Lab_10_aa07190

November 8, 2024

1 CS 316 : Introduction to Deep Learning - Fall 2024

1.1 Lab 10 : Convolutional Neural Networks

1.1.1 Dr. Abdul Samad

2 Instructions

1. Make a copy of this notebook on google colab at start of the lab.
2. Please rename your notebook as *Lab_10_aa12345.ipynb* before starting the lab. Notebooks which do not follow appropriate naming convention will not be graded.
3. You have to submit this lab during the lab timings. You are allowed to submit till 11:59 PM on the day of your lab with a 30% penalty. No submissions will be accepted afterwards.
4. At the end of the lab, download the notebook (ipynb extension file) and upload it on canvas as a file. Submitting link to notebook or any other file will not be accepted.
5. Each task has points assigned to it. Total Lab is of 100 points.
6. Use of for loops is strictly prohibited.
7. For every theoretical question, there is a separate cell given at the end. You have to write your answer there.
8. If you have any questions, please feel free to reach out to the course instructor or RA.

2.1 Task Overview

In this lab we will work on CNNs. This Lab is going to be short. Work through the cells below, running each cell in turn. In various places you will see the words "TODO". Follow the instructions at these places and make predictions about what is going to happen or write code to complete the functions.

Let's start with importing Libraries first

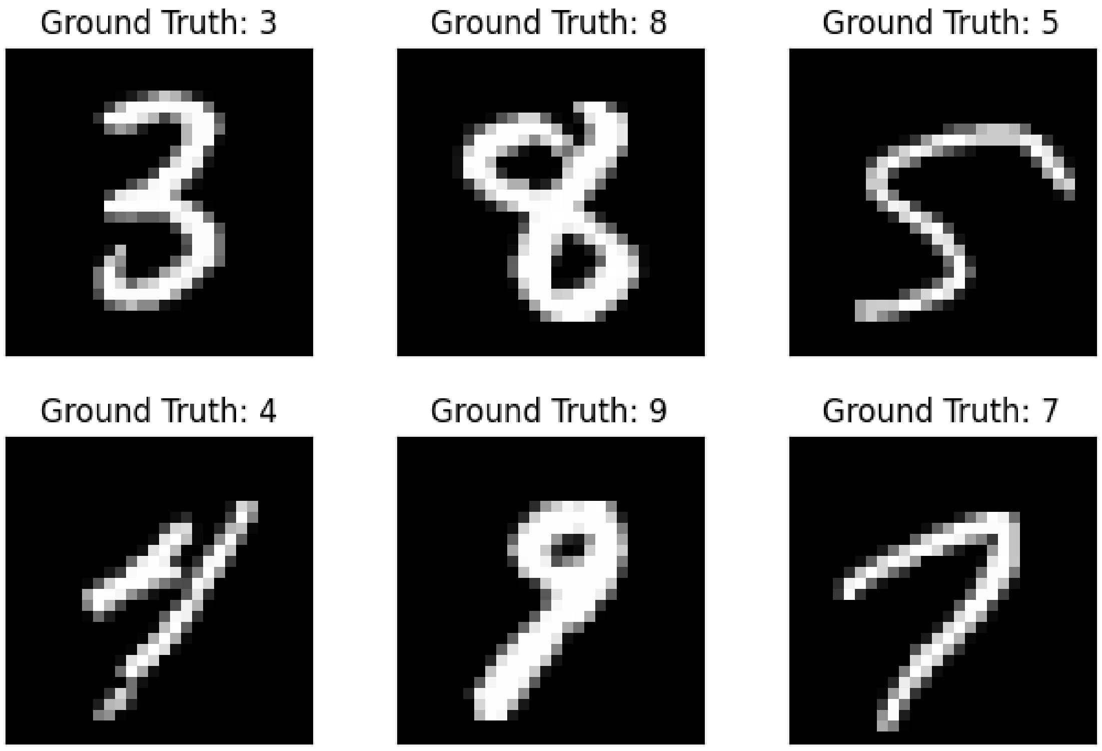
[6]:
```python
import torch
import torch.nn as nn
import torchvision
from torch.autograd import Variable
import torch.nn.functional as F
import torch.optim as optim
import matplotlib.pyplot as plt
import random
```

3 Convolution for MNIST

This notebook builds a proper network for 2D convolution. It works with the MNIST dataset , which was the original classic dataset for classifying images. The network will take a 28x28 grayscale image and classify it into one of 10 classes representing a digit.

[7]:
```
# Run this once to load the train and test data straight into a dataloader class
# that will provide the batches

# (It may complain that some files are missing because the files seem to have␣
↪been
# reorganized on the underlying website, but it still seems to work). If␣
↪everything is working
# properly, then the whole notebook should run to the end without further␣
↪problems
# even before you make changes.
batch_size_train = 64
batch_size_test = 1000
train_loader = torch.utils.data.DataLoader(
  torchvision.datasets.MNIST('./files/', train=True, download=True,
                             transform=torchvision.transforms.Compose([
                               torchvision.transforms.ToTensor(),
                               torchvision.transforms.Normalize(
                                 (0.1307,), (0.3081,))
                             ])),
  batch_size=batch_size_train, shuffle=True)

test_loader = torch.utils.data.DataLoader(
  torchvision.datasets.MNIST('./files/', train=False, download=True,
                             transform=torchvision.transforms.Compose([
                               torchvision.transforms.ToTensor(),
                               torchvision.transforms.Normalize(
                                 (0.1307,), (0.3081,))
                             ])),
  batch_size=batch_size_test, shuffle=True)
```

[8]:
```
# Let's draw some of the training data
examples = enumerate(test_loader)
batch_idx, (example_data, example_targets) = next(examples)

fig = plt.figure()
for i in range(6):
  plt.subplot(2,3,i+1)
  plt.tight_layout()
  plt.imshow(example_data[i][0], cmap='gray', interpolation='none')
  plt.title("Ground Truth: {}".format(example_targets[i]))
  plt.xticks([])
  plt.yticks([])
plt.show()
```



Define the network. This is a more typical way to define a network than the sequential structure. We define a class for the network, and define the parameters in the constructor. Then we use a function called forward to actually run the network. It's easy to see how you might use residual connections in this format.

4 Q1: TODO [30 Points]

Build this network architecture by completing the function given below. 1. A valid convolution with kernel size 5, 1 input channel and 10 output channels 2. A max pooling operation over a 2x2 area 3. A Relu 4. A valid convolution with kernel size 5, 10 input channels and 20 output channels 5. A 2D Dropout layer 6. A max pooling operation over a 2x2 area 7. A relu 8. A flattening operation 9. A fully connected layer mapping from (whatever dimensions we are at– find out using .shape) to 50 10. A ReLU 11. A fully connected layer mapping from 50 to 10 dimensions 12. A softmax function.

[9]:
```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.model=nn.Sequential(

            #nn.MaxPool2d( kernel_size=2, stride=2) This code can be used for␣
 ↪max pooling. Google it to understand it.
            #nn.Dropout() can be used for dropout
            nn.Conv2d(1, 10, kernel_size=5), # 1. image channel, 10 output␣
 ↪channels, 5x5 square convolution kernel
            nn.MaxPool2d(kernel_size=2, stride=2), # 2. max pooling 2x2
            nn.ReLU(), # 3. Relu activation function

            nn.Conv2d(10, 20, kernel_size=5), # 4. 10 input channels, 20 output␣
 ↪channels, 5x5 square convolution kernel
            nn.Dropout2d(), # 5. dropout
            nn.MaxPool2d(kernel_size=2, stride=2), # 6. max pooling 2x2
            nn.ReLU(), # 7. Relu activation function

            nn.Flatten(), # 8. flattening operation

            nn.Linear(320, 50), # 9. 320 input features, 50 output features
            nn.ReLU(), # 10. Relu activation function

            nn.Linear(50, 10), # 11. 50 input features, 10 output features
            nn.Softmax(dim=1) # 12. softmax activation function
        )

    def forward(self, x):
        x = self.model(x)
        return x
```

[10]:
```
#Do NOT EDIT
#USE THIS CODE FOR TESTING
#Since we can't compare the model, its better to look at the its printed␣
 ↪structure and compare it with layers given in architecture
net = Net()
print(net)

# Test the network with a random input of shape (batch_size, channels, height,␣
 ↪width)
x = torch.randn(16, 1, 28, 28)  # Example batch of 16 samples, 1 channel, 28x28␣
 ↪images
```

```python
output = net(x)
print(output.shape)  # Output should be (16, 10)

assert output.shape == (16, 10), "Output shape does not match expected shape"
```

```
Net(
  (model): Sequential(
    (0): Conv2d(1, 10, kernel_size=(5, 5), stride=(1, 1))
    (1): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (2): ReLU()
    (3): Conv2d(10, 20, kernel_size=(5, 5), stride=(1, 1))
    (4): Dropout2d(p=0.5, inplace=False)
    (5): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (6): ReLU()
    (7): Flatten(start_dim=1, end_dim=-1)
    (8): Linear(in_features=320, out_features=50, bias=True)
    (9): ReLU()
    (10): Linear(in_features=50, out_features=10, bias=True)
    (11): Softmax(dim=1)
  )
)
torch.Size([16, 10])
```

[11]:
```python
# Initialization of weights
def weights_init(layer_in):
  if isinstance(layer_in, nn.Linear):
    nn.init.kaiming_uniform_(layer_in.weight)
    layer_in.bias.data.fill_(0.0)
```

[12]:
```python
# Main training routine
def train(epoch):
  model.train()
  # Get each
  for batch_idx, (data, target) in enumerate(train_loader):
    optimizer.zero_grad()
    output = model(data)
    loss = F.nll_loss(output, target)
    loss.backward()
    optimizer.step()
    # Store results
    if batch_idx % 10 == 0:
      print('Train Epoch: {} [{}/{}]\tLoss: {:.6f}'.format(
        epoch, batch_idx * len(data), len(train_loader.dataset), loss.item()))
```

[13]:
```
# Run on test data
def test():
  model.eval()
  test_loss = 0
  correct = 0
  with torch.no_grad():
    for data, target in test_loader:
      output = model(data)
      test_loss += F.nll_loss(output, target, size_average=False).item()
      pred = output.data.max(1, keepdim=True)[1]
      correct += pred.eq(target.data.view_as(pred)).sum()
  test_loss /= len(test_loader.dataset)
  print('\nTest set: Avg. loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
    test_loss, correct, len(test_loader.dataset),
    100. * correct / len(test_loader.dataset)))
```

[15]:
```
# Create network
model = Net()
# Initialize model weights
model.apply(weights_init)
# Define optimizer
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
```

