```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
import numpy as np
import matplotlib.pyplot as plt
import torch.optim as optim
```

```
train_set = torchvision.datasets.FashionMNIST(root='./data/FashionMNIST',
                                              train=True, #we want the data for training s
                                              download=True, # download data if its not sp
                                              transform=transforms.Compose([transforms.ToT
```

```
train_loader = torch.utils.data.DataLoader(train_set, batch_size=100)
```

```
len(train_set)
```

```
60000
```

```
train_set.train_labels
```

```
/usr/local/lib/python3.6/dist-packages/torchvision/datasets/mnist.py:43: UserWarning:
  warnings.warn("train_labels has been renamed targets")
tensor([9, 0, 0,  ..., 3, 0, 5])
```

```
train_set.train_labels.bincount()
```

```
/usr/local/lib/python3.6/dist-packages/torchvision/datasets/mnist.py:43: UserWarning:
  warnings.warn("train_labels has been renamed targets")
tensor([6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000, 6000])
```

```
sample = next(iter(train_set))
```

```
len(sample)
```

```
2
```

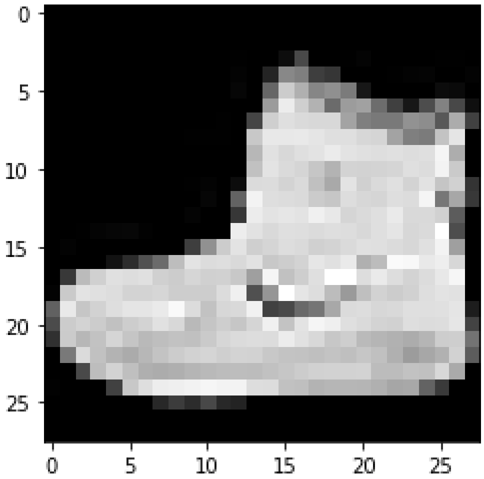
```
type(sample)
```

```
tuple
```

```
image, label = sample
```

```
plt.imshow(image.squeeze(), cmap='gray')
label
```

```
9
```



```
batch = next(iter(train_loader))
```

```
len(batch)
```

```
2
```

```
type(batch)
```

```
list
```

```
images, labels = batch
```

```
images.shape
```

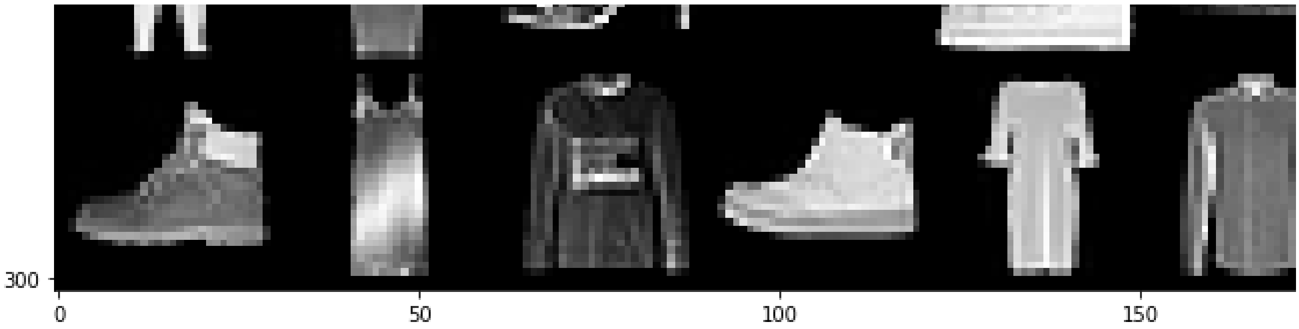
```
torch.Size([100, 1, 28, 28])
```

```
labels.shape
```

```
torch.Size([100])
```

```
grid = torchvision.utils.make_grid(images, nrow=10)
plt.figure(figsize=(20,20))
plt.imshow(np.transpose(grid, (1,2,0)))
labels
```

```
tensor([9, 0, 0, 3, 0, 2, 7, 2, 5, 5, 0, 9, 5, 5, 7, 9, 1, 0, 6, 4, 3, 1, 4, 8,
        4, 3, 0, 2, 4, 4, 5, 3, 6, 6, 0, 8, 5, 2, 1, 6, 6, 7, 9, 5, 9, 2, 7, 3,
        0, 3, 3, 3, 7, 2, 2, 6, 6, 8, 3, 3, 5, 0, 5, 5, 0, 2, 0, 0, 4, 1, 3, 1,
        6, 3, 1, 4, 4, 6, 1, 9, 1, 3, 5, 7, 9, 7, 1, 7, 9, 9, 9, 3, 2, 9, 3, 6,
        4, 1, 1, 8])
```



pytorch's nn library gives us the tools to construct layers of an NN. Each layer has tranformation ar

Here according to OOPS- a transformation will be a method(code) and weights will be attributes(d

The class Module is the base class for all NN modules. So, all the layers and NN(a bunch of layers

```
class Network(nn.Module):
  def __init__(self):
    super(Network, self).__init__()
    self.conv1 = nn.Conv2d(in_channels=1, out_channels=6, kernel_size=5)
    self.conv2 = nn.Conv2d(in_channels=6, out_channels=12, kernel_size=5)

    self.fc1 = nn.Linear(in_features=12*4*4, out_features=120)
    self.fc2 = nn.Linear(in_features=120, out_features=60)
