CS498-AML Homework 9

Part 1: Convolutional Neural Network on MNIST Dataset

Eloise Rosen & Mark Berman

Part 1 A:

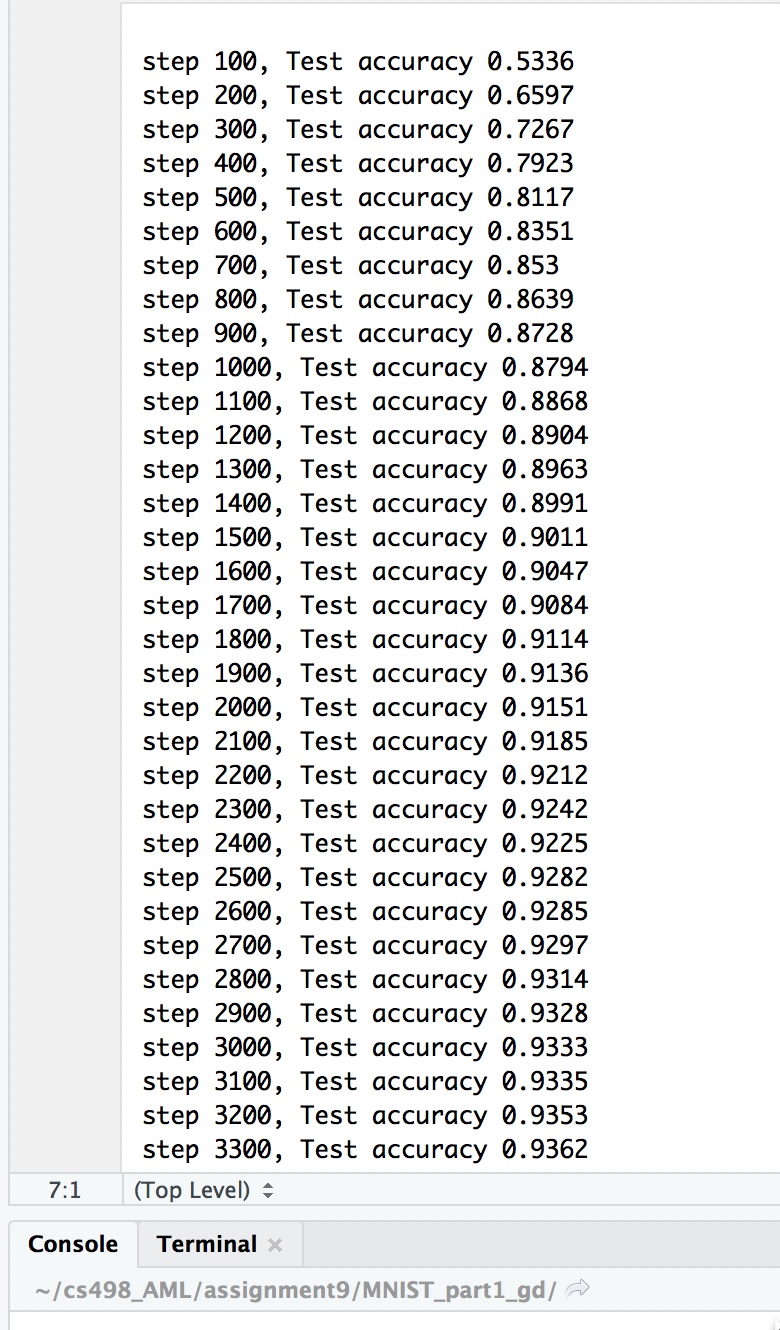
The Python based tutorial that was cited in the homework instructions was used as the baseline for our submission for Part 1. The Python tutorial was manually converted to the R based Tensorflow API and then integrated with TensorBoard “summary” operations to log histograms of output tensors and scalar metrics to capture “loss” (e.g., “cross entropy”) and test set accuracy. Loss and test set accuracy were logged to TensorBoard every 100 iterations of the training loop. The training loop was set for a run of 10,000 iterations. The “R” code file for Part 1 is available for inspection and is labeled “hw9\_mnist\_part1\_gd.Rmd”.

The Convolutional Neural Network architecture for P 1 A Is as follows.

