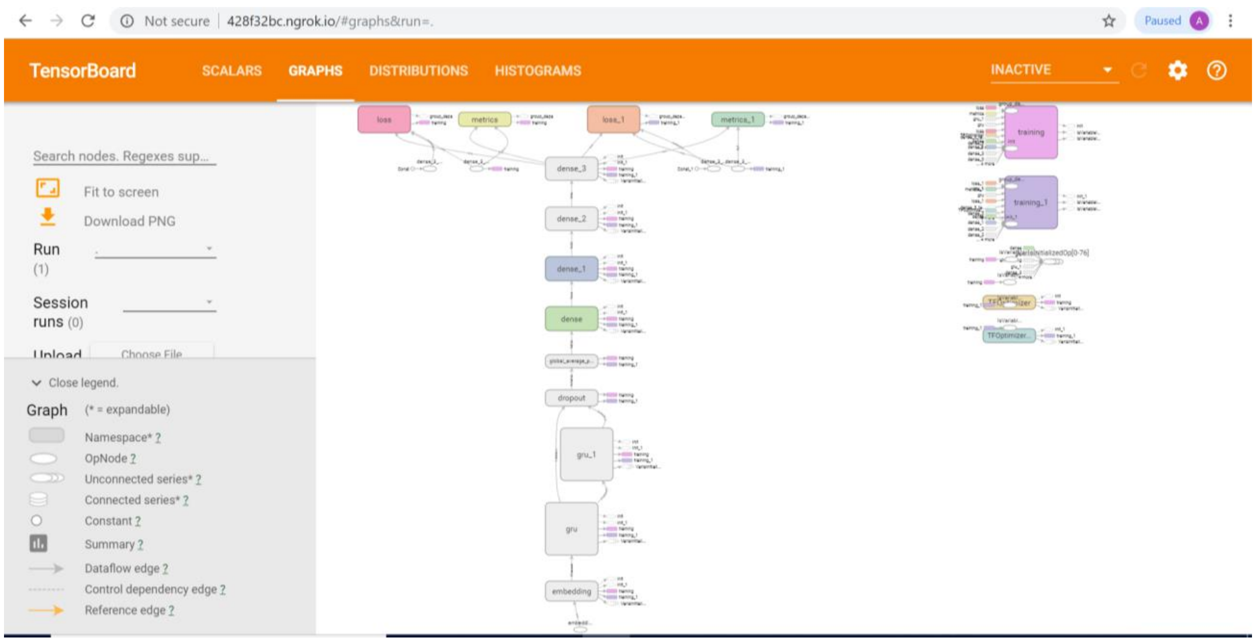
Deep Learning Homework 2

Team:
Ayan Agrawal – aka398
Joshua Abraham – jma672

Name: Ayan Agrawal

Final Architecture:

```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, None, 126)         1260000
_________________________________________________________________
gru (GRU)                    (None, None, 512)         981504
_________________________________________________________________
gru_1 (GRU)                  (None, None, 256)         590592
_________________________________________________________________
dropout (Dropout)            (None, None, 256)         0
_________________________________________________________________
global_average_pooling1d (Gl (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 64)                16448
_________________________________________________________________
dense_1 (Dense)              (None, 32)                2080
_________________________________________________________________
dense_2 (Dense)              (None, 16)                528
_________________________________________________________________
dense_3 (Dense)              (None, 1)                 17
=================================================================
Total params: 2,851,169
Trainable params: 2,851,169
Non-trainable params: 0
_________________________________________________________________
```

