Deep Learning

Lecture 3
Training Neural Networks

Asst. Lect. Ali Al-khawaja

Class Room

Training Neural Networks

- Training a neural network involves the process of adjusting the weights and biases of the network to minimize the difference between the predicted outputs and the actual outputs for a given set of training data.

- This process is typically done through a combination of forward propagation and backpropagation.

- This process is typically done through a combination of forward propagation and backpropagation. Here's a step-by-step overview of the training process:

Training Neural Networks

- Training a neural network involves the process of adjusting the weights and biases of the network to minimize the difference between the predicted outputs and the actual outputs for a given set of training data.

- This process is typically done through a combination of forward propagation and backpropagation.

- This process is typically done through a combination of forward propagation and backpropagation. Here's a step-by-step overview of the training process:

Training Neural Networks

On each layer we learn different representation that gets more complicated with later hidden layers.Below is an example of a 3-layers neural network (we don't count input layer):

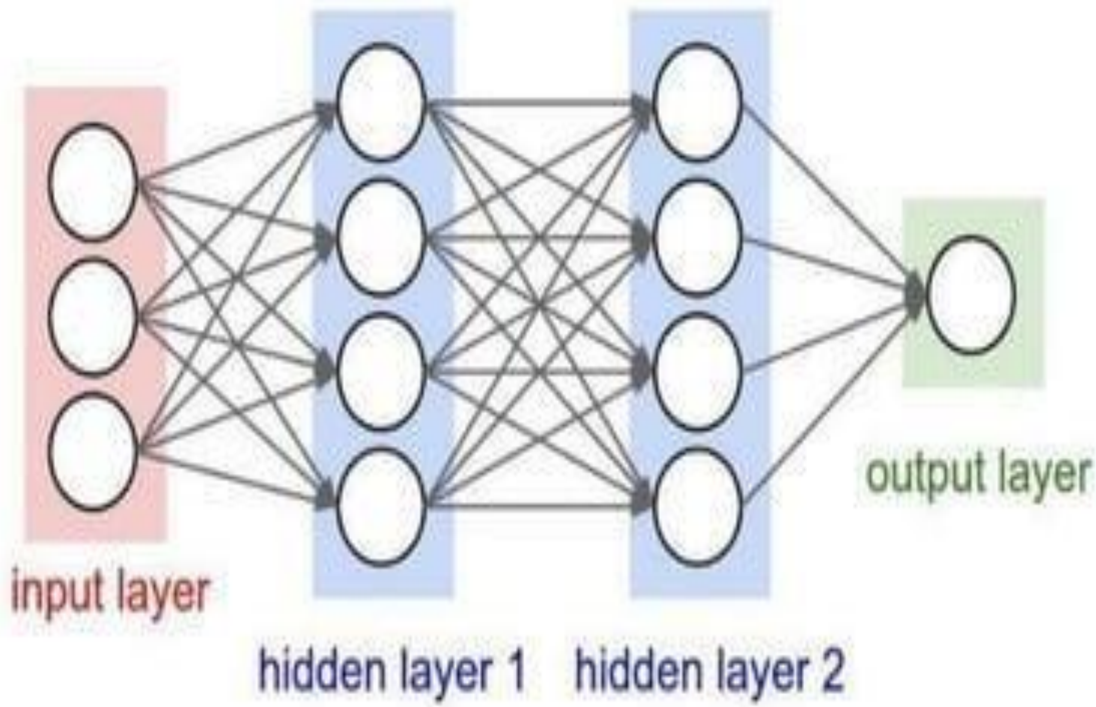
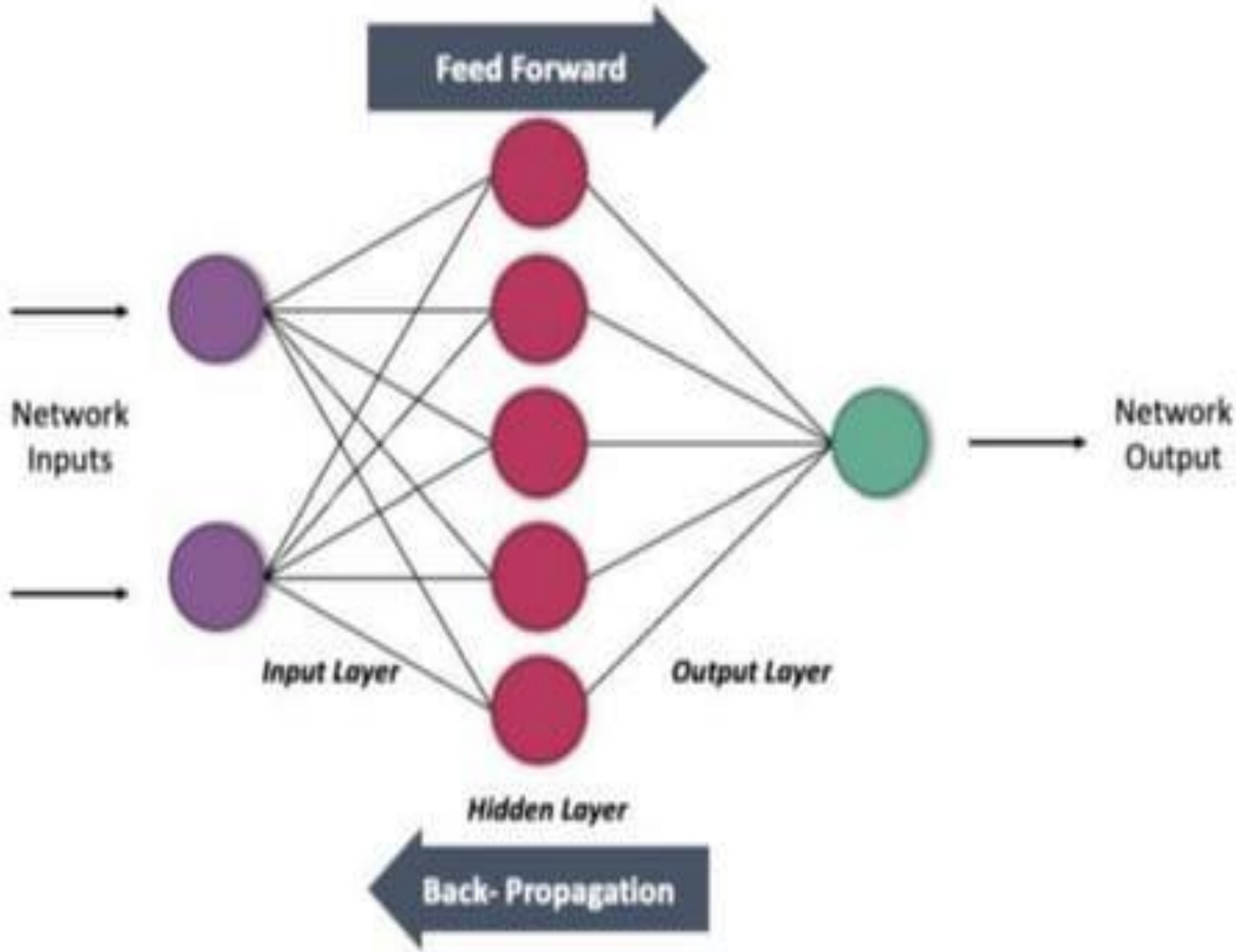
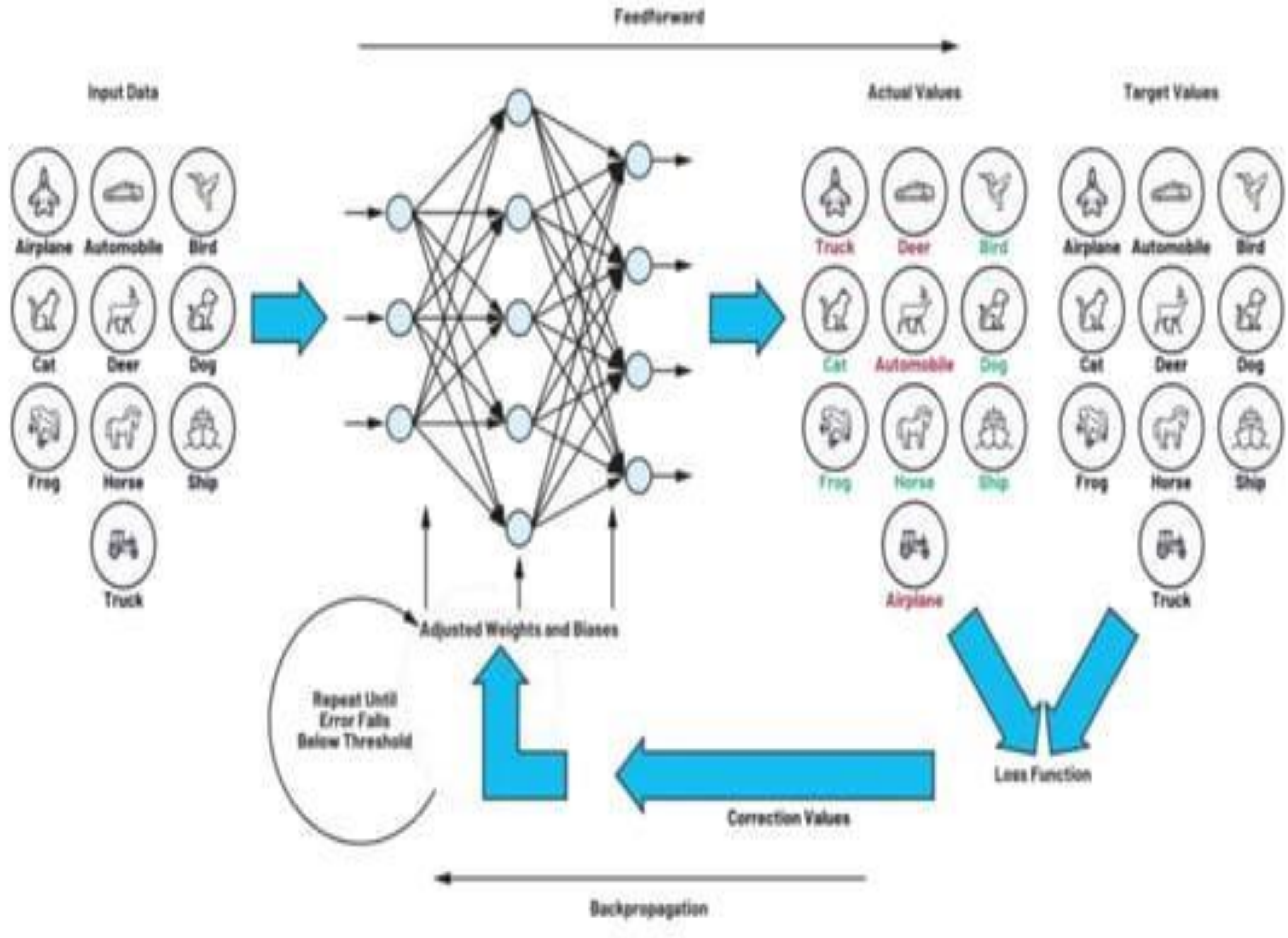


