

**CE4042 Neural Networks and Deep Learning**

**Project Report 2**

**by**

**Lai Qing Hui, U1622339G**

**Ron Ng Jian Ying, U1622393B**

SCHOOL OF COMPUTER SCIENCE & ENGINEERING

2019

Contents

[Part B: Text Classification 3](#_Toc23774575)

[Character CNN Classifier 3](#_Toc23774576)

[Word CNN Classifier 5](#_Toc23774577)

[Embedding Layer vs One Hot Encoding 7](#_Toc23774578)

[Character RNN Classifier 8](#_Toc23774579)

[Word RNN Classifier 9](#_Toc23774580)

[Comparison of Accuracy and Running Times with/without dropouts 10](#_Toc23774581)

[Character CNN Classifier 10](#_Toc23774582)

[Training Loss for dropouts 0.1 to 1.0 10](#_Toc23774583)

[Test Accuracy for dropouts 0.1 to 1.0 11](#_Toc23774584)

[Word CNN Classifier 12](#_Toc23774585)

[Training Loss for dropouts 0.1 to 1.0 12](#_Toc23774586)

[Test Accuracy for dropouts 0.1 to 1.0 14](#_Toc23774587)

[Character RNN Classifier 14](#_Toc23774588)

[Training Loss for dropouts 0.1 to 1.0 14](#_Toc23774589)

[Test Accuracy for dropouts 0.1 to 1.0 16](#_Toc23774590)

[Word RNN Classifier 17](#_Toc23774591)

[Training Loss for dropouts 0.1 to 1.0 17](#_Toc23774592)

[Test Accuracy for dropouts 0.1 to 1.0 18](#_Toc23774593)

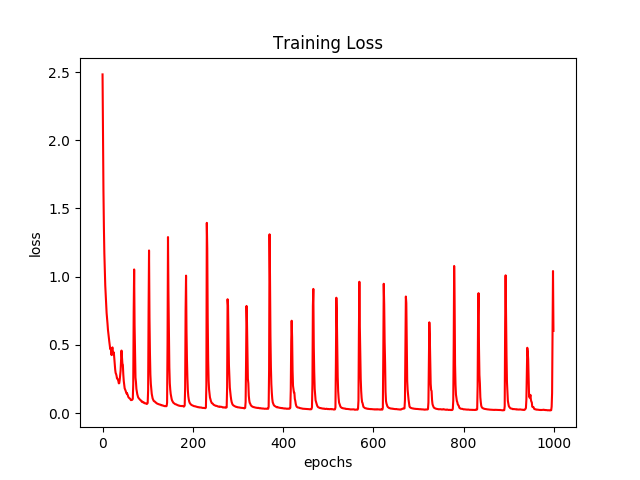
# Part B: Text Classification

## Character CNN Classifier

The character classifier consists of two convolution layers. The table below shows the layers and properties of the CNN.

Table 1: Character CNN Layers and Settings

|  |  |  |
| --- | --- | --- |
|  | C1 | C2 |
| Filter Number | 10 | 10 |
| Window Size | 20x256 | 20x1 |
| Padding | VALID | VALID |
| Activation | ReLU | ReLU |
| Pooling | Max, 4x4 | Max, 4x4 |
| Stride | 2 | 2 |
| Stride Padding | SAME | SAME |
| Learning rate | 0.01 | |
| Batch size | 128 | |





The two figures shown above are the **Training Loss** and **Test Accuracy** plots of the Character CNN classifier. The **Training Loss** starts to become noisy at around 70 epochs. The **Test Accuracy** begins to drop after around 40 epochs and fluctuates after that. Evidence of overfitting can be seen – when the training loss decreases, the test accuracy increases and vice versa.

This could be due to hyperparameters being set too high such as the learning rate. The number of weights could affect this as well as does the value of the weights – weights becoming too large in size would cause overfitting as the model memorizes training data instead of learning from it.

Lastly, because the classifier looks only at characters, it might not have enough information to classify the input thereby causing the noise as well.

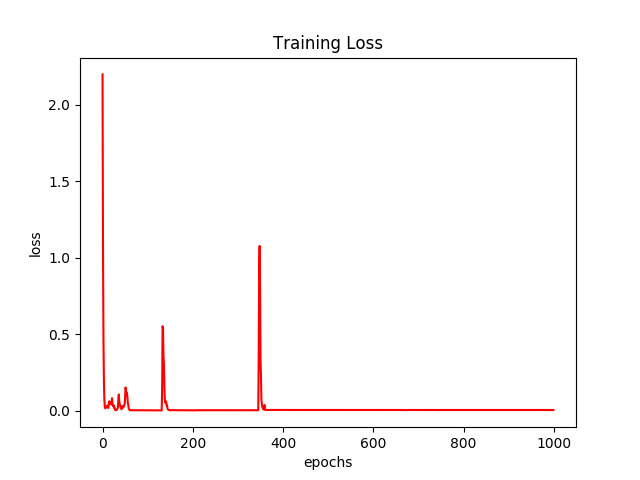
## Word CNN Classifier

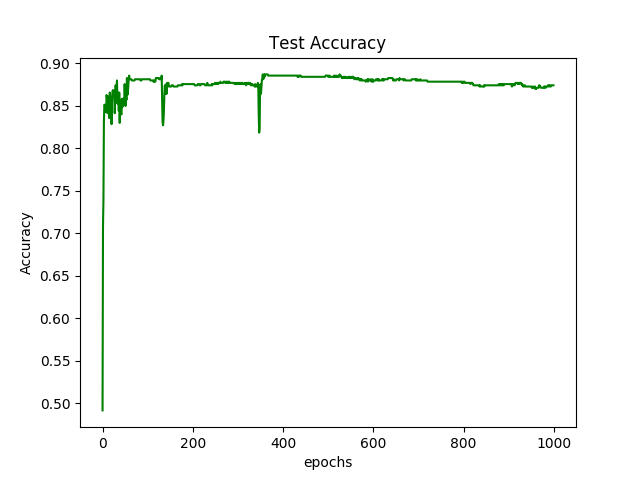
The word classifier consists of two convolution layers. The table below shows the layers and properties of the CNN.

Table 1: Word CNN Layers and Settings

|  |  |  |
| --- | --- | --- |
|  | C1 | C2 |
| Filter Number | 10 | 10 |
| Window Size | 20x20 | 20x1 |
| Padding | VALID | VALID |
| Activation | ReLU | ReLU |
| Pooling | Max, 4x4 | Max, 4x4 |
| Stride | 2 | 2 |
| Stride Padding | SAME | SAME |
| Learning rate | 0.01 | |
| Batch size | 128 | |

In addition, there is an **embedding layer** of size 20 before the input is fed into the CNN. The embedding layer is required as we are working with words instead of characters.





**Training Loss** and **Test Accuracy** is much better compared to the Character classifier with no evidence of overfitting. Note that the hyperparameters were the same in both classifiers.

## Embedding Layer vs One Hot Encoding

One Hot Encoding was used in the Character Classifier. The reason was because we only had 256 possible characters. In this case, One Hot Encoding creates a vector of length 256 filled with ‘0’s and a ‘1’ at the index corresponding to the character. This creates a sparse 256x256 matrix in the end since each character requires one vector to represent.

