UNIT-5

Neural Networks and Deep Learning: Introduction to Artificial Neural Networks with Keras, Implementing MLPs with Keras, Installing TensorFlow 2, Loading and Pre-processing Data with TensorFlow.

Introduction to Artificial Neural Networks:

A neural network is a method in artificial intelligence that teaches computers to process data in a way that is inspired by the *human brain*. It is a type of machine learning process, called deep learning, that uses interconnected ***nodes or neurons*** in a layered structure that resembles the human brain.

Neural networks have several use cases across many industries, such as the following:

- Medical diagnosis, Targeted marketing, Financial predictions, Electrical load and energy demand forecasting process and quality control, Chemical compound identification, like image recognition, natural language processing, and more

Types of Neural Networks in Machine Learning:

1. **ANN:**

ANN is also known as an *artificial neural network*. It is a feed-forward neural network because the inputs are sent in the forward direction. It is comparatively less powerful than CNN and RNN.

2. **CNN:**

Convolutional Neural Networks are mainly used for Image Data. It is more powerful than both ANN and RNN.

3. **RNN:**

It is also known as Recurrent Neural Networks. It is used to process and interpret time series data. The most known types of RNN are LSTM (Long Short Term Memory) Networks.

ANN(Artificial Neural Network): -

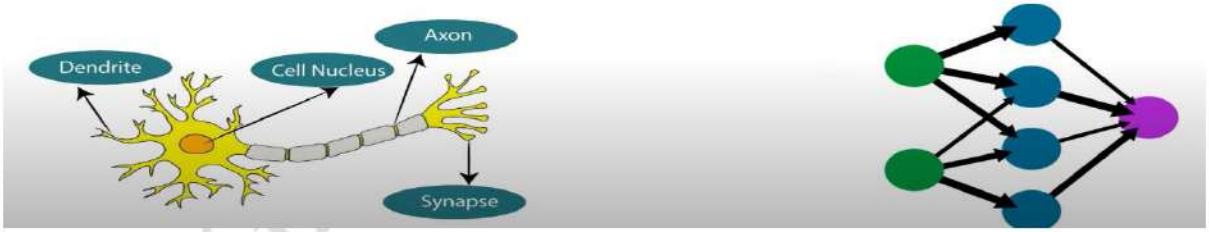
The term ***"Artificial Neural Network"*** is derived from Biological neural networks that develop the structure of a human brain.

ANN was first introduced in 1943 by the neurophysiologist ***Warren McCulloch*** and the mathematician ***Walter Pitts***.

ANN is inspired by the working of a human brain. The brain has neurons interconnected to one another, and ANN also has neurons, that are interconnected to one another in various layers of the networks. These neurons are known as **"nodes"**

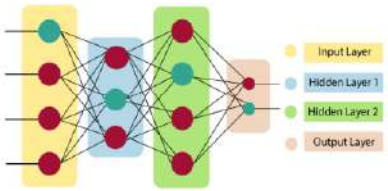
Neural Network is a set of algorithms that try to recognize the patterns, relationships, and information from the data through a process that is inspired by and works like the human brain.

Biological Neural Network	Artificial Neural Network
Dendrites	Inputs
Cell nucleus	Nodes
Synapse	Weights
Axon	Output



Components of ANN:

- Input layer
- Hidden layer (Middle layer)
- Output layer



Input layer: Input nodes receive inputs/information from the outside world.

Hidden layer: Hidden layer is the set of neurons where all the computations are performed on the input data.

There can be any number of hidden layers in a neural network. The simplest network consists of a single hidden layer.

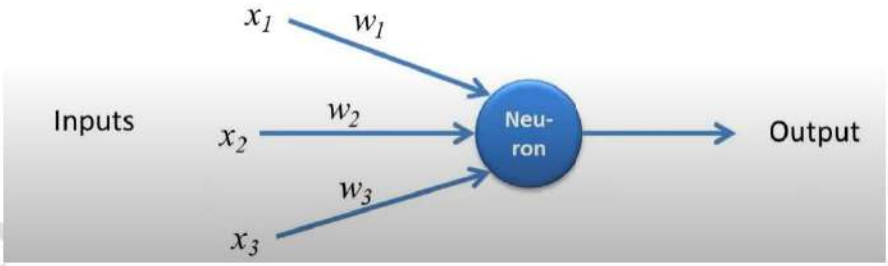
Output layer: The output layer is the output/Conclusions derived from all the computations performed. There can be single or multiple nodes in the output layer.

If we have a binary classification problem the output node is 1 but in the case of multi-class classification, the output nodes can be more than 1.

The Perceptron:

The Perceptron is one of the simplest ANN architectures, invented in 1957 by ***Frank Rosenblatt***. It is based on a slightly different artificial neuron (see Figure 10-4) calleda threshold logic unit (TLU), or sometimes a linear threshold unit (LTU): the inputs and output are now numbers (instead of binary on/off values) and each input connection is associated with a weight. The TLU computes a weighted sum of its inputs ($z = w_1 x_1 + w_2 x_2 + \cdots + w_n x_n = x^T w$), then applies a *step function* to that sum and outputs the result:

$h_w(x) = \text{step}(z)$, where $z = x^T w$.



Note: Perceptron is a simple form of Neural Network and consists of a single layer where all the mathematical computations are performed.

Computation performed in hidden layers is done in 2 steps which are as follows:

Step 1: All the inputs are multiplied by their weights

Weight is the gradient or coefficient of each variable. It shows the strength of the particular input. After assigning the strength of the particular input. After assigning the weights, a bias variable is added.

Bias is constant which helps the model to fit in the best way possible.

$z = w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n + b$

Where w_1, w2, w3, w4.. Are the weights assigned to the inputs $x_1, x_2, x_3 \ldots x_n$

and b is the **bias**.

