RNN\_Pytorch-2 (Score: 100.0 / 100.0)

1. [Test cell](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#cell-a4afe162f389b429) (Score: 50.0 / 50.0)
2. [Test cell](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#cell-6db64059f333988b) (Score: 50.0 / 50.0)

Predicting Google Stock Prices Using RNN in PyTorch[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg" \l "Predicting-Google-Stock-Prices-Using-RNN-in-PyTorch)

In this exercise, you will build and train a RNN network to predict Google stock prices based on historical data. You will process the data, design an RNN model, and evaluate its performance by comparing the predicted stock prices with the actual prices.

To build an RNN model for Google stock price prediction using PyTorch, follow these steps: 1. Data Preparation 1.Import Necessary libraries 2.Load the dataset from the path. 3.Preprocess the data: 1.Normalize the data for better model performance. 2.Split into training and testing datasets. 3.Prepare the sequences for the RNN. 2. Define the RNN Model 1.Use nn.RNN or nn.LSTM/nn.GRU from PyTorch. 2.Include input size, hidden size, number of layers, and output size. 3.Add a fully connected layer to predict stock prices. 3. Training the Model 1.Define a loss function, e.g., Mean Squared Error (MSE). 2.Use an optimizer like Adam. 3.Train the model over multiple epochs. 4. Evaluate the Model 1.Test the model on unseen data. 2.Plot predictions vs. actual values to visualize performance.

1. Data Preparation[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#1.-Data-Preparation)

1.Import Necessary libraries[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#1.Import-Necessary-libraries)

In [ ]:

**import** **numpy** **as** **np**

**import** **pandas** **as** **pd**

**import** **torch**

**import** **torch.nn** **as** **nn**

**from** **sklearn.preprocessing** **import** MinMaxScaler

**from** **sklearn.model\_selection** **import** train\_test\_split

**import** **matplotlib.pyplot** **as** **plt**

2. Load dataset[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#2.--Load-dataset)

Load the Dataset: Use the pandas library to load the dataset Google\_Stock\_Price.csv. The dataset contains columns like "Date," "Open," "High," "Low," and "Close." For this task, you will only use the 'Open' column. Key Steps: Extract the "Open" column as the feature for prediction and store it in "prices". Convert the extracted data into a NumPy array and reshape it into a column vector.

In [ ]:

*# Load dataset*

data = pd.read\_csv('/srv/shareddata/datasets/ps2/Google\_Stock\_Price/Google\_Stock\_Price.csv') *# Replace with your file path*

prices = data['Open'].values.reshape(-1, 1)

In [ ]:

data.shape

In [ ]:

prices.shape

3. Data normalization[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#3.-Data-normalization)

Steps to Normalize the Data:[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#Steps-to-Normalize-the-Data:)

1.Import the Required Library: Import the MinMaxScaler class from the sklearn.preprocessing module.

2.Initialize the Scaler(**scaler**): Create an instance of MinMaxScaler and specify the desired range for scaling (default is [0, 1]).

3.Fit and Transform: Use the fit\_transform method to compute the minimum and maximum values and transform the stock prices into the normalized range and save it in \*\*prices\_scaled variable \*\*.

In [ ]:

*# Data normalization*

scaler = MinMaxScaler(feature\_range=(0, 1))

prices\_scaled = scaler.fit\_transform(prices)

4.Prepare sequences[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#4.Prepare-sequences)

In time series prediction, the goal is to predict the next value based on previous values. To do this, we use sliding window sequences: Each sequence contains a fixed number of previous timesteps (e.g., 60 timesteps). The target value is the next timestep following the sequence. Steps to Prepare Sequences: 1.Choose the Sequence Length: Decide the number of previous timesteps to use for prediction. For this task, use sequence\_length = 60. 2.Initialize Input (X) and Output (y): 3.Create two empty lists: X: For storing sequences of 60 timesteps. y: For storing the target values corresponding to each sequence. 4.Generate Sequences: 1.Iterate through the scaled price data starting from sequence\_length to the end. For each position i, extract: A sequence of 60 previous timesteps (prices\_scaled[i-sequence\_length:i, 0]) and append it to X. The next value (prices\_scaled[i, 0]) as the target and append it to y. 4.Convert to NumPy Arrays: Once all sequences and targets are collected, convert X and y to NumPy arrays for further processing.

In [ ]:

*# Prepare sequences*

sequence\_length = 60 *# Use 60 previous timesteps to predict the next*

X, y = [], []

**for** i **in** range(sequence\_length, len(prices\_scaled)):

X.append(prices\_scaled[i-sequence\_length:i, 0])

y.append(prices\_scaled[i, 0])

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

X.shape,y.shape

5. Train-test split[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#5.-Train-test-split)

Purpose of Train-Test Split: Training Set: Used to train the LSTM model. Testing Set: Used to evaluate the model's performance on unseen data.A typical split is 80% training and 20% testing.

