浙江大学

计算机视觉作业报告

(此模板仅适用于前三个简单的编程作业)

作业名称：	**Learning CNN**
姓　　名：	汪鑫
学　　号：	21921164
电子邮箱：	wangxin96_2015@163.com
联系电话：	18757584990
导　　师：	孙建伶

2019 年　12 月　27 日

作业名称

(撰写上简明扼要、开门见山，无需废话，文字不在于多)

一、 作业已实现的功能简述及运行简要说明

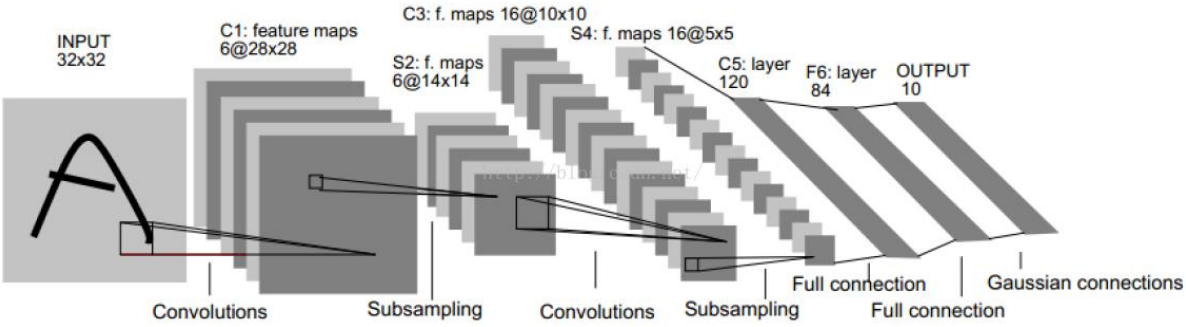
1. 按照 LeNet5 的结构，用 pytorch 实现了网络，并将其用于手写数字识别，在 mnist 数据集上准确率达到了 99%；
2. 用 pytorch 搭建了 CNN 网络进行物体分类，在数据集 cifar10 上准确率达到了 82%；

一、 作业的开发与运行环境

1. python3.6
2. pytoch1.3.1
3. cuda10

二、 系统或算法的基本思路、原理、流程或步骤等

1. LeNet5 结构



具体结构如下：

```
LeNet5(
  (conv1): Sequential(
    (0): Conv2d(1, 6, kernel_size=(5, 5), stride=(1, 1), padding=
(2, 2))
    (1): ReLU()
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (conv2): Sequential(
    (0): Conv2d(6, 16, kernel_size=(5, 5), stride=(1, 1))
    (1): ReLU()
```

```
    (2): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
  )
  (fc1): Sequential(
    (0): Linear(in_features=400, out_features=120, bias=True)
    (1): ReLU()
  )
  (fc2): Sequential(
    (0): Linear(in_features=120, out_features=84, bias=True)
    (1): ReLU()
  )
  (fc3): Linear(in_features=84, out_features=10, bias=True)
)
```

2. CNN 结构

该部分搭建了一个具有 6 个卷积层、2 全连接层的神经网络，具体结构如下：

```
CNN(
  (conv1): Sequential(
    (0): Conv2d(3, 32, kernel_size=(3, 3), stride=(1, 1), padding=
(3, 3))
    (1): ELU(alpha=1.0)
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
  )
  (conv2): Sequential(
    (0): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1))
    (1): ELU(alpha=1.0)
    (2): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (conv3): Sequential(
    (0): Conv2d(32, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1))
    (1): ELU(alpha=1.0)
    (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
  )
  (conv4): Sequential(
    (0): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=
(1, 1))
```

```
    (1): ELU(alpha=1.0)
    (2): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True, trac
k_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (conv5): Sequential(
    (0): Conv2d(64, 128, kernel_size=(3, 3), stride=(1, 1), padding
=(1, 1))
    (1): ELU(alpha=1.0)
    (2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, tra
ck_running_stats=True)
  )
  (conv6): Sequential(
    (0): Conv2d(128, 128, kernel_size=(3, 3), stride=(1, 1), paddin
g=(1, 1))
    (1): ELU(alpha=1.0)
    (2): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True, tra
ck_running_stats=True)
    (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1,
ceil_mode=False)
    (4): Dropout(p=0.25, inplace=False)
  )
  (fc1): Sequential(
    (0): Linear(in_features=2048, out_features=512, bias=True)
    (1): ELU(alpha=1.0)
  )
  (fc2): Linear(in_features=512, out_features=10, bias=True)
)
```

