Tensorflow Implementation

```
def tf_NN_model(config):

    # Placeholders
    X = tf.placeholder("float", [None, config.DIMENSIONS])
    y = tf.placeholder("float", [None, config.NUM_CLASSES])

    # Weights
    W1 = tf.Variable(tf.random_normal([config.DIMENSIONS, config.NUM_HIDDEN_UNITS], stddev=0.01), "W1")
    W2 = tf.Variable(tf.random_normal([config.NUM_HIDDEN_UNITS, config.NUM_CLASSES], stddev=0.01), "W2")

    with tf.name_scope('forward_pass') as scope:
        z_2 = tf.matmul(X, W1)
        a_2 = tf.nn.relu(z_2)
        logits = tf.matmul(a_2, W2)
```

$\longrightarrow$ forward pass

```
    # Add summary ops to collect data
    W_1 = tf.histogram_summary("W1", W1)
    W_2 = tf.histogram_summary("W2", W2)

    with tf.name_scope('cost') as scope:
        cost = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits, y)) + 0.5 * config.REG * tf.reduce_sum(W1*W1)
+ 0.5 * config.REG * tf.reduce_sum(W2*W2)
        tf.scalar_summary("cost", cost)
```

$\longrightarrow$ write to summary

```
    with tf.name_scope('train') as scope:
        optimizer = tf.train.AdamOptimizer(learning_rate=config.LEARNING_RATE).minimize(cost)
```

$\longrightarrow$ Adam optimizer

```
    return dict(X=X, y=y, W1=W1, W2=W2, logits=logits, cost=cost, optimizer=optimizer)
```

Tensorflow Implementation

```
def train(config, g, X, y):

    # Merge all summaries into a single operator
    merged_summary_op = tf.merge_all_summaries()

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

        summary_writer = tf.train.SummaryWriter('logs', graph=sess.graph)

        # y to categorical
        Y = tf.one_hot(y, 3).eval()

        for epoch_num in range(config.NUM_EPOCHS):
            logits, training_loss, _ = sess.run([g['logits'],
                                            g['cost'],
                                            g['optimizer']],
                                            feed_dict={g['X']:X, g['y']:Y})
            # Display
            if epoch_num%config.DISPLAY_STEP == 0:
                print "EPOCH %i: \n Training loss: %.3f, Accuracy: %.3f" % (
                    epoch_num, training_loss, np.mean(np.argmax(logits, 1) == y))

                # Write logs for each epoch_num
                summary_str = sess.run(merged_summary_op, feed_dict={g['X']:X, g['y']:Y})
                summary_writer.add_summary(summary_str, epoch_num)

if __name__ == '__main__':
    config = parameters()
    config, X, y = load_data(config)
    g = tf_NN_model(config)
    train(config, g, X, y)
```

