ECE324 Assignment 5

Frederick Boyd

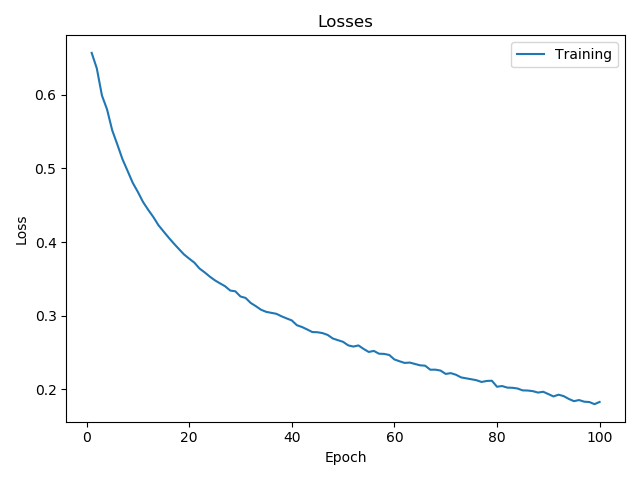
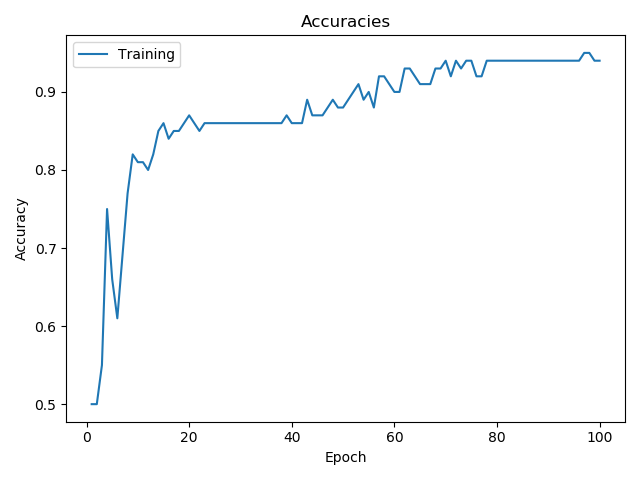
November 7, 2019

**NOTE:** unless otherwise specified, torch.manual\_seed was set to 1

# 3.1 – Create Train/Validation/Test Splits

|  |  |  |
| --- | --- | --- |
| Training:  1 3200  0 3200 | Validation:  1 800  0 800 | Test:  1 1000  0 1000 |

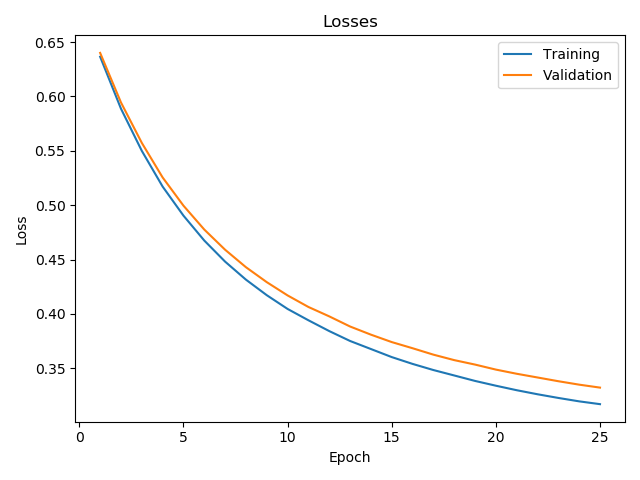
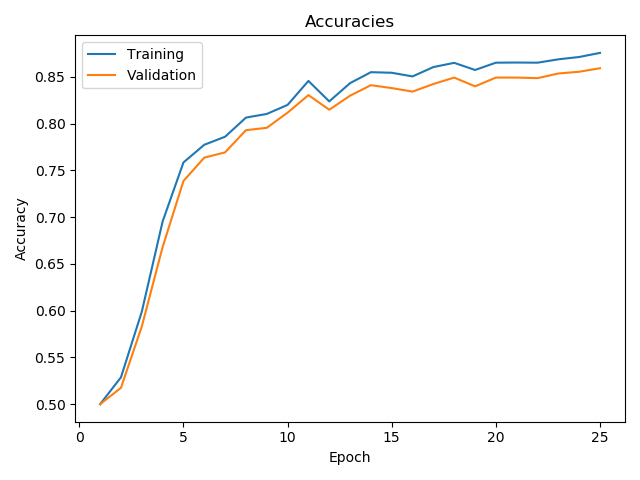
# 4.4 – Overfitting to Debug



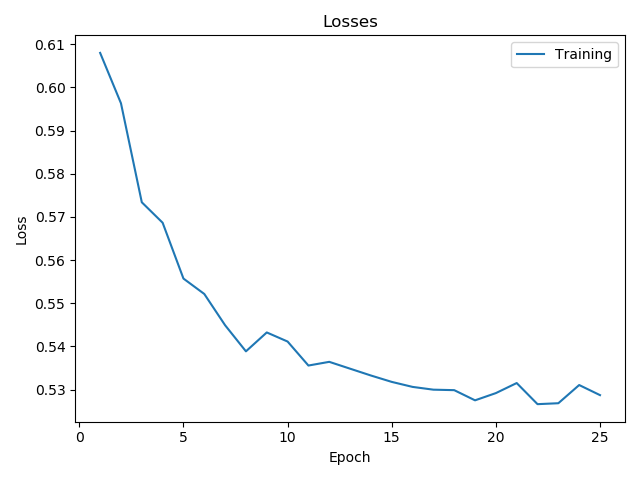
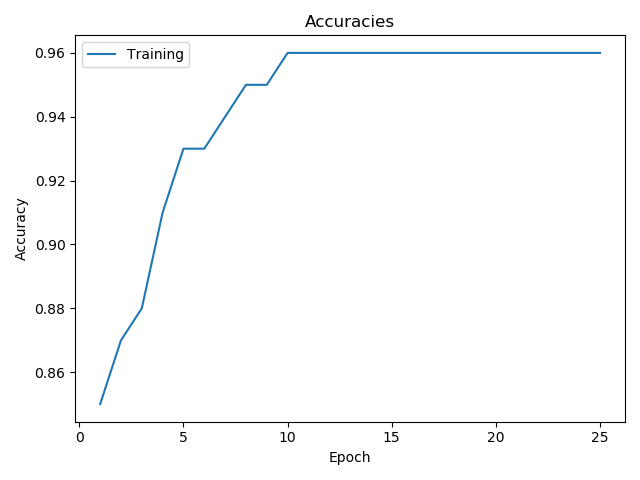
The above figures show the accuracies and losses for the overfit data. A learning rate of 0.05 was used because the default rate of 0.001 was found to be too slow. 100 epochs were used.

# 4.5 – Full Training Data

* Final Test Accuracy: 0.87578125



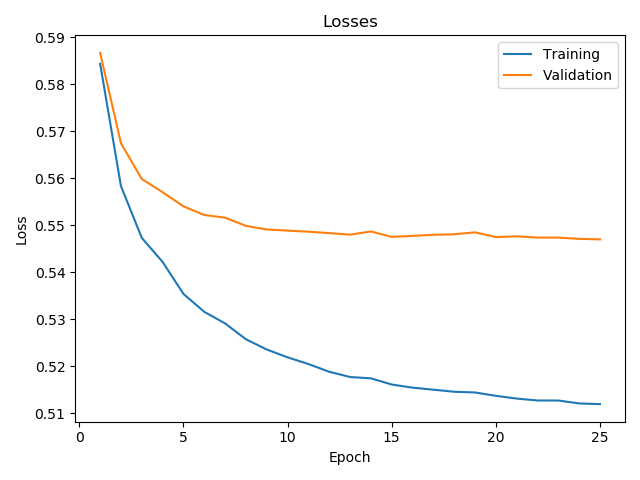
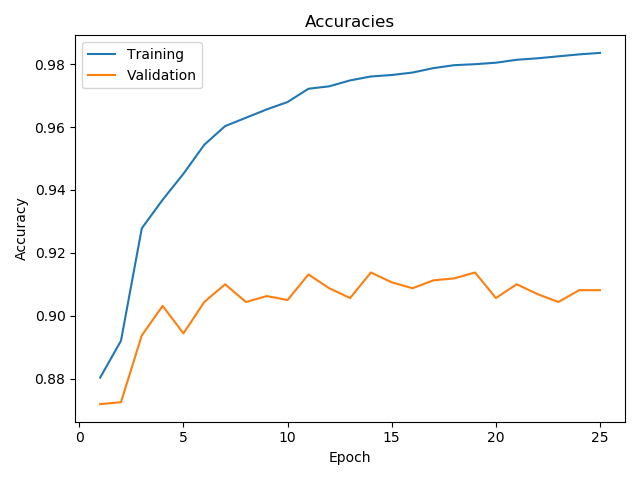
# 5.4 – Overfitting to Debug



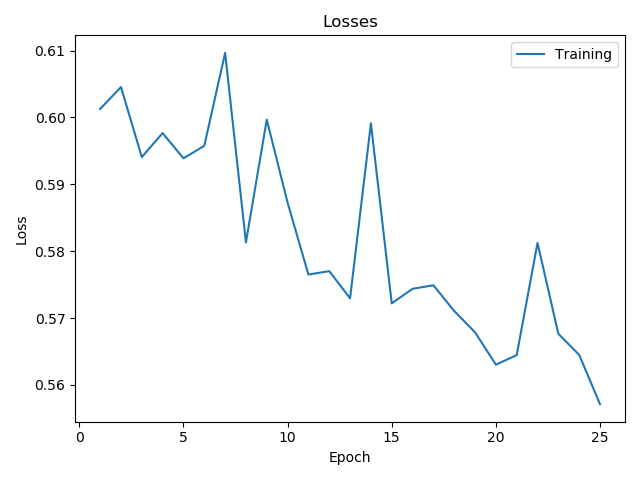
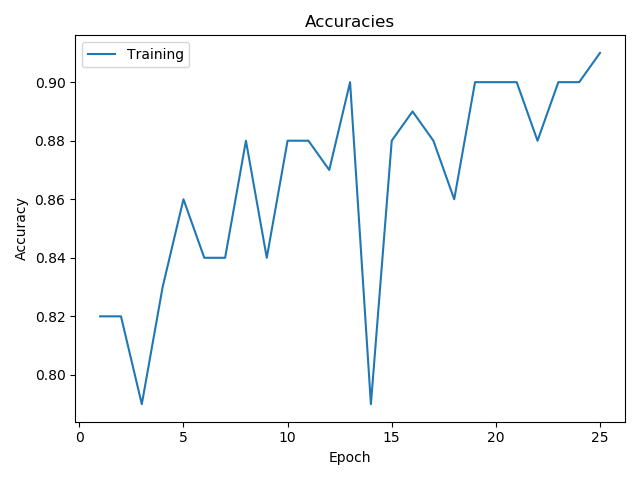
The figures above show the accuracies and losses for the overfit data using a CNN.

# 5.5 – Full Training Data

* Final Test Accuracy: 0.98359375



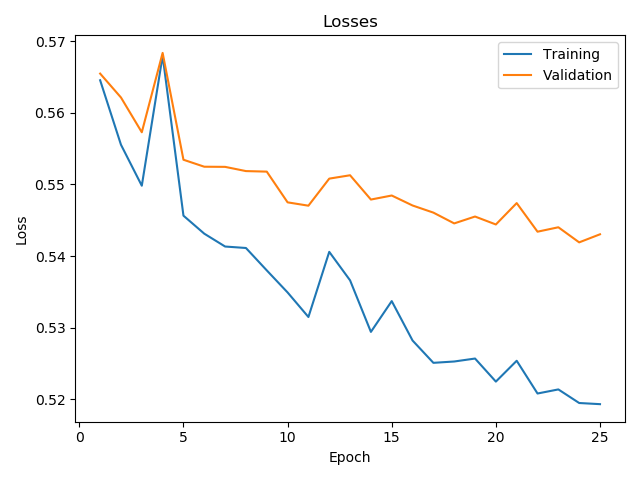
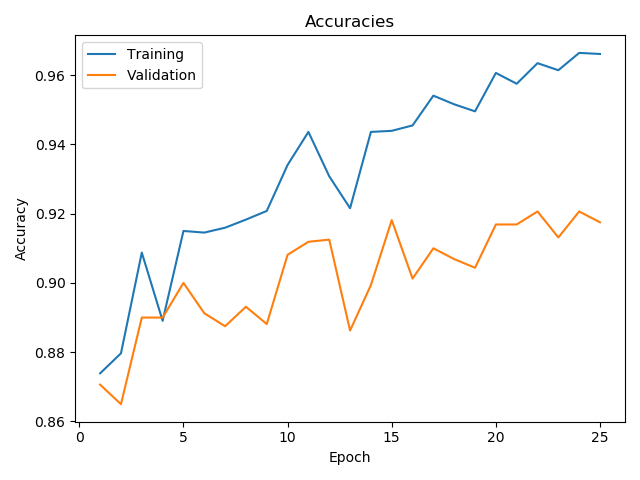
# 6.4 – Overfitting to Debug



The figures above show the accuracies and losses for the overfit data using an RNN.

# 6.5 – Full Training Data

* Final Test Accuracy: 0.96609375



# 8 – Experimental and Conceptual Questions

1. *After training on the three models, report the loss and accuracy on the train/validation/test in a total. There should be a total of 18 numbers. Which model performed the best? Is there a significant difference between the validation and test accuracy? Provide a reason for your answer.*

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | *Baseline* | | | *CNN* | | | *RNN* | | |
| *Train* | *Valid* | *Test* | *Train* | *Valid* | *Test* | *Train* | *Valid* | *Test* |
| *Accuracy* | 0.864 | 0.847 | 0.864 | 0.984 | 0.909 | 0.984 | 0.96 | 0.917 | 0.96 |
| *Loss* | 0.317 | 0.333 | 0.317 | 0.512 | 0.548 | 0.512 | 0.522 | 0.542 | 0.522 |

1. *In the baseline model, what information contained in the original sentence is being ignored? How will the performance of the baseline model inform you about the importance of that information?*

By averaging information for the baseline model, the context of words within the sentence is lost. Feeding the same sentence to the baseline model in a different order would theoretically produce the same output, despite that rearranged sentence having a very different meaning (or none).

Furthermore, two sentences that are very different could happen to have word vectors that average to be approximately the same, since information about each individual word vector is lost. As such, this would result in the model potentially seeing two very different inputs as being the same thing.

Information about individual words vectors and their context within each sentence appears to be relatively important, as the baseline model performs approximately 10% worse than the CNN and RNN models.

1. *For the RNN architecture, examine the effect of using pack padded sequence to ensure that we did indeed get the correct last hidden state (Figure 5 (Right)). Train the RNN and report the loss and accuracy on the train/validation/test under these 3 scenarios:*
   1. *Default scenario, with using pack padded sequence and using the BucketIterator*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Train* | *Valid* | *Test* |
| *Accuracy* | 0.96 | 0.917 | 0.96 |
| *Loss* | 0.522 | 0.542 | 0.522 |

* 1. *Without calling pack padded sequence, and using the BucketIterator*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Train* | *Valid* | *Test* |
| *Accuracy* | 0.949 | 0.916 | 0.949 |
| *Loss* | 0.527 | 0.542 | 0.527 |

* 1. *Without calling pack padded sequence, and using the Iterator. What do you notice about the lengths of the sentences in the batch when using Iterator class instead?*

|  |  |  |  |
| --- | --- | --- | --- |
|  | *Train* | *Valid* | *Test* |
| *Accuracy* | 0.902 | 0.879 | 0.879 |
| *Loss* | 0.549 | 0.561 | 0.568 |

*Given the results of the experiment, explain how you think these two factors affect the performance and why.*

Not calling pack padded sequence in (b) does not appear to impact the performance significantly compared to the default scenario in (a). This small impact in performance is because the BucketIterator is still in place, which ensures that the lengths of sentences in each batch are similar. As such, removing the padding has minimal impact because each batch’s sentences already have nearly identical lengths.

However, when both the pack padded sequence and BucketIterator are removed, the sentences in each batch now have a wide variety of lengths. Without padding, this variety in lengths negative impacts the ability of the RNN to understand sentences because shorter sentences are padded with zeros to ensure that the lengths match before feeding them into the network. As such, the RNN may interpret the padded zeros at the end of the shorter sentences as being some kind of word when there shouldn’t be any word there. This may result in the RNN’s perception of the sentence becoming skewed, thereby reducing the overall accuracy.

1. *In the CNN architecture, what do you think the kernels are learning to detect? When performing max-pooling on the output of the convolutions, what kind of information is the model discarding? Compare how this is different or similar to the baseline model's discarding of information.*

In the CNN architecture, the kernels are learning to detect short phrases within sentences that are key indicators as to whether that sentence is objective or subjective. Since the kernel sizes were ({2, 4}, embedding\_dim), it appears that the CNN is looking at phrases with lengths of two and four words. When performing max-pooling, we lose information about the context of those words within the sentence.

This loss of information is similar to the baseline model in the sense that some of those words’/phrases’ contexts are lost. However, rather than calculating an average over the whole sentence, information about key phrases is emphasized using the kernels. As such, not as much relevant information is lost. Additionally, the max pooling emphasizes all the subjective phrases, since we defined those to have a value of one during categorization.

1. *Try running the subjective\_bot.py script on 4 sentences that you come up with yourself, where 2 are definitely objective/subjective, while 2 are borderline subjective/objective, according to your opinion. Include your console output in the write up. Comment on how the three models performed and whether they are behaving as you expected. Do they agree with each other? Does the majority vote of the models lead to correct answer for the 4 cases? Which model seems to be performing the best?*

|  |  |  |
| --- | --- | --- |
| *Sentence* | *Category* | *Output* |
| “there are 365 days in a year” | Objective | Model baseline: objective (-0.055)  Model cnn: objective (0.021)  Model rnn: objective (0.403) |
| “I don’t like eggplants” | Subjective | Model baseline: subjective (3.767)  Model cnn: subjective (1.0)  Model rnn: subjective (0.998) |
| “capitalism is good for society” | Subjective | Model baseline: subjective (3.517)  Model cnn: subjective (0.637)  Model rnn: objective (0.483) |
| “you must have a degree to get a good job” | Subjective | Model baseline: objective (0.358)  Model cnn: objective (0.065)  Model rnn: subjective (0.954) |

The RNN appears to performing the best. The first two sentences are definitely objective and subjective, and the last two sentences are borderline objective/subjective. For the first two cases, all the models agree with each other. However, they disagree with each other for the last two sentences.

A majority vote leads to the correct answer for all but the fourth sentence: “you must have a degree to get a good job”. The baseline and CNN models have a very confident prediction that this is an objective statement, with values close to 0. However, the RNN confidently and correctly predicts this as a subjective statement. It is interesting to see how the models predict very opposite values.

For the last sentence, the baseline and CNN models may have predicted incorrectly due to the word “must” being in the sentence. However, the RNN is able to look at the whole context of the sentence and appropriately gauge that this is subjective.

Although the RNN only predicted ¾ sentences correctly, the prediction value for the sentence it got wrong was very close to the subjective threshold of 0.5. For the third sentence, “capitalism is good for society”, the RNN predicted it as objective, whereas it should have predicted it as subjective. However, it has a prediction value of 0.483 which is very close to the subjective threshold. Although the other two models also only predicted ¾ sentences correctly, this proximity to the threshold leads me to believe that the RNN is the best-performing model.

1. *Describe your experience with Assignment 5:*
   1. *How much time did you spend on Assignment 5?*

20 hours

* 1. *What did you find challenging?*

Finding the one function/method to use by searching through copious quantities of tutorials and documentations.

* 1. *What did you enjoy?*

Putting in my own sentences to see what the models would predict.

* 1. *What did you find confusing?*

Programming the RNN architecture. Conceptually it makes sense but figuring out how to implement it in PyTorch was very confusing.

* 1. *What was helpful?*

Reading the assignment carefully, Google.