认知科学与类脑计算课程实验报告

实验题目： HMAX 模型的实现		学号：201600181073
日期：2019.5.30	班级： 智能 16	姓名： 唐超

实验目的：

根据 HMAX 模型的相关知识，在 MNIST 数据集上实现一个用于图像分类的 HMAX 模型。

实验环境：

Jupyter notebook & python3.7 & pytorch

主要算法：

HMAX 模型是一个层次式的结构，分为 5 层：S1 层、C1 层、S2 层、C2 层、VTU 层。

S1 层。该层事实上是用 4 个方向及 16 个尺度的 Gabor 滤波器组对输入图像进行滤波，得到 64 个响应图。在尺度上，HMAX 算法分的相当细致，它将 16 个尺度分成 8 个子带，每个子带包含两个相邻尺度，以便在之后的 C1 层进行整合。

C1 层。如前所述，64 个响应图依子带编号分成 8 组，每组包括 S1 层得到的 2 个相邻尺度及所属每个尺度上所有的 4 个方向响应。C1 层的操作是依组依方向进行的。具体做法是先将任一个滤波器响应图划分成 8*8 的格子，在每个格子中求响应的最大值。

S2 层。在该层中，再次使用滤波器对 C1 层的输出进行滤波，产生新一轮的响应图。只是这次滤波器变成了之前随机抽取的那些，而非第一层硬性规定的 Gabor 滤波器。滤波的方式也有所不同，不再是卷积运算，而是直接以 L2 距离作比较，再用高斯核映射成相似度。注意到随机抽取的 patch 是 n*n*4 的，因此做完滤波后，对每一个 patch 我们能得到 8 个组各一张图的响应。假设 patch 数目是 K，则共有 8*K 张响应作为 S2 的输出。

C2 层。对每一个 patch 遍历 8 个组的响应，找到最大的那个值。如此，对每张输入图片，最后得到一个 K 维向量。

VTU 层。该层可以使用 SVM 及其他方法，将 C2 层输出的 K 维向量放入该层进行训练。

程序设计&调试分析：

（1）下载 MNIST 数据集。

直接用 torchvision 中的 torchvision.datasets 类进行下载和 transform。

```
transform=transforms.Compose([
    transforms.Grayscale(),
    transforms.ToTensor(),
    transforms.Lambda(lambda x: x * 255),
])

trainset = tv.datasets.MNIST(
    root='./data/',
    train=True,
    download=True,
    transform=transform)
```

（2）构建 HMAX 模型。

按照上一栏中的实验原理，分别定义 S1 层、C1 层、S2 层、C2 层，最后使用 SVM 作为 VTU 层。

（3）使用 MNIST 数据集中的训练集训练网络，使用测试集测试训练好的网络。

训练 10 个 epoch，用交叉熵作为损失函数，Adam 作为优化器，训练部分代码如下：

```
for epoch in range(num_epoches):
    for data in trainloader:
        inputs, labels = data
        c2_opt = model.forward(inputs)
        c2_opt = c2_opt.view(c2_opt.size(0), -1)

        c2_opt = Variable(c2_opt)
        labels = Variable(labels)

        c2_opt = c2_opt.to(device)
        labels = labels.to(device)

        out = model2(c2_opt)
        loss = criterion(out, labels)
        print_loss = loss.data.item()

        optimizer.zero_grad()
        loss.backward()
        optimizer.step()
        print('epoch: {}, loss: {:.4}'.format(epoch, loss.data.item()))
```

测试结果及分析：

训练 10 个 epoch 后，模型在测试集上的准确率为 88.6%.