Features in TensorFlow 2.0

TensorFlow 2.0 comes with new features for building machine learning models. Some of these new features include

- A more pythonic feel to model design and debugging with eager execution as the de facto execution mode.
- Eager execution enables instant evaluation of TensorFlow operations. This is opposed to previous versions of Tensorflow where we first construct a computational graph and then execute it in a session.
- Using tf.function to transform a Python method into high-performance TensorFlow graphs.
- Using Keras as the core high-level API for model design.
- Using FeatureColumns to parse data as input into Keras models.
- The ease of training on distributed architectures and devices.

To install and work with TensorFlow 2.0 on Google Colab, run

```
!pip install -q tensorflow==2.0.0-beta0
```

The GCP Deep Learning VM has images with TensorFlow 2.0 pre-configured.

A Simple TensorFlow Program

Let's start by building a simple TF program. Here, we will build a graph to find the roots of the quadratic expression $x^2 + 3x - 4 = 0$.

```
# import tensorflow
import tensorflow as tf

# Quadratic expression: x**2 + 3x - 4 = 0.
a = tf.constant(1.0)
b = tf.constant(3.0)
c = tf.constant(-4.0)
```

```
print(a)
print(b)
print(c)
```

```
'Output':
tf.Tensor(1.0, shape=(), dtype=float32)
tf.Tensor(3.0, shape=(), dtype=float32)
tf.Tensor(-4.0, shape=(), dtype=float32)
```

tf.constant() is a Tensor for storing a constant type. Now let's calculate the roots of the expression.

```
x1 = (-b + tf.math.sqrt(b**2 - (4*a*c))) / 2**a
x2 = (-b - tf.math.sqrt(b**2 - (4*a*c))) / 2**a
```

```
roots = (x1, x2)
print(roots)
```

```
'Output':
(<tf.Tensor: id=163, shape=(), dtype=float32, numpy=1.0>, <tf.Tensor:
id=175, shape=(), dtype=float32, numpy=-4.0>)
```

TensorFlow 2.0 is eager-first; this implies that operations are executed immediately after they are defined, just like regular python code.

Building Efficient Input Pipelines with the Dataset API

The Dataset API **'tf.data'** offers an efficient mechanism for building robust input pipelines for passing data into a TensorFlow program. This section uses the Boston housing dataset to illustrate working with the Dataset API methods for building data input pipelines in TensorFlow.

```
# import packages
import tensorflow as tf
from tensorflow.keras.datasets import boston_housing
```

```
# load dataset and split in train and test sets
(X_train, y_train), (X_test, y_test) = boston_housing.load_data()
```