***Loss Function***: -

The loss function is the function that computes the difference (or distance) between the current output of the algorithm and the target output. In other words, they are used to determine the error between the current output of the algorithm and the given target value.  
They are divided into two categories, i.e., classification and regression.

Classification: -

1. *Cross-entropy Loss*: - The Cross-entropy loss measures the difference between two averages of number of bits of distribution of information. It is also called as the logarithmic loss, log loss or the logistic loss. It can be used with binary and multiclass classification problems. The different types of cross-entropy are binary cross-entropy, categorial cross-entropy and sparse cross-entropy.
2. *Hinge Loss*: - The Hinge loss function was developed to correct the hyperplane of SVM algorithm in the task of classification. The goal is to make different penalties at the point that are not correctly predicted or too closed of the hyperplane. Its mathematical expression is given by,

Hinge = max(0, 1 - y \* f(X))

1. *Kullback Leibler Divergence Loss*: - The KL divergence is the score of two different probability distribution functions. It is also known as relative entropy and tells us how one probability distribution is different from another. The difference between probability distribution functions f(X) and g(x) can be written as,

KL(G||F) = - sum( g(x) \* log(f(x) / g(x)) or sum(g(x) \* log(g(x) / f(x))

Regression: -

1. *Mean Square Error Loss*: - MSE is also known as L2 Regularization and it is the square difference between the current output and the expected output. The MSE function is very sensitive to outliers because the difference is a square that gives more importance to outliers. The behaviour(plot) of MSE is a quadratic curve, which is useful for gradient descent algorithm. The mathematical expression is given by,

MSE = ∑(y - Y) ^ 2 / n

1. *Mean Absolute Error Loss*: - MAE is also known as L1 Regularization. It is mostly like MSE, just the square is replaced by absolute. The behaviour of MAE is a "V" form. The MAE function is more robust to outliers because it is based on absolute value compared to the square of the MSE. The mathematical expression is given by,

MAE = ∑|y - Y| / n

1. *Huber Loss*: - Huber Loss is a combination of L1(MAE) and L2(MSE) regularization, though it depends on an additional parameter delta which influences the shape of the loss function. When the values are large (far from the minima), the function has the behaviour of the MAE and when values are small (close to the minima), the function behaves like the MSE. So, the delta parameter is the sensitivity to outliers. But this function needs fine-tuning delta but it’s computationally expensive. The mathematical expression is given by,

L(y, f(x)) = 0.5 \* (y - f(x)) ^ 2, for |y - f(x)| <= δ or δ|y - f(x)| - 0.5 \* δ ^ 2

***Optimizers***: -

Optimizers are algorithms or methods used to minimize a loss function or to maximize the efficiency of a neural network. They are mathematical functions which are dependent on model’s learnable parameters, i.e., weights & biases.

Types of Optimizers: -

1. Gradient Descent: - GD is the most common optimizer used in neural networks. The weights are updated when the whole dataset gradient is calculated, if there is a huge amount of data weights updating takes more time and more memory space, which increases the computational time. In some cases, problems like Vanishing Gradient or Exploding Gradient may also occur due to incorrect parameter initialization.
2. Stochastic Gradient Descent: - SGD is a variant of GD. It updates the model parameters one by one, i.e., if the model has 10K dataset SGD will update the model parameters 10k times. It is faster than GD, and less computationally expensive. The oscillation of SGD may jump to a better local minimum.
3. Mini-Batch Gradient Descent: - MBGD is a combination of the concepts of SGD and batch gradient descent. It simply splits the training dataset into small batches and performs an update for each of those batches, which creates a balance between the robustness of stochastic gradient descent and the efficiency of batch gradient descent. This results in less memory usage and low variance in the model.
4. SGD with Momentum: - SGD with Momentum is a stochastic optimization method that adds a momentum term to regular SGD. In this method, the previous update is retained to a certain extent during the update, while the current update gradient is used to fine-tune the final update direction. It is a method which helps accelerate gradients vectors in the right directions, thus leading to faster converging and reduce the noise.
5. Adaptive Gradient Descent: - AdaGrad is a modified SGD with different learning rates with each parameter based on iteration. Learning rate is an important hyper-parameter varying which can change the pace of training. After each iteration the learning rate decreases gradually, changing to a new weight. This in turn causes the learning rate to shrink and eventually become very small, where the algorithm is not able to acquire any further knowledge.
6. Root Mean Square Propagation / Adadelta: - RMS-Prop or Adadelta is an extension of Adagrad that attempts to solve its radically diminishing learning rates. The idea behind Adadelta is that instead of summing up all the past squared gradients from 1 to “t” time steps, it restricts the window size. Adadelta basically combines momentum with AdaGrad.
7. Adaptive Moment Estimation: - Adam is a method that computes adaptive learning rates for each parameter. It stores both the decaying average of the past gradients, similar to momentum and also the decaying average of the past squared gradients, similar to RMS-Prop and Adadelta. Thus, it combines the advantages of both the methods. It is the most commonly used optimizer. It has many benefits like low memory requirements, works best with large data and parameters with efficient computation.