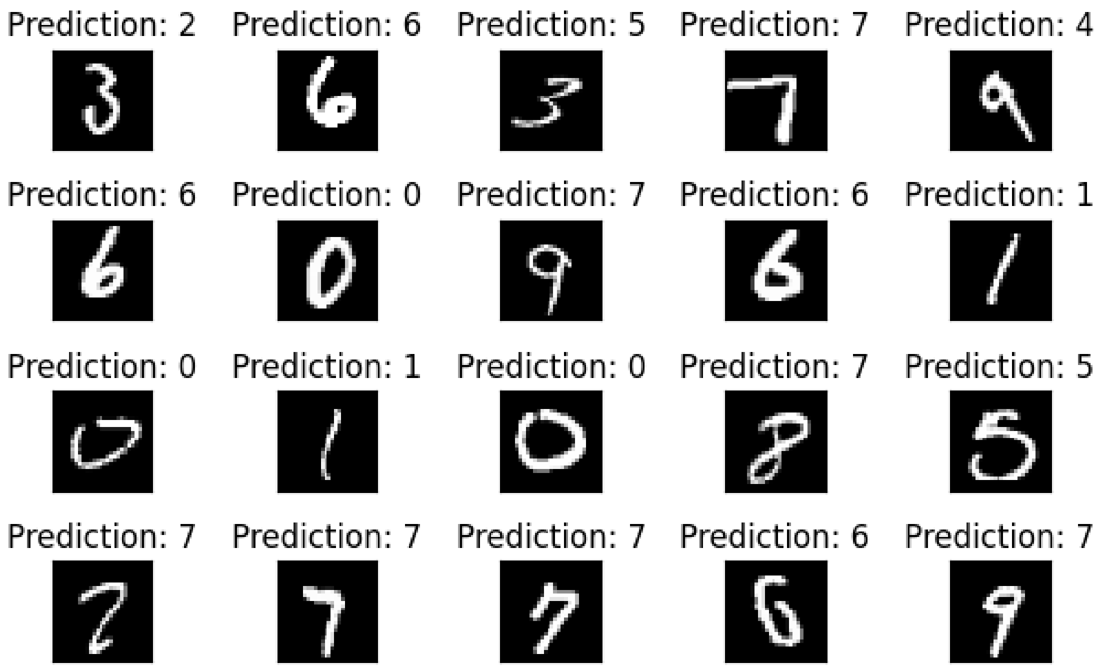
[16]:
```
# Train for three epochs
n_epochs = 3
for epoch in range(1, n_epochs + 1):
  train(epoch)
  test()
```

```
Train Epoch: 1 [0/60000]        Loss: -0.101131
Train Epoch: 1 [640/60000]      Loss: -0.099985
Train Epoch: 1 [1280/60000]     Loss: -0.100898
Train Epoch: 1 [1920/60000]     Loss: -0.118542
Train Epoch: 1 [2560/60000]     Loss: -0.114387
Train Epoch: 1 [3200/60000]     Loss: -0.098014
Train Epoch: 1 [3840/60000]     Loss: -0.131624
Train Epoch: 1 [4480/60000]     Loss: -0.111092
Train Epoch: 1 [5120/60000]     Loss: -0.117117
Train Epoch: 1 [5760/60000]     Loss: -0.138078
Train Epoch: 1 [6400/60000]     Loss: -0.148713
Train Epoch: 1 [7040/60000]     Loss: -0.137987
Train Epoch: 1 [7680/60000]     Loss: -0.199400
Train Epoch: 1 [8320/60000]     Loss: -0.129732
Train Epoch: 1 [8960/60000]     Loss: -0.150108
Train Epoch: 1 [9600/60000]     Loss: -0.226297
Train Epoch: 1 [10240/60000]    Loss: -0.204680
Train Epoch: 1 [10880/60000]    Loss: -0.179540
Train Epoch: 1 [11520/60000]    Loss: -0.212509
Train Epoch: 1 [12160/60000]    Loss: -0.241888
Train Epoch: 1 [12800/60000]    Loss: -0.226496
Train Epoch: 1 [13440/60000]    Loss: -0.254100
Train Epoch: 1 [14080/60000]    Loss: -0.242153
Train Epoch: 1 [14720/60000]    Loss: -0.216317
Train Epoch: 1 [15360/60000]    Loss: -0.282390
Train Epoch: 1 [16000/60000]    Loss: -0.275054
Train Epoch: 1 [16640/60000]    Loss: -0.283097
Train Epoch: 1 [17280/60000]    Loss: -0.330993
Train Epoch: 1 [17920/60000]    Loss: -0.323328
Train Epoch: 1 [18560/60000]    Loss: -0.388412
Train Epoch: 1 [19200/60000]    Loss: -0.373068
Train Epoch: 1 [19840/60000]    Loss: -0.373483
Train Epoch: 1 [20480/60000]    Loss: -0.342372
Train Epoch: 1 [21120/60000]    Loss: -0.406032
Train Epoch: 1 [21760/60000]    Loss: -0.531021
Train Epoch: 1 [22400/60000]    Loss: -0.564071
Train Epoch: 1 [23040/60000]    Loss: -0.486898
Train Epoch: 1 [23680/60000]    Loss: -0.516196
Train Epoch: 1 [24320/60000]    Loss: -0.524935
Train Epoch: 1 [24960/60000]    Loss: -0.468695
Train Epoch: 1 [25600/60000]    Loss: -0.498044
Train Epoch: 1 [26240/60000]    Loss: -0.556677
Train Epoch: 1 [26880/60000]    Loss: -0.550768
Train Epoch: 1 [27520/60000]    Loss: -0.570247
Train Epoch: 1 [28160/60000]    Loss: -0.449002
Train Epoch: 1 [28800/60000]    Loss: -0.592676
Train Epoch: 1 [29440/60000]    Loss: -0.607358
Train Epoch: 1 [30080/60000]    Loss: -0.553971
Train Epoch: 1 [30720/60000]    Loss: -0.558537
Train Epoch: 1 [31360/60000]    Loss: -0.614689
Train Epoch: 1 [32000/60000]    Loss: -0.596239
Train Epoch: 1 [32640/60000]    Loss: -0.564226
Train Epoch: 1 [33280/60000]    Loss: -0.570021
Train Epoch: 1 [33920/60000]    Loss: -0.551311
Train Epoch: 1 [34560/60000]    Loss: -0.678421
Train Epoch: 1 [35200/60000]    Loss: -0.688249
Train Epoch: 1 [35840/60000]    Loss: -0.660631
Train Epoch: 1 [36480/60000]    Loss: -0.531748
Train Epoch: 1 [37120/60000]    Loss: -0.642142
Train Epoch: 1 [37760/60000]    Loss: -0.536315
Train Epoch: 1 [38400/60000]    Loss: -0.547156
Train Epoch: 1 [39040/60000]    Loss: -0.528854
Train Epoch: 1 [39680/60000]    Loss: -0.566093
Train Epoch: 1 [40320/60000]    Loss: -0.560190
Train Epoch: 1 [40960/60000]    Loss: -0.655219
Train Epoch: 1 [41600/60000]    Loss: -0.635748
Train Epoch: 1 [42240/60000]    Loss: -0.627117
Train Epoch: 1 [42880/60000]    Loss: -0.536732
Train Epoch: 1 [43520/60000]    Loss: -0.702740
Train Epoch: 1 [44160/60000]    Loss: -0.577382
Train Epoch: 1 [44800/60000]    Loss: -0.615754
Train Epoch: 1 [45440/60000]    Loss: -0.559068
Train Epoch: 1 [46080/60000]    Loss: -0.582822
Train Epoch: 1 [46720/60000]    Loss: -0.600931
Train Epoch: 1 [47360/60000]    Loss: -0.577555
Train Epoch: 1 [48000/60000]    Loss: -0.580154
Train Epoch: 1 [48640/60000]    Loss: -0.621372
Train Epoch: 1 [49280/60000]    Loss: -0.484013
Train Epoch: 1 [49920/60000]    Loss: -0.609300
Train Epoch: 1 [50560/60000]    Loss: -0.678620
Train Epoch: 1 [51200/60000]    Loss: -0.716142
Train Epoch: 1 [51840/60000]    Loss: -0.690345
Train Epoch: 1 [52480/60000]    Loss: -0.577386
Train Epoch: 1 [53120/60000]    Loss: -0.550344
Train Epoch: 1 [53760/60000]    Loss: -0.703756
Train Epoch: 1 [54400/60000]    Loss: -0.646195
Train Epoch: 1 [55040/60000]    Loss: -0.580920
Train Epoch: 1 [55680/60000]    Loss: -0.565823
Train Epoch: 1 [56320/60000]    Loss: -0.613335
Train Epoch: 1 [56960/60000]    Loss: -0.656463
Train Epoch: 1 [57600/60000]    Loss: -0.603948
Train Epoch: 1 [58240/60000]    Loss: -0.650277
Train Epoch: 1 [58880/60000]    Loss: -0.633086
Train Epoch: 1 [59520/60000]    Loss: -0.602668
```

```
/home/alimuhammad/.local/lib/python3.10/site-packages/torch/nn/_reduction.py:51:
UserWarning: size_average and reduce args will be deprecated, please use
reduction='sum' instead.
  warnings.warn(warning.format(ret))


Test set: Avg. loss: -0.6637, Accuracy: 6684/10000 (67%)

Train Epoch: 2 [0/60000]        Loss: -0.559267
Train Epoch: 2 [640/60000]      Loss: -0.663822
Train Epoch: 2 [1280/60000]     Loss: -0.639676
Train Epoch: 2 [1920/60000]     Loss: -0.654603
Train Epoch: 2 [2560/60000]     Loss: -0.593948
Train Epoch: 2 [3200/60000]     Loss: -0.674888
Train Epoch: 2 [3840/60000]     Loss: -0.616088
Train Epoch: 2 [4480/60000]     Loss: -0.547521
Train Epoch: 2 [5120/60000]     Loss: -0.673225
Train Epoch: 2 [5760/60000]     Loss: -0.546283
Train Epoch: 2 [6400/60000]     Loss: -0.639926
Train Epoch: 2 [7040/60000]     Loss: -0.631537
Train Epoch: 2 [7680/60000]     Loss: -0.606824
Train Epoch: 2 [8320/60000]     Loss: -0.581745
Train Epoch: 2 [8960/60000]     Loss: -0.636393
Train Epoch: 2 [9600/60000]     Loss: -0.555195
Train Epoch: 2 [10240/60000]    Loss: -0.685454
Train Epoch: 2 [10880/60000]    Loss: -0.630742
Train Epoch: 2 [11520/60000]    Loss: -0.646365
Train Epoch: 2 [12160/60000]    Loss: -0.718778
Train Epoch: 2 [12800/60000]    Loss: -0.597505
Train Epoch: 2 [13440/60000]    Loss: -0.459769
Train Epoch: 2 [14080/60000]    Loss: -0.619223
Train Epoch: 2 [14720/60000]    Loss: -0.562999
Train Epoch: 2 [15360/60000]    Loss: -0.669336
Train Epoch: 2 [16000/60000]    Loss: -0.680074
Train Epoch: 2 [16640/60000]    Loss: -0.610821
Train Epoch: 2 [17280/60000]    Loss: -0.699690
Train Epoch: 2 [17920/60000]    Loss: -0.625481
Train Epoch: 2 [18560/60000]    Loss: -0.612948
Train Epoch: 2 [19200/60000]    Loss: -0.568221
Train Epoch: 2 [19840/60000]    Loss: -0.688006
Train Epoch: 2 [20480/60000]    Loss: -0.649823
Train Epoch: 2 [21120/60000]    Loss: -0.618200
Train Epoch: 2 [21760/60000]    Loss: -0.523843
Train Epoch: 2 [22400/60000]    Loss: -0.578707
Train Epoch: 2 [23040/60000]    Loss: -0.719320
Train Epoch: 2 [23680/60000]    Loss: -0.723654
Train Epoch: 2 [24320/60000]    Loss: -0.702957
Train Epoch: 2 [24960/60000]    Loss: -0.628886
Train Epoch: 2 [25600/60000]    Loss: -0.684077
Train Epoch: 2 [26240/60000]    Loss: -0.623595
Train Epoch: 2 [26880/60000]    Loss: -0.687876
Train Epoch: 2 [27520/60000]    Loss: -0.686607
Train Epoch: 2 [28160/60000]    Loss: -0.706995
Train Epoch: 2 [28800/60000]    Loss: -0.588414
Train Epoch: 2 [29440/60000]    Loss: -0.640492
Train Epoch: 2 [30080/60000]    Loss: -0.534621
Train Epoch: 2 [30720/60000]    Loss: -0.539079
Train Epoch: 2 [31360/60000]    Loss: -0.601319
Train Epoch: 2 [32000/60000]    Loss: -0.613452
Train Epoch: 2 [32640/60000]    Loss: -0.746390
Train Epoch: 2 [33280/60000]    Loss: -0.675874
Train Epoch: 2 [33920/60000]    Loss: -0.633570
Train Epoch: 2 [34560/60000]    Loss: -0.501548
Train Epoch: 2 [35200/60000]    Loss: -0.558009
Train Epoch: 2 [35840/60000]    Loss: -0.628163
Train Epoch: 2 [36480/60000]    Loss: -0.623780
Train Epoch: 2 [37120/60000]    Loss: -0.633125
Train Epoch: 2 [37760/60000]    Loss: -0.588262
Train Epoch: 2 [38400/60000]    Loss: -0.654039
Train Epoch: 2 [39040/60000]    Loss: -0.663808
Train Epoch: 2 [39680/60000]    Loss: -0.724195
Train Epoch: 2 [40320/60000]    Loss: -0.561451
Train Epoch: 2 [40960/60000]    Loss: -0.663390
Train Epoch: 2 [41600/60000]    Loss: -0.746495
Train Epoch: 2 [42240/60000]    Loss: -0.675052
Train Epoch: 2 [42880/60000]    Loss: -0.626581
Train Epoch: 2 [43520/60000]    Loss: -0.617907
Train Epoch: 2 [44160/60000]    Loss: -0.717704
Train Epoch: 2 [44800/60000]    Loss: -0.768586
Train Epoch: 2 [45440/60000]    Loss: -0.606094
Train Epoch: 2 [46080/60000]    Loss: -0.639892
Train Epoch: 2 [46720/60000]    Loss: -0.713506
Train Epoch: 2 [47360/60000]    Loss: -0.733411
Train Epoch: 2 [48000/60000]    Loss: -0.712817
Train Epoch: 2 [48640/60000]    Loss: -0.577284
Train Epoch: 2 [49280/60000]    Loss: -0.621387
Train Epoch: 2 [49920/60000]    Loss: -0.722664
Train Epoch: 2 [50560/60000]    Loss: -0.585734
Train Epoch: 2 [51200/60000]    Loss: -0.704489
Train Epoch: 2 [51840/60000]    Loss: -0.639479
Train Epoch: 2 [52480/60000]    Loss: -0.608937
Train Epoch: 2 [53120/60000]    Loss: -0.731561
Train Epoch: 2 [53760/60000]    Loss: -0.607747
Train Epoch: 2 [54400/60000]    Loss: -0.703608
Train Epoch: 2 [55040/60000]    Loss: -0.651096
Train Epoch: 2 [55680/60000]    Loss: -0.733718
Train Epoch: 2 [56320/60000]    Loss: -0.580122
Train Epoch: 2 [56960/60000]    Loss: -0.666387
Train Epoch: 2 [57600/60000]    Loss: -0.665299
Train Epoch: 2 [58240/60000]    Loss: -0.632285
Train Epoch: 2 [58880/60000]    Loss: -0.664572
Train Epoch: 2 [59520/60000]    Loss: -0.658409

Test set: Avg. loss: -0.6716, Accuracy: 6740/10000 (67%)

Train Epoch: 3 [0/60000]        Loss: -0.717880
Train Epoch: 3 [640/60000]      Loss: -0.580610
Train Epoch: 3 [1280/60000]     Loss: -0.620144
Train Epoch: 3 [1920/60000]     Loss: -0.763970
Train Epoch: 3 [2560/60000]     Loss: -0.657811
Train Epoch: 3 [3200/60000]     Loss: -0.705937
Train Epoch: 3 [3840/60000]     Loss: -0.643359
Train Epoch: 3 [4480/60000]     Loss: -0.637090
Train Epoch: 3 [5120/60000]     Loss: -0.644063
Train Epoch: 3 [5760/60000]     Loss: -0.701771
Train Epoch: 3 [6400/60000]     Loss: -0.722672
Train Epoch: 3 [7040/60000]     Loss: -0.682965
Train Epoch: 3 [7680/60000]     Loss: -0.707342
Train Epoch: 3 [8320/60000]     Loss: -0.687749
Train Epoch: 3 [8960/60000]     Loss: -0.790220
Train Epoch: 3 [9600/60000]     Loss: -0.518230
Train Epoch: 3 [10240/60000]    Loss: -0.668574
Train Epoch: 3 [10880/60000]    Loss: -0.638259
Train Epoch: 3 [11520/60000]    Loss: -0.539672
Train Epoch: 3 [12160/60000]    Loss: -0.684249
Train Epoch: 3 [12800/60000]    Loss: -0.671207
Train Epoch: 3 [13440/60000]    Loss: -0.693473
Train Epoch: 3 [14080/60000]    Loss: -0.777510
Train Epoch: 3 [14720/60000]    Loss: -0.602130
Train Epoch: 3 [15360/60000]    Loss: -0.676739
Train Epoch: 3 [16000/60000]    Loss: -0.618993
Train Epoch: 3 [16640/60000]    Loss: -0.553310
Train Epoch: 3 [17280/60000]    Loss: -0.706257
Train Epoch: 3 [17920/60000]    Loss: -0.712322
Train Epoch: 3 [18560/60000]    Loss: -0.718308
Train Epoch: 3 [19200/60000]    Loss: -0.672414
Train Epoch: 3 [19840/60000]    Loss: -0.697674
Train Epoch: 3 [20480/60000]    Loss: -0.631582
Train Epoch: 3 [21120/60000]    Loss: -0.655502
Train Epoch: 3 [21760/60000]    Loss: -0.715823
Train Epoch: 3 [22400/60000]    Loss: -0.620172
Train Epoch: 3 [23040/60000]    Loss: -0.614591
Train Epoch: 3 [23680/60000]    Loss: -0.727719
Train Epoch: 3 [24320/60000]    Loss: -0.662324
Train Epoch: 3 [24960/60000]    Loss: -0.662433
Train Epoch: 3 [25600/60000]    Loss: -0.576011
Train Epoch: 3 [26240/60000]    Loss: -0.597863
Train Epoch: 3 [26880/60000]    Loss: -0.714018
Train Epoch: 3 [27520/60000]    Loss: -0.736945
Train Epoch: 3 [28160/60000]    Loss: -0.685224
Train Epoch: 3 [28800/60000]    Loss: -0.643143
Train Epoch: 3 [29440/60000]    Loss: -0.645308
Train Epoch: 3 [30080/60000]    Loss: -0.704708
Train Epoch: 3 [30720/60000]    Loss: -0.636429
Train Epoch: 3 [31360/60000]    Loss: -0.673116
Train Epoch: 3 [32000/60000]    Loss: -0.737914
Train Epoch: 3 [32640/60000]    Loss: -0.594340
Train Epoch: 3 [33280/60000]    Loss: -0.663947
Train Epoch: 3 [33920/60000]    Loss: -0.585499
Train Epoch: 3 [34560/60000]    Loss: -0.731426
Train Epoch: 3 [35200/60000]    Loss: -0.701900
Train Epoch: 3 [35840/60000]    Loss: -0.625270
Train Epoch: 3 [36480/60000]    Loss: -0.666435
Train Epoch: 3 [37120/60000]    Loss: -0.727797
Train Epoch: 3 [37760/60000]    Loss: -0.727936
Train Epoch: 3 [38400/60000]    Loss: -0.726231
Train Epoch: 3 [39040/60000]    Loss: -0.644205
Train Epoch: 3 [39680/60000]    Loss: -0.585189
Train Epoch: 3 [40320/60000]    Loss: -0.604979
Train Epoch: 3 [40960/60000]    Loss: -0.623222
Train Epoch: 3 [41600/60000]    Loss: -0.671184
Train Epoch: 3 [42240/60000]    Loss: -0.581036
Train Epoch: 3 [42880/60000]    Loss: -0.633535
Train Epoch: 3 [43520/60000]    Loss: -0.655853
Train Epoch: 3 [44160/60000]    Loss: -0.587631
Train Epoch: 3 [44800/60000]    Loss: -0.708074
Train Epoch: 3 [45440/60000]    Loss: -0.676694
Train Epoch: 3 [46080/60000]    Loss: -0.560758
Train Epoch: 3 [46720/60000]    Loss: -0.662860
Train Epoch: 3 [47360/60000]    Loss: -0.663773
Train Epoch: 3 [48000/60000]    Loss: -0.695894
Train Epoch: 3 [48640/60000]    Loss: -0.562099
Train Epoch: 3 [49280/60000]    Loss: -0.711579
Train Epoch: 3 [49920/60000]    Loss: -0.686872
Train Epoch: 3 [50560/60000]    Loss: -0.539622
Train Epoch: 3 [51200/60000]    Loss: -0.637496
Train Epoch: 3 [51840/60000]    Loss: -0.699095
Train Epoch: 3 [52480/60000]    Loss: -0.615067
Train Epoch: 3 [53120/60000]    Loss: -0.704349
Train Epoch: 3 [53760/60000]    Loss: -0.662648
Train Epoch: 3 [54400/60000]    Loss: -0.629095
Train Epoch: 3 [55040/60000]    Loss: -0.688520
Train Epoch: 3 [55680/60000]    Loss: -0.588750
Train Epoch: 3 [56320/60000]    Loss: -0.656941
Train Epoch: 3 [56960/60000]    Loss: -0.728638
Train Epoch: 3 [57600/60000]    Loss: -0.563942
Train Epoch: 3 [58240/60000]    Loss: -0.548167
Train Epoch: 3 [58880/60000]    Loss: -0.682120
Train Epoch: 3 [59520/60000]    Loss: -0.701719

Test set: Avg. loss: -0.6785, Accuracy: 6794/10000 (68%)
```

