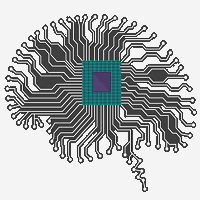
**Unit V:**

**Neural Networks and Deep Learning**: Introduction to Artificial Neural Networks with Keras, Implementing MLPs with Keras, Installing TensorFlow 2, Loading and Preprocessing Data with TensorFlow

Artificial Neural Network Tutorial



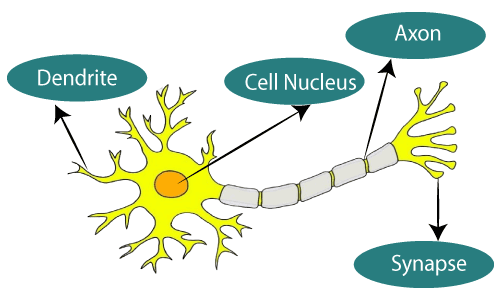
Artificial Neural Network Tutorial provides basic and advanced concepts of ANNs. Our Artificial Neural Network tutorial is developed for beginners as well as professions.

The term "Artificial neural network" refers to a biologically inspired sub-field of artificial intelligence modeled after the brain. An Artificial neural network is usually a computational network based on biological neural networks that construct the structure of the human brain. Similar to a human brain has neurons interconnected to each other, artificial neural networks also have neurons that are linked to each other in various layers of the networks. These neurons are known as nodes.

Artificial neural network tutorial covers all the aspects related to the artificial neural network. In this tutorial, we will discuss ANNs, Adaptive resonance theory, Kohonen self-organizing map, Building blocks, unsupervised learning, Genetic algorithm, etc.

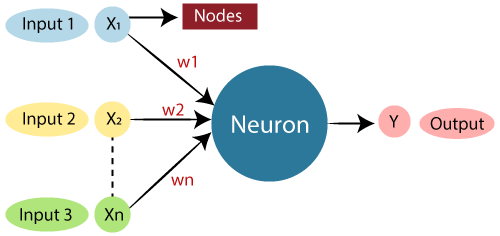
What is Artificial Neural Network?

The term "**Artificial Neural Network**" is derived from Biological neural networks that develop the structure of a human brain. Similar to the human brain that has neurons interconnected to one another, artificial neural networks also have neurons that are interconnected to one another in various layers of the networks. These neurons are known as nodes.



**The given figure illustrates the typical diagram of 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.

Relationship between Biological neural network and artificial neural network:

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

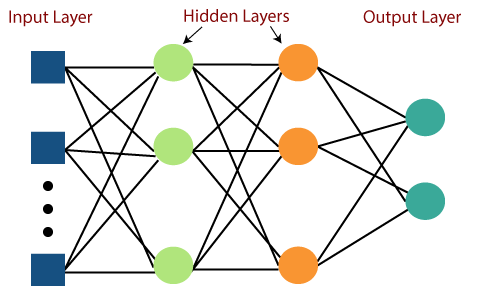
An **Artificial Neural Network** in the field of **Artificial intelligence** where it attempts to mimic the network of neurons makes up a human brain so that computers will have an option to understand things and make decisions in a human-like manner. The artificial neural network is designed by programming computers to behave simply like interconnected brain cells.

There are around 1000 billion neurons in the human brain. Each neuron has an association point somewhere in the range of 1,000 and 100,000. In the human brain, data is stored in such a manner as to be distributed, and we can extract more than one piece of this data when necessary from our memory parallelly. We can say that the human brain is made up of incredibly amazing parallel processors.

We can understand the artificial neural network with an example, consider an example of a digital logic gate that takes an input and gives an output. "OR" gate, which takes two inputs. If one or both the inputs are "On," then we get "On" in output. If both the inputs are "Off," then we get "Off" in output. Here the output depends upon input. Our brain does not perform the same task. The outputs to inputs relationship keep changing because of the neurons in our brain, which are "learning."

Multi-Layer perceptron defines the most complex architecture of artificial neural networks. It is substantially formed from multiple layers of the perceptron. TensorFlow is a very popular deep learning framework released by, and this notebook will guide to build a neural network with this library. If we want to understand what is a Multi-layer perceptron, we have to develop a multi-layer perceptron from scratch using Numpy.

