Part1. Change hyper parameters of the Text Classification with new dataset

Algorithm:

Load positive and negative sentences from the raw data files.

Clean the text data using the same code as the original paper.

Pad each sentence to the maximum sentence length. Padding sentences to the same length is useful because it allows us to efficiently batch our data since each example in a batch must be of the same length.

Build a vocabulary index and map each word to an integer between 0 and 6,413 (the vocabulary size). Each sentence becomes a vector of integers.

**a. Optimizers**

Parameters:

ALLOW\_SOFT\_PLACEMENT=True

BATCH\_SIZE=32

CHECKPOINT\_EVERY=100

DEV\_SAMPLE\_PERCENTAGE=0.1

DROPOUT\_KEEP\_PROB=0.5

EMBEDDING\_DIM=128

EVALUATE\_EVERY=100

FILTER\_SIZES=3,4,5

L2\_REG\_LAMBDA=0.0

LOG\_DEVICE\_PLACEMENT=False

NUM\_CHECKPOINTS=5

NUM\_EPOCHS=100

NUM\_FILTERS=128

|  |  |  |
| --- | --- | --- |
| AdamOptimizer: | GradientDescentOptimizer: | RMSPropOptimizer: |
| Evaluation:  step 2800  loss 1.59227e+07  acc 0.86875 | Evaluation:  step 2800  loss 3.46284  acc 0.72449 | Evaluation:  step 2800  loss 457629  acc 0.806122 |
| Conclusion: AdamOptimizer with filter size(3,4,5) have the best accuracy(0.86875) | | |

**b. Filter size (fixed the optimizer with AdamOptimizer)**

Parameters:

ALLOW\_SOFT\_PLACEMENT=True

BATCH\_SIZE=32

CHECKPOINT\_EVERY=100

DEV\_SAMPLE\_PERCENTAGE=0.1

DROPOUT\_KEEP\_PROB=0.5

EMBEDDING\_DIM=128

EVALUATE\_EVERY=100

FILTER\_SIZES=1,2,3

L2\_REG\_LAMBDA=0.0

LOG\_DEVICE\_PLACEMENT=False

NUM\_CHECKPOINTS=5

NUM\_EPOCHS=100

NUM\_FILTERS=128

Loading data...

Vocabulary Size: 6413

Train/Dev split: 886/98

|  |  |  |
| --- | --- | --- |
| 3, 4, 5: | 6, 8, 10: | 1, 2, 3: |
| Evaluation:  step 2800  loss 1.59227e+07  acc 0.86875 | Evaluation:  step 2800  loss 3.68776e+07  acc 0.765306 | Evaluation:  step 2800  loss 457629  acc 0.806122 |
| Conclusion: AdamOptimizer with filter size(3,4,5) have the best accuracy(0.86875) | | |

**c. Number of filters:**

Parameters:

ALLOW\_SOFT\_PLACEMENT=True

BATCH\_SIZE=32

CHECKPOINT\_EVERY=100

DEV\_SAMPLE\_PERCENTAGE=0.1

DROPOUT\_KEEP\_PROB=0.5

EMBEDDING\_DIM=128

EVALUATE\_EVERY=100

FILTER\_SIZES=3,4,5

L2\_REG\_LAMBDA=0.0

LOG\_DEVICE\_PLACEMENT=False

NUM\_CHECKPOINTS=5

NUM\_EPOCHS=100

NUM\_FILTERS=64

|  |  |  |
| --- | --- | --- |
| 64: | 128: | 256: |
| Evaluation:  step 2800  loss 6.24601e+06  acc 0.826531 | Evaluation:  step 2800  loss 1.59227e+07  acc 0.86875 | Evaluation:  step 2800  loss 6.6569e+07  acc 0.785714 |
| Conclusion: AdamOptimizer with filter size(3,4,5) , number of filter=128 have the best accuracy(0. 86875) | | |

**d. Dropout probability:**

Parameters:

ALLOW\_SOFT\_PLACEMENT=True

BATCH\_SIZE=32

CHECKPOINT\_EVERY=100

DEV\_SAMPLE\_PERCENTAGE=0.1

DROPOUT\_KEEP\_PROB=0.5

EMBEDDING\_DIM=128

EVALUATE\_EVERY=100

FILTER\_SIZES=3,4,5

L2\_REG\_LAMBDA=0.0

LOG\_DEVICE\_PLACEMENT=False

NUM\_CHECKPOINTS=5

NUM\_EPOCHS=100

NUM\_FILTERS=128

|  |  |  |
| --- | --- | --- |
| 0.25: | 0.5: | 0.75: |
| Evaluation:  step 2800  loss 3.14969e+07  acc 0.806122 | Evaluation:  step 2800  loss 6.24601e+06  acc 0.826531 | Evaluation:  step 2800  loss 1.25843e+07  cc 0.77551 |
| Conclusion: AdamOptimizer with filter size(3,4,5), number of filter=128, dropout probability= 0.5 have the best accuracy(0.826531) | | |

**e. Batch size:**

Parameters:

ALLOW\_SOFT\_PLACEMENT=True

BATCH\_SIZE=64

CHECKPOINT\_EVERY=100

DEV\_SAMPLE\_PERCENTAGE=0.1

DROPOUT\_KEEP\_PROB=0.5

EMBEDDING\_DIM=128

EVALUATE\_EVERY=100

FILTER\_SIZES=3,4,5

L2\_REG\_LAMBDA=0.0

LOG\_DEVICE\_PLACEMENT=False

NEGATIVE\_DATA\_FILE=C:/Users/disfly/PycharmProjects/untitled/newdata2.txt

NUM\_CHECKPOINTS=5

NUM\_EPOCHS=100

NUM\_FILTERS=128

|  |  |  |
| --- | --- | --- |
| 32: | 64: | 128: |
| Evaluation:  step 2800  loss 6.24601e+06  acc 0.826531 | Evaluation:  step 1400  loss 8.95438e+06  acc 0.755102 | Evaluation:  step 700  loss 2.76903e+06  acc 0.734694 |
| Conclusion: AdamOptimizer with filter size(3,4,5), number of filter=128, dropout probability= 0.5 batch size=64 have the best accuracy(0.826531) | | |

**f. Number of epochs:**

Parameters:

ALLOW\_SOFT\_PLACEMENT=True

BATCH\_SIZE=32

CHECKPOINT\_EVERY=100

DEV\_SAMPLE\_PERCENTAGE=0.1

DROPOUT\_KEEP\_PROB=0.5

EMBEDDING\_DIM=128

EVALUATE\_EVERY=100

FILTER\_SIZES=3,4,5

L2\_REG\_LAMBDA=0.0

LOG\_DEVICE\_PLACEMENT=False

NEGATIVE\_DATA\_FILE=C:/Users/disfly/PycharmProjects/untitled/newdata2.txt

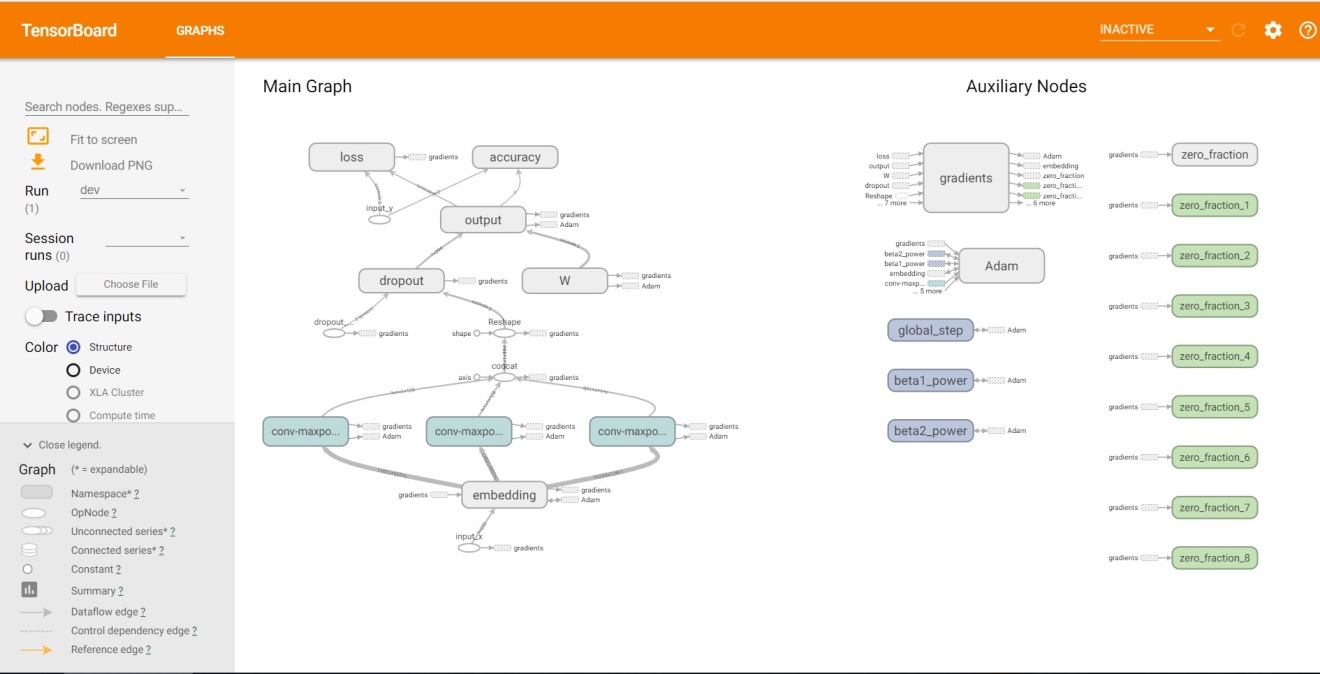
NUM\_CHECKPOINTS=5

NUM\_EPOCHS=200

NUM\_FILTERS=128

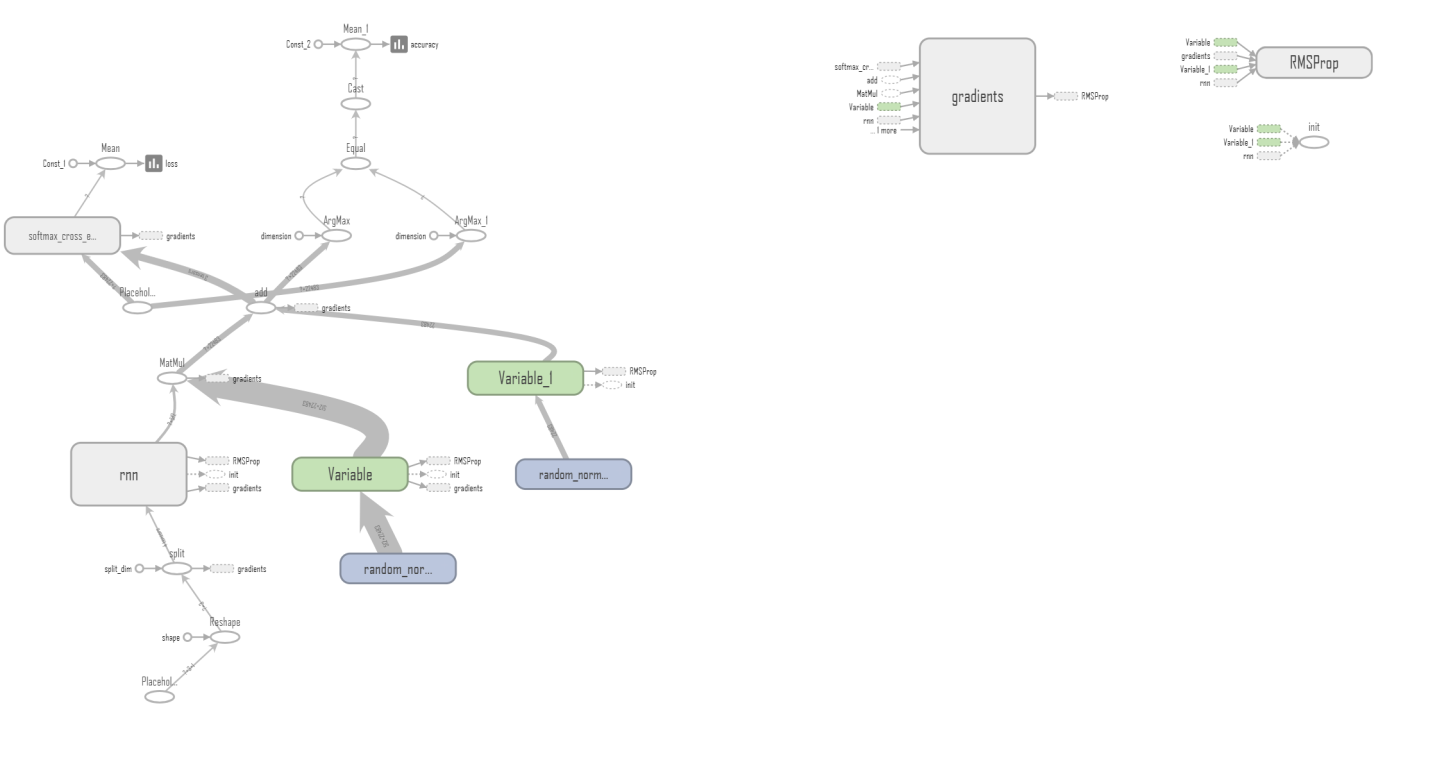
|  |  |  |
| --- | --- | --- |
| 50: | 100: | 200: |
| Evaluation:  step 1400  loss 1.02921e+07  acc 0.795918 | Evaluation:  step 2800  loss 6.24601e+06  acc 0.826531 | Evaluation:  step 5600  loss 3.19904e+07  acc 0.816327 |
| Conclusion: AdamOptimizer with filter size(3,4,5), number of filter=128, dropout probability= 0.5 batch size=64, number of epochs =100, have the best accuracy(0.826531) | | |

