```
#importing packages and setting style preferences
import torch
import os
import numpy as np
import pandas as pd
from tqdm import tqdm
import seaborn as sns
from pylab import rcParams
import matplotlib.pyplot as plt
from matplotlib import rc
from sklearn.preprocessing import MinMaxScaler
from pandas.plotting import register_matplotlib_converters
from torch import nn, optim
import csv

%matplotlib inline
%config InlineBackend.figure_format='retina'

sns.set(style='whitegrid', palette='muted', font_scale=1.2)

HAPPY_COLORS_PALETTE = ["#01BEFE", "#FFDD00", "#FF7D00", "#FF006D", "#93D30C", "#8F00FF"]

sns.set_palette(sns.color_palette(HAPPY_COLORS_PALETTE))

rcParams['figure.figsize'] = 14, 10
register_matplotlib_converters()

RANDOM_SEED = 42
np.random.seed(RANDOM_SEED)
torch.manual_seed(RANDOM_SEED)
```

```
<torch._C.Generator at 0x7fac1f0a86b0>
```

```
!gdown --id 1zDF5bhbt8wSteObQrDoc2MFMoB5w1NtU #since we are using a google collab we can easily access files uploaded on
```

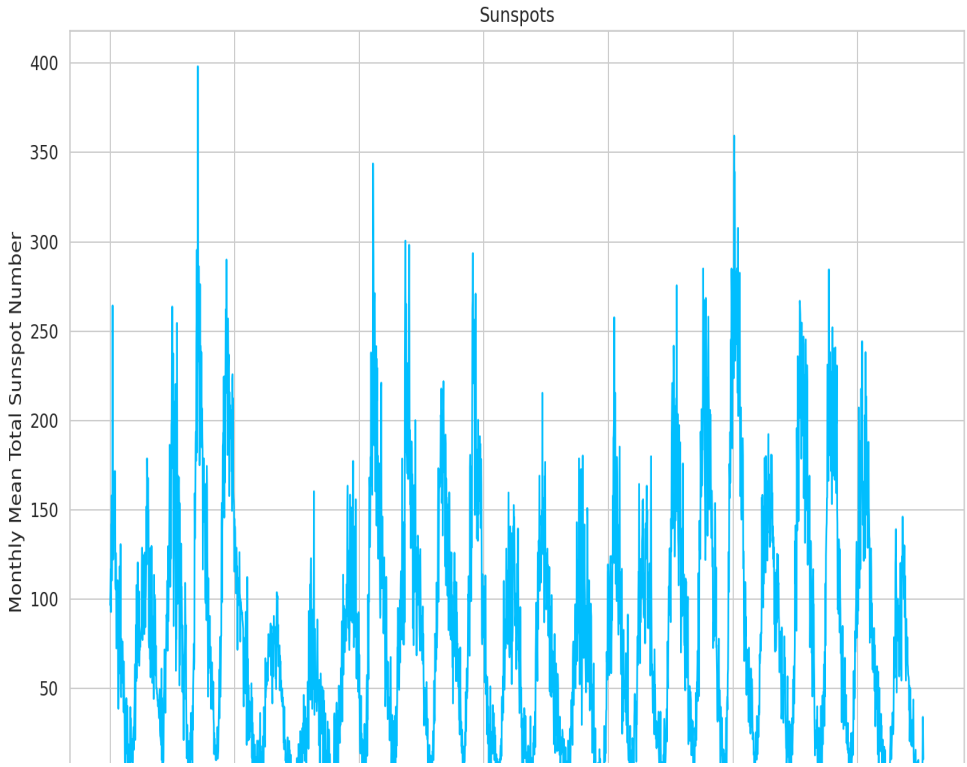
```
/usr/local/lib/python3.7/dist-packages/gdown/cli.py:131: FutureWarning: Option `--id` was deprecated in version 4.3
  category=FutureWarning,
Downloading...