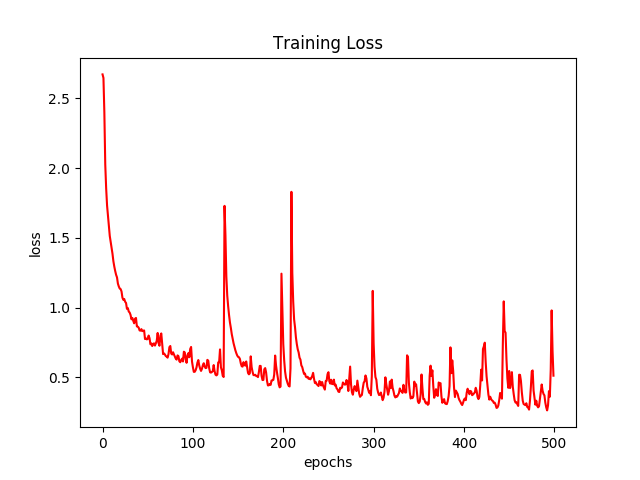
It is appropriate for the Character classifier for two reasons. Firstly, the matrix is small and would result in faster training of the model. Secondly, characters often do not have correlation with each other.

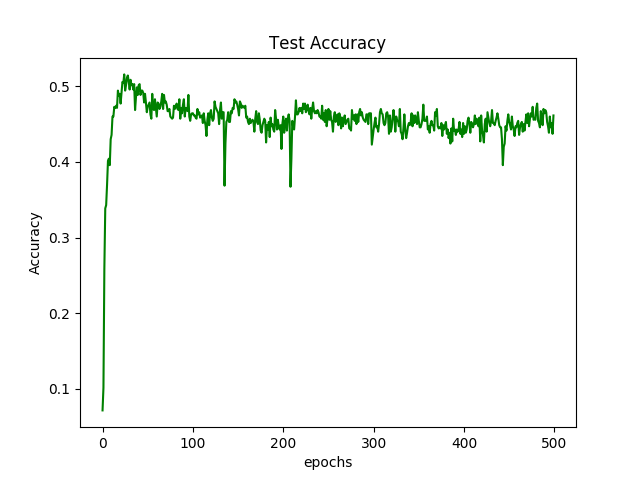
Embedding was used in the Word Classifier due to the bigger amount of data – we are taking whole words now instead of characters and there are much more than 256 unique words in our data set. The next important reason is that Embedding groups co-occurring items together since certain words always come after another. This is how word prediction works when you are typing a message in your phone.

As shown, using Embedding allows for a better model compared to One Hot Encoding in regards to text classification.

## Character RNN Classifier

The Character RNN classifier uses a GRU layer with hidden-layer of size 20

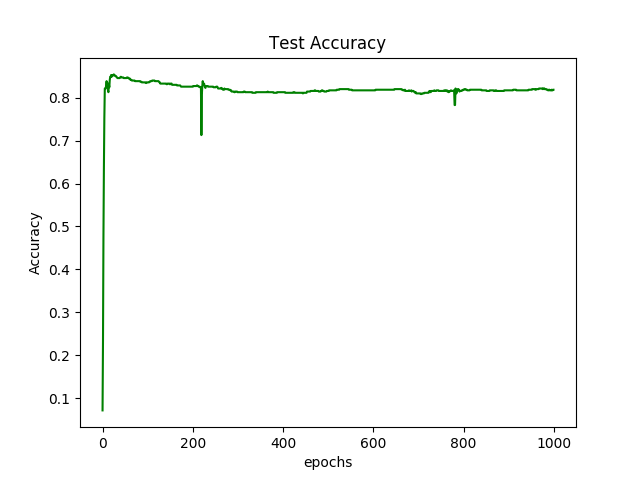


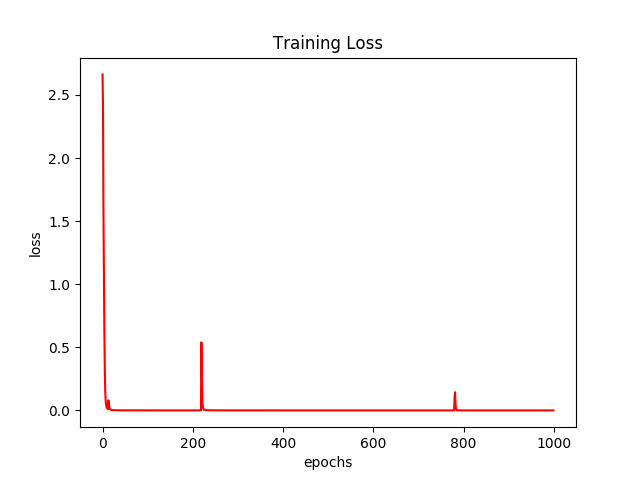


The figures above are the results. We can see that as compared to the Character CNN Classifier, the RNN classifier achieves a 51% accuracy which is about 6% more than the CNN classifier.

## Word RNN Classifier

Using Words instead of Characters for classification is even better with almost no noise in the training and test. The model is trained at around 20 epochs compared to the other models.



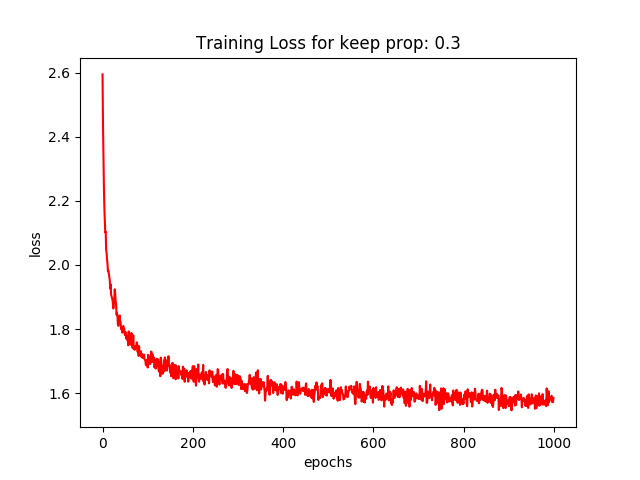
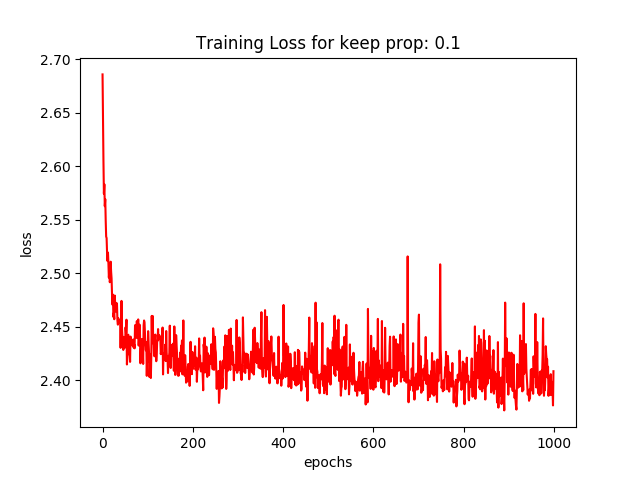


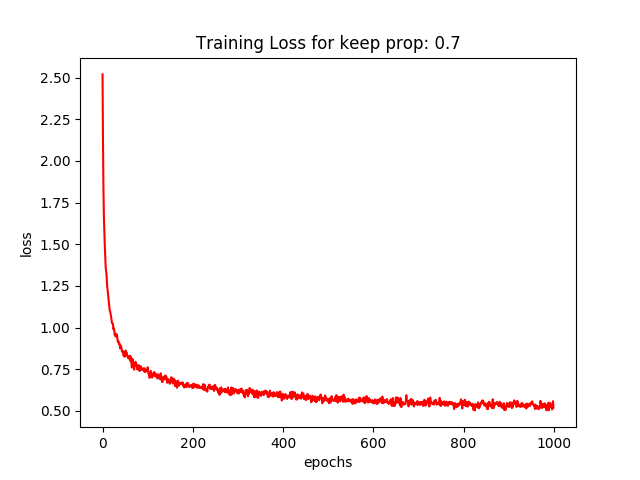
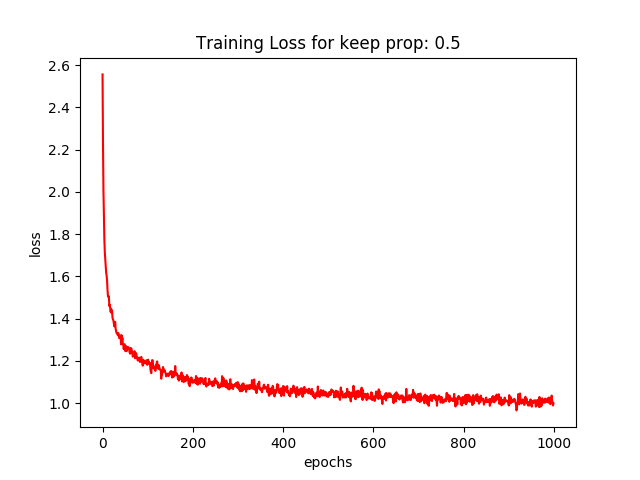
This shows that for this particular dataset, RNN is better compared to CNN for text classification.

