TENSORFLOW

*What is Tenssorflow..?

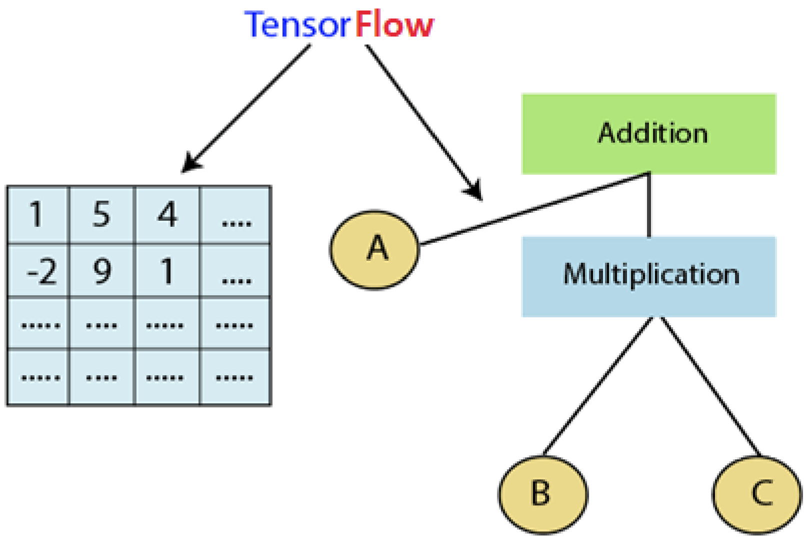
TensorFlow is a popular framework of **machine learning** and **deep learning**. It is **a free** and **open-source** library which is released on **9 November 2015** and developed by **Google Brain Team**. It is entirely based on Python programming language and use for numerical computation and data flow, which makes machine learning faster and easier.

TensorFlow can train and run the deep neural networks for image recognition, handwritten digit classification, recurrent neural network, **word embedding**, **natural language processing**, video detection, and many more. TensorFlow is run on multiple **CPU**s or **GPU**s and also mobile operating systems.

The word TensorFlow is made by two words, i.e., Tensor and Flow

1. **Tensor** is a multidimensional array
2. **Flow** is used to define the flow of data in operation.

TensorFlow is used to define the flow of data in operation on a multidimensional array or Tensor.



History of TensorFlow

Many years ago, deep learning started to exceed all other machine learning algorithms when giving extensive data. Google has seen it could use these deep neural networks to upgrade its services:

- Google search engine
- Gmail
- Photo

They build a framework called TensorFlow to permit researchers and developers to work together in an **AI** model. Once it approved and scaled, it allows lots of people to use it.

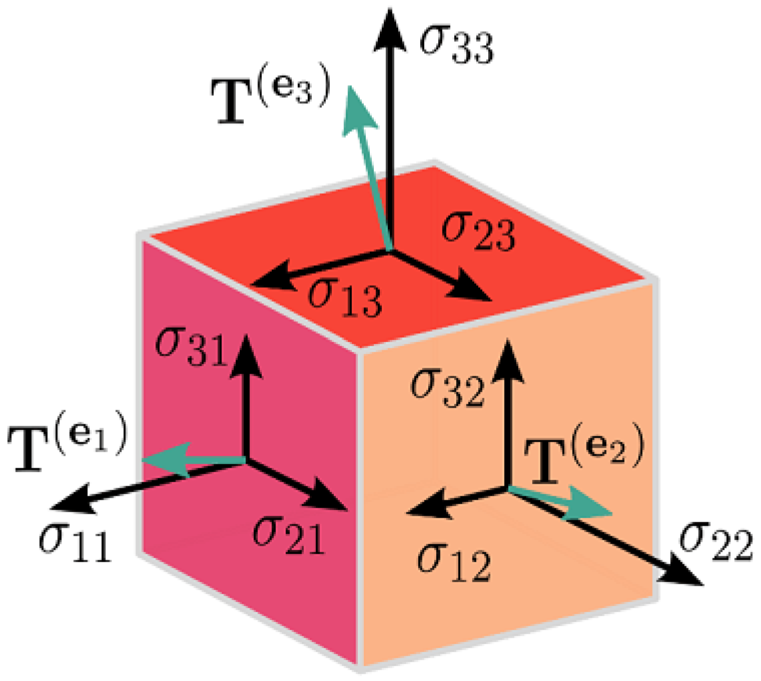
It was first released in 2015, while the first stable version was coming in **2017**. It is an open- source platform under Apache Open Source License. We can use it, modify it, and reorganize the revised version for free without paying anything to Google.

Components of TensorFlow

Tensor

The name TensorFlow is derived from its core framework, "**Tensor**." A tensor is a vector or a matrix of n-dimensional that represents all type of data. All values in a tensor hold similar data type with a known shape. The shape of the data is the dimension of the matrix or an array.

A tensor can be generated from the input data or the result of a computation. In TensorFlow, all operations are conducted inside a graph. The group is a set of calculation that takes place successively. Each transaction is called an op node are connected.



Graphs

TensorFlow makes use of a graph framework. The chart gathers and describes all the computations done during the training.

Advantages

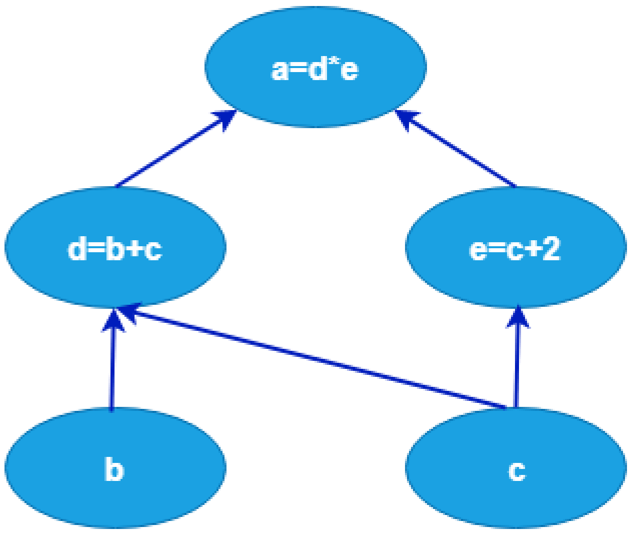
- It was fixed to run on multiple CPUs or GPUs and mobile operating systems.
- The portability of the graph allows to conserve the computations for current or later use. The graph can be saved because it can be executed in the future.
- All the computation in the graph is done by connecting tensors together.

Consider the following expression a= (b+c)*(c+2)

We can break the functions into components given below:

d=b+c
e=c+2
a=d*e

Now, we can represent these operations graphically below:



Session

A session can execute the operation from the graph. To feed the graph with the value of a tensor, we need to open a session. Inside a session, we must run an operator to create an output.

Why is TensorFlow popular?