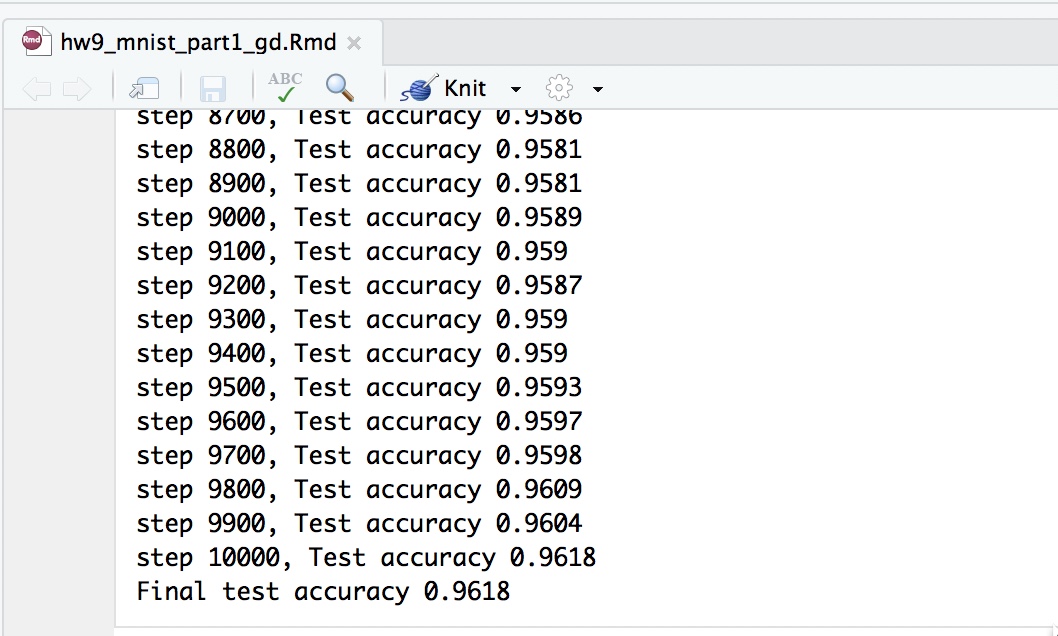
* Loss Minimization function: cross\_entropy <- tf$reduce\_mean(-tf$reduce\_sum(y\_ \* tf$log(y\_conv), reduction\_indices=1L))
* Optimization function: GradientDescentOptimizer
* Learning Rate: 0.001
* Mini-batch size: 100
* Keep probability: 0.4
* Max. # of steps: 10,000
* # Training Elements: 50,000
* # Test Elements: 10,000
* Two Convolutional layers; with each of these layers followed by a pooling layer. Followed by two fully connected layers and final” Softmax” function call that produces a matrix of 50,000 predicted image labels.
  + Convolutional Layer 1:
    - Dim of each input training element (a[0]): h =28, w=28, channels =1
    - kernel size: h=5, w=5, channels=1
    - # kernels: 32
    - stride: 1
    - padding: 2 (e.g., “SAME”)
    - bias: 32
    - activation function: RELU
    - Dim of each output training element (a[1] ): h=28, w=28, channels=32
  + Pooling Layer 1:
    - Dim of each training input element (a[1] ): h=28, w=28, channels=32
    - Kernel size: h=2, w=2, channels=1
    - # kernels: 32
    - stride: 2
    - Dim of each training output element (a[2] ): h=14, w=14, channels=32
  + Convolutional Layer 2:
    - Dim of each training input element (a[2]): h =14, w=14, channels =32
    - kernel size: h=5, w=5, channels=32
    - # kernels: 64
    - stride: 1
    - padding: 2 (e.g., “SAME”)
    - bias: 64
    - activation function: RELU
    - Dim of each training output element (a[3] ): h=14, w=14, channels=64
  + Pooling Layer 2:
    - Dim of each training input element (a[3]): h =14, w=14, channels =64
    - Kernel size: h=2, w=2, channels=1
    - # kernels: 64
    - stride: 2
    - Dim of each training output element (a[4] ): h=7, w=7, channels=64
  + Fully Connected Layer 1:
    - Dim of each input training element (after flattening a[4]): h=1, w=3136
    - Dim of each output training element (a[5]): h=1, w=1024
  + Fully Connected Layer 2:
    - Dim of each input training element (a[5]): h=1, w=1024
    - Dim of each output training element (a[6]): h=1, w=10
  + Softmax Function
    - Dim of each predicted element (a[7] ): h=1, w=10
    - Dim of prediction matrix: h=50,000, w=10

Findings:

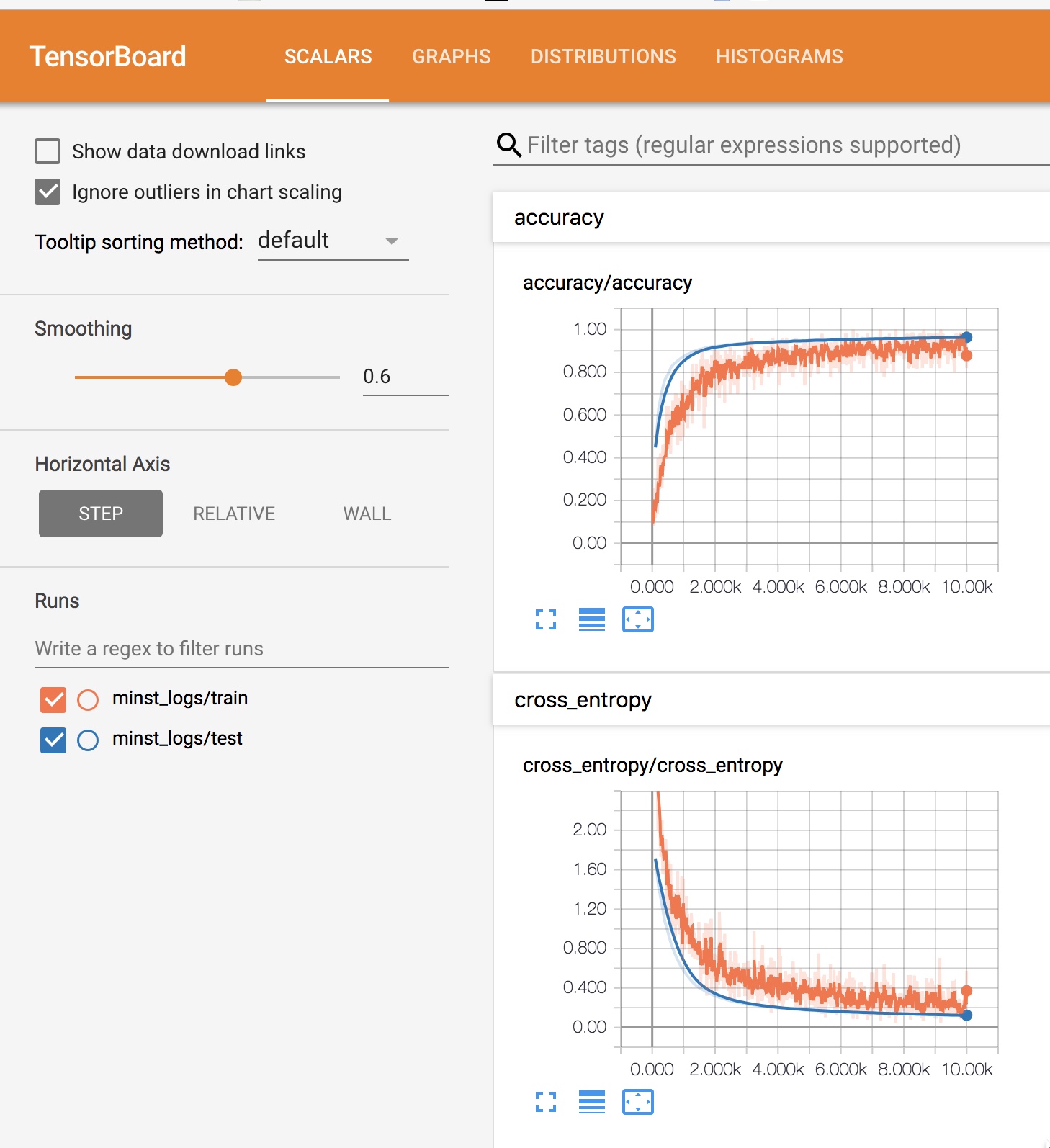
Test set accuracy is measured every 100 iterations the Convolutional Neural Network (CNN) performs training over the training data set. Test set accuracy is 91.51 percent at 2,000th iteration of the CNN over the training data set. This is shown in the following screen capture from R-studio.



