McMaster University ENGINEERING

W BOOTH SCHOOL OF ENGINEERING PRACTICE AND TECHNOLOGY

SEP 785: Machine Learning

Lecture 11: Introduction to Neural Networks

Instructor: Dr. Dalia Mahmoud, PhD
(Mechanical Engineering, McMaster University)
Email: mahmoudd@mcmaster.ca

Intended Learning Outcome

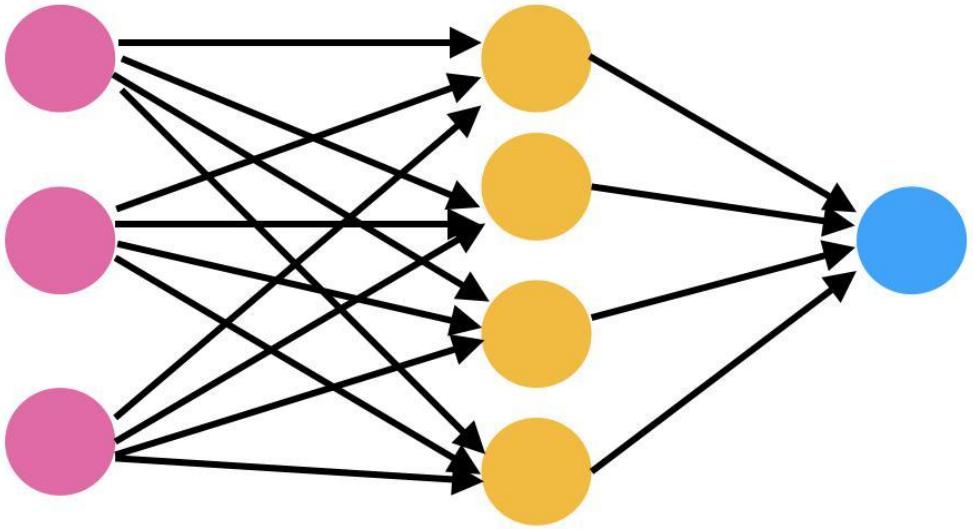
- Describe the fundamental structure and components of a feedforward neural network.
- Explain the role of neurons, layers, weights, biases, and activation functions in neural computation.
- Illustrate the forward propagation process through a neural network with appropriate equations and dimensions.
- Identify the purpose of loss functions and their impact on model learning.
- Summarize the high-level concept of backpropagation and how neural networks learn.

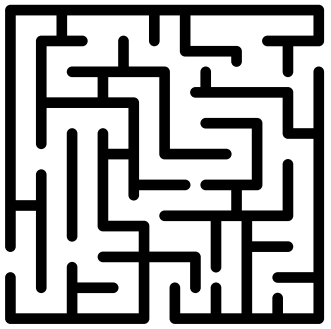
Contents

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Why Neural Networks

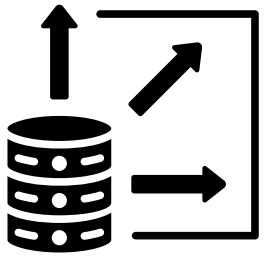
- Neural networks are a flexible and powerful tool for learning from data, especially when patterns are non-obvious or hard to define by hand.



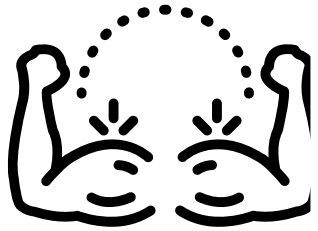


Complexity

Versatility



Scalability

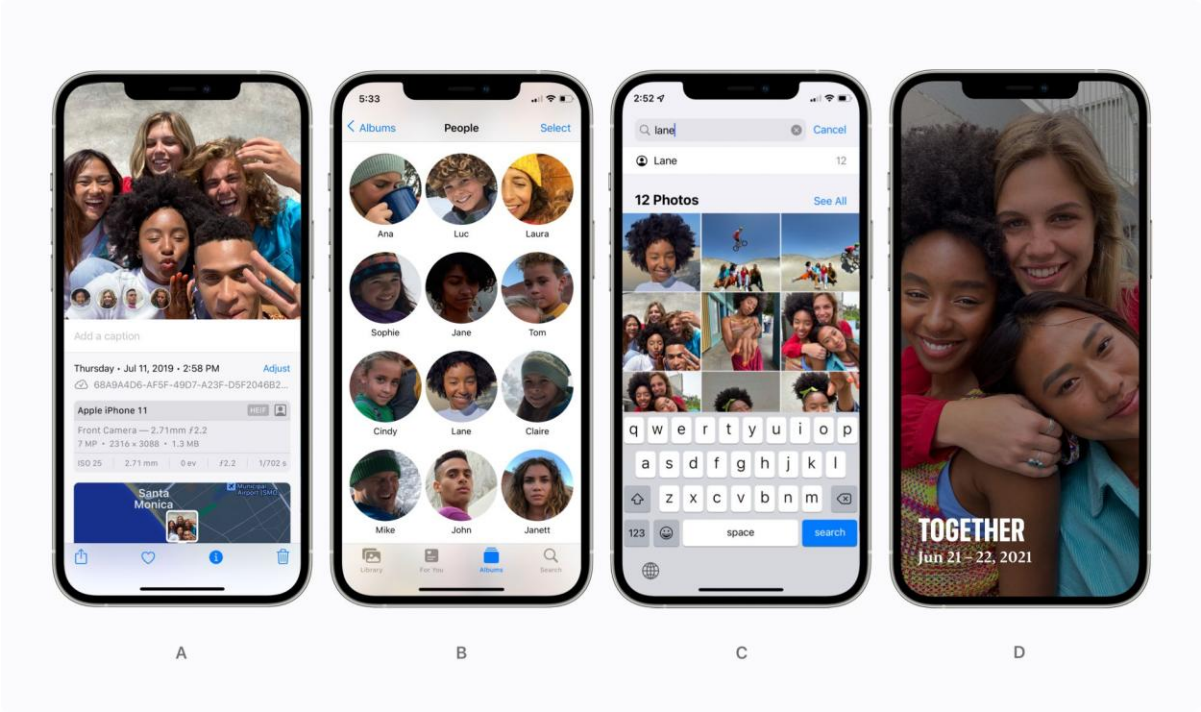


Strength

Applications of Neural Networks

Image Classification (Computer Vision)

- **Use case:** Classifying handwritten digits (MNIST), dogs vs. cats, facial recognition
- **Networks used:** Convolutional Neural Networks (CNNs)
- **Example:** auto-tagging faces or objects



Applications of Neural Networks

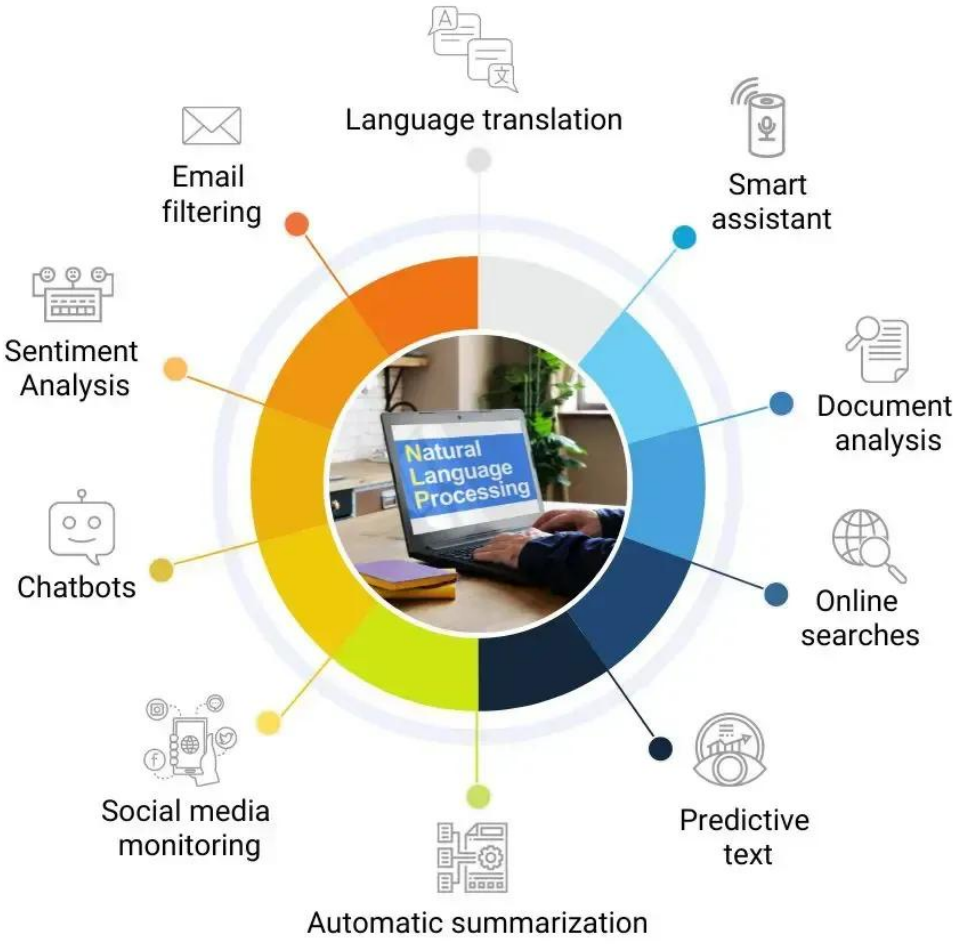
Speech Recognition (Audio)

- **Use case:** Converting spoken language to text
- **Networks used:** Recurrent Neural Networks (RNNs), LSTMs, Transformers
- **Example:** Voice assistants like Siri or Alexa

Applications of Neural Networks

Language Understanding (NLP)

- **Use case:** Text classification, translation, summarization, chatbots
- **Networks used:** Transformers (e.g., BERT, GPT)
- **Example:** ChatGPT or Google Translate



Applications of Neural Networks

Autonomous Vehicles

- **Use case:** Detecting pedestrians, reading traffic signs, path planning
- **Networks used:** Combination of CNNs, reinforcement learning, and sensor fusion
- **Example:** Tesla Autopilot, Waymo

Applications of Neural Networks

Medical Diagnosis

- **Use case:** Detecting tumors in MRI scans, predicting patient outcomes
- **Networks used:** CNNs for imaging, deep MLPs or LSTMs for time-series data
- **Example:** AI models for early cancer detection

