Appendix

Funding in direct support of this work: Google LLC. There is no additional revenues related to this work.

A Broader Impacts

We propose a single framework for multi-dataset multi-task segmentation, which especially benefits memory-limited application scenarios, *e.g.*, our work can make a single segmentation model that meets all kinds of segmentation needs on memory-limited mobile devices (semantic segmentation for photo editing, instance segmentation for object detection and image search, *etc.*), instead of using separate models for various segmentation tasks.

Moreover, our approach improves segmentation performance from a data-centric view: We leverage segmentation data from multiple sources to improve the overall segmentation performance, especially on datasets of smaller scales, *e.g.*, ADE20k semantic. We also transfer knowledge from other sources to enable weakly-supervised segmentation. This is helpful on the scenario where segmentation data are very hard to collect, we can cotrain on similar data to help improve the performance, or even train on data with weaker supervision. *e.g.*, if we want to train a segmentation model on rare wild animals, we can co-train a model on a small set of these kinds of animals, and a larger dataset of more common animals.

As for bias and fairness, our model learns from the training data, so it may inherit the bias and fairness issues existing in the training data. Nonetheless, our model trains on multiple sources, and the fairer datasets may compensate the bias and fairness issues in other datasets.

B More about our newly labeled Objects365 instance segmentation evaluation dataset

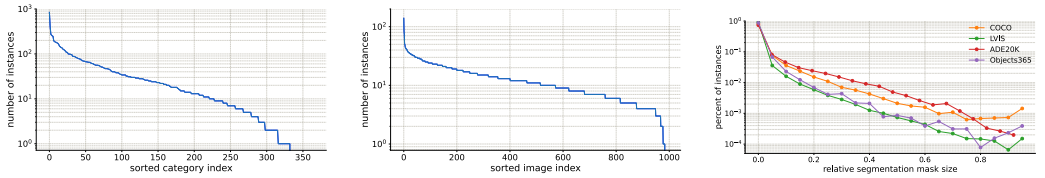
We randomly sampled 1,000 images from the Objects365 V2 validation set. Different from the "federated" LVIS dataset [18] (not all categories in each image are completely labeled), the bounding boxes in Objects365 are completely labeled for all categories and we follow this complete annotation in our instance masks. For each bounding box annotation, we generate a foreground/background annotation question for the raters. In the question, raters are asked to paint a semi-transparent instance mask on the original image, with the groundtruth bbox shown on the image as a guide and the groundtruth category displayed on the side. We do not crop the original image to the bbox region, but show the entire image to provide more context information, which is especially helpful for small objects. If the boundary is too blurry or too dark to annotate the instance mask, the raters can skip the question.

The annotation tool is a free painting tool, which allows the raters to freely draw the instance mask. We ask the raters to try to draw within the bbox, but if the object is obviously exceeding the bbox, then they can draw outside the bbox. The size of the stroke is adjustable. Raters can zoom-in/out, draw straight lines, and fill the inside of a plotted boundary with a single click (which is quite useful for instance segmentation). Unlike polygons in COCO *etc.*, the annotation tool allows us to save higher-resolution binary masks.

We inserted a total of 13,372 questions. Following the common practice in instance segmentation, we did not label the crowd instances, as they are skipped in both training and inference. In the end, we obtained 12,836 valid instance mask annotations. It took a total of 800.59 rater hours. On average, raters spent 3.74 minutes on each valid mask.

In Fig. 5, we show some statistics about our Objects365 instance segmentation dataset. The number of masks per category follows a relatively long-tailed distribution (Fig. 5(a)). The number of masks per image reveals that a large number of images have many instance annotations (Fig. 5(b)). And the statistics of the relative mask size is close to that of the LVIS datasets (Fig. 5(c)).

As visualized in Fig. 6, the mask annotations are very high-quality. The raters carefully handled small objects, thin structures, complicated shapes, occluded objects, *etc*.



(a) Number of annotated instance masks per category. It follows a relatively long tail distribution.

(b) Number of annotated instance masks per image. More than 500 images have more than 10 instances.

(c) Relative segmentation mask size (square root of the ratio of mask area to image area), compared with COCO, LVIS, and ADE20K. Our statistics are similar to LVIS.

Figure 5: **Dataset statistics for our Objects365 instance segmentation evaluation set.**

Figure 6: **Visualization of our Objects365 instance segmentation dataset.** We store binary masks, which has higher mask quality compared with polygons. The quality on small objects are good (first two images). Complex structures are also labeled delicately, *e.g.*, the camels (1st image on 2nd row) and the decoration on the cake (1st image on 3rd row). For occlusions, raters carefully avoided the occluded regions as they are not a part of the object to annotate, *e.g.*, the fruits in the basket are not included in the basket masks (last image on 1st row).

For the objects which do not have an instance mask since raters skipped it, it's possible that the model still predicts an instance mask for that object. We include the skipped annotations to make it optional to ignore them during evaluation, so as to avoid treating such cases as false positive.

We release this dataset. We believe this effort can provide the research community a good benchmark for instance segmentation evaluation, *e.g.*, for evaluating weakly-supervised instance segmentation learned from Objects365 bounding boxes.

C Open-vocabulary semantic segmentation on PASCAL VOC

We evaluate DaTaSeg on the PASCAL VOC 2012 semantic segmentation dataset [14], which includes 20 object classes and a background class. We directly transfer our trained model to its validation set with 1,449 images without finetuning. We follow the setting in MaskCLIP+ [76] and ignore the background class during evaluation.