The pictorial representation of multi-layer perceptron learning is as shown below-



MLP networks are used for supervised learning format. A typical learning algorithm for MLP networks is also called **back propagation's algorithm**.

A multilayer perceptron (MLP) is a feed forward artificial neural network that generates a set of outputs from a set of inputs. An MLP is characterized by several layers of input nodes connected as a directed graph between the input nodes connected as a directed graph between the input and output layers. MLP uses backpropagation for training the network. MLP is a deep learning method.

Now, we are focusing on the implementation with MLP for an image classification problem.

# Import MINST data

from tensorflow.examples.tutorials.mnist **import** input\_data

mnist = input\_data.read\_data\_sets("/tmp/data/", one\_hot = True)

**import** tensorflow as tf

**import** matplotlib.pyplot as plt

# Parameters

learning\_rate = 0.001

training\_epochs = 20

batch\_size = 100

display\_step = 1

# Network Parameters

n\_hidden\_1 = 256

# 1st layer num features

n\_hidden\_2 = 256 # 2nd layer num features

n\_input = 784 # MNIST data input (img shape: 28\*28) n\_classes = 10

# MNIST total classes (0-9 digits)

  # tf Graph input   x = tf.placeholder("float", [None, n\_input])

y = tf.placeholder("float", [None, n\_classes])

  # weights layer 1

h = tf.Variable(tf.random\_normal([n\_input, n\_hidden\_1])) # bias layer 1   bias\_layer\_1 = tf.Variable(tf.random\_normal([n\_hidden\_1]))

# layer 1 layer\_1 = tf.nn.sigmoid(tf.add(tf.matmul(x, h), bias\_layer\_1))

# weights layer 2

w = tf.Variable(tf.random\_normal([n\_hidden\_1, n\_hidden\_2]))

# bias layer 2   bias\_layer\_2 = tf.Variable(tf.random\_normal([n\_hidden\_2]))

# layer 2

layer\_2 = tf.nn.sigmoid(tf.add(tf.matmul(layer\_1, w), bias\_layer\_2))

# weights output layer

output = tf.Variable(tf.random\_normal([n\_hidden\_2, n\_classes]))

# biar output layer

bias\_output = tf.Variable(tf.random\_normal([n\_classes])) # output layer

output\_layer = tf.matmul(layer\_2, output) + bias\_output

# cost function

cost = tf.reduce\_mean(tf.nn.sigmoid\_cross\_entropy\_with\_logits(

   logits = output\_layer, labels = y))

#cost = tf.reduce\_mean(tf.nn.sigmoid\_cross\_entropy\_with\_logits(output\_layer, y))

# optimizer

optimizer = tf.train.AdamOptimizer(learning\_rate = learning\_rate).minimize(cost)

# optimizer = tf.train.GradientDescentOptimizer(

   learning\_rate = learning\_rate).minimize(cost)

# Plot settings

avg\_set = []

epoch\_set = []

# Initializing the variables

init = tf.global\_variables\_initializer()

# Launch the graph

with tf.Session() as sess:

   sess.run(init)

        # Training cycle

**for** epoch in range(training\_epochs):

      avg\_cost = 0.

      total\_batch = **int**(mnist.train.num\_examples / batch\_size)

      # Loop over all batches

**for** i in range(total\_batch):

         batch\_xs, batch\_ys = mnist.train.next\_batch(batch\_size)

         # Fit training using batch data sess.run(optimizer, feed\_dict = {

            x: batch\_xs, y: batch\_ys})            # Compute average loss

         avg\_cost += sess.run(cost, feed\_dict = {x: batch\_xs, y: batch\_ys}) / total\_batch

      # Display logs per epoch step

**if** epoch % display\_step == 0:

         print            Epoch:", '%04d' % (epoch + 1), "cost=", "{:.9f}".format(avg\_cost)

      avg\_set.append(avg\_cost)

      epoch\_set.append(epoch + 1)

   print

   "Training phase finished"

        plt.plot(epoch\_set, avg\_set, 'o', label = 'MLP Training phase')

   plt.ylabel('cost')

   plt.xlabel('epoch')

   plt.legend()

   plt.show()

