

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 ($a^{[1]}$): $h=28, w=28, \text{channels}=32$
 - Kernel size: $h=2, w=2, \text{channels}=1$
 - # kernels: 32
 - stride: 2
 - Dim of each training output element ($a^{[2]}$): $h=14, w=14, \text{channels}=32$
- Convolutional Layer 2:
 - Dim of each training input element ($a^{[2]}$): $h=14, w=14, \text{channels}=32$
 - kernel size: $h=5, w=5, \text{channels}=32$
 - # kernels: 64
 - stride: 1
 - padding: 2 (e.g., "SAME")
 - bias: 64
 - activation function: RELU
 - Dim of each training output element ($a^{[3]}$): $h=14, w=14, \text{channels}=64$
- Pooling Layer 2:
 - Dim of each training input element ($a^{[3]}$): $h=14, w=14, \text{channels}=64$
 - Kernel size: $h=2, w=2, \text{channels}=1$
 - # kernels: 64
 - stride: 2
 - Dim of each training output element ($a^{[4]}$): $h=7, w=7, \text{channels}=64$
- 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.

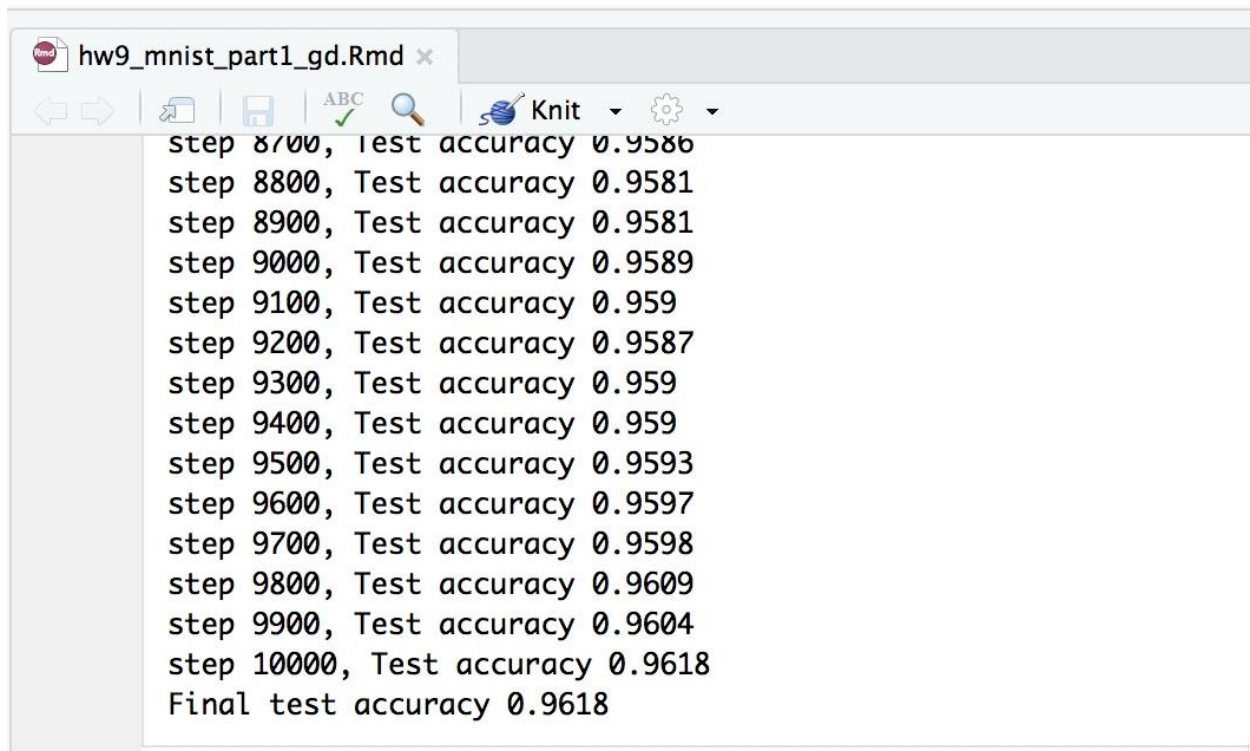
```
step 100, Test accuracy 0.5336
step 200, Test accuracy 0.6597
step 300, Test accuracy 0.7267
step 400, Test accuracy 0.7923
step 500, Test accuracy 0.8117
step 600, Test accuracy 0.8351
step 700, Test accuracy 0.853
step 800, Test accuracy 0.8639
step 900, Test accuracy 0.8728
step 1000, Test accuracy 0.8794
step 1100, Test accuracy 0.8868
step 1200, Test accuracy 0.8904
step 1300, Test accuracy 0.8963
step 1400, Test accuracy 0.8991
step 1500, Test accuracy 0.9011
step 1600, Test accuracy 0.9047
step 1700, Test accuracy 0.9084
step 1800, Test accuracy 0.9114
step 1900, Test accuracy 0.9136
step 2000, Test accuracy 0.9151
step 2100, Test accuracy 0.9185
step 2200, Test accuracy 0.9212
step 2300, Test accuracy 0.9242
step 2400, Test accuracy 0.9225
step 2500, Test accuracy 0.9282
step 2600, Test accuracy 0.9285
step 2700, Test accuracy 0.9297
step 2800, Test accuracy 0.9314
step 2900, Test accuracy 0.9328
step 3000, Test accuracy 0.9333
step 3100, Test accuracy 0.9335
step 3200, Test accuracy 0.9353
step 3300, Test accuracy 0.9362
```

7:1 (Top Level) ↕

Console Terminal x

~/cs498_AML/assignment9/MNIST_part1_gd/ ➔

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.


A screenshot of the R-Studio console window. The title bar shows a file named 'hw9_mnist_part1_gd.Rmd'. The console displays a series of log messages for steps 8700 through 10000, showing test accuracy values. The accuracy starts at 0.9586 for step 8700 and increases to 0.9618 by step 10000. The final line states 'Final test accuracy 0.9618'.

```
step 8700, test accuracy 0.9586
step 8800, Test accuracy 0.9581
step 8900, Test accuracy 0.9581
step 9000, Test accuracy 0.9589
step 9100, Test accuracy 0.959
step 9200, Test accuracy 0.9587
step 9300, Test accuracy 0.959
step 9400, Test accuracy 0.959
step 9500, Test accuracy 0.9593
step 9600, Test accuracy 0.9597
step 9700, Test accuracy 0.9598
step 9800, Test accuracy 0.9609
step 9900, Test accuracy 0.9604
step 10000, Test accuracy 0.9618
Final test accuracy 0.9618
```

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](#)”.

☐ Show data download links☒ Ignore outliers in chart scalingTooltip sorting method: **default** ▼

Smoothing

 0.6

Horizontal Axis

STEP

RELATIVE

WALL

Runs

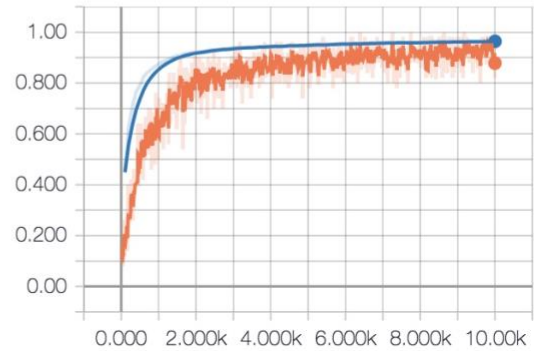
Write a regex to filter runs

☒ minst_logs/train☒ minst_logs/test

🔍 Filter tags (regular expressions supported)

