An Introduction to Google's Machine Learning Tool TensorFlow - Part Two

Dr. Frazer K. Noble

MASSEY UNIVERSITY
TE KUNENGA KI PŪREHUROA
UNIVERSITY OF NEW ZEALAND

Outline

In today's lecture, we will:

- Explain how to visualise a graph.
- Explain how to save and restore a graph.
- Explain how to use a file as an input to a graph.
- Work through a number of examples.
- Write some programs in Python using TensorFlow.

Today, you will need the following tools[1]:

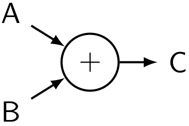
- Git [3].
- Anaconda [1].
- Visual Studio Code [9].

[1] Additional tools are required to run on a NVIDIA GPU.

Introduction I

So far, we have looked at defining simple graphs and feeding data into them directly, e.g.

```
A = tf.placeholder(tf.float32, None, ’A’)
B = tf.placeholder(tf.float32, None, ’B’)

C = A + B
```



Today, we’ll look at using TensorBoard to visualise our graphs and other sueful tools, which you’ll need.

For detailed descriptions of the topics discussed today, refer to TensorFlow’s documentation at: `https://www.tensorflow.org/api_docs/`.

Visualisation I

“TensorBoard operates by reading TensorFlow events files, which contain summary data that is generated when running TensorFlow [6].

Visualisation II

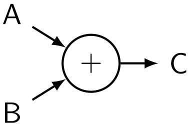
Consider the following program[2]:

```
import tensorflow as tf

A = tf.placeholder(tf.float32, None, 'A')
B = tf.placeholder(tf.float32, None, 'B')

C = A + B

tf.summary.scalar('A', A)
tf.summary.scalar('B', B)
tf.summary.scalar('C', C)

with tf.Session() as s:

summary_writer = tf.summary.FileWriter('./logs', s.graph)

merged = tf.summary.merge_all()

feed_dict = {A: 1.0, B: 2.0}
summary, ans = s.run([merged, C], feed_dict)

summary_writer.add_summary(summary, 0)
```

Listing 1: main.py

Visualisation III

Here, a simple graph is created:



and nodes A, B, and C's values are recorded.

The lines:

```
tf.summary.scalar('A', A)
tf.summary.scalar('B', B)
tf.summary.scalar('C', C)
```

"attach tf.summary.scalar ops to the nodes and give each scalar_summary a meaningful tag, e.g. 'A'" [6].

Visualisation IV

“Operations in TensorFlow don’t do anything until they are run, or an op that depends on their output is” [6]. Therefore, we need to run each summary, but this would be 'tedious’.

The line:

```
merged = tf.sumary.merge_all()
```

combines all the summary nodes into a single op, which generates all the summary data.

The line:

```
summary_writer = tf.summary.FileWriter('./logs', s.graph)
```

defines where the summary data is written to and whether to export session’s graph.

Visualisation V

The line:

```
summary, ans = s.run([merged, C], feed_dict)
```

runs the merged summary op, which generates a serialised Summary object with all the summary data at a given step.

The line:

```
summary_writer.add_summary(summary, 0)
```

adds the summary data to a FileWriter, which writes it to disk.

Visualisation VI

To run TensorBoard, use the following command:

tensorboard –logdir=path/to/log-directory

where logdir points to the directory where the FileWriter serialised its data.

Once TensorBoard is running, navigate to localhost:6006 to view the TensorBoard.

Visualisation VII

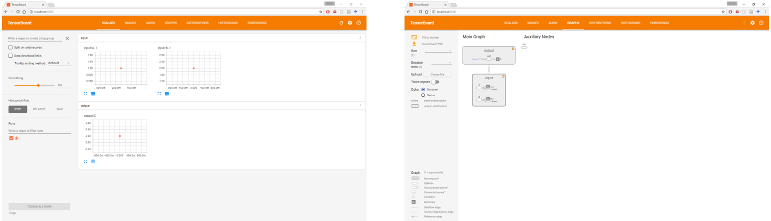


Figure: A TensorBoard example.

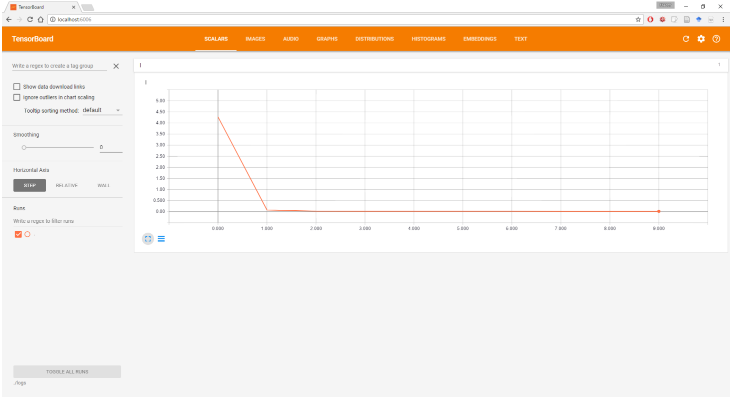
[2] A more complete example can be found at `https://github.com/FKNoble/tensorflow_projects/blob/master/visualising_a_graph/main.py`.

Activity I

Visualise the loss function for a simple program.

Write a program in Python, using TensorFlow, that visualises the loss function during training.

Activity II



□ Show data download links

□ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0

Figure: TensorBoard visualisation of resulting loss function during training.

An example is available at: `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/activity_3.py`.

Saving and Restoring a Model I

“The easiest way to save and restore a model is to use a tf.train.Saver object. The constructor adds save and restore ops to the graph for all, or some, variables in the graph.” [8].

Saving and Restoring a Model II

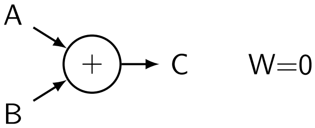
Consider the following program[3]

```
import tensorflow as tf

A = tf.placeholder(tf.float32, None, 'A')
B = tf.placeholder(tf.float32, None, 'B')

W = tf.Variable(0)

C = A + B

init = [tf.global_variables_initializer()]

with tf.Session() as s:

    s.run(init)

    saver = tf.train.Saver()

    feed_dict = {A: 2.0, B: 4.0}
    print(s.run(C, feed_dict=feed_dict))

    saver.save(s, './model/main.ckpt', 0)

with tf.Session() as l:

    loader = tf.train.Saver()

    ckpt = tf.train.latest_checkpoint('./model
           /')

    loader.restore(l, ckpt)

    feed_dict = {A: 2.0, B: 4.0}
    print(l.run(C, feed_dict=feed_dict))
```

Listing 2: main.py

Saving and Restoring a Model III

Here, a simple graph is created:



and is saved to a directory and then restored.

The line:

```
saver = tf.train.Saver()
```

adds ops to save and restore variables to and from checkpoints.

Saving and Restoring a Model IV

The line:

```
saver.save(s, ’./model/main.ckpt’, 0)
```

runs the op for saving variables.

The line:

```
loader = tf.train.Saver()
```

adds the ops to save and restore variables to and from checkpoints.

Saving and Restoring a Model V

The line:

```
ckpt = tf.train.latest_checkpoint('./model/')
```

finds the filename of the latest saved checkpoint file.

The line:

```
loader.restore(l, ckpt)
```

runs the op for restoring variables.

[3] A mode complete example can be found at: https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/tutorial_5.py.

Activity I

Save and restore a graph.

Write a program in Python, using TensorFlow, that exports a trained graph every 10 iterations.

Activity II

Homework.

Training a graph in one program, the inferring its output in another requires a MetaGraph file.

Search on-line for the term:

- "Exporting and Importing a MetaGraph".

Read about how to export and import a MetaGraph file.

Reading Data From a File I

A typical pipeline for reading data from files has the following stages:

1. Get a list of file names.
2. Create a file name queue.
3. Define the reader for the file format.
4. Decode the reader's record.
5. Create an output queue.

Reading Data From a File II

Consider the following program:

```
import tensorflow as tf
import utilities as utils

features = utils.read_CSV(['./data/data.csv'])

init = [tf.global_variables_initializer(), tf.
      local_variables_initializer()]

with tf.Session() as s:

    s.run(init)

    coord = tf.train.Coordinator()
    threads = tf.train.start_queue_runners(
          coord=coord)

try:
    while not coord.should_stop():

        f = s.run([features])

except tf.errors.OutOfRangeError:
    print('EoF')
finally:
    coord.request_stop()

coord.join(threads)
```

Listing 3: main.py

Reading Data From a File III

```
def read_CSV(filenames):
    '''read_CSV'''

    filename_queue = tf.train.
          string_input_producer(
        filenames, num_epochs=1, shuffle=
              True)

    reader = tf.TextLineReader()
    _, record_string = reader.read(
          filename_queue)

    record_defaults = [[0], [0]]
    col1, col2 = tf.decode_csv(record_string,
          record_defaults)

    example = tf.stack([col1, col2])
    example_batch = tf.train.batch(
        [example], batch_size=1)

    return example_batch
```

Listing 4: utilities.py

Reading Data From a File IV

The line:

```
filename_queue = tf.train.string_input_producer(filenames, num_epochs=1, shuffle=True)
```

"creates a FIFO queue for holding the file names until the reader needs them" [4], adds the file names to the queue num_epoch times, and shuffles the file names.

The lines:

```
reader = tf.TextLineReader()
  _, record_string = reader.read(filename_queue)
```

uses a tf.TextLineReader to read text files, and parses the filename_queue to get a single line from the file.

Reading Data From a File V

The lines:

```
    record_defaults = [[0], [0]]
    col1, col2 = tf.decode_csv(record_string, record_defaults)
```

decodes a line into a list of tensors; where, record_defaults determines the type of tensor and the default values to use.

The lines:

```
  example = tf.stack([col1, col2])
  example_batch = tf.train.batch([example], batch_size=1)
```

stack together the decoded tensors and creates a batch, which can be used for training, evaluation, or inference.

Activity I

Read a Comma Separated Value (CSV) file

Given the following "file", write a program in Python, using TensorFlow, that read its contents and prints each line to the console.

`data.csv`:

```
1.0, 2.0, 3.0
2.0, 4.0, 2.0
3.0, 6.0, 1.0
```

Activity II

An expected output is:

```
[array([[1, 2, 3]])]
[array([[2, 4, 2]])]
[array([[3, 6, 1]])]
EoF
```

An example is available at: `https://github.com/FKNoble/tensorflow_projects/blob/master/tutorials/tutorial_6.py`.

Homework I

Train, save, and infer the output of a neural network.

Write a program in Python, using TensorFlow, that trains a neural network, saves the model, restores the trained model, and infers an output.

Do the following:

1. Write the program.
2. Write comments for each line in the program.
3. Print the program's source code.

TensorFlow's documentation is available at: `http://www.tensorflow.org/api_docs/`.

Conclusion

In today's lecture, we have:

- Explained how to visualise a graph.
- Explained how to save and restore a graph.
- Explained how to use a file as an input to a graph.
- Worked through a number of examples.
- Written some programs in Python using TensorFlow.

Questions?

Bibliography I

Download anaconda now! — continuum.
https://www.continuum.io/downloads.
(Accessed on 06/01/2017).

Getting started with tensorflow — tensorflow.
https://www.tensorflow.org/get_started/get_started.
(Accessed on 06/02/2017).

Git.
https://git-scm.com/.
(Accessed on 06/01/2017).

Reading data — tensorflow.
https://www.tensorflow.org/programmers_guide/reading_data.
(Accessed on 06/13/2017).

Bibliography II

Tensorboard: Graph visualization — tensorflow.
https://www.tensorflow.org/get_started/graph_viz.
(Accessed on 06/12/2017).

Tensorboard: Visualizing learning — tensorflow.
https://www.tensorflow.org/get_started/summaries_and_tensorboard.
(Accessed on 06/12/2017).

Tensorflow.
https://www.tensorflow.org/.
(Accessed on 06/01/2017).

Variables: Creation, initialization, saving, and loading — tensorflow.
https://www.tensorflow.org/programmers_guide/variables.
(Accessed on 06/12/2017).

Bibliography III

Visual studio code - code editing. redefined. https://code.visualstudio.com/. (Accessed on 06/01/2017).