**Tensorboard:**



Part2. Implement the text classification with RNN/LSTM model, with a new dataset which is not used in the class

For part 2 I used a multi-layer LSTM with 2 layers. With both data set one and two I varied the learning rate (0.01, 0.001) as well as the n\_hidden (512, 1024) Below is the graph of the RNN.



### Conclusion

As you will see from the output, I did not receive an accuracy greater than about 7.00%. It's extremely low. This leads me to believe that a LSTM RNN is not the correct choice for predicting this type of data.

Part3. Compare the results of CNN and RNN/LSTM models, for the text classification (same dataset for 2 models to compare) and describe, which model is best for the text classification based on your results

Compare the results of CNN and RNN/LSTM models, for the text classification (same dataset for 2 models to compare) and describe, which model is best for the text classification based on your results :

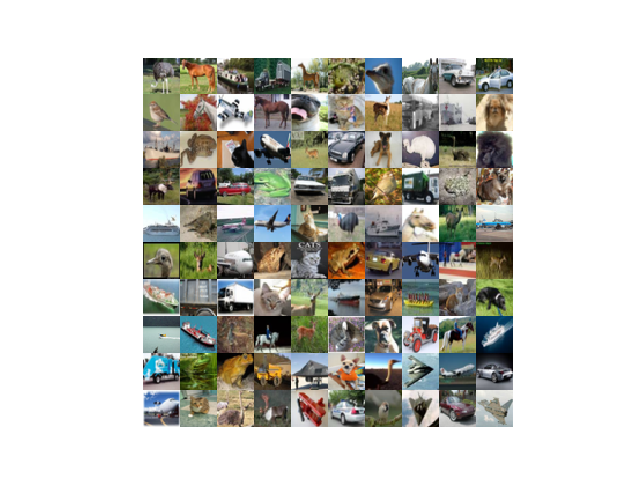
CNN: Conclusion: AdamOptimizer with filter size(3,4,5), number of filter=128, dropout probability= 0.5 batch size=64, number of epochs =100, have the best accuracy(0.826531)

RNN/LSTM: with 10000 iterations, the average loss was 11.035806, and the average accuracy was 3.80%

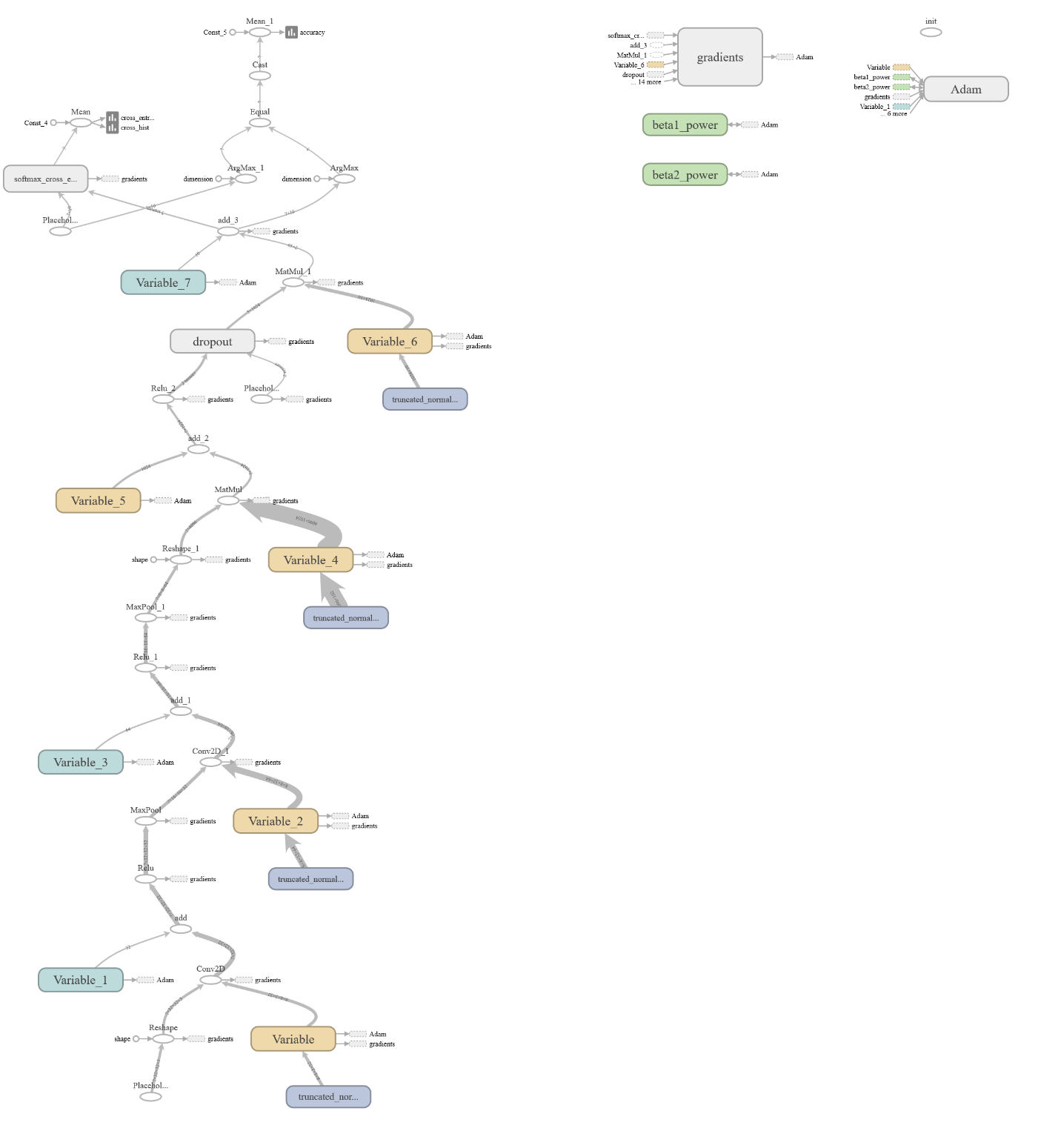
There are various designs of the RNN and CNN depends on which dataset we are choosing. According to test result the accuracy and loss of question 1 and question 2, there is a huge difference between both networks. The CNN works better, has a high accuracy and lower loss. In performance between them although both were used and CNN is better depends on the problem. When we were running both models, the RNN is a more 'natural' approach, given that text is naturally sequential. However, RNNs are quite slow and fickle to train.

4. Implement the image classification with CNN model, with a new dataset which is not used in the class (E.g. CIFAR 10 dataset)

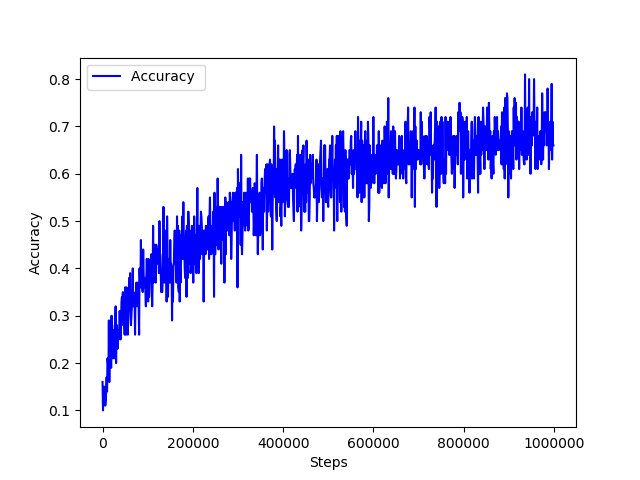
For our data set we chose CIFAR-10 a 60,000 32x32 color images divided equally in to ten categorizes. The data set is further divided into 50,000 train and 10,000 test images. The data set can be found at [CS Toronto](https://www.cs.toronto.edu/~kriz/cifar.html) website. Below is a small sample of the images in the data set.

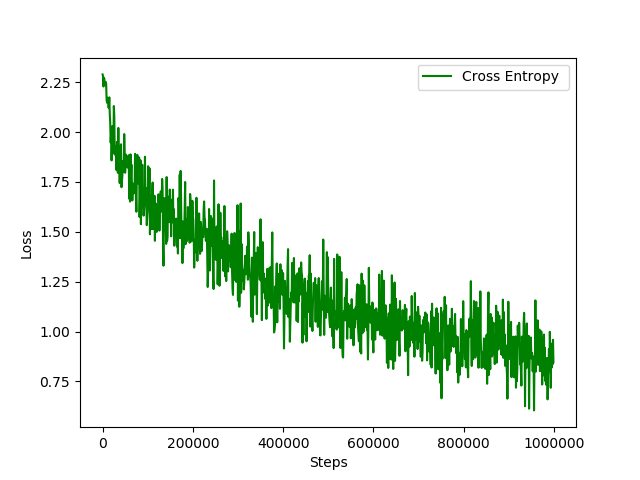


For the CNN, I used the code provided by our instructors with some minor modifications. In addition, I used some code to manage the CIFAR-10 data set. I got the code from “Learning TensorFlow A guide to build Deep Learning Systems” by Tom Hope, Yehezkel S. Resheff, and Itay Lider and can be found in the file cifar.py.  
I tried numerous changes to the hyper parameters. Those included three different optimizers, the learning rates, filter sizes, batch sizes, and the number of iterations. Below is the basic graph of the CNN.



The best I was able to get was 65.87% percent with the Adam optimizer, the learning rate set to .001, batch size 100, 10,000 iterations, and the filters set to 16,32,64. Below is are accuracy and loss graphs on the training data.





With the other optimizers, despite all of the changes I made to the hyper parameters, I never was able to get above 60%.

### Conclusion

With this CNN the best optimizer is Adam. In addition you will want to reduce the filter size and increase the batch size and the number of iterations. But to get a program that can be used in the real world you will need a better CNN.

To view a quick summary of all of the changes I made see Notes.txt. To view all of the data open the runs folder with Tensorboard.