Identify the category of foliar diseases in apple trees

刘芳新

中山大学 智能科学与技术 21312482

Abstract

本课程设计报告探讨了Kaggle举办的FGVC8(CVPR2021-workshop)Plant Pathology-2021数据集分类任务。通过将标签进行独热编码转化为多标签分类问题，我们研究了两种卷积神经网络模型来解决本分类任务。本报告分析了Plant Pathology-2021数据集的样本细节，并详细介绍了LeNet模型与Vision Transformer模型的构建、训练与优化过程以及它们在Plant Pathology-2021分类任务上不同的性能差距。我们还讨论了ArcFaceLoss与CrossEntropyLoss在本任务上的效果对比，以及对比数据集裁剪的效果。最后，本报告提出了一些可能的优化方向。

Keywords： one-hot，LeNet，VIT，Plant Pathology-2021，ArcFaceLoss

1 Introduction

苹果是世界上最重要的温带水果作物之一。叶病对苹果园的整体生产力和质量构成重大威胁。目前苹果园的疾病诊断过程是基于人类的人工侦察，这既耗时又昂贵。尽管基于计算机视觉的模型在植物疾病识别方面显示出了前景，但仍有一些局限性需要解决。不同苹果品种或栽培中新品种的单一疾病视觉症状的巨大差异是基于计算机视觉的疾病识别的主要挑战。这些变化源于自然和图像捕获环境的差异，例如，叶片颜色和叶片形态、感染组织的年龄、不均匀的图像背景以及成像过程中不同的光照等。 Plant Pathology-2021是数据集通过表示在不同成熟阶段和一天中不同时间在不同焦距相机设置下拍摄的叶片图像的非均匀背景来反映真实的现场场景。我们所获得的数据集包含3000张图片的训练集，600张图片的验证集与600张图片的测试集。

学号: 21312482
课程: 计算机视觉
日期：2023 年 11 月 25 日
Email:liufx7@mail2.sysu.edu.cn

2 Data Analysis

2.1 Label Distribution

我们首先关注训练集内标签数量，简单统计后如图1所示，可见不同标签之间会有极大的样本数量差异。

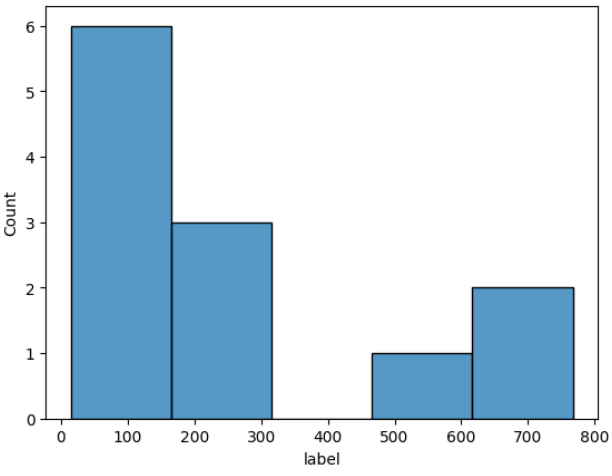

图 1: 不同样本数量的标签统计

此外，我们可以使用条形统计图来直观演示

训练集中各标签的样本数量：

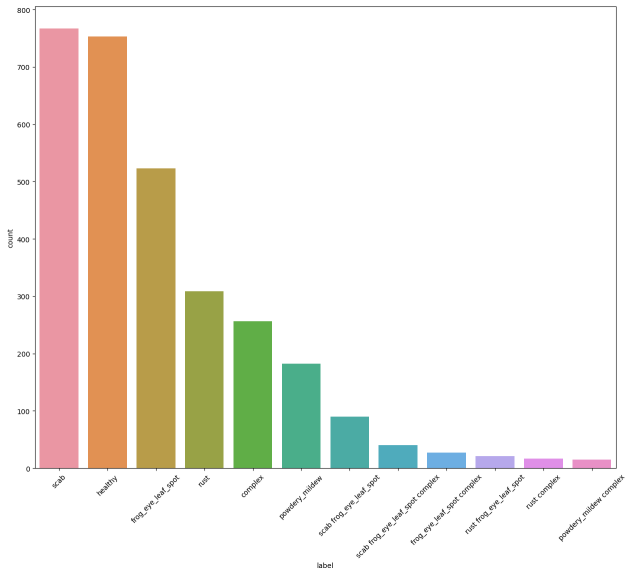

图 2: 不同标签的样本数量统计

观察条形统计图我们可以发现，所有样本一共有12个类别，其中[scab]、[healthy]、[frog_eye_leaf_spot]、[rust]、[complex]、[powdery_mildew]这6个类别的样本数量较多，而另外6个类别则由前6个类别组合而来，且样本数量较少，所以可以采用独热编码将这12个类别的分类任务转化为6个标签的多标签分类任务。

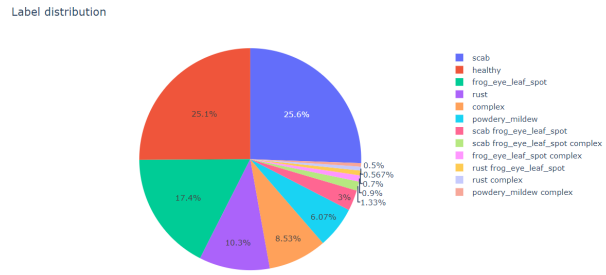

图 3: 不同标签的样本数量占比图

2.2 One-Hot State Assignment

我们将6个标签编码为1和0，并添加在原数据集的后面，如下图所示：

	images	labels	label_list	healthy	complex	rust	frog_eye_leaf_spot	powdery_mildew	scab
0	a8ab965f868fc44c.jpg	healthy	[healthy]	1	0	0	0	0	0
1	a5d8924f7fad18a0.jpg	healthy	[healthy]	1	0	0	0	0	0
2	fab3f2b1c0d2a982.jpg	scab	[scab]	0	0	0	0	0	1
3	d9b283cd98b19d13.jpg	scab	[scab]	0	0	0	0	0	1
4	852979c129dde25d.jpg	frog_eye_leaf_spot	[frog_eye_leaf_spot]	0	0	0	1	0	0
...	...	...	...	...	...	...	...	...	...
2995	eb8100abfd56157c.jpg	healthy	[healthy]	1	0	0	0	0	0
2996	a89097fa5afb011b.jpg	healthy	[healthy]	1	0	0	0	0	0
2997	f05ad49e5f0a0d4b.jpg	frog_eye_leaf_spot	[frog_eye_leaf_spot]	0	0	0	1	0	0
2998	be7ef6d45ac18401.jpg	complex	[complex]	0	1	0	0	0	0
2999	c5b420ff00b9fc2d.jpg	scab	[scab]	0	0	0	0	0	1

图 4: 独热编码后的数据集

对于独热编码后的数据集，我们使用以下热度图来进行可视化：

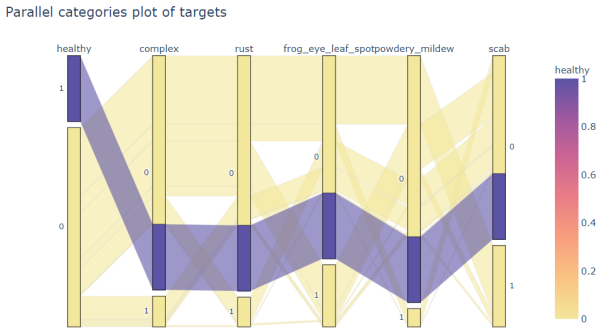

图 5: 标签平行类别图

在上图中，我们可以看到所有6个标签之间的关系。正如预期的那样，一片健康的叶子不可能出现结痂、铁锈或多种疾病。此外，每一片不健康的叶子都有一种结痂、铁锈或多种疾病。

2.3 Fine-Grained Classification

细粒度图像分类问题是对大类下的子类进行识别。细粒度图像分析任务相对通用图像（General/GenericImages）任务的区别和难点在于其图像所属类别的粒度更为精细。对于本次任务来说，所要分类的目标都是苹果树叶，只是有微小的部分差异，明显属于细粒度分类问题。我们随机挑选一些样本如下图所示：

