

$$i\hbar\partial_t\psi(\mathbf{r},t) = H\psi(\mathbf{r},t)$$

$$= \left[-\frac{\hbar^2}{2m} \nabla^2 + U(\mathbf{r}) \right] \psi(\mathbf{r},t)$$

2025年10月28号

人工智能系统综合实践

cityscapes 数据集上的图像语义分割

胡 & 姜 著

学院：信息工程学院

学号：1210604006 & 1221005171

姓名：胡 & 姜

日期：2025 年 10 月 28 号

首都师范大学

在线文档地址

1 目录

目录

1 目录	1
2 任务分析	2
3 设计思路	3
4 系统实现	3
4.1 数据分析	3
4.1.1 数据分析代码 (类)	3
4.1.2 数据可视化代码 (类)	4
4.2 数据集处理	5
4.3 deeplabv3plus 的训练, 测试与可视化	6
5 优化前后对比	10
5.0.1 尝试一: 调整优化器和损失	10
5.0.2 尝试二: 修改 backbone 骨干网络	11
5.0.3 参数量和计算量对比	13
5.0.4 可视化对比	14
6 mask2former	16
6.1 疑问	17
7 分工	18
8 总结与展望	18

2 任务分析

对于 cityscapes 数据集进行语义分割，需要完成以下任务：

- ✓ 1. 数据分析
- ✓ 2. 数据增强
 - ✓ 缩放 & 裁剪 & 水平翻转
- ✓ 3. deeplabv3plus-r50 模型训练
- ✓ 4. deeplabv3plus-r50 模型测试以及可视化
 - ✓ loss & mIoU 曲线
 - ✓ 预测结果 & CAM 图
 - ✓ 混淆矩阵
- ✓ 5. 绘制模型结构
- ✓ 6. 模型优化
 - ✓ 调整优化器，组合损失函数等
 - ✓ 更换 backbone 为 hrnet
 - 增加数据增强方法
- ✓ 7. 结果对比
 - ✓ 不同优化器结果对比
 - ✓ 不同 backbone 结果对比
 - ✓ 参数量和计算量对比
 - ✓ mIoU 和准确率对比
 - ✓ 模型结构对比
 - ✓ 预测结果对比
 - ✓ 混淆矩阵对比

3 设计思路

对数据集进行分析处理, 分析图像均值、标准差、不同类别像素数量。将计算得到的均值和标准差用于数据归一化。使用随机缩放、随机裁剪、随机水平翻转等方法进行数据增强。选择 deeplabv3plus-r50 模型进行训练, 使用交叉熵损失函数和 SGD 优化器。训练过程中记录 loss 和 mIoU 曲线, 并在测试集上进行测试, 生成预测结果和 CAM 图, 绘制混淆矩阵。最后对模型进行优化, 调整优化器 (AdamW) 和组合损失函数 (添加 FocalLoss), 更换 backbone 为 hrnet, 并对不同配置下的结果进行对比分析。

4 系统实现

4.1 数据分析

1. 分析图像均值、标准差、不同类别像素数量。

4.1.1 数据分析代码 (类)

CityscapesAnalyzer 类中定义了 5 个函数。

表 1: 数据分析函数与功能说明

序号	函数	功能
1	<code>get_image_paths(self, split='train')</code>	地址处理, 输入 cityscapes 地址, 输出图像和标签的地址。
2	<code>calculate_mean_std(self, image_paths, sample_size=1000)</code>	计算均值和标准差, 采样大小是对多少图像进行计算 (共 2975 张), 返回均值和标准差。
3	<code>analyze_class_distribution(self, label_paths, sample_size=500)</code>	分析类的分布, 每个类别占比是多少, 返回各类别频率。
4	<code>analyze_image_sizes(self, image_paths, sample_size=500)</code>	统计图像尺寸 (全是 2048×1024), 这个函数没啥用。
5	<code>save_stats(self, save_path='xxx')</code>	保存统计的数据, 下次就不用计算了。

4.1.2 数据可视化代码 (类)

表 2: 绘图函数与功能说明

序号	函数	功能
1	<code>plot_mean_std(self, mean, std, save_path=None, figsize=(12, 5))</code>	绘制 RGB 三通道的均值和标准差的条形图。
2	<code>plot_class_distribution(self, class_distribution, save_path=None, figsize=(14, 12))</code>	绘制各类别占比的分布图和饼图。
3	<code>plot_image_sizes(self, size_stats, save_path=None, figsize=(10, 5))</code>	绘制图像大小占比的分布图, 这个函数没啥用。

