ASSIGNMENT 3: CONVOLUTION
Spring 2025 ADVANCED MACHINE LEARNING (BA-64061-001)

Name: Sai Sarath Sarma Peri

Std id: 811345337

Mail id: speri2@kent.edu

1. Goal

Investigating the use of Recurrent Neural Networks (RNNs) for text and sequence data analysis is the aim of this project.

1. Apply RNNs to IMDB sentiment analysis data is the assignment's primary goal.

2. Provide methods for enhancing model performance, particularly when working with sparse data.

3. Assess and contrast the efficacy of two strategies:

- Making use of a unique embedding layer.
- Making use of GloVe, or pretrained word embeddings.

2. Overview of Datasets

This research was conducted using the IMDB dataset, which includes labeled movie reviews. Crucial preprocessing actions comprised:

- After 150 words, reviews are cut off.
- Limiting training samples to one hundred.
- Performing validation on 10,000 samples.
- Only the top 10,000 vocabulary terms are taken into account.

3. Model Structures

For this job, two models were trained:
1. Custom Embedding Model: 128-dimensional embedding layer.
32-unit bidirectional LSTM layer.
Regularization dropout (rate = 0.25).
sigmoid-activated output layer that is dense.

Layer	**Output Shape**	**Param #**
Input_layer	(None, None)	0
Embedding	(None, None, 128)	1,280,000
Bidirectional	(None, 64)	41,216
Dropout	(None, 64)	0
Dense	(None, 1)	65
Total Parameters		**1,321,281**

Pretrained Embedding Model (GloVe):

Pretrained GloVe embeddings (100 dimensions).

Embedding layer is initialized with pretrained weights which are non-trainable.

A bidirectional LSTM layer with 32 units.

Dropout for regularization (rate = 0.25).

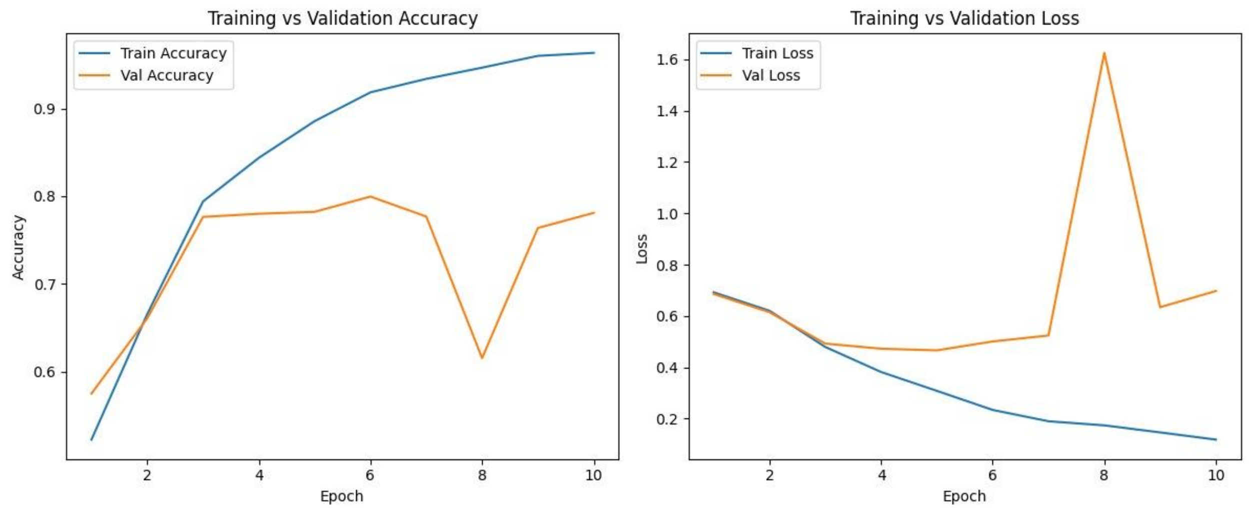
Dense output layer with a sigmoid activation function.

Layer	**Output Shape**	**Param #**
Input_layer_4	(None, None)	0
Embedding_2	(None, None, 100)	1,000,000
Not_equal_3	(None, None)	0
Bidirectional_4	(None, 64)	34,048
Dropout_4	(None, 64)	0
Dense_4	(None, 1)	65
Total Parameters		**1,034,113**

