University of Athens
Department of Informatics and Telecommunications

Deep Learning for NLP

Student name: *Theodoros Diamantopoulos*
sdi: sdi1800266

Course: *Artificial Intelligence II (M138, M226, M262, M325)*
Semester: *Spring Semester 2025*

Contents

1. Introduction

In this project, GloVe word embeddings of varying dimensions are utilized to train a feedforward Neural Network for binary sentiment classification on English-language tweets. The main objective is to evaluate the effectiveness of dense vector representations in capturing the semantic meaning and improving classification accuracy.
A variety of fine-tuning techniques are used, such as learning rate adjustment, different loss function choices, optimization algorithms and more. The model's performance is assessed by standard evaluation metrics such as accuracy, precision, recall and F1-score.

2. Data processing and Vectorization

2.1. Pre-processing

Pre-processing is a critical step in any machine learning and AI application as it prepares raw data for model training and ensures that it is clean, structured, and ready for analysis. In this project, various data cleaning techniques were implemented to process the text data and enhance the model's performance.

The following methods were applied during the pre-processing phase:

- **HTML Entity Decoding**: The text was decoded to remove HTML entities. This step ensures that any encoded HTML symbols, such as `&amp;` for & are converted back to their original characters.

- **Lowercasing**: The entire text was converted to lowercase. This simplifies the data by ensuring that words like `Apple` and `apple` are treated as the same token, which helps in reducing redundancy and improving consistency.

- **Email Address Removal**: All email addresses were replaced with a generic placeholder, `xxx@email.com`. This was done to anonymize any personal information and avoid any potential bias that might arise from email addresses.

- **Twitter Usernames Removal**: Usernames (e.g., `@username`) were removed from the text. This is particularly important in tweet datasets where usernames don't contribute to the meaning or sentiment of the message.

- **URL Removal**: Any URLs in the text were replaced with a placeholder (`httpxxx`). Since URLs don't contribute to the sentiment analysis, removing them ensures that they do not affect the model's performance.

- **Punctuation Removal**: All punctuation marks, such as periods, commas, and question marks, were removed. This step ensures that punctuation doesn't interfere with tokenization and word analysis, as it generally doesn't carry meaningful information in this context.

- **Number Removal**: Numbers were also removed from the text. In most natural language processing (NLP) tasks, numerical values are irrelevant unless the task specifically requires them.

- **Non-Alphanumeric Characters and Emoji Removal**: Any non-alphanumeric characters, including emojis, were eliminated from the text to reduce noise in the data. This is essential in datasets containing a mix of standard text and special characters.

- **Whitespace Normalization**: Extra spaces and tabs were removed, ensuring that the text is clean and properly formatted. This step improves the accuracy of tokenization and ensures consistency in text representation.

- **Repetitive Character Removal**: Any characters that appeared three or more times consecutively (e.g., `loooool`) were reduced to just two occurrences (e.g., `lol`). This helps in normalizing exaggerated expressions often seen in informal text.

- **Tokenization**: The text was tokenized using the `TweetTokenizer` from the NLTK library. This tokenizer is particularly suited for handling social media text, as it preserves important features like hashtags and mentions while splitting the text into individual words or tokens.

- **Stopword Removal**: Common stopwords like `the`, `and`, and `is` were removed from the text. These words are usually insignificant in text classification tasks as they do not contribute meaningful information.

- **Lemmatization**: The text was lemmatized using the `WordNetLemmatizer` from NLTK. This step reduces words to their base or root form (e.g., `running` becomes `run`), which helps to reduce dimensionality and improve the model's performance.

After applying these pre-processing steps, the data was transformed into a cleaned and structured format suitable for training machine learning models. The pre-processed text was then stored in a new column (`preprocessed_text`) for further analysis and modeling.

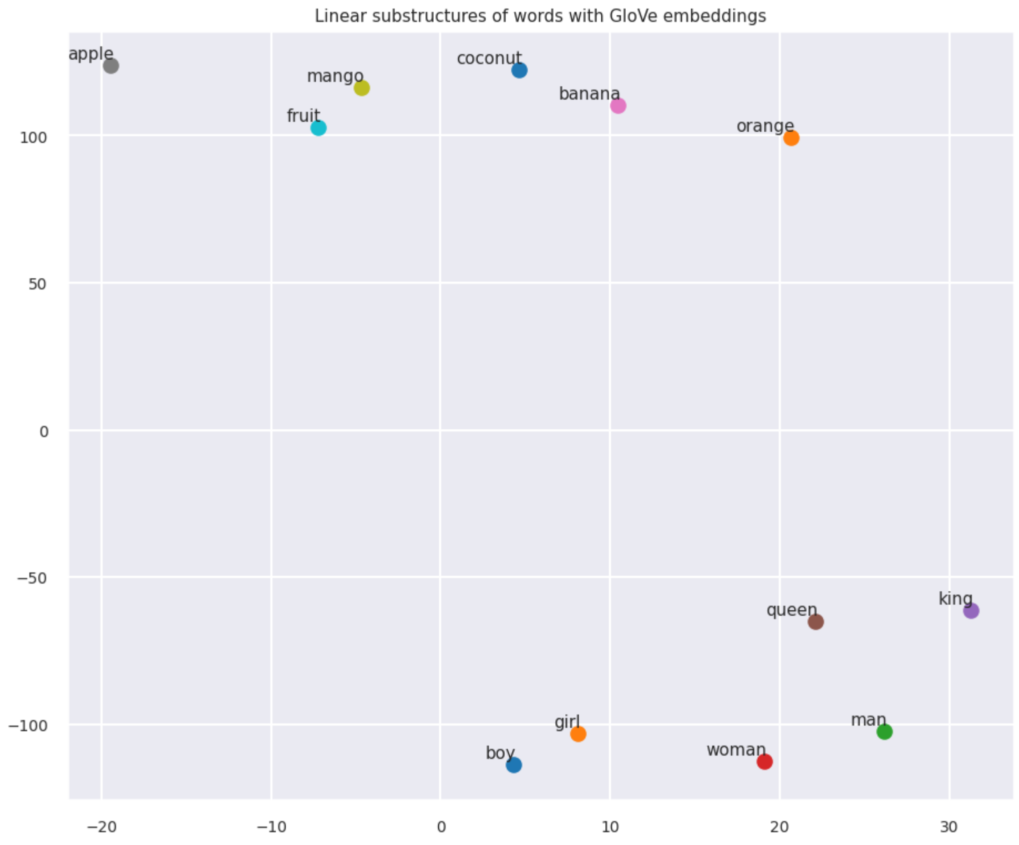
2.2. Vectorization

For this project, text vectorization was performed using pre-trained GloVe word embeddings. GloVe(Global Vectors for Word Representation) provides vector representation of words, capturing semantic relationships based on global word co-occurrence statistics. These embeddings do not require training from scratch and serve as the foundations of the model. Below is an example of the Linear substructures of words with GloVe Embeddings.

To convert each tweet into a fixed-length input suitable for neural network training, the model uses the mean of the word vectors corresponding to the tokens from the tweets. This returns a single dense vector per tweet by averaging the embeddings of all known words while ignoring the words that do not exist in the embedding vocabulary and having the tweets with no valid token represented as 'zero vectors'.

The resulting vectors for each dataset were converted into PyTorch tensors and the labels were reshaped to match the binary classification format.

