lab6

February 20, 2025

[1]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import Dataset, DataLoader
import numpy as np
import matplotlib.pyplot as plt
from sklearn.metrics import accuracy_score
from sklearn import metrics
import torchvision
from torchvision import transforms, datasets
import torch.nn.functional as F
```

[2]:
```
transform = transforms.Compose([transforms.ToTensor(), transforms.Normalize((0.
 ↪5,), (0.5,))])

batch_size = 4
train_set = torchvision.datasets.MNIST(root='./data/', train=True,␣
 ↪download=True, transform=transform)
train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True,␣
 ↪num_workers=2)

test_set = torchvision.datasets.MNIST(root='./data/', train=False,␣
 ↪download=True, transform=transform)
test_loader = DataLoader(test_set, batch_size=batch_size, shuffle=False,␣
 ↪num_workers=2)
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-
idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-images-
idx3-ubyte.gz to ./data/MNIST/raw/train-images-idx3-ubyte.gz

100%|                    | 9.91M/9.91M [00:50<00:00, 195kB/s]

Extracting ./data/MNIST/raw/train-images-idx3-ubyte.gz to ./data/MNIST/raw
```

```
Downloading http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-
idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/train-labels-
idx1-ubyte.gz to ./data/MNIST/raw/train-labels-idx1-ubyte.gz

100%|                    | 28.9k/28.9k [00:00<00:00, 52.1kB/s]

Extracting ./data/MNIST/raw/train-labels-idx1-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-
idx3-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-images-
idx3-ubyte.gz to ./data/MNIST/raw/t10k-images-idx3-ubyte.gz

100%|                    | 1.65M/1.65M [00:35<00:00, 45.9kB/s]

Extracting ./data/MNIST/raw/t10k-images-idx3-ubyte.gz to ./data/MNIST/raw

Downloading http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz
Failed to download (trying next):
HTTP Error 403: Forbidden

Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-
idx1-ubyte.gz
Downloading https://ossci-datasets.s3.amazonaws.com/mnist/t10k-labels-
idx1-ubyte.gz to ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz

100%|                    | 4.54k/4.54k [00:00<00:00, 37.8MB/s]

Extracting ./data/MNIST/raw/t10k-labels-idx1-ubyte.gz to ./data/MNIST/raw
```

[3]:
```
device = 'cuda' if torch.cuda.is_available() else 'cpu'

class CNNNetwork(nn.Module):
    def __init__(self):
        super(CNNNetwork, self).__init__()
        self.feature_extractor = nn.Sequential(nn.Conv2d(1, 10, kernel_size=5),
                                               nn.MaxPool2d((2, 2), stride=None),
                                               nn.ReLU(),
                                               nn.Conv2d(10, 20, kernel_size=5),
```

```
                                                #nn.Dropout2d(),
                                                nn.MaxPool2d((2, 2), stride=None),
                                                nn.ReLU()
                                                )
        self.flatten = nn.Flatten()
        self.classification_head = nn.Sequential(nn.Linear(320, 64, bias=True),
                                                 nn.ReLU(),
                                                 nn.Linear(64, 10, bias=True))


    def forward(self, x):
        x = self.feature_extractor(x)
        x = self.flatten(x)    # x = x.view(-1, 320)
        x = self.classification_head(x)
        x = F.log_softmax(x, dim=1)
        return x



model = CNNNetwork().to(device)
model.train()

criterion = F.nll_loss
optimizer = optim.Adam(model.parameters(), lr=1e-3)
```

[4]:
```
losses = []
num_epochs = 3
model.train()

for epoch in range(1, num_epochs+1):
    epoch_loss = 0
    for idx, (x_batch, y_batch) in enumerate(train_loader):
        batch_size = len(x_batch)
        loader_size = len(train_loader)

        x_batch, y_batch = x_batch.to(device), y_batch.to(device)
        y_pred = model(x_batch)
        y_pred = y_pred.reshape(batch_size,-1)

        loss = criterion(y_pred, y_batch)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        epoch_loss += loss

        if (idx % 1000 == 0):
```

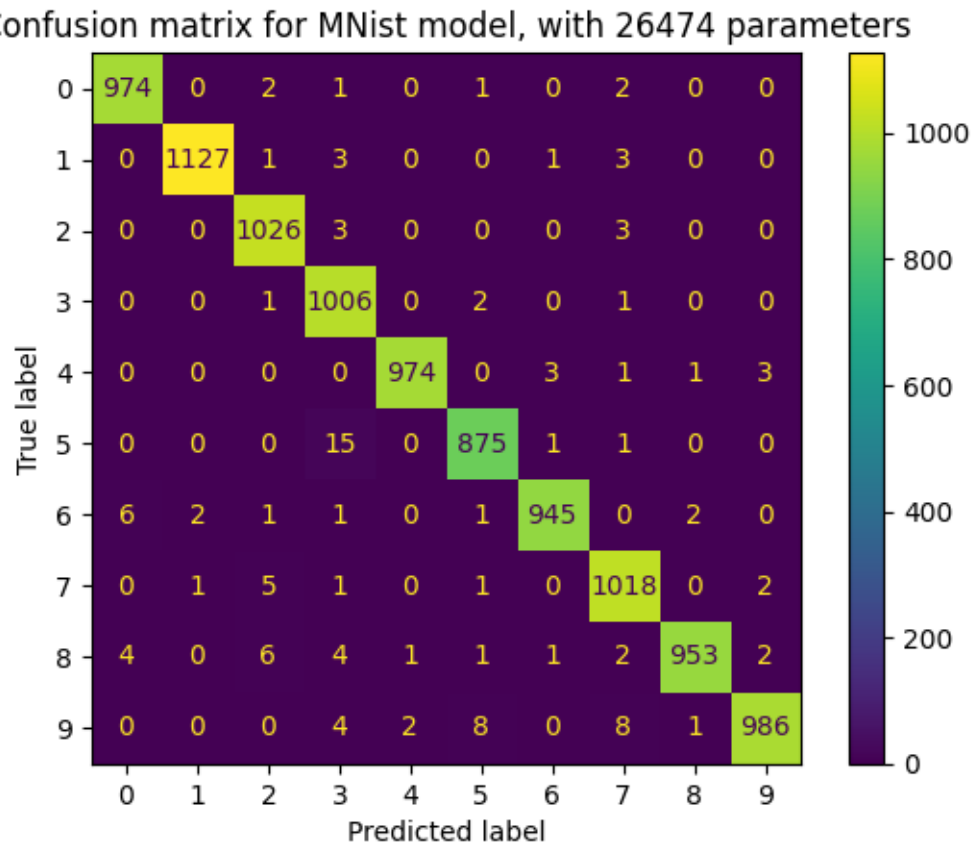
```
            progress = (idx / loader_size)
            print(f"{idx * batch_size} / {loader_size * batch_size} : Progress␣
↪= {progress:.2%}")

    if epoch%1 == 0:
        losses.append(epoch_loss.detach().cpu())
        print(f"Epoch {epoch}: Loss = {epoch_loss}")
```

```
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 1: Loss = 1742.9490966796875
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 2: Loss = 778.3883056640625
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
```

```
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 3: Loss = 590.4749755859375
```

[5]:
```python
model.eval()

losses = []
y_eval, y_preds = [], []


for x_batch, y_batch in test_loader:
    x_batch, y_batch = x_batch.to(device), y_batch.to(device)
    y_pred = model(x_batch)
    y_pred = y_pred.argmax(dim=1)

    y_eval += y_batch.detach().cpu().numpy().tolist()
    y_preds += y_pred.detach().cpu().numpy().tolist()


total_params = sum(p.numel() for p in model.parameters())
print(f"Number of parameters: {total_params}")

confusion_matrix = metrics.confusion_matrix(y_eval, y_preds)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =␣
 ↪confusion_matrix, display_labels = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

cm_display.plot()

plt.title(f"Confusion matrix for MNist model, with {total_params} parameters")
plt.show()

acc1 = accuracy_score(y_preds, y_eval)

f"{accuracy_score(y_preds, y_eval):.2%}"
```

```
Number of parameters: 26474
```



[5]: '98.84%'

[6]:
```
print("Model's state dict:")
for param_tensor in model.state_dict():
    print(param_tensor, "\t", model.state_dict()[param_tensor].size())

print("Optimizer's state dict:")
for var_name in optimizer.state_dict():
    print(var_name, "\t", optimizer.state_dict()[var_name])
```

```
Model's state dict:
feature_extractor.0.weight      torch.Size([10, 1, 5, 5])
feature_extractor.0.bias        torch.Size([10])
feature_extractor.3.weight      torch.Size([20, 10, 5, 5])
feature_extractor.3.bias        torch.Size([20])
classification_head.0.weight    torch.Size([64, 320])
classification_head.0.bias      torch.Size([64])
classification_head.2.weight    torch.Size([10, 64])
classification_head.2.bias      torch.Size([10])
Optimizer's state dict:
```

```
state    {0: {'step': tensor(45000.), 'exp_avg': tensor([[[[ 5.9395e-04,
-2.5405e-03, -2.5326e-03,  2.8868e-03, -3.6589e-03],
          [ 2.3277e-03,  8.8792e-05,  5.8034e-03,  7.7230e-03, -2.8851e-03],
          [-2.0003e-04, -2.8577e-03,  9.2847e-03,  6.5774e-03, -7.2076e-03],
          [-7.4372e-03, -8.0123e-03,  2.3022e-03, -1.3013e-03, -3.8307e-03],
          [-7.1725e-03, -6.9612e-03, -4.3974e-03, -9.3733e-04, -3.1843e-03]]],


        [[[ 6.0175e-03,  1.3476e-02,  2.5649e-02,  1.5866e-02,  7.6610e-03],
          [-5.7837e-03,  6.5175e-03,  1.0883e-02,  2.8940e-03, -4.8822e-03],
          [-5.7124e-03,  1.1340e-03,  2.8654e-03,  2.4404e-03, -1.0586e-02],
          [-2.3117e-03, -7.5987e-03, -1.8277e-03, -9.1777e-03, -1.2611e-02],
          [ 9.2007e-04, -1.6096e-03, -1.6028e-03, -1.3477e-02, -1.0514e-02]]],


        [[[ 1.8275e-02,  5.5715e-03, -8.7258e-03, -1.0734e-02, -7.5948e-03],
          [ 1.3527e-02,  6.4665e-03, -7.8348e-03, -9.0626e-03,  5.9020e-04],
          [ 1.0772e-02,  5.4534e-03, -3.5474e-03,  3.5611e-03,  6.5766e-03],
          [ 4.8645e-03,  1.1767e-03,  3.0604e-03, -3.3649e-04,  7.2366e-03],
          [-2.5089e-03, -3.1497e-03, -3.5327e-03,  2.3505e-03,  8.7156e-03]]],


        [[[ 3.0521e-03, -6.0719e-04,  1.8250e-04,  4.1379e-03,  4.3279e-03],
          [ 2.5124e-03,  2.6795e-03,  4.7256e-05, -3.6890e-03, -6.5033e-03],
          [-1.3997e-03,  5.0055e-03, -7.9583e-04,  2.7281e-03, -4.6484e-03],
          [-1.0678e-03,  4.0158e-03,  5.2227e-03,  6.0156e-03, -4.5618e-04],
          [ 4.0437e-03,  6.0477e-03,  2.7417e-03, -5.5872e-03, -4.2512e-04]]],


        [[[ 5.0612e-03, -2.3468e-03, -7.7911e-03, -3.7735e-03,  7.2000e-03],
          [ 7.1942e-04, -8.9734e-03, -6.0676e-03,  6.3674e-03,  1.4278e-02],
          [-7.3794e-03, -1.5646e-02,  2.9098e-04,  1.3601e-02,  1.5581e-02],
          [-1.8338e-02, -9.4652e-03,  1.1094e-02,  1.4857e-02,  1.8511e-02],
          [-1.9209e-02,  2.6619e-03,  1.1950e-02,  1.6825e-02,  1.2233e-02]]],


        [[[-6.6162e-04, -8.4559e-04, -9.0842e-04, -3.2750e-03, -3.1962e-03],
          [ 4.2140e-04,  5.7134e-04, -2.0903e-03, -2.9083e-03, -1.7859e-03],
          [ 3.0462e-03,  2.7328e-03, -7.0869e-04, -5.0076e-04,  1.5313e-03],
          [ 2.1244e-03, -1.5193e-04, -6.4552e-04,  3.4180e-03,  3.8922e-03],
          [ 2.4330e-05, -2.2963e-03, -8.9370e-04,  2.1281e-03,  1.7666e-03]]],


        [[[ 1.4513e-03,  9.7305e-03,  1.6806e-02,  7.1418e-03, -3.0536e-03],
          [-1.0747e-03,  1.0469e-02,  1.4940e-02,  3.9115e-04, -8.9077e-03],
          [-3.9308e-03,  1.3943e-03,  2.1413e-03, -4.8926e-03, -6.8188e-03],
          [ 6.5439e-04,  5.9002e-04, -1.0472e-02, -8.8506e-03,  3.2903e-03],
          [ 7.9233e-03,  2.8615e-03, -1.1279e-02, -4.3866e-03,  2.7185e-03]]],
```

```
        [[[ 6.0027e-04,  1.6049e-03,  7.4404e-03,  1.0460e-02,  1.6805e-02],
          [-3.2249e-03,  2.5889e-03,  7.3548e-03,  9.1547e-03,  7.0016e-03],
          [-6.8606e-03,  4.0367e-03,  1.4987e-02,  1.8249e-02,  1.2042e-02],
          [ 1.6011e-03,  1.1102e-02,  2.1541e-02,  2.3550e-02,  1.1897e-02],
          [-9.7489e-04,  5.6323e-03,  2.0639e-02,  1.9211e-02,  1.2908e-02]]],


        [[[-5.5874e-03, -3.5702e-03,  2.2915e-04,  6.4013e-03,  1.7711e-02],
          [-7.3718e-03, -6.8415e-03, -5.1741e-03,  8.4093e-03,  1.4299e-02],
          [-1.4145e-02, -1.1953e-02, -6.1614e-03,  8.1416e-03,  1.2341e-02],
          [-1.7843e-02, -1.2629e-02, -3.0430e-03,  1.3516e-02,  1.4036e-02],
          [-1.2582e-02, -1.2000e-02,  5.2497e-03,  1.2442e-02,  1.0440e-02]]],


        [[[ 5.9185e-03,  4.6382e-03, -1.3333e-02, -1.9846e-02, -1.7183e-03],
          [ 1.4347e-04, -1.1745e-02, -1.7604e-02, -9.5451e-03,  1.2028e-02],
          [-4.5027e-03, -1.6634e-02, -1.6967e-02,  1.3806e-03,  1.8121e-02],
          [-8.1766e-03, -1.0396e-02, -3.7186e-03,  8.1669e-03,  2.0170e-02],
          [-1.1368e-02,  1.1530e-05,  6.7719e-03,  1.2018e-02,  1.8394e-02]]]],
       device='cuda:0'), 'exp_avg_sq': tensor([[[[0.0076, 0.0052, 0.0039,
0.0059, 0.0064],
          [0.0087, 0.0048, 0.0049, 0.0049, 0.0048],
          [0.0081, 0.0048, 0.0040, 0.0056, 0.0041],
          [0.0074, 0.0061, 0.0036, 0.0054, 0.0043],
          [0.0077, 0.0063, 0.0051, 0.0057, 0.0058]]],


        [[[0.0079, 0.0066, 0.0069, 0.0068, 0.0081],
          [0.0071, 0.0069, 0.0087, 0.0086, 0.0074],
          [0.0057, 0.0052, 0.0068, 0.0090, 0.0066],
          [0.0060, 0.0042, 0.0042, 0.0050, 0.0064],
          [0.0089, 0.0057, 0.0056, 0.0066, 0.0066]]],


        [[[0.0124, 0.0108, 0.0112, 0.0149, 0.0172],
          [0.0130, 0.0120, 0.0106, 0.0122, 0.0138],
          [0.0153, 0.0138, 0.0122, 0.0127, 0.0144],
          [0.0150, 0.0125, 0.0133, 0.0138, 0.0149],
          [0.0142, 0.0114, 0.0129, 0.0139, 0.0167]]],


        [[[0.0064, 0.0051, 0.0050, 0.0049, 0.0053],
          [0.0046, 0.0039, 0.0043, 0.0042, 0.0042],
          [0.0037, 0.0051, 0.0048, 0.0055, 0.0052],
          [0.0056, 0.0042, 0.0033, 0.0047, 0.0063],
          [0.0066, 0.0059, 0.0050, 0.0050, 0.0061]]],
```

```
          [[[0.0060, 0.0068, 0.0063, 0.0054, 0.0061],
            [0.0064, 0.0066, 0.0052, 0.0054, 0.0076],
            [0.0061, 0.0054, 0.0045, 0.0076, 0.0088],
            [0.0065, 0.0038, 0.0064, 0.0088, 0.0084],
            [0.0060, 0.0049, 0.0072, 0.0076, 0.0080]]],


          [[[0.0017, 0.0017, 0.0015, 0.0017, 0.0019],
            [0.0020, 0.0018, 0.0014, 0.0023, 0.0023],
            [0.0019, 0.0015, 0.0020, 0.0023, 0.0021],
            [0.0016, 0.0017, 0.0021, 0.0019, 0.0023],
            [0.0012, 0.0021, 0.0022, 0.0022, 0.0020]]],


          [[[0.0111, 0.0121, 0.0097, 0.0099, 0.0108],
            [0.0106, 0.0087, 0.0089, 0.0082, 0.0108],
            [0.0101, 0.0090, 0.0053, 0.0085, 0.0094],
            [0.0090, 0.0073, 0.0074, 0.0107, 0.0091],
            [0.0107, 0.0098, 0.0107, 0.0099, 0.0087]]],


          [[[0.0124, 0.0117, 0.0111, 0.0095, 0.0088],
            [0.0125, 0.0118, 0.0100, 0.0087, 0.0085],
            [0.0111, 0.0100, 0.0079, 0.0074, 0.0093],
            [0.0080, 0.0078, 0.0083, 0.0091, 0.0117],
            [0.0063, 0.0073, 0.0094, 0.0106, 0.0113]]],


          [[[0.0117, 0.0107, 0.0102, 0.0089, 0.0070],
            [0.0099, 0.0088, 0.0080, 0.0073, 0.0075],
            [0.0091, 0.0073, 0.0060, 0.0041, 0.0082],
            [0.0085, 0.0071, 0.0046, 0.0065, 0.0095],
            [0.0090, 0.0064, 0.0050, 0.0072, 0.0083]]],


          [[[0.0126, 0.0121, 0.0127, 0.0108, 0.0137],
            [0.0095, 0.0129, 0.0121, 0.0090, 0.0175],
            [0.0095, 0.0177, 0.0102, 0.0110, 0.0228],
            [0.0117, 0.0158, 0.0088, 0.0155, 0.0229],
            [0.0125, 0.0111, 0.0112, 0.0160, 0.0209]]]], device='cuda:0')}, 1:
{'step': tensor(45000.), 'exp_avg': tensor([ 0.0095, -0.0022, -0.0167, -0.0053,
-0.0170, -0.0037, -0.0132, -0.0066,
         0.0129, -0.0203], device='cuda:0'), 'exp_avg_sq': tensor([0.0086,
0.0137, 0.0213, 0.0085, 0.0089, 0.0030, 0.0126, 0.0142, 0.0123,
        0.0323], device='cuda:0')}, 2: {'step': tensor(45000.), 'exp_avg':
tensor([[[[-4.0043e-05, -3.2945e-04,  3.6093e-03,  3.5170e-03, -2.3716e-05],
```

```
  [-1.7967e-04, -1.5533e-04, -2.2462e-04,  3.4399e-03,  5.5605e-08],
  [ 5.8682e-04,  8.5214e-04, -1.5929e-04,  8.6588e-04, -4.8490e-06],
  [ 6.8016e-04,  2.9170e-03, -5.2024e-05,  1.4798e-03,  4.8249e-05],
  [-1.6248e-04,  1.7014e-03, -2.9512e-05,  2.4779e-03,  4.5150e-04]],

 [[-2.6649e-05,  5.9015e-05, -6.1717e-06,  8.9757e-04,  7.3170e-04],
  [ 8.9692e-04,  8.7699e-04,  1.2198e-03,  4.7978e-05,  1.0664e-04],
  [ 5.9944e-06,  1.0308e-03,  5.9107e-04,  5.3411e-05,  1.1226e-03],
  [ 4.4401e-04, -1.8365e-05,  8.9194e-04,  5.0804e-06,  5.0506e-04],
  [ 1.0708e-03,  1.1225e-05,  7.4298e-05,  4.0634e-05,  1.5588e-03]],

 [[-5.5120e-04,  4.0957e-04,  3.0030e-04, -1.9605e-05,  4.8449e-06],
  [ 4.1068e-05,  1.4222e-03,  2.6144e-03,  1.7188e-03,  2.6033e-03],
  [-4.3128e-04, -2.6843e-04,  8.8049e-05,  3.4283e-04,  5.0896e-03],
  [ 3.5581e-03,  3.2589e-05, -2.1215e-04,  4.1003e-04,  2.7937e-03],
  [ 1.1489e-03, -1.0397e-04,  1.0702e-04,  1.6434e-03,  1.4085e-03]],

 …,

 [[-1.4371e-04, -1.7202e-05, -1.6292e-05,  1.8822e-03,  5.4069e-03],
  [ 3.3026e-03,  3.0575e-03,  2.3874e-03,  5.6766e-04,  6.9617e-04],
  [ 1.2002e-03,  2.4697e-03,  3.5668e-03,  3.2150e-04, -3.7381e-05],
  [-1.7267e-04,  8.0179e-04,  2.5093e-03,  1.9416e-03,  3.0806e-04],
  [ 1.1647e-03,  1.2417e-03,  7.2109e-04,  1.9225e-03,  1.9569e-03]],

 [[-1.1501e-04,  4.4832e-04,  7.8204e-04,  1.3697e-03,  1.5980e-03],
  [ 5.5660e-05, -1.8070e-04,  1.2895e-03,  3.0880e-03,  5.5365e-04],
  [-1.3215e-06, -1.7357e-04,  2.6812e-03,  2.9157e-03,  1.7035e-07],
  [ 9.2717e-04,  1.6338e-03,  3.0180e-03,  1.1974e-03,  2.2509e-06],
  [ 1.5494e-03,  3.1655e-03,  2.0174e-03,  1.3354e-03,  9.8638e-05]],

 [[-1.7628e-04, -3.0340e-04, -2.5528e-04,  1.6158e-03,  1.2031e-04],
  [-2.8274e-04, -2.6227e-04, -1.5199e-04, -2.2008e-04,  2.5821e-03],
  [-7.9634e-05,  5.9153e-04,  1.6747e-04,  6.1487e-04,  4.4721e-03],
  [-5.2202e-05,  1.2069e-03,  2.0137e-03,  2.5028e-03,  4.9337e-03],
  [-6.5971e-05,  2.5434e-04,  3.4009e-03,  2.1926e-03,  2.8931e-03]]],


[[[ 6.0612e-04, -9.4162e-04, -2.9766e-03,  4.8438e-05, -1.4521e-03],
  [-8.0461e-04,  1.1688e-03,  3.3678e-05, -5.7178e-07, -6.3339e-04],
  [-8.0260e-04, -5.4444e-04,  2.9923e-04,  2.1018e-04, -1.7244e-05],
  [ 4.1947e-04, -4.1428e-04, -2.0006e-04,  7.7417e-05, -3.8477e-06],
  [-3.0425e-04, -1.0968e-03, -1.4434e-04,  2.0962e-05, -5.8337e-07]],

 [[-2.1987e-03, -2.5873e-03, -1.3166e-04,  1.2372e-04,  2.7203e-05],
  [-2.4672e-04, -3.5302e-04, -3.4315e-04,  1.2492e-04, -5.8159e-04],
  [-3.1575e-04, -3.1597e-05, -1.7606e-05, -6.0675e-05, -3.2932e-05],
  [ 2.0110e-04,  6.5571e-04,  4.2448e-04,  1.5360e-04, -4.8053e-05],
```

```
  [-3.8117e-04, -3.7636e-04,  2.7154e-04,  4.8942e-04,  2.1144e-04]],

 [[-5.1113e-03, -7.3561e-03, -4.7627e-03, -4.4322e-03, -4.9109e-03],
  [-1.0629e-04, -1.0578e-03, -2.2570e-03, -5.2946e-03, -4.9085e-03],
  [-2.6826e-03, -6.5051e-04, -3.9557e-05, -1.0048e-03,  3.7315e-04],
  [ 1.6216e-04,  1.5610e-03,  5.0803e-04,  1.0811e-03,  7.0242e-04],
  [-1.4509e-03, -3.0626e-03, -1.8190e-03, -1.8207e-04,  7.4399e-05]],

 …,

 [[-1.1826e-03, -8.9790e-04,  1.2848e-03,  1.7362e-03,  8.0323e-04],
  [ 1.1427e-03, -1.0062e-03,  5.3469e-04,  1.1679e-03,  9.7596e-04],
  [ 1.1250e-03, -4.7803e-04, -3.5714e-04, -5.2614e-07,  2.1562e-04],
  [-2.0891e-04,  1.8353e-04,  9.4058e-04,  1.8664e-04, -1.4933e-04],
  [-1.6611e-03,  4.9705e-05,  7.3816e-04,  7.6423e-04,  1.0004e-03]],

 [[ 2.7607e-04, -1.5525e-03, -1.8843e-03, -1.5213e-04, -2.8401e-04],
  [ 7.0406e-05, -2.3440e-03, -1.1251e-03, -8.4252e-05, -1.4135e-04],
  [-1.6090e-03, -2.5743e-03,  3.1875e-05, -7.8165e-05, -2.2524e-04],
  [-3.0654e-03, -1.0398e-03,  6.5153e-05, -5.5650e-05, -2.8050e-04],
  [-1.3131e-03, -2.9361e-04,  2.9775e-05, -1.4673e-04, -3.1531e-04]],

 [[-1.4048e-03,  9.8491e-04,  1.6478e-04, -2.6434e-03, -2.4828e-04],
  [-2.6826e-04, -8.1180e-05, -2.4252e-03, -2.7624e-03,  1.6754e-04],
  [ 6.6743e-04, -1.2736e-03, -5.1534e-03, -1.6174e-03,  3.0276e-04],
  [ 6.0913e-04, -2.9904e-03, -2.8638e-03,  1.4426e-04,  7.5678e-05],
  [ 5.7513e-04, -3.6528e-03, -1.3669e-03,  2.6595e-04, -1.9180e-05]]],


[[[-6.2017e-05,  2.9352e-04, -8.4017e-05, -1.3108e-03, -8.3698e-03],
  [-1.2609e-04, -4.9381e-03, -8.3702e-05, -3.5888e-04, -4.0818e-03],
  [-5.8282e-04, -8.0305e-03, -3.8135e-03, -3.1351e-04, -5.4420e-05],
  [-1.2912e-03, -6.4627e-03, -9.3554e-03, -2.7765e-04,  7.5554e-06],
  [ 1.6372e-04,  2.2504e-04, -5.5386e-03,  4.2648e-04,  2.0402e-04]],

 [[ 8.4736e-04, -9.7925e-05, -2.6956e-03, -2.4671e-03,  3.8122e-05],
  [-6.0883e-04, -3.4250e-05, -2.1910e-03, -2.9523e-03, -8.2095e-04],
  [-2.1939e-04,  4.8360e-04, -4.1615e-03, -1.8674e-04, -1.9896e-04],
  [-5.6600e-04, -1.0890e-03, -1.4268e-03,  2.2636e-04, -6.8943e-04],
  [-3.2739e-04, -2.5746e-03, -2.7750e-04, -2.8144e-04, -1.4355e-04]],

 [[ 1.4439e-03,  2.0858e-03, -5.3958e-03, -6.7012e-03,  1.0382e-03],
  [-3.3194e-04,  9.4895e-05, -7.3868e-04, -2.1153e-03, -1.5176e-03],
  [-9.4204e-04, -5.2481e-04,  1.2266e-03,  2.2460e-04, -2.9239e-03],
  [-3.4759e-03, -2.8203e-03, -4.7601e-03, -7.9800e-03, -1.1134e-02],
  [ 1.1763e-03, -3.4782e-03, -6.9712e-03, -1.0514e-02, -9.5031e-03]],

 …,
```

```
 [[-3.7084e-04, -9.2423e-04, -1.0649e-03, -1.8979e-03, -1.1982e-03],
  [-4.6615e-04, -2.0532e-03, -4.5545e-03, -7.0178e-03, -5.2909e-03],
  [ 1.6165e-04, -1.1370e-03, -6.7937e-03, -8.3768e-03, -2.6810e-03],
  [-2.9479e-04, -8.9891e-04, -1.8205e-03, -4.1102e-03,  1.4734e-04],
  [-6.1542e-04, -1.7445e-03, -3.8898e-04, -7.8041e-05, -2.0147e-05]],

 [[-2.5440e-04, -2.0748e-03, -1.7774e-03, -4.7207e-03, -4.3301e-03],
  [ 2.1264e-05, -3.8816e-03, -2.0502e-03, -6.0966e-03, -4.7170e-03],
  [-3.6038e-04, -5.1544e-03, -2.0950e-03, -4.8345e-03, -1.9697e-03],
  [-1.0952e-03, -3.0507e-03, -3.1657e-03, -1.6868e-03, -1.0165e-04],
  [-8.0025e-04, -4.0968e-03, -4.0528e-03,  3.9975e-04,  1.8481e-04]],

 [[-6.8486e-05, -2.6369e-04, -2.1497e-03,  3.3144e-04, -4.9112e-03],
  [-3.0360e-05,  1.1412e-03, -4.9685e-03, -2.3179e-03, -1.8172e-03],
  [-2.9378e-04,  3.2208e-03, -6.1563e-03, -1.7925e-03, -3.3391e-03],
  [ 1.0980e-03,  1.5382e-03, -3.0608e-03, -2.4684e-03, -4.7304e-03],
  [ 2.0885e-03,  1.9808e-04, -1.1522e-03, -6.3009e-03, -2.4661e-03]]],


…,


[[[ 3.0015e-05, -5.2841e-05, -6.7457e-04,  4.9914e-09, -5.9763e-04],
  [ 8.4981e-06,  6.4644e-04, -1.6001e-03, -6.5572e-06, -3.2209e-05],
  [-9.1431e-05,  1.8806e-04, -2.8546e-03, -2.9434e-04, -3.9029e-05],
  [-7.1125e-05, -4.8277e-04, -4.1844e-03, -3.6744e-03, -1.2547e-04],
  [-1.8301e-05, -1.5279e-05, -1.7846e-04, -3.3282e-03, -4.5274e-04]],

 [[-2.3567e-04, -8.8831e-05, -1.8022e-04, -1.1590e-03, -7.8436e-04],
  [-1.1000e-04, -2.9522e-05,  8.6233e-05, -7.9995e-04, -1.9915e-03],
  [-2.1648e-04, -7.6399e-05, -4.5445e-04, -3.0251e-03, -4.4780e-04],
  [-2.6215e-04,  2.5286e-04,  3.7748e-04, -9.8482e-04, -2.7868e-04],
  [-3.7418e-04, -5.6785e-04, -1.5027e-03, -1.4255e-03, -1.0497e-03]],

