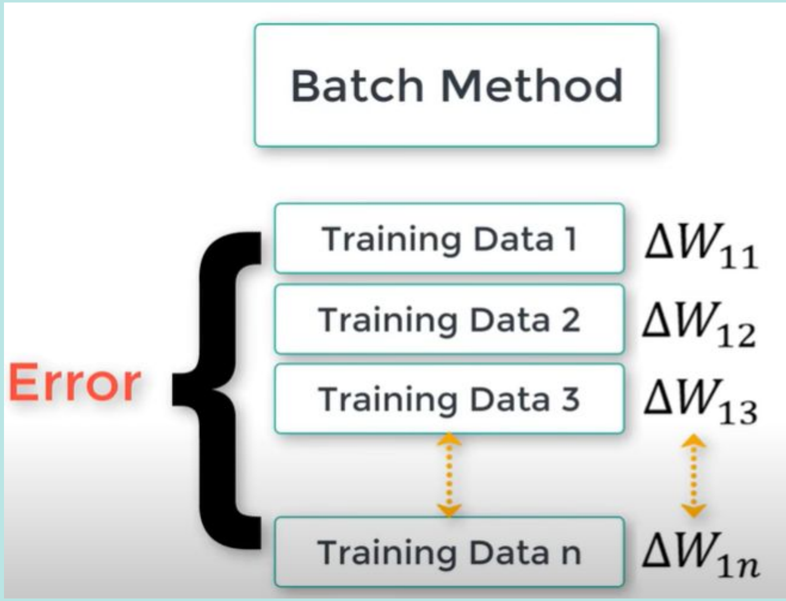
- SGD is an optimization algorithm used to minimize the cost function or loss function in machine learning.
- It updates the model parameters using a small subset of the training data, called a mini-batch.
- It computes the gradient of the cost function with respect to the parameters using the mini-batch.
- It updates the parameters by subtracting the gradient from the current parameter values multiplied by a learning rate.
- SGD converges faster than standard Gradient Descent but may converge to a local minimum instead of the global minimum.
- Weight is updated N times in each epoch
- Random



RESEARCH GROUP

Batch Method

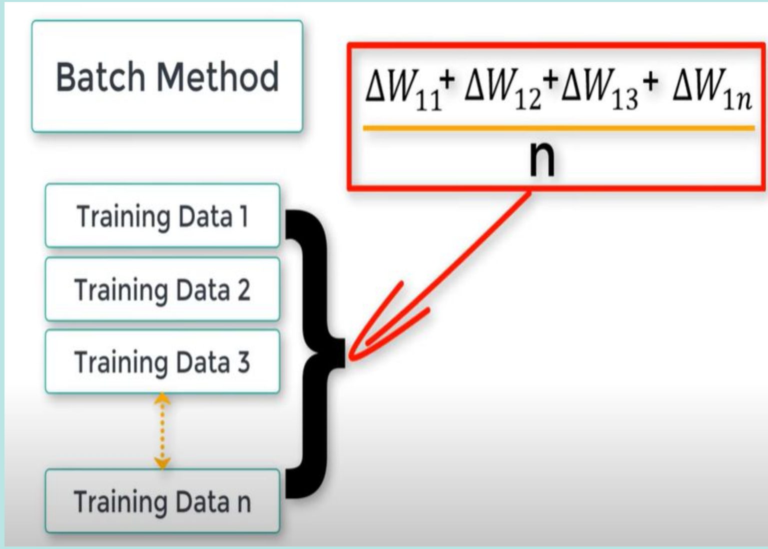
- *Batch methods are a class of optimization algorithms used in neural networks to minimize the cost function or loss function.*
- *They use the entire training dataset to compute the gradient of the cost function with respect to the parameters and update the parameters accordingly.*
- *They are less noisy than Stochastic Gradient Descent (SGD) and are more likely to converge to the global minimum of the cost function.*
- *Batch methods are computationally expensive because they require computing the gradient using the entire training dataset.*
- *Examples of batch methods used in neural networks include Batch Gradient Descent, Conjugate Gradient, and L-BFGS.*



Batch method

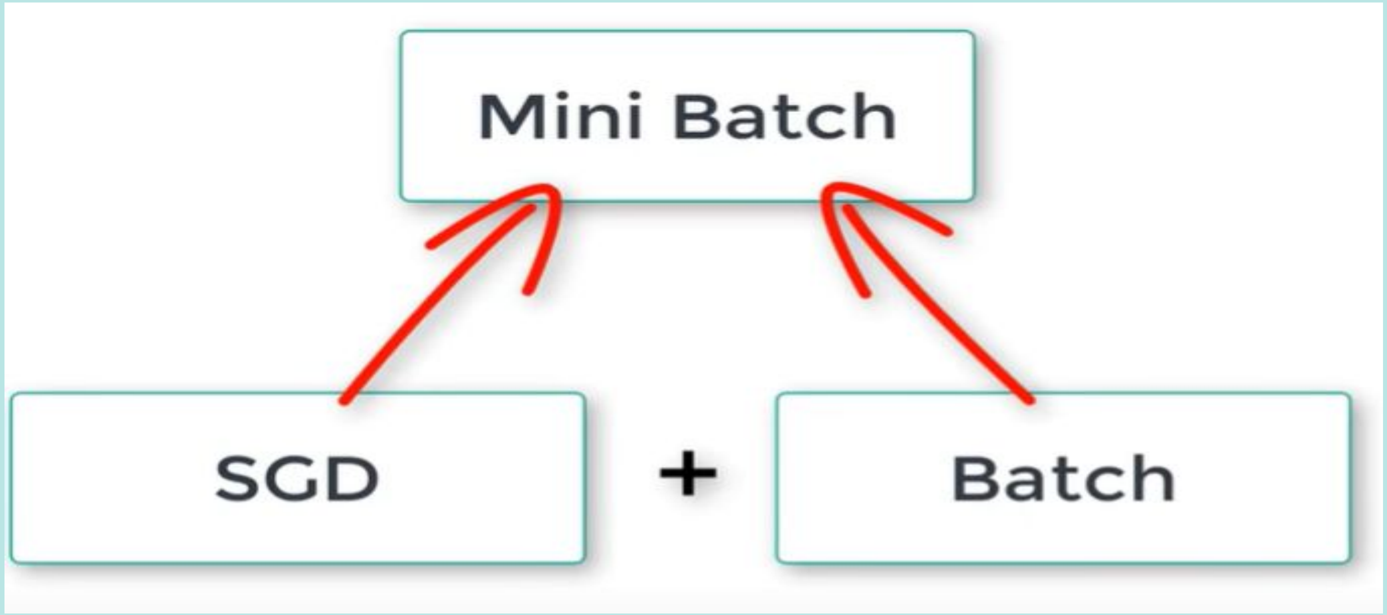
- Weight are updated only one times in each epoch

- Because of the average method the training takes long time in batch method


Batch Method
$\frac{\Delta W_{11} + \Delta W_{12} + \Delta W_{13} + \Delta W_{1n}}{n}$
Training Data 1
Training Data 2
Training Data 3
Training Data n

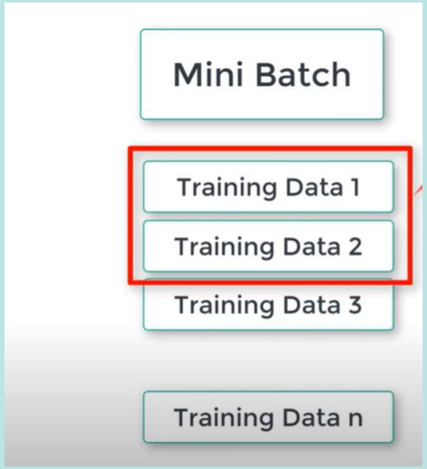
RESEARCH GROUP

Mini Batch Method



Mini Batch Method

- ➢ *Mini-batch method is a variant of Stochastic Gradient Descent (SGD) optimization algorithm.*
- ➢ *It updates the model parameters by computing the gradients of the cost function with respect to the parameters using a small random subset of the training data, called a mini-batch*
- ➢ *It is computationally faster and more memory-efficient than the batch method, which uses the entire training dataset*
- ➢ *The size of the mini-batch is a hyperparameter that can be tuned during training and is usually between 10 to 1000 examples.*



RESEARCH GROUP

Mini Batch SGD and Batch Method

Advantages of Mini-Batch Method:

- ➢ It is computationally faster and more memory-efficient than the batch method, which uses the entire training dataset
- ➢ It leads to more stable updates compared to the stochastic method
- ➢ It reduces the variance in the parameter updates compared to the stochastic method
- ➢ It allows for the use of parallel processing to speed up training
- ➢ It can converge more quickly than the stochastic method.

