Supplementary Material: Part-aware Panoptic Segmentation

Daan de Geus[1*] Panagiotis Meletis[1*] Chenyang Lu[1] Xiaoxiao Wen[2] Gijs Dubbelman[1]
[1]Eindhoven University of Technology [2]University of Amsterdam
{d.c.d.geus, p.c.meletis} @tue.nl

We provide the following information in addition to the paper:

1. Elaborate implementation details for the methods on the subtasks used for generating the baselines on part-aware panoptic segmentation, in Section 1.
2. Detailed per-class results of the state-of-the-art baselines, in Section 2.
3. Details about the annotation procedure for Cityscapes Panoptic Parts, in Section 3.
4. Additional qualitative examples of annotations of the presented datasets, and predictions of the baselines, in Section 4.

Code and data: https://github.com/tue-mps/panoptic_parts.

1. Implementation details baselines

For reproducibility, we provide more information about the implementation details of the methods used to generate baselines for part-level panoptic segmentation (PPS).

To guarantee the state-of-the-art performance of the used methods, we use existing trained models when possible. If there is no trained model available, we train a network ourselves.

1.1. Citycapes Panoptic Parts

The baseline results for Cityscapes Panoptic Parts (CPP), as presented in Section 5.1.2 and Table 2 of the main paper, are generated by merging results from existing methods. Below, we describe, for each method in Table 2, how we acquire the results of these methods.

For CPP, all models are trained only on the images in the Cityscapes `train` split [7], unless otherwise indicated.

1.1.1 Panoptic segmentation

For panoptic segmentation results on Cityscapes Pascal Parts, we use both single-network panoptic segmentation methods, and results generated by merging semantic and instance segmentation methods following the heuristics presented in [11].

EfficientPS. For state-of-the-art panoptic segmentation method EfficientPS [14], the predictions for the Cityscapes `val` were generously provided to us by the authors of the work. During inference, multi-scale testing is applied.

UPSNet. The results for UPSNet [19] were generated using the official code repository. Specifically, we run inference using the trained model with a ResNet-50 backbone [10].

HRNet-OCR & PolyTransform. To get further state-of-the-art panoptic segmentation results, we fuse the state-of-the-art semantic segmentation and instance segmentation results from HRNet-OCR [20] and PolyTransform [12], respectively.

The HRNet-OCR [20] results are generated using a trained model from the official code repository. Specifically, we pick the model with a HRNetv2-W48 [18] backbone, without using test-time augmentations.

For PolyTransform [12], the results on the Cityscapes `val` split were generously provided to us by the authors of the paper. We note that this instance segmentation model is pre-trained on the COCO dataset [13].

DeepLabv3+ & Mask R-CNN. The results for both DeepLabv3+ [4] and Mask R-CNN [10] are generated using existing trained models from official code repositories.

For DeepLabv3+ [4], we select a model with an Xception-65 backbone [6], and we do not use test-time augmentations.

For Mask R-CNN [9], we generate the results using a trained model with a ResNet-50 backbone [10], pre-trained on the COCO dataset [13].

1.1.2 Part segmentation

To generate the part-level segmentation predictions, which are required to generate the PPS predictions, we use two networks to perform the part segmentation task: state-of-the-art

*Both authors contributed equally.

BSANet [23] and commonly used DeepLabv3+ [4]. Since the part-level annotations of Cityscapes dataset [7] are newly proposed by us, there are no trained models available, so we train two networks with settings similar to those proposed in [23, 4].

BSANet. We use the official repository provided by the authors of BSANet [23] and keep most of the training settings same as in the official version. We change the crop size for training to 512×1024 and the number of output classes according to our part label definition. During the training, an SGD optimizer is applied with polynomial learning rate decay. We set base learning rate to 0.01, decay power to 0.9, and weight decay to 4e-5. We train the network for 100 epochs with batch size of 3.

DeepLabv3+. We use the popular *mmsegmentation* [16] respository to train a DeepLabv3+ [4] model. We use one of default configurations provided by the repository, with a ResNet-50 backbone [10]. The crop size during training is set to 769×769. Again, we use an SGD optimizer with polynomial learning rate decay. The base learning rate, decay power, and weight decay are set to 0.01, 0.9, and 5e-4, respectively. We train the network for 40k iterations with batch size of 4.

1.2. Pascal Panoptic Parts

For the baselines results for Pascal Panoptic Parts (PPP), as presented in Section 5.1.3 and Table 3 of the main paper, we also provide further implementation details.