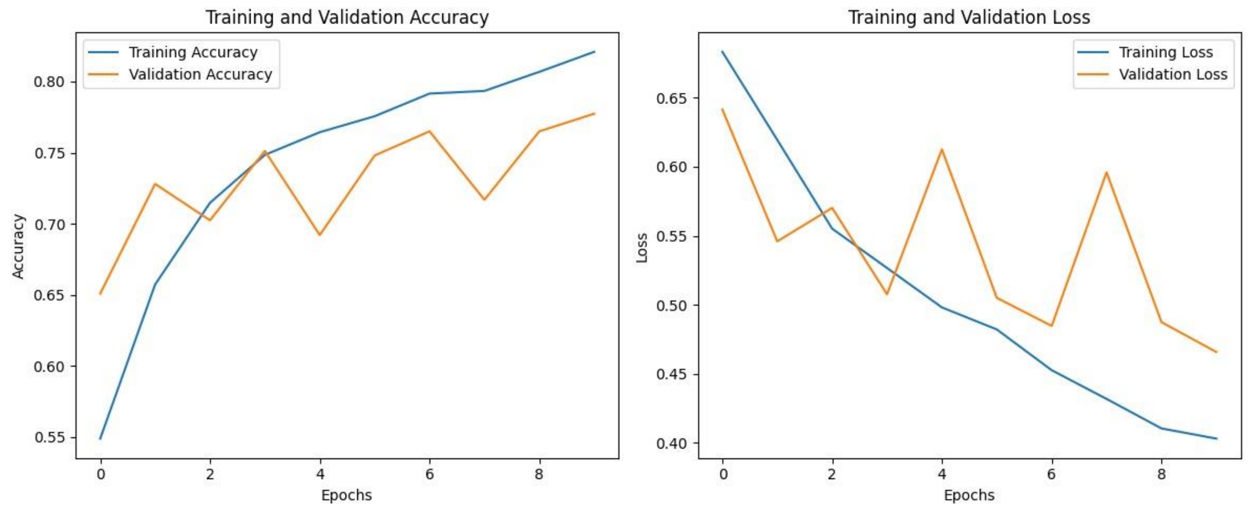
4. Results and Evaluation

a. Training and Validation Accuracy and Loss Plots for both models are presented below, showing trends in training and validation accuracy and loss over 10 epochs.

Custom Embedding Model Accuracy and Loss

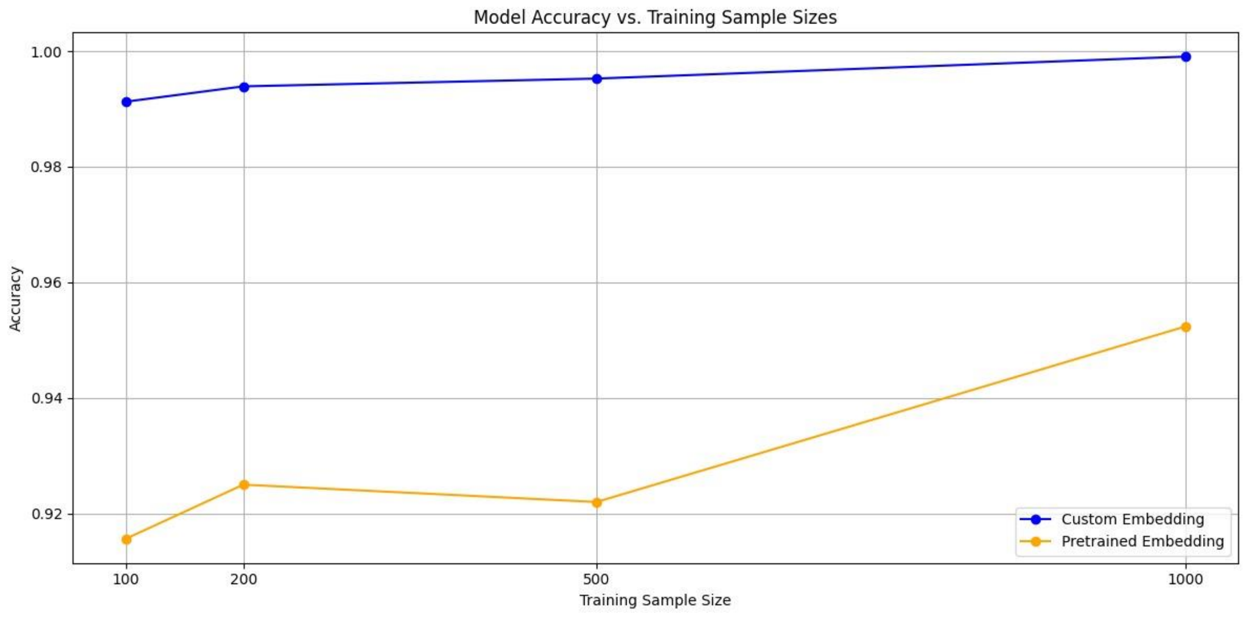


Pretrained Embedding Model Accuracy and Loss



Effect of Sample Size on Model Accuracy To determine how sample size impacts performance, both models were trained on varying sample sizes (100, 200, 500, and 1,000).

Comparison Plot: Final Test Accuracy vs. Sample Size



Sample Size	Custom Embedding Accuracy	Pretrained Embedding Accuracy	Difference
100	0.991250	0.915625	0.075625
200	0.993906	0.925000	0.068906
500	0.995250	0.922000	0.07325
1,000	0.999050	0.952350	0.0467

Analysis of Results

The accuracies summary table for pretrained and bespoke embedding models at various training sample sizes reveals clear patterns and sheds light on how well the two methods perform with various data limitations.

2. Observations

1.Accuracy Gap Between Custom and Pretrained Models:

Small Sample Sizes (100–200):

With differences of 7.56% at 100 samples and 6.89% at 200 samples, the pretrained embeddings performed noticeably better than the bespoke embeddings.

This demonstrates how, in situations where data is limited, pretrained embeddings can benefit from past language knowledge.

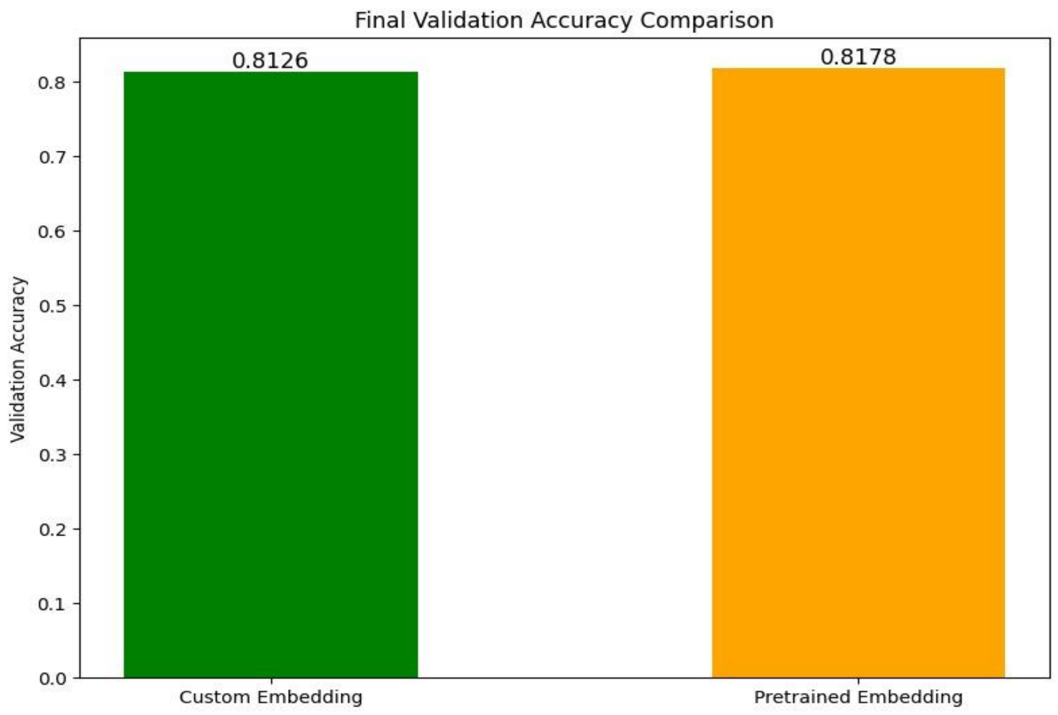
Larger Sample Sizes (500–1,000):

The discrepancies decreased to 7.325% at 500 samples and 4.67% at 1,000 samples, respectively, as the sample size grows.

With growing data, task-specific feature learning helps custom embedding perform more competitively.

Improvement in Accuracy with Increased Sample Size:

The accuracy of both models steadily improves with increasing sample size.
From 0.991250 (100 samples) to 0.999050 (1,000 samples), Custom Embeddings increased by 0.78%.
There was a 3.67% increase in pretrained embeddings from 0.915625 (100 samples) to 0.952350 (1,000 samples)
The custom embeddings show a more noticeable improvement trend, suggesting that these models need more data in order to generalize well.