Figure 1: Neural Network with two hidden layers

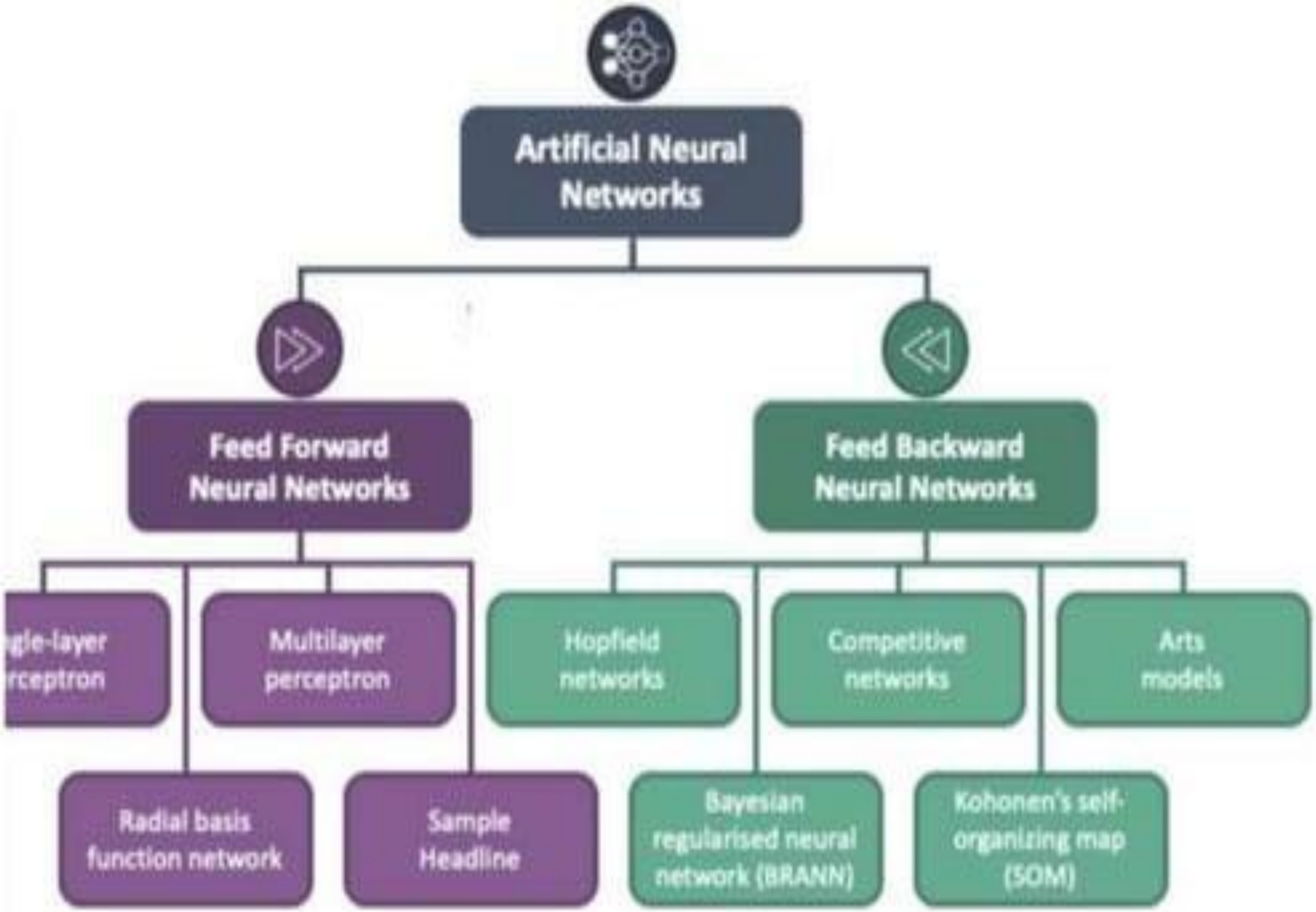
Training Neural Networks



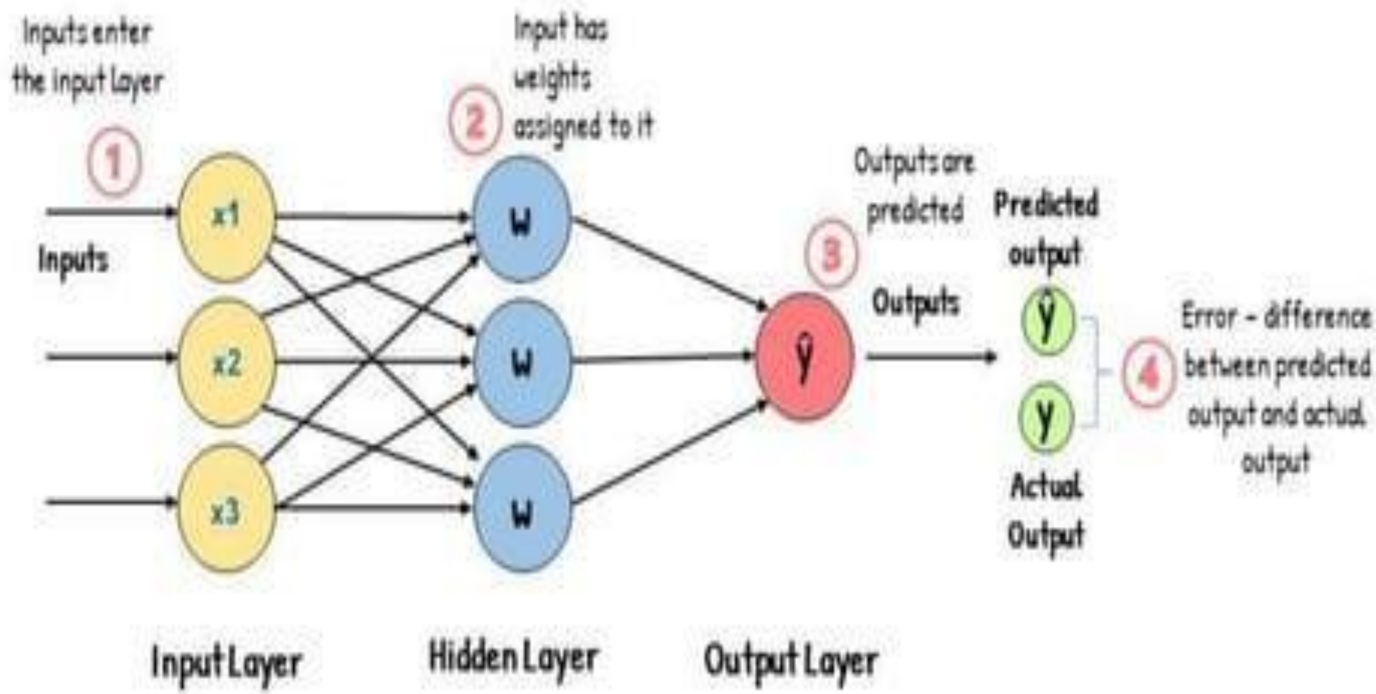
Training Neural Networks



Training Neural Networks



Feed- Forward Neural Network



Feed- Forward Neural Network

