< Deep Learning - PART1 TF2 Basics >

Ch 1. Workshop - TensorFlow 2.0 Installation & Testing

2021/10/01

`<< Installation of TF 2.0 with Anaconda3 >>`

- **First, install `Anaconda 3 for Windows/macOS/Linux` from [https://www.anaconda.com/distribution/ (https://www.anaconda.com/distribution/)](https://www.anaconda.com/distribution/)**

- **Next, run `TensorFlow 2 (for CPU)` Setup on `Anaconda Prompt :`**

```
conda install tensorflow
```

[Reference]:

- TensorFlow.org, **"Install TensorFlow 2"** [https://www.tensorflow.org/install (https://www.tensorflow.org/install)](https://www.tensorflow.org/install)
- 海萨, **"Anaconda 安装tensorflow 2.0 报错解决办法"** [https://zhuanlan.zhihu.com/p/62031082 (https://zhuanlan.zhihu.com/p/62031082)](https://zhuanlan.zhihu.com/p/62031082)
- TensorFlow.org, **"Get Started with TensorFlow"** [https://www.tensorflow.org/tutorials/#get-started-with-tensorflow (https://www.tensorflow.org/tutorials/#get-started-with-tensorflow)](https://www.tensorflow.org/tutorials/#get-started-with-tensorflow)

[Content]

1. Testing TF 2.0

In [1]:

```python
import tensorflow as tf
print(tf.__version__)
```

```
2.4.1
```

In [2]:

```
# --------------------------------------------
# The following code is adopted from
# Tutorial document of TensorFlow.org
# for testing TensorFlow 2.0 setup:
#
# "Get Started with TensorFlow"
#  https://www.tensorflow.org/tutorials/#get-started-with-tensorflow
# --------------------------------------------

mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0

model = tf.keras.models.Sequential([
  tf.keras.layers.Flatten(input_shape=(28, 28)),
  tf.keras.layers.Dense(512, activation=tf.nn.relu),
  tf.keras.layers.Dropout(0.2),
  tf.keras.layers.Dense(10, activation=tf.nn.softmax)
])
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])

model.fit(x_train, y_train, epochs=5)
model.evaluate(x_test, y_test, verbose=0) # verbose: Verbosity mode. 0=sile
```

```
Epoch 1/5
1875/1875 [==============================] - 18s 9ms/step - los
s: 0.3670 - accuracy: 0.8910
Epoch 2/5
1875/1875 [==============================] - 15s 8ms/step - los
s: 0.0985 - accuracy: 0.9696
Epoch 3/5
1875/1875 [==============================] - 15s 8ms/step - los
s: 0.0697 - accuracy: 0.9785
Epoch 4/5
1875/1875 [==============================] - 15s 8ms/step - los
s: 0.0506 - accuracy: 0.9837
Epoch 5/5
1875/1875 [==============================] - 14s 8ms/step - los
s: 0.0406 - accuracy: 0.9865
```

Out[2]:

```
[0.07084144651889801, 0.9793000221252441]
```

---

2. How to run TensorFlow 1.x code on TF 2.0

- **It is still possible to run 1.X code, unmodified (except for contrib), in TensorFlow 2.0:**

```
import tensorflow.compat.v1 as tf

tf.disable_v2_behavior()
```

[NOTE]:

- **More detailed information regarding "** `Migrate your TensorFlow 1 code to TensorFlow 2` **" can be found here:** [https://www.tensorflow.org/guide/migrate (https://www.tensorflow.org/guide/migrate)](https://www.tensorflow.org/guide/migrate)

In [7]:

```
import tensorflow.compat.v1 as tf
tf.disable_v2_behavior()

print(tf.__version__)
```

```
WARNING:tensorflow:From C:\Users\USER\Anaconda3\lib\site-package
s\tensorflow_core\python\compat\v2_compat.py:65: disable_resourc
e_variables (from tensorflow.python.ops.variable_scope) is depre
cated and will be removed in a future version.
Instructions for updating:
non-resource variables are not supported in the long term
2.0.0
```

