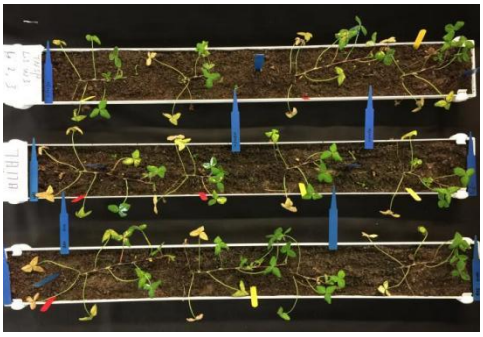
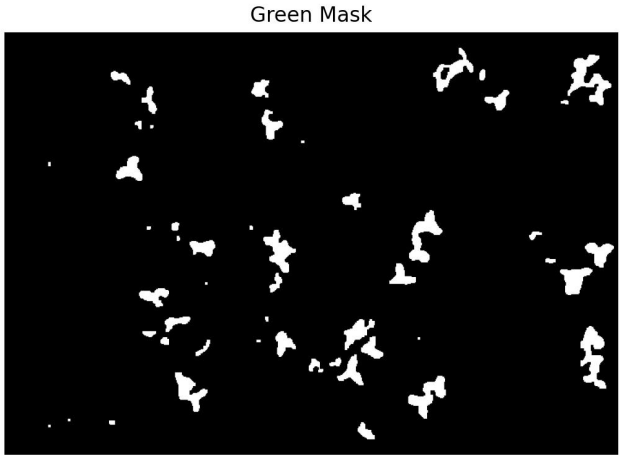
这里用 python 来完成作业 1-8，这里用的是 pycharm 作为 ide，py 版本为 3.9.20。

一、提取图中的植物部分，并估算植物的绿色部分面积，已知植物生长的槽的宽是 20 cm。



读取图像，进行空间颜色转换，定义绿色阈值，生成绿色掩码，去除噪声，通过宽度和图像宽度来计算像素和实际距离的比例求出绿色像素的面积



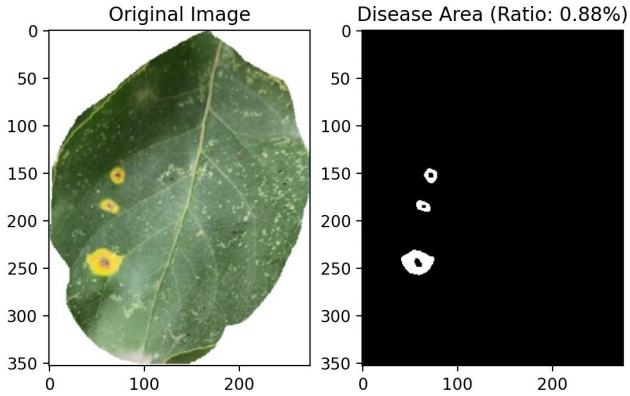
绿色面积为 11.54 平方厘米

具体代码详见 work1.py

二、提取图像中的叶片病害的图斑数目、估算病害图斑占叶片的总面比例。

进行图像读取和颜色空间转换，病害区域读取，掩膜优化，通过计算图斑数量，通过灰度图片计算叶片的面积，用病害还面积与叶片面积比值算出病害区域的面积，作为病害区域面积计算病害面积占比：

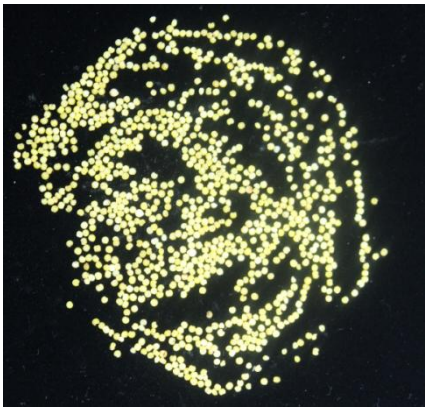
(病害面积 / 叶片总面积) × 100%



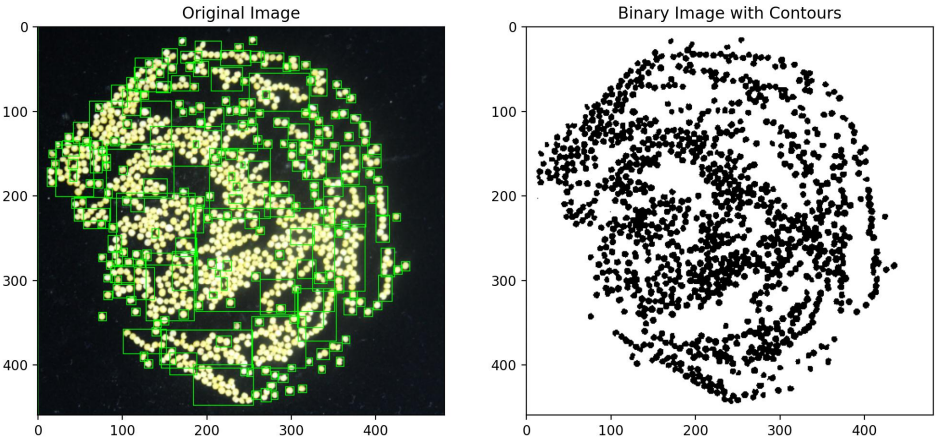
比例为 0.88%

具体代码详见 work2.py

三、采用图像形态学获取图像中小米的粒数、每粒小米的最大投影面积或者外接最大正方形的长与宽。



读取图像并转换为灰度图，通过固定阈值或自适应阈值，将图像转化为二值图，突出小米颗粒并抑制背景，使用 cv.findContours 检测小米颗粒的轮廓，遍历轮廓，计算每颗颗粒的面积，过滤小面积噪声，在原图上绘制外接框，以便直观验证结果。



数量为 313

具体代码详见 work3.py

四、分割获取谷子的各叶片，即把每个叶片独立分割，并用不同颜色表示。

读取输入图像并转化为灰度图像，通过图片去噪，减少噪声对边缘检测的干扰，使用 Canny 算法检测图像中的边缘，使用 cv.findContours 获取每片叶片的轮廓。直接展示结果。

具体代码详见 work4.py

五、还原图中被部分遮挡的苹果

遮挡苹果　　还原苹果

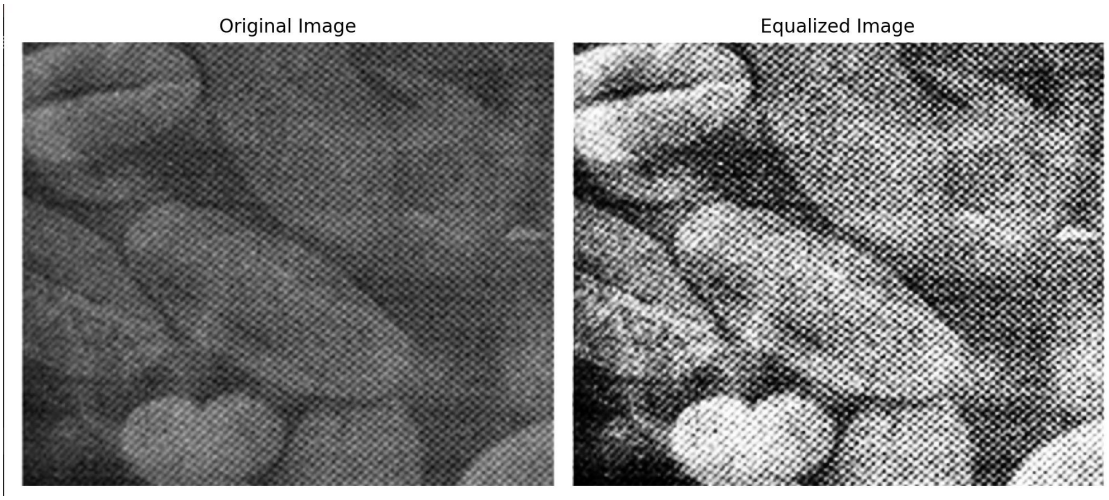
读取目标图像并显示，创建窗口并初始化掩码，设置鼠标回调函数，设置死循环，用鼠标左键画出需要修复的图像区域，按 i 或者空格进行修复，使用修复算法 INPAINT_TELEA 来进行平滑处理

