1번 대한민국 육국 신체 측정 데이터 [몸무게->허리둘레예측]

In [250]:

```
import numpy as np
import matplotlib.pyplot as plt
import torch
import torch.nn as nn
import torch.optim as optim
import torch.nn.init as init
import torchvision.datasets as dset
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
```

In [251]:

```
f = open('./armybody.csv', "r", newline='₩r')
data = f.readline()

lines = data.split('₩n')
header = lines[0].split(',')
print(header)

data = []
while 1:
    line = f.readline()
    data.append(line)
    if not line: break
f.close()
```

```
['순번', '측정 일자', '가슴 둘레 센티미터', '소매길이 센티미터', '신장 센티미터',
'허리 둘레 센티미터', '샅높이 센티미터', '머리 둘레 센티미터', '발 길이 센티미터',
'몸무게 킬로그램₩r']
```

In [252]:

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

```
(135672,)
```

In [253]:

```
print(data[108400])
sample = data[108400].strip('₩n').split(',')
print(sample)
print(len(sample))
```

```
108401,20160307,,85.9,171.3,91.7,79,57.1,25.2,69.5
['108401', '20160307', '', '85.9', '171.3', '91.7', '79', '57.1', '25.2', '69.5
₩r']
10
```

In [254]:

```
new_data = []
for i in range(0, len(data)):
    seq = data[i].strip().split(',')
    if not len(seq)==10:
        print(i)
    if i==108400:
        continue
    if len(seq)==10:
        new_data.append(seq)
df = np.array(new_data, 'float')
```

```
135670
135671
```

In [255]:

```
print(data[135670])
```

1-1 데이터 세트의 차원 형태(dimension shape)를 출력해 보세요.

In [348]:

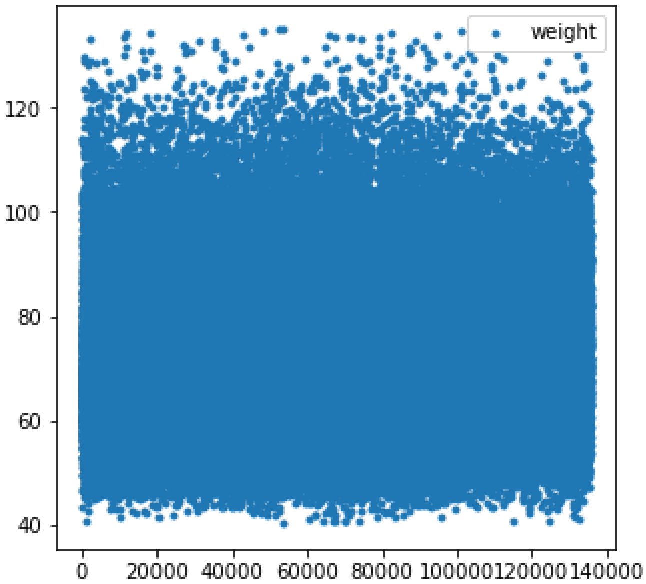
```
print(df.shape)
print('결측치가 있는 108400데이터와, 읽어지지 않는 135670두개의 데이터를 제외한 결과입니다.')
```

```
(135669, 10)
결측치가 있는 108400데이터와, 읽어지지 않는 135670두개의 데이터를 제외한 결과입니
다.
```

1-2 2차원 평면에 몸무게와 허리둘레 데이터 포인트들을 가시화 해보세요.

In [257]:

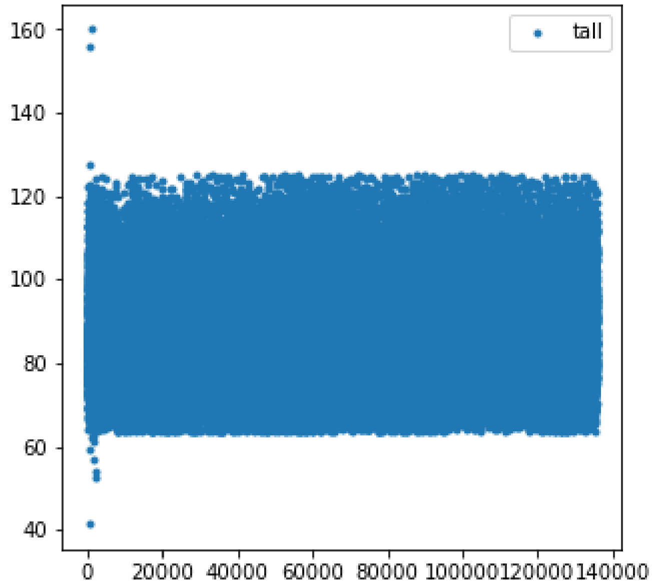
```
plt.plot(df[:,-1], '.')
plt.legend(['weight'])
plt.figure(figsize=(10,15))
plt.show()
```



<Figure size 720x1080 with 0 Axes>

In [258]:

```
plt.plot(df[:,5], '.')
plt.legend(['tall'])
plt.figure(figsize=(10,15))
plt.show()
```




```
<Figure size 720x1080 with 0 Axes>
```

1-3 데이터 세트 중 100000개를 학습 세트로 하고 나머지를 테스트 세트로 나누고 각 세트의 마지막 데이터를 출력해 보세요.

In [259]:

```
trainX = df[:100000, -1].reshape(-1,1)
trainY = df[:100000, 5].reshape(-1,1)
testX = df[100000:, -1].reshape(-1,1)
testY = df[100000:, 5].reshape(-1,1)
print(trainX.shape)
print(trainY.shape)
print(testX.shape)
print(testY.shape)
```

```
(100000, 1)
(100000, 1)
(35669, 1)
(35669, 1)
```

In [260]:

```
print(trainX[-1], trainY[-1])
print(testX[-1], testY[-1])
```

```
[65.9] [79.5]
[90.6] [105.9]
```

1-4 선형회귀(linear regression) 모델을 설계하고 학습 모델의 구성하고 각 계층별 요약과 파라미터 수를 출력해 보세요.

In [261]:

```
model = nn.Linear(1,1)

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.0002)
```

In [262]:

```
print(model)
print(list(model.parameters()))
```

```
Linear(in_features=1, out_features=1, bias=True)
[Parameter containing:
tensor([[-0.9989]], requires_grad=True), Parameter containing:
tensor([0.0727], requires_grad=True)]
```

1-5 에포크를 1000, 학습률 0.001로 튜닝하고 테스트 데이터로 에포크별 MAE(mean absolute error) 손실을 평가하여 그래프로 그려보세요.

In [263]:

```
num_epoch = 1000
```

In [264]:

```
trainX_tensor = torch.FloatTensor(trainX)
trainY_tensor = torch.FloatTensor(trainY)
testX_tensor = torch.FloatTensor(testX)
testY_tensor = torch.FloatTensor(testY)
```

In [266]:

```
print(trainX_tensor.shape)
print(trainY_tensor.shape)
```

```
torch.Size([100000, 1])
torch.Size([100000, 1])
```

In [267]:

```
loss_arr =[]

for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(trainX_tensor)
    loss = loss_func(output,trainY_tensor)
    loss.backward()

    optimizer.step()

    loss_arr.append(loss.detach().numpy())
```

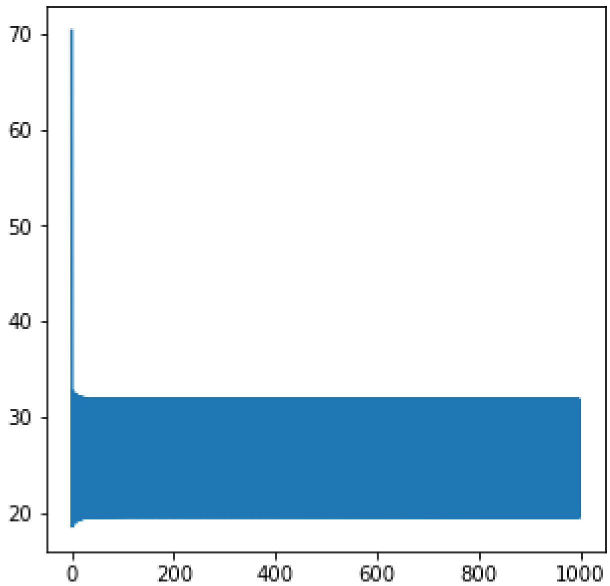
In [172]:

```
print('lr = 0.01')
plt.plot(loss_arr)
```

lr = 0.01

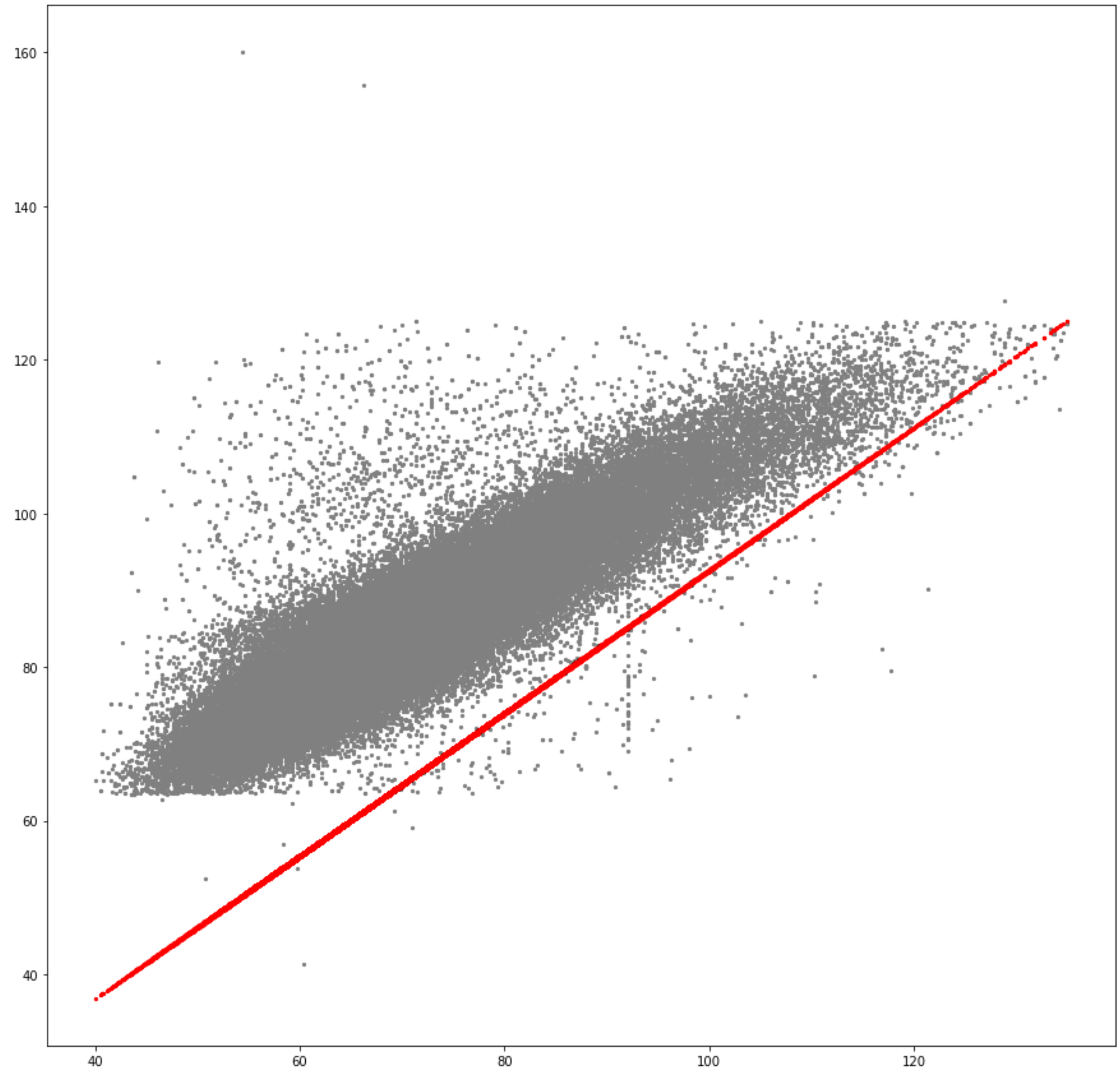
Out[172]:

[<matplotlib.lines.Line2D at 0x26a327667c8>]



In [173]:

```
plt.figure(figsize=(15,15))
plt.scatter(trainX_tensor.numpy(),trainY_tensor.numpy(),s=5,c="gray")
plt.scatter(trainX_tensor.detach().numpy(),output.detach().numpy(),s=5,c="red")
plt.show()
print('lr = 0.01')
```




```
lr = 0.01
```

1-6 손실함수로 MSE(mean squared error)를 사용해도 성능이 잘 나오도록 튜닝해 보세요.

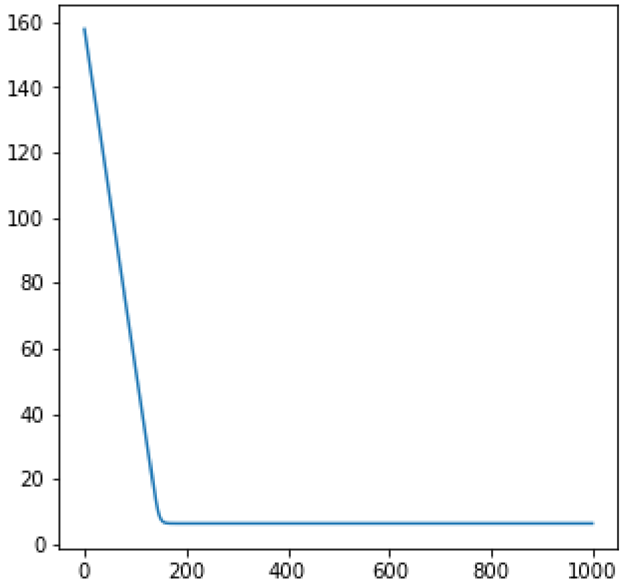
In [268]:

```
print('lr = 0.0002')
plt.plot(loss_arr)
```

```
lr = 0.0002
```

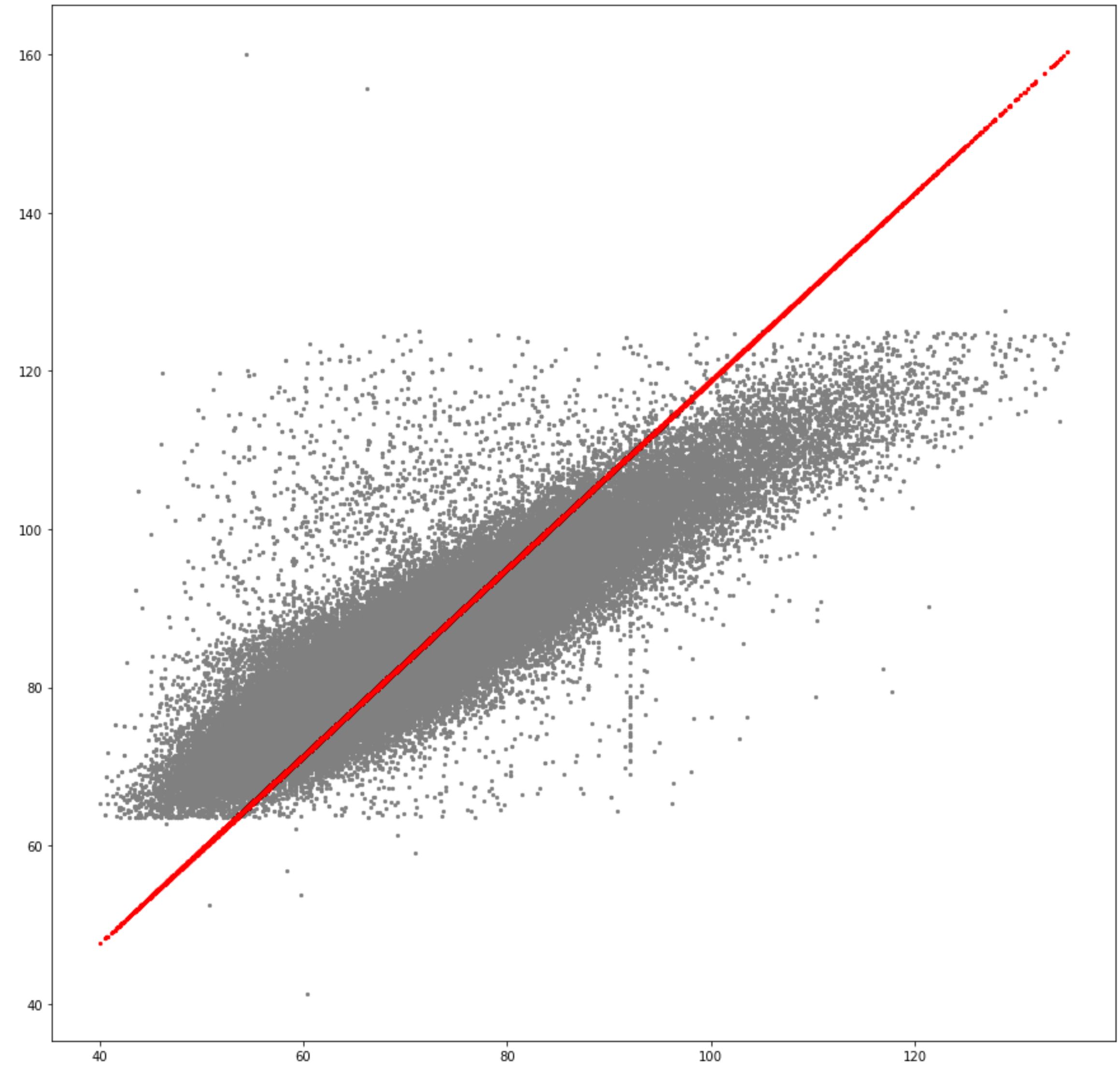
Out[268]:

```
[<matplotlib.lines.Line2D at 0x26a32325308>]
```



