

AI and Deep Learning



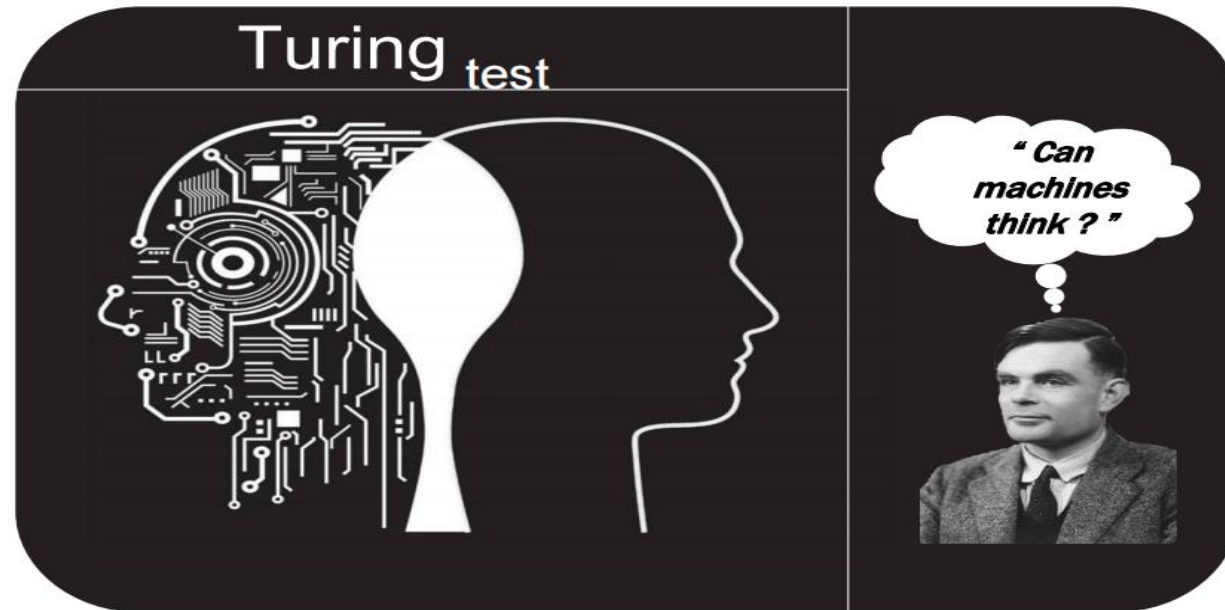
Artificial Intelligence

Artificial intelligence (AI) is a branch of computer science that attempts to simulate human intelligence in machines.



The Turing Test

In 1950, Alan Turing wrote a paper proposing a test that looked at whether a computer can think like a human being.

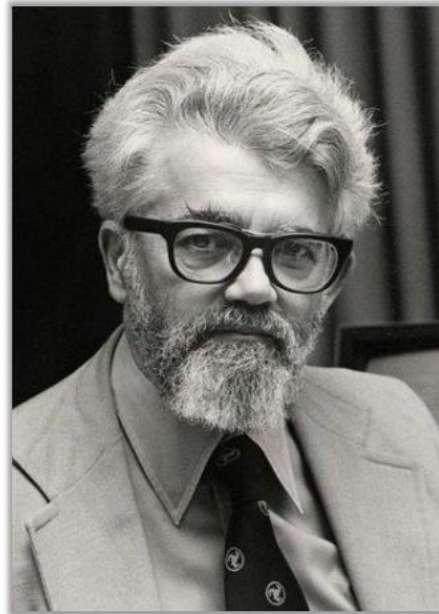


The paper focused on answering the question, "Can machines think?"



Father of AI

In 1955, John McCarthy coined the term “**Artificial Intelligence**” in a conference at Dartmouth.



He was very influential in the early development of AI and also helped develop the Lisp programming language.

State of AI

Artificial Intelligence is already proliferating in our lives. For example:



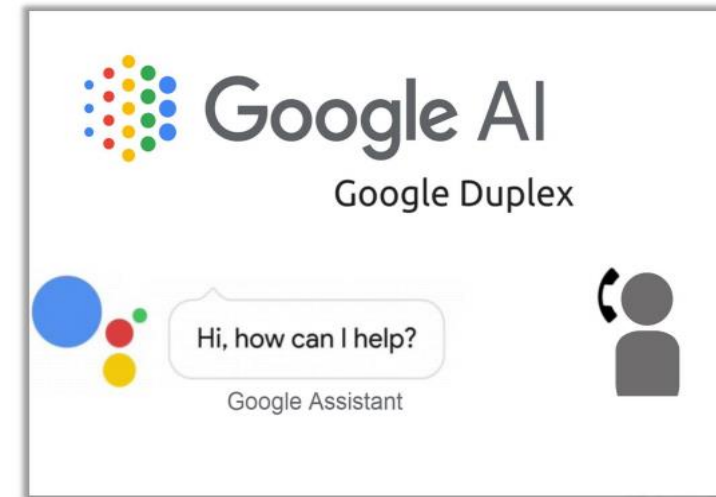
Pepper

Can recognize human faces
and basic emotions



Da Vinci® Surgical Systems

Can perform
minimally invasive surgeries



Google Duplex

Can make reservations
or appointments, over the phone

Advantages of AI

Error Reduction

With incredible precision, accuracy, and speed, AI has a low error rate compared to humans.

Digital Assistants

AI can assist users in their daily chores and activities.

Difficult Explorations

AI can work in hostile environments that would injure or kill humans to complete dangerous tasks



Medical Applications

AI has a significant application in early detection and monitoring of various diseases.

Repetitive Jobs

Machines think faster than humans and can perform repetitive jobs swiftly.

No Breaks

Machines, unlike humans, don't need to sleep, rest, take breaks, or get entertained.

Limitations of AI

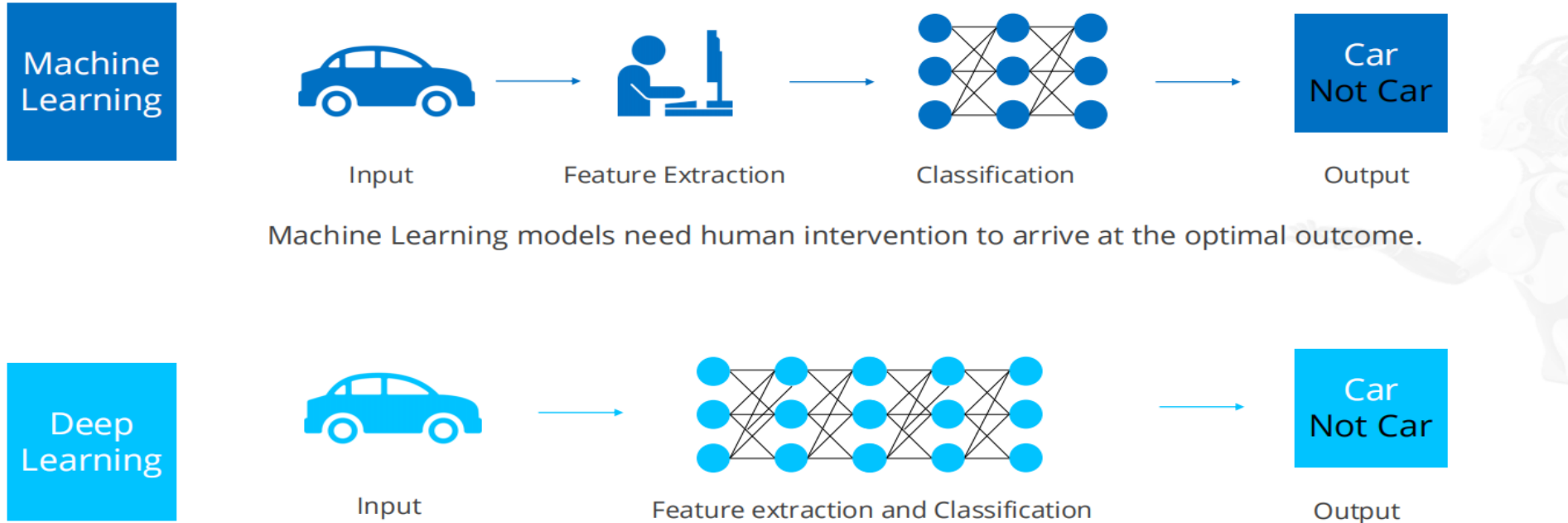
- 1 Very challenging to capture all traits of human intelligence
- 2 Current focus of development lies with lucrative domain specific applications
- 3 Limited ability with respect to the diversity of domain problems
- 4 Unpredictability or failure in emergency situations
- 5 High cost for R&D and production



Deep Learning

ML vs. DL

Deep Learning is a subset of Machine Learning and is used to extract useful patterns from data.



Deep Learning models make predictions independent of human intervention.

AI vs. ML vs. DL

ARTIFICIAL INTELLIGENCE

A technique which enables machines to mimic human behavior



MACHINE LEARNING

Subset of AI technique which use statistical methods to enable machines to improve with experience



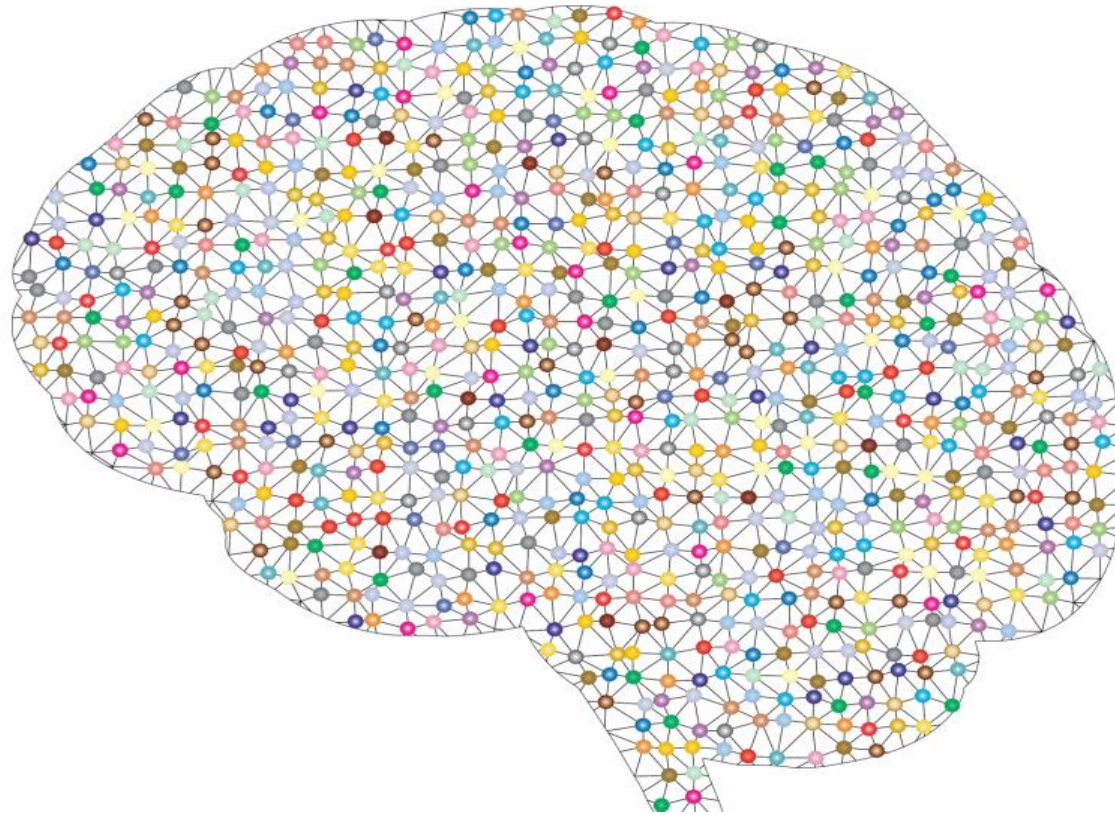
DEEP LEARNING

Subset of ML which makes the computation of multi-layer neural network feasible



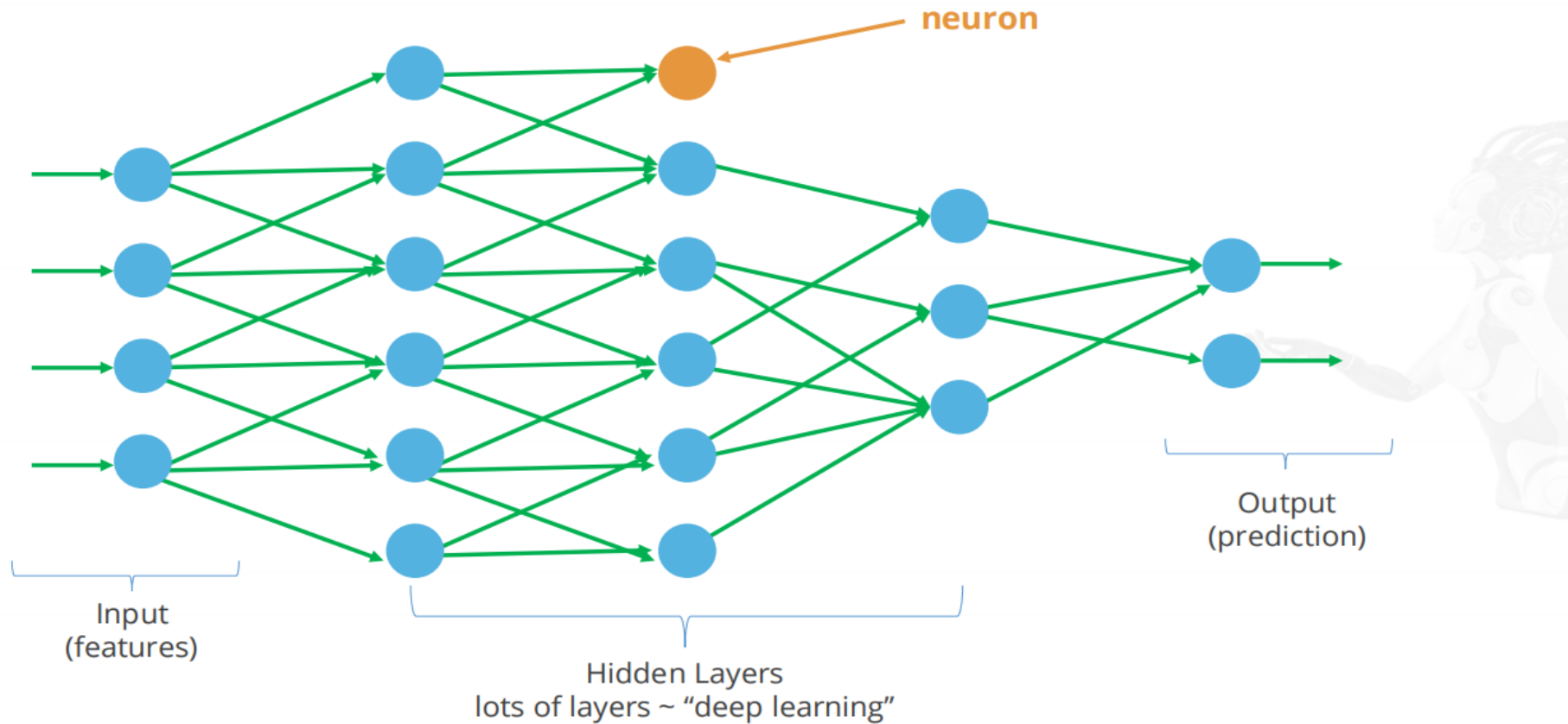
Neural Networks

Deep Learning models utilize artificial neural networks (ANN) that are inspired by the biological neural network of the human brain.



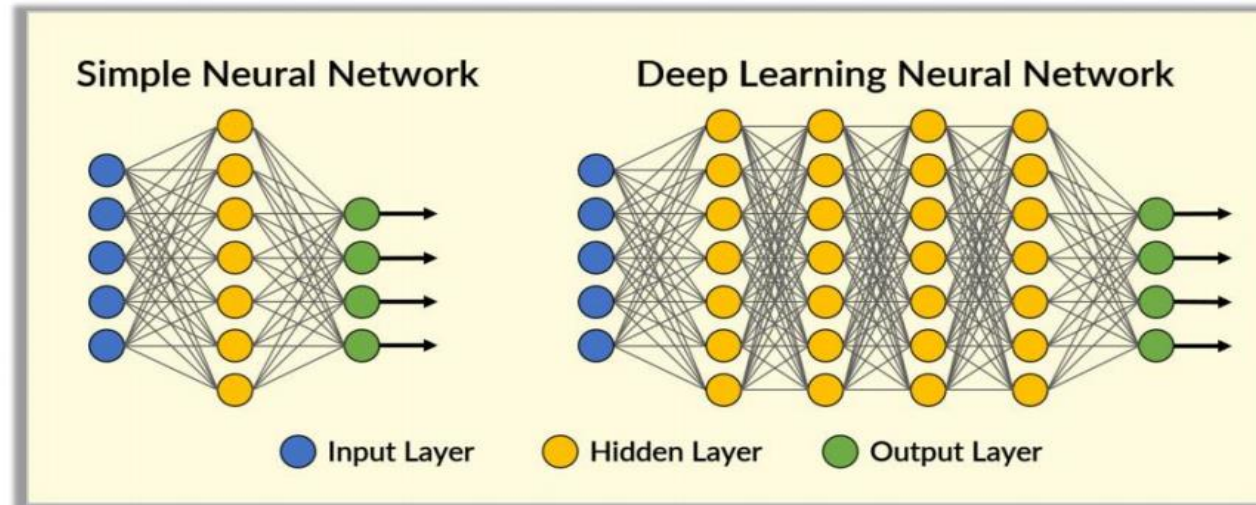
Deep Learning

The deep in Deep Learning refers to the large number of layers of neurons that help to learn various representations of data.



Network Architecture

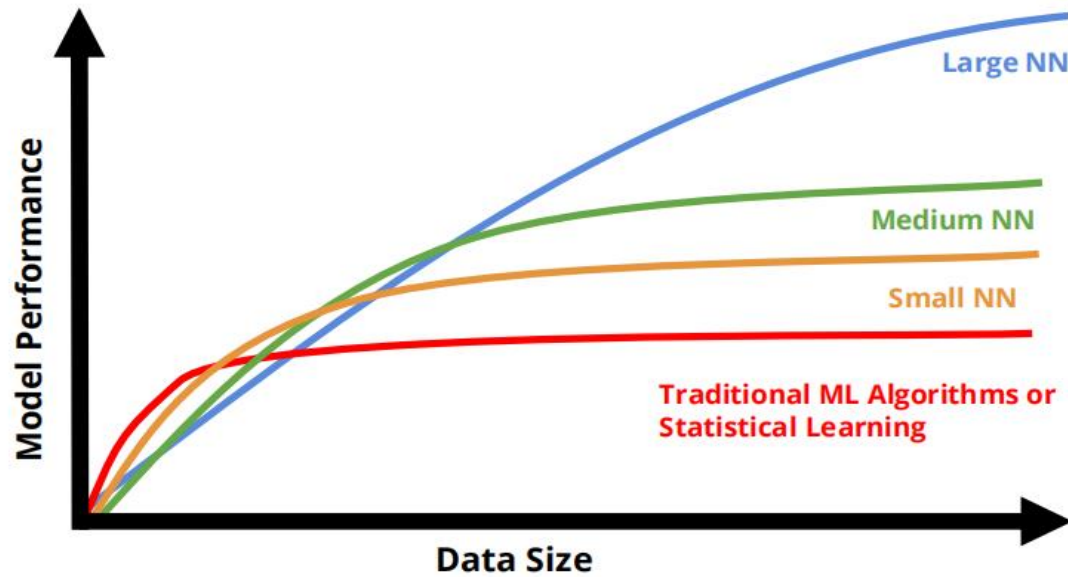
One can programmatically set the number and type of neural layers and the number of neurons comprising each layer.



