A3_Q1

August 4, 2023

1 Computer Vision 2023 Assignment 3: Deep Learning for Perception Tasks

This assignment contains 2 questions. The first question probes understanding of deep learning for classification. The second question is a more challenging classification experiment on a larger dataset. Answer the questions in separate Python notebooks.

1.1 Question 1: A simple classifier, 20 marks

For this exercise, we provide demo code showing how to train a network on a small dataset called Fashion-MNIST. Please run through the code "tutorial-style" to get a sense of what it is doing. Then use the code alongside lecture notes and other resources to understand how to use pytorch libraries to implement, train and use a neural network.

For the Fashion-MNIST dataset the lables from 0-9 correspond to various clothing classes so you might find it convenient to create a python list as follows:

class_names = ['T-shirt/top', 'Trouser', 'Pullover', 'Dress', 'Coat', 'Sandal', 'Shirt', 'Sneaker', 'Bag', 'Ankle boot']

You will need to answer various questions about the system, how it operates, the results of experiments with it and make modifications to it yourself. You can change the training scheme and the network structure.

Organize your own text and code cell to show the answer of each questions.

Detailed requirements:

Q1.1 (1 point)

Extract 3 images of different types of clothing from the training dataset, print out the size/shape of the training images, and display the three with their corresponding labels.

Q1.2 (2 points)

Run the training code for 10 epochs, for different values of the learning rate. Fill in the table below and plot the loss curves for each experiment:

Lr	Accuracy
1	19.8
0.1	87.3
0.01	83.2

Lr	Accuracy
0.001	71.1

Q1.3 (3 points)

Report the number of epochs when the accuracy reaches 85%. Fill in the table below and plot the loass curve for each experiment:

Lr	Accuracy	Epoch
1	10.0-20.7	Nan
0.1	85.0	4
0.01	85.2	17
0.001	85.0	120

Q1.4 (2 points)

Compare the results in table 1 and table 2, what is your observation and your understanding of learning rate?

Q1.5 (5 points)

Build a wider network by modifying the code that constructs the network so that the hidden layer(s) contain more perceptrons, and record the accuracy along with the number of trainable parameters in your model. Now modify the oroginal network to be deeper instead of wider (i.e. by adding more hidden layers). Record your accuracy and network size findings. Plot the loss curve for each experiment. Write down your conclusions about changing the network structure?

Structures	Accuracy	Parameters
Base	87.3	669706
Deeper	86.6	798474
Wider	87.5	1863690

Q1.6 (2 points)

Calculate the mean of the gradients of the loss to all trainable parameters. Plot the gradients curve for the first 100 training steps. What are your observations? Note that this gradients will be saved with the training weight automatically after you call loss.backwards(). Hint: the mean of the gradients decrease.

For more exlanation of q1.7, you could refer to the following simple instructions: https://colab.research.google.com/drive/1XAsyNegGSvMf3_B6MrsXht7-fHqtJ7OW?usp=sharing

Q1.7 (5 points)

Modify the network structure and training/test to use a small convolutional neural network instead of an MLP. Discuss your findings with rehgard to convergence, accuracy and number of parameters, relative to MLPs.

Hint: Look at the structure of the CNN in the Workshop 3 examples.

[36]:
```
# import numpy as np # This is for mathematical operations

# this is used in plotting
import matplotlib.pyplot as plt
import time
import pylab as pl
from IPython import display

%matplotlib inline

%load_ext autoreload
%autoreload 2
%reload_ext autoreload
```

```
The autoreload extension is already loaded. To reload it, use:
  %reload_ext autoreload
```

[2]:
```
#### Tutorial Code
####PyTorch has two primitives to work with data: torch.utils.data.DataLoader␣
 ↪and torch.utils.data.Dataset.
#####Dataset stores samples and their corresponding labels, and DataLoader␣
 ↪wraps an iterable around the Dataset.
import torch
from torch import nn
from torch.utils.data import DataLoader
from torchvision import datasets
from torchvision.transforms import ToTensor, Lambda, Compose
import matplotlib.pyplot as plt

# Download training data from open datasets.
##Every TorchVision Dataset includes two arguments:
##transform and target_transform to modify the samples and labels respectively.

training_data = datasets.FashionMNIST(
    root="data",
    train=True,
    download=True,
    transform=ToTensor(),
)

# Download test data from open datasets.
test_data = datasets.FashionMNIST(
    root="data",
    train=False,
    download=True,
```

```
    transform=ToTensor(),
)
```

NOTE: For consistency with the original data set, we call our validation data "test_data". It is important to keep in mind though that we are using the data for model validation and not for testing the final, trained model (which requires data not used when training the model parameters).

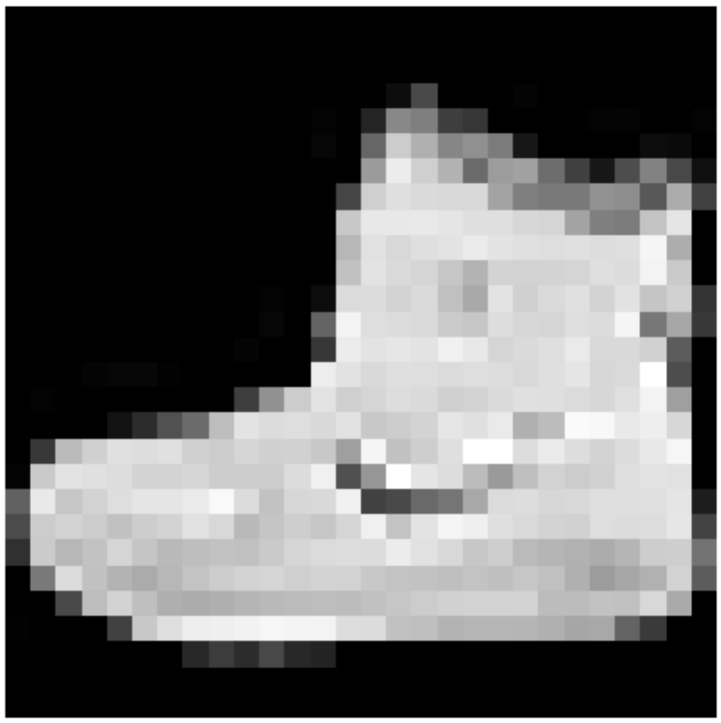
We pass the Dataset as an argument to DataLoader. This wraps an iterable over our dataset and supports automatic batching, sampling, shuffling, and multiprocess data loading. Here we define a batch size of 64, i.e. each element in the dataloader iterable will return a batch of 64 features and labels.

[146]:
```
batch_size = 64

# Create data loaders.
train_dataloader = DataLoader(training_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)

for X, y in test_dataloader:
    print("Shape of X [N, C, H, W]: ", X.shape)
    print("Shape of y: ", y.shape, y.dtype)
    break
```

```
Shape of X [N, C, H, W]:  torch.Size([64, 1, 28, 28])
Shape of y:  torch.Size([64]) torch.int64
```

Add in a code cell to inspect the training data, as per Q1.1. Each element of the training_data structure has a greyscale image (which you can use plt.imshow(img[0,:,:]) to display, just like you did in previous assignments.

[147]:
```
# Code cell for training image display
img_index = 0
img, label = training_data[img_index]
plt.imshow(img[0,:,:], cmap='gray')
plt.axis('off')
plt.show()
```



To define a neural network in PyTorch, we create a class that inherits from nn.Module. We define the layers of the network in the init function and specify how data will pass through the network in the forward function. To accelerate operations in the neural network, we move it to the GPU if available.

[99]:
```
# Get cpu or gpu device for training.
device = "cuda" if torch.cuda.is_available() else "cpu"
print("Using {} device".format(device))

import torch.nn.functional as F

# Define model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 10)
        )
```

```
    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)

        return logits

model = NeuralNetwork().to(device)
print(model)
total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print("Total Trainable Parameters:", total_params)
```

```
Using cpu device
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=784, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=10, bias=True)
  )
)
Total Trainable Parameters: 669706
```

[38]:
```
###Define the loss function and the optimizer
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1e-3)
```

In a single training loop, the model makes predictions on the training dataset (fed to it in batches), and backpropagates the prediction error to adjust the model's parameters.

[105]:
```
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # Compute prediction error
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch % 100 == 0:
```

```
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")
```

[106]:
```
##Define a test function
def test(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    model.eval()
    test_loss, correct = 0, 0
    with torch.no_grad():
        for X, y in dataloader:
            X, y = X.to(device), y.to(device)
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    test_loss /= num_batches
    correct /= size
    print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss:␣
 ↪{test_loss:>8f} \n")
    return test_loss,correct
```

[24]:
```
#Train and test the model
epochs = 5
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    test(test_dataloader, model, loss_fn)
print("Done!")
```

```
Epoch 1
-------------------------------
loss: 2.298520  [    0/60000]
loss: 2.285323  [ 6400/60000]
loss: 2.264351  [12800/60000]
loss: 2.266253  [19200/60000]
loss: 2.244990  [25600/60000]
loss: 2.225540  [32000/60000]
loss: 2.223236  [38400/60000]
loss: 2.193966  [44800/60000]
loss: 2.194777  [51200/60000]
loss: 2.167247  [57600/60000]
Test Error:
 Accuracy: 53.1%, Avg loss: 2.152243

Epoch 2
-------------------------------
loss: 2.160160  [    0/60000]
```

```
loss: 2.147828  [ 6400/60000]
loss: 2.089111  [12800/60000]
loss: 2.118133  [19200/60000]
loss: 2.060937  [25600/60000]
loss: 2.002837  [32000/60000]
loss: 2.029651  [38400/60000]
loss: 1.950569  [44800/60000]
loss: 1.968227  [51200/60000]
loss: 1.900613  [57600/60000]
Test Error:
 Accuracy: 60.6%, Avg loss: 1.886415

Epoch 3
-------------------------------
loss: 1.915884  [    0/60000]
loss: 1.882141  [ 6400/60000]
loss: 1.765858  [12800/60000]
loss: 1.822511  [19200/60000]
loss: 1.705983  [25600/60000]
loss: 1.654059  [32000/60000]
loss: 1.679516  [38400/60000]
loss: 1.577269  [44800/60000]
loss: 1.611212  [51200/60000]
loss: 1.509900  [57600/60000]
Test Error:
 Accuracy: 62.9%, Avg loss: 1.516664

Epoch 4
-------------------------------
loss: 1.579946  [    0/60000]
loss: 1.539382  [ 6400/60000]
loss: 1.390337  [12800/60000]
loss: 1.472165  [19200/60000]
loss: 1.347935  [25600/60000]
loss: 1.343577  [32000/60000]
loss: 1.352143  [38400/60000]
loss: 1.277868  [44800/60000]
loss: 1.319884  [51200/60000]
loss: 1.220875  [57600/60000]
Test Error:
 Accuracy: 64.2%, Avg loss: 1.242510

Epoch 5
-------------------------------
loss: 1.316898  [    0/60000]
loss: 1.294198  [ 6400/60000]
loss: 1.130843  [12800/60000]
loss: 1.242455  [19200/60000]
```

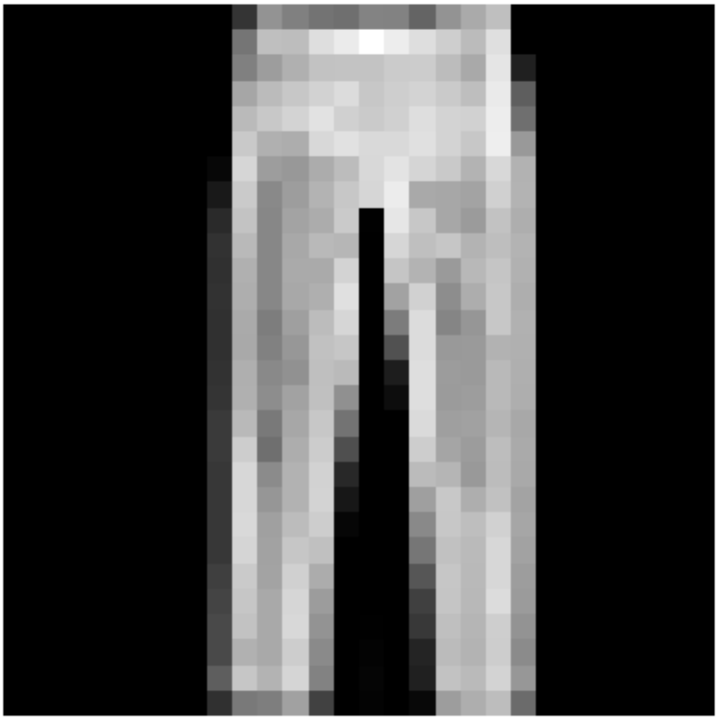
```
loss: 1.115808  [25600/60000]
loss: 1.142785  [32000/60000]
loss: 1.154951  [38400/60000]
loss: 1.094656  [44800/60000]
loss: 1.143121  [51200/60000]
loss: 1.058707  [57600/60000]
Test Error:
 Accuracy: 65.5%, Avg loss: 1.076736

Done!
```

[55]:
```python
#Code for question1.1
batch_size = 64
train_dataloader = DataLoader(training_data, batch_size=batch_size)
test_dataloader = DataLoader(test_data, batch_size=batch_size)
print("size:", len(training_data))
print("shape:", training_data.data.shape)
print("label_shape:", training_data.targets.shape)
print("label_number:", len(training_data.classes))
labels = training_data.targets
unique_labels = torch.unique(labels)
print(unique_labels)
select_labels=unique_labels[0:3]
print(select_labels)
images=[]
im1=[]
im2=[]
im3=[]
for i in range(len(training_data)):
    if training_data[i][1]==0:
        im1.append(training_data[i])
    elif training_data[i][1]==1:
        im2.append(training_data[i])
    elif training_data[i][1]==2:
        im3.append(training_data[i])
images.append(im1[0])
images.append(im2[0])
images.append(im3[0])
img_index = 0
img,label = images[img_index]
plt.imshow(img[0,:,:], cmap='gray')
plt.axis('off')
plt.show()
print(label)
img_index = 1
img, label = images[img_index]
plt.imshow(img[0,:,:], cmap='gray')
```

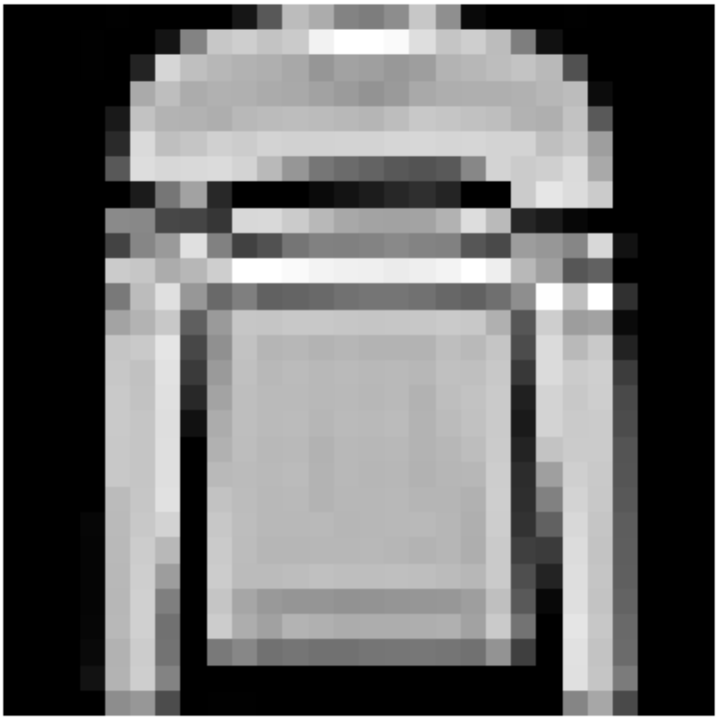
```
plt.axis('off')
plt.show()
print(label)
img_index = 2
img, label = images[img_index]
plt.imshow(img[0,:,:], cmap='gray')
plt.axis('off')
plt.show()
print(label)
```

```
size: 60000
shape: torch.Size([60000, 28, 28])
label_shape: torch.Size([60000])
label_number: 10
tensor([0, 1, 2, 3, 4, 5, 6, 7, 8, 9])
tensor([0, 1, 2])
```

```
0
```



1



2

[69]:
```
#code for question 1.2
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1)
epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss = test(test_dataloader, model, loss_fn)
    losses.append(loss)
```

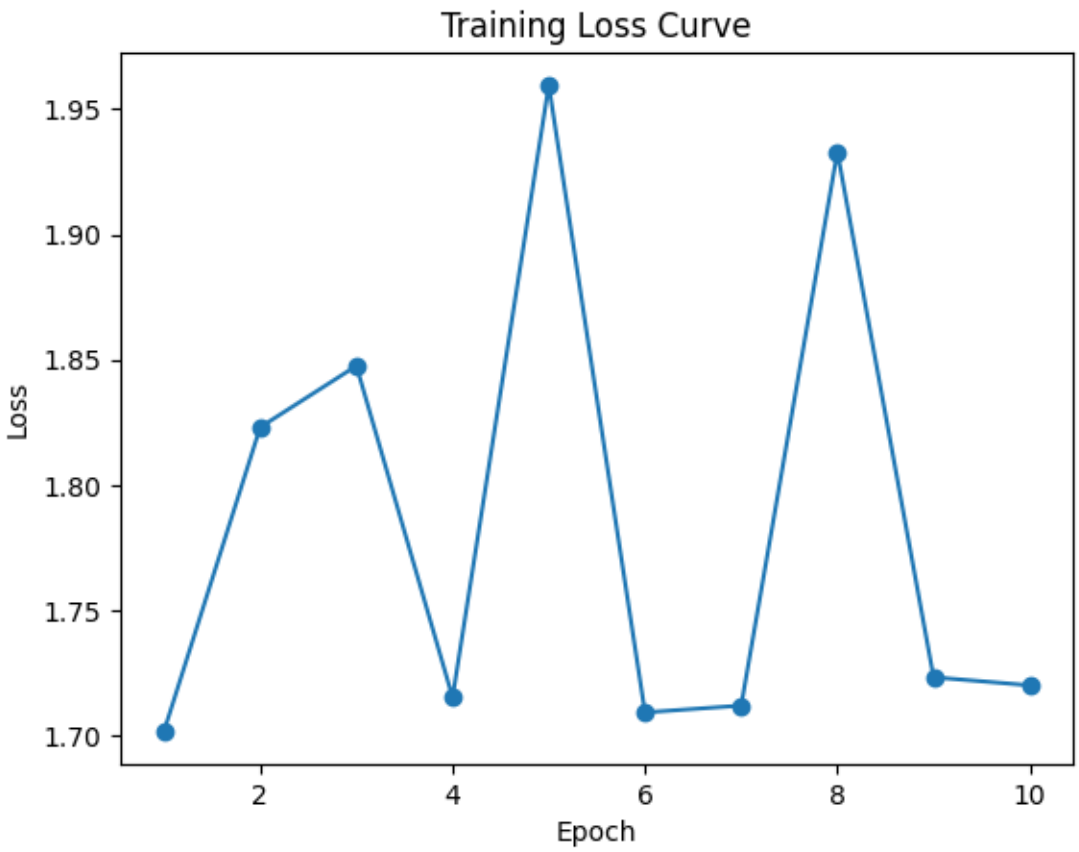
```
Epoch 1
-------------------------------
loss: 2.300789  [    0/60000]
loss: 3.457046  [ 6400/60000]
loss: 2.327987  [12800/60000]
loss: 1.858342  [19200/60000]
loss: 1.811761  [25600/60000]
loss: 1.724597  [32000/60000]
loss: 1.700926  [38400/60000]
loss: 1.711275  [44800/60000]
loss: 1.657892  [51200/60000]
loss: 1.689269  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.701855

Epoch 2
-------------------------------
loss: 1.680199  [    0/60000]
loss: 1.686081  [ 6400/60000]
loss: 1.739178  [12800/60000]
loss: 1.822188  [19200/60000]
loss: 1.772454  [25600/60000]
loss: 1.754710  [32000/60000]
loss: 1.705502  [38400/60000]
loss: 1.457183  [44800/60000]
loss: 1.649590  [51200/60000]
loss: 1.659614  [57600/60000]
Test Error:
 Accuracy: 20.1%, Avg loss: 1.823084

Epoch 3
-------------------------------
loss: 1.759024  [    0/60000]
```

```
loss: 1.643453  [ 6400/60000]
loss: 1.736741  [12800/60000]
loss: 1.668282  [19200/60000]
loss: 1.828320  [25600/60000]
loss: 1.774383  [32000/60000]
loss: 1.700145  [38400/60000]
loss: 2.300188  [44800/60000]
loss: 1.788137  [51200/60000]
loss: 1.752398  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.847590

Epoch 4
-------------------------------
loss: 1.830524  [    0/60000]
loss: 1.663223  [ 6400/60000]
loss: 1.768336  [12800/60000]
loss: 1.709029  [19200/60000]
loss: 1.814993  [25600/60000]
loss: 1.873284  [32000/60000]
loss: 1.883120  [38400/60000]
loss: 1.786241  [44800/60000]
loss: 1.680378  [51200/60000]
loss: 1.708153  [57600/60000]
Test Error:
 Accuracy: 19.8%, Avg loss: 1.715365

Epoch 5
-------------------------------
loss: 1.682933  [    0/60000]
loss: 1.645075  [ 6400/60000]
loss: 1.723709  [12800/60000]
loss: 1.665877  [19200/60000]
loss: 1.686074  [25600/60000]
loss: 1.763168  [32000/60000]
loss: 1.734288  [38400/60000]
loss: 1.711366  [44800/60000]
loss: 1.694515  [51200/60000]
loss: 2.027760  [57600/60000]
Test Error:
 Accuracy: 19.6%, Avg loss: 1.959574

Epoch 6
-------------------------------
loss: 1.878739  [    0/60000]
loss: 1.944532  [ 6400/60000]
loss: 1.919005  [12800/60000]
loss: 1.751726  [19200/60000]
```

```
loss: 1.658031  [25600/60000]
loss: 1.719165  [32000/60000]
loss: 1.695127  [38400/60000]
loss: 1.754496  [44800/60000]
loss: 1.722711  [51200/60000]
loss: 1.673069  [57600/60000]
Test Error:
 Accuracy: 19.8%, Avg loss: 1.709362

Epoch 7
-------------------------------
loss: 1.675520  [    0/60000]
loss: 1.687132  [ 6400/60000]
loss: 1.731127  [12800/60000]
loss: 1.689162  [19200/60000]
loss: 1.687062  [25600/60000]
loss: 1.749513  [32000/60000]
loss: 1.729985  [38400/60000]
loss: 1.847061  [44800/60000]
loss: 1.649827  [51200/60000]
loss: 1.659789  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.712082

Epoch 8
-------------------------------
loss: 1.722216  [    0/60000]
loss: 1.643319  [ 6400/60000]
loss: 1.731439  [12800/60000]
loss: 1.658665  [19200/60000]
loss: 1.695460  [25600/60000]
loss: 2.246026  [32000/60000]
loss: 1.916512  [38400/60000]
loss: 1.958946  [44800/60000]
loss: 1.825570  [51200/60000]
loss: 1.984891  [57600/60000]
Test Error:
 Accuracy: 19.4%, Avg loss: 1.932954

Epoch 9
-------------------------------
loss: 1.877149  [    0/60000]
loss: 1.852642  [ 6400/60000]
loss: 1.729550  [12800/60000]
loss: 1.679164  [19200/60000]
loss: 1.819057  [25600/60000]
loss: 1.739041  [32000/60000]
loss: 1.715823  [38400/60000]
```

```
loss: 1.809714  [44800/60000]
loss: 1.651994  [51200/60000]
loss: 1.671833  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.723379

Epoch 10
-------------------------------
loss: 1.727067  [    0/60000]
loss: 1.658970  [ 6400/60000]
loss: 1.770836  [12800/60000]
loss: 1.667042  [19200/60000]
loss: 1.682061  [25600/60000]
loss: 1.752351  [32000/60000]
loss: 1.679673  [38400/60000]
loss: 1.708115  [44800/60000]
loss: 1.654001  [51200/60000]
loss: 1.677871  [57600/60000]
Test Error:
 Accuracy: 19.8%, Avg loss: 1.720166
```

[71]:
```
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



[75]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss = test(test_dataloader, model, loss_fn)
    losses.append(loss)
```

```
Epoch 1
-------------------------------
loss: 2.308426  [    0/60000]
loss: 0.898664  [ 6400/60000]
loss: 0.574847  [12800/60000]
loss: 0.725962  [19200/60000]
loss: 0.592252  [25600/60000]
loss: 0.506215  [32000/60000]
loss: 0.537008  [38400/60000]
loss: 0.592733  [44800/60000]
```

