```
pip install torch torchvision
```

```
Requirement already satisfied: torch in /usr/local/lib/python3.10/dist-packages (2.2.1+cu121)
Requirement already satisfied: torchvision in /usr/local/lib/python3.10/dist-packages (0.17.1+cu121)
Requirement already satisfied: filelock in /usr/local/lib/python3.10/dist-packages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in /usr/local/lib/python3.10/dist-packages (from torch) (4.10.0)
Requirement already satisfied: sympy in /usr/local/lib/python3.10/dist-packages (from torch) (1.12)
Requirement already satisfied: networkx in /usr/local/lib/python3.10/dist-packages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in /usr/local/lib/python3.10/dist-packages (from torch) (3.1.3)
Requirement already satisfied: fsspec in /usr/local/lib/python3.10/dist-packages (from torch) (2023.6.0)
Collecting nvidia-cuda-nvrtc-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_nvrtc_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (23.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 23.7/23.7 MB 26.4 MB/s eta 0:00:00
Collecting nvidia-cuda-runtime-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_runtime_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (823 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 823.6/823.6 kB 49.7 MB/s eta 0:00:00
Collecting nvidia-cuda-cupti-cu12==12.1.105 (from torch)
  Downloading nvidia_cuda_cupti_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (14.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 14.1/14.1 MB 46.9 MB/s eta 0:00:00
Collecting nvidia-cudnn-cu12==8.9.2.26 (from torch)
  Downloading nvidia_cudnn_cu12-8.9.2.26-py3-none-manylinux1_x86_64.whl (731.7 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 731.7/731.7 MB 739.7 kB/s eta 0:00:00
Collecting nvidia-cublas-cu12==12.1.3.1 (from torch)
  Downloading nvidia_cublas_cu12-12.1.3.1-py3-none-manylinux1_x86_64.whl (410.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 410.6/410.6 MB 1.3 MB/s eta 0:00:00
Collecting nvidia-cufft-cu12==11.0.2.54 (from torch)
  Downloading nvidia_cufft_cu12-11.0.2.54-py3-none-manylinux1_x86_64.whl (121.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 121.6/121.6 MB 8.6 MB/s eta 0:00:00
Collecting nvidia-curand-cu12==10.3.2.106 (from torch)
  Downloading nvidia_curand_cu12-10.3.2.106-py3-none-manylinux1_x86_64.whl (56.5 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 56.5/56.5 MB 11.9 MB/s eta 0:00:00
Collecting nvidia-cusolver-cu12==11.4.5.107 (from torch)
  Downloading nvidia_cusolver_cu12-11.4.5.107-py3-none-manylinux1_x86_64.whl (124.2 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 124.2/124.2 MB 7.1 MB/s eta 0:00:00
Collecting nvidia-cusparse-cu12==12.1.0.106 (from torch)
  Downloading nvidia_cusparse_cu12-12.1.0.106-py3-none-manylinux1_x86_64.whl (196.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 196.0/196.0 MB 5.9 MB/s eta 0:00:00
Collecting nvidia-nccl-cu12==2.19.3 (from torch)
  Downloading nvidia_nccl_cu12-2.19.3-py3-none-manylinux1_x86_64.whl (166.0 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 166.0/166.0 MB 2.3 MB/s eta 0:00:00
Collecting nvidia-nvtx-cu12==12.1.105 (from torch)
  Downloading nvidia_nvtx_cu12-12.1.105-py3-none-manylinux1_x86_64.whl (99 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 99.1/99.1 kB 11.6 MB/s eta 0:00:00
Requirement already satisfied: triton==2.2.0 in /usr/local/lib/python3.10/dist-packages (from torch) (2.2.0)
Collecting nvidia-nvjitlink-cu12 (from nvidia-cusolver-cu12==11.4.5.107->torch)
  Downloading nvidia_nvjitlink_cu12-12.4.99-py3-none-manylinux2014_x86_64.whl (21.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 21.1/21.1 MB 62.3 MB/s eta 0:00:00
Requirement already satisfied: numpy in /usr/local/lib/python3.10/dist-packages (from torchvision) (1.25.2)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in /usr/local/lib/python3.10/dist-packages (from torchvision) (9.4.0)
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.10/dist-packages (from jinja2->torch) (2.1.5)
Requirement already satisfied: mpmath>=0.19 in /usr/local/lib/python3.10/dist-packages (from sympy->torch) (1.3.0)
Installing collected packages: nvidia-nvtx-cu12, nvidia-nvjitlink-cu12, nvidia-nccl-cu12, nvidia-curand-cu12, nvidia-cufft-cu12, nvidia-
Successfully installed nvidia-cublas-cu12-12.1.3.1 nvidia-cuda-cupti-cu12-12.1.105 nvidia-cuda-nvrtc-cu12-12.1.105 nvidia-cuda-runtime-c
```