Method	Backbone	mIoU	mIoU-unseen	mIoU-seen
SPNet [65]	DeepLabv2-ResNet101	56.9	0.0	75.8
SPNet-C [65]		63.2	15.6	78.0
ZS3Net [3]		61.6	17.7	77.3
CaGNet [17]		65.5	26.6	78.4
STRICT [51]		70.9	35.6	82.7
MaskCLIP+ [76]		88.1	86.1	88.8
Fully-supervised		88.2	87.0	88.6
DaTaSeg	ResNet-50	89.4	87.7	89.9
	ViTDet-B	92.8	90.9	93.4
	ViTDet-L	94.7	94.3	94.9

Table 6: **Open-vocabulary semantic segmentation on the PASCAL VOC 2012 dataset.** DaTaSeg outperforms all other zero-shot/open-vocabulary methods, and even the fully-supervised baseline. Comparison methods are trained on 15 seen classes of PASCAL VOC and only 5 classes are unseen (mIoU-unseen). In contrast, DaTaSeg has never seen the PASCAL VOC dataset during training (it's trained on several other datasets). We attribute DaTaSeg's performance improvement to our multi-dataset multi-task training. Following [76], the background class is ignored. The numbers of the comparison methods are from [76]. SPNet-C stands for SPNet with calibration.

	Fully-supervised		Weakly-supervised transfer	
Dataset sampling ratio	ADE semantic mIoU	COCO panoptic PQ	ADE semantic → panoptic PQ	O365 box → instance mask AP
1:4:4	48.1	49.0	29.8	14.3
1:2:2	46.8	48.6	29.1	12.8
1:1:1	45.3	48.0	28.4	13.7

Table 7: **Ablation study on dataset sampling ratio (ADE:COCO:O365).** Evaluated on a ResNet50 backbone. Results show that our adopted sampling ratio (1:4:4) is better than the other sampling ratios.

We compare with other zero-shot and open-vocabulary segmentation methods in Table 6, which are trained on 15 seen classes with 5 classes held out during training as unseen classes (potted plant, sheep, sofa, train, and TV monitor). DaTaSeg has never seen the PASCAL VOC dataset during training, though it is trained on other segmentation datasets (COCO panoptic, ADE semantic, and Objects365 detection). DaTaSeg with a ResNet-50 backbone outperforms all these methods with DeepLabv2-ResNet101 backbones. Surprisingly, DaTaSeg even outperforms the fully-supervised counterpart. We attribute DaTaSeg's performance improvement to our multi-dataset multi-task training. We note this is not an apple-to-apple comparison, since the training sources are different. The main purpose of this comparison is to show how our "data-centric" DaTaSeg is positioned against zero-shot/open-vocabulary segmentation methods on the PASCAL VOC dataset, using a direct transfer manner.

D Ablation studies

We perform more ablation studies in this section.

Ablation study on dataset sampling ratio: Our dataset sampling strategy (Sec. 3.5) avoids the imbalance of dataset sampling by specifying the per-dataset sampling ratio. The proportion of training samples coming from each dataset in the whole training process is determined by that sampling ratio. In our main results, the sampling ratio is 1:4:4 for ADE:COCO:O365. We ablate the dataset sampling ratio in Table 7. Results show that our adopted sampling ratio is better than the other sampling ratios.

Ablation study on the portion of Objects365 training data: Readers may have the question about the trade-offs between the quality and quantity of the training samples when we introduce the weakly-supervised training data (Objects365). In Table 2, we show the results with and without cotraining on Objects365. The performance is generally better when cotraining on Objects365. In addition, we experiment with cotraining on different portions of the dataset: we cotrain DaTaSeg on the full COCO panoptic and ADE semantic datasets, and 10% / 25% / 50% / 100% of O365 training data. We show the results in Table 8. Results show that when increasing the number of O365 weakly-supervised training samples, O365 performance increases, COCO panoptic performance is not affected, and ADE20k performance slightly decreases. Overall, the gains are larger than the losses.

Portion of O365 training data	Fully-supervised		Weakly-supervised transfer	
	ADE semantic mIoU	COCO panoptic PQ	ADE semantic → panoptic PQ	O365 box → instance mask AP
10%	48.5	48.7	30.0	10.4
25%	48.6	48.3	30.6	12.0
50%	47.7	48.5	29.8	13.1
100%	47.2	48.7	29.4	14.5

Table 8: **Ablation study on the portion of O365 training data.** Evaluated on a ResNet50 backbone. When increasing the number of O365 weakly-supervised samples, O365 performance increases, COCO panoptic performance is not affected, and ADE20k performance slightly decreases.

λ_{proj}	μ_{proj}	ADE semantic	COCO panoptic	ADE semantic → panoptic	O365 box → instance
5.0	1.0	45.5	48.3	28.4	12.5
2.0	0.5	47.2	48.7	29.4	14.5

Table 9: **Ablation study on the projection loss $\mathcal{L}_{proj}$ weights.** Evaluated on a ResNet50 backbone. The results indicate that the 2nd setting (our final setting) has a better performance on all datasets.

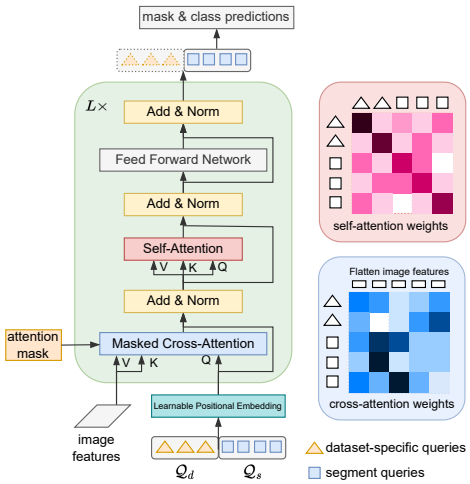


Figure 7: **Dataset-specific queries in the transformer decoder.** We use different dataset-specific queries $\mathcal{Q}_d$ for each dataset d, which are treated the same as the segment queries $\mathcal{Q}_s$, and involved in both cross-attention and self-attention. All dataset-specific queries are discarded before predicting masks and classes.

Ablation study on projection loss $\mathcal{L}_{proj}$ weights: Our settings of λ and μ (Table 13) closely follow Maskformer series [10, 8]. We add the the bounding-box projection loss $\mathcal{L}_{proj}$ for Objects365. We run an ablation study on the weights (λ_{proj} and μ_{proj}) for the projection loss in Table 9. The results indicate that the 2nd setting (our final setting) has a better performance on all datasets. Finding the optimal weights for multiple losses is an interesting yet challenging problem, and we leave it for future work.

E Adding dataset-specific modules hurts performance

In this section, we first introduce our designed dataset-specific modules and then show more ablation studies on these modules.

