人工智能基础 大作业二

图像分类

班级：自 83

姓名：门恺文

学号：2018011501

2020 年 12 月 13 日

0. 说明

由于本项目总体文件过大，所以上交的压缩文件中只包含了源代码，报告，测试结果，和必要的说明 README.txt，没有包含代码运行过程中记录结果的文件夹和得到最终结果需要的权重。如果需要复现结果可以从此云盘链接下载整个项目：https://cloud.tsinghua.edu.cn/d/c58fc39a350b47b99d57/

1. 任务综述

本次大作业中我的选题是图像分类。采用 pytorch 神经网络框架，重写数据集类，并将带标签的数据划分为训练集、验证集、自测集，将不带标签的数据作为最终的测试集。使用了几种自己实现的神经网络进行训练，并且使用了迁移学习的方法，比较最终的结果择优选择。最后通过不同类别之间正确率的不同，找到神经网络容易识别对的和容易识别错的类别，重新进行训练，赋予准确率低的类别更高的损失，以达到更高的准确率。最后，由于单一的网络总有缺陷，得到的结果存在瓶颈，所以采用多种网络得到的结果进行投票，得到最终概率最大的结果。这样在自测集上得到了更好的测试效果。最终自测集上粗分类最高正确率 94.25%，细分类最高 88.45%。

2. 数据预处理

2.1 重写 dataset

数据预处理过程中重写了 dataset，加入__len__和__getitem__函数，便于后续程序的使用。

2.2 处理训练集和测试集

使用两次 train_test_split 划分出训练集、验证集、自测集，每次测试集的比重都是 0.2，所以最后使用的训练集有 40000 张图片，验证集有 8000 张图片，自测集有 2000 张图片。coarse 和 fine 的数据集都以这个办法划分。划分后使用每个数据集第 0 维（既图片数）*3*32*32 调整数据集的形状，使得数据集适合神经网络模型。由于 pytorch 中数据需要以 tensor 数据类型才能顺利训练，所以将每个数据集都转换成 tensor 类型。之后定义两种 transform，一种针对训练集，可以进行大小变换、裁剪、翻转、正则化及自己加入的 cutout，即随机剪掉图像中的某

一区域；另一种针对测试和验证集，改变大小并正则化，不进行其他的图像变换。之后构建数据集即可。

最终的测试集的构建只需要加入测试集数据和测试集的变换即可，与训练集、验证集、自测集相比的区别是没有标签，其他的与自测集类似。

3. 自行实现的模型和优化策略

3.1 网络的选择

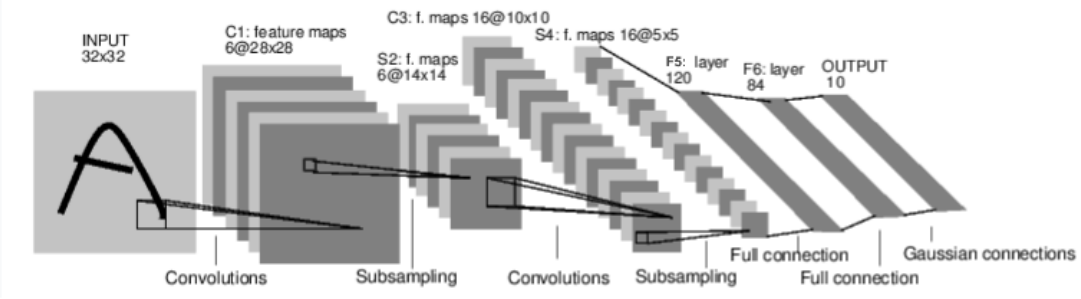
pytorch 官网上的指导很完善，并且提供了众多图像分类中可用的深度学习网络的论文和源码。参照官网的指导并结合数据集的特点，自行实现了 Lenet，简化的 VGG16，resnet，shake-shake，验证集上正确率分别如下：

说明：由于最开始寻找最佳网络时是在 20 分类上做测试，所以有些网络没有 100 分类的结果，另外，在方法尝试的过程中由于资源和时间有限，所以部分模型的最高正确率可能会出现不合理之处（比如 Resnet34 比 Resnet18 正确率低很多），后续会具体分析提升正确率的方法，此处表格仅供参考。

Network	Coarse Accuracy	Fine Accuracy
Lenet	28.81%	None
VGG	69.8%	None
ResNet18	83.86%	75.61%
ResNet34	77.74%	None
ResNet50	77.76%	None
ResNet101	78.07%	None
ResNet152	77.93%	None
ResNext50	77.36%	None
ResNext101	83.53%	None
wide_ResNet50	78.76%	None
wide_ResNet101	78.71%	None
shake-shake	66.85%	None

3.1.1 Lenet

Lenet 网络结构如下[1]，Lenet 是最基础最简单的网络，只有七层，但是也可以达到不错的效果。这是我最开始尝试的网络，在最初始的不够完善的设置下，20 分类可以达到 28.8%的正确率。



3.1.2 VGG

这里我的 VGG 是简化的 vgg-16，结构如下图[2]。可见与 Lenet 相比，VGG 结构更加复杂，层数更深，并且使用了大量 3*3 卷积核提高网络性能。VGG 的正确率可以达到 69.8%，也说明更深、更复杂的网络层数确实可以提取更多有效特征，达到更好的分类效果。

ConvNet Configuration					
A	A-LRN	B	C	D	E
11 weight layers	11 weight layers	13 weight layers	16 weight layers	16 weight layers	19 weight layers
input (224 × 224 RGB image)					
conv3-64	conv3-64 **LRN**	conv3-64 **conv3-64**	conv3-64 conv3-64	conv3-64 conv3-64	conv3-64 conv3-64
maxpool					
conv3-128	conv3-128	conv3-128 **conv3-128**	conv3-128 conv3-128	conv3-128 conv3-128	conv3-128 conv3-128
maxpool					
conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256	conv3-256 conv3-256 **conv1-256**	conv3-256 conv3-256 **conv3-256**	conv3-256 conv3-256 conv3-256 **conv3-256**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512	conv3-512 conv3-512 **conv1-512**	conv3-512 conv3-512 **conv3-512**	conv3-512 conv3-512 conv3-512 **conv3-512**
maxpool					
FC-4096					
FC-4096					
FC-1000					
soft-max					

3.1.3 ResNet

[1] https://pytorch.apachecn.org/docs/1.0/blitz_neural_networks_tutorial.html
[2] https://arxiv.org/pdf/1409.1556.pdf

ResNet 网络结构如下图[3]。这里我参照官网实现了 ResNet18, ResNet34, ResNet50, ResNet101, ResNet152。比较关键的点在于残差块的实现和对于网络的调整。

定义两种残差块，一种用于层数较少的网络，如 resnet18，resnet34，一种用于层数较多的网络，如 resnet50，resnet101，resnet152。残差块通过建立与卷积层并行的短连接解决随着网络层数加深出现的层级消失问题，大大提高了模型的性能，减少了对模型层数加深的限制，是目前准确率最高的网络之一。

layer name	output size	18-layer	34-layer	50-layer	101-layer	152-layer
conv1	112×112	7×7, 64, stride 2				
		3×3 max pool, stride 2				
conv2_x	56×56	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 64 \\ 3\times3, 64 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3_x	28×28	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 128 \\ 3\times3, 128 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times8$
conv4_x	14×14	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 256 \\ 3\times3, 256 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times23$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times36$
conv5_x	7×7	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times2$	$\begin{bmatrix} 3\times3, 512 \\ 3\times3, 512 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	average pool, 1000-d fc, softmax				
FLOPs		1.8×10^9	3.6×10^9	3.8×10^9	7.6×10^9	11.3×10^9

从图中也可以看出来，resnet 的第一层在用 7*7 的卷积核提取图片特征的同时对图片的长宽进行了缩小四倍的处理，这是因为 resnet 原本训练的数据的大小是 224*224，所以应当缩小一些去除冗余的特征，提取出主要的特征。但是这对于 32*32 的数据效果就不太合适，因为本来已经很小了，再缩小就只有 8*8 的大小了，容易损失重要特征，另外，7*7 的卷积核对于 32*32 的图片来说也太大了，所以在自行实现的时候改写了 resnet 第一层，将卷积核改成 3*3 大小，并且将 stride 设为 1。在优化的过程中也尝试了在最后增加更多的全连接层，但是这个没有明显的效果。以 resnet18 为例，改进过程中的验证集正确率提升如下（20 分类）：

改进	无改进	卷积核 3*3	除去缩小	增加 fc 层
正确率	56.75%	69.25%	76.15%	74.16%

由于在实验中发现增加 fc 层不太能改善网络，所以最后将增加 fc 层部分注释了，在源代码中仍然可以看到此部分改进。

实验过程中发现在充分训练的情况下，增加网络层数一般可以略微提高准确率，但是相差不多。不同网络的验证集准确率如下（20 分类）：

[3] https://arxiv.org/pdf/1512.03385.pdf

网络	Resnet18	Resnet34	Resnet50	Resnet101	Resnet152
正确率	76.15%	77.74%	77.76%	78.08%	77.93%

3.1.4 ResNext

ResNext 网络结构如下图[4]，将 ResNet 的单个卷积改成了多支路的卷积，从而进一步提高精度。

stage	output	ResNet-50	**ResNeXt-50 (32×4d)**
conv1	112×112	7×7, 64, stride 2	7×7, 64, stride 2
conv2	56×56	3×3 max pool, stride 2	3×3 max pool, stride 2
		$\begin{bmatrix} 1\times1, 64 \\ 3\times3, 64 \\ 1\times1, 256 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128, C=32 \\ 1\times1, 256 \end{bmatrix}\times3$
conv3	28×28	$\begin{bmatrix} 1\times1, 128 \\ 3\times3, 128 \\ 1\times1, 512 \end{bmatrix}\times4$	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256, C=32 \\ 1\times1, 512 \end{bmatrix}\times4$
conv4	14×14	$\begin{bmatrix} 1\times1, 256 \\ 3\times3, 256 \\ 1\times1, 1024 \end{bmatrix}\times6$	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512, C=32 \\ 1\times1, 1024 \end{bmatrix}\times6$
conv5	7×7	$\begin{bmatrix} 1\times1, 512 \\ 3\times3, 512 \\ 1\times1, 2048 \end{bmatrix}\times3$	$\begin{bmatrix} 1\times1, 1024 \\ 3\times3, 1024, C=32 \\ 1\times1, 2048 \end{bmatrix}\times3$
	1×1	global average pool 1000-d fc, softmax	global average pool 1000-d fc, softmax
# params.		**25.5**×10^6	**25.0**×10^6
FLOPs		**4.1**×10^9	**4.2**×10^9

本次大作业中我实现了 ResNext50, ResNext101，与 ResNet50, ResNet101 在验证集上的正确率对比如下表：

网络	ResNet50	ResNext50	ResNet101	ResNext101
正确率	77.76%	77.36%	78.08%	78.87%

3.1.5 wide_ResNet

wide_ResNet 的网络结构如下图[5]，与 ResNet 相比，wide_ResNet 增加了并行的残差层，拓宽了每一层的宽度，从而优化了 ResNet，在网络结构不深的情况下取得更好的效果。

[4] https://arxiv.org/pdf/1611.05431.pdf
[5] https://arxiv.org/pdf/1605.07146.pdf

group name	output size	block type = $B(3,3)$
conv1	32×32	[3×3, 16]
conv2	32×32	$\begin{bmatrix} 3\times3, 16\times k \\ 3\times3, 16\times k \end{bmatrix}\times N$
conv3	16×16	$\begin{bmatrix} 3\times3, 32\times k \\ 3\times3, 32\times k \end{bmatrix}\times N$
conv4	8×8	$\begin{bmatrix} 3\times3, 64\times k \\ 3\times3, 64\times k \end{bmatrix}\times N$
avg-pool	1×1	[8 × 8]

本次大作业中我实现了 wide_ResNet50, wide_ResNet101，与 ResNet50, ResNet101 在验证集上的正确率对比如下表：

网络	ResNet50	wide_ResNet50	ResNet101	wide_ResNet101
正确率	77.76%	78.76%	78.08%	78.71%

从以上结果可知，resnet 系列的准确率从 76.15-78.87%不等，差异比较微小，说明需要使用其他方法提升学习率。

3.1.6 shake-shake

这是我在查找提升 resnet 准确率的过程中找到的，主要思想在于将不同的卷积层的结果随机组合，达到更多样的结果[6]。shake-shake 网络对于 20 分类验证集的准确率为：

66.85%

看着自己一点一点好不容易提升的正确率一下子降了 10%，我后续的实验中没有再考虑 shake-shake。

3.2 优化器的选择

参照官网给出的 pytorch 优化器[7]，试验了不同参数的 Adam，SGD，RMSprop 这几种常用的优化器，验证集上正确率如下。由于做此实验时还没考虑优化策略，并且以查看效果为目的训练轮次比较少，所以此正确率仅供参考。

其中 Adam 的 betas 使用默认值，Adam2 的 betas=(0.9, 0.99)

优化器	SGD	Adam	Adam2	RMSprop
正确率	52.50%	70.15%	68.93%	69.06%

当时的能达到最优效果的是学习率为 1e-4 的 Adam，不过后来找到更好的优化策略后有所调整。

[6] https://arxiv.org/pdf/1705.07485.pdf
[7] https://pytorch.org/docs/stable/optim.html

3.3 数据增强的取舍

torchvision 中自带的 transforms 为训练集数据增强带来了很大的便利，但数据增强不是越多越好。我最开始的数据增强采用了水平翻转、随机缩放裁剪、随机旋转，之后又加入过改变颜色等方法，结果发现正确率降低了，也尝试加入正则化，因为查阅资料表明加入正则化可以加快收敛，但是没有很明显的效果。同时，由于看到查到的大部分博客采用了先扩大后裁剪，而我原有的是先缩放后裁剪，所以也尝试了另一种办法，发现先扩大后裁剪效果好一些。所以最终尝试下保留了随机水平翻转和先扩大后裁剪，以及正则化。同时，我也自己实现了一种数据增强方式，即 cutout[8]，即随机裁去图像中一部分面积，使用此方法正确率也可以有部分的提升。以 ResNet18 上的 20 分类为例，为节省时间仅训练 50epoch，各种数据增强达到的验证集正确率如下表。

数据增强	原本	过多增强	正则化	改变裁剪	加入 cutout
正确率	75.11%	62.94%	75.33%	76.36%	78.41%

之后进行充分的训练，20 分类和 100 分类在 ResNet18 上正确率分别为：

类别数	20	100
正确率	81.69%	70.03%

3.4 优化策略的调整

后来有幸看到了此 GitHub[9]，开始尝试优化策略的选择，进一步提升了正确率，分别尝试了自适应调整，cosLR 调整策略，还有阶梯式下降学习率，以 20 分类为例，验证集正确率如下：

其中 Multi1 调整 3 次学习率，Multi2 调整 4 次学习率。

策略	不调整	自适应	cosLR	Multi1	Multi2
正确率	81.69%	81.86%	52.61%	81.28%	83.14%

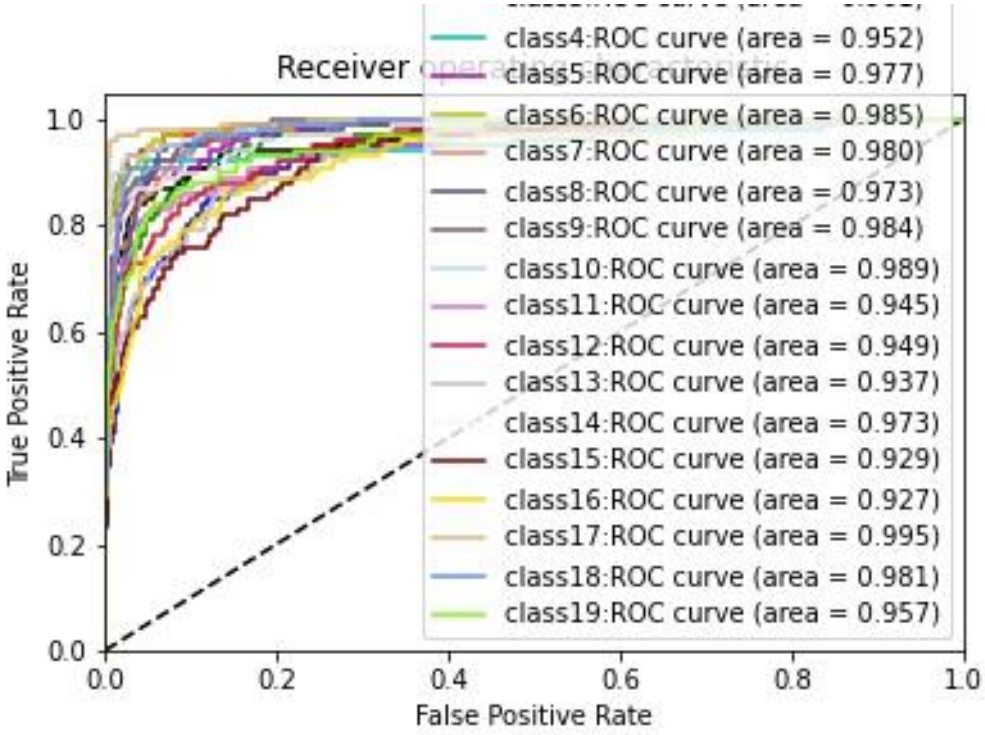
可见，第二种 MultiStep 更优，而自适应调整在训练过程中发现容易钻牛角尖，所以还是人为调控比较好，即使用第二种 MultiStep 策略。充分训练后，20 分类和 100 分类在 ResNet18 上正确率分别为：

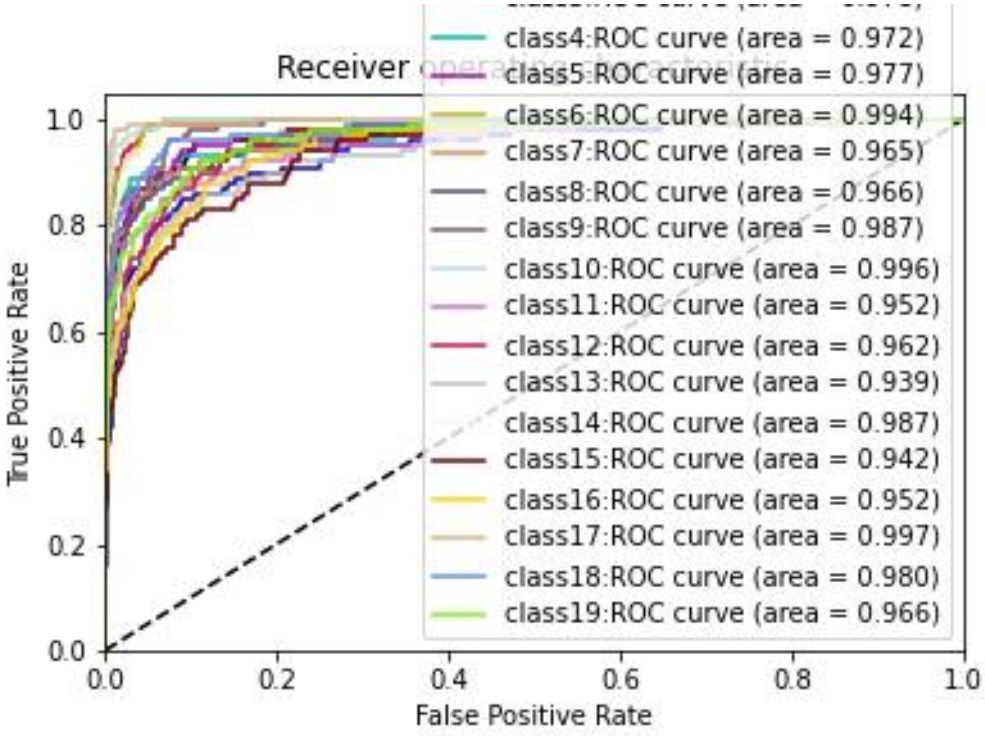
类别数	20	100
正确率	83.14%	75.61%

[8] https://arxiv.org/pdf/1708.04552.pdf
[9] https://github.com/weiaicunzai/pytorch-cifar100

3.4 结合自测结果 loss 函数的改进

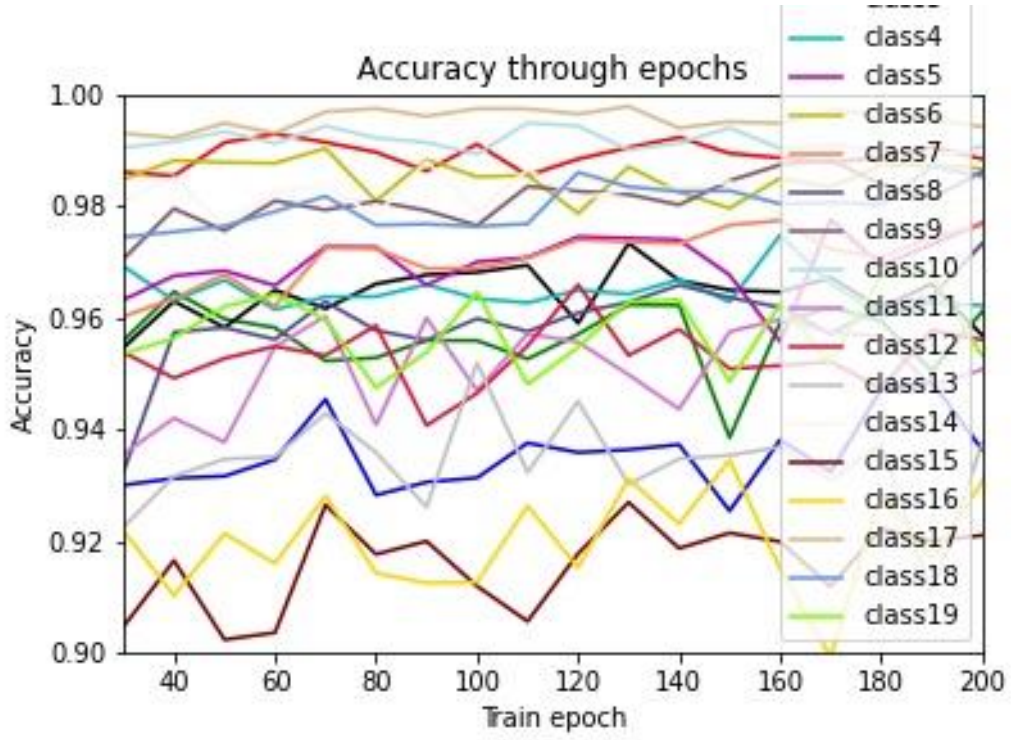
原本使用的损失函数即 CrossEntropyLoss，之后分析测试结果时，发现部分类别识别的很不好，以 20 分类为例，各个类别在 resnet18，resnet34 上预测后的 roc 曲线分别如下：

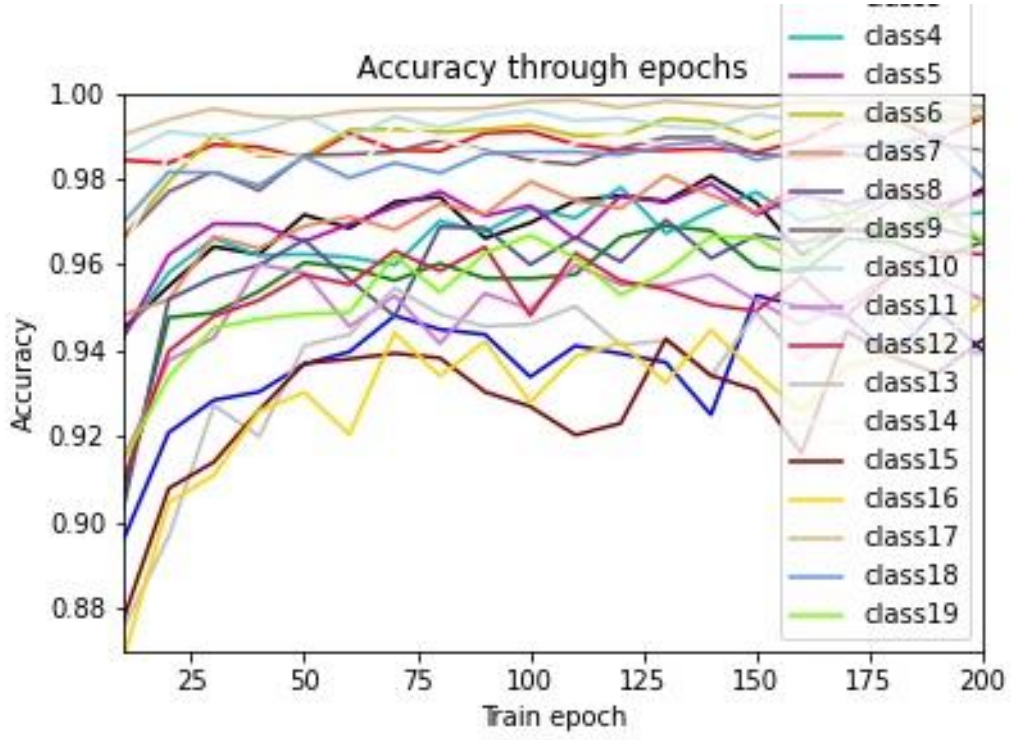




为了更加直观，做出不同类别随着 epoch 增加 auc 的变化如下，分别是 resnet18

和 resnet34：





从图像可以大致看出各个类别的效果好坏，但是由于类别有点多，不太方便看，所以转换成文字形式，即

对于 ResNet18：

Average AUC:

0.9357256578947369

0.9568844736842106

0.9897694736842105

0.9625602631578948

0.9647202631578947

0.9702863157894738

0.9854086842105264

0.967009605263158

0.9565901315789475

0.9824659210526315

0.9911269736842107

0.9503565789473682

0.9506311842105264

0.9320613157894737

0.9814513157894738

0.9187786842105264

0.9128303947368419

0.9956542105263159

0.9812436842105263

0.9558253947368422

Rank	AUC
17	0.9956542105263159
10	0.9911269736842107
2	0.9897694736842105
6	0.9854086842105264
9	0.9824659210526315
14	0.9814513157894738
18	0.9812436842105263
5	0.9702863157894738
7	0.967009605263158
4	0.9647202631578947

3 0.9625602631578948

1 0.9568844736842106

8 0.9565901315789475

19 0.9558253947368422

12 0.9506311842105264

11 0.9503565789473682

0 0.9357256578947369

13 0.9320613157894737

15 0.9187786842105264

16 0.9128303947368419

对于 ResNet34：

Average AUC:

0.937269736842105

0.9572076315789474

0.9883928947368421

0.9690401315789474

0.9676426315789473

0.9709761842105264

0.9894713157894737

0.9702255263157895

0.9601036842105263

0.9842502631578945

0.9929103947368422

0.9500268421052631

0.9529060526315789

0.9363186842105262

0.9860513157894737

0.9281709210526315

0.9297752631578946

0.996325394736842

0.9839459210526316

0.9564082894736841

Rank AUC

17 0.996325394736842

10 0.9929103947368422

6 0.9894713157894737

2 0.9883928947368421

14 0.9860513157894737

9 0.9842502631578945

18 0.9839459210526316

5 0.9709761842105264

7 0.9702255263157895

3 0.9690401315789474

4 0.9676426315789473

8 0.9601036842105263

1 0.9572076315789474

19 0.9564082894736841

12 0.9529060526315789

11 0.9500268421052631

0 0.937269736842105

13 0.9363186842105262

16 0.9297752631578946

15 0.9281709210526315

这样可以直观地看出两种网络对于不同类别识别的好坏，这样可以根据各个类别的 auc，对损失函数中识别较差的类别赋予更高的损失。加入权重后 resnet18 上 20 分类的验证集正确率如下：

是否加入权重	加入权重	未加权重
正确率	83.35%	83.14%

可见有轻微的提高。100 分类不太好加权重，考虑到提高效果不明显，100 分类

中没有加入权重。

最终自我实现的网络达到的 20 分类验证集最高正确率为 83.53%，100 分类验证集最高准确率为 73.74%。

4. 迁移学习

由于样本有限，训练难免会遇到瓶颈，所以总难有所突破。之后发现使用迁移学习，读取 torchvision 中自带的预训练权重，可以大大提高准确率。

4.1 更改配置

由于 torchvision 的预训练权重是在 224*224 的图像上产生的（inception_v3 除外，是 299*299），所以在图像变换中需要将图像 resize 到 224，其余的数据预处理都容易降低训练效果，所以只保留了随机水平翻转和正则化。另外，收敛也比较快，但是由于图像的增大训练变得困难，所以减少了 epoch，相应减少了学习率调整的间隔。各种网络的验证集正确率如下，由于实在是太慢，所以只试验了其中几种网络。

网络	20 分类	100 分类
resnet18	88.29%	81.25%
resnet34	89.63%	83.01%
resnet50	88.88%	84.03%
resnet101	88.85%	None
resnext50	90.35%	85.2%
resnext101	90.34%	None
wide_resnet50	None	85.58%
inception_v3	89.7%	None
densenet161	89.67%	None

4.2 使用 mixup

迁移学习中基本没用到数据增强，所以需要其他的方法提升准确率。mixup 是一种可以在训练过程中增加数据集不确定度的方法，公式如下[10]：

[10] https://arxiv.org/pdf/1710.09412.pdf

$$\tilde{x} = \lambda x_i + (1-\lambda)x_j, \quad \text{where } x_i, x_j \text{ are raw input vectors}$$
$$\tilde{y} = \lambda y_i + (1-\lambda)y_j, \quad \text{where } y_i, y_j \text{ are one-hot label encodings}$$

将不同的样本和对应的标签混合一下，可以得到新的样本，这个方法在迁移学习中可以获得比较好的效果。采用 mixup 后验证集准确率如下，依旧是训练了部分网络得到的结果。

网络	20 分类	100 分类
resnet34	90.78%	83.01%
resnext50	91.15%	84.82%

可见同原来相比有些许提升。

4.3 改用 LabelSmoothingCrossEntropy

在迁移学习的训练过程中观察到了严重的过拟合，为了解决此问题，我尝试使用 LabelSmoothingCrossEntropy，将独热码转变为数值，然后计算交叉熵。验证集正确率如下：

网络	20 分类	100 分类
resnet18	None	82.15%
resnet34	92.1%	85.95%
resnet50	91.8%	85.0%
resnext50	92.25%	85.2%
resnext101	90.75%	None
wide_resnet50	None	85.9%

与之前相比有些许提升。

5. 训练结果分析

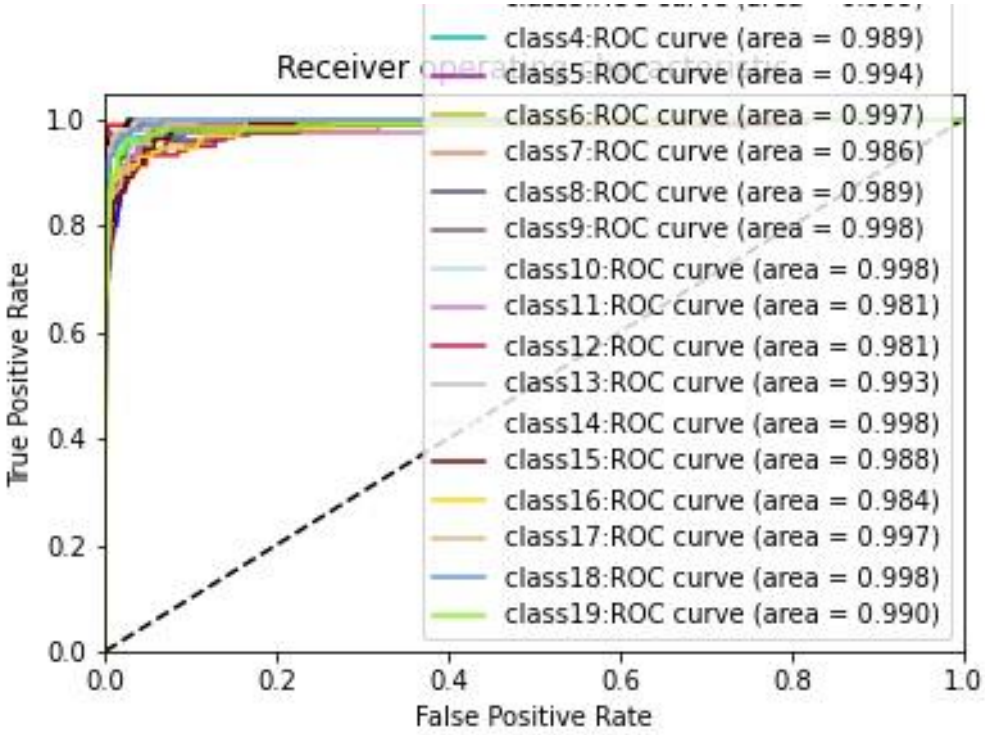
我事先划分出了自己的测试集，虽然验证集对于网络的训练没有影响，但是还是划出了自测集便于操作。

