《神经网络与深度学习》

- 基础实验

曹兵

实验内容

- 基于卷积神经网络的图像识别算法实现与改进
  - ✓实验环境
  - ✓任务介绍
  - ✓作业提交

实现环境

◘ Pytorch（建议）环境安装

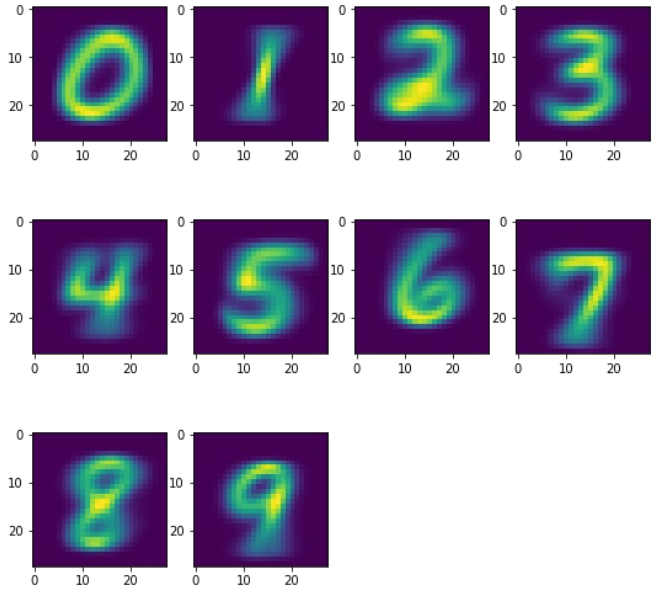
1. Anaconda下载及安装，下载地址 https://www.anaconda.com/products/individual
2. 创建PyTorch环境(在终端下使用命令行安装，Linux: Terminal Mac: iTerm Windows: Xshell)
   i) conda create –n env_name python=3.6 ## 创建虚拟环境，env_name为环境名称.
   ii) conda activate env_name ## 激活新建的环境
   iii) conda install pytorch torchvision ## 安装pytorch 和 torchvision
3. Anaconda的使用
   i) 进入程序所在目录下；
   ii) 激活PyTorch环境；
   iii) 执行训练文件

任务介绍

□ 数据集介绍

MNIST数据集是一个手写体数据集，其主要由四部分组成分别是训练数据的图像和标签以及测试数据的图像和标签。其中训练集包含60000个样本，测试集包含10000个样本。总共包含数字0-9十个类别，每张图像的分辨率为28×28。

数字图像平均像素分布

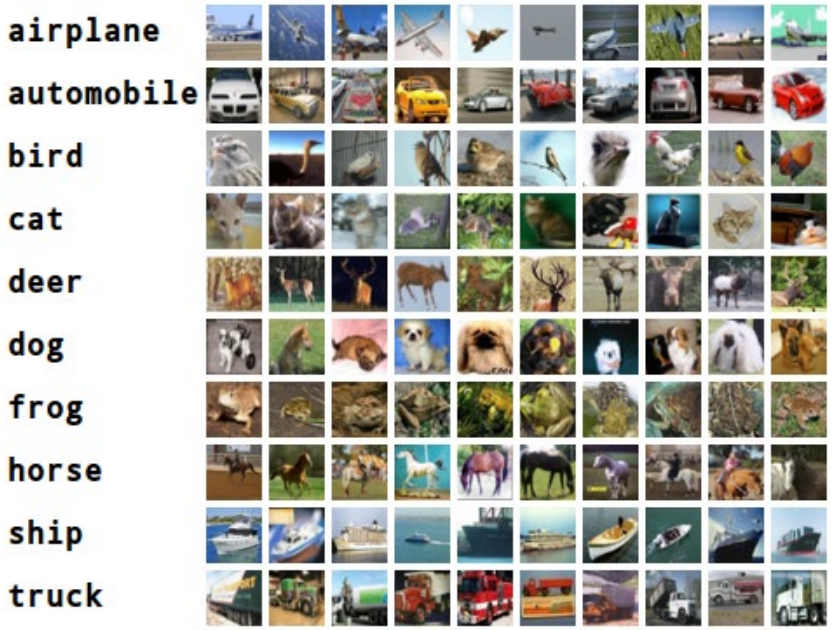


数据集图像实例

任务介绍

□ 数据集介绍

Cifar-10**数据集**由60000张32X32的RGB彩色图片构成，共10个分类。50000张训练，10000张测试。每张图像的分辨率为28×28。



数据库下载网址：

http://www.cs.toronto.edu/~kriz/cifar.html

任务介绍

数据加载

```
train_loader = torch.utils.data.DataLoader(
        datasets.MNIST('data', train=True, download=True,
                       transform=transforms.Compose([
                           transforms.ToTensor(),
                           transforms.Normalize((0.1307,), (0.3081,))
                       ])),
        batch_size=BATCH_SIZE, shuffle=True)


test_loader = torch.utils.data.DataLoader(
        datasets.MNIST('data', train=False, transform=transforms.Compose([
                           transforms.ToTensor(),
                           transforms.Normalize((0.1307,), (0.3081,))
                       ])),
        batch_size=BATCH_SIZE, shuffle=True)
```

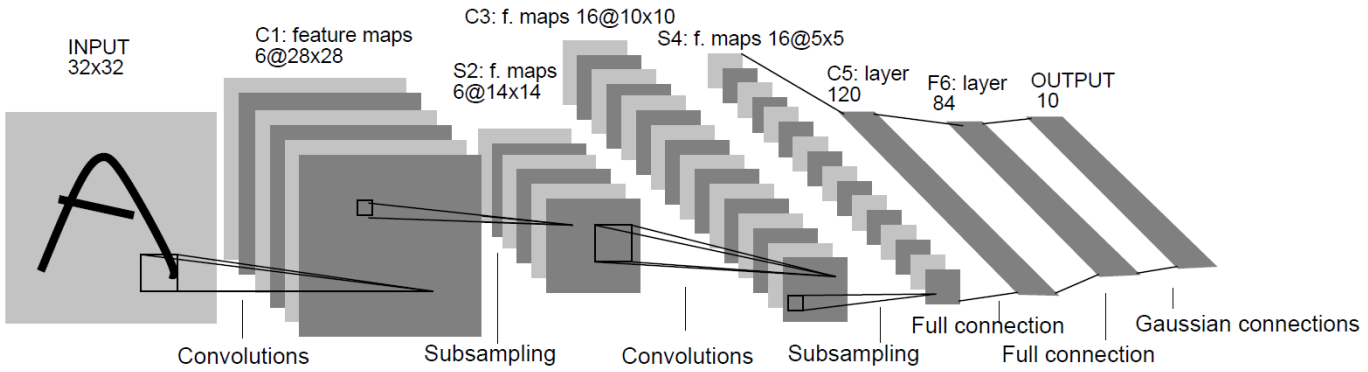
I. 定义数据增强的相关变换transform，并将读取的数据转换维pytorch的Tensor类型；

II. 根据传入的transform对数据进行相应的数据增强，并生成dataset，支持依赖于index读取数据；

III. 传入dataset生成dataloader，对数据集进行划分，切分为小批次（batch），支持后续的训练任务。

任务介绍

模型定义



LeNet-5

```python
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 120, 5)
        self.fc1 = nn.Linear(120,84)
        self.fc2 = nn.Linear(84, 10)
    def forward(self,x):
        in_size = x.size(0)
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out)  #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)
        out = out.view(in_size,-1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out,dim=1)
        return out
```

Lécun Y, Bottou L, Bengio Y, et al. Gradient-based learning applied to document recognition[J]. Proceedings of the IEEE, 1998, 86(11):2278-2324.

任务介绍

模型训练和测试

```python
def train(model, device, train_loader, optimizer, epoch):
    model.train()
    for batch_idx, (data, target) in enumerate(train_loader):
        data, target = data.to(device), target.to(device)
        optimizer.zero_grad()
        output = model(data)
        loss = F.nll_loss(output, target)
        loss.backward()
        optimizer.step()
        if(batch_idx+1)%30 == 0:
            print('Train Epoch: {} [{}/{} ({:.0f}%)]\tLoss: {:.6f}'.format(
                epoch, batch_idx * len(data), len(train_loader.dataset),
                100. * batch_idx / len(train_loader), loss.item()))
```

模型训练

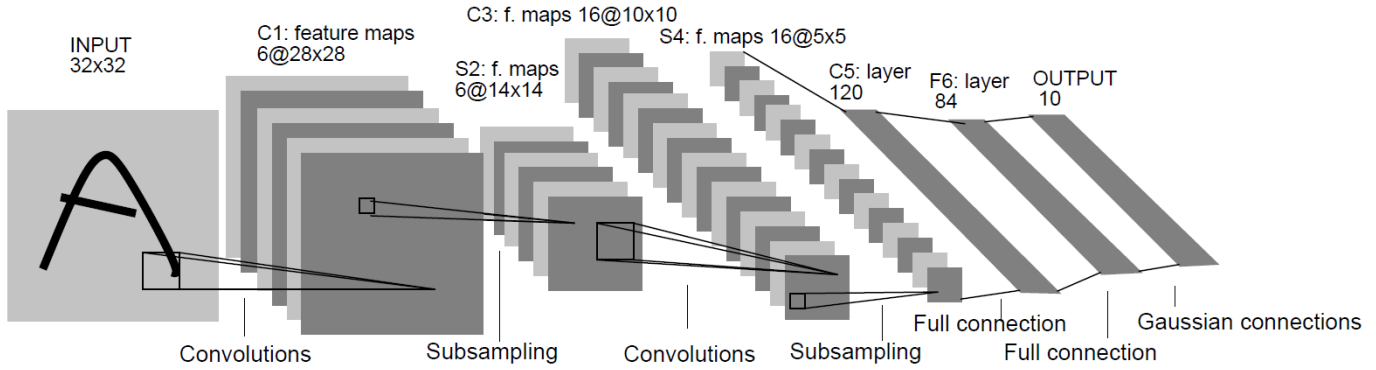
```python
def test(model, device, test_loader):
    model.eval()
    test_loss = 0
    correct = 0
    with torch.no_grad():
        for data, target in test_loader:
            data, target = data.to(device), target.to(device)
            output = model(data)
            test_loss += F.nll_loss(output, target, reduction='sum').item() # 将一批的损失相加
            pred = output.max(1, keepdim=True)[1] # 找到概率最大的下标
            correct += pred.eq(target.view_as(pred)).sum().item()

    test_loss /= len(test_loader.dataset)
    print('\nTest set: Average loss: {:.4f}, Accuracy: {}/{} ({:.0f}%)\n'.format(
        test_loss, correct, len(test_loader.dataset),
        100. * correct / len(test_loader.dataset)))
```

模型测试

任务介绍

任务一：LeNet-5的模型架构改进

```python
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 120, 5)
        self.fc1 = nn.Linear(120,84)
        self.fc2 = nn.Linear(84, 10)
    def forward(self,x):
        in_size = x.size(0)
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out)  #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)
        out = out.view(in_size,-1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out,dim=1)
        return out
```



I. 根据自身硬件资源对LeNet-5(/其他网络)的层数/滤波器大小/滤波器数量等方面改进；

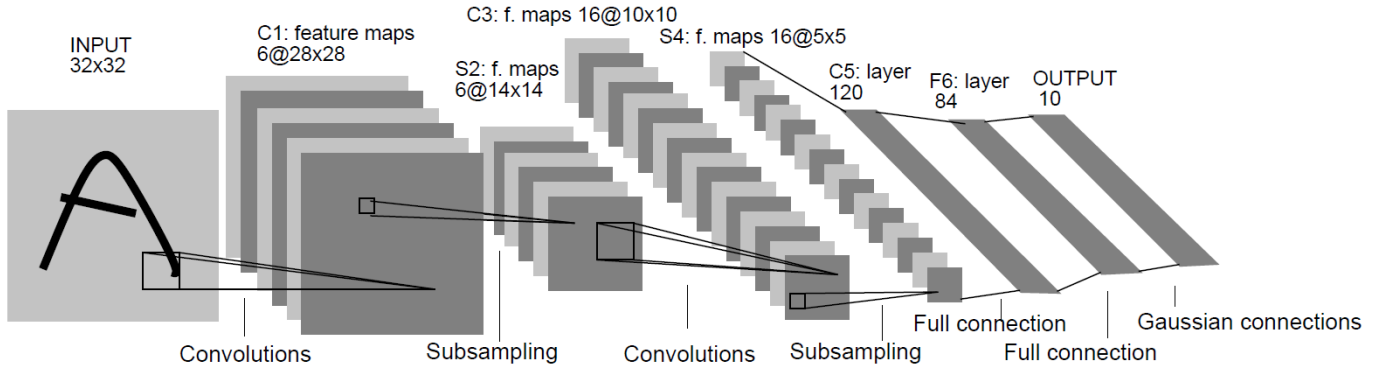
II. 提出至少一种改进模型；

III. 在MNIST/Cifar-10数据集与LeNet-5进行比较验证。

任务介绍

任务二：LeNet-5的模型激活函数比较分析

```python
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 120, 5)
        self.fc1 = nn.Linear(120,84)
        self.fc2 = nn.Linear(84, 10)
    def forward(self,x):
        in_size = x.size(0)
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out)  #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)
        out = out.view(in_size,-1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out,dim=1)
        return out
```



I. 对LeNet-5 (/其他网络)的激活函数进行比较分析或改进；

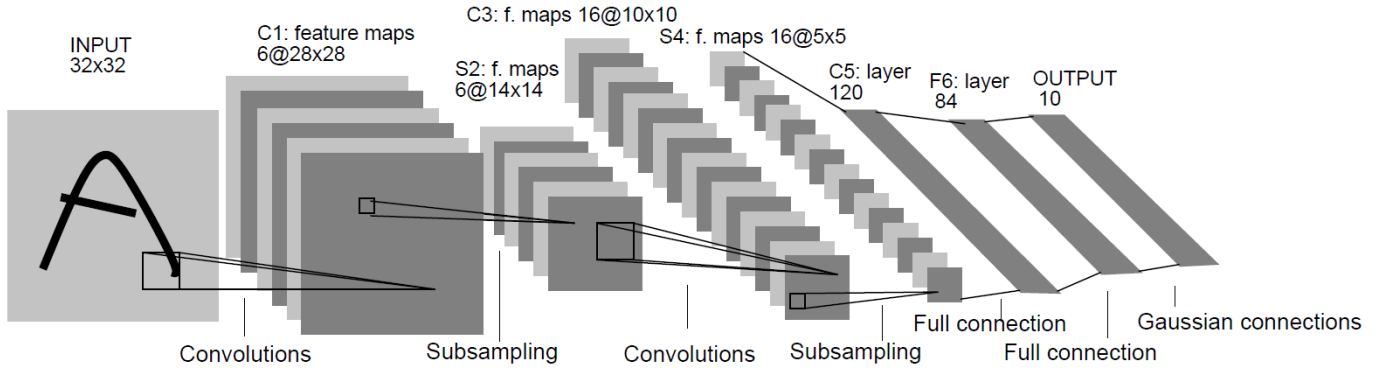
II. 比较至少三种的激活函数（例如ReLU , GELU, Tanh, ELU等）；

III. 在MNIST/Cifar-10数据集分析比较不同激活函数的收敛特性。

任务介绍

- 任务三：LeNet-5的模型归一化方法的比较分析

```python
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 120, 5)
        self.fc1 = nn.Linear(120,84)
        self.fc2 = nn.Linear(84, 10)
    def forward(self,x):
        in_size = x.size(0)
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out)  #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)
        out = out.view(in_size,-1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out,dim=1)
        return out
```



I. 对LeNet-5 (/其他网络)的归一化方法进行比较分析或改进；

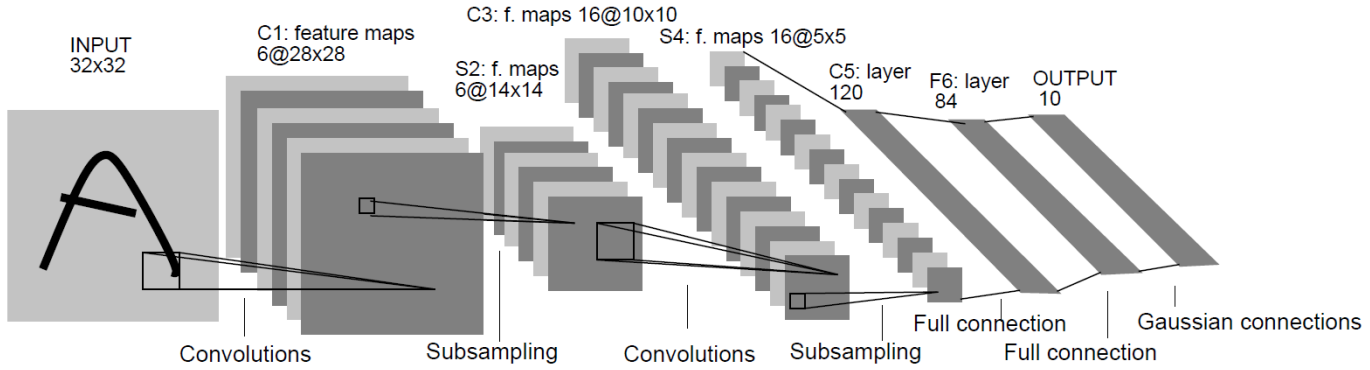
II. 比较至少三种的归一化方法（例如无归一化，BN, LN, GN等）；

III. 在MNIST/Cifar-10数据集分析比较不同归一化方法在不同Batch-size下的性能。

任务介绍

- 任务四：LeNet-5的模型优化方法的比较分析

```python
class ConvNet(nn.Module):
    def __init__(self):
        super().__init__()
        self.conv1 = nn.Conv2d(1, 6, 5, padding=2) # 28x28
        self.pool1 = nn.MaxPool2d(2, 2) # 14x14
        self.conv2 = nn.Conv2d(6, 16, 5) # 10x10
        self.pool2 = nn.MaxPool2d(2, 2) # 5x5
        self.conv3 = nn.Conv2d(16, 120, 5)
        self.fc1 = nn.Linear(120,84)
        self.fc2 = nn.Linear(84, 10)
    def forward(self,x):
        in_size = x.size(0)
        out = self.conv1(x) #24
        out = F.relu(out)
        out = self.pool1(out)  #12
        out = self.conv2(out) #10
        out = F.relu(out)
        out = self.pool2(out)
        out = self.conv3(out)
        out = out.view(in_size,-1)
        out = self.fc1(out)
        out = F.relu(out)
        out = self.fc2(out)
        out = F.log_softmax(out,dim=1)
        return out
```



I. 对LeNet-5 (/其他网络)模型优化方法进行比较分析或改进；

II. 比较至少三种的优化方法（例如动量SGD, AdaDelta, Adam等）；

III. 在MNIST/Cifar-10数据集分析比较不同优化方法的收敛行为和泛化性。

作业提交

- 提供LeNet-5（鼓励自己创造新的网络结构）+ MNIST基础代码和数据
- 以上四个任务任选一个即可
- 智慧树提交实验报告与代码
- 截止日期：12月31号晚上12点