Advanced Computer vision

August 15, 2024

0.1 ADVANCED COMPUTER VISION

[5]:
```
!pip install torch torchvision
```

```
Requirement already satisfied: torch in c:\users\user\anaconda3\lib\site-
packages (2.4.0)
Requirement already satisfied: torchvision in c:\users\user\anaconda3\lib\site-
packages (0.19.0)
Requirement already satisfied: filelock in c:\users\user\anaconda3\lib\site-
packages (from torch) (3.13.1)
Requirement already satisfied: typing-extensions>=4.8.0 in
c:\users\user\anaconda3\lib\site-packages (from torch) (4.11.0)
Requirement already satisfied: sympy in c:\users\user\anaconda3\lib\site-
packages (from torch) (1.12)
Requirement already satisfied: networkx in c:\users\user\anaconda3\lib\site-
packages (from torch) (3.2.1)
Requirement already satisfied: jinja2 in c:\users\user\anaconda3\lib\site-
packages (from torch) (3.1.4)
Requirement already satisfied: fsspec in c:\users\user\anaconda3\lib\site-
packages (from torch) (2024.3.1)
Requirement already satisfied: setuptools in c:\users\user\anaconda3\lib\site-
packages (from torch) (69.5.1)
Requirement already satisfied: numpy in c:\users\user\anaconda3\lib\site-
packages (from torchvision) (1.26.4)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
c:\users\user\anaconda3\lib\site-packages (from torchvision) (10.3.0)
Requirement already satisfied: MarkupSafe>=2.0 in
c:\users\user\anaconda3\lib\site-packages (from jinja2->torch) (2.1.3)
Requirement already satisfied: mpmath>=0.19 in c:\users\user\anaconda3\lib\site-
packages (from sympy->torch) (1.3.0)
```

[7]:
```
import torch
import torchvision
import torchvision.transforms as transforms
```

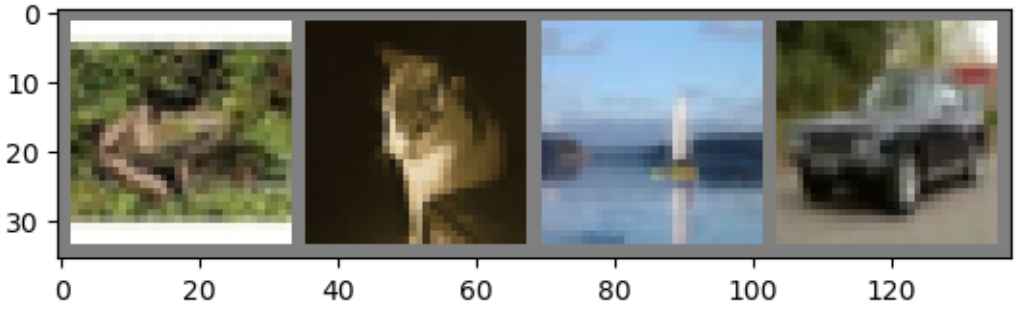
[11]:
```
transform = transforms.Compose(
    [transforms.ToTensor(),
     transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))])
```

```
batch_size = 4

trainset = torchvision.datasets.CIFAR10(root='./data', train=True,
                                        download=True, transform=transform)
trainloader = torch.utils.data.DataLoader(trainset, batch_size=batch_size,
                                          shuffle=True, num_workers=2)

testset = torchvision.datasets.CIFAR10(root='./data', train=False,
                                       download=True, transform=transform)

testloader = torch.utils.data.DataLoader(testset, batch_size=batch_size,
                                         shuffle=False, num_workers=2)

classes = ('plane', 'car', 'bird', 'cat',
           'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

```
Files already downloaded and verified
Files already downloaded and verified
```

[13]:
```
import matplotlib.pyplot as plt
import numpy as np
```

[15]:
```
def imshow(img):
    img = img/2 + 0.5
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

detaiter = iter(trainloader)
images, labels = next(detaiter)
imshow(torchvision.utils.make_grid(images))
print(' '.join(f'{classes[labels[j]]:5s}]' for j in range(batch_size)))
```




```
frog ] cat  ] ship ] car  ]
```

[ ]:

[16]:
```
import torch.nn as nn
import torch.nn.functional as F


class Net(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(3, 6, 5)
        self.pool = nn.MaxPool2d(2, 2)
        self.conv2 = nn.Conv2d(6, 16, 5)
        self.fc1 = nn.Linear(16 * 5 * 5, 120)
        self.fc2 = nn.Linear(120, 84)
        self.fc3 = nn.Linear(84, 10)


    def forward(self, x):
        x = self.pool(F.relu(self.conv1(x)))
        x = self.pool(F.relu(self.conv2(x)))
        x = torch.flatten(x, 1) #flatten all dimensions
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

[ ]:

[19]:
```
import torch.optim as optim

criterion = nn.CrossEntropyLoss()

optimizer = optim.SGD(net.parameters(), lr=0.001, momentum = 0.9 )
```

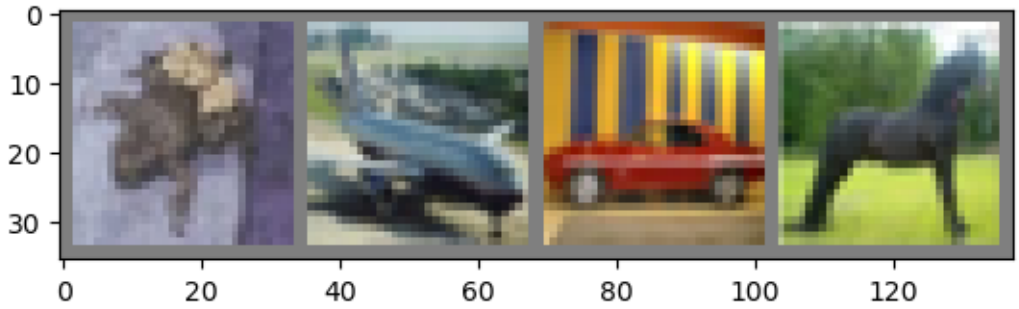
[23]:
```
for epoch in range(2):
    running_loss = 0.0
    for i, data in enumerate(trainloader, 0):
        # get the inputs; data is a list of [inputs, labels]
        inputs, labels = data

        # zero the parameter gradients
        optimizer.zero_grad()

        # forward + backward + optimize
```

```
        outputs = net(inputs)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()

        # print statistics
        running_loss+= loss.item()
        if i % 2000 == 1999:  # print every 2000 mini-batches
            print(f'[{epoch + 1}, {i + 1:5d}] loss: {running_loss / 2000:.3f}')
            running_loss = 0.0

print('Finished Training')
```

```
[1,  2000] loss: 1.244
[1,  4000] loss: 1.244
[1,  6000] loss: 1.241
[1,  8000] loss: 1.233
[1, 10000] loss: 1.238
[1, 12000] loss: 1.196
[2,  2000] loss: 1.155
[2,  4000] loss: 1.137
[2,  6000] loss: 1.160
[2,  8000] loss: 1.146
[2, 10000] loss: 1.135
[2, 12000] loss: 1.151
Finished Training
```

[ ]:

[25]:
```
dataiter = iter(testloader)
images, labels = next(detaiter)

# print images
imshow(torchvision.utils.make_grid(images))
print('GroundTruth:', ' '.join(f'{classes[labels[j]]:5s}' for j in range(4)))
```



```
GroundTruth: frog  plane car   horse
```

Ruth.O.Ajagunna