姓名：陈聪

学号：202021090323

日期：2020.9.20

**作业1**

一、概述（简要叙述目的，设计思路等）

MNIST手写数字数据集来源于是美国国家标准与技术研究所，是著名的公开数据集之一，通常这个数据集都会被作为深度学习的入门案例。数据集中的数字图片是由250个不同职业的人纯手写绘制。

本实验旨在通过深度学习模型，识别MNIST手写数字数据集，目标为达到一个较高的精确度。实验设计了两种不同的模型，一种基于**DNN**的线性分类模型，另一种基于**CNN**的卷积神经网络模型。实验基于开源的pytorch深度学习框架。实验代码主要包括四个部分。**数据处理**，**模型设计搭建**，**模型训练推理**与**实验结果分析**。

数据处理：使用torchvision库内置的MNIST数据集方法获取数据并处理。

模型设计：基于pytorch搭建的DNN模型和CNN的模型。

模型的训练：使用SGD优化方法和交叉熵作为损失函数进行训练。

实验结果分析：分析错误的原因，包括热力图展示等。

二、数据集（数据集的来源，部分数据样本截图）

数据集来源于torchvision库的内置方法提供的下载链接[[1]](#footnote-1)，包括四个部分，训练图片集、测试图片集、训练标签、测试标签。图片样例如下：



图1 mnist数据集样例

三、模型设计（详细阐述模型设计的思路，最好配以公式和图）

模型由堆叠的块构成，每个块内部结构相似。对于线性模型，每个块由线性层、dropout层、Relu激活函数层构成。对于卷积模型，由二维卷积、批归一化层、激活层、dropout层构成。如图2和图3所示。两个模型均由5个块构成。

|  |  |
| --- | --- |
| 图2 线性模型结构 | 图3 卷积模型结构 |

两个模型具体的每层参数设置如表1和表2所示：

表1 线性模型参数设置

|  |  |  |  |
| --- | --- | --- | --- |
| Layer | input size | Output size | Dropout rate |
| 1 Linear | 28\*28 | 1024 | 0.2 |
| 2 Linear | 1024 | 2048 | 0.2 |
| 3 Linear | 2048 | 1024 | 0.2 |
| 4 Linear | 1024 | 512 | 0.2 |
| 5 Linear | 512 | 10 | 0.2 |

表2 卷积模型参数设置

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Layer | in channels | Kernel | Stride | Out channels | Dropout |
| 1Conv | 1 | 3\*3 | 1 | 64 | 0.2 |
| 2Conv | 64 | 3\*3 | 2 | 256 | 0.2 |
| 3Conv | 256 | 3\*3 | 2 | 512 | 0.2 |
| 4Conv | 512 | 3\*3 | 2 | 1024 | 0.2 |
| 5Conv | 1024 | 3\*3 | 2 | 256 | 0.2 |
| FC | 256\*2\*2 | - | - | 10 | - |

四、核心代码（关键代码及解释）

|  |
| --- |
| Model.py  import torch  import torch.nn as nn  class LinerModel(nn.Module):  def \_\_init\_\_(self,iamge\_w:int,image\_h:int):  """  模型输入为(N,1,28,28),线性模型期望输入为(N,L)  因此首先需要将输入转化，这里模型仅为线性的组合  """  super().\_\_init\_\_()  self.drop\_rate = 0.2  self.liner = nn.Sequential(  nn.Linear(iamge\_w\*image\_h,1024),  nn.Dropout(p=self.drop\_rate),  nn.PReLU(),  nn.Linear(1024,2048),  nn.Dropout(p=self.drop\_rate),  nn.ReLU(),  nn.Linear(2048,1024),  nn.Dropout(p=self.drop\_rate),  nn.ReLU(),  nn.Linear(1024,512),  nn.Dropout(p=self.drop\_rate),  nn.ReLU(),  nn.Linear(512,10),  nn.Dropout(p=self.drop\_rate),  nn.ReLU(),  )  def forward(self, x): #(N,1,28,28)  trans\_x = torch.squeeze(x,1).view([x.size(0),x.size(2)\*x.size(3)])  out = self.liner(trans\_x) #(N,10)  # out = nn.Softmax(dim=-1)(out)  return out  class CnnModel(nn.Module):  def \_\_init\_\_(self):  super().\_\_init\_\_()  self.drop\_rate = 0.2  self.cnn = nn.Sequential(  nn.Conv2d(1,64,3,stride=1,padding=1),  nn.BatchNorm2d(64),  nn.PReLU(),  nn.Dropout(p=self.drop\_rate),  nn.Conv2d(64,256,3,stride=2,padding=1),  nn.BatchNorm2d(256),  nn.ReLU(),  nn.Dropout(p=self.drop\_rate),  nn.Conv2d(256,512,3,stride=2,padding=1),  nn.BatchNorm2d(512),  nn.ReLU(),  nn.Dropout(p=self.drop\_rate),  nn.Conv2d(512,1024,3,stride=2,padding=1),  nn.BatchNorm2d(1024),  nn.ReLU(),  nn.Dropout(p=self.drop\_rate),  nn.Conv2d(1024,256,3,stride=2,padding=1),  nn.BatchNorm2d(256),  nn.ReLU(),  nn.Dropout(p=self.drop\_rate),  )  self.linear = nn.Linear(256\*2\*2,10)  def forward(self,x):  out = self.cnn(x)  out = out.view(out.size(0),out.size(1)\*out.size(2)\*out.size(3))  out = self.linear(out)  return out |

完整代码见GitHub，地址 <https://github.com/kouyt5/postgraduate/tree/master/cource-neural-network>

五、结果（准确率及识别结果）

线性模型准确率：**98.5%**

卷积模型准确率：**99.3%**

对测试集的分类错误图片进行分析，发现主要容易将4识别成9，7识别成2和1，5识别成3。数据中也存在一定量的**标注错误**。如下表所示：

表3 识别结果

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 预测错误数字 |  |  |  |  |  |  |  |
| 真实值 | 4 | 7 | 5 | 7 | 0 | 4 | 3 |
| 预测值 | 9 | 2 | 6 | 1 | 6 | 9 | 5 |

对于人类识别其中的某些数字来说，也难以辨认，因为这些数字具有一定的**迷惑性**。这也是导致模型难以完全预测正确的原因之一。

图4和图5是热力图，横坐标为正确的标签，纵坐标为预测的标签，数值为对应的错误分类的数量。可以发现出错较多的主要为**长相相似**的数字。

还可以发现的是，由于数字的相似性，因此热力图近似对称分布。

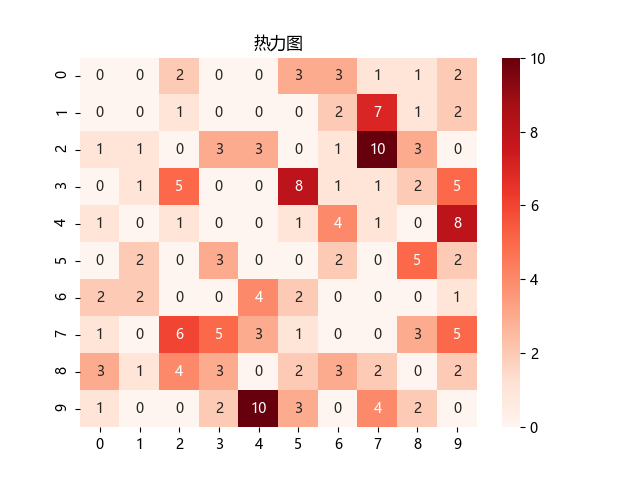


图4 错误统计热力图98.5%准确率

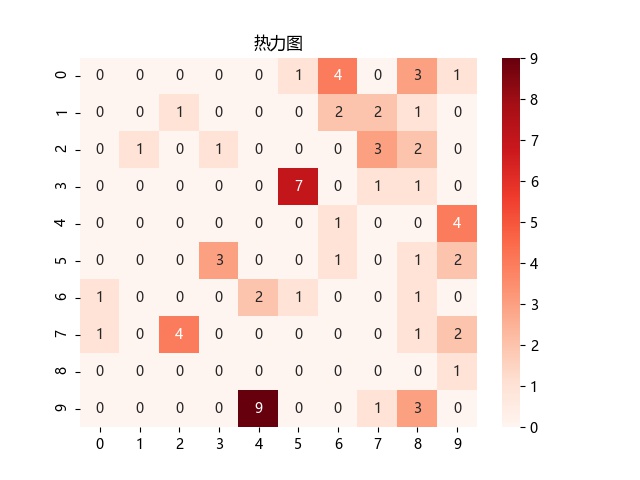


图5 错误统计热力图99.3%准确率

1. http://yann.lecun.com/exdb/mnist/train-images-idx3-ubyte.gz

   http://yann.lecun.com/exdb/mnist/train-labels-idx1-ubyte.gz

   http://yann.lecun.com/exdb/mnist/t10k-images-idx3-ubyte.gz

   http://yann.lecun.com/exdb/mnist/t10k-labels-idx1-ubyte.gz [↑](#footnote-ref-1)