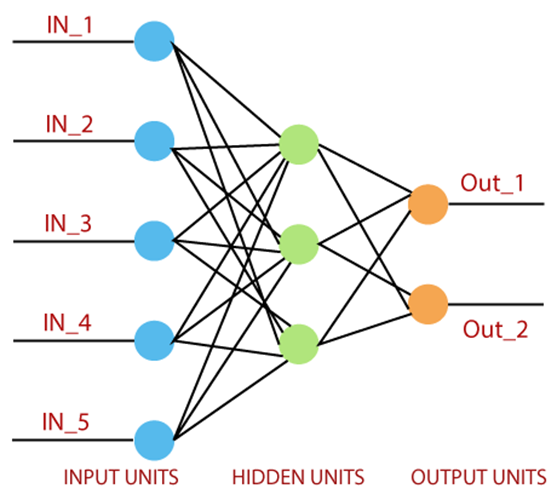
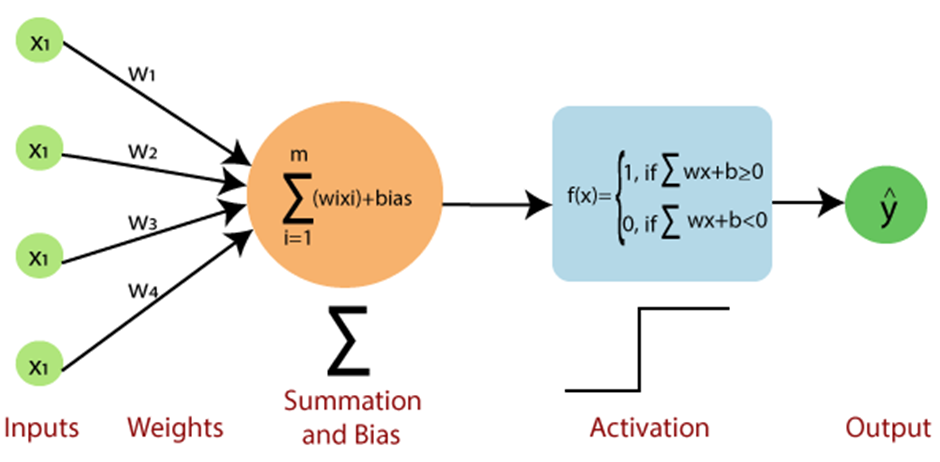
**Neural Network and Deep Learning Models**

**Overview**

* A neural network is formed when a collection of nodes or neurons are interlinked through synaptic connections
* There are three layers in every artificial neural network – input layer, hidden layer, and output layer
* The input layer that is formed from a collection of several nodes or neurons receives inputs
* Every neuron in the network has a function, and every connection has a weight value associated with it
* Inputs then move from the input layer to layer made from a separate set of neurons – the hidden layer
* The output layer gives the final outputs





**Input value or One input layer:** The input layer of the perceptron is made of artificial input neurons and takes the initial data into the system for further processing. Input features are denoted as x1, x2, x3…xn

X - feature value

n - total occurrences of these features

Along with this, one input type is there i.e bias.

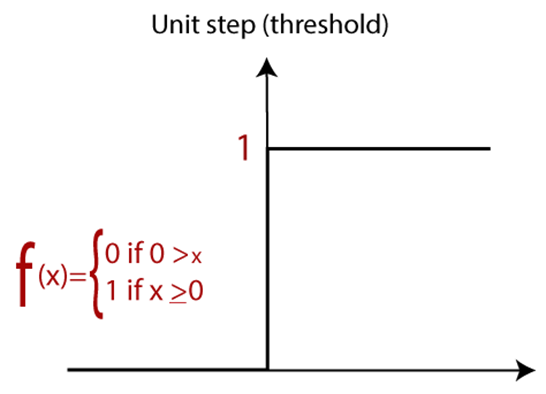
**Weights and Bias**

**Weight:** Weights are values that are calculated during the training of the model. It represents the dimension or strength of the connection between units. If the weight to node 1 to node 2 has a higher quantity, then neuron 1 has a more considerable influence on the neuron. With every occurrence of a training error, the values of weights are updated. Weights are represented as w1, w2,w3…wn.

**Bias:** It is the same as the intercept added in a linear equation. It is an additional parameter whose task is to modify the output along with the weighted sum of the input to the other neuron. It allows the classifier to move the decision boundary around from its original position to the right, left, upper, down. The objective of bias is to shift each point in a particular direction. **Decision boundary,** is a boundary or a dividing line that separates different classes or categories in a feature space.

**Net sum:** It calculates the total sum.

**Activation Function:** A neuron can be activated or not, it is determined by an activation function. The activation function calculates a weighted sum and further adds bias with it to give the result.



* The weights are initialized with the random values at the origination of each training.
* For each element of the training set, the error is calculated with the difference between the desired output and the actual output. The calculated error is used to adjust the weight.
* The process is repeated until the fault made on the entire training set is less than the specified limit until the maximum number of iterations has been reached.

**ANN - Artificial Neural Network Types**

**Feedforward Neural Network (FNN):**

* The most basic type of neural network where information travels in one direction—from the input layer to the output layer.
* No cycles or loops in the network architecture.

**Multilayer Perceptron (MLP):**

* A type of feedforward neural network with at least one hidden layer between the input and output layers.
* Suitable for a wide range of tasks, including classification and regression.

**Radial Basis Function Network (RBFN):**

* Employs radial basis functions as activation functions.
* Commonly used for function approximation and pattern recognition.

**Recurrent Neural Network (RNN):**

* Designed to work with sequential data.
* Contains connections that form cycles, allowing information to be stored in the network and used for tasks like natural language processing and time series prediction.

**Long Short-Term Memory (LSTM) Network:**

A specialized type of RNN designed to overcome the vanishing gradient problem, making it more effective for learning and remembering over long sequences.

**Convolutional Neural Network (CNN):**

* Primarily used for image-related tasks.
* Employs convolutional layers to automatically and adaptively learn spatial hierarchies of features from input images.

**Autoencoder:**

* A type of neural network designed for unsupervised learning.
* Consists of an encoder and a decoder, used for dimensionality reduction, feature learning, and generative tasks.

**Generative Adversarial Network (GAN):**

* Comprises a generator and a discriminator, trained simultaneously through adversarial training.
* Used for generating new data instances, such as images or text.

**Self-Organizing Map (SOM):**

* An unsupervised learning algorithm used for clustering and visualization.
* Particularly useful for exploring the underlying structure of high-dimensional data.

**RNN and CNN Architecture**

**Recurrent Neural Network (RNN)**

**Architecture:**

Sequential Processing: RNNs are designed to work with sequential data, where the order of elements matters. Examples include time series, text, and speech.

Recurrent Connections: RNNs have recurrent connections that allow information to be stored and passed from one step of the sequence to the next. This allows them to capture dependencies and relationships within sequential data.

**Key Components:**

Hidden States: At each time step, an RNN processes an input along with the hidden state from the previous time step, producing an output and updating the hidden state.

Long Short-Term Memory (LSTM): A specialized type of RNN that addresses the vanishing gradient problem. LSTMs have memory cells and gating mechanisms, enabling them to capture long-range dependencies in sequences.

**Use Cases:**

* Natural Language Processing (NLP): RNNs are commonly used for tasks such as language modeling, sentiment analysis, and machine translation.
* Time Series Prediction: Predicting future values in time series data.
* Challenges:
* Vanishing Gradient Problem: Standard RNNs can struggle to capture long-range dependencies due to the vanishing gradient problem, where gradients become very small during backpropagation through time.

**Convolutional Neural Network (CNN)**

**Architecture:**

* Spatial Hierarchies: CNNs are primarily designed for image-related tasks and excel at capturing spatial hierarchies of features.
* Convolutional Layers: CNNs use convolutional layers to automatically and adaptively learn patterns and features from input images. These layers include filters that slide over the input to detect local patterns.

**Key Components:**

* Convolutional Layers: These layers consist of filters that slide over the input to detect local patterns, allowing the network to learn hierarchical representations.
* Pooling Layers: Used to downsample the spatial dimensions of the input, reducing computational complexity.
* Fully Connected Layers: Typically follow the convolutional and pooling layers, acting as a classifier based on the learned features.

**Use Cases:**

* Image Classification: Identifying objects within images.
* Object Detection: Locating and classifying objects within an image.
* Image Segmentation: Assigning a label to each pixel in an image.

**Challenges:**

* Overfitting: CNNs can be prone to overfitting, especially when dealing with limited labeled data.
* Computational Intensity: Training deep CNNs can be computationally demanding, requiring significant resources.

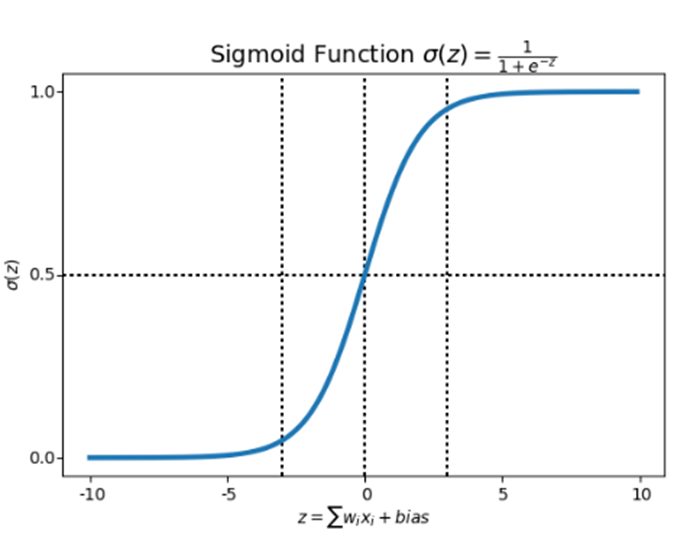
**Commonalities and Differences:**

* Commonalities: Both RNNs and CNNs are types of neural networks that have specific architectures tailored to different types of data.
* Differences: While RNNs are designed for sequential data, CNNs are optimized for grid-like data, such as images. Each has its strengths and weaknesses, and sometimes they are combined in hybrid architectures for tasks that involve both sequential and spatial dependencies, like video analysis.

**Types of Activation Functions**

**Sigmoid Activation Function**

The sigmoid is a mathematical function that maps input values to a value between 0 and 1, making it useful for binary classification and logistic regression problems. It is commonly used as an activation function in artificial neural networks, particularly in feedforward neural networks.



The formula for the sigmoid activation function (σ) is given by:



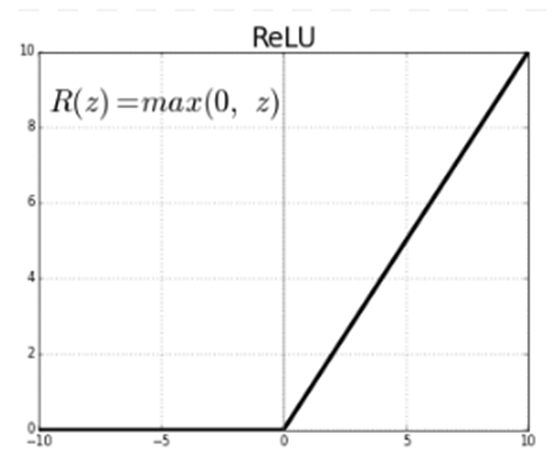
In this formula:

x is the input to the sigmoid function.

e is the base of the natural logarithm, approximately equal to 2.71828.

**ReLU Activation Function**

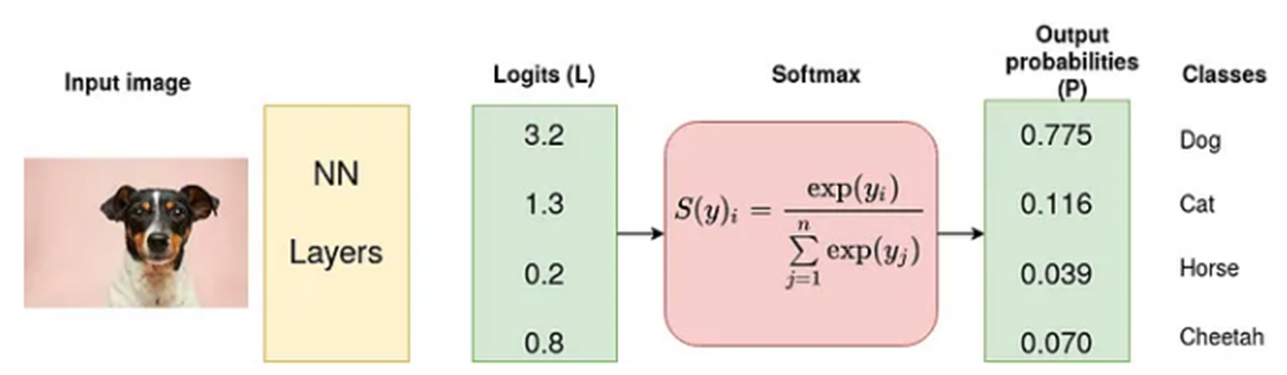
The rectified linear activation function or ReLU for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

****

f(x) = max(0,x) is the formula for reLU activation function.