Advantages of (SGD):

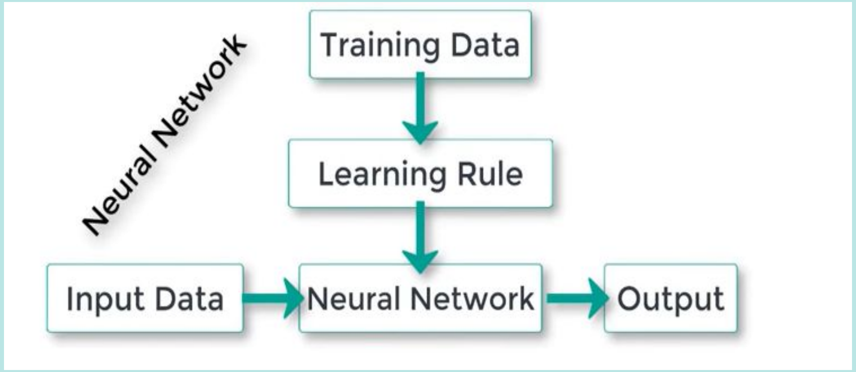
- ➢ It is computationally efficient and can handle large datasets
- ➢ It is less likely to get stuck in local minima
- ➢ It can update the parameters in every iteration, leading to fast convergence
- ➢ It can be used for online learning where the data keeps coming
- ➢ It can work well with noisy or sparse data.

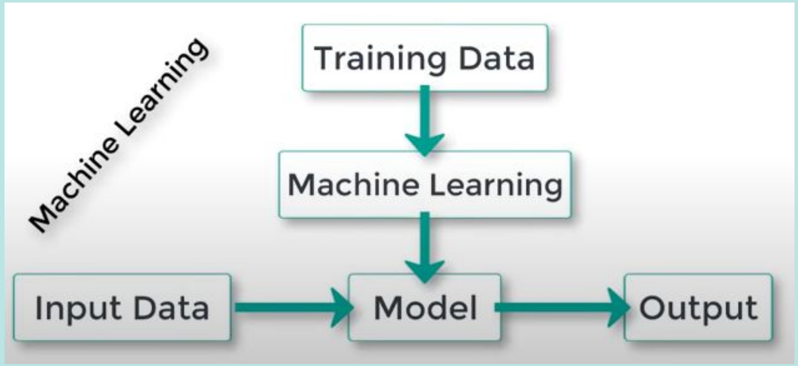
Advantages of Batch Method:

- ➢ It is more accurate and stable than the stochastic method
- ➢ It is guaranteed to converge to the global minimum for convex cost functions
- ➢ It requires fewer iterations to converge than the stochastic method
- ➢ It leads to smooth updates compared to the stochastic method
- ➢ It is less sensitive to the choice of learning rate than the stochastic method.

RESEARCH GROUP

Neural Network vs Machine Learning

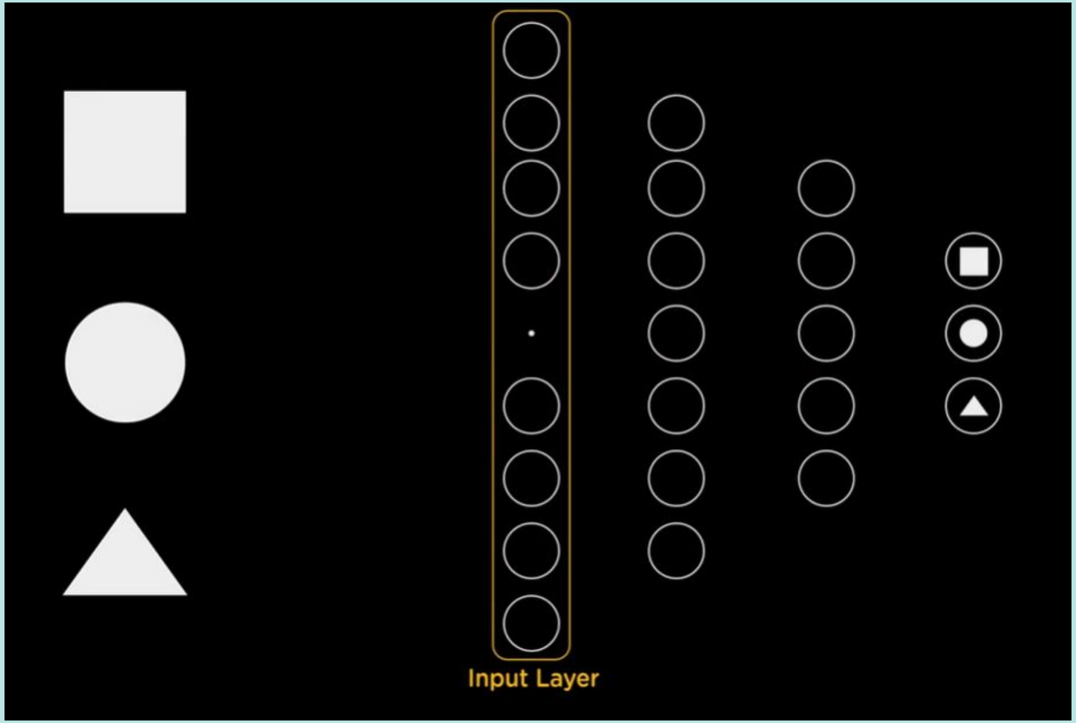




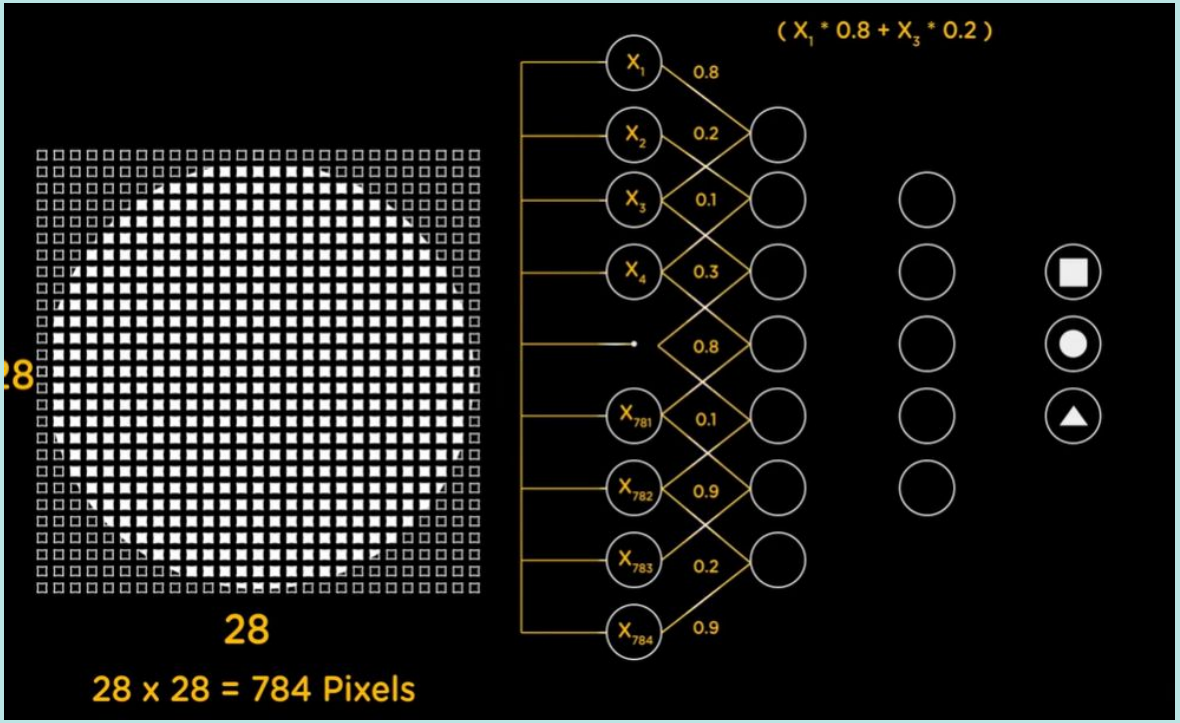
Neural Network vs Machine Learning

- Machine learning is a broader concept that refers to the ability of computer systems to learn from experience, without being explicitly programmed, while neural networks are a specific type of machine learning algorithm.
- Machine learning includes different categories of algorithms such as supervised learning, unsupervised learning, and reinforcement learning, while neural networks belong to the category of supervised learning.
- Machine learning algorithms can be linear or nonlinear, while neural networks are non-linear and are designed to simulate the behavior of the human brain.
- Machine learning algorithms can work with various types of data, including text, images, and numerical data, while neural networks are commonly used for processing complex data such as images, speech, and natural language.

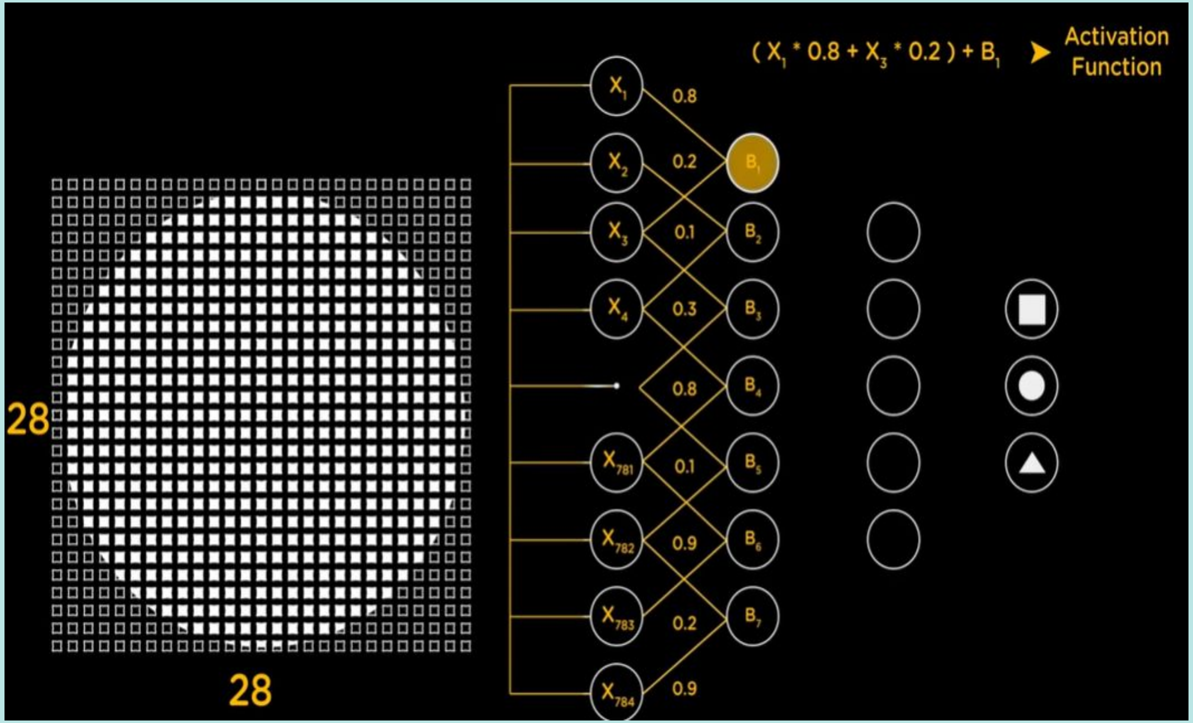
Simple example



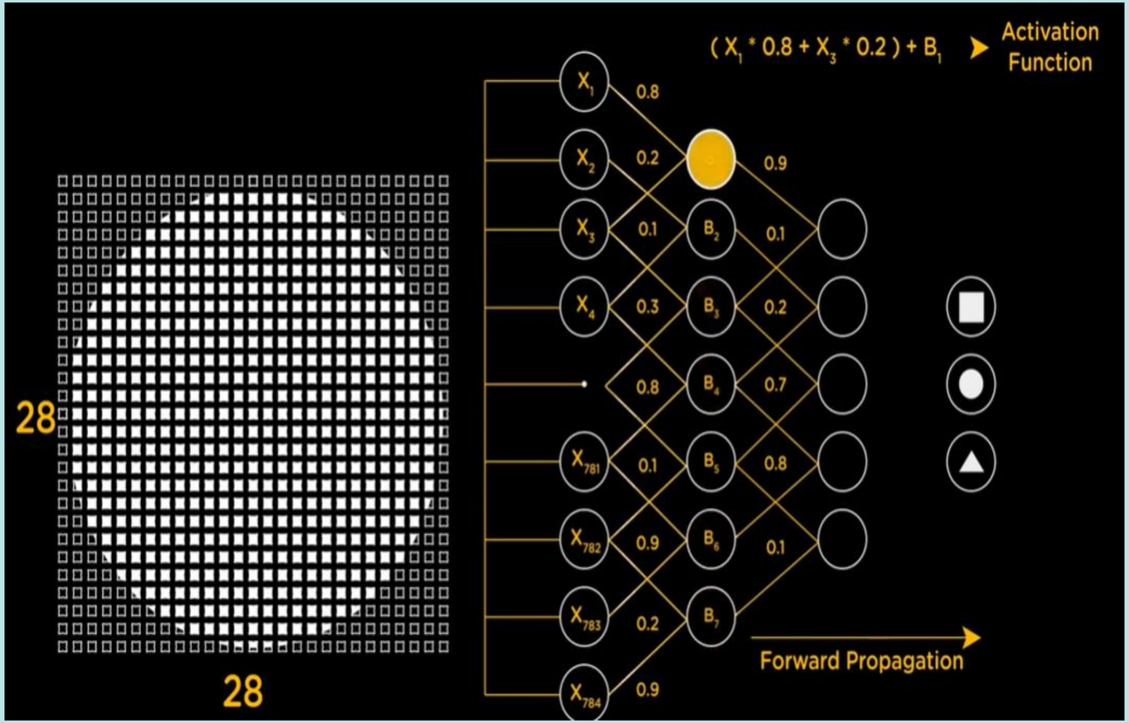
Representation of Image



Input Layer

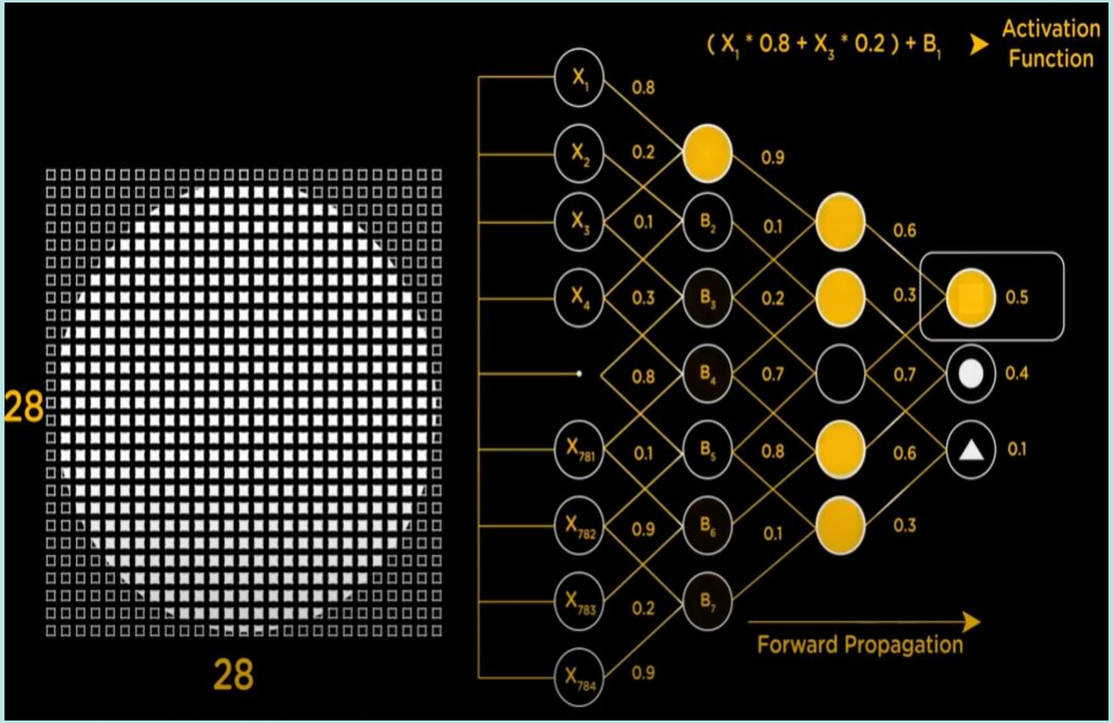

$(X_1 * 0.8 + X_3 * 0.2) + B_1$
Activation Function
28
28
X_1
X_2
X_3
X_4
X_{781}
X_{782}
X_{783}
X_{784}
0.8
0.2
0.1
0.3
0.8
0.1
0.9
0.2
0.9
B_1
B_2
B_3
B_4
B_5
B_6
B_7