Test set accuracy climbs to 96.18 percent accuracy after the 10,000th iteration over the training set by the CNN. This is shown below in the following screen capture from R-Studio.



The trajectory of the increase in test set accuracy is shown below in the following screen capture from TensorBoard. The slope of the test set accuracy curve (blue line) begins to flatten out at 3,000 iterations over the training set by the CNN. A larger version of this TensorBoard plot is available for inspection and is labeled “tensor\_board\_part1.jpeg”.



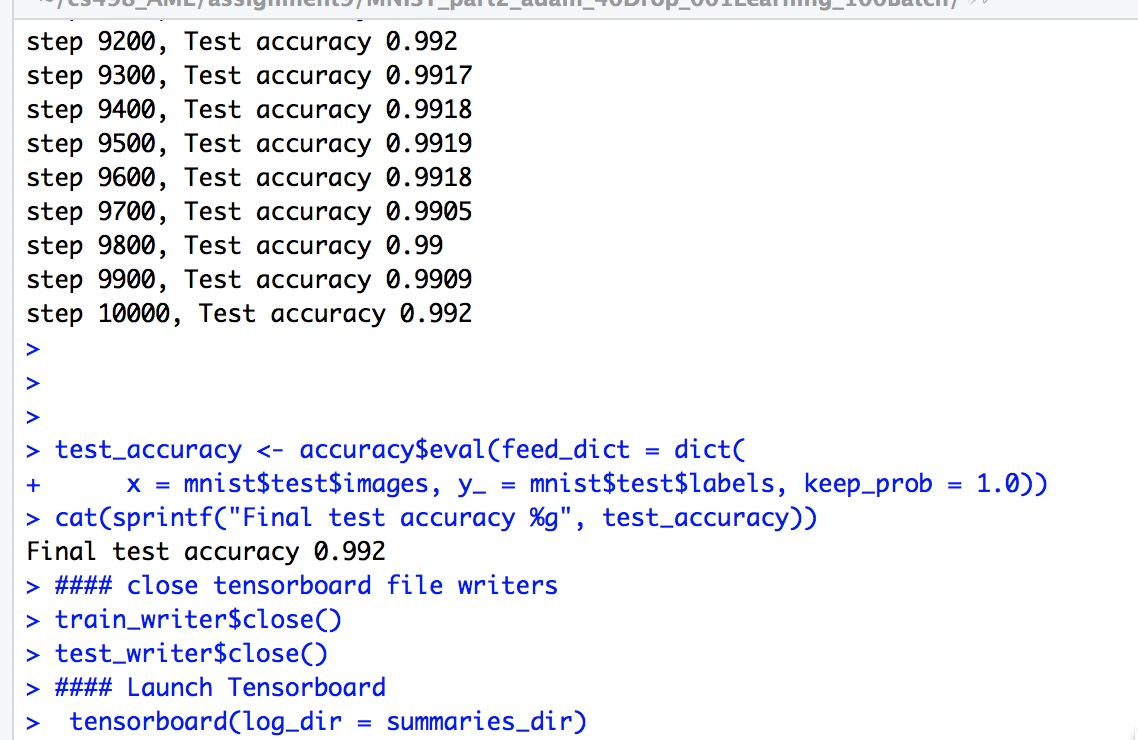
Part 1-B

We identified four different alternatives to the CNN architecture that out-performed test set accuracy at both 2,000 iterations and 10,000 iterations of the CNN over the training set.