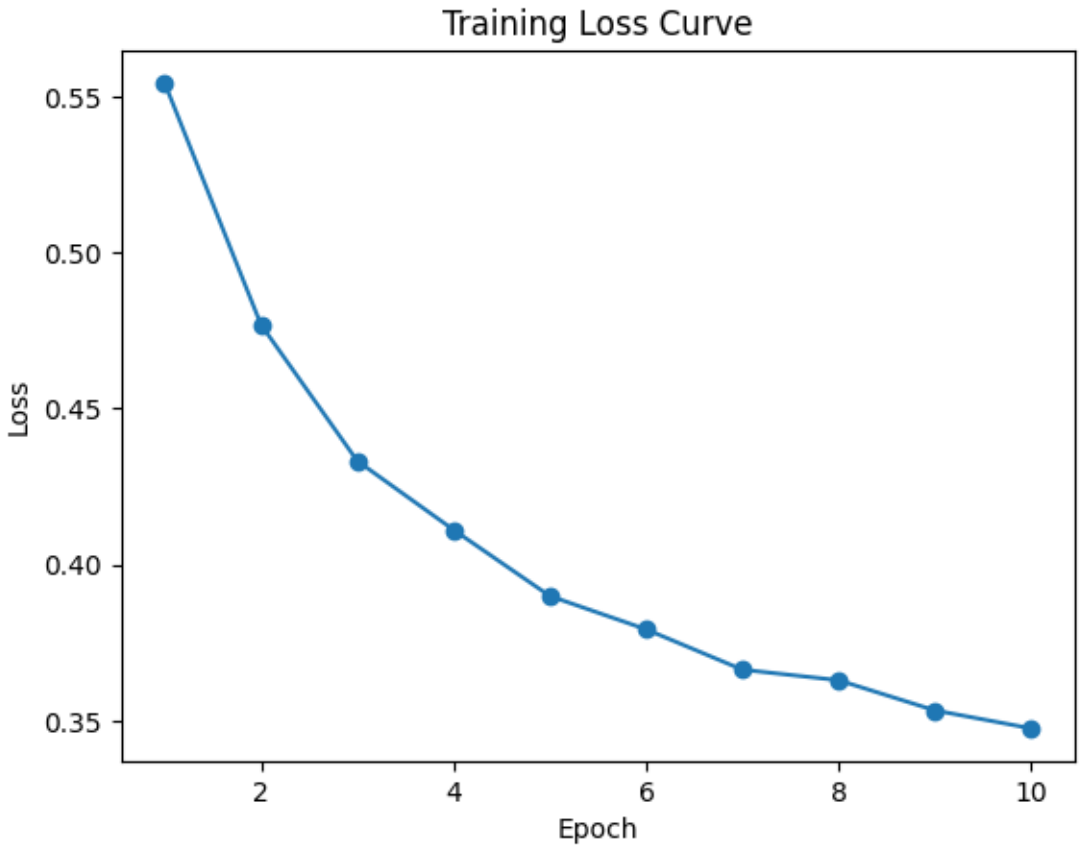
```
loss: 0.613278  [51200/60000]
loss: 0.451039  [57600/60000]
Test Error:
 Accuracy: 79.0%, Avg loss: 0.554276

Epoch 2
-------------------------------
loss: 0.437682  [    0/60000]
loss: 0.443061  [ 6400/60000]
loss: 0.373202  [12800/60000]
loss: 0.439612  [19200/60000]
loss: 0.401262  [25600/60000]
loss: 0.448174  [32000/60000]
loss: 0.408267  [38400/60000]
loss: 0.516229  [44800/60000]
loss: 0.510745  [51200/60000]
loss: 0.419899  [57600/60000]
Test Error:
 Accuracy: 81.9%, Avg loss: 0.476883

Epoch 3
-------------------------------
loss: 0.331300  [    0/60000]
loss: 0.357865  [ 6400/60000]
loss: 0.317665  [12800/60000]
loss: 0.360228  [19200/60000]
loss: 0.341693  [25600/60000]
loss: 0.418307  [32000/60000]
loss: 0.358192  [38400/60000]
loss: 0.470783  [44800/60000]
loss: 0.453340  [51200/60000]
loss: 0.408241  [57600/60000]
Test Error:
 Accuracy: 84.0%, Avg loss: 0.433235

Epoch 4
-------------------------------
loss: 0.261861  [    0/60000]
loss: 0.329416  [ 6400/60000]
loss: 0.275812  [12800/60000]
loss: 0.319904  [19200/60000]
loss: 0.319660  [25600/60000]
loss: 0.395287  [32000/60000]
loss: 0.328391  [38400/60000]
loss: 0.421554  [44800/60000]
loss: 0.420476  [51200/60000]
loss: 0.382431  [57600/60000]
Test Error:
```

```
 Accuracy: 85.0%, Avg loss: 0.411156

Epoch 5
-------------------------------
loss: 0.234672  [    0/60000]
loss: 0.306867  [ 6400/60000]
loss: 0.237342  [12800/60000]
loss: 0.292875  [19200/60000]
loss: 0.305020  [25600/60000]
loss: 0.375492  [32000/60000]
loss: 0.309667  [38400/60000]
loss: 0.387509  [44800/60000]
loss: 0.385809  [51200/60000]
loss: 0.372442  [57600/60000]
Test Error:
 Accuracy: 85.7%, Avg loss: 0.389956

Epoch 6
-------------------------------
loss: 0.216322  [    0/60000]
loss: 0.289744  [ 6400/60000]
loss: 0.210238  [12800/60000]
loss: 0.270142  [19200/60000]
loss: 0.293674  [25600/60000]
loss: 0.355884  [32000/60000]
loss: 0.289412  [38400/60000]
loss: 0.360666  [44800/60000]
loss: 0.366627  [51200/60000]
loss: 0.360900  [57600/60000]
Test Error:
 Accuracy: 86.3%, Avg loss: 0.379275

Epoch 7
-------------------------------
loss: 0.208407  [    0/60000]
loss: 0.281375  [ 6400/60000]
loss: 0.189392  [12800/60000]
loss: 0.262074  [19200/60000]
loss: 0.282607  [25600/60000]
loss: 0.347970  [32000/60000]
loss: 0.277682  [38400/60000]
loss: 0.339628  [44800/60000]
loss: 0.356043  [51200/60000]
loss: 0.347811  [57600/60000]
Test Error:
 Accuracy: 86.6%, Avg loss: 0.366351

Epoch 8
```

```
-------------------------------
loss: 0.193559  [    0/60000]
loss: 0.261782  [ 6400/60000]
loss: 0.170217  [12800/60000]
loss: 0.244251  [19200/60000]
loss: 0.279743  [25600/60000]
loss: 0.332650  [32000/60000]
loss: 0.271245  [38400/60000]
loss: 0.322689  [44800/60000]
loss: 0.331066  [51200/60000]
loss: 0.338936  [57600/60000]
Test Error:
 Accuracy: 86.6%, Avg loss: 0.362971

Epoch 9
-------------------------------
loss: 0.182403  [    0/60000]
loss: 0.257018  [ 6400/60000]
loss: 0.163267  [12800/60000]
loss: 0.238797  [19200/60000]
loss: 0.282550  [25600/60000]
loss: 0.326223  [32000/60000]
loss: 0.268751  [38400/60000]
loss: 0.314629  [44800/60000]
loss: 0.315116  [51200/60000]
loss: 0.315090  [57600/60000]
Test Error:
 Accuracy: 86.9%, Avg loss: 0.353302

Epoch 10
-------------------------------
loss: 0.181148  [    0/60000]
loss: 0.242100  [ 6400/60000]
loss: 0.157806  [12800/60000]
loss: 0.231158  [19200/60000]
loss: 0.270750  [25600/60000]
loss: 0.305989  [32000/60000]
loss: 0.251668  [38400/60000]
loss: 0.300570  [44800/60000]
loss: 0.296168  [51200/60000]
loss: 0.307082  [57600/60000]
Test Error:
 Accuracy: 87.3%, Avg loss: 0.347547
```

[76]:
```
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
```

```
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



[81]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss = test(test_dataloader, model, loss_fn)
    losses.append(loss)
```

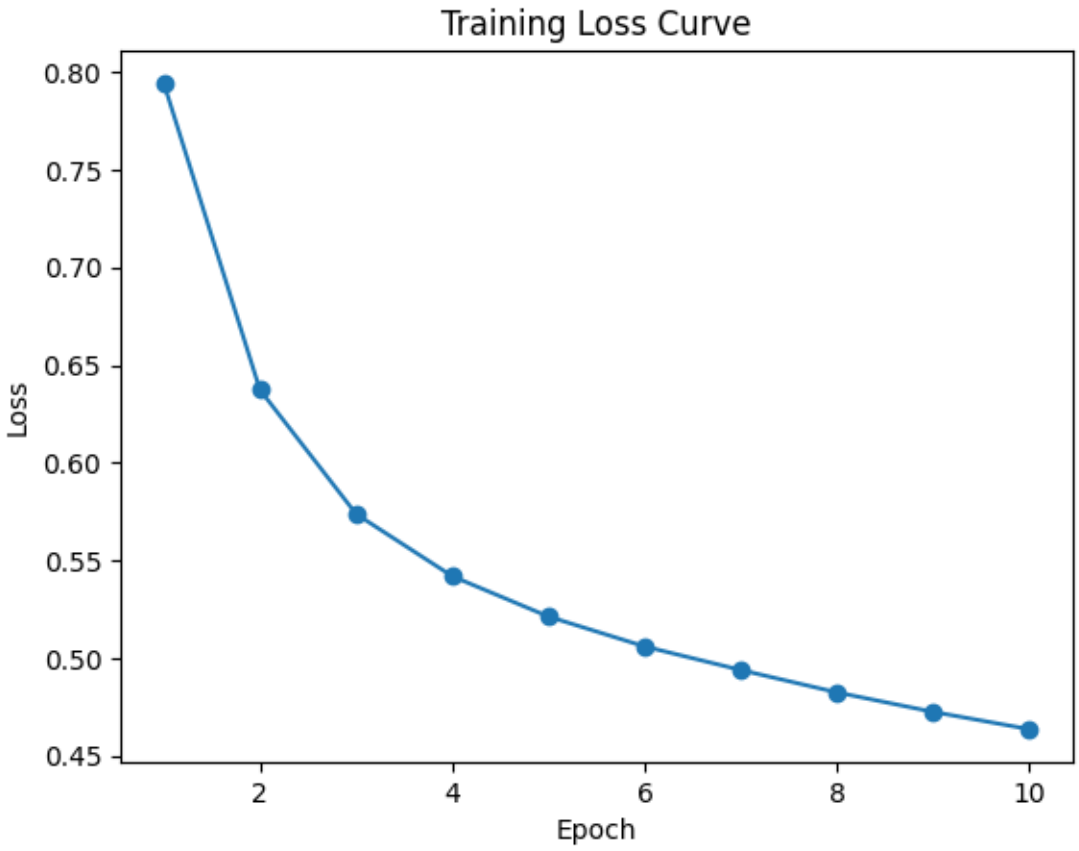
```
Epoch 1
-------------------------------
loss: 2.303613  [    0/60000]
loss: 2.163389  [ 6400/60000]
loss: 1.803041  [12800/60000]
loss: 1.505120  [19200/60000]
```

```
loss: 1.159484  [25600/60000]
loss: 1.055620  [32000/60000]
loss: 1.009546  [38400/60000]
loss: 0.864177  [44800/60000]
loss: 0.886052  [51200/60000]
loss: 0.808347  [57600/60000]
Test Error:
 Accuracy: 71.2%, Avg loss: 0.794302

Epoch 2
-------------------------------
loss: 0.795051  [    0/60000]
loss: 0.850702  [ 6400/60000]
loss: 0.584835  [12800/60000]
loss: 0.782208  [19200/60000]
loss: 0.665936  [25600/60000]
loss: 0.646174  [32000/60000]
loss: 0.720465  [38400/60000]
loss: 0.679714  [44800/60000]
loss: 0.709282  [51200/60000]
loss: 0.641554  [57600/60000]
Test Error:
 Accuracy: 78.1%, Avg loss: 0.637269

Epoch 3
-------------------------------
loss: 0.570789  [    0/60000]
loss: 0.660285  [ 6400/60000]
loss: 0.436693  [12800/60000]
loss: 0.666538  [19200/60000]
loss: 0.589266  [25600/60000]
loss: 0.568355  [32000/60000]
loss: 0.603899  [38400/60000]
loss: 0.634811  [44800/60000]
loss: 0.667913  [51200/60000]
loss: 0.550688  [57600/60000]
Test Error:
 Accuracy: 79.8%, Avg loss: 0.573706

Epoch 4
-------------------------------
loss: 0.482038  [    0/60000]
loss: 0.570894  [ 6400/60000]
loss: 0.377980  [12800/60000]
loss: 0.600954  [19200/60000]
loss: 0.538680  [25600/60000]
loss: 0.528707  [32000/60000]
loss: 0.550011  [38400/60000]
```

```
loss: 0.635669  [44800/60000]
loss: 0.644615  [51200/60000]
loss: 0.486820  [57600/60000]
Test Error:
 Accuracy: 80.5%, Avg loss: 0.541850

Epoch 5
-------------------------------
loss: 0.428624  [    0/60000]
loss: 0.524471  [ 6400/60000]
loss: 0.345209  [12800/60000]
loss: 0.556315  [19200/60000]
loss: 0.495258  [25600/60000]
loss: 0.497748  [32000/60000]
loss: 0.517756  [38400/60000]
loss: 0.634752  [44800/60000]
loss: 0.618868  [51200/60000]
loss: 0.447586  [57600/60000]
Test Error:
 Accuracy: 81.1%, Avg loss: 0.521203

Epoch 6
-------------------------------
loss: 0.389048  [    0/60000]
loss: 0.499161  [ 6400/60000]
loss: 0.321507  [12800/60000]
loss: 0.526159  [19200/60000]
loss: 0.465246  [25600/60000]
loss: 0.475726  [32000/60000]
loss: 0.493556  [38400/60000]
loss: 0.624921  [44800/60000]
loss: 0.595440  [51200/60000]
loss: 0.425816  [57600/60000]
Test Error:
 Accuracy: 81.6%, Avg loss: 0.505882

Epoch 7
-------------------------------
loss: 0.358636  [    0/60000]
loss: 0.481091  [ 6400/60000]
loss: 0.303705  [12800/60000]
loss: 0.505564  [19200/60000]
loss: 0.444012  [25600/60000]
loss: 0.459272  [32000/60000]
loss: 0.473917  [38400/60000]
loss: 0.611162  [44800/60000]
loss: 0.574476  [51200/60000]
loss: 0.413938  [57600/60000]
```

```
Test Error:
 Accuracy: 82.1%, Avg loss: 0.493837

Epoch 8
-------------------------------
loss: 0.336137  [    0/60000]
loss: 0.465651  [ 6400/60000]
loss: 0.289802  [12800/60000]
loss: 0.490907  [19200/60000]
loss: 0.426680  [25600/60000]
loss: 0.446917  [32000/60000]
loss: 0.458003  [38400/60000]
loss: 0.596760  [44800/60000]
loss: 0.555667  [51200/60000]
loss: 0.407059  [57600/60000]
Test Error:
 Accuracy: 82.6%, Avg loss: 0.482376

Epoch 9
-------------------------------
loss: 0.318369  [    0/60000]
loss: 0.451863  [ 6400/60000]
loss: 0.279624  [12800/60000]
loss: 0.479955  [19200/60000]
loss: 0.410649  [25600/60000]
loss: 0.436213  [32000/60000]
loss: 0.443573  [38400/60000]
loss: 0.583998  [44800/60000]
loss: 0.540075  [51200/60000]
loss: 0.402055  [57600/60000]
Test Error:
 Accuracy: 82.8%, Avg loss: 0.472397

Epoch 10
-------------------------------
loss: 0.305233  [    0/60000]
loss: 0.438853  [ 6400/60000]
loss: 0.271566  [12800/60000]
loss: 0.468927  [19200/60000]
loss: 0.396333  [25600/60000]
loss: 0.426050  [32000/60000]
loss: 0.430541  [38400/60000]
loss: 0.572789  [44800/60000]
loss: 0.527023  [51200/60000]
loss: 0.398016  [57600/60000]
Test Error:
 Accuracy: 83.2%, Avg loss: 0.463546
```

[82]:
```
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



[86]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.001)
epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss = test(test_dataloader, model, loss_fn)
    losses.append(loss)
```

```
Epoch 1
-------------------------------
loss: 2.300165  [    0/60000]
loss: 2.284601  [ 6400/60000]
```

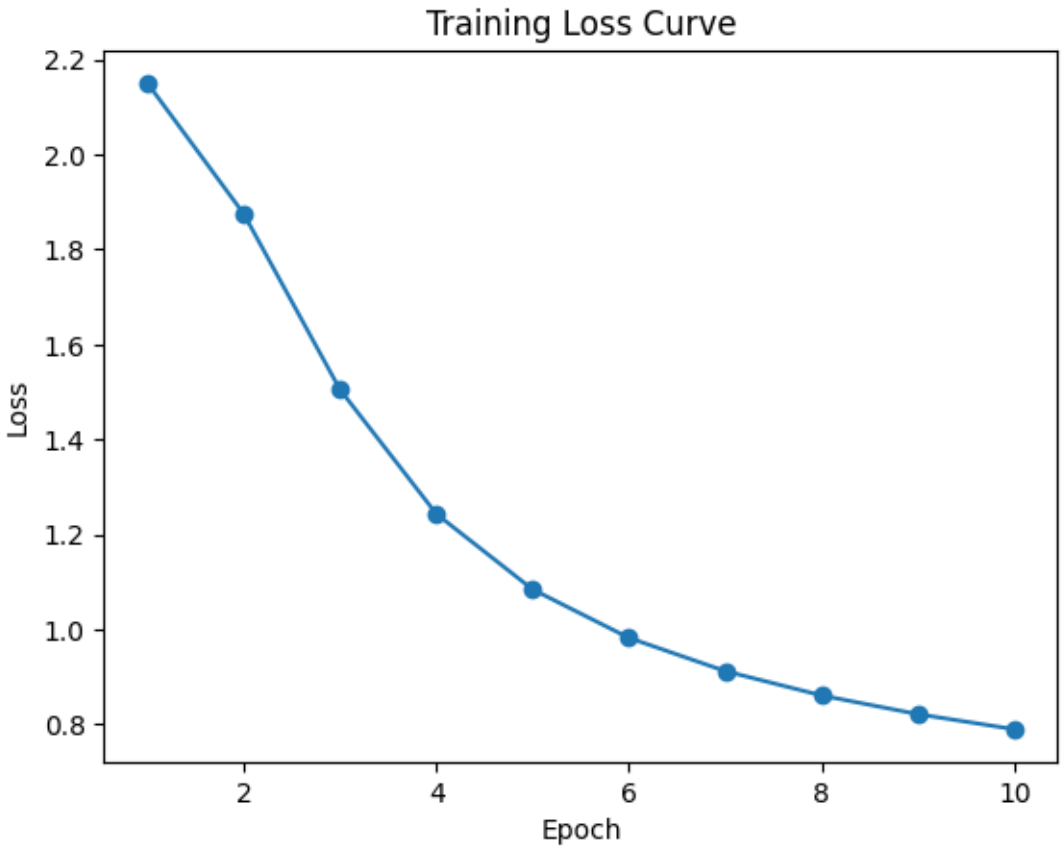
```
loss: 2.266656  [12800/60000]
loss: 2.265265  [19200/60000]
loss: 2.240025  [25600/60000]
loss: 2.213260  [32000/60000]
loss: 2.225255  [38400/60000]
loss: 2.187065  [44800/60000]
loss: 2.195795  [51200/60000]
loss: 2.152463  [57600/60000]
Test Error:
 Accuracy: 41.1%, Avg loss: 2.150631

Epoch 2
-------------------------------
loss: 2.160432  [    0/60000]
loss: 2.148810  [ 6400/60000]
loss: 2.093259  [12800/60000]
loss: 2.107784  [19200/60000]
loss: 2.054480  [25600/60000]
loss: 1.994823  [32000/60000]
loss: 2.017271  [38400/60000]
loss: 1.938818  [44800/60000]
loss: 1.959037  [51200/60000]
loss: 1.861998  [57600/60000]
Test Error:
 Accuracy: 59.7%, Avg loss: 1.875245

Epoch 3
-------------------------------
loss: 1.908182  [    0/60000]
loss: 1.876211  [ 6400/60000]
loss: 1.762657  [12800/60000]
loss: 1.794682  [19200/60000]
loss: 1.688977  [25600/60000]
loss: 1.641677  [32000/60000]
loss: 1.651636  [38400/60000]
loss: 1.561282  [44800/60000]
loss: 1.600421  [51200/60000]
loss: 1.467627  [57600/60000]
Test Error:
 Accuracy: 61.2%, Avg loss: 1.504714

Epoch 4
-------------------------------
loss: 1.570541  [    0/60000]
loss: 1.535538  [ 6400/60000]
loss: 1.388505  [12800/60000]
loss: 1.450739  [19200/60000]
loss: 1.338767  [25600/60000]
```

```
loss: 1.338099  [32000/60000]
loss: 1.345365  [38400/60000]
loss: 1.276167  [44800/60000]
loss: 1.325932  [51200/60000]
loss: 1.207770  [57600/60000]
Test Error:
 Accuracy: 62.8%, Avg loss: 1.242839

Epoch 5
-------------------------------
loss: 1.317291  [    0/60000]
loss: 1.300599  [ 6400/60000]
loss: 1.134154  [12800/60000]
loss: 1.234025  [19200/60000]
loss: 1.114428  [25600/60000]
loss: 1.141767  [32000/60000]
loss: 1.161572  [38400/60000]
loss: 1.100979  [44800/60000]
loss: 1.156883  [51200/60000]
loss: 1.057504  [57600/60000]
Test Error:
 Accuracy: 64.1%, Avg loss: 1.083607

Epoch 6
-------------------------------
loss: 1.150883  [    0/60000]
loss: 1.155872  [ 6400/60000]
loss: 0.971629  [12800/60000]
loss: 1.101893  [19200/60000]
loss: 0.981119  [25600/60000]
loss: 1.011628  [32000/60000]
loss: 1.049082  [38400/60000]
loss: 0.990642  [44800/60000]
loss: 1.047174  [51200/60000]
loss: 0.963555  [57600/60000]
Test Error:
 Accuracy: 65.6%, Avg loss: 0.981582

Epoch 7
-------------------------------
loss: 1.035656  [    0/60000]
loss: 1.062706  [ 6400/60000]
loss: 0.861439  [12800/60000]
loss: 1.014072  [19200/60000]
loss: 0.897225  [25600/60000]
loss: 0.919972  [32000/60000]
loss: 0.974701  [38400/60000]
loss: 0.918284  [44800/60000]
```

```
loss: 0.970642  [51200/60000]
loss: 0.899971  [57600/60000]
Test Error:
 Accuracy: 66.9%, Avg loss: 0.911452

Epoch 8
-------------------------------
loss: 0.950388  [    0/60000]
loss: 0.997575  [ 6400/60000]
loss: 0.781895  [12800/60000]
loss: 0.951483  [19200/60000]
loss: 0.840554  [25600/60000]
loss: 0.852631  [32000/60000]
loss: 0.921456  [38400/60000]
loss: 0.869384  [44800/60000]
loss: 0.914741  [51200/60000]
loss: 0.853118  [57600/60000]
Test Error:
 Accuracy: 68.2%, Avg loss: 0.860206

Epoch 9
-------------------------------
loss: 0.884402  [    0/60000]
loss: 0.947888  [ 6400/60000]
loss: 0.721671  [12800/60000]
loss: 0.904216  [19200/60000]
loss: 0.799193  [25600/60000]
loss: 0.801518  [32000/60000]
loss: 0.880405  [38400/60000]
loss: 0.834459  [44800/60000]
loss: 0.872352  [51200/60000]
loss: 0.816430  [57600/60000]
Test Error:
 Accuracy: 69.7%, Avg loss: 0.820743

Epoch 10
-------------------------------
loss: 0.831161  [    0/60000]
loss: 0.907281  [ 6400/60000]
loss: 0.674246  [12800/60000]
loss: 0.866956  [19200/60000]
loss: 0.767048  [25600/60000]
loss: 0.761892  [32000/60000]
loss: 0.846749  [38400/60000]
loss: 0.808322  [44800/60000]
loss: 0.839175  [51200/60000]
loss: 0.786322  [57600/60000]
Test Error:
```

```
Accuracy: 71.1%, Avg loss: 0.789007
```