   # Test model

   correct\_prediction = tf.equal(tf.argmax(output\_layer, 1), tf.argmax(y, 1))

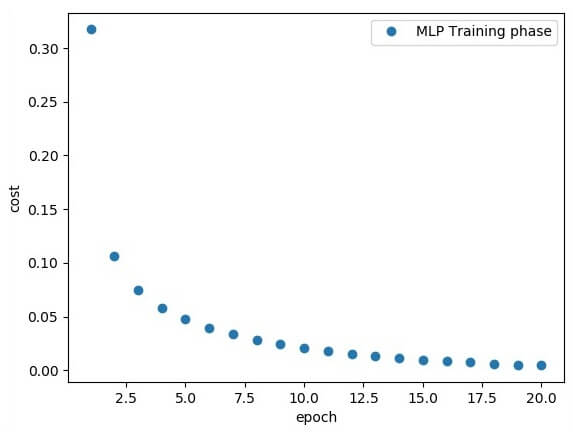
# Calculate accuracy

   accuracy = tf.reduce\_mean(tf.cast(correct\_prediction, "float"))

   print

   "Model Accuracy:", accuracy.eval({x: mnist.test.images, y: mnist.test.labels})

**The above line of codes generating the following output-**



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.

sess = tf.InteractiveSession()

# Installation of TensorFlow Through pip

In this tutorial, we will describe that how to install TensorFlow in Windows 10.

**We can download TensorFlow in our system in 2 ways:**

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

### 1. Through pip

So, firstly we have to install and set-up anaconda in our system through pip.

The following are the requirement for TensorFlow to work on our computer.