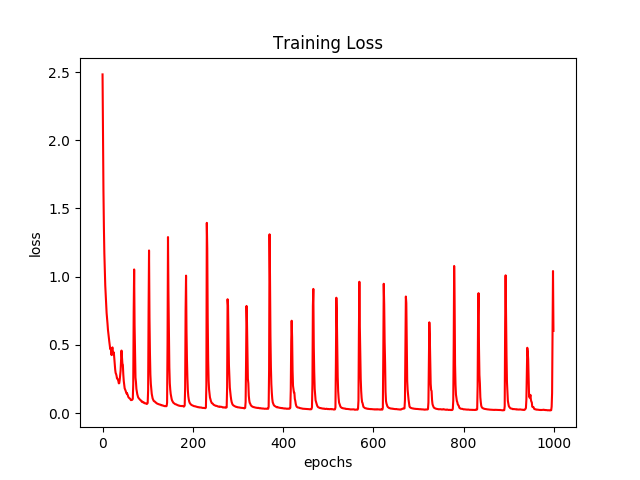
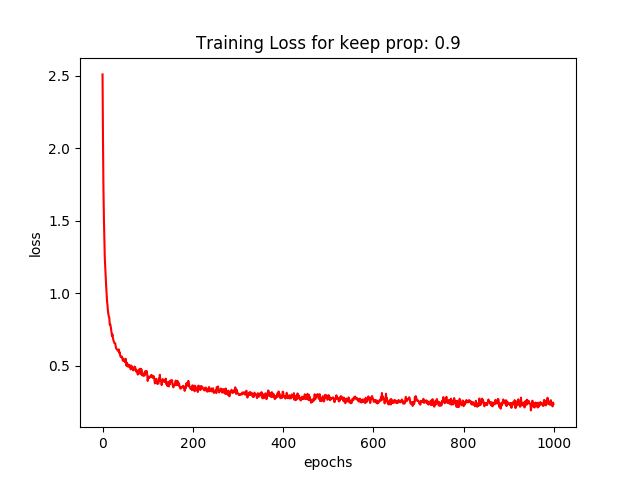
# Comparison of Accuracy and Running Times with/without dropouts

## Character CNN Classifier

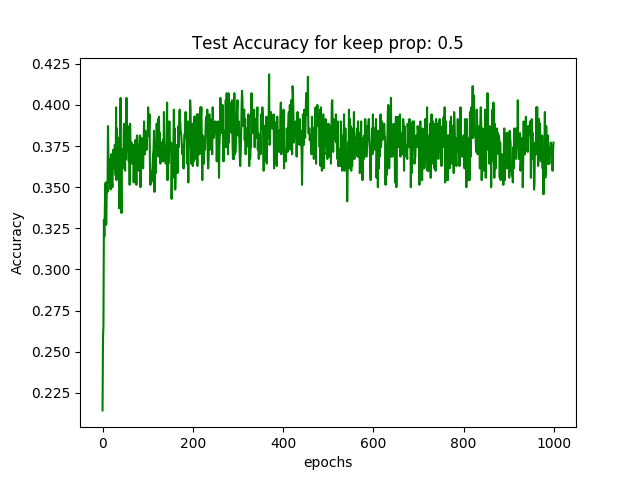
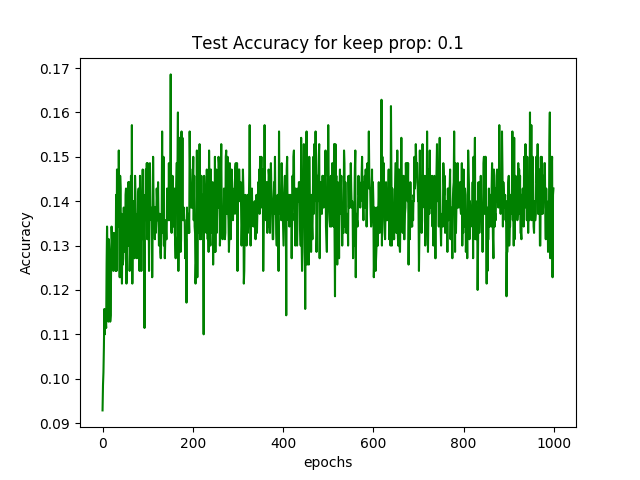
### Training Loss for dropouts 0.1 to 1.0

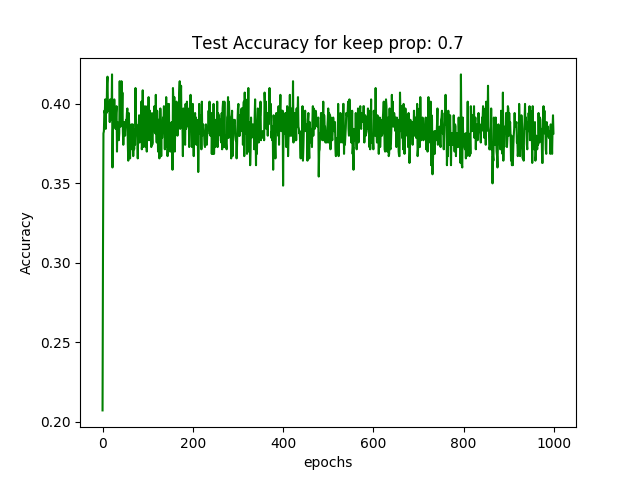
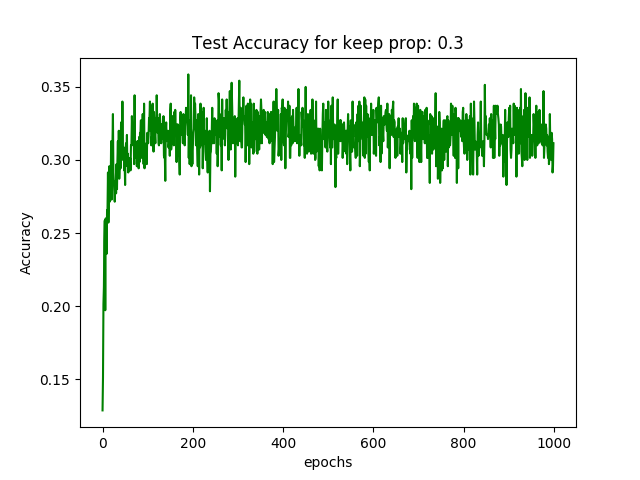