Applications of Neural Networks

Finance

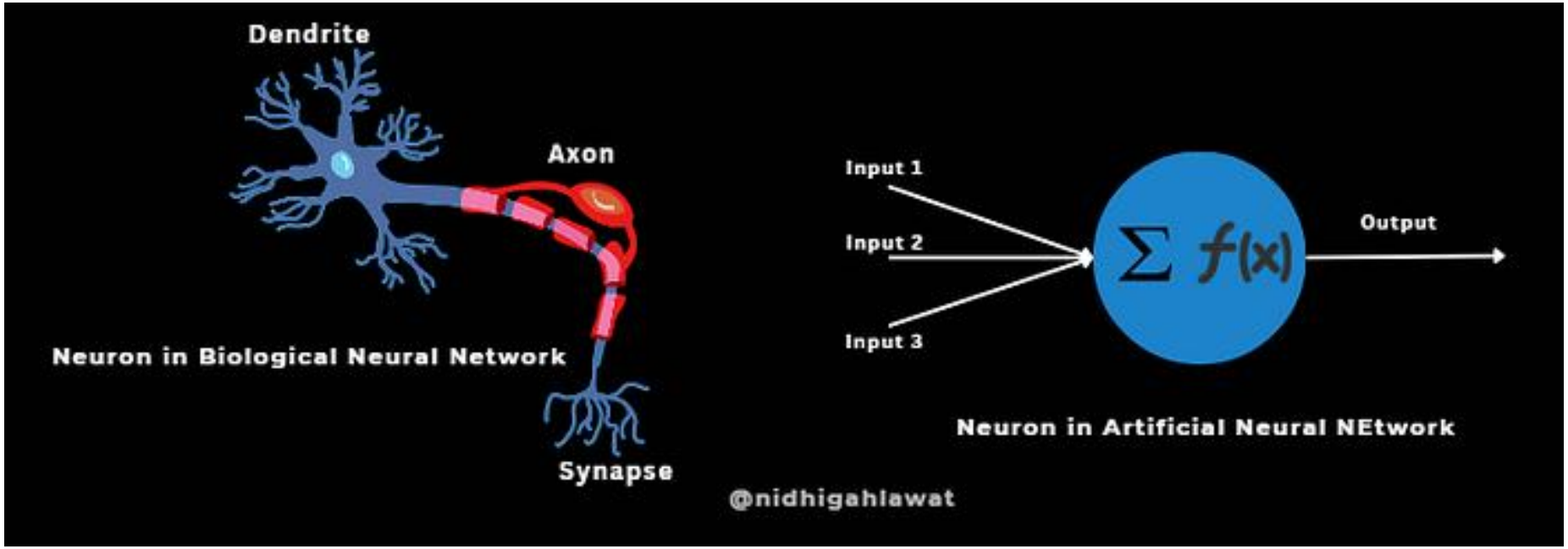
- **Use case:** Fraud detection, algorithmic trading, credit scoring
- **Networks used:** Feedforward NNs, LSTMs (for sequential data)
- **Example:** Real-time fraud detection on credit card transactions

Contents

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Neurons

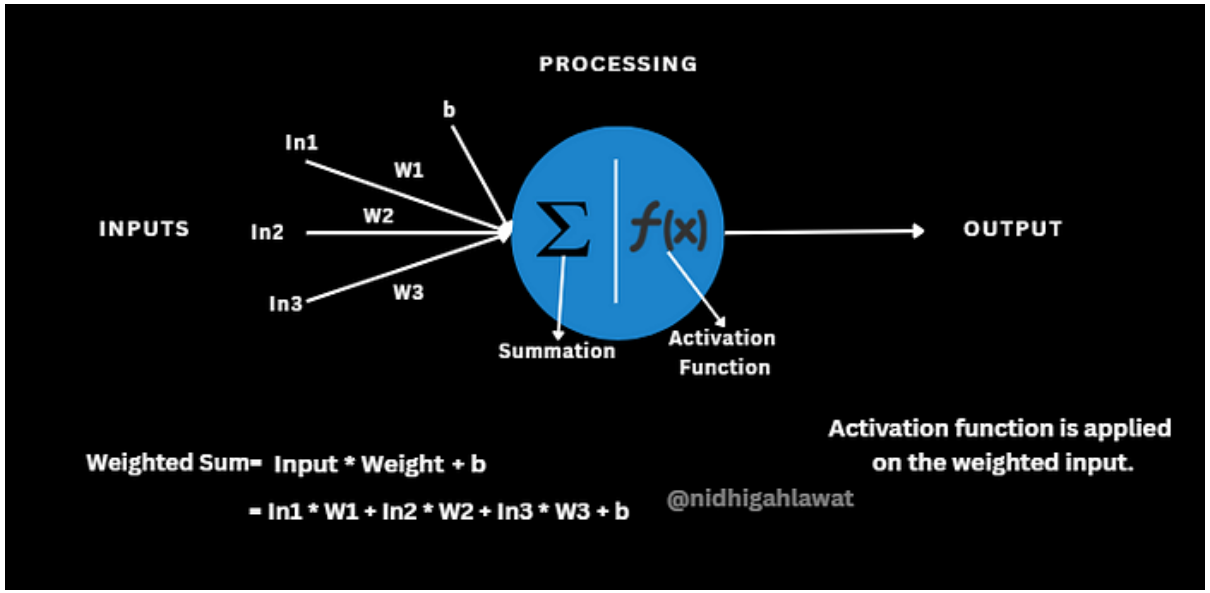
- biological neurons and artificial neurons have a designated place to take input, process it and produce output.



Neurons: Inputs

Inputs (Signals) = Nerve Impulses:

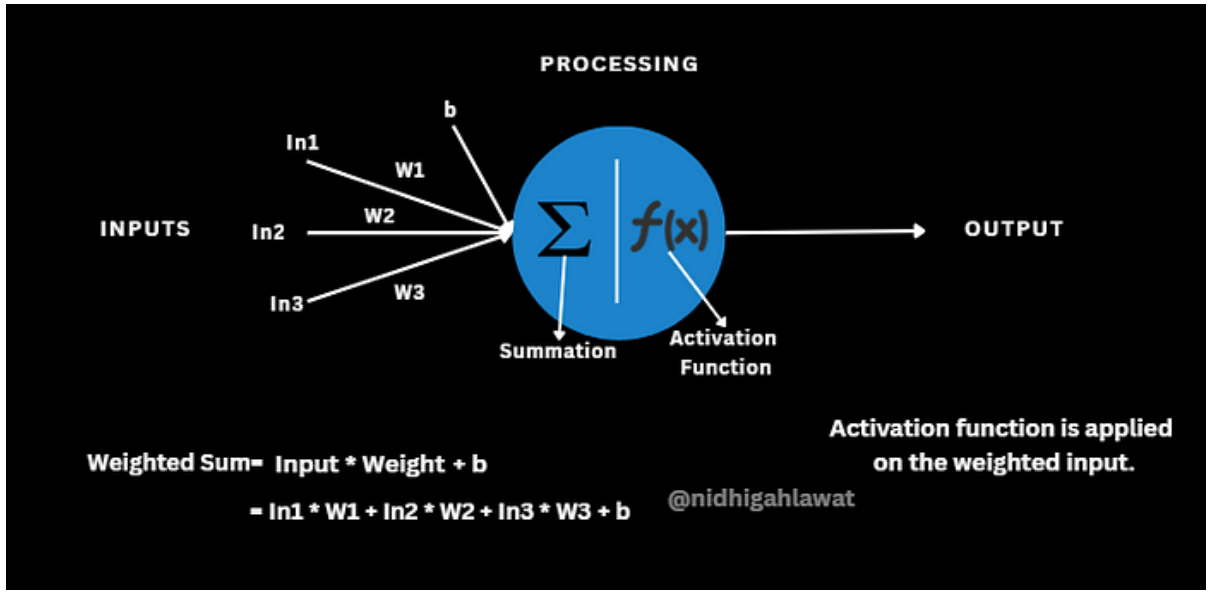
- Just like your brain sends **nerve impulses** to your muscles to tell them to contract, a neuron receives **inputs** (signals) from other neurons.
- **Example**: You want to lift a weight, so your brain sends signals to muscles (neurons) to contract.



Neurons: Weights

Weights = Strength of the Signal:

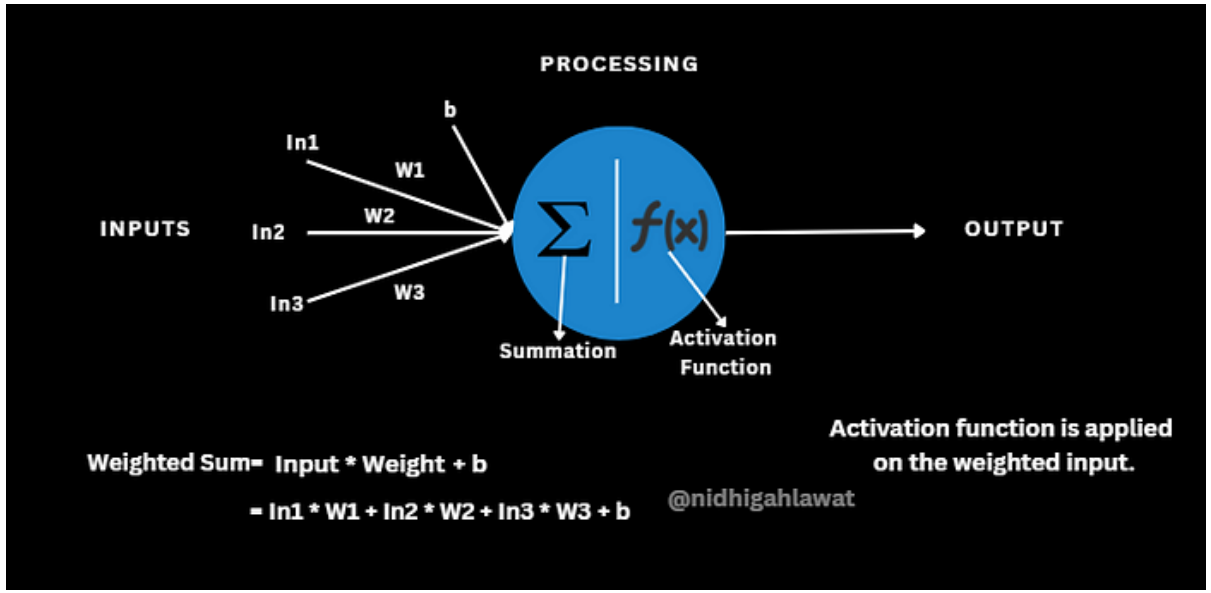
- Each input to the neuron has a **weight**, similar to how a signal might vary in strength. Some signals may have a stronger effect on the neuron than others.
- **Example**: The nerve impulses to a bicep muscle might be stronger (greater weight) when you want to lift a heavy weight and weaker (lower weight) when lifting a light object.



