OOD 实验报告

杨炫锟 523030910248

2024 年 7 月 21 日

目录

1 模型构建 **2**

2 问题引入 **2**

3 数据增广 **4**

4 可视化分析 **9**

4.1 参数敏感性及参数敏感性分析 9

4.2 特征图可视化 10

4.3 卷积核可视化 13

4.4 注意力分析（grad_cam） 13

5 IRM **15**

6 小组创新 **16**

6.1 创新思路 16

6.2 举例 17

6.3 待解决的问题 19

7 实验总结 **19**

1 模型构建

在本次实验中，我们使用的模型是 LeNet 网络，其具体网络结构如下：

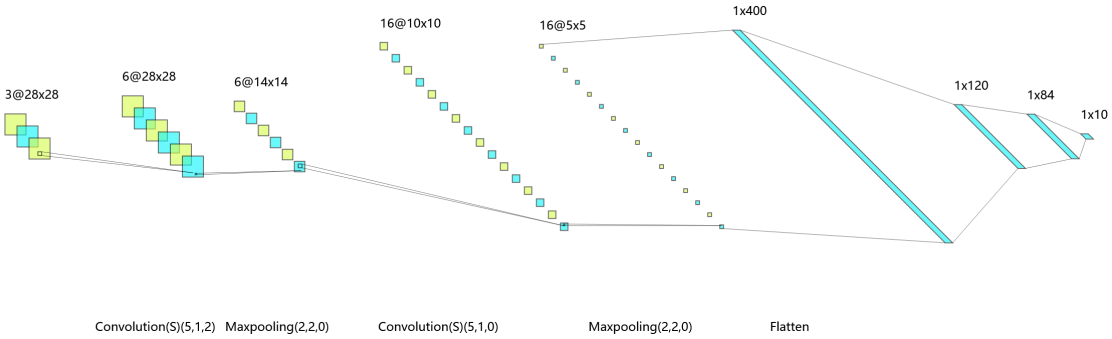

图 1: LeNet

看到网上很多博客的解读，大部分会有 3 层卷积，也就是将我们这个 lenet 的第一个全连接层改为卷积。最开始我写的网络结构也是如此，但是发现在做 grad_cam 的时候遇到了麻烦，若是三层卷积，最后的卷积结果直接变成 120*1*1，grad_cam 难以进行下去。后来，我发现，最后一层卷积和全连接的性质是一样的，二者参数量也是一样，120*16*5*5+bias。于是，我将其改成了现在的这个结构，这样一来，grad_cam 也能正常进行了。

该神经网络作图开源：NN-SVG

其余在本次实验过程中，找到的使我对 CNN 理解更生动、透彻的网页：CNN explainer,机器学习有意思的讲解

2 问题引入

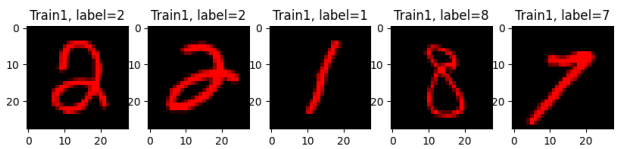

图 2: Train 1

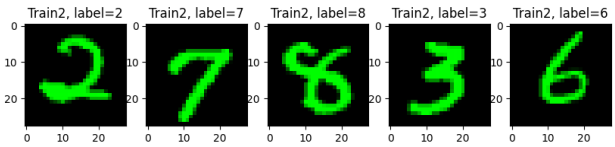

图 3: Train 2

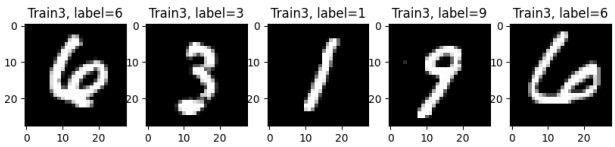

图 4: Train 3

初级阶段，在三种训练环境中，数据分别为红色数字黑色背景、绿色数字黑色背景、白色数字黑色背景，在测试阶段，数据分别为黄色数字白色背景、蓝色数字白色背景。

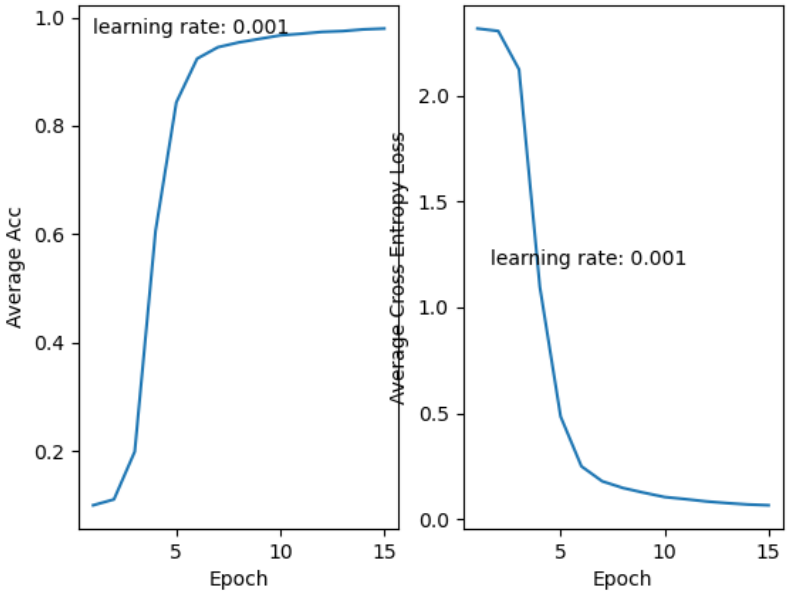

图 5: Basic train

发现，即使在 train1、train2、train3 上的平均正确率已经达到一个较高值，在 test1、test2 上测试出来的正确率却很低，10% 左右，跟乱猜差不多。

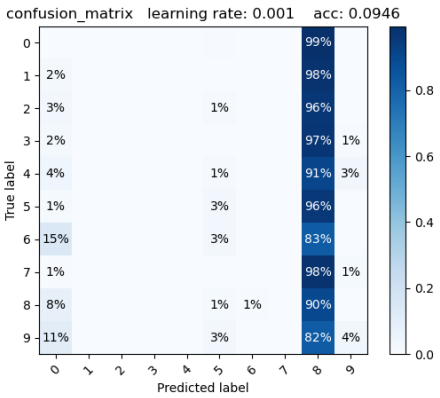

图 6: Basic test1

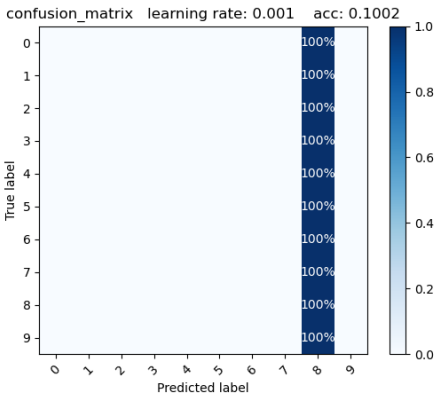

图 7: Basic test2

由此，我们对 OOD 现象有了一个初步的认识，分别在不同的字体颜色和背景颜色的组合下训练和测试，测试结果总是会很低，其实也就是该模型的泛化性很差，受背景等 spurious feature 的影响较大.

混淆矩阵作图参考：混淆矩阵 _CSDN

3 数据增广

为应对模型泛化性差的问题，我们采取对数据进行增广的策略，保证模型“见过”颜色组合尽可能多的数据。对此，我们对数据进行了第一次增广（主要由队友余俊洁负责各种颜色组合数据的调节和生成），其中增广之后的数据如下：

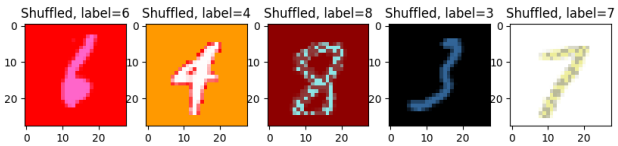

图 8: Shuffled augmented trainset 1

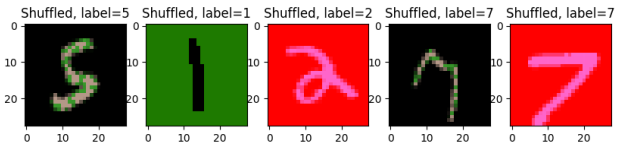

图 9: Shuffled augmented trainset 2

我们将增广后的所有数据综合之前的 train1、train2、train3 得到 data-12，并将其 shuffle，并取 3:1 的训练验证占比，结果如下：

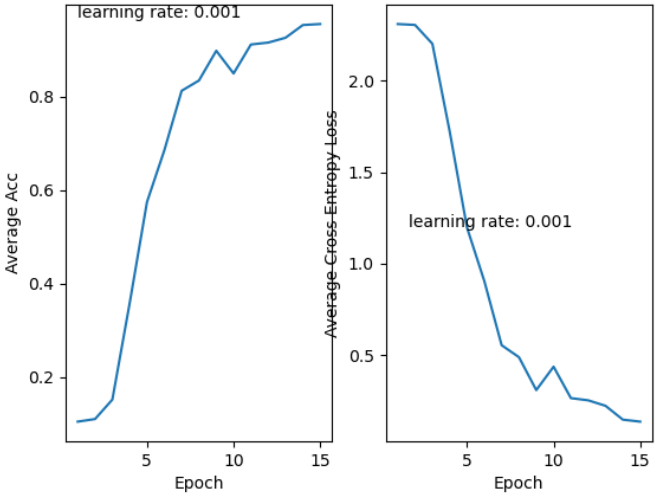

图 10: Train on data-12

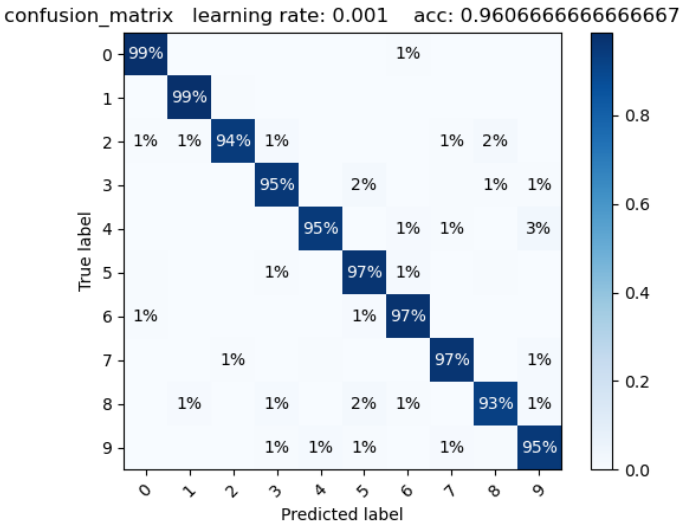

图 11: Validation on data-12

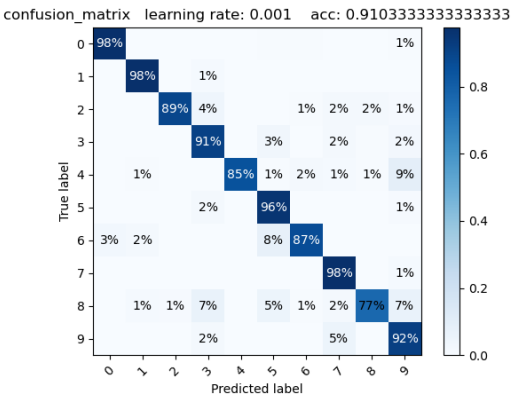

图 12: Test1

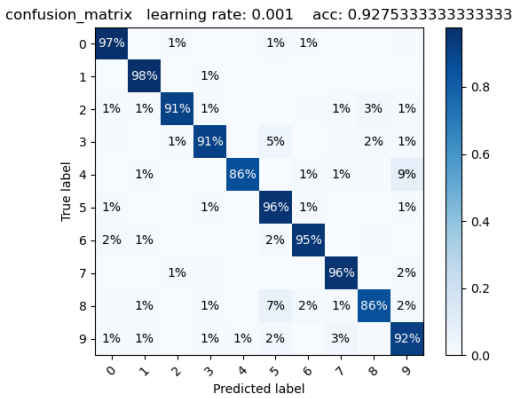

图 13: Test2

在训练该模型时，最开始，我取初始学习率为 0.001，并加入学习率优化器，改变步长取 10，乘积因子取 0.1，发现在 epoch10 附近 acc 会骤降再上升，后取 gamma=0.8，便得到上面的结果，在 train 和 validation 上都达到了 95% 以上的正确率，在 test1、test2 上都有 90% 左右的正确率。可见，在进行了数据增广（模型并未“见过”test1、test2）之后，模型的泛化性得到了不少提升，由此得到 model-12。

此外，队友还对数据添加了一些噪声，得到如下的 data-3：

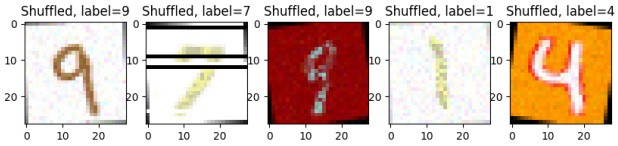

图 14: Shuffled trainset pro 1

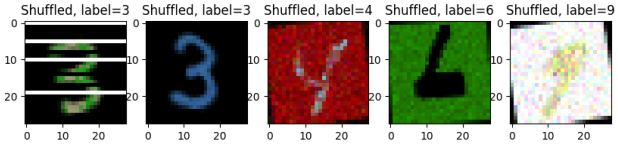

图 15: Shuffled trainset pro 2

我们在 model-12 的基础上，在 data3 上取 75% 的训练占比，再在 test1、test2 上测试，得到了 model-3，结果如下：

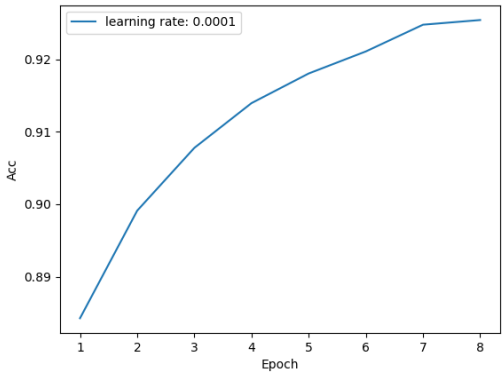

图 16: Train on data3

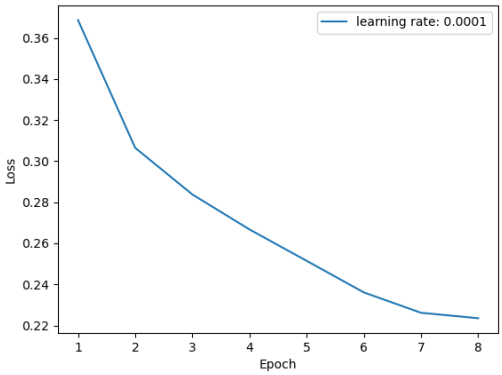

图 17: Validation on data3

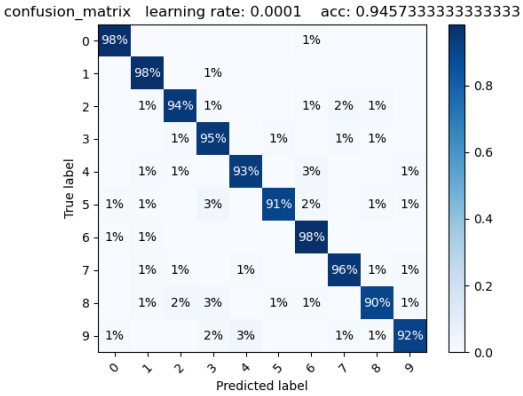

图 18: Test1

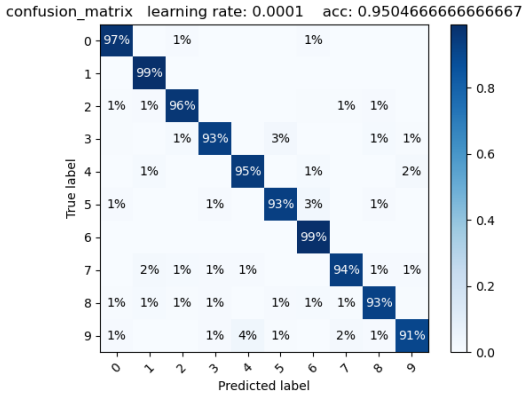

图 19: Test2

发现，经过 data-3 训练后，在 test1、test2 上的正确率提高到了 95% 以上。

此外，队友还对数据进行了进一步地增强，加了光晕等效果，但是训练效果很不理想，容易过拟合，故舍弃。以下是 hard 版本的数据：

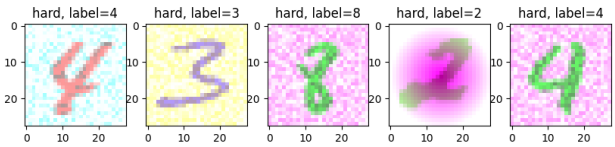

图 20: Shuffled hard trainset 1

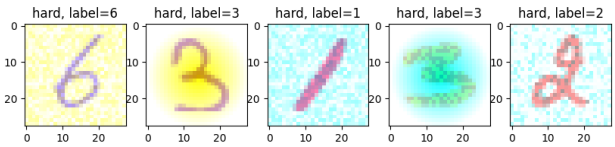

图 21: Shuffled hard trainset 2

4 可视化分析

4.1 参数敏感性及参数敏感性分析

在该分析中，我关闭了学习率优化器，分别选取学习率为

$$0.0001, 0.0005, 0.0008, 0.001, 0.003$$

epoch 为 1 到 15，在 data-12 上训练和验证 3 : 1（各组取同样的随机数种子来 shuffle 数据），结果如下：
在训练集上：

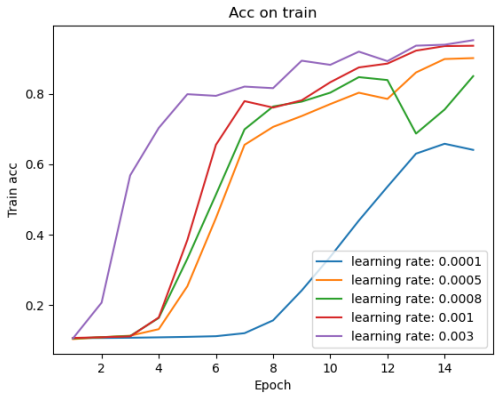

图 22: Train acc on data-12

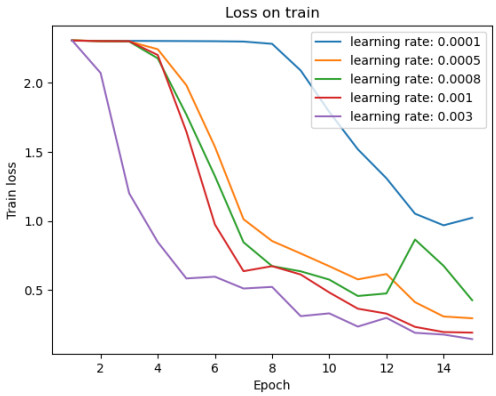

图 23: Train loss on data-12

在验证集上：

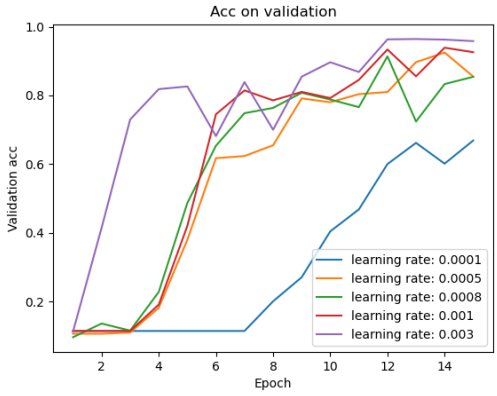

图 24: Validation acc on data-12

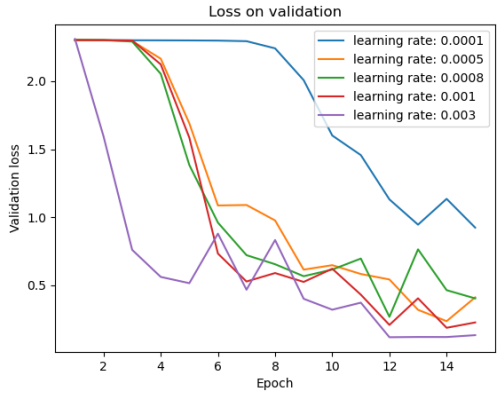

图 25: Validation loss on data-12

test1、test2 上的平均 acc 和 loss

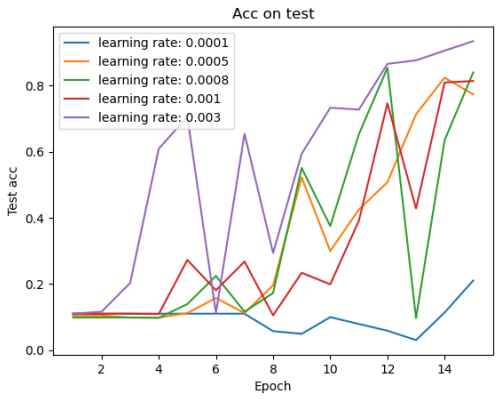

图 26: Test acc on data-12

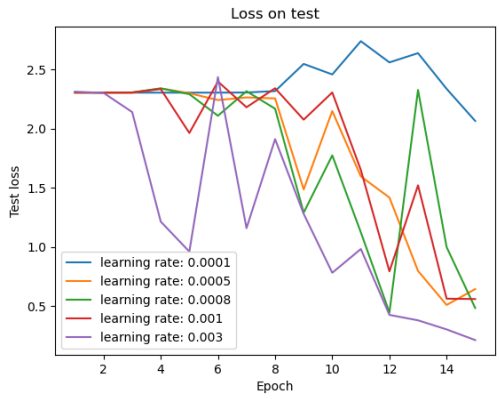

图 27: Test loss on data-12

可见, 当学习率较小时，结果的变化幅度较小，当学习率过大时，又容易导致波动过大，取 0.001 较为合适。关于 epoch，仍是取 15 合适。

4.2 特征图可视化

原图如下（本人手写）：

图 28: figure-3

第一次卷积和激活后的特征图：

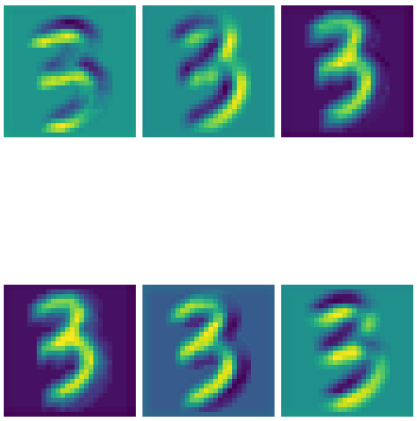

图 29: Convolution 1

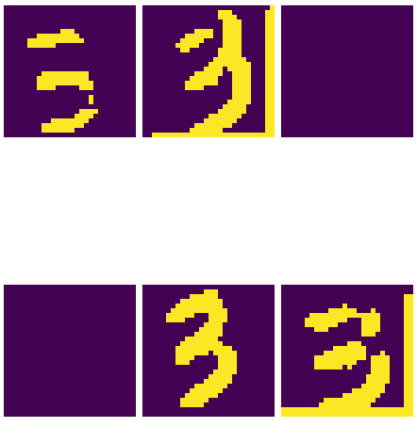

图 30: Activate 1

第一次池化和第二次卷积后的特征图：

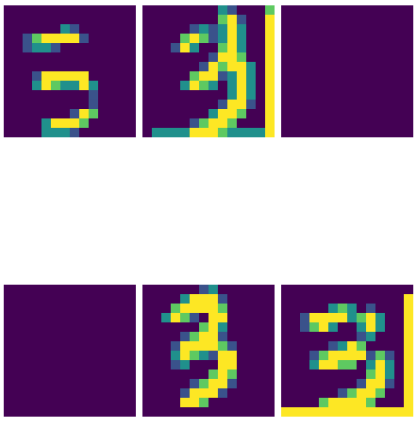

图 31: Maxpooling 1

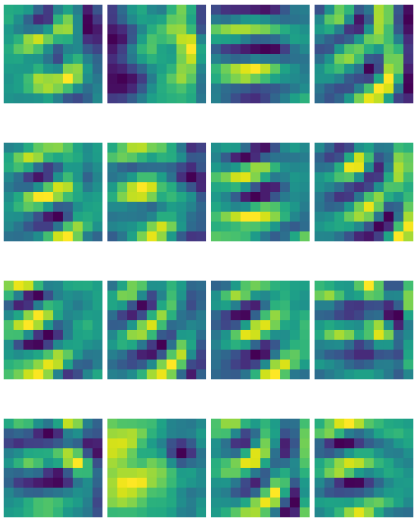

图 32: Convolution 2

第二次激活和池化后的特征图：

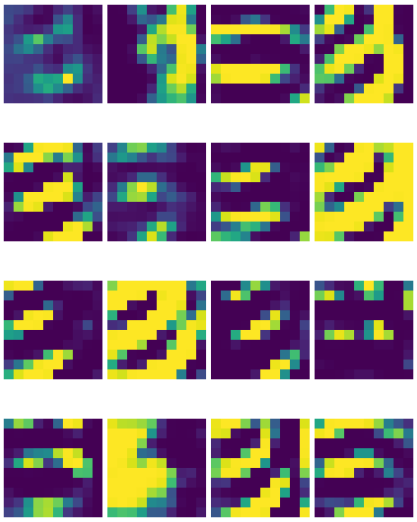

图 33: Activate 2

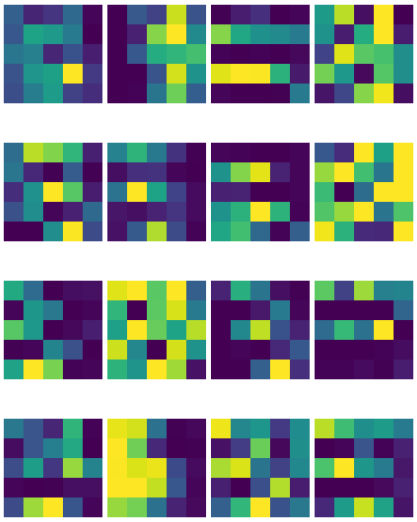

图 34: Maxpooling 2

可见，在最开始的几次操作中，是能够桥清楚地看出来数字的，到后来

逐渐变得抽象和高度浓缩，这其实体现出了模型去“认识”图片的过程。

特征图及卷积核作图格式参考：Larry 同学（哔哩哔哩）

Larry 同学 bilibili

4.3 卷积核可视化

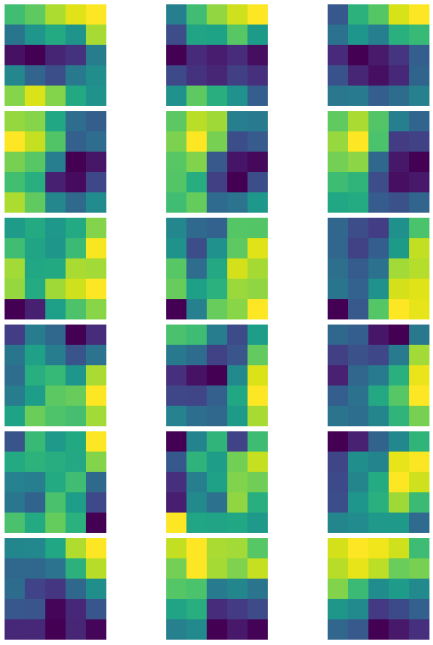

图 35: Kernel 1

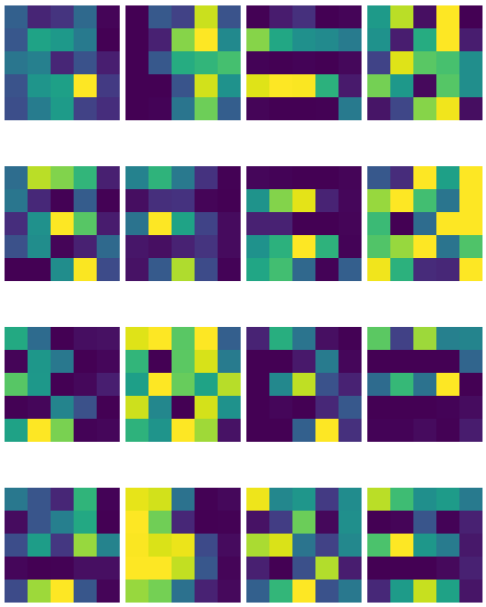

图 36: Kernel 2

4.4 注意力分析（grad_cam）

原图如下（本人手写）：

图 37: figure-9

取用 model-3，将目标层定为第二个卷积层，得到的热力图：

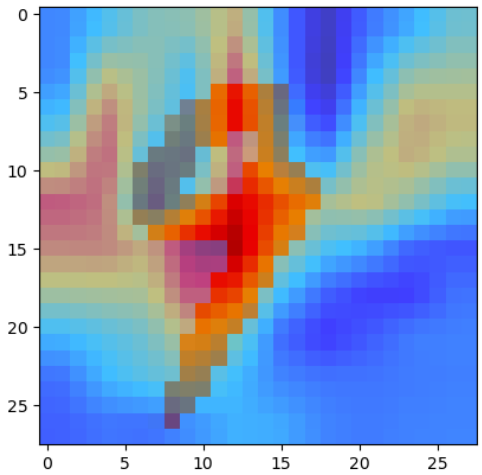

图 38: grad_cam on figure-9

发现，该模型基本能将主要注意力放在数字 9 的关键部位，但也有不少注意力分散到了背景上。

Grad_cam 作图参考：paper

pytorch_grad_cam 的用法 _CSDN

5 IRM

在该模块中，最初我看了一下关于 IRM 的论文及诸多阅读笔记（具体网址贴在下面），然后我的思路是在模型的输出中添加上预测环境标签的一项，对该标签的预测以及真实数字标签的预测的损失进行正则化，再去做 Gradient Desent，但是，跑了很多次，结果都不理想。环境预测损失项的系数 λ 若是太小，容易受环境的影响，若是过大，真实标签预测的正确率又会过低。后来，我在 GitHub 上找到了一份开源的 MNIST 二分类的 IRM 实现，该博主对比了 IRM 和 ERM 两种思路下的训练和测试正确率。

关于该开源代码，我的解读有以下几点：

1. 博主使用了放缩过后的 IRMv1，个人觉得很厉害的是，其中关于环境的 penalty 会随 epoch 来改变（系数 = 1.6*epoch），这个问题在那篇知乎中有提及，减小了 overfitting 的风险

2. 注意求导的对象，应该是对 dummy_w，虚拟权重的值不会影响模型的预测结果，它只是在计算惩罚项时起到了一个辅助作用，个人觉得这就是原始 IRMv1 表达式中的 w 一项

3. loss 要分奇偶去算，是因为传入的 traindata 奇偶索引处的环境不同，分别来自 trainset1 和 trainset2

4. 在论文中，作者也给出了对应使用 minibatch SGD 的时候的一个获取“unbiased estimate of the squared gradient norm”的方法，上述代码中都有体现

在本地，我将源代码中的网络结构改成了 LeNet 二分类的形式，对最终结果作图分析，得到的结果如下：

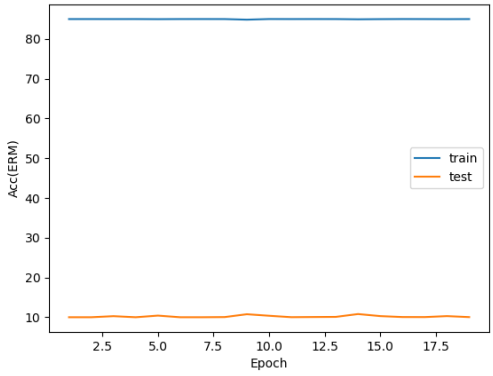

图 39: ERM

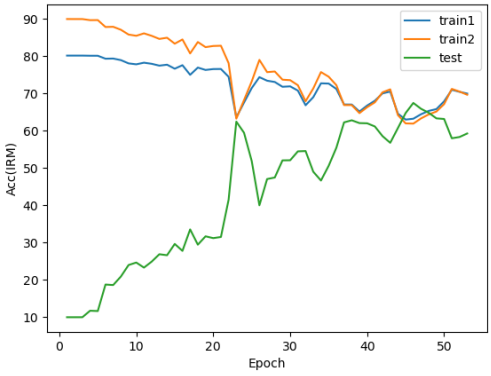

图 40: IRM

发现，采用 IRM 的算法，加了关于环境的惩罚项之后，在源码中的 train1、train2 上训练，在 test 上较 ERM 的正确率提高了不少。

我本地下的源码,源码博主推荐的 official，IRM 相关的论文,IRM 相关研究 _ 知乎, paper,paper with code

6 小组创新

6.1 创新思路

在该部分，我和队友致力于将复杂的背景和数字组合全转化为简单的图论方式思考，并得出下面的一些思考：

1. 对图片进行高斯平滑去噪，这样就去除了背景噪声的影响
2. 对 RGB 三个通道的数组进行遍历，对梯度边缘的数值赋为 1，其余全为 0，即是相邻有色差的色块的边缘为 1，色块内部与图形边界为 0，对于每个点记录四个方向上的梯度数据，记作梯度图
3. 将值为 1 的数值所对应的矩阵点相邻连接，转化为图论问题，由此关联到图论中的邻接矩阵

4. 在梯度图中，随机取点：若值为 1，利用“逢弯不拐，逢弯少拐”的原则进行边缘的勾勒，并将勾勒过的点赋值为-1；若值为 0 或-1，跳出，继续随机取点；每次遍历一个点，记录下该点下一个遍历数据的方向

5. 当连续跳出达到某规定阈值之后，停止随机去点，勾勒结束，每个点都应该有五个数据，将这些数据及标签传给模型进行训练

注：逢弯不拐，逢弯少拐原则：

1. 若在图像的外侧，以顺时针遍历为例，从该结点“后方”出发，顺时针遇到的第一条边连接的结点为下一结点

2. 若在图像的内侧，以逆时针遍历为例，从该结点“后方”出发，顺时针遇到的第一条边连接的结点为下一结点

下面给出例子来说明该原则：证明：利用该原则勾勒出来的边即是图形的边缘：
（反证）外侧，以顺时针遍历为例，倘若选出来的不是顺时针遇到的第一条边，那么一定存在一条边位于“该边缘”的外侧，与所选出来的边为边缘，矛盾，故得证。内侧，以顺时针遍历为例，倘若选出来的不是逆时针遇到的第一条边，那么一定存在一条边位于“该边缘”的外侧，与所选出来的边为边缘，矛盾，故得证。

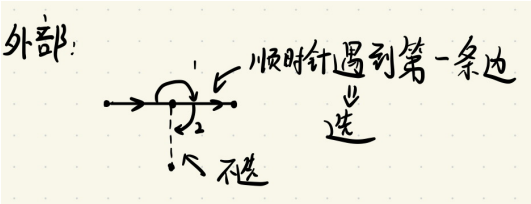

图 41: Outside points

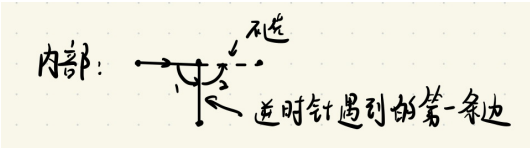

图 42: Inside points

6.2 举例

纯色无交叉：

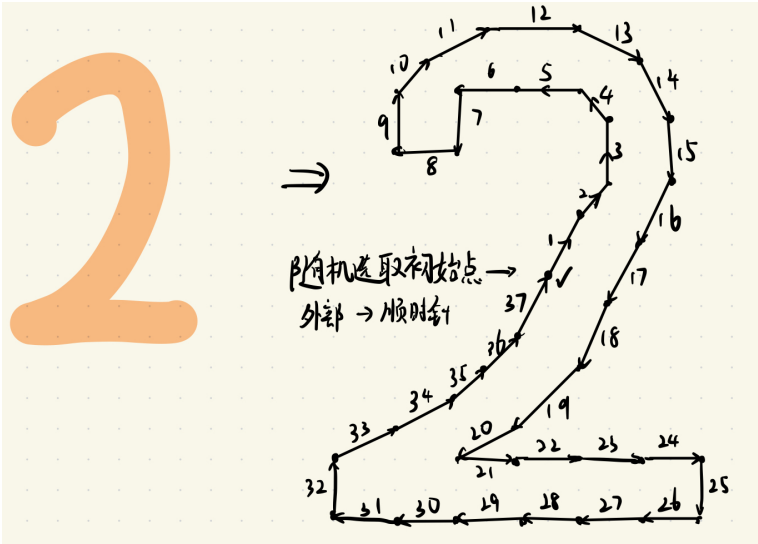

图 43: eg-1

纯色有交叉：

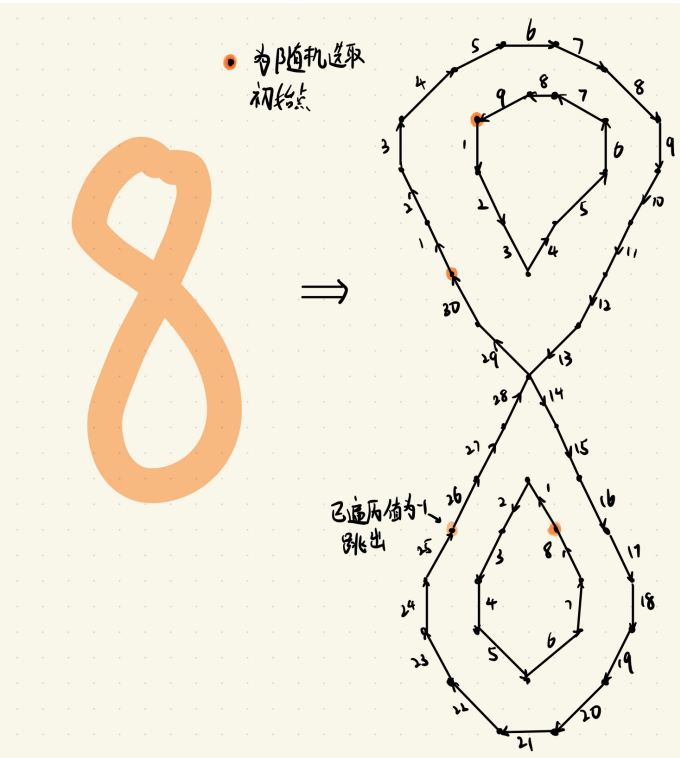

图 44: eg-2

杂色：

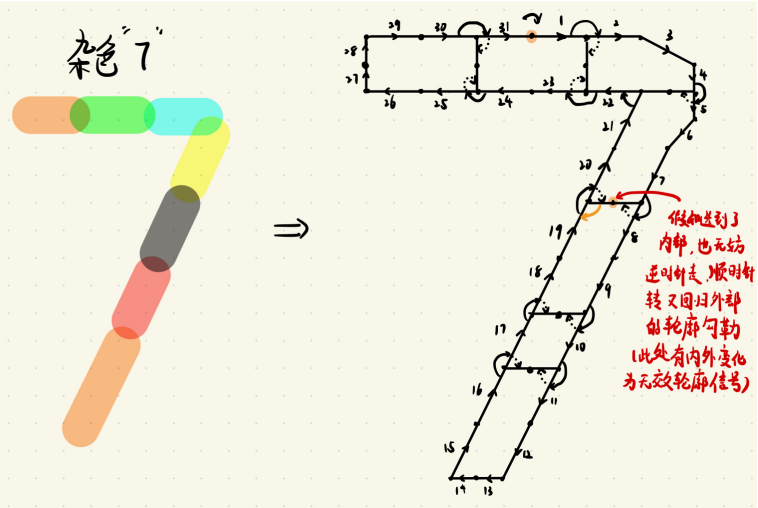

图 45: eg-3

6.3 待解决的问题

1. 如何快速得到“梯度图”
2. 如何确定“内外”
3. 图片数据较小的时候，该思路可能失效，如何提高泛化性
4. 图片遍历过程中，存在随机性，如何控制

7 实验总结

1. 通过本次实验，我基本了解了 OOD 问题及其常见的解决方式
2. 通过本次实验，我基本掌握了深度学习中的一些基本技巧，如模型基础训练、可视化分析等
3. 通过本次实验，我对将来更深入的学习和研究激发了浓厚的兴趣，更有志于向前沿科研看齐

4. 通过本次实验，我认识到在将来的工作中，与队友合作的重要性

最后，我们将本次任务中的关键代码以及报告都打包上传至了 GitHub 上：GitHub 主页