64.2M

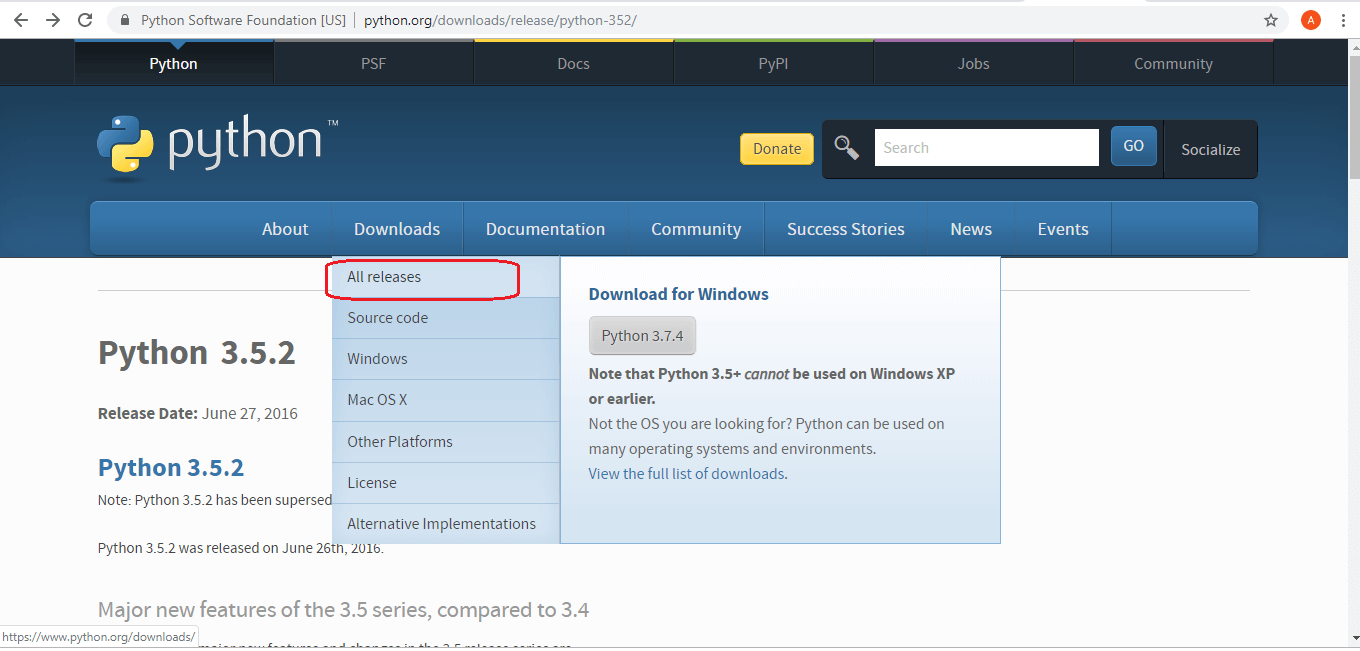
1.3K

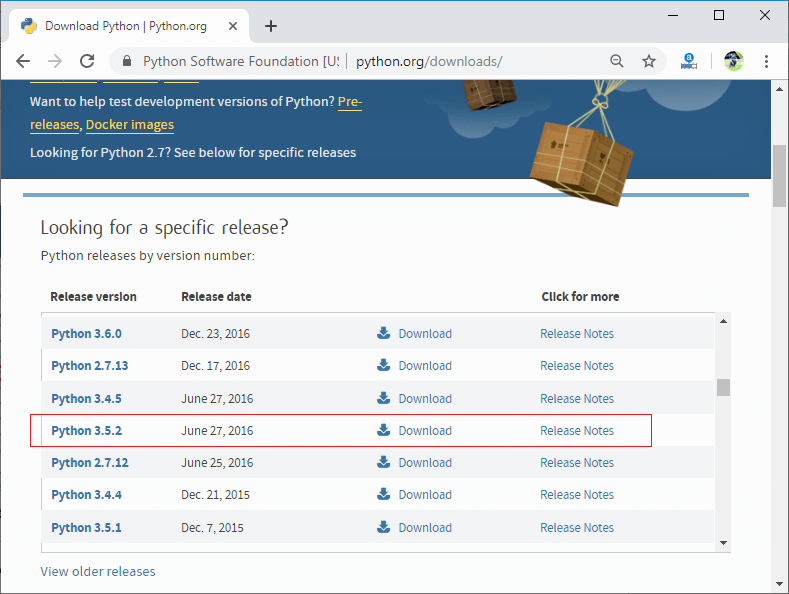
OOPs Concepts in Java

**Next**

**Stay**

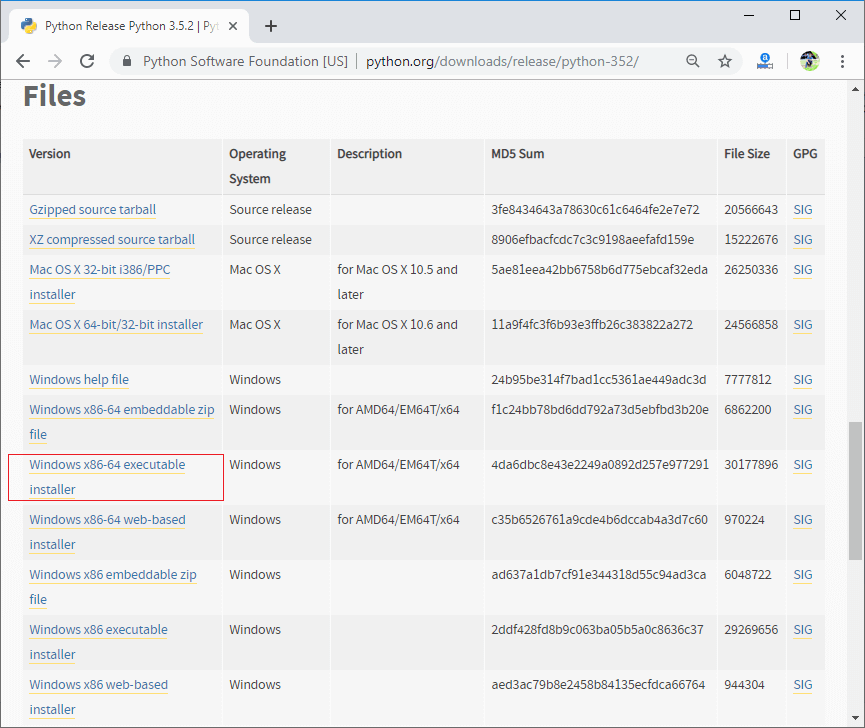
* TensorFlow has only supported 64-bit Python **3.5.x** or Python **3.6.x** on Windows
* When we download the Python **3.5.x** version, it comes with the **pip3** package manager. (Which is the program that we are going to need for our users to install TensorFlow on Windows).