[87]:
```python
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



[104]:
```python
#code for question 1.3
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=1)
epochs = 30
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss,accuracy = test(test_dataloader, model, loss_fn)
    losses.append(loss)
    if accuracy>=0.85:
```

```
        break
```

```
Epoch 1
-------------------------------
loss: 2.298406  [    0/60000]
loss: 1.939781  [ 6400/60000]
loss: 1.825169  [12800/60000]
loss: 1.454690  [19200/60000]
loss: 1.793104  [25600/60000]
loss: 1.890293  [32000/60000]
loss: 1.705379  [38400/60000]
loss: 1.625860  [44800/60000]
loss: 1.696273  [51200/60000]
loss: 1.631885  [57600/60000]
Test Error:
 Accuracy: 20.7%, Avg loss: 1.747973

Epoch 2
-------------------------------
loss: 1.785257  [    0/60000]
loss: 1.677019  [ 6400/60000]
loss: 1.790307  [12800/60000]
loss: 1.799545  [19200/60000]
loss: 1.448986  [25600/60000]
loss: 1.714263  [32000/60000]
loss: 1.544805  [38400/60000]
loss: 1.759948  [44800/60000]
loss: 1.663105  [51200/60000]
loss: 1.672362  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.704452

Epoch 3
-------------------------------
loss: 1.718217  [    0/60000]
loss: 1.670483  [ 6400/60000]
loss: 1.763575  [12800/60000]
loss: 1.678757  [19200/60000]
loss: 1.677581  [25600/60000]
loss: 1.750289  [32000/60000]
loss: 1.687610  [38400/60000]
loss: 1.698185  [44800/60000]
loss: 1.696303  [51200/60000]
loss: 1.713100  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.743979
```

```
Epoch 4
-------------------------------
loss: 1.780648  [    0/60000]
loss: 1.689393  [ 6400/60000]
loss: 1.752667  [12800/60000]
loss: 1.758458  [19200/60000]
loss: 1.670449  [25600/60000]
loss: 1.568082  [32000/60000]
loss: 1.513159  [38400/60000]
loss: 1.615601  [44800/60000]
loss: 1.684460  [51200/60000]
loss: 1.673936  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.740159

Epoch 5
-------------------------------
loss: 1.689200  [    0/60000]
loss: 1.687880  [ 6400/60000]
loss: 1.743864  [12800/60000]
loss: 1.784785  [19200/60000]
loss: 1.688761  [25600/60000]
loss: 1.785084  [32000/60000]
loss: 1.728136  [38400/60000]
loss: 1.771796  [44800/60000]
loss: 1.658046  [51200/60000]
loss: 1.688752  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.745869

Epoch 6
-------------------------------
loss: 1.688639  [    0/60000]
loss: 1.701255  [ 6400/60000]
loss: 1.877613  [12800/60000]
loss: 1.666364  [19200/60000]
loss: 1.661683  [25600/60000]
loss: 1.759628  [32000/60000]
loss: 1.714443  [38400/60000]
loss: 1.649155  [44800/60000]
loss: 1.649973  [51200/60000]
loss: 1.671591  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.711470

Epoch 7
-------------------------------
loss: 1.713050  [    0/60000]
```

```
loss: 1.640613  [ 6400/60000]
loss: 1.725815  [12800/60000]
loss: 1.670095  [19200/60000]
loss: 1.672201  [25600/60000]
loss: 1.761411  [32000/60000]
loss: 1.671356  [38400/60000]
loss: 1.706624  [44800/60000]
loss: 1.653449  [51200/60000]
loss: 1.668104  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.709821

Epoch 8
-------------------------------
loss: 1.713356  [    0/60000]
loss: 1.706185  [ 6400/60000]
loss: 1.727884  [12800/60000]
loss: 1.659283  [19200/60000]
loss: 1.691061  [25600/60000]
loss: 1.754264  [32000/60000]
loss: 1.720810  [38400/60000]
loss: 1.705221  [44800/60000]
loss: 1.651249  [51200/60000]
loss: 1.670937  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.707725

Epoch 9
-------------------------------
loss: 1.669325  [    0/60000]
loss: 1.682483  [ 6400/60000]
loss: 1.726396  [12800/60000]
loss: 1.722840  [19200/60000]
loss: 1.660793  [25600/60000]
loss: 1.709567  [32000/60000]
loss: 1.681348  [38400/60000]
loss: 1.703589  [44800/60000]
loss: 1.652000  [51200/60000]
loss: 1.660738  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.694646

Epoch 10
-------------------------------
loss: 1.708891  [    0/60000]
loss: 1.639471  [ 6400/60000]
loss: 1.770751  [12800/60000]
loss: 1.657450  [19200/60000]
```

```
loss: 1.684791  [25600/60000]
loss: 1.755225  [32000/60000]
loss: 1.771730  [38400/60000]
loss: 1.681075  [44800/60000]
loss: 1.668508  [51200/60000]
loss: 1.664998  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.698258

Epoch 11
-------------------------------
loss: 1.703451  [    0/60000]
loss: 1.811564  [ 6400/60000]
loss: 1.733862  [12800/60000]
loss: 1.658749  [19200/60000]
loss: 1.651561  [25600/60000]
loss: 1.746825  [32000/60000]
loss: 1.699062  [38400/60000]
loss: 1.779067  [44800/60000]
loss: 1.644208  [51200/60000]
loss: 1.667435  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.693273

Epoch 12
-------------------------------
loss: 1.671069  [    0/60000]
loss: 1.682264  [ 6400/60000]
loss: 1.731868  [12800/60000]
loss: 1.726760  [19200/60000]
loss: 1.714035  [25600/60000]
loss: 1.665361  [32000/60000]
loss: 1.666628  [38400/60000]
loss: 1.706369  [44800/60000]
loss: 1.653732  [51200/60000]
loss: 1.659009  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.691784

Epoch 13
-------------------------------
loss: 1.710839  [    0/60000]
loss: 1.681440  [ 6400/60000]
loss: 1.728851  [12800/60000]
loss: 1.665320  [19200/60000]
loss: 1.676002  [25600/60000]
loss: 1.713880  [32000/60000]
loss: 1.665727  [38400/60000]
```

```
loss: 1.695002  [44800/60000]
loss: 1.648698  [51200/60000]
loss: 1.653619  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.778445

Epoch 14
-------------------------------
loss: 1.842272  [    0/60000]
loss: 1.637040  [ 6400/60000]
loss: 1.727791  [12800/60000]
loss: 1.645502  [19200/60000]
loss: 1.677488  [25600/60000]
loss: 1.720343  [32000/60000]
loss: 1.675095  [38400/60000]
loss: 1.769130  [44800/60000]
loss: 1.649701  [51200/60000]
loss: 1.662604  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.694167

Epoch 15
-------------------------------
loss: 1.675404  [    0/60000]
loss: 1.645079  [ 6400/60000]
loss: 1.724476  [12800/60000]
loss: 1.656218  [19200/60000]
loss: 1.674983  [25600/60000]
loss: 1.717711  [32000/60000]
loss: 1.670728  [38400/60000]
loss: 1.762585  [44800/60000]
loss: 1.651156  [51200/60000]
loss: 1.665197  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.841616

Epoch 16
-------------------------------
loss: 1.822586  [    0/60000]
loss: 1.696973  [ 6400/60000]
loss: 1.760712  [12800/60000]
loss: 1.709503  [19200/60000]
loss: 1.638321  [25600/60000]
loss: 1.668630  [32000/60000]
loss: 1.677683  [38400/60000]
loss: 1.701306  [44800/60000]
loss: 1.651116  [51200/60000]
loss: 1.666376  [57600/60000]
```

```
Test Error:
 Accuracy: 20.0%, Avg loss: 1.688908

Epoch 17
-------------------------------
loss: 1.673806  [    0/60000]
loss: 1.681291  [ 6400/60000]
loss: 1.727407  [12800/60000]
loss: 1.664349  [19200/60000]
loss: 1.628668  [25600/60000]
loss: 1.712722  [32000/60000]
loss: 1.698511  [38400/60000]
loss: 1.708500  [44800/60000]
loss: 1.651562  [51200/60000]
loss: 1.657293  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.690290

Epoch 18
-------------------------------
loss: 1.670068  [    0/60000]
loss: 1.640297  [ 6400/60000]
loss: 1.734534  [12800/60000]
loss: 1.655419  [19200/60000]
loss: 1.647531  [25600/60000]
loss: 1.698123  [32000/60000]
loss: 1.667831  [38400/60000]
loss: 1.706499  [44800/60000]
loss: 1.648221  [51200/60000]
loss: 1.657268  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.690713

Epoch 19
-------------------------------
loss: 1.670227  [    0/60000]
loss: 1.630829  [ 6400/60000]
loss: 1.721807  [12800/60000]
loss: 1.656635  [19200/60000]
loss: 1.640148  [25600/60000]
loss: 1.663757  [32000/60000]
loss: 1.667933  [38400/60000]
loss: 1.704816  [44800/60000]
loss: 1.652229  [51200/60000]
loss: 1.659448  [57600/60000]
Test Error:
 Accuracy: 19.9%, Avg loss: 1.688684
```

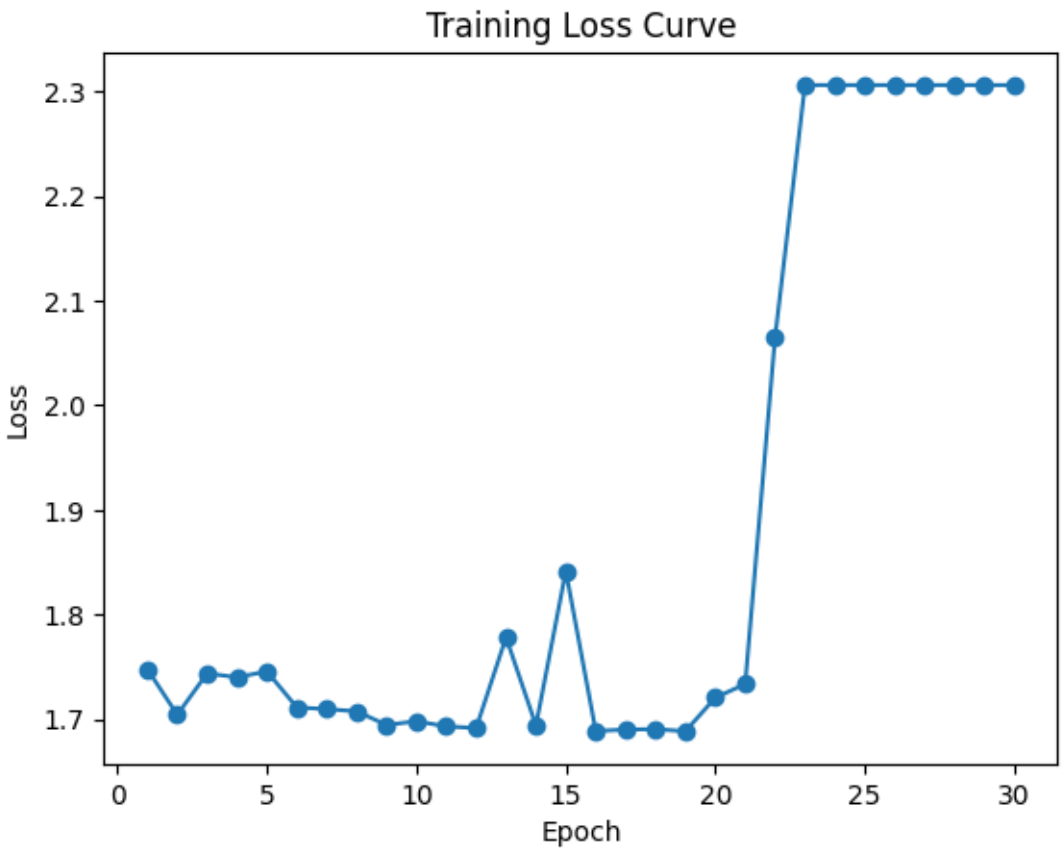
```
Epoch 20
-------------------------------
loss: 1.676083  [    0/60000]
loss: 1.675752  [ 6400/60000]
loss: 1.721833  [12800/60000]
loss: 1.666295  [19200/60000]
loss: 2.259532  [25600/60000]
loss: 1.739357  [32000/60000]
loss: 1.688744  [38400/60000]
loss: 1.713910  [44800/60000]
loss: 1.835145  [51200/60000]
loss: 1.694607  [57600/60000]
Test Error:
 Accuracy: 20.1%, Avg loss: 1.720727

Epoch 21
-------------------------------
loss: 1.806135  [    0/60000]
loss: 1.681937  [ 6400/60000]
loss: 1.749744  [12800/60000]
loss: 1.678025  [19200/60000]
loss: 1.643755  [25600/60000]
loss: 1.679477  [32000/60000]
loss: 1.672430  [38400/60000]
loss: 1.708060  [44800/60000]
loss: 1.649469  [51200/60000]
loss: 2.198649  [57600/60000]
Test Error:
 Accuracy: 20.0%, Avg loss: 1.733721

Epoch 22
-------------------------------
loss: 1.805484  [    0/60000]
loss: 1.689298  [ 6400/60000]
loss: 1.700085  [12800/60000]
loss: 1.746101  [19200/60000]
loss: 1.670526  [25600/60000]
loss: 1.721203  [32000/60000]
loss: 2.055944  [38400/60000]
loss: 2.126494  [44800/60000]
loss: 1.997346  [51200/60000]
loss: 2.066105  [57600/60000]
Test Error:
 Accuracy: 18.5%, Avg loss: 2.064946

Epoch 23
-------------------------------
loss: 2.100077  [    0/60000]
```

```
loss: 1.812885  [ 6400/60000]
loss: 1.896050  [12800/60000]
loss: 1.952951  [19200/60000]
loss: 2.106313  [25600/60000]
loss: 1.761659  [32000/60000]
loss: 2.307550  [38400/60000]
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845

Epoch 24
-------------------------------
loss: 2.313046  [    0/60000]
loss: 2.306795  [ 6400/60000]
loss: 2.312776  [12800/60000]
loss: 2.304621  [19200/60000]
loss: 2.281377  [25600/60000]
loss: 2.304950  [32000/60000]
loss: 2.305945  [38400/60000]
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845

Epoch 25
-------------------------------
loss: 2.313046  [    0/60000]
loss: 2.306795  [ 6400/60000]
loss: 2.312776  [12800/60000]
loss: 2.304621  [19200/60000]
loss: 2.281377  [25600/60000]
loss: 2.304950  [32000/60000]
loss: 2.305945  [38400/60000]
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845

Epoch 26
-------------------------------
loss: 2.313046  [    0/60000]
loss: 2.306795  [ 6400/60000]
loss: 2.312776  [12800/60000]
loss: 2.304621  [19200/60000]
```

```
loss: 2.281377  [25600/60000]
loss: 2.304950  [32000/60000]
loss: 2.305945  [38400/60000]
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845

Epoch 27
-------------------------------
loss: 2.313046  [    0/60000]
loss: 2.306795  [ 6400/60000]
loss: 2.312776  [12800/60000]
loss: 2.304621  [19200/60000]
loss: 2.281377  [25600/60000]
loss: 2.304950  [32000/60000]
loss: 2.305945  [38400/60000]
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845

Epoch 28
-------------------------------
loss: 2.313046  [    0/60000]
loss: 2.306795  [ 6400/60000]
loss: 2.312776  [12800/60000]
loss: 2.304621  [19200/60000]
loss: 2.281377  [25600/60000]
loss: 2.304950  [32000/60000]
loss: 2.305945  [38400/60000]
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845

Epoch 29
-------------------------------
loss: 2.313046  [    0/60000]
loss: 2.306795  [ 6400/60000]
loss: 2.312776  [12800/60000]
loss: 2.304621  [19200/60000]
loss: 2.281377  [25600/60000]
loss: 2.304950  [32000/60000]
loss: 2.305945  [38400/60000]
```

```
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845

Epoch 30
-------------------------------
loss: 2.313046  [    0/60000]
loss: 2.306795  [ 6400/60000]
loss: 2.312776  [12800/60000]
loss: 2.304621  [19200/60000]
loss: 2.281377  [25600/60000]
loss: 2.304950  [32000/60000]
loss: 2.305945  [38400/60000]
loss: 2.288610  [44800/60000]
loss: 2.300421  [51200/60000]
loss: 2.330204  [57600/60000]
Test Error:
 Accuracy: 10.0%, Avg loss: 2.305845
```

[105]:
```python
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



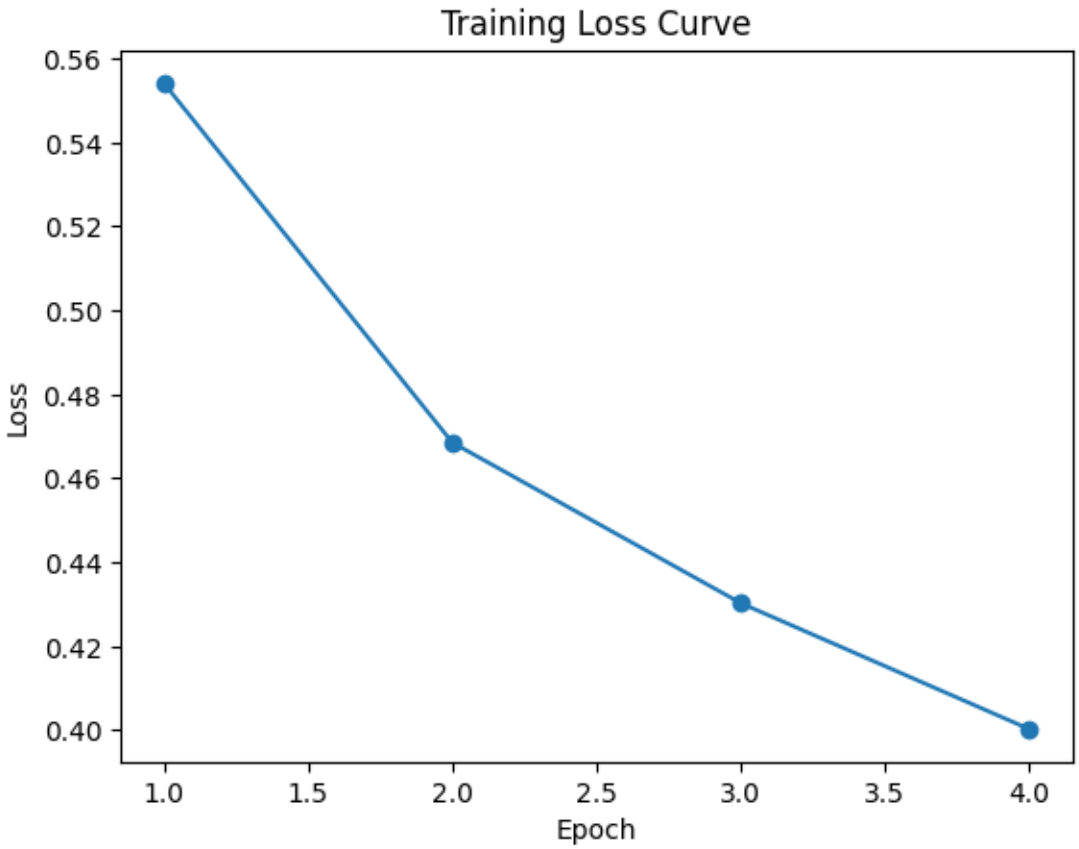
[109]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
epochs = 30
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss,accuracy = test(test_dataloader, model, loss_fn)
    losses.append(loss)
    if accuracy>=0.85:
        break
```

```
Epoch 1
-------------------------------
loss: 2.307095  [    0/60000]
loss: 0.909839  [ 6400/60000]
loss: 0.588225  [12800/60000]
loss: 0.715487  [19200/60000]
loss: 0.603918  [25600/60000]
loss: 0.506992  [32000/60000]
```

```
loss: 0.535688  [38400/60000]
loss: 0.596213  [44800/60000]
loss: 0.615340  [51200/60000]
loss: 0.451781  [57600/60000]
Test Error:
 Accuracy: 79.1%, Avg loss: 0.554144

Epoch 2
-------------------------------
loss: 0.441613  [    0/60000]
loss: 0.432551  [ 6400/60000]
loss: 0.364211  [12800/60000]
loss: 0.436951  [19200/60000]
loss: 0.414820  [25600/60000]
loss: 0.438897  [32000/60000]
loss: 0.406148  [38400/60000]
loss: 0.498841  [44800/60000]
loss: 0.508832  [51200/60000]
loss: 0.416277  [57600/60000]
Test Error:
 Accuracy: 82.5%, Avg loss: 0.468526

Epoch 3
-------------------------------
loss: 0.327046  [    0/60000]
loss: 0.354691  [ 6400/60000]
loss: 0.307137  [12800/60000]
loss: 0.362279  [19200/60000]
loss: 0.349934  [25600/60000]
loss: 0.403979  [32000/60000]
loss: 0.349643  [38400/60000]
loss: 0.461938  [44800/60000]
loss: 0.449331  [51200/60000]
loss: 0.399792  [57600/60000]
Test Error:
 Accuracy: 83.8%, Avg loss: 0.430261

Epoch 4
-------------------------------
loss: 0.270155  [    0/60000]
loss: 0.320561  [ 6400/60000]
loss: 0.259152  [12800/60000]
loss: 0.318088  [19200/60000]
loss: 0.329149  [25600/60000]
loss: 0.386338  [32000/60000]
loss: 0.309116  [38400/60000]
loss: 0.427329  [44800/60000]
loss: 0.408258  [51200/60000]
```

```
loss: 0.383933  [57600/60000]
Test Error:
 Accuracy: 85.0%, Avg loss: 0.400244
```

[111]:
```
plt.plot(range(1, 5), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



[115]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
epochs = 30
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss,accuracy = test(test_dataloader, model, loss_fn)
    losses.append(loss)
```

```
    if accuracy>=0.85:
        break
```

```
Epoch 1
-------------------------------
loss: 2.317503  [    0/60000]
loss: 2.167695  [ 6400/60000]
loss: 1.809497  [12800/60000]
loss: 1.509888  [19200/60000]
loss: 1.147692  [25600/60000]
loss: 1.057303  [32000/60000]
loss: 1.011451  [38400/60000]
loss: 0.870990  [44800/60000]
loss: 0.873955  [51200/60000]
loss: 0.817020  [57600/60000]
Test Error:
 Accuracy: 70.9%, Avg loss: 0.799516

Epoch 2
-------------------------------
loss: 0.798313  [    0/60000]
loss: 0.858873  [ 6400/60000]
loss: 0.599378  [12800/60000]
loss: 0.788364  [19200/60000]
loss: 0.655564  [25600/60000]
loss: 0.638938  [32000/60000]
loss: 0.703215  [38400/60000]
loss: 0.683855  [44800/60000]
loss: 0.690665  [51200/60000]
loss: 0.634622  [57600/60000]
Test Error:
 Accuracy: 77.7%, Avg loss: 0.637581

Epoch 3
-------------------------------
loss: 0.565197  [    0/60000]
loss: 0.663491  [ 6400/60000]
loss: 0.444509  [12800/60000]
loss: 0.673222  [19200/60000]
loss: 0.578515  [25600/60000]
loss: 0.555186  [32000/60000]
loss: 0.585648  [38400/60000]
loss: 0.629390  [44800/60000]
loss: 0.653324  [51200/60000]
loss: 0.546518  [57600/60000]
Test Error:
 Accuracy: 79.7%, Avg loss: 0.573971
```

```
Epoch 4
------------------------------
loss: 0.474335  [    0/60000]
loss: 0.575101  [ 6400/60000]
loss: 0.383539  [12800/60000]
loss: 0.603926  [19200/60000]
loss: 0.528993  [25600/60000]
loss: 0.513830  [32000/60000]
loss: 0.530188  [38400/60000]
loss: 0.624693  [44800/60000]
loss: 0.636878  [51200/60000]
loss: 0.487315  [57600/60000]
Test Error:
 Accuracy: 80.4%, Avg loss: 0.542026

Epoch 5
------------------------------
loss: 0.419824  [    0/60000]
loss: 0.530536  [ 6400/60000]
loss: 0.350460  [12800/60000]
loss: 0.556962  [19200/60000]
loss: 0.487933  [25600/60000]
loss: 0.486284  [32000/60000]
loss: 0.498635  [38400/60000]
loss: 0.622653  [44800/60000]
loss: 0.617550  [51200/60000]
loss: 0.451271  [57600/60000]
Test Error:
 Accuracy: 81.1%, Avg loss: 0.520890

Epoch 6
------------------------------
loss: 0.379144  [    0/60000]
loss: 0.504333  [ 6400/60000]
loss: 0.327587  [12800/60000]
loss: 0.525490  [19200/60000]
loss: 0.457545  [25600/60000]
loss: 0.468990  [32000/60000]
loss: 0.476403  [38400/60000]
loss: 0.612900  [44800/60000]
loss: 0.597557  [51200/60000]
loss: 0.431026  [57600/60000]
Test Error:
 Accuracy: 81.6%, Avg loss: 0.505586

Epoch 7
------------------------------
```

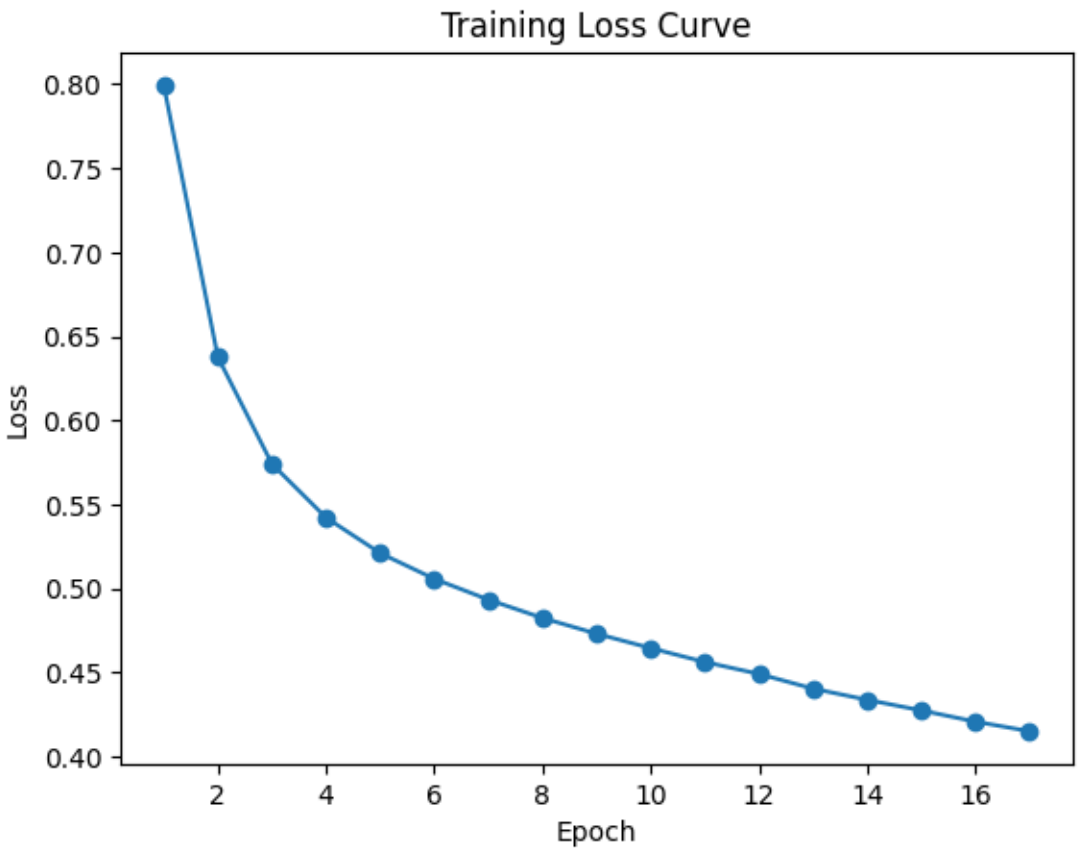
```
loss: 0.347927  [    0/60000]
loss: 0.485084  [ 6400/60000]
loss: 0.310688  [12800/60000]
loss: 0.503884  [19200/60000]
loss: 0.434557  [25600/60000]
loss: 0.456883  [32000/60000]
loss: 0.459763  [38400/60000]
loss: 0.599233  [44800/60000]
loss: 0.579794  [51200/60000]
loss: 0.419297  [57600/60000]
Test Error:
 Accuracy: 82.0%, Avg loss: 0.493189

Epoch 8
-------------------------------
loss: 0.323452  [    0/60000]
loss: 0.469262  [ 6400/60000]
loss: 0.295841  [12800/60000]
loss: 0.487316  [19200/60000]
loss: 0.412685  [25600/60000]
loss: 0.447257  [32000/60000]
loss: 0.445189  [38400/60000]
loss: 0.585422  [44800/60000]
loss: 0.562697  [51200/60000]
loss: 0.411765  [57600/60000]
Test Error:
 Accuracy: 82.4%, Avg loss: 0.482281

Epoch 9
-------------------------------
loss: 0.304589  [    0/60000]
loss: 0.456083  [ 6400/60000]
loss: 0.284438  [12800/60000]
loss: 0.474541  [19200/60000]
loss: 0.396411  [25600/60000]
loss: 0.439390  [32000/60000]
loss: 0.434305  [38400/60000]
loss: 0.573763  [44800/60000]
loss: 0.548332  [51200/60000]
loss: 0.404921  [57600/60000]
Test Error:
 Accuracy: 82.8%, Avg loss: 0.472917

Epoch 10
-------------------------------
loss: 0.290813  [    0/60000]
loss: 0.442601  [ 6400/60000]
loss: 0.276508  [12800/60000]
```

```
loss: 0.463918  [19200/60000]
loss: 0.382020  [25600/60000]
loss: 0.432327  [32000/60000]
loss: 0.424457  [38400/60000]
loss: 0.561432  [44800/60000]
loss: 0.533906  [51200/60000]
loss: 0.399090  [57600/60000]
Test Error:
 Accuracy: 83.0%, Avg loss: 0.464345

Epoch 11
-------------------------------
loss: 0.280574  [    0/60000]
loss: 0.430337  [ 6400/60000]
loss: 0.269973  [12800/60000]
loss: 0.454289  [19200/60000]
loss: 0.368427  [25600/60000]
loss: 0.425945  [32000/60000]
loss: 0.416230  [38400/60000]
loss: 0.549983  [44800/60000]
loss: 0.521517  [51200/60000]
loss: 0.394832  [57600/60000]
Test Error:
 Accuracy: 83.4%, Avg loss: 0.456073

Epoch 12
-------------------------------
loss: 0.271804  [    0/60000]
loss: 0.418747  [ 6400/60000]
loss: 0.264842  [12800/60000]
loss: 0.444590  [19200/60000]
loss: 0.357118  [25600/60000]
loss: 0.418930  [32000/60000]
loss: 0.408687  [38400/60000]
loss: 0.540369  [44800/60000]
loss: 0.509839  [51200/60000]
loss: 0.391946  [57600/60000]
Test Error:
 Accuracy: 83.8%, Avg loss: 0.448978

Epoch 13
-------------------------------
loss: 0.264526  [    0/60000]
loss: 0.408638  [ 6400/60000]
loss: 0.260973  [12800/60000]
loss: 0.433770  [19200/60000]
loss: 0.345651  [25600/60000]
loss: 0.412746  [32000/60000]
```

```
loss: 0.400900  [38400/60000]
loss: 0.530270  [44800/60000]
loss: 0.498629  [51200/60000]
loss: 0.389782  [57600/60000]
Test Error:
 Accuracy: 84.1%, Avg loss: 0.440231

Epoch 14
-------------------------------
loss: 0.257814  [    0/60000]
loss: 0.399352  [ 6400/60000]
loss: 0.255647  [12800/60000]
loss: 0.424671  [19200/60000]
loss: 0.337013  [25600/60000]
loss: 0.406737  [32000/60000]
loss: 0.393154  [38400/60000]
loss: 0.521010  [44800/60000]
loss: 0.489071  [51200/60000]
loss: 0.386834  [57600/60000]
Test Error:
 Accuracy: 84.5%, Avg loss: 0.433583

Epoch 15
-------------------------------
loss: 0.251967  [    0/60000]
loss: 0.390863  [ 6400/60000]
loss: 0.251522  [12800/60000]
loss: 0.415709  [19200/60000]
loss: 0.329366  [25600/60000]
loss: 0.401082  [32000/60000]
loss: 0.385292  [38400/60000]
loss: 0.514874  [44800/60000]
loss: 0.481270  [51200/60000]
loss: 0.384272  [57600/60000]
Test Error:
 Accuracy: 84.8%, Avg loss: 0.427277

Epoch 16
-------------------------------
loss: 0.245690  [    0/60000]
loss: 0.383784  [ 6400/60000]
loss: 0.247609  [12800/60000]
loss: 0.408109  [19200/60000]
loss: 0.324283  [25600/60000]
loss: 0.396354  [32000/60000]
loss: 0.377529  [38400/60000]
loss: 0.506654  [44800/60000]
loss: 0.473181  [51200/60000]
```

```
loss: 0.381333  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.420585

Epoch 17
-------------------------------
loss: 0.239885  [    0/60000]
loss: 0.376429  [ 6400/60000]
loss: 0.244142  [12800/60000]
loss: 0.400974  [19200/60000]
loss: 0.317202  [25600/60000]
loss: 0.391041  [32000/60000]
loss: 0.370751  [38400/60000]
loss: 0.501282  [44800/60000]
loss: 0.463861  [51200/60000]
loss: 0.379177  [57600/60000]
Test Error:
 Accuracy: 85.2%, Avg loss: 0.414888
```

[117]:
```
plt.plot(range(1, 18), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



[121]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.001)
epochs = 50
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss,accuracy = test(test_dataloader, model, loss_fn)
    losses.append(loss)
    if accuracy>=0.85:
        break
```

```
Epoch 1
-------------------------------
loss: 2.304540  [    0/60000]
loss: 2.285547  [ 6400/60000]
loss: 2.265503  [12800/60000]
loss: 2.253840  [19200/60000]
loss: 2.232651  [25600/60000]
loss: 2.213239  [32000/60000]
```

```
loss: 2.206540  [38400/60000]
loss: 2.177657  [44800/60000]
loss: 2.182170  [51200/60000]
loss: 2.133466  [57600/60000]
Test Error:
 Accuracy: 50.4%, Avg loss: 2.134931

Epoch 2
-------------------------------
loss: 2.157035  [    0/60000]
loss: 2.136670  [ 6400/60000]
loss: 2.075629  [12800/60000]
loss: 2.083957  [19200/60000]
loss: 2.021820  [25600/60000]
loss: 1.973156  [32000/60000]
loss: 1.986270  [38400/60000]
loss: 1.910672  [44800/60000]
loss: 1.928576  [51200/60000]
loss: 1.827210  [57600/60000]
Test Error:
 Accuracy: 55.5%, Avg loss: 1.841932

Epoch 3
-------------------------------
loss: 1.888944  [    0/60000]
loss: 1.845117  [ 6400/60000]
loss: 1.732115  [12800/60000]
loss: 1.768230  [19200/60000]
loss: 1.643775  [25600/60000]
loss: 1.614583  [32000/60000]
loss: 1.623812  [38400/60000]
loss: 1.538963  [44800/60000]
loss: 1.572747  [51200/60000]
loss: 1.450471  [57600/60000]
Test Error:
 Accuracy: 60.9%, Avg loss: 1.481591

Epoch 4
-------------------------------
loss: 1.557357  [    0/60000]
loss: 1.511993  [ 6400/60000]
loss: 1.371642  [12800/60000]
loss: 1.443846  [19200/60000]
loss: 1.317917  [25600/60000]
loss: 1.327743  [32000/60000]
loss: 1.330972  [38400/60000]
loss: 1.266626  [44800/60000]
loss: 1.304569  [51200/60000]
```

```
loss: 1.202868  [57600/60000]
Test Error:
 Accuracy: 63.5%, Avg loss: 1.229961

Epoch 5
-------------------------------
loss: 1.310291  [    0/60000]
loss: 1.281964  [ 6400/60000]
loss: 1.123084  [12800/60000]
loss: 1.237053  [19200/60000]
loss: 1.108816  [25600/60000]
loss: 1.137952  [32000/60000]
loss: 1.151915  [38400/60000]
loss: 1.096377  [44800/60000]
loss: 1.137157  [51200/60000]
loss: 1.057598  [57600/60000]
Test Error:
 Accuracy: 64.9%, Avg loss: 1.074824

Epoch 6
-------------------------------
loss: 1.146822  [    0/60000]
loss: 1.137757  [ 6400/60000]
loss: 0.962250  [12800/60000]
loss: 1.108562  [19200/60000]
loss: 0.981750  [25600/60000]
loss: 1.010478  [32000/60000]
loss: 1.041717  [38400/60000]
loss: 0.987953  [44800/60000]
loss: 1.027578  [51200/60000]
loss: 0.966701  [57600/60000]
Test Error:
 Accuracy: 66.0%, Avg loss: 0.974933

Epoch 7
-------------------------------
loss: 1.033643  [    0/60000]
loss: 1.044070  [ 6400/60000]
loss: 0.852464  [12800/60000]
loss: 1.022744  [19200/60000]
loss: 0.900726  [25600/60000]
loss: 0.920871  [32000/60000]
loss: 0.969224  [38400/60000]
loss: 0.916575  [44800/60000]
loss: 0.951099  [51200/60000]
loss: 0.904919  [57600/60000]
Test Error:
 Accuracy: 67.0%, Avg loss: 0.906250
```

```
Epoch 8
-------------------------------
loss: 0.950165  [    0/60000]
loss: 0.978633  [ 6400/60000]
loss: 0.772904  [12800/60000]
loss: 0.961486  [19200/60000]
loss: 0.845460  [25600/60000]
loss: 0.855016  [32000/60000]
loss: 0.917597  [38400/60000]
loss: 0.868049  [44800/60000]
loss: 0.895419  [51200/60000]
loss: 0.859351  [57600/60000]
Test Error:
 Accuracy: 68.3%, Avg loss: 0.856208

Epoch 9
-------------------------------
loss: 0.885139  [    0/60000]
loss: 0.929286  [ 6400/60000]
loss: 0.712621  [12800/60000]
loss: 0.915768  [19200/60000]
loss: 0.805206  [25600/60000]
loss: 0.804824  [32000/60000]
loss: 0.877954  [38400/60000]
loss: 0.833654  [44800/60000]
loss: 0.853121  [51200/60000]
loss: 0.823628  [57600/60000]
Test Error:
 Accuracy: 69.7%, Avg loss: 0.817763

Epoch 10
-------------------------------
loss: 0.832086  [    0/60000]
loss: 0.889550  [ 6400/60000]
loss: 0.664814  [12800/60000]
loss: 0.880278  [19200/60000]
loss: 0.773931  [25600/60000]
loss: 0.765490  [32000/60000]
loss: 0.845658  [38400/60000]
loss: 0.807837  [44800/60000]
loss: 0.819653  [51200/60000]
loss: 0.794400  [57600/60000]
Test Error:
 Accuracy: 70.9%, Avg loss: 0.786854

Epoch 11
-------------------------------
```

```
loss: 0.787499  [    0/60000]
loss: 0.855833  [ 6400/60000]
loss: 0.625589  [12800/60000]
loss: 0.851730  [19200/60000]
loss: 0.748310  [25600/60000]
loss: 0.734140  [32000/60000]
loss: 0.817958  [38400/60000]
loss: 0.787404  [44800/60000]
loss: 0.792421  [51200/60000]
loss: 0.769623  [57600/60000]
Test Error:
 Accuracy: 72.2%, Avg loss: 0.760993

Epoch 12
-------------------------------
loss: 0.749046  [    0/60000]
loss: 0.826056  [ 6400/60000]
loss: 0.592652  [12800/60000]
loss: 0.828052  [19200/60000]
loss: 0.726591  [25600/60000]
loss: 0.708620  [32000/60000]
loss: 0.793257  [38400/60000]
loss: 0.770287  [44800/60000]
loss: 0.769556  [51200/60000]
loss: 0.747922  [57600/60000]
Test Error:
 Accuracy: 73.2%, Avg loss: 0.738627

Epoch 13
-------------------------------
loss: 0.715299  [    0/60000]
loss: 0.799074  [ 6400/60000]
loss: 0.564484  [12800/60000]
loss: 0.807942  [19200/60000]
loss: 0.707770  [25600/60000]
loss: 0.687570  [32000/60000]
loss: 0.770681  [38400/60000]
loss: 0.755510  [44800/60000]
loss: 0.749834  [51200/60000]
loss: 0.728554  [57600/60000]
Test Error:
 Accuracy: 74.1%, Avg loss: 0.718794

Epoch 14
-------------------------------
loss: 0.685330  [    0/60000]
loss: 0.774282  [ 6400/60000]
loss: 0.540115  [12800/60000]
```

```
loss: 0.790343  [19200/60000]
loss: 0.691332  [25600/60000]
loss: 0.669855  [32000/60000]
loss: 0.749730  [38400/60000]
loss: 0.742387  [44800/60000]
loss: 0.732622  [51200/60000]
loss: 0.710921  [57600/60000]
Test Error:
 Accuracy: 74.8%, Avg loss: 0.700883

Epoch 15
-------------------------------
loss: 0.658375  [    0/60000]
loss: 0.751333  [ 6400/60000]
loss: 0.518885  [12800/60000]
loss: 0.774629  [19200/60000]
loss: 0.676845  [25600/60000]
loss: 0.654721  [32000/60000]
loss: 0.730133  [38400/60000]
loss: 0.730557  [44800/60000]
loss: 0.717523  [51200/60000]
loss: 0.694729  [57600/60000]
Test Error:
 Accuracy: 75.6%, Avg loss: 0.684530

Epoch 16
-------------------------------
loss: 0.634076  [    0/60000]
loss: 0.730110  [ 6400/60000]
loss: 0.500265  [12800/60000]
loss: 0.760344  [19200/60000]
loss: 0.663897  [25600/60000]
loss: 0.641661  [32000/60000]
loss: 0.711756  [38400/60000]
loss: 0.719946  [44800/60000]
loss: 0.704274  [51200/60000]
loss: 0.679756  [57600/60000]
Test Error:
 Accuracy: 76.3%, Avg loss: 0.669496

Epoch 17
-------------------------------
loss: 0.612142  [    0/60000]
loss: 0.710580  [ 6400/60000]
loss: 0.483697  [12800/60000]
loss: 0.747193  [19200/60000]
loss: 0.652324  [25600/60000]
loss: 0.630247  [32000/60000]
```

```
loss: 0.694491  [38400/60000]
loss: 0.710378  [44800/60000]
loss: 0.692755  [51200/60000]
loss: 0.665902  [57600/60000]
Test Error:
 Accuracy: 76.8%, Avg loss: 0.655660

Epoch 18
-------------------------------
loss: 0.592233  [    0/60000]
loss: 0.692646  [ 6400/60000]
loss: 0.468978  [12800/60000]
loss: 0.734994  [19200/60000]
loss: 0.641963  [25600/60000]
loss: 0.620205  [32000/60000]
loss: 0.678224  [38400/60000]
loss: 0.701761  [44800/60000]
loss: 0.682795  [51200/60000]
loss: 0.652923  [57600/60000]
Test Error:
 Accuracy: 77.3%, Avg loss: 0.642905

Epoch 19
-------------------------------
loss: 0.574168  [    0/60000]
loss: 0.676089  [ 6400/60000]
loss: 0.455816  [12800/60000]
loss: 0.723601  [19200/60000]
loss: 0.632538  [25600/60000]
loss: 0.611225  [32000/60000]
loss: 0.662944  [38400/60000]
loss: 0.694079  [44800/60000]
loss: 0.674208  [51200/60000]
loss: 0.640815  [57600/60000]
Test Error:
 Accuracy: 77.9%, Avg loss: 0.631152

Epoch 20
-------------------------------
loss: 0.557711  [    0/60000]
loss: 0.660822  [ 6400/60000]
loss: 0.443902  [12800/60000]
loss: 0.712997  [19200/60000]
loss: 0.623911  [25600/60000]
loss: 0.603243  [32000/60000]
loss: 0.648566  [38400/60000]
loss: 0.687365  [44800/60000]
loss: 0.666952  [51200/60000]
```

```
loss: 0.629485  [57600/60000]
Test Error:
 Accuracy: 78.4%, Avg loss: 0.620300

Epoch 21
-------------------------------
loss: 0.542653  [    0/60000]
loss: 0.646797  [ 6400/60000]
loss: 0.433114  [12800/60000]
loss: 0.703035  [19200/60000]
loss: 0.616002  [25600/60000]
loss: 0.596067  [32000/60000]
loss: 0.635096  [38400/60000]
loss: 0.681624  [44800/60000]
loss: 0.660822  [51200/60000]
loss: 0.618781  [57600/60000]
Test Error:
 Accuracy: 78.8%, Avg loss: 0.610284

Epoch 22
-------------------------------
loss: 0.528832  [    0/60000]
loss: 0.633933  [ 6400/60000]
loss: 0.423235  [12800/60000]
loss: 0.693684  [19200/60000]
loss: 0.608563  [25600/60000]
loss: 0.589489  [32000/60000]
loss: 0.622517  [38400/60000]
loss: 0.676830  [44800/60000]
loss: 0.655664  [51200/60000]
loss: 0.608619  [57600/60000]
Test Error:
 Accuracy: 79.1%, Avg loss: 0.601042

Epoch 23
-------------------------------
loss: 0.516047  [    0/60000]
loss: 0.622148  [ 6400/60000]
loss: 0.414220  [12800/60000]
loss: 0.684820  [19200/60000]
loss: 0.601456  [25600/60000]
loss: 0.583460  [32000/60000]
loss: 0.610830  [38400/60000]
loss: 0.672936  [44800/60000]
loss: 0.651343  [51200/60000]
loss: 0.598871  [57600/60000]
Test Error:
 Accuracy: 79.4%, Avg loss: 0.592492
```

```
Epoch 24
-------------------------------
loss: 0.504203  [    0/60000]
loss: 0.611276  [ 6400/60000]
loss: 0.405893  [12800/60000]
loss: 0.676383  [19200/60000]
loss: 0.594641  [25600/60000]
loss: 0.577834  [32000/60000]
loss: 0.599981  [38400/60000]
loss: 0.669845  [44800/60000]
loss: 0.647758  [51200/60000]
loss: 0.589520  [57600/60000]
Test Error:
 Accuracy: 79.7%, Avg loss: 0.584572

Epoch 25
-------------------------------
loss: 0.493211  [    0/60000]
loss: 0.601296  [ 6400/60000]
loss: 0.398160  [12800/60000]
loss: 0.668245  [19200/60000]
loss: 0.588035  [25600/60000]
loss: 0.572457  [32000/60000]
loss: 0.589890  [38400/60000]
loss: 0.667433  [44800/60000]
loss: 0.644760  [51200/60000]
loss: 0.580570  [57600/60000]
Test Error:
 Accuracy: 79.8%, Avg loss: 0.577237

Epoch 26
-------------------------------
loss: 0.482947  [    0/60000]
loss: 0.592096  [ 6400/60000]
loss: 0.391008  [12800/60000]
loss: 0.660528  [19200/60000]
loss: 0.581544  [25600/60000]
loss: 0.567321  [32000/60000]
loss: 0.580488  [38400/60000]
loss: 0.665697  [44800/60000]
loss: 0.642199  [51200/60000]
loss: 0.571936  [57600/60000]
Test Error:
 Accuracy: 80.1%, Avg loss: 0.570434

Epoch 27
-------------------------------
```

```
loss: 0.473387  [    0/60000]
loss: 0.583554  [ 6400/60000]
loss: 0.384389  [12800/60000]
loss: 0.653221  [19200/60000]
loss: 0.575161  [25600/60000]
loss: 0.562342  [32000/60000]
loss: 0.571704  [38400/60000]
loss: 0.664541  [44800/60000]
loss: 0.640020  [51200/60000]
loss: 0.563524  [57600/60000]
Test Error:
 Accuracy: 80.4%, Avg loss: 0.564112

Epoch 28
-------------------------------
loss: 0.464394  [    0/60000]
loss: 0.575622  [ 6400/60000]
loss: 0.378258  [12800/60000]
loss: 0.646218  [19200/60000]
loss: 0.568810  [25600/60000]
loss: 0.557496  [32000/60000]
loss: 0.563512  [38400/60000]
loss: 0.663840  [44800/60000]
loss: 0.638128  [51200/60000]
loss: 0.555327  [57600/60000]
Test Error:
 Accuracy: 80.6%, Avg loss: 0.558226

Epoch 29
-------------------------------
loss: 0.455936  [    0/60000]
loss: 0.568250  [ 6400/60000]
loss: 0.372559  [12800/60000]
loss: 0.639500  [19200/60000]
loss: 0.562463  [25600/60000]
loss: 0.552702  [32000/60000]
loss: 0.555937  [38400/60000]
loss: 0.663511  [44800/60000]
loss: 0.636441  [51200/60000]
loss: 0.547313  [57600/60000]
Test Error:
 Accuracy: 80.7%, Avg loss: 0.552743

Epoch 30
-------------------------------
loss: 0.447970  [    0/60000]
loss: 0.561396  [ 6400/60000]
loss: 0.367187  [12800/60000]
```

```
loss: 0.633052  [19200/60000]
loss: 0.556153  [25600/60000]
loss: 0.547944  [32000/60000]
loss: 0.548808  [38400/60000]
loss: 0.663452  [44800/60000]
loss: 0.634890  [51200/60000]
loss: 0.539534  [57600/60000]
Test Error:
 Accuracy: 80.8%, Avg loss: 0.547621

Epoch 31
-------------------------------
loss: 0.440426  [    0/60000]
loss: 0.555029  [ 6400/60000]
loss: 0.362105  [12800/60000]
loss: 0.626879  [19200/60000]
loss: 0.549878  [25600/60000]
loss: 0.543261  [32000/60000]
loss: 0.542136  [38400/60000]
loss: 0.663648  [44800/60000]
loss: 0.633457  [51200/60000]
loss: 0.532061  [57600/60000]
Test Error:
 Accuracy: 81.0%, Avg loss: 0.542829

Epoch 32
-------------------------------
loss: 0.433304  [    0/60000]
loss: 0.549114  [ 6400/60000]
loss: 0.357307  [12800/60000]
loss: 0.620918  [19200/60000]
loss: 0.543672  [25600/60000]
loss: 0.538691  [32000/60000]
loss: 0.535907  [38400/60000]
loss: 0.663959  [44800/60000]
loss: 0.632107  [51200/60000]
loss: 0.524818  [57600/60000]
Test Error:
 Accuracy: 81.1%, Avg loss: 0.538342

Epoch 33
-------------------------------
loss: 0.426582  [    0/60000]
loss: 0.543581  [ 6400/60000]
loss: 0.352835  [12800/60000]
loss: 0.615142  [19200/60000]
loss: 0.537576  [25600/60000]
loss: 0.534159  [32000/60000]
```

```
loss: 0.530103  [38400/60000]
loss: 0.664441  [44800/60000]
loss: 0.630744  [51200/60000]
loss: 0.517816  [57600/60000]
Test Error:
 Accuracy: 81.2%, Avg loss: 0.534138

Epoch 34
-------------------------------
loss: 0.420204  [    0/60000]
loss: 0.538425  [ 6400/60000]
loss: 0.348666  [12800/60000]
loss: 0.609588  [19200/60000]
loss: 0.531625  [25600/60000]
loss: 0.529702  [32000/60000]
loss: 0.524679  [38400/60000]
loss: 0.665009  [44800/60000]
loss: 0.629398  [51200/60000]
loss: 0.511067  [57600/60000]
Test Error:
 Accuracy: 81.5%, Avg loss: 0.530193

Epoch 35
-------------------------------
loss: 0.414155  [    0/60000]
loss: 0.533602  [ 6400/60000]
loss: 0.344710  [12800/60000]
loss: 0.604267  [19200/60000]
loss: 0.525811  [25600/60000]
loss: 0.525302  [32000/60000]
loss: 0.519560  [38400/60000]
loss: 0.665621  [44800/60000]
loss: 0.628056  [51200/60000]
loss: 0.504592  [57600/60000]
Test Error:
 Accuracy: 81.5%, Avg loss: 0.526483

Epoch 36
-------------------------------
loss: 0.408403  [    0/60000]
loss: 0.529057  [ 6400/60000]
loss: 0.340988  [12800/60000]
loss: 0.599115  [19200/60000]
loss: 0.520166  [25600/60000]
loss: 0.521027  [32000/60000]
loss: 0.514676  [38400/60000]
loss: 0.666216  [44800/60000]
loss: 0.626727  [51200/60000]
```

```
loss: 0.498461  [57600/60000]
Test Error:
 Accuracy: 81.7%, Avg loss: 0.522992

Epoch 37
-------------------------------
loss: 0.402915  [    0/60000]
loss: 0.524797  [ 6400/60000]
loss: 0.337514  [12800/60000]
loss: 0.594144  [19200/60000]
loss: 0.514634  [25600/60000]
loss: 0.516784  [32000/60000]
loss: 0.510110  [38400/60000]
loss: 0.666731  [44800/60000]
loss: 0.625343  [51200/60000]
loss: 0.492621  [57600/60000]
Test Error:
 Accuracy: 81.7%, Avg loss: 0.519695

Epoch 38
-------------------------------
loss: 0.397672  [    0/60000]
loss: 0.520720  [ 6400/60000]
loss: 0.334217  [12800/60000]
loss: 0.589411  [19200/60000]
loss: 0.509313  [25600/60000]
loss: 0.512648  [32000/60000]
loss: 0.505842  [38400/60000]
loss: 0.667105  [44800/60000]
loss: 0.623909  [51200/60000]
loss: 0.487091  [57600/60000]
Test Error:
 Accuracy: 81.8%, Avg loss: 0.516582

Epoch 39
-------------------------------
loss: 0.392653  [    0/60000]
loss: 0.516872  [ 6400/60000]
loss: 0.331058  [12800/60000]
loss: 0.584848  [19200/60000]
loss: 0.504169  [25600/60000]
loss: 0.508573  [32000/60000]
loss: 0.501791  [38400/60000]
loss: 0.667391  [44800/60000]
loss: 0.622418  [51200/60000]
loss: 0.481852  [57600/60000]
Test Error:
 Accuracy: 81.8%, Avg loss: 0.513630
```

```
Epoch 40
-------------------------------
loss: 0.387822  [    0/60000]
loss: 0.513218  [ 6400/60000]
loss: 0.328046  [12800/60000]
loss: 0.580371  [19200/60000]
loss: 0.499175  [25600/60000]
loss: 0.504620  [32000/60000]
loss: 0.497985  [38400/60000]
loss: 0.667483  [44800/60000]
loss: 0.620907  [51200/60000]
loss: 0.476919  [57600/60000]
Test Error:
 Accuracy: 81.9%, Avg loss: 0.510828

Epoch 41
-------------------------------
loss: 0.383140  [    0/60000]
loss: 0.509778  [ 6400/60000]
loss: 0.325218  [12800/60000]
loss: 0.576044  [19200/60000]
loss: 0.494272  [25600/60000]
loss: 0.500799  [32000/60000]
loss: 0.494382  [38400/60000]
loss: 0.667465  [44800/60000]
loss: 0.619334  [51200/60000]
loss: 0.472240  [57600/60000]
Test Error:
 Accuracy: 81.9%, Avg loss: 0.508165

Epoch 42
-------------------------------
loss: 0.378600  [    0/60000]
loss: 0.506490  [ 6400/60000]
loss: 0.322522  [12800/60000]
loss: 0.571831  [19200/60000]
loss: 0.489534  [25600/60000]
loss: 0.497077  [32000/60000]
loss: 0.490937  [38400/60000]
loss: 0.667333  [44800/60000]
loss: 0.617695  [51200/60000]
loss: 0.467816  [57600/60000]
Test Error:
 Accuracy: 82.0%, Avg loss: 0.505627

Epoch 43
-------------------------------
```

```
loss: 0.374218  [    0/60000]
loss: 0.503362  [ 6400/60000]
loss: 0.319938  [12800/60000]
loss: 0.567766  [19200/60000]
loss: 0.484987  [25600/60000]
loss: 0.493475  [32000/60000]
loss: 0.487667  [38400/60000]
loss: 0.667105  [44800/60000]
loss: 0.616062  [51200/60000]
loss: 0.463622  [57600/60000]
Test Error:
 Accuracy: 82.0%, Avg loss: 0.503205

Epoch 44
-------------------------------
loss: 0.369995  [    0/60000]
loss: 0.500390  [ 6400/60000]
loss: 0.317378  [12800/60000]
loss: 0.563826  [19200/60000]
loss: 0.480629  [25600/60000]
loss: 0.489918  [32000/60000]
loss: 0.484546  [38400/60000]
loss: 0.666707  [44800/60000]
loss: 0.614469  [51200/60000]
loss: 0.459655  [57600/60000]
Test Error:
 Accuracy: 82.1%, Avg loss: 0.500885

Epoch 45
-------------------------------
loss: 0.365938  [    0/60000]
loss: 0.497556  [ 6400/60000]
loss: 0.314931  [12800/60000]
loss: 0.560002  [19200/60000]
loss: 0.476433  [25600/60000]
loss: 0.486554  [32000/60000]
loss: 0.481603  [38400/60000]
loss: 0.666117  [44800/60000]
loss: 0.612864  [51200/60000]
loss: 0.455875  [57600/60000]
Test Error:
 Accuracy: 82.2%, Avg loss: 0.498663

Epoch 46
-------------------------------
loss: 0.362036  [    0/60000]
loss: 0.494844  [ 6400/60000]
loss: 0.312555  [12800/60000]
```

```
loss: 0.556294  [19200/60000]
loss: 0.472365  [25600/60000]
loss: 0.483344  [32000/60000]
loss: 0.478842  [38400/60000]
loss: 0.665355  [44800/60000]
loss: 0.611269  [51200/60000]
loss: 0.452297  [57600/60000]
Test Error:
 Accuracy: 82.3%, Avg loss: 0.496537

Epoch 47
-------------------------------
loss: 0.358288  [    0/60000]
loss: 0.492234  [ 6400/60000]
loss: 0.310312  [12800/60000]
loss: 0.552771  [19200/60000]
loss: 0.468471  [25600/60000]
loss: 0.480215  [32000/60000]
loss: 0.476223  [38400/60000]
loss: 0.664499  [44800/60000]
loss: 0.609670  [51200/60000]
loss: 0.448928  [57600/60000]
Test Error:
 Accuracy: 82.4%, Avg loss: 0.494496

Epoch 48
-------------------------------
loss: 0.354707  [    0/60000]
loss: 0.489743  [ 6400/60000]
loss: 0.308179  [12800/60000]
loss: 0.549386  [19200/60000]
loss: 0.464683  [25600/60000]
loss: 0.477221  [32000/60000]
loss: 0.473703  [38400/60000]
loss: 0.663538  [44800/60000]
loss: 0.608095  [51200/60000]
loss: 0.445728  [57600/60000]
Test Error:
 Accuracy: 82.4%, Avg loss: 0.492532

Epoch 49
-------------------------------
loss: 0.351223  [    0/60000]
loss: 0.487366  [ 6400/60000]
loss: 0.306130  [12800/60000]
loss: 0.546173  [19200/60000]
loss: 0.461054  [25600/60000]
loss: 0.474408  [32000/60000]
```

```
loss: 0.471311  [38400/60000]
loss: 0.662438  [44800/60000]
loss: 0.606487  [51200/60000]
loss: 0.442663  [57600/60000]
Test Error:
 Accuracy: 82.5%, Avg loss: 0.490638

Epoch 50
-------------------------------
loss: 0.347864  [    0/60000]
loss: 0.485096  [ 6400/60000]
loss: 0.304129  [12800/60000]
loss: 0.543055  [19200/60000]
loss: 0.457543  [25600/60000]
loss: 0.471693  [32000/60000]
loss: 0.469039  [38400/60000]
loss: 0.661262  [44800/60000]
loss: 0.604928  [51200/60000]
loss: 0.439729  [57600/60000]
Test Error:
 Accuracy: 82.5%, Avg loss: 0.488808
```

[122]:
```
epochs = 50
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss,accuracy = test(test_dataloader, model, loss_fn)
    losses.append(loss)
    if accuracy>=0.85:
        break
```

```
Epoch 1
-------------------------------
loss: 0.344601  [    0/60000]
loss: 0.482886  [ 6400/60000]
loss: 0.302215  [12800/60000]
loss: 0.540035  [19200/60000]
loss: 0.454177  [25600/60000]
loss: 0.469070  [32000/60000]
loss: 0.466863  [38400/60000]
loss: 0.659998  [44800/60000]
loss: 0.603401  [51200/60000]
loss: 0.436971  [57600/60000]
Test Error:
 Accuracy: 82.6%, Avg loss: 0.487041

Epoch 2
```

```
-------------------------------
loss: 0.341444  [    0/60000]
loss: 0.480729  [ 6400/60000]
loss: 0.300395  [12800/60000]
loss: 0.537127  [19200/60000]
loss: 0.450891  [25600/60000]
loss: 0.466536  [32000/60000]
loss: 0.464754  [38400/60000]
loss: 0.658710  [44800/60000]
loss: 0.601874  [51200/60000]
loss: 0.434399  [57600/60000]
Test Error:
 Accuracy: 82.7%, Avg loss: 0.485330

Epoch 3
-------------------------------
loss: 0.338396  [    0/60000]
loss: 0.478675  [ 6400/60000]
loss: 0.298627  [12800/60000]
loss: 0.534309  [19200/60000]
loss: 0.447681  [25600/60000]
loss: 0.464103  [32000/60000]
loss: 0.462718  [38400/60000]
loss: 0.657293  [44800/60000]
loss: 0.600362  [51200/60000]
loss: 0.431932  [57600/60000]
Test Error:
 Accuracy: 82.7%, Avg loss: 0.483671

Epoch 4
-------------------------------
loss: 0.335424  [    0/60000]
loss: 0.476658  [ 6400/60000]
loss: 0.296944  [12800/60000]
loss: 0.531582  [19200/60000]
loss: 0.444579  [25600/60000]
loss: 0.461753  [32000/60000]
loss: 0.460768  [38400/60000]
loss: 0.655807  [44800/60000]
loss: 0.598843  [51200/60000]
loss: 0.429622  [57600/60000]
Test Error:
 Accuracy: 82.8%, Avg loss: 0.482069

Epoch 5
-------------------------------
loss: 0.332516  [    0/60000]
loss: 0.474731  [ 6400/60000]
```

```
loss: 0.295290  [12800/60000]
loss: 0.528994  [19200/60000]
loss: 0.441587  [25600/60000]
loss: 0.459503  [32000/60000]
loss: 0.458940  [38400/60000]
loss: 0.654257  [44800/60000]
loss: 0.597264  [51200/60000]
loss: 0.427405  [57600/60000]
Test Error:
 Accuracy: 82.8%, Avg loss: 0.480511

Epoch 6
-------------------------------
loss: 0.329722  [    0/60000]
loss: 0.472830  [ 6400/60000]
loss: 0.293680  [12800/60000]
loss: 0.526468  [19200/60000]
loss: 0.438682  [25600/60000]
loss: 0.457356  [32000/60000]
loss: 0.457138  [38400/60000]
loss: 0.652658  [44800/60000]
loss: 0.595702  [51200/60000]
loss: 0.425331  [57600/60000]
Test Error:
 Accuracy: 82.9%, Avg loss: 0.479001

Epoch 7
-------------------------------
loss: 0.326990  [    0/60000]
loss: 0.470972  [ 6400/60000]
loss: 0.292125  [12800/60000]
loss: 0.524037  [19200/60000]
loss: 0.435838  [25600/60000]
loss: 0.455309  [32000/60000]
loss: 0.455325  [38400/60000]
loss: 0.651008  [44800/60000]
loss: 0.594097  [51200/60000]
loss: 0.423396  [57600/60000]
Test Error:
 Accuracy: 82.9%, Avg loss: 0.477531

Epoch 8
-------------------------------
loss: 0.324338  [    0/60000]
loss: 0.469147  [ 6400/60000]
loss: 0.290622  [12800/60000]
loss: 0.521689  [19200/60000]
loss: 0.433074  [25600/60000]
```

```
loss: 0.453312  [32000/60000]
loss: 0.453562  [38400/60000]
loss: 0.649336  [44800/60000]
loss: 0.592480  [51200/60000]
loss: 0.421537  [57600/60000]
Test Error:
 Accuracy: 83.0%, Avg loss: 0.476097

Epoch 9
-------------------------------
loss: 0.321741  [    0/60000]
loss: 0.467380  [ 6400/60000]
loss: 0.289171  [12800/60000]
loss: 0.519385  [19200/60000]
loss: 0.430406  [25600/60000]
loss: 0.451403  [32000/60000]
loss: 0.451889  [38400/60000]
loss: 0.647609  [44800/60000]
loss: 0.590863  [51200/60000]
loss: 0.419806  [57600/60000]
Test Error:
 Accuracy: 83.0%, Avg loss: 0.474699

Epoch 10
-------------------------------
loss: 0.319251  [    0/60000]
loss: 0.465627  [ 6400/60000]
loss: 0.287758  [12800/60000]
loss: 0.517107  [19200/60000]
loss: 0.427815  [25600/60000]
loss: 0.449581  [32000/60000]
loss: 0.450319  [38400/60000]
loss: 0.645910  [44800/60000]
loss: 0.589257  [51200/60000]
loss: 0.418143  [57600/60000]
Test Error:
 Accuracy: 83.1%, Avg loss: 0.473333

Epoch 11
-------------------------------
loss: 0.316891  [    0/60000]
loss: 0.463921  [ 6400/60000]
loss: 0.286404  [12800/60000]
loss: 0.514874  [19200/60000]
loss: 0.425255  [25600/60000]
loss: 0.447826  [32000/60000]
loss: 0.448745  [38400/60000]
loss: 0.644172  [44800/60000]
```

```
loss: 0.587607  [51200/60000]
loss: 0.416582  [57600/60000]
Test Error:
 Accuracy: 83.2%, Avg loss: 0.471998

Epoch 12
-------------------------------
loss: 0.314607  [    0/60000]
loss: 0.462269  [ 6400/60000]
loss: 0.285082  [12800/60000]
loss: 0.512762  [19200/60000]
loss: 0.422752  [25600/60000]
loss: 0.446120  [32000/60000]
loss: 0.447193  [38400/60000]
loss: 0.642407  [44800/60000]
loss: 0.585996  [51200/60000]
loss: 0.415103  [57600/60000]
Test Error:
 Accuracy: 83.2%, Avg loss: 0.470692

Epoch 13
-------------------------------
loss: 0.312403  [    0/60000]
loss: 0.460624  [ 6400/60000]
loss: 0.283790  [12800/60000]
loss: 0.510677  [19200/60000]
loss: 0.420365  [25600/60000]
loss: 0.444469  [32000/60000]
loss: 0.445702  [38400/60000]
loss: 0.640708  [44800/60000]
loss: 0.584424  [51200/60000]
loss: 0.413689  [57600/60000]
Test Error:
 Accuracy: 83.2%, Avg loss: 0.469412

Epoch 14
-------------------------------
loss: 0.310269  [    0/60000]
loss: 0.459017  [ 6400/60000]
loss: 0.282502  [12800/60000]
loss: 0.508639  [19200/60000]
loss: 0.417966  [25600/60000]
loss: 0.442929  [32000/60000]
loss: 0.444279  [38400/60000]
loss: 0.638986  [44800/60000]
loss: 0.582804  [51200/60000]
loss: 0.412354  [57600/60000]
Test Error:
```

```
 Accuracy: 83.2%, Avg loss: 0.468158

Epoch 15
-------------------------------
loss: 0.308175  [    0/60000]
loss: 0.457442  [ 6400/60000]
loss: 0.281219  [12800/60000]
loss: 0.506649  [19200/60000]
loss: 0.415619  [25600/60000]
loss: 0.441430  [32000/60000]
loss: 0.442940  [38400/60000]
loss: 0.637281  [44800/60000]
loss: 0.581194  [51200/60000]
loss: 0.411081  [57600/60000]
Test Error:
 Accuracy: 83.3%, Avg loss: 0.466933

Epoch 16
-------------------------------
loss: 0.306172  [    0/60000]
loss: 0.455857  [ 6400/60000]
loss: 0.279995  [12800/60000]
loss: 0.504697  [19200/60000]
loss: 0.413313  [25600/60000]
loss: 0.440012  [32000/60000]
loss: 0.441614  [38400/60000]
loss: 0.635549  [44800/60000]
loss: 0.579644  [51200/60000]
loss: 0.409844  [57600/60000]
Test Error:
 Accuracy: 83.3%, Avg loss: 0.465736

Epoch 17
-------------------------------
loss: 0.304228  [    0/60000]
loss: 0.454282  [ 6400/60000]
loss: 0.278789  [12800/60000]
loss: 0.502782  [19200/60000]
loss: 0.411037  [25600/60000]
loss: 0.438601  [32000/60000]
loss: 0.440290  [38400/60000]
loss: 0.633852  [44800/60000]
loss: 0.578068  [51200/60000]
loss: 0.408616  [57600/60000]
Test Error:
 Accuracy: 83.3%, Avg loss: 0.464563

Epoch 18
```

```
-------------------------------
loss: 0.302336  [    0/60000]
loss: 0.452717  [ 6400/60000]
loss: 0.277585  [12800/60000]
loss: 0.500880  [19200/60000]
loss: 0.408813  [25600/60000]
loss: 0.437179  [32000/60000]
loss: 0.439024  [38400/60000]
loss: 0.632188  [44800/60000]
loss: 0.576497  [51200/60000]
loss: 0.407352  [57600/60000]
Test Error:
 Accuracy: 83.4%, Avg loss: 0.463409

Epoch 19
-------------------------------
loss: 0.300527  [    0/60000]
loss: 0.451151  [ 6400/60000]
loss: 0.276438  [12800/60000]
loss: 0.499084  [19200/60000]
loss: 0.406641  [25600/60000]
loss: 0.435748  [32000/60000]
loss: 0.437761  [38400/60000]
loss: 0.630529  [44800/60000]
loss: 0.574938  [51200/60000]
loss: 0.406200  [57600/60000]
Test Error:
 Accuracy: 83.5%, Avg loss: 0.462278

Epoch 20
-------------------------------
loss: 0.298771  [    0/60000]
loss: 0.449560  [ 6400/60000]
loss: 0.275324  [12800/60000]
loss: 0.497330  [19200/60000]
loss: 0.404413  [25600/60000]
loss: 0.434365  [32000/60000]
loss: 0.436503  [38400/60000]
loss: 0.628873  [44800/60000]
loss: 0.573307  [51200/60000]
loss: 0.405046  [57600/60000]
Test Error:
 Accuracy: 83.5%, Avg loss: 0.461167

Epoch 21
-------------------------------
loss: 0.297054  [    0/60000]
loss: 0.447987  [ 6400/60000]
```

```
loss: 0.274274  [12800/60000]
loss: 0.495709  [19200/60000]
loss: 0.402214  [25600/60000]
loss: 0.432983  [32000/60000]
loss: 0.435282  [38400/60000]
loss: 0.627161  [44800/60000]
loss: 0.571584  [51200/60000]
loss: 0.403982  [57600/60000]
Test Error:
 Accuracy: 83.5%, Avg loss: 0.460072

Epoch 22
-------------------------------
loss: 0.295319  [    0/60000]
loss: 0.446471  [ 6400/60000]
loss: 0.273282  [12800/60000]
loss: 0.494127  [19200/60000]
loss: 0.400116  [25600/60000]
loss: 0.431669  [32000/60000]
loss: 0.434086  [38400/60000]
loss: 0.625553  [44800/60000]
loss: 0.569930  [51200/60000]
loss: 0.402960  [57600/60000]
Test Error:
 Accuracy: 83.6%, Avg loss: 0.458998

Epoch 23
-------------------------------
loss: 0.293618  [    0/60000]
loss: 0.444986  [ 6400/60000]
loss: 0.272338  [12800/60000]
loss: 0.492569  [19200/60000]
loss: 0.398151  [25600/60000]
loss: 0.430436  [32000/60000]
loss: 0.432909  [38400/60000]
loss: 0.624005  [44800/60000]
loss: 0.568377  [51200/60000]
loss: 0.401984  [57600/60000]
Test Error:
 Accuracy: 83.6%, Avg loss: 0.457941

Epoch 24
-------------------------------
loss: 0.291973  [    0/60000]
loss: 0.443550  [ 6400/60000]
loss: 0.271379  [12800/60000]
loss: 0.491019  [19200/60000]
loss: 0.396221  [25600/60000]
```

```
loss: 0.429241  [32000/60000]
loss: 0.431769  [38400/60000]
loss: 0.622479  [44800/60000]
loss: 0.566893  [51200/60000]
loss: 0.401004  [57600/60000]
Test Error:
 Accuracy: 83.7%, Avg loss: 0.456906

Epoch 25
-------------------------------
loss: 0.290381  [    0/60000]
loss: 0.442168  [ 6400/60000]
loss: 0.270424  [12800/60000]
loss: 0.489561  [19200/60000]
loss: 0.394323  [25600/60000]
loss: 0.428061  [32000/60000]
loss: 0.430603  [38400/60000]
loss: 0.621001  [44800/60000]
loss: 0.565376  [51200/60000]
loss: 0.400068  [57600/60000]
Test Error:
 Accuracy: 83.7%, Avg loss: 0.455891

Epoch 26
-------------------------------
loss: 0.288879  [    0/60000]
loss: 0.440798  [ 6400/60000]
loss: 0.269476  [12800/60000]
loss: 0.488102  [19200/60000]
loss: 0.392436  [25600/60000]
loss: 0.426927  [32000/60000]
loss: 0.429487  [38400/60000]
loss: 0.619564  [44800/60000]
loss: 0.563874  [51200/60000]
loss: 0.399180  [57600/60000]
Test Error:
 Accuracy: 83.7%, Avg loss: 0.454890

Epoch 27
-------------------------------
loss: 0.287362  [    0/60000]
loss: 0.439406  [ 6400/60000]
loss: 0.268566  [12800/60000]
loss: 0.486664  [19200/60000]
loss: 0.390589  [25600/60000]
loss: 0.425779  [32000/60000]
loss: 0.428449  [38400/60000]
loss: 0.618120  [44800/60000]
```

```
loss: 0.562432  [51200/60000]
loss: 0.398325  [57600/60000]
Test Error:
 Accuracy: 83.7%, Avg loss: 0.453910

Epoch 28
-------------------------------
loss: 0.285851  [    0/60000]
loss: 0.438033  [ 6400/60000]
loss: 0.267667  [12800/60000]
loss: 0.485234  [19200/60000]
loss: 0.388812  [25600/60000]
loss: 0.424643  [32000/60000]
loss: 0.427466  [38400/60000]
loss: 0.616696  [44800/60000]
loss: 0.561000  [51200/60000]
loss: 0.397518  [57600/60000]
Test Error:
 Accuracy: 83.7%, Avg loss: 0.452945

Epoch 29
-------------------------------
loss: 0.284407  [    0/60000]
loss: 0.436679  [ 6400/60000]
loss: 0.266744  [12800/60000]
loss: 0.483811  [19200/60000]
loss: 0.386999  [25600/60000]
loss: 0.423491  [32000/60000]
loss: 0.426491  [38400/60000]
loss: 0.615208  [44800/60000]
loss: 0.559556  [51200/60000]
loss: 0.396737  [57600/60000]
Test Error:
 Accuracy: 83.7%, Avg loss: 0.451993

Epoch 30
-------------------------------
loss: 0.282984  [    0/60000]
loss: 0.435283  [ 6400/60000]
loss: 0.265852  [12800/60000]
loss: 0.482347  [19200/60000]
loss: 0.385254  [25600/60000]
loss: 0.422398  [32000/60000]
loss: 0.425484  [38400/60000]
loss: 0.613719  [44800/60000]
loss: 0.558200  [51200/60000]
loss: 0.395940  [57600/60000]
Test Error:
```

```
 Accuracy: 83.7%, Avg loss: 0.451053

Epoch 31
-------------------------------
loss: 0.281584  [    0/60000]
loss: 0.433939  [ 6400/60000]
loss: 0.264968  [12800/60000]
loss: 0.480888  [19200/60000]
loss: 0.383563  [25600/60000]
loss: 0.421322  [32000/60000]
loss: 0.424510  [38400/60000]
loss: 0.612206  [44800/60000]
loss: 0.556866  [51200/60000]
loss: 0.395210  [57600/60000]
Test Error:
 Accuracy: 83.8%, Avg loss: 0.450129

Epoch 32
-------------------------------
loss: 0.280260  [    0/60000]
loss: 0.432619  [ 6400/60000]
loss: 0.264119  [12800/60000]
loss: 0.479447  [19200/60000]
loss: 0.381873  [25600/60000]
loss: 0.420243  [32000/60000]
loss: 0.423597  [38400/60000]
loss: 0.610754  [44800/60000]
loss: 0.555477  [51200/60000]
loss: 0.394436  [57600/60000]
Test Error:
 Accuracy: 83.8%, Avg loss: 0.449217

Epoch 33
-------------------------------
loss: 0.278985  [    0/60000]
loss: 0.431314  [ 6400/60000]
loss: 0.263281  [12800/60000]
loss: 0.478020  [19200/60000]
loss: 0.380117  [25600/60000]
loss: 0.419224  [32000/60000]
loss: 0.422700  [38400/60000]
loss: 0.609283  [44800/60000]
loss: 0.554095  [51200/60000]
loss: 0.393670  [57600/60000]
Test Error:
 Accuracy: 83.8%, Avg loss: 0.448315

Epoch 34
```

```
-------------------------------
loss: 0.277766  [    0/60000]
loss: 0.430021  [ 6400/60000]
loss: 0.262444  [12800/60000]
loss: 0.476584  [19200/60000]
loss: 0.378462  [25600/60000]
loss: 0.418237  [32000/60000]
loss: 0.421751  [38400/60000]
loss: 0.607882  [44800/60000]
loss: 0.552671  [51200/60000]
loss: 0.392928  [57600/60000]
Test Error:
 Accuracy: 83.8%, Avg loss: 0.447430

Epoch 35
-------------------------------
loss: 0.276567  [    0/60000]
loss: 0.428736  [ 6400/60000]
loss: 0.261616  [12800/60000]
loss: 0.475187  [19200/60000]
loss: 0.376832  [25600/60000]
loss: 0.417247  [32000/60000]
loss: 0.420799  [38400/60000]
loss: 0.606553  [44800/60000]
loss: 0.551339  [51200/60000]
loss: 0.392203  [57600/60000]
Test Error:
 Accuracy: 83.9%, Avg loss: 0.446560

Epoch 36
-------------------------------
loss: 0.275367  [    0/60000]
loss: 0.427460  [ 6400/60000]
loss: 0.260814  [12800/60000]
loss: 0.473797  [19200/60000]
loss: 0.375257  [25600/60000]
loss: 0.416236  [32000/60000]
loss: 0.419873  [38400/60000]
loss: 0.605158  [44800/60000]
loss: 0.550060  [51200/60000]
loss: 0.391502  [57600/60000]
Test Error:
 Accuracy: 84.0%, Avg loss: 0.445696

Epoch 37
-------------------------------
loss: 0.274196  [    0/60000]
loss: 0.426175  [ 6400/60000]
```

```
loss: 0.260032  [12800/60000]
loss: 0.472380  [19200/60000]
loss: 0.373654  [25600/60000]
loss: 0.415256  [32000/60000]
loss: 0.418981  [38400/60000]
loss: 0.603817  [44800/60000]
loss: 0.548775  [51200/60000]
loss: 0.390833  [57600/60000]
Test Error:
 Accuracy: 84.0%, Avg loss: 0.444843

Epoch 38
-------------------------------
loss: 0.273014  [    0/60000]
loss: 0.424941  [ 6400/60000]
loss: 0.259224  [12800/60000]
loss: 0.470955  [19200/60000]
loss: 0.372069  [25600/60000]
loss: 0.414219  [32000/60000]
loss: 0.418094  [38400/60000]
loss: 0.602479  [44800/60000]
loss: 0.547558  [51200/60000]
loss: 0.390162  [57600/60000]
Test Error:
 Accuracy: 84.0%, Avg loss: 0.443998

Epoch 39
-------------------------------
loss: 0.271905  [    0/60000]
loss: 0.423717  [ 6400/60000]
loss: 0.258509  [12800/60000]
loss: 0.469558  [19200/60000]
loss: 0.370549  [25600/60000]
loss: 0.413168  [32000/60000]
loss: 0.417183  [38400/60000]
loss: 0.601168  [44800/60000]
loss: 0.546347  [51200/60000]
loss: 0.389540  [57600/60000]
Test Error:
 Accuracy: 84.1%, Avg loss: 0.443164

Epoch 40
-------------------------------
loss: 0.270819  [    0/60000]
loss: 0.422459  [ 6400/60000]
loss: 0.257848  [12800/60000]
loss: 0.468198  [19200/60000]
loss: 0.369075  [25600/60000]
```

```
loss: 0.412190  [32000/60000]
loss: 0.416285  [38400/60000]
loss: 0.599836  [44800/60000]
loss: 0.545104  [51200/60000]
loss: 0.388896  [57600/60000]
Test Error:
 Accuracy: 84.2%, Avg loss: 0.442341

Epoch 41
-------------------------------
loss: 0.269714  [    0/60000]
loss: 0.421145  [ 6400/60000]
loss: 0.257164  [12800/60000]
loss: 0.466862  [19200/60000]
loss: 0.367615  [25600/60000]
loss: 0.411234  [32000/60000]
loss: 0.415298  [38400/60000]
loss: 0.598537  [44800/60000]
loss: 0.543737  [51200/60000]
loss: 0.388353  [57600/60000]
Test Error:
 Accuracy: 84.2%, Avg loss: 0.441532

Epoch 42
-------------------------------
loss: 0.268654  [    0/60000]
loss: 0.419892  [ 6400/60000]
loss: 0.256456  [12800/60000]
loss: 0.465567  [19200/60000]
loss: 0.366202  [25600/60000]
loss: 0.410264  [32000/60000]
loss: 0.414241  [38400/60000]
loss: 0.597341  [44800/60000]
loss: 0.542450  [51200/60000]
loss: 0.387810  [57600/60000]
Test Error:
 Accuracy: 84.2%, Avg loss: 0.440731

Epoch 43
-------------------------------
loss: 0.267637  [    0/60000]
loss: 0.418617  [ 6400/60000]
loss: 0.255819  [12800/60000]
loss: 0.464247  [19200/60000]
loss: 0.364788  [25600/60000]
loss: 0.409263  [32000/60000]
loss: 0.413208  [38400/60000]
loss: 0.596119  [44800/60000]
```

```
loss: 0.541153  [51200/60000]
loss: 0.387235  [57600/60000]
Test Error:
 Accuracy: 84.2%, Avg loss: 0.439933

Epoch 44
-------------------------------
loss: 0.266652  [    0/60000]
loss: 0.417394  [ 6400/60000]
loss: 0.255161  [12800/60000]
loss: 0.462969  [19200/60000]
loss: 0.363425  [25600/60000]
loss: 0.408291  [32000/60000]
loss: 0.412185  [38400/60000]
loss: 0.595005  [44800/60000]
loss: 0.539881  [51200/60000]
loss: 0.386691  [57600/60000]
Test Error:
 Accuracy: 84.2%, Avg loss: 0.439152

Epoch 45
-------------------------------
loss: 0.265685  [    0/60000]
loss: 0.416146  [ 6400/60000]
loss: 0.254499  [12800/60000]
loss: 0.461715  [19200/60000]
loss: 0.362067  [25600/60000]
loss: 0.407301  [32000/60000]
loss: 0.411095  [38400/60000]
loss: 0.593929  [44800/60000]
loss: 0.538677  [51200/60000]
loss: 0.386146  [57600/60000]
Test Error:
 Accuracy: 84.2%, Avg loss: 0.438378

Epoch 46
-------------------------------
loss: 0.264748  [    0/60000]
loss: 0.414897  [ 6400/60000]
loss: 0.253788  [12800/60000]
loss: 0.460477  [19200/60000]
loss: 0.360747  [25600/60000]
loss: 0.406285  [32000/60000]
loss: 0.410005  [38400/60000]
loss: 0.592795  [44800/60000]
loss: 0.537491  [51200/60000]
loss: 0.385613  [57600/60000]
Test Error:
```

```
 Accuracy: 84.3%, Avg loss: 0.437612

Epoch 47
-------------------------------
loss: 0.263824  [    0/60000]
loss: 0.413666  [ 6400/60000]
loss: 0.253075  [12800/60000]
loss: 0.459243  [19200/60000]
loss: 0.359482  [25600/60000]
loss: 0.405350  [32000/60000]
loss: 0.408953  [38400/60000]
loss: 0.591687  [44800/60000]
loss: 0.536266  [51200/60000]
loss: 0.385076  [57600/60000]
Test Error:
 Accuracy: 84.3%, Avg loss: 0.436856

Epoch 48
-------------------------------
loss: 0.262921  [    0/60000]
loss: 0.412460  [ 6400/60000]
loss: 0.252412  [12800/60000]
loss: 0.457999  [19200/60000]
loss: 0.358222  [25600/60000]
loss: 0.404429  [32000/60000]
loss: 0.407930  [38400/60000]
loss: 0.590606  [44800/60000]
loss: 0.535077  [51200/60000]
loss: 0.384528  [57600/60000]
Test Error:
 Accuracy: 84.3%, Avg loss: 0.436104

Epoch 49
-------------------------------
loss: 0.262075  [    0/60000]
loss: 0.411278  [ 6400/60000]
loss: 0.251709  [12800/60000]
loss: 0.456776  [19200/60000]
loss: 0.357004  [25600/60000]
loss: 0.403439  [32000/60000]
loss: 0.406887  [38400/60000]
loss: 0.589452  [44800/60000]
loss: 0.533892  [51200/60000]
loss: 0.384009  [57600/60000]
Test Error:
 Accuracy: 84.4%, Avg loss: 0.435356

Epoch 50
```

```
-------------------------------
loss: 0.261290  [    0/60000]
loss: 0.410056  [ 6400/60000]
loss: 0.250997  [12800/60000]
loss: 0.455600  [19200/60000]
loss: 0.355860  [25600/60000]
loss: 0.402466  [32000/60000]
loss: 0.405922  [38400/60000]
loss: 0.588374  [44800/60000]
loss: 0.532718  [51200/60000]
loss: 0.383542  [57600/60000]
Test Error:
 Accuracy: 84.4%, Avg loss: 0.434611
```

[123]:
```
epochs = 50
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss,accuracy = test(test_dataloader, model, loss_fn)
    losses.append(loss)
    if accuracy>=0.85:
        break
```

```
Epoch 1
-------------------------------
loss: 0.260494  [    0/60000]
loss: 0.408835  [ 6400/60000]
loss: 0.250325  [12800/60000]
loss: 0.454397  [19200/60000]
loss: 0.354677  [25600/60000]
loss: 0.401491  [32000/60000]
loss: 0.405057  [38400/60000]
loss: 0.587352  [44800/60000]
loss: 0.531559  [51200/60000]
loss: 0.383180  [57600/60000]
Test Error:
 Accuracy: 84.5%, Avg loss: 0.433865

Epoch 2
-------------------------------
loss: 0.259720  [    0/60000]
loss: 0.407630  [ 6400/60000]
loss: 0.249706  [12800/60000]
loss: 0.453252  [19200/60000]
loss: 0.353562  [25600/60000]
loss: 0.400520  [32000/60000]
loss: 0.404164  [38400/60000]
```

```
loss: 0.586314  [44800/60000]
loss: 0.530584  [51200/60000]
loss: 0.382768  [57600/60000]
Test Error:
 Accuracy: 84.5%, Avg loss: 0.433123

Epoch 3
-------------------------------
loss: 0.258956  [    0/60000]
loss: 0.406446  [ 6400/60000]
loss: 0.249102  [12800/60000]
loss: 0.452031  [19200/60000]
loss: 0.352452  [25600/60000]
loss: 0.399683  [32000/60000]
loss: 0.403283  [38400/60000]
loss: 0.585351  [44800/60000]
loss: 0.529589  [51200/60000]
loss: 0.382354  [57600/60000]
Test Error:
 Accuracy: 84.6%, Avg loss: 0.432395

Epoch 4
-------------------------------
loss: 0.258238  [    0/60000]
loss: 0.405261  [ 6400/60000]
loss: 0.248446  [12800/60000]
loss: 0.450782  [19200/60000]
loss: 0.351211  [25600/60000]
loss: 0.398743  [32000/60000]
loss: 0.402368  [38400/60000]
loss: 0.584360  [44800/60000]
loss: 0.528606  [51200/60000]
loss: 0.381867  [57600/60000]
Test Error:
 Accuracy: 84.6%, Avg loss: 0.431679

Epoch 5
-------------------------------
loss: 0.257530  [    0/60000]
loss: 0.404131  [ 6400/60000]
loss: 0.247785  [12800/60000]
loss: 0.449544  [19200/60000]
loss: 0.349929  [25600/60000]
loss: 0.397679  [32000/60000]
loss: 0.401382  [38400/60000]
loss: 0.583263  [44800/60000]
loss: 0.527481  [51200/60000]
loss: 0.381291  [57600/60000]
```

```
Test Error:
 Accuracy: 84.6%, Avg loss: 0.430978

Epoch 6
-------------------------------
loss: 0.256858  [    0/60000]
loss: 0.402953  [ 6400/60000]
loss: 0.247150  [12800/60000]
loss: 0.448344  [19200/60000]
loss: 0.348653  [25600/60000]
loss: 0.396640  [32000/60000]
loss: 0.400446  [38400/60000]
loss: 0.582254  [44800/60000]
loss: 0.526345  [51200/60000]
loss: 0.380813  [57600/60000]
Test Error:
 Accuracy: 84.7%, Avg loss: 0.430289

Epoch 7
-------------------------------
loss: 0.256193  [    0/60000]
loss: 0.401835  [ 6400/60000]
loss: 0.246476  [12800/60000]
loss: 0.447128  [19200/60000]
loss: 0.347397  [25600/60000]
loss: 0.395655  [32000/60000]
loss: 0.399462  [38400/60000]
loss: 0.581360  [44800/60000]
loss: 0.525165  [51200/60000]
loss: 0.380354  [57600/60000]
Test Error:
 Accuracy: 84.7%, Avg loss: 0.429600

Epoch 8
-------------------------------
loss: 0.255597  [    0/60000]
loss: 0.400696  [ 6400/60000]
loss: 0.245727  [12800/60000]
loss: 0.445874  [19200/60000]
loss: 0.346029  [25600/60000]
loss: 0.394633  [32000/60000]
loss: 0.398467  [38400/60000]
loss: 0.580471  [44800/60000]
loss: 0.523914  [51200/60000]
loss: 0.379836  [57600/60000]
Test Error:
 Accuracy: 84.8%, Avg loss: 0.428915
```

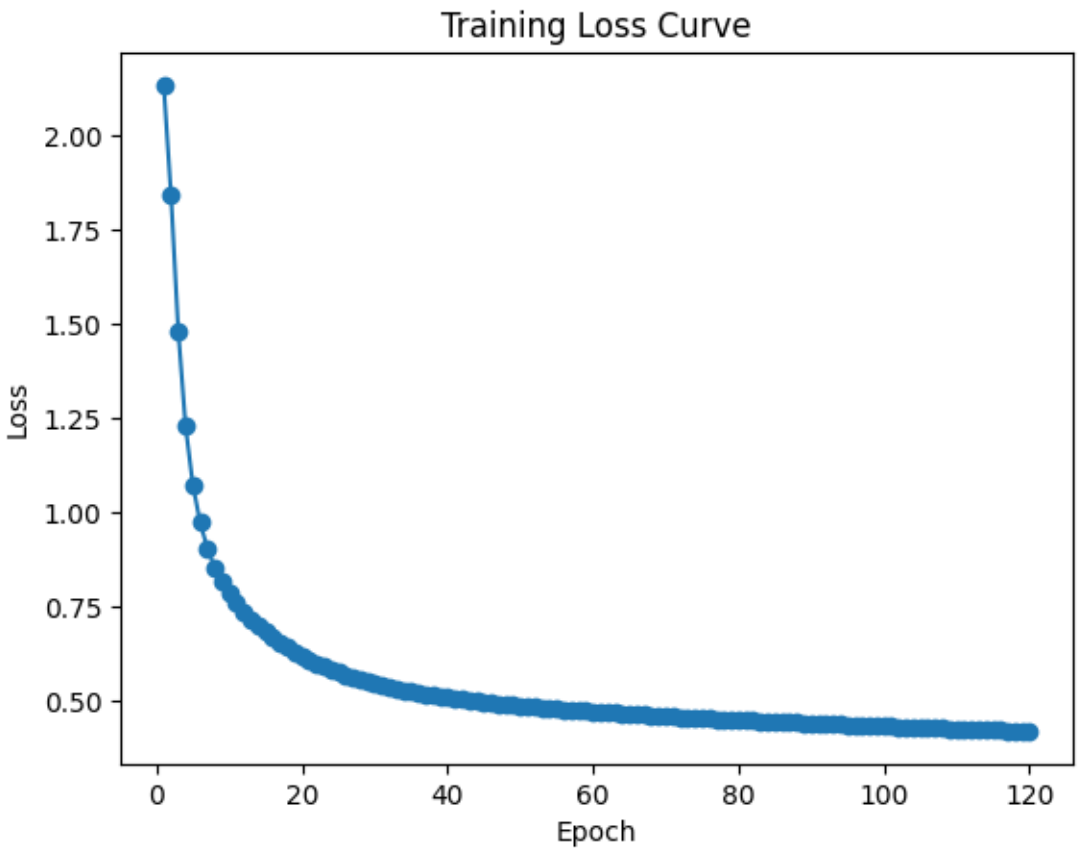
```
Epoch 9
-------------------------------
loss: 0.254965  [    0/60000]
loss: 0.399586  [ 6400/60000]
loss: 0.245084  [12800/60000]
loss: 0.444595  [19200/60000]
loss: 0.344780  [25600/60000]
loss: 0.393612  [32000/60000]
loss: 0.397506  [38400/60000]
loss: 0.579513  [44800/60000]
loss: 0.522699  [51200/60000]
loss: 0.379332  [57600/60000]
Test Error:
 Accuracy: 84.8%, Avg loss: 0.428239

Epoch 10
-------------------------------
loss: 0.254380  [    0/60000]
loss: 0.398494  [ 6400/60000]
loss: 0.244416  [12800/60000]
loss: 0.443333  [19200/60000]
loss: 0.343561  [25600/60000]
loss: 0.392684  [32000/60000]
loss: 0.396630  [38400/60000]
loss: 0.578572  [44800/60000]
loss: 0.521519  [51200/60000]
loss: 0.378871  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.427576

Epoch 11
-------------------------------
loss: 0.253738  [    0/60000]
loss: 0.397391  [ 6400/60000]
loss: 0.243746  [12800/60000]
loss: 0.442142  [19200/60000]
loss: 0.342400  [25600/60000]
loss: 0.391820  [32000/60000]
loss: 0.395807  [38400/60000]
loss: 0.577550  [44800/60000]
loss: 0.520361  [51200/60000]
loss: 0.378407  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.426919

Epoch 12
-------------------------------
loss: 0.253181  [    0/60000]
```

```
loss: 0.396273  [ 6400/60000]
loss: 0.243084  [12800/60000]
loss: 0.440949  [19200/60000]
loss: 0.341219  [25600/60000]
loss: 0.390839  [32000/60000]
loss: 0.394929  [38400/60000]
loss: 0.576469  [44800/60000]
loss: 0.519180  [51200/60000]
loss: 0.377933  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.426264

Epoch 13
-------------------------------
loss: 0.252629  [    0/60000]
loss: 0.395191  [ 6400/60000]
loss: 0.242473  [12800/60000]
loss: 0.439786  [19200/60000]
loss: 0.340042  [25600/60000]
loss: 0.389952  [32000/60000]
loss: 0.394038  [38400/60000]
loss: 0.575360  [44800/60000]
loss: 0.518039  [51200/60000]
loss: 0.377467  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.425609

Epoch 14
-------------------------------
loss: 0.252089  [    0/60000]
loss: 0.394177  [ 6400/60000]
loss: 0.241921  [12800/60000]
loss: 0.438606  [19200/60000]
loss: 0.338890  [25600/60000]
loss: 0.389094  [32000/60000]
loss: 0.393132  [38400/60000]
loss: 0.574328  [44800/60000]
loss: 0.516962  [51200/60000]
loss: 0.377120  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.424965

Epoch 15
-------------------------------
loss: 0.251520  [    0/60000]
loss: 0.393168  [ 6400/60000]
loss: 0.241370  [12800/60000]
loss: 0.437440  [19200/60000]
```

```
loss: 0.337873  [25600/60000]
loss: 0.388272  [32000/60000]
loss: 0.392220  [38400/60000]
loss: 0.573317  [44800/60000]
loss: 0.515904  [51200/60000]
loss: 0.376723  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.424322

Epoch 16
-------------------------------
loss: 0.250904  [    0/60000]
loss: 0.392187  [ 6400/60000]
loss: 0.240862  [12800/60000]
loss: 0.436294  [19200/60000]
loss: 0.336902  [25600/60000]
loss: 0.387394  [32000/60000]
loss: 0.391346  [38400/60000]
loss: 0.572325  [44800/60000]
loss: 0.514819  [51200/60000]
loss: 0.376351  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.423685