Step 2: Activation function is applied to the linear equation $Z(z = w_1 * x_1 + w_2 * x_2 + \cdots + w_n * x_n + b)$

The **activation function** is a *nonlinear transformation that* is applied to the input before sending it to the next layer of neurons.

The computation results from the hidden layer pass to the last layer i.e. output layer which gives us the final output.

The process explained above is known as ***forwarding propagation***.

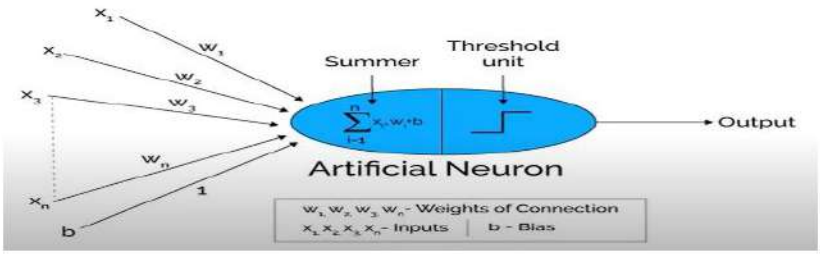
After getting the prediction from the output layer, the error is calculated i.e. the difference between the *actual* and the *predicted* output.

If the error is large, the steps are taken to minimize the error and for the same purpose, ***Back Propagation*** is performed.

Back Propagation:

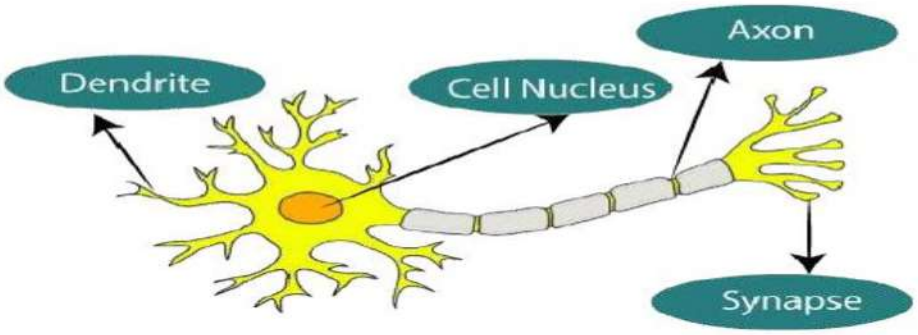
Back Propagation is the process of updating and finding the optimal values of the weights or coefficients which helps to minimize the error i.e. difference between the actual and predicted values.

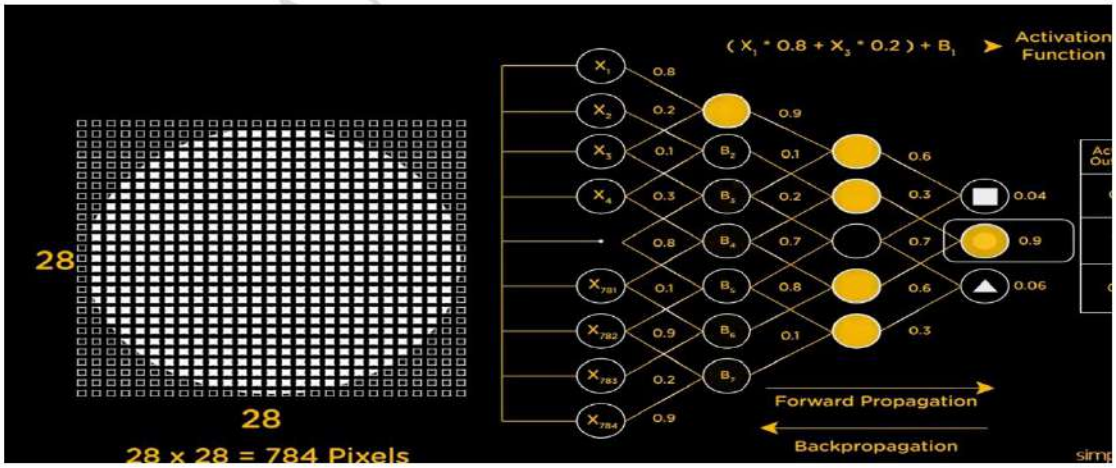
Weights are updated with the help of the optimizers. Optimizers are the methods/mathematical formulations to change the attributes of neural networks i.e. weights to minimize the error.



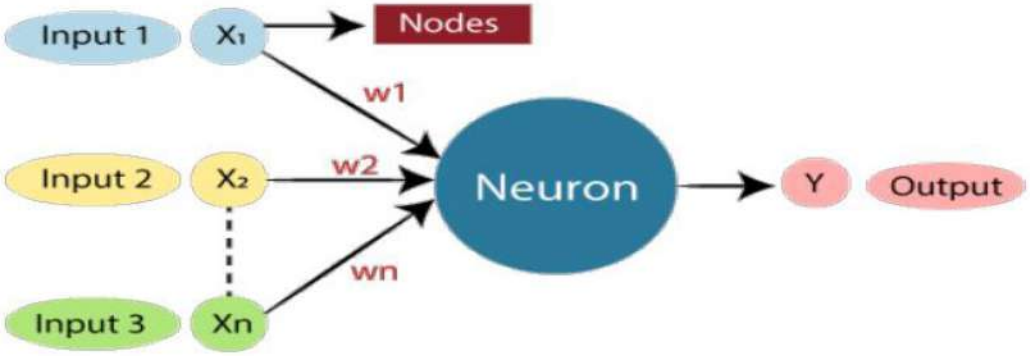
Implementing MLPs with Keras:

Multilayer Perceptron also known as ANN consists of more than one perception which is grouped together to form a multiple layer neural network.





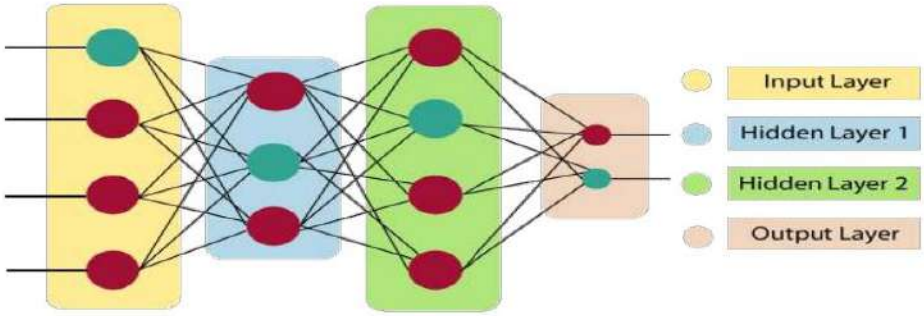
The above figure illustrates the ***typical diagram of a Biological Neural Network. The typical Artificial Neural Network looks something like the given figure.***



Dendrites from Biological Neural Network represent inputs in Artificial Neural Networks, cell nucleus represents Nodes, synapse represents Weights, and Axon represents Output.

The architecture of an artificial neural network:

To understand the concept of the architecture of an artificial neural network, we have to understand what a neural network consists of.Todefine a neural network that consists of a large number of artificial neurons, which are termed units arranged in a sequence of layers. Let us look at various types of layers available in an artificial neural network.



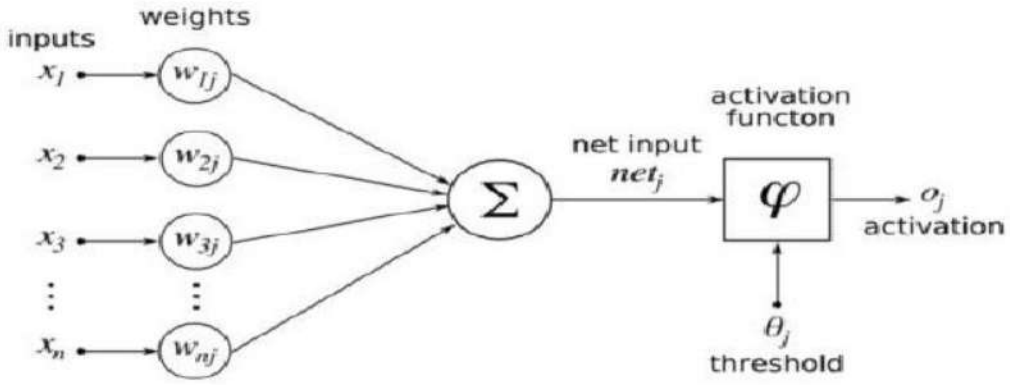
Input Layer: As the name suggests, it accepts inputs in several different formats provided by the programmer.

Hidden Layer: The hidden layer presents in between the input and output layers. It performs all the calculations to find hidden features and patterns.

Output Layer: The input goes through a series of transformations using the hidden layer, which finally results in output that is conveyed using this layer. The artificial neural network takes input and computes the weighted sum of the inputs and includes a bias. This computation is represented in the form of a transfer function.

$$\sum_{i=1}^{n} Wi * Xi + b$$

It determines weighted total is passed as an inputto an activation function toproduce the output. Activation functions choose whether a node shouldfire or not. Only those who are fired make it to the output layer.There are distinctive activationfunctions available that can be applied to the sort of task we are performing.

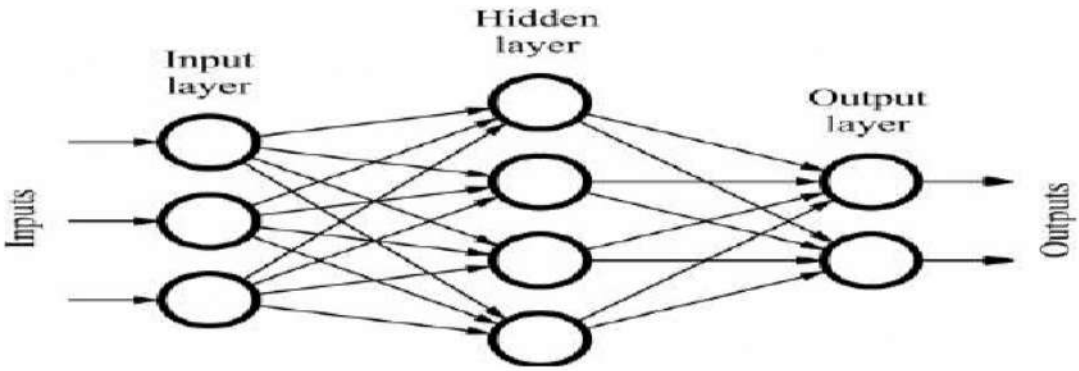


Types of Artificial Neural Network:

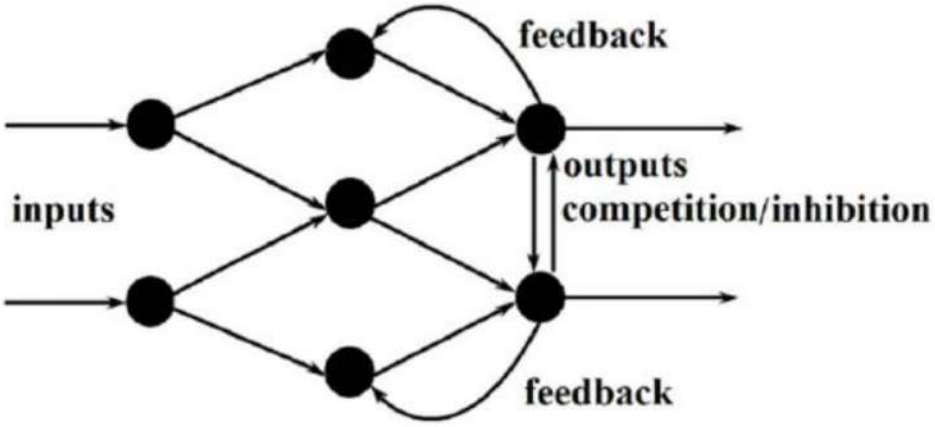
There are various types of Artificial Neural Networks (ANN) depending upon the human brainneuronandnetworkfunctions, anartificialneuralnetworksimilarly performs tasks.

1. **Feed Forward Neural Network:** Signals travel in one way i.e. from input to output only in Feed Forward Neural Network. There is no feedback or loops. The output of any layer does not affect that same layer in suchnetworks.Feed-forward neuralnetworksare straightforward networksthat associate inputs with outputs. They have fixed inputs and

outputs. They are mostly used in pattern generation, pattern recognition, and classification.



2. **Feedback Neural Network:** Signals can travelin both directions in Feedback neural networks. Feedback neural networks are very powerful and can get very complicated. Feedback neural networks are dynamic. The 'state' in such a network keeps changing until it reaches an equilibrium point. They remain at the equilibrium point until the input changes and a new equilibrium needs to be found. Feedback neural network architecture is also referred to as interactive or recurrent, although the latter term is often used to denote feedback connections in single-layer organizations. Feedback loops are allowed in such networks. They are used in content addressable memories.

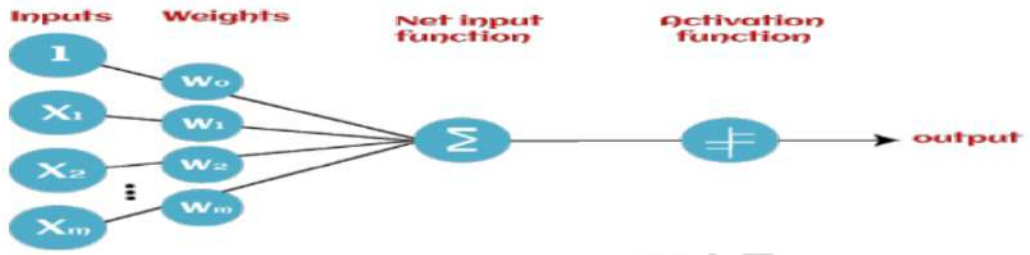


Perceptron in Machine Learning:

Perceptron is a Machine Learning algorithm for supervised learning of various binary classification tasks. Further, Perceptron is also understood as an Artificial Neuron or neural network unit that helps to detect certain input data computations in business intelligence.

Basic Components of Perceptron

Mr. Frank Rosenblatt invented the perceptron model as a binary classifier which contains three main components. These are as follows:



Input Nodes or Input Layer: This is the primary component of Perceptron which accepts the initial data into the system for further processing .Each input node contains a real numerical value.

Wight and Bias: The weight parameter represents the strength of the connection between units. This is another important parameter of Perceptron components. Weight is directly proportional to the strength of the associated input neuron in deciding the output. Further, Bias can be considered as the line of intercept in a linear equation.

Activation Function:

These are the final and important components that help to determine whether the neuron will fire or not. Activation Function can be considered primarily as a step function.

Types of Activation functions:

- Sign function
- Step function, and
- Sigmoid function

Types of Perceptron Models Based on the layers, Perceptron models are divided into two types.

These are as follows:

1. Single-layer Perceptron Model

2. Multi-layer Perceptron model

Single Layer Perceptron Model:

This is one of the easiest artificial neural networks (ANN) types. A single-layered perceptron model consists feed-forward network and also includes a threshold transfer function inside the model. The main objective of the single-layer perceptron model is to analyze the linearly separable objects with binary outcomes.

In a single-layer perceptron model, its algorithms do not contain recorded data, so it begins with an inconstantly allocated input for weight parameters. Further, it sums up all inputs (weight). After adding all inputs, if the total sum of all inputs is more than a predetermined value, the model gets activated and shows the output value as +1.

If the outcome is the same as the pre-determined threshold value, then the performance of this mode l is stated as satisfied, and weight demand does not change. However, this model consists of a few discrepancies triggered when multiple weight input values are fed into the model. Hence, to find the desired output and minimize errors, some changes should be necessary for the weights input.

"Single-layer perceptron can learn only linearly separable patterns."

Multi-Layered Perceptron Model: Like a single-layer perceptron model, a multi-layer perceptron model also has the same model structure but has a greater number of hidden layers.

The multi-layer perceptron model is also known as the Back propagation algorithm, which executes in two stages as follows:

Forward Stage: Activation functions start from the input layer in the forward stage and terminate on the output layer.

Backward Stage: In the backward stage, weight and bias values are modified as per the model's requirement. In this stage, the error between actual output and demanded originated backward on the output layer and ended on the input layer.

Hence, a multi-layered perceptron model is considered as multiple artificial neural network shaving various layers in which the activation function does not remain linear, similar to a single-layer perceptron model. Instead oflinear, the activation function can be executed as sigmoid, TanH, ReLU, etc., for deployment.

A multi-layer perceptron model has greater processing power and can process linear and non-linear patterns. Further, it can also implement logic gates such as AND, OR, XOR, NAND, NOT, XNOR, and NOR.

Advantages of Multi-Layer Perceptron:

- A multi-layered perceptron model can be used to solve complex non-linear problems.
- It works well with both small and large input data.
- It helps us to obtain quick predictions after the training.
- It helps to obtain the same accuracy ratio with large as well as small data.

Disadvantages of Multi-Layer Perceptron:

- In Multi-layer perceptron, computations are difficult and time-consuming.
- In multi-layer Perceptron, it is difficult to predict how much the dependent variable affects each independent variable.
- The model's functioning depends on the quality of the training.

Perceptron Function:

Perceptron function "f(x)" can be achieved as output by multiplying the input 'x' with the learned weight coefficient 'w'.

