Lab Assignment 8

Problem Statement

In this assignment, your goal is to build a classification model for the CIFAR-10 dataset using AlexNet architecture. The CIFAR-10 dataset consists of 60,000 32x32 color images in 10 classes, with 6,000 images per class. You are required to implement the solution using PyTorch.

Objective

Learn how to preprocess and load data using PyTorch utilities. Understand the implementation of AlexNet for image classification. Train the model on the CIFAR-10 dataset and evaluate its performance.

Hints to Solve the Problem

Framework

Use PyTorch to implement the solution. PyTorch provides modules for handling datasets, building neural network models, and training them efficiently.

Preprocessing

Since AlexNet expects larger input dimensions (224x224), you will need to resize the CIFAR-10 images. Apply the following transformations:

- Resize to 256x256.
- Perform center cropping to 224x224.
- Normalize the image data using mean `[0.485, 0.456, 0.406]` and standard deviation `[0.229, 0.224, 0.225]`.

Model Architecture

Use PyTorch's prebuilt AlexNet model (`torchvision.models.alexnet`). Modify the final fully connected layers in the classifier to match the 10 output classes of the CIFAR-10 dataset.

Training Loop

- Use `torch.optim.SGD` for the optimizer with learning rate 0.001 and momentum 0.9.
- Use `torch.nn.CrossEntropyLoss` for the loss function.
- Train the model over multiple epochs.
- Use `torch.utils.data.DataLoader` to load training and testing datasets in mini-batches.

Evaluation:

- After training, evaluate the model on the test set and calculate the accuracy.

Pseudocode for the Approach

- Import Libraries

- Prepare Dataset

  - Download CIFAR-10 dataset.
  - Apply transformations (resize, crop, normalize).
  - Use `torch.utils.data.DataLoader` for batching and shuffling.
- Load Pretrained AlexNet

  - Use torchvision.models.alexnet(pretrained=True).
  - Modify the classifier to have 10 output classes.
- Set Up Training

  - Define the loss function (torch.nn.CrossEntropyLoss).
  - Set up the optimizer (torch.optim.SGD).
- Training Loop

  - For each epoch:
    - Loop over mini-batches of the dataset.
    - Move inputs and labels to the device (CPU/GPU).
    - Perform forward pass, calculate loss, and backpropagate.
    - Update weights using the optimizer.
- Evaluate the Model

  - Calculate accuracy on the test dataset.

Hints on Specific PyTorch Functions

- Data Loading:

  - Use `torchvision.datasets.CIFAR10` for dataset loading and `torch.utils.data.DataLoader` for creating data loaders.
- Preprocessing Transformations:

  - Use `torchvision.transforms` for image resizing, cropping, and normalization.
- Model Architecture:

  - Use `torchvision.models.alexnet(pretrained=True)` to load the AlexNet model and modify the classifier for CIFAR-10.
- Training Utilities:

  - `torch.optim.SGD` for optimization.
  - `torch.nn.CrossEntropyLoss` for the loss function.
  - `model.to(device)` to move the model to GPU if available.

- Evaluation Metrics:
  - Calculate accuracy by comparing predicted labels with true labels.

In [ ]:
```
### WRITE CODE HERE ###
import torch
import torchvision
import torchvision.transforms as transforms
import torch.nn as nn
import torch.optim as optim
import time

transform = transforms.Compose([
    transforms.Resize(256),
    transforms.CenterCrop(224),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406], std=[0.229, 0.224,
                                                          0.225]),
])

# Download the CIFAR-10 dataset
train_data = torchvision.datasets.CIFAR10(root='./data', train=True,
                                         download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(train_data, batch_size=64,
                                                 shuffle=True, num_workers=2)
test_data = torchvision.datasets.CIFAR10(root='./data', train=False,
                                         download=True, transform=transform)
testloader = torch.utils.data.DataLoader(test_data, batch_size=64,
                                                      shuffle=False, num_workers=2)
classes = ('Airplane', 'Car', 'Bird', 'Cat', 'Deer', 'Dog', 'Frog',
                                                'Horse', 'Ship', 'Truck')
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifa
r-10-python.tar.gz
100%|██████████| 170M/170M [00:18<00:00, 9.09MB/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

In [ ]:
```
# Define the AlexNet model
model = torchvision.models.alexnet(pretrained=True)
model.classifier[1] = nn.Linear(9216,4096)
model.classifier[4] = nn.Linear(4096,1024)
model.classifier[6] = nn.Linear(1024,10)
model.eval()

criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.9)

