9/12/25

WEEK-8

Aim:- To implement an LSTM model for predicting sequential data and analyze its performance on a time-series dataset

Objective:-

1.) understand the working of LSTM Network.
2.) Train an LSTM model on sequential time series data.
3.) Evaluate prediction accuracy and observe the learning behaviour over epochs.

Algorithm:-

→ **Data preprocessing:-**
Normalize data and split into train/test sets.

→ **Model Design:-**
Create LSTM layers with Input, hidden and output Layers.

→ **Training:-**
Feed sequences into LSTM, compute Loss and optimize weights using back propagation through time

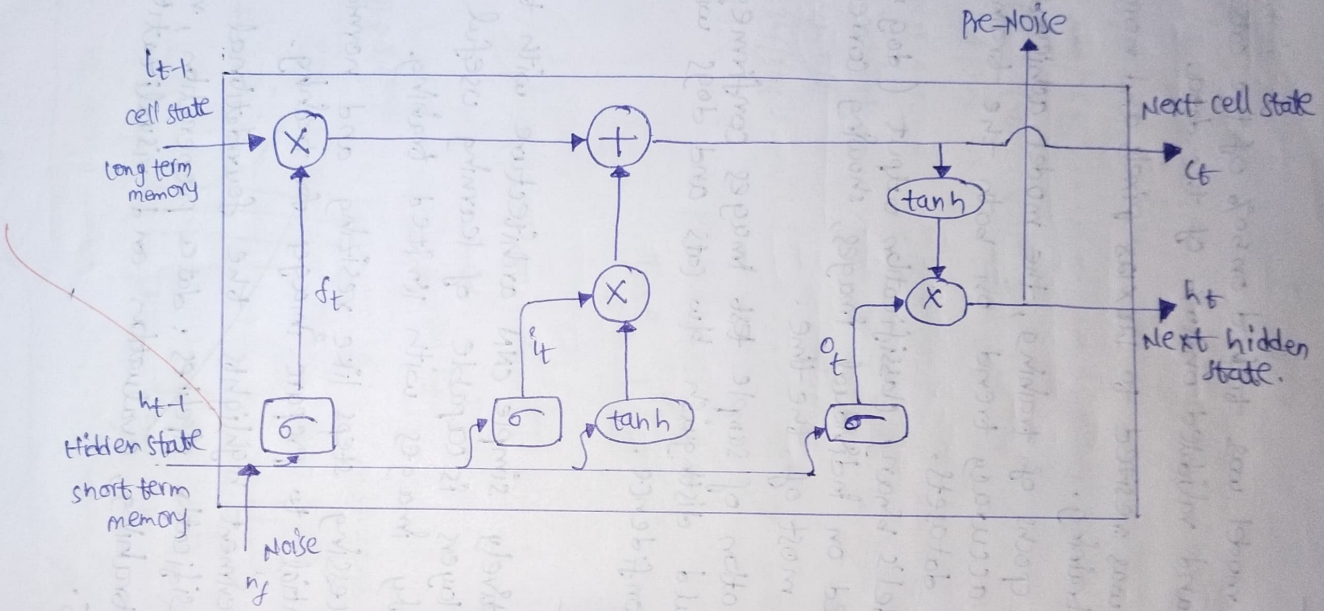
→ **Prediction:-**
Use the trained model to predict the future values.

→ **Evaluation:-** compare predictions with actual values using metrics like MSE (or) RMSE.

