**Introduction**  
This study looks at how a neural network used for sentiment analysis with the IMDB dataset is presented. It includes how the data was prepared, how the model was built, trained, and tested. The report also shares the results, methods used, conclusions, and helpful suggestions for improving the model’s performance and understanding its predictions.

* **Loading the Dataset:**

The IMDB dataset was loaded using TensorFlow’s imdb.load\_data() function, keeping only the 10,000 most common words. The data was then divided into training and test sets. After loading the data, the structure of the dataset was checked to understand its format, and the data was properly cleaned and prepared for use in building the neural network model.

* **Preparing the Data:**

The integer sequences were changed into multi-hot encoded vectors so the neural network could understand them. The labels were also converted into NumPy arrays with float32 and float64 data types. This step made sure that all the data matched TensorFlow’s required format and could be smoothly used in training and testing the neural network model.

* **Model Construction:**

The neural network had four fully connected layers. The hidden layers used the **tanh** activation function, and the output layer used **sigmoid**. To prevent overfitting, **dropout layers** with a rate of 0.5 and **L2 regularization** of 0.005 were added. The model used **accuracy** to measure performance, **Mean Squared Error (MSE)** as the loss function, and the **RMSprop optimizer** for stable and adaptive learning.

* **Model Training, Validation, and Test Evaluation**

The dataset was divided into training, validation, and test sets. The neural network, built with four layers using tanh and sigmoid functions, was trained for 20 epochs with a batch size of 512. Dropout and L2 regularization helped prevent overfitting. During training, loss and accuracy were monitored, and finally, the model was tested on unseen data to check its performance, accuracy, and generalization ability.

**Observations:  
Training dynamics:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Epoch** | **Training Accuracy** | **Validation Accuracy** | **Validation Loss** |
| 1 | 64.8 % | 82.9 % | 0.29 |
| 10 | 91.6 % | 87.1 % | 0.13 |
| 20 | 96.1 % | 86.0 % | 0.14 |

Training accuracy steadily increased from **0.65 → 0.96**, while validation accuracy stabilized around **0.86 – 0.88**, confirming effective learning with minimal overfitting.

**Evaluation of Test Data:**

The model achieved the following performance on unseen test data:

|  |  |
| --- | --- |
| **Metric** | **Result** |
| **Test Accuracy** | **0.849 (84.9 %)** |
| **Test Loss (MSE)** | **0.148** |

A secondary model, trained with **Adam optimizer** for **4 epochs**, yielded **87.6 % accuracy** and **0.163 loss**, showing a small but consistent improvement in accuracy  
  
The model’s training accuracy improved from 64.8% to 96.1%, and validation accuracy reached about 87–88%. Test accuracy rose from 84.9% to 87.6%, while loss dropped from 0.67 to 0.06, proving the network learned IMDB review classification effectively.

**Comments:**

The tanh function worked well with both positive and negative values. Dropout and L2 regularization helped prevent overfitting. Using the Adam optimizer slightly improved accuracy. Training for more than 20 epochs may increase accuracy but risks overfitting, so early stopping is advised.

**Conclusion:**

The final neural network reached about **88%** accuracy on validation data and **85–88%** on test data, showing strong performance in predicting sentiments. The study shows that a balanced model design, proper dropout regularization, and the right optimizer choice play a key role in creating an effective and reliable text classification model for sentiment analysis tasks.

**Recommendations:**

Future improvements can include trying the **ReLU** activation function for faster training and adding **Batch Normalization** to make gradient updates more stable. Testing different **dropout rates** between 0.3 and 0.6 can help find the best regularization level. Using **cross-validation** will make hyperparameter tuning more reliable, and fine-tuning the **learning rate** and **L2 regularization** can help lower the model’s loss even further.