

EXP No: 8 :- Experiment Using LSTM

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 behavior over epochs.

Algorithm:-

1. Data preprocessing: Normalize data and split into train / test sets.
2. Model Design: Create LSTM layers with input, hidden, and output layers.
3. Training: Feed sequences into LSTM, compute loss, and optimize weights using backpropagation through time.
4. Prediction: Use the trained model to predict future values.
5. Evaluation: Compare predictions with actual values using metrics like MSE or RMSE.

Pseudo Code:

Load dataset

Normalize data

Split dataset into train and test

Initialize LSTM model

for each epoch:

for each batch in training data:

predict output

Compute loss

Backpropagate loss

update weights

Predict on test data

Denormalize predictions

Calculate evaluation metrics (MSE/RMSE)

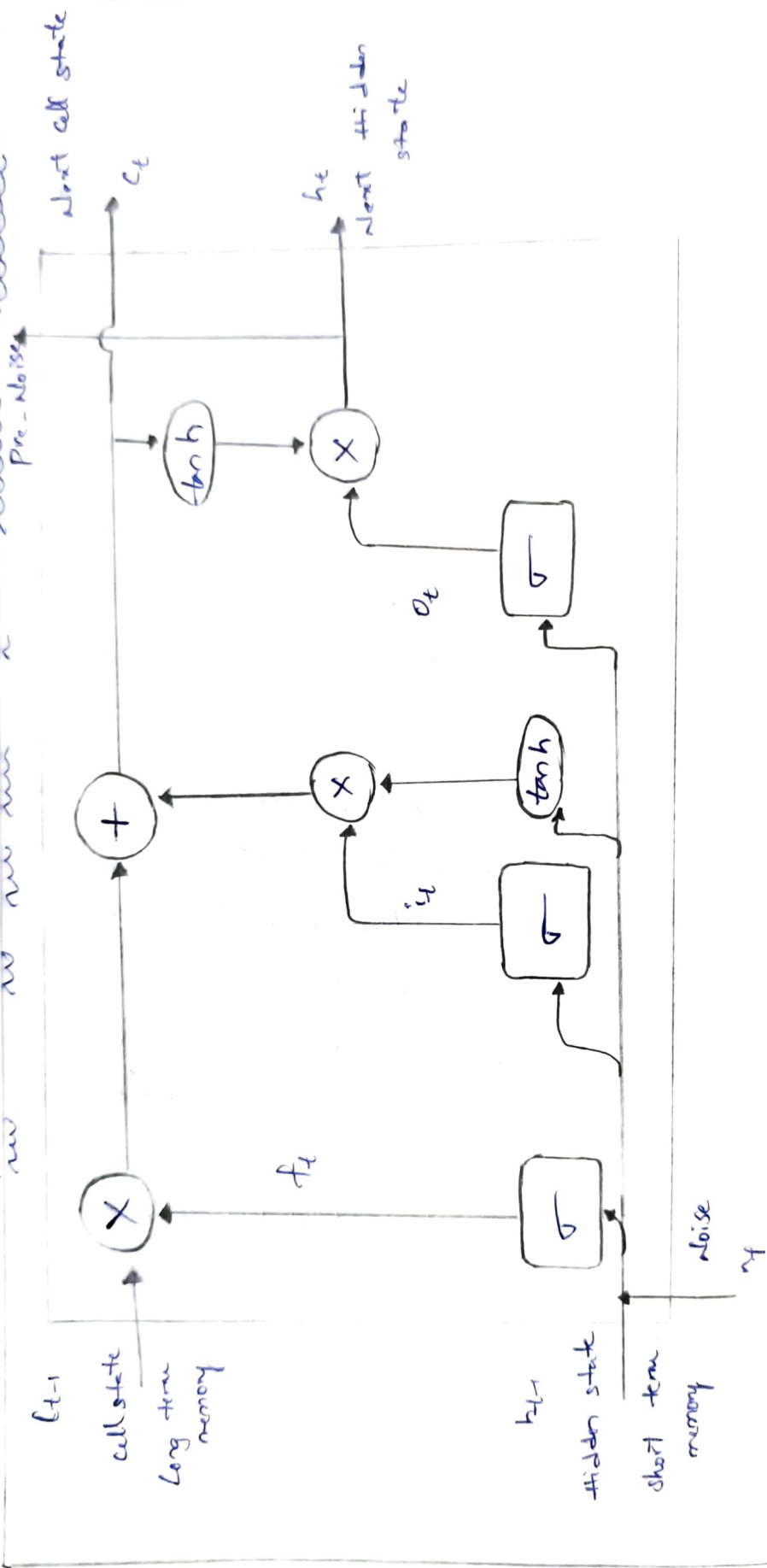
Observation:-

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

Conclusions:

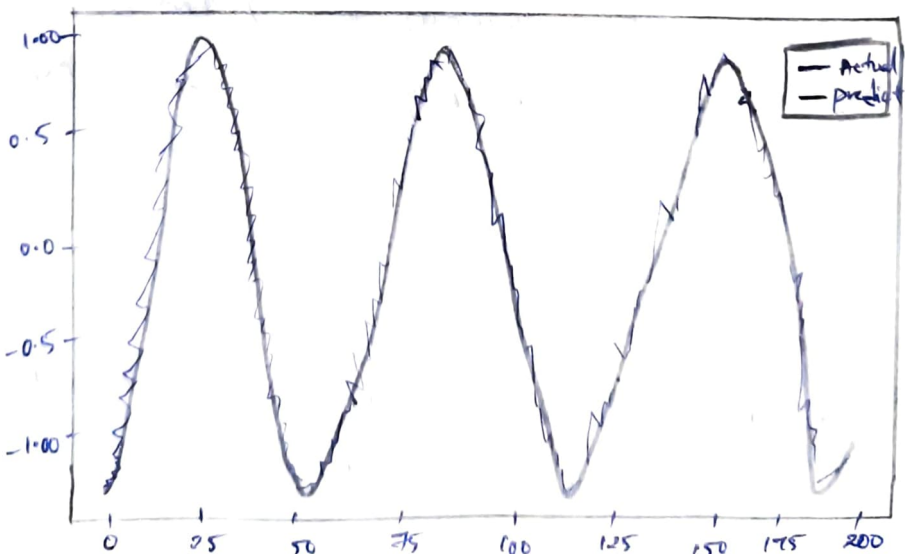
- LSTM is effective for sequential and time-series.
- it overcomes the vanishing gradient problem of traditional RNNs.