In [269]:

```
plt.figure(figsize=(15,15))
plt.scatter(trainX_tensor.numpy(),trainY_tensor.numpy(),s=5,c="gray")
plt.scatter(trainX_tensor.detach().numpy(),output.detach().numpy(),s=5,c="red")
plt.show()
print('lr = 0.0002')
```



lr = 0.0002

2번 인공신경망 설계

In [271]:

```
num_data =1000
```

In [272]:

```
noise = init.normal_(torch.FloatTensor(num_data,1),std=1)
x = init.uniform_(torch.Tensor(num_data,1),-15,15)
y = (x+noise)**3
y_noise = y + noise
```

In [273]:

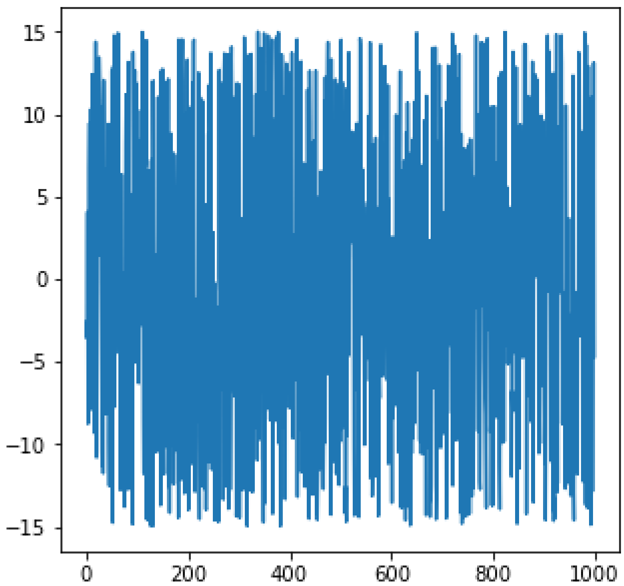
```
print(x.shape)
print(y.shape)
```

```
torch.Size([1000, 1])
torch.Size([1000, 1])
```

2-1 2차원 평면에 x, y 포인트들을 가시화 해보세요.

In [274]:

```
plt.plot(x)
plt.rcParams['figure.figsize'] = [5, 5]
```

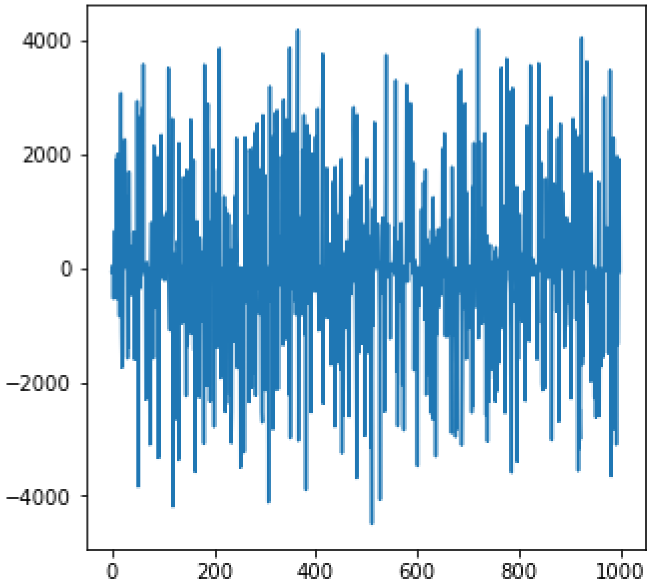


In [275]:

```
plt.plot(y)
```

Out[275]:

```
[<matplotlib.lines.Line2D at 0x26a3236b9c8>]
```

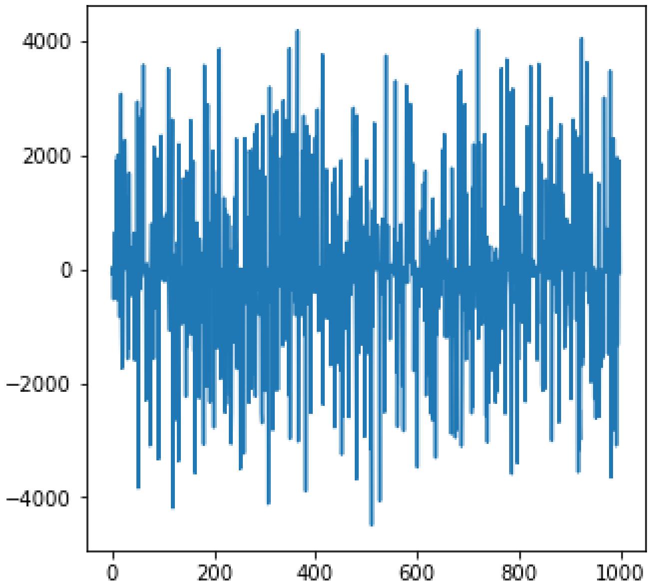


In [276]:

```
plt.plot(y_noise)
```

Out[276]:

```
[<matplotlib.lines.Line2D at 0x26a3270dd08>]
```



2-2 특성의 개수가 1 -> 50 -> 100 -> 1개로 변하는 완전연결 신경망(fully connected neural network) 모델을 설계하고 학습 모델의 구성과 각 계층별 요약과 파라미터 수를 출력해 보세요.

In [279]:

```
model = nn.Sequential(
          nn.Linear(1,50),
          nn.ReLU(),
          nn.Linear(50,100),
          nn.ReLU(),
          nn.Linear(100,1)
      )

loss_func = nn.L1Loss()
optimizer = optim.SGD(model.parameters(),lr=0.0002)
```

In [280]:

```
print(model)
print(list(model.parameters()))
```

```
Sequential(
  (0): Linear(in_features=1, out_features=50, bias=True)
  (1): ReLU()
  (2): Linear(in_features=50, out_features=100, bias=True)
  (3): ReLU()
  (4): Linear(in_features=100, out_features=1, bias=True)
)
[Parameter containing:
tensor([[ 0.6889],
        [ 0.7615],
        [ 0.6350],
        [-0.4935],
        [-0.4293],
        [ 0.3529],
        [ 0.7595],
        [-0.2271],
        [ 0.9399],
        [ 0.7234],
        [ 0.0991],
        [-0.5423],
        [ 0.6210],
        [-0.4160],
        [ 0.9601],
        [-0.9942],
        [-0.3650],
        [-0.0538],
        [-0.8180],
        [-0.9769],
        [ 0.0070],
        [-0.8975],
        [ 0.3065],
        [ 0.3240],
        [ 0.3079],
        [-0.6589],
        [ 0.9711],
        [-0.0983],
        [-0.9970],
        [ 0.9710],
        [-0.5667],
        [-0.9331],
        [ 0.0765],
        [ 0.1807],
        [ 0.3450],
        [-0.0554],
        [-0.0644],
        [-0.8512],
        [ 0.3380],
        [-0.1693],
        [-0.2905],
        [ 0.9146],
        [ 0.3973],
        [-0.6453],
        [ 0.6901],
        [-0.3293],
        [ 0.8313],
        [ 0.1344],
        [ 0.6190],
        [ 0.9309]], requires_grad=True), Parameter containing:
tensor([-0.5251,  0.5015,  0.0624,  0.3049, -0.2317, -0.9087,  0.7586, -0.8024,
         0.4438, -0.1884,  0.4242, -0.4967,  0.5658,  0.4770,  0.7975,  0.4401,
        -0.1686,  0.1154,  0.6465, -0.7017, -0.8131,  0.3066,  0.0422,  0.3200,
```

```
        -0.6791,  0.8217, -0.8211,  0.9654, -0.4926, -0.2379, -0.5888, -0.0942,
         0.6315, -0.9432,  0.6416,  0.6962, -0.3162,  0.4853, -0.5199,  0.8079,
         0.9897,  0.6902, -0.1436,  0.4817,  0.7120,  0.1835, -0.6940,  0.2788,
         0.8105,  0.0650], requires_grad=True), Parameter containing:
tensor([[-0.0787, -0.1330,  0.1355,  ...,  0.1087,  0.0908, -0.1005],
        [-0.0002,  0.0918,  0.1127,  ...,  0.0937,  0.0131, -0.1303],
        [ 0.0391,  0.0369, -0.0379,  ...,  0.1108, -0.0324,  0.0049],
        ...,
        [-0.0035,  0.0577,  0.0776,  ...,  0.1057, -0.1378, -0.0365],
        [-0.0633,  0.0309,  0.0526,  ...,  0.1073,  0.0420,  0.0232],
        [-0.0422, -0.0585, -0.0055,  ...,  0.0775,  0.0400,  0.0221]],
       requires_grad=True), Parameter containing:
tensor([-0.0006,  0.1402,  0.1066,  0.1241,  0.0185,  0.0186, -0.0959,  0.0081,
         0.1276, -0.0492,  0.0037, -0.0925,  0.0648, -0.1283, -0.1132, -0.1107,
        -0.1375, -0.0645,  0.0507, -0.0688, -0.1127, -0.0920,  0.0525, -0.0829,
         0.0938,  0.0929,  0.0968,  0.0782, -0.0266,  0.0435, -0.0250, -0.0797,
         0.1342, -0.0077, -0.0668,  0.0484, -0.0262, -0.0460, -0.0217,  0.0424,
         0.1130, -0.0662, -0.0635,  0.0642,  0.0562,  0.0482,  0.0562, -0.0126,
        -0.1313, -0.0214,  0.0938, -0.0639, -0.1047, -0.0366, -0.0028,  0.0750,
         0.0008, -0.1065,  0.0093, -0.1281, -0.1195, -0.1030,  0.0008,  0.0054,
         0.0262, -0.0561, -0.0947, -0.0783, -0.1224,  0.0658,  0.1200, -0.0677,
         0.0799,  0.0406,  0.0684, -0.0686, -0.0627, -0.1217, -0.0102,  0.1378,
        -0.0376,  0.1311,  0.0383, -0.0797, -0.1366, -0.0459, -0.0757, -0.0447,
         0.0384,  0.1312, -0.0326, -0.0437,  0.1333,  0.0060,  0.1073, -0.0215,
         0.1342, -0.0313,  0.1167,  0.0182], requires_grad=True), Parameter contai
ning:
tensor([[-0.0158,  0.0154,  0.0988,  0.0563, -0.0839,  0.0272, -0.0675, -0.0724,
          0.0723,  0.0676,  0.0457,  0.0771, -0.0267,  0.0652,  0.0566,  0.0068,
         -0.0017, -0.0977, -0.0699,  0.0293,  0.0519, -0.0682, -0.0801,  0.0866,
          0.0755,  0.0634, -0.0221,  0.0875,  0.0704, -0.0089, -0.0407, -0.0885,
          0.0287, -0.0595,  0.0826,  0.0079, -0.0215,  0.0990,  0.0532,  0.0780,
          0.0069, -0.0544, -0.0448,  0.0913, -0.0042, -0.0306, -0.0640,  0.0784,
          0.0686,  0.0259, -0.0096, -0.0479, -0.0552, -0.0730, -0.0385,  0.0438,
          0.0951, -0.0057, -0.0530,  0.0919,  0.0312, -0.0967, -0.0854, -0.0077,
         -0.0077, -0.0765, -0.0798, -0.0568, -0.0083, -0.0715, -0.0411,  0.0969,
          0.0260, -0.0073, -0.0907, -0.0317,  0.0275, -0.0427, -0.0265, -0.0021,
          0.0217,  0.0978,  0.0782,  0.0309, -0.0017, -0.0021, -0.0633,  0.0401,
          0.0056,  0.0760, -0.0089, -0.0556, -0.0041,  0.0086,  0.0156,  0.0504,
          0.0676,  0.0634, -0.0724,  0.0301]], requires_grad=True), Parameter cont
aining:
tensor([-0.0422], requires_grad=True)]
```

2-3 에포크 10000, 학습률 0.0002로 설정하고 에포크별 MAE(mean absolute error) 손실을 평가하여 그래프로 그려보세요.

In [281]:

```
num_epoch = 10000
```

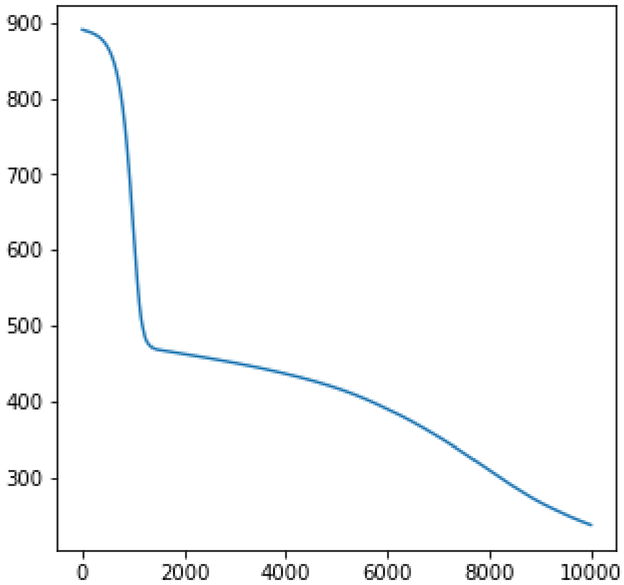
In [282]:

```
loss_array = []
for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(x)

    loss = loss_func(output,y_noise)
    loss.backward()
    optimizer.step()

    loss_array.append(loss)
```

In [285]:

```
plt.plot(loss_array)
print('lr=0.0002')
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-285-cb3031b1efda> in <module>
      1 plt.plot(loss_array)
----> 2 print(lr=0.0002)

TypeError: 'lr' is an invalid keyword argument for print()
```



2-4 손실함수로 MSE(mean squared error)를 사용해도 성능이 잘 나오도록 튜닝해 보세요.

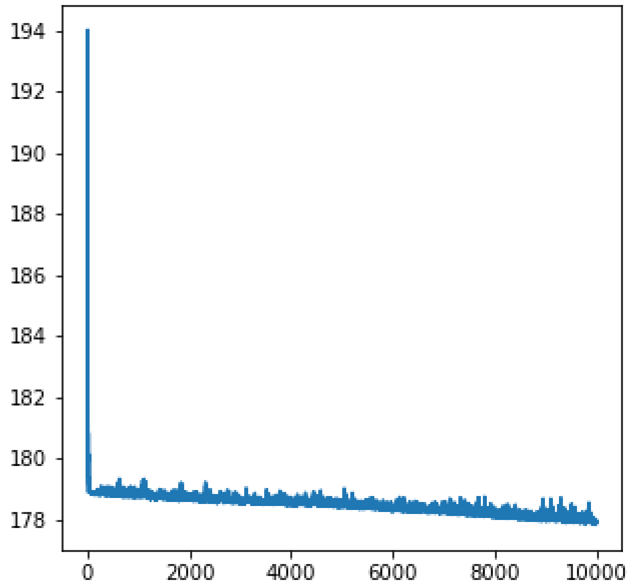
In [294]:

```
loss_func = nn.L1Loss()
optimizer = optim.Adam(model.parameters(),lr=0.005)
loss_array = []
for i in range(num_epoch):
    optimizer.zero_grad()
    output = model(x)

    loss = loss_func(output,y_noise)
    loss.backward()
    optimizer.step()

    loss_array.append(loss)
```

In [295]:

```
plt.plot(loss_array)
print('Adam,lr=0.0001')
```

Adam,lr=0.0001



3번 보너스

In [284]:

```
mnist_train = dset.FashionMNIST(root="../", train=True, transform=transforms.ToTensor(), target
_transform=None, download=True)
mnist_test = dset.FashionMNIST(root="../", train=False, transform=transforms.ToTensor(), target
_transform=None, download=True)
```

3-1 학습 세트의 마지막 데이터를 이미지로 출력해 보세요.