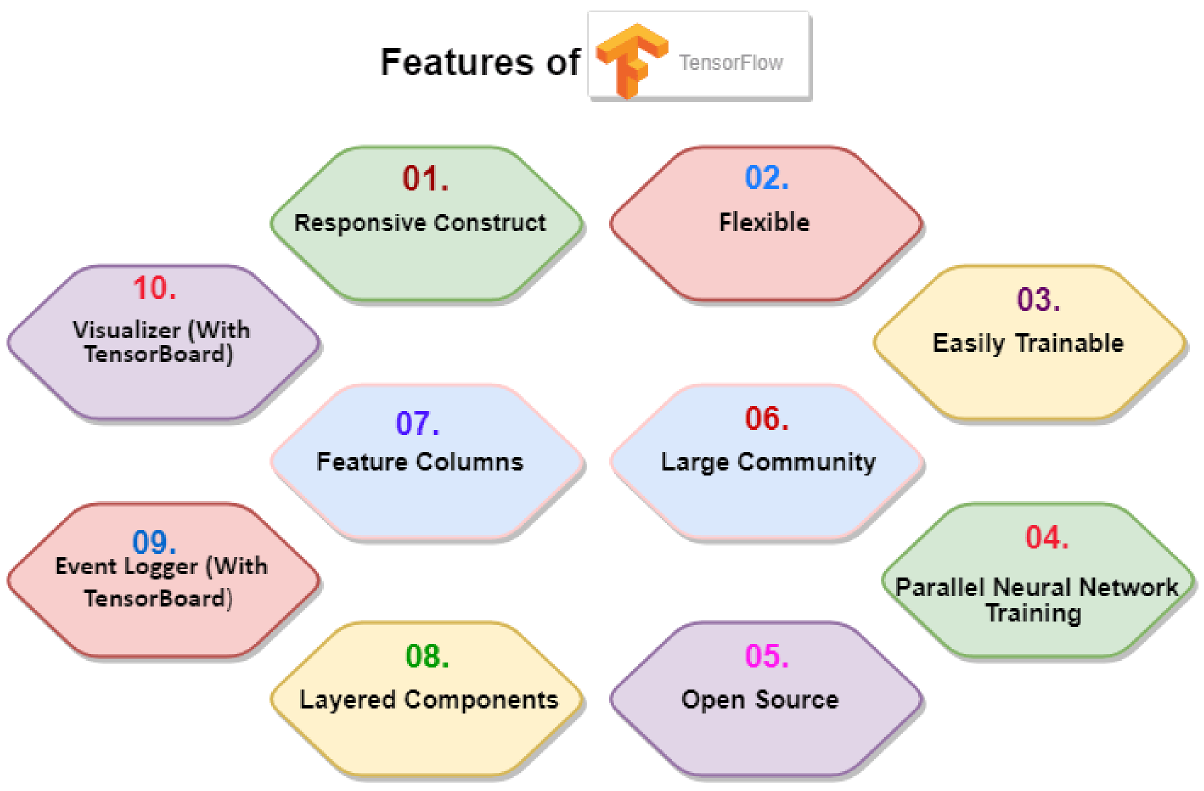
TensorFlow is the better library for all because it is accessible to everyone. TensorFlow library integrates different **API** to create a scale deep learning architecture like **CNN (Convolutional Neural Network)** or **RNN (Recurrent Neural Network)**.

TensorFlow is based on graph computation; it can allow the developer to create the construction of the neural network with Tensorboard. This tool helps debug our program. It runs on CPU (Central Processing Unit) and GPU (Graphical Processing Unit).

Features of TensorFlow

TensorFlow has an interactive **multiplatform** programming interface which is scalable and reliable compared to other deep learning libraries which are available.

These features of TensorFlow will tell us about the popularity of TensorFlow.



1. Responsive Construct

We can visualize each part of the graph, which is not an option while using **Numpy** or **SciKit**. To develop a deep learning application, firstly, there are two or three components that are required to create a deep learning application and need a programming language.

2. Flexible

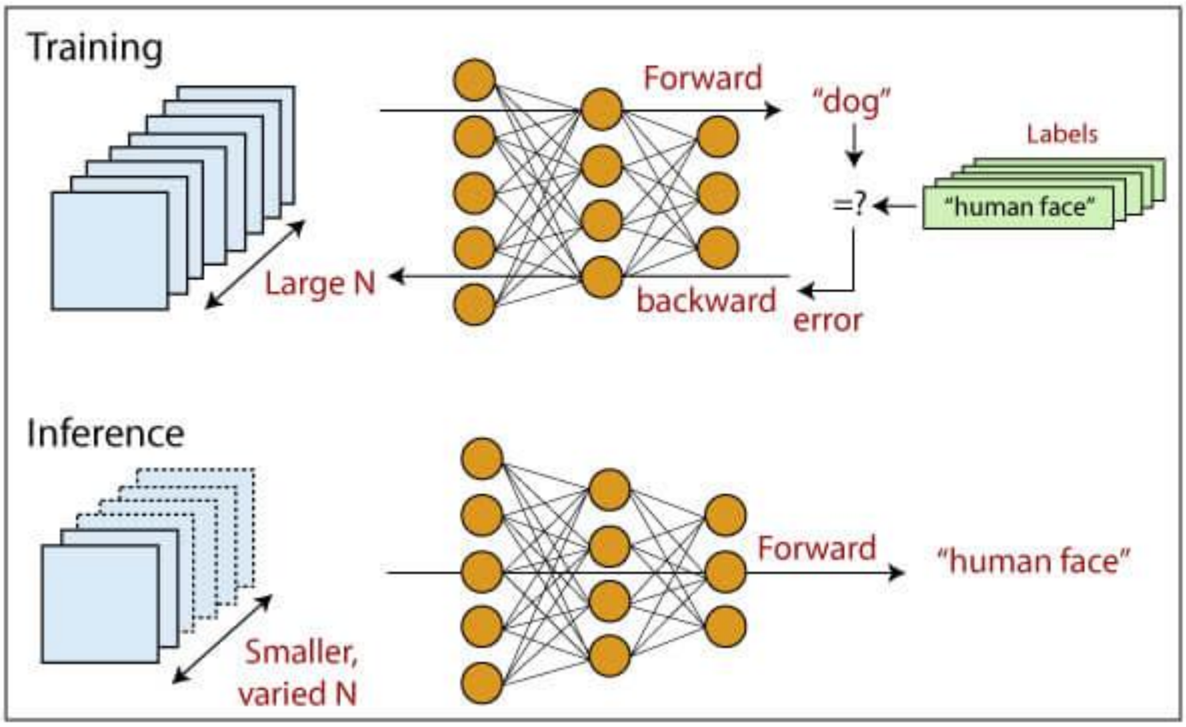
It is one of the essential TensorFlow Features according to its operability. It has modularity and parts of it which we want to make standalone.

3. Easily Trainable

It is easily trainable on CPU and for GPU in distributed computing.

4. Parallel Neural Network Training

TensorFlow offers to the pipeline in the sense that we can train multiple neural networks and various **GPUs**, which makes the models very efficient on large-scale systems.



5. Large Community

Google has developed it, and there already is a large team of software engineers who work on stability improvements continuously.

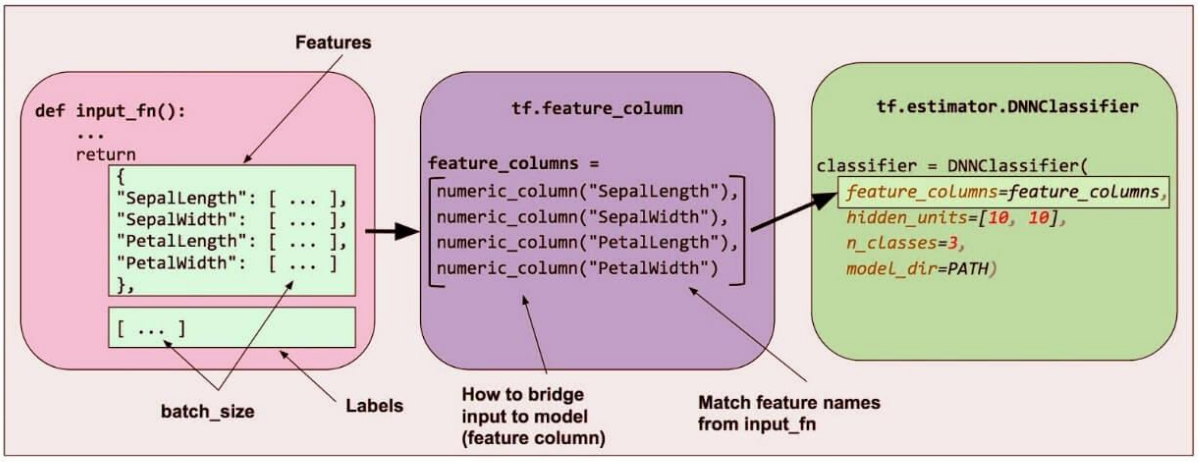
6. Open Source

The best thing about the machine learning library is that it is open source so anyone can use it as much as they have internet connectivity. So, people can manipulate the library and come up with a fantastic variety of useful products. And it has become another **DIY** community which has a massive forum for people getting started with it and those who find it hard to use it.

