Course: CS5590 Python Programming
Lab Assignment - #4

Report

Team id: #16

Name: Revanth Chakilam | ClassID: 02

Name: Swati Singh | ClassID: 33

Github: https://github.com/revanthchakilam/DeepLearning/wiki/Deep-Learning-Lab-2

Youtube: https://youtu.be/MkHp03hxZa4

# Introduction

This Wiki/Report contains the Deep Learning lab assignment (Lab 4) which has Neural network algorithms.

# Objectives

The objective of this lab assignment is to apply CNN on text classfication and image classfication followed by RNN with text classification and comparing CNN vs RNN for text classfication

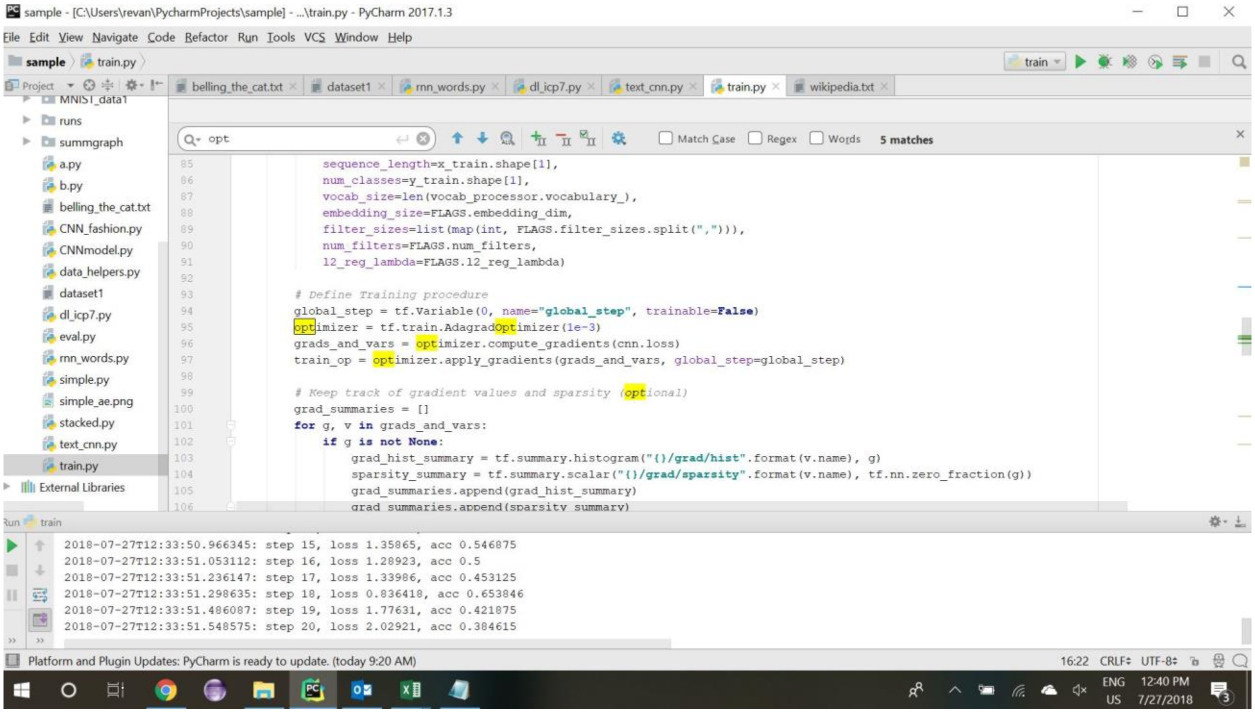
# Task 1: Text Classification with CNN

## Approach/Method:

I have used the wikipedia dataset which is not used in the class. I have calculated the accuracy and loss using all the three different optimisers and tried to analyse which optimiser suits well for the model dataset.

I have found best results with Wikipedia dataset (plain text dataset) with Adam optimiser gives the most accuracy and minute loss. And using RMS optimiser and reducing the number of epochs we get the best results for accuracy as well. So in conclusion using RMS optimiser with number of epochs gives the best result. And same hyper parameters with adam optimiser also gives better results.

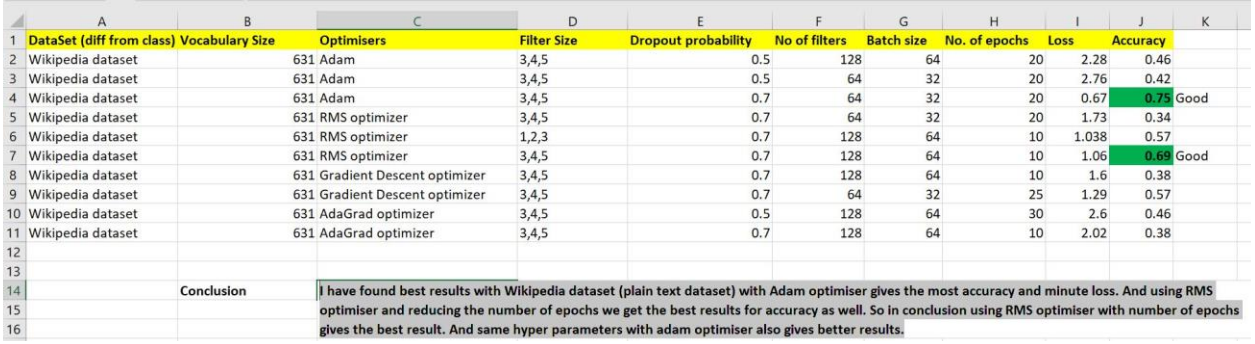
### code



```
            sequence_length=x_train.shape[1],
            num_classes=y_train.shape[1],
            vocab_size=len(vocab_processor.vocabulary_),
            embedding_size=FLAGS.embedding_dim,
            filter_sizes=list(map(int, FLAGS.filter_sizes.split(","))),
            num_filters=FLAGS.num_filters,
            l2_reg_lambda=FLAGS.l2_reg_lambda)

        # Define Training procedure
        global_step = tf.Variable(0, name="global_step", trainable=False)
        optimizer = tf.train.AdagradOptimizer(1e-3)
        grads_and_vars = optimizer.compute_gradients(cnn.loss)
        train_op = optimizer.apply_gradients(grads_and_vars, global_step=global_step)

        # Keep track of gradient values and sparsity (optional)
        grad_summaries = []
        for g, v in grads_and_vars:
            if g is not None:
                grad_hist_summary = tf.summary.histogram("{}/grad/hist".format(v.name), g)
                sparsity_summary = tf.summary.scalar("{}/grad/sparsity".format(v.name), tf.nn.zero_fraction(g))
                grad_summaries.append(grad_hist_summary)
                grad_summaries.append(sparsity_summary)
```

```
2018-07-27T12:33:50.966345: step 15, loss 1.35865, acc 0.546875
2018-07-27T12:33:51.053112: step 16, loss 1.28923, acc 0.5
2018-07-27T12:33:51.236147: step 17, loss 1.33986, acc 0.453125
2018-07-27T12:33:51.298635: step 18, loss 0.836418, acc 0.653846
2018-07-27T12:33:51.486087: step 19, loss 1.77631, acc 0.421875
2018-07-27T12:33:51.548575: step 20, loss 2.02921, acc 0.384615
```

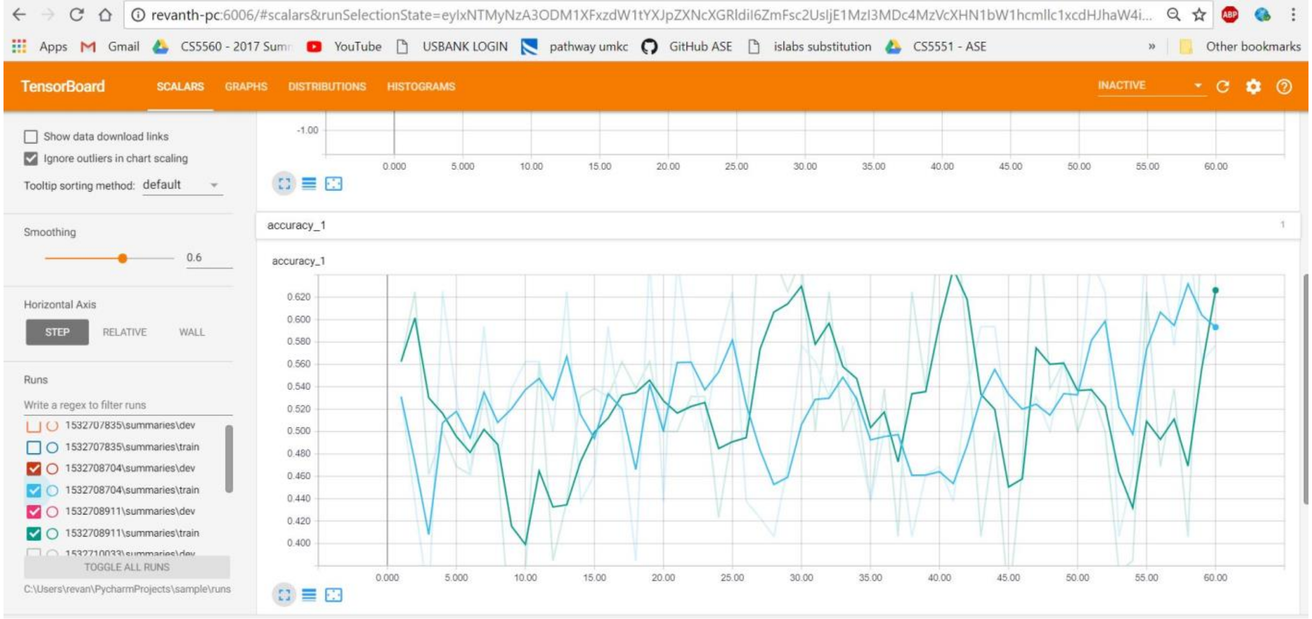
### Results



	A	B	C	D	E	F	G	H	I	J	K
1	DataSet (diff from class)	Vocabulary Size	Optimisers	Filter Size	Dropout probability	No of filters	Batch size	No. of epochs	Loss	Accuracy	
2	Wikipedia dataset	631	Adam	3,4,5	0.5	128	64	20	2.28	0.46	
3	Wikipedia dataset	631	Adam	3,4,5	0.5	64	32	20	2.76	0.42	
4	Wikipedia dataset	631	Adam	3,4,5	0.7	64	32	20	0.67	0.75	Good
5	Wikipedia dataset	631	RMS optimizer	3,4,5	0.7	64	32	20	1.73	0.34	
6	Wikipedia dataset	631	RMS optimizer	1,2,3	0.7	128	64	10	1.038	0.57	
7	Wikipedia dataset	631	RMS optimizer	3,4,5	0.7	128	64	10	1.06	0.69	Good
8	Wikipedia dataset	631	Gradient Descent optimizer	3,4,5	0.7	128	64	10	1.6	0.38	
9	Wikipedia dataset	631	Gradient Descent optimizer	3,4,5	0.7	64	32	25	1.29	0.57	
10	Wikipedia dataset	631	AdaGrad optimizer	3,4,5	0.5	128	64	30	2.6	0.46	
11	Wikipedia dataset	631	AdaGrad optimizer	3,4,5	0.7	128	64	10	2.02	0.38	
12											
13											
14		Conclusion	I have found best results with Wikipedia dataset (plain text dataset) with Adam optimiser gives the most accuracy and minute loss. And using RMS optimiser and reducing the number of epochs we get the best results for accuracy as well. So in conclusion using RMS optimiser with number of epochs gives the best result. And same hyper parameters with adam optimiser also gives better results.								
15											
16											

### Tensor Board Graphs

### Adam Optimiser



revanth-pc:6006/#scalars&runSelectionState=eylxNTMyNzA3ODM1XFxzdW1tYXJpZXNcXGRldiI6ZmFsc2UsIjE1MzI3MDc4MzVcXHN1bW1hcmllc1xcdHJhaW4i...

TensorBoard

☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

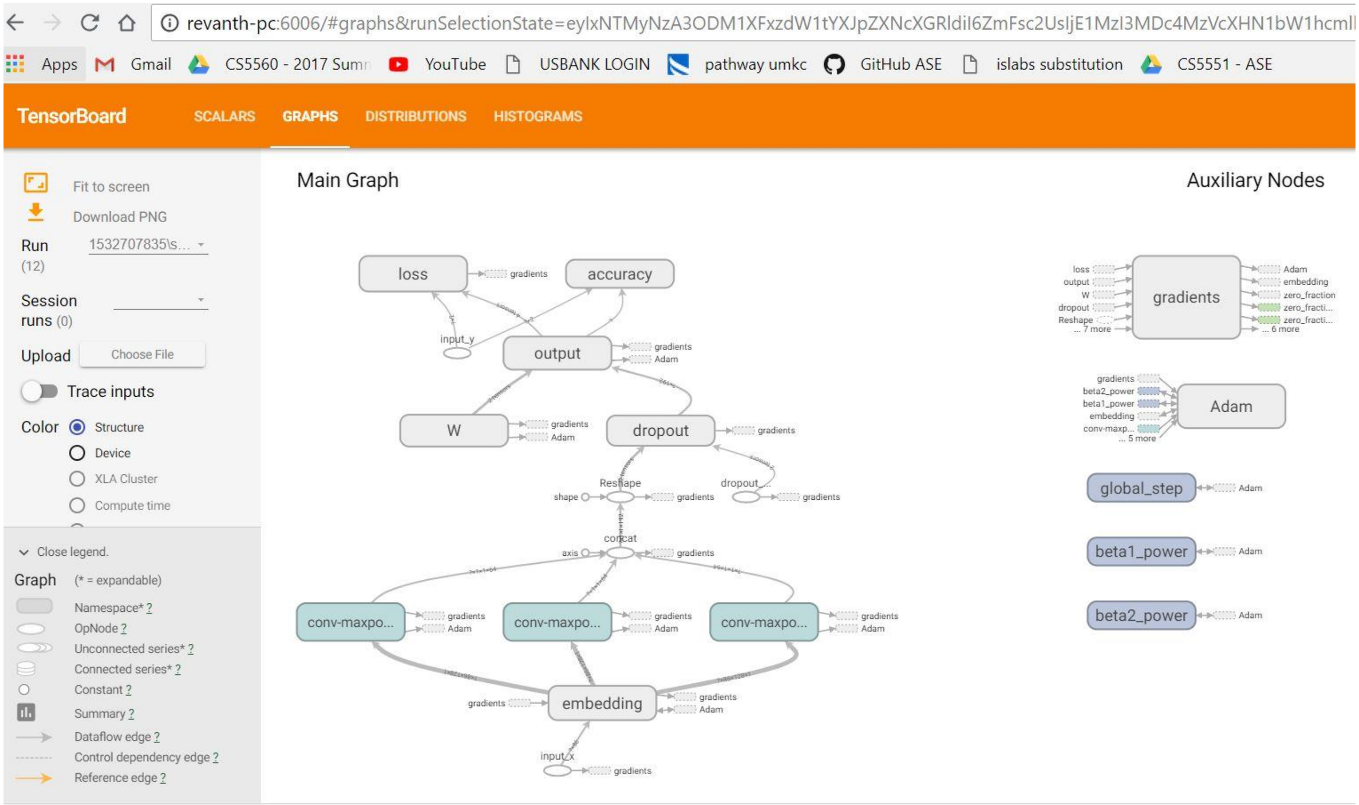
Horizontal Axis: STEP RELATIVE WALL

Runs

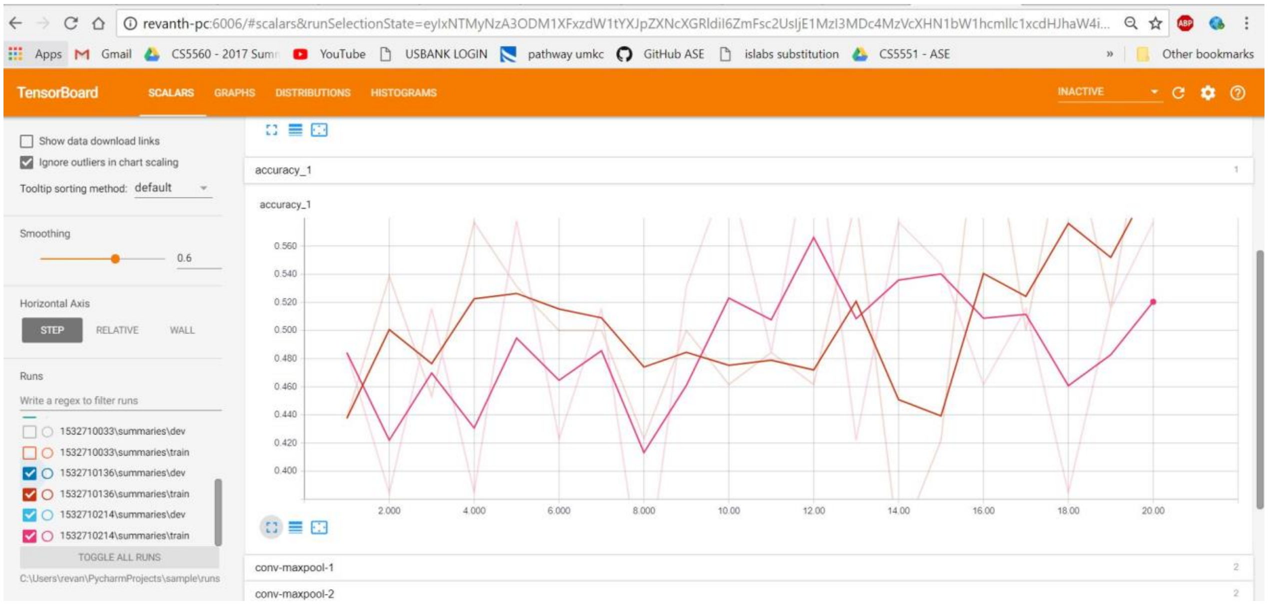
☐ ○ 1532707835\summaries\dev
☐ ○ 1532707835\summaries\train
☑ ○ 1532708704\summaries\dev
☑ ○ 1532708704\summaries\train
☑ ○ 1532708911\summaries\dev
☑ ○ 1532708911\summaries\train
☐ ○ 1532710033\summaries\dev

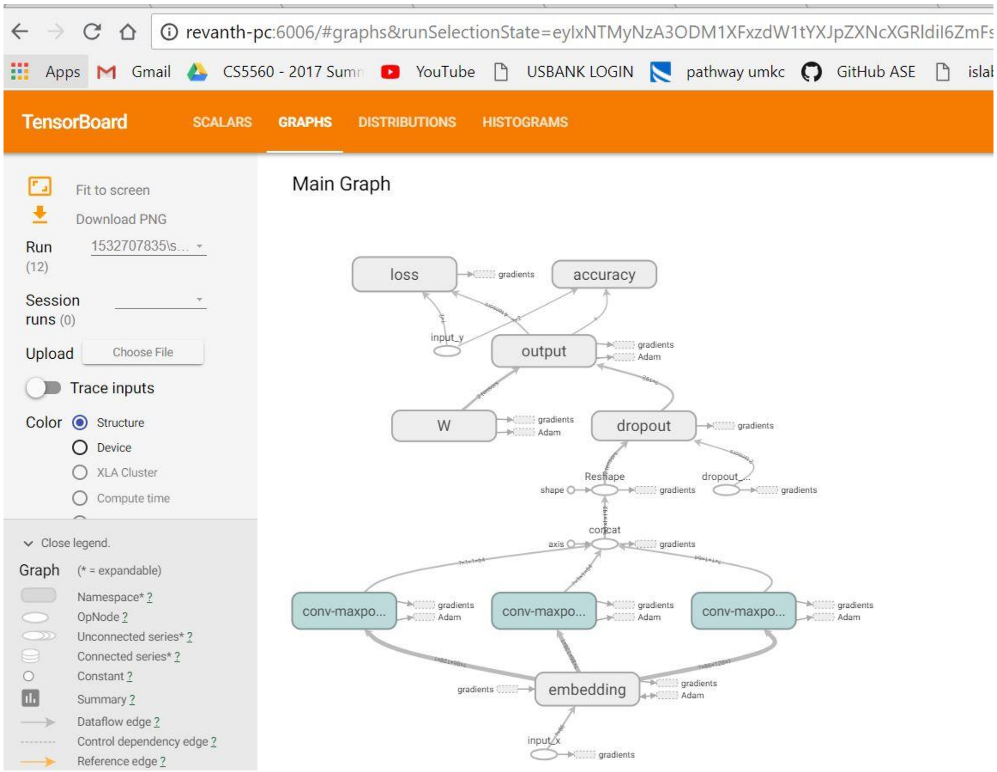
C:\Users\revan\PycharmProjects\sample\runs

accuracy_1



### RMS Optimiser


revanth-pc:6006/#scalars&runSelectionState=eyIxNTMyNzA3ODM1XFxzdW1tYXJpZXNcXGRldiI6ZmFsc2UsIjE1MzI3MDc4MzVcXHN1bW1hcmllc1xcdHJhaW4i...
Apps
Gmail
CS5560 - 2017 Summ
YouTube
USBANK LOGIN
pathway umkc
GitHub ASE
islabs substitution
CS5551 - ASE
Other bookmarks
TensorBoard
SCALARS
GRAPHS
DISTRIBUTIONS
HISTOGRAMS
INACTIVE
□ Show data download links
☑ Ignore outliers in chart scaling
Tooltip sorting method: default
Smoothing
0.6
Horizontal Axis
STEP
RELATIVE
WALL
Runs
Write a regex to filter runs
□ 1532710033\summaries\dev
□ 1532710033\summaries\train
☑ 1532710136\summaries\dev
☑ 1532710136\summaries\train
☑ 1532710214\summaries\dev
☑ 1532710214\summaries\train
TOGGLE ALL RUNS
C:\Users\revan\PycharmProjects\sample\runs
accuracy_1
1
accuracy_1
0.560
0.540
0.520
0.500
0.480
0.460
0.440
0.420
0.400
2.000
4.000
6.000
8.000
10.00
12.00
14.00
16.00
18.00
20.00
conv-maxpool-1
2
conv-maxpool-2
2



revanth-pc:6006/#graphs&runSelectionState=eylxNTMyNzA3ODM1XFxzdW1tYXJpZXNcXGRldiI6ZmFs

Run (12) 1532707835\s...

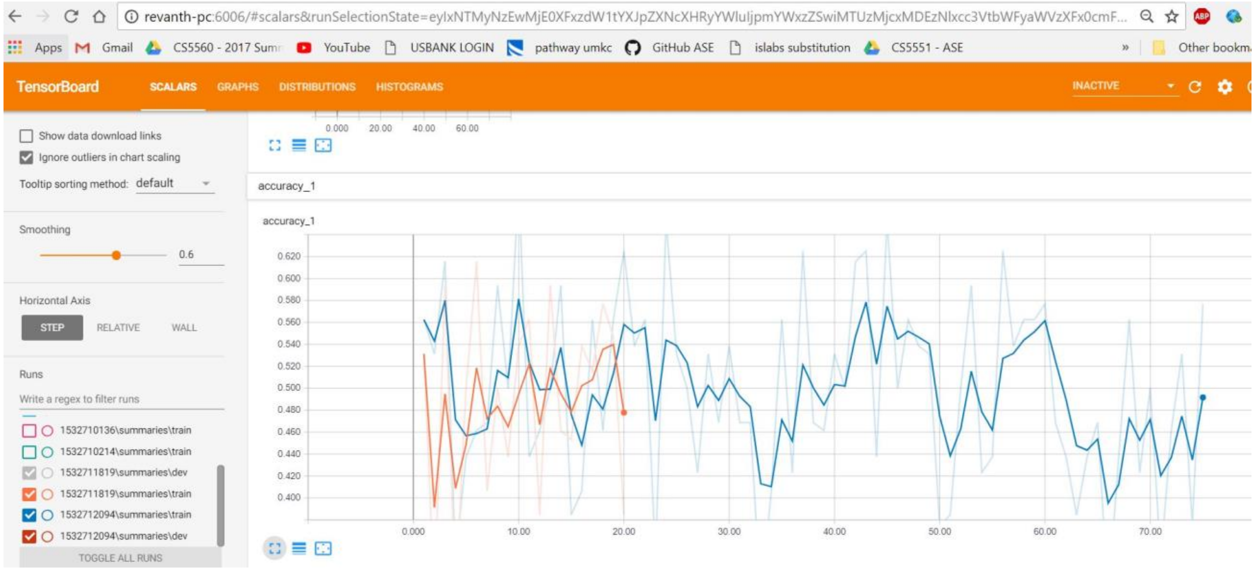
Session runs (0)

Upload Choose File

Trace inputs

Color ● Structure ○ Device ○ XLA Cluster ○ Compute time

### Gradient Descent Optimiser



revanth-pc:6006/#scalars&runSelectionState=eylxNTMyNzEwMjE0XFxzdW1tYXJpZXNcXHRyYWluIjpmYWxzZSwiMTUzMjcxMDEzNlxcc3VtbWFyaWVzXFx0cmF...

□ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

□ ○ 1532710136\summaries\train
□ ○ 1532710214\summaries\train
☑ ○ 1532711819\summaries\dev
☑ ○ 1532711819\summaries\train
☑ ○ 1532712094\summaries\train
☑ ○ 1532712094\summaries\dev

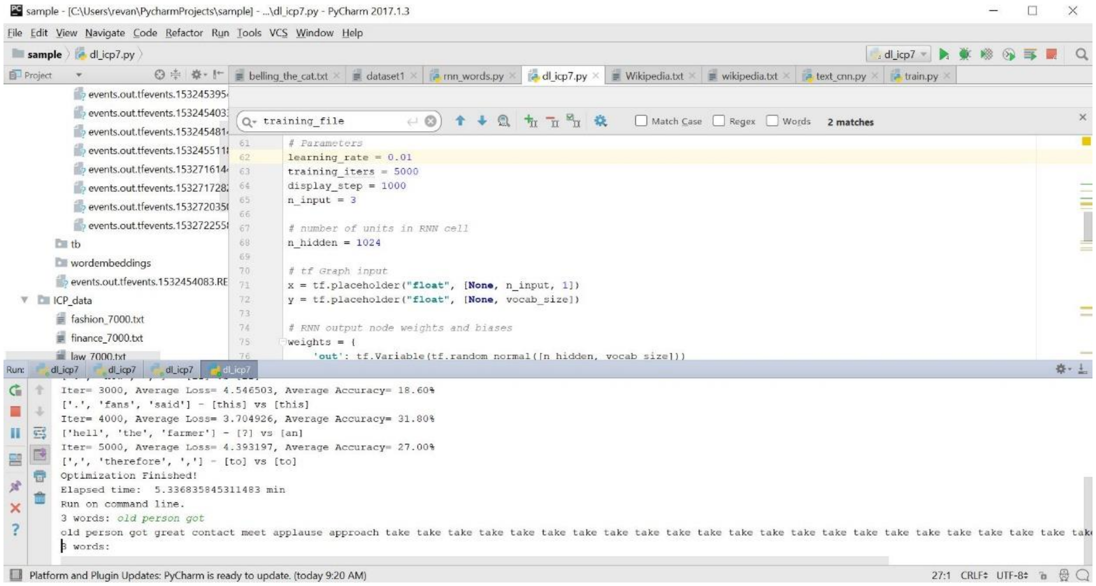
# Task 2: Text Classification with RNN LSTM

## Approach/Method:

I have used the same wikipedia dataset which is not used in the class. I have calculated the accuracy and loss using all the RMS optimiser with different learning rates and iteration steps and tried to analyse which combination is suited for the classification using RNN LTSM

I think that basic lstm worked well with the 0.01 learning rate and more number of iterations. By increasing the learning rate I wast unable to catch the accuracy and with multi rnn using 2 stack unlike the basic also gave good accuracy with the 0.01 learning rate

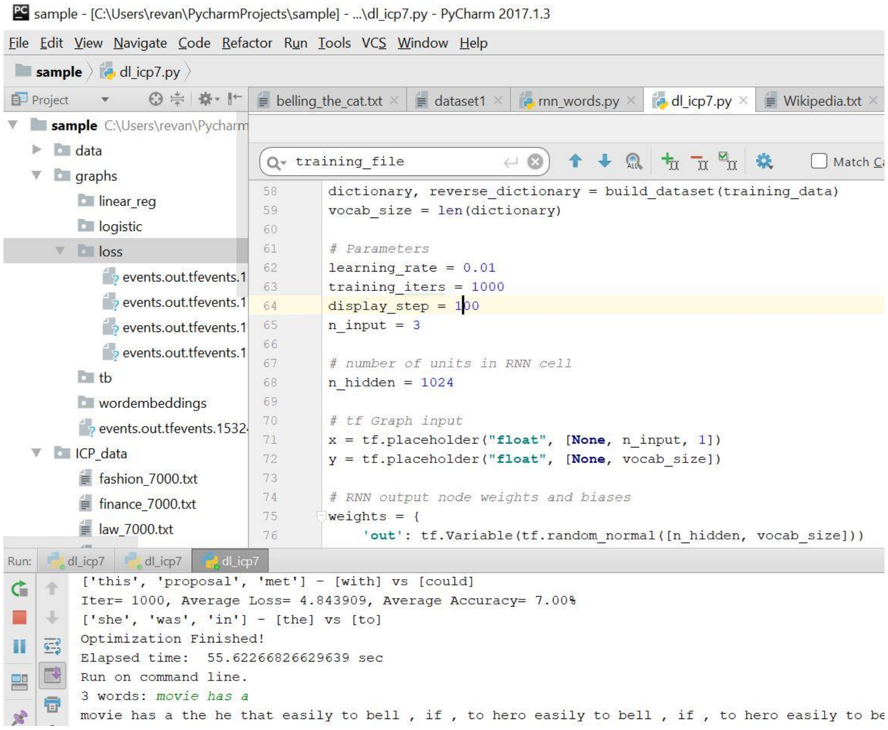
### code



sample - [C:\Users\revan\PycharmProjects\sample] - ...\dl_icp7.py - PyCharm 2017.1.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
sample
dl_icp7.py
dl_icp7
Project
belling_the_cat.txt
dataset1
rnn_words.py
dl_icp7.py
Wikipedia.txt
wikipedia.txt
text_cnn.py
train.py
events.out.tfevents.153245395
events.out.tfevents.153245403
events.out.tfevents.153245481
events.out.tfevents.153245511
events.out.tfevents.153271614
events.out.tfevents.153271728
events.out.tfevents.153272035
events.out.tfevents.153272255
tb
wordembeddings
events.out.tfevents.1532454083.RE
ICP_data
fashion_7000.txt
finance_7000.txt
law_7000.txt
training_file
☐ Match Case
☐ Regex
☐ Words
2 matches
Run:
dl_icp7
dl_icp7
dl_icp7
dl_icp7
Platform and Plugin Updates: PyCharm is ready to update. (today 9:20 AM)
27:1 CRLF UTF-8

```
# Parameters
learning_rate = 0.01
training_iters = 5000
display_step = 1000
n_input = 3

# number of units in RNN cell
n_hidden = 1024

# tf Graph input
x = tf.placeholder("float", [None, n_input, 1])
y = tf.placeholder("float", [None, vocab_size])

# RNN output node weights and biases
weights = {
    'out': tf.Variable(tf.random_normal([n_hidden, vocab_size]))
```

