

# Homework 6

林宇辰 2200013211

December 2023

## 1 Data augmentation

我使用了 pytorch 内置的数据增强函数。在创建数据集的同时，使用随机水平翻转和随机旋转进行数据增强。

## 2 VGG

我实现了经典的 VGG 模型，在每个卷积层之后我都加上了 batchnorm 层进行批次归一化。在分类层中，我在每个线性层之间加上了 dropout 来防止过拟合。由于显存的限制，在这个任务之中，我限制图片的大小为  $60 \times 60$ ，并限制每个批次的大小为 32。经过 10 个 epoch 的训练，本模型可以达到 85% 左右的准确率。可视化结果如图 1 所示。

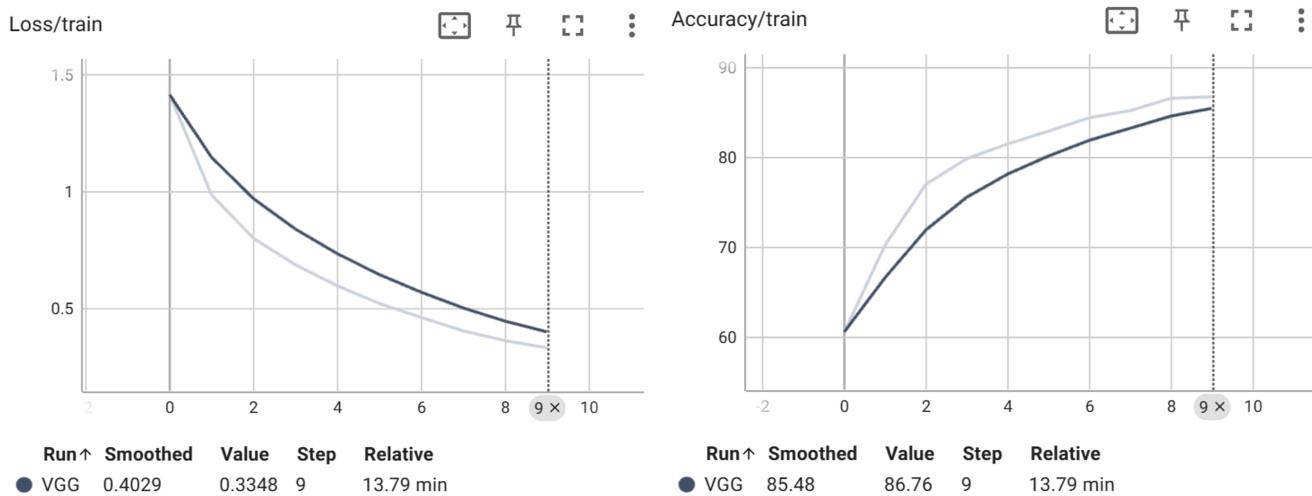


图 1: VGG

## 3 ResNet

在这个任务中，我先实现了残差连接的块，然后将他们搭建起来。如果输入输出维度不一样，则用  $1 \times 1$  卷一遍再相加。对于这个模型，我做了两次实验，第一次输入的图片维度为 60，最后的分类器只是一层线性层，训练了 20 个 epoch，可视化效果如图 2 所示，可以发现，数据很快就过拟合了，成功率并不是非常高。第二次，我输入的维度为 224，并效仿 VGG 的分类器作为分类器，这次效果好了不少，经过 10 个 epoch 的训练可以达到 85% 左右的成功率，可视化效果如图 3。接着我又做了第三次实验，让输入的维度为 60，使用 VGG 的分类器（有一个隐层），发现第三次实验和第一次实验的情况类似，可视化结果见图 4。我认为出现这种情况是模型发生

了过拟合，当输入是  $60 \times 60$  的时候，输入的信息过少，而模型的拟合能力又很强，会出现过拟合。这就是为什么利用残差的网络甚至比不利用残差的网络（VGG）表现来的差的原因。然后我又做了几次实验，发现输入数据维度  $224 \times 224$  的时候，即使使用一个 Linear 作为分类，也可以达到很好的效果，可视化如图 5。（也就是 ResNet18 用的结构）并且模型收敛的速度还要更快一些，说明卷积层就已经很好地提取了特征，不需要再引入复杂度了。

