6 - Data Project – Team Winrate - Neural Networks - Summer 2019

June 1, 2020

[1]:
```
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_validate
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
from keras.wrappers.scikit_learn import KerasClassifier
import matplotlib.pyplot as plt
import graphviz as gviz
%matplotlib inline

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

```
Using TensorFlow backend.
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:516: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:517: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:518: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
```

```
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:519: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:520: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorflow\python\framework\dtypes.py:525: FutureWarning: Passing
(type, 1) or '1type' as a synonym of type is deprecated; in a future version of
numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:541: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint8 = np.dtype([("qint8", np.int8, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:542: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint8 = np.dtype([("quint8", np.uint8, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:543: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint16 = np.dtype([("qint16", np.int16, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:544: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_quint16 = np.dtype([("quint16", np.uint16, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:545: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  _np_qint32 = np.dtype([("qint32", np.int32, 1)])
C:\Users\Triplea657\anaconda3\lib\site-
packages\tensorboard\compat\tensorflow_stub\dtypes.py:550: FutureWarning:
Passing (type, 1) or '1type' as a synonym of type is deprecated; in a future
version of numpy, it will be understood as (type, (1,)) / '(1,)type'.
  np_resource = np.dtype([("resource", np.ubyte, 1)])

<IPython.core.display.HTML object>
```

[2]:
```
#year = "2019"                                                    #choose year␣
 ↪to get data from
#split = "summer"                                                 #choose split␣
 ↪to get data from(spring, summer, worlds)
#infile = r"C:\Users\Triplea657\000 MSCS-335 2020\Datasets\League_"#path
#inf = "-Wrangled.csv"                                            #file to read
#filein = infile+year+"\\"+year+'-'+split+'-'+inf
#data = pd.read_csv(filein,low_memory=False)
#data.head(10)

#changed for submission version
data = pd.read_csv("Datasets/League_2019/2019-summer-Wrangled.csv", index_col=0,␣
 ↪low_memory=False)
data.head()
```

[2]:

	league_CBLoL	league_LCK	league_LCS	league_LEC	league_LMS	gamelength	\
0	0.0	0.0	1.0	0.0	0.0	35.500000	
1	0.0	0.0	1.0	0.0	0.0	35.500000	
2	0.0	0.0	1.0	0.0	0.0	29.700000	
3	0.0	0.0	1.0	0.0	0.0	29.700000	
4	0.0	0.0	1.0	0.0	0.0	31.983333	

	result	k	d	a	fb	kpm	okpm	ckpm	fd	\
0	1.0	21.0	14.0	52.0	0.0	0.591549	0.394366	0.985915	0.0	
1	0.0	14.0	21.0	32.0	1.0	0.394366	0.591549	0.985915	1.0	
2	1.0	11.0	4.0	25.0	1.0	0.370370	0.134680	0.505051	1.0	
3	0.0	4.0	11.0	10.0	0.0	0.134680	0.370370	0.505051	0.0	
4	1.0	12.0	3.0	26.0	1.0	0.375195	0.093799	0.468994	0.0	

	fdtime	teamdragkills	oppdragkills	elementals	oppelementals	\
0	12.556633	2.0	2.0	2.0	2.0	
1	12.556633	2.0	2.0	2.0	2.0	
2	12.306967	2.0	1.0	2.0	1.0	
3	12.306967	1.0	2.0	1.0	2.0	
4	10.158933	3.0	1.0	3.0	1.0	

	firedrakes	waterdrakes	earthdrakes	airdrakes	elders	oppelders	herald	\
0	2.0	0.0	0.0	0.0	0.0	0.0	1.0	
1	1.0	0.0	0.0	1.0	0.0	0.0	0.0	
2	0.0	0.0	1.0	1.0	0.0	0.0	0.0	
3	0.0	1.0	0.0	0.0	0.0	0.0	1.0	
4	1.0	0.0	0.0	2.0	0.0	0.0	0.0	

	heraldtime	ft	fttime	firstmidouter	firsttothreetowers	\
0	13.369417	1.0	15.162683	1.0	0.0	

```
1   13.369417  0.0  15.162683            0.0                 1.0
2   12.377433  1.0  12.791600            1.0                 1.0
3   12.377433  0.0  12.791600            0.0                 0.0
4   12.242783  0.0  14.386333            1.0                 1.0

   teambaronkills  oppbaronkills  dmgtochamps  dmgtochampsperminute  wards  \
0             1.0            0.0      70545.0           1987.183099  109.0
1             0.0            1.0      71736.0           2020.732394  108.0
2             1.0            0.0      51538.0           1735.286195   96.0
3             0.0            1.0      38185.0           1285.690236   93.0
4             1.0            0.0      49421.0           1545.211047  143.0

