Deep Learning. Assignment 2

Nico Sharei (nsharei), Artem Bisliouk (abisliou)

May 5, 2024

1 Pre-trained Embeddings & Visualization

1.1 a)

The file *data/word-embeddings.txt* contains a subset of 29841 pre-trained GloVe embeddings. Each row has the format:

word dim1 dim2 dim3 ... dimN

- `word` is a string that represents a specific word in the vocabulary.
- `dim1` to `dimN` are the numerical values (floats) that constitute the embedding vector of the word. The length (`N`) of this vector is the dimension of the embedding space, which is 100 in our case.

An embedding is a vector that encodes a word with arbitrary features. While there are various algorithms for learning these features (e.g. skip-gram [5], co-occurrence matrix factorization [6]), they all aim to encode semantic relations between words in a higher-dimensional space. As illustrated in Figure 1, the words *Man* and *Male* are conceptually similar, resulting in their feature vectors being closely related. Furthermore, the offset between the tuples $(Man, Woman)$ and $(King, Queen)$ appears to be identical. This leads to the assumption that the feature on the x-axis may, to some degree, encode the gender of the word. A more feminine word would therefore have a higher value, while a more masculine word would have a lower value. A comparison of dense word representations with sparse representations, such as one-hot encoding, reveals that the latter's key advantage is the capturing of semantic relationships, which leads to better generalisation. Simultaneously, they typically have a much lower dimensionality. While a one-hot-encoded vector has the size of the vocabulary (in our case, 29841), our GloVe representations only have a length of 100 numbers.

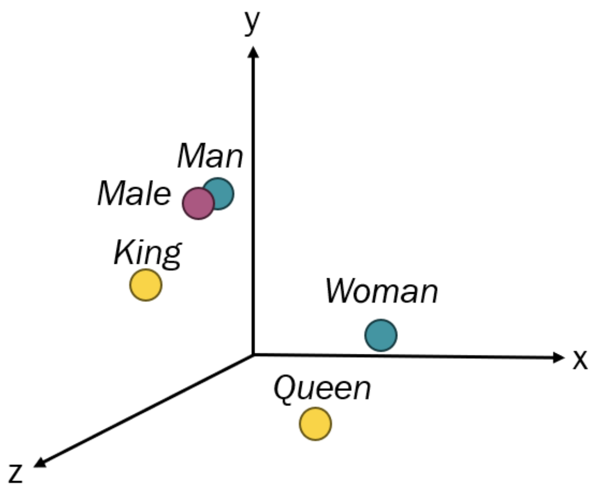


Figure 1: Distinct word embeddings within a highly compressed three-dimensional space.

1.2 b)

T-SNE serves as a dimensionality reduction technique [4] that reduces the number of parameters in individual word-embeddings from 100 to just 2 parameters when applied on it. This enables us to visualise the embedding space depicted in Figure 2. The phenomenon described in Section 1.1 can also be clearly observed. The words *film* and *movie* are considered to be semantically identical and therefore share nearly identical vectors with an approximate value of $(4.5, -6.2)$. It is noteable that *good* $(-1.8, -5.8)$ and *bad* $(-1.7, -6.1)$ are very close to each other, despite their antonymous nature. One might hypothesize that their antonymous nature would result in an equivalent disparity in word embeddings. However, their positioning appears plausible when considering the methodology employed in obtaining these embeddings. As previously stated, the *GloVe* algorithm utilises co-occurrence matrices, taking into account the words surrounding the given word. This implies that both *good* and *bad* both appear in similar contexts, regardless of whether they are positive or negative. This is logical given that they are both evaluative adjectives and sentiment is typically a determining factor in the surrounding words (e.g. in "The movie was x", both *good* and *bad* would be equally suitable for x).

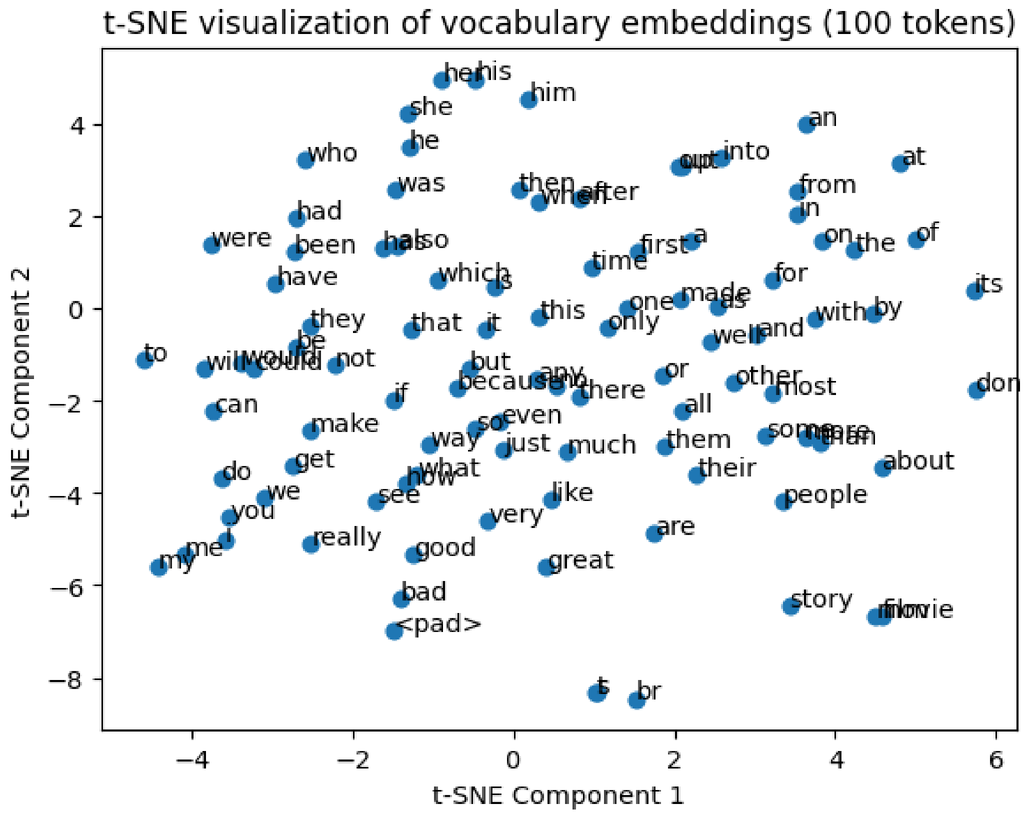


Figure 2: Word-embeddings reduced to two dimensions with the *t-SNE algorithm.*

Figure 3 illustrates a different distribution of words when the PCA algorithm is used for dimensionality reduction. When compared to the results from the t-SNE algorithm in Figure 2, it becomes clear that the PCA performs worse. The t-SNE synonymously appearing words *movie* (-0.2, 0) and *film* (0.8, 3) exhibit a significant offset in PCA, which is much larger as the previously mentioned difference between *good* and *bad*. This is just one of many examples where PCA performs worse. One factor that may contribute to this is that PCA operates by finding the orthogonal axes (principal components) that capture the most variance in the data [1]. While this linear transformation is highly effective for certain types of data, word embeddings often exhibit complex and non-linear relationships, which can be more effectively captured by t-SNE [4]. This is because t-SNE focuses on preserving local relationships between data points, rather than global relationships such as PCA [2] [4]. Consequently, it is more likely for word embeddings to retain the same neighbours after dimensionality reduction when using t-SNE than with PCA.

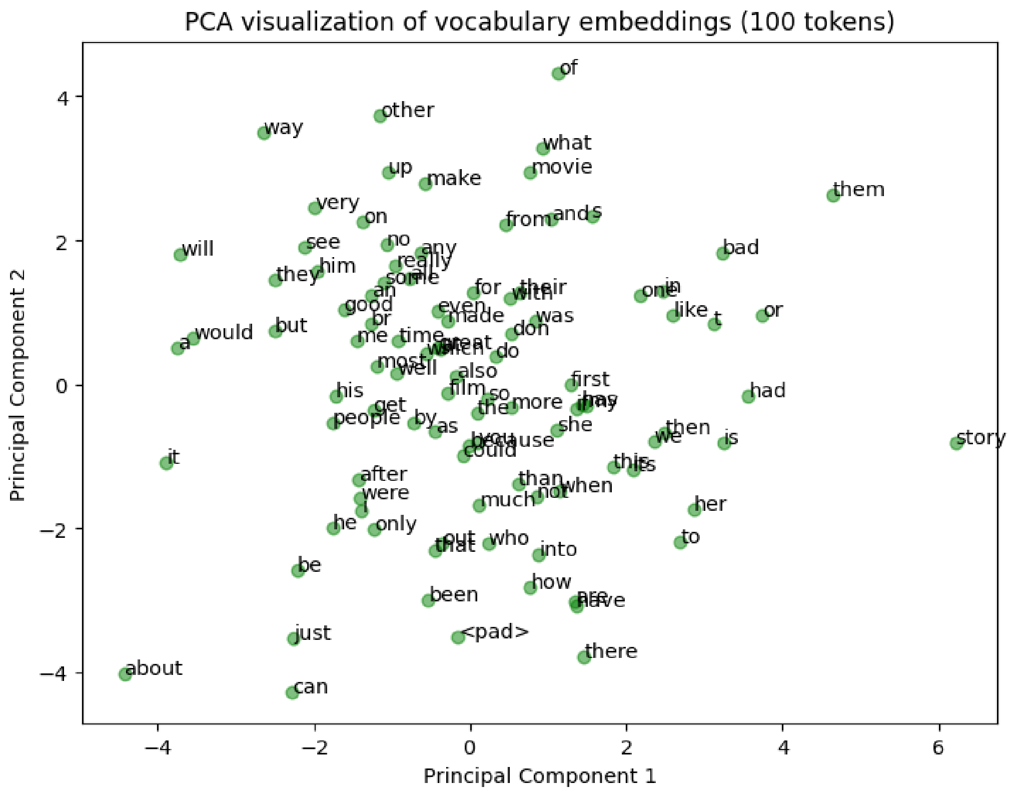
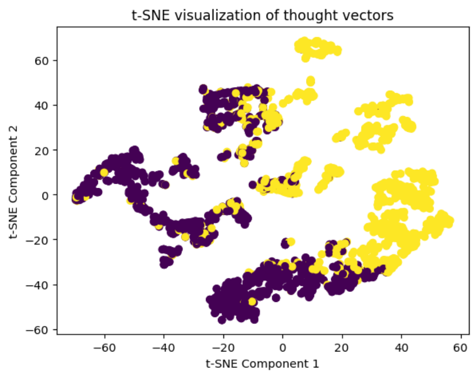


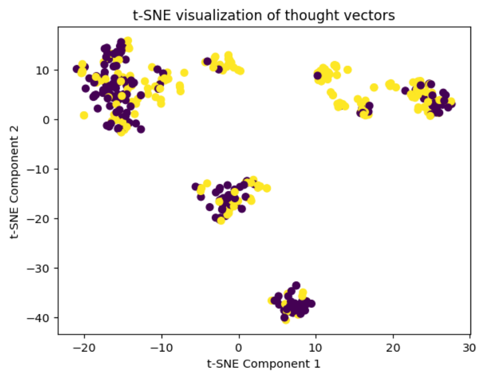
Figure 3: Word-embeddings reduced to two dimensions with the *PCA algorithm*.

1.3 c)

Figure 4 illustrates the final hidden vector representation of the RNN on the training 4(a) and validation 4(b) datasets. The representation of the final hidden vector is chosen because it encapsulates all previous hidden states and is therefore the most expressive. As in the previous tasks, it underwent *t-SNE* to achieve a humanly interpretable visualisation. Figure 4(a) illustrates that a recognisable separation between positively and negatively labelled embeddings in the training data has been achieved. However, there is a significant overlap between both quantities. The effect of the t-SNE algorithm is evident in the validation data, as illustrated in Figure 4(b). While the cluster in the interval $([4, 10], [-30, -40])$ achieves a relatively high purity for one class, majority of the other clusters (e.g. $([-20, -10], [30, -5]))$ exhibit a high variety in classes.



(a) Training data

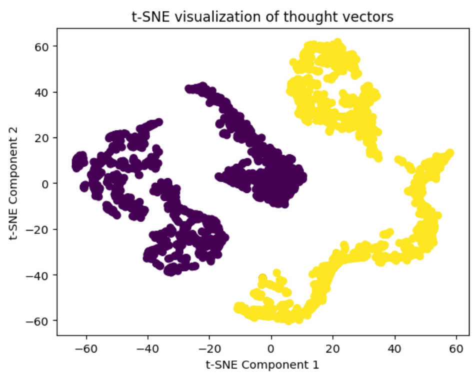


(b) Validation data

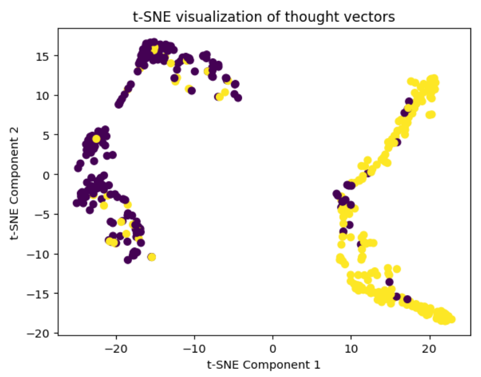
Figure 4: Embeddings of the thought vectors of the respective data subset

1.4 d)

Figure 5 illustrates the final hidden vector representation of the model on the training 4(a) and validation 4(b) datasets when using pre-trained *GloVe* embeddings and fine-tuning them on the task-specific dataset. It can be observed that the seperation achieved is perfect in the training data and very good in the validation data. A comparison of these results with those presented in Figure 4 suggests that the use of pre-trained embeddings and subsequent fine-tuning with task specific data is a more effective approach than learning the embeddings from scratch for this task. This discrepancy may be attributed to the fact that the *GloVe* embeddings have been trained on a vast corpus of text [6] and therefore achieve a more precise word representation than the RNN, which has been trained on a limited subset. Furthermore, the training data can be fully utilised to fine-tune on the task of sentiment analysis, providing a significant advantage due to the fact that the embeddings have already been learned.



(a) Training data

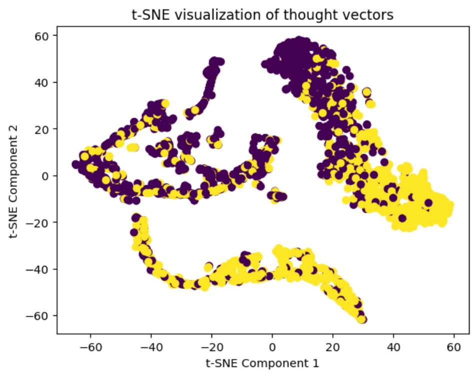


(b) Validation data

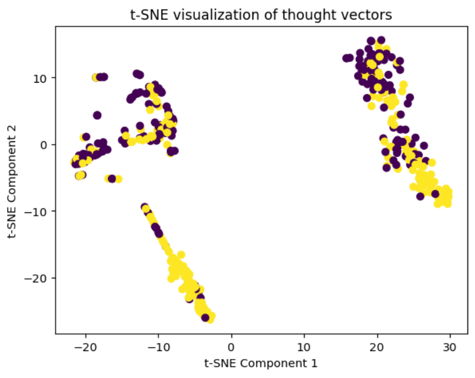
Figure 5: Pre-Training and Fine-Tuning

1.5 e)

Figure 5 illustrates the final hidden vector representation of the RNN on the training 4(a) and validation 4(b) datasets when using pre-trained *GloVe* embeddings but not updating them during training. A comparison of the results in Figures 5(a) and 5(b) with those of Figures 6(a) and 6(b) reveals that fixing the embeddings to their initial state and not adjusting them to the task-specific data results in a worse performance. It is hypothesised that this phenomenon is the result of multiple factors. Firstly, the GloVe embeddings have been trained on a diverse range of text data, which capture general linguistic patterns. Fine-tuning allows the model to adapt the embeddings specifically to movie reviews data,thereby leading to enhanced generalisation. Additionally, fine-tuned embeddings are capable of more effectively capturing contextual nuances in the dataset, such as the effect of negations, word-sense disambiguation and domain-specific language.