7. Feature Columns

TensorFlow has feature columns which could be thought of as intermediates between raw data and estimators; accordingly, **bridging** input data with our model.

The feature below describes how the feature column is implemented.



```
def input_fn():
    ...
    return
      {
      "SepalLength": [ ... ],
      "SepalWidth":  [ ... ],
      "PetalLength": [ ... ],
      "PetalWidth":  [ ... ]
      },
      [ ... ]
```

```
feature_columns =
[numeric_column("SepalLength"),
 numeric_column("SepalWidth"),
 numeric_column("PetalLength"),
 numeric_column("PetalWidth")]
```

```
classifier = DNNClassifier(
   feature_columns=feature_columns,
   hidden_units=[10, 10],
   n_classes=3,
   model_dir=PATH)
```

Features
tf.feature_column
tf.estimator.DNNClassifier
batch_size
Labels
How to bridge input to model (feature column)
Match feature names from input_fn

8. Availability of Statistical Distributions

This library provides distributions functions including Bernoulli, Beta, Chi2, Uniform, Gamma, which are essential, especially where considering probabilistic approaches such as Bayesian models.

9. Layered Components

TensorFlow produces layered operations of weight and biases from the function such as tf.contrib.layers and also provides batch normalization, convolution layer, and dropout layer. So **tf.contrib.layers.optimizers** have optimizers such as **Adagrad**, **SGD**, **Momentum** which are often used to solve optimization problems for numerical analysis.

10. Visualizer (With TensorBoard)

We can inspect a different representation of a model and make the changed necessary while debugging it with the help of TensorBoard.

11.Event Logger (With TensorBoard)

It is just like UNIX, where we use **tail - f** to monitor the output of tasks at the cmd. It checks, logging events and summaries from the graph and production with the TensorBoard.

INSTALL PKG

Conda create -n tensorflow pip python.

Architecture of TensorFlow

The TensorFlow runtime is a cross-platform library. The system architecture which makes this combination of scale flexible. We have basic familiarity with TensorFlow programming concepts such as the computation graph, operations, and sessions.

Some terms need to be understood first to understand TensorFlow architecture. The terms are TensorFlow Servable, servable Streams, TensorFlow Models, Loaders, Sources, Manager, and Core. The term and their functionality in the architecture of TensorFlow are described below.

TensorFlow architecture is appropriate to read and modify the core TensorFlow code.

1. TensorFlow Servable

These are the central uncompleted units in TensorFlow serving. Servables are the objects that the clients use to perform the computation.

The size of a servable is flexible. A single servable may consist of anything from a lookup table to a unique model in a tuple of interface models. Servable should be of any type and interface, which enabling flexibility and future improvements such as:

- Streaming results
- Asynchronous modes of operation.
- Experimental APIs

2. Servable Versions

TensorFlow server can handle one or more versions of the servables, over the lifetime of any single server instance. It opens the door for new algorithm configurations, weights, and other data can be loaded over time. They also can enable more than one version of a servable to be charged at a time. They also allow more than one version of a servable to be loaded concurrently, supporting roll-out and experimentation gradually.

3. Servable Streams

A sequence of versions of any servable sorted by increasing version of numbers.

4. TensorFlow Models

A serving represents a model in one or more servables. A machine-learned model includes one or more algorithm and lookup the embedding tables. A servable can also serve like a fraction of a model; for example, an example, a large lookup table be served as many instances.

5. TensorFlow Loaders

Loaders manage a servable's life cycle. The loader API enables common infrastructure which is independent of the specific learning algorithm, data, or product use-cases involved.

6. Sources in TensorFlow Architecture

In simple terms, sources are modules that find and provide servable. Each reference provides zero or more servable streams at a time. For each servable stream, a source supplies only one loader instance for every servable.

Each source also provides zero or more servable streams. For each servable stream, a source supplies only one loader instance and makes available to be loaded.

7. TensorFlow Managers

TensorFlow managers handle the full lifecycle of a Servables, including:

- Loading Servables
- Serving Servables
- Unloading Servables

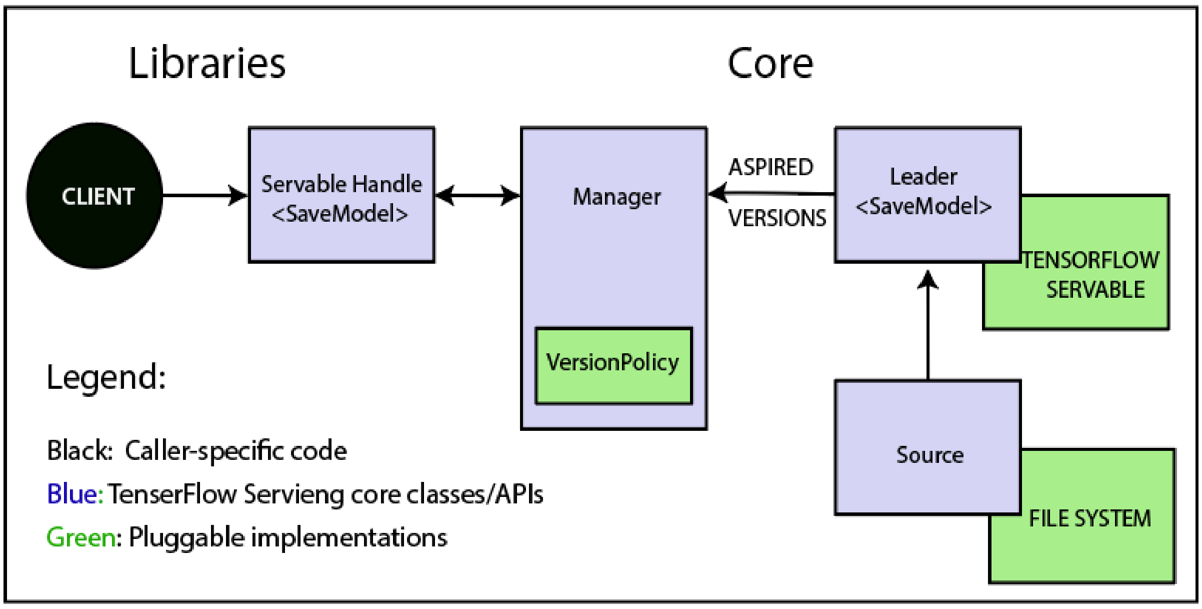
Manager observes to sources and tracks all versions. The Manager tries to fulfill causes, but it can refuse to load an Aspired version.

Managers may also postpone an "**unload**." For example, a manager can wait to unload as far as a newer version completes loading, based on a policy to assure that at least one version is loaded all the times.

For example, GetServableHandle (), for clients to access the loaded servable instances.