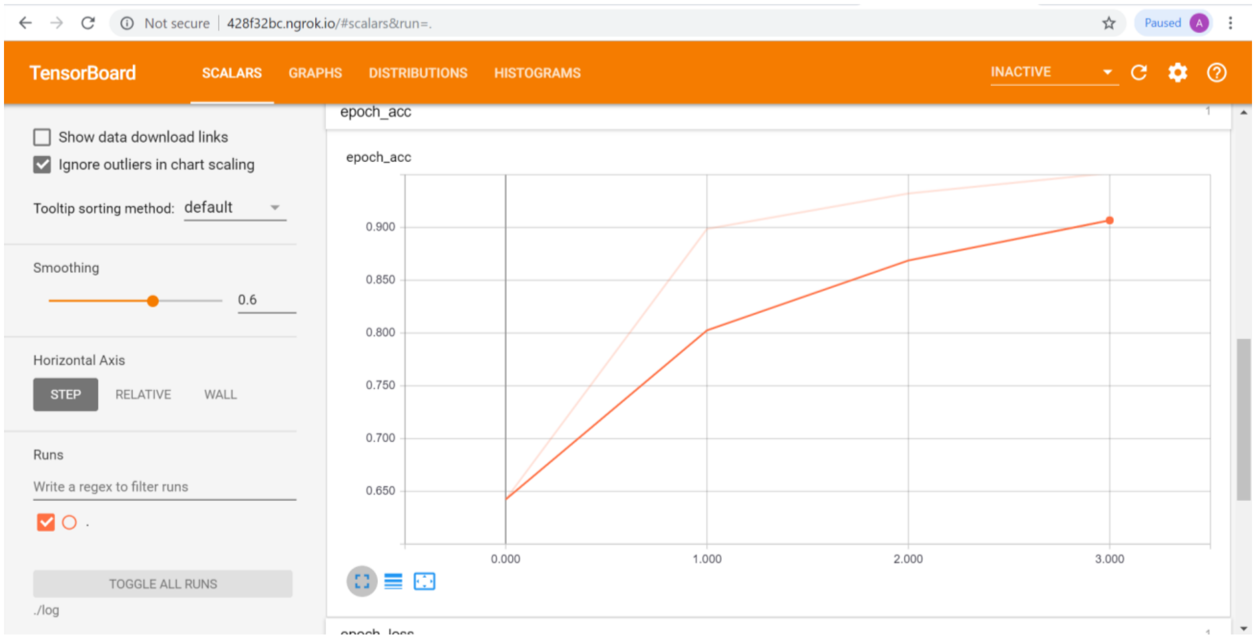


Loss Function Used: binary_crossentropy

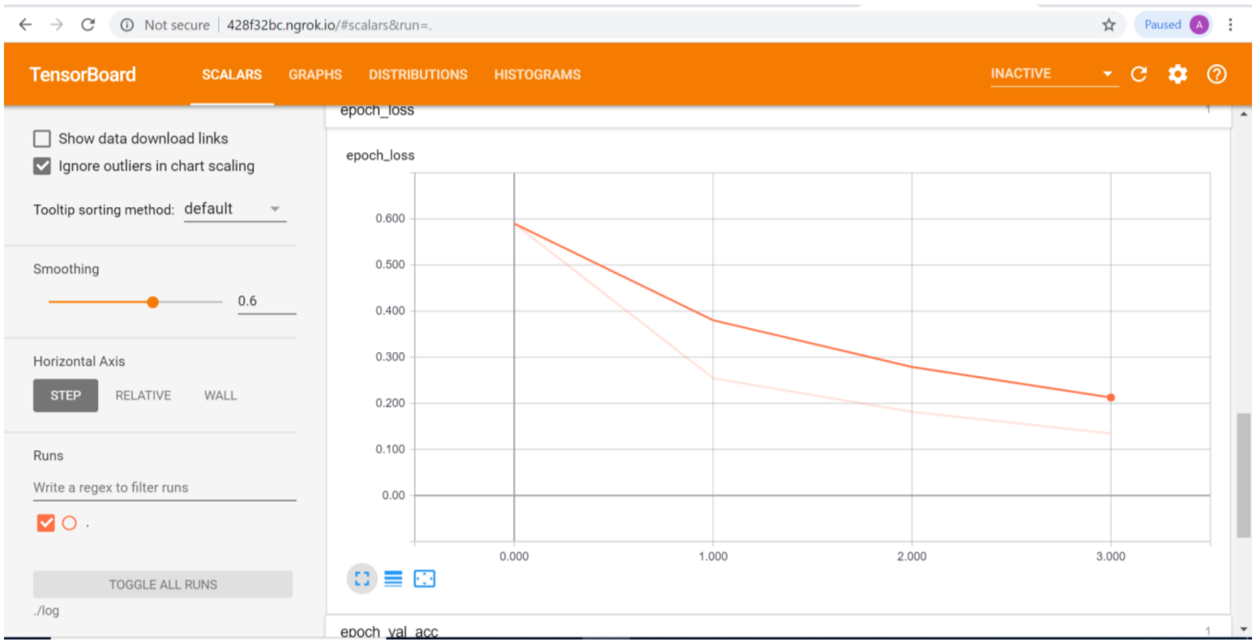
Optimizer: Adam

Screenshots:

Training Accuracy



Training Loss



Not secure | 428f32bc.ngrok.io/#scalars&run=.