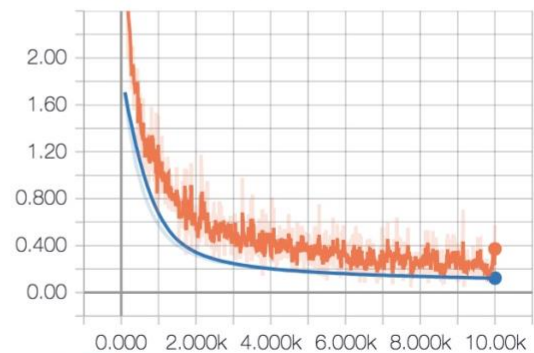
accuracy

accuracy/accuracy



cross_entropy

cross_entropy/cross_entropy



Part 1-B

We identified four different alternatives to the CNN architecture that outperformed 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.


```
step 100, Test accuracy 0.938
step 200, Test accuracy 0.9617
step 300, Test accuracy 0.968
step 400, Test accuracy 0.9751
step 500, Test accuracy 0.9658
step 600, Test accuracy 0.9778
step 700, Test accuracy 0.9786
step 800, Test accuracy 0.982
step 900, Test accuracy 0.9822
step 1000, Test accuracy 0.9866
step 1100, Test accuracy 0.9865
step 1200, Test accuracy 0.9869
step 1300, Test accuracy 0.9862
step 1400, Test accuracy 0.9858
step 1500, Test accuracy 0.987
step 1600, Test accuracy 0.9851
step 1700, Test accuracy 0.9885
step 1800, Test accuracy 0.9869
step 1900, Test accuracy 0.9887
step 2000, Test accuracy 0.9899
step 2100, Test accuracy 0.9885
step 2200, Test accuracy 0.9896
step 2300, Test accuracy 0.9876
step 2400, Test accuracy 0.987
step 2500, Test accuracy 0.9872
step 2600, Test accuracy 0.9886
step 2700, Test accuracy 0.9897
step 2800, Test accuracy 0.9891
step 2900, Test accuracy 0.9886
step 3000, Test accuracy 0.9872
step 3100, Test accuracy 0.9909
step 3200, Test accuracy 0.9913
step 3300, Test accuracy 0.9908
step 3400, Test accuracy 0.9903
```

