TensorFlow II

StartOnAI Deep Learning Tutorial 4

This tutorial goes further in-depth on TensorFlow and covers some of its more advanced capabilities

1. What is TensorBoard

In machine learning it can be hard to understand how exactly a neural net is progressing and maybe there is a need for more data to be seen about it than just accuracy. Visualizing what is happening in highly complex code makes it easy to better understand what is happening and what needs to be done. TensorFlow has a visualization toolkit known as *TensorBoard* which provides various tools for experimenting with a neural net. This software has extremely useful features such as tracking and visualizing metrics such as loss and accuracy, visualizing the model graph, creating histograms of weights, biases, and other tensors over time, and much more. This becomes helpful to debug the application as tweaking the code by just looking at it would take far more time. Using TensorBoard's plethora of features identifying what needs to be fixed and where allows the developer to eliminate wasted time by writing separate programs to collect this data. Overall, TensorBoard's features offer a way to dive deeper into exactly how a neural net is changing over time.

2. Setting up for TensorBoard

In this tutorial writing code and launching TensorBoard will be done using Jupyter notebook. The following code will be similar from the TensorFlow I tutorial with some minor, but important changes.

```
%load_ext tensorboard
```

This step loads the TensorBoard extension into the notebook and we will be launching it later in the tutorial.

```
import tensorflow as tf
```

```
import datetime

!rm -rf ./logs/
```

Basic required imports and removing previous logs that might have used TensorBoard.

```
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

def create_model():
  return 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')
  ])
```

Similar to earlier, but the model is being put into a method that is going to be called later. In addition, the first dense layer has 512 outputs for higher accuracy. However, keep in mind that higher input does not always mean higher accuracy as it can cause overfitting.

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

log_dir = "logs/fit/" + datetime.datetime.now().strftime("%Y%m%d-%H%M%S")
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])
```

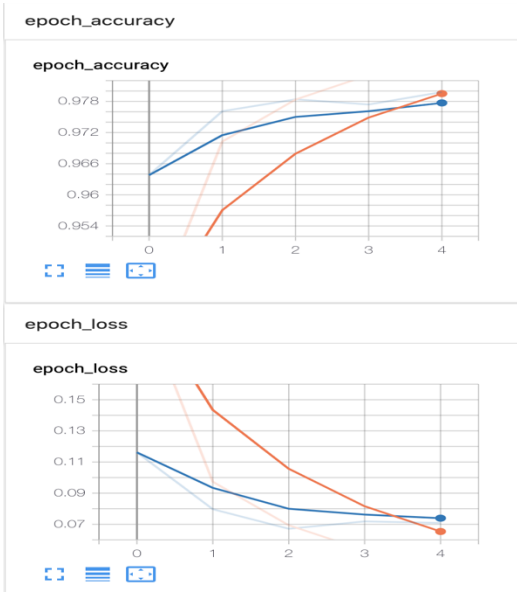
The `log_dir` variable is storing the directory of the log using the current date and time. `tf.keras.callbacks.TensorBoard()` allows for the log to be created and stored. The histogram parameter calculates the data for the histogram every *epoch.*

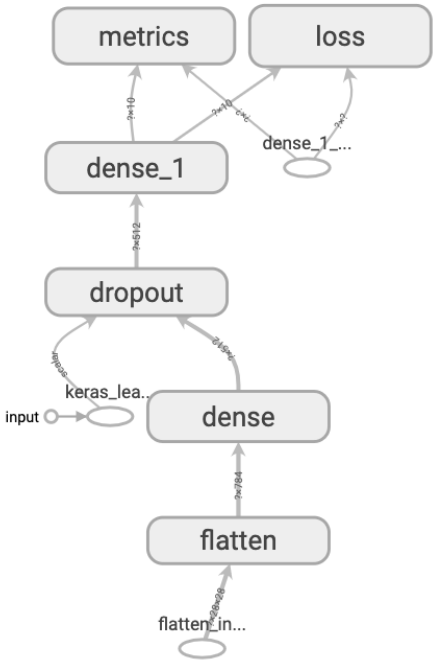
```
%tensorboard --logdir logs/fit
```

This launches TensorBoard and will show data about the neural net.

3. TensorBoard Walkthrough

Under the scalars tab the graphs shows how the loss and accuracy is changing with every epoch. It can also be used to track training speed and learning rate. Next, the graphs tab provides a great visual for the model. When `create_model()` was written a flatten layer was added, then a dense, then dropout, and finally another dense layer and this can be seen in the graph. This visual becomes more helpful when more complex neural nets are involved. The distributions and histogram tab show the distribution of a tensor over time. This includes changes regarding *weights* and *biases.*





4. Automatic Differentiation

A neural network's goal is to reduce its *loss* and it does that through a process known as *gradient descent*. This process involves tweaking the weights and biases. To figure out how much the network's weights and biases have to be tweaked, the *partial derivative* of the loss function must be calculated. Automatic differentiation is used by computers to calculate this. To sum it up, a neural network uses gradient descent to minimize its loss and automatic differentiation is used to calculate how that is supposed to happen. TensorFlow comes with automatic differentiation and calculates it using *chain rule*.

5. More Resources

https://deepnotes.io/tensorflow
https://towardsdatascience.com/automatic-differentiation-explained-b4ba8e60c2ad
https://www.tensorflow.org/tensorboard
https://jupyter.org/install
https://youtu.be/IHZwWFHWa-w

6. References

"Get started with TensorBoard." TensorFlow, 1 Apr. 2020, www.tensorflow.org/tensorboard/get_started.