Mathematically, we can express it as follows:

f(x)=1; if w.x+b>0

otherwise, f(x)=0

'w' represents real-valued weights vector

'b' represents the bias

'x' represents a vector of input x values.

Characteristics of Perceptron:

The perceptron model has the following characteristics.

1. Perceptron is a machine learning algorithm for supervised learning of binary classifiers.

2. In Perceptron, the weight coefficient is automatically learned.

3. Initially, weights are multiplied with input features, and the decision is made whether the neuron is fired or not.

4. The activation function applies as a rule to check whether the weight function is greater than zero.

5. The linear decision boundary is drawn, enabling the distinction between the two linearly separable classes +1 and -1.

6. If the added sum of all input values is more than the threshold value, it must have an output signal; otherwise, no output will be shown.

Lilimitations of the Perceptron Model:

A perceptron model has limitations as follows:

- The output of a perceptron can only be a binary number (0 or 1) due to the hard limit transfer function.
- Perceptron can only be used to classify the linearly separable sets of input vectors. If input vectors are non-linear, it is not easy to classify them propearly

Implementing MLPs with Keras:

Keras is a high-level Deep Learning API that allows you to easily build, train, evaluate, and execute all sorts of neural networks. »https://keras.io» Computation backend: Tensor Flow, Microsoft Cognitive Toolkit (CNTK), Theano, Apache MXNet, Apple's Core ML, JavaScript or TypeScript, and PlaidML.

• tf.Keras: Extended Keras implementation based on Tensor Flow with Tensor Flow-specific features.

• PyTorch is also quite popular. Multi backend Keras vs. tf.keras.

Keras provides some utility functions to fetch and load common datasets.

• Loading data from Keras is different from Scikit-Learn: » Every image is represented as a 28 × 28 array rather than a 1D array of size 784. » The pixel intensities are represented as integers (from 0 to 255) rather than floats (from 0.0 to 255.0)

• Since we are going to train the neural network using Gradient Descent, we must scale the input features down to the 0–1 range by dividing them by 255.0:

Installing Tensor Flow 2:

We can download TensorFlow in our system in 2 ways:

1. Through pip (Python package library)
2. Through Anaconda Navigator (conda)

Through pip :- So, firstly we have to install and set-up anaconda in our system through pip. The following are the requirements for TensorFlow to work on our computer.

o Tensor Flow has only supported 64-bit Python 3.5.x or Python 3.6.x on Windows

o When we download the Python 3.5.x version, it comes with the pip3 package manager. (This is the program that we are going to need for our users to install Tensor Flow on Windows).

Step1: Download Python from the below link: https://www.python.org/downloads/release/python-352/ After that,

Step 2: We will be brought to another page, where we will need to select either the x86-64 or amd 64 installer to install Python. We use here Windows x86-64 executable installer.

Step 3: For this tutorial, I'll be choosing to Add Python 3.5 to PATH

Step4: Now, we will be able to see the message "Set-up was successful." A way to confirm that it has been installed successfully is to open your Command Prompt and check the version.

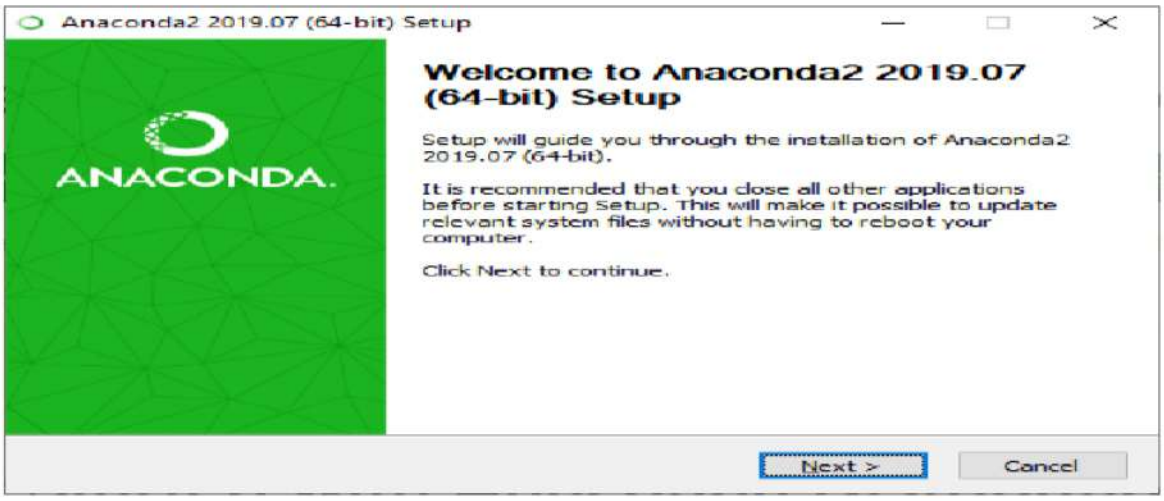
Ininstallation of Tensor Flow through conda

These are the following steps which are given below:

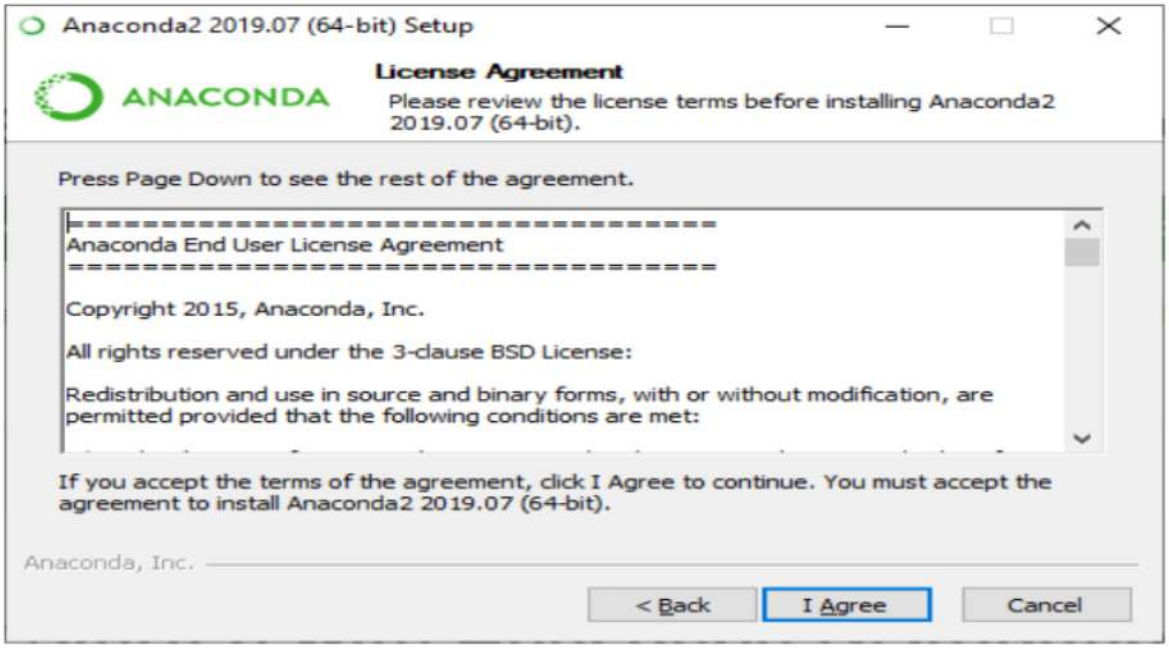
Firstly, we have to open the official site of Anaconda and download Anaconda from the below link: https://www.anaconda.com/distribution/

After that, we have to download Anaconda from the below highlighted Python 2.7 version.

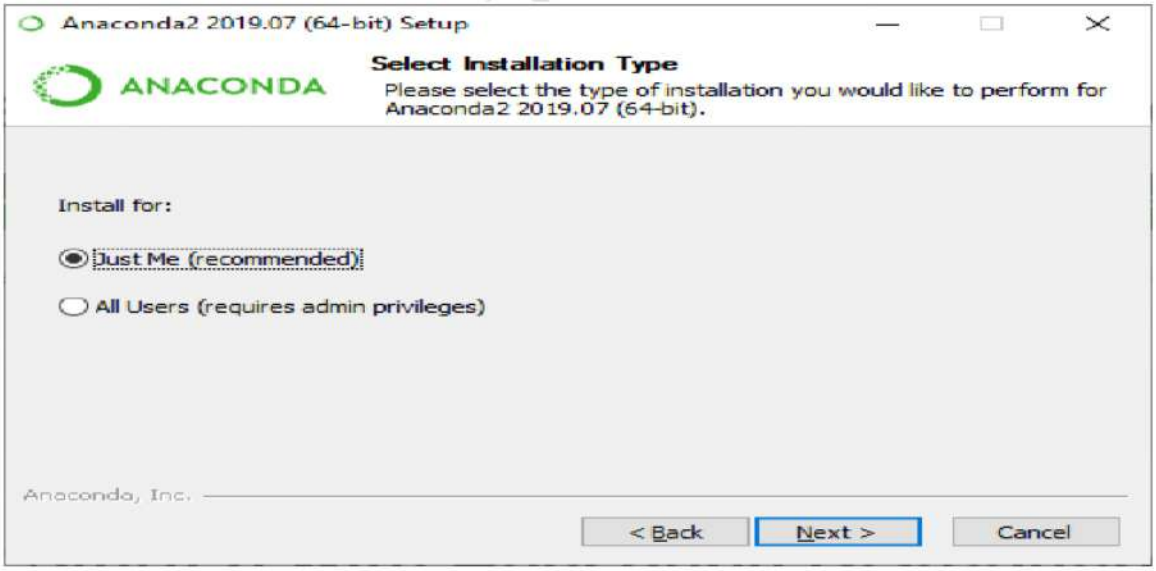
It will successfully be downloaded into our system. After that, we have to install Anaconda in our system.



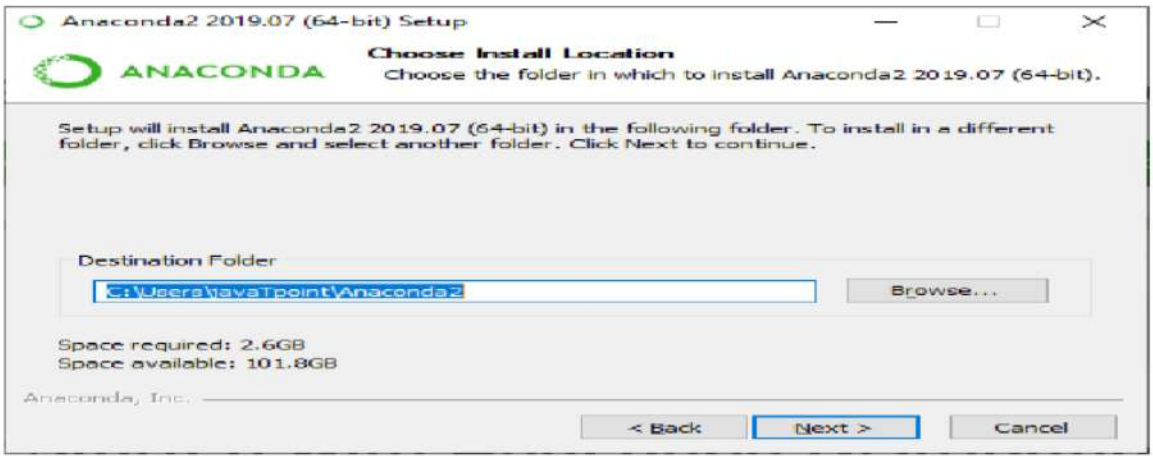
Loading and Pre-processing Data with Click on "Next."