DataLoaders were then constructed to implement batch training and evaluation within the PyTorch framework.



3. Algorithms and Experiments

Experiments: In this section a series of experiments are conducted in order to explore and fine-tune key hyperparameters of the Neural Network with the purpose of identifying the most effective configuration for sentiment classification.

The hyperparameters that will be tuned are:

1. **Embedding Size and Training Epochs:** The dimensionality of the input embeddings influences how much semantic information is encoded in the input vectors. Higher-dimensional embeddings may capture richer linguistic patterns, while lower-dimensional ones can be faster but less expressive. Similarly, the number of training epochs determines how long the model trains — more epochs may allow deeper learning, but also risk overfitting if not controlled properly.

2. **Learning Rate:** A scalar that determines how much the model weights are updated during training. A learning rate that is too high can cause the model to diverge, while one that is too low can result in slow convergence or getting stuck in local minima.

3. **Optimizer:** The algorithm used to update the weights of the neural network based on the gradients computed from the loss function. Different optimizers' results may vary, each with their own behavior in terms of convergence speed and stability.

4. **Loss Function:** A function that quantifies the difference between the model's predictions and the actual labels. It serves as the objective the model tries to minimize.

5. **Number and Size of Hidden Layer(s):** Defines the architecture of the neural network. More layers and larger layers can help the model learn more complex functions, but they also increase computational cost and the risk of overfitting. Choosing the right architecture is crucial for balancing performance and efficiency.

3.1. Baseline Model

First, a baseline model was created on the 50-dimensional GloVe embeddings and the hyperparameters provided by the course instructor(*learning rate = 1e-4, Hidden Layers: H1 = 128, H2 = 64, H3 = 32, loss_func = nn.BCELoss(reduction='sum'), optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)*).

Table 1: Training and Validation Metrics per Epoch

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Validation F1
1	44.0929	43.8364	0.6141	0.6138	0.5997
2	43.3260	42.4152	0.6437	0.6444	0.6442
3	41.0594	39.6873	0.6588	0.6665	0.6605
4	39.3136	38.8500	0.6672	0.6686	0.6685
5	38.8892	38.6050	0.6697	0.6727	0.6722
6	38.7362	38.4896	0.6711	0.6744	0.6738
7	38.6415	38.4099	0.6719	0.6754	0.6747
8	38.5733	38.3703	0.6727	0.6760	0.6760
9	38.5089	38.2866	0.6732	0.6770	0.6769
10	38.4453	38.2387	0.6743	0.6777	0.6777

Below are the plots for **Training Validation Loss over Epochs, Train and Validation Accuracy over Epochs, Validation F1 Score over Epochs, Confusion Matrix, Receiver Operating Characteristic(ROC) Curve and Precision Recall Curve** to help visualize the condition during the starting point of the model.

- **Training and Validation Loss over Epochs:** Illustrates how the model's loss (error) changes during training and evaluation across epochs. It helps identify underfitting (consistently high loss) or overfitting (decreasing training loss with increasing validation loss).

Table 2: Classification Report on Validation Set

Class	Precision	Recall	F1-score	Support
Negative	0.67	0.69	0.68	21197
Positive	0.68	0.67	0.68	21199
Accuracy			0.68	
Macro avg	0.68	0.68	0.68	42396
Weighted avg	0.68	0.68	0.68	42396

Best Validation F1 Score: 0.6777 in epoch 10

- **Training and Validation Accuracy over Epochs:** Tracks the model's classification performance on both the training and validation sets. A significant gap between training and validation accuracy may indicate overfitting.
- **Validation F1 Score over Epochs:** Shows the evolution of the F1 score on the validation set. This metric balances precision and recall, making it especially useful for imbalanced datasets.
- **Confusion Matrix:** Summarizes the model's performance by showing counts of true positives, true negatives, false positives, and false negatives. It offers insight into the types of errors made by the classifier.
- **Receiver Operating Characteristic (ROC) Curve:** Plots the true positive rate against the false positive rate across different thresholds. The area under the curve (AUC) reflects the model's ability to distinguish between classes.
- **Precision–Recall Curve:** Displays the trade-off between precision and recall for various classification thresholds. It is particularly informative when evaluating models on datasets with class imbalance.

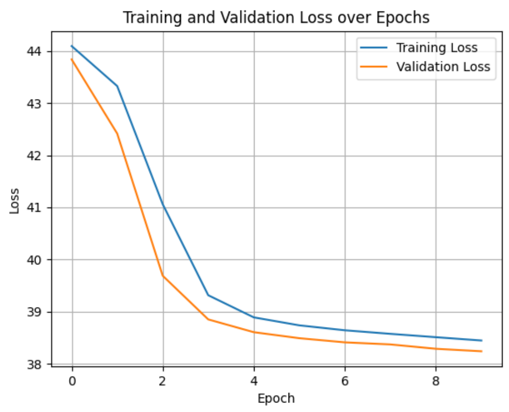


Figure 1: Training and Validation Loss over Epochs

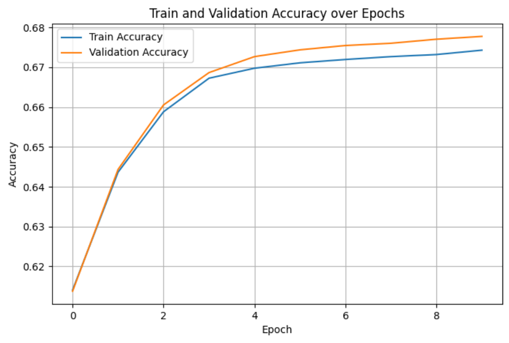


Figure 2: Training and Validation Accuracy over Epochs

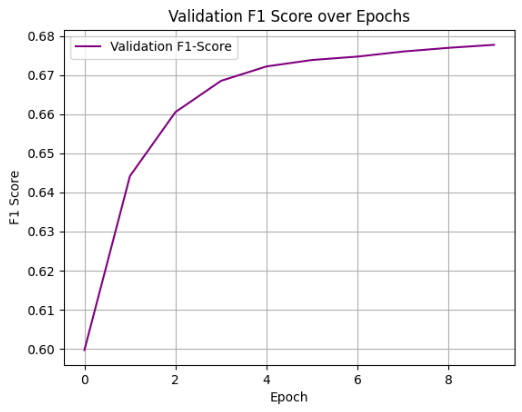


Figure 3: Validation F1 Score over Epochs

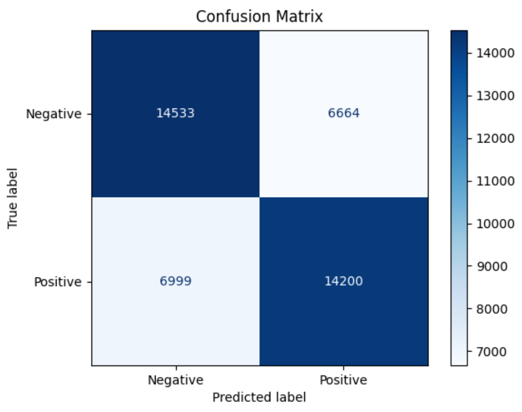


Figure 4: Confusion Matrix

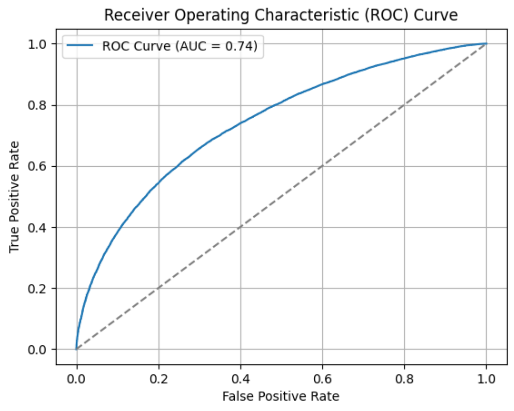


Figure 5: Receiver Operating Characteristic (ROC) Curve

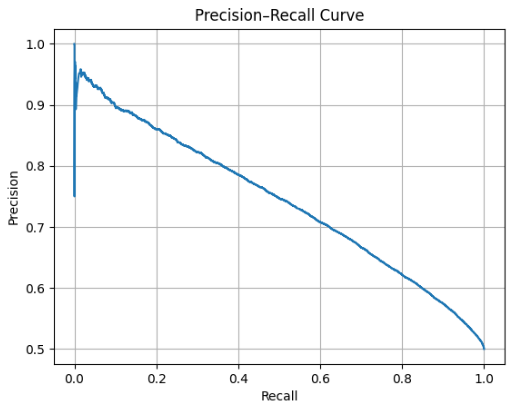


Figure 6: Precision–Recall Curve

Loss over epochs: The training and validation loss curves show a consistent downward trend, indicating effective learning. The validation loss closely follows the training loss with no significant divergence, suggesting the model is not overfitting and generalizes well to unseen data.

Accuracy over epochs: Both training and validation accuracy steadily increase and converge, with the validation accuracy slightly outperforming the training accuracy. This suggests good generalization without overfitting.

F1 score over epochs: The F1-score improves with each epoch and stabilizes toward the end, showing that the model becomes more balanced in handling precision and recall across classes.

Confusion matrix: The confusion matrix reveals a relatively balanced performance across the two classes. However, there are still a noticeable number of misclassifications, which might suggest further potential for improvement, possibly through more advanced modeling or preprocessing.

ROC curve: The ROC curve shows a fairly good separation capability, with an AUC of 0.74. This indicates the model performs better than random guessing and can reasonably distinguish between the classes.

Precision–Recall curve: The precision-recall curve confirms that the model maintains high precision at low recall levels and gradually trades off precision as recall increases. This performance is suitable for imbalanced datasets where maintaining high precision is crucial.

Comment: Although GLOVE offers multiple embedding sizes (50d, 100d, 200d, 300d), this report moves directly from the baseline 50-dimensional embeddings to the 300-dimensional ones. This decision is motivated by both established research and the theoretical principles presented in the course lectures, which emphasize that higher-dimensional embeddings typically result in better performance on natural language processing tasks.[?]

3.2. Increasing Dimensionality(from d=50 to d=300)

3.2.1. For 10 Epochs. In this experiment, the dimension-size of the GloVe embedding file was change from 50d to 300d. The rest of the model configuration remained unchanged(*learning rate = 1e-4, Hidden Layers: H1 = 128, H2 = 64, H3 = 32, loss_func = nn.BCELoss(reduction='sum'), optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)*).

Table 3: Training and Validation Metrics per Epoch

Epoch	Training Loss	Validation Loss	Training Accuracy	Validation Accuracy	Validation F1
1	44.3558	44.2948	0.5780	0.5819	0.5809
2	44.3023	44.2400	0.5875	0.5903	0.5538
3	44.2203	44.1094	0.6349	0.6409	0.6355
4	43.9694	43.6392	0.6404	0.6442	0.6313
5	42.7653	41.0696	0.6745	0.6769	0.6765
6	38.5494	36.3639	0.7049	0.7062	0.7058
7	35.8019	35.0895	0.7200	0.7229	0.7226
8	35.1127	34.6752	0.7250	0.7283	0.7281
9	34.8358	34.4544	0.7274	0.7300	0.7298
10	34.6823	34.3366	0.7286	0.7302	0.7302

Table 4: Classification Report on Validation Set

Class	Precision	Recall	F1-score	Support
Negative	0.73	0.74	0.73	21197
Positive	0.73	0.72	0.73	21199
Accuracy		0.73		
Macro avg	0.73	0.73	0.73	42396
Weighted avg	0.73	0.73	0.73	42396

Best Validation F1 Score: 0.7302 in epoch 10

The results indicate that inceasing the GloVe embeddings from 50-dimensional to 300-dimensional led to improvements in both training and validation performances.

- **Validation F1 Score:** The best F1 score on the validation set increased from 50d to 300d, from 0.6777 (50d) to 0.7302 (300d), showing that the higher-dimensional embeddings captured richer semantic information.
- **Validation Accuracy:** The highest validation accuracy improved when using the 300d model, from 0.6777 to 0.7302, indicating better generalization to unseen data.
- **Loss Reduction:** The final validation loss was noticeably lower in the 300d configuration, from 38.2387 to 34.3366, suggesting more effective convergence during training.

3.2.2. For 50 Epochs. In this experiment the Epochs used for training the Neural Network were change from 10 to 50 while the rest of the model configuration remained unchanged(*learning rate = 1e-4, Hidden Layers: H1 = 128, H2 = 64, H3 = 32, loss_func = nn.BCELoss(reduction='sum'), optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)*).

Depicted below are the Epochs 1-10, 28-35, 45-50:

Table 5: Training and Validation Metrics for Selected Epochs

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	44.3285	44.2540	0.5385	0.5408	0.4573
2	44.2305	44.1160	0.6147	0.6189	0.6182
3	43.9735	43.6328	0.6335	0.6346	0.6287
4	42.8203	41.2972	0.6688	0.6713	0.6699
5	38.9927	36.8721	0.6997	0.7035	0.7029
6	36.0351	35.4497	0.7156	0.7181	0.7162
7	35.2216	34.7463	0.7232	0.7266	0.7266
8	34.9254	34.5074	0.7256	0.7291	0.7288
9	34.7350	34.3638	0.7271	0.7305	0.7300
10	34.5761	34.1944	0.7295	0.7325	0.7323
28	33.6524	33.5916	0.7377	0.7382	0.7382
29	33.6405	33.5691	0.7378	0.7384	0.7384
30	33.6266	33.5535	0.7376	0.7390	0.7388
31	33.6211	33.5473	0.7380	0.7392	0.7391
32	33.6159	33.5451	0.7381	0.7391	0.7390
33	33.6103	33.5412	0.7384	0.7391	0.7390
34	33.6016	33.5435	0.7381	0.7390	0.7390
35	33.5983	33.5351	0.7382	0.7394	0.7393
45	33.5812	33.5253	0.7385	0.7393	0.7393
46	33.5805	33.5230	0.7384	0.7392	0.7392
47	33.5801	33.5223	0.7384	0.7393	0.7392
48	33.5800	33.5220	0.7385	0.7395	0.7394
49	33.5796	33.5219	0.7384	0.7394	0.7393
50	33.5790	33.5217	0.7385	0.7395	0.7394

Table 6: Classification Report on Validation Set

Class	Precision	Recall	F1-score	Support
Negative	0.73	0.76	0.74	21197
Positive	0.75	0.72	0.73	21199
Accuracy		0.74		
Macro avg	0.74	0.74	0.74	42396
Weighted avg	0.74	0.74	0.74	42396

Best Validation F1 Score: 0.7394 in epoch 48

The results indicate that the model demonstrates consistent performance improvements up to approximately epoch 30. Beyond this point, gains in validation accuracy and F1-score become minimal and tend to plateau. While the training loss continues to decrease slightly, the evaluation metrics suggest that the network has largely converged. As such, stopping earlier—around epoch 30 to 35—could be a practical choice to reduce computational cost without significantly impacting performance.

Comments: Although **Early Stopping** was implemented and included in the training loop to prevent overfitting and save resources, it was not activated during training. This implies that the model continued to improve, albeit gradually, until the final epoch. The steady (though small) performance gains beyond epoch 30 were sufficient to reset the early stopping counter, indicating that extended training still contributes positively, even if marginally, to overall performance.

3.3. Tuning learning rate

3.3.1. *learning rate = 5e-4.* **Learning Rate Experiment:** In this experiment, we modify the model's learning rate from *1e-4* to *5e-4*(Hidden Lay- ers: H1 = 128, H2 = 64, H3 = 32, loss_func = nn.BCELoss(reduction='sum'), optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate)). The objective is to assess whether a higher learning rate accelerates convergence or improves final performance.

Table 7: Training and Validation Metrics (Learning Rate = 5e-4)

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	43.9248	41.0667	0.6710	0.6718	0.6705
2	36.3687	34.8050	0.7223	0.7249	0.7231
3	35.0157	34.1131	0.7307	0.7328	0.7326
4	34.5064	33.7992	0.7346	0.7352	0.7347
5	34.0985	34.6412	0.7196	0.7199	0.7136
10	32.0363	32.2126	0.7588	0.7530	0.7529
15	31.2008	31.8490	0.7669	0.7552	0.7548
20	29.9284	31.5715	0.7775	0.7584	0.7582
25	29.4533	31.5209	0.7810	0.7601	0.7601
28	29.2472	31.4467	0.7835	0.7619	0.7618
30	29.2046	31.5149	0.7833	0.7600	0.7600

Table 8: Classification Report on Validation Set (Learning Rate = 5e-4)

Class	Precision	Recall	F1-score	Support
Negative	0.76	0.75	0.76	21197
Positive	0.76	0.77	0.76	21199
Accuracy			0.76	
Macro avg	0.76	0.76	0.76	42396
Weighted avg	0.76	0.76	0.76	42396

Best Validation F1 Score: 0.7618 in epoch 28

Comparison: Compared to the previous experiment with a learning rate of *1e-4*, the model trained with *5e-4* reached a higher F1 score (0.7618 vs 0.7394) and converged faster — achieving over 0.75 F1-score by epoch 10. This suggests that increasing the learning rate allowed for more aggressive optimization and better early improvements.

However, similar to the previous setting, the improvements after approximately epoch 28 are marginal, indicating that the model has converged. Thus, early stopping around that point could again be considered for efficiency without compromising accuracy.

Table 9: Training and Validation Metrics with Early Stopping

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	40.6158	35.0983	0.7200	0.7226	0.7211
2	35.3552	34.6353	0.7239	0.7252	0.7241
3	34.5550	33.6591	0.7357	0.7364	0.7362
4	34.1040	34.0110	0.7321	0.7332	0.7318
5	33.6411	33.0694	0.7456	0.7442	0.7440
6	33.2697	33.0657	0.7441	0.7420	0.7404
7	32.9403	32.7063	0.7523	0.7471	0.7469
8	32.6597	32.5026	0.7554	0.7491	0.7490
9	32.3711	32.1897	0.7598	0.7541	0.7540
10	32.0556	32.6208	0.7561	0.7477	0.7460
11	31.7875	32.3328	0.7608	0.7504	0.7490
12	31.5401	32.4619	0.7615	0.7498	0.7486
13	31.3467	32.2201	0.7661	0.7503	0.7492

Table 10: Classification Report on Validation Set (Early Stopping at Epoch 14)

Class	Precision	Recall	F1-score	Support
Negative	0.72	0.81	0.76	21197
Positive	0.78	0.69	0.73	21199
Accuracy		0.75		
Macro avg	0.75	0.75	0.75	42396
Weighted avg	0.75	0.75	0.75	42396

Best Validation F1 Score: 0.7540 in epoch 9

3.3.2. learning rate = 1e-3. **Comparison:** Setting the learning rate to 1e-3 helped the model converge more quickly in the early stages, reaching a validation F1 score above 0.75 by epoch 9. However, the performance gains leveled off shortly after, and the model began to show signs of instability. The best F1 score achieved (0.7540) was slightly lower than the one observed with 5e-4 (0.7618). Notably, early stopping was triggered at epoch 14, confirming that the model had stopped improving and reinforcing the idea that a higher learning rate can sometimes lead to quicker—but shallower—optimization. Overall, while 1e-3 offered faster results, the learning rate of 5e-4 provided a better balance between training efficiency and final performance.

Conclusion: Over the course of 30 epochs, the model achieved its best performance with a learning rate of 5e-4. This setting led to the highest validation F1 score and provided a stable training process with consistent improvements. In contrast, while 1e-3 converged more quickly, it plateaued earlier and resulted in slightly lower overall performance. Therefore,5e-4 is considered the optimal learning rate among those tested for this configuration.

3.4. Optimizer Tuning

Up to this point, the model has been trained using the `SGD` optimizer alongside the `BCELoss`-loss function. In this section, we investigate how alternative optimization algorithms affect the training process and final performance. Specifically, we test two popular choices: **Adam** and **AdamW**.
Both optimizers will be evaluated under the same conditions used in previous experiments(*GloVe 300d embeddings, the optimal learning rate of 5e-4, Hidden Layers: H1 = 128, H2 = 64, H3 = 32, and loss_func = nn.BCELoss(reduction='sum'*). The goal is to determine whether either optimizer provides measurable improvements in training efficiency or validation performance compared to `SGD`.

3.4.1. Adam Optimizer.

Adam is a widely used optimizer that combines the benefits of both momentum and adaptive learning rates, often resulting in faster and more stable convergence.

Table 11: Training and Validation Metrics with Early Stopping

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	35.2717	33.6580	0.7367	0.7366	0.7363
2	33.3131	32.9493	0.7502	0.7450	0.7446
3	32.2772	32.4651	0.7589	0.7515	0.7512
4	31.5963	32.2237	0.7647	0.7496	0.7495
5	31.0164	31.8638	0.7744	0.7549	0.7549
6	30.5405	31.9603	0.7782	0.7548	0.7548
7	30.0460	33.7342	0.7624	0.7417	0.7372
8	29.5998	31.9713	0.7887	0.7567	0.7567
9	29.1588	32.1375	0.7910	0.7542	0.7540
10	28.7238	32.6818	0.7925	0.7530	0.7527
11	28.3377	32.4933	0.7986	0.7546	0.7546
12	27.0770	32.8843	0.8095	0.7561	0.7561

Table 12: Classification Report on Validation Set (Early Stopping at Epoch 13)

Class	**Precision**	**Recall**	**F1-score**	**Support**
Negative	0.77	0.74	0.75	21197
Positive	0.75	0.77	0.76	21199
Accuracy			0.76	
Macro avg	0.76	0.76	0.76	42396
Weighted avg	0.76	0.76	0.76	42396

Best Validation F1 Score: 0.7567 in epoch 8

Comparison: Switching from `SGD` to `Adam` led to faster convergence and slightly better overall validation performance. The model reached its peak F1 score of 0.7567 at epoch 8, with early stopping triggered at epoch 13. Compared to previous experiments, Adam helped the model achieve higher validation accuracy and F1 earlier in training, suggesting more efficient learning dynamics. Precision and recall remained balanced across both classes, and the macro F1 score of 0.76 reflects strong generalization. Overall, Adam improved both training stability and validation metrics, marking it as a strong alternative to `SGD`.

