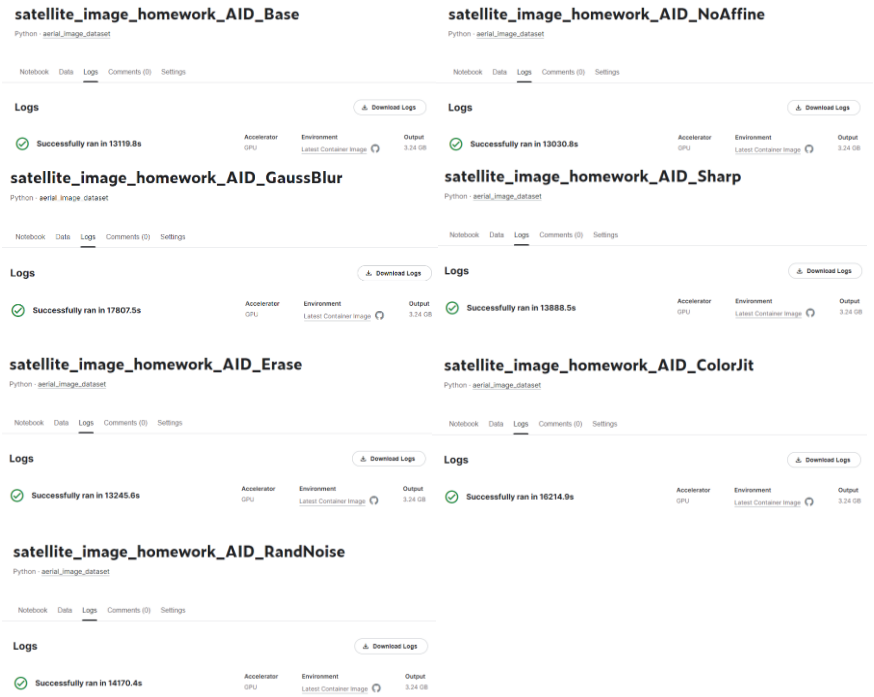
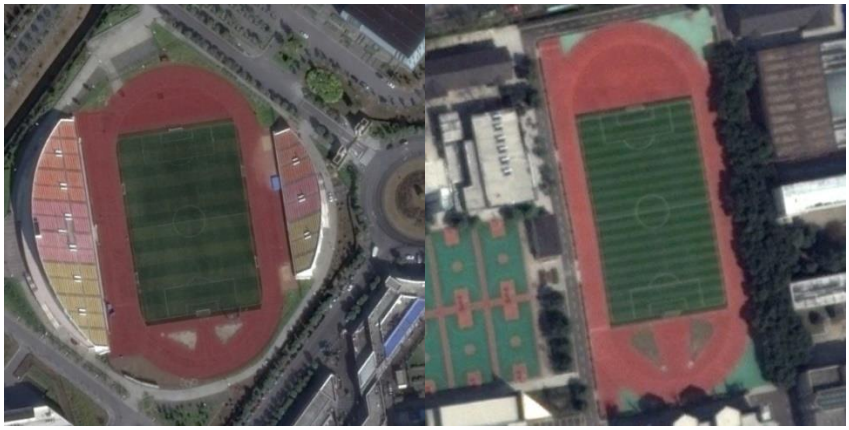
数据集介绍

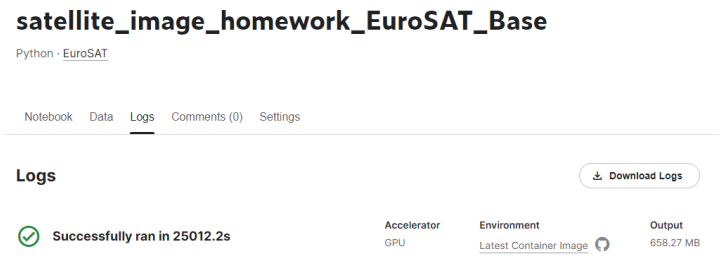
我们候选的一些数据集包括 AID（Aerial Image Dataset）、BigEarthNet-S2-v1.0、Cactus Aerial Photos、EuroSAT、RESISC45 等，这些数据集都是已经做好了标签的遥感图像，可以直接用来训练模型。他们的具体情况如下所述：

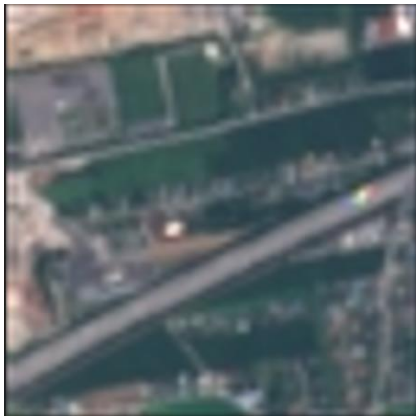
- AID
  - 图像数量：10000
  - 类别数：30
  - 波段：RGB
  - 来源：Google Earth
  - 图像大小：600×600 像素
  - 优点：
    - 数据量适中
      - 使用 Kaggle 提供的 P100 GPU，使用我们预先设定好的参数与模型训练一次大约需要 4-8 小时，速度相对而言比较快，如下所示：
      - 

    - RGB 真彩色图像，与人的视觉感受类似，能够更好的模拟人类决策过程
      - 不过，这在一定程度上也是缺点，具体请见后文
    - 图像尺寸相对较大，便于更细节的分析处理
  - 缺点：
    - 类别数过多，有些类别十分相似

- 
- 上图中，左侧为 stadium 类别的一个地物，右侧为 playground 类别的一个地物，两者之间实际上并没有特别大的差别，很容易造成误分类；
- 
- 上图中，左侧两张图片是 church 类型的地物，而右侧两张图片是 commercial 类型的地物，除了色调之外，实际上没有十分明显的差别。
- 对于这个现象，在后文的误差分析部分会加以更深层次的分析。

- BigEarthNet-S2-v1.0
  - 图像数量：59 万
  - 类别数：43
  - 波段：与哨兵二号波段数量相同，共 12 波段
  - 图像大小：因波段而不同，RGB 波段为 120×120 像素
  - 来源：哨兵二号卫星
  - 优点：
    - 数据量大
    - 来源权威
  - 缺点：
    - 数据量过大，在有限的时间内很难训练完
      - 一共有 100 多 G 的数据，已经接近了 Kaggle 的私有数据集 repo 上限；56 万张数据，按照 AID 的经验，至少需要 200 多小时训练完成。
    - 数据格式复杂
      - 使用的是 TIFF 格式的原始影像，而且没有经过彩色合成使之成为三通道图像；
      - 标签使用 JSON 格式存储，每个图像都有一个文件，读取 IO 开销大；
      - 是多标签数据集，每个图像有一个或者多个类别，训练过程与代码量较大。
    - 图幅相对较小
      - 只有 120×120 像素，有些更微观的结构难以很好的区分开，从而丢失了一部分信息。

- Cactus Aerial Photos
  - 图像数量：16000
  - 类别数：2（仙人掌、非仙人掌）
  - 波段：RGB
  - 图像大小：不定
  - 来源：多种来源的航空相片
  - 优点：
    - 数据量适中
    - 任务明确，分类很少
  - 缺点：
    - 任务过于简易，无法完整体现出 VGG 的分类优势
- EuroSAT
  - 图像数量：2.7 万
  - 类别数：10
  - 波段：RGB
  - 图像大小：64×64
  - 来源：哨兵二号卫星
  - 优点：
    - 数据量适中
      - 比 AID 数据量多两倍，但是训练时间在可以接受的范围内
      - 

    - 来源权威
      - 相比 AID 的 Google Earth，欧洲的哨兵二号卫星是更加权威的。
  - 缺点
    - 图像大小过低
      - 过低的分辨率会使得地物的一些细节丢失，从而影响识别效果，如下图所示的 highway 类别的一个地物
      - 
- RESISC45

- 图片数量：3.1 万
- 类别数：45，均匀分布，每类 700 张
- 波段：RGB
- 图片大小：256×256 像素
- 优点：
  - 数据量适中
  - RGB 真彩色图像，与人的视觉感受类似，能够更好的模拟人类决策过程
- 缺点：
  - 种类数太多，某些类别过于类似
    - 在没有使用任何增强手段的情况下，甚至出现了模型不收敛的现象，表现为准确率一直为一个稳定的很小值，loss 一直处于十分大的状态
  - 图幅相对较小，有些细节可能会丢失

数据增强方法介绍

数据增强，是指在数据分析中，向原始数据加以少量变换，从而达到增加训练数据数量、提高鲁棒性的一种方法。数据增强有很多优势，例如可以防止过拟合、增加图像识别时模型的几何不变性与空间不变性等。

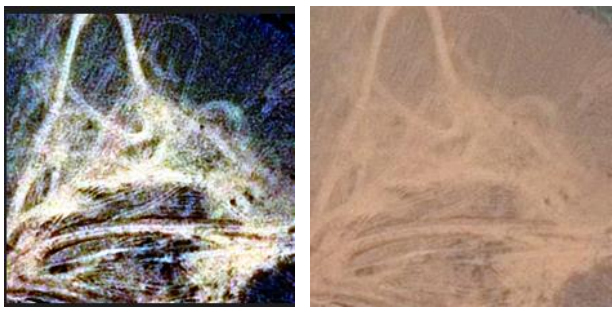
数据增强包括以下几种方法：

- 几何变换
  - 几何变换中，最有代表性的就是仿射。仿射就是给一个空间中的对象施加一次线性变换，然后再加以平移，从而变换到另一个向量空间。经过仿射之后，图像的尺寸可以变化，朝向也可以变化，从而更好的贴近现实世界中地物的复杂性和多样性。
  - 仿射的效果如下图所示：
    - 
  - 除了仿射之外，镜像变换也是十分常用的一种增强几何不变性的方式，效果如下图所示：
    - 
- 颜色变换
  - 颜色变换可以通过调整亮度、饱和度、对比度、色调来实现，其可以将物体原来的色彩加以适当的改变，从而提高模型对于不同颜色的同一类地物的识别能力。

- 颜色变换的效果如下所示：
  - 
- 随机噪声
  - 根据相关的研究，随机噪声可以增强模型的准确性与鲁棒性。遥感成像过程中也会有许多随机噪声，例如 CMOS 成像单元之间的差异、大气湍流等的干扰。通过添加随机噪声可以在一定程度上提高模型所具有的识别能力。
  - 随机噪声的效果如下所示：
    - 
- 滤波
  - 滤波可以令图像变得更为锐利或者平滑，也可以用于边缘提取等操作之中。图像平滑可以模拟由于对焦误差、运动、湍流等造成的模糊，而图像锐化则可以增强图像的边缘细节，提高模型对边缘的识别能力。
  - 平滑滤波的效果如下所示（高斯滤波）：
    - 
  - 锐化的效果如下所示：
    - 
- 随机擦除
  - 随机擦除就是将图像中的一部分区域去除。由于一幅遥感图像中可能会有除了主要

地物之外的其它地物，同时主要地物也有可能没有完全位于范围内，进行随机擦除可以降低不相关地物之间的错误联系，以及提高对于主要地物部分区域的识别能力。

- 随机擦除的效果如下所示：
  - 

- 直方图均衡化
  - 直方图均衡化是在遥感影像处理中十分常用的一种方法，其可以将原本聚集在一起的灰度级均匀拉伸到整个图像灰度级上，从而增强几何效果。
  - 在数据预处理中，直方图均衡化之后模型出现了不收敛的现象，表现为 loss 一直保持很大的值，而准确度随着 epoch 增加没有提升，具体原因请见效果：
    - 
    - 即使是人，也难以通过均衡化后的图像识别出这是裸地，地物经过均衡化之后与原始特征差别过大。
- 除了上述内容之外，数据增强还有图像混合、平移、缩放、主成分分析、风格转换等。

数据增强部分参考文献与资料：

- Shorten, Connor, and Taghi M. Khoshgoftaar. "A survey on image data augmentation for deep learning." Journal of Big Data 6.1 (2019): 1-48.
- Jung, A. "Image augmentation for machine learning experiments." (2017).
- Berthelot, David, et al. "Mixmatch: A holistic approach to semi-supervised learning." arXiv preprint arXiv:1905.02249 (2019).
- Zhong, Zhun, et al. "Random erasing data augmentation." Proceedings of the AAAI Conference on Artificial Intelligence. Vol. 34. No. 07. 2020.
- Pytorch 官方文档，transforms 部分

优化方法介绍

Epochs

对于不同的 epoch，其训练过的数据数量不尽相同；由于学习率、函数收敛速度的限制，往往对于一个数据集，需要训练很多次才能够到达误差极小值的位置，达到最好的识别效果。

在训练过程中，记录不同的 epoch 对应的准确率，可以反应训练次数与模型完备程度的关系。
根据实际经验，epoch 最大为 100 时，往往模型准确率就不会继续改变，loss 也稳定在很小的一个范围内。因此，在研究优化参数时，将最大 epoch 设置为 100，同时分析不同的 epoch 处对应的准确率差别。

Batch size

一般而言，如果训练一幅图像就进行一次梯度回归，会造成速度上的下降。单张图片所计算出的梯度值可能很小，但是回归时需要对大量参数进行修正，从而增添很多额外的开销。将 batch size 增大可以通过很多幅图像进行一次梯度回归，从而极大提高计算速度。
不过 batch size 的增加也有可能使得参数的变化不稳定，难以收敛到最优解，甚至直接跳跃到另一个局部最优解上去，因此有必要对 batch size 的影响加以探究。本次实验中选取了三种 batch size：16、32 和 128 来进行分析。

训练具体过程

VGG 对 EuroSAT 的识别

得益于我们的标准化代码，几乎所有参数都通过变量来输入，而没有硬编码进代码内，因此我们只在 VGG 识别 AID 的代码上修改了一下数据集的路径，就得到了对 EuroSAT 的处理结果。训练时使用的数据增强方法如下：

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                             transforms.RandomHorizontalFlip(),
                             transforms.RandomVerticalFlip(),
                             transforms.RandomAffine(degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                             transforms.ToTensor(),
                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

首先是变换大小，然后水平、垂直翻转，之后仿射，之后转换为张量进行计算。
其它参数，如最大 epoch 数、学习率、优化器等，均与 VGG 识别 AID 的过程一样，请参见上文（我希望夏罗生写了）

ResNet 对 EuroSAT 和 RESISC45 的识别

得益于标准化的代码，我们也只是改了几个路径参数就实现了 ResNet 的训练。数据预处理方法与 ResNet 识别 AID 一样，其他如学习率、优化器等参数也都相同。

数据增强过程

我们在本次实验中，尝试了如下的几种数据增强方法的效果：仿射、颜色变换、图像平滑、图像锐化、随机擦除、随机噪声和直方图均衡化，其中随机噪声和直方图均衡化因为变换对

图像的影响过大，造成了空间不连续性，模型训练时出现了不收敛的现象。以下是这些模型训练时的增强方法。

- 仿射
  - 由于仿射是在基准代码中就存在的一个变换，因此我们新建了一个没有仿射的训练代码来进行测试。两者比较如下：
  - （有仿射，即基准代码）

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                             transforms.RandomHorizontalFlip(),
                             transforms.RandomVerticalFlip(),
                             transforms.RandomAffine(degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                             transforms.ToTensor(),
                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

  - （无仿射）

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                             transforms.RandomHorizontalFlip(),
                             transforms.RandomVerticalFlip(),
                             transforms.ToTensor(),
                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

- 颜色变换

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                transforms.RandomHorizontalFlip(),
                transforms.RandomVerticalFlip(),
                transforms.RandomAffine(
                    degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                transforms.ColorJitter(
                    brightness=0.1, contrast=0.1, saturation=0.1, hue=0.1),
                transforms.ToTensor(),
                transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

- 图像平滑
  - 具体的图像平滑过程中，使用的是高斯模糊

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                             transforms.RandomHorizontalFlip(),
                             transforms.RandomVerticalFlip(),
                             transforms.GaussianBlur(5),
                             transforms.RandomAffine(degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                             transforms.ToTensor(),
                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

- 图像锐化

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                             transforms.RandomHorizontalFlip(),
                             transforms.RandomVerticalFlip(),
                             transforms.RandomAffine(degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                             transforms.RandomAdjustSharpness(8),
                             transforms.ToTensor(),
                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

- 随机擦除

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                             transforms.RandomHorizontalFlip(),
                             transforms.RandomVerticalFlip(),
                             transforms.RandomAffine(degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                             transforms.ToTensor(),
                             transforms.RandomErasing(),
                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

- 随机噪声
  - 由于 pytorch 没有附带随机噪声的内置变换，因此这里使用了 Lambda 变换生成器，通过生成一个随机张量，并且与原图像相加的方式来增加一些噪声

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                transforms.RandomHorizontalFlip(),
                transforms.RandomVerticalFlip(),
                transforms.RandomAffine(
                    degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                transforms.ToTensor(),
                transforms.Lambda(
                    lambda x: x + 0.05*torch.rand(3, 224, 224)),
                transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

- 直方图均衡化

```
"train": transforms.Compose([transforms.Resize((224, 224)),
                             transforms.RandomHorizontalFlip(),
                             transforms.RandomVerticalFlip(),
                             transforms.RandomAffine(degrees=(0, 180), translate=(0.1, 0.3), scale=(0.5, 1)),
                             transforms.RandomEqualize(),
                             transforms.ToTensor(),
                             transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))]),
```

优化参数探究过程

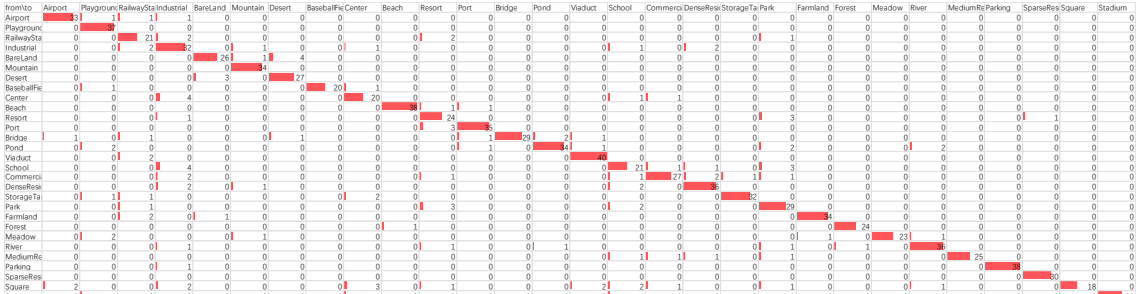
对于 batch size，我们通过修改参数来进行探究，具体如下：

`batch_size = 16` `batch_size = 128` `batch_size = 32`

研究结果

VGG 识别 AID 数据集的误差矩阵分析

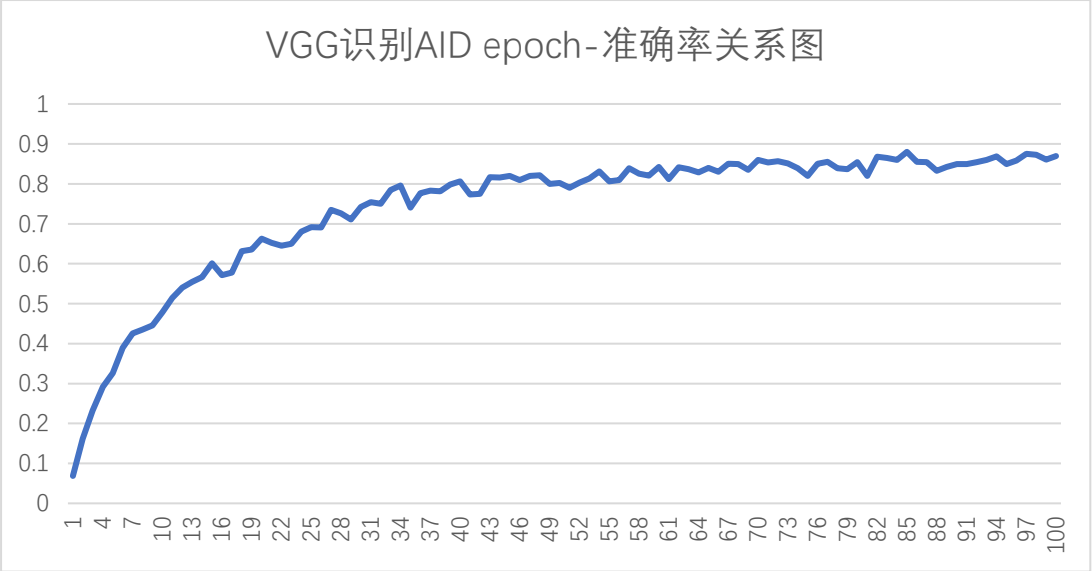
误差矩阵可以得到不同类别图像之间的误分类情况，这些误分类结果一方面可以表现模型对不同类别区分的完备程度，另一方面也表现了数据集中不同类别地物的相近程度：越相似、越难互相区分的地物越容易被误分类。以下就是 VGG 识别 AID 的误差矩阵：



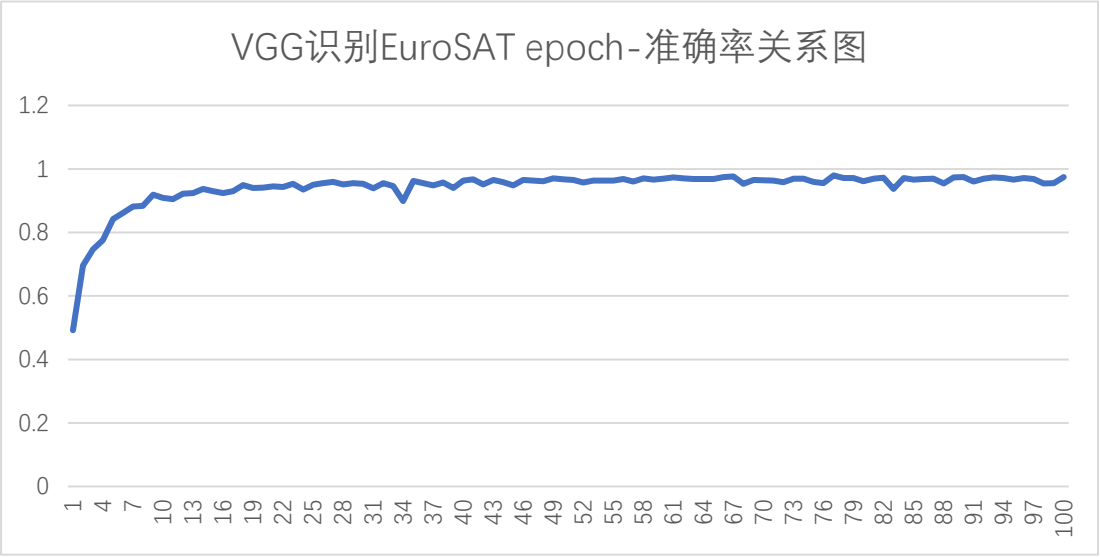
总体而言，AID 中各类别的相似程度并不很高，误差矩阵中没有出现大量的误分类情况，在此挑选几个典型来进行分析。BareLand 和 Desert 被多次误分类，根据实际经验，裸地和沙漠在地物特征上确实有着很多的相似之处，被误分类也是意料之中的。Resort、School 和 Park 也容易出现误分类的情况，度假胜地处往往也有着丰富的娱乐设施，学校有操场、人工修整的树木等景观，这和公园的特征有一定的相似之处。

不同模型，不同数据集的 Epoch 与准确率关系

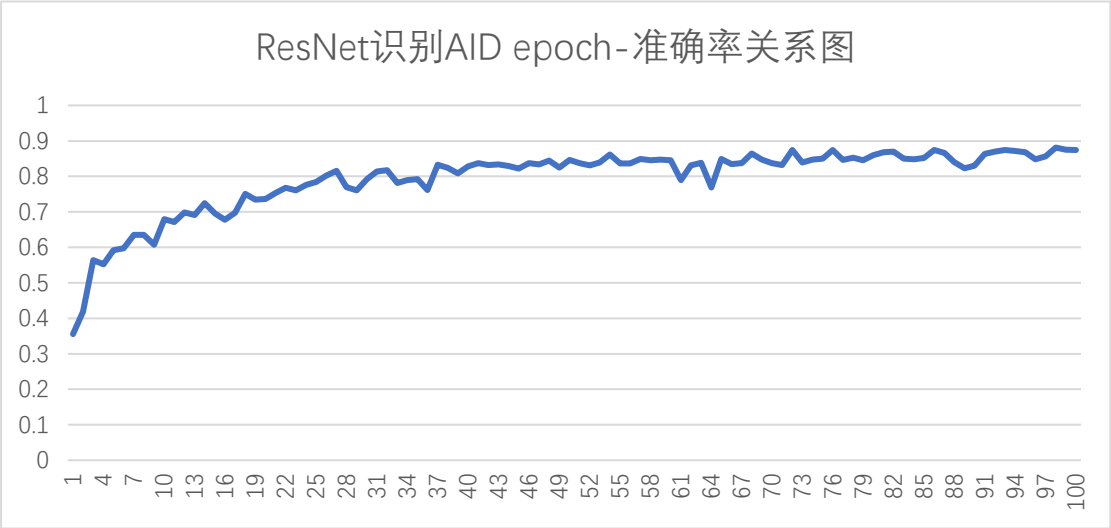
在研究中，最直观也是最简单的研究内容就是 Epoch 与准确率的关系，准确把握这个规律，就可以恰当的设置迭代次数，从而节约算力和时间。
在 VGG 识别 AID 的基准训练过程中，两者关系如下：

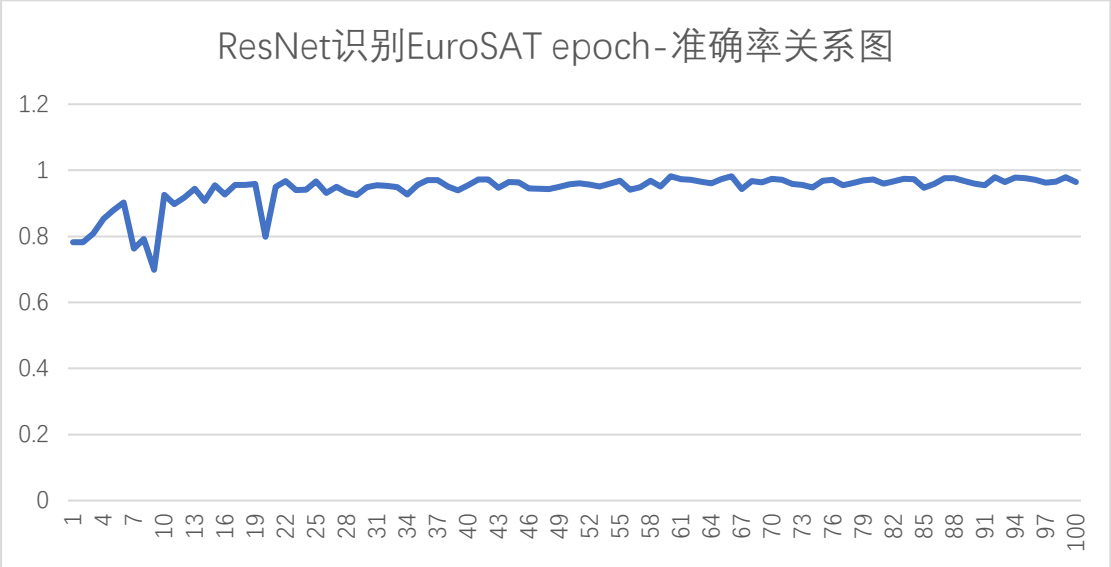


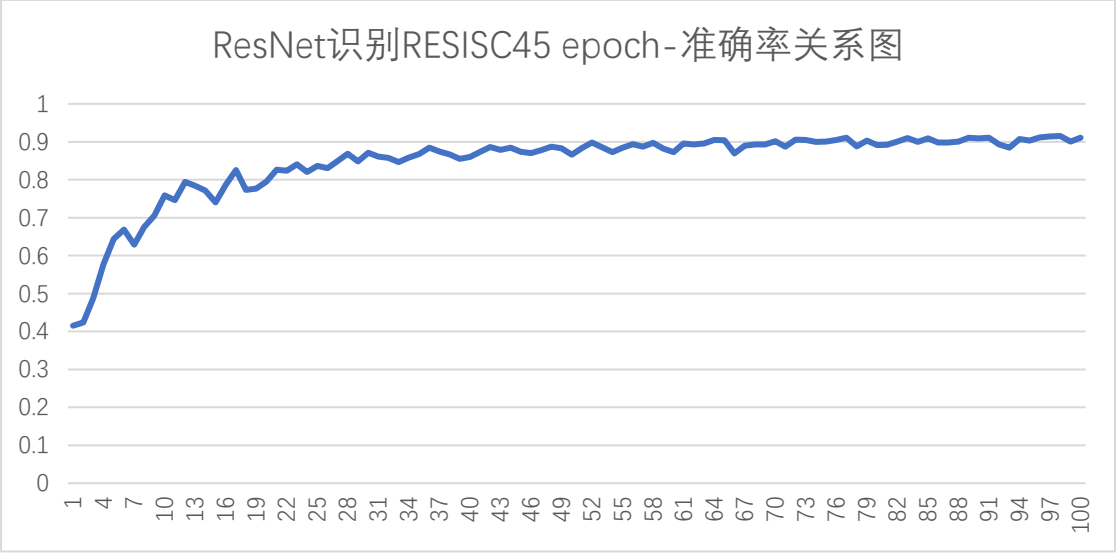
可以看到，当 epoch 到达 50 之后，准确率的提升就十分缓慢了，到达 70 之后准确率达到稳定状态，不再继续变化。
VGG 对 EuroSAT 的训练中，趋势也大致相同，如下所示：



由于 EuroSAT 的种类相对较少，图像的尺寸小，收敛所需的时间相对较低，因此仅 16 个 epoch 就能够令模型的准确率达到 95%的水平。
除了 VGG 模型之外，我们还试验了 ResNet 对 AID、EuroSAT、RESISC45 这三种数据集的识别效果，其中 Epoch 与准确率的关系如下所示：

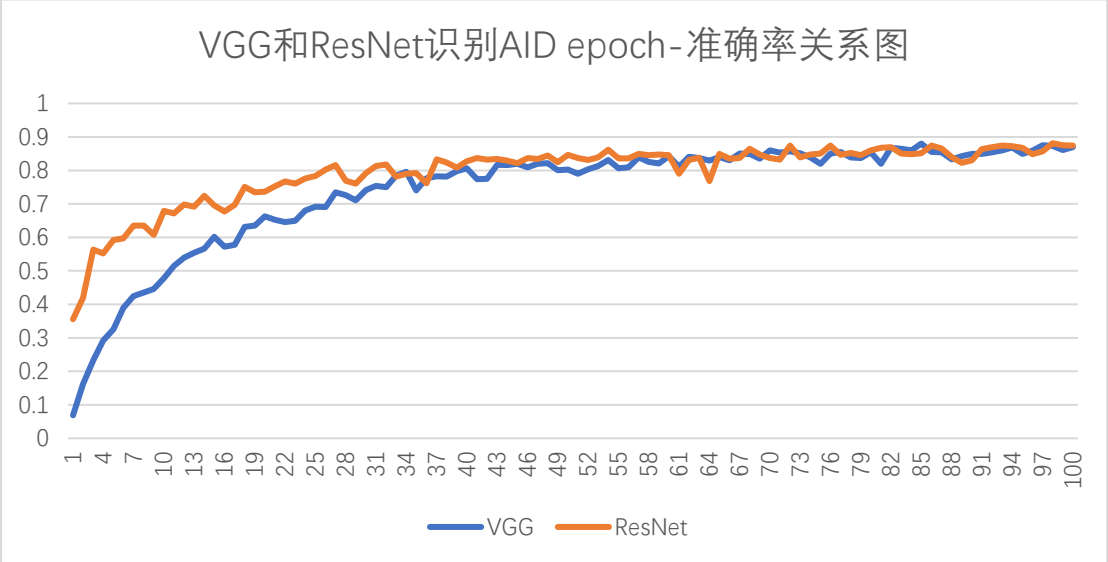






可以看到，虽然 ResNet 达到最终稳定所需的 epoch 数与 VGG 相差不多，但是其初次训练后的准确率相当高，甚至 EuroSAT 初次训练后就达到了 80%的水准。比较高的初次训练准确率表明其从初始化的混沌状态到接近局部最小值的速度很快，而达到稳定所需的时长与

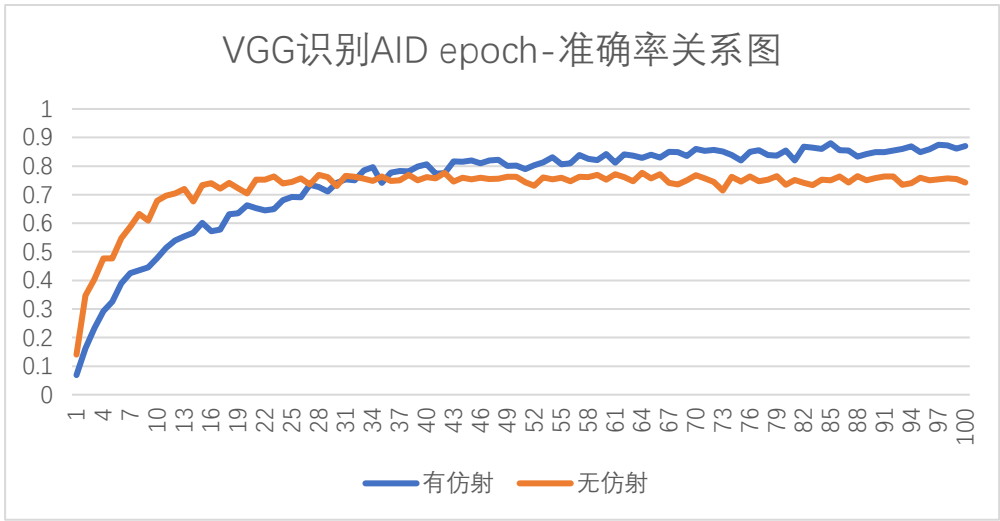
VGG 相差不多，则表明其在局部最小值附近的梯度比较和缓，不容易发生突变。总体而言，ResNet 的收敛性能比 VGG 要好许多。更直观的比较，请见下图所示：



除了这些差别之外，可以注意到，上面几幅图中缺少了 VGG-RESISC45 的关系图，这是由于训练过程中模型无法收敛所致。RESISC45 的类别数比较大，相近类别的特征差别较小，VGG 无法准确区分开这些差别，从而难以找到局部最小值，而 ResNet 则能够区分开这些差别，甚至达到了 90%的精度。

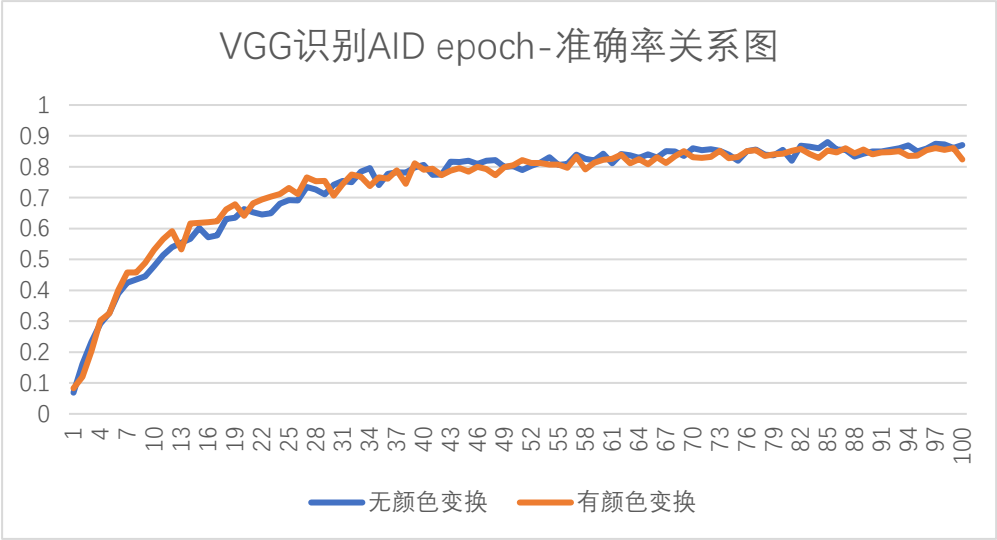
不同数据增强方法的 Epoch-准确率关系

- 仿射

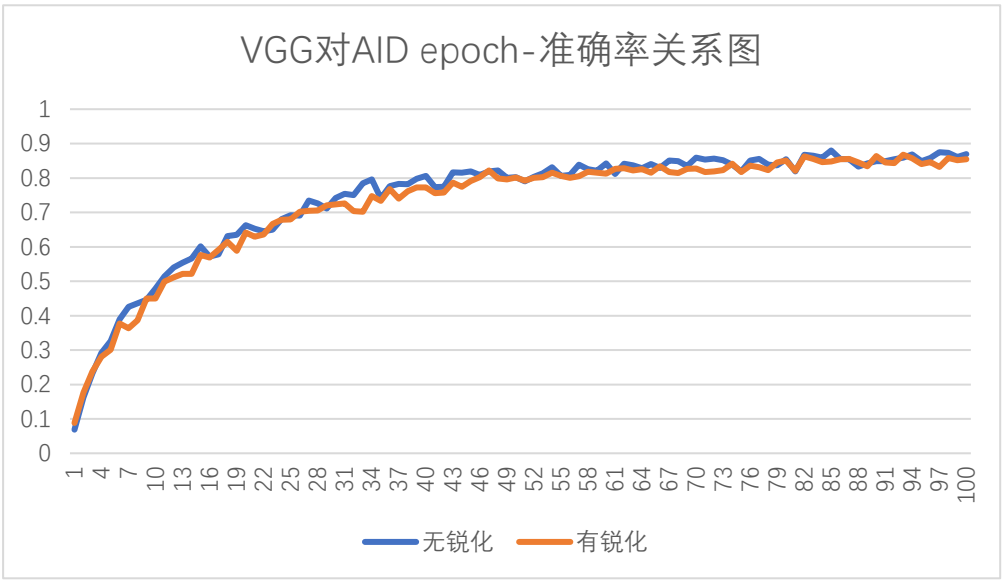


  -
  - 没有仿射时，准确率随 epoch 的上升速度比增加仿射时的要快，但是最终达到的准确率要比有仿射的低接近 10%。
  - 经过讨论，我们认为这是由于仿射所带来的空间位移和遮掩效果引起的。一方面，仿射引起的遮掩会使得数据中主要地物相对显示不全，从而模型无法一次性获得地物的全部信息，需要多个 epoch 才能对一幅图像中全部的信息加以完整获取，导致初始上升速度较慢；另一方面，这种遮掩、旋转和缩放会提高模型对于不同空间位置、几何构型的地物的识别能力，从而使得最终的准确率达到比较高的水平。

- 颜色变换

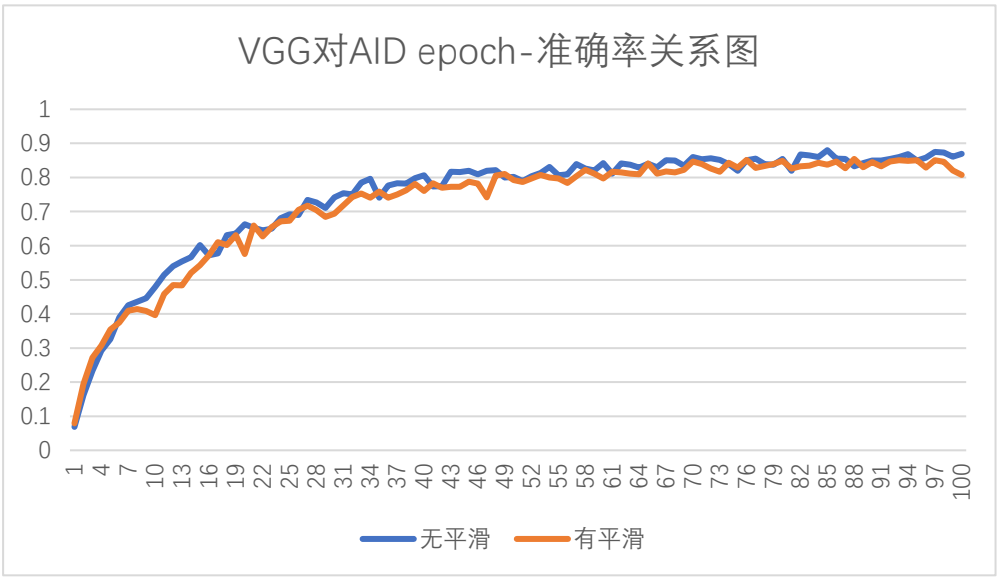


  -
  - 颜色变换对 VGG 识别效果没有十分明显的影响，这是由于数据集的特征导致的。实际上，遥感影像中同一类地物的颜色并不是完全一致的，而是有着天然的差别。这种差别已经起到了类似于颜色变换的效果，足以提高模型对颜色差异的鲁棒性。这种差异具体的实例如下所示：
    - 
    - 
    - 
- 图像锐化



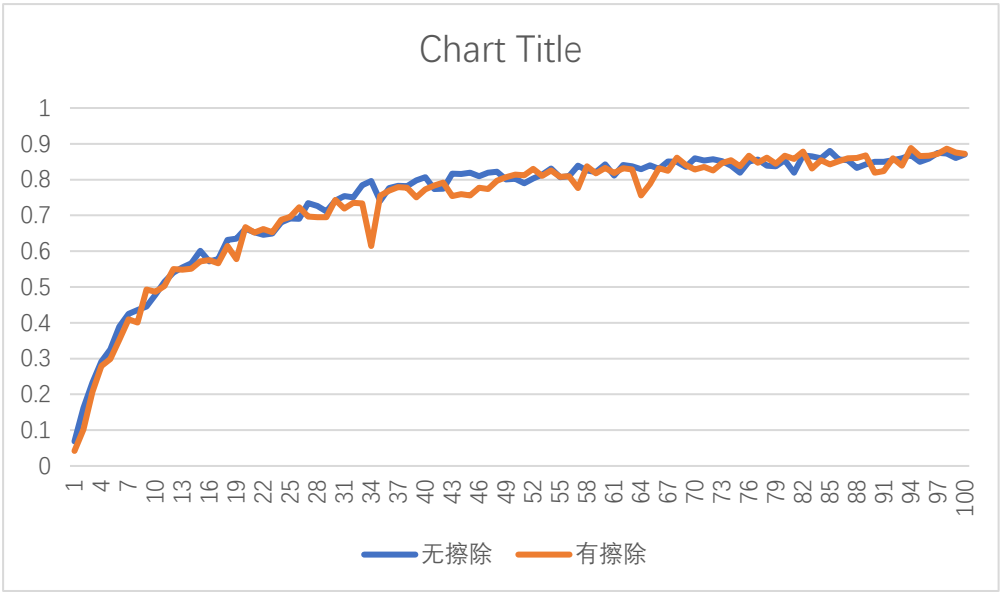
-
- 可以看到，锐化实际上对于准确率也没有太大的影响。VGG 的前几层为卷积+池化层，这几层具有提取地物边缘和特征几何信息、识别特殊构型的功能，因此锐化与否并不会显著影响模型的识别能力。

- 图像平滑



-
- 与锐化一样，平滑也对准确率没有特别大的影响，其原因实际上和锐化基本一样：VGG 的前几层具有提取特征的能力，模糊处理并不能显著改变这些特征的形态，因此不会明显影响准确率。

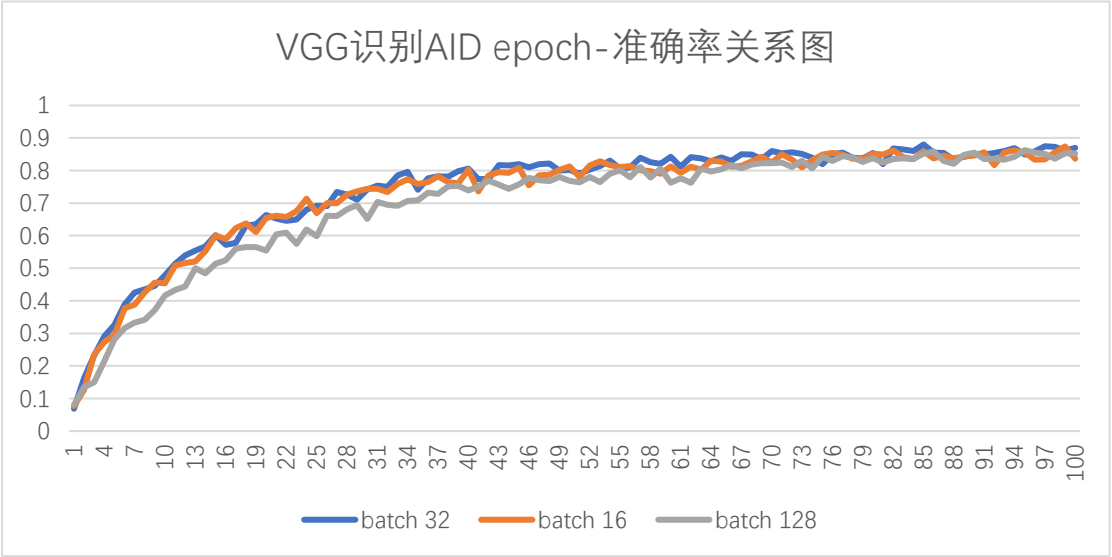
- 随机擦除



- 擦除的作用实际上与仿射有很大的重复，在一定概率下，仿射可以把图像的部分区域遮掩（如下所示），从而达到与擦除类似的效果。

- 随机噪声、直方图均衡化
  - 这些预处理方法对特征的改变较大，训练时出现了不收敛的现象，准确率表现为一直稳定在 10%以下，在此不再继续呈现这些数据。

不同 batch size 的 epoch-准确率关系



在我们所拥有的 GPU 的显存允许的范围内，不同的 batch-size 对训练的最终效果影响主要表现在每个 epoc 结束时的增量上。对于 batch-size 为 128 而言，由于单次步进的幅度较大，向局部最优解前进时可能会产生很大的扰动，从而使得接近局部最优解的速度较慢；而其它的 batch-size 因为单次梯度回归步进幅度小，因此能以比较稳定的路线前往局部最优解。除此之外，128 的 batch-size 在每个 epoch 结束时准确率波动相对较大，出现了很多上下起伏的趋势，而其他两个则比较平滑。

简易应用尝试：制作遥感图像分类平台

运用 python 丰富的库，可以很方便将 pytorch 的模型通过 Web 暴露给用户使用。简单地通过 WebSocket 传输图像数据，在服务器接收到图像后使用 net.eval()模式进行分类识别，然后再通过 ws 发送给用户，就可以实现分类功能。以下是几张我们平台的截图：