# Final Project Report

Group Members:
Parth Patel, Gaurang Kakade, Parth Ahir, Zelin Li, Ahmed Mosaad

## Task, Dataset, and Preprocessing

**Task:** The objective for this project is to perform sentiment analysis on IMDB movie reviews, whether a given review is positive or negative about that movie. The two deep learning systems used for this project are Convolutional Neural Network and Recurrent Neural Network. For Recurrent Neural Network, we used one of its types which is Long Short-Term Memory (LSTM).

**Dataset:** We used the IMDB dataset, which was loaded from TensorFlow, and this dataset comprises of two columns, the first column contains the reviews and the second column contains the label, whether it is “positive” or “negative” sentiment.

**CNN Preprocessing:**

- The reviews are tokenized, which is converting words into integer sequences. Then a word index is created which retains the top 10,000 frequent words from the reviews.
- Sequences are padded to a fixed length of 300, and this ensures a uniform input size for the neural network.
- Dataset is also split into training and test sets for model evaluation

**LSTM Preprocessing:**

- Similar to CNN model, reviews are tokenized using a word index and only the top 10,000 frequent words are kept.
- Sequences are padded to a fixed length of 100 to ensure uniform input size.
- A validation set is also created by extracting the first 10000 samples from the training data, and the remaining is used for training.

## Implementation and Architectures

**CNN:**

- Architecture:
  · Embedding layer: Input dimension of 10000, output dimension of 64 and input length of 300
  · Conv1D Layer: 32 filters and kernel size of 5
  · Reducing the spatial dimensions through MaxPooling1D Layer
  · Global Max pooling operation for spatial data
  · Dense Layers with 32 units and ReLU activation, to capture high level features learned by convolutional layers.
  · Output Layer: Dense layer with Sigmoid activation feature, which produces the final prediction, if the review is “positive” or “negative”
- Implementation:
  · Firstly, sequential model is created using Keras Sequential API, which allows for linear stacking of layers.
  · Then layers are added one by one, specifying input shapes wherever necessary.

**LSTM:**

- Architecture:
  · Embedding Layer: Input dimension of 10000, output dimension of 16 and input length of 100, which converts integer-encoded reviews into dense vectors.
  · LSTM Layer: 32 units for processing sequential data and capturing dependencies in sequential data.
  · Output Layer: Dense layer with a sigmoid activation function, which produces binary classification output.
- Implementation:
  · Sequential model composed using Keras Sequential API, where layers are added in sequence defining the input shape for the first year.
  · Each layer is responsible for specific aspects of feature extraction and representation

## Training Details

**CNN:**

· Optimizer: Adam optimizer with default learning rate

· Binary Crossentropy loss function, which is suitable for binary classification problems

· PrintAccuracy() function for real time accuracy monitoring, which prints training and test accuracy after each epoch

· Early Stopping: Patience set to 5, which stops training if validation loss does not improve for 5 consecutive epochs.

· 30 Epochs, which is the number of passes through the entire training dataset.

**LSTM:**

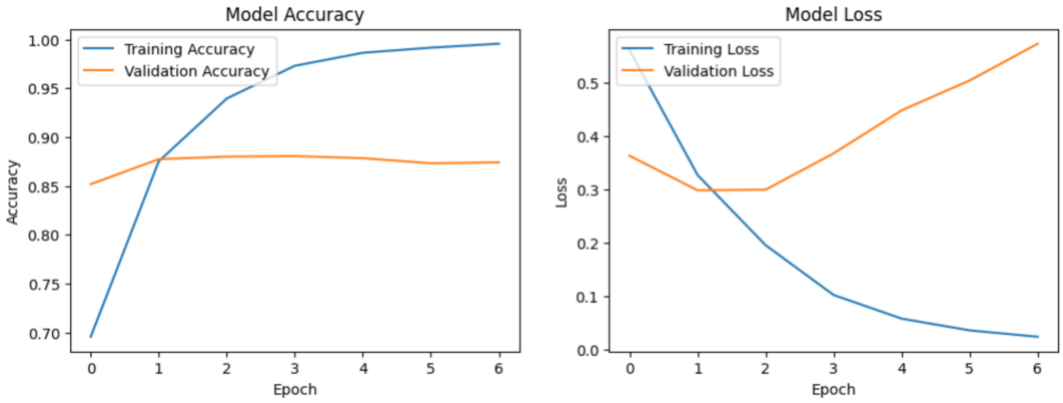
· Optimizer: The Adam optimizer was used for its effectiveness in handling sparse gradients and its adaptive learning rate capabilities, which are beneficial for datasets with varied input lengths and complexities.