- [ ] Show data download links
- [x] Ignore outliers in chart scaling

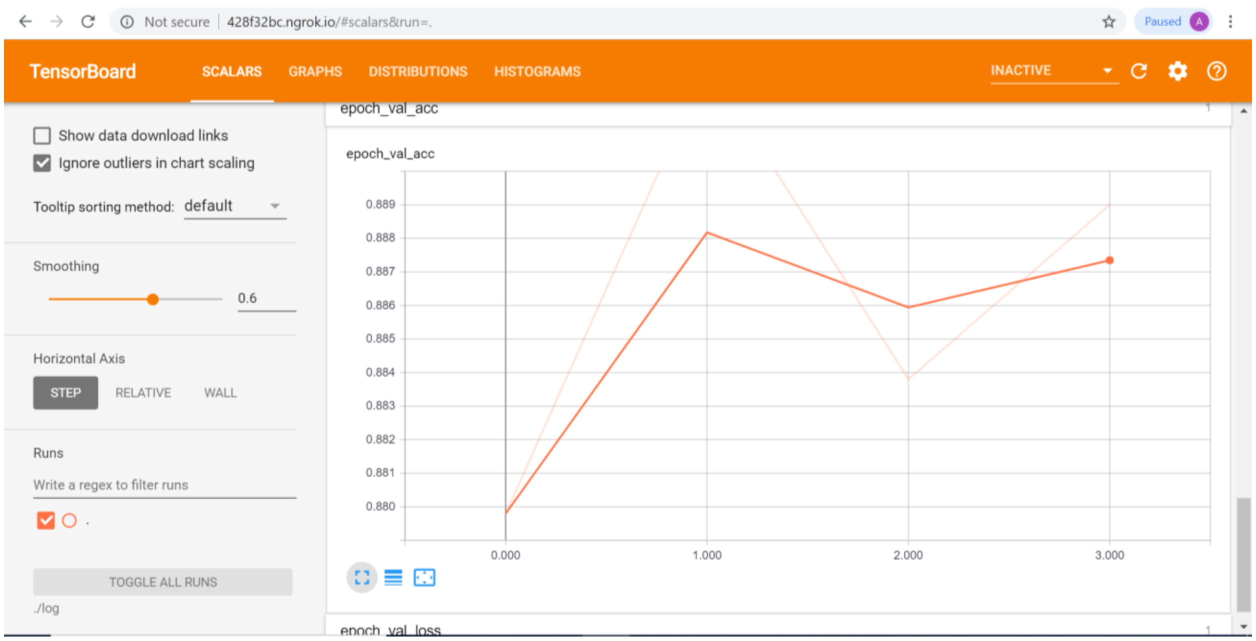
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP

Runs: ./log

Validation Accuracy



Not secure | 428f32bc.ngrok.io/#scalars&run=.