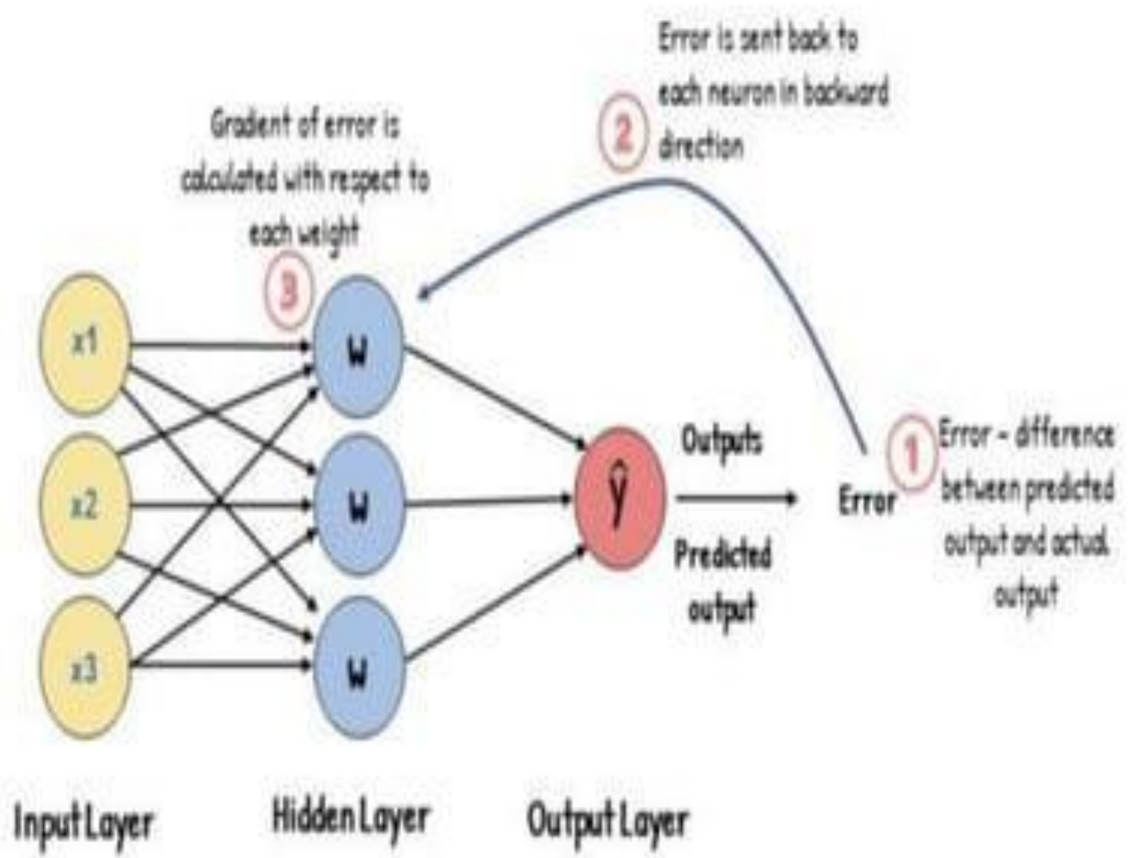
Forward Propagation:

- Input a training sample into the neural network.
- Propagate the input forward through the network to compute the predicted output.
- Apply activation functions at each neuron to introduce non-linearity.

