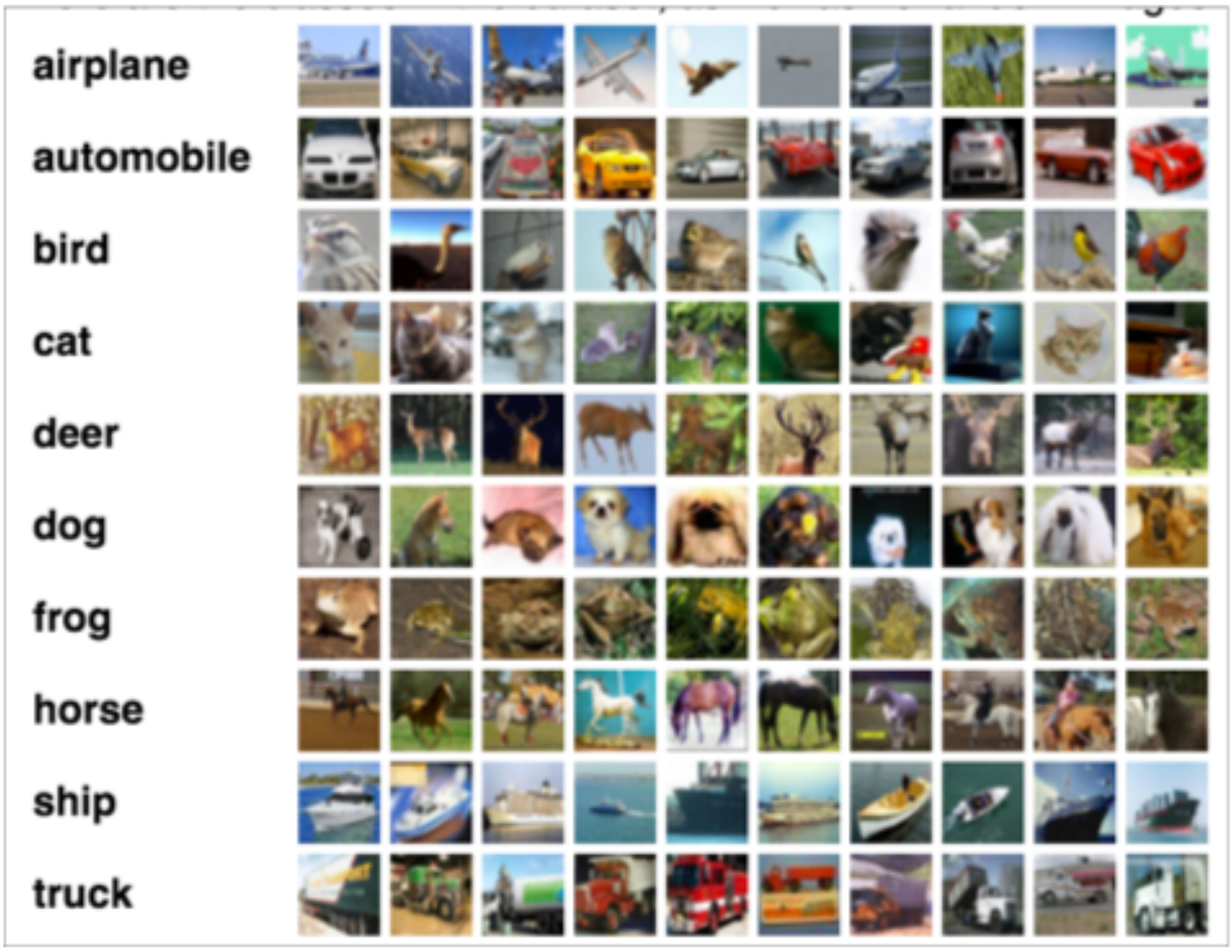
FINAL PROJECT REPORT
IMAGE RECOGNITION

Mehul Suresh Kumar

U52982215 Neural Networks and Deep Learning

Aim:

To design and implement a complex convolution neural network (CNN) to recognize images. The dataset used here is the CIFAR-10 dataset (http://www.cs.toronto.edu/~kriz/cifar.html). The objective of the network is to take RGB images and successfully recognize and classify them into 10 classes.

