Preliminaries

Set up taken from rnn_notebook demo from week 8

See here: https://colab.research.google.com/drive/1whCpnoUGoB1FhpaxQTfLDdAzB7yMvkgE

In [ ]:
```
import csv
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
import torchtext
import torchvision.datasets
import torchvision.transforms as transforms
import numpy as np
import matplotlib.pyplot as plt
import os
import pandas as pd
import pdb
import math
```

Mount Google Drive (taken from A2):

In [ ]:
```
from google.colab import drive
drive.mount('/content/gdrive', force_remount=True)
```

```
Mounted at /content/gdrive
```

We will work with the ETFs folder since it has less data, and we will seperate one file path in particular to have a closer look at it

In [ ]:
```
etf_path = "/content/gdrive/My Drive/CSC413FinalProject/data/unzipped/ETFs/"
stock_path = "/content/gdrive/My Drive/CSC413FinalProject/data/unzipped/Stocks/"

smalldata_path = "/content/gdrive/My Drive/CSC413FinalProject/data/unzipped/smallDataSet/"

somedata_path = "/content/gdrive/My Drive/CSC413FinalProject/data/unzipped/ETFs/aadr.us.txt"
```

In [ ]:
```
# file_paths: the google drive directory to retrieve the data from
# num_days: the number of days into the future we will make predictions for
# cols: Feed in array of form ["Close", ...] with the column headers for the data you want to extract

def init_data(file_paths, num_days, cols):
  data = []


  for path in file_paths:
    for filename in os.listdir(path):
        f = os.path.join(path, filename)
        if os.path.getsize(f) > 0:
          # Extract data from only 2016 onwards
          df = pd.read_csv(f, parse_dates = True)

          mask = df['Date'] >= "2016-01-01"

          data.append(
                      (
                       torch.tensor(df.loc[mask][cols].to_numpy()[:-num_days]),
                       torch.tensor(df.loc[mask][cols].to_numpy()[-num_days:])
                      )
                     )

  return data
```

In [ ]:
```
dataset = init_data([stock_path, etf_path], 1, ["Close"])
```

In [ ]:
```
print(len(dataset))

for i in range(len(dataset)):
  if i > 5:
    break
  print(len(dataset[i][0]))
```

```
8507
```

```
8507
469
469
374
469
469
45
```

In [ ]:
```
smalldataset = init_data([smalldata_path], 1, ["Close"])
total = 0
t = 0
for d in smalldataset:
  print(len(d[0]), len(d[1]))
  total += len(d[0])
  t+=1
print(t)
# print(smalldataset[0])
# print(smalldataset[1])
```

```
469 1
469 1
469 1
469 1
469 1
469 1
469 1
459 1
469 1
151 1
10
```

Divide data into train, validation, test

Shuffles the data array and makes a 60, 20, 20 split of the data into training, validation, and test set

In [ ]:
```
# Divides the data into training, validation, test, with 60%, 20%, 20% split

def divide_data(data, shuffle_seed=31415926):
  train = []
  valid = []
  test = []

  rng = np.random.RandomState(shuffle_seed)
  perm = rng.permutation(len(data))

  for i in range(len(data)):
    if i % 5 < 3:
      train.append(data[perm[i]])
    elif i % 5 == 3:
      valid.append(data[perm[i]])
    else:
      test.append(data[perm[i]])

  return train, valid, test
```

In [ ]:
```
train, valid, test = divide_data(dataset)
```

In [ ]:
```
print(len(train))
print(len(valid))
print(len(test))
```

```
5105
1701
1701
```

In [ ]:
```
for i in range(len(train)):
  if i > 5:
    break
  print(train[i])
  print(train[i][0].shape)
  print(train[i][1].shape)
```

```
(tensor([[25.8020]
```

```
(tensor([[35.8030],
         [35.8750],
         [35.4490],
         [34.8300],
         [34.1690],
         [34.5940],
         [34.6820],
         [33.5800],
         [33.6500],
         [32.9120],
         [32.9420],
         [32.8430],
         [33.3840],
         [33.0260],
         [33.3710],
         [33.2070],
         [33.3510],
         [33.9440],
         [34.3920],
         [33.8800],
         [33.5800],
         [33.1090],
         [32.0550],
         [31.5830],
         [31.6420],
         [31.9090],
         [31.6330],
         [32.3030],
         [32.9120],
         [33.4630],
         [33.0690],
         [33.1520],
         [33.4730],
         [33.7000],
         [33.7660],
         [34.3870],
         [34.8420],
         [34.9530],
         [34.7010],
         [35.1840],
         [34.9100],
         [35.0650],
         [35.0940],
         [35.0850],
         [35.5280],
         [35.7920],
         [35.6940],
         [35.8620],
         [36.1570],
         [36.5300],
         [36.4420],
         [36.3430],
         [36.1360],
         [35.7720],
         [35.9990],
         [36.1080],
         [36.7380],
         [36.8050],
         [36.8450],
         [36.4420],
         [36.0680],
         [36.3040],
         [35.6330],
         [35.5210],
         [35.3490],
         [35.3190],
         [35.9500],
         [35.6930],
         [35.7870],
         [36.0880],
         [35.9690],
         [36.0380],
         [35.9610],
         [35.8020],
         [35.4480],
         [35.7890],
         [35.8740],
         [35.4080],
         [35.1130],
         [35.3100],
         [35.2220],
         [35.2700],
         [34.7780],
         [35.2020],
```

[35.1130],
[34.4320],
[34.3270],
[33.8720],
[34.2970],
[33.2620],
[33.1450],
[33.5770],
[33.6960],
[33.4900],
[33.8920],
[33.8920],
[33.9760],
[34.1970],
[34.1570],
[34.3460],
[34.3460],
[34.4720],
[34.1490],
[34.3920],
[34.4140],
[34.0600],
[33.9010],
[33.5670],
[33.4410],
[33.8880],
[33.6570],
[33.7510],
[33.8480],
[33.9090],
[34.1440],
[33.6320],
[33.3350],
[33.9240],
[34.2540],
[34.5900],
[34.2940],
[34.7070],
[34.5100],
[35.1630],
[35.4390],
[35.4900],
[35.1730],
[35.3700],
[35.2800],
[35.4590],
[35.5810],
[35.2900],
[35.4000],
[35.7050],
[35.9320],
[35.7640],
[35.9540],
[35.3410],
[35.4490],
[35.5280],
[36.0410],
[35.7970],
[35.8240],
[36.2280],
[36.1780],
[36.5150],
[36.2580],
[35.9950],
[36.3470],
[36.5750],
[36.5050],
[36.9090],
[36.6830],
[36.1690],
[35.6350],
[35.6350],
[35.4610],
[35.6950],
[35.7250],
[35.4090],
[35.3750],
[34.5900],
[34.9260],
[34.4010],
[34.8270],
[34.7350],
[34.7690],

[34.5930],
[34.8050],
[35.0920],
[35.0120],
[34.7150],
[34.7550],
[34.3890],
[34.2510],
[34.5480],
[34.3840],
[34.2210],
[34.5130],
[34.5830],
[34.6960],
[34.8150],
[34.4580],
[34.8160],
[34.6180],
[34.4880],
[34.4480],
[34.3500],
[34.3750],
[34.4680],
[34.2110],
[34.2610],
[33.8340],
[34.0250],
[33.9840],
[33.7180],
[33.7670],
[33.1810],
[33.5000],
[33.7550],
[33.5410],
[34.0440],
[35.1310],
[35.2060],
[36.3480],
[36.3280],
[36.2630],
[36.5560],
[36.3380],
[37.0430],
[36.8730],
[37.0000],
[36.8730],
[36.7640],
[36.3950],
[36.5950],
[36.6250],
[36.9210],
[37.3570],
[37.9410],
[38.4340],
[38.1650],
[37.8120],
[38.0250],
[37.7080],
[37.8180],
[37.8160],
[37.5680],
[37.8660],
[37.5580],
[36.3590],
[36.2090],
[36.0100],
[36.0500],
[35.6730],
[35.6630],
[36.2790],
[35.5240],
[35.4980],
[35.7720],
[35.5240],
[35.4690],
[35.3860],
[35.6730],
[35.5040],
[35.1060],
[35.0580],
[34.6310],
[34.8350],
[35.0380],

[34.7100],
[34.1050],
[34.2240],
[34.6020],
[34.3040],
[34.3040],
[34.4820],
[34.2340],
[34.0850],
[34.4910],
[35.1580],
[35.3460],
[35.1290],
[35.3210],
[35.1380],
[35.3060],
[35.7130],
[35.6630],
[35.0180],
[35.6210],
[34.8800],
[34.9390],
[35.2190],
[34.7750],
[34.6020],
[34.2640],
[34.5490],
[34.2410],
[34.3030],
[34.2440],
[34.8450],
[35.0510],
[34.6550],
[33.9990],
[34.1040],
[34.2370],
[34.2260],
[34.2070],
[34.4250],
[34.8430],
[34.8030],
[34.8230],
[34.3470],
[34.0200],
[34.1570],
[34.1770],
[34.4010],
[34.5660],
[34.5100],
[34.3130],
[34.7530],
[35.2690],
[35.2890],
[35.5160],
[35.6270],
[36.0490],
[35.7070],
[35.6870],
[35.6570],
[35.5280],
[35.7370],
[35.9050],
[35.2130],
[35.1410],
[34.5250],
[34.5850],
[34.7530],
[34.9620],
[34.7330],
[34.2770],
[34.8430],
[34.9030],
[34.8330],
[34.7190],
[35.5480],
[35.5830],
[35.4090],
[34.9910],
[35.1110],
[35.3890],
[35.3790],
[35.4190],
[35.5380],

[35.3490],
[34.5850],
[33.8700],
[34.3490],
[33.8800],
[33.6210],
[33.7610],
[34.3990],
[34.2990],
[34.7090],
[34.3580],
[34.5980],
[34.4090],
[34.0800],
[33.9890],
[33.4120],
[33.2730],
[33.5420],
[33.7010],
[33.8690],
[33.9400],
[34.1000],
[34.2790],
[34.1300],
[33.8700],
[34.0740],
[34.3990],
[34.3890],
[34.4880],
[34.7870],
[34.8370],
[34.8920],
[34.8360],
[34.5980],
[34.9620],
[35.0160],
[34.0800],
[33.7310],
[33.8750],
[34.0500],
[33.5420],
[33.6510],
[33.0130],
[32.8690],
[32.9540],
[32.8440],
[32.2760],
[32.3460],
[31.8890],
[32.0570],
[32.1960],
[32.6760],
[33.0080],
[33.4520],
[33.3850],
[33.3620],
[33.8580],
[34.1200],
[33.9900],
[33.9165],
[33.5800],
[33.4400],
[33.5396],
[33.7060],
[33.7874],
[34.2000],
[33.9811],
[33.9738],
[34.1350],
[34.1550],
[34.2500],
[34.1400],
[33.3600],
[33.4610],
[33.4850],
[33.4400],
[33.6156],
[33.3500],
[33.3460],
[33.6550],
[33.6000],
[33.9725],
[34.3300],

```
        [34.1100],
        [33.6400],
        [33.9950],
        [33.3600],
        [33.3000],
        [33.2850],
        [33.4100],
        [33.4020],
        [33.2822],
        [32.9307],
        [33.2472]], dtype=torch.float64), tensor([[33.5500]], dtype=torch.float64))
torch.Size([427, 1])
torch.Size([1, 1])
(tensor([[20.5430],
        [20.7570],
        [20.5910],
        [20.2690],
        [20.0480],
        [20.1780],
        [20.2260],
        [19.8960],
        [20.0880],
        [19.8400],
        [19.8900],
        [19.8540],
        [19.8800],
        [19.9700],
        [19.8320],
        [20.0530],
        [19.9630],
        [20.0300],
        [20.2180],
        [20.2850],
        [20.1150],
        [20.0550],
        [19.9300],
        [19.7750],
        [19.6520],
        [19.6820],
        [19.6920],
        [19.5940],
        [19.7210],
        [19.8350],
        [19.8750],
        [19.8850],
        [19.9390],
        [19.9680],
        [19.9410],
        [19.9100],
        [20.0310],
        [19.9780],
        [19.8550],
        [20.0580],
        [19.9590],
        [19.9900],
        [20.0590],
        [20.0540],
        [20.0680],
        [20.0730],
        [20.0580],
        [20.1890],
        [20.2550],
        [20.1810],
        [20.2240],
        [20.2680],
        [20.2880],
        [20.2680],
        [20.2680],
        [20.1510],
        [20.1310],
        [20.2500],
        [20.3180],
        [20.4080],
        [20.2980],
        [20.3180],
        [20.2880],
        [20.2200],
        [20.2500],
        [20.1810],
        [20.2480],
        [20.1710],
        [20.1910],
        [20.2780],
```

```
[20.1990],
[20.1610],
[20.1900],
[20.2290],
[20.2210],
[20.1210],
[20.1510],
[20.1510],
[20.1920],
[20.2100],
[20.0010],
[19.8830],
[20.0710],
[19.9510],
[19.9510],
[19.9370],
[19.9510],
[20.0910],
[20.1430],
[19.9350],
[19.9730],
[19.8130],
[19.9510],
[19.6450],
[19.5650],
[19.7220],
[19.8130],
[19.7530],
[20.0750],
[20.0610],
[20.1510],
[20.2230],
[20.1990],
[20.2380],
[20.3280],
[20.2890],
[20.2480],
[20.2800],
[20.4510],
[20.4320],
[20.2860],
[20.1310],
[20.1110],
[20.2480],
[20.1620],
[20.1420],
[20.3110],
[20.2380],
[20.2690],
[20.4770],
[20.0170],
[19.6250],
[19.8350],
[20.1920],
[20.4130],
[20.5160],
[20.4690],
[20.5590],
[20.4900],
[20.8510],
[20.9740],
[20.9860],
[21.0140],
[21.0440],
[20.9900],
[21.0340],
[20.9830],
[21.1130],
[20.7760],
[20.9940],
[20.9770],
[21.0340],
[21.0040],
[21.0940],
[21.1520],
[21.1520],
[20.9160],
[20.8830],
[20.9250],
[20.9960],
[21.0170],
[20.9890],
[21.0040],
```

[21.1160],
[21.1200],
[21.1180],
[20.9450],
[20.8960],
[20.9740],
[21.0140],
[21.0540],
[21.2020],
[20.9860],
[21.0840],
[21.0240],
[21.1100],
[21.0180],
[20.9940],
[21.0340],
[21.1860],
[21.1500],
[21.2320],
[21.0940],
[20.5790],
[20.6870],
[20.4450],
[20.3160],
[20.4550],
[20.3890],
[20.4060],
[20.3460],
[20.5670],
[20.7520],
[20.6730],
[20.4850],
[20.5350],
[20.5750],
[20.4060],
[20.6920],
[20.5900],
[20.4800],
[20.6100],
[20.7020],
[20.5590],
[20.6730],
[20.3880],
[20.5150],
[20.3580],
[20.3660],
[20.3460],
[20.3940],
[20.5250],
[20.4450],
[20.5050],
[20.5900],
[20.3960],
[20.2360],
[20.0810],
[20.1290],
[20.1970],
[19.9790],
[19.8810],
[19.8280],
[19.8520],
[20.1390],
[20.2170],
[20.7220],
[21.1000],
[21.4170],
[21.8730],
[21.9270],
[21.8920],
[22.0810],
[22.1400],
[22.1710],
[22.3920],
[22.5640],
[22.5900],
[22.3780],
[22.3640],
[22.2890],
[22.3680],
[22.4380],
[22.6820],
[22.8790],
[23.1900],

```
[23.4400],
[23.4000],
[23.0640],
[23.1110],
[23.0160],
[23.1260],
[22.9670],
[22.9490],
[23.0820],
[23.0860],
[22.8210],
[22.8640],
[22.9230],
[22.6690],
[22.6790],
[22.5690],
[22.6350],
[22.8990],
[22.6430],
[22.6990],
[22.5190],
[22.6840],
[22.7250],
[22.4860],
[22.5990],
[22.4000],
[22.4320],
[22.3110],
[22.3710],
[22.3210],
[22.6540],
[22.8590],
[22.7990],
[22.6720],
[22.5070],
[22.5610],
[22.5690],
[22.3310],
[22.5840],
[22.5690],
[22.5690],
[22.4690],
[22.7590],
[22.9280],
[23.0370],
[23.1470],
[23.2410],
[23.2540],
[23.1870],
[23.3750],
[23.3260],
[23.0870],
[23.0660],
[23.1960],
[23.0070],
[23.4350],
[23.2560],
[23.2760],
[23.1770],
[23.1060],
[23.0870],
[23.0120],
[23.1470],
[23.1790],
[23.1680],
[23.4270],
[23.3900],
[23.3870],
[23.2780],
[22.8560],
[22.8000],
[22.9000],
[22.7920],
[22.8150],
[23.0010],
[23.0330],
[23.1380],
[23.1430],
[22.7810],
[22.7500],
[22.6800],
[22.8200],
[22.7650],
```

```
[22.8070],
[22.9270],
[22.7450],
[22.5820],
[22.7300],
[22.7700],
[22.8550],
[23.2030],
[23.1580],
[23.4270],
[23.6910],
[23.8250],
[23.7250],
[23.5120],
[23.5240],
[23.4770],
[23.3870],
[23.4170],
[23.5120],
[23.4820],
[23.4370],
[23.4120],
[23.3410],
[23.1280],
[23.1830],
[23.1090],
[22.6250],
[22.6300],
[22.7990],
[22.8700],
[22.8500],
[22.8750],
[22.8810],
[22.9080],
[22.8600],
[22.9000],
[23.2000],
[23.3330],
[23.1190],
[23.0240],
[23.0380],
[23.1980],
[23.3480],
[23.4080],
[23.6000],
[23.5370],
[23.5470],
[23.3870],
[23.5070],
[23.3380],
[23.1780],
[22.9780],
[22.9980],
[23.1580],
[23.0440],
[23.3290],
[23.2490],
[23.3390],
[23.5350],
[23.4810],
[23.2490],
[23.4890],
[23.4590],
[23.2640],
[23.3840],
[23.3990],
[23.4990],
[23.4390],
[23.4090],
[23.4690],
[23.5840],
[23.5400],
[23.4890],
[23.5250],
[23.1900],
[23.0290],
[22.9670],
[23.0540],
[23.1140],
[23.0490],
[22.9120],
[23.0390],
[22.9790],
```

```
        [23.0340],
        [22.8600],
        [22.4450],
        [22.4400],
        [22.6510],
        [22.5550],
        [22.5300],
        [22.2150],
        [22.0340],
        [21.9860],
        [22.1230],
        [21.9610],
        [21.9790],
        [22.0760],
        [22.0310],
        [21.9960],
        [22.0180],
        [22.1160],
        [22.2400],
        [22.0020],
        [22.0810],
        [21.9970],
        [22.1510],
        [22.2650],
        [22.4200],
        [22.3600],
        [22.3670],
        [22.4100],
        [22.5700],
        [22.5750],
        [22.6800],
        [22.6600],
        [22.7800],
        [22.8536],
        [22.9350],
        [23.2250],
        [23.3210],
        [23.3136],
        [23.4300],
        [23.4300],
        [23.3810],
        [23.3950],
        [23.3900],
        [23.3800],
        [23.4400],
        [23.4250],
        [23.4410],
        [23.4850],
        [23.5584],
        [23.5100],
        [23.6000],
        [23.6050],
        [23.7750],
        [23.6850],
        [23.8100],
        [23.7038],
        [24.0000],
        [24.0783],
        [23.7200],
        [23.8046],
        [23.7387],
        [23.8000],
        [23.7850],
        [23.7900],
        [23.5800],
        [23.5794],
        [23.4631]], dtype=torch.float64), tensor([[23.5500]], dtype=torch.float64))
torch.Size([469, 1])
torch.Size([1, 1])
(tensor([[14.2900],
        [13.4100],
        [13.6300],
        [13.9200],
        [13.9800],
        [13.0000],
        [11.9400],
        [12.2300],
        [12.1200],
        [12.8200],
        [12.6100],
        [12.5555],
        [13.0600],
        [12.6700],
```

```
        [12.8000],
        [13.9300],
        [14.2000],
        [15.3000],
        [15.4800],
        [15.9500],
        [16.3600],
        [16.2300],
        [16.2400],
        [15.8800],
        [15.9700],
        [16.4000],
        [16.1100],
        [16.3400],
        [18.2600],
        [17.9900],
        [18.5800],
        [18.5500],
        [19.8300],
        [21.3500],
        [20.5600],
        [20.2900],
        [20.3800]], dtype=torch.float64), tensor([[21.6500]], dtype=torch.float64))
torch.Size([37, 1])
torch.Size([1, 1])
(tensor([[ 60.6680],
        [ 60.9340],
        [ 59.2990],
        [ 56.4730],
        [ 55.2640],
        [ 55.2940],
        [ 56.1560],
        [ 53.3990],
        [ 55.1450],
        [ 52.7560],
        [ 52.8850],
        [ 51.6350],
        [ 52.1710],
        [ 54.2820],
        [ 52.6360],
        [ 54.0440],
        [ 52.9050],
        [ 53.4690],
        [ 55.9870],
        [ 56.0370],
        [ 53.9350],
        [ 54.6110],
        [ 54.7680],
        [ 52.7360],
        [ 51.2590],
        [ 51.2190],
        [ 51.1590],
        [ 49.9310],
        [ 51.8940],
        [ 53.6070],
        [ 55.3830],
        [ 54.9370],
        [ 54.8770],
        [ 56.4540],
        [ 55.1150],
        [ 55.5810],
        [ 56.9080],
        [ 56.6430],
        [ 55.6410],
        [ 58.3870],
        [ 58.9130],
        [ 59.3190],
        [ 59.6770],
        [ 59.8550],
        [ 58.4870],
        [ 59.0810],
        [ 59.1400],
        [ 61.1040],
        [ 60.8940],
        [ 60.7680],
        [ 61.4600],
        [ 62.2040],
        [ 62.7100],
        [ 62.8700],
        [ 62.7900],
        [ 62.0070],
        [ 61.9280],
        [ 61.9780],
```