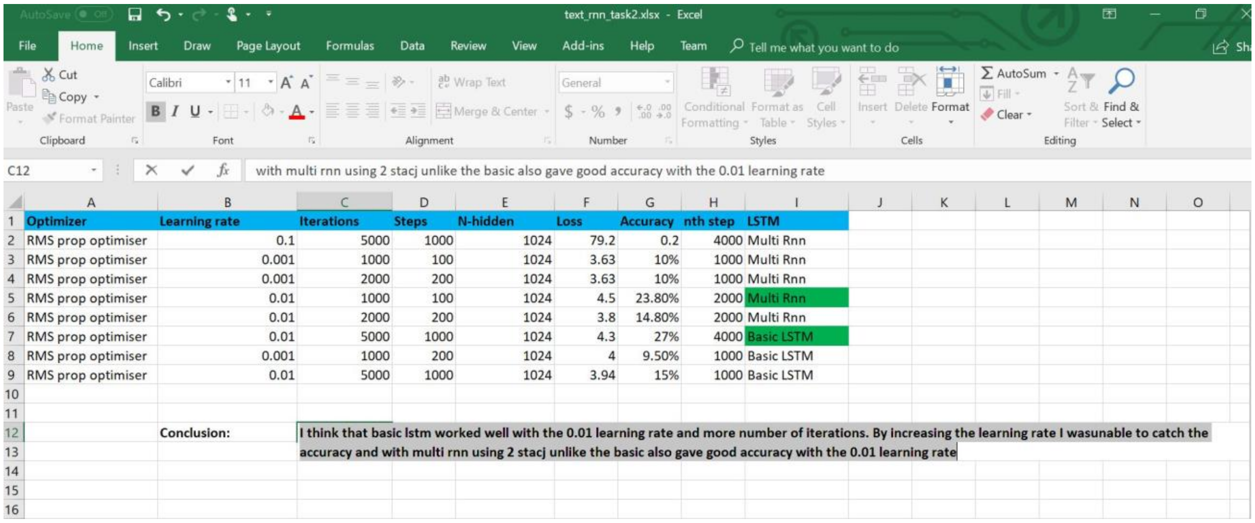
```
Iter= 3000, Average Loss= 4.546503, Average Accuracy= 18.60%
['.', 'fans', 'said'] - [this] vs [this]
Iter= 4000, Average Loss= 3.704926, Average Accuracy= 31.80%
['hell', 'the', 'farmer'] - [?] vs [an]
Iter= 5000, Average Loss= 4.393197, Average Accuracy= 27.00%
[',', 'therefore', ','] - [to] vs [to]
Optimization Finished!
Elapsed time:  5.336835845311483 min
Run on command line.
3 words: old person got
old person got great contact meet applause approach take take take take take take take take take take take take take take take take take take take take take take tak
3 words:
```



```
dictionary, reverse_dictionary = build_dataset(training_data)
vocab_size = len(dictionary)

# Parameters
learning_rate = 0.01
training_iters = 1000
display_step = 100
n_input = 3

# number of units in RNN cell
n_hidden = 1024

# tf Graph input
x = tf.placeholder("float", [None, n_input, 1])
y = tf.placeholder("float", [None, vocab_size])

# RNN output node weights and biases
weights = {
    'out': tf.Variable(tf.random_normal([n_hidden, vocab_size]))
```

```
['this', 'proposal', 'met'] - [with] vs [could]
Iter= 1000, Average Loss= 4.843909, Average Accuracy= 7.00%
['she', 'was', 'in'] - [the] vs [to]
Optimization Finished!
Elapsed time:  55.62266826629639 sec
Run on command line.
3 words: movie has a
movie has a the he that easily to bell , if , to hero easily to bell , if , to hero easily to be
```

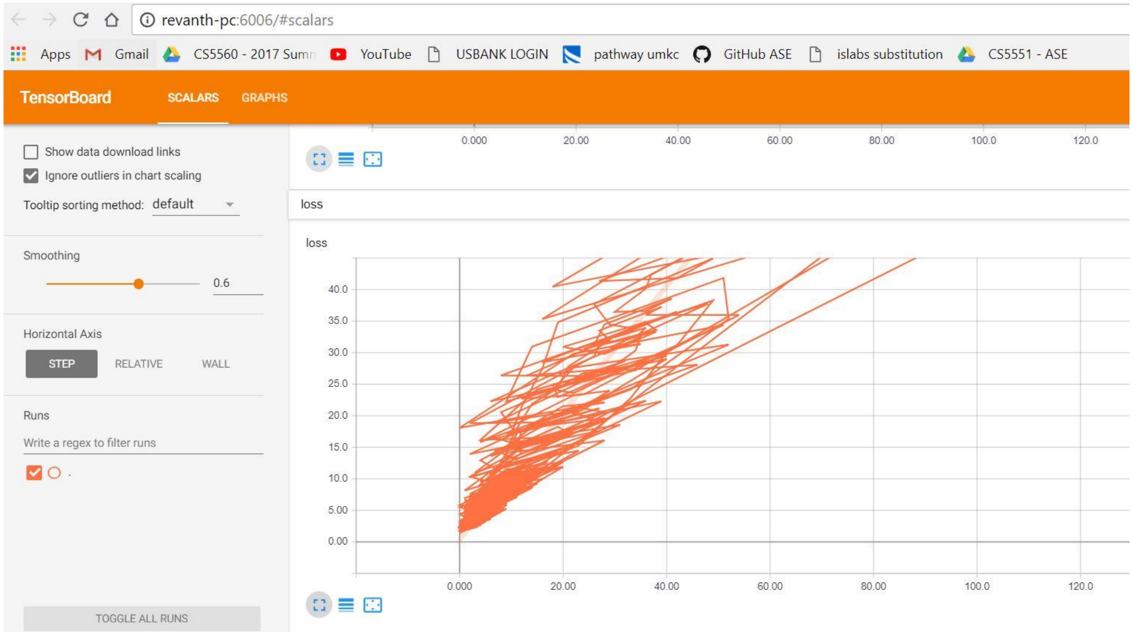
### Results



with multi rnn using 2 stacj unlike the basic also gave good accuracy with the 0.01 learning rate

	A	B	C	D	E	F	G	H	I
1	Optimizer	Learning rate	Iterations	Steps	N-hidden	Loss	Accuracy	nth step	LSTM
2	RMS prop optimiser	0.1	5000	1000	1024	79.2	0.2	4000	Multi Rnn
3	RMS prop optimiser	0.001	1000	100	1024	3.63	10%	1000	Multi Rnn
4	RMS prop optimiser	0.001	2000	200	1024	3.63	10%	1000	Multi Rnn
5	RMS prop optimiser	0.01	1000	100	1024	4.5	23.80%	2000	Multi Rnn
6	RMS prop optimiser	0.01	2000	200	1024	3.8	14.80%	2000	Multi Rnn
7	RMS prop optimiser	0.01	5000	1000	1024	4.3	27%	4000	Basic LSTM
8	RMS prop optimiser	0.001	1000	200	1024	4	9.50%	1000	Basic LSTM
9	RMS prop optimiser	0.01	5000	1000	1024	3.94	15%	1000	Basic LSTM
10									
11									
12		Conclusion:	I think that basic lstm worked well with the 0.01 learning rate and more number of iterations. By increasing the learning rate I wasunable to catch the accuracy and with multi rnn using 2 stacj unlike the basic also gave good accuracy with the 0.01 learning rate						
13									
14									
15									
16									

### Tensor Board Graphs


revanth-pc:6006/#scalars

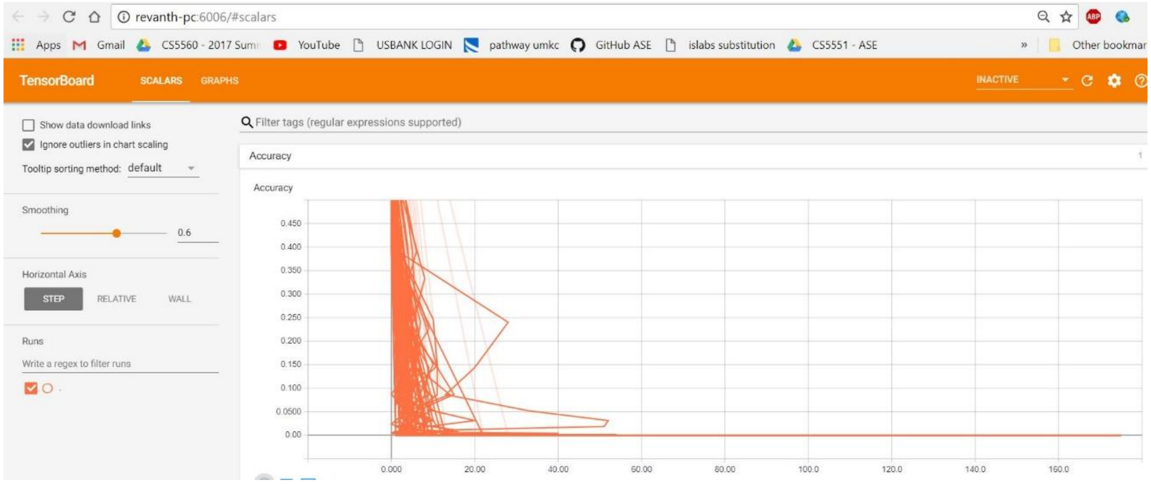
☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP


revanth-pc:6006/#scalars

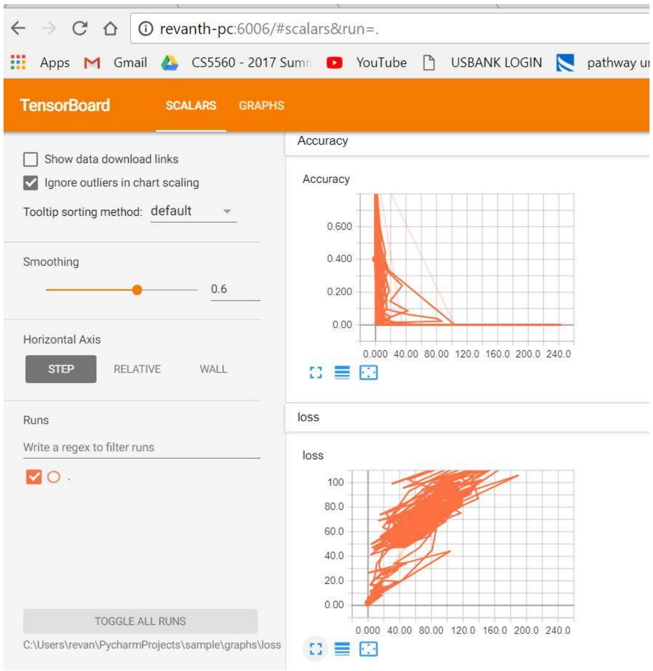
☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP



revanth-pc:6006/#scalars&run=.

☐ Show data download links

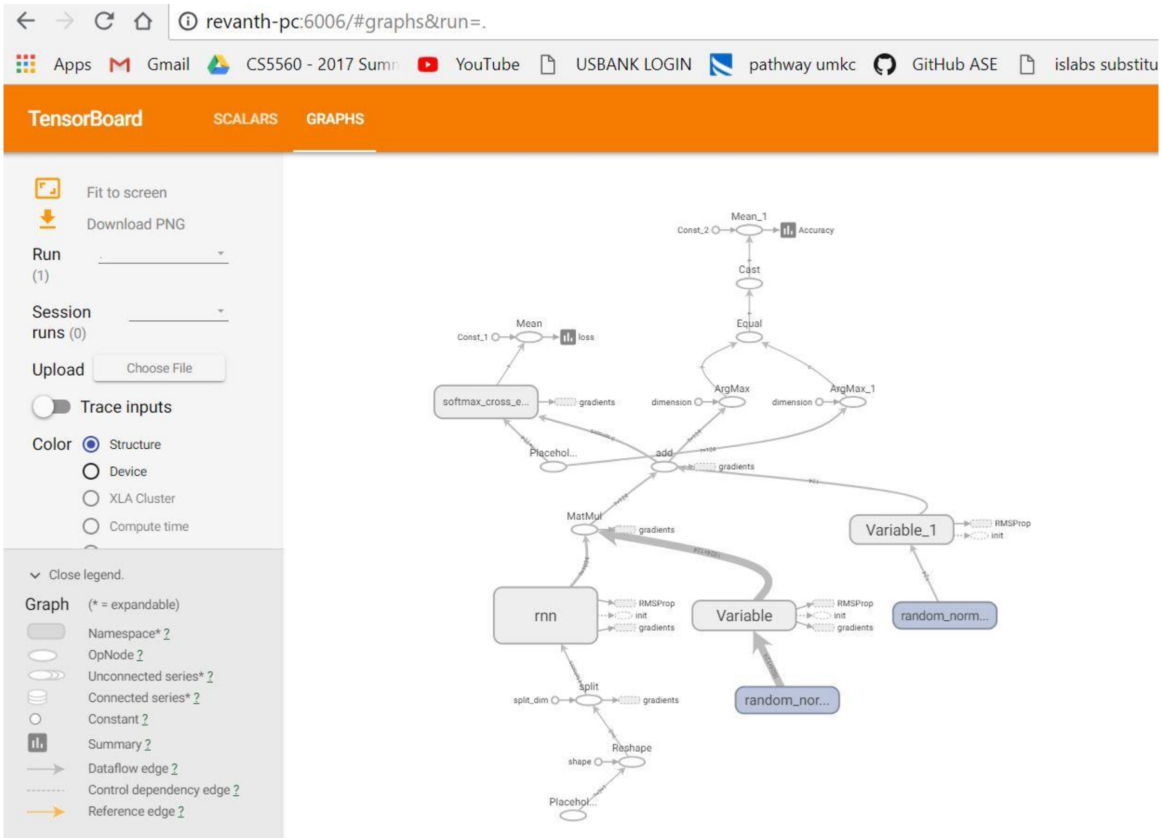
☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP

C:\Users\revan\PycharmProjects\sample\graphs\loss



revanth-pc:6006/#graphs&run=.

Run (1) .

Session runs (0)

Trace inputs

Color: ● Structure ○ Device ○ XLA Cluster ○ Compute time

# Task 3: Text Classification of CNN vs RNN LSTM

### CNN

I have found best results with Wikipedia dataset (plain text dataset) with Adam optimiser gives the most accuracy and minute loss. And using RMS optimiser and reducing the number of epochs we get the best results for accuracy as well. So in conclusion using RMS optimiser with number of epochs gives the best result. And same hyper parameters with adam optimiser also gives better results.

### RNN LSTM

I think that basic lstm worked well with the 0.01 learning rate and more number of iterations. By increasing the learning rate I wast unable to catch the accuracy and with multi rnn using 2 stack unlike the basic also gave good accuracy with the 0.01 learning rate

### Conclusion

Based on the dataset used,I will chose the CNN with Adam/RMS optimiser

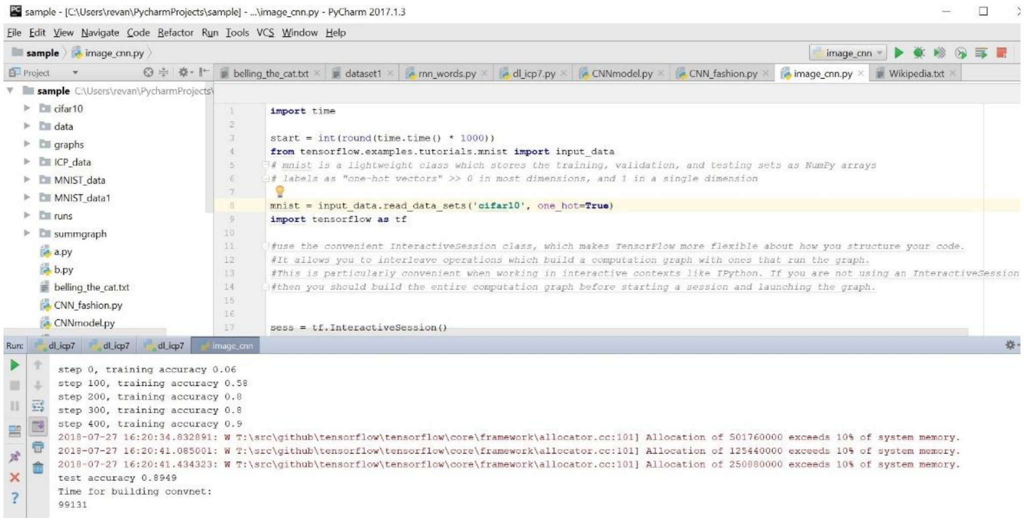
# Task 4: Image Classification with CNN

## Approach/Method:

I have used Cifar10 dataset. Load the data and find the accuracy using all the four optimisers Adam, RMS, Adagrad, Gradient.

