University of Athens
Dept. of Informatics & Telecommunications

Artificial Intelligence II (Deep Learning for Natural Language Processing)

Rontogiannis Dimitrios – `sdi1900165`

Fall 2022 - Homework 2: Sentiment Classifier

1 Introduction

We implement a sentiment classifier using feed for the dataset *imdb-reviews.csv* which is provided using **feed forward neural networks**. In order to choose all the details for developing a good model we run different experiments, with various hidden layer sizes, learning rates, batch sizes, and optimization algorithms. In the given ipynb the experiments are commented out so the reviewer can easily run the test dataset they want. If one would like to re-run the experiments they should feel free to un-comment the code.

2 Preprocessing - Cleaning data

The section below is **similar** with the one from the previous homework as preprocessing the data is an important part of the process. Text data contains noise in various forms like punctuation, text in a different case, symbols with no meaning etc. We perform **seven** different steps to make sure that our data will be ready to be fed to our model.

Initial head of data frame:

	url	rating	review
0	http://www.imdb.com/title/tt0120623/usercomments	10.0	I thought this was a quiet good movie. It was ...
1	http://www.imdb.com/title/tt0043117/usercomments	9.0	Wagon Master is a very unique film amongst Joh...
2	http://www.imdb.com/title/tt0043117/usercomments	10.0	This film has to be as near to perfect a film ...
3	http://www.imdb.com/title/tt0835204/usercomments	4.0	I gave this 4 stars because it has a lot of in...
4	http://www.imdb.com/title/tt0499603/usercomments	10.0	This movie is really genuine and random. It's ...

2.1 HTML

These reviews are taken straight from the IMDB site, so there are some HTML tags that do not affect the sentiment of the sentence. We use regular expressions to filter the tags.

2.2 Number

Numbers might indicate the number of stars a user rated the movie, but also they could indicate the time they watched it, it's price when it was released in the cinemas, etc. Therefore, we use reguler expressions to filter every numeric character.

2.3 Punctuation

Punctuation could show both positive and negative emotion. For example the exclamation mark in the sentence "I wish I could watch this movie again!" has a positive sentiment, but in the sentence "I turned off the tv after 5 minuntes!" has a negative sentiment. We use regular expressions to replace any non-word and non-sentence.

2.4 Uppercase

We lowercase every letter because our model would treat upper case and lower case. We would not want 'A' and 'a' to be seen as different characters.

2.5 Tokenization - Bonus step

Before we continue with the next preprocessing techniques we need to tokenize our text into separate tokens which are essentially individual words.

2.6 Stopwords

We remove these words, which have low-level information, from our text, in order to give more focus to the important information.

2.7 Lemmatization

Lemmatization helps us to achieve the root forms of words. We aim to remove inflectional endings only and to return the base or dictionary form of a word. This way, words like "give", "gave", "giving" will be treated equally.

2.8 Rating - Bonus step

We know that a rating in the range $[0, 4.0]$ is categorized as negative and a rating in the range $[7.0, 10.0]$ is categorized as positive. As we will use Logistic Regression, we transform every negative rating to 0 and every positive rating to 1.

2.9 Rare/Frequent words

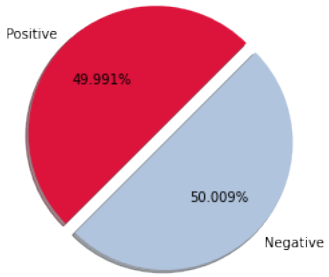
Rare and frequent words almost always offer no information for the sentiment analysis. For example words like "this", "and", "so" etc. are not useful for our model training.

Final head of data frame:

	url	rating	review
0	http://www.imdb.com/title/tt0120623/usercomments	1	[think, quiet, fun, watch, best, outtake, end,...
1	http://www.imdb.com/title/tt0043117/usercomments	1	[wagon, master, unique, amongst, john, ford, w...
2	http://www.imdb.com/title/tt0043117/usercomments	1	[near, perfect, john, ford, magic, masterpiece...
3	http://www.imdb.com/title/tt0835204/usercomments	0	[give, star, lot, interest, theme, many, alrea...
4	http://www.imdb.com/title/tt0499603/usercomments	1	[really, genuine, random, really, hard, find, ...