[63.2000],
[63.7060],
[63.4180],
[64.2330],
[63.8260],
[62.5630],
[63.8760],
[62.3450],
[62.7330],
[62.3940],
[63.5970],
[64.8300],
[64.8400],
[64.6600],
[65.5850],
[65.9720],
[66.1200],
[65.4350],
[65.4550],
[65.1570],
[65.3950],
[65.6710],
[64.4910],
[63.7960],
[64.7500],
[63.6270],
[62.8830],
[62.9310],
[63.3580],
[63.4870],
[64.9990],
[63.8160],
[63.8460],
[62.7330],
[63.9450],
[62.8030],
[62.8230],
[62.3840],
[63.1010],
[62.9410],
[64.5610],
[65.4050],
[65.4750],
[66.0410],
[65.8230],
[66.0900],
[66.4600],
[66.0700],
[66.7670],
[66.9360],
[67.3640],
[67.1840],
[65.9420],
[64.9100],
[64.6600],
[64.4410],
[64.8600],
[64.3720],
[65.1660],
[65.5350],
[65.3130],
[66.9840],
[62.1190],
[59.9300],
[62.0790],
[64.2480],
[65.9500],
[66.2380],
[65.3030],
[66.0780],
[65.9700],
[67.9490],
[68.4570],
[69.4320],
[69.3920],
[70.1580],
[70.0090],
[70.3770],
[70.2080],
[70.8150],
[70.2380],
[70.9250],
[70.5470],

[70.5570],
[70.4070],
[70.5570],
[70.8050],
[70.6670],
[69.7910],
[70.2080],
[70.2970],
[71.4220],
[71.3920],
[71.4320],
[71.1240],
[71.7400],
[71.6210],
[72.0300],
[71.3130],
[71.5320],
[71.8700],
[71.6800],
[71.6510],
[71.8900],
[71.2330],
[71.1040],
[70.8150],
[71.4920],
[71.2430],
[70.8750],
[70.8550],
[71.4820],
[71.9300],
[71.9400],
[71.6010],
[68.1190],
[70.0790],
[68.0590],
[67.9690],
[69.3920],
[68.8260],
[68.8460],
[68.8650],
[70.3490],
[71.2760],
[70.4790],
[69.3530],
[70.2000],
[70.8970],
[69.6330],
[70.6790],
[70.2700],
[69.6530],
[70.2100],
[70.3290],
[69.8120],
[70.5090],
[68.7970],
[68.9460],
[68.4670],
[68.5370],
[68.0890],
[68.8560],
[69.1940],
[68.9950],
[69.0150],
[69.5830],
[69.1640],
[68.8660],
[68.5270],
[68.0990],
[68.0790],
[67.1420],
[66.2960],
[65.6890],
[65.5100],
[68.3570],
[68.9760],
[70.5190],
[70.8270],
[70.5090],
[70.6290],
[71.7030],
[71.4550],
[72.1930],
[71.8730],

[72.8890],
[73.2090],
[73.2880],
[73.8250],
[73.1890],
[73.4270],
[72.9990],
[72.4500],
[72.5700],
[73.4270],
[73.9050],
[75.7860],
[76.2260],
[77.0920],
[76.9220],
[77.9580],
[76.6930],
[77.2520],
[77.0220],
[77.2720],
[77.8580],
[77.5140],
[77.2350],
[77.4240],
[77.7930],
[76.5570],
[76.4770],
[75.9180],
[77.0350],
[77.9420],
[77.8030],
[78.3710],
[77.8520],
[77.8420],
[78.2710],
[77.9120],
[78.2410],
[77.6830],
[78.0120],
[77.4740],
[77.9320],
[77.5940],
[78.5110],
[79.8470],
[79.7580],
[79.5480],
[78.5910],
[78.5510],
[78.5710],
[78.6110],
[79.7180],
[79.4380],
[79.4680],
[79.6780],
[80.5250],
[81.1520],
[82.0600],
[82.7590],
[83.5760],
[83.4760],
[83.6460],
[84.6530],
[84.5430],
[84.6930],
[84.8420],
[85.1510],
[84.7030],
[86.9460],
[85.9790],
[86.0290],
[85.5010],
[85.0320],
[84.6530],
[84.8720],
[85.4010],
[85.5310],
[84.9120],
[86.3080],
[86.0190],
[85.6900],
[85.4810],
[83.2870],
[83.6850],

[83.4850],
[83.3850],
[83.1660],
[84.3540],
[84.5630],
[85.0230],
[84.6830],
[84.3740],
[84.5130],
[83.9750],
[84.3640],
[84.2140],
[84.3140],
[84.1240],
[83.4250],
[82.3570],
[83.7550],
[83.2060],
[82.9270],
[84.2140],
[83.7450],
[85.5220],
[86.5000],
[86.4500],
[86.5600],
[86.2500],
[86.6100],
[86.6690],
[86.4600],
[86.6300],
[87.3780],
[87.3180],
[87.1390],
[87.4080],
[87.0790],
[86.8390],
[87.6580],
[87.5580],
[84.4830],
[85.0820],
[86.2100],
[87.0590],
[87.4380],
[87.8270],
[88.6160],
[88.6560],
[88.4860],
[88.4460],
[89.7740],
[90.3630],
[90.2930],
[89.7640],
[90.0130],
[90.1230],
[89.8340],
[89.7740],
[90.6220],
[90.4620],
[90.0730],
[90.0830],
[91.5400],
[90.3630],
[90.2750],
[90.1850],
[90.4340],
[90.4740],
[89.0860],
[90.5740],
[89.0860],
[89.3650],
[89.6850],
[90.0150],
[88.4360],
[89.5450],
[89.7350],
[89.5450],
[90.9140],
[91.1840],
[92.0030],
[92.0330],
[92.1330],
[93.0920],
[93.1420],

```
        [ 93.0320],
        [ 92.9420],
        [ 93.3420],
        [ 93.4320],
        [ 93.2420],
        [ 92.9820],
        [ 92.8920],
        [ 93.2420],
        [ 93.3320],
        [ 93.0420],
        [ 93.3320],
        [ 93.6410],
        [ 93.2420],
        [ 93.1320],
        [ 90.5840],
        [ 90.7640],
        [ 92.5920],
        [ 92.5720],
        [ 92.8620],
        [ 89.9850],
        [ 89.6750],
        [ 89.8250],
        [ 91.6130],
        [ 91.0240],
        [ 90.6140],
        [ 91.0040],
        [ 91.0440],
        [ 91.1740],
        [ 92.0530],
        [ 93.1820],
        [ 93.4720],
        [ 92.0930],
        [ 92.6920],
        [ 92.6720],
        [ 92.4130],
        [ 94.3210],
        [ 95.0000],
        [ 95.1100],
        [ 95.0400],
        [ 95.3200],
        [ 95.6890],
        [ 95.8290],
        [ 95.9090],
        [ 95.3800],
        [ 95.4200],
        [ 95.0200],
        [ 95.1200],
        [ 95.8600],
        [ 96.0800],
        [ 96.7400],
        [ 97.5200],
        [ 97.9900],
        [ 98.2300],
        [ 99.3300],
        [ 99.1400],
        [ 98.8100],
        [ 99.3000],
        [ 99.5800],
        [ 99.2600],
        [ 99.5200],
        [ 99.8000],
        [ 99.9400],
        [100.0700],
        [100.1200],
        [101.1600],
        [100.3800],
        [100.6900],
        [ 99.7100],
        [ 99.9400],
        [101.5400],
        [100.8200],
        [101.1200],
        [101.3900],
        [101.4500],
        [102.0900],
        [102.3600],
        [102.3300],
        [102.6600],
        [101.8800]], dtype=torch.float64), tensor([[101.7900]], dtype=torch.float64))
torch.Size([469, 1])
torch.Size([1, 1])
(tensor([[0.1900],
        [0.1900],
```

[0.1500],
[0.1500],
[0.1500],
[0.2500],
[0.2300],
[0.1501],
[0.1951],
[0.2500],
[0.2200],
[0.2000],
[0.2000],
[0.2000],
[0.2200],
[0.1500],
[0.1945],
[0.2000],
[0.2000],
[0.2000],
[0.1800],
[0.1800],
[0.1800],
[0.1800],
[0.1500],
[0.1600],
[0.0900],
[0.0800],
[0.0600],
[0.0750],
[0.0800],
[0.1000],
[0.0800],
[0.0784],
[0.0810],
[0.0883],
[0.1016],
[0.0984],
[0.0984],
[0.0900],
[0.0801],
[0.0899],
[0.0900],
[0.0900],
[0.0900],
[0.0900],
[0.0800],
[0.0800],
[0.0900],
[0.0900],
[0.0850],
[0.0600],
[0.0600],
[0.0600],
[0.0564],
[0.0600],
[0.0600],
[0.0600],
[0.0601],
[0.0600],
[0.0655],
[0.0800],
[0.0850],
[0.0700],
[0.0700],
[0.0693],
[0.0600],
[0.0694],
[0.0700],
[0.0700],
[0.0700],
[0.0700],
[0.0700],
[0.0630],
[0.0600],
[0.0600],
[0.0525],
[0.0589],
[0.0660],
[0.0512],
[0.0510],
[0.0410],
[0.0550],
[0.0500],
[0.0405],

[0.0500],
[0.0500],
[0.0400],
[0.0421],
[0.0300],
[0.0339],
[0.0350],
[0.0400],
[0.0350],
[0.0350],
[0.0370],
[0.0400],
[0.0400],
[0.0300],
[0.0316],
[0.0426],
[0.0450],
[0.0600],
[0.0550],
[0.0600],
[0.0801],
[0.0900],
[0.0800],
[0.0900],
[0.2500],
[0.2100],
[0.2000],
[0.2000],
[0.2700],
[0.2570],
[0.2200],
[0.1900],
[0.1800],
[0.1900],
[0.2100],
[0.2500],
[0.2100],
[0.2500],
[0.2200],
[0.1810],
[0.2100],
[0.2100],
[0.1408],
[0.2500],
[0.2400],
[0.2500],
[0.2316],
[0.2200],
[0.2001],
[0.2085],
[0.2000],
[0.2000],
[0.2200],
[0.2200],
[0.2063],
[0.2000],
[0.2000],
[0.2029],
[0.2576],
[0.3000],
[0.3090],
[0.3000],
[0.2700],
[0.2600],
[0.2500],
[0.2619],
[0.2700],
[0.2900],
[0.3000],
[0.3000],
[0.2900],
[0.3100],
[0.3200],
[0.3100],
[0.3100],
[0.3049],
[0.3200],
[0.3600],
[0.3200],
[0.3200],
[0.3200],
[0.3300],
[0.3500],

```
[0.3800],
[0.4000],
[0.3800],
[0.3581],
[0.4000],
[0.4000],
[0.3500],
[0.3474],
[0.3200],
[0.3401],
[0.3630],
[0.3400],
[0.3100],
[0.3400],
[0.2801],
[0.2700],
[0.2823],
[0.2700],
[0.2500],
[0.2499],
[0.2000],
[0.2200],
[0.2500],
[0.2338],
[0.2200],
[0.2300],
[0.2300],
[0.2046],
[0.2200],
[0.2425],
[0.2300],
[0.2159],
[0.2201],
[0.2224],
[0.2100],
[0.2300],
[0.2300],
[0.3300],
[0.2900],
[0.2743],
[0.2600],
[0.2530],
[0.2400],
[0.2902],
[0.2910],
[0.2509],
[0.2200],
[0.2300],
[0.2200],
[0.2500],
[0.2500],
[0.2500],
[0.2467],
[0.2250],
[0.2300],
[0.2500],
[0.2200],
[0.2069],
[0.2200],
[0.1800],
[0.2300],
[0.2600],
[0.2400],
[0.2400],
[0.2300],
[0.2300],
[0.2000],
[0.1600],
[0.2200],
[0.2200],
[0.2000],
[0.2000],
[0.1918],
[0.1650],
[0.2000],
[0.1700],
[0.2200],
[0.2701],
[0.2512],
[0.2500],
[0.2500],
[0.2100],
[0.2300],
```

```
        [0.1701],
        [0.1710],
        [0.1563],
        [0.1196]], dtype=torch.float64), tensor([[0.1300]], dtype=torch.float64))
torch.Size([255, 1])
torch.Size([1, 1])
(tensor([[21.1900],
        [21.4200],
        [20.9700],
        [20.0600],
        [19.2000],
        [18.1500],
        [19.4900],
        [18.7800],
        [19.9400],
        [18.9000],
        [18.8800],
        [18.2700],
        [18.3100],
        [18.6500],
        [18.2400],
        [18.0500],
        [16.9000],
        [16.4000],
        [17.8300],
        [18.1400],
        [18.4200],
        [17.6600],
        [18.0800],
        [17.0800],
        [16.5400],
        [17.2000],
        [17.5200],
        [17.2300],
        [16.9400],
        [17.6000],
        [18.3400],
        [18.4700],
        [18.7100],
        [18.0000],
        [17.5800],
        [18.4500],
        [18.7100],
        [18.7500],
        [18.0800],
        [18.3900],
        [18.2900],
        [18.5400],
        [18.7800],
        [18.7500],
        [18.0300],
        [18.2300],
        [17.8800],
        [18.4700],
        [18.3700],
        [17.4600],
        [16.7900],
        [16.1200],
        [16.6500],
        [16.3700],
        [16.8100],
        [16.3800],
        [16.1500],
        [16.1500],
        [17.7700],
        [18.4600],
        [19.0000],
        [19.0000],
        [18.5900],
        [18.6400],
        [18.8100],
        [18.6900],
        [18.2400],
        [17.6900],
        [17.8600],
        [18.3500],
        [18.4400],
        [18.6900],
        [19.4400],
        [19.2100],
        [18.8000],
        [19.1800],
        [19.2000],
```

```
[19.1400],
[19.6600],
[19.6800],
[20.0400],
[20.0900],
[20.2475],
[20.5200],
[20.1000],
[20.3400],
[13.8100],
[13.9850],
[14.4200],
[13.6500],
[12.2600],
[12.3500],
[12.5900],
[12.5100],
[12.7000],
[12.6400],
[13.2400],
[12.8600],
[12.9400],
[12.9200],
[12.6000],
[13.1100],
[13.1300],
[13.5050],
[13.7800],
[13.7800],
[14.2200],
[14.6300],
[14.7700],
[14.6900],
[14.4100],
[14.4100],
[14.3700],
[14.2100],
[14.1300],
[13.7200],
[13.9500],
[13.1000],
[12.9600],
[13.5400],
[12.8300],
[12.1900],
[12.5200],
[12.8500],
[12.9300],
[13.0500],
[12.8000],
[13.1300],
[13.1300],
[13.5800],
[13.5100],
[13.5100],
[13.2400],
[13.5200],
[13.8200],
[15.2600],
[15.7400],
[15.6500],
[15.5400],
[15.5900],
[15.5800],
[15.7000],
[15.9200],
[15.6500],
[15.8400],
[16.0900],
[16.0200],
[16.3700],
[15.5000],
[15.1500],
[14.7500],
[14.9300],
[14.5100],
[14.6100],
[14.4200],
[14.8200],
[14.6700],
[14.5800],
[14.5200],
[14.7000],
```

[15.0600],
[15.4200],
[15.5100],
[15.8100],
[15.6400],
[15.8400],
[15.9500],
[15.7600],
[15.7050],
[15.9800],
[15.7000],
[15.5700],
[15.4800],
[14.6000],
[14.9700],
[14.8200],
[14.7500],
[14.7000],
[14.6900],
[14.7000],
[14.9400],
[15.4000],
[15.8200],
[15.7800],
[16.0200],
[16.2100],
[16.1200],
[15.6100],
[15.8400],
[16.0900],
[16.5300],
[16.6000],
[16.2300],
[16.1900],
[16.5000],
[16.2000],
[16.1000],
[16.5000],
[16.6000],
[17.2000],
[17.2500],
[16.6000],
[16.5000],
[16.6500],
[16.7000],
[16.2000],
[15.5500],
[15.3500],
[14.7000],
[14.6000],
[14.7000],
[8.3500],
[9.2500],
[10.4500],
[11.6000],
[12.0000],
[12.2000],
[12.7000],
[12.6500],
[12.4000],
[12.7000],
[12.2500],
[11.8000],
[11.9500],
[11.8500],
[11.3500],
[11.3500],
[11.2500],
[10.8500],
[10.7500],
[10.4000],
[9.7500],
[9.4500],
[9.2500],
[9.7500],
[10.6000],
[10.8000],
[11.2000],
[11.0500],
[11.1500],
[11.2000],
[11.0500],
[11.3500],

```
[11.1500],
[11.4000],
[11.5000],
[11.5500],
[11.8500],
[12.3000],
[11.6000],
[11.9500],
[12.1000],
[12.0000],
[12.0000],
[12.2000],
[11.8500],
[13.0500],
[13.1500],
[13.2500],
[13.1500],
[13.3500],
[13.3500],
[13.0500],
[12.6000],
[12.4500],
[12.0000],
[12.2500],
[12.8000],
[12.9500],
[13.2500],
[13.0000],
[13.4500],
[13.3500],
[13.4500],
[13.6500],
[13.4500],
[12.9500],
[12.6500],
[13.3500],
[13.3500],
[13.2000],
[13.2000],
[13.2000],
[13.1500],
[13.2000],
[12.8500],
[12.8500],
[12.5500],
[13.2000],
[13.8500],
[13.6000],
[14.4000],
[14.3500],
[13.9500],
[14.0000],
[14.1000],
[14.6500],
[14.8500],
[15.4000],
[15.4500],
[15.0500],
[15.6000],
[15.9500],
[16.4000],
[16.4000],
[16.3500],
[15.7000],
[16.3500],
[16.8000],
[17.3000],
[17.1000],
[17.4000],
[17.4000],
[17.1500],
[16.9500],
[16.5000],
[16.3500],
[16.2000],
[16.3500],
[16.0500],
[16.1500],
[16.2000],
[16.2000],
[16.6000],
[16.6000],
[17.2000],
```

```
[17.9000],
[17.5500],
[17.6000],
[18.0500],
[18.3000],
[18.6000],
[19.0000],
[19.9500],
[20.0500],
[24.1000],
[25.0000],
[24.4500],
[24.3500],
[24.4000],
[22.4000],
[23.2250],
[23.1000],
[23.3000],
[23.3000],
[22.2000],
[22.3000],
[22.7500],
[23.2000],
[23.2000],
[23.7500],
[24.0000],
[25.1500],
[25.3000],
[25.3000],
[26.0500],
[27.0500],
[26.3000],
[26.3000],
[26.1000],
[27.1500],
[26.0000],
[25.9000],
[26.0500],
[26.7500],
[26.6500],
[26.0500],
[26.4000],
[26.7500],
[26.6500],
[27.1000],
[27.6500],
[27.1500],
[26.9000],
[27.2500],
[27.0500],
[27.9500],
[28.1000],
[27.9500],
[27.2500],
[27.5000],
[27.5000],
[27.6000],
[27.8500],
[28.1000],
[29.0000],
[28.3000],
[29.0500],
[28.6500],
[28.5000],
[27.9000],
[28.6000],
[28.4500],
[28.2000],
[27.3000],
[27.3000],
[27.4000],
[27.5500],
[30.1000],
[29.5500],
[29.7000],
[30.5250],
[30.5500],
[30.1500],
[29.6500],
[30.4000],
[30.9000],
[31.0500],
[31.1500],
```

```
        [30.5000],
        [30.2000],
        [30.3500],
        [30.6000],
        [30.9000],
        [30.7500],
        [29.5500],
        [29.7500],
        [30.1000],
        [30.0000],
        [30.9000],
        [31.6000],
        [31.7500],
        [31.7500],
        [31.8000],
        [31.9500],
        [32.8000],
        [31.0000],
        [30.3000],
        [30.4000],
        [30.3000],
        [30.6000],
        [30.3750],
        [29.2500],
        [29.2500],
        [29.7500],
        [28.9000],
        [28.5500],
        [28.6000],
        [29.1500],
        [31.1500],
        [30.8000],
        [29.5000],
        [30.8000],
        [31.6000],
        [30.6500],
        [29.3500],
        [28.7000],
        [28.4000],
        [27.8500],
        [28.4000],
        [28.7000],
        [28.6500],
        [29.1500],
        [29.8000],
        [30.6000],
        [30.0000],
        [29.6000],
        [29.5000],
        [29.3500],
        [29.7500],
        [29.0250],
        [29.6500],
        [28.3000],
        [28.7500],
        [28.9500],
        [28.4500],
        [28.6500],
        [28.5000],
        [28.8000]], dtype=torch.float64), tensor([[28.1500]], dtype=torch.float64))
torch.Size([469, 1])
torch.Size([1, 1])
```

Data Augmentation

After having discussions with our TA, we determined that having 10,000 total data points is necessary. Thus, since our dataset originally had only 8,500 datapoints (sequences), we will add an additional 1,500 datapoints. To do this, we pick a random 1,500 datapoints from training, and random "gaussian noise" to these values and add them to the training set.

In [ ]:

```
# Create copy of training set
train_copy = train.copy()

rng = np.random.RandomState(31415926)
# Add Random Noise to the data within train_copy
for data, labels in train_copy:
  myDataNoise = rng.normal(loc=0,
                           scale=0.3,
                           size=(data.shape[0], data.shape[1]))
```

```
  myLabelNoise = rng.normal(loc=0,
                            scale=0.3,
                            size=(labels.shape[0], labels.shape[1]))

  data += myDataNoise
  labels += myLabelNoise

# Pick a random selection of these points to include in the training set
# Note: Change size to produce more random points
# Picking 1500 so that we have 10000 total points
rnd_array = np.random.randint(len(train), size=1500)

for i in rnd_array:
  train.append(train_copy[i])

print(len(train))
```

```
6605
```

Batch Data by length

Will use batching to combat the use of padding and improve runtime (using same idea from Tweet Sentiment Analysis tutorial from class). We batch together sequences of stock data that contain the same amount of data (are the same length).

