**程序设计实训-小作业3**

**赖宇航 2020010920** [**laiyh20@mails.tsinghua.edu.cn**](mailto:laiyh20@mails.tsinghua.edu.cn)

**作业文件说明**

1. 根据作业要求，本次上交的文件将包含原压缩包中的所有文件，但是会删掉LeNet-PyTorch\_initial/data/MNIST文件夹；
2. 根据调整后的代码，直接运行train.py，除了该文件本来的一系列输出之外，还会在每次epoch输出结束后，Accuracy输出前，输出一系列图片以及对应的推理结果（用于作业2.7部分），手动关闭弹出的所有图片后函数继续运行；
3. 常规输出完毕后，还会跳出标题为“Loss per epoch”的图片，关掉后函数返回。

**2.1 图片可视化**



**2.2 推理**

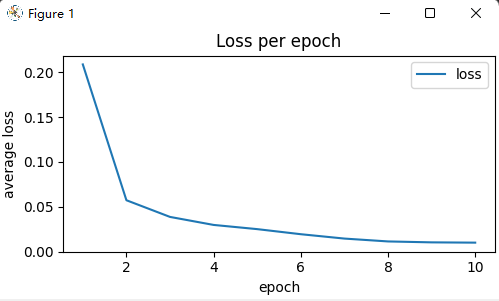
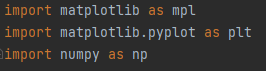
**第一次实验，**

**没有输出结果，推测有两种可能，一是运行时间不足，没有打印结果；二是图片大小不够，不能完整呈现结果。最终决定重新进行实验**

**第二次实验**

**备注：由于上一次呈现的结果不佳，重新运行了task-1，得到新的结果后强行放大，使其能够容纳pred的结果，最终结果如右图所示，图片经缩小以适应排版。（另：由于图片变大，训练时间相应变长）**

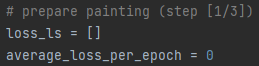
**2.3 绘制Loss曲线**



最终绘制出的图线如上，其中横轴代表epoch的编号，

纵轴代表对应编号的epoch所有step上loss的平均值

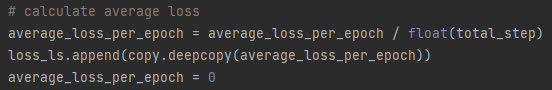
绘制过程分三步：

第一步：准备阶段

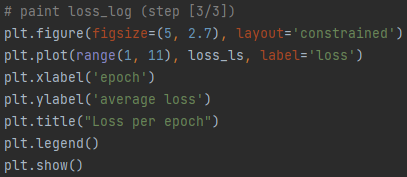
首先引入第三方库matplotlib，然后在train函数体中插入一个列表loss\_ls用于存储数据，由于本次记录的数据是平均值，故还需要记录总和。

第二步：记录每个epoch的loss平均值并装进列表

这其中需要注意的是深度复制，以及每次记录后都要将上次的数值清空（这里由于忘了后者致使推倒重来）



第三步：绘制图像

这里利用matplotlib的pyplot-style风格，人为设置x轴和y轴，其余没有详细设定，matplotlib的默认设置效果就很好

**从而，绘制图像的代码内嵌在train函数里，由于添加了plt的show方法，直接运行train.py，在输出结束后会自动跳出matplotlib绘制的loss折线图。**

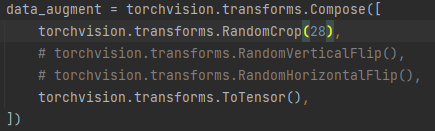
**2.4 更换优化器**



但是从结果上看，使用Adam做优化比使用SGD做优化效果好，体现在模型正确率高和loss曲线收敛速度快，故以下实验切换回Adam优化器

**2.5 添加数据预处理**

**添加方法如下图所示**

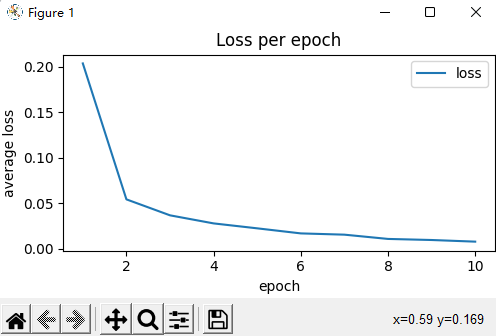


**各个epoch的平均loss**

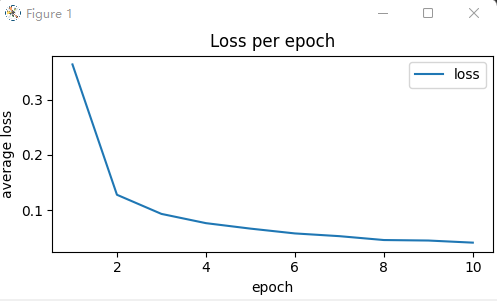
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| epoch编号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 正常 | 0.217 | 0.057 | 0.040 | 0.030 | 0.024 | 0.018 | 0.016 | 0.012 | 0.010 | 0.011 |
| 剪裁 | 0.199 | 0.056 | 0.037 | 0.027 | 0.024 | 0.016 | 0.016 | 0.011 | 0.010 | 0.007 |
| 垂直翻转 | 0.365 | 0.127 | 0.093 | 0.075 | 0.066 | 0.056 | 0.053 | 0.046 | 0.046 | 0.040 |
| 水平翻转 | 0.378 | 0.149 | 0.098 | 0.079 | 0.070 | 0.060 | 0.055 | 0.049 | 0.045 | 0.042 |

**各个epoch的模型测试准确度（%）**

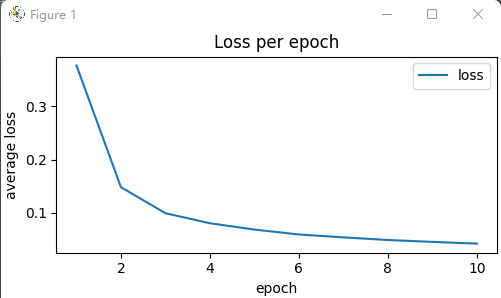
|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| epoch编号 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
| 正常 | 98.14 | 98.78 | 98.96 | 98.82 | 98.95 | 99.17 | 99.13 | 99.00 | 99.03 | 99.21 |
| 剪裁 | 98.19 | 98.75 | 98.92 | 99.00 | 99.12 | 99.11 | 99.12 | 99.04 | 99.16 | 99.07 |
| 垂直翻转 | 94.88 | 96.74 | 97.34 | 97.39 | 97.64 | 97.37 | 97.96 | 97.90 | 98.06 | 97.93 |
| 水平翻转 | 89.94 | 96.33 | 97.53 | 97.45 | 97.85 | 97.69 | 97.71 | 97.95 | 97.96 | 97.96 |

**随机剪裁**

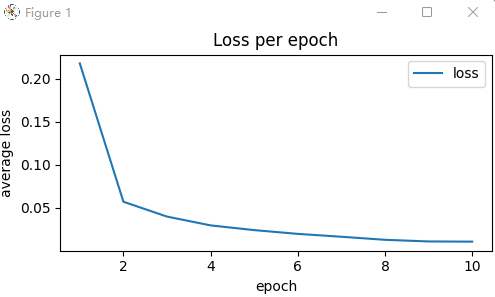
**随机垂直翻转**



**随机水平翻转**



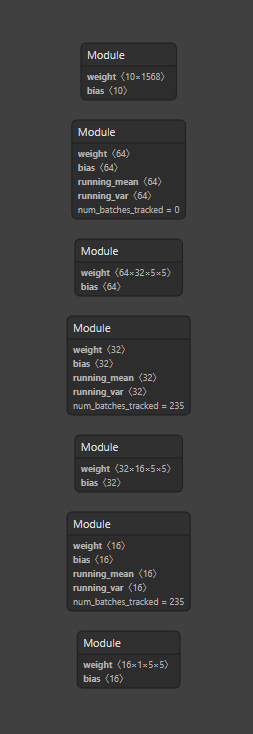
**正常情况**



**评价：根据测试获得的数据，随机垂直翻转和随机垂直翻转对于模型训练效果而言都是负提升，造成loss整体上升，而准确度整体下降。相比而言，以训练数据中最小的图片尺寸为剪裁目标的随机剪裁可以对训练效果作出轻微的优化，但是优化效果整体上非常微小，效果不太明显。**

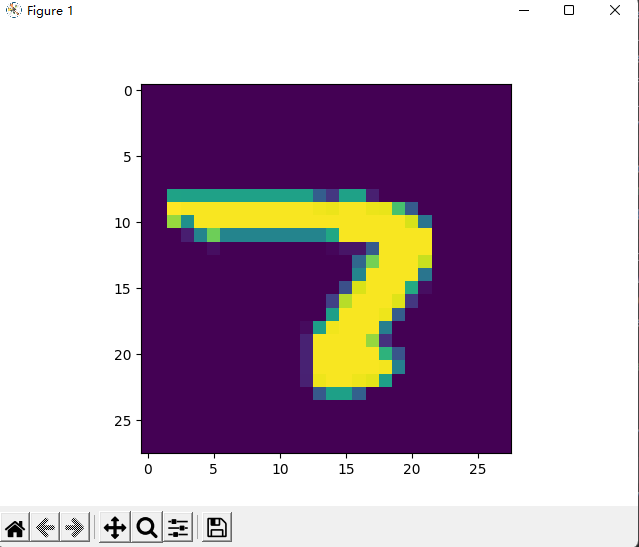
**另：由于另外两个起到反效果，故此后的实验将只采用随机剪裁来进行预处理**

**2.6 模型导出及模型可视化**

****

**2.7 错误样例分析**

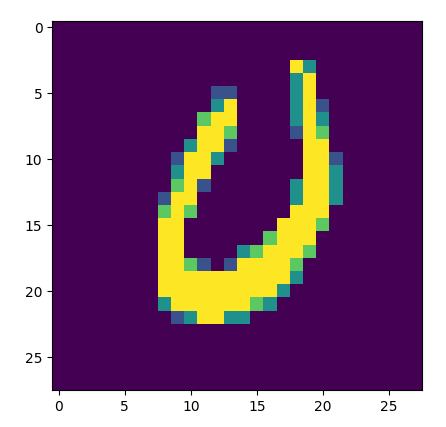
**案例一：**

****

**推理结果为：7**

**错因分析：这张图片即使对人类（我）而言，都有一定判断难度，但人类（我）根据下方的小尾巴可以判断是2，机器可能是由于将那截尾巴忽略了。**

**由于用Adam优化的模型正确率太高，以至于半天找不出一个错误，所以从这里开始改用效果差一些的SGD进行优化**

****

**案例二：**

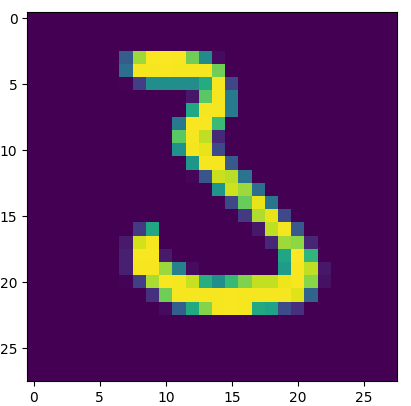
****

**推理结果为： 6**

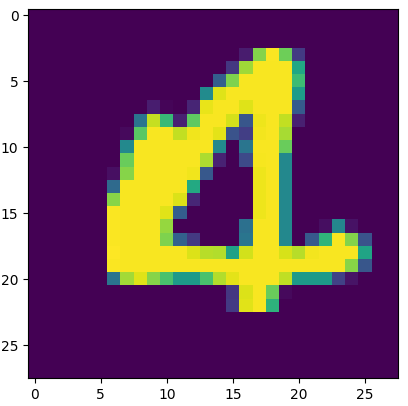
**错因分析：看上去该图片更像是数字0，但是由于没有封口，而且一边比另一边长，所以学习不充分的模型会产生误判。**

**案例三：**

****

**推理结果为：2**

**错因分析：以人类视角，这显然是3，机器可能是将处在下方的两条线都标成了数字“2”下面的一横（有人确实会把那一横写得很歪）**

**案例四：**

****

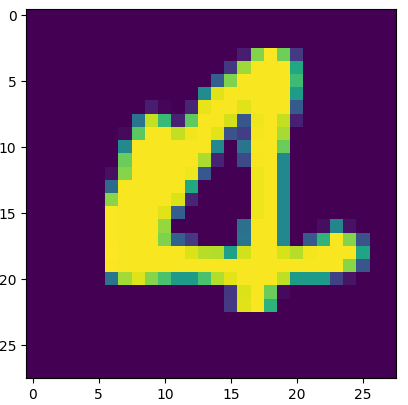
**推理结果为： 2**

**错因分析：可能是AI在试图读取图片信息时直接忽略了数字“4”的那一竖，剩下的“折横”看起来就像是头部短了一点的“2”**

**案例五：**

****

**推理结果为：0**

**错因分析：可能是AI只检测到中央的一个空洞，但是忽略了周边多出来的枝节，这个“4”相较于标准的数字“4”而言，确实太过圆润了。**