SUPPLEMENTARY MATERIAL – Rethinking Semantic Segmentation: A Prototype View

Tianfei Zhou[1], Wenguan Wang[2,1*], Ender Konukoglu[1], Luc Van Gool[1]

[1] Computer Vision Lab, ETH Zurich [2] ReLER, AAII, University of Technology Sydney

https://github.com/tfzhou/ProtoSeg

This document provides more details of our approach and additional experimental results, organized as follows:

- §S1: Description of large-vocabulary datasets
- §S2: Pseudo-code of online clustering algorithm
- §S3: Additional experimental results
- §S4: More in-depth discussions

S1. Large-Vocabulary Dataset Description

First, we present the details of the datasets used for the study of large-vocabulary semantic segmentation (§5.3). In particular, the study is based on ADE20K-Full [19], which contains 25K and 2K images for `train` and `val`, respectively. It is elaborately annotated in an open-vocabulary setting with more than 3,000 semantic concepts. Following [3], we only keep the 847 concepts that are appearing in both `train` and `val` sets. Then, for each variant ADE20K-x in Table 4, we choose the top-x ($x \in \{300, 500, 700, 847\}$) classes based on their appearing frequencies. Note that for ADE20K-150, we follow the default setting in the SceneParse150 challenge to use the specified 150 classes for evaluation.

S2. Online Clustering Algorithm

Algorithm 1 provides a pseudo-code for our online clustering algorithm to solve Eq. 9. The algorithm only includes a small number of matrix-matrix products, and can run efficiently on a GPU card.

Algorithm 1 Pseudo-code of Online Clustering Algorithm in the PyTorch-like style.

```
# P: non-learnable prototypes (D x K)
# X: pixel embeddings (D x N)
# iters: sinhorn-knopp iteration number
# kappa: hyper-parameter (Eq.9)
# L: pixel-to-prototype assignment (K x N, Eq.9)

def online_clustering(P, X, iters=3, kappa=0.05)
   L = mm(P.transpose(), X)
   L = torch.exp(L / kappa)
   L /= torch.sum(L)

   for _ in range(iters):
     # normalize each row
     L /= torch.sum(L, dim=1, keepdim=True)
     L /= K

     # normalize each column
     L /= torch.sum(L, dim=0, keepdim=True)
     L /= N

   # make sure the sum of each column to be 1
   L *= N

   return L
```

mm: matrix multiplication.

S3. More Experimental Result

S3.1. Quantitative Result on Cityscapes `test`

Table S1 reports the results on Cityscapes `test`. All the models are trained on `train+val` sets. Note that we do not include any coarsely labeled Cityscapes data for training. For fair comparison with [15], we train our model with a cropping size of 1024×1024, and adopt sliding window inference with a window size of 1024×1024. As seen, our approach reaches **83.0%** mIoU, which is **0.8%** higher than SegFormer [15]. In addition, it greatly outperforms many famous segmentation models, such as HANet [4], HRNetV2 [13], SETR [18].

Method	Backbone	# Param (M)	mIoU (%)
PSPNet [CVPR17] [16]	ResNet-101 [7]	65.9	78.4
PSANet [ECCV18] [17]	ResNet-101 [7]	-	80.1
ContrastiveSeg [ICCV21] [14]	ResNet-101 [7]	58.0	80.3
†SETR [ICCV19] [18]	ViT [6]	318.3	81.0
HRNetV2 [PAMI20] [13]	HRNetV2-W48 [13]	65.9	81.6
CCNet [ICCV19] [8]	ResNet-101 [7]	-	81.9
HANet [CVPR20] [4]	ResNet-101 [7]	-	82.1
SegFormer [NeurIPS21] [15]	MiT-B5 [15]	84.7	82.2
Ours		84.6	**83.0** ↑ **0.8**

†: backbone is pre-trained on ImageNet-22K.

Table S1. **Quantitative results** (§S3.1) on Cityscapes [5] `test`.

S3.2. Quantitative Result with Lightweight Backbones

In Table S2, we compare our approach against four competitors using lightweight backbones (*i.e.*, MobileNet-

*Corresponding author: *Wenguan Wang*.

Method	Backbone	# Param (M)	mIoU (%)
FCN [CVPR15] [11]	MobileNet-V2 [12]	9.8	19.7
PSPNet [CVPR17] [16]	MobileNet-V2 [12]	13.7	29.6
DeepLabV3+ [ECCV18] [2]	MobileNet-V2 [12]	15.4	34.0
SegFormer [NeurIPS21] [15]	MiT-B0 [15]	3.8	37.4
Ours	MiT-B0 [15]	**3.7**	**38.5** ↑ **1.1**

Table S2. **Quantitative results** on ADE20K [19] `val` with lightweight backbones. See §S3.2 for details.

V2 [12], MiT-B0 [15]) on ADE20K [19] `val`. With MiT-B0, our model achieves the best performance (*i.e.*, **38.5%** mIoU) with the smallest number of parameters (*i.e.*, **3.7** M).

S3.3. Hyper-parameter Analysis of λ_1 and λ_2

Table S3 summarizes the influence of hyper-parameters λ_1 and λ_2 to model performance on ADE20K [19] `val`. We observe that our model is robust to the two coefficients, and achieves the best performance at $\lambda_1 = 0.01, \lambda_2 = 0.01$.

λ_1	0.001	0.005	0.01	0.02	0.03	0.05
mIoU (%)	46.1	46.3	**46.4**	**46.4**	46.2	46.3
λ_2	0.001	0.005	0.01	0.02	0.03	0.05
mIoU (%)	46.2	46.2	**46.4**	46.3	46.3	46.1

Table S3. Analysis of λ_1 and λ_2 on ADE20K [19] `val`.

S3.4. Embedding Structure Visualization

Fig. S1 visualizes the embedding learned by (left) parametric [15], and (right) our nonparametric segmentation model. As seen, in our algorithm, the pixel embeddings belonging to the same prototypes are well separated. This is because that our model is essentially based on a distance-based point-wise classifier and its embedding is directly supervised by the metric learning losses, which help reshape the feature space by encoding latent data structure into the embedding space.

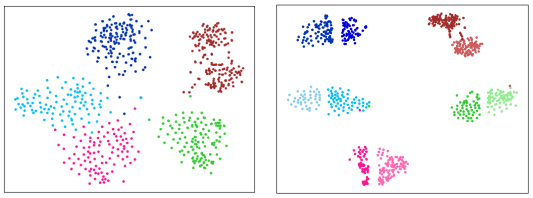

Figure S1. **Embedding spaces** learned by (left) parametric model [15], and (right) our nonparametric model. For better visualization, we show five classes of Cityscapes [5] with two prototypes per class.

S3.5. Additional Qualitative Result

We show more qualitative results on ADE20K [19] (Fig. S2), Cityscapes [5] (Fig. S3) and COCO-Stuff [1] (Fig. S4). As observed, our approach generally gives more accurate predictions than SegFormer [15].

S4. Discussion

Limitation Analysis. One limitation of our approach is that it needs a clustering procedure during training, which increases the time complexity. However, in practice, the clustering algorithm imposes a minor computational burden, only taking about 2.5 ms to cluster 10K pixels into 10 prototypes. Additionally, like many other semantic segmentation models, our approach is subject to some factors such as domain gaps, label quality and fairness. We will put more efforts on improving the "in the wild" robustness of our model in the future research.

Broader Impact. This work provides a prototype perspective to unify existing mask decoding strategies, and accordingly introduces a novel non-learnable prototype based nonparametric segmentation scheme. On the positive side, the approach pushes the boundary of segmentation algorithms in terms of model efficiency and accuracy, and shows great potentials in unrestricted segmentation scenarios with thousands of semantic categories. Thus, the research could find diverse real-world applications such as self-driving cars and robot navigation. On the negative side, our model can be misused to segment the minority groups for malicious purposes. In addition, the problematic segmentation may cause inaccurate decision or planning of systems based on the results.

Future Work. This work also comes with new challenges, certainly worth further exploration:

- **Closer Ties to Unsupervised Representation Learning.** Our segmentation model directly learns the pixel embedding space with non-learnable prototypes. A critical success factor of recent unsupervised representation learning methods lies on the direct *comparison* of embeddings. By sharing such regime, our nonparametric model has good potential to make full use of unsupervised representations.
- **Further Enhancing Interpretability.** Our model only uses the mean of several embedded 'support' pixel samples as the prototype for each (sub-)class. To pursue better interpretability, one can optimize the prototypes to directly resemble actual pixels, or region-level observations [9, 10].
- **Unifying Image-Wise and Pixel-Wise Classification.** A common practice of building segmentation models is to remove the classification head from a pretrained classifier and leave the encoder. This is not optimal as lots of 'knowledge' are directly dropped. However, with prototype learning, one can transfer the 'knowledge' of a nonparametric classier to a nonparametric segmenter intactly, and formulate image-wise and pixel-wise classification in a unified paradigm.

References

[1] Holger Caesar, Jasper Uijlings, and Vittorio Ferrari. Cocostuff: Thing and stuff classes in context. In *CVPR*, 2018. S2, S6

[2] 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. S2

[3] Bowen Cheng, Alexander G. Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021. S1

[4] Sungha Choi, Joanne T Kim, and Jaegul Choo. Cars can't fly up in the sky: Improving urban-scene segmentation via height-driven attention networks. In *CVPR*, 2020. S1

[5] 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. S1, S2, S5

[6] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2020. S1

[7] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. S1

[8] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *ICCV*, 2019. S1

[9] Jyh-Jing Hwang, Stella X Yu, Jianbo Shi, Maxwell D Collins, Tien-Ju Yang, Xiao Zhang, and Liang-Chieh Chen. Segsort: Segmentation by discriminative sorting of segments. In *ICCV*, 2019. S2

[10] Oscar Li, Hao Liu, Chaofan Chen, and Cynthia Rudin. Deep learning for case-based reasoning through prototypes: A neural network that explains its predictions. In *AAAI*, 2018. S2

