LAB 9: Build a Recurrent Neural Network

AIM: To design, implement and evaluate a RNN model for sequential data, such as text and analyse its performance.

Pseudo Code:- Load the dataset

```
• ~~Pp~~ Preprocess data
  → Convert sequences into input-output pairs
• Define RNN model:
  → RNN layer + Dense output layer with activation
• Compile model:
  → Select optimizer
• Train Model:
  → Fit data into RNN for given epoch & batch size.
  → Monitor validation loss.
• Evaluate Model:
  → Test data
• Visualize results:
  → Plot accuracy & loss curves
• Conclude observation & Results
```

Observations

→ The training accuracy increases with epochs, while the loss decreases.

→ Overfitting can occur if too many epochs are used without regularization (dropout)

→ RNN captures seq. dependencies better than feed forward networks

→ LSTM variants perform more efficiently on long sequences due to vanishing gradient mitigation.

→ Validation performance depends on dataset complexity & preprocessing quality.

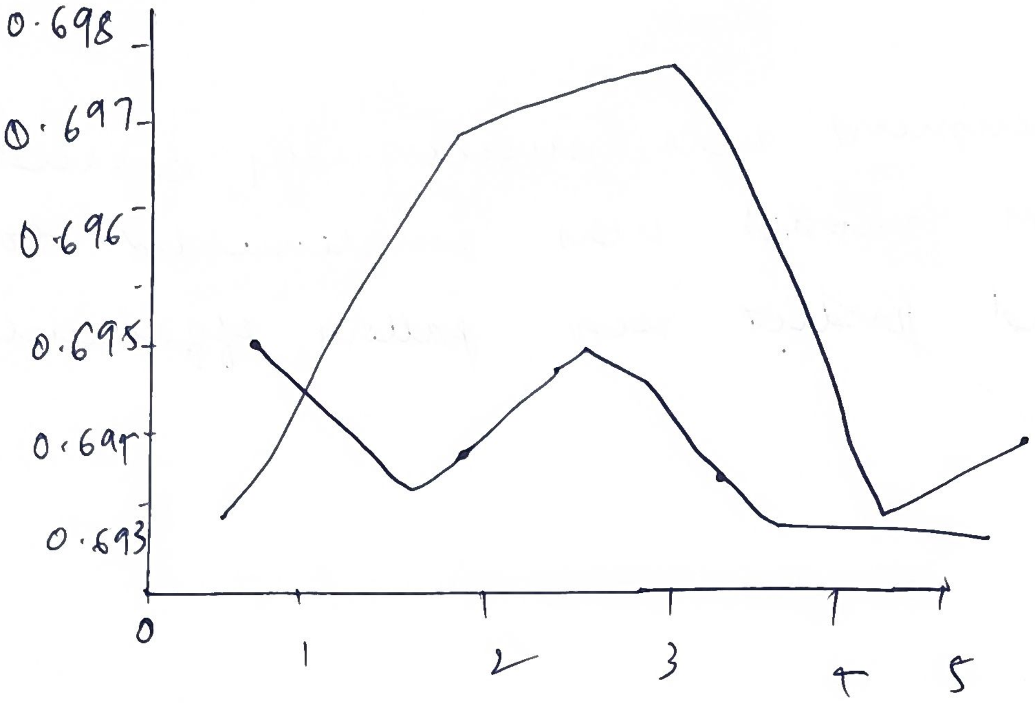
Result:

→ A RNN was successfully built and trained on sequential data.

19/10/25

O/P :-

Epoch v/s Test and Train Loss



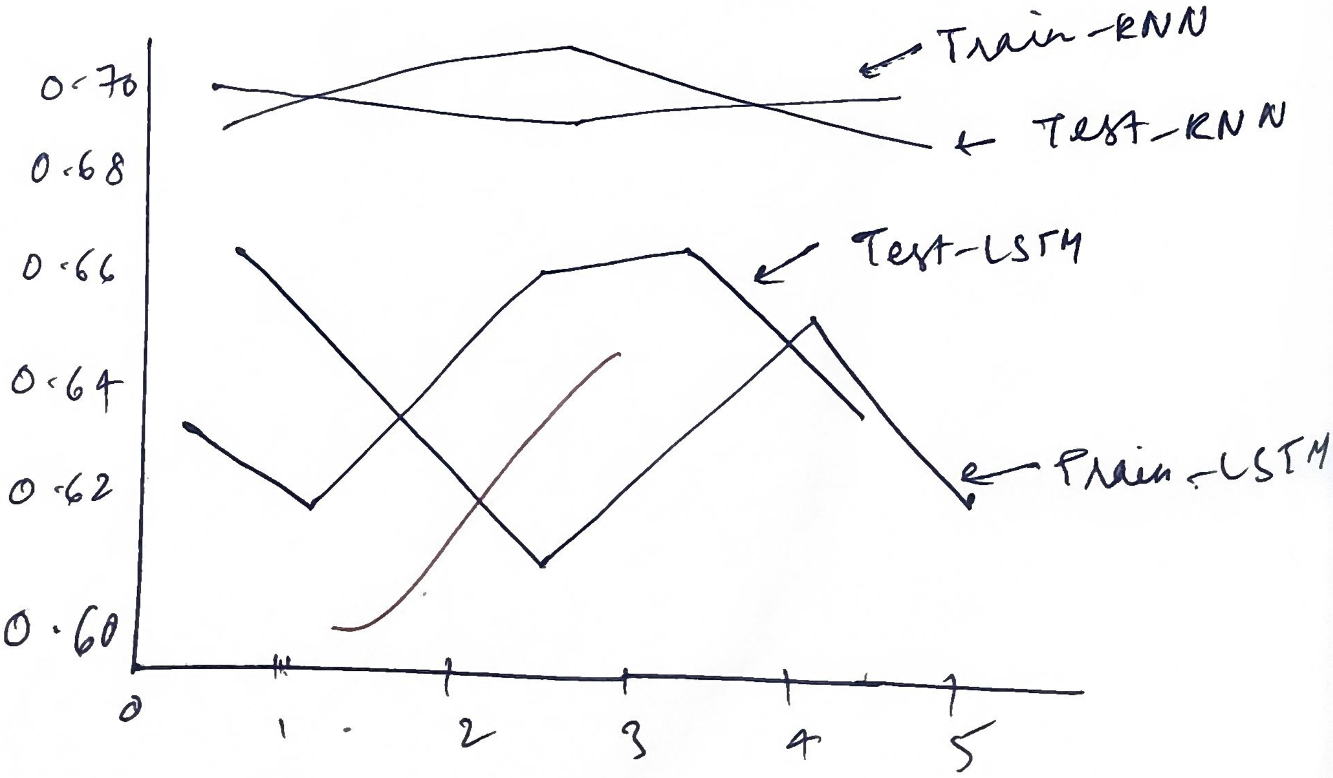
Accuracy : 61.15%

Prediction :-

I/P : Movie is good
O/P : Positive
I/P : movie was bad
O/P : negative

Comparision with LSTM

Train / Test vs Epoch



Output :-

Epoch 1 :- loss :- 0.6832 - Accuracy : 0.4938

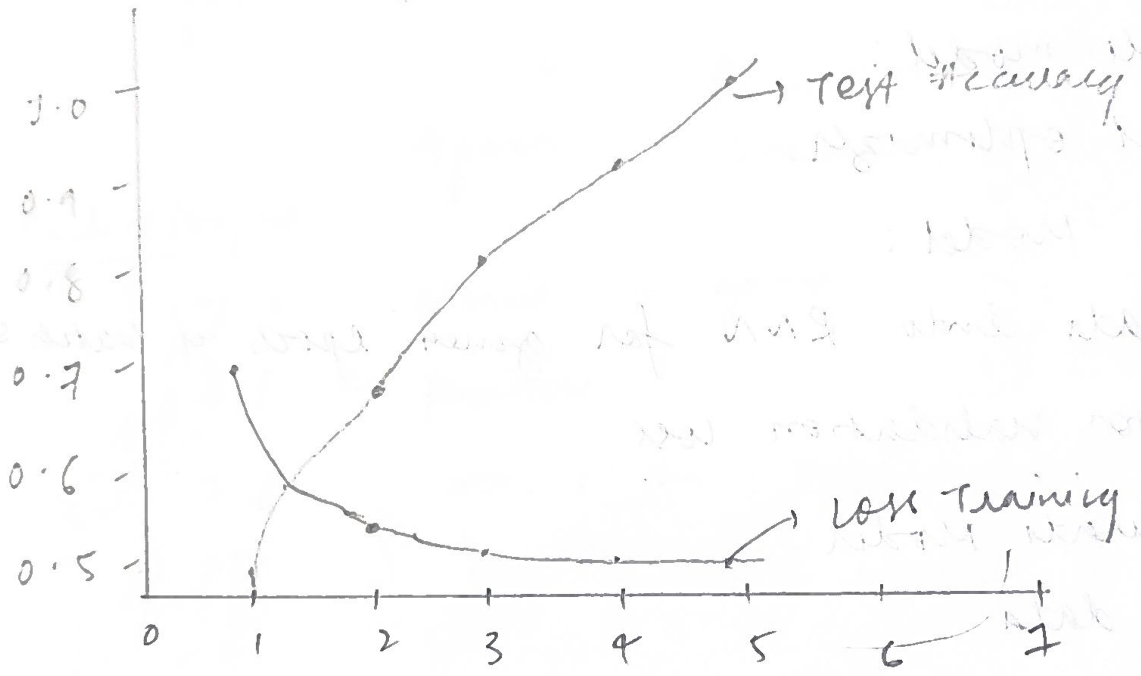
Epoch 2 :- loss - 0.0474 - Accuracy :- 0.6832

Epoch 3 :- loss - 0.0019 - Accuracy - 0.7848

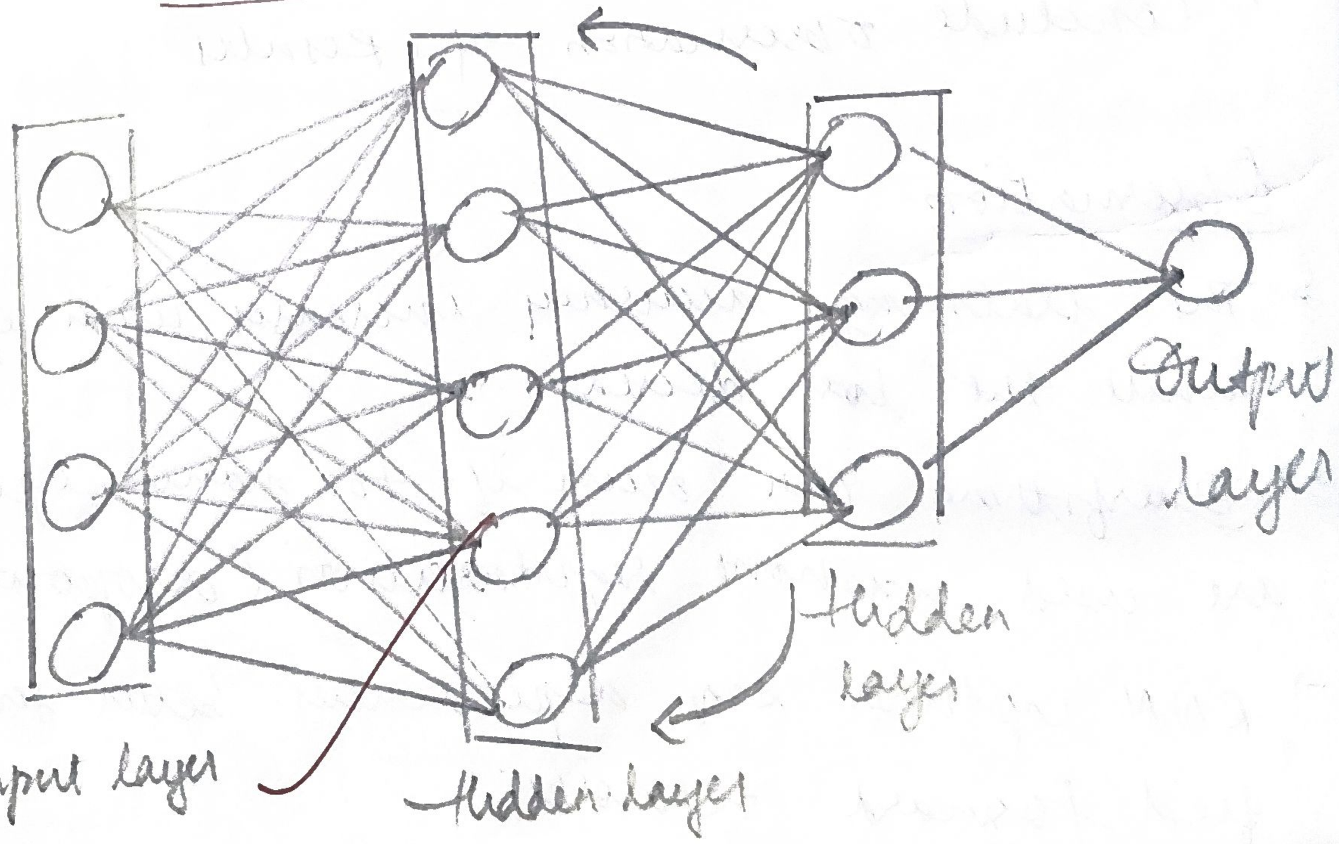
Epoch 4 :- loss - 0.0011 - Accuracy .- 0.8492

Epoch 5 :- loss - 0.0009 - Accuracy - 0.9631

Test & Training - Accuracy



RNN Architecture


Input layer
Hidden layer
Hidden layer
Output layer