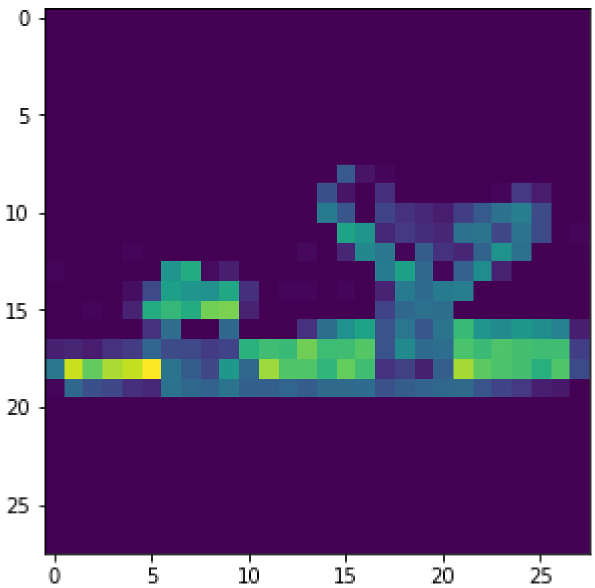
In [310]:

```
print(mnist_train.__getitem__(0)[0].size(), mnist_train.__len__())
print(mnist_test.__getitem__(0)[0].size(), mnist_test.__len__())
print(len(mnist_train),len(mnist_test))
```

```
torch.Size([1, 28, 28]) 60000
torch.Size([1, 28, 28]) 10000
60000 10000
```

In [311]:

```
plt.imshow(mnist_train.__getitem__(59999)[0].reshape(28,28))
plt.show()
```



3-2 학습 세트의 마지막 데이터에 대하여 1개의 채널을 입력으로 받아서 3개의 채널이 나오는 컨볼루션 연산을 적용해서 출력해 보세요.

In [326]:

```
model = nn.Sequential(
            nn.Conv2d(in_channels=1,out_channels=3,kernel_size=1)
      )

output = model(mnist_train.__getitem__(59999)[0].reshape(-1,1,28,28))
print(output.shape)
print(output)
```

```
torch.Size([1, 3, 28, 28])
tensor([[[[-0.7178, -0.7178, -0.7178,  ..., -0.7178, -0.7178, -0.7178],
          [-0.7178, -0.7178, -0.7178,  ..., -0.7178, -0.7178, -0.7178],
          [-0.7178, -0.7178, -0.7178,  ..., -0.7178, -0.7178, -0.7178],
          ...,
          [-0.7178, -0.7178, -0.7178,  ..., -0.7178, -0.7178, -0.7178],
          [-0.7178, -0.7178, -0.7178,  ..., -0.7178, -0.7178, -0.7178],
          [-0.7178, -0.7178, -0.7178,  ..., -0.7178, -0.7178, -0.7178]],

         [[-0.4956, -0.4956, -0.4956,  ..., -0.4956, -0.4956, -0.4956],
          [-0.4956, -0.4956, -0.4956,  ..., -0.4956, -0.4956, -0.4956],
          [-0.4956, -0.4956, -0.4956,  ..., -0.4956, -0.4956, -0.4956],
          ...,
          [-0.4956, -0.4956, -0.4956,  ..., -0.4956, -0.4956, -0.4956],
          [-0.4956, -0.4956, -0.4956,  ..., -0.4956, -0.4956, -0.4956],
          [-0.4956, -0.4956, -0.4956,  ..., -0.4956, -0.4956, -0.4956]],

         [[ 0.1318,  0.1318,  0.1318,  ...,  0.1318,  0.1318,  0.1318],
          [ 0.1318,  0.1318,  0.1318,  ...,  0.1318,  0.1318,  0.1318],
          [ 0.1318,  0.1318,  0.1318,  ...,  0.1318,  0.1318,  0.1318],
          ...,
          [ 0.1318,  0.1318,  0.1318,  ...,  0.1318,  0.1318,  0.1318],
          [ 0.1318,  0.1318,  0.1318,  ...,  0.1318,  0.1318,  0.1318],
          [ 0.1318,  0.1318,  0.1318,  ...,  0.1318,  0.1318,  0.1318]]]],
       grad_fn=<MkldnnConvolutionBackward>)
```

3-3 학습 세트의 마지막 데이터에 대하여 커널 사이즈 3인 컨볼루션 연산을 적용해서 출력해 보세요.

In [327]:

```
model = nn.Sequential(
            nn.Conv2d(in_channels=1,out_channels=1,kernel_size=3)
      )
output = model(mnist_train.__getitem__(59999)[0].reshape(-1,1,28,28))
print(output.shape)
print(output)
```

```
torch.Size([1, 1, 26, 26])
tensor([[[[-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2963, -0.2962, -0.2974, -0.3420,
           -0.3447, -0.3897, -0.3133, -0.3004, -0.2963, -0.2962, -0.2968,
           -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2970, -0.2965, -0.3374, -0.3947,
           -0.4132, -0.3293, -0.3139, -0.3353, -0.2964, -0.2960, -0.2954,
           -0.2982, -0.3250, -0.3360, -0.3560, -0.3182],
          [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
           -0.2956, -0.2956, -0.2956, -0.2961, -0.2972, -0.4112, -0.3618,
           -0.4875, -0.4166, -0.3594, -0.3879, -0.3664, -0.3704, -0.3905,
           -0.4559, -0.5388, -0.5372, -0.4580, -0.3625],
          [-0.2963, -0.2962, -0.2968, -0.2963, -0.2968, -0.2974, -0.2968,
           -0.2963, -0.2968, -0.2974, -0.2966, -0.2967, -0.3158, -0.5138,
           -0.4619, -0.5546, -0.5679, -0.3904, -0.4006, -0.4499, -0.5144,
           -0.5866, -0.5755, -0.5312, -0.4466, -0.3495],
          [-0.2982, -0.2965, -0.2992, -0.2986, -0.3015, -0.2958, -0.2954,
           -0.2964, -0.2968, -0.2965, -0.2950, -0.3014, -0.1963, -0.4414,
           -0.5621, -0.4352, -0.5160, -0.4852, -0.3818, -0.4593, -0.4704,
           -0.4620, -0.5409, -0.5621, -0.4192, -0.2707],
          [-0.2994, -0.2979, -0.2983, -0.2961, -0.3805, -0.4714, -0.5451,
           -0.4984, -0.3160, -0.3223, -0.2969, -0.3002, -0.2979, -0.1701,
           -0.3706, -0.5017, -0.4061, -0.5892, -0.5578, -0.3900, -0.4527,
           -0.5890, -0.4772, -0.3332, -0.2891, -0.2756],
          [-0.2942, -0.2961, -0.2972, -0.3395, -0.5337, -0.7015, -0.6664,
           -0.6172, -0.5634, -0.4920, -0.3384, -0.2930, -0.3050, -0.2726,
           -0.1988, -0.3723, -0.5585, -0.4414, -0.5614, -0.5551, -0.5339,
           -0.3976, -0.2565, -0.3164, -0.2674, -0.2964],
          [-0.2990, -0.2982, -0.3224, -0.4583, -0.5242, -0.7647, -0.9021,
           -0.8073, -0.7382, -0.5332, -0.3217, -0.2991, -0.2962, -0.2975,
           -0.2967, -0.2425, -0.4008, -0.5521, -0.6175, -0.4655, -0.3606,
           -0.3136, -0.2740, -0.2886, -0.2968, -0.2958],
          [-0.2982, -0.2966, -0.3059, -0.4033, -0.4576, -0.5026, -0.5123,
           -0.4580, -0.4386, -0.3489, -0.2779, -0.3180, -0.3762, -0.4689,
           -0.5742, -0.5892, -0.5735, -0.5480, -0.5354, -0.4983, -0.6216,
           -0.6196, -0.6092, -0.5971, -0.5997, -0.5226],
```

```
     [-0.3464, -0.3639, -0.3416, -0.3094, -0.4509, -0.4050, -0.2682,
      -0.3765, -0.5684, -0.5003, -0.6914, -0.7594, -0.8117, -0.8752,
      -0.8740, -0.6890, -0.6637, -0.5304, -0.5828, -0.7663, -0.7436,
      -0.8464, -0.8521, -0.8604, -0.8353, -0.6722],
     [-0.6930, -0.8582, -0.8451, -0.8932, -0.7825, -0.8049, -0.5067,
      -0.4655, -0.6895, -0.8101, -0.8135, -0.9355, -0.8476, -0.8333,
      -0.8177, -0.7397, -0.5476, -0.4669, -0.4193, -0.5471, -0.7816,
      -0.8805, -0.8314, -0.8232, -0.8292, -0.7171],
     [-0.6055, -0.6447, -0.6316, -0.5846, -0.5330, -0.4506, -0.5389,
      -0.5841, -0.4664, -0.6082, -0.5997, -0.5289, -0.5747, -0.5608,
      -0.5733, -0.5632, -0.3737, -0.4697, -0.4788, -0.5509, -0.6361,
      -0.5296, -0.5043, -0.4485, -0.4370, -0.4395],
     [-0.3158, -0.1881, -0.1973, -0.1609, -0.3822, -0.3083, -0.3439,
      -0.2688, -0.3738, -0.1708, -0.3178, -0.2502, -0.2841, -0.2513,
      -0.2920, -0.3889, -0.2864, -0.3770, -0.3056, -0.2125, -0.3069,
      -0.2286, -0.2263, -0.2290, -0.1818, -0.3212],
     [-0.2928, -0.2584, -0.2800, -0.2746, -0.2108, -0.2678, -0.2455,
      -0.2282, -0.2612, -0.2430, -0.2427, -0.2550, -0.2348, -0.2463,
      -0.2412, -0.2675, -0.2374, -0.2480, -0.2434, -0.2430, -0.2645,
      -0.2634, -0.2688, -0.2880, -0.2792, -0.3015],
     [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
     [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
     [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
     [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
     [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956],
     [-0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956, -0.2956,
      -0.2956, -0.2956, -0.2956, -0.2956, -0.2956]]]],
   grad_fn=<MkldnnConvolutionBackward>)
```

3-4 학습 세트의 마지막 데이터에 대하여 커널 사이즈 3, 스트라이드 3인 컨볼루션 연산을 적용해서 출력해 보세요.

In [328]:

```
model = nn.Sequential(
            nn.Conv2d(in_channels=1,out_channels=1,kernel_size=3, stride=3)
      )
output = model(mnist_train.__getitem__(59999)[0].reshape(-1,1,28,28))
print(output.shape)
print(output)
```

```
torch.Size([1, 1, 9, 9])
tensor([[[[0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219,
           0.2219],
          [0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219,
           0.2219],
          [0.2219, 0.2219, 0.2219, 0.2219, 0.2217, 0.1906, 0.2221, 0.2219,
           0.2219],
          [0.2221, 0.2221, 0.2215, 0.2211, 0.2846, 0.2454, 0.2724, 0.3250,
           0.3735],
          [0.2264, 0.2336, 0.3747, 0.1772, 0.2251, 0.4004, 0.4442, 0.3838,
           0.2942],
          [0.2073, 0.1903, 0.4867, 0.4669, 0.2072, 0.3469, 0.5069, 0.4970,
           0.4372],
          [0.5155, 0.5952, 0.5634, 0.5239, 0.6676, 0.7166, 0.4423, 0.6638,
           0.4990],
          [0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219,
           0.2219],
          [0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219, 0.2219,
           0.2219]]]], grad_fn=<MkldnnConvolutionBackward>)
```

3-5 학습 세트를 자신이 설계한 CNN 모델로 학습하고 테스트 세트로 정확도를 평가해 보세요.

In [338]:

```
batch_size = 64
learning_rate = 0.0002
num_epoch = 10
```

In [339]:

```
train_loader = DataLoader(mnist_train,batch_size=batch_size, shuffle=True,num_workers=2,drop_last=True)
test_loader = DataLoader(mnist_test,batch_size=batch_size, shuffle=False,num_workers=2,drop_last=True)
```

In [340]:

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN,self).__init__()
        self.layer = nn.Sequential(
            nn.Conv2d(in_channels=1,out_channels=16,kernel_size=5),          # [batch_size,1,28,28] -> [batch_size,16,24,24]
            nn.ReLU(),                                                        # 필터의 개수는 1개(흑백이미지)에서 16개로 늘어나도록 임의로 설정했습니다.
            nn.Conv2d(in_channels=16,out_channels=32,kernel_size=5),         # [batch_size,16,24,24] -> [batch_size,32,20,20]
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2,stride=2),                             # [batch_size,32,20,20] -> [batch_size,32,10,10]
            nn.Conv2d(in_channels=32, out_channels=64, kernel_size=5),       # [batch_size,32,10,10] -> [batch_size,64,6,6]
            nn.ReLU(),
            nn.MaxPool2d(kernel_size=2,stride=2)                              # [batch_size,64,6,6] -> [batch_size,64,3,3]
        )
        self.fc_layer = nn.Sequential(
            nn.Linear(64*3*3,100),                                            # [batch_size,64*3*3] -> [batch_size,100]
            nn.ReLU(),
            nn.Linear(100,10)                                                 # [batch_size,100] -> [batch_size,10]
        )

    def forward(self,x):
        out = self.layer(x)                                                   # self.layer에 정의한 Sequential의 연산을 차례대로 다 실행합니다.
        out = out.view(batch_size,-1)                                         # view 함수를 이용해 텐서의 형태를 [batch_size,나머지]로 바꿔줍니다.
                                                                              # ex) 2x3 형태였던 텐서를 .view(1,-1) 해주면 1x6의 형태로 바뀝니다. .view(3,-1)이면 3x2로 바뀜.
                                                                              # 만약 전체 텐서의 크기가 batch_size로 나누어 떨어지지 않으면 오류가 납니다.
        out = self.fc_layer(out)
        return out
```

In [341]:

```
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
print(device)

model = CNN().to(device)

loss_func = nn.CrossEntropyLoss()

optimizer = torch.optim.Adam(model.parameters(), lr=0.002)
```

```
cuda:0
```

In [342]:

```
loss_arr =[]
for i in range(num_epoch):
    for j,[image,label] in enumerate(train_loader):

        x = image.to(device)
        y_= label.to(device)

        optimizer.zero_grad()
        output = model.forward(x)
        loss = loss_func(output,y_)
        loss.backward()
        optimizer.step()

        if j % 1000 == 0:
            print(j)
            print(image.size)
            print(label.size)
            print(loss)
            loss_arr.append(loss.cpu().detach().numpy())
```

```
0
<built-in method size of Tensor object at 0x0000026A360D9CC8>
<built-in method size of Tensor object at 0x0000026A328FFE08>
tensor(2.3081, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A35F01098>
<built-in method size of Tensor object at 0x0000026A35F01458>
tensor(0.2424, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A3612CD68>
<built-in method size of Tensor object at 0x0000026A35F018B8>
tensor(0.2617, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A35F011D8>
<built-in method size of Tensor object at 0x0000026A328FFB38>
tensor(0.1833, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A366CB278>
<built-in method size of Tensor object at 0x0000026A366CBEF8>
tensor(0.2101, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A328F4EF8>
<built-in method size of Tensor object at 0x0000026A328F4368>
tensor(0.2012, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A36702048>
<built-in method size of Tensor object at 0x0000026A36702EA8>
tensor(0.0992, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A35F01098>
<built-in method size of Tensor object at 0x0000026A35F01C78>
tensor(0.2520, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A3612C318>
<built-in method size of Tensor object at 0x0000026A32D8AB88>
tensor(0.2400, device='cuda:0', grad_fn=<NllLossBackward>)
0
<built-in method size of Tensor object at 0x0000026A32D85868>
<built-in method size of Tensor object at 0x0000026A32D8A9A8>
tensor(0.1388, device='cuda:0', grad_fn=<NllLossBackward>)
```

In [344]:

```
correct = 0
total = 0

with torch.no_grad():
    for image,label in test_loader:
        print(image.shape)
        print(label.shape)

        x = image.to(device)
        y_= label.to(device)


        output = model.forward(x)
        _,output_index = torch.max(output,1)

        total += label.size(0)
        correct += (output_index == y_).sum().float()

    print("Accuracy of Test Data: {}%".format(100*correct/total))
```

```
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
```

```
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
```

```
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
```

```
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
```

```
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
```

```
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
torch.Size([64, 1, 28, 28])
torch.Size([64])
Accuracy of Test Data: 90.63501739501953%
```

3-6 학습 모델의 각 계층별 요약과 파라미터 수를 출력해 보세요.

In [346]:

```
print(model)
print(list(model.parameters()))
```

```
CNN(
  (layer): Sequential(
    (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU()
    (2): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=Fals
e)
    (5): Conv2d(32, 64, kernel_size=(5, 5), stride=(1, 1))
    (6): ReLU()
    (7): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=Fals
e)
  )
  (fc_layer): Sequential(
    (0): Linear(in_features=576, out_features=100, bias=True)
    (1): ReLU()
    (2): Linear(in_features=100, out_features=10, bias=True)
  )
)
[Parameter containing:
tensor([[[[-1.4531e-01,  3.4147e-02,  2.9623e-01, -8.6458e-02, -3.8937e-01],
          [ 5.7209e-02, -1.9056e-01,  4.0041e-02,  1.5629e-01, -4.4271e-02],
          [-7.6948e-02,  2.2185e-01, -2.0429e-01,  4.4132e-02,  1.4092e-01],
          [-3.9637e-01, -1.3540e-01, -1.6280e-01,  2.7554e-01,  2.3662e-01],
          [-1.9212e-01,  1.2840e-01, -3.2989e-01, -6.9444e-02,  1.8193e-01]]],


        [[[ 2.7294e-02, -4.7589e-01, -5.7900e-01,  2.6975e-01, -2.5075e-01],
          [ 1.7635e-01, -2.0237e-01,  5.5234e-02,  1.0652e-01,  2.4122e-01],
          [-1.7315e-01,  2.7480e-01,  1.5789e-01,  1.0703e-01,  3.6340e-01],
          [ 3.2224e-01,  3.2744e-01, -1.2951e-01, -2.3798e-01, -5.9665e-02],
          [-7.8105e-02,  1.5422e-02, -1.1216e-01, -3.6502e-01, -6.7292e-02]]],


        [[[-1.1751e-01,  1.8176e-01, -2.4766e-01, -3.8583e-01, -5.2116e-02],
          [ 2.8013e-01,  3.1039e-01,  3.3151e-02,  1.4126e-01, -1.8540e-01],
          [ 3.5809e-02, -1.4447e-01, -2.5168e-01, -2.3377e-02, -2.1330e-01],
          [-1.2505e-01, -3.1857e-02,  3.4292e-02, -1.3837e-03, -1.8092e-01],
          [ 2.4279e-01,  6.9753e-02, -2.0538e-01,  2.6049e-01,  3.2178e-01]]],


        [[[ 9.0685e-02, -8.9343e-02,  3.6107e-01,  5.3057e-02, -7.6093e-02],
          [-7.0896e-02, -4.5746e-02,  1.5421e-01, -1.3317e-01,  1.0715e-01],
          [-1.7953e-01, -6.5211e-01, -1.7452e-01, -5.5006e-03, -5.1708e-02],
          [ 2.9165e-01, -4.8929e-01, -1.9312e-01,  2.6000e-01,  1.9523e-01],
          [-4.1313e-01, -1.0204e+00, -1.9221e-01, -1.1093e-01, -2.1604e-01]]],


        [[[-1.6108e+00, -1.9704e+00, -6.6881e-02, -1.7168e-02, -1.9123e-01],
          [-1.0149e+00, -2.0719e+00, -5.2867e-01,  1.6332e-03,  2.7052e-01],
          [ 1.0951e-01, -1.0360e-01, -4.7971e-02,  2.9696e-01,  7.8843e-02],
          [ 2.9057e-01, -1.8351e-01,  1.4093e-01,  1.3832e-01, -1.8857e-01],
          [-2.2420e-02,  8.5482e-02, -7.9006e-02,  1.6286e-01, -1.5581e-01]]],


        [[[ 2.7356e-01, -2.1939e-01, -4.4195e-01, -5.4984e-01, -1.4873e-01],
          [ 9.1493e-02, -4.7583e-01, -9.8391e-01, -7.0799e-01, -6.0486e-01],
          [-1.8775e-01,  1.6412e-01, -3.2287e-01, -4.8917e-01, -7.1528e-02],
          [-2.5060e-01,  2.2946e-01, -2.1939e-01,  3.1417e-02, -9.0405e-02],
          [ 9.4968e-02,  1.9434e-01, -1.4728e-01, -1.3437e-01, -1.6256e-01]]],
```

```
[[[ 3.2813e-01,  3.4610e-01, -3.4063e-01, -7.5113e-02, -1.9286e-01],
  [ 2.9235e-02,  6.3908e-02, -3.1199e-01,  1.0557e-01,  1.2317e-01],
  [ 3.4519e-01,  2.2062e-01, -1.2183e-01,  2.4522e-01,  8.8298e-02],
  [-8.4096e-02,  9.2599e-02, -2.5342e-01, -1.3058e-02,  1.4023e-01],
  [-3.3636e-01, -2.8383e-01, -2.2163e-01,  1.9980e-01,  9.2521e-02]]],


[[[ 3.5433e-01,  1.1173e-01, -2.4608e-01, -2.4843e-01,  4.4725e-01],
  [ 2.9631e-01, -1.6360e-01, -2.8739e-01, -2.4491e-01, -7.1777e-02],
  [ 1.4632e-01,  1.9429e-01,  1.6442e-01, -4.9659e-01, -3.8563e-01],
  [-2.3028e-01,  1.3101e-01,  2.5257e-01,  2.0970e-01, -8.1484e-02],
  [ 1.1585e-02, -1.1310e-01, -5.9741e-02, -1.0757e-01,  8.5389e-03]]],


[[[ 2.5859e-01,  6.9984e-03,  4.7666e-01,  3.1789e-01, -9.1624e-02],
  [-1.4078e-01, -2.8714e-01,  9.3490e-02,  3.3990e-01,  2.5606e-01],
  [-1.8003e-01, -2.5264e-02, -1.8540e-01, -2.2347e-01, -7.4464e-02],
  [-4.2390e-02,  3.1273e-01,  2.4786e-01, -2.3436e-01,  1.8449e-01],
  [-3.5500e-02,  1.8784e-01, -3.8592e-01, -4.2744e-01, -7.6374e-02]]],


[[[-3.2243e-01, -8.3613e-02,  3.4216e-02, -2.1899e-01,  6.6986e-02],
  [-3.5930e-01,  2.0227e-01,  5.8903e-02, -9.9545e-02, -1.4667e-02],
  [-1.6249e-01,  2.7105e-01,  1.0258e-01,  7.8456e-02,  1.3377e-01],
  [-2.2144e-01, -1.0188e-01, -1.0036e-01,  2.8555e-01,  1.9500e-01],
  [ 7.4076e-02, -5.6583e-02, -4.9288e-02,  6.7177e-02,  1.9942e-01]]],


[[[-3.6011e-01,  1.3861e-01, -3.4993e-01, -3.6471e-02, -1.4442e-01],
  [-2.6183e-03,  8.8383e-02, -2.3785e-01, -4.1887e-02,  4.8942e-02],
  [-3.7141e-01,  2.5996e-01, -5.1651e-02, -2.1772e-01, -2.4998e-01],
  [ 6.3949e-02,  1.4543e-01,  6.4821e-02,  5.8345e-02,  9.4667e-02],
  [ 4.5410e-02,  2.0171e-01, -3.3624e-02, -3.1360e-01, -3.3569e-02]]],


[[[ 2.6537e-01, -1.8158e-01, -4.1210e-01, -2.6115e-01, -1.0537e-02],
  [ 4.1576e-01, -1.3749e-01, -2.1062e-01, -1.6261e-01,  3.5245e-01],
  [-1.1299e-02, -8.0467e-03,  1.1839e-01, -1.6215e-01, -2.1016e-01],
  [ 1.3118e-01,  4.3101e-02, -4.6419e-03, -1.9834e-01, -4.7616e-02],
  [ 3.4119e-01,  2.0780e-02,  2.1124e-01,  2.0622e-01, -1.3309e-02]]],


[[[ 2.0844e-01, -3.6117e-01, -1.4049e+00, -1.9061e+00, -3.1223e-01],
  [ 4.7640e-02,  1.4241e-01, -1.5562e-01, -4.3921e-01,  2.4755e-01],
  [ 8.8767e-02,  7.6085e-03,  1.1717e-01,  2.7461e-02, -2.8275e-02],
  [ 1.5724e-01, -1.4564e-02, -5.9751e-02, -4.9184e-03, -4.4215e-02],
  [ 4.4611e-02, -9.1736e-02, -8.0361e-02,  1.0041e-01,  9.4391e-02]]],


[[[ 3.3611e-01, -6.7562e-02, -1.3252e-01, -1.9158e-01, -3.1174e-01],
  [ 2.3869e-01, -3.5523e-02,  2.9678e-01,  1.9904e-02, -2.5108e-02],
  [ 3.6405e-01, -3.9163e-01, -6.8804e-02, -2.6310e-02,  1.8177e-01],
  [ 1.1384e-02, -4.1825e-01,  7.9084e-02,  1.8581e-01,  1.3766e-02],
  [ 2.6131e-01, -2.3130e-01, -1.2950e-01,  6.5088e-02, -3.3412e-02]]],


[[[-5.6683e-01, -2.8870e-01, -3.0342e-01,  4.4500e-03, -8.6476e-02],
  [-1.5840e-01,  2.3334e-01,  2.5644e-01,  1.6673e-02, -6.9473e-02],
  [-8.3825e-02,  5.2112e-02, -9.9952e-02, -1.3713e-01,  3.4042e-02],
  [-4.6784e-02, -6.3275e-02,  2.2482e-01, -1.0167e-01, -1.7375e-01],
```

```
          [-1.2079e-01,  1.6458e-02,  1.2365e-01,  1.4317e-01, -2.6381e-01]]],


         [[[ 1.4256e-02,  1.3537e-01, -2.0107e-01, -6.5872e-03,  2.0957e-01],
           [ 1.3585e-01, -7.8220e-02, -1.7565e-02, -2.3473e-01, -6.9731e-02],
           [ 4.8532e-03, -2.2437e-01,  1.1118e-01, -1.5984e-01, -1.1862e-01],
           [-7.9847e-02, -1.4142e-01,  1.8507e-01,  1.7620e-01,  1.2232e-03],
           [ 1.4163e-01, -1.3468e-01, -2.5267e-01,  7.0737e-02, -1.1688e-01]]]],
        device='cuda:0', requires_grad=True), Parameter containing:
tensor([ 0.0533,  0.0098, -0.0970,  0.2878,  0.2335,  0.2209,  0.0689, -0.0745,
         -0.1249, -0.3333, -0.0831, -0.0094,  0.2188,  0.0299,  0.2554, -0.1144],
        device='cuda:0', requires_grad=True), Parameter containing:
tensor([[[[ 3.7907e-01,  1.1135e-01,  1.7796e-01, -1.5060e-01, -1.8196e-02],
          [-4.7649e-02, -5.5816e-02,  3.2733e-01,  3.4180e-03, -2.2717e-01],
          [ 6.0282e-02, -4.4016e-02,  1.3167e-01, -2.1346e-01, -2.8922e-02],
          [-7.1203e-02, -3.5062e-02, -1.0480e-01, -9.0784e-02, -2.3246e-01],
          [ 6.0349e-02, -3.6000e-01, -4.1288e-01, -2.1018e-01, -1.3647e-01]],

         [[-4.2256e-01, -3.5724e-01,  2.6515e-01, -1.1727e-01, -1.2595e-02],
          [-9.0640e-02,  1.8330e-01,  2.9015e-02, -5.0176e-01, -1.4238e-01],
          [ 3.7684e-01, -1.3304e-01,  2.2754e-02, -3.9411e-02,  6.9121e-02],
          [ 3.2043e-01, -1.2479e-01, -1.6502e-01, -4.4898e-02, -7.9888e-02],
          [ 1.0053e-01, -1.8701e-01, -1.1221e-01,  3.1846e-02, -1.9369e-01]],

         [[-6.6177e-02, -2.6726e-02, -4.7412e-02, -1.2799e-01, -6.6989e-03],
          [ 6.1112e-02, -2.4974e-02, -4.0632e-01, -8.6513e-02,  2.0335e-02],
          [-3.1447e-02, -2.0853e-03, -3.7839e-01, -1.7232e-02,  9.0986e-03],
          [-2.5828e-02, -3.9749e-01, -6.0977e-02,  5.8132e-02, -2.9356e-02],
          [-3.9659e-01, -1.8325e-01,  1.5151e-01, -1.0118e-01,  1.4636e-01]],

         ...,

         [[ 4.6093e-01,  1.4339e-01, -7.4836e-01,  1.9122e-01,  4.5870e-02],
          [ 1.7670e-01, -5.0655e-02, -7.0977e-01,  2.8633e-01,  1.2689e-02],
          [ 1.3943e-02,  2.9977e-01, -6.6006e-01,  3.4387e-01,  1.6696e-01],
          [-6.7065e-02,  2.3433e-01, -8.8311e-01,  2.5904e-01,  1.4469e-01],
          [ 1.5445e-01,  4.6039e-01, -5.8045e-01, -1.5119e-02,  3.4588e-02]],

         [[ 6.7033e-02,  1.7507e-01,  2.1018e-01,  1.4839e-01,  1.3694e-01],
          [ 1.3527e-01,  7.0550e-02,  6.6860e-02,  7.2581e-02,  3.1771e-03],
          [ 5.3790e-02,  1.4882e-02,  9.1508e-02, -5.3823e-02,  1.7977e-02],
          [ 4.2061e-02, -6.3887e-02,  6.8814e-02, -1.5254e-01,  1.2309e-01],
          [-5.1933e-03,  1.8536e-01,  2.5542e-01, -1.1512e-01, -4.2601e-02]],

         [[-6.6818e-03, -8.6306e-02,  5.2961e-03,  1.0123e-01,  5.3744e-03],
          [-5.6748e-02, -4.8725e-02, -8.7623e-02,  1.0803e-01,  1.0440e-01],
          [-1.2555e-01, -5.3539e-02, -7.1951e-02,  5.7748e-02, -5.9818e-03],
          [-1.0337e-01, -1.1240e-01, -4.8049e-02, -2.8046e-02, -5.7603e-02],
          [-1.8986e-02, -1.1162e-01, -7.2713e-03,  7.1562e-02, -3.2732e-02]]],


        [[[ 1.1930e-01, -2.0872e-01, -3.1707e-01, -5.8418e-02,  1.3489e-02],
          [ 2.0736e-01, -1.5497e-02, -3.7585e-01, -1.6999e-01,  2.0573e-02],
          [ 2.6230e-01,  1.1434e-01, -2.8244e-01, -3.3611e-02, -2.4460e-03],
          [ 1.6277e-01,  1.6383e-01, -2.2970e-03,  2.8938e-02,  8.5519e-02],
          [ 5.6431e-02,  1.1961e-01,  3.8325e-03, -2.3319e-02,  1.0174e-01]],

         [[-2.5659e-01, -2.5171e-02,  2.4346e-01,  1.6814e-01, -3.5818e-02],
          [-1.0567e-01, -2.0385e-01, -3.2312e-02, -2.3002e-02, -3.5035e-01],
          [-6.5678e-02, -7.7213e-01, -4.4719e-01, -2.2638e-01, -3.2248e-01],
          [ 2.5931e-01, -3.5928e-01, -1.7730e-01, -4.1068e-01, -3.7119e-01],
```

```
          [ 5.1281e-01, -2.1621e-03, -3.1249e-01, -3.6358e-01,  6.7213e-02]],

         [[ 2.2985e-02, -2.4204e-01, -3.0621e-01, -1.2788e-01, -3.2316e-01],
          [ 6.2540e-02,  4.9709e-02,  5.3227e-02,  2.4817e-01,  1.4696e-02],
          [-2.0556e-01,  2.6815e-02,  1.1866e-01,  1.8372e-01,  1.9141e-01],
          [-4.6615e-02,  6.6212e-02, -2.7286e-01,  6.1252e-02,  8.5225e-02],
          [-7.0346e-02,  1.3703e-01, -1.5364e-01,  6.0361e-02, -6.0719e-02]],

         ...,

         [[ 3.1884e-01, -9.0638e-02, -1.7256e-01, -4.2016e-01, -2.8319e-01],
          [ 4.4530e-01,  3.0127e-01, -1.3784e-01, -4.2668e-01, -2.8100e-01],
          [ 2.9691e-02,  1.1258e-01,  2.3097e-03,  4.1738e-02, -1.4641e-02],
          [ 8.4407e-02,  1.0396e-01, -4.4328e-01, -1.8867e-01,  1.4876e-01],
          [-4.3124e-01,  2.3196e-01, -2.4098e-01, -2.7797e-01,  2.1270e-01]],

         [[-3.2794e-02,  1.6369e-01,  6.9687e-02,  2.1505e-01,  3.4235e-01],
          [-3.5534e-02,  2.3058e-01,  2.3773e-01,  1.9922e-01,  2.7390e-01],
          [-2.1820e-01,  1.2365e-01,  1.0967e-01, -1.5900e-01, -1.2566e-01],
          [-1.6178e-01,  1.0451e-02,  1.8935e-01, -4.8464e-02, -3.0000e-01],
          [-8.0505e-02, -9.9697e-02,  2.0291e-01,  1.2628e-02, -4.7622e-02]],

         [[ 2.0747e-01, -6.9679e-02, -1.1715e-01, -1.1378e-01, -1.4560e-01],
          [-5.7590e-02, -6.6412e-02, -1.0255e-01, -3.5442e-02, -1.9385e-03],
          [-1.2525e-01, -8.5566e-03,  1.1323e-04,  6.6165e-02,  5.6902e-02],
          [-9.4973e-02,  3.5019e-02, -8.1128e-02,  5.4194e-02,  1.0373e-01],
          [-1.6112e-01,  9.8190e-03, -9.2947e-02, -1.3436e-02,  7.4021e-02]]],


        [[[-1.2289e-01, -4.7996e-02,  2.0719e-01, -1.7001e-01, -4.3139e-01],
          [-6.6756e-02, -1.3980e-01,  1.1368e-01,  1.9544e-02, -3.1018e-01],
          [ 7.2131e-02, -2.9680e-01,  1.0601e-01,  9.6820e-05, -1.4670e-01],
          [ 3.1240e-01, -3.3409e-02,  1.0468e-01, -1.1046e-01,  2.8626e-03],
          [ 2.7103e-01, -3.5254e-02,  1.8533e-01, -2.8091e-01, -1.6617e-01]],

         [[-2.3263e-01, -1.9923e-01, -3.9431e-01, -5.8445e-01, -5.4573e-01],
          [-2.9146e-01, -4.6275e-01, -3.5840e-01, -3.2422e-01, -6.8831e-01],
          [-5.0409e-01, -8.8668e-01, -2.8369e-02,  3.5637e-05,  2.0013e-01],
          [-5.9385e-01, -3.4449e-01, -5.3982e-01, -1.5333e-01,  1.0369e-01],
          [-8.4390e-02,  1.0782e-02, -3.4186e-01, -4.6358e-01, -1.2821e-01]],

         [[-8.6938e-01, -3.1672e-01,  1.4155e-01,  2.4803e-01,  4.6049e-01],
          [-6.2673e-01, -5.0721e-01, -1.9698e-01, -6.9302e-02,  2.9725e-01],
          [-5.4614e-01, -3.7411e-01, -2.8177e-02, -3.7028e-02,  2.0093e-01],
          [-6.5434e-01, -3.2313e-01, -8.4631e-02, -2.3059e-01, -1.2615e-01],
          [-3.9432e-01, -1.4851e-01, -1.1816e-01, -2.5277e-01, -2.5025e-01]],

         ...,

         [[-1.8392e-01, -2.7647e-01, -8.5689e-02,  1.9373e-01,  2.7803e-01],
          [-4.2472e-02, -4.6626e-01, -2.9102e-02,  6.2859e-02, -4.1870e-02],
          [ 1.9728e-03, -4.4288e-01,  1.3586e-01,  4.7284e-01,  2.5678e-01],
          [ 1.7046e-01, -2.2522e-01, -1.2062e-01,  1.2358e-01,  1.8601e-01],
          [ 4.2168e-01, -3.1802e-01, -2.1772e-02, -2.1618e-02, -1.9083e-01]],

         [[ 1.0985e-01,  2.5755e-01,  1.6452e-01,  2.0983e-02, -2.7608e-01],
          [ 1.5994e-01,  1.1940e-01, -3.9562e-02,  5.0537e-02, -3.4195e-02],
          [ 3.2635e-02,  2.5346e-02, -4.0000e-02,  1.6486e-01,  1.6963e-02],
          [ 2.5270e-02,  6.9480e-02, -6.5975e-02,  2.4702e-01,  4.7033e-02],
          [ 8.9337e-02, -8.0496e-02, -2.0535e-01,  6.4377e-02, -1.0106e-01]],
```

```
 [[-2.0336e-01, -8.1713e-02,  3.6229e-02,  1.0614e-01,  3.1714e-02],
  [-4.4016e-02, -8.0086e-02,  2.6700e-02,  5.1467e-04,  1.6905e-03],
  [-2.3517e-03, -1.1434e-01,  2.3241e-02, -7.7704e-02, -3.8401e-02],
  [-7.0609e-02, -1.0090e-01,  7.0390e-02, -4.4287e-03, -1.9920e-01],
  [-2.3072e-01, -1.0100e-01,  1.4891e-01, -6.7152e-03, -1.7633e-01]]],


...,


[[[-1.1237e-01, -3.1215e-01, -1.1965e-01,  3.7912e-01,  4.5194e-03],
  [-5.1809e-02,  3.5150e-04, -3.0566e-01, -1.2134e-01,  7.6947e-02],
  [-2.6228e-01,  8.8171e-02, -1.1973e-01, -9.3091e-02, -1.3045e-01],
  [-3.6517e-01, -1.8668e-01, -3.1634e-01,  2.0135e-03, -9.8297e-02],
  [-1.2643e-01, -4.2339e-02, -3.2894e-01, -4.0695e-01, -1.7564e-01]],

 [[ 2.3383e-01, -2.7096e-01,  7.6615e-02,  5.4690e-01, -2.7326e-01],
  [ 3.1803e-01, -4.3617e-01, -4.9175e-01,  2.2330e-01, -5.9484e-02],
  [ 3.4116e-01, -5.8588e-01, -4.7675e-01,  2.3775e-01,  9.1438e-02],
  [ 4.0255e-01,  1.4654e-01, -2.4794e-01, -2.6550e-01,  2.2717e-01],
  [ 1.6129e-01,  1.4490e-01, -3.2064e-01, -4.0890e-01,  2.0431e-01]],

 [[-1.0482e-01, -5.2257e-02, -2.2861e-02, -1.3259e-01, -2.7999e-01],
  [-1.5641e-01,  8.4361e-02,  1.1718e-01, -7.6963e-02, -1.3258e-01],
  [-1.6017e-02,  8.6500e-02,  9.4839e-02, -2.3639e-02, -1.0500e-01],
  [-3.3033e-02, -7.5759e-02, -1.2478e-01, -9.5227e-02, -5.5493e-02],
  [-2.8164e-02,  4.9005e-02,  2.5165e-03, -6.1154e-02,  6.8490e-02]],

 ...,

 [[ 1.0737e-03,  1.5536e-01, -3.3820e-01,  1.4555e-01, -1.2772e-01],
  [ 1.5427e-01,  1.8531e-01, -1.0527e-02, -3.0443e-01, -1.9680e-01],
  [ 5.0982e-02,  1.1846e-01,  2.3156e-01, -2.9859e-01, -6.4638e-02],
  [ 3.9436e-02, -3.0847e-02, -6.1513e-03, -4.4507e-01, -5.4470e-02],
  [ 1.1413e-01,  2.3340e-01,  2.8852e-01, -3.7074e-01, -8.2058e-02]],

 [[-8.4386e-02,  1.0513e-01,  1.3990e-01, -5.4335e-02,  6.5745e-02],
  [-7.0038e-02,  6.5608e-02,  1.6362e-01, -1.8939e-01,  4.7958e-02],
  [ 1.9788e-02, -2.9201e-02,  1.1028e-01,  1.0026e-01,  3.1582e-02],
  [-2.7719e-02, -8.0013e-03,  1.7489e-01,  2.0367e-01, -1.2699e-02],
  [-8.7755e-02, -3.4640e-02,  1.2190e-01,  1.3176e-01, -6.6278e-02]],

 [[-1.7775e-01, -2.4203e-02,  2.2888e-02, -1.7540e-01, -1.1461e-01],
  [-2.9428e-01, -1.7075e-01,  1.2794e-01, -1.0103e-01, -4.2919e-02],
  [-1.6403e-01,  9.0290e-03,  8.3227e-02, -3.4631e-02, -4.8183e-02],
  [-1.2331e-01, -1.0385e-01,  6.6548e-03,  4.7799e-02, -1.2030e-01],
  [-9.2759e-02, -1.8377e-02,  7.1734e-02,  3.0184e-02, -9.3098e-02]]],


[[[ 6.9284e-02, -1.5523e-01, -2.7744e-01,  1.4745e-01,  3.1750e-01],
  [-8.4093e-03, -1.1634e-02, -1.0116e-01, -5.1677e-02,  2.9343e-01],
  [ 1.4600e-01,  6.3250e-02, -1.5657e-03, -1.0739e-01,  1.7835e-01],
  [-4.7191e-02, -2.1228e-02, -2.8015e-01, -1.1758e-01,  8.2982e-02],
  [-1.9971e-01, -2.9723e-01, -4.1300e-01, -1.8196e-01, -2.4188e-01]],

 [[ 2.5396e-01,  6.1544e-02,  5.2186e-02, -2.6233e-01, -2.1556e-01],
  [-2.5600e-01, -1.7845e-01, -5.4882e-01, -2.0045e-01, -2.3243e-01],
  [-3.0394e-01, -1.5207e-01, -4.3276e-01, -2.5090e-01,  6.6223e-02],
  [-8.6100e-02,  3.2427e-02, -1.1789e-01,  4.5671e-02,  1.7864e-01],
  [-5.4861e-02,  6.7316e-02,  4.4852e-01,  2.7953e-01,  1.1947e-02]],
```

```
[[-9.2157e-02, -1.5441e-01, -1.6779e-01, -2.2121e-01,  2.2751e-01],
 [ 2.2518e-01,  1.5695e-01,  1.1147e-01, -1.0316e-02,  2.9512e-01],
 [ 2.0792e-01, -6.2260e-02, -2.4242e-01, -3.7323e-01, -2.8983e-01],
 [ 1.3211e-01, -9.2249e-03, -4.8295e-03, -1.1335e-01, -2.0730e-01],
 [ 1.8776e-01,  2.5284e-01, -1.3799e-01, -1.3281e-01, -1.1922e-01]],

...,

[[-2.7087e-01, -2.5118e-01, -3.6024e-01, -1.9143e-01,  8.3009e-02],
 [ 2.3918e-02, -1.7148e-02, -7.7388e-02, -1.6237e-01, -7.2539e-02],
 [ 1.1447e-01,  3.0299e-02, -5.0456e-02, -2.3929e-01, -1.4988e-01],
 [ 2.2252e-01,  2.4238e-01,  1.5872e-01,  7.2744e-02,  1.5817e-01],
 [ 2.3268e-01,  1.1393e-01,  7.2797e-02, -7.6463e-02, -2.2536e-01]],

[[ 6.8746e-02,  1.0539e-01, -1.7870e-01, -2.1531e-01,  2.2006e-02],
 [-1.2500e-01, -1.0368e-01, -3.0129e-02, -2.0355e-01, -1.0975e-01],
 [-2.2997e-01, -7.1618e-02, -2.3804e-01, -2.9234e-01,  7.5711e-03],
 [-3.1291e-01, -1.1200e-01, -3.1372e-01, -1.6213e-01,  3.5446e-01],
 [-6.4041e-02, -1.8735e-01, -1.0293e-01, -1.7346e-01, -3.0740e-02]],

[[-1.8350e-01, -1.4629e-01, -7.6374e-02, -2.1014e-01, -7.7669e-02],
 [-7.2691e-02,  1.3014e-03, -1.2689e-01, -1.1512e-01, -1.3832e-01],
 [-6.0089e-02,  1.0396e-02, -7.0854e-02, -7.3984e-02, -5.0987e-02],
 [-3.6336e-02, -1.0460e-02, -8.9678e-02, -3.9532e-02,  2.1872e-02],
 [ 4.9038e-02,  9.1269e-02, -3.5983e-03, -5.5865e-02,  1.5704e-02]]],


[[[-1.0415e-01, -1.4116e-01, -3.7911e-01, -4.5096e-01, -2.0363e-01],
  [-2.8857e-01, -3.3587e-01, -1.7380e-01, -1.4530e-01, -3.3483e-01],
  [-2.5981e-01,  2.3174e-02, -2.2821e-03, -4.8301e-01,  7.2607e-02],
  [-2.9011e-01,  6.8773e-02,  3.5941e-02, -3.3320e-01,  1.3468e-01],
  [ 8.2824e-02, -1.5044e-01, -5.7803e-02, -9.1797e-02,  6.7818e-02]],

 [[-7.5632e-02, -2.3676e-02,  1.6105e-01,  6.5091e-02, -1.9906e-01],
  [-3.6921e-01, -1.1230e-01, -1.4719e-01,  7.0465e-04, -1.0012e-02],
  [-2.5590e-01,  5.5239e-02,  1.9219e-01,  1.0357e-01,  1.0027e-01],
  [ 1.5298e-01,  4.2260e-01,  2.9612e-01,  1.1828e-01,  1.1387e-01],
  [-5.4972e-01,  1.1692e-01,  7.3910e-02, -1.9786e-01, -5.5518e-01]],

 [[-1.1550e-01,  1.8750e-02, -1.1985e-02,  5.2922e-03,  7.3701e-02],
  [-1.8743e-02, -1.4830e-01, -3.8253e-02, -7.4191e-02, -1.3771e-01],
  [-4.2067e-01, -5.4324e-01, -4.7354e-01, -2.3941e-01, -3.6072e-01],
  [ 1.4149e-01,  1.9415e-02,  1.8385e-01,  1.5613e-01,  1.3265e-01],
  [-2.3013e-01, -2.7726e-01,  1.4390e-01,  8.4125e-02, -1.6408e-02]],

 ...,

 [[ 1.8832e-01,  3.3323e-01, -2.6141e-01, -2.6726e-01, -2.0944e-01],
  [ 1.2128e-01, -2.6713e-01, -3.7167e-01, -1.5900e-01, -2.6005e-01],
  [ 2.0735e-01, -6.3287e-02, -2.8468e-01, -9.1837e-02, -3.9749e-02],
  [ 3.4096e-01,  3.5642e-02,  2.3264e-01,  3.7094e-01,  1.7259e-01],
  [ 2.0441e-01, -4.3426e-01,  8.1615e-03,  1.5929e-01, -2.6731e-01]],

 [[ 6.3257e-02, -9.9706e-02,  2.2107e-01,  4.6050e-01,  1.9718e-01],
  [ 1.9366e-01, -2.1785e-03,  2.7681e-02,  2.2802e-01,  7.1511e-02],
  [-1.1266e-02, -4.9440e-02, -1.4161e-01, -5.7950e-02, -1.3993e-01],
  [-2.3834e-01, -1.3940e-01, -3.0257e-02,  2.9625e-01,  1.5383e-01],
  [-1.3291e-01, -4.3497e-02, -2.6888e-01, -2.2534e-02,  5.0894e-02]],

 [[-1.0809e-01, -1.3236e-01, -2.4344e-01, -1.4836e-01,  2.1636e-02],
  [ 1.0071e-02,  5.0059e-02,  8.2792e-02,  5.0396e-02,  5.6064e-02],
```

```
          [ 8.0546e-02, -6.6471e-02, -8.0535e-02,  5.5243e-02, -1.2448e-01],
          [-1.3630e-02, -7.9736e-02, -1.2770e-01, -3.2113e-02,  1.5166e-02],
          [ 1.0531e-01, -4.4577e-02,  4.0463e-02,  8.0294e-02, -5.9821e-02]]]],
       device='cuda:0', requires_grad=True), Parameter containing:
tensor([ 0.0310, -0.0967,  0.1560, -0.0840, -0.0917,  0.1356,  0.0056, -0.1281,
        -0.0924, -0.0454,  0.2033, -0.0987, -0.0682,  0.0602,  0.0336,  0.1099,
         0.2071, -0.0095, -0.0628,  0.0655, -0.2318,  0.2747, -0.1478,  0.0986,
         0.1728,  0.0249,  0.0048, -0.1001,  0.3206, -0.1632,  0.1593,  0.0332],
       device='cuda:0', requires_grad=True), Parameter containing:
tensor([[[[-1.0215e-01, -8.9664e-02, -4.6588e-01, -8.3055e-01, -1.3872e+00],
          [ 1.2996e-01,  5.7016e-02, -8.2641e-02, -2.7731e-01, -4.8064e-01],
          [-3.3504e-02,  6.0037e-02, -1.3245e-01, -5.0567e-01, -3.2095e-01],
          [-1.1757e-02,  4.1050e-02, -2.7802e-02, -3.7965e-01, -5.0318e-01],
          [ 2.0446e-01,  1.0446e-01,  8.6335e-02, -2.9193e-01, -3.8343e-01]],