In [ ]:

```
import random

class StockBatcher:
    def __init__(self, stocks, batch_size=128, drop_last=False):
        # store stocks by length
        self.stock_by_length = {}
        for stock_data, label in stocks:
            # compute the length of the stock_data
            slen = stock_data.shape[0]
            # Do not consider empty datapoints
            if slen == 0:
              continue
            # put the stock in the correct key inside self.stock_by_length
            if slen not in self.stock_by_length:
                self.stock_by_length[slen] = []
            self.stock_by_length[slen].append((stock_data.float(),
                                               label.float()))

        #  create a DataLoader for each set of stocks with the same amount of data
        self.loaders = {slen : torch.utils.data.DataLoader(
                                    stock_data,
                                    batch_size=batch_size,
                                    shuffle=True,
                                    drop_last=drop_last) # omit last batch if smaller than batch_size
            for slen, stock_data in self.stock_by_length.items()}

    def __iter__(self): # called by Python to create an iterator
        # make an iterator for every tweet length
        iters = [iter(loader) for loader in self.loaders.values()]
        while iters:
            # pick an iterator (a length)
            im = random.choice(iters)
            try:
                yield next(im)
            except StopIteration:
                # no more elements in the iterator, remove it
                iters.remove(im)
```

Taking a look at the stock batcher data, to get a feel for the format of the data

In [ ]:

```
for i, (stock_data, labels) in enumerate(StockBatcher(train, drop_last=False)):
    print(stock_data.shape, labels.shape)
```

```
torch.Size([1, 396, 1]) torch.Size([1, 1, 1])
torch.Size([6, 329, 1]) torch.Size([6, 1, 1])
torch.Size([10, 2, 1]) torch.Size([10, 1, 1])
torch.Size([4, 302, 1]) torch.Size([4, 1, 1])
torch.Size([7, 449, 1]) torch.Size([7, 1, 1])
torch.Size([4, 151, 1]) torch.Size([4, 1, 1])
torch.Size([2, 349, 1]) torch.Size([2, 1, 1])
torch.Size([2, 265, 1]) torch.Size([2, 1, 1])
torch.Size([3, 308, 1]) torch.Size([3, 1, 1])
torch.Size([3, 48, 1]) torch.Size([3, 1, 1])
torch.Size([9, 6, 1]) torch.Size([9, 1, 1])
```

```
torch.Size([2, 348, 1]) torch.Size([2, 1, 1])
torch.Size([38, 463, 1]) torch.Size([38, 1, 1])
torch.Size([7, 21, 1]) torch.Size([7, 1, 1])
torch.Size([5, 45, 1]) torch.Size([5, 1, 1])
torch.Size([1, 243, 1]) torch.Size([1, 1, 1])
torch.Size([22, 456, 1]) torch.Size([22, 1, 1])
torch.Size([3, 261, 1]) torch.Size([3, 1, 1])
torch.Size([37, 465, 1]) torch.Size([37, 1, 1])
torch.Size([1, 104, 1]) torch.Size([1, 1, 1])
torch.Size([2, 173, 1]) torch.Size([2, 1, 1])
torch.Size([1, 36, 1]) torch.Size([1, 1, 1])
torch.Size([1, 129, 1]) torch.Size([1, 1, 1])
torch.Size([4, 234, 1]) torch.Size([4, 1, 1])
torch.Size([6, 422, 1]) torch.Size([6, 1, 1])
torch.Size([128, 468, 1]) torch.Size([128, 1, 1])
torch.Size([2, 46, 1]) torch.Size([2, 1, 1])
torch.Size([6, 47, 1]) torch.Size([6, 1, 1])
torch.Size([5, 426, 1]) torch.Size([5, 1, 1])
torch.Size([3, 258, 1]) torch.Size([3, 1, 1])
torch.Size([3, 355, 1]) torch.Size([3, 1, 1])
torch.Size([3, 385, 1]) torch.Size([3, 1, 1])
torch.Size([14, 457, 1]) torch.Size([14, 1, 1])
torch.Size([7, 118, 1]) torch.Size([7, 1, 1])
torch.Size([9, 413, 1]) torch.Size([9, 1, 1])
torch.Size([5, 405, 1]) torch.Size([5, 1, 1])
torch.Size([3, 382, 1]) torch.Size([3, 1, 1])
torch.Size([8, 84, 1]) torch.Size([8, 1, 1])
torch.Size([9, 279, 1]) torch.Size([9, 1, 1])
torch.Size([2, 223, 1]) torch.Size([2, 1, 1])
torch.Size([6, 169, 1]) torch.Size([6, 1, 1])
torch.Size([6, 291, 1]) torch.Size([6, 1, 1])
torch.Size([4, 209, 1]) torch.Size([4, 1, 1])
torch.Size([19, 458, 1]) torch.Size([19, 1, 1])
torch.Size([5, 262, 1]) torch.Size([5, 1, 1])
torch.Size([4, 417, 1]) torch.Size([4, 1, 1])
torch.Size([1, 202, 1]) torch.Size([1, 1, 1])
torch.Size([9, 40, 1]) torch.Size([9, 1, 1])
torch.Size([4, 111, 1]) torch.Size([4, 1, 1])
torch.Size([8, 16, 1]) torch.Size([8, 1, 1])
torch.Size([1, 191, 1]) torch.Size([1, 1, 1])
torch.Size([3, 34, 1]) torch.Size([3, 1, 1])
torch.Size([2, 252, 1]) torch.Size([2, 1, 1])
torch.Size([3, 93, 1]) torch.Size([3, 1, 1])
torch.Size([8, 341, 1]) torch.Size([8, 1, 1])
torch.Size([1, 294, 1]) torch.Size([1, 1, 1])
torch.Size([1, 198, 1]) torch.Size([1, 1, 1])
torch.Size([5, 402, 1]) torch.Size([5, 1, 1])
torch.Size([5, 229, 1]) torch.Size([5, 1, 1])
torch.Size([5, 189, 1]) torch.Size([5, 1, 1])
torch.Size([5, 26, 1]) torch.Size([5, 1, 1])
torch.Size([8, 445, 1]) torch.Size([8, 1, 1])
torch.Size([4, 23, 1]) torch.Size([4, 1, 1])
torch.Size([4, 326, 1]) torch.Size([4, 1, 1])
torch.Size([5, 66, 1]) torch.Size([5, 1, 1])
torch.Size([4, 222, 1]) torch.Size([4, 1, 1])
torch.Size([1, 156, 1]) torch.Size([1, 1, 1])
torch.Size([1, 157, 1]) torch.Size([1, 1, 1])
torch.Size([6, 27, 1]) torch.Size([6, 1, 1])
torch.Size([1, 153, 1]) torch.Size([1, 1, 1])
torch.Size([2, 170, 1]) torch.Size([2, 1, 1])
torch.Size([4, 81, 1]) torch.Size([4, 1, 1])
torch.Size([9, 37, 1]) torch.Size([9, 1, 1])
torch.Size([1, 322, 1]) torch.Size([1, 1, 1])
torch.Size([2, 240, 1]) torch.Size([2, 1, 1])
torch.Size([2, 135, 1]) torch.Size([2, 1, 1])
torch.Size([3, 270, 1]) torch.Size([3, 1, 1])
torch.Size([3, 242, 1]) torch.Size([3, 1, 1])
torch.Size([1, 251, 1]) torch.Size([1, 1, 1])
torch.Size([5, 307, 1]) torch.Size([5, 1, 1])
torch.Size([2, 275, 1]) torch.Size([2, 1, 1])
torch.Size([8, 359, 1]) torch.Size([8, 1, 1])
torch.Size([6, 337, 1]) torch.Size([6, 1, 1])
torch.Size([5, 292, 1]) torch.Size([5, 1, 1])
torch.Size([6, 280, 1]) torch.Size([6, 1, 1])
torch.Size([5, 269, 1]) torch.Size([5, 1, 1])
torch.Size([3, 319, 1]) torch.Size([3, 1, 1])
torch.Size([1, 190, 1]) torch.Size([1, 1, 1])
torch.Size([2, 296, 1]) torch.Size([2, 1, 1])
torch.Size([4, 266, 1]) torch.Size([4, 1, 1])
torch.Size([2, 215, 1]) torch.Size([2, 1, 1])
torch.Size([4, 408, 1]) torch.Size([4, 1, 1])
torch.Size([3, 95, 1]) torch.Size([3, 1, 1])
torch.Size([2, 55, 1]) torch.Size([2, 1, 1])
```

```
torch.Size([13, 427, 1]) torch.Size([13, 1, 1])
torch.Size([17, 453, 1]) torch.Size([17, 1, 1])
torch.Size([1, 257, 1]) torch.Size([1, 1, 1])
torch.Size([2, 204, 1]) torch.Size([2, 1, 1])
torch.Size([1, 75, 1]) torch.Size([1, 1, 1])
torch.Size([3, 62, 1]) torch.Size([3, 1, 1])
torch.Size([2, 366, 1]) torch.Size([2, 1, 1])
torch.Size([2, 39, 1]) torch.Size([2, 1, 1])
torch.Size([3, 30, 1]) torch.Size([3, 1, 1])
torch.Size([2, 345, 1]) torch.Size([2, 1, 1])
torch.Size([6, 406, 1]) torch.Size([6, 1, 1])
torch.Size([3, 193, 1]) torch.Size([3, 1, 1])
torch.Size([2, 225, 1]) torch.Size([2, 1, 1])
torch.Size([4, 374, 1]) torch.Size([4, 1, 1])
torch.Size([3, 96, 1]) torch.Size([3, 1, 1])
torch.Size([2, 32, 1]) torch.Size([2, 1, 1])
torch.Size([2, 364, 1]) torch.Size([2, 1, 1])
torch.Size([3, 94, 1]) torch.Size([3, 1, 1])
torch.Size([3, 14, 1]) torch.Size([3, 1, 1])
torch.Size([8, 20, 1]) torch.Size([8, 1, 1])
torch.Size([3, 110, 1]) torch.Size([3, 1, 1])
torch.Size([5, 171, 1]) torch.Size([5, 1, 1])
torch.Size([3, 58, 1]) torch.Size([3, 1, 1])
torch.Size([4, 141, 1]) torch.Size([4, 1, 1])
torch.Size([9, 432, 1]) torch.Size([9, 1, 1])
torch.Size([3, 71, 1]) torch.Size([3, 1, 1])
torch.Size([6, 333, 1]) torch.Size([6, 1, 1])
torch.Size([6, 420, 1]) torch.Size([6, 1, 1])
torch.Size([1, 131, 1]) torch.Size([1, 1, 1])
torch.Size([4, 332, 1]) torch.Size([4, 1, 1])
torch.Size([1, 217, 1]) torch.Size([1, 1, 1])
torch.Size([2, 325, 1]) torch.Size([2, 1, 1])
torch.Size([5, 122, 1]) torch.Size([5, 1, 1])
torch.Size([4, 299, 1]) torch.Size([4, 1, 1])
torch.Size([2, 392, 1]) torch.Size([2, 1, 1])
torch.Size([2, 44, 1]) torch.Size([2, 1, 1])
torch.Size([3, 228, 1]) torch.Size([3, 1, 1])
torch.Size([1, 354, 1]) torch.Size([1, 1, 1])
torch.Size([1, 35, 1]) torch.Size([1, 1, 1])
torch.Size([6, 380, 1]) torch.Size([6, 1, 1])
torch.Size([5, 314, 1]) torch.Size([5, 1, 1])
torch.Size([8, 440, 1]) torch.Size([8, 1, 1])
torch.Size([4, 264, 1]) torch.Size([4, 1, 1])
torch.Size([9, 11, 1]) torch.Size([9, 1, 1])
torch.Size([4, 435, 1]) torch.Size([4, 1, 1])
torch.Size([8, 286, 1]) torch.Size([8, 1, 1])
torch.Size([37, 466, 1]) torch.Size([37, 1, 1])
torch.Size([6, 443, 1]) torch.Size([6, 1, 1])
torch.Size([2, 196, 1]) torch.Size([2, 1, 1])
torch.Size([3, 154, 1]) torch.Size([3, 1, 1])
torch.Size([1, 207, 1]) torch.Size([1, 1, 1])
torch.Size([5, 41, 1]) torch.Size([5, 1, 1])
torch.Size([1, 317, 1]) torch.Size([1, 1, 1])
torch.Size([1, 197, 1]) torch.Size([1, 1, 1])
torch.Size([2, 126, 1]) torch.Size([2, 1, 1])
torch.Size([2, 109, 1]) torch.Size([2, 1, 1])
torch.Size([5, 311, 1]) torch.Size([5, 1, 1])
torch.Size([8, 438, 1]) torch.Size([8, 1, 1])
torch.Size([8, 451, 1]) torch.Size([8, 1, 1])
torch.Size([8, 433, 1]) torch.Size([8, 1, 1])
torch.Size([7, 12, 1]) torch.Size([7, 1, 1])
torch.Size([5, 393, 1]) torch.Size([5, 1, 1])
torch.Size([9, 15, 1]) torch.Size([9, 1, 1])
torch.Size([8, 431, 1]) torch.Size([8, 1, 1])
torch.Size([1, 125, 1]) torch.Size([1, 1, 1])
torch.Size([4, 407, 1]) torch.Size([4, 1, 1])
torch.Size([3, 247, 1]) torch.Size([3, 1, 1])
torch.Size([4, 138, 1]) torch.Size([4, 1, 1])
torch.Size([6, 132, 1]) torch.Size([6, 1, 1])
torch.Size([8, 29, 1]) torch.Size([8, 1, 1])
torch.Size([6, 436, 1]) torch.Size([6, 1, 1])
torch.Size([6, 397, 1]) torch.Size([6, 1, 1])
torch.Size([1, 117, 1]) torch.Size([1, 1, 1])
torch.Size([3, 33, 1]) torch.Size([3, 1, 1])
torch.Size([3, 351, 1]) torch.Size([3, 1, 1])
torch.Size([3, 77, 1]) torch.Size([3, 1, 1])
torch.Size([1, 284, 1]) torch.Size([1, 1, 1])
torch.Size([2, 376, 1]) torch.Size([2, 1, 1])
torch.Size([1, 239, 1]) torch.Size([1, 1, 1])
torch.Size([3, 263, 1]) torch.Size([3, 1, 1])
torch.Size([8, 278, 1]) torch.Size([8, 1, 1])
torch.Size([1, 238, 1]) torch.Size([1, 1, 1])
torch.Size([5, 399, 1]) torch.Size([5, 1, 1])
```

```
torch.Size([1, 330, 1]) torch.Size([1, 1, 1])
torch.Size([1, 318, 1]) torch.Size([1, 1, 1])
torch.Size([1, 320, 1]) torch.Size([1, 1, 1])
torch.Size([5, 411, 1]) torch.Size([5, 1, 1])
torch.Size([5, 336, 1]) torch.Size([5, 1, 1])
torch.Size([22, 460, 1]) torch.Size([22, 1, 1])
torch.Size([6, 149, 1]) torch.Size([6, 1, 1])
torch.Size([5, 412, 1]) torch.Size([5, 1, 1])
torch.Size([4, 338, 1]) torch.Size([4, 1, 1])
torch.Size([1, 250, 1]) torch.Size([1, 1, 1])
torch.Size([1, 78, 1]) torch.Size([1, 1, 1])
torch.Size([7, 277, 1]) torch.Size([7, 1, 1])
torch.Size([3, 323, 1]) torch.Size([3, 1, 1])
torch.Size([1, 371, 1]) torch.Size([1, 1, 1])
torch.Size([13, 462, 1]) torch.Size([13, 1, 1])
torch.Size([2, 358, 1]) torch.Size([2, 1, 1])
torch.Size([7, 455, 1]) torch.Size([7, 1, 1])
torch.Size([6, 390, 1]) torch.Size([6, 1, 1])
torch.Size([5, 276, 1]) torch.Size([5, 1, 1])
torch.Size([4, 9, 1]) torch.Size([4, 1, 1])
torch.Size([1, 61, 1]) torch.Size([1, 1, 1])
torch.Size([3, 97, 1]) torch.Size([3, 1, 1])
torch.Size([1, 315, 1]) torch.Size([1, 1, 1])
torch.Size([3, 289, 1]) torch.Size([3, 1, 1])
torch.Size([6, 415, 1]) torch.Size([6, 1, 1])
torch.Size([8, 379, 1]) torch.Size([8, 1, 1])
torch.Size([2, 108, 1]) torch.Size([2, 1, 1])
torch.Size([1, 409, 1]) torch.Size([1, 1, 1])
torch.Size([4, 343, 1]) torch.Size([4, 1, 1])
torch.Size([29, 464, 1]) torch.Size([29, 1, 1])
torch.Size([1, 176, 1]) torch.Size([1, 1, 1])
torch.Size([9, 352, 1]) torch.Size([9, 1, 1])
torch.Size([7, 418, 1]) torch.Size([7, 1, 1])
torch.Size([3, 69, 1]) torch.Size([3, 1, 1])
torch.Size([4, 90, 1]) torch.Size([4, 1, 1])
torch.Size([1, 162, 1]) torch.Size([1, 1, 1])
torch.Size([8, 74, 1]) torch.Size([8, 1, 1])
torch.Size([5, 404, 1]) torch.Size([5, 1, 1])
torch.Size([3, 282, 1]) torch.Size([3, 1, 1])
torch.Size([2, 233, 1]) torch.Size([2, 1, 1])
torch.Size([4, 350, 1]) torch.Size([4, 1, 1])
torch.Size([5, 158, 1]) torch.Size([5, 1, 1])
torch.Size([2, 220, 1]) torch.Size([2, 1, 1])
torch.Size([7, 429, 1]) torch.Size([7, 1, 1])
torch.Size([2, 140, 1]) torch.Size([2, 1, 1])
torch.Size([9, 434, 1]) torch.Size([9, 1, 1])
torch.Size([1, 246, 1]) torch.Size([1, 1, 1])
torch.Size([10, 416, 1]) torch.Size([10, 1, 1])
torch.Size([3, 85, 1]) torch.Size([3, 1, 1])
torch.Size([10, 342, 1]) torch.Size([10, 1, 1])
torch.Size([3, 353, 1]) torch.Size([3, 1, 1])
torch.Size([2, 166, 1]) torch.Size([2, 1, 1])
torch.Size([4, 370, 1]) torch.Size([4, 1, 1])
torch.Size([1, 181, 1]) torch.Size([1, 1, 1])
torch.Size([9, 414, 1]) torch.Size([9, 1, 1])
torch.Size([1, 212, 1]) torch.Size([1, 1, 1])
torch.Size([6, 18, 1]) torch.Size([6, 1, 1])
torch.Size([14, 442, 1]) torch.Size([14, 1, 1])
torch.Size([2, 145, 1]) torch.Size([2, 1, 1])
torch.Size([2, 124, 1]) torch.Size([2, 1, 1])
torch.Size([2, 301, 1]) torch.Size([2, 1, 1])
torch.Size([1, 346, 1]) torch.Size([1, 1, 1])
torch.Size([1, 107, 1]) torch.Size([1, 1, 1])
torch.Size([3, 327, 1]) torch.Size([3, 1, 1])
torch.Size([2, 100, 1]) torch.Size([2, 1, 1])
torch.Size([3, 232, 1]) torch.Size([3, 1, 1])
torch.Size([14, 428, 1]) torch.Size([14, 1, 1])
torch.Size([4, 120, 1]) torch.Size([4, 1, 1])
torch.Size([6, 387, 1]) torch.Size([6, 1, 1])
torch.Size([1, 160, 1]) torch.Size([1, 1, 1])
torch.Size([1, 245, 1]) torch.Size([1, 1, 1])
torch.Size([5, 285, 1]) torch.Size([5, 1, 1])
torch.Size([2, 167, 1]) torch.Size([2, 1, 1])
torch.Size([9, 452, 1]) torch.Size([9, 1, 1])
torch.Size([6, 255, 1]) torch.Size([6, 1, 1])
torch.Size([4, 236, 1]) torch.Size([4, 1, 1])
torch.Size([6, 143, 1]) torch.Size([6, 1, 1])
torch.Size([4, 127, 1]) torch.Size([4, 1, 1])
torch.Size([10, 383, 1]) torch.Size([10, 1, 1])
torch.Size([17, 444, 1]) torch.Size([17, 1, 1])
torch.Size([1, 88, 1]) torch.Size([1, 1, 1])
torch.Size([4, 347, 1]) torch.Size([4, 1, 1])
torch.Size([3, 256, 1]) torch.Size([3, 1, 1])
```

```
torch.Size([6, 424, 1]) torch.Size([6, 1, 1])
torch.Size([5, 161, 1]) torch.Size([5, 1, 1])
torch.Size([4, 367, 1]) torch.Size([4, 1, 1])
torch.Size([3, 313, 1]) torch.Size([3, 1, 1])
torch.Size([6, 147, 1]) torch.Size([6, 1, 1])
torch.Size([1, 89, 1]) torch.Size([1, 1, 1])
torch.Size([2, 259, 1]) torch.Size([2, 1, 1])
torch.Size([8, 450, 1]) torch.Size([8, 1, 1])
torch.Size([10, 49, 1]) torch.Size([10, 1, 1])
torch.Size([8, 339, 1]) torch.Size([8, 1, 1])
torch.Size([4, 42, 1]) torch.Size([4, 1, 1])
torch.Size([9, 459, 1]) torch.Size([9, 1, 1])
torch.Size([7, 137, 1]) torch.Size([7, 1, 1])
torch.Size([3, 298, 1]) torch.Size([3, 1, 1])
torch.Size([1, 377, 1]) torch.Size([1, 1, 1])
torch.Size([1, 362, 1]) torch.Size([1, 1, 1])
torch.Size([5, 52, 1]) torch.Size([5, 1, 1])
torch.Size([6, 218, 1]) torch.Size([6, 1, 1])
torch.Size([1, 226, 1]) torch.Size([1, 1, 1])
torch.Size([1, 324, 1]) torch.Size([1, 1, 1])
torch.Size([2, 102, 1]) torch.Size([2, 1, 1])
torch.Size([1, 155, 1]) torch.Size([1, 1, 1])
torch.Size([9, 425, 1]) torch.Size([9, 1, 1])
torch.Size([3, 288, 1]) torch.Size([3, 1, 1])
torch.Size([1, 53, 1]) torch.Size([1, 1, 1])
torch.Size([5, 146, 1]) torch.Size([5, 1, 1])
torch.Size([24, 10, 1]) torch.Size([24, 1, 1])
torch.Size([3, 253, 1]) torch.Size([3, 1, 1])
torch.Size([5, 230, 1]) torch.Size([5, 1, 1])
torch.Size([1, 305, 1]) torch.Size([1, 1, 1])
torch.Size([2, 386, 1]) torch.Size([2, 1, 1])
torch.Size([71, 468, 1]) torch.Size([71, 1, 1])
torch.Size([2, 51, 1]) torch.Size([2, 1, 1])
torch.Size([4, 185, 1]) torch.Size([4, 1, 1])
torch.Size([1, 295, 1]) torch.Size([1, 1, 1])
torch.Size([4, 8, 1]) torch.Size([4, 1, 1])
torch.Size([12, 439, 1]) torch.Size([12, 1, 1])
torch.Size([6, 398, 1]) torch.Size([6, 1, 1])
torch.Size([8, 271, 1]) torch.Size([8, 1, 1])
torch.Size([12, 3, 1]) torch.Size([12, 1, 1])
torch.Size([6, 148, 1]) torch.Size([6, 1, 1])
torch.Size([1, 119, 1]) torch.Size([1, 1, 1])
torch.Size([3, 130, 1]) torch.Size([3, 1, 1])
torch.Size([3, 287, 1]) torch.Size([3, 1, 1])
torch.Size([9, 423, 1]) torch.Size([9, 1, 1])
torch.Size([49, 467, 1]) torch.Size([49, 1, 1])
torch.Size([2, 168, 1]) torch.Size([2, 1, 1])
torch.Size([6, 5, 1]) torch.Size([6, 1, 1])
torch.Size([10, 4, 1]) torch.Size([10, 1, 1])
torch.Size([2, 54, 1]) torch.Size([2, 1, 1])
torch.Size([3, 205, 1]) torch.Size([3, 1, 1])
torch.Size([11, 357, 1]) torch.Size([11, 1, 1])
torch.Size([9, 446, 1]) torch.Size([9, 1, 1])
torch.Size([3, 384, 1]) torch.Size([3, 1, 1])
torch.Size([1, 64, 1]) torch.Size([1, 1, 1])
torch.Size([2, 244, 1]) torch.Size([2, 1, 1])
torch.Size([2, 123, 1]) torch.Size([2, 1, 1])
torch.Size([5, 133, 1]) torch.Size([5, 1, 1])
torch.Size([7, 25, 1]) torch.Size([7, 1, 1])
torch.Size([4, 300, 1]) torch.Size([4, 1, 1])
torch.Size([1, 174, 1]) torch.Size([1, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([11, 454, 1]) torch.Size([11, 1, 1])
torch.Size([2, 150, 1]) torch.Size([2, 1, 1])
torch.Size([4, 267, 1]) torch.Size([4, 1, 1])
torch.Size([7, 80, 1]) torch.Size([7, 1, 1])
torch.Size([2, 184, 1]) torch.Size([2, 1, 1])
torch.Size([4, 273, 1]) torch.Size([4, 1, 1])
torch.Size([5, 363, 1]) torch.Size([5, 1, 1])
torch.Size([2, 144, 1]) torch.Size([2, 1, 1])
torch.Size([5, 437, 1]) torch.Size([5, 1, 1])
torch.Size([3, 73, 1]) torch.Size([3, 1, 1])
torch.Size([8, 365, 1]) torch.Size([8, 1, 1])
torch.Size([2, 76, 1]) torch.Size([2, 1, 1])
torch.Size([1, 309, 1]) torch.Size([1, 1, 1])
torch.Size([3, 227, 1]) torch.Size([3, 1, 1])
torch.Size([8, 361, 1]) torch.Size([8, 1, 1])
torch.Size([18, 447, 1]) torch.Size([18, 1, 1])
torch.Size([11, 369, 1]) torch.Size([11, 1, 1])
torch.Size([6, 281, 1]) torch.Size([6, 1, 1])
torch.Size([5, 237, 1]) torch.Size([5, 1, 1])
torch.Size([7, 421, 1]) torch.Size([7, 1, 1])
torch.Size([2, 303, 1]) torch.Size([2, 1, 1])
```

```
torch.Size([3, 177, 1]) torch.Size([3, 1, 1])
torch.Size([2, 82, 1]) torch.Size([2, 1, 1])
torch.Size([5, 60, 1]) torch.Size([5, 1, 1])
torch.Size([7, 163, 1]) torch.Size([7, 1, 1])
torch.Size([4, 328, 1]) torch.Size([4, 1, 1])
torch.Size([3, 186, 1]) torch.Size([3, 1, 1])
torch.Size([2, 241, 1]) torch.Size([2, 1, 1])
torch.Size([3, 70, 1]) torch.Size([3, 1, 1])
torch.Size([3, 378, 1]) torch.Size([3, 1, 1])
torch.Size([4, 274, 1]) torch.Size([4, 1, 1])
torch.Size([5, 105, 1]) torch.Size([5, 1, 1])
torch.Size([5, 254, 1]) torch.Size([5, 1, 1])
torch.Size([7, 395, 1]) torch.Size([7, 1, 1])
torch.Size([2, 113, 1]) torch.Size([2, 1, 1])
torch.Size([4, 195, 1]) torch.Size([4, 1, 1])
torch.Size([3, 283, 1]) torch.Size([3, 1, 1])
torch.Size([2, 335, 1]) torch.Size([2, 1, 1])
torch.Size([5, 63, 1]) torch.Size([5, 1, 1])
torch.Size([5, 182, 1]) torch.Size([5, 1, 1])
torch.Size([2, 206, 1]) torch.Size([2, 1, 1])
torch.Size([6, 373, 1]) torch.Size([6, 1, 1])
torch.Size([2, 375, 1]) torch.Size([2, 1, 1])
torch.Size([4, 321, 1]) torch.Size([4, 1, 1])
torch.Size([4, 268, 1]) torch.Size([4, 1, 1])
torch.Size([2, 310, 1]) torch.Size([2, 1, 1])
torch.Size([3, 224, 1]) torch.Size([3, 1, 1])
torch.Size([3, 152, 1]) torch.Size([3, 1, 1])
torch.Size([6, 221, 1]) torch.Size([6, 1, 1])
torch.Size([3, 83, 1]) torch.Size([3, 1, 1])
torch.Size([1, 403, 1]) torch.Size([1, 1, 1])
torch.Size([1, 106, 1]) torch.Size([1, 1, 1])
torch.Size([3, 214, 1]) torch.Size([3, 1, 1])
torch.Size([5, 312, 1]) torch.Size([5, 1, 1])
torch.Size([7, 19, 1]) torch.Size([7, 1, 1])
torch.Size([1, 192, 1]) torch.Size([1, 1, 1])
torch.Size([4, 272, 1]) torch.Size([4, 1, 1])
torch.Size([8, 200, 1]) torch.Size([8, 1, 1])
torch.Size([8, 441, 1]) torch.Size([8, 1, 1])
torch.Size([2, 112, 1]) torch.Size([2, 1, 1])
torch.Size([5, 340, 1]) torch.Size([5, 1, 1])
torch.Size([2, 24, 1]) torch.Size([2, 1, 1])
torch.Size([18, 461, 1]) torch.Size([18, 1, 1])
torch.Size([4, 201, 1]) torch.Size([4, 1, 1])
torch.Size([1, 248, 1]) torch.Size([1, 1, 1])
torch.Size([9, 231, 1]) torch.Size([9, 1, 1])
torch.Size([4, 360, 1]) torch.Size([4, 1, 1])
torch.Size([1, 394, 1]) torch.Size([1, 1, 1])
torch.Size([9, 187, 1]) torch.Size([9, 1, 1])
torch.Size([11, 31, 1]) torch.Size([11, 1, 1])
torch.Size([6, 344, 1]) torch.Size([6, 1, 1])
torch.Size([4, 65, 1]) torch.Size([4, 1, 1])
torch.Size([2, 136, 1]) torch.Size([2, 1, 1])
torch.Size([6, 38, 1]) torch.Size([6, 1, 1])
torch.Size([8, 290, 1]) torch.Size([8, 1, 1])
torch.Size([3, 389, 1]) torch.Size([3, 1, 1])
torch.Size([1, 219, 1]) torch.Size([1, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([3, 210, 1]) torch.Size([3, 1, 1])
torch.Size([5, 401, 1]) torch.Size([5, 1, 1])
torch.Size([7, 43, 1]) torch.Size([7, 1, 1])
torch.Size([1, 172, 1]) torch.Size([1, 1, 1])
torch.Size([2, 194, 1]) torch.Size([2, 1, 1])
torch.Size([9, 430, 1]) torch.Size([9, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([3, 297, 1]) torch.Size([3, 1, 1])
torch.Size([3, 293, 1]) torch.Size([3, 1, 1])
torch.Size([1, 91, 1]) torch.Size([1, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([10, 17, 1]) torch.Size([10, 1, 1])
torch.Size([2, 159, 1]) torch.Size([2, 1, 1])
torch.Size([1, 175, 1]) torch.Size([1, 1, 1])
torch.Size([17, 448, 1]) torch.Size([17, 1, 1])
torch.Size([1, 134, 1]) torch.Size([1, 1, 1])
torch.Size([3, 211, 1]) torch.Size([3, 1, 1])
torch.Size([2, 306, 1]) torch.Size([2, 1, 1])
torch.Size([6, 1, 1]) torch.Size([6, 1, 1])
torch.Size([6, 213, 1]) torch.Size([6, 1, 1])
torch.Size([2, 142, 1]) torch.Size([2, 1, 1])
torch.Size([5, 216, 1]) torch.Size([5, 1, 1])
torch.Size([9, 368, 1]) torch.Size([9, 1, 1])
torch.Size([4, 391, 1]) torch.Size([4, 1, 1])
torch.Size([4, 235, 1]) torch.Size([4, 1, 1])
torch.Size([3, 400, 1]) torch.Size([3, 1, 1])
```

```
torch.Size([4, 334, 1]) torch.Size([4, 1, 1])
torch.Size([2, 164, 1]) torch.Size([2, 1, 1])
torch.Size([1, 86, 1]) torch.Size([1, 1, 1])
torch.Size([5, 13, 1]) torch.Size([5, 1, 1])
torch.Size([2, 101, 1]) torch.Size([2, 1, 1])
torch.Size([6, 410, 1]) torch.Size([6, 1, 1])
torch.Size([1, 67, 1]) torch.Size([1, 1, 1])
torch.Size([1, 50, 1]) torch.Size([1, 1, 1])
torch.Size([3, 372, 1]) torch.Size([3, 1, 1])
torch.Size([4, 249, 1]) torch.Size([4, 1, 1])
torch.Size([3, 28, 1]) torch.Size([3, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([14, 7, 1]) torch.Size([14, 1, 1])
torch.Size([6, 99, 1]) torch.Size([6, 1, 1])
torch.Size([3, 419, 1]) torch.Size([3, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([1, 331, 1]) torch.Size([1, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([1, 116, 1]) torch.Size([1, 1, 1])
torch.Size([5, 87, 1]) torch.Size([5, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([128, 469, 1]) torch.Size([128, 1, 1])
torch.Size([39, 469, 1]) torch.Size([39, 1, 1])
```