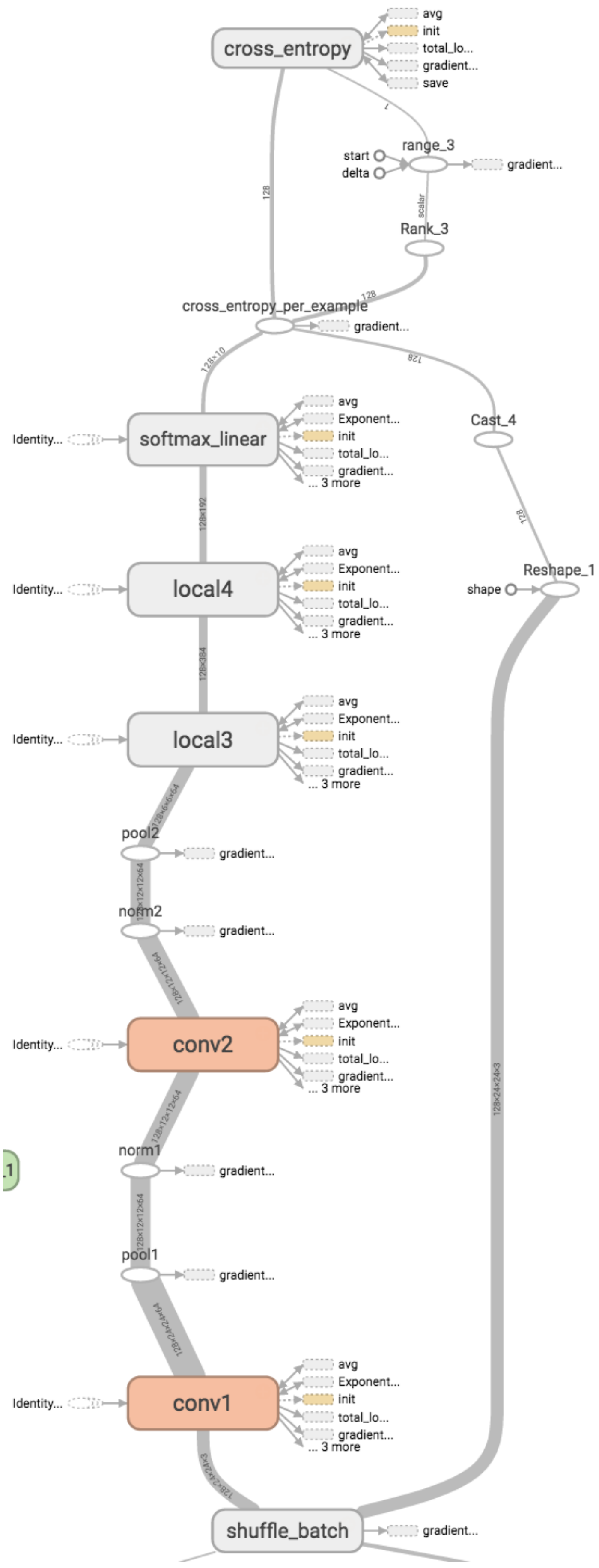


The CIFAR-10 DATASET

How to Run:

1. Install Tensor Flow by following instructions from https://www.tensorflow.org/versions/r0.7/get_started/os_setup.html
2. Enter the following
   source activate tensorflow
   Python cifar10_train.py
   python cifar10_eval.py
   tensorboard --logdir=/tmp/cifar10_train
3. Go to http://localhost:6006/ to view graphs and networks