 [[-1.9927e-04, -9.7849e-05,  8.8439e-04, -3.7246e-03, -7.2208e-03],
  [-4.1603e-05, -3.2896e-04, -1.0298e-03, -4.1697e-03, -4.4878e-03],
  [ 5.7729e-04,  1.1406e-03,  8.3442e-04, -3.7989e-04, -6.8519e-04],
  [ 1.4016e-05,  4.2090e-04,  7.8965e-04, -2.3843e-03, -5.6917e-03],
  [ 8.6130e-04, -1.9084e-03, -5.4654e-03, -7.6075e-03, -7.9426e-03]],

 …,

 [[ 2.3134e-03, -1.0861e-04, -9.4362e-04, -1.1712e-04,  3.7426e-04],
  [ 5.8325e-04,  5.9667e-04,  2.9987e-04, -7.4696e-04, -4.1064e-03],
  [-3.5534e-04, -3.6691e-04, -2.8464e-03, -7.7584e-03, -9.4965e-03],
  [-2.1844e-04, -2.8935e-04, -2.5516e-04, -1.4754e-03, -3.8833e-03],
  [-1.0195e-03, -1.0508e-03, -1.3088e-03, -9.4031e-04, -8.7996e-04]],
```

```
 [[ 7.9501e-04, -1.9313e-05, -2.0533e-03, -2.6766e-04, -1.0608e-03],
  [ 7.6484e-05, -1.3331e-03, -1.4030e-03,  2.3156e-04, -2.0270e-03],
  [-3.8546e-05, -1.5123e-03, -2.0657e-03, -5.5331e-04, -2.0020e-03],
  [ 1.7268e-06, -1.3373e-03, -1.5646e-03, -9.1979e-04, -1.0177e-03],
  [-6.1002e-05, -5.8223e-04, -1.4319e-03, -1.6152e-03, -1.3041e-04]],

 [[-2.3271e-05,  9.5865e-04,  1.6223e-03, -1.7266e-03, -1.4751e-04],
  [-5.3221e-05,  1.4277e-03,  8.8500e-04, -3.0726e-03, -2.5764e-04],
  [ 9.8267e-04, -3.5857e-05, -3.2992e-04, -4.1956e-03, -2.0061e-04],
  [ 1.3454e-03,  9.8488e-05, -2.7567e-04, -2.5567e-03, -2.1736e-04],
  [ 6.8180e-04,  8.5216e-06, -5.7717e-04, -1.0107e-03, -2.3935e-03]]],


[[[-3.8970e-04, -3.5839e-05, -6.5540e-04, -3.1655e-04, -4.1985e-05],
  [-8.6631e-04, -1.8725e-06, -4.1903e-04, -1.3133e-03, -1.8142e-04],
  [-2.8132e-04, -4.3915e-05,  1.3254e-04, -6.7899e-04, -2.9112e-04],
  [-1.1001e-03, -1.5627e-04, -7.8410e-05, -1.6332e-04, -2.3678e-04],
  [-1.3119e-03, -1.0677e-03, -2.3419e-04, -3.3621e-04, -7.2613e-05]],

 [[-9.0613e-05, -3.5542e-05, -3.2845e-04, -1.0619e-03, -5.9668e-04],
  [-8.0221e-05, -5.0716e-04, -4.3311e-04, -3.1599e-04, -6.2511e-04],
  [-3.8151e-05, -1.4366e-04, -7.2745e-04, -3.8611e-04, -3.8877e-05],
  [-6.3913e-05, -7.7383e-04, -7.7864e-05, -3.6920e-04, -7.7840e-05],
  [ 6.9426e-05, -1.3773e-04, -7.4987e-05, -1.2553e-04, -2.3579e-04]],

 [[-7.2014e-05,  1.0829e-04,  2.5807e-04, -4.8792e-04, -4.3392e-04],
  [-9.6138e-04, -1.9337e-03, -3.5094e-03, -2.5590e-03, -1.2923e-03],
  [-4.6888e-04, -2.6574e-03, -3.8887e-03, -2.8376e-03, -1.1515e-03],
  [-2.9848e-04, -1.0746e-03, -9.9520e-04, -2.2061e-04,  3.0129e-06],
  [-1.0050e-03, -6.9569e-04, -3.1428e-04, -7.8099e-04, -6.3049e-04]],

 …,

 [[-1.6105e-03, -1.9928e-03, -2.2630e-03, -2.4015e-03, -2.0542e-03],
  [ 1.2676e-04, -3.9661e-04, -3.6618e-04, -7.0835e-04, -1.9093e-03],
  [ 4.5798e-04,  5.7205e-05, -3.3885e-04, -6.3040e-04, -6.9577e-04],
  [-2.9295e-04, -1.2669e-03, -1.5332e-03, -9.0873e-04, -9.7135e-04],
  [-5.1766e-05, -2.8425e-04, -8.9192e-04, -4.4748e-04, -5.9893e-04]],

 [[-3.8324e-04, -7.7867e-04, -1.4014e-04,  1.1216e-04,  1.4749e-05],
  [-1.5264e-03, -1.8052e-04, -2.5099e-04, -2.9545e-04, -6.3089e-05],
  [-1.2033e-03,  8.1417e-06, -5.4149e-04, -5.1962e-04, -9.6821e-05],
  [-6.3505e-04, -2.6703e-05, -6.4401e-04, -4.4481e-04,  1.5885e-05],
  [-2.4294e-04, -2.4954e-04, -4.1895e-04, -1.6628e-04,  2.3893e-04]],

 [[-2.2944e-04,  2.6879e-04, -4.4231e-04, -1.0956e-03, -4.2505e-04],
  [ 1.7938e-04, -7.8577e-04, -3.7837e-04, -8.3347e-04, -1.0936e-03],
```

```
          [-1.6915e-04, -1.6478e-03, -1.9417e-04,  1.3358e-05, -1.0830e-03],
          [-1.1093e-03, -1.7220e-03,  7.8506e-05, -4.8585e-04, -1.0683e-03],
          [-8.9634e-04, -5.2228e-04,  2.6145e-06, -7.0578e-04, -1.1076e-03]]],


        [[[ 2.9587e-04,  2.0053e-04,  5.8504e-04,  8.4387e-04, -5.0790e-04],
          [ 9.0337e-04,  4.7709e-04,  1.2589e-03,  2.5389e-03, -9.0394e-04],
          [ 5.6701e-04,  7.6081e-04,  1.3702e-03,  2.1922e-03,  1.8325e-04],
          [ 1.2820e-03,  1.0679e-05, -6.0944e-05, -6.1319e-05, -6.4228e-05],
          [ 1.6571e-03,  1.4203e-03, -2.7546e-05,  1.7383e-05, -1.0763e-04]],

         [[ 6.9656e-04,  9.9589e-04,  2.1549e-03,  3.2072e-03,  9.5382e-04],
          [ 1.9949e-04,  4.8702e-04,  1.1619e-04,  2.0380e-04,  9.4979e-04],
          [ 1.2913e-03,  1.8267e-03,  1.8285e-03,  9.4442e-05,  1.6986e-05],
          [-3.9919e-06,  8.1895e-04, -3.8860e-05,  3.9395e-05,  8.2442e-05],
          [ 6.1088e-04,  5.5317e-04,  1.8384e-04,  8.3188e-05,  6.9830e-05]],

         [[ 5.7803e-05,  5.2374e-04,  9.1783e-04,  1.1680e-03,  6.5469e-04],
          [ 1.6576e-03,  3.9446e-03,  2.6366e-03, -1.4173e-03, -8.5367e-04],
          [ 1.0922e-02,  1.3476e-02,  1.0171e-02,  3.1446e-03,  1.3701e-03],
          [ 2.0298e-03,  8.0129e-04,  3.6795e-04,  1.7447e-03,  2.5290e-03],
          [ 1.1042e-03,  3.7406e-04,  2.4038e-03,  4.8132e-03,  2.3488e-03]],

         …,

         [[ 7.9993e-03,  8.6944e-03,  9.6519e-03,  1.0232e-02,  6.1737e-03],
          [ 2.2544e-03, -2.3841e-04,  1.2261e-03,  4.0610e-03,  5.9860e-03],
          [ 1.8270e-04,  2.2370e-04,  8.2269e-04,  1.3470e-04,  8.8200e-04],
          [ 1.5489e-03,  2.5436e-03,  1.6645e-03, -3.7132e-05, -4.8279e-05],
          [ 7.1550e-03,  5.0432e-03,  2.3624e-03, -1.6551e-04,  5.0925e-04]],

         [[ 2.0963e-03,  7.6749e-04, -7.2916e-05, -5.7997e-05,  5.9303e-04],
          [ 1.5825e-03,  1.2970e-05,  1.0607e-03,  8.0937e-04, -5.1287e-04],
          [ 7.6226e-04,  1.9835e-03,  4.7628e-03,  2.1891e-03, -4.9355e-04],
          [ 1.0158e-03,  1.9790e-03,  2.5724e-03,  1.2027e-04, -7.0414e-05],
          [ 1.6521e-03,  2.6451e-03,  8.1557e-04, -3.1237e-04, -2.0197e-05]],

         [[ 1.4917e-04,  6.7824e-04,  1.7141e-03,  8.8126e-04,  4.9385e-04],
          [ 6.7310e-04,  1.7385e-03,  1.1177e-03,  2.4885e-03,  2.5086e-03],
          [ 1.5019e-03,  1.9748e-03,  2.7078e-04,  3.7236e-03,  3.6911e-03],
          [ 1.8246e-03,  1.7289e-03,  1.9230e-03,  4.4473e-03,  2.6671e-03],
          [ 1.3854e-03,  1.9771e-03,  3.3724e-03,  2.6698e-03,  6.6473e-04]]]],
       device='cuda:0'), 'exp_avg_sq': tensor([[[[4.1200e-05, 1.5835e-04,
5.9776e-04, 3.4430e-04, 5.5082e-05],
          [1.2581e-04, 2.3926e-05, 5.4081e-04, 7.7180e-04, 1.2534e-04],
          [1.9526e-04, 1.2260e-04, 3.1961e-04, 9.7018e-04, 2.9471e-04],
          [8.3031e-05, 3.1470e-04, 3.5887e-04, 1.0829e-03, 3.9340e-04],
          [6.3918e-05, 1.3145e-04, 3.8709e-04, 7.9637e-04, 4.7304e-04]],
```

```
 [[4.9566e-05, 1.8220e-05, 4.9155e-05, 7.9182e-05, 3.7773e-05],
  [9.7747e-05, 1.0371e-04, 6.5717e-05, 4.0043e-05, 7.3242e-05],
  [3.8527e-05, 8.2770e-05, 7.5606e-05, 6.0042e-05, 7.3626e-05],
  [5.8587e-05, 2.7390e-05, 1.1356e-05, 3.6293e-05, 1.6029e-04],
  [6.9590e-05, 4.2654e-05, 3.5422e-05, 4.0596e-05, 3.5210e-05]],

 [[6.4440e-04, 5.3512e-04, 1.9563e-04, 8.2769e-05, 1.4026e-04],
  [2.8027e-04, 3.4252e-04, 1.9739e-04, 5.2414e-05, 1.5343e-04],
  [2.3391e-04, 1.9825e-04, 1.7307e-04, 2.0038e-04, 4.2982e-04],
  [1.2283e-03, 8.7575e-04, 4.4089e-04, 4.7261e-04, 7.4984e-04],
  [5.8418e-04, 3.7077e-04, 4.1246e-04, 6.0766e-04, 5.1750e-04]],

 …,

 [[2.2478e-04, 1.2070e-04, 1.0439e-04, 4.3708e-04, 7.9944e-04],
  [7.7576e-04, 7.4756e-04, 3.7373e-04, 2.5599e-04, 6.3089e-04],
  [3.2224e-04, 4.4763e-04, 3.5030e-04, 1.7134e-04, 2.5777e-04],
  [1.7181e-04, 1.5715e-04, 1.3602e-04, 9.2893e-05, 3.1061e-04],
  [4.1506e-04, 3.5228e-04, 1.3405e-04, 1.5948e-04, 2.8339e-04]],

 [[2.8521e-05, 7.3375e-05, 2.9912e-04, 2.6501e-04, 7.2328e-05],
  [2.7853e-05, 7.4061e-05, 4.5499e-04, 2.8775e-04, 2.8659e-05],
  [1.0822e-05, 8.7863e-05, 6.1356e-04, 2.3210e-04, 6.7912e-06],
  [3.1944e-05, 2.4717e-04, 5.9989e-04, 1.5846e-04, 1.7674e-05],
  [1.0208e-04, 3.0813e-04, 3.4142e-04, 1.0365e-04, 3.5858e-05]],

 [[2.3310e-04, 1.0120e-04, 8.0232e-05, 6.1896e-04, 7.4136e-04],
  [3.4035e-04, 9.1977e-05, 3.7502e-05, 7.5683e-04, 1.1736e-03],
  [3.7579e-04, 5.9364e-05, 4.4572e-05, 9.9564e-04, 1.5189e-03],
  [3.5549e-04, 4.1296e-05, 1.6902e-04, 1.3768e-03, 1.3220e-03],
  [2.8026e-04, 3.4035e-05, 3.6093e-04, 1.3839e-03, 7.3564e-04]]],


[[[2.1863e-04, 2.8888e-04, 1.4296e-04, 3.6038e-05, 1.4057e-04],
  [3.0533e-04, 1.5348e-04, 6.2893e-05, 1.5802e-05, 1.5009e-04],
  [3.8908e-04, 5.8005e-05, 2.3256e-05, 1.1676e-05, 1.3015e-04],
  [3.9461e-04, 2.8405e-04, 4.6731e-05, 8.8848e-06, 3.3458e-05],
  [5.2867e-04, 6.9766e-04, 1.7947e-04, 1.7421e-05, 4.6173e-06]],

 [[1.0459e-04, 4.5899e-05, 2.3440e-05, 2.2502e-05, 1.7032e-05],
  [1.8817e-04, 7.2662e-05, 6.9888e-06, 2.9428e-06, 4.0299e-06],
  [5.7090e-05, 2.3364e-05, 1.1353e-05, 5.8414e-06, 4.5939e-06],
  [3.9206e-05, 1.0251e-04, 8.8776e-05, 2.5304e-05, 1.2378e-05],
  [1.4851e-04, 2.0512e-04, 1.5141e-04, 6.0979e-05, 4.4108e-05]],

 [[7.0463e-04, 4.3628e-04, 1.5125e-04, 9.1663e-05, 2.0995e-04],
  [5.4860e-04, 3.2522e-04, 3.4943e-04, 5.5163e-04, 4.7895e-04],
```

```
  [3.5448e-04, 4.1290e-04, 1.2898e-03, 1.2447e-03, 5.4862e-04],
  [4.6666e-04, 1.1503e-03, 1.0324e-03, 4.5737e-04, 1.2416e-04],
  [1.7944e-03, 1.2330e-03, 2.2247e-04, 3.2261e-05, 1.2907e-04]],

 …,

 [[1.6749e-03, 1.4347e-03, 1.0949e-03, 7.2870e-04, 3.0724e-04],
  [1.3096e-03, 9.6367e-04, 4.1407e-04, 1.3512e-04, 7.3808e-05],
  [5.0873e-04, 5.5698e-04, 2.5641e-04, 4.4222e-05, 2.4028e-05],
  [1.2917e-04, 1.8204e-04, 4.1807e-04, 3.2849e-04, 9.6050e-05],
  [1.8028e-04, 5.2841e-04, 1.0187e-03, 9.7089e-04, 5.0624e-04]],

 [[7.0811e-05, 1.6383e-04, 1.3391e-04, 4.0708e-05, 7.3616e-05],
  [2.2160e-04, 2.3063e-04, 5.4240e-05, 1.4465e-05, 1.0502e-04],
  [3.2129e-04, 1.1401e-04, 4.0146e-06, 4.0485e-05, 1.4084e-04],
  [2.0286e-04, 4.1756e-05, 1.7721e-05, 6.0749e-05, 1.2136e-04],
  [1.1709e-04, 8.6369e-05, 3.3133e-05, 3.4834e-05, 7.6506e-05]],