Model

We have implemented an RNN utilizing GRU units that consists of 1 layer

In [ ]:

```
class StockPredictGRU(nn.Module):
  def __init__(self,
               input_size=1,
               hidden_size=128,
               num_layers=1,
               output_size=1):
    super(StockPredictGRU, self).__init__()
    self.hidden_size = hidden_size
    # batch_first = True so that the layer knows that the input format is
    # [batch_size, seq_len, repr_dim] -- in general case
    # [n = ?, l = ?, in = 1] -- in our case
    # Note: nn.GRU returns two values: the output and final hidden state
    self.gru = nn.GRU(input_size,
                      hidden_size,
                      num_layers,
                      batch_first = True)
    self.fc = nn.Linear(hidden_size,
                        output_size)

  def forward(self, x, hidden = None):
    # The initial hidden state is set to all zeros by default
    # every time-step after should have an input for hidden to make future pred.
    output, hidden = self.gru(x, hidden) # forward propogate the GRU
    # Pass the output of the last time step to the FC layer
    output = self.fc(output[:, -1, :])
    return output, hidden
```

Training Code:

Decisions to be made about train() implementation:

- The plan is to train the model to predict the next value over, so teach it to predict the n+1 element (n being the sequence length)

Loss Function: **Mean Squared Loss**

- Since we have a regression problem the mean squared loss function might be the best choice.

Optimizer: **Adam**

- Effective and Efficient Optimizer

Checkpointing:

- Will be useful for cutting training time and building off of existing weights.

After each epoch the costs of both the training and validation sets will be recorded.

- Will also plot the training curve to get an overview of how the training was being conducted.

In [ ]:

```
def train_rnn_network(model, train, valid, num_epochs=100, learning_rate=0.01,
                      weight_decay = 0, checkpoint_path=None):
  criterion = nn.MSELoss()
  optimizer = torch.optim.Adam(model.parameters(), lr=learning_rate,
                               weight_decay=weight_decay)
  losses, train_cost, valid_cost = [], [], []
  epochs = []
  iterations = 0
  for epoch in range(num_epochs):
    for stock_data, labels in train:
      model.train()
      iterations += 1
      optimizer.zero_grad()
      pred, _ = model(stock_data)
      loss = criterion(pred, labels.reshape(pred.shape[0], pred.shape[1]))
      loss.backward()
      optimizer.step()

    losses.append(float(loss))
    epochs.append(epoch)
    train_cost.append(get_cost(model, train))
    valid_cost.append(get_cost(model, valid))
    print("Epoch %d; Iteration %d; Loss %f; Train Cost %f; Val Cost %f" % (
            epoch+1, iterations, loss, train_cost[-1], valid_cost[-1]))

    if checkpoint_path is not None:
      torch.save({'epoch': epoch,
                  'model_state_dict': model.state_dict(),
                  'optimizer_state_dict': optimizer.state_dict(),
                  'loss': train_cost[-1]
                        }, checkpoint_path.format(iterations))


  # plotting
  plt.title("Learning Curve")
  plt.plot(losses, label="Train")
  plt.xlabel("Epoch")
  plt.ylabel("Cost")
  plt.show()

  plt.title("Training Curve")
  plt.plot(epochs, train_cost, label="Train")
  plt.plot(epochs, valid_cost, label="Validation")
  plt.xlabel("Epoch")
  plt.ylabel("Cost")
  plt.legend(loc='best')
  plt.show()
```

Finding an Alternative to Accuracy:

Cost

- The fact that we have somewhat of a regression problem makes having an "accuracy function" pointless
- Alternative would be to have a cost function

In [ ]:

```
def get_cost(model, data):
```

```
  loss_function = nn.MSELoss()
  loss = 0
  total = 0

  for stock_data, labels in data:
    output, _ = model(stock_data)
    # Calculate the MSE and add it to total loss
    loss += loss_function(output,
                          labels.reshape(output.shape[0], output.shape[1])).item()
    # Divide by number of stock_data points
    total += stock_data.shape[0]

  return loss/total
```

Performing the Training (including overfitting on small dataset)

In [ ]:

```
# Initialize Model
model = StockPredictGRU()

# Perform a split on the small_dataset as well
sm_train, sm_valid, sm_test = divide_data(smalldataset)

sm_train_loader = StockBatcher(sm_train, batch_size= 1, drop_last=False)
sm_valid_loader = StockBatcher(sm_valid, batch_size= 1, drop_last=False)

# Overfit on small dataset
train_rnn_network(model, sm_train_loader, sm_valid_loader, num_epochs=500)
```

```
Epoch 1; Loss 1583.770020; Train Loss 16747.753586; Val Loss 65.897919
Epoch 2; Loss 546.688843; Train Loss 14079.870663; Val Loss 84.519079
Epoch 3; Loss 306.951233; Train Loss 11546.965611; Val Loss 229.634690
Epoch 4; Loss 141.609985; Train Loss 9499.199552; Val Loss 490.045052
Epoch 5; Loss 2926.108154; Train Loss 8085.053240; Val Loss 808.703293
Epoch 6; Loss 4348.474121; Train Loss 7086.119765; Val Loss 1152.996002
Epoch 7; Loss 3842.825928; Train Loss 6341.975755; Val Loss 1528.851379
Epoch 8; Loss 403.531860; Train Loss 5776.699789; Val Loss 1940.860291
Epoch 9; Loss 67.550674; Train Loss 5441.135963; Val Loss 2254.784485
Epoch 10; Loss 2846.758301; Train Loss 5263.677499; Val Loss 2321.613464
Epoch 11; Loss 1336.581299; Train Loss 5055.189212; Val Loss 2381.367004
Epoch 12; Loss 1.191421; Train Loss 4909.451384; Val Loss 2510.991516
Epoch 13; Loss 1.680599; Train Loss 4863.114808; Val Loss 2572.789612
Epoch 14; Loss 1093.295410; Train Loss 4784.187931; Val Loss 2602.795471
Epoch 15; Loss 1069.479126; Train Loss 4719.114758; Val Loss 2593.888611
Epoch 16; Loss 551.379700; Train Loss 4802.772859; Val Loss 2678.244873
Epoch 17; Loss 147.959961; Train Loss 4704.387447; Val Loss 2648.959839
Epoch 18; Loss 5.370422; Train Loss 4781.603582; Val Loss 2654.689087
Epoch 19; Loss 718.365845; Train Loss 4703.271982; Val Loss 2685.996704
Epoch 20; Loss 1108.350708; Train Loss 4583.240768; Val Loss 2521.364990
Epoch 21; Loss 646.975891; Train Loss 4406.725228; Val Loss 2431.032471
Epoch 22; Loss 2378.528564; Train Loss 4484.734989; Val Loss 2116.160645
Epoch 23; Loss 70.807808; Train Loss 4230.158002; Val Loss 642.215027
Epoch 24; Loss 73.343849; Train Loss 3517.529333; Val Loss 131.717595
Epoch 25; Loss 950.575073; Train Loss 3446.270469; Val Loss 383.061874
Epoch 26; Loss 2208.843750; Train Loss 3223.589895; Val Loss 619.142914
Epoch 27; Loss 250.896362; Train Loss 2942.374065; Val Loss 607.119354
Epoch 28; Loss 29.581385; Train Loss 2524.386472; Val Loss 498.913498
Epoch 29; Loss 690.436462; Train Loss 2192.759627; Val Loss 490.166595
Epoch 30; Loss 96.289795; Train Loss 1874.192510; Val Loss 492.905807
Epoch 31; Loss 18.056826; Train Loss 1728.558427; Val Loss 404.453506
Epoch 32; Loss 0.284778; Train Loss 1504.040848; Val Loss 435.192719
Epoch 33; Loss 0.096674; Train Loss 1169.759456; Val Loss 394.134308
Epoch 34; Loss 220.963623; Train Loss 1225.500701; Val Loss 359.591583
Epoch 35; Loss 817.601685; Train Loss 1055.517616; Val Loss 351.941330
Epoch 36; Loss 63.707245; Train Loss 912.849851; Val Loss 326.053558
Epoch 37; Loss 5.905782; Train Loss 774.318612; Val Loss 212.457413
Epoch 38; Loss 73.394035; Train Loss 798.739529; Val Loss 181.560635
Epoch 39; Loss 671.212036; Train Loss 907.471767; Val Loss 114.077671
Epoch 40; Loss 2.221543; Train Loss 644.295226; Val Loss 257.899521
Epoch 41; Loss 52.617023; Train Loss 608.236300; Val Loss 309.556984
Epoch 42; Loss 639.155090; Train Loss 617.625743; Val Loss 189.652893
Epoch 43; Loss 1.610877; Train Loss 491.227388; Val Loss 179.819000
Epoch 44; Loss 1.650235; Train Loss 384.885455; Val Loss 237.021759
Epoch 45; Loss 325.703186; Train Loss 307.752644; Val Loss 201.456749
Epoch 46; Loss 10.464452; Train Loss 266.482521; Val Loss 221.941872
Epoch 47; Loss 10.463675; Train Loss 217.561824; Val Loss 202.753281
Epoch 48; Loss 0.627107; Train Loss 193.168275; Val Loss 170.008862
Epoch 49; Loss 9.026020; Train Loss 174.259486; Val Loss 189.010597
Epoch 50; Loss 153.206161; Train Loss 162.377668; Val Loss 196.672699
Epoch 51; Loss 4.329995; Train Loss 164.271170; Val Loss 182.306107
Epoch 52; Loss 11.372067; Train Loss 165.926163; Val Loss 254.379494
Epoch 53; Loss 6.871692; Train Loss 183.682434; Val Loss 145.431908
```

```
Epoch 54; Loss 41.118622; Train Loss 201.309461; Val Loss 196.995857
Epoch 55; Loss 8.587987; Train Loss 147.246300; Val Loss 107.696945
Epoch 56; Loss 4.736116; Train Loss 145.495521; Val Loss 161.548687
Epoch 57; Loss 4.988102; Train Loss 278.800424; Val Loss 126.976677
Epoch 58; Loss 0.028594; Train Loss 186.144304; Val Loss 139.494061
Epoch 59; Loss 0.796437; Train Loss 153.751869; Val Loss 197.393028
Epoch 60; Loss 0.373223; Train Loss 141.000852; Val Loss 222.150047
Epoch 61; Loss 12.592990; Train Loss 126.825568; Val Loss 232.644995
Epoch 62; Loss 115.483780; Train Loss 117.361018; Val Loss 142.194639
Epoch 63; Loss 0.692314; Train Loss 103.951247; Val Loss 143.867842
Epoch 64; Loss 0.786209; Train Loss 103.134193; Val Loss 80.193714
Epoch 65; Loss 87.515694; Train Loss 218.513397; Val Loss 162.579937
Epoch 66; Loss 57.328491; Train Loss 158.378516; Val Loss 414.068703
Epoch 67; Loss 4.478623; Train Loss 169.077734; Val Loss 382.245697
Epoch 68; Loss 8.967003; Train Loss 98.692788; Val Loss 267.550163
Epoch 69; Loss 19.641197; Train Loss 85.995186; Val Loss 245.528419
Epoch 70; Loss 44.689091; Train Loss 70.126822; Val Loss 226.144104
Epoch 71; Loss 0.903132; Train Loss 65.990988; Val Loss 228.566036
Epoch 72; Loss 0.687362; Train Loss 45.863952; Val Loss 160.106476
Epoch 73; Loss 1.474997; Train Loss 40.623438; Val Loss 124.979767
Epoch 74; Loss 0.095035; Train Loss 35.508709; Val Loss 102.831367
Epoch 75; Loss 7.610159; Train Loss 35.496770; Val Loss 107.035355
Epoch 76; Loss 1.886704; Train Loss 31.714142; Val Loss 96.049692
Epoch 77; Loss 8.349142; Train Loss 36.321427; Val Loss 74.534246
Epoch 78; Loss 0.748182; Train Loss 32.660199; Val Loss 109.972055
Epoch 79; Loss 2.639187; Train Loss 30.558027; Val Loss 95.454855
Epoch 80; Loss 0.348839; Train Loss 28.678533; Val Loss 85.132353
Epoch 81; Loss 17.639910; Train Loss 28.402742; Val Loss 89.252530
Epoch 82; Loss 0.367161; Train Loss 25.168256; Val Loss 75.199348
Epoch 83; Loss 10.326932; Train Loss 24.752655; Val Loss 78.922791
Epoch 84; Loss 0.061138; Train Loss 23.625010; Val Loss 72.524014
Epoch 85; Loss 0.019608; Train Loss 18.820360; Val Loss 60.708210
Epoch 86; Loss 0.021757; Train Loss 14.300150; Val Loss 57.251839
Epoch 87; Loss 13.374748; Train Loss 14.041810; Val Loss 61.449065
Epoch 88; Loss 0.093468; Train Loss 12.203693; Val Loss 63.215636
Epoch 89; Loss 0.004155; Train Loss 11.441821; Val Loss 60.716371
Epoch 90; Loss 0.135829; Train Loss 10.611131; Val Loss 58.874923
Epoch 91; Loss 0.067292; Train Loss 9.656286; Val Loss 53.321699
Epoch 92; Loss 0.016025; Train Loss 9.024698; Val Loss 50.655512
Epoch 93; Loss 0.048440; Train Loss 8.594166; Val Loss 49.604338
Epoch 94; Loss 1.892871; Train Loss 8.160427; Val Loss 48.462250
Epoch 95; Loss 0.037893; Train Loss 7.968777; Val Loss 46.094118
Epoch 96; Loss 2.351835; Train Loss 7.615024; Val Loss 52.398179
Epoch 97; Loss 0.000095; Train Loss 7.228705; Val Loss 48.449139
Epoch 98; Loss 0.001424; Train Loss 7.059693; Val Loss 49.207427
Epoch 99; Loss 2.298738; Train Loss 6.858034; Val Loss 46.069577
Epoch 100; Loss 0.005836; Train Loss 6.572813; Val Loss 48.016841
Epoch 101; Loss 5.477011; Train Loss 6.694145; Val Loss 46.247444
Epoch 102; Loss 0.032516; Train Loss 6.585808; Val Loss 49.090768
Epoch 103; Loss 0.071068; Train Loss 6.761508; Val Loss 54.386849
Epoch 104; Loss 0.073298; Train Loss 6.493685; Val Loss 45.037520
Epoch 105; Loss 4.273062; Train Loss 6.872469; Val Loss 48.907462
Epoch 106; Loss 2.721976; Train Loss 7.123438; Val Loss 44.166437
Epoch 107; Loss 3.170270; Train Loss 5.918637; Val Loss 51.122811
Epoch 108; Loss 0.167525; Train Loss 6.444107; Val Loss 54.678553
Epoch 109; Loss 0.076289; Train Loss 5.968568; Val Loss 44.950878
Epoch 110; Loss 1.268674; Train Loss 3.535417; Val Loss 45.189245
Epoch 111; Loss 3.190461; Train Loss 7.185401; Val Loss 49.593869
Epoch 112; Loss 0.098265; Train Loss 4.944368; Val Loss 45.572813
Epoch 113; Loss 0.023540; Train Loss 3.434257; Val Loss 51.752481
Epoch 114; Loss 1.857870; Train Loss 6.236327; Val Loss 83.835485
Epoch 115; Loss 1.973228; Train Loss 26.637120; Val Loss 53.227917
Epoch 116; Loss 18.922022; Train Loss 8.656323; Val Loss 94.889593
Epoch 117; Loss 14.872317; Train Loss 59.833855; Val Loss 148.173193
Epoch 118; Loss 76.775505; Train Loss 134.905342; Val Loss 34.986247
Epoch 119; Loss 0.342867; Train Loss 50.823166; Val Loss 149.536011
Epoch 120; Loss 18.104094; Train Loss 30.499736; Val Loss 165.928642
Epoch 121; Loss 14.432573; Train Loss 26.584835; Val Loss 185.517208
Epoch 122; Loss 1.966204; Train Loss 26.709893; Val Loss 122.027540
Epoch 123; Loss 2.544467; Train Loss 12.996562; Val Loss 137.591503
Epoch 124; Loss 3.598129; Train Loss 16.977981; Val Loss 165.009762
Epoch 125; Loss 1.911126; Train Loss 16.377196; Val Loss 96.280416
Epoch 126; Loss 4.663556; Train Loss 427.864848; Val Loss 40.954288
Epoch 127; Loss 148.414856; Train Loss 721.044117; Val Loss 166.120319
Epoch 128; Loss 76.619057; Train Loss 431.890625; Val Loss 233.216568
Epoch 129; Loss 14.634220; Train Loss 455.454722; Val Loss 197.232752
Epoch 130; Loss 337.220459; Train Loss 457.749270; Val Loss 295.402246
Epoch 131; Loss 441.921417; Train Loss 473.450862; Val Loss 214.633421
Epoch 132; Loss 71.828705; Train Loss 327.738402; Val Loss 263.423920
Epoch 133; Loss 342.708038; Train Loss 252.287308; Val Loss 168.533722
Epoch 134; Loss 7.304060; Train Loss 173.256500; Val Loss 245.333860
Epoch 135; Loss 11.196844; Train Loss 150.735962; Val Loss 257.923248
Epoch 136; Loss 10.148973; Train Loss 133.748168; Val Loss 177.027552
```