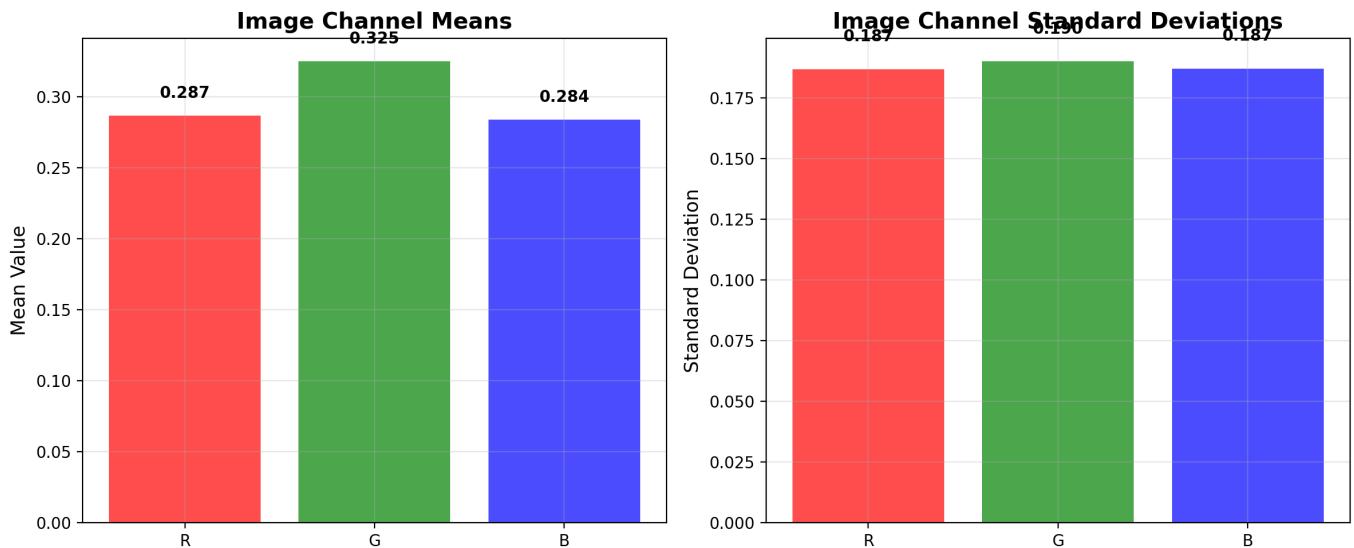


图 1: cityscapes 数据集图像均值与标准差

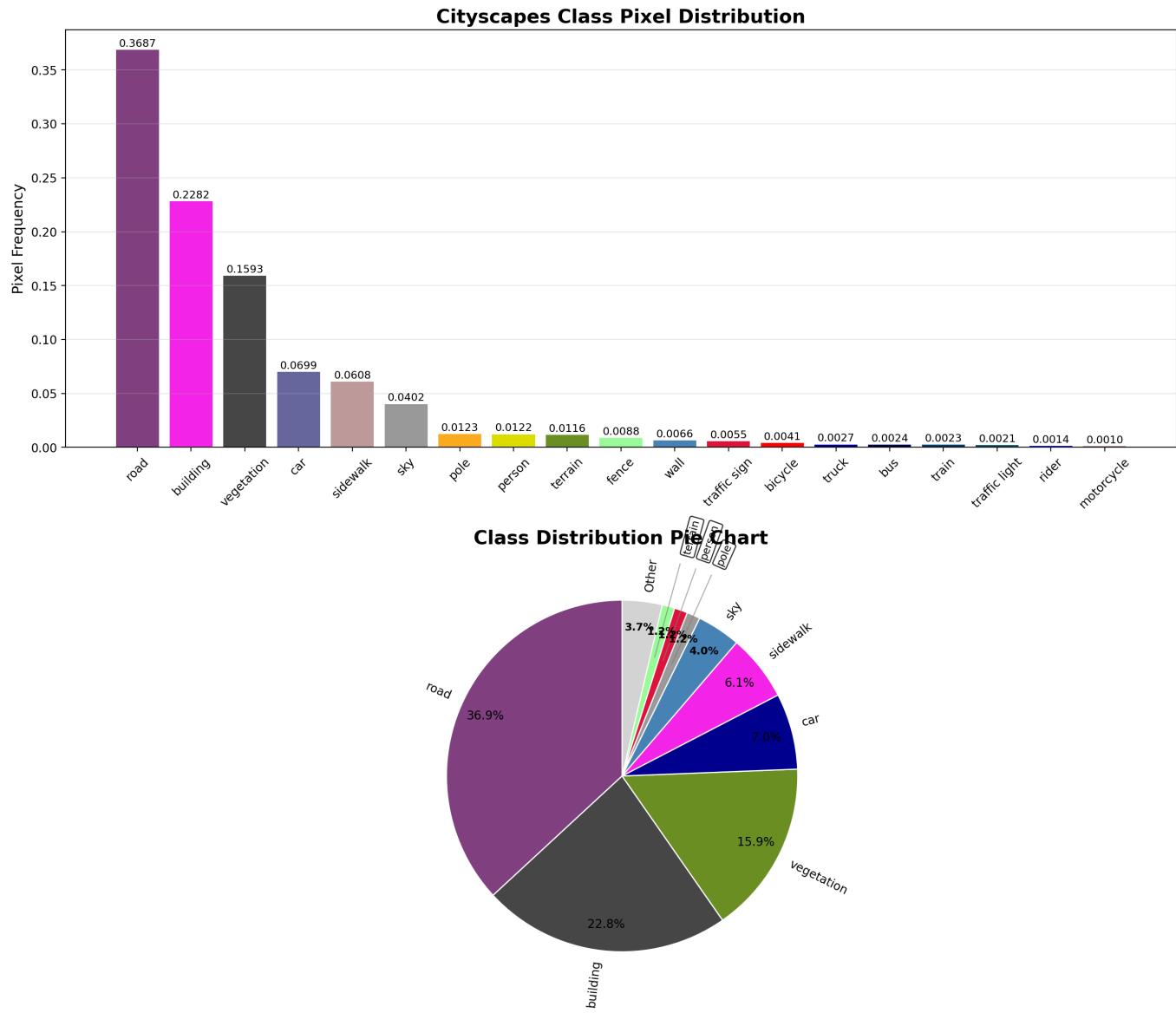


图 2: cityscapes 数据集各类别占比

4.2 数据集处理

下载好 cityscapes 数据集后解压，放到..../mmsegmentation/data/中，运行

```
python tools/dataset_converters/cityscapes.py ./data/cityscapes --nproc 8
```