         [[ 3.9526e-01, -3.4353e-02, -9.7762e-02, -5.0698e-02, -7.5802e-02],
          [ 2.0790e-01, -6.6414e-02, -1.9251e-01, -5.9093e-02,  3.8516e-02],
          [ 1.6610e-01, -1.7752e-01, -2.7589e-01, -2.7690e-01, -1.2750e-01],
          [ 5.2681e-03, -5.2361e-01, -2.0877e-01, -3.0563e-01, -2.0447e-01],
          [-1.4163e-01, -4.7684e-01, -1.4215e-01, -2.0110e-01, -1.4445e-01]],

         [[ 8.6792e-02,  2.5503e-01, -2.2947e-01, -1.6544e-01, -6.7644e-02],
          [ 1.6205e-01,  9.6010e-02, -6.3292e-02, -2.1384e-01, -1.4785e-01],
          [ 1.5580e-01,  1.5417e-02,  3.8225e-02, -2.3994e-01, -4.4096e-01],
          [ 2.9324e-02, -1.4117e-01, -4.1396e-02, -1.6509e-02,  2.6495e-02],
          [ 1.3486e-01,  5.9475e-02,  1.7259e-01,  1.6568e-01,  1.2736e-01]],

         ...,

         [[-5.2014e-02, -3.0366e-01, -7.6196e-02,  3.0675e-02, -1.1543e-01],
          [-3.2047e-02, -2.2287e-01, -7.5387e-02, -7.7950e-02, -2.2592e-01],
          [-3.5127e-01, -1.8013e-01,  1.8718e-01, -1.2381e-01, -1.6173e-01],
          [-2.6385e-01,  1.1390e-01,  1.0017e-01, -1.4459e-02, -7.0881e-02],
          [-2.3939e-01, -6.1846e-02, -1.7365e-01,  2.4554e-01,  2.4331e-01]],

         [[-1.1279e-01,  9.6252e-02,  3.2090e-02, -1.3676e-02, -2.2390e-01],
          [ 1.8566e-02,  1.6007e-01,  2.4147e-01, -8.2463e-02,  2.8448e-02],
          [ 7.5015e-02,  1.5685e-01,  5.5735e-02, -2.5004e-02, -1.2314e-01],
          [-2.8695e-02, -2.2827e-02,  5.3546e-02, -1.2718e-01, -1.4476e-01],
          [ 1.2711e-02,  1.9088e-01,  1.1590e-01,  1.1080e-02,  1.6781e-03]],

         [[ 4.6365e-01,  9.3120e-02, -3.7639e-01,  2.8456e-01, -1.2984e-02],
          [ 1.9729e-01, -2.1560e-02,  9.8053e-02,  2.7224e-01,  2.4069e-01],
          [-2.2030e-01, -1.8886e-01,  7.7912e-02, -1.2606e-01, -3.5226e-02],
          [-2.7612e-01, -1.7283e-01, -1.6504e-01,  2.1046e-01, -7.3019e-02],
          [-2.2052e-01,  2.9779e-01, -3.8900e-02, -2.8311e-02,  1.1807e-01]]],


        [[[-9.1042e-02, -1.3909e-01,  3.3711e-01, -2.4374e-01,  6.7585e-02],
          [ 1.8401e-02, -3.1062e-02,  2.1392e-01, -1.1411e-01,  1.1285e-01],
          [-9.3632e-02,  4.5029e-02,  1.5084e-01,  1.1476e-01,  5.8524e-02],
          [-2.1933e-01, -2.2468e-01,  1.4327e-01,  1.8071e-01,  1.8638e-03],
          [-3.5978e-01, -1.6111e-01, -1.0625e-02,  2.1159e-01, -7.7753e-02]],

         [[-1.1246e-01, -2.4236e-01,  3.0732e-01,  3.1311e-01,  4.2516e-02],
          [-1.1727e-01, -1.6091e-01,  1.3868e-01,  3.6190e-01, -5.5658e-02],
          [-6.1410e-02, -2.1501e-01,  2.1259e-01, -6.0000e-02, -2.0244e-01],
          [ 1.7713e-02, -3.1171e-01,  2.3625e-01,  1.7944e-01, -4.7521e-01],
          [ 2.6423e-01, -2.1951e-01, -1.9832e-01,  9.0760e-02, -5.4900e-02]],

         [[-3.3618e-02,  1.6840e-01, -1.0864e-01, -6.9150e-01,  1.4442e-02],
```

```
 [ 9.3212e-02, -5.0117e-02, -1.3319e-01, -3.9276e-01,  6.8052e-02],
 [-5.2711e-01, -1.4380e-01, -6.1960e-03, -2.4085e-01, -2.2174e-01],
 [-1.5697e-01, -3.2869e-01, -2.8765e-01, -4.1833e-01, -2.6543e-01],
 [-3.6946e-02, -2.3381e-01, -5.1371e-02, -1.2953e-01, -3.4868e-01]],


...,

[[-3.5153e-01, -3.4408e-02,  3.7401e-01,  2.2914e-01, -8.6094e-02],
 [-1.7816e-01, -1.7264e-01,  5.4988e-01,  2.0088e-01,  9.9666e-02],
 [ 4.3800e-02, -1.2008e-01,  2.8620e-01, -1.0740e-02, -8.7155e-02],
 [ 9.3658e-02, -2.6104e-01,  2.6141e-01,  1.8978e-01, -2.1705e-01],
 [ 3.0789e-01, -2.7612e-01, -1.3121e-02, -8.1370e-03, -4.5735e-02]],

[[ 9.2550e-02, -1.8227e-01,  4.7098e-02, -1.7951e-01, -4.1198e-01],
 [-1.6194e-01, -3.2464e-01, -1.7475e-01, -3.0187e-01, -4.0922e-01],
 [ 1.0751e-02, -2.3151e-01,  2.0285e-01,  4.0789e-02,  2.7006e-02],
 [ 5.7054e-01,  1.6072e-01, -1.2832e-01, -1.4531e-01, -2.3954e-01],
 [ 1.7826e-01,  2.1798e-01, -1.0115e-01, -1.5728e-01, -1.8883e-01]],

[[ 1.7055e-01, -1.3315e-01, -1.6691e-01, -6.0119e-02,  5.3524e-02],
 [ 4.2417e-02, -5.0431e-02, -1.5076e-01, -2.3217e-01, -5.7641e-02],
 [ 2.0965e-01, -9.1702e-02,  2.8210e-01, -3.2845e-03, -2.0081e-01],
 [ 2.5185e-01,  8.7125e-02,  2.3257e-02, -9.8673e-02, -2.2658e-01],
 [ 2.7801e-01,  4.2366e-02, -2.6416e-01, -7.1900e-02,  1.7522e-01]]],


[[[-2.0906e-01,  1.0161e-01, -2.1026e-01, -2.0877e-02, -1.1229e-01],
  [-7.0384e-01,  8.6930e-02, -2.5646e-01,  3.3410e-02,  6.9244e-02],
  [-6.8033e-01, -1.2845e-02, -2.9124e-02,  1.2764e-01, -5.6210e-02],
  [-2.8132e-01,  8.1631e-02, -7.2651e-02,  1.7358e-01, -2.3652e-01],
  [-4.0610e-01, -6.4953e-03, -6.8387e-02, -4.3565e-02, -3.8099e-01]],

 [[-1.5278e-01, -1.0126e-01, -1.8362e-01, -1.8272e-01, -2.1895e-01],
  [ 2.1012e-02, -2.9658e-02, -1.5685e-01,  1.2914e-02,  1.2389e-01],
  [-7.7721e-02, -3.1477e-02,  1.9051e-02,  5.9399e-02,  1.8480e-01],
  [ 7.9510e-02,  2.9587e-01,  1.2667e-01,  5.8623e-02,  1.0543e-01],
  [-9.9708e-02,  3.1210e-01, -1.2285e-01, -7.2532e-02, -1.0095e-01]],

 [[-1.8138e-01, -1.9289e-01, -3.9306e-01, -2.0940e-01, -1.8766e-01],
  [-2.0792e-01, -1.7747e-01, -4.7055e-01, -3.8495e-01, -1.6154e-01],
  [-1.2623e-01,  2.5011e-02, -3.3273e-01, -2.6538e-02, -9.4016e-02],
  [-2.2855e-01,  2.0120e-02,  2.0003e-02,  2.0000e-02, -1.3054e-01],
  [-5.0618e-01,  3.2521e-01,  3.0346e-01,  5.0409e-02, -3.4867e-01]],


 ...,

 [[ 1.6765e-02, -2.6025e-02, -1.5120e-01, -1.4923e-01, -5.3137e-01],
  [ 2.7754e-02,  1.3643e-01,  3.4715e-02,  3.6979e-02, -2.6798e-01],
  [-1.0008e-01,  2.9039e-01,  1.1799e-01, -2.7590e-03, -2.0527e-01],
  [ 5.8264e-02,  4.2138e-01,  1.6050e-02,  1.4176e-01, -1.7972e-01],
  [-2.9461e-01,  2.9275e-01,  1.0509e-01,  1.3328e-01, -1.6687e-01]],

 [[-2.8700e-01,  1.4529e-01, -5.2216e-02,  5.0124e-02, -1.7897e-01],
  [-7.3637e-02, -1.3581e-01, -2.4172e-01, -3.3447e-01, -4.2120e-01],
  [ 5.0621e-03, -2.8647e-01, -4.8997e-02, -3.5640e-01, -2.8012e-01],
  [-2.0021e-01, -3.7684e-01, -5.9267e-02,  1.6466e-02, -9.4508e-02],
  [-1.2774e-01,  3.1215e-02, -5.5207e-02, -2.4536e-01, -2.1002e-01]],

 [[ 2.5077e-02,  9.6354e-02,  1.0663e-01, -3.4198e-02,  3.9891e-01],
  [-2.1832e-01, -3.6060e-02,  1.3765e-01, -3.1929e-01, -1.0646e-02],
  [ 1.0275e-01,  1.4687e-01,  1.2552e-01, -2.1630e-01, -1.1147e-01],
```

```
          [ 9.6264e-02, -1.5790e-01, -1.3941e-01, -4.3647e-02, -3.4951e-02],
          [-2.8664e-01, -2.4633e-01, -6.8482e-02,  1.3487e-01,  2.4759e-01]]],


        ...,


        [[[-4.7117e-02, -4.1656e-02, -3.8384e-02, -8.1896e-02, -5.1965e-02],
          [-3.3246e-02, -4.2736e-02, -6.8537e-03, -1.2650e-01, -2.5682e-02],
          [-4.4194e-03, -4.3388e-02,  7.1514e-02, -1.1380e-01,  1.6794e-02],
          [-1.3240e-02,  2.2805e-02, -3.5745e-02, -1.1145e-01, -3.9085e-02],
          [-1.3640e-02,  1.5596e-02, -3.0437e-02, -8.4616e-02, -3.9608e-03]],

         [[ 9.2372e-04, -5.4998e-02, -4.9994e-02, -1.0246e-01,  3.8864e-03],
          [-2.6071e-02, -2.4868e-03, -1.5253e-02, -5.0420e-02, -3.3239e-03],
          [-1.7441e-02, -2.3423e-02,  2.5868e-02, -1.8375e-02, -2.3134e-02],
          [-1.1401e-02,  1.3388e-02,  4.1576e-02, -3.4947e-02, -6.0724e-02],
          [-3.9179e-02, -3.3645e-02,  1.9318e-02, -3.1440e-02, -1.3620e-02]],

         [[-4.4584e-02, -5.4867e-03, -3.0030e-02, -3.9725e-02, -5.4499e-02],
          [-3.5481e-02,  6.1201e-03,  1.4517e-02, -1.8325e-02, -8.1779e-02],
          [-1.4222e-02, -1.3229e-02, -4.1548e-03, -2.5895e-02, -7.0552e-02],
          [-4.2668e-02, -5.1316e-02, -1.8444e-02, -2.0367e-02, -4.9372e-02],
          [-3.0172e-02,  2.4492e-03, -5.6030e-02, -1.9308e-02, -2.9788e-02]],

         ...,

         [[-2.1905e-02, -2.0418e-02, -3.4552e-02, -9.0978e-02, -7.7790e-03],
          [-3.4785e-02, -3.5812e-02, -8.8125e-03, -8.0203e-02,  4.5629e-02],
          [-7.4658e-03,  3.9296e-02, -7.6775e-03, -1.1554e-02, -5.8469e-02],
          [ 5.8886e-03, -8.5715e-03, -9.7266e-03, -4.5659e-02, -8.3813e-02],
          [-4.5191e-02, -9.2815e-04, -2.5495e-02, -3.3778e-02, -1.2397e-01]],

         [[ 4.9243e-03,  1.3177e-02,  8.0350e-03,  3.3287e-02, -3.3640e-02],
          [ 9.8279e-03, -3.0743e-02,  2.1493e-02, -3.7792e-02,  7.5454e-03],
          [-2.1124e-02,  1.0637e-03, -1.3177e-02, -1.8751e-02,  3.0467e-02],
          [-1.8285e-02, -4.7300e-02,  2.5524e-02,  6.7375e-03, -2.8934e-02],
          [-9.0098e-03, -2.6060e-02,  8.9541e-03, -3.9241e-02, -1.3051e-02]],

         [[-3.9521e-02, -4.5517e-02, -2.4250e-02, -3.3195e-02, -2.8776e-03],
          [-8.5059e-03,  5.7047e-03, -3.4138e-02, -1.9737e-02, -3.6967e-02],
          [-3.9790e-02, -7.5547e-02, -4.5481e-02, -4.4113e-02,  1.2985e-03],
          [-2.8851e-02, -4.4649e-02, -9.1624e-04, -4.0129e-02, -8.4302e-03],
          [-7.1234e-02, -3.8091e-02, -3.8379e-03, -1.8952e-02,  9.3426e-03]]],


        [[[-7.5333e-02,  1.0624e-01,  2.0520e-01, -7.1702e-02,  6.2639e-03],
          [ 6.9636e-02,  1.9810e-02,  9.8949e-02,  3.3770e-02,  8.9203e-02],
          [ 8.8296e-02,  9.6988e-02,  1.6659e-01,  9.8381e-02,  1.0705e-01],
          [ 4.5391e-02, -1.9230e-01,  1.3884e-01,  2.3666e-01, -1.9242e-01],
          [-2.1606e-01, -2.7065e-01, -1.8232e-01, -2.3494e-02, -1.4904e-01]],

         [[ 1.5584e-01, -2.5475e-01,  1.4592e-01,  1.6475e-01,  1.4026e-01],
          [ 6.1368e-02, -3.8784e-01, -5.0123e-02,  3.5378e-01,  2.4204e-01],
          [ 4.3356e-01, -1.8594e-01,  4.3338e-02, -2.0989e-01, -1.1766e-01],
          [ 2.6488e-01, -1.7041e-01, -2.9293e-01, -5.5544e-02,  2.0480e-02],
          [-8.6463e-02,  1.1210e-02,  2.8279e-01,  1.4811e-01, -3.6127e-01]],

         [[-1.5821e-01,  3.5971e-01,  1.6651e-02, -2.6556e-01, -2.2868e-01],
          [ 1.1667e-01, -1.8016e-01, -3.6973e-01, -2.8768e-01, -4.1026e-01],
          [-9.4482e-02, -1.3149e-01, -2.3248e-01,  1.8619e-01, -7.4635e-02],
```

```
  [ 1.5647e-02, -1.6495e-01, -1.5263e-01, -5.8590e-02, -1.4561e-01],
  [-6.9911e-02, -1.4848e-01, -3.4837e-01, -2.4843e-01, -1.1989e-01]],