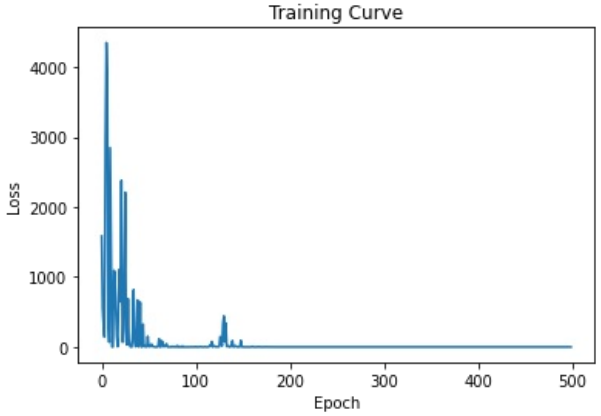
```
Epoch 137; Loss 0.033450; Train Loss 193.979182; Val Loss 124.188313
Epoch 138; Loss 16.366701; Train Loss 262.778633; Val Loss 149.930176
Epoch 139; Loss 7.206795; Train Loss 179.880063; Val Loss 194.471630
Epoch 140; Loss 89.982674; Train Loss 192.522601; Val Loss 203.087780
Epoch 141; Loss 2.627401; Train Loss 151.843245; Val Loss 310.896606
Epoch 142; Loss 1.715724; Train Loss 97.481275; Val Loss 237.905352
Epoch 143; Loss 17.363092; Train Loss 90.255736; Val Loss 207.011286
Epoch 144; Loss 8.520977; Train Loss 113.612247; Val Loss 214.606623
Epoch 145; Loss 1.569240; Train Loss 142.462733; Val Loss 148.579679
Epoch 146; Loss 0.259795; Train Loss 116.795011; Val Loss 150.947807
Epoch 147; Loss 5.958528; Train Loss 129.782411; Val Loss 162.182988
Epoch 148; Loss 2.017495; Train Loss 103.800537; Val Loss 149.643243
Epoch 149; Loss 92.620651; Train Loss 94.490816; Val Loss 155.017487
Epoch 150; Loss 4.364976; Train Loss 67.023046; Val Loss 178.137818
Epoch 151; Loss 0.000448; Train Loss 56.706568; Val Loss 166.076415
Epoch 152; Loss 0.138119; Train Loss 53.435088; Val Loss 156.302227
Epoch 153; Loss 0.997152; Train Loss 49.800710; Val Loss 159.741448
Epoch 154; Loss 0.217899; Train Loss 46.413238; Val Loss 155.310374
Epoch 155; Loss 0.791664; Train Loss 42.115608; Val Loss 144.635913
Epoch 156; Loss 0.018690; Train Loss 36.958567; Val Loss 146.610324
Epoch 157; Loss 0.179902; Train Loss 26.766718; Val Loss 120.880800
Epoch 158; Loss 1.170346; Train Loss 23.579109; Val Loss 124.469378
Epoch 159; Loss 0.098017; Train Loss 23.749879; Val Loss 104.637669
Epoch 160; Loss 0.011227; Train Loss 11.410342; Val Loss 96.358557
Epoch 161; Loss 7.084836; Train Loss 11.701898; Val Loss 129.395957
Epoch 162; Loss 3.555988; Train Loss 5.139454; Val Loss 135.247904
Epoch 163; Loss 0.052979; Train Loss 5.788517; Val Loss 120.310884
Epoch 164; Loss 0.014359; Train Loss 2.075373; Val Loss 118.138556
Epoch 165; Loss 0.241408; Train Loss 8.194846; Val Loss 123.142083
Epoch 166; Loss 2.610775; Train Loss 7.543205; Val Loss 117.204218
Epoch 167; Loss 0.001769; Train Loss 6.391572; Val Loss 107.377959
Epoch 168; Loss 3.897184; Train Loss 5.617012; Val Loss 104.083158
Epoch 169; Loss 1.448667; Train Loss 4.417032; Val Loss 96.662731
Epoch 170; Loss 0.227610; Train Loss 3.828934; Val Loss 80.773847
Epoch 171; Loss 0.038592; Train Loss 3.098107; Val Loss 80.306751
Epoch 172; Loss 0.632702; Train Loss 3.467449; Val Loss 56.038829
Epoch 173; Loss 0.307932; Train Loss 3.298523; Val Loss 85.770885
Epoch 174; Loss 0.002273; Train Loss 2.804017; Val Loss 74.420409
Epoch 175; Loss 0.007215; Train Loss 2.426613; Val Loss 77.647763
Epoch 176; Loss 1.273951; Train Loss 3.178314; Val Loss 88.259442
Epoch 177; Loss 1.723417; Train Loss 3.388601; Val Loss 74.235354
Epoch 178; Loss 0.131104; Train Loss 2.229274; Val Loss 79.656219
Epoch 179; Loss 1.158409; Train Loss 3.818588; Val Loss 87.062651
Epoch 180; Loss 0.723522; Train Loss 2.425089; Val Loss 73.786534
Epoch 181; Loss 0.279026; Train Loss 2.962655; Val Loss 71.270563
Epoch 182; Loss 0.102625; Train Loss 2.253158; Val Loss 82.509149
Epoch 183; Loss 0.012737; Train Loss 2.103289; Val Loss 81.313108
Epoch 184; Loss 0.079611; Train Loss 1.902986; Val Loss 72.924713
Epoch 185; Loss 0.162532; Train Loss 1.686367; Val Loss 73.194341
Epoch 186; Loss 0.000328; Train Loss 1.709037; Val Loss 75.065729
Epoch 187; Loss 0.819700; Train Loss 1.534181; Val Loss 69.849981
Epoch 188; Loss 0.034802; Train Loss 1.476696; Val Loss 68.679879
Epoch 189; Loss 0.075155; Train Loss 1.413178; Val Loss 67.650776
Epoch 190; Loss 0.000431; Train Loss 1.364814; Val Loss 69.406604
Epoch 191; Loss 0.702649; Train Loss 1.306700; Val Loss 67.263261
Epoch 192; Loss 0.685892; Train Loss 1.263077; Val Loss 67.349040
Epoch 193; Loss 0.021400; Train Loss 1.231871; Val Loss 67.037801
Epoch 194; Loss 0.667541; Train Loss 1.190011; Val Loss 67.639147
Epoch 195; Loss 0.009508; Train Loss 1.137699; Val Loss 67.019867
Epoch 196; Loss 0.649168; Train Loss 1.106500; Val Loss 68.052622
Epoch 197; Loss 0.044850; Train Loss 1.072509; Val Loss 69.566252
Epoch 198; Loss 0.021871; Train Loss 1.019292; Val Loss 66.873804
Epoch 199; Loss 0.006093; Train Loss 1.091820; Val Loss 67.607403
Epoch 200; Loss 0.479151; Train Loss 1.139716; Val Loss 63.629080
Epoch 201; Loss 0.020793; Train Loss 1.480264; Val Loss 71.512534
Epoch 202; Loss 0.377913; Train Loss 0.999089; Val Loss 71.600209
Epoch 203; Loss 0.067726; Train Loss 1.129908; Val Loss 67.373289
Epoch 204; Loss 0.490658; Train Loss 0.951739; Val Loss 71.866984
Epoch 205; Loss 0.042218; Train Loss 0.867283; Val Loss 69.979046
Epoch 206; Loss 0.462203; Train Loss 0.810635; Val Loss 68.497134
Epoch 207; Loss 0.546581; Train Loss 0.797872; Val Loss 66.507617
Epoch 208; Loss 0.048420; Train Loss 0.691642; Val Loss 67.680264
Epoch 209; Loss 0.004144; Train Loss 0.647086; Val Loss 67.448446
Epoch 210; Loss 0.005096; Train Loss 0.561033; Val Loss 64.714301
Epoch 211; Loss 0.217387; Train Loss 0.528027; Val Loss 67.566665
Epoch 212; Loss 0.008637; Train Loss 0.471218; Val Loss 62.966355
Epoch 213; Loss 0.002108; Train Loss 0.290686; Val Loss 65.176085
Epoch 214; Loss 0.182865; Train Loss 0.239357; Val Loss 64.476234
Epoch 215; Loss 0.043514; Train Loss 0.195470; Val Loss 65.285174
Epoch 216; Loss 0.096160; Train Loss 0.165815; Val Loss 63.950516
Epoch 217; Loss 0.180569; Train Loss 0.195766; Val Loss 67.759304
Epoch 218; Loss 0.001217; Train Loss 0.223503; Val Loss 61.449080
Epoch 219; Loss 0.010939; Train Loss 0.087366; Val Loss 65.198475
```

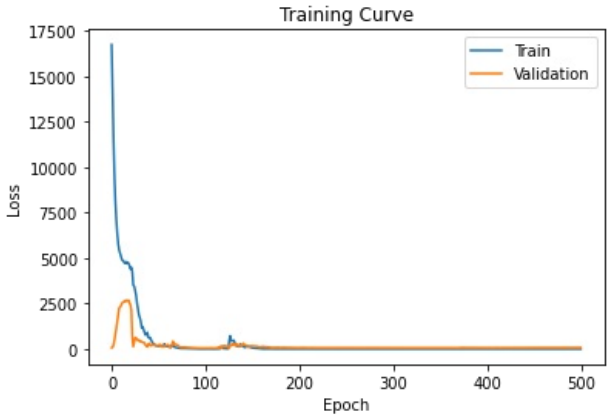
```
Epoch 220; Loss 0.003062; Train Loss 0.110915; Val Loss 65.841737
Epoch 221; Loss 0.002360; Train Loss 0.070142; Val Loss 64.034729
Epoch 222; Loss 0.014108; Train Loss 0.060652; Val Loss 65.307186
Epoch 223; Loss 0.010303; Train Loss 0.085217; Val Loss 62.882325
Epoch 224; Loss 0.006156; Train Loss 0.057486; Val Loss 65.479277
Epoch 225; Loss 0.004103; Train Loss 0.080327; Val Loss 60.649709
Epoch 226; Loss 0.009613; Train Loss 0.022180; Val Loss 63.436696
Epoch 227; Loss 0.003906; Train Loss 0.019603; Val Loss 62.714140
Epoch 228; Loss 0.000805; Train Loss 0.009805; Val Loss 63.060638
Epoch 229; Loss 0.000280; Train Loss 0.071543; Val Loss 59.865829
Epoch 230; Loss 0.001324; Train Loss 0.306763; Val Loss 72.077904
Epoch 231; Loss 0.002433; Train Loss 0.069043; Val Loss 67.555117
Epoch 232; Loss 0.009479; Train Loss 0.072732; Val Loss 65.630100
Epoch 233; Loss 0.050256; Train Loss 0.071237; Val Loss 68.616643
Epoch 234; Loss 0.000143; Train Loss 0.072807; Val Loss 64.443549
Epoch 235; Loss 0.003583; Train Loss 0.042015; Val Loss 67.380653
Epoch 236; Loss 0.000070; Train Loss 0.056480; Val Loss 68.607632
Epoch 237; Loss 0.022559; Train Loss 0.044231; Val Loss 66.656213
Epoch 238; Loss 0.003345; Train Loss 0.036500; Val Loss 66.027685
Epoch 239; Loss 0.000001; Train Loss 0.071878; Val Loss 68.429453
Epoch 240; Loss 0.000004; Train Loss 0.035979; Val Loss 66.105013
Epoch 241; Loss 0.000841; Train Loss 0.042581; Val Loss 65.366813
Epoch 242; Loss 0.000003; Train Loss 0.046019; Val Loss 66.808529
Epoch 243; Loss 0.021153; Train Loss 0.035455; Val Loss 65.128684
Epoch 244; Loss 0.000618; Train Loss 0.025251; Val Loss 65.749899
Epoch 245; Loss 0.025494; Train Loss 0.023340; Val Loss 65.354016
Epoch 246; Loss 0.003895; Train Loss 0.053233; Val Loss 62.809393
Epoch 247; Loss 0.003573; Train Loss 0.031928; Val Loss 65.550962
Epoch 248; Loss 0.001950; Train Loss 0.023489; Val Loss 64.906443
Epoch 249; Loss 0.006455; Train Loss 0.039938; Val Loss 63.021608
Epoch 250; Loss 0.001539; Train Loss 0.032579; Val Loss 65.299969
Epoch 251; Loss 0.000056; Train Loss 0.020805; Val Loss 64.210039
Epoch 252; Loss 0.000038; Train Loss 0.016221; Val Loss 63.606354
Epoch 253; Loss 0.000525; Train Loss 0.018247; Val Loss 63.775283
Epoch 254; Loss 0.000426; Train Loss 0.032256; Val Loss 63.608023
Epoch 255; Loss 0.006508; Train Loss 0.028621; Val Loss 64.698742
Epoch 256; Loss 0.023918; Train Loss 0.117314; Val Loss 59.980954
Epoch 257; Loss 0.063333; Train Loss 0.057675; Val Loss 62.903672
Epoch 258; Loss 0.034106; Train Loss 0.349131; Val Loss 70.573491
Epoch 259; Loss 0.012060; Train Loss 0.138001; Val Loss 62.690443
Epoch 260; Loss 0.032444; Train Loss 0.041680; Val Loss 64.990968
Epoch 261; Loss 0.000729; Train Loss 0.066034; Val Loss 66.993065
Epoch 262; Loss 0.000550; Train Loss 0.025789; Val Loss 63.910016
Epoch 263; Loss 0.009331; Train Loss 0.017287; Val Loss 64.499345
Epoch 264; Loss 0.025346; Train Loss 0.021332; Val Loss 63.111350
Epoch 265; Loss 0.004135; Train Loss 0.015816; Val Loss 64.388074
Epoch 266; Loss 0.004440; Train Loss 0.020792; Val Loss 64.521711
Epoch 267; Loss 0.010533; Train Loss 0.045041; Val Loss 61.740538
Epoch 268; Loss 0.000008; Train Loss 0.052208; Val Loss 65.100004
Epoch 269; Loss 0.050386; Train Loss 0.041216; Val Loss 64.178140
Epoch 270; Loss 0.001072; Train Loss 0.032372; Val Loss 61.937220
Epoch 271; Loss 0.001269; Train Loss 0.025159; Val Loss 63.448404
Epoch 272; Loss 0.000001; Train Loss 0.015200; Val Loss 62.758482
Epoch 273; Loss 0.000211; Train Loss 0.008159; Val Loss 62.275259
Epoch 274; Loss 0.026364; Train Loss 0.021244; Val Loss 60.825554
Epoch 275; Loss 0.001523; Train Loss 0.072608; Val Loss 64.793303
Epoch 276; Loss 0.000693; Train Loss 0.026695; Val Loss 65.695815
Epoch 277; Loss 0.000942; Train Loss 0.047259; Val Loss 62.698298
Epoch 278; Loss 0.000583; Train Loss 0.045489; Val Loss 64.969049
Epoch 279; Loss 0.000032; Train Loss 0.009914; Val Loss 64.305561
Epoch 280; Loss 0.000009; Train Loss 0.008573; Val Loss 63.886042
Epoch 281; Loss 0.004713; Train Loss 0.005441; Val Loss 64.140913
Epoch 282; Loss 0.003278; Train Loss 0.008271; Val Loss 64.549002
Epoch 283; Loss 0.000938; Train Loss 0.014329; Val Loss 62.818524
Epoch 284; Loss 0.000036; Train Loss 0.003243; Val Loss 64.130181
Epoch 285; Loss 0.000027; Train Loss 0.009495; Val Loss 64.895752
Epoch 286; Loss 0.002232; Train Loss 0.013919; Val Loss 62.890189
Epoch 287; Loss 0.000005; Train Loss 0.007775; Val Loss 64.447231
Epoch 288; Loss 0.000829; Train Loss 0.004196; Val Loss 63.848707
Epoch 289; Loss 0.008146; Train Loss 0.005636; Val Loss 63.188416
Epoch 290; Loss 0.000235; Train Loss 0.004446; Val Loss 63.935605
Epoch 291; Loss 0.000391; Train Loss 0.002375; Val Loss 63.456484
Epoch 292; Loss 0.000285; Train Loss 0.001639; Val Loss 63.421498
Epoch 293; Loss 0.003208; Train Loss 0.002053; Val Loss 63.205597
Epoch 294; Loss 0.000091; Train Loss 0.001745; Val Loss 63.606708
Epoch 295; Loss 0.000015; Train Loss 0.001052; Val Loss 63.241552
Epoch 296; Loss 0.002045; Train Loss 0.001307; Val Loss 63.036544
Epoch 297; Loss 0.000300; Train Loss 0.003304; Val Loss 63.791939
Epoch 298; Loss 0.000168; Train Loss 0.002131; Val Loss 62.995104
Epoch 299; Loss 0.000282; Train Loss 0.002784; Val Loss 63.140584
Epoch 300; Loss 0.000017; Train Loss 0.001044; Val Loss 63.004082
Epoch 301; Loss 0.000432; Train Loss 0.000708; Val Loss 63.116811
Epoch 302; Loss 0.000011; Train Loss 0.001904; Val Loss 62.510552
```

```
Epoch 303; Loss 0.000503; Train Loss 0.000509; Val Loss 62.950634
Epoch 304; Loss 0.000002; Train Loss 0.000377; Val Loss 63.040715
Epoch 305; Loss 0.000050; Train Loss 0.000615; Val Loss 62.636008
Epoch 306; Loss 0.000278; Train Loss 0.000319; Val Loss 62.809619
Epoch 307; Loss 0.000015; Train Loss 0.000293; Val Loss 62.931947
Epoch 308; Loss 0.000035; Train Loss 0.000145; Val Loss 62.810424
Epoch 309; Loss 0.000113; Train Loss 0.000219; Val Loss 62.756758
Epoch 310; Loss 0.000009; Train Loss 0.000293; Val Loss 62.793408
Epoch 311; Loss 0.000012; Train Loss 0.000155; Val Loss 62.713725
Epoch 312; Loss 0.000015; Train Loss 0.000160; Val Loss 62.613506
Epoch 313; Loss 0.000027; Train Loss 0.000547; Val Loss 62.946267
Epoch 314; Loss 0.000008; Train Loss 0.000064; Val Loss 62.665923
Epoch 315; Loss 0.000088; Train Loss 0.000246; Val Loss 62.531044
Epoch 316; Loss 0.000071; Train Loss 0.000172; Val Loss 62.770950
Epoch 317; Loss 0.000001; Train Loss 0.000267; Val Loss 62.458534
Epoch 318; Loss 0.000106; Train Loss 0.000096; Val Loss 62.708676
Epoch 319; Loss 0.000068; Train Loss 0.000092; Val Loss 62.685548
Epoch 320; Loss 0.000019; Train Loss 0.000111; Val Loss 62.579886
Epoch 321; Loss 0.000003; Train Loss 0.000033; Val Loss 62.606969
Epoch 322; Loss 0.000008; Train Loss 0.000022; Val Loss 62.603344
Epoch 323; Loss 0.000019; Train Loss 0.000021; Val Loss 62.563794
Epoch 324; Loss 0.000000; Train Loss 0.000115; Val Loss 62.723979
Epoch 325; Loss 0.000004; Train Loss 0.000007; Val Loss 62.572249
Epoch 326; Loss 0.000001; Train Loss 0.000025; Val Loss 62.565159
Epoch 327; Loss 0.000001; Train Loss 0.000010; Val Loss 62.563072
Epoch 328; Loss 0.000000; Train Loss 0.000004; Val Loss 62.591958
Epoch 329; Loss 0.000002; Train Loss 0.000082; Val Loss 62.575287
Epoch 330; Loss 0.000015; Train Loss 0.000066; Val Loss 62.519595
Epoch 331; Loss 0.000000; Train Loss 0.000047; Val Loss 62.636726
Epoch 332; Loss 0.000000; Train Loss 0.000067; Val Loss 62.482232
Epoch 333; Loss 0.000077; Train Loss 0.000018; Val Loss 62.521710
Epoch 334; Loss 0.000025; Train Loss 0.000424; Val Loss 62.793460
Epoch 335; Loss 0.000008; Train Loss 0.000438; Val Loss 62.471255
Epoch 336; Loss 0.000041; Train Loss 0.000070; Val Loss 62.614391
Epoch 337; Loss 0.000070; Train Loss 0.000124; Val Loss 62.622591
Epoch 338; Loss 0.000010; Train Loss 0.000266; Val Loss 62.392913
Epoch 339; Loss 0.000023; Train Loss 0.000255; Val Loss 62.552456
Epoch 340; Loss 0.000002; Train Loss 0.000109; Val Loss 62.548657
Epoch 341; Loss 0.000001; Train Loss 0.000016; Val Loss 62.597504
Epoch 342; Loss 0.000021; Train Loss 0.000036; Val Loss 62.482263
Epoch 343; Loss 0.000001; Train Loss 0.000071; Val Loss 62.619113
Epoch 344; Loss 0.000003; Train Loss 0.000046; Val Loss 62.615214
Epoch 345; Loss 0.000054; Train Loss 0.000041; Val Loss 62.488722
Epoch 346; Loss 0.000012; Train Loss 0.000019; Val Loss 62.587366
Epoch 347; Loss 0.000000; Train Loss 0.000045; Val Loss 62.476326
Epoch 348; Loss 0.000008; Train Loss 0.000151; Val Loss 62.620670
Epoch 349; Loss 0.000014; Train Loss 0.000120; Val Loss 62.498054
Epoch 350; Loss 0.000000; Train Loss 0.000191; Val Loss 62.693918
Epoch 351; Loss 0.000003; Train Loss 0.000142; Val Loss 62.497629
Epoch 352; Loss 0.000023; Train Loss 0.000070; Val Loss 62.681115
Epoch 353; Loss 0.000036; Train Loss 0.000151; Val Loss 62.595337
Epoch 354; Loss 0.000195; Train Loss 0.000189; Val Loss 62.451180
Epoch 355; Loss 0.000000; Train Loss 0.000237; Val Loss 62.725561
Epoch 356; Loss 0.000013; Train Loss 0.000104; Val Loss 62.576829
Epoch 357; Loss 0.000296; Train Loss 0.000117; Val Loss 62.657861
Epoch 358; Loss 0.000509; Train Loss 0.000217; Val Loss 62.696750
Epoch 359; Loss 0.000003; Train Loss 0.000348; Val Loss 62.586109
Epoch 360; Loss 0.000001; Train Loss 0.000023; Val Loss 62.561355
Epoch 361; Loss 0.000002; Train Loss 0.002007; Val Loss 62.649626
Epoch 362; Loss 0.000082; Train Loss 0.003458; Val Loss 62.678327
Epoch 363; Loss 0.000835; Train Loss 0.004797; Val Loss 62.174436
Epoch 364; Loss 0.017607; Train Loss 0.007039; Val Loss 63.210456
Epoch 365; Loss 0.000366; Train Loss 0.006656; Val Loss 62.029656
Epoch 366; Loss 0.001352; Train Loss 0.009244; Val Loss 63.006840
Epoch 367; Loss 0.022086; Train Loss 0.003376; Val Loss 62.573910
Epoch 368; Loss 0.059337; Train Loss 0.026826; Val Loss 61.903292
Epoch 369; Loss 0.006715; Train Loss 0.178610; Val Loss 63.269577
Epoch 370; Loss 0.027079; Train Loss 0.274597; Val Loss 64.930929
Epoch 371; Loss 0.000797; Train Loss 0.508613; Val Loss 58.906329
Epoch 372; Loss 0.061770; Train Loss 0.337071; Val Loss 71.315804
Epoch 373; Loss 0.214549; Train Loss 0.214007; Val Loss 71.029569
Epoch 374; Loss 0.002698; Train Loss 0.088955; Val Loss 64.772833
Epoch 375; Loss 0.001633; Train Loss 0.032905; Val Loss 67.309495
Epoch 376; Loss 0.146919; Train Loss 0.052535; Val Loss 67.373046
Epoch 377; Loss 0.000590; Train Loss 0.054295; Val Loss 66.102203
Epoch 378; Loss 0.055236; Train Loss 0.022268; Val Loss 66.292058
Epoch 379; Loss 0.000417; Train Loss 0.027303; Val Loss 67.417091
Epoch 380; Loss 0.000465; Train Loss 0.026358; Val Loss 65.554249
Epoch 381; Loss 0.070378; Train Loss 0.026585; Val Loss 65.389720
Epoch 382; Loss 0.003221; Train Loss 0.062630; Val Loss 65.743214
Epoch 383; Loss 0.009209; Train Loss 0.098160; Val Loss 66.334217
Epoch 384; Loss 0.000682; Train Loss 0.142899; Val Loss 62.658529
Epoch 385; Loss 0.001971; Train Loss 0.039712; Val Loss 67.856515
```

```
Epoch 386; Loss 0.027159; Train Loss 0.010970; Val Loss 67.216508
Epoch 387; Loss 0.009529; Train Loss 0.021433; Val Loss 65.319583
Epoch 388; Loss 0.005905; Train Loss 0.032700; Val Loss 68.146597
Epoch 389; Loss 0.008251; Train Loss 0.020620; Val Loss 66.882424
Epoch 390; Loss 0.008379; Train Loss 0.013653; Val Loss 65.819273
Epoch 391; Loss 0.003268; Train Loss 0.006399; Val Loss 66.617227
Epoch 392; Loss 0.001540; Train Loss 0.009108; Val Loss 65.320692
Epoch 393; Loss 0.001968; Train Loss 0.007557; Val Loss 66.351328
Epoch 394; Loss 0.005226; Train Loss 0.019069; Val Loss 64.097225
Epoch 395; Loss 0.000066; Train Loss 0.003883; Val Loss 65.625129
Epoch 396; Loss 0.000066; Train Loss 0.000254; Val Loss 65.330523
Epoch 397; Loss 0.001064; Train Loss 0.004194; Val Loss 64.882697
Epoch 398; Loss 0.000307; Train Loss 0.003577; Val Loss 65.900205
Epoch 399; Loss 0.000016; Train Loss 0.001371; Val Loss 65.117656
Epoch 400; Loss 0.000751; Train Loss 0.000633; Val Loss 65.232298
Epoch 401; Loss 0.000581; Train Loss 0.000532; Val Loss 65.494551
Epoch 402; Loss 0.000754; Train Loss 0.001326; Val Loss 65.050219
Epoch 403; Loss 0.000951; Train Loss 0.002175; Val Loss 65.944108
Epoch 404; Loss 0.000165; Train Loss 0.012100; Val Loss 65.403310
Epoch 405; Loss 0.000413; Train Loss 0.028225; Val Loss 65.712326
Epoch 406; Loss 0.074835; Train Loss 0.035991; Val Loss 64.288295
Epoch 407; Loss 0.003030; Train Loss 0.046357; Val Loss 67.544494
Epoch 408; Loss 0.001188; Train Loss 0.152831; Val Loss 63.997604
Epoch 409; Loss 0.049237; Train Loss 0.020108; Val Loss 64.557438
Epoch 410; Loss 0.000443; Train Loss 0.523348; Val Loss 73.345899
Epoch 411; Loss 0.175957; Train Loss 0.421492; Val Loss 62.956132
Epoch 412; Loss 0.155198; Train Loss 0.264663; Val Loss 70.411886
Epoch 413; Loss 0.011612; Train Loss 1.095575; Val Loss 66.849148
Epoch 414; Loss 0.029176; Train Loss 1.190459; Val Loss 71.274834
Epoch 415; Loss 0.026260; Train Loss 2.004174; Val Loss 62.043054
Epoch 416; Loss 1.102378; Train Loss 1.112265; Val Loss 70.369200
Epoch 417; Loss 0.132660; Train Loss 1.099899; Val Loss 68.051522
Epoch 418; Loss 0.001832; Train Loss 0.707529; Val Loss 66.052232
Epoch 419; Loss 0.020817; Train Loss 0.614832; Val Loss 64.667152
Epoch 420; Loss 0.471629; Train Loss 0.518053; Val Loss 69.331228
Epoch 421; Loss 0.051258; Train Loss 0.660823; Val Loss 66.951594
Epoch 422; Loss 0.075211; Train Loss 0.637541; Val Loss 72.139142
Epoch 423; Loss 0.059361; Train Loss 0.601216; Val Loss 63.563137
Epoch 424; Loss 0.007746; Train Loss 0.364953; Val Loss 69.201391
Epoch 425; Loss 0.025023; Train Loss 0.383654; Val Loss 69.013477
Epoch 426; Loss 0.008181; Train Loss 0.295479; Val Loss 67.172314
Epoch 427; Loss 0.015555; Train Loss 0.279229; Val Loss 66.939639
Epoch 428; Loss 0.215959; Train Loss 0.275603; Val Loss 69.339087
Epoch 429; Loss 0.000488; Train Loss 0.294304; Val Loss 66.754671
Epoch 430; Loss 0.017880; Train Loss 0.224888; Val Loss 68.099984
Epoch 431; Loss 0.080571; Train Loss 0.339695; Val Loss 72.701605
Epoch 432; Loss 0.034962; Train Loss 0.351187; Val Loss 65.211637
Epoch 433; Loss 0.003363; Train Loss 0.173650; Val Loss 70.375541
Epoch 434; Loss 0.016396; Train Loss 0.178899; Val Loss 70.679030
Epoch 435; Loss 0.000315; Train Loss 0.137250; Val Loss 68.849684
Epoch 436; Loss 0.235840; Train Loss 0.181330; Val Loss 69.767795
Epoch 437; Loss 0.004740; Train Loss 0.256399; Val Loss 68.290877
Epoch 438; Loss 0.027036; Train Loss 0.154475; Val Loss 72.837771
Epoch 439; Loss 0.091413; Train Loss 0.531412; Val Loss 62.644287
Epoch 440; Loss 0.022260; Train Loss 0.244252; Val Loss 70.772960
Epoch 441; Loss 0.001655; Train Loss 0.131155; Val Loss 69.030063
Epoch 442; Loss 0.034446; Train Loss 0.091149; Val Loss 72.234858
Epoch 443; Loss 0.009264; Train Loss 0.076176; Val Loss 70.394841
Epoch 444; Loss 0.053790; Train Loss 0.056691; Val Loss 71.258954
Epoch 445; Loss 0.000047; Train Loss 0.048371; Val Loss 70.809842
Epoch 446; Loss 0.028864; Train Loss 0.051972; Val Loss 71.946399
Epoch 447; Loss 0.012434; Train Loss 0.046301; Val Loss 72.275930
Epoch 448; Loss 0.072524; Train Loss 0.052581; Val Loss 71.568857
Epoch 449; Loss 0.006846; Train Loss 0.064212; Val Loss 72.421221
Epoch 450; Loss 0.001607; Train Loss 0.044194; Val Loss 70.752032
Epoch 451; Loss 0.019397; Train Loss 0.038217; Val Loss 73.239228
Epoch 452; Loss 0.008866; Train Loss 0.035342; Val Loss 71.233976
Epoch 453; Loss 0.000240; Train Loss 0.026843; Val Loss 72.564743
Epoch 454; Loss 0.000607; Train Loss 0.026743; Val Loss 72.131229
Epoch 455; Loss 0.000829; Train Loss 0.022518; Val Loss 73.489467
Epoch 456; Loss 0.001162; Train Loss 0.021450; Val Loss 72.042798
Epoch 457; Loss 0.000317; Train Loss 0.011071; Val Loss 73.093253
Epoch 458; Loss 0.006256; Train Loss 0.009709; Val Loss 72.535818
Epoch 459; Loss 0.000165; Train Loss 0.008121; Val Loss 72.834772
Epoch 460; Loss 0.000100; Train Loss 0.008278; Val Loss 73.019954
Epoch 461; Loss 0.000811; Train Loss 0.009509; Val Loss 72.586414
Epoch 462; Loss 0.000612; Train Loss 0.006453; Val Loss 73.378295
Epoch 463; Loss 0.000613; Train Loss 0.006487; Val Loss 72.635890
Epoch 464; Loss 0.008812; Train Loss 0.005276; Val Loss 73.199544
Epoch 465; Loss 0.000114; Train Loss 0.009134; Val Loss 73.315750
Epoch 466; Loss 0.001293; Train Loss 0.012622; Val Loss 73.324072
Epoch 467; Loss 0.003696; Train Loss 0.023132; Val Loss 72.461745
Epoch 468; Loss 0.004303; Train Loss 0.011174; Val Loss 74.465212
```

```
Epoch 469; Loss 0.010431; Train Loss 0.013417; Val Loss 72.581394
Epoch 470; Loss 0.019074; Train Loss 0.006140; Val Loss 73.095418
Epoch 471; Loss 0.003109; Train Loss 0.014166; Val Loss 74.745532
Epoch 472; Loss 0.012770; Train Loss 0.011702; Val Loss 74.387974
Epoch 473; Loss 0.000609; Train Loss 0.001958; Val Loss 73.390071
Epoch 474; Loss 0.000062; Train Loss 0.001614; Val Loss 73.362421
Epoch 475; Loss 0.000317; Train Loss 0.001591; Val Loss 73.743536
Epoch 476; Loss 0.003143; Train Loss 0.001538; Val Loss 73.784651
Epoch 477; Loss 0.000056; Train Loss 0.002044; Val Loss 73.349293
Epoch 478; Loss 0.000224; Train Loss 0.003666; Val Loss 73.844183
Epoch 479; Loss 0.000460; Train Loss 0.007850; Val Loss 72.633551
Epoch 480; Loss 0.000011; Train Loss 0.002452; Val Loss 74.190772
Epoch 481; Loss 0.001114; Train Loss 0.002165; Val Loss 73.986826
Epoch 482; Loss 0.000259; Train Loss 0.001547; Val Loss 73.792255
Epoch 483; Loss 0.000018; Train Loss 0.002502; Val Loss 73.810285
Epoch 484; Loss 0.000060; Train Loss 0.003225; Val Loss 74.392201
Epoch 485; Loss 0.000081; Train Loss 0.003674; Val Loss 73.497248
Epoch 486; Loss 0.000125; Train Loss 0.004460; Val Loss 74.586933
Epoch 487; Loss 0.000341; Train Loss 0.005922; Val Loss 73.270625
Epoch 488; Loss 0.000000; Train Loss 0.003914; Val Loss 74.064841
Epoch 489; Loss 0.000439; Train Loss 0.010458; Val Loss 73.554563
Epoch 490; Loss 0.000159; Train Loss 0.020061; Val Loss 74.149989
Epoch 491; Loss 0.000963; Train Loss 0.023355; Val Loss 72.833603
Epoch 492; Loss 0.000504; Train Loss 0.033470; Val Loss 75.810769
Epoch 493; Loss 0.000660; Train Loss 0.037938; Val Loss 72.971806
Epoch 494; Loss 0.000004; Train Loss 0.030157; Val Loss 75.832252
Epoch 495; Loss 0.000050; Train Loss 0.039322; Val Loss 72.857912
Epoch 496; Loss 0.002665; Train Loss 0.057268; Val Loss 75.993757
Epoch 497; Loss 0.000362; Train Loss 0.082166; Val Loss 71.825258
Epoch 498; Loss 0.003663; Train Loss 0.145666; Val Loss 76.952801
Epoch 499; Loss 0.002766; Train Loss 0.300085; Val Loss 71.969804
Epoch 500; Loss 0.001764; Train Loss 0.132302; Val Loss 76.963231
```





Maham Training

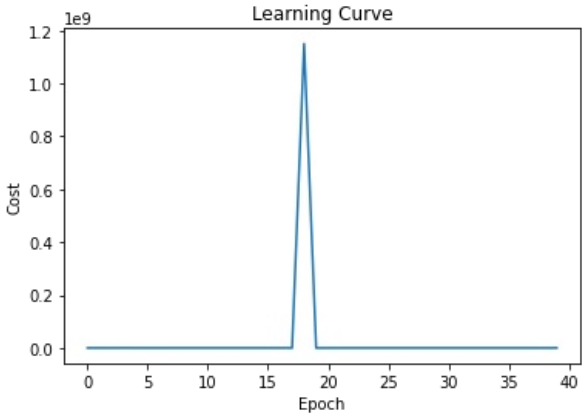
Here is the first training run, using default hyper parameter choices

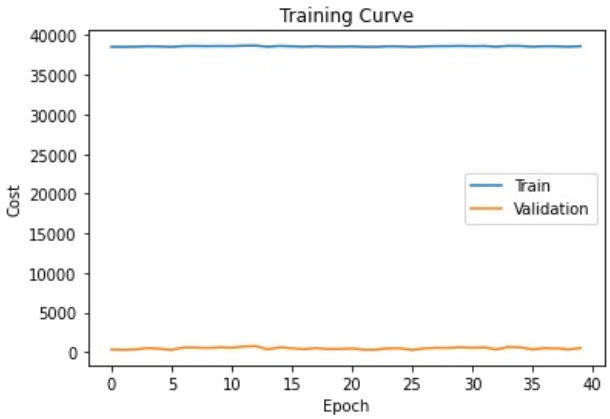
In [ ]:

```
model1 = StockPredictGRU()
# Initialize training and validation batches
train_loader = StockBatcher(train, batch_size=300, drop_last=False)
valid_loader = StockBatcher(valid, batch_size=300, drop_last=False)

train_rnn_network(model1, train_loader, valid_loader, weight_decay=0.0, learning_rate=0.1, num_epochs=40)
```

```
Epoch 1; Iteration 453; Loss 789.206726; Train Cost 38551.502197; Val Cost 339.839523
```

```
Epoch 2; Iteration 906; Loss 588.274780; Train Cost 38536.497083; Val Cost 291.000959
Epoch 3; Iteration 1359; Loss 1887.537476; Train Cost 38557.959318; Val Cost 345.884667
Epoch 4; Iteration 1812; Loss 151511.328125; Train Cost 38608.363089; Val Cost 523.613654
Epoch 5; Iteration 2265; Loss 1118.515137; Train Cost 38585.205343; Val Cost 433.401996
Epoch 6; Iteration 2718; Loss 696.411682; Train Cost 38536.837156; Val Cost 289.711277
Epoch 7; Iteration 3171; Loss 1694.158325; Train Cost 38636.204011; Val Cost 601.824719
Epoch 8; Iteration 3624; Loss 1928.826294; Train Cost 38644.781564; Val Cost 591.119013
Epoch 9; Iteration 4077; Loss 13893.999023; Train Cost 38623.943166; Val Cost 532.975603
Epoch 10; Iteration 4530; Loss 2615.634521; Train Cost 38640.457686; Val Cost 621.194703
Epoch 11; Iteration 4983; Loss 2077.883057; Train Cost 38624.957883; Val Cost 572.707605
Epoch 12; Iteration 5436; Loss 2316.899170; Train Cost 38679.094511; Val Cost 710.984225
Epoch 13; Iteration 5889; Loss 2211.541748; Train Cost 38688.690760; Val Cost 759.060410
Epoch 14; Iteration 6342; Loss 431.926605; Train Cost 38559.852110; Val Cost 353.971494
Epoch 15; Iteration 6795; Loss 2399.171387; Train Cost 38648.588514; Val Cost 622.621711
Epoch 16; Iteration 7248; Loss 2576.208252; Train Cost 38608.816648; Val Cost 496.054789
Epoch 17; Iteration 7701; Loss 639.464417; Train Cost 38563.504650; Val Cost 372.670673
Epoch 18; Iteration 8154; Loss 1668.257935; Train Cost 38616.788290; Val Cost 522.503868
Epoch 19; Iteration 8607; Loss 1150428672.000000; Train Cost 38565.050780; Val Cost 386.576043
Epoch 20; Iteration 9060; Loss 1480.611084; Train Cost 38568.936960; Val Cost 406.667673
Epoch 21; Iteration 9513; Loss 1596.412476; Train Cost 38590.107693; Val Cost 468.044783
Epoch 22; Iteration 9966; Loss 126.190247; Train Cost 38537.923203; Val Cost 308.794257
Epoch 23; Iteration 10419; Loss 827.590698; Train Cost 38540.666218; Val Cost 316.177797
Epoch 24; Iteration 10872; Loss 2559.895752; Train Cost 38601.872068; Val Cost 484.170838
Epoch 25; Iteration 11325; Loss 1973.863892; Train Cost 38596.389211; Val Cost 488.903059
Epoch 26; Iteration 11778; Loss 4551.390625; Train Cost 38533.705987; Val Cost 279.686375
Epoch 27; Iteration 12231; Loss 2073.544189; Train Cost 38593.943008; Val Cost 478.222814
Epoch 28; Iteration 12684; Loss 1655.202393; Train Cost 38628.650627; Val Cost 560.675220
Epoch 29; Iteration 13137; Loss 1503.936401; Train Cost 38626.787739; Val Cost 554.517112
Epoch 30; Iteration 13590; Loss 15014.802734; Train Cost 38651.020011; Val Cost 622.165070
Epoch 31; Iteration 14043; Loss 2364.752197; Train Cost 38623.915239; Val Cost 567.533508
Epoch 32; Iteration 14496; Loss 1937.820190; Train Cost 38641.991235; Val Cost 613.154673
Epoch 33; Iteration 14949; Loss 3872.751465; Train Cost 38558.578241; Val Cost 354.167146
Epoch 34; Iteration 15402; Loss 2174.188965; Train Cost 38658.775611; Val Cost 654.320855
Epoch 35; Iteration 15855; Loss 3394.889648; Train Cost 38644.124136; Val Cost 610.832520
Epoch 36; Iteration 16308; Loss 562.390991; Train Cost 38556.550057; Val Cost 346.039228
Epoch 37; Iteration 16761; Loss 4448.698242; Train Cost 38610.214685; Val Cost 504.270197
Epoch 38; Iteration 17214; Loss 877.532471; Train Cost 38602.280872; Val Cost 481.923219
Epoch 39; Iteration 17667; Loss 781.620728; Train Cost 38550.211950; Val Cost 343.991492
Epoch 40; Iteration 18120; Loss 151420.562500; Train Cost 38617.118164; Val Cost 531.838752
```





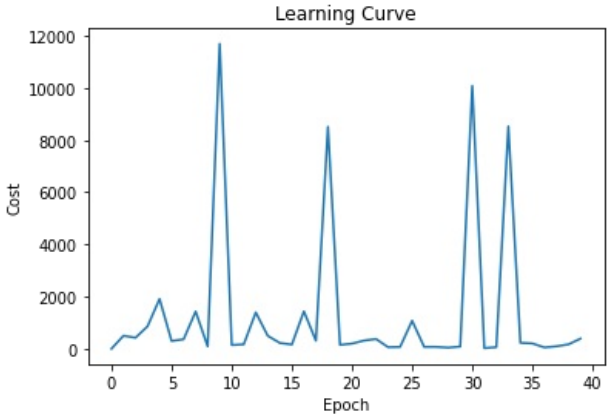
Second training run, increased weight decay, decreased learning rate to reduce fluctuations.