# move the input and model to GPU for speed if available
device = torch.device("cuda:0" if torch.cuda.is_available() else "cpu")
model.to(device)
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarn
ing: The parameter 'pretrained' is deprecated since 0.13 and may be removed in the
future, please use 'weights' instead.
  warnings.warn(
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:223: UserWarn
ing: Arguments other than a weight enum or `None` for 'weights' are deprecated sin
ce 0.13 and may be removed in the future. The current behavior is equivalent to pa
ssing `weights=AlexNet_Weights.IMAGENET1K_V1`. You can also use `weights=AlexNet_W
eights.DEFAULT` to get the most up-to-date weights.
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/alexnet-owt-7be5be79.pth" to /ro
ot/.cache/torch/hub/checkpoints/alexnet-owt-7be5be79.pth
100%|██████████| 233M/233M [00:01<00:00, 174MB/s]
```

Out[ ]:

```
AlexNet(
  (features): Sequential(
    (0): Conv2d(3, 64, kernel_size=(11, 11), stride=(4, 4), padding=(2, 2))
    (1): ReLU(inplace=True)
    (2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=Fals
e)
    (3): Conv2d(64, 192, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (4): ReLU(inplace=True)
    (5): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=Fals
e)
    (6): Conv2d(192, 384, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (7): ReLU(inplace=True)
    (8): Conv2d(384, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (9): ReLU(inplace=True)
    (10): Conv2d(256, 256, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1))
    (11): ReLU(inplace=True)
    (12): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=Fals
e)
  )
  (avgpool): AdaptiveAvgPool2d(output_size=(6, 6))
  (classifier): Sequential(
    (0): Dropout(p=0.5, inplace=False)
    (1): Linear(in_features=9216, out_features=4096, bias=True)
    (2): ReLU(inplace=True)
    (3): Dropout(p=0.5, inplace=False)
    (4): Linear(in_features=4096, out_features=1024, bias=True)
    (5): ReLU(inplace=True)
    (6): Linear(in_features=1024, out_features=10, bias=True)
  )
)
```

In [ ]:

```python
for epoch in range(10): # loop over the dataset multiple times
    running_loss = 0.0
    start_time = time.time()

    for i, data in enumerate(trainloader, 0):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data[0].to(device), data[1].to(device)

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
        output = model(inputs)
        loss = criterion(output, labels)
        loss.backward()
        optimizer.step()

        # time
        end_time = time.time()
        time_taken = end_time - start_time
```

```
        # print statistics
        running_loss += loss.item()

        if i % 2000 == 0:
            print('[%d, %5d] loss: %.3f' % (epoch + 1, i + 1,
                                            running_loss / 2000))
            print('Time:',time_taken)
            running_loss = 0.0

print('Finished Training of AlexNet')
```

```
[1,     1] loss: 0.001
Time: 1.6806905269622803
[2,     1] loss: 0.000
Time: 0.4197852611541748
[3,     1] loss: 0.000
Time: 0.32155418395996094
[4,     1] loss: 0.000
Time: 0.48380613327026367
[5,     1] loss: 0.000
Time: 0.33129167556762695
[6,     1] loss: 0.000
Time: 0.3194406032562256
[7,     1] loss: 0.000
Time: 0.3374602794647217
[8,     1] loss: 0.000
Time: 0.3190732002258301
[9,     1] loss: 0.000
Time: 0.3065779209136963
[10,     1] loss: 0.000
Time: 0.3313891887664795
Finished Training of AlexNet
```

In [ ]:

```
correct = 0
total = 0
with torch.no_grad():
    for data in testloader:
        images, labels = data[0].to(device), data[1].to(device)
        outputs = model(images)
        _, predicted = torch.max(outputs.data, 1)
        total += labels.size(0)
        correct += (predicted == labels).sum().item()
        print('Accuracy of the network on the 10000 test images:
                      %.2f %%' % (100 * correct / total))
```