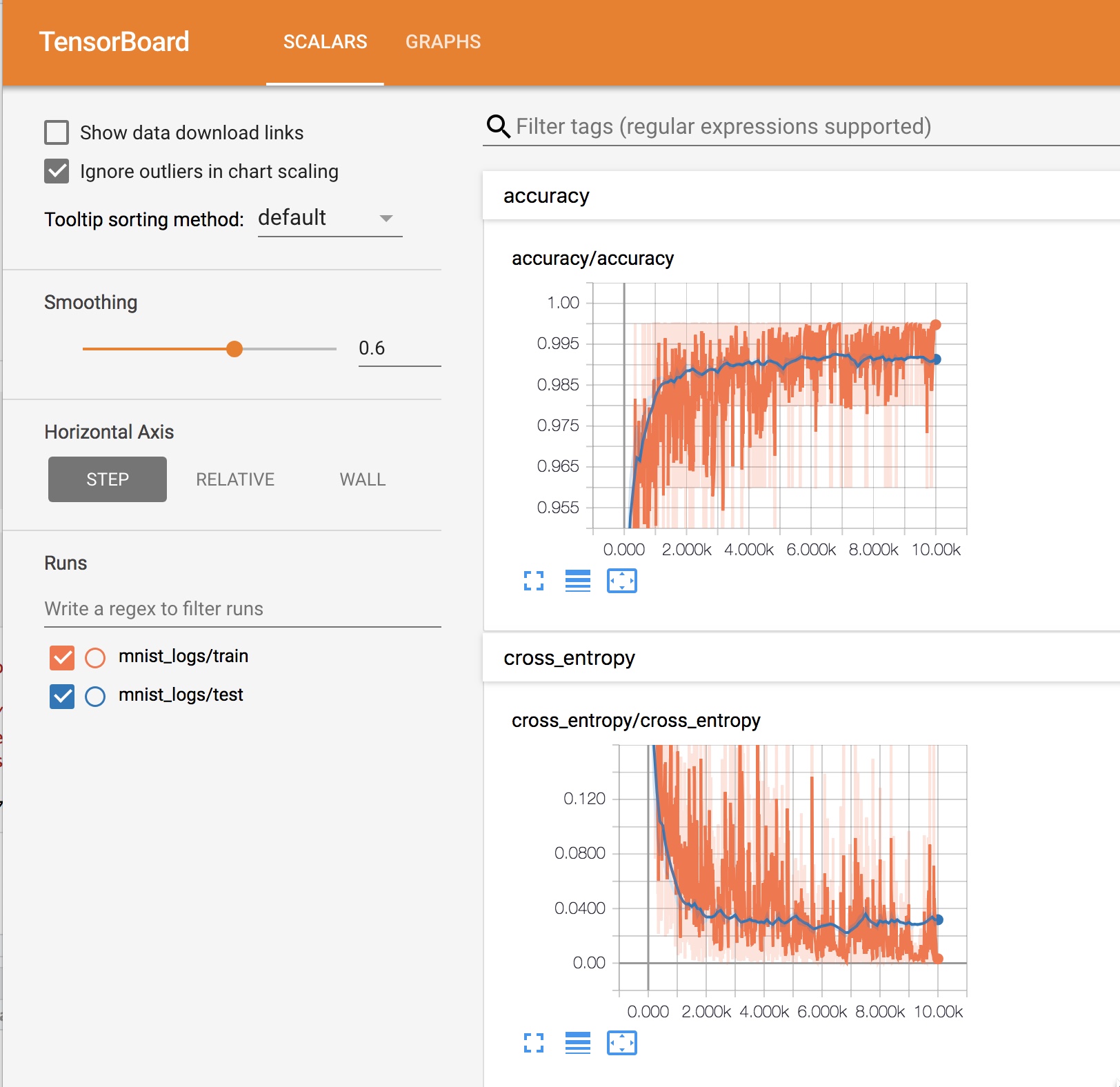
The first alternative uses the same number and type of layers – with the same dimensions – as the tutorial. The loss optimization function was changed from the “GradientDescentOptimizer” to “AdamOptimizer”. All other parameters were kept the same as the tutorial. This change resulted in a test set accuracy of 98.99 percent test set accuracy at 2,000 iterations and 99.20 percent test set accuracy at 10,000 iterations. The R code for this alternative is available for inspection, and is labeled “hw9\_mnist\_part2\_adam.Rmd”.

The following two screen captures from R-Studio show test set accuracy for this alternative at 2,000 iterations and 10,000 iterations, respectively.