对数据集进行处理，之后就可以进行训练了。

对于图像裁切,翻转,尺度变换等数据增强甚至光照变化都在 mmsegmentation 的./config/_base_/dataset 中进行了定义。

```

1 # dataset settings
2 dataset_type = 'CityscapesDataset'
3 data_root = 'data/cityscapes/'
4 crop_size = (512, 1024)

```

```

5 train_pipeline = [
6     dict(type='LoadImageFromFile'),
7     dict(type='LoadAnnotations'),
8     dict(
9         type='RandomResize',
10        scale=(2048, 1024),
11        ratio_range=(0.5, 2.0),
12        keep_ratio=True),
13     dict(type='RandomCrop', crop_size=crop_size, cat_max_ratio=0.75),
14     dict(type='RandomFlip', prob=0.5),
15     dict(type='PhotoMetricDistortion'),
16     dict(type='PackSegInputs')
17 ]

```

4.3 deeplabv3plus 的训练，测试与可视化

1. 开始训练

使用 deeplabv3plus_r50-d8_4xb2-80k_cityscapes-512x1024.py 进行训练。训练结果如下：

Class	IoU	Acc
road	96.46	97.5
sidewalk	80.23	89.45
building	88.73	95.47
wall	34.81	37.31
fence	51.58	65.53
pole	59.78	74.3
traffic light	64.04	77.32
traffic sign	72.82	81.9
vegetation	91.0	96.6
terrain	56.71	63.49
sky	93.27	97.52
person	77.4	86.94
rider	53.55	78.05
car	92.94	97.23
truck	50.72	68.15
bus	56.41	62.71
train	35.35	41.1
motorcycle	45.51	52.7
bicycle	70.84	83.11

10/24 01:14:40 - mmengine - INFO - Iter(val) [500/500] aAcc: 94.3700 mIoU: 66.9600 mAcc: 76.1200 data_time: 0.0099 time: 0.1673

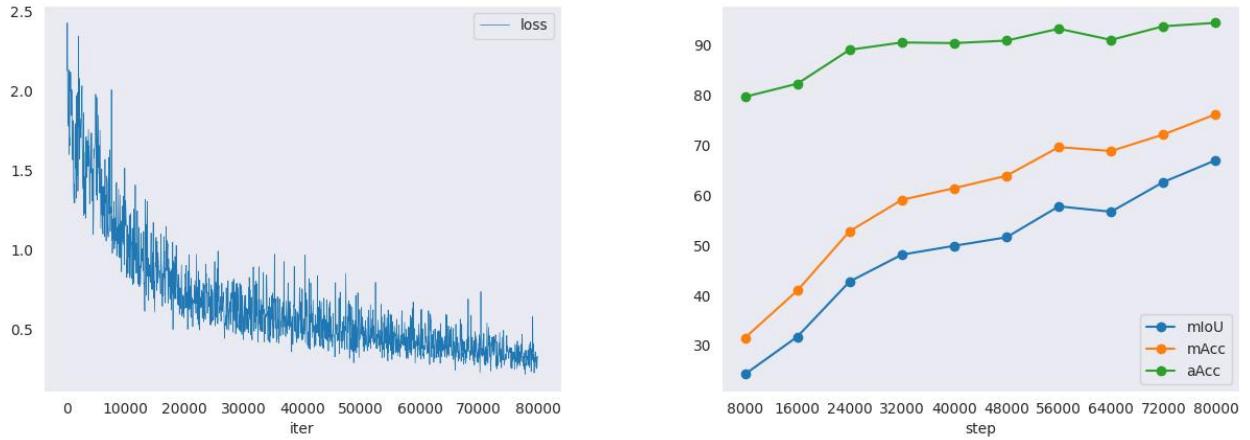


图 3: deeplabv3plus 模型的 loss 曲线 (左) 与 mIoU 曲线 (右)

分割结果展示如下:

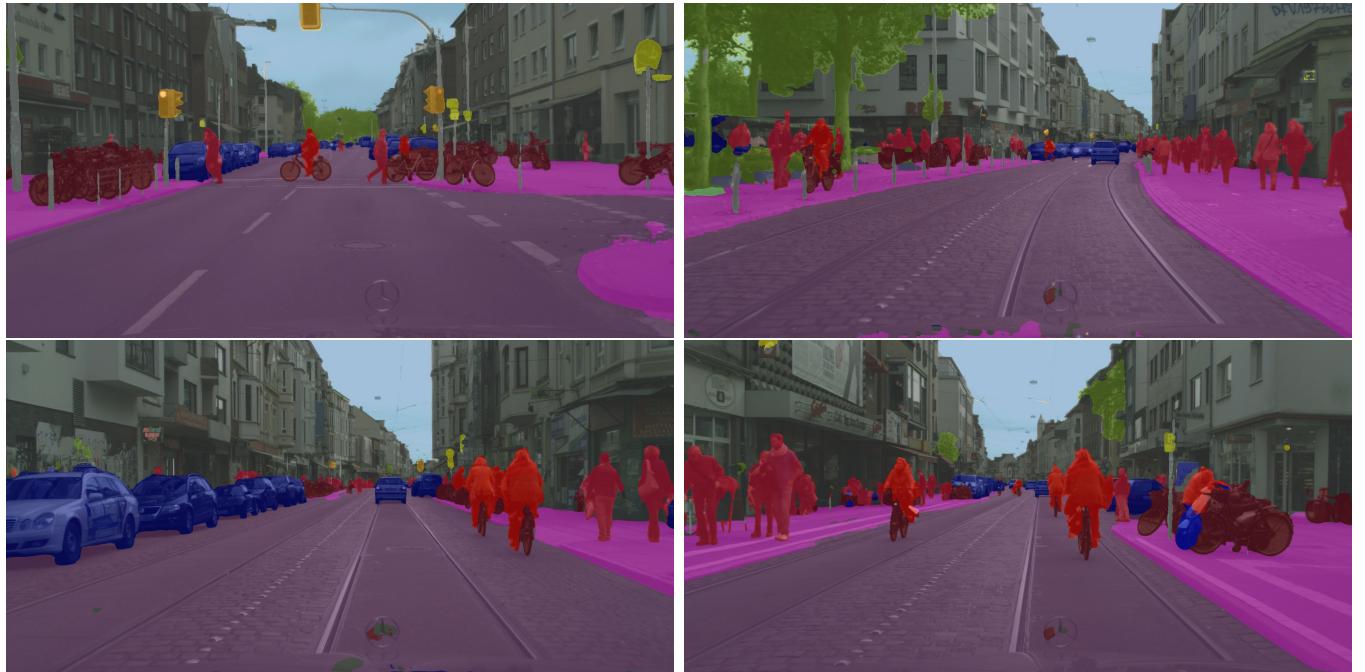


图 4: deeplabv3plus 模型的分割结果展示



图 5: deeplabv3plus 模型的预测结果展示



图 6: deeplabv3plus 模型的 cam 图展示

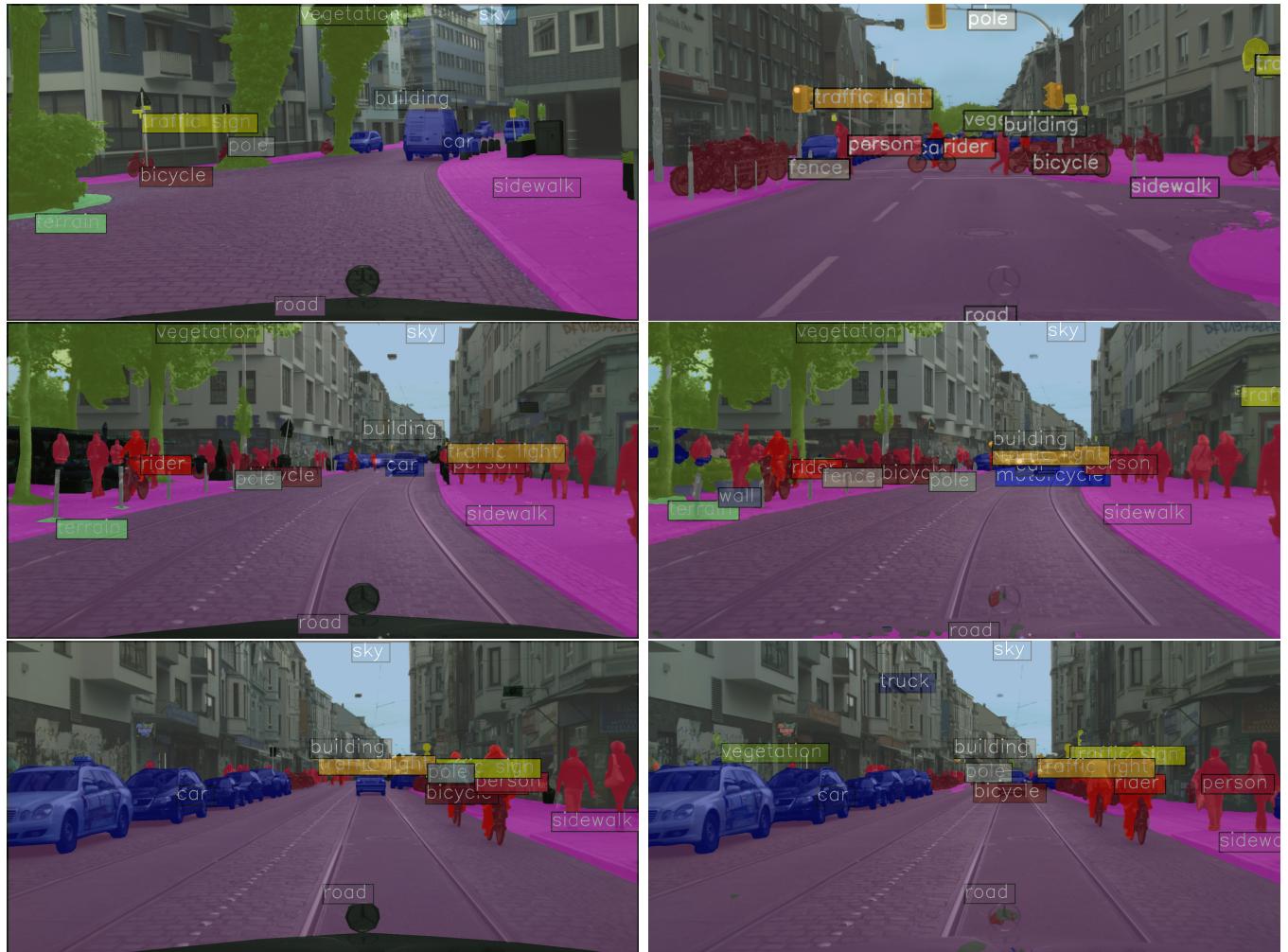


图 7: 标注数据 (左) 和预测数据 (右) 对比

这个代码在 mmsegmentation 中没有, 可以在文档中找到这段代码, 并添加到文件中。这是代码所在的链接 https://github.com/open-mmlab/mmsegmentation/blob/main/docs/en/user_guides/visualizations/wandb : <https://wandb.ai/alionf-robodiff/uncategorized/runs/njilzcwj>

预测的图像和原来标注的数据基本保持一致, 在某些图片中分割出的物体更多, 比如7中第二张图将单元门口上的灯检测成了路灯, 7中第三张图将小区的窗户当成了卡车, 对于自动驾驶来说, 这样的检测效果并不影响车辆的行驶。如果无法完美的匹配标注的数据, 那么检测出更多的物体比检测不出物体更好。

5 优化前后对比

5.0.1 尝试一：调整优化器和损失

deeplabv3plus 使用的是 SGD 优化器和交叉熵损失函数，我们将优化器改为 AdamW，并采用组合损失的形式，在交叉熵的基础上添加 FocalLoss，因为进行数据分析时得到各个类别的占比不同，采用 focalloss 来缓解数据分布不均衡的问题

```

1 # 使用组合损失函数
2 # loss_decode=dict(type='CrossEntropyLoss', use_sigmoid=False, loss_weight=1.0)),
3     loss_decode=[
4         dict(type='CrossEntropyLoss', use_sigmoid=False, loss_weight=1.0),
5         dict(type='FocalLoss', loss_weight=0.5, gamma=2.0, alpha=0.25)
6     ],
7 # 更换优化器
8 # optimizer = dict(type='SGD', lr=0.01, momentum=0.9, weight_decay=0.0005)
9 optimizer = dict(type='AdamW', lr=0.001, betas=(0.9, 0.999), weight_decay=0.01)
```

结果如下：

Class	IoU	Acc
road	97.53	98.9
sidewalk	81.78	89.42
building	90.78	95.04
wall	32.08	35.23
fence	48.6	64.57
pole	61.37	77.97
traffic light	65.15	80.14
traffic sign	72.09	86.23
vegetation	90.76	96.63
terrain	48.42	53.8
sky	92.62	98.37
person	77.06	87.68
rider	54.4	74.12
car	91.98	96.04
truck	43.18	74.06
bus	36.22	40.58
train	7.7	7.9
motorcycle	47.69	61.45
bicycle	72.95	84.52

10/26 01:50:54 - mmengine - INFO - Iter(val) [500/500] aAcc: 94.6500 mIoU: 63.8100 mAcc: 73.8200 data_time: 0.0112 time: 0.1686

mIoU 还降低了 3 个点，效果不是很好。

5.0.2 尝试二：修改 backbone 骨干网络

backbone 将官方提供的 resnet-50 替换为 hrnet

```
1 model = dict(
2     type='EncoderDecoder',
3     data_preprocessor=data_preprocessor,
4     pretrained='open-mmlab://msra/hrnetv2_w32',