Shallow neural networks consist of only 1 or 2 hidden layers.

Performance

Deep AI models have greater precision than conventional techniques but require more information to train and attain this precision.



Self-Driving Cars

A self-driving car uses a variety of cameras and sensors to perceive its surroundings and moves safely with little or no human input.



Why the Excitement for Self-Driving?

Self-driving vehicles will be a significant innovation in transportation since they can radically transform how people and goods move around.



Autonomous vehicles will help blind, elderly, and disabled people to move independently, regardless of distance.



Self-driving trucks can drastically reduce delivery times and accidents. Driver and maintenance costs will also reduce.



Advantages of Self-Driving



Less stressful driving

Automated lane changes, parking, and braking



Energy efficiency

Improved fuel efficiency



Reduced crash rates

Driver error causes 90% of all crashes



Ease traffic congestion

Better coordinated traffic leads to less congestion



Over the air updates

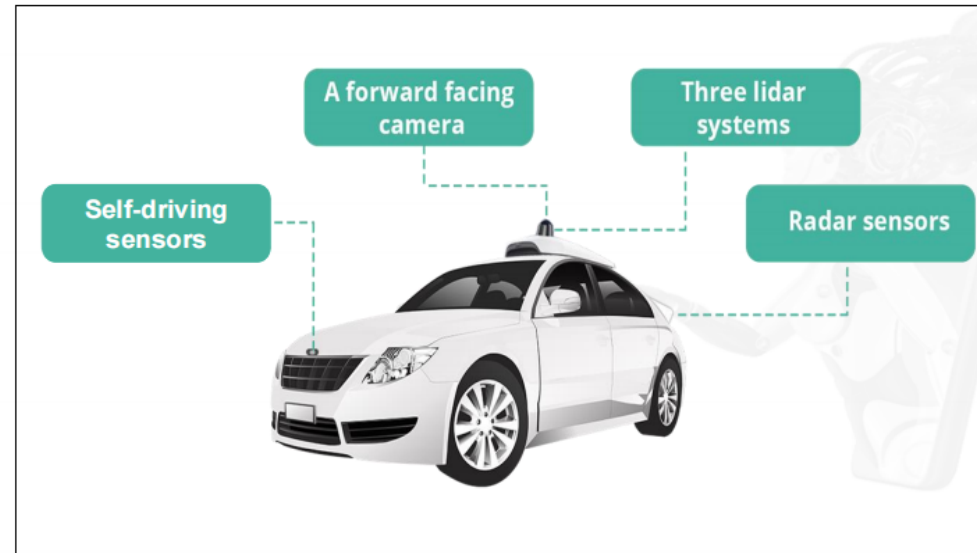
Neural nets get better with more data

How Does a Self-Driving Car See?

- 1 Using high-powered sensors and cameras, the vehicle creates and maintains a map of its environment.

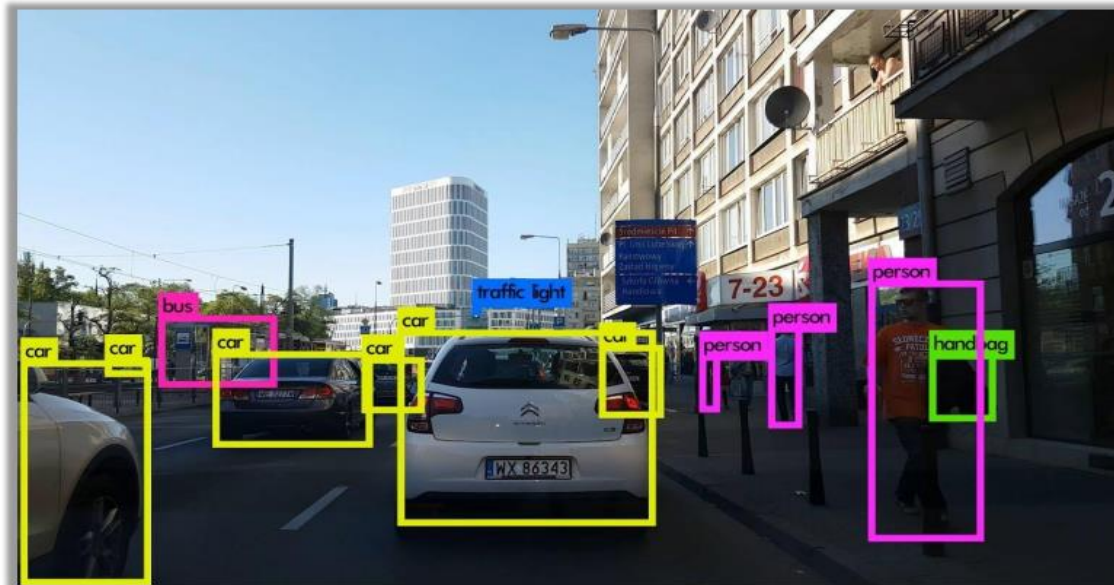
- 2 A powerful onboard computer processes the information in a matter of milliseconds using hard-coded rules and obstacle avoidance algorithms.

- 3 Then, it plots the route and sends instructions regarding vehicle's steering, acceleration, and braking.