具体代码详见 work5.py

六、采用直方图均衡方法处理图像 A，结果类似图 B。

图 原始图（左图）和直方图均衡方法处理后图像（右图）

读取图像，用直方图均衡化增强图像对比度，将图像的像素灰度值分布拉伸，展示结果



具体代码详见 work6.py

这里使用keras2的库, 用tensorflow来进行训练, keras为2.10.0, tensorflow 为 2.10.0，均使用 cudnn 进行加速，其余的库版本，详见 requirements.txt

七、利用 LSTM 模型预测一个时间序列数据的未来趋势。

这里我选择了收集了 1990 年至 2016 年 8 个国家的每日汇率数据。

数据集没有丢失值，每 1D 的数值颗粒度，数据时间段为 1990/01/01 00:00—2010/10/10 00:00，共 7588 条数据，包含澳大利亚、英国、加拿大、瑞士、中国、日本、新西兰和新加坡等 8 个国家的每日汇率值。

数据集使用.csv 格式保存，包含 9 个数据字段，具体如下：

date：1D 颗粒度的时间戳

0：澳大利亚汇率

1：英国汇率

2：加拿大汇率

3：瑞士汇率

4：中国汇率

5：日本汇率

6：新西兰汇率

OT：新加坡汇率

优势：标准 RNN 在反向传播长序列时，梯度会随着时间步指数衰减或爆炸，导致模型难以学习长期依赖。LSTM 的细胞状态通过直

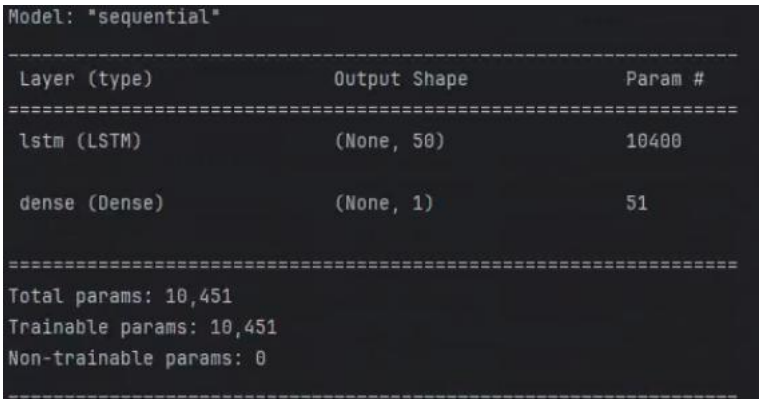
接传递和门控机制，允许梯度直接流过多个时间步，大大缓解了梯度消失问题。能够同时捕获长短期依赖，特别适合处理时间序列数据中的动态模式和复杂关系。

实际代码：

具体看代码 work7.py

数据预处理的功能由 read_and_preprocess 实现

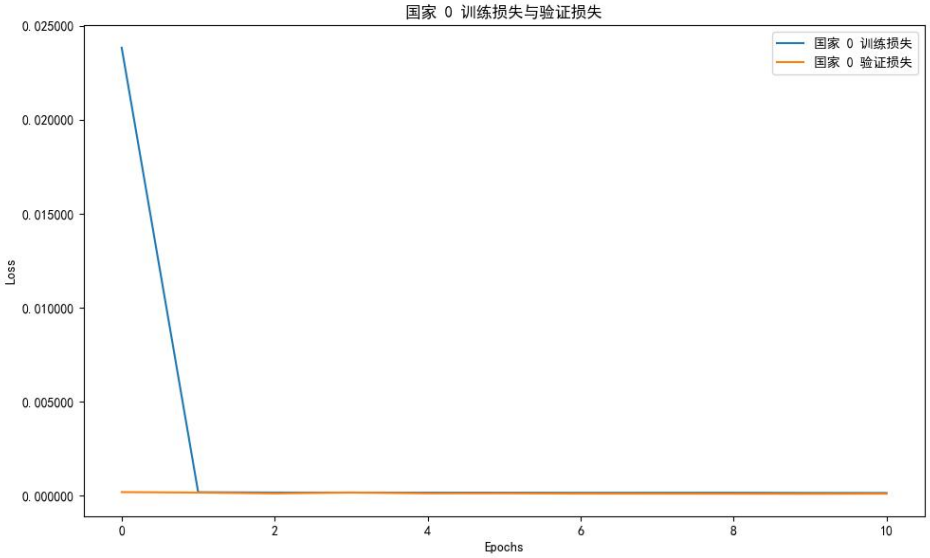
LSTM 模型构建由 build_lstm_model 来实现

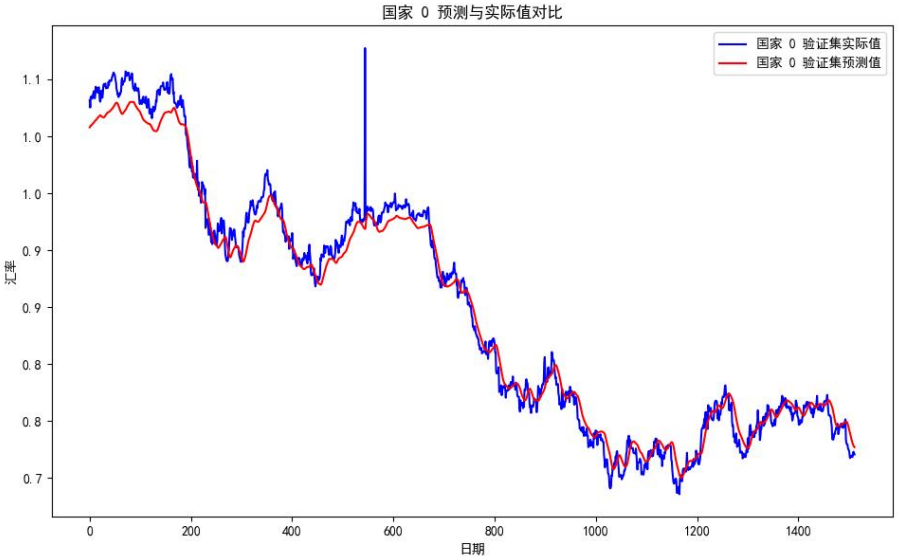


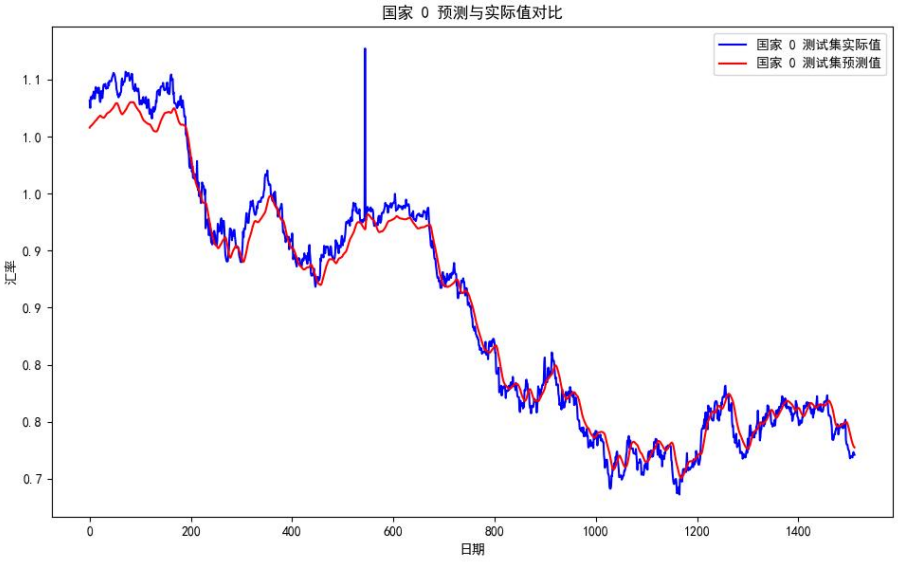
模型训练：这里使用 keras 自带的函数，定义了学习率衰减和早停机制。用的是 model.fit

训练过程记录功能是由 keras 的 model.fit 函数自带的

预测结果


国家 0 训练损失与验证损失
国家 0 训练损失
国家 0 验证损失
Loss
Epochs


国家 0 预测与实际值对比
国家 0 验证集实际值
国家 0 验证集预测值
汇率
日期


国家 0 预测与实际值对比
国家 0 测试集实际值
国家 0 测试集预测值
汇率
日期

总体分析：数据的总体趋势是对的，但是很多点的仍然不准确。仍需对模型继续改进。

八、基于迁移学习实现一个图像分类任务，例如：使用预训练模型（如 ResNet、VGG）对新的小型图像数据集（如猫狗分类）进行分类。

这里我选择用 VGG16 来训练。

迁移学习是一种机器学习方法，它将一个任务上训练得到的知识应用到另一个相关任务上。通常情况下，迁移学习通过使用在大型数据集（如 ImageNet）上预训练的模型，并将其应用于新的、数据较少的任务上，显著提高新任务的学习效果。

迁移学习非常适合数据量较小的场景，因为预训练模型已经在大量数据上学习了有效的特征，可以有效地帮助新任务，即使新任务的数据较少。

当目标任务的数据样本较少时，直接训练一个深度学习模型往往会导致过拟合。而迁移学习能够有效避免这一问题，因为预训练模型已经在大规模数据上学到了一些通用的特征。

这里选择的数据集是 CIFAR-10，keras2 目前可以直接加载 CIFAR-10 的数据，但是每次要进行在线下载，所以我直接本地读取了数据。

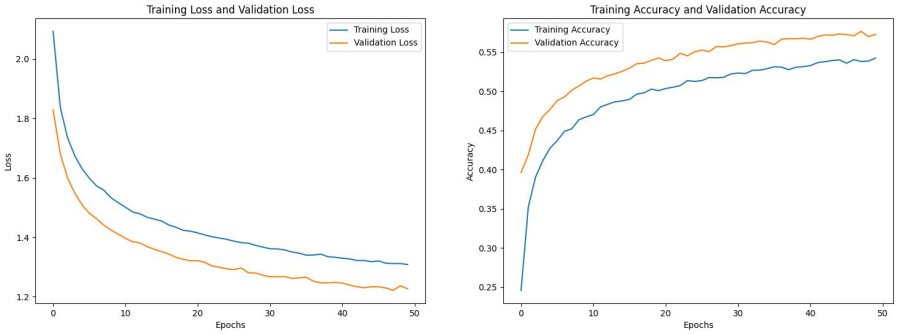
这里还是用 keras2 来进行编写代码。

详细代码见 work8.py

数据加载和预处理代码在 load_and_preprocess_cifar10 函数里。

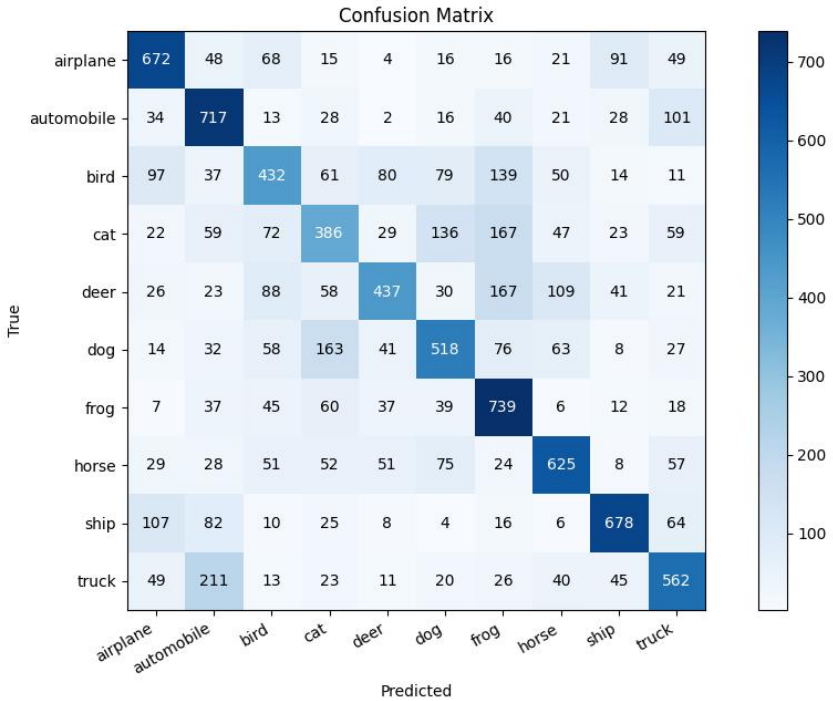
这里的模型构建使用的是 Keras 自带的 VGG16 模型。

我们这里使用了数据增强，函数为 creat_data_generator,使用 build_vgg16_model 构建 vgg16 模型并定义新顶层，训练模型由 train_model 来实现，评估模型由 evaliate_model 来实现

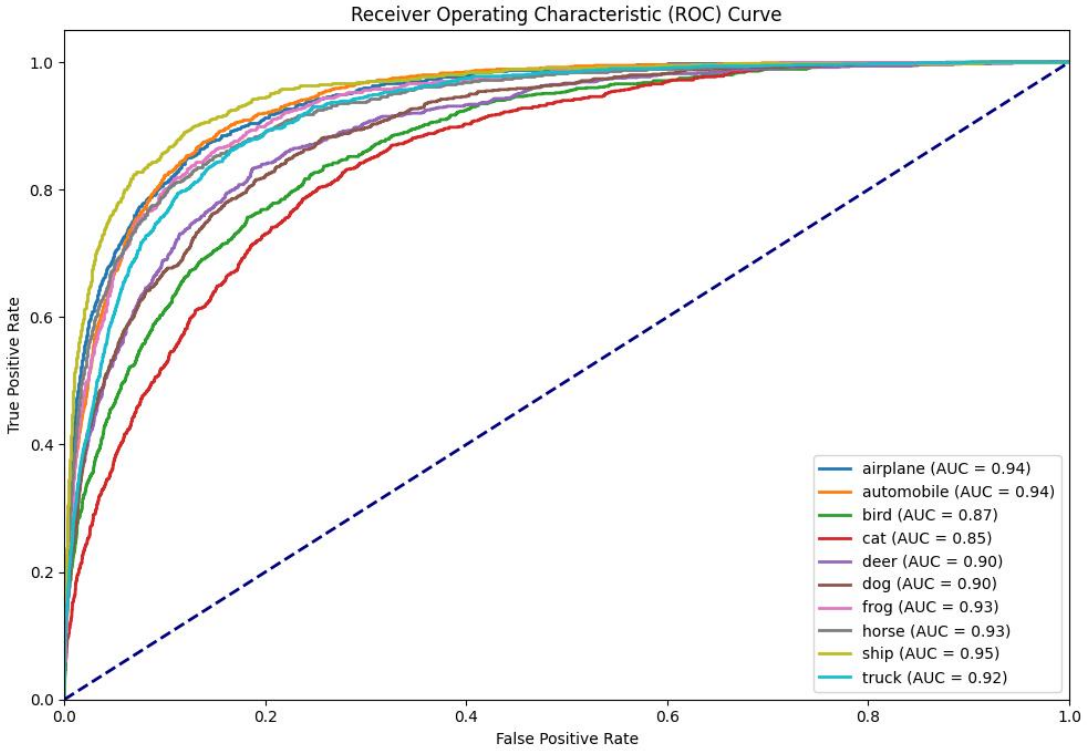


训练时间是：6.83 分钟

这是训练过程中的训练 ac 和 loss 以及测试的 ac 和 loss，训练过程稳定。



这是混淆矩阵图，其中可以发现分类出现有问题的数量大多数都是猫狗和汽车和卡车之间的分类等。



这是 ROC 曲线的结果。

数据增强已经写了，所以就不作为优化方向的选择了。

使用 VGG 进行微调，虽然当前冻结了 VGG16 的卷积层，微调部分卷积层可能会进一步提高模型在 CIFAR-10 上的性能。微调通常通过解冻最后几层进行。可以尝试使用其他优化器（如 Adam 或 RMSprop），以提高模型的收敛速度和准确度。