COMP 551 Comparison of Different Text Classification Models

Negar Hassantabar - 260939318

Fall 2019

1 RESULTS

In this project, variety type of text classification models has implemented and evaluated. The following table represents the best result for each model, as it is shown in table 1, the best results obtained from Logistic Regression model with implementing the ***TfidfTransformer()*** for preprocessing input vector.

Model	Best Validation Accuracy obtained	Rank
Logistic Regression	54.27	1
LSTM	51.09	2
Simple RNN	21.45	6
GRU	49.91	3
CuDNNGRU	48.07	5
CuDNNLSTM	48.56	4

Table 1 – Best model's performance

According to the table one, The LSTM model and GRU model has provided almost the same validation accuracy and Simple RNN model provided poor performance for this text classification task.

1-1 LOGESTIC REGRESION

As a linear model the Logistic Regression model has been implemented and the impact of ***N_garm, Stop_words, Tf-idf.*** has been evaluated. The results of the implemented models is provided in table 2; the best result has been obtained from model with ***TfidfTransformer()*** , customized ***Stop_Words*** for ***CountVectorizer()*** and 1 ***N_gram.*** The details are provided in table 2.

Model	Parameters	Accuracy	Rank
LR1	Tfidf = False	50.19	4
	Stop word = Customized		
	N_gram = (1,1)		
LR2	Tfidf = False	50.38	3
	Stop word = English		
	N_gram = (1,1)		
LR3	Tfidf = True	54.26	2
	Stop word = English		
	N_gram = (1,1)		
LR4	Tfidf = true	54.27	1
	Stop word = Customized		
	N_gram = (1,1)		

Table 2 – Logistic Regression Models Results

According to table 2, the ***TfidfTransfo-rmer()*** has a huge impact on the validation accuracy of Logistic Regression model and improves it by 6 percent. However, the customized ***Stop_words***, which contained most frequent words in comments, did not influence the validation accuracy as much as other hyper parameters.

1-1-1 N_gram

N_gram is one of the model that influences the number of features remarkably, however, its impact on validation accuracy depends on the nature of the data. Hence, for the Reddit comments dataset the range of (1, 1) to (1, 4) N_garm is implemented and the following result is obtained and provided in table 3.

Model	N_gram	Accuracy
Tfidf = True, Stop_Word = Customized	(1, 1)	54.27
	(1, 2)	54.26
	(1, 3)	53.63
	(1, 4)	53.36

Table 3 – N_gram Influence

According to table 3, increasing N_gram doesn't lead to a better validation accuracy.

1-2 LSTM

In the LSTM implementation, three different main models has been evaluated and the range of validation accuracy derived from LSTM models is below the accuracy of Logistic Regression. The details of each implemented models is provided as the Appendix 1. In a LSTM layer the input data format is different from Logistic Regression or any other previously implemented models. The Reddit comment's words needed to be transferred to non-sparse numerical vectors thus, instead of counting the repetition of a word or appearance a word, each number represents a unique word. To do this task, first of all, a text preprocessing included eliminating the symbols, stop words, and space between words has been done. Also, ***PorterStemmer()*** and ***WordNetLemmatizer()*** were added for stemming and lemmatisation. Then, the ***nltk.tokenize.word_tokenize()*** used to provide vectors with separated word as their object and ***keras.preprocessing.text.one_hot()*** used to transfer each unique word to a number. For equalizing the input shape of all vector length comments, ***keras.preprocessing. sequence.pad_sequences()*** was used and the length has chosen according to distribution of length of comments and obtained results. According to the following table, vector length was changed from small number of words to the higher numbers and vector length equal to 40 has been provided best result among all implemented models. Higher vector lengths were unable to predict the desired class and predicting the same class for most of the comments.

Model	Vector Lenght	Accuracy	Rank
3	8	0.0500	5
4	10	0.3000	4
6	20	0.3870	3
7	35	0.4000	2
17	40	0.4653	1
29	66	0.0500	5
30	100	0.0500	5

Table 4 – Vector length Vs Accuracy

1-2-1 BEST RESULT

According to detailed tables in Appendix 1, the best result for LSTM model structure it's obtained from One-LSTM-Layer-Structure model that is shown in table 5.

Model Number	31
Vector Length	40
Hidden Size	512
Epoch	7
Number of LSTM layers	1
Activation Function	Relu
Drop out	0.4
Classification Function	Softmax
Learning Rate	0.00085
Batch size	200
Fit Shuffle	False
Validation Accuracy (%)	51.09
Training Accuracy (%)	72.00

Table 5 – Model 31 parameters

Also for this model 90 percent of data was allocated for training set and 10 percent for validation set. This model provides the best validation accuracy and also the least overfitting among all LSTM models that have been implemented.