Neurons: Bias

Bias = Resting Tension (Threshold to Activate):

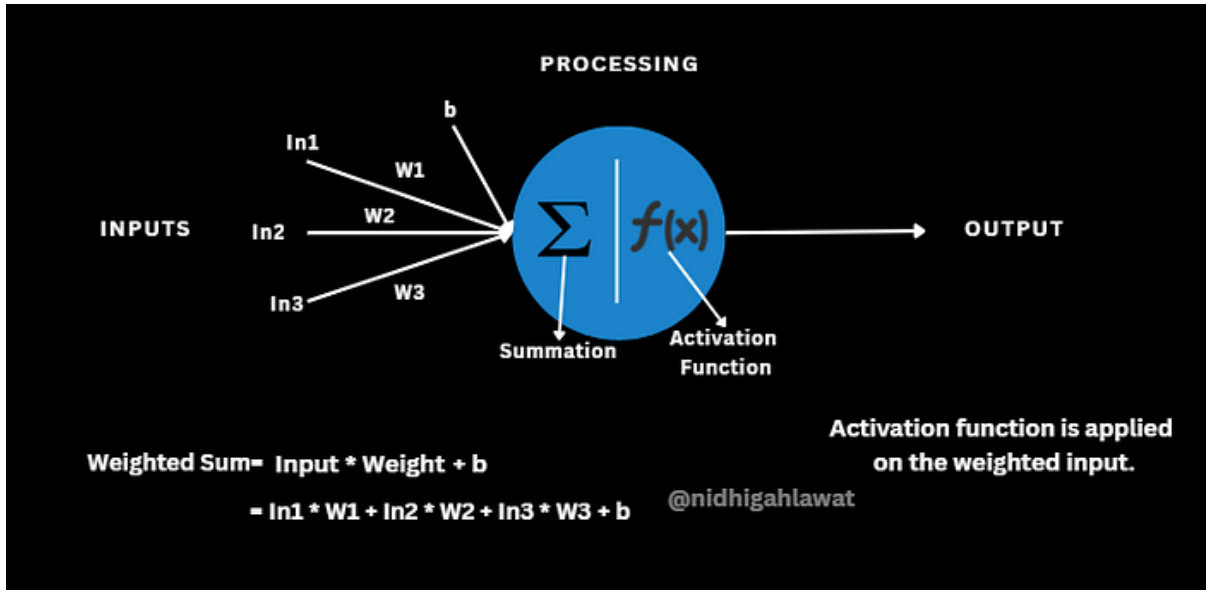
- The **bias** acts like the **resting tension** in the muscle — it's the minimum amount of energy or stimulus needed to trigger a response.
- **Example**: A muscle is always somewhat tense (bias), but it needs a certain level of stimulation to contract fully. If the signal (weighted sum of inputs) doesn't reach this threshold, the muscle won't contract (the neuron won't fire).



Neurons: Activation Function

Activation Function = Muscle Contraction (Decision to Fire):

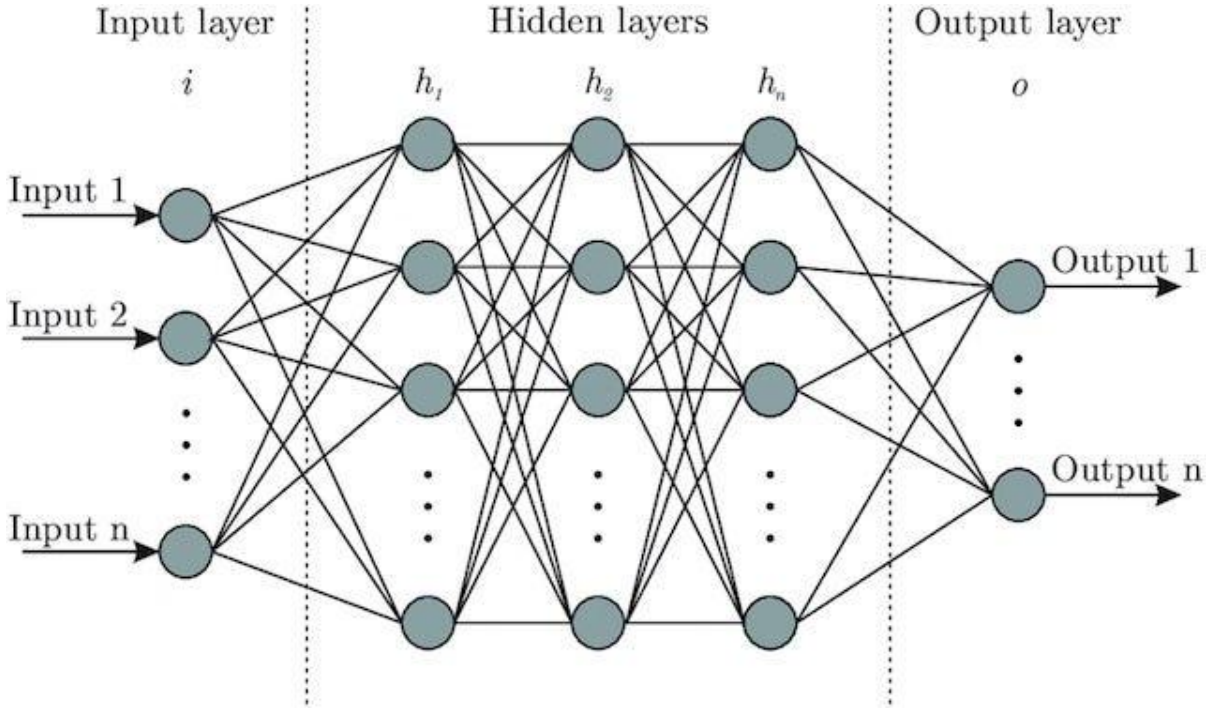
- The **activation function** determines whether the neuron will "fire" (or if the muscle will contract). It's like how your muscle contracts if enough force (stimulus) is applied.
- **Example**: If the signal is strong enough (i.e., the weighted sum exceeds the threshold), the muscle contracts (neuron activates). If not, it stays relaxed (neuron remains inactive).
- Common functions: **ReLU** (which is like saying "contract if the signal is strong enough, or stay relaxed").



Neural Network Architecture

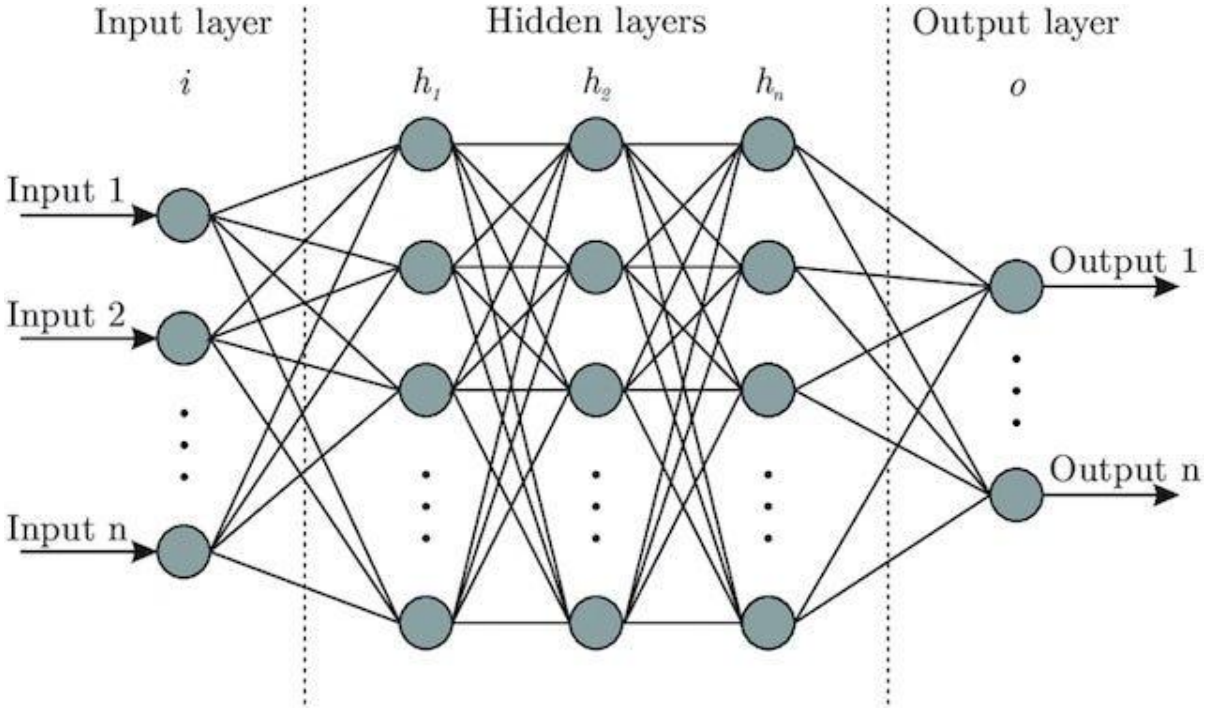
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- Input Layer: This is the ‘layer’ or step where the data is inputted into the Neural Network.
- Hidden Layer: These layers are where the process of interpreting the data happens. The data is filtered, and becomes more abstract.
- Output Layer: This layer is the output, or interpretation. In the case of an image classification of a bird, this is where the computer says “Yep, that’s a pigeon.”
- Node: These are the things that make up the layers. Each node is a filter that takes in input (the data from the previous layer) and is responsible for recognizing a specific pattern, outputting whether or not it sees the pattern.



Neural Network Architecture

- Activation Function: This is the function for the node, that takes all the inputs from the previous layer, and compresses into the weight (a decimal between 0 and 1)
- Weight: This is the output of each activation function, a number between 1 and 0 that represents the pattern the activation function was looking for.
- Parameter Value: Decides how high the weight needs to be for the information to pass to the next hidden layer.

