Sentiment Analysis on the IMDB Dataset

ZHANG JINGYI G2505210L

1. Introduction

Sentiment analysis is a classical task in natural language processing, aiming to identify the sentiment polarity of text. This experiment is based on the IMDB dataset and compares the impact of different optimizers, numbers of epochs, model structures, and word vector initializations on classification performance. All experiments make minimal modifications to the RNN framework provided on Colab to ensure fairness and consistency of the comparison.

2. Dataset and Experimental Setup

The IMDB dataset contains movie reviews with a balanced distribution of sentiment labels. The original dataset is split into training and test sets, each accounting for 50% of the data, and the training set is further divided into training and validation subsets. A fixed random seed of 1234 is set to ensure reproducibility. Each word is mapped to a 100-dimensional embedding vector. The model is trained with a batch size of 64, and the loss function used is BCEWithLogitsLoss. The evaluation metric is accuracy.

3. Experimental Design and Result Analysis

3.1 Warm-up: Baseline RNN

The experiment first reimplements a baseline RNN model in Colab to perform a binary classification task on movie reviews. The performance is reported in Table 1.

Table 1 Performance of the Baseline RNN on the Test Set

Dataset	Loss	Accuracy
Test Set	0.685	54.47%

As shown in Table 1, the baseline model achieves only marginally higher accuracy than the random baseline, indicating relatively poor performance. This is mainly because RNN models have limited capability in modeling long-term dependencies. This baseline result provides a reference for the subsequent analysis.

3.2 Comparison of Different Optimizers

Based on the baseline RNN, we further compare the performance of three optimizers, SGD, Adam, and Adagrad, on the validation and test sets. The results are shown in Table 2, and the training curves are presented in Figure 1.

Table 2: Performance Comparison of Different Optimizers

Optimizer	Train Acc (%)	Val Acc (%)	Test Acc (%)	Test Loss
SGD	56.49	56.66	55.33	0.683
Adam	82.62	71.17	67.63	0.622
Adagrad	74.99	72.83	71.84	0.564

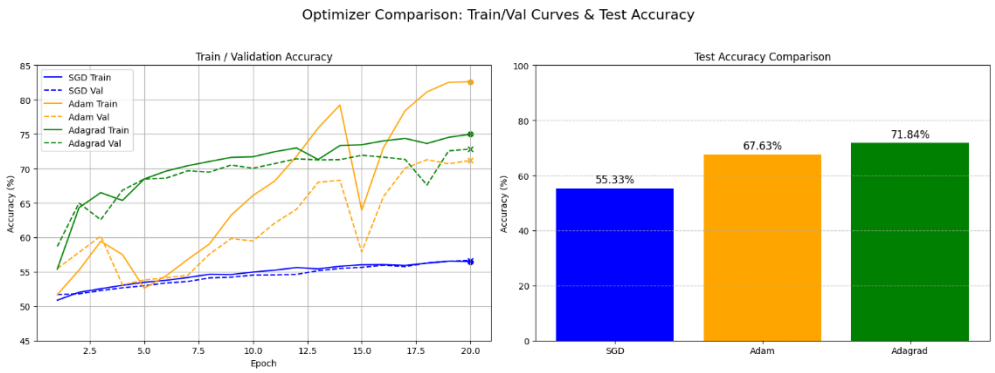

Figure 1 Illustration of Training Curve

Different optimizers exhibit clear trade-offs between convergence speed and generalization performance. SGD converges slowly and achieves the lowest test accuracy of about 56%. Adam converges the fastest and reaches a training accuracy of 82.62%, but its lower test accuracy (67.63%) and the noticeable gap between training and test accuracy suggest a tendency toward overfitting. In contrast, Adagrad shows more stable performance, with a training accuracy of 74.99% and a smaller gap between training, validation, and test results.

Overall, under identical model architectures and experimental settings, Adam is suitable for fast convergence, while Adagrad provides more stable optimization and better generalization performance.

3.3 Experiment on Different Training Rounds

Under the Adam optimizer, this experiment compares performance across 5, 10, 20, and 50 epochs. The detailed results are shown in Table 3.

Table 3: Performance of Baseline RNN with Different Numbers of Training Epochs

Epochs	Train Acc (%)	Val Acc (%)	Test Acc (%)	Test Loss
5	66.61	63.89	60.33	0.673
10	72.02	62.4	69.1	0.619
20	81.9	71.32	68.83	0.604
50	74.58	59.75	60.06	0.683

The number of epochs has a significant impact on model performance. When the number of epochs is small (5 epochs), the model does not fully learn text features, leading to underfitting with lower training (66.61%) and test accuracy (60.33%). As the number of epochs increases, test performance improves and achieves relatively good overall performance at 20 epochs (68.83%). However, when the number of epochs increases to 50, test accuracy drops to 60.06%, indicating overfitting.

Combined with the results in Section 3.2, Adam's performance is sensitive to the choice of training epochs: too few epochs make it difficult for the model to converge, while too many lead to overfitting. Therefore, introducing early stopping can automatically select an appropriate number of epochs based on the training process, balancing convergence and generalization. In this experiment, 20 epochs are selected as the optimal training setting.

3.4 Comparison of Different Models

Under the Adam optimizer, with 50 training epochs and random initialization of word vectors, six different models are compared, and the results are shown in Table 4. Figure 2 further visualizes the comparison on the training set, as well as test accuracy and loss.