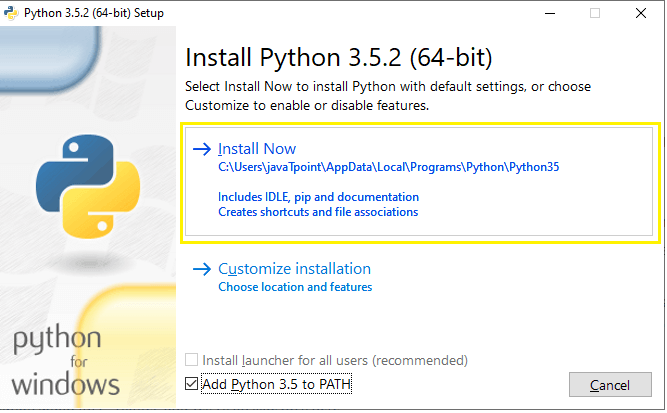
**Step 1:** Download Python from the below link:  <https://www.python.org/downloads/release/python-352/>



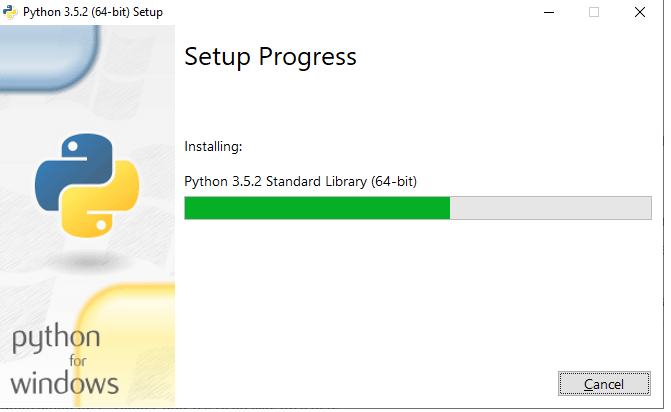
**After that,**

**Step 3:** We will be brought to another page, where we will need to select either the **x86-64** or **amd64 installer** to install Python.

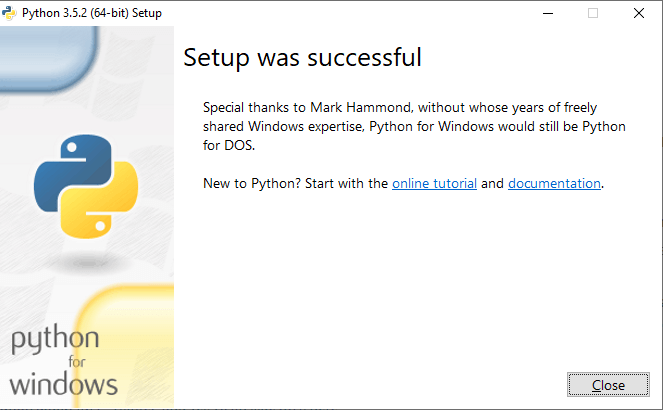




Now, Python is installing successfully.



**Step 4:** For this tutorial, I'll be choosing to Add **Python 3.5 to PATH**.



**Step 5:** Now, we will be able to see the message "**Set-up was successful**." A way to confirm that it hs installed successfully is to open your **Command Prompt** and check the version.

## What is pip?

**pip** is known as a **package management system** which is used to install and manage the software package, which is written in Python or any other languages. pip is used to download, search, install, uninstall, and manage the 3rd party python package. (pip3 is the latest version of it which comes with new Python 3.5.x version that we had just downloaded)

### Installing our TensorFlow

Once we have downloaded the latest version of Python, we can now put our finishing touches by installing our **TensorFlow**.