 ...,

 [[ 2.8126e-02, -1.8155e-01,  1.1944e-01,  2.1409e-01,  3.6817e-02],
  [-1.0005e-01, -2.1441e-01, -7.1561e-02,  2.7700e-01,  1.5890e-02],
  [ 1.9176e-01,  5.9414e-02, -6.3876e-02, -3.0889e-02, -9.5076e-02],
  [-1.0204e-01,  2.3630e-01,  6.2946e-02, -3.3965e-02, -7.2961e-03],
  [-4.0101e-01,  1.9580e-01,  1.0246e-01, -2.6892e-01, -4.5591e-01]],

 [[-8.7029e-02,  2.6225e-01,  1.3254e-01,  6.9837e-02, -3.6107e-01],
  [-1.8279e-01, -3.1940e-01, -2.3140e-02,  2.0742e-01, -8.7358e-02],
  [-2.8531e-02, -6.1422e-02, -5.1328e-02,  6.8450e-02,  5.7530e-02],
  [ 4.1795e-02, -4.1061e-02, -1.2197e-01,  1.2622e-01,  3.4968e-01],
  [ 6.7586e-02, -2.5932e-01, -2.6715e-01, -1.2379e-01,  7.3879e-02]],

 [[ 1.5713e-01,  2.4623e-01, -6.2247e-02, -3.9418e-01,  1.0314e-01],
  [ 3.2373e-01,  4.2236e-01,  6.7484e-02, -1.2527e-01,  1.2406e-01],
  [ 1.5735e-01,  1.1317e-01,  1.1834e-01,  6.3888e-02, -8.2424e-02],
  [-1.4180e-01,  1.0495e-01, -2.0776e-02,  1.3580e-03,  9.1675e-03],
  [-2.5782e-01, -1.7304e-01,  4.0624e-02,  1.3358e-01, -6.4618e-02]]],


[[[-5.9274e-02, -4.2069e-02, -9.2240e-02, -2.9190e-02, -8.0862e-02],
  [-8.5045e-02, -9.9287e-02, -4.8486e-02,  5.1093e-03, -1.0912e-01],
  [-9.1153e-02,  3.3833e-03, -5.6614e-02, -3.3545e-02, -8.1893e-02],
  [-6.2667e-02, -4.5669e-02, -9.2617e-02, -7.2396e-02, -7.0893e-02],
  [-8.2521e-02, -7.7952e-02, -5.3715e-02, -1.5715e-02, -6.8024e-02]],

 [[-3.6430e-02,  1.9542e-02, -2.6256e-02,  2.2361e-02, -1.3561e-02],
  [-2.2264e-02, -2.6094e-02, -1.0375e-02, -3.8835e-02, -9.3664e-03],
  [-4.8542e-03, -4.4865e-02, -2.1247e-02,  2.2912e-03,  1.8919e-02],
  [-7.5301e-03, -7.6488e-03,  2.7383e-03, -7.1008e-03, -7.4580e-03],
  [ 2.0862e-03, -4.7750e-02, -1.5575e-02,  1.3139e-02,  4.3528e-03]],

 [[-6.5782e-02, -3.9375e-02,  2.2420e-02, -4.7215e-02, -4.2195e-02],
  [-4.5685e-02, -1.5268e-02,  1.3699e-02, -4.5740e-03, -2.2978e-02],
  [-5.2247e-02,  4.5178e-03, -2.7612e-02, -4.9901e-02, -7.6546e-02],
  [-4.6252e-02, -4.1006e-02, -1.2991e-02, -1.6819e-02, -9.6892e-02],
  [-8.8937e-02, -1.9113e-02,  4.1270e-03, -1.1055e-03,  2.2166e-03]],

 ...,

 [[-4.0507e-02,  4.2506e-02, -1.3459e-02, -2.0234e-02, -4.6733e-02],
  [ 2.7049e-02,  2.4070e-03, -1.7630e-02,  2.5078e-02,  1.5973e-02],
  [ 1.5149e-02, -2.5283e-02, -4.1340e-02, -1.2742e-02,  1.5126e-02],
  [ 4.2067e-02,  1.3830e-02, -6.0733e-03, -4.3066e-02, -3.7013e-02],
  [ 1.3543e-02, -4.5820e-02, -1.9219e-02,  2.9956e-02, -1.8424e-02]],

 [[ 2.7344e-04, -5.5967e-03, -4.3073e-02,  4.3854e-03, -4.4227e-02],
  [-3.0725e-02,  1.2873e-02, -4.6018e-02,  2.0213e-03, -8.5541e-03],
  [-4.3099e-03,  5.6385e-03, -1.0667e-03, -1.2588e-02, -1.9392e-02],
  [-1.8370e-02,  2.1326e-03, -1.6606e-02, -2.1964e-02, -4.5459e-02],
  [-2.1641e-02, -6.2107e-03, -3.1716e-02, -3.4388e-02, -5.1801e-03]],

 [[ 2.8842e-02, -4.5672e-02, -3.0856e-02, -2.7566e-02,  8.0109e-04],
  [ 1.2998e-02, -2.6420e-03,  1.2836e-02,  6.2356e-03,  1.8225e-02],
  [-3.0681e-02, -8.8020e-03, -2.2261e-02, -4.0659e-02, -1.6354e-02],
  [-1.4333e-02, -3.7614e-02, -2.5364e-02, -2.5160e-02, -4.6373e-02],
  [ 3.4415e-03, -4.1521e-02, -2.6396e-02, -8.0345e-03,  1.6625e-02]]]],
```

```
       device='cuda:0', requires_grad=True), Parameter containing:
tensor([ 0.2077, -0.1010, -0.0957,  0.0153, -0.0829, -0.0295, -0.2197, -0.3433,
        -0.0376, -0.0092,  0.0161, -0.0299, -0.0674,  0.2414, -0.0476, -0.0771,
         0.0391,  0.3258, -0.1325,  0.0230, -0.0094, -0.0347, -0.0372, -0.0369,
        -0.2097, -0.0222, -0.0462, -0.1497,  0.0431, -0.0683, -0.0072, -0.0283,
         0.0140, -0.1739,  0.2142,  0.0836,  0.3786,  0.1640, -0.0861, -0.2308,
         0.3578,  0.3230,  0.2261,  0.0122, -0.0224,  0.0569, -0.0399, -0.0361,
        -0.0853,  0.0634, -0.2581, -0.0078, -0.0232, -0.1766, -0.1162, -0.1569,
        -0.0262,  0.0289, -0.0530, -0.0351, -0.1423, -0.1301, -0.1894, -0.0270],
       device='cuda:0', requires_grad=True), Parameter containing:
tensor([[-0.0510, -0.0431, -0.0279,  ...,  0.0349, -0.0025, -0.0131],
        [-0.0836,  0.0592,  0.0185,  ...,  0.0135, -0.0203,  0.0236],
        [-0.1256, -0.2342, -0.1042,  ..., -0.0103, -0.0210, -0.0520],
        ...,
        [-0.1350, -0.0743, -0.0050,  ...,  0.0314, -0.0028,  0.0346],
        [ 0.0620,  0.0144,  0.0339,  ...,  0.0263, -0.0149, -0.0518],
        [ 0.3897,  0.2707,  0.0226,  ...,  0.0034, -0.0291, -0.0029]],
       device='cuda:0', requires_grad=True), Parameter containing:
tensor([-0.0434, -0.1569,  0.1693, -0.1176, -0.0311, -0.1210,  0.1520,  0.0822,
         0.1479, -0.0026, -0.0690,  0.1773, -0.0552,  0.1126, -0.0878, -0.0917,
        -0.0834, -0.0599, -0.1623, -0.1416,  0.0164,  0.2482,  0.1187, -0.1132,
        -0.1269, -0.1048, -0.0246, -0.1789,  0.1613, -0.1403,  0.2459,  0.1385,
        -0.0671, -0.0146, -0.0752,  0.1596, -0.0718,  0.2219, -0.0330,  0.0583,
        -0.0024, -0.0847, -0.2571,  0.3670, -0.0370,  0.2905, -0.0818, -0.0860,
         0.4134, -0.1015,  0.2893, -0.0539,  0.1597,  0.2028,  0.0662, -0.1439,
         0.2913,  0.2051, -0.0663, -0.0473, -0.0734, -0.0556,  0.3000,  0.0133,
         0.0541, -0.0839,  0.1747,  0.1243,  0.1596,  0.1161,  0.0069, -0.0825,
        -0.0544, -0.1102,  0.0925,  0.0978, -0.0263,  0.0568,  0.0822,  0.0159,
        -0.0232,  0.2191, -0.0994, -0.1522,  0.0822, -0.0681,  0.0609, -0.0146,
         0.0046, -0.0220, -0.0123, -0.0275, -0.0388,  0.0192,  0.0487, -0.0568,
         0.2123,  0.1319, -0.0820,  0.3200], device='cuda:0',
       requires_grad=True), Parameter containing:
tensor([[ 5.7143e-02, -7.2030e-02,  8.8622e-02,  3.1997e-02, -4.0704e-01,
          4.7902e-02,  1.0172e-02, -4.9895e-02, -6.1448e-02, -7.8599e-02,
         -9.2518e-02, -1.5926e-01,  1.3352e-01, -2.9315e-01, -4.1273e-02,
         -4.9303e-02,  3.1285e-02,  7.5511e-02, -1.3703e-01,  8.1462e-02,
         -4.4709e-02, -1.7449e-01, -3.3524e-02, -6.1972e-02, -6.6814e-02,
         -1.1345e-01, -3.9574e-03, -2.0258e-01,  1.0621e-01, -3.4665e-01,
          3.4836e-02, -2.4402e-01, -2.2840e-03,  3.9086e-02, -4.3014e-02,
         -1.5132e-01,  5.5926e-02,  9.5938e-03, -2.5057e-02, -1.5656e-01,
          2.0158e-02,  2.7693e-03, -3.8518e-01, -3.0612e-02,  7.4980e-02,
          1.2062e-01,  2.2570e-02, -1.7398e-01, -1.7178e-01, -8.3294e-02,
         -1.9459e-01, -8.5325e-02, -3.2370e-01,  1.9864e-02, -1.2758e-01,
          5.2297e-02,  9.7021e-02,  1.2413e-01, -3.0800e-01, -2.8940e-02,
          6.2251e-02, -5.1505e-03, -2.5056e-01,  1.4846e-01,  7.3634e-02,
         -4.4944e-02, -1.0134e-01,  1.9772e-01, -1.2482e-01, -1.8379e-01,
         -4.1944e-01, -2.1851e-02,  8.7019e-02, -5.9117e-02,  1.2505e-02,
          1.3837e-01, -2.6912e-01,  2.0489e-01,  3.3174e-01,  3.4126e-02,
          8.2847e-02, -7.5555e-04, -1.9343e-01, -1.4624e-01, -4.1031e-01,
          3.2973e-02, -8.7356e-03,  6.3280e-02,  1.6068e-01,  2.9520e-01,
         -6.0218e-02, -1.1856e-02, -8.5591e-02,  2.4471e-02, -2.0700e-01,
         -1.4965e-02,  6.8423e-02,  1.8094e-01,  5.6778e-02,  1.2412e-01],
        [-7.2570e-02, -2.0865e-02, -1.4066e-01, -8.6714e-02,  1.4967e-03,
          6.1470e-03,  1.7686e-02, -1.1965e-01,  1.2437e-01,  3.7301e-02,
         -2.0872e-01, -1.8511e-01, -3.5332e-01,  1.4415e-01, -1.4115e-01,
         -5.9505e-03,  8.4694e-02,  2.3765e-02, -1.5717e-01, -1.8204e-01,
         -9.3463e-02, -2.9597e-02, -5.8349e-01, -1.0439e-01, -1.8406e-01,
         -2.9875e-02,  1.8048e-01, -7.5898e-02,  1.6116e-01,  1.9240e-01,
         -3.8941e-02, -6.5209e-01, -5.7420e-02,  3.7780e-02,  1.4031e-01,
         -3.3522e-01, -2.1944e-01, -4.2868e-01, -1.1347e-01,  2.0530e-01,
         -2.1347e-02, -4.6713e-02,  2.8869e-01,  1.7408e-04, -6.5268e-02,
```