[17]:

```
# Run network on data we got before and show predictions
output = model(example_data)

fig = plt.figure()
for i in range(20):
  plt.subplot(5,5,i+1)
  plt.tight_layout()
  plt.imshow(example_data[i][0], cmap='gray', interpolation='none')
  plt.title("Prediction: {}".format(
    output.data.max(1, keepdim=True)[1][i].item()))
  plt.xticks([])
  plt.yticks([])
plt.show()
```



5 Q2: TODO [45 Points]

You implemented the previous given architecture. Now let's implement a modified version of AlexNet on mnist.

if you don't know what alexnet is, don't worry. This lab doesn't require the background of it. Its just an architecture and we are implementing it.

Here is AlexNet Architecture: 1. Convolution with kernel size 5, stride 1, padding 1, 1 input channel and 32 output channels 2. A Relu 3. Convolution with kernel size 3, padding 1, 32 input channels and 64 output channels 4. A Relu 5. A max pooling operation over a 2x2 area, stride 2 6. Convolution with kernel size 3, padding 1, 64 input channels and 96 output channels 7. A Relu 8. Convolution with kernel size 3, padding 1, 96 input channels and 64 output channels 9. A Relu 10. Convolution with kernel size 3, padding 1, 64 input channels and 32 output channels 11. A Relu 12. A max pooling operation over a 2x2 area, stride 1

13. A flattening operation
14. A dropout
15. A fully connected layer mapping from (whatever dimensions we are at– find out using .shape) to 2048
16. A ReLU

17. A fully connected layer mapping from 2048 to 1024
18. A ReLU
19. A fully connected layer mapping from 1024 to 10

[14]:
```
class Net(nn.Module):
    def __init__(self, num=10):
        super(Net, self).__init__()
        self.model=nn.Sequential(

            #nn.MaxPool2d( kernel_size=2, stride=2) This code can be used for␣
 ↪max pooling. Google it to understand it.
            #nn.Dropout() can be used for dropout

            #1. Conv Net kernel size 5, stride 1, padding 1, 1 input channel,␣
 ↪32 output channels
            nn.Conv2d(1, 32, kernel_size=5, stride=1, padding=1),
            nn.ReLU(), # 2. Relu activation function
            #3. Conv with kernel size 3, padding 1, 32 input channels, 64␣
 ↪output channels
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(), # 4. Relu activation function
            #5. Max pooling operation over 2x2 area, stride 2
            nn.MaxPool2d(kernel_size=2, stride=2),
            #6. Conv with kernel size 3, padding 1, 64 input channels, 96␣
 ↪output channels
            nn.Conv2d(64, 96, kernel_size=3, padding=1),
            nn.ReLU(), # 7. Relu activation function
            #8. Conv with kernel size 3, padding 1, 96 input channels, 64␣
 ↪output channels
            nn.Conv2d(96, 64, kernel_size=3, padding=1),
            nn.ReLU(), # 9. Relu activation function
            #10. Conv with kernel size 3, padding 1, 64 input channels and 32␣
 ↪output channels
            nn.Conv2d(64, 32, kernel_size=3, padding=1),
            nn.ReLU(), # 11. Relu activation function
            #12. Max pooling operation over 2x2 area, stride 1
            nn.MaxPool2d(kernel_size=2, stride=1),
            #13. Flatten the output
            nn.Flatten(),
            #14. Dropout
            nn.Dropout(),
            #15. Fully Connected Layer mapping
            nn.Linear(4608, 2048),
            nn.ReLU(), # 16. Relu activation function
            #17. Fully Connected Layer mapping from 2048 to 1024
            nn.Linear(2048, 1024),
            nn.ReLU(), # 18. Relu activation function
            #19. Fully Connected Layer mapping from 1024 to 10
            nn.Linear(1024, 10),
        )

    def forward(self, x):
        x = self.model(x)
        return x
```

[15]:
```
#Do NOT EDIT
#USE THIS CODE FOR TESTING
#Since we can't compare the model, its better to look at the its printed␣
 ↪structure and compare it with layers given in architecture
model = Net()
print(model)

model.apply(weights_init)
if torch.cuda.is_available():
    model.cuda()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
```

```
Net(
  (model): Sequential(
    (0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (5): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(96, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1,
ceil_mode=False)
    (12): Flatten(start_dim=1, end_dim=-1)
    (13): Dropout(p=0.5, inplace=False)
    (14): Linear(in_features=4608, out_features=2048, bias=True)
    (15): ReLU()
    (16): Linear(in_features=2048, out_features=1024, bias=True)
    (17): ReLU()
    (18): Linear(in_features=1024, out_features=10, bias=True)
  )
)
```

[16]:
```
# Main training routine
def train(epoch):
  model.train()
  # Get each
  for batch_idx, (data, target) in enumerate(train_loader):
    if torch.cuda.is_available():
      data, target = data.cuda(), target.cuda()
    data, target = Variable(data), Variable(target)
    optimizer.zero_grad()
    output = model(data)
    loss = F.cross_entropy(output, target)
    loss.backward()
    optimizer.step()
    # Store results
    if batch_idx % 10 == 0:
      print('Train Epoch: {} [{}/{}]\tLoss: {:.6f}'.format(
        epoch, batch_idx * len(data), len(train_loader.dataset), loss.item()))
```

[17]:
```
# Run on test data
def test():
  model.eval()
  test_loss = 0
  correct = 0
  for data, target in test_loader:
      if torch.cuda.is_available():
          data, target = data.cuda(), target.cuda()
      data, target = Variable(data, volatile=True), Variable(target)
      output = model(data)
      test_loss += F.cross_entropy(output, target, size_average=False).item()#␣
 ↪sum up batch loss
      pred = output.data.max(1, keepdim=True)[1]# get the index of the max␣
 ↪log-probability
      correct += pred.eq(target.data.view_as(pred)).long().cpu().sum()
  test_loss /= len(test_loader.dataset)
  correct=float(correct.to(torch.device('cpu')).numpy())
  print('\nTest set: Avg. loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
    test_loss, correct, len(test_loader.dataset),
    100. * correct / len(test_loader.dataset)))
```

[38]:
```
# Train for three epochs
n_epochs = 3
for epoch in range(1, n_epochs + 1):
  train(epoch)
  test()
```

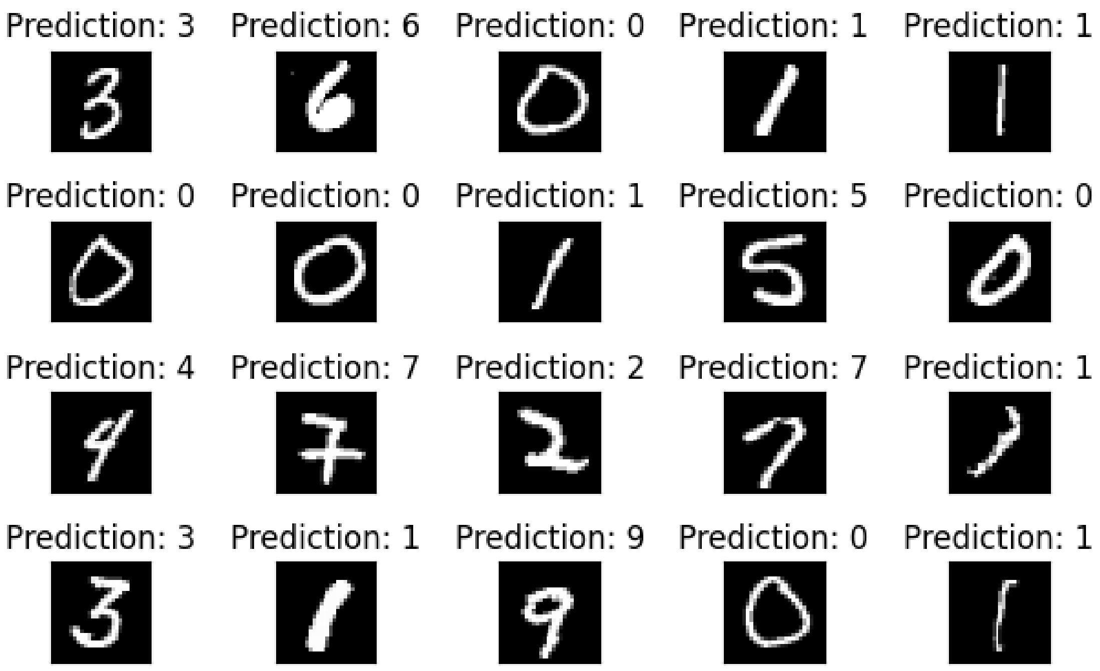
```
Train Epoch: 1 [0/60000]        Loss: 2.299699
Train Epoch: 1 [640/60000]      Loss: 2.306395
Train Epoch: 1 [1280/60000]     Loss: 2.302206
Train Epoch: 1 [1920/60000]     Loss: 2.298068
Train Epoch: 1 [2560/60000]     Loss: 2.278997
Train Epoch: 1 [3200/60000]     Loss: 2.247399
Train Epoch: 1 [3840/60000]     Loss: 2.264837
Train Epoch: 1 [4480/60000]     Loss: 2.196402
Train Epoch: 1 [5120/60000]     Loss: 2.112981
Train Epoch: 1 [5760/60000]     Loss: 1.790514
Train Epoch: 1 [6400/60000]     Loss: 1.215119
Train Epoch: 1 [7040/60000]     Loss: 1.169026
Train Epoch: 1 [7680/60000]     Loss: 0.758016
Train Epoch: 1 [8320/60000]     Loss: 0.609010
Train Epoch: 1 [8960/60000]     Loss: 0.449561
Train Epoch: 1 [9600/60000]     Loss: 0.436160
Train Epoch: 1 [10240/60000]    Loss: 0.383146
Train Epoch: 1 [10880/60000]    Loss: 0.564202
Train Epoch: 1 [11520/60000]    Loss: 0.432428
Train Epoch: 1 [12160/60000]    Loss: 0.591433
Train Epoch: 1 [12800/60000]    Loss: 0.586567
Train Epoch: 1 [13440/60000]    Loss: 0.337868
Train Epoch: 1 [14080/60000]    Loss: 0.324821
Train Epoch: 1 [14720/60000]    Loss: 0.215696
Train Epoch: 1 [15360/60000]    Loss: 0.201340
Train Epoch: 1 [16000/60000]    Loss: 0.375102
Train Epoch: 1 [16640/60000]    Loss: 0.163078
Train Epoch: 1 [17280/60000]    Loss: 0.164357
Train Epoch: 1 [17920/60000]    Loss: 0.234158
Train Epoch: 1 [18560/60000]    Loss: 0.131528
Train Epoch: 1 [19200/60000]    Loss: 0.100199
Train Epoch: 1 [19840/60000]    Loss: 0.167667
Train Epoch: 1 [20480/60000]    Loss: 0.307461
Train Epoch: 1 [21120/60000]    Loss: 0.259345
Train Epoch: 1 [21760/60000]    Loss: 0.297175
Train Epoch: 1 [22400/60000]    Loss: 0.311141
Train Epoch: 1 [23040/60000]    Loss: 0.432944
Train Epoch: 1 [23680/60000]    Loss: 0.087990
Train Epoch: 1 [24320/60000]    Loss: 0.217024
Train Epoch: 1 [24960/60000]    Loss: 0.073831
Train Epoch: 1 [25600/60000]    Loss: 0.210217
Train Epoch: 1 [26240/60000]    Loss: 0.308926
Train Epoch: 1 [26880/60000]    Loss: 0.109871
Train Epoch: 1 [27520/60000]    Loss: 0.159654
Train Epoch: 1 [28160/60000]    Loss: 0.111380
Train Epoch: 1 [28800/60000]    Loss: 0.135359
Train Epoch: 1 [29440/60000]    Loss: 0.169289
Train Epoch: 1 [30080/60000]    Loss: 0.435845
Train Epoch: 1 [30720/60000]    Loss: 0.093585
Train Epoch: 1 [31360/60000]    Loss: 0.178362
Train Epoch: 1 [32000/60000]    Loss: 0.061511
Train Epoch: 1 [32640/60000]    Loss: 0.253264
Train Epoch: 1 [33280/60000]    Loss: 0.075764
Train Epoch: 1 [33920/60000]    Loss: 0.085600
Train Epoch: 1 [34560/60000]    Loss: 0.072068
Train Epoch: 1 [35200/60000]    Loss: 0.159014
Train Epoch: 1 [35840/60000]    Loss: 0.215856
Train Epoch: 1 [36480/60000]    Loss: 0.176219
Train Epoch: 1 [37120/60000]    Loss: 0.104767
Train Epoch: 1 [37760/60000]    Loss: 0.258646
Train Epoch: 1 [38400/60000]    Loss: 0.112057
Train Epoch: 1 [39040/60000]    Loss: 0.091206
Train Epoch: 1 [39680/60000]    Loss: 0.340690
Train Epoch: 1 [40320/60000]    Loss: 0.071073
Train Epoch: 1 [40960/60000]    Loss: 0.119849
Train Epoch: 1 [41600/60000]    Loss: 0.114836
Train Epoch: 1 [42240/60000]    Loss: 0.015412
Train Epoch: 1 [42880/60000]    Loss: 0.066384
Train Epoch: 1 [43520/60000]    Loss: 0.146144
Train Epoch: 1 [44160/60000]    Loss: 0.115393
Train Epoch: 1 [44800/60000]    Loss: 0.082780
Train Epoch: 1 [45440/60000]    Loss: 0.044503
Train Epoch: 1 [46080/60000]    Loss: 0.140778
Train Epoch: 1 [46720/60000]    Loss: 0.197400
Train Epoch: 1 [47360/60000]    Loss: 0.195818
Train Epoch: 1 [48000/60000]    Loss: 0.043226
Train Epoch: 1 [48640/60000]    Loss: 0.095829
Train Epoch: 1 [49280/60000]    Loss: 0.188767
Train Epoch: 1 [49920/60000]    Loss: 0.086067
Train Epoch: 1 [50560/60000]    Loss: 0.161676
Train Epoch: 1 [51200/60000]    Loss: 0.217341
Train Epoch: 1 [51840/60000]    Loss: 0.233875
Train Epoch: 1 [52480/60000]    Loss: 0.210614
Train Epoch: 1 [53120/60000]    Loss: 0.108079
Train Epoch: 1 [53760/60000]    Loss: 0.113697
Train Epoch: 1 [54400/60000]    Loss: 0.186737
Train Epoch: 1 [55040/60000]    Loss: 0.117776
Train Epoch: 1 [55680/60000]    Loss: 0.070366
Train Epoch: 1 [56320/60000]    Loss: 0.060346
Train Epoch: 1 [56960/60000]    Loss: 0.068695
Train Epoch: 1 [57600/60000]    Loss: 0.082392
Train Epoch: 1 [58240/60000]    Loss: 0.114569
Train Epoch: 1 [58880/60000]    Loss: 0.127668
Train Epoch: 1 [59520/60000]    Loss: 0.121821
```

```
/tmp/ipykernel_336811/4025279313.py:9: UserWarning: volatile was removed and now
has no effect. Use `with torch.no_grad():` instead.
  data, target = Variable(data, volatile=True), Variable(target)
/home/alimuhammad/.local/lib/python3.10/site-packages/torch/nn/_reduction.py:51:
```

```
UserWarning: size_average and reduce args will be deprecated, please use
reduction='sum' instead.
  warnings.warn(warning.format(ret))


Test set: Avg. loss: 0.0707, Accuracy: 9765.0/10000 (98%)

Train Epoch: 2 [0/60000]        Loss: 0.080573
Train Epoch: 2 [640/60000]      Loss: 0.329982
Train Epoch: 2 [1280/60000]     Loss: 0.075456
Train Epoch: 2 [1920/60000]     Loss: 0.131028
Train Epoch: 2 [2560/60000]     Loss: 0.027370
Train Epoch: 2 [3200/60000]     Loss: 0.195240
Train Epoch: 2 [3840/60000]     Loss: 0.081495
Train Epoch: 2 [4480/60000]     Loss: 0.140562
Train Epoch: 2 [5120/60000]     Loss: 0.038412
Train Epoch: 2 [5760/60000]     Loss: 0.175197
Train Epoch: 2 [6400/60000]     Loss: 0.038040
Train Epoch: 2 [7040/60000]     Loss: 0.153490
Train Epoch: 2 [7680/60000]     Loss: 0.045921
Train Epoch: 2 [8320/60000]     Loss: 0.042664
Train Epoch: 2 [8960/60000]     Loss: 0.021339
Train Epoch: 2 [9600/60000]     Loss: 0.024144
Train Epoch: 2 [10240/60000]    Loss: 0.011187
Train Epoch: 2 [10880/60000]    Loss: 0.143326
Train Epoch: 2 [11520/60000]    Loss: 0.080839
Train Epoch: 2 [12160/60000]    Loss: 0.079510
Train Epoch: 2 [12800/60000]    Loss: 0.118302
Train Epoch: 2 [13440/60000]    Loss: 0.104994
Train Epoch: 2 [14080/60000]    Loss: 0.139171
Train Epoch: 2 [14720/60000]    Loss: 0.032052
Train Epoch: 2 [15360/60000]    Loss: 0.136346
Train Epoch: 2 [16000/60000]    Loss: 0.117822
Train Epoch: 2 [16640/60000]    Loss: 0.162188
Train Epoch: 2 [17280/60000]    Loss: 0.317313
Train Epoch: 2 [17920/60000]    Loss: 0.061296
Train Epoch: 2 [18560/60000]    Loss: 0.028817
Train Epoch: 2 [19200/60000]    Loss: 0.082624
Train Epoch: 2 [19840/60000]    Loss: 0.117350
Train Epoch: 2 [20480/60000]    Loss: 0.136852
Train Epoch: 2 [21120/60000]    Loss: 0.173204
Train Epoch: 2 [21760/60000]    Loss: 0.063771
Train Epoch: 2 [22400/60000]    Loss: 0.170708
Train Epoch: 2 [23040/60000]    Loss: 0.136173
Train Epoch: 2 [23680/60000]    Loss: 0.042773
Train Epoch: 2 [24320/60000]    Loss: 0.072848
Train Epoch: 2 [24960/60000]    Loss: 0.162445
Train Epoch: 2 [25600/60000]    Loss: 0.064141
Train Epoch: 2 [26240/60000]    Loss: 0.041075
Train Epoch: 2 [26880/60000]    Loss: 0.099653
Train Epoch: 2 [27520/60000]    Loss: 0.069126
Train Epoch: 2 [28160/60000]    Loss: 0.082936
Train Epoch: 2 [28800/60000]    Loss: 0.034982
Train Epoch: 2 [29440/60000]    Loss: 0.080916
Train Epoch: 2 [30080/60000]    Loss: 0.152011
Train Epoch: 2 [30720/60000]    Loss: 0.096990
Train Epoch: 2 [31360/60000]    Loss: 0.014572
Train Epoch: 2 [32000/60000]    Loss: 0.113477
Train Epoch: 2 [32640/60000]    Loss: 0.239572
Train Epoch: 2 [33280/60000]    Loss: 0.052811
Train Epoch: 2 [33920/60000]    Loss: 0.017200
Train Epoch: 2 [34560/60000]    Loss: 0.016063
Train Epoch: 2 [35200/60000]    Loss: 0.229454
Train Epoch: 2 [35840/60000]    Loss: 0.035919
Train Epoch: 2 [36480/60000]    Loss: 0.060326
Train Epoch: 2 [37120/60000]    Loss: 0.042950
Train Epoch: 2 [37760/60000]    Loss: 0.170962
Train Epoch: 2 [38400/60000]    Loss: 0.193078
Train Epoch: 2 [39040/60000]    Loss: 0.115925
Train Epoch: 2 [39680/60000]    Loss: 0.031844
Train Epoch: 2 [40320/60000]    Loss: 0.034775
Train Epoch: 2 [40960/60000]    Loss: 0.116329
Train Epoch: 2 [41600/60000]    Loss: 0.025779
Train Epoch: 2 [42240/60000]    Loss: 0.111320
Train Epoch: 2 [42880/60000]    Loss: 0.130208
Train Epoch: 2 [43520/60000]    Loss: 0.042637
Train Epoch: 2 [44160/60000]    Loss: 0.040592
Train Epoch: 2 [44800/60000]    Loss: 0.108256
Train Epoch: 2 [45440/60000]    Loss: 0.011585
Train Epoch: 2 [46080/60000]    Loss: 0.141506
Train Epoch: 2 [46720/60000]    Loss: 0.019326
Train Epoch: 2 [47360/60000]    Loss: 0.043295
Train Epoch: 2 [48000/60000]    Loss: 0.019296
Train Epoch: 2 [48640/60000]    Loss: 0.043576
Train Epoch: 2 [49280/60000]    Loss: 0.086973
Train Epoch: 2 [49920/60000]    Loss: 0.152605
Train Epoch: 2 [50560/60000]    Loss: 0.087624
Train Epoch: 2 [51200/60000]    Loss: 0.247345
Train Epoch: 2 [51840/60000]    Loss: 0.105383
Train Epoch: 2 [52480/60000]    Loss: 0.079466
Train Epoch: 2 [53120/60000]    Loss: 0.067113
Train Epoch: 2 [53760/60000]    Loss: 0.052333
Train Epoch: 2 [54400/60000]    Loss: 0.093426
Train Epoch: 2 [55040/60000]    Loss: 0.043035
Train Epoch: 2 [55680/60000]    Loss: 0.017454
Train Epoch: 2 [56320/60000]    Loss: 0.013431
Train Epoch: 2 [56960/60000]    Loss: 0.077037
Train Epoch: 2 [57600/60000]    Loss: 0.011778
Train Epoch: 2 [58240/60000]    Loss: 0.078614
Train Epoch: 2 [58880/60000]    Loss: 0.060827
Train Epoch: 2 [59520/60000]    Loss: 0.030520

Test set: Avg. loss: 0.0447, Accuracy: 9840.0/10000 (98%)

Train Epoch: 3 [0/60000]        Loss: 0.075227
Train Epoch: 3 [640/60000]      Loss: 0.129816
Train Epoch: 3 [1280/60000]     Loss: 0.016172
Train Epoch: 3 [1920/60000]     Loss: 0.077524
Train Epoch: 3 [2560/60000]     Loss: 0.015969
Train Epoch: 3 [3200/60000]     Loss: 0.047004
Train Epoch: 3 [3840/60000]     Loss: 0.030636
Train Epoch: 3 [4480/60000]     Loss: 0.094136
Train Epoch: 3 [5120/60000]     Loss: 0.093861
Train Epoch: 3 [5760/60000]     Loss: 0.033166
Train Epoch: 3 [6400/60000]     Loss: 0.172267
Train Epoch: 3 [7040/60000]     Loss: 0.086585
Train Epoch: 3 [7680/60000]     Loss: 0.072561
Train Epoch: 3 [8320/60000]     Loss: 0.097636
Train Epoch: 3 [8960/60000]     Loss: 0.113387
Train Epoch: 3 [9600/60000]     Loss: 0.073412
Train Epoch: 3 [10240/60000]    Loss: 0.018251
Train Epoch: 3 [10880/60000]    Loss: 0.079054
Train Epoch: 3 [11520/60000]    Loss: 0.039550
Train Epoch: 3 [12160/60000]    Loss: 0.039739
Train Epoch: 3 [12800/60000]    Loss: 0.135620
Train Epoch: 3 [13440/60000]    Loss: 0.051504
Train Epoch: 3 [14080/60000]    Loss: 0.034107
Train Epoch: 3 [14720/60000]    Loss: 0.019974
Train Epoch: 3 [15360/60000]    Loss: 0.176739
Train Epoch: 3 [16000/60000]    Loss: 0.062975
Train Epoch: 3 [16640/60000]    Loss: 0.050910
Train Epoch: 3 [17280/60000]    Loss: 0.012853
Train Epoch: 3 [17920/60000]    Loss: 0.009985
Train Epoch: 3 [18560/60000]    Loss: 0.067407
Train Epoch: 3 [19200/60000]    Loss: 0.033844
Train Epoch: 3 [19840/60000]    Loss: 0.066806
Train Epoch: 3 [20480/60000]    Loss: 0.033219
Train Epoch: 3 [21120/60000]    Loss: 0.076635
Train Epoch: 3 [21760/60000]    Loss: 0.103104
Train Epoch: 3 [22400/60000]    Loss: 0.060729
Train Epoch: 3 [23040/60000]    Loss: 0.017559
Train Epoch: 3 [23680/60000]    Loss: 0.115763
Train Epoch: 3 [24320/60000]    Loss: 0.012266
Train Epoch: 3 [24960/60000]    Loss: 0.132735
Train Epoch: 3 [25600/60000]    Loss: 0.030350
Train Epoch: 3 [26240/60000]    Loss: 0.013092
Train Epoch: 3 [26880/60000]    Loss: 0.059016
Train Epoch: 3 [27520/60000]    Loss: 0.048329
Train Epoch: 3 [28160/60000]    Loss: 0.014979
Train Epoch: 3 [28800/60000]    Loss: 0.024349
Train Epoch: 3 [29440/60000]    Loss: 0.118508
Train Epoch: 3 [30080/60000]    Loss: 0.016659
Train Epoch: 3 [30720/60000]    Loss: 0.008146
Train Epoch: 3 [31360/60000]    Loss: 0.102524
Train Epoch: 3 [32000/60000]    Loss: 0.129704
Train Epoch: 3 [32640/60000]    Loss: 0.012553
Train Epoch: 3 [33280/60000]    Loss: 0.004595
Train Epoch: 3 [33920/60000]    Loss: 0.053838
Train Epoch: 3 [34560/60000]    Loss: 0.104775
Train Epoch: 3 [35200/60000]    Loss: 0.150009
Train Epoch: 3 [35840/60000]    Loss: 0.063603
Train Epoch: 3 [36480/60000]    Loss: 0.103215
Train Epoch: 3 [37120/60000]    Loss: 0.060353
Train Epoch: 3 [37760/60000]    Loss: 0.108725
Train Epoch: 3 [38400/60000]    Loss: 0.023666
Train Epoch: 3 [39040/60000]    Loss: 0.017713
Train Epoch: 3 [39680/60000]    Loss: 0.117181
Train Epoch: 3 [40320/60000]    Loss: 0.012874
Train Epoch: 3 [40960/60000]    Loss: 0.136160
Train Epoch: 3 [41600/60000]    Loss: 0.009886
Train Epoch: 3 [42240/60000]    Loss: 0.048342
Train Epoch: 3 [42880/60000]    Loss: 0.028873
Train Epoch: 3 [43520/60000]    Loss: 0.006229
Train Epoch: 3 [44160/60000]    Loss: 0.093690
Train Epoch: 3 [44800/60000]    Loss: 0.021718
Train Epoch: 3 [45440/60000]    Loss: 0.086357
Train Epoch: 3 [46080/60000]    Loss: 0.068129
Train Epoch: 3 [46720/60000]    Loss: 0.058219
Train Epoch: 3 [47360/60000]    Loss: 0.011244
Train Epoch: 3 [48000/60000]    Loss: 0.016489
Train Epoch: 3 [48640/60000]    Loss: 0.149930
Train Epoch: 3 [49280/60000]    Loss: 0.021854
Train Epoch: 3 [49920/60000]    Loss: 0.014818
Train Epoch: 3 [50560/60000]    Loss: 0.195184
Train Epoch: 3 [51200/60000]    Loss: 0.013990
Train Epoch: 3 [51840/60000]    Loss: 0.010398
Train Epoch: 3 [52480/60000]    Loss: 0.045115
Train Epoch: 3 [53120/60000]    Loss: 0.011328
Train Epoch: 3 [53760/60000]    Loss: 0.054220
Train Epoch: 3 [54400/60000]    Loss: 0.043525
Train Epoch: 3 [55040/60000]    Loss: 0.092797
Train Epoch: 3 [55680/60000]    Loss: 0.094825
Train Epoch: 3 [56320/60000]    Loss: 0.016119
Train Epoch: 3 [56960/60000]    Loss: 0.032239
Train Epoch: 3 [57600/60000]    Loss: 0.026798
Train Epoch: 3 [58240/60000]    Loss: 0.018730
Train Epoch: 3 [58880/60000]    Loss: 0.190900
Train Epoch: 3 [59520/60000]    Loss: 0.057338

Test set: Avg. loss: 0.0341, Accuracy: 9898.0/10000 (99%)
```

[39]:
```
# Run network on data we got before and show predictions
examples = enumerate(test_loader)
batch_idx, (example_data, example_target) = next(examples)
if torch.cuda.is_available():
  example_data, example_target = example_data.cuda(), example_target.cuda()
example_data, example_target = Variable(example_data, volatile=True),␣
 ↪Variable(example_target)
output = model(example_data)

fig = plt.figure()
for i in range(20):
  plt.subplot(5,5,i+1)
  plt.tight_layout()
  plt.imshow(example_data[i][0].cpu(), cmap='gray', interpolation='none')
  plt.title("Prediction: {}".format(
    output.data.max(1, keepdim=True)[1][i].item()))
  plt.xticks([])
  plt.yticks([])
plt.show()
```

```
/tmp/ipykernel_336811/2673449976.py:6: UserWarning: volatile was removed and now
has no effect. Use `with torch.no_grad():` instead.
  example_data, example_target = Variable(example_data, volatile=True),
Variable(example_target)
```



6 Q3: TODO [25 Points]

Try 2 additional techniques 1. Data augmentation for increasing data 2. Number of epochs 3. Batch size in dataloader

implement this under this cell and give your opinion about does data augmentation helps in improving accuracy. WHat effetc does changing epochs have on the training. What effect did changing batch size have. You can utilize AI Tools for this as well.

[18]:
```
class Net(nn.Module):
    def __init__(self, num=10):
        super(Net, self).__init__()
        self.model=nn.Sequential(

            #nn.MaxPool2d( kernel_size=2, stride=2) This code can be used for␣
↪max pooling. Google it to understand it.
            #nn.Dropout() can be used for dropout

            #1. Conv Net kernel size 5, stride 1, padding 1, 1 input channel,␣
↪32 output channels
            nn.Conv2d(1, 32, kernel_size=5, stride=1, padding=1),
            nn.ReLU(), # 2. Relu activation function
            #3. Conv with kernel size 3, padding 1, 32 input channels, 64␣
↪output channels
            nn.Conv2d(32, 64, kernel_size=3, padding=1),
            nn.ReLU(), # 4. Relu activation function
            #5. Max pooling operation over 2x2 area, stride 2
            nn.MaxPool2d(kernel_size=2, stride=2),
            #6. Conv with kernel size 3, padding 1, 64 input channels, 96␣
↪output channels
            nn.Conv2d(64, 96, kernel_size=3, padding=1),
            nn.ReLU(), # 7. Relu activation function
            #8. Conv with kernel size 3, padding 1, 96 input channels, 64␣
↪output channels
            nn.Conv2d(96, 64, kernel_size=3, padding=1),
            nn.ReLU(), # 9. Relu activation function
            #10. Conv with kernel size 3, padding 1, 64 input channels and 32␣
↪output channels
            nn.Conv2d(64, 32, kernel_size=3, padding=1),
            nn.ReLU(), # 11. Relu activation function
            #12. Max pooling operation over 2x2 area, stride 1
            nn.MaxPool2d(kernel_size=2, stride=1),
            #13. Flatten the output
            nn.Flatten(),
            #14. Dropout
            nn.Dropout(),
            #15. Fully Connected Layer mapping
            nn.Linear(4608, 2048),
            nn.ReLU(), # 16. Relu activation function
            #17. Fully Connected Layer mapping from 2048 to 1024
            nn.Linear(2048, 1024),
            nn.ReLU(), # 18. Relu activation function
            #19. Fully Connected Layer mapping from 1024 to 10
            nn.Linear(1024, 10),
        )

    def forward(self, x):
        x = self.model(x)
        return x
```

[25]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision.transforms as transforms
import torchvision
from torch.autograd import Variable
import matplotlib.pyplot as plt
import torch.nn.functional as F
from torch.utils.data import ConcatDataset, Subset, DataLoader
import torchvision.transforms as transforms
import numpy as np
```

```
# Define data augmentation transformations
transform_augmented = transforms.Compose([
    transforms.RandomRotation(10),          # Random rotation by 10 degrees
    transforms.RandomHorizontalFlip(),       # Random horizontal flip
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])

# Original dataset (without augmentation)
original_dataset = torchvision.datasets.MNIST(
    './files/', train=True, download=True,
    transform=transforms.Compose([
        transforms.ToTensor(),
        transforms.Normalize((0.1307,), (0.3081,))
    ])
)

# Subset of the original data to apply transformations
augmented_subset_size = int(0.5 * len(original_dataset))  # Increase size by 50%
indices = np.random.choice(len(original_dataset), augmented_subset_size,␣
 ↪replace=False)
augmented_subset = Subset(original_dataset, indices)

# Create an augmented dataset using the subset and transformations
augmented_dataset = torchvision.datasets.MNIST(
    './files/', train=True, download=True, transform=transform_augmented
)
augmented_dataset = Subset(augmented_dataset, indices)

# Combine original and augmented datasets
combined_dataset = ConcatDataset([original_dataset, augmented_dataset])

# DataLoader with the combined dataset
train_loader_augmented = DataLoader(combined_dataset,␣
 ↪batch_size=batch_size_train, shuffle=True)

#Do NOT EDIT
#USE THIS CODE FOR TESTING
#Since we can't compare the model, its better to look at the its printed␣
 ↪structure and compare it with layers given in architecture
model = Net()
print(model)

model.apply(weights_init)
if torch.cuda.is_available():
    model.cuda()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.5)
```

```
Net(
  (model): Sequential(
    (0): Conv2d(1, 32, kernel_size=(5, 5), stride=(1, 1), padding=(1, 1))
    (1): ReLU()
    (2): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (3): ReLU()
    (4): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (5): Conv2d(64, 96, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (6): ReLU()
    (7): Conv2d(96, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (8): ReLU()
    (9): Conv2d(64, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (10): ReLU()
    (11): MaxPool2d(kernel_size=2, stride=1, padding=0, dilation=1,
ceil_mode=False)
    (12): Flatten(start_dim=1, end_dim=-1)
    (13): Dropout(p=0.5, inplace=False)
    (14): Linear(in_features=4608, out_features=2048, bias=True)
    (15): ReLU()
    (16): Linear(in_features=2048, out_features=1024, bias=True)
    (17): ReLU()
    (18): Linear(in_features=1024, out_features=10, bias=True)
  )
)
```

[26]:
```
# Main training routine
def train(epoch):
  model.train()
  # Get each
  for batch_idx, (data, target) in enumerate(train_loader_augmented):
    if torch.cuda.is_available():
      data, target = data.cuda(), target.cuda()
    data, target = Variable(data), Variable(target)
    optimizer.zero_grad()
    output = model(data)
    loss = F.cross_entropy(output, target)
    loss.backward()
    optimizer.step()
    # Store results
    if batch_idx % 10 == 0:
      print('Train Epoch: {} [{}/{}]\tLoss: {:.6f}'.format(
        epoch, batch_idx * len(data), len(train_loader_augmented.dataset), loss.
↪item()))


# Run on test data
```

```
def test():
  model.eval()
  test_loss = 0
  correct = 0
  for data, target in test_loader:
      if torch.cuda.is_available():
          data, target = data.cuda(), target.cuda()
      data, target = Variable(data, volatile=True), Variable(target)
      output = model(data)
      test_loss += F.cross_entropy(output, target, size_average=False).item()#␣
↪sum up batch loss
      pred = output.data.max(1, keepdim=True)[1]# get the index of the max␣
↪log-probability
      correct += pred.eq(target.data.view_as(pred)).long().cpu().sum()
  test_loss /= len(test_loader.dataset)
  correct=float(correct.to(torch.device('cpu')).numpy())
  print('\nTest set: Avg. loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
    test_loss, correct, len(test_loader.dataset),
    100. * correct / len(test_loader.dataset)))
```

[27]:
```
# Train for three epochs
n_epochs = 3
for epoch in range(1, n_epochs + 1):
  train(epoch)
  test()
```

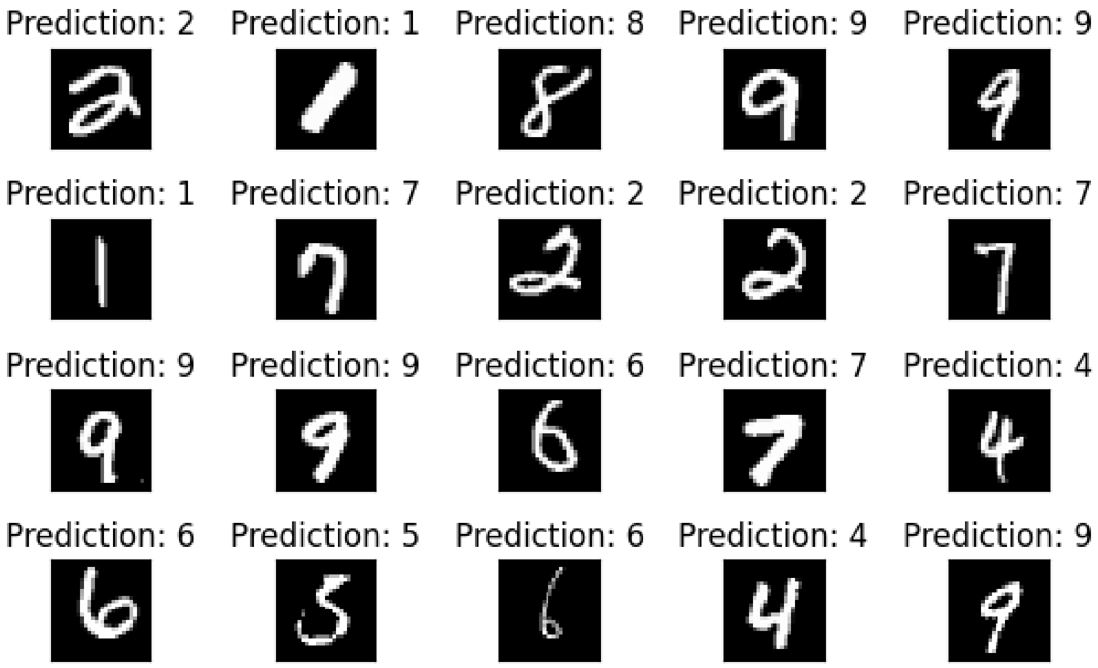
```
Train Epoch: 1 [0/90000]        Loss: 2.308034
Train Epoch: 1 [640/90000]      Loss: 2.308485
Train Epoch: 1 [1280/90000]     Loss: 2.298017
Train Epoch: 1 [1920/90000]     Loss: 2.295219
Train Epoch: 1 [2560/90000]     Loss: 2.288102
Train Epoch: 1 [3200/90000]     Loss: 2.265552
Train Epoch: 1 [3840/90000]     Loss: 2.259609
Train Epoch: 1 [4480/90000]     Loss: 2.199659
Train Epoch: 1 [5120/90000]     Loss: 2.118025
Train Epoch: 1 [5760/90000]     Loss: 1.947024
Train Epoch: 1 [6400/90000]     Loss: 1.500429
Train Epoch: 1 [7040/90000]     Loss: 1.356968
Train Epoch: 1 [7680/90000]     Loss: 1.195196
Train Epoch: 1 [8320/90000]     Loss: 1.080561
Train Epoch: 1 [8960/90000]     Loss: 0.769689
Train Epoch: 1 [9600/90000]     Loss: 0.888492
Train Epoch: 1 [10240/90000]    Loss: 0.760998
Train Epoch: 1 [10880/90000]    Loss: 0.742269
Train Epoch: 1 [11520/90000]    Loss: 0.641482
Train Epoch: 1 [12160/90000]    Loss: 0.422565
Train Epoch: 1 [12800/90000]    Loss: 0.466517
Train Epoch: 1 [13440/90000]    Loss: 0.546700
Train Epoch: 1 [14080/90000]    Loss: 0.497548
Train Epoch: 1 [14720/90000]    Loss: 0.653372
Train Epoch: 1 [15360/90000]    Loss: 0.350555
Train Epoch: 1 [16000/90000]    Loss: 0.505466
Train Epoch: 1 [16640/90000]    Loss: 0.556982
Train Epoch: 1 [17280/90000]    Loss: 0.430359
Train Epoch: 1 [17920/90000]    Loss: 0.397817
Train Epoch: 1 [18560/90000]    Loss: 0.395571
Train Epoch: 1 [19200/90000]    Loss: 0.463087
Train Epoch: 1 [19840/90000]    Loss: 0.297907
Train Epoch: 1 [20480/90000]    Loss: 0.291638
Train Epoch: 1 [21120/90000]    Loss: 0.447677
Train Epoch: 1 [21760/90000]    Loss: 0.420550
Train Epoch: 1 [22400/90000]    Loss: 0.432241
Train Epoch: 1 [23040/90000]    Loss: 0.405439
Train Epoch: 1 [23680/90000]    Loss: 0.216314
Train Epoch: 1 [24320/90000]    Loss: 0.460079
Train Epoch: 1 [24960/90000]    Loss: 0.345293
Train Epoch: 1 [25600/90000]    Loss: 0.380422
Train Epoch: 1 [26240/90000]    Loss: 0.378678
Train Epoch: 1 [26880/90000]    Loss: 0.269789
Train Epoch: 1 [27520/90000]    Loss: 0.333817
Train Epoch: 1 [28160/90000]    Loss: 0.311567
Train Epoch: 1 [28800/90000]    Loss: 0.273063
Train Epoch: 1 [29440/90000]    Loss: 0.176317
Train Epoch: 1 [30080/90000]    Loss: 0.225385
Train Epoch: 1 [30720/90000]    Loss: 0.331343
Train Epoch: 1 [31360/90000]    Loss: 0.480726
Train Epoch: 1 [32000/90000]    Loss: 0.175652
Train Epoch: 1 [32640/90000]    Loss: 0.261497
Train Epoch: 1 [33280/90000]    Loss: 0.256476
Train Epoch: 1 [33920/90000]    Loss: 0.290802
Train Epoch: 1 [34560/90000]    Loss: 0.311080
Train Epoch: 1 [35200/90000]    Loss: 0.347536
Train Epoch: 1 [35840/90000]    Loss: 0.219922
Train Epoch: 1 [36480/90000]    Loss: 0.339675
Train Epoch: 1 [37120/90000]    Loss: 0.365372
Train Epoch: 1 [37760/90000]    Loss: 0.151710
Train Epoch: 1 [38400/90000]    Loss: 0.349802
Train Epoch: 1 [39040/90000]    Loss: 0.322032
Train Epoch: 1 [39680/90000]    Loss: 0.246024
Train Epoch: 1 [40320/90000]    Loss: 0.327061
Train Epoch: 1 [40960/90000]    Loss: 0.269655
Train Epoch: 1 [41600/90000]    Loss: 0.235560
Train Epoch: 1 [42240/90000]    Loss: 0.222073
Train Epoch: 1 [42880/90000]    Loss: 0.255770
Train Epoch: 1 [43520/90000]    Loss: 0.129676
Train Epoch: 1 [44160/90000]    Loss: 0.373548
Train Epoch: 1 [44800/90000]    Loss: 0.193603
Train Epoch: 1 [45440/90000]    Loss: 0.207721
Train Epoch: 1 [46080/90000]    Loss: 0.273420
Train Epoch: 1 [46720/90000]    Loss: 0.261993
Train Epoch: 1 [47360/90000]    Loss: 0.123231
Train Epoch: 1 [48000/90000]    Loss: 0.218159
Train Epoch: 1 [48640/90000]    Loss: 0.233426
Train Epoch: 1 [49280/90000]    Loss: 0.174329
Train Epoch: 1 [49920/90000]    Loss: 0.235424
Train Epoch: 1 [50560/90000]    Loss: 0.177412
Train Epoch: 1 [51200/90000]    Loss: 0.099113
Train Epoch: 1 [51840/90000]    Loss: 0.348885
Train Epoch: 1 [52480/90000]    Loss: 0.151901
Train Epoch: 1 [53120/90000]    Loss: 0.194500
Train Epoch: 1 [53760/90000]    Loss: 0.224197
Train Epoch: 1 [54400/90000]    Loss: 0.320250
Train Epoch: 1 [55040/90000]    Loss: 0.118888
Train Epoch: 1 [55680/90000]    Loss: 0.203010
Train Epoch: 1 [56320/90000]    Loss: 0.202817
Train Epoch: 1 [56960/90000]    Loss: 0.257999
Train Epoch: 1 [57600/90000]    Loss: 0.134775
Train Epoch: 1 [58240/90000]    Loss: 0.264640
Train Epoch: 1 [58880/90000]    Loss: 0.207574
Train Epoch: 1 [59520/90000]    Loss: 0.217817
Train Epoch: 1 [60160/90000]    Loss: 0.236570
Train Epoch: 1 [60800/90000]    Loss: 0.331584
Train Epoch: 1 [61440/90000]    Loss: 0.159875
Train Epoch: 1 [62080/90000]    Loss: 0.217499
Train Epoch: 1 [62720/90000]    Loss: 0.368566
Train Epoch: 1 [63360/90000]    Loss: 0.113405
Train Epoch: 1 [64000/90000]    Loss: 0.170341
Train Epoch: 1 [64640/90000]    Loss: 0.109500
Train Epoch: 1 [65280/90000]    Loss: 0.177127
Train Epoch: 1 [65920/90000]    Loss: 0.335601
Train Epoch: 1 [66560/90000]    Loss: 0.249785
Train Epoch: 1 [67200/90000]    Loss: 0.116247
Train Epoch: 1 [67840/90000]    Loss: 0.238111
Train Epoch: 1 [68480/90000]    Loss: 0.305635
Train Epoch: 1 [69120/90000]    Loss: 0.188557
Train Epoch: 1 [69760/90000]    Loss: 0.146206
Train Epoch: 1 [70400/90000]    Loss: 0.212445
Train Epoch: 1 [71040/90000]    Loss: 0.118737
Train Epoch: 1 [71680/90000]    Loss: 0.123492
Train Epoch: 1 [72320/90000]    Loss: 0.119601
Train Epoch: 1 [72960/90000]    Loss: 0.195817
Train Epoch: 1 [73600/90000]    Loss: 0.239527
Train Epoch: 1 [74240/90000]    Loss: 0.158668
Train Epoch: 1 [74880/90000]    Loss: 0.042520
Train Epoch: 1 [75520/90000]    Loss: 0.393896
Train Epoch: 1 [76160/90000]    Loss: 0.074459
Train Epoch: 1 [76800/90000]    Loss: 0.233111
Train Epoch: 1 [77440/90000]    Loss: 0.395271
Train Epoch: 1 [78080/90000]    Loss: 0.318348
Train Epoch: 1 [78720/90000]    Loss: 0.105807
Train Epoch: 1 [79360/90000]    Loss: 0.347427
Train Epoch: 1 [80000/90000]    Loss: 0.160854
Train Epoch: 1 [80640/90000]    Loss: 0.183263
Train Epoch: 1 [81280/90000]    Loss: 0.123769
Train Epoch: 1 [81920/90000]    Loss: 0.414839
Train Epoch: 1 [82560/90000]    Loss: 0.369157
Train Epoch: 1 [83200/90000]    Loss: 0.161574
Train Epoch: 1 [83840/90000]    Loss: 0.203792
Train Epoch: 1 [84480/90000]    Loss: 0.116824
Train Epoch: 1 [85120/90000]    Loss: 0.163114
Train Epoch: 1 [85760/90000]    Loss: 0.216042
Train Epoch: 1 [86400/90000]    Loss: 0.180488
Train Epoch: 1 [87040/90000]    Loss: 0.145792
Train Epoch: 1 [87680/90000]    Loss: 0.081125
Train Epoch: 1 [88320/90000]    Loss: 0.243844
Train Epoch: 1 [88960/90000]    Loss: 0.122226
Train Epoch: 1 [89600/90000]    Loss: 0.188743