Again, to guarantee state-of-the-art performance performance on the subtasks, and to facilitate reproducibility, we use publicly available trained models when possible. Methods for semantic segmentation are trained on 59 classes of Pascal-Context [15]; part segmentation is trained on 58 part-level classes from Pascal-Parts [5], as defined in [23]. We train the instance segmentation models on the 20 *things* classes from our PPP dataset. We evaluate all methods on our annotations for the PPP `validation` set, using the same class definition as for training.

As explained in the main manuscript, there are discrepancies between the various different annotation sets for Pascal VOC 2010 [8]. In our Pascal Panoptic Part dataset, we resolve such conflicts, and generate consistent annotations for multiple levels of abstraction. As a result, the absolute scores on the other annotation sets for Pascal VOC 2010 are not directly comparable with results on our PPP.

1.2.1 Panoptic segmentation

Panoptic segmentation results for PPP are generated by fusing semantic segmentation and instance segmentation results using the heuristics described in [11].

DeepLabv3-ResNeSt269 & DetectoRS. State-of-the-art results for semantic segmentation are generated using a DeepLabv3 model [3] with a ResNeSt-269 backbone [22]. For this, we use a trained model from the *PyTorch-Encoding* repository [21], which is the official semantic segmentation repository for the paper introducing ResNeSt [22].

For instance segmentation, there is no trained model available, so we train a model using the commonly used *mmdetection* repository [2]. Specifically, to generate state-of-the-art results, we train a DetectoRS [17] model with a HTC-ResNet-50 backbone [1, 10]. We do not use auxiliary semantic labels for training. For training, we use a batch size of 4, a learning rate of 0.0025 and weight decay of 1e-4. We train for 24 epochs and decrease the learning rate by a factor of 10 after 18 epochs.

DeepLabv3 & Mask R-CNN. To set another reference for semantic segmentation, we also train a DeepLabv3 [3] model with a ResNet-50 backbone [10]. Again, we use the *PyTorch-Encoding* repository [21]. This model is trained for 80 epochs, with a batch size of 16, a weight decay of 1e-4, and a polynomial learning rate schedule with an initial learning rate of 0.001 and a decay of 0.9.

For instance segmentation results on Mask R-CNN [9], we train another model using the *mmdetection* repository [2]. Specifically, we train Mask R-CNN with a ResNet-50 backbone [10]. Again, we use a batch size of 4, a learning rate of 0.0025 and weight decay of 1e-4. We train for 24 epochs and reduce the learning rate by a factor of 10 after 18 epochs.

1.2.2 Part segmentation

To generate part segmentation results for PPP, we use the same two networks as for CPP (See Section 1.1).

BSANet. The `validation` set prediction samples of [23] are made publicly available by the authors. Therefore, we directly use their predictions in our experiments.

DeepLabv3+. To generate DeepLabv3+ [4] predictions for PPP, we also train a network using the *mmsegmentation* repository [16]. As in Section 1.1, we use one of the default configurations provided by *mmsegmentation*, with a ResNet-50 backbone [10]. Again, we use an SGD optimizer with polynomial learning rate decay. The base learning rate, decay power, and weight decay are set to 0.004, 0.9, and 1e-4, respectively. We train the network for 40k iterations with batch size of 8.

2. Detailed results

In this section, we provide detailed per-class results for the highest scoring baselines on the PPS task, for both the

Class	PQ	SQ	RQ	PartPQ	PartSQ	PartRQ
road	98.3	98.4	99.9	98.3	98.4	99.9
sidewalk	80.4	86.6	92.7	80.4	86.6	92.7
building	90.3	91.0	99.3	90.3	91.0	99.3
wall	37.7	76.8	49.2	37.7	76.8	49.2
fence	44.0	76.0	57.9	44.0	76.0	57.9
pole	63.4	71.0	89.3	63.4	71.0	89.3
traffic light	58.5	73.3	79.8	58.5	73.3	79.8
traffic sign	74.5	80.9	92.1	74.5	80.9	92.1
vegetation	90.9	91.6	99.2	90.9	91.6	99.2
terrain	41.1	77.0	53.4	41.1	77.0	53.4
sky	88.8	92.6	95.9	88.8	92.6	95.9
person*†	60.8	79.3	76.6	44.1	57.8	76.2
rider*†	58.0	75.6	76.8	45.3	59.6	76.0
car*†	71.8	85.5	84.0	53.3	63.5	84.0
truck*†	57.1	89.0	64.2	36.4	56.7	64.2
bus*†	73.5	91.1	80.7	49.7	61.6	80.7
train*	67.9	85.9	79.1	67.9	85.9	79.1
motorcycle*	50.2	77.0	65.2	50.2	77.0	65.2
bicycle*	51.6	74.4	69.3	51.6	74.4	69.3
Things	61.3	82.2	74.5	49.8	67.1	74.3
Stuff	69.8	83.2	82.6	69.8	83.2	82.6
Parts	64.2	84.1	76.4	45.8	59.9	76.2
No Parts	67.0	82.3	80.2	67.0	82.3	80.2
All	66.2	82.8	79.2	61.4	76.4	79.1

Table 1. Detailed results for the highest scoring baseline on the Cityscapes Panoptic Parts dataset (HRNet-OCR & PolyTransform & BSANet [20, 12, 23]). We report both scores for PQ and PartPQ, for the panoptic segmentation and part-level panoptic segmentation task, respectively. * Indicates things classes; † indicates classes with parts.

Cityscapes Panoptic Part and Pascal Panoptic Part datasets. Similarly to the original Panoptic Quality [11], the Part-aware Panoptic Quality (PartPQ) can be split into two parts, the Segmentation Quality (PartSQ) and the Recognition Quality (PartRQ). Spefically

$$\text{PartPQ} = \text{PartSQ} \times \text{PartRQ}, \tag{1}$$

where

$$\text{PartSQ} = \frac{\sum_{(p,g)\in TP} \text{IOU}_\text{p}(p,g)}{|TP|}; \tag{2}$$

$$\text{PartRQ} = \frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}. \tag{3}$$

In Table 1, we show the per-class results of the highest scoring baseline on Cityscapes Panoptic Parts, for both panoptic segmentation and part-level panoptic segmentation. The Pascal Panoptic Parts results are provided in Table 2. A few things can be noted. 1) First of all, the performance for scene-level classes without parts is identical for PQ and PartPQ. This is as expected, as the task definition is the same for those classes, *i.e.*, if no part classes are defined, we just predict the scene-level segment. 2) Secondly, for the classes with parts, the scores for PartPQ are consistently lower than for PQ. This is also explainable, as these segments have to

Class	PQ	SQ	RQ	PartPQ	PartSQ	PartRQ
aeroplane*†	69.4	82.5	84.1	45.4	53.5	85.0
bag	24.4	73.6	33.1	24.4	73.6	33.1
bed	4.1	62.1	6.5	4.1	62.1	6.5
bedclothes	21.6	81.6	26.5	21.6	81.6	26.5
bench	7.5	61.6	12.2	7.5	61.6	12.2
bicycle*†	59.1	75.9	77.9	54.8	70.4	77.9
bird*†	68.2	84.4	80.8	44.6	54.6	81.7
boat*	50.8	77.7	65.3	50.8	77.7	65.3
book	26.9	70.7	38.1	26.9	70.7	38.1
bottle*†	56.2	85.6	65.7	42.7	64.8	65.8
building	47.9	78.6	60.9	47.9	78.6	60.9
bus*†	79.5	92.5	85.9	64.2	73.9	86.9
cabinet	26.7	76.5	34.9	26.7	76.5	34.9
car*†	69.0	86.7	79.5	45.5	56.9	80.1
cat*†	79.9	89.4	89.4	60.6	67.8	89.4
ceiling	48.3	80.1	60.4	48.3	80.1	60.4
chair*	38.3	77.5	49.4	38.3	77.5	49.4
cloth	9.3	71.9	12.9	9.3	71.9	12.9
computer	25.1	69.9	35.8	25.1	69.9	35.8
cow*†	59.0	84.4	69.9	45.6	65.2	70.0
cup	21.6	73.2	29.5	21.6	73.2	29.5
curtain	38.9	78.2	49.8	38.9	78.2	49.8
dog*†	75.1	87.2	86.1	56.3	65.3	86.3
door	15.4	72.4	21.2	15.4	72.4	21.2
fence	25.1	70.1	35.8	25.1	70.1	35.8
floor	51.7	82.9	62.4	51.7	82.9	62.4
flower	11.4	69.0	16.5	11.4	69.0	16.5
food	20.8	73.0	28.4	20.8	73.0	28.4
grass	61.2	84.0	72.8	61.2	84.0	72.8
ground	40.5	80.0	50.7	40.5	80.0	50.7
horse*†	62.9	80.1	78.5	51.3	65.0	78.9
keyboard	34.1	68.8	49.5	34.1	68.8	49.5
light	26.4	70.7	37.4	26.4	70.7	37.4
motorbike*†	67.5	81.2	83.2	63.7	76.6	83.2
mountain	39.3	76.8	51.2	39.3	76.8	51.2
mouse	16.3	74.5	21.9	16.3	74.5	21.9
person*†	65.5	80.8	81.0	46.6	57.5	81.1
plate	7.5	73.0	10.3	7.5	73.0	10.3
platform	35.1	82.7	42.4	35.1	82.7	42.4
pottedplant*†	45.5	77.8	58.5	38.9	66.5	58.5
road	44.9	85.9	52.3	44.9	85.9	52.3
rock	29.0	76.2	38.1	29.0	76.2	38.1
sheep*†	68.1	85.2	79.9	56.8	69.7	81.4
shelves	11.1	66.8	16.6	11.1	66.8	16.6
sidewalk	15.9	70.4	22.5	15.9	70.4	22.5
sign	24.9	79.6	31.3	24.9	79.6	31.3
sky	82.9	92.6	89.5	82.9	92.6	89.5
snow	53.8	81.1	66.3	53.8	81.1	66.3
sofa*	47.0	82.3	57.1	47.0	82.3	57.1
table*	35.3	73.0	48.4	35.3	73.0	48.4
track	50.2	71.4	70.3	50.2	71.4	70.3
train*	76.3	88.4	86.3	76.3	88.4	86.3
tree	63.1	82.2	76.8	63.1	82.2	76.8
truck	9.9	77.2	12.8	9.9	77.2	12.8
tvmonitor*†	65.3	86.2	75.8	57.6	76.0	75.8
wall	55.4	80.6	68.7	55.4	80.6	68.7
water	72.2	89.9	80.3	72.2	89.9	80.3
window	28.1	73.8	38.1	28.1	73.8	38.1
wood	9.8	74.5	13.2	9.8	74.5	13.2
Things	61.9	82.9	74.1	51.1	69.1	74.4
Stuff	31.7	75.8	40.5	31.7	75.8	40.5
Parts	66.0	84.0	78.4	51.6	65.6	78.8
No Parts	33.8	76.3	42.8	33.8	76.3	42.8
All	42.0	78.3	51.9	38.3	73.6	52.0

Table 2. Detailed results for the highest scoring baseline on the Pascal Panoptic Parts dataset (DeepLabv3-ResNeSt269 & DetectoRS & BSANet [3, 22, 17, 23]). We report both scores for PQ and PartPQ, for the panoptic segmentation and part-level panoptic segmentation task, respectively. * Indicates things classes; † indicates classes with parts.

Part class	Definition
Window	Windows, wind shields and other glass surfaces on vehicles.
Wheel	All wheels and tires under vehicles (excluding spare tires on the back of vehicles).
Light	Light source present on vehicles, including taxi sign.
License plate	License plate on front/back of vehicles.
Chassis	Part of vehicle body not belonging to above classes.
Unlabeled	Ambiguous or not clearly visible regions.

Table 3. *Vehicle* part classes for Cityscapes Panoptic Parts.

Part class	Definition
Torso	Core of human body, excluding limbs and head.
Head	Human head.
Arm	Arms, from shoulders to hands.
Leg	Legs, from hips to feet.
Unlabeled	Ambiguous or not clearly visible regions.

Table 4. *Human* part classes Cityscapes Panoptic Parts.

be segmented further into parts, and are evaluated on multi-class IOU instead of instance-level IOU. 3) Thirdly, note that, although we expect a change in PartSQ for scene-level classes with parts, we do not expect the PartRQ to change, as the *TP*, *FP* and *FN* are evaluated on scene-level segments, as in the original PQ. However, in both datasets, there are ground-truth segments that are labeled on instance-level, but not on part-level, due to ambiguity or because the instances are so small that parts could not be distinguished. In these rare cases, these segments are ignored during evaluation, causing the PartRQ to differ slightly from the RQ (*e.g.* for *person*, in Table 1).

3. Cityscapes Panoptic Parts annotations

3.1. Part definitions

The Cityscapes dataset [7] focuses on urban scene understanding and automated driving. Adhering to that direction, we choose to annotate three important *vehicle* classes, *i.e.* *car*, *truck*, *bus*, and all *human* classes, *i.e.* *person*, *rider*. *Vehicle* and *human* categories describe semantic classes with similar parts, thus we define the same semantic parts for each of the classes in these categories. The part classes are defined in Tables 3 and 4.

3.2. Manual labeling protocol

The annotators of Cityscapes Panoptic Parts were asked to start annotation from background to foreground objects, and to annotate each object with part labels in the order they appear in Tables 3 and 4. The regions to be annotated were extracted using the instance masks from the original Cityscapes dataset. This way, we maintain consistency with the original dataset. Very small instances, ambiguous regions, or indistinguishable parts are not annotated with part labels, so they stay annotated on scene-level only. Moreover, it is not necessary for object instances to contain all part classes. These aspects are taken into consideration by the PartPQ metric.

More details on the annotation procedure and the protocol can be found on https://github.com/tue-mps/panoptic_parts.

4. Qualitative examples

We provide more qualitative examples of the highest scoring baselines on CPP and PPP in Figure 1 and 2, respectively.

References

[1] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid Task Cascade for Instance Segmentation. In *CVPR*, 2019. 2

[2] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019. 2

[3] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking Atrous Convolution for Semantic Image Segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 2, 3, 6

[4] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-Decoder with Atrous Separable Convolution for Semantic Image Segmentation. In *ECCV*, 2018. 1, 2

[5] Xianjie Chen, Roozbeh Mottaghi, Xiaobai Liu, Sanja Fidler, Raquel Urtasun, and Alan Yuille. Detect what you can: Detecting and representing objects using holistic models and body parts. In *CVPR*, 2014. 2

