UNIVERSITY AT BUFFALO, STATE UNIVERSITY OF NEW YORK

COMPUTER SCIENCE

CSE474/574: INTRODUCTION TO MACHINE LEARNING(FALL 2017)

Project 4: Introduction to Deep Learning

Author: Shivraj D. WABALE
Person Number: 50248709
Author: Mayur M. POPADE
Person Number: 50246313

Instructor: Dr. Sargur N. Srihari
TAs: Junchu, Mengdi Huai, Tianhang Zheng, Junfei Wang

December 6, 2017

Contents

1 Abstract

- To implement the binary CNN classification on CelabA
- To implement augmentation of images
- To tune hyper-parameters
- To analyze impact of regularization method like dropout
- To analyze impact of changing image resolution, bigger training set on accuracy

2 Note

- The code is developed and tested in AWS deep learning AMI, the code was developed in Python3 using Tensorflow.
- There is main.py which run the code and get the accuracy of the test data in each iteration.
- The hyper parameter are tuned based upon the results from the validation dataset(train and test). The data is generated using the input queue of the Tensorflow.
- The code submitted generate two datasets, validation dataset where using during the tuning of the model.
- The config.py take the parameters required for training and testing.
- shallow_cnn_celeba.py is having the shallow convolutional neural network architecture.

3 Introduction

There are four parts of architectures :

1. The custom cnn with configurations 1.
2. The custom cnn with configurations 2.
3. The custom cnn with configurations 3 along with the data augmentations.
4. The shallow cnn with configurations mentioned in the paper.

Details of the configuration are mentioned in the section of 'result of various configurations'.

4 Dropout - Regularization Method

- Neural Networks with a large number of parameters is efficient way to train a model. But sometimes it may result in a over-fitting which is a serious issue. Overfitting is handled by dropout, where we randomly drop units along with their connection during training.
- Dropping a unit means temporarily removing it from the network along with its incoming and outgoing connections. Each unit is dropped with a fixed probability of p. But for wide variety of networks optimum value of p is 0.5.
- For input units, value of p is closer to 1 than to 0.5
- We tried changing value of dropout rate from 0.3 to 0.8 with increment of 0.1 in each step. The accuracy of the network is maximum for dropout rate of 0.5.

5 Tuning Hyper-parameters

We tried changing values of hyper-parameters like dropout rate, number of layers and number of nodes in each layer. Following is observed impact of hyper-parameters on accuracy of the model.

5.1 Changing number of nodes

- We changed values of number of nodes in layers from 8 to 128 on complete dataset.
- We did not observe any significant change in the value of accuracy. But with fine tuning accuracy can be increased.

5.2 Changing number of layers

- We added one additional layer and trained the model on dataset.
- There is no significant gain in terms of accuracy for the model. Accuracy increased marginally by 0.01%.
- So adding an extra layer did not have any impact on the accuracy.

6 Impact of using Higher Resolutions

- Training model on inputs with higher resolutions is very lengthy.
- Instead of input image size of 28 x 28, we resized input images to dimensions 512 x 512. The model was running on 'Springsteen' server for 24 hours.
- Accuracy achieved with higher resolution is 93.27. There is no significant gain in accuracy for considering inputs with higher resolutions.

7 Impact of bigger sizes of training set

- After completing coding part, at first we ran the model on a training set of 5000 input images. We got accuracy of 91.53%.
- With increase in number of input images, accuracy of the model increased by small value in each step.
- For training on 1,80,000 images, we achieved highest accuracy of 97.3% on validation dataset.

8 Tensor Graphs

8.1 Without image standardization

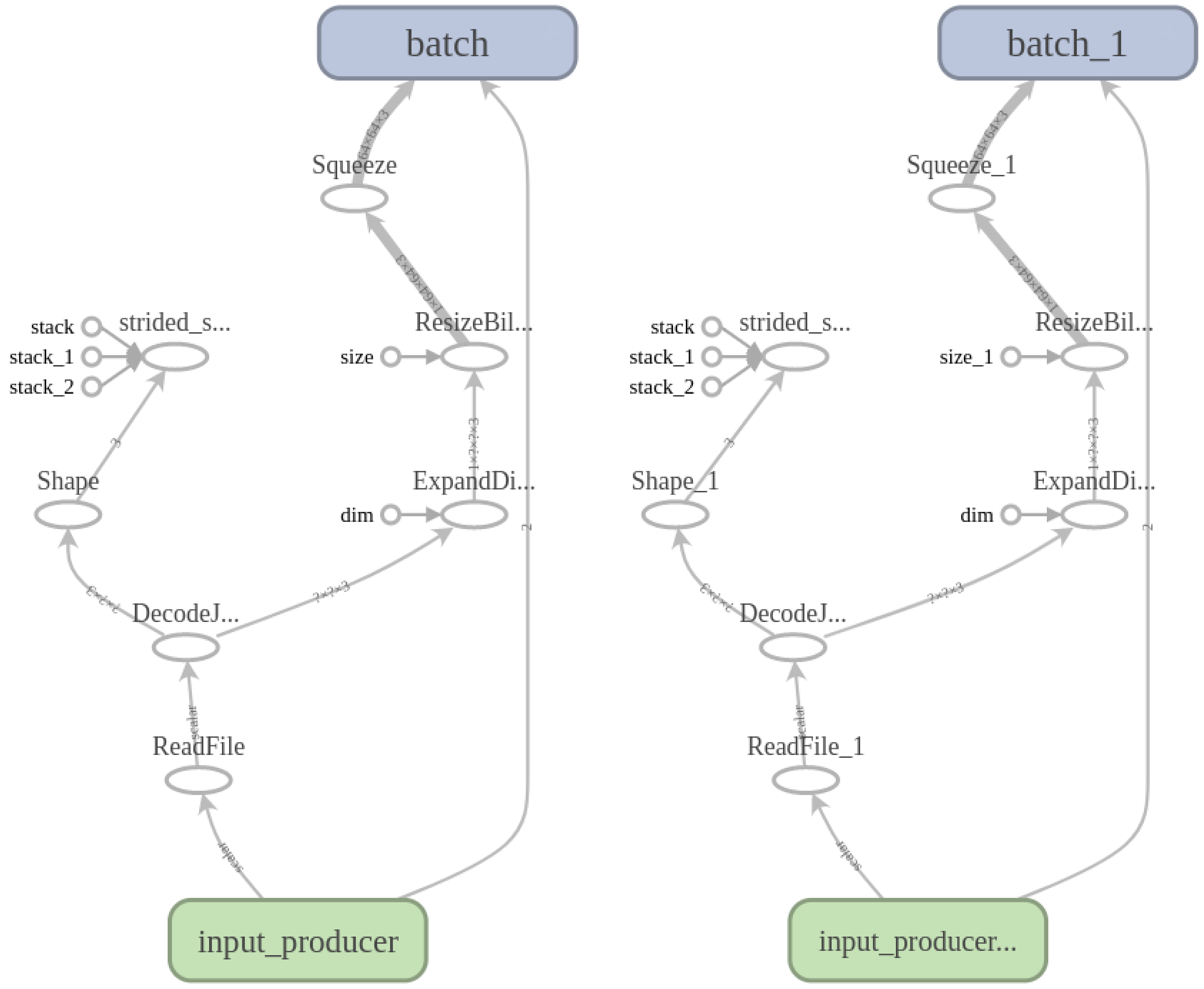


Figure 1: Without image standardization

8.2 With image standardization

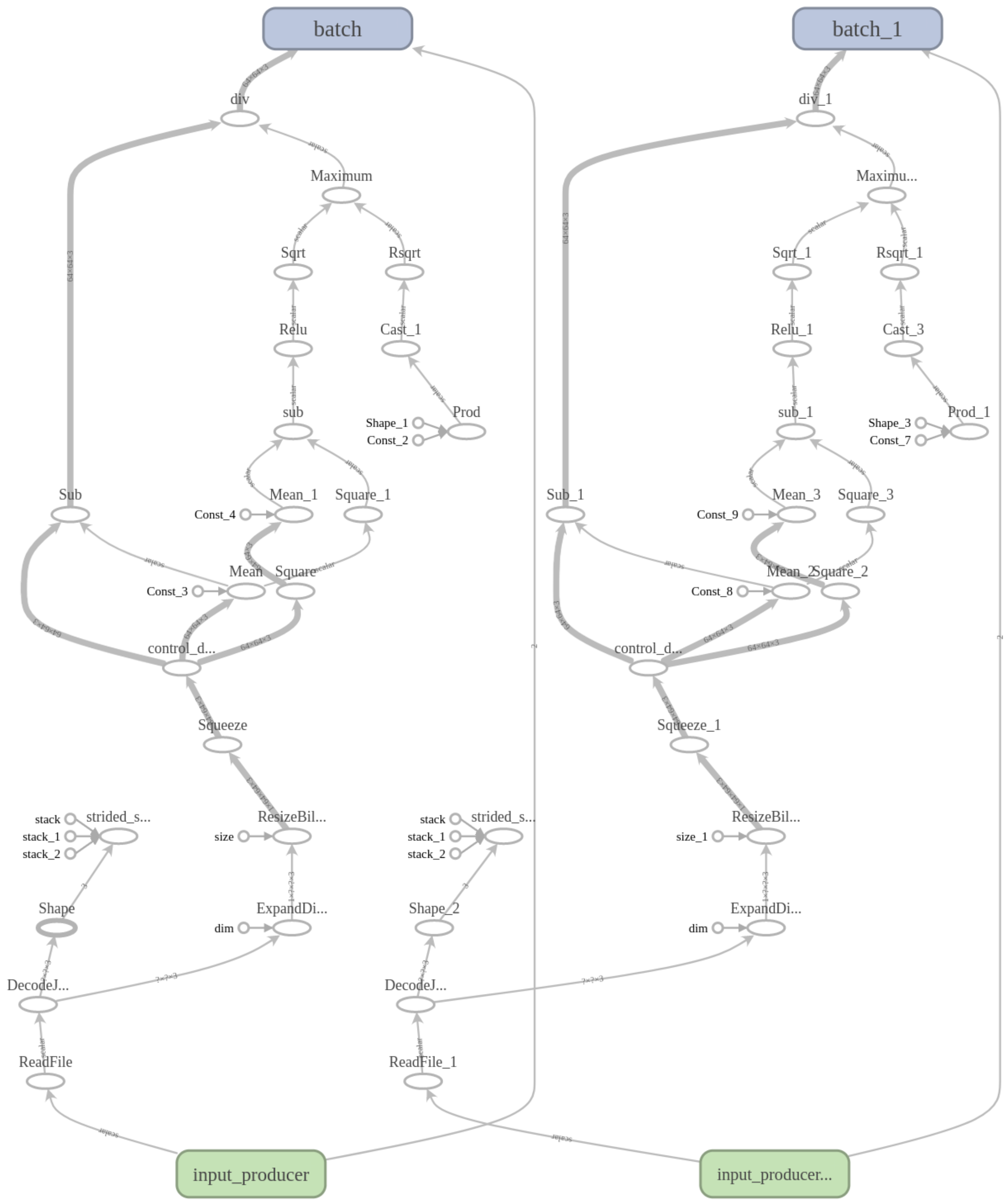


Figure 2: Image standardization

8.3 Reading input with augmentation with image standardization

For data augmentation we have changed the contrast with lower=0.2, upper=1.8, brightness with max_delta=63, flipped the image and cropped the image all with uniform randomness. The images where augmented depending upon the value of 'distortion_range' defined in the 'config.py'. This value decide the probability of image to be augmented. If value is 2 then the probability of a image being augmented is 0.5, if it's 3 then probability is 1/3, it 4 then 0.25 and so on. The variable 'augmentation' decided to do augmentation or not, True for augmentation and False for without augmentation.