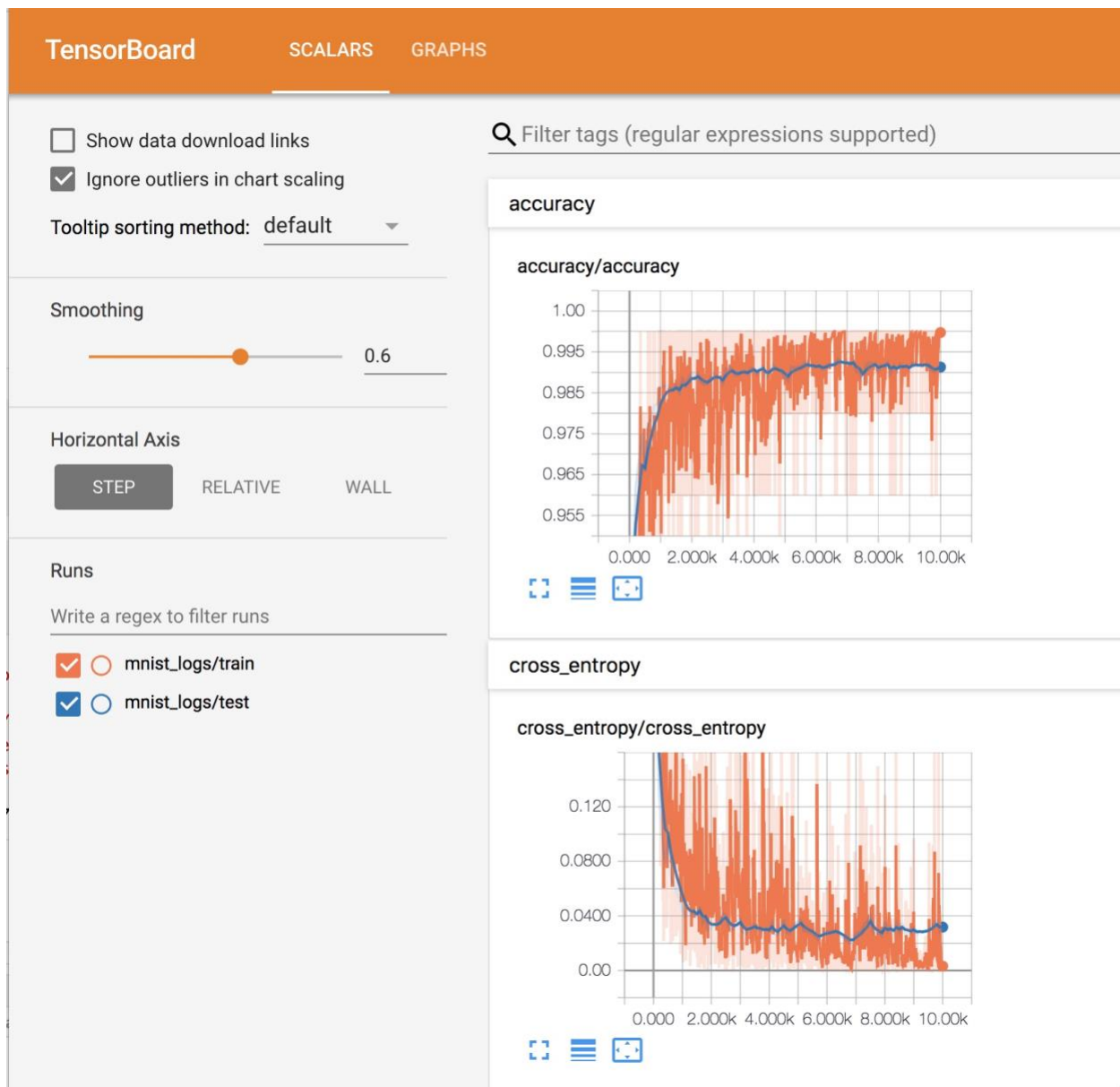
245:38  Chunk 18 ↕

Console Terminal ×

~/cs498_AML/assignment9/MNIST_part2_adam_40Drop_001Learning_100Batch/ ↗

```
/cst200_files/assignments/mnist_part2_adam_TOTOP_v0/learning_100batch/
step 9200, Test accuracy 0.992
step 9300, Test accuracy 0.9917
step 9400, Test accuracy 0.9918
step 9500, Test accuracy 0.9919
step 9600, Test accuracy 0.9918
step 9700, Test accuracy 0.9905
step 9800, Test accuracy 0.99
step 9900, Test accuracy 0.9909
step 10000, Test accuracy 0.992
>
>
>
> test_accuracy <- accuracy$eval(feed_dict = dict(
+   x = mnist$test$images, y_ = mnist$test$labels, keep_prob = 1.0))
> cat(sprintf("Final test accuracy %g", test_accuracy))
Final test accuracy 0.992
> ##### close tensorboard file writers
> train_writer$close()
> test_writer$close()
> ##### Launch Tensorboard
> tensorboard(log_dir = summaries_dir)
```

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.

```
step 100, Test accuracy 0.8453
step 200, Test accuracy 0.9043