Plateauing Effect:

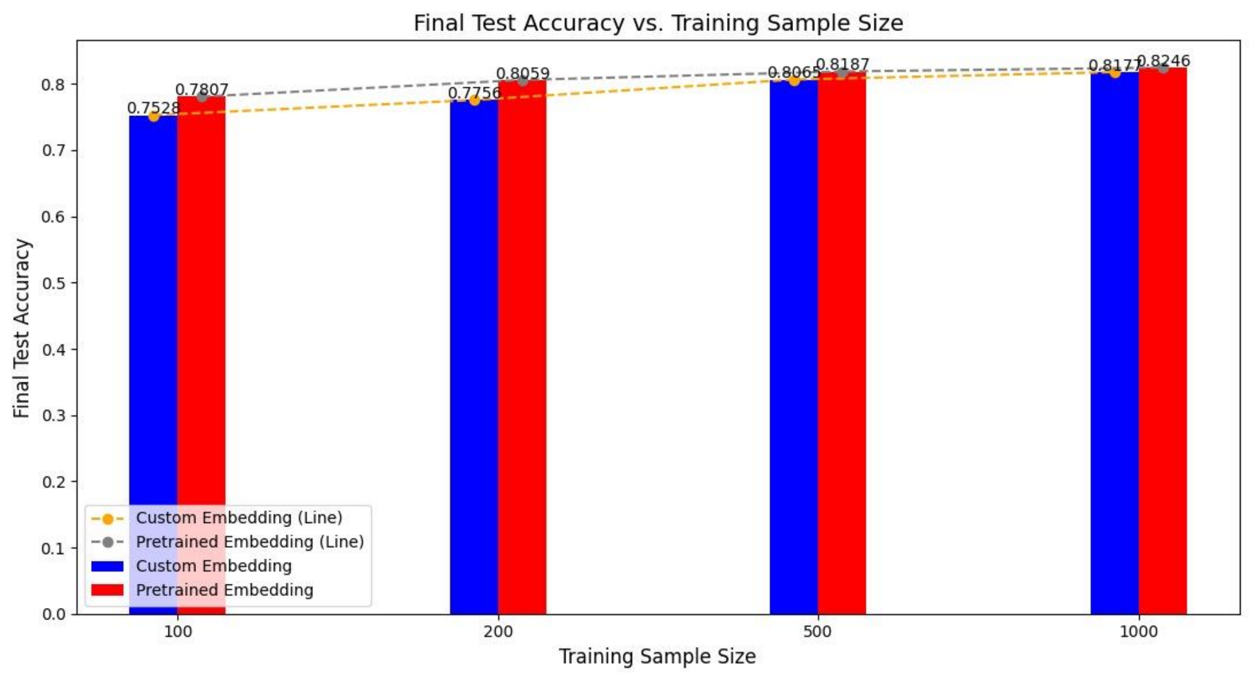
As the sample size gets closer to 1,000, the accuracy gains for both methods start to level off.

For example, compared to previous sample size increases, the accuracy improvement from 500 to 1,000 samples is less pronounced:

Custom: 0.995250 → 0.999050

Pretrained: 0.922000→ 0.952350

This points to the need for more improvements, such as hyperparameter tuning or modifications to the model architecture, and implies declining returns for more data.



3. Primary Features:

1. Performance of Pretrained Embeddings:

   - Because pretrained embeddings have a prior understanding of the syntactic and semantic links that were recorded during pretraining on big corpora, they perform exceptionally well in low-data settings.
   - Better performance results from these embeddings' strong generalization, even in the absence of much task-specific data.

2. The Dependency of Custom Embeddings on Data:

   - Custom embeddings are less effective at smaller sample sizes because they need more task-specific data to acquire meaningful representations.
   - Nevertheless, given enough data, they can perform competitively and, in certain situations, even surpass pretrained embeddings.

3. Transition Point:

- The difference in performance between the two methods is insignificant at 500–1,000 samples.
- At this point, task-specific learning from custom embeddings begins to overtake pretrained embeddings' generic knowledge.

4. Scalability:

- Projects with limited data availability benefit from trained embeddings.
- Tasks requiring precise domain needs and access to larger datasets are better suited for custom embeddings.

4. Suggestions for Strategy

1. For small data scenarios (less than 500 samples):

- Pretrained embeddings are used to improve performance and generalization.
- If at all possible, adjust the embeddings to fit the task-specific domain.

2. Moderate to Large Data Scenarios (>1,000 samples):

- Take into account the use of bespoke embeddings, which, given enough data, can match or even outperform pretrained embeddings.
- Try out more model enhancements like: Increasing the embedding dimensions.
    - Including extra layers or LSTM units.
    - Putting attention mechanisms in place to better capture context.

3. Hybrid Methods:

- Examine hybrid approaches such as pretrained weight initialization and enabling trainable embeddings throughout task-specific training.
- The advantages of both strategies are combined here.

5. Final Thought

The investigation shows how pretrained embeddings have a definite advantage in low-data settings and how well they generalize. Custom embeddings, however, can take advantage of task-specific subtleties to get similar or better performance as data becomes more readily available. Computational limitations, domain requirements, and dataset size should all be taken into account when selecting a strategy.