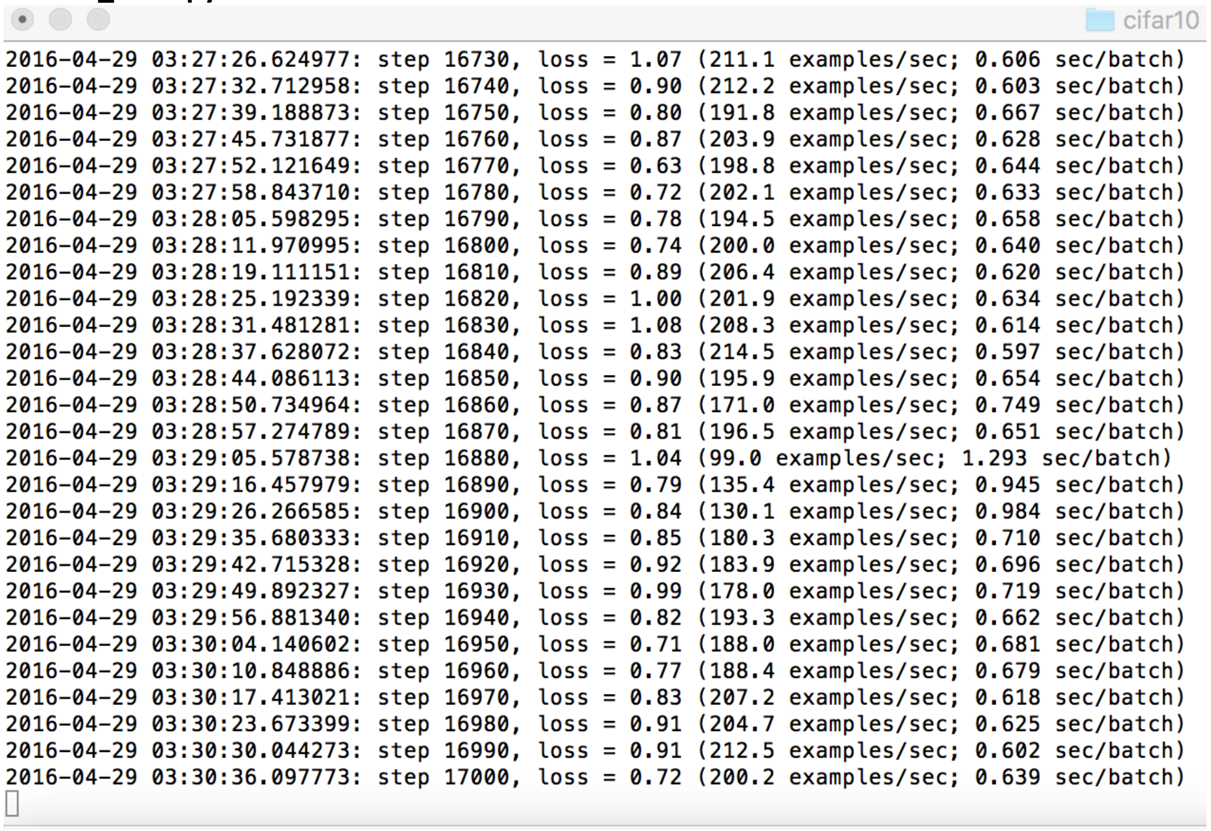
Network Architecture:


cross_entropy
avg
init
total_lo...
gradient...
save
range_3
start
delta
gradient...
Rank_3
scalar
128
cross_entropy_per_example
gradient...
128×10
softmax_linear
Identity...
Exponent...
... 3 more
Cast_4
128×192
local4
Reshape_1
shape
128×384
local3
128×6×6×64
pool2
norm2
128×12×12×64
conv2
norm1
pool1
128×24×24×64
conv1
128×24×24×3
shuffle_batch

- Convolution layer 1 (Kernel: 5x5x3, Stride: 1, Features: 64)
- Max Pooling layer 1 (Stride 2x2)
- Local Response Normalization
- Convolution layer 2 (Kernel: 5x5x64, Stride: 1, Features: 64)
- Max Pooling layer 2 (Stride 2x2)
- Local Response Normalization
- Fully connected Layer 1(Neurons : 384)
- Fully connected layer 2(Neurons : 192)
- Softmax Layer

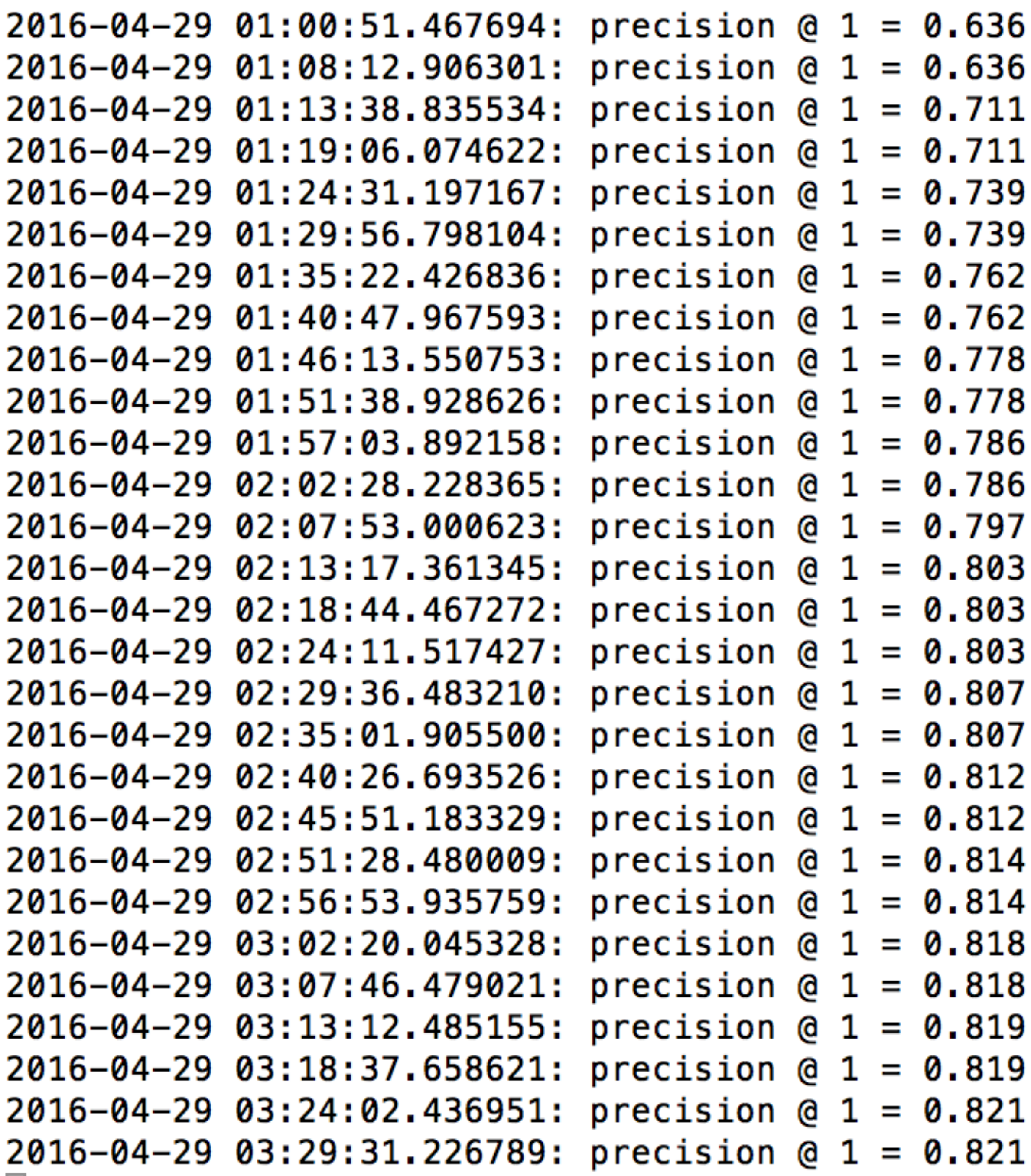
Sample Output:

Cifar10_train.py



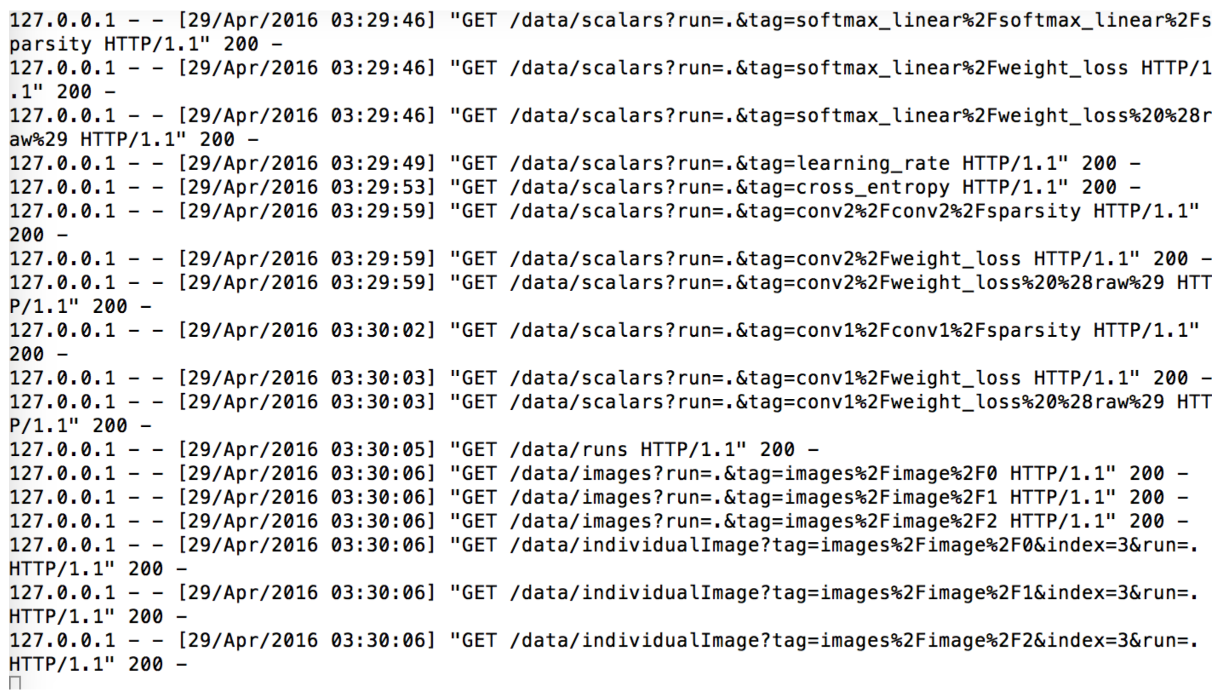
```
2016-04-29 03:27:26.624977: step 16730, loss = 1.07 (211.1 examples/sec; 0.606 sec/batch)
2016-04-29 03:27:32.712958: step 16740, loss = 0.90 (212.2 examples/sec; 0.603 sec/batch)
2016-04-29 03:27:39.188873: step 16750, loss = 0.80 (191.8 examples/sec; 0.667 sec/batch)
2016-04-29 03:27:45.731877: step 16760, loss = 0.87 (203.9 examples/sec; 0.628 sec/batch)
2016-04-29 03:27:52.121649: step 16770, loss = 0.63 (198.8 examples/sec; 0.644 sec/batch)
2016-04-29 03:27:58.843710: step 16780, loss = 0.72 (202.1 examples/sec; 0.633 sec/batch)
2016-04-29 03:28:05.598295: step 16790, loss = 0.78 (194.5 examples/sec; 0.658 sec/batch)
2016-04-29 03:28:11.970995: step 16800, loss = 0.74 (200.0 examples/sec; 0.640 sec/batch)
2016-04-29 03:28:19.111151: step 16810, loss = 0.89 (206.4 examples/sec; 0.620 sec/batch)
2016-04-29 03:28:25.192339: step 16820, loss = 1.00 (201.9 examples/sec; 0.634 sec/batch)
2016-04-29 03:28:31.481281: step 16830, loss = 1.08 (208.3 examples/sec; 0.614 sec/batch)
2016-04-29 03:28:37.628072: step 16840, loss = 0.83 (214.5 examples/sec; 0.597 sec/batch)
2016-04-29 03:28:44.086113: step 16850, loss = 0.90 (195.9 examples/sec; 0.654 sec/batch)
2016-04-29 03:28:50.734964: step 16860, loss = 0.87 (171.0 examples/sec; 0.749 sec/batch)
2016-04-29 03:28:57.274789: step 16870, loss = 0.81 (196.5 examples/sec; 0.651 sec/batch)
2016-04-29 03:29:05.578738: step 16880, loss = 1.04 (99.0 examples/sec; 1.293 sec/batch)
2016-04-29 03:29:16.457979: step 16890, loss = 0.79 (135.4 examples/sec; 0.945 sec/batch)
2016-04-29 03:29:26.266585: step 16900, loss = 0.84 (130.1 examples/sec; 0.984 sec/batch)
2016-04-29 03:29:35.680333: step 16910, loss = 0.85 (180.3 examples/sec; 0.710 sec/batch)
2016-04-29 03:29:42.715328: step 16920, loss = 0.92 (183.9 examples/sec; 0.696 sec/batch)
2016-04-29 03:29:49.892327: step 16930, loss = 0.99 (178.0 examples/sec; 0.719 sec/batch)
2016-04-29 03:29:56.881340: step 16940, loss = 0.82 (193.3 examples/sec; 0.662 sec/batch)
2016-04-29 03:30:04.140602: step 16950, loss = 0.71 (188.0 examples/sec; 0.681 sec/batch)
2016-04-29 03:30:10.848886: step 16960, loss = 0.77 (188.4 examples/sec; 0.679 sec/batch)
2016-04-29 03:30:17.413021: step 16970, loss = 0.83 (207.2 examples/sec; 0.618 sec/batch)
2016-04-29 03:30:23.673399: step 16980, loss = 0.91 (204.7 examples/sec; 0.625 sec/batch)
2016-04-29 03:30:30.044273: step 16990, loss = 0.91 (212.5 examples/sec; 0.602 sec/batch)
2016-04-29 03:30:36.097773: step 17000, loss = 0.72 (200.2 examples/sec; 0.639 sec/batch)
```