From: https://drive.google.com/uc?id=1zDF5bhbt8wSteObQrDoc2MFMoB5w1NtU
To: /content/Sunspots2.0.csv
100% 69.5k/69.5k [00:00<00:00, 77.6MB/s]
```

```
df = pd.read_csv('Sunspots2.0.csv')
df = df.drop(columns=['Index', 'Date'])
df.head()
df.transpose()
#viewing data
```

	0	1	2	3	4	5	6	7	8	9	...	3255	3256
Monthly Mean Sunspots	96.7	104.3	116.7	92.8	141.7	139.2	158.0	110.5	126.5	125.8	...	5.2	0.2

1 rows × 3265 columns

```
plt.plot(df) #plotting data
plt.xlabel("Months after January 1749");
plt.ylabel("Monthly Mean Total Sunspot Number");
plt.title("Sunspots");
```




```
df.shape
data = np.array(df)

data = data.flatten()
data.shape #viewing shape
```

```
(3265,)
```

```
test_data_size = 1000

train_data = data[:-test_data_size]
test_data = data[-test_data_size:]

train_data.shape
#using partial data instead of full data and splitting for train/test
```

```
(2265,)
```

```
scaler = MinMaxScaler()

scaler = scaler.fit(np.expand_dims(train_data, axis=1))

train_data = scaler.transform(np.expand_dims(train_data, axis=1))

test_data = scaler.transform(np.expand_dims(test_data, axis=1))
#we scale the data to increase speed and performance


def create_sequences(data, seq_length):
    xs = []
    ys = []

    for i in range(len(data)-seq_length-1):
        x = data[i:(i+seq_length)]
        y = data[i+seq_length]
        xs.append(x)
        ys.append(y)

    return np.array(xs), np.array(ys)
#create sequences to assign x values and y values for train and test
# for example:
# [1,2,3,4]
```

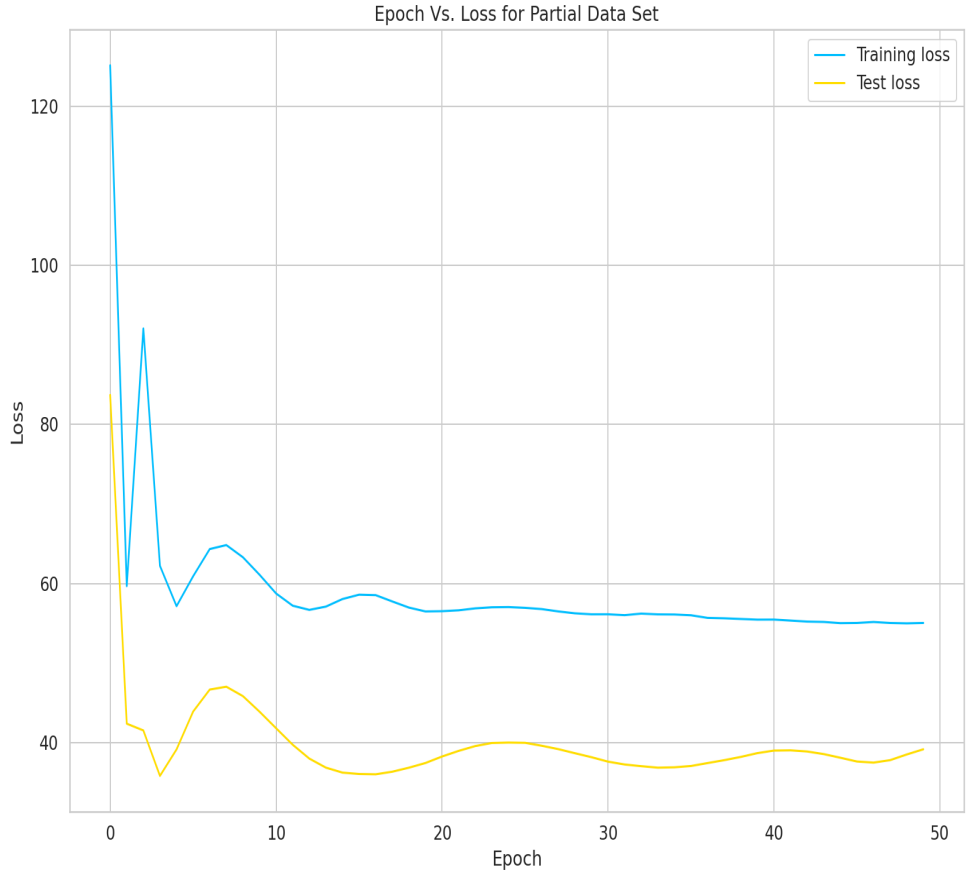
```
# [2,3,4,5]
# for a sequence length of 3 the x values will be: [1,2,3] and [2,3,4]
#the corresponding y values will be [4] and [5]


seq_length = 5
X_train, y_train = create_sequences(train_data, seq_length)
X_test, y_test = create_sequences(test_data, seq_length)

X_train = torch.from_numpy(X_train).float()
y_train = torch.from_numpy(y_train).float()

X_test = torch.from_numpy(X_test).float()
y_test = torch.from_numpy(y_test).float()
#splitting data


#creating our lstm model
class sunspotPredictor(nn.Module):

  def __init__(self, n_features, n_hidden, seq_len, n_layers=2):
    super(sunspotPredictor, self).__init__()

    self.n_hidden = n_hidden
    self.seq_len = seq_len
    self.n_layers = n_layers

    self.lstm = nn.LSTM(
      input_size=n_features,
      hidden_size=n_hidden,
      num_layers=n_layers,
      dropout=0.5
    )

    self.linear = nn.Linear(in_features=n_hidden, out_features=1)

  def reset_hidden_state(self):
    self.hidden = (