- [ ] Show data download links
- [x] Ignore outliers in chart scaling

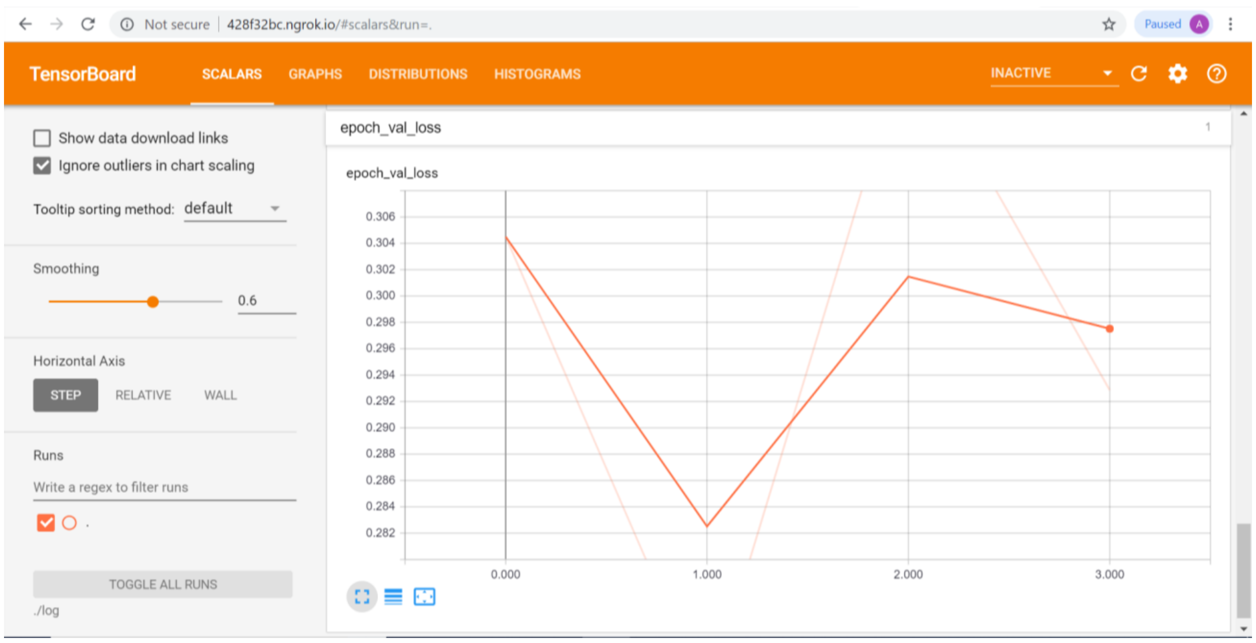
Tooltip sorting method: default

Smoothing 0.6

Horizontal Axis: STEP

Runs: ./log

Validation Loss



Test Accuracy

```
25000/25000 [==============================] - 498s 20ms/step
[0.32112948688983917, 0.8807599999999999]
```

Approach

1) Firstly, we started with a normal LSTM layer just to see what the integration looks like. We faced few errors in this with respect to the shape of the output being fed to the dense layer. Later we found the parameter "return_sequences = True" to be set in the LSTM layer. This solved our error of shape.
2) We had put the LSTM input units to be 8 and noticed that it just gave us the accuracy of 54%.
3) Then we read about what the parameter "input units" means in LSTM. So basically, if we have a review let say 256 words and we just put 8 inputs units in LSTM, we are discarding the remaining 248 words from the review. This resulted in loss of information.
4) We then increased the input units to 512 and saw the test accuracy shot to 83% with a good validation accuracy of about 85%.
5) Later on, we thought of preprocessing the data well by including the concept of bi-gram. But we were not sure how the embeddings work and how 2 words will be represented as vector. Moreover, this will increase the number of words (unigram + bigram) which means we need to increase the input units of LSTM. Thus, more computation time. Hence, we dropped that idea.

6) We then looked at the number of words for each review and noticed the mean length was around 500 words. We took the standard deviation of the number of words per review and found that only 4% of reviews were over 900 words. Thus, we changed our padding length (length of each word after padding) to 900 words.
7) We added one more layer of LSTM and increased the dense layer as well. We noticed that after few epochs, the validation accuracy was decreasing. That means the model was overfitting the data. To overcome that, we included dropout parameter in LSTM.
8) We were able to achieve the maximum accuracy of 85% with this architecture which was just near the architecture with just dense layers.
9) Thus, instead of LSTM we tried the same architecture including GRU. We achieved the accuracy of 87% and decided to stick with GRU instead of LSTM.
10) We did not go with the RNN because it has the problem of vanishing gradients.
11) We then thought of having the already trained embeddings (word2vec). We were not able to understand exactly how we should include into our code. Because we had indexes representing the words and we had to get the whole matrix representing word2vec which needs to be fed to the LSTM/GRU layer.
12) Later we thought of implementing stopwords. But the review like "It is not a good movie" gives us "good movie" after implementing stopwords. Thus, we discarded this approach.
13) Then later we thought of boosting the weights for word like 'good', 'brilliant', 'bad' and many more related to movie review. But this we were not able to identify how to feed into the LSTM/GRU layers.
14) We also changed the optimizer to SGD, but did not have any difference in comparison to Adam.
15) We changed the batch size too. This was having computation issue and showed us the more ETA to run an epoch.
16) We then included regularizers in GRUs, increased the dropout factor but did not give more efficient output.
17) We increased the vocab size to 90000 because total number of indexes in the review is 88000.
18) Then we kept on changing the parameters of GRU and increased the dense layer and checked the validation accuracy and loss.
19) We also included the concept of earlyStopping and patience parameter in our model. This would stop training the model once the validation loss drop below a previous epoch.