Deep Learning Assignment 2 Report

The tables below show the different techniques that were explored to try and improve the performance of the models in Task 1. The values shown are Mean Squared Error (MSE) and Mean Absolute Error (MAE). Note that results are compounded, i.e. changes that yield improved results are kept and implemented before the next technique is applied.

Key: 🟩 Improved / Best result so far 🟥 Degraded / Stayed the same

Trying different word embeddings

Model Type	Word Embedding	MSE	MAE
MLP - Single vector	Word2Vec	1.698410916328	1.0142000198364258
	GloVe	1.6982539653778077	1.014128828048706
MLP - Sequence	Word2Vec	1.6672266721725464	1.000108242034912
	GloVe	1.666319727897644	0.9994981288909912
CNN	Word2Vec	1.6667307615280151	0.9998515844345093
	GloVe	1.6659255027770996	0.9994120001792908
RNN	Word2Vec	1.6655337810516357	0.9991471767425537
	GloVe	1.6651641130447388	0.9990426898002625

In the first 3 models, it was found that training the model with a GloVe word embedding provided slightly lower (and thus improved) MSE and MAE results than those obtained after using Word2Vec with the same test dataset. This improvement may be due to GloVe's superior ability to prevent frequent words with little training value like "the", "a", etc from dominating the training process; GloVe does this by assigning lower weight for these high frequency words. This is particularly applicable to the test dataset used as these 'filler' words are very frequent due to the informal nature of the reviews.

The RNN model saw more improvement with the Word2Vec word embedding than with GloVe, this may be due to suitability between the recurrent nature of the RNN model and the way in which Word2Vec uses data from relationships between local co-occuring words rather than GloVe which uses global co-occurrence between words i.e. over the whole corpus.

Changing the batch size

Model Type	Batch Size	MSE	MAE
MLP - Single vector	32	1.698410916328	1.0142000198364258
	16	1.6980005502700806	1.0140003442764283
MLP - Sequence	32	1.6672266721725464	1.000108242034912

	16	1.6656978130340576	0.9992766380310059
CNN	32	1.6659255027770996	0.9994120001792908
	16	1.6653740406036377	0.9991756677627563
RNN	32	1.6655337810516357	0.9991471767425537
	16	1.6652402877807617	0.999068558216095

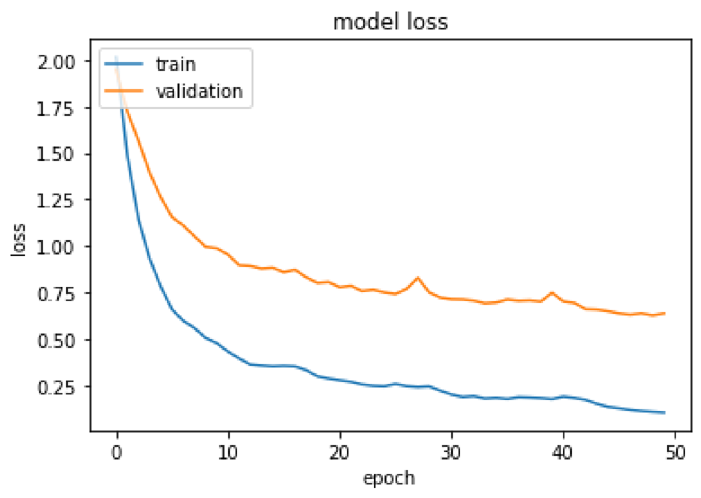
By parameter tuning, it was found that reducing the batch size to 16 was the most suitable value as it yields an improved MSE and MAE results for all 4 models without drastically increasing the computation time, as found with smaller batch sizes.