 [[2.0145e-04, 2.9609e-04, 2.5163e-04, 4.3313e-04, 2.1658e-04],
  [2.2024e-04, 6.9625e-04, 5.0149e-04, 4.9539e-04, 1.6196e-04],
  [5.2992e-04, 1.3159e-03, 6.5006e-04, 2.9591e-04, 3.7483e-05],
  [1.0890e-03, 1.2634e-03, 5.7482e-04, 1.1884e-04, 5.1935e-05],
  [1.5188e-03, 7.5581e-04, 5.0220e-04, 1.2829e-04, 8.0048e-05]]],


[[[1.8124e-04, 2.7245e-04, 2.1781e-04, 1.3630e-04, 1.3310e-04],
  [2.3560e-04, 4.4029e-04, 2.0296e-04, 7.6049e-05, 5.6357e-05],
  [1.9848e-04, 7.1926e-04, 3.6073e-04, 6.6780e-05, 3.1643e-05],
  [1.5996e-04, 6.4464e-04, 5.8200e-04, 1.5506e-04, 5.1050e-05],
  [2.1105e-04, 2.8173e-04, 3.5831e-04, 1.7863e-04, 1.4066e-04]],

 [[1.8747e-05, 3.2590e-05, 4.7681e-05, 2.9594e-05, 2.2917e-05],
  [2.3036e-05, 5.0691e-05, 7.8339e-05, 4.2279e-05, 2.3660e-05],
  [1.0340e-04, 1.3201e-04, 1.7657e-04, 1.2598e-04, 4.5100e-05],
  [4.6006e-05, 6.1601e-05, 7.5996e-05, 5.1177e-05, 5.7464e-05],
  [4.1892e-05, 7.6765e-05, 7.7211e-05, 5.4647e-05, 4.7406e-05]],

 [[6.4788e-04, 4.5919e-04, 4.1301e-04, 5.5004e-04, 7.0638e-04],
  [2.0546e-04, 1.9973e-04, 3.4403e-04, 6.1928e-04, 5.9935e-04],
  [1.1050e-04, 1.3562e-04, 2.4853e-04, 3.2025e-04, 4.1377e-04],
  [4.1484e-04, 7.6942e-04, 8.9066e-04, 8.2959e-04, 7.8558e-04],
  [1.7921e-03, 2.7829e-03, 3.2404e-03, 2.4306e-03, 1.5015e-03]],

 …,

 [[7.9568e-05, 1.4527e-04, 3.3515e-04, 4.8183e-04, 3.8351e-04],
  [2.2291e-04, 2.0601e-04, 4.6268e-04, 4.4075e-04, 4.1110e-04],
  [9.4922e-04, 9.5332e-04, 9.6379e-04, 1.1138e-03, 7.7637e-04],
```

```
  [4.3320e-04, 4.4886e-04, 4.9589e-04, 6.4332e-04, 5.1479e-04],
  [1.6773e-05, 5.0030e-05, 2.7933e-05, 4.3676e-05, 6.6003e-05]],

 [[2.8330e-04, 3.2014e-04, 2.8063e-04, 1.9957e-04, 2.2639e-04],
  [2.9372e-04, 3.4643e-04, 2.9718e-04, 1.7208e-04, 2.4270e-04],
  [2.7755e-04, 3.9921e-04, 1.6787e-04, 1.2298e-04, 1.3826e-04],
  [2.2924e-04, 2.0271e-04, 8.3158e-05, 6.5305e-05, 7.7460e-05],
  [1.1157e-04, 1.4573e-04, 1.2509e-04, 1.4384e-04, 1.1921e-04]],

 [[7.6307e-05, 5.4043e-04, 7.7358e-04, 6.8384e-04, 4.3957e-04],
  [9.4407e-05, 6.8883e-04, 7.9506e-04, 7.9934e-04, 5.4368e-04],
  [2.3083e-04, 6.3396e-04, 7.6147e-04, 7.0968e-04, 4.8945e-04],
  [3.1401e-04, 4.2206e-04, 5.6448e-04, 5.2783e-04, 4.2121e-04],
  [3.0035e-04, 3.9812e-04, 3.9151e-04, 4.0316e-04, 3.7470e-04]]],


…,


[[[6.5713e-05, 6.7358e-05, 1.7070e-05, 1.6027e-05, 7.7168e-05],
  [1.3647e-04, 1.9065e-04, 1.9106e-04, 1.1502e-05, 3.9903e-05],
  [2.2324e-04, 3.9804e-04, 8.8688e-04, 1.6413e-04, 3.8450e-05],
  [2.4684e-04, 2.3349e-04, 7.4436e-04, 7.1743e-04, 2.3160e-04],
  [1.9834e-04, 1.7981e-04, 2.7759e-04, 3.6428e-04, 3.0274e-04]],

 [[4.6895e-05, 4.5538e-05, 3.5230e-05, 1.7668e-05, 3.6982e-05],
  [7.5157e-06, 6.2656e-05, 1.9476e-04, 1.4111e-04, 4.1043e-05],
  [2.8621e-05, 5.6131e-05, 1.0769e-04, 1.8448e-04, 1.0004e-04],
  [4.7144e-05, 1.5687e-04, 8.1611e-05, 4.7631e-05, 1.2380e-04],
  [3.1367e-05, 8.1703e-05, 1.1994e-04, 1.0276e-04, 5.9330e-05]],

 [[1.2048e-04, 5.5167e-05, 2.5589e-04, 9.3125e-04, 1.6163e-03],
  [2.0827e-04, 2.4926e-04, 9.6984e-04, 1.4417e-03, 1.2171e-03],
  [6.3318e-04, 7.3644e-04, 3.2555e-04, 2.0210e-04, 3.0968e-04],
  [1.4220e-03, 2.1829e-03, 2.3568e-03, 1.7242e-03, 1.4506e-03],
  [1.2332e-03, 1.3450e-03, 2.1328e-03, 2.6110e-03, 2.2159e-03]],

 …,

 [[5.3022e-04, 6.4861e-04, 5.9322e-04, 5.4502e-04, 4.3748e-04],
  [3.2404e-04, 4.2835e-04, 8.1617e-04, 9.8230e-04, 6.6998e-04],
  [8.9708e-05, 1.1454e-04, 5.4556e-04, 1.0156e-03, 1.3319e-03],
  [2.9525e-05, 9.4618e-05, 1.7436e-04, 2.0947e-04, 6.9504e-04],
  [8.6042e-05, 1.3867e-04, 1.8543e-04, 1.0067e-04, 9.4895e-05]],

 [[2.7688e-04, 3.8804e-04, 3.4358e-04, 1.2223e-04, 1.3959e-04],
  [4.0378e-04, 3.6301e-04, 2.9333e-04, 7.0199e-05, 1.5633e-04],
  [2.0569e-04, 2.6317e-04, 2.6246e-04, 7.7464e-05, 1.1052e-04],
```

```
  [7.4755e-05, 1.9998e-04, 2.3413e-04, 4.7984e-05, 1.1479e-04],
  [5.9641e-05, 2.6632e-04, 1.9541e-04, 1.1603e-04, 2.1679e-04]],

 [[4.2109e-05, 2.1783e-04, 4.3210e-04, 4.7891e-04, 2.0877e-04],
  [1.7121e-05, 4.9456e-04, 5.3748e-04, 6.5656e-04, 2.0131e-04],
  [7.5055e-05, 5.4103e-04, 3.0531e-04, 7.5561e-04, 2.7061e-04],
  [1.7057e-04, 2.8609e-04, 1.0161e-04, 7.1003e-04, 4.2491e-04],
  [1.1689e-04, 1.2021e-04, 1.3911e-04, 6.5316e-04, 3.6645e-04]]],


[[[6.3310e-05, 6.8526e-05, 1.9751e-04, 1.6186e-04, 8.4676e-05],
  [6.4087e-05, 2.4280e-05, 1.9923e-04, 2.6209e-04, 4.7951e-04],
  [4.6286e-04, 2.5075e-05, 3.6541e-05, 7.9153e-05, 3.0057e-04],
  [9.6020e-04, 2.2642e-04, 1.6754e-05, 4.5494e-05, 1.0600e-04],
  [3.9271e-04, 3.8071e-04, 1.2487e-04, 6.0418e-05, 1.6092e-04]],

 [[1.1965e-05, 2.0828e-05, 4.9340e-05, 1.2446e-04, 3.3827e-04],
  [5.9112e-05, 3.2564e-05, 6.9709e-05, 5.0482e-05, 4.4271e-05],
  [2.0783e-04, 2.1025e-04, 6.7371e-05, 9.7444e-05, 8.5920e-05],
  [3.6177e-05, 2.0297e-04, 1.3764e-04, 3.8561e-05, 4.7140e-05],
  [3.3191e-05, 1.8997e-05, 5.8161e-05, 5.0447e-05, 1.7785e-05]],

 [[1.4283e-04, 1.4395e-04, 2.3996e-04, 4.3558e-04, 6.0125e-04],
  [4.7749e-04, 1.2350e-03, 2.6797e-03, 2.6983e-03, 1.3461e-03],
  [4.1599e-04, 9.7893e-04, 1.7335e-03, 2.1372e-03, 1.5689e-03],
  [3.6961e-04, 3.5597e-04, 5.6961e-04, 5.7802e-04, 4.2678e-04],
  [1.2762e-03, 1.3171e-03, 1.0403e-03, 7.5836e-04, 5.8660e-04]],

 …,

 [[1.8048e-04, 3.2885e-04, 5.7026e-04, 7.9452e-04, 9.3554e-04],
  [1.5269e-04, 9.2512e-05, 7.7476e-05, 1.1748e-04, 2.3688e-04],
  [2.7280e-04, 6.5888e-04, 4.7560e-04, 3.6411e-04, 2.6053e-04],
  [3.3485e-05, 4.9690e-04, 8.4779e-04, 5.0705e-04, 2.8300e-04],
  [2.9913e-05, 5.6074e-05, 2.0419e-04, 2.6769e-04, 1.8218e-04]],

 [[1.3307e-04, 1.5443e-04, 1.0662e-04, 1.7760e-05, 2.9311e-05],
  [1.9713e-04, 5.8463e-05, 5.6291e-05, 4.1442e-05, 5.2580e-05],
  [1.4163e-04, 8.5530e-06, 3.4033e-05, 4.2980e-05, 8.3473e-05],
  [8.7919e-05, 8.9537e-06, 2.3712e-05, 1.8887e-05, 1.2233e-04],
  [5.7381e-05, 1.8341e-05, 3.0910e-05, 1.8152e-05, 6.6622e-05]],

 [[1.2971e-04, 1.4749e-04, 1.2305e-04, 3.0584e-04, 2.6918e-04],
  [2.2455e-04, 2.2489e-04, 1.3816e-04, 2.6921e-04, 1.9203e-04],
  [3.0769e-04, 4.2401e-04, 1.1983e-04, 2.1487e-04, 1.0447e-04],
  [2.1400e-04, 4.3459e-04, 9.3591e-05, 1.7406e-04, 9.1905e-05],
  [1.1159e-04, 3.0745e-04, 9.5433e-05, 1.3504e-04, 8.2445e-05]]],
```

```
        [[[1.0345e-04, 4.7341e-05, 1.0175e-04, 1.5280e-04, 1.3873e-04],
          [1.1539e-04, 3.0706e-05, 1.7687e-04, 2.9199e-04, 2.5685e-04],
          [1.0194e-04, 1.6511e-05, 1.6169e-04, 2.2825e-04, 1.8440e-04],
          [1.3549e-04, 2.6205e-05, 4.7270e-05, 1.5232e-04, 1.5161e-04],
          [1.5712e-04, 1.7436e-05, 3.7286e-05, 1.7051e-04, 1.6521e-04]],

         [[2.9470e-05, 3.6523e-05, 6.8722e-05, 8.6893e-05, 9.2874e-05],
          [2.6570e-05, 1.5577e-05, 4.4404e-05, 4.0280e-05, 2.5240e-05],
          [2.9744e-05, 5.1969e-05, 6.4616e-05, 3.1429e-05, 1.9614e-05],
          [2.4948e-05, 8.1909e-06, 6.9739e-06, 3.0499e-06, 2.2035e-05],
          [5.7590e-05, 2.0243e-05, 7.4559e-06, 1.3605e-05, 1.9523e-05]],

         [[1.1308e-04, 1.1984e-04, 7.8531e-05, 5.1781e-05, 5.1414e-05],
          [7.4645e-04, 1.4198e-03, 1.7676e-03, 1.4117e-03, 7.3858e-04],
          [1.4106e-03, 2.0955e-03, 1.7460e-03, 9.0837e-04, 5.4190e-04],
          [5.8726e-04, 4.3681e-04, 7.3170e-05, 1.4827e-04, 5.8123e-04],
          [2.7106e-04, 1.1081e-04, 2.0660e-04, 5.8238e-04, 5.7368e-04]],

         …,

         [[7.4443e-04, 8.7818e-04, 9.9262e-04, 1.0905e-03, 1.0649e-03],
          [2.0823e-04, 1.1640e-04, 4.3674e-05, 2.6611e-04, 4.3411e-04],
          [1.3155e-05, 2.7887e-05, 4.3481e-05, 4.7969e-05, 5.4570e-05],
          [6.7756e-05, 8.9461e-05, 3.9896e-05, 1.8222e-05, 3.7753e-05],
          [3.6569e-04, 3.2845e-04, 7.2843e-05, 1.0894e-05, 3.6591e-05]],

         [[7.2231e-05, 7.6907e-05, 3.4261e-05, 3.1426e-05, 4.8434e-05],
          [8.4581e-05, 2.5645e-05, 3.0376e-05, 1.3756e-04, 1.0288e-04],
          [2.3866e-05, 1.5484e-05, 2.8347e-04, 3.0068e-04, 7.5281e-05],
          [1.0372e-05, 1.0184e-04, 3.6447e-04, 1.2130e-04, 5.2902e-05],
          [3.0753e-05, 1.9697e-04, 1.8216e-04, 7.1281e-05, 5.2134e-05]],

         [[3.1344e-04, 2.0243e-04, 1.2940e-04, 1.5356e-04, 1.2315e-04],
          [2.9262e-04, 2.3023e-04, 9.0696e-05, 1.4505e-04, 2.5385e-04],
          [4.3447e-04, 2.7853e-04, 2.7550e-05, 1.8902e-04, 4.6811e-04],
          [6.0294e-04, 1.2463e-04, 1.3042e-05, 3.4223e-04, 5.6773e-04],
          [4.1449e-04, 4.9492e-05, 9.6569e-05, 5.1926e-04, 3.7777e-04]]]],
       device='cuda:0')}, 3: {'step': tensor(45000.), 'exp_avg': tensor([
5.1049e-03, -2.9336e-03, -5.6260e-03, -6.5375e-03,  9.5344e-04,
         6.5660e-03,  5.6052e-45,  5.4204e-04, -2.5407e-03, -1.4006e-03,
         2.0424e-03,  1.0069e-03, -5.5762e-03,  1.8487e-03, -1.8557e-11,
         3.8305e-03,  8.2418e-04, -4.3304e-03, -1.7404e-03,  6.0184e-03],
       device='cuda:0'), 'exp_avg_sq': tensor([1.4178e-03, 1.6717e-03,
1.3972e-03, 2.4062e-03, 1.6855e-03, 1.6202e-03,
        1.9738e-08, 1.6048e-04, 1.6432e-03, 2.2271e-03, 8.0161e-04, 1.9635e-03,
        2.2933e-03, 6.1936e-04, 2.3654e-05, 2.0373e-03, 1.4698e-03, 1.6672e-03,
        1.3814e-03, 1.1043e-03], device='cuda:0')}, 4: {'step': tensor(45000.),
```