+ Code + Text

Importing packages and libraries

```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.transforms as transforms
from torch.utils.data import DataLoader
```

Define the CNN architecture

```
class CNN(nn.Module):
    def __init__(self):
        super(CNN, self).__init__()
        self.conv1 = nn.Conv2d(3, 32, 3, padding=1)
        self.conv2 = nn.Conv2d(32, 64, 3, padding=1)
        self.conv3 = nn.Conv2d(64, 128, 3, padding=1)
        self.fc1 = nn.Linear(128 * 4 * 4, 512)
```

```
        self.fc2 = nn.Linear(512, 10)

    def forward(self, x):
        x = torch.relu(self.conv1(x))
        x = torch.max_pool2d(x, 2, 2)
        x = torch.relu(self.conv2(x))
        x = torch.max_pool2d(x, 2, 2)
        x = torch.relu(self.conv3(x))
        x = torch.max_pool2d(x, 2, 2)
        x = x.view(-1, 128 * 4 * 4)
        x = torch.relu(self.fc1(x))
        x = self.fc2(x)
        return x
```

Data preprocessing and loading

```
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])
```

Training the dataset

```
train_set = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
train_loader = DataLoader(train_set, batch_size=64, shuffle=True, num_workers=2)
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%|██████████| 170498071/170498071 [00:02<00:00, 80285451.49it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
```

Testing the dataset

```
test_set = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
test_loader = DataLoader(test_set, batch_size=64, shuffle=False, num_workers=2)
```

```
Files already downloaded and verified
```

Initialize the network, loss function, and optimizer

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = CNN().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.Adam(model.parameters(), lr=0.001)
```

Training loop

```
num_epochs = 10
for epoch in range(num_epochs):
    running_loss = 0.0
    for i, data in enumerate(train_loader, 0):
        inputs, labels = data[0].to(device), data[1].to(device)
        optimizer.zero_grad()
        outputs = model(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        running_loss += loss.item()
        if i % 100 == 99:    # Print every 100 mini-batches
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1, running_loss / 100))
            running_loss = 0.0

print('Finished Training')
```

```
[1,   100] loss: 1.871
[1,   200] loss: 1.582
[1,   300] loss: 1.428
[1,   400] loss: 1.340
[1,   500] loss: 1.241
[1,   600] loss: 1.187
[1,   700] loss: 1.116
[2,   100] loss: 1.014
[2,   200] loss: 0.966
[2,   300] loss: 0.927
[2,   400] loss: 0.910
[2,   500] loss: 0.886
[2,   600] loss: 0.882
[2,   700] loss: 0.869
[3,   100] loss: 0.753
[3,   200] loss: 0.729
[3,   300] loss: 0.712
[3,   400] loss: 0.704
[3,   500] loss: 0.715
[3,   600] loss: 0.673
[3,   700] loss: 0.688
[4,   100] loss: 0.593
[4,   200] loss: 0.580
[4,   300] loss: 0.582
[4,   400] loss: 0.581
[4,   500] loss: 0.585
[4,   600] loss: 0.559
[4,   700] loss: 0.558
[5,   100] loss: 0.438
[5,   200] loss: 0.455
[5,   300] loss: 0.465
[5,   400] loss: 0.455
[5,   500] loss: 0.454
[5,   600] loss: 0.445
[5,   700] loss: 0.464
[6,   100] loss: 0.320
[6,   200] loss: 0.308
[6,   300] loss: 0.340
[6,   400] loss: 0.357
[6,   500] loss: 0.353
[6,   600] loss: 0.374
[6,   700] loss: 0.366
[7,   100] loss: 0.216
[7,   200] loss: 0.227
[7,   300] loss: 0.221
[7,   400] loss: 0.270
[7,   500] loss: 0.255
[7,   600] loss: 0.278
[7,   700] loss: 0.297
[8,   100] loss: 0.164
[8,   200] loss: 0.152
[8,   300] loss: 0.173
[8,   400] loss: 0.176
[8,   500] loss: 0.195
[8,   600] loss: 0.192
[8,   700] loss: 0.220
[9,   100] loss: 0.106
[9,   200] loss: 0.113
```

Testing loop

```
correct = 0
total = 0
with torch.no_grad():
    for data in test_loader:
        inputs, labels = data[0].to(device), data[1].to(device)
        outputs = model(inputs)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
```

Predicting the accuracy

```
print('Accuracy of the network on the 10000 test images: %d %%' % (100 * correct / total))
```

```
Accuracy of the network on the 10000 test images: 75 %
```