As prior works have pointed out [33, 61], there are many inconsistencies across segmentation datasets, including labels having their own particular definition, being annotated using different protocols, *etc*. We explore if adding dataset-specific modules can improve overall performance by designing the following modules.

Dataset-specific queries: In our architecture design introduced in Sec. 3.4, an input image will not get any dataset-specific information until the text embedding classifier, which is almost at the final stage of the network. We thus propose to add a few dataset-specific queries $\mathcal{Q}_d$ as an auxiliary input to the beginning of transformer decoder. Each training dataset d has its own $\mathcal{Q}_d$. The dataset-specific

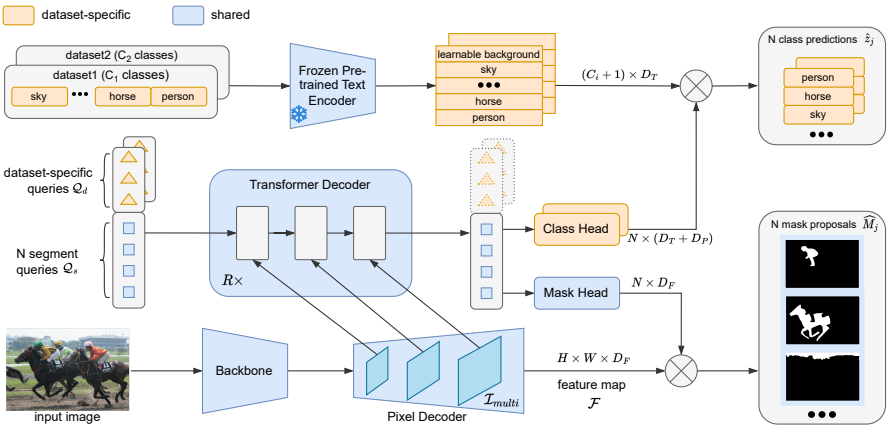


Figure 8: **Overview of incorporating all our designed dataset-specific modules into our universal multi-dataset multi-task segmentation model (DaTaSeg)**. We design a set of *dataset-specific queries* fed into the transformer decoder, *dataset-specific classification head*, and *dataset-specific background classifier*. They are highlighted in orange. All these modules are learnable. Different datasets have separate parameters for these modules.

queries are treated the same as regular segment queries $\mathcal{Q}_s$. The only difference is that we do not make mask and class predictions for the dataset-specific queries. Fig. 7 shows the details. In the cross-attention layer, $\mathcal{Q}_d$ cross attends to the image features — the cross-attention weight is:

$$\mathcal{W}_{\text{cross\_atten}} = \text{softmax}(\mathcal{A} + W_q\left[\mathcal{Q}_d\ \mathcal{Q}_s\right] \cdot \mathcal{I}_{multi}^T W_k^T), \tag{6}$$

where $\mathcal{I}_{multi}$ are the multi-scale image features in Sec. 3.4. W_q, W_k denote the linear projection weights; $\mathcal{A}$ denotes the attention mask.

In the self-attention layers, there are interactions for all pairs of $\mathcal{Q}_d$ and $\mathcal{Q}_s$, allowing the segment queries to be aware of the dataset-specific information at the beginning of transformer decoder. This can be shown by the self-attention weight calculation:

$$\begin{aligned}\mathcal{W}_{\text{self\_atten}} &= \text{softmax}(W_q \begin{bmatrix}\mathcal{Q}_d \\ \mathcal{Q}_s\end{bmatrix} \begin{bmatrix}\mathcal{Q}_d^T & \mathcal{Q}_s^T\end{bmatrix} W_k^T) \\ &= \text{softmax}(W_q \begin{bmatrix}\mathcal{Q}_d\mathcal{Q}_d^T & \mathcal{Q}_d\mathcal{Q}_s^T \\ \mathcal{Q}_s\mathcal{Q}_d^T & \mathcal{Q}_s\mathcal{Q}_s^T\end{bmatrix} W_k^T).\end{aligned} \tag{7}$$

Dataset-specific classification head and background classifier: We observe that dataset inconsistency tends to primarily be a classification issue, rather than localization issue. Hence, for all datasets we use the same set of parameters for localization (backbone, pixel decoder, transformer decoder, mask embedding head) to enhance knowledge sharing. For classification, we design some dataset-specific components: specifically, we use a dataset-specific MLP class embedding head as well as a learnable dataset-specific background classifier, since the definitions of background vary from dataset to dataset.

Overall dataset-specific architecture: The overall architecture equipped with all our dataset-specific modules is shown in Fig. 8. We design the dataset-specific modules to be light-weight, which allows us to save on memory costs. With this design of a mixture of a mostly shared network and light-weight dataset-specific modules, the model has the freedom to choose whether to leverage dataset-specific information, or to use the shared knowledge across datasets/tasks. However, one significant disadvantage of adding dataset-specific modules is that it's hard to decide which set of dataset-specific parameters to use when directly transferring to other datasets (Sec. 4.5), and thus it *hurts the open-vocabulary capability*.

Ablation studies on dataset-specific modules: We ablate on the number of dataset-specific queries $\mathcal{Q}_d$ in Table 10. It shows that removing the dataset-specific queries achieves the best results on COCO panoptic and Objects365 by a large margin. It also achieves reasonable performance on other datasets. Using 30 queries hurts the performance on COCO panoptic and ADE semantic.

# of dataset-specific queries	Fully-supervised		Weakly-supervised transfer	
	ADE semantic mIoU	COCO panoptic PQ	ADE semantic → panoptic PQ	O365 box → instance mask AP
0	47.8	48.1	28.6	13.1
10	48.0	46.2	28.0	11.2
20	48.6	46.6	28.5	11.7
30	47.1	45.8	27.4	11.7

Table 10: **Ablation study on the number of dataset-specific queries $\mathcal{Q}_d$.** Evaluated on a ResNet50 backbone. Without dataset-specific queries, the performance on COCO panoptic and Objects365 instance segmentation is much better. Using a large number of dataset-specific queries, *e.g.*, 30, hurts performance on many datasets, since it weakens knowledge sharing among datasets & tasks.

		w/ dataset-specific classification	w/o dataset-specific classification
ADE semantic	mIoU	47.9	48.1 (+0.2)
COCO panoptic	PQ	48.5	49.0 (+0.5)
ADE panoptic†	PQ	28.7	29.8 (+1.1)
O365 instance†	AP	12.9	14.3 (+1.4)

Table 11: **Adding dataset-specific classification modules hurt performance on almost all datasets,** especially the weakly-supervised transfer results†. The dataset-specific classification modules include the classification embedding head and the learnable background classifier. Evaluated on a ResNet50 backbone, without other dataset-specific modules.