```
'exp_avg': tensor([[ 2.8022e-04,  4.6071e-07, -1.1372e-06,  …, -2.3156e-03,
         -7.1511e-04, -4.2806e-04],
        [-4.6343e-04,  2.8650e-08, -4.0476e-07,  …, -6.0255e-05,
         -1.6029e-05, -1.0877e-05],
        [-1.3854e-07, -4.7608e-07, -9.5964e-08,  …, -1.2143e-06,
          5.9557e-06, -1.9607e-06],
        …,
        [-3.4395e-07, -4.3099e-06, -2.2860e-06,  …, -8.6318e-06,
          2.0630e-06, -8.9751e-07],
        [ 5.6052e-45,  5.6052e-45,  5.6052e-45,  …,  5.6052e-45,
          5.6052e-45,  5.6052e-45],
        [-9.7920e-09, -2.7151e-06,  1.7650e-09,  …,  2.5294e-06,
         -2.9787e-18, -1.8390e-06]], device='cuda:0'), 'exp_avg_sq':
tensor([[1.1741e-06, 1.3681e-06, 2.8737e-06,  …, 2.0435e-05, 6.0407e-05,
         2.4876e-04],
        [2.4671e-05, 4.5500e-06, 6.4869e-06,  …, 2.4452e-05, 1.7182e-05,
         7.2806e-06],
        [3.2690e-05, 8.8362e-06, 1.0838e-05,  …, 3.4093e-05, 9.2987e-05,
         1.5537e-04],
        …,
        [1.3231e-05, 1.8558e-05, 4.1844e-06,  …, 1.1022e-05, 1.9767e-05,
         1.7509e-05],
        [4.0108e-25, 1.2063e-24, 2.0748e-27,  …, 1.7433e-25, 1.3980e-24,
         1.5729e-24],
        [1.2122e-05, 2.5279e-05, 2.7497e-06,  …, 2.8581e-06, 1.2094e-06,
         1.5634e-06]], device='cuda:0')}, 5: {'step': tensor(45000.), 'exp_avg':
tensor([-2.2536e-04, -1.3790e-03, -1.5602e-05,  5.6052e-45, -3.7556e-05,
         5.6052e-45, -3.0483e-03,  9.5882e-04,  5.6052e-45, -4.4780e-05,
         2.8714e-05, -4.7969e-04, -1.0583e-03, -2.5178e-03, -9.5552e-06,
         2.7012e-04, -1.2790e-05, -6.6241e-05, -7.8423e-05,  1.9696e-03,
         6.4896e-05, -2.7087e-04,  5.6052e-45,  1.3309e-04,  2.0354e-07,
         2.2838e-04,  1.0058e-03, -7.4925e-06, -5.3890e-05,  3.1293e-04,
         1.2006e-03, -1.8889e-03, -4.5166e-05, -8.1923e-04, -1.2460e-03,
         3.0779e-05, -1.7461e-05, -2.2141e-04, -6.0751e-06, -6.6489e-05,
         1.6122e-04, -2.2096e-03, -1.0604e-04,  1.9956e-03,  7.5480e-04,
        -4.0445e-05,  4.1782e-04,  3.4841e-04,  2.3466e-03, -3.2859e-05,
        -3.9997e-05, -6.3057e-05, -6.7404e-05,  1.8942e-03,  1.3757e-04,
        -1.4734e-07, -2.1389e-05,  3.8204e-05,  1.5882e-05, -2.5683e-04,
         1.3007e-03, -3.8829e-05,  5.6052e-45,  7.7863e-06], device='cuda:0'),
'exp_avg_sq': tensor([4.7674e-05, 5.4261e-05, 5.6960e-05, 4.8592e-18,
2.7991e-05, 1.7133e-25,
        1.2059e-04, 6.0181e-05, 2.8227e-25, 7.5018e-05, 3.9335e-05, 9.4358e-05,
        9.8390e-05, 4.5878e-05, 2.8385e-05, 1.8754e-05, 2.7791e-05, 6.4086e-05,
        3.4489e-05, 1.0970e-04, 6.7124e-05, 1.2272e-04, 6.1572e-26, 4.9661e-05,
        1.5014e-05, 3.5807e-05, 6.4793e-05, 1.3936e-04, 2.5428e-05, 7.1226e-05,
        5.0731e-05, 9.9979e-05, 4.7667e-05, 1.0748e-04, 2.5846e-05, 8.8450e-05,
        5.3286e-05, 9.7093e-05, 1.1730e-05, 9.4811e-05, 6.5762e-05, 5.2829e-05,
        8.9864e-05, 1.7290e-05, 5.0288e-05, 4.5609e-05, 6.5581e-05, 7.2571e-05,
```

```
        1.2522e-04, 1.0437e-04, 4.1216e-05, 3.4269e-05, 4.1112e-05, 1.3641e-04,
        6.1594e-05, 2.0187e-05, 6.2627e-05, 5.5699e-05, 3.0053e-05, 6.0938e-05,
        7.3309e-05, 4.5360e-05, 1.8634e-25, 2.1421e-05], device='cuda:0')}, 6:
{'step': tensor(45000.), 'exp_avg': tensor([[ 1.0522e-04,  2.5256e-04,
-2.7722e-05,  5.6052e-45,  1.6626e-04,
          5.6052e-45,  1.3859e-04, -1.2506e-05,  5.6052e-45,  3.5687e-05,
         -7.6245e-06,  1.6151e-04,  3.3817e-06,  3.1963e-05,  1.4932e-05,
          9.7418e-07,  2.3952e-05,  4.2414e-05,  2.4849e-07,  1.2976e-05,
          1.9600e-04,  4.2053e-05,  5.6052e-45,  1.5833e-05,  1.5135e-04,
          2.8089e-04,  1.7719e-04,  8.0959e-06, -5.3621e-05, -3.1597e-06,
         -5.1645e-05,  7.9164e-05,  8.6361e-07,  1.3877e-06,  6.3698e-08,
          1.0521e-04,  3.2903e-05, -6.1572e-06, -2.8885e-05,  1.3836e-04,
          1.2849e-05, -8.1320e-06, -7.5945e-05,  1.8052e-04,  4.0628e-06,
          1.0633e-05,  1.7828e-06, -1.8350e-05, -2.3023e-06,  7.0824e-06,
          7.0334e-06, -3.4390e-05,  3.1234e-05,  8.8918e-05,  2.8372e-05,
          1.3172e-04,  6.3280e-07,  1.6846e-05,  1.3666e-05,  6.9338e-06,
          7.1985e-06,  4.1798e-06,  5.6052e-45,  5.9742e-06],
        [-1.4838e-04,  1.0262e-05,  1.9559e-06,  5.6052e-45,  6.3025e-09,
          5.6052e-45,  6.8163e-06,  3.0209e-05,  5.6052e-45, -1.3392e-05,
         -1.5518e-04,  1.4241e-04, -2.3369e-05,  2.2216e-05, -6.0479e-05,
          8.3708e-07,  2.6025e-05, -7.9169e-05,  2.2575e-06, -9.5687e-05,
         -2.0763e-05, -1.9777e-04,  5.6052e-45,  1.3842e-04,  2.5144e-08,
          1.3906e-05, -3.9537e-05, -2.1004e-05, -1.3778e-04, -4.2670e-06,
          8.2396e-05,  3.4259e-05, -9.1516e-07,  1.0978e-05,  7.0828e-06,
         -3.1670e-05,  1.2084e-06, -8.3898e-05, -1.6782e-05,  2.1629e-05,
         -1.1843e-04,  7.8612e-06, -3.9412e-04,  1.3074e-05, -3.1427e-05,
         -1.2923e-04,  9.3182e-05, -4.6905e-05,  8.1909e-05, -5.6580e-05,
         -2.9316e-04,  2.1947e-05, -1.4654e-05, -1.9436e-05,  1.3762e-04,
          1.3695e-08, -5.8928e-06, -1.9342e-05,  1.4715e-08,  6.9559e-06,
          3.5357e-05, -1.2520e-04,  5.6052e-45,  2.8345e-09],
        [ 5.5163e-05,  1.9150e-04, -1.1319e-04,  5.6052e-45,  1.5033e-05,
          5.6052e-45,  1.4779e-02,  1.3245e-02, -5.6052e-45, -1.1361e-04,
         -5.2978e-05,  1.8528e-02,  4.3606e-04,  6.4448e-03,  1.9400e-05,
         -4.5204e-04,  8.0280e-05,  1.0262e-04,  8.0974e-07, -2.0068e-04,
          1.4245e-04, -3.4810e-05,  5.6052e-45,  1.0956e-02,  2.8382e-06,
         -9.9452e-04,  5.6295e-04,  8.9123e-06,  1.2374e-04, -3.1178e-04,
          1.3154e-03,  2.1658e-03,  4.4907e-06,  1.0523e-02,  2.3999e-04,
          5.1649e-05,  2.9782e-05, -4.6906e-03,  2.0767e-05,  8.8282e-05,
         -1.6496e-03,  1.9164e-04,  3.6431e-04,  8.5166e-03,  4.8999e-04,
          3.5188e-05,  1.6739e-02,  1.1139e-04,  6.5248e-03,  8.5725e-03,
          8.4521e-05,  1.3550e-02,  2.6280e-05,  7.5436e-03,  1.0836e-02,
          2.4939e-06,  7.4509e-06,  1.2085e-05, -1.2444e-07,  1.0590e-05,
          4.3605e-03,  6.8605e-05,  5.6052e-45,  4.6363e-06],
        [ 4.8863e-05,  1.8464e-03,  4.1196e-05,  5.6052e-45, -1.4525e-04,
          5.6052e-45,  2.7713e-04,  4.1981e-05,  5.6052e-45,  5.4324e-05,
          6.0325e-07,  1.7149e-03,  4.2469e-04,  1.5581e-04,  2.9400e-06,
          7.6182e-04,  3.5469e-04,  2.5702e-06,  3.1333e-04,  3.3755e-04,
         -1.4517e-04,  1.4716e-03,  5.6052e-45,  1.4396e-04, -2.7507e-05,
```

```
 -8.1133e-06,  3.6515e-04,  2.8350e-03,  1.1293e-05,  5.4642e-05,
  2.4088e-06,  3.1793e-03,  2.2360e-08,  2.8940e-04,  1.5586e-03,
  3.9276e-03,  1.3196e-04,  4.0012e-05,  8.7473e-07,  2.4807e-03,
  3.9412e-05,  3.1694e-03,  1.5942e-05,  1.5136e-04,  3.8898e-04,
  1.8156e-06,  3.7407e-03,  2.4060e-04,  3.7036e-04,  1.8187e-04,
  1.3445e-06,  3.5214e-05,  1.0519e-06,  1.8688e-03,  1.3205e-04,
 -7.3871e-05,  2.8835e-04,  3.4253e-04,  6.4718e-07,  9.6211e-05,
  1.8909e-03,  4.2639e-06,  5.6052e-45, -1.2349e-05],
[ 6.7924e-05,  5.6717e-04,  4.9313e-07,  5.6052e-45,  3.0422e-06,
  5.6052e-45,  1.4204e-03, -1.8866e-06,  5.6052e-45,  1.5623e-07,
 -3.3420e-04,  1.7881e-03,  1.5550e-03,  8.8337e-05,  4.8621e-06,
  6.6337e-07,  1.9305e-05,  1.5884e-05, -3.7029e-05,  2.2219e-04,
  1.2205e-06, -2.9440e-04,  5.6052e-45,  1.9917e-05,  5.1942e-09,
  4.3603e-05,  6.6428e-05,  1.0015e-04, -1.0370e-04, -1.0361e-05,
  1.0598e-04,  6.7429e-05, -6.1899e-04,  7.0076e-04, -2.0990e-04,
  1.5645e-05, -1.4005e-06,  8.8221e-05,  3.6327e-06,  1.0457e-03,
 -1.0416e-04,  1.2305e-04,  1.0580e-04,  2.3030e-05,  1.3179e-04,
 -7.7730e-05,  3.8009e-07,  2.9546e-05, -1.4568e-05,  2.0788e-05,
 -2.8477e-04, -2.2246e-04, -1.2301e-04,  5.4007e-05,  2.8804e-04,
 -2.6732e-05,  7.9441e-06,  2.1341e-06,  9.7864e-07,  1.7074e-04,
  2.0547e-03, -1.2954e-04,  5.6052e-45,  1.0171e-07],
[ 1.2141e-05,  2.8373e-04,  1.0378e-05,  5.6052e-45,  2.4552e-04,
  5.6052e-45,  8.9058e-04,  5.6147e-06,  5.6052e-45,  3.0498e-05,
  6.0005e-06,  2.3404e-03,  3.5636e-03,  1.0643e-03,  6.7592e-07,
 -7.2699e-04,  7.6405e-04,  2.5303e-04,  2.4666e-03,  7.0553e-04,
  5.1404e-04, -1.1098e-03,  5.6052e-45, -1.1036e-05,  3.8176e-07,
  2.0285e-06,  2.9999e-04, -2.1125e-03,  7.0943e-06,  1.9146e-04,
  9.7342e-05, -9.8183e-04,  2.0958e-06,  1.1634e-03, -4.2995e-04,
 -2.6697e-03,  1.6239e-04,  1.4026e-05,  1.6916e-06, -1.0032e-03,
  1.1817e-05, -1.7734e-03,  1.7285e-05,  1.8614e-05,  1.9149e-03,
  4.6530e-06, -3.6581e-03,  4.6205e-04,  3.2108e-05,  4.0874e-06,
  9.5755e-06,  7.8125e-06,  1.6565e-05, -1.9594e-03,  7.9320e-05,
  1.0372e-06, -1.0071e-04, -3.1829e-05,  2.5526e-05,  1.0616e-04,
  1.4785e-03,  4.4789e-05,  5.6052e-45,  1.1136e-07],
[ 1.2229e-06, -1.1104e-04,  2.8508e-06,  5.6052e-45, -1.1742e-06,
  5.6052e-45,  1.8161e-06, -4.6347e-06,  5.6052e-45, -9.6569e-05,
  4.1100e-06,  3.3614e-06,  5.8854e-06,  1.0041e-08, -2.8631e-05,
  1.4387e-06, -8.7350e-05, -3.9154e-04, -1.2256e-05, -1.9089e-06,
 -2.0888e-04, -3.8913e-04,  5.6052e-45,  2.9392e-06,  8.5004e-10,
  5.2632e-06, -1.4411e-04, -3.8964e-04,  2.5526e-06,  4.1760e-07,
 -8.1809e-05,  3.8148e-06,  1.7715e-07,  6.6296e-07,  4.4020e-08,
 -6.9344e-04, -3.2848e-04, -1.7583e-05,  2.3030e-06, -1.1147e-04,
 -9.4754e-06, -9.8964e-05,  1.4051e-05,  5.1756e-06, -2.7045e-05,
 -1.8167e-05, -3.3920e-05,  4.1657e-07, -2.4124e-04,  5.7931e-06,
  5.0967e-06, -3.0453e-06, -7.6850e-05,  2.0448e-06,  2.1005e-06,
  2.3343e-09,  4.2846e-06, -3.4109e-04,  1.8289e-08, -4.1501e-05,
 -2.1821e-04, -5.7336e-05,  5.6052e-45,  3.6952e-07],
[ 1.8821e-04,  2.1685e-03,  5.8981e-05,  5.6052e-45, -5.5123e-05,
```