Split the input (X) and output (y) into X\_train and y\_train: For training X\_test and y\_test: For testing and Convert X\_train, X\_test, y\_train, and y\_test to tensors using torch.tensor().

In [ ]:

*# Train-test split*

X\_train, X\_test, y\_train, y\_test = train\_test\_split(X, y, test\_size=0.2, random\_state=42)

X\_train, X\_test = torch.tensor(X\_train, dtype=torch.float32), torch.tensor(X\_test, dtype=torch.float32)

y\_train, y\_test = torch.tensor(y\_train, dtype=torch.float32), torch.tensor(y\_test, dtype=torch.float32)

In [ ]:

X\_train.shape

6.Reshape for RNN input[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#6.Reshape-for-RNN-input)

Purpose of Reshaping: RNNs, including LSTMs, require input data to be in a specific 3D tensor format:(n\_samples,sequence\_length,n\_features) n\_samples: Number of sequences in the dataset. sequence\_length: Length of each sequence (e.g., 60 timesteps). n\_features: Number of features per timestep (e.g., 1 for stock prices). The current shape of X\_train and X\_test is 2D:(n\_samples,sequence\_length) We need to add a third dimension for n\_features to indicate that each timestep has only one feature. Steps to Reshape Data: Use PyTorch's .view() method to reshape the tensors. Maintain the batch size (n\_samples) and sequence length while adding an additional dimension for features.

In [ ]:

*# Reshape for RNN input*

X\_train = X\_train.view(X\_train.shape[0], X\_train.shape[1], 1)

X\_test = X\_test.view(X\_test.shape[0], X\_test.shape[1], 1)

In [ ]:

X\_train.shape,y\_train.shape

2. RNN Model Building[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#2.-RNN-Model-Building)

1.Defining the RNN Model[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#1.Defining-the-RNN-Model)

1.Use nn.RNN or nn.LSTM/nn.GRU from PyTorch. 2.Include input size, hidden size, number of layers, and output size. 3.Add a fully connected layer to predict stock prices.1.Structure of the RNN Model: Input Layer: Accepts input sequences of shape (n\_samples,sequence\_length,n\_features), where: n\_samples: Number of sequences. sequence\_length: Length of each sequence (e.g., 60 timesteps). n\_features: Number of features per timestep (e.g., 1 for stock prices). Hidden Layer: Uses an RNN module to process the input sequences. Parameters: input\_size: Number of features per timestep. hidden\_size: Number of neurons in the hidden layer. num\_layers: Number of stacked RNN layers. batch\_first=True: Ensures the input has batch size as the first dimension. Output Layer: A fully connected (dense) layer maps the RNN's hidden state to the desired output size. 2.Steps to Implement the RNN Model: Define the Class: Create a class (StockPriceRNN )inheriting from torch.nn.Module. Initialize the Layers: Use nn.RNN for the recurrent layer. Use nn.Linear for the fully connected layer to produce the output. Define the Forward Pass (forward): Pass the input through the RNN. Use only the last time step's output(save it in out variable) for prediction.

In [ ]:

*# Define RNN model*

**class** **StockPriceRNN**(nn.Module):

**def** \_\_init\_\_(self, input\_size=1, hidden\_size=50, num\_layers=2, output\_size=1):

super(StockPriceRNN, self).\_\_init\_\_()

self.rnn = nn.RNN(input\_size, hidden\_size, num\_layers, batch\_first=**True**)

self.fc = nn.Linear(hidden\_size, output\_size)

**def** forward(self, x):

out, \_ = self.rnn(x)

print(out.shape)

print(out[:, -1, :].shape)

out = self.fc(out[:, -1, :]) *# Use the last time step's output*

print("final out",out.shape)

**return** out

2.Initializing the Model, Loss Function, and Optimizer[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#2.Initializing-the-Model,-Loss-Function,-and-Optimizer)

You are tasked with building an RNN model to predict Google stock prices. Implement the following steps: Define an RNN model (StockPriceRNN) that takes in the past 60 days' stock prices to predict the next day's price. Set up the loss function using Mean Squared Error (MSE) for the regression task. Initialize the Adam optimizer with a learning rate of 0.001 to optimize the model parameters. Use the defined model(model), loss function(criterion), and optimizer(optimizer) to train the model on the stock price dataset.

In [ ]:

Student's answer[(Top)](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#top)

*# YOUR CODE HERE*

model = StockPriceRNN()

In [ ]:

model

3.Training the model[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#3.Training-the-model)

You need to implement the training loop for the StockPriceRNN model to predict Google stock prices. The model should be trained as follows: Training Loop: Train the model for 50 epochs. For each epoch: Set the model to training model using model.train(). Pass the input data (X\_train) through the model to obtain predictions. Compute the loss using the Mean Squared Error (MSE) between the predicted output and the actual stock price (y\_train). Perform backpropagation: Zero the gradients with optimizer.zero\_grad(). Compute the gradients via loss.backward(). Update the model parameters using optimizer.step(). Every 10 epochs, print the current epoch number and the loss value. Your task is to implement this training loop to optimize the model for stock price prediction.

