拓展题

1.卷积神经网络(CNN)实践

难度

> ：中等

在解决了很多困难之后，恭喜你来到本阶段的最后一题，你已经很棒了！

如今，文字识别功能已经非常普及。早在1989年，来自 AT&T 的研究院 Yann LeCun 提出了一个非常有时代意义的模型：LeNet(LetNet-5)。这个模型被应用于识别手写字体并且大获成功。今天，让我们追随前人的足迹，将这个模型重现出来。

（你的电脑可以在没有独显的情况下运行这个程序）

知识准备

- 查阅和 CNN 有关的资料：吴恩达的网课和各种博客

https://blog.csdn.net/AI_dataloads/article/details/133250229

https://blog.csdn.net/lucklycoder/article/details/128606200

阅读文献后你需要掌握的概念

- 卷积的数学公式，平移不变性，局部性（了解即可）
- 互相关运算，卷积核，1*1卷积核，输入输出通道，池化层，汇聚层，卷积层（必须掌握）
- 卷积层的设计，卷积运算过程（也非常重要）
- LetNet-5模型的构造（重要性不言而喻）
- 填充，步幅（拓展）

进行实践

准备就绪，让我们快乐地开始吧！

这里仅要求你们补全关键代码并且理解整个实现流程。**需要注意的是，我们使用的 minist 数据集每个图片大小都是28*28像素，这与 LetNet 数据集是32*32像素有所不同。**主要地，我们有以下6个流程：

1. **定义 LeNet 模型**：一个简单的卷积神经网络。
2. **数据预处理**：定义了数据转换操作，包括将图像转换为张量和标准化。
3. **加载 MNIST 数据集**：下载并加载训练和测试数据集。
4. **设置训练参数**：初始化模型、损失函数和优化器。
5. **训练模型**：定义了一个训练函数，进行模型训练。
6. **测试模型**：定义了一个测试函数，评估模型在测试集上的表现。

你将主要完成**定义 LeNet 模型**部分，如果一切顺利，你可以在完成本题后将自己手写的数字交给模型识别！

本代码需要用到的库：

```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import datasets, transforms
from PIL import Image
```

定义 LeNet 模型：

```
class LeNet(nn.Module):
    def __init__(self):
        super(LeNet, self).__init__()
        #定义卷积核，全连接层
        #只需使用nn.Conv2d和nn.Linear实现
•
        def forward(self, x):
        #前向传播设计 使用torch库函数实现
```

数据预处理：

```
#问题1：:为什么要这么做
transform = transforms.Compose([
    transforms.ToTensor(),
    transforms.Normalize((0.1307,), (0.3081,))
])
```

加载 MNIST 数据集：

```
train_dataset = datasets.MNIST(root='./data', train=True, transform=transform,
download=True)
test_dataset = datasets.MNIST(root='./data', train=False, transform=transform,
download=True)
#问题2：说明Dataloader的功能
train_loader = DataLoader(train_dataset, batch_size=64, shuffle=True)
test_loader = DataLoader(test_dataset, batch_size=1000, shuffle=False)
```

训练准备：

```
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
model = LeNet().to(device)
criterion = nn.CrossEntropyLoss()
optimizer = optim.SGD(model.parameters(), lr=0.01, momentum=0.9)
```

训练模型：

```
def train(model, device, train_loader, optimizer, criterion, epoch):
    model.train()
    #问题3：train_loader内存放的数据是什么？格式是什么？
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
```

```
        output = model(data)
        loss = criterion(output, target)
        loss.backward()
        optimizer.step()
        if batch_idx % 100 == 0:
            print(
    f'Train Epoch: {epoch} [{batch_idx * len(data)}/{len(train_loader.dataset)}
     ({100. * batch_idx / len(train_loader):.0f}%)]\tLoss: {loss.item():.6f}')
```

测试模型与模型保存：

为了避免在没有改变模型参数时进行不必要的训练，这里提供模型的保存和模型调用功能

```
def test(model, device, test_loader, criterion):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():#问题4：为什么要no_grad
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += criterion(output, target).item()
            pred = output.argmax(dim=1, keepdim=True)
            correct += pred.eq(target.view_as(pred)).sum().item()
•
    test_loss /= len(test_loader.dataset)
    print(
        f'\nTest set: Average loss: {test_loss:.4f},
        Accuracy: {correct}/{len(test_loader.dataset)}
        ({100. * correct / len(test_loader.dataset):.0f}%)\n')
•
    for epoch in range(1, 5):
    train(model, device, train_loader, optimizer, criterion, epoch)
    test(model, device, test_loader, criterion)
•
# 保存模型
torch.save(model.state_dict(), 'lenet_model.pth')
print("Model saved to lenet_model.pth")
```

模型调用和手写数字测试：

```
def predict_image(image_path, model, device):
    model.eval()
    image = Image.open(image_path).convert('L')
    image = image.resize((28, 28))
    image = transform(image).unsqueeze(0).to(device)
•
    with torch.no_grad():
        output = model(image)
        prediction = output.argmax(dim=1)
    return prediction.item()
#调用模型
model = LeNet().to(device)
model.load_state_dict(torch.load('lenet_model.pth'))
model.eval()
print("Model loaded from lenet_model.pth")
```

```
image_path = 'path.png'  # 替换为你自己的图片路径
prediction = predict_image(image_path, model, device)
print(f'The predicted digit is: {prediction}')
```

需要提交部分：

1. 在已经给出代码部分有4个问题，请回答。
2. 使用 pytorch 提供的库函数，按照如下流程图中的模型按成“**定义 LeNet 模型”**并提交该函数：
3. 为什么最后的“FC:1*10”需要把矩阵变成1*10，变成10*1行不行？变成1*20呢？
4. 写一个小总结记录学习过程（切中要点即可）
5. （拓展）如果你学习了填充和步幅，请运用所学知识自行设计一个适用于本任务的模型，层数<=6。提交LetNet部分源代码和测试结果截图。

拓展

在第三题和第四题中，我们都自定义实现了深度学习框架中的常见组件，现在，让我们来尝试实现一下**卷积类**和**池化类**的**源代码。**

Conv2d

```
class Conv2d:
    def __init__(self, ...):
        ...
        self.weight = ...
        self.bias = ...
        ...

    def forward(self, ...):
        ...
```

对于卷积函数的实现，您需要考虑以下内容：

对于卷积函数的实现，您需要考虑以下内容：

1.Conv2d是针对几维input的?

2.Conv2d需要哪些输入?

3.weight和bias如何初始化?

4.forward方法如何实现?

MaxPool2d&&AvgPool2d

```
class MaxPool2d:
    def __init__(self, ...):
        ...
    def forward(self, ...):
        ...
class AvgPool2d:
    def __init__(self, ...):
        ...
    def forward(self, ...):
        ...
```

注意：对本拓展部分，您只需要完成上面三个类的源代码的编写即可，不需要使用上面三个类去搭建LeNet来完成任务，这不同于第四题的拓展。所以在提交时，针对本拓展部分来说，您只需要提交**源代码**即可。

无需提交部分

- 把整个代码看懂，机器学习中有很多格式固定的语句，请充分熟悉整个流程，面试有可能考到。

注意事项

1. 源代码（必要的注释和良好的规范）
2. 认真完成拓展内容

提交方式

将题目中要求的提交的总结内容利用 Markdowm 格式进行编辑，并存为 PDF 文件。将其与你的源代码一起提交至邮箱：gimmerml@163.com

文件名要求：姓名-学号-拓展题.pdf

出题人

皇家饼干（学长）

QQ: 3081962771

本题拓展：

Khalil（学长）

QQ: 2053296645