Compute Loss:

- Compare the predicted output with the actual output using a loss function.
- The loss function quantifies the error between the predicted and actual outputs.

Backpropagation



Backpropagation

Backward Pass:

Compute the gradient of the loss with respect to the output of the neural network.

$\partial Loss / \partial Output$

Backpropagate Errors:

- Propagate the gradient backward through the network to compute the gradients of the loss with respect to the weights and biases.
- Use the chain rule to calculate the contribution of each weight and bias to the overall loss.

Backpropagation

Update Weights and Biases:

- Adjust the weights and biases to minimize the loss by descending along the negative gradient direction.
- Update each weight and bias using the gradient and a chosen optimization algorithm (e.g., stochastic gradient descent)

New Weight=Old Weight−Learning Rate× ∂Weight / ∂Loss

Repeat for Each Training Sample or Batch:

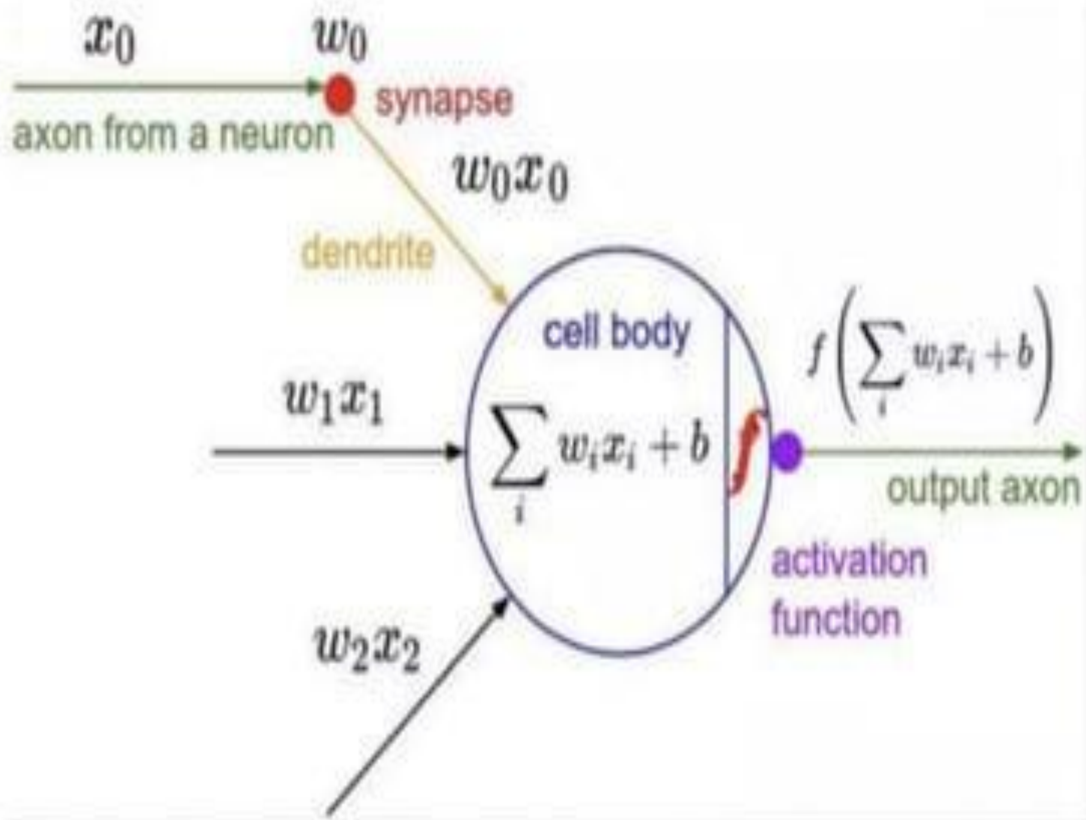
- Iterate through all training samples or batches, performing steps 1-5 for each.
- The algorithm can use full-batch, mini-batch, or stochastic gradient descent.

Repeat for Multiple Epochs:

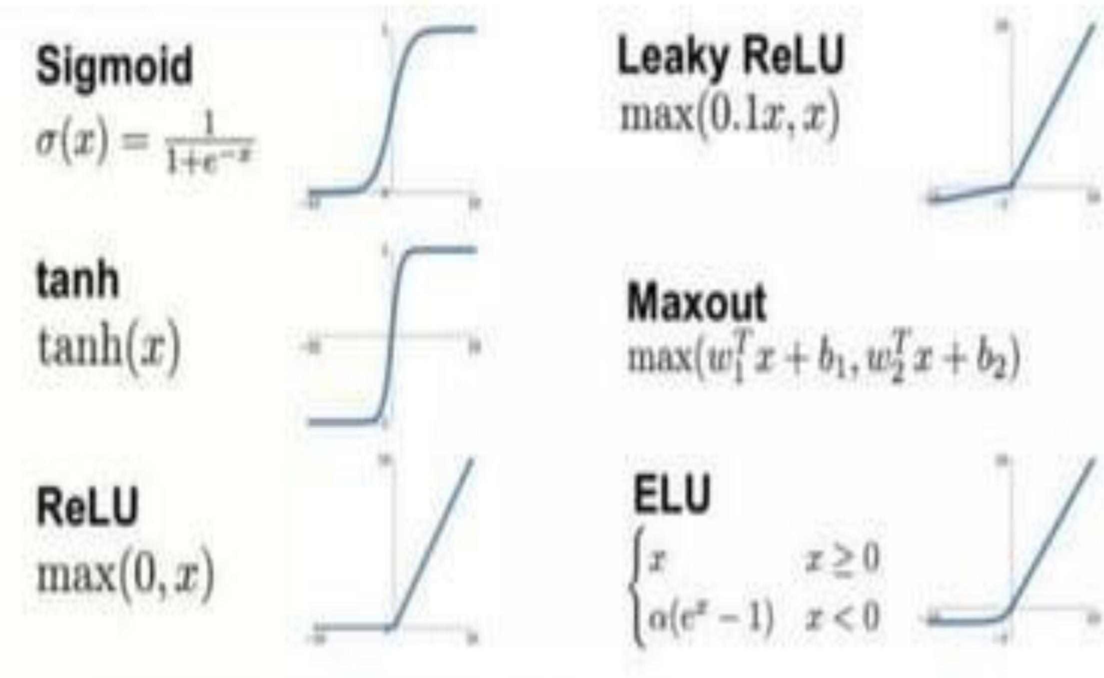
- Continue the forward and backward passes for a specified number of epochs or until the loss converges.

Activation Function

- Importance of Activation function is to introduce **non-linearity** into the network



Activation Function



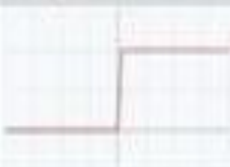
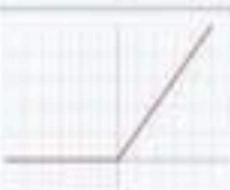
For Output layer