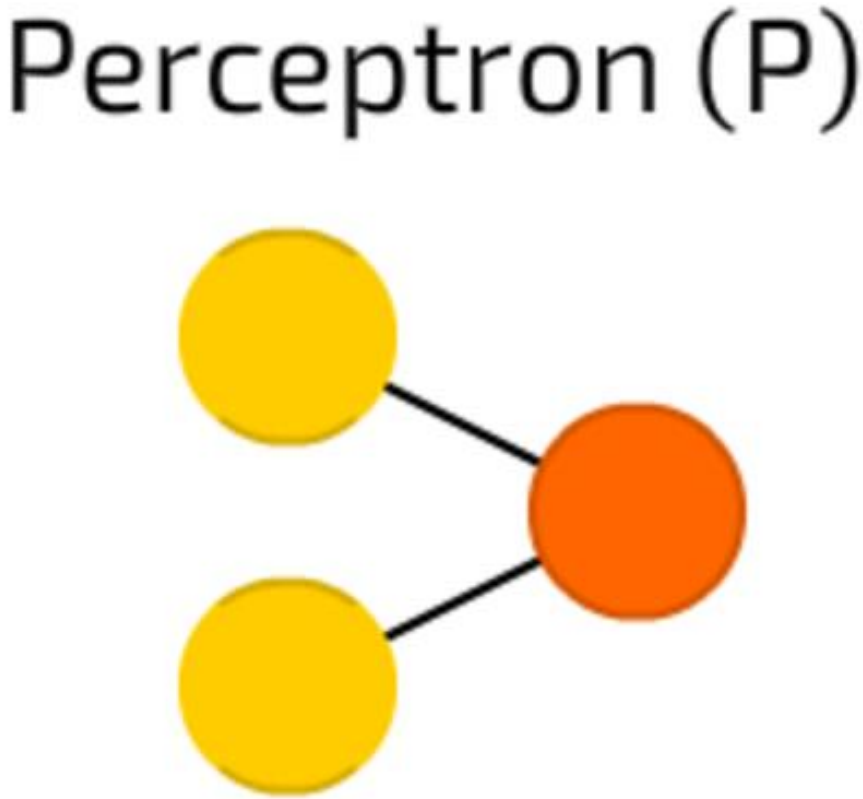


Contents

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Types of Neural Networks

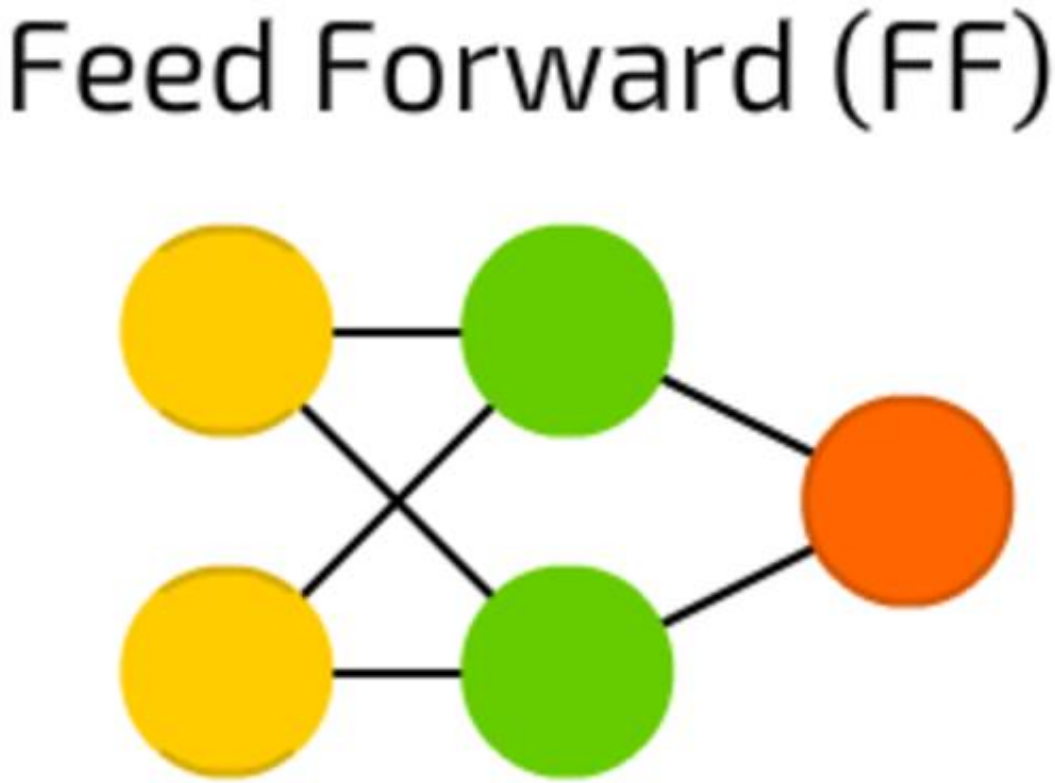
- **Perceptron**. The simplest and oldest model of Neuron, as we know it. Takes some inputs, sums them up, applies activation function and passes them to output layer. No magic here.



Types of Neural Networks

Feedforward Neural Networks (FNNs)

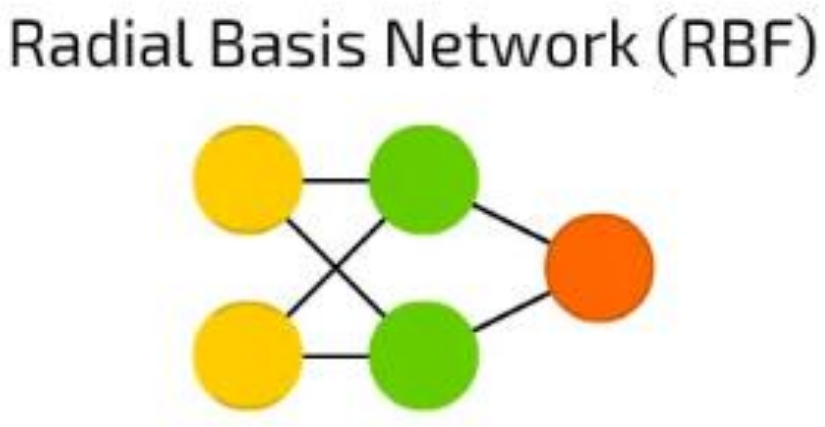
- Data flows from **input layer** to **output layer** in one direction, with no feedback loops.
- **Fully connected** layers, where each neuron in one layer is connected to all neurons in the next.
- Typically used for **classification** or **regression** tasks with fixed input-output mappings.



Types of Neural Networks

Radial Basis Function (RBF) Networks

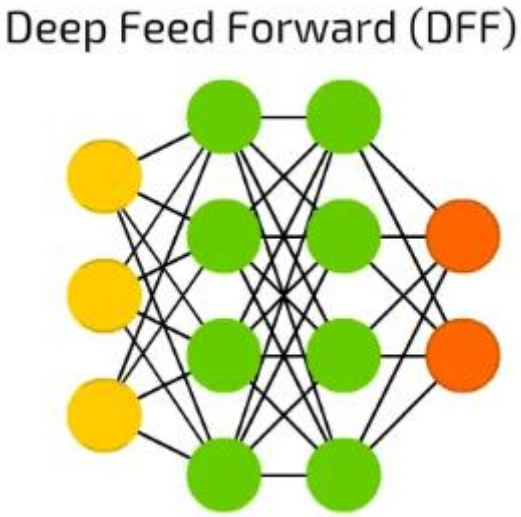
- Uses **radial basis functions** as activation functions, often Gaussian functions, to measure the distance between input and a **set of centers** (prototypes).
- Typically consists of an **input layer**, a **single hidden layer** with RBF neurons, and an **output layer** for classification or regression.
- **Non-linear transformation** of input space into a higher-dimensional space, making it effective for problems requiring non-linear decision boundaries.



Types of Neural Networks

Deep Feedforward Neural Networks (DFNNs)

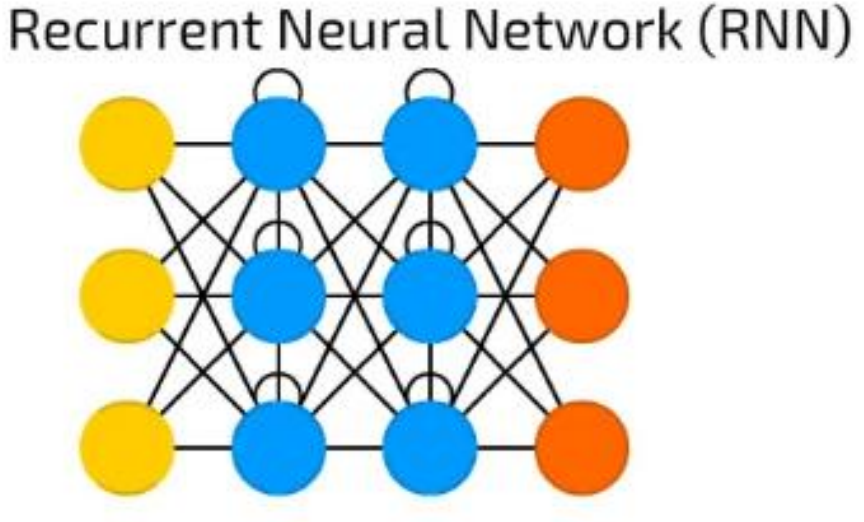
- **Deep architectures** with multiple hidden layers between the input and output layers, allowing for **complex representations** of data.
- Each layer performs weighted summation and applies an activation function, enabling the network to learn hierarchical features.
- Commonly used for **complex classification**, **regression**, and **representation learning** tasks due to their ability to capture intricate patterns.



Types of Neural Networks

Recurrent Neural Networks (RNNs)

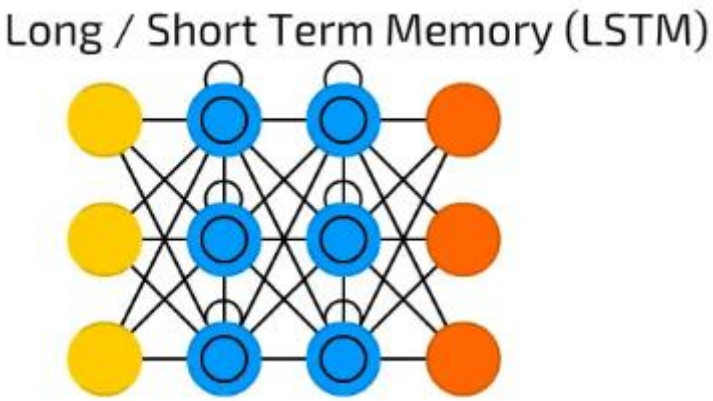
- Designed for **sequential data**, processing one element at a time while maintaining a **memory** of previous inputs.
- Uses loops to pass information from one time step to the next, capturing **temporal dependencies**.
- Suitable for tasks like **speech recognition** or **language modeling** where the order of inputs matters.



Types of Neural Networks

Long Short-Term Memory (LSTM) Networks

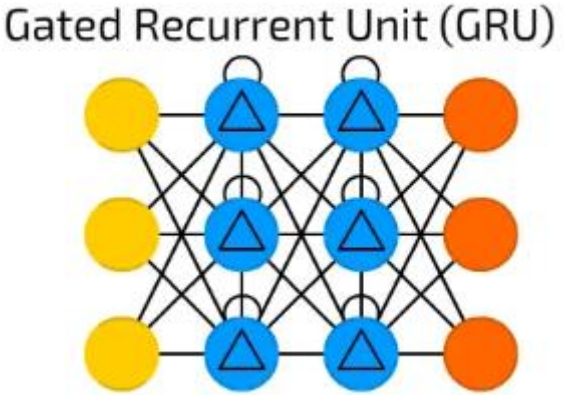
- A type of **Recurrent Neural Network (RNN)** designed to overcome the **vanishing gradient problem** by maintaining **long-term memory** of past inputs.
- LSTMs use **gates** (input, forget, and output) to control the flow of information and decide which data to remember or forget over time.
- Ideal for tasks involving **sequential data** with long-range dependencies, such as **speech recognition**, **language modeling**, and **time-series forecasting**.



Types of Neural Networks

Gated Recurrent Units (GRUs)

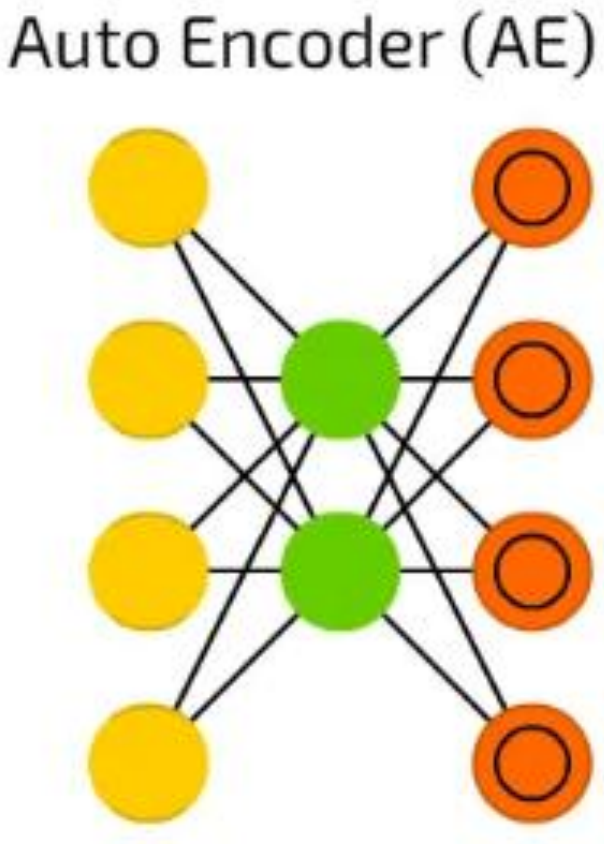
- A variation of **LSTM** networks that simplifies the architecture by using only two gates: the **update gate** and the **reset gate**.
- GRUs control the flow of information to preserve long-term dependencies and allow the network to decide what to keep or forget.
- Like LSTMs, GRUs are well-suited for tasks involving **sequential data**, but are often computationally more efficient due to their simpler design.



Types of Neural Networks

Autoencoders (AEs)

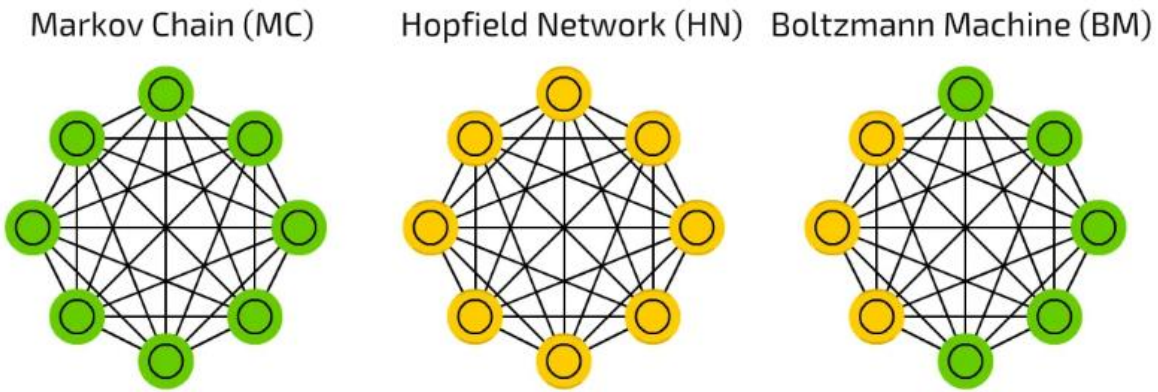
- **Unsupervised learning** models used for **dimensionality reduction** and **feature learning** by learning a compressed, lower-dimensional representation (encoding) of the input data.
- Composed of two parts: an **encoder** (maps input data to a compressed representation) and a **decoder** (reconstructs the data from the compressed form).
- Often used for **data compression**, **denoising**, or **anomaly detection**.



Types of Neural Networks

Markov Chains

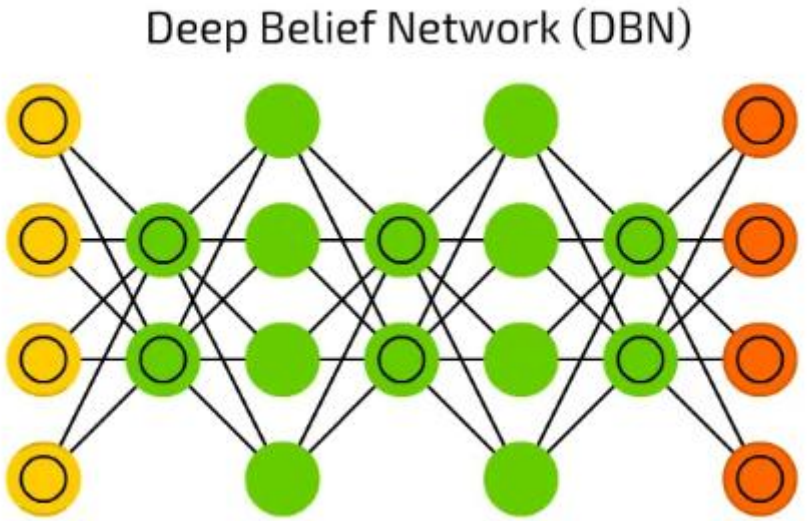
- A **stochastic process** where the future state depends only on the current state, and not on the sequence of events that preceded it (this is called the **Markov property**).
- Represented as a set of **states** and **transition probabilities**, describing the likelihood of moving from one state to another.
- Commonly used for modeling **sequential data** or systems that evolve over time, such as **weather prediction**, **speech recognition**, and **economics**.



Types of Neural Networks

Deep Belief Networks (DBNs)

- A type of **generative model** composed of multiple layers of **Restricted Boltzmann Machines (RBMs)**, which are stacked to form a deep network.
- **Unsupervised learning** is used to pre-train the network, followed by **fine-tuning** with supervised learning for classification tasks.
- Often used for **feature extraction**, **dimensionality reduction**, and **unsupervised pre-training** in complex models like deep neural networks.

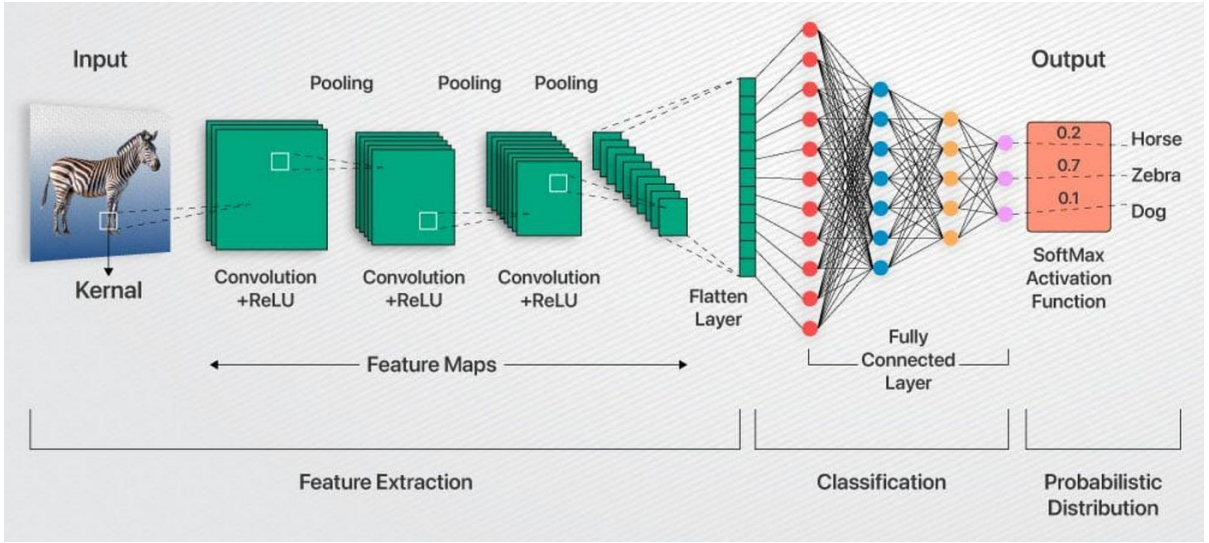


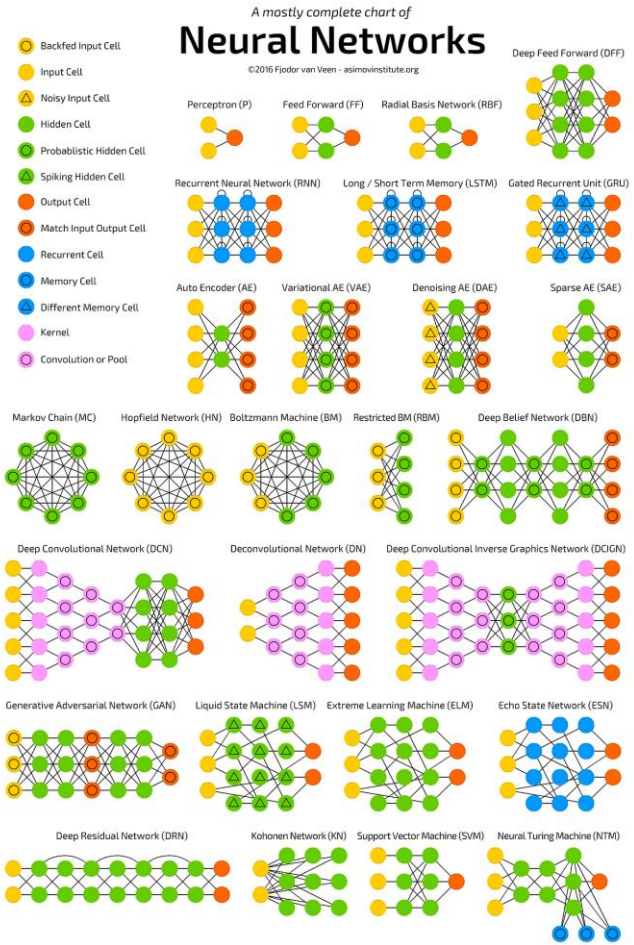
Types of Neural Networks

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Convolutional Neural Networks (CNNs)

- Designed for **grid-like data** (e.g., images, videos), CNNs utilize **convolutional layers** to automatically detect **spatial hierarchies** and local patterns (like edges, textures).
- Composed of layers such as **convolutional layers**, **pooling layers** (for downsampling), and **fully connected layers** (for final classification or regression).
- Primarily used in **image recognition**, **object detection**, and **video analysis** tasks due to their ability to learn hierarchical features in data.





https://medium.com/data-science/the-mostly-complete-chart-of-neural-networks-explained-3fb6f2367464

Contents

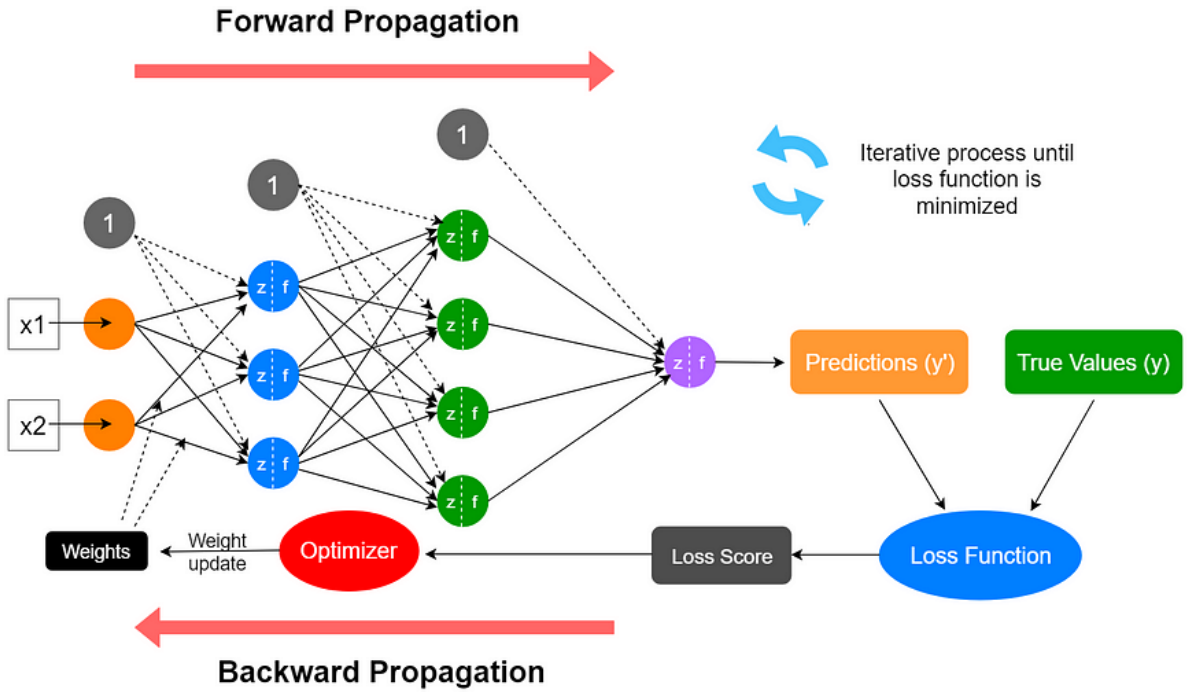
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Training a Neural Netwrok

The learning (training) process of a neural network is an iterative process in which the calculations are carried out forward and backward through each layer in the network until the loss function is minimized.

The entire learning process can be divided into three main parts:

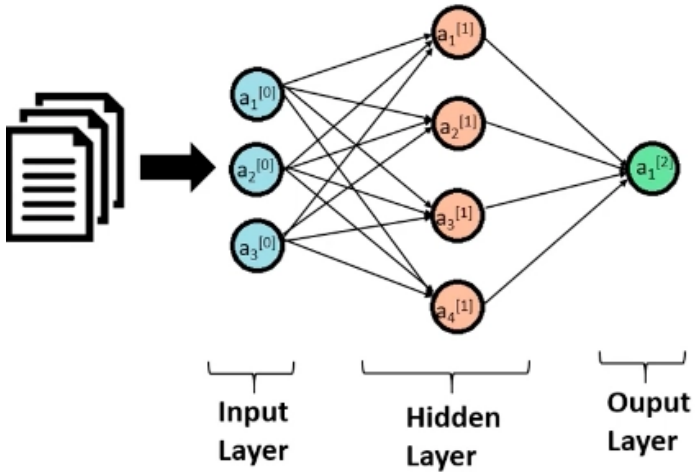
- **Forward propagation (Forward pass)**
- **Calculation of the loss function**
- **Backward propagation (Backward pass/Backpropagation)**



Forward Propagation

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- **Input Layer**: Data is fed into the input layer of the network.
- **Hidden Layers**: Data is processed through one or more hidden layers. Each neuron in the hidden layer performs a weighted sum of inputs and passes the result through an activation function.
- **Output Layer**: The final layer outputs the network's prediction.



$a_1^{[1]}$ = activation_function($W_{11}^{[1]} * a_1^{[0]} + W_{12}^{[1]} * a_2^{[0]} + W_{13}^{[1]} * a_3^{[1]} + B1$)

$a_2^{[1]}$ = activation_function($W_{21}^{[1]} * a_1^{[1]} + W_{22}^{[1]} * a_2^{[1]} + W_{23}^{[1]} * a_3^{[1]} + B1$)

n_h = size of hidden layer n_x = size of input layer

$$W = \begin{bmatrix} W_{11} & W_{12} & W_{13} \\ W_{21} & W_{22} & W_{23} \\ W_{31} & W_{32} & W_{33} \\ W_{41} & W_{42} & W_{43} \end{bmatrix} \begin{bmatrix} a_1^{[0]} \\ a2^{[0]} \\ a_3^{[0]} \end{bmatrix} = \begin{bmatrix} W_{11}^{[1]} * a_1^{[0]} + W_{12}^{[1]} * a_2^{[0]} + w_{13}^{[1]} * a_3^{[1]} \\ W_{21}^{[1]} * a_1^{[0]} + W_{22}^{[1]} * a_2^{[0]} + w_{23}^{[1]} * a_3^{[1]} \\ W_{31}^{[1]} * a_1^{[0]} + W_{32}^{[1]} * a_2^{[0]} + w_{33}^{[1]} * a_3^{[1]} \\ W_{41}^{[1]} * a_1^{[0]} + W_{42}^{[1]} * a_2^{[0]} + w_{43}^{[1]} * a_3^{[1]} \end{bmatrix}$$

shape(n_h, n_x) = shape (4, 3) shape(n_x, 1) = shape(3, 1) shape(n_h, 1) = shape(4, 1)

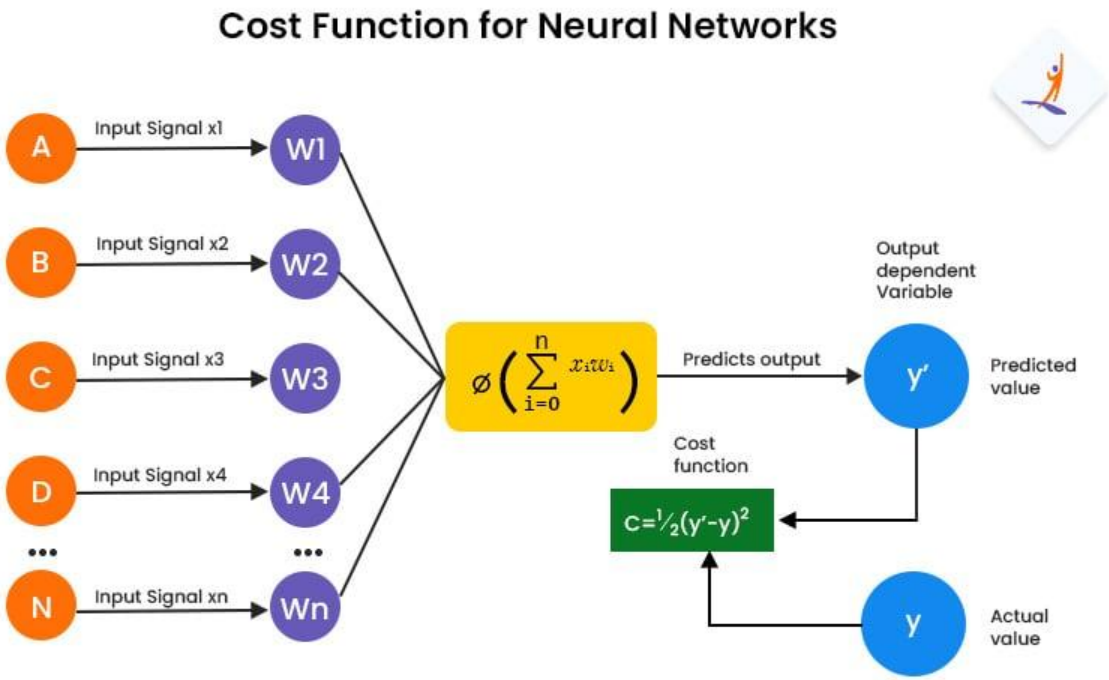
$$f\left(\begin{bmatrix} W_{11}^{[1]} * a_1^{[0]} + W_{12}^{[1]} * a_2^{[0]} + w_{13}^{[1]} * a_3^{[1]} \\ W_{21}^{[1]} * a_1^{[0]} + W_{22}^{[1]} * a_2^{[0]} + w_{23}^{[1]} * a_3^{[1]} \\ W_{31}^{[1]} * a_1^{[0]} + W_{32}^{[1]} * a_2^{[0]} + w_{33}^{[1]} * a_3^{[1]} \\ W_{41}^{[1]} * a_1^{[0]} + W_{42}^{[1]} * a_2^{[0]} + w_{43}^{[1]} * a_3^{[1]} \end{bmatrix} + B1 \right) = \begin{bmatrix} a_1^{[1]} \\ a2^{[1]} \\ a_3^{[1]} \\ a4^{[1]} \end{bmatrix} = A1$$

A1 = f (W1 * A0 + B1) **A2 = f (W2 * A1 + B2)**

*(Here, W1 is matrix containing all the weight associated with first layer)

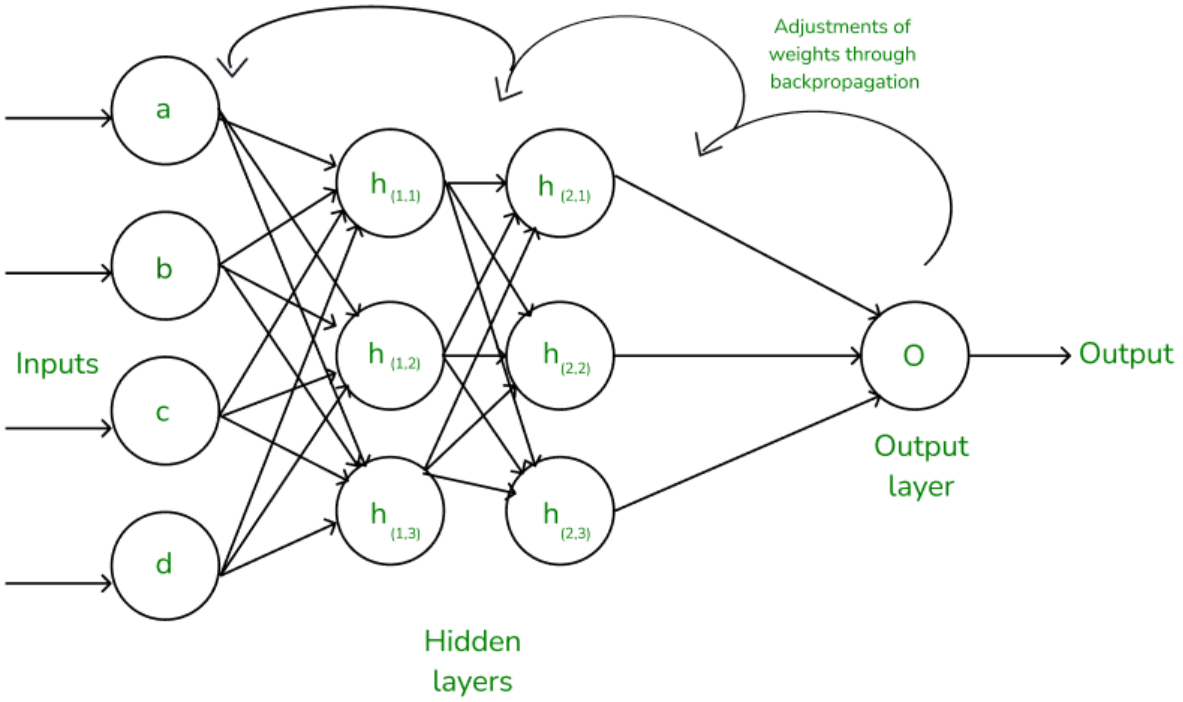
Loss Function

- **Objective**: The loss function calculates how far the network's predictions are from the actual target values.
  - For **regression tasks**, common loss functions include **Mean Squared Error (MSE)**.
  - For **classification tasks**, common loss functions include **Cross-Entropy Loss**.
- **Example**:
  - **MSE (Regression)**: Measures the average squared difference between predicted and actual values.
  - **Cross-Entropy Loss (Classification)**: Measures the difference between two probability distributions — predicted and true labels.



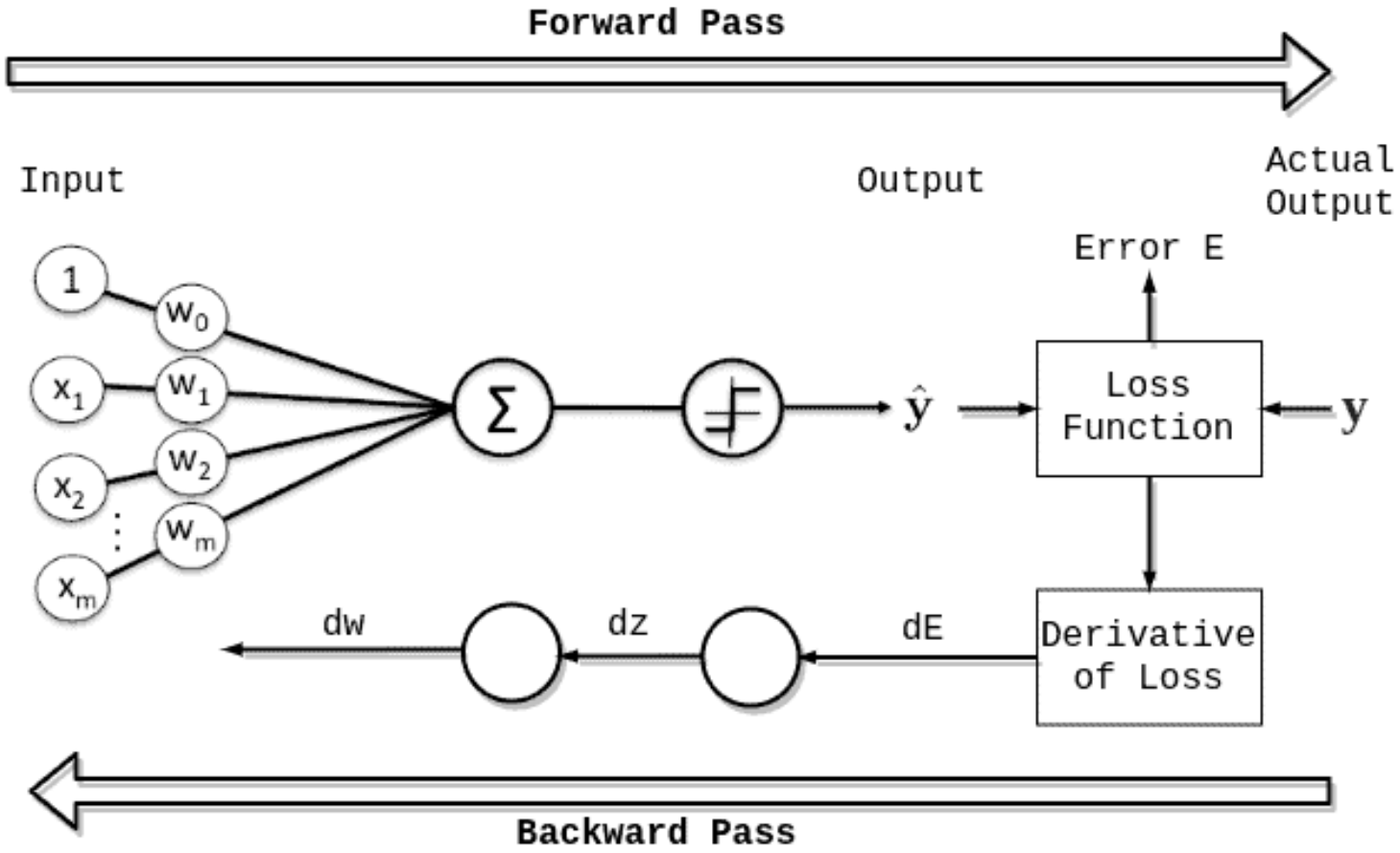
Backpropagation

- **What is Backpropagation?**
  - A technique used to adjust the weights of the neural network by computing the gradient of the loss function with respect to the weights.
  - The process involves calculating the **gradient** of the loss function for each weight and bias in the network and then using this information to update the weights.
- **Gradient Descent**: The optimization technique used to minimize the loss function. Weights are updated in the opposite direction of the gradient.
  - **Learning Rate**: Determines the size of the step taken in the direction of the gradient.
  - **Stochastic Gradient Descent (SGD)**: Updates weights after each batch of data.
  - **Mini-batch Gradient Descent**: A compromise between batch and stochastic gradient descent, updating weights after a small batch.



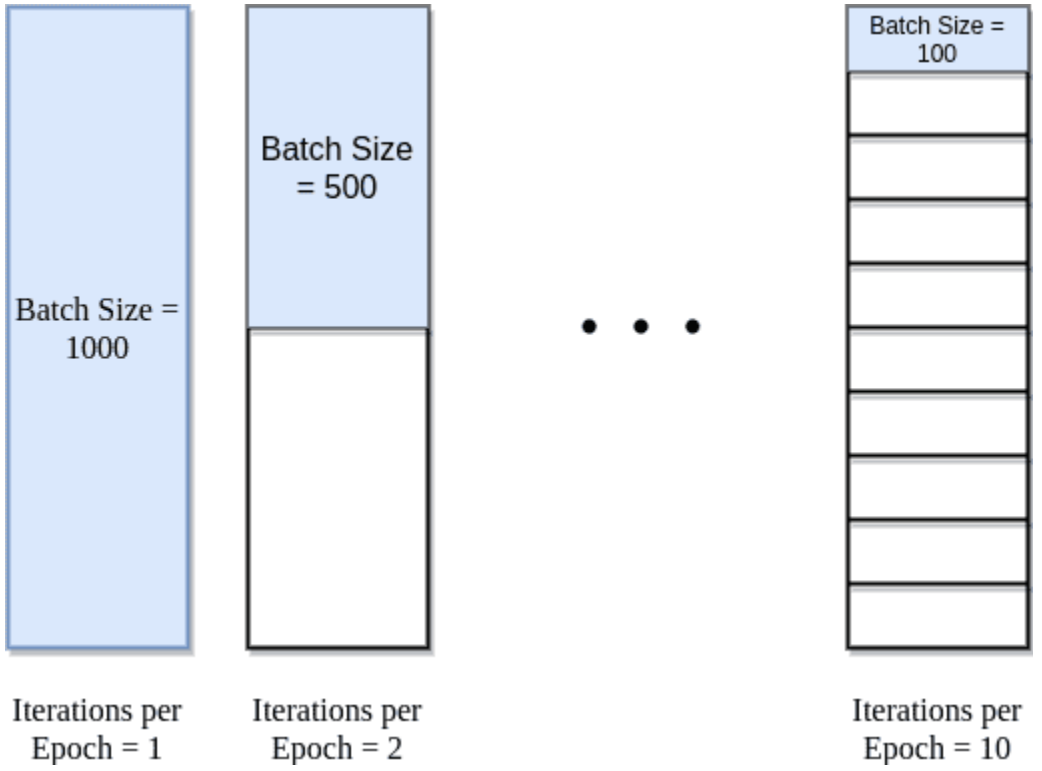
Epochs

- **Epoch**: One full pass through the entire training dataset. The model's weights are updated after each batch.



Batch Size

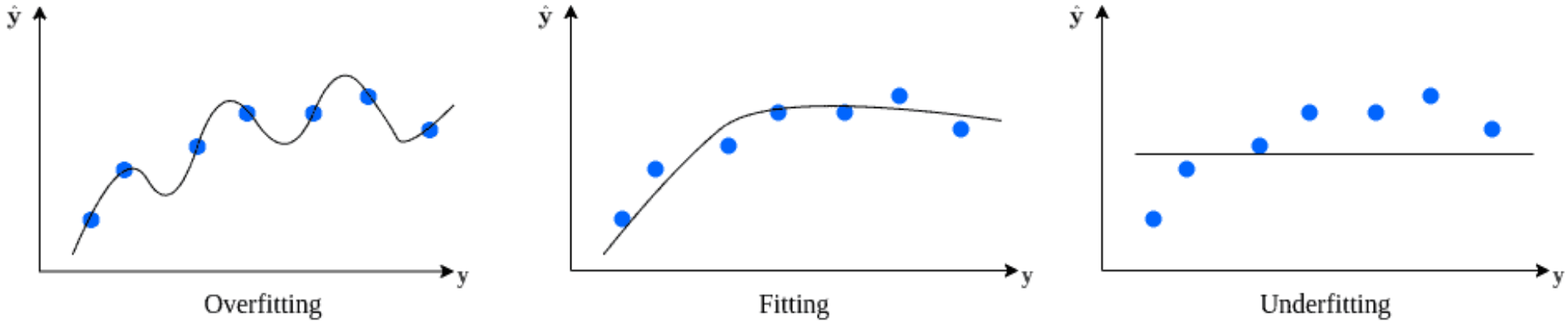
- An epoch is made up of one or more batches, where we use a part of the dataset to train the neural network. We call passing through the training examples in a batch an iteration.
- Number of training examples used in one iteration of updating weights.
  - **Small Batch Size**: Can provide better generalization but may result in noisy gradients.
  - **Large Batch Size**: More stable but might lead to overfitting.



Overfitting and Regularization

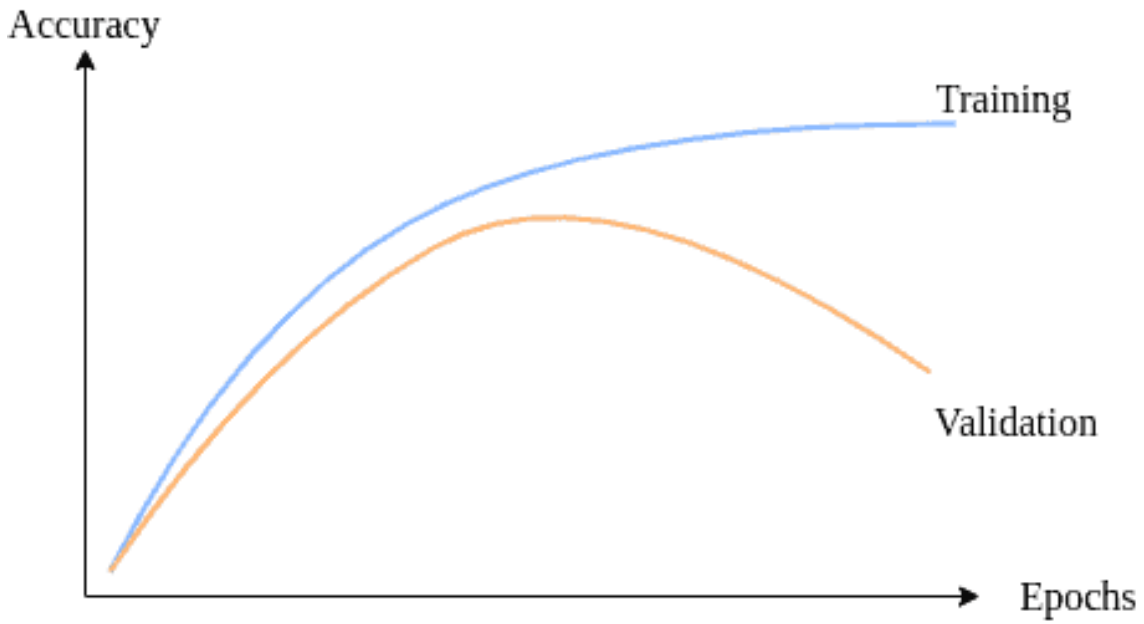
McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- **Overfitting**: Occurs when the model learns the training data too well, including noise and outliers, which negatively affects its ability to generalize to unseen data.
- **Regularization**: Techniques to prevent overfitting:
  - **Dropout**: Randomly deactivates a fraction of neurons during training to prevent reliance on specific neurons.
  - **L2 Regularization (Ridge)**: Adds a penalty to the loss function proportional to the square of the magnitude of weights, discouraging large weights.
  - **L1 Regularization (Lasso)**: Adds a penalty proportional to the absolute value of the weights, leading to sparse models.



Model Evaluation

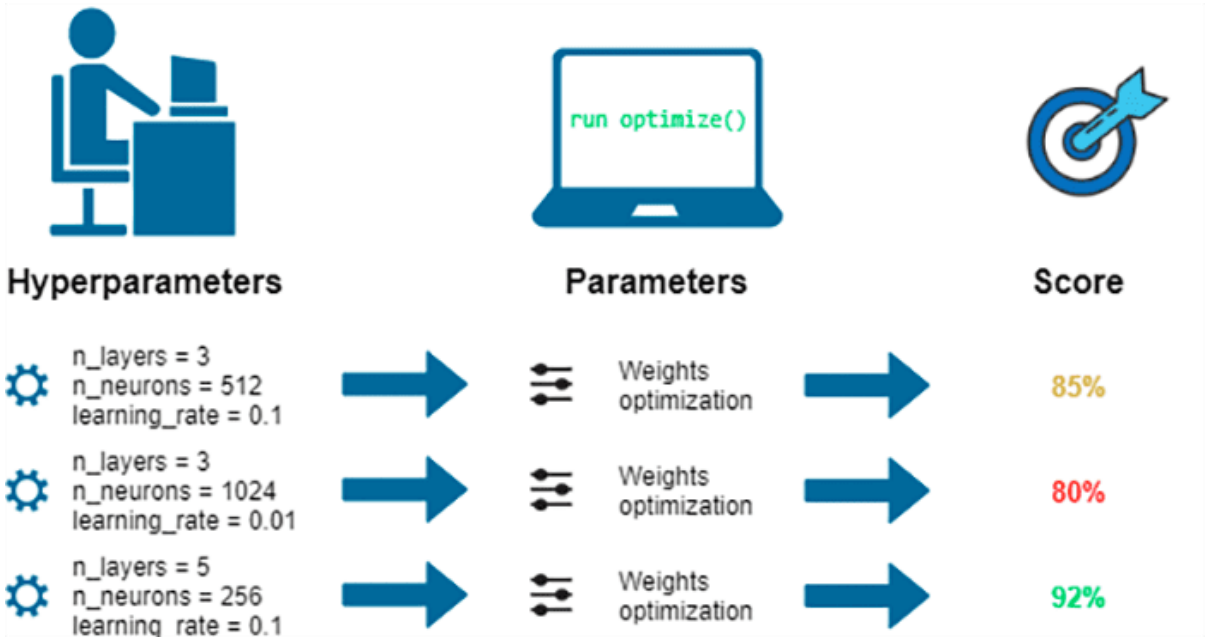
- **Validation Set**: Used during training to monitor model performance and adjust hyperparameters to avoid overfitting.
- **Test Set**: After training, the model is tested on data it has never seen to evaluate its true performance.
- **Metrics**: Common metrics include accuracy, precision, recall, and F1 score (for classification) or mean squared error (for regression).



Hyperparameter Tuning

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

- **What are Hyperparameters?**: Hyperparameters are parameters set before training, such as:
  - **Learning Rate**
  - **Batch Size**
  - **Number of Hidden Layers**
  - **Number of Neurons per Layer**
- **Grid Search & Random Search**: Methods for systematically exploring different combinations of hyperparameters to find the best-performing model.
- **Bayesian Optimization**: A more advanced technique for hyperparameter optimization.



Contents

McMaster University
ENGINEERING
W Booth School of Engineering Practice and Technology

Aspect	Traditional Machine Learning (ML)	Neural Networks (NN)
Definition	Broad field of algorithms that learn from data	Subset of ML inspired by the human brain
Data Dependency	Works well with small, structured datasets	Requires large datasets for good performance
Feature Engineering	Manual feature extraction is crucial	Automatically learns features from data
Model Complexity	Simpler models (e.g., Decision Trees, SVM)	Complex models with many parameters
Interpretability	Easier to interpret and explain	Often a “black box”
Examples	Linear Regression, SVM, Random Forest	CNN, RNN, Transformers
Training Time	Faster to train	Slower, more computationally intensive
Performance	Great for structured/tabular data	Excels at unstructured data (images, audio, text)
Use Cases	Fraud detection, churn prediction, recommendation systems	Image classification, speech recognition, language modeling