· Loss Function: Binary cross-entropy was selected as the loss function, which is appropriate for binary classification tasks.

· The model was trained over multiple epochs with a defined batch size. These parameters were fine-tuned to ensure comprehensive learning without overfitting. The number of epochs was chosen to allow the model sufficient time to converge, while the batch size was set to balance computational efficiency and model performance.

## Results, Observations and Conclusions

**CNN:**



- Plot Analysis:

  · High Accuracy: The model achieved a high accuracy on the test dataset, indicating its effectiveness in classifying sentiment in movie reviews.

  · Stable Learning Curve: The plots of training and validation accuracy and loss demonstrate stable learning without significant signs of overfitting. This stability is indicative of the model's ability to generalize well to unseen data.

- Reflection:

  · Training vs. Test Performance: The difference in performance between training (around 98.72% accuracy) and testing (87.22% accuracy) suggests that the model might be overfitting.

  · The training accuracy is printed after each epoch, and model employs increased dropout and reduced complexity.

**LSTM:**


- Plot Analysis:

  · Training Accuracy: This increases steadily over the epochs, which is a good sign that the model is learning effectively from the training data.

  · Validation Accuracy: It improves initially but then plateaus, indicating that the model isn't generalizing as well beyond the training data.

  · Training Loss: It decreases over time, as expected when a model is learning.

· Validation Loss: It starts to increase after a few epochs, which is a classic sign of overfitting. The model is performing well on the training data but is failing to generalize that performance to the validation data

- Reflection:
  · Training vs. Test Performance: The difference in performance between training (around 97.71% accuracy) and testing (76.99% accuracy) suggests that the model might be overfitting.
  · Validation Loss Increase: The increasing trend in validation loss also supports the overfitting observation.

Conclusion (CNN and LSTM):

· The CNN model, with its current architecture and training strategy, proves to be a robust choice for sentiment analysis tasks.
· The increased dropout rate likely contributed to reducing overfitting, and the careful selection of hyperparameters resulted in a model that balances well between accuracy and generalization.

## Challenges, Obstacles and Solutions

- **Challenge 1: Overfitting**
  · Description: CNN model particularly in its initial iterations, showed signs of overfitting, where it performed well on training data but less so on validation and test data.
  · Solution: Increased the dropout rate in the CNN model. Additionally, employed early stopping to prevent the models from learning noise and irrelevant patterns from the training data.

- **Challenge 2: Model Selection and Architecture Tuning**
  · Description: Deciding on the appropriate model architecture and tuning hyperparameters like the number of layers, units in LSTM, filter sizes in CNN, and choosing the right optimizer and learning rate.

· Solution: Conducted experiments with various configurations, started with simpler models and gradually increased complexity, and monitored model performance to find the optimal settings.

- **Challenge 3: Computational Constraints**

  · Description: Handling the computational intensity and long training times, especially when experimenting with different model architectures and hyperparameters.

  · Solution: Utilized Google Colab for its free GPU resources, which significantly sped up training times. Also implemented efficient data handling practices to optimize resource usage

- **Challenge 4: Balancing Accuracy and Generalization**

  · Description: Striking a balance between achieving high accuracy and ensuring the model generalizes well to unseen data.

  · Solution: Used a validation dataset during training to monitor generalization and tweaked the models accordingly. Also, employed techniques like dropout and early stopping to avoid overfitting.