Activation Function

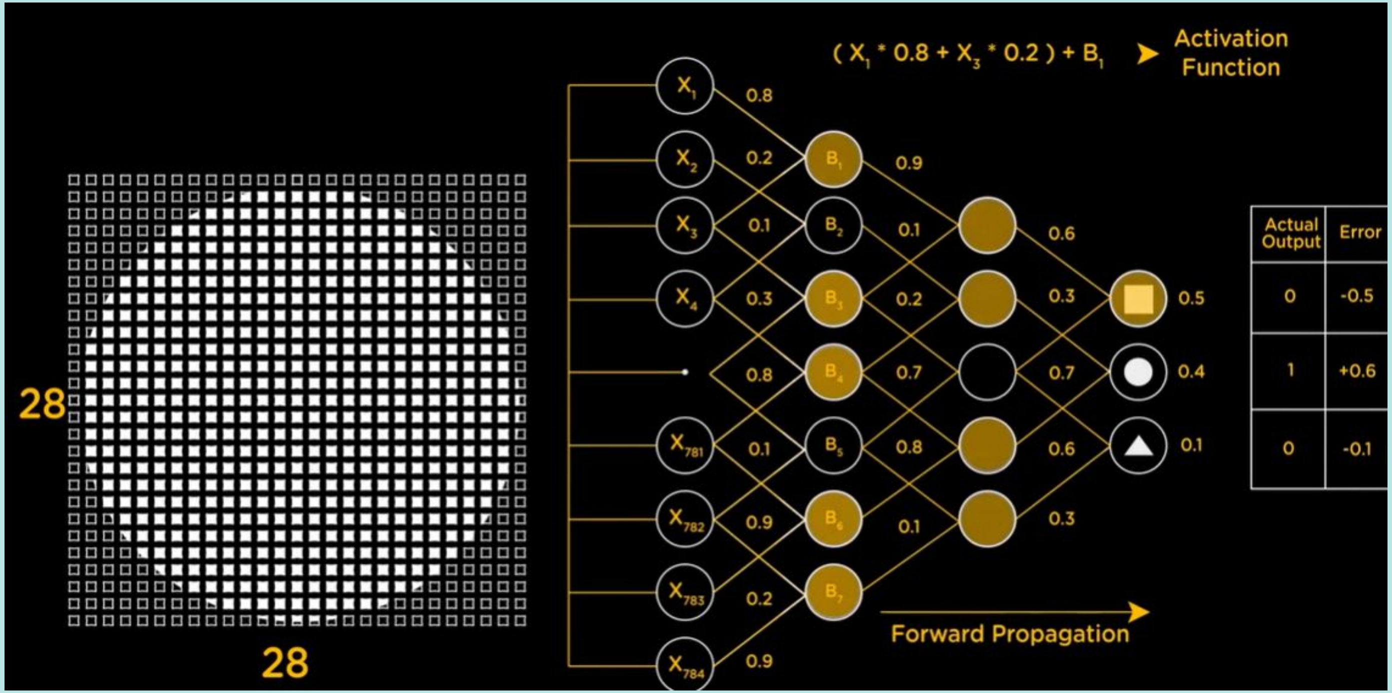


RESEARCH GROUP

Forward Propagation

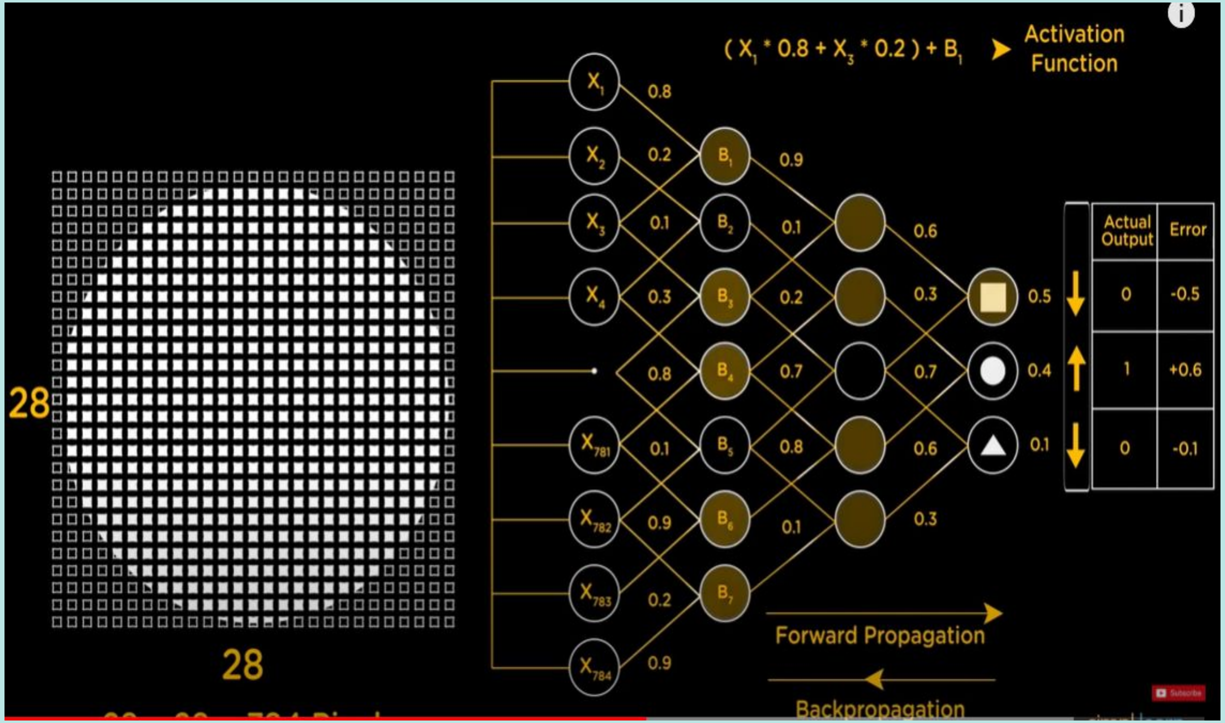


Calculating the error



Actual Output	Error
0	-0.5
1	+0.6
0	-0.1

Back Propagation



Actual Output	Error
0	-0.5
1	+0.6
0	-0.1

RESEARCH GROUP

Projects

- ➢ Iris Flower Classification
- ➢ Cancer detection
- ➢ Wine Classification

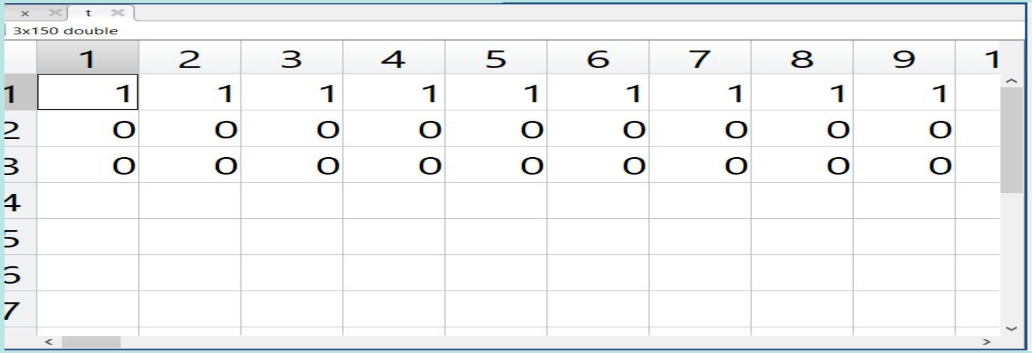
Iris Flower Classification

Four Features has been extracted

- *In this example we attempt to build a neural network that clusters iris flowers into natural classes, such that similar classes are grouped together. Each iris is described by four features:*
  - ***Sepal length in cm***
  - ***Sepal width in cm***
  - ***Petal length in cm***
  - ***Petal width in cm***

RESEARCH GROUP

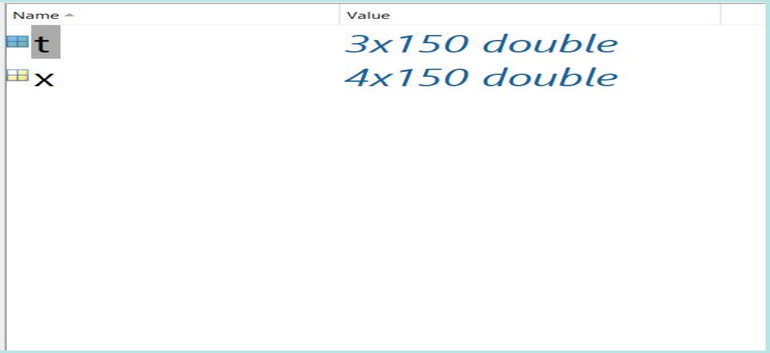
Load the iris dataset



t

3x150 double

	1	2	3	4	5	6	7	8	9	1
1	1	1	1	1	1	1	1	1	1	
2	0	0	0	0	0	0	0	0	0	
3	0	0	0	0	0	0	0	0	0	
4										
5										
6										
7										



Name	Value
t	3x150 double
x	4x150 double

- X is the feature
- T is the target class
- 150 data points

RESEARCH GROUP

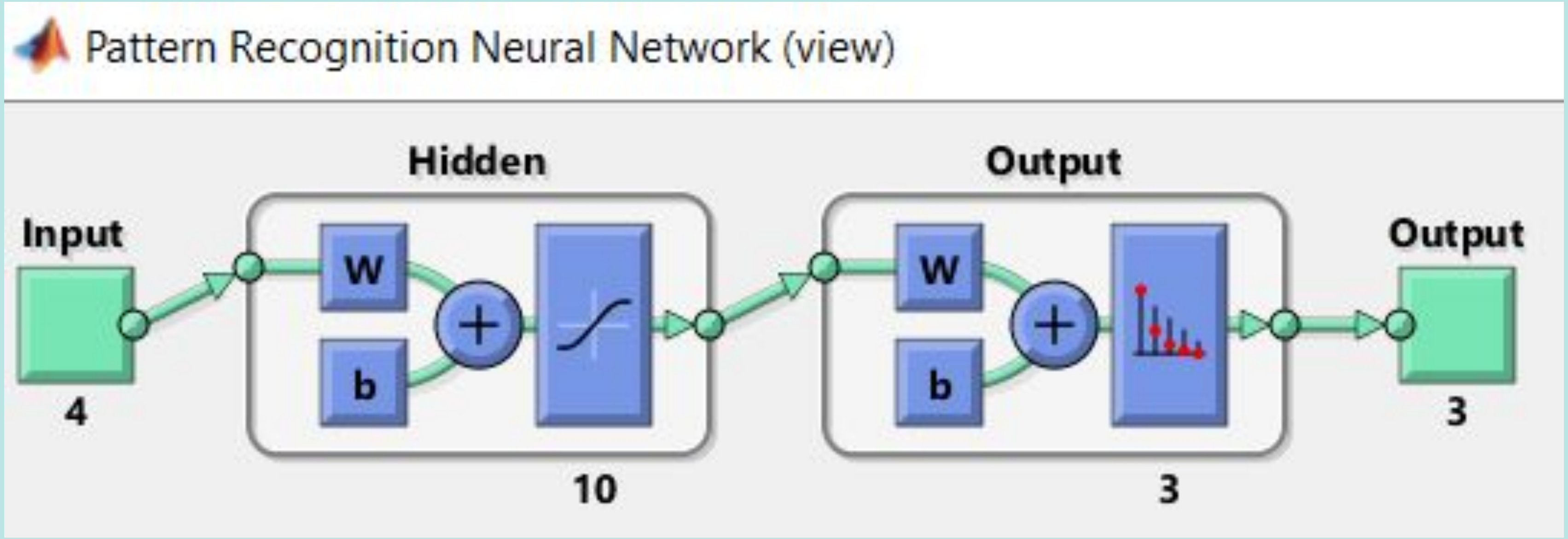
Matlab code for Pattern Recognition Neural Network

```
clc;clear;close all;
[x,t] = iris_dataset;
net = patternnet(10);
net = train(net,x,t);
view(net)
testdata=x(:,2)
save net net
perf = perform(net,t,y);
```

