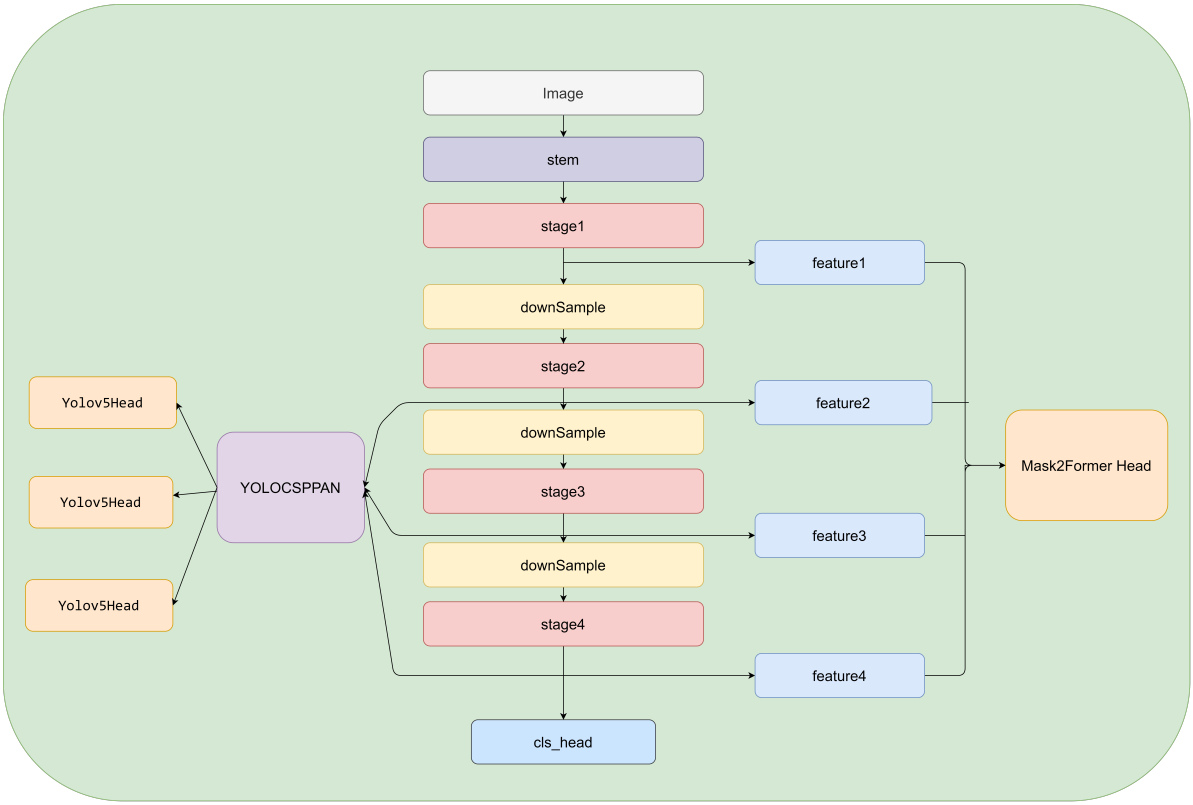
第二届广州·琶洲算法大赛-智能交通CV模型赛题第4名方案

1.模型结构图、思路步骤详述、代码组织结构介绍

1.1模型结构图



1.2 思路步骤详述

1. 难点在于分割和小目标检测
2. 在backbone选型的问题上我们有一点小分歧，Convnext-Large 和Swin-L
3. 检测头和分割头的选型 （DINO，RT-DETR，YOLOv5，yolov7） （upernet，maskformer，mask2former）
4. 多尺度训练
5. 检测的sahi切图方案。
6. 检测的copypaste方案。
7. TTA 后处理 （检测的多尺度推理，分割的多次结果ensemble mulit scale推理）
8. 多任务head梯度对整体网络的影响
9. VPT 架构微调

1.3 代码组织结构介绍

```
├─configs 模型配置 train test
├─data     数据集
├─detectron2 训练trainer相关
├─engine
├─evaluation  验证代码
```

```
├─fastreid
├─layers
├─logs
├─modeling   多任务架构 模型 head
├─PaddleSeg  分割头 maskformer mask2former
├─PaddleYOLO 检测头 yolov5head
├─scripts    启动训练脚本
├─solver
├─tools      训练流程，测试流程
└─utils      辅助函数
```

2.数据增强/清洗策略

2.1数据合并

```
import os
import json
d = "datasets/track1_train_data_all/dec"
with open(os.path.join(d,"train.json"),"r") as f:
    train = json.load(f)
with open(os.path.join(d,"val.json"),"r") as f:
    val = json.load(f)
train["images"] += val["images"]
train["annotations"] += val["annotations"]
with open(os.path.join(d,"train.json"),"w") as f:
    json.dump(train,f)
```

sh preprocess.sh

```
echo "Begin extract"

mkdir -p datasets

tar xf /home/aistudio/data/data203253/track1_test_data.tar -C  datasets
tar xf /home/aistudio/data/data203253/track1_train_data.tar -C  datasets

echo "End extract"


echo "Begin copy"

SRC=datasets/track1_train_data
DEST=datasets/track1_train_data_all


# make new dirs

# cls
mkdir -vp $DEST/cls/train
mkdir -vp $DEST/cls/val

# dec
mkdir -vp $DEST/dec/train
mkdir -vp $DEST/dec/val

# seg images
```

```
mkdir -vp $DEST/seg/images/train
mkdir -vp $DEST/seg/images/val

# seg labels
mkdir -vp $DEST/seg/label/train
mkdir -vp $DEST/seg/label/val

# copy cls val to train dir
echo "Copy cls"

cp $SRC/cls/train/* $DEST/cls/train
cp $SRC/cls/val/* $DEST/cls/val
cp $SRC/cls/val/* $DEST/cls/train
cp $SRC/cls/val.txt $DEST/cls/val.txt
cat $SRC/cls/train.txt $SRC/cls/val.txt > $DEST/cls/train.txt

# copy seg val to train dir
echo "Copy seg"

cp $SRC/seg/images/train/* $DEST/seg/images/train/
cp $SRC/seg/images/val/* $DEST/seg/images/val/
cp $SRC/seg/images/val/* $DEST/seg/images/train/

cp $SRC/seg/label/train/* $DEST/seg/label/train/
cp $SRC/seg/label/val/* $DEST/seg/label/val/
cp $SRC/seg/label/val/* $DEST/seg/label/train/


# copy dec val to train dir
echo "Copy dec"

cp $SRC/dec/train/* $DEST/dec/train
cp $SRC/dec/val/* $DEST/dec/val
cp $SRC/dec/val/* $DEST/dec/train
cp $SRC/dec/train.json $DEST/dec/
cp $SRC/dec/val.json $DEST/dec/

# merge json
python process.py

echo "End copy"
```

