D. Khaja Nawaz

RA2311047010037

Deep Learning Techniques (LAB)

Date	Title	Sign
24/07/2025	1. Exploring the deep learning platform	
31/07/25	2. Implement a classifier using open-source dataset	
31/07/25	3. STudy of the classifiers with respect to statistical parameters	
14/08/25	4. Build a simple feed forword neural network to recognize handwritten character	
22/08/25	5. STudy of Activation functions and their role	
09/09/25	6. Implement gradient descent and backpropagation in deep neural network	
16/09/25	7. Build a CNN model to classify Cat and Dog image	
30/09/2025	8. Experiment of LSTM	
30/09/2025	9. Build a Recurrent Neural Network	

30/09/25 Experiment of LSTM

*Aim :-

To build and train a long short-term memory (LSTM) based Recurrent Neural Network

*objective :-

1. To understand the working of LSTM networks.
2. To process sequential data and feed it to an LSTM model
3. To implement, train, and evaluate an LSTM model using pytorch
4. To analyze model performance using loss and accuracy metrics

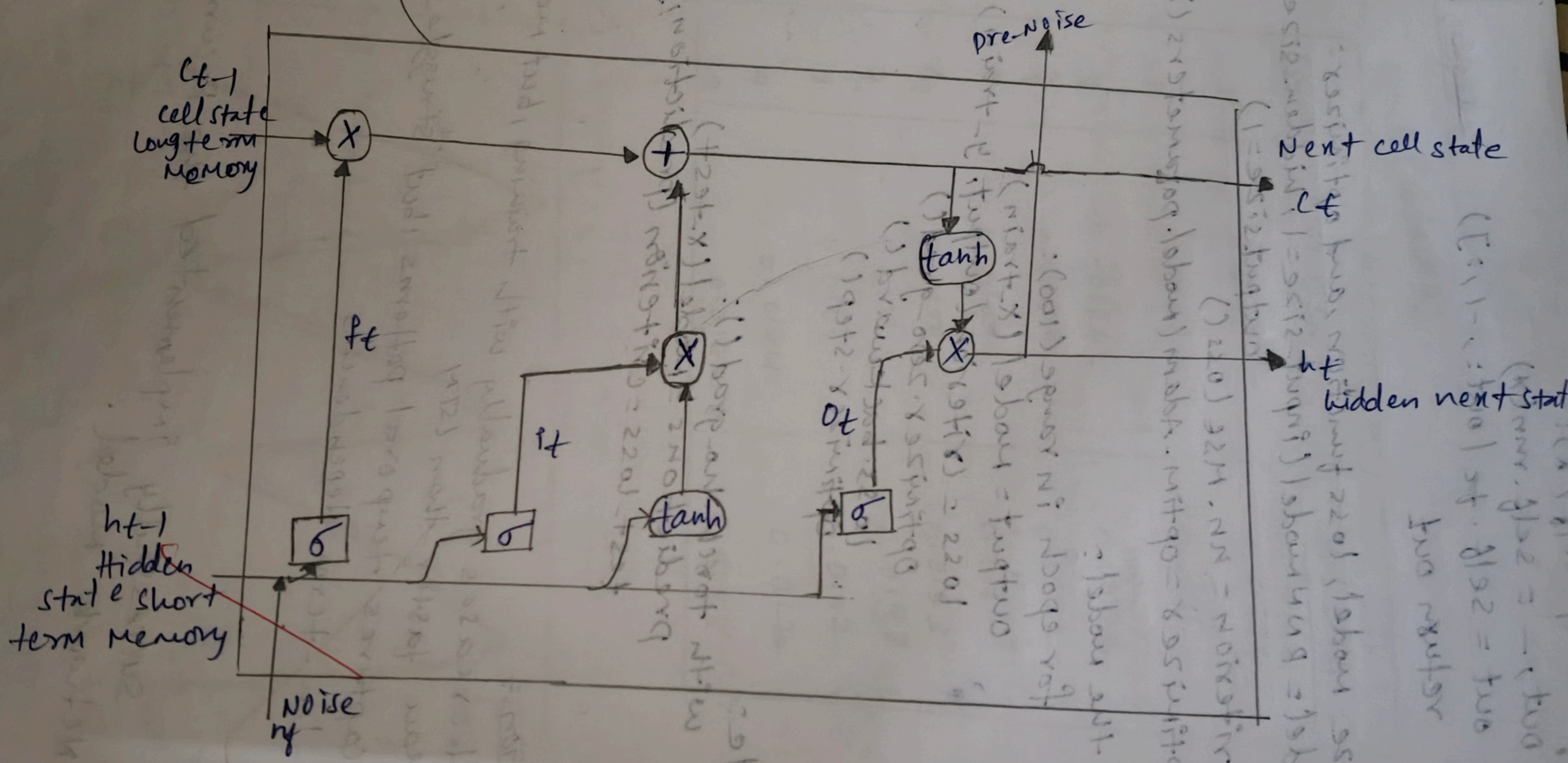
*pseudocode :-

1. Import libraries :
   import torch, torch.nn, torch.optim, numpy

2. prepare dataset :
   → Generate sinewave or sequential data
   → Create sequences (X, y) for supervised learning
   → convert to pytorch tensors

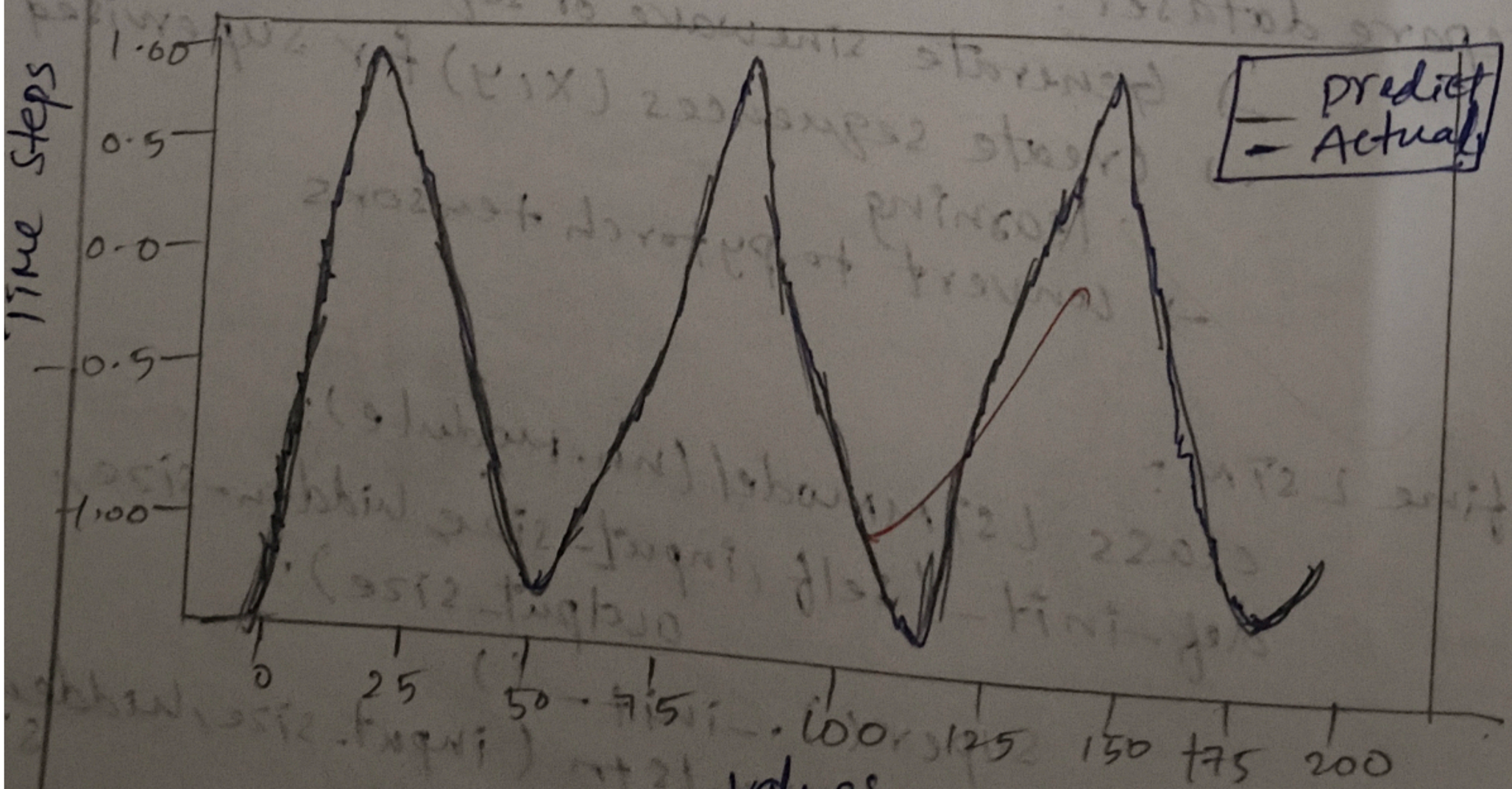
3. Define LSTM :
   class LSTMModel(nn.Module):
   def __init__(self, input_size, hidden_size, output_size):
   super().__init__()
   self.lstm = nn.lstm(input_size, hidden_siz

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	7.982 e-04	3.5373e-04
5	3.9283e-04	3.0203e-04
6	2.4878e-04	2.3451e-04
7	2.9676e-04	1.7862e-04
8	2.5063e-04	2.2236e-04
9	2.1934 e-04	1.3776e-04
10	1.6286e-04	1.0536e-04



```
self.fc = nn.Linear(hidden_size, output_size)

def forward(self, x):
    out, _ = self.lstm(x)
    out = self.fc(out[:, -1, :])
    return out
```

4. Initialize model, loss, optimizer:

```
model = LSTMModel(1, 64, 1)
criterion = nn.MSELoss()
optimizer = Adam(model.parameters())
```

5. Train model:

For each epoch:

→ Forward pass
→ compute loss
→ Backward pass
→ update weights

6. Evaluate model:

→ predict test data
→ compute MAE, RMSE, R^2 score
→ plot actual vs predicted

*Observation:-

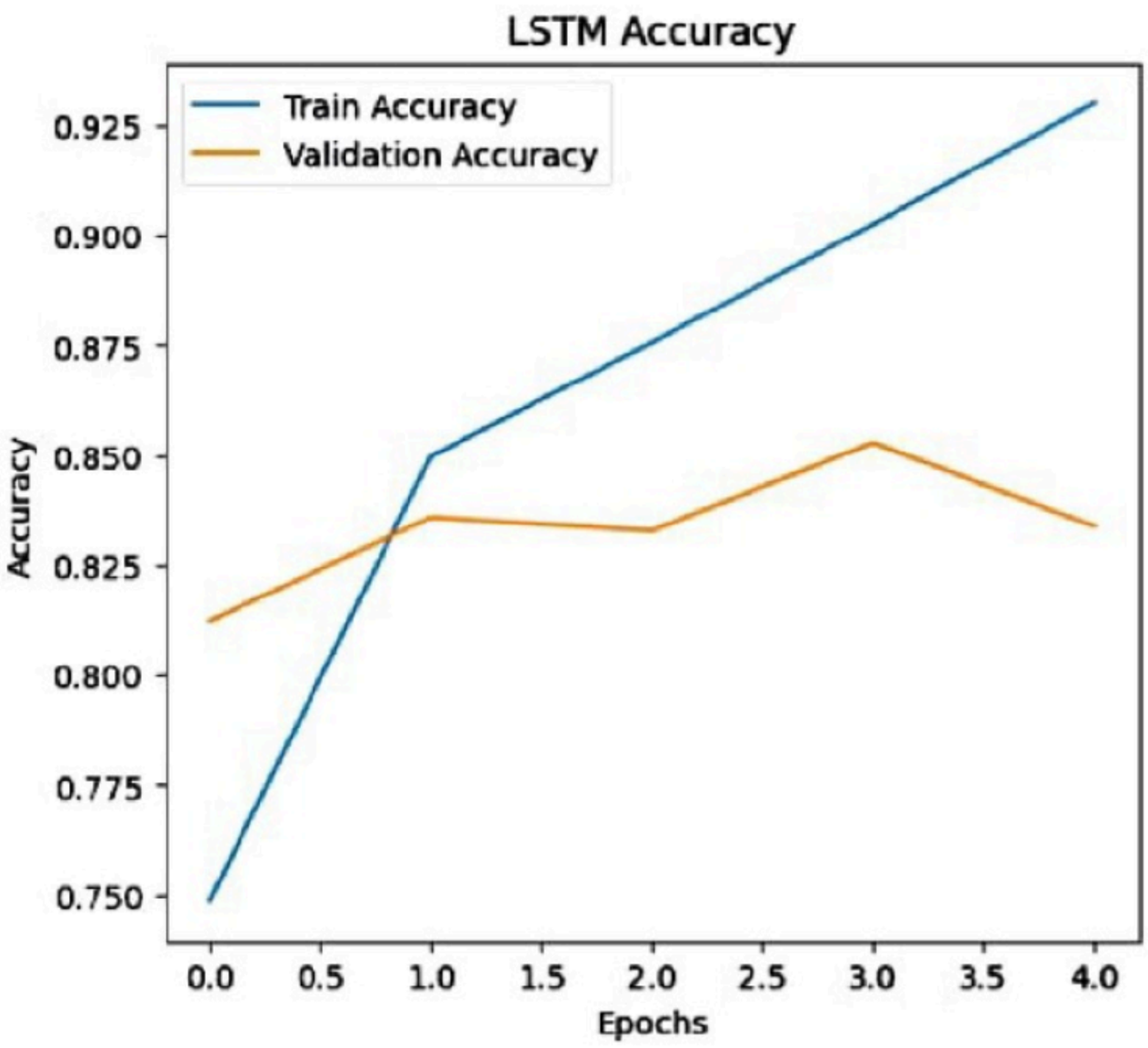
→ loss decreases gradually over epochs
→ LSTM captures temporal dependencies better than simple RNNs.
→ predictions closely follow the trend of actual data but may slightly lag behind sharp changes

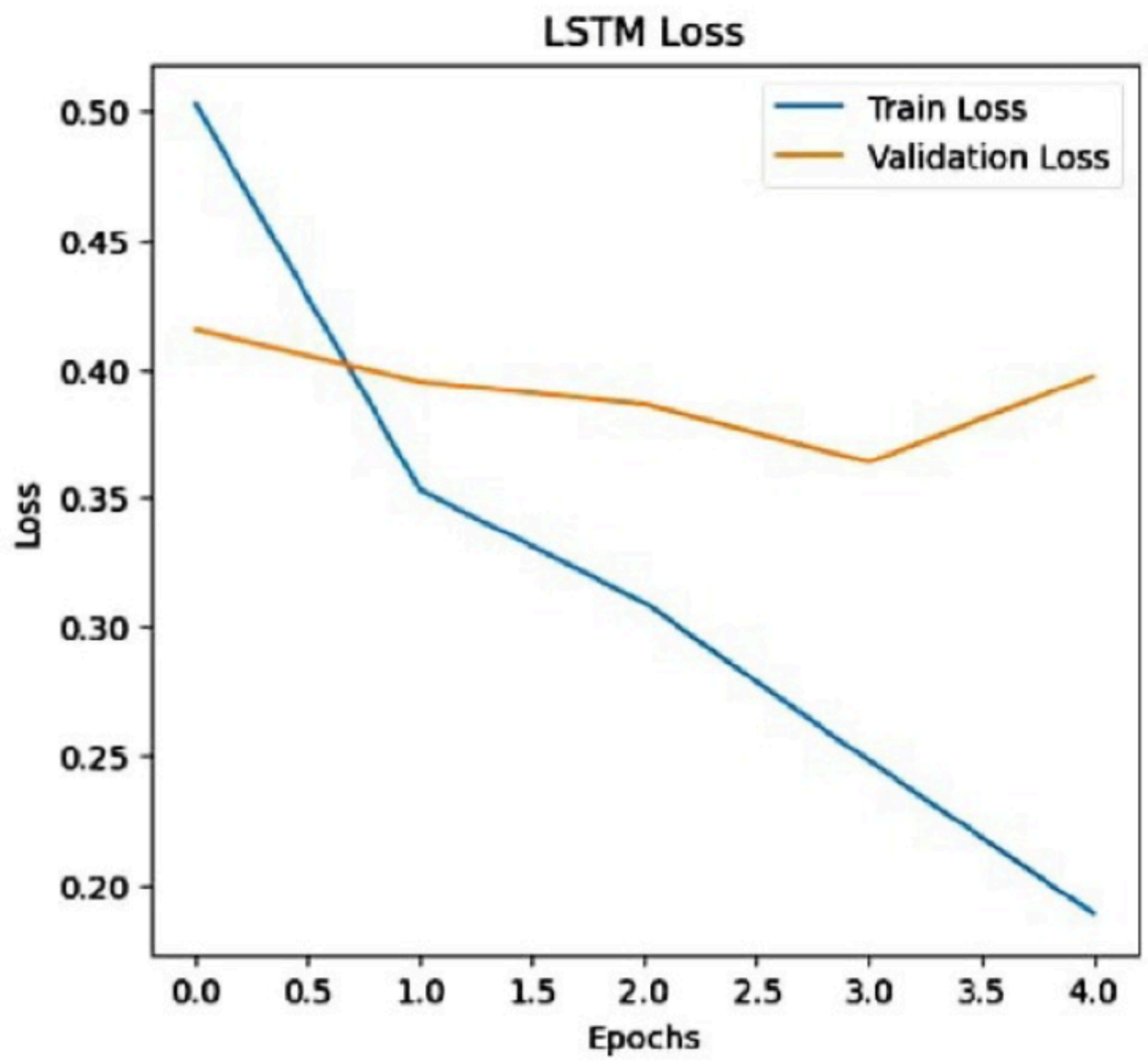
*Conclusion:-

LSTM is effective for sequential and time-series If it overcomes the vanishig gradient problem of traditional RNNs.

```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
0.925
0.900
0.875
0.850
0.825
0.800
0.775
0.750
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs


LSTM Loss
Train Loss
Validation Loss
Loss
0.50
0.45
0.40
0.35
0.30
0.25
0.20
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0
Epochs