Tensorflow tutorial

CSML reading group practicals, UCL

Yuttapong Thawornwattana

September 2016

Basics

- Constant, variable, session

```
a = tf.constant(np.array([(0, 1), (2, 3)]))
b = tf.Variable(tf.ones((1, 2)))
c = tf.Variable(tf.random_uniform([2, 3], -1.0, 1.0)) # iid U(-1,1) random variates
d = tf.matmul(b, c)

with tf.Session() as sess:
    print(a.eval())  # or sess.run(a)

    sess.run(tf.initialize_all_variables())  # initialize variables
    print(b.eval())
    print(c.eval())
    print(d.eval())
```

- Fetching variable state

`run()` takes a list of output names needed to be computed (and optional tensors to be fed into the graph in place of certain outputs of nodes)

```
with tf.Session() as sess:
  x, y, z = sess.run([b, c, d])

print(x)
print(y)
print(z)
```

Basics

- Placeholder

```
a = tf.placeholder(tf.float32)
b = tf.placeholder(tf.float32)
c = tf.mul(a, b)
with tf.Session() as sess:
  print(sess.run([c], feed_dict={a: [7], b: [2]}))
```

- Create variables using `tf.get_variable()`
    - name_scope

    ```
    with tf.name_scope('scope1'):
        a = tf.Variable(tf.constant(0.1))
        b = tf.get_variable('b', [1])
        print(a.name)  # scope1/Variable:0
        print(b.name)  # b:0
    ```

    - variable_scope

    ```
    with tf.variable_scope('scope1'):
        a = tf.Variable(tf.constant(0.1))
        b = tf.get_variable('b', [1])
        print(a.name)  # scope1_1/Variable:0
        print(b.name)  # scope1/b:0
    ```