step 300, Test accuracy 0.9266
step 400, Test accuracy 0.9345
step 500, Test accuracy 0.9461
step 600, Test accuracy 0.9477
step 700, Test accuracy 0.9539
step 800, Test accuracy 0.9516
step 900, Test accuracy 0.9599
step 1000, Test accuracy 0.9623
step 1100, Test accuracy 0.962
step 1200, Test accuracy 0.9617
step 1300, Test accuracy 0.9679
step 1400, Test accuracy 0.9672
step 1500, Test accuracy 0.9702
step 1600, Test accuracy 0.9704
step 1700, Test accuracy 0.9716
step 1800, Test accuracy 0.9704
step 1900, Test accuracy 0.9736
step 2000, Test accuracy 0.9751
step 2100, Test accuracy 0.9759
step 2200, Test accuracy 0.9722
step 2300, Test accuracy 0.9764
step 2400, Test accuracy 0.9768
step 2500, Test accuracy 0.9782
step 2600, Test accuracy 0.9781
step 2700, Test accuracy 0.9788
step 2800, Test accuracy 0.9793
step 2900, Test accuracy 0.98
step 3000, Test accuracy 0.9789
step 3100, Test accuracy 0.9808
step 3200, Test accuracy 0.9806
step 3300, Test accuracy 0.9806
```

