📅 Day 17: Recurrent Neural Networks (RNN) and LSTM – Sentiment Analysis on IMDB Dataset

◆ What are Recurrent Neural Networks (RNNs)?

Recurrent Neural Networks (RNNs) are a type of artificial neural network specially designed for working with **sequential or time-series data**. Unlike traditional feedforward neural networks, RNNs retain memory of previous inputs using loops within their architecture.

This makes RNNs extremely useful in real-life applications such as:

- Text classification
- Speech recognition
- Language translation
- Stock price prediction
- Time series forecasting

◈ Working of RNNs:

RNNs have a "memory" that captures information about what has been calculated so far. At each step, the RNN takes an input and a hidden state and produces an output along with an updated hidden state. The same weights are shared across time steps.

However, basic RNNs struggle with long sequences due to issues like **vanishing gradients**, making it hard to learn long-term dependencies.

◈ Long Short-Term Memory (LSTM)

LSTM is a special kind of RNN that solves the limitations of basic RNNs by introducing **gates**:

- **Forget Gate**: Decides what information to discard.
- **Input Gate**: Decides what new information to store.
- **Output Gate**: Decides what to output.

This structure helps LSTMs **remember information for long periods**, making them suitable for tasks such as text understanding and sentiment analysis.

□ Practical Implementation: Sentiment Analysis using LSTM on IMDB Dataset

✅ Objective:

Build a Recurrent Neural Network (LSTM) that can predict whether a movie review from the IMDB dataset is **positive** or **negative**.

🔧 Step-by-Step Explanation:

1️⃣ Load the IMDB Dataset

```
from tensorflow.keras.datasets import imdb
vocab_size = 10000
(X_train, y_train), (X_test, y_test) =
imdb.load_data(num_words=vocab_size)
```

- IMDB dataset contains 50,000 reviews labeled as positive or negative.
- Only the top 10,000 most frequent words are kept.

2️⃣ Preprocess the Data

```
from tensorflow.keras.preprocessing.sequence import pad_sequences
max_length = 200
X_train = pad_sequences(X_train, maxlen=max_length)
X_test = pad_sequences(X_test, maxlen=max_length)
```

- The reviews (as word indexes) are padded to ensure **uniform length**.

3️⃣ Build the LSTM-based RNN Model

```
model =
tf.keras.Sequential([tf.keras.layers.Embedding(input_dim=vocab_size,
output_dim=64, input_length=max_length),
    tf.keras.layers.LSTM(64),
    tf.keras.layers.Dense(1, activation='sigmoid')
])
```

- **Embedding Layer**: Converts word indexes into dense vectors.
- **LSTM Layer**: Learns sequence-based patterns.
- **Dense Layer**: Outputs a single value (positive or negative sentiment).

4️⃣ Compile and Train the Model

```
model.compile(loss='binary_crossentropy', optimizer='adam',
metrics=['accuracy'])
history = model.fit(X_train, y_train, epochs=5, batch_size=64,
validation_split=0.2)
```

- **Binary Crossentropy** is used for binary classification.
- Trained with **80% data** and validated on 20%.

5️⃣ Evaluate the Model

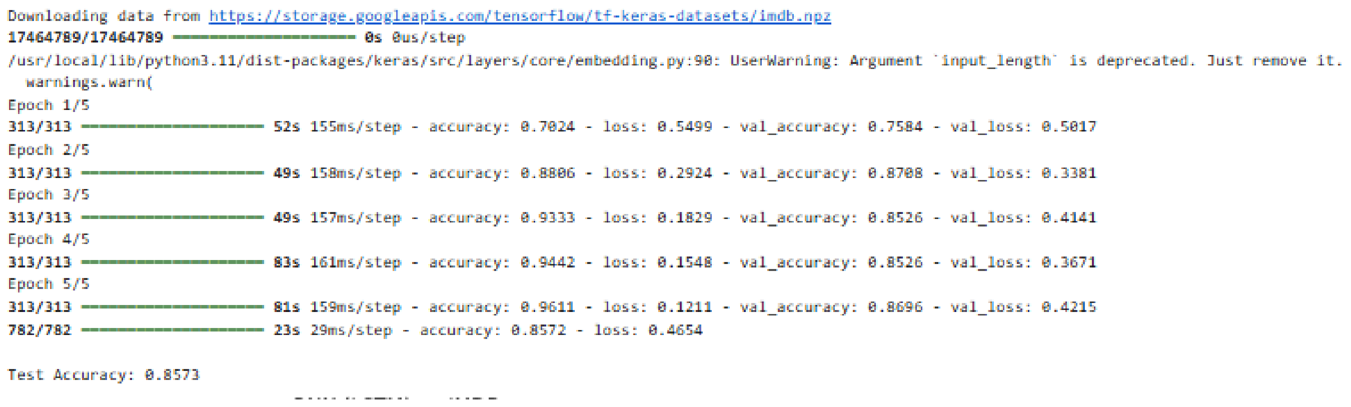
```
loss, acc = model.evaluate(X_test, y_test)
```

```
print(f"Test Accuracy: {acc:.4f}")
```

6️⃣ Visualize Accuracy During Training

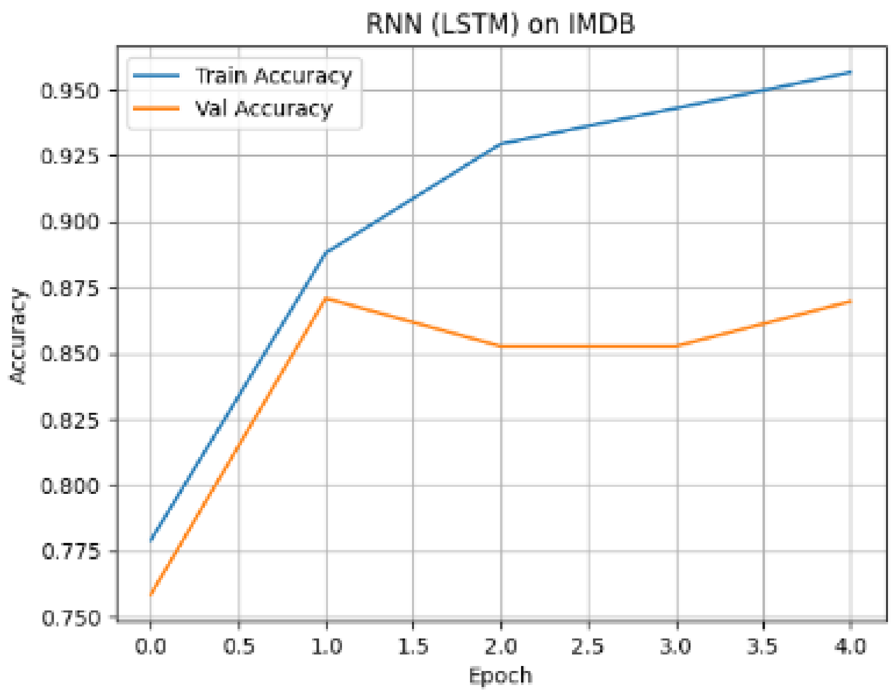
```
plt.plot(history.history['accuracy'], label='Train Accuracy')
plt.plot(history.history['val_accuracy'], label='Val Accuracy')
plt.xlabel("Epoch")
plt.ylabel("Accuracy")
plt.title("RNN (LSTM) on IMDB")
plt.legend()
plt.grid(True)
plt.show()
```

OUTPUTS:-



```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
/usr/local/lib/python3.11/dist-packages/keras/src/layers/core/embedding.py:90: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
Epoch 1/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 52s 155ms/step - accuracy: 0.7024 - loss: 0.5499 - val_accuracy: 0.7584 - val_loss: 0.5017
Epoch 2/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 49s 158ms/step - accuracy: 0.8806 - loss: 0.2924 - val_accuracy: 0.8708 - val_loss: 0.3381
Epoch 3/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 49s 157ms/step - accuracy: 0.9333 - loss: 0.1829 - val_accuracy: 0.8526 - val_loss: 0.4141
Epoch 4/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 83s 161ms/step - accuracy: 0.9442 - loss: 0.1548 - val_accuracy: 0.8526 - val_loss: 0.3671
Epoch 5/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 81s 159ms/step - accuracy: 0.9611 - loss: 0.1211 - val_accuracy: 0.8696 - val_loss: 0.4215
782/782 ━━━━━━━━━━━━━━━━━━━━ 23s 29ms/step - accuracy: 0.8572 - loss: 0.4654

Test Accuracy: 0.8573
```



📌 Results:

- Achieved good accuracy in predicting sentiment using LSTM.
- Visualized how training and validation accuracy evolved with epochs.

📈 Key Learnings:

- Understood the difference between basic RNN and LSTM.
- Applied LSTM to real-world dataset (IMDB) for sentiment analysis.
- Understood the power of sequence models in text data.

✅ Conclusion:

Recurrent Neural Networks, especially LSTMs, are effective tools for handling sequential data such as text. This session gave us hands-on experience in building a sentiment analysis model using LSTM with TensorFlow and demonstrated the power of deep learning for Natural Language Processing (NLP) tasks.