(a) Training data



(b) Validation data

Figure 6: Pre-Training without Fine-Tuning

2 Exploration

2.1 a)

In order to facilitate interpretability, the results of all possible combinations of dropout probability (0, 0.4 or 0.8), cell type (RNN, GRU or LSTM) and mode (unidirectional or bidirectional) are extracted from both the models' best performing training epoch and the last training epoch on the validation dataset and presented in Table A. It seems logical to include both, as in practice, the best performing training epoch is the most relevant and will be used as the final model state. However, comparison to the final training epoch allows for the derivation of valuable information. This is particularly evident in the unidirectional GRU with a dropout rate of 0.4 (9). Although the GRU with the lowest validation accuracy at the final epoch (9.2) exhibits the poorest GRU performance in the entire table, its second epoch (9.1) is the most successful model. This is also evident in Figure 7, which also demonstrates that model 9.2 has a statically increasing training accuracy, while declining in validation accuracy. This leads to the assumption that model 9 exhibits overfitting tendencies, resulting in a poorer generalisation performance on the training data. This is also supported by the considerable discrepancy between a low training loss of 0.0065 and a high validation loss of 1.1038 in epoch 10 (9.2). This phenomenon of overfitting is particularly evident in GRUs and LSTMs which have a Dropout of 0 (3, 4, 5, 6). These models achieve a near-perfect validation accuracy, yet perform much worse on the validation data.

Interestingly, GRU architectures appear to generalise more effectively than LSTM architectures, despite recalling a similar train accuracy. While research indicates that both architectures perform equally well [3], a straightforward explanation for this phenomenon is the difference in complexity presented in each architecture. LSTMs rely on a cell state, whereas GRUs utilise a hidden state to transform information, resulting in fewer parameters and complexity. This leads to enhanced generalisation and reduced risk of overfitting, which explains the lower discrepancy in train accuracy and validation accuracy. This argument is reinforced by the observation that a notable increase of 0.8 in dropout leads to similar or even better validation accuracy in the GRU (comparing 3,4 with 15,16) but significantly worse results in the LSTM (comparing 5,6 with 17,18). It appears that the relatively high accuracy of the LSTM without dropout (5,6) was achieved through overfitting. However, once the model is forced to generalise by a high dropout (17,18), it is unable to achieve similar results. This becomes evident when examining Figure 8, which illustrates the discrepancy in accuracy between the GRU and LSTM with a dropout of 0.8. While the GRU is capable of rapidly acquiring knowledge from the training data, the RNN requires numerous epochs to achieve a marginal increase in the training accuracy. It is noteworthy that both the GRU and LSTM achieve an equally high validation accuracy when given a dropout of 0.4.

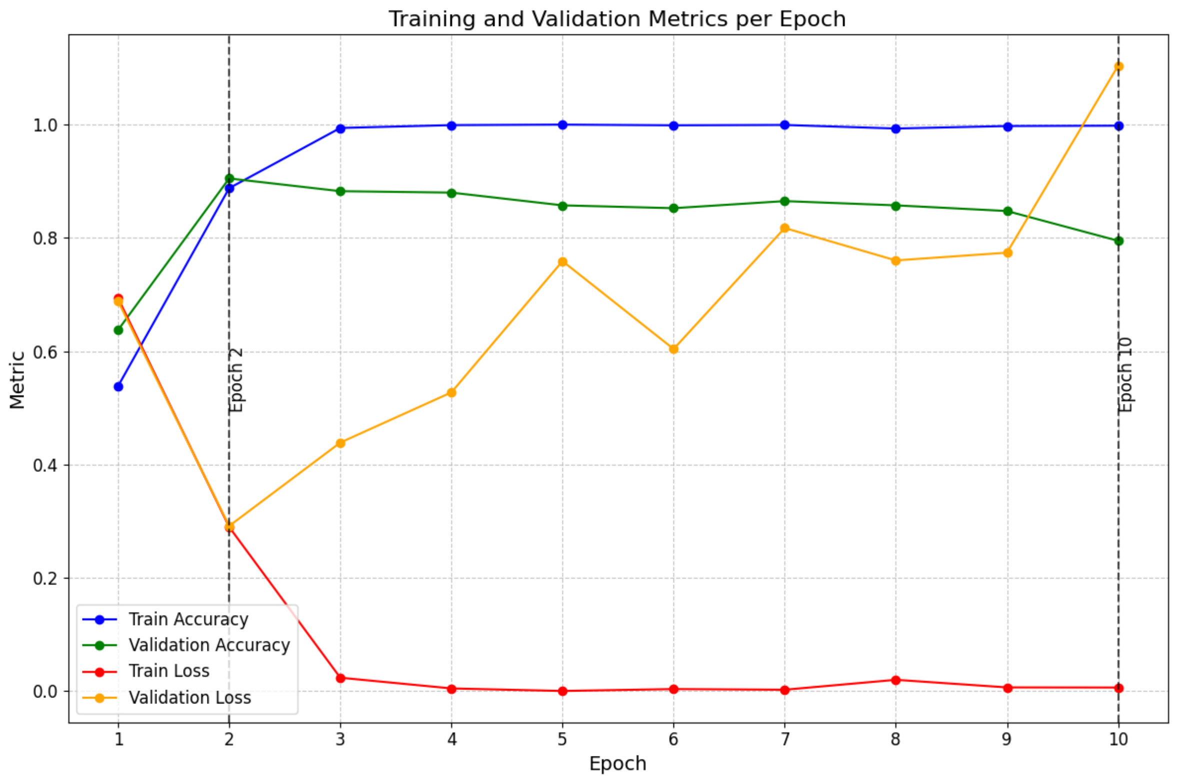


Figure 7: Line plot showing the difference in epochs and metric for model 9.2.

Notably, the GRUs achieve their optimal performance in an earlier iteration (epoch 2 in 9.1 and epoch 3 in 10.1), while the LSTMs reach their peak in a later phase (epoch 10 in 11.1 and epoch 9 in 12.1). Consequently, the validation loss at the final iteration is considerably lower in the GRU. Figure 9 illustrates this discrepancy. While the optimal choice for this particular dropout value between the two models remains unclear, it appears that this dropout value represents an optimal balance between regularisation and enabling the LSTM to learn effectively. A relevant difference in mode is not observable. The average validation accuracy in unidirectional models is 0.7206, while the average validation accuracy in bidirectional models is 0.7278. Despite this, bidirectional models collectively achieve a much lower training loss, indicating that the models' ability to capture both past and future context helps the model to learn the training data better, but not to generalise beyond that.

Finally, the RNN architecture performs significantly worse than the GRU and LSTM architectures. They achieve a performance close to the 50% mark, which represents the baseline for random selection in the context of binary classification problems. This suggests that the reviews in the training data are either too complex or too large in size (the longest review has 709 tokens). Consequently, this most likely lead to the vanishing/exploding gradient problem. The GRU and LSTM both possess the capacity to identify and prioritise relevant information, thereby enabling them to learn the training data in a more effective manner and avoiding the problems [7].

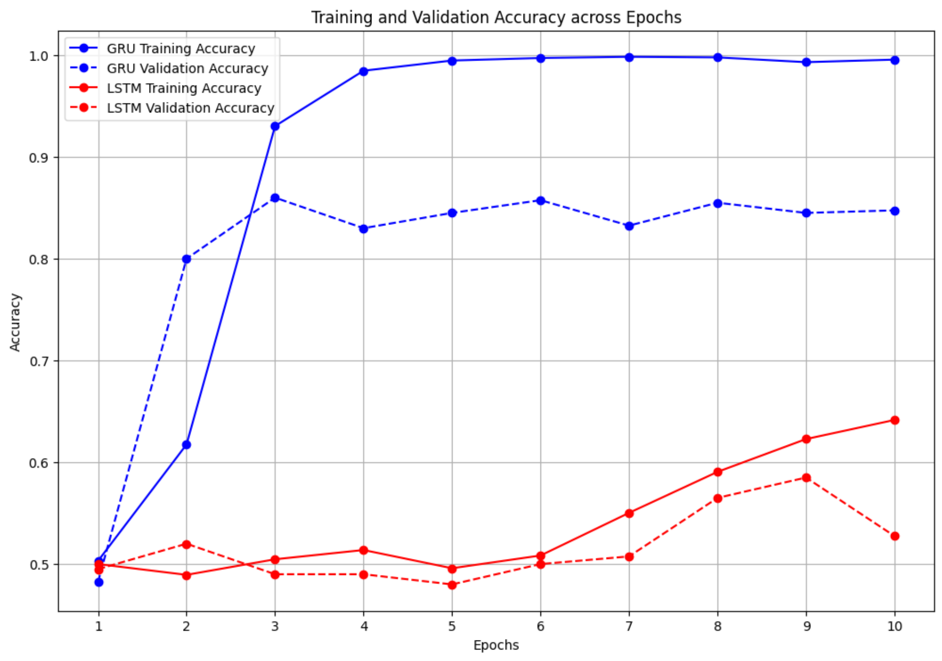


Figure 8: Line plot showing the difference in metrics between models 17.2 (LSTM) and 15.2 (GRU).

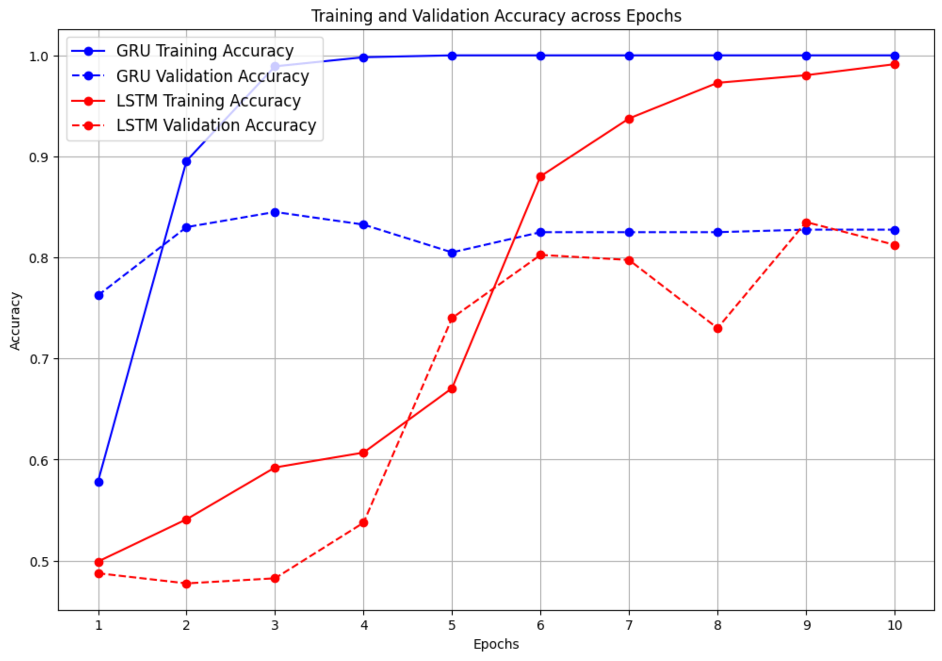


Figure 9: Line plot showing the difference in metrics between models 10.2 (LSTM) and 12.2 (GRU).

2.2 b)

In the course of our investigation, we identified a model configuration that yielded superior performance, with a validation accuracy of 0.9050 and a validation loss of 0.2916. The parameters of this model are presented in Table 1.

Parameter	Value
Dropout	0.4
RNN cell type	GRU
RNN direction	Unidirectional

Table 1: Model Parameters.

This peak performance was observed during the second epoch of the training process. To further optimise this model, we kept the above parameters constant and conducted an exploration to identify the optimal hyperparameters. The hyperparameters we considered were as shown in the Table 2.

Hyperparameter	Values
`hidden_dim`	50, 100
`num_layers`	1, 2
`n_epochs_values`	5

Table 2: Hyperparameters for the model.

`hidden_dim` is the number of features in the hidden state, `num_layers` is the number of recurrent layers, and `n_epochs_values` is the number of epochs for training respectively.

As in our previous research the following interesting fact was revealed: the model tends to reach its minimum validation loss (Figure 10) and maximum validation accuracy (Figure 11) during the second epoch. Upon visualising the model's validation accuracy and loss from the second epoch which is showed in Figure 12 we identified two sets of hyperparameters that yielded optimal results which are reflected in the Table 3.

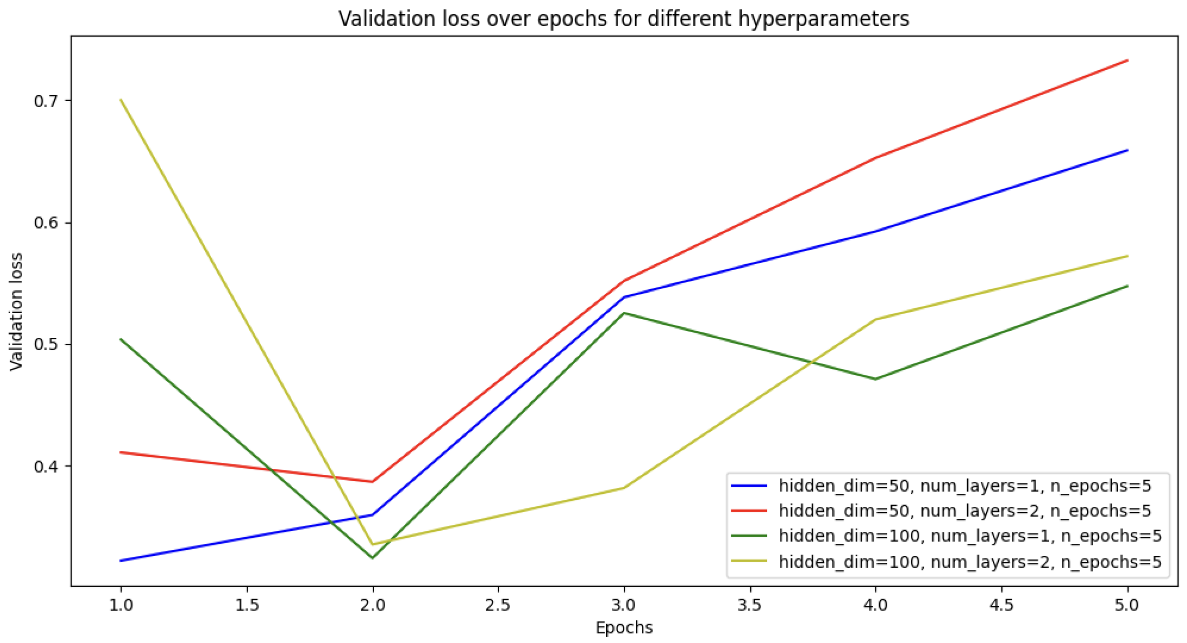


Figure 10: Line plot showing the validation loss differences over epochs

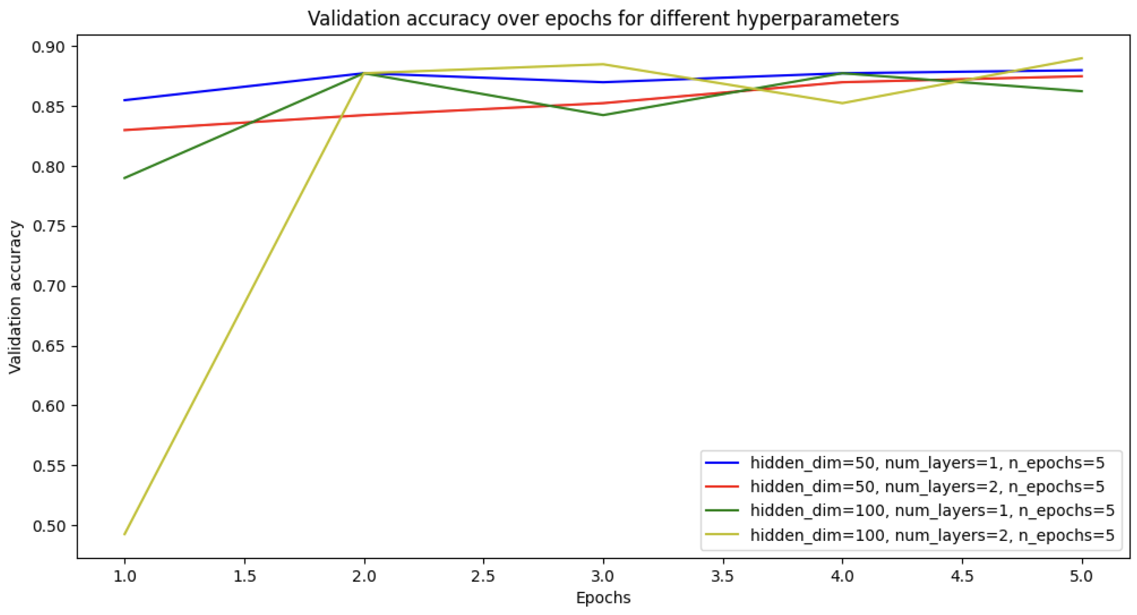


Figure 11: Line plot showing the validation accuracy differences over epochs

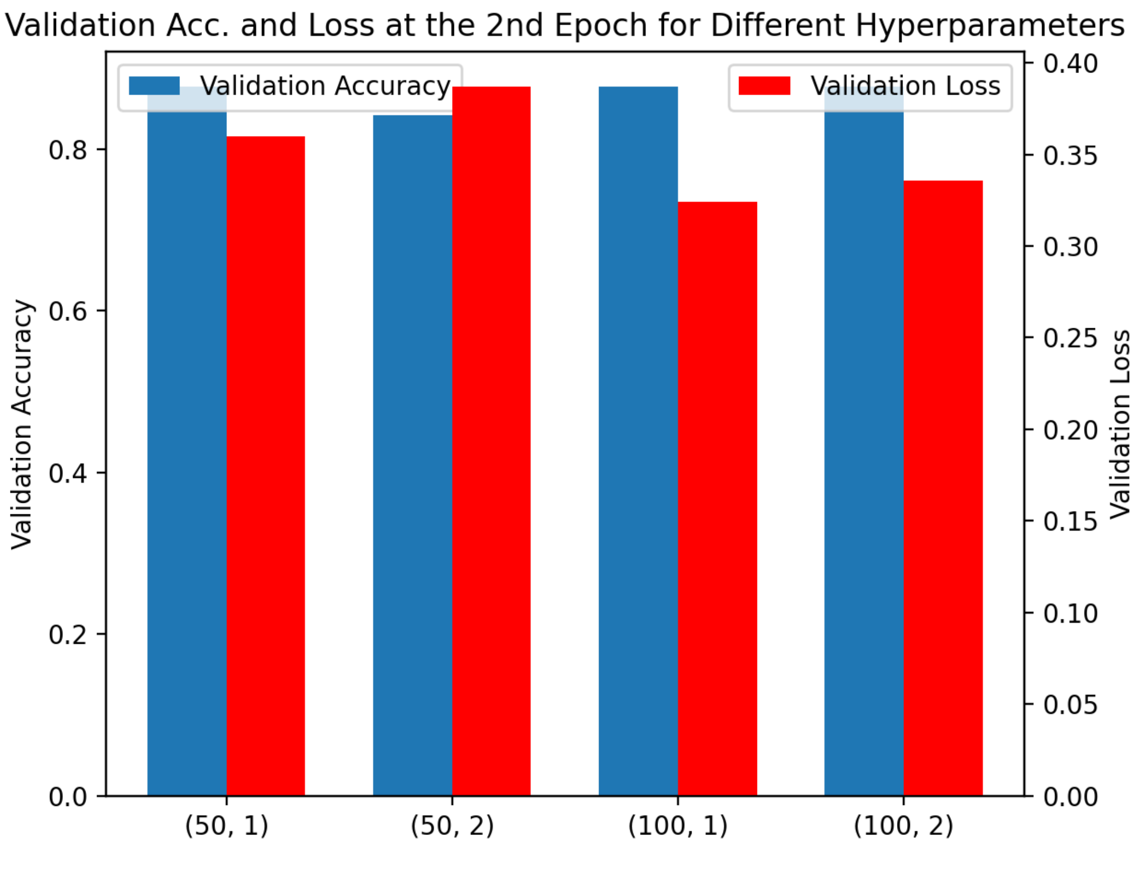


Figure 12: Comparison of Accuracy and Loss for Hyperparameters

Set	`hidden_dim`	`num_layers`	**Accuracy**	**Loss**
1	100	1	0.8775	0.3243
2	100	2	0.8775	0.3355

Table 3: Results for different sets of hyperparameters.

Given the identical accuracy for both sets and the negligible difference in loss values, we had to consider additional factors. We noted a moderate `n_epoch`=2 value and a relatively high `dropout`=0.4 parameter, which introduces a significant level of regularization. Taking these factors into account, we selected the second set of hyperparameters (`hidden_dim`=100, `num_layers`=2). This configuration allows the second layer of GRUs to capture more complex data patterns, effectively counterbalancing the level of regularization introduced by the `n_epoch` and `dropout` hyperparameters. Finally, as can be seen in the table 4, the final set of hyperparameters was obtained.

Hyperparameter	**Value**
`hidden_dim`	100
`num_layers`	2
`n_epochs`	2
`dropout`	0.4
`bidirectional`	False
`cell_type`	'gru'

Table 4: Final hyperparameters for the model

By applying this set of hyperparameters during model estimation on the test data, the expected results were obtained (Tables 5 and 6), which show an excellent level of generalization, as evidenced by the results on the test data being even higher than on the validation data. In addition, a high level of learning is very well visible.

Epoch	**Train Accuracy**	**Train Loss**	**Val Accuracy**	**Val Loss**
1	0.5900	0.6332	0.8175	0.4341
2	0.9353	0.1697	0.8625	0.3648

Table 5: Training and validation results for each epoch.

Test Accuracy	**Test Loss**
0.9025	0.2814

Table 6: Test results

In the light of the aforementioned, the results are due to the good balance of the hyperparameters that were identified during the optimization phase of the model. Although relatively high dropout value (`dropout`=0.4) and a small number of epochs in the training process (`n_epochs` = 2) prevent overfitting, the second layer of blocks (`num_layers`=2) and the size of the hidden layer (`hidden_dim`=100) allow the model to catch complex dependencies between the data, organizing the balance between the so-called data regularization and overfitting. This was the result of high accuracy and low loss value on test data, surpassing even the performance on validation data.

A Evaluation table

id	Dropout	Cell	Mode	Train Acc.	Train Loss	Val Acc.	Val Loss	Epoch
1.1	0	RNN	Uni	0.6434	0.5669	**0.5425**	0.8228	8
1.2				0.6725	0.5062	**0.5000**	0.9907	10
2.1	0	RNN	Bi	0.5956	0.6530	**0.5350**	0.6974	5
2.2				0.6094	0.6001	**0.4775**	0.7557	10
3.1	0	GRU	Uni	0.9053	0.2511	**0.8650**	0.2973	2
3.2				1.0000	0.0000	**0.8575**	0.8993	10
4.1	0	GRU	Bi	0.9681	0.0965	**0.8575**	0.4100	4
4.2				1.0000	0.0000	**0.8525**	1.0208	10
5.1	0	LSTM	Uni	0.9791	0.0654	**0.7950**	0.6579	7
5.2				0.9947	0.0131	**0.7725**	0.9023	10
6.1	0	LSTM	Bi	0.9494	0.1424	**0.7900**	0.6476	9
6.2				0.9812	0.0601	**0.7750**	0.8637	10
7.1	0.4	RNN	Uni	0.6378	0.5750	**0.5475**	0.8477	7
7.2				0.6891	0.4663	**0.5025**	0.9573	10
8.1	0.4	RNN	Bi	0.5887	0.6619	**0.5300**	0.7248	4
8.2				0.6809	0.5035	**0.4975**	0.9562	10
9.1	0.4	GRU	Uni	0.8878	0.2898	**0.9050**	0.2916	2
9.2				0.9981	0.0065	**0.7950**	1.1038	10
10.1	0.4	GRU	Bi	0.9891	0.0357	**0.8450**	0.6082	3
10.2				1.0000	0.0000	**0.8275**	1.1752	10
11.1	0.4	LSTM	Uni	0.9897	0.0469	**0.8575**	0.6696	10
11.2				0.9897	0.0469	**0.8575**	0.6696	10
12.1	0.4	LSTM	Bi	0.9803	0.0744	**0.8350**	0.6508	9
12.2				0.9912	0.0353	**0.8125**	0.6955	10
13.1	0.8	RNN	Uni	0.4891	0.7167	**0.5275**	0.6945	3
13.2				0.5022	0.7036	**0.5225**	0.6941	10
14.1	0.8	RNN	Bi	0.5553	0.6820	**0.5325**	0.7119	5
14.2				0.6069	0.6278	**0.4925**	0.7899	10
15.1	0.8	GRU	Uni	0.9956	0.1884	**0.8600**	0.3518	3
15.2				0.9956	0.0136	**0.8475**	0.7808	10
16.1	0.8	GRU	Bi	0.9953	0.0220	**0.8875**	0.3903	7
16.2				0.9969	0.0098	**0.8875**	0.5010	10
17.1	0.8	LSTM	Uni	0.6228	0.6114	**0.5850**	0.7594	9
17.2				0.6416	0.5660	**0.5275**	0.8274	10
18.1	0.8	LSTM	Bi	0.8550	0.3500	**0.7375**	0.6093	9
18.2				0.8741	0.3255	**0.6350**	0.8643	10

Table 7: Possible combinations between Dropout, Cell and Mode. Each architecture is displayed along its best performing epoch and last performing epoch.

References

[1] Hervé Abdi and Lynne J Williams. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics*, 2(4):433–459, 2010.

[2] Farzana Anowar, Samira Sadaoui, and Bassant Selim. Conceptual and empirical comparison of dimensionality reduction algorithms (pca, kpca, lda, mds, svd, lle, isomap, le, ica, t-sne). *Computer Science Review*, 40:100378, 2021.

[3] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[4] George C Linderman, Manas Rachh, Jeremy G Hoskins, Stefan Steinerberger, and Yuval Kluger. Efficient algorithms for t-distributed stochastic neighborhood embedding. *arXiv preprint arXiv:1712.09005*, 2017.

[5] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[6] Jeffrey Pennington, Richard Socher, and Christopher D Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[7] Farhad Mortezapour Shiri, Thinagaran Perumal, Norwati Mustapha, and Raihani Mohamed. A comprehensive overview and comparative analysis on deep learning models: Cnn, rnn, lstm, gru. *arXiv preprint arXiv:2305.17473*, 2023.