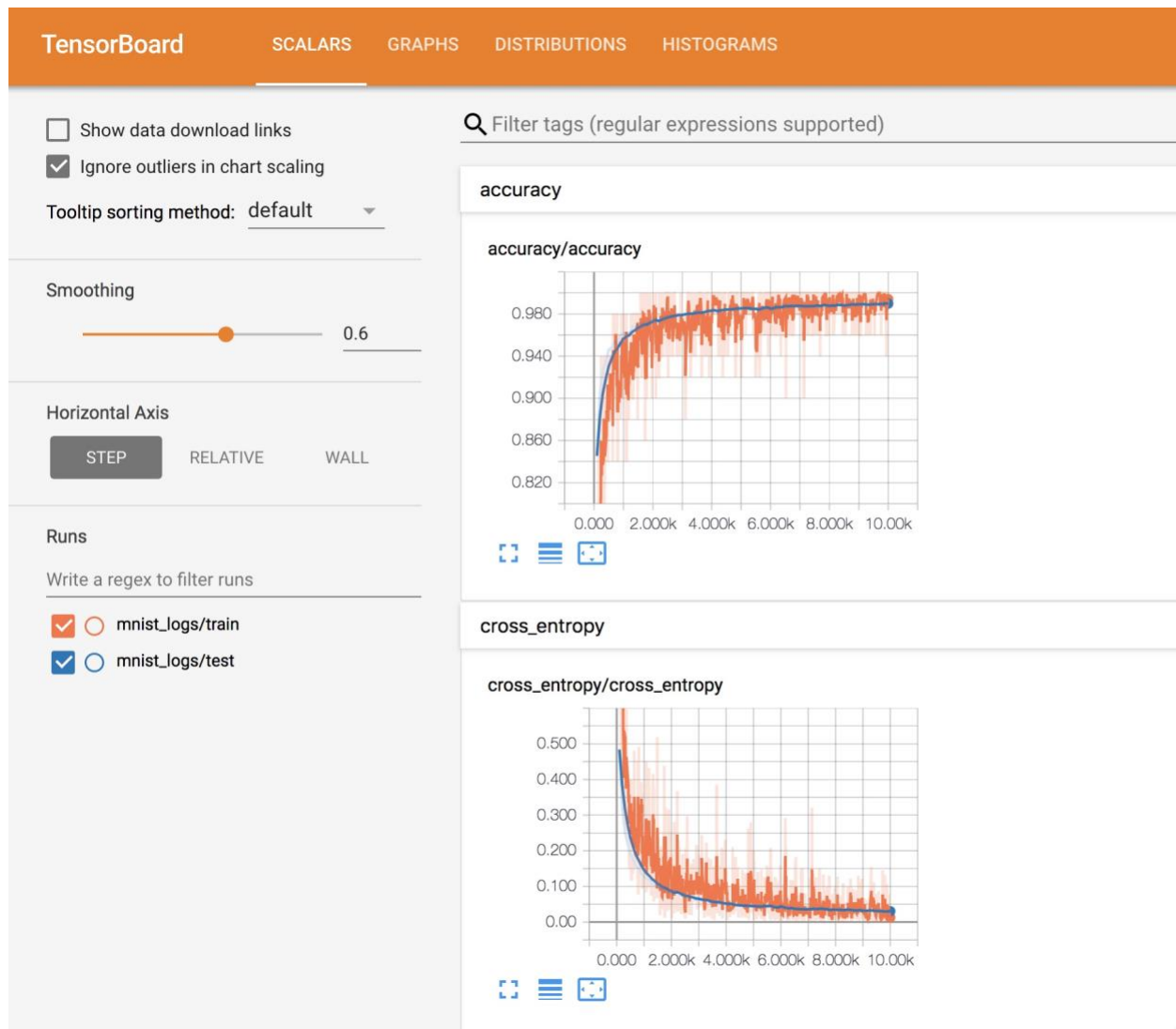
1:1 # tensorflowTutorial2 ↕

Console Terminal ×

~/cs498_AML/assignment9/MNIST_part_2_adam_50Drop_0001Learning_50Batch/ ➡

```
step 8700, Test accuracy 0.9903  
step 8800, Test accuracy 0.9909  
step 8900, Test accuracy 0.9896  
step 9000, Test accuracy 0.9906  
step 9100, Test accuracy 0.9902  
step 9200, Test accuracy 0.9905  
step 9300, Test accuracy 0.9893  
step 9400, Test accuracy 0.989  
step 9500, Test accuracy 0.9904  
step 9600, Test accuracy 0.9908  
step 9700, Test accuracy 0.9907  
step 9800, Test accuracy 0.9912  
step 9900, Test accuracy 0.9897  
step 10000, Test accuracy 0.992  
Final test accuracy 0.992
```


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.

```
step 100, Test accuracy 0.8579
step 200, Test accuracy 0.9018
step 300, Test accuracy 0.9242
step 400, Test accuracy 0.9397
step 500, Test accuracy 0.954
step 600, Test accuracy 0.9514
step 700, Test accuracy 0.9565
step 800, Test accuracy 0.965
step 900, Test accuracy 0.9678
step 1000, Test accuracy 0.9648
step 1100, Test accuracy 0.9682
step 1200, Test accuracy 0.97
step 1300, Test accuracy 0.9718
step 1400, Test accuracy 0.9756
step 1500, Test accuracy 0.9769
step 1600, Test accuracy 0.9733
step 1700, Test accuracy 0.9735
step 1800, Test accuracy 0.9785
step 1900, Test accuracy 0.975
step 2000, Test accuracy 0.977
step 2100, Test accuracy 0.9777
step 2200, Test accuracy 0.9786
step 2300, Test accuracy 0.9802
step 2400, Test accuracy 0.9825
step 2500, Test accuracy 0.9807
step 2600, Test accuracy 0.9823
step 2700, Test accuracy 0.9802
step 2800, Test accuracy 0.9804
step 2900, Test accuracy 0.9818
step 3000, Test accuracy 0.9824
step 3100, Test accuracy 0.9809
step 3200, Test accuracy 0.9807
step 3300, Test accuracy 0.9805
step 3400, Test accuracy 0.9822
```

20:29  Chunk 2 ▾

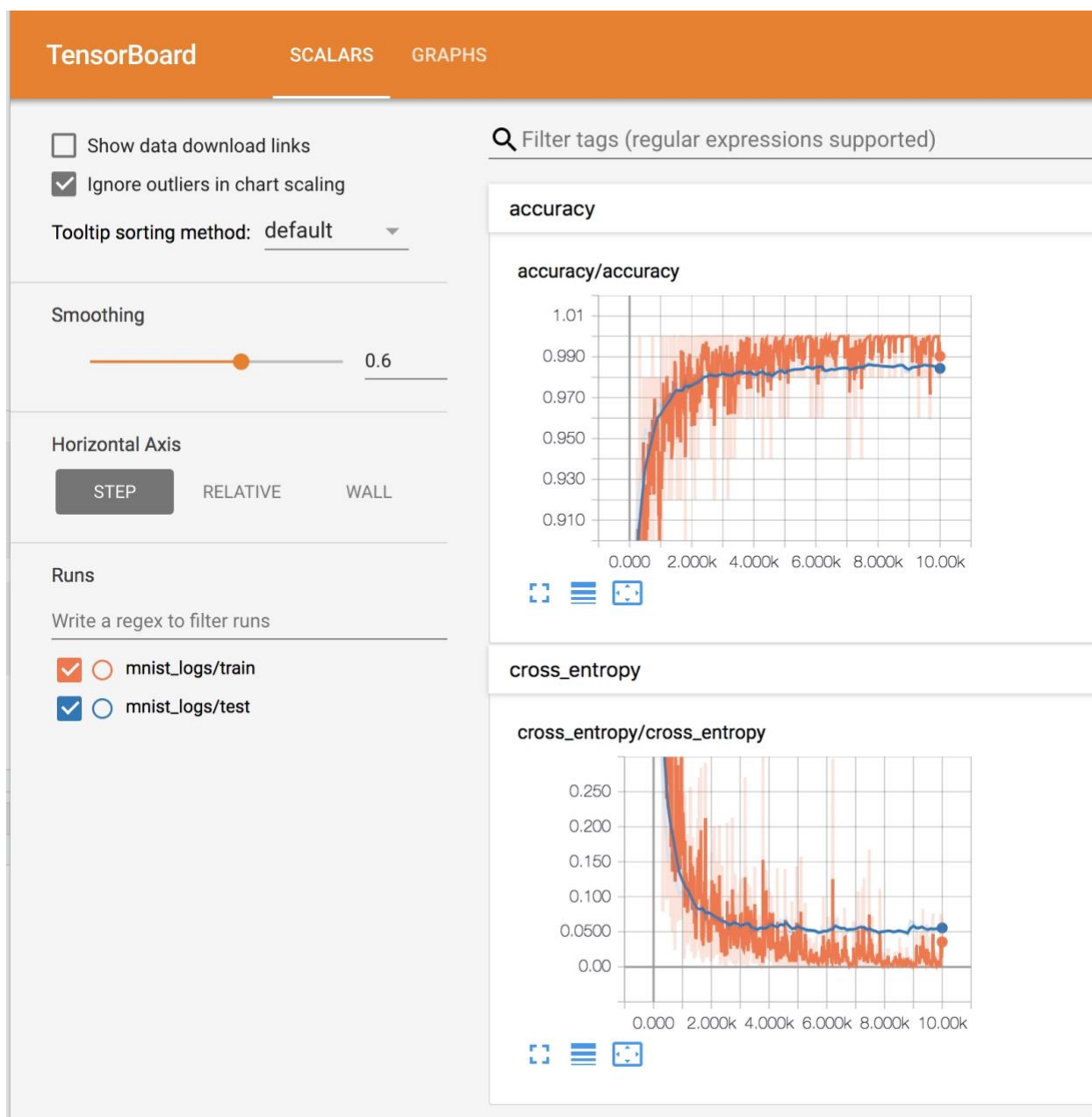
Console

Terminal 

~/cs498_AML/assignment9/MNIST_part2_adam_3conv/ 


```
step 9200, Test accuracy 0.9855
step 9300, Test accuracy 0.9851
step 9400, Test accuracy 0.9861
step 9500, Test accuracy 0.9867
step 9600, Test accuracy 0.9856
step 9700, Test accuracy 0.9856
step 9800, Test accuracy 0.9849
step 9900, Test accuracy 0.9849
step 10000, Test accuracy 0.983
>
>
>
> test_accuracy <- accuracy$eval(feed_dict = dict(
+   x = mnist$test$images, y_ = mnist$test$labels, keep_prob = 1.0))
> cat(sprintf("Final test accuracy %g", test_accuracy))
Final test accuracy 0.983
> #### close tensorboard file writers
> train_writer$close()
> test_writer$close()
> #### Launch Tensorboard
> tensorboard(log_dir = summaries_dir)
```

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".