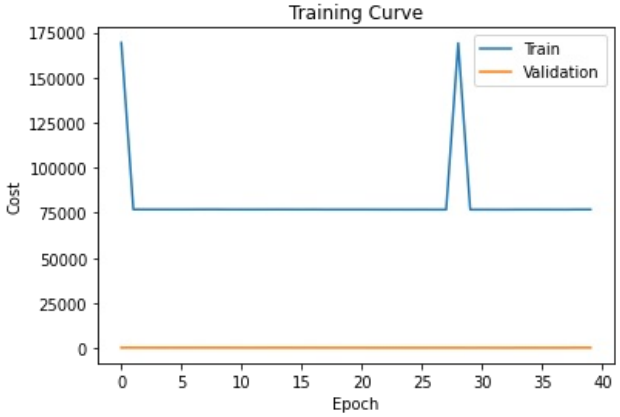
In [ ]:

```
model2 = StockPredictGRU()
# Initialize training and validation batches
train_loader = StockBatcher(train, batch_size=300, drop_last=False)
```

```
valid_loader = StockBatcher(valid, batch_size=300, drop_last=False)

train_rnn_network(model2, train_loader, valid_loader, weight_decay=0.01, learning_rate=0.01, num_epochs=40)
```

```
Epoch 1; Iteration 453; Loss 0.528757; Train Cost 169328.295378; Val Cost 240.907101
Epoch 2; Iteration 906; Loss 502.244843; Train Cost 76889.521144; Val Cost 241.269342
Epoch 3; Iteration 1359; Loss 425.357025; Train Cost 76877.465221; Val Cost 227.869557
Epoch 4; Iteration 1812; Loss 861.351990; Train Cost 76875.771002; Val Cost 224.755693
Epoch 5; Iteration 2265; Loss 1912.194092; Train Cost 76875.248851; Val Cost 228.264231
Epoch 6; Iteration 2718; Loss 300.101196; Train Cost 76864.866192; Val Cost 221.429943
Epoch 7; Iteration 3171; Loss 364.813904; Train Cost 76887.164061; Val Cost 234.519004
Epoch 8; Iteration 3624; Loss 1430.421021; Train Cost 76904.998194; Val Cost 228.988458
Epoch 9; Iteration 4077; Loss 91.781174; Train Cost 76900.938629; Val Cost 229.250957
Epoch 10; Iteration 4530; Loss 11694.152344; Train Cost 76863.771828; Val Cost 219.268567
Epoch 11; Iteration 4983; Loss 145.071518; Train Cost 76853.373256; Val Cost 206.533886
Epoch 12; Iteration 5436; Loss 170.735535; Train Cost 76844.615022; Val Cost 199.875281
Epoch 13; Iteration 5889; Loss 1394.375000; Train Cost 76866.939036; Val Cost 211.842331
Epoch 14; Iteration 6342; Loss 496.122253; Train Cost 76882.641717; Val Cost 224.300547
Epoch 15; Iteration 6795; Loss 223.736328; Train Cost 76867.862916; Val Cost 213.135740
Epoch 16; Iteration 7248; Loss 162.234833; Train Cost 76845.836325; Val Cost 204.419464
Epoch 17; Iteration 7701; Loss 1436.535645; Train Cost 76875.414967; Val Cost 217.129675
Epoch 18; Iteration 8154; Loss 311.264008; Train Cost 76835.454566; Val Cost 199.914339
Epoch 19; Iteration 8607; Loss 8511.582031; Train Cost 76839.293042; Val Cost 200.250575
Epoch 20; Iteration 9060; Loss 149.826218; Train Cost 76841.925451; Val Cost 204.243598
Epoch 21; Iteration 9513; Loss 199.718170; Train Cost 76840.223633; Val Cost 206.515280
Epoch 22; Iteration 9966; Loss 316.341888; Train Cost 76827.991177; Val Cost 189.958744
Epoch 23; Iteration 10419; Loss 377.078888; Train Cost 76811.715789; Val Cost 185.629892
Epoch 24; Iteration 10872; Loss 62.326370; Train Cost 76818.988725; Val Cost 196.269601
Epoch 25; Iteration 11325; Loss 71.256523; Train Cost 76804.973514; Val Cost 177.471406
Epoch 26; Iteration 11778; Loss 1075.908936; Train Cost 76821.208918; Val Cost 181.528420
Epoch 27; Iteration 12231; Loss 73.308517; Train Cost 76796.475230; Val Cost 184.748382
Epoch 28; Iteration 12684; Loss 73.711021; Train Cost 76801.674700; Val Cost 190.870363
Epoch 29; Iteration 13137; Loss 44.478607; Train Cost 169160.284707; Val Cost 180.387553
Epoch 30; Iteration 13590; Loss 85.722656; Train Cost 76793.124734; Val Cost 169.631060
Epoch 31; Iteration 14043; Loss 10083.016602; Train Cost 76805.476020; Val Cost 182.233514
Epoch 32; Iteration 14496; Loss 25.447054; Train Cost 76793.463592; Val Cost 170.060649
Epoch 33; Iteration 14949; Loss 61.736584; Train Cost 76788.174354; Val Cost 163.840297
Epoch 34; Iteration 15402; Loss 8527.880859; Train Cost 76825.434149; Val Cost 188.568102
Epoch 35; Iteration 15855; Loss 230.238235; Train Cost 76827.975424; Val Cost 185.929976
Epoch 36; Iteration 16308; Loss 205.152161; Train Cost 76832.615981; Val Cost 198.478619
Epoch 37; Iteration 16761; Loss 52.720432; Train Cost 76819.604463; Val Cost 195.185048
Epoch 38; Iteration 17214; Loss 95.500214; Train Cost 76814.843674; Val Cost 188.456585
Epoch 39; Iteration 17667; Loss 170.549133; Train Cost 76868.370351; Val Cost 250.668358
Epoch 40; Iteration 18120; Loss 391.289093; Train Cost 76860.095289; Val Cost 229.433258
```



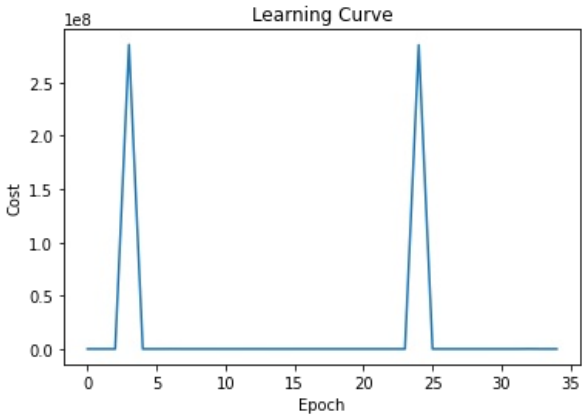


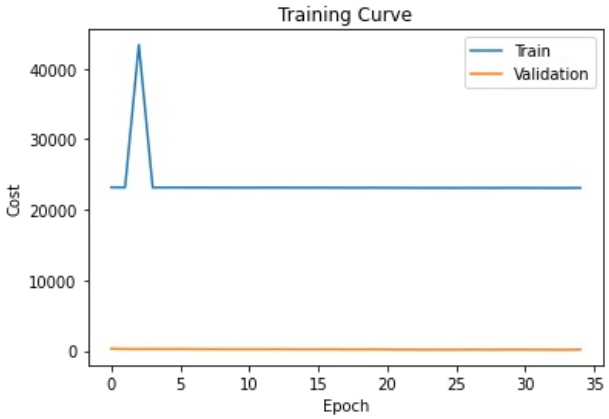
Third training run, increased batch size, reduced epochs, removed weight decay. To decrease variance

In [ ]:

```
model3 = StockPredictGRU()
# Initialize training and validation batches
train_loader = StockBatcher(train, batch_size=500, drop_last=False)
valid_loader = StockBatcher(valid, batch_size=500, drop_last=False)

train_rnn_network(model3, train_loader, valid_loader, weight_decay=0.0, learning_rate=0.01, num_epochs=35)
```

```
Epoch 1; Iteration 447; Loss 544.619080; Train Cost 23185.387694; Val Cost 281.402192
Epoch 2; Iteration 894; Loss 3.401223; Train Cost 23165.382848; Val Cost 235.670053
Epoch 3; Iteration 1341; Loss 3.989877; Train Cost 43411.963184; Val Cost 223.621297
Epoch 4; Iteration 1788; Loss 285478528.000000; Train Cost 23164.527857; Val Cost 237.364861
Epoch 5; Iteration 2235; Loss 3.692450; Train Cost 23159.486725; Val Cost 222.341236
Epoch 6; Iteration 2682; Loss 504.832001; Train Cost 23158.921699; Val Cost 236.204860
Epoch 7; Iteration 3129; Loss 41811.003906; Train Cost 23152.906040; Val Cost 214.632494
Epoch 8; Iteration 3576; Loss 3476.242676; Train Cost 23148.682148; Val Cost 204.702134
Epoch 9; Iteration 4023; Loss 4416.174316; Train Cost 23146.042879; Val Cost 205.919014
Epoch 10; Iteration 4470; Loss 469.536804; Train Cost 23137.486671; Val Cost 198.414089
Epoch 11; Iteration 4917; Loss 402.019836; Train Cost 23137.814889; Val Cost 206.544145
Epoch 12; Iteration 5364; Loss 243.897919; Train Cost 23144.348165; Val Cost 196.321054
Epoch 13; Iteration 5811; Loss 4673.761719; Train Cost 23145.517984; Val Cost 209.953457
Epoch 14; Iteration 6258; Loss 188.264740; Train Cost 23138.778495; Val Cost 196.275758
Epoch 15; Iteration 6705; Loss 9.402424; Train Cost 23135.922093; Val Cost 191.087177
Epoch 16; Iteration 7152; Loss 197.690704; Train Cost 23140.261198; Val Cost 193.109666
Epoch 17; Iteration 7599; Loss 1013.041016; Train Cost 23132.002119; Val Cost 198.856636
Epoch 18; Iteration 8046; Loss 1000.761169; Train Cost 23123.424667; Val Cost 175.709940
Epoch 19; Iteration 8493; Loss 120.214584; Train Cost 23122.945546; Val Cost 174.665243
Epoch 20; Iteration 8940; Loss 490.475098; Train Cost 23129.672011; Val Cost 189.435931
Epoch 21; Iteration 9387; Loss 1079.980469; Train Cost 23116.927099; Val Cost 172.033775
Epoch 22; Iteration 9834; Loss 5786.249023; Train Cost 23110.763809; Val Cost 152.961493
Epoch 23; Iteration 10281; Loss 326.730042; Train Cost 23108.809513; Val Cost 148.524880
Epoch 24; Iteration 10728; Loss 0.568280; Train Cost 23098.982919; Val Cost 138.364163
Epoch 25; Iteration 11175; Loss 285188640.000000; Train Cost 23105.409271; Val Cost 141.659444
Epoch 26; Iteration 11622; Loss 85.637398; Train Cost 23095.969461; Val Cost 139.359127
Epoch 27; Iteration 12069; Loss 22.090635; Train Cost 23108.494596; Val Cost 154.048452
Epoch 28; Iteration 12516; Loss 18.709137; Train Cost 23100.796578; Val Cost 146.496643
Epoch 29; Iteration 12963; Loss 480.499939; Train Cost 23096.576611; Val Cost 143.259754
Epoch 30; Iteration 13410; Loss 2.234340; Train Cost 23112.251669; Val Cost 155.338002
Epoch 31; Iteration 13857; Loss 68.401688; Train Cost 23100.059411; Val Cost 146.910329
Epoch 32; Iteration 14304; Loss 2917.834473; Train Cost 23094.408892; Val Cost 146.544339
Epoch 33; Iteration 14751; Loss 91832.281250; Train Cost 23089.378082; Val Cost 134.335738
Epoch 34; Iteration 15198; Loss 24.985926; Train Cost 23090.882049; Val Cost 127.805559
Epoch 35; Iteration 15645; Loss 645.217468; Train Cost 23109.042306; Val Cost 162.596835
```





Here's the second training run, decreased batch size, added weight decay,

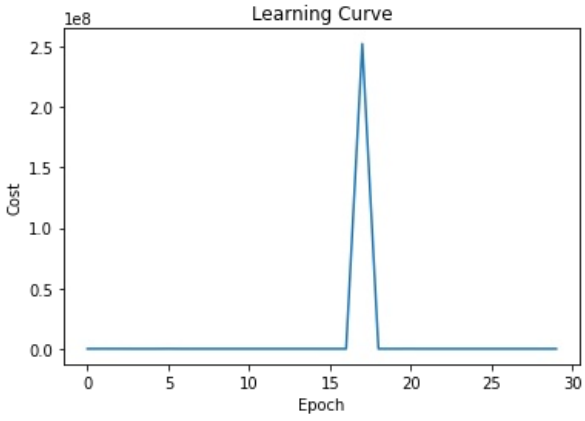
Mac Training

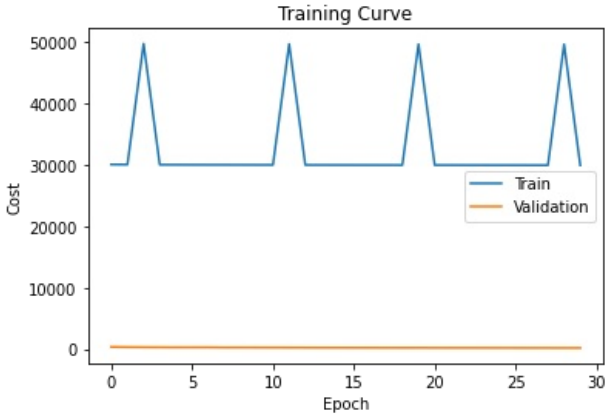
Mac Training

For the next 3 sets of training runs, experimented with a smaller learning rate and varied the value for weight decay. First we used 5×10^{-4}.

In [ ]:
```
model4 = StockPredictGRU()
# Initialize training and validation batches
train_loader = StockBatcher(train, batch_size=500, drop_last=False)
valid_loader = StockBatcher(valid, batch_size=500, drop_last=False)

train_rnn_network(model4, train_loader, valid_loader, weight_decay=5*(10**(-4)), learning_rate=0.001, num_epochs=
```

```
Epoch 1; Iteration 445; Loss 2214.310547; Train Cost 30029.734195; Val Cost 328.481722
Epoch 2; Iteration 890; Loss 59069.589844; Train Cost 30014.593183; Val Cost 298.865638
Epoch 3; Iteration 1335; Loss 36391.789062; Train Cost 49695.167514; Val Cost 277.555205
Epoch 4; Iteration 1780; Loss 30.602365; Train Cost 29991.061082; Val Cost 266.575729
Epoch 5; Iteration 2225; Loss 0.007248; Train Cost 29984.288496; Val Cost 256.157176
Epoch 6; Iteration 2670; Loss 57442.265625; Train Cost 29978.241486; Val Cost 253.851584
Epoch 7; Iteration 3115; Loss 699.238342; Train Cost 29973.679942; Val Cost 259.200698
Epoch 8; Iteration 3560; Loss 1028.793823; Train Cost 29972.113834; Val Cost 234.573792
Epoch 9; Iteration 4005; Loss 4159.569824; Train Cost 29966.352762; Val Cost 240.167978
Epoch 10; Iteration 4450; Loss 659.599792; Train Cost 29961.679147; Val Cost 233.586174
Epoch 11; Iteration 4895; Loss 1.288036; Train Cost 29962.262395; Val Cost 228.724117
Epoch 12; Iteration 5340; Loss 878.576843; Train Cost 49644.426795; Val Cost 219.294130
Epoch 13; Iteration 5785; Loss 4062.452637; Train Cost 29951.875847; Val Cost 212.697590
Epoch 14; Iteration 6230; Loss 1610.398071; Train Cost 29947.770079; Val Cost 207.465725
Epoch 15; Iteration 6675; Loss 0.041642; Train Cost 29945.822053; Val Cost 203.582564
Epoch 16; Iteration 7120; Loss 0.196857; Train Cost 29941.631331; Val Cost 199.880776
Epoch 17; Iteration 7565; Loss 254.200745; Train Cost 29938.929306; Val Cost 201.401388
Epoch 18; Iteration 8010; Loss 252180736.000000; Train Cost 29936.560038; Val Cost 193.835838
Epoch 19; Iteration 8455; Loss 167.383041; Train Cost 29932.860534; Val Cost 190.187502
Epoch 20; Iteration 8900; Loss 0.820196; Train Cost 49611.843330; Val Cost 193.834556
Epoch 21; Iteration 9345; Loss 32672.332031; Train Cost 29932.077638; Val Cost 184.314111
Epoch 22; Iteration 9790; Loss 1.353788; Train Cost 29931.329626; Val Cost 180.417591
Epoch 23; Iteration 10235; Loss 31.404306; Train Cost 29924.985714; Val Cost 180.201295
Epoch 24; Iteration 10680; Loss 0.034904; Train Cost 29922.519285; Val Cost 177.223786
Epoch 25; Iteration 11125; Loss 0.270856; Train Cost 29919.083987; Val Cost 175.979813
Epoch 26; Iteration 11570; Loss 0.668878; Train Cost 29918.913680; Val Cost 172.214575
Epoch 27; Iteration 12015; Loss 15.802783; Train Cost 29915.452643; Val Cost 169.173174
Epoch 28; Iteration 12460; Loss 137.756210; Train Cost 29910.529355; Val Cost 167.026284
Epoch 29; Iteration 12905; Loss 0.160565; Train Cost 49583.860974; Val Cost 163.377343
Epoch 30; Iteration 13350; Loss 0.477753; Train Cost 29906.194459; Val Cost 163.746322
```



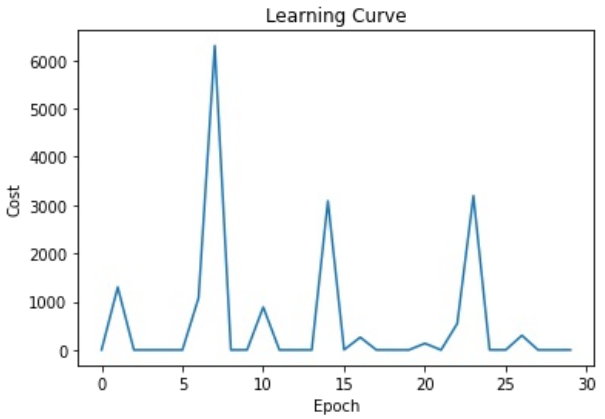


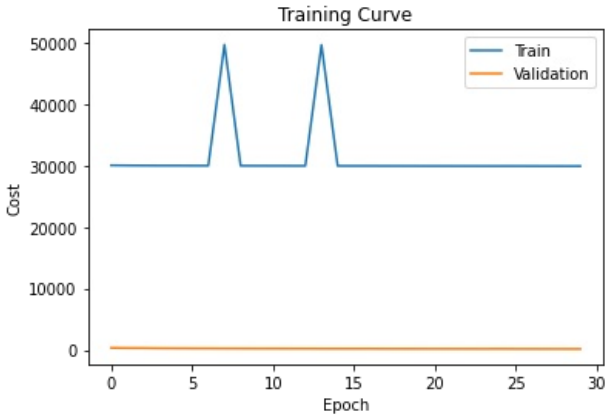
Kept learning rate from previous one, changed weight decay to 0.