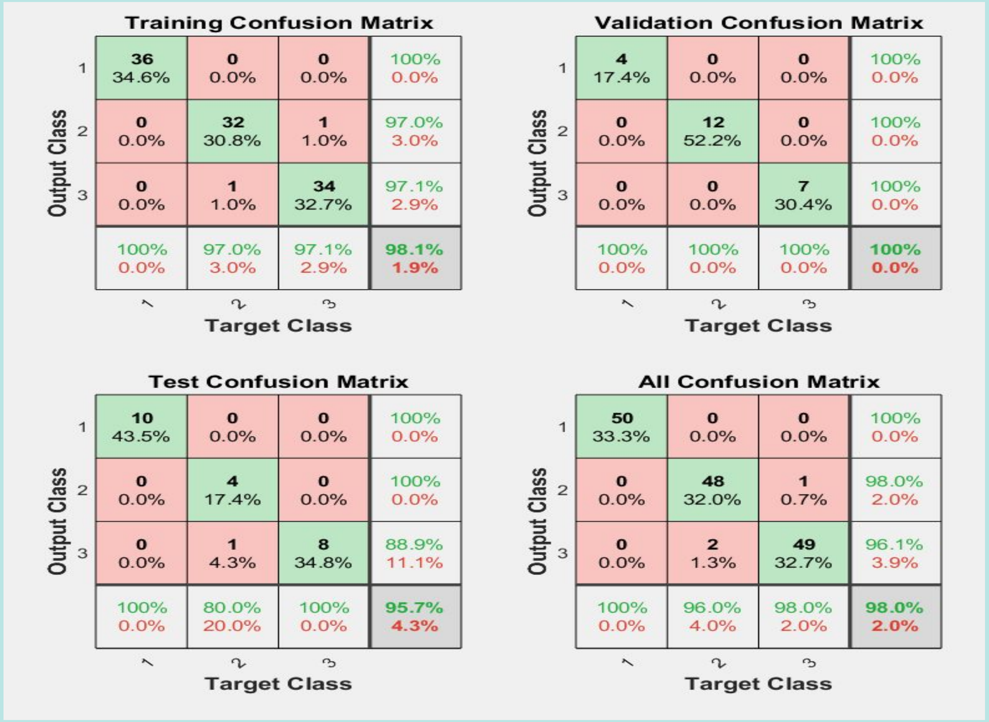
Testing

```
clc;clear;close all;
[x,t] = iris_dataset;
load net
inputvalue=input('enter the column-->');
testdata=x(:,inputvalue);
y = net(testdata);
class = vec2ind(y);
disp(class)
```

Neural Network Architecture



Confusion Matrix



Training Confusion Matrix

Output Class \ Target Class	1	2	3	
1	**36** 34.6%	**0** 0.0%	**0** 0.0%	100% 0.0%
2	**0** 0.0%	**32** 30.8%	**1** 1.0%	97.0% 3.0%
3	**0** 0.0%	**1** 1.0%	**34** 32.7%	97.1% 2.9%
	100% 0.0%	97.0% 3.0%	97.1% 2.9%	**98.1%** **1.9%**

Validation Confusion Matrix

Output Class \ Target Class	1	2	3	
1	**4** 17.4%	**0** 0.0%	**0** 0.0%	100% 0.0%
2	**0** 0.0%	**12** 52.2%	**0** 0.0%	100% 0.0%
3	**0** 0.0%	**0** 0.0%	**7** 30.4%	100% 0.0%
	100% 0.0%	100% 0.0%	100% 0.0%	**100%** **0.0%**

Test Confusion Matrix

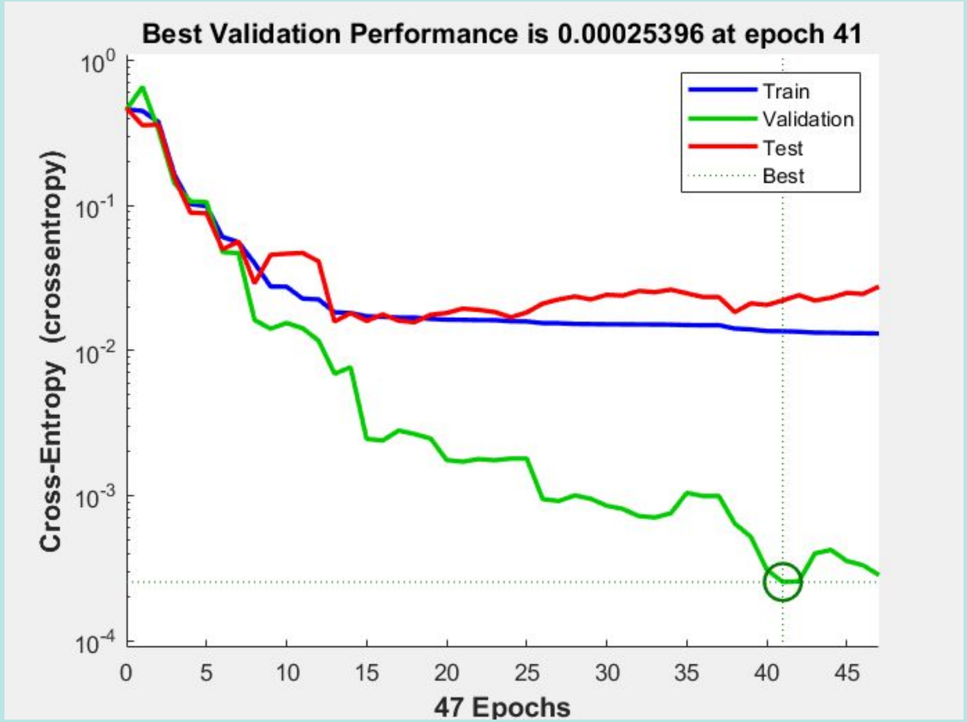
Output Class \ Target Class	1	2	3	
1	**10** 43.5%	**0** 0.0%	**0** 0.0%	100% 0.0%
2	**0** 0.0%	**4** 17.4%	**0** 0.0%	100% 0.0%
3	**0** 0.0%	**1** 4.3%	**8** 34.8%	88.9% 11.1%
	100% 0.0%	80.0% 20.0%	100% 0.0%	**95.7%** **4.3%**

All Confusion Matrix

Output Class \ Target Class	1	2	3	
1	**50** 33.3%	**0** 0.0%	**0** 0.0%	100% 0.0%
2	**0** 0.0%	**48** 32.0%	**1** 0.7%	98.0% 2.0%
3	**0** 0.0%	**2** 1.3%	**49** 32.7%	96.1% 3.9%
	100% 0.0%	96.0% 4.0%	98.0% 2.0%	**98.0%** **2.0%**

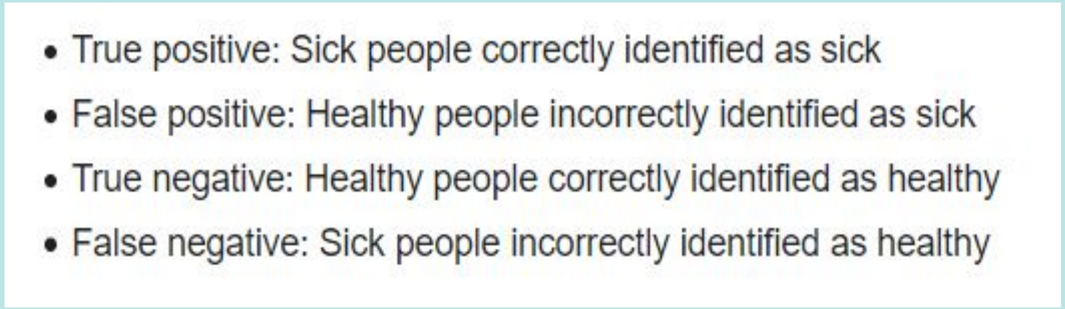
RESEARCH GROUP

Error

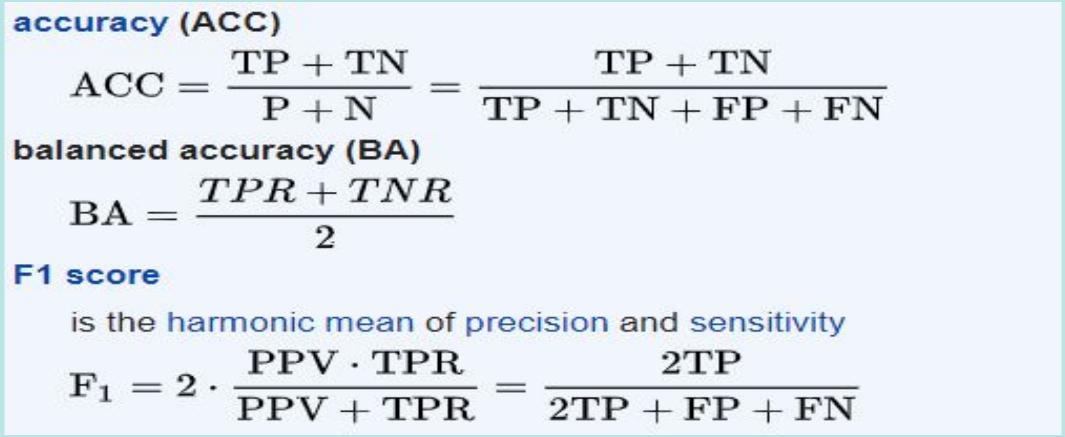


RESEARCH GROUP

Evaluation Metrics



- True positive: Sick people correctly identified as sick
- False positive: Healthy people incorrectly identified as sick
- True negative: Healthy people correctly identified as healthy
- False negative: Sick people incorrectly identified as healthy



accuracy (ACC)

$$\mathrm{ACC} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{P} + \mathrm{N}} = \frac{\mathrm{TP} + \mathrm{TN}}{\mathrm{TP} + \mathrm{TN} + \mathrm{FP} + \mathrm{FN}}$$