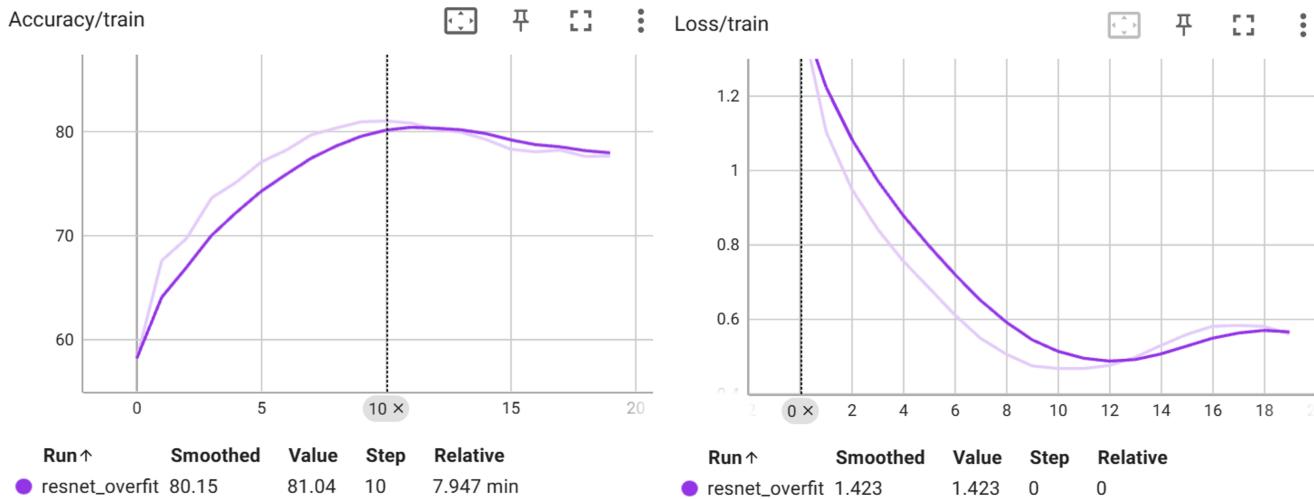


图 2: ResNet 1

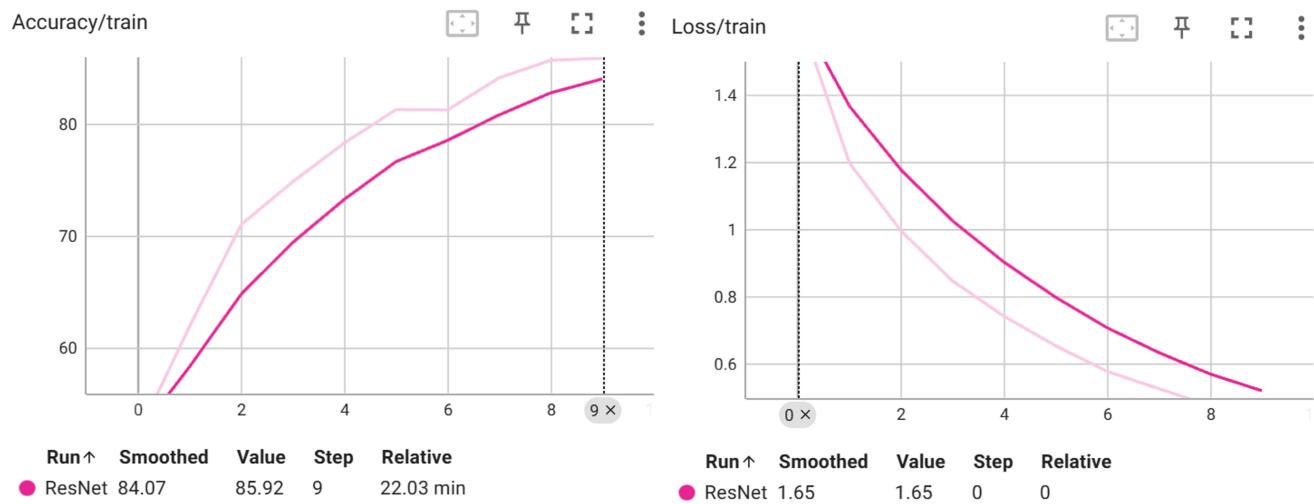


图 3: ResNet 2

## 4 ResNext

在上个任务的基础上改一改残差块就可以。在这个任务中我做了很多次实验，我发现使用我的默认参数，模型无法达到前两个模型的预测效果，我在学习率 0.001，每个批次使用 32 个样本的 ResNext 训练可视化效果如图 6 所示，可以发现，经过多轮训练，模型只能达到 70% 左右的预测成功率。在第 0 次实验中，我花了一个多小时训练了一个含 6 个残差块的模型，在第 1 次实验中，我尝试简化模型，只使用 3 个残差块，发现效果仍然不佳。我认为这是由于新模型比原来的模型要更复杂，由于显卡的限制，我的训练次数还不够多，发生了欠拟合。所以我调节学习率为 0.01，每个批次使用 64 个样本进行实验，这次效果果然好了很多。经过 10 个 epoch 的训练，可以达到 81% 左右的准确率，可视化效果如图 7 所示。

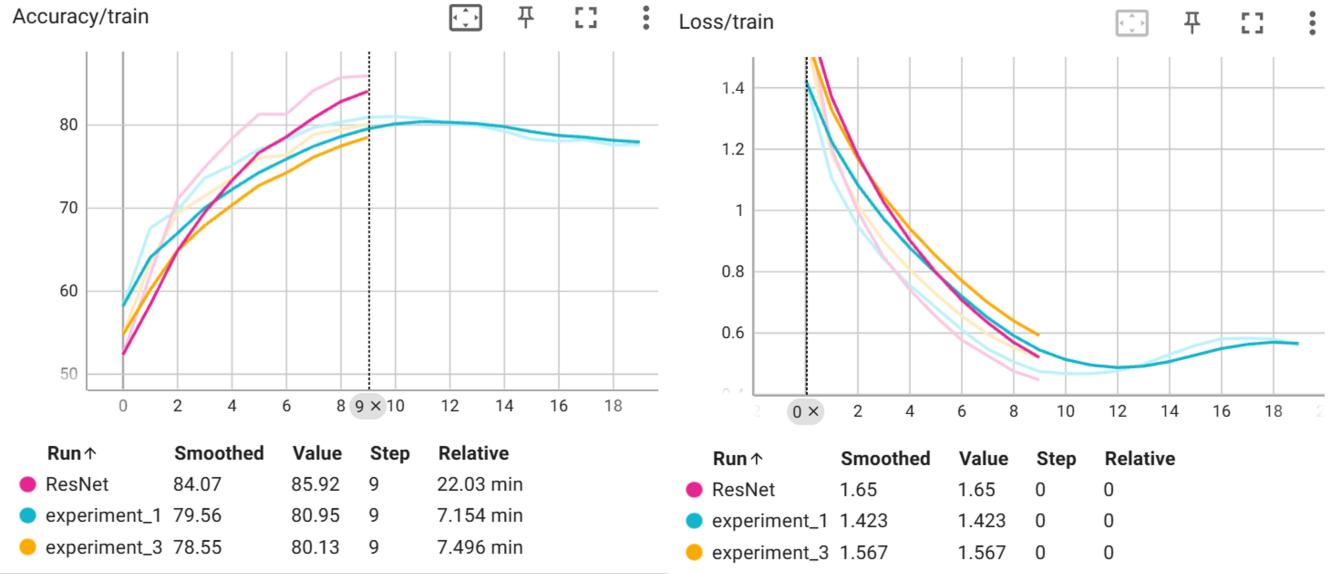


图 4: ResNet 3

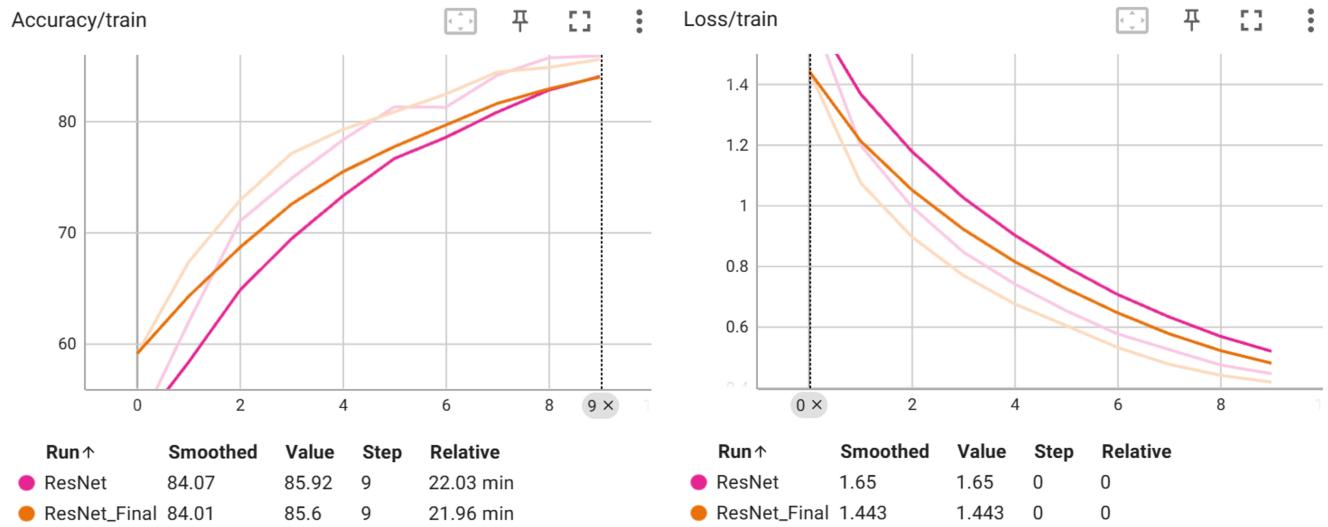


图 5: ResNet 4

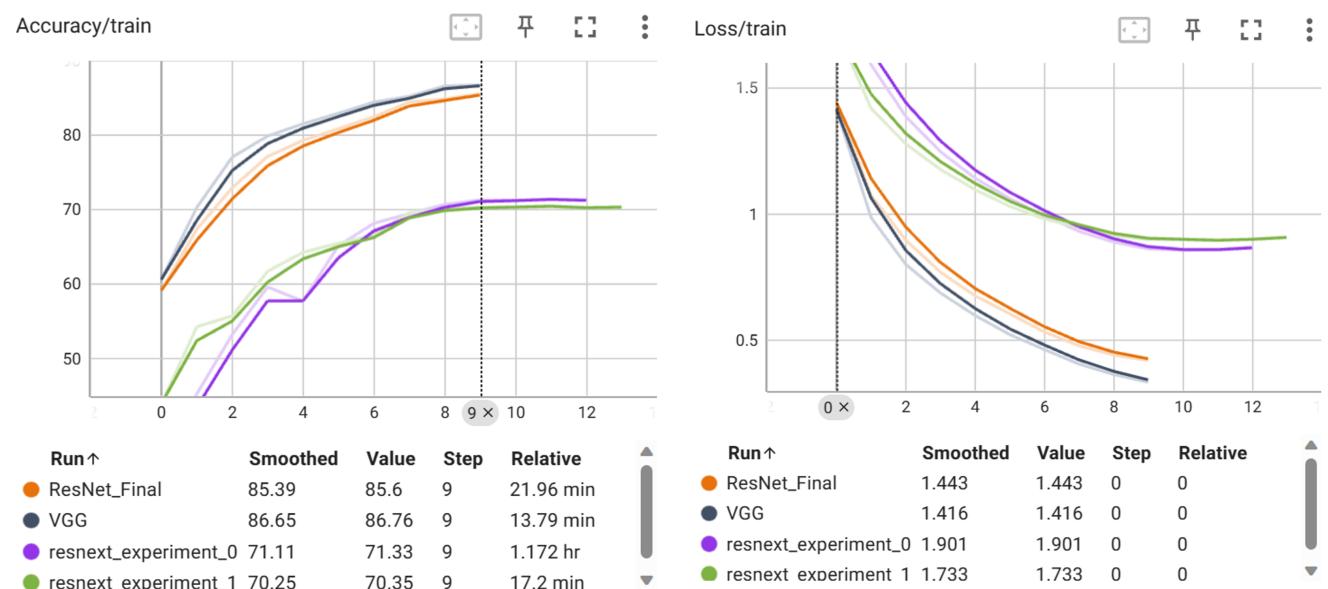


图 6: ResNeXt Experiment

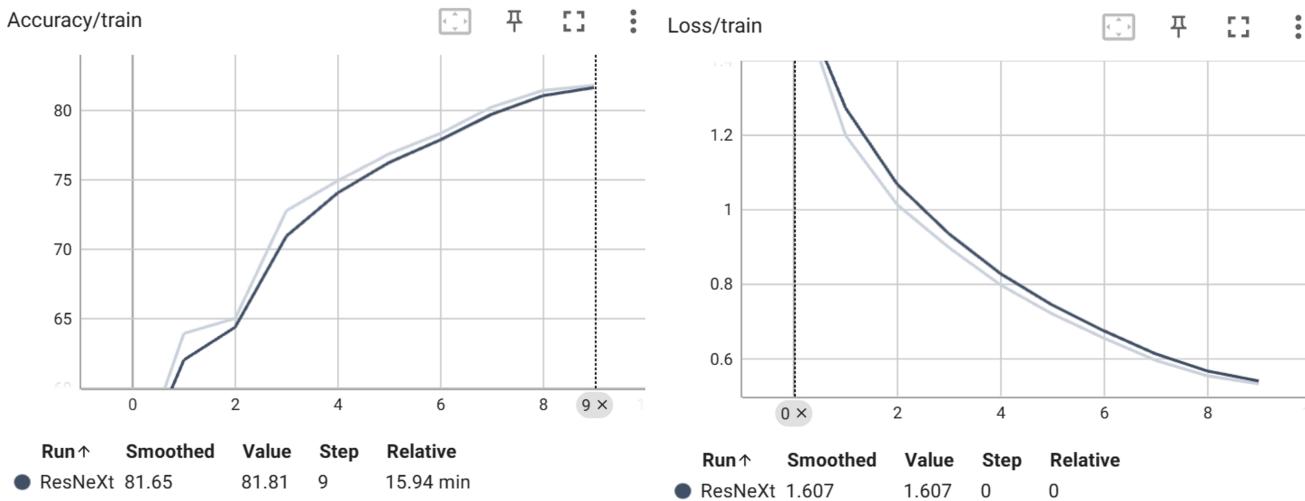


图 7: ResNeXt

## 5 Train and Test

复用作业 5 的代码。我加入了进度条，让训练过程的可视化变得优雅了一点。最后我们来对比一下三个模型的表现，如图 8 所示。我发现宏观来看 vgg 的效果要好于 resnet，resnet 又好于 resnext。但是这种情况本来不该出现（应该倒过来），我觉得这是因为后面两个模型相较于 vgg 要更复杂一些，收敛的速度会比较慢，需要更多轮次的训练。并且残差连接网络的优势在于模型可以拥有很大的深度，但是考虑到训练的效率，我并没有编写特别深层的网络。同时，后两者可能也需要更精细的超参选择。但是，按照我的实现，三个模型在合理的参数调整之下，都可以在 10 个 epoch 下达到 80% 以上的准确率。训练的命令如下：

```
python main.py --model vgg
python main.py --model resnet --size 224
python main.py --model resnext --size 224 --learning_rate 0.01 --batch_size 64
```

对于测试，我提供了一个使用 resnet，输入图片维度为 224，训练 10 轮的模型，预期可以达到 85% 左右的准确率，可以通过以下代码调用：

```
python main.py --run test --model resnet --size 224
```

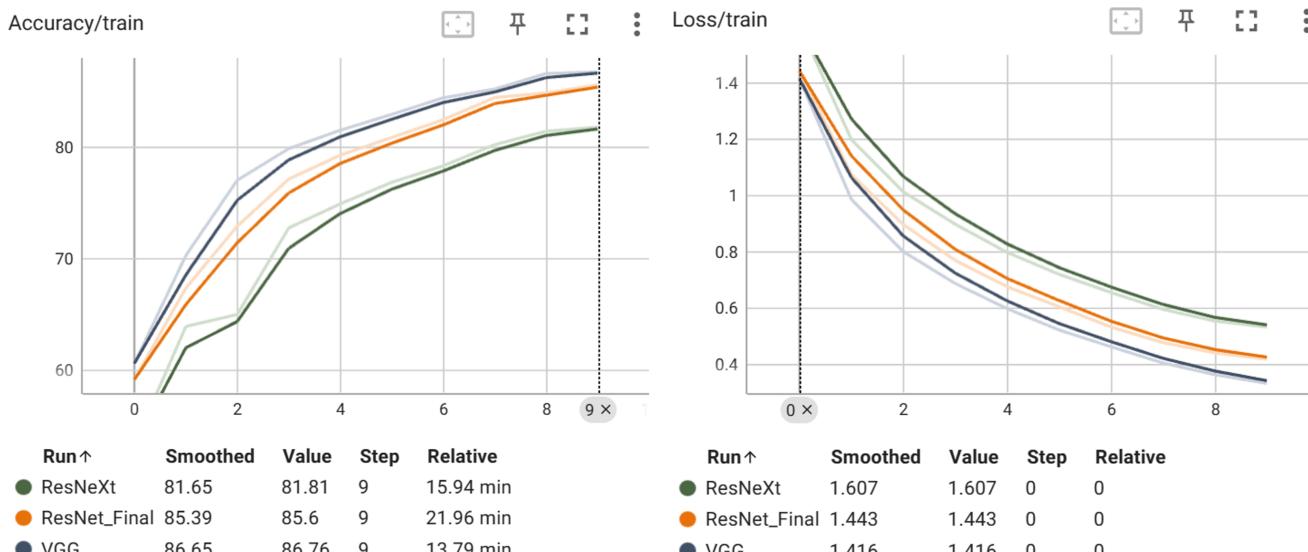


图 8: Models Comparison