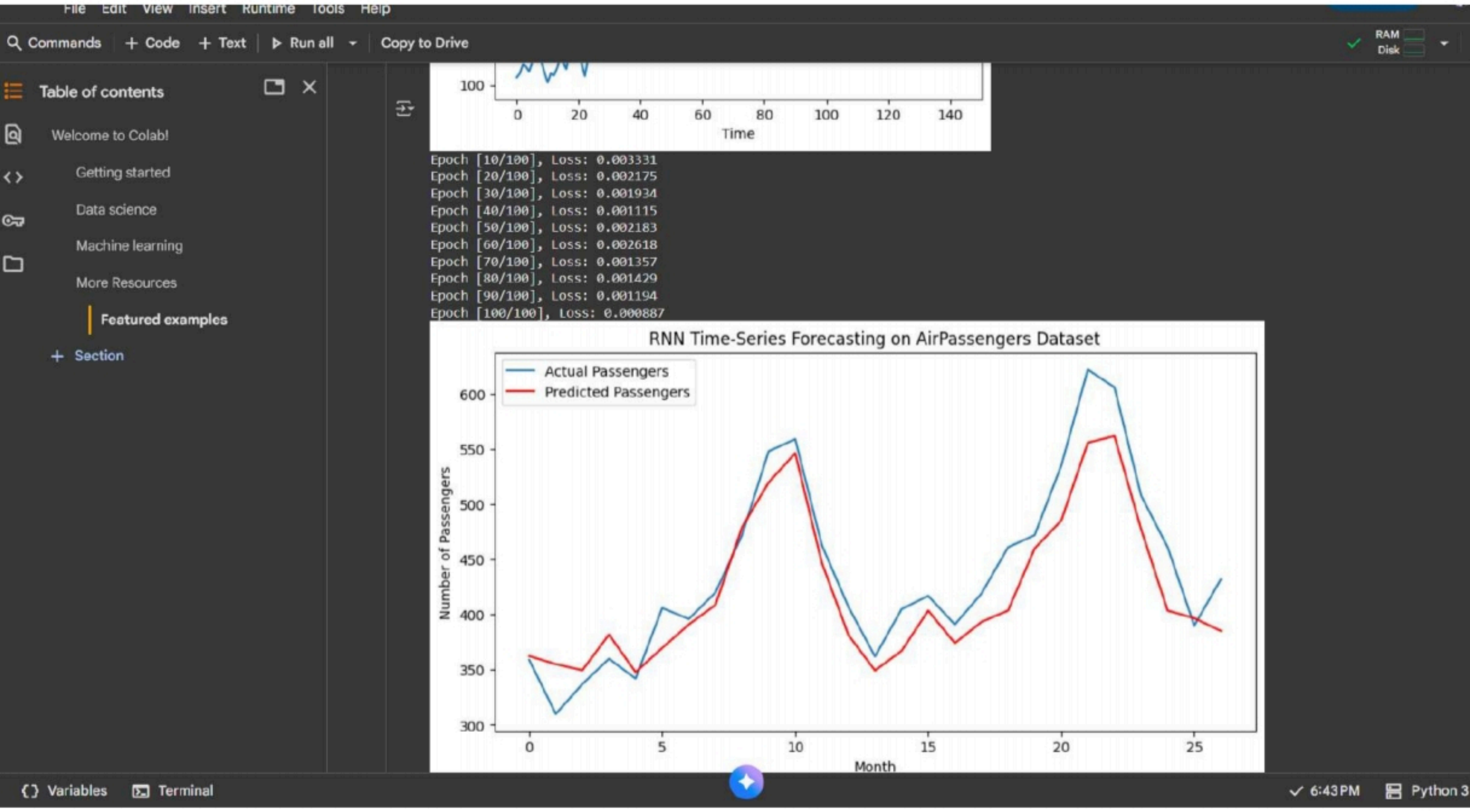
```python
import torch
import torch.nn as nn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing import MinMaxScaler
from torch.utils.data import DataLoader, TensorDataset

# ------------------------------------
# 1. Load Dataset
# ------------------------------------
url = "https://raw.githubusercontent.com/jbrownlee/Datasets/master/airline-passengers.csv"
df = pd.read_csv(url, usecols=[1])
plt.plot(df.values)
plt.title("Monthly Air Passengers")
plt.xlabel("Time")
plt.ylabel("Passengers")
plt.show()

# Normalize data to [0, 1]
scaler = MinMaxScaler()
data = scaler.fit_transform(df.values.astype(float))

# ------------------------------------
# 2. Prepare Sequences
# ------------------------------------
seq_length = 12  # use past 12 months to predict next month
X, y = [], []
for i in range(len(data) - seq_length):
    X.append(data[i:i + seq_length])
    y.append(data[i + seq_length])

X = np.array(X)
y = np.array(y)

X = torch.tensor(X, dtype=torch.float32)
y = torch.tensor(y, dtype=torch.float32)
```


File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Copy to Drive
RAM
Disk
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More Resources
Featured examples
+ Section
Time
Epoch [10/100], Loss: 0.003331
Epoch [20/100], Loss: 0.002175
Epoch [30/100], Loss: 0.001934
Epoch [40/100], Loss: 0.001115
Epoch [50/100], Loss: 0.002183
Epoch [60/100], Loss: 0.002618
Epoch [70/100], Loss: 0.001357
Epoch [80/100], Loss: 0.001429
Epoch [90/100], Loss: 0.001194
Epoch [100/100], Loss: 0.000887
RNN Time-Series Forecasting on AirPassengers Dataset
Actual Passengers
Predicted Passengers
Number of Passengers
Month
Variables
Terminal
6:43 PM
Python 3

Commands | + Code | + Text | Run all | Copy to Drive

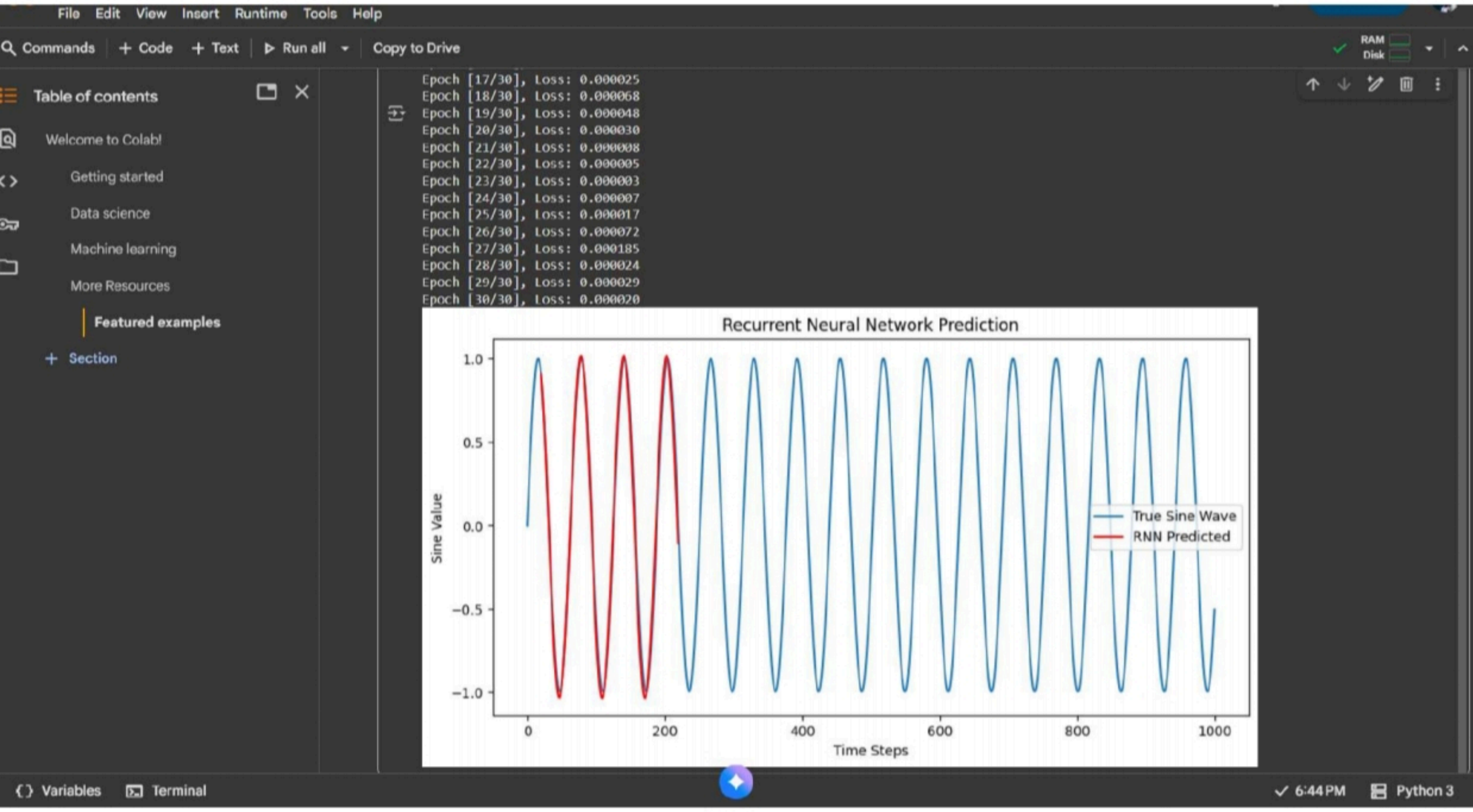
Table of contents

+ Section

[2]

```python
# ==============================

import torch
import torch.nn as nn
import numpy as np
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader, TensorDataset

# ------------------------------------
# 1. Create Sequential Dataset (Sine Wave)
# ------------------------------------
x = np.linspace(0, 100, 1000)
data = np.sin(x)

# Prepare input sequences and labels
seq_length = 20
X, y = [], []

for i in range(len(data) - seq_length):
    X.append(data[i:i + seq_length])
    y.append(data[i + seq_length])

X = np.array(X)
y = np.array(y)

X = torch.tensor(X, dtype=torch.float32).unsqueeze(-1)  # (samples, seq_len, 1)
y = torch.tensor(y, dtype=torch.float32).unsqueeze(-1)  # (samples, 1)

dataset = TensorDataset(X, y)
train_loader = DataLoader(dataset, batch_size=32, shuffle=True)

# ------------------------------------
# 2. Define the RNN Model
# ------------------------------------
class RNNModel(nn.Module):
    def __init__(self, input_size=1, hidden_size=50, num_layers=1, output_size=1):
        super(RNNModel, self).__init__()
        self.hidden_size = hidden_size
```

Variables | Terminal | 6:44 PM | Python 3


File Edit View Insert Runtime Tools Help
Commands
+ Code
+ Text
Run all
Copy to Drive
RAM
Disk
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More Resources
Featured examples
+ Section
Epoch [17/30], Loss: 0.000025
Epoch [18/30], Loss: 0.000068
Epoch [19/30], Loss: 0.000048
Epoch [20/30], Loss: 0.000030
Epoch [21/30], Loss: 0.000008
Epoch [22/30], Loss: 0.000005
Epoch [23/30], Loss: 0.000003
Epoch [24/30], Loss: 0.000007
Epoch [25/30], Loss: 0.000017
Epoch [26/30], Loss: 0.000072
Epoch [27/30], Loss: 0.000185
Epoch [28/30], Loss: 0.000024
Epoch [29/30], Loss: 0.000029
Epoch [30/30], Loss: 0.000020
Recurrent Neural Network Prediction
True Sine Wave
RNN Predicted
Sine Value
Time Steps
Variables
Terminal
6:44PM
Python 3

NAME: M. Rigan [RA2311047010022].

STD: A1 DIV: A ROLL NO.:

SUBJECT: DEEP LEARNING TECHNIQUES.

INDEX

SR. NO.	DATE	TITLE	PAGE NO.	TEACHER'S SIGN
1.	24/7	Exploring Deep Learning platform		
2.	31/7	Implement a classifier Using Open source dataset		
3.	14/8	Study of the Classifiers with respect to statistical parameter		22/8/25
4.	14/8	Build a simple feed forward neural network to registeru		
5	28/8/	Study of activation function.		
6	9/9/25	implement Gradient descent of Backpropagation In Deep Learning network		
7	16/9/25	cnn model to classify dog and cat images.		
8.	9/10/25	Experiment using LSTM		9/10/25
9.	9/10/25	Build a Recurrent Neural model		
10		Perform compression on MNIST dataset Using autoencoder		
11		Variational Autoencoder		

9. Build a Recurrent Neural Network

Aim:

To build and train a Recurrent neural Network (RNN) for sequence modelling

Objective

→ To understand the working principles of RNN

→ To preprocess sequential Data for RNN

→ To design and implement an RNN using pytorch

→ To train the RNN model and evaluate the performance

→ To analyse the output.

Pseudocode.

1. Start
2. Import necessary libraries
3. Load dataset
4. Preprocess dataset
   → clean data
   → tokenize / create sequences
   → pad / truncate sequences
   → split into training and validation.

Epochs

Epoche 1/5 Trainloss : 0.6432 Test loss : 0.6521

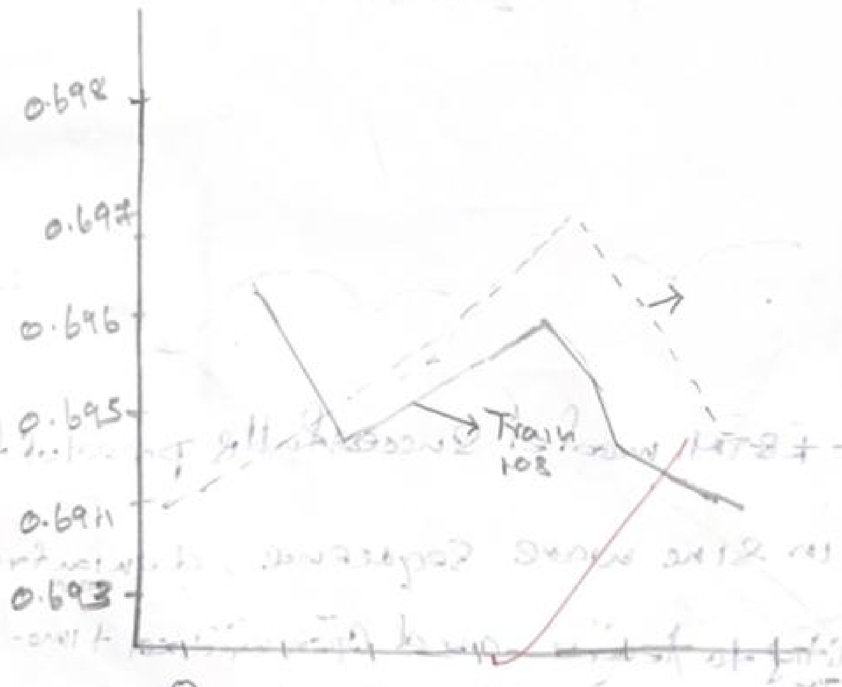
Epoche 2/5 Trainloss : 0.6435 Test loss : 0.6831

Epoche 3/5 Trainloss : 0.6021 Test loss : 0.6998

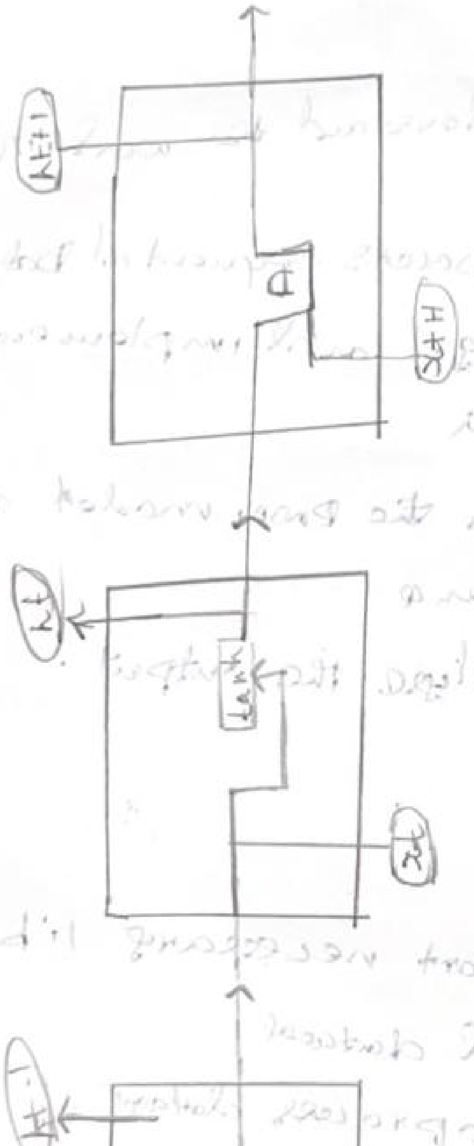
Epoche 4/5 Trainloss : 0.6632 Test loss : 0.6912

Epoche 5/5 Trainloss : 0.6015 Test loss : 0.6775

Test / Train Loss Graph:



RNN Architecture



→ Specify optimizer
→ Specify loss function
→ Specify Evaluation no

7. Evaluation node
→ Test on unseen data
→ Print accuracy loss metric

8. END

Observation

	Precision	recall	f1-Score	Support
0.0	0.71	0.70	0.70	4961
1.0	0.71	0.72	0.71	5039
accuracy			0.71	10000
Macro Avg	0.71	0.71	0.71	10000
weight Avg	0.71	0.71	0.71	1000

Result:

~~Save successfully~~
Therefore a Recurrent Neural Network has build successfully.

8. Experiment using LSTM

Aim:

To implement and analyze a long short-term memory (LSTM) neural network for predicting future values in time series dataset

Objectives:

1. To understand the architecture and working of an LSTM network

2. To prepare sequental data a suitable for LSTM input.

3. To train and evaluate the model on time-series data

4. To train and evaluate the model on time series data

5. To visualize model prediction versus actual target values

Psuedocode

1. Import required libraries

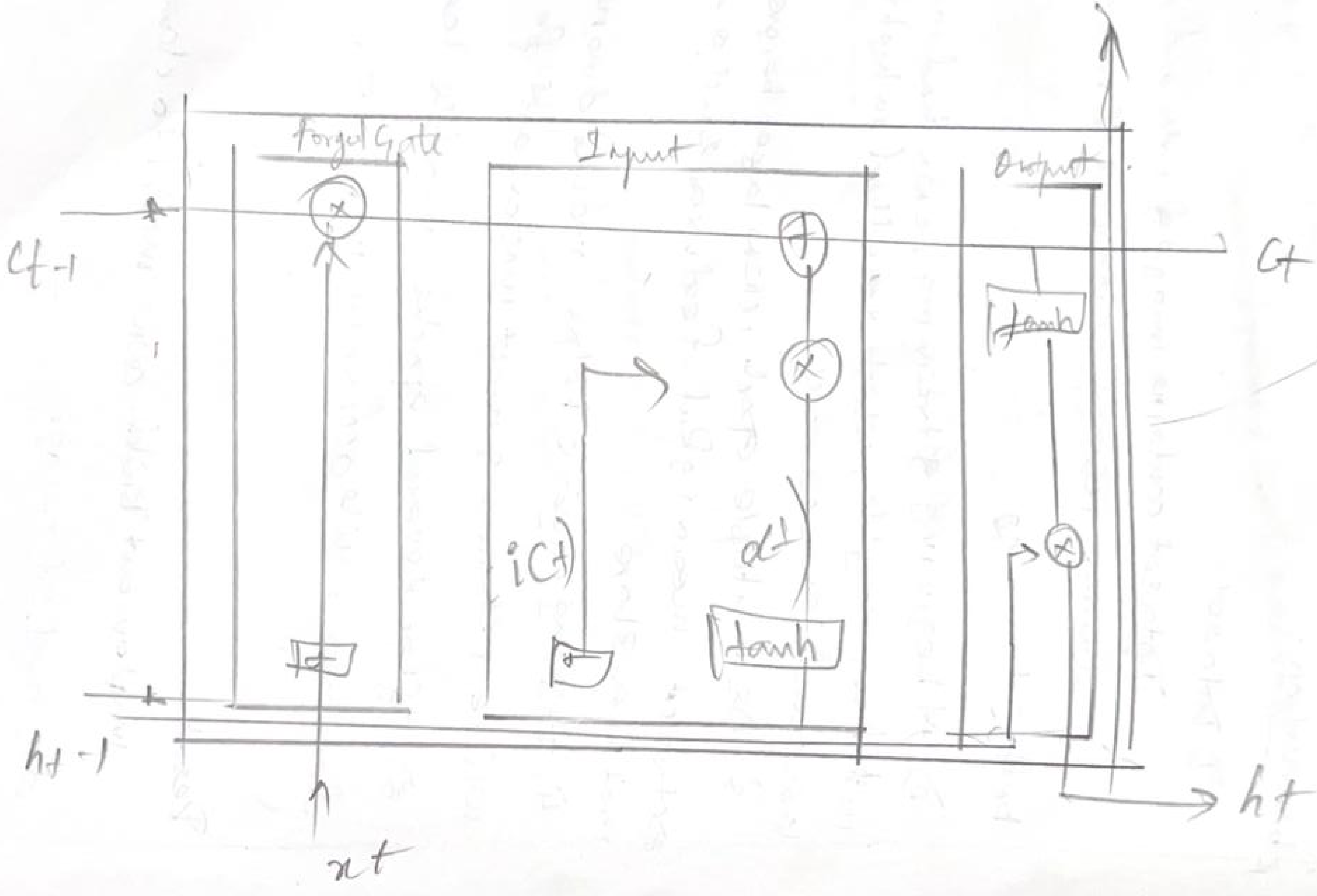
2. Generate or load a sequential dataset (eg. sine wave)

3. Normalize and prepare input output pairs for training

4. Define LSTM Model

- Input layer,

LSTM Architecture



- LSTM Layer
- fully connected output layer

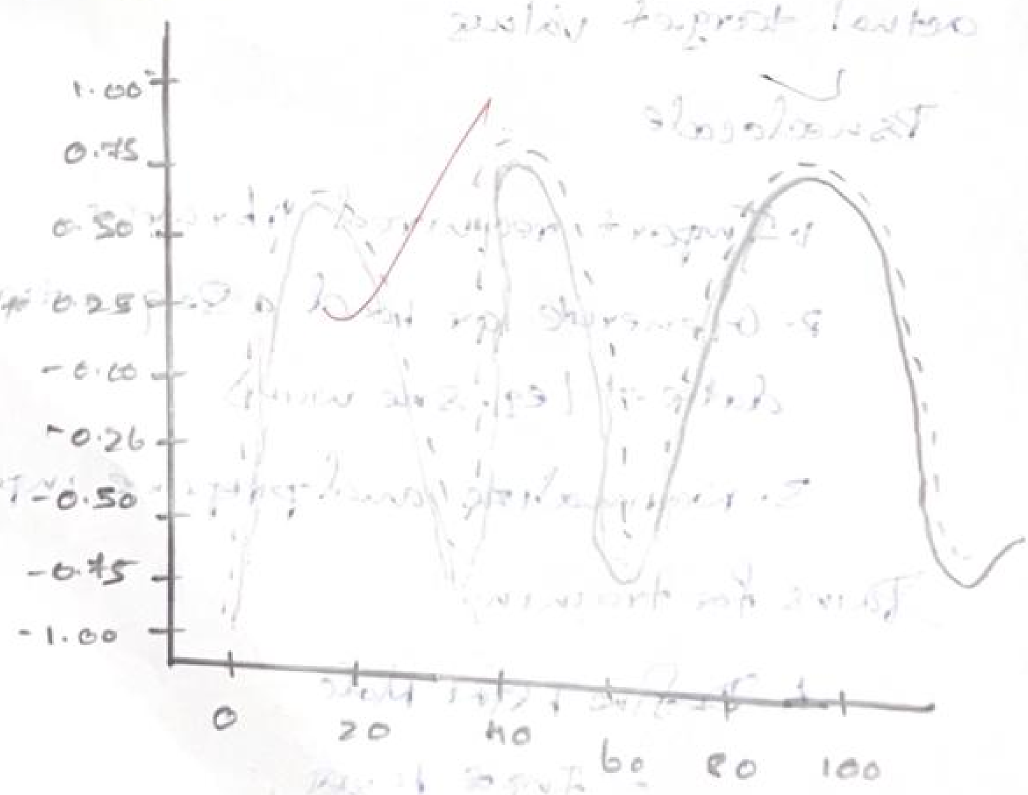
5. Define loss function and optimizer

Observation

* The LSTM learn temporal pattern from sequential data
* Loss decreases gradually as training proceeds
* predicted sine wave closely follows the actual curve after sufficient training

~~Result~~

~~The LSTM~~



Result (epoch, loss) ?

Result

The LSTM model successfully predict future value in sine wave sequence, demonstrating it ability to learn and generalize time-based dependence.