    self.out = nn.Linear(in_features=60, out_features=10)

  def forward(self, t):
    # (1) Input layer
    t=t

    # (2) Hidden conv layer
    t=self.conv1(t)
    t=F.relu(t)
    t=F.max_pool2d(t, kernel_size=2, stride=2)

    # (3) Hidden conv layer
    t=self.conv2(t)
    t=F.relu(t)
    t=F.max_pool2d(t, kernel_size=2, stride=2)

    # (4) Hidden Linear layer
    t=t.reshape(-1, 12*4*4)
    t=self.fc1(t)
    t=F.relu(t)

    # (5) Hidden Linear layer
    t=self.fc2(t)
    t=F.relu(t)

    # (6) Output layer
    t=self.out(t)

    return t

network = Network()
network = Network()


for name, param in network.named_parameters():
  print(name, '\t\t', param.shape)
```

```
conv1.weight             torch.Size([6, 1, 5, 5])
conv1.bias               torch.Size([6])
conv2.weight             torch.Size([12, 6, 5, 5])
conv2.bias               torch.Size([12])
fc1.weight               torch.Size([120, 192])
fc1.bias                 torch.Size([120])
fc2.weight               torch.Size([60, 120])
fc2.bias                 torch.Size([60])
out.weight               torch.Size([10, 60])
out.bias                 torch.Size([10])
```

```
torch.set_grad_enabled(True)
```

```
<torch.autograd.grad_mode.set_grad_enabled at 0x7f23977249b0>
```

```
image.shape
```

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

```
image.unsqueeze(0).shape
```

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

```
pred = network(image.unsqueeze(0))
```

```
pred.shape
```

```
torch.Size([1, 10])
```

```
pred
```

```
tensor([[ 0.0600,  0.0306,  0.1291,  0.0470,  0.0291, -0.1243,  0.1202,  0.1796,
         -0.0860,  0.0468]], grad_fn=<AddmmBackward>)
```

```
pred.argmax(dim=1)
```

```
tensor([7])
```

```
F.softmax(pred, dim=1)
```

```
    tensor([[0.1013, 0.0984, 0.1086, 0.1000, 0.0982, 0.0843, 0.1076, 0.1142, 0.0875,
```

```
F.softmax(pred, dim=1).argmax(dim=1)
```

```
    tensor([7])
```

```
preds = network(images)
```

```
preds.shape
```

```
    torch.Size([100, 10])
```

```
preds # Batch size = 10 with 10 prediction classes
```

```
tensor([[ 0.0600,  0.0306,  0.1291,  0.0470,  0.0291, -0.1243,  0.1202,  0.1796,
         -0.0860,  0.0468],
        [ 0.0707,  0.0228,  0.1288,  0.0576,  0.0249, -0.1251,  0.1277,  0.1835,
         -0.0789,  0.0409],
        [ 0.0570,  0.0372,  0.1151,  0.0649,  0.0347, -0.1028,  0.1150,  0.1704,
         -0.0800,  0.0462],
        [ 0.0602,  0.0313,  0.1222,  0.0642,  0.0276, -0.1113,  0.1189,  0.1764,
         -0.0823,  0.0479],
        [ 0.0647,  0.0188,  0.1275,  0.0614,  0.0235, -0.1204,  0.1226,  0.1832,
         -0.0860,  0.0479],
        [ 0.0643,  0.0278,  0.1325,  0.0596,  0.0227, -0.1248,  0.1285,  0.1824,
         -0.0809,  0.0454],
        [ 0.0653,  0.0374,  0.1219,  0.0549,  0.0254, -0.1084,  0.1255,  0.1734,
         -0.0877,  0.0535],
        [ 0.0608,  0.0262,  0.1373,  0.0561,  0.0240, -0.1328,  0.1283,  0.1867,
         -0.0743,  0.0427],
        [ 0.0502,  0.0429,  0.1155,  0.0624,  0.0404, -0.1072,  0.1108,  0.1632,
         -0.0836,  0.0486],
        [ 0.0466,  0.0401,  0.1251,  0.0630,  0.0378, -0.1202,  0.1101,  0.1710,
         -0.0808,  0.0471],
        [ 0.0613,  0.0251,  0.1276,  0.0635,  0.0247, -0.1152,  0.1212,  0.1807,
         -0.0838,  0.0485],
        [ 0.0574,  0.0279,  0.1326,  0.0500,  0.0297, -0.1284,  0.1199,  0.1866,
         -0.0830,  0.0455],
        [ 0.0606,  0.0403,  0.1255,  0.0570,  0.0261, -0.1136,  0.1181,  0.1760,
         -0.0861,  0.0511],
        [ 0.0529,  0.0439,  0.1219,  0.0630,  0.0271, -0.1065,  0.1170,  0.1718,
         -0.0872,  0.0507],
        [ 0.0632,  0.0429,  0.1171,  0.0573,  0.0228, -0.1022,  0.1259,  0.1673,
         -0.0883,  0.0529],
        [ 0.0597,  0.0337,  0.1313,  0.0492,  0.0260, -0.1218,  0.1176,  0.1805,
         -0.0889,  0.0463],
        [ 0.0641,  0.0282,  0.1265,  0.0538,  0.0329, -0.1209,  0.1224,  0.1811,
         -0.0779,  0.0425],
        [ 0.0660,  0.0242,  0.1274,  0.0623,  0.0214, -0.1179,  0.1251,  0.1823,
         -0.0828,  0.0495],
        [ 0.0583,  0.0317,  0.1311,  0.0605,  0.0221, -0.1213,  0.1266,  0.1795,
         -0.0810,  0.0476],
        [ 0.0579,  0.0370,  0.1217,  0.0649,  0.0315, -0.1084,  0.1153,  0.1762,
         -0.0803,  0.0452],
        [ 0.0697,  0.0265,  0.1237,  0.0575,  0.0251, -0.1180,  0.1242,  0.1816,
         -0.0818,  0.0481],
        [ 0.0657,  0.0231,  0.1262,  0.0585,  0.0243, -0.1157,  0.1261,  0.1823,
         -0.0816,  0.0487],
        [ 0.0616,  0.0253,  0.1323,  0.0572,  0.0228, -0.1213,  0.1242,  0.1836,
         -0.0804,  0.0489],
        [ 0.0634,  0.0301,  0.1324,  0.0544,  0.0215, -0.1232,  0.1250,  0.1793,
         -0.0815,  0.0479],
        [ 0.0638,  0.0243,  0.1339,  0.0587,  0.0212, -0.1243,  0.1292,  0.1849,
         -0.0806,  0.0468],
        [ 0.0670,  0.0289,  0.1300,  0.0595,  0.0186, -0.1224,  0.1302,  0.1838,
         -0.0825,  0.0486],
        [ 0.0610,  0.0260,  0.1335,  0.0624,  0.0174, -0.1234,  0.1262,  0.1827,
         -0.0830,  0.0486],
        [ 0.0618,  0.0240,  0.1356,  0.0543,  0.0256, -0.1334,  0.1271,  0.1850,
         -0.0766,  0.0415],
        [ 0.0618,  0.0313,  0.1262,  0.0655,  0.0256, -0.1168,  0.1230,  0.1804,
         -0.0803,  0.0484],
        [ 0.0639,  0.0278,  0.1331,  0.0561,  0.0240, -0.1274,  0.1271,  0.1824,
         -0.0767,  0.0455],
        [ 0.0579,  0.0387,  0.1222,  0.0598,  0.0240, -0.1015,  0.1228,  0.1729,
         -0.0899,  0.0494],
        [ 0.0591,  0.0309,  0.1228,  0.0620,  0.0254, -0.1075,  0.1196,  0.1739,
         -0.0867,  0.0491],