        wpm  wardkills      wcpm  totalgold    earnedgpm  goldspent      gspd  \
0  3.070423       51.0  1.436620    69022.0  1293.464789    65108.0  0.110966
1  3.042254       37.0  1.042254    61541.0  1082.732394    58263.0 -0.110966
2  3.232323       44.0  1.481481    59081.0  1330.861953    50910.0  0.135867
3  3.131313       41.0  1.380471    45794.0   883.488215    44433.0 -0.135867
4  4.471079       44.0  1.375717    61326.0  1262.351225    54340.0  0.158169

   monsterkillsownjungle  monsterkillsenemyjungle       cspm  goldat10  \
0                  151.0                     24.0  31.802817   16118.0
1                  155.0                      4.0  32.985915   15436.0
2                  102.0                     56.0  35.656566   16270.0
3                   82.0                      0.0  33.265993   14985.0
4                  128.0                     18.0  34.299114   16157.0

   oppgoldat10  gdat10  goldat15  oppgoldat15  gdat15   xpat10  oppxpat10  \
0      15436.0   682.0   24287.0      23616.0   671.0  19260.0    18621.0
1      16118.0  -682.0   23616.0      24287.0  -671.0  18621.0    19260.0
2      14985.0  1285.0   27399.0      23026.0  4373.0  19015.0    18226.0
3      16270.0 -1285.0   23026.0      27399.0 -4373.0  18226.0    19015.0
4      14365.0  1792.0   26339.0      22782.0  3557.0  19284.0    18656.0

   xpdat10  csat10  oppcsat10  csdat10  csat15  oppcsat15  csdat15
0    639.0   334.0      316.0     18.0   548.0      535.0     13.0
1   -639.0   316.0      334.0    -18.0   535.0      548.0    -13.0
2    789.0   316.0      335.0    -19.0   509.0      506.0      3.0
3   -789.0   335.0      316.0     19.0   506.0      509.0     -3.0
4    628.0   322.0      305.0     17.0   512.0      470.0     42.0
```

[3]:
```
var = []
for i in data:
    var.append(i)
print(var)
for i, v in enumerate(var):
    print(i, v)
```

```
['league_CBLoL', 'league_LCK', 'league_LCS', 'league_LEC', 'league_LMS',
'gamelength', 'result', 'k', 'd', 'a', 'fb', 'kpm', 'okpm', 'ckpm', 'fd',
'fdtime', 'teamdragkills', 'oppdragkills', 'elementals', 'oppelementals',
'firedrakes', 'waterdrakes', 'earthdrakes', 'airdrakes', 'elders', 'oppelders',
'herald', 'heraldtime', 'ft', 'fttime', 'firstmidouter', 'firsttothreetowers',
'teambaronkills', 'oppbaronkills', 'dmgtochamps', 'dmgtochampsperminute',
'wards', 'wpm', 'wardkills', 'wcpm', 'totalgold', 'earnedgpm', 'goldspent',
'gspd', 'monsterkillsownjungle', 'monsterkillsenemyjungle', 'cspm', 'goldat10',
'oppgoldat10', 'gdat10', 'goldat15', 'oppgoldat15', 'gdat15', 'xpat10',
'oppxpat10', 'xpdat10', 'csat10', 'oppcsat10', 'csdat10', 'csat15', 'oppcsat15',
'csdat15']
0 league_CBLoL
1 league_LCK
2 league_LCS
3 league_LEC
4 league_LMS
5 gamelength
6 result
7 k
8 d
9 a
10 fb
11 kpm
12 okpm
13 ckpm
14 fd
15 fdtime
16 teamdragkills
17 oppdragkills
18 elementals
19 oppelementals
20 firedrakes
21 waterdrakes
22 earthdrakes
23 airdrakes
24 elders
25 oppelders
26 herald
27 heraldtime
28 ft
29 fttime
30 firstmidouter
31 firsttothreetowers
32 teambaronkills
33 oppbaronkills
34 dmgtochamps
35 dmgtochampsperminute
36 wards
```

```
37 wpm
38 wardkills
39 wcpm
40 totalgold
41 earnedgpm
42 goldspent
43 gspd
44 monsterkillsownjungle
45 monsterkillsenemyjungle
46 cspm
47 goldat10
48 oppgoldat10
49 gdat10
50 goldat15
51 oppgoldat15
52 gdat15
53 xpat10
54 oppxpat10
55 xpdat10
56 csat10
57 oppcsat10
58 csdat10
59 csat15
60 oppcsat15
61 csdat15
```

[4]:
```
X = data.iloc[:,data.columns != 'gamelength']
Y = data.iloc[:,data.columns == 'gamelength']
#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_state=2020)
print(Xtr.shape,Xtst.shape,Ytr.shape,Ytst.shape)
Ytr = pd.DataFrame(Ytr)
```

```
Xtr(Xtrain),Xtst(Xtest),Ytr(Ytrain),Ytst(Ytest) shapes:
(1155, 61) (289, 61) (1155,) (289,)
```

[5]:
```
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)
```

[6]:
```
'''
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.hidden3Size)
                        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([[ 0.0388,  0.1013,  0.1157,  ..., -0.1052, -0.0967, -0.0428],
        [-0.0092,  0.0366, -0.0787,  ...,  0.0367, -0.0146, -0.0793],
        [ 0.0820, -0.1232, -0.0910,  ...,  0.0141,  0.0855,  0.0146],
        ...,
        [ 0.0277,  0.0616, -0.0854,  ..., -0.1273, -0.0461, -0.0347],
        [-0.0885, -0.0376, -0.1153,  ...,  0.0691,  0.1215, -0.0200],
        [ 0.0332,  0.1149, -0.0170,  ..., -0.0568, -0.1111, -0.0625]],
       requires_grad=True)
Parameter containing:
tensor([ 0.0934,  0.0152, -0.0124, -0.1057, -0.0228, -0.0982, -0.1204, -0.0362,
        -0.1142,  0.0463,  0.1173,  0.0019,  0.0323,  0.0397,  0.1117,  0.0472,
        -0.1130, -0.1237, -0.0743, -0.0341,  0.0063, -0.0447, -0.1101, -0.0155,
         0.0611], requires_grad=True)
Parameter containing:
tensor([[ 0.1282,  0.1975, -0.1186,  0.1294, -0.0736,  0.1879, -0.0985,  0.0556,
          0.1742,  0.1811, -0.1557, -0.0775,  0.1162,  0.0522,  0.1669, -0.0156,
         -0.0307, -0.1315, -0.0123, -0.1643, -0.0633, -0.1234, -0.0986, -0.1840,
         -0.0619]], requires_grad=True)
Parameter containing:
tensor([-0.0972], requires_grad=True)
```

[7]:
```
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)
```

[7]:
```
1093.6089640299479
```

[8]:
```
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=61, out_features=1, bias=True)
)
Using GPU
```

[9]:
```
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
```

[10]:
```
#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 trained␣
 ↪on 1155 training data points\n")
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
training data points


Untrained score:            1092.8280598958333

epoch 000001 test data score: 1059.9759399414063
epoch 000011 test data score: 1059.7825439453125
epoch 000021 test data score: 1059.7604410807292
epoch 000031 test data score: 1059.7499674479166
epoch 000041 test data score: 1059.7434895833333
epoch 000051 test data score: 1059.7392252604166
```

```
epoch 000061 test data score: 1059.7361083984374
epoch 000071 test data score: 1059.733837890625
epoch 000081 test data score: 1059.732071940104
epoch 000091 test data score: 1059.7306396484375
epoch 000101 test data score: 1059.7294759114584
epoch 000111 test data score: 1059.7285237630208
epoch 000121 test data score: 1059.7277180989583
epoch 000131 test data score: 1059.7269856770833
epoch 000141 test data score: 1059.726416015625
epoch 000151 test data score: 1059.7258707682292
epoch 000161 test data score: 1059.7254150390625
epoch 000171 test data score: 1059.7250244140625
epoch 000181 test data score: 1059.7246500651042
epoch 000191 test data score: 1059.72431640625
epoch 000201 test data score: 1059.724031575521
epoch 000211 test data score: 1059.7237630208333
epoch 000221 test data score: 1059.7235188802083
epoch 000231 test data score: 1059.7233072916667
epoch 000241 test data score: 1059.7230875651042
epoch 000251 test data score: 1059.7229085286458
epoch 000261 test data score: 1059.7227376302083
epoch 000271 test data score: 1059.7225830078125
epoch 000281 test data score: 1059.7224283854166
epoch 000291 test data score: 1059.7222981770833
epoch 000301 test data score: 1059.72216796875
epoch 000311 test data score: 1059.7220540364583
epoch 000321 test data score: 1059.7219401041666
epoch 000331 test data score: 1059.721834309896
epoch 000341 test data score: 1059.7217366536458
epoch 000351 test data score: 1059.7216471354166
epoch 000361 test data score: 1059.7215576171875
epoch 000371 test data score: 1059.7214762369792
epoch 000381 test data score: 1059.7213948567708
epoch 000391 test data score: 1059.7213216145833
epoch 000401 test data score: 1059.7212565104167
```

[11]:
```
tr_scr = test(rnet, criterion, tr_dload)
print("Score on training data for comparison: {}".format(tr_scr))
```

```
Score on training data for comparison: 1031.466375942888

C:\Users\Triplea657\anaconda3\lib\site-packages\torch\nn\modules\loss.py:432:
UserWarning: Using a target size (torch.Size([20, 1])) that is different 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\lib\site-packages\torch\nn\modules\loss.py:432:
UserWarning: Using a target size (torch.Size([15, 1])) that is different to the
```

```
input size (torch.Size([15])). 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)
```

[ ]: