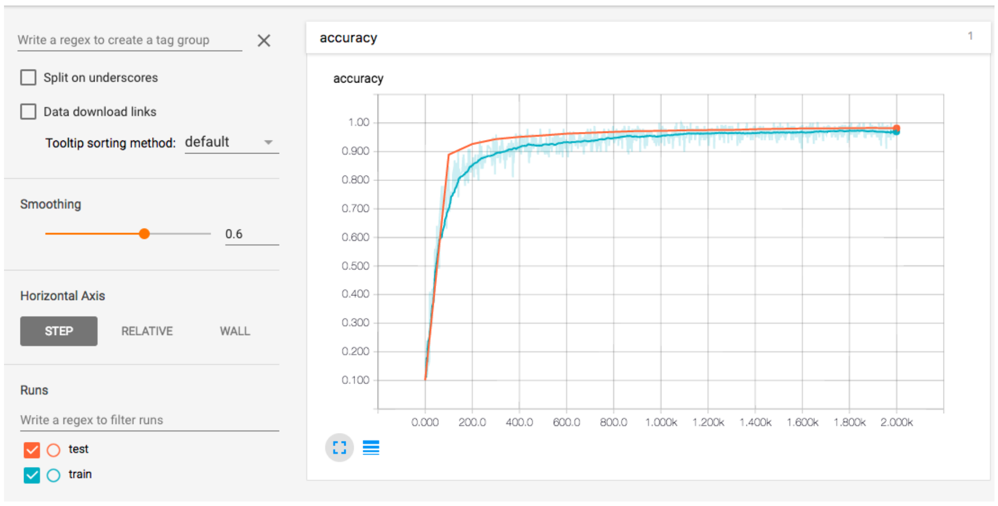
Dimytro Myronenko / myronen2
Bruno Calogero / brunoc2
Faisal Abu Jabal / abujaba2

Homework 8 - TensorFlow and Convolutional Neural Networks

Part I - Default Deep MNIST


Tooltip sorting method: default
Smoothing 0.6
Horizontal Axis: STEP
Runs: ✓ test, ✓ train

The accuracy of test & training is shown below.

The final test accuracy was: 0.9818
The final train accuracy was: 0.9700

Part II - Modified Deep MNIST

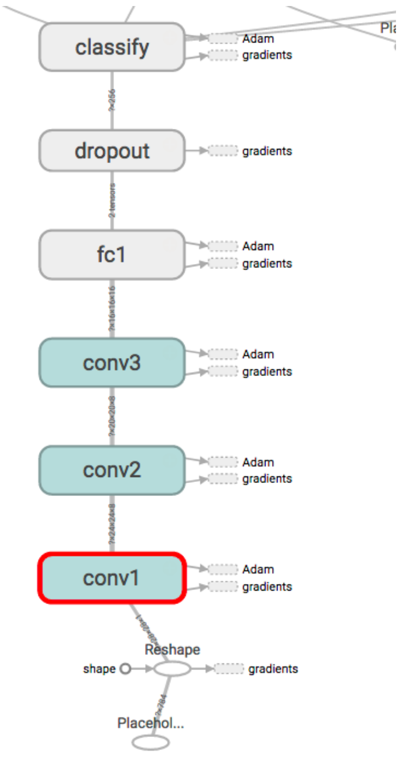
Version 1

For this part, we made 2 different versions. We dropped the max-pooling layers. Converted the first two convolutional layers to depth 8 and added one more on top of that of depth 16. We also changed each convolutional layer to be unpadded; thus after each iteration, the output images gets smaller in dimension. We also changed the fully connected layer to be from length 1024 to 256.

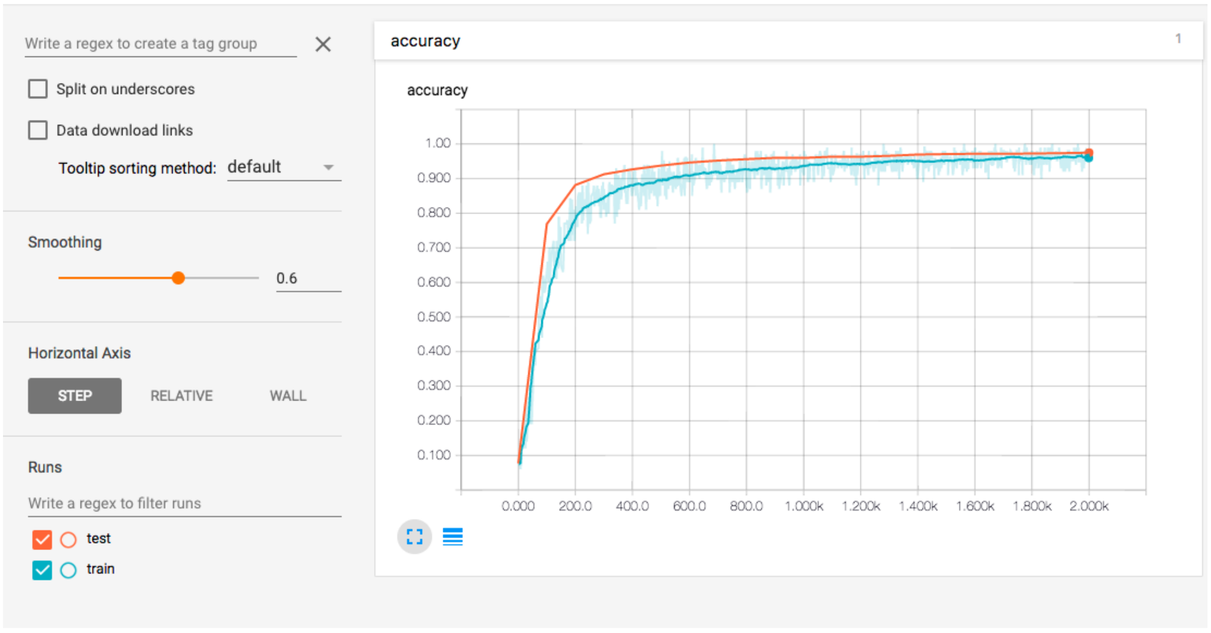
We did this to see if having more, yet shallower, convolutional layers is better than less layers that are deeper. Also having a fully-connected layer of length 1024 was felt like overkill.

Dimytro Myronenko / myronen2
Bruno Calogero / brunoc2
Faisal Abu Jabal / abujaba2

The new graph is shown below.



The accuracy of test & training is show below.



Write a regex to create a tag group

☐ Split on underscores

☐ Data download links

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

✓ test

✓ train

The final test accuracy was: 0.9750
The final train accuracy was: 0.9600

This version of the network did not perform that much worse; we would say not significantly worse. In addition, it ran much faster. One could say that this is an improvement because it only sacrificed a little bit of precision for a dramatic increase in

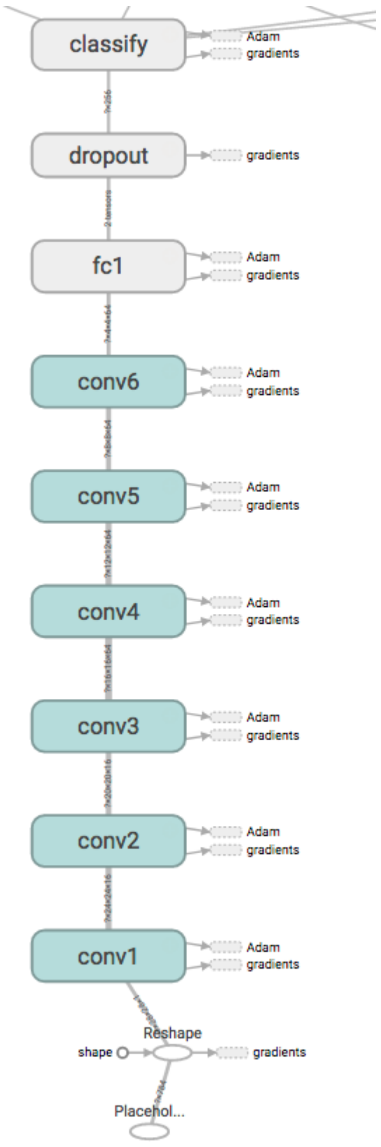
Dimytro Myronenko / myronen2
Bruno Calogero / brunoc2
Faisal Abu Jabal / abujaba2

speed. I believe that simply running more cycles will increase the accuracy while still being faster than the default.

Version 2

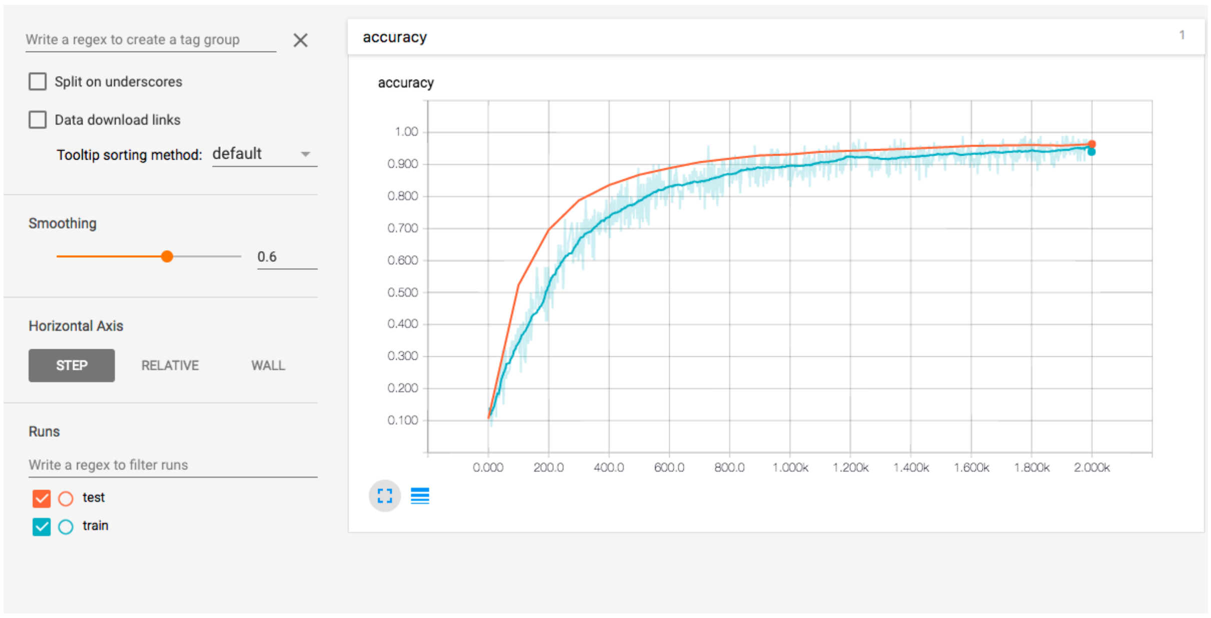
This part was somewhat inspired by this Computerphile YouTube video. The idea was to have 2 convolutional layers of depth 16 and 4 convolutional layers of depth 64. Once again, there was no max-pooling or padding to decrease the image dimension after each layer. Each layer had a filter size of 5x5 so each layer removed 4 pixels of each dimension. So $28 - (6 \times 4) = 4$. The final image output was only 4x4 pixels. Once again, having a 256 long fully-connected layer.

The new diagram is shown below.



The accuracy of test & training is show below.

Dimytro Myronenko / myronen2
Bruno Calogero / brunoc2
Faisal Abu Jabal / abujaba2


Write a regex to create a tag group

☐ Split on underscores

☐ Data download links

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: **STEP** RELATIVE WALL

Runs

Write a regex to filter runs

☑ test

☑ train

The final test accuracy was: 0.9638
The final train accuracy was: 0.9400

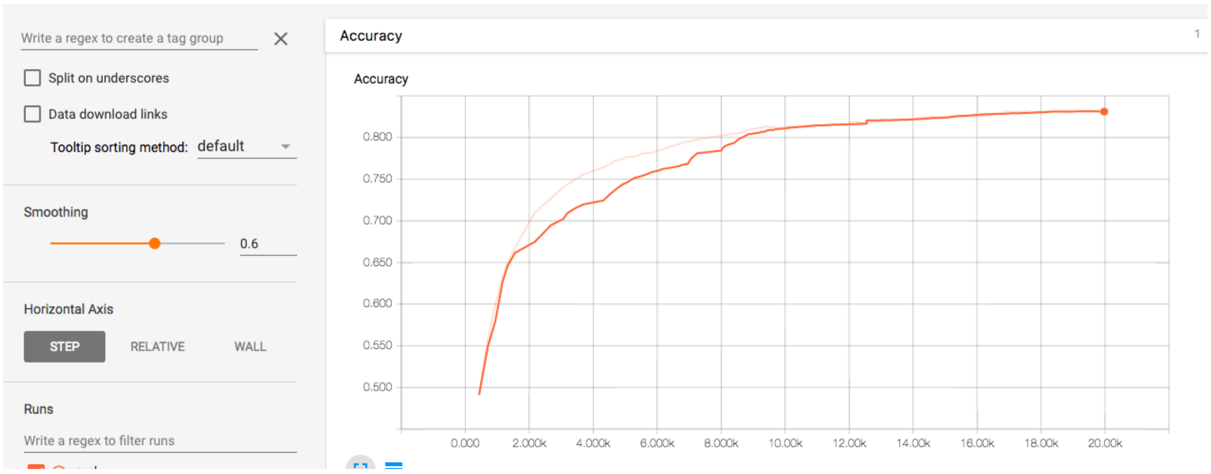
In conclusion, adding so much unneeded complexity actually decreased the accuracy. This version ran significantly longer for no added benefit. However, it is interesting to note that the test-accuracy curve for Version 2 is much smoother than Version 1 or the default network.

