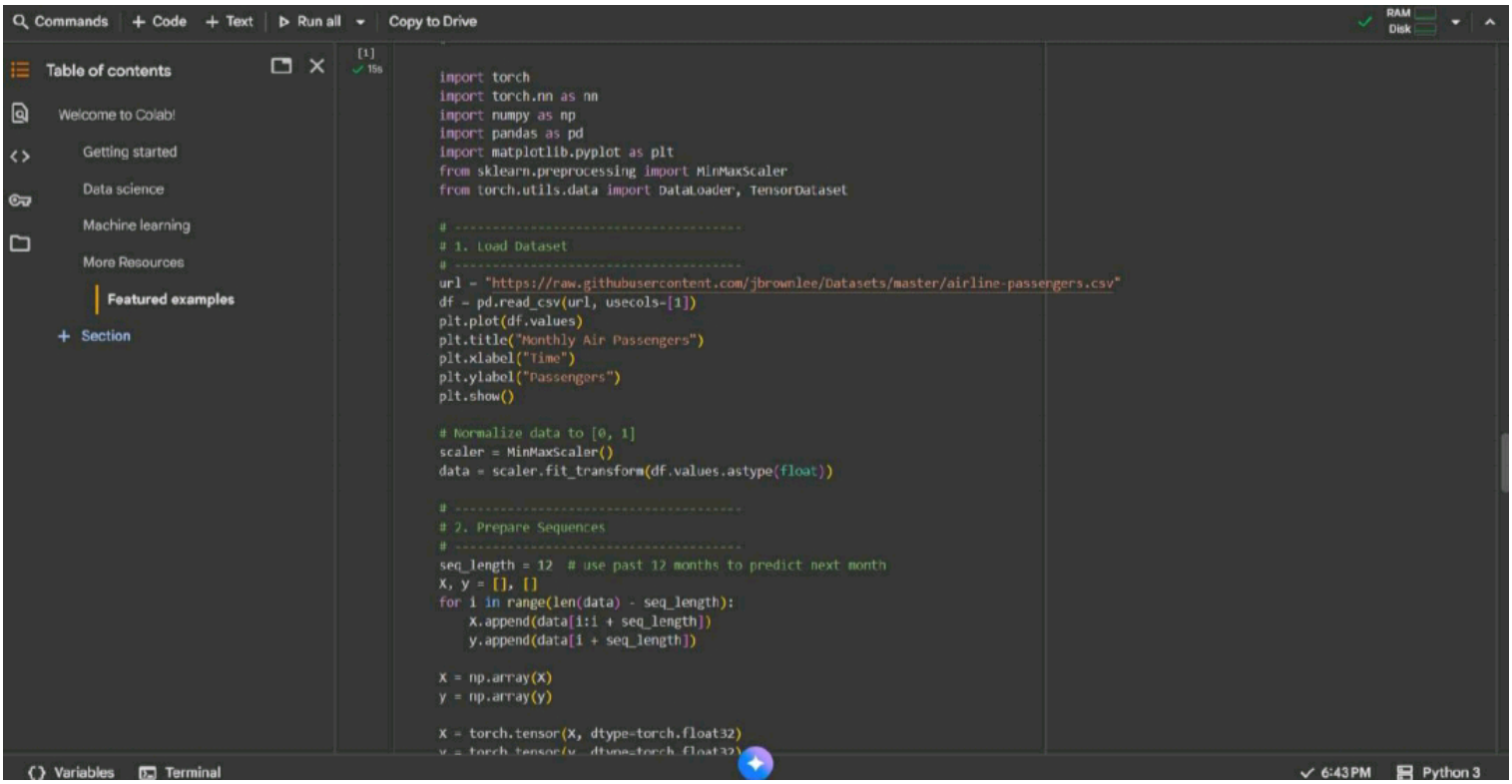
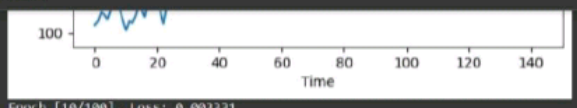