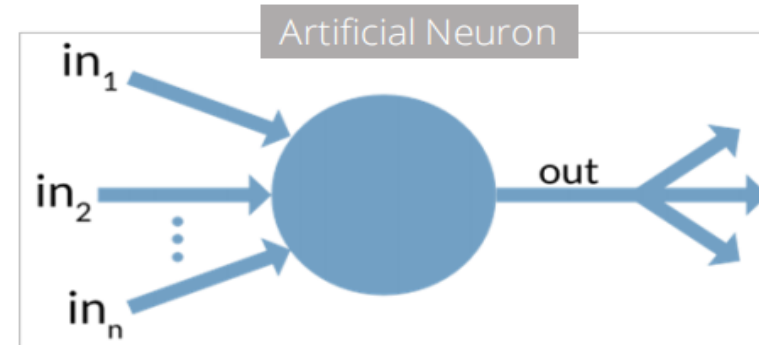
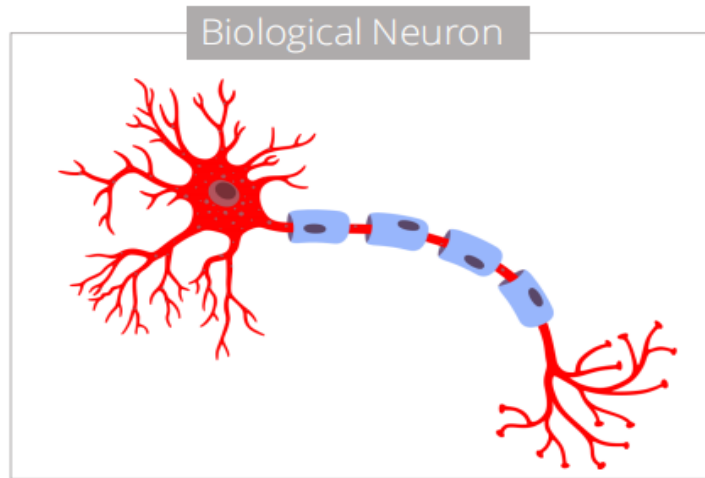
How Does a Self-Driving Car See?

Precise detection of vehicles, pedestrians, and street signs help cars drive as safely as humans.



Detecting moving objects in video streams is a promising and challenging task for modern developers.

Rise of Artificial Neurons



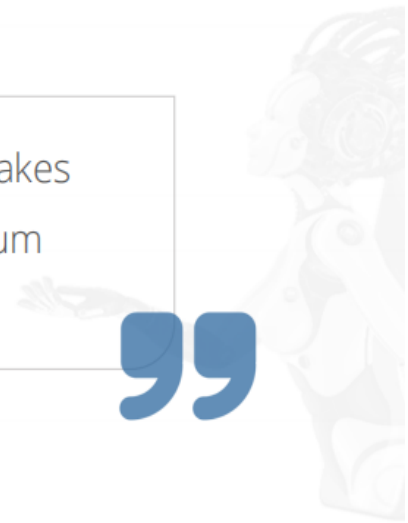
- Researchers Warren McCullock and Walter Pitts published their first concept of simplified brain cell in 1943.
- Nerve cell was considered similar to a simple logic gate with binary outputs.
- Dendrites can be assumed to process the input signal with a certain threshold such that if the signal exceeds the threshold, the output signal is generated.

Definition of Artificial Neuron

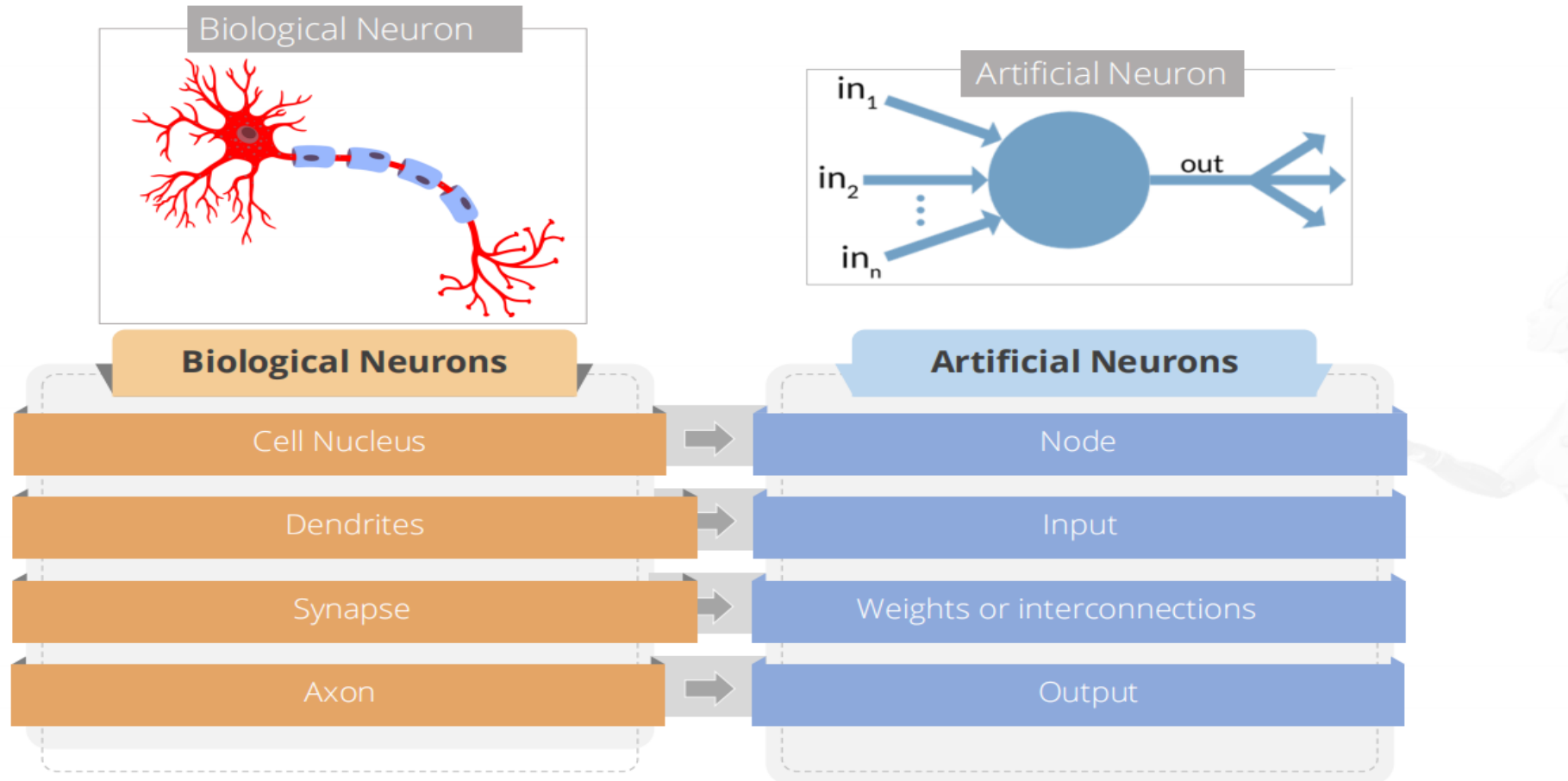
“

An artificial neuron is analogous to biological neurons, where each neuron takes inputs, adds weights to them separately, sums them up, and passes this sum through a transfer function to produce a nonlinear output.