Pseudocode:-

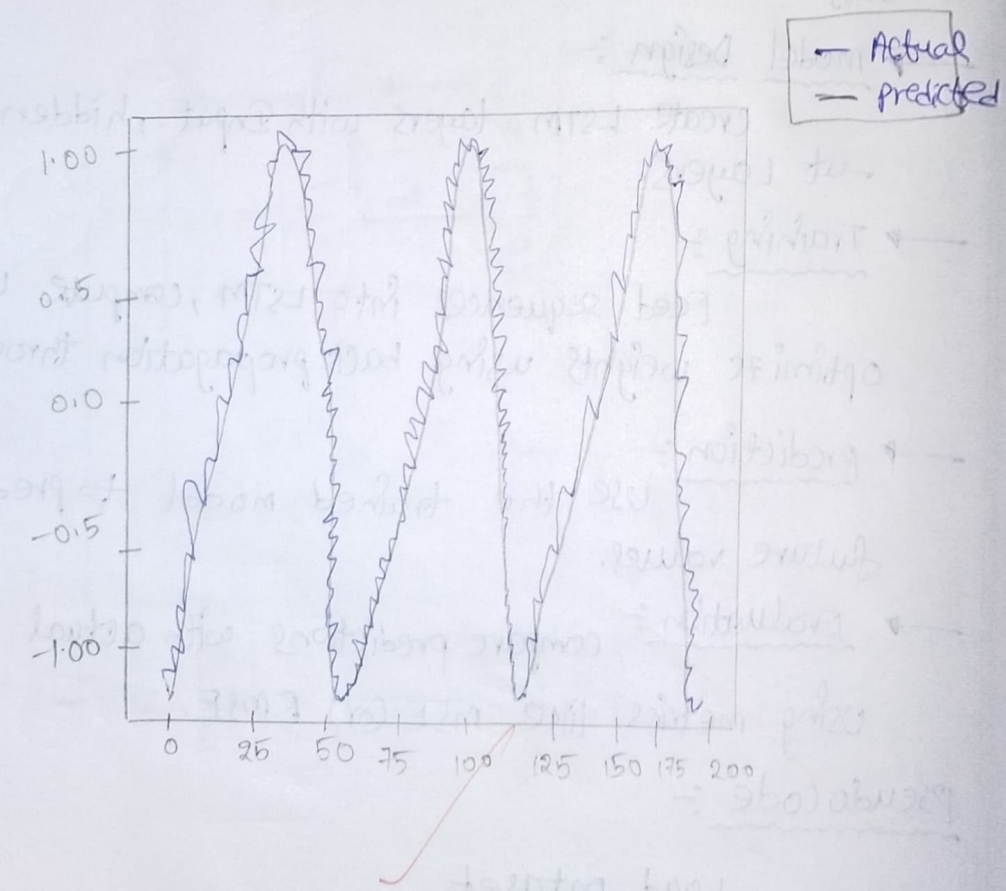
Load Dataset
Normalize data
split dataset into train and test
Initialize LSTM model
for each epoch

The Structure of LSTM model



output :-

Epoch	step-loss	val_loss
1	0.2998	0.1087
2	0.0705	0.0050
3	0.0044	0.0011
4	$7.9828e^{-04}$	$3.5373e^{-04}$
5	$3.9283e^{-04}$	$3.0203e^{-04}$
6	$2.1934e^{-04}$	$1.3776e^{-04}$
7	$1.6286e^{-04}$	$1.0536e^{-04}$



For each batch in training data
predict output
compute Loss
Back propagate Loss
Update weights
predict on test data
Denormalize predictions
calculate evaluation metrics (MSE / RMSE)

observation :-

*) Loss decreases gradually over epochs.

*) LSTM captures temporal dependencies better than sample RNN's.

*) predictions closely follow the trend of actual data but may slightly lag behind sharp changes.