The following code is adopted for testing TensorFlow 2.0 setup from the reference below:

- Tom Hope, Yehezkel S. Resheff, and Itay Lieder, "**Learning TensorFlow : A Guide to Building Deep Learning Systems**," Chapter 2 & 4, O'Reilly, 2017. [https://goo.gl/iEmehh (https://goo.gl/iEmehh)](https://goo.gl/iEmehh)
- Download the code from GitHub : [https://github.com/gigwegbe/Learning-TensorFlow (https://github.com/gigwegbe/Learning-TensorFlow)](https://github.com/gigwegbe/Learning-TensorFlow)

Loading the MNIST dataset (from TensorFlow 2.0)

In [8]:

```
mnist = tf.keras.datasets.mnist

(x_train, y_train),(x_test, y_test) = mnist.load_data()
x_train, x_test = x_train / 255.0, x_test / 255.0
```

In [9]:

```python
import numpy as np

x_train = np.array([x_train[i].flatten() for i in range(len(x_train))])
x_train.shape
```

Out[9]:

```
(60000, 784)
```

In [10]:

```python
x_test = np.array([x_test[i].flatten() for i in range(len(x_test))])
x_test.shape
```

Out[10]:

```
(10000, 784)
```

In [11]:

```python
y_train[0], y_test[0]
```

Out[11]:

```
(5, 7)
```

In [12]:

```python
def one_hot(vec, vals=10):
    n = len(vec)
    out = np.zeros((n, vals))
    out[range(n), vec] = 1
    return out
```

In [13]:

```python
y_train = one_hot(y_train)
y_train[0]
```

Out[13]:

```
array([0., 0., 0., 0., 0., 1., 0., 0., 0., 0.])
```

In [14]:

```
y_test = one_hot(y_test)
y_test[0]
```

Out[14]:

```
array([0., 0., 0., 0., 0., 0., 0., 1., 0., 0.])
```

Building a Computation Graph on TF 1.x

In [15]:

```
# Each Input Image, X, with 28*28 (= 784) pixels
X = tf.placeholder(tf.float32, [None, 784])

# y_true : the training labeled dataset
y_true = tf.placeholder(tf.float32,[None, 10])
```

In [16]:

```
# Initializing Weights & Biases for Nodes in All Hidden Layers
def weight_variable(shape):
    initial = tf.truncated_normal(shape, stddev=0.1)
    return tf.Variable(initial)

def bias_variable(shape):
    initial = tf.constant(0.1, shape=shape)
    return tf.Variable(initial)
```

In [17]:

```
# Building a Fully-Connected Deep Network
def full_layer(inputs, size):
    in_size = int(inputs.get_shape()[1])
    W = weight_variable([in_size, size])
    b = bias_variable([size])
    return tf.add(tf.matmul(inputs, W), b)
```

In [18]:

```python
keep_prob = tf.placeholder(tf.float32)

# < Hidden Layer 1 >
layer_1_drop = tf.nn.dropout(X, keep_prob=keep_prob)
#   Activation Function : ReLU
layer_1_Outputs = tf.nn.relu(full_layer(layer_1_drop, 256))

# < Hidden Layer 2 >
layer_2_drop = tf.nn.dropout(layer_1_Outputs, keep_prob=keep_prob)
#   Activation Function : ReLU
layer_2_Outputs = tf.nn.relu(full_layer(layer_2_drop, 128))

# < Output Layer >
output_drop = tf.nn.dropout(layer_2_Outputs, keep_prob=keep_prob)
# Without Activation Function
y_pred = full_layer(output_drop, 10)
```

```
WARNING:tensorflow:From <ipython-input-18-931f684597d4>:4: calli
ng dropout (from tensorflow.python.ops.nn_ops) with keep_prob is
deprecated and will be removed in a future version.
Instructions for updating:
Please use `rate` instead of `keep_prob`. Rate should be set to
`rate = 1 - keep_prob`.
```

In [19]:

```python
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(log
gd_step = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

correct_mask = tf.equal(tf.argmax(y_pred, 1), tf.argmax(y_true, 1))
accuracy = tf.reduce_mean(tf.cast(correct_mask, tf.float32))
```

```
WARNING:tensorflow:From <ipython-input-19-315e39f82a5d>:1: softm
ax_cross_entropy_with_logits (from tensorflow.python.ops.nn_ops)
is deprecated and will be removed in a future version.
Instructions for updating:

Future major versions of TensorFlow will allow gradients to flow
into the labels input on backprop by default.

See `tf.nn.softmax_cross_entropy_with_logits_v2`.
```

Launching the Computation Graph on TF 1.x

In [20]:

```
def next_batch(i, images, labels, batch_size):
    i_start = (i * batch_size) % len(images)
    x, y = images[i_start : i_start+batch_size], labels[i_start : i_start+b
    return x, y
```

In [21]:

```python
NUM_STEPS = 8000
MINIBATCH_SIZE = 100
Display_Step = 1000

with tf.Session() as sess:
    sess.run(tf.global_variables_initializer())

    for i in range(NUM_STEPS):
        batch_xs, batch_ys = next_batch(i, x_train, y_train, MINIBATCH_SIZI
        sess.run(gd_step, feed_dict ={X: batch_xs,
                                      y_true: batch_ys,
                                      keep_prob: 0.5})

        if (i+1) % Display_Step == 0:
            # Calculate batch loss and accuracy
            loss_temp, accu_temp = sess.run([cross_entropy, accuracy],
                                            feed_dict={X: batch_xs,
                                                       y_true: batch_ys,
                                                       keep_prob: 1.0})
            print("Step " + str(i+1).rjust(4) + \
                  " : Loss = " + "{:.4f}".format(loss_temp) + \
                  ", Accuracy = " + "{:.3f}".format(accu_temp))

    print("\n Computing the test accuracy ... ", end = " ")

    ##  ------------------------------------------------------------------
    ##  Split the test procedure into 10 blocks of 1,000 images each.
    ##  Doing this is important mostly for much larger datasets.
    ##  ------------------------------------------------------------------
    ##  mnist.test.images.shape : (10000, 784)
    X_test = x_test.reshape(10, 1000, 784)
    ##  mnist.test.labels.shape : (10000, 10)
    Y_test = y_test.reshape(10, 1000, 10)

    test_loss = np.mean([sess.run(cross_entropy,
                                  feed_dict={X: X_test[i],
                                             y_true: Y_test[i],
                                             keep_prob: 1.0})
                                  for i in range(10)])
    test_accu = np.mean([sess.run(accuracy,
                                  feed_dict={X: X_test[i],
                                             y_true: Y_test[i],
                                             keep_prob: 1.0})
                                  for i in range(10)])
    print("\n [ Test  Accuracy ] : {}".format(test_accu) +
      "\n [ Test Loss Score ] : {}".format(test_loss))
```

```
Step 1000 : Loss = 0.2206, Accuracy = 0.960
Step 2000 : Loss = 0.1248, Accuracy = 0.970
Step 3000 : Loss = 0.1113, Accuracy = 0.990
Step 4000 : Loss = 0.0721, Accuracy = 0.980
Step 5000 : Loss = 0.0949, Accuracy = 0.960
Step 6000 : Loss = 0.1253, Accuracy = 0.990
Step 7000 : Loss = 0.0690, Accuracy = 0.990
Step 8000 : Loss = 0.0743, Accuracy = 0.980
```

```
Computing the test accuracy ...
[ Test  Accuracy ] : 0.9604999423027039
[ Test Loss Score ] : 0.12836746871471405
```