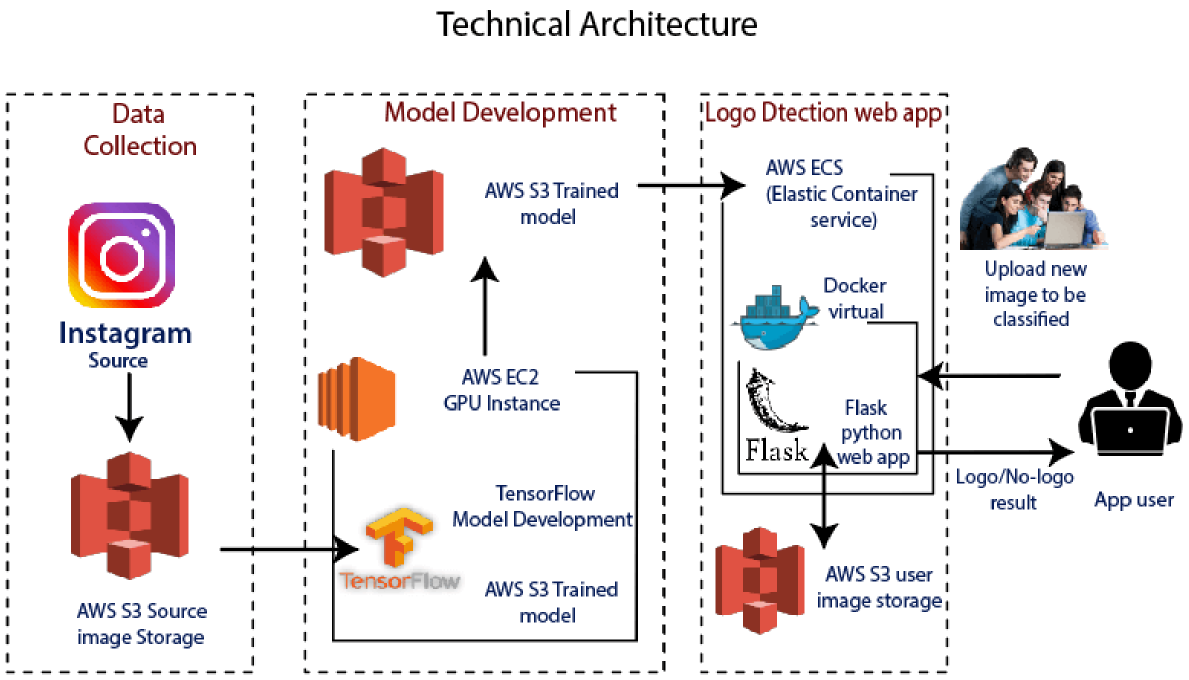
8. TensorFlow Core

This manages the below aspects of servables:


Libraries
Core
CLIENT
Servable Handle <SaveModel>
Manager
ASPIRED
VERSIONS
Leader <SaveModel>
TENSORFLOW SERVABLE
VersionPolicy
Source
FILE SYSTEM
Legend:
Black: Caller-specific code
Blue: TenserFlow Servieng core classes/APIs
Green: Pluggable implementations

- Lifecycle
- Metrics
- TensorFlow serving core satisfaction servables and loaders as the opaque objects.

9. Life of a Servable



TensorFlow Technical Architecture:

- Sources create loaders for Servable Versions, and then loaders are sent as Aspired versions to the Manager, which will load and serve them to client requests.

- The Loader contains metadata, and it needs to load the servable.
- The source uses a callback to convey the Manager of Aspired version.
- The Manager applies the effective version policy to determine the next action to take.
- If the Manager determines that it gives the Loader to load a new version, clients ask the Manager for the servable, and specifying a version explicitly or requesting the current version. The Manager returns a handle for servable. The dynamic Manager applies the version action and decides to load the newer version of it.
- The dynamic Manager commands the Loader that there is enough memory.
- A client requests a handle for the latest version of the model, and dynamic Manager returns a handle to the new version of servable.

10. TensorFlow Loaders

TensorFlow is one such algorithm backend. For example, we will implement a new loader to load, provide access, and unload an instance of a new type of servable of the machine learning model.

11. Batcher in TensorFlow Architecture

Batching of TensorFlow requests into a single application can significantly reduce the cost f performing inference, especially in the presence of hardware accelerators and GPUs. TensorFlow serving has a claim batching device that approves clients to batch their type-specific assumption beyond request into batch quickly. And request that algorithm systems can process more efficiently.

Advantage and Disadvantage of TensorFlow

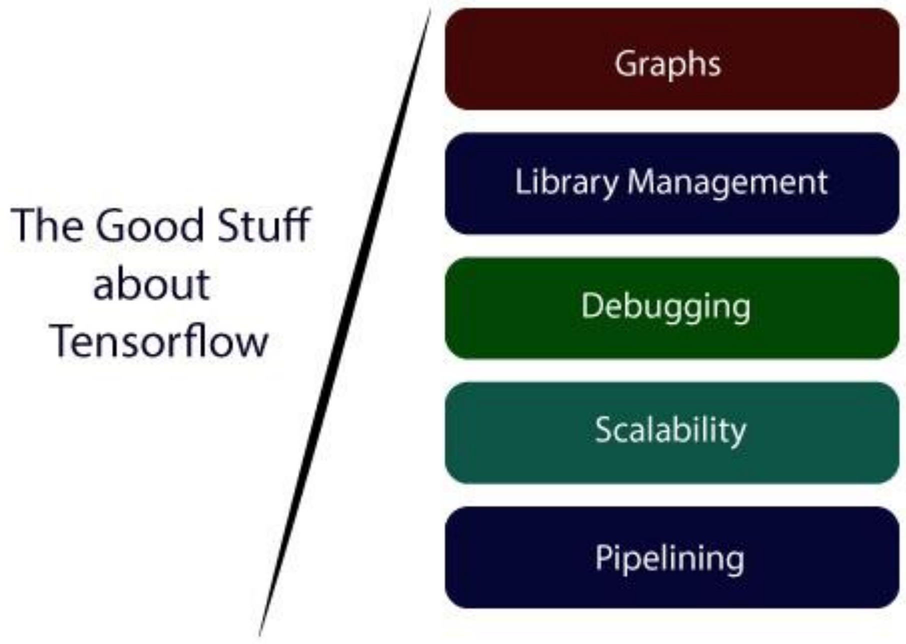
TensorFlow is an open-source machine learning concept which is designed and developed by Google. It offers a very high level and abstract approach to organizing low-level numerical programming. And supporting libraries that can allow our software to run without changes on regular CPU.

It supported platforms include **Linux**, **macOS**, **Windows**, and **Android**.

TensorFlow models can also be run without a traditional computer platform in the Google Cloud Machine Learning Engine.

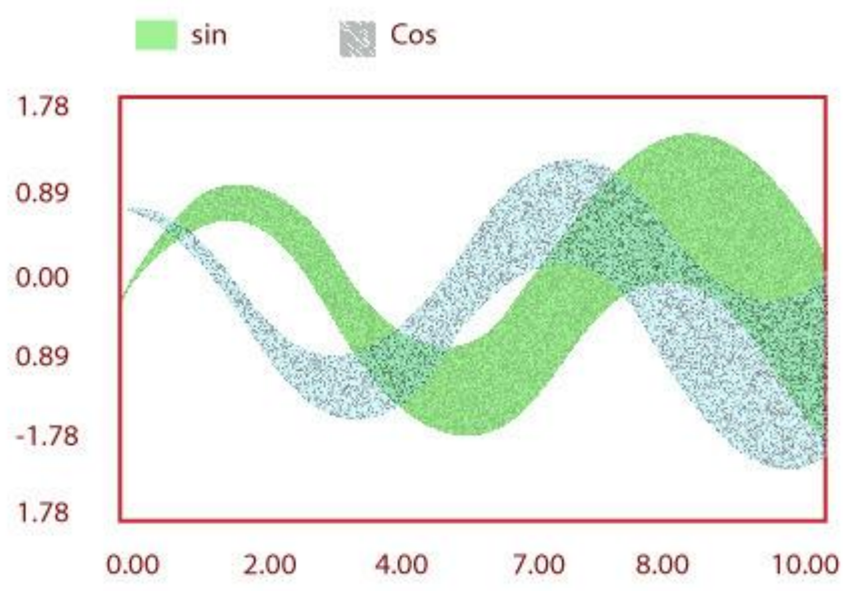
The more advanced technology, and the more useful it can be, but everything has its downside and also this machine learning library. When comparing TensorFlow with other libraries like **Torch**, **SciKit**, **Theano**, **Neon**, there are drawbacks in several features that the library lets us manipulate. This library is designed and updated by Google, so needless to say, and it has come a far way since its initial release.