        [ 0.0668,  0.0247,  0.1356,  0.0569,  0.0203, -0.1304,  0.1298,  0.1850,
         -0.0780,  0.0466],
        [ 0.0531,  0.0399,  0.1112,  0.0654,  0.0324, -0.0991,  0.1117,  0.1675,
         -0.0868,  0.0481],
        [ 0.0544,  0.0381,  0.1147,  0.0654,  0.0327, -0.1002,  0.1126,  0.1719,
         -0.0830,  0.0480],
        [ 0.0541,  0.0347,  0.1297,  0.0564,  0.0287, -0.1085,  0.1181,  0.1781,
         -0.0831,  0.0508],
        [ 0.0496,  0.0358,  0.1147,  0.0590,  0.0393, -0.1103,  0.1111,  0.1702,
         -0.0822,  0.0495],
        [ 0.0580,  0.0368,  0.1218,  0.0670,  0.0260, -0.1091,  0.1211,  0.1714,
         -0.0852,  0.0503],
        [ 0.0667,  0.0193,  0.1317,  0.0523,  0.0299, -0.1210,  0.1231,  0.1884,
         -0.0760,  0.0392],
        [ 0.0609,  0.0237,  0.1363,  0.0574,  0.0225, -0.1306,  0.1274,  0.1857,
         -0.0777,  0.0445],
        [ 0.0625,  0.0271,  0.1332,  0.0587,  0.0213, -0.1247,  0.1315,  0.1831,
         -0.0821,  0.0466],
        [ 0.0651,  0.0391,  0.1212,  0.0541,  0.0300, -0.1120,  0.1210,  0.1735,
         -0.0875,  0.0488],
        [ 0.0651,  0.0328,  0.1271,  0.0464,  0.0296, -0.1220,  0.1202,  0.1784,
         -0.0861,  0.0467],
        [ 0.0578,  0.0352,  0.1267,  0.0523,  0.0276, -0.1133,  0.1210,  0.1743,
         -0.0879,  0.0511],
        [ 0.0620,  0.0361,  0.1298,  0.0507,  0.0288, -0.1327,  0.1244,  0.1822,
         -0.0779,  0.0503],
        [ 0.0590,  0.0311,  0.1288,  0.0656,  0.0220, -0.1172,  0.1241,  0.1786,
         -0.0834,  0.0505],
        [ 0.0614,  0.0342,  0.1249,  0.0509,  0.0287, -0.1105,  0.1276,  0.1759,
         -0.0891,  0.0497],
        [ 0.0680,  0.0244,  0.1276,  0.0650,  0.0194, -0.1182,  0.1260,  0.1818,
         -0.0870,  0.0495],
        [ 0.0681,  0.0245,  0.1294,  0.0576,  0.0233, -0.1211,  0.1242,  0.1813,
         -0.0805,  0.0462],
        [ 0.0676,  0.0287,  0.1272,  0.0594,  0.0272, -0.1264,  0.1249,  0.1867,
         -0.0760,  0.0449],
        [ 0.0698,  0.0283,  0.1214,  0.0582,  0.0290, -0.1180,  0.1267,  0.1821,
         -0.0800,  0.0446],
        [ 0.0655,  0.0264,  0.1233,  0.0580,  0.0257, -0.1148,  0.1229,  0.1801,
         -0.0852,  0.0462],
        [ 0.0670,  0.0364,  0.1229,  0.0551,  0.0260, -0.1097,  0.1247,  0.1761,
         -0.0882,  0.0522],
        [ 0.0636,  0.0261,  0.1353,  0.0559,  0.0243, -0.1340,  0.1303,  0.1858,
         -0.0752,  0.0451],
        [ 0.0557,  0.0375,  0.1188,  0.0669,  0.0292, -0.1039,  0.1151,  0.1699,
         -0.0861,  0.0486],
        [ 0.0616,  0.0247,  0.1298,  0.0606,  0.0232, -0.1210,  0.1217,  0.1814,
         -0.0809,  0.0478],
        [ 0.0722,  0.0247,  0.1280,  0.0565,  0.0171, -0.1162,  0.1314,  0.1849,
         -0.0783,  0.0475],
        [ 0.0630,  0.0272,  0.1325,  0.0534,  0.0212, -0.1136,  0.1245,  0.1796,
