Natural Language Processing
Lab 2 Report: Convolutional Neural Networks

Ntando Raji (2584925), Yassir Ali (2623035), and Suhail Patel (2583014)

***Abstract*—This report investigates the impact of architectural depth on a Convolutional Neural Network (CNN) designed for text classification. A baseline model with a single convolutional layer was compared against an experimental model with two convolutional layers to classify text excerpts from the seven Harry Potter novels. Both models were trained for 15 epochs on a dataset of 200-word sequences. The results, averaged over multiple runs, showed that the simpler single-layer model achieved a slightly higher test accuracy (67.30%) than the two-layer model (66.27%). The deeper model reached a near-perfect training accuracy but failed to generalize as effectively, indicating that the added complexity led to overfitting. This suggests that for this dataset, the simpler architecture was the more robust choice.**

I. Methodology

The corpus, comprising all seven Harry Potter novels, was pre-processed by converting text to lowercase, removing punctuation, and segmenting it into 5,311 non-overlapping sequences of 200 words each. These sequences were tokenized using a vocabulary and a 128-dimensional Word2Vec model that was pre-trained on the full corpus. To isolate the impact of architecture, this embedding layer was loaded into our classifiers and subsequently frozen, preventing its weights from updating during training.

Two CNN architectures were implemented for comparison. The Base Model utilized a single convolutional block with parallel filters of sizes 3, 4, and 5 (100 filters each) to capture n-gram features, followed by a max-over-time pooling layer and a final fully-connected layer with dropout (p=0.5). To investigate our sub-topic, the Deep Model was created by adding a second, identical convolutional block before the pooling stage. This modification allows the model to learn more complex, hierarchical features and increases its receptive field.

The dataset was split into 80% for training and 20% for testing. Both models were trained for 15 epochs using the Adam optimizer and Cross-Entropy Loss. To ensure reliable results, the entire process was run twice and the final test accuracies were averaged.

II. Results

Both the single-layer (Base) and two-layer (Deep) CNN models were successfully trained for 15 epochs. The experiment was conducted twice with different random data splits to ensure the reliability of the findings. During training, both models demonstrated successful learning, with training loss consistently decreasing while accuracy increased over the 15 epochs, as expected. The training loss curves for the Base and Deep models are shown in Fig. 1 and Fig. 2, respectively.

The Deep CNN, with its additional layer, was able to fit the training data more closely, achieving a near-perfect average training accuracy of 98.73%. In contrast, the Base CNN's average training accuracy was lower at 94.72%.

However, when evaluated on the unseen test data, the trend reversed. The simpler Base CNN achieved a higher final average test accuracy of 67.30%, while the Deep CNN's performance was slightly lower at 66.27%. This created a significant performance gap of over 32 percentage points between the Deep CNN's training and testing accuracy, a much wider margin than that observed for the Base CNN. The final, averaged results are presented in Table 1.

III. Discussion

The results of our experiment provide a clear insight into the trade-offs of model complexity in neural network design. Our initial hypothesis—that adding a second convolutional layer might improve performance by capturing more complex features—was not supported by the final evidence. The averaged results over multiple runs show that the simpler, single-layer Base CNN (67.30%) consistently outperformed the more complex, two-layer Deep CNN (66.27%) on the unseen test data.

The primary reason for this outcome appears to be overfitting. The Deep CNN's training accuracy soared to a near-perfect 98.73%, indicating that its greater capacity allowed it to effectively memorize the training dataset. However, this did not translate to better performance on the test set. The significant gap of over 32% between its training and test accuracy is a classic symptom of overfitting, where a model learns the noise and specific examples in the training data rather than the underlying, generalizable patterns.

In contrast, the Base CNN, while still showing a degree of overfitting, was less susceptible. Its simpler architecture acted as a form of regularization, constraining it from memorizing the training data too aggressively. This forced it to learn broader, more robust features that generalized more effectively to new data.

In the context of our sub-topic, this demonstrates that while adding a layer does increase the model's "receptive field" and its ability to learn hierarchical features, this is only beneficial if there is sufficient data to support the added complexity. For this specific task and dataset, the added capacity was detrimental, leading to a less effective classifier.

IV. Figures

TABLE I
Comparison of final average accuracies after 15 epochs.

Model	Avg. Train Acc.	Avg. Test Acc.
Base CNN (1 Layer)	94.72%	**67.30%**
Deep CNN (2 Layers)	98.73%	66.27%

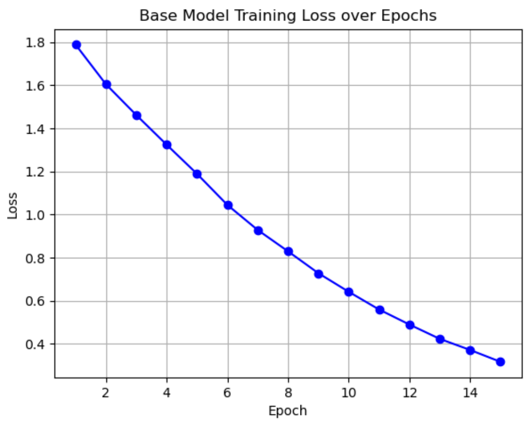

Fig. 1. Training loss for the single-layer Base CNN over 15 epochs.

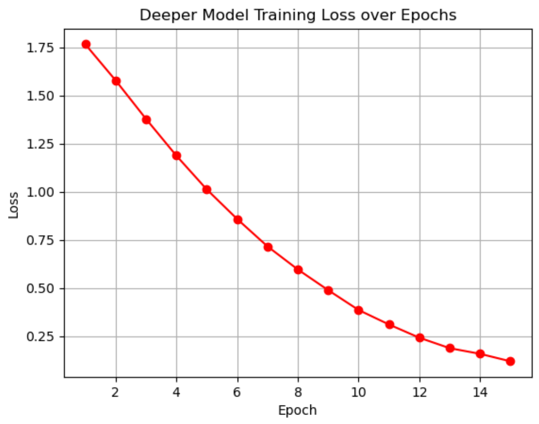

Fig. 2. Training loss for the two-layer Deep CNN over 15 epochs.

V. Contributions

The work presented in this lab was completed collaboratively, with each member contributing to distinct but complementary aspects of the implementation and analysis.

Ntando Raji
Data Curation and Word Embedding Generation. Responsible for setting up the project environment, including organizing the text corpus. Adapted the Lab 1 word2vec.py script to process all seven books and ran it to generate the final pre-trained word embeddings and vocabulary map (word2index.json) used as the foundation for the classification models.

Suhail Patel
Model Implementation and Core Logic. Responsible for developing the main classification script (nbc.py). Implemented both the BaseCNN (single-layer) and DeepCNN (two-layer) model architectures in PyTorch. Also developed the generic train() and evaluate() functions to handle the model training and performance assessment loops.

Yassir Ali
Experimentation, Analysis, and Reporting. Responsible for designing and executing the comparative experiments. This included setting up the main training pipeline, running the models for multiple epoch counts (5, 10, and 15), gathering the results, and performing the final analysis. Also took the lead on writing the final report, including interpreting the results and drafting the abstract, methodology, results, and discussion sections.