基于深度学习的农作物及农业生物识别技术

针对几何形变和光照变化两大核心挑战，分别构建专注于几何特征增强和色彩特征增强的专家模型；基于加权框融合算法，使用 python 语言，实现 YOLOv8s 和 YOLOv8m 两个专家模型预测结果的智能互补，确保在农作物密集和光照突变等极端场景下，系统整体检测仍保持高准确率；3.方案验证：通过系统的模型评估与 A/B 测试，验证双专家模型融合方案在农学识别技术中的性能优势与有效性。

融合后的 mAP@50 提升至 0.87425，较最优单模型提升 2.1%，有效增加了模型的鲁棒性，实现 1+1>2 的效果。

一、关键代码

1 单模型性能评估函数（用于自动权重）

```
Def get_model_performance(model,data_yaml_path,device):
model_name=os.path.basename(model.ckpt_path)
print(f"正在评估模型{model_name}的基准性能...")
metrics=model.val(data=data_yaml_path,split='val',device=device,plots=False,verbose=False,batch=8,workers=0)
map50=metrics.box.map
print(f"模型{model_name}mAP@.50:{map50:.4f}")
returnmap50
```

2 通用融合函数封装（支持 WBF、Soft-NMS、NMW 等）

```
Def ensemble_predictions(boxes_list,scores_list,labels_list,method='wbf',iou_thr=0.6,skip_box_thr=0.01,weights=None,sigma=0.1):
non_empty_indices=[ifori,boxesinenumerate(boxes_list)iflen(boxes)>0]

ifnotnon_empty_indices:
```

returnnp.array([]),np.array([]),np.array([])

boxes_list=[boxes_list[i]foriinnon_empty_indices]

scores_list=[scores_list[i]foriinnon_empty_indices]

labels_list=[labels_list[i]foriinnon_empty_indices]

ifweights:

weights=[weights[i]foriinnon_empty_indices]

ifmethod=='wbf':

returnweighted_boxes_fusion(boxes_list,scores_list,labels_list,weights=weights,iou_thr=iou_thr,

skip_box_thr=skip_box_thr)

elifmethod=='nmw':

returnnon_maximum_weighted(boxes_list,scores_list,labels_list,weights=weights,iou_thr=iou_thr,

skip_box_thr=skip_box_thr)

elifmethod=='soft_nms':

returnsoft_nms(boxes_list,scores_list,labels_list,iou_thr=iou_thr,sigma=sigma,thresh=skip_box_thr)

elifmethod=='nms':

returnnms(boxes_list,scores_list,labels_list,weights=weights,iou_thr=iou_thr)

else:

raiseValueError(f"未知的融合方法:{method}")

3 主流程中的预测与融合处理

formodelinmodels:

results=model(img,verbose=False)

pred_boxes=results[0].boxes.xyxy.cpu()

pred_scores=results[0].boxes.conf.cpu()

```
pred_labels=results[0].boxes.cls.cpu()
pred_boxes_normalized=pred_boxes/torch.tensor([img_w,img_h,img_w,img_h],dtype=torch.float32)
boxes_list.append(pred_boxes_normalized.tolist())
scores_list.append(pred_scores.tolist())
labels_list.append(pred_labels.tolist())
```

4 融合结果处理与 GT 读取

```
iflen(fused_boxes)>0:
preds=[{
'boxes':torch.tensor(fused_boxes,dtype=torch.float32).to(device)*torch.tensor([img_w,img_h,img_w,img_h]).to(device),
'scores':torch.tensor(fused_scores,dtype=torch.float32).to(device),
'labels':torch.tensor(fused_labels,dtype=torch.long).to(device),
}]
else:
preds=[{
'boxes':torch.empty(0,4,dtype=torch.float32).to(device),
'scores':torch.empty(0,dtype=torch.float32).to(device),
'labels':torch.empty(0,dtype=torch.long).to(device),
}]
```

5mAP 评估器更新与打印

```
map_metric.update(preds,target)
final_metrics=map_metric.compute()
print(f"---策略'{strategy_name}'结果---")
print(f"mAP@.50:{final_metrics['map_50']:.4f}")
print("-"*(len(strategy_name)+14))
```

6 目标检测图片输出

```
    Def
visualize_test_results(model_path,data_dir,classes_path,output_dir='test_
results',num_images=5):
test_images_dir=os.path.join(data_dir,'test','images')
ifnotos.path.exists(test_images_dir):
test_images_dir=os.path.join(data_dir,'test')
    ifnotos.path.exists(test_images_dir):
raiseFileNotFoundError(f"测试集图片目录不存在:{test_images_dir}")

test_images=[os.path.join(test_images_dir,f)forfinos.listdir(test_images_
dir)
iff.lower().endswith(('.jpg','.jpeg','.png','.bmp'))]

ifnottest_images:
raiseValueError(f"测试集目录中没有找到图片:{test_images_dir}")

iflen(test_images)>num_images:
np.random.seed(42)
test_images=np.random.choice(test_images,num_images,replace=False).
tolist()

forimage_pathintest_images:
print(f"\n 处理图片:{image_path}")
#调用 predict_image，每张图片单独保存
predict_image(model_path,image_path,classes_path,output_dir)
print(f"\n 所有测试图片结果已保存至:{output_dir}")
```

7 目标检测结果输出

```
result=results[0]

boxes=result.boxes

class_ids=boxes.cls.cpu().numpy().astype(int)

confidences=boxes.conf.cpu().numpy()

bboxes=boxes.xyxy.cpu().numpy().astype(int)


class_counts=Counter()

forclass_idinclass_ids:

class_name=classes[class_id]ifclasseselsef"Class{class_id}"

class_counts[class_name]+=1
```