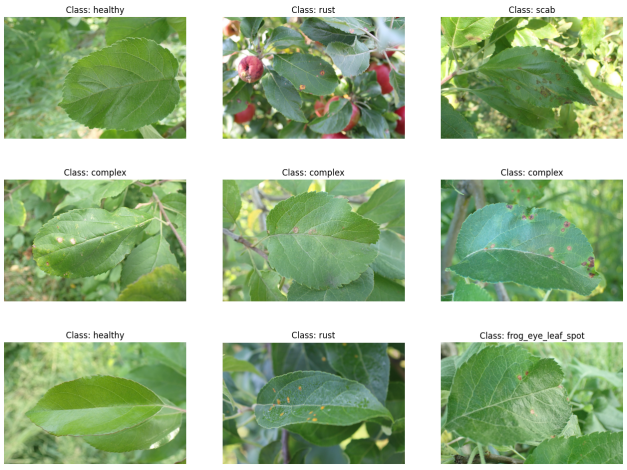

图 6: 随机挑选的训练集样本

由于分类的粒度很小，细粒度图像分类非常困难，在某些类别上甚至人眼都难以区分。这是因为各个类别之间差异细微，只在某个局部上有细微差异，如[healthy]样本与[scab]样本之间的差异只有一些小斑点，而内部差异巨大，如同种类的样本的姿态、背景都会有很大差别。

3 Deep LeNet

3.1 Neural Network Structure

模型LeNet诞生于1994年，是最早的卷积神经网络之一。LeNet通过巧妙的设计，利用卷积、参数共享、池化等操作提取特征，避免了大量的计算成本，最后再使用全连接神经网络进行分类识别，这个网络也是最近大量神经网络架构的起点。我们使用TensorFlow框架中的keras模型搭建神经网络，修改了传统的7层LeNet的部分尺寸参数以适应图片分辨率，各卷积层之间选用relu函数作为激活函数，最后的全连接层使用sigmoid激活函数并调整输出维度为6个特征。

3.2 LeNet Training

在LeNet的训练过程中，我们将图片resize为128的分辨率，然后设置batch_size=64，进行50个epoch的训练。优化器选择Adam，设置学习率lr=0.001。损失函数方面，我们首先选择了交叉熵损失。训练过程的准确率曲线与损失曲线如下：

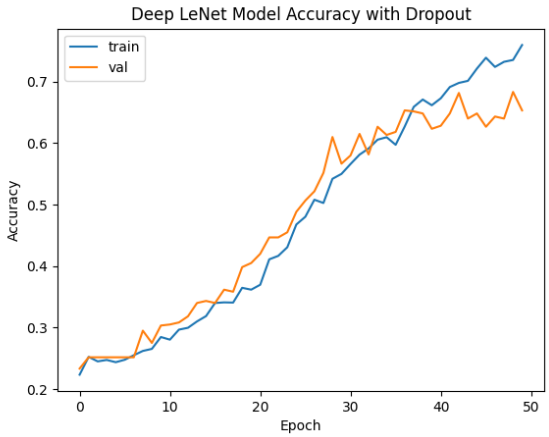

图 7: crossentropyloss accuracy

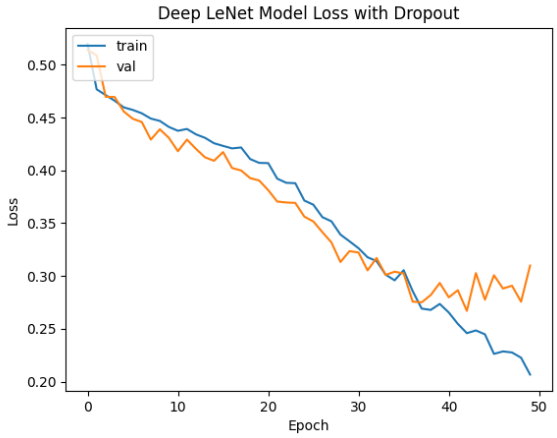

图 8: crossentropyloss loss

值得一提的是，在准确率的计算方法上，我们考虑了两种不同的计算方式：

1. 计算所有标签都正确的样本占总样本的比例：
   - 优点：这种方式关注的是整体样本中所有标签都被正确预测的情况。这对于确保每个样本的所有标签都被正确预测是有益的，特别是在每个标签都很重要的情况下。
   - 缺点：它可能对样本中某些标签的重要性不够敏感，因为它只关注所有标签的总体准确率，而不考虑每个标签的独立性。

2. 计算预测正确的标签占全部预测标签的比例：

   - 优点：这种方式更注重每个标签的独立性，确保对每个标签的预测都是相对准确的。这对于每个标签的独立重要性更强调的任务是有帮助的。
   - 缺点：它可能忽略了样本中所有标签都被正确预测的情况，因为它只关注每个样本中每个标签的独立准确率。

最终我们选定使用第一种准确率计算方式，这是因为样本中的每个标签都很重要，而由于样本中各标签数量的不平衡，例如[scab]标签的数量占了几乎50%，如果我们只计算预测正确的标签占全部预测标签的比例，那么只关注[scab]标签的模型反而会获得更高的分数。

我们可以看到，虽然在训练集上的效果随着训练轮数的增加不断提升，但是在验证集上的效果则在大约35个epoch后不再提升，最终的验证准确率停留在65%左右。

除了交叉熵损失，我们同样考虑了更换其他损失函数来优化模型。ArcFaceLoss是人脸识别领域常用的损失函数，人脸识别同样属于细粒度问题，与我们的任务有相似性，于是我们尝试了将损失函数更换为ArcFaceLoss，并以同样的配置再次训练了一个新的LeNet网络。但很遗憾的是，根据以下的准确率曲线，我们发现使用ArcFaceLoss对于本网络并没有什么提升。

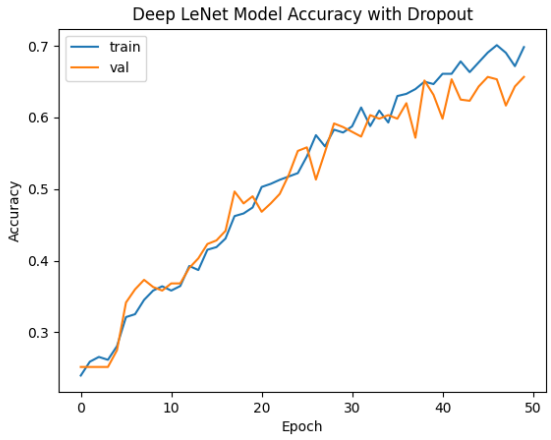

图 9: crossentropyloss accuracy

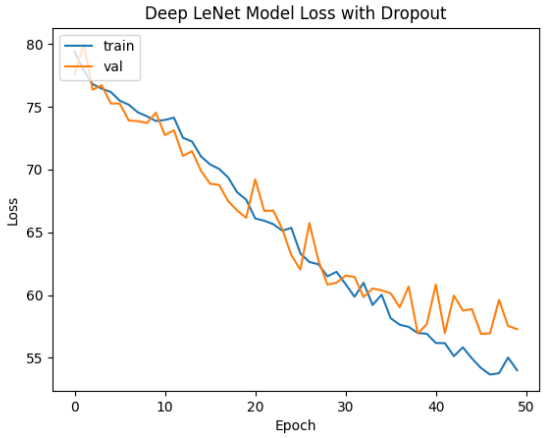

图 10: crossentropyloss loss

4 Vision Transformer

Vision Transformer(VIT)是Transformer在视觉领域的应用，具有非常好的效果。我们在LeNet的基础上，也尝试使用VIT来训练模型。

4.1 First VIT Training

我们使用pytorch_lightning库中的lightning模型搭建了我们自己的VITbase16模型，并设置了输入尺寸为224，首次训练了30个epoch，损失曲线如下：

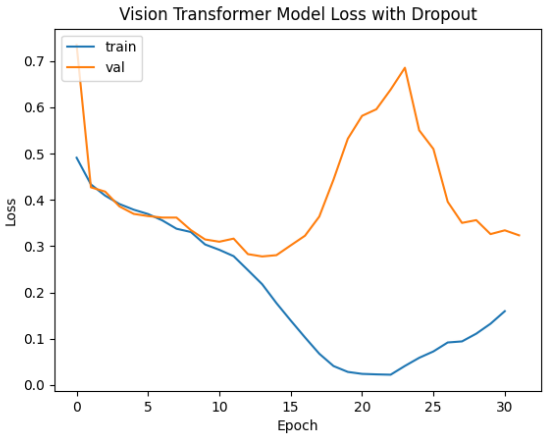

图 11: loss curve of 30 epoch

不难看出，从大约第15轮开始，模型出现了过

拟合现象。而20轮之后，模型在训练集上的性能也开始下降，其中，在验证集上的性能并没有上升，我们从准确率曲线中可以看出这一点。

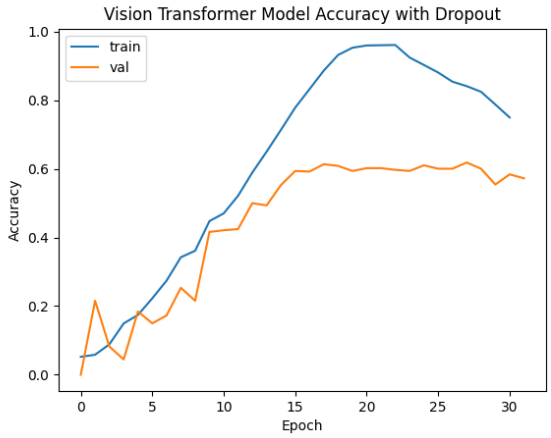

图 12: accuracy curve of 30 epoch

4.2 Early Stopping

基于4.1的实验现象，我们设置了一个早停反馈器，监视每一轮中验证误差的变化，当验证误差连续三轮不变或升高，就会提前停止训练。以下是添加了早停的训练结果：

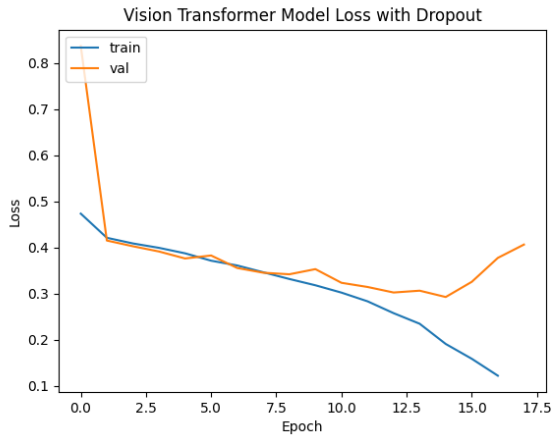

图 13: 早停后的损失曲线

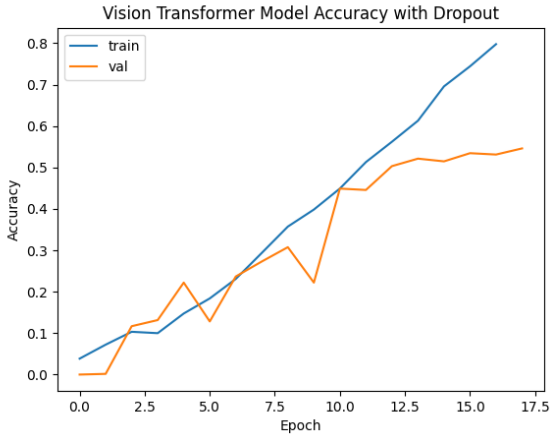

图 14: 早停后的准确率曲线

从准确率来看，VIT模型的效果甚至不如之前的LeNet模型，这可能是VIT模型并没能很好地提取图像特征。

5 Leaf Extraction

观察数据集中的图片，我们可以发现这些图片普遍都具有较大的分辨率，往往是4000*2672或者其他差不多的分辨率。这样大分辨率的图片在被resize到256分辨率然后输入神经网络时，容易丢失重要信息，尤其是对于细粒度问题来说，关键信息往往都是很小的区域。而数据集的许多图片中，叶片只占了画面中较少的比例，更多的则是背景环境等无关信息。

针对这个问题，我们可以考虑使用边缘算法或者训练另一个神经网络，先提取出叶片区域，按照这个区域裁剪一遍图片，然后再进行resize，这样子可以尽量避免无关信息的干扰，且减少resize后的信息丢失。

我们使用Canny边缘检测算法提取图像的边缘，并使用一个尽可能小的框来包含这些边缘，这样框内的物体大概率就是我们所需要关注的叶子，将图片按照框的边来裁剪，并resize到256分辨率，我们就可以获得一个更精确的数据集。值得一提的是，Canny边缘检测算法在面对部分边缘时其实效果并不是特别完美，有些图片可能只能检测出少量边缘，所以我们设置了一个阈值，若是框的长

宽小于1000像素则不裁剪这张图片。

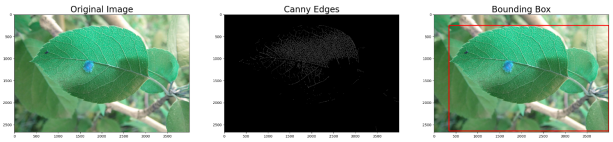

图 15: Effect of Leaf Extraction

5.1 LeNet with Leaf Extraction

我们用裁剪过的数据集再次训练了一个新的LeNet，令人遗憾的是，裁剪数据集并没有取得更好的成果，新训练的模型在验证集上的性能反而降低了5%左右。

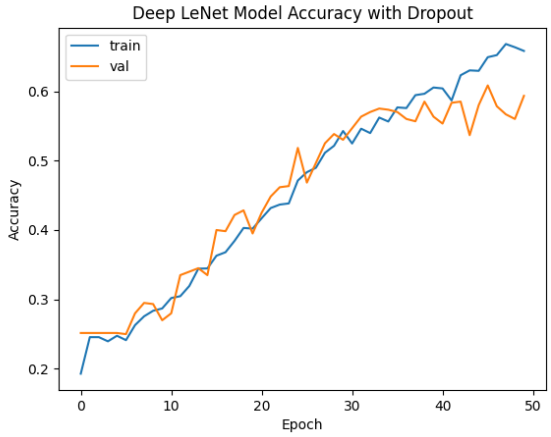

图 16: Accuracy of LeNet with Leaf Extraction

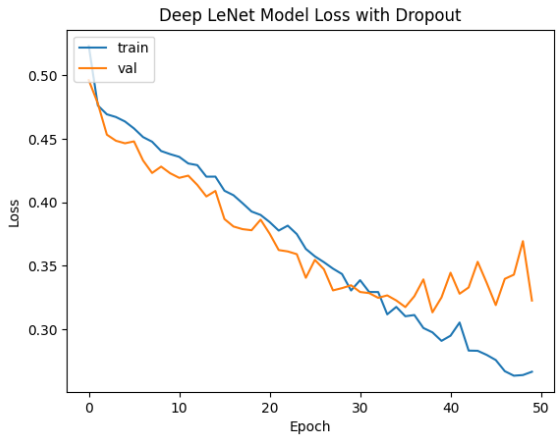

图 17: Loss of LeNet with Leaf Extraction

5.2 VIT with Leaf Extraction

我们同样用裁剪过的数据集再次训练了一个新的VIT，性能比之前有少量提升。

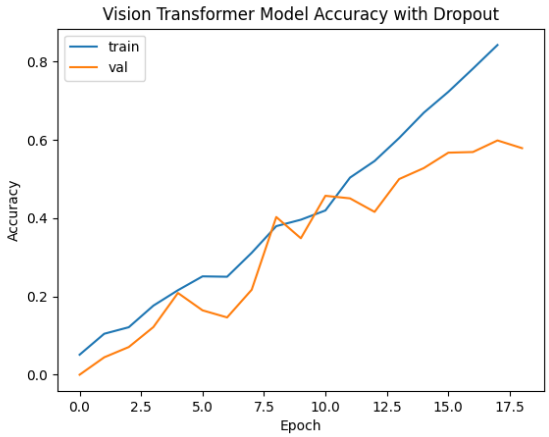

图 18: Accuracy of VIT with Leaf Extraction

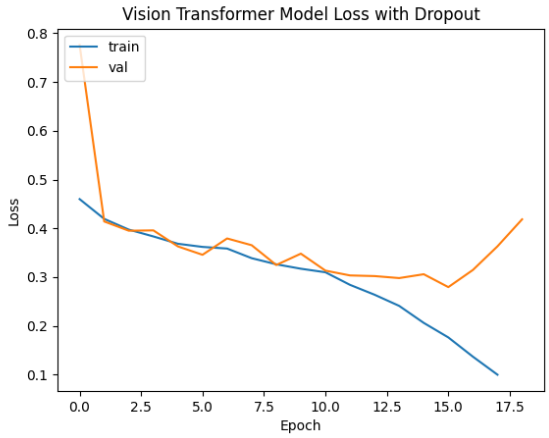

图 19: Loss of VIT with Leaf Extraction

5.3 Conclusion

我们在测试集上对训练出来的各个模型进行准确率、精确率与召回率的测试，结果如表1所示。

从表1中可见，裁剪后的数据集对于VIT有7%左右的性能提升，但是对于LeNet却起了反效果。我们猜测这可能是因为VIT的注意力机制更容易从裁剪后的图片中提取关键特征。

表 1: Performance of each model

model	accuracy	precision	recall
LeNet with CrossEntropy-Loss	57%	71.57%	56.92%
LeNet with ArcFaceLoss	50.67%	60.95%	62.99%
VIT	53.45%	61.48%	60.13%
LeNet with Leaf Extraction	43.5%	61.78%	43.23%
VIT with Leaf Extraction	60.2%	68.39%	64.58%

裁剪后的数据集对模型性能的提升并没有很大，我们猜测可能的原因是Canny边缘检测算法对叶子的提取并不精确，因此，我们推测使用深度学习的方式提取更加精细的叶子位置应该是更加可行的方法。

6 Improvement Directions

在本节我们将介绍一些可能的改进方向，我们发现了这些特性，但由于各种限制我们无法实验。

6.1 Distribution of RGB

我们在训练集中随机选取100个样本，通过观察可以发现病害的叶片上往往有许多棕色斑点，而在RGB通道中，图像的绿色部分具有非常低的蓝值，但相比之下，棕色部分具有较高的蓝值。这表明图像的绿色（健康）部分具有较低的蓝值，而不健康的部分更有可能具有较高的蓝值。这可能表明蓝通道可能是检测植物疾病的关键。

我们将这100个样本的RGB通道值分别统计并画出如下的RGB分布图：

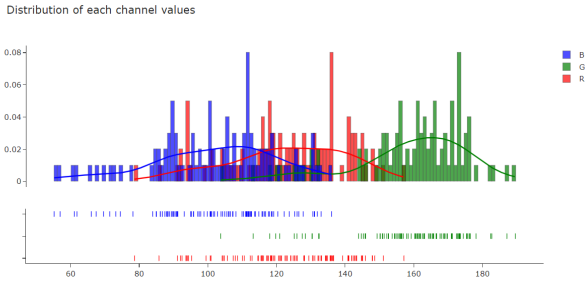

图 20: Distribution of each channel values

观察RGB分布图，我们发现绿通道值普遍较高，这与叶片的绿色是相符合的。而我们重点关注的蓝通道值中，有相当一部分都较高，这与[scab]特征的叶片比例最多的事实也相吻合。

所以一个可能的优化方向是定制数据增强方法，设计一个能提取RGB关键信息来进行初步预分类的算法。

6.2 Using Bounding Box

在细粒度分类问题中，基于定位-识别的方法需要先找到有区分度的局部，然后进行特征提取和分类。在这个过程中，如果能获得类别标签以外的部件标注和关键部位框，往往能取得不错的效果，但是该方法的缺点在于需要昂贵的人工标注，而且人工标注的位置不一定是最佳的区别性区域，这完全依赖于标注者的认知水平。由于本数据集的限制，我们无法获取Bounding Box的辅助标识，因此仅作为一个优化方向在此提出。

7 Experience and Insights

Plant Pathology-2021分类任务难度较高，除了本报告中介绍的这两种模型，我们还尝试了许多模型，包括无DROPOUT的LeNet等模型结构。但是往往是这样的情况：模型在训练集上的效果非常好，但是在测试集上效果提升缓慢且最终准确率并不高。当训练轮数增加时，验证误差曲线呈现先降后增的趋势。

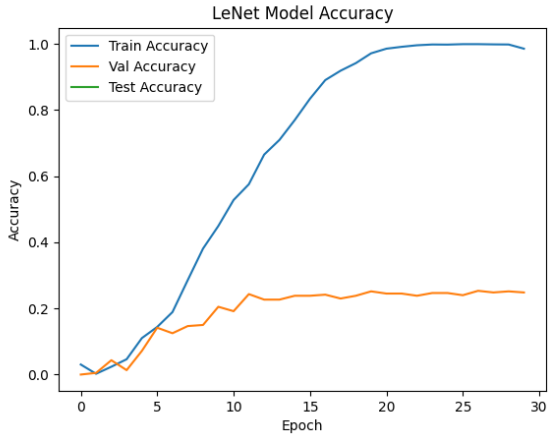

图 21: Accuracy of LeNet Train on Pytorch

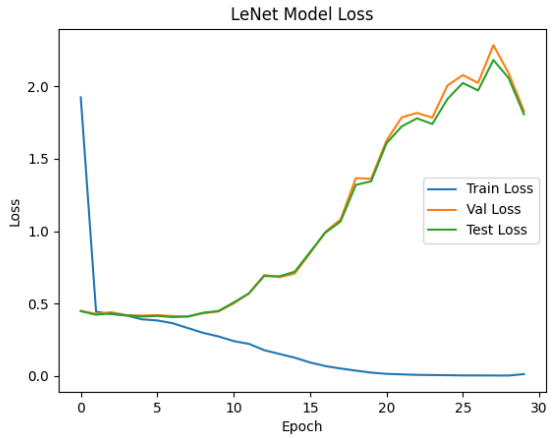

图 22: Loss ofLeNet Train on Pytorch

我们甚至将本报告中的LeNet模型在pytorch上复现了一个，选用一模一样的参数，但是与在TensorFlow上训练的效果却天差地别。

PS:在本报告中之所以使用“我们”而不是“我”，这是因为一般的论文中都是用“we”，用“I”反而会很奇怪。

8 Code Introduction

以下是本报告附件中代码功能的介绍，在使用这些代码之前，您可能需要先获取Plant Pathology-2021数据集，配置代码运行环境，安装所需要的库，并修改代码中的数据集路径(我们会在下面的介绍中注明)。

- edge_and_cut.py：进行Canny边缘检测并裁剪图片，需要修改第59、60行的图片文件夹路径
- resize_to_256.py：将图片缩放到256分辨率，需要修改第30、32行的图片文件夹路径
- Data Analysis.ipynb：用来进行第2节中的数据分析，需要修改训练集的文件夹路径
- LeNet/main_train.py：LeNet网络的训练
- LeNet/main_test.py：LeNet网络的测试
- LeNet/data_preprocessing.py：数据读取文件，在运行训练或测试前需要先配置第11/12/13行的数据集路径
- LeNet/ArcLoss.py：ArcFaceLoss损失函数的实现代码
- LeNet/CONFIG.py：超参数的配置文件，超参数的修改在此
- LeNet/plot.py：用来绘制网络结构
- LeNet/pytorch_train.py：使用pytorch构建的LeNet网络的训练与测试
- VIT/main_train.py：VIT网络的训练，需要修改第17行的数据集路径
- VIT/main_test.py：VIT网络的测试
- VIT/data_preprocessing.py：数据读取文件，在运行训练或测试前需要先配置get_df函数与setup函数中的数据集路径
- VIT/ArcLoss.py：ArcFaceLoss损失函数的实现代码
- VIT/CONFIG.py：超参数的配置文件，超参数的修改在此
- VIT/plot.py：用来绘制网络结构
- VIT/models.py：定义ViT模型