Commands
+ Code
+ Text
Run all
Copy to Drive
RAM
Disk
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More Resources
Featured examples
+ Section
[1]
15s
import torch
import torch.nn as nn
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.preprocessing 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)
Variables
Terminal
6:43 PM
Python 3

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

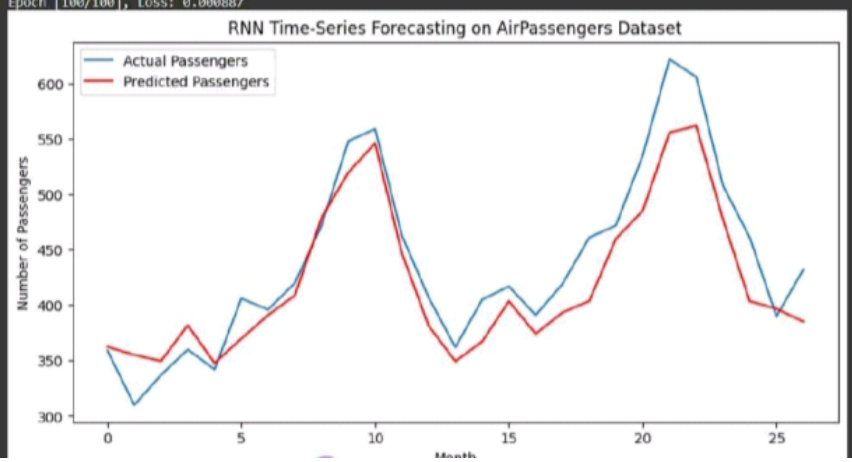
RAM
Disk

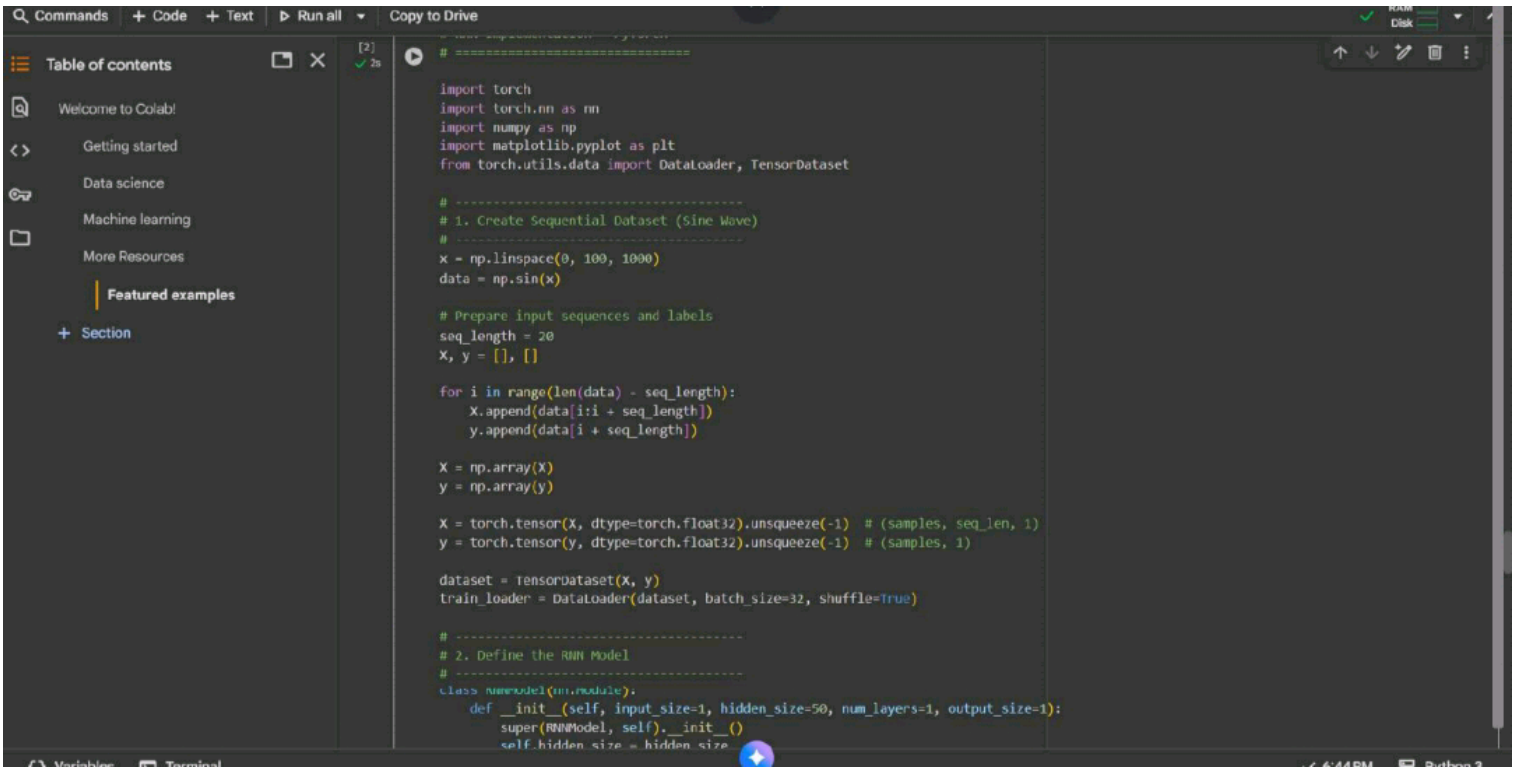
Table of contents

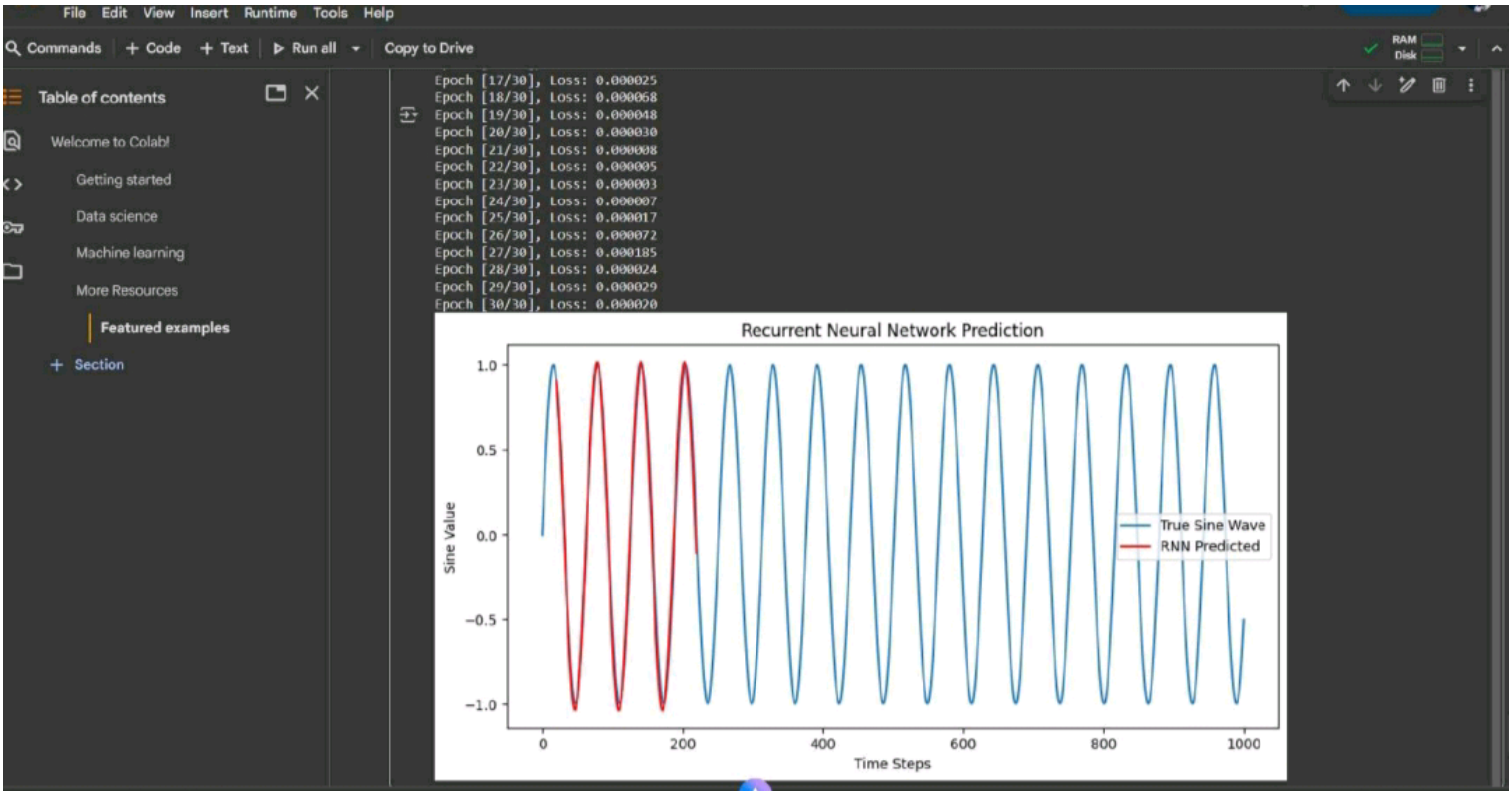
- Welcome to Colab!
- Getting started
- Data science
- Machine learning
- More Resources
  - Featured examples
- + Section




```
Epoch [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
```