Console

Terminal x

~/cs498_AML/assignment9/MNIST_Part2_4conv_adam/

```
+ summary <- result[[1]]  
+ train_writer$add_summary(summary, i)  
+ }  
+ }  
step 100, Test accuracy 0.8513  
step 200, Test accuracy 0.8966  
step 300, Test accuracy 0.9255  
step 400, Test accuracy 0.9272  
step 500, Test accuracy 0.923  
step 600, Test accuracy 0.9517  
step 700, Test accuracy 0.9547  
step 800, Test accuracy 0.9555  
step 900, Test accuracy 0.9577  
step 1000, Test accuracy 0.9593  
step 1100, Test accuracy 0.9595  
step 1200, Test accuracy 0.9632  
step 1300, Test accuracy 0.9714  
step 1400, Test accuracy 0.969  
step 1500, Test accuracy 0.9709  
step 1600, Test accuracy 0.9689  
step 1700, Test accuracy 0.9738  
step 1800, Test accuracy 0.9752  
step 1900, Test accuracy 0.9707  
step 2000, Test accuracy 0.975  
step 2100, Test accuracy 0.9773  
step 2200, Test accuracy 0.976  
step 2300, Test accuracy 0.9786  
step 2400, Test accuracy 0.9794  
step 2500, Test accuracy 0.9798  
step 2600, Test accuracy 0.9782  
step 2700, Test accuracy 0.9803  
step 2800, Test accuracy 0.9821  
step 2900, Test accuracy 0.9805  
step 3000, Test accuracy 0.9816  
step 3100, Test accuracy 0.9812  
step 3200, Test accuracy 0.9822  
step 3300, Test accuracy 0.9791
```

Console

Terminal x

~/cs498_AML/assignment9/MNIST_Part2_4conv_adam/ ↗

```
step 8400, Test accuracy 0.9865
step 8500, Test accuracy 0.9859
step 8600, Test accuracy 0.9839
step 8700, Test accuracy 0.9863
step 8800, Test accuracy 0.9857
step 8900, Test accuracy 0.9859
step 9000, Test accuracy 0.9871
step 9100, Test accuracy 0.9886
step 9200, Test accuracy 0.9883
step 9300, Test accuracy 0.9877
step 9400, Test accuracy 0.9875
step 9500, Test accuracy 0.9864
step 9600, Test accuracy 0.9864
step 9700, Test accuracy 0.986
step 9800, Test accuracy 0.9849
step 9900, Test accuracy 0.9849
step 10000, Test accuracy 0.9877
```

```
>
```

```
>
```

```
>
```

```
> test_accuracy <- accuracy$eval(feed_dict = dict(
+   x = mnist$test$images, y_ = mnist$test$labels, keep_prob = 1.0))
> cat(sprintf("Final test accuracy %g", test_accuracy))
Final test accuracy 0.9877
```

```
- ##### close tensorflow file writers
```

☐ Show data download links☒ Ignore outliers in chart scalingTooltip sorting method: **default** ▼

Smoothing

 0.6

Horizontal Axis

STEP

RELATIVE

WALL

Runs

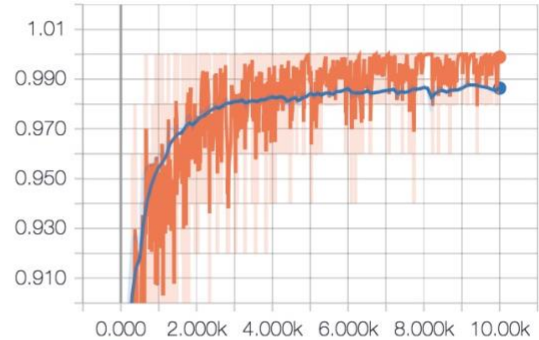
Write a regex to filter runs

☒ ☐ mnist_logs/train☒ ☐ mnist_logs/test

🔍 Filter tags (regular expressions supported)

accuracy

accuracy/accuracy



cross_entropy

cross_entropy/cross_entropy

