框架模型

为了方便看结构我把每个功能单列为一个 python 文件放在了 Structure1 文件夹下 - custom_model：对本题网络的搭建实现，计算 flop 和参数量（已被注释）

- datasets：读取数据集的方法，对数据集的通道，大小进行调整统一，实现了实验数据集和验证数据集的分离
- params：超参数控制台，存放了所有超参数便于后续调参，也存放了文件路径
- run：训练模型的主体代码
- train_valid_split：单独的一块代码，在开始 run 之前要单独运行一次，目的是把一个完整的数据集划分成一个 train 和一个 valid
- log：存放 log 模型和最后画图的 function
- output 文件夹：存放实验中保存的训练模型和训练日志

Train-valid split

本题数据集没有划分 train 和 valid, 所以需要自己划分，这个程序就是从原本的数据集中随机抽 80% 作为训练集，剩下 20% 作为验证集，单独写出来这一个程序是为了方便后边 Task2，Task3 都可以用。

Custom-Model

本体已经给出了网络的图，按着图一行一行打就行，为了避免多次重复操作，残差的那一个小 block 我单独拎出来写了一个 class，在搭建完整网络的时候直接插入就好，在这个网络后边有一个被注释掉的小块代码

是**计算网络 flop 和 params** 用的，这里我调了 thop 包，为了保证代码运行过程连贯这一步的结果我单独放在这个报告里。

Datasets

这一个文件我写的有点长了，操作有些地方也比较不清晰，但原理就是对图片进行统一 resize 到 112x112，如果不是 RGB 三通道，那就调整到三通道

然后一个一个把文件和 label 从文件夹里读出来放在 files 和 labels 列表里方便后续读取，同时把 Train 和 Valid 数据集分开读取。

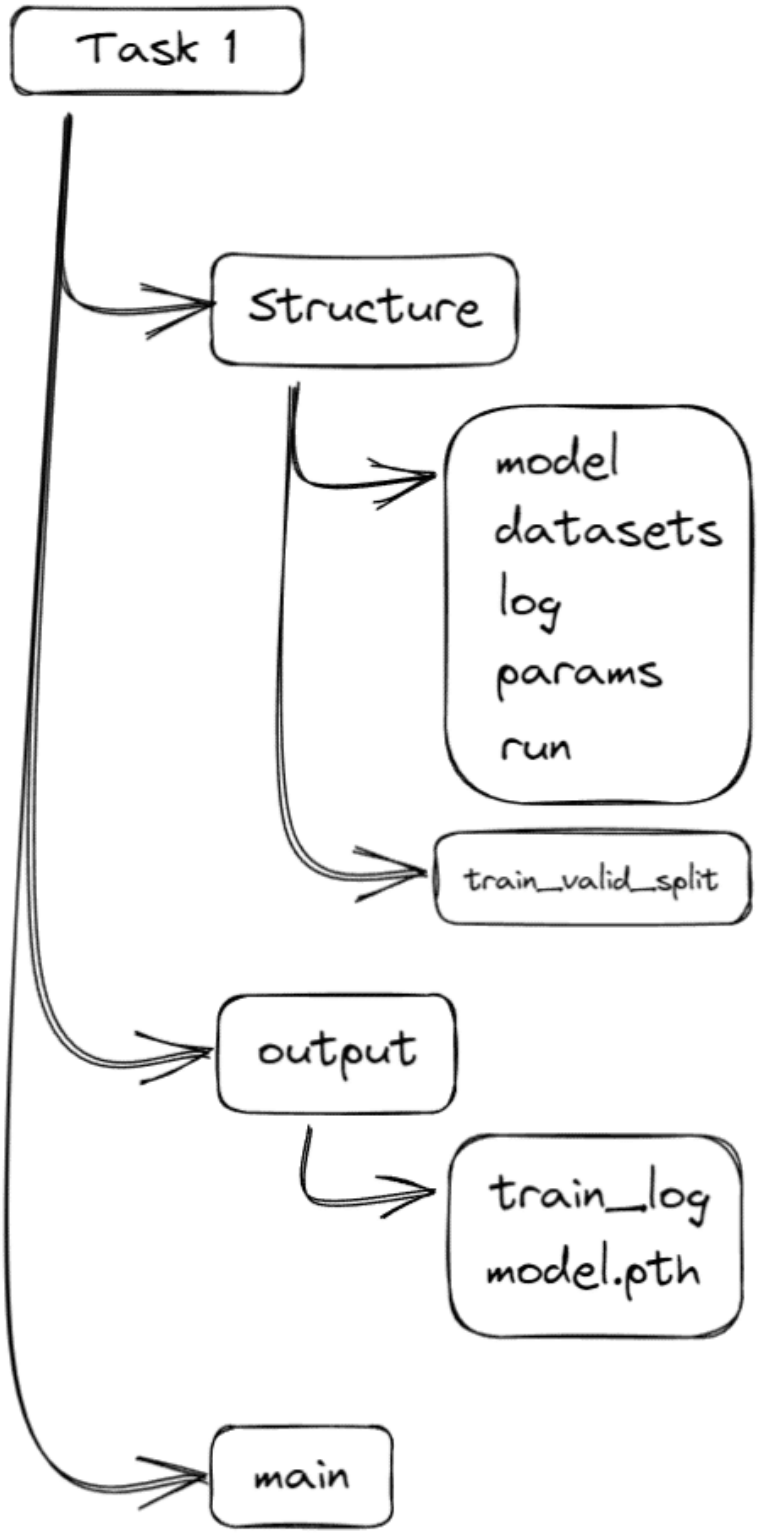

Figure 1: Structure|400

```
84  ## 计算flop和参数量
85  # if __name__ == '__main__':
86  #     net = Task1Net(3)
87  #     input = torch.rand(126, 3, 112, 112)
88  #     flops, params = profile(net, (input, ))
89  #     print('flops:', flops, 'params:', params)
90
```

Figure 2: Counting params and flops

```
Python 3.10.8 | packaged by conda-forge | (main, Nov 24 2022, 14:07:00) [MSC v.1916 64 bit (AMD64)]
[INFO] Register count_convNd() for <class 'torch.nn.modules.conv.Conv2d'>.
[INFO] Register count_normalization() for <class 'torch.nn.modules.batchnorm.BatchNorm2d'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.activation.ReLU'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.container.Sequential'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.pooling.MaxPool2d'>.
[INFO] Register count_avgpool() for <class 'torch.nn.modules.pooling.AvgPool2d'>.
[INFO] Register count_linear() for <class 'torch.nn.modules.linear.Linear'>.
[INFO] Register zero_ops() for <class 'torch.nn.modules.dropout.Dropout'>.
flops: 22142421504.0 params: 1108358.0
```

Figure 3: output

```
self.Trans = torchvision.transforms.Compose([
    torchvision.transforms.Resize([PARAMS.img_size, PARAMS.img_size]),
    torchvision.transforms.ToTensor()
])
```

Figure 4: Pic Resize

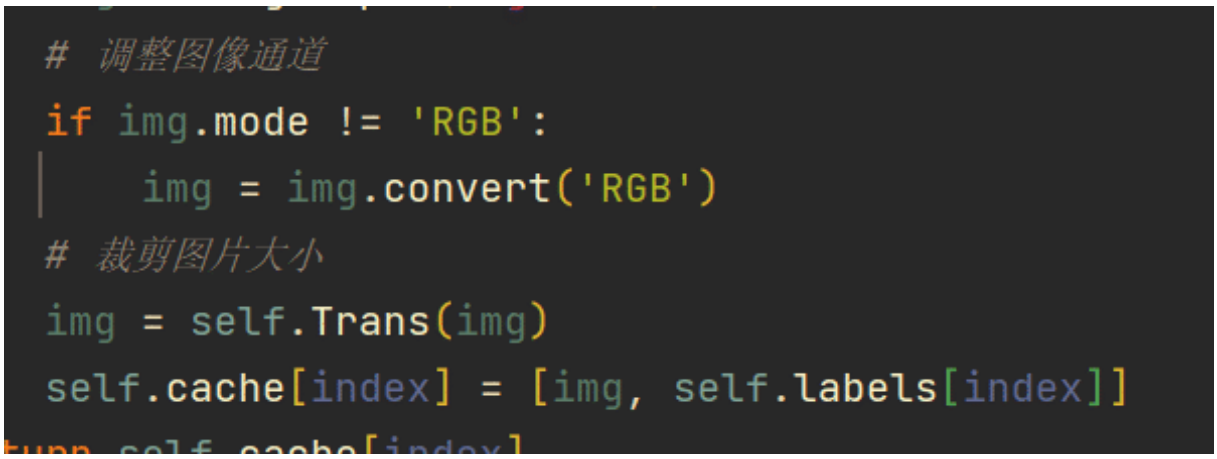

Figure 5: Convert Channels

Log

也没啥可说的，先有一个模板，然后分别输出到日志文件 (./output/log.txt) 里和终端里让用户看到目前的情况，在日志函数后还有一个画图的函数，把日志文件里的数据用正则表达式一个一个读取到列表里，最后统一画图，用正则表达式来实现画图的操作是因为我觉得最后从一个 txt 文件里读数据再画图会快很多。

Run

训练模型的主体部分，把数据读取出来，清空一下日志文件（因为写的时候我是用 append 方法写进去的不会自己删除），然后调用规定好的超参数开始训练，把网络，优化器，损失函数都规定好以后就可以扔 DEVICE 里开始训练了。

放一下结果图

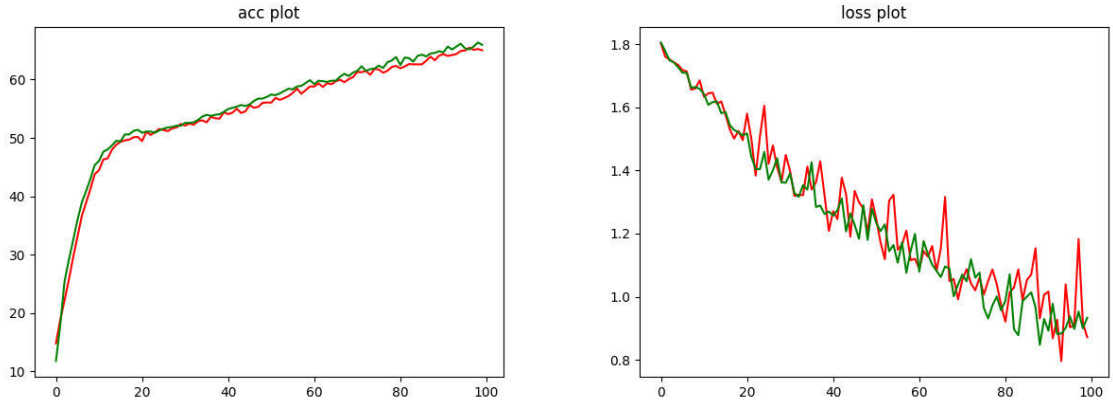

Figure 6: Results