Graphic Era Hill University

MACHINE LEARNING USING TENSORFLOW

Deepankar Sharma
BCA(5)

C O N T E N T S

TENSORFLOW

TensorFlow is a free and open-source software library for machine learning and artificial intelligence. It can be used across a range of tasks but has a particular focus on training and inference of deep neural networks.

Fun Fact:
TensorFlow is used by many internal Google products and teams including: Search, Gmail, Translate, Maps, Android, Photos, Speech, YouTube, and Play.

```
import tensorflow as tf
print("TensorFlow version:", tf.__version__)
```

ADVANTAGES OF TENSORFLOW

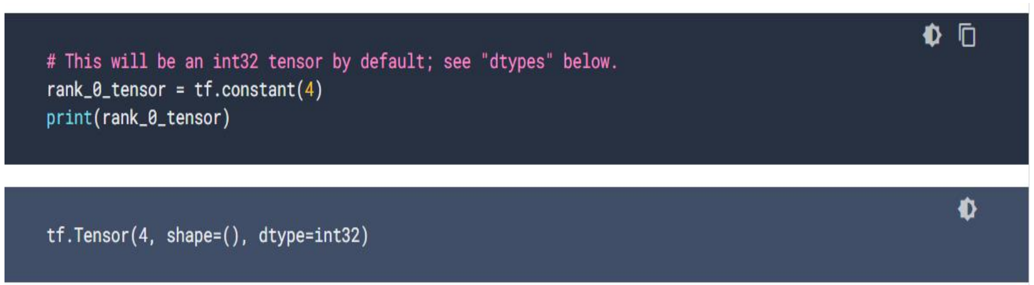
- TensorFlow has better computational graph visualizations which are inherent when compared to other libraries like Torch and Theano. Google backs it. And has the advantages of seamless performance, quick updates, and frequent new releases with new features.

- This library is designed and updated by Google, so needless to say, and it has come a far way since its initial release.

- The TensorFlow architecture uses TPU which makes computation faster than CPU and GPU. Models which are built over TPU can be easily deployed over clouds and works faster compared to the other two.

TENSORS

Tensors are multi-dimensional arrays with a uniform type (called dtype).
All tensors are immutable like Python numbers and strings: you can never update the contents of a tensor, only create a new one.

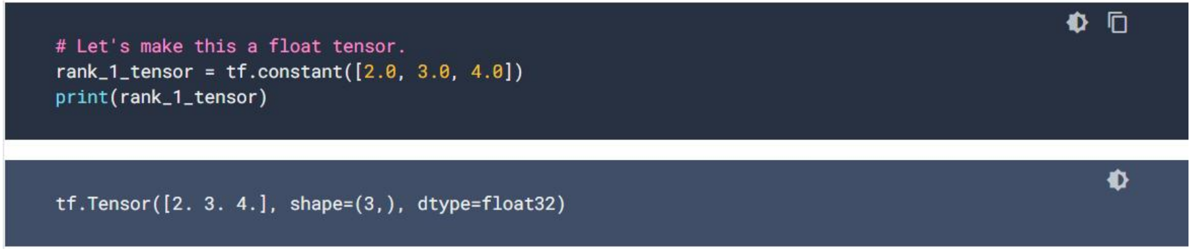


```
# This will be an int32 tensor by default; see "dtypes" below.
rank_0_tensor = tf.constant(4)
print(rank_0_tensor)
```

```
tf.Tensor(4, shape=(), dtype=int32)
```

1D-TENSORS

A "vector" or "rank-1" tensor is like a list of values. A vector has one axis:

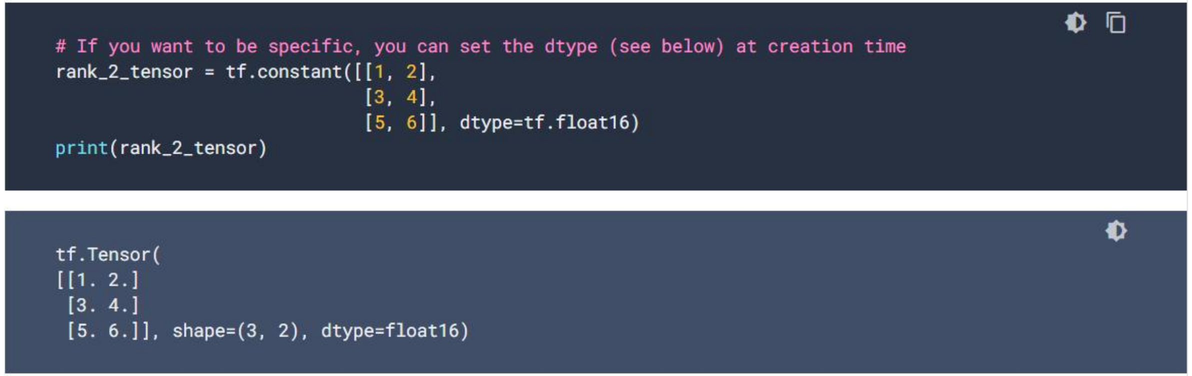


```
# Let's make this a float tensor.
rank_1_tensor = tf.constant([2.0, 3.0, 4.0])
print(rank_1_tensor)
```

```
tf.Tensor([2. 3. 4.], shape=(3,), dtype=float32)
```

2D-TENSORS

A "matrix" or "rank-2" tensor has two axes:

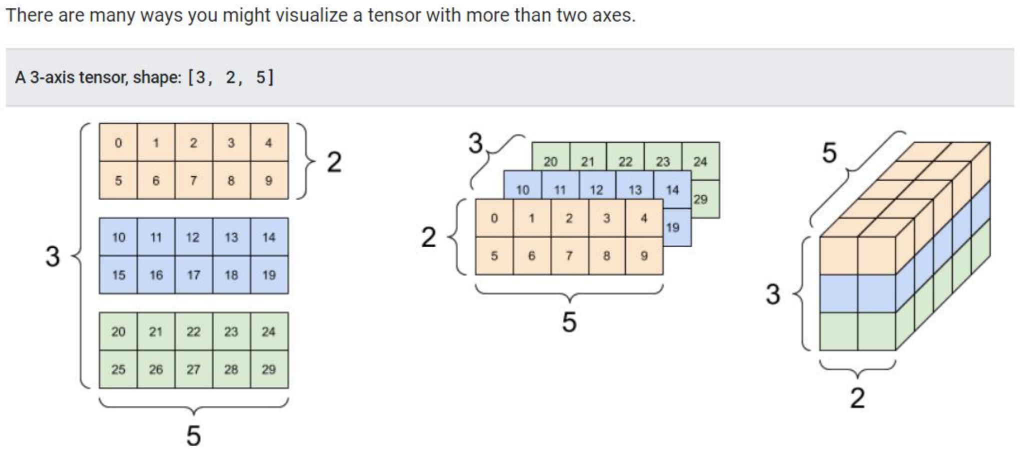


```
# If you want to be specific, you can set the dtype (see below) at creation time
rank_2_tensor = tf.constant([[1, 2],
                             [3, 4],
                             [5, 6]], dtype=tf.float16)
print(rank_2_tensor)
```

```
tf.Tensor(
[[1. 2.]
 [3. 4.]
 [5. 6.]], shape=(3, 2), dtype=float16)
```

N-D-TENSORS

Tensors may have more axes; here is a tensor with three axes:



There are many ways you might visualize a tensor with more than two axes.

A 3-axis tensor, shape: [3, 2, 5]

KERAS

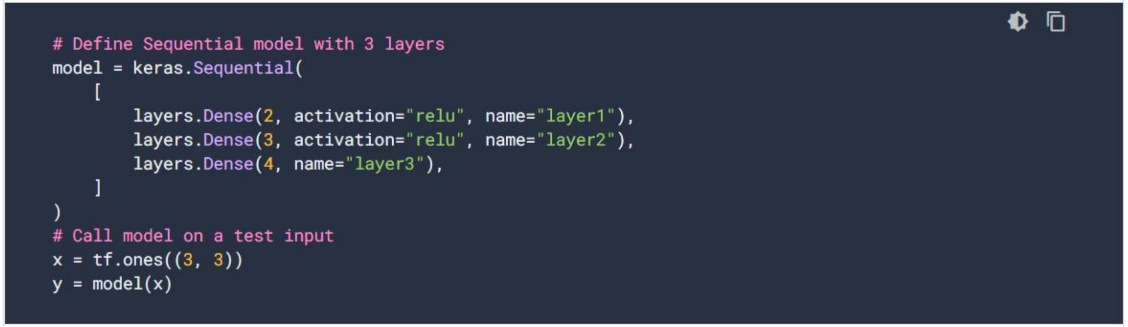
Keras is an Open Source Neural Network library written in Python that runs on top of Theano or TensorFlow. It is designed to be modular, fast and easy to use. Keras is high-level API wrapper for the low-level API, capable of running on top of TensorFlow, CNTK, or Theano.