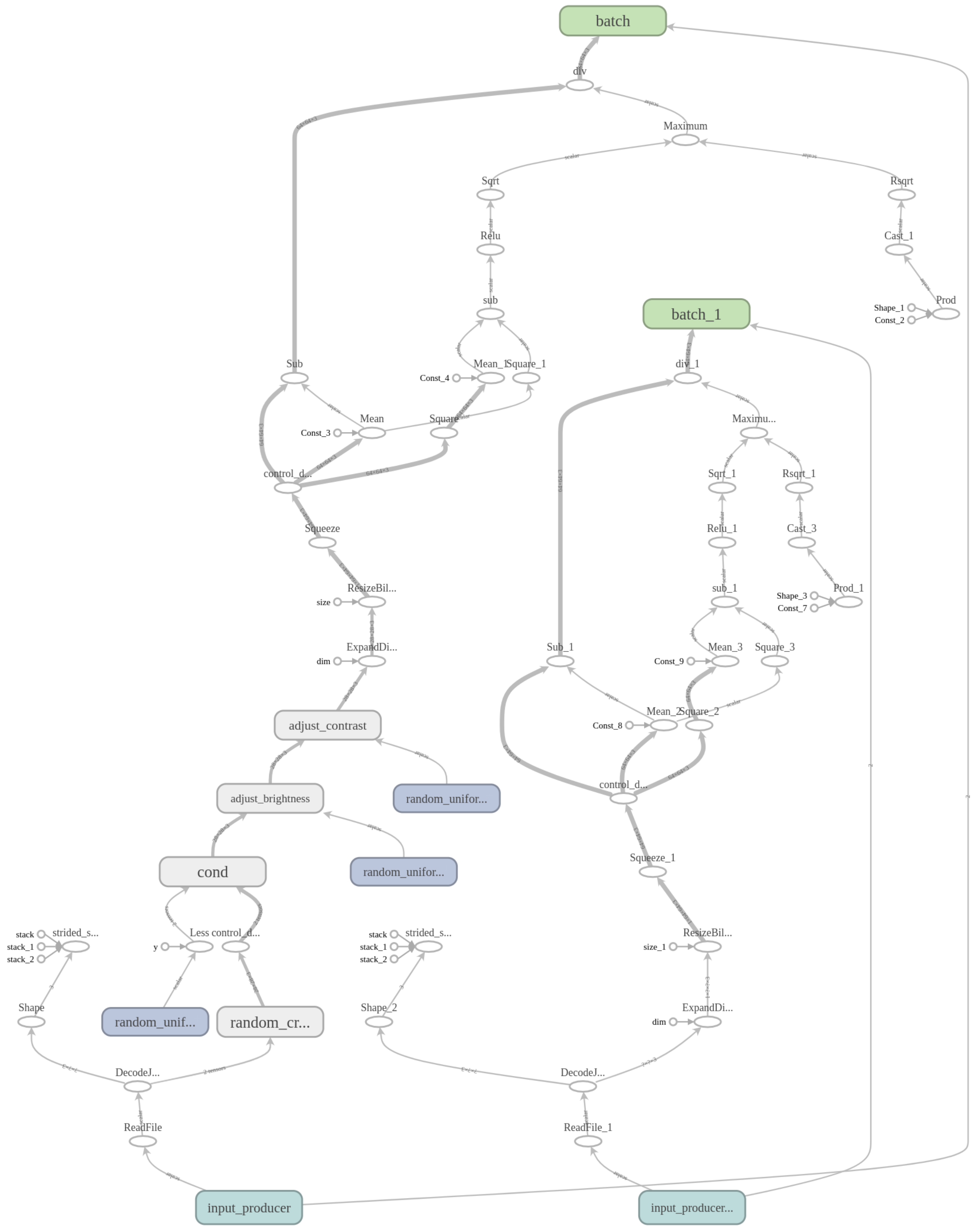


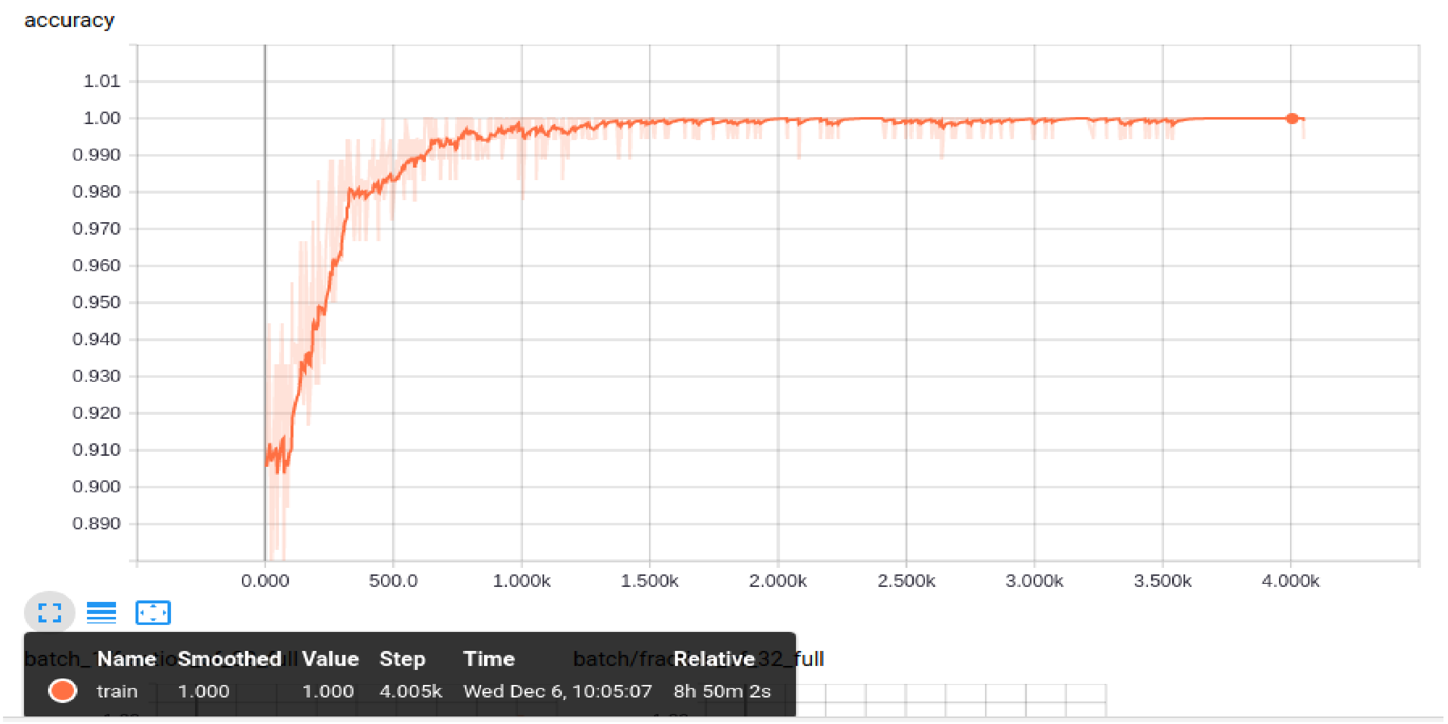
Figure 3: Reading input with augmentation with image standardization

9 Result of various configurations

Conf 1 and conf 2 results are from the input without augmentation. For each changes in the input dimension we need to make changes in read_input.py in the resize command to match the input size of the architecture.

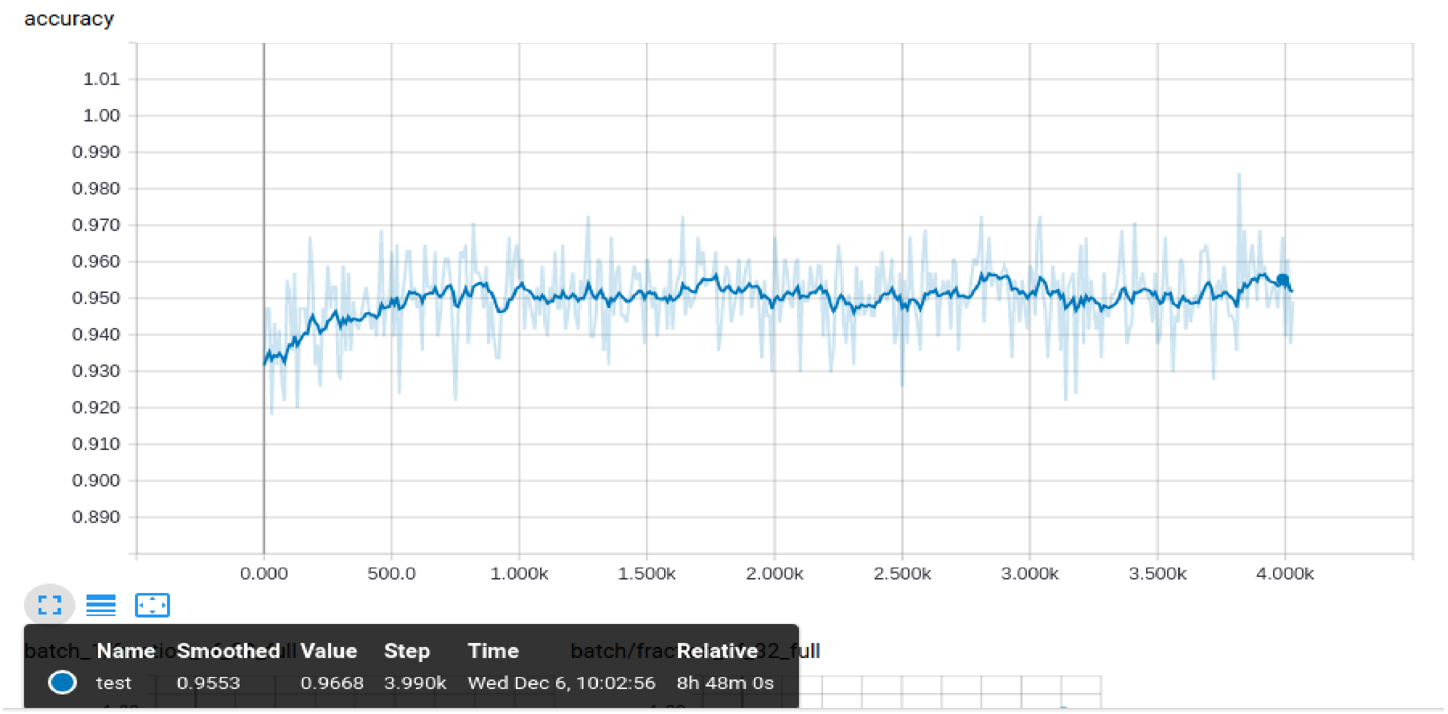
9.1 Results with configurations 1

This configuration where designed using 3 conv layers each with relu and 2*2 max pooling and then at end 1024 and 2 nodes FC layers. The conv layers has 3*3, 3*3 and 5*5 kernel size with filters of 8, 16 and 32 in first 3 conv layers respectively. The input dimension where 512 * 512 with all 3 channels. Below graphs show the results achieved during the training and testing. File name : cnn_celeba_arch1.py



Name	Smoothed	Value	Step	Time	Relative
● train	1.000	1.000	4.005k	Wed Dec 6, 10:05:07	8h 50m 2s

Figure 4: Train Accuracy with configuration 1



Name	Smoothed	Value	Step	Time	Relative
test	0.9553	0.9668	3.990k	Wed Dec 6, 10:02:56	8h 48m 0s

Figure 5: Test Accuracy with configuration 1

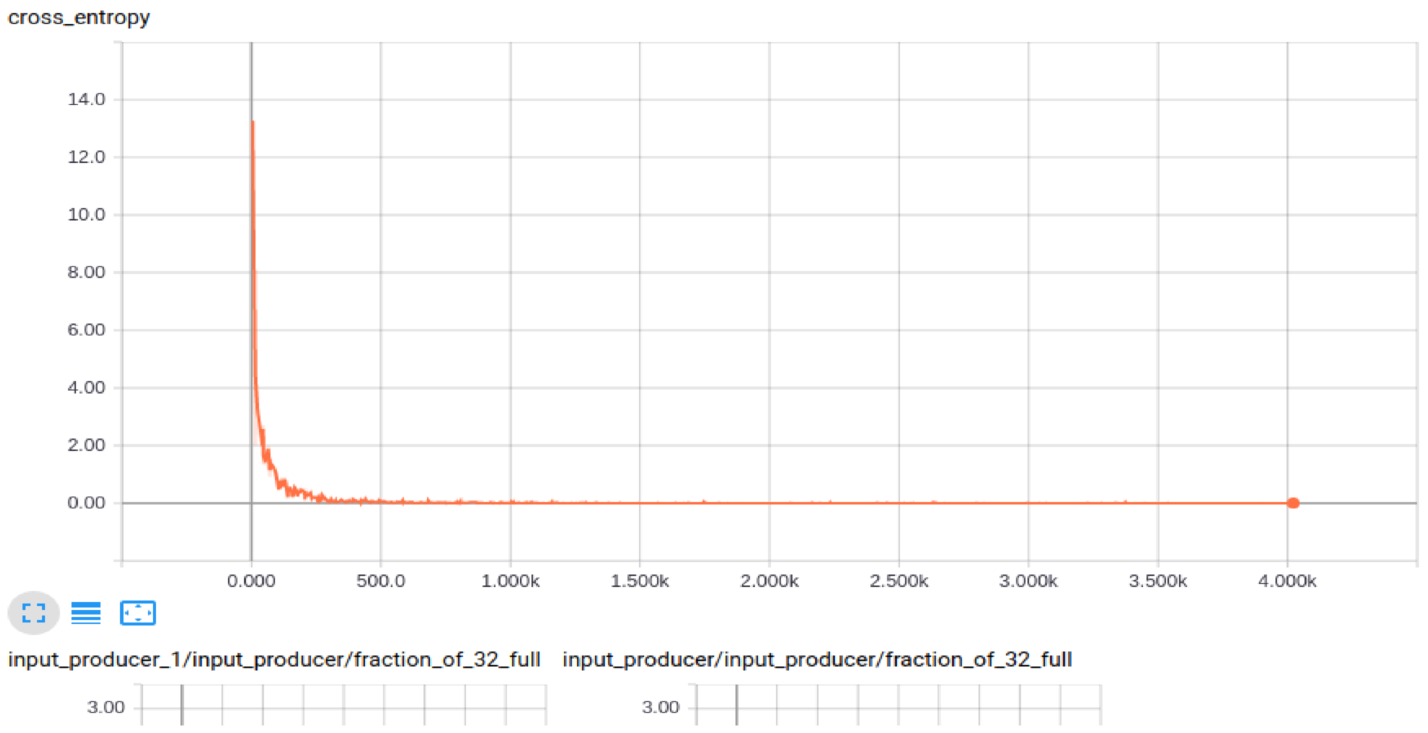


Figure 6: Train Error with configuration 1

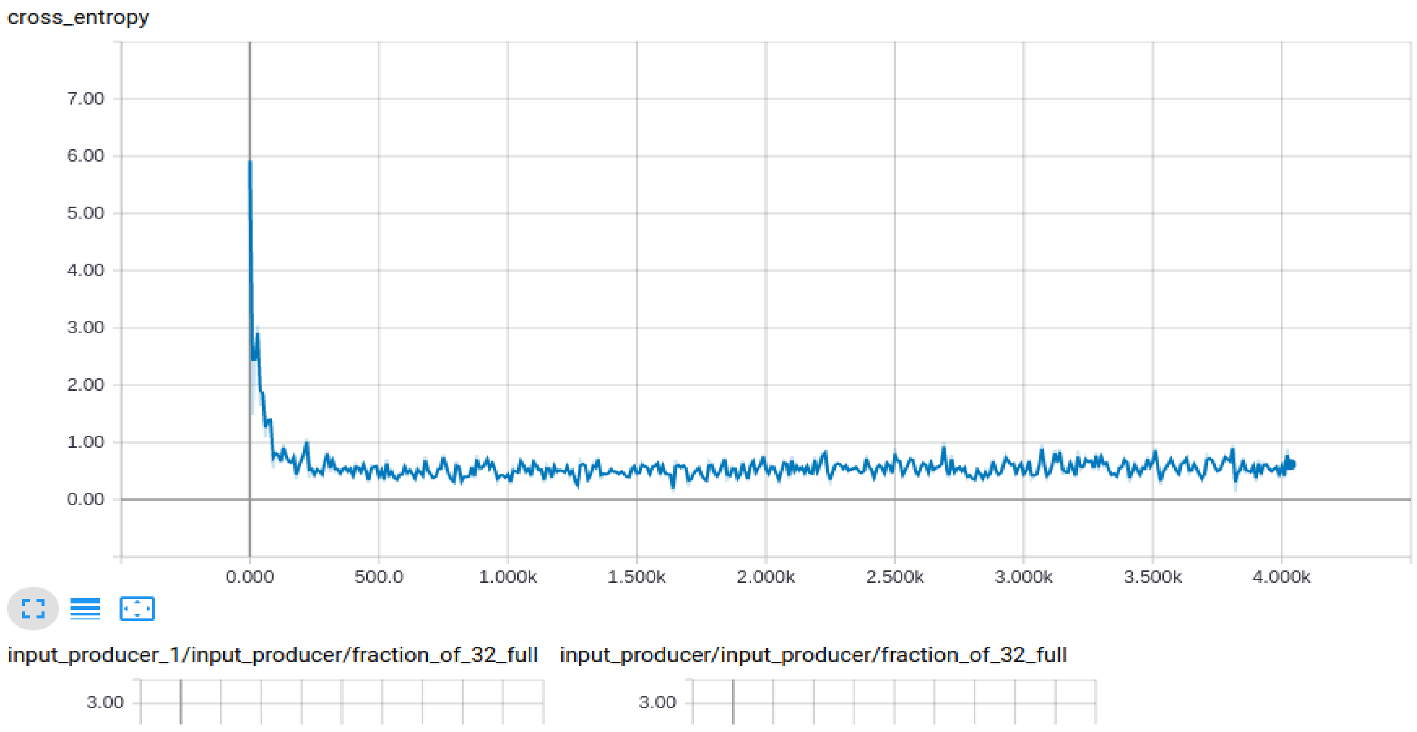


Figure 7: Test Error with configuration 1

9.2 Results with configurations 2

This architecture where designed using 3 conv layers each with relu and 2*2 max pooling and then at end 1024 and 2 nodes FC layers. The conv layers has 5*5, 5*5 and 7*7 kernel size with filters of 32, 64 and 128 in first 3 conv layers respectively. The input dimension where 64 * 64 with all 3 channels. Below graphs show the results achieved during the training and testing.

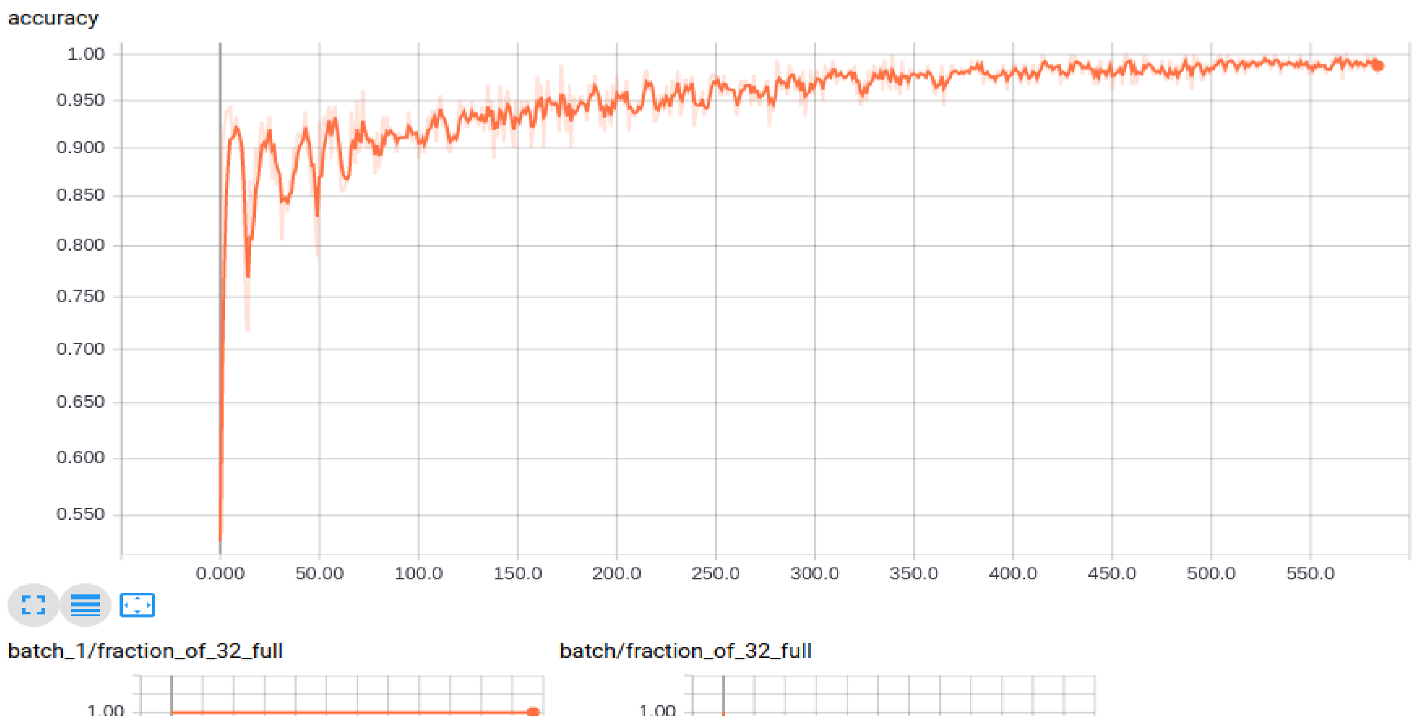


Figure 8: Train Accuracy with configuration 2

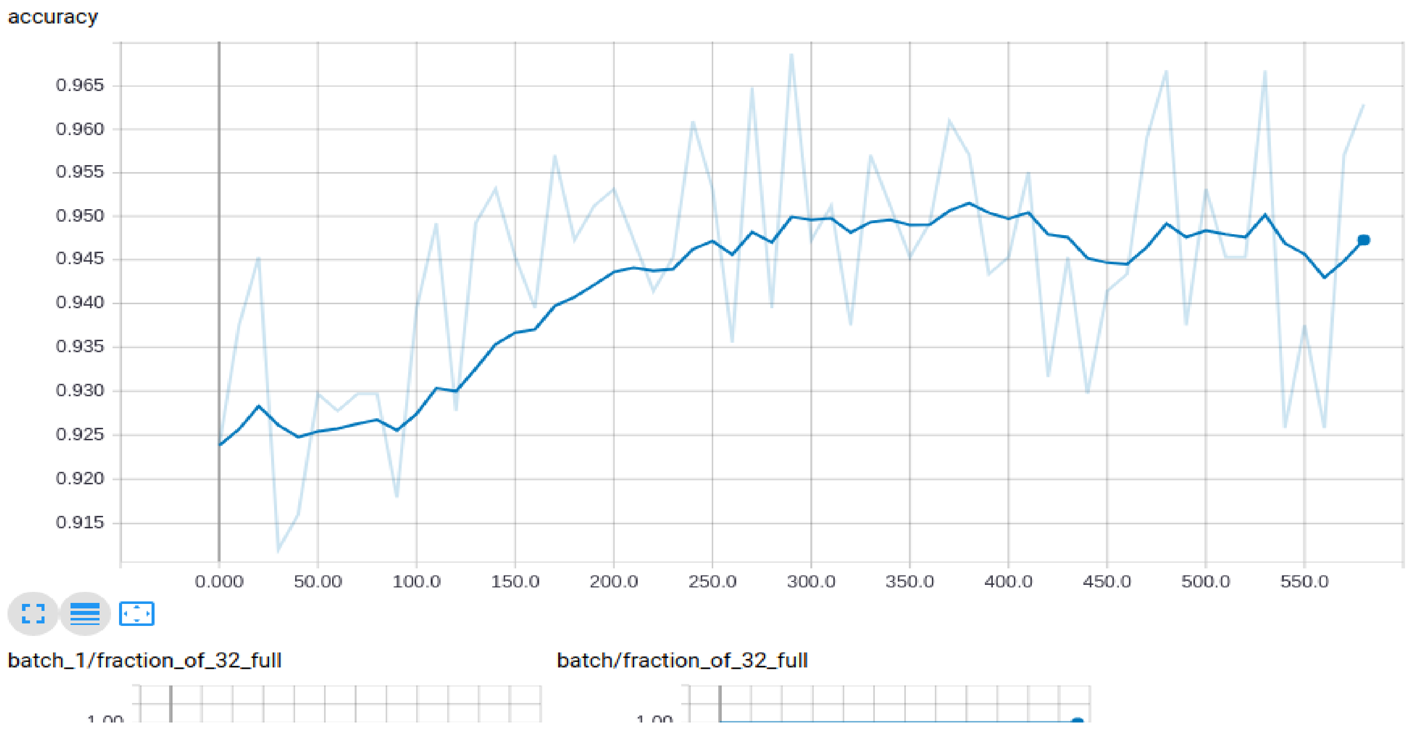


Figure 9: Test Accuracy with configuration 2

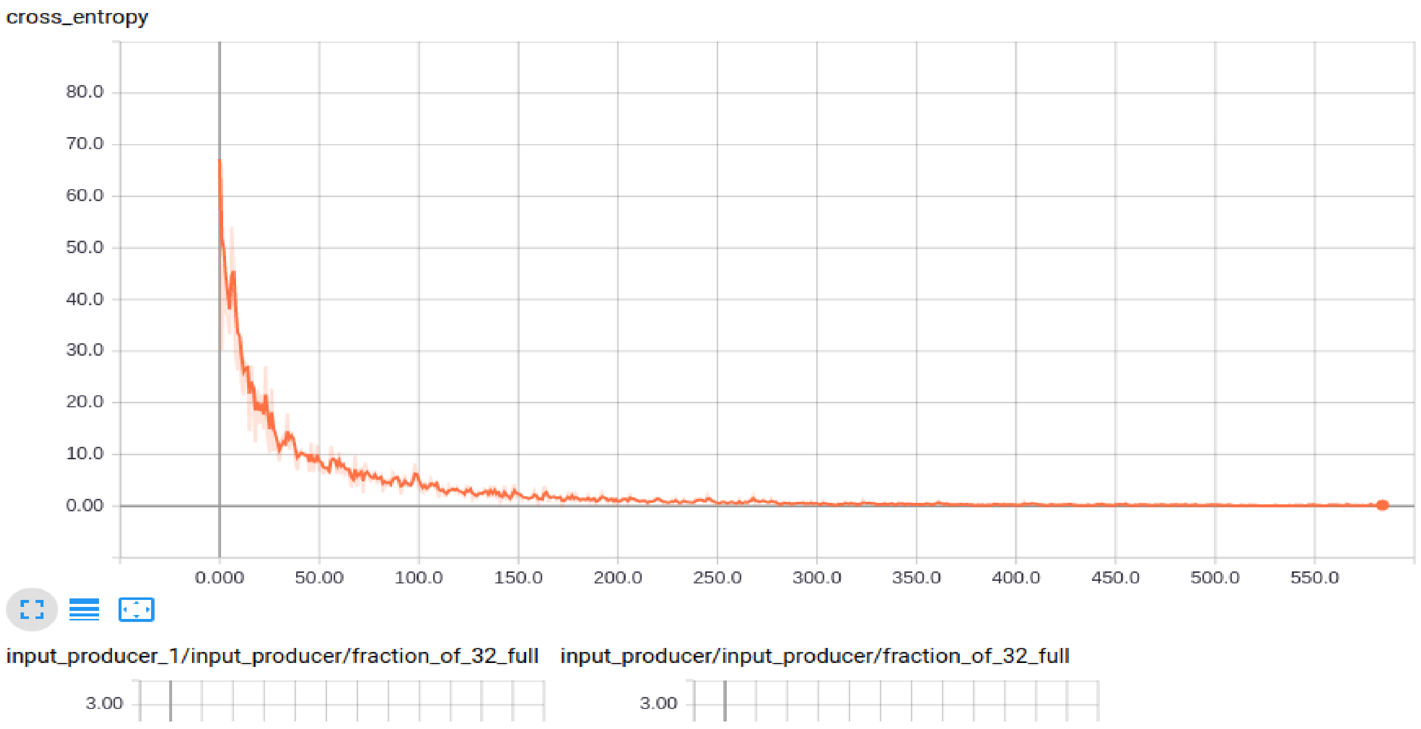


Figure 10: Train Error with configuration 2

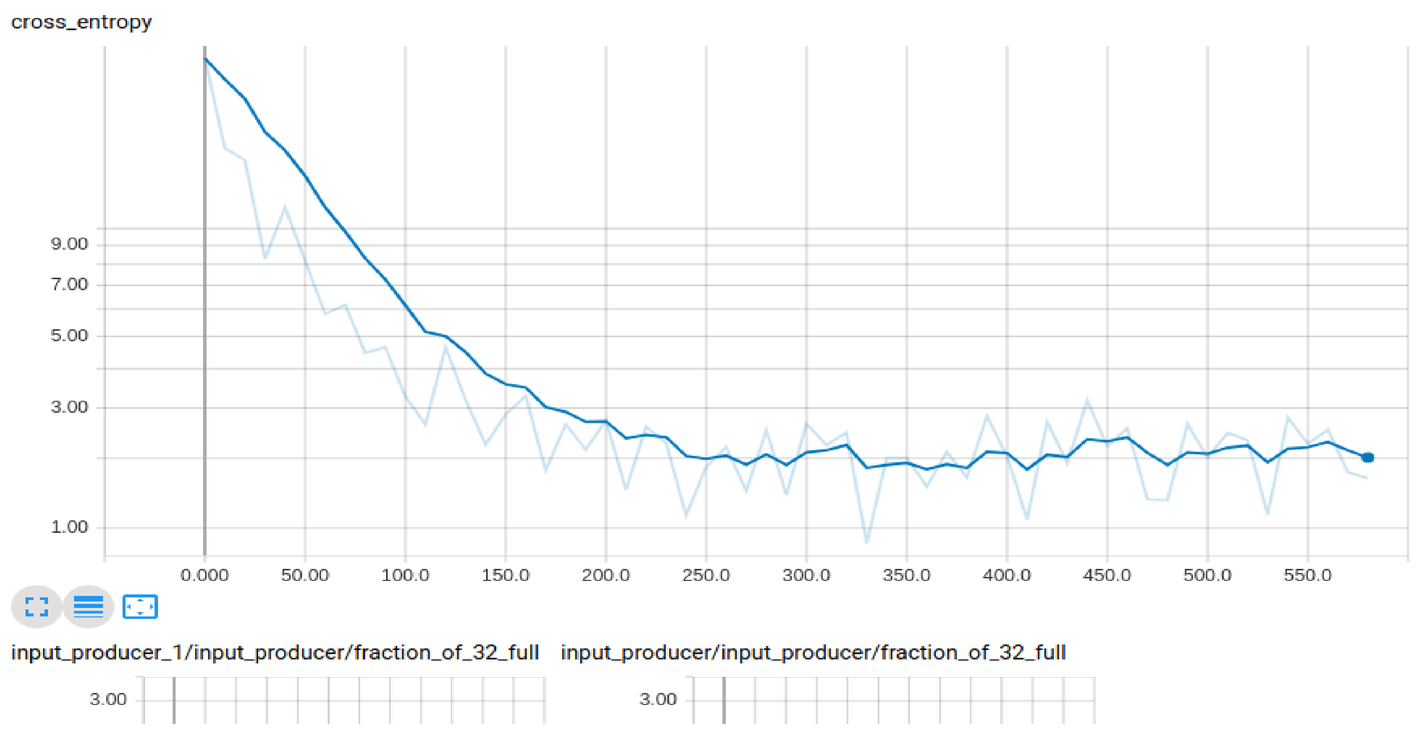
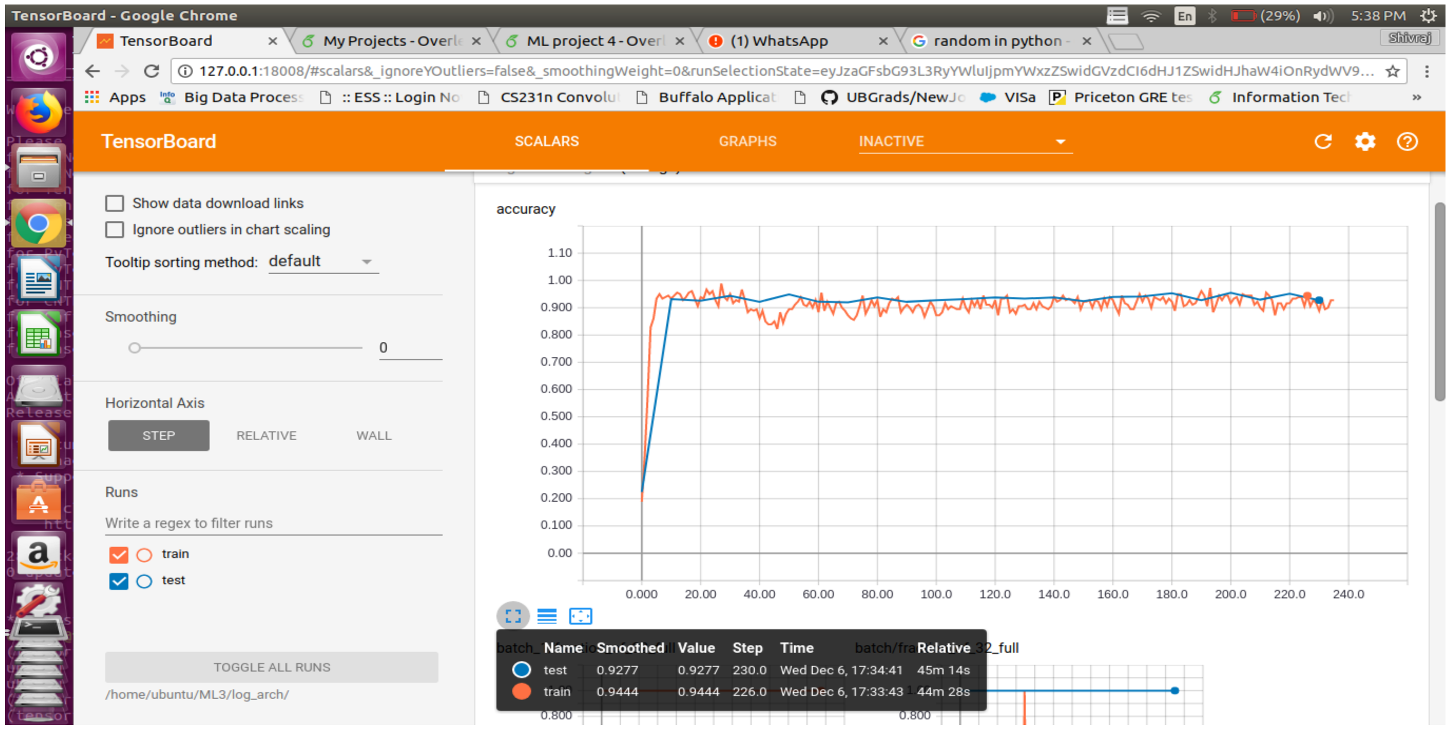


Figure 11: Test Error with configuration 2

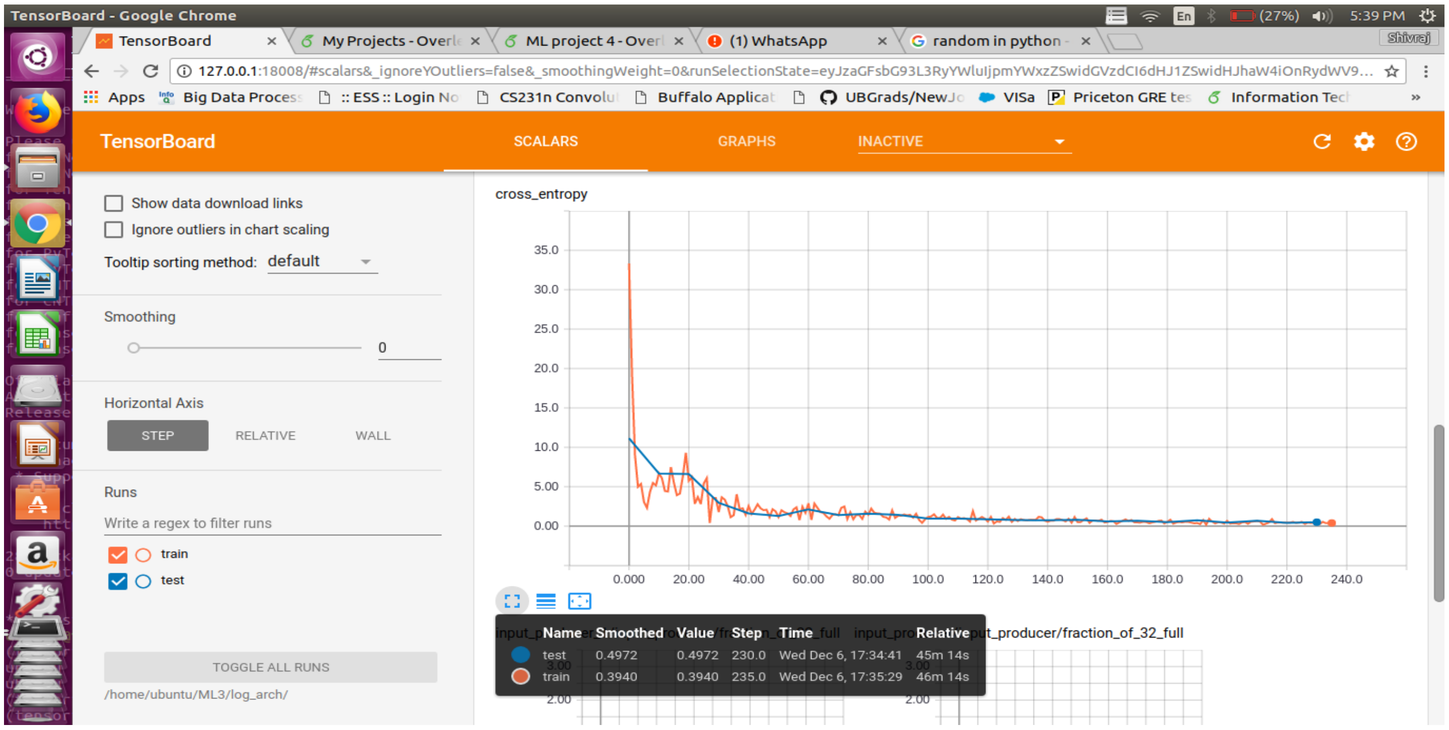
9.3 Results with configurations 3

This architecture where designed using 3 conv layers each with relu and 2*2 max pooling and then at end 512 and 2 nodes FC layers. The conv layers has 5*5, 5*5 and 7*7 kernel size with filters of 8, 32 and 16 in first 3 conv layers respectively. The input dimension where 64 * 64 with all 3 channels. Below graphs show the results achieved during the training and testing. This architecture was ran on the augmented input with probability of augmentation of a image as 0.5.



Name	Smoothed	Value	Step	Time	Relative
test	0.9277	0.9277	230.0	Wed Dec 6, 17:34:41	45m 14s
train	0.9444	0.9444	226.0	Wed Dec 6, 17:33:43	44m 28s

Figure 12: Train Test Accuracy with configuration 3



Show data download links
Ignore outliers in chart scaling
Tooltip sorting method: default

Smoothing: 0

Horizontal Axis: [✓ STEP] RELATIVE WALL

Runs
Write a regex to filter runs
[✓ train]
[✓ test]

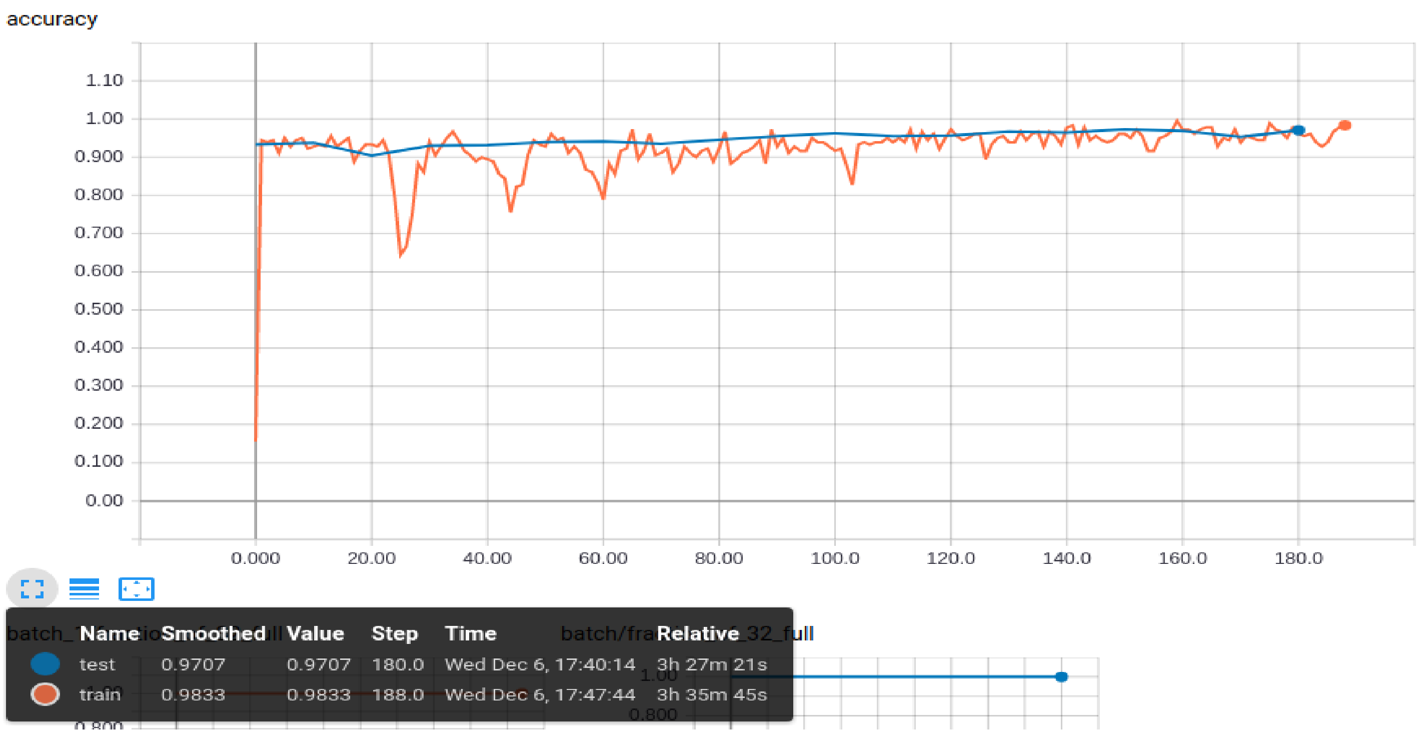
/home/ubuntu/ML3/log_arch/

	Name	Smoothed	Value	Step	Time	Relative
🔵	test	0.4972	0.4972	230.0	Wed Dec 6, 17:34:41	45m 14s
🟠	train	0.3940	0.3940	235.0	Wed Dec 6, 17:35:29	46m 14s

Figure 13: Train Test Error with configuration 3

10 Shallow convolutional neural network

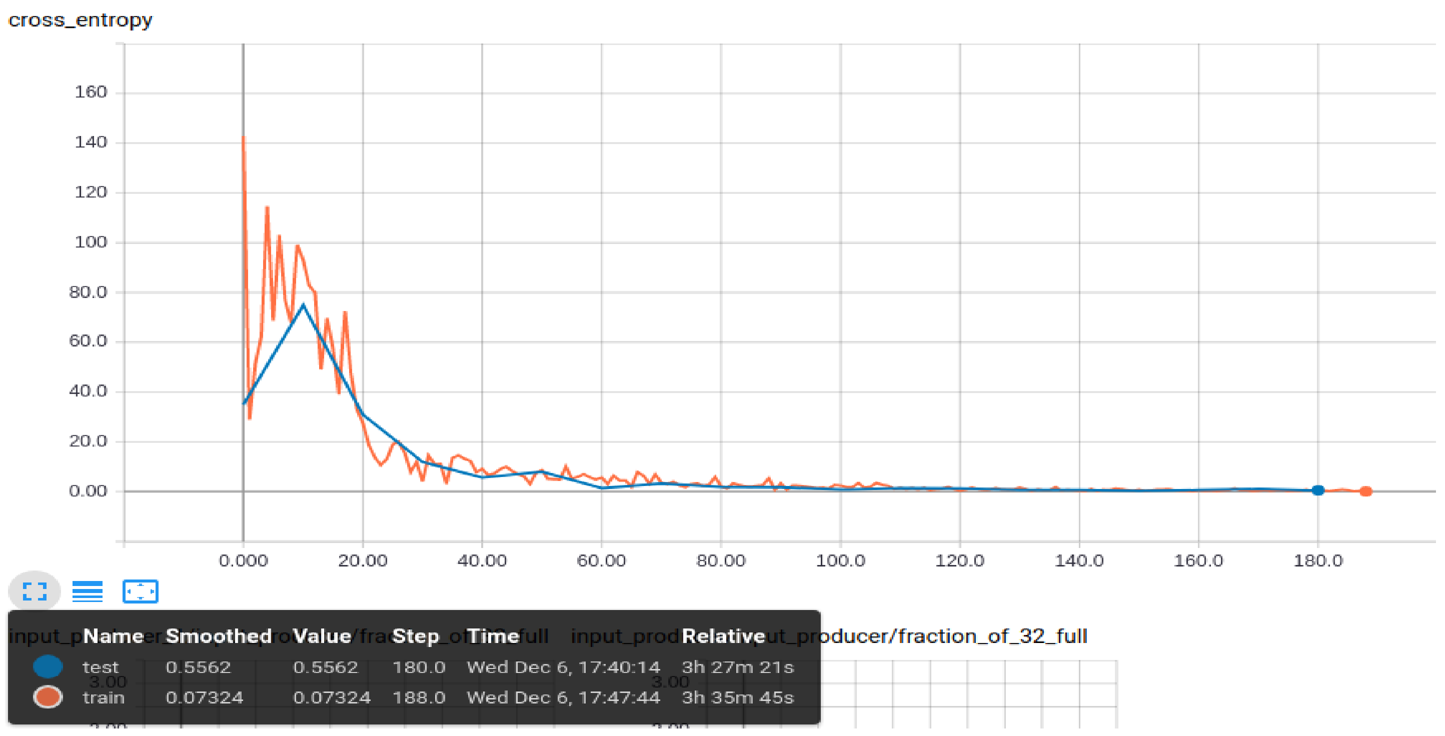
The shallow convolutional neural network was inspired by GoogleNet[1]. We ran the CNN architecture in the paper and did not ran the SVM structure and found below output. This is the best result we achieved in all the architecture. File name : shallow_cnn_celeba.py



	Name	Smoothed	Value	Step	Time	Relative
🔵	test	0.9707	0.9707	180.0	Wed Dec 6, 17:40:14	3h 27m 21s
🟠	train	0.9833	0.9833	188.0	Wed Dec 6, 17:47:44	3h 35m 45s

Figure 14: Train Test Accuracy with Shallow CNN

[1]http://ieeexplore.ieee.org.gate.lib.buffalo.edu/document/8101617/?part=undefined%7Cdeqn1#deqn1



	Name	Smoothed	Value	Step	Time	Relative
●	test	0.5562	0.5562	180.0	Wed Dec 6, 17:40:14	3h 27m 21s
●	train	0.07324	0.07324	188.0	Wed Dec 6, 17:47:44	3h 35m 45s

Figure 15: Train Test Error with Shallow CNN

11 Comparing Accuracy for Different Architecture

	Train	Test
Arch1	1	96.68
Arch2	98.79	94.56
Arch3	94.44	92.77
Shallow	98.33	**97.07**

12 Comparing Cross Entropy for Different Architecture

	Train	Test
Arch1	$3 * 10^{-3}$	0.78
Arch2	$2.4 * 10^{-4}$	2
Arch3	0.3940	0.4972
Shallow	0.07324	**0.5562**

13 Interesting Observation

- We observed the best result with the Shallow CNN architecture.
- The augmentation of the images could help to achieve the generalization and the better testing results. But as we've limited time and limited resources to run, we could not achieve the expected results.
- In custom architecture we've seen that the training accuracy and errors get better and better but it does not affect the accuracy and the errors of the test and validation dataset.
- In data augmentation we have observed that the accuracy and the error of test and train data set various uniformly, which have given the better results than training on dataset without augmentation.
- We need augmentation data to train the CNN because it has some drawbacks. CNN can not detect the rotation, it detects the images in parts etc. The recent paper of Geoffrey E Hinton propose new idea of CapNets which has overcome these drawbacks on MNIST dataset.[2]
- But in Shallow CNN, we can see from the graphs that the result of the accuracy and the error for both the datasets namely train and test is uniform. Hence shallow architecture gives the best results compared to others.

14 One hot representation

The labels where represented in 2 classed using the one hot vector representation. The eyeglasses images where represented as [1 0] and images without eyeglasses where represented as [0 1].

[2]https://arxiv.org/abs/1710.09829

15 Conclusion

Successfully implemented data augmentation, CNN with different configurations. We've also implemented the shallow CNN and achieved best results in this assignment. The best accuracy we achieved is of around **97.07%**.