- Sigmoid
- Tanh
- Softmax

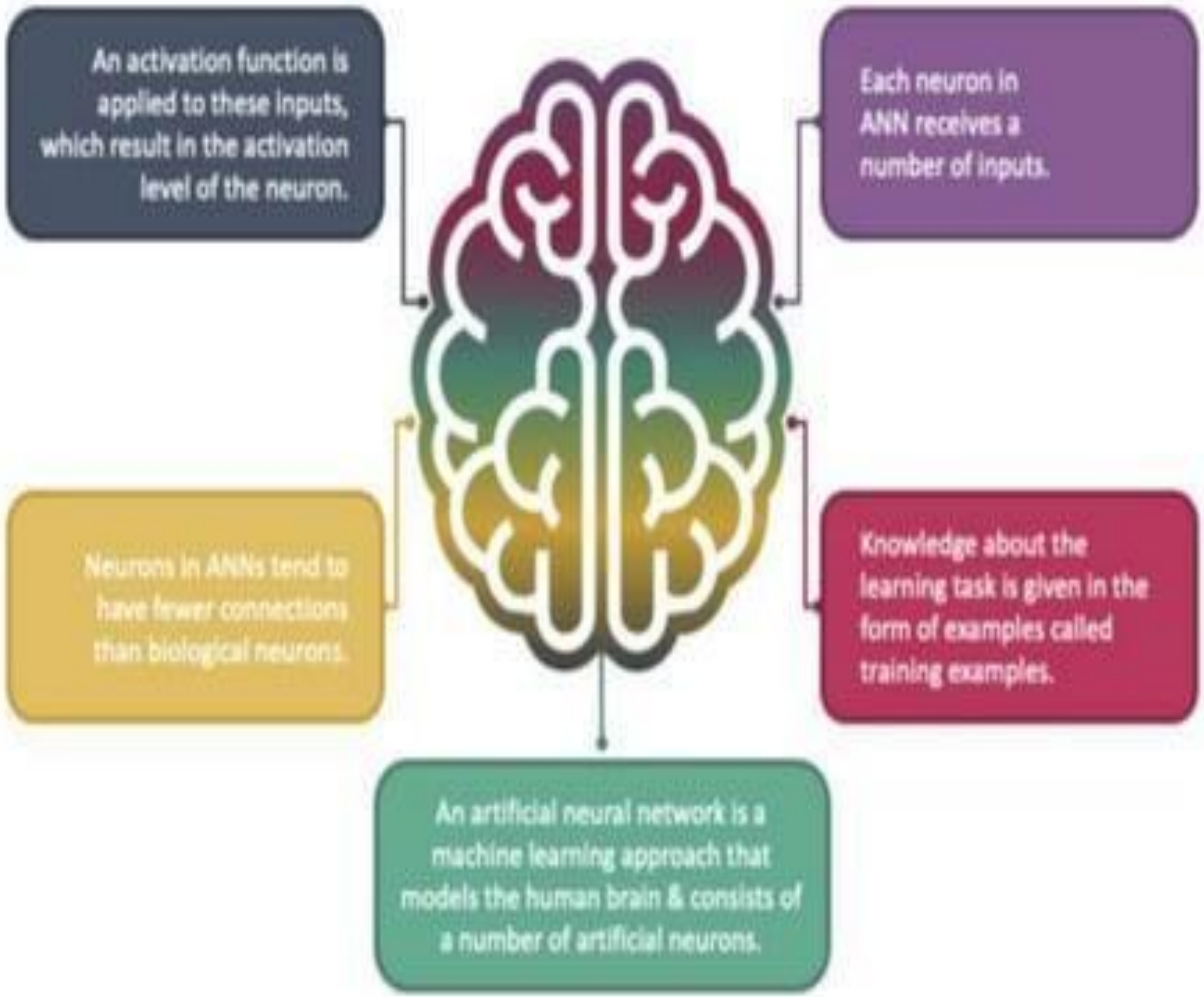
For Hidden Layer

- ReLU
- Leaky ReLU
- ELU

Activation Function

Name	Plot	Function, $f(x)$	Derivative of f, $f'(x)$	Range
Identity		x	1	$(-\infty, \infty)$
Binary step		$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x \geq 0 \end{cases}$	$\begin{cases} 0 & \text{if } x \neq 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$\{0, 1\}$
Logistic, sigmoid, or soft step		$\sigma(x) = \frac{1}{1 + e^{-x}}$ [1]	$f(x)(1 - f(x))$	$(0, 1)$
tanh		$\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$	$1 - f(x)^2$	$(-1, 1)$
Rectified linear unit (ReLU)[11]		$\begin{cases} 0 & \text{if } x \leq 0 \\ x & \text{if } x > 0 \end{cases}$ $= \max\{0, x\} = x\mathbf{1}_{x>0}$	$\begin{cases} 0 & \text{if } x < 0 \\ 1 & \text{if } x > 0 \\ \text{undefined} & \text{if } x = 0 \end{cases}$	$[0, \infty)$

Activation Function



Wrap-up

In this lecture we have learned

- The neural network made of interconnected neurons. Each neuron is characterized by its weight, bias and activation function.
- The input is fed to the input layer, the neurons perform a linear transformation on this input using the weights and biases.

$$x = (\text{weight} * \text{input}) + \text{bias}$$

Wrap-up

In this lecture we have learned

- **Forward propagation:** Passes input through the network to produce predictions.
- **Backpropagation:** Adjusts weights and biases based on the gradient of the loss to improve the model's performance.
- **Activation functions:** Introduce non-linearity to the model.
- **Loss function:** Measures the difference between predicted and actual outputs.

Thank you…

Any questions??

My google site

يرجى مسح رمز الاستجابة السريعة QR Code لتعبئة نموذج التغذية الراجعة حول المحاضرة. ملاحظاتكم مهمة لتحسين المحاضرات القادمة.