5.1 20 分类

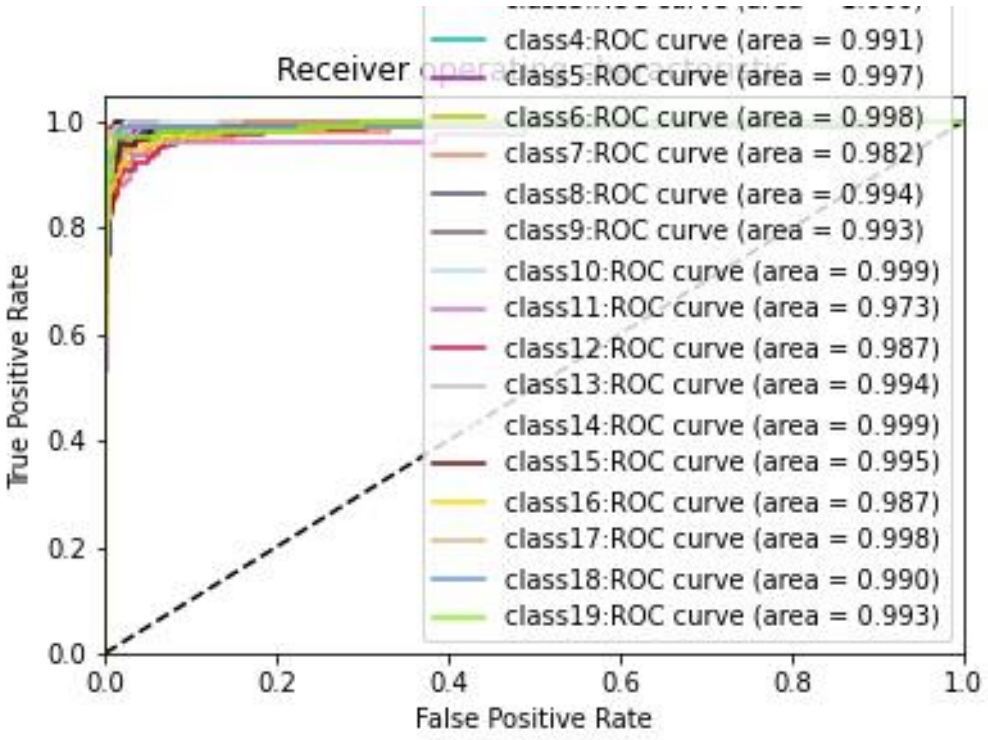
各种模型在自测集上的准确率如下:

```
Get Model weights/coarse/resnet34mpl.tar
loss on test: 0.839144, accuracy on test: 0.921000
Get Model weights/coarse/resnext101.tar
loss on test: 0.477916, accuracy on test: 0.907500
Get Model weights/coarse/resnext50Adam.tar
loss on test: 0.373280, accuracy on test: 0.911000
Get Model weights/coarse/resnext50init0.01SGD.tar
loss on test: 0.338991, accuracy on test: 0.922500
Get Model weights/coarse/resnext50mp.tar
loss on test: 0.309580, accuracy on test: 0.918000
Get Model weights/coarse/weights_resnet34_init_lr0.010000_SGD_MLR_50epoch_weight.tar
loss on test: 0.419843, accuracy on test: 0.908000
Get Model weights/coarse/weights_resnext50_32x4d_init_lr0.010000_SGD_MLR_50epochmp.tar
loss on test: 0.330013, accuracy on test: 0.908000
Get Model weights/coarse/resnet18mpl.tar
loss on test: 0.960978, accuracy on test: 0.878000
```

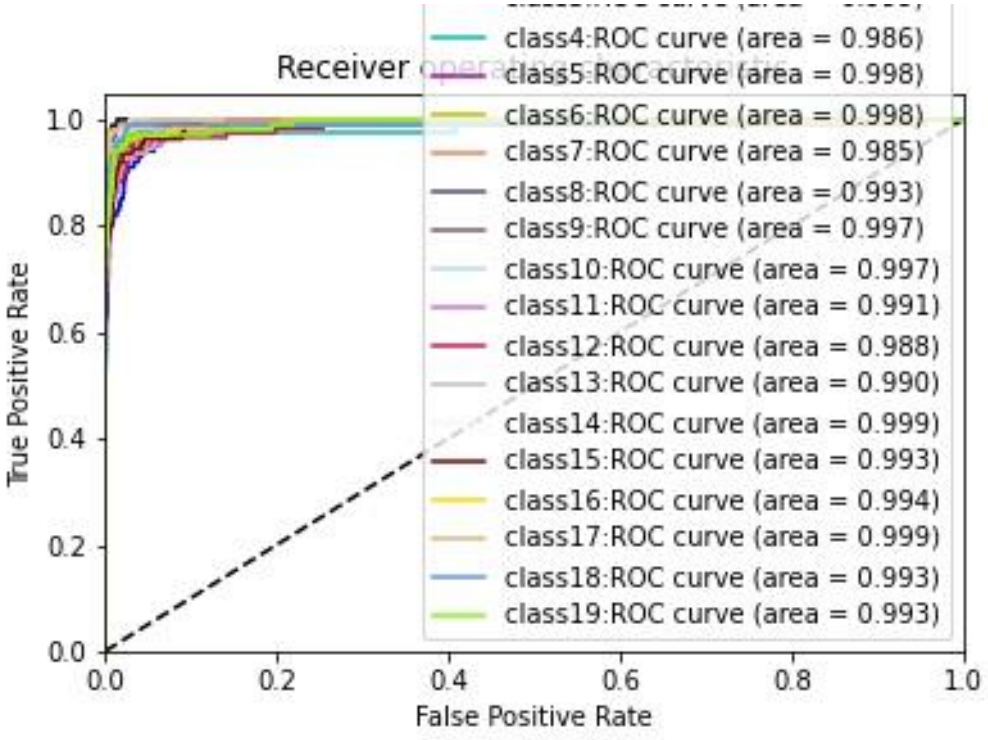
resnet34 测试集上各类别 roc 曲线如下，自测集正确率为 90.8%



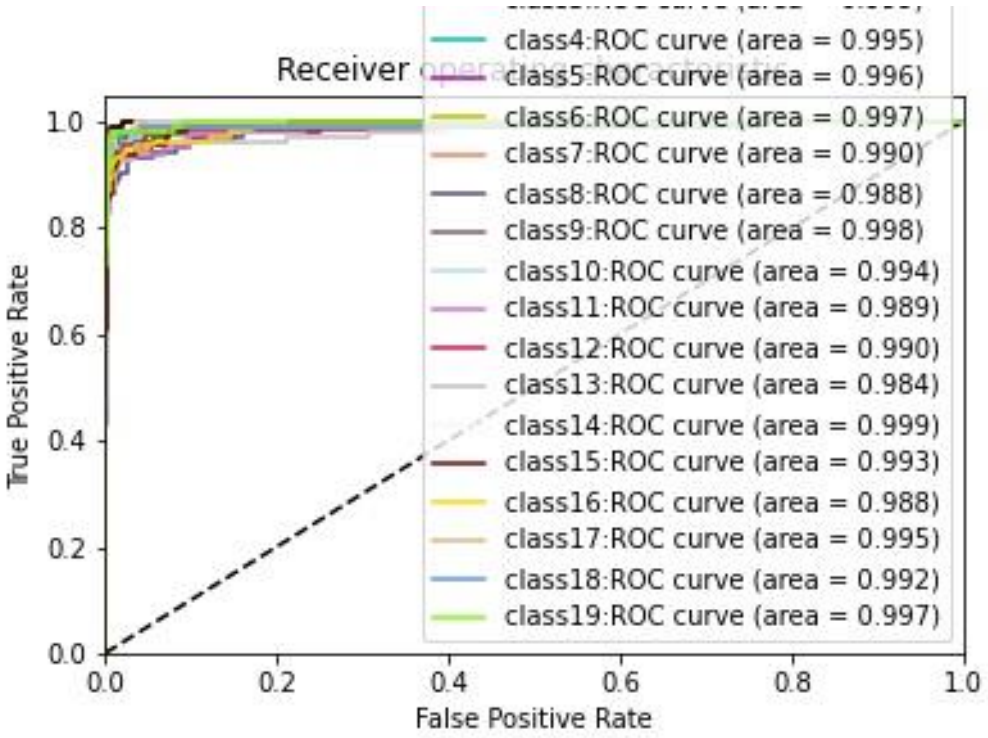
resnext50 使用 Adam 优化器各类别 roc 如下，自测集正确率 91.1%。



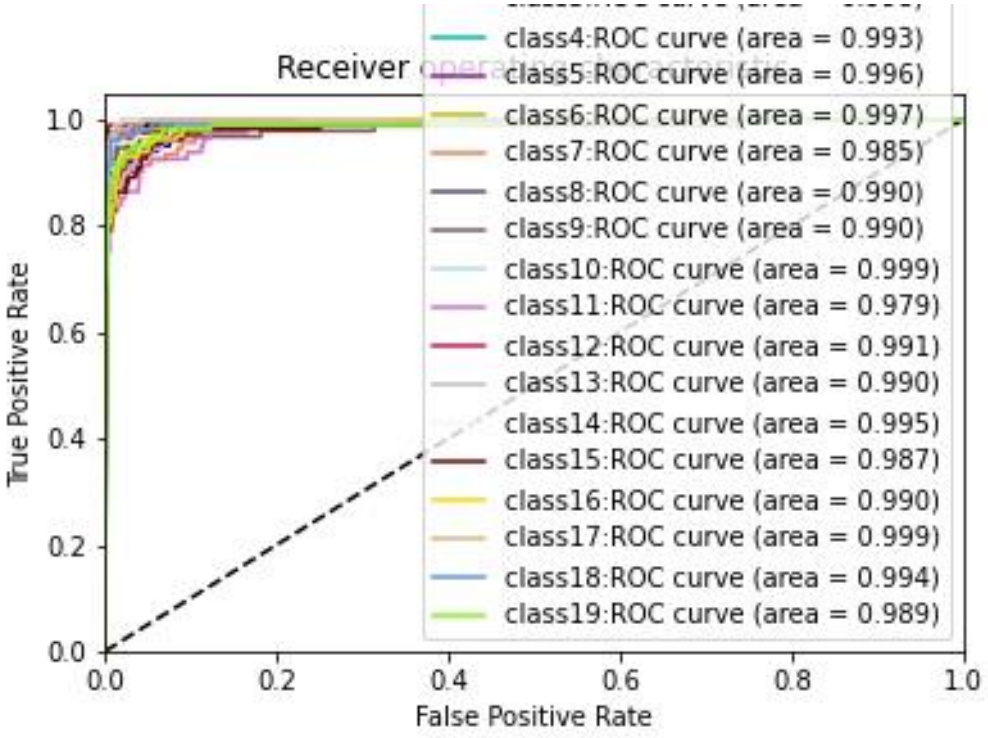
resnext50 使用 MultiStep 策略 SGD 优化器各类别 roc 如下，自测集正确率 92.25%。



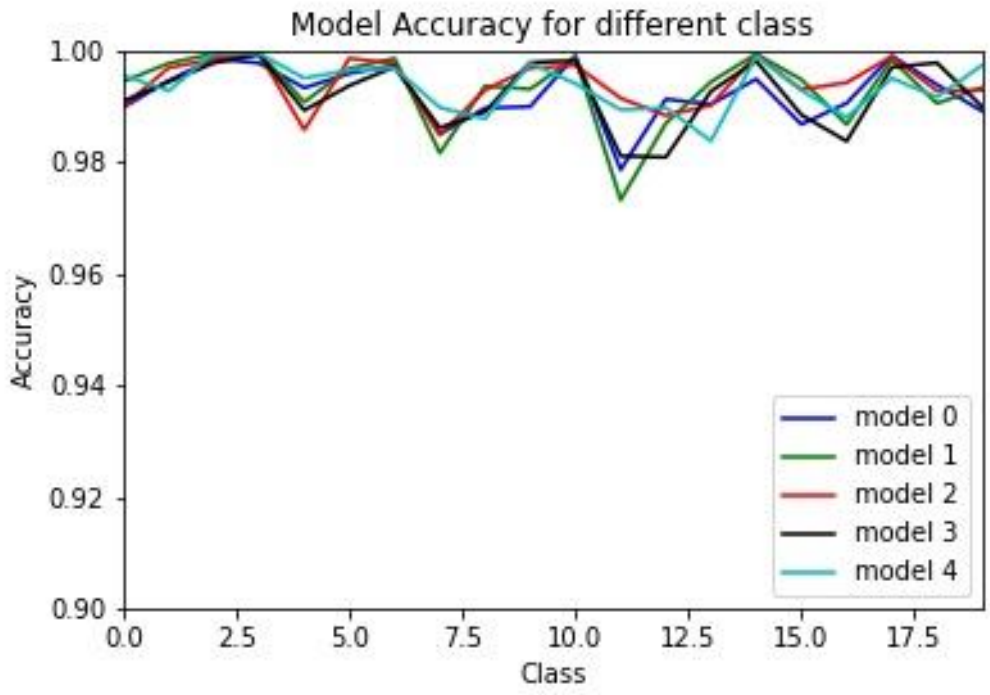
resnext50 使用 mixup 和 LabelSmoothingCrossEntropy 的各类别 roc 如下，自测集正确率 90.8%



resnext101 各类别 roc 曲线如下。验证集正确率 90.75%



可见，还是有不同类别正确率不均衡的问题，做出各个模型在不同类别上的正确率如下图，可见 resnet 系列对于不同类别的准确率比较一致，但是也会有一些不同，说明可以互补。



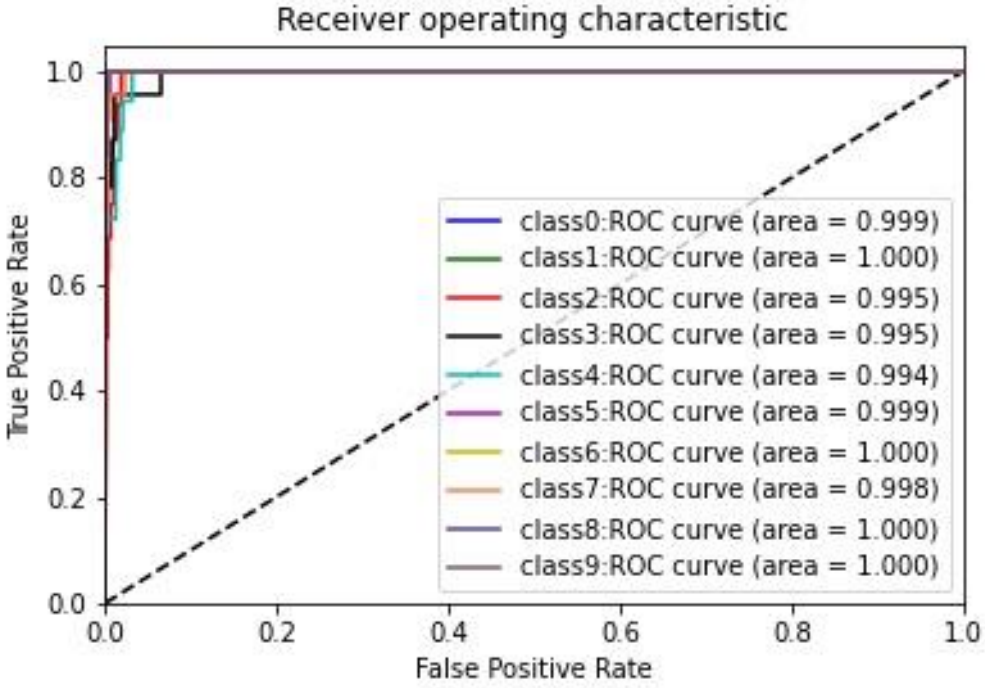
5.2 100 分类

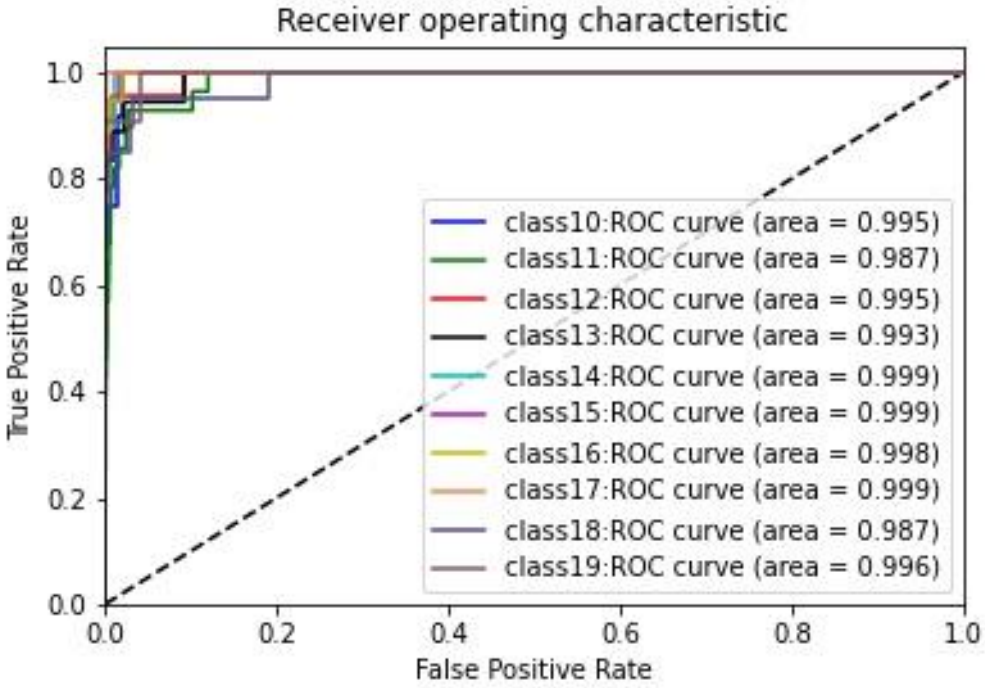
各种模型在自测集上准确率如下:

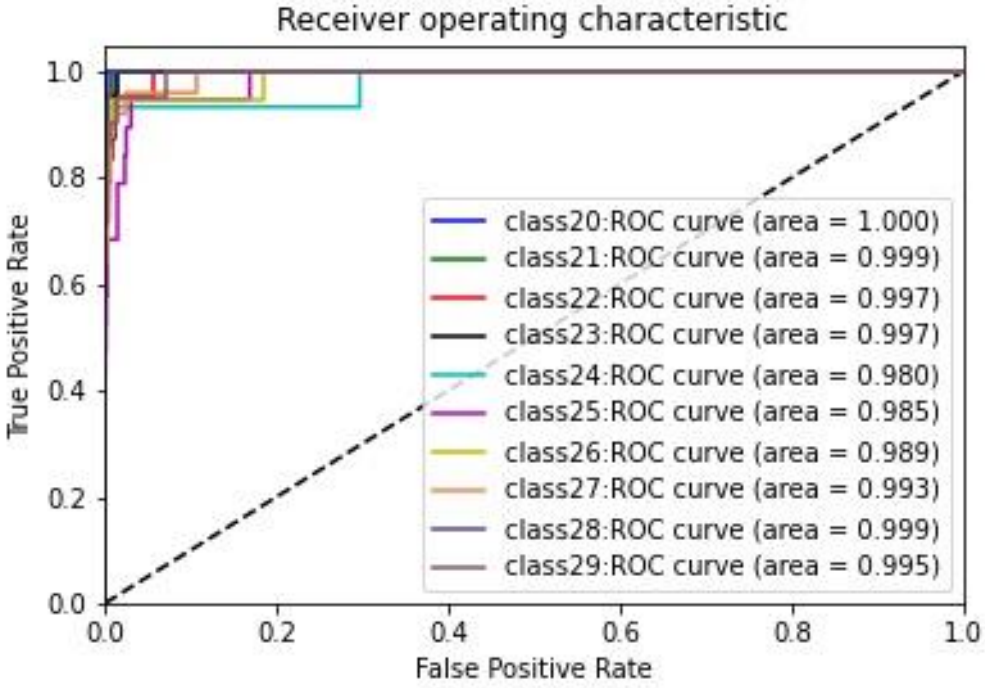
```
Get Model weights/fine/resnet34init0.01SGD.tar
loss on test: 0.627428, accuracy on test: 0.852000
Get Model weights/fine/resnet34mpl.tar
loss on test: 1.293145, accuracy on test: 0.859500
Get Model weights/fine/resnext50init0.01SGD.tar
loss on test: 0.619847, accuracy on test: 0.852000


<ipython-input-17-d49b7e8b8ee4>:86: RuntimeWarning: More than 20 figures have been opened. Figures created through the pyplot interface
(`matplotlib.pyplot.figure`) are retained until explicitly closed and may consume too much memory. (To control this warning, see the rcP
aram `figure.max_open_warning`).
  plt.figure()


Get Model weights/fine/wide_resnet50.tar
loss on test: 0.578852, accuracy on test: 0.859000
Get Model weights/fine/resnet18mpl.tar
loss on test: 1.485849, accuracy on test: 0.821500
Get Model weights/fine/resnet50mp.tar
loss on test: 0.586301, accuracy on test: 0.850000
Get Model weights/fine/resnext50epoch100.tar
loss on test: 0.590023, accuracy on test: 0.850000
```

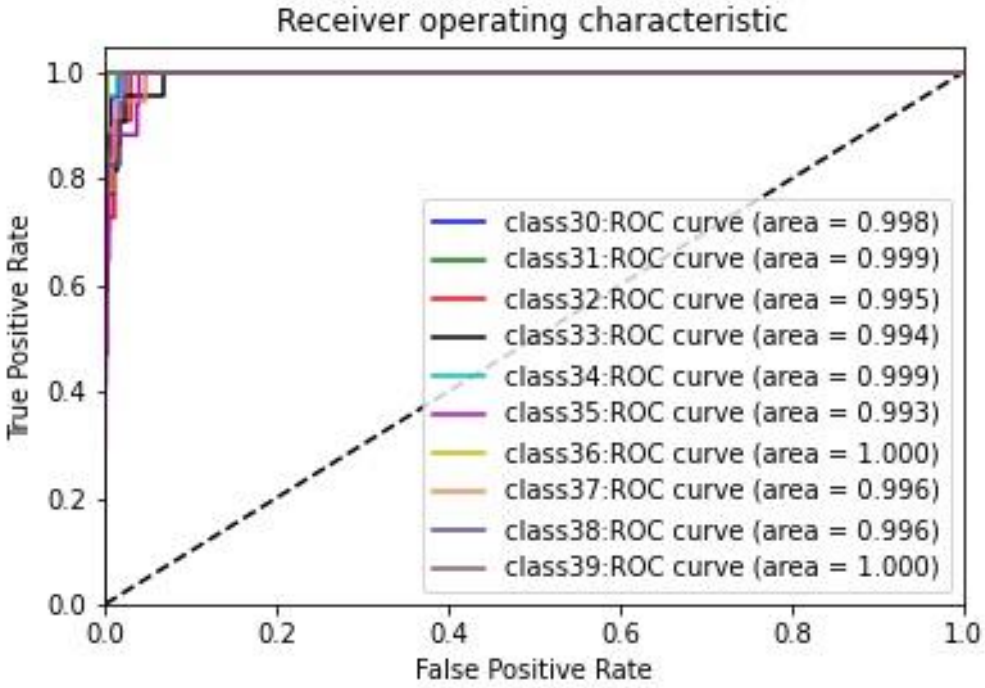
由于 100 分类类别太多，所以每个模型预测出的 roc 曲线分在 10 张图上。

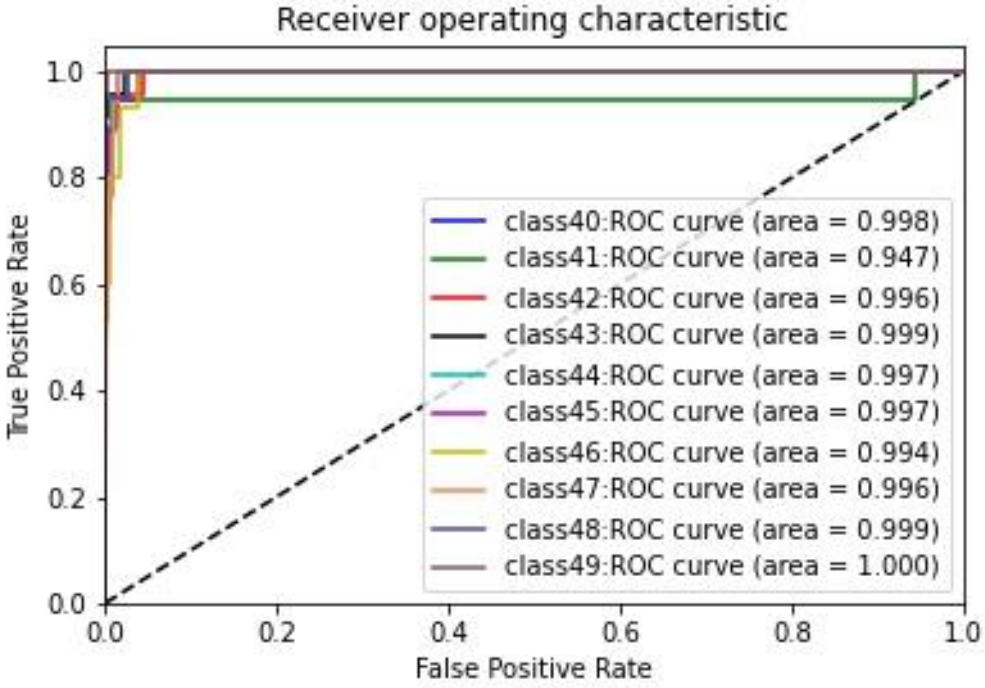
resnet34 测试集上各类别 roc 曲线如下，自测集正确率为 85.2%

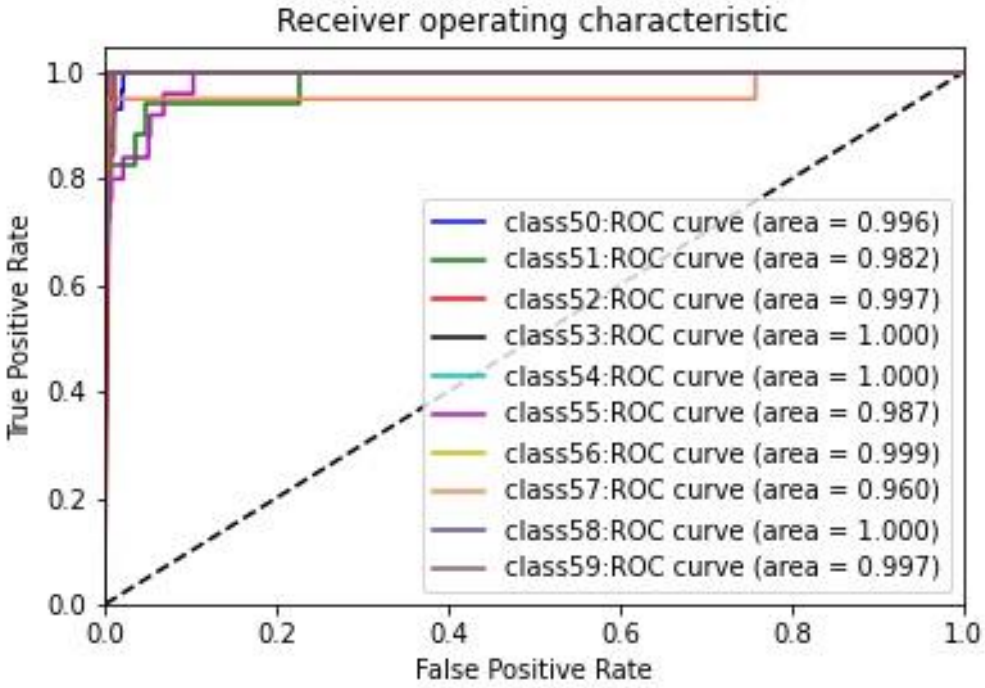

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class0:ROC curve (area = 0.999)
class1:ROC curve (area = 1.000)
class2:ROC curve (area = 0.995)
class3:ROC curve (area = 0.995)
class4:ROC curve (area = 0.994)
class5:ROC curve (area = 0.999)
class6:ROC curve (area = 1.000)
class7:ROC curve (area = 0.998)
class8:ROC curve (area = 1.000)
class9:ROC curve (area = 1.000)

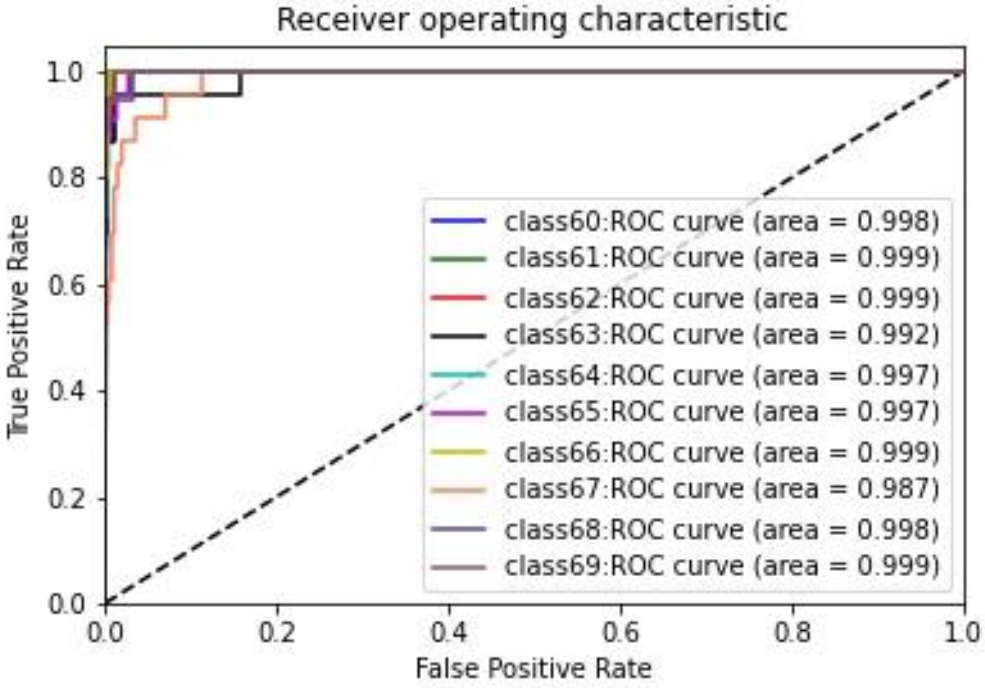

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class10:ROC curve (area = 0.995)
class11:ROC curve (area = 0.987)
class12:ROC curve (area = 0.995)
class13:ROC curve (area = 0.993)
class14:ROC curve (area = 0.999)
class15:ROC curve (area = 0.999)
class16:ROC curve (area = 0.998)
class17:ROC curve (area = 0.999)
class18:ROC curve (area = 0.987)
class19:ROC curve (area = 0.996)

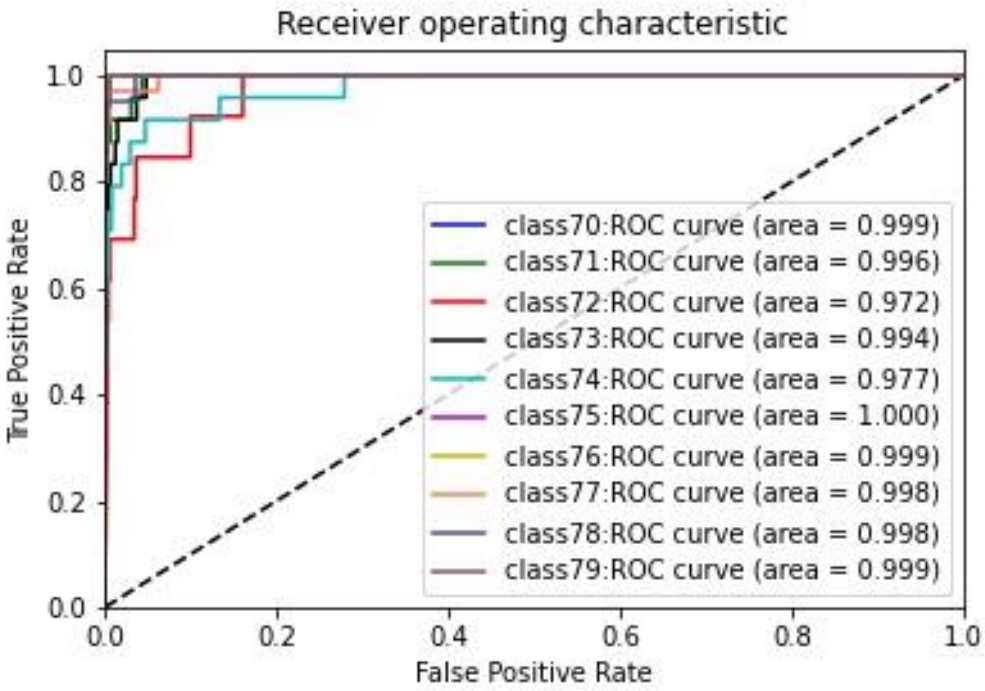

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class20:ROC curve (area = 1.000)
class21:ROC curve (area = 0.999)
class22:ROC curve (area = 0.997)
class23:ROC curve (area = 0.997)
class24:ROC curve (area = 0.980)
class25:ROC curve (area = 0.985)
class26:ROC curve (area = 0.989)
class27:ROC curve (area = 0.993)
class28:ROC curve (area = 0.999)
class29:ROC curve (area = 0.995)

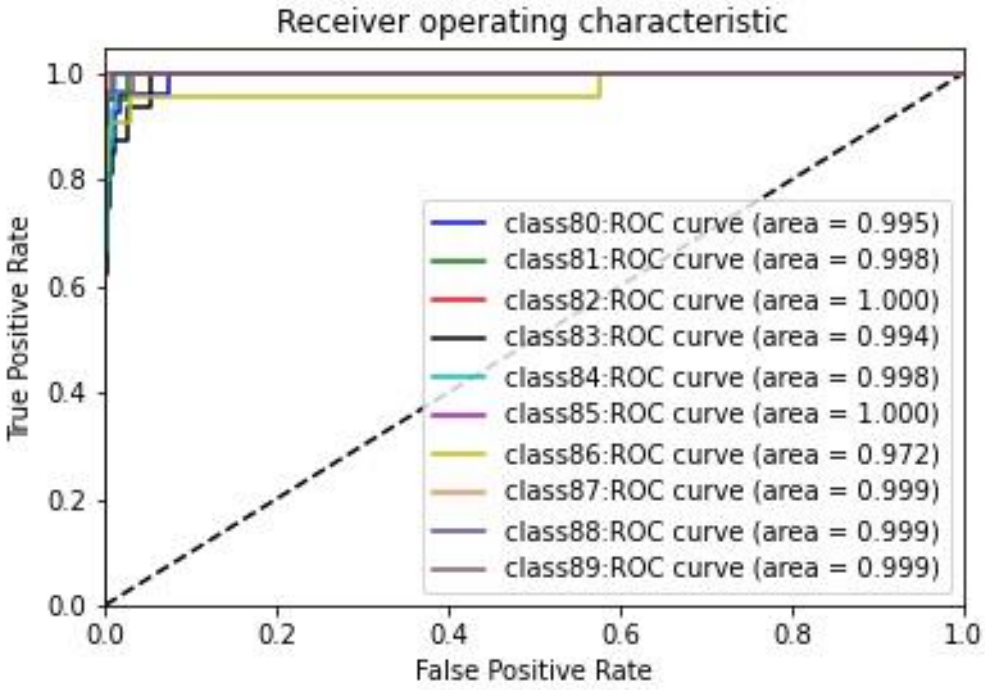

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class30:ROC curve (area = 0.998)
class31:ROC curve (area = 0.999)
class32:ROC curve (area = 0.995)
class33:ROC curve (area = 0.994)
class34:ROC curve (area = 0.999)
class35:ROC curve (area = 0.993)
class36:ROC curve (area = 1.000)
class37:ROC curve (area = 0.996)
class38:ROC curve (area = 0.996)
class39:ROC curve (area = 1.000)

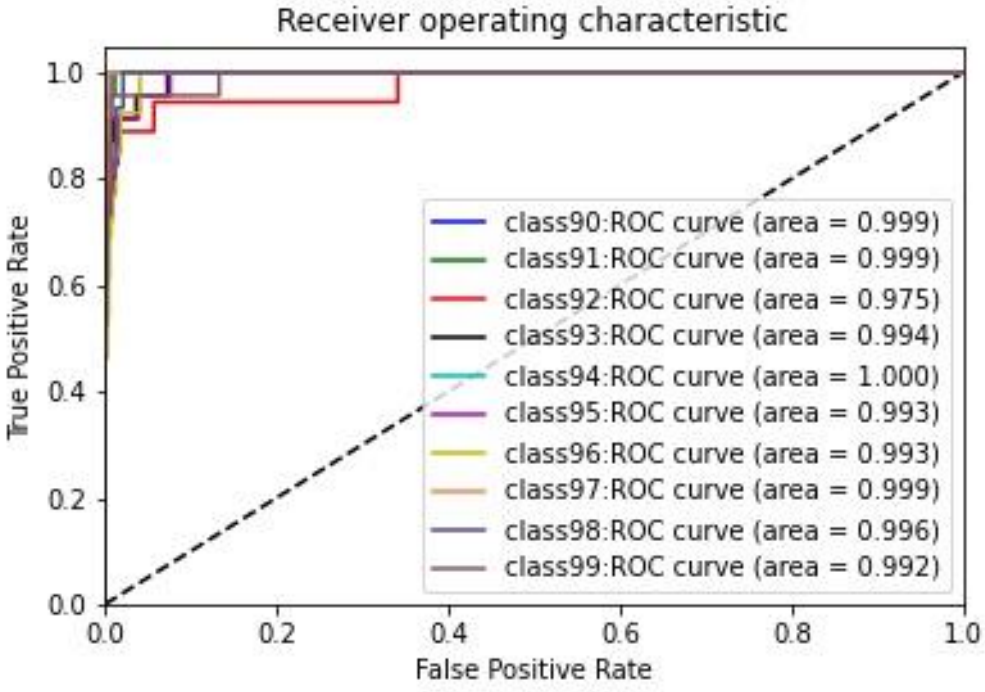

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class40:ROC curve (area = 0.998)
class41:ROC curve (area = 0.947)
class42:ROC curve (area = 0.996)
class43:ROC curve (area = 0.999)
class44:ROC curve (area = 0.997)
class45:ROC curve (area = 0.997)
class46:ROC curve (area = 0.994)
class47:ROC curve (area = 0.996)
class48:ROC curve (area = 0.999)
class49:ROC curve (area = 1.000)


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class50:ROC curve (area = 0.996)
class51:ROC curve (area = 0.982)
class52:ROC curve (area = 0.997)
class53:ROC curve (area = 1.000)
class54:ROC curve (area = 1.000)
class55:ROC curve (area = 0.987)
class56:ROC curve (area = 0.999)
class57:ROC curve (area = 0.960)
class58:ROC curve (area = 1.000)
class59:ROC curve (area = 0.997)

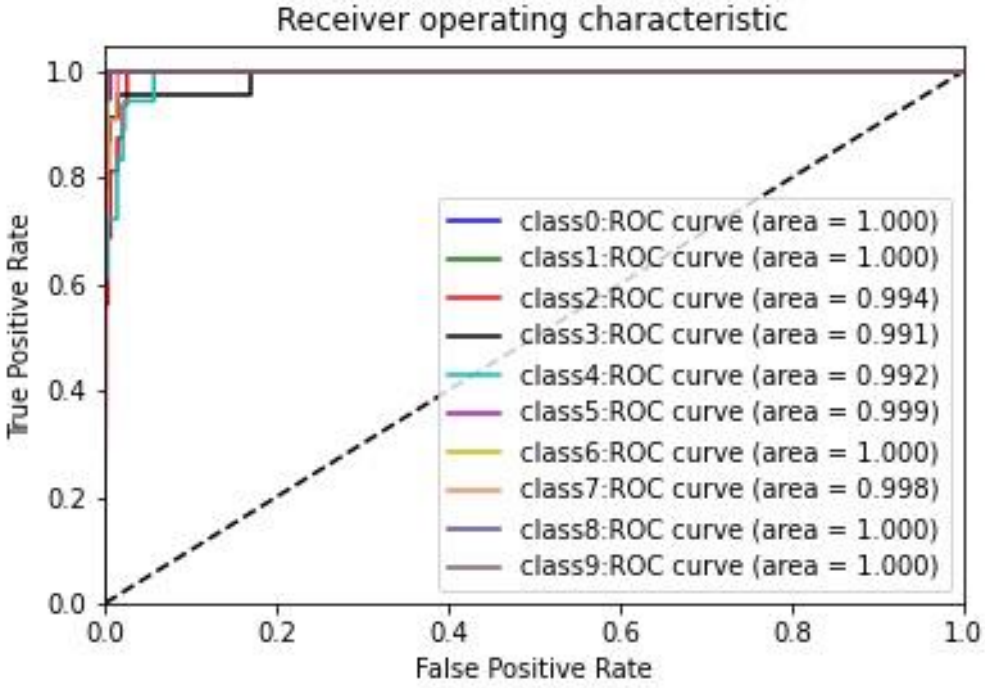

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class60:ROC curve (area = 0.998)
class61:ROC curve (area = 0.999)
class62:ROC curve (area = 0.999)
class63:ROC curve (area = 0.992)
class64:ROC curve (area = 0.997)
class65:ROC curve (area = 0.997)
class66:ROC curve (area = 0.999)
class67:ROC curve (area = 0.987)
class68:ROC curve (area = 0.998)
class69:ROC curve (area = 0.999)

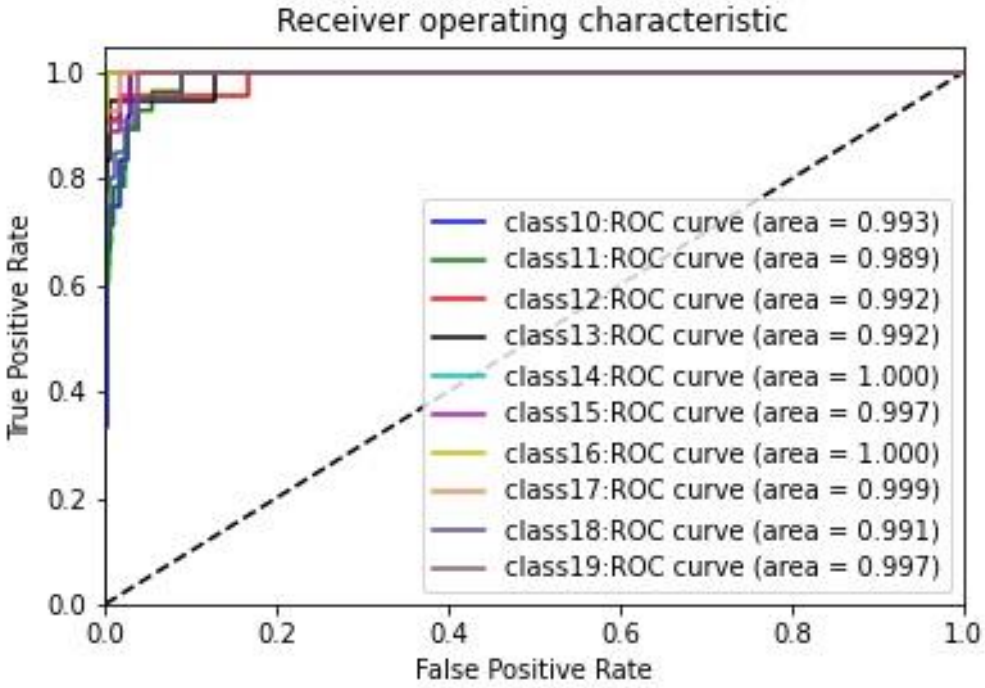

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class70:ROC curve (area = 0.999)
class71:ROC curve (area = 0.996)
class72:ROC curve (area = 0.972)
class73:ROC curve (area = 0.994)
class74:ROC curve (area = 0.977)
class75:ROC curve (area = 1.000)
class76:ROC curve (area = 0.999)
class77:ROC curve (area = 0.998)
class78:ROC curve (area = 0.998)
class79:ROC curve (area = 0.999)

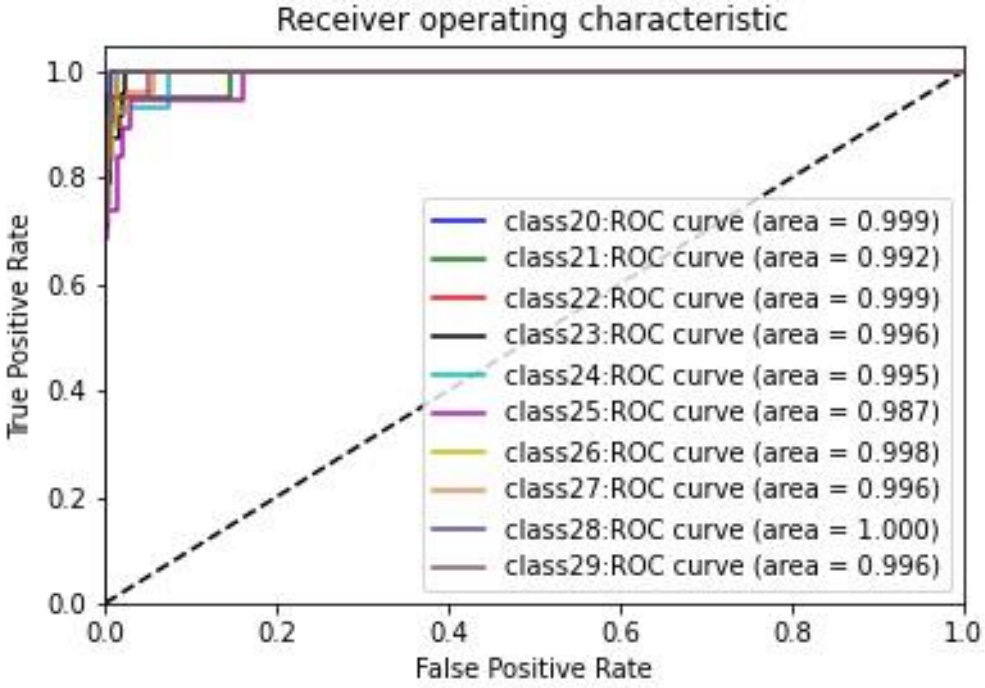


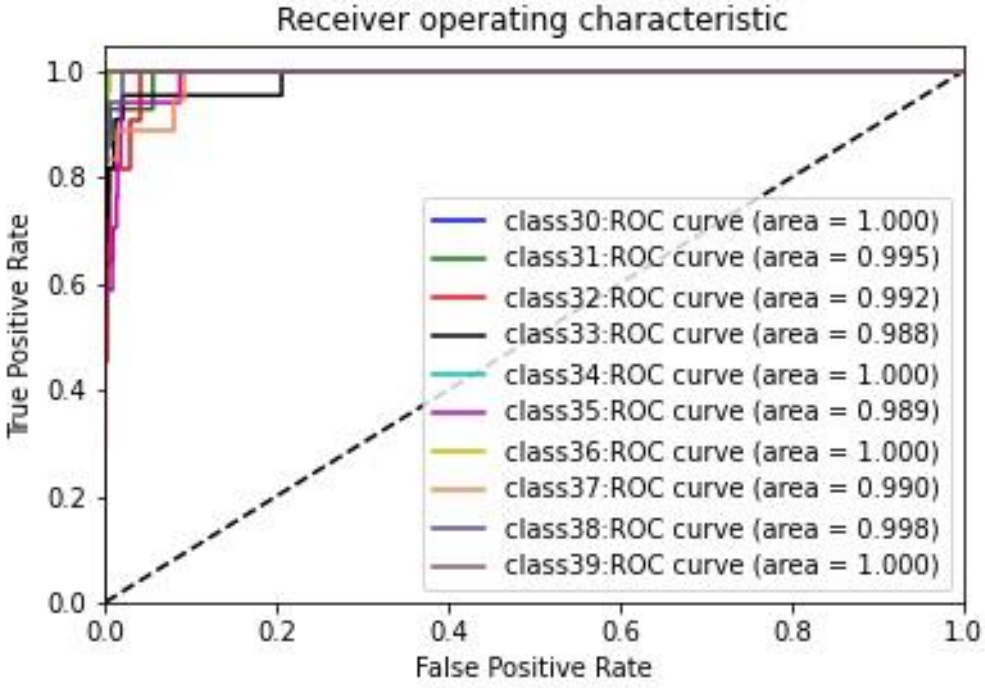


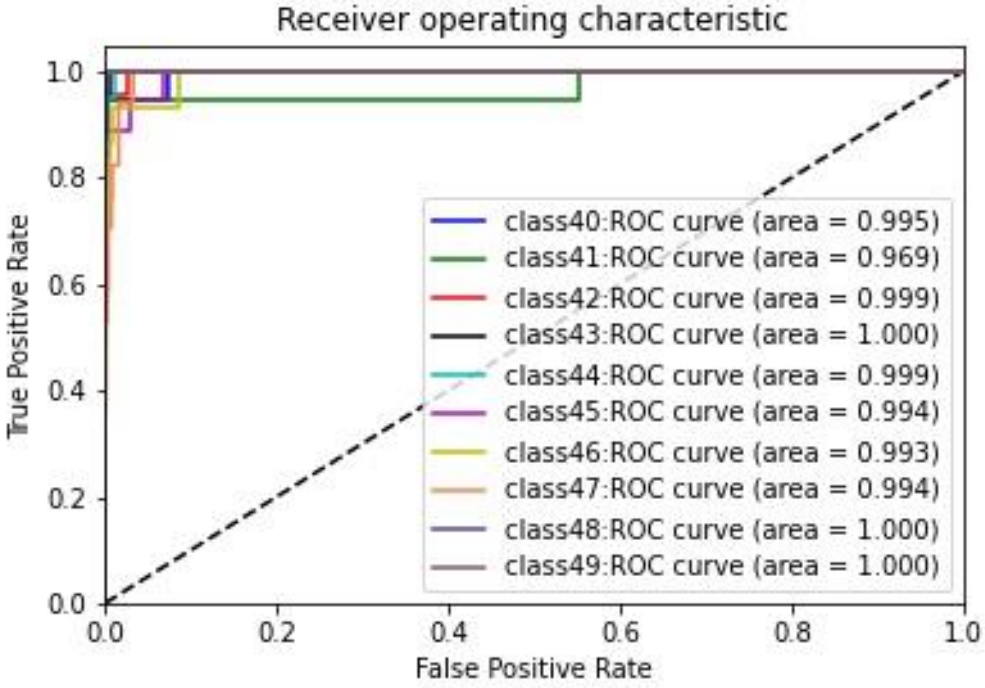
resnext50 各类别 roc 曲线如下，自测集准确率 85.2%

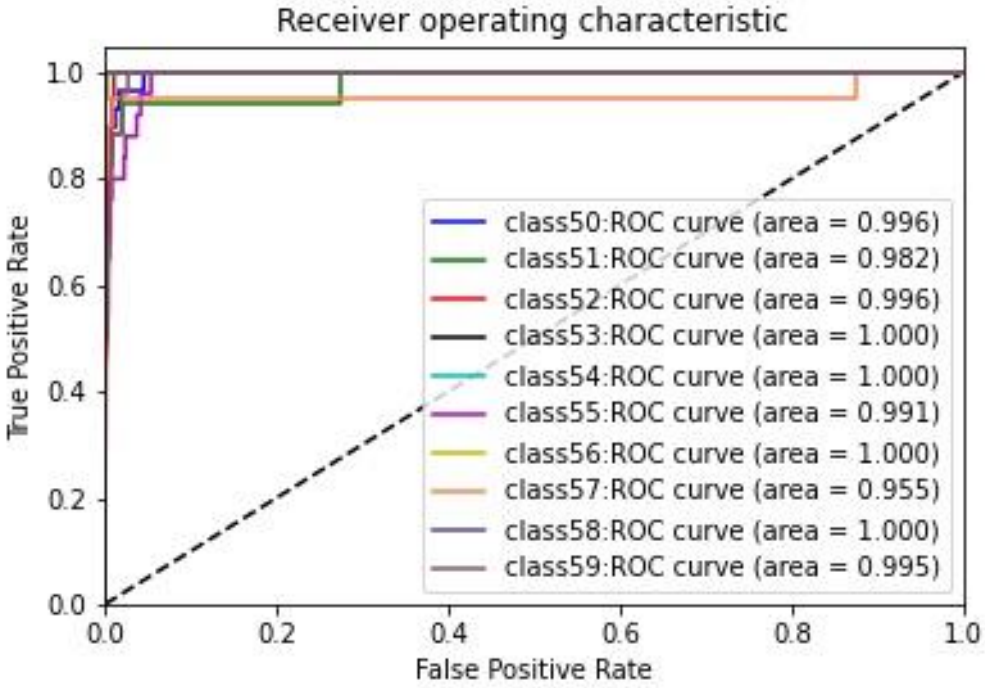

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class0:ROC curve (area = 1.000)
class1:ROC curve (area = 1.000)
class2:ROC curve (area = 0.994)
class3:ROC curve (area = 0.991)
class4:ROC curve (area = 0.992)
class5:ROC curve (area = 0.999)
class6:ROC curve (area = 1.000)
class7:ROC curve (area = 0.998)
class8:ROC curve (area = 1.000)
class9:ROC curve (area = 1.000)

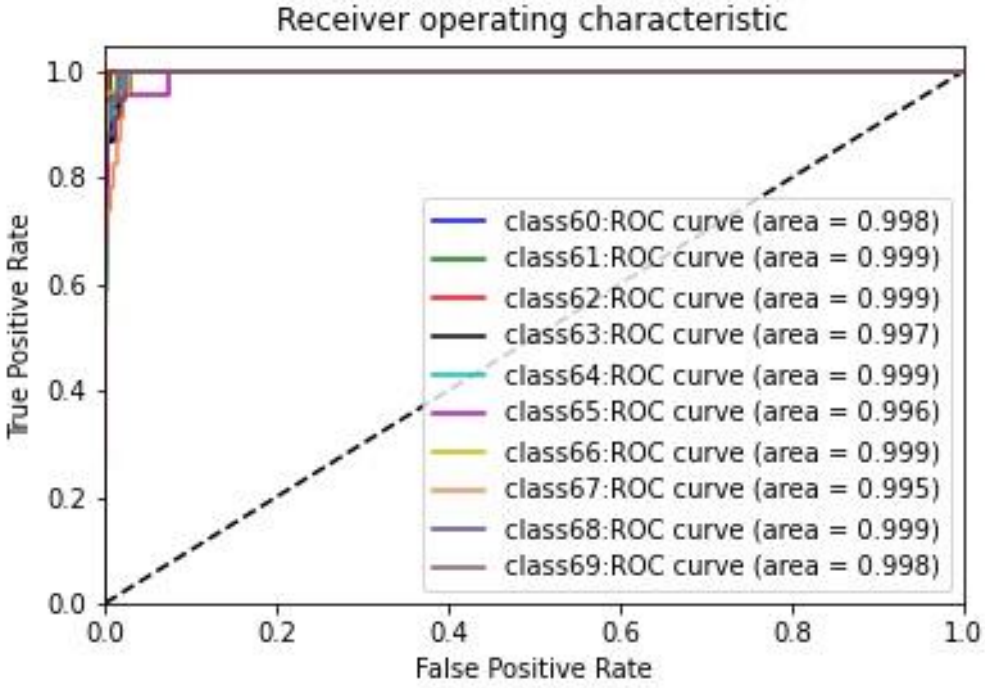

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class10:ROC curve (area = 0.993)
class11:ROC curve (area = 0.989)
class12:ROC curve (area = 0.992)
class13:ROC curve (area = 0.992)
class14:ROC curve (area = 1.000)
class15:ROC curve (area = 0.997)
class16:ROC curve (area = 1.000)
class17:ROC curve (area = 0.999)
class18:ROC curve (area = 0.991)
class19:ROC curve (area = 0.997)

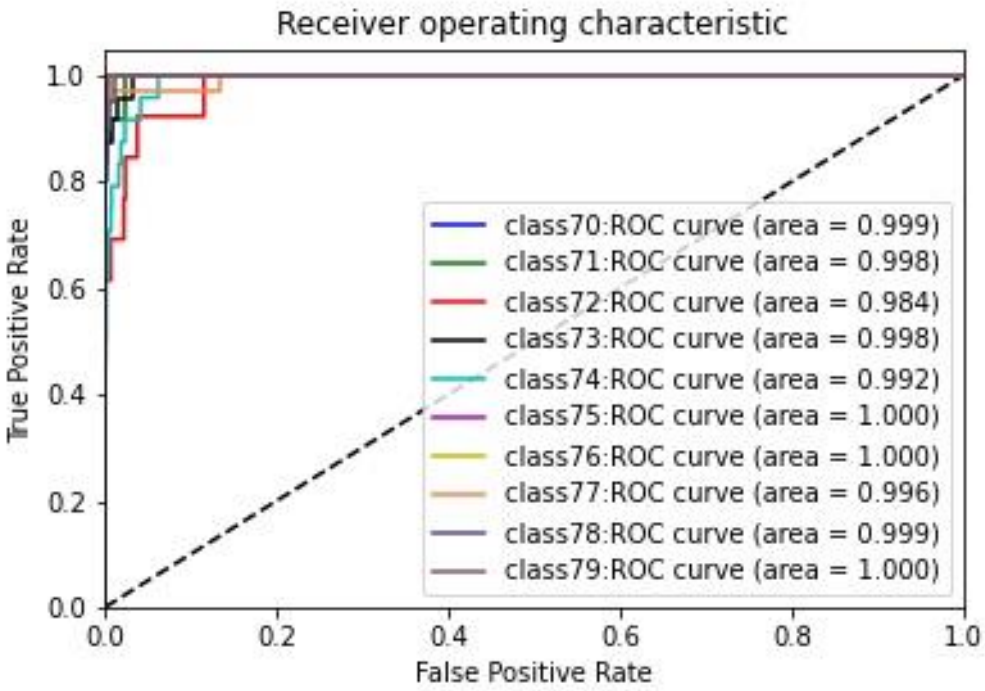

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class20:ROC curve (area = 0.999)
class21:ROC curve (area = 0.992)
class22:ROC curve (area = 0.999)
class23:ROC curve (area = 0.996)
class24:ROC curve (area = 0.995)
class25:ROC curve (area = 0.987)
class26:ROC curve (area = 0.998)
class27:ROC curve (area = 0.996)
class28:ROC curve (area = 1.000)
class29:ROC curve (area = 0.996)

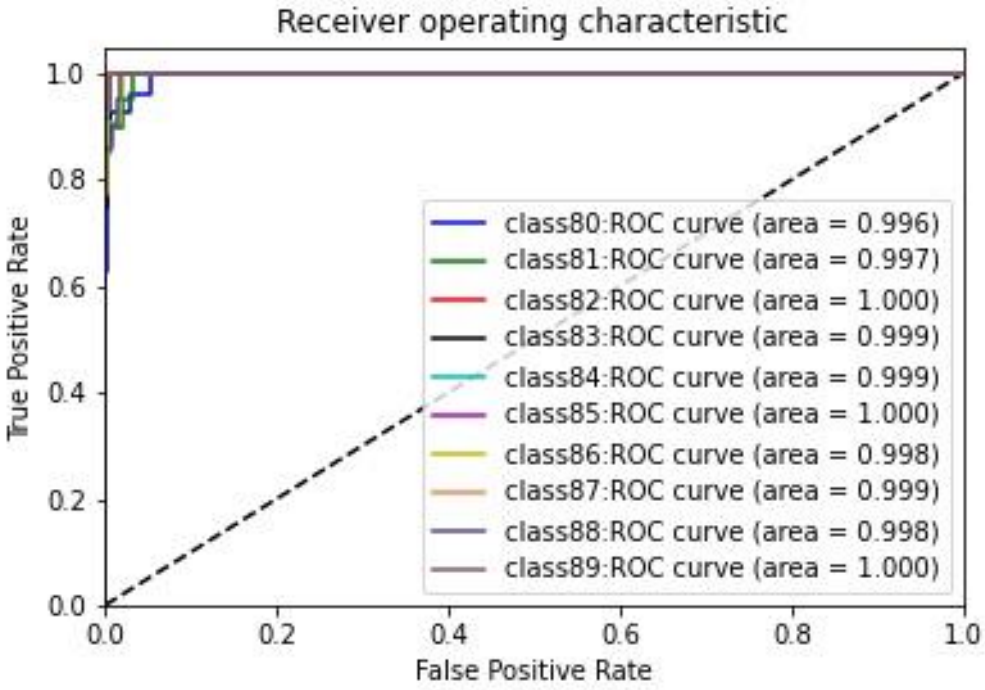

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class30:ROC curve (area = 1.000)
class31:ROC curve (area = 0.995)
class32:ROC curve (area = 0.992)
class33:ROC curve (area = 0.988)
class34:ROC curve (area = 1.000)
class35:ROC curve (area = 0.989)
class36:ROC curve (area = 1.000)
class37:ROC curve (area = 0.990)
class38:ROC curve (area = 0.998)
class39:ROC curve (area = 1.000)

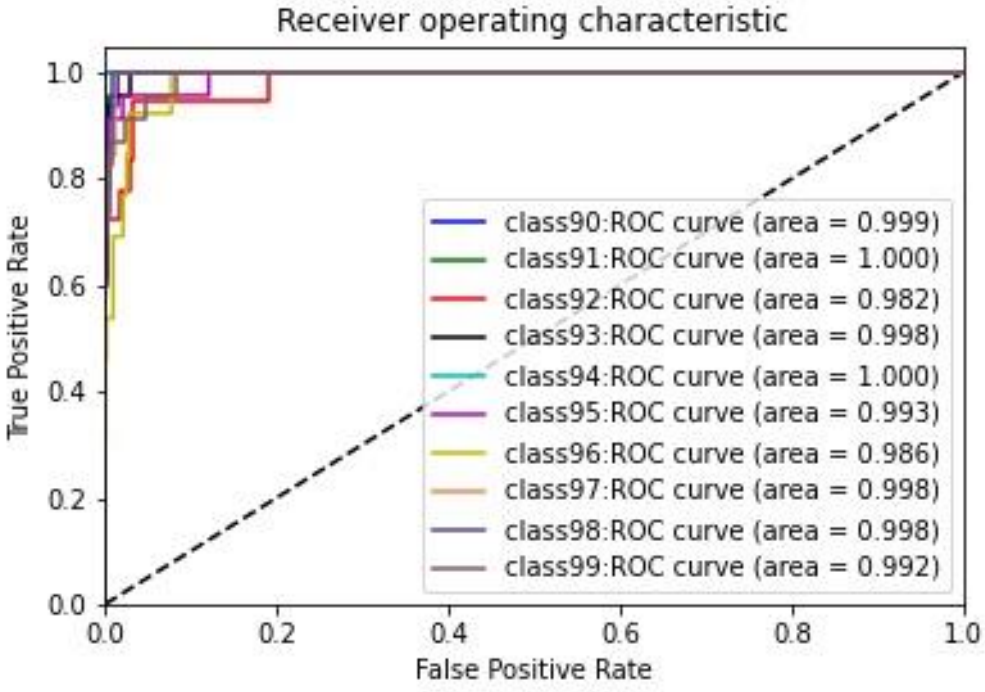

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class40:ROC curve (area = 0.995)
class41:ROC curve (area = 0.969)
class42:ROC curve (area = 0.999)
class43:ROC curve (area = 1.000)
class44:ROC curve (area = 0.999)
class45:ROC curve (area = 0.994)
class46:ROC curve (area = 0.993)
class47:ROC curve (area = 0.994)
class48:ROC curve (area = 1.000)
class49:ROC curve (area = 1.000)


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class50:ROC curve (area = 0.996)
class51:ROC curve (area = 0.982)
class52:ROC curve (area = 0.996)
class53:ROC curve (area = 1.000)
class54:ROC curve (area = 1.000)
class55:ROC curve (area = 0.991)
class56:ROC curve (area = 1.000)
class57:ROC curve (area = 0.955)
class58:ROC curve (area = 1.000)
class59:ROC curve (area = 0.995)

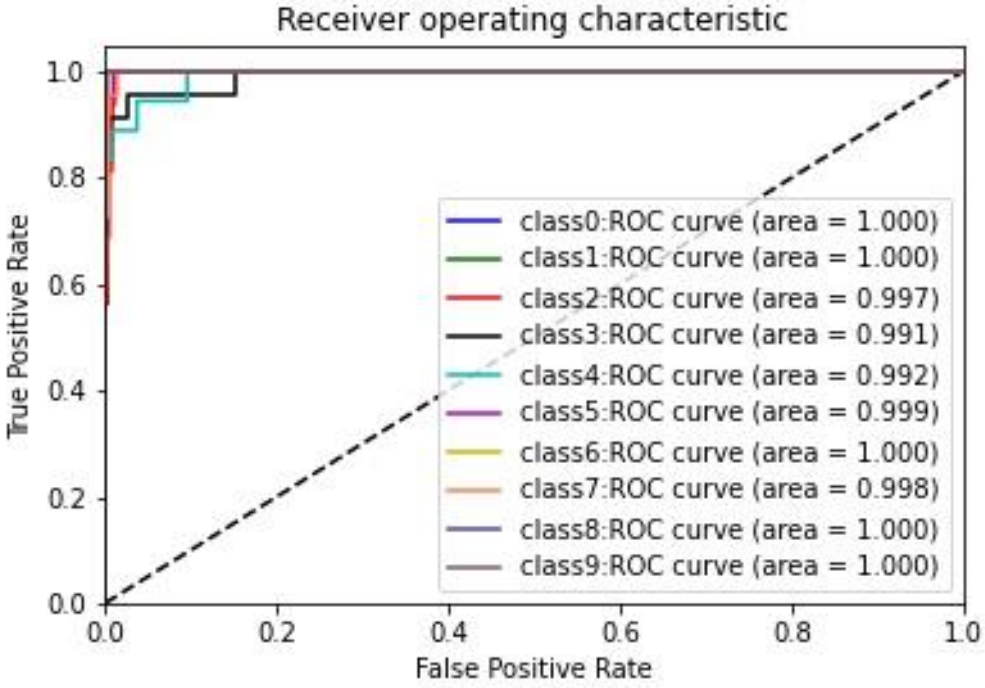

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class60:ROC curve (area = 0.998)
class61:ROC curve (area = 0.999)
class62:ROC curve (area = 0.999)
class63:ROC curve (area = 0.997)
class64:ROC curve (area = 0.999)
class65:ROC curve (area = 0.996)
class66:ROC curve (area = 0.999)
class67:ROC curve (area = 0.995)
class68:ROC curve (area = 0.999)
class69:ROC curve (area = 0.998)

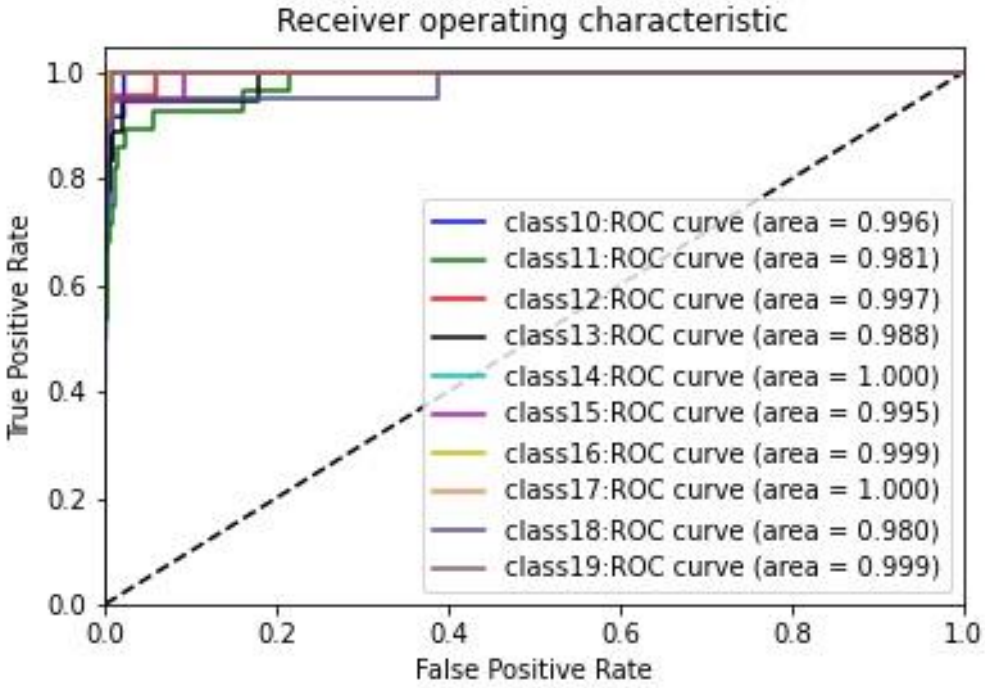

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class70:ROC curve (area = 0.999)
class71:ROC curve (area = 0.998)
class72:ROC curve (area = 0.984)
class73:ROC curve (area = 0.998)
class74:ROC curve (area = 0.992)
class75:ROC curve (area = 1.000)
class76:ROC curve (area = 1.000)
class77:ROC curve (area = 0.996)
class78:ROC curve (area = 0.999)
class79:ROC curve (area = 1.000)

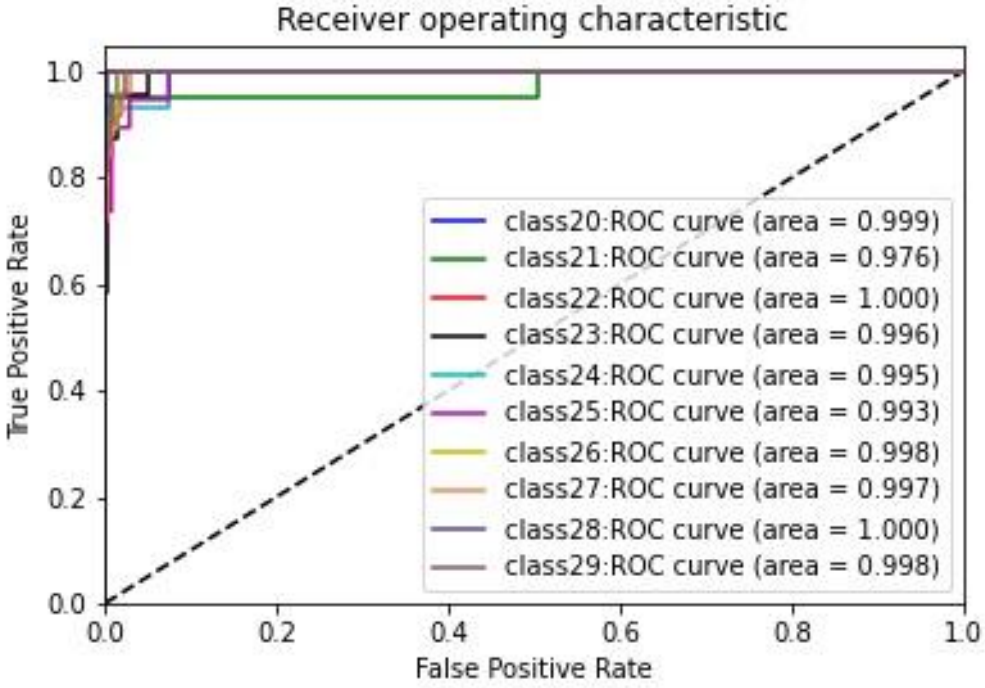


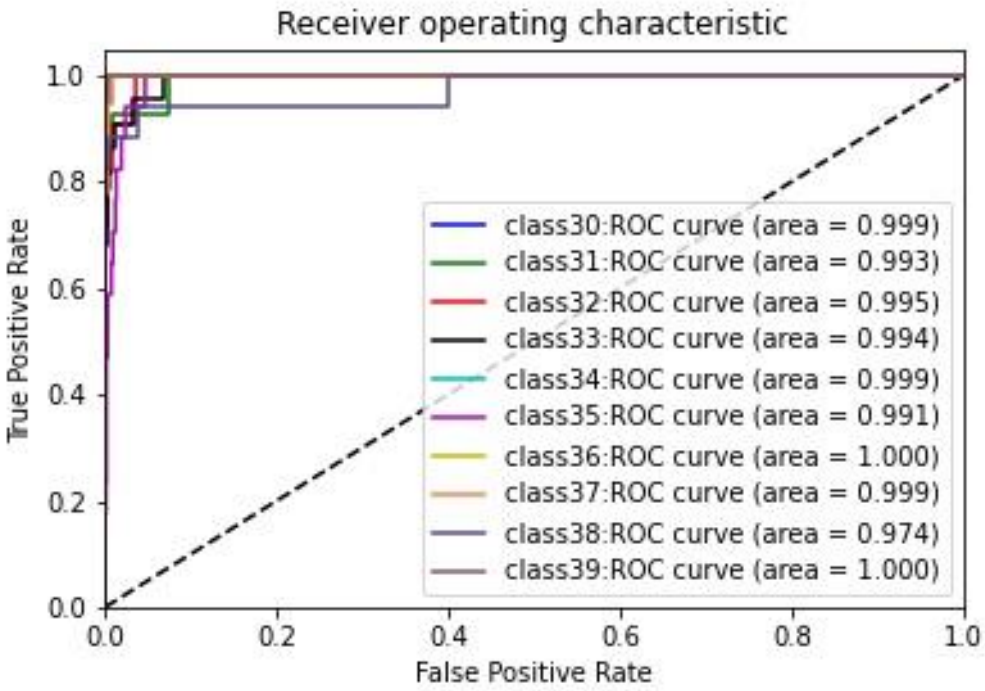


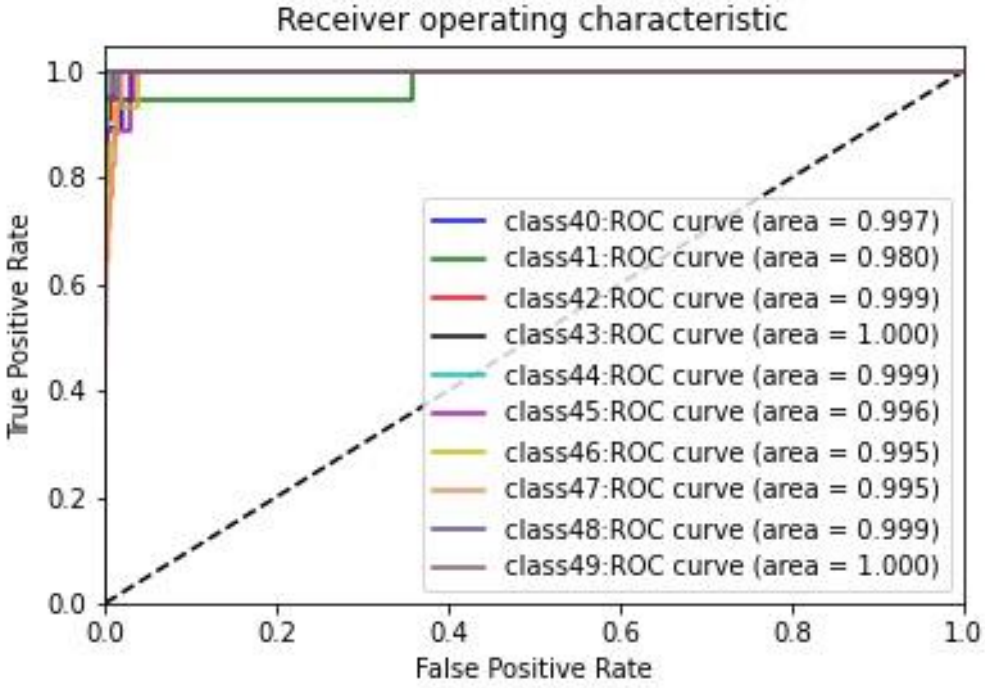
wide_resnet50 各类别 roc 曲线如下，自测集准确率 85.9%。

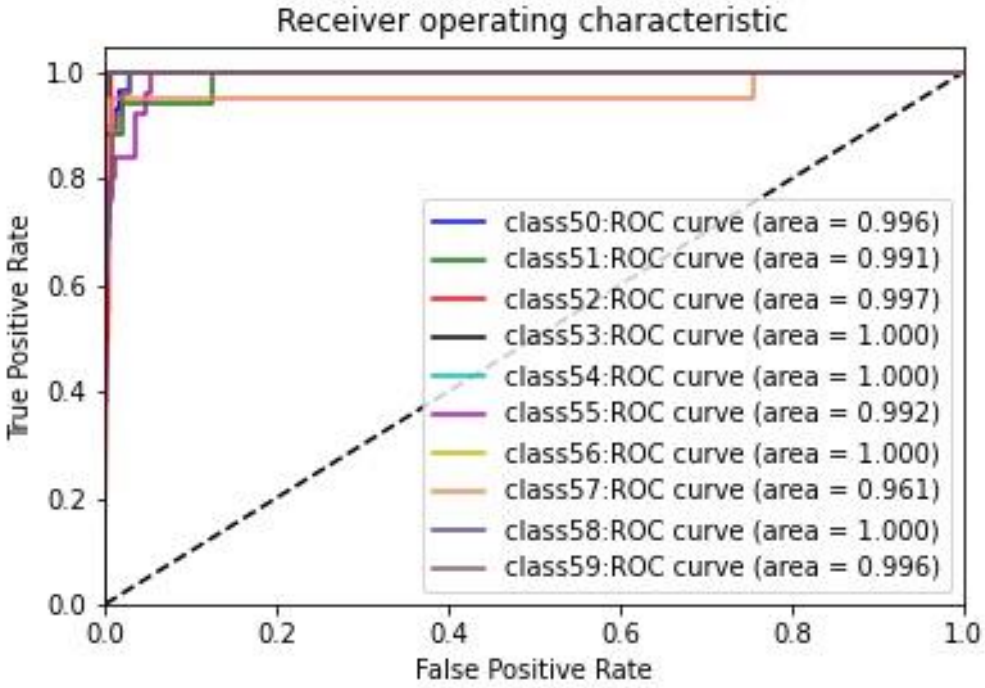

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class0:ROC curve (area = 1.000)
class1:ROC curve (area = 1.000)
class2:ROC curve (area = 0.997)
class3:ROC curve (area = 0.991)
class4:ROC curve (area = 0.992)
class5:ROC curve (area = 0.999)
class6:ROC curve (area = 1.000)
class7:ROC curve (area = 0.998)
class8:ROC curve (area = 1.000)
class9:ROC curve (area = 1.000)

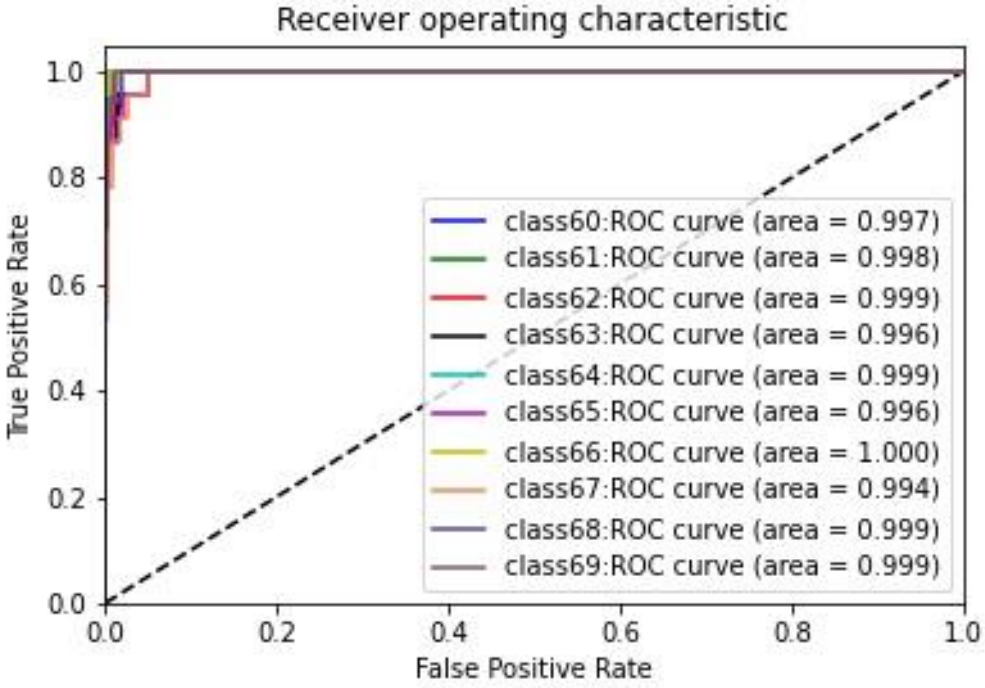

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class10:ROC curve (area = 0.996)
class11:ROC curve (area = 0.981)
class12:ROC curve (area = 0.997)
class13:ROC curve (area = 0.988)
class14:ROC curve (area = 1.000)
class15:ROC curve (area = 0.995)
class16:ROC curve (area = 0.999)
class17:ROC curve (area = 1.000)
class18:ROC curve (area = 0.980)
class19:ROC curve (area = 0.999)

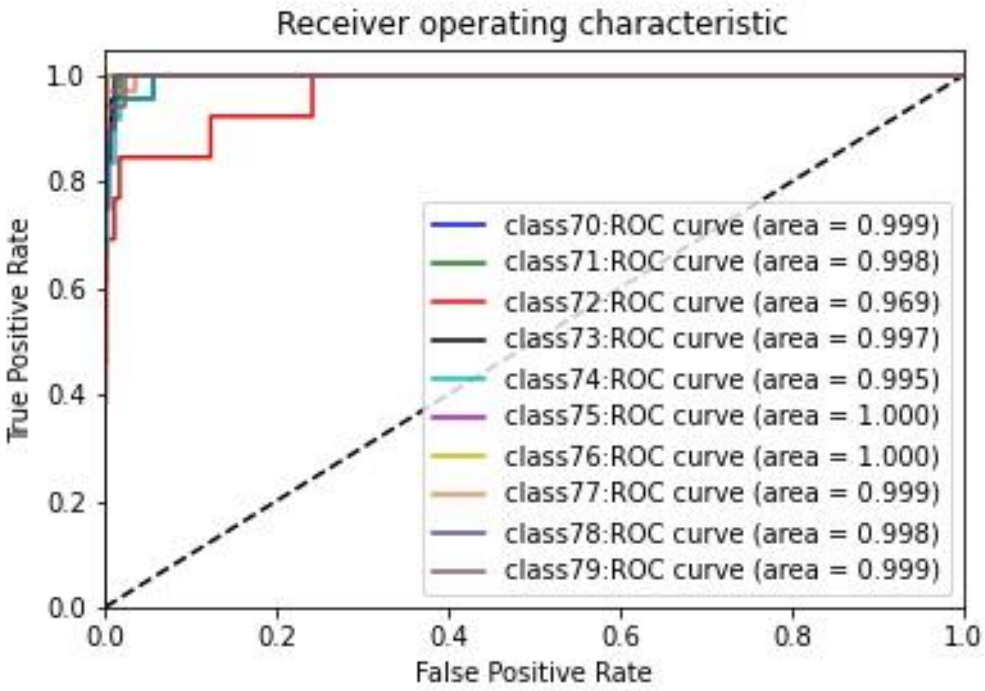

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class20:ROC curve (area = 0.999)
class21:ROC curve (area = 0.976)
class22:ROC curve (area = 1.000)
class23:ROC curve (area = 0.996)
class24:ROC curve (area = 0.995)
class25:ROC curve (area = 0.993)
class26:ROC curve (area = 0.998)
class27:ROC curve (area = 0.997)
class28:ROC curve (area = 1.000)
class29:ROC curve (area = 0.998)

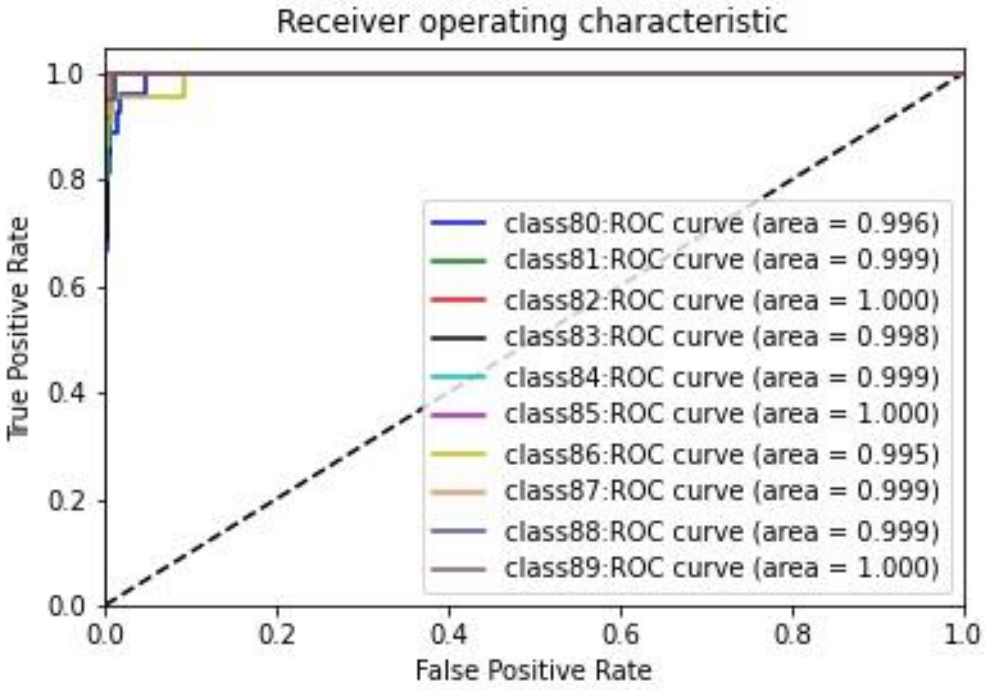

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class30:ROC curve (area = 0.999)
class31:ROC curve (area = 0.993)
class32:ROC curve (area = 0.995)
class33:ROC curve (area = 0.994)
class34:ROC curve (area = 0.999)
class35:ROC curve (area = 0.991)
class36:ROC curve (area = 1.000)
class37:ROC curve (area = 0.999)
class38:ROC curve (area = 0.974)
class39:ROC curve (area = 1.000)

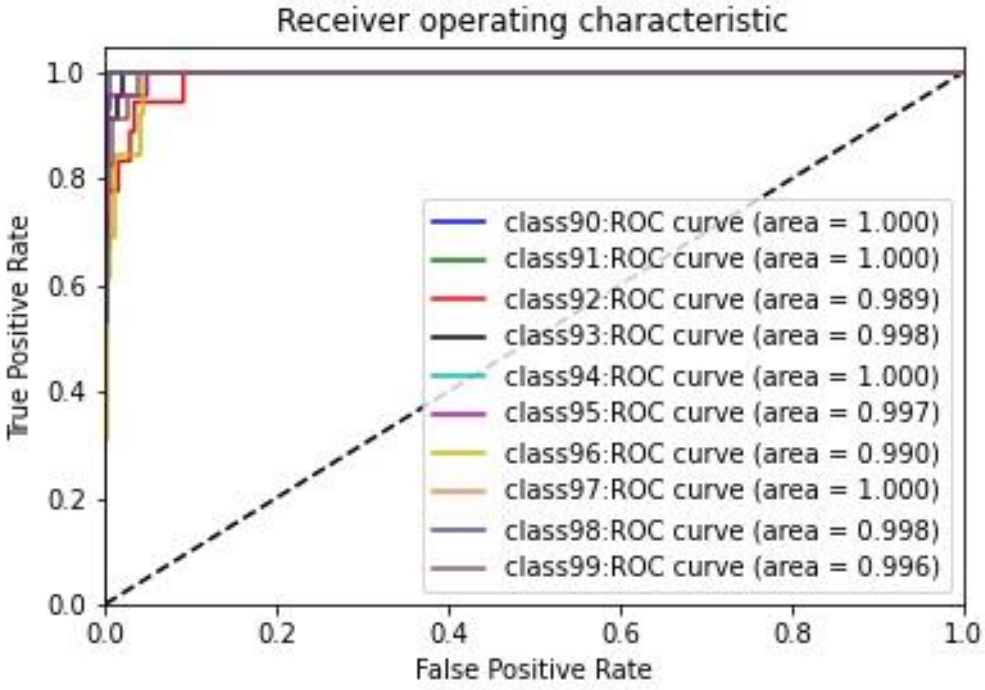

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class40:ROC curve (area = 0.997)
class41:ROC curve (area = 0.980)
class42:ROC curve (area = 0.999)
class43:ROC curve (area = 1.000)
class44:ROC curve (area = 0.999)
class45:ROC curve (area = 0.996)
class46:ROC curve (area = 0.995)
class47:ROC curve (area = 0.995)
class48:ROC curve (area = 0.999)
class49:ROC curve (area = 1.000)


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class50:ROC curve (area = 0.996)
class51:ROC curve (area = 0.991)
class52:ROC curve (area = 0.997)
class53:ROC curve (area = 1.000)
class54:ROC curve (area = 1.000)
class55:ROC curve (area = 0.992)
class56:ROC curve (area = 1.000)
class57:ROC curve (area = 0.961)
class58:ROC curve (area = 1.000)
class59:ROC curve (area = 0.996)

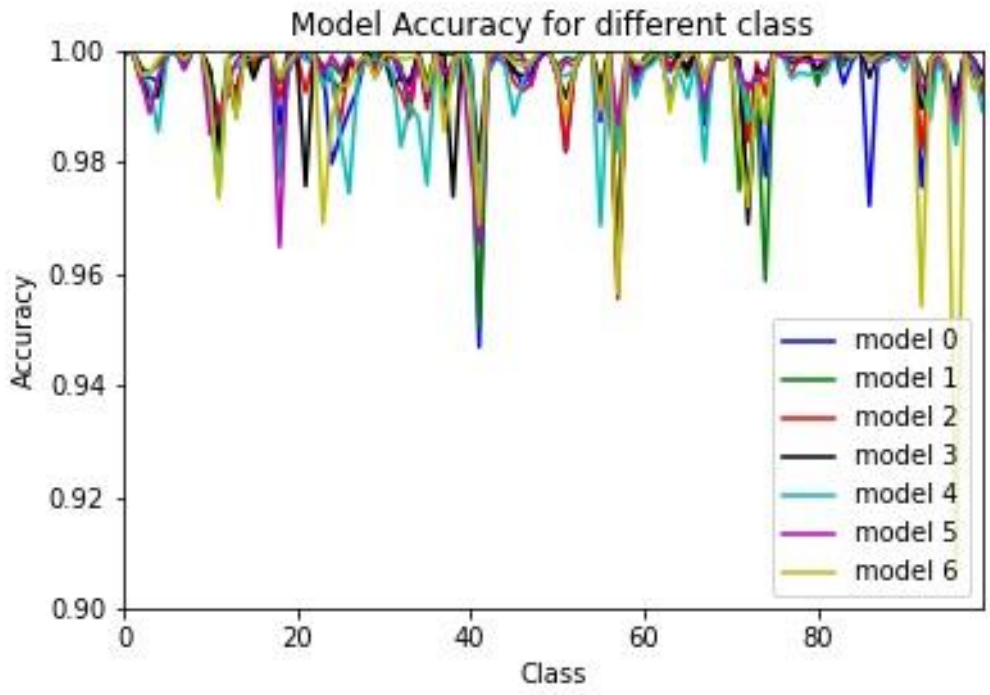

Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class60:ROC curve (area = 0.997)
class61:ROC curve (area = 0.998)
class62:ROC curve (area = 0.999)
class63:ROC curve (area = 0.996)
class64:ROC curve (area = 0.999)
class65:ROC curve (area = 0.996)
class66:ROC curve (area = 1.000)
class67:ROC curve (area = 0.994)
class68:ROC curve (area = 0.999)
class69:ROC curve (area = 0.999)


Receiver operating characteristic
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
class70:ROC curve (area = 0.999)
class71:ROC curve (area = 0.998)
class72:ROC curve (area = 0.969)
class73:ROC curve (area = 0.997)
class74:ROC curve (area = 0.995)
class75:ROC curve (area = 1.000)
class76:ROC curve (area = 1.000)
class77:ROC curve (area = 0.999)
class78:ROC curve (area = 0.998)
class79:ROC curve (area = 0.999)





各个模型在各类别的 auc 如下图：



可见也有一些比较明显的某些类别正确率低的缺陷，但也不完全重合，存在互补的可能。

6. 预测结果

从训练结果分析中已知，单一的 resnet 模型有自己的缺陷，而不同种网络还是有互补的可能。同时，从课堂听讲以及资料查阅中也了解到，resnet 的 top5 正确率很高，top1 正确率就没那么高了，所以要思考如何发挥 top5 的优势；而 top5 正确率高说明正确结果出现在比较靠前的位置的概率比较高，所以可以将不同模型在每张图片得到的输出 testout 累加，找到最终的 testout 最大值所在位置，就是预测结果。这种方法类似投票，通过此方法，进一步提高了自测集上的准确率，粗分类正确率达到了 94.25%，细分类正确率达到了 88.45%。

7. 总结与收获

这次大作业任务看似比较常见，但是真正把它做得足够好还是很有挑战性的。本次大作业收获颇多，从最开始粗分类准确率能上 70%就觉得挺不容易，到后来逐渐尝试摸索，一点点把准确率提高，再后来迁移学习拓宽思路，使得准确率又有了一次飞跃，据不完全统计，总共的有效训练尝试不下 100 次，一次次尝试中也

学到了很多新知识。另外，在做大作业的过程中发现 pytorch 的官网指导内容很完善，通过官网上的学习感觉也节省了很多时间精力, 少走了不少弯路。