Advantages of TensorFlow



1) Graphs:

TensorFlow has better computational graph visualizations. Which are inherent when compared to other libraries like Torch and Theano.



2) Library management:

Google backs it. And has the advantages of seamless performance, quick updates, and frequent new releases with new features.

3) Debugging:

It helps us execute subpart of a graph which gives it an upper hand as we can introduce and retrieve discrete data

4) Scalability:

The libraries are deployed on a hardware machine, which is a cellular device to the computer with a complex setup.

5) Pipelining:

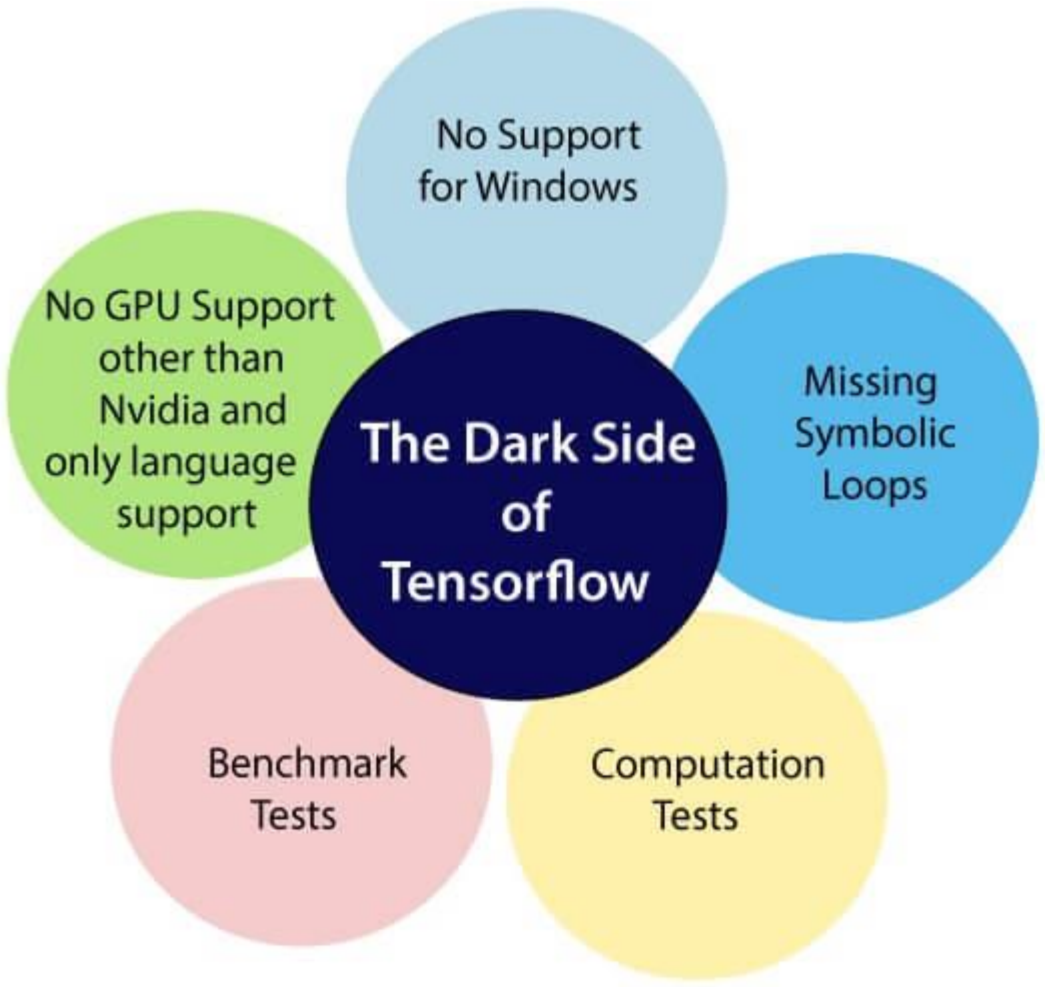
TensorFlow is designed to use various backend software (GPUs, ASIC), etc. and also highly parallel.

6) It has a unique approach that allows monitoring the training progress of our models and tracking several metrics.

7) TensorFlow has excellent community support.

8) Its performance is high and matching the best in the industry.

Disadvantages of TensorFlow


No Support for Windows
No GPU Support other than Nvidia and only language support
The Dark Side of Tensorflow
Missing Symbolic Loops
Benchmark Tests
Computation Tests

1) Missing Symbolic loops:

When we say about the variable-length sequence, the feature is more required. Unfortunately, TensorFlow does not offer functionality, but finite folding is the right solution to it.

2) No supports for windows:

There is a wide variety of users who are comfortable in a window environment rather than Linux, and TensorFlow doesn't satisfy these users. But we need not worry about that if we are a window user we can also install it through conda or python package library (pip).

3) Benchmark tests:

TensorFlow lacks in both speed and usage when it is compared to its competitors.

4) No GPU support for Nvidia and only language support:

Currently, the single supported GPUs are **NVIDIA** and the only full language support of Python, which makes it a drawback as there is a hike of other languages in deep learning as well as the **Lau**.

5) Computation Speed:

This is a field where TF is delaying behind, but we focus on the production environment ratherish than the performance, it is still the right choice.

6) No support for OpenCL.

7) It requires fundamental knowledge of advanced calculus and **linear algebra** along with a good understanding of **machine learning** also.

8) TensorFlow has a unique structure, so it's hard to find an error and difficult to debug.

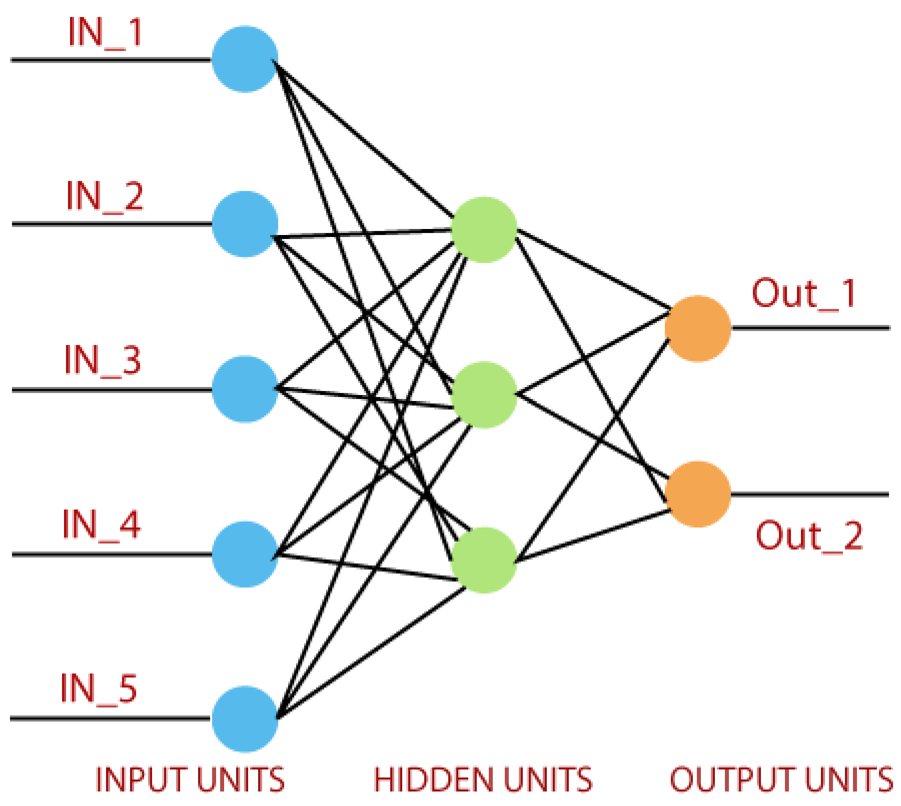
9) There is no need for any super low-level matter.