        torch.zeros(self.n_layers, self.seq_len, self.n_hidden),
        torch.zeros(self.n_layers, self.seq_len, self.n_hidden)
    )

  def forward(self, sequences):
    lstm_out, self.hidden = self.lstm(
      sequences.view(len(sequences), self.seq_len, -1),
      self.hidden
    )
    last_time_step = \
      lstm_out.view(self.seq_len, len(sequences), self.n_hidden)[-1]
    y_pred = self.linear(last_time_step)
    return y_pred


#helper function to view loss statistics and runn model
def train_model(
  model,
  train_data,
  train_labels,
  test_data=None,
  test_labels=None
):
  loss_fn = torch.nn.MSELoss(reduction='sum')

  optimiser = torch.optim.Adam(model.parameters(), lr=1e-3)
  num_epochs = 50 #CHANGE THIS NUMBER TO SOMETHING BETWEEN 1 AND 10 IF YOU WANT TO WAIT LESS

  train_hist = np.zeros(num_epochs)
  test_hist = np.zeros(num_epochs)

  for t in range(num_epochs):
    model.reset_hidden_state()
```

```
      y_pred = model(X_train)

      loss = loss_fn(y_pred.float(), y_train)

      if test_data is not None:
        with torch.no_grad():
          y_test_pred = model(X_test)
          test_loss = loss_fn(y_test_pred.float(), y_test)
        test_hist[t] = test_loss.item()

        if t % 2 == 0:
          print(f'Epoch {t} train loss: {loss.item()} test loss: {test_loss.item()}')
      elif t % 2 == 0:
        print(f'Epoch {t} train loss: {loss.item()}')

      train_hist[t] = loss.item()

      optimiser.zero_grad()

      loss.backward()

      optimiser.step()

    return model.eval(), train_hist, test_hist


#running model
model = sunspotPredictor(
  n_features=1,
  n_hidden=512,
  seq_len=seq_length,
  n_layers=2
)
model, train_hist, test_hist = train_model(
  model,
  X_train,
  y_train,
  X_test,
  y_test
)
```

```
     Epoch 0 train loss: 125.12258911132812 test loss: 83.70722961425781
     Epoch 2 train loss: 92.0558090209961 test loss: 41.54801940917969
     Epoch 4 train loss: 57.164939880371094 test loss: 39.16176223754883
     Epoch 6 train loss: 64.33879852294922 test loss: 46.682769775390625
     Epoch 8 train loss: 63.30725860595703 test loss: 45.85732650756836
     Epoch 10 train loss: 58.75537872314453 test loss: 41.82322311401367
     Epoch 12 train loss: 56.69623565673828 test loss: 38.01093292236328
     Epoch 14 train loss: 58.058414459228516 test loss: 36.24689483642578
     Epoch 16 train loss: 58.547637939453125 test loss: 36.036895751953125
     Epoch 18 train loss: 56.986732482910156 test loss: 36.86479187011719
     Epoch 20 train loss: 56.531768798828125 test loss: 38.27210998535156
     Epoch 22 train loss: 56.88663864135742 test loss: 39.58380126953125
     Epoch 24 train loss: 57.04829406738281 test loss: 40.02330017089844
     Epoch 26 train loss: 56.797122955322266 test loss: 39.625789642333984
     Epoch 28 train loss: 56.26460266113281 test loss: 38.68706512451172
     Epoch 30 train loss: 56.14307403564453 test loss: 37.626094818115234
     Epoch 32 train loss: 56.226531982421875 test loss: 37.055931091308594
     Epoch 34 train loss: 56.117393493652344 test loss: 36.91493225097656
     Epoch 36 train loss: 55.689918518066406 test loss: 37.44791793823242
     Epoch 38 train loss: 55.5533561706543 test loss: 38.22016525268555
     Epoch 40 train loss: 55.4777717590332 test loss: 39.01694869995117
     Epoch 42 train loss: 55.22487258911133 test loss: 38.89985656738281
     Epoch 44 train loss: 55.029640197753906 test loss: 38.116390228271484
     Epoch 46 train loss: 55.175025939941406 test loss: 37.505531311035156
     Epoch 48 train loss: 55.00346755981445 test loss: 38.52797317504883
```

```
#plotting loss and epochs
plt.plot(train_hist, label="Training loss")
plt.plot(test_hist, label="Test loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
```

```
plt.title("Epoch Vs. Loss for Partial Data Set") #testing accuracy of model against known data
plt.legend();
```



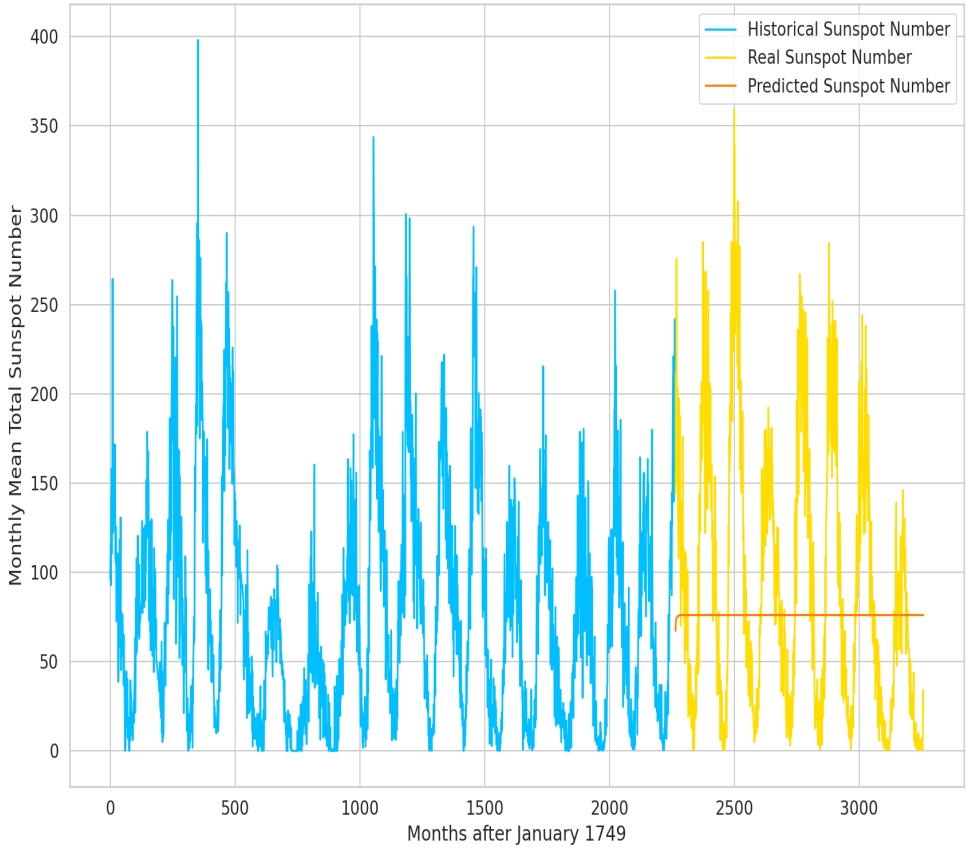

```
#preidicting future values using predicted values as input for next months
with torch.no_grad():
  test_seq = X_test[:1]
  preds = []
  for _ in range(len(X_test)):
    y_test_pred = model(test_seq)
    pred = torch.flatten(y_test_pred).item()
    preds.append(pred)
    new_seq = test_seq.numpy().flatten()
    new_seq = np.append(new_seq, [pred])
    new_seq = new_seq[1:]
    test_seq = torch.as_tensor(new_seq).view(1, seq_length, 1).float()


#reverse it
true_spots = scaler.inverse_transform(
    np.expand_dims(y_test.flatten().numpy(), axis=0)
).flatten()

predicted_sunspots = scaler.inverse_transform(
  np.expand_dims(preds, axis=0)
).flatten()
print(len(predicted_sunspots))
print(len(true_spots))
#print(predicted_cases)
#data2 = data
data2 = np.append(data, predicted_sunspots)
```

```
994
994
```

```
df.index[-1]
```

```
3264
```

```
#comparing lsmt predictions against known data
plt.plot(
  df.index[:len(train_data)],
  scaler.inverse_transform(train_data).flatten(),
  label='Historical Sunspot Number'
)

plt.plot(
  df.index[len(train_data):len(train_data) + len(true_spots)],
  true_spots,
  label='Real Sunspot Number'
)

plt.plot(
  df.index[len(train_data):len(train_data) + len(true_spots)],
  predicted_sunspots,
  label='Predicted Sunspot Number'
)

plt.xlabel("Months after January 1749");
plt.ylabel("Monthly Mean Total Sunspot Number");
plt.legend();
```



Using All Data for Training Now:

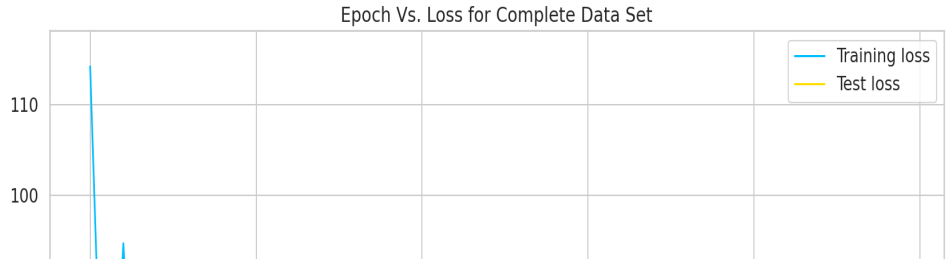
```
#working with complete dataset for training and testing now
scaler = MinMaxScaler()

scaler = scaler.fit(np.expand_dims(data, axis=1))

all_data = scaler.transform(np.expand_dims(data, axis=1))

all_data.shape
```

```
(3265, 1)
```

```
#runnning model
X_all, y_all = create_sequences(all_data, seq_length)

X_all = torch.from_numpy(X_all).float()
y_all = torch.from_numpy(y_all).float()

model = sunspotPredictor(
  n_features=1,
  n_hidden=512,
  seq_len=seq_length,
  n_layers=2
)
model, train_hist, _ = train_model(model, X_all, y_all)
```

```
Epoch 0 train loss: 114.22173309326172
Epoch 2 train loss: 94.72232055664062
Epoch 4 train loss: 57.480812072753906
Epoch 6 train loss: 65.10433959960938
Epoch 8 train loss: 63.63462448120117
Epoch 10 train loss: 58.94943618774414
Epoch 12 train loss: 56.75322723388672
Epoch 14 train loss: 58.46199035644531
Epoch 16 train loss: 58.48960876464844
Epoch 18 train loss: 57.04864501953125
Epoch 20 train loss: 56.60457229614258
Epoch 22 train loss: 57.058719635009766
Epoch 24 train loss: 57.1917724609375
Epoch 26 train loss: 56.89204025268555
Epoch 28 train loss: 56.37528610229492
Epoch 30 train loss: 56.13559341430664
Epoch 32 train loss: 56.366981506347656
Epoch 34 train loss: 56.22993087768555
Epoch 36 train loss: 55.97620391845703
Epoch 38 train loss: 55.685733795166016
Epoch 40 train loss: 55.696571350097656
Epoch 42 train loss: 55.42679977416992
Epoch 44 train loss: 55.12117004394531
Epoch 46 train loss: 55.213077545166016
Epoch 48 train loss: 55.12204360961914
```

```
plt.plot(train_hist, label="Training loss")
plt.plot(test_hist, label="Test loss")
plt.xlabel("Epoch")
plt.ylabel("Loss")
plt.title("Epoch Vs. Loss for Complete Data Set") #testing accuracy of model with all data as training data
plt.legend();
```




```
#using newly trained model to predict 100 months in future using same procedure
MONTHS_TO_PREDICT = 100

with torch.no_grad():
  test_seq = X_all[:1]
  preds = []
  for _ in range(MONTHS_TO_PREDICT):
    y_test_pred = model(test_seq)
    pred = torch.flatten(y_test_pred).item()
    preds.append(pred)
    new_seq = test_seq.numpy().flatten()
    new_seq = np.append(new_seq, [pred])
    new_seq = new_seq[1:]
    test_seq = torch.as_tensor(new_seq).view(1, seq_length, 1).float()
```

```
predicted_sunspots = scaler.inverse_transform(
  np.expand_dims(preds, axis=0)
).flatten()
```

```
#plotting future predictions and historical data
data2 = np.append(data, predicted_sunspots)
plt.plot(data2, label='Predicted Sunspots')
plt.plot(data, label='Historical Sunspots')
plt.xlabel("Months after January 1749");
plt.ylabel("Monthly Mean Total Sunspot Number");
plt.title("Predicted Sunspots")
plt.legend();
```

Predicted Sunspots

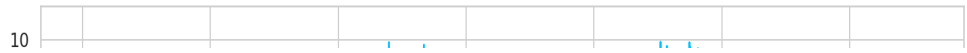
Creating Calculated Values using Sunspot Equation

```
#importing values from files for monthly sunspot count NOT numbers as above
monthly_mean_sunspot_count = []

with open('monthly_mean_sunspot_count.csv') as csvfile:
  reader = csv.reader(csvfile, delimiter=';')
  for row in reader:
    monthly_mean_sunspot_count.append(float(row[3]))

print(monthly_mean_sunspot_count)
#plt.plot(monthly_mean_sunspot_count)
#plt.yticks(monthly_mean_sunspot_count[::1000])
#plt.show()
```

```
[96.7, 104.3, 116.7, 92.8, 141.7, 139.2, 158.0, 110.5, 126.5, 125.8, 264.3, 142.0, 122.2, 126.5, 148.7, 147.2, 150.
```

```
monthly_mean_group_number = []
x = []

for i in range(1104):
  monthly_mean_group_number.append(1)

with open('monthly_mean_group_number.txt') as txtfile:
  reader = csv.reader(txtfile, delimiter=' ')
  for row in reader:
    monthly_mean_group_number.append(float(row[4]))

for i in range(547):
  monthly_mean_group_number.append(1)

print(len(monthly_mean_group_number))

for i in range(len(monthly_mean_group_number)):
  x.append(i)

plt.plot(x, monthly_mean_group_number)
#plt.yticks(monthly_mean_group_number[::500])
```

```
3283
[<matplotlib.lines.Line2D at 0x7fac01c3f550>]
```




```
def calculate(sunspot, group):
  return ((10*group) + sunspot)
#sunspot equation: R = k(10g + s)
```


```
#comparing calculated sunspots to lstm predictions...
sunspot_number = []
time = []

for i in range(len(monthly_mean_group_number)):
  res = calculate(monthly_mean_sunspot_count[i], monthly_mean_group_number[i])
  sunspot_number.append(res)

for i in range(3283):
  time.append(i)

plt.plot(data2, label='Predicted Sunspots')
plt.plot(data, label='Historical Sunspots')
plt.xlabel("Months after January 1749");
plt.ylabel("Monthly Mean Total Sunspot Number");
plt.title("Predicted Sunspots")

plt.plot(time, sunspot_number, label = 'Calculated using Sunspot Eqn')
plt.legend();
```