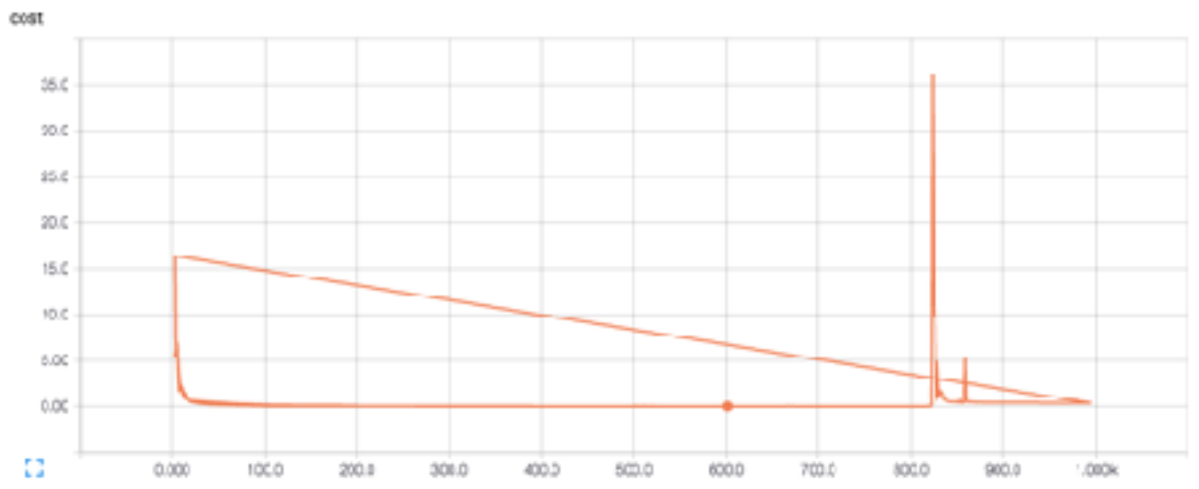
$\rightarrow$ merge all summaries

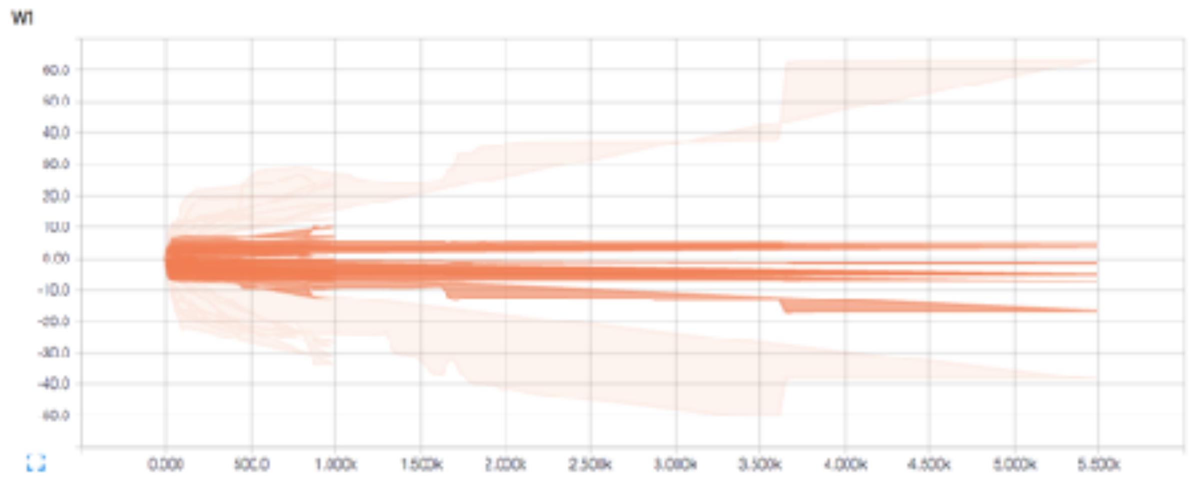
$\rightarrow$ write to logs directory

$\rightarrow$ add summary for each epoch

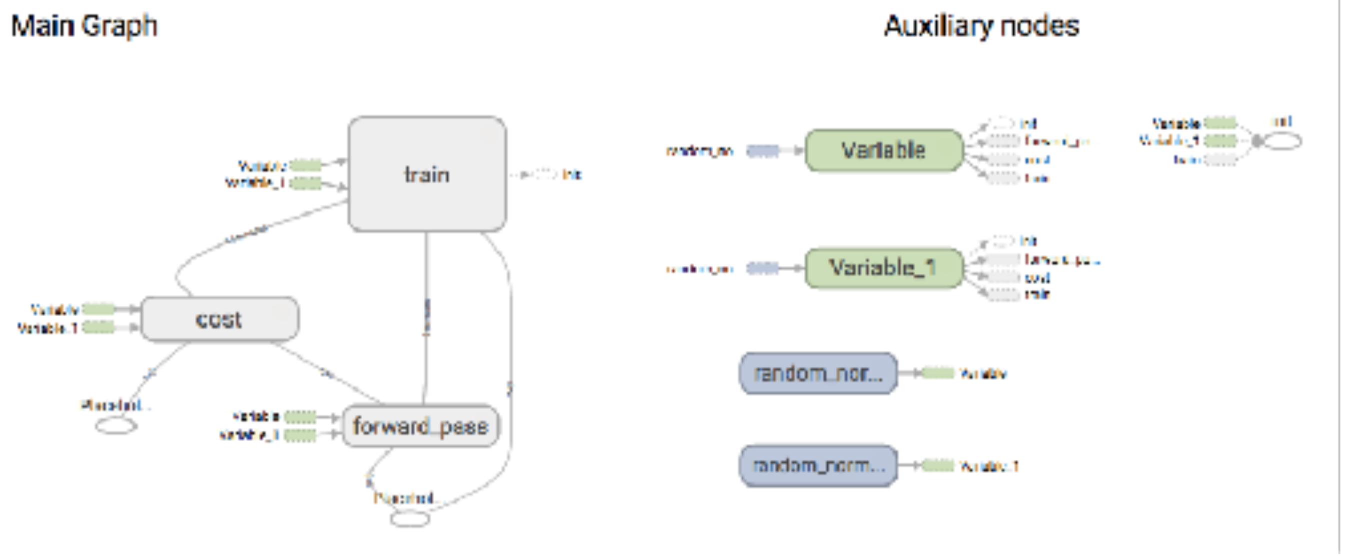
Tensorboard

```
tensorboard --logdir=logs
```





```
EPOCH 0:
 Training loss: 1.099, Accuracy: 0.333
EPOCH 10:
 Training loss: 3.569, Accuracy: 0.600
EPOCH 20:
 Training loss: 2.830, Accuracy: 0.840
EPOCH 30:
 Training loss: 2.815, Accuracy: 0.929
EPOCH 40:
 Training loss: 2.427, Accuracy: 0.949
EPOCH 50:
 Training loss: 1.888, Accuracy: 0.980
EPOCH 60:
 Training loss: 1.448, Accuracy: 0.988
EPOCH 70:
 Training loss: 1.137, Accuracy: 0.985
EPOCH 80:
 Training loss: 0.923, Accuracy: 0.979
EPOCH 90:
 Training loss: 0.770, Accuracy: 0.985
```


Main Graph
Auxiliary nodes
train
cost
forward_pass
Variable
Variable_1
random_nor...
random_norm...