In Table 11, we compare using and not using dataset-specific classification modules (the classification embedding head and background classifier). Results indicate that removing dataset-specific classification improves performance on all datasets & tasks, especially the weakly-supervised tasks. We suspect it's because weakly-supervised tasks rely more on knowledge sharing.

System-level comparison between with and without dataset-specific modules: In the main paper, due to space limit, we only show results of adding all dataset-specific modules on a ResNet50 backbone. In Table 12, we show the same comparison on more backbones, *i.e.*, ResNet50, ViTDet-B, and ViTDet-L. The observation is consistent across all backbones: adding dataset-specific modules hurts performance on all datasets. The difference is more significant on backbones of smaller scales, and on COCO panoptic and Objects365 instance segmentation datasets.

F Comparison with related contemporary work

We discuss the differences between DaTaSeg and some contemporary related work as below.

Comparison with Segment Anything [30]: SAM [30] is a related work on multi-task universal segmentation model. There are several major differences between DaTaSeg and SAM. 1) The main use-case for SAM is prompt-based segmentation, i.e., the user inputs a point or box, and the model segments the object referred by the prompt. While we aim to segment all objects and stuff in a bottom-up way. 2) Besides, SAM built their own large-scale dataset with more than 1 billion masks (SA-1B) by designing a data engine consisting of model-assisted manual annotation and semi-automatic stage, which is very costly; while we only utilize multiple publicly available datasets and train a joint segmentation model to improve the performance. Hence, DaTaSeg and SAM's results are not comparable in terms of the training data. 3) Except for the explicitly trained text-to-mask task, SAM's outputs are class-agnostic binary masks (the SA-1B datasets are also class-agnostic), while our model outputs the class predictions together with the mask predictions.

Comparison with X-Decoder [80]: X-Decoder is a related work on multi-task segmentation. There are multiple differences between our work and X-Decoder, which makes it hard to have a strict comparison. We detail the differences below: 1) *Task and Focus*: The pre-training in X-Decoder includes panoptic segmentation, referring segmentation, and image-text pairs (image-text retrieval and image captioning), which has a focus on vision-language tasks. These tasks are very different from ours: we cotrain on three mainstream segmentation tasks (panoptic/semantic/instance) and exclusively focus on segmentation. 2) *Training paradigm*: We directly cotrain on multiple datasets using a shared set of parameters (single model), while the ADE and COCO results in X-Decoder Table 1 is task-specific transfer. That is, X-Decoder first pretrains on large-scale data and then fine-tunes on each target dataset using different sets of fine-tuned parameters. 3) *Training data*:

Backbone	Model	Fully-Supervised		Weakly-Supervised Transfer	
		ADE semantic mIoU	COCO panoptic PQ	ADE semantic → panoptic PQ	O365 box → instance mask AP
ResNet50	+D-S modules	48.1	46.0	26.9	10.9
	DaTaSeg	48.1 (+0.0)	49.0 (+3.0)	29.8 (+2.9)	14.3 (+3.4)
ViTDet-B	+D-S modules	51.0	50.8	31.0	13.1
	DaTaSeg	51.4 (+0.4)	52.8 (+2.0)	32.9 (+1.9)	16.1 (+3.0)
ViTDet-L	+D-S modules	53.9	52.3	33.5	13.7
	DaTaSeg	54.0 (+0.1)	53.5 (+1.2)	33.4 (-0.1)	16.4 (+2.7)

Table 12: **Comparing DaTaSeg with the alternative architecture adding all dataset-specific modules on various backbones.** Results show that removing the dataset-specific modules improve performance on all datasets and all backbones. The gains are most significant on Objects365 instance and COCO panoptic segmentation (with dataset-specific modules, it cannot outperform the separately trained baseline on COCO panoptic).

X-Decoder pretrains on COCO panoptic and referring segmentation and 4M image-text pairs with a long schedule, and we quote from their paper: "all the pre-trained models are trained with 50 epochs of COCO data and roughly 45 epochs of 10 million image-text pairs". Afterwards, they fine-tune the model on COCO and ADE20k. By contrast, we only train on COCO panoptic, ADE20k semantic, and Objects365 v2 detection with a total of 1.8M of training images, which is much fewer than X-Decoder. We also do not train on large amounts of text data. 4) *Model architecture*: X-Decoder adopts Mask2Former. Our model architecture is different as explained in Sec. G. More importantly, we don't have an online text encoder in our model architecture.

G Architecture change from Mask2Former

Our network architecture is based on Mask2Former [8]. However, due to hardware difference (TPU vs. GPU), we need to make modifications to Mask2Former architecture to run on our hardware. Unlike GPUs, TPUs require a static computation graph with *fixed-shape* data (otherwise, it recompiles for each computation graph change, and causes significant slowdown). Therefore, we drop all the TPU-unfriendly operations in Mask2Former. In particular, we change the *multi-scale deformable attention transformer [79]* pixel decoder to a plain FPN [43]. The performance comparison is shown in Table 4(e) in the Mask2Former paper (51.9 PQ of deformable-attention vs. 50.7 PQ of FPN on COCO panoptic). Also, we do not use the *sampled point loss* [9] in either the matching loss or the training loss calculation, and use the vanilla mask loss. The performance comparison is shown in Table 5 in the Mask2Former paper (51.9 PQ of point loss vs. 50.3 PQ of mask loss). Besides, we *do not use varying input size* during evaluation and use a fixed input size (first resize then pad).