[11] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. S2

[12] Mark Sandler, Andrew Howard, Menglong Zhu, Andrey Zhmoginov, and Liang-Chieh Chen. Mobilenetv2: Inverted residuals and linear bottlenecks. In *CVPR*, 2018. S2

[13] 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 TPAMI*, 2020. S1

[14] Wenguan Wang, Tianfei Zhou, Fisher Yu, Jifeng Dai, Ender Konukoglu, and Luc Van Gool. Exploring cross-image pixel contrast for semantic segmentation. In *ICCV*, 2021. S1

[15] Enze Xie, Wenhai Wang, Zhiding Yu, Anima Anandkumar, Jose M Alvarez, and Ping Luo. Segformer: Simple and efficient design for semantic segmentation with transformers. In *NeurIPS*, 2021. S1, S2, S4, S5, S6

[16] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. S1, S2

[17] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. Psanet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018. S1

[18] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip HS Torr, et al. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021. S1

[19] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *CVPR*, 2017. S1, S2, S4

Figure S2. **Qualitative results** of Segformer [15] and our approach on ADE20K [19] `val`.

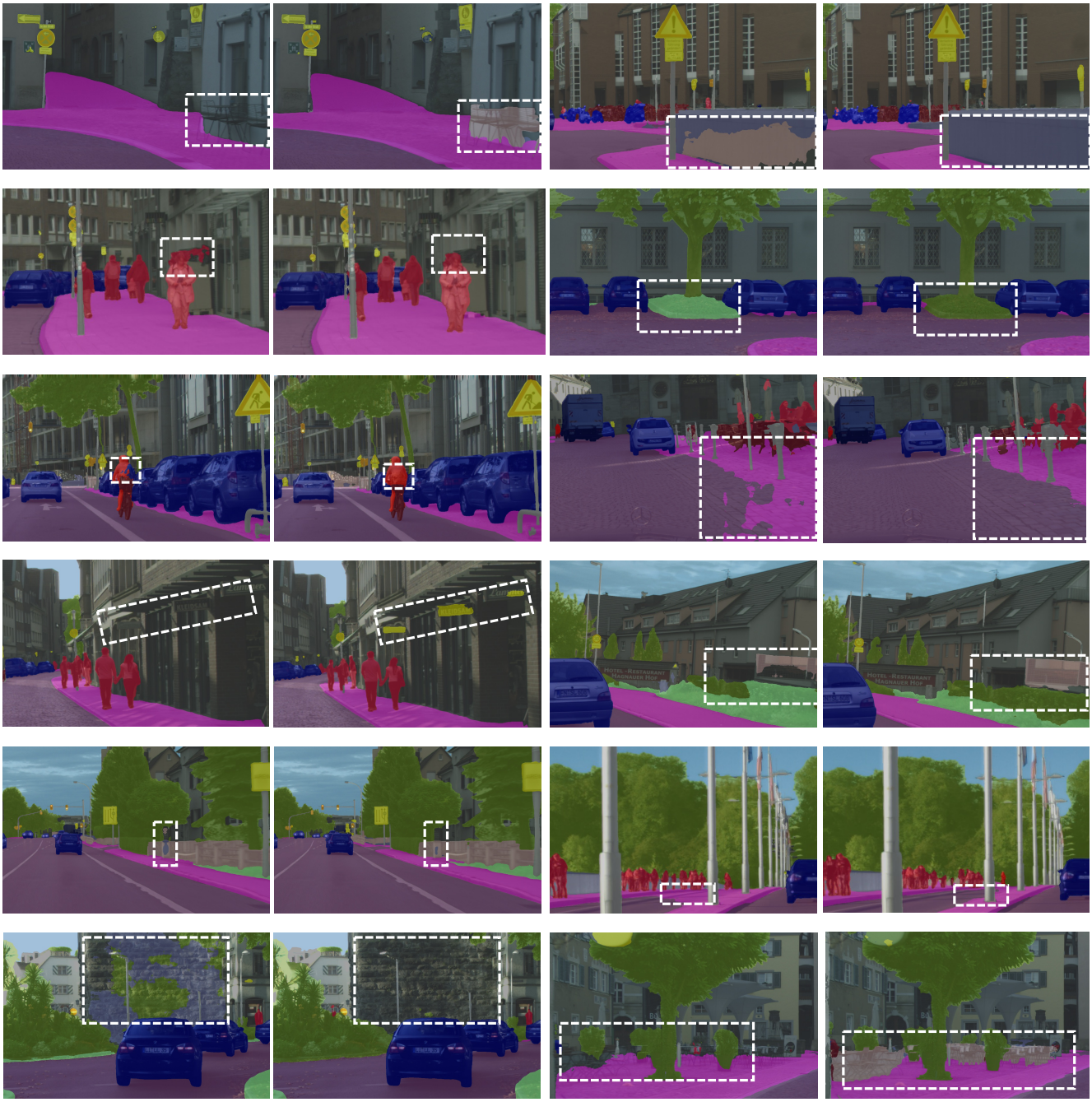

Figure S3. **Qualitative results** of Segformer [15] and our approach on Cityscapes [5] `val`.

Figure S4. **Qualitative results** of Segformer [15] and our approach on COCO-Stuff [1] `test`.