华中科技大学计算机科学与技术学院

《计算机视觉》实验一报告

专　　业：计算机科学与技术

班　　级：CS1903

学　　号：U201914376

姓　　名：沈承磊

指导教师：李贤芝

完成日期： 2022 年 5 月 25 日

目录

一、实验题目：基于卷积神经网络的 MNIST 手写数字体识别

二、实验要求

2.1 实验任务

设计一个卷积神经网络，在 MNIST 数据集上实现 10 分类手写数字体识别。

2.2 注意事项

a) 不能直接导入现有的 CNN 网络，比如 VGG、ResNet 等，可以以现有网络为基础进行改进。
b) 可以尝试不同的卷积神经网络设计、使用不同的激活函数等，观察网络性能。
c) 深度学习框架任选。

三、实验环境与平台

实验环境：Anaconda 虚拟环境
实验框架：Pytorch 版本 torch1.11.0+cu113
实验平台：Pycharm Professional
实验设备：AMD Ryzen 7 5800H with Radeon Graphics
GPU 型号：RTX3060

四、算法设计

4.1 数据处理

a) **数据下载：**利用 pytorch 的图形库 torchvision 下载 mnist 数据集，也可以在 huggingface 手动下载。
b) **数据预处理：**由于数据集中每张图片均为 28*28 个大小在[0,255]之间的像素点，因此对每个像素点可采用除以 255 的方式实现归一化。另外避免模型学到输入的顺序信息，每轮训练对训练集使用 shuffle。
c) **注意事项：**由于本次实验将计算部分放置在 GPU 上进行，所以对数据所在设备也要进行处理。其中输入样本要以 FloatTensor 的形式放置在 cuda 上，

便于调用 GPU 的算力；标签以 numpy 的形式放置在 cpu 上，便于后期使用 numpy 的一些函数简化 F1 和正确率的计算。

4.2 模型算法

本次实验重点在于网络搭建部分。在实验过程中我基于 CNN 的方式尝试搭建了各种不同形式的网络。总结如下：

a) 模型构建：

较为基础的（卷积-激活-池化）*2-（全连接）*2 形式：在此基础上尝试增添了不同层的深度，并尝试缩小了卷积核的大小，得到不同的效果。

原理解析：

1. 卷积层的作用是进行特征提取，池化层的作用是进行特征选择，降低特征数量从而减少参数数量。另外池化层还能实现平移不变性、旋转不变性、尺度不变性，从而提高模型的泛华能力。全连接层作用是将提取到的部分特征进行拼接，组合出完整的特征。
2. 卷积神经网络整体趋势是使用更小的卷积核以减少模型参数量，同时采用更深的网络结构以增强模型的拟合能力，因此一种策略是使用多层较小卷积核代替大卷积核。

b) 模型构建：

在上述网络添加了 dropout 正则化方法以解决过拟合问题，在此基础上测试了不同保留比例下模型的效果变化。

原理解析：

Dropout 解决过拟合的原理有如下两个方面：

1. 取平均。每次进行 Dropout 就会随机舍弃掉一些神经元，剩下的神经元相当于组成一个新的网络，训练过程会产生很多这样的网络。假设相同数据训练 5 个不同神经网络分别输出 5 个不同结果，相当于采取 5 个结果取均值，或者多数取胜的投票策略，有利于让一些过拟合的网络相互抵消。
2. 减少神经元之间复杂的共适应关系。Dropout 导致两个神经元不一定同时起作用，权值更新不再依赖有固定关系的隐含结点的共同作用，增加鲁棒性。

c) 模型构建：

基于 resnet 的思想搭建了十层隐藏层的残差网络，同样采用了 dropout 的方法防止过拟合。

原理解析：

卷积神经网络并不是直观上越深效果越好，随着网络加深，网络的退化现象严重，这是因为多层的非线性转换将数据映射到更高纬的空间以便于更好的完成“数据分类”。随着网络深度的不断增大，所引入的激活函数也越来越多，数据被映射到更加离散的空间，此时已经难以让数据回到原点（恒等变换），也就失去了数据最初的特征。因此，可以为深层的网络层增添一条直接接通浅层网络层的通路，从而保留数据更为基础的特征，此即残差网络。

d) 模型构建：

在上述网络基础上，改变部分网络层的激活函数，从而增加神经元的多

样性进而提高了模型的拟合能力。

原理解析：

直观上，提高神经元的多样性，能提高模型的拟合能力。采用不同激活函数后能让网络以更多样的方式拟合函数，从而达到效果提升。

五、参数设定及实验结果

a) 采用较小的两层卷积核（3*3）代替原来的一层核（5*5）。使用输出矩阵计算公式可以得出让输出矩阵和输入矩阵尺寸不变的几种卷积核参数配置，例如：表 5-1 所示：

Kernel_size=7*7	stride=1	padding=3
Kernel_size=5*5	stride=1	padding=2
Kernel_size=3*3	stride=1	padding=1

表 5-1 常用输入输出尺寸相等的卷积核参数配置

算法 a)配置如下：

- Learning_Rate 设为 1e-4。
- Max_epoch 设为 20。
- 训练集设定为经 shuffle 后的全部训练样本。
- 测试集设定为全部测试样本的前 500 个切片。
- 输出为 0-9 各个标签下的 F1-Score 和总体的正确率。
- 网络结构设置如图 5-1 所示：

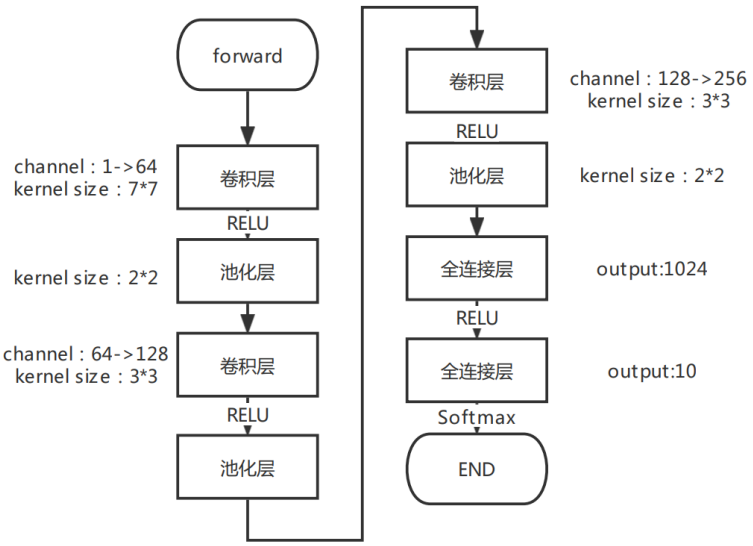

图 5-1 算法 a)对应网络结构模型

- 结果记录如图 5-2 所示：

```
======= epoch: 0 ===== test F1 is [0.944 0.981 0.973 0.892 0.927 0.913 0.940 0.889 0.850 0.916] ===== test sum accuracy is 0.913
======= epoch: 1 ===== test F1 is [0.935 0.985 0.938 0.936 0.938 0.980 0.943 0.929 0.900 0.900] ===== test sum accuracy is 0.952
======= epoch: 2 =====test F1 is [0.956 0.985 0.973 0.928 0.964 0.959 0.965 0.959 0.925 0.962] ===== test sum accuracy is 0.956
======= epoch: 3 ===== test F1 is [0.965 0.914 0.953 0.967 0.972 0.970 0.953 0.960 0.937 0.962] ===== test sum accuracy is 0.966
======= epoch: 4 ===== test F1 is [0.946 0.953 0.981 0.956 0.943 0.980 0.932 0.961 0.940 0.944] ===== test sum accuracy is 0.978
======= epoch: 5 ===== test F1 is [0.952 0.985 0.982 0.964 0.972 0.960 0.953 0.951 0.938 0.963] ===== test sum accuracy is 0.976
======= epoch: 6 =====test F1 is [0.935 0.985 0.964 0.945 0.972 0.950 0.965 0.961 0.892 0.964] ===== test sum accuracy is 0.972
======= epoch: 7 ===== test F1 is [0.965 0.913 0.982 0.967 0.964 0.980 0.953 0.984 0.914 0.954] ===== test sum accuracy is 0.976
======= epoch: 8 ===== test F1 is [0.926 0.985 0.943 0.978 0.972 0.990 0.965 0.940 0.949 0.972] ===== test sum accuracy is 0.976
======= epoch: 9 =====test F1 is [0.946 0.985 0.977 0.968 0.953 0.970 0.953 0.981 0.952 0.953] ===== test sum accuracy is 0.984
======= epoch: 10 =====test F1 is [0.977 0.993 0.982 0.967 0.943 0.961 0.965 0.935 0.920 0.906] ===== test sum accuracy is 0.972
======= epoch: 11 =====test F1 is [0.955 0.985 0.954 0.943 0.982 0.990 0.965 0.980 0.949 0.972] ===== test sum accuracy is 0.986
======= epoch: 12 ===== test F1 is [0.955 0.993 0.982 0.967 0.982 0.970 0.965 0.970 0.950 0.972] ===== test sum accuracy is 0.98
======= epoch: 13 ===== test F1 is [0.966 0.993 0.972 0.968 0.979 0.970 0.977 0.990 0.963 0.972] ===== test sum accuracy is 0.986
======= epoch: 14 =====test F1 is [0.966 0.993 0.991 0.989 0.982 0.936 0.977 0.991 0.917 0.952] ===== test sum accuracy is 0.984
======= epoch: 15 ===== test F1 is [0.966 0.985 0.972 0.945 0.982 0.970 0.965 0.980 0.941 0.962] ===== test sum accuracy is 0.98
======= epoch: 16 =====test F1 is [0.966 0.993 0.982 0.968 0.982 0.970 0.965 0.990 0.950 0.981] ===== test sum accuracy is 0.984
======= epoch: 17 ===== test F1 is [0.955 0.984 0.972 0.967 0.991 0.961 0.965 0.990 0.950 0.981] ===== test sum accuracy is 0.984
======= epoch: 18 ===== test F1 is [0.978 0.993 0.973 0.966 0.953 0.980 0.977 0.990 0.909 0.973] ===== test sum accuracy is 0.982
======= epoch: 19 =====test F1 is [0.958 0.983 0.976 0.974 0.976 0.990 0.965 0.970 0.936 0.982] ===== test sum accuracy is 0.982
```

图 5-2 算法 a)对应实验结果

b) 引入 Dropout 机制，将其应用在第一个全连接层上。

- Dropout 参数为节点选择丢弃的概率，设定为 0.5。
- 其余参数配置同上。
- 网络结构设置如图 5-3 所示：

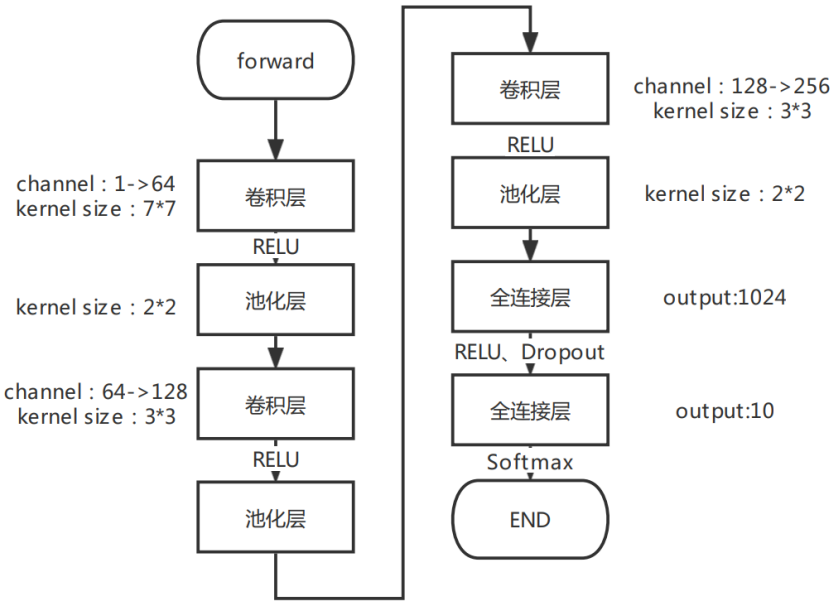

图 5-3 算法 b)对应网络模型

- 实验结果记录如图 5-4：

```
======== epoch: 0 ===== test F1 is [ 0.944  0.993  0.933  0.899  0.957  0.929  0.940  0.889  0.850  0.916] ===== test sum accuracy is 0.928
======== epoch: 1 ===== test F1 is [ 0.955  0.985  0.938  0.936  0.938  0.980  0.943  0.929  0.900  0.900] ===== test sum accuracy is 0.952
======== epoch: 2 =====test F1 is [ 0.966  0.985  0.973  0.928  0.964  0.959  0.965  0.959  0.925  0.962] ===== test sum accuracy is 0.97
======== epoch: 3 ===== test F1 is [ 0.955  0.993  0.973  0.957  0.982  0.970  0.953  0.960  0.937  0.962] ===== test sum accuracy is 0.976
======== epoch: 4 ===== test F1 is [ 0.966  0.993  0.982  0.966  0.943  0.980  0.932  0.961  0.940  0.944] ===== test sum accuracy is 0.972
======== epoch: 5 ===== test F1 is [ 0.955  0.985  0.982  0.944  0.972  0.960  0.953  0.951  0.938  0.963] ===== test sum accuracy is 0.972
======== epoch: 6 =====test F1 is [ 0.955  0.985  0.964  0.946  0.972  0.950  0.965  0.961  0.892  0.964] ===== test sum accuracy is 0.968
======== epoch: 7 ===== test F1 is [ 0.955  0.993  0.982  0.967  0.964  0.980  0.953  0.980  0.914  0.954] ===== test sum accuracy is 0.976
======== epoch: 8 ===== test F1 is [ 0.966  0.985  0.973  0.978  0.982  0.990  0.965  0.950  0.949  0.972] ===== test sum accuracy is 0.982
======== epoch: 9 =====test F1 is [ 0.966  0.985  0.973  0.968  0.953  0.970  0.953  0.980  0.952  0.953] ===== test sum accuracy is 0.976
======== epoch: 10 =====test F1 is [ 0.977  0.993  0.982  0.967  0.943  0.971  0.965  0.990  0.920  0.906] ===== test sum accuracy is 0.972
======== epoch: 11 =====test F1 is [ 0.955  0.985  0.991  0.978  0.982  0.990  0.965  0.980  0.949  0.972] ===== test sum accuracy is 0.986
======== epoch: 12 ===== test F1 is [ 0.955  0.993  0.982  0.967  0.982  0.970  0.965  0.970  0.950  0.972] ===== test sum accuracy is 0.982
======== epoch: 13 ===== test F1 is [ 0.966  0.993  0.982  0.968  0.972  0.970  0.977  0.990  0.963  0.972] ===== test sum accuracy is 0.986
======== epoch: 14 =====test F1 is [ 0.966  0.993  0.991  0.989  0.982  0.990  0.977  0.990  0.918  0.952] ===== test sum accuracy is 0.986
======== epoch: 15 ===== test F1 is [ 0.966  0.985  0.972  0.945  0.982  0.970  0.965  0.980  0.941  0.962] ===== test sum accuracy is 0.978
======== epoch: 16 =====test F1 is [ 0.966  0.993  0.982  0.968  0.982  0.970  0.965  0.990  0.950  0.981] ===== test sum accuracy is 0.986
======== epoch: 17 ===== test F1 is [ 0.955  0.985  0.982  0.967  0.991  0.961  0.965  0.990  0.950  0.981] ===== test sum accuracy is 0.984
======== epoch: 18 ===== test F1 is [ 0.977  0.993  0.973  0.966  0.953  0.980  0.977  0.990  0.909  0.953] ===== test sum accuracy is 0.978
======== epoch: 19 =====test F1 is [ 0.955  0.993  0.973  0.978  0.973  0.990  0.965  0.980  0.937  0.981] ===== test sum accuracy is 0.984
```

图 5-4 算法 b)对应实验结果

c) 引入 ResNet 的残差思想，重新搭建网络。

- Dropout 参数调整为 0.2。
- 其余参数配置同上。
- 网络结构设置如图 5-5 所示：

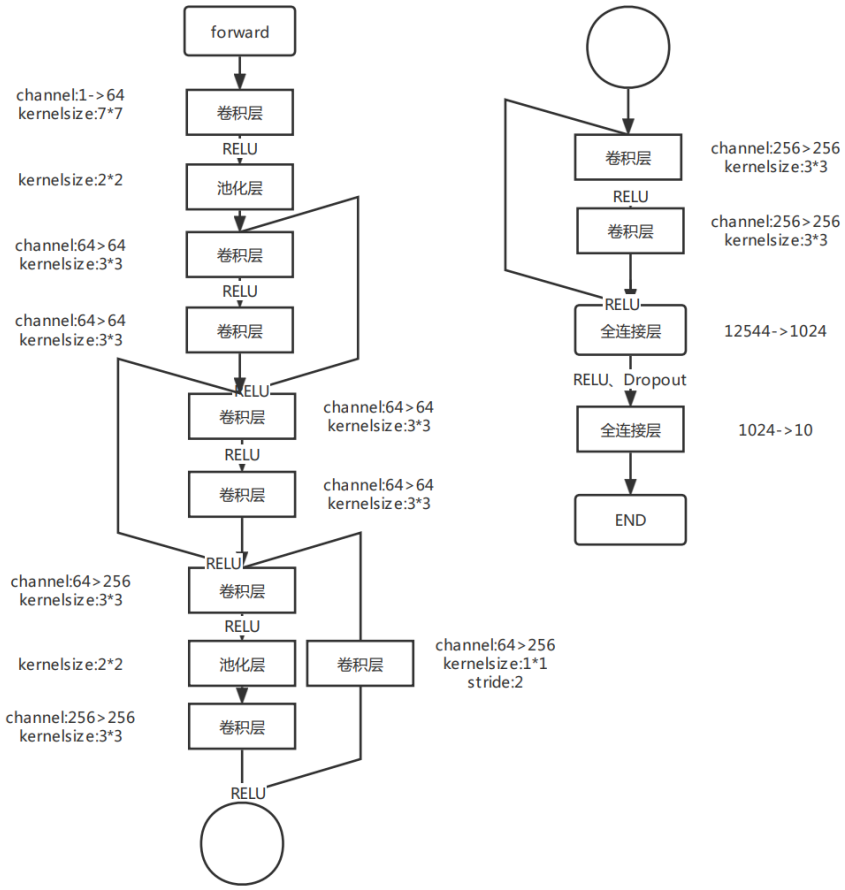

图 5-5 算法 c)对应网络模型

- 实验结果记录如图 5-6 所示：

```
======= epoch: 0 ===== test F1 is [ 0.966  0.978  0.982  0.957  0.926  0.970  0.933  0.970  0.923  0.964] ===== test sum accuracy is 0.968
======= epoch: 1 ===== test F1 is [ 0.977  0.993  0.973  0.968  0.973  0.970  0.989  0.980  0.950  0.981] ===== test sum accuracy is 0.986
======= epoch: 2 ===== test F1 is [ 0.955  0.993  0.982  0.968  0.982  0.980  0.953  0.970  0.950  0.972] ===== test sum accuracy is 0.982
======= epoch: 3 ===== test F1 is [ 0.988  0.993  0.982  0.957  0.982  0.970  0.977  0.970  0.941  0.952] ===== test sum accuracy is 0.982
======= epoch: 4 ===== test F1 is [ 0.977  0.985  0.982  0.978  0.982  0.980  0.966  0.980  0.988  0.981] ===== test sum accuracy is 0.99
======= epoch: 5 ===== test F1 is [ 0.988  0.993  0.982  0.978  0.991  0.980  0.989  0.980  0.964  0.972] ===== test sum accuracy is 0.992
======= epoch: 6 ===== test F1 is [ 0.988  0.993  0.991  0.968  0.982  0.959  0.966  0.990  0.988  0.991] ===== test sum accuracy is 0.992
======= epoch: 7 ===== test F1 is [ 0.988  0.993  0.982  0.968  0.972  0.980  0.989  0.990  0.975  0.982] ===== test sum accuracy is 0.992
======= epoch: 8 ===== test F1 is [ 0.966  0.993  0.991  0.956  0.982  0.962  0.952  0.990  0.975  0.982] ===== test sum accuracy is 0.986
======= epoch: 9 ===== test F1 is [ 0.966  0.993  0.982  0.978  0.973  0.990  0.953  0.990  0.963  0.981] ===== test sum accuracy is 0.988
======= epoch: 10 ===== test F1 is [ 0.988  0.985  0.982  0.957  0.991  0.970  0.989  0.990  0.964  0.981] ===== test sum accuracy is 0.99
======= epoch: 11 ===== test F1 is [ 0.966  0.993  0.991  0.978  0.982  0.980  0.953  0.990  0.988  0.991] ===== test sum accuracy is 0.992
======= epoch: 12 ===== test F1 is [ 0.977  0.993  0.982  0.978  0.982  0.980  0.977  0.990  0.988  0.991] ===== test sum accuracy is 0.994
======= epoch: 13 ===== test F1 is [ 0.988  0.993  0.991  0.978  0.991  0.980  0.989  0.990  0.988  0.991] ===== test sum accuracy is 0.998
======= epoch: 14 ===== test F1 is [ 0.977  0.985  0.982  0.978  0.982  0.980  0.966  0.990  0.988  0.991] ===== test sum accuracy is 0.992
======= epoch: 15 ===== test F1 is [ 0.977  0.993  0.991  0.957  0.982  0.950  0.953  0.990  0.988  0.991] ===== test sum accuracy is 0.988
======= epoch: 16 ===== test F1 is [ 0.988  0.993  0.991  0.967  0.982  0.970  0.977  0.990  0.988  0.991] ===== test sum accuracy is 0.994
======= epoch: 17 ===== test F1 is [ 0.988  0.993  0.991  0.978  0.982  0.980  0.977  0.990  0.975  0.982] ===== test sum accuracy is 0.994
======= epoch: 18 ===== test F1 is [ 0.988  0.993  0.982  0.989  0.991  0.990  0.989  0.980  0.988  0.991] ===== test sum accuracy is 0.998
======= epoch: 19 ===== test F1 is [ 0.977  0.993  0.972  0.978  0.991  0.980  0.966  0.980  0.988  0.991] ===== test sum accuracy is 0.992
```

图 5-6 算法 c)对应实验结果

d) 在算法 c)的基础上改变部分激活函数为 Sigmoid。

- 其余参数配置同上。
- 网络结构设置如图 5-7 所示：

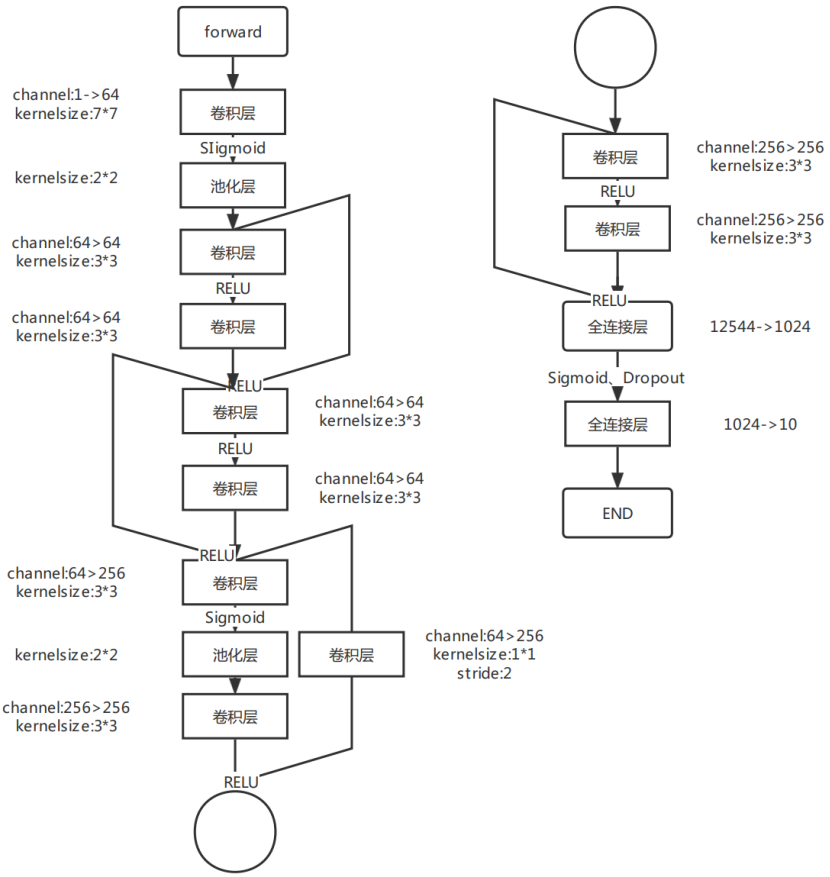

图 5-7 算法 d)对应网络模型

实验结果记录如图 5-8 所示：

```
======= epoch: 0 ===== test F1 is [ 0.966  0.993  0.973  0.957  0.964  0.970  0.977  0.959  0.916  0.953] ===== test sum accuracy is 0.974
======= epoch: 1 ===== test F1 is [ 0.955  0.993  0.973  0.978  0.964  0.980  0.965  0.961  0.949  0.972] ===== test sum accuracy is 0.98
======= epoch: 2 ===== test F1 is [ 0.977  0.985  0.973  0.967  0.982  0.961  0.989  0.980  0.975  0.972] ===== test sum accuracy is 0.986
======= epoch: 3 ===== test F1 is [ 0.955  0.993  0.973  0.978  0.972  0.980  0.965  0.980  0.975  0.982] ===== test sum accuracy is 0.986
======= epoch: 4 ===== test F1 is [ 0.977  0.993  0.973  0.978  0.972  0.980  0.989  0.980  0.975  0.982] ===== test sum accuracy is 0.99
======= epoch: 5 ===== test F1 is [ 0.977  0.993  0.991  0.978  0.973  0.980  0.977  0.990  0.975  0.981] ===== test sum accuracy is 0.992
======= epoch: 6 ===== test F1 is [ 0.988  0.993  0.991  0.978  0.963  0.970  0.977  0.990  0.988  0.964] ===== test sum accuracy is 0.99
======= epoch: 7 ===== test F1 is [ 0.988  0.985  0.972  0.978  0.982  0.980  0.977  0.970  0.988  0.981] ===== test sum accuracy is 0.99
======= epoch: 8 ===== test F1 is [ 0.977  0.993  0.982  0.968  0.982  0.980  0.966  0.970  0.988  0.972] ===== test sum accuracy is 0.988
======= epoch: 9 ===== test F1 is [ 0.966  0.993  0.991  0.978  0.972  0.980  0.953  0.990  0.988  0.982] ===== test sum accuracy is 0.99
======= epoch: 10 ===== test F1 is [ 0.977  0.993  0.982  0.978  0.982  0.980  0.966  0.980  0.988  0.991] ===== test sum accuracy is 0.992
======= epoch: 11 ===== test F1 is [ 0.988  0.993  0.982  0.967  0.982  0.980  0.977  0.990  0.963  0.991] ===== test sum accuracy is 0.992
======= epoch: 12 ===== test F1 is [ 0.988  0.993  0.982  0.968  0.982  0.970  0.989  0.990  0.988  0.991] ===== test sum accuracy is 0.994
======= epoch: 13 ===== test F1 is [ 0.988  0.993  0.991  0.978  0.972  0.980  0.977  0.990  0.988  0.982] ===== test sum accuracy is 0.994
======= epoch: 14 ===== test F1 is [ 0.988  0.993  0.982  0.978  0.982  0.980  0.966  0.980  0.988  0.981] ===== test sum accuracy is 0.992
======= epoch: 15 ===== test F1 is [ 0.988  0.993  0.991  0.967  0.982  0.980  0.977  0.990  0.976  0.991] ===== test sum accuracy is 0.994
======= epoch: 16 ===== test F1 is [ 0.988  0.993  0.991  0.978  0.973  0.980  0.977  0.990  0.988  0.981] ===== test sum accuracy is 0.994
======= epoch: 17 ===== test F1 is [ 0.977  0.993  0.973  0.968  0.982  0.970  0.977  0.980  0.988  0.991] ===== test sum accuracy is 0.99
======= epoch: 18 ===== test F1 is [ 0.988  0.993  0.982  0.978  0.972  0.970  0.977  0.990  0.975  0.972] ===== test sum accuracy is 0.99
======= epoch: 19 ===== test F1 is [ 0.977  0.993  0.982  0.978  0.982  0.980  0.977  0.990  0.988  0.991] ===== test sum accuracy is 0.994
```

图 5-8 算法 d)对应实验结果

六、结果分析

1) 算法 b)引入 dropout 机制后，在多轮次训练下相较于算法 a)正确率得到一定程度上的增加，这可以解释为 dropout 机制能有效防止多轮训练过程中过拟合现象发生。
2) 在算法 c)中引入了残差学习，可以看到训练 10 轮之后模型正确率基本稳定在 0.99 以上，尤其在第 14 和第 19 轮训练后更是达到了 0.998 的准确率（已接近此任务的 SOTA 准确率）。
3) 在算法 d)中引入了混合激活函数的方式，直观理解神经元的多样性能有效提高模型拟合能力，但从结果来看并没有很大的变化，推测原因应该是算法 c)中 RELU 函数的弊端并没有彰显出来，即“死亡 RELU”现象并没有发生，因此引入其他激活函数对该算法并没有明显提高。

七、实验心得

此次实验让我对课堂上老师讲的理论知识有了进一步的理解，对于卷积核大小，通道数这些概念有了图形化的认知，解决了课堂上留存的疑惑。对课堂上提及的 ResNet 残差网络也进行了实操，收获颇丰。由于调整为残差网络后网络层数进一步加深，参数量也增大了不少，我尝试将代码改成 GPU 版本，其中也遇到一些问题譬如 numpy 不支持 GPU 版本，我通过将训练数据和网络放在 GPU 上运算，将标签等数据的后期处理放在 CPU 上运算解决了此问题。代码能力的提高总是日积月累的，本次实验的几次碰壁让我学到不少新知识。

另外，我对 torch 的运用也更加熟练了，借助此次实验我较为系统地阅读了 torch 官方文档，相比于之前百度搜索各种函数的做法显然要更加高效和权威。由于电脑中配有其他课程的环境为了不产生关联影响，我采用 conda 的虚拟环境有效地安装了本实验对应版本的 pytorch 框架以及要用到的对应版本的库，conda 虚拟环境之间是相互独立的，不会相互影响，这有助于深度学习人员将精力集中在算法本身而不是环境配置上，很有帮助，在此强烈推荐。

最后很感谢贤芝老师对我们的悉心指导，我在群聊中提的每一个问题老师都

会耐心解答，给人很亲切的感觉，完全感受不到距离感。贤芝老师的 CV 课也是为数不多我愿意通过课堂进行学习，并实实在在能在课堂学到东西的课，为老师的认真负责点赞！（下次实验一定去机房！）