```
         -3.1297e-01,  7.3462e-03,  1.3428e-01,  2.6492e-01, -4.0168e-02,
         -2.5467e-01, -1.2298e-01,  1.5769e-01,  1.1442e-01, -5.0230e-02,
         -9.3365e-03, -2.0795e-01, -5.0598e-01,  2.3288e-01,  2.7794e-02,
         -8.9882e-02,  1.7291e-02, -3.0502e-01,  2.4540e-01,  1.8880e-01,
         -6.9861e-02, -6.2916e-02,  1.5937e-01,  7.3682e-03,  1.0111e-01,
          2.7039e-03, -1.2298e-01,  2.7405e-02,  1.1674e-01, -1.5104e-01,
         -2.9063e-01, -1.8056e-02,  2.0593e-02, -7.5353e-02,  2.4181e-02,
         -7.4571e-02, -5.4458e-02, -1.4196e-01, -2.3709e-01, -3.6914e-01,
         -6.6947e-02, -1.1561e-01, -2.8797e-02, -5.0376e-01, -2.0768e-02,
         -4.4275e-02, -1.2812e-01,  1.8809e-02,  4.1548e-02, -4.7197e-02,
         -3.2217e-01, -2.9937e-01, -2.4766e-01, -3.1536e-04,  9.4620e-02],
        [-1.6154e-02, -1.5976e-01, -2.3061e-01, -5.8076e-02, -5.8995e-01,
         -9.1462e-03, -1.7004e-01, -1.5008e-01, -7.5133e-02,  2.0347e-02,
         -1.5954e-02, -1.1753e-01,  6.9019e-02, -2.6420e-02,  3.8631e-02,
         -2.0261e-03,  7.6971e-02, -1.6653e-02, -1.7359e-01,  1.0952e-01,
         -4.9405e-02, -1.7696e-01, -9.7381e-02, -1.9850e-03, -2.1737e-03,
         -1.3733e-01, -5.9512e-02, -1.7873e-01,  1.5378e-02,  1.0615e-01,
          2.8536e-02,  1.6955e-01, -1.3361e-01,  6.9099e-02, -2.8074e-01,
         -2.3195e-01, -4.6956e-02, -1.4597e-01, -3.2729e-02, -4.6719e-01,
         -7.7164e-02, -5.1436e-02, -3.7305e-02,  8.0317e-02, -3.5088e-02,
          2.9955e-02,  2.8876e-02, -1.1606e-01, -2.0935e-01, -1.8755e-03,
          8.3269e-02, -1.6259e-01, -3.1781e-01,  1.4456e-03, -2.3040e-02,
         -8.4861e-02, -7.8231e-02,  1.1831e-01, -2.7594e-01,  3.0188e-02,
         -5.4017e-02, -3.4098e-02, -1.2299e-01, -1.6231e-02, -4.4202e-02,
          8.9359e-02, -3.2076e-01, -3.2527e-01, -2.2253e-02, -1.3062e-01,
         -1.4716e-02, -1.0342e-01, -3.9025e-03, -1.5241e-01,  9.7526e-02,
         -2.1783e-01,  1.1419e-01, -1.8459e-01,  6.4831e-02, -6.0154e-02,
         -2.9728e-01,  1.0473e-01, -3.4877e-02, -6.1223e-02, -1.6071e-01,
         -7.0899e-02,  1.7586e-01, -3.9623e-02,  1.3092e-01,  1.6404e-01,
         -5.8658e-02, -5.6699e-02,  4.8576e-02, -5.3683e-02,  2.4590e-01,
          1.0391e-01, -3.6717e-01, -1.9069e-01, -9.9590e-02,  1.3035e-01],
        [ 6.8935e-02, -4.1869e-02,  9.5382e-02, -8.5252e-02, -3.7089e-02,
         -4.9872e-02, -3.7473e-01, -2.0778e-01,  1.4093e-01,  4.4595e-02,
         -9.9574e-02,  6.0161e-03, -3.4371e-02,  9.2904e-02,  1.2254e-02,
         -4.3458e-02, -3.3625e-02,  5.1440e-02, -2.9813e-01, -1.5883e-01,
          4.3563e-02, -3.7897e-02, -1.0838e-01,  7.5430e-02,  2.6958e-02,
          7.7189e-03, -1.0819e-01, -2.5969e-01, -1.8141e-01, -3.7578e-01,
         -3.1725e-01, -5.1314e-01, -8.7991e-03, -5.8794e-02,  4.7945e-02,
         -3.1945e-01, -2.0263e-01,  1.7021e-01, -1.3933e-01,  9.1874e-03,
          5.3836e-02,  8.8866e-03, -3.7487e-02,  1.1843e-01, -6.2467e-03,
          1.2324e-01,  1.9162e-02,  1.4854e-01,  1.5797e-01, -1.0675e-01,
          1.8639e-01,  8.5715e-02,  1.5476e-01,  1.5806e-01, -1.5213e-01,
          1.7117e-02, -2.5028e-01, -4.5163e-03, -2.3287e-02, -2.4540e-02,
          6.7674e-02, -5.2687e-03,  2.6872e-02, -1.1243e-01, -3.6477e-01,
          9.5760e-02, -4.0678e-03,  6.2970e-02,  1.8774e-01,  6.2600e-03,
         -2.8407e-01, -1.0810e-01, -1.3798e-01,  8.8193e-02,  6.6271e-02,
          4.7893e-02, -6.9801e-02, -4.1162e-01,  3.7270e-02, -5.4092e-02,
          4.0297e-02, -2.1158e-01, -7.6316e-03,  4.4268e-02, -3.0565e-01,
          3.4771e-02,  9.6039e-02,  2.1346e-03, -3.4767e-01, -1.6679e-01,
          3.8715e-02, -3.6733e-02,  6.0154e-02, -2.8750e-03,  7.8096e-03,
         -2.9617e-02, -1.0556e-01,  5.2749e-02, -1.3708e-01,  1.1060e-01],
        [ 7.7009e-02, -1.0964e-01,  1.2133e-01, -1.1001e-03, -2.4992e-01,
          6.3419e-02,  6.7664e-02,  5.9200e-02,  1.8723e-02, -2.6566e-02,
          5.2246e-02,  4.1239e-02,  7.7050e-02, -5.1848e-02, -1.4012e-02,
          9.5218e-02,  7.5507e-02,  3.1701e-02,  4.9126e-02,  1.2077e-01,
          8.9052e-02, -2.9616e-01,  1.1492e-01,  4.6793e-02, -9.3456e-04,
          9.2522e-03,  1.3495e-01, -3.0421e-01, -2.5229e-01,  1.8362e-01,
         -4.0410e-02,  8.6818e-02,  1.0523e-01, -5.8345e-03, -6.2229e-02,
         -1.4897e-01,  1.2596e-01, -1.1890e-01, -6.1924e-02,  9.3281e-02,
         -1.6307e-02,  3.7358e-02, -1.9237e-01, -1.3154e-01,  4.1396e-04,
          3.2547e-02, -1.3682e-02, -2.1394e-01, -2.0198e-01, -6.1460e-02,
```

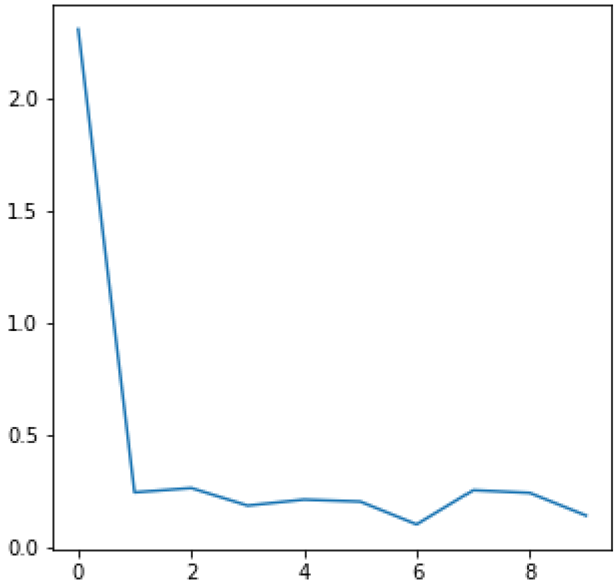
```
 -9.9782e-02,  2.6803e-01,  1.3856e-02,  7.1092e-02,  1.3694e-01,
 -1.7674e-01, -3.2628e-01,  1.3844e-02, -2.3613e-01, -3.4452e-02,
  4.0775e-03, -3.8700e-02, -6.2957e-02, -2.3775e-01, -4.7590e-02,
  1.4608e-01, -5.6528e-02, -1.1938e-01,  4.9944e-02,  4.0149e-02,
 -1.9414e-01, -4.4864e-02,  5.8044e-02, -8.1371e-02,  5.6516e-02,
 -2.4408e-02,  1.5726e-01, -3.3731e-01, -5.2610e-02, -8.9620e-03,
 -1.2260e-01, -3.0160e-01, -1.6411e-02,  1.5417e-01, -4.4777e-01,
 -7.0025e-02, -1.3453e-01,  1.5855e-02, -2.2769e-01,  9.3089e-02,
 -7.1732e-02, -1.3294e-01, -4.7488e-02,  5.9300e-03,  5.2100e-02,
  1.2311e-01, -4.3249e-01, -2.8712e-01,  4.0042e-02, -1.7258e-01],
[ 1.0461e-01, -1.2779e-01, -2.1079e-01,  4.3242e-02,  1.7667e-01,
 -1.9940e-02, -2.9817e-01,  1.4802e-01,  4.9128e-02,  2.9967e-02,
 -7.4707e-02,  3.1769e-02, -1.4575e-01,  7.3253e-02,  3.7885e-02,
 -1.3839e-01, -1.1171e-01, -7.6558e-02,  1.8090e-01, -1.5605e-02,
 -4.1355e-02, -2.6591e-01, -3.7813e-01,  1.9269e-02, -2.4946e-02,
 -3.2216e-02, -1.7643e-01, -7.0274e-02, -1.9051e-01, -3.0773e-01,
  1.2097e-01, -6.2928e-02, -4.6948e-02,  7.1305e-02, -2.1708e-02,
 -2.6747e-01, -3.0437e-01, -1.3789e-01,  4.7101e-02,  2.7628e-03,
  7.3493e-02, -2.9699e-02, -8.3589e-02, -4.8751e-01, -2.6028e-02,
 -1.4806e-01,  5.7272e-03, -8.8313e-02, -8.7166e-02,  7.1343e-02,
 -1.5853e-01, -5.1084e-01, -4.6908e-02, -3.5644e-01,  1.6460e-01,
  4.0795e-03,  1.4769e-01, -3.5408e-01,  3.8178e-02,  4.5123e-02,
  8.5766e-02,  7.0368e-02, -2.3643e-02,  8.2688e-02, -5.5988e-02,
 -5.5290e-03,  1.4079e-01, -4.8820e-01,  1.5869e-02, -3.9587e-01,
  1.5672e-01, -1.0915e-01,  4.5929e-03, -3.2850e-02,  1.8805e-01,
 -1.2590e-02, -2.7600e-01,  1.2546e-01,  2.6376e-02, -4.5058e-02,
  1.1030e-01,  8.9342e-02,  5.0247e-02, -3.9883e-01, -2.8011e-01,
 -3.7486e-02, -2.1217e-01,  2.2782e-02, -1.0148e-01, -1.4433e-01,
 -6.2411e-02,  1.0618e-02,  2.6477e-02,  7.2574e-02, -1.7782e-01,
 -5.1406e-01,  9.7159e-02,  1.0992e-01,  7.5107e-02, -5.6852e-02],
[ 1.6213e-02, -5.1974e-02, -4.6903e-02,  8.9303e-03, -5.3495e-01,
 -5.6254e-02,  5.1569e-02,  3.2472e-02, -3.0767e-01,  5.2848e-02,
  2.2100e-02,  6.1641e-02,  1.1257e-01, -1.4062e-01, -1.1553e-01,
 -1.2431e-03,  5.8413e-02,  5.0580e-02, -1.9555e-01,  1.3988e-01,
 -1.0401e-02, -2.3116e-01,  2.0878e-01,  3.7142e-02,  1.2302e-02,
 -3.2927e-02, -1.0980e-01, -6.8169e-02,  8.2388e-02, -6.6873e-02,
 -1.3023e-02, -1.5904e-02,  6.6576e-02, -8.2049e-02, -3.0061e-02,
  3.3345e-01, -3.0191e-02,  1.9575e-01, -4.3798e-02, -4.1373e-02,
 -5.7599e-02, -3.9909e-02, -3.6032e-01,  1.7903e-01, -6.0865e-02,
  1.0425e-01, -3.8132e-02, -1.8084e-01, -1.4638e-01, -1.8462e-02,
  2.6445e-02, -7.0232e-02,  4.3113e-02, -9.2579e-02, -3.3775e-01,
 -7.9722e-02, -2.4025e-01,  7.2110e-02, -1.5758e-01, -3.5371e-02,
 -6.1895e-03, -6.0668e-02,  6.2554e-02,  7.5052e-02,  1.1470e-01,
  1.0109e-01, -1.2606e-01,  1.0182e-01, -2.2127e-01,  1.1818e-01,
  1.0936e-01,  2.8806e-02,  4.2809e-02, -3.2005e-02, -1.1558e-01,
  2.0856e-01,  1.0732e-01, -2.2401e-01, -2.9765e-01,  2.8030e-02,
 -2.9141e-01, -5.0561e-02, -7.7171e-02, -9.8516e-02, -1.6640e-01,
 -7.2373e-02,  3.3825e-02,  4.7471e-02,  1.0320e-01, -7.9059e-02,
 -1.7777e-02, -9.1796e-02, -1.7390e-02,  5.5435e-02, -2.7133e-01,
  1.0576e-01,  2.2293e-03,  1.3967e-01,  4.2847e-02, -6.0563e-02],
[ 2.8386e-02, -2.2646e-04,  4.9148e-02,  4.5281e-02, -9.0058e-03,
  4.5009e-02, -7.7937e-03,  1.0021e-01, -3.0434e-02,  3.0370e-02,
 -1.9662e-02,  1.0231e-01, -3.3128e-01, -2.2114e-02, -5.2500e-02,
 -1.0097e-02,  6.7616e-02,  3.2551e-02,  6.4782e-02, -3.4484e-01,
  7.1244e-02,  8.3202e-02, -4.1012e-01, -6.0132e-02, -4.5466e-02,
 -1.2872e-02, -5.0247e-01, -2.7441e-01, -2.7043e-01, -2.1576e-01,
 -4.3982e-02, -6.4784e-02,  2.4352e-02, -8.5806e-03, -3.7381e-02,
 -3.0846e-01,  2.6477e-02,  1.6376e-01, -6.2468e-02,  1.5550e-01,
 -4.1710e-02, -1.5400e-01, -1.5515e-01, -5.0971e-01, -6.5587e-02,
 -3.3040e-02, -4.6413e-02, -1.8468e-01, -2.0426e-02,  2.5110e-02,
  8.3415e-02, -3.1227e-02,  9.7265e-02, -6.3229e-01,  1.6178e-01,
```

```
         -1.8589e-02,  5.5860e-02, -4.9002e-01,  1.4453e-01, -1.3626e-02,
         -3.6009e-02, -1.0135e-02, -3.7486e-02,  5.9745e-02, -1.2244e-01,
         -2.5710e-01, -1.8989e-01, -2.5543e-01, -2.3935e-01, -3.4274e-01,
          1.2221e-01, -7.2019e-02,  8.4410e-02, -1.2421e-01,  4.4303e-02,
          1.3863e-01, -1.7099e-01,  6.6651e-02, -1.5034e-01, -4.5826e-02,
          8.2874e-03, -4.8305e-02,  4.6696e-02, -8.9668e-02,  2.0697e-01,
         -2.5889e-02,  4.0856e-02,  2.8036e-02,  2.8838e-02, -6.5187e-02,
          5.2311e-02, -6.5691e-02, -1.0912e-02, -4.5842e-02,  1.4983e-01,
         -6.2439e-01,  1.0351e-01, -2.5176e-01, -6.1014e-02, -3.6418e-01],
        [-3.3443e-02, -9.3358e-02, -1.0660e-01,  2.7709e-02, -3.2675e-01,
         -1.7543e-01,  1.8788e-01,  1.3256e-01,  1.1089e-01, -9.1650e-02,
          1.0976e-01,  1.5594e-01, -2.6152e-01, -2.1029e-01, -1.3801e-01,
          5.5719e-02, -2.6836e-02,  2.0725e-02, -3.2895e-01, -1.8136e-01,
          6.2851e-02,  3.6506e-01, -1.4330e-01,  5.5896e-02, -1.3835e-01,
         -1.2972e-01,  9.8813e-02,  2.0012e-02, -7.1377e-02, -5.1748e-01,
          8.2732e-02, -2.0449e-02, -1.9212e-01, -6.2213e-02, -7.5005e-02,
         -1.7428e-01,  5.8944e-02, -7.4237e-02,  3.8578e-02, -7.5024e-02,
          5.7765e-02,  4.0888e-02,  2.1087e-01, -1.9219e-02, -2.4918e-02,
          1.0693e-01, -1.0380e-01,  5.8615e-02,  3.4055e-01, -2.5844e-02,
         -6.6770e-03, -1.1889e-01, -2.1694e-01, -4.4506e-01, -1.2106e-01,
         -2.5978e-01, -7.1178e-02,  9.7565e-02, -1.6723e-01,  7.1275e-02,
         -4.0860e-02,  1.5720e-03,  8.6988e-02,  1.0441e-02, -1.3354e-01,
         -1.8489e-01,  2.6685e-01, -1.1928e-01,  1.2518e-01, -3.2059e-01,
         -2.1486e-01, -1.3389e-01,  1.0739e-01, -3.9560e-02,  5.6417e-02,
         -3.1851e-01,  1.1974e-01, -3.6936e-01, -3.8215e-01,  1.3287e-02,
         -1.4287e-01,  2.0580e-01,  5.5004e-02, -3.1064e-01, -3.3316e-01,
         -4.9100e-03, -3.2288e-01,  2.7639e-03, -2.4532e-01,  1.5233e-01,
          7.7781e-02, -9.8876e-02, -5.8799e-02,  6.5887e-02, -9.0236e-03,
         -4.2026e-01,  7.4739e-02, -3.6455e-02, -1.5884e-02, -1.4322e-02],
        [ 2.8414e-02,  4.8766e-03, -9.6379e-02, -1.0472e-02,  1.4807e-01,
          6.7678e-02,  5.0211e-02, -3.3602e-03, -2.1048e-01, -3.2398e-02,
         -5.9997e-02, -9.3172e-02,  2.1559e-02, -1.7208e-01, -9.8474e-02,
          3.9630e-02,  3.6889e-02,  5.2488e-02,  9.3479e-02, -1.8842e-01,
         -5.1980e-02,  2.3376e-02, -4.9483e-01, -1.2570e-02,  9.4144e-02,
         -4.7984e-02, -3.5634e-01,  3.0653e-01, -8.8813e-02,  5.3975e-02,
         -3.1818e-01, -5.2982e-02,  2.3235e-02,  7.2879e-02, -4.7779e-02,
         -4.6345e-01, -7.0943e-02, -6.5514e-02, -8.5919e-02, -2.6304e-01,
         -3.9569e-03, -1.8057e-02, -7.4955e-02, -7.5004e-01,  4.2716e-02,
         -1.5118e-01, -4.6264e-02, -4.3734e-02,  3.3104e-02, -5.9829e-03,
         -1.3210e-02, -5.2037e-01, -1.9274e-01, -3.7451e-01,  3.7011e-02,
         -6.5948e-02, -1.4790e-01, -3.8413e-01,  1.0566e-01, -2.4841e-02,
          4.3274e-02, -1.0796e-02,  1.4076e-01,  8.2543e-02,  6.0957e-02,
         -2.4245e-02,  7.9292e-02, -2.5647e-01,  2.3656e-02, -1.1065e-01,
          1.4926e-01, -1.8292e-01, -2.7795e-02,  5.3339e-02, -1.4386e-01,
          1.2099e-01,  2.5200e-02,  1.7773e-01, -5.0244e-01, -1.4746e-03,
         -1.0749e-01,  1.3150e-01, -3.2764e-03,  3.4438e-03,  1.4708e-01,
         -4.1229e-03,  1.5064e-01, -4.0130e-02,  2.5548e-02,  9.6411e-03,
         -6.2890e-02,  5.1170e-02,  7.5422e-02, -1.0938e-01,  1.3651e-01,
         -4.5955e-01,  1.4833e-01, -2.1815e-01,  7.9369e-02, -1.1339e-01]],
       device='cuda:0', requires_grad=True), Parameter containing:
tensor([ 0.0169, -0.4006,  0.1733,  0.1873, -0.1918, -0.0579,  0.1337,  0.0370,
         0.2219, -0.0316], device='cuda:0', requires_grad=True)]
```

3-7 에포크를 5 이상으로 설정하고 에포크별 손실(loss) 그래프를 그려보세요.

In [343]:

```
plt.plot(loss_arr)
plt.show()
```



In [ ]: