作品说明书

一、目标问题与意义价值

1.1 目标问题

我国自古以来便是农业大国，拥有着丰富的农业资源和悠久的农耕文化。农作物的产量与品质对于国家粮食安全的保障、农业经济的稳定增长乃至整个社会的和谐稳定都具有至关重要的意义。然而，随着全球气候变化的加剧以及环境压力的日益增大，农作物病虫害问题愈发凸显，成为了威胁农业生产的重要因素。

根据国家统计局关于2022年粮食产量数据的公告，2022年全国粮食播种面积为118332千公顷，比2021年增加701千公顷，增长0.6%。我国农田面积广阔，农作物种植多。然而，近年来，农作物病虫害的发生频次和面积持续攀升。更为严峻的是，近年来，农作物病虫害的发生频次和面积呈现出不断攀升的趋势。据全国农作物病虫害测报网监测和专家会商分析，预计2024年小麦、水稻、玉米等粮食作物病虫害呈重发态势，发生面积31.5亿亩次，同比增加24%，预计产量损失风险在3600亿斤以上。油菜、大豆重大病虫害发生1.1亿亩次，同比增加15%。病虫害的爆发不仅导致农作物减产甚至绝收，给农民带来巨大经济损失，还严重影响农产品的品质和市场竞争力。因此，如何有效地监测和预防农作物病虫害，确保农业生产的顺利进行，已成为当前农业领域面临的重要课题。

然而，传统的农业疾病监测方式主要依赖人工巡检，这种方式存在诸多弊端。首先，人工巡检效率低下，难以适应现代农业大规模、高效率的生产需求。其次，人工巡检主观性强，准确性难以保证。由于不同人员的经验和技能水平存在差异，对于病虫害的识别和判断往往存在偏差。此外，随着农业规模的不断扩大，人工巡检的覆盖面也愈发有限，很难实现对农田的全面、及时监测。

《“十四五”推进农业农村现代化规划》中指出，政府应将“三农”工作确立为国家发展战略的核心，凸显了农业现代化对于全面建设社会主义现代化国家的极端重要性。这也标志着我国农业农村现代化迈向了一个全新的发展阶段。农业疾病自动监测系统的研发和应用，作为推动农业现代化、提升农业生产效率和质量的关键手段，具有较高的战略价值。

图 2- 1“十四五”推进农业农村现代化规划

因此，开发一种高效、准确的农业疾病自动监测系统具有深远意义。系统能够利用现代科技手段，如深度学习、图像处理等，实现对农作物病虫害的自动识别和监测。相比传统的人工巡检方式，这种系统具有更高的效率和准确性，能够实现对农田的全面覆盖和及时监测。

1.2 意义价值

挑战点一：要实现对农业疾病的精准分类，首先要识别出农作物的叶片，才能针对识别到叶片图像，进行农业疾病的分类。而在针对实际场景的模型训练中，叶片样本受光照、遮挡等因素干扰较大，因此需要在设计视觉识别算法过程中需要考虑这些低质量样本训练效果的影响。

挑战点二：对于农业疾病分类系统，其疾病特征可能不是单方面，而是多尺度、多元的。因此，在设计农业疾病分类算法的过程中，要考虑图像的多尺度特征的提取。

挑战点三：农业疾病分类系统的研发涉及深度学习、图像处理等相关算法和技术。这些模型算法部署到机载电脑可能会涉及到环境的配置以及版本的兼容问题。

挑战点四：用户使用体验。农业疾病自动监测系统的主要面向对象是农民及农业工作者，他们不一定能很熟练地使用系统，因此需要在系统的优化过程中，注重用户的体验。

二、设计思路与方案

2.1 边缘端Jetson Xavier nx的环境配置

1.使用虚拟机ubuntu20.04作为HOST端下载SDK-Manager,并选择JetPack5.1.1,将会使Jetson xavier nx为arm64架构的系统;

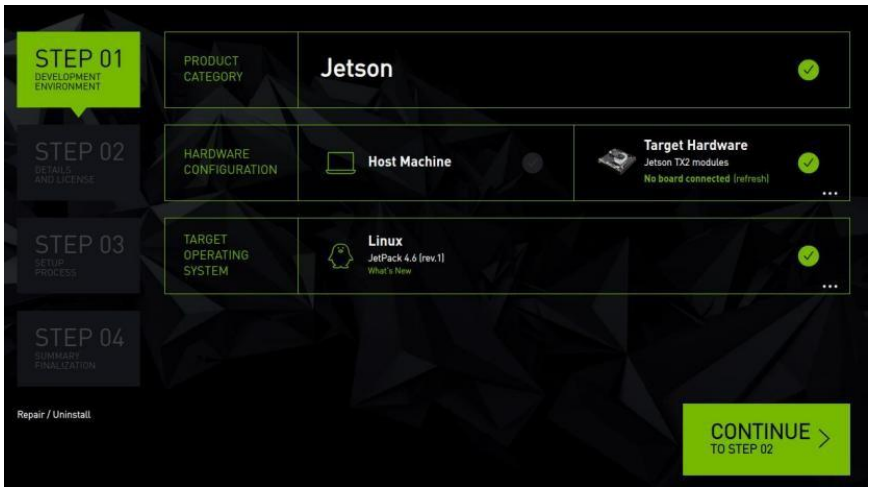


图 5-1 Jetson 部署系统

2.将Jetson xavier nx更新设备树与刷机，使用 USB Type-C 线连接载板上的 USB OTG 接口,再将开机键拨至 MP（手动开机），摁住 REC 键，再摁 PWR 开机，松开 REC 键进 入 Recovery 模式，此时 VMWare 右下角会出现 NVIDIA 的 USB 驱动标志，或者 打开终端，输入 lsusb 命令，会发现Nvidia Corp;

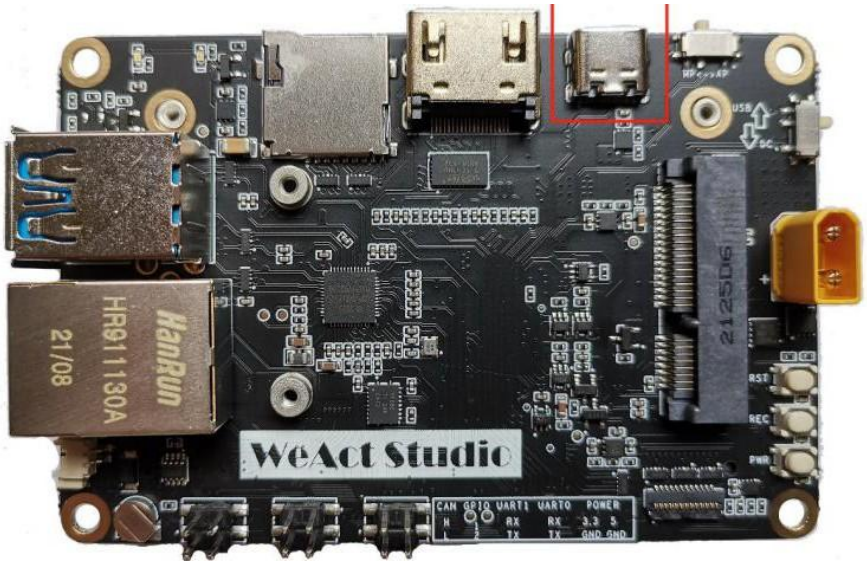


图 5-2 Jetson 更新设备树与刷机

3. 安装NVIDIA组件，如CUDA、深度加速引擎等；

图 5- 3 安装NVIDIA组件

4. 将系统迁移至SD卡,将系统启动文件迁移至SD卡，同时启动emmc引导完成基础配置；

```
[   30.4511 ] Coldbooting the device
[   30.4521 ] tegradevflash_v2 --reboot coldboot
[   30.4531 ] Bootloader version 01.00.0000
[   30.6253 ]
*** The target t186ref has been flashed successfully. ***
Make the target filesystem available to the device and reset the board to boot f
rom external mmcblk1p1.
```

图 5-4 系统迁移至SD卡

2.2 使用改进Yolov5对数据集进行训练

本团队共拍摄农业叶片数据集600张，其中农作物叶片训练集500张图片，测试集100张图片，部分数据及标注如下图所示。

选定数据集后，对数据进行预处理，为了提高模型的泛化能力，对图像进行归一化、去噪等预处理操作，消除图像中的无关信息和噪声。

接着，使用labellmg对图像进行标注，得到.xml文件后开始配置参数，采用基于改进yolov5的Wise-IoU方法对数据集进行训练，运行train.py文件训练数据集。

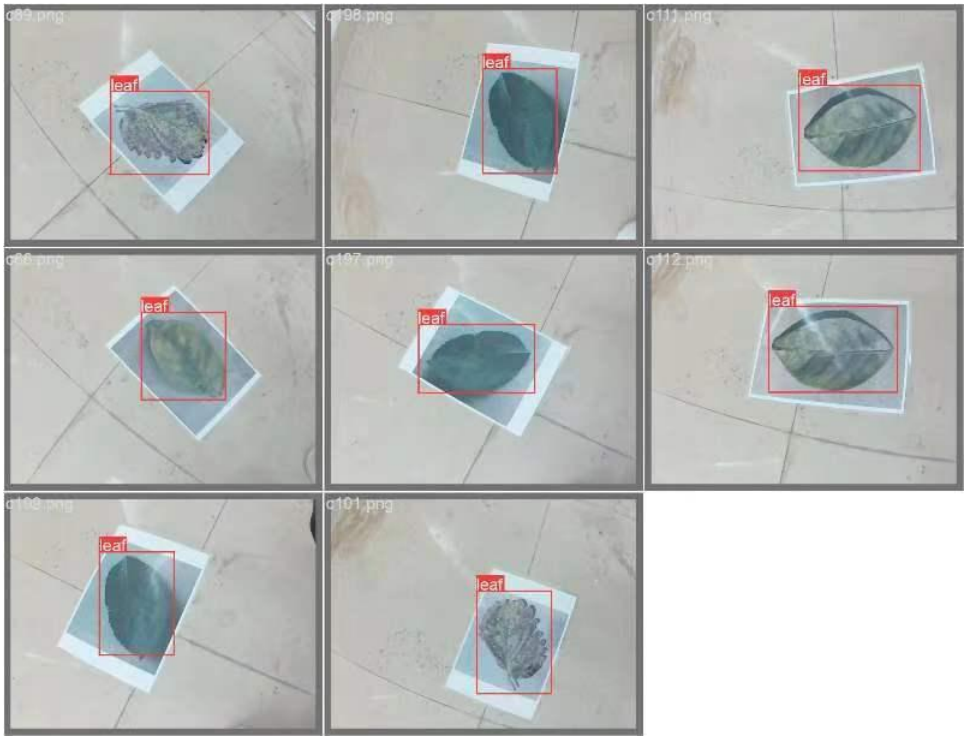


图 5-5 Yolov5拍摄数据集

```
def parse_opt(known=False):
    """Parses command-line arguments for YOLOv5 training, validation, and testing."""
    parser = argparse.ArgumentParser()
    parser.add_argument("--weights", type=str, default='weights/best', help="initial weights path")
    parser.add_argument("--cfg", type=str, default='models/yolov5s.yaml', help="model.yaml path")
    parser.add_argument("--data", type=str, default=ROOT / "data/leaf.yaml", help="dataset.yaml path")
    parser.add_argument("--hyp", type=str, default=ROOT / "data/hyps/hyp.scratch-low.yaml", help="hyperparameters path")
    parser.add_argument("--epochs", type=int, default=1000, help="total training epochs")
    parser.add_argument("--batch-size", type=int, default=4, help="total batch size for all GPUs, -1 for autobatch")
    parser.add_argument("--imgsz", "--img", "--img-size", type=int, default=640, help="train, val image size (pixels)")
    parser.add_argument("--rect", action="store_true", help="rectangular training")
    parser.add_argument("--resume", nargs="?", const=True, default=False, help="resume most recent training")
    parser.add_argument("--nosave", action="store_true", help="only save final checkpoint")
    parser.add_argument("--noval", action="store_true", help="only validate final epoch")
    parser.add_argument("--noautoanchor", action="store_true", help="disable AutoAnchor")
    parser.add_argument("--noplots", action="store_true", help="save no plot files")
    parser.add_argument("--evolve", type=int, nargs="?", const=300, help="evolve hyperparameters for x generations")
    parser.add_argument(
        "--evolve_population", type=str, default=ROOT / "data/hyps", help="location for loading population"
    )
```

图 5-6 Yolov5训练参数局部代码

训练2000层epochs后，权重和训练日志保存到runs/train/exp文件夹下，使用tensorboard打开，查看训练好的权重及可视化日志。

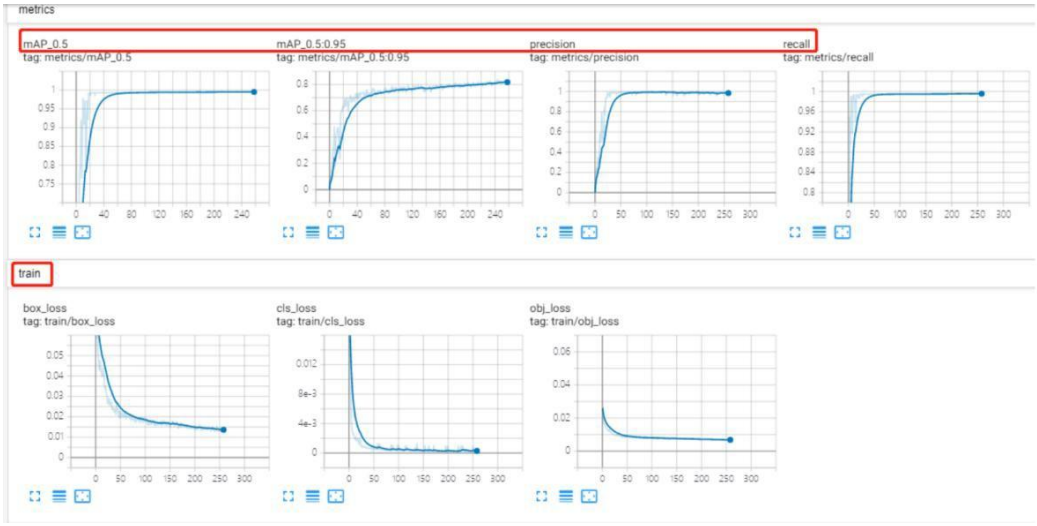


图 5-7 Yolov5训练完成准确率图

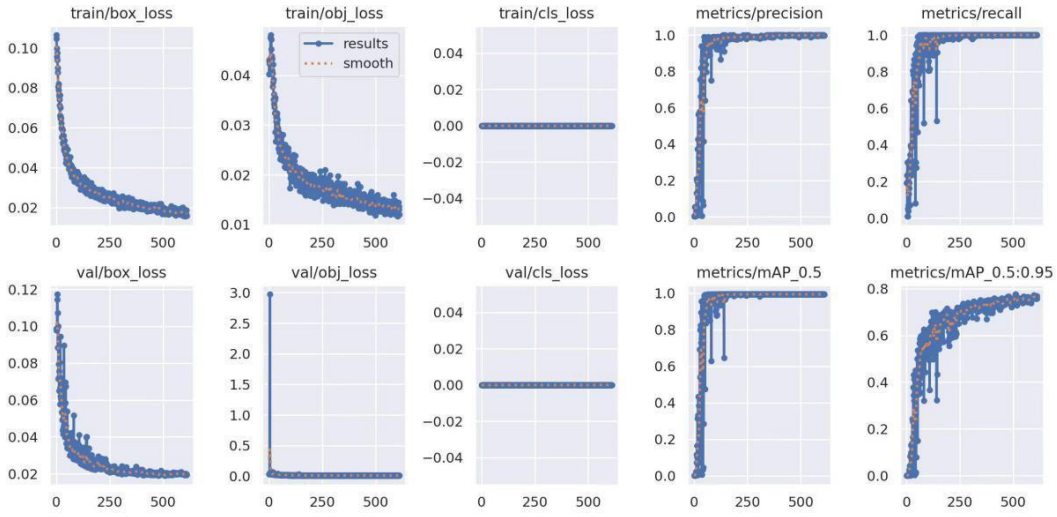


图 5-8 Yolov5训练完成准确率图

此外，我们还对比了本目标检测算法和其他相关算法,其中包括F-RCNN和Yolov3算法，如下图可见，本技术的农作物识别速度快、准确率高，分别达到140FPS和96.78%。

表 1 Yolov5与其他算法对比

模型	速度	准确率
F-RCNN	20FPS	96.55%

Yolov3	120FPS	95.88%
Yolov5	140FPS	96.78%

将本方法训练得到的权重best.pt使用100张验证集进行实际验证，实际验证效果如下：

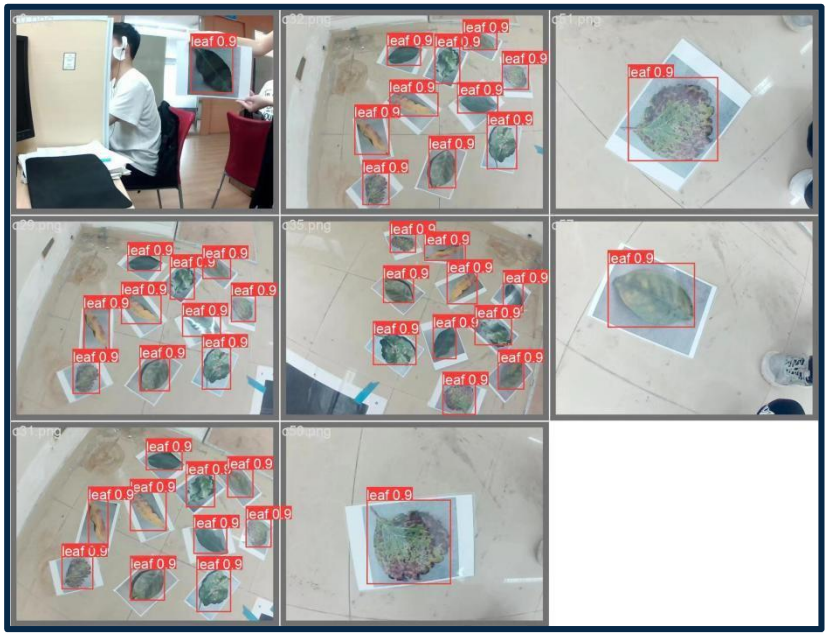


图 5-9 实际验证效果

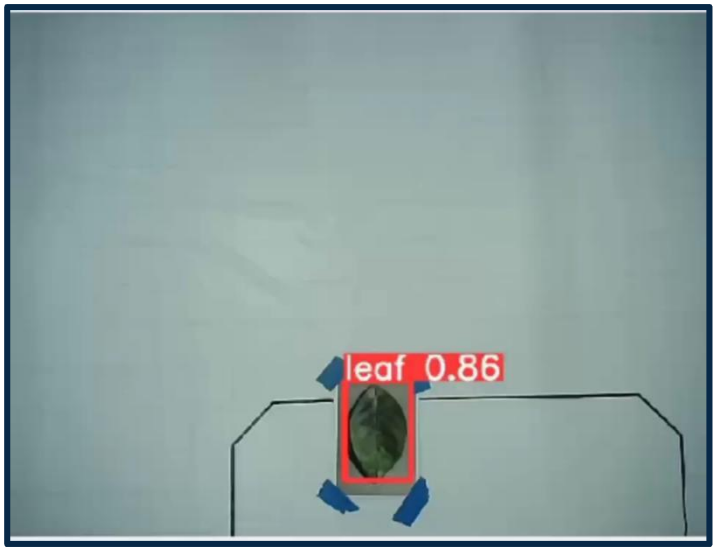


图 5-10 实际验证效果

2.3 基于cross-attention对数据进行分类训练

本团队依据本届中国计算机设计大赛官网所给农业叶片数据集，经过数据清洗后得出有效数据集，具体如下：将农业叶片数据划分为数据集与验证集，总共划分为29种农业疾病类型，其中数据集共24688张图片，验证集3553张图片，数据集，部分数据及如下图所示。

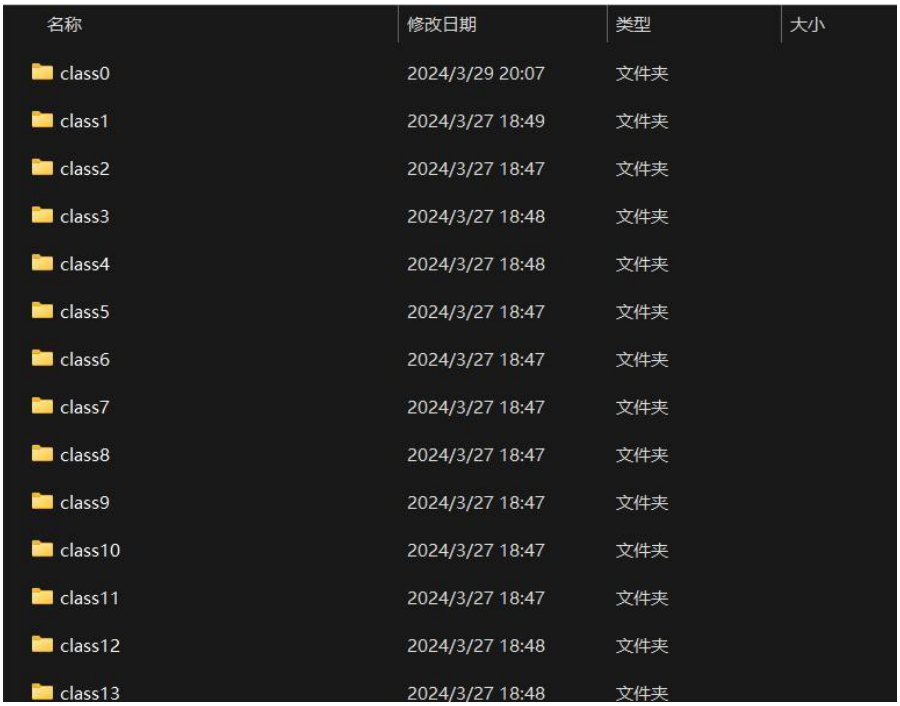


图 5-11 分类数据集文件目录

使用pytorch架构开始训练，导入库文件，关键训练参数如下所示:

```python
def get_args_parser():
    parser = argparse.ArgumentParser('CrossViT training and evaluation script', add_help=False)
    parser.add_argument('--batch-size', default=34, type=int)
    parser.add_argument('--epochs', default=350, type=int)

    # Model parameters
    parser.add_argument('--model', default='crossvit_small_224', type=str, metavar='MODEL',
                        help='Name of model to train')
    parser.add_argument('--input-size', default=240, type=int, help='images input size')

    parser.add_argument('--drop', type=float, default=0.0, metavar='PCT',
                        help='Dropout rate (default: 0.)')
    parser.add_argument('--drop-path', type=float, default=0.1, metavar='PCT',
                        help='Drop path rate (default: 0.1)')
    parser.add_argument('--drop-block', type=float, default=None, metavar='PCT',
                        help='Drop block rate (default: None)')
    parser.add_argument('--pretrained', action='store_true', help='load imagenet1k pretrained model')

    # Optimizer parameters
    parser.add_argument('--opt', default='adamw', type=str, metavar='OPTIMIZER',
                        help='Optimizer (default: "adamw"')
    parser.add_argument('--opt-eps', default=1e-8, type=float, metavar='EPSILON',
                        help='Optimizer Epsilon (default: 1e-8)')
    parser.add_argument('--opt-betas', default=None, type=float, nargs='+', metavar='BETA',
```

图 5-12 分类模型关键训练参数局部代码

创建交叉注意力模型，将训练集与验证集导入；

```python
print(f"Creating model: {args.model}")
model = create_model(
    args.model,
    pretrained=args.pretrained,
    num_classes=args.nb_classes,
    drop_rate=args.drop,
    drop_path_rate=args.drop_path,
    drop_block_rate=args.drop_block,
)
```

图 5-13 分类模型训练集与验证集导入局部代码

使用笔记本电脑单GPU训练，并将输出权重实时保存；

```python
if args.initial_checkpoint:
    print("Loading pretrained model")
    checkpoint = torch.load(args.initial_checkpoint, map_location='cpu')
    utils.load_checkpoint(model, checkpoint['model'])

if args.auto_resume:
    if args.resume == '':
        args.resume = str(output_dir / "checkpoint.pth")
        if not os.path.exists(args.resume):
            args.resume = ''

if args.resume:
    if args.resume.startswith('https'):
        checkpoint = torch.hub.load_state_dict_from_url(
            args.resume, map_location='cpu', check_hash=True)
    else:
        checkpoint = torch.load(args.resume, map_location='cpu')
    utils.load_checkpoint(model, checkpoint['model'])
    if not args.eval and 'optimizer' in checkpoint and 'lr_scheduler' in checkpoint and 'epoch'
        optimizer.load_state_dict(checkpoint['optimizer'])
        lr_scheduler.load_state_dict(checkpoint['lr_scheduler'])
        args.start_epoch = checkpoint['epoch'] + 1
        if 'scaler' in checkpoint and args.resume_loss_scaler:
            print("Resume with previous loss scaler state")
            loss_scaler.load_state_dict(checkpoint['scaler'])
```

图 5-14 分类模型输出权重实时保存局部代码

使用验证集实时测试准确率，并调整学习率；

```python
def evaluate(data_loader, model, device, world_size, distributed=False, amp=False):
    criterion = torch.nn.CrossEntropyLoss()

    metric_logger = utils.MetricLogger(delimiter="  ")
    header = 'Test:'

    # switch to evaluation mode
    model.eval()

    outputs = []
    targets = []

    for images, target in metric_logger.log_every(data_loader, 10, header):
        images = images.to(device, non_blocking=True)
        target = target.to(device, non_blocking=True)
        # compute output
        with torch.cuda.amp.autocast(enabled=amp):
            output = model(images)

        if distributed:
            outputs.append(concat_all_gather(output))
            targets.append(concat_all_gather(target))
        else:
            outputs.append(output)
            targets.append(target)
```

图 5-15 分类模型调整学习率局部代码

将epoch 350层训练完后，准确率最高的权重保存至model_best.pth中；本分类模型分类效果准确率达到90.4%。

```python
if args.output_dir:
    checkpoint_paths = [output_dir / 'checkpoint.pth']
    if test_stats["acc1"] == max_accuracy:
        checkpoint_paths.append(output_dir / 'model_best.pth')
    for checkpoint_path in checkpoint_paths:
        state_dict = {
            'model': model.state_dict(),
            'optimizer': optimizer.state_dict(),
            'lr_scheduler': lr_scheduler.state_dict(),
            'epoch': epoch,
            'args': args,
            'scaler': loss_scaler.state_dict(),
            'max_accuracy': max_accuracy
        }
        utils.save_on_master(state_dict, checkpoint_path)

log_stats = {**{f'train_{k}': v for k, v in train_stats.items()},
             **{f'test_{k}': v for k, v in test_stats.items()},
             'epoch': epoch,
             'n_parameters': n_parameters}
```

图 5-16 分类模型权重保存局部代码

模型权重与参数训练完毕后，存放入边缘端，边缘端配好环境后，只需创建交叉注意力模型，加载权重，输出结果即可；

图 5-17 分类模型效果测试

三、方案实现

3.1 技术方案

目前农业疾病监测存在人工巡检效率低、人工农业疾病识别对经验要求高等问题。针对这些问题，本项目提出了基于Cross-attention机制的无人机农业疾病分类系统。用户仅需在用户界面按下按键，无人机即可实现一键起飞，实现农田的自动巡检；在巡检的过程当中，无人机上搭载摄像头获取实时图像，并使用改进Yolov5对图像中的农作物叶子进行识别；基于Cross-attention农业疾病分类系统将对识别到的叶子图像进行处理，返回农作物的疾病类型；无人机根据识别到的疾病类型喷洒对应的农药，实现对症下药。

系统的整体框图如下：

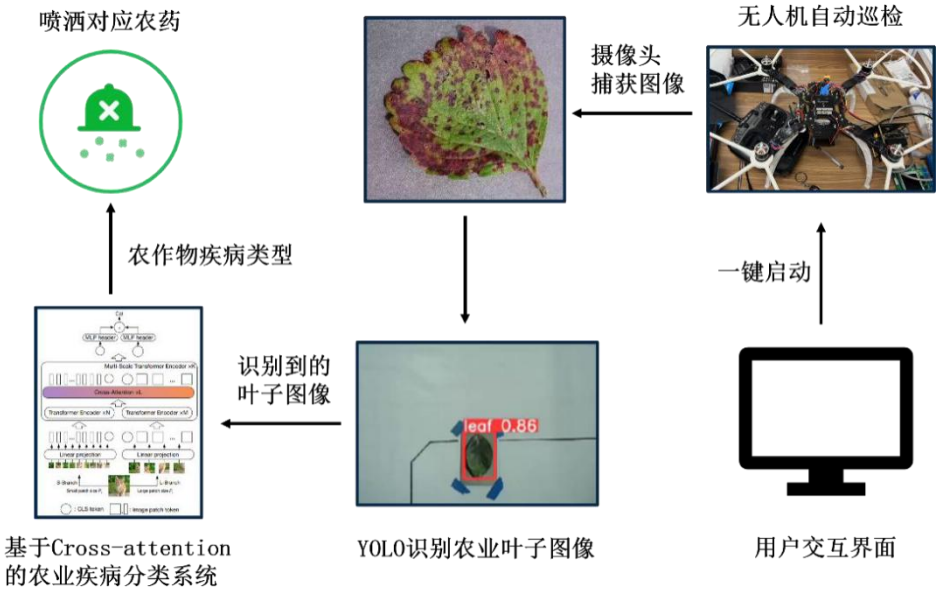


图 3-1 系统整体框图

针对人工巡检效率低的问题，本项目提出了使用无人机对农田进行自动巡检，并根据疾病类型对症下药的解决方案。无人机飞行速度快，成本相对较低，方便用于监测。使用无人机对农田进行自动巡检可以大大缩短巡检所需时间，并且由于其是从高处俯视，覆盖面更广。在识别出具体的疾病类别后，无人机还能自主根据疾病类型，控制电机喷洒对应农药，实现对症下药。

针对人工识别农业疾病的问题，本项目使用yolov5对农作物叶子进行识别，并使用Wise-IoU的方式对yolov5模型中的损失函数进行改进，使得农作物叶子识别更为精准；识别到叶子的图片后，基于cross-attention机制对农作物疾病进行分类识别。相较于人工对农业疾病进行识别，改技术更为科学可靠。

3.2 系统实现

3.2.1 基于边缘端Jetson Xavier nx的无人机环境配置

1. 使用Ubuntu 20.04 安装SDK 并将系统烧录至Jetson Xavier NX 边缘智能设备；

2. 将CUDA、加速引擎、GPU等安装至Jetson Xavier NX 边缘智能设备；

3. PX4飞控连接QGC地面站，修改飞行参数；

4. Jetson Xavier NX安装Yolov5、Pytorch等深度学习模型；

5. Jetson Xavier NX安装SDK-Realsense连接T265双目摄像头；

6. Jetson Xavier NX安装Mavros、Mavlink等进行定位数据传输；

7. Jetson Xavier NX 下载编译Cross-Attention农业疾病分类代码、Yolov5目标检测代码、无人机飞行轨迹代码；

3.2.2基于改进Yolov5农作物叶子识别技术

A.Yolov5模型原理

YOLOv5是一种目标检测算法，其模型结构主要包括以下组成部分：

1. 输入端：YOLOv5的Head网络由3个不同的输出层组成，分别负责检测大中小尺度的目标；主要由Mosaic 图像增强、自适应锚框计算以及自适应图片缩放 组成

2. Backbone网络：YOLOv5使用CSPDarknet53作为其主干网络，其具有较强的特征提取能力和计算效率。

3. Neck网络：YOLOv5使用的是FPN(FPN网络能够在不同的特征图层次上进行检测，可以提高目标检测的性能)网络，可以融合来自不同特征图层次的信息。

4. 输出端：损失函数，YOLOv5使用的是Focal Loss损失函数，该函数可以缓解目标检测中类别不平衡的问题，提高模型的性能。非极大值抑制（NMS），YOLOv5在输出结果后，会对重叠的目标框进行NMS处理，以得到最终的检测结果。

5. 激活函数，YOLOv5使用的是Mish激活函数，该函数是一种替代ReLU的激活函数，可以提高模型的性能。

总体来说，YOLOv5的模型结构相对简单，但其使用了多种技术和策略，如CSP结构、FPN网络、Mish激活函数和Focal Loss损失函数等，以提高模型的性能和鲁棒性。

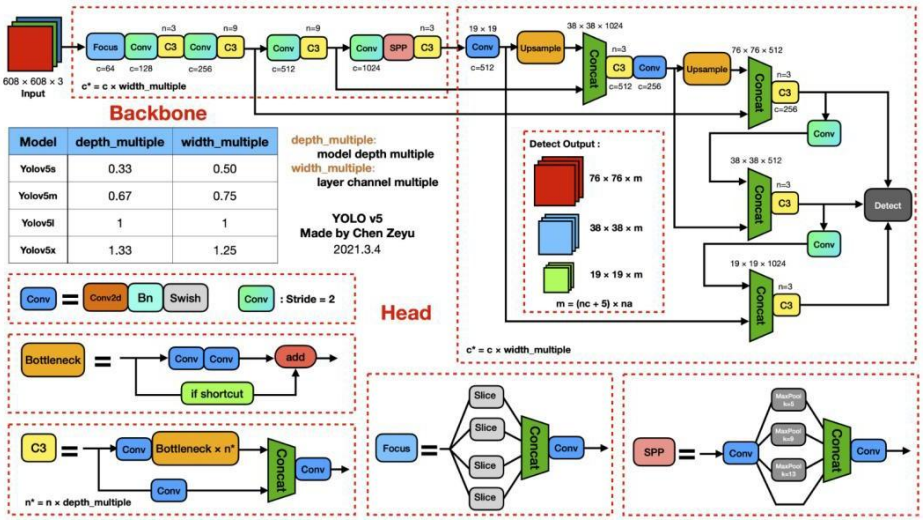


图 4-1 Yolov5模型结构图

B.基于Yolov5的改进

1.概述

目标检测作为计算机视觉的核心问题，其检测性能依赖于损失函数的设计。边界框损失函数作为目标检测损失函数的重要组成部分，其良好的定义将为目标检测模型带来显著的性能提升。

但我们注意到，如果一味地强化边界框对低质量示例的回归，显然会危害模型检测性能的提升。因此，我们提出了动态非单调的聚焦机制，设计了Wise-IoU (WIoU)。动态非单调聚焦机制使用“离群度”替代 IoU 对锚框进行质量评估，并提供了明智的梯度增益分配策略。

该策略在降低高质量锚框的竞争力的同时，也减小了低质量示例产生的有害梯度。这使得 WIoU 可以聚焦于普通质量的锚框，并提高检测器的整体性能。将WIoU应用于YOLOv5 时，并在 MS-COCO 数据集上进行测试， AP-75 从 53.03% 提升到 54.50%。

2.改进方法

WIoU首先构造了基于注意力的边界框损失函数，并在此基础上通过构造梯度增益 (聚焦系数) 的计算方法来附加聚焦机制。

因为训练数据中难以避免地包含低质量示例，所以如距离、纵横比之类的几何度量都会加剧对低质量示例的惩罚从而使模型的泛化性能下降。好的损失函数应该在锚框与目标框较好地重合时削弱几何度量的惩罚，不过多地干预训练将使模型有更好的泛化能力。在此基础上，我们根据距离度量构建了距离注力，得到了具有两层注意力机制的 WIoU：

$R_{WIoU} \in [1, e)$，这将显著放大普通质量锚框的L_{IoU}

$L_{IoU} \in [0,1]$，这将显著降低高质量锚框的R_{WIoU}，并在锚框与目标框重合较好的情况下显著降低其对中心点距离的关注

$$L_{WIoU} = R_{WIoU} L_{IoU}$$

$$R_{WIoU} = \exp\left(\frac{(x - x_{gt})^2 + (y - y_{gt})^2}{(W_g^2 + H_g^2)^*}\right),$$

为了防止 R_{WIoU} 产生阻碍收敛的梯度，将W_g ,H_g从计算图中分离 (上标 * 表示此操作)。因为它有效地消除了阻碍收敛的因素，所以我们没有引入新的度量指标，如纵横比。

定义离群度以描述锚框的质量，其定义为：

$$\beta = \frac{L_{IoU}^{*}}{L_{IoU}} \in [0, +\infty]$$

离群度小意味着锚框质量高，我们为其分配一个小的梯度增益，以便使边界框回归聚焦到普通质量的锚框上。对离群度较大的锚框分配较小的梯度增益，将有效防止低质量示例产生较大的有害梯度。我们利用δ 构造了一个非单调 聚焦系数并将其应用于 WIoU：

$$r = \frac{\beta}{\delta \alpha^{\beta-\delta}}$$

其中，当$\beta = \delta$时,当锚框的离群程度满足 $\beta = C$ (C为定值)时，锚框将获得最高的梯度增益。由于L_{IoU}是动态的，锚框的质量划分标准也是动态的，这使得 WIoU在每一时刻都能做出最符合当前情况的梯度增益分配策略.

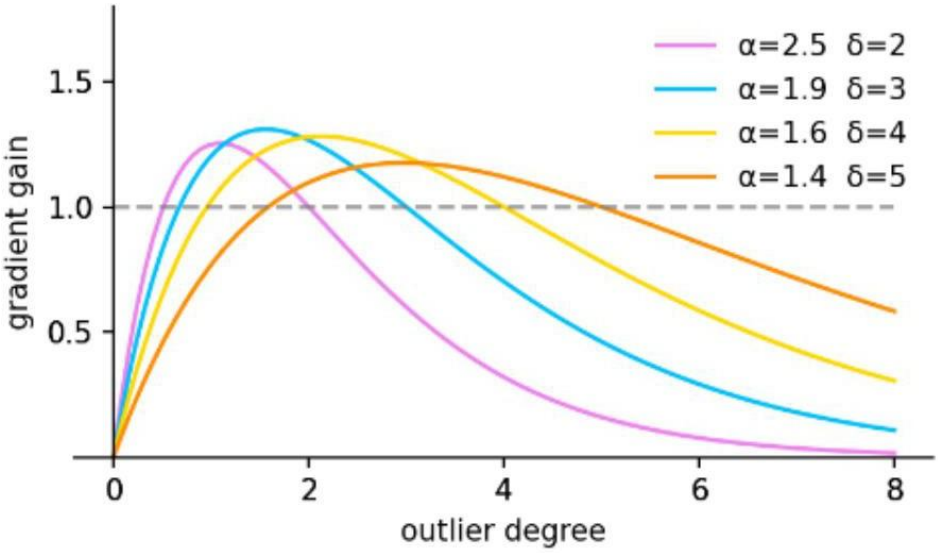


图 4-2 不同α 和δ 下的β 和r 的关系图

3.2.3 基于cross-attention的农业疾病分类技术

A.视觉变换器

视觉变换器（Vision Transformer，简称ViT）是一种基于Transformer架构的计算机视觉模型。它最早由Alexey Dosovitskiy等人在2020年提出。ViT是将自然语言处理领域的Transformer架构引入到计算机视觉领域的一种尝试。ViT的基本工作原理是将图像分割成若干个固定大小的图像补丁（patch），然后将每个补丁展平成一维向量。每个补丁的维度与Transformer模型中输入的词嵌入（word embedding）维度保持一致。接下来，ViT将这些图像补丁的嵌入序列输入到一个标准的Transformer编码器中。编码器处理这些嵌入序列，通过自注意力机制捕捉不同补丁之间的关系。最终，编码器输出一个用于分类或其他任务的特征表示。

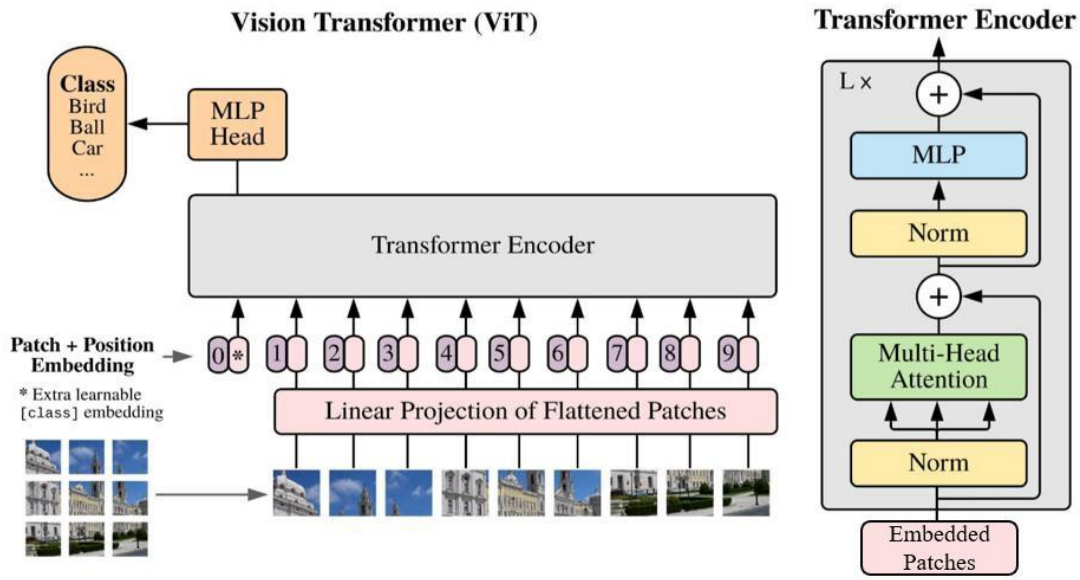


图 4-3 ViT 系统框图

视觉变换器应用于分类任务时，需要在序列中添加一个额外的分类输入输出维度，由于Transformer编码器中的自注意是与位置无关的，而视觉应用高度需要位置信息，因此ViT向每个输入维度添加了位置编码，包括CLS维度。然后，将所有token输入到堆叠的Transformer编码器，最后使用CLS token进行分类。Transformer编码器由一系列块组成，其中每个块包含带有FNN和多头自注意(MSA)。FFN在隐藏层上包含具有扩展比为r的两层多层感知器，在第一个线性层后应用一个GELU非线性激活函数。在每个块之前应用层归一化(LN)，在

每个块之后应用残差连接。因此，VIT的输入和第k个块的输出之间的关系可用下列式子表示：

$$x_0 = [x_{CLS} \| x_{patch}] + x_{pos}$$
$$y_k = x_{k-1} + MSA(LN(x_{k-1}))$$
$$x_k = y_k + FFN(LN(y_k))$$

B.多尺度变换器

本文将视觉变换器应用于疾病分类中，并在Transformer模型中加入了多尺度特征表示。尺度的大小会影响ViT的准确性和复杂性。所谓多尺度，实际就是对信号的不同粒度的采样。通常在不同的尺度下我们可以观察到图片不同的特征，从而完成不同的任务。

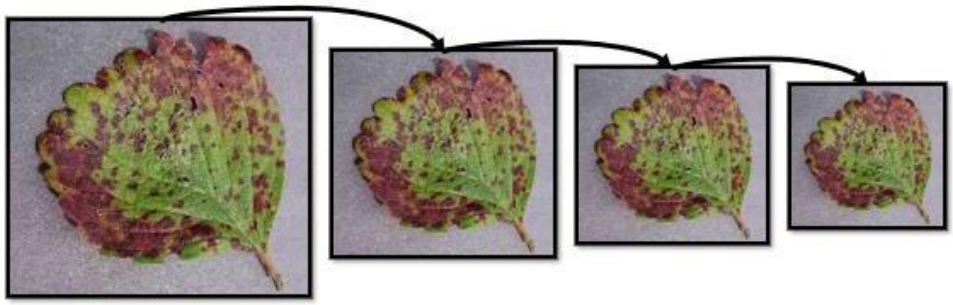


图 4-4 图片下采样

粒度更小、更密集的采样可以看到更多的细节，粒度更大、更稀疏的采样可以看到整体的趋势。使用多尺度，就可以提取更全面的信息，既有全局的整体信息，又有局部的详细信息。

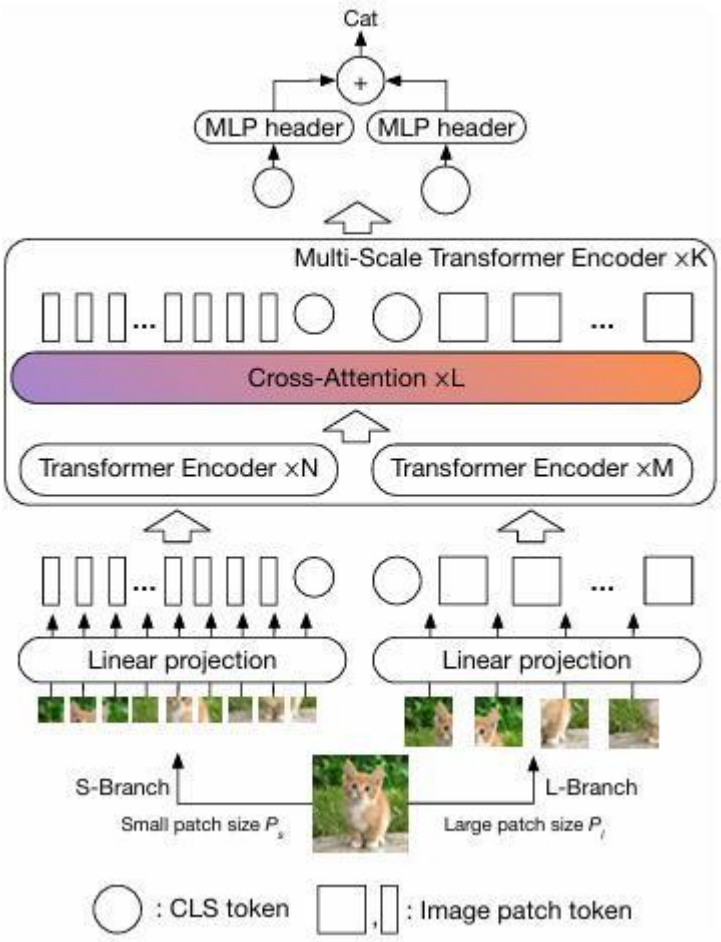


图 4-5 多尺度视觉网络结构

上图为本文使用的多尺度视觉网络结构，模型主要由k个多尺度Transformer编码器组成，其中每个编码器由两个分支组成：L-Branch：大分支利用粗粒度的图片进行操作，使用更多的Transformer编码器和更大的编码维度。S-Branch：小分支对细粒度的图片进行操作，具有更少的编码器和更小的编码维度。两个分支的输出特征在交叉注意力机制中进行融合，利用末端的两个分支对分类结果进行预测。

C.多尺度特征融合机制

有效的特征融合是学习多尺度特征表示的关键。在本文中，本文的结构采用了cross-attention的融合机制：

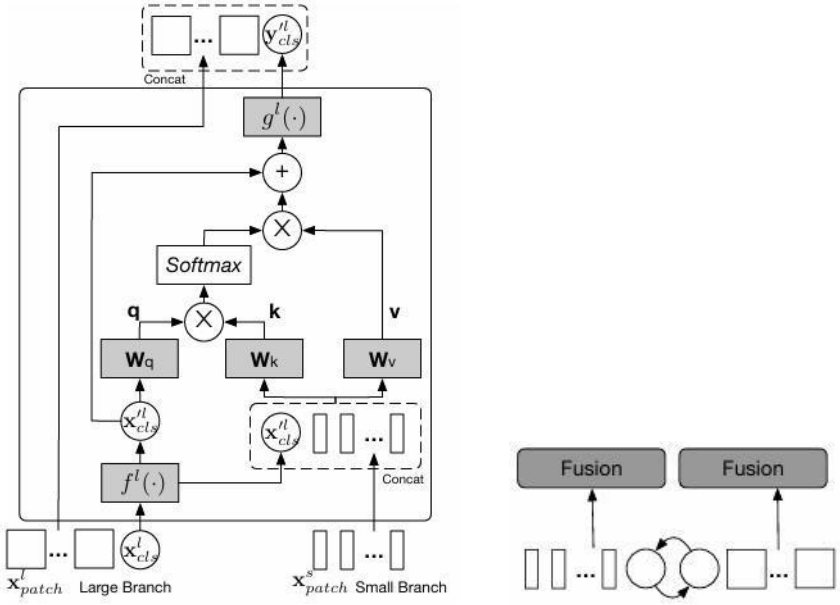


图 4- 6 cross-attention的融合过程

设是分支i上的序列(包括patch和CLS维度)，其中i可以是l或者s，分别代表大分支和小分支。圆和矩形分别代表分支i的CLS维度和patch。上图显示了本文提出的Cross-Attention Fusion的基本思想，其中融合涉及到一个分支的CLS维度和另一个分支的patch维度。此外，为了更有效地融合多尺度特征，首先利用每个分支的CLS维度作为代理，在另一个分支的patch维度之间交换信息，然后将其投影到自己的分支。

$$x'^{l} = [f^{l}(x^{l}_{CLS}) \| x^{s}_{patch}]$$

其中，$f^{l}(.)$是用于特征对齐的函数。然后，在该模块之中执行交叉注意力（CA）：

$$q = x'^{l}_{CLS} W_q, k = x'^{l} W_k, v = x'^{l} W_v,$$
$$A = soft\max\left(qk^{T}/\sqrt{C/h}\right),\ CA(x'^{l}) = Av$$

由于CLS维度已经学习了自己分支中所有patch维度之间的抽象信息，因此与另一个分支中的patch维度的交互有助于融合不同尺度的信息。与其他分支token融合后，CLS维度在下一层Transformer编码器上再次与自己的patch维度交互，在这一步中，它又能够将来自另一个分支的学习信息传递给自己的patch维度，以丰富每个patch维度的表示。

此外，我们的使用也在CA中使用多个注意力头，具体来说，CA的输出如下：

$$y_{CLS}^{l} = f^{l}(x_{CLS}^{l}) + MCA\left(LN\left(\left[f^{l}(x_{CLS}^{l}) \| x_{patch}^{s}\right]\right)\right)$$
$$z^{l} = \left[g^{l}(y_{CLS}^{l}) \| x_{patch}^{l}\right]$$

3.2.4 无人机自动巡检系统

A.无人机架构组成

1. 开源飞控 Pixhawk 6C (无人机控制系统)
2. Intel Realsense T265 双目鱼眼摄像头（无人机定位系统）
3. Jetson Xavier NX 边缘智能设备 (边缘端计算系统)
4. LIDAR-Lite v3 激光测距仪 (无人机定高系统)
5. 单目摄像头 (摄像头拍摄系统)
6. STM32-C8T6 嵌入式设备 (喷洒农药系统)
7. LTE-Link SE 4G图传数传一体

图 4-7 无人机结构图

B.自动巡检飞行功能

1. 将农田区域进行划分，将农田的巡检轨迹编写入无人机的边缘端设备；

2. 用户登录UI界面，开启无人机巡检功能，检查是否正常打开定位、目标追踪、农业分类等功能；

3. 用户通过UI界面，一键起飞，无人机自动巡检，并将拍摄的画面通过图传实时传输至UI界面，用户可观察农田画面；

4. 无人机自动巡检中，在划分的每个区域内停留并使用Yolov5对农作物进行目标追踪，最后将拍摄的画面进行深度学习分析，得出患病结果，并将结果通过数传实时传入到UI界面；

5. 无人机巡检完毕后，自主降落，并上锁保证用户安全；

C.无人机喷洒农药功能

1. 无人机将搭载多种典型农药进行自主巡检；

2. 无人机通过深度学习得到的农业患病结果，将结果通过串口通信传入至stm32-c8t6，经过触发中断打开不同的农药喷雾口，实现精准对症喷洒；

3.2.5 无人机飞行状态监测系统

A.基于4G网络与LTE LINK SE实现实时数据和图像传输

为了实现对飞控系统的实时监控与控制，我们采用了基于4G网络与LTE LINK SE的数据传输方案。此方案的核心在于通过4G网络建立稳定的通信链路，使地面站能够实时接收到无人机的飞行数据和图像信息。具体来说，我们选择了CUAV开发的"非攻透传"软件作为数据传输的关键工具，该软件利用UDP协议实现了飞行数据和图像的传输，确保了数据的及时性和准确性。在"非攻透传"软件上，我们在绑定无人机后，对于数据传输和视频传输分别设置两个端口来供地面站和进行交互的主机来接收。

图 4-8 “非攻透传”界面

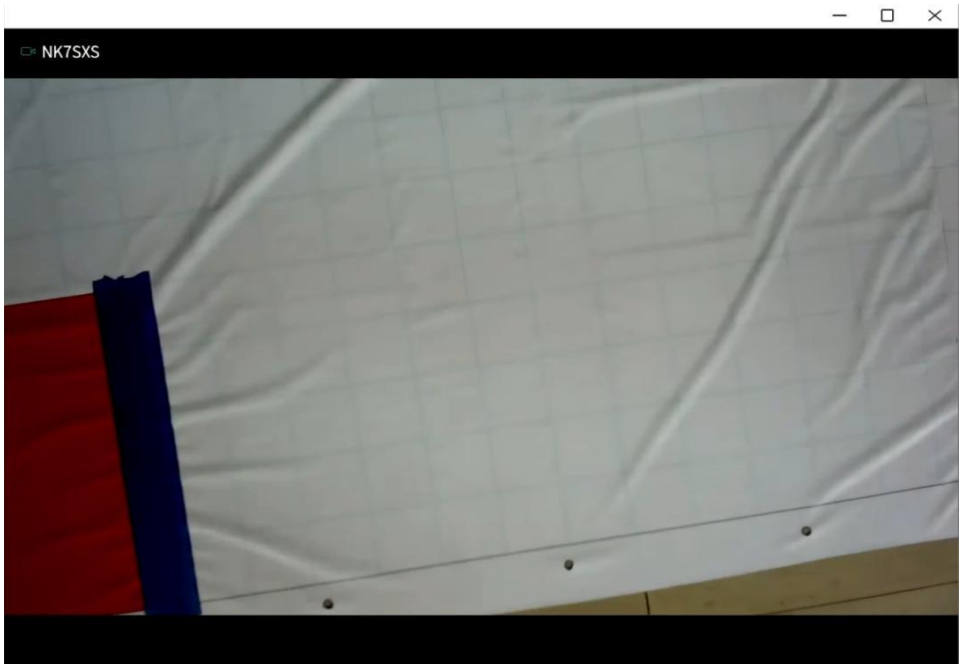


图 4-9 “非攻透传”实时画面显示

在地面站上，我们设置相应的端口号以接收来自无人机的飞行高度、速度、角度等关键信息，同时也能实时显示飞行画面。地面站可以通过分析这些飞行数据来实时评估无人机的飞行状况，并根据实时数据进行相应的调整和控制，以确保飞行器的安全飞行和稳定运行。同时，利用实时显示的飞行画面，操作

人员可以进行远程导航和飞行监控，及时发现潜在的风险因素并采取措施进行应对，从而提高飞行操作的效率和安全性。

图 4- 10 QGC地面站接收数据

此外，我们在LTE LINK SE上另外创建了一个端口，利用Python中的pymavlink库实现了对飞行控制系统发送的状态信息和传感器数据的接收，并将其实时显示在用户交互界面上。这个综合的数据传输方案不仅确保了飞行数据的及时传输和准确性，也为用户提供了便捷的飞行监控与控制手段。

图 4- 11 用户交互界面接收数据

B.用户交互界面

图 4-12 用户登录界面

用户通过输入正确的账号密码，登录无人机农业疾病分类系统的用户交互界面。

图 4-13 用户交互界面

交互平台由四个部分组成，分别是飞行状态、轨迹监测、实时画面和其他帮助。其中，“其他帮助”模块主要是对我们交互平台的操作以及无人机农业疾病分类系统进行了简要介绍。

在飞行状态界面，我们可以获取并显示无人机的飞行速度、飞行高度、信号强度、电池状态、摄像头状态等信息。点击“开始”按钮可向无人机发送指

令，无人机接收到指令后开始工作，实现远程控制无人机工作的功能，并显示无人机飞行过程中的各状态信息；

图 4-14 用户飞行状态界面

点击“校准”按钮可以对无人机的飞行角度和飞行高度进行校准，以当前的角度和飞行高度作为初始值，后续将基于该初始值进行实时的状态显示；“停止”按钮是避免无人机出现突发情况，让无人机在合适的地点停止工作并降落。

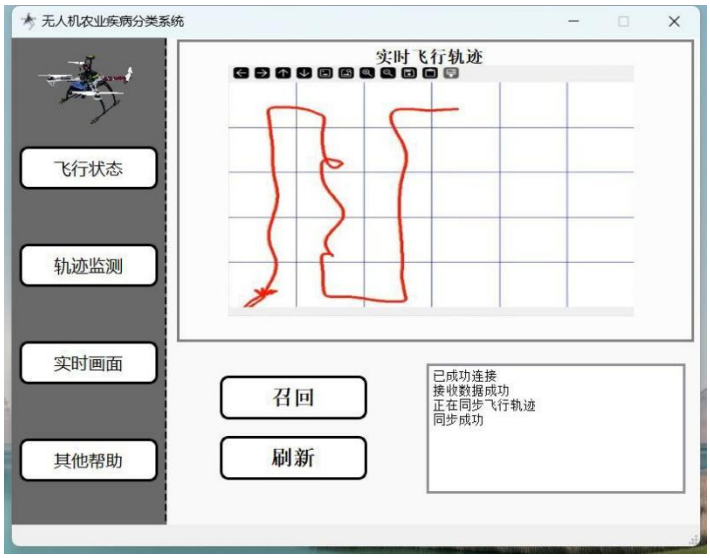


图 4-15 轨迹监测界面

在“轨迹监测”模块，用户可以通过该模块实时查看无人机的飞行路径测试分析，有助于监测无人机的飞行状态和执行任务的进度。记录无人机的历史飞行轨迹对于事后分析和评估飞行性能、检查飞行任务执行情况以及优化飞行路径都能起到很大帮助。与此同时，右下角的消息提示栏可以显示平台与无人机数

据传输等情况，同时向用户显示相应的错误消息或警告信息，为用户提供了及时的信息反馈和状态更新。

在这个模块中，当用户观察到飞行轨迹异常或在无人机飞行状态异常时，用户可以点击“召回”按钮，将无人机召回指定地点。

在“实时画面”模块，无人机的机载电脑将视频流传输至用户，交互系统接收后可以显示实时画面，同时用户也可以通过跳转的“非攻透传”应用软件来观看实时的无人机第一视角的高清工作画面。

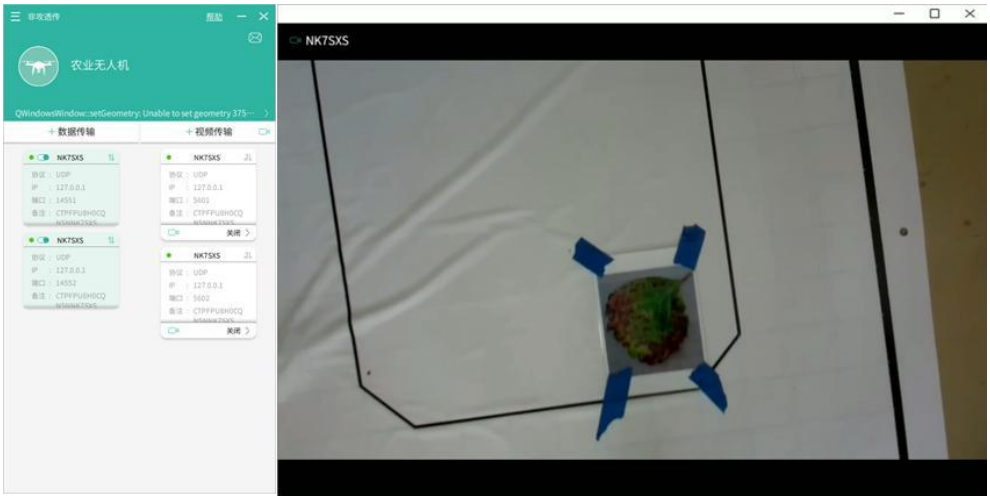


图 4-16 实时画面显示

C.移动端实时监测

为了满足用户对便携性和实时监测的需求，我们选择了将CUAV GS作为移动端的地面站应用程序。CUAV GS是由CUAV开发的一款功能强大的移动应用，使用与“非攻透传”相同的账户，在为用户提供了实时监测、飞行参数调整、航点设定等功能的同时统一了无人机设备的管理，实现了数据同步，使得用户可以随时随地轻松掌控无人机的飞行。CUAV GS的界面简洁直观，操作便捷，用户可以通过智能手机或平板电脑轻松进行各项操作。

图 4- 17 CUAV GS登录界面

输入4.5.1提到的“非攻透传”中的用户信息后，点击“登录”进入到设备界面，先前在PC端绑定的无人机设备“NK7SXS”已经在其中，无需再次进行绑定。将飞控系统连接的LTE LINK SE打开后，CUAV GS即可连接到对应设备。

图 4- 18 CUAV GS主界面

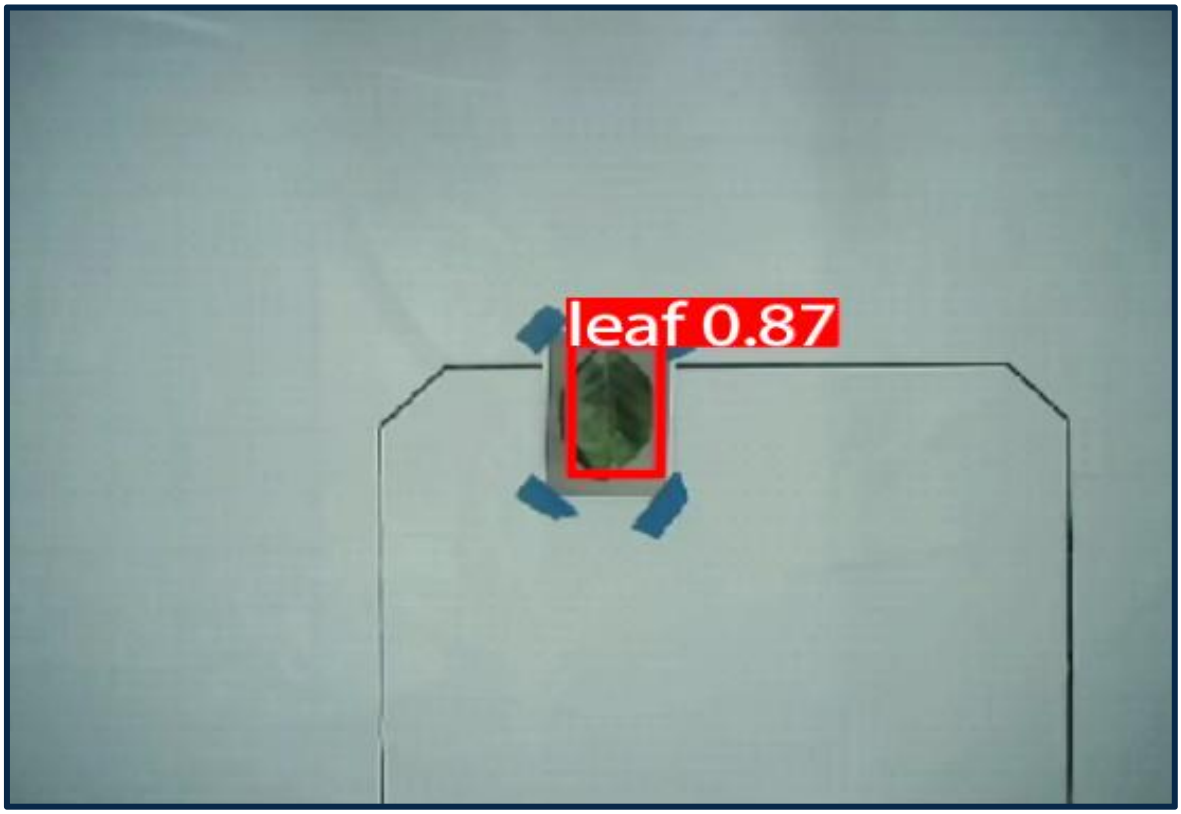
点击“进入飞行界面”按键后即可进入地面站监测画面。我们可以实时观察到无人机飞行的角度、速度、电池剩余电量等无人机状态。不仅如此，该界面

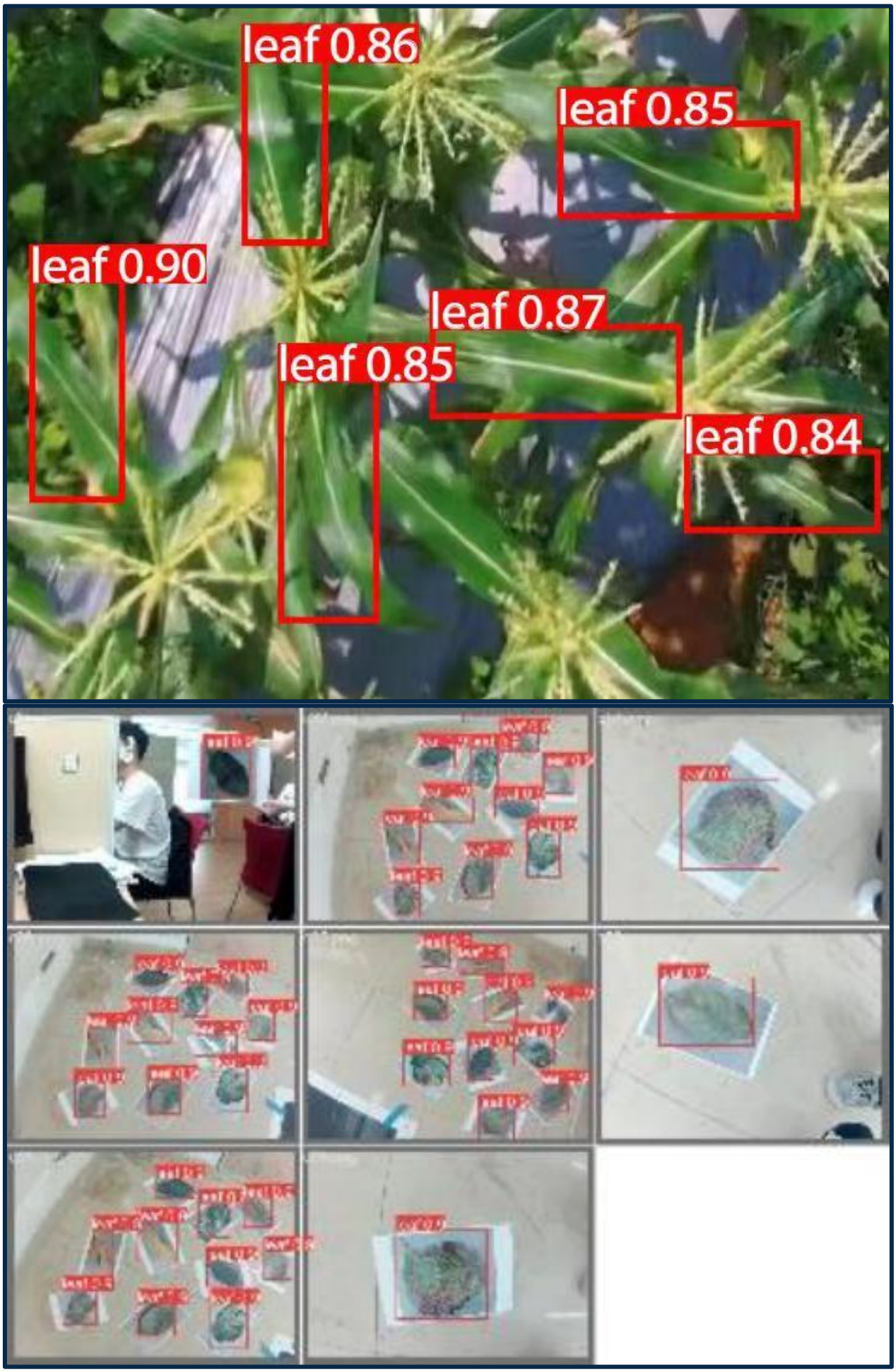
还支持飞行航线的切换以及拍摄等功能，为用户提供了全面的飞行监控与控制体验。

图 4- 19 移动端地面站实时监测

四、应用效果

4.1 yolov5检测效果

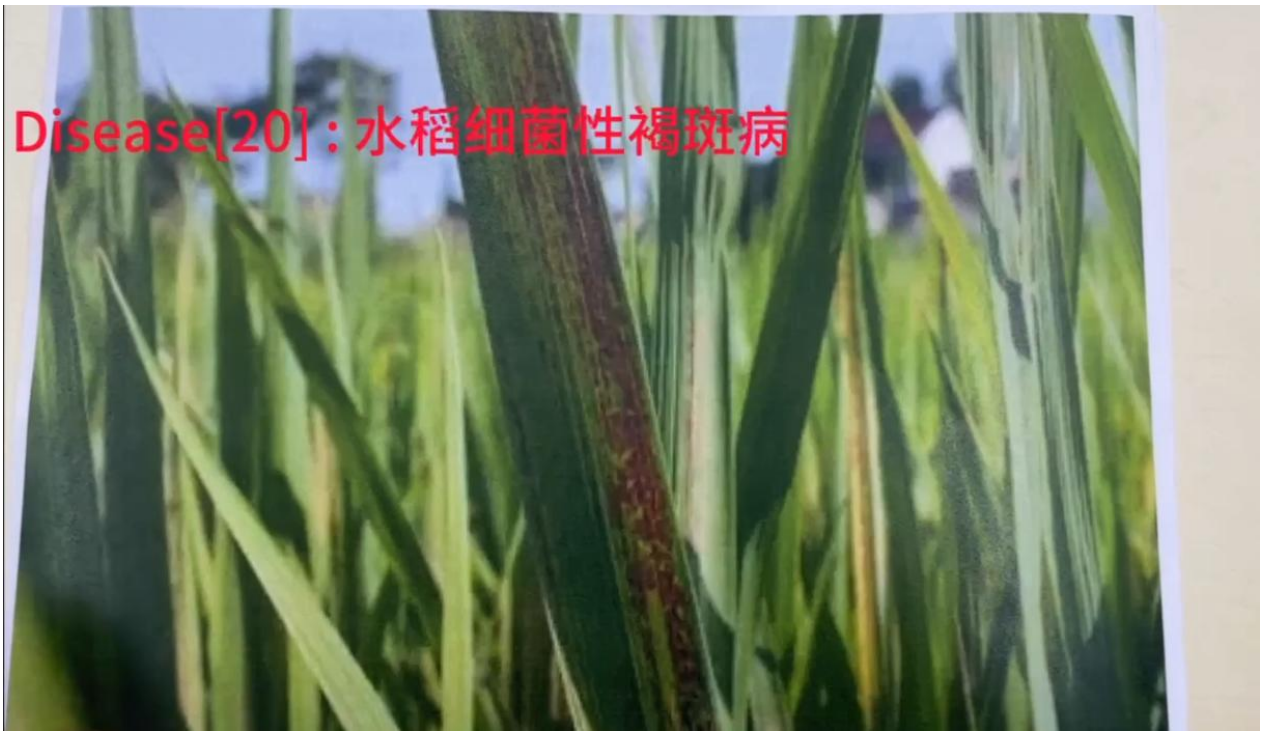



leaf 0.86
leaf 0.85
leaf 0.90
leaf 0.87
leaf 0.85
leaf 0.84

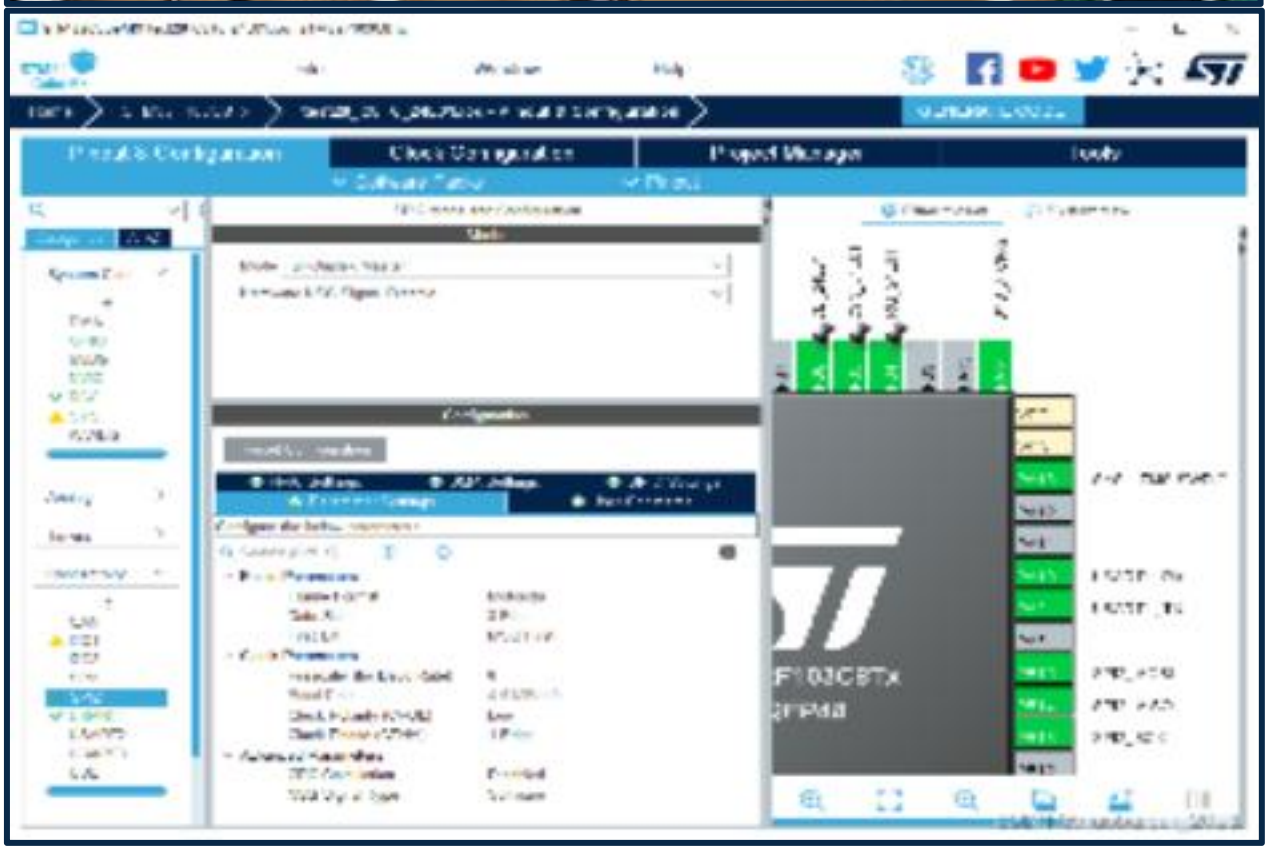
4.2 疾病分类效果

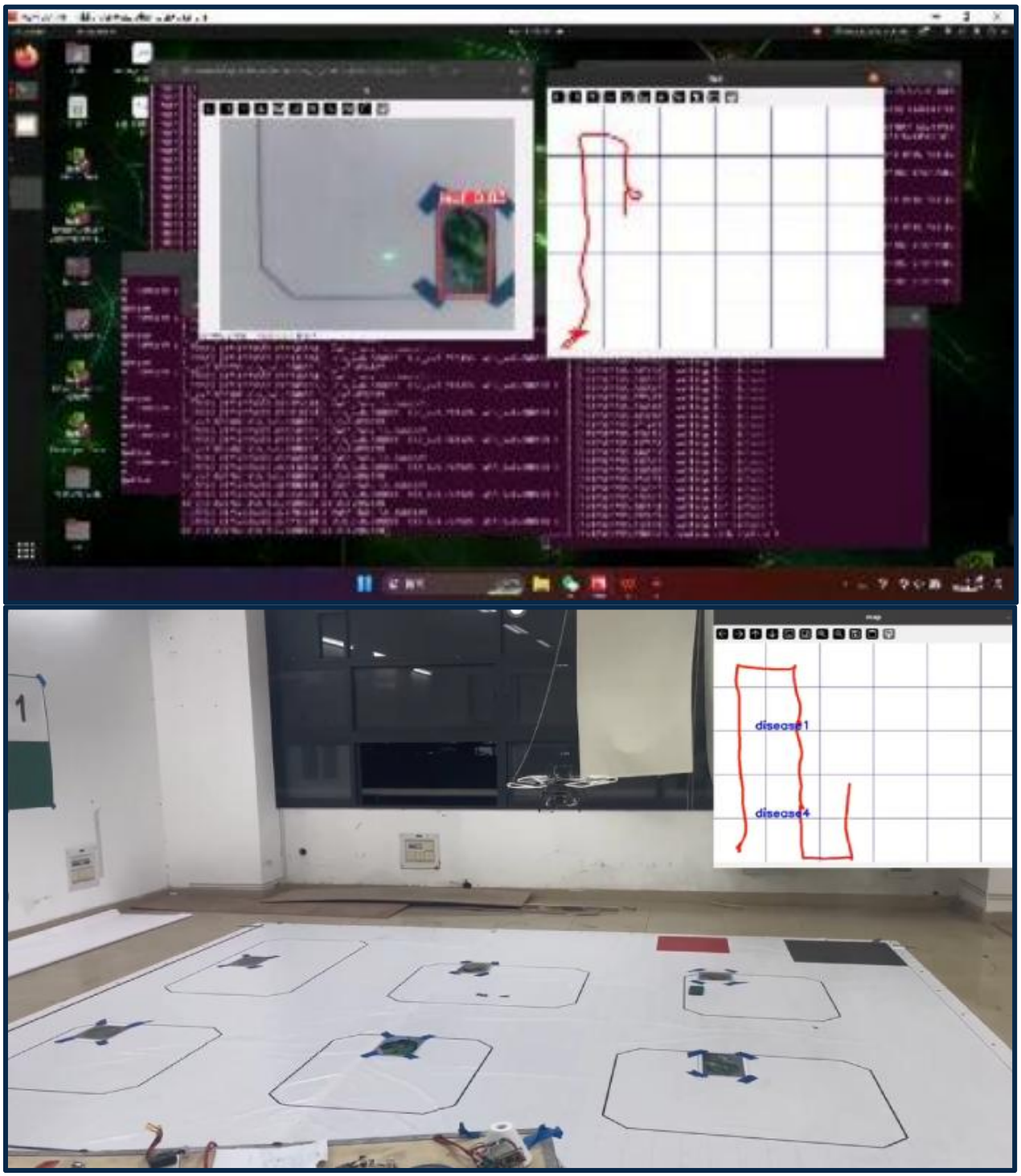

Disease[5]：番茄灰叶霉病


Disease[3]：水稻胡麻斑病


Disease[20]：水稻细菌性褐斑病

4.3 无人机调式




1
disease1
disease4

五、创新点与特色

5.1 基于改进Wlou-yolov5的目标检测

目标检测作为计算机视觉的核心问题，其检测性能依赖于损失函数的设计。边界框损失函数作为目标检测损失函数的重要组成部分，其良好的定义将为目标检测模型带来显著的性能提升。近年来的研究大多假设训练数据中的示例有较高的质量，致力于强化边界框损失的拟合能力。但我们注意到目标检测训练集中含有低质量示例，如果一味地强化边界框对低质量示例的回归，显然会危害模型检测性能的提升。Focal-EIoU v1 被提出以解决这个问题，但由于其聚焦机制是静态的，并未充分挖掘非单调聚焦机制的潜能。基于这个观点，我们 提出了动态非单调的聚焦机制，设计了 Wise-IoU (WIoU)。

动态非单调聚焦机制使用“离群度”替代 IoU 对锚框进行质量评估，并提供了明智的梯度增益分配策略。训练初期，WIOU v1不断降低低质量事例的占比，训练中后期，WIOU v3降低低质量实例对模型性能的增益。该策略在降低高质量锚框的竞争力的同时，也减小了低质量示例产生的有害梯度。这使得 WIoU 可以聚焦于普通质量的锚框，并提高检测器的整体性能。

我们的方法基于MS-COCO 数据集上对比了传统CIoU 、SIoU以及不同参数α和β下WIoU的平均准确度，应用于最先进的单级检测器 YOLOv7 时， 本方法AP-75 从 53.03% 提升到 54.50%。

TABLE I: Performance of each bounding box loss.

	AP_{75}^{val}	AP_{50}^{val}	AP^{val}
CIoU [11]	53.03	63.14	45.20
CIoU v2 ($\gamma = 0.5$)	53.47 (+0.44)	63.41 (+0.27)	45.12
CIoU v3 ($\alpha = 1.4, \delta = 5$)	53.25 (+0.22)	63.34 (+0.20)	44.76
CIoU v3 ($\alpha = 1.6, \delta = 4$)	53.68 (+0.65)	63.34 (+0.20)	45.10
CIoU v3 ($\alpha = 1.9, \delta = 3$)	53.04	62.92	44.91
SIoU [13]	53.15	63.46	45.21
SIoU v2 ($\gamma = 0.5$)	53.07	63.12	44.66
SIoU v3 ($\alpha = 1.4, \delta = 5$)	53.27 (+0.12)	64.13 (+0.67)	45.15
SIoU v3 ($\alpha = 1.6, \delta = 4$)	53.21	63.48	44.89
SIoU v3 ($\alpha = 1.9, \delta = 3$)	53.42 (+0.27)	63.28	45.03
EIoU [12]	53.55	63.17	45.39
Focal-EIoU [12]	52.88	63.37 (+0.20)	44.75
WIoU v1	52.82	63.15	44.87
WIoU v2 ($\gamma = 0.5$)	53.67 (+0.85)	64.15 (+1.00)	45.56 (+0.68)
WIoU v3 ($\alpha = 1.4, \delta = 5$)	53.75 (+1.07)	64.05 (+0.90)	45.15 (+0.28)
WIoU v3 ($\alpha = 1.6, \delta = 4$)	53.91 (+1.09)	64.16 (+1.01)	45.44 (+0.57)
WIoU v3 ($\alpha = 1.9, \delta = 3$)	**54.50 (+1.68)**	**64.20 (+1.05)**	**45.68 (+0.81)**

5.2 基于交叉注意力机制的多尺度分类器

我们提出了一种新的双分支视觉转换器来提取多尺度特征表示用于农业疾病分类。此外，我们开发了一种简单而有效的基于交叉注意的令牌融合方案，该方案在计算和内存上都是线性的，可以将不同尺度的特征组合在一起。

我们的方法比其他基于视觉转换器(ViT)的方法表现要更好，下面是基于CrossViT模型下的不同融合方式的在公开数据集ImageNet1K上的精确率、计算速度以及参数量的对比。

Fusion	Top-1 Acc. (%)	FLOPs (G)	Params (M)	Single Branch Acc. (%) L-Branch	S-Branch
None	80.2	5.3	23.7	80.2	0.1
All-Attention	80.0	7.6	27.7	79.9	0.5
Class Token	80.3	5.4	24.2	80.6	7.6
Pairwise	80.3	5.5	24.2	80.3	7.3
Cross-Attention	81.0	5.6	26.7	68.1	47.2

从表格中可以看到，我们的方法Cross-Attention精度最高，计算速度较快且模型参数量较小。

5.3 无人机精准喷洒系统

高精度定位：无人机搭载高精度定位系统，能够准确识别农作物和病虫害，从而实现精准喷洒，减少农药浪费和环境污染。

高效喷洒：无人机将搭载多种典型农药进行自主巡检，无人机通过深度学习得到的农业患病结果，将结果通过串口通信传入至stm32-c8t6，经过触发中断打开不同的农药喷雾口，实现精准对症喷洒；

自动巡检飞行：无人机可以通过农田巡检面积区域，实时规划无人机飞行航线，确保区域内均能准确巡检识别，并在判定为喷洒区域内进行快速喷洒。

5.4 多平台交互与实时监测

全面飞行状态监控与控制：我们的界面设计涵盖飞行状态、轨迹监测、实时画面等关键功能模块。在飞行状态界面中，用户可以即时查看无人机的各项关键参数，如飞行速度、高度、信号强度、电池状态和摄像头状态。这些信息以直观的方式展示，帮助用户全面掌握无人机的飞行状况。通过简单的按钮操作，比如点击“开始”按钮，用户可以向无人机发送指令，启动或结束任务，并且可以在界面上实时监控无人机的飞行过程。此外，我们的界面设计还包括“校准”和“停止”按钮，允许用户对无人机进行飞行角度和高度的精确调整和控制，从而确保飞行的安全性和效率。

智能化操作与反馈：我们注重实现智能化的操作体验和即时反馈机制。界面上设有消息提示栏，及时更新用户关键数据的传输状态和可能的错误信息。例如，在轨迹监测模块中，用户可以通过“召回”按钮快速响应无人机飞行轨迹异常或状态不佳的情况，保障任务的顺利执行和飞行安全。这种智能化的设计不仅简化了用户的操作流程，还增强了用户在复杂飞行环境中的响应能力和控制能力，为用户提供了更加安全和高效的操作体验。

无人机飞行画面与状态的跨平台实时监测：为了满足用户对便携性和实时监测的需求，我们利用了CUAV GS移动应用作为扩展，与“非攻透传”PC端

应用和QGC地面站相配合。CUAV GS应用界面简洁直观，支持智能手机和平板电脑，用户可以随时随地通过移动设备实时观察和控制无人机的飞行状态。在交互界面点击时，PC端上的“非攻透传”软件会弹出，利用它可以通过4G通信技术将绑定的无人机的飞行画面和飞行状态传输到QGC地面站。在QGC地面站上，用户可以实时查看并分析无人机的飞行数据，PC端的交互界面也会实时显示飞行的状态信息，同时在移动设备上的CUAV GS应用也能即时显示这些数据，确保用户在不同设备上都能获得一致和及时的信息。这种跨平台支持大大提升了用户的操作灵活性和便捷性，无论用户身处何地，他们都能轻松地管理和操作无人机，实现实时监控和数据获取的需求。

实时数据分析与优化：我们的系统不仅提供基本的监控和控制功能，还支持实时数据分析和飞行轨迹的记录。用户可以通过系统记录和分析无人机的飞行数据，实时评估飞行性能和优化飞行路径。这些分析结果对于提高农业疾病分类任务的效率和准确性至关重要，用户可以根据数据分析结果调整飞行策略和路径规划，提升整体工作效能。