1-2-2 Lemmatisation, Stemming and stop Words

The input format and parameters has the same importance for deep learning models, thus, after finding the best model parameters, input data parameters have been evaluated. For this task, the data set has been split; Shuffle was set as ***False*** to evaluating models on the same validation and training dataset. As mentioned in table 5, the best model implemented with stemming and lemmatisation, and eliminating stop words.

Models	Accuracy (%)	Epoch
With Stemming, lemmatisation and Stop Words	49.34	6
	49.93	6
Without Preprocessing	50.99	4
	50.36	6
With Stemming and Lemmatisation	51.09	7

Table 6 – preprocessing parameters

According to table 6, the stop words has negative effect on validation accuracy. One of the possible reasons could be reducing the number of unique words from 68,205 words to 61,769 words and disrupting the relation among words in a sentence with eliminating some words. However, Stemming and Lemmatisation has improved the validation accuracy and reduced the number of unique words to 61,784 words but do not disrupt the sequence of the comments.

1-2-3 Learning Rate

In this part, the influence of Learning Rate has been evaluated in order to get best result on models. Hence, different Learning Rate have been implemented on each LSTM model. A sample result is shown in table 7 and the complete result is given in Appendix 1.

Learning Rate	Accuracy	Epoch	Rank
0.00070	49.6	6	3
0.00075	50.6	6	1
0.00078	50.3	5	2
0.00080	48.69	8	4

Table 7 – Learning Rate influence on Accuracy

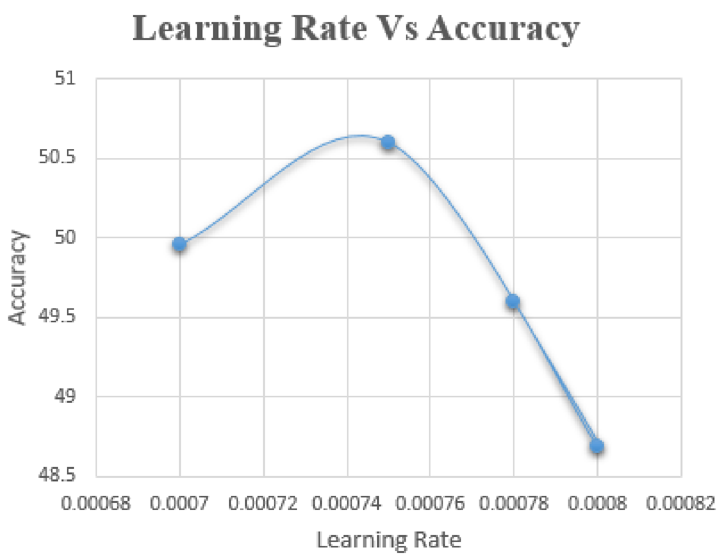


Figure 1 – Learning Rate Vs Accuracy

According to table 6 and figure 1, there is no linear relation between Learning Rate and Accuracy and also according to Appendix 1 for each model there is a different relation between these parameters.

All the implemented models were showing a considerable overfitting and although we obtained a high accuracy on the training dataset, we were unable to obtain a good prediction on the validation dataset. (51.09%). Figure 2 is provided as a sample showing how our models behaved. The high resolution of figure 2 is provided as Appendix 3.

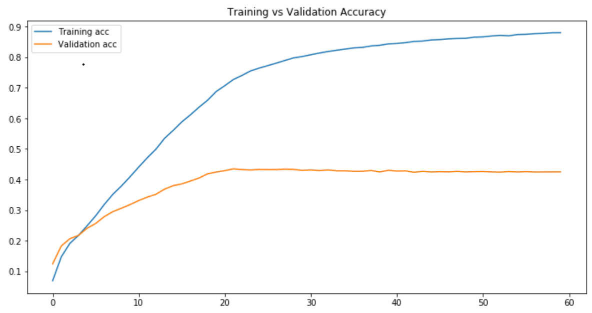


Figure 2 – A sample of our model behavior

1-3Simple RNN

According to table 8, the simple RNN model has poor performance in comparison with other implemented Recurrent Neural Network (RNN) algorithms. As you see in figure 3, the Simple RNN provides a slow learning rate on the training set, and also the curve of training and validation accuracy has a huge rate of noise. The high resolution of figure 3 provided as Appendix 3