**Softmax Activation Function**

Softmax is an activation function that scales numbers/logits into probabilities. The output of a Softmax is a vector (say v) with probabilities of each possible outcome. The probabilities in vector v sums to one for all possible outcomes or classes.

****

Given an input vector z = z1, z2, z3…zk, the softmax function 𝞼(z) produces an output vector 𝞼(z).= (𝞼(z1), 𝞼(z2), 𝞼(z3)..... 𝞼(zk)), where:



zi is the ith element of the input vector z

e is the base of the natural logarithm, approximately equal to 2.71828.

**Types of Optimizers**

Optimizers plays a crucial role in training models. It adjusts the model parameters during training to minimize the loss function. The choice of optimizer can impact the convergence speed and the final performance of the model.

**Adam Optimizer**

Adaptive Moment Estimation (Adam) ia popular optimization algorithm that combines ideas from two other optimizers i.e. RMSprop and Momentum. When training a model, the objective is to minimize a loss function by adjusting the model’s parameters. This process involves finding the optimal set of parameters that results in the lowest possible loss. Optimizers automate this process by iteratively adjusting the parameters based on the gradients of the loss function with respect to those parameters.

**RMSprop (Root Mean Square Propogation)** adjusts the learning rate for each parameter individually based on the historical gradients. It helps to handle different scales of features.

**Momentum** helps optimizers to keep moving in the right direction even when there are fluctuations in the gradient. It introduces a velocity term, and the optimizer gains momentum as it moves down the gradient.

**Types of Loss Functions**

**Binary Cross Entropy**

It is also known as logrithmic loss or logistic loss, it’s a loss function used for binary classification problems. It measures the difference between the true labels and the predicted probabilities assigned to those labels.

L(y,Y) = -[y.log(Y) + (1 - y).log(1 - Y)]

L(y,Y) - is the binary cross entropy loss

y - is the true label (either 0 or 1)

Y - is the predicted probability of the instance belonging to class 1

**Sparse Categorical Cross Entropy**

It is one of the loss functions commonly used in multiclass classification. This loss function is used when, each input belongs to exactly one class.

SCC(y,p) = -1/N ∑(i=1 to N) log(pi,yi)

y - True class labels

p - Predicted probabilities for each other

N - no.of samples

pi,yi - probability assigned to the true class yi, for the ith sample

SCC - is a suitable choice when dealing with integer class labels

**Deep Learning Algorithm**

**YOLO**

YOLO (You Only Look Once) is a popular object detection algorithm in deep learning. It's known for its real-time object detection capabilities and has been widely used in various applications, including computer vision, robotics, and autonomous vehicles. YOLO differs from traditional object detection algorithms by framing object detection as a regression problem, predicting bounding box coordinates and class probabilities directly from the input image.

**Here are key features and components of the YOLO algorithm:**

**Single Forward Pass**

* YOLO performs object detection in a single forward pass through the neural network, making it computationally efficient.
* The entire image is divided into a grid, and predictions are made for each grid cell.

**Grid Division**

The image is divided into an S×S grid. Each grid cell is responsible for predicting bounding boxes and class probabilities for objects that fall within it.

**Bounding Box Prediction**

* Each grid cell predicts multiple bounding boxes along with associated confidence scores.
* Each bounding box is defined by its center coordinates (x,y), width(w), height(h), and a confidence score (P(object) \* IoU), where IoU is the intersection over union.

**Class Prediction**

For each bounding box, YOLO predicts class probabilities for the detected object.

Class probabilities are typically computed using softmax activation.

**Confidence Thresholding**

During post-processing, a confidence threshold is applied to filter out bounding boxes with low confidence scores.

**Non-Maximum Suppression (NMS)**

To eliminate duplicate detections, YOLO employs non-maximum suppression.

Bounding boxes with high overlap are suppressed, retaining only the one with the highest confidence score.

**Architecture Variants**

YOLO has several versions, including YOLOv1, YOLOv2 (YOLO9000), YOLOv3, and YOLOv4. Each version introduces improvements in terms of accuracy, speed, or capabilities.

**Real-Time Object Detection**

YOLO is known for its real-time performance, making it suitable for applications that require low-latency object detection.

YOLO has been influential in advancing the field of object detection, and its variants have been widely adopted. YOLOv4, for example, introduced improvements in accuracy and speed over previous versions.