Part III - Default CIFAR10

Doing some preliminary testing, the first 1000 batches took 230 seconds to process, or $0.23\ \frac{s}{batch}$. This was done on an AWS machine to control for variations in CPU load. This was done a second time and the total time differed by less than 1 second. Since the training and eval (test) are ran as 2 separate processes, we modified the eval to run (approximately) every 100 batches (23 seconds). Accordingly, we modified the train step to save the checkpoints every 10 seconds. We wanted the checkpoints saved at least twice as often as the period of the eval script. This prevents the eval script from using the same checkpoint two times in a row due to fluctuations in either process.

At one point, the speed was close to $1.2\ \frac{s}{batch}$ but it eventually reverted to about $0.35\ \frac{s}{batch}$. Also while running this overnight, the logs showed that at one point, the speed was $4234\ \frac{s}{batch}$. We are not sure why this happened, but a possible explanation is that the OS took the processes out of context or there system ran out of memory. After 20,000 iterations here are the results:

Dimytro Myronenko / myronen2
Bruno Calogero / brunoc2
Faisal Abu Jabal / abujaba2


Write a regex to create a tag group

Split on underscores

Data download links

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis STEP RELATIVE WALL

Runs

Write a regex to filter runs

The final test accuracy was: 0.8311

Part IV - Modified CIFAR10

Due to weird performance issues in Part III, we wanted to explore how we can sacrifice accuracy for better running time. First, we changed the convolutional neural networks to be 32 deep instead of 64. Also we added another CNN-norm-pool layer. The first 2 pool nodes became **average pooling** but the third one is max pooling. Also the pool1 and norm1 nodes switched places to have more consistency.

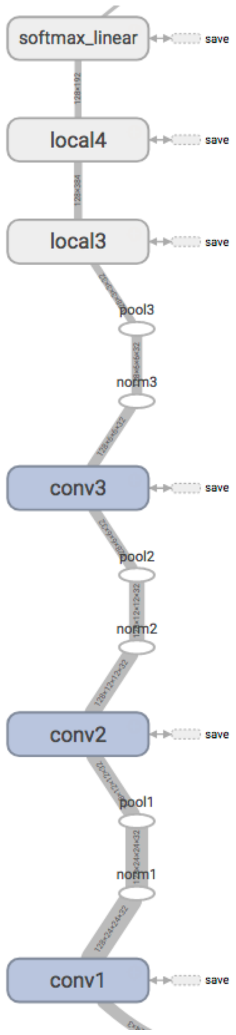
The **old** network was
CNN1→pool1→norm1→CNN2→norm2→pool2→....
The **new** network is
CNN1→norm1→pool1→CNN2→norm2→pool2→CNN3→norm3→pool3→...

We first ran the modified code on 20,000 iterations. The accuracy was 0.7911 but it ran much faster and with much less fluctuations per iteration than the default CIFAR10 code. Then we ran 40,000 iterations. Once again, it ran faster than the default but the difference was not as extreme as previously noted. However, the accuracy was close to that from the default code.

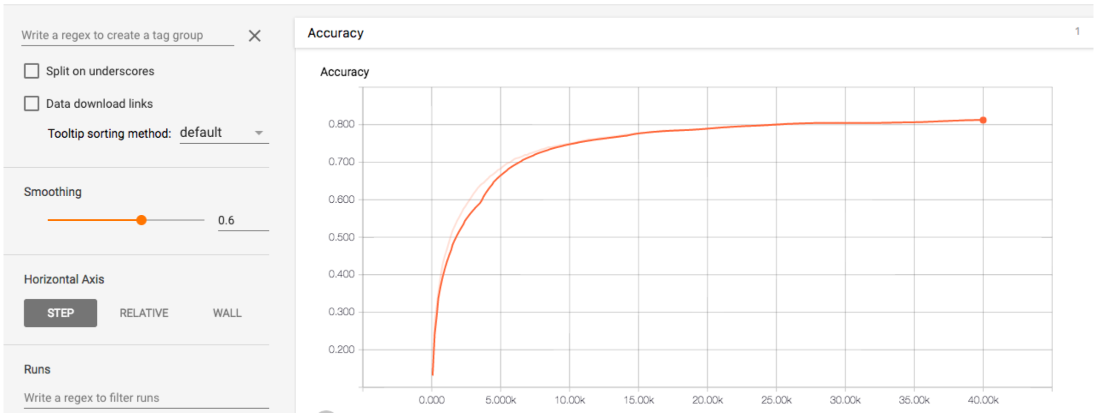
We believe that if the two versions (Part III and Part IV) ran for the same amount of time (independent of number of iterations) the modified version would have better accuracy. However, due to time constraints (and not wanting to pay any more for AWS services), we leave this observation unanswered.

Dimytro Myronenko / myronen2
Bruno Calogero / brunoc2
Faisal Abu Jabal / abujaba2

The new diagram is shown below:



The accuracy of test & training is shown below.


Write a regex to create a tag group

☐ Split on underscores

☐ Data download links

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

The final test accuracy was: 0.8121, clearly close to the default if not more promising!