实验要求

 , 查看  版本

官网 <https://pytorch.org/get-started/locally/>找到对应 官网会直接提供  下载指令)

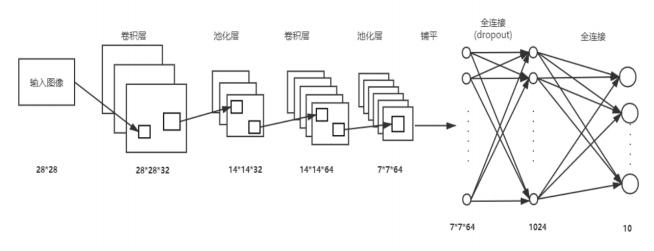
(: 一般

版本的

创建新环境并下载

检查是否安装成功

用 python 的 模块实现卷积神经网络。网络结构为一个输入层、两个卷积层、一个全连接层、一个输出 层



实验数据

手写体数字识别数据集。

实验流程

环境配置

1. 终端中输入

2. 进入 来说

3. 终端中使用

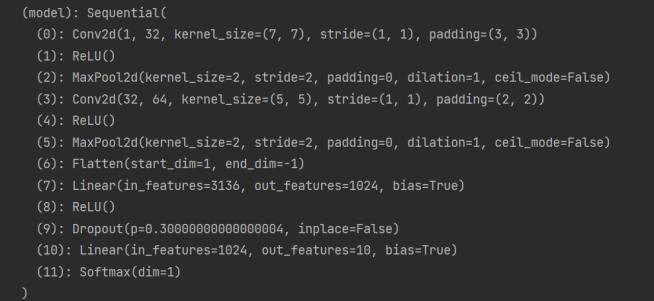
4. 进入

|  |  |  |
| --- | --- | --- |
| 1  2  3  4 | import torch  #成功 import 则安 装  torch . cuda . i s  #输出 True， 说明 服 | 成功  \_ a v a i l a b l e ( )  务器GPU可用 |

5. 本实验为了更好可视化训练和测试结果，需使用 安装  库

网络结构

使用2个卷积+池化板块,在 之后做一个 ,其中套用做正则化,使用激活函数,最后经过 运算后得到10分类的概率

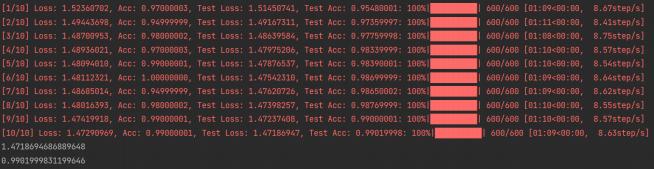


参数优化

原始参数为

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 |  |  |  |  |
| 初始值 | 100 | 10 |  | 0.7 |

按照上面的参数跑一边模型之后,发现其实初始参数的值并不差,结果如下

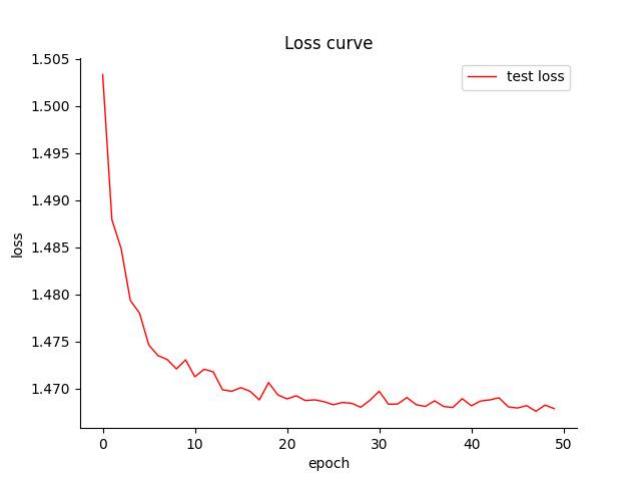


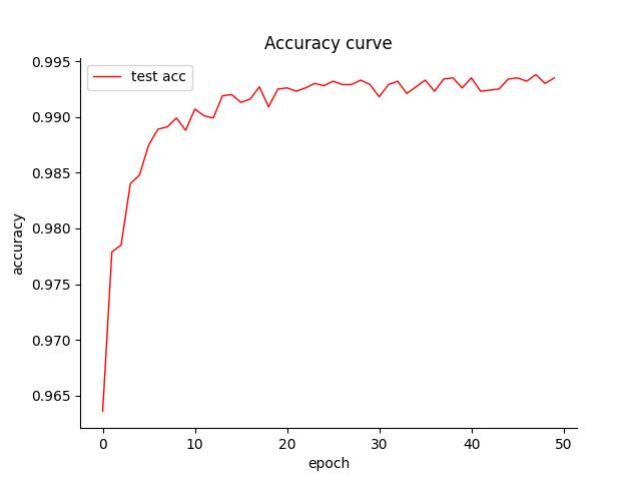
其实一般情况下,当我们选取一个比较大的的时候,说明它每次步数比较大,按照极端情况来看,假设我 的 ,其中为样本总数,那么很显然我们一次就会抓取完所有的样本,那就应该适当的放大



在这里是因为我们仅仅做的是一个参数寻优,只是比较它们哪一种组合的效果最好,因此我们选择定住某一个参数之 后再寻找其他参数的最优值

为了降低训练时间,我们先确定的最优值,设定最优 的参数范围  ,用 循环重复运行函数主体,将 得到的 和 保存下来,并据此画出其随 的曲线图





观察完上图之后我们发现 在23左右开始趋于收敛,但由于是在测试集上的结果,准确率会稍有上下浮动.

我认为我们要寻找的最优参数既要收敛,又不能太大,从上图来看,可能在后面还有比我所定义的"最优参数"跑出来的 还要高的,但总体来看提升并不明显,因为我的已经达到 的区间了,而加大 

提升的那一点准确率却是用巨大的时间成本换来的,因此我从保存下来的epoch的值种选取了   

以 附上画出曲线的代码,其中 \_是之前跑数据时保存下来的结果,要绘制 的曲线图只需要 更改文件名(之前也要保存 \_ )和纵坐标的值就可以了

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34 | import numpy as np  from matplotlib import pyplot as plt |
| def data\_read(dir\_path):  with open(dir\_path, "r") as f:  raw\_data = f.read()  data = raw\_data[1:-1].split(", ") # [-1:1]是为了去除文件中的前后中括号"[]"  return np.asfarray(data, float)  if \_\_name\_\_ == "\_\_main\_\_":  test\_loss\_path = r"D:\sukkart\my projects\neural\_network\exp2\test\_acc.txt" |
| y\_test\_acc = data\_read(test\_loss\_path)  x\_test\_acc = range(len(y\_test\_acc)) |
| plt.figure()  # 去除顶部和右边框框  ax = plt.axes()  ax.spines['top'].set\_visible(False)  ax.spines['right'].set\_visible(False)  plt.xlabel('epoch') # x轴标签  plt.ylabel('accuracy') # y轴标签  # 以x\_test\_loss为横坐标，y\_test\_loss为纵坐标，曲线宽度为1，实线，增加标签，训练损失， # 默认颜色，如果想更改颜色，可以增加参数color='red',这是红色。  plt.plot(x\_test\_acc, y\_test\_acc, color="red", linewidth=1, label="test acc") plt.legend()  plt.title('Accuracy curve')  plt.savefig('epoch\_test\_acc.jpg')  plt.show() |

在确定了 的最优值之后,我们来寻找 的最优值

给定三个参数的备选列表,使用一个三重 循环来找寻它们的最优值

|  |  |
| --- | --- |
| 1  2  3 | Batch\_size\_list = [16, 32, 64, 128, 256,512]  learning\_rate\_list = [1e-3, 1e-4, 1e-5, 1e-6]  keep\_prob\_rate\_list = [0.5, 0.6, 0.7, 0.8] |



for Batch\_size in Batch\_size\_list:

for learning\_rate in learning\_rate\_list:

for keep\_prob\_rate in keep\_prob\_rate\_list:

1

2

3

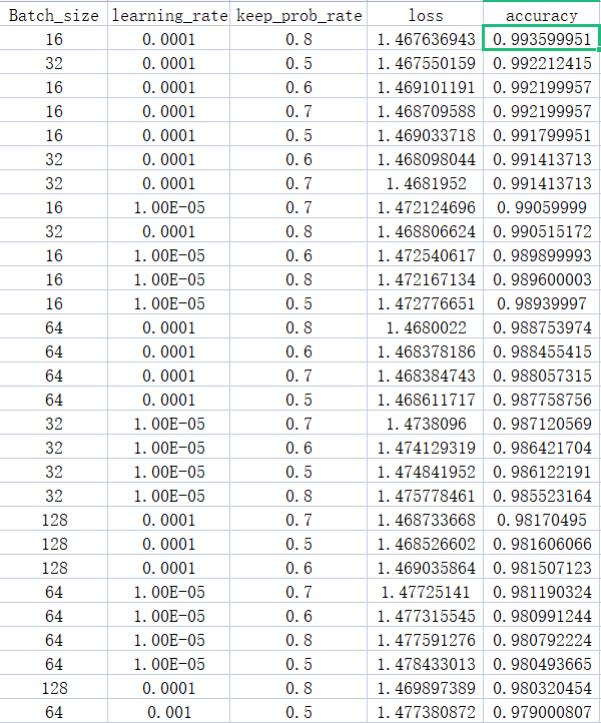


在本机上测试没有语法错误之后选择丢到itc的服务器上跑

下面附上参数寻优及保存结果部分代码

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21 | params = ['Batch\_size', 'learning\_rate', 'keep\_prob\_rate', 'loss', 'accuracy']//列名 with open('params.csv', 'w', newline='') as csv\_file:  writer = csv.writer(csv\_file)  writer.writerow(['Batch\_size', 'learning\_rate', 'keep\_prob\_rate', 'loss', 'accuracy'])  writer.writerow(params)  for Batch\_size in Batch\_size\_list:  for learning\_rate in learning\_rate\_list:  for keep\_prob\_rate in keep\_prob\_rate\_list:  params = []//每次保存五个值 ,在循环开始时清空  params.append(Batch\_size)  params.append(learning\_rate)  params.append(keep\_prob\_rate)  ...//程序主体  print(test\_loss)//测试集上的loss值  print(test\_acc)//测试集上的准确率  params.append(test\_loss)  params.append(test\_acc)  with open('params.csv', 'a', newline='') as csv\_file:  writer = csv.writer(csv\_file)  writer.writerow(params)  print(params) |

保存出来的 文件有96条数据,我们对其排序后取最高值,下面时是排序后的部分结果



基于上述方法,寻得最优参数

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| 参数名称 |  |  |  |  |
| 参数值 | 16 | 26 |  | 0.8 |

问题记录

1. 由于先入为主的思想,一开始并没有想到要画和曲线,因为的初始值 为10,我想当然的就想在里面找到一个最优解,后面仔细一想发现显然不对,于是将范围改为 了 

2. 知道范围之后,需要保存最后的值来画曲线,并不需要保存中间训练集训练过程的值,我一开始因为想着保存中间 结果却又不知以何种格式保存花了不少时间

3. 确定完 之后,发现跑一次模型需要 左右, 由于剩下的三个参数有96种组合方式,那么寻优的时间 将达到 ,一开始放在上跑总是会掉线,后来学习指令可以后台挂起运行代码了,之后中间有一 次过了一晚上确实跑出结果了,但由于那时保存文件使用的 覆盖参数而非 追加参数导致结果并没有保存 下来,后面再次使用指令的时候每次过一段时间登录还是会掉线,因此最后想到了分段跑保存结果,每 次选择30种组合放进球跑,这样子模型运行一次的时间在 左右,即使掉线结果也已经保存,最后把四份结果 拼起来得到最终的 文件

实验心得

本次实验通过 实现了卷积神经网络的搭建与训练过程，通过改变神经网络的超参数，比较前后模型学习 能力，与理论知识相结合，有很大的收获。根据实验结果对比分析可以看出，随着epoch的增大，模型的拟合能力 越来越强，但是，随着epoch增加，模型也会发生过拟合，如果增大训练层数也会不可避免会发生梯度消失或者梯 度爆炸等问题。同时，我也得到了如下心得体会：

1. 卷积网络入门其实没有想象的那么难，但是越往高处走越来越需要我们去细心琢磨。

2. 在卷积神经网络中，有必要去深入了解一些经典的卷积神经网络。由浅入深，例如：  、 、 、网络、网络等等。

3. 在了解这些网络的时候应该多考虑别人为什么要这么做，这样做又什么好处。

4. 在学习卷积神经网络的过程中，应该更多地去做一些实践，这样不仅仅能够更加深入的去理解网络的构成流 程而且能产生更大的兴趣去学习去钻研。

5. 在学习卷积网络的过程中，应该适当的去修改一些经典网络，看看在修改过后有什么的不同，在从这些不同 的地方吸取一些教训，然后在以后碰到类似的问题的时候能够有一些经验。