10) It is a very low level with a steep learning curve.

Single Layer Perceptron in TensorFlow

The perceptron is a single processing unit of any neural network. **Frank Rosenblatt** first proposed in **1958** is a simple neuron which is used to classify its input into one or two categories. Perceptron is a linear classifier, and is used in supervised learning. It helps to organize the given input data.

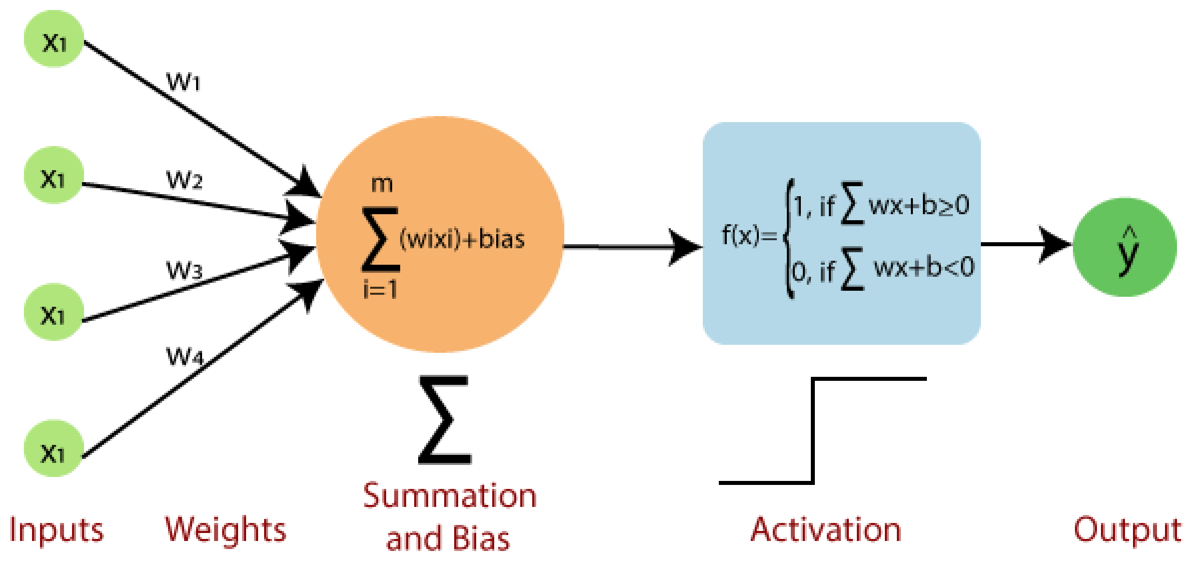
A perceptron is a neural network unit that does a precise computation to detect features in the input data. Perceptron is mainly used to classify the data into two parts. Therefore, it is also known as **Linear Binary Classifier**.



Perceptron uses the step function that returns +1 if the weighted sum of its input 0 and -1.

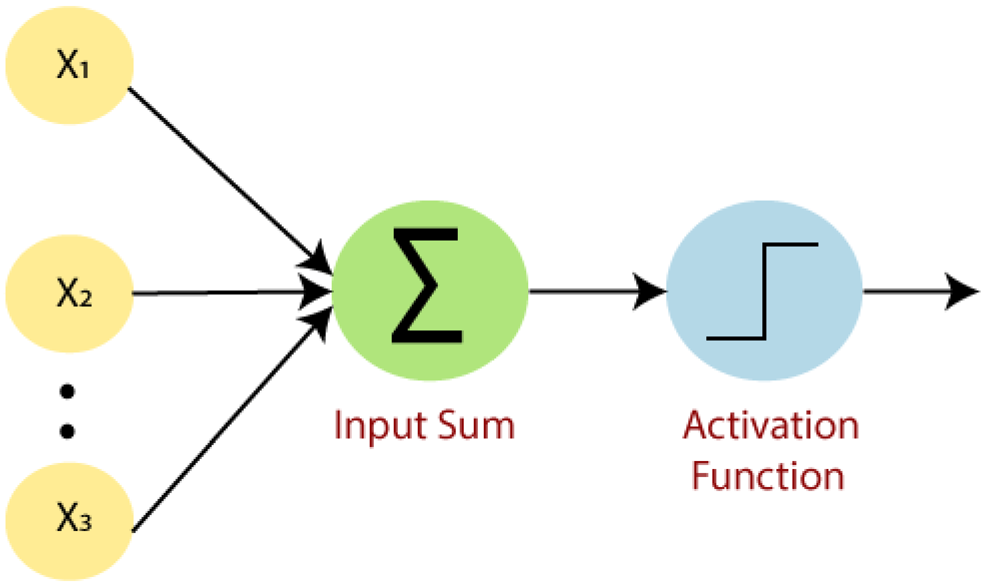
The activation function is used to map the input between the required value like (0, 1) or (-1, 1).

A regular neural network looks like this:

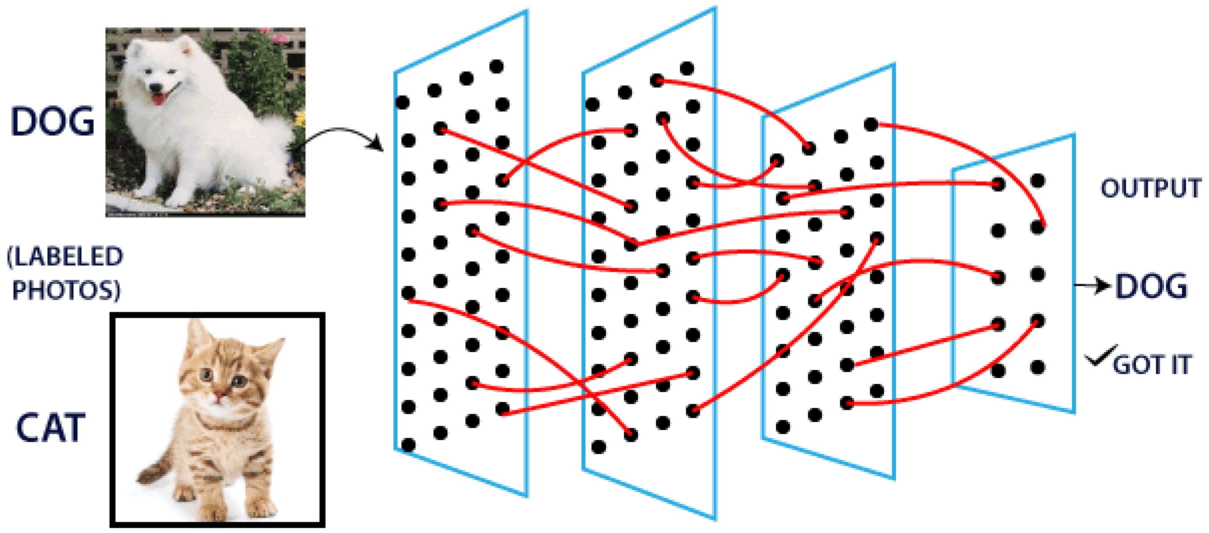

X1
X1
X1
X1
W1
W2
W3
W4
$\sum_{i=1}^{m}$ (wixi)+bias
f(x)= 1, if ∑ wx+b≥0 ; 0, if ∑ wx+b<0
ŷ
Inputs
Weights
Summation and Bias
Activation
Output

The perceptron consists of 4 parts.

- **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.
- **Weights and Bias:**
  Weight: 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.
  Bias: It is the same as the intercept added in a linear equation. It is an additional parameter which task is to modify the output along with the weighted sum of the input to the other neuron.
- **Net sum:** It calculates the total sum.
- **Activation Function:** A neuron can be activated or not, is determined by an activation function. The activation function calculates a weighted sum and further adding bias with it to give the result.