To improve the training speed, we *modify the deep supervision* in the masked attention transformer decoder: for every decoder layer, we use the same matching indices computed from the outputs of the last decoder layer, but use a different prediction head. We *do not use the mask predictions from the initial queries* as attention masks, since the predictions are not made based on the input image.

H Additional implementation details

In addition to the implementation details described in Sec. 4, we provide more datails here.

Preprocessing. In order to do 1:1 Hungarian matching, we preprocess the groundtruth into the same format as the predictions described in Sec. 3.2. In **panoptic segmentation**, we transfer the groundtruth into a set of binary masks with class labels. For "thing" categories, we group the pixels belonging to each instance into a separate groundtruth mask; for "stuff" categories, we group all pixels in each stuff category as the groundtruth mask. Similarly, in **semantic segmentation**, we group all pixels in each category as the groundtruth mask. For **instance segmentation**, since we are using bounding box weak supervision, we do not need this preprocessing step. For all mask proposal groundtruth, we downsample it by 4 times, to save memory cost during training.

Training settings. We randomly scale the input image in the range of [0.1, 2.0] and then pad or crop it to 1024×1024. For ADE20k dataset, since the image size is smaller than other datasets, we use a scaling range of [0.5, 2.0]. We use the AdamW optimizer [46] with a weight decay of 0.05. We

	μ_{ce}	μ_{focal}	μ_{dice}	μ_{proj}	λ_{ce}	λ_{focal}	λ_{dice}	λ_{proj}
ADE semantic	1	20	5	0	1	20	5	0
COCO panoptic	1	0	1	0	1	20	5	0
O365 instance (box GT)	1	0	0	0.5	1	0	0	2

Table 13: **The weights we use to compute the matching cost and total loss (Eqn. 4,5 in the main paper) for all training datasets.** μ's are for the matching cost and λ's are for the total training loss.

clip the gradients with a max norm of 0.1. The weight for the background class is set to 0.05 in $\mathcal{L}_{ce}$. The matching cost and loss weight settings for Eqn. 4,5 in the main paper are shown in Table 13. We use a dataset sampling ratio of 1:4:4 for ADE semantic, COCO panoptic, and Objects365 detection. We adopt a different learning rate multiplier for each dataset: We multiply the learning rate on ADE semantic, COCO panoptic, and Objects365 detection by 3, 5, 2, respectively. Since Objects365 detection dataset has a large vocabulary with imbalanced distribution, we apply repeat factor sampling with a frequency threshold $t = 0.01$ [18]. On ResNet50 backbones, we use a batch size of 384 and train 500k iterations, with a learning rate of 3e-5. We adopt the step learning rate schedule: We multiply the learning rate by $0.1\times$ at the 0.9 and 0.95 fractions of the total training iterations. On the ViTDet-B backbones, we train 600k iterations with a learning rate of 6e-5. On ViTDet-L, we use a batch size of 256 and train 540.5k iterations with a learning rate of 4e-5. For ResNet50, we train on 64 TPU v4 chips; for ViTDet backbones, we train on 128 TPU v4 chips. All evaluations are conducted on 4 V100 GPUs with a batch size of 8.

Postprocessing. We first apply the `MERGE` operation described in Sec. 3.2. We follow the postprocessing operations in [8] with some modifications: In panoptic segmentation, we use a confidence threshold of 0.85 to filter mask proposals for COCO panoptic, and set the threshold to 0.8 for ADE panoptic. For panoptic and instance segmentation, we filter out final segments whose area is smaller than 4. For instance segmentation, we return a maximum of 100 instances per image with a score threshold of 0.0. For all tasks, the final scores are the production of classification scores and localization scores (obtained from binary mask classification).

I Limitations

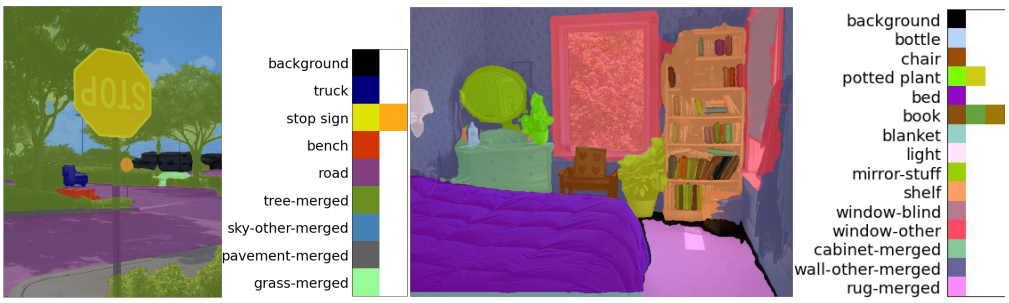
Our proposed method is not without limitations. Since we are the first work to explore universal multi-dataset multi-task segmentation models, we do not introduce the complexity for an efficient framework. There are several ways to improve the model efficiency, *e.g.*, calculating the mask loss on sampled points only [8], while we do not deploy this in our framework, since it's not our key contribution. Moreover, as shown in the experiment section (Table 1), our framework is orthogonal to the detailed network architectures as long as the network is able to output our universal segmentation representation.

For weakly-supervised panoptic segmentation on ADE20k panoptic, we notice that sometimes multiple instances of the same category are not separated in the prediction. In the weakly-supervised instance segmentation results on Objects365, there are some visual artifacts in the mask prediction. We believe using certain data augmentation techniques, *e.g.*, MixUp [73] or Copy-Paste [15], may farther enhance knowledge transfer between datasets of different tasks, and thus may help mitigate this issue. We leave it for future work to improve on these limitations.

