

中山大學



《计算机视觉小组作业》

Vis-Efficient与Eva-CLIP改进方案

授课教师 : 陈俊周

小组成员 : 周宇平 王忠灿 张佩伦 李梓牧 吴振哲

日期 : 2024.12.9

基于EfficientNet与EVA-CLIP的模型改进与可视化方案

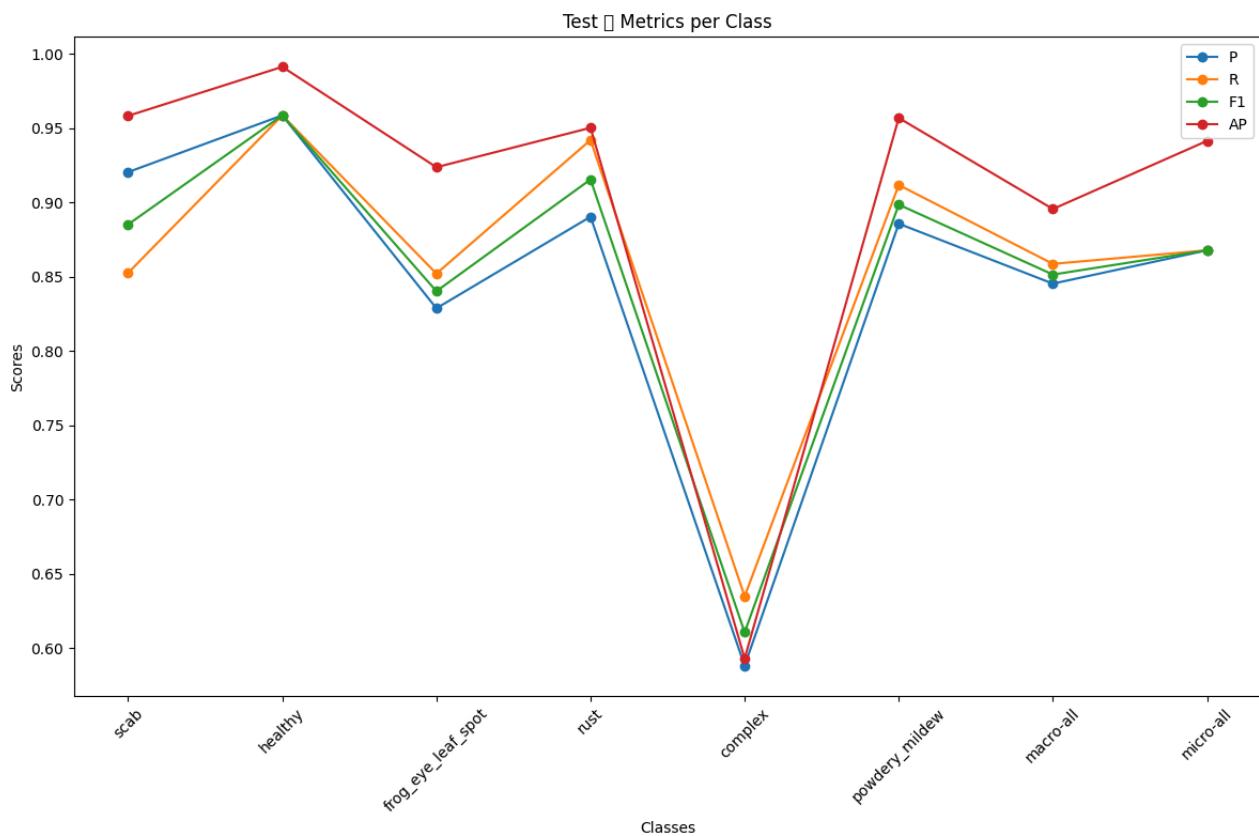
摘要: 在信息化时代背景下,计算机操作系统在各行业领域中得到广泛应用。本文试图通过对计算机系统概念、功能和分类、历史的介绍,大致地给出对计算机操作系统的认识,并通过对当前主流桌面操作系统Windows和macOS的简要介绍,使读者能够对现代图形界面操作系统有一个较为全面的了解,从而喜欢上某一个操作系统,供读者参考。

关键词: 计算机; 操作系统; 功能; 发展; Windows; macOS;

1 概况

本项目主要使用了两个模型,分别是[EfficientNet](#)与[EVA-CLIP](#)。在已有模型的基础上,我们对EfficientNet与EVA-CLIP进行了模型架构上的优化,在训练过程中尝试了多种优化方案,例如数据集增强,学习率动态调整,梯度裁剪等方法,使得模型更适合Plant Pathology-2021的细粒度问题,取得了一定效果。并且我们应用GradCam对EfficientNet进行可视化,生成热力图来突出显示输入图像中对模型预测结果贡献最大区域,同时我们基于Attention的原理,实现了对EVA-CLIP的可视化,生成热力图来突出显示输入图像中对模型预测结果贡献最大区域。结果如下,在后续会详细介绍:

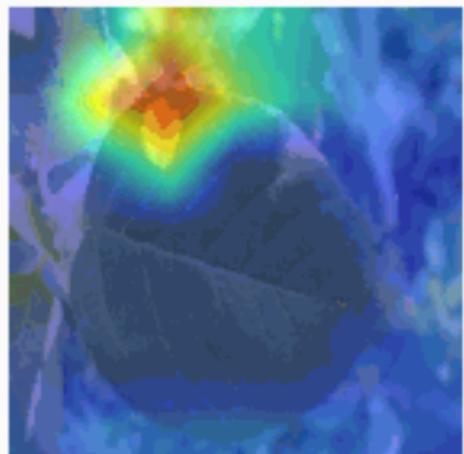
EfficientNet模型在Test数据集上的指标:



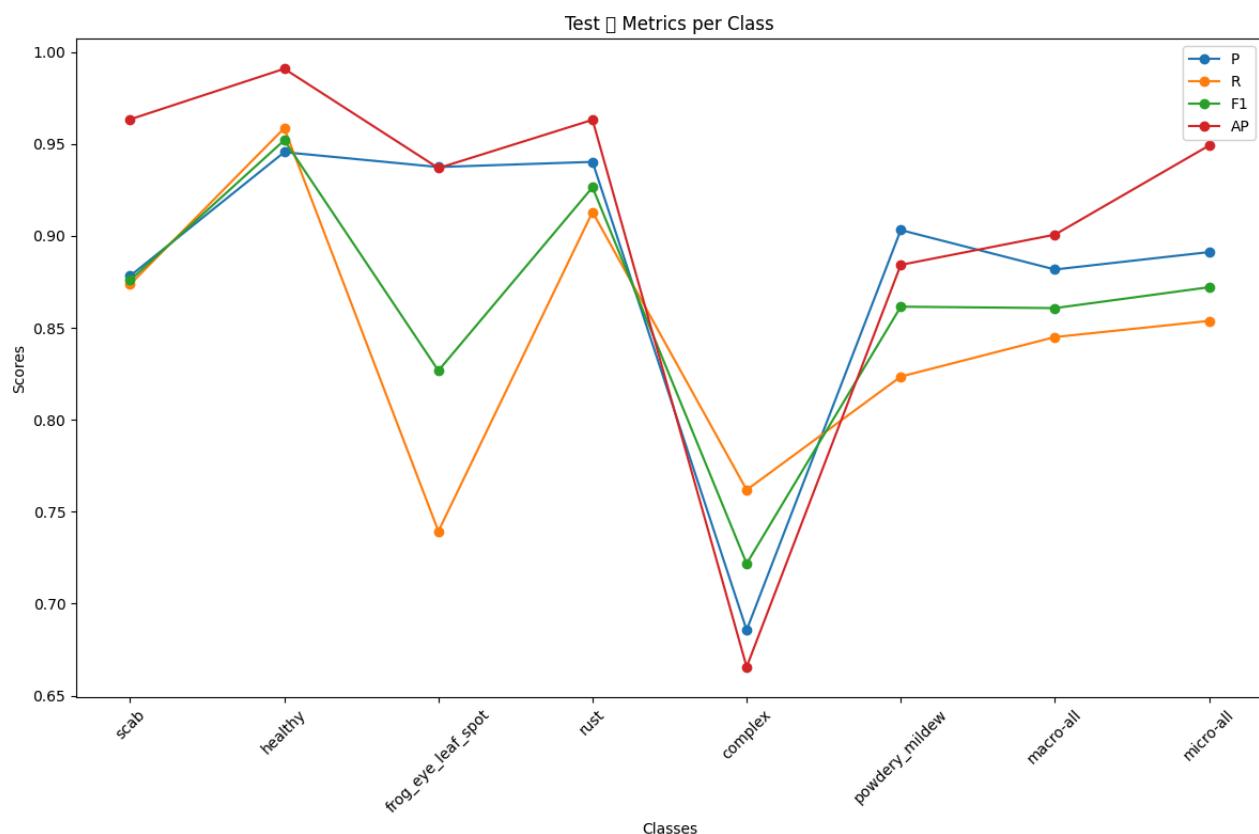
Class	Threshold	P	R	F1	AP
scab	0.5500	0.9205	0.8526	0.8852	0.9583
healthy	0.5000	0.9586	0.9586	0.9586	0.9913
frog_eye_leaf_spot	0.5000	0.8288	0.8521	0.8403	0.9237
rust	0.5000	0.8904	0.9420	0.9155	0.9503
complex	0.4000	0.5882	0.6349	0.6107	0.5932
powdery_mildew	0.5500	0.8857	0.9118	0.8986	0.9569
macro-all	N/A	0.8454	0.8587	0.8515	0.8956
micro-all	N/A	0.8678	0.8678	0.8678	0.9414

EfficientNet模型的可视化结果：

Original Image Trained EfficientNet-B0



EVA-CLIP模型在Test数据集上的指标：



Class	Threshold	P	R	F1	AP
scab	0.5000	0.8783	0.8737	0.8760	0.9634
healthy	0.6000	0.9456	0.9586	0.9521	0.9909
frog_eye_leaf_spot	0.6500	0.9375	0.7394	0.8268	0.9369
rust	0.5000	0.9403	0.9130	0.9265	0.9632
complex	0.3500	0.6857	0.7619	0.7218	0.6656
powdery_mildew	0.7500	0.9032	0.8235	0.8615	0.8843
macro-all	N/A	0.8818	0.8450	0.8608	0.9007
micro-all	N/A	0.8912	0.8538	0.8721	0.9492

EVA-CLIP模型的可视化结果：



2 数据集介绍

2.1 数据集概况

本项目使用的Plant Pathology-2021包含12个类别，每个类别对应不同的叶片病害类型。然而，不同类别之间存在标签重复（例如，`scab`在多个类别中出现），这增加了模型训练的复杂性。因此，我们对数据集进行了类别合并，将原有的12个类别转换为6个类别，并将其视为一个多标签分类问题。

2.2 类别说明

经过合并后的6个类别如下：

- scab
- healthy
- frog_eye_leaf_spot
- rust
- complex
- powdery_mildew

每个类别通过*One-Hot*编码进行标签表示，以适应多标签分类任务。

3 数据处理

3.1 One-Hot编码操作

One-Hot编码是一种将分类数据转换为二进制向量的技术。在多标签分类任务中，每个样本可以同时属于多个类别。**One-Hot**编码能够有效地表示这种多标签关系，使得模型能够处理每个类别的独立预测。

在本项目中，由于原始数据集包含12个类别，且存在标签重复（如scab在多个类别中出现），我们将类别合并为6个类别，并将其转换为多标签分类问题。具体操作如下：

1. **读取原始标签文件**：使用 `pandas` 读取包含图像名称及其对应标签的 CSV 文件。
 2. **One-Hot编码**：使用 `MultiLabelBinarizer` 将标签列表转换为 One-Hot 编码向量。
 3. **合并数据**：将图像名称与编码后的标签向量合并，生成最终的标签文件
- 具体的代码实现可以参考项目中 `label.py` 文件，处理后生成三个csv文件，分别对应训练集，验证集与测试集存放于data文件夹中。

以下为One-Hot编码后的csv文件示例：

```
images,scab,healthy,frog_eye_leaf_spot,rust,complex,powdery_mildew
80bcfd9f60f06307.jpg,0,0,0,0,0,1
e062c0b63cd36f1b.jpg,0,0,0,0,0,1
df3b8800e2f892fa.jpg,0,0,0,0,1,0
cff02d717850c0b7.jpg,1,0,0,0,0,0
ac4fd4118a9e2df8.jpg,1,0,0,0,0,0
fd37c332c54d120d.jpg,1,0,0,0,0,0
84f81ac3c5ed6566.jpg,0,1,0,0,0,0
db9961a5d34242cf.jpg,0,0,1,0,0,0
cbe3ad8ca8898bb4.jpg,1,0,0,0,0,0
.....
```

可见标签列表已经被转换为 One-Hot 编码向量。

3.2 数据增强策略

在深度学习中，数据增强（Data Augmentation）是一种通过对训练数据进行随机变换来生成更多样本的方法。这不仅能够增加数据的多样性，减少过拟合，还能帮助模型更好地泛化到未见过的数据。在本项目中，由于叶片病害分类属于细粒度分类，类内差异大、类间差异小，数据增强显得尤为重要。

在本项目的[train.py](#)中，我们对训练数据应用了多种数据增强技术，具体如下：

```
train_transform = transforms.Compose([
    transforms.Resize((224, 224)),
    transforms.RandomHorizontalFlip(),
    transforms.RandomRotation(15),
    transforms.ColorJitter(brightness=0.2, contrast=0.2, saturation=0.2, hue=0.2),
    transforms.RandomAffine(degrees=10, translate=(0.1, 0.1), scale=(0.9, 1.1)),
    transforms.ToTensor(),
    transforms.Normalize(mean=[0.485, 0.456, 0.406],
                        std=[0.229, 0.224, 0.225]),
])
```

具体方法介绍

1. Resize：将所有图像调整为统一的尺寸（224x224），确保模型输入的一致性。
2. RandomHorizontalFlip：随机水平翻转图像。此方法可以模拟植物叶片在不同方向上的自然生长情况，增加模型对方向变化的鲁棒性。
3. RandomRotation：随机旋转图像最多15度。旋转变换能够使模型更好地识别不同角度下的叶片病害特征，提升模型的泛化能力。
4. ColorJitter：模拟不同光照环境下的数据。此方法可以使模型更好地识别不同光照下的叶片病害特征，提升模型的泛化能力。
5. RandomAffine：可以进行更广泛的空间变换，如平移、旋转和缩放。这样可以增加对视角、位置和尺度的鲁棒性。
6. ToTensor：将PIL图像或NumPy ndarray转换为形状为(C, H, W)的张量，并且将像素值归一化到[0, 1]之间。
7. Normalize：因为选用的模型在ImageNet上进行预训练，使用ImageNet的均值和标准差对图像进行归一化处理，有助于加快训练收敛速度，提高模型性能。

这些数据增强的操作显著提升了模型的泛化能力，并且强化了对细粒度特征的学习。

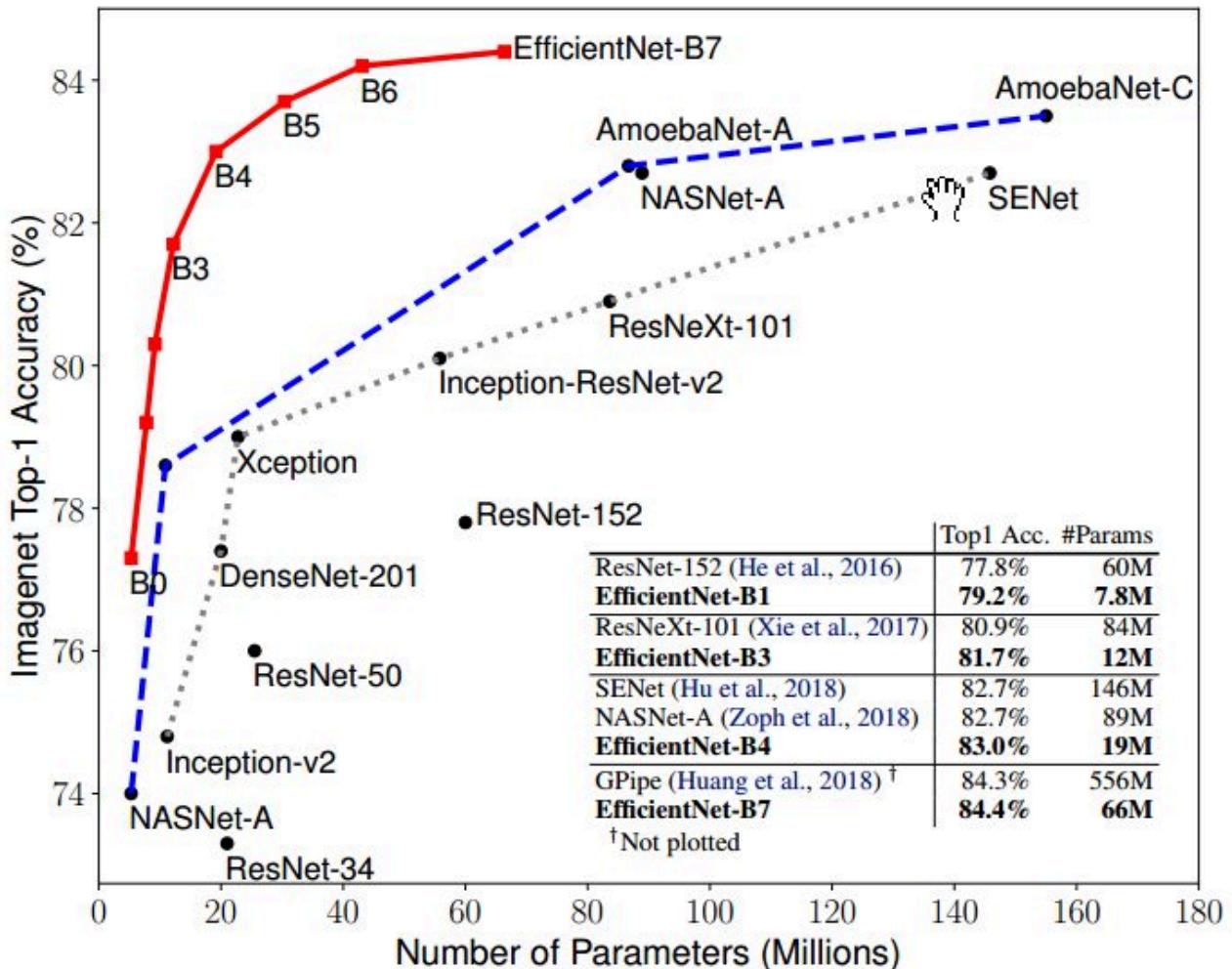
4 模型介绍

本次实验我们引入了两个模型，分别是[EfficientNet](#)与[EVA-CLIP](#)。

4.1 EfficientNet介绍

EfficientNet源自Google Brain的论文EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks. 从标题也可以看出，这篇论文最主要的创新点是Model Scaling. 论文提出了compound scaling，混合缩放，把网络缩放的三种方式：深度、宽度、分辨率，组合起来按照一定规则缩放，从而提高网络的效果。EfficientNet在网络变大时效果提升明显，把精度上限进一步提升。EfficientNet-B7在ImageNet上获得了先进的84.4%的top-1精度和97.1%的top-5精度，比在EfficientNet之前最好的卷积网络（GPipe, Top-1: 84.3%, Top-5: 97.0%）大小缩小8.4倍、速度提升6.1倍。

EfficientNet的主要创新点并不是结构，不像ResNet、SENet发明了shortcut或attention机制，EfficientNet的base结构是利用结构搜索搜出来的，然后使用compound scaling规则放缩，得到一系列表现优异的网络：B0~B7。下面得图是ImageNet的Top-1 Accuracy随参数量变化关系图，可以看到EfficientNet饱和值高，并且到达速度快。



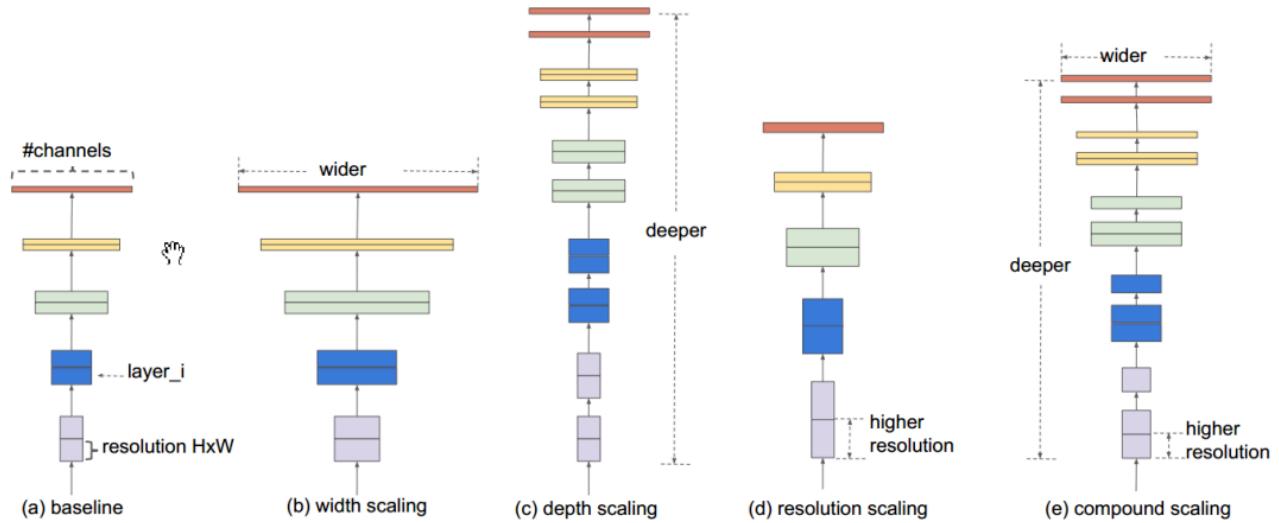
增加网络参数可以获得更好的精度（有足够的数据，不过拟合的条件下），例如ResNet可以加深从ResNet-18到ResNet-200，GPipe将baseline模型放大四倍在ImageNet数据集上获得了84.3%的top-1精度。增加网络参数的方式有三种：深度、宽度和分辨率。

深度是指网络的层数，宽度指网络中卷积的channel数（例如wide resnet中通过增加channel数获得精度收益），分辨率是指通过网络输入大小（例如从112x112到224x224）

直观上来讲，这三种缩放方式并不独立。对于分辨率高的图像，应该用更深的网络，因为需要更大的感受野，同时也应该增加网络宽度来获得更细粒度的特征。

之前增加网络参数都是单独放大这三种方式中的一种，并没有同时调整，也没有调整方式的研究。EfficientNet使用了compound scaling方法，统一缩放网络深度、宽度和分辨率。

如下图所示，(a)为baseline网络，(b)、(c)、(d)为单独通过增加width，depth以及resolution使得网络变大的方式，(e)为compound scaling的方式。



EfficientNet中的base网络是和MNAS采用类似的方法（唯一区别在于目标从硬件延时改为了FLOPS），使用了compound scaling后，效果非常显著，在不同参数量和计算量都取得了多倍的提升。

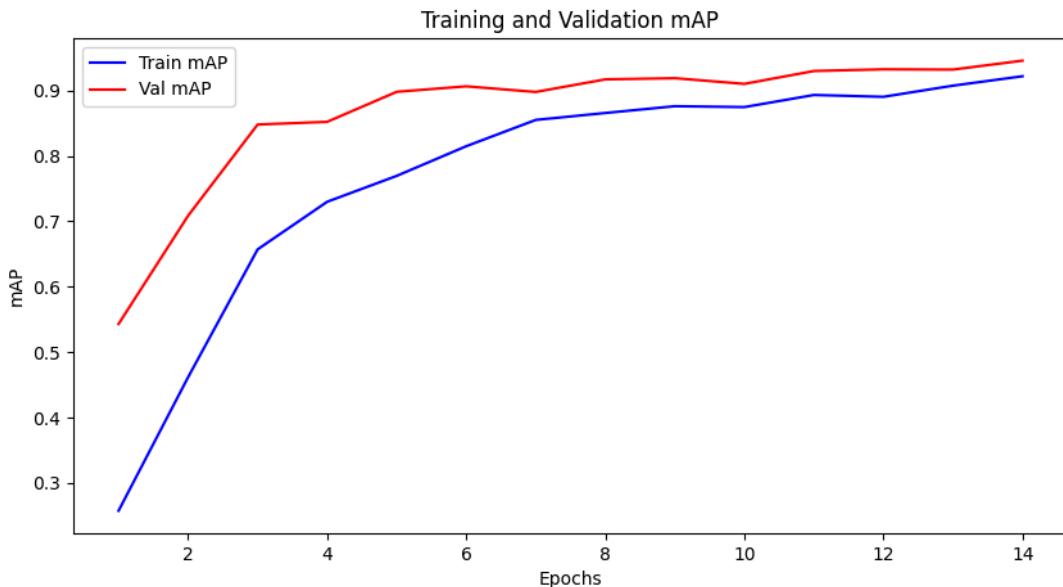
4.2 EfficientNet改进

我们对于EfficientNet的改进主要在模型推理效率方面，参考了[How we made EfficientNet more efficient](#)的方法，我们在原始的模型上引入了分组卷积，EfficientNet将深度卷积用于所有空间卷积操作。它们以其FLOP和参数量中的效率而闻名，因此已经成功地应用于许多最先进的CNN中。然而，在实践中，EfficientNet在加速方面遇到了若干挑战。

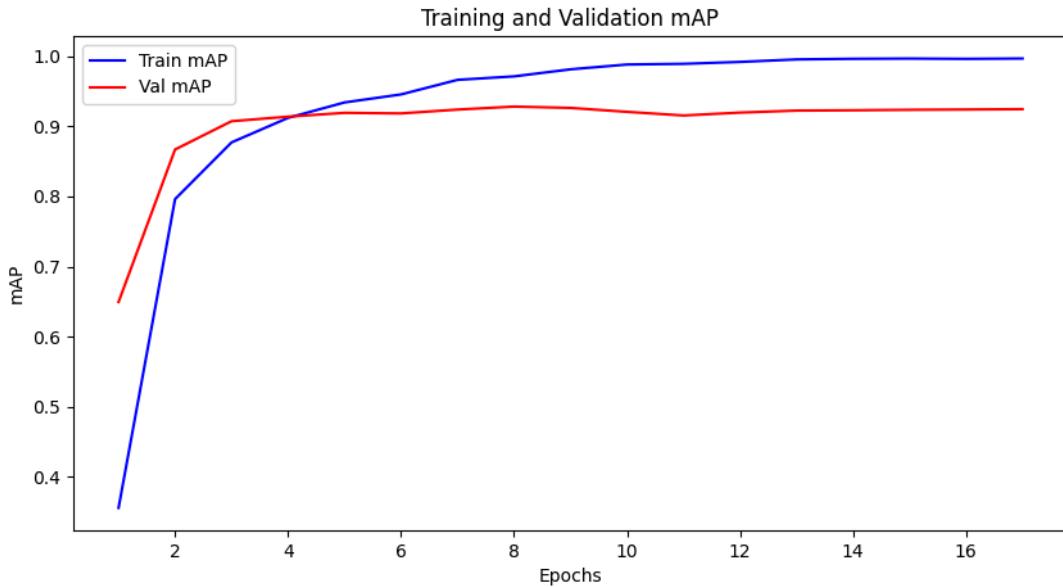
例如，由于每个空间核都需要独立考虑，因此，通常由向量乘积型硬件加速的点积操作的长度是有限的。这意味着硬件不能总是被充分利用，导致“浪费”了周期。

为了解决问题，我们对MBConv块做了一个简单但重要的修改。在模型的config文件中进行修改，我们将卷积的组大小从1增加到16；这将带来更好的IPU硬件利用率。然后，为了补偿FLOPs和参数的增加，并解决内存问题，我们将扩展因子降低到4。这将带来一个更高效的内存和计算紧凑的EfficientNet。

这是修改前的mAP曲线：



这是修改后的mAP曲线：



可以看到虽然这些改变主要是由于吞吐量的提高，但我们也发现，在所有模型大小上，它们使我们能够实现比普通的组大小为1的基线模型更高的ImageNet验证精度。这一修改导致了实际效率的显著提高。

4.3 EfficientNet的可视化实现

我们应用了 Grad-CAM (Gradient-weighted Class Activation Mapping) 对 EfficientNet 进行可视化，EfficientNet 是一种基于卷积神经网络的架构，具有明确的卷积层结构，而Grad-CAM主要用于卷积层的特征图，所以我们考虑应用Grad-CAM来对可视化进行实现。

GradCam的基本原理：

1. 目标层选择：

- 选择靠近输出层的卷积层作为目标层。
- 这些层包含高层次语义信息，同时保留空间位置信息。

2. 前向传播：

- 输入图像通过模型进行前向传播，得到预测结果（例如分类概率或得分）。

3. 梯度计算：

- 对于目标类别 (c)，计算其得分 (y^c) 相对于目标卷积层特征图 (A^k) 的梯度：

$$\frac{\partial y^c}{\partial A^k}$$

- 梯度反映了特征图中每个通道对目标类别的重要性。

4. 权重计算：

- 对每个通道的梯度进行全局平均池化，得到每个通道的权重 α_k^c ：

$$\alpha_k^c = \frac{1}{Z} \sum_i \sum_j \frac{\partial y^c}{\partial A_{i,j}^k}$$

- 其中 (Z) 是特征图的像素数量。

5. 生成热力图：

- 使用权重 α_k^c 对目标卷积层的特征图进行加权求和：

$$L^c = \text{ReLU} \left(\sum_k \alpha_k^c A^k \right)$$

- ReLU 激活函数去除负值，仅保留对目标类别有积极贡献的区域。

6. 上采样与叠加：

- 将生成的热力图 (L^c) 上采样到与原始图像相同的尺寸。
- 与原始图像叠加，生成可视化结果。

代码实现

在加载模型后，我们首先对图像进行预处理：

```
inputs = [Compose([Resize((224,224)), ToTensor(), image_net_preprocessing])(x).unsqueeze(0) for x in images]
inputs = [i.to(device) for i in inputs]
```

然后将Grad—CAM实例化：

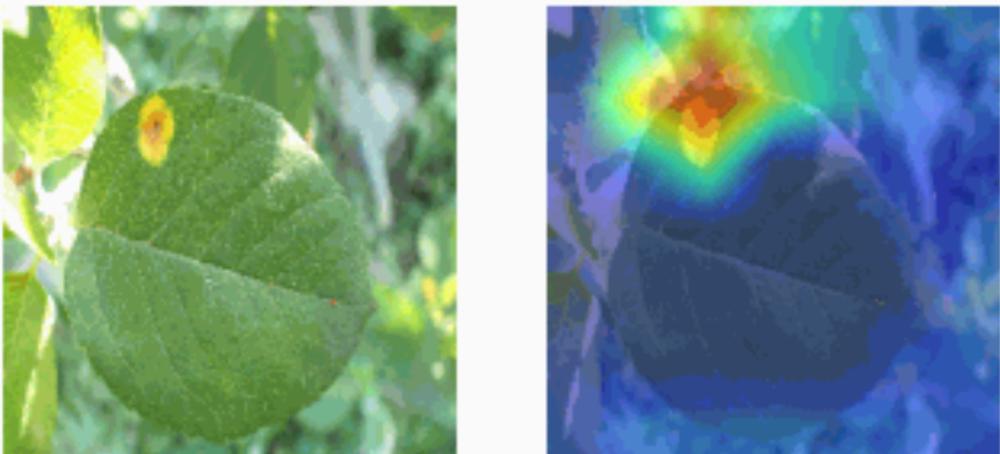
```
vis = GradCam(model, device)
target_layer = model._conv_head
cam_result = vis(inp, target_layer, postprocessing=image_net_postprocessing)[0]
cam_img = tensor2img(cam_result)
```

这样可以对每张预处理后的图像应用 Grad-CAM，生成热力图并转换为图像格式。

具体的代码实现在 [/src/Efficient_visualize.py](#) 中，最后可以得到的结果存放在 [outputs/example.gif](#) 中。

得到结果如下：

Original Image Trained EfficientNet-B0



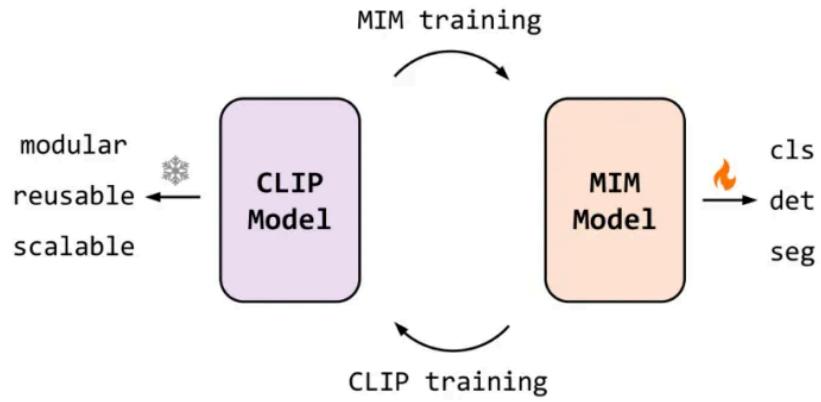
可以看到优化后的EfficientNet模型对于特征的提取效果不错，即使在光照或图像噪声的干扰下仍然能准确的关注图像中的特征。

4.4 EVA-CLIP模型介绍

EVA-CLIP-18B沿用了EVA系列weak-to-strong的视觉模型scale up策略，实现了视觉模型规模的渐进式扩增。该策略遵循“以小教大，以弱引强”的规模扩增思想。

具体而言，团队首先使用一个较小的EVA-CLIP-5B模型作为教师，以掩码图像建模为训练目标，蒸馏出一个较大的EVA-18B纯视觉模型。随后，EVA-18B作为EVA-CLIP-18B视觉-语言对比训练中视觉编码器的初始化，帮助EVA-CLIP模型进一步scale up。

研究结果表明，使用较弱的EVA-CLIP模型引导更大的EVA模型完成视觉训练，再使用得到的EVA模型作为更大EVA-CLIP训练的初始化，这种渐进式地用弱模型引导大模型的scale up方式，可以有效解决大型视觉模型训练中的不稳定问题，并加速模型训练收敛。



4.5 EVA-CLIP模型改进

我们在EVA-CLIP中引入了STN(空间变换网络)模块，空间变换网络（Spatial Transformer Network, STN）是一种可学习的模块，旨在增强神经网络对输入图像的空间变换（如旋转、缩放、剪切和透视变换）的鲁棒性。STN 通过引入一个可微分的模块，允许网络在前向传播过程中对输入图像进行空间变换，从而自动学习最有助于分类的几何变换。

而该问题属于细粒度的分类问题，要求模型能够捕捉到细微的特征差异。然而，这些细微差异往往受到图像中的姿态、尺度、背景等因素的影响。STN 的引入带来以下几个优势：

1. 对齐关键信息：通过自动学习和应用空间变换，STN 可以将图像中的关键信息对齐到一个标准位置，减少姿态和尺度的变化对分类的影响。这有助于模型更好地聚焦于细粒度特征，而不是被不相关的背景或姿态变化干扰。
2. 增强模型的鲁棒性：STN 使模型对输入图像的空间变换更加鲁棒，能够处理不同视角、旋转角度和尺度的图像，提高了模型的泛化能力。
3. 减少数据依赖：在细粒度分类中，数据集可能包含大量姿态和尺度的变化。通过引入 STN，模型可以自动适应这些变化，减少对数据集中每种变换的依赖，从而在有限的数据下仍能取得较好的性能。
4. 提高特征表达能力：对齐后的图像使得后续的卷积层能够更有效地提取细粒度特征，因为这些特征在对齐后的图像中具有更一致的位置和尺度。

我们在 `/src/enhanced_modules.py` 代码中具体有以下实现：

```
class STN(nn.Module):
    def __init__(self):
        super(STN, self).__init__()
        self.localization = nn.Sequential(
            nn.Conv2d(3, 8, kernel_size=7),
            nn.MaxPool2d(2, stride=2),
            nn.ReLU(True),
            nn.Conv2d(8, 10, kernel_size=5),
            nn.MaxPool2d(2, stride=2),
            nn.ReLU(True)
        )
        # 这里的输入维度调整为 10 * 52 * 52
        self.fc_loc = nn.Sequential(
            nn.Linear(10 * 52 * 52, 32), # 27040 是 10 * 52 * 52 的计算结果
            nn.ReLU(True),
            nn.Linear(32, 3 * 2)
        )
        self.fc_loc[2].weight.data.zero_()
        self.fc_loc[2].bias.data.copy_(torch.tensor([1, 0, 0, 0, 1, 0], dtype=torch.float))
```

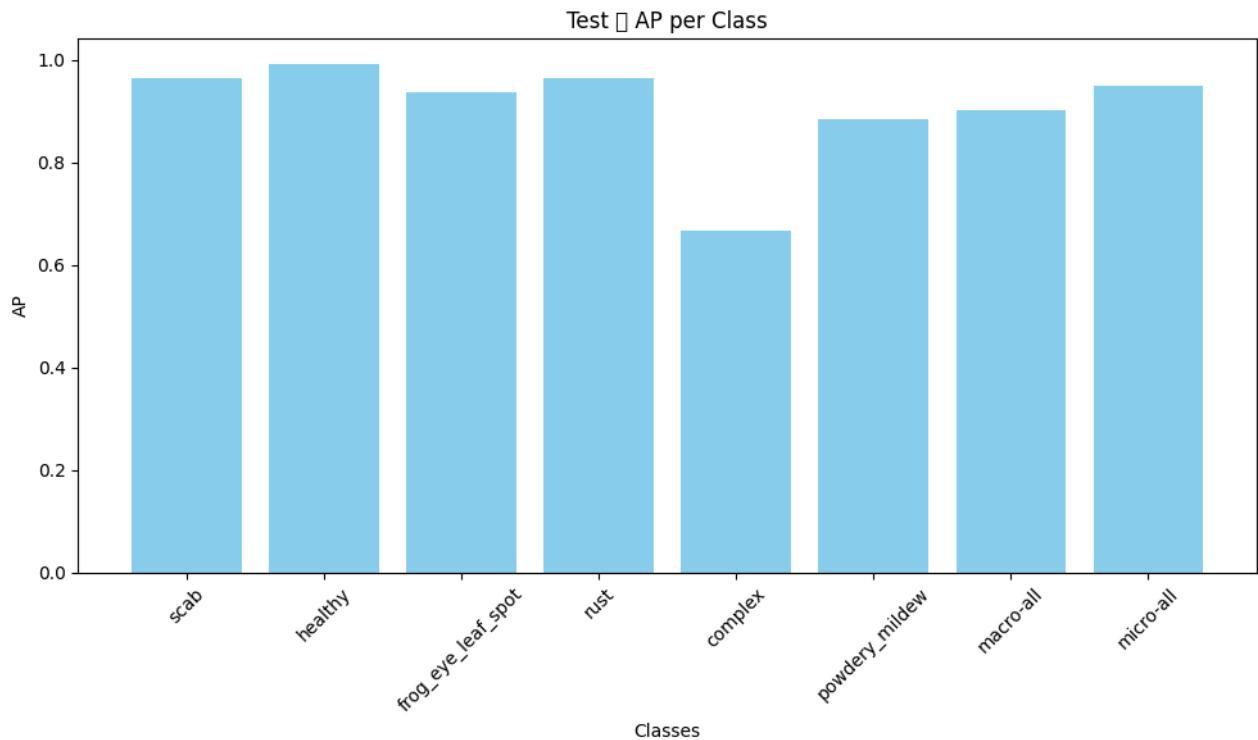
```

def forward(self, x):
    xs = self.localization(x)
    x_size_lst = list(xs.size())
    total_size = x_size_lst[0] * x_size_lst[1] * x_size_lst[2] * x_size_lst[3]
    xs = xs.view(x_size_lst[0], total_size // x_size_lst[0]) # 展开成二维张量
    theta = self.fc_loc(xs)
    theta = theta.view(-1, 2, 3)
    grid = F.affine_grid(theta, x.size())
    x = F.grid_sample(x, grid)
    return x

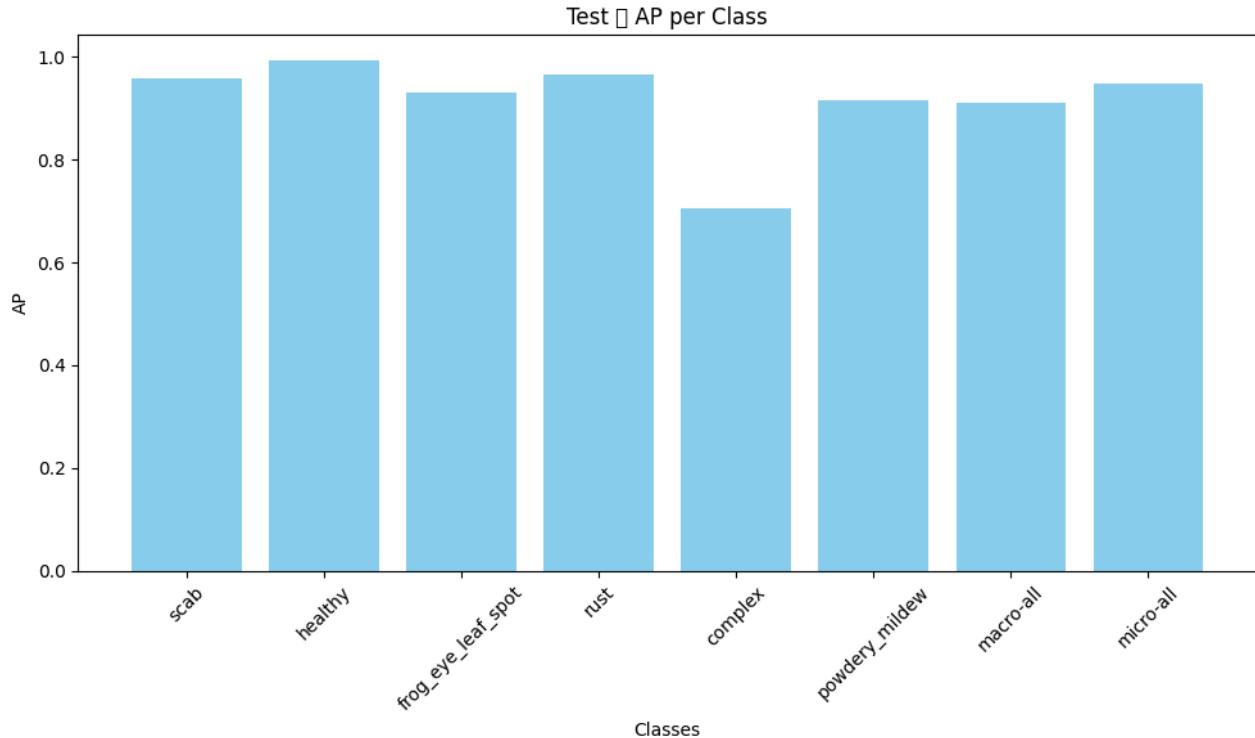
```

在集成了STN模块后，我们对模型进行重新测试和训练，得到了如下结果：

未加入STN模块的模型：



加入STN模块的模型：



可以看到在complex这个特征复杂的类中，加入了STN模块的模型提升显著，说明新增了STN模块后模型能够更好的捕捉到更细节更复杂的图像特征。

4.6 EVA-CLIP的可视化实现

EVA-CLIP模型依赖于注意力机制来捕捉图像中的全局依赖关系。注意力机制通过计算不同图像区域之间的关联性，决定模型在处理当前任务时应关注哪些部分。于是我们引入了[AttentionExtract](#)，从模型中提取注意力图。具体来说，它利用torch.fx的符号跟踪功能，捕捉模型在前向传播过程中各层的注意力权重。这些权重反映了模型在处理图像时对不同区域的关注程度。提取了多层注意力图后，我们参考了Ross Wightman的[timmAttentionViz](#)的实现方法，使用[rollout](#)方法用于整合多层的注意力图，生成一个综合的注意力图。这种方法通过累积各层的注意力权重，生成一个最终的注意力分布，能够更好地捕捉模型对全局特征的关注。

具体的代码实现在[src/attentionvis.py](#)中

我们对每一层都进行了可视化，分别保存为png图像，存放于[/outputs/](#)中，这些png图像最终拼接为一个gif图像，存放于[/gifs](#)文件夹中

可视化的结果如下：



5 评估指标说明

在此先说明我们的评估指标

5.1 1. Precision (精确率, P)

定义:

精确率表示模型预测为正类的样本中，实际为正类的比例。换句话说，它衡量的是模型预测的准确性。

公式:

$$\text{Precision} = \frac{TP}{TP + FP}$$

- **TP (True Positive) :** 真正例，模型正确预测为正类的样本数。
- **FP (False Positive) :** 假正例，模型错误预测为正类的样本数。

意义:

高精确率意味着模型在预测为正类时，错误的概率较低，即模型的假正例较少。

5.2 2. Recall (召回率, R)

定义:

召回率表示实际为正类的样本中，模型正确预测为正类的比例。它衡量的是模型的覆盖能力。

公式:

$$\text{Recall} = \frac{TP}{TP + FN}$$

- **FN (False Negative) :** 假负例，模型错误预测为负类的样本数。

意义:

高召回率意味着模型能够识别出大部分的正类样本，假负例较少。

5.3 3. F1 Score (F1 分数, F1)

定义:

F1 分数是精确率和召回率的调和平均值，综合考虑了两者的平衡。它是一个权衡精确率和召回率的指标。

公式:

$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

意义:

F1 分数在精确率和召回率之间寻找平衡，特别适用于类别不平衡的情况。

5.4 4. Average Precision (平均精度, AP)

定义:

平均精度是基于精确率-召回率曲线 (Precision-Recall Curve) 下的面积。它衡量的是模型在不同阈值下的整体性能。

公式:

$$AP = \int_0^1 P(r) dr$$

意义:

AP 提供了模型在所有可能的阈值下的性能概述，较高的 AP 表明模型在整体上表现良好。

5.5 宏平均 (macro-all) 和微平均 (micro-all) 说明

在多类别或多标签分类任务中，宏平均和微平均是两种常用的聚合方法，用于计算整体的评估指标。

5.5.1 1. 宏平均 (Macro Average)

定义:

宏平均是对每个类别的评估指标（如 Precision、Recall、F1、AP）分别计算后，再取这些指标的简单平均值。每个类别在计算时被赋予相同的权重。

计算方法:

- 对于每个类别，计算 Precision、Recall、F1 和 AP。
- 将所有类别的 Precision、Recall、F1 和 AP 分别取平均。

公式:

$$\text{Macro Average} = \frac{1}{N} \sum_{i=1}^N \text{Metric}_i$$

其中， N 是类别的总数， Metric_i 是第 i 个类别的某一指标。

意义:

宏平均强调各类别的独立表现，适用于类别数量较少且每个类别同等重要的情况。然而，对于类别不平衡的数据集，宏平均可能会被少数类别的表现所主导。

5.5.2 2. 微平均 (Micro Average)

定义：

微平均是将所有类别的 True Positives (TP)、False Positives (FP) 和 False Negatives (FN) 累加后，再计算总体的评估指标。各类别在计算时的权重与其样本数成正比。

计算方法：

- 将所有类别的 TP、FP 和 FN 累加。
- 使用累加后的 TP、FP 和 FN 计算 Precision、Recall 和 F1。

公式：

$$\text{Micro Average Precision} = \frac{\sum_{i=1}^N TP_i}{\sum_{i=1}^N (TP_i + FP_i)}$$

$$\text{Micro Average Recall} = \frac{\sum_{i=1}^N TP_i}{\sum_{i=1}^N (TP_i + FN_i)}$$

$$\text{Micro Average F1} = \frac{2 \times \text{Micro Precision} \times \text{Micro Recall}}{\text{Micro Precision} + \text{Micro Recall}}$$

意义：

微平均考虑了各类别的样本数量，对于类别不平衡的数据集，微平均更能反映整体性能。然而，它可能会掩盖少数类别的表现。

6 训练过程概述

6.1 数据预处理与增强

这里的逻辑在前文已经介绍过，可以[点此翻阅](#)。

6.2 数据加载

数据通过自定义的 `PlantPathologyDataset` 类进行加载，并使用 `DataLoader` 进行批量处理。`DataLoader` 提供了高效的数据读取和批处理机制，支持多线程加载，加快训练过程。

6.3 损失函数与优化器

本项目采用了 `BCEWithLogitsLoss` 作为损失函数，适用于多标签二分类任务。优化器选择了 `Adam`，因其在处理大规模数据和参数时表现出色，能够快速收敛。

- **损失函数：** `BCEWithLogitsLoss` 结合了 Sigmoid 激活和二元交叉熵损失，适合细粒度多标签分类任务。
- **优化器：** Adam 优化器凭借自适应学习率调整能力，能够在复杂的损失曲面上表现优异。
- **学习率调度器：** 使用 `CosineAnnealingLR`：学习率随训练周期呈余弦退火，初期较大，后期逐渐减小，有助于模型在收敛时更细致地调整参数。

6.4 混合精度训练

为了加快训练速度并减少显存占用，项目引入了 **混合精度训练**。通过 `torch.cuda.amp` 中的 `autocast` 和 `GradScaler`，实现了半精度浮点数计算，提升了训练效率，同时保持了模型的精度。

6.5 早停机制

为了防止模型在训练集上过拟合，引入了 **早停机制**。该机制通过监控验证集的 F1 分数，当连续多个 epoch 验证分数未见提升时，提前终止训练。

- **监控指标**: 主要监控验证集的 F1 分数，以衡量模型的整体性能。
- **模型保存**: 每当验证 F1 分数提升时，保存当前最优模型的检查点。
- **提前终止**: 当连续 `patience` 个 epoch 内验证 F1 分数未提升时，提前停止训练，避免过拟合。

6.6 模型保存与加载

训练过程中，最佳模型的检查点被保存，以便后续的评估和部署。通过 `utils.py` 中的 `save_checkpoint` 和 `load_checkpoint` 函数，实现了模型的保存与加载。

- **保存内容**: 包括当前 epoch、模型参数、优化器状态以及验证损失。
- **加载模型**: 在评估和时，使用 `load_checkpoint` 函数加载最佳模型，确保评估的准确性和一致性。

6.7 验证机制

同时，在 `evaluate.py` 中，也设置了为每一类别寻找最佳的 `threshold`，多标签分类任务中，因为不同类别具有不同的特性和难易程度，因此一个统一的阈值可能无法为所有类别提供最佳性能。并且我们保存了最后得到的最佳结果

6.8 训练完成

训练结束后，脚本输出最佳验证 F1 分数及其对应的 epoch，标志着训练过程的完成。

7 实验结果

8 项目结构说明

8.1 目录

`data`

文件夹中需要自己导入图片数据，有处理好的 `processed_train_labels.csv` 等标签文件

`checkpoints`

- 存放了训练好的EVA-CLIP模型权重，需要时可以自行导入
- 存放了test与validation的指标曲线图与表格

`checkpoints_enhanced`

- 存放了优化后的训练好的EVA-CLIP模型权重，需要时可以自行导入
- 存放了优化后的test与validation的指标曲线图与表格

`outputs`

- 主要存放可视化的结果，按照图片ID为索引创建每个图片的注意力可视化结果
- 存放了EfficientNet的可视化gif图片
- `gifs`

存放了注意力可视化拼接后的gif图片

8.2 代码结构

项目代码位于 `src/` 目录下，主要包括以下脚本：

`dataset.py`

负责数据集的加载和预处理。定义了 `PlantPathologyDataset` 类，用于读取图像及其对应的标签，并应用必要的图像变换。

`model.py`

定义了模型结构。通过 `get_model` 函数加载EfficientNet 模型或EVA-CLIP模型。

`train.py`

包含训练和验证的循环逻辑。利用 GPU 进行加速训练，使用混合精度训练技术提高效率，并实现早停机制防止过拟合。

`evaluate.py`

用于在验证集上评估训练好的模型性能。计算损失、F1 分数和准确率等指标，并且在验证集上动态更新模型的 `threshold`，在得到最优的 `threshold` 后，在测试集上进行测试，并且生成对应的曲线

`visualize.py`

应用GradCam，将EfficientNet的运行过程可视化，生成训练过程的热力图

`attentionvis.py`

引入注意力可视化机制，将EVA-CLIP的运行过程可视化，生成训练过程的热力图

`enhanced_modules`

定义STN模块，导入到EVA-CLIP模型中进行优化

`utils.py`

提供辅助功能，包括：

- `calculate_metrics`: 计算 F1 分数和准确率。
- `save_checkpoint`: 保存模型检查点。
- `load_checkpoint`: 加载模型检查点。
- `tensor2img`: 将张量转换为图像格式。
- 其他可视化辅助函数。