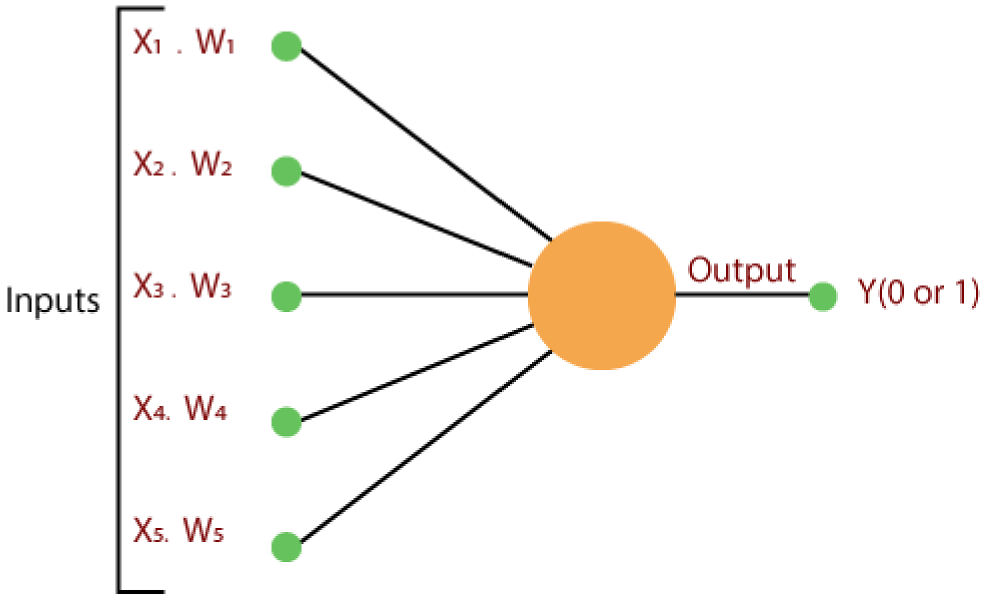
A standard neural network looks like the below diagram.



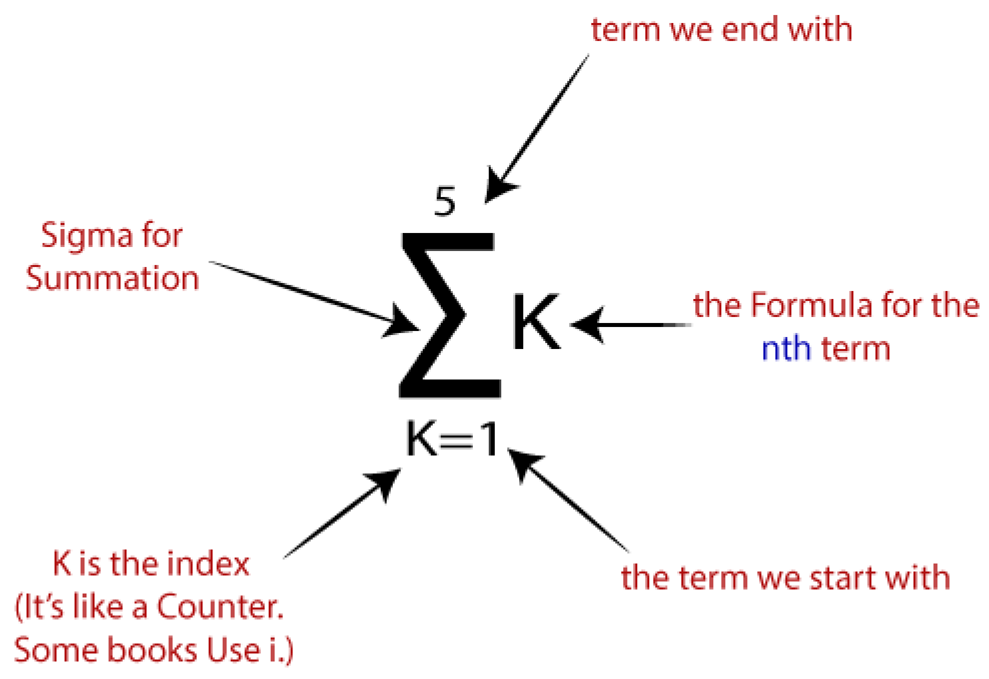
How does it work?

The perceptron works on these simple steps which are given below:

a. In the first step, all the inputs x are multiplied with their weights **w**.



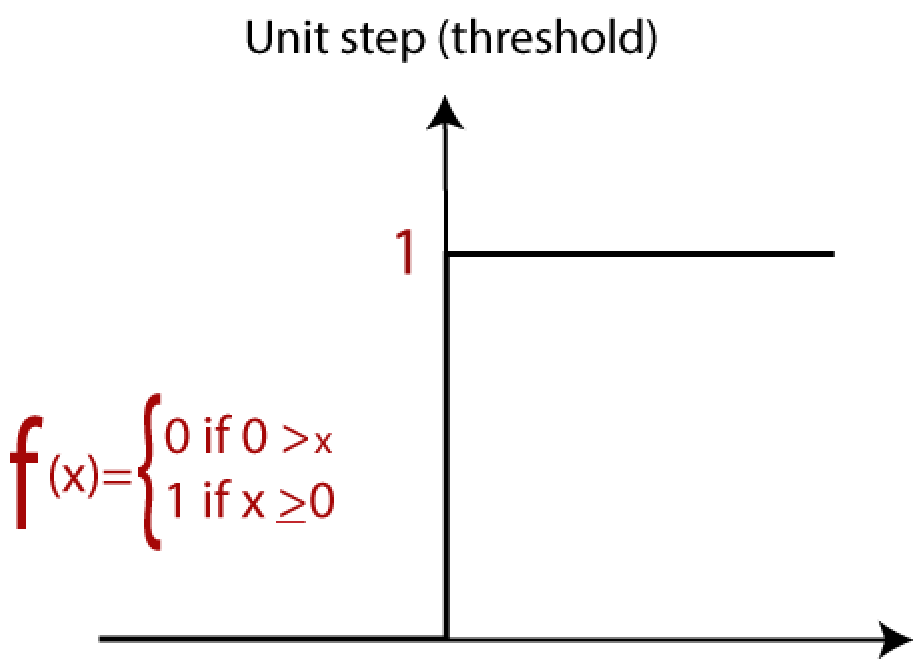
b. In this step, add all the increased values and call them the **Weighted sum**.


term we end with
5
Sigma for Summation
K
the Formula for the nth term
K=1
K is the index (It's like a Counter. Some books Use i.)
the term we start with

c. In our last step, apply the weighted sum to a correct **Activation Function**.

For Example:

A Unit Step Activation Function



There are two types of architecture. These types focus on the functionality of artificial neural networks as follows-

- Single Layer Perceptron
- Multi-Layer Perceptron

Single Layer Perceptron

The single-layer perceptron was the first neural network model, proposed in 1958 by Frank Rosenbluth. It is one of the earliest models for learning. Our goal is to find a linear decision function measured by the weight vector w and the bias parameter b.