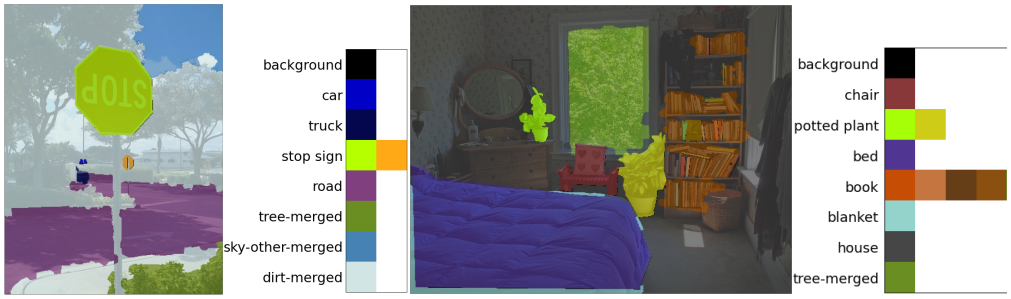
J Additional qualitative results

We notice sometimes our predictions do not match the groundtruth, but it does not necessarily mean the predictions are not good. Fig. 9 shows two examples on COCO panoptic. On the left column, DaTaSeg does a better job in segmenting 'sky' and 'tree-merged' than the groundtruth. The classification of 'pavement-merged' is also better than 'dirt-merged, which is probably due to *language ambiguity*: the British meaning of 'pavement' is sidewalk, while the annotator may be more used to the North American usage of 'pavement'. On the right column, DaTaSeg predicts 'window-other', while the groundtruth is 'tree-merged' for the scene through the window, and both make sense due to *label ambiguity*. In addition, DaTaSeg is able to predict the 'mirror-stuff', 'shelf', 'light', 'bottle', and more 'book's, which are missing from the groundtruth.

We show qualitative results for the direct transfer experiments in Fig. 10,11,12,13, on PASCAL Context 59 (PC-59), PASCAL Context 459 (PC-459), COCO semantic, and Cityscapes panoptic



(a) DaTaSeg predictions



(b) Groundtruths

Figure 9: **Examples on COCO panoptic showing that sometimes DaTaSeg's predictions do not match the groundtruth, but it does not necessarily mean they are wrong.** On the left, DaTaSeg's prediction is 'pavement-merged', while the groundtruth is 'dirt-merged' (Probably because the British meaning of 'pavement' is sidewalk, while the annotator is more used to the North American meaning of 'pavement'). Ours also segments 'tree' and 'sky' better. On the right, the definition for the scene through the window is ambiguous: 'window-other' (ours) v.s. 'tree-merged' (GT). The groundtruth misses to label several objects, while DaTaSeg recognizes more objects, *e.g.*, 'mirror-stuff', 'light', 'shelf', 'bottle'. Legends for 'book' are truncated due to space limit.

datasets, respectively. These results serve as a supplementary material for Table 3 in the main paper. They show DaTaSeg directly transfer to other segmentation datasets unseen during training with high quality both in localization and classification. DaTaSeg performs well on large-vocabulary segmentation (PC-459), and handles hard cases well (thin structures, small objects, complicated scenes, occlusions, *etc*.).

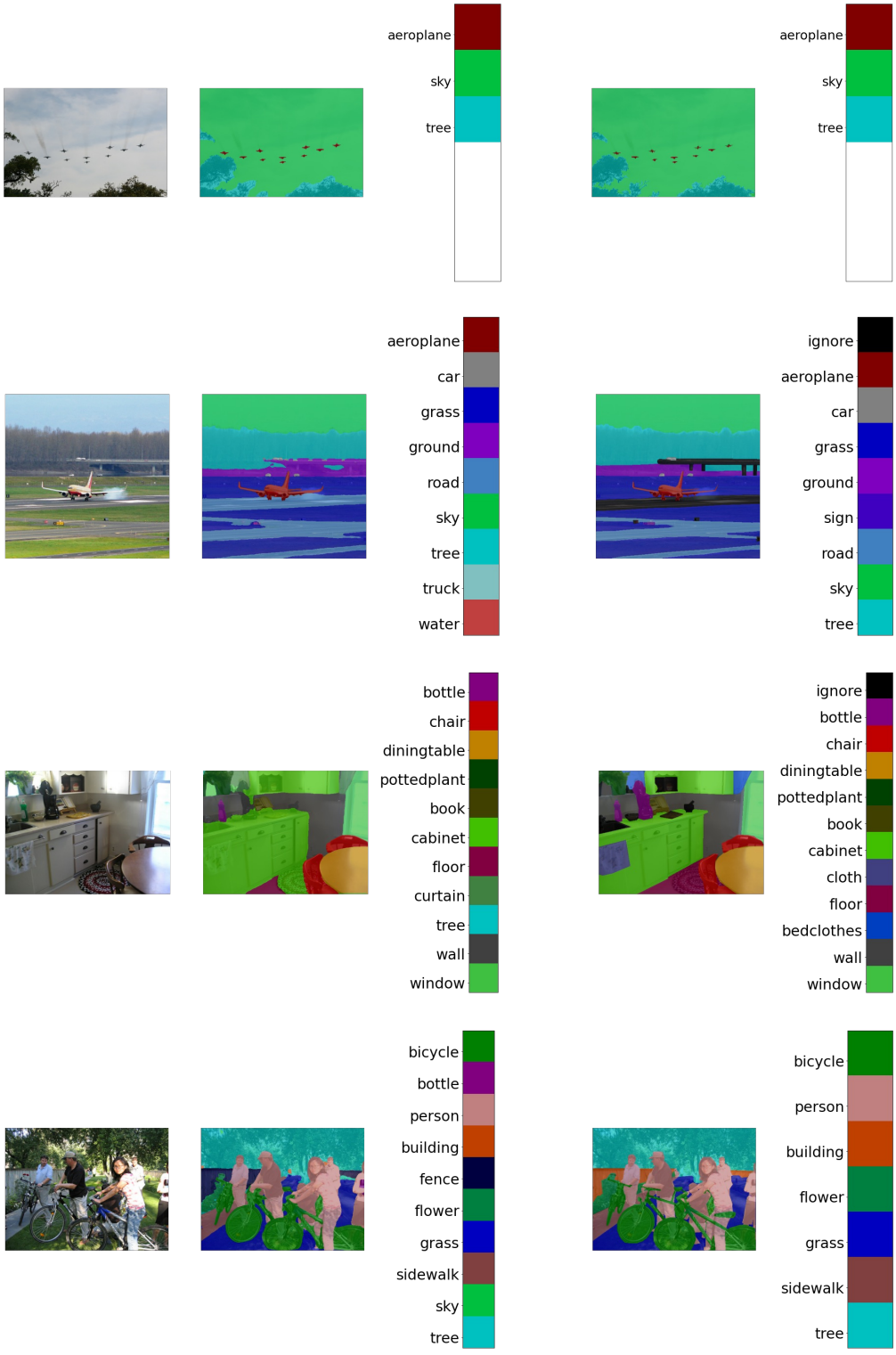


Figure 10: **Qualitative results of DaTaSeg directly transferring to PASCAL Context semantic dataset with 59 categories (PC-59).** The results demonstrate DaTaSeg has good generalization ability. The top row shows DaTaSeg is able to segment small objects (aeroplane), and the last row indicates DaTaSeg segments fine structures (bicycles) well. For each row, the left is the input image, the middle is our prediction, and the right is groundtruth. With a ViTDet-L backbone.