/tmp/ipykernel_362210/4254681509.py:28: UserWarning: volatile was removed and
now has no effect. Use `with torch.no_grad():` instead.
  data, target = Variable(data, volatile=True), Variable(target)
/home/alimuhammad/.local/lib/python3.10/site-packages/torch/nn/_reduction.py:51:
UserWarning: size_average and reduce args will be deprecated, please use
reduction='sum' instead.
  warnings.warn(warning.format(ret))


Test set: Avg. loss: 0.0734, Accuracy: 9746.0/10000 (97%)

Train Epoch: 2 [0/90000]        Loss: 0.068878
Train Epoch: 2 [640/90000]      Loss: 0.067305
Train Epoch: 2 [1280/90000]     Loss: 0.052763
Train Epoch: 2 [1920/90000]     Loss: 0.117778
Train Epoch: 2 [2560/90000]     Loss: 0.128157
Train Epoch: 2 [3200/90000]     Loss: 0.215850
Train Epoch: 2 [3840/90000]     Loss: 0.094536
Train Epoch: 2 [4480/90000]     Loss: 0.097020
Train Epoch: 2 [5120/90000]     Loss: 0.063844
Train Epoch: 2 [5760/90000]     Loss: 0.078291
Train Epoch: 2 [6400/90000]     Loss: 0.062068
Train Epoch: 2 [7040/90000]     Loss: 0.286543
Train Epoch: 2 [7680/90000]     Loss: 0.267243
Train Epoch: 2 [8320/90000]     Loss: 0.097589
Train Epoch: 2 [8960/90000]     Loss: 0.113585
Train Epoch: 2 [9600/90000]     Loss: 0.136737
Train Epoch: 2 [10240/90000]    Loss: 0.118621
Train Epoch: 2 [10880/90000]    Loss: 0.235140
Train Epoch: 2 [11520/90000]    Loss: 0.209516
Train Epoch: 2 [12160/90000]    Loss: 0.170564
Train Epoch: 2 [12800/90000]    Loss: 0.173359
Train Epoch: 2 [13440/90000]    Loss: 0.078621
Train Epoch: 2 [14080/90000]    Loss: 0.020001
Train Epoch: 2 [14720/90000]    Loss: 0.065883
Train Epoch: 2 [15360/90000]    Loss: 0.212008
Train Epoch: 2 [16000/90000]    Loss: 0.101643
Train Epoch: 2 [16640/90000]    Loss: 0.129164
Train Epoch: 2 [17280/90000]    Loss: 0.073127
Train Epoch: 2 [17920/90000]    Loss: 0.034642
Train Epoch: 2 [18560/90000]    Loss: 0.096963
Train Epoch: 2 [19200/90000]    Loss: 0.220791
Train Epoch: 2 [19840/90000]    Loss: 0.228345
Train Epoch: 2 [20480/90000]    Loss: 0.439520
Train Epoch: 2 [21120/90000]    Loss: 0.116295
Train Epoch: 2 [21760/90000]    Loss: 0.136141
Train Epoch: 2 [22400/90000]    Loss: 0.067529
Train Epoch: 2 [23040/90000]    Loss: 0.114530
Train Epoch: 2 [23680/90000]    Loss: 0.111623
Train Epoch: 2 [24320/90000]    Loss: 0.172743
Train Epoch: 2 [24960/90000]    Loss: 0.111428
Train Epoch: 2 [25600/90000]    Loss: 0.036784
Train Epoch: 2 [26240/90000]    Loss: 0.229353
Train Epoch: 2 [26880/90000]    Loss: 0.150450
Train Epoch: 2 [27520/90000]    Loss: 0.160752
Train Epoch: 2 [28160/90000]    Loss: 0.118467
Train Epoch: 2 [28800/90000]    Loss: 0.100110
Train Epoch: 2 [29440/90000]    Loss: 0.102075
Train Epoch: 2 [30080/90000]    Loss: 0.186587
Train Epoch: 2 [30720/90000]    Loss: 0.214511
Train Epoch: 2 [31360/90000]    Loss: 0.295471
Train Epoch: 2 [32000/90000]    Loss: 0.111761
Train Epoch: 2 [32640/90000]    Loss: 0.112724
Train Epoch: 2 [33280/90000]    Loss: 0.110761
Train Epoch: 2 [33920/90000]    Loss: 0.149322
Train Epoch: 2 [34560/90000]    Loss: 0.045225
Train Epoch: 2 [35200/90000]    Loss: 0.201817
Train Epoch: 2 [35840/90000]    Loss: 0.068724
Train Epoch: 2 [36480/90000]    Loss: 0.157533
Train Epoch: 2 [37120/90000]    Loss: 0.043342
Train Epoch: 2 [37760/90000]    Loss: 0.061237
Train Epoch: 2 [38400/90000]    Loss: 0.132880
Train Epoch: 2 [39040/90000]    Loss: 0.092692
Train Epoch: 2 [39680/90000]    Loss: 0.084062
Train Epoch: 2 [40320/90000]    Loss: 0.020567
Train Epoch: 2 [40960/90000]    Loss: 0.056668
Train Epoch: 2 [41600/90000]    Loss: 0.271939
Train Epoch: 2 [42240/90000]    Loss: 0.146541
Train Epoch: 2 [42880/90000]    Loss: 0.103050
Train Epoch: 2 [43520/90000]    Loss: 0.134079
Train Epoch: 2 [44160/90000]    Loss: 0.104763
Train Epoch: 2 [44800/90000]    Loss: 0.181249
Train Epoch: 2 [45440/90000]    Loss: 0.117213
Train Epoch: 2 [46080/90000]    Loss: 0.088903
Train Epoch: 2 [46720/90000]    Loss: 0.067394
Train Epoch: 2 [47360/90000]    Loss: 0.146856
Train Epoch: 2 [48000/90000]    Loss: 0.096873
Train Epoch: 2 [48640/90000]    Loss: 0.104349
Train Epoch: 2 [49280/90000]    Loss: 0.089979
Train Epoch: 2 [49920/90000]    Loss: 0.031521
Train Epoch: 2 [50560/90000]    Loss: 0.173391
Train Epoch: 2 [51200/90000]    Loss: 0.148899
Train Epoch: 2 [51840/90000]    Loss: 0.131618
Train Epoch: 2 [52480/90000]    Loss: 0.031442
Train Epoch: 2 [53120/90000]    Loss: 0.033112
Train Epoch: 2 [53760/90000]    Loss: 0.167778
Train Epoch: 2 [54400/90000]    Loss: 0.145349
Train Epoch: 2 [55040/90000]    Loss: 0.169585
Train Epoch: 2 [55680/90000]    Loss: 0.055568
Train Epoch: 2 [56320/90000]    Loss: 0.104923
Train Epoch: 2 [56960/90000]    Loss: 0.014684
Train Epoch: 2 [57600/90000]    Loss: 0.117471
Train Epoch: 2 [58240/90000]    Loss: 0.019492
Train Epoch: 2 [58880/90000]    Loss: 0.103245
Train Epoch: 2 [59520/90000]    Loss: 0.118847
Train Epoch: 2 [60160/90000]    Loss: 0.267963
Train Epoch: 2 [60800/90000]    Loss: 0.092508
Train Epoch: 2 [61440/90000]    Loss: 0.055685
Train Epoch: 2 [62080/90000]    Loss: 0.233646
Train Epoch: 2 [62720/90000]    Loss: 0.227353
Train Epoch: 2 [63360/90000]    Loss: 0.102583
Train Epoch: 2 [64000/90000]    Loss: 0.195859
Train Epoch: 2 [64640/90000]    Loss: 0.060628
Train Epoch: 2 [65280/90000]    Loss: 0.196354
Train Epoch: 2 [65920/90000]    Loss: 0.133866
Train Epoch: 2 [66560/90000]    Loss: 0.115005
Train Epoch: 2 [67200/90000]    Loss: 0.122073
Train Epoch: 2 [67840/90000]    Loss: 0.038702
Train Epoch: 2 [68480/90000]    Loss: 0.198436
Train Epoch: 2 [69120/90000]    Loss: 0.066488
Train Epoch: 2 [69760/90000]    Loss: 0.108062
Train Epoch: 2 [70400/90000]    Loss: 0.098745
Train Epoch: 2 [71040/90000]    Loss: 0.088147
Train Epoch: 2 [71680/90000]    Loss: 0.161043
Train Epoch: 2 [72320/90000]    Loss: 0.075779
Train Epoch: 2 [72960/90000]    Loss: 0.105229
Train Epoch: 2 [73600/90000]    Loss: 0.099424
Train Epoch: 2 [74240/90000]    Loss: 0.114319
Train Epoch: 2 [74880/90000]    Loss: 0.040461
Train Epoch: 2 [75520/90000]    Loss: 0.090228
Train Epoch: 2 [76160/90000]    Loss: 0.082958
Train Epoch: 2 [76800/90000]    Loss: 0.061998
Train Epoch: 2 [77440/90000]    Loss: 0.088002
Train Epoch: 2 [78080/90000]    Loss: 0.026009
Train Epoch: 2 [78720/90000]    Loss: 0.115034
Train Epoch: 2 [79360/90000]    Loss: 0.024461
Train Epoch: 2 [80000/90000]    Loss: 0.103085
Train Epoch: 2 [80640/90000]    Loss: 0.050353
Train Epoch: 2 [81280/90000]    Loss: 0.268642
Train Epoch: 2 [81920/90000]    Loss: 0.047433
Train Epoch: 2 [82560/90000]    Loss: 0.051391
Train Epoch: 2 [83200/90000]    Loss: 0.089887
Train Epoch: 2 [83840/90000]    Loss: 0.032864
Train Epoch: 2 [84480/90000]    Loss: 0.092773
Train Epoch: 2 [85120/90000]    Loss: 0.115002
Train Epoch: 2 [85760/90000]    Loss: 0.033671
Train Epoch: 2 [86400/90000]    Loss: 0.113952
Train Epoch: 2 [87040/90000]    Loss: 0.067561
Train Epoch: 2 [87680/90000]    Loss: 0.029052
Train Epoch: 2 [88320/90000]    Loss: 0.123785
Train Epoch: 2 [88960/90000]    Loss: 0.017154
Train Epoch: 2 [89600/90000]    Loss: 0.180402

Test set: Avg. loss: 0.0452, Accuracy: 9839.0/10000 (98%)

Train Epoch: 3 [0/90000]        Loss: 0.199112
Train Epoch: 3 [640/90000]      Loss: 0.214362
Train Epoch: 3 [1280/90000]     Loss: 0.084238
Train Epoch: 3 [1920/90000]     Loss: 0.022361
Train Epoch: 3 [2560/90000]     Loss: 0.164655
Train Epoch: 3 [3200/90000]     Loss: 0.029917
Train Epoch: 3 [3840/90000]     Loss: 0.161255
Train Epoch: 3 [4480/90000]     Loss: 0.174170
Train Epoch: 3 [5120/90000]     Loss: 0.123900
Train Epoch: 3 [5760/90000]     Loss: 0.147428
Train Epoch: 3 [6400/90000]     Loss: 0.053407
Train Epoch: 3 [7040/90000]     Loss: 0.181701
Train Epoch: 3 [7680/90000]     Loss: 0.069408
Train Epoch: 3 [8320/90000]     Loss: 0.054643
Train Epoch: 3 [8960/90000]     Loss: 0.175799
Train Epoch: 3 [9600/90000]     Loss: 0.040410
Train Epoch: 3 [10240/90000]    Loss: 0.037370
Train Epoch: 3 [10880/90000]    Loss: 0.166872
Train Epoch: 3 [11520/90000]    Loss: 0.148745
Train Epoch: 3 [12160/90000]    Loss: 0.183185
Train Epoch: 3 [12800/90000]    Loss: 0.097090
Train Epoch: 3 [13440/90000]    Loss: 0.111757
Train Epoch: 3 [14080/90000]    Loss: 0.094408
Train Epoch: 3 [14720/90000]    Loss: 0.156184
Train Epoch: 3 [15360/90000]    Loss: 0.059159
Train Epoch: 3 [16000/90000]    Loss: 0.096948
Train Epoch: 3 [16640/90000]    Loss: 0.038741
Train Epoch: 3 [17280/90000]    Loss: 0.178502
Train Epoch: 3 [17920/90000]    Loss: 0.109068
Train Epoch: 3 [18560/90000]    Loss: 0.048863
Train Epoch: 3 [19200/90000]    Loss: 0.045937
Train Epoch: 3 [19840/90000]    Loss: 0.205490
Train Epoch: 3 [20480/90000]    Loss: 0.118918
Train Epoch: 3 [21120/90000]    Loss: 0.054073
Train Epoch: 3 [21760/90000]    Loss: 0.104540
Train Epoch: 3 [22400/90000]    Loss: 0.154882
Train Epoch: 3 [23040/90000]    Loss: 0.181812
Train Epoch: 3 [23680/90000]    Loss: 0.056970
Train Epoch: 3 [24320/90000]    Loss: 0.023050
Train Epoch: 3 [24960/90000]    Loss: 0.250016
Train Epoch: 3 [25600/90000]    Loss: 0.006840
Train Epoch: 3 [26240/90000]    Loss: 0.230728
Train Epoch: 3 [26880/90000]    Loss: 0.027718
Train Epoch: 3 [27520/90000]    Loss: 0.064739
Train Epoch: 3 [28160/90000]    Loss: 0.024682
Train Epoch: 3 [28800/90000]    Loss: 0.051892
Train Epoch: 3 [29440/90000]    Loss: 0.114173
Train Epoch: 3 [30080/90000]    Loss: 0.032590
Train Epoch: 3 [30720/90000]    Loss: 0.122763
Train Epoch: 3 [31360/90000]    Loss: 0.052390
Train Epoch: 3 [32000/90000]    Loss: 0.166578
Train Epoch: 3 [32640/90000]    Loss: 0.102438
Train Epoch: 3 [33280/90000]    Loss: 0.068186
Train Epoch: 3 [33920/90000]    Loss: 0.084678
Train Epoch: 3 [34560/90000]    Loss: 0.267233
Train Epoch: 3 [35200/90000]    Loss: 0.102274
Train Epoch: 3 [35840/90000]    Loss: 0.086288
Train Epoch: 3 [36480/90000]    Loss: 0.140451
Train Epoch: 3 [37120/90000]    Loss: 0.089043
Train Epoch: 3 [37760/90000]    Loss: 0.136031
Train Epoch: 3 [38400/90000]    Loss: 0.076625
Train Epoch: 3 [39040/90000]    Loss: 0.081578
Train Epoch: 3 [39680/90000]    Loss: 0.053034
Train Epoch: 3 [40320/90000]    Loss: 0.039292
Train Epoch: 3 [40960/90000]    Loss: 0.068801
Train Epoch: 3 [41600/90000]    Loss: 0.176284
Train Epoch: 3 [42240/90000]    Loss: 0.058028
Train Epoch: 3 [42880/90000]    Loss: 0.108183
Train Epoch: 3 [43520/90000]    Loss: 0.069697
Train Epoch: 3 [44160/90000]    Loss: 0.005465
Train Epoch: 3 [44800/90000]    Loss: 0.093693
Train Epoch: 3 [45440/90000]    Loss: 0.038506
Train Epoch: 3 [46080/90000]    Loss: 0.141356
Train Epoch: 3 [46720/90000]    Loss: 0.072124
Train Epoch: 3 [47360/90000]    Loss: 0.060886
Train Epoch: 3 [48000/90000]    Loss: 0.048905
Train Epoch: 3 [48640/90000]    Loss: 0.075555
Train Epoch: 3 [49280/90000]    Loss: 0.085731
Train Epoch: 3 [49920/90000]    Loss: 0.079661
Train Epoch: 3 [50560/90000]    Loss: 0.037915
Train Epoch: 3 [51200/90000]    Loss: 0.107092
Train Epoch: 3 [51840/90000]    Loss: 0.039508
Train Epoch: 3 [52480/90000]    Loss: 0.051764
Train Epoch: 3 [53120/90000]    Loss: 0.027844
Train Epoch: 3 [53760/90000]    Loss: 0.042032
Train Epoch: 3 [54400/90000]    Loss: 0.023755
Train Epoch: 3 [55040/90000]    Loss: 0.068358
Train Epoch: 3 [55680/90000]    Loss: 0.201794
Train Epoch: 3 [56320/90000]    Loss: 0.200838
Train Epoch: 3 [56960/90000]    Loss: 0.164726
Train Epoch: 3 [57600/90000]    Loss: 0.008208
Train Epoch: 3 [58240/90000]    Loss: 0.136378
Train Epoch: 3 [58880/90000]    Loss: 0.068137
Train Epoch: 3 [59520/90000]    Loss: 0.113546
Train Epoch: 3 [60160/90000]    Loss: 0.034170
Train Epoch: 3 [60800/90000]    Loss: 0.101486
Train Epoch: 3 [61440/90000]    Loss: 0.148505
Train Epoch: 3 [62080/90000]    Loss: 0.043182
Train Epoch: 3 [62720/90000]    Loss: 0.092053
Train Epoch: 3 [63360/90000]    Loss: 0.146804
Train Epoch: 3 [64000/90000]    Loss: 0.094422
Train Epoch: 3 [64640/90000]    Loss: 0.065239
Train Epoch: 3 [65280/90000]    Loss: 0.014523
Train Epoch: 3 [65920/90000]    Loss: 0.059713
Train Epoch: 3 [66560/90000]    Loss: 0.085893
Train Epoch: 3 [67200/90000]    Loss: 0.028421
Train Epoch: 3 [67840/90000]    Loss: 0.132131
Train Epoch: 3 [68480/90000]    Loss: 0.114995
Train Epoch: 3 [69120/90000]    Loss: 0.057998
Train Epoch: 3 [69760/90000]    Loss: 0.025162
Train Epoch: 3 [70400/90000]    Loss: 0.011313
Train Epoch: 3 [71040/90000]    Loss: 0.072324
Train Epoch: 3 [71680/90000]    Loss: 0.207378
Train Epoch: 3 [72320/90000]    Loss: 0.118700
Train Epoch: 3 [72960/90000]    Loss: 0.066082
Train Epoch: 3 [73600/90000]    Loss: 0.052762
Train Epoch: 3 [74240/90000]    Loss: 0.021948
Train Epoch: 3 [74880/90000]    Loss: 0.025099
Train Epoch: 3 [75520/90000]    Loss: 0.067755
Train Epoch: 3 [76160/90000]    Loss: 0.176652
Train Epoch: 3 [76800/90000]    Loss: 0.090222
Train Epoch: 3 [77440/90000]    Loss: 0.125110
Train Epoch: 3 [78080/90000]    Loss: 0.151199
Train Epoch: 3 [78720/90000]    Loss: 0.088645
Train Epoch: 3 [79360/90000]    Loss: 0.201782
Train Epoch: 3 [80000/90000]    Loss: 0.052343
Train Epoch: 3 [80640/90000]    Loss: 0.022871
Train Epoch: 3 [81280/90000]    Loss: 0.061482
Train Epoch: 3 [81920/90000]    Loss: 0.034449
Train Epoch: 3 [82560/90000]    Loss: 0.039309
Train Epoch: 3 [83200/90000]    Loss: 0.044597
Train Epoch: 3 [83840/90000]    Loss: 0.096269
Train Epoch: 3 [84480/90000]    Loss: 0.164087
Train Epoch: 3 [85120/90000]    Loss: 0.203054
Train Epoch: 3 [85760/90000]    Loss: 0.207474
Train Epoch: 3 [86400/90000]    Loss: 0.158605
Train Epoch: 3 [87040/90000]    Loss: 0.027835
Train Epoch: 3 [87680/90000]    Loss: 0.088856
Train Epoch: 3 [88320/90000]    Loss: 0.076620
Train Epoch: 3 [88960/90000]    Loss: 0.089814
Train Epoch: 3 [89600/90000]    Loss: 0.097386

Test set: Avg. loss: 0.0386, Accuracy: 9872.0/10000 (99%)
```

[28]:

```
# Run network on data we got before and show predictions
examples = enumerate(test_loader)
batch_idx, (example_data, example_target) = next(examples)
if torch.cuda.is_available():
  example_data, example_target = example_data.cuda(), example_target.cuda()
example_data, example_target = Variable(example_data, volatile=True),␣
 ↪Variable(example_target)
output = model(example_data)

fig = plt.figure()
for i in range(20):
  plt.subplot(5,5,i+1)
```

```
    plt.tight_layout()
    plt.imshow(example_data[i][0].cpu(), cmap='gray', interpolation='none')
    plt.title("Prediction: {}".format(
      output.data.max(1, keepdim=True)[1][i].item()))
    plt.xticks([])
    plt.yticks([])
plt.show()
```

```
/tmp/ipykernel_362210/2673449976.py:6: UserWarning: volatile was removed and now
has no effect. Use `with torch.no_grad():` instead.
  example_data, example_target = Variable(example_data, volatile=True),
Variable(example_target)
```



Data Augmentation: This makes the model more robust to variations in the image thus improving generalization.
Increased Epochs: This allows the model to learn more from the data since we increase the training passes, which often helps the model reach higher accuracy, although it may lead to overfitting.
Batch Size Change: Longer batches improve generalization but also increase load on resources such as memory which has slowed down training.