# Assignment 6 – Sentiment analysis using LSTM network or GRU.

Name : Aayush Bokde

Roll No: 382016

PRN: 22310316

## Problem Statement

Implement a Long Short-Term Memory (LSTM) based Recurrent Neural Network in Python using Keras and TensorFlow to classify movie reviews from the IMDB dataset into positive or negative sentiments.

## Objectives

* To understand Natural Language Processing (NLP) tasks like sentiment analysis.
* To preprocess textual data using tokenization and padding.
* To implement LSTM for text classification.
* To evaluate performance using accuracy and loss curves.
* To test model predictions on custom reviews.

## Requirements

* Operating System: Windows/Linux/MacOS
* Python Version: 3.x
* Tools: Jupyter Notebook / Anaconda / Google Colab
* Libraries Used: TensorFlow, Keras, NumPy, Matplotlib

## Theory

Sentiment Analysis is a text classification task that determines whether a given review is positive or negative. The IMDB dataset contains 50,000 movie reviews labeled as positive or negative.  
  
Word Embedding Layer: Maps integer-encoded words to dense vectors.  
LSTM Layer: Captures long-term dependencies and context from sequences.  
Dense Layer: Produces binary sentiment classification (positive/negative).  
Output: Probability of sentiment using sigmoid activation.

## Methodology

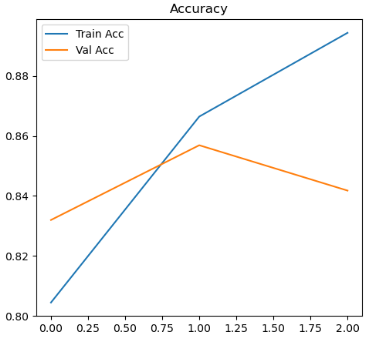
1. Data Acquisition: Load IMDB dataset using Keras (imdb.load\_data).
2. Data Preparation: Pad sequences to fixed length (200 words).
3. Model Architecture: Embedding layer (10000 → 128), LSTM layer (128 units), Dense layer (1 neuron, sigmoid activation).
4. Model Compilation: Optimizer = Adam, Loss Function = Binary Crossentropy, Metric = Accuracy.
5. Model Training: Batch size = 64, Epochs = 3, Validation on test set.
6. Model Evaluation: Test accuracy measured and plotted.
7. Prediction: Decode numerical review back to words and predict sentiment.

## Code Explanation (Snippet)

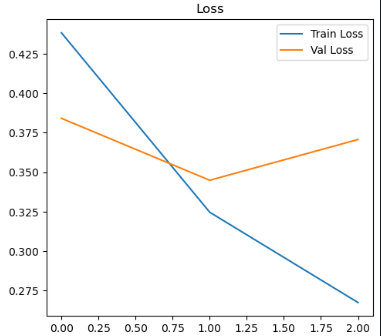
model = Sequential()  
model.add(Embedding(vocab\_size, 128, input\_length=maxlen))  
model.add(LSTM(128, dropout=0.2, recurrent\_dropout=0.2))  
model.add(Dense(1, activation="sigmoid"))  
  
model.compile(loss="binary\_crossentropy",  
 optimizer="adam",  
 metrics=["accuracy"])  
  
history = model.fit(X\_train, y\_train,  
 batch\_size=64,  
 epochs=3,  
 validation\_data=(X\_test, y\_test),  
 verbose=1)

## Graphs and Visualizations

• Accuracy vs Epochs: Shows training and validation accuracy over 3 epochs.



• Loss vs Epochs: Shows training and validation loss convergence.

  
• Sample Review Prediction: Displays decoded review and predicted sentiment.

## Advantages

* LSTM handles long-term dependencies in text better than simple RNNs.
* Embedding layer improves representation of words.
* Achieves high accuracy on standard sentiment analysis benchmarks.

## Limitations

* Training is computationally expensive compared to simple models.
* Limited vocabulary (10,000 words) may exclude rare words.
* Requires careful preprocessing for real-world noisy text.

## Applications

* Movie and product review analysis.
* Social media sentiment monitoring.
* Customer feedback classification.
* Opinion mining in politics and finance.

## Working / Algorithm

1. Import libraries and load IMDB dataset.
2. Preprocess reviews using padding and integer encoding.
3. Build LSTM model (Embedding → LSTM → Dense).
4. Compile with Adam optimizer and binary crossentropy loss.
5. Train the model with validation.
6. Evaluate accuracy on test data.
7. Plot accuracy and loss curves.
8. Decode review and predict sentiment.

## Conclusion

A Long Short-Term Memory (LSTM) based sentiment analysis model was implemented using the IMDB dataset. The model achieved good test accuracy and effectively classified reviews into positive and negative categories. Performance analysis using accuracy/loss plots confirmed the effectiveness of LSTMs for NLP tasks.