**Step 1:** To install TensorFlow, start the terminal. Make sure that we run the cmd as an administrator.

**If we do not know how to run your cmd as an administrator**

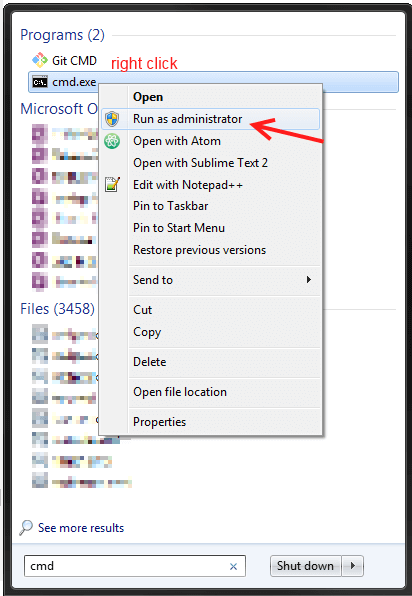
Here's how we can run in our cmd as an administrator.

Open the Start menu, search for **cmd**, and then right-click on it and **Run as an administrator**.

Open the Start menu, search for **cmd**, and then right-click on it and **Run as an administrator**.

**Step 2:** Once we are done with that, then we have to write the command in **command prompt** for finish installing Tensorflow in our Windows.

Enter this command:



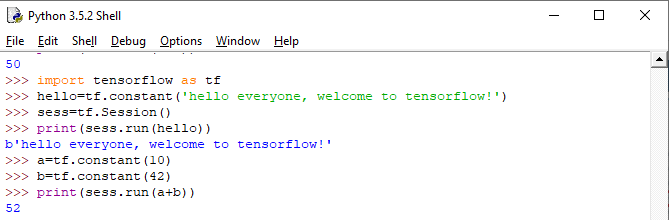
C:\pip3 install -upgrade tensorflow

Now, TensorFlow is successfully installed in our system.

### Testing our TensorFlow

Here, we try and prove whether our new TensorFlow works smoothly without any problems.

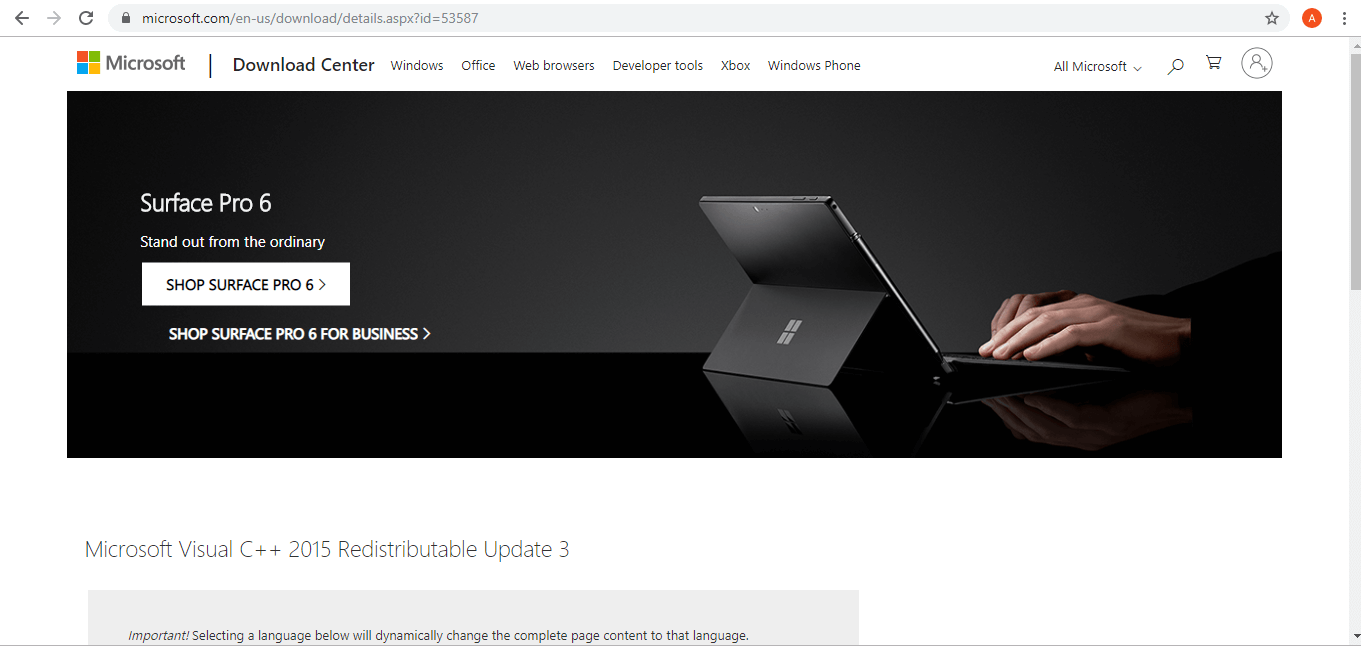
Below is an example that you can write to the test.



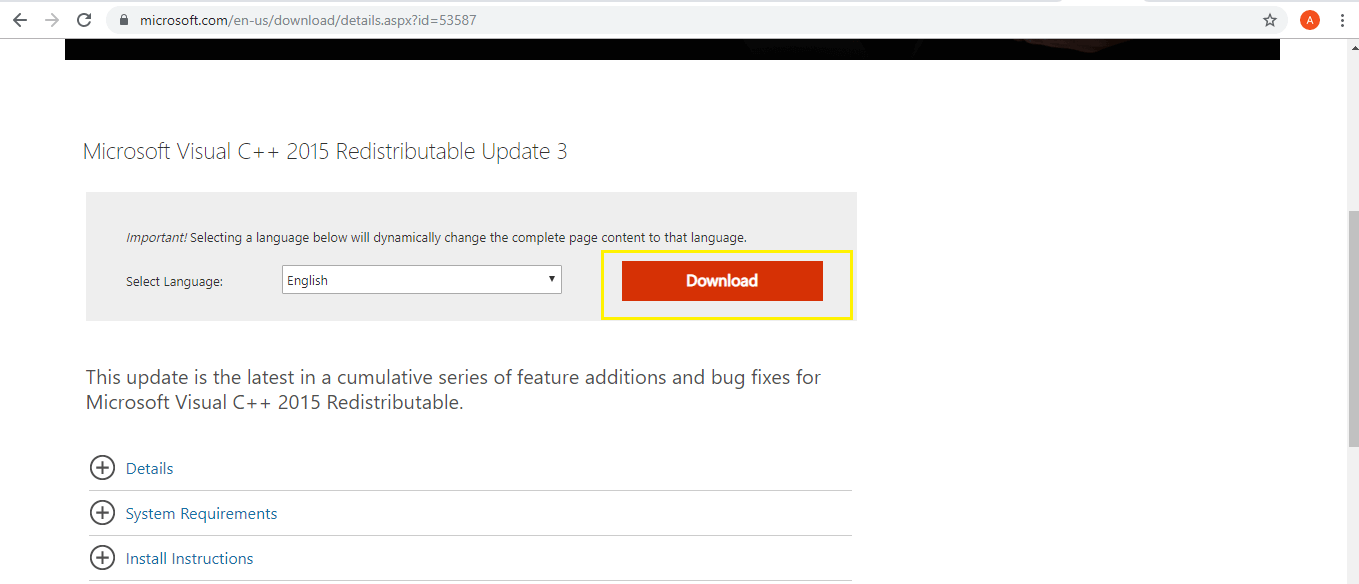
**TensorFlow is successfully working now.**

Otherwise, If we are getting any problem to run the program, then we have to install **Microsoft Visual C++ 2015**. And set up in our system then the TensorFlow will be run in the system.

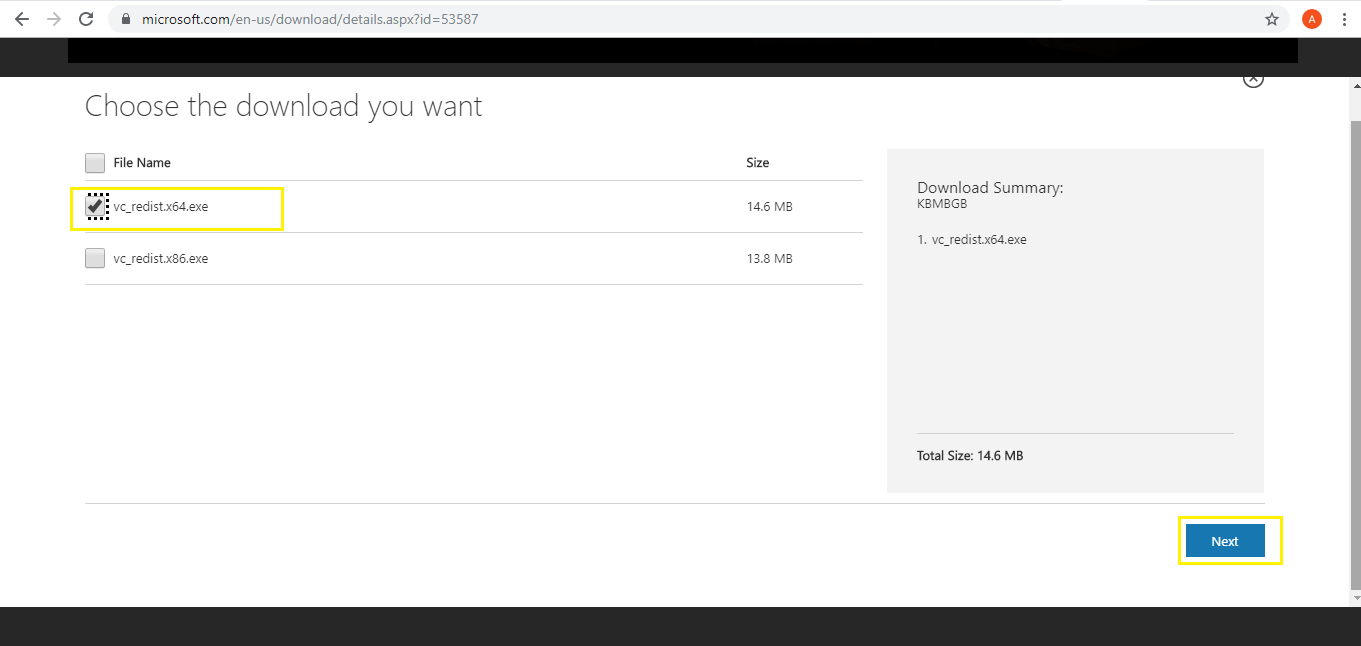
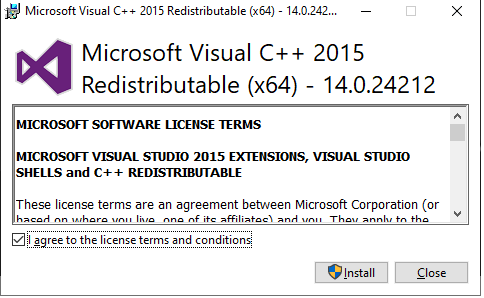
We can download **Microsoft Visual C++ 2015** from the below link: <https://www.microsoft.com/en-us/download/confirmation.aspx?id=53587>



We can download from here.



Choose the **vc\_redist.x64.exe** on the page and click on "**Next"** after that it will be downloaded.

At last, it will successfully installed in our system.

We will read **conda install TensorFlow** in our next tutorial.

**How is data loaded with TensorFlow?**

In memory data  
  
For any small CSV dataset the simplest way to train a TensorFlow model on it is to **load it into memory as a pandas Dataframe or a NumPy array**. A relatively simple example is the abalone dataset. The dataset is small. All the input features are all limited-range floating point values.

**Data preprocessing for ML using TensorFlow Transform**

1. Run Notebook 1.
2. Overview of the pipeline.
3. Read raw training data from BigQuery.
4. Transform raw training data.
5. Write transformed training data.
6. Read, transform, and write evaluation data.
7. Save the graph.