# REPORT

**ASSIGNMENT-01 NEURAL NETWORKS ADVANCED MACHINE LEARNING:**

# STEPS THAT ARE DONE BEFORE BUILDING UP THE MODEL:

* The IMDB dataset is loaded using the load\_data() function from the imdb module, which retrieves the dataset and splits it into training and test sets as tuples of NumPy arrays. By setting the num\_words parameter to 10,000, only the 10,000 most frequently occurring words across all reviews are included in the dataset. This approach helps minimize input size while filtering out infrequent words that may have little impact on the classification task.
* The first movie review in the training set of data is then printed after accessing train data[0]. The end product is a list of integers, each of which is a number that indicates a word's location in the dictionary of the 10,000 most common terms.
* Since the integer values reflect a word's index in a dictionary of the 10,000 most common words, the maximum value that results, 9999, represents the unique number of words in the dataset. The output of this code gives an approximate idea of the size of the neural network's input layer, which is what will be utilized to categorize the movie reviews.
* The variable review. Decode would return a string containing the original text of the first movie review in existence within the training set. This code snippet is useful in understanding how the input data is structured and that the input has indeed been loaded and pre-compiled correctly.
* Arrays will be the targets used to evaluate and train the neural network. The network is expected to discover a relationship between the sentiment labels y\_train and y\_test and the one-hot encoded input sequences x\_train and x\_test.

**Findings:**

1. The IMDB dataset was successfully loaded using the load\_data() function, which filtered the top 10,000 most frequently occurring words to reduce input size.
2. The model consisted of four fully connected layers with 64 neurons each, using tanh for hidden layers and sigmoid for the output layer.
3. Regularization techniques such as L2 regularization (regularizers.L2(0.005)) and dropout layers (0.5 dropout rate) were implemented to prevent overfitting.
4. The model was trained using the Adam optimizer with a mean squared error (MSE) loss function. It was trained with a batch size of 512 for 20 epochs.
5. Training accuracy increased from 73.29% to 93.85%, and validation accuracy improved from 82.22% to 87.61%, suggesting effective learning.
6. The final model achieved a test accuracy of 88% using Adam optimizer and 86% using RMSprop optimizer.
7. The best validation loss was 0.1, observed when using tanh activation with L2 regularization and dropout.

**Procedures:**

1. **Data Preprocessing:**
   * Loaded the IMDB dataset and limited vocabulary to the 10,000 most frequent words.
   * Converted reviews into sequences of integers representing word indices.
   * Split data into **training (x\_train, y\_train) and test sets (x\_test, y\_test)**.
2. **Model Construction:**
   * Defined a neural network with **four dense layers** of **64 neurons** each.
   * Used **tanh** activation for hidden layers and **sigmoid** for the output layer.
   * Applied **L2 regularization (0.005) and dropout (0.5 rate)** for regularization.
3. **Compilation & Training:**
   * Used **Adam optimizer** with **MSE loss function**.
   * Trained for **20 epochs** with a **batch size of 512**.
   * Evaluated the model after each epoch on a validation set.
4. **Model Evaluation:**
   * Measured **test accuracy (88%) and test loss (0.1313)**.
   * Compared different configurations, optimizers, and activation functions.
5. **Comparison with RMSprop Optimizer:**
   * Used RMSprop optimizer to compare performance.
   * Observed a slightly lower **test accuracy (86%)**, indicating that Adam was better suited for this problem

**Comments:**

* The use of **L2 regularization and dropout layers** helped in reducing overfitting, leading to stable model performance.
* The **tanh activation function** performed better in most configurations compared to ReLU and binary cross-entropy (BCE).
* The Adam optimizer provided the highest test accuracy (**88%**), showing its suitability for sentiment classification tasks.
* The **batch size of 512** ensured efficient training without significantly increasing computational cost.
* Training and validation accuracy trends suggest that the model generalizes well to unseen data.

# Summary:

* During the same period, the training accuracy increases from 0.7329 in the first epoch to 0.9385 in the last epoch, while the validation accuracy increases from 0.8222 to 0.8761. This could suggest that the algorithm is learning how to properly classify the reviews rather than overfitting.
* During the same time period, the training loss drops from 0.5242 in the first epoch to 0.0781 in the last epoch, and the validation loss drops from 0.2947 to 0.1257. This shows that the algorithm is getting better at identifying and categorizing the reviews and is becoming more confident in its predictions.
* Based on its accuracy of 0.8761 on the validation set, the model is thought to be doing well overall on this task. It is crucial to keep in mind that the model's performance may vary based on the particular dataset and use case.
* According to the result, the model obtained a test accuracy of 0.8680 and a test loss of 0.1313.
* The model's test accuracy is represented by the second value, 0.8680, while the test loss is represented by the first value, 0.1313.
* The model's evaluation metrics on the test dataset following retraining are included in the results variable. The model's accuracy on the test dataset is shown by the second value in the list, 0.8727, while the first number, 0.1656, represents the MSE loss on the test dataset.
* In the test set, the optimizer "adam" has been utilized. Adam is an optimization technique with a number of appealing characteristics. It adjusts the learning rate for each component parameter based on past gradients, making it computationally efficient and impervious to the issue of assigning hyper-parameters. Using Adam Optimizer, my model's accuracy is 88%.
* RMSprop is an optimization approach that uses a running average of the gradient's recent magnitudes to modify the learning rate for each parameter. As a result, it avoids oscillations and damping when there are unsettling gradients present. RMSprop is frequently used for tasks that resemble image recognition, speech recognition, and natural language processing. The test set's dependability is 86% when using the rmsprop optimizer.

**Conclusions:**

* The **IMDB sentiment analysis model performed well** with an accuracy of **88%** using the Adam optimizer.
* **Tanh activation function with L2 regularization** and **dropout** produced the best validation loss (0.1), making it the optimal configuration.
* **Adam outperformed RMSprop**, showing that adaptive learning rate adjustments helped achieve better results.
* The **dropout layers improved generalization**, reducing overfitting.
* **The model was able to effectively classify movie reviews** as positive or negative with high accuracy and low loss values.

**Recommendations:**

1. **Hyperparameter Tuning:**
   * Experiment with different values for **L2 regularization**, **dropout rate**, and **learning rates** to further improve performance.
   * Try **smaller batch sizes** (e.g., 256 or 128) to observe their impact on model convergence.
2. **Alternative Architectures:**
   * Use **Recurrent Neural Networks (RNNs) or LSTMs**, which may improve accuracy by capturing word order and contextual dependencies.
   * Implement **bidirectional LSTMs** to process text sequences in both forward and backward directions.
3. **Different Optimizers:**
   * Test **SGD with momentum** and **Adagrad** to compare their performance against Adam and RMSprop.
4. **Fine-Tune Embeddings:**
   * Instead of using word indices, try **pre-trained embeddings like GloVe or Word2Vec** to provide richer contextual understanding.
5. **More Advanced Regularization:**
   * Apply **batch normalization** to stabilize training and improve generalization.
   * Implement **early stopping** to prevent unnecessary training when validation loss stops improving.
6. **Deploying the Model:**
   * Convert the model into a web API or mobile app for real-world sentiment analysis applications.
   * Evaluate performance on **real-world reviews** to ensure robustness in practical use cases.