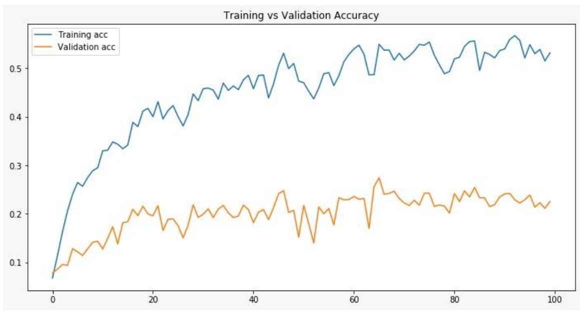


Figure 3 – Simple RNN Accuracy Curve

The best result for SimpleRNN model obtained from model 5 with parameters which is provided in table 8.

Model Number	5
Vector Length	40
Hidden Size	512
Epoch	100
Number of LSTM layers	1
Activation Function	Relu
Drop out	0.4
Classification Function	Softmax
Learning Rate	0.00080
Batch size	200
Fit Shuffle	False
Validation Accuracy (%)	21.45
Training Accuracy (%)	53.29

Table 8 – SimpleRNN parameters

According to table 8, the SimpleRNN model provides the least accuracy and also the largest overfitting gap.

1-4 GRU

GRU is older version of LSTM Network, so, GRU models has been implemented and evaluated based on the model's parameters which has best performance on LSTM model. Then, hyper parameters has been evaluated and best performance was obtained from the model with parameters that is provided in table 9.

Model Number	3
Vector Length	40
Hidden size	512
Epoch	5
Activation Function	Relu
Dropout	0.6
Classification	Softmax
Learning Rate	0.001
Batch Size	200
Fit Shuffle	False
Validation Accuracy	49.91
Training Accuracy	65.45

Table 9 – Model 3 Parameters

The results for all implemented models is provided in Appendix 2. The main parameters that has been evaluated in this project for GRU models are Learning Rate and Drop out.

Model	Dropout	Learning Rate	Accuracy	Rank
1	0.2	0.0010	49.30	5
2	0.4	0.0010	49.74	3
3	0.6	0.0010	49.91	1
4	0.6	0.0009	49.87	2
5	0.6	0.0008	49.69	4
6	0.4	0.0008	48.94	6

Table 10 – GRU Models Result

1-5 CuDNNGRU

Another implemented neural network algorithm is CuDNNRU which is not an RNNcell and known as a faster version of GRU. The performance of the CuDNNGRU models seems to be almost in the same range as GRU. The model was implemented with different range of Learning Rate and other hyper parameters were set based on the best GRU model. The best obtained results is provided in table 11. According to the results, higher Learning Rate leads to the better performance.

Model	Learning Rate	Accuracy	Rank
1	0.0007	47.94	2
2	0.0006	47.79	3
3	0.0010	48.07	1

Table 11 – CuDNNGRU Results

According to table 11, the best result of the model is almost 2 percent lower than normal GRU model.

1-6 CuDNNLSTM

CuDNNLSTM is also known as the faster version of LSTM same as CuDNNGRU. The CuDNNLSTM model was implemented based on the hyper parameters of the best LSTM model obtained and Learning Rate was used as a Changeable parameter to obtain the best result. The results of all implementations is provided in table 12.

Model	Learning Rate	Accuracy	Rank
1	0.0085	48.31	2
2	0.0080	48.06	3
3	0.0010	48.56	1

Table 12 – CuDNNLSTM Results

According to tables 11 and 12, the faster versions of both models (CuDNNLSTM and CuDNNGRU) perform in the same way of their base models and with CuDNNLSTM we obtained better performance in comparison with CuDNNGRU. The both models are shown 4 percents decrease in validation accuracy; however, the hyper parameters are set almost as same as their base models as it is shown in table 13.