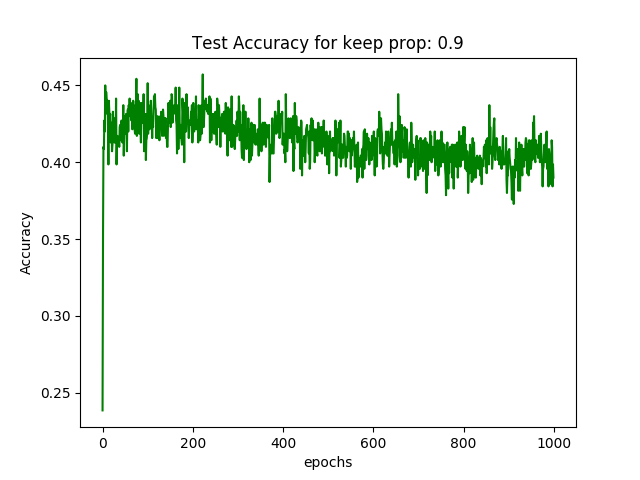




### Test Accuracy for dropouts 0.1 to 1.0



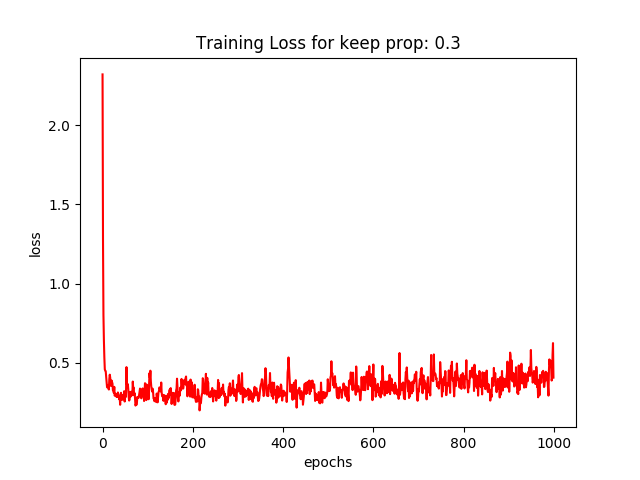
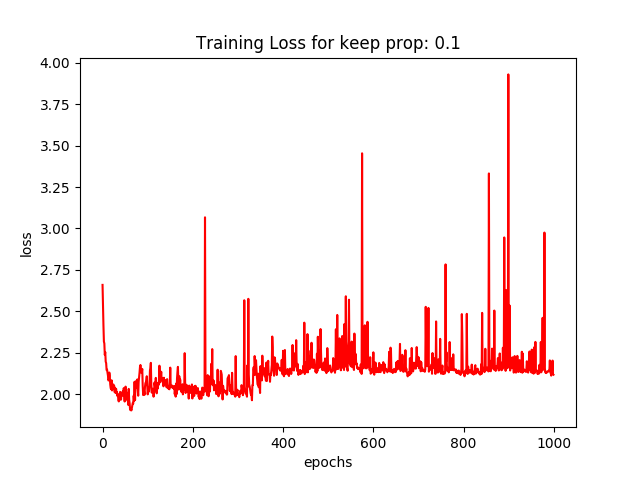


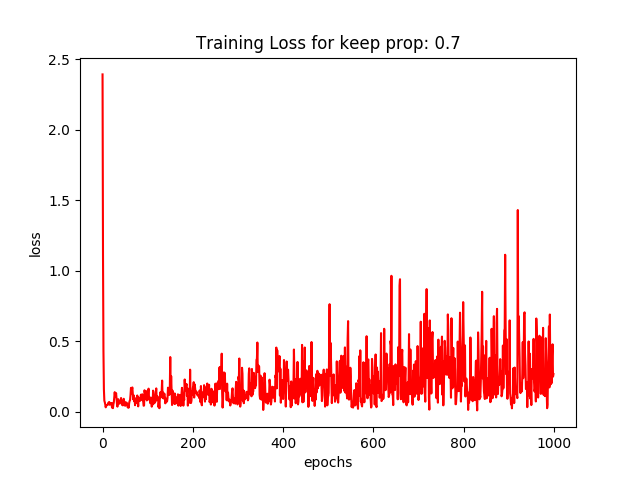
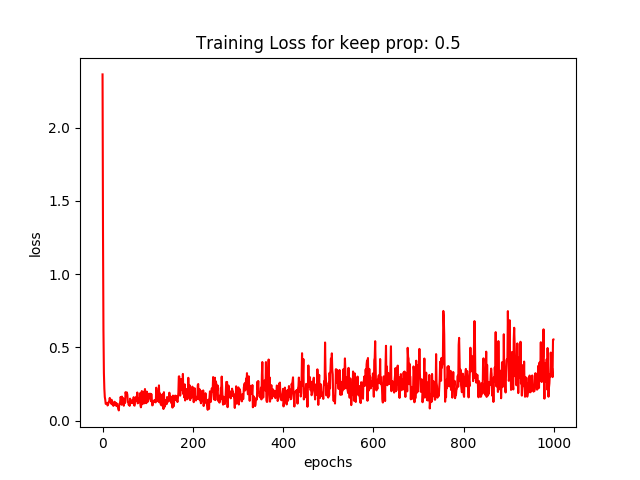


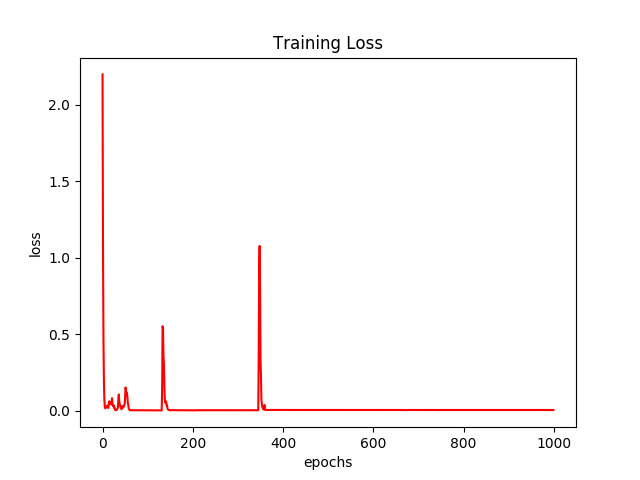
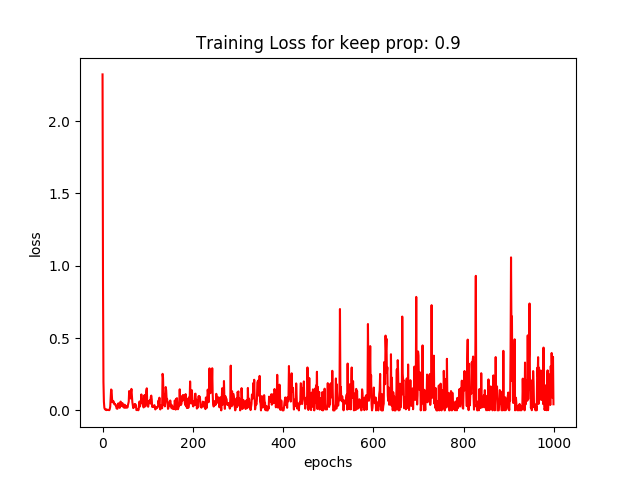
When dropout increases, the training loss gets less noisy except for when the setting was at no drop outs. The most optimal being dropout with keep probability of 90% and early stopping at around 250 epochs.

## Word CNN Classifier

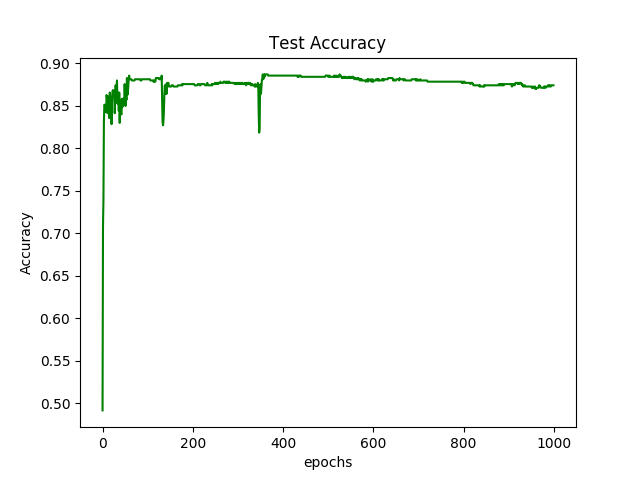
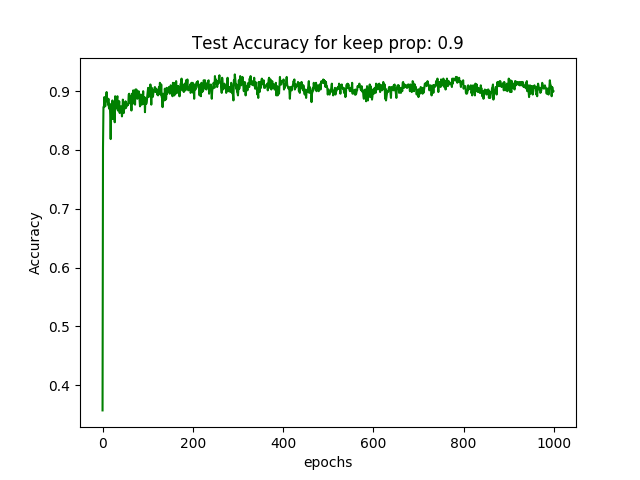
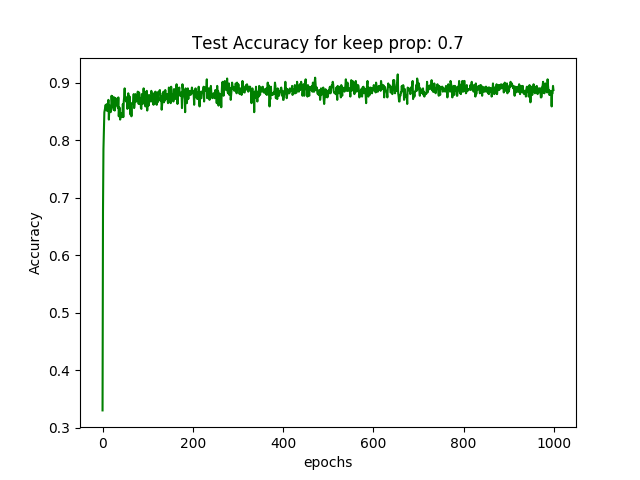
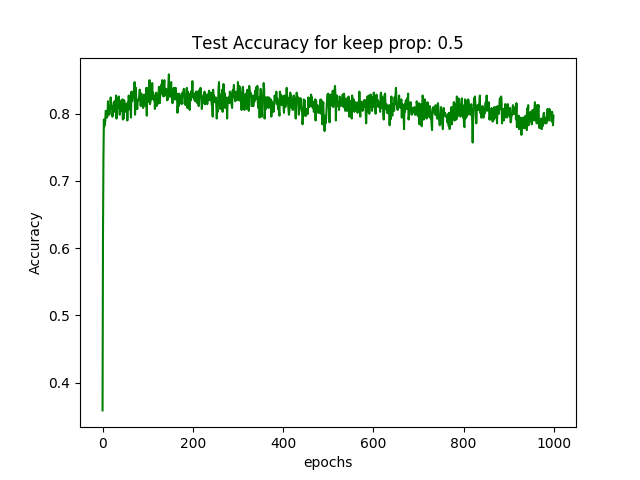
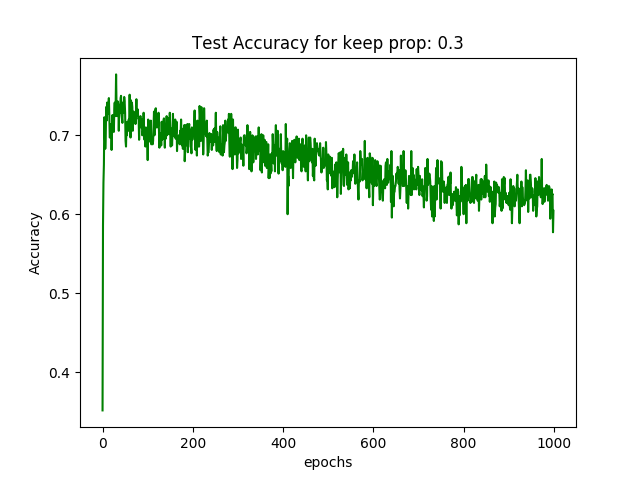
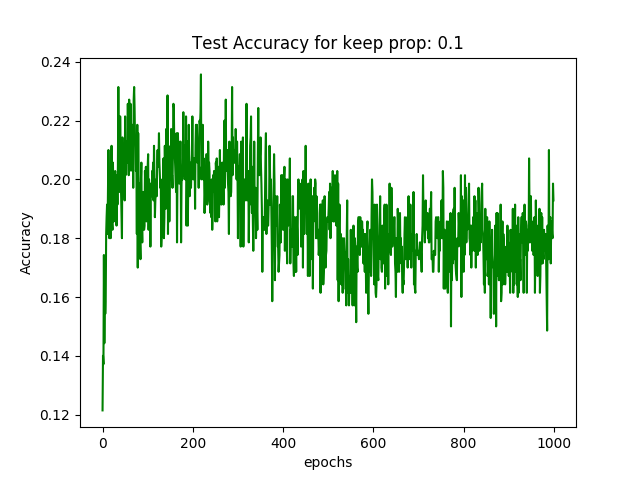
### Training Loss for dropouts 0.1 to 1.0