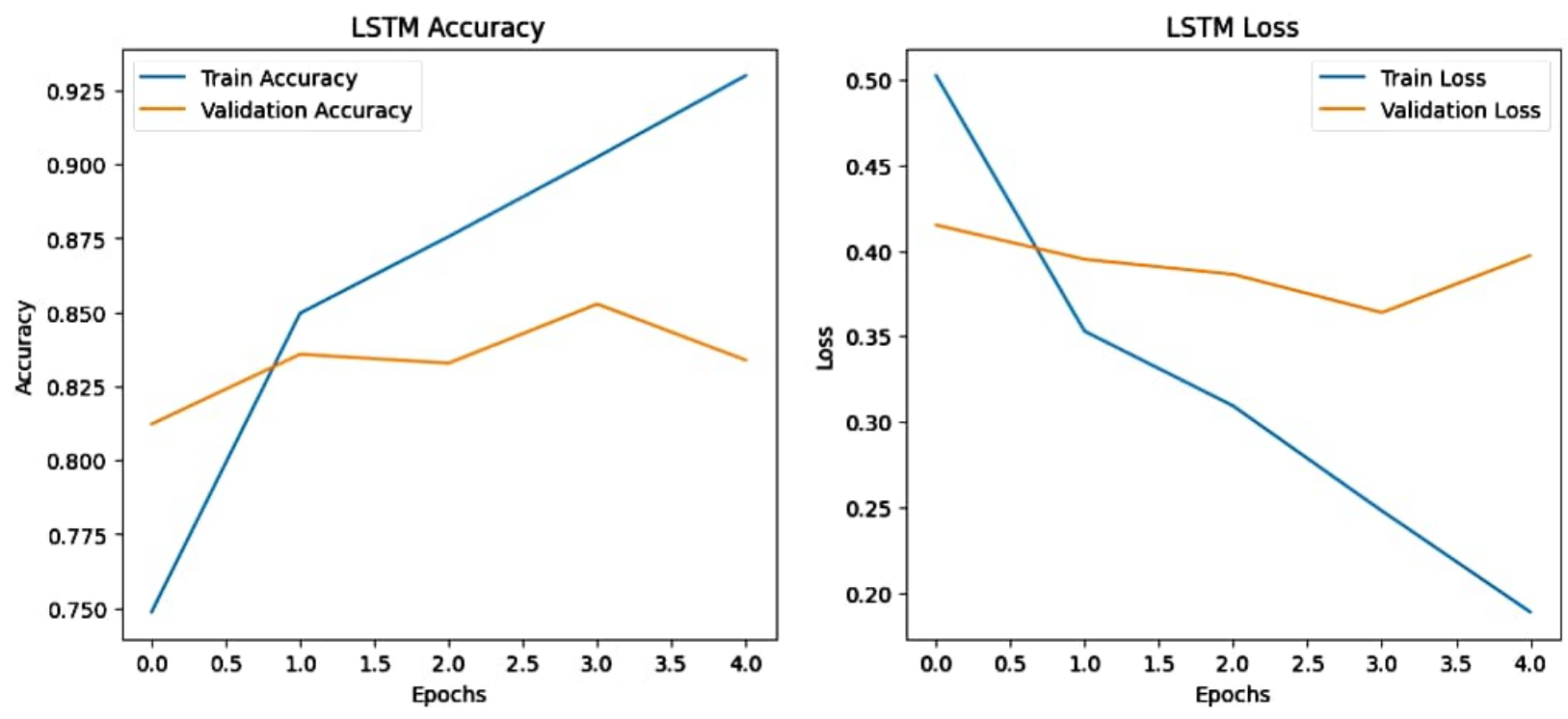
conclusion :-

*) LSTM is effective for sequential and time series.

*) It overcomes the vanishing gradient problems of traditional RNN's.

```python
import tensorflow as tf
from tensorflow.keras.datasets import imdb
from tensorflow.keras.preprocessing.sequence import pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, LSTM, Dense, Dropout
import matplotlib.pyplot as plt
# Load IMDB dataset
vocab_size = 10000
maxlen = 200
(x_train, y_train), (x_test, y_test) = imdb.load_data(num_words=vocab_size)
x_train = pad_sequences(x_train, maxlen=maxlen)
x_test = pad_sequences(x_test, maxlen=maxlen)
# Build LSTM model
lstm_model = Sequential([
    Embedding(input_dim=vocab_size, output_dim=128, input_length=maxlen),
    LSTM(128, dropout=0.2, recurrent_dropout=0.2),
    Dense(1, activation='sigmoid')
])
lstm_model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
# Train model
lstm_history = lstm_model.fit(
    x_train, y_train,
    epochs=5,
    batch_size=64,
    validation_split=0.2
)
# Evaluate model
lstm_score = lstm_model.evaluate(x_test, y_test)
print(f"LSTM Test Accuracy: {lstm_score[1]*100:.2f}% | Loss: {lstm_score[0]:.4f}")
# Plot Accuracy and Loss
plt.figure(figsize=(12,5))
# Accuracy
plt.subplot(1,2,1)
plt.plot(lstm_history.history['accuracy'], label='Train Accuracy')
plt.plot(lstm_history.history['val_accuracy'], label='Validation Accuracy')
plt.title('LSTM Accuracy')
plt.xlabel('Epochs')
plt.ylabel('Accuracy')
plt.legend()
# Loss
plt.subplot(1,2,2)
plt.plot(lstm_history.history['loss'], label='Train Loss')
plt.plot(lstm_history.history['val_loss'], label='Validation Loss')
plt.title('LSTM Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

```
Downloading data from https://storage.googleapis.com/tensorflow/tf-keras-datasets/imdb.npz
17464789/17464789 ━━━━━━━━━━━━━━━━━━━━ 0s 0us/step
Epoch 1/5
/usr/local/lib/python3.12/dist-packages/keras/src/layers/core/embedding.py:97: UserWarning: Argument `input_length` is deprecated. Just remove it.
  warnings.warn(
313/313 ━━━━━━━━━━━━━━━━━━━━ 193s 603ms/step - accuracy: 0.6676 - loss: 0.5828 - val_accuracy: 0.8122 - val_loss: 0.4151
Epoch 2/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 198s 593ms/step - accuracy: 0.8517 - loss: 0.3529 - val_accuracy: 0.8358 - val_loss: 0.3951
Epoch 3/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 206s 606ms/step - accuracy: 0.8840 - loss: 0.2872 - val_accuracy: 0.8328 - val_loss: 0.3862
Epoch 4/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 204s 611ms/step - accuracy: 0.9019 - loss: 0.2525 - val_accuracy: 0.8528 - val_loss: 0.3639
Epoch 5/5
313/313 ━━━━━━━━━━━━━━━━━━━━ 198s 598ms/step - accuracy: 0.9328 - loss: 0.1832 - val_accuracy: 0.8338 - val_loss: 0.3972
782/782 ━━━━━━━━━━━━━━━━━━━━ 62s 79ms/step - accuracy: 0.8309 - loss: 0.4013
LSTM Test Accuracy: 83.48% | Loss: 0.3946
```


LSTM Accuracy
Train Accuracy
Validation Accuracy
Accuracy
Epochs
LSTM Loss
Train Loss
Validation Loss
Loss
Epochs