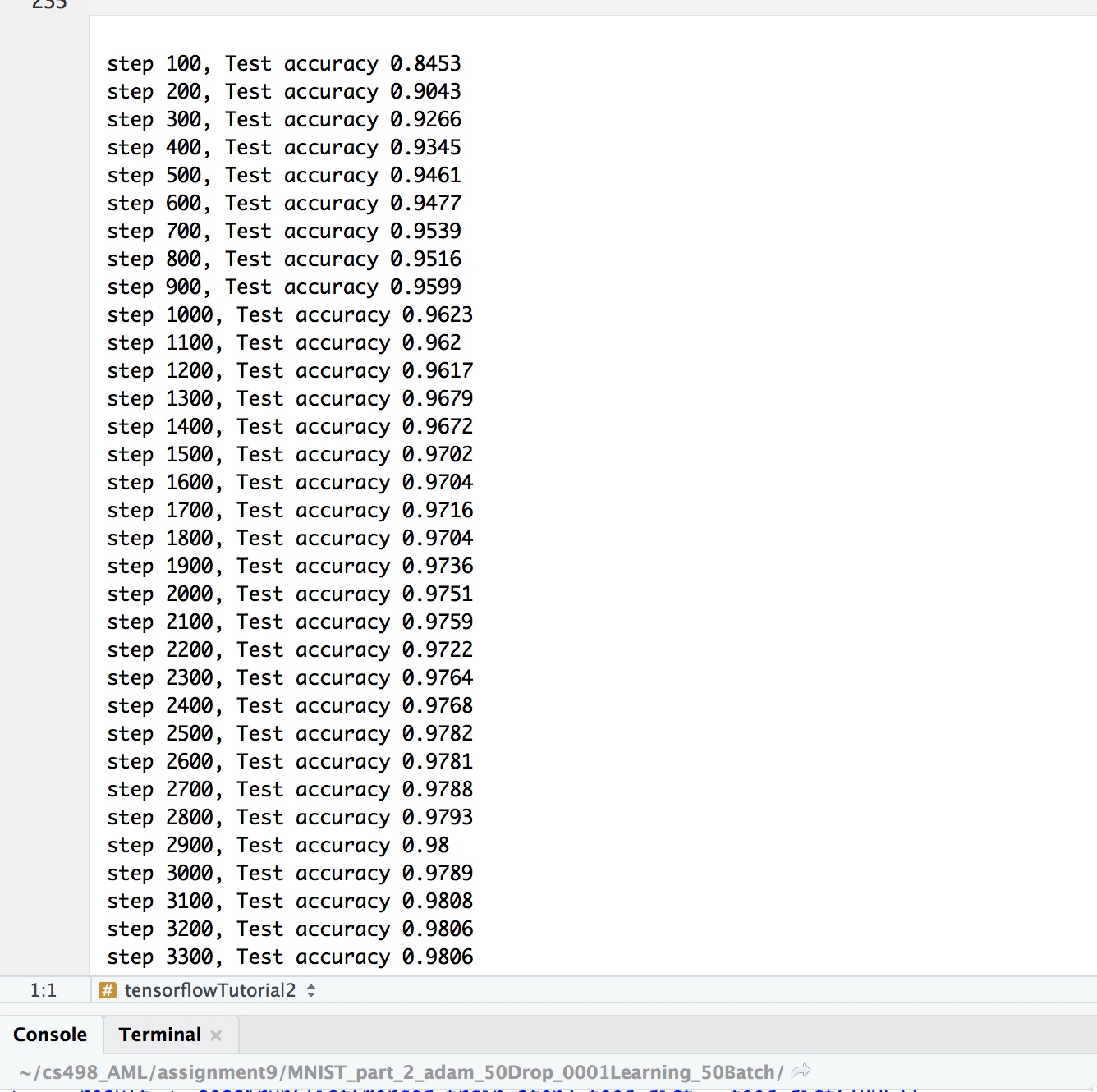


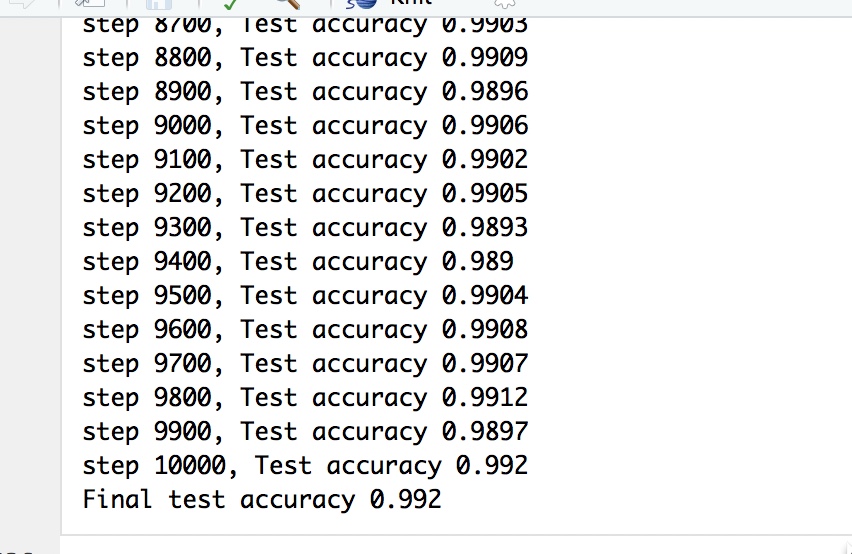
The trajectory of the increase in test set accuracy is shown below in the following screen capture from TensorBoard. The slope of the test set accuracy curve (blue line) rises sharply starting at iteration 100 and begins to flatten out at 1,000 iterations. A larger version of this TensorBoard plot is available for inspection and is labeled “tesorboard\_part2\_adam.jpeg”.