3 Training

We train our model with data with almost the same number of positive and negative reviews:



3.1 GloVe

GloVe is an unsupervised learning algorithm for obtaining vector representations for words. For our dataset we will use the pre-trained vectors included in the file **glove.6B.zip** take from stanford's website. We will split our dataset in training, validation, and testing datasets and for each set we will vectorize our sentences by taking the mean vector of the vectors presented in the sentence.

The file **glove.6B.zip** contains vectors of 400.000 words but some of the words presented in our dataset may not be recognized. However, actually only around **0.06 percent** of the dataset is not recognized, which is very acceptable. Below you can the percentage of recognized words in the datasets for training, validating, and testing.

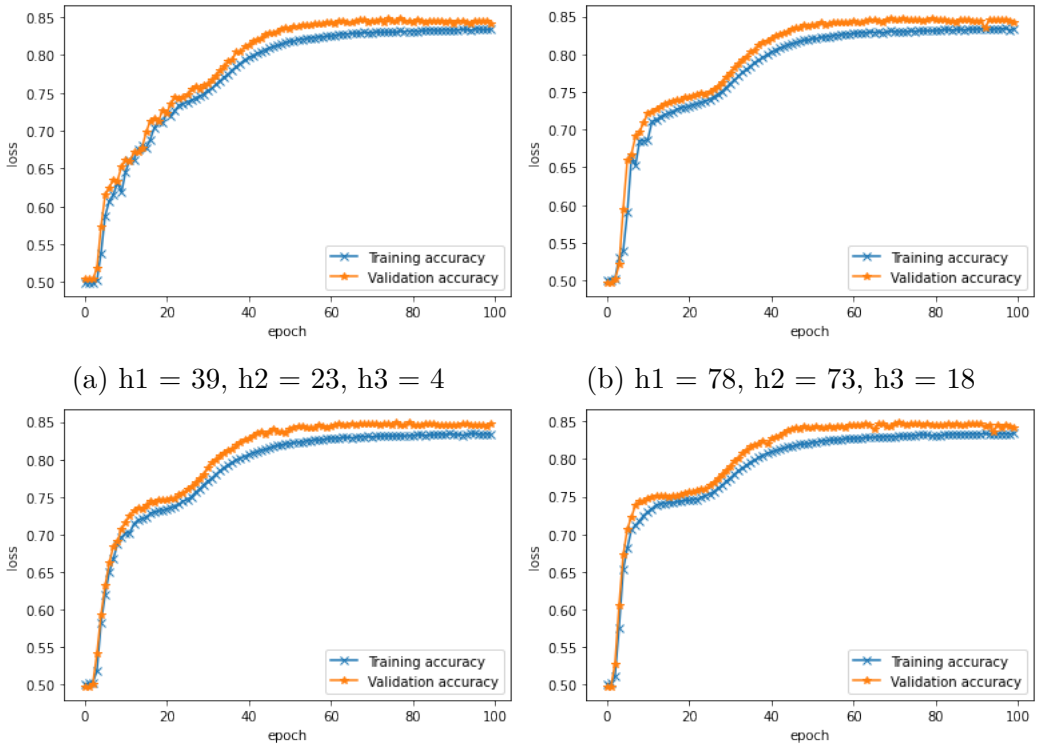
```
Recognized words: 3754726 Total words: 3922278 Ratio: 0.9572819672649414

Recognized words: 473874 Total words: 494881 Ratio: 0.9575514113493951

Recognized words: 469472 Total words: 489972 Ratio: 0.9581608744989509
```

We experimented with 50 and 200 features but as it was expected the accuracy of the model was significantly lower. So, for this homework 300 features will be used for vectorization of the words.

3.2 Feed Forward Neural Network

For our Neural Network we designed it to have 3 hidden layers (four functions in total), with the addition of the sigmoid function in the end. We experimented with more hidden layers but the accuracy difference was insignificant and the performance timewise was much worse. We built a function **train_and_test_hyperparameters** which takes as arguments the hidden layer sizes, the learning rate, the batch size, the number of epochs, the optimizer, and the loss function of our model. We used this function for experimenting with multiple different arguments i.e. models. We also built a function **plot_loss_accuracy** with arguments the number of epochs, the average loss and accuracy of batches on the training set and the average loss and accuracy of batches on the validation set, which make the necessary plots. We use this function alongside the plotting of ROC curve to spot changes in performance. Below only the plot for accruacy on training and validation set is presented so the reader is not overwhelmed by the huge amount of plots and data.