In [ ]:

Student's answer[(Top)](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#top)

*# YOUR CODE HERE*

**import** **torch.optim** **as** **optim**

epochs = 50

optimizer = optim.Adam(model.parameters(), lr = 0.001)

criterian = nn.MSELoss()

**for** epoch **in** range(epochs):

model.train()

y\_pred = model(X\_train)

loss = criterian(y\_pred, y\_train)

optimizer.zero\_grad()

loss.backward()

optimizer.step()

**if** epoch % 10 == 0:

print(f"Epoch: **{**epoch+1**}** / 50, Loss: **{**loss.item()**}**:.4f")

Test Case 2: Forward Pass Output Shape[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#Test-Case-2:-Forward-Pass-Output-Shape)

In [ ]:

Grade cell: cell-a4afe162f389b429Score: 50.0 / 50.0 [(Top)](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#top)

Test Case 3: Test Optimizer Updates[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#Test-Case-3:-Test-Optimizer-Updates)

In [ ]:

Grade cell: cell-6db64059f333988bScore: 50.0 / 50.0 [(Top)](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#top)

3. Evaluate the Model[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#3.-Evaluate-the-Model)

You need to implement the testing phase of the StockPriceRNN model after training. The steps should be: Switch to Evaluation Mode: Set the model to evaluation mode using model.eval(). This ensures that layers like dropout or batch normalization (if used) behave correctly during inference. Generate Predictions: Use the trained model to predict the stock prices for the test dataset (X\_test) and store it in predictions variable. Use torch.no\_grad() to avoid tracking gradients during testing (this saves memory and computation). Your task is to complete the testing phase by generating predictions(save it in predictions) and preparing them for further analysis.

In [ ]:

Student's answer[(Top)](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#top)

*# YOUR CODE HERE*

model.eval()

**with** torch.no\_grad():

predictions = model(X\_test)

In [ ]:

4. Plot results[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#4.-Plot-results)

After testing the model, you need to rescale the predicted and actual stock prices back to the original scale and visualize the results. The steps are: Rescale the Predictions and Actual Values: convert the predictions and the actual values (y\_test) back from the scaled range [0, 1] to the original stock price values Plot the Results

In [ ]:

Student's answer[(Top)](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#top)

*# YOUR CODE HERE*

5.Predict on a new sample[¶](http://10.11.51.204:8000/user/23bd1a054t/files/ps2/RNN_Pytorch-2/feedback/2025-01-24%2005%3A53%3A13.029751%20UTC/RNN_Pytorch-2.html?_xsrf=MnwxOjB8MTA6MTczNzk1MTIxN3w1Ol94c3JmfDEzMjpPRFJsWXpobU9Ua3haRGxtTkRaa1pqbG1OamhoWkdJNFlqRXpNamRsT0RFNk56UTJZemhoTVRZM1pEbGpPR1ZpWlRVME9EZGtZamMwT1dZNU5qSmpOR0V3TkRObE56SmhNV1ZtWkRGbE1EQmlZems0TUdSak1qWTVZekUxTjJWbE1RPT18MDE0YjMxNjQ1NDMyNTAwZDgzNDdlODhhN2NiYzVmYWRiMjE5OWUyN2E4NWVjZjQwNWYxN2JkNDhkZDc2MjU1Yg#5.Predict-on-a-new-sample)

In [ ]:

*# Predict on a new sample*

sample\_sequence = prices[-60:] *# Use the last 60 days as an example*

sample\_sequence\_scaled = scaler.transform(sample\_sequence)

*# Convert to tensor and reshape for the model*

sample\_tensor = torch.tensor(sample\_sequence\_scaled, dtype=torch.float32).view(1, sequence\_length, 1)

*# Make prediction*

model.eval()

**with** torch.no\_grad():

predicted\_scaled = model(sample\_tensor).item()

print("model(sample\_tensor)",model(sample\_tensor))

print("predicted\_scaled",predicted\_scaled)

print("[[predicted\_scaled]]",[[predicted\_scaled]])

*# Inverse transform to get the actual predicted price*

predicted\_price = scaler.inverse\_transform([[predicted\_scaled]])[0][0]

predicted\_price

*# The actual value (next day's price)*

actual\_price = prices[-1] *# Replace this with the actual value if available*

print(f"Actual Stock Price: **{**actual\_price[0]**:**.2f**}**")

print(f"Predicted Stock Price: **{**predicted\_price**:**.2f**}**")

In [ ]: