



中国科学院大学

University of Chinese Academy of Sciences

课程作业报告

基于卷积神经网络的手写数字识别

作者姓名: 段宏键

学科专业: 计算机系统结构

所在单位: 中国科学院大学计算机科学与技术学院

2020 年 6 月

1. 摘要

本项目是为了实现对手写数字的识别。本项目设计并实现了一个轻量级的 CNN 网络架构（卷积，ReLU，最大池化，卷积，ReLU，最大池化，全连接层），使用 MNIST 标准数据集，经过短短的 5 轮迭代就达到了 99% 的识别准确率。

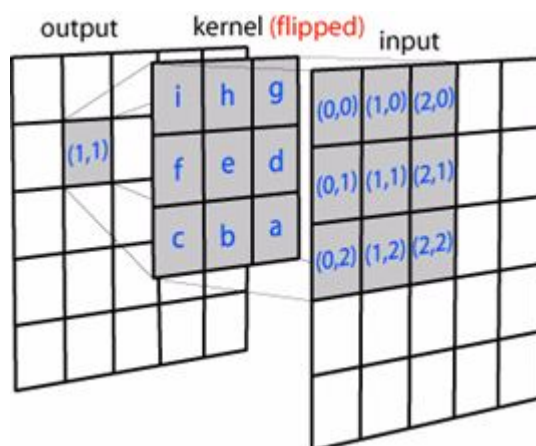
代码设计采用训练（train.py）与测试（test.py）分离的模式，训练代码会保存模型的参数，测试代码加载参数能在本地快速完成测试工作。

1. 介绍

(1) 图像天然就是规格非常整齐的数据，所以简单的处理图像对计算机而言是比较容易的。但对图像的识别却并不简单。图像识别是指利用计算机对图像进行处理、分析和理解，以识别各种不同模式的目标和对像的技术。图像识别的发展经历了三个阶段：文字识别、数字图像处理与识别、物体识别。机器学习领域一般将此类识别问题转化为分类问题。

手写数字识别是常见的图像识别任务。计算机通过手写体图片来识别出图片中的字，与印刷字体不同，手写数字风格大小不一，这会造成计算机对手写识别任务的困难。手写数字识别由于其有限的类别（0~9 共 10 个数字）成为了相对简单识别任务。MNIST 是常用的手写数字识别数据集。

(2) 针对手写数字大小不一的特点，可以采用 BatchNormlize 来对数据进行预处理；针对数字的风格不同，可以采用卷积层来提取图像的特征。对于最后的分类（分为 0-9 这 10 类），可以使用全连接层来根据卷积提取出来的特征进行分类。

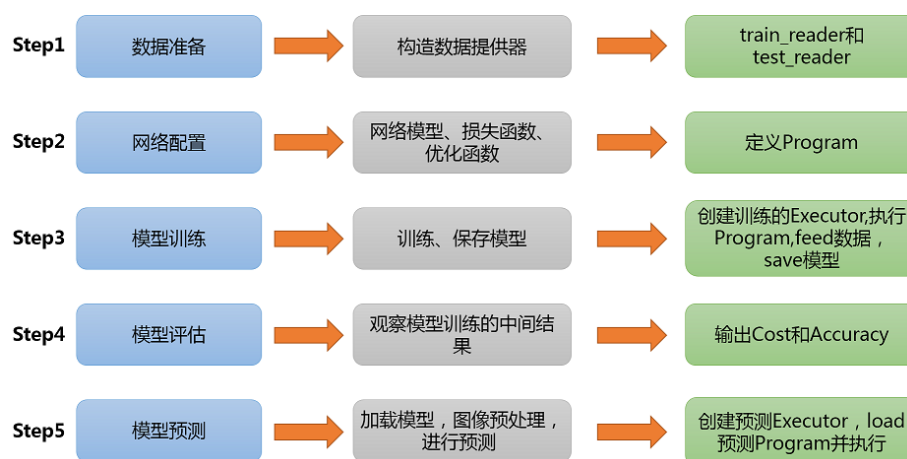


(3) 我采用类似 LeNet-5 的结构来构造我网络,使用第一个卷积层来提取图片中细粒度的特征,使用第二个卷积层来把细粒度特征汇总成比较大的特征,同时每次卷积之后均使用 ReLU 和最大池化操作来减少特征维度,突出重要特征。最后使用全连接层来根据这些特征进行分类。其中训练程序(train.py)会保存训练模型,测试程序(test.py)可以直接加载模型参数来进行准确率验证,最后的准确率能达到 98.99%。

2. 解决方案

实验代码采用了模块化设计的方法,并且把模型训练与测试分开,这样能解耦代码结构,同时也能在把训练模型得到的模型数据保存,以便测试代码能在弱计算环境下顺利测试,而训练代码则可以使用云平台或其他平台。同时模型的训练与测试均兼容了纯 CPU 平台与 CPU+GPU (cuda) 平台。

整个实验过程遵循以下几个步骤:



数据使用使用 pytorch 中自带的 MNIST 数据集并加载,对训练数据以及测试数据分别在训练代码和测试代码中加载就可。设置了数据 batch_size 为 100。

网络设计采用了 卷积+ReLU+MaxPooling+卷积+ReLU+MaxPooling+线性全连接层,详细的网络结构设计如下:

```
CNN(
  (conv1): Sequential(
    (0): Conv2d(1, 16, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
    (1): BatchNorm2d(16, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
```

```

        (2): ReLU()
        (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (conv2): Sequential(
      (0): Conv2d(16, 32, kernel_size=(5, 5), stride=(1, 1), padding=(2, 2))
      (1): BatchNorm2d(32, eps=1e-05, momentum=0.1, affine=True, track_running_stats=True)
      (2): ReLU()
      (3): MaxPool2d(kernel_size=2, stride=2, padding=0, dilation=1, ceil_mode=False)
    )
    (fc): Linear(in_features=1568, out_features=10, bias=True)
  )

```

网络的搭建使用了 pytorch 的序列工具来快速构建。

训练优化方法与损失函数如下：

```

#4、 选择损失函数和优化方法
loss_func = nn.CrossEntropyLoss()
optimizer = torch.optim.Adam(cnn.parameters(), lr=learning_rate)

```

损失函数的设计采用交叉熵损失函数，优化方法是 Adam。

设置了学习率为 0.001，对与这个简单的分类实验，学习率采用固定模式是完全可以的，能够取得好的效果。

具体 train 的设计可以参见代码。

测试部分的设计与训练部分很相似，只是加载的数据集变为了测试数据集，同时去掉了误差反向传播的过程。

3. 实验结果和分析

实验配置如下：

深度学习平台：anaconda + pytorch（具体环境包依赖可以参见实验目录下的“环境依赖.txt”）

数据集：MNIST

训练环境：google colabatory(训练使用的 runtime 是 GPU 类型)

测试环境：自己的笔记本电脑，无 GPU

训练过程中会定时输出 loss 值，并在最后保存训练模型的参数以便用于测

试。

最后在训练 5 轮保存下来的参数下的测试结果如下：

```
processed [0/10000] images, now the correctness is 1.0000
processed [1000/10000] images, now the correctness is 0.9855
processed [2000/10000] images, now the correctness is 0.9876
processed [3000/10000] images, now the correctness is 0.9897
processed [4000/10000] images, now the correctness is 0.9885
processed [5000/10000] images, now the correctness is 0.9880
processed [6000/10000] images, now the correctness is 0.9884
processed [7000/10000] images, now the correctness is 0.9885
processed [8000/10000] images, now the correctness is 0.9884
processed [9000/10000] images, now the correctness is 0.9891
end...
the final correctness in test dataset is 0.9899
```

除此之外，我观察到在训练过程中的 loss 如下，其中 Epoch[1/5], Iter[100/600] 表明是训练的第一轮，训练的 batch 中的第 100 个（其中 batch_size 是 100，即已经训练了 $100 \times 100 = 10000$ 张图片了）

其中 loss 会上升，但是总体是下降的。

```
Epoch [1/5], Iter [100/600] Loss: 0.0804
Epoch [1/5], Iter [200/600] Loss: 0.0917
Epoch [1/5], Iter [300/600] Loss: 0.0566
Epoch [1/5], Iter [400/600] Loss: 0.0825
Epoch [1/5], Iter [500/600] Loss: 0.1271
Epoch [1/5], Iter [600/600] Loss: 0.0960
Epoch [2/5], Iter [100/600] Loss: 0.0311
Epoch [2/5], Iter [200/600] Loss: 0.0213
Epoch [2/5], Iter [300/600] Loss: 0.0132
Epoch [2/5], Iter [400/600] Loss: 0.0555
Epoch [2/5], Iter [500/600] Loss: 0.0304
Epoch [2/5], Iter [600/600] Loss: 0.0543
Epoch [3/5], Iter [100/600] Loss: 0.0355
Epoch [3/5], Iter [200/600] Loss: 0.0585
Epoch [3/5], Iter [300/600] Loss: 0.0124
Epoch [3/5], Iter [400/600] Loss: 0.0461
Epoch [3/5], Iter [500/600] Loss: 0.0527
Epoch [3/5], Iter [600/600] Loss: 0.0161
Epoch [4/5], Iter [100/600] Loss: 0.0399
Epoch [4/5], Iter [200/600] Loss: 0.0124
Epoch [4/5], Iter [300/600] Loss: 0.0470
Epoch [4/5], Iter [400/600] Loss: 0.0110
Epoch [4/5], Iter [500/600] Loss: 0.1061
```

```
Epoch [4/5], Iter [600/600] Loss: 0.0762
Epoch [5/5], Iter [100/600] Loss: 0.0299
Epoch [5/5], Iter [200/600] Loss: 0.0234
Epoch [5/5], Iter [300/600] Loss: 0.0088
Epoch [5/5], Iter [400/600] Loss: 0.0175
Epoch [5/5], Iter [500/600] Loss: 0.0849
Epoch [5/5], Iter [600/600] Loss: 0.0214
```

4. 结论

这个实验我使用两层卷积的卷积神经网络对在数字集 MNIST 上进行了训练和测试。依次按照数据准备、网络构建、模型训练、模型评估以及模型测试这一系列步骤来完成这个任务。这是本课程中我完成的第一个课程项目，代码按照训练与测试分离解耦的方式组织，既方便了线上的训练，也方便了线下的测试。同时通过本实验我理解了卷积以及池化这写层的作用，并且认识到简单的两层卷积网络就能在 MNIST 数据集上取得非常好的效果（99%）。

5. 参考文献

- [1] 课程资源《实验一 手写数字识别实验指导书》
- [2] Belongie, Serge, Jitendra Malik, and Jan Puzicha. "Shape matching and object recognition using shape contexts." IEEE transactions on pattern analysis and machine intelligence 24.4 (2002): 509-522.