```
          -5.6052e-45, -1.2604e-02, -1.3310e-02,  5.6052e-45,  4.7032e-05,
           1.4891e-04, -1.5244e-02,  1.6956e-03, -5.8417e-03,  3.2991e-05,
           4.0569e-04,  1.5291e-04,  2.0019e-05,  2.6737e-05,  2.2565e-03,
          -1.3453e-04,  1.8696e-04,  5.6052e-45, -1.0032e-02, -1.2809e-04,
           7.0143e-04,  1.5724e-03,  2.6123e-05,  1.9865e-04,  3.6370e-04,
          -1.4088e-03, -5.3512e-06,  7.3143e-06, -8.7790e-03,  1.1064e-03,
          -1.6345e-05,  5.1442e-07,  4.7359e-03,  2.8194e-06,  8.4375e-05,
           1.6941e-03,  4.5992e-04,  3.5535e-05, -8.1731e-03,  1.9503e-03,
           5.5390e-05, -1.6782e-02,  1.5267e-03, -5.2906e-03, -8.8669e-03,
           8.3713e-05, -1.3596e-02,  1.2166e-05, -6.9169e-03, -1.1018e-02,
          -6.2316e-05,  1.6001e-06,  1.1531e-05, -1.3328e-05,  5.9122e-05,
          -2.4873e-03, -1.4953e-05,  5.6052e-45,  7.5671e-08],
         [ 2.1917e-03,  1.9232e-02,  6.4837e-06, -5.6052e-45,  1.8905e-06,
           5.6052e-45,  1.7412e-02,  6.1981e-06,  5.6052e-45,  5.5852e-05,
           1.9625e-04,  2.3506e-02,  1.2708e-02,  5.8390e-03,  1.2935e-05,
           2.1586e-05, -4.0655e-04,  3.3295e-05,  4.6529e-05,  1.3940e-02,
           6.3176e-05,  1.8921e-04, -5.6052e-45,  6.7088e-03,  7.2505e-08,
          -2.6158e-04,  1.6525e-02,  8.2085e-05, -7.4942e-05, -1.2386e-04,
           4.4628e-05,  1.5697e-02,  1.5298e-04,  1.3084e-02,  9.9520e-03,
          -1.9867e-04,  7.4630e-06, -8.0179e-05,  1.2676e-05,  9.4751e-04,
           1.1408e-04,  2.9508e-03, -8.7486e-05,  4.8203e-03,  1.4950e-02,
           4.2823e-05, -1.4777e-04,  1.3842e-02,  8.2439e-03,  1.1122e-04,
           1.5536e-04,  2.1532e-04,  1.0170e-04,  4.4513e-03,  4.4664e-04,
           2.6780e-05,  1.6839e-05,  6.9454e-06, -5.6739e-07, -5.9148e-05,
           1.0125e-02,  1.9705e-04, -5.6052e-45,  2.3292e-07],
         [-2.5221e-03, -2.4441e-02,  1.8581e-05,  5.6052e-45, -2.3019e-04,
           5.6052e-45, -2.2322e-02,  5.3162e-07,  5.6052e-45,  5.9340e-08,
           1.9412e-04, -3.2940e-02, -2.0368e-02, -7.8048e-03,  3.7988e-07,
          -1.3967e-05, -9.2727e-04,  8.9267e-07, -2.8073e-03, -1.7176e-02,
          -4.0750e-04,  1.3611e-04,  5.6052e-45, -7.9432e-03,  9.2083e-07,
           2.1710e-04, -1.9386e-02, -5.3715e-04,  2.6750e-05, -1.5679e-04,
          -1.0588e-04, -2.0239e-02,  4.5197e-04, -1.6994e-02, -1.2224e-02,
          -4.9028e-04, -3.6328e-05,  3.8682e-07,  9.0369e-07, -3.6918e-03,
           9.5259e-06, -5.0221e-03,  4.6709e-06, -5.5556e-03, -1.9771e-02,
           7.4638e-05,  4.7174e-05, -1.6147e-02, -9.7044e-03,  2.0142e-05,
           2.3131e-04,  2.5219e-05,  2.5527e-05, -5.1129e-03, -9.3239e-04,
           8.7797e-07, -2.2050e-04,  2.0457e-07, -2.6832e-05, -3.5607e-04,
          -1.7246e-02,  8.1774e-06,  5.6052e-45,  8.5321e-07]], device='cuda:0'),
'exp_avg_sq': tensor([[2.0813e-05, 2.3474e-03, 1.5221e-03, 1.8028e-21,
2.6769e-03, 1.3534e-26,
          1.4443e-05, 9.4945e-05, 1.9747e-31, 6.1638e-04, 3.9017e-04, 1.5174e-04,
          2.5061e-05, 1.6902e-05, 1.9656e-04, 2.7956e-04, 6.7783e-04, 1.6054e-03,
          3.6145e-04, 3.8359e-04, 2.4133e-03, 5.4339e-04, 5.2555e-28, 3.3103e-04,
          8.1158e-05, 1.0394e-03, 5.8872e-03, 2.0707e-03, 1.2306e-03, 9.6019e-04,
          4.6197e-03, 4.3115e-05, 8.7316e-07, 3.2477e-04, 3.0582e-06, 8.2823e-03,
          2.6414e-04, 2.1780e-03, 4.1963e-04, 3.3924e-04, 5.5061e-04, 7.0441e-04,
          2.2953e-03, 5.5320e-05, 1.2335e-03, 2.7435e-04, 4.3087e-04, 1.5312e-03,
          1.6600e-03, 1.0468e-04, 1.3185e-04, 1.9372e-04, 5.6951e-05, 1.5094e-04,
```

```
 4.0529e-04, 1.8781e-05, 1.0569e-04, 1.2255e-04, 8.5078e-04, 5.8161e-04,
 1.7360e-04, 4.1132e-03, 1.4097e-27, 5.3845e-04],
[4.6916e-04, 2.2326e-05, 1.4254e-04, 6.2990e-19, 2.3271e-08, 9.8277e-31,
 4.6761e-03, 2.6357e-07, 3.7742e-30, 6.1524e-04, 2.0989e-03, 1.4756e-03,
 2.2730e-03, 6.6166e-05, 8.3018e-04, 4.1724e-04, 3.2116e-05, 3.8091e-04,
 6.7137e-06, 1.8613e-03, 6.4276e-04, 2.9538e-03, 8.6247e-27, 4.2934e-05,
 1.5845e-04, 3.5327e-05, 4.2689e-04, 2.3888e-03, 1.4136e-03, 1.8753e-03,
 9.6071e-05, 2.2351e-05, 5.7901e-04, 1.5306e-03, 8.4327e-06, 1.9778e-03,
 1.5590e-04, 3.4635e-03, 1.9382e-04, 1.9879e-05, 3.2612e-03, 1.6375e-04,
 5.7094e-03, 1.1868e-03, 6.5891e-05, 1.2164e-03, 1.8781e-03, 2.3555e-03,
 2.5942e-03, 9.3153e-03, 3.2692e-03, 9.2816e-05, 2.2445e-04, 1.4952e-03,
 7.7095e-04, 7.1156e-06, 1.1765e-03, 5.3660e-04, 1.3795e-05, 6.1540e-05,
 4.8203e-04, 1.5034e-03, 2.7997e-28, 9.3484e-05],
[4.4433e-04, 9.9241e-04, 2.5635e-03, 5.9849e-19, 7.7182e-04, 4.6112e-29,
 2.1558e-03, 8.3635e-04, 4.1382e-24, 4.9750e-04, 1.5690e-04, 6.9060e-03,
 1.8794e-04, 1.2094e-03, 7.0769e-04, 1.0263e-03, 2.2444e-03, 9.2941e-05,
 2.0388e-05, 8.8522e-04, 1.0873e-03, 8.7021e-04, 2.1170e-27, 7.4920e-03,
 4.5326e-05, 8.5458e-04, 2.0729e-03, 2.4597e-04, 5.7050e-04, 5.6808e-04,
 5.0223e-03, 1.8072e-03, 8.7728e-05, 4.6572e-03, 4.7896e-04, 7.6502e-04,
 6.0020e-04, 4.9063e-03, 1.2599e-04, 1.2112e-05, 3.8714e-03, 2.5640e-03,
 1.0271e-03, 1.7996e-03, 3.2079e-03, 7.4992e-04, 6.0725e-03, 3.6554e-04,
 1.5298e-02, 1.8458e-02, 9.9480e-05, 9.6046e-04, 2.6129e-05, 2.9835e-03,
 6.0373e-03, 1.3498e-04, 3.7913e-04, 3.7677e-04, 3.3207e-04, 1.0192e-05,
 5.7369e-04, 1.3077e-04, 1.7847e-24, 7.0651e-06],
[6.2962e-05, 7.6119e-04, 9.9819e-04, 1.1024e-17, 1.7316e-04, 1.8417e-30,
 4.5067e-04, 1.8084e-04, 4.4279e-26, 2.3848e-05, 7.1345e-06, 6.4642e-03,
 3.7602e-03, 5.1254e-04, 4.9993e-04, 6.6941e-04, 8.1524e-04, 2.2511e-05,
 8.5701e-04, 2.1262e-04, 7.2974e-04, 8.9674e-04, 4.5307e-28, 9.1058e-03,
 1.0182e-05, 1.8940e-05, 1.8200e-04, 1.4025e-03, 1.0607e-04, 2.0819e-03,
 2.2189e-03, 7.7324e-03, 4.8829e-05, 1.6378e-04, 2.4191e-03, 4.6523e-03,
 1.6916e-04, 4.8303e-04, 1.3674e-05, 4.0687e-04, 1.4003e-03, 4.8704e-03,
 2.4134e-04, 2.3504e-04, 6.1125e-03, 1.8498e-05, 5.4912e-03, 1.9775e-03,
 1.4752e-02, 3.0775e-03, 2.3786e-08, 1.0686e-03, 2.7284e-06, 9.1138e-03,
 1.0286e-03, 8.6188e-06, 2.5116e-03, 8.6495e-04, 5.1344e-05, 1.4460e-04,
 8.6974e-04, 1.3982e-04, 3.2773e-24, 1.8319e-05],
[7.9439e-05, 3.2718e-04, 3.6622e-05, 1.9463e-20, 6.3957e-04, 9.7537e-31,
 9.0406e-03, 7.9593e-05, 2.4095e-29, 1.6251e-04, 2.1987e-03, 3.6953e-03,
 9.4129e-03, 3.9207e-04, 1.2593e-05, 3.3312e-05, 2.7806e-05, 8.0150e-05,
 3.1635e-03, 1.2458e-03, 2.3028e-04, 2.9866e-03, 3.1334e-28, 1.4994e-04,
 1.5308e-04, 7.0819e-05, 1.1288e-04, 2.4596e-04, 2.4607e-04, 6.2654e-03,
 9.8679e-04, 7.1519e-04, 3.5240e-03, 5.7398e-03, 5.9242e-04, 7.1088e-04,
 4.3338e-04, 2.3855e-04, 4.9600e-04, 5.8415e-04, 9.3208e-04, 1.5701e-04,
 5.1154e-04, 2.3034e-05, 2.5269e-04, 1.2381e-03, 6.2477e-05, 3.3834e-04,
 1.7257e-04, 6.6173e-04, 3.7495e-03, 8.1304e-04, 4.2453e-04, 3.0418e-04,
 1.8533e-03, 4.8992e-04, 1.1544e-04, 4.6165e-05, 7.3401e-05, 6.3287e-03,
 9.0182e-04, 7.4610e-04, 2.6348e-28, 3.7421e-05],
[2.6815e-05, 1.6061e-03, 1.0578e-03, 1.7275e-18, 3.2858e-04, 1.4715e-29,
 3.8949e-04, 4.0889e-05, 1.6163e-31, 5.8906e-04, 1.8790e-05, 4.4301e-03,
```

```
 6.0354e-03, 1.9801e-05, 8.7896e-05, 6.3847e-04, 4.5763e-03, 1.1613e-04,
 3.2974e-03, 2.2867e-03, 2.3852e-03, 5.4985e-03, 4.8671e-28, 1.7911e-03,
 1.6685e-05, 3.5418e-06, 4.3545e-04, 8.9068e-03, 2.5470e-04, 3.7606e-04,
 3.4867e-04, 5.0034e-03, 1.5146e-04, 1.1784e-04, 1.0787e-03, 1.7977e-02,
 3.7263e-03, 9.8039e-05, 1.5998e-04, 2.9993e-03, 7.9383e-04, 8.8404e-04,
 1.8253e-04, 5.7511e-05, 3.3943e-03, 6.3272e-05, 3.4004e-03, 4.9917e-04,
 3.7066e-03, 6.0025e-05, 8.4853e-06, 1.3542e-04, 3.7380e-04, 5.9971e-03,
 6.8971e-04, 1.7307e-05, 5.2208e-03, 7.7806e-04, 2.3883e-04, 5.1236e-04,
 6.8820e-03, 2.4835e-03, 1.6489e-27, 2.9464e-05],
[1.7711e-06, 2.1604e-03, 6.8302e-04, 4.1569e-18, 1.8592e-03, 9.4003e-29,
 1.3749e-05, 1.1863e-06, 6.5719e-31, 1.9707e-03, 5.3129e-04, 1.4700e-04,
 2.1197e-03, 5.0448e-11, 2.2210e-04, 3.9582e-04, 1.2159e-03, 1.3452e-03,
 5.5372e-04, 1.3782e-03, 4.6742e-03, 4.8724e-03, 4.9507e-28, 1.1284e-04,
 4.8777e-05, 1.9884e-04, 3.4357e-03, 8.9739e-03, 1.1834e-03, 4.6241e-04,
 1.7178e-03, 7.0823e-05, 6.2512e-05, 1.6751e-04, 3.8853e-07, 2.1861e-02,
 4.0399e-03, 5.7100e-04, 7.9759e-04, 8.1867e-04, 2.5441e-03, 5.1798e-04,
 9.0031e-04, 1.7121e-05, 2.6906e-03, 1.1390e-03, 1.3899e-04, 2.8495e-03,
 1.6762e-03, 9.1106e-05, 2.7787e-04, 2.9614e-04, 9.8408e-04, 1.8524e-04,
 2.4941e-04, 5.3170e-07, 6.2976e-04, 2.5966e-04, 5.5553e-04, 1.0148e-03,
 3.2228e-03, 7.6691e-03, 8.0145e-28, 4.4585e-04],
[1.3936e-03, 1.8748e-03, 5.5407e-04, 2.1423e-21, 5.7617e-04, 6.8232e-26,
 9.2715e-03, 5.1565e-04, 3.1946e-24, 8.1921e-05, 1.0973e-03, 1.0152e-02,
 9.2442e-04, 2.1430e-03, 1.5025e-04, 4.8515e-04, 3.4439e-04, 3.0225e-06,
 2.6065e-04, 1.5600e-03, 1.2129e-03, 4.6978e-04, 1.2434e-27, 1.6483e-03,
 2.8678e-04, 7.6112e-04, 8.6330e-04, 6.6287e-04, 3.2781e-04, 2.2014e-03,
 6.0247e-04, 1.3082e-03, 4.4007e-04, 7.4623e-03, 8.9184e-05, 1.5185e-03,
 2.9336e-04, 5.0607e-03, 4.8542e-05, 7.2406e-04, 2.5554e-03, 7.0686e-04,
 2.0221e-03, 3.1588e-03, 1.9165e-04, 5.3352e-04, 2.1726e-03, 9.5415e-04,
 4.9106e-03, 1.9380e-02, 3.6019e-04, 5.0998e-04, 1.2879e-06, 4.2880e-03,
 6.2936e-03, 6.2889e-04, 1.2962e-03, 4.4589e-04, 1.4224e-04, 5.4929e-05,
 1.7321e-04, 5.6277e-04, 1.0166e-26, 4.4587e-05],
[9.2526e-04, 1.7124e-03, 9.6319e-04, 7.2476e-17, 1.1266e-04, 1.2081e-30,
 2.6852e-03, 1.8412e-04, 1.0892e-27, 2.3634e-04, 6.4391e-04, 8.0735e-03,
 8.5115e-03, 4.9893e-04, 4.6128e-04, 4.6282e-04, 4.7441e-03, 1.8398e-04,
 9.0405e-04, 7.0345e-03, 7.9264e-04, 3.3172e-03, 1.1943e-26, 2.3416e-03,
 8.2736e-06, 3.1259e-04, 3.0287e-03, 3.2245e-03, 1.6915e-03, 4.2894e-04,
 2.8593e-04, 2.8596e-03, 1.8447e-04, 5.0071e-03, 7.0326e-04, 1.4434e-02,
 9.9438e-04, 8.5833e-04, 4.7030e-05, 5.0346e-04, 1.0897e-03, 1.1566e-03,
 3.0152e-03, 5.7634e-04, 9.8491e-03, 2.9384e-04, 8.8405e-04, 4.8719e-03,
 5.6407e-03, 1.0993e-03, 1.5557e-03, 1.0541e-03, 2.1276e-04, 1.8958e-03,
 9.2181e-04, 1.4041e-04, 4.6070e-03, 4.2626e-05, 1.0045e-04, 4.9583e-04,
 6.6222e-03, 6.9080e-03, 1.1187e-23, 1.3026e-04],
[1.0081e-03, 3.8995e-03, 4.7313e-04, 1.9562e-21, 1.6368e-03, 1.7823e-26,
 1.2156e-02, 1.1120e-05, 1.3487e-30, 8.1214e-09, 9.0237e-04, 1.5182e-02,
 1.7011e-02, 1.6812e-03, 6.5756e-05, 3.1306e-04, 1.1754e-03, 9.7109e-06,
 6.2128e-03, 1.8745e-03, 1.5935e-04, 1.3885e-03, 3.9526e-28, 1.1920e-03,
 9.3629e-05, 7.9304e-04, 1.7222e-03, 9.7327e-04, 9.1139e-05, 9.0934e-03,
 1.5381e-03, 4.4959e-03, 3.0317e-03, 1.1131e-02, 2.2716e-03, 1.9361e-03,
```

```
        5.9173e-05, 2.6131e-04, 1.2549e-04, 1.6713e-03, 8.0633e-04, 3.9167e-03,
        8.3626e-04, 4.4869e-04, 4.5455e-03, 4.3655e-04, 7.1388e-04, 2.2360e-03,
        1.6185e-03, 1.4083e-03, 2.3678e-03, 2.8378e-04, 6.6274e-06, 1.4392e-03,
        1.0348e-03, 1.7156e-04, 1.4773e-03, 9.6358e-05, 4.5472e-04, 6.9777e-03,
        4.6531e-03, 1.8413e-04, 3.7592e-27, 1.2421e-04]], device='cuda:0')}, 7:
{'step': tensor(45000.), 'exp_avg': tensor([ 2.5709e-05, -1.5478e-05,
1.6756e-03,  4.9572e-04,  2.4787e-04,
        1.5968e-04, -6.9184e-05, -1.0391e-03,  4.6189e-03, -6.0996e-03],
      device='cuda:0'), 'exp_avg_sq': tensor([0.0003, 0.0003, 0.0005, 0.0004,
0.0003, 0.0004, 0.0004, 0.0005, 0.0006,
       0.0007], device='cuda:0')}}
param_groups     [{'lr': 0.001, 'betas': (0.9, 0.999), 'eps': 1e-08,
'weight_decay': 0, 'amsgrad': False, 'maximize': False, 'foreach': None,
'capturable': False, 'differentiable': False, 'fused': None, 'params': [0, 1, 2,
3, 4, 5, 6, 7]}]
```

[9]:
```
import os
os.makedirs("./ModelFiles", exist_ok=True)
torch.save(model, "./ModelFiles/MNISTmodel.pt")
```

[10]:
```
torch.save(model.state_dict(), './ModelFiles/MNISTmodel.pth')
```

[11]:
```
mnist_train = datasets.FashionMNIST(root='./data', train=True, download=True,␣
 ↪transform=transforms.ToTensor())
train_loader = DataLoader(mnist_train, batch_size=batch_size, shuffle=False)

mnist_test = datasets.FashionMNIST(root='./data', train=False, download=True,␣
 ↪transform=transforms.ToTensor())
test_loader = DataLoader(mnist_test, batch_size=batch_size, shuffle=False)
```

```
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
images-idx3-ubyte.gz to ./data/FashionMNIST/raw/train-images-idx3-ubyte.gz

100%|                    | 26.4M/26.4M [00:02<00:00, 9.50MB/s]

Extracting ./data/FashionMNIST/raw/train-images-idx3-ubyte.gz to
./data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-central-1.amazonaws.com/train-
labels-idx1-ubyte.gz to ./data/FashionMNIST/raw/train-labels-idx1-ubyte.gz

100%|                     | 29.5k/29.5k [00:00<00:00, 156kB/s]

Extracting ./data/FashionMNIST/raw/train-labels-idx1-ubyte.gz to
./data/FashionMNIST/raw
```

```
Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-images-idx3-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-images-idx3-ubyte.gz to
./data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz

100%|                    | 4.42M/4.42M [00:01<00:00, 2.83MB/s]

Extracting ./data/FashionMNIST/raw/t10k-images-idx3-ubyte.gz to
./data/FashionMNIST/raw

Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz
Downloading http://fashion-mnist.s3-website.eu-
central-1.amazonaws.com/t10k-labels-idx1-ubyte.gz to
./data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz

100%|                    | 5.15k/5.15k [00:00<00:00, 2.55MB/s]

Extracting ./data/FashionMNIST/raw/t10k-labels-idx1-ubyte.gz to
./data/FashionMNIST/raw
```

[12]:
```
model = CNNNetwork()
model = torch.load('./ModelFiles/MNISTmodel.pt')
model.to(device)

optimizer = optim.Adam(model.parameters(), lr=1e-2)
```

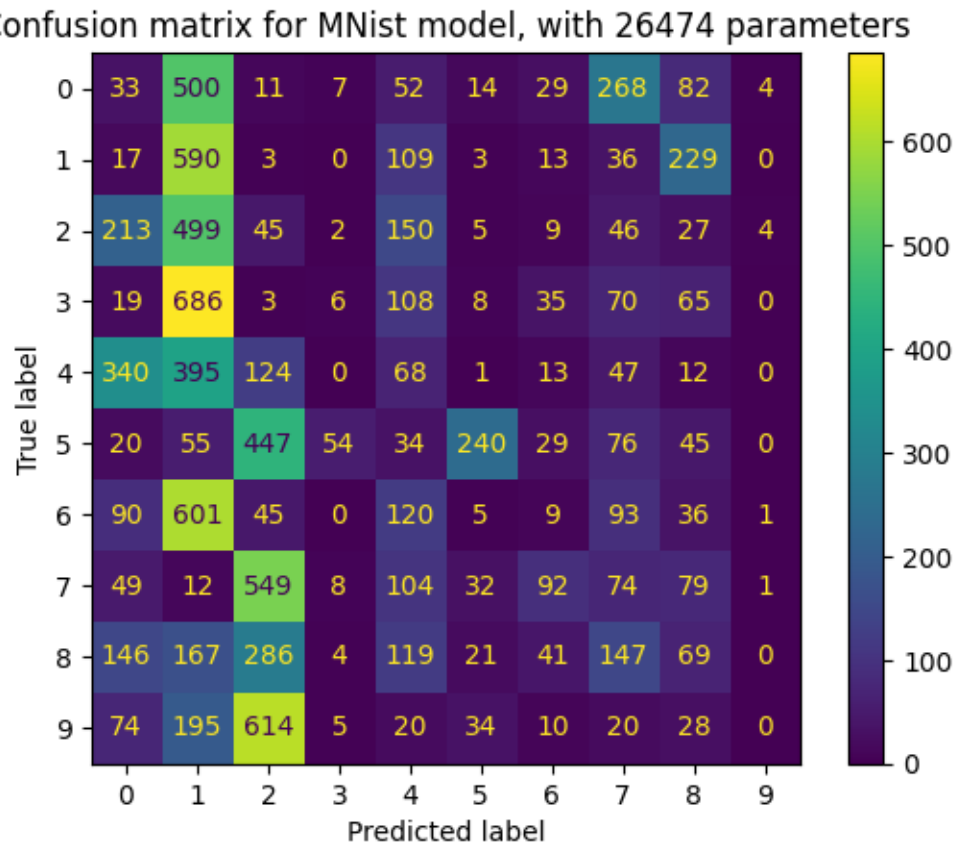
```
/tmp/ipykernel_40797/3878080169.py:2: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  model = torch.load('./ModelFiles/MNISTmodel.pt')
```

[13]:
```
print("Model's state dict:")
for param_tensor in model.state_dict():
    print(param_tensor, "\t", model.state_dict()[param_tensor].size())
```

```
Model's state dict:
```

```
feature_extractor.0.weight      torch.Size([10, 1, 5, 5])
feature_extractor.0.bias        torch.Size([10])
feature_extractor.3.weight      torch.Size([20, 10, 5, 5])
feature_extractor.3.bias        torch.Size([20])
classification_head.0.weight    torch.Size([64, 320])
classification_head.0.bias      torch.Size([64])
classification_head.2.weight    torch.Size([10, 64])
classification_head.2.bias      torch.Size([10])
```

[14]:
```python
model.eval()

losses = []
y_eval, y_preds = [], []


for x_batch, y_batch in test_loader:
    x_batch, y_batch = x_batch.to(device), y_batch.to(device)
    y_pred = model(x_batch)
    y_pred = y_pred.argmax(dim=1)

    y_eval += y_batch.detach().cpu().numpy().tolist()
    y_preds += y_pred.detach().cpu().numpy().tolist()


confusion_matrix = metrics.confusion_matrix(y_eval, y_preds)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =␣
 ↪confusion_matrix, display_labels = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

cm_display.plot()

plt.title(f"Confusion matrix for MNist model, with {total_params} parameters")
plt.show()

acc1 = accuracy_score(y_preds, y_eval)

f"{accuracy_score(y_preds, y_eval):.2%}"
```



[14]:
```
'11.34%'
```

[15]:
```
losses = []
num_epochs = 5
model.train()

for epoch in range(1, num_epochs+1):
    epoch_loss = 0
    for idx, (x_batch, y_batch) in enumerate(train_loader):
        batch_size = len(x_batch)
        loader_size = len(train_loader)

        x_batch, y_batch = x_batch.to(device), y_batch.to(device)
        y_pred = model(x_batch)
        y_pred = y_pred.reshape(batch_size,-1)

        loss = criterion(y_pred, y_batch)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()
```

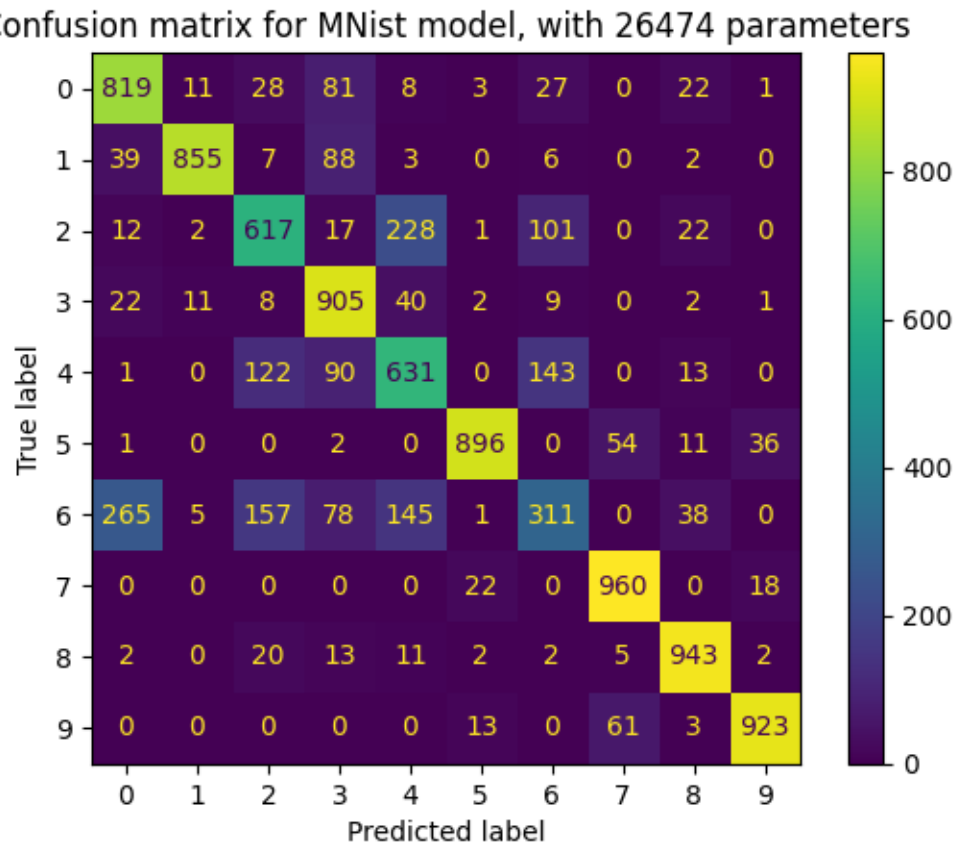
```
        epoch_loss += loss

        if (idx % 1000 == 0):
            progress = (idx / loader_size)
            print(f"{idx * batch_size} / {loader_size * batch_size} : Progress␣
↪= {progress:.2%}")

    if epoch%1 == 0:
        losses.append(epoch_loss.detach().cpu())
        print(f"Epoch {epoch}: Loss = {epoch_loss}")
```

```
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 1: Loss = 9315.986328125
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 2: Loss = 8791.5732421875
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
```

```
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 3: Loss = 8960.255859375
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 4: Loss = 9032.974609375
0 / 60000 : Progress = 0.00%
4000 / 60000 : Progress = 6.67%
8000 / 60000 : Progress = 13.33%
12000 / 60000 : Progress = 20.00%
16000 / 60000 : Progress = 26.67%
20000 / 60000 : Progress = 33.33%
24000 / 60000 : Progress = 40.00%
28000 / 60000 : Progress = 46.67%
32000 / 60000 : Progress = 53.33%
36000 / 60000 : Progress = 60.00%
40000 / 60000 : Progress = 66.67%
44000 / 60000 : Progress = 73.33%
48000 / 60000 : Progress = 80.00%
52000 / 60000 : Progress = 86.67%
56000 / 60000 : Progress = 93.33%
Epoch 5: Loss = 9067.96484375
```

[16]:
```
model.eval()
```

```
losses = []
y_eval, y_preds = [], []


for x_batch, y_batch in test_loader:
    x_batch, y_batch = x_batch.to(device), y_batch.to(device)
    y_pred = model(x_batch)
    y_pred = y_pred.argmax(dim=1)

    y_eval += y_batch.detach().cpu().numpy().tolist()
    y_preds += y_pred.detach().cpu().numpy().tolist()


confusion_matrix = metrics.confusion_matrix(y_eval, y_preds)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =␣
 ↪confusion_matrix, display_labels = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9])

cm_display.plot()

plt.title(f"Confusion matrix for MNist model, with {total_params} parameters")
plt.show()

acc1 = accuracy_score(y_preds, y_eval)

f"{accuracy_score(y_preds, y_eval):.2%}"
```



[16]:
```
'78.60%'
```

[17]:
```
torch.save(model.state_dict(), './ModelFiles/MNISTFashionModel.pth')
```

[18]:
```
import os
from PIL import Image
```

[21]:
```
class imgDataset(Dataset):
    def __init__(self, dir="/home/student/Desktop/220962049_aiml/
 ↪cats_and_dogs_filtered/train/"):
        cats_dir = os.path.join(dir, "cats/")
        dogs_dir = os.path.join(dir, "dogs/")
        self.x = [os.path.join(cats_dir, f) for f in os.listdir(cats_dir) if f.
 ↪endswith(".jpg")] + [os.path.join(dogs_dir, f) for f in os.listdir(dogs_dir)␣
 ↪if f.endswith(".jpg")]
        self.y = [0 for f in os.listdir(cats_dir) if f.endswith(".jpg")] + [1␣
 ↪for f in os.listdir(dogs_dir) if f.endswith(".jpg")]
        self.y = torch.tensor(self.y)
        self.transform = transforms.Compose([transforms.Resize(256),
```