In [ ]:
```
model5 = StockPredictGRU()
```

```
model5 = StockPredictGRU()
# Initialize training and validation batches
train_loader = StockBatcher(train, batch_size=500, drop_last=False)
valid_loader = StockBatcher(valid, batch_size=500, drop_last=False)

train_rnn_network(model5, train_loader, valid_loader, weight_decay=0, learning_rate=0.001, num_epochs=30)
```

```
Epoch 1; Iteration 445; Loss 3.862795; Train Cost 30038.728191; Val Cost 337.112597
Epoch 2; Iteration 890; Loss 1301.302124; Train Cost 30015.774751; Val Cost 304.311358
Epoch 3; Iteration 1335; Loss 0.032888; Train Cost 30000.893953; Val Cost 298.728530
Epoch 4; Iteration 1780; Loss 0.082459; Train Cost 29992.718483; Val Cost 268.995021
Epoch 5; Iteration 2225; Loss 0.018141; Train Cost 29990.927183; Val Cost 258.606545
Epoch 6; Iteration 2670; Loss 0.040800; Train Cost 29980.827747; Val Cost 250.568082
Epoch 7; Iteration 3115; Loss 1079.681396; Train Cost 29975.454392; Val Cost 243.220980
Epoch 8; Iteration 3560; Loss 6304.453125; Train Cost 49667.856853; Val Cost 237.833730
Epoch 9; Iteration 4005; Loss 0.223532; Train Cost 29969.070892; Val Cost 231.475644
Epoch 10; Iteration 4450; Loss 1.421142; Train Cost 29963.336521; Val Cost 226.098501
Epoch 11; Iteration 4895; Loss 887.299927; Train Cost 29959.845134; Val Cost 222.447847
Epoch 12; Iteration 5340; Loss 0.002169; Train Cost 29955.848291; Val Cost 216.459281
Epoch 13; Iteration 5785; Loss 0.319271; Train Cost 29952.753426; Val Cost 214.123998
Epoch 14; Iteration 6230; Loss 0.001498; Train Cost 49637.080990; Val Cost 209.124355
Epoch 15; Iteration 6675; Loss 3087.448975; Train Cost 29946.758543; Val Cost 205.230849
Epoch 16; Iteration 7120; Loss 3.975872; Train Cost 29943.900977; Val Cost 201.789749
Epoch 17; Iteration 7565; Loss 261.778198; Train Cost 29943.004899; Val Cost 206.706907
Epoch 18; Iteration 8010; Loss 0.915869; Train Cost 29936.769629; Val Cost 193.491965
Epoch 19; Iteration 8455; Loss 0.142348; Train Cost 29933.591070; Val Cost 190.014330
Epoch 20; Iteration 8900; Loss 0.051899; Train Cost 29931.232430; Val Cost 187.866966
Epoch 21; Iteration 9345; Loss 135.881210; Train Cost 29928.071236; Val Cost 184.036709
Epoch 22; Iteration 9790; Loss 0.030044; Train Cost 29925.843994; Val Cost 183.569090
Epoch 23; Iteration 10235; Loss 546.894836; Train Cost 29922.748796; Val Cost 178.321357
Epoch 24; Iteration 10680; Loss 3193.667969; Train Cost 29919.953444; Val Cost 174.265048
Epoch 25; Iteration 11125; Loss 2.633058; Train Cost 29918.101463; Val Cost 187.104717
Epoch 26; Iteration 11570; Loss 1.362639; Train Cost 29921.169572; Val Cost 172.608441
Epoch 27; Iteration 12015; Loss 300.884796; Train Cost 29914.667754; Val Cost 169.831866
Epoch 28; Iteration 12460; Loss 0.519497; Train Cost 29911.407681; Val Cost 166.385499
Epoch 29; Iteration 12905; Loss 0.260414; Train Cost 29909.558584; Val Cost 163.230102
Epoch 30; Iteration 13350; Loss 1.251674; Train Cost 29907.429824; Val Cost 161.584015
```



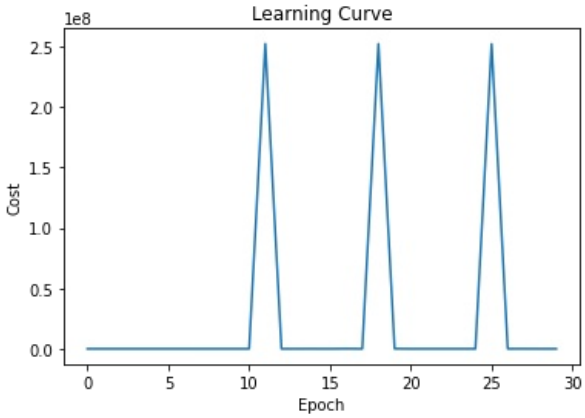


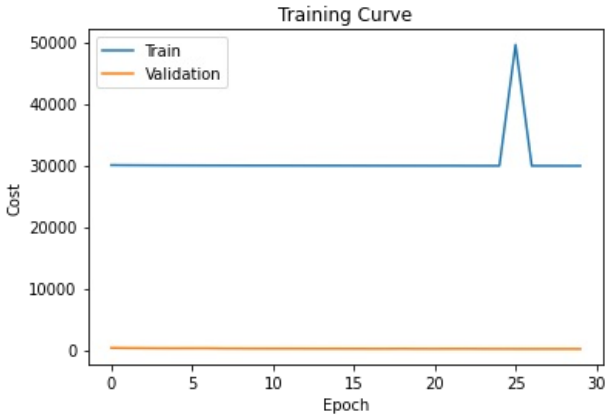
Changed the weight decay to 5×10^{-3}

In [ ]:

```
model6 = StockPredictGRU()
# Initialize training and validation batches
train_loader = StockBatcher(train, batch_size=500, drop_last=False)
valid_loader = StockBatcher(valid, batch_size=500, drop_last=False)

train_rnn_network(model6, train_loader, valid_loader, weight_decay=5*(10**(-3)), learning_rate=0.001, num_epochs=
```

```
Epoch 1; Iteration 445; Loss 902.137268; Train Cost 30029.861025; Val Cost 330.187828
Epoch 2; Iteration 890; Loss 0.239042; Train Cost 30012.377227; Val Cost 299.831699
Epoch 3; Iteration 1335; Loss 846.572937; Train Cost 30001.566895; Val Cost 282.644320
Epoch 4; Iteration 1780; Loss 0.150965; Train Cost 29991.683229; Val Cost 267.087240
Epoch 5; Iteration 2225; Loss 0.004648; Train Cost 29985.486534; Val Cost 259.984632
Epoch 6; Iteration 2670; Loss 1297.093384; Train Cost 29980.711316; Val Cost 267.035189
Epoch 7; Iteration 3115; Loss 0.131336; Train Cost 29975.184483; Val Cost 264.845648
Epoch 8; Iteration 3560; Loss 0.373291; Train Cost 29971.227793; Val Cost 236.680129
Epoch 9; Iteration 4005; Loss 340.335999; Train Cost 29966.921754; Val Cost 231.229962
Epoch 10; Iteration 4450; Loss 1.417768; Train Cost 29962.132803; Val Cost 224.141969
Epoch 11; Iteration 4895; Loss 3533.397461; Train Cost 29961.897996; Val Cost 221.674668
Epoch 12; Iteration 5340; Loss 252244736.000000; Train Cost 29956.417344; Val Cost 216.967303
Epoch 13; Iteration 5785; Loss 3376.964600; Train Cost 29952.341867; Val Cost 212.007835
Epoch 14; Iteration 6230; Loss 0.095984; Train Cost 29948.726360; Val Cost 207.230295
Epoch 15; Iteration 6675; Loss 34173.320312; Train Cost 29945.308057; Val Cost 208.770210
Epoch 16; Iteration 7120; Loss 0.223718; Train Cost 29942.872313; Val Cost 201.771852
Epoch 17; Iteration 7565; Loss 90640.101562; Train Cost 29939.405142; Val Cost 197.320880
Epoch 18; Iteration 8010; Loss 6358.762695; Train Cost 29936.636664; Val Cost 193.463027
Epoch 19; Iteration 8455; Loss 252179392.000000; Train Cost 29934.416814; Val Cost 205.972374
Epoch 20; Iteration 8900; Loss 87156.437500; Train Cost 29933.050690; Val Cost 187.548313
Epoch 21; Iteration 9345; Loss 0.025427; Train Cost 29928.619409; Val Cost 184.649566
Epoch 22; Iteration 9790; Loss 0.277478; Train Cost 29933.570045; Val Cost 195.341100
Epoch 23; Iteration 10235; Loss 0.043527; Train Cost 29924.481608; Val Cost 195.508819
Epoch 24; Iteration 10680; Loss 0.148088; Train Cost 29920.492257; Val Cost 177.164979
Epoch 25; Iteration 11125; Loss 0.029515; Train Cost 29918.786062; Val Cost 174.752590
Epoch 26; Iteration 11570; Loss 252110896.000000; Train Cost 49598.491862; Val Cost 171.217120
Epoch 27; Iteration 12015; Loss 3272.985107; Train Cost 29912.300642; Val Cost 166.945953
Epoch 28; Iteration 12460; Loss 34613.980469; Train Cost 29914.273150; Val Cost 164.534319
Epoch 29; Iteration 12905; Loss 0.056904; Train Cost 29908.764210; Val Cost 167.507908
Epoch 30; Iteration 13350; Loss 580.347534; Train Cost 29909.844177; Val Cost 161.306922
```





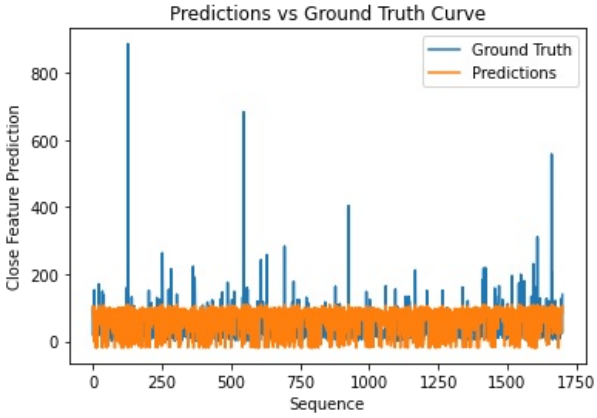
Model Test Set Evaluation

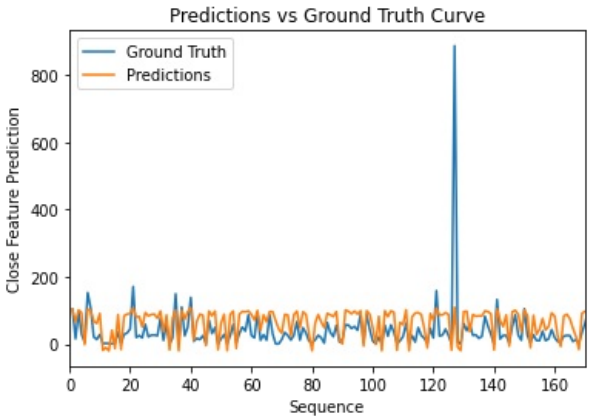
In [ ]:

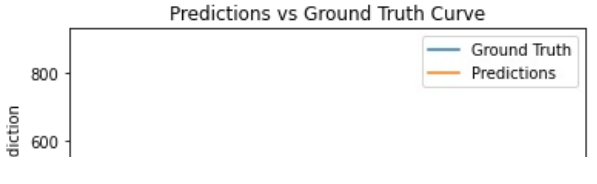
```
model3.eval()

truevals = []
pred = []
iters = []

"""""
The following function compares the predictions of our model with the ground truth labels of the test set.
"""""
def get_testcost(model):
```

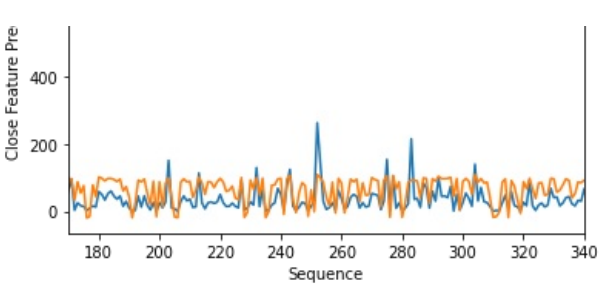
```
  count = 0

  for inp, label in test:
    if len(inp) > 0:
      count += 1
      v1 = torch.flatten(inp[0]).unsqueeze(0)
      v2 = torch.tensor(v1.unsqueeze(2), dtype=torch.float)
      output, _ = model(v2)
      pred.append(float(output))
      truevals.append(float(label))
      iters.append(count)


  plt.title("Predictions vs Ground Truth Curve")
  plt.plot(iters, truevals, label="Ground Truth")
  plt.plot(iters, pred, label="Predictions")
  plt.xlabel("Sequence")
  plt.ylabel("Close Feature Prediction")
  plt.legend(loc='best')
  plt.show()

  lim = 0
  for i in range(10):
    plt.title("Predictions vs Ground Truth Curve")
    plt.plot(iters, truevals, label="Ground Truth")
    plt.plot(iters, pred, label="Predictions")
    plt.xlabel("Sequence")
    plt.ylabel("Close Feature Prediction")
    plt.legend(loc='best')
    plt.xlim([lim, lim + 170])
    lim += 170
    plt.show()



get_testcost(model3)
```

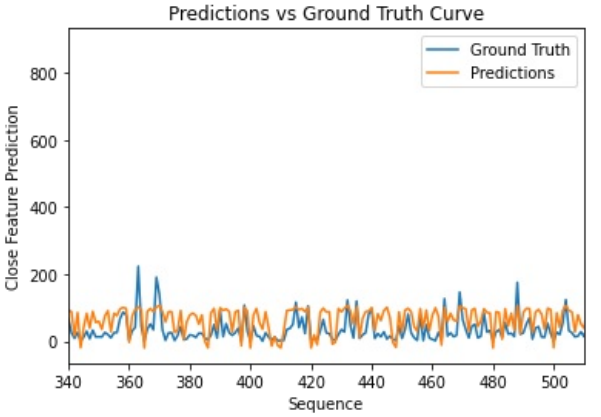
```
/usr/local/lib/python3.7/dist-packages/ipykernel_launcher.py:15: UserWarning: To copy construct from a tensor, it is recommended to use sourceTensor.clone().detach() or sourceTensor.clone().detach().requires_grad_(True), rather than torch.tensor(sourceTensor).
  from ipykernel import kernelapp as app
```

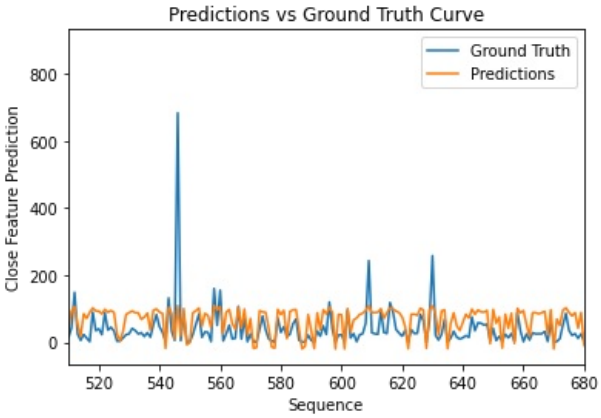


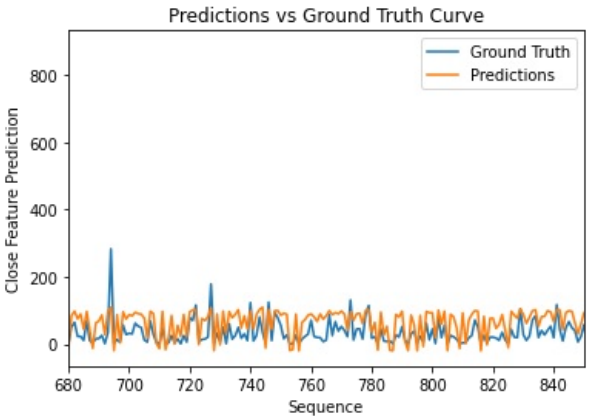


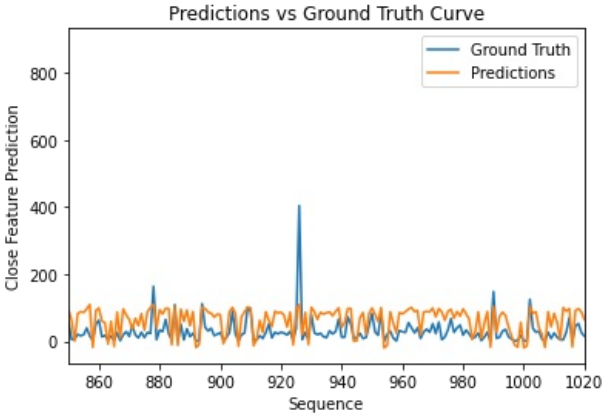


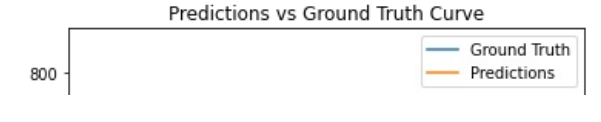

Close Feature Pre
400
200
0
180
200
220
240
260
280
300
320
340
Sequence

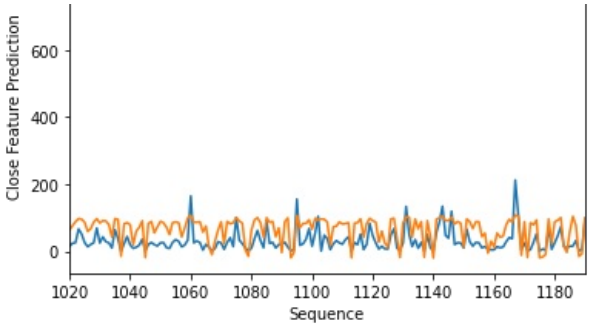

Predictions vs Ground Truth Curve
Ground Truth
Predictions
Close Feature Prediction
800
600
400
200
0
340
360
380
400
420
440
460
480
500
Sequence

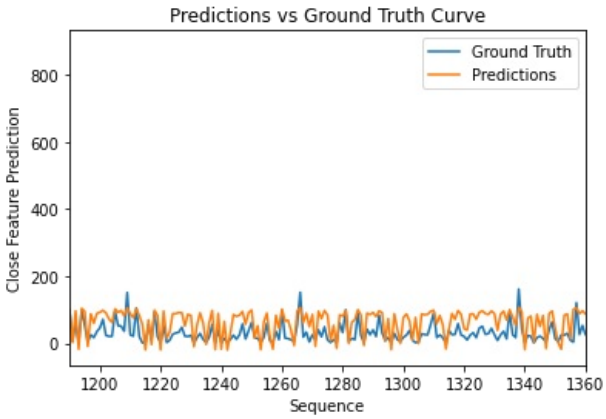

Predictions vs Ground Truth Curve
Ground Truth
Predictions
Close Feature Prediction
800
600
400
200
0
520
540
560
580
600
620
640
660
680
Sequence

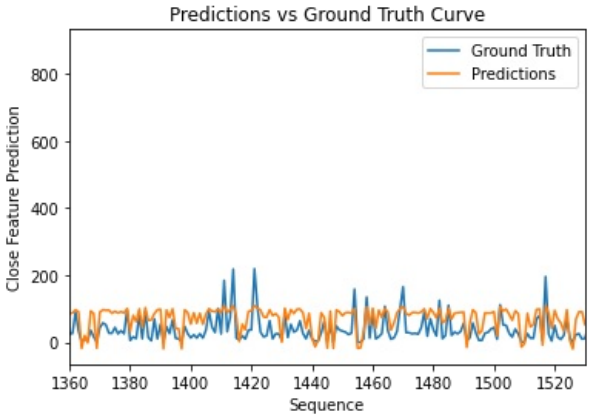

Predictions vs Ground Truth Curve
Ground Truth
Predictions
Close Feature Prediction
800
600
400
200
0
680
700
720
740
760
780
800
820
840
Sequence

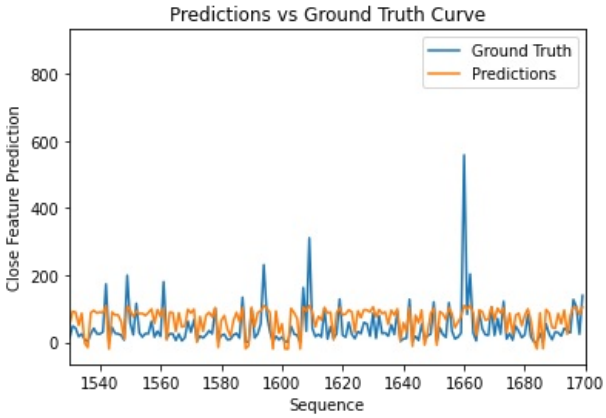

Predictions vs Ground Truth Curve
Ground Truth
Predictions
Close Feature Prediction
800
600
400
200
0
860
880
900
920
940
960
980
1000
1020
Sequence


Predictions vs Ground Truth Curve
Ground Truth
Predictions
800









Computed the closest ("most correct") prediction in comparison with ground truth

In [ ]:

```
for i in range(len(truevals)):
  if abs(truevals[i] - pred[i]) < 1:
    print(truevals[i], pred[i], i)
    print(abs(truevals[i] - pred[i]))
    print("\n")
```

```
103.3 103.12125396728516 0
0.1787460327148409


96.41 95.99547576904297 170
```

```
0.41452423095702784


3.5 4.364557266235352 298
0.8645572662353516


4.67 4.944619178771973 377
0.2746191787719727


12.64 12.927889823913574 407
0.28788982391357365


3.91 3.9494571685791016 526
0.03945716857910142


104.76 104.87962341308594 828
0.11962341308593238


101.45 101.5103530883789 908
0.06035308837890341


4.65 4.854091644287109 1011
0.20409164428710902


12.9 12.843536376953125 1065
0.056463623046875355


101.0 100.5049819946289 1074
0.49501800537109375


15.0 14.597397804260254 1182
0.4026021957397461


98.63 99.37718963623047 1189
0.7471896362304733


16.53 15.908927917480469 1250
0.6210720825195324


105.53 105.02574920654297 1668
0.5042507934570324


105.08 104.44143676757812 1696
0.6385632324218733
```

Computed the worst ("least correct") prediction in comparison with ground truth (an example of a prediction that was the least accurate)

In [ ]:

```
curr_highest = 0
curr_true = None
curr_pred = None
curr_i = None

for i in range(len(truevals)):

  if abs(truevals[i] - pred[i]) > curr_highest:
    curr_highest = abs(truevals[i] - pred[i])
    curr_true = truevals[i]
    curr_pred = pred[i]
    curr_i = i

print(curr_highest, curr_true, curr_pred, curr_i)
```

```
778.2845889282227 887.87 109.58541107177734 126
```