课程项目二报告

陈华宇 自33 2023010964

设计深度学习方法

卷积神经网络模型

我将设计的卷积神经网络分为卷积层部分和线性层部分，前面的卷积层部分负责提取图片的特征，然后线性层负责完成分类任务。卷积层中有三个卷积块，每个卷积块由一个nn.Conv2d，一个非线性层，一个池化层组成。这样三个卷积块相连，图片的高层特征就能被提取出来。接下来的线性层部分，让前面卷积层部分的输出经过两个线性层，中间经过一个非线性层，得到num_classes维的输出。

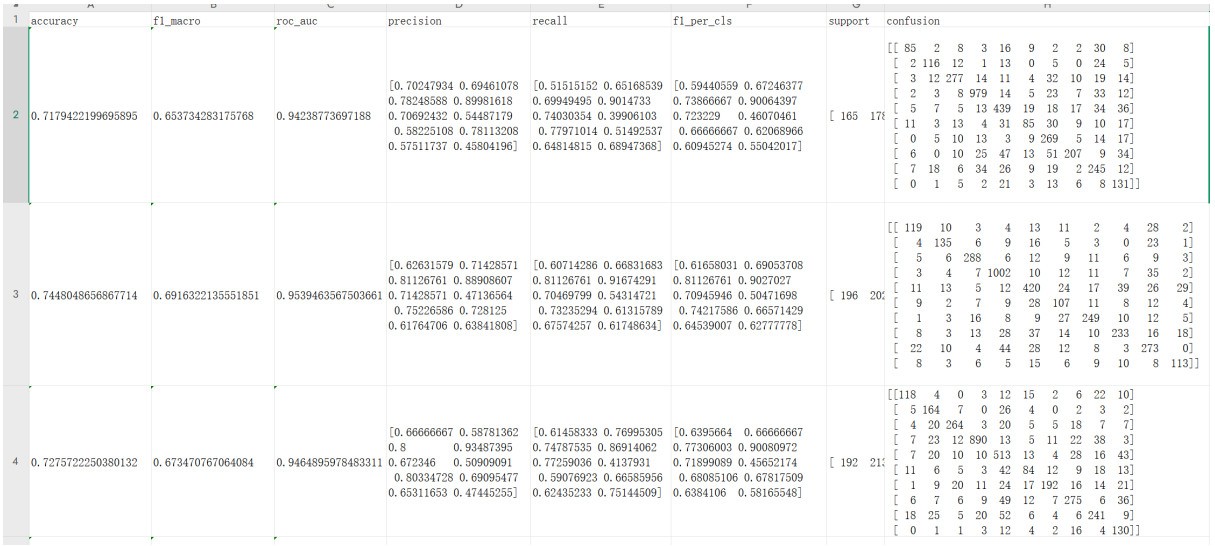
深度学习框架实现

我利用pytorch作为该卷积网络的框架实现。让我设计的模型继承nn.Module，然后再重写forward方法，补充train_model和evaluate_model方法，就有了一个完整的可以运行的深度神经网络。

测试方法和模型评估

我采用了K-折交叉验证来评估模型。以5折为例，将总样本集合等分成5份，每一折独立训练。在每一折训练的时候，初始化一个新的模型，让一折作为验证集，其余4折作为训练集。在验证集上评估的时候，将模型中间的概率分布也提取出来，由此计算Accuracy, Precision, Recall, F1, confusion matrix等等指标。这种验证方法能够充分利用数据，比较各折之间的指标也可以看出模型性能是否稳定，对模型的泛化能力也有保障。

训练baseline model得到的前三折的结果如下：



	accuracy	f1_macro	roc_auc	precision	recall	f1_per_cls	support	confusion
2	0.7179422199695895	0.653734283175768	0.94238773697188	[0.70247934 0.69461078 0.78248588 0.89981618 0.70692432 0.84487179 0.58225108 0.78113208 0.57511737 0.45804196]	[0.51515152 0.65168539 0.69949495 0.9014733 0.74030354 0.39906103 0.77971014 0.51492537 0.64814815 0.68947368]	[0.59440559 0.67246377 0.73866667 0.90064397 0.723229 0.46070461 0.66666667 0.62068966 0.60945274 0.55042017]	[165 178	[[85 2 8 3 16 9 2 2 30 8] [2 116 12 1 13 0 5 0 24 5] [3 12 277 14 11 4 32 10 19 14] [2 3 8 979 14 5 23 7 33 12] [5 7 5 13 439 19 18 17 34 36] [11 3 13 4 31 85 30 9 10 17] [0 5 10 13 3 9 269 5 14 17] [6 0 10 25 47 13 51 207 9 34] [7 18 6 34 26 9 19 2 245 12] [0 1 5 2 21 3 13 6 8 131]]
3	0.7448048656867714	0.6916322135551851	0.9539463567503661	[0.62631579 0.71428571 0.81126761 0.88908607 0.71428571 0.47136564 0.75226586 0.728125 0.61764706 0.63841808]	[0.60714286 0.66831683 0.81126761 0.91674291 0.70469799 0.54314721 0.73235294 0.61315789 0.67574257 0.61748634]	[0.61658031 0.69053708 0.81126761 0.9027027 0.70945946 0.50471698 0.74217586 0.66571429 0.64539007 0.62777778]	[196 202	[[119 10 3 4 13 11 2 4 28 2] [4 135 6 9 16 5 3 0 23 1] [5 6 288 6 12 9 11 6 9 3] [3 4 7 1002 10 12 11 7 35 2] [11 13 5 12 420 24 17 39 26 29] [9 2 7 9 28 107 11 8 12 4] [1 3 16 8 9 27 249 10 12 5] [8 3 13 28 37 14 10 233 16 18] [22 10 4 44 28 12 8 3 273 0] [8 3 6 5 15 6 9 10 8 113]]
4	0.7275722250380132	0.673470767064084	0.9464959784833111	[0.66666667 0.58781362 0.8 0.93487395 0.672346 0.50909091 0.80334728 0.69095477 0.65311653 0.47445255]	[0.61458333 0.76995305 0.74787535 0.86914062 0.77259036 0.4137931 0.59076923 0.66585956 0.62435233 0.75144509]	[0.6395664 0.66666667 0.77306003 0.90080972 0.71899089 0.45652174 0.68085106 0.67817509 0.6384106 0.58165548]	[192 213	[[118 4 0 3 12 15 2 6 22 10] [5 164 7 0 26 4 0 2 3 2] [4 20 264 3 20 5 5 18 7 7] [7 23 12 890 13 5 11 22 38 3] [7 20 10 10 513 13 4 28 16 43] [11 6 5 3 42 84 12 9 18 13] [1 9 20 11 24 17 192 16 14 21] [6 7 6 9 49 12 7 275 6 36] [18 25 5 20 52 6 4 6 241 9] [0 1 1 3 12 4 2 16 4 130]]

准确率都在0.7几，并不算一个很高的数字，说明模型能力仍然有待提升。F1比较低，少数类性能远低于Accuracy，说明有些类错判或者错漏现象比较严重。观察precision，可以看到5,6,8,9这四个索引对应的precision比较低，这也可以从confusion matrix看出。索引为5,6,8,9的四列，非对角线所占的比例也比较大，说明模型更容易把不是这几类的归为这几类，错判率比较高。Clothes（索引3）的Precision，Recall，F1都接近0.9，这一类的样本区分度最高，错判和错漏都很少；Metal（索引5）的Precision，Recall，F1都只有0.4几，False Negative和False Positive都比较严重。

以第一个Fold的Confusion Matrix的索引第五行为例，可以看到索引4和索引6处的值比较高，说明Metal最被容易错判为Paper和Glass，这说明模型容易将金属和纸张、玻璃的特征相混淆。

[11 3 13 4 31 85 30 9 10 17]

类似的也可以发现Carboard最被常错判为Paper（Confusion Matrix索引第二行）。

总体来看，模型整体分类性能尚可，多次实验下平均 Accuracy ≈ 72%（F1-macro ≈ 66%），说明模型学到了大部分类别的主要特征，对多数类能给出正确判断。对于那些样本量比较大的类别，模型的表现比较稳定。例如 Clothes（类 3）、Cardboard（类 2）、Paper（类 6）等，Precision/Recall/F1 都在 0.74–0.90 区间，说明网络在样本丰富、纹理明显的类上表现尤为可靠。平均 ROC-AUC ≈ 94.6%，表明模型输出的概率排序能力很强，即使不使用固定阈值，模型仍能较好地区分正负样本，为后续可调阈值或二次筛选提供了空间。利用K折交叉验证，也能发现模型的训练过程较为稳定。5 折交叉验证中各折 Accuracy 方差很小（≈ 0.9%），说明模型对数据子集划分的敏感性低，泛化性能较一致。

模型的不足也很明显。对于少数类别的分类，模型的性能较低。Metal的F1 ≈ 0.42，Recall ≈ 0.40，Precision ≈ 0.46，远低于整体水平，说明网络对该类特征提取不足，且容易出现误报和漏检。观察网络的结构，其复杂程度也许不足以捕捉足够精细的纹理特征。三层卷积 + 两层全连接的结构对某些细节特征（如光泽、反光面）提取不够，导致纹理相似的类别（如 Metal vs Glass、Cardboard vs Paper）易混淆。

模型改进

数据增强

在transform里面增加选项，使得输入数据随机裁剪，缩放，模拟目标在不同尺度和不同位置出现的情况，增强模型对目标距离和构图的鲁棒性；随机水平翻转，使模型不再“记住”左右的固定特征，增加对左右对称的泛化能力；颜色扰动，模拟不同光照、色温、相机设置等对颜色的影响，让模型对色彩变化更鲁棒；随机灰度化，让模型在部分情况下不依赖颜色信息，而更多关注形状和纹理特征。

```
transform = transforms.Compose([
    transforms.RandomResizedCrop(size=384, scale=(0.8, 1.0), ratio=(3/4, 4/3)),
    transforms.RandomHorizontalFlip(p=0.5),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.02),
    transforms.RandomGrayscale(p=0.1),
    transforms.ToTensor(),
    transforms.Normalize(mean=mean, std=std),
])
```

仅仅进行数据增强处理之后，模型的性能得到了一定的提升，Accuracy从72%上升到74%，F1也上升到69%，虽然仍然有提升空间，但是也说明数据增强对模型性能的提升是有帮助的，让模型能够不受数据本身拍摄方式，光照条件等等影响，从而更容易去捕捉到真实的数据特征。Precision和Recall没有太大的改观，说明类别之间的混淆情况仍然存在，实验结果也表明这种问题也许是不能通过数据增强来解决的。

模型结构改进

原先的模型设计中，并没有加入诸如dropout、batch normalization这样的模块，可能导致网络性能不够，收敛不够快，所以我另外设计了一个CNN网络，加入了这些模块，仍然保持三个卷积层和两个线性层这样的结构，只不过每一个卷积块变成了这样一个较为完整的模块：

```
self.block1 = nn.Sequential(
    nn.Conv2d(C,  32, kernel_size=3, padding=1),
    nn.BatchNorm2d(32),
    nn.ReLU(inplace=True),
    nn.MaxPool2d(2),
    nn.Dropout2d(dropout_conv)
)
```

我希望这样的设计能够改善训练的稳定性、加速收敛，并在一定程度上起到正则化、减缓过拟合的作用。在相对于baseline只改进了模型结构的时候，实验结果也有一定的提升，Accuracy提升到了75%，F1提升到了70%。相比之下，模型的改进相对于数据的增强对模型性能的提升更有优势。对于类别间的混淆问题来说，结构改进仍然不是解决这个问题的首选。

问题及展望

1. 类别混淆的问题在我的多次改进之后都没有得到很好的改善，尤其是索引5 Metal，Precision和Recall一直都不高。这是一个可以后续进行改进的地方。可以通过对Metal类增加权重，额外增强Metal类的数据来改善。
2. 在运行python文件的时候，我采用了parser的方法，便于直接在bash脚本中修改所需的参数，这样的写法是我在科研中阅读别人的代码接触到的，应用到自己写的代码之中，让代码更加简洁，容易扩展。
3. python中有很多很方便的函数，调用他们可以让我的代码写的更加简洁。例如数据的预处理，通过transform就可以直接调整，非常方便我进行后续的改进。
4. 这是我第一次编写全流程的深度学习代码，里面有许多小地方需要注意。例如device的问题，中间报错了好几次不在同一个device上面，把所有应该加上`to(device)`的地方都加上代码才能正常运行。这种需要在CPU和GPU进行数据交换的代码，以及各种地方需要倒梯度和不需要倒梯度的代码，都需要额外注意，才能写出正确的深度学习代码。