Increase the number of epochs

Model Type	Epochs	MSE	MAE
MLP - Single vector	30	1.698410916328	1.0142000198364258
	50	1.6979999542236328	1.0140000581741333
	100	1.6979999542236328	1.0140000581741333
MLP - Sequence	30	1.6672266721725464	1.000108242034912
	50	1.6649999618530273	0.9990000128746033
	100	1.6649999618530273	0.9990000128746033
CNN	30	1.6653740406036377	0.9991756677627563
	50	1.6649999618530273	0.9990000128746033
	100	1.6649999618530273	0.9990000128746033
RNN	30	1.6652402877807617	0.999068558216095
	50	1.665003776550293	0.9990010261535645
	100	1.665003776550293	0.9990010261535645

Increasing the number of epochs means the number of times the weights are changed in the neural network are increased and the curve goes from underfitting to optimal to overfitting. 50 epochs was found to be a suitable value as it doesn't cause overfitting or underfitting and improves both MSE and MAE in the four models, whereas in some cases 100 epochs caused undefit learning curves.

Example: CNN model with 50 epochs



Change the structure and activation functions of the Neural Network

Structure 1: Relu (25), Relu (25), Sigmoid (1)
Structure 2: Relu (25), Relu (25), Relu (1)
Structure 3: Relu (25), Relu (25), Relu (12), Relu (12), Softmax (1)
Structure 4: Relu (50), Relu (50), Softmax (5)

Model Type	Structure no.	MSE	MAE
MLP - Single vector	1	1.698410916328	1.0142000198364258
	2	0.004566929768770933	0.04744506627321243
	3	1.6979999542236328	1.0140000581741333
	4	3.9604000568389894	1.8139999628067016
MLP - Sequences	1	1.6672266721725464	1.000108242034912
	2	0.005426486488431692	0.05510782822966575
	3	1.6649999618530273	0.9990000128746033
	4	3.9034006595611572	1.7990009784698486
CNN	1	1.6651870012283325	0.9990929961204529
	2	0.0078877843916416	0.0681717246770858
	3	1.6649999618530273	0.9990000128746033
	4	3.9034006595611572	1.7990009784698486
RNN	1	1.665000081062317 Underfitting curve observed	0.999000072479248 Underfitting curve observed
	2	0.001831210334785282	0.02916676737368107

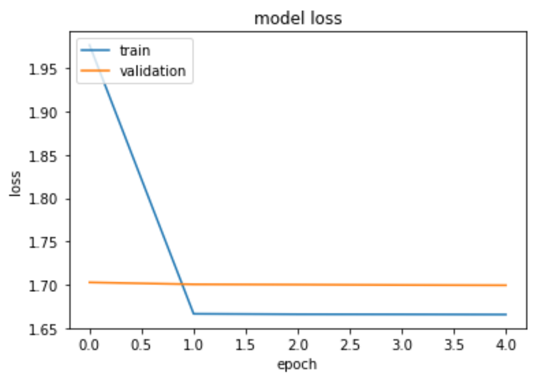
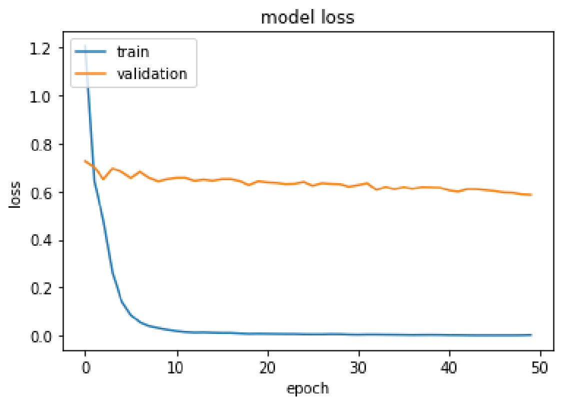
	3	1.6649999618530273	0.9990000128746033
	4	3.9034006595611572	1.7990009784698486

Surprisingly, adding more hidden layers (structure 3) into the 4 models only caused a very small decrease in MSE and MAE, not as much as in structure 2 which saw the most dramatic improvement out of the 4 tested structures. Adding more neurons to hidden layers and the output layer (structure 4) had a strange effect of significantly increasing the MSE and MAE. The results also seemed to stagnate across models with this structure. In the RNN model, using Sigmoid or Softmax activation functions in the output layer resulted in underfitting learning curves. To avoid this, the Relu activation function was used throughout the entire neural network model which resulted in more optimal looking curves and better MSE and MAE results.