Model	Learning Rate	CuDNN	Base
LSTM	0.00085	48.56	51.09
GRU	0.00100	48.07	49.91

Table 13 – CuDNN Vs Base models Accuracy

Appendix 1

The detailed parameters table of all 1 LSTM layer models

Model Number	Vector Length	Hidden Size	Epoch	Number of Layer	Activation Function layer 1	Drop out layer 1	Classification Function	Learning Rate	Batch Size	Fit Shuffle	Validation Accuracy	Training Accuracy	Comment
22	40	512	30	1	Relu	0.4	Softmax	0.001	200	FALSE	0.4894	0.6572	
23	40	512	6	1	Relu	0.4	Softmax	0.001	200	FALSE	0.4966	0.7200	Recurrent Dropout 0.4
24	40	512	15	1	Relu	0.4	Softmax	0.001	200	FALSE	0.4469	0.9521	Embedding (mask_zero = True)
25	40	512	15	1	Relu	0.25	Softmax	0.001	200	FALSE	0.4842	0.7000	add 2 Dropout Layer with 0.25
26	40	512	15	1	Relu	0.4	Softmax	0.001	200	FALSE	0.4879	0.6600	add 2 Dropout Layer with 0.26
27	40	512	15	1	Relu	0.4	Softmax	0.001	200	FALSE	0.4933	0.6952	add 2 Dropout Layer with 0.3
28	40	512	15	1	Relu	0.4	Softmax	0.0001	200	FALSE	0.4800	0.6700	add 2 Dropout Layer with 0.4
29	40	512	5	1	Relu	0.4	Softmax	0.001	200	FALSE	0.5080	0.6800	Allocated more Data for training (0.9 Traing , 0.1 Validation)
30	40	512	7	1	Relu	0.4	Softmax	0.001	200	FALSE	0.5060	0.7584	Allocated more Data for training (0.95 Traing , 0.05 Validation)
31*	40	512	7	1	Relu	0.4	Softmax	0.00085	200	FALSE	0.5109	0.7200	Allocated more Data for training (0.9 Traing , 0.1 Validation)
32	40	512	7	1	Relu	0.4	Softmax	0.00080	200	FALSE	0.4966	0.6470	Allocated more Data for training (0.9 Traing , 0.1 Validation)

*The best model is highlighted in yellow

The detailed parameters table of all 2 LSTM layer models

Model Number	Vector Length	Hidden Size	Epoch	Number of Layer	Activation Function layer 1	Drop out layer 1	Activation Function layer 2	Drop out layer 2	Classification Function	Learning Rate	Batch Size	Fit Shuffle	Validation Accuracy	Training Accuracy	Comment
1	40	64	20	2	Relu	0.2	Relu	0.4	Sigmoid	0.01	100	TRUE	0.0500	0.0600	
3	8	128	10	2	Relu	0.15	Relu	0.15	Sigmoid	0.001	200	FALSE	0.0500	0.0500	Add (0.0001) to X_train and X_test
4	10	256	20	2	Relu	0.15	Relu	0.15	Sigmoid	0.001	200	FALSE	0.3000	0.8130	Add (1) to y_train and y_test
5	10	64	20	2	Relu	0.3	Relu	0.3	Sigmoid	0.001	200	FALSE	0.3060	0.8180	
6	20	64	20	2	Relu	0.3	Relu	0.3	Sigmoid	0.001	200	FALSE	0.3870	0.7850	
7	35	64	30	2	Relu	0.3	Relu	0.3	Sigmoid	0.001	200	FALSE	0.4000	0.6990	
8	40	64	30	2	Relu	0.3	Relu	0.3	Sigmoid	0.001	200	FALSE	0.3800	0.5620	
9	40	64	100	2	Relu	1.3	Relu	1.3	Sigmoid	0.0015	200	FALSE	0.4390	0.7900	Dropped to 0.422
10	40	64	30	2	Relu	1.3	Relu	1.3	Sigmoid	0.0015	200	FALSE	0.4334	0.8360	
15	40	128	35	2	Relu	0.3	Relu	0.4	Softmax	0.0015	200	FALSE	0.4559	0.8000	
16	40	200	35	2	Relu	0.3	Relu	0.4	Softmax	0.0015	200	FALSE	0.4400	0.9000	
17	40	512	30	2	Relu	0.3	Relu	0.4	Softmax	0.0015	200	FALSE	0.4653	0.7100	
19	40	512	30	2	Relu	0.4	Relu	0.4	Softmax	0.0015	200	FALSE	0.4600	0.9400	
20	50	512	30	2	Relu	0.4	Relu	0.4	Softmax	0.0015	200	FALSE	0.4580	0.9200	
21	40	512	30	2	Relu	0.4	Relu	0.4	Softmax	0.0001	201	FALSE	0.4255	0.7450	

The detailed parameters table of all 3 LSTM layer models

Model Number	Vector Length	Hidden Size	Epoch	Number of Layer	Activation Function Layer 1	Drop out Layer 1	Activation Function Layer 2	Drop out Layer 2	Activation Function Layer 3	Drop out Layer 3	Classification Function	Learning Rate	Batch Size	Fit Shuffle	Validation Accuracy	Training Accuracy	Comment
2	8	128	50	3	Relu	0.2	Tanh	0.2	Relu	0.2	Sigmoid	0.001	200	TRUE	0.0600	0.0500	
11	40	32	30	3	Relu	0.3	Relu	0.3	Relu	0.3	Sigmoid	0.0015	200	FALSE	0.2500	0.6200	
12	30	32	30	3	Relu	0.15	Tanh	0.15	Relu	0.15	Sigmoid	0.0015	200	FALSE	0.3860	0.7390	
13	30	32	30	3	Relu	0.15	Tanh	0.15	Relu	0.2	Sigmoid	0.0001	200	FALSE	0.2250	0.3180	Both Rising
14	40	64	50	3	Relu	0.3	Tanh	0.3	Relu	0.4	Softmax	0.0015	200	FALSE	0.4230	0.8580	

Appendix 2

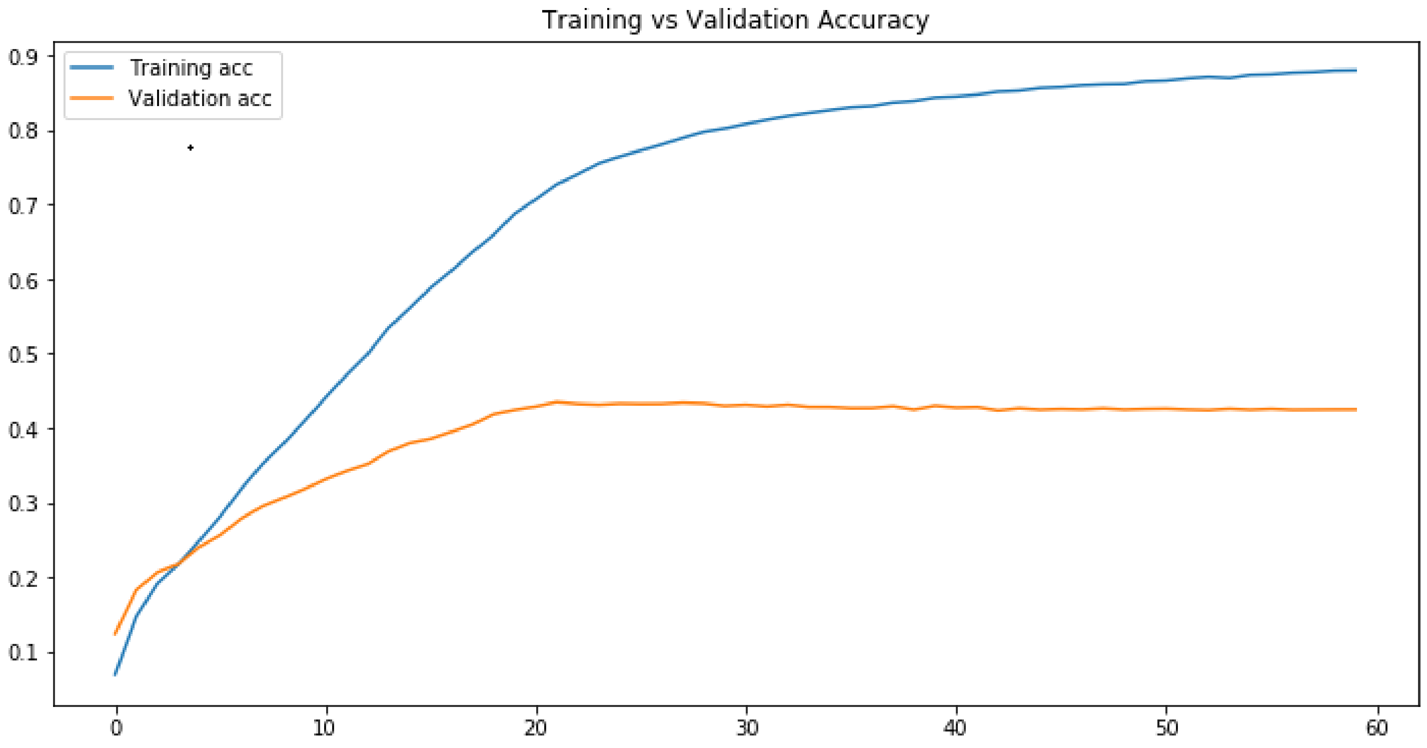
The detailed parameters table of all GRU models

Model Number	Vector Length	Hidden Size	Epoch	Number of Layer	Activation Function Layer	Drop out Layer	Classification Function	Learning Rate	Batch Size	Fit Shuffle	Validation Accuracy	Training Accuracy	Comment
1	40	512	6	1	Relu	0.2	Softmax	0.0010	200	TRUE	0.4930	0.7300	
2	40	512	6	1	Relu	0.4	Softmax	0.0010	200	FALSE	0.4974	0.6340	
3*	40	512	5	1	Relu	0.60	Softmax	0.0010	200	FALSE	0.4991	0.6545	
4	40	512	4	1	Relu	0.60	Softmax	0.0009	200	FALSE	0.4987	0.5704	
5	40	512	8	1	Relu	0.60	Softmax	0.0008	200	FALSE	0.4969	0.7448	
6	40	512	6	1	Relu	0.4	Softmax	0.0008	200	FALSE	0.4894	0.7000	

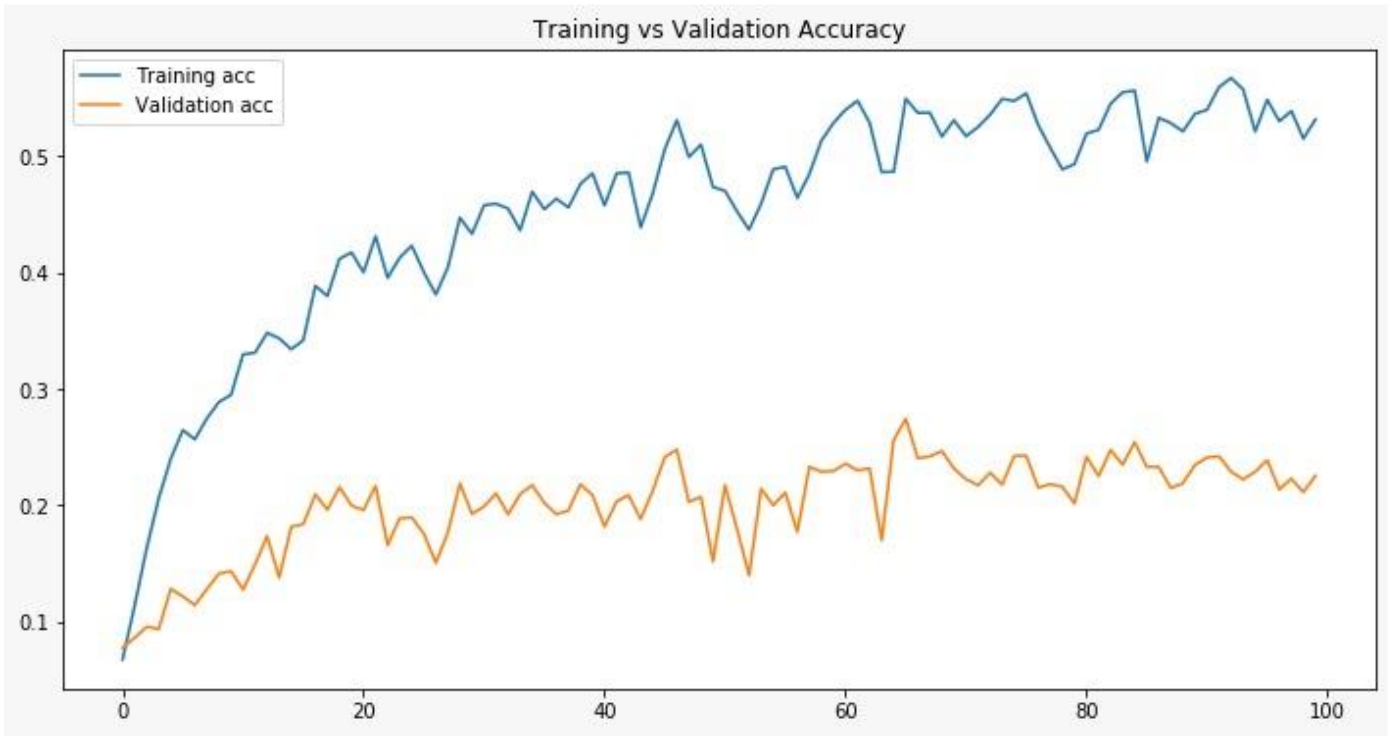
*The best model is highlighted in yellow

Appendix 3

A sample of our models behavior



Simple RNN Accuracy Curve


Training vs Validation Accuracy
Training acc
Validation acc