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 | 7.9828×10^{-4} | 3.5373×10^{-4} |
| 5 | 3.9283×10^{-4} | 3.0203×10^{-4} |
| 6 | 2.4878×10^{-4} | 2.3451×10^{-4} |
| 7 | 2.9676×10^{-4} | 1.7862×10^{-4} |
| 8 | 2.5063×10^{-4} | 2.2236×10^{-4} |
| 9 | 2.1934×10^{-4} | 1.3776×10^{-4} |
| 10 | 1.6286×10^{-4} | 1.0536×10^{-4} |



```
[ ] import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense
```

```
[ ] t = np.linspace(0, 100, 1000)
data = np.sin(t)
def create_sequences(data, seq_length):
    x, y = [], []
    for i in range(len(data)-seq_length):
        x.append(data[i:i+seq_length])
        y.append(data[i+seq_length])
    return np.array(x), np.array(y)
seq_length = 20
X, y = create_sequences(data, seq_length)
X = X.reshape((X.shape[0], X.shape[1], 1))
```

```
[ ] train_size = int(len(X)*0.8)
X_train, X_test = X[:train_size], X[train_size:]
y_train, y_test = y[:train_size], y[train_size:]
```

```
[ ] model = Sequential([
    LSTM(50, activation='tanh', input_shape=(seq_length,1)),
    Dense(1)
])
model.compile(optimizer='adam', loss='mse')
```

/usr/local/lib/python3.12/dist-packages/keras/src/layers/rnn/rnn.py:199: UserWarning: Do not pass an `input_shape`/`input_dim` argument to a layer. When using Sequential models, super().__init__(**kwargs)

```

▶ history = model.fit(
    X_train, y_train,
    epochs=20,
    batch_size=32,
    validation_split=0.1
)

```

```

↩ Epoch 1/20
23/23 ————— 2s 23ms/step - loss: 0.3686 - val_loss: 0.1371
Epoch 2/20
23/23 ————— 0s 11ms/step - loss: 0.0717 - val_loss: 0.0080
Epoch 3/20
23/23 ————— 0s 11ms/step - loss: 0.0047 - val_loss: 0.0011
Epoch 4/20
23/23 ————— 0s 11ms/step - loss: 8.0864e-04 - val_loss: 4.7188e-04
Epoch 5/20
23/23 ————— 0s 12ms/step - loss: 4.5733e-04 - val_loss: 3.2457e-04
Epoch 6/20
23/23 ————— 0s 11ms/step - loss: 3.5262e-04 - val_loss: 2.9506e-04
Epoch 7/20
23/23 ————— 0s 15ms/step - loss: 2.7902e-04 - val_loss: 2.3991e-04
Epoch 8/20
23/23 ————— 0s 18ms/step - loss: 2.4308e-04 - val_loss: 1.9712e-04
Epoch 9/20
23/23 ————— 0s 17ms/step - loss: 1.7848e-04 - val_loss: 1.4722e-04
Epoch 10/20
23/23 ————— 0s 17ms/step - loss: 1.4743e-04 - val_loss: 1.1760e-04
Epoch 11/20
23/23 ————— 1s 18ms/step - loss: 1.1176e-04 - val_loss: 1.1962e-04
Epoch 12/20
23/23 ————— 0s 15ms/step - loss: 9.2827e-05 - val_loss: 8.1773e-05
Epoch 13/20
23/23 ————— 0s 12ms/step - loss: 9.5071e-05 - val_loss: 7.8213e-05
Epoch 14/20
23/23 ————— 0s 12ms/step - loss: 6.5347e-05 - val_loss: 4.3287e-05
Epoch 15/20
23/23 ————— 0s 11ms/step - loss: 4.4903e-05 - val_loss: 4.0221e-05
Epoch 16/20
23/23 ————— 0s 12ms/step - loss: 3.7359e-05 - val_loss: 2.7925e-05
Epoch 17/20
23/23 ————— 0s 11ms/step - loss: 2.4863e-05 - val_loss: 2.3925e-05
Epoch 18/20
23/23 ————— 0s 11ms/step - loss: 2.0035e-05 - val_loss: 1.8149e-05
Epoch 19/20
23/23 ————— 0s 12ms/step - loss: 1.5852e-05 - val_loss: 1.0586e-05
Epoch 20/20
23/23 ————— 0s 11ms/step - loss: 1.0313e-05 - val_loss: 7.5582e-06

```

```
▶ y_pred = model.predict(x_test)
plt.figure(figsize=(10,5))
plt.plot(y_test, label='Actual', color='blue')
plt.plot(y_pred, label='Predicted (LSTM)', color='orange')
plt.title('LSTM Prediction on Sine Wave')
plt.xlabel('Time Step')
plt.ylabel('Value')
plt.legend()
plt.show()
```

7/7 — 0s 19ms/step

