## 训练程序（详细参见后面的配置文件，共训练8000轮）：

# !/usr/bin/env python3

# -\*- coding:utf-8 -\*-

from mmengine import Config

from mmengine.runner import Runner

from mmseg.utils import register\_all\_modules

cfg = Config.fromfile('pspnet-watermelon.py')

register\_all\_modules(init\_default\_scope=False)

runner = Runner.from\_cfg(cfg)

runner.train()

## 推理程序：

# !/usr/bin/env python3

# -\*- coding:utf-8 -\*-

import numpy as np

import matplotlib.pyplot as plt

from mmseg.apis import init\_model, inference\_model, show\_result\_pyplot

import mmcv

import cv2

from mmengine.runner import Runner

from mmseg.utils import register\_all\_modules

from mmengine import Config

# 载入 config 配置文件

cfg = Config.fromfile('pspnet-watermelon.py')

checkpoint\_path = './work\_dirs/watermelon/iter\_8000.pth'

model = init\_model(cfg, checkpoint\_path, 'cuda:0')

register\_all\_modules(init\_default\_scope=False)

runner = Runner.from\_cfg(cfg)

img = mmcv.imread('data/watermelon.jpeg')

result = inference\_model(model, img)

print(result.keys()) # ['seg\_logits', 'pred\_sem\_seg']

pred\_mask = result.pred\_sem\_seg.data[0].cpu().numpy()

print(np.unique(pred\_mask)) # [0, 1, 2, 3, 4]

visualization = show\_result\_pyplot(model, img, result, opacity=0.5, out\_file='pred.jpg')

plt.imshow(mmcv.bgr2rgb(visualization))

# plt.show()

plt.savefig('watermelon.png')

## 配置文件：

norm\_cfg = dict(type='BN', requires\_grad=True)

data\_preprocessor = dict(

type='SegDataPreProcessor',

mean=[123.675, 116.28, 103.53],

std=[58.395, 57.12, 57.375],

bgr\_to\_rgb=True,

pad\_val=0,

seg\_pad\_val=255,

size=(256, 256))

model = dict(

type='EncoderDecoder',

data\_preprocessor=dict(

type='SegDataPreProcessor',

mean=[123.675, 116.28, 103.53],

std=[58.395, 57.12, 57.375],

bgr\_to\_rgb=True,

pad\_val=0,

seg\_pad\_val=255,

size=(256, 256)),

pretrained='open-mmlab://resnet50\_v1c',

backbone=dict(

type='ResNetV1c',

depth=50,

num\_stages=4,

out\_indices=(0, 1, 2, 3),

dilations=(1, 1, 2, 4),

strides=(1, 2, 1, 1),

norm\_cfg=dict(type='BN', requires\_grad=True),

norm\_eval=False,

style='pytorch',

contract\_dilation=True),

decode\_head=dict(

type='PSPHead',

in\_channels=2048,

in\_index=3,

channels=512,

pool\_scales=(1, 2, 3, 6),

dropout\_ratio=0.1,

num\_classes=6,

norm\_cfg=dict(type='BN', requires\_grad=True),

align\_corners=False,

loss\_decode=dict(

type='CrossEntropyLoss', use\_sigmoid=False, loss\_weight=1.0)),

auxiliary\_head=dict(

type='FCNHead',

in\_channels=1024,

in\_index=2,

channels=256,

num\_convs=1,

concat\_input=False,

dropout\_ratio=0.1,

num\_classes=6,

norm\_cfg=dict(type='BN', requires\_grad=True),

align\_corners=False,

loss\_decode=dict(

type='CrossEntropyLoss', use\_sigmoid=False, loss\_weight=0.4)),

train\_cfg=dict(),

test\_cfg=dict(mode='whole'))

dataset\_type = 'WatermelonDataset'

data\_root = 'Watermelon87\_Semantic\_Seg\_Mask/'

crop\_size = (256, 256)

train\_pipeline = [

dict(type='LoadImageFromFile'),

dict(type='LoadAnnotations'),

dict(

type='RandomResize',

scale=(2048, 1024),

ratio\_range=(0.5, 2.0),

keep\_ratio=True),

dict(type='RandomCrop', crop\_size=(64, 64), cat\_max\_ratio=0.75),

dict(type='RandomFlip', prob=0.5),

dict(type='PhotoMetricDistortion'),

dict(type='PackSegInputs')

]

test\_pipeline = [

dict(type='LoadImageFromFile'),

dict(type='Resize', scale=(2048, 1024), keep\_ratio=True),

dict(type='LoadAnnotations'),

dict(type='PackSegInputs')

]

img\_ratios = [0.5, 0.75, 1.0, 1.25, 1.5, 1.75]

tta\_pipeline = [

dict(type='LoadImageFromFile', file\_client\_args=dict(backend='disk')),

dict(

type='TestTimeAug',

transforms=[[{

'type': 'Resize',

'scale\_factor': 0.5,

'keep\_ratio': True

}, {

'type': 'Resize',

'scale\_factor': 0.75,

'keep\_ratio': True

}, {

'type': 'Resize',

'scale\_factor': 1.0,

'keep\_ratio': True

}, {

'type': 'Resize',

'scale\_factor': 1.25,

'keep\_ratio': True

}, {

'type': 'Resize',

'scale\_factor': 1.5,

'keep\_ratio': True

}, {

'type': 'Resize',

'scale\_factor': 1.75,

'keep\_ratio': True

}],

[{

'type': 'RandomFlip',

'prob': 0.0,

'direction': 'horizontal'

}, {

'type': 'RandomFlip',

'prob': 1.0,

'direction': 'horizontal'

}], [{

'type': 'LoadAnnotations'

}], [{

'type': 'PackSegInputs'

}]])

]

train\_dataloader = dict(

batch\_size=16,

num\_workers=2,

persistent\_workers=True,

sampler=dict(type='InfiniteSampler', shuffle=True),

dataset=dict(

type='WatermelonDataset',

data\_root=' data/watermelon/',

data\_prefix=dict(

img\_path='img\_dir/train', seg\_map\_path='ann\_dir/train'),

pipeline=[

dict(type='LoadImageFromFile'),

dict(type='LoadAnnotations'),

dict(

type='RandomResize',

scale=(2048, 1024),

ratio\_range=(0.5, 2.0),

keep\_ratio=True),

dict(type='RandomCrop', crop\_size=(64, 64), cat\_max\_ratio=0.75),

dict(type='RandomFlip', prob=0.5),

dict(type='PhotoMetricDistortion'),

dict(type='PackSegInputs')

]))

val\_dataloader = dict(

batch\_size=1,

num\_workers=4,

persistent\_workers=True,

sampler=dict(type='DefaultSampler', shuffle=False),

dataset=dict(

type='WatermelonDataset',

data\_root='data/watermelon/',

data\_prefix=dict(img\_path='img\_dir/val', seg\_map\_path='ann\_dir/val'),

pipeline=[

dict(type='LoadImageFromFile'),

dict(type='Resize', scale=(2048, 1024), keep\_ratio=True),

dict(type='LoadAnnotations'),

dict(type='PackSegInputs')

]))

test\_dataloader = dict(

batch\_size=4,

num\_workers=4,

persistent\_workers=True,

sampler=dict(type='DefaultSampler', shuffle=False),

dataset=dict(

type='WatermelonDataset',

data\_root='Watermelon87\_Semantic\_Seg\_Mask/',

data\_prefix=dict(img\_path='img\_dir/val', seg\_map\_path='ann\_dir/val'),

pipeline=[

dict(type='LoadImageFromFile'),

dict(type='Resize', scale=(2048, 1024), keep\_ratio=True),

dict(type='LoadAnnotations'),

dict(type='PackSegInputs')

]))

val\_evaluator = dict(type='IoUMetric', iou\_metrics=['mIoU'])

test\_evaluator = dict(type='IoUMetric', iou\_metrics=['mIoU'])

default\_scope = 'mmseg'

env\_cfg = dict(

cudnn\_benchmark=True,

mp\_cfg=dict(mp\_start\_method='fork', opencv\_num\_threads=0),

dist\_cfg=dict(backend='nccl'))

vis\_backends = [dict(type='LocalVisBackend')]

visualizer = dict(

type='SegLocalVisualizer',

vis\_backends=[dict(type='LocalVisBackend')],

name='visualizer')

log\_processor = dict(by\_epoch=False)

log\_level = 'INFO'

load\_from = None

resume = False

tta\_model = dict(type='SegTTAModel')

optimizer = dict(type='SGD', lr=0.01, momentum=0.9, weight\_decay=0.0005)

optim\_wrapper = dict(

type='OptimWrapper',

optimizer=dict(type='SGD', lr=0.01, momentum=0.9, weight\_decay=0.0005),

clip\_grad=None)

param\_scheduler = [

dict(

type='PolyLR',

eta\_min=0.0001,

power=0.9,

begin=0,

end=40000,

by\_epoch=False)

]

train\_cfg = dict(type='IterBasedTrainLoop', max\_iters=8000, val\_interval=400)

val\_cfg = dict(type='ValLoop')

test\_cfg = dict(type='TestLoop')

default\_hooks = dict(

timer=dict(type='IterTimerHook'),

logger=dict(type='LoggerHook', interval=100, log\_metric\_by\_epoch=False),

param\_scheduler=dict(type='ParamSchedulerHook'),

checkpoint=dict(type='CheckpointHook', by\_epoch=False, interval=2000),

sampler\_seed=dict(type='DistSamplerSeedHook'),

visualization=dict(type='SegVisualizationHook'))

work\_dir = './work\_dirs/Watermelon'

randomness = dict(seed=0)

最终结果第8000轮

#### 测试精度

+------------+-------+-------+

| Class | IoU | Acc |

+------------+-------+-------+

| background | 89.67 | 93.67 |

| red | 88.34 | 98.19 |

| green | 62.03 | 69.47 |

| white | 76.06 | 87.34 |

| seed-black | 57.23 | 63.66 |

| seed-white | 39.31 | 41.92 |

+------------+-------+-------+

06/15 10:45:29 - mmengine - INFO - Iter(test) [10/10] aAcc: 92.4400 mIoU: 68.7700 mAcc: 75.7100 data\_time: 0.0169 time: 1.018

## 原图：



## 分割图：

