**TENSORFLOW:**

* TensorFlow is an open-source machine learning framework developed by Google Brain for building and training machine learning models.
* It is an end-to-end open source framework for deep learning that wors with python.
* It was especially build for python but can be used for other languages like c#,R,rust etc.



* Flow of a tensor is the fundamental working of a neural network.

**Tensor**:

* Tensors are the primary objects used for storing and transforming data, such as input features, model parameters, and intermediate activations during the training process.
* Tensors can have different ranks, shapes, and data types, allowing them to store and manipulate various types of data efficiently.

**Working of TensorFlow:**

* TensorFlow enables developers to design dataflow graphs.
* Dataflow graphs define how data flows via a graph or set of processing nodes.
* Each node in the graph represents a mathematical process, and each edge between nodes is a tensor, a multi-layered data array.
* TensorFlow applications can execute on various targets, including local PCs, cloud clusters, iOS and Android phones, CPUs, and GPUs.
* Google's cloud infrastructure allows running TensorFlow on TensorFlow Processing Unit (TPU) hardware for additional acceleration**.**

**Architecture of TensorFlow:**

Tensorflow’s architecture consists of three components:

* Data processing in advance
* Creating the model
* Developing and evaluating the model

**TensorFlow Components:**

**Tensor:**

* Tensors are the fundamental data structure in TensorFlow.
* They represent multi-dimensional arrays or matrices that hold various types of data.
* Tensors have a known or partially known shape, representing the dimensionality of the data.
* All computations in TensorFlow involve tensors, which can originate from input data or as the result of operations within a graph.

**Graphs:**

* TensorFlow uses a graph framework to manage and describe computations.
* A computational graph consists of nodes (ops) representing operations and edges representing tensors (data).
* Graphs enable TensorFlow to run computations on multiple devices, including CPUs, GPUs, and mobile operating systems.
* The portability of graphs allows computations to be preserved for immediate or future use, as they can be saved and executed later.
* Tensors flow through the graph, connecting nodes and carrying data between operations

**TensorFlow Algorithms:**

* Linear regression: tf.estimator.LinearRegressor
* Classification:tf.estimator.LinearClassifier
* Deep learning classification: tf.estimator.DNNClassifier
* Deep learning wipe and deep: tf.estimator.DNNLinearCombinedClassifier
* Booster tree regression: tf.estimator.BoostedTreesRegressor
* Boosted tree classification: tf.estimator.BoostedTreesClassifier

**Tensorflow Operations:**

1. Dense Layer:

* Fully connected layer that performs matrix multiplication followed by an activation function.
* Implemented using “tf.keras.layers.Dense”.

2. Convolutional Layer:

* Performs convolution operation to extract features from input data.
* Implemented using ”tf.keras.layers.Conv2D” for 2D convolution and “tf.keras.layers.Conv1D” for 1D convolution.

3. Pooling Layer:

* Reduces the spatial dimensions of the input data by summarizing regions.
* Implemented using “tf.keras.layers.MaxPooling2D” for max pooling and “tf.keras.layers.AveragePooling2D” for average pooling.

4. Dropout Layer:

* Regularization technique that randomly drops a fraction of units during training to prevent overfitting.
* Implemented using `tf.keras.layers.Dropout`.

5. Batch Normalization:

* Normalizes the activations of the previous layer to speed up training and increase stability.
* Implemented using `tf.keras.layers.BatchNormalization`.

6. Activation Function:

* Introduces non-linearity to the model, enabling it to learn complex patterns.

7. Loss Function:

* Measures the difference between the predicted output and the actual target.
* Common loss functions include Mean Squared Error (`tf.keras.losses.mean\_squared\_error`), Categorical Crossentropy (`tf.keras.losses.categorical\_crossentropy`), and Binary Crossentropy (`tf.keras.losses.binary\_crossentropy`).

8. Optimizer:

* Updates the model parameters during training to minimize the loss function.
* Common optimizers include Adam (`tf.keras.optimizers.Adam`), SGD (`tf.keras.optimizers.SGD`), and RMSprop (`tf.keras.optimizers.RMSprop`).

9. Model Compilation:

* Compiles the neural network model with a specific loss function, optimizer, and evaluation metrics.
* Implemented using `model.compile()` function in Keras.