Neural Network

November 9, 2021

1 Libraries

Import your libraries https://curiousily.com/posts/build-your-first-neural-network-with-pytorch/

[1]:
```
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.utils import resample
from sklearn.model_selection import train_test_split
from sklearn.metrics import confusion_matrix, classification_report
from torch import nn, optim
import torch.nn.functional as F
```

[2]:
```
!nvidia-smi
```

```
Tue Nov  9 14:31:26 2021
+-----------------------------------------------------------------------------+
| NVIDIA-SMI 470.63.01    Driver Version: 470.63.01    CUDA Version: 11.4     |
|-------------------------------+----------------------+----------------------+
| GPU  Name        Persistence-M| Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp  Perf  Pwr:Usage/Cap|         Memory-Usage | GPU-Util  Compute M. |
|                               |                      |               MIG M. |
|===============================+======================+======================|
|   0  NVIDIA GeForce …  Off  | 00000000:05:00.0 Off |                  N/A |
| 29%   26C    P8    18W / 250W |   3398MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   1  NVIDIA GeForce …  Off  | 00000000:09:00.0 Off |                  N/A |
| 30%   27C    P8    20W / 250W |   8560MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   2  NVIDIA GeForce …  Off  | 00000000:0A:00.0 Off |                  N/A |
```

```
| 30%   26C    P8     3W / 250W |   7976MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   3  NVIDIA GeForce …  Off  | 00000000:85:00.0 Off |                  N/A |
| 29%   26C    P8     2W / 250W |      3MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+
|   4  NVIDIA GeForce …  Off  | 00000000:89:00.0 Off |                  N/A |
| 29%   25C    P8     1W / 250W |      3MiB / 11019MiB |      0%      Default |
|                               |                      |                  N/A |
+-------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------+
| Processes:                                                                  |
|  GPU   GI   CI        PID   Type   Process name                  GPU Memory |
|        ID   ID                                                   Usage      |
|=============================================================================|
|    0   N/A  N/A      6012      C   …erhub/anaconda/bin/python     1013MiB |
|    0   N/A  N/A      6155      C   …erhub/anaconda/bin/python     1271MiB |
|    0   N/A  N/A     19260      C   …erhub/anaconda/bin/python     1111MiB |
|    1   N/A  N/A     35000      C   …erhub/anaconda/bin/python     8557MiB |
|    2   N/A  N/A     32596      C   …erhub/anaconda/bin/python     7973MiB |
+-----------------------------------------------------------------------------+
```

2 Importing Data

Import the CSV file with Actions, Sum and Div as a Dataframe called df. Fill the empty values of Action with 0. Replace NaN values with 0. Delete first 100 rows.

[3]:
```
#load in df
Player = 15
Game = 2

df = pd.read_csv('matrix_Player_' + str(Player) + '_game_' + str(Game) +␣
 ↪'_Processed_Action.csv')

#Delete first 99 rows, so df starts at timeLine == 1.00
df = df.iloc[99: , :]

df.head()
```

[3]:

	Unnamed: 0	frAcc	frRoAcc	frDispl	frRoAng	frSpeed	timeLine	\
99	99	0.0	0.0	0.0	0.0	0.0	1.00	
100	100	0.0	0.0	0.0	0.0	0.0	1.01	
101	101	0.0	0.0	0.0	0.0	0.0	1.02	
102	102	0.0	0.0	0.0	0.0	0.0	1.03	

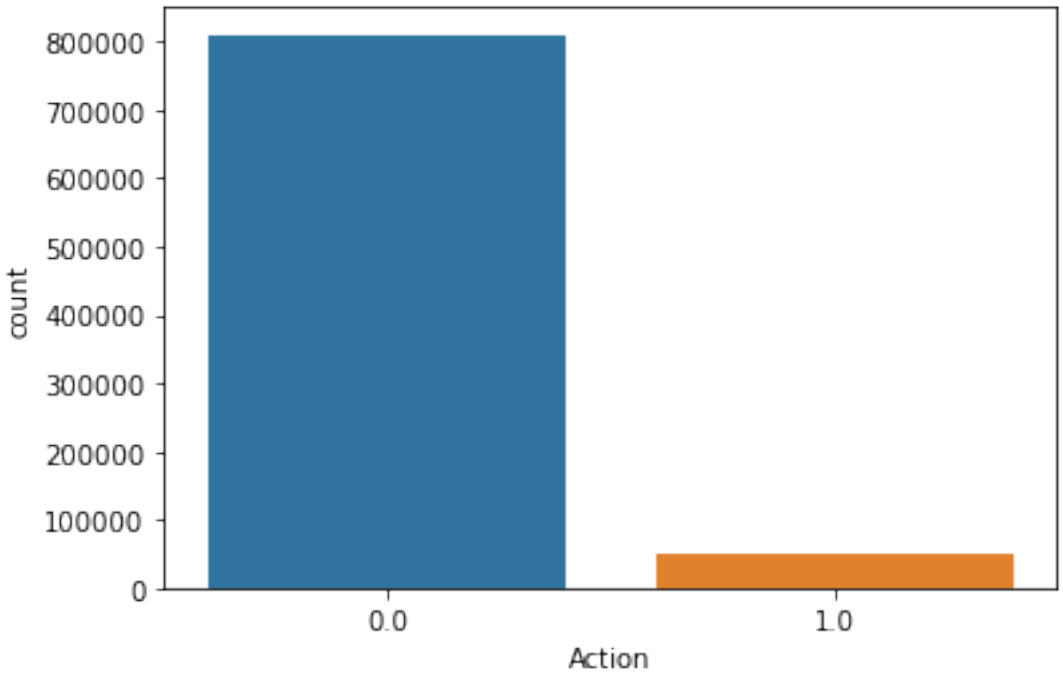
```
103         103    0.0      0.0      0.0      0.0      0.0      1.04

     frameRotationalSpeedX  frameRotationalSpeedY  frameRotationalSpeedZ  \
99                     NaN                    NaN                    NaN
100                    NaN                    NaN                    NaN
101                    NaN                    NaN                    NaN
102                    NaN                    NaN                    NaN
103                    NaN                    NaN                    NaN

     wheelRotationalSpeedX  wheelRotationalSpeedY  wheelRotationalSpeedZ  \
99                     NaN                    NaN                    NaN
100                    NaN                    NaN                    NaN
101                    NaN                    NaN                    NaN
102                    NaN                    NaN                    NaN
103                    NaN                    NaN                    NaN

     frRoSpeed  Sum_WheelX_FrameZ  Div_FrameZ_WheelX  Filt_WheelX  \
99         0.0                NaN                NaN          0.0
100        0.0                NaN                NaN          0.0
101        0.0                NaN                NaN          0.0
102        0.0                NaN                NaN          0.0
103        0.0                NaN                NaN          0.0

     Filt_FrameZ  Action
99           0.0     NaN
100          0.0     NaN
101          0.0     NaN
102          0.0     NaN
103          0.0     NaN
```

3 Pre Preprocessing

[4]:
```
cols = ['wheelRotationalSpeedX','frameRotationalSpeedY','frAcc','Action']

df = df[cols]

# Fill NaN with 0
df.replace([np.inf, -np.inf], np.nan, inplace=True)
df = df.fillna(0)

#Convert Sprinting to 1
df.Action.replace({'Sprinting': 1},inplace=True)

sns.countplot(df['Action'])
```

```
/opt/jupyterhub/anaconda/lib/python3.9/site-packages/seaborn/_decorators.py:36:
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[4]: <AxesSubplot:xlabel='Action', ylabel='count'>

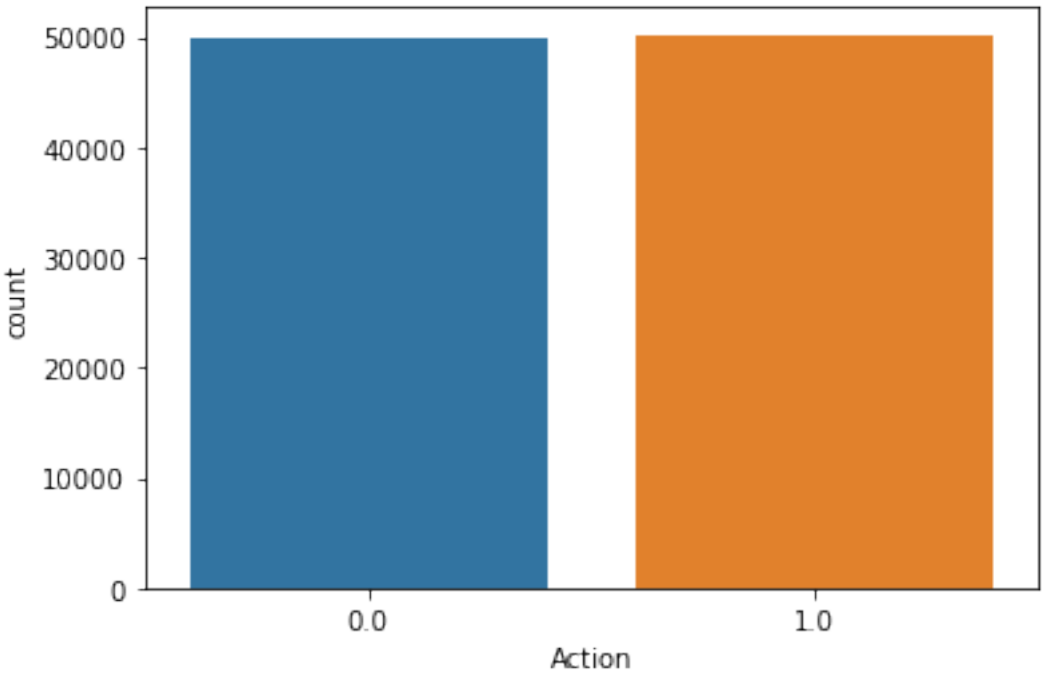


[5]:
```
# Separate majority and minority classes
df_majority = df[df.Action==0]
df_minority = df[df.Action==1]

# Downsample majority class
df_majority_downsampled = resample(df_majority,
                                   replace=False,    # sample without replacement
                                   n_samples=50000,     # to match minority class
                                   random_state=123) # reproducible results

# Combine minority class with downsampled majority class
df = pd.concat([df_majority_downsampled, df_minority])

sns.countplot(df['Action'])
```

```
/opt/jupyterhub/anaconda/lib/python3.9/site-packages/seaborn/_decorators.py:36:
```

```
FutureWarning: Pass the following variable as a keyword arg: x. From version
0.12, the only valid positional argument will be `data`, and passing other
arguments without an explicit keyword will result in an error or
misinterpretation.
  warnings.warn(
```

[5]:
```
<AxesSubplot:xlabel='Action', ylabel='count'>
```



[6]:
```
X = df[['wheelRotationalSpeedX','frameRotationalSpeedY','frAcc']]
y = df[['Action']]

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,␣
↪random_state=42)
```

Convert the X_train, X_test, y_train, y_test to Tensors

[7]:
```
X_train = torch.from_numpy(X_train.to_numpy()).float()
y_train = torch.squeeze(torch.from_numpy(y_train.to_numpy()).float())
X_test = torch.from_numpy(X_test.to_numpy()).float()
y_test = torch.squeeze(torch.from_numpy(y_test.to_numpy()).float())
print(X_train.shape, y_train.shape)
print(X_test.shape, y_test.shape)
```

```
torch.Size([80152, 3]) torch.Size([80152])
torch.Size([20039, 3]) torch.Size([20039])
```

4 NN def

Define the Neural Network

[8]:
```
class NN_model(nn.Module):
  def __init__(self, n_features):
    super(NN_model, self).__init__()
    self.fc1 = nn.Linear(n_features, first)
    self.fc2 = nn.Linear(first, second)
    self.fc3 = nn.Linear(second, 1)

  def forward(self, x):
    x = F.relu(self.fc1(x))
    x = F.relu(self.fc2(x))
    return torch.sigmoid(self.fc3(x))

first = 50
second = 10

NN = NN_model(X_train.shape[1])
```

5 Training the NN on the GPU

[9]:
```
criterion = nn.BCELoss()

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

device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
X_train = X_train.to(device)
y_train = y_train.to(device)
X_test = X_test.to(device)
y_test = y_test.to(device)
NN = NN.to(device)
criterion = criterion.to(device)
```

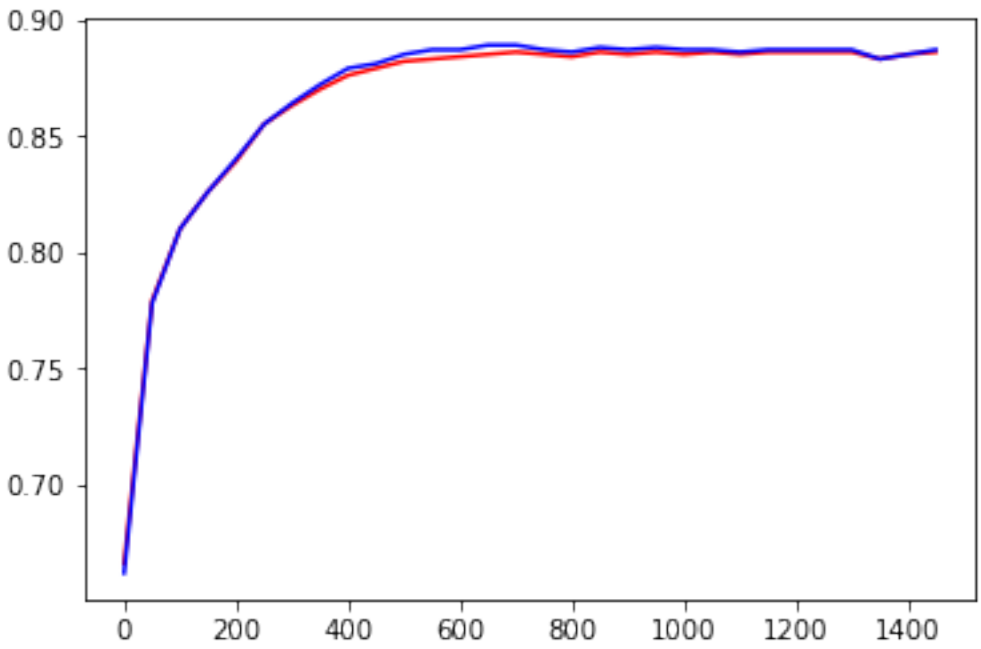
6 Define functions for calculating Accuracy of model

[10]:
```
def calculate_accuracy(y_true, y_pred):
    predicted = y_pred.ge(.5).view(-1)
    return (y_true == predicted).sum().float() / len(y_true)
```

7 For loop through different epochs

[11]:
```
def round_tensor(t, decimal_places=3):
  return round(t.item(), decimal_places)

Results = pd.DataFrame(columns = ['Epoch', 'Acc_train', 'Acc_test'])

for epoch in range(1500):
        y_pred = NN(X_train)
        y_pred = torch.squeeze(y_pred)
        train_loss = criterion(y_pred, y_train)

        if epoch % 50 == 0:

            train_acc = calculate_accuracy(y_train, y_pred)

            y_test_pred = NN(X_test)

            y_test_pred = torch.squeeze(y_test_pred)

            test_loss = criterion(y_test_pred, y_test)

            test_acc = calculate_accuracy(y_test, y_test_pred)

            result = {'Epoch': epoch, 'Acc_train': round_tensor(train_acc),␣
 ↪'Acc_test': round_tensor(test_acc)}

            Results = Results.append(result, ignore_index=True)

        #print(
#f'''epoch {epoch}
#Train set - loss: {round_tensor(train_loss)}, accuracy:␣
 ↪{round_tensor(train_acc)}
#Test  set - loss: {round_tensor(test_loss)}, accuracy: {round_tensor(test_acc)}
#''')
        optimizer.zero_grad()

        train_loss.backward()

        optimizer.step()
```

[12]:
```
plt.plot(Results.Epoch,Results.Acc_train,'r',Results.Epoch,Results.Acc_test,'b')
plt.show()
```



8 Validate/Tune Model

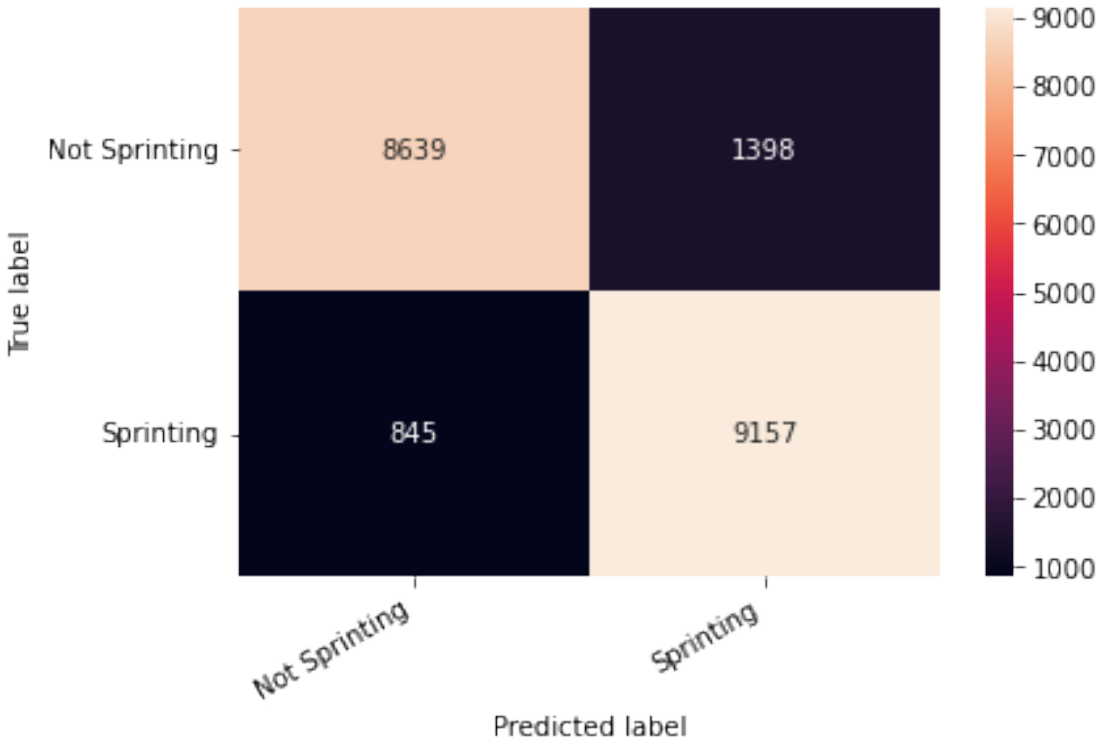
Validate results of the model (Precision/Recall). Tune the parameters of the model to achieve better results

[13]:
```
classes = ['Not Sprinting', 'Sprinting']
y_pred = NN(X_test)
y_pred = y_pred.ge(.5).view(-1).cpu()
y_test = y_test.cpu()
print(classification_report(y_test, y_pred, target_names=classes))
```

```
              precision    recall  f1-score   support

Not Sprinting       0.91      0.86      0.89     10037
    Sprinting       0.87      0.92      0.89     10002

     accuracy                           0.89     20039
    macro avg       0.89      0.89      0.89     20039
 weighted avg       0.89      0.89      0.89     20039
```

[14]:
```
cm = confusion_matrix(y_test, y_pred)
df_cm = pd.DataFrame(cm, index=classes, columns=classes)
hmap = sns.heatmap(df_cm, annot=True, fmt="d")
```

```
hmap.yaxis.set_ticklabels(hmap.yaxis.get_ticklabels(), rotation=0, ha='right')
hmap.xaxis.set_ticklabels(hmap.xaxis.get_ticklabels(), rotation=30, ha='right')
plt.ylabel('True label')
plt.xlabel('Predicted label');
```



9 Saving Model

MODEL_PATH = 'model.pth'

torch.save(NN, MODEL_PATH)

10 Restoring Model

NN = torch.load(MODEL_PATH)