Model Selection

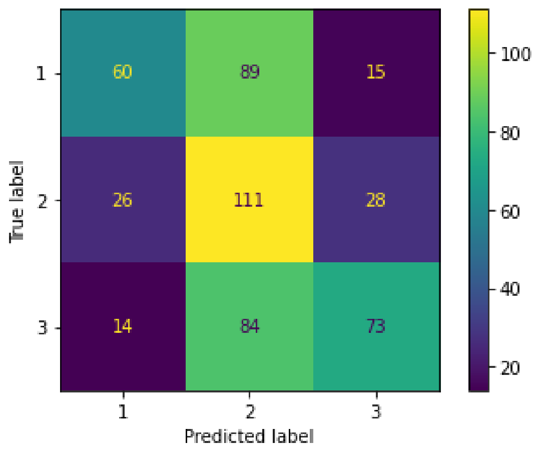
The RNN model from Task 1 was selected as the optimal it performed the best and gave the lowest results for MSE and MAE. Although the end results are very similar, RNN was chosen because it is an improvement upon the two MLP models and because RNNs are better suited to analysing text data than CNNs which are more commonly used alongside spatial datasets such as images. The following table shows the final MSE and MAE results after the models have been improved and optimised.

Model Type	MSE	MAE
MLP - Single vector	0.004566929768770933	0.04744506627321243
MLP - Sequence	0.005426486488431692	0.05510782822966575
CNN	0.007887784391641617	0.06817172467708588
RNN	0.001831210334785282	0.02916676737368107

RNN before improvement	**RNN after improvement**
	

Confusion Matrices

Regression Model



This 3x3 confusion matrix shows the true label values against the predicted label values. The top row represents values that actually have the label 1, the second row represents values that actually have the label 2 and so on for the last row. The main diagonal (60, 111, 73) gives correct predictions where the true values and predicted values are the same. The middle row shows that the majority of label 2's were correctly predicted (111 correct predictions and 54 incorrect predictions) but for rows 1 and 3 (Row 1: 60 correct predictions and 104 incorrect predictions) (Row 3: 73 correct predictions and 98 incorrect predictions), most predictions were incorrect, this could be due to the informal and difficult to contextualise tone of either very good reviews or very bad reviews. In future, trying the same model with different word embeddings that consider context may help improve this imbalance.

Conclusion / further improvements

In conclusion, this project was successful in determining which of the four ANN models was the most suitable for handling the given dataset. All four models were implemented in Python, they underwent numerous improvements to achieve a more optimum result and at the end they were compared fairly over the same test data. In the end RNN was found to be the most suitable for this text classification scenario and produced the lowest MSE and MAE results and presented an optimal learning curve..

Before the improvement section of this report, it was observed that the graphs for each of these models initially were quite unusual and displayed the characteristics of 'underfitting' or gradient vanishing, although I initially thought this may have been a problem with the model's implementation, it was fixed by applying various different improvement techniques i.e. using Relu function instead of Sigmoid helped to improve the RNN graph's underfitting problem.

In future, it would be beneficial to explore other ANN models that specialise in handling text data. For example, character-level CNNs that can model text data at the character level or 'deeper' ANN models may give a more accurate result. Other pre-trained word embeddings could also be considered for their differences and advantages over Word2Vec and GloVe; such as BERT (which utilises transformer architecture to calculate relationships between words), ELMo (which learns context-dependant data about words) or Fasttext (which considers sub-words and learn words that aren't already in its vocabulary, something Word2Vec and GloVe cannot do).

Given more time, the number of epochs and batch sizes could have been better fine tuned to find the optimum values for this dataset, although this was not explored as extensively because a large number of epochs and/or a small batch size can cost significant computational time.