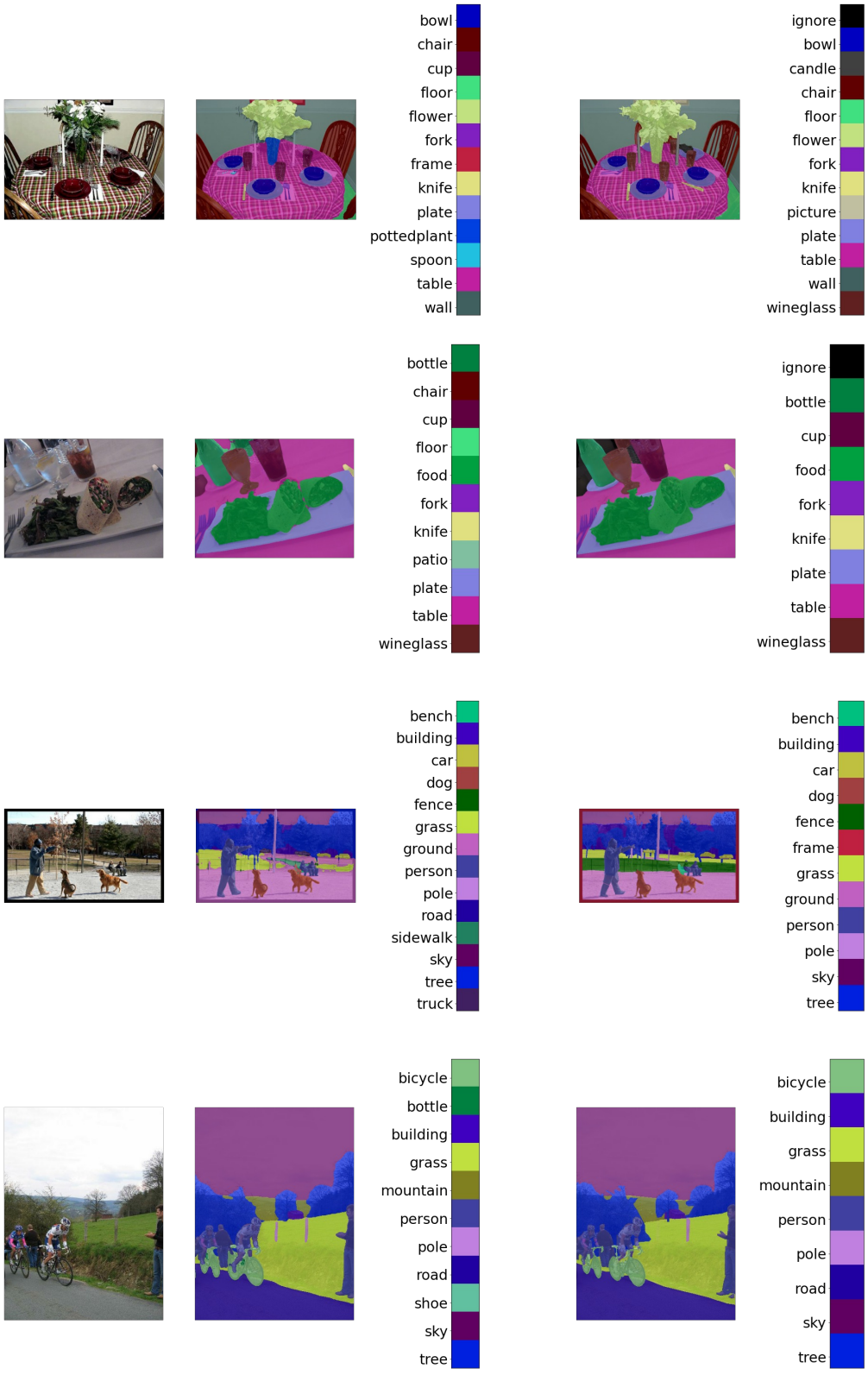


Figure 11: **Qualitative results of DaTaSeg directly transferring to PASCAL Context semantic dataset with 459 categories (PC-459).** The results demonstrate DaTaSeg has good generalization ability, and enables open-vocabulary segmentation. For each row, the left is the input image, the middle is our prediction, and the right is groundtruth. With a ViTDet-L backbone.