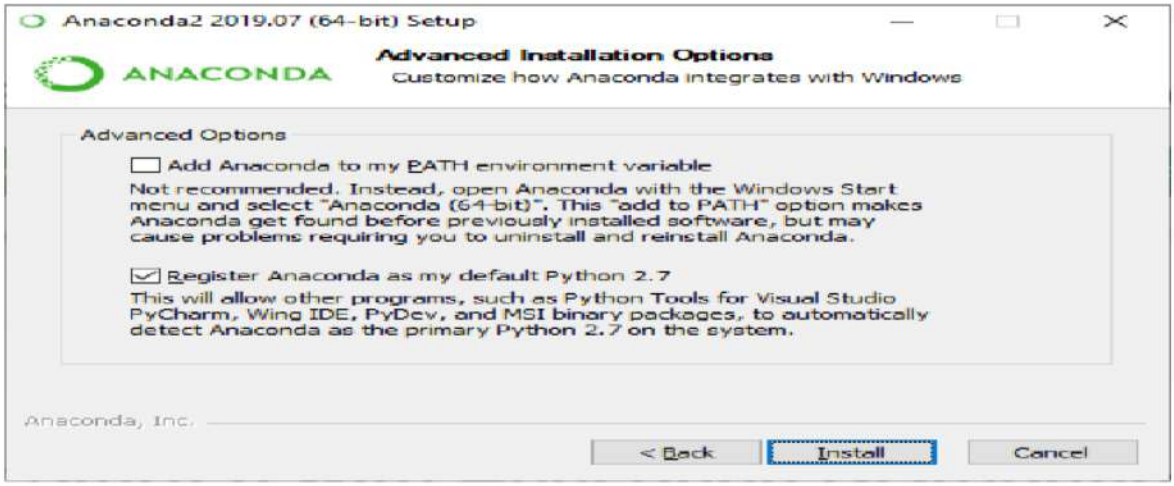
Click on "I Agree."



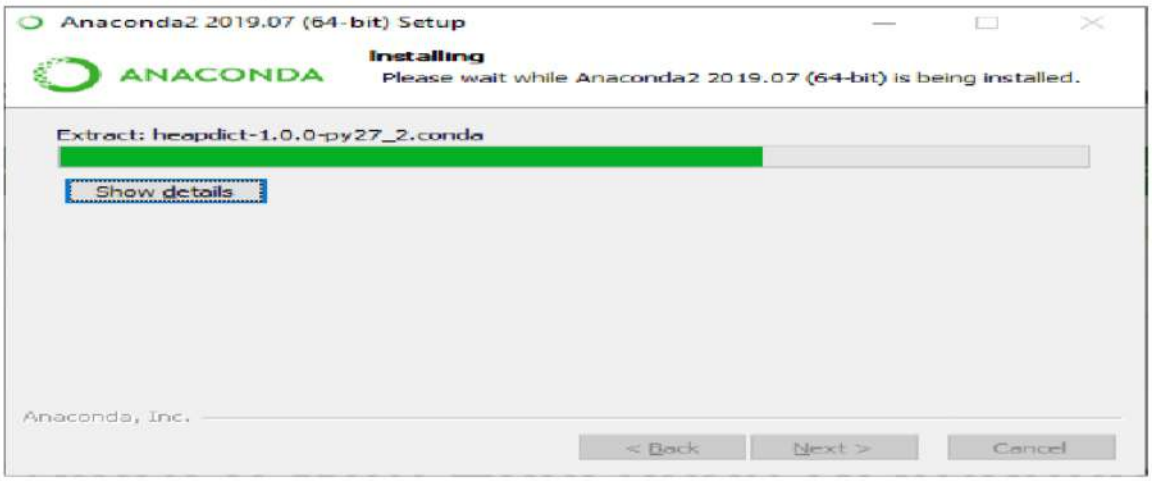
Again click on "Next."



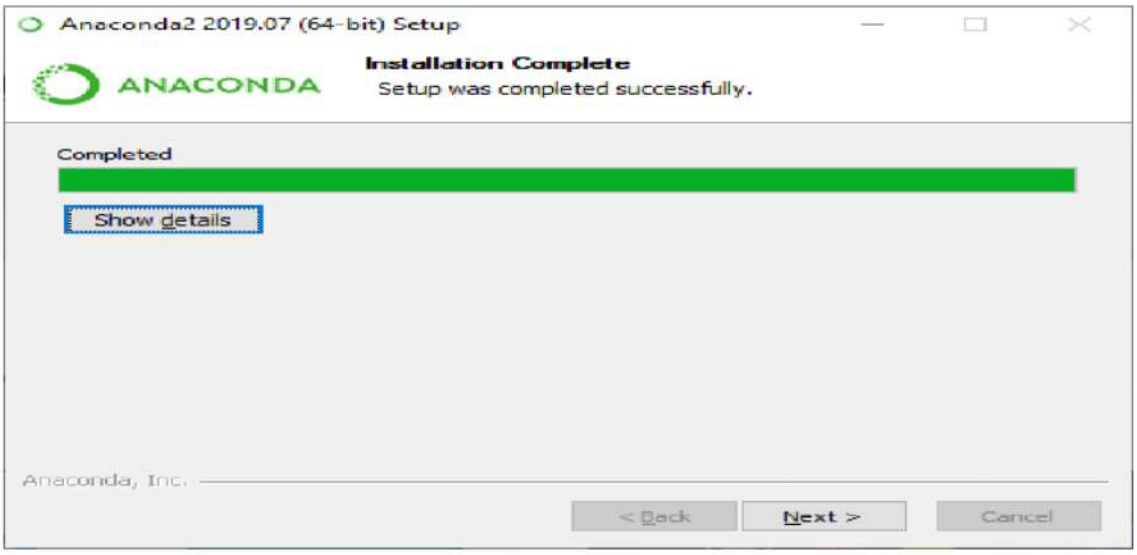
Click on "Next" again.



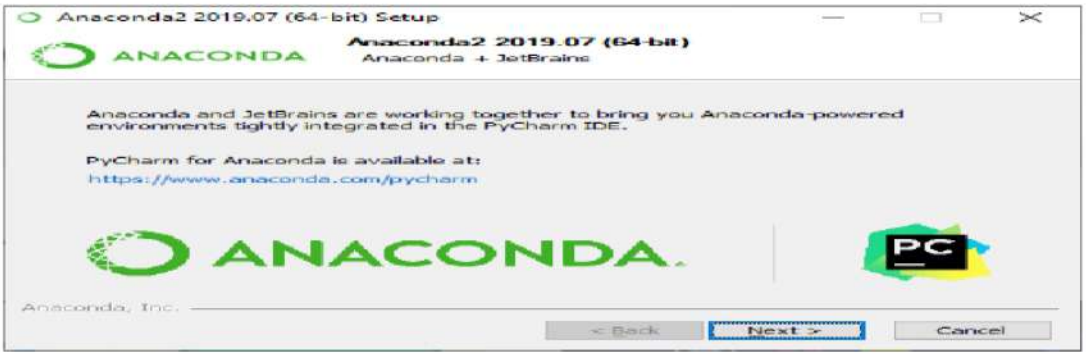
Click on "Install."



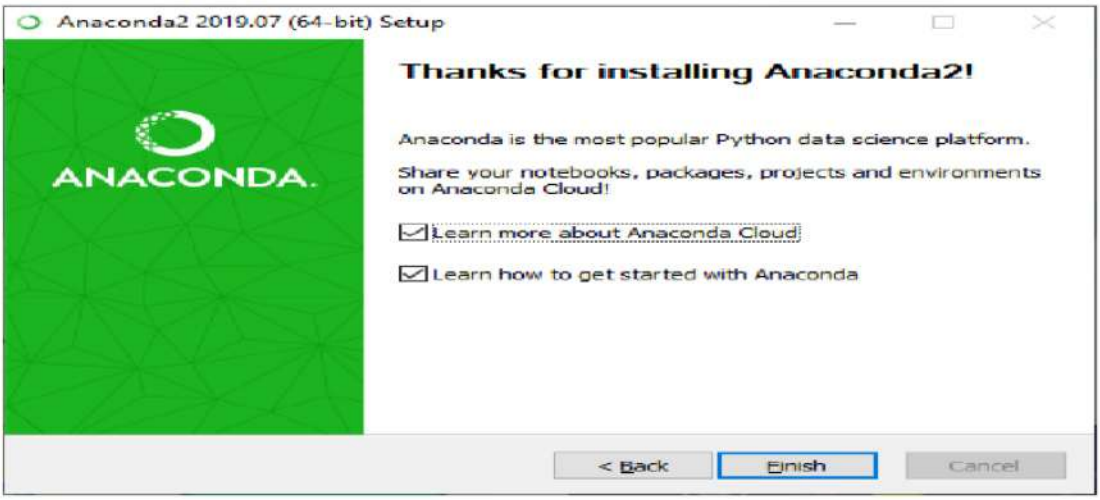
Click on "Next."



Click on "Next."



Click on "Next."



After clicking on "finish."

It will be successfully installed in our system.

After that, we have to run the given command to set up the Tensor Flow and libraries.

Loading and Pre-processing Data with Tensor Flow.

 :First, you will use high-level Keras pre-processing utilities (such as tf.keras.utils.image_dataset_from_directory) and layers (such as tf.keras.layers.Rescaling) to read a directory of images on disk.

 Next, you will write your input pipeline from scratch using tf.data.

 Finally, you will download a dataset from the large catalogue available in Tensor Flow Datasets.

Setup

```
import numpy as np
import os
import PIL
import PIL.Image
import tensorflow as tf
import tensorflow_datasets as tfds
```

2023-01-1302:21:47.621249:W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer.so.7'; dlerror: libnvinfer.so.7: cannot open shared object file: No such file or directory 2023-01-13

02:21:47.621371: W tensorflow/compiler/xla/stream_executor/platform/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer_plugin.so.7'; dlerror: libnvinfer_plugin.so.7: cannot open shared object file: No such file or directory

2023-01-13 02:21:47.621382: W tensorflow/compiler/tf2tensorrt/utils/py_utils.cc:38] TF-TRT Warning: Cannot dlopen some TensorRT libraries.

If you would like to use Nvidia GPU with TensorRT, please make sure the missing libraries mentioned above are installed properly.

print(tf.__version__)

2.11.0

Download the flowers dataset This tutorial uses a dataset of several thousand photos of flowers.

The flowers dataset contains five sub-directories, one per class:

flowers_photos/

daisy/

dandelion/

roses/

sunflowers/

tulips/

Note: all images are licensed CC-BY, creators are listed in the LICENSE.txt file.

```
import pathlib
dataset_url="https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz"
archive = tf.keras.utils.get_file (origin=dataset_url, extract=True)
data_dir = pathlib.Path (archive).with_suffix (")
```

Downloading data from

https://storage.googleapis.com/download.tensorflow.org/example_images/flower_photos.tgz
228813984/228813984 [==============================] - 1s 0us/step

After downloading (218MB), you should now have a copy of the flower photos available. There are 3,670 total images:

```
image_count = len (list(data_dir.glob('*/*.jpg')))
print(image_count)
```

3670

Each directory contains images of that type of flower.
Here are some roses:

```
roses = list (data_dir.glob('roses/*'))
PIL.Image.open (str (roses[0]))
```

```
roses = list(data_dir.glob('roses/*'))
PIL.Image.open(str(roses[1]))
```