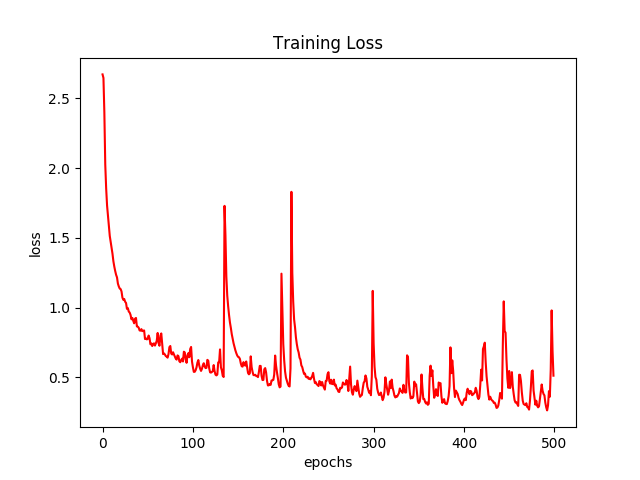
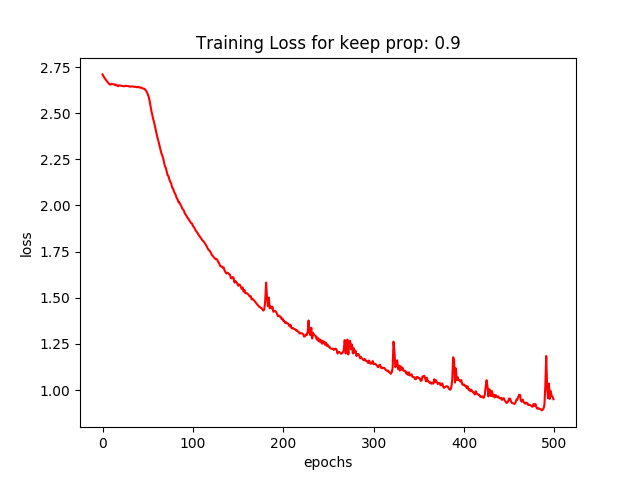
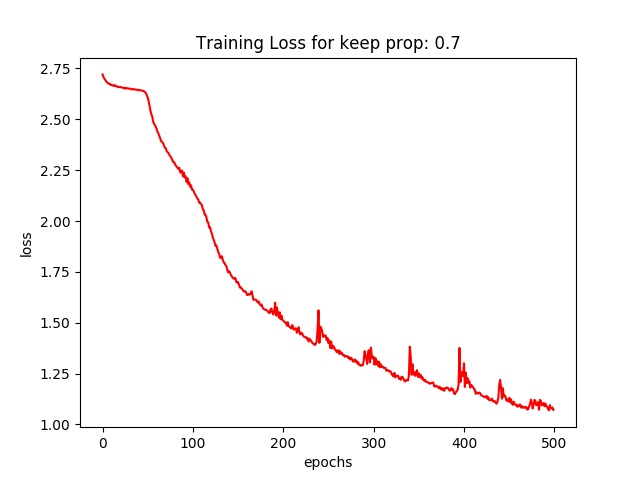
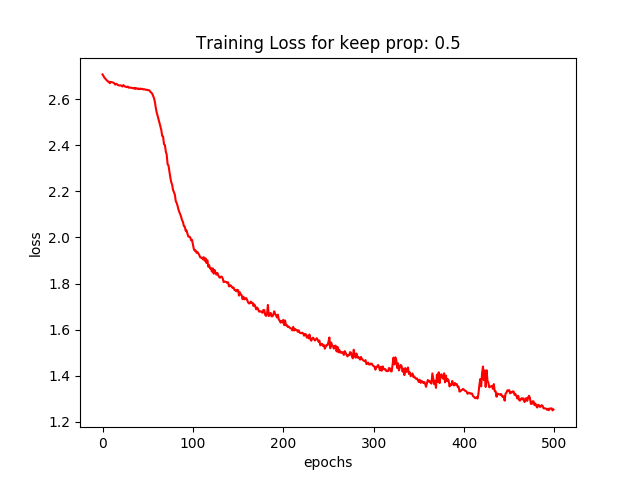
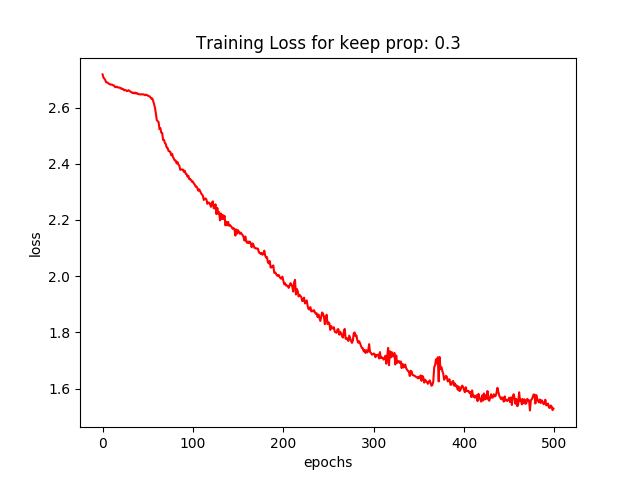
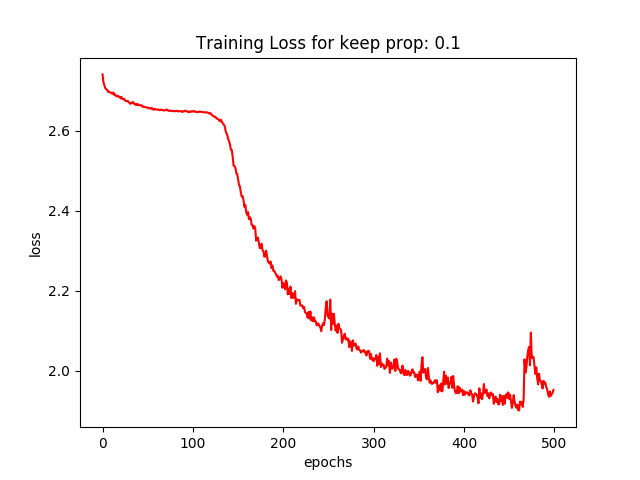
### Test Accuracy for dropouts 0.1 to 1.0



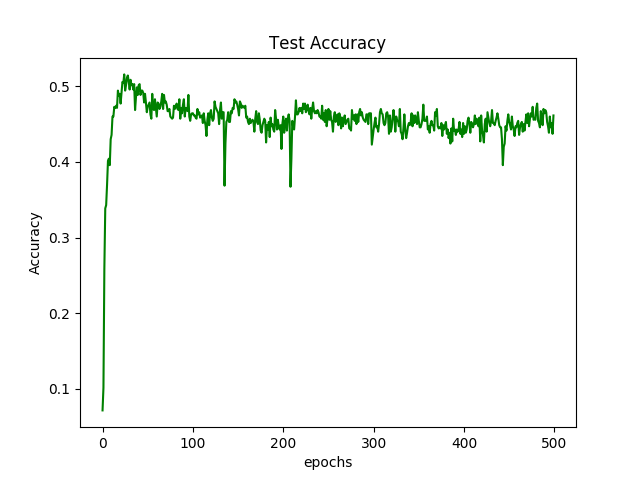
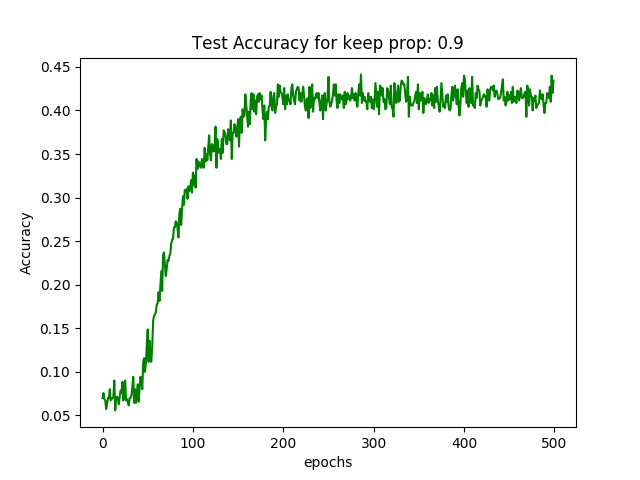
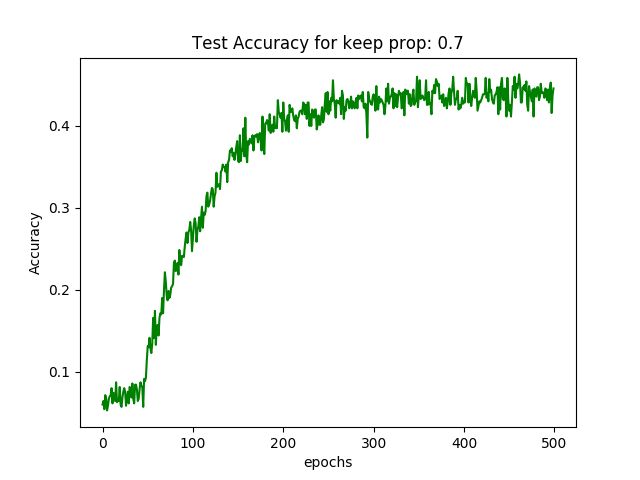
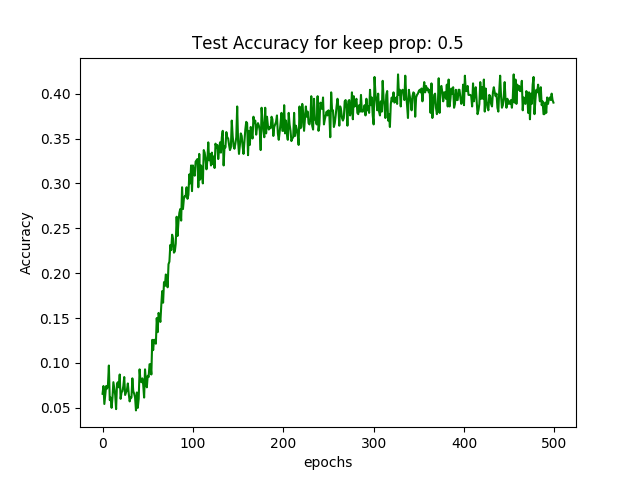
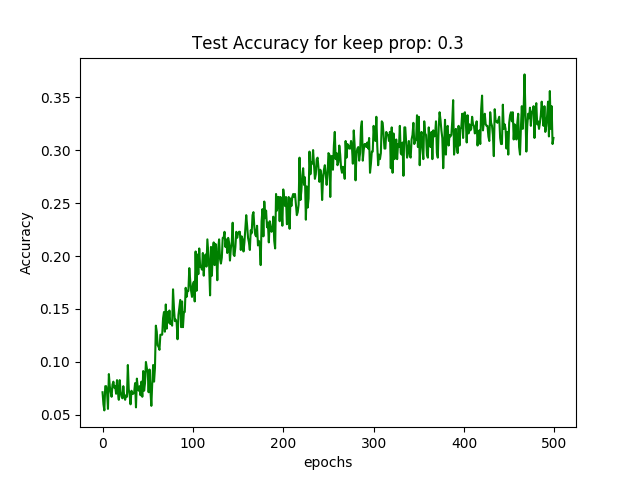
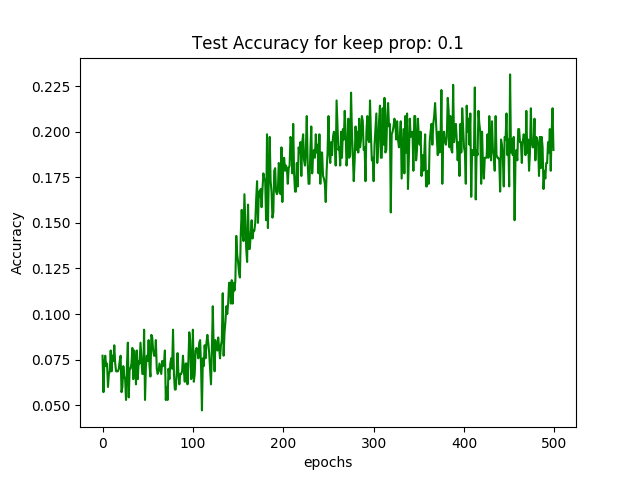
Compared to Character CNN Classifier, having no dropouts is much better for the training of this model. However, it takes a longer time to train as the model is at its most accurate at around 410 epochs. As mentioned, the Word CNN Classifiers gives a much better classification compared to just using characters.