```
Accuracy of the network on the 10000 test images: 92.19 %
Accuracy of the network on the 10000 test images: 92.19 %
Accuracy of the network on the 10000 test images: 90.10 %
Accuracy of the network on the 10000 test images: 90.23 %
Accuracy of the network on the 10000 test images: 90.00 %
Accuracy of the network on the 10000 test images: 89.32 %
Accuracy of the network on the 10000 test images: 90.40 %
Accuracy of the network on the 10000 test images: 90.04 %
Accuracy of the network on the 10000 test images: 90.28 %
Accuracy of the network on the 10000 test images: 90.47 %
Accuracy of the network on the 10000 test images: 90.06 %
Accuracy of the network on the 10000 test images: 89.19 %
Accuracy of the network on the 10000 test images: 89.06 %
Accuracy of the network on the 10000 test images: 89.17 %
Accuracy of the network on the 10000 test images: 89.06 %
Accuracy of the network on the 10000 test images: 89.45 %
Accuracy of the network on the 10000 test images: 89.61 %
Accuracy of the network on the 10000 test images: 89.32 %
Accuracy of the network on the 10000 test images: 89.56 %
Accuracy of the network on the 10000 test images: 89.22 %
Accuracy of the network on the 10000 test images: 89.21 %
Accuracy of the network on the 10000 test images: 89.20 %
Accuracy of the network on the 10000 test images: 89.20 %
Accuracy of the network on the 10000 test images: 89.06 %
Accuracy of the network on the 10000 test images: 89.00 %
Accuracy of the network on the 10000 test images: 89.00 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 89.12 %
Accuracy of the network on the 10000 test images: 89.06 %
Accuracy of the network on the 10000 test images: 88.91 %
Accuracy of the network on the 10000 test images: 88.87 %
Accuracy of the network on the 10000 test images: 88.83 %
Accuracy of the network on the 10000 test images: 88.83 %
Accuracy of the network on the 10000 test images: 88.71 %
Accuracy of the network on the 10000 test images: 88.59 %
Accuracy of the network on the 10000 test images: 88.56 %
Accuracy of the network on the 10000 test images: 88.57 %
Accuracy of the network on the 10000 test images: 88.46 %
Accuracy of the network on the 10000 test images: 88.52 %
Accuracy of the network on the 10000 test images: 88.53 %
Accuracy of the network on the 10000 test images: 88.58 %
Accuracy of the network on the 10000 test images: 88.59 %
Accuracy of the network on the 10000 test images: 88.57 %
Accuracy of the network on the 10000 test images: 88.54 %
Accuracy of the network on the 10000 test images: 88.65 %
Accuracy of the network on the 10000 test images: 88.53 %
Accuracy of the network on the 10000 test images: 88.51 %
Accuracy of the network on the 10000 test images: 88.52 %
Accuracy of the network on the 10000 test images: 88.47 %
Accuracy of the network on the 10000 test images: 88.45 %
Accuracy of the network on the 10000 test images: 88.46 %
Accuracy of the network on the 10000 test images: 88.53 %
Accuracy of the network on the 10000 test images: 88.57 %
Accuracy of the network on the 10000 test images: 88.61 %
Accuracy of the network on the 10000 test images: 88.64 %
Accuracy of the network on the 10000 test images: 88.60 %
Accuracy of the network on the 10000 test images: 88.66 %
Accuracy of the network on the 10000 test images: 88.72 %
Accuracy of the network on the 10000 test images: 88.70 %
Accuracy of the network on the 10000 test images: 88.73 %
Accuracy of the network on the 10000 test images: 88.71 %
Accuracy of the network on the 10000 test images: 88.76 %
Accuracy of the network on the 10000 test images: 88.77 %
```

