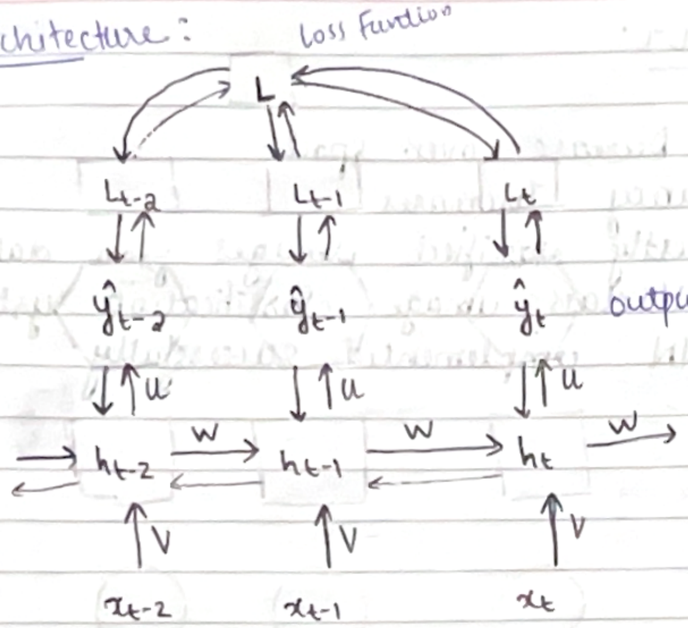
RNN architecture:



$$L = \sum_i L_i(\hat{y}_t, y_t)$$

Forward Pass:

$$h_t, \hat{y}_t, L_t, L$$

Backward Pass:

$$\frac{dL}{dU}, \frac{dL}{dV}, \frac{dL}{dW}, \frac{dL}{db_h}, \frac{dL}{db_y}$$

9/10/25

LAB-9

BUILD RECURRENT NEURAL NETWORK

AIM:

Build and train a simple RNN using PyTorch.

OBJECTIVES:

1. To understand how RNN works.
2. Train an RNN on sequential data.
3. Plot training loss and accuracy.
4. Evaluate model performance on test sequences.

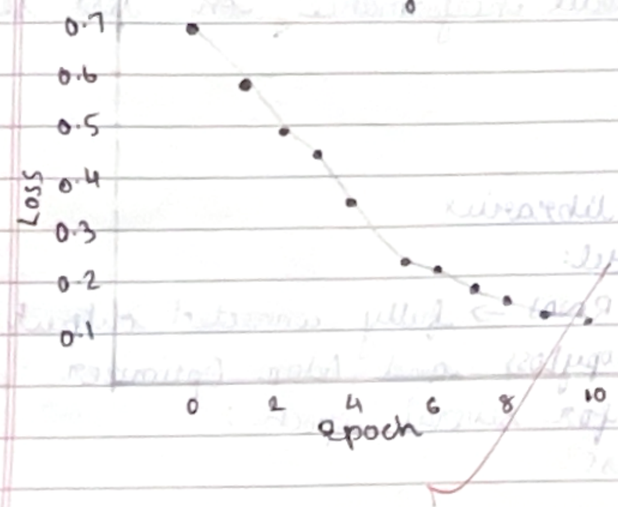
PSEUDOCODE:

1. Import libraries
2. Define RNN model:
   Embedding → RNN → fully connected output
3. Define CrossEntropyLoss and Adam Optimizer
4. Train model for several epochs:
   - Forward pass
   - Compute loss
   - Back propogation
   - Update weights
   - Track accuracy and loss
5. Plot training loss and accuracy.
6. Evaluate on test set.

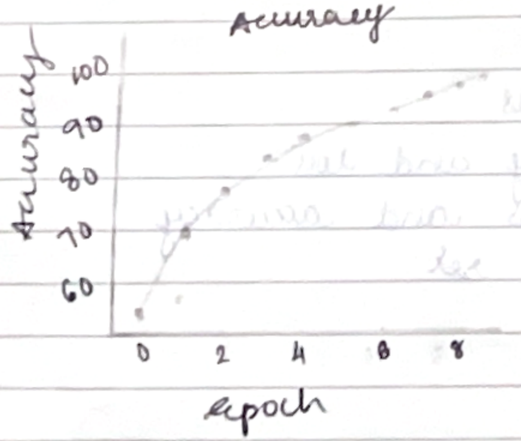
OUTPUT:

Epoch 1/10 - Loss: 0.6837 Accuracy: 54.38%
Epoch 2/10 - Loss: 0.5913 Accuracy: 70.00%
Epoch 3/10 - Loss: 0.4544 Accuracy: 78.75%
Epoch 4/10 - Loss: 0.3515 Accuracy: 84.62%
Epoch 5/10 - Loss: 0.2822 Accuracy: 88.12%
Epoch 6/10 - Loss: 0.2370 Accuracy: 90.88%
Epoch 7/10 - Loss: 0.1907 Accuracy: 92.62%
Epoch 8/10 - Loss: 0.1671 Accuracy: 94.50%
Epoch 9/10 - Loss: 0.1383 Accuracy: 95.38%
Epoch 10/10 - Loss: 0.1125 Accuracy: 96.38%

Training Loss



Accuracy



Test Accuracy : 87.50%.

OBSERVATION:

- Model trains quickly
- Accuracy improves steadily over epochs
- RNN captures simple sequential patterns.

Result:

-> RNN successfully classifies sequences
-> Achieves high training and test accuracy
-> Shows how sequential dependencies are learned by RNN.