Lab Report 5

Introduction to PyTorch

Goutham Krishnan

21BAI1007

Aim

1. PyTorch basics and classification of CIFAR-10 dataset
2. PyTorch and GPU
3. GoogLeNet using PyTorch
4. Implementing VGG16 using PyTorch for classifying the CIFAR-10 dataset
5. Implementing ResNet using PyTorch for classifying the CIFAR-10 dataset
6. Implementing GoogleNet using PyTorch for classifying the CIFAR-10 dataset and comparing it with the reults of VGG16

Observations and Code

PyTorch basics and classification of the CIFAR-10 dataset

```
import torch
import numpy as np
```

```
data = [[1, 2], [3, 4]]
x_data = torch.tensor(data)
```

```
np_array = np.array(data)
x_np = torch.from_numpy(np_array)
```

```
x_ones = torch.ones_like(x_data)
print(f"Ones Tensor: \n{x_ones} \n")

x_rand = torch.rand_like(x_data, dtype=torch.float)
print(f"Random Tensor: \n{x_rand} \n")
```

```
Ones Tensor:
tensor([[1, 1],
        [1, 1]])

Random Tensor:
tensor([[0.2310, 0.5999],
        [0.7543, 0.1943]])
```

```
shape = (2, 3,)

rand_tensor = torch.rand(shape)
ones_tensor = torch.ones(shape)
zeros_tensor = torch.zeros(shape)

print(f"Random Tensor: \n {rand_tensor} \n")
print(f"Ones Tensor: \n {ones_tensor} \n")
print(f"Zeros Tensor: \n {zeros_tensor}")
```

21BAI1007-IntrotoTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```
Random Tensor:
 tensor([[0.6495, 0.3243, 0.7834],
        [0.1601, 0.3088, 0.3156]])

Ones Tensor:
 tensor([[1., 1., 1.],
        [1., 1., 1.]])

Zeros Tensor:
 tensor([[0., 0., 0.],
        [0., 0., 0.]])
```

```
tensor = torch.rand(3, 4)

print(f"Shape of tensor: {tensor.shape}")
print(f"Datatype of tensor: {tensor.dtype}")
print(f"Device tensor is stored on: {tensor.device}")
```

```
Shape of tensor: torch.Size([3, 4])
Datatype of tensor: torch.float32
Device tensor is stored on: cpu
```

```
if torch.cuda.is_available():
  tensor = tensor.to('cuda')
  print(f"Device tensor is stored on: {tensor.device}")
```

```
Device tensor is stored on: cuda:0
```

```
tensor = torch.ones(4, 4)
tensor[:,1] = 0
print(tensor)
```

```
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])
```

21BAI1007-IntrotoTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```
t1 = torch.cat([tensor, tensor, tensor], dim=1)
print(t1)
```

```
tensor([[1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.],
        [1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.],
        [1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.],
        [1., 0., 1., 1., 1., 0., 1., 1., 1., 0., 1., 1.]])
```

```
print(f"tensor.mul(tensor) \n {tensor.mul(tensor)} \n")
print(f"tensor * tensor \n {tensor * tensor}")
```

```
tensor.mul(tensor)
 tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])

tensor * tensor
 tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])
```

```
print(tensor, "\n")
tensor.add_(5)
print(tensor)
```

```
tensor([[1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.],
        [1., 0., 1., 1.]])

tensor([[6., 5., 6., 6.],
        [6., 5., 6., 6.],
        [6., 5., 6., 6.],
        [6., 5., 6., 6.]])
```

21BAI1007-IntrotoTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```
t = torch.ones(5)
print(f"t: {t}")
n = t.numpy()
print(f"n: {n}")
```

```
t: tensor([1., 1., 1., 1., 1.])
n: [1. 1. 1. 1. 1.]
```

```
t.add_(1)
print(f"t: {t}")
print(f"n: {n}")
```

```
t: tensor([2., 2., 2., 2., 2.])
n: [2. 2. 2. 2. 2.]
```

```
n = np.ones(5)
t = torch.from_numpy(n)
```

```
np.add(n, 1, out=n)
print(f"t: {t}")
print(f"n: {n}")
```

```
t: tensor([2., 2., 2., 2., 2.], dtype=torch.float64)
n: [2. 2. 2. 2. 2.]
```

```
import torch
from torchvision.models import resnet18, ResNet18_Weights
model = resnet18(weights=ResNet18_Weights.DEFAULT)
data = torch.rand(1, 3, 64, 64)
labels = torch.rand(1, 1000)
```

```
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100%|██████████| 44.7M/44.7M [00:00<00:00, 172MB/s]
```

21BAI1007-IntrotoTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```
[ ] prediction = model(data)
```

```
[ ] loss = (prediction - labels).sum()
    loss.backward()
```

```
[ ] optim = torch.optim.SGD(model.parameters(), lr=1e-2, momentum=0.9)
```

```
[ ] optim.step()
```

```
import torch
import torch.nn as nn
import torch.nn.functional as F

class Net(nn.Module):

    def __init__(self):
        super(Net, self).__init__()

        self.conv1 = nn.Conv2d(1, 6, 5)
        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 = F.max_pool2d(F.relu(self.conv1(x)), (2, 2))
        x = F.max_pool2d(F.relu(self.conv2(x)), 2)
        x = torch.flatten(x, 1)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
print(net)
```

21BAI1007-IntrotoTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```
[ ] Net(
      (conv1): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1))
      (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
      (fc1): Linear(in_features=400, out_features=120, bias=True)
      (fc2): Linear(in_features=120, out_features=84, bias=True)
      (fc3): Linear(in_features=84, out_features=10, bias=True)
    )
```

```
[ ] params = list(net.parameters())
    print(len(params))
    print(params[0].size())
```

```
10
torch.Size([6, 1, 5, 5])
```

```
[ ] input = torch.randn(1, 1, 32, 32)
    out = net(input)
    print(out)
```

```
tensor([[ 0.0430, -0.0843, -0.0752, -0.0426,  0.1442, -0.1004,  0.1183,  0.1131,
         -0.1151, -0.1125]], grad_fn=<AddmmBackward0>)
```

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

21BAI1007-IntrotoTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```
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')
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%|[redacted]| 170498071/170498071 [00:15<00:00, 10693577.83it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

```
import matplotlib.pyplot as plt
import numpy as np



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


dataiter = iter(trainloader)
images, labels = next(dataiter)

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

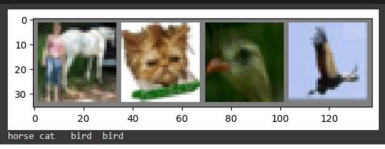
21BAI1007-IntrotoTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

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


dataiter = iter(trainloader)
images, labels = next(dataiter)

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



```
horse cat   bird  bird
```

PyTorch and GPU

21BAI1007-GPU&PyTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

```
import torch
import torchvision
import torchvision.transforms as transforms
```

```
[ ] device = torch.device('cuda:0' if torch.cuda.is_available() else 'cpu')

    print(device)
```

```
cpu
```

```
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')
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%|[redacted]| 170498071/170498071 [00:03<00:00, 44363955.35it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

21BAI1007-GPU&PyTorch.ipynb

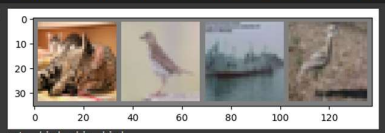
File Edit View Insert Runtime Tools Help Last edited on 1 March

```
import matplotlib.pyplot as plt
import numpy as np


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


dataiter = iter(trainloader)
images, labels = next(dataiter)

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



```
cat   bird  ship  bird
```

21BAI1007-GPU&PyTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```python
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)
        x = F.relu(self.fc1(x))
        x = F.relu(self.fc2(x))
        x = self.fc3(x)
        return x


net = Net()
```

```python
net.to(device)
```

```
Net(
  (conv1): Conv2d(3, 6, kernel_size=(5, 5), stride=(1, 1))
  (pool): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
  (conv2): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
  (fc1): Linear(in_features=400, out_features=120, bias=True)
  (fc2): Linear(in_features=120, out_features=84, bias=True)
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

21BAI1007-GPU&PyTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text

```python
import torch.optim as optim

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

```python
for epoch in range(2):
    running_loss = 0.0
    for i, (inputs, labels) in enumerate(trainloader, 0):
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()

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

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

print('Finished Training')
```

```
[1,  2000] loss: 2.243
[1,  4000] loss: 1.861
[1,  6000] loss: 1.686
[1,  8000] loss: 1.587
[1, 10000] loss: 1.553
[1, 12000] loss: 1.472
[2,  2000] loss: 1.418
[2,  4000] loss: 1.386
[2,  6000] loss: 1.365
[2,  8000] loss: 1.347
[2, 10000] loss: 1.335
[2, 12000] loss: 1.302
Finished Training
```

21BAI1007-GPU&PyTorch.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

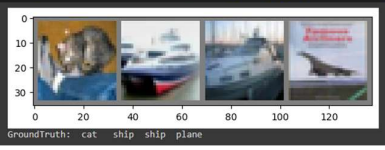
+ Code + Text

```
Finished Training
```

```python
PATH = './cifar_net.pth'
torch.save(net.state_dict(), PATH)
```

```python
dataiter = iter(testloader)
images, labels = next(dataiter)

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




```
GroundTruth:  cat  ship  ship  plane
```

GoogLeNet using PyTorch

21BAI1007-GoogleNet.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

```python
import torch
model = torch.hub.load('pytorch/vision:v0.10.0', 'googlenet', pretrained=True)
model.eval()
```

```
Downloading: "https://github.com/pytorch/vision/zipball/v0.10.0" to /root/.cache/torch/hub/v0.10.0.zip
Downloading: "https://download.pytorch.org/models/googlenet-1378be20.pth" to /root/.cache/torch/hub/checkpoints/googlenet-1378be20.pth
100%|[redacted]| 49.7M/49.7M [00:01<00:00, 46.1MB/s]
GoogLeNet(
  (conv1): BasicConv2d(
    (conv): Conv2d(3, 64, kernel_size=(7, 7), stride=(2, 2), padding=(3, 3), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (maxpool1): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=True)
  (conv2): BasicConv2d(
    (conv): Conv2d(64, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
    (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (conv3): BasicConv2d(
    (conv): Conv2d(64, 192, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
    (bn): BatchNorm2d(192, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
  )
  (maxpool2): MaxPool2d(kernel_size=3, stride=2, padding=0, dilation=1, ceil_mode=True)
  (inception3a): Inception(
    (branch1): BasicConv2d(
      (conv): Conv2d(192, 64, kernel_size=(1, 1), stride=(1, 1), bias=False)
      (bn): BatchNorm2d(64, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
    )
    (branch2): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(192, 96, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(96, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
      (1): BasicConv2d(
        (conv): Conv2d(96, 128, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1), bias=False)
        (bn): BatchNorm2d(128, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
      )
    )
    (branch3): Sequential(
      (0): BasicConv2d(
        (conv): Conv2d(192, 16, kernel_size=(1, 1), stride=(1, 1), bias=False)
        (bn): BatchNorm2d(16, eps=0.001, momentum=0.1, affine=True, track_running_stats=True)
```

21BAI1007-GoogleNet.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

```python
import urllib
url, filename = ("https://www.petfinder.com/sites/default/files/images/content/great-pyrenees-detail-scaled.jpg", "dog1.jpg")
try: urllib.URLopener().retrieve(url, filename)
except: urllib.request.urlretrieve(url, filename)
```

Start coding or generate with AI.

```python
from PIL import Image
from torchvision import transforms
input_image = Image.open(filename)
preprocess = 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]),
])
input_tensor = preprocess(input_image)
input_batch = input_tensor.unsqueeze(0)

if torch.cuda.is_available():
    input_batch = input_batch.to('cuda')
    model.to('cuda')

with torch.no_grad():
    output = model(input_batch)
print(output[0])
probabilities = torch.nn.functional.softmax(output[0], dim=0)
print(probabilities)
```

```
 5.3578e-01,  2.9988e+00, -2.2399e+00, -1.0880e+00, -1.6731e+00,
 7.7500e-01, -1.9073e-02,  9.5905e+00,  3.7346e-01,  1.8798e+00,
-2.3966e-01,  1.8589e+00,  3.2657e-01,  3.7690e+00,  3.7703e+00,
-3.6041e-01,  2.5268e+00,  2.4819e+00,  1.8436e+00,  2.1437e+00,
 1.8793e+00, -1.2440e+00, -3.2574e+00,  6.3516e-01,  1.2638e+00,
 1.3484e+00,  1.1224e+00, -1.0272e+00,  1.4962e+00,  5.6659e+00,
-1.5790e+00,  6.2905e-02,  5.2499e+00,  2.8096e+00,  3.5361e+00,
 1.5872e+00,  2.8156e-01, -4.1983e-01, -1.6286e+00,  5.3392e-01,
 4.5186e+00,  5.4759e+00,  9.4162e+00,  5.5186e+00, -1.0489e+00,
 4.3244e+00,  1.2259e+00, -3.9555e-02,  2.3952e-01,  3.8589e-02,
-2.6092e-01,  5.7976e-01,  1.8086e+00, -1.9968e+00,  5.7119e-01,
 2.2774e+00, -1.0974e+00, -6.9003e-01, -3.3932e-01, -1.8294e+00,
-2.5347e+00, -1.2625e+00, -1.9331e+00, -1.6466e+00,  1.0382e+00,
-8.7401e-01, -5.1659e-01,  5.0003e-01, -3.0772e-01, -9.7351e-01,
```

21BAI1007-GoogleNet.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

```
-1.1807e-01, -2.6179e-01,  4.6785e-01, -4.8546e-01, -6.2394e-01,
-9.2121e-01,  1.4884e+00,  3.0323e-01, -5.5521e-01,  4.0046e-01,
-1.2766e+00, -3.6673e-01, -2.6146e-01,  2.3016e-01, -3.4320e-01,
```

```python
# Download ImageNet labels
!wget https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt
```

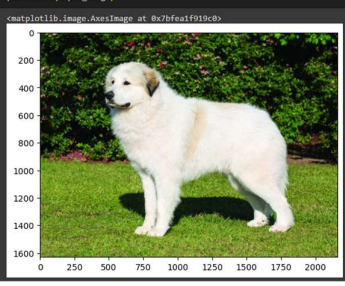
```
--2024-03-01 09:54:47--  https://raw.githubusercontent.com/pytorch/hub/master/imagenet_classes.txt
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.111.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.111.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 10472 (10K) [text/plain]
Saving to: ‘imagenet_classes.txt.2’

imagenet_classes.tx 100%[===================>]  10.23K  --.-KB/s    in 0.001s

2024-03-01 09:54:47 (10.2 MB/s) - ‘imagenet_classes.txt.2’ saved [10472/10472]
```

```python
plt.imshow(input_image)
```

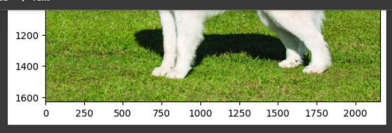
```
<matplotlib.image.AxesImage at 0x7bfea1f919c0>
```



21BAI1007-GoogleNet.ipynb

File Edit View Insert Runtime Tools Help Last edited on 1 March

+ Code + Text




```
with open("imagenet_classes.txt", "r") as f:
    categories = [s.strip() for s in f.readlines()]
top5_prob, top5_catid = torch.topk(probabilities, 5)
for i in range(top5_prob.size(0)):
    print(categories[top5_catid[i]], top5_prob[i].item())
```

```
kuvasz 0.4887969493865967
Great Pyrenees 0.41863711047172546
Tibetan mastiff 0.009654349647462368
golden retriever 0.008967381902039051
Samoyed 0.00833194050937891
```

Implementing VGG16 using PyTorch for classifying the CIFAR-10 dataset

21BAI1007-VGG16.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

Implementing VGG16 - 21BAI1007 Goutham Krishnan

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
from torchvision import models
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

```
num_epochs = 1
batch_size = 40
learning_rate = 0.001
classes = ('plane', 'car' , 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

```
transform = transforms.Compose([
        transforms.Resize(size=(224, 224)),
        transforms.ToTensor(),
        transforms.Normalize(
           (0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)
        )
    ])
train_dataset = torchvision.datasets.CIFAR10(root= './data', train = True, download =True, transform = transform)
test_dataset = torchvision.datasets.CIFAR10(root= './data', train = False, download =True, transform = transform)
```

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

21BAI1007-VGG16.ipynb

File Edit View Insert Runtime Tools Help All changes saved

+ Code + Text

```
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size = batch_size, shuffle = True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size = batch_size, shuffle = True)
n_total_step = len(train_loader)
print(n_total_step)
```

```
1250
```

```
model = models.vgg16(pretrained = True)
input_lastLayer = model.classifier[6].in_features
model.classifier[6] = nn.Linear(input_lastLayer,10)
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr = learning_rate, momentum=0.9,weight_decay=5e-4)
```

```
for epoch in range(num_epochs):
    for i, (imgs, labels) in enumerate(train_loader):
        imgs, labels = imgs.to(device), labels.to(device)

        labels_hat = model(imgs)
        n_corrects = (labels_hat.argmax(axis=1) == labels).sum().item()
        loss_value = criterion(labels_hat, labels)

        loss_value.backward()
        optimizer.step()
        optimizer.zero_grad()

        if (i + 1) % 250 == 0:
            print(f'epoch {epoch + 1}/{num_epochs}, step: {i + 1}/{len(train_loader)}: loss = {loss_value:.5f}, acc = {100 * (n_corrects / labels.size(0)):.2f}%')

    print()
```

```
epoch 1/1, step: 250/1250: loss = 0.41883, acc = 82.50%
epoch 1/1, step: 500/1250: loss = 0.69434, acc = 70.00%
epoch 1/1, step: 750/1250: loss = 0.48321, acc = 87.50%
epoch 1/1, step: 1000/1250: loss = 0.40966, acc = 82.50%
epoch 1/1, step: 1250/1250: loss = 0.10697, acc = 97.50%
```

21BAI1007-VGG16.ipynb

```
epoch 1/1, step: 750/1250: loss = 0.48321, acc = 87.50%
epoch 1/1, step: 1000/1250: loss = 0.40966, acc = 82.50%
epoch 1/1, step: 1250/1250: loss = 0.10697, acc = 97.50%
```

```
with torch.no_grad():
    number_corrects = 0
    number_samples = 0
    for i, (test_images_set , test_labels_set) in enumerate(test_loader):
        test_images_set = test_images_set.to(device)
        test_labels_set = test_labels_set.to(device)

        y_predicted = model(test_images_set)
        labels_predicted = y_predicted.argmax(axis = 1)
        number_corrects += (labels_predicted==test_labels_set).sum().item()
        number_samples += test_labels_set.size(0)
    print(f'Overall accuracy {(number_corrects / number_samples)*100}%')
```

```
Overall accuracy 89.64999999999999%
```

Implementing ResNet using PyTorch for classifying the CIFAR-10 dataset

21BAI1007-Resnet.ipynb

Implementing ResNet - 21BAI1007 Goutham Krishnan

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
from torchvision import models
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)
```

```
cuda
```

```
num_epochs = 1
batch_size = 40
learning_rate = 0.001
classes = ('plane', 'car' , 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')
```

21BAI1007-Resnet.ipynb

```
transform = transforms.Compose([
    transforms.Resize(size=(224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(
      (0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)
    )
])
train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)
```

```
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%|██████████| 170498071/170498071 [00:03<00:00, 43125975.10it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
```

```
train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size, shuffle=True)
n_total_step = len(train_loader)
print(n_total_step)
```

```
1250
```

```
model = models.resnet18(pretrained=True)
# Modify the last layer for CIFAR-10 (since it's originally trained for ImageNet)
model.fc = nn.Linear(model.fc.in_features, 10)
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate, momentum=0.9, weight_decay=5e-4)
```

```
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: 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: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equ
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/resnet18-f37072fd.pth" to /root/.cache/torch/hub/checkpoints/resnet18-f37072fd.pth
100%|██████████| 44.7M/44.7M [00:00<00:00, 135MB/s]
```

21BAI1007-Resnet.ipynb

```
for epoch in range(num_epochs):
    for i, (imgs, labels) in enumerate(train_loader):
        imgs, labels = imgs.to(device), labels.to(device)

        labels_hat = model(imgs)
        n_corrects = (labels_hat.argmax(axis=1) == labels).sum().item()
        loss_value = criterion(labels_hat, labels)

        loss_value.backward()
        optimizer.step()
        optimizer.zero_grad()

        if (i + 1) % 250 == 0:
            print(f'epoch {epoch + 1}/{num_epochs}, step: {i + 1}/{len(train_loader)}: loss = {loss_value:.5f}, acc = {100 * (n_corrects / labels.size(0)):.2f}%')

    print()
```

```
epoch 1/1, step: 250/1250: loss = 0.22127, acc = 95.00%
epoch 1/1, step: 500/1250: loss = 0.37424, acc = 85.00%
epoch 1/1, step: 750/1250: loss = 0.23278, acc = 90.00%
epoch 1/1, step: 1000/1250: loss = 0.19245, acc = 97.50%
epoch 1/1, step: 1250/1250: loss = 0.31020, acc = 82.50%
```

21BAI1007-Resnet.ipynb

```
epoch 1/1, step: 750/1250: loss = 0.23278, acc = 90.00%
epoch 1/1, step: 1000/1250: loss = 0.19245, acc = 97.50%
epoch 1/1, step: 1250/1250: loss = 0.31020, acc = 82.50%
```

```
with torch.no_grad():
    number_corrects = 0
    number_samples = 0
    for i, (test_images_set , test_labels_set) in enumerate(test_loader):
        test_images_set = test_images_set.to(device)
        test_labels_set = test_labels_set.to(device)

        y_predicted = model(test_images_set)
        labels_predicted = y_predicted.argmax(axis = 1)
        number_corrects += (labels_predicted==test_labels_set).sum().item()
        number_samples += test_labels_set.size(0)
    print(f'Overall accuracy {(number_corrects / number_samples)*100}%')
```

```
Overall accuracy 91.88%
```

Implementing GoogleNet using PyTorch for classifying the CIFAR-10 dataset and comparing it with the reults of VGG16

21BAI1007-HOTS-GoogleNet.ipynb

```
import torch
import torch.nn as nn
import torch.nn.functional as F
import torchvision
import torchvision.transforms as transforms
from torchvision import models
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
print(device)

num_epochs = 1
batch_size = 40
learning_rate = 0.001
classes = ('plane', 'car' , 'bird', 'cat', 'deer', 'dog', 'frog', 'horse', 'ship', 'truck')

transform = transforms.Compose([
    transforms.Resize(size=(224, 224)),
    transforms.ToTensor(),
    transforms.Normalize(
       (0.4914, 0.4822, 0.4465), (0.2023, 0.1994, 0.2010)
    )
])
train_dataset = torchvision.datasets.CIFAR10(root='./data', train=True, download=True, transform=transform)
test_dataset = torchvision.datasets.CIFAR10(root='./data', train=False, download=True, transform=transform)

train_loader = torch.utils.data.DataLoader(train_dataset, batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(test_dataset, batch_size=batch_size, shuffle=True)
n_total_step = len(train_loader)
print(n_total_step)

# Use GoogleNet (InceptionV1) instead of VGG16
model = models.googlenet(pretrained=True)
# Modify the last layer for CIFAR-10
model.fc = nn.Linear(model.fc.in_features, 10)
model = model.to(device)
criterion = nn.CrossEntropyLoss()
optimizer = torch.optim.SGD(model.parameters(), lr=learning_rate, momentum=0.9, weight_decay=5e-4)
```

21BAI1007-HOTS-GoogleNet.ipynb

```
for epoch in range(num_epochs):
    for i, (imgs, labels) in enumerate(train_loader):
        imgs, labels = imgs.to(device), labels.to(device)

        labels_hat = model(imgs)
        n_corrects = (labels_hat.argmax(axis=1) == labels).sum().item()
        loss_value = criterion(labels_hat, labels)

        loss_value.backward()
        optimizer.step()
        optimizer.zero_grad()

        if (i + 1) % 250 == 0:
            print(f'epoch {epoch + 1}/{num_epochs}, step: {i + 1}/{len(train_loader)}: loss = {loss_value:.5f}, acc = {100 * (n_corrects / labels.size(0)):.2f}%')

    print()
```

```
cuda
Downloading https://www.cs.toronto.edu/~kriz/cifar-10-python.tar.gz to ./data/cifar-10-python.tar.gz
100%|▓▓▓▓▓▓▓▓| 170498071/170498071 [00:03<00:00, 48804987.53it/s]
Extracting ./data/cifar-10-python.tar.gz to ./data
Files already downloaded and verified
1250
/usr/local/lib/python3.10/dist-packages/torchvision/models/_utils.py:208: UserWarning: 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: UserWarning: Arguments other than a weight enum or `None` for 'weights' are deprecated since 0.13 and may be removed in the future. The current behavior is equ
  warnings.warn(msg)
Downloading: "https://download.pytorch.org/models/googlenet-1378be20.pth" to /root/.cache/torch/hub/checkpoints/googlenet-1378be20.pth
100%|▓▓▓▓▓▓▓▓| 49.7M/49.7M [00:00<00:00, 170MB/s]
epoch 1/1, step: 250/1250: loss = 0.58048, acc = 85.00%
epoch 1/1, step: 500/1250: loss = 0.36317, acc = 90.00%
epoch 1/1, step: 750/1250: loss = 0.33483, acc = 90.00%
epoch 1/1, step: 1000/1250: loss = 0.47738, acc = 90.00%
epoch 1/1, step: 1250/1250: loss = 0.31067, acc = 90.00%
```

Comparsion of VGG16 and GoogleNet on CIFAR-10 dataset classification

Accuracy for VGG16: **89.64%**

Accuracy for GoogleNet: **91.06%**