## Character RNN Classifier

### Training Loss for dropouts 0.1 to 1.0



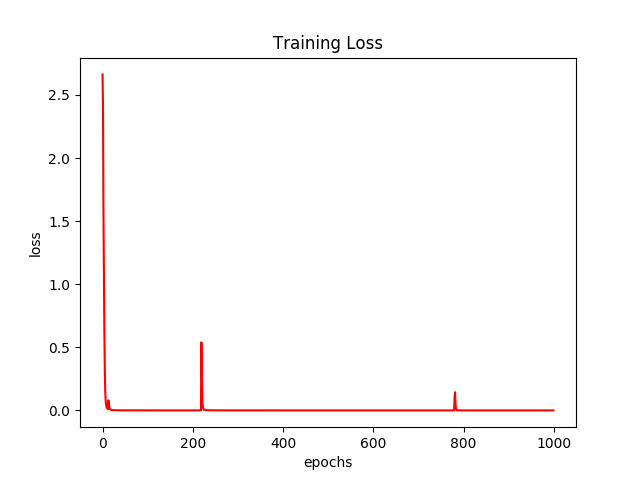
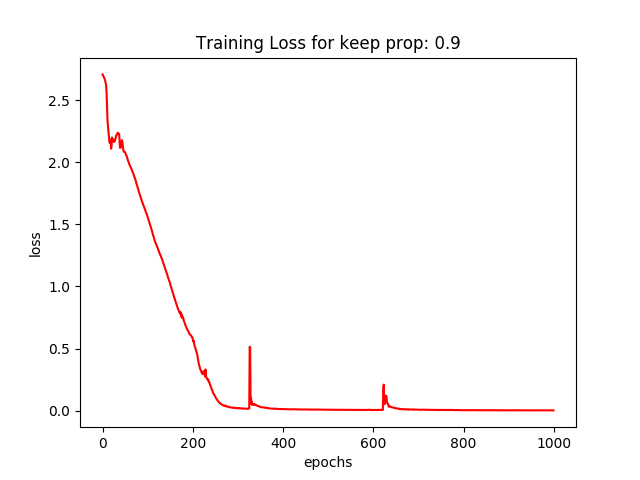
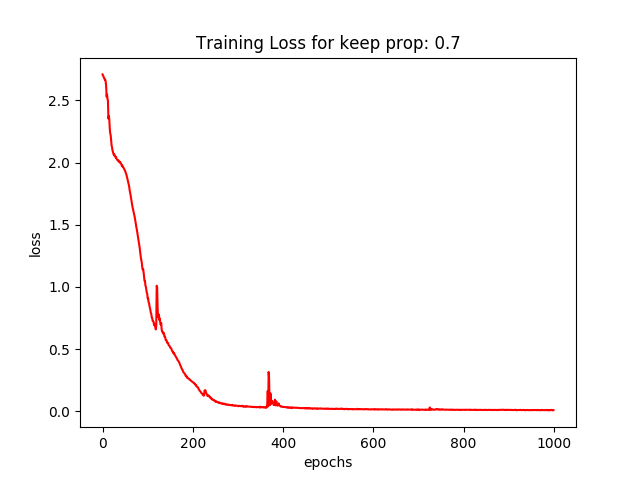
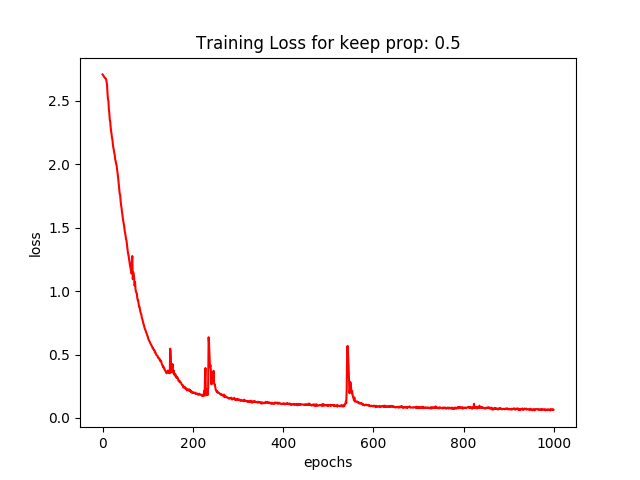
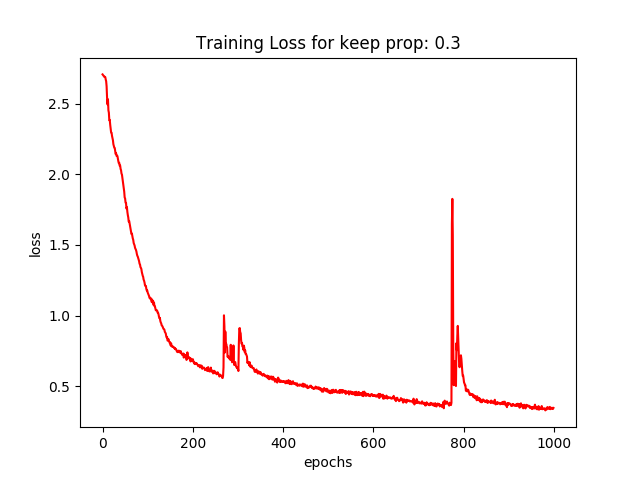
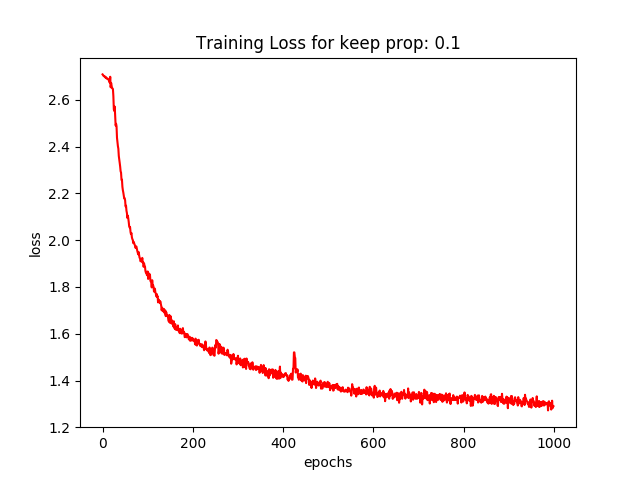
### Test Accuracy for dropouts 0.1 to 1.0



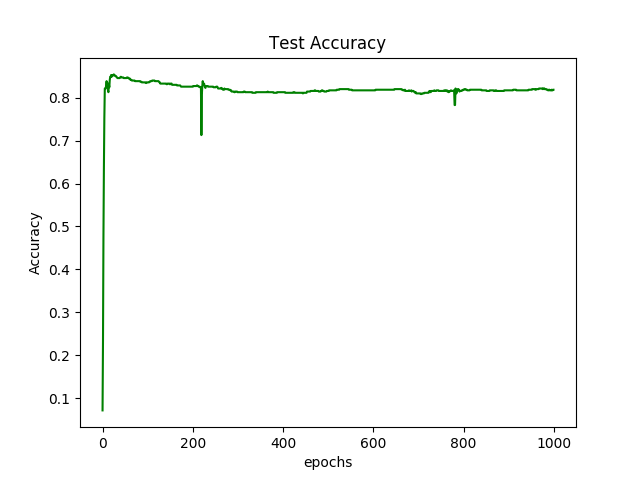
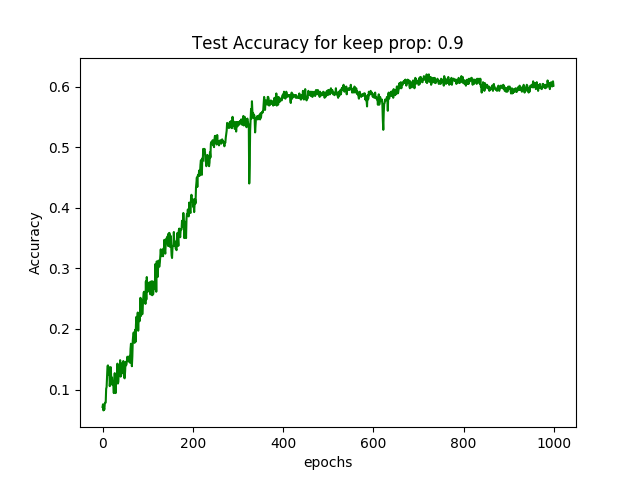
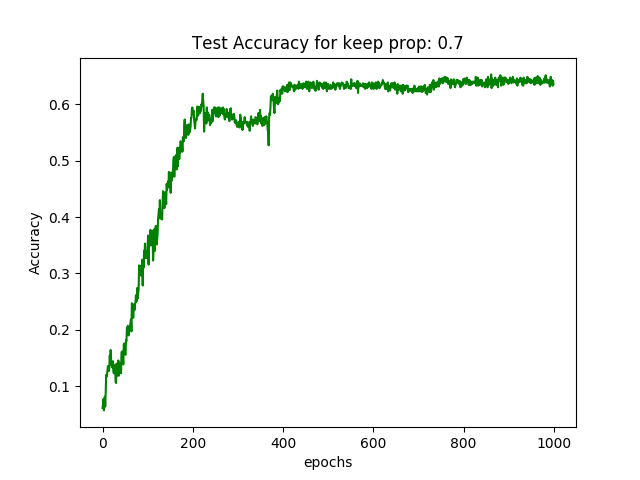
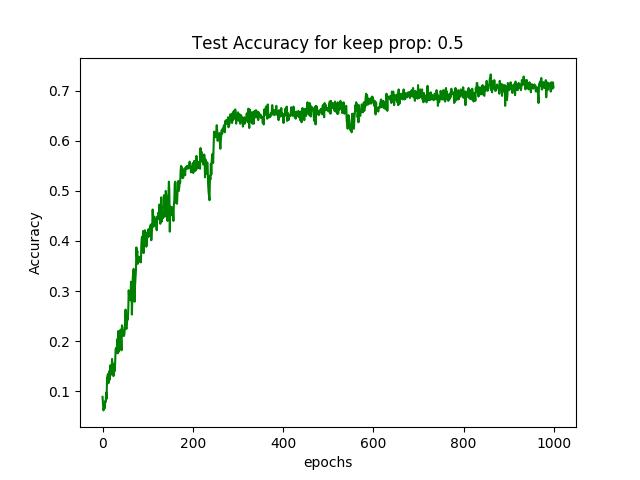
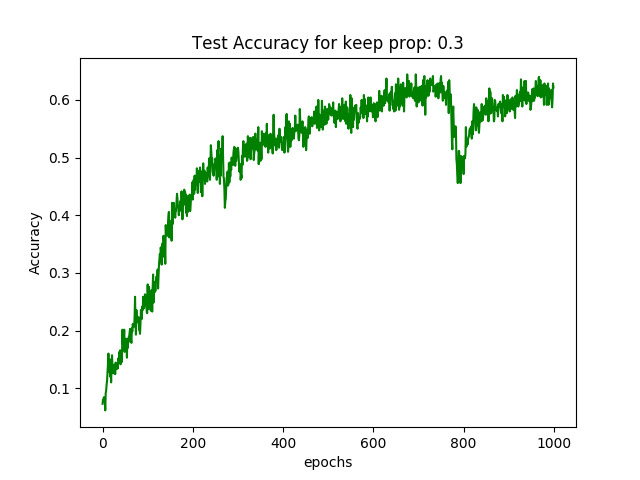
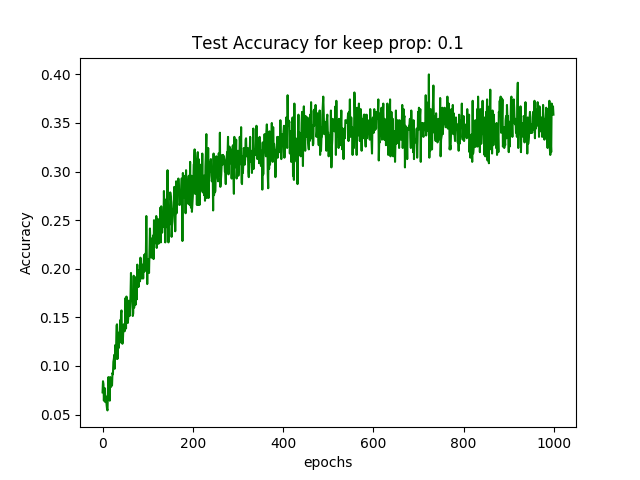
RNN training loss takes awhile to converge regardless of whether dropout is used or not. However, performance is much better when using dropouts as it results in lesser noise. The best setting system to be with a dropout of 50% keep probability as the Test Accuracy converges to a similar value as the rest of the settings but training is less noisy.

## Word RNN Classifier

### Training Loss for dropouts 0.1 to 1.0



### Test Accuracy for dropouts 0.1 to 1.0



The recurring theme for the two models is that using whole words to train a model instead of just the characters result in a better test accuracy. For Word RNN Classifier, the best setting is one where dropout is not used as Training Loss and Test Accuracy