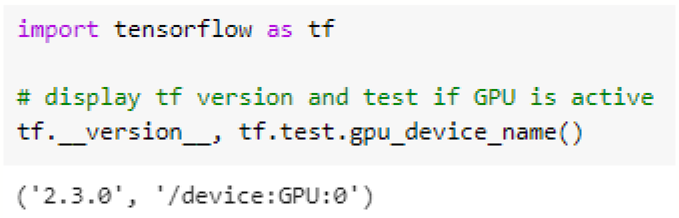
Assignment 4 (key)

1. Enable GPU:



```python
import tensorflow as tf

# display tf version and test if GPU is active
tf.__version__, tf.test.gpu_device_name()
```

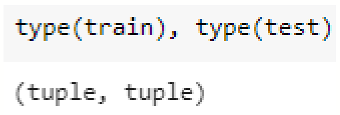
```
('2.3.0', '/device:GPU:0')
```

2. Load Keras **MNIST** dataset (https://keras.io/api/datasets/mnist/):

```python
train, test = tf.keras.datasets.mnist.load_data(path='mnist.npz')
```

3. Inspect data.

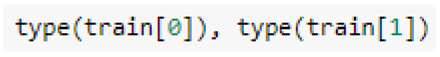
a. display datatypes for train and test sets:



```python
type(train), type(test)
```

```
(tuple, tuple)
```

b. display datatypes for image and label data for train set:



```python
type(train[0]), type(train[1])
```

(note: data are numpy arrays)

c. display shapes for train and test sets:

```
br = '\n'
print ('train data:', br)
print (train[0].shape)
print (train[1].shape, br)
print ('test data:', br)
print (test[0].shape)
print (test[1].shape)
```

```
train data:

(60000, 28, 28)
(60000,)

test data:

(10000, 28, 28)
(10000,)
```

4. Build the input pipeline.

a. create variables to hold image and label data:

```
train_images, train_labels = train
test_images, test_labels = test
```

b. scale feature images for train and test sets:

```
train_img = train_images / 255
test_img = test_images / 255
```

(note: use different variables names to avoid dividing more than once)

c. prepare data for TensorFlow consumption:

```
train_k = tf.data.Dataset.from_tensor_slices(
    (train_img, train_labels))
test_k = tf.data.Dataset.from_tensor_slices(
    (test_img, test_labels))
```

d. shuffle, batch, and prefetch:

```
BATCH_SIZE = 128
SHUFFLE_BUFFER_SIZE = 1000

train_kd = train_k.shuffle(
    SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE).prefetch(1)
test_kd = test_k.batch(BATCH_SIZE).prefetch(1)
```

5. Create the model.

   a. import libraries:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Flatten, Dropout
```

   b. clear any previous models:

```
tf.keras.backend.clear_session()
```

   c. get input shape:

```
in_shape = train[0].shape[1:]
in_shape
```

   d. build the model:

```
model = Sequential([
  Flatten(input_shape=in_shape),
  Dense(512, activation='relu'),
  Dropout(0.5),
  Dense(10, activation='softmax')
])
```

   e. display model summary:

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
flatten_1 (Flatten)          (None, 784)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 512)               401920    
_________________________________________________________________
dropout_1 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_3 (Dense)              (None, 10)                5130      
=================================================================
Total params: 407,050
Trainable params: 407,050
Non-trainable params: 0
_________________________________________________________________
```

6. Train model:

   a. compile:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

b. train for 10 epochs:

```
epochs = 10
history = model.fit(train_kd, epochs=epochs, verbose=1,
                    validation_data=test_kd)
```

7. Load sklearn **wine** dataset (https://scikit-learn.org/stable/modules/generated/sklearn.datasets.load_wine.html):

```
from sklearn.datasets import load_wine

data = load_wine()
```

8. Inspect wine data.

a. get keys:

```
data.keys()
```

```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

b. get shape of feature data and target data:

```
data.data.shape, data.target.shape
```

```
((178, 13), (178,))
```

c. get feature names:

```
data.feature_names
```

```
['alcohol',
 'malic_acid',
 'ash',
 'alcalinity_of_ash',
 'magnesium',
 'total_phenols',
 'flavanoids',
 'nonflavanoid_phenols',
 'proanthocyanins',
 'color_intensity',
 'hue',
 'od280/od315_of_diluted_wines',
 'proline']
```

d. get target names:

```
data.target_names
```

```
array(['class_0', 'class_1', 'class_2'], dtype='<U7')
```

9. Build the input pipeline.

a. assign data to variables for convenience:

```
X, y = data.data, data.target
```

b. split data into train and test sets:

```
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(
    X, y, test_size=0.25, random_state=0)
```

c. scale data (use **StandardScaler**):

```
from sklearn.preprocessing import StandardScaler

scaler = StandardScaler()
X_train_std = scaler.fit_transform(X_train)
X_test_std = scaler.fit_transform(X_test)
```

d. prepare data for TensorFlow consumption:

```
train_wine = tf.data.Dataset.from_tensor_slices(
    (X_train_std, y_train))
test_wine = tf.data.Dataset.from_tensor_slices(
    (X_test_std, y_test))
```

e. shuffle, batch, and prefetch:

```
BATCH_SIZE = 16
SHUFFLE_BUFFER_SIZE = 100

train_wine_ds = train_wine.shuffle(
    SHUFFLE_BUFFER_SIZE).batch(BATCH_SIZE).prefetch(1)
test_wine_ds = test_wine.batch(BATCH_SIZE).prefetch(1)
train_wine_ds, test_wine_ds
```

10. Train.

a. clear previous models:

```
import numpy as np

tf.keras.backend.clear_session()
np.random.seed(0)
tf.random.set_seed(0)
```

b. get input shape:

```
in_shape = X_train.shape[1:]
```

(note: this wine dataset has 13 features)

c. build model:

```
model = Sequential([
  Dense(30, activation='relu', input_shape=in_shape),
  Dense(11, activation='softmax')
])
```

d. compile:

```
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
```

e. train:

```
history = model.fit(train_wine_ds, epochs=10,
                    validation_data=test_wine_ds)
```