```
                                                transforms.CenterCrop(224),
                                                transforms.ToTensor(),
                                                transforms.Normalize(mean=[0.485,␣
 ↪0.456, 0.406], std=[0.229, 0.224, 0.225]),])

    def __len__(self):
        return len(self.x)

    def __getitem__(self, idx):
        img = Image.open(self.x[idx]).convert('RGB')
        tensor = self.transform(img)
        return tensor.to(device), self.y[idx].to(device)
```

[23]:
```
batch_size = 4
train_set = imgDataset()
val_set = imgDataset(dir="/home/student/Desktop/220962049_aiml/
 ↪cats_and_dogs_filtered/validation/")

train_loader = DataLoader(train_set, batch_size=batch_size, shuffle=True)
val_loader = DataLoader(val_set, batch_size=batch_size, shuffle=False)
```

[26]:
```
len([f for f in os.listdir("/home/student/Desktop/220962049_aiml/
 ↪cats_and_dogs_filtered/train/cats") if f.endswith(".jpg")] + [f for f in os.
 ↪listdir("/home/student/Desktop/220962049_aiml/cats_and_dogs_filtered/train/
 ↪dogs") if f.endswith(".jpg")])
```

[26]:
```
2000
```

[28]:
```
from torchvision.models import AlexNet_Weights

#model = torch.hub.load('pytorch/vision:v0.10.0', model='alexnet',␣
 ↪weights=AlexNet_Weights.DEFAULT)
model = torch.hub.load('pytorch/vision:v0.10.0', 'alexnet', pretrained=True)

model.classifier[6] = torch.nn.Linear(in_features=4096, out_features=2)
model = model.to(device)

loss_fn = torch.nn.CrossEntropyLoss()

optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)
```

```
Downloading: "https://github.com/pytorch/vision/zipball/v0.10.0" to
/home/student/.cache/torch/hub/v0.10.0.zip
/home/student/Desktop/220962049_aiml/.venv/lib/python3.12/site-
packages/torchvision/models/_utils.py:208: UserWarning: The parameter
'pretrained' is deprecated since 0.13 and may be removed in the future, please
use 'weights' instead.
  warnings.warn(
```

```
/home/student/Desktop/220962049_aiml/.venv/lib/python3.12/site-
packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a
weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed
in the future. The current behavior is equivalent to passing
`weights=AlexNet_Weights.IMAGENET1K_V1`. You can also use
`weights=AlexNet_Weights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/alexnet-owt-7be5be79.pth" to
/home/student/.cache/torch/hub/checkpoints/alexnet-owt-7be5be79.pth
100%|                      | 233M/233M [02:46<00:00, 1.47MB/s]
```

[29]:
```
for idx, (x_batch, y_batch) in enumerate(train_loader):
    with torch.no_grad():
        output = model(x_batch)
    # Tensor of shape 1000, with confidence scores over ImageNet's 1000 classes
    print(output[0])
    # The output has unnormalized scores. To get probabilities, you can run a␣
 ↪softmax on it.
    probabilities = torch.nn.functional.softmax(output[0], dim=0)
    print(probabilities)
    break
```

```
tensor([0.0444, 0.2377], device='cuda:0')
tensor([0.4518, 0.5482], device='cuda:0')
```

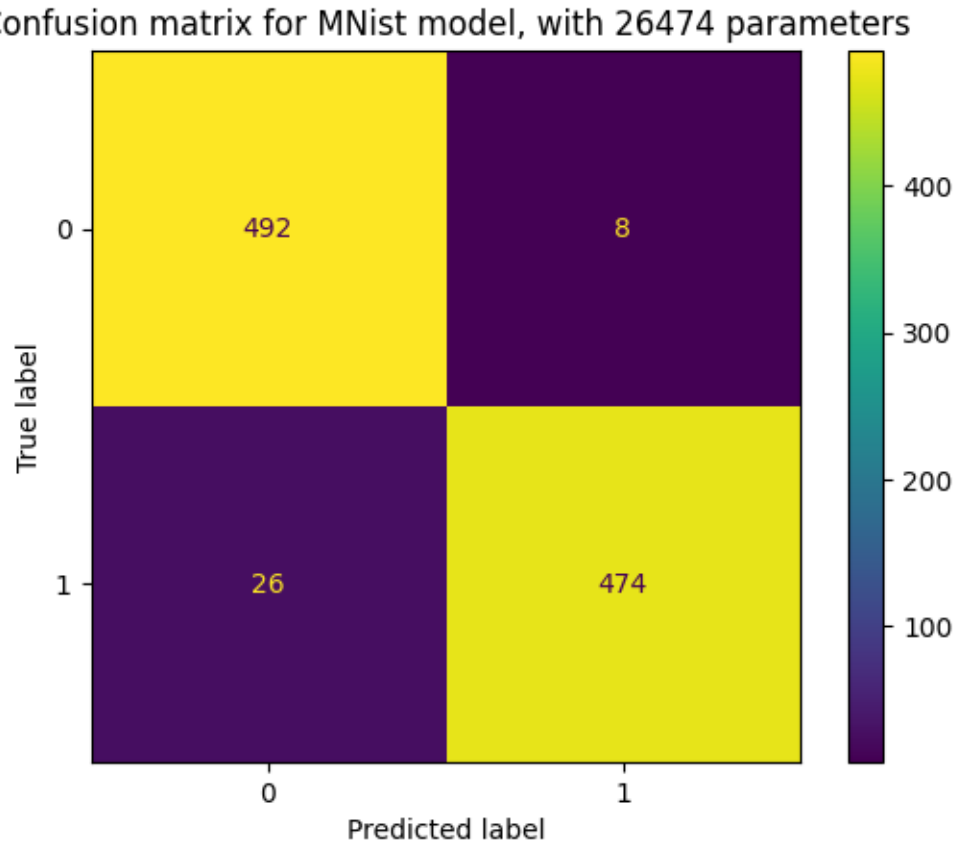
[30]:
```
NUM_EPOCHS = 5
losses = []
model.train()

for epoch in range(1, NUM_EPOCHS+1):
    epoch_loss = 0
    for idx, (x_batch, y_batch) in enumerate(train_loader):
        batch_size = y_batch.shape[0]
        loader_size = len(train_loader)
        output = model(x_batch)
        y_pred = torch.nn.functional.softmax(output, dim=1)

        loss = loss_fn(y_pred, y_batch)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        epoch_loss += loss

        if (idx % 200 == 0):
            progress = (idx / loader_size)
            print(f"{idx * batch_size} / {loader_size * batch_size} : Progress␣
 ↪= {progress:.2%}")
```

```
    if epoch%1 == 0:
        losses.append(epoch_loss.detach().cpu())
        print(f"Epoch {epoch}: Loss = {epoch_loss}")
```

```
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 1: Loss = 194.6973876953125
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 2: Loss = 176.95059204101562
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 3: Loss = 171.05784606933594
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 4: Loss = 169.0035400390625
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 5: Loss = 167.19216918945312
```

[31]:
```
model.eval()

losses = []
y_eval, y_preds = [], []


for x_batch, y_batch in val_loader:
    x_batch, y_batch = x_batch.to(device), y_batch.to(device)
    output = model(x_batch)
    y_pred = F.softmax(output, dim=1)
    y_pred = y_pred.argmax(dim=1)

    y_eval += y_batch.detach().cpu().numpy().tolist()
    y_preds += y_pred.detach().cpu().numpy().tolist()


confusion_matrix = metrics.confusion_matrix(y_eval, y_preds)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =␣
 ↪confusion_matrix, display_labels = [0, 1])
```

```
cm_display.plot()

plt.title(f"Confusion matrix for MNist model, with {total_params} parameters")
plt.show()

acc1 = accuracy_score(y_preds, y_eval)

f"{accuracy_score(y_preds, y_eval):.2%}"
```



[31]: '96.60%'

[32]:
```
torch.save({'epoch': NUM_EPOCHS,
            'model_state_dict': model.state_dict(),
            'optimizer_state_dict': optimizer.state_dict(),
            'loss': epoch_loss},
           './ModelFiles/CustomAlex_1.pth')
```

[33]:
```
CHECKPOINT_PATH = './ModelFiles/CustomAlex_1.pth'

model = torch.hub.load('pytorch/vision:v0.10.0', 'alexnet', pretrained=False)
```

```
model.classifier[6] = torch.nn.Linear(in_features=4096, out_features=2)

optimizer = optim.SGD(model.parameters(), lr=1e-3)


checkpoint = torch.load(CHECKPOINT_PATH)
model.load_state_dict(checkpoint['model_state_dict'])
optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
epoch = checkpoint['epoch']
loss = checkpoint['loss']

model.to(device)
```

```
Using cache found in /home/student/.cache/torch/hub/pytorch_vision_v0.10.0
/home/student/Desktop/220962049_aiml/.venv/lib/python3.12/site-
packages/torchvision/models/_utils.py:208: UserWarning: The parameter
'pretrained' is deprecated since 0.13 and may be removed in the future, please
use 'weights' instead.
  warnings.warn(
/home/student/Desktop/220962049_aiml/.venv/lib/python3.12/site-
packages/torchvision/models/_utils.py:223: UserWarning: Arguments other than a
weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed
in the future. The current behavior is equivalent to passing `weights=None`.
  warnings.warn(msg)
/tmp/ipykernel_40797/3809406884.py:9: FutureWarning: You are using `torch.load`
with `weights_only=False` (the current default value), which uses the default
pickle module implicitly. It is possible to construct malicious pickle data
which will execute arbitrary code during unpickling (See
https://github.com/pytorch/pytorch/blob/main/SECURITY.md#untrusted-models for
more details). In a future release, the default value for `weights_only` will be
flipped to `True`. This limits the functions that could be executed during
unpickling. Arbitrary objects will no longer be allowed to be loaded via this
mode unless they are explicitly allowlisted by the user via
`torch.serialization.add_safe_globals`. We recommend you start setting
`weights_only=True` for any use case where you don't have full control of the
loaded file. Please open an issue on GitHub for any issues related to this
experimental feature.
  checkpoint = torch.load(CHECKPOINT_PATH)
```

[33]:
```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
```

```
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=4096, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=4096, out_features=2, bias=True)
  )
)
```

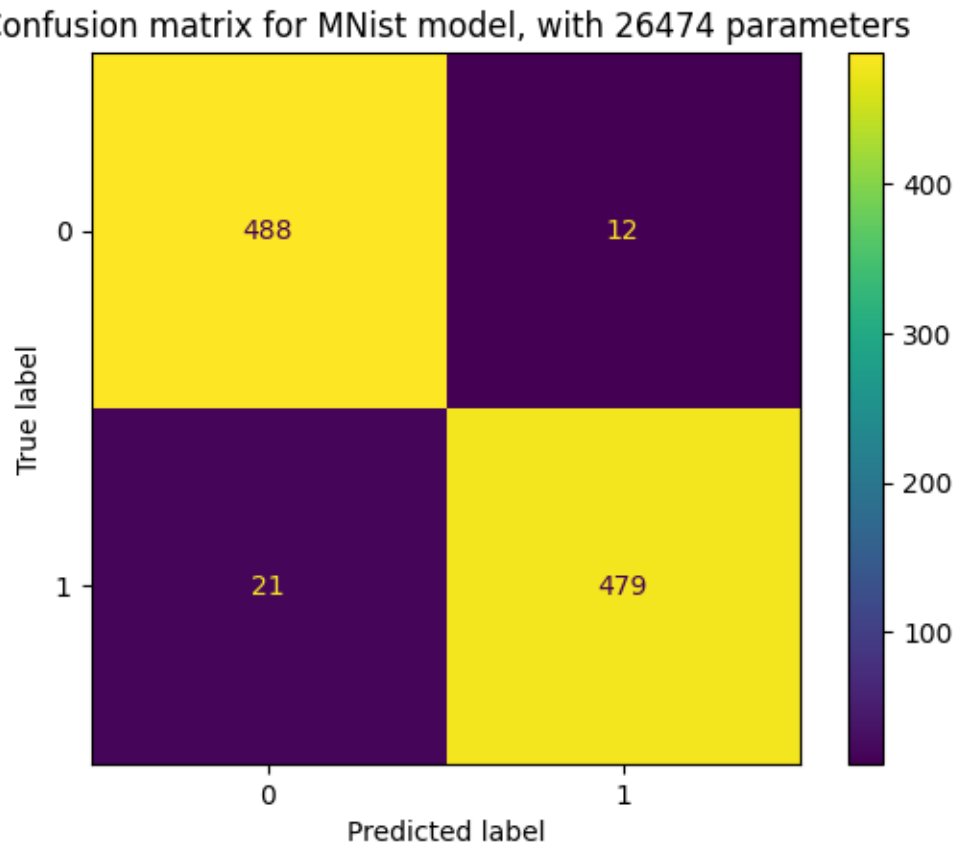
[34]:
```
NUM_EPOCHS = 15
losses = []
model.train()

for epoch in range(epoch+1, epoch+NUM_EPOCHS+1):
    epoch_loss = 0
    for idx, (x_batch, y_batch) in enumerate(train_loader):
        batch_size = y_batch.shape[0]
        loader_size = len(train_loader)
        output = model(x_batch)
        y_pred = torch.nn.functional.softmax(output, dim=1)

        loss = loss_fn(y_pred, y_batch)
        loss.backward()
        optimizer.step()
        optimizer.zero_grad()

        epoch_loss += loss

        if (idx % 200 == 0):
            progress = (idx / loader_size)
            print(f"{idx * batch_size} / {loader_size * batch_size} : Progress␣
 ↪= {progress:.2%}")
```

```
    if epoch%1 == 0:
        losses.append(epoch_loss.detach().cpu())
        print(f"Epoch {epoch}: Loss = {epoch_loss}")
```

```
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 6: Loss = 164.65228271484375
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 7: Loss = 164.41830444335938
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 8: Loss = 161.91432189941406
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 9: Loss = 161.81333923339844
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 10: Loss = 161.7922821044922
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 11: Loss = 160.76625061035156
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 12: Loss = 160.68064880371094
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 13: Loss = 159.85540771484375
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 14: Loss = 159.89569091796875
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 15: Loss = 159.346435546875
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 16: Loss = 159.41331481933594
```

```
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 17: Loss = 159.31260681152344
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 18: Loss = 158.78424072265625
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 19: Loss = 158.9048614501953
0 / 2000 : Progress = 0.00%
800 / 2000 : Progress = 40.00%
1600 / 2000 : Progress = 80.00%
Epoch 20: Loss = 158.00303649902344
```

[35]:
```python
model.eval()

losses = []
y_eval, y_preds = [], []


for x_batch, y_batch in val_loader:
    x_batch, y_batch = x_batch.to(device), y_batch.to(device)
    output = model(x_batch)
    y_pred = F.softmax(output, dim=1)
    y_pred = y_pred.argmax(dim=1)

    y_eval += y_batch.detach().cpu().numpy().tolist()
    y_preds += y_pred.detach().cpu().numpy().tolist()


confusion_matrix = metrics.confusion_matrix(y_eval, y_preds)

cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix =␣
 ↪confusion_matrix, display_labels = [0, 1])

cm_display.plot()

plt.title(f"Confusion matrix for MNist model, with {total_params} parameters")
plt.show()

acc1 = accuracy_score(y_preds, y_eval)

f"{accuracy_score(y_preds, y_eval):.2%}"
```



[35]:
```
'96.70%'
```

[36]:
```
torch.save({'epoch': NUM_EPOCHS,
            'model_state_dict': model.state_dict(),
            'optimizer_state_dict': optimizer.state_dict(),
            'loss': epoch_loss},
           './ModelFiles/CustomAlex_2.pth')
```

[ ]:
```

```