         -0.0873,  0.0462],
        [ 0.0678,  0.0247,  0.1233,  0.0554,  0.0254, -0.1174,  0.1263,  0.1822,
         -0.0830,  0.0459],
        [ 0.0625,  0.0301,  0.1241,  0.0563,  0.0264, -0.1134,  0.1206,  0.1788,
         -0.0818,  0.0472],
        [ 0.0490,  0.0287,  0.1355,  0.0542,  0.0377, -0.1244,  0.1142,  0.1822,
         -0.0803,  0.0427],
        [ 0.0608,  0.0236,  0.1321,  0.0613,  0.0237, -0.1191,  0.1204,  0.1836,
        [ 0.0608,  0.0290,  0.1321,  0.0615,  0.0257, -0.1191,  0.1204,  0.1830,
         -0.0835,  0.0465],
        [ 0.0551,  0.0424,  0.1202,  0.0644,  0.0288, -0.1022,  0.1154,  0.1727,
         -0.0900,  0.0492],
        [ 0.0548,  0.0447,  0.1190,  0.0629,  0.0251, -0.1009,  0.1182,  0.1686,
         -0.0887,  0.0522],
        [ 0.0645,  0.0259,  0.1290,  0.0588,  0.0250, -0.1221,  0.1235,  0.1811,
         -0.0822,  0.0502],
        [ 0.0629,  0.0302,  0.1289,  0.0614,  0.0216, -0.1205,  0.1270,  0.1798,
         -0.0791,  0.0488],
        [ 0.0660,  0.0236,  0.1282,  0.0625,  0.0228, -0.1198,  0.1261,  0.1827,
         -0.0816,  0.0497],
        [ 0.0607,  0.0247,  0.1294,  0.0591,  0.0184, -0.1159,  0.1248,  0.1795,
         -0.0913,  0.0501],
        [ 0.0611,  0.0328,  0.1296,  0.0616,  0.0244, -0.1225,  0.1242,  0.1804,
         -0.0806,  0.0483],
        [ 0.0647,  0.0278,  0.1256,  0.0548,  0.0296, -0.1149,  0.1216,  0.1801,
         -0.0786,  0.0461],
        [ 0.0689,  0.0230,  0.1310,  0.0520,  0.0261, -0.1296,  0.1262,  0.1856,
         -0.0764,  0.0463],
        [ 0.0671,  0.0247,  0.1274,  0.0579,  0.0262, -0.1173,  0.1217,  0.1835,
         -0.0798,  0.0435],
        [ 0.0628,  0.0273,  0.1276,  0.0629,  0.0209, -0.1113,  0.1213,  0.1791,
         -0.0867,  0.0493],
        [ 0.0664,  0.0228,  0.1251,  0.0556,  0.0284, -0.1197,  0.1212,  0.1830,
         -0.0849,  0.0457],
        [ 0.0648,  0.0256,  0.1293,  0.0526,  0.0296, -0.1192,  0.1207,  0.1840,
         -0.0772,  0.0426],
        [ 0.0677,  0.0202,  0.1321,  0.0570,  0.0200, -0.1222,  0.1262,  0.1861,
         -0.0805,  0.0438],
        [ 0.0671,  0.0251,  0.1292,  0.0548,  0.0246, -0.1223,  0.1261,  0.1820,
         -0.0818,  0.0437],
        [ 0.0630,  0.0274,  0.1288,  0.0602,  0.0214, -0.1187,  0.1272,  0.1795,
         -0.0822,  0.0479],
        [ 0.0629,  0.0290,  0.1266,  0.0561,  0.0278, -0.1141,  0.1216,  0.1793,
         -0.0810,  0.0440],
        [ 0.0621,  0.0314,  0.1292,  0.0517,  0.0254, -0.1225,  0.1233,  0.1780,
         -0.0854,  0.0471],
        [ 0.0693,  0.0240,  0.1274,  0.0529,  0.0271, -0.1165,  0.1265,  0.1859,
         -0.0754,  0.0438],
        [ 0.0590,  0.0366,  0.1163,  0.0642,  0.0294, -0.1046,  0.1173,  0.1709,
         -0.0841,  0.0496],
        [ 0.0506,  0.0361,  0.1299,  0.0625,  0.0309, -0.1159,  0.1098,  0.1709,