Epoch 17
-------------------------------
loss: 0.250338  [    0/60000]
loss: 0.391170  [ 6400/60000]
loss: 0.240345  [12800/60000]
loss: 0.435170  [19200/60000]
loss: 0.335923  [25600/60000]
loss: 0.386527  [32000/60000]
loss: 0.390407  [38400/60000]
loss: 0.571345  [44800/60000]
loss: 0.513822  [51200/60000]
loss: 0.375933  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.423059

Epoch 18
-------------------------------
loss: 0.249765  [    0/60000]
loss: 0.390184  [ 6400/60000]
loss: 0.239805  [12800/60000]
loss: 0.434023  [19200/60000]
loss: 0.334968  [25600/60000]
loss: 0.385694  [32000/60000]
loss: 0.389524  [38400/60000]
```

```
loss: 0.570351  [44800/60000]
loss: 0.512783  [51200/60000]
loss: 0.375511  [57600/60000]
Test Error:
 Accuracy: 85.0%, Avg loss: 0.422434

Epoch 19
-------------------------------
loss: 0.249211  [    0/60000]
loss: 0.389177  [ 6400/60000]
loss: 0.239282  [12800/60000]
loss: 0.432818  [19200/60000]
loss: 0.334058  [25600/60000]
loss: 0.384942  [32000/60000]
loss: 0.388665  [38400/60000]
loss: 0.569288  [44800/60000]
loss: 0.511768  [51200/60000]
loss: 0.375078  [57600/60000]
Test Error:
 Accuracy: 85.0%, Avg loss: 0.421815

Epoch 20
-------------------------------
loss: 0.248645  [    0/60000]
loss: 0.388212  [ 6400/60000]
loss: 0.238827  [12800/60000]
loss: 0.431618  [19200/60000]
loss: 0.333106  [25600/60000]
loss: 0.384140  [32000/60000]
loss: 0.387773  [38400/60000]
loss: 0.568198  [44800/60000]
loss: 0.510753  [51200/60000]
loss: 0.374695  [57600/60000]
Test Error:
 Accuracy: 85.0%, Avg loss: 0.421201
```

[124]:
```python
plt.plot(range(1, 121), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```



[2]:
```
#answer for question 1.4
#When learning rate is at a moderate rage like 0.1 and 0.01,not only will we␣
 ↪get good output in limited epochs,but also
#we can reach to a good performance faster.When lr is too small like 0.
 ↪001,within limited steps we may cannot get a good
#output because the model need more steps to converge,but we still can reach to␣
 ↪a good performance finally-though we need
#too many steps.When lr is too large like 1,it is likely that our model's␣
 ↪performance oscillates and cannot get a good output,
#because the output cannot converge.
#We can make a conclusion that only by setting learning rate properly can we␣
 ↪get a good performance on training models.
```

[24]:
```
#code for question 1.5
#This is a wider model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
```

```
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 1024),
            nn.ReLU(),
            nn.Linear(1024, 1024),
            nn.ReLU(),
            nn.Linear(1024, 10)
        )

    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)

        return logits

model = NeuralNetwork().to(device)
print(model)
total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print("Total Trainable Parameters:", total_params)
```

```
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=784, out_features=1024, bias=True)
    (1): ReLU()
    (2): Linear(in_features=1024, out_features=1024, bias=True)
    (3): ReLU()
    (4): Linear(in_features=1024, out_features=10, bias=True)
  )
)
Total Trainable Parameters: 1863690
```

[27]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss = test(test_dataloader, model, loss_fn)
    losses.append(loss)
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```

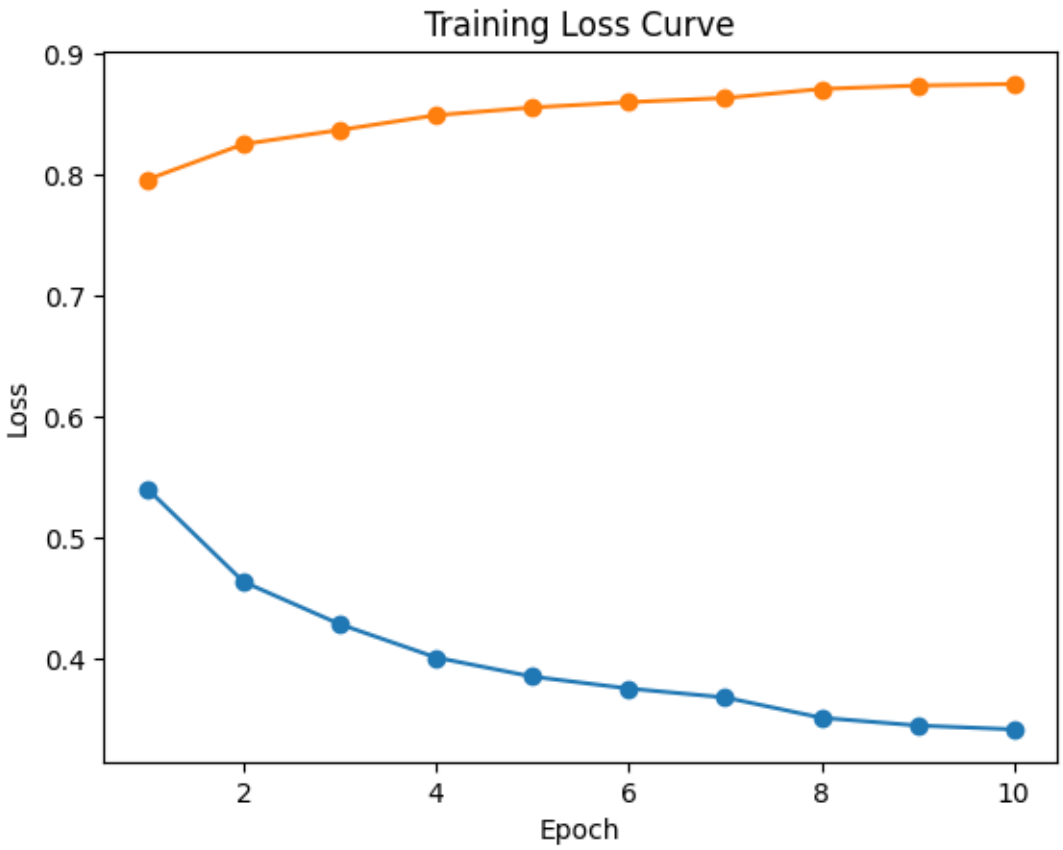
```
Epoch 1
-------------------------------
```

```
loss: 2.307396  [    0/60000]
loss: 0.871352  [ 6400/60000]
loss: 0.562661  [12800/60000]
loss: 0.687149  [19200/60000]
loss: 0.589823  [25600/60000]
loss: 0.498661  [32000/60000]
loss: 0.537491  [38400/60000]
loss: 0.594361  [44800/60000]
loss: 0.588546  [51200/60000]
loss: 0.455800  [57600/60000]
Test Error:
 Accuracy: 79.6%, Avg loss: 0.540709

Epoch 2
-------------------------------
loss: 0.424136  [    0/60000]
loss: 0.422318  [ 6400/60000]
loss: 0.367479  [12800/60000]
loss: 0.428811  [19200/60000]
loss: 0.414222  [25600/60000]
loss: 0.439105  [32000/60000]
loss: 0.410221  [38400/60000]
loss: 0.494550  [44800/60000]
loss: 0.510030  [51200/60000]
loss: 0.419943  [57600/60000]
Test Error:
 Accuracy: 82.5%, Avg loss: 0.463743

Epoch 3
-------------------------------
loss: 0.330608  [    0/60000]
loss: 0.349209  [ 6400/60000]
loss: 0.307218  [12800/60000]
loss: 0.358426  [19200/60000]
loss: 0.343613  [25600/60000]
loss: 0.410841  [32000/60000]
loss: 0.354875  [38400/60000]
loss: 0.443365  [44800/60000]
loss: 0.456592  [51200/60000]
loss: 0.407405  [57600/60000]
Test Error:
 Accuracy: 83.7%, Avg loss: 0.428697

Epoch 4
-------------------------------
loss: 0.280058  [    0/60000]
loss: 0.315982  [ 6400/60000]
loss: 0.251896  [12800/60000]
```

```
loss: 0.321840  [19200/60000]
loss: 0.318680  [25600/60000]
loss: 0.385172  [32000/60000]
loss: 0.312817  [38400/60000]
loss: 0.399631  [44800/60000]
loss: 0.419093  [51200/60000]
loss: 0.389819  [57600/60000]
Test Error:
 Accuracy: 84.9%, Avg loss: 0.401165

Epoch 5
-------------------------------
loss: 0.244083  [    0/60000]
loss: 0.301185  [ 6400/60000]
loss: 0.218271  [12800/60000]
loss: 0.296128  [19200/60000]
loss: 0.303500  [25600/60000]
loss: 0.364828  [32000/60000]
loss: 0.288318  [38400/60000]
loss: 0.357075  [44800/60000]
loss: 0.382619  [51200/60000]
loss: 0.373190  [57600/60000]
Test Error:
 Accuracy: 85.5%, Avg loss: 0.385374

Epoch 6
-------------------------------
loss: 0.226074  [    0/60000]
loss: 0.295473  [ 6400/60000]
loss: 0.196340  [12800/60000]
loss: 0.276832  [19200/60000]
loss: 0.295558  [25600/60000]
loss: 0.352219  [32000/60000]
loss: 0.259972  [38400/60000]
loss: 0.335367  [44800/60000]
loss: 0.364474  [51200/60000]
loss: 0.364973  [57600/60000]
Test Error:
 Accuracy: 86.0%, Avg loss: 0.375565

Epoch 7
-------------------------------
loss: 0.217451  [    0/60000]
loss: 0.284861  [ 6400/60000]
loss: 0.184275  [12800/60000]
loss: 0.260213  [19200/60000]
loss: 0.292193  [25600/60000]
loss: 0.343781  [32000/60000]
```

```
loss: 0.239825  [38400/60000]
loss: 0.304151  [44800/60000]
loss: 0.342752  [51200/60000]
loss: 0.344757  [57600/60000]
Test Error:
 Accuracy: 86.3%, Avg loss: 0.368183

Epoch 8
-------------------------------
loss: 0.202833  [    0/60000]
loss: 0.275680  [ 6400/60000]
loss: 0.174896  [12800/60000]
loss: 0.242167  [19200/60000]
loss: 0.288213  [25600/60000]
loss: 0.334119  [32000/60000]
loss: 0.231833  [38400/60000]
loss: 0.286144  [44800/60000]
loss: 0.330308  [51200/60000]
loss: 0.336536  [57600/60000]
Test Error:
 Accuracy: 87.1%, Avg loss: 0.351417

Epoch 9
-------------------------------
loss: 0.190573  [    0/60000]
loss: 0.263025  [ 6400/60000]
loss: 0.170667  [12800/60000]
loss: 0.233673  [19200/60000]
loss: 0.276650  [25600/60000]
loss: 0.324352  [32000/60000]
loss: 0.222577  [38400/60000]
loss: 0.270736  [44800/60000]
loss: 0.313013  [51200/60000]
loss: 0.317013  [57600/60000]
Test Error:
 Accuracy: 87.4%, Avg loss: 0.344993

Epoch 10
-------------------------------
loss: 0.176942  [    0/60000]
loss: 0.249220  [ 6400/60000]
loss: 0.159429  [12800/60000]
loss: 0.227108  [19200/60000]
loss: 0.265567  [25600/60000]
loss: 0.312458  [32000/60000]
loss: 0.205556  [38400/60000]
loss: 0.257322  [44800/60000]
loss: 0.304709  [51200/60000]
```

```
loss: 0.307137  [57600/60000]
Test Error:
 Accuracy: 87.5%, Avg loss: 0.341795
```



[30]:
```python
#This is a deeper model
class NeuralNetwork(nn.Module):
    def __init__(self):
        super(NeuralNetwork, self).__init__()
        self.flatten = nn.Flatten()
        self.linear_relu_stack = nn.Sequential(
            nn.Linear(28*28, 512),
            nn.ReLU(),
            nn.Linear(512, 512),
            nn.ReLU(),
            nn.Linear(512, 256),
            nn.ReLU(),
            nn.Linear(256, 10)
        )
```

```
    def forward(self, x):
        x = self.flatten(x)
        logits = self.linear_relu_stack(x)

        return logits

model = NeuralNetwork().to(device)
print(model)
total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print("Total Trainable Parameters:", total_params)
```

```
NeuralNetwork(
  (flatten): Flatten(start_dim=1, end_dim=-1)
  (linear_relu_stack): Sequential(
    (0): Linear(in_features=784, out_features=512, bias=True)
    (1): ReLU()
    (2): Linear(in_features=512, out_features=512, bias=True)
    (3): ReLU()
    (4): Linear(in_features=512, out_features=256, bias=True)
    (5): ReLU()
    (6): Linear(in_features=256, out_features=10, bias=True)
  )
)
Total Trainable Parameters: 798474
```

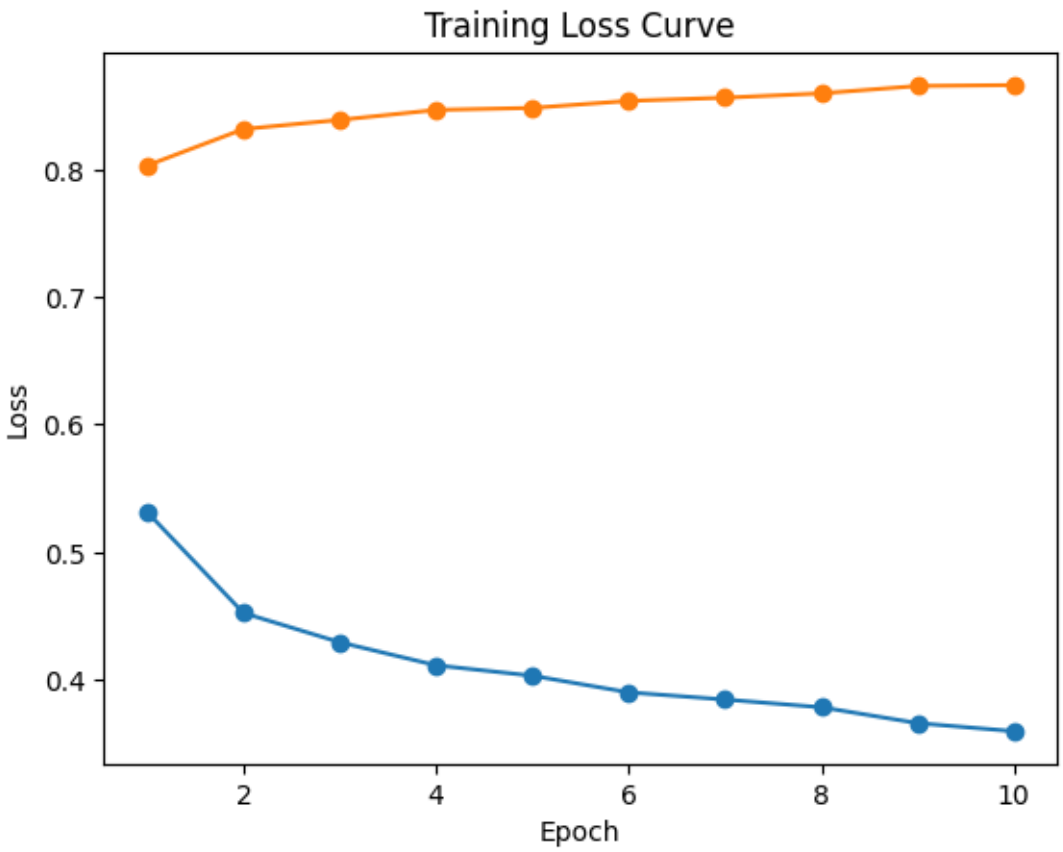
[33]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss = test(test_dataloader, model, loss_fn)
    losses.append(loss)
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```

```
Epoch 1
-------------------------------
loss: 2.291851  [    0/60000]
loss: 1.105268  [ 6400/60000]
loss: 0.688111  [12800/60000]
loss: 0.769105  [19200/60000]
loss: 0.567580  [25600/60000]
loss: 0.493879  [32000/60000]
```

```
loss: 0.548029  [38400/60000]
loss: 0.613203  [44800/60000]
loss: 0.590509  [51200/60000]
loss: 0.459583  [57600/60000]
Test Error:
 Accuracy: 80.3%, Avg loss: 0.531451

Epoch 2
-------------------------------
loss: 0.410913  [    0/60000]
loss: 0.460559  [ 6400/60000]
loss: 0.416702  [12800/60000]
loss: 0.434720  [19200/60000]
loss: 0.394074  [25600/60000]
loss: 0.469821  [32000/60000]
loss: 0.408441  [38400/60000]
loss: 0.518384  [44800/60000]
loss: 0.510045  [51200/60000]
loss: 0.420006  [57600/60000]
Test Error:
 Accuracy: 83.2%, Avg loss: 0.452255

Epoch 3
-------------------------------
loss: 0.305714  [    0/60000]
loss: 0.364338  [ 6400/60000]
loss: 0.334382  [12800/60000]
loss: 0.358212  [19200/60000]
loss: 0.337037  [25600/60000]
loss: 0.438513  [32000/60000]
loss: 0.346485  [38400/60000]
loss: 0.462002  [44800/60000]
loss: 0.439283  [51200/60000]
loss: 0.411022  [57600/60000]
Test Error:
 Accuracy: 83.9%, Avg loss: 0.429306

Epoch 4
-------------------------------
loss: 0.264533  [    0/60000]
loss: 0.325105  [ 6400/60000]
loss: 0.285297  [12800/60000]
loss: 0.327752  [19200/60000]
loss: 0.313769  [25600/60000]
loss: 0.406831  [32000/60000]
loss: 0.300566  [38400/60000]
loss: 0.428658  [44800/60000]
loss: 0.410690  [51200/60000]
```

```
loss: 0.397978  [57600/60000]
Test Error:
 Accuracy: 84.7%, Avg loss: 0.411244

Epoch 5
-------------------------------
loss: 0.243324  [    0/60000]
loss: 0.324714  [ 6400/60000]
loss: 0.239690  [12800/60000]
loss: 0.304618  [19200/60000]
loss: 0.305535  [25600/60000]
loss: 0.399668  [32000/60000]
loss: 0.271418  [38400/60000]
loss: 0.388691  [44800/60000]
loss: 0.375603  [51200/60000]
loss: 0.376097  [57600/60000]
Test Error:
 Accuracy: 84.8%, Avg loss: 0.403061

Epoch 6
-------------------------------
loss: 0.226493  [    0/60000]
loss: 0.308348  [ 6400/60000]
loss: 0.208303  [12800/60000]
loss: 0.285093  [19200/60000]
loss: 0.293534  [25600/60000]
loss: 0.386094  [32000/60000]
loss: 0.253703  [38400/60000]
loss: 0.363990  [44800/60000]
loss: 0.368906  [51200/60000]
loss: 0.361244  [57600/60000]
Test Error:
 Accuracy: 85.4%, Avg loss: 0.390028

Epoch 7
-------------------------------
loss: 0.210006  [    0/60000]
loss: 0.291232  [ 6400/60000]
loss: 0.179523  [12800/60000]
loss: 0.267850  [19200/60000]
loss: 0.289094  [25600/60000]
loss: 0.357647  [32000/60000]
loss: 0.240984  [38400/60000]
loss: 0.334109  [44800/60000]
loss: 0.341371  [51200/60000]
loss: 0.347259  [57600/60000]
Test Error:
 Accuracy: 85.6%, Avg loss: 0.384387
```

```
Epoch 8
-------------------------------
loss: 0.197256  [    0/60000]
loss: 0.273131  [ 6400/60000]
loss: 0.164888  [12800/60000]
loss: 0.257632  [19200/60000]
loss: 0.293921  [25600/60000]
loss: 0.344426  [32000/60000]
loss: 0.229172  [38400/60000]
loss: 0.315058  [44800/60000]
loss: 0.315144  [51200/60000]
loss: 0.340157  [57600/60000]
Test Error:
 Accuracy: 86.0%, Avg loss: 0.378336

Epoch 9
-------------------------------
loss: 0.184949  [    0/60000]
loss: 0.258340  [ 6400/60000]
loss: 0.153859  [12800/60000]
loss: 0.235673  [19200/60000]
loss: 0.282981  [25600/60000]
loss: 0.327060  [32000/60000]
loss: 0.217967  [38400/60000]
loss: 0.300208  [44800/60000]
loss: 0.312277  [51200/60000]
loss: 0.313032  [57600/60000]
Test Error:
 Accuracy: 86.6%, Avg loss: 0.365885

Epoch 10
-------------------------------
loss: 0.167611  [    0/60000]
loss: 0.249167  [ 6400/60000]
loss: 0.143007  [12800/60000]
loss: 0.206440  [19200/60000]
loss: 0.281243  [25600/60000]
loss: 0.299545  [32000/60000]
loss: 0.204777  [38400/60000]
loss: 0.287972  [44800/60000]
loss: 0.307300  [51200/60000]
loss: 0.283305  [57600/60000]
Test Error:
 Accuracy: 86.6%, Avg loss: 0.359539
```



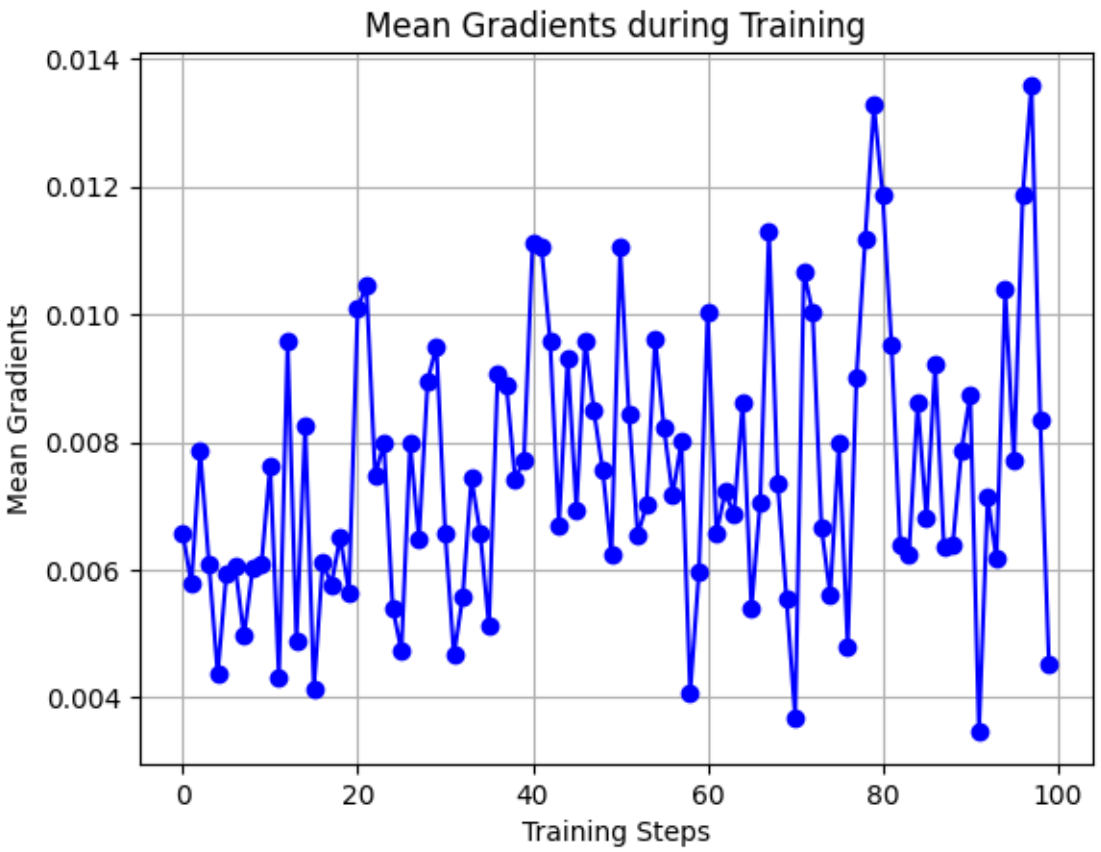
[34]:
```
#Answer for question 1.5
#We can see that both the approaches of increasing the depth and width of the␣
 ↪model increase the training parameters of the model,
#but as far as the results are concerned these two approaches behave differently
#Here we use a lr of 0.1, which performs best in the original model, and an␣
 ↪eppoch setting of 10 to observe the accuracy
#and loss curves of different models after training.
#It can be seen that compared to the baseline, after training, the approach of␣
 ↪increasing the width improves the accuracy of the model
#, and the approach of increasing the depth has a slightly worse model␣
 ↪performance than the baseline.
#However,increasing the width greatly increases the model parameters, and␣
 ↪increasing the depth slightly increases the parameters
#While increasing model width improves model performance, the parameters are␣
 ↪also greatly increased, revealing a trade-off between
#computational cost and model performance in model selection and training.
```