”

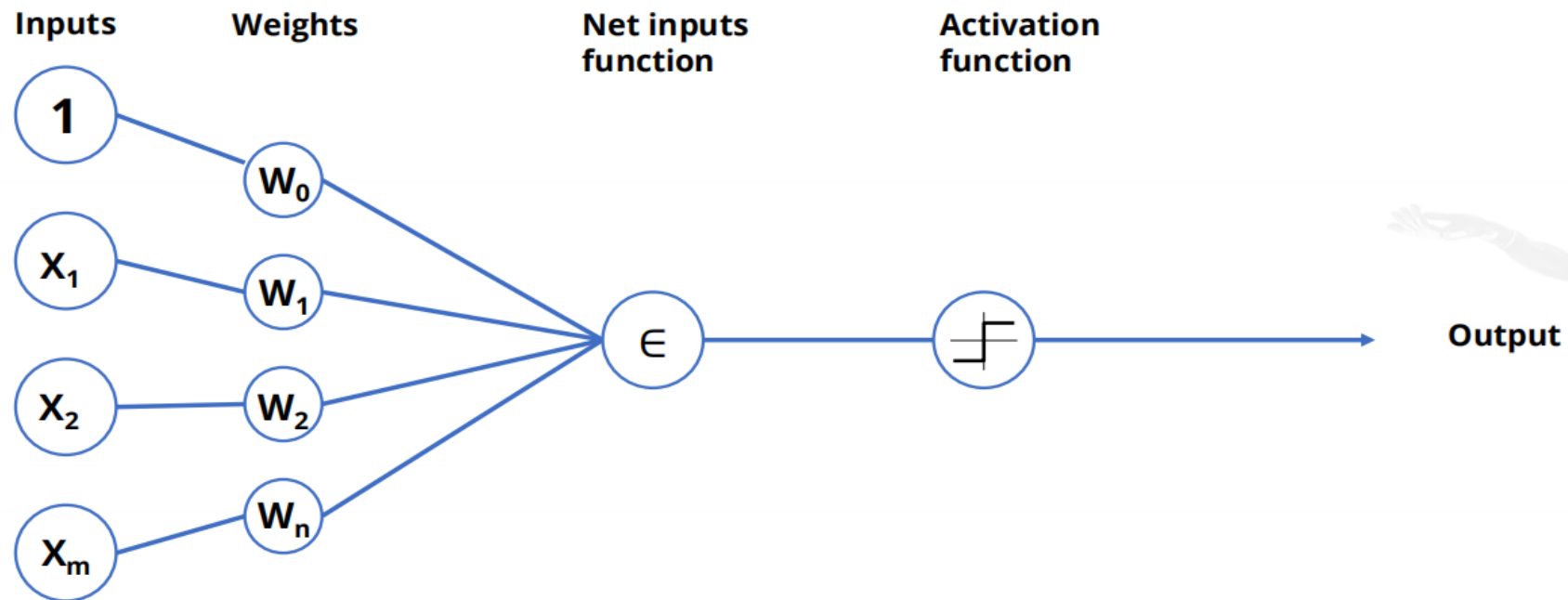


Biological Neurons and Artificial Neurons: A Comparison

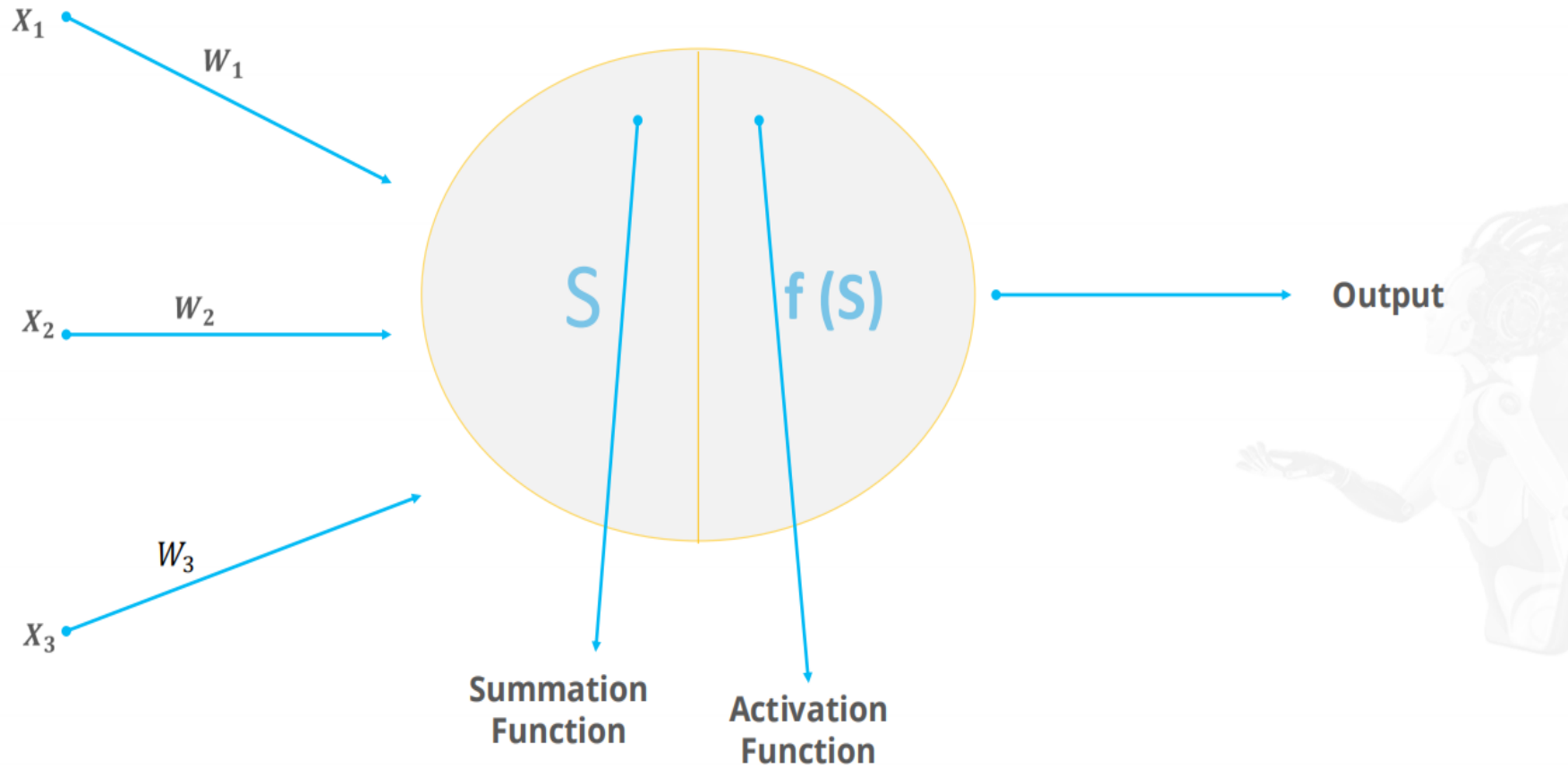


Perceptron

- Single layer neural network
- Consists of weights, the summation processor, and an activation function

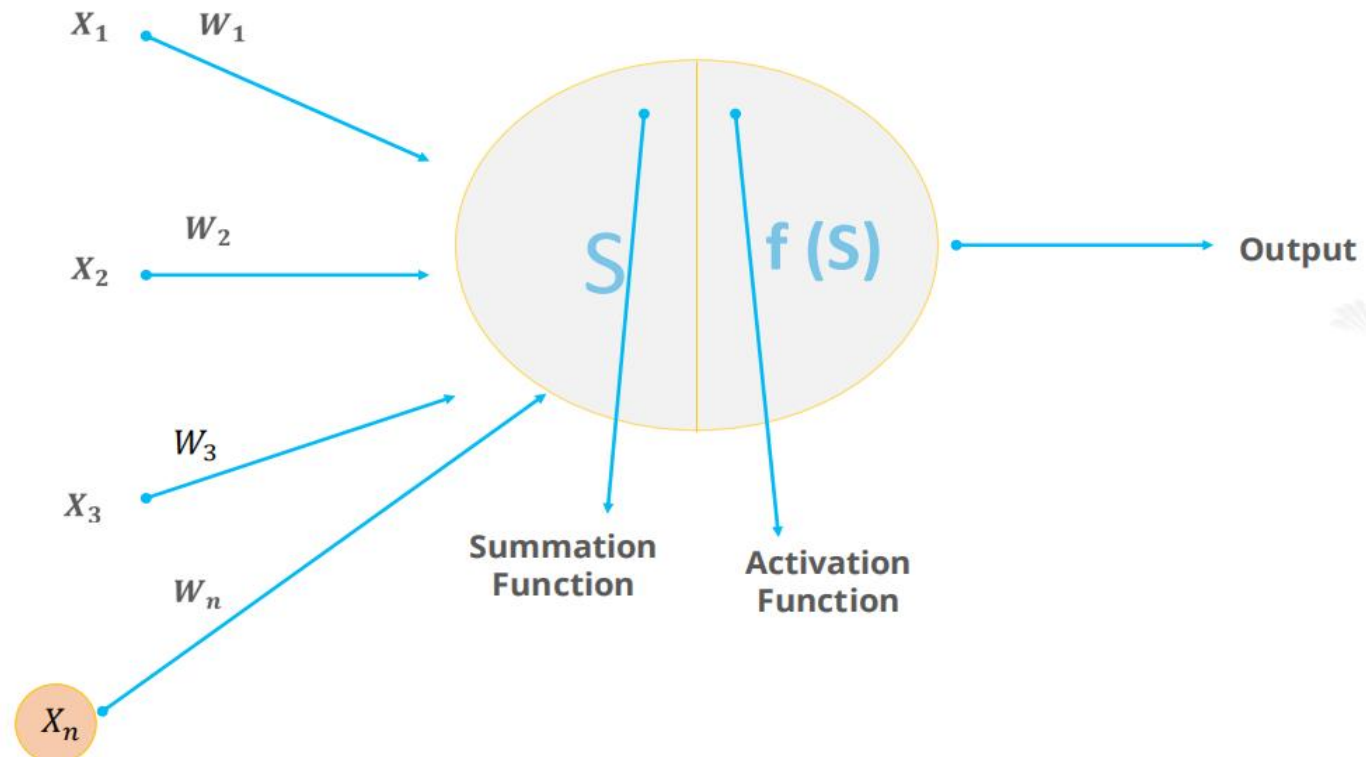


Perceptron: The Main Processing Unit



Weights and Biases in a Perceptron

While the weights determine the slope of the equation, bias shifts the output line towards left or right.



Weight and Bias:

► **Weight:**

- **Simple Definition:** In machine learning and neural networks, a weight is a parameter that determines the importance or strength of the input feature in relation to the output prediction.
- **Why It's Important:** Weights are crucial because they help the model learn and adjust how each input feature affects the final output. By adjusting the weights during training, the model learns to make accurate predictions.

► **Bias:**

- **Simple Definition:** A bias is an additional parameter in machine learning models that allows the model to shift the output function to better fit the data.
- **Why It's Important:** Biases are important because they help the model make predictions even when all input features are zero. They allow the model to better fit the data by providing the flexibility to adjust the output independently of the input values.

Activation Function:

- ▶ **Simple Definition:** An activation function is a mathematical function applied to a neural network's output to introduce non-linearity, allowing the network to model complex relationships in the data.
- ▶ **Why It's Important:** Activation functions are crucial because they enable neural networks to learn and represent complex patterns. Without them, the network would only be able to model linear relationships, severely limiting its capability.

Activation Function:

► Importance:

1. **Introduces Non-linearity:** Activation functions allow neural networks to capture non-linear relationships in the data, which is essential for solving complex problems.
2. **Enables Deep Learning:** By stacking multiple layers with non-linear activation functions, deep neural networks can learn hierarchical features and representations.
3. **Differentiability:** Most activation functions are designed to be differentiable, which is important for the backpropagation algorithm used in training neural networks.
4. **Feature Learning:** Different activation functions can help in learning different types of features, providing flexibility in designing neural networks.