         -0.0847,  0.0462],
        [ 0.0671,  0.0348,  0.1249,  0.0522,  0.0218, -0.1117,  0.1300,  0.1760,
         -0.0880,  0.0509],
        [ 0.0610,  0.0320,  0.1257,  0.0483,  0.0353, -0.1237,  0.1159,  0.1775,
         -0.0811,  0.0457],
        [ 0.0616,  0.0358,  0.1224,  0.0541,  0.0271, -0.1099,  0.1239,  0.1723,
         -0.0890,  0.0488],
        [ 0.0690,  0.0279,  0.1249,  0.0558,  0.0275, -0.1163,  0.1253,  0.1826,
         -0.0760,  0.0450],
        [ 0.0628,  0.0402,  0.1202,  0.0581,  0.0283, -0.1062,  0.1242,  0.1718,
         -0.0858,  0.0525],
        [ 0.0561,  0.0343,  0.1297,  0.0504,  0.0344, -0.1262,  0.1183,  0.1782,
         -0.0809,  0.0423],
        [ 0.0603,  0.0373,  0.1243,  0.0530,  0.0321, -0.1165,  0.1159,  0.1752,
         -0.0889,  0.0489],
        [ 0.0531,  0.0420,  0.1183,  0.0577,  0.0415, -0.1140,  0.1078,  0.1714,
         -0.0822,  0.0475],
        [ 0.0613,  0.0315,  0.1229,  0.0588,  0.0224, -0.1104,  0.1201,  0.1741,
         -0.0856,  0.0521],
        [ 0.0623,  0.0385,  0.1220,  0.0619,  0.0200, -0.1055,  0.1229,  0.1725,
         -0.0854,  0.0535],
        [ 0.0618,  0.0355,  0.1248,  0.0498,  0.0307, -0.1198,  0.1179,  0.1782,
         -0.0863,  0.0483],
        [ 0.0665,  0.0247,  0.1283,  0.0593,  0.0217, -0.1156,  0.1254,  0.1792,
         -0.0838,  0.0426],
        [ 0.0581,  0.0309,  0.1296,  0.0668,  0.0217, -0.1170,  0.1218,  0.1784,
         -0.0848,  0.0506],
        [ 0.0566,  0.0401,  0.1144,  0.0661,  0.0356, -0.1037,  0.1141,  0.1695,
         -0.0798,  0.0446],
        [ 0.0657,  0.0233,  0.1291,  0.0461,  0.0297, -0.1192,  0.1225,  0.1843,
         -0.0733,  0.0361],
        [ 0.0674,  0.0279,  0.1246,  0.0574,  0.0241, -0.1156,  0.1247,  0.1798,
         -0.0793,  0.0463],
```

```
preds.argmax(dim=1)
```

```
tensor([7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7, 7,
        7, 7, 7, 7])
```

```
labels
```

```
tensor([9, 0, 0, 3, 0, 2, 7, 2, 5, 5, 0, 9, 5, 5, 7, 9, 1, 0, 6, 4, 3, 1, 4, 8,
        4, 3, 0, 2, 4, 4, 5, 3, 6, 6, 0, 8, 5, 2, 1, 6, 6, 7, 9, 5, 9, 2, 7, 3,
        0, 3, 3, 3, 7, 2, 2, 6, 6, 8, 3, 3, 5, 0, 5, 5, 0, 2, 0, 0, 4, 1, 3, 1,
        6, 3, 1, 4, 4, 6, 1, 9, 1, 3, 5, 7, 9, 7, 1, 7, 9, 9, 9, 3, 2, 9, 3, 6,
        4, 1, 1, 8])
```

```
preds.argmax(dim=1).eq(labels).sum()
```

```
tensor(8)
```

```
def get_num_correct(preds, labels):
  return preds.argmax(dim=1).eq(labels).sum().item()
```

```
get_num_correct(preds, labels)
```

```
8
```

```
loss = F.cross_entropy(preds, labels)
loss.item()
```

```
2.30240535736084
```

```
print(network.conv1.weight.grad) # Because we havent back propagated yet. The gradients wi
```

```
None
```

```
loss.backward() # Calculating the gradients
```

```
network.conv1.weight.grad.shape
```