3.3 Hidden layer sizes



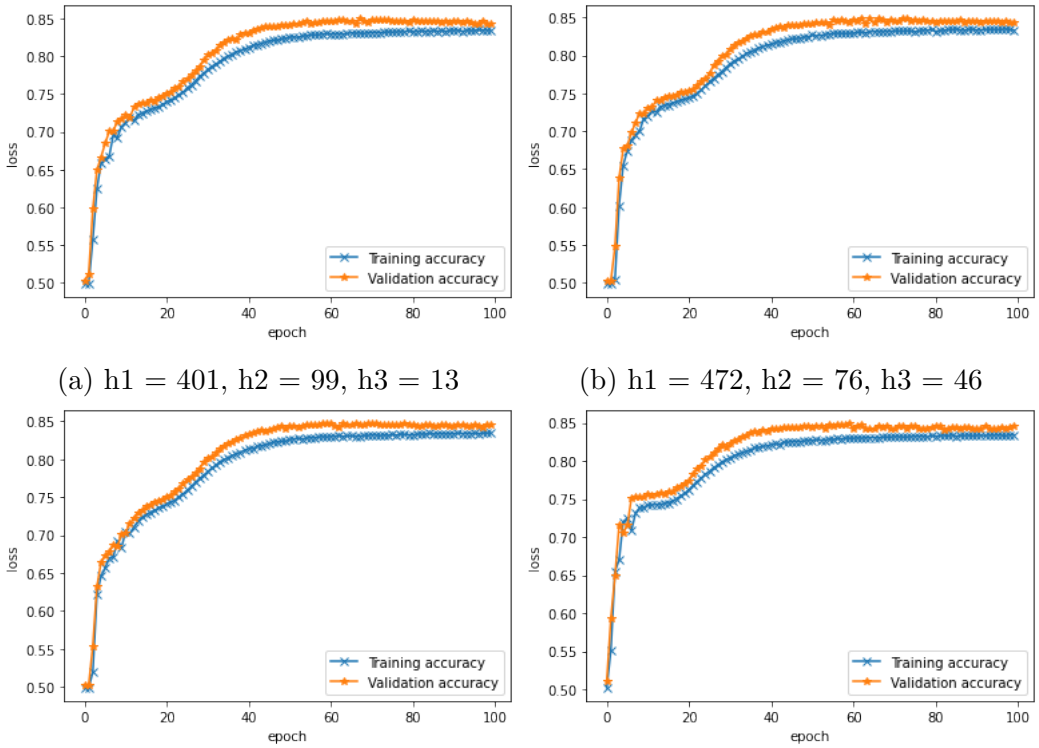
(a) h1 = 39, h2 = 23, h3 = 4

(b) h1 = 78, h2 = 73, h3 = 18

(c) h1 = 146, h2 = 74, h3 = 19

(d) h1 = 398, h2 = 55, h3 = 15

Figure 1: Different hidden layer sizes



(a) h1 = 401, h2 = 99, h3 = 13

(b) h1 = 472, h2 = 76, h3 = 46

(c) h1 = 522, h2 = 13, h3 = 8

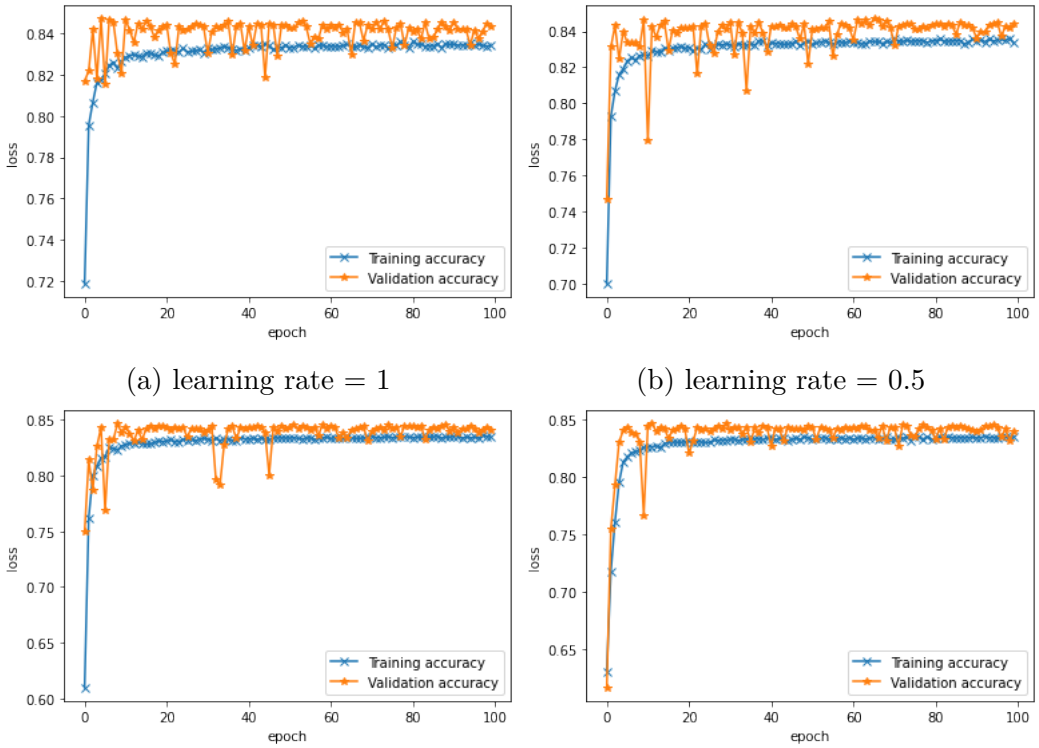
(d) h1 = 543, h2 = 336, h3 = 200

Figure 2: Different hidden layer sizes

We can see that the differences in accuracy of the final model in the testing dataset are 0.01 to 0.015 at most. However, we have significantly larger differences in the average accuracy of batches in training and validations datasets. For our next experiments we will use h1 = 256, h2 = 128, h3 = 64.

3.4 Learning rate

In contrast, we can observe noteworthy results by experimenting on the learning rate:



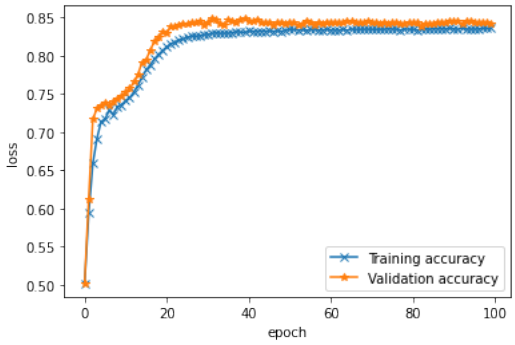
(a) learning rate = 1

(b) learning rate = 0.5

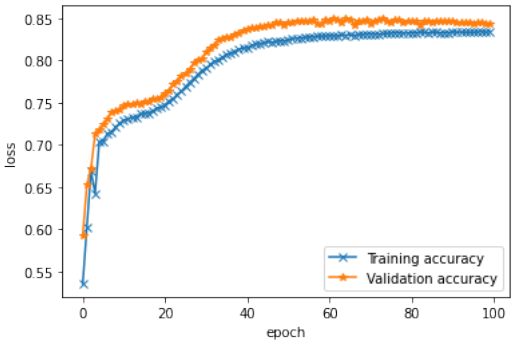
(c) learning rate = 0.1

(d) learning rate = 0.05

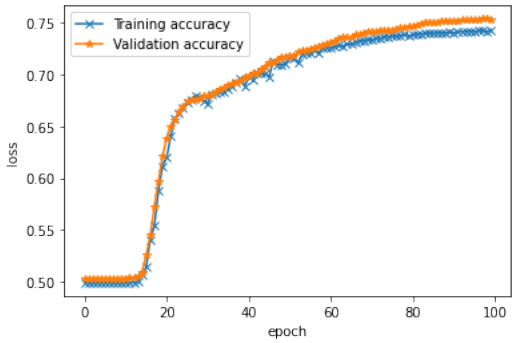
Figure 3: Different learning rates



(a) learning rate = 0.01



(b) learning rate = 0.005



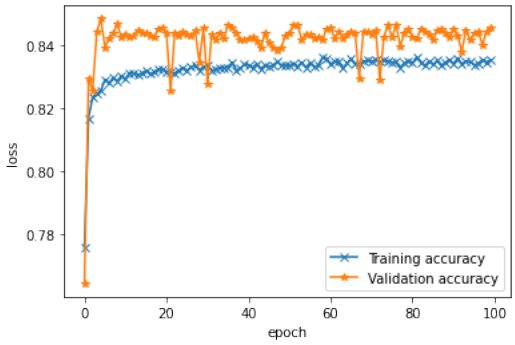
(c) learning rate = 0.001

Figure 4: Different learning rates

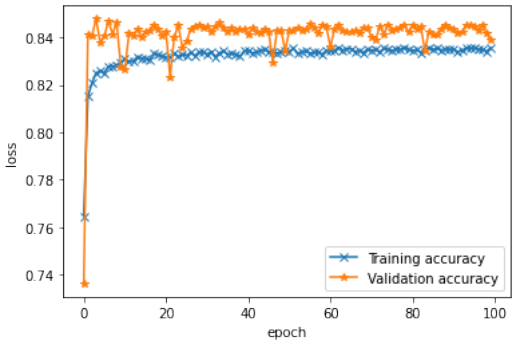
Here we can see that with high learning rates the accuracy is very unstable. However, with very low learning rates the model probably needs more epochs, but it is more stable. We will use as best learning rate 0.005.

3.5 Batch sizes

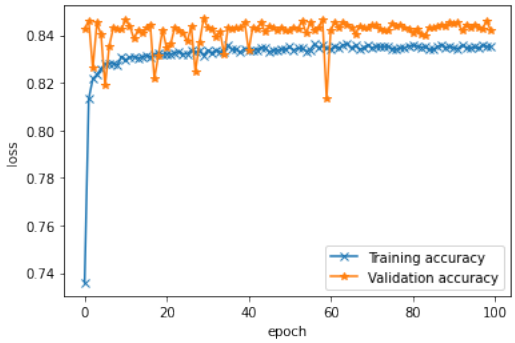
Below we can see the difference in model's performance bases in batch sizes:



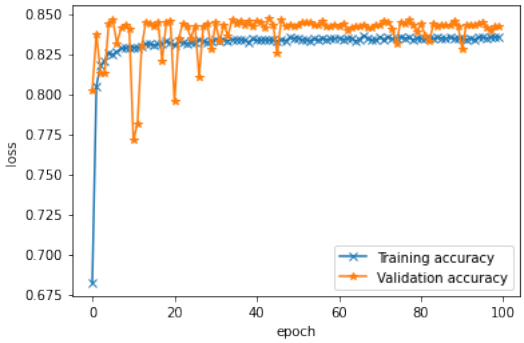
(a) batch size = 2



(b) batch size = 4

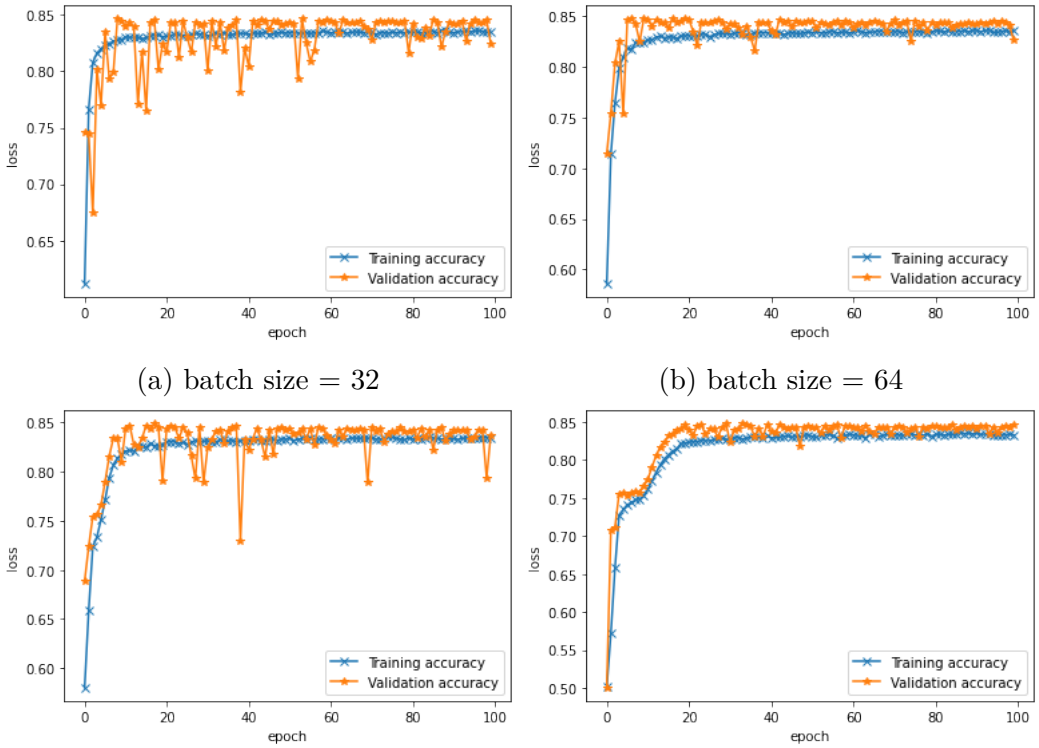


(c) batch size = 8



(d) batch size = 16

Figure 5: Different batch sizes



(a) batch size = 32

(b) batch size = 64

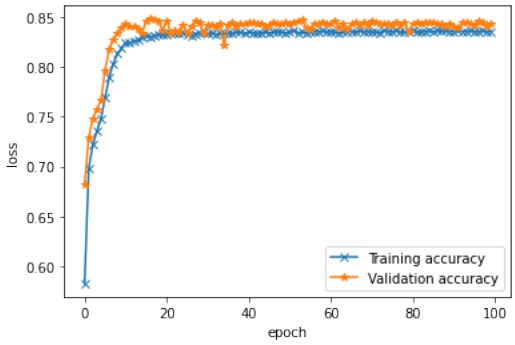
(c) batch size = 128

(d) batch size = 256

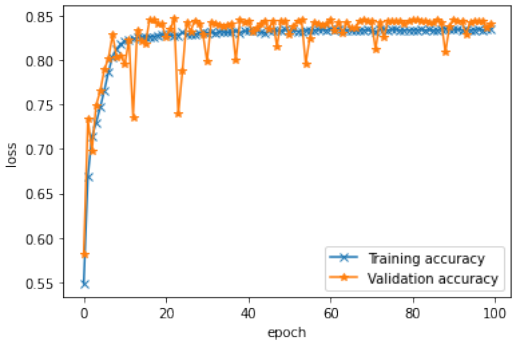
Figure 6: Different batch sizes

3.6 Optimizers

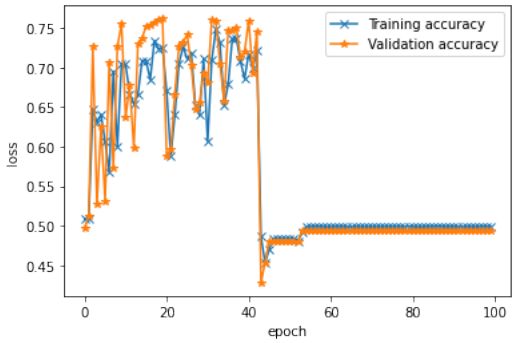
Below we experiment with different optimizers. An optimizer is a function or an algorithm that modifies the attributes of the neural network, such as weights and learning rate. Thus, it helps in reducing the overall loss and improve the accuracy. We test Adadelta, ASGD, Adamax, Rprop, and SGD.



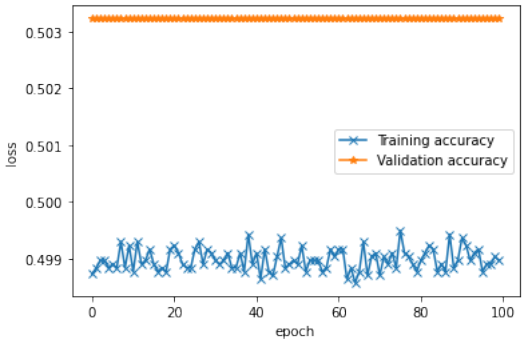
(a) Adadelta



(b) ASGD

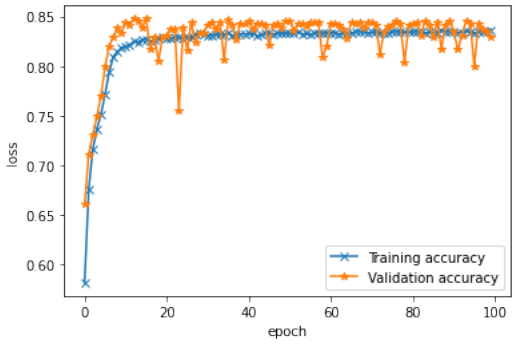


(c) Adamax



(d) Rprop

Figure 7: Different optimizers



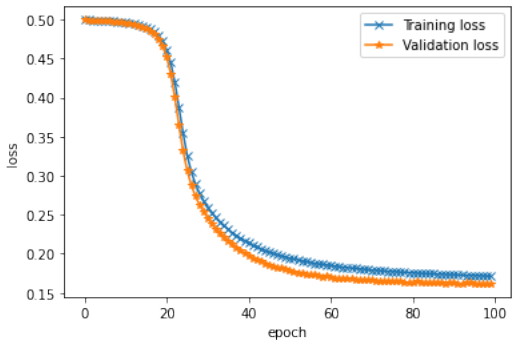
(a) SGD

Figure 8: Different optimizers

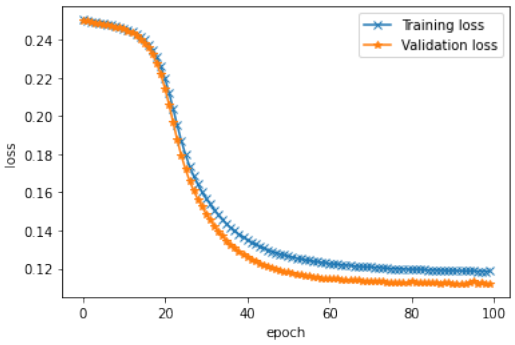
It is obvious that Adamax (a variant of Adam based on infinity norm) and Rprop (implements the resilient backpropagation algorithm) are not suitable for this kind of problem. However, the other three optimizers have more or less the same accuracy, with the Adadelta algorithm being the more ”stable” one.

3.7 Loss Function

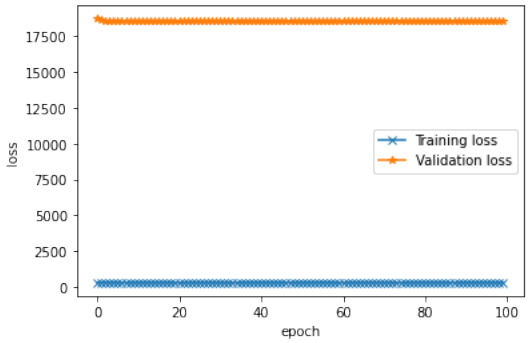
We will also experiment with different loss functions for our model. Below we present the curves of the loss for training and validation according to each epoch.



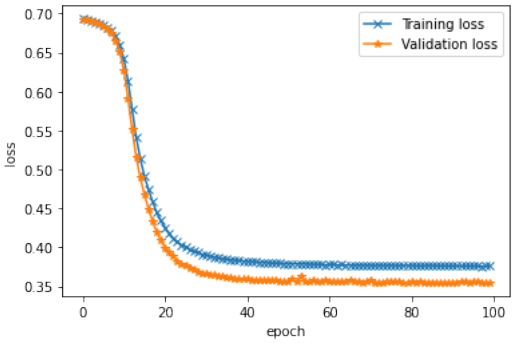
(a) L1Loss



(b) MSELoss

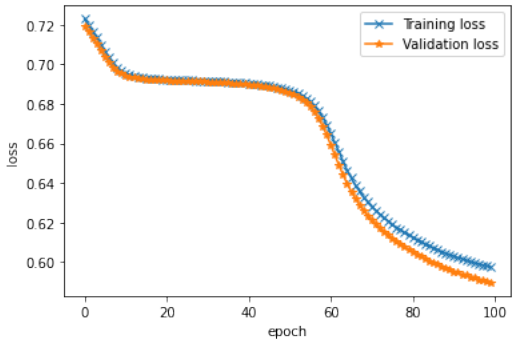


(c) CrossEntropyLoss

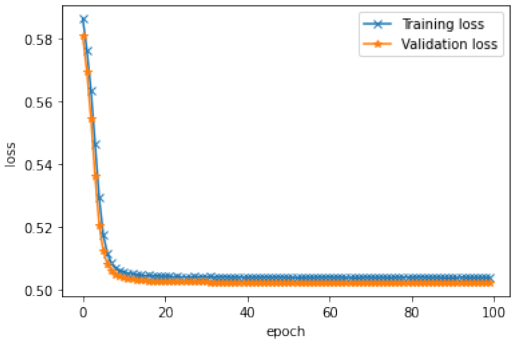


(d) BCELoss

Figure 9: Different Loss Functions



(a) BCEWithLogitsLoss



(b) SoftMarginLoss

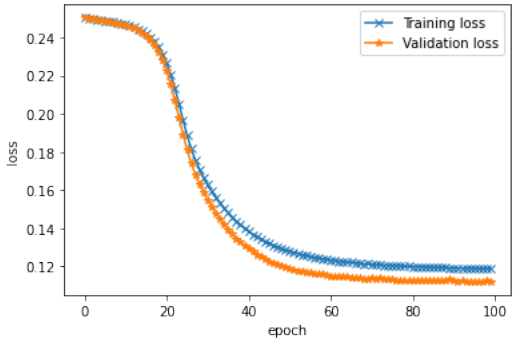
Figure 10: Different Loss Functions

It seems that MSELoss and BCELoss have around the same accuracy and ROC auc area.

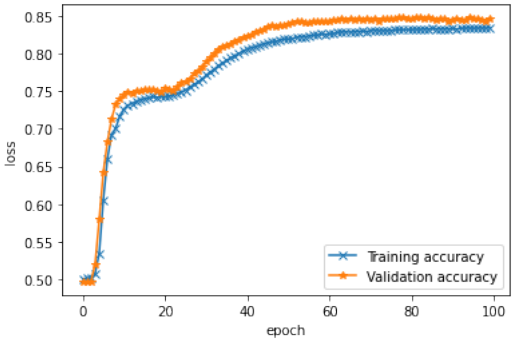
However, as BCE loss seems to be more suitable for binary classification, so we will use this one.

4 Learning Curve

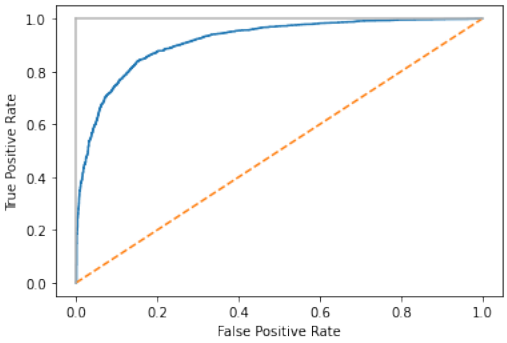
So now we have come to the end of our experiments with the best model having these characteristics: Optimizer - Adadelta, h1 = 256, h2 = 128, h3 = 64, learning rate = 0.01, batch size = 128. Below we can see all the necessary curves for our model:



(a)



(b)



(c)

	precision	recall	f1-score	support
0.0	0.84	0.84	0.84	2224
1.0	0.85	0.84	0.84	2277
accuracy			0.84	4501
macro avg	0.84	0.84	0.84	4501
weighted avg	0.84	0.84	0.84	4501

(d)

Figure 11: Learning Curves

ROC auc score: 0.9164646543634657

5 Testing

The reviewer is welcome to change the values of dataset and testset path in the beginning of the ipynb file. The testing can be done by executing the **two last cells**, which trains the model with the hyperparameters found in the above experiments, with the dataset from the dataset path and then predicting the sentiment of the reviews from the testset from testset path.

6 Conclusion

Even though our "best model" has a decent accuracy of around 85 percent we still can not reproduce the results we have one the previous homework. A huge role play the number of features which is only 300 in this case and not 16500 that we had. However, as we had seen in the previous conclusion, the model with accuracy around 90 percent was a bit overfitting, and the real best model had accuracy of around 85 percent. So we have a similar performance overall.