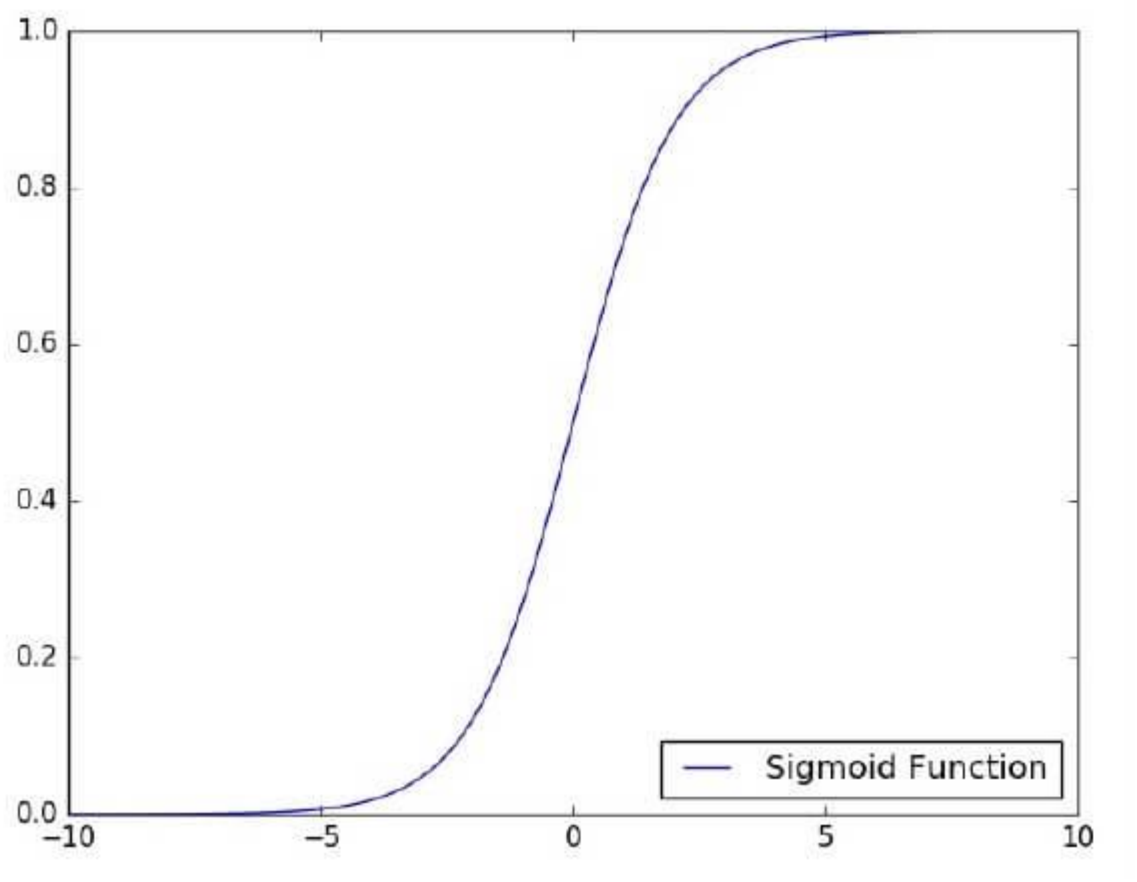
To understand the perceptron layer, it is necessary to comprehend artificial neural networks (ANNs).

The artificial neural network (ANN) is an information processing system, whose mechanism is inspired by the functionality of biological neural circuits. An artificial neural network consists of several processing units that are interconnected.

This is the first proposal when the neural model is built. The content of the neuron's local memory contains a vector of weight.

The single vector perceptron is calculated by calculating the sum of the input vector multiplied by the corresponding element of the vector, with each increasing the amount of the corresponding component of the vector by weight. The value that is displayed in the output is the input of an activation function.

Let us focus on the implementation of a single-layer perceptron for an image classification problem using TensorFlow. The best example of drawing a single-layer perceptron is through the representation of "**logistic regression**."



Now, We have to do the following necessary steps of training logistic regression-

- 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.

Hidden Layer Perceptron in TensorFlow

A hidden layer is an artificial neural network that is a layer in between **input layers** and **output layers**. Where the artificial neurons take in a set of weighted inputs and produce an output through an activation function. It is a part of nearly and neural in which engineers simulate the types of activity that go on in the human brain.

The hidden neural network is set up in some techniques. In many cases, weighted inputs are randomly assigned. On the other hand, they are fine-tuned and calibrated through a process called **backpropagation**.

The artificial neuron in the hidden layer of perceptron works as a biological neuron in the brain- it takes in its probabilistic input signals, and works on them. And it converts them into an output corresponding to the biological neuron's axon.

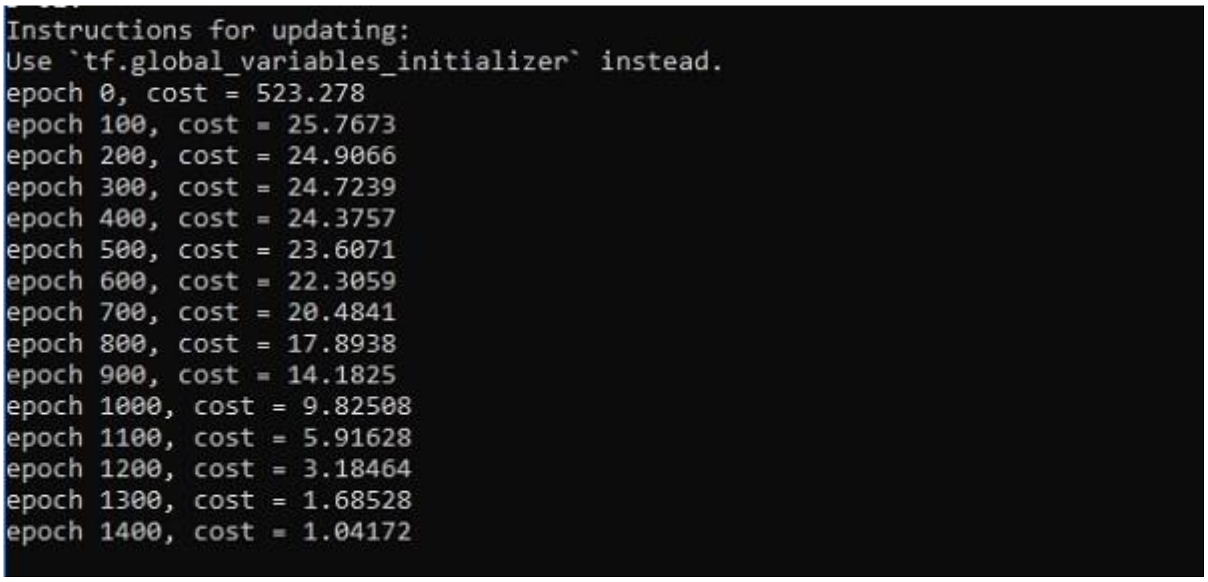
Layers after the input layer are called hidden because they are directly resolved to the input. The simplest network structure is to have a single neuron in the hidden layer that directly outputs the value.

Deep learning can refer to having many hidden layers in our neural network. They are deep because they will have been unimaginably slow to train historically, but may take seconds or minutes to prepare using modern techniques and hardware.

A single hidden layer will build a simple network.

The code for the hidden layers of the perceptron is shown below:

The two data are: **train** and **validation**, which are described in distinct colors as visible in the legend section.



```
Instructions for updating:
Use `tf.global_variables_initializer` instead.
epoch 0, cost = 523.278
epoch 100, cost = 25.7673
epoch 200, cost = 24.9066
epoch 300, cost = 24.7239
epoch 400, cost = 24.3757
epoch 500, cost = 23.6071
epoch 600, cost = 22.3059
epoch 700, cost = 20.4841
epoch 800, cost = 17.8938
epoch 900, cost = 14.1825
epoch 1000, cost = 9.82508
epoch 1100, cost = 5.91628
epoch 1200, cost = 3.18464
epoch 1300, cost = 1.68528
epoch 1400, cost = 1.04172
```

Creating an interactive section

We have two basic options when using TensorFlow to run our code:

- Build graphs and run sessions [Do all the set-up and then execute a session to implement a session to evaluate tensors and run operations].
- Create our coding and run on the fly.

For this first part, we will use the interactive session that is more suitable for an environment like Jupiter notebook.

1. sess = tf.InteractiveSession()

Creating placeholders

It's a best practice to create placeholder before variable assignments when using TensorFlow. Here we'll create placeholders to inputs ("Xs") and outputs ("Ys").

Placeholder "X": Represent the 'space' allocated input or the images.

- Each input has 784 pixels distributed by a 28 width x 28 height matrix.
- The 'shape' argument defines the tensor size by its dimensions.