TensorFlow Input Pipeline

TensorFlow for Deep Learning Research
Lecture 8

[redacted]

Agenda

Data Readers Revisited

TFRecord

Variable Initializer

Graph Collection

Style Transfer

Queues

tf.Session objects are designed to multithreaded
→ can run ops in parallel

Queues

Important TensorFlow objects for computing tensors **asynchronously** in a graph.

- Multiple threads prepare training examples and push them in the queue
- A training thread executes a training op that dequeues mini-batches from the queue

Queues

Important TensorFlow objects for computing tensors **asynchronously** in a graph.

- All threads must be able to stop together
- Exceptions must be caught and reported
- Queues must be properly closed when stopping.

Queues

TensorFlow queues can’t run without proper threading, but threading isn’t exactly pleasant in Python

tf.Coordinator and tf.train.QueueRunner

- QueueRunner
  create a number of threads cooperating to enqueue tensors in the same queue

tf.Coordinator and tf.train.QueueRunner

- QueueRunner
  create a number of threads cooperating to enqueue tensors in the same queue
- Coordinator
  help multiple threads stop together and report exceptions to a program that waits for them to stop

Very similar to threadpool in CS110
Don’t worry if this sounds confusing.
Example in a bit

Queues

Queue	What's it?	Ops supported
tf.FIFOQueue	Dequeues elements in first in first out order	enqueue enqueue_many dequeue
tf.RandomShuffleQueue	Dequeues elements in a random order	enqueue enqueue_many dequeue
tf.PaddingFIFOQueue	FIFOQueue with padding to supports batching variable_size tensors	enqueue enqueue_many dequeue dequeue_many
tf.PriorityQueue	FIFOQueue whose enqueue and queue have another argument: priority	enqueue enqueue_many dequeue

Queues


```
q = tf.FIFOQueue(3, "float")
init = q.enqueue_many(([0.,0.,0.],))

x = q.dequeue()
y = x+1
q_inc = q.enqueue([y])

init.run()
q_inc.run()
q_inc.run()
q_inc.run()
q_inc.run()
```

Create a queue

tf.FIFOQueue(capacity, min_after_dequeue, dtypes, shapes=None, names=None ...)

Same for other queues

Queue example

09_queue_example.py

```
all_data = 10 * np.random.randn(N_SAMPLES, 4) + 1
all_target = np.random.randint(0, 2, size=N_SAMPLES)

queue = tf.FIFOQueue(capacity=50, dtypes=[tf.float32, tf.int32], shapes=[[4], []])

enqueue_op = queue.enqueue_many([all_data, all_target])
data_sample, label_sample = queue.dequeue()

qr = tf.train.QueueRunner(queue, [enqueue_op] * NUM_THREADS)
with tf.Session() as sess:
     # create a coordinator, launch the queue runner threads.
     coord = tf.train.Coordinator()
     enqueue_threads = qr.create_threads(sess, coord=coord, start=True)
     for step in xrange(100): # do to 100 iterations
          if coord.should_stop():
               break
          one_data, one_label = sess.run([data_sample, label_sample])
     coord.request_stop()
     coord.join(enqueue_threads)
```

Queue example

```
# dummy data
all_data = 10 * np.random.randn(N_SAMPLES, 4) + 1
all_target = np.random.randint(0, 2, size=N_SAMPLES)
```

In practice, you can use any op to read in your data,
even placeholder!

Queue example

```
...

queue = tf.FIFOQueue(capacity=50, dtypes=[tf.float32, tf.int32], shapes=[[4], []])
# create queue.
# dtypes specifies types of data and label
# shapes specifies shape of data and label

...
```

Queue example

```
...

enqueue_op = queue.enqueue_many([all_data, all_target])
data_sample, label_sample = queue.dequeue()

# a common practice is to enqueue all data at once, but dequeue one by one
...
```

Queue example

```
...

qr = tf.train.QueueRunner(queue, [enqueue_op] * NUM_THREADS)
with tf.Session() as sess:
    # create a coordinator, launch the queue runner threads.
    coord = tf.train.Coordinator()
    enqueue_threads = qr.create_threads(sess, coord=coord, start=True)
    for step in xrange(100): # do to 100 iterations
        if coord.should_stop():
            break
        one_data, one_label = sess.run([data_sample, label_sample])
    coord.request_stop()
    coord.join(enqueue_threads)
....
```

You can use data_sample and label_sample to do all the training ops as if with placeholders

Dequeue multiple elements?

tf.train.batch or tf.train.shuffle_batch if you want to your batch to be shuffled

I have never been able to get these to work with independent queues

Re: dequeue_many is tricky with queues

tf.Coordinator

Can be used to manage the threads you created without queues

tf.Coordinator

```
import threading

# thread body: loop until the coordinator indicates a stop was requested.
# if some condition becomes true, ask the coordinator to stop.

def my_loop(coord):
      while not coord.should_stop():
            ...do something...
      if ...some condition...:
            coord.request_stop()

# main code: create a coordinator.
coord = tf.Coordinator()

# create 10 threads that run 'my_loop()'
# you can also create threads using QueueRunner as the example above
threads = [threading.Thread(target=my_loop, args=(coord,)) for _ in xrange(10)]

# start the threads and wait for all of them to stop.
for t in threads: t.start()
coord.join(threads)
```

Just like threadpool

Take CS110 for more threading fun!

Data Readers

Three ways to read in data

1. Through tf.constant (make everything a constant)

It'll seriously bloat your graph
(you'll see in assignment 2)

Three ways to read in data

1. Through tf.constant (make everything a constant) **NO**

2. Feed dict

Slow when client and workers are on different machines

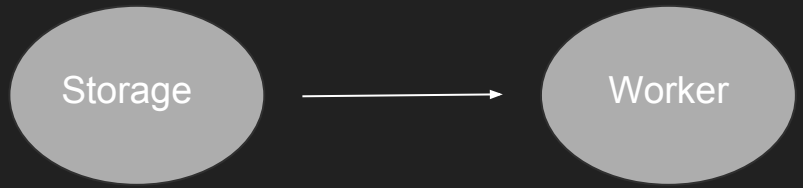
Three ways to read in data

1. Through tf.constant (make everything a constant)
   NO

2. Feed dict
   MAYBE ...

3. Data readers

Data Readers



Readers allow us to load data directly into the worker process.

Different Readers for different file types

tf.TextLineReader
Outputs the lines of a file delimited by newlines
E.g. text files, CSV files

tf.FixedLengthRecordReader
Outputs the entire file when all files have same fixed lengths
E.g. each MNIST file has 28 x 28 pixels, CIFAR-10 32 x 32 x 3

tf.WholeFileReader
Outputs the entire file content

tf.TFRecordReader
Reads samples from TensorFlow’s own binary format (TFRecord)

tf.ReaderBase
To allow you to create your own readers

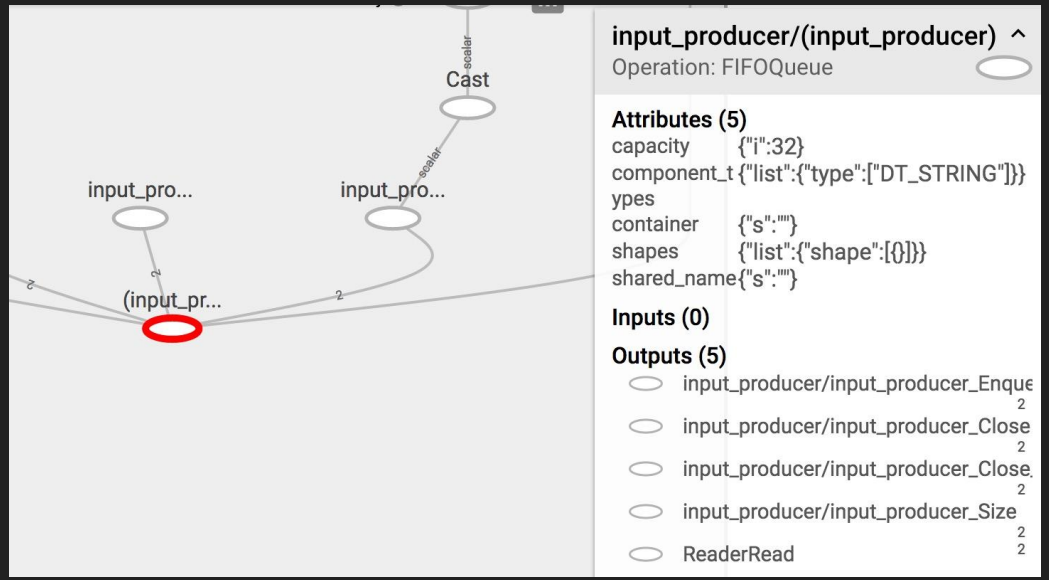
Read in files from queues

```
filename_queue = tf.train.string_input_producer(["file0.csv", "file1.csv"])

reader = tf.TextLineReader()
key, value = reader.read(filename_queue)
```

Read in files from queues

```
filename_queue = tf.train.string_input_producer(["heart.csv"])

reader = tf.TextLineReader(skip_header_lines=1)
key, value = reader.read(filename_queue)
```

string_input_producer is really a queue



Read in files from queues

```
filename_queue = tf.train.string_input_producer(["heart.csv"])

reader = tf.TextLineReader(skip_header_lines=1)
key, value = reader.read(filename_queue)

with tf.Session() as sess:
        coord = tf.train.Coordinator()
        threads = tf.train.start_queue_runners(coord=coord)
        for _ in range(1): # generate 1 example
            features, labels = sess.run([data_batch, label_batch])
        coord.request_stop()
        coord.join(threads)
```

Need Coordinator and QueueRunner

Read in files from queues

```
filename_queue = tf.train.string_input_producer(["heart.csv"])

reader = tf.TextLineReader(skip_header_lines=1)
key, value = reader.read(filename_queue)

with tf.Session() as sess:
        coord = tf.train.Coordinator()
        threads = tf.train.start_queue_runners(coord=coord)
        for _ in range(1): # generate 1 example
            key, value = sess.run([key, value])
            print valuee # 144,0.01,4.41,28.61,Absent,55,28.87,2.06,63,1
            print key # data/heart.csv:2
        coord.request_stop()
        coord.join(threads)
```

Value is just text. Need to convert to 2 tensors:

+ Features tensor
+ Label tensor

Live example (05_csv_reader.py)

TFRecord

TensorFlow's binary file format

a serialized tf.train.Example protobuf object

Why binary?

- make better use of disk cache
- faster to move around
- can store data of different types (so you can put both images and labels in one place)

Convert normal files to TFRecord

- Super easy
- Live example

Read in TFRecord

- Using TFRecordReader, duh
- Live example

Style Transfer

Not too much math,
but implementation is tricky

Mathy stuff

Find a new image:

- whose content is closest to the content image and
- whose style is closest to the style image

It’s all about the loss functions

- Content loss
  To measure the content loss between the content of the generated image and the content of the content image
- Style loss
  To measure the style loss between the style of the generated image and the style of the style image

What is the content/style of an image?

Content/style of an image

Feature visualization have shown that:

- lower layers extract features related to content
- higher layers extract features related to style

Loss functions revisited

- Content loss
  To measure the content loss between **the feature map in the content layer** of the generated image and the content image
- Style loss
  To measure the style loss between **the feature maps in the style layers** of the generated image and the style image

Loss functions revisited

- Content loss
  To measure the content loss between **the feature map in the content layer** of the generated image and the content image

  Paper: ‘conv4_4’

- Style loss
  To measure the style loss between **the gram matrices of feature maps in the style layers** of the generated image and the style image

  Paper: [‘conv1_1’, ‘conv2_1’, ‘conv3_1’, ‘conv4_1’ and ‘conv5_1’]

Loss functions revisited

- Content loss

  To measure the content loss between **the feature map in the content layer** of the generated image and the content image

  Paper: ‘conv4_4’

- Style loss

  To measure the style loss between **the gram matrices of feature maps in the style layers** of the generated image and the style image

  Paper: [‘conv1_1’, ‘conv2_1’, ‘conv3_1’, ‘conv4_1’ and ‘conv5_1’]

  Give more weight to deeper layers
  E.g. 1.0 for ‘conv1_1’, 2.0 for ‘conv2_1’, ...

Loss functions revisited

- Content loss

$$\mathcal{L}_{content}(\vec{p}, \vec{x}, l) = \frac{1}{2} \sum_{i,j} \left(F_{ij}^{l} - P_{ij}^{l} \right)^2$$

- Style loss

$$E_l = \frac{1}{4N_l^2 M_l^2} \sum_{i,j} \left(G_{ij}^{l} - A_{ij}^{l} \right)^2$$

$$\mathcal{L}_{style}(\vec{a}, \vec{x}) = \sum_{l=0}^{L} w_l E_l$$

Optimizer

Optimizes the initial image to minimize the combination of the two losses

$$\mathcal{L}_{total}(\vec{p}, \vec{a}, \vec{x}) = \alpha \mathcal{L}_{content}(\vec{p}, \vec{x}) + \beta \mathcal{L}_{style}(\vec{a}, \vec{x})$$

Do not optimize the weights!

Tricky implementation details

1. Train input instead of weights

Tricky implementation details

1. Train input instead of weights
2. Multiple tensors share the same variable to avoid assembling identical subgraphs
    a. Content image
    b. Style image
    c. Initial image

Tricky implementation details

1. Train input instead of weights
2. Multiple tensors share the same variable to avoid assembling identical subgraphs
3. Use pre-trained weights (from VGG-19)
    a. Weights and biases already loaded for you
    b. They are numpy, so need to be converted to tensors
    c. Must not be trainable!!

Next class

RNNs!

Example: translate

Feedback:

Thanks!