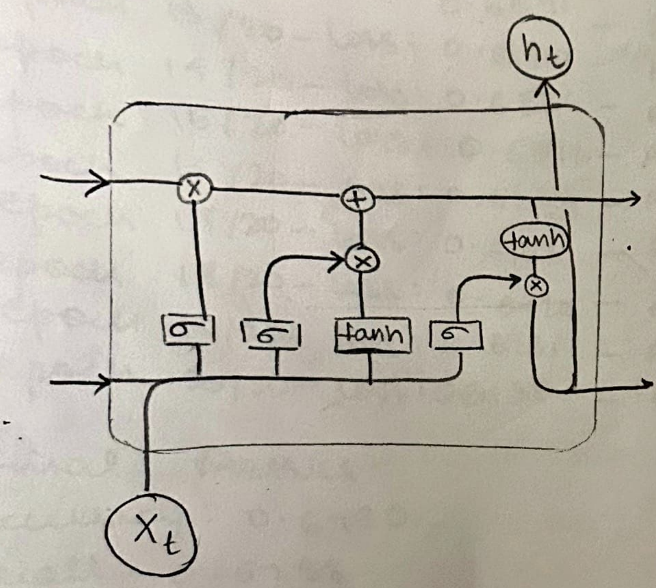
Exp 8:

Experiment using LSTM.

Aim: To impement and understand LSTM model in deep learning.

Pseudo code:

1. Import req. libraries
2. Load and preprocess text data
   - → Tokenize text
   - → convert to numerical sequences.
   - → Pad sequences for uniform length
   - → split into train & test sets.
3. Define LSTM model:
   - → Embedding layer
   - → LSTM layers
   - → Fully connected (linear) layers.
   - → sigmoid activation.
4. Define loss function & optimizer.
5. Train the model:
   - → loop through epochs
   - → Forward pass
   - → compute loss
   - → Backpropagation
   - → update weights
6. Evaluate on test data.
7. Print accuracy & loss.



LSTM

diagram

Justification:

LSTM (Long-Short Term memory) networks are a special type of Recurrent Neural Network (RNN) capable of learning long-term dependencies.

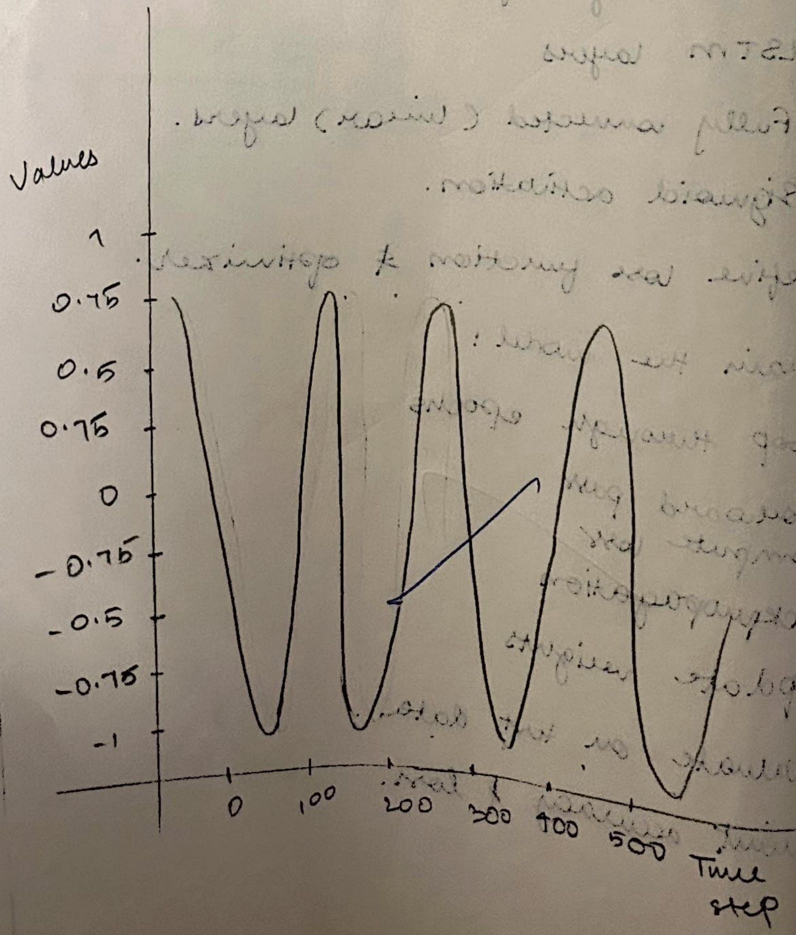
They are widely used in NLP tasks like:

- Sentiment Analysis
- Text generation
- Speech Recognition.

Result: Program implemented successfully.

Output:

Epoch [10, 100] , loss = 0.032186
Epoch [20, 100] , loss = 0.017898
Epoch [30 100] , loss = 0.018660
Epoch [40 , 100] , loss = 0.016763
Epoch [50 , 100] , loss = 0.015243
Epoch [60 , 100], loss = 0.014493
Epoch [70 , 100] , loss = 0.013756
Epoch [80 , 100], loss = 0.016046
Epoch [90, 100] , loss = 0.013142
Epoch [100, 100], loss = 0.012629



Untitled8.ipynb Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

RAM Disk

Share

[2]

```python
import torch
import torch.nn as nn
import torch.optim as optim
from sklearn.metrics import accuracy_score, f1_score, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(42)
X = np.random.rand(1000, 10, 1)
y = (X.sum(axis=1) > 5).astype(int).flatten()
X_train, X_test = torch.tensor(X[:800], dtype=torch.float32), torch.tensor(X[800:], dtype=torch.float32)
y_train, y_test = torch.tensor(y[:800], dtype=torch.long), torch.tensor(y[800:], dtype=torch.long)
class LSTMModel(nn.Module):
    def __init__(self, input_size=1, hidden_size=32, num_layers=1, num_classes=2):
        super(LSTMModel, self).__init__()
        self.lstm = nn.LSTM(input_size, hidden_size, num_layers, batch_first=True)
        self.fc = nn.Linear(hidden_size, num_classes)
    def forward(self, x):
        out, (hn, cn) = self.lstm(x)
        out = self.fc(hn[-1])
        return out

model = LSTMModel()
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
num_epochs = 6
train_losses, test_accuracies = [], []

for epoch in range(num_epochs):
    # Forward
```

Variables Terminal 2:41 PM Python 3

Untitled8.ipynb   Saving...

File Edit View Insert Runtime Tools Help

Commands + Code + Text ▷ Run all   RAM Disk

[2]

```python
optimizer = optim.Adam(model.parameters(), lr=0.001)
num_epochs = 6
train_losses, test_accuracies = [], []

for epoch in range(num_epochs):
    # Forward
    outputs = model(X_train)
    loss = criterion(outputs, y_train)

    # Backward
    optimizer.zero_grad()
    loss.backward()
    optimizer.step()

    # Evaluate on test set
    with torch.no_grad():
        test_outputs = model(X_test)
        _, predicted = torch.max(test_outputs, 1)
        acc = accuracy_score(y_test, predicted)

    train_losses.append(loss.item())
    test_accuracies.append(acc)
    print(f"Epoch [{epoch+1}/{num_epochs}] Loss: {loss.item():.4f}, Test Acc: {acc:.4f}")
with torch.no_grad():
    y_pred = torch.argmax(model(X_test), dim=1)
    acc = accuracy_score(y_test, y_pred)
    f1 = f1_score(y_test, y_pred)
    cm = confusion_matrix(y_test, y_pred)

print("\n✅ Final Metrics:")
print(f"Accuracy: {acc:.4f}")
print(f"F1 Score: {f1:.4f}")
```

Variables Terminal   ✓ 2:41 PM   Python 3

Untitled8.ipynb   Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

RAM
Disk

[2]
```python
print("\n✅ Final Metrics:")
print(f"Accuracy: {acc:.4f}")
print(f"F1 Score: {f1:.4f}")
print("Confusion Matrix:\n", cm)
plt.figure(figsize=(10,4))

plt.subplot(1,2,1)
plt.plot(train_losses, label="Train Loss")
plt.title("Training Loss")
plt.xlabel("Epochs")
plt.ylabel("Loss")
plt.legend()

plt.subplot(1,2,2)
plt.plot(test_accuracies, label="Test Accuracy", color='orange')
plt.title("Test Accuracy")
plt.xlabel("Epochs")
plt.ylabel("Accuracy")
plt.legend()

plt.tight_layout()
plt.show()
```

```
Epoch [1/6] Loss: 0.6975, Test Acc: 0.4300
Epoch [2/6] Loss: 0.6967, Test Acc: 0.4300
Epoch [3/6] Loss: 0.6959, Test Acc: 0.4300
Epoch [4/6] Loss: 0.6952, Test Acc: 0.4300
Epoch [5/6] Loss: 0.6945, Test Acc: 0.4300
Epoch [6/6] Loss: 0.6939, Test Acc: 0.4300

✅ Final Metrics:
Accuracy: 0.4300
```

Variables Terminal 2:41 PM

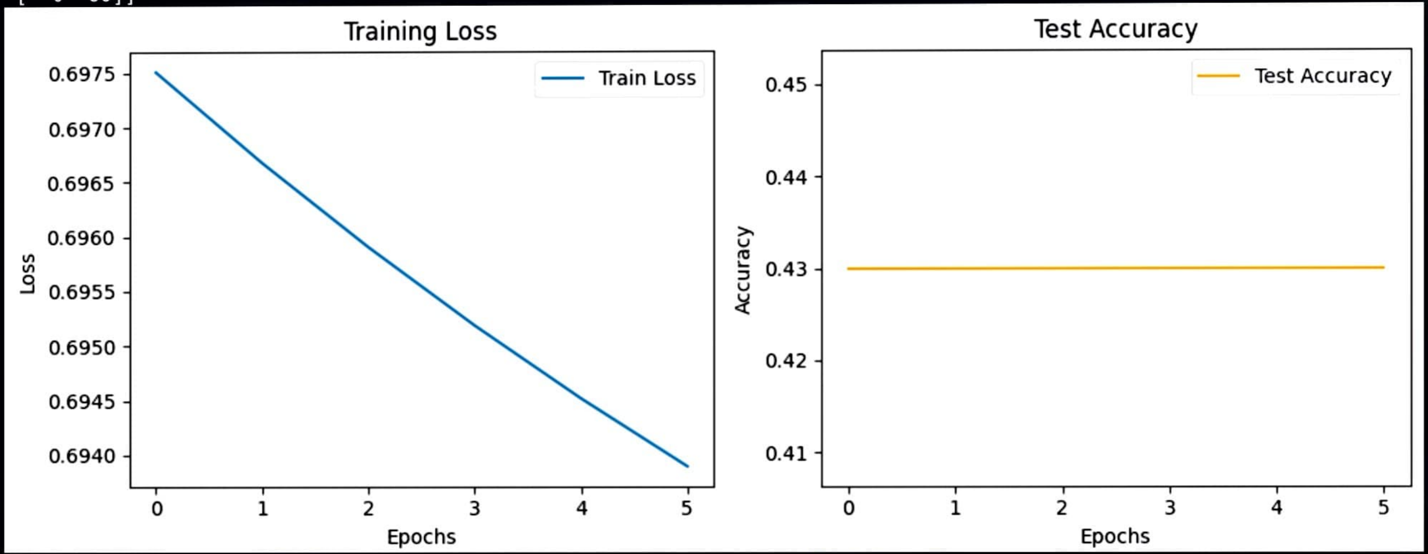
Untitled8.ipynb Saving failed since 2:39 PM

File Edit View Insert Runtime Tools Help

Commands + Code + Text Run all

```
Epoch [1/6] Loss: 0.6975, Test Acc: 0.4300
Epoch [2/6] Loss: 0.6967, Test Acc: 0.4300
Epoch [3/6] Loss: 0.6959, Test Acc: 0.4300
Epoch [4/6] Loss: 0.6952, Test Acc: 0.4300
Epoch [5/6] Loss: 0.6945, Test Acc: 0.4300
Epoch [6/6] Loss: 0.6939, Test Acc: 0.4300

✅ Final Metrics:
Accuracy: 0.4300
F1 Score: 0.6014
Confusion Matrix:
 [[  0 114]
 [  0  86]]
```



Variables Terminal ✓ 2:41 PM