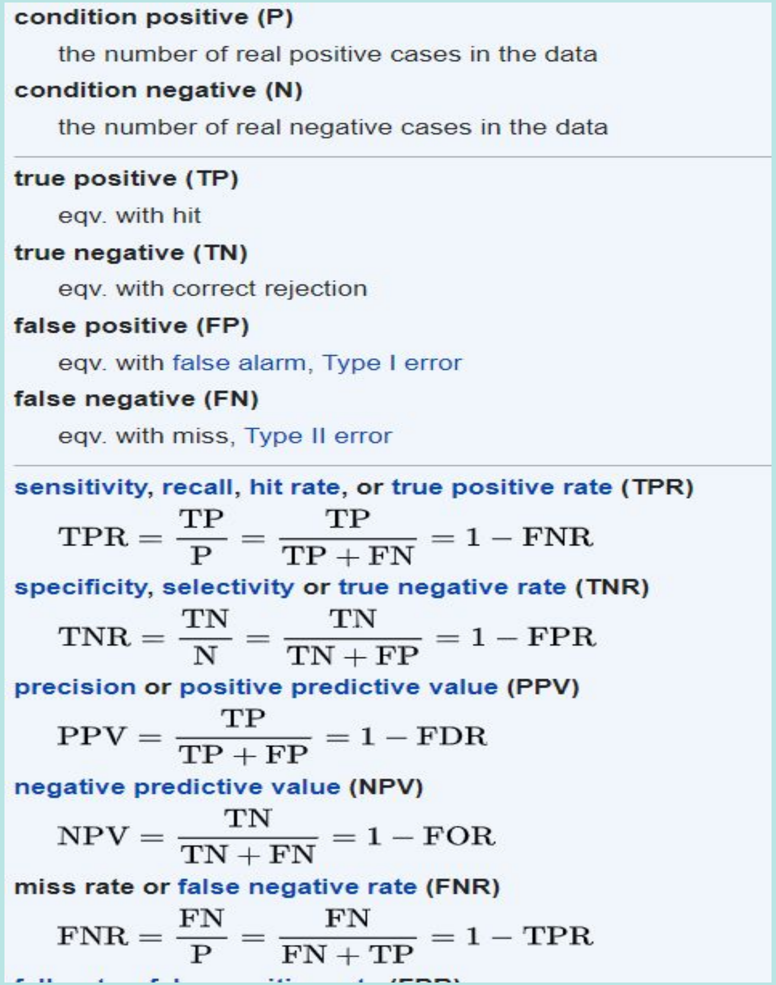
balanced accuracy (BA)

$$\mathrm{BA} = \frac{TPR + TNR}{2}$$

F1 score

is the harmonic mean of precision and sensitivity

$$\mathrm{F}_1 = 2 \cdot \frac{\mathrm{PPV} \cdot \mathrm{TPR}}{\mathrm{PPV} + \mathrm{TPR}} = \frac{2\mathrm{TP}}{2\mathrm{TP} + \mathrm{FP} + \mathrm{FN}}$$



condition positive (P)
the number of real positive cases in the data

condition negative (N)
the number of real negative cases in the data

true positive (TP)
eqv. with hit

true negative (TN)
eqv. with correct rejection

false positive (FP)
eqv. with false alarm, Type I error

false negative (FN)
eqv. with miss, Type II error

sensitivity, recall, hit rate, or true positive rate (TPR)

$$\mathrm{TPR} = \frac{\mathrm{TP}}{\mathrm{P}} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FN}} = 1 - \mathrm{FNR}$$

specificity, selectivity or true negative rate (TNR)

$$\mathrm{TNR} = \frac{\mathrm{TN}}{\mathrm{N}} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FP}} = 1 - \mathrm{FPR}$$

precision or positive predictive value (PPV)

$$\mathrm{PPV} = \frac{\mathrm{TP}}{\mathrm{TP} + \mathrm{FP}} = 1 - \mathrm{FDR}$$

negative predictive value (NPV)

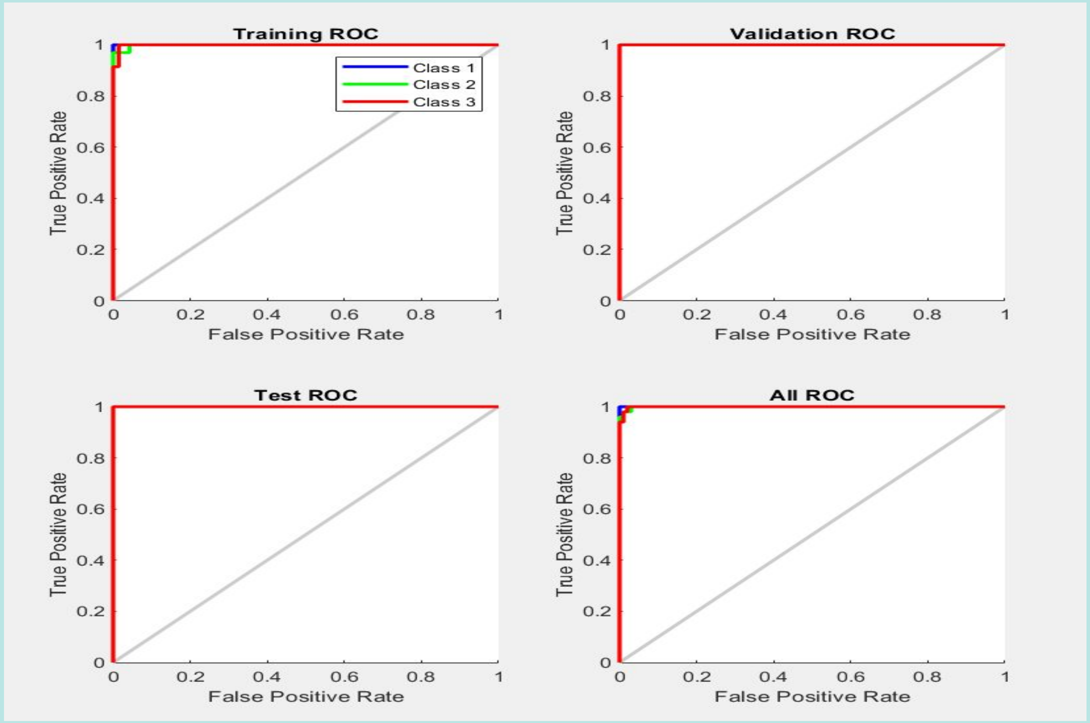
$$\mathrm{NPV} = \frac{\mathrm{TN}}{\mathrm{TN} + \mathrm{FN}} = 1 - \mathrm{FOR}$$

miss rate or false negative rate (FNR)

$$\mathrm{FNR} = \frac{\mathrm{FN}}{\mathrm{P}} = \frac{\mathrm{FN}}{\mathrm{FN} + \mathrm{TP}} = 1 - \mathrm{TPR}$$

[illegible]

ROC-Receiver Operating Characteristics



RESEARCH GROUP

Diabetes Prediction NN

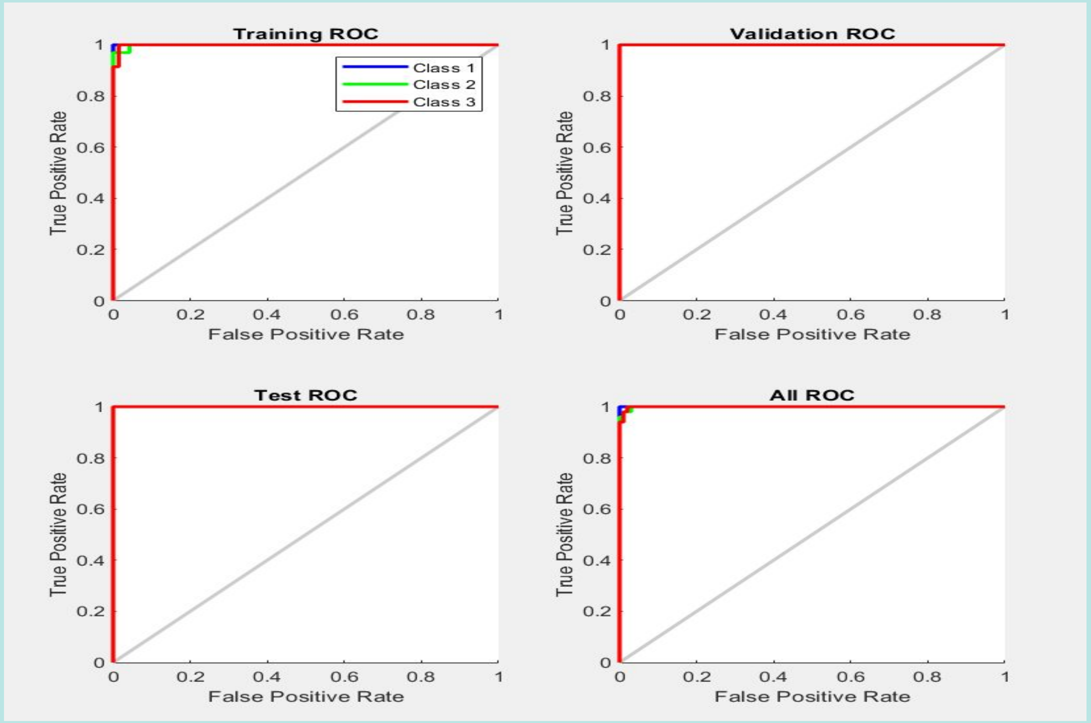
```
x = input';
t = taget';
% Choose a Training Function
% For a list of all training functions type: help nntrain % 'trainlm' is usually fastest.% 'trainbr' takes longer but
may be better for challenging problems.% 'trainscg' uses less memory. Suitable in low memory situations.
trainFcn = 'trainlm';  % Levenberg-Marquardt backpropagation.
% Create a Fitting Network
hiddenLayerSize = 15;
net = fitnet(hiddenLayerSize,trainFcn);
% Setup Division of Data for Training, Validation, Testing
net.divideParam.trainRatio = 70/100;
net.divideParam.valRatio = 15/100;
net.divideParam.testRatio = 15/100;
% Train the Network
[net,tr] = train(net,x,t);
% Test the Network
y = net(x);
e = gsubtract(t,y);
performance = perform(net,t,y)% View the Network
view(net)
```

RESEARCH GROUP

```
clc; close all; clear;
data = csvread('pima.csv')
% % Split the data into inputs and outputs
 inputs = data(:, 1:8)';
outputs = data(:, 9)';
% % Create a neural network with 10 hidden layers
hiddenLayerSize = 10;
 net = patternnet(hiddenLayerSize);
% % Set the training parameters
 net.divideParam.trainRatio = 0.7;
 net.divideParam.valRatio = 0.15;
 net.divideParam.testRatio = 0.15;
 net.trainParam.max_fail = 20;
 net.trainParam.epochs = 100;
% % Train the neural network
 [net,tr] = train(net,inputs,outputs);
% % Make predictions on the test data
 testInputs = inputs(:,tr.testInd);
 testTargets = outputs(:,tr.testInd);
 testOutputs = net(testInputs);
% Plot the confusion matrix
 plotconfusion(testTargets,testOutputs);
% Save the model
 save net net
```

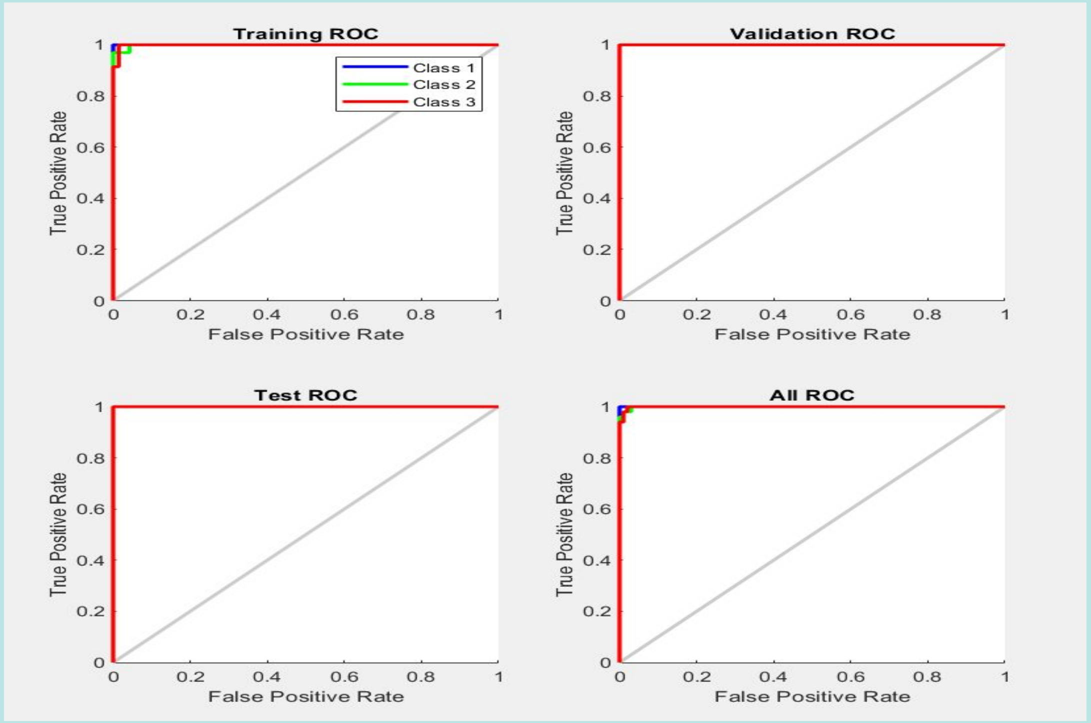
```
clc; close all; clear;
% Load the saved neural network model
load net net;

% Get the input features from the user
input_features = zeros(1, 8);
for i = 1:8
    input_features(i) = input(sprintf('Enter value for feature %d: ', i));
end

% % Normalize the input features
% input_features = (input_features - mean_input_features) ./ std_input_features;

% Use the trained neural network to classify the input features
output = net(input_features');

% Display the predicted class
if output >= 0.5
    fprintf('the person have diabetes');
else
    fprintf('The person didnt not have diabetes');
end
```

ROC-Receiver Operating Characteristics



RESEARCH GROUP

ROC-Receiver Operating Characteristics



RESEARCH GROUP

RESEARCH GROUP