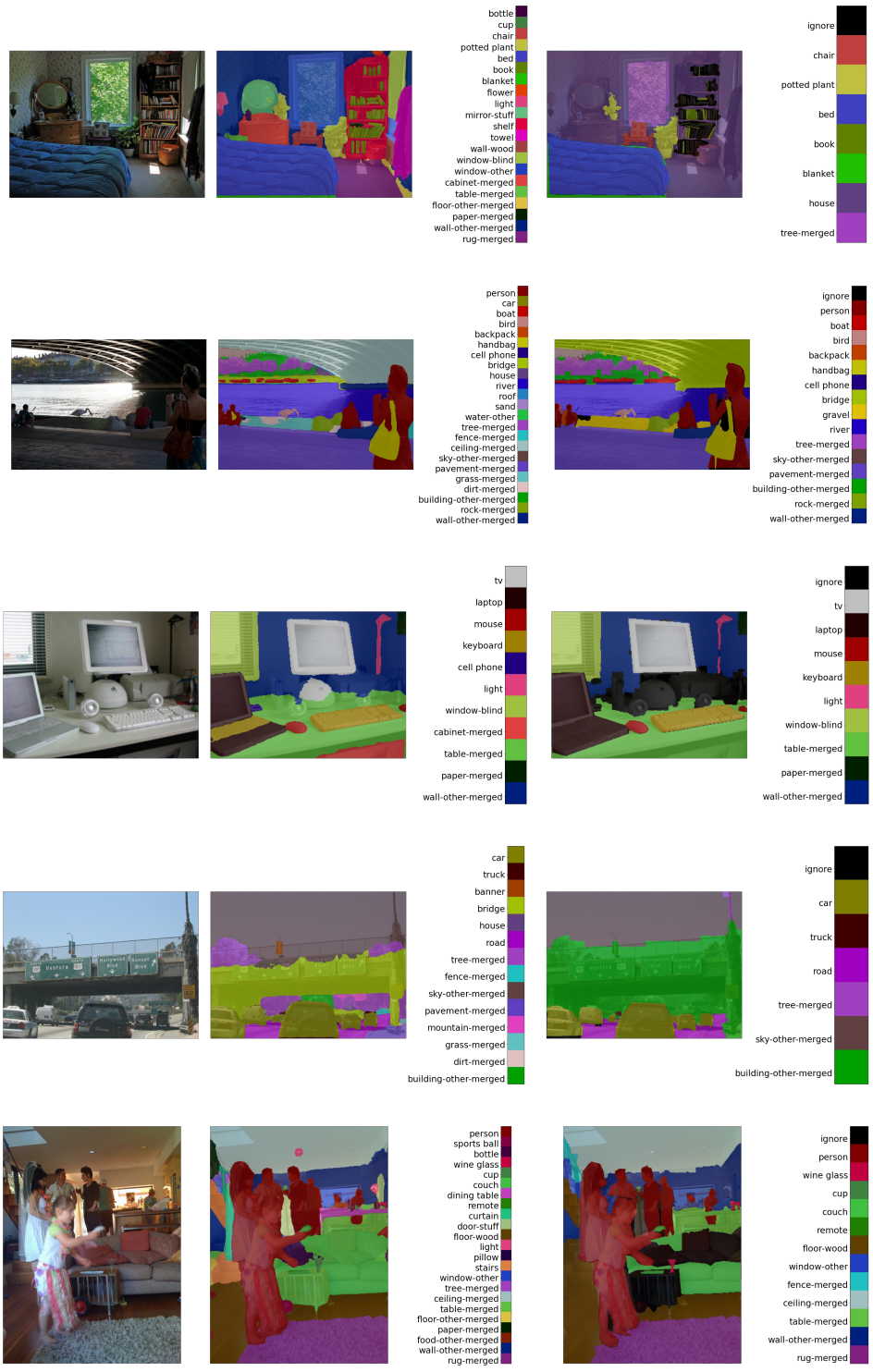


Figure 12: **Qualitative results on COCO semantic dataset.** DaTaSeg does high-quality semantic segmentation on COCO. Note that DaTaSeg trains on COCO panoptic. For each row, the left is the input image, the middle is our prediction, and the right is groundtruth. Our model is with a ViTDet-L backbone.

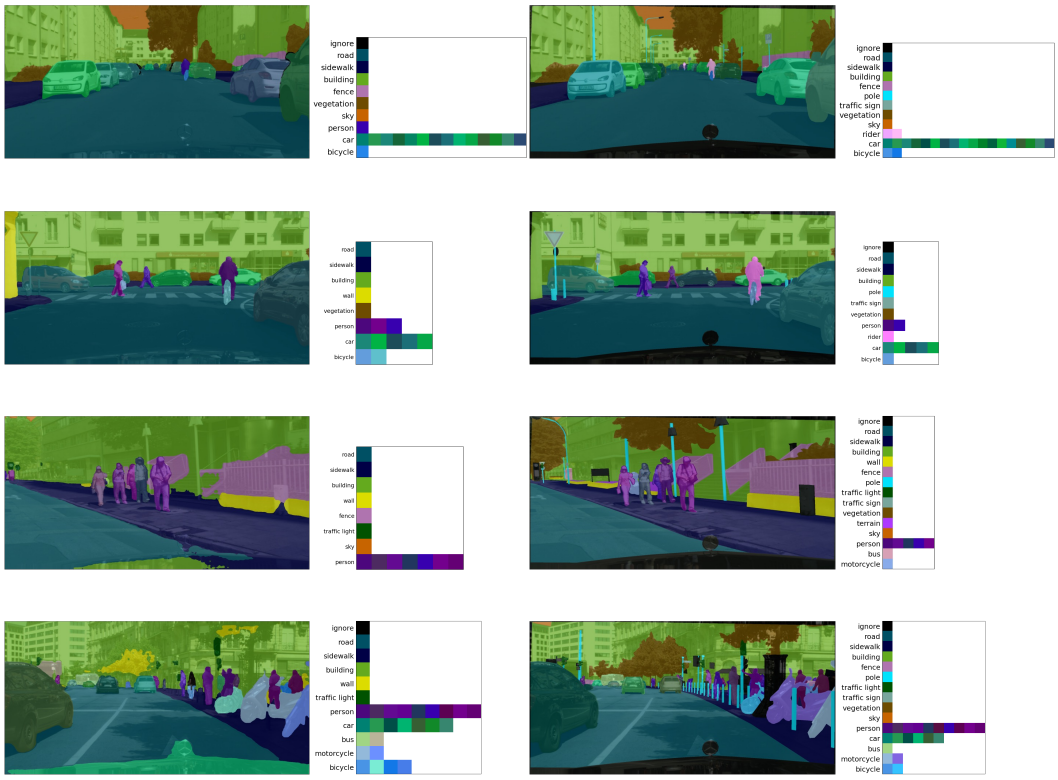


Figure 13: **Qualitative results of DaTaSeg directly transferring to Cityscapes panoptic dataset.** Cityscapes focuses on street view, which is different from all our training datasets. The results demonstrate good generalization ability. For each row, the left is our prediction and the right is groundtruth. Our model is with a ViTDet-L backbone.