< Deep Learning - PART1 TF2 Basics >

Ch 3. TensorBoard - ML Workflow Visualization

2021/10/01

[Reference]

> 1. TensorFlow Core - Tutorials : **`Get started with TensorBoard`** .
> https://www.tensorflow.org/tensorboard/get_started (https://www.tensorflow.org/tensorboard/get_started)
>
> 2. **`Hands-on TensorBoard (TensorFlow Dev Summit 2017)`**
>
> - Youtube: https://youtu.be/eBbEDRsCmv4 (https://youtu.be/eBbEDRsCmv4)
> - Code: https://goo.gl/ZwGnPE (https://goo.gl/ZwGnPE)

> 3. **TensorBoard on Colab -** **`TensorBoard.dev`** https://tensorboard.dev/ (https://tensorboard.dev/)
>
> - [Example Code]: https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/doc (https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/do

- `TensorBoard` is a tool for providing the measurements and visualizations needed during the machine learning workflow.
- It enables tracking experiment metrics like loss and accuracy, visualizing the model graph, projecting embeddings to a lower dimensional space, and much more.

< Content >

1. TensorBoard on `localhost`

In [1]:

```
import tensorflow as tf
import datetime

print(tf.__version__)
```

```
2.4.1
```

Loading the MNIST dataset

In [2]:

```
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
```

Building the Keras Neural Network Model with dropout

In [3]:

```
model = tf.keras.models.Sequential([
    tf.keras.layers.Flatten(input_shape=(28, 28)),
    tf.keras.layers.Dense(512, activation='relu'),
    tf.keras.layers.Dropout(0.2),
    tf.keras.layers.Dense(10, activation='softmax')
  ])
```

1.1 TensorBoard with Keras `Model.fit()`

> - When training with Keras's **`Model.fit()`**, adding the **`tf.keras.callback.TensorBoard`** callback ensures that logs are created and stored.

- Additionally, enable histogram computation every epoch with **`histogram_freq=1`** (this is off by default.)

Place the logs in a `timestamped subdirectory` to allow easy selection of different training runs:

In [4]:

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

# Building Timestamped Directories For Windows :
log_dir="logs\\fit\\" + datetime.datetime.now().strftime('%Y%m%d_%H%M%S')

# Building Timestamped Directories For Mac/Linux :
# log_dir="logs/fit/" + datetime.strftime(datetime.now(), '%Y%m%d_%H%M%S')

# callback method for Tensorboard
tensorboard_callback = tf.keras.callbacks.TensorBoard(log_dir=log_dir,
                                                      histogram_freq=1)

model.fit(x=x_train,
          y=y_train,
          epochs=5,
          validation_data=(x_test, y_test),
          callbacks=[tensorboard_callback])
```

```
Epoch 1/5
1875/1875 [==============================] - 23s 11ms/step - los
s: 0.3651 - accuracy: 0.8932 - val_loss: 0.1045 - val_accuracy:
0.9686
Epoch 2/5
1875/1875 [==============================] - 18s 9ms/step - los
s: 0.1026 - accuracy: 0.9687 - val_loss: 0.0810 - val_accuracy:
0.9733
Epoch 3/5
1875/1875 [==============================] - 17s 9ms/step - los
s: 0.0682 - accuracy: 0.9783 - val_loss: 0.0740 - val_accuracy:
0.9763
Epoch 4/5
1875/1875 [==============================] - 16s 9ms/step - los
s: 0.0485 - accuracy: 0.9844 - val_loss: 0.0641 - val_accuracy:
0.9801
Epoch 5/5
1875/1875 [==============================] - 17s 9ms/step - los
s: 0.0409 - accuracy: 0.9862 - val_loss: 0.0691 - val_accuracy:
0.9788
```

Out[4]:

```
<tensorflow.python.keras.callbacks.History at 0x1dd1e103e20>
```

To run TensorBoard, run the following command on Anaconda (Powershell) Prompt :

```
tensorboard --logdir= path/to/log-directory
```

- For instance, **`tensorboard --logdir logs/fit`**

Connecting to **`http://localhost:6006`**

1.2 TensorBoard with `tf.summary`

- When training with methods such as **`tf.GradientTape()`** (https://www.tensorflow.org/api_docs/python/tf/GradientTape (https://www.tensorflow.org/api_docs/python/tf/GradientTape)), use **`tf.summary`** to log the required information.

- Use the same dataset as above, but convert it to **`tf.data.Dataset`** to take advantage of batching capabilities.
- [NOTE]: The following training code is adopted from the **advanced quickstart** (https://www.tensorflow.org/tutorials/quickstart/advanced (https://www.tensorflow.org/tutorials/quickstart/advanced)) tutorial.

In [5]:

```
train_dataset = tf.data.Dataset.from_tensor_slices((x_train, y_train))
test_dataset = tf.data.Dataset.from_tensor_slices((x_test, y_test))

train_dataset = train_dataset.shuffle(60000).batch(64)
test_dataset = test_dataset.batch(64)
```

In [6]:

```
loss_object = tf.keras.losses.SparseCategoricalCrossentropy()
optimizer = tf.keras.optimizers.Adam()
```

Create stateful metrics that can be used to accumulate values during training and logged at any point:

In [7]:

```
# Define our metrics
train_loss = tf.keras.metrics.Mean('train_loss', dtype=tf.float32)
train_accuracy = tf.keras.metrics.SparseCategoricalAccuracy('train_accuracy
test_loss = tf.keras.metrics.Mean('test_loss', dtype=tf.float32)
test_accuracy = tf.keras.metrics.SparseCategoricalAccuracy('test_accuracy'
```

Define the training and test functions:

In [8]:

```
def train_step(model, optimizer, x_train, y_train):
    with tf.GradientTape() as tape:
        predictions = model(x_train, training=True)
        loss = loss_object(y_train, predictions)
    grads = tape.gradient(loss, model.trainable_variables)
    optimizer.apply_gradients(zip(grads, model.trainable_variables))

    train_loss(loss)
    train_accuracy(y_train, predictions)

def test_step(model, x_test, y_test):
    predictions = model(x_test)
    loss = loss_object(y_test, predictions)

    test_loss(loss)
    test_accuracy(y_test, predictions)
```

Set up summary writers to write the summaries to disk in a different logs directory:

In [11]:

```
current_time = datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
train_log_dir = 'logs/gradient_tape/' + current_time + '/train'
test_log_dir = 'logs/gradient_tape/' + current_time + '/test'
train_summary_writer = tf.summary.create_file_writer(train_log_dir)
test_summary_writer = tf.summary.create_file_writer(test_log_dir)
```

Start training. Use `tf.summary.scalar()` to log metrics (loss and accuracy) during training/testing within the scope of the summary writers to write the summaries to disk. You have control over which metrics to log and how often to do it. Other `tf.summary` functions enable logging other types of data.

In [13]:

```
EPOCHS = 5

for epoch in range(EPOCHS):
    for (x_train, y_train) in train_dataset:
        train_step(model, optimizer, x_train, y_train)
    with train_summary_writer.as_default():
        tf.summary.scalar('loss', train_loss.result(), step=epoch)
        tf.summary.scalar('accuracy', train_accuracy.result(), step=epoch)

    for (x_test, y_test) in test_dataset:
        test_step(model, x_test, y_test)
    with test_summary_writer.as_default():
        tf.summary.scalar('loss', test_loss.result(), step=epoch)
        tf.summary.scalar('accuracy', test_accuracy.result(), step=epoch)

    template = 'Epoch {}, Loss: {}, Accuracy: {}, Test Loss: {}, Test Accur
    print (template.format(epoch+1,
                           train_loss.result(),
                           train_accuracy.result()*100,
                           test_loss.result(),
                           test_accuracy.result()*100))

    # Reset metrics every epoch
    train_loss.reset_states()
    test_loss.reset_states()
    train_accuracy.reset_states()
    test_accuracy.reset_states()
```

```
Epoch 1, Loss: 0.013252157717943192, Accuracy: 99.5533370971679
7, Test Loss: 0.06811141967773438, Test Accuracy: 98.32999420166
016
Epoch 2, Loss: 0.010231240652501583, Accuracy: 99.6533355712890
6, Test Loss: 0.06884602457284927, Test Accuracy: 98.22000122070
312
Epoch 3, Loss: 0.011641748249530792, Accuracy: 99.6116638183593
8, Test Loss: 0.06621949374675751, Test Accuracy: 98.40999603271
484
Epoch 4, Loss: 0.010976376011967659, Accuracy: 99.6333389282226
6, Test Loss: 0.07315755635499954, Test Accuracy: 98.29000091552
734
Epoch 5, Loss: 0.012019790709018707, Accuracy: 99.5849990844726
6, Test Loss: 0.07585285604000092, Test Accuracy: 98.25999450683
594
```

To run TensorBoard, run the following command on Anaconda (Powershell) Prompt :

```
tensorboard --logdir= path/to/log-directory
```

- For instance, **`tensorboard --logdir logs/gradient_tape`**

Connecting to **`http://localhost:6006`**

2. Tensorboard on Google `Colab`

- **TensorBoard on Colab - `TensorBoard.dev`** https://tensorboard.dev/ (https://tensorboard.dev/)
  - [Example Code]: https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/tbdev_ (https://colab.research.google.com/github/tensorflow/tensorboard/blob/master/docs/tbdev