Aimbot报告

汇报人：陈稷豪

CONTENTS

PART

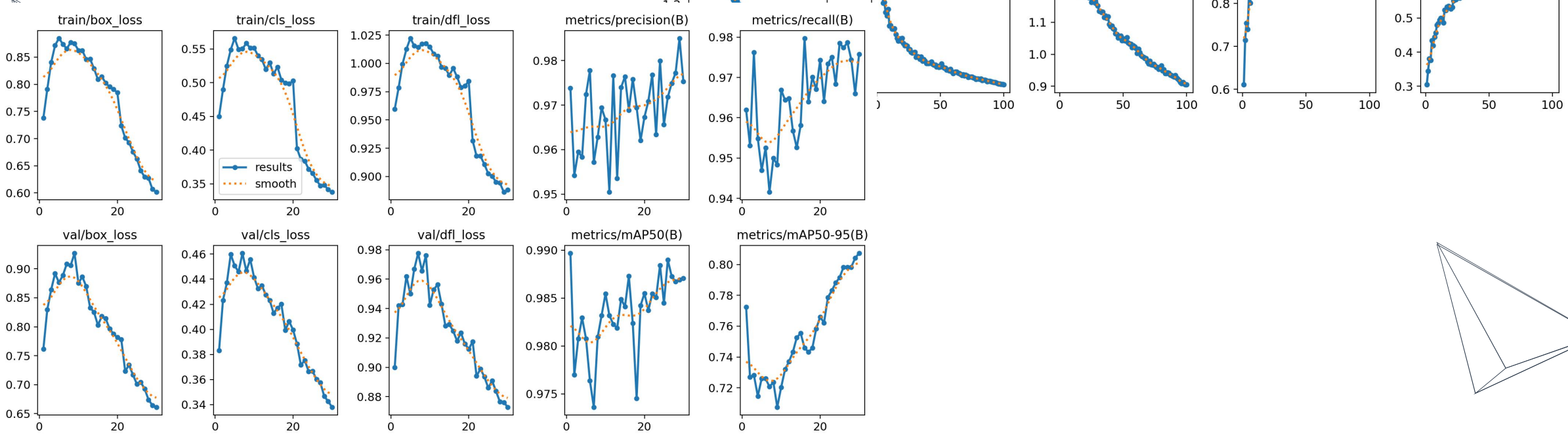
01

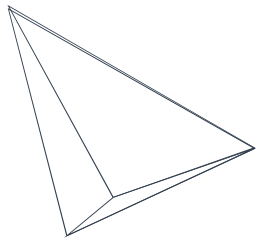
模型/算法

model/algorithm

训练

使用预训练模型yolov8n.pt。分别训练了100轮，40轮。





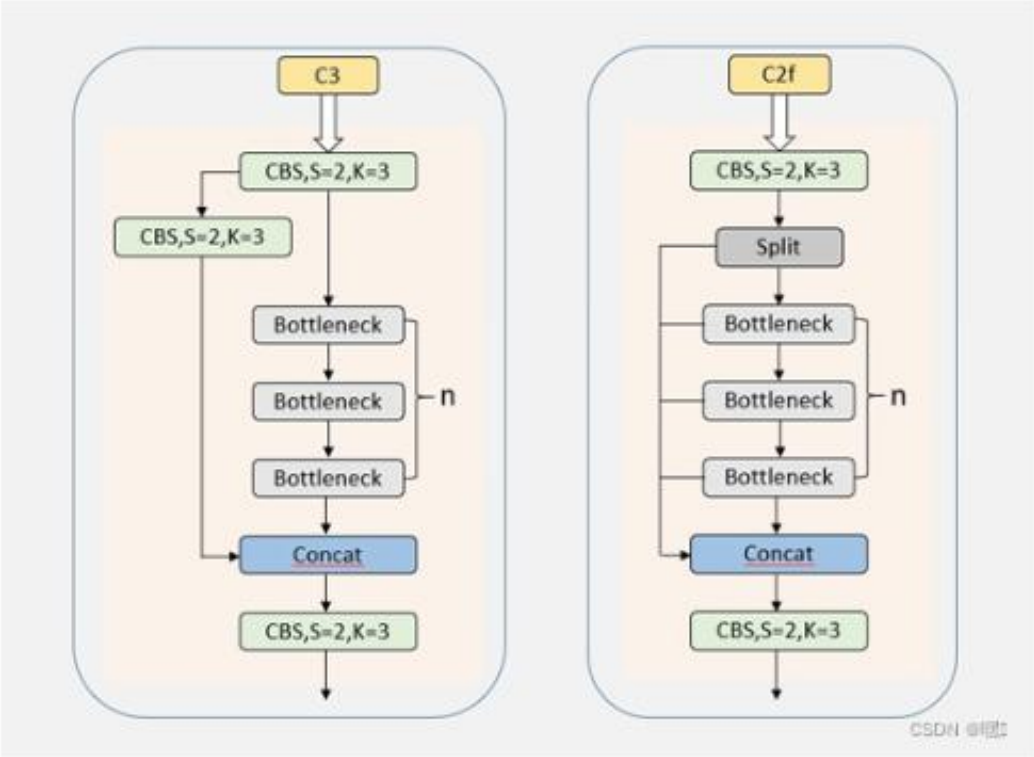
train23的模型情况

```
Validating runs\detect\train23\weights\best.pt...
Ultralytics YOLOv8.2.28 🚀 Python-3.10.14 torch-2.3.0+cu121 CUDA:0 (NVIDIA GeForce GTX 1650 Ti, 409
Model summary (fused): 168 layers, 3006428 parameters, 0 gradients, 8.1 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
                   all        160        372      0.975      0.976      0.987      0.807
                     0         88        110      0.983          1      0.995      0.882
                     1         82        104       0.96      0.917      0.974      0.686
                     2         70         84      0.995          1      0.995      0.917
                     3         62         74      0.964      0.986      0.984      0.744
Speed: 0.6ms preprocess, 6.8ms inference, 0.0ms loss, 1.9ms postprocess per image
Results saved to runs\detect\train23
```

模型参数较多，在运行一段时间后可能会出现卡顿现象。

剪枝

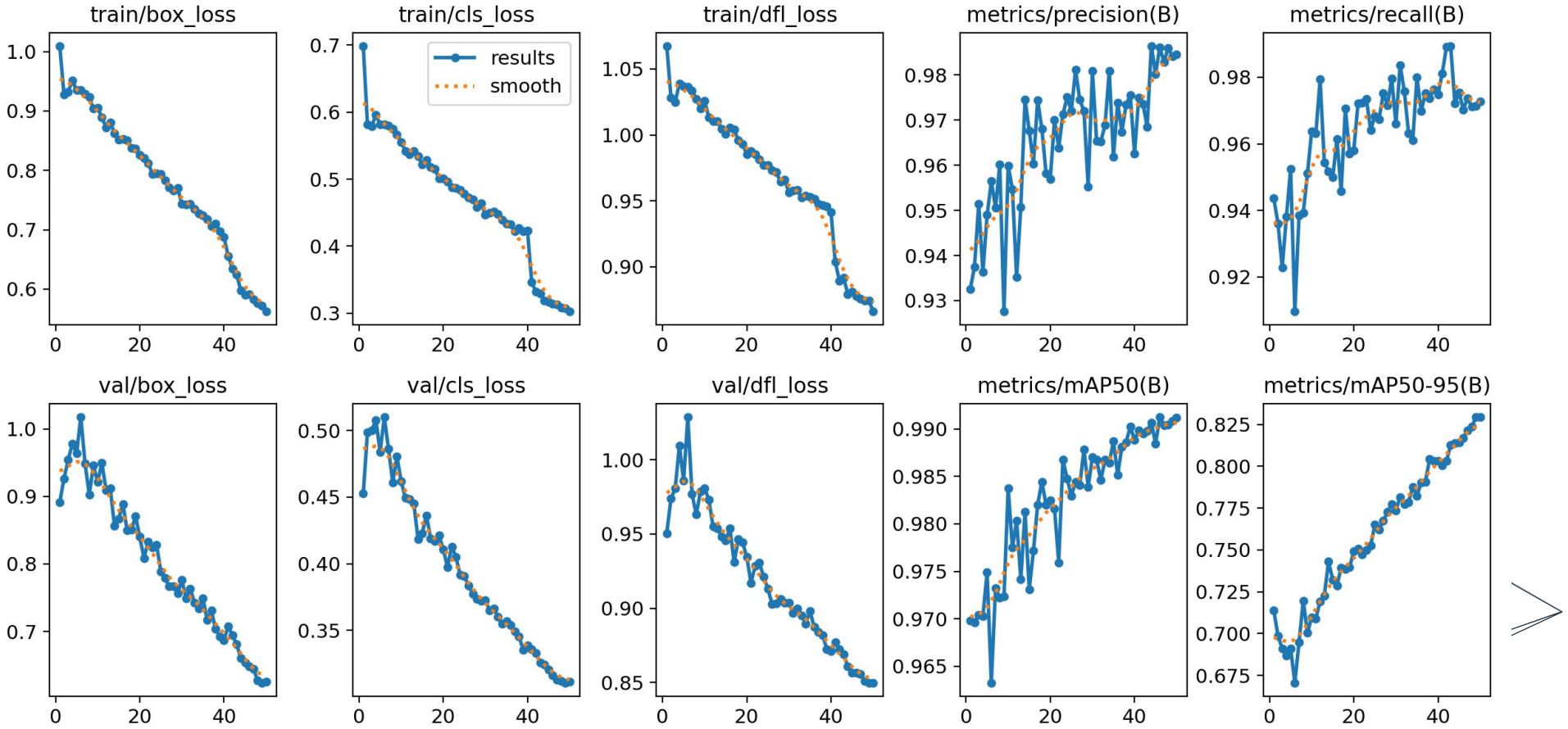
将C2f层替换为C2f_v2层后，使用torch-pruning进行剪枝。使用组归一化的 L2 范数来计算每个参数的重要性。用于确定哪些参数可以被移除，而不会显著影响模型性能



```
Validating runs\detect\step_0_finetune2\weights\best.pt...
Ultralytics YOLOv8.2.28 🚀 Python-3.10.14 torch-2.3.0+cu121 CUDA:0 (NVIDIA GeForce GTX 1650 Ti, 4096MiB)
Model summary (fused): 185 layers, 2582323 parameters, 0 gradients, 7.0 GFLOPs
                 Class     Images  Instances      Box(P          R      mAP50  mAP50-95): 100%|
| 5/5 [00:01<00:00,  2.67i
                   all        160        372      0.985      0.973      0.991      0.829
                     0         88        110          1      0.996      0.995      0.893
                     1         82        104      0.979      0.908       0.99      0.713
                     2         70         84      0.995          1      0.995      0.937
                     3         62         74      0.964      0.986      0.984      0.776
Speed: 0.5ms preprocess, 7.2ms inference, 0.0ms loss, 1.2ms postprocess per image
Ultralytics YOLOv8.2.28 🚀 Python-3.10.14 torch-2.3.0+cu121 CUDA:0 (NVIDIA GeForce GTX 1650 Ti, 4096MiB)
```

剪枝

训练50轮后的结果

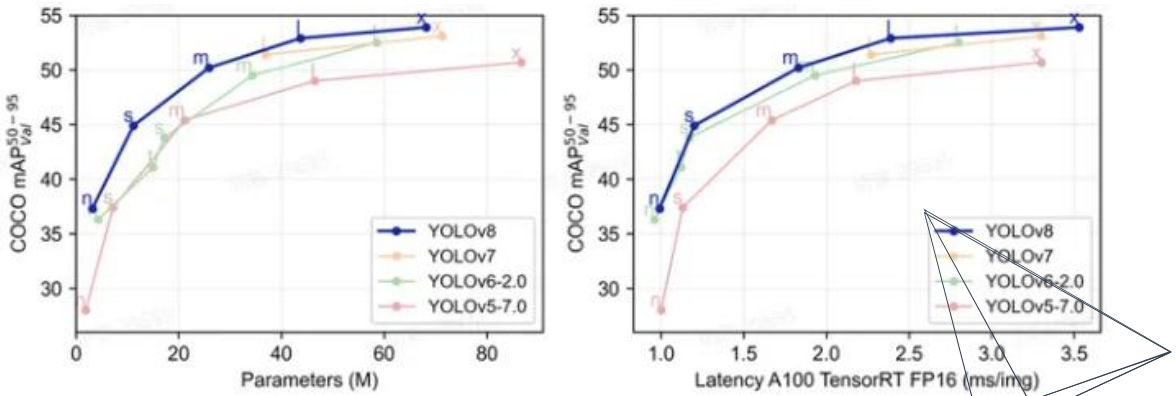


yolov5
yolov8

		YOLOV5	YOLOV8
核心思想及实现	Label Assign	采用grid+anchor组合策略来完成正负样本的分配	anchor即grid，采用TaskAlignedAssigner方法根据对齐指标分配正负样本
	Bbox Decode	需根据grid计算出中心点坐标然后根据anchor计算出宽高	根据grid即anchor得出grid中心点坐标，根据dist分布求得左上、右下角点对于grid中心点的距离偏移，进而转化为xywh坐标形式
网络结构	Backbone	C3	C2f
	Neck	C3	C2f
	Head	非解耦头，位置和置信度一起回归，输入为640*640推理时最终输出shape为 (batch，25200，85)	解耦头，位置和置信度单独回归，删除了object分支，输入为640*640推理时最终输出shape为 (batch，8400， 84)
损失函数	Loss	BCE + CIOU	BCE + CIOU + DFL
训练过程	epoch	300	500
	Compute anchor	auto	no need
推理过程	post process	nms需要obj_score	nms不需要obj_score

TABLE I: Performance metrics across all models.

Architecture	mAP (0.5)	mAP (0.5:0.95)	Precision	Recall
YOLOv8m	0.964	0.821	0.885	0.878
YOLOv8l	0.966	0.826	0.886	0.878
YOLOv8x	0.951	0.772	0.865	0.866
YOLOv5m	0.978	0.829	0.918	0.917
YOLOv5l	0.976	0.768	0.867	0.852
YOLOv5x	0.983	0.834	0.917	0.913



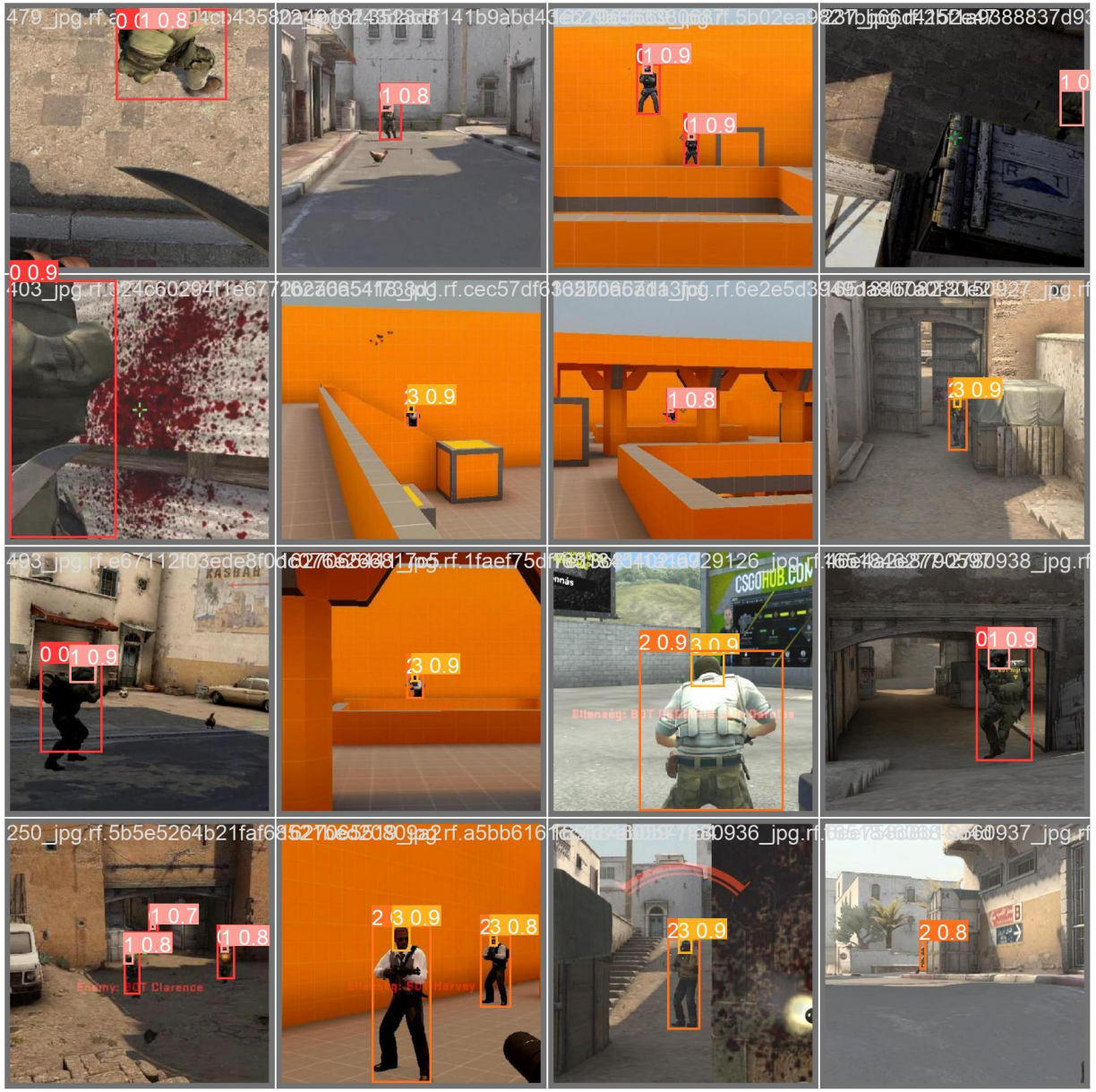
YOLOV5&V8参数量和性能比对

PART
02

数据集

dataset

数据集



https://universe.roboflow.com/miktory/cs2-fyd8y

一个拥有3306张图片的目标检测数据集，有四个标签包含CT头，CT身体，T头，T身体。

PART

03

效果展示

presentation

效果展示可视化

在训练完模型后，为测试模型在实际应用中的效果，先使用模型检测一些csgo图片，视频，并将结果可视化，来评估在实际应用中的效果。

可以看到模型检测的结果，以及该结果的置信度。

应用场景可视化

实现自动瞄准和自动开火。

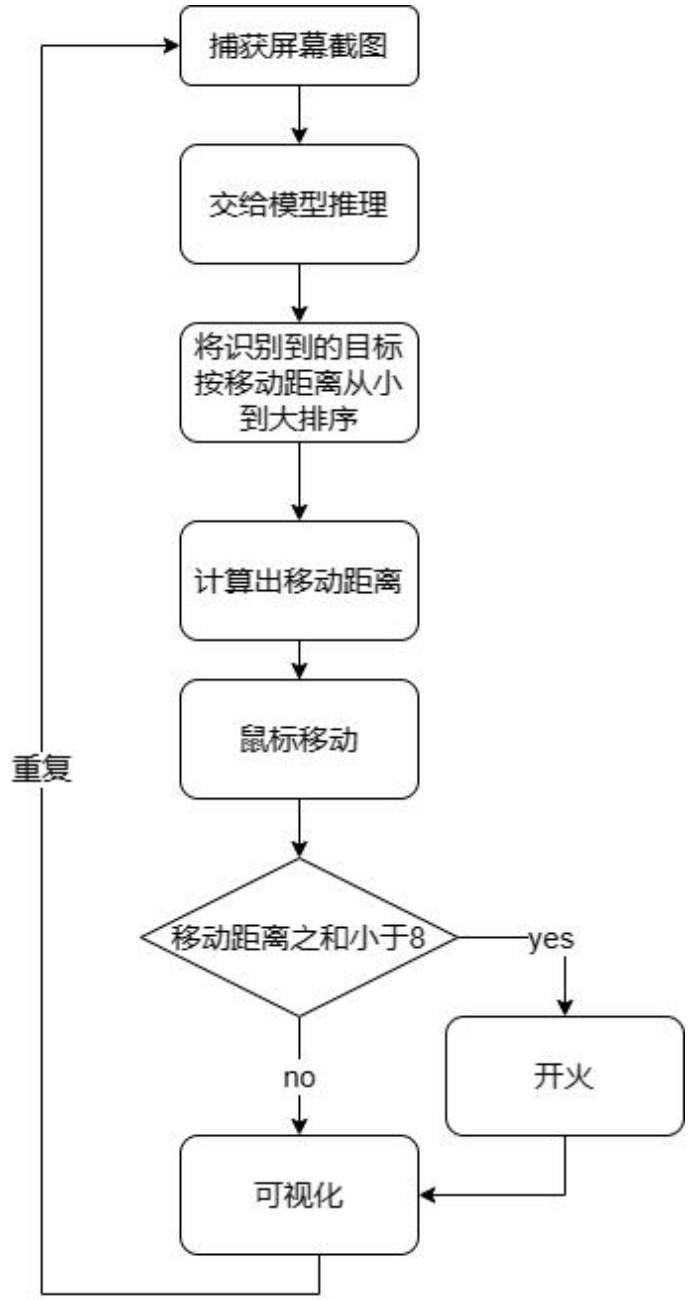
可以选择半自动，全自动和手动模式。

半自动：只有摁住右键时才会自瞄和自动开火。

全自动：按下鼠标侧键进入全自动，识别到敌人就开火。

手动：在半自动状态下按一次鼠标中键，变成全手动。

模型推理时间和大小等方面的可视化



在应用层面提高准确度

1.在之前的视频测试时发现，若截图整个屏幕，出现错误识别的概率会提高，不相关的东西更多。且参考ppt中aimmy_bot的做法，将FOV设置成屏幕中间一块，提高准确度，且符合应用场景。

2.虽然在数据集中有四个label，CT头，CT身体，T头，T身体。但因模型准确度和实战强度的考虑，只在检测到头部时进行自瞄。此外，要求置信度在0.5以上时才自瞄。

可调整参数：窗口大小，最大移动距离和。

鼠标控制

在鼠标点击时，使用pydirectinput直接输入。

监听鼠标动作时，使用pynput.mouse的Listener。

在控制鼠标移动时，发现pydirectinput和pyautogui有时没反应，或是感觉移动速度较慢。一次使用了更为底层的api，即ctypes中的sendinput。

PART

04

总结

presentation

未来展望

对于鼠标移动光滑度可以进一步调整。

可以进一步对模型优化

YOU LOGO

演示完毕感谢观看