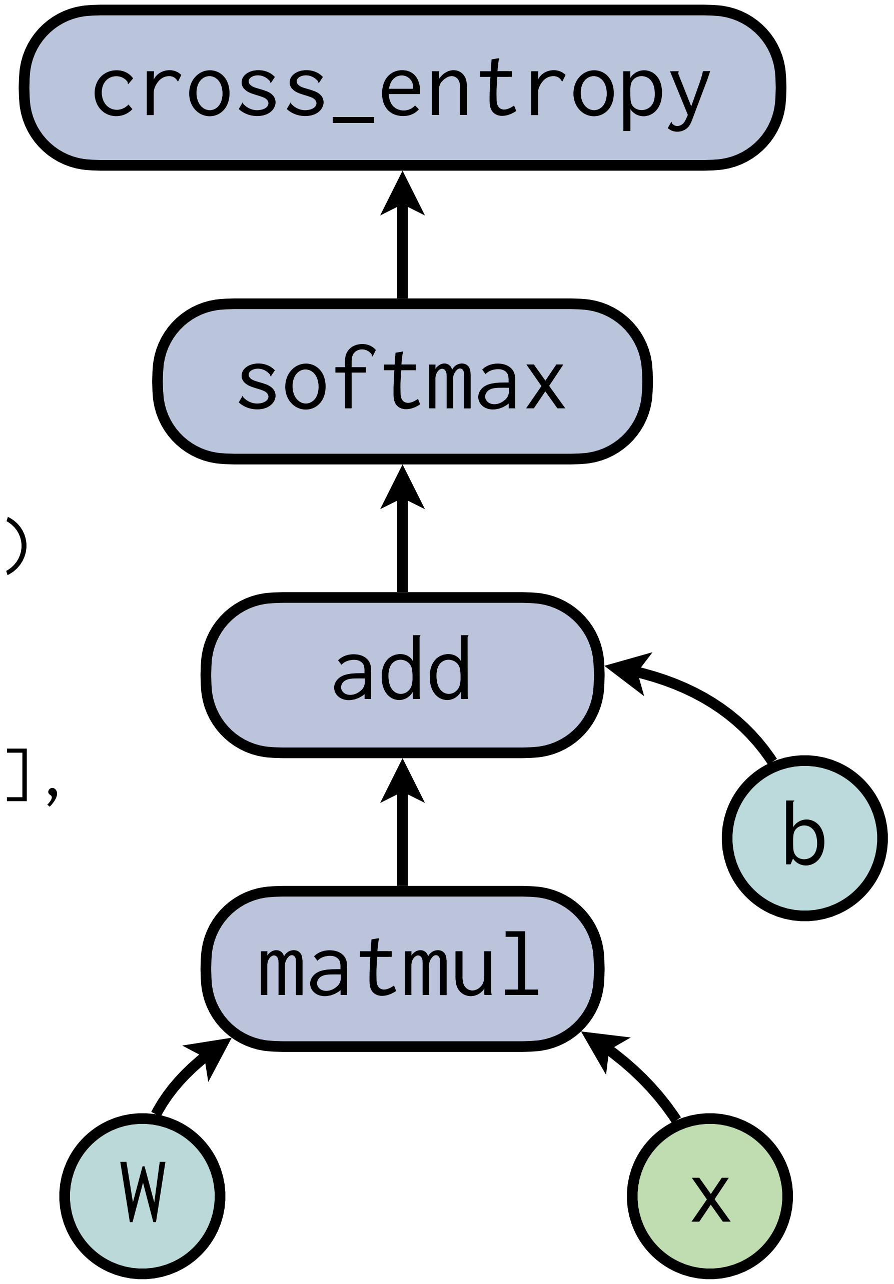
Models

1. Logistic regression
2. Neural network with one hidden layer

Logistic regression

$$y = \frac{1}{1 + e^{-(Wx+b)}}$$

```
x = tf.placeholder(tf.float32,
                   shape=[None, input_dim])
y_true = tf.placeholder(tf.float32,
                        shape=[None, output_dim])

w = tf.Variable(
      tf.truncated_normal([input_dim, output_dim],
                          stddev=0.1))
b = tf.Variable(
      tf.constant(0.1, shape=[output_dim]))
y = tf.nn.softmax(tf.matmul(x, W) + b)

cross_entropy = tf.reduce_mean(
  -tf.reduce_sum(y_true * tf.log(y),
                 reduction_indices=[1]))
```