[84]:
```
#code for question 1.6
def train(dataloader, model, loss_fn, optimizer):
```

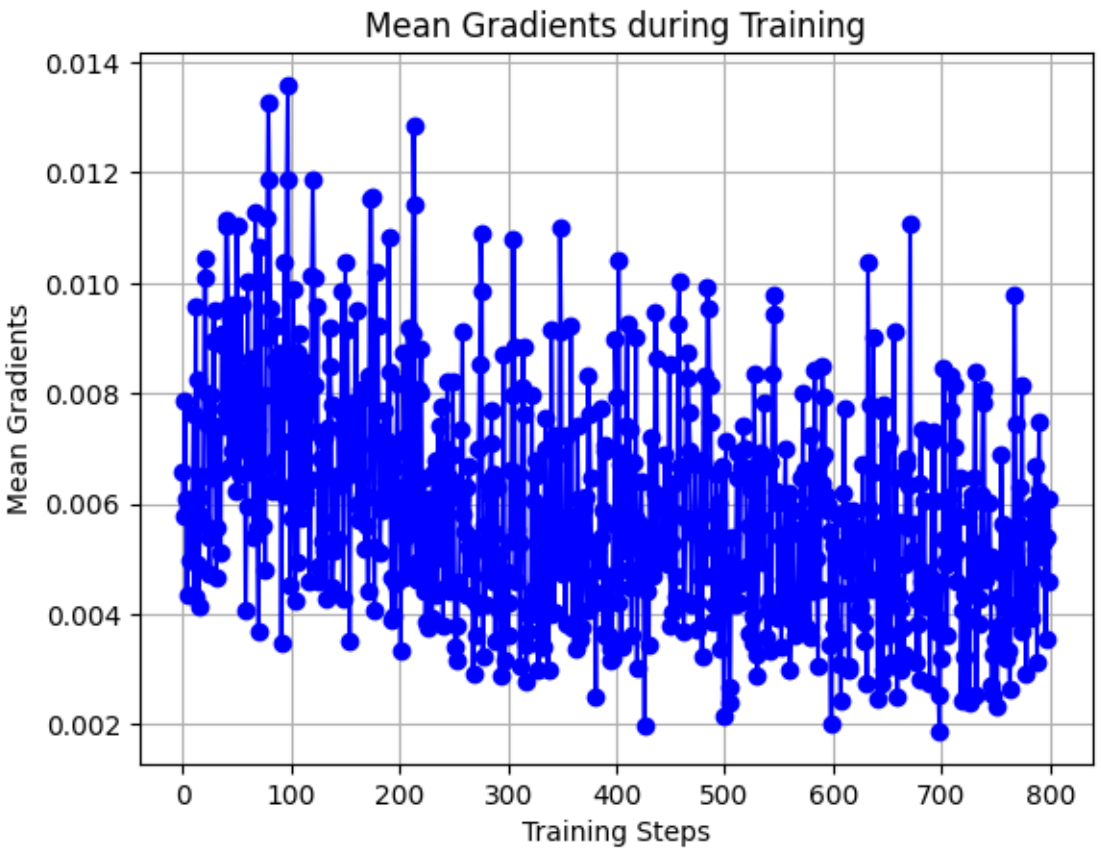
```
    size = len(dataloader.dataset)
    model.train()
    mean_gradients = []
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)
        pred = model(X)
        loss = loss_fn(pred, y)
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        mean_gradients.append(torch.mean(torch.stack([p.grad.abs().mean() for p␣
 ↪in model.parameters()])).item())
        if batch % 100 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")
    return mean_gradients
model = NeuralNetwork().to(device)
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.1)
epochs = 1
gradients=[]
for t in range(epochs):
    mean_gradients=train(train_dataloader, model, loss_fn, optimizer)
print("Done!")
```

```
loss: 2.300651  [    0/60000]
loss: 0.884644  [ 6400/60000]
loss: 0.576079  [12800/60000]
loss: 0.691893  [19200/60000]
loss: 0.603040  [25600/60000]
loss: 0.518191  [32000/60000]
loss: 0.544318  [38400/60000]
loss: 0.591451  [44800/60000]
loss: 0.618802  [51200/60000]
loss: 0.452078  [57600/60000]
Done!
```

[88]:
```
steps=100
training_steps = [i for i in range(steps)]
plt.plot(training_steps, mean_gradients[:steps], marker='o', linestyle='-',␣
 ↪color='b')
plt.xlabel('Training Steps')
plt.ylabel('Mean Gradients')
plt.title('Mean Gradients during Training')
plt.grid(True)
plt.show()
```



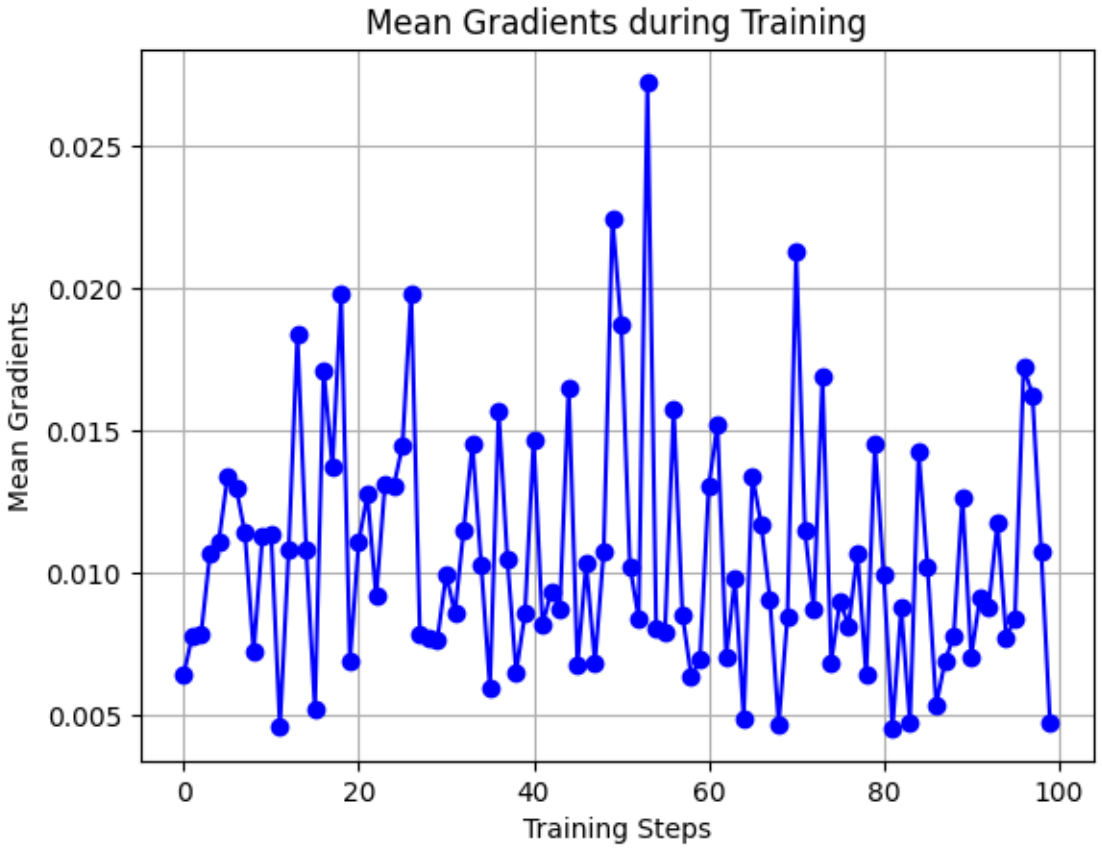
[90]:
```
steps=800
training_steps = [i for i in range(steps)]
plt.plot(training_steps, mean_gradients[:steps], marker='o', linestyle='-',␣
 ↪color='b')
plt.xlabel('Training Steps')
plt.ylabel('Mean Gradients')
plt.title('Mean Gradients during Training')
plt.grid(True)
plt.show()
```



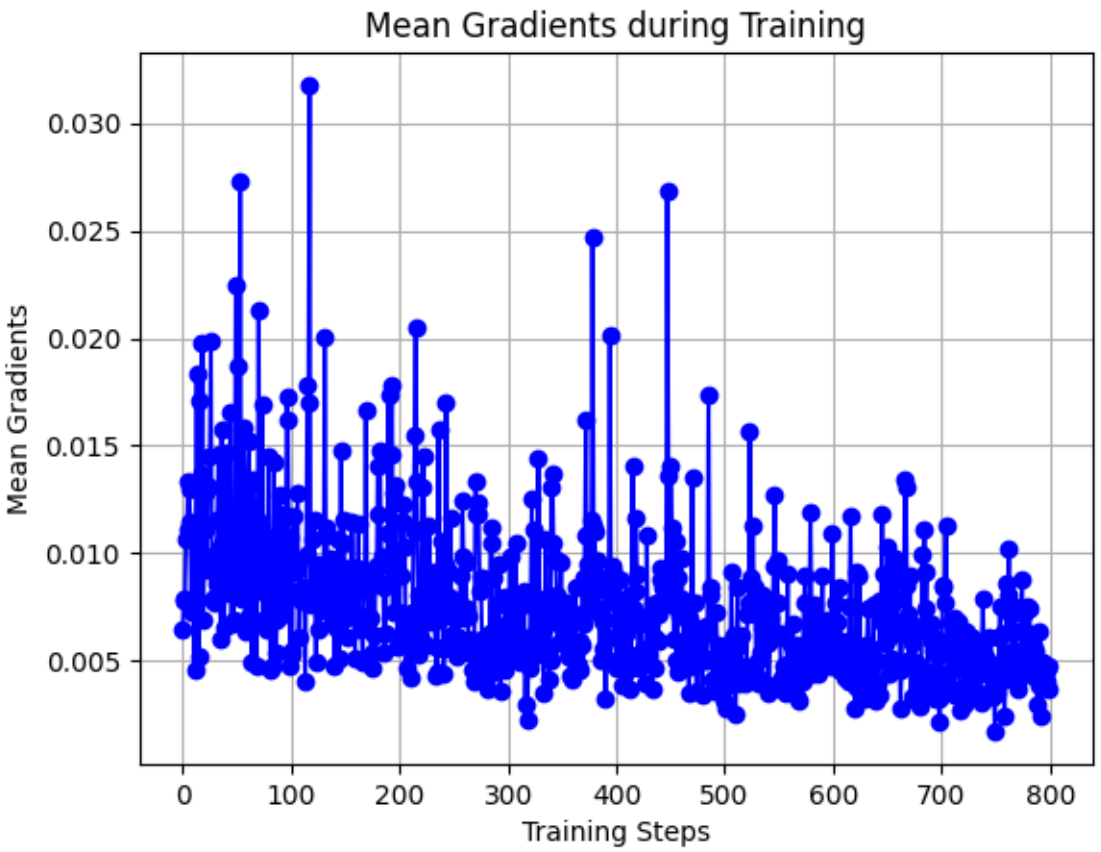
[95]:
```
model = NeuralNetwork().to(device)
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.8)
epochs = 1
gradients=[]
for t in range(epochs):
    mean_gradients=train(train_dataloader, model, loss_fn, optimizer)
print("Done!")
```

```
loss: 2.285686  [    0/60000]
loss: 1.685009  [ 6400/60000]
loss: 1.435848  [12800/60000]
loss: 1.071812  [19200/60000]
loss: 1.124079  [25600/60000]
loss: 1.127894  [32000/60000]
loss: 1.183237  [38400/60000]
loss: 0.948740  [44800/60000]
loss: 1.876494  [51200/60000]
loss: 0.763795  [57600/60000]
Done!
```

[97]:
```
steps=100
training_steps = [i for i in range(steps)]
plt.plot(training_steps, mean_gradients[:steps], marker='o', linestyle='-',␣
 ↪color='b')
plt.xlabel('Training Steps')
plt.ylabel('Mean Gradients')
plt.title('Mean Gradients during Training')
plt.grid(True)
plt.show()
```



[98]:
```
steps=800
training_steps = [i for i in range(steps)]
plt.plot(training_steps, mean_gradients[:steps], marker='o', linestyle='-',␣
 ↪color='b')
plt.xlabel('Training Steps')
plt.ylabel('Mean Gradients')
plt.title('Mean Gradients during Training')
plt.grid(True)
plt.show()
```



[3]:
```
#answer for question 1.6 the mean of the gradients decrease.
#We compare the models at two learning rates and set up the images at different␣
 ↪learning steps.
#We can find that when the learning step is 100, the mean of the gradient is␣
 ↪oscillating, and when
#the learning step is 800, it is obvious that the mean of the gradient is␣
 ↪gradually decreasing in
#the process of oscillating, which means that the gradient decreases gradually␣
 ↪and mildly through training.
```

[169]:
```
#code for question 1.7
class CONVNeuralNetwork(nn.Module):
    def __init__(self):
        super(CONVNeuralNetwork, self).__init__()
        self.conv1 = nn.Conv2d(1, 16, 3, 1, padding=1, bias=False)
        self.conv2 = nn.Conv2d(16, 32, 3, 1, padding=1, bias=False)
        self.linear = nn.Linear(25088, 10, bias=False)
    def set_grad(var):
        def hook(grad):
```

```
            var.grad = grad
        return hook
    def forward(self, x):
        x1 = self.conv1(x)
        x2 = self.conv2(x1)
        x3 = x2.view(x2.size(0), -1)
        x4 = self.linear(x3)
        return x4
model = CONVNeuralNetwork()
total_params = sum(p.numel() for p in model.parameters() if p.requires_grad)
print("Total Trainable Parameters:", total_params)
```

```
Total Trainable Parameters: 255632
```

[170]:
```
def train(dataloader, model, loss_fn, optimizer):
    size = len(dataloader.dataset)
    model.train()
    for batch, (X, y) in enumerate(dataloader):
        X, y = X.to(device), y.to(device)

        # Compute prediction error
        pred = model(X)
        loss = loss_fn(pred, y)

        # Backpropagation
        optimizer.zero_grad()
        loss.backward()
        optimizer.step()

        if batch % 100 == 0:
            loss, current = loss.item(), batch * len(X)
            print(f"loss: {loss:>7f}  [{current:>5d}/{size:>5d}]")
```

[171]:
```
def test(dataloader, model, loss_fn):
    size = len(dataloader.dataset)
    num_batches = len(dataloader)
    model.eval()
    test_loss, correct = 0, 0
    with torch.no_grad():
        for X, y in dataloader:
            X, y = X.to(device), y.to(device)
            pred = model(X)
            test_loss += loss_fn(pred, y).item()
            correct += (pred.argmax(1) == y).type(torch.float).sum().item()
    test_loss /= num_batches
    correct /= size
```

```
    print(f"Test Error: \n Accuracy: {(100*correct):>0.1f}%, Avg loss:␣
 ↪{test_loss:>8f} \n")
    return test_loss,correct
```

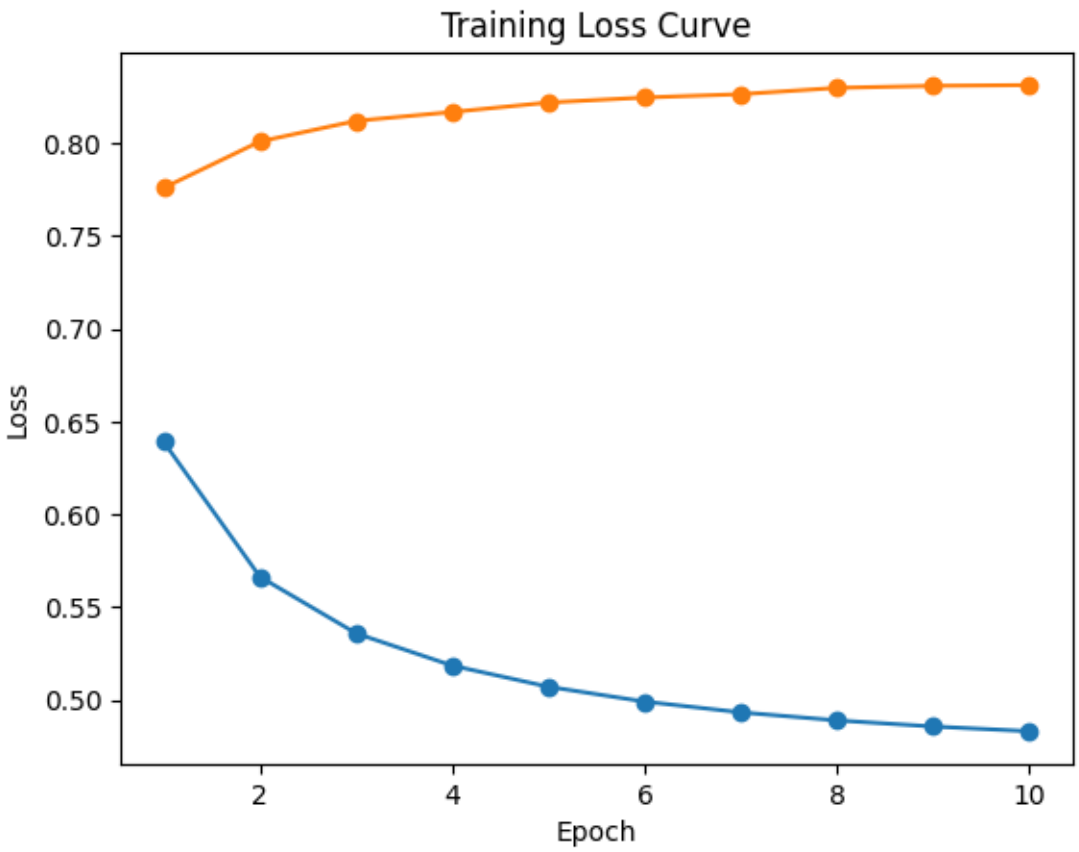
[172]:
```
losses = []
loss_fn = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=0.01)
epochs = 10
for t in range(epochs):
    print(f"Epoch {t+1}\n-------------------------------")
    train(train_dataloader, model, loss_fn, optimizer)
    loss = test(test_dataloader, model, loss_fn)
    losses.append(loss)
plt.plot(range(1, epochs+1), losses, marker='o')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.title('Training Loss Curve')
plt.show()
```

```
Epoch 1
-------------------------------
loss: 2.309434  [    0/60000]
loss: 0.849713  [ 6400/60000]
loss: 0.665767  [12800/60000]
loss: 0.752747  [19200/60000]
loss: 0.760533  [25600/60000]
loss: 0.593379  [32000/60000]
loss: 0.615206  [38400/60000]
loss: 0.647822  [44800/60000]
loss: 0.790064  [51200/60000]
loss: 0.471145  [57600/60000]
Test Error:
 Accuracy: 77.6%, Avg loss: 0.639007

Epoch 2
-------------------------------
loss: 0.543718  [    0/60000]
loss: 0.515979  [ 6400/60000]
loss: 0.394376  [12800/60000]
loss: 0.585516  [19200/60000]
loss: 0.526562  [25600/60000]
loss: 0.497756  [32000/60000]
loss: 0.507748  [38400/60000]
loss: 0.614087  [44800/60000]
loss: 0.715681  [51200/60000]
loss: 0.412486  [57600/60000]
Test Error:
```

```
 Accuracy: 80.1%, Avg loss: 0.566311

Epoch 3
-------------------------------
loss: 0.433677  [    0/60000]
loss: 0.463981  [ 6400/60000]
loss: 0.353289  [12800/60000]
loss: 0.550277  [19200/60000]
loss: 0.482604  [25600/60000]
loss: 0.468853  [32000/60000]
loss: 0.477101  [38400/60000]
loss: 0.590957  [44800/60000]
loss: 0.672588  [51200/60000]
loss: 0.393224  [57600/60000]
Test Error:
 Accuracy: 81.2%, Avg loss: 0.535801

Epoch 4
-------------------------------
loss: 0.394996  [    0/60000]
loss: 0.443858  [ 6400/60000]
loss: 0.334635  [12800/60000]
loss: 0.533021  [19200/60000]
loss: 0.463994  [25600/60000]
loss: 0.450593  [32000/60000]
loss: 0.463053  [38400/60000]
loss: 0.580188  [44800/60000]
loss: 0.644166  [51200/60000]
loss: 0.384538  [57600/60000]
Test Error:
 Accuracy: 81.7%, Avg loss: 0.518414

Epoch 5
-------------------------------
loss: 0.376640  [    0/60000]
loss: 0.434959  [ 6400/60000]
loss: 0.324571  [12800/60000]
loss: 0.520834  [19200/60000]
loss: 0.452115  [25600/60000]
loss: 0.437417  [32000/60000]
loss: 0.455971  [38400/60000]
loss: 0.575134  [44800/60000]
loss: 0.623667  [51200/60000]
loss: 0.379914  [57600/60000]
Test Error:
 Accuracy: 82.2%, Avg loss: 0.507020

Epoch 6
```

```
-------------------------------
loss: 0.366117  [    0/60000]
loss: 0.430478  [ 6400/60000]
loss: 0.319068  [12800/60000]
loss: 0.511243  [19200/60000]
loss: 0.443007  [25600/60000]
loss: 0.426696  [32000/60000]
loss: 0.452843  [38400/60000]
loss: 0.572510  [44800/60000]
loss: 0.608215  [51200/60000]
loss: 0.377293  [57600/60000]
Test Error:
 Accuracy: 82.5%, Avg loss: 0.499027

Epoch 7
-------------------------------
loss: 0.359203  [    0/60000]
loss: 0.427851  [ 6400/60000]
loss: 0.316156  [12800/60000]
loss: 0.503385  [19200/60000]
loss: 0.435577  [25600/60000]
loss: 0.417328  [32000/60000]
loss: 0.452081  [38400/60000]
loss: 0.570841  [44800/60000]
loss: 0.596346  [51200/60000]
loss: 0.375797  [57600/60000]
Test Error:
 Accuracy: 82.7%, Avg loss: 0.493216

Epoch 8
-------------------------------
loss: 0.354167  [    0/60000]
loss: 0.425973  [ 6400/60000]
loss: 0.314745  [12800/60000]
loss: 0.496784  [19200/60000]
loss: 0.429369  [25600/60000]
loss: 0.408864  [32000/60000]
loss: 0.452652  [38400/60000]
loss: 0.569460  [44800/60000]
loss: 0.587174  [51200/60000]
loss: 0.374972  [57600/60000]
Test Error:
 Accuracy: 83.0%, Avg loss: 0.488890

Epoch 9
-------------------------------
loss: 0.350185  [    0/60000]
loss: 0.424333  [ 6400/60000]
```

```
loss: 0.314211  [12800/60000]
loss: 0.491123  [19200/60000]
loss: 0.424108  [25600/60000]
loss: 0.401097  [32000/60000]
loss: 0.453869  [38400/60000]
loss: 0.568073  [44800/60000]
loss: 0.580079  [51200/60000]
loss: 0.374565  [57600/60000]
Test Error:
 Accuracy: 83.1%, Avg loss: 0.485604

Epoch 10
-------------------------------
loss: 0.346815  [    0/60000]
loss: 0.422720  [ 6400/60000]
loss: 0.314180  [12800/60000]
loss: 0.486176  [19200/60000]
loss: 0.419586  [25600/60000]
loss: 0.393919  [32000/60000]
loss: 0.455289  [38400/60000]
loss: 0.566564  [44800/60000]
loss: 0.574589  [51200/60000]
loss: 0.374438  [57600/60000]
Test Error:
 Accuracy: 83.1%, Avg loss: 0.483059
```



[5]:
```
#Answer for question 1.7:
#Findings:Convergence:We can see the CNN model convergers faster than␣
 ↪MLP,especially the several beginning epochs,
#which means the model learns informantion faster.
#Accuracy:the accuracy of CNN after 10 epochs is 83.1,which is slightly worse␣
 ↪than MLPs.
#Number of parameters:the number of parameters of CNN is 255632,much less than␣
 ↪the baseline model of MLPs,which means
#our CNN model is much simpler than MLPs model.
#In conclusion,this CNN models behaves faster learning,much simpler computation␣
 ↪and a slightly worse output,which is acceptable
#on some conditions with less arithmetic power and moderate requriment of␣
 ↪accuracy.
```