Commands
+ Code
+ Text
Run all
Copy to Drive
Table of contents
Welcome to Colab!
Getting started
Data science
Machine learning
More Resources
Featured examples
+ Section
[2]
2s
# ==============================
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
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
Sine Value
Time Steps
True Sine Wave
RNN Predicted
1.0
0.5
0.0
−0.5
−1.0
0
200
400
600
800
1000

Ex:8

using LSTM algorithm

Aim:

To implement the LSTM algorithm

Algorithm:

* LSTM is a type of recurrent neural network
* capable of learning long term dependencies, especially in sequential data
* It uses gets to control the flow of information

Pseudo code:

1) Load the dataset
2) Preprocess the dataset
   a) Normalize the value
   b) convert the data into sequences
3) split the dataset into training and testing sets
4) Define the LSTM model

output:

Epoch 1/10 loss: 0.3440 - val_var = 0.0025

Epoch 2/10 loss: 0.0120 - val_var = 0.032

Epoch 3/10 loss: 6.0014 - val_var = 3.1339e

Epoch 4/10 loss: 2.239e-04 - val_var 1.944e-03

Epoch 5/10 loss: 2.0197e-05 - val_loss = 7.4402e-06

⋮

Epoch 10/10 loss: 1.2401e-07, val_loss: 1.258e-0

a) Input layer

b) LSTM layer

c) Dense layer

5) compile the model

a) Loss function, mean squared error

b) optimizer

6) Train the model on training data

7) Predict on test data

8) Inverse transform the predicted values

9) Evaluate the model

a) Plot actual vs Predicted values

b) calculate RMSE or MAE

Observation:

1) Initial Performance:

when the model was trained with a small no of epochs, the LSTM show basic learning capability but had relatively higher error values.

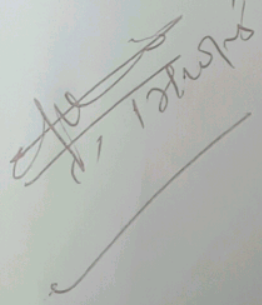
2) Improved accuracy with more training

increasing the no. of epochs to 100 and 200 significantly improved the model's performance.

3) Model stability :-

Overall, the LSTM model was stable and effective for the Prediction.

Result :



Result :

The LSTM model successfully learned Patterns in time series.

Ex:09 Build a Recurrent neural network

Aim:

To build and implement a Recurrent neural network using LSTM on time series.

Algorithm:

* LSTM is a type of RNN capable of learning long term dependencies.

* It is effective in handling time series and sequential data by maintaining memory over long sequence.

Pseudo code:

import necessary libraries or generate time series data

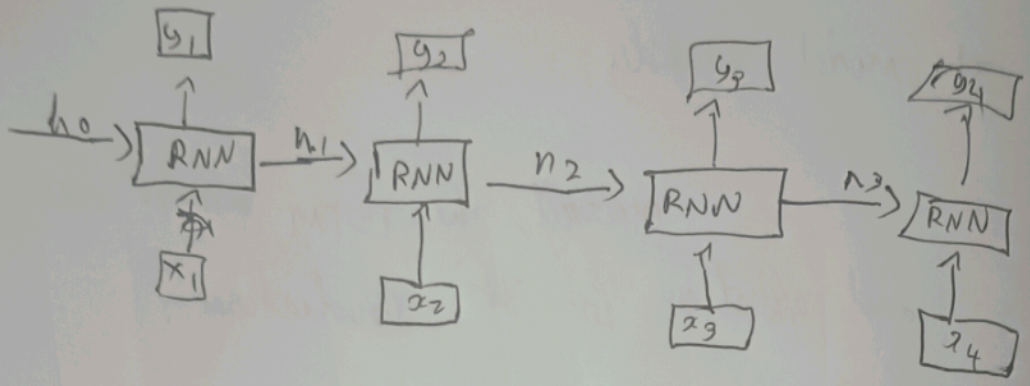
a) Normalize the data using minmax scaler

b) convert data into input output sequences using sliding windows.

c) Prepare data for RNN

d) Reshape input data into [sample, time, steps, features]

e) build the RNN model



output:

epoch 1/20 loss: 0.2938

epoch 2/20 loss: 0.0180

epoch 3/20 loss: 8.0956e-03

epoch 4/20 loss: 7.4476e-04

epoch 7/20 loss: 1.0246e-04

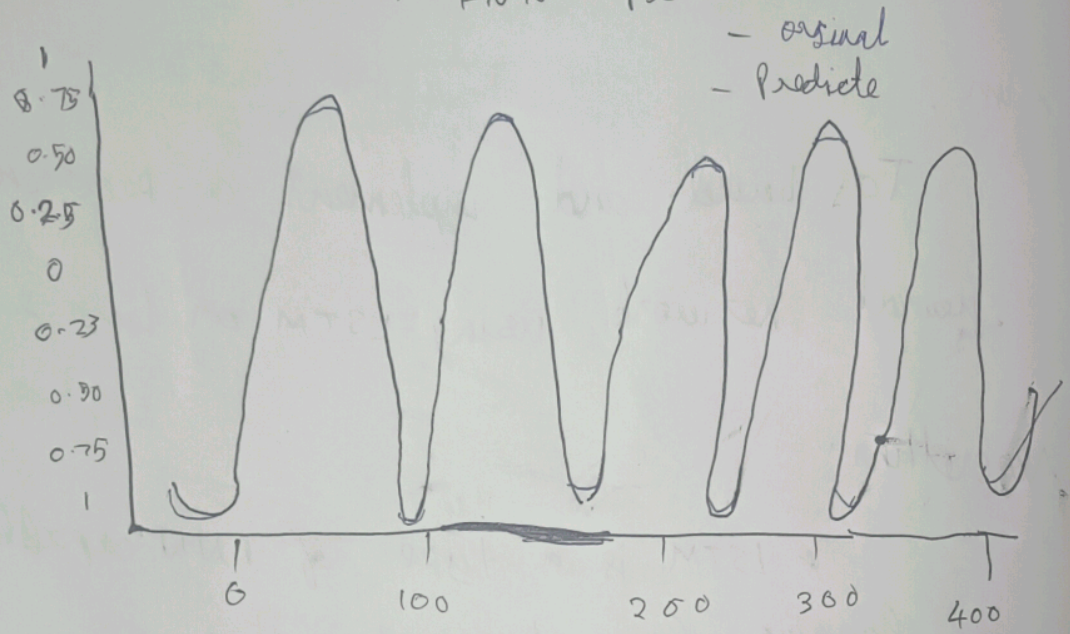
.