Cifar10_eval.py



```
2016-04-29 01:00:51.467694: precision @ 1 = 0.636
2016-04-29 01:08:12.906301: precision @ 1 = 0.636
2016-04-29 01:13:38.835534: precision @ 1 = 0.711
2016-04-29 01:19:06.074622: precision @ 1 = 0.711
2016-04-29 01:24:31.197167: precision @ 1 = 0.739
2016-04-29 01:29:56.798104: precision @ 1 = 0.739
2016-04-29 01:35:22.426836: precision @ 1 = 0.762
2016-04-29 01:40:47.967593: precision @ 1 = 0.762
2016-04-29 01:46:13.550753: precision @ 1 = 0.778
2016-04-29 01:51:38.928626: precision @ 1 = 0.778
2016-04-29 01:57:03.892158: precision @ 1 = 0.786
2016-04-29 02:02:28.228365: precision @ 1 = 0.786
2016-04-29 02:07:53.000623: precision @ 1 = 0.797
2016-04-29 02:13:17.361345: precision @ 1 = 0.803
2016-04-29 02:18:44.467272: precision @ 1 = 0.803
2016-04-29 02:24:11.517427: precision @ 1 = 0.803
2016-04-29 02:29:36.483210: precision @ 1 = 0.807
2016-04-29 02:35:01.905500: precision @ 1 = 0.807
2016-04-29 02:40:26.693526: precision @ 1 = 0.812
2016-04-29 02:45:51.183329: precision @ 1 = 0.812
2016-04-29 02:51:28.480009: precision @ 1 = 0.814
2016-04-29 02:56:53.935759: precision @ 1 = 0.814
2016-04-29 03:02:20.045328: precision @ 1 = 0.818
2016-04-29 03:07:46.479021: precision @ 1 = 0.818
2016-04-29 03:13:12.485155: precision @ 1 = 0.819
2016-04-29 03:18:37.658621: precision @ 1 = 0.819
2016-04-29 03:24:02.436951: precision @ 1 = 0.821
2016-04-29 03:29:31.226789: precision @ 1 = 0.821
```