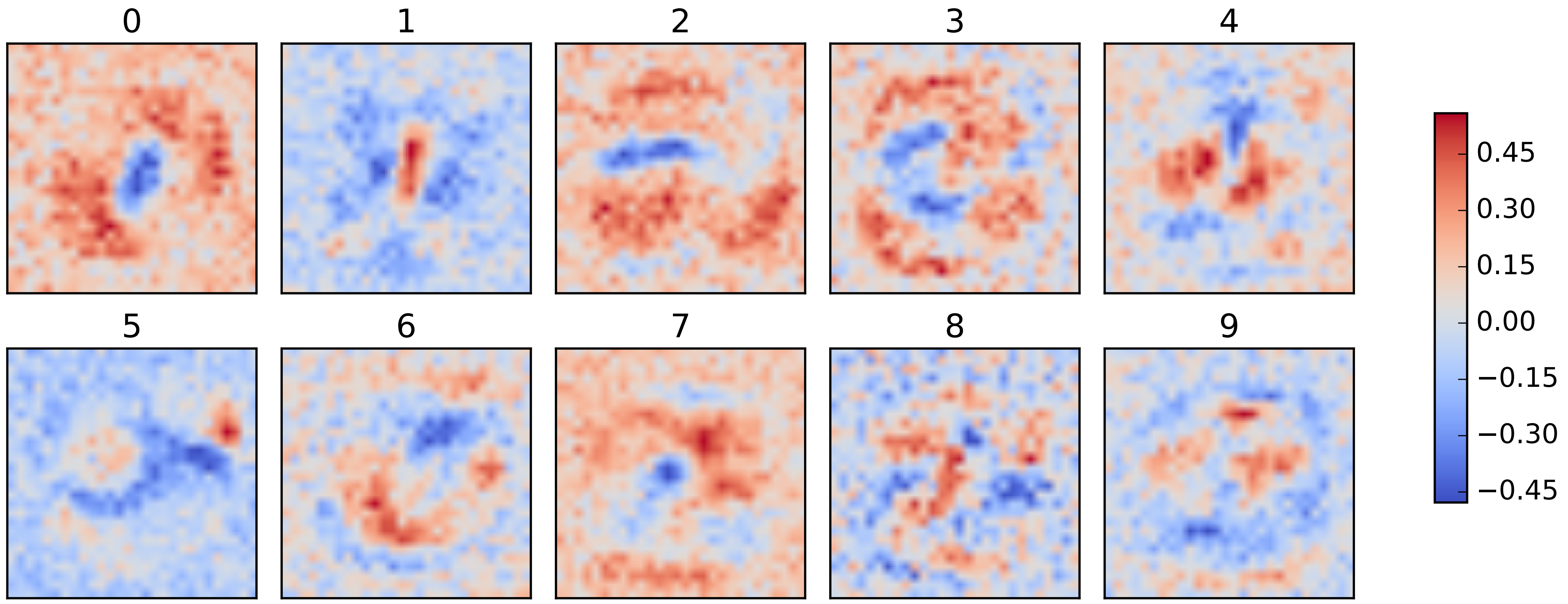
mnist_1.py

Training

```
optimizer = tf.train.GradientDescentOptimizer(0.5).minimize(cross_entropy)

correct_prediction = tf.equal(tf.argmax(y, 1), tf.argmax(y_true, 1))
accuracy = tf.mul(tf.reduce_mean(tf.cast(correct_prediction, tf.float32)),
100.0)

num_steps = 1000
batch_size = 100

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

    for i in range(num_steps):
        batch_x, batch_y = mnist.train.next_batch(batch_size)
        sess.run(optimizer, feed_dict={x: batch_x, y_true: batch_y})

    # prediction accuracy on test data
    print(accuracy.eval(
      feed_dict={x: mnist.test.images, y_true: mnist.test.labels}))
```

Logistic regression

Visualize learned weights `W`



`mnist_1.py`

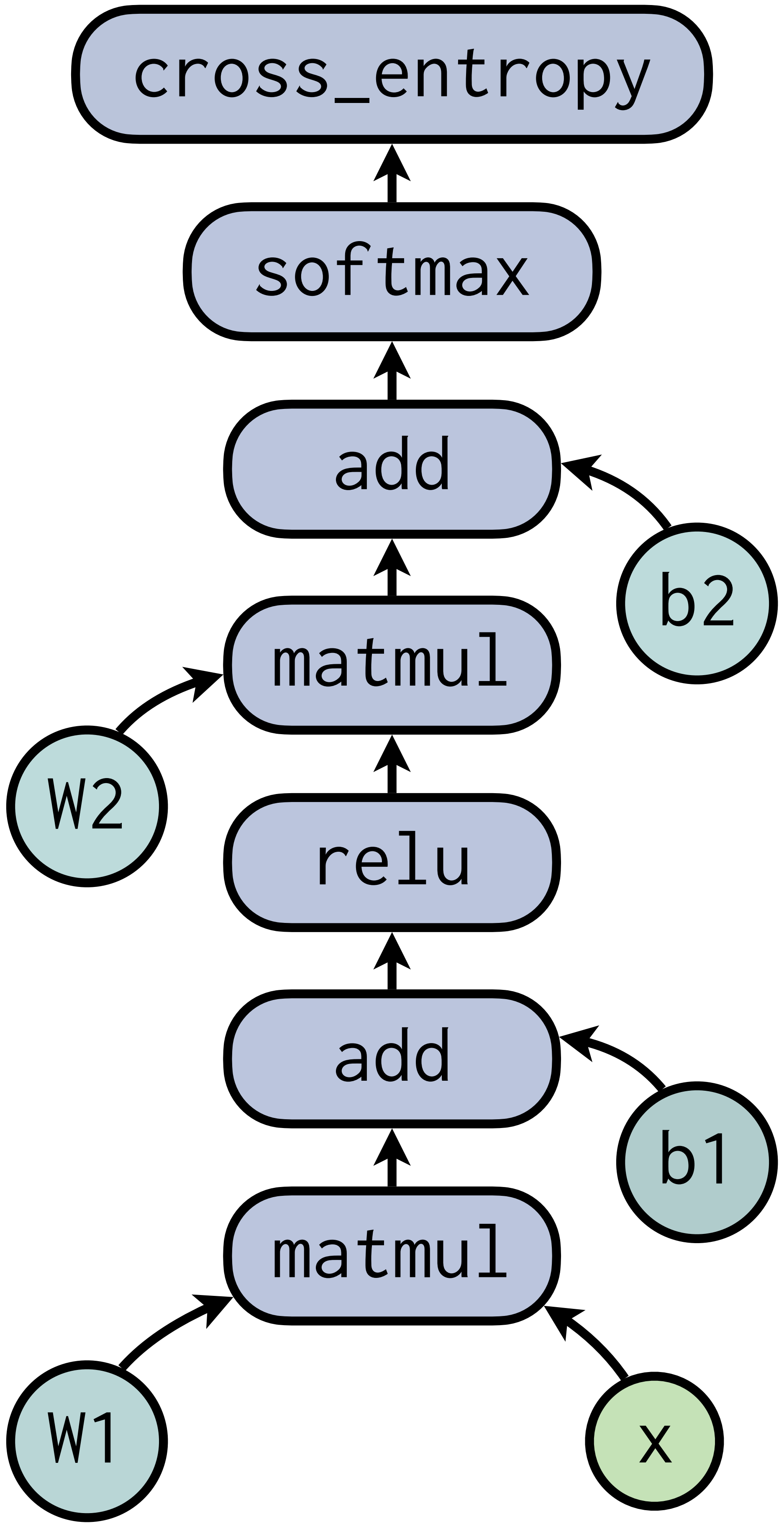
Neural network model

```
w2 = weight_variable([num_hidden, output_dim])
b2 = bias_variable([output_dim])
y = tf.nn.softmax(tf.matmul(h, w2) + b2)
```

$$y = \frac{1}{1 + e^{-(W_2 h + b_2)}}$$

```
w1 = weight_variable([input_dim, num_hidden])
b1 = bias_variable([num_hidden])
h = tf.nn.relu(tf.matmul(x, w1) + b1)
```

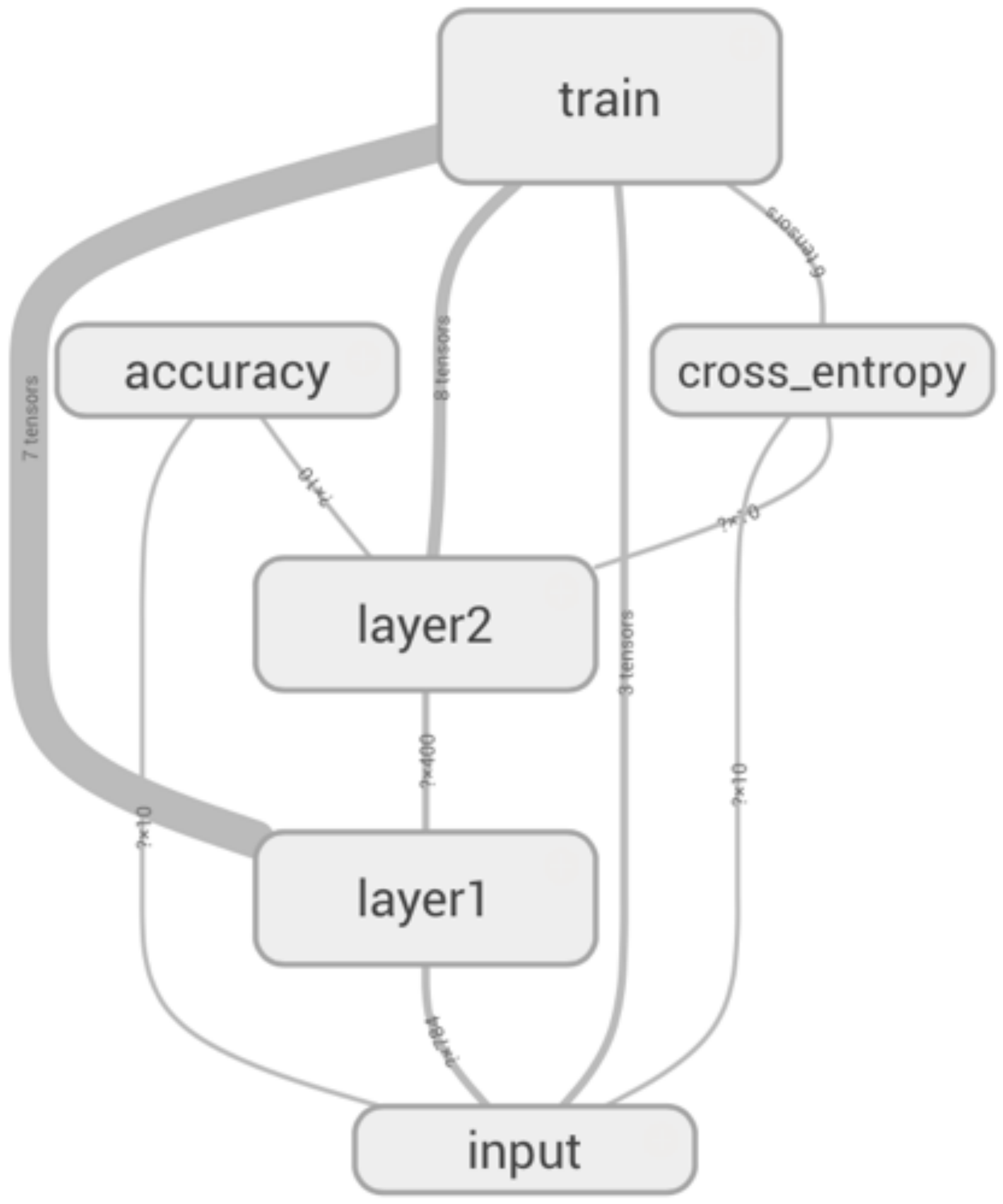
$$h = \mathrm{ReLU}(W_1 x + b_1)$$



mnist_2.py

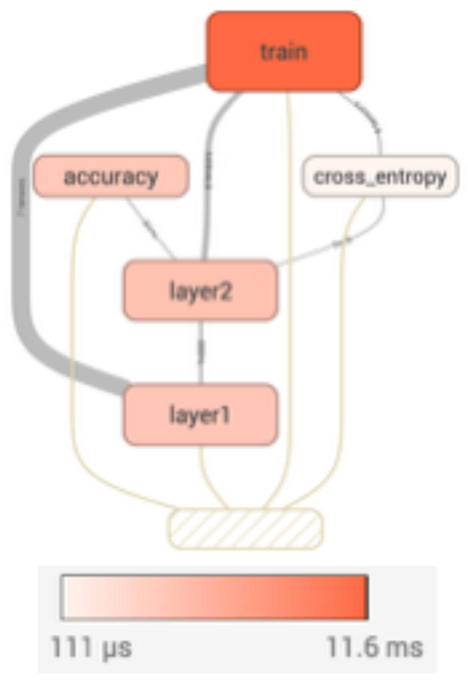
Visualization with **TensorBoard**

Computation graph

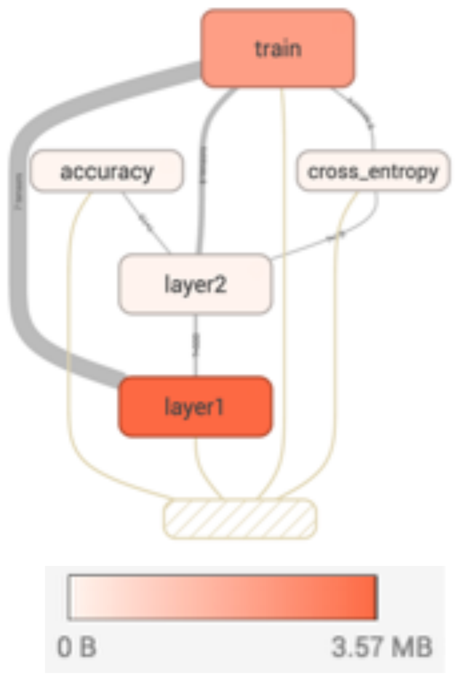


Runtime statistics

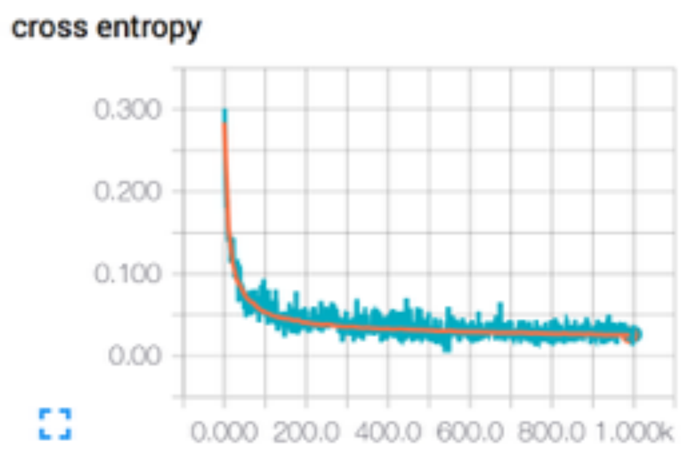
Compute time



Memory



Training



`mnist_2_summary.py`