5     backbone=dict(
6         type='HRNet',
7         extra=dict(
8             stage1=dict(
9                 num_modules=1,
10                num_branches=1,
11                block='BOTTLENECK',
12                num_blocks=(4,),
13                num_channels=(64,)),
14            stage2=dict(
15                num_modules=1,
16                num_branches=2,
17                block='BASIC',
18                num_blocks=(4, 4),
19                num_channels=(32, 64)),
20            stage3=dict(
21                num_modules=4,
22                num_branches=3,
23                block='BASIC',
24                num_blocks=(4, 4, 4),
25                num_channels=(32, 64, 128)),
26            stage4=dict(
27                num_modules=3,
28                num_branches=4,
29                block='BASIC',
30                num_blocks=(4, 4, 4, 4),
31                num_channels=(32, 64, 128, 256))),
32        norm_cfg=norm_cfg),
33    decode_head=dict(
34        type='DepthwiseSeparableASPPHead',
35        in_channels=256, # HRNet 最终输出通道数
36        in_index=3, # 使用 HRNet 的最后一个输出
37        channels=512,
38        dilations=(1, 12, 24, 36),
39        c1_in_channels=32, # HRNet stage2 的输出通道数
```

```
40     c1_channels=48,
41     dropout_ratio=0.1,
42     num_classes=19,
43     norm_cfg=norm_cfg,
44     align_corners=False,
45     loss_decode=dict(type='CrossEntropyLoss', use_sigmoid=False, loss_weight=1.0)
46     ),
47     auxiliary_head=dict(
48         type='FCNHead',
49         in_channels=128,
50         in_index=2,
51         channels=256,
52         num_convs=1,
53         concat_input=False,
54         dropout_ratio=0.1,
55         num_classes=19,
56         norm_cfg=norm_cfg,
57         align_corners=False,
58         loss_decode=dict(
59             type='CrossEntropyLoss', use_sigmoid=False, loss_weight=0.4)),
60     train_cfg=dict(),
61     test_cfg=dict(mode='whole'))
```

结果如下：

1	Class	IoU	Acc
2	road	98.04	98.76
3	sidewalk	84.14	92.99
4	building	91.68	96.42
5	wall	47.09	51.32
6	fence	59.11	71.92
7	pole	64.39	75.53
8	traffic light	67.52	77.51
9	traffic sign	77.9	85.58
10	vegetation	92.09	96.87
11	terrain	60.12	66.58
12	sky	93.65	95.94
13	person	80.55	89.63
14	rider	59.71	76.24
15	car	94.11	97.54
16	truck	70.43	84.99
17	bus	64.96	76.05
18	train	33.13	35.41

```

21 | motorcycle | 46.01 | 53.64 |
22 | bicycle   | 73.77 | 87.24 |
23 +-----+-----+-----+
24 10/26 16:53:49 - mmengine - INFO - Iter(val) [500/500]      aAcc: 95.6400  mIoU:
    71.5000  mAcc: 79.4800  data_time: 0.0105  time: 0.0984

```

这个结果相当不错，比原本的 resnet 提高了 5 个点，我感觉 80k 批次没有完全收敛，改成 160k 试一下。

```

1 10/26 16:53:49 - mmengine - INFO - Iter(val) [500/500]      aAcc: 95.6400  mIoU:
    71.5000  mAcc: 79.4800  data_time: 0.0105  time: 0.0984

```

修改为 160K 之后的训练效果并没有显著提升，并且在训练中期出现了 mIoU 的下降，之后又回升。可能是学习率设置的太高导致的。

由于 160k 次训练和 80k 的效果相近，所以后面使用 80k 的模型进行对比。

5.0.3 参数量和计算量对比

```

1 =====
2 Compute type: direct: randomly generate a picture
3 Input shape: (2048, 1024)
4 Flops: 1.413T
5 Params: 41.225M
6 =====
7
8 =====
9 Compute type: direct: randomly generate a picture
10 Input shape: (2048, 1024)
11 Flops: 0.431T
12 Params: 42.346M
13 =====

```

表 3: 参数量和计算量对比

序号	计算量 (Flops)	参数量 (Params)
1	1.413T	41.225M
2	0.431T	42.346M

5.0.4 可视化对比

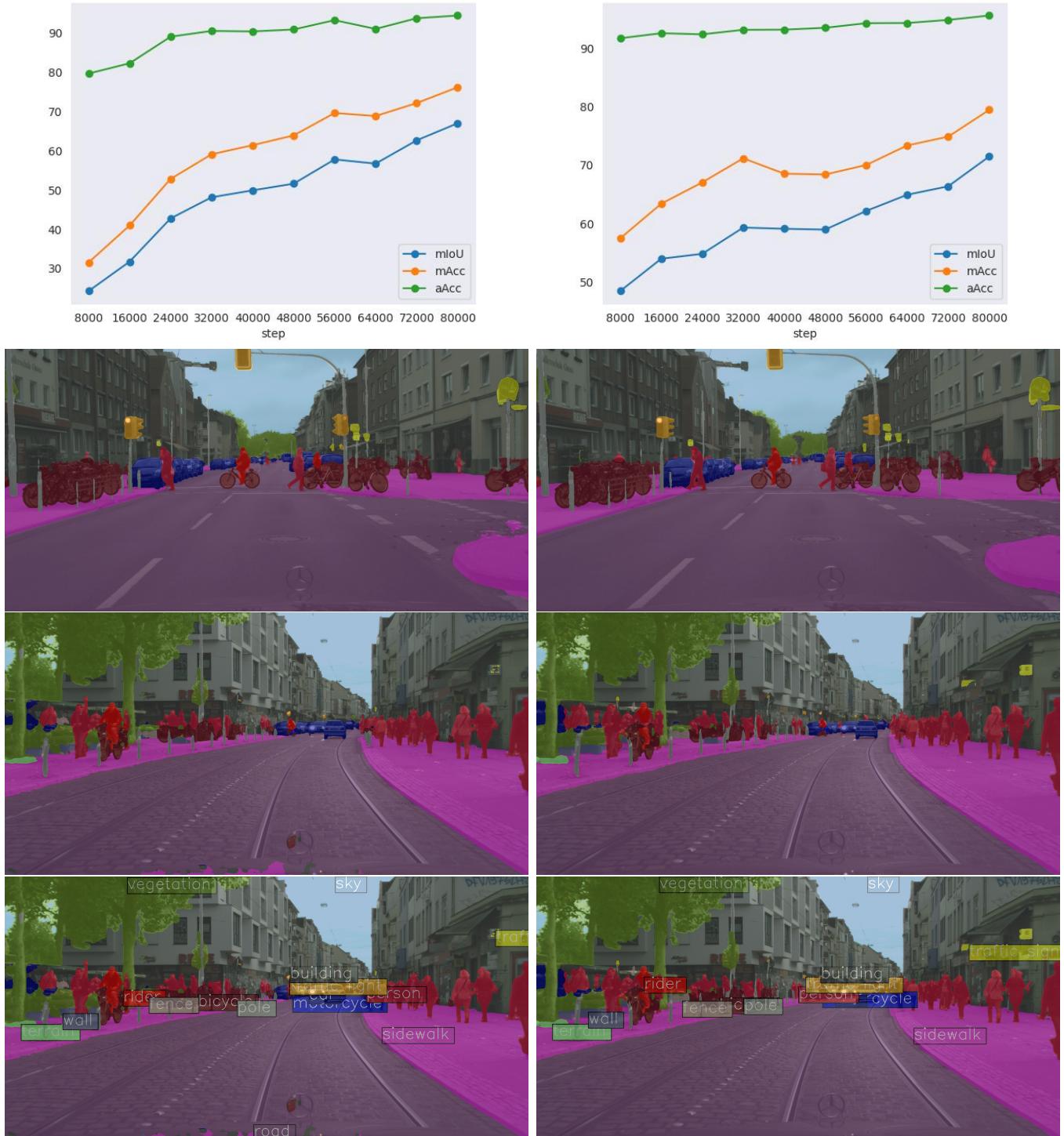


图 8: resnet-r50 骨干网络 (左) 和 hrnet 骨干网络 (右) 分割结果对比

从上面的结果可以看出，hrnet 作为骨干网络时，mIoU 提升了 5 个点，计算量减少了约 3 倍，参数量略有增加，但增加并不明显。分割结果相近。

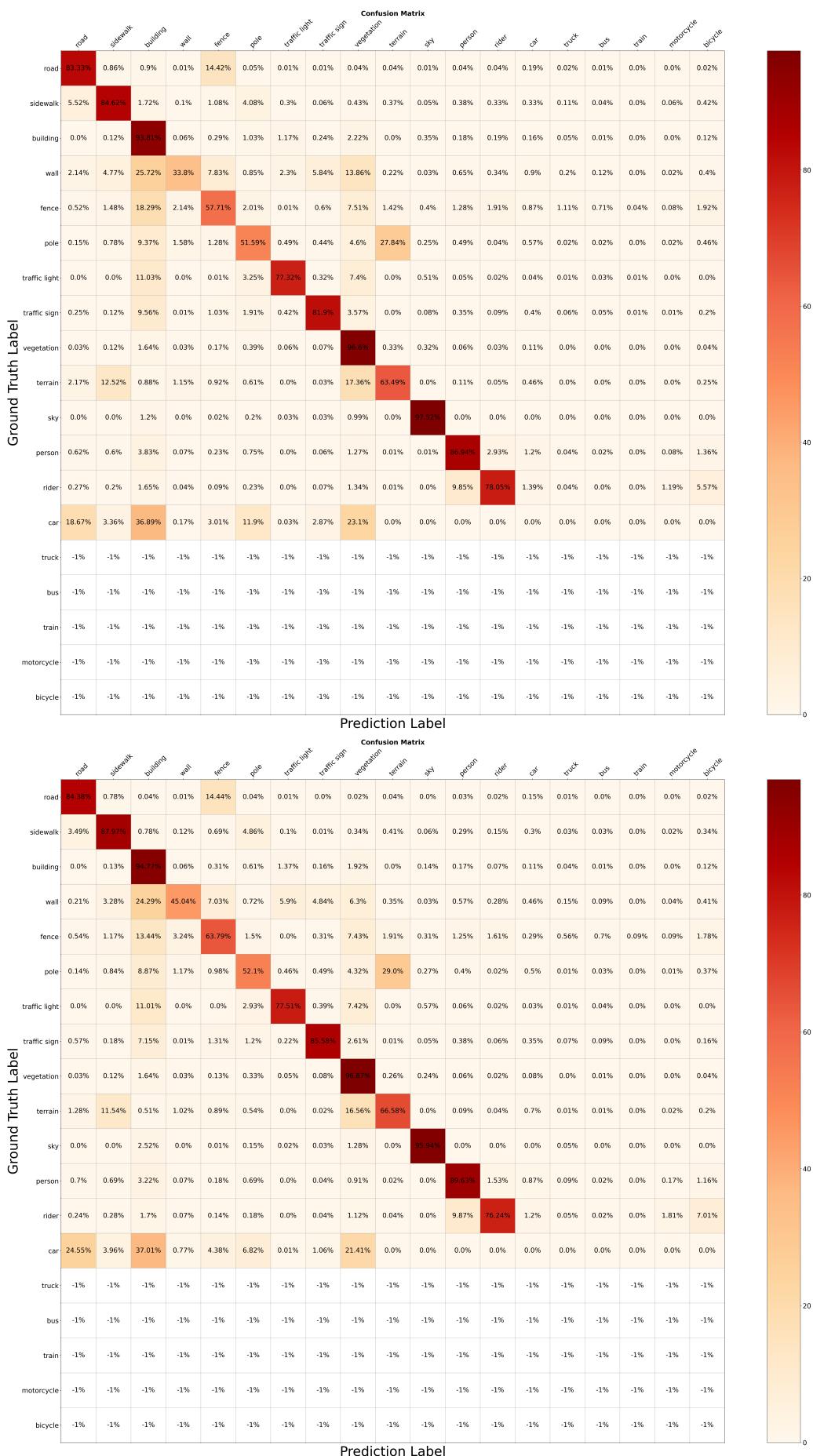


图 9: 混淆矩阵对比 resnet-r50(上)hrnet(下)

观察 deeplabv3plus-r50 的混淆矩阵, 模型对“墙”的预测效果较差, 25.72% 的概率预测为建筑物 (好像也没啥毛病), 13.86% 概率预测为植被 (观察数据集, 有些植被和墙形成了相互遮挡的关系), 7.83% 的概率预测为栅栏。同样的栅栏也有 18.29% 的概率被预测为墙。小汽车完全没有预测成功, 按照占比排列分别预测为建筑物 (36.89%), 植被 (23.1%), 路 (18.67%) 和杆 (11.9%), 可是看预测图5, 里面有 car 的预测结果, 不能就这一张预测到了吧!!!

6 mask2former

使用 mask2former_r50_8xb2-90k_cityscapes-512x1024.py 进行训练, 结果如下:

	Class	IoU	Acc
1	road	98.18	98.74
2	sidewalk	85.45	94.45
3	building	92.91	96.56
4	wall	53.09	63.58
5	fence	61.31	77.18
6	pole	67.98	78.16
7	traffic light	73.02	83.53
8	traffic sign	81.03	87.14
9	vegetation	92.47	96.08
10	terrain	62.94	79.07
11	sky	94.81	98.35
12	person	83.84	90.99
13	rider	67.07	81.54
14	car	95.35	97.51
15	truck	76.71	90.54
16	bus	84.41	94.37
17	train	56.59	61.95
18	motorcycle	61.6	76.49
19	bicycle	79.2	89.25
20			
21			
22			
23			
24			2025/10/25 03:09:36 - mmengine - INFO - Iter(val) [500/500] aAcc: 96.1600 mIoU: 77.2600 mAcc: 86.0800 data_time: 0.0103 time: 0.1363

mIoU 比 deeplabv3plus-r50 高 10 个点, 效果非常好。

6.1 疑问

Cityscapes

Method	Backbone	Crop Size	Lr schd	Mem (GB)	Inf time (fps)	Device	mIoU	mIoU(ms+flip)	config	download
