Appendix

A. Implementation Details

A.1. Panoptic SegFormer.

Our settings mainly follow DETR [1] and Deformable DETR [2] for simplicity. The hyper-parameters in deformable attention are the same as Deformable DETR [2]. We use Channel Mapper [2, 3] to map dimensions of the backbone's outputs to 256. The location decoder contains 6 deformable attention layers, and the mask decoder contains 6 vanilla cross-attention layers [4]. The spatial positional encoding is the commonly used fixed absolute encoding that is the same as DETR. The window size of Swin-L [5] we used is 7. Since we equally treat each query. λ_{things} and λ_{stuff} are dynamically adjusted according to the relative proportion of things and stuff in each image, and their sum is 1. λ_{cls}, λ_{seg}, and λ_{det} in **??** are set to 2, 1, 1, respectively.

During the training phase, the predicted masks that be assigned $\varnothing$ will have a weight of zero in computing $\mathcal{L}_{seg}$. While using the mass center of instance to replace the bounding box, we only use L1 loss to supervise the mass center of predicted mask and mass center of ground truth. We employ a threshold 0.5 to obtain binary masks from soft masks. Threshold t_{cnf} and t_{keep} are 0.25 (0.3) and 0.6, respectively. α and β in **??** are 1 and 2, respectively. All experiments are trained on one NVIDIA DGX node with 8 Tesla V100 GPUs.

By default, for COCO dataset [6], We train our models with 24 epochs, a batch size of 1 per GPU, a learning rate of 1.4×10^{-4} (decayed at the 18th epoch by a factor of 0.1, learning rate multiplier of the backbone is 0.1). We use a multi-scale training strategy with the maximum image-side not exceeding 1333 and the minimum image size varying from 480 to 800, and random crop augmentations is applied during training. The number of thing queries N_{th} is set to 300. Stuff queries have tge equal number of stuff classes, and it is 53 in COCO.

For the ADE20K dataset [7], we train our model with 100 epochs (decayed at 80th epoch), image size varying from 512 to 2048. Since ADE20K contains 50 stuff, we use 50 stuff queries. Other settings are the same to COCO dataset.

FPS and FLOPs. FPS in Tab.5 is measured on a V100 GPU with a batch size of 1. "DETR" and "DETR+mask wise merging" are from Detectron2 [8] and DETR's implementation. Others are from Mmdet [3] and our own implementation. Our framework is slightly more efficient than DETR. FLOPs of DETR are measured from Detectron2 on an average of 100 images.

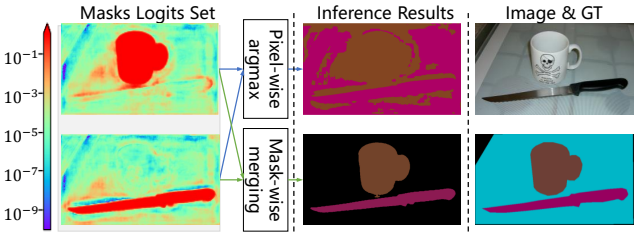

Figure B.1. **Pixel-wise Argmax *vs*. Mask-wise Merging.** We use DETR-R50 to compare the results generated through pixel-wise armgax and mask-wise merging. Firstly, DETR-R50 detects a cup and a knife from the image. When using pixel-wise argmax, other pixels (dining table) are incorrectly filled with "cup" or "knife". It mistakenly believes that the largest mask logit is the correct result, regardless of its value. However, our mask-wise merging strategy generates the correct results since we binarize each mask.

A.2. Deformable DETR for Panoptic Segmentation

Following DETR for panoptic segmentation, we transplanted the panoptic head of DETR to Deformable DETR. To ensure consistency, we only generate the attention maps with the spatial shape of 32s. When using single scale deformable DETR, the process of generating attention maps is the same as DETR. When using multi-scale deformable DETR, we only multiply queries and the features (from C5) to generate attention maps. Other settings of deformable DETR for object detection are kept unchanged. We apply iterative bounding box refinement as the default setting for Deformable DETR. We use 300 queries and this brings huge computation costs, although this model achieves pretty good performance.

B. Discussion

We will deliver more ablation studies, more detailed analysis in this section.

Effect of Deformation Attention. To ablate the effect of deformable attention, we extend Deformable DETR on panoptic segmentation with the panoptic head of DETR. For more implementation details, please refer to the Appendix A. As shown in Tab. B.1, multi-scale deformable attention improves 2.9% PQ compared to DETR. Multi-scale attention outperforms single-scale attention by 5.7% PQ, highlighting the important role of multi-scale features for segmentation task.