Common Activation Functions:

1. Sigmoid: $\sigma(x) = \frac{1}{1+e^{-x}}$

- Output range: (0, 1)
- Commonly used in binary classification.

2. Tanh: $\tanh(x) = \frac{e^x - e^{-x}}{e^x + e^{-x}}$

- Output range: (-1, 1)
- Useful for zero-centered outputs.

3. ReLU (Rectified Linear Unit): $\text{ReLU}(x) = \max(0, x)$

- Output range: $[0, \infty)$
- Popular due to its simplicity and efficiency in training deep networks.

Common Activation Functions:

4. Leaky ReLU: $\text{Leaky ReLU}(x) = \max(0.01x, x)$

- Addresses the "dying ReLU" problem by allowing a small gradient when the unit is not active.

5. Softmax: $\text{Softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$

- Used in the output layer of classification networks to convert logits to probabilities.

Backpropagation

- ▶ Backpropagation, short for "backward propagation of errors," is a key algorithm used for training artificial neural networks, especially feed-forward networks. It is an efficient method for calculating the gradient of the loss function with respect to the weights in the network, which allows for the adjustment of the weights to minimize the error. Here's a detailed explanation of the backpropagation algorithm:
- ▶ **Key Concepts:**
 1. **Loss Function:** A function that measures the discrepancy between the predicted output of the network and the actual target values. Common loss functions include Mean Squared Error (MSE) for regression and Cross-Entropy Loss for classification.
 2. **Gradient Descent:** An optimization algorithm used to minimize the loss function by iteratively adjusting the weights. Variants include Stochastic Gradient Descent (SGD), Mini-batch Gradient Descent, and others.

Backpropagation

► Steps of Backpropagation:

1. Initialization:

1. Initialize weights and biases in the network, usually with small random values.

2. Forward Pass:

1. Input data is passed through the network layer by layer.
2. Each neuron computes a weighted sum of its inputs, applies an activation function, and passes the output to the next layer.
3. The final output is compared with the actual target to compute the loss.

3. Backward Pass (Backpropagation):

1. Compute the gradient of the loss function with respect to each weight by the chain rule, moving backward from the output layer to the input layer.

Backpropagation

► Steps in the backward pass:

- **Output Layer:**

- Compute the error at the output layer (difference between predicted and actual values).
- Compute the gradient of the loss function with respect to the outputs of the output layer neurons.
- Use the activation function's derivative to find the gradient with respect to the weighted input to the output neurons.

- **Hidden Layers:**

- For each layer, starting from the one before the output and moving backwards:
 - Compute the gradient of the loss with respect to the outputs of the neurons in this layer.
 - Use the derivative of the activation function to find the gradient with respect to the weighted input to these neurons.
 - Compute the gradient with respect to the weights connecting the neurons in this layer to the previous layer.

Backpropagation

Weight Update:

- Update the weights and biases using the gradients computed during the backward pass. The update rule for each weight w and bias b is typically:

$$w_{new} = w_{old} - \eta \frac{\partial L}{\partial w}$$


$$b_{new} = b_{old} - \eta \frac{\partial L}{\partial b}$$

where η is the learning rate, and $\frac{\partial L}{\partial w}$ and $\frac{\partial L}{\partial b}$ are the gradients of the loss function with respect to the weights and biases, respectively.

Example:

Consider a simple neural network with one hidden layer:

- **Forward Pass:**
 1. Compute activations for the hidden layer: $h = f(W_{ih} \cdot x + b_h)$
 2. Compute output: $y_{pred} = f(W_{ho} \cdot h + b_o)$
 3. Compute loss: $L = \text{loss}(y_{pred}, y_{true})$
- **Backward Pass:**
 1. Compute gradient of loss with respect to output weights and biases.
 2. Propagate the error back to compute gradients for hidden layer weights and biases.
 3. Update weights and biases.



Advantages:

- Efficiently computes gradients, making training feasible even for deep networks.
- Allows for the adjustment of all weights in the network simultaneously.

Disadvantages:

- Can get stuck in local minima or saddle points.
- Sensitive to the choice of hyperparameters (e.g., learning rate).
- Requires differentiable activation functions.

Backpropagation is a cornerstone of deep learning, enabling the training of complex models by leveraging the power of gradient descent optimization.

Batch normalization

- ▶ Batch normalization is a technique used in neural networks to improve training speed and stability. It normalizes the inputs of each layer by adjusting and scaling the activations. This is done by:
 1. **Normalizing:** Computing the mean and variance of the activations for each mini-batch and using these statistics to normalize the activations.
 2. **Scaling and Shifting:** Applying learned scaling and shifting parameters to the normalized activations.

This helps in reducing internal covariate shift, making the optimization process faster and more stable.

Optimization Algorithms

- ▶ An optimization algorithm is a method used to adjust the parameters of a model to minimize or maximize a particular objective function, often assigned as the loss or cost function. In the context of neural networks and machine learning, optimization algorithms are used to find the set of weights and biases that minimize the loss function, thereby improving the model's performance on a given task.
- ▶ **Key Aspects of Optimization Algorithms:**
 1. **Objective Function:** The function that needs to be minimized (e.g., mean squared error for regression, cross-entropy for classification).
 2. **Parameters:** The variables in the model that are adjusted to minimize the objective function (e.g., weights and biases in a neural network).

Optimization Algorithms

► Common Optimization Algorithms:

1. Gradient Descent:

1. **Batch Gradient Descent:** Computes the gradient using the entire dataset and updates the parameters in one large step. It can be slow and computationally expensive for large datasets.
2. **Stochastic Gradient Descent (SGD):** Computes the gradient and updates the parameters using one sample at a time. This introduces noise but can lead to faster convergence.
3. **Mini-batch Gradient Descent:** A compromise between batch and stochastic gradient descent, it computes the gradient using small subsets (mini-batches) of the dataset.

Optimization Algorithms

- ▶ **Common Optimization Algorithms:**
- ▶ **Adagrad:** Adapting the learning based on the historical gradient information, giving smaller updates for parameters associated with frequently occurring features.
- ▶ **RMSprop**(Root Mean Square Propagation): An extension of Adagrad that deals with its diminishing learning problem by using a moving average of squared gradients.
- ▶ **Adam (Adaptive Moment Estimation):** Combines the advantages of both momentum and RMSprop by keeping an exponentially decaying average of past gradients and squared gradients.