DeepLabV3+	R-50-D8	512x1024	40000	7.5	3.94	V100	79.61	81.01	config	model log
DeepLabV3+	R-101-D8	512x1024	40000	11	2.60	V100	80.21	81.82	config	model log
DeepLabV3+	R-50-D8	769x769	40000	8.5	1.72	V100	78.97	80.46	config	model log
DeepLabV3+	R-101-D8	769x769	40000	12.5	1.15	V100	79.46	80.50	config	model log
DeepLabV3+	R-18-D8	512x1024	80000	2.2	14.27	V100	76.89	78.76	config	model log
DeepLabV3+	R-50-D8	512x1024	80000	-	-	V100	80.09	81.13	config	model log

Cityscapes

Method	Backbone	Crop Size	Lr schd	Mem (GB)	Inf time (fps)	Device	mIoU	mIoU(ms+flip)	config	download
Mask2Former	R-50-D32	512x1024	90000	5.67	9.17	A100	80.44	-	config	model log
Mask2Former	R-101-D32	512x1024	90000	6.81	7.11	A100	80.80	-	config	model log
Mask2Former	Swin-T	512x1024	90000	6.36	7.18	A100	81.71	-	config	model log
Mask2Former	Swin-S	512x1024	90000	8.09	5.57	A100	82.57	-	config	model log
Mask2Former	Swin-B (in22k)	512x1024	90000	10.89	4.32	A100	83.52	-	config	model log
Mask2Former	Swin-L (in22k)	512x1024	90000	15.83	2.86	A100	83.65	-	config	model log

图 10: 官方给出的效果 deeplabv3plus(上)mask2former(下)

按照 mmsegmentation 提供的文件进行训练，得到的 mIoU 只有 66.96，与 github 中给出的 80.09 相差很多，而同样是官方提供的 mask2former 配置文件训练得到的 mIoU 有 77.26，与标注的 80.44 相差并不是很多。

表 4: 参数量和计算量对比

序号	姓名 (Name)	分工 (Works)
1	胡	deeplabv3plus 训练与优化 文档 模型结构可视化 绘制混淆矩阵
2	姜	数据分析 mask2former 训练 模型结果可视化 PPT 制作

7 分工

8 总结与展望