.

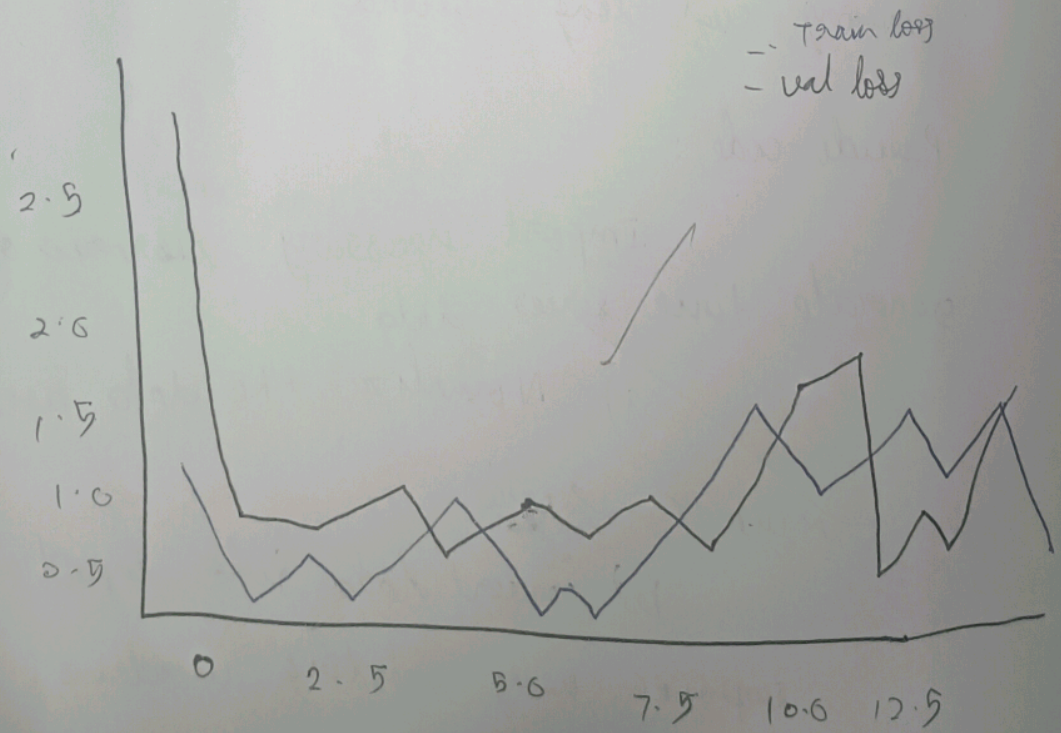
.

epoch 20/20 loss: 8.0972e-04

original sine vs RNN predies



Train and val loss :



6)

observation:

1) The LSTM model learned the sequence pattern the time data effectively after several epochs on data.

2) During initial epochs, the model showed higher error due to limited learning

3) overfitting was not observed in this case due to a simple dataset.

4) visualization of predicted vs actual value showed that the model was making accurate.

Result:-

The RNN model using LSTM successfully.