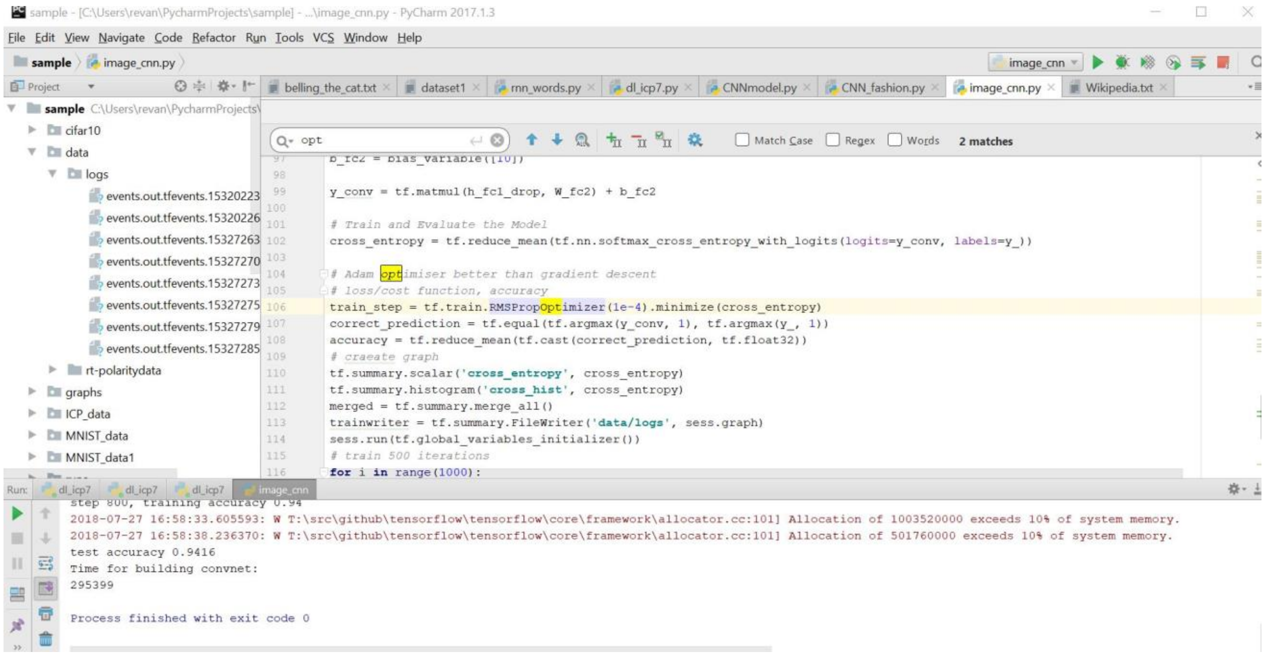
Well Adam optimiser gave the better accuracy compared to other optimiser with the Cifar10 dataset. RMS prop optimiser is also almost second in the race. With 500 steps Adam wins the race with almost 0.90 accuracy. Gradient and dagrad works very poor with Cifar10 dataset. When increased the steps tp 1000 the same trend followed by adam optimiser leading with RMS as second. Adagrad and gradient opt have poor outputs

### Code



```
import time

start = int(round(time.time() * 1000))
from tensorflow.examples.tutorials.mnist import input_data
# mnist is a lightweight class which stores the training, validation, and testing sets as NumPy arrays
# labels as "one-hot vectors" >> 0 in most dimensions, and 1 in a single dimension

mnist = input_data.read_data_sets('cifar10', one_hot=True)
import tensorflow as tf

#use the convenient InteractiveSession class, which makes TensorFlow more flexible about how you structure your code.
#It allows you to interleave operations which build a computation graph with ones that run the graph.
#This is particularly convenient when working in interactive contexts like IPython. If you are not using an InteractiveSession
#then you should build the entire computation graph before starting a session and launching the graph.


sess = tf.InteractiveSession()
```

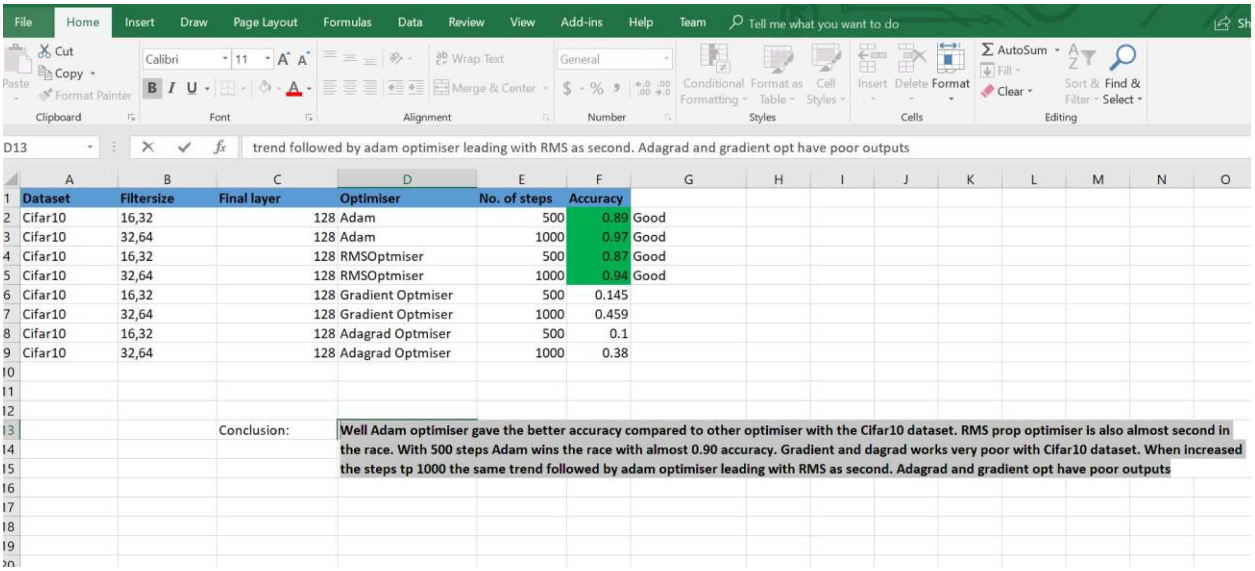
```
step 0, training accuracy 0.06
step 100, training accuracy 0.58
step 200, training accuracy 0.8
step 300, training accuracy 0.8
step 400, training accuracy 0.9
2018-07-27 16:20:34.832891: W T:\src\github\tensorflow\tensorflow\core\framework\allocator.cc:101] Allocation of 501760000 exceeds 10% of system memory.
2018-07-27 16:20:41.085001: W T:\src\github\tensorflow\tensorflow\core\framework\allocator.cc:101] Allocation of 125440000 exceeds 10% of system memory.
2018-07-27 16:20:41.434323: W T:\src\github\tensorflow\tensorflow\core\framework\allocator.cc:101] Allocation of 250880000 exceeds 10% of system memory.
test accuracy 0.8949
Time for building convnet:
99131
```


sample - [C:\Users\revan\PycharmProjects\sample] - ...\image_cnn.py - PyCharm 2017.1.3
File Edit View Navigate Code Refactor Run Tools VCS Window Help
sample image_cnn.py
image_cnn
Project
belling_the_cat.txt
dataset1
rnn_words.py
dl_icp7.py
CNNmodel.py
CNN_fashion.py
image_cnn.py
Wikipedia.txt
sample C:\Users\revan\PycharmProjects\
cifar10
data
logs
events.out.tfevents.15320223
events.out.tfevents.15320226
events.out.tfevents.15327263
events.out.tfevents.15327270
events.out.tfevents.15327273
events.out.tfevents.15327275
events.out.tfevents.15327279
events.out.tfevents.15327285
rt-polaritydata
graphs
ICP_data
MNIST_data
MNIST_data1
opt
Match Case
Regex
Words
2 matches
Run:
dl_icp7
dl_icp7
dl_icp7
image_cnn

```
b_fc2 = bias_variable([10])

y_conv = tf.matmul(h_fc1_drop, W_fc2) + b_fc2

# Train and Evaluate the Model
cross_entropy = tf.reduce_mean(tf.nn.softmax_cross_entropy_with_logits(logits=y_conv, labels=y_))

# Adam optimiser better than gradient descent
# loss/cost function, accuracy
train_step = tf.train.RMSPropOptimizer(1e-4).minimize(cross_entropy)
correct_prediction = tf.equal(tf.argmax(y_conv, 1), tf.argmax(y_, 1))
accuracy = tf.reduce_mean(tf.cast(correct_prediction, tf.float32))
# craeate graph
tf.summary.scalar('cross_entropy', cross_entropy)
tf.summary.histogram('cross_hist', cross_entropy)
merged = tf.summary.merge_all()
trainwriter = tf.summary.FileWriter('data/logs', sess.graph)
sess.run(tf.global_variables_initializer())
# train 500 iterations
for i in range(1000):
```

```
step 800, training accuracy 0.94
2018-07-27 16:58:33.605593: W T:\src\github\tensorflow\tensorflow\core\framework\allocator.cc:101] Allocation of 1003520000 exceeds 10% of system memory.
2018-07-27 16:58:38.236370: W T:\src\github\tensorflow\tensorflow\core\framework\allocator.cc:101] Allocation of 501760000 exceeds 10% of system memory.
test accuracy 0.9416
Time for building convnet:
295399

Process finished with exit code 0
```

### Results

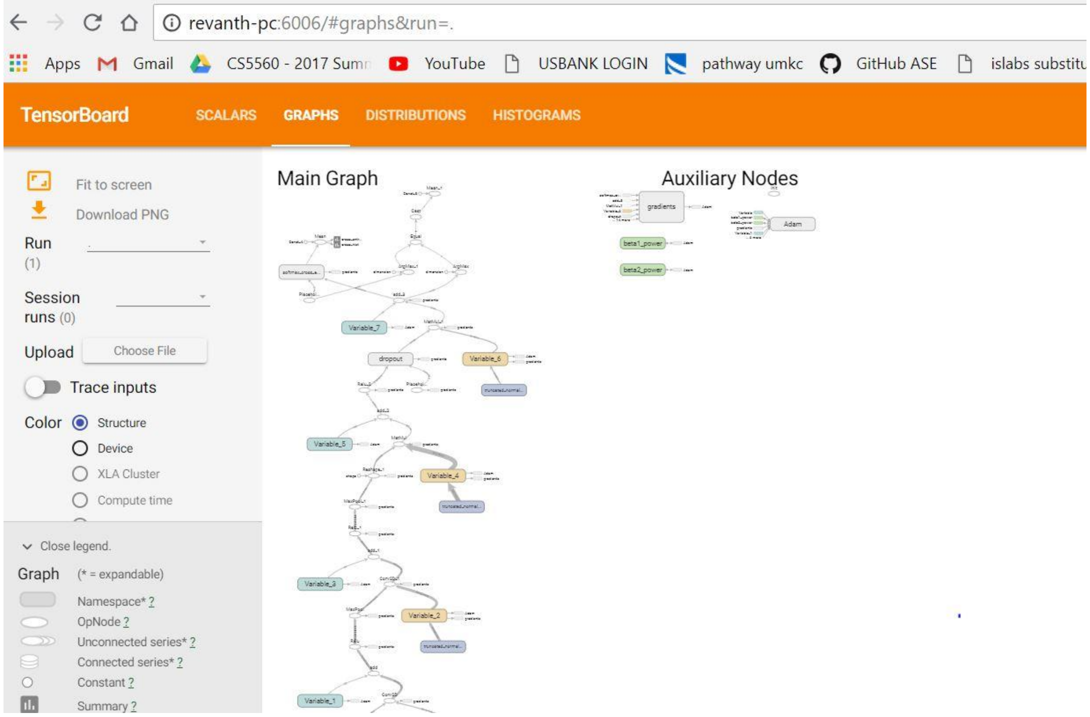


D13 trend followed by adam optimiser leading with RMS as second. Adagrad and gradient opt have poor outputs

Dataset	Filtersize	Final layer	Optimiser	No. of steps	Accuracy	
Cifar10	16,32	128	Adam	500	0.89	Good
Cifar10	32,64	128	Adam	1000	0.97	Good
Cifar10	16,32	128	RMSOptmiser	500	0.87	Good
Cifar10	32,64	128	RMSOptmiser	1000	0.94	Good
Cifar10	16,32	128	Gradient Optmiser	500	0.145	
Cifar10	32,64	128	Gradient Optmiser	1000	0.459	
Cifar10	16,32	128	Adagrad Optmiser	500	0.1	
Cifar10	32,64	128	Adagrad Optmiser	1000	0.38	

Conclusion: **Well Adam optimiser gave the better accuracy compared to other optimiser with the Cifar10 dataset. RMS prop optimiser is also almost second in the race. With 500 steps Adam wins the race with almost 0.90 accuracy. Gradient and dagrad works very poor with Cifar10 dataset. When increased the steps tp 1000 the same trend followed by adam optimiser leading with RMS as second. Adagrad and gradient opt have poor outputs**

### Tensor Board graphs



revanth-pc:6006/#graphs&run=.

Run (1)

Session runs (0)

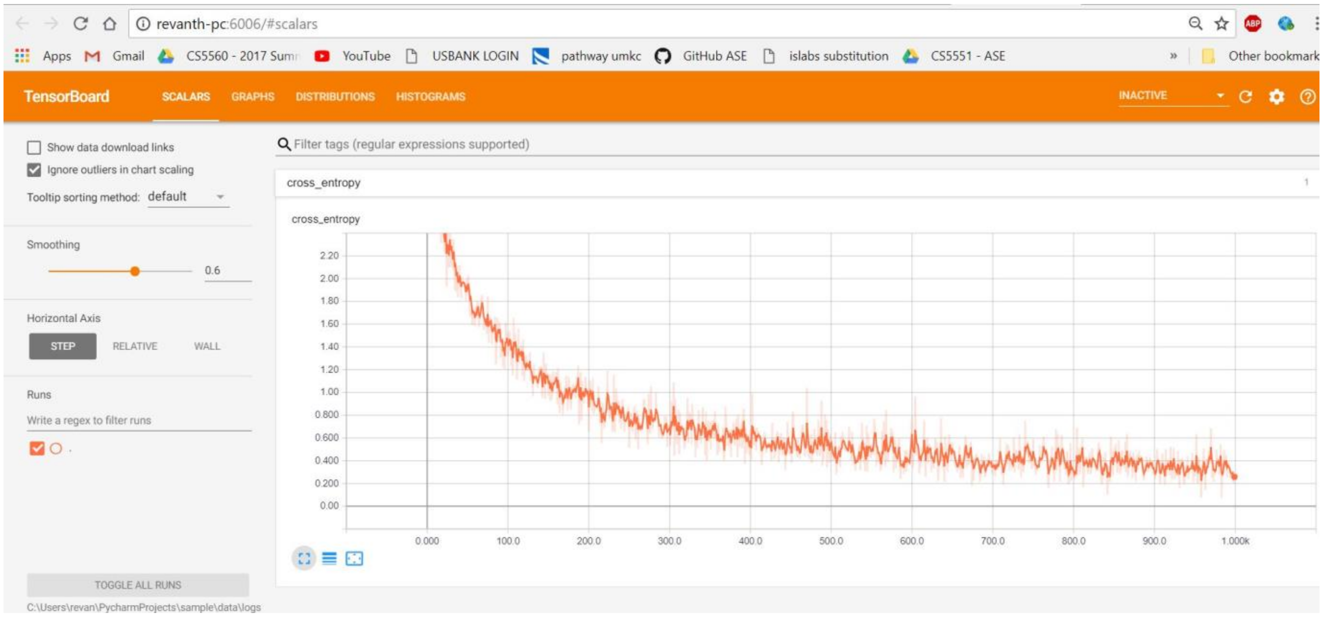
Upload

○ Trace inputs

Color ● Structure ○ Device ○ XLA Cluster ○ Compute time

Graph (* = expandable)

Namespace* ?
OpNode ?
Unconnected series* ?
Connected series* ?
Constant ?
Summary ?



revanth-pc:6006/#scalars

☐ Show data download links

☑ Ignore outliers in chart scaling

Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis [✓ STEP] RELATIVE WALL

Runs

Write a regex to filter runs

☑ ○ .

C:\Users\revan\PycharmProjects\sample\data\logs