2.2 数据增强

赛程前中期的aug策略

```
transforms=[
    L(ResizeStepScaling)(min_scale_factor=0.5, max_scale_factor=2.0,
scale_step_size=0.25),
    L(RandomPaddingCrop)( crop_size=(1024, 512)),
    L(RandomHorizontalFlip)(),
    L(RandomDistort)(brightness_range=0.2, contrast_range=0.2,
saturation_range=0.2),
    L(GenerateInstanceTargets)(num_classes=seg_num_classes),
    L(GenerateMaskFormerTrainTargets)(),
```

```
    L(Normalize)()],
mode='train'),

transforms=L(build_transforms_lazy)(
                    is_train=True,
                    size_train=[448, 448],
                    do_rea=True,
                    rea_prob=0.5,
                    do_flip=True,
                    do_autoaug=True,
                    autoaug_prob=0.5,
                    mean=[0.485 * 255, 0.456 * 255, 0.406 * 255],
                    std=[0.229 * 255, 0.224 * 255, 0.225 * 255],
                ),
transforms=[
    dict(Decode=dict(),),
    dict(AugmentHSV=dict(fraction=0.50,is_bgr=True,hgain=0.015, sgain=0.7,
vgain=0.4)),
    dict(RandomCrop=dict(
        aspect_ratio=[0.5, 2.0],
        thresholds=[.0, 0.1, 0.3, 0.5, 0.7, 0.9],
        scaling=[0.3, 0.6],
        num_attempts=50,
        allow_no_crop=True,
        cover_all_box=False,
        is_mask_crop=False
    )),
    dict(Resize=dict(target_size=[1024, 1024], keep_ratio=True,interp=1)),
    dict(Pad=dict(size=[1024,1024],
    fill_value=[114., 114., 114.])),
    dict(RandomFlip=dict(prob=0.5),),
    dict(NormalizeImage=dict(
        is_scale=True,
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225])
    ),
    dict(NormalizeBox=dict()),
    dict(BboxXYXY2XYWH=dict()),
    dict(Permute=dict()),
],
```

赛程后期

```
transforms=[
                        L(ResizeStepScaling)(min_scale_factor=0.5,
max_scale_factor=2.0, scale_step_size=0),
                        L(RandomPaddingCrop)(crop_size=[1024, 512]),
                        L(RandomHorizontalFlip)(),
                        L(One_of_aug)(method = [
                                    A.GaussNoise(var_limit=(10.0, 30.0), p=0.5),
                                    A.GaussianBlur(blur_limit=[3,5], sigma_limit=
[0,1],p = 0.5),
                                    ],p = 0.3 , only_img = True
                         ),
                        L(One_of_aug)(method = [
                                    A.Rotate (limit=15, p=0.4),
                                    A.ShiftScaleRotate(rotate_limit=15, p=0.4),
```

```
                                    A.GridDistortion(num_steps = 10,p = 0.2),
                                    ],p = 0.3,only_img = False
                        ),
                        L(RandomSelectAug)(
                                transforms1 = L(RandomDistort)
(brightness_range=0.25,
                                                brightness_prob=1,
                                                contrast_range=0.25,
                                                contrast_prob=1,
                                                saturation_range=0.25,
                                                saturation_prob=1,
                                                hue_range=63,
                                                hue_prob=1),
                                transforms2 = L(RandomWeather)(),
                                p = 0.95
                        ),
                        L(GenerateInstanceTargets)(num_classes=seg_num_classes),
                        L(Normalize)(mean=[0.485, 0.456, 0.406],std=[0.229,
0.224, 0.225])],
                    mode='train'),

transforms=L(build_transforms_lazy)(
                    is_train=True,
                    size_train=[448, 448],
                    do_rea=True,
                    rea_prob=0.5,
                    do_flip=True,
                    do_autoaug=True,
                    autoaug_prob=0.5,
                    mean=[0.485 * 255, 0.456 * 255, 0.406 * 255],
                    std=[0.229 * 255, 0.224 * 255, 0.225 * 255],
                ),
transforms=[
    dict(Decode=dict(),),
    dict(AugmentHSV=dict(fraction=0.50,is_bgr=True,hgain=0.015, sgain=0.7,
vgain=0.4)),
    dict(RandomCrop=dict(
        aspect_ratio=[0.5, 2.0],
        thresholds=[.0, 0.1, 0.3, 0.5, 0.7, 0.9],
        scaling=[0.3, 0.6],
        num_attempts=50,
        allow_no_crop=True,
        cover_all_box=False,
        is_mask_crop=False
    )),
    dict(Resize=dict(target_size=[1024, 1024], keep_ratio=True,interp=1)),
    dict(Pad=dict(size=[1024,1024],
    fill_value=[114., 114., 114.])),
    dict(RandomFlip=dict(prob=0.5),),
    dict(NormalizeImage=dict(
        is_scale=True,
        mean=[0.485, 0.456, 0.406],
        std=[0.229, 0.224, 0.225])
    ),
    dict(NormalizeBox=dict()),
    dict(BboxXYXY2XYWH=dict()),
    dict(Permute=dict()),
],
```

3.调参优化策略

整体考虑到显存数据集规模和模型收敛的节点 50000 iter 附近

训练设置 58200 max_iter

由于是余弦周期 考虑在50000 节点 衰减学习率solver_steps=[50000],

```
optimizer = L(build_lr_optimizer_lazy)(
    optimizer_type='AdamW',
    base_lr=1e-4,
    weight_decay=1e-4,
    grad_clip_enabled=True,
    grad_clip_norm=0.1,
    apply_decay_param_fun=None,
    lr_multiplier=L(build_lr_scheduler_lazy)(
        max_iters=60000,
        warmup_iters=200,
        solver_steps=[50000],
        solver_gamma=0.1,
        base_lr=1e-4,
        sched='CosineAnnealingLR',
    ),
)

train.amp.enabled = False

# data settings
sample_num = 7000
epochs=200
dataloader.train.task_loaders.segmentation.total_batch_size = 3* 8
dataloader.train.task_loaders.fgvc.total_batch_size = 4*8
dataloader.train.task_loaders.trafficsign.total_batch_size =  3*8

iters_per_epoch = sample_num //
dataloader.train.task_loaders.segmentation.total_batch_size ## 7000//24=291

max_iters = iters_per_epoch * epochs  ##58200

# optimizer
optimizer.lr_multiplier.max_iters = max_iters
optimizer.base_lr = optimizer.lr_multiplier.learning_rate = 1e-4
optimizer.lr_multiplier.solver_steps = [int(max_iters * 0.8)]
```

调参策略:

- swin-tiny
- upernet
- maskformer
- RT-DETR
- yolov5
- swin-transformer
- slice det image

- copypaste

调优过程：

我们从swin-tiny开始 再到convnext-L的upernet，dino（减少transformer层的）检测的效果一直不佳。

优先考虑单任务的调参，从Segmentation开始，从PaddleSeg调试maskformer，mask2former其实在同阶段已经调通，我们为了节省调优的时间，mask2former需要更长的训练周期 2Days more，而且maskformer的精度是接近当时榜上的最好的成绩。检测部分仍然很差，开始调优ppyoloe-head，ppyolo的head train需要额外的参，过于复杂，所以调试了RT-DETR，这个过程中尝试了slice切图，2048的大尺度推理等策略，经过多轮的调参，我们把检测的head换回yolov5 anchor base的head，继续尝试多尺度的ms tta。在6月底，重新修改了baseline。使用convnext-L+maskformer-fc head - yolov5 head,作为新基线。继续尝试1024 切图，mosaic 等transform。发现反而掉点，接下来对检测的目标进行crop，随机贴图全类别做会导致seg的掉点，我们平衡了数据集，将<400 个目标的class 进行贴图训练，指标能到94.8 接下来换backbone，我们将backbone 换成swin-transformer-L 那么当前的结构为 swin-L+maskformer +fc head yolov5。seg 直接涨到 0.69 但是 配上RT-DETR head 检测不佳。最终我们换回new base 7.1号的方案，把maskformer换成mask2former。

提交者	score	seg	cls	det	time	tricks
VTAMD7	0.8648	0.68878	0.95996	0.94554	2023-08-04 19:55	more ms scale
VTAMD7	**0.8653**	**0.6903**	**0.95996**	**0.94554**	**2023-08-04 17:43**	best seg 1.0 1.2 1.6
VTAMD7	0.8636	0.68867	0.95821	0.94383	2023-08-02 20:17	seg 1.0 1.5 2.0 scale
VTAMD7	0.8649	0.69065	0.95821	0.94584	2023-08-02 11:51	mask2former
VTAMD7	0.8621	0.68866	0.95821	0.93933	2023-08-02 11:28	mask2former
VTAMD7	0.8642	0.68866	0.95821	0.94584	2023-07-30 16:10	mask2former
VTAMD7	0.8636	0.6867	0.95821	0.94584	2023-07-30 14:25	mask2former
VTAMD7	0.8621	0.68225	0.95821	0.94584	2023-07-30 12:00	convnext-large
VTAMD7	0.7601	0.69345	0.95378	0.6331	2023-07-28 18:17	swin-large 4096
VTAMD7	0.861	0.69345	0.9539	0.93564	2023-07-28 17:57	swin-large 2048
VTAMD7	0.8588	0.67372	0.95875	0.94398	2023-07-27 15:39	swin-large rt-detr fineune 2048
VTAMD7	0.8618	0.68271	0.95875	0.94397	2023-07-27 14:11	swin-large rt-detr fineune 2048
VTAMD7	0.8464	0.69227	0.95316	0.89382	2023-07-26 19:33	swin-large rt-detr

提交者	score	seg	cls	det	time	tricks
VTAMD7	0.8589	0.69227	0.95303	0.9315	2023-07-26 18:08	swin-large rt-detr
VTAMD7	0.8603	0.69227	0.95303	0.93557	2023-07-26 17:52	swin-large rt-detr
VTAMD7	0.8474	0.69048	0.95353	0.89819	2023-07-25 23:48	swin-large rt-detr
VTAMD7	0.8564	0.69048	0.95353	0.92514	2023-07-25 22:26	swin-large rt-detr
VTAMD7	0.8628	0.69048	0.95353	0.94426	2023-07-24 10:25	swin-large rt-detr
VTAMD7	0.8607	0.68062	0.95224	0.94929	2023-07-21 09:36	swin-large rt-detr
VTAMD7	0.8601	0.68062	0.95224	0.94754	2023-07-21 00:03	copy paste <400 class ms
VTAMD7	0.8589	0.67448	0.95361	0.94874	2023-07-19 16:58	copy paste <400 class
VTAMD7	0.8569	0.67448	0.95361	0.9427	2023-07-19 16:01	copy paste seg drop 1%
VTAMD7	0.8587	0.67583	0.95088	0.94952	2023-07-16 11:09	yolov5 ms inference
VTAMD7	0.8623	0.68596	0.95137	0.9495	2023-07-11 23:04	2rd aug 2048 infer
VTAMD7	0.5498	0.68135	0.95423	0.01389	2023-07-11 00:25	共享fpn
VTAMD7	0.8601	0.68333	0.95548	0.94141	2023-07-08 21:55	seg maskformer + yolov5

提交者	score	seg	cls	det	time	tricks
VTAMD7	0.8616	0.68333	0.95473	0.94667	2023-07-08 11:39	seg maskformer + yolov5
VTAMD7	0.8593	0.68604	0.95013	0.94164	2023-07-06 23:43	seg maskformer + yolov5
VTAMD7	0.8498	0.67013	0.94155	0.93767	2023-07-06 19:49	seg maskformer + yolov5
VTAMD7	0.8597	0.68429	0.94366	0.951	2023-07-06 00:31	1536+2048 ms infer
VTAMD7	0.8578	0.68429	0.94366	0.94558	2023-07-05 20:39	2048 infer
VTAMD7	0.806	0.63023	0.90611	0.88176	2023-07-04 23:07	large size ms 31499 no convergence
VTAMD7	0.8607	0.6853	0.95075	0.94619	2023-07-04 22:57	mosaic no use
VTAMD7	0.5641	0.68464	0.95324	0.05435	2023-07-02 23:19	slice 1024 bad
VTAMD7	0.8635	0.68881	0.95436	0.94735	2023-07-01 17:40	seg maskformer + yolov5 seg aug +random 2rd
VTAMD7	0.8608	0.68258	0.95361	0.94614	2023-07-01 03:52	seg maskformer + yolov5 baseline
VTAMD7	0.6826	0.68717	0.95001	0.41073	2023-06-29 23:45	seg maskformer + yolov5 ms
VTAMD7	0.675	0.67836	0.93906	0.40769	2023-06-29 15:14	seg maskformer + yolov5 ms
VTAMD7	0.3604	0.18948	0.89118	0.00055	2023-06-28 21:51	seg maskformer + yolov5 ms

提交者	score	seg	cls	det	time	tricks
VTAMD7	0.8415	0.6604	0.95511	0.90895	2023-06-26 10:14	seg maskformer + yolov5 slice 1024
VTAMD7	0.8501	0.67297	0.95015	0.92707	2023-06-26 09:09	seg maskformer + yolov5 1280
VTAMD7	0.6689	0.66563	0.95239	0.3886	2023-06-24 12:05	seg maskformer + yolov5 1280
VTAMD7	0.8465	0.66563	0.95239	0.92156	2023-06-24 11:18	seg maskformer + yolov5
VTAMD7	0.843	0.65927	0.94704	0.92257	2023-06-24 00:19	seg maskformer + det rt-detr fittune 2048 more iter
VTAMD7	0.8477	0.65633	0.95015	0.93671	2023-06-23 12:46	seg maskformer + det rt-detr fittune 2048
VTAMD7	0.8477	0.65633	0.95027	0.9365	2023-06-23 00:07	seg maskformer + det rt-detr fittune 2048
VTAMD7	0.8501	0.67297	0.95015	0.92707	2023-06-21 07:55	seg maskformer + det rt-detr fittune 2048
VTAMD7	0.7223	0.66554	0.94841	0.55308	2023-06-21 00:09	seg maskformer + det rt-detr 2048
VTAMD7	0.8446	0.66554	0.94841	0.91997	2023-06-20 17:57	seg maskformer + det rt-detr 1280
土豆君slov2-tiny真的是神器。	0.8316	0.64107	0.95661	0.89706	2023-06-19 23:25	seg maskformer + det rt-detr

提交者	score	seg	cls	det	time	tricks
VTAMD7	0.8331	0.64439	0.95388	0.90106	2023-06-19 14:04	seg maskformer + det rt-detr
土豆君slov2-tiny真的是神器。	0.8209	0.64971	0.95786	0.85509	2023-06-18 17:39	seg maskformer
VTAMD7	0.7467	0.63127	0.95462	0.65433	2023-06-17 07:13	2048det
VTAMD7	0.8216	0.63127	0.95462	0.87884	2023-06-16 23:34	convnext L upernet
VTAMD7	0.7618	0.49722	0.93958	0.84867	2023-06-14 13:26	swin_tiny

4.训练脚本/代码

paddle2.4.2（mask2former need）

```
cd OneForAll_mask2former
pip install -r requirements.txt

cd OneForAll_mask2former/PaddleYOLO
pip install -e .  or python setup.py install

cd OneForAll_mask2former/PaddleSeg
pip install -e .  or python setup.py install

cd OneForAll_mask2former/PaddleSeg/contrib/PanopticSeg/
pip install -e .  or python setup.py install

cd
OneForAll_mask2former/PaddleSeg/contrib/PanopticSeg/paddlepanseg/models/ops/ms_deform_attn/
python setup.py install
```

sh scripts/train.sh

5.测试脚本/代码

paddle2.3.2

```
python -m pip install paddlepaddle-gpu==2.3.2.post116 -f
https://www.paddlepaddle.org.cn/whl/linux/mkl/avx/stable.html
```

sh scripts/test.sh

6.reference

https://aistudio.baidu.com/aistudio/projectdetail/6337782?sUid=930878&shared=1&ts=1691544272476

https://github.com/xiteng01/CVPR2023_foundation_model_Track1

https://github.com/HandsLing/CVPR2023-Track1-2rd-Solution