Tensorboard:



```
127.0.0.1 - - [29/Apr/2016 03:29:46] "GET /data/scalars?run=.&tag=softmax_linear%2Fsoftmax_linear%2Fs
parsity HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:29:46] "GET /data/scalars?run=.&tag=softmax_linear%2Fweight_loss HTTP/1
.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:29:46] "GET /data/scalars?run=.&tag=softmax_linear%2Fweight_loss%20%28r
aw%29 HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:29:49] "GET /data/scalars?run=.&tag=learning_rate HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:29:53] "GET /data/scalars?run=.&tag=cross_entropy HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:29:59] "GET /data/scalars?run=.&tag=conv2%2Fconv2%2Fsparsity HTTP/1.1"
200 -
127.0.0.1 - - [29/Apr/2016 03:29:59] "GET /data/scalars?run=.&tag=conv2%2Fweight_loss HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:29:59] "GET /data/scalars?run=.&tag=conv2%2Fweight_loss%20%28raw%29 HTT
P/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:02] "GET /data/scalars?run=.&tag=conv1%2Fconv1%2Fsparsity HTTP/1.1"
200 -
127.0.0.1 - - [29/Apr/2016 03:30:03] "GET /data/scalars?run=.&tag=conv1%2Fweight_loss HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:03] "GET /data/scalars?run=.&tag=conv1%2Fweight_loss%20%28raw%29 HTT
P/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:05] "GET /data/runs HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:06] "GET /data/images?run=.&tag=images%2Fimage%2F0 HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:06] "GET /data/images?run=.&tag=images%2Fimage%2F1 HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:06] "GET /data/images?run=.&tag=images%2Fimage%2F2 HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:06] "GET /data/individualImage?tag=images%2Fimage%2F0&index=3&run=.
HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:06] "GET /data/individualImage?tag=images%2Fimage%2F1&index=3&run=.
HTTP/1.1" 200 -
127.0.0.1 - - [29/Apr/2016 03:30:06] "GET /data/individualImage?tag=images%2Fimage%2F2&index=3&run=.
HTTP/1.1" 200 -
```

Accuracy:

To test the accuracy we run cifar10_eval.py
Parameters to train : 1,068,298
Reached 82.1% accuracy after running for 4 hours on an Intel i7 CPU
Reaches 86% accuracy after running overnight

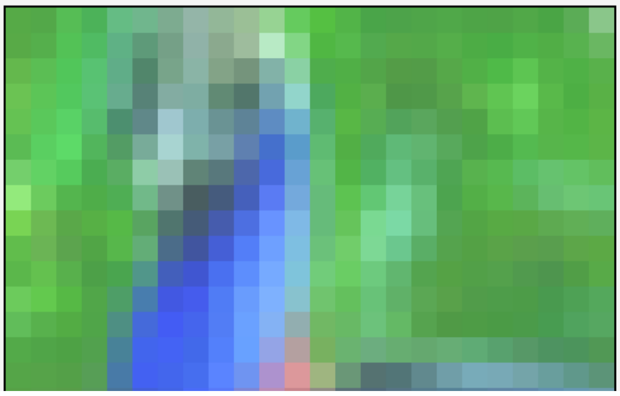
Code Info:
Cifar10.py – Builds the entire network model
Cifar10_train.py – trains the model
Cifar10_eval.py – Evaluates the performance of the trained network