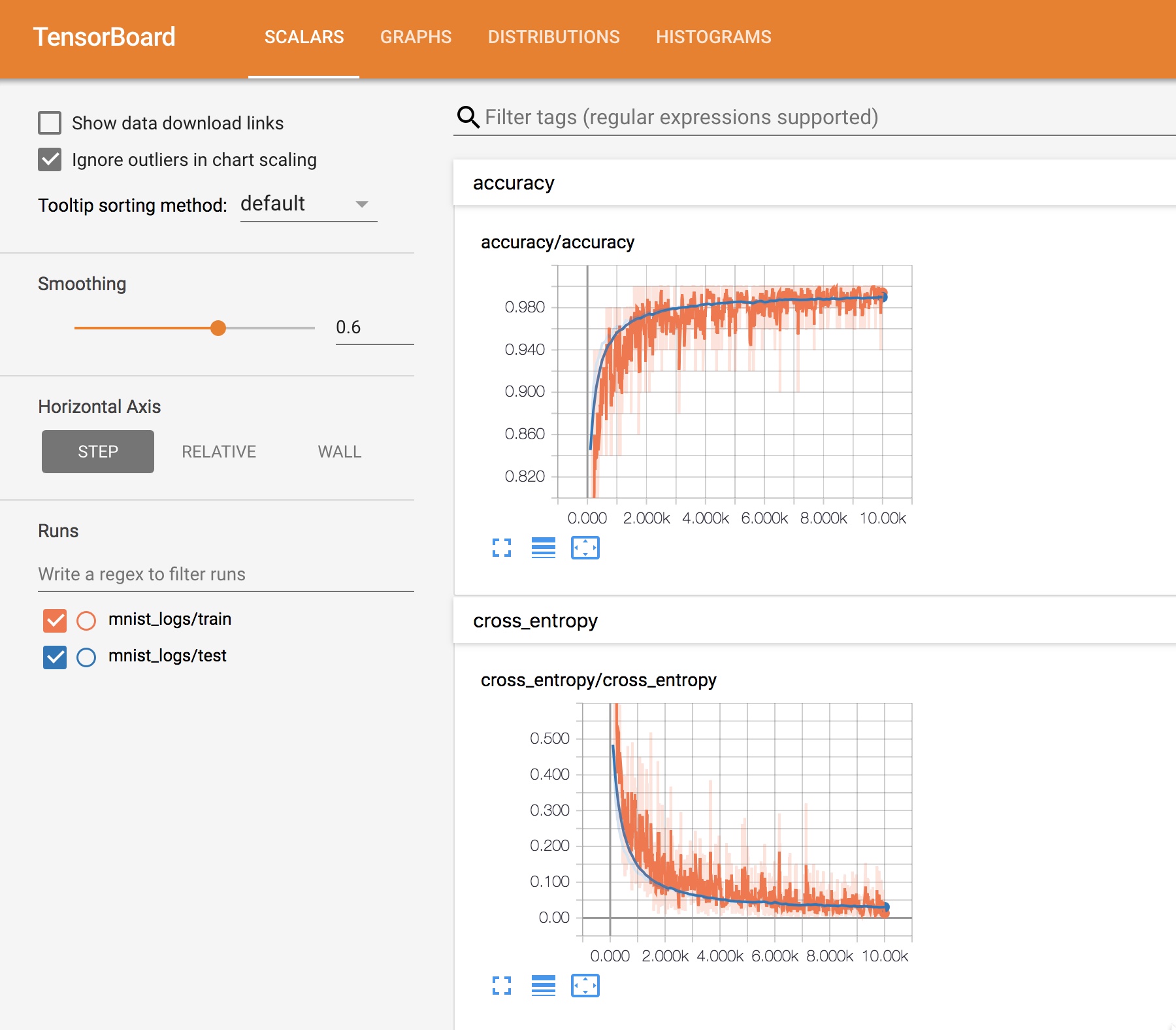
The second alternative is a variation on the first alternative. This alternative uses the “AdamOptimizer” and the same number and types of layers as the tutorial – including the same layer dimensions as the tutorial. The “learning rate” is changed from 0.001 to 0.0001. “Keep probability” is changed from 0.4 to 0.5 and the “batch size” Is changed from 100 to 50. These changes result in test set accuracy of 97.51 percent at 2,000 iterations and 99.20 percent at 10,000 iterations. The R-Code for this alternative is available for inspection and is labeled “hw9\_mnist\_part\_2\_adam\_50Drop\_0001Learning\_50Batch.Rmd”.

The following two screen captures from R-Studio show test set accuracy at 2,000 iterations and 10,000 iterations, respectively.



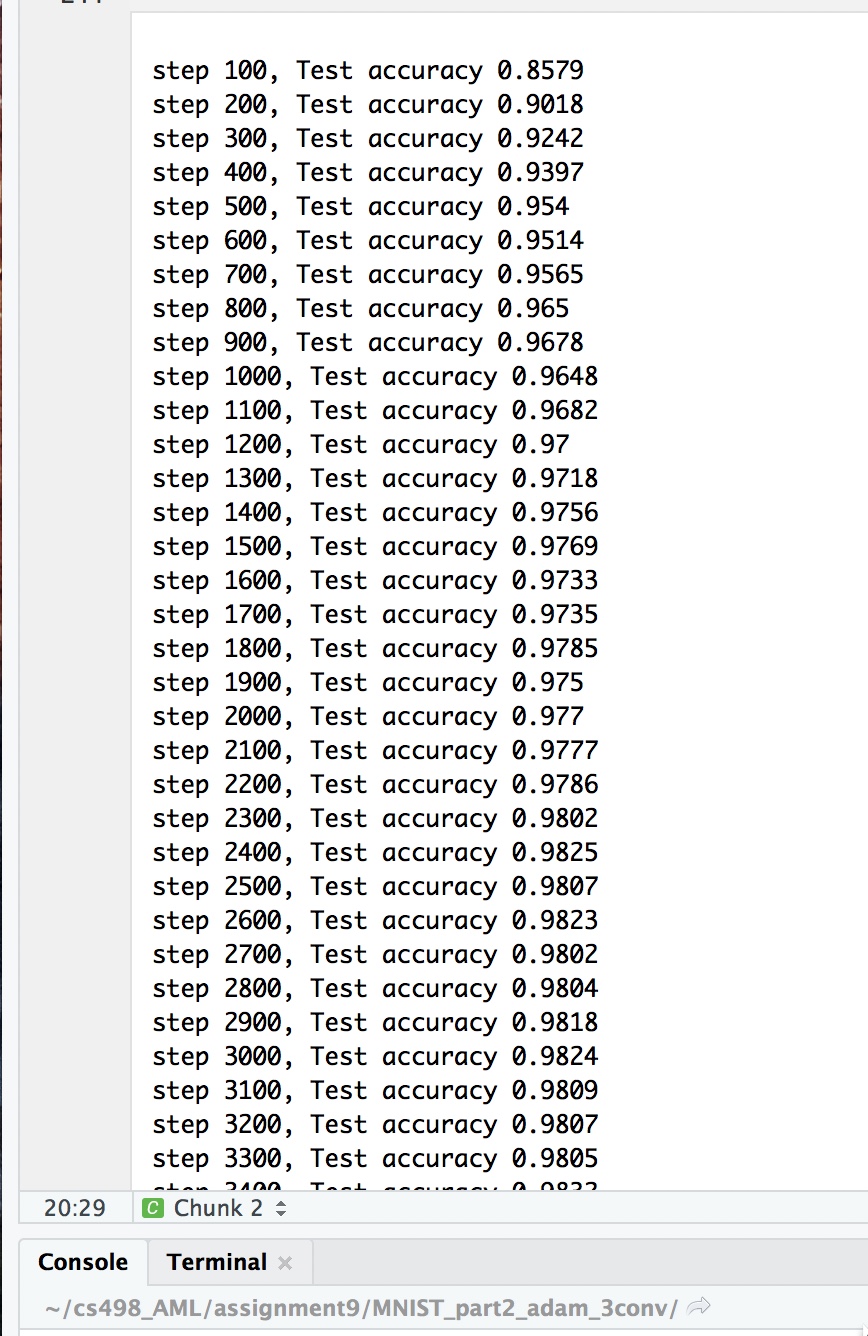


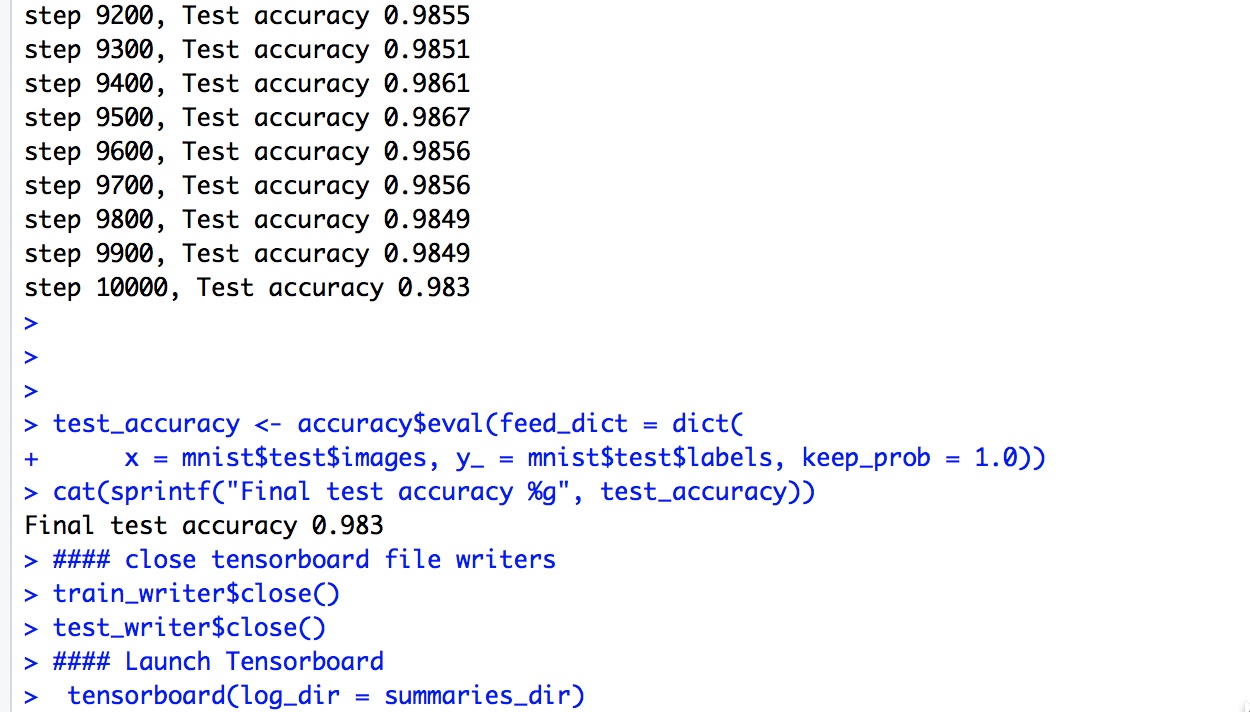
The trajectory of the increase in test set accuracy is similar to alternative 1. One difference is that alternative 2’s test set accuracy trajectory is smoother than alternative 1’s test set accuracy trajectory. The following screen capture from TensorBoard illustrates this fact. A larger version of this TensorBoard accuracy plot is available for inspection and is labeled “tensorboard\_part\_2\_adam\_50Drop\_0001Learning\_50Batch.jpeg”.



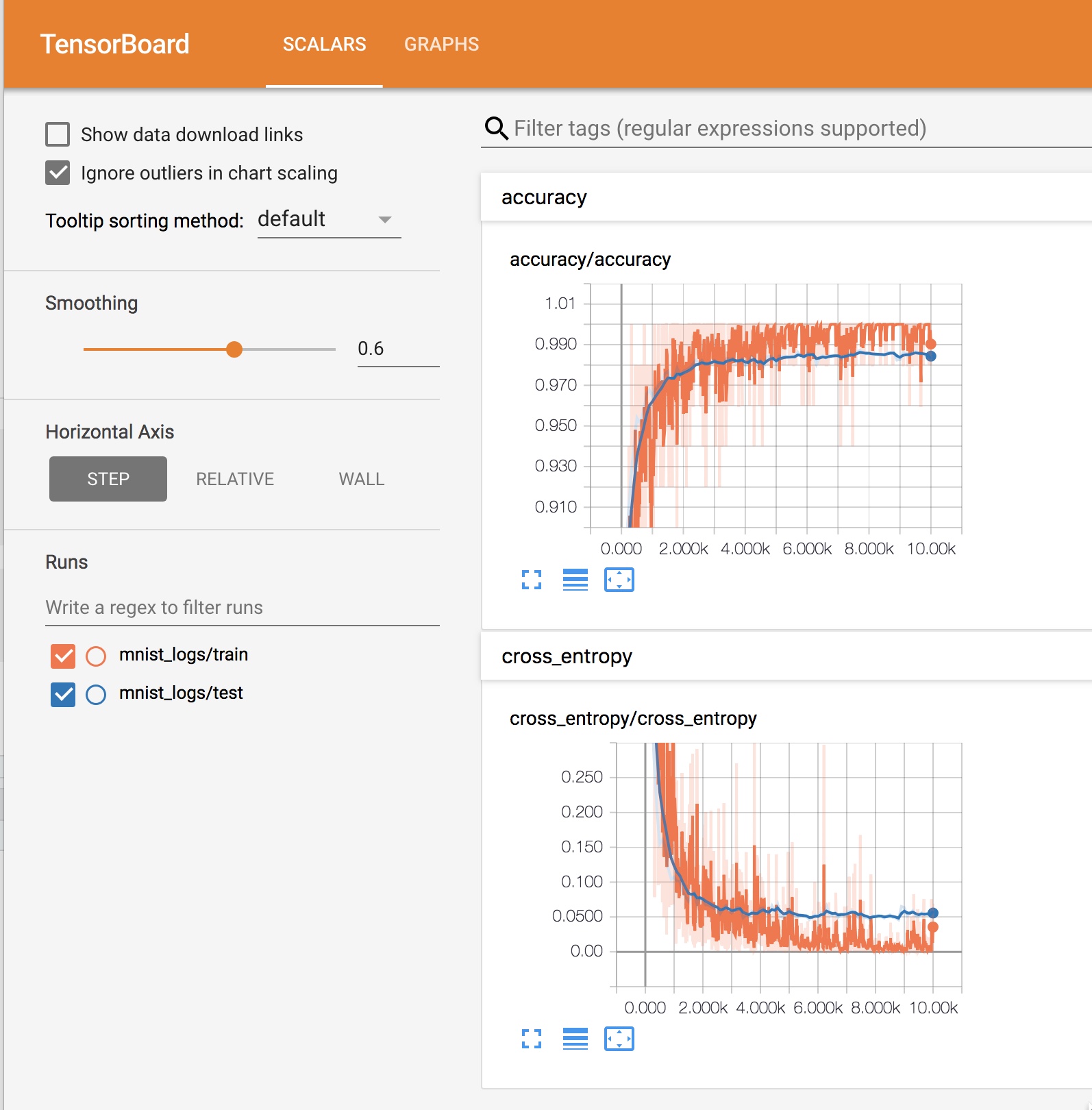
Alternative three modifies alternative 2 by adding a third convolutional layer and removing the two max pooling layers. The kernel dimension of the first convolutional layer is changed from shape(5L, 5L, 1L, 32L) to shape(5L, 5L, 8L, 8L). The kernel dimension of the second convolutional layer is changed from shape(5L, 5L, 32L, 64L) to shape(5L, 5L, 8L, 8L). The kernel dimension of the third convolutional layer is set to shape(5L, 5L, 8L, 8L). Test set accuracy Is 97.70 percent and 98.30 percent at 2,000 and 10,000 iterations respectively. The R code for this alternative is available for inspection and is labeled “hw9\_mnist\_part2\_adam\_3conv.Rmd”.

The following two screen captures show the test set accuracy at 2,000 and 10,000 iterations, respectively.





The trajectory of the increase in test-accuracy is shown in the following screen capture from TensorBoard. The trajectory of the test set accuracy curve is steep over the first 1500 iterations and flattens out at 5000 iterations. A larger version of this TensorBoard accuracy plot is available for inspection and is labeled “tensorboard\_part2\_adam\_3conv.jpeg”.



Alternative four is a variation on alternative 3. A fourth convolutional layer is added with a kernel dimension of shape(5L, 5L, 8L, 8L). The test set accuracy at iterations 2,000 and 10,000 are virtually the same as alternative 3. The R code for this alternative is available for inspection and is labeled “hw9\_mnist\_part2\_4conv\_adam.Rmd”.

The test set accuracy at 2,000 iterations and 10,000 iterations and the TensorBoard plot of the test set accuracy trajectory are shown below. A larger version of this TensorBoard accuracy plot is available for inspection and is labeled “tensorboard\_part2\_4conv\_adam.jpeg”.