Together with TensorFlow 2, Keras has more adoption than any other deep learning solution -- in every vertical. You are already constantly interacting with features built with Keras -- it is in use at Netflix, Uber.

```
import tensorflow as tf
from tensorflow import keras
```

KERAS SEQUENTIAL API

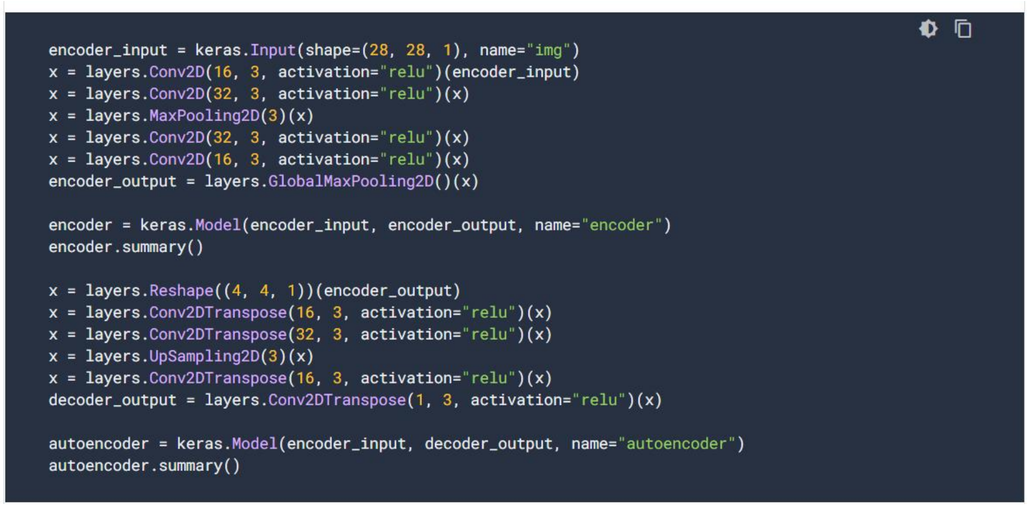
A Sequential model is appropriate for a plain stack of layers where each layer has exactly one input tensor and one output tensor.



```
# Define Sequential model with 3 layers
model = keras.Sequential(
    [
        layers.Dense(2, activation="relu", name="layer1"),
        layers.Dense(3, activation="relu", name="layer2"),
        layers.Dense(4, name="layer3"),
    ]
)
# Call model on a test input
x = tf.ones((3, 3))
y = model(x)
```

KERAS FUNCTIONAL API

The Keras functional API is a way to create models that are more flexible than the tf.keras.Sequential API. The functional API can handle models with non-linear topology, shared layers, and even multiple inputs or outputs.The main idea is that a deep learning model is usually a directed acyclic graph (DAG) of layers. So the functional API is a way to build graphs of layers.



```
encoder_input = keras.Input(shape=(28, 28, 1), name="img")
x = layers.Conv2D(16, 3, activation="relu")(encoder_input)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.MaxPooling2D(3)(x)
x = layers.Conv2D(32, 3, activation="relu")(x)
x = layers.Conv2D(16, 3, activation="relu")(x)
encoder_output = layers.GlobalMaxPooling2D()(x)

encoder = keras.Model(encoder_input, encoder_output, name="encoder")
encoder.summary()

x = layers.Reshape((4, 4, 1))(encoder_output)
x = layers.Conv2DTranspose(16, 3, activation="relu")(x)
x = layers.Conv2DTranspose(32, 3, activation="relu")(x)
x = layers.UpSampling2D(3)(x)
x = layers.Conv2DTranspose(16, 3, activation="relu")(x)
decoder_output = layers.Conv2DTranspose(1, 3, activation="relu")(x)

autoencoder = keras.Model(encoder_input, decoder_output, name="autoencoder")
autoencoder.summary()
```

T H A N K S