Table 4: Performance Comparison of Models on the IMDB Dataset

Model	Configuration	Final Train Acc (%)	Final Val Acc (%)	Test Acc (%)	Test Loss
1-layer MLP	[500]	100.00	87.50	86.58	0.351
2-layer MLP	[500, 300]	99.63	86.95	85.08	0.379
3-layer MLP	[500, 300, 200]	99.71	87.57	85.60	0.357
CNN	filters=[1,2,3], n_filters=100	100.00	89.40	87.73	0.289
LSTM (Uni)	hidden_dim=256	100.00	86.83	86.01	0.375
LSTM (Bi)	hidden_dim=256, bi=True	100.00	87.16	85.99	0.376

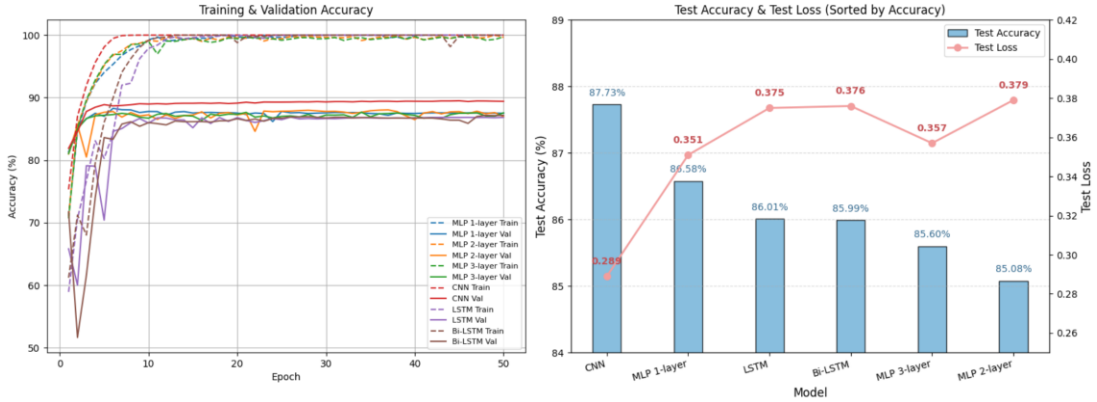

Figure 2 IMDB Model Comparison

CNN achieves the best performance on the validation and test sets, with a test accuracy of 87.73%. In contrast, the MLP series reaches an accuracy close to 100% on the training set, but shows only limited improvement on the test set, indicating slight overfitting. A single-layer MLP is already sufficient to fit text features, while deeper structures do not bring clear benefits. The unidirectional LSTM achieves validation and test accuracies of 86.83% and 86.01%, respectively, while the bidirectional LSTM achieves a slightly lower test accuracy of 85.99%, failing to provide a clear performance advantage.

The performance differences can be understood from model mechanisms and data characteristics. IMDB movie reviews are typically 200–300 words long, with sentiment information often concentrated in a few key phrases, such as “mess of a movie deserves” in negative reviews and “greatest game ever created” or “so addicting” in positive reviews. CNN can capture local n-gram features through convolutional kernels, which leads to better performance than MLP and LSTM on the validation and test sets, along with higher training efficiency and a lower risk of overfitting. In contrast, MLP relies on averaged word vectors and has difficulty capturing sequential or local information. Although LSTM is able to model long-term dependencies, sentiment classification often depends on partial key information, so its advantage is not fully exploited in this task.

Overall, CNN is the most suitable model for sentiment classification tasks where local textual patterns are important. MLP shows stable performance but limited improvement, while the advantages of LSTM are not fully utilized. The results indicate that model performance strongly

depends on how well the model characteristics match the data, providing motivation for further experiments with word vector pre-training.

3.5 Impact of Word Embedding Initialization

To evaluate the impact of word vector initialization on performance, the experiment re-runs the models used in Section 3.4, including one-layer MLP, two-layer MLP, three-layer MLP, and CNN. The results are shown in Table 5.

Table 5: Random vs Word2Vec Embeddings

Model	With Word2Vec (%)	Without Word2Vec (%)	Difference
1-layer MLP	87.94	86.58	1.36
2-layer MLP	87.48	85.08	2.4
3-layer MLP	87.92	85.60	2.32
CNN	89.30	87.73	1.57

The performance of all models improves after using Word2Vec embeddings. This is mainly because Word2Vec provides semantic prior information by mapping semantically similar words to nearby positions in the vector space. For example, words such as “good” and “excellent” have a small distance in the embedding space, which allows the model to utilize semantic relationships at an early stage of training. In contrast, random initialization requires the model to learn semantic information from scratch, which is more difficult and slower for a dataset of moderate size such as IMDB. Therefore, pre-trained word vectors can help enhance model performance.

Different models show different levels of dependency on word vectors. Deeper MLP models are more sensitive to embedding quality: both two-layer and three-layer MLP show significant improvements, while the one-layer MLP improves only slightly. CNN can capture n-gram features, so even with randomly initialized word vectors, it can partially learn discriminative local patterns. As a result, the improvement brought by pre-trained embeddings is relatively small (+1.57%).

Overall, word vectors have a positive impact on all models, especially on deeper MLPs, by providing a more robust semantic representation, which helps reduce the noise caused by random initialization and improves generalization capability.

4. Conclusion

CNN efficiently extracts local n-gram features, which makes it suitable for short-text sentiment analysis. In addition, deep MLP models are more sensitive to high-quality word vectors. The number of training epochs should be combined with early stopping to avoid underfitting or overfitting.

In future work, the advantages of LSTM can be further verified on longer texts or larger datasets. It is also worth exploring model performance with frozen or trainable word vector settings.