Method	Epoch	PQ	PQ^{th}	PQ^{st}
DETR [1]	500	43.4	48.2	36.3
D-DETR-SS	50	40.6	44.0	35.4
D-DETR-MS	50	46.3	51.9	37.9

Table B.1. "D-", "SS" and "MS" refers to "Deformable", single-scale and multi-scale.

B.1. Post-processing Method

Defects of Pixel-wise Argmax. Pixel-wise argmax only considers the mask logits of each pixel. It has multiple issues that may lead to incorrect results. First of all, the

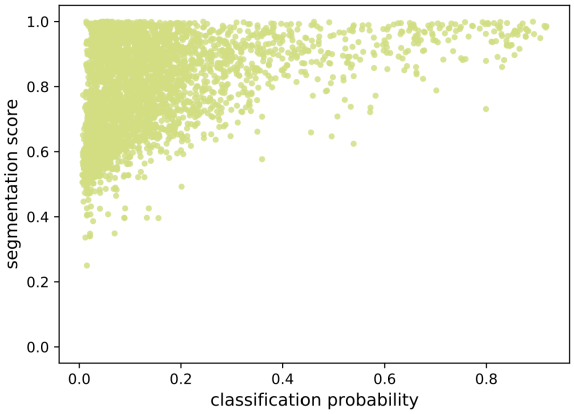

Figure B.2. The joint distribution for classification probability and segmentation score. We can observe that segmentation scores can be high while the masks have low classification probability.

Post-Processing Method	PQ	PQ^{th}	PQ^{st}
Pixel-wise Argmax	48.4	53.2	41.3
Heuristic Procedure [10]	48.4	54.3	39.4
Mask-wise Mering	49.6	54.4	42.4

Table B.2. The results of Panoptic SegFormer (R50) with different post-processing methods. Because the heuristic procedure always prefers things, it has the lowest PQ^{st}

α	β	PQ	PQ^{th}	PQ^{st}
1	0	48.7	53.5	41.3
0	1	44.4	52.1	32.7
1	1	49.3	54.1	42.1
1	2	49.6	54.4	42.4
1	3	49.7	54.5	42.2

Table B.3. The weights of the classification score and segmentation score determine the priority of masks. We can observe that employing both of them will perform better. According to the results on multiple models, we choose $\alpha=1$, $\beta=2$ as our default setting.

pixel value generated from argmax may be extremely small, as shown in Fig. B.1, which will generate plenty of false-positive results. The second issue is that the pixel with max mask logit may be the suboptimal result, as shown in **??** of the paper. This kind of error frequently appears in the segmentation maps generated by pixle-wise argmax. MaskFormer [9] alleviates this problem by multiplying the classification probability by the masks logits. But this kind of error will still exist.

Heuristic Procedure. The heuristic procedure [10] was the first proposed post-processing method of panoptic segmentation. It uses different strategies to handle things and stuff separately. Pixel-wise argmax was still used in its stuff workflow. One apparent defect of this method is that it solves the overlap problem of stuff and things by always preferring things. This is an unfair way of dealing with stuff. Tab. B.2 shows that PQ^{st} of using heuristic procedure is lower than other methods because all stuff are treated unfairly.

t_{cnf} \ t_{keep}	0.9	0.8	0.7	0.6	0.5
0.20	48.9	**49.5**	**49.6**	**49.6**	49.4
0.25	48.9	**49.6**	**49.7**	**49.7**	**49.5**
0.30	48.8	**49.5**	**49.6**	**49.6**	**49.5**
0.35	48.3	49.1	49.2	49.2	49.1
0.40	47.4	48.1	48.2	48.2	48.1

Table B.4. We use two thresholds t_{cnf} and t_{keep} in our mask-wise merging. We evaluate the results by combining different thresholds with Panoptic SegFormer (R50) to verify whether our algorithm is sensitive to these thresholds. Results higher than 49.5 are displayed in bold.

#Head	PQ	PQ^{th}	PQ^{st}
1	49.2	54.0	42.0
8	49.6	54.4	42.4

Table B.5. We varied the number of heads in our mask decoder. More heads can bring slight performance improvements.

Masks-wise Merging. Post-processing of panoptic segmentation aims to solve the overlap problem between masks. Although pixel-wise argmax uses an intuitive method to solve the overlap problem, it has defects mentioned above. We solve the overlap problem by giving different masks different priorities. Mask-wise merging guarantees that high-quality masks have higher priority by sorting the masks with confidence scores. This strategy ensures that low-quality instances will not cover high-quality instances. In order to be able to effectively distinguish the quality of the masks, we consider both classification probability and segmentation score as the confidence score of each mask. The segmentation score $\text{average}(\mathbb{1}_{\{m_i[h,w]>0.5\}} m_i[h,w])$ represents the confidence of the overall segmentation quality of the mask. Tab. B.3 shows the results of varying α and β in Eq.6. Applying both classification probability and segmentation scores always have better performance. Fig. B.2 shows the relationship of classification probability and segmentation score. While one mask has a low classification probability ($[0, 0.4]$), it may have a large segmentation score. Large segmentation score means it has many pixels with high logits and this may generate false-positive results through pixel-wise argmax since its classification probability is pretty low.

Our mask-wise merging needs two thresholds to filter out undesirable results. Tab. B.4 shows that our algorithm is not very dependent on the choice of threshold t_{cnf} and t_{keep}. Tab. B.4

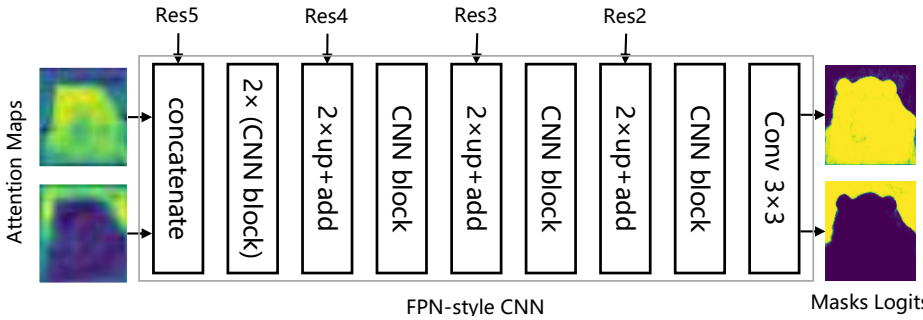

Figure B.3. Illustration of DETR's panoptic head. CNN block consists of 3×3 convolution, GN, and ReLU.

P_t	#Query	Stuff			Things		
		TP	TP+FP	Precision	TP	TP+FP	Precision
[0.0, 0.1)	44	7318	10060	0.73	136	222	0.61
[0.1, 0.2)	17	839	1308	0.64	140	198	0.71
[0.2, 0.3)	4	121	212	0.57	53	69	0.77
[0.3, 0.4)	11	339	646	0.52	252	368	0.68
[0.4, 0.5)	10	211	446	0.47	212	365	0.58
[0.5, 0.6)	15	327	684	0.48	477	903	0.53
[0.6, 0.7)	24	339	810	0.42	1001	1465	0.68
[0.7, 0.8)	40	400	1094	0.37	2019	3255	0.62
[0.8, 0.9)	83	539	1586	0.34	6325	9687	0.65
[0.9, 1.0]	105	309	1029	0.30	11252	16724	0.67
Total	353	10742	17875	0.60	21867	33256	0.66

Table B.6. We divide 353 queries into ten groups according to their P_t. For each group, we calculate their precision on stuff and things. Queries with higher P_t have very low precision when they predict stuff. This demonstrates that things may interfere with the prediction of stuff and using queries to target both things and stuff is suboptimal.

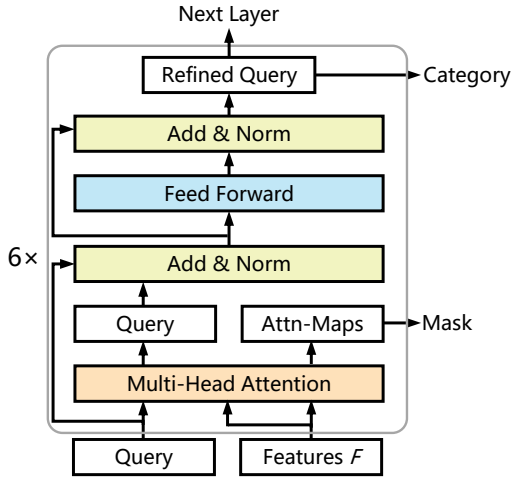

Figure B.4. **Architecture of mask decoder.** Attn-Maps notes attention maps.

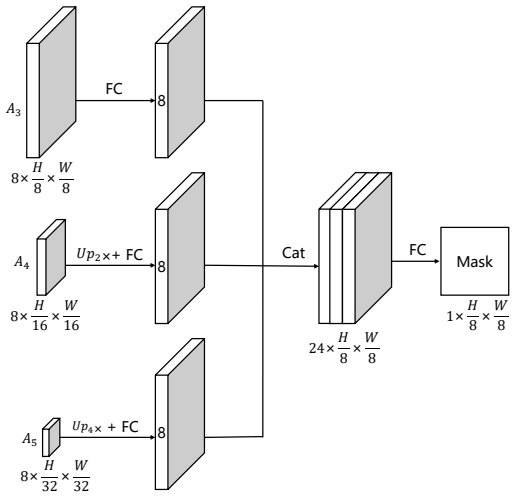

Figure B.5. Illustration of the module that generates mask from multi-scale multi-head attention maps. Up_{2x} means upsampling by two times. FC notes fully connected layer. Cat notes concatenate. While using 8 heads in the attention module, this module only contains 200+ parameters.

Although our proposed mask-wise merging strategy has achieved better results than other post-processing methods, it also has several shortcomings. First of all, we binaries the mask through a fixed threshold. This may cause one pixel to be easily assigned a void label because the values of all candidate instances at this pixel are below the threshold. Secondly, our strategy highly depends on the accuracy of

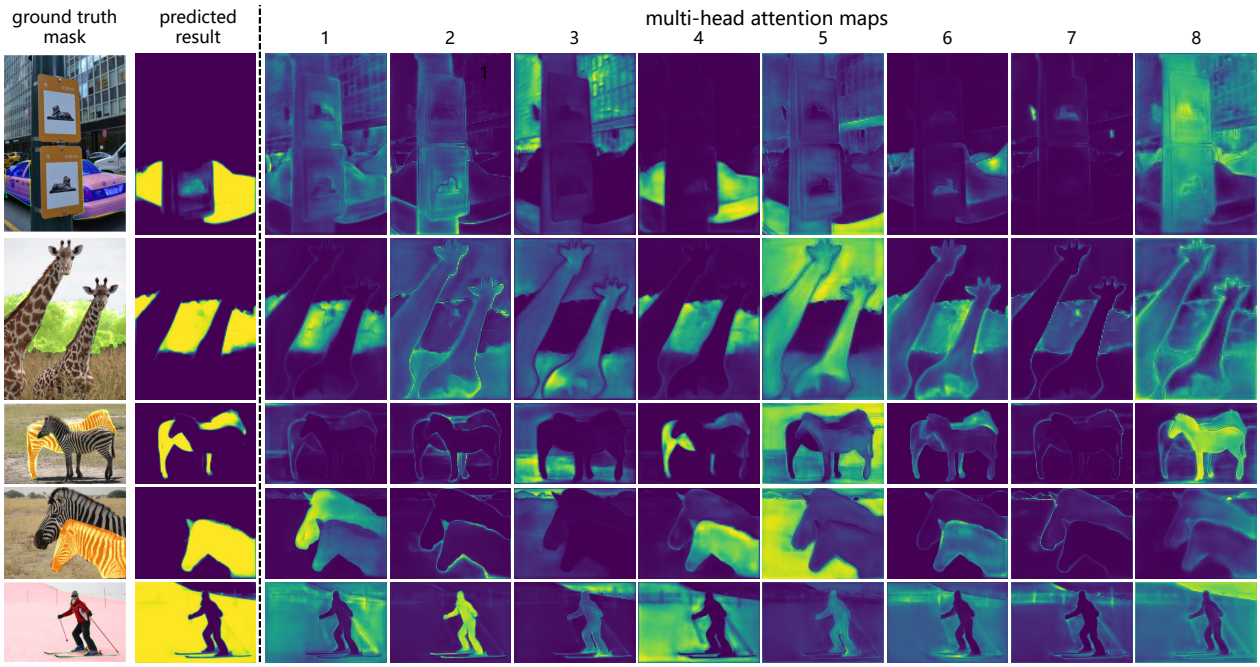

Figure B.6. **Visualization of multi-head attention maps and corresponding outputs from mask decoder.** Different heads have different preferences. Head 4 and Head 1 pay attention to foreground regions, and Head 8 prefers regions that occlude foreground. Head 5 always pays attention to the background that is around the foreground. Through the collaboration of these heads, Panoptic SegFormer can predict accurate masks. The 3rd row shows an impressive result of a horse that is highly obscured by the other horse.

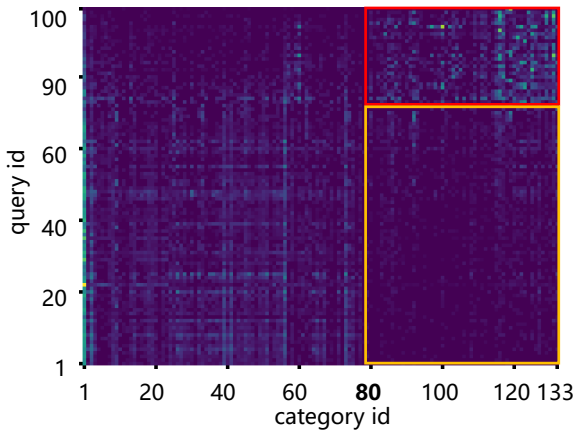

Figure B.7. **The Joint Distribution for Queries and Category in DETR.** We can observe that queries prefer either things or stuff. Although a few queries predict most of the stuff results (within the red box), other queries still generate a considerable proportion of stuff results (within the yellow box). Our experimental results demonstrate that the results in the yellow box are usually of low quality. We sort the query ids for better visualization. Other literature [11] reports similar phenomenon.

confidence scores. If the confidence scores are not accurate, it will produce a low-quality panoptic format mask.

B.2. Location Decoder

Although we use the location decoder to detect the bounding boxes of things, our workflow is still very different from the previous box-based panoptic segmentation. For example, Panoptic FPN performs instance segmentation with Mask R-CNN style. The two-stage method usually needs to extract regions from the feature based on the bboxes and then use these regions to perform segmentation. The quality of segmentation is heavily dependent on the quality of detection. However, our location decoder is used to assist in learning the location clues of the query and distinguishing different instances. Mask will not have the wrong boundary due to the wrong boundary prediction of the bbox since the bbox does not constrain the mask. We also show that using mass centers of masks to replace bboxes can still learn location clues.

Another valuable function of the location decoder is to help filter out low-quality thing queries during the training and inference phase. This can greatly save memory. Current transformer-based panoptic segmentation methods always consume a lot of GPU memory. For example, MaskFormer takes up more than 20G of GPU memory with a batch size

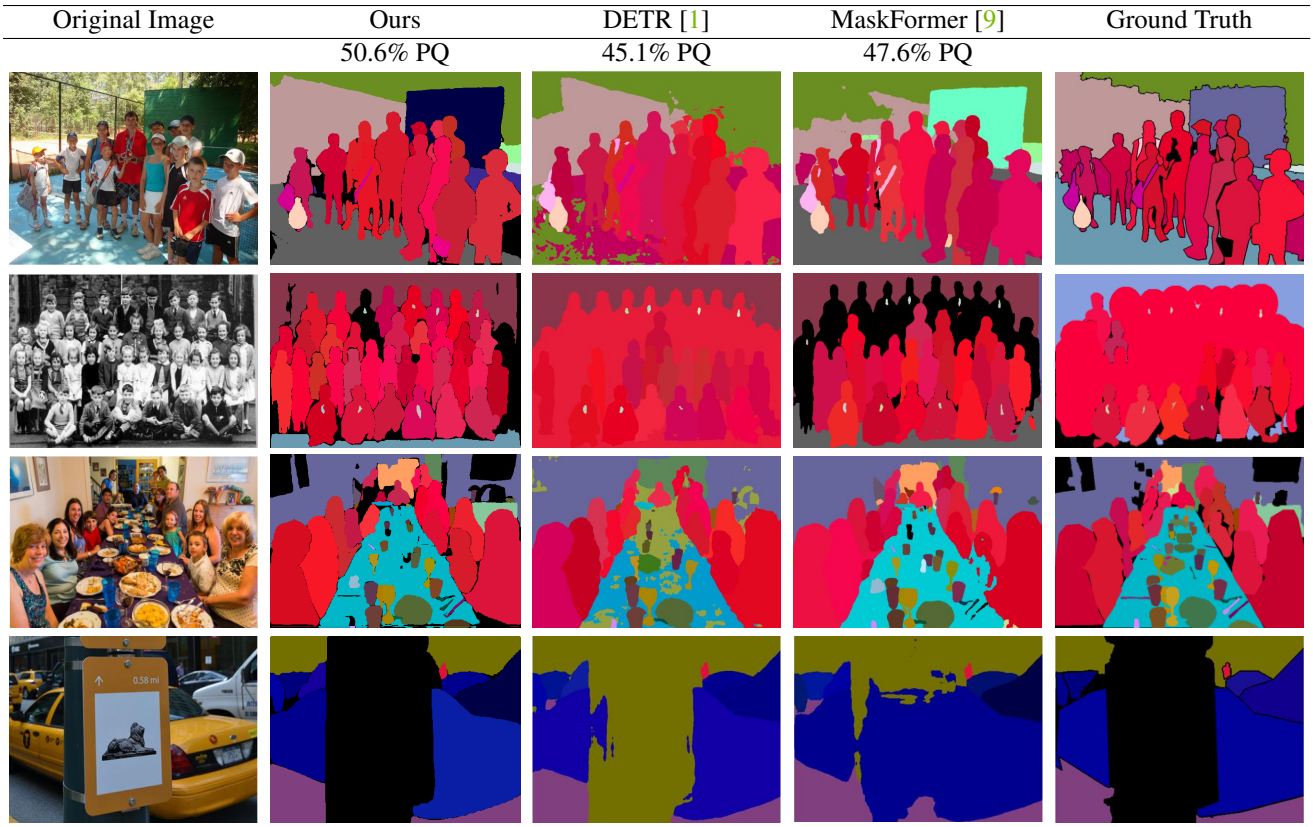

Figure B.8. **Comparing visualization results of Panoptic SegFormer with other methods on the COCO `val` set.** For a fair comparison, all results are generated with ResNet-101 [12] backbone. The second and fourth row results show that our method still performs well in highly crowded or occluded scenes. Benefits from our mask-wise inference strategy, our results have few artifacts, which often appear in the results of DETR [1] (*e.g.*, dining table of the third row).

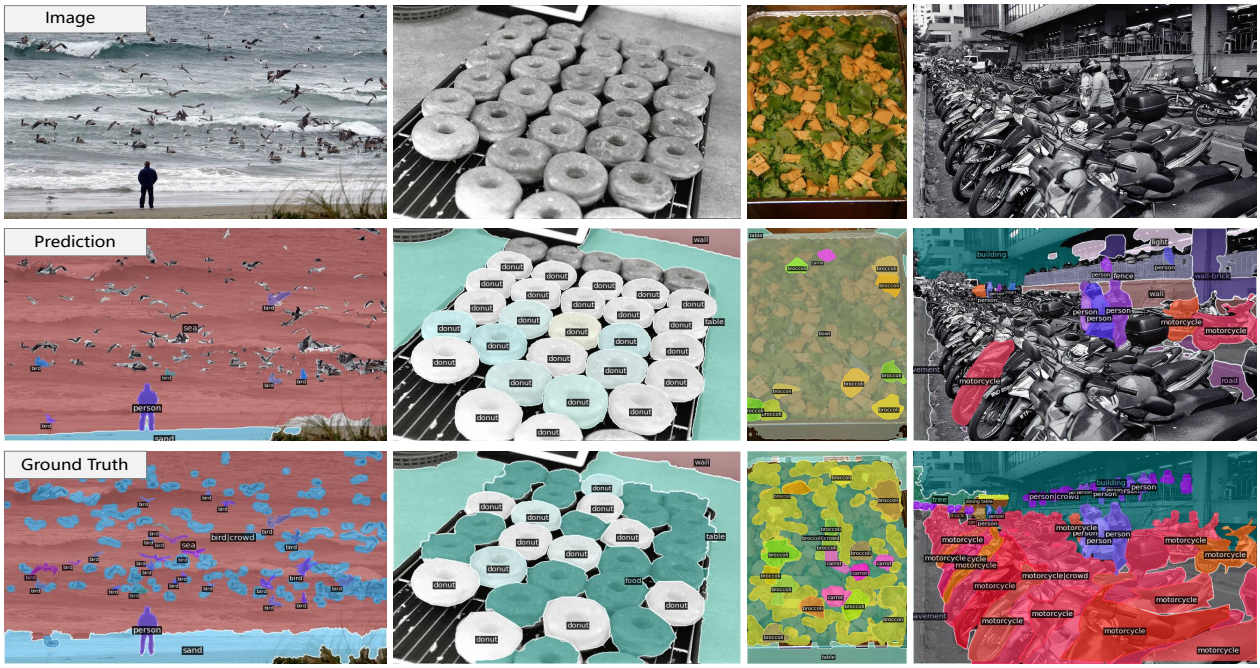

Figure B.9. Failure case of Panoptic SegFormer.

Figure B.10. Visualization results of some complex scenes.

of 1 and R50 backbone. Although these methods have achieved excellent results, they also require high hardware resources. However, our Panoptic SegFormer can be trained with taking up less than 12G memory by using a location decoder to filter out low-quality thing queries. In particular, we use bipartite matching for multiple rounds of matching in the detection phase. The thing query that already be matched will not participate in the next round of matching. After several rounds, we can select partial promising thing queries. Only these promising thing queries will be fed to the mask decoder. with this strategy, the mask decoder usually only needs to handle less than 100 thing and stuff queries.

B.3. Mask Decoder

Fig. B.3 shows the architecture of DETR's panoptic head. Although it only contains 1.2M parameters, it has a huge computational burden (about 150G FLOPs). DETR adds ResNet features to each attention map, and this process repeats 100 times since there are 100 attention maps. Fig. B.4 shows the model architecture of our mask decoder. Fig. B.5 shows the process of converting multi-scale multi-head attention maps to mask. We found that discarding the self-attention in the decoder does not affect the effectiveness of the model. The computational cost of our mask decoder is around 30G FLOPs.

Multi-head attention maps. Fig. B.6 shows some samples of multi-head attention maps. Through a multi-head attention mechanism, different heads of one query learn

their own attention preference. We observe that some heads pay attention to foreground regions, some prefer boundaries, and others prefer background regions. This shows that each mask is generated by considering various comprehensive information in the image. Tab. B.5 shows that utilizing a multi-head attention mechanism will outperform single-head attention by 0.4% PQ.

B.4. Advantage of Query Decoupling Strategy

DETR uses the same recipe to predict boxes of things and stuff (To facilitate the distinction between the query decoupling strategy we proposed, we refer to the DETR's strategy as a joint matching strategy.). However, detecting bboxes for stuff like DETR is suboptimal. We counted the ratio of the area of masks to the area of bboxes on the COCO `train2017`. The ratios of things and stuff are 52.5% and 9.2%, separately. This shows that bounding boxes can not represent stuff well since the stuff is amorphous and dispersed. We also observe that bbox AP of DETR drops from 42.0 to 38.8 after training on panoptic segmentation. This may be due to the interference of stuff on things, since predicting stuff bboxes needs to re-adapt the model.

Fig. B.7 shows that DETR seems to learn automatic segregation between things and stuff, and each query either prefers things or stuff. However, we argue that this automatically learned segregation is not ideal. If one query prefers things, it will perform poorly when it generates stuff results. This situation is very common, and our following experiments based on Panoptic SegFormer will give detailed data. Following DETR, we use 353 queries to predict things and stuff with the same recipe. Specifically, the input of the location decoder is 353 queries, which will detect both things and stuff. The refined queries are fed to the mask decoder to predict category labels and masks. We define a query's preference for things as P_t, which can be calculated by:

$$\mathrm{P}_t^i = \mathrm{N}_{\text{things}}^i / (\mathrm{N}_{\text{things}}^i + \mathrm{N}_{\text{stuff}}^i), \tag{1}$$

where $\mathrm{N}_{\text{things}}$ and $\mathrm{N}_{\text{stuff}}$ are the number of things and stuff masks that i-th query predicted on COCO val set. $\mathrm{P}_t^i > 0.5$ means that i-th query prefers things more than stuff. The predicted mask is a true positive (TP) if IoU between it and one ground truth mask is larger than 0.5 and the category of them is the same. Then we can calculate the precision of queries' predicted masks. Tab. B.6 shows relevant statistical results. First, we can observe that the queries that own lower P_t basically have higher precision. The stuff precision of the queries that have the highest P_t ($[0.9, 1.0]$) only is 0.30, which is much lower than the average stuff precision (0.60) on all queries. These erroneous results are mainly due to errors in the predicted category. Queries that have no obvious preference for stuff and things(P_t in $[0.4, 0.6)$) performs poorly both on stuff and things. These results demonstrate that using one query set to predict things and stuff simultaneously is flawed. This joint matching strategy is suboptimal for stuff.

In order to avoid mutual interference between stuff and things, we propose the query decoupling strategy to handle things and queries with a separate query set. Compared to stuff query, thing query will go through an additional location decoder. However, all queries will produce the outputs in the same format. Things and stuff use the same loss for training, except that things use an additional detection loss. During inference, we can use our mask-wise merging strategy to merge them uniformly. This is different from the previous methods that modeled panoptic segmentation into instance segmentation and semantic segmentation. For example, Panoptic FPN uses one branch to generate things and one branch to generate stuff. The things and stuff generated by Panoptic FPN are in different formats and need different training strategies and post-processing methods. $\mathrm{PQ}^{\mathrm{st}}$ with query decoupling outperforms joint matching strategy by 2.9% PQ and experimental results verify the effectiveness of our method. The stuff precision by using query decoupling is 0.66, better than the joint matching strategy.

C. Visualization

Fig. B.8 shows our visualization result against DETR and MaskFormer. We use the original codes that they officially implemented. First of all, compared with other methods, we can observe that our results are more consistent with ground truths. Due to the defects of pixel-wise argmax we discussed in Appendix B.1, DETR always generates results with artifacts. MaskFormer performs better because they improved pixel-wise argmax by considering classification probabilities. However, it may still fail in hard cases. For example, it recognizes the billboard as a car in the fourth row. Fig. B.9 shows some failure cases of our model. Firstly, our model may have lower recall when facing crowded scenarios filled with the same things, especially for the small targets. Another typical failure mode is that large stuff with a high confidence score occupies most of the space, causing other things not to be added to the canvas. Fig. B.10 shows the results on some complex scenes.

D. Various Backbones

We give all the panoptic segmentation results under various backbones, as shown in Tab. D.1. Fig. D.1 shows two training curves with backbone ResNet-101 and Swin-L. With Swin-L, Panoptic SegFormer with training for 24 epochs even performs better than training for 50 epochs.

Backbone	PQ	SQ	RQ	PQ^{th}	SQ^{th}	RQ^{th}	PQ^{st}	SQ^{st}	RQ^{st}
ResNet-50 [12]	49.6	81.6	59.9	54.4	82.7	65.1	42.4	79.9	52.1
ResNet-101 [12]	50.6	81.9	60.9	55.5	83.0	66.3	43.2	80.1	52.9
PVTv2-B0 [13]	49.5	82.4	59.2	55.3	83.3	65.8	40.6	80.9	49.2
PVTv2-B2 [13]	52.5	82.7	62.7	58.5	83.6	69.5	43.4	81.4	52.4
PVTv2-B5 [13]	55.4	82.9	66.1	61.2	84.0	72.4	46.6	81.3	56.5
Swin-L [5]	55.8	82.6	66.8	61.7	83.7	73.3	46.9	80.9	57.0

Table D.1. Panoptic segmentation results on COCO val with various backbones.

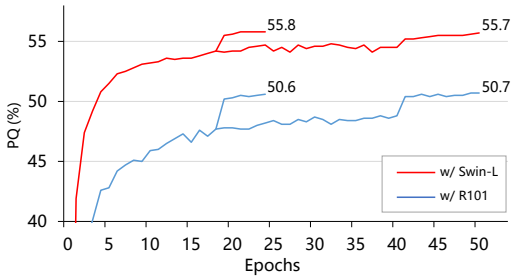

Figure D.1. By using ResNet-101 [12] and Swin-L as the backbone, we train our model for 24 epochs and 50 epochs, separately. We can observe that our model that training for 24 epochs can achieve comparable or even higher results while comparing the models that training for 50 epochs.

E. Code and Data

We use the official implementations of DETR[1], MaskFormer[2], Panoptic FCN[3] to perform additional experiments. The models they provide all can reproduce the same scores they reported in their literature. Deformable DETR is from Mmdet[4].

References

[1] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020. 1, 5

[2] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. In *ICLR*, 2020. 1

[3] 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:1906.07155*, 2019. 1

[4] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017. 1

[5] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *ICCV*, 2021. 1, 8

[6] 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

[7] Bolei Zhou, Hang Zhao, Xavier Puig, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Scene parsing through ade20k dataset. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 633–641, 2017. 1

[8] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. `https://github.com/facebookresearch/detectron2`, 2019. 1

[9] Bowen Cheng, Alexander G Schwing, and Alexander Kirillov. Per-pixel classification is not all you need for semantic segmentation. In *NeurIPS*, 2021. 2, 5

[10] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In *CVPR*, 2019. 2

[11] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. In *CVPR*, 2021. 4

[12] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 5, 8

[13] Wenhai Wang, Enze Xie, Xiang Li, Deng-Ping Fan, Kaitao Song, Ding Liang, Tong Lu, Ping Luo, and Ling Shao. Pvtv2: Improved baselines with pyramid vision transformer. *arXiv:2106.13797*, 2021. 8

[1] https://github.com/facebookresearch/detr
[2] https://github.com/facebookresearch/MaskFormer
[3] https://github.com/dvlab-research/PanopticFCN
[4] https://github.com/open-mmlab/mmdetection