```
Accuracy of the network on the 10000 test images: 88.70 %
Accuracy of the network on the 10000 test images: 88.75 %
Accuracy of the network on the 10000 test images: 88.83 %
Accuracy of the network on the 10000 test images: 88.86 %
Accuracy of the network on the 10000 test images: 88.84 %
Accuracy of the network on the 10000 test images: 88.84 %
Accuracy of the network on the 10000 test images: 88.86 %
Accuracy of the network on the 10000 test images: 88.78 %
Accuracy of the network on the 10000 test images: 88.81 %
Accuracy of the network on the 10000 test images: 88.81 %
Accuracy of the network on the 10000 test images: 88.75 %
Accuracy of the network on the 10000 test images: 88.80 %
Accuracy of the network on the 10000 test images: 88.76 %
Accuracy of the network on the 10000 test images: 88.76 %
Accuracy of the network on the 10000 test images: 88.81 %
Accuracy of the network on the 10000 test images: 88.77 %
Accuracy of the network on the 10000 test images: 88.79 %
Accuracy of the network on the 10000 test images: 88.74 %
Accuracy of the network on the 10000 test images: 88.82 %
Accuracy of the network on the 10000 test images: 88.88 %
Accuracy of the network on the 10000 test images: 88.82 %
Accuracy of the network on the 10000 test images: 88.88 %
Accuracy of the network on the 10000 test images: 88.83 %
Accuracy of the network on the 10000 test images: 88.83 %
Accuracy of the network on the 10000 test images: 88.80 %
Accuracy of the network on the 10000 test images: 88.80 %
Accuracy of the network on the 10000 test images: 88.84 %
Accuracy of the network on the 10000 test images: 88.81 %
Accuracy of the network on the 10000 test images: 88.81 %
Accuracy of the network on the 10000 test images: 88.85 %
Accuracy of the network on the 10000 test images: 88.80 %
Accuracy of the network on the 10000 test images: 88.83 %
Accuracy of the network on the 10000 test images: 88.82 %
Accuracy of the network on the 10000 test images: 88.82 %
Accuracy of the network on the 10000 test images: 88.90 %
Accuracy of the network on the 10000 test images: 88.94 %
Accuracy of the network on the 10000 test images: 88.89 %
Accuracy of the network on the 10000 test images: 88.96 %
Accuracy of the network on the 10000 test images: 88.97 %
Accuracy of the network on the 10000 test images: 88.97 %
Accuracy of the network on the 10000 test images: 89.02 %
Accuracy of the network on the 10000 test images: 88.94 %
Accuracy of the network on the 10000 test images: 88.97 %
Accuracy of the network on the 10000 test images: 88.96 %
Accuracy of the network on the 10000 test images: 88.99 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.98 %
Accuracy of the network on the 10000 test images: 88.96 %
Accuracy of the network on the 10000 test images: 88.94 %
Accuracy of the network on the 10000 test images: 88.97 %
Accuracy of the network on the 10000 test images: 89.01 %
Accuracy of the network on the 10000 test images: 88.98 %
Accuracy of the network on the 10000 test images: 89.01 %
Accuracy of the network on the 10000 test images: 89.02 %
Accuracy of the network on the 10000 test images: 89.00 %
Accuracy of the network on the 10000 test images: 89.00 %
Accuracy of the network on the 10000 test images: 89.00 %
Accuracy of the network on the 10000 test images: 88.91 %
Accuracy of the network on the 10000 test images: 88.92 %
Accuracy of the network on the 10000 test images: 88.97 %
Accuracy of the network on the 10000 test images: 88.99 %
Accuracy of the network on the 10000 test images: 89.01 %
Accuracy of the network on the 10000 test images: 89.01 %
Accuracy of the network on the 10000 test images: 89.03 %
```

```
Accuracy of the network on the 10000 test images: 89.03 %
Accuracy of the network on the 10000 test images: 89.00 %
Accuracy of the network on the 10000 test images: 88.99 %
Accuracy of the network on the 10000 test images: 88.98 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.92 %
Accuracy of the network on the 10000 test images: 88.91 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.95 %
Accuracy of the network on the 10000 test images: 88.97 %
Accuracy of the network on the 10000 test images: 89.00 %
Accuracy of the network on the 10000 test images: 89.02 %
Accuracy of the network on the 10000 test images: 89.07 %
Accuracy of the network on the 10000 test images: 89.02 %
Accuracy of the network on the 10000 test images: 89.04 %
Accuracy of the network on the 10000 test images: 89.04 %
Accuracy of the network on the 10000 test images: 89.05 %
Accuracy of the network on the 10000 test images: 89.06 %
Accuracy of the network on the 10000 test images: 89.07 %
Accuracy of the network on the 10000 test images: 89.08 %
Accuracy of the network on the 10000 test images: 89.11 %
Accuracy of the network on the 10000 test images: 89.13 %
Accuracy of the network on the 10000 test images: 89.09 %
Accuracy of the network on the 10000 test images: 89.07 %
Accuracy of the network on the 10000 test images: 89.04 %
Accuracy of the network on the 10000 test images: 89.04 %
```

In [ ]:

```
```