In [2]:

```python
import numpy as np
import pandas as pd
pd.options.display.max_columns=100
from sklearn.model_selection import train_test_split, cross_val_score, cross
import sklearn.metrics
from sklearn.preprocessing import StandardScaler as SSc
import torch
from torch import nn, optim
from torch.autograd import Variable
import torch.nn.functional as F
from torch.utils.data import TensorDataset, DataLoader
import matplotlib.pyplot as plt
%matplotlib inline

#set width of window to preference
from IPython.core.display import display, HTML
display(HTML("<style>.container { width:90% !important; }</style>"))
```

In [3]:

```python
data = pd.read_csv("Data-Prepped.csv",index_col=0)
data = data.astype(np.float32)
data.head()
```

Out[3]:

	Bronze	Silver	Gold	Platinum	Diamond	Master	GrandMaster	LeagueIndex	Age	HoursPerWe
0	0.0	0.0	0.0	0.0	1.0	0.0	0.0	5.0	27.0	1
1	0.0	0.0	0.0	0.0	1.0	0.0	0.0	5.0	23.0	1
2	0.0	0.0	0.0	1.0	0.0	0.0	0.0	4.0	30.0	1
3	0.0	0.0	1.0	0.0	0.0	0.0	0.0	3.0	19.0	2
4	0.0	0.0	1.0	0.0	0.0	0.0	0.0	3.0	32.0	1

In [5]:

```python
X = data.iloc[:,data.columns != 'APM']
Y = data.iloc[:,data.columns == 'APM']
#transform input data (normalize scaling)
ssc = SSc()
Xft = ssc.fit_transform(X)
X = pd.DataFrame(Xft)
print("Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes: ")
Xtr,Xtst,Ytr,Ytst = train_test_split(X,Y.values.ravel(),test_size=0.2,random
print(Xtr.shape,Xtst.shape,Ytr.shape,Ytst.shape)
Ytr = pd.DataFrame(Ytr)
```

```
Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes:
(2670, 25) (668, 25) (2670,) (668,)
```

In [6]:

```
x_tr = torch.tensor(Xtr.values.astype(np.float64))
y_tr = torch.tensor(Ytr.values.astype(np.float32))
x_tst = torch.tensor(Xtst.values.astype(np.float64))
y_tst = torch.tensor(Ytst.astype(np.float32))

btch_sz = 20

tr_dset = TensorDataset(x_tr,y_tr)
tr_dload = DataLoader(dataset=tr_dset, batch_size=btch_sz, shuffle=True)

tst_dset = TensorDataset(x_tst,y_tst)
tst_dload = DataLoader(dataset=tst_dset, batch_size=btch_sz)
```

In [7]:

```python
'''
class Swish(torch.autograd.Function):
    @staticmethod
    def forward(ctx, i):
        result = i * torch.sigmoid(i)
        ctx.save_for_backward(i)
        return result

    @staticmethod
    def backward(ctx, grad_output):
        i = ctx.saved_variables[0]
        sigmoid_i = torch.sigmoid(i)
        return grad_output * (sigmoid_i * (1 + i * (1 - sigmoid_i)))

class swish(nn.Module):
    def forward(self, input_tensor):
        return Swish.apply(input_tensor)
'''


class resultNet(nn.Module):
    def __init__(self, X_sz, Y_sz, a=0, b=0, c=0, d=0):
        super(resultNet, self).__init__()

        self.inputSize = len(X.columns)
        self.outputSize = len(Y.columns)
        self.hidden0Size = a
        self.hidden1Size = b
        self.hidden2Size = c
        self.hidden3Size = d
        self.activation = F.selu
        self.outactivation = F.selu
        self.outsquish = torch.sigmoid

        #Connect network
        self.dpth = 0
        if (self.hidden0Size != 0):
            self.c1 = nn.Linear(self.inputSize,self.hidden0Size)
            self.dpth += 1
            print("adding layer 1")
            if (self.hidden1Size != 0):
                self.c2 = nn.Linear(self.hidden0Size,self.hidden1Size)
                self.dpth += 1
                print("adding layer 2")
                if (self.hidden2Size != 0):
                    self.c3 = nn.Linear(self.hidden1Size,self.hidden2Size)
                    self.dpth += 1
                    print("adding layer 3")
                    if (self.hidden3Size != 0):
                        self.c4 = nn.Linear(self.hidden2Size,self.hidden3Siz
                        self.dpth += 1
                        print("adding layer 4")
                        self.c5 = nn.Linear(self.hidden3Size,self.outputSize
                    else:
                        self.c4 = nn.Linear(self.hidden2Size,self.outputSize
                else:
                    self.c3 = nn.Linear(self.hidden1Size,self.outputSize)
            else:
                self.c2 = nn.Linear(self.hidden0Size,self.outputSize)
        else:
            self.c1 = nn.Linear(self.inputSize,self.outputSize)

    def forward(self, x):

        if (self.dpth == 0):
            out = self.outsquish(self.outactivation(self.c1(x)))
            #print("fwd dpth 0")
        elif (self.dpth == 1):
            x = self.activation(self.c1(x))
            out = self.outsquish(self.outactivation(self.c2(x)))
            #print("fwd dpth 1")
        elif (self.dpth == 2):
            x = self.activation(self.c1(x))
            x = self.activation(self.c2(x))
            out = self.outsquish(self.outactivation(self.c3(x)))
            #print("fwd dpth 2")
        elif (self.dpth == 3):
            x = self.activation(self.c1(x))
            x = self.activation(self.c2(x))
            x = self.activation(self.c3(x))
            out = self.outsquish(self.outactivation(self.c4(x)))
            #print("fwd dpth 3")
        elif (self.dpth == 4):
            x = self.activation(self.c1(x))
            x = self.activation(self.c2(x))
            x = self.activation(self.c3(x))
            x = self.activation(self.c4(x))
            out = self.outsquish(self.outactivation(self.c5(x)))
            #print("fwd dpth 4")
        return out

h_size = 12
testNet = resultNet(len(Xtr.columns),len(Ytr.columns),25)
for p in testNet.parameters():
    print(p)
```

```
adding layer 1
Parameter containing:
tensor([[-4.9726e-02,  7.2450e-02, -1.2798e-02,  6.9392e-02,  4.2379e-03,
          1.5457e-01, -8.1887e-02,  4.7553e-02, -1.8418e-01, -6.9317e-02,
          4.4955e-02,  3.7756e-02, -1.7748e-01,  1.6583e-01,  6.9771e-02,
         -1.5166e-01, -5.2824e-02, -1.4619e-01, -1.2801e-01, -6.1047e-02,
         -1.0535e-01, -1.3497e-01, -4.7729e-02, -4.6701e-03, -8.0913e-02],
        [-1.9027e-02, -1.8105e-01,  1.6733e-01, -8.6210e-02,  1.3645e-01,
         -5.0487e-02, -1.0134e-02, -1.2486e-01, -5.8060e-02,  3.8344e-02,
          1.4308e-03,  1.4626e-01,  6.9895e-02, -1.5107e-01, -1.4297e-01,
          2.5415e-03, -1.0102e-01,  1.9226e-01,  8.1118e-02, -1.3433e-01,
         -6.0465e-02,  2.0874e-02,  8.3659e-02, -1.3880e-01, -2.6513e-02],
        [-6.5117e-02, -5.8546e-02,  9.3555e-02, -6.2392e-02,  4.5130e-02,
         -1.6837e-01, -1.7615e-01,  1.4988e-01,  1.7800e-01, -1.5659e-01,
         -8.7009e-02, -1.3793e-01, -9.8461e-02, -6.4829e-02,  1.7095e-01,
         -1.1134e-01,  1.6048e-01, -1.1313e-02,  1.6490e-01,  4.3963e-02,
         -1.4891e-01,  1.8135e-01, -1.0309e-01, -1.8124e-01,  1.0441e-01],
        [ 9.3367e-02,  1.1437e-01,  4.0472e-02, -7.7403e-02,  1.5619e-01,
 -4.5427e-02, -9.2631e-02,  1.6182e-01, -1.6522e-01, -2.4257e-02,
 -1.9721e-01,  7.8905e-02, -1.4422e-01,  1.0654e-01,  8.8935e-02,
 -1.5538e-01,  8.6732e-02, -8.4107e-02, -1.4852e-01,  2.9176e-02,
 -1.7694e-01, -1.9953e-02, -1.2150e-01,  7.3786e-02, -1.0623e-01],
[-8.3896e-02, -1.9690e-01, -1.5250e-01,  8.1280e-02,  6.3132e-02,
 -2.2535e-02, -5.2225e-02,  6.2031e-03, -2.8253e-05, -1.8381e-01,
  1.6160e-01, -1.2606e-01, -1.3396e-01, -6.5983e-03, -8.7718e-02,
  1.5692e-01,  8.4878e-02, -1.3181e-03,  1.9050e-01,  1.2335e-01,
  1.4841e-01, -2.0509e-02,  1.3392e-01, -1.3219e-01,  9.4986e-03],
[-5.6178e-02,  2.0543e-02, -1.0369e-01, -4.1768e-03,  1.6970e-01,
  1.2345e-01, -1.0315e-02, -6.1518e-02,  1.1414e-01, -2.6514e-02,
  5.8703e-02, -1.4733e-01, -1.2215e-01,  3.2763e-02, -1.5461e-01,
  1.1840e-01, -3.0028e-02, -5.5495e-02, -1.8380e-01,  9.4733e-02,
  9.5917e-02, -1.6660e-01, -6.3243e-02,  7.0572e-02, -1.9957e-01],
[-2.6598e-02, -8.9574e-02,  1.2658e-01, -1.2637e-01,  9.6542e-02,
 -1.0682e-01,  5.6621e-02,  1.8323e-01,  1.7365e-01,  1.9963e-01,
 -1.1993e-01,  1.2030e-01, -1.0219e-01,  1.6713e-01,  7.3980e-02,
  4.3735e-04,  1.0706e-01, -4.2136e-02, -1.4037e-01,  3.3892e-02,
 -1.9478e-01, -1.5558e-01,  6.1862e-02, -1.8208e-01, -1.0812e-01],
[ 6.9980e-02, -1.1405e-01,  1.2093e-01,  1.1315e-01,  1.4914e-01,
 -4.6804e-02,  1.4379e-01,  2.2850e-02, -1.9405e-01, -5.3148e-02,
  1.7642e-01, -5.7664e-02,  6.7478e-02, -7.1075e-02, -4.5036e-02,
 -1.6889e-01, -1.7813e-01, -1.5274e-01, -1.6184e-01, -5.9396e-02,
 -8.4732e-02, -5.4343e-02, -3.0810e-02, -1.5192e-01, -1.4946e-01],
[ 1.5694e-01, -9.8477e-03, -1.8140e-01, -5.0209e-02,  2.3250e-02,
 -2.0589e-02, -1.9677e-01, -1.2507e-02, -1.3158e-01,  1.6407e-01,
  1.9821e-01,  1.8981e-01,  1.7687e-01,  1.9926e-02,  1.8099e-01,
  5.6629e-02,  1.5627e-01,  1.6251e-01,  5.8097e-02, -1.2044e-01,
 -1.0063e-01,  6.8739e-02, -1.2935e-02,  1.6080e-01,  1.4027e-01],
[ 1.6004e-01, -2.1858e-02, -2.1213e-02, -1.6898e-01,  1.2983e-01,
  1.4639e-02, -2.3076e-02,  1.7536e-04,  1.7117e-01, -1.1233e-01,
  1.1800e-01, -2.2909e-02,  1.0469e-02, -4.2951e-02, -7.1457e-02,
  6.5126e-02,  2.0020e-02, -1.4266e-01, -3.6146e-02,  2.6456e-02,
  7.0626e-03, -9.8886e-02,  5.4128e-02, -1.8679e-01, -1.9385e-01],
[-6.6501e-02,  1.3836e-01,  4.6624e-03,  1.7313e-01, -6.6045e-02,
  2.1509e-02,  1.6112e-01, -3.2690e-02, -1.2574e-01,  2.4311e-02,
 -1.9818e-01,  1.5131e-01,  1.0274e-01,  9.1943e-02, -1.7211e-01,
  1.2953e-01, -8.3247e-03,  1.0696e-01,  1.1110e-01, -7.6865e-02,
 -1.9210e-01,  1.3055e-01,  2.0542e-02, -1.3203e-01,  2.3964e-02],
[ 1.0099e-01, -1.4394e-01, -1.8909e-01,  1.3681e-01, -1.2464e-01,
 -5.3568e-02, -1.2551e-01, -1.4678e-02, -1.8795e-02,  1.7816e-01,
  5.1939e-02, -2.9060e-02, -1.0036e-01,  9.2935e-02, -3.9390e-02,
 -1.8451e-01,  8.8700e-02,  1.1135e-01, -3.0405e-02, -1.7822e-02,
 -1.3655e-01, -9.3165e-02, -3.1783e-02, -5.7297e-02,  5.2184e-03],
[ 1.9027e-01,  1.0206e-01, -1.8198e-01,  9.0239e-02,  1.2954e-01,
 -5.8924e-02, -1.1975e-01,  4.1764e-02,  7.5705e-02,  1.2268e-01,
 -1.4117e-01,  6.5761e-02, -1.8869e-01, -3.7705e-02,  9.7864e-02,
 -8.5139e-02,  7.8417e-02,  1.1561e-01,  8.9269e-03, -8.5471e-02,
  6.0019e-02, -1.3770e-01,  1.7942e-01, -4.5300e-02, -1.5361e-02],
[ 3.5728e-02, -1.4435e-01, -1.4702e-01, -2.7942e-02, -6.6047e-02,
  1.6253e-01, -1.5193e-01, -4.2620e-02, -3.7562e-02,  4.9141e-03,
  4.8050e-02,  1.4703e-02, -1.2934e-01,  1.9297e-01, -2.5615e-02,
 -1.0941e-01,  4.5162e-02, -7.3983e-02,  1.8864e-01, -8.1243e-03,
 -1.4694e-01,  8.7730e-02, -9.5069e-02,  1.5918e-01, -1.3380e-01],
[ 5.9872e-02,  5.0369e-02,  6.8324e-02, -1.3551e-03,  3.3911e-02,
 -8.4457e-02,  7.4450e-02,  5.6175e-02,  8.5864e-03, -6.0270e-02,
 -1.1889e-01, -1.0296e-01,  3.3603e-02,  5.7360e-02,  1.4184e-01,
          -3.6037e-03, -1.1368e-01, -1.2160e-01,  1.3787e-01,  1.2960e-01,
          -2.9712e-02,  1.1078e-01, -1.4296e-01,  1.7076e-01, -1.2129e-01],
        [-1.7988e-01, -6.7554e-02,  3.5499e-02,  8.0318e-02,  1.8235e-01,
          1.5061e-02, -7.3728e-02, -7.6322e-02,  1.2350e-01,  7.2003e-02,
          3.3107e-02, -2.9712e-02, -1.8075e-01, -3.1826e-02,  1.5862e-01,
          1.2187e-01, -2.5763e-02,  2.9106e-02, -5.5254e-03, -1.1331e-02,
          1.8827e-01, -1.3439e-01,  1.6036e-01, -1.7546e-01,  1.8753e-02],
        [-2.1183e-02,  1.1973e-01,  1.9757e-01, -1.0325e-01,  9.7311e-02,
         -3.8571e-02, -1.0744e-01, -7.8720e-03, -1.7611e-01,  1.0195e-01,
         -5.6791e-02,  1.1572e-01,  1.0981e-01,  1.2097e-01, -2.9717e-02,
          4.9454e-02, -8.5520e-02,  2.0730e-03,  1.3108e-02, -1.5291e-01,
         -1.8707e-01, -2.8657e-03,  1.1246e-01,  2.9373e-02, -8.8397e-03],
        [ 1.0829e-01,  7.5640e-02,  7.9466e-02, -3.3456e-02,  6.0217e-02,
         -1.6933e-01,  5.5247e-02, -6.0667e-02,  1.5435e-02,  1.7024e-01,
          1.5971e-01,  1.6483e-01,  1.8779e-01,  1.4586e-02,  1.8057e-01,
          2.0173e-02,  1.3086e-01,  2.3562e-02,  6.3787e-02, -3.0604e-02,
         -1.4998e-01,  2.1211e-02, -6.8725e-02,  7.5709e-02,  4.0058e-02],
        [ 7.3223e-02, -8.2749e-03,  1.5512e-01, -8.9979e-02,  1.8165e-01,
         -1.0621e-02, -1.8014e-01,  3.0187e-02,  2.0492e-02,  6.5284e-02,
          2.7864e-02, -6.9564e-03, -1.0343e-01, -1.4312e-01, -1.2055e-01,
         -1.4899e-01,  1.3283e-01, -1.8422e-01,  2.5360e-02, -7.1377e-02,
          1.6692e-01,  2.8694e-02,  6.3054e-02, -1.0352e-01,  3.0035e-02],
        [ 7.9126e-02,  1.7494e-01, -2.8201e-02, -5.8252e-02, -1.8056e-01,
          1.0945e-01,  1.5930e-01, -1.0172e-01,  6.1010e-02, -9.8125e-02,
         -1.8588e-01, -1.8094e-01, -1.6226e-01,  3.2213e-02, -1.6620e-02,
         -7.0988e-03,  1.7726e-01, -9.0794e-02,  8.0767e-02, -1.5633e-01,
          9.6780e-02,  1.7226e-01, -1.6501e-01,  1.9576e-01, -3.8552e-02],
        [ 7.2448e-02,  1.1377e-01,  7.5516e-02,  5.7062e-03,  2.9235e-02,
          2.3766e-02, -9.2294e-02, -1.5419e-01, -7.4455e-03, -1.3624e-01,
          1.1480e-01,  4.5473e-02, -8.2256e-02, -1.2419e-01,  8.2068e-02,
          3.8314e-02,  1.2421e-02,  1.7014e-01, -8.8892e-02, -1.6399e-01,
          1.7885e-02, -6.5746e-02,  5.9393e-02, -6.7934e-02,  1.6929e-02],
        [ 4.9501e-02,  4.7415e-02,  3.5440e-02,  1.2602e-01, -1.0344e-01,
          7.3622e-02,  1.7213e-01, -1.8428e-01, -7.9557e-02, -1.3440e-01,
         -8.9407e-02, -1.9046e-01, -1.3240e-01,  1.8474e-01, -5.9592e-02,
          1.6687e-01,  1.7235e-01, -2.5301e-02, -1.9441e-01,  9.6408e-02,
          1.5312e-01,  1.4971e-01,  1.3069e-01,  1.2851e-01, -4.8243e-02],
        [-5.9204e-02,  1.6375e-01,  1.3253e-01,  1.2904e-01,  1.0376e-01,
         -5.5826e-02,  3.4106e-02,  1.6267e-01, -1.7492e-01,  1.3794e-01,
         -1.7415e-01,  4.8067e-02, -1.5267e-01, -1.2204e-01,  1.6415e-01,
         -8.6648e-02, -1.2976e-01,  1.5013e-01, -7.4827e-02, -1.0655e-01,
          1.3625e-01,  8.0267e-02,  1.4857e-01, -2.6532e-02,  1.7689e-01],
        [-1.3446e-01, -1.5251e-01,  1.2839e-02,  6.3518e-02, -1.2462e-01,
          1.4922e-01, -1.8389e-01, -1.4673e-02, -1.1384e-01, -9.0172e-02,
          1.6069e-01,  1.8548e-02, -1.2511e-01, -4.0951e-02,  1.0248e-01,
         -1.0560e-01,  1.0589e-01,  1.7890e-01, -4.0324e-02, -1.3308e-01,
          2.0105e-02, -7.0236e-03, -8.2950e-02, -4.0359e-02, -1.9940e-02],
        [-1.6024e-02,  1.4258e-01, -1.5780e-01, -1.0095e-01,  9.8480e-02,
          1.4795e-01, -2.2709e-02,  1.9031e-01,  4.1711e-03,  1.9988e-01,
         -1.5515e-01,  1.8523e-02,  9.3929e-02,  8.4581e-02,  6.8450e-03,
         -9.4609e-03,  1.6125e-01,  1.7621e-01,  3.9856e-02,  1.7383e-01,
         -1.4914e-01,  1.5930e-01,  9.3127e-02,  1.5983e-01,  1.2004e-01]],
       requires_grad=True)
Parameter containing:
tensor([-0.1468,  0.0856,  0.1191,  0.1105, -0.0675, -0.0679,  0.0974, -0.190
2,
        -0.1052, -0.1337, -0.1347, -0.0526,  0.0032,  0.1431, -0.1516,  0.061
```

```
8,
        -0.1948,  0.0931,  0.0579,  0.0799, -0.1024,  0.0350,  0.1849, -0.148
1,
         0.0133], requires_grad=True)
Parameter containing:
tensor([[ 0.0362, -0.1253, -0.1335, -0.1459,  0.1180, -0.1118,  0.1488,  0.01
61,
         -0.0589, -0.0037,  0.1676,  0.0412,  0.1529,  0.0831, -0.0608, -0.03
04,
          0.0924,  0.1343,  0.1946, -0.0999,  0.1608,  0.0416,  0.1986,  0.10
57,
         -0.0635]], requires_grad=True)
Parameter containing:
tensor([0.1583], requires_grad=True)
```

In [8]:

```python
def test(model, lss_fn, tst_dload):
    scores = []
    with torch.no_grad():
        model.eval()
        for (x_btch, y_btch) in tst_dload:
            out_btch = model(x_btch.float())
            lss = lss_fn(out_btch.float()[:,0], y_btch.long())
            scores.append(lss.item())
        model.train()
    return np.array(scores).mean()

test(testNet, nn.MSELoss(), tst_dload)
```

Out[8]:

```
14623.01309742647
```

In [9]:

```python
rnet = resultNet(Xtr,Ytr)
print(rnet)

learn_rate = 1
inertia = .8

criterion = nn.MSELoss()
optimizer = optim.SGD(rnet.parameters(), lr=learn_rate, momentum = inertia)


gpu_rdy = torch.cuda.is_available()
if gpu_rdy:

    print("Using GPU")
else:
    print("Using CPU")
```

```
resultNet(
  (c1): Linear(in_features=25, out_features=1, bias=True)
)
Using GPU
```

In [10]:

```
device = torch.device("cuda" if gpu_rdy else "cpu")

OGscr = test(rnet, criterion, tst_dload)

n_epochs = 201
idx = 0

tr_shp = Xtr.shape[0]

X_tr = torch.from_numpy(Xtr.values)
X_tr.to(device)

t_epochs = 0
```

In [11]:

```
#you can keep iterating this block to continue training the network

if gpu_rdy:
    print("On_GPU")
print("\nDisplayed score is MSE on 289 test data points while model is train
rnet.train() #just in case

print("\nUntrained score:            {}\n".format(OGscr))

lr_ = lambda epoch: (0.95 ** epoch)/10
scheduler = optim.lr_scheduler.LambdaLR(optimizer, lr_lambda=lr_)
for epoch in range(n_epochs):

    '''
    if idx + btch_sz >= tr_shp:
        idx = 0
    else:
        idx += btch_sz

    x_tr = Variable(x_tr[idx:(idx+btch_sz)].clone())
    '''


    for i, (x_btch, y_btch) in enumerate(tr_dload):
        if gpu_rdy:
            rnet.to(device)
            x_btch = x_btch.cuda()
            y_btch = y_btch.cuda()

        optimizer.zero_grad()

        out_btch = rnet(x_btch.float())
        out_lss = criterion(out_btch, y_btch)

        out_lss.backward()
        optimizer.step()
    t_epochs += 1
    rnet.to('cpu')
    scr = test(rnet, criterion, tst_dload)
    if (epoch %5) == 0:
        print("epoch {:06d} test data score: {}".format(t_epochs,scr))
    t_epochs += 1
```

```
On_GPU

Displayed score is MSE on 289 test data points while model is trained on 1155 t
raining data points


Untrained score:            14624.968778722427

epoch 000001 test data score: 14514.310403262867
epoch 000011 test data score: 14514.310345818014
epoch 000021 test data score: 14514.310288373163
epoch 000031 test data score: 14514.310288373163
epoch 000041 test data score: 14514.310288373163
epoch 000051 test data score: 14514.310259650736
epoch 000061 test data score: 14514.310259650736
epoch 000071 test data score: 14514.310259650736
epoch 000081 test data score: 14514.310259650736
epoch 000091 test data score: 14514.310259650736
epoch 000101 test data score: 14514.310259650736
epoch 000111 test data score: 14514.310259650736
epoch 000121 test data score: 14514.31023092831
epoch 000131 test data score: 14514.31023092831
epoch 000141 test data score: 14514.31023092831
epoch 000151 test data score: 14514.31023092831
epoch 000161 test data score: 14514.31023092831
epoch 000171 test data score: 14514.31023092831
epoch 000181 test data score: 14514.31023092831
epoch 000191 test data score: 14514.31023092831
epoch 000201 test data score: 14514.31023092831
epoch 000211 test data score: 14514.31023092831
epoch 000221 test data score: 14514.31023092831
epoch 000231 test data score: 14514.31023092831
epoch 000241 test data score: 14514.31023092831
epoch 000251 test data score: 14514.310202205883
epoch 000261 test data score: 14514.310202205883
epoch 000271 test data score: 14514.310202205883
epoch 000281 test data score: 14514.310202205883
epoch 000291 test data score: 14514.310202205883
epoch 000301 test data score: 14514.310202205883
epoch 000311 test data score: 14514.310202205883
epoch 000321 test data score: 14514.310202205883
epoch 000331 test data score: 14514.310202205883
epoch 000341 test data score: 14514.310202205883
epoch 000351 test data score: 14514.310202205883
epoch 000361 test data score: 14514.310202205883
epoch 000371 test data score: 14514.310202205883
epoch 000381 test data score: 14514.310202205883
epoch 000391 test data score: 14514.310202205883
epoch 000401 test data score: 14514.310202205883
```

In [12]:

```python
tr_scr = test(rnet, criterion, tr_dload)
print("Score on training data for comparison: {}".format(tr_scr))

```

```
Score on training data for comparison: 15276.030535797574

C:\Users\Triplea657\anaconda3\envs\MSCS335\lib\site-packages\torch\nn\modules\l
oss.py:528: UserWarning: Using a target size (torch.Size([20, 1])) that is diff
erent to the input size (torch.Size([20])). This will likely lead to incorrect
results due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
C:\Users\Triplea657\anaconda3\envs\MSCS335\lib\site-packages\torch\nn\modules\l
oss.py:528: UserWarning: Using a target size (torch.Size([10, 1])) that is diff
erent to the input size (torch.Size([10])). This will likely lead to incorrect
results due to broadcasting. Please ensure they have the same size.
  return F.mse_loss(input, target, reduction=self.reduction)
```

I wasn't really able to find anything interesting with neural networks on this data. Perhaps if I had planned out my schedule better and had devoted more time to just letting my

computer sit and crunch while I worked on other things, I would've been able to test more.