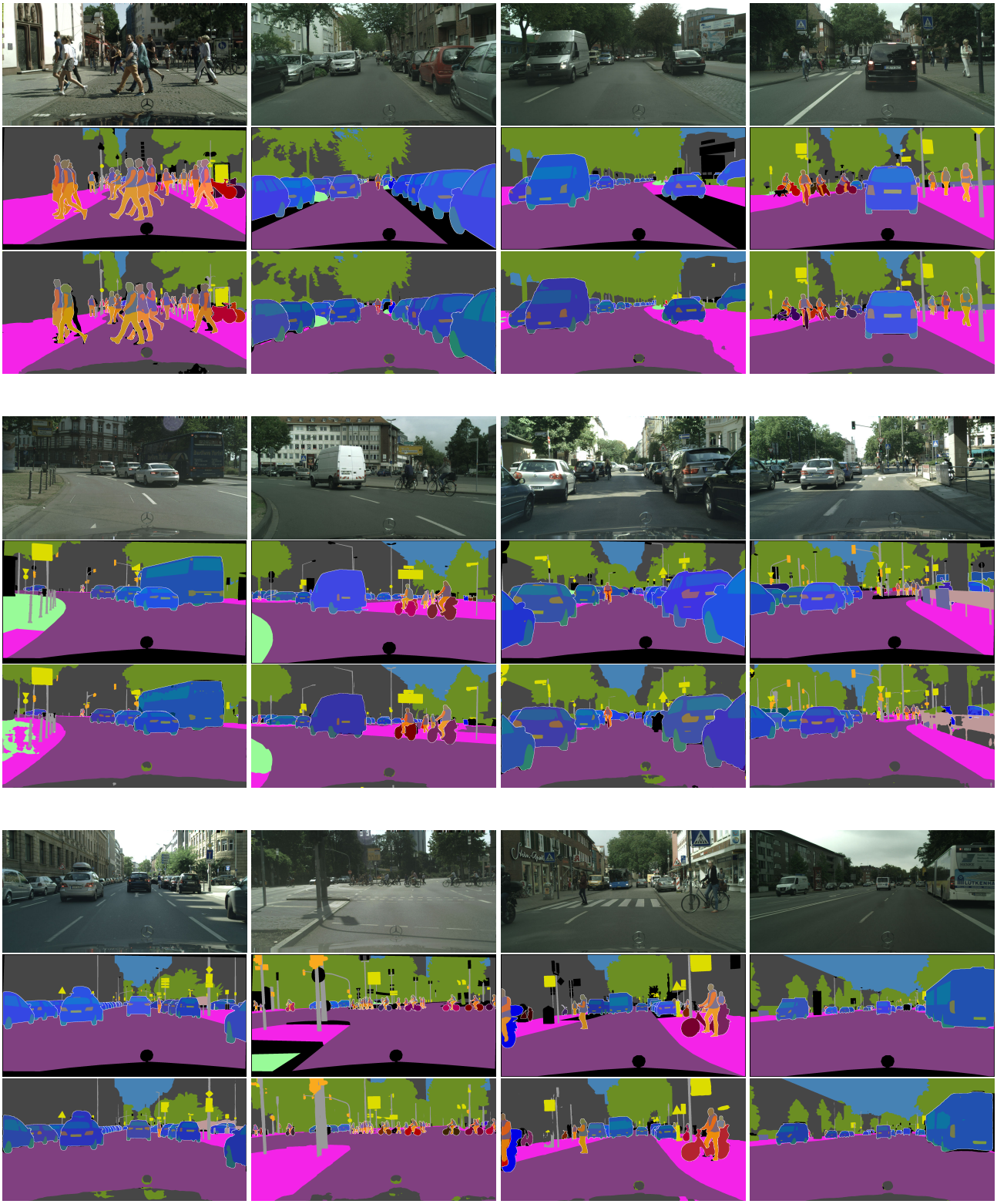

Figure 1. Qualitative examples for the highest-scoring part-aware panoptic segmentation baseline on Cityscapes Panoptic Part (HRNet-OCR & PolyTransform & BSANet [20, 12, 23]). On each of the three rows, we show the input images (top), ground truth (middle) and predictions (bottom).

Figure 2. Qualitative examples for the highest-scoring part-aware panoptic segmentation baseline on PASCAL Panoptic Parts (DeepLabv3-ResNeSt269 & DetectoRS & BSANet [3, 22, 17, 23]). On each of the three rows, we show the input images (top), ground truth (middle) and predictions (bottom).

[6] Francois Chollet. Xception: Deep Learning With Depthwise Separable Convolutions. In *CVPR*, 2017. 1

[7] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The Cityscapes Dataset for Semantic Urban Scene Understanding. In *CVPR*, 2016. 1, 2, 4

[8] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The Pascal Visual Object Classes (VOC) Challenge. *International Journal of Computer Vision*, 88(2):303–338, June 2010. 2

[9] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask R-CNN. In *ICCV*, 2017. 1, 2

[10] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun.

Deep Residual Learning for Image Recognition. In *CVPR*, 2016. 1, 2

[11] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic Segmentation. In *CVPR*, 2019. 1, 2, 3

[12] Justin Liang, Namdar Homayounfar, Wei-Chiu Ma, Yuwen Xiong, Rui Hu, and Raquel Urtasun. PolyTransform: Deep Polygon Transformer for Instance Segmentation. In *CVPR*, 2020. 1, 3, 5

[13] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: Common Objects in Context. In *ECCV*, 2014. 1

[14] Rohit Mohan and Abhinav Valada. EfficientPS: Efficient panoptic segmentation. *International Journal of Computer Vision*, 2021. 1

[15] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The Role of Context for Object Detection and Semantic Segmentation in the Wild. In *CVPR*, 2014. 2

[16] OpenMMLab. *MMSegmentation*. https://github.com/open-mmlab/mmsegmentation (accessed October 5, 2020). 2

[17] Siyuan Qiao, Liang-Chieh Chen, and Alan Yuille. DetectoRS: Detecting Objects with Recursive Feature Pyramid and Switchable Atrous Convolution. *arXiv preprint arXiv:2006.02334*, 2020. 2, 3, 6

[18] Jingdong Wang, Ke Sun, Tianheng Cheng, Borui Jiang, Chaorui Deng, Yang Zhao, Dong Liu, Yadong Mu, Mingkui Tan, Xinggang Wang, et al. Deep High-Resolution Representation Learning for Visual Recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2020. 1

[19] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. UPSNet: A unified panoptic segmentation network. In *CVPR*, 2019. 1

[20] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-Contextual Representations for Semantic Segmentation. In *ECCV*, 2020. 1, 3, 5

[21] Hang Zhang. *PyTorch-Encoding*. https://github.com/zhanghang1989/PyTorch-Encoding (accessed October 14, 2020). 2

[22] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola. ResNeSt: Split-Attention Networks. *arXiv preprint arXiv:2004.08955*, 2020. 2, 3, 6

[23] Yifan Zhao, Jia Li, Yu Zhang, and Yonghong Tian. Multi-Class Part Parsing With Joint Boundary-Semantic Awareness. In *ICCV*, 2019. 2, 3, 5, 6