*3.4.2. **AdamW Optimizer.*** **AdamW** is a variant that decouples weight decay from the gradient update, offering improved generalization in many deep learning tasks.

Table 13: Training and Validation Metrics with Early Stopping

Epoch	**Training Loss**	**Validation Loss**	**Train Acc**	**Val Acc**	**Val F1**
1	35.1711	33.5770	0.7389	0.7393	0.7393
2	33.2662	33.2960	0.7447	0.7392	0.7377
3	32.3165	33.1530	0.7506	0.7409	0.7393
4	31.6472	32.1522	0.7674	0.7538	0.7537
5	31.0835	31.9402	0.7708	0.7539	0.7539
6	30.6113	31.9063	0.7768	0.7547	0.7546
7	30.1636	32.1515	0.7805	0.7544	0.7544
8	29.7167	31.9096	0.7879	0.7568	0.7568
9	29.3176	31.9567	0.7895	0.7566	0.7562
10	28.9029	32.7708	0.7891	0.7511	0.7506
11	28.5704	32.4299	0.7974	0.7528	0.7521
12	28.1504	33.3252	0.7963	0.7489	0.7483
13	27.8049	32.6966	0.8073	0.7570	0.7569
14	27.4438	33.1048	0.8084	0.7524	0.7523
15	27.1156	33.4498	0.8062	0.7460	0.7454
16	25.7853	33.9786	0.8211	0.7527	0.7527
17	25.3812	34.8028	0.8219	0.7512	0.7507

Table 14: Classification Report on Validation Set (Early Stopping at Epoch 18)

Class	Precision	Recall	F1-score	Support
Negative	0.75	0.76	0.75	21197
Positive	0.75	0.75	0.75	21199
Accuracy			0.75	
Macro avg	0.75	0.75	0.75	42396
Weighted avg	0.75	0.75	0.75	42396

Best Validation F1 Score: 0.7569 in epoch 13

Comparison: The AdamW optimizer performed comparably to Adam, reaching a slightly higher validation F1 score of 0.7569 (vs. 0.7567) at epoch 13. While the improvement is marginal, AdamW maintained more stable performance over a longer stretch of epochs, whereas Adam peaked earlier and plateaued. This suggests that AdamW may offer slightly better regularization and generalization in this setting, making it a solid alternative to Adam when training deeper or more complex models.

Conclusion: Although `SGD` with a learning rate of 5×10^{-4} achieved the highest overall validation F1 score (0.7618), both `Adam` and `AdamW` attained nearly comparable performance in significantly fewer epochs. This demonstrates the efficiency and rapid convergence of adaptive optimizers, which can be particularly beneficial in time-sensitive or resource-limited training contexts.

Nevertheless, `AdamW`—while slightly outperforming `Adam`—is more susceptible to overfitting, especially on smaller or moderately sized datasets, due to its adaptive learning dynamics and decoupled weight decay. Since computational cost is not a primary concern in this study, and the dataset used is relatively balanced and of moderate size, stability and generalization are prioritized over training speed.
Considering these factors, and given its superior final performance, **SGD is selected as the optimizer for subsequent experiments**.

3.5. Loss Functions

In this section, different loss functions will be evaluated to determine their impact on model performance in the binary sentiment classification task.All loss functions are tested under the same model architecture(*GloVe 300d embeddings, the optimal learning rate of 5e-4, Hidden Layers: H1 = 128, H2 = 64, H3 = 32 and the SGE optimizer*) to ensure a fair comparison. The experiments so far have been run with the commonly used `BCELoss`, which assumes that the model outputs probabilities after a sigmoid activation.

3.5.1. MSELoss.

MSELoss loss function is explored, although it is typically used in regression tasks. It serves here as a point of comparison to assess how a non-log-likelihood-based loss behaves in a binary classification context.

Table 15: Training and Validation Metrics with Early Stopping

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	35.1711	33.5770	0.7389	0.7393	0.7393
2	33.2662	33.2960	0.7447	0.7392	0.7377
3	32.3165	33.1530	0.7506	0.7409	0.7393
4	31.6472	32.1522	0.7674	0.7538	0.7537
5	31.0835	31.9402	0.7708	0.7539	0.7539
6	30.6113	31.9063	0.7768	0.7547	0.7546
7	30.1636	32.1515	0.7805	0.7544	0.7544
8	29.7167	31.9096	0.7879	0.7568	0.7568
9	29.3176	31.9567	0.7895	0.7566	0.7562
10	28.9029	32.7708	0.7891	0.7511	0.7506
11	28.5704	32.4299	0.7974	0.7528	0.7521
12	28.1504	33.3252	0.7963	0.7489	0.7483
13	27.8049	32.6966	0.8073	0.7570	0.7569
14	27.4438	33.1048	0.8084	0.7524	0.7523
15	27.1156	33.4498	0.8062	0.7460	0.7454
16	25.7853	33.9786	0.8211	0.7527	0.7527
17	25.3812	34.8028	0.8219	0.7512	0.7507

Table 16: Classification Report on Validation Set (Early Stopping at Epoch 18)

Class	Precision	Recall	F1-score	Support
Negative	0.75	0.76	0.75	21197
Positive	0.75	0.75	0.75	21199
Accuracy			0.75	
Macro avg	0.75	0.75	0.75	42396
Weighted avg	0.75	0.75	0.75	42396

Best Validation F1 Score: 0.7569 in epoch 13

Comparison: When using `MSELoss`, the model achieved a maximum validation F1 score of 0.7569 at epoch 13. While this result is competitive, it is slightly lower than the best performance obtained with `BCELoss`, which reached a validation F1 score of 0.7618 at epoch 28. Additionally, the training curve under `BCELoss` demonstrated more stable and consistent improvements over time. In contrast, `MSELoss` showed a quicker convergence but plateaued earlier, with minor fluctuations in later epochs. Given that **BCELoss** is specifically designed for binary classification tasks and better aligns with the probabilistic output of the model, it **remains the more suitable choice for this project**.

3.5.2. BCEWithLogitsLoss. **BCEWithLogitsLoss** is tested as a numerically more stable alternative that combines a sigmoid layer with binary cross-entropy in a single function. This variant allows the model to output raw logits, avoiding potential instability in the probability space.

Table 17: Training and Validation Metrics with Early Stopping

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	43.1760	37.3086	0.6559	0.6569	0.6317
2	35.8389	34.9759	0.6828	0.6815	0.6630
3	34.8825	34.2291	0.6996	0.6978	0.6855
4	34.4220	33.9886	0.7281	0.7283	0.7260
5	34.0237	33.7785	0.6994	0.6975	0.6833
6	33.6210	33.1310	0.7176	0.7144	0.7051
7	33.3075	32.9020	0.7250	0.7214	0.7138
8	33.0055	32.9786	0.7473	0.7435	0.7422
9	32.6940	32.4490	0.7403	0.7339	0.7293
10	32.4110	32.5857	0.7299	0.7241	0.7159
11	32.1680	32.5079	0.7578	0.7477	0.7465
12	31.9131	32.6418	0.7233	0.7141	0.7025
13	31.6635	33.2928	0.7134	0.7059	0.6911
14	31.4690	32.0116	0.7446	0.7336	0.7275
15	31.2766	31.6587	0.7566	0.7449	0.7412

Table 18: Classification Report on Validation Set (Early Stopping at Epoch 16)

Class	Precision	Recall	F1-score	Support
Negative	0.70	0.85	0.77	21197
Positive	0.81	0.64	0.72	21199
Accuracy			0.75	
Macro avg	0.76	0.75	0.74	42396
Weighted avg	0.76	0.75	0.74	42396

Best Validation F1 Score: 0.7465 in epoch 11

Comparison: The experiment using `BCEWithLogitsLoss` resulted in a best validation F1 score of 0.7465, reached at epoch 11. In comparison, `BCELoss` achieved a higher peak validation F1 score of 0.7618 at epoch 28. While `BCEWithLogitsLoss` led to faster convergence and reasonable early performance, it did not surpass the results obtained with `BCELoss`. Moreover, performance fluctuations in later epochs and the earlier triggering of early stopping suggest that the model under `BCEWithLogitsLoss` may not have generalized as effectively in this configuration. Given the superior performance and stability observed with **BCELoss, it remains the preferred loss function for this task**.

Conclusion: Based on the evaluation of the experimented loss functions, **BCELoss** provided the best overall performance, achieving the highest validation F1 score (0.7618) and demonstrating consistent improvements throughout training. While both `MSELoss` and `BCEWithLogitsLoss` showed competitive results, neither surpassed the performance or training stability of `BCELoss`. Therefore, `BCELoss` is selected as the loss function for subsequent experiments in this project.

3.6. Hidden Layers Tuning

In the previous experiments, the neural network architecture consisted of three hidden layers with 128, 64, and 32 units respectively. In this section, the effect of varying the number and size of hidden layers on model performance is explored. All other parameters are kept constant to ensure a fair comparison(*GloVe 300d embeddings, the optimal learning rate of 5e-4, and the SGE optimizer*). The goal is to identify whether deeper or wider configurations offer meaningful improvements in accuracy and generalization for the sentiment classification task.

3.6.1. H1, H2, H3 = 256, 128, 64. For the first experiment the Hidden Layers were increased to H1, H2, H3 = 256, 128, 64 respectively.

Table 19: Training and Validation Metrics with Early Stopping (Epochs 1–10, 21–28)

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	43.5557	39.1181	0.6622	0.6648	0.6562
2	35.9945	34.6019	0.7257	0.7277	0.7277
3	34.8624	33.9964	0.7318	0.7335	0.7332
4	34.3099	33.7199	0.7337	0.7348	0.7337
5	33.8864	33.5140	0.7360	0.7358	0.7339
6	33.4957	33.5947	0.7389	0.7383	0.7370
7	33.1356	32.6092	0.7513	0.7469	0.7469
8	32.7799	32.5105	0.7537	0.7481	0.7481
9	32.4636	34.2470	0.7318	0.7264	0.7209
10	31.6296	31.9528	0.7640	0.7532	0.7531
21	29.3776	31.5237	0.7860	0.7595	0.7593
22	29.2406	31.6835	0.7847	0.7591	0.7588
23	28.7018	31.4423	0.7897	0.7608	0.7604
24	28.5927	31.4195	0.7925	0.7612	0.7612
25	28.5087	31.3871	0.7936	0.7608	0.7607
26	28.4152	31.4463	0.7949	0.7608	0.7608
27	28.3212	31.4277	0.7960	0.7612	0.7611
28	27.9911	31.4488	0.7973	0.7611	0.7610

Table 20: Classification Report on Validation Set (Early Stopping at Epoch 29)

Class	Precision	Recall	F1-score	Support
Negative	0.76	0.76	0.76	21197
Positive	0.76	0.76	0.76	21199
Accuracy			0.76	
Macro avg	0.76	0.76	0.76	42396
Weighted avg	0.76	0.76	0.76	42396

Best Validation F1 Score: 0.7612 in epoch 24

Comparison: Increasing the size of the hidden layers to $H_1 = 256$, $H_2 = 128$, and $H_3 = 64$ led to a slight improvement in validation performance. The model reached a best F1 score of 0.7612 at epoch 24, which is marginally higher than the previously best-performing configuration ($H_1 = 128$, $H_2 = 64$, $H_3 = 32$), which achieved an F1 of 0.7610 at epoch 28. In addition to this small performance gain, the wider architecture converged earlier, suggesting better learning efficiency. However, the improvement is minimal and may not justify the additional computational cost unless higher model capacity is required for downstream tasks. Overall, the wider configuration offers a modest benefit in accuracy, but both architectures perform comparably.

3.6.2. H1, H2, H3 = 512, 256, 128. For the second experiment the Hidden Layers were increased to H1, H2, H3 = 512, 256, 128 respectively.

Table 21: Training and Validation Metrics with Early Stopping

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	42.7100	36.4277	0.7047	0.7064	0.7056
2	35.6206	34.4313	0.7281	0.7295	0.7292
3	34.7796	33.9549	0.7334	0.7342	0.7337
4	34.2231	34.8498	0.7193	0.7191	0.7125
5	33.7299	33.2604	0.7402	0.7407	0.7396
6	33.2590	32.6937	0.7496	0.7477	0.7474
7	32.8307	32.5312	0.7555	0.7497	0.7497
8	32.4952	32.3764	0.7558	0.7503	0.7493
9	32.1041	32.3777	0.7553	0.7480	0.7460
10	31.7728	31.9643	0.7662	0.7540	0.7540
11	31.5140	32.0055	0.7659	0.7548	0.7539
12	31.1802	31.7094	0.7740	0.7555	0.7554
13	30.9540	32.0356	0.7678	0.7542	0.7531
14	30.6092	31.7089	0.7752	0.7564	0.7552
15	30.3382	32.2239	0.7740	0.7475	0.7458
16	30.0675	33.8135	0.7614	0.7382	0.7343

Table 22: Classification Report on Validation Set (Early Stopping at Epoch 17)

Class	Precision	Recall	F1-score	Support
Negative	0.82	0.61	0.70	21197
Positive	0.69	0.86	0.77	21199
Accuracy			0.74	
Macro avg	0.75	0.74	0.73	42396
Weighted avg	0.75	0.74	0.73	42396

Best Validation F1 Score: 0.7554 in epoch 12

Comparison: Increasing the hidden layer sizes to $H_1 = 512$, $H_2 = 256$, and $H_3 = 128$ did not result in an improvement over the previous configurations. The best validation F1 score achieved was 0.7554 at epoch 12, which is lower than the 0.7612 reached by the 256-128-64 architecture. Additionally, the model demonstrated early signs of overfitting, with performance beginning to degrade after epoch 13 and early stopping being triggered at epoch 17. While a wider architecture can offer greater representational capacity, in this case it did not yield better generalization and led to imbalanced class performance, as reflected in the precision-recall tradeoff. These results suggest that excessively increasing the number of neurons in each layer may introduce unnecessary complexity without practical benefits for this task.

Conclusion: After evaluating multiple hidden layer configurations, the architecture with hidden layers $H_1 = 128$, $H_2 = 64$, and $H_3 = 32$ demonstrated the most optimal balance between performance and computational cost. It achieved the highest validation F1 score (0.7618) among all tested configurations while maintaining a relatively lightweight structure that allowed for efficient training. Larger architectures, although offering greater representational capacity, did not lead to significant performance improvements and in some cases introduced overfitting or unnecessary complexity. Therefore, the 128-64-32 configuration is selected as the optimal hidden layer setup for this model.

4. Results and Overall Analysis

4.1. Best Trial

After extensive experimentation and systematic tuning of architectural and training parameters, the final model configuration was selected based on a combination of performance, convergence behavior, and computational efficiency. Multiple settings were tested across various components of the training pipeline, including embeddings, optimizers, loss functions, learning rates, and network depth.
The final model setup includes the following hyperparameters and architecture choices:

- **Embedding Size:** GloVe 300-dimensional word vectors, selected for their ability to capture rich semantic relationships and consistently outperform lower-dimensional alternatives.
- **Epochs:** 30 training epochs, providing sufficient time for convergence while avoiding overfitting.
- **Learning Rate:** 5e-4, found to offer a stable tradeoff between convergence speed and model performance.
- **Optimizer:** Stochastic Gradient Descent (SGD), selected for its reliable generalization and top-performing results when paired with the chosen learning rate.
- **Loss Function:** Binary Cross Entropy Loss (`BCELoss`), which delivered the highest F1 score and stable training dynamics compared to alternatives.
- **Hidden Layers:** A three-layer architecture with sizes $H_1 = 128$, $H_2 = 64$, and $H_3 = 32$, identified as the most effective in balancing expressiveness and computational cost.

Table 23: Training and Validation Metrics Across 30 Epochs

Epoch	Training Loss	Validation Loss	Train Acc	Val Acc	Val F1
1	43.9248	41.0667	0.6710	0.6718	0.6705
2	36.3687	34.8050	0.7223	0.7249	0.7231
3	35.0157	34.1131	0.7307	0.7328	0.7326
4	34.5064	33.7992	0.7346	0.7352	0.7347
5	34.0985	34.6412	0.7196	0.7199	0.7136
6	33.7147	33.2559	0.7401	0.7404	0.7396
7	33.3708	33.0271	0.7467	0.7444	0.7440
8	33.0844	32.9319	0.7458	0.7429	0.7413
9	32.7671	32.9068	0.7497	0.7446	0.7440
10	32.0363	32.2126	0.7588	0.7530	0.7529
11	31.8516	32.2980	0.7577	0.7500	0.7490
12	31.6705	32.4336	0.7584	0.7491	0.7486
13	31.4791	32.0637	0.7627	0.7526	0.7522
14	31.3517	32.1305	0.7641	0.7504	0.7502
15	31.2008	31.8490	0.7669	0.7552	0.7548
16	30.6212	31.7182	0.7715	0.7578	0.7576
17	30.4976	31.6928	0.7729	0.7576	0.7576
18	30.4033	31.6657	0.7741	0.7587	0.7587
19	30.2879	31.6987	0.7742	0.7568	0.7562
20	29.9284	31.5715	0.7775	0.7584	0.7582
21	29.8447	31.8242	0.7757	0.7552	0.7547
22	29.7772	31.4967	0.7794	0.7599	0.7599
23	29.7138	31.5301	0.7804	0.7585	0.7584
24	29.5014	31.4786	0.7821	0.7608	0.7607
25	29.4533	31.5209	0.7810	0.7601	0.7601
26	29.4135	31.4585	0.7826	0.7607	0.7607
27	29.3807	31.4790	0.7828	0.7612	0.7611
28	29.2472	31.4467	0.7835	0.7619	0.7618
29	29.2310	31.4686	0.7837	0.7610	0.7610
30	29.2046	31.5149	0.7833	0.7600	0.7600

Table 24: Classification Report on Validation Set (Epoch 30)

Class	Precision	Recall	F1-score	Support
Negative	0.76	0.75	0.76	21197
Positive	0.76	0.77	0.76	21199
Accuracy			0.76	
Macro avg	0.76	0.76	0.76	42396
Weighted avg	0.76	0.76	0.76	42396

Best Validation F1 Score: 0.7618 in epoch 28

4.2. Plots for the best trial

4.2.1. Training and Validation Loss over Epochs.

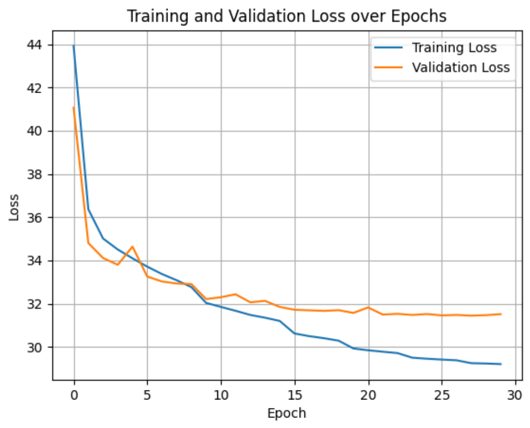


Figure 7: Training and validation loss across epochs for the best model (300d GloVe).

The loss curves indicate consistent and stable convergence. The validation loss closely follows the training loss without divergence, suggesting that the model generalizes well and avoids overfitting throughout the training process.

4.2.2. Training and Validation Accuracy over Epochs.

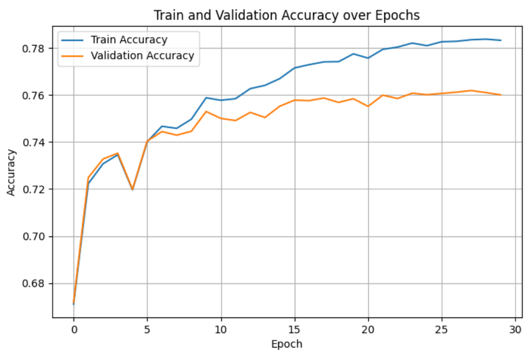


Figure 8: Training and validation accuracy across epochs for the best model (300d GloVe).

Training accuracy steadily improves and surpasses 78%, while validation accuracy stabilizes around 76%. The relatively small gap between the curves demonstrates that the model maintains strong generalization ability with no significant overfitting.

4.3. Plots for the Best Trial

4.3.1. Training and Validation Loss over Epochs.

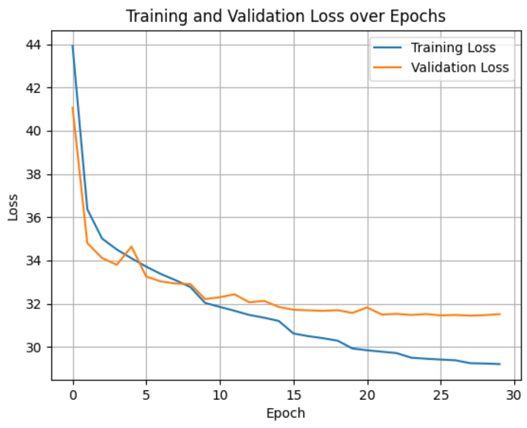


Figure 9: Training and validation loss across epochs for the best model (300-dimensional GloVe).

The training and validation loss curves demonstrate steady and reliable convergence. Throughout the epochs, the validation loss closely tracks the training loss, showing no signs of divergence. This alignment suggests that the model generalizes effectively to unseen data and avoids overfitting during the training process.

4.3.2. Training and Validation Accuracy over Epochs.

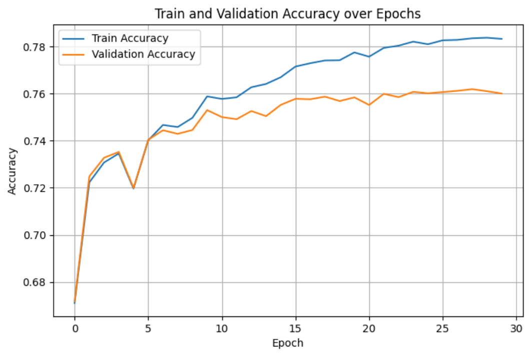


Figure 10: Training and validation accuracy across epochs for the best model (300-dimensional GloVe).

The training accuracy increases consistently over time, while validation accuracy stabilizes at a slightly lower yet comparable level. The narrow gap between the two curves is indicative of strong generalization, with the model learning effectively without becoming overly specialized to the training set.

4.3.3. Validation F1 Score over Epochs.

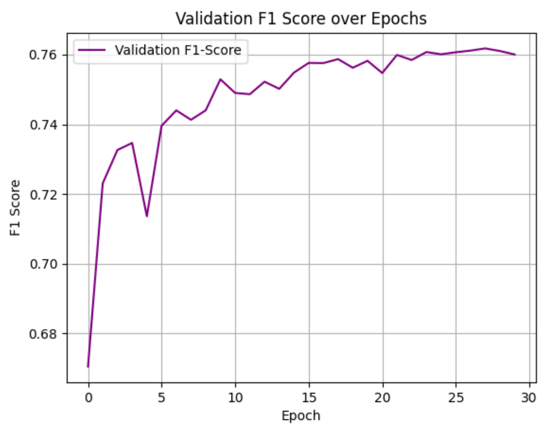


Figure 11: Validation F1 score progression across epochs for the best model (300-dimensional GloVe).

The F1 score improves significantly in the initial stages of training and then gradually plateaus. The leveling off after epoch 28 suggests that performance gains become marginal beyond that point, indicating that the model has effectively converged and further training yields diminishing returns.

4.3.4. Confusion Matrix.

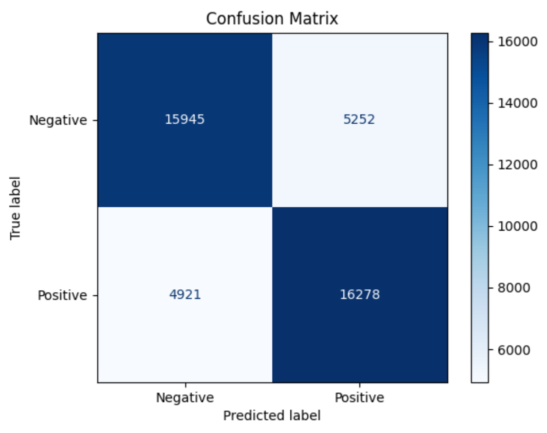


Figure 12: Confusion matrix on the validation set for the best model.

The confusion matrix reflects balanced classification performance across both classes. While a number of false positives and false negatives persist, the majority of predictions align with the ground truth, validating the model's overall robustness and reliability.

4.3.5. Receiver Operating Characteristic (ROC) Curve.

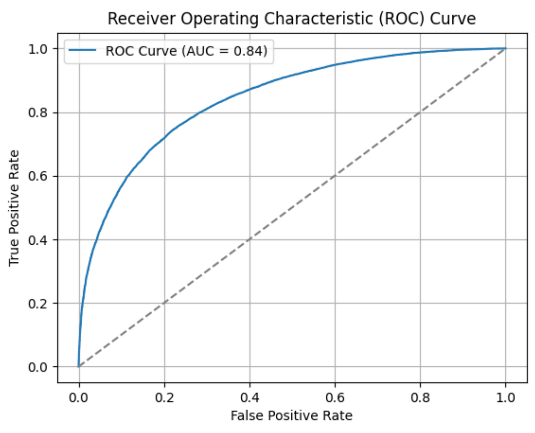


Figure 13: ROC curve for the best model with an area under the curve (AUC) of 0.84.

The ROC curve demonstrates strong discriminative capability, with the model achieving a high true positive rate while maintaining a low false positive rate across various thresholds. An AUC of 0.84 indicates that the classifier distinguishes effectively between the two classes.

4.3.6. Precision–Recall Curve.

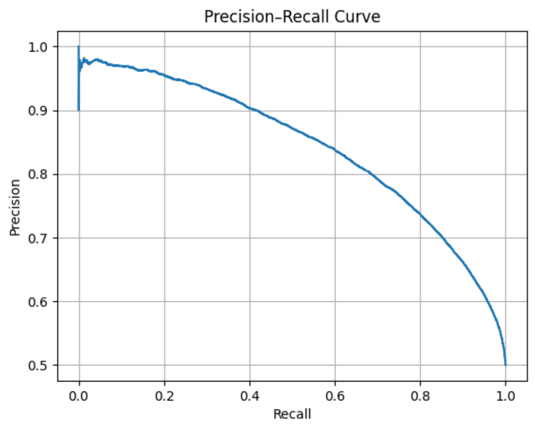


Figure 14: Precision–recall curve for the best model on the validation set.

The precision–recall curve reveals that the model maintains strong precision at lower recall levels, gradually trading off as recall increases. This behavior is particularly advantageous in scenarios where minimizing false positives is critical, such as in imbalanced or cost-sensitive classification problems.

4.4. Comparison with the first project

The first project employed a traditional machine learning methods, utilizing TF-IDF vectorization and logistic regression. This approach achieved strong results, with higher overall validation accuracy and F1 score. Its effectiveness stemmed largely from the use of a high-dimensional, sparse feature space crafted through extensive feature engineering, including n-gram tokenization and vocabulary tuning.

In contrast, the current project adopted a deep learning method based on pretrained GloVe embeddings and a feedforward neural network. While its accuracy and F1 score were slightly lower, the model demonstrated smoother convergence and superior performance in terms of ROC AUC. This indicates that the neural network developed a stronger ability to distinguish between classes under varying decision thresholds, particularly in probabilistic outputs.

The first project serves as a strong example of classical feature-based modeling, offering high performance with interpretable results. The second project, on the other hand, leverages semantic word representations and non-linear decision boundaries, showcasing the advantages of deep learning in terms of flexibility and scalability. Although it did not surpass the earlier model in raw classification metrics, it exhibited promising generalization behavior and creates the basis for more advanced architectures in future iterations.

5. Bibliography

References

[1] Geeks for Geeks. How to handle overfitting in pytorch models using early stopping. https://www.geeksforgeeks.org/how-to-handle-overfitting-in-pytorch-models-using-early-stopping/, 2024.

[2] Charalampos M. Liapis, Aikaterini Karanikola, and Sotiris Kotsiantis. Enhancing sentiment analysis with distributional emotion embeddings. https://www.sciencedirect.com/science/article/pii/S0925231225004941/, 2025.

[2] [1]