Input:
Images are cropped to 24x24
Distort image brightness and contrast
Randomly flip images

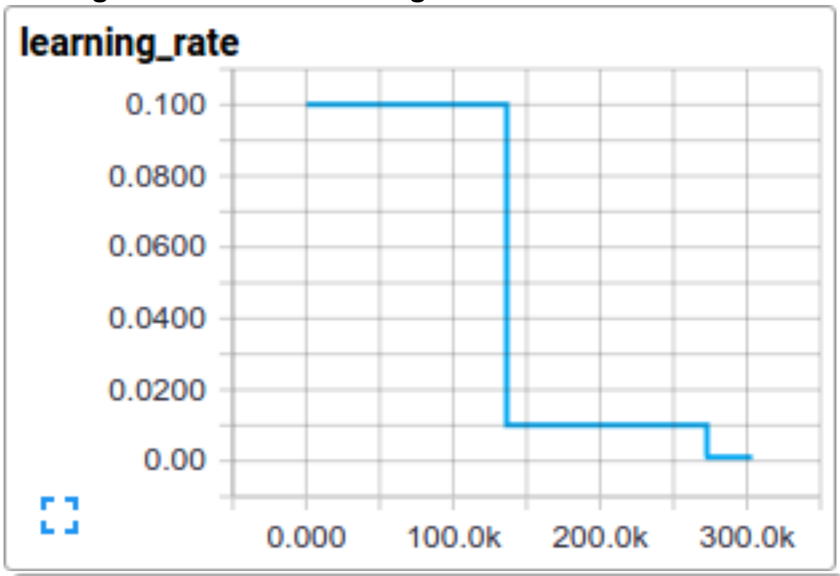
images/image/0

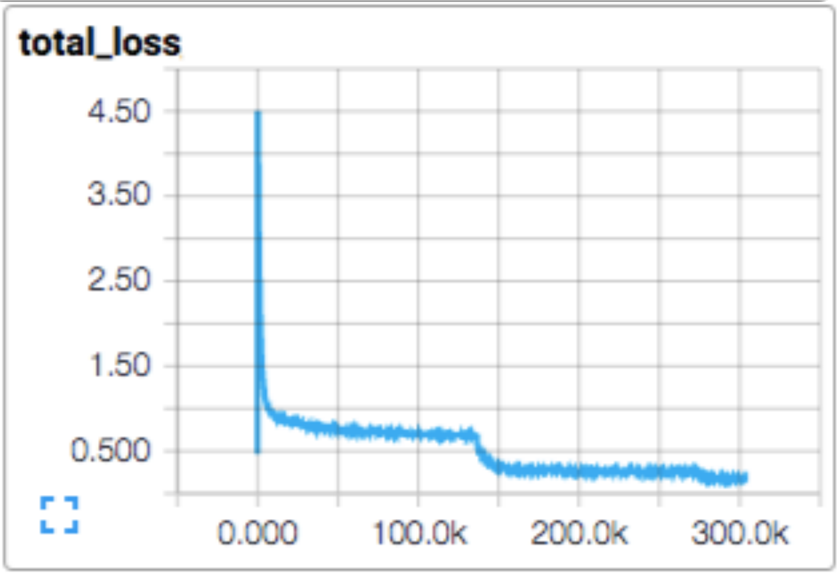
images/image/1

images/image/2



Plotting Total Loss and Learning Rate:





Conclusion:

Thus a Convolution Neural network to recognize images was created and trained. It achieves remarkable results when compared to the industry standard. The network architecture and the outputs are logged in this report.