三、 具体如何实现，例如关键（伪）代码、主要用到函数与算法等

1. LeNet5 实现手写数字识别

（1） 读取数据集

torchvision 模块中自带一些常用的数据集，如：MNIST、cifar10、ImageNet 等，通过相应的接口即可实现下载和调用。本部分数据处理代码如下：

```
batch_size = 64
transform = transforms.ToTensor()

trainset = tv.datasets.MNIST(root = './data/', train = True, download = Fals
e, transform = transform)
trainloader = torch.utils.data.DataLoader(
```

```
    trainset,
    batch_size = batch_size,
    shuffle = True
)
testset = tv.datasets.MNIST(root = './data/', train = False, download = Fals
e, transform = transform)
testloader = torch.utils.data.DataLoader(
    testset,
    batch_size = batch_size,
    shuffle = True
)
```

其中，batch_size 设置为 64，MNIST()函数实现读取本地或者从网络下载 mnist 数据集并转换到相应的格式，DataLoader()函数实现读出 1 个 epoch 的数据

（2） 搭建 LeNet5 模型

模型以类的方式实现，继承了 nn 模块中的 Module，使用 torch.nn.Sequential()函数将各个层拼接在一起，forward()方法按照对应层计算相应的结果即可实现。代码如下：

```
class LeNet5(nn.Module):
    def __init__(self):
        super(LeNet5, self).__init__()
        self.conv1 = nn.Sequential(
            nn.Conv2d(1, 6, 5, padding = 2),
            # 尺寸 5*5，共 6 个，边缘补充 2 个，输出 6 个 28 * 28 的特征图
            nn.ReLU(),
            nn.MaxPool2d(kernel_size = 2, stride = 2)
        )
        self.conv2 = nn.Sequential(
            nn.Conv2d(6, 16, 5),
            # 16 个 6*5*5 卷积核，输出 16 * 10* 10 的特征图
            nn.ReLU(),
            nn.MaxPool2d(2,2)
        )
        self.fc1 = nn.Sequential(
            nn.Linear(16 * 5 * 5, 120),
            nn.ReLU()
        )
        self.fc2 = nn.Sequential(
            nn.Linear(120, 84),
            nn.ReLU()
        )
```

```
        self.fc3 = nn.Linear(84, 10)
    def forward(self, x):
        x = self.conv1(x)
        x = self.conv2(x)
        x = x.view(x.size()[0], -1)
        x = self.fc1(x)
        x = self.fc2(x)
        x = self.fc3(x)
        return x
```

（3） 训练和测试模型

网络训练采用了 Adam 算法，学习率设置为 0.001，其余参数采用默认值，梯度计算采用了 pytorch 中的自动微分模块。训练中，每 100 个 batch 计算一次平均损失，每个 epoch 结束后进行一次测试。具体代码略过。

2. cifar10 物体识别

（1） 数据处理

这部分的数据读取方法同 LeNet5，为了增强模型的泛化能力，采用了数据增强方式，具体代码如下：

```
img_aug = tfs.Compose([
        #tfs.Resize(120),
        tfs.RandomHorizontalFlip(),
        tfs.RandomCrop(28),
        tfs.ColorJitter(brightness=0.5, contrast=0.5, hue=0.5),
        tfs.ToTensor(),
        tfs.Normalize([0.5, 0.5, 0.5], [0.5, 0.5, 0.5])
    ])
```

（2） 搭建 CNN 模型

代码与 LeNet5 的相似，卷积层交替使用（conv，ELU，BN，maxpooling，dropout）和（conv，ELU，BN），其中 BN 和 dropout 是为了抑制过拟合。

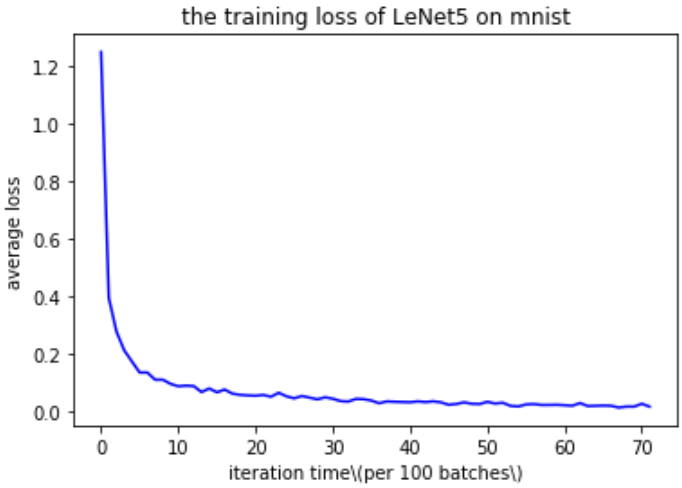
（3） 模型训练与测试

同 LeNet5

四、 实验结果与分析

1. LeNet5

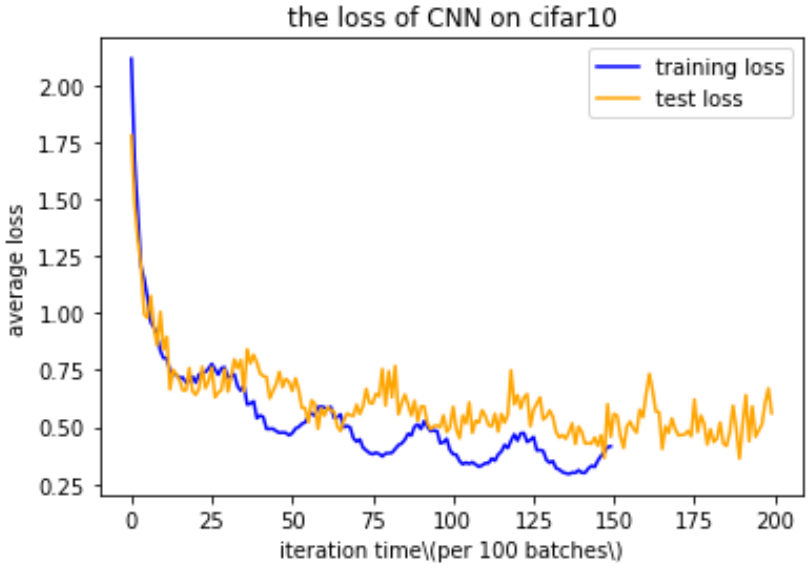
训练误差变化：



最终测试精度：99%

2. CNN 物体识别

训练和测试误差变化：



测试精度：batch_size = 128, epochs = 50，测试精度在 80%到 83%

五、 结论与心得体会

第一个任务基本没有什么难度，很快也达到了一个较好的结果，经过第一个任务，我对 pytorch 的基本使用没有什么问题。

在第二个任务中，我尝试自己设计 CNN 并调优，在实践过程中，总结了以下几点

经验：

1. 在 CNN 中，卷积层对结果的影响比全连接层要更大。在最初的设计中，我选用了 2 个卷积层和 5 个全连接层，在训练中，训练损失能够降到 0.2 左右，但是测试损失只能降到 0.8，这表明网络出现了过拟合，即使在全连接层中加入 dropout，也没有获得更好的结果；
2. batch_size 对训练也有一定的影响。过小或者过大都会降低收敛速度和计算速度；
3. 采用变学习率训练方案，如余弦退火、指数衰减等。

六、 参考文献