Appendices

A. Introduction

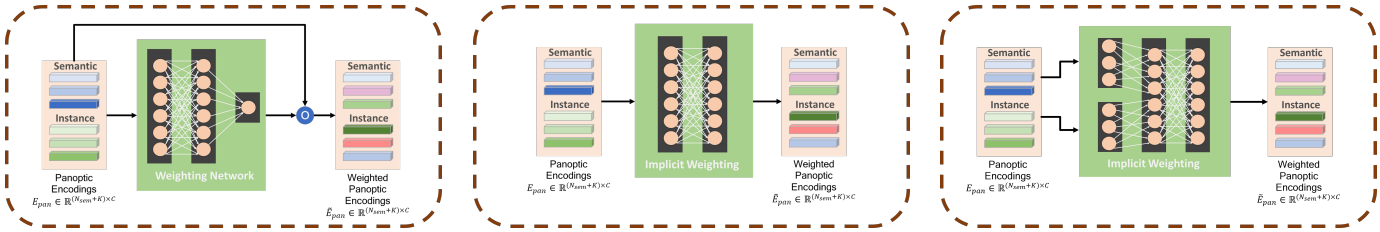
As part of the supplementary materials for this paper, we present our hyper-parameters and show more visual and quantitative results as an extension to the ones shown in the paper. The supplementary materials contain:

- Further experiments and ablation studies for alternative feature processing schemes on Cityscapes.
- Additional qualitative results on ADE20K comparing the predictions produced by the PISR models to those predicted by the baseline network.
- Visualization of the t-SNE projections of semantic and instance encodings on Cityscapes, produced by PISR.
- Implementation details and training hyper-parameters for our networks based on Panoptic Deeplab [11], Panoptic FPN [21] and MaskFormer [12], which are mentioned in the main paper.
- Detailed explanation for generating the semantic heatmaps visualized in Figure 4 and Figure 5 of the main paper.
- Visualization of the intermediate instance heatmaps which are used to generate instance encodings as described in Section 3.2 of the main paper.

B. Additional Experiments

In this section, we provide additional ablation studies on alternative feature processing schemes which were not included in the main paper.

B.1. Comparing various Reweighting Schemes



(a) Explicit reweighting (Default) (b) Implicit reweighting (c) Separated reweighting

Figure B.1. Detailed structure of alternative reweighting networks to comparing with our proposed Explicit reweighting scheme. Note that (a) is the proposed structure.

In Section 3 of the main paper, we describe our method of reweighting the panoptic encodings E_{pan} to reflect importance. In this section, we describe alternate methods to perform this reweighting operation. An illustration of these approaches is provided in Figure B.1.

Explicit reweighting (Default): We display this method in Section 3 of the main paper and use it as the default reweighting method throughout the paper. The reweighting network consists of linear layers followed by a sigmoid layer. It explicitly multiplies the learned weights with panoptic encodings E_{pan} to produce the final reweighted panoptic encodings $\tilde{E}_{pan}$. This method is illustrated in Figure B.1 (a).

Implicit reweighting: Illustrated in Figure B.1 (b), this reweighting network directly computes the final encodings instead of multiplying computed weights with input encodings. Hence, the weighting operation is implicitly computed and the panoptic encodings are transformed to weighted encodings.

Separated reweighting: For this reweighting scheme, we process semantic and instance encodings using separate linear layers and then concatenate them to form a fused implicit representation. This is done to reduce any mismatch in scale between the two encodings. After concatenation, the reweighting network computes the final encodings in an implicit fashion. This method is illustrated in Figure B.1 (c).

Method	Reweighting	# Hidden layers	PQ	IoU	iIoU	Δ
Panoptic DeepLab	-	-	59.9	78.5	62.5	0
Panoptic DeepLab + PISR	Default	2	**62.2**	**80.2**	**64.4**	**+5.9**
	Default	4	61.5	79.4	63.7	+3.7
	Implicit	2	61.2	79.0	63.3	+2.6
	Separate	2	62.0	80.1	64.2	+5.4

Table B.1. **On Cityscapes val:** Effect of applying various reweighting schemes described in Figure B.1 to generate the weighted Panoptic Encodings. Gray rows are our models introduced in this paper.

As observed from Table B.1, We obtain the best results using our default explicit weighting scheme using two hidden layers. Our experiments show that increasing the number of hidden layers of the default network reduces performance. We hypothesize that the larger number of parameters leads to overfitting this network.

We also observe the decrease in PQ after using the implicit reweighting scheme. This is due to a scale mismatch, as semantic and instance encodings are computed from different sources. We can fix this issue by processing each of the encodings using separate linear layers before fusion, which is the input to the implicit network.

B.2. Varying the Correlation functions in Panoptic Relational Attention:

In Section 3.4 of the main paper, we propose a method to perform Panoptic Relational Attention(PRA) to leverage the relational context between things and stuff. This includes computing the spatial feature map $F_{sp} = g_s(\tilde{E}_{pan})h(F)$, which correlates panoptic encodings and global features. Next, the spatial features are correlated with panoptic encodings, obtaining the panoptic features $F_{pan} = g_p(\tilde{E}^T_{pan})h(F_{sp})$. In this section, we observe various schemes denoted by $f_1(\cdot,\cdot)$ and $f_2(\cdot,\cdot)$ to compute the correlation for both operations such that $F_{sp} = f_1(g_s(\tilde{E}_{pan}), h(F))$ and $F_{pan} = f_2(g_p(\tilde{E}_{pan}), h(F_{sp}))$. By default, we assume f_1 and f_2 to represent the matrix product in the main paper. Table B.2 summarizes the effect of applying cosine similarity or p-norm distance instead of the matrix product.

Method	f_1	f_2	PQ	mIoU	iIoU	Δ
Panoptic DeepLab	-	-	59.9	78.5	62.5	0
Panoptic DeepLab + PISR	Cosine Sim	Matmul	61.4	79.3	63.5	+3.2
	Matmul	Cosine Sim	61.4	80.0	64.0	+4.5
	L2 norm	L2 norm	61.7	79.7	63.9	+4.4
	Matmul	L2 norm	62.1	79.9	64.1	+5.2
	Matmul	Matmul	**62.2**	**80.2**	**64.4**	**+5.9**

Table B.2. **On Cityscapes val:** Effect of varying the correlation functions f_1 and f_2 to compute correlation from the given panoptic encodings and backbone features in the Panoptic Relational Attention. CosineSim and MatMul represent Cosine distance and matrix multiplication respectively.

As observed from Table B.2, we observe that matrix multiplication provides the best results and balances out the improvements in semantic, instance and panoptic segmentation metrics in all cases. Hence, we use this as our default method to compute correlations in the main paper. On the other hand, using cosine similarity distance for f_2 shows lower increase in PQ whereas using L2 norm distance for f_2 displays lower increase in mIoU. We hypothesize that cosine similarity fails to represent panoptic relationships and L2 norm induces errors in segmentation quality.

B.3. Varying the value of the Intermediate loss weight:

In Equation 1 of the main paper, we discuss the total training loss function. This includes intermediate losses L'_{sem} and L'_{ins} from the initial semantic and instance heads. These are weighted by the term γ. In this section, we discuss the effect of varying the value of γ after applying PISR to a Panoptic-DeepLab-ResNet50 base network. Table B.3 summarizes the results on Citysapes val. We find an optimal value of $\gamma = 0.5$ and fix it for all the experiments in the paper.

C. Additional Visual Results

In this section, we show further visual results from the ones shown in the paper. First, we show some qualitative examples on ADE20K. Next, we visualize the panoptic encodings of all the things and stuff classes in a scene.

Method	γ	PQ	mIoU	iIoU	Δ
Panoptic DeepLab	-	59.9	78.5	62.5	0
Panoptic DeepLab + PISR	0.0	61.9	79.5	63.4	+3.9
	0.2	61.9	79.9	63.7	+4.6
	0.5	**62.2**	**80.2**	**64.4**	**+5.9**
	0.8	61.5	79.3	63.5	+3.6

Table B.3. **On Cityscapes val:** Effect of varying the intermediate loss weight γ. We find an optimal value at 0.5 and keep it fixed for rest of the experiments.

C.1. Qualitative Examples on ADE20K

In Figure C.1, we provide more results on ADE20K for qualitative comparison with the base models. These are an extension to Figure 7. The base model used is Maskformer-R50 and we train it with and without PISR. As observed from the images, training the base model with PISR shows clear visual improvements.

C.2. Visualizing Semantic and Instance Encodings

We show t-SNE projections of the learned semantic and panoptic encodings in Figure C.2. These encodings are from a Panoptic-DeepLab-R50 model trained with PISR. To compare their separation, we visualize both semantic and instance encodings in the same feature space. Note that these are heavy approximations as the encoding space is 256-dimensional. It can be seen that both the semantic classes and instances are separated. While instances of the same class are closer to each other than to those of other classes, they are still properly separated and hence, provide more information about the scene than simply using semantic encodings.

D. Implementation Details

In this section, we describe our implementation details for each experiment in Section 4 of the paper. All our models are trained using Pytorch on 4 Nvidia Tesla-A100/Tesla-V100 GPUs.

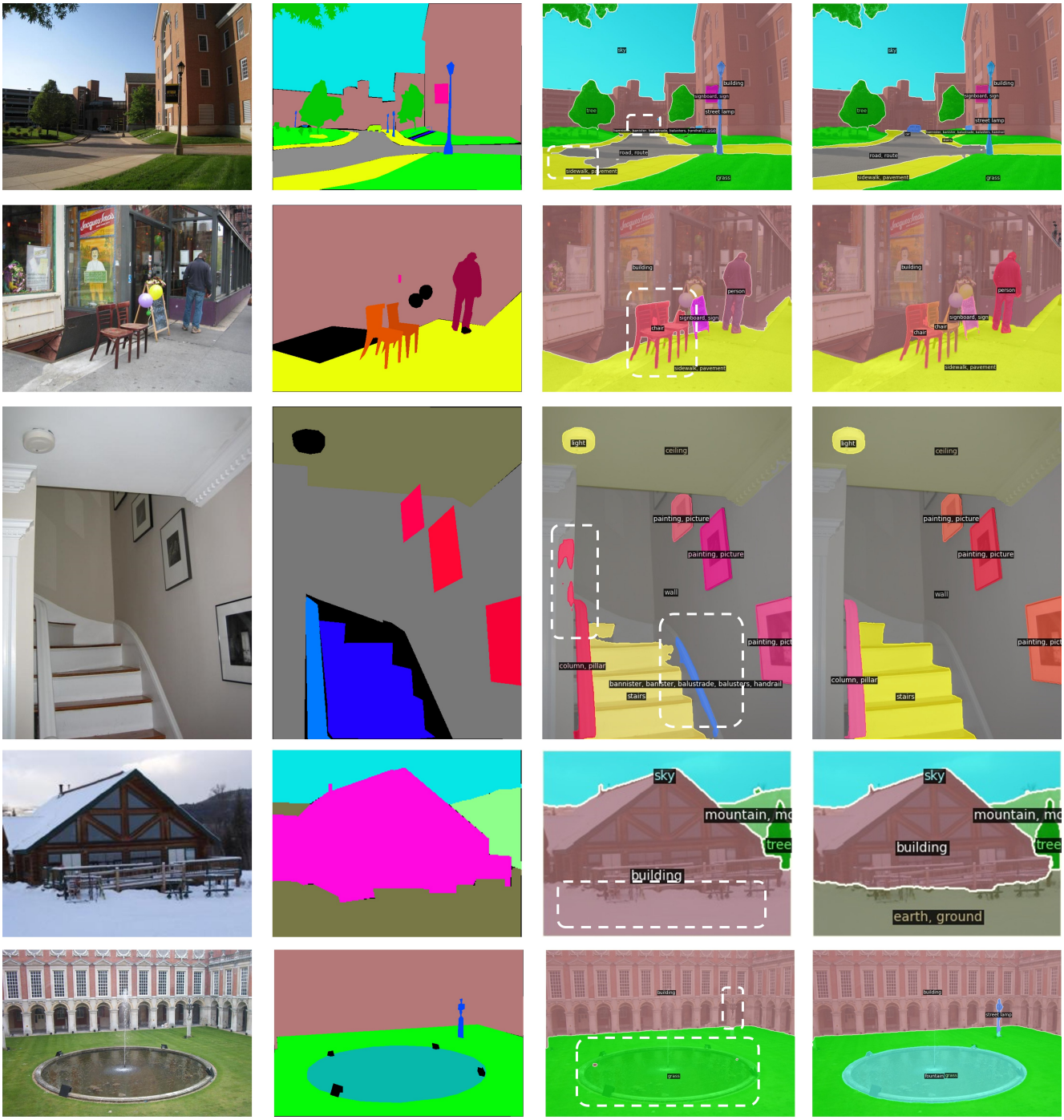
D.1. Cityscapes

Panoptic Deeplab: We use the pytorch version of Panoptic Deeplab open-source implementation to produce baseline results with ResNet50, ResNet101[1] and HRNet[2] backbones. **We use the default hyper-parameters for reproducing both baseline and PISR networks.** During training, we utilize the standard random scale jittering between 0.5 and 2.0 for data augmentation and scale back the images to 1024×2048 for equal size with batch size 32. The networks are trained with the default 'poly' learning rate policy with an initial learning rate of 0.001, and exponent 0.9 for 90k iterations. For the PISR experiment, we add the PISR block to Panoptic-DeepLab after the centers and semantic segmentation prediction. We apply the method described in Section 3 to generate panoptic features, and feed these shared features into semantic and instance segmentation heads. The training loss is obtained from Equation 1 in the main paper where the loss weights for auxiliary loss is $\gamma = 0.5$. The auxiliary loss term is considered as the output of the first prediction from the backbone architecture, and the final loss is applied after our PISR module. We use 4 Nvidia Tesla-A100 GPUs to train each of these networks.

D.2. COCO

Panoptic Deeplab: We use the same Panoptic Deeplab open-source implementation to produce baseline results with ResNet50[1] and HRNet[2] backbones. **We use the default hyper-parameters for reproducing both baseline and PISR networks.** During training, we utilize the standard random cropping for data augmentation to 640×640. The training batch size is 64. The networks are trained with the default 'poly' learning rate policy with an initial learning rate of 0.0005, and exponent 0.9 for 200k iterations. For the PISR experiment, we add the PISR block to Panoptic-DeepLab after the centers and semantic segmentation prediction. We apply the method described in Section 3 to generate panoptic features, and feed these shared features into semantic and instance segmentation heads. The training loss is obtained from Equation 1 in the main paper where the loss weights for auxiliary loss is $\gamma = 0.5$. The auxiliary loss term is considered as the output of the

[1]https://github.com/facebookresearch/detectron2
[2]https://github.com/bowenc0221/panoptic-deeplab

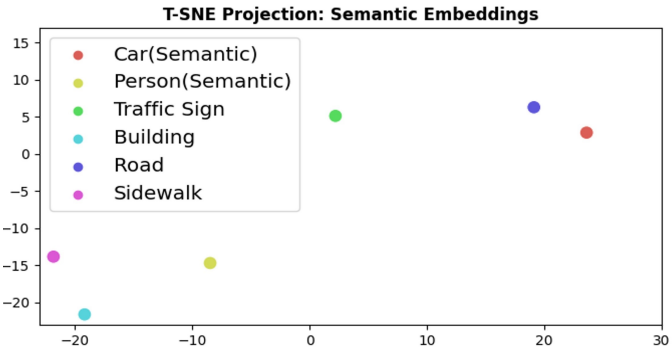


(a) Input (b) Ground truth (c) Base model (d) Base model + PISR

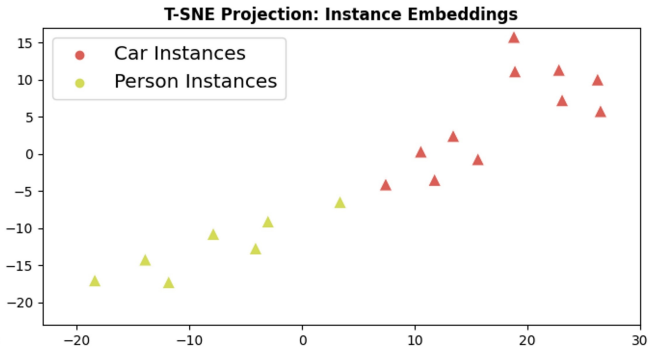
Figure C.1. Qualitative results **on ADE20K:** (a) Input images. (b) Ground truth masks. (c) Predictions by MaskFormer (ResNet-50). (d) Our results by applying PISR to MaskFormer (ResNet-50). After using PISR, the overall panoptic segmentation quality improves. Dashed boxes highlight sample regions where PISR significantly enhances the baseline prediction.

first prediction from the backbone architecture, and the final loss is applied after our PISR module. We used either 4 Nvidia Tesla-V100 or Tesla-A100 GPUs to train each of these networks.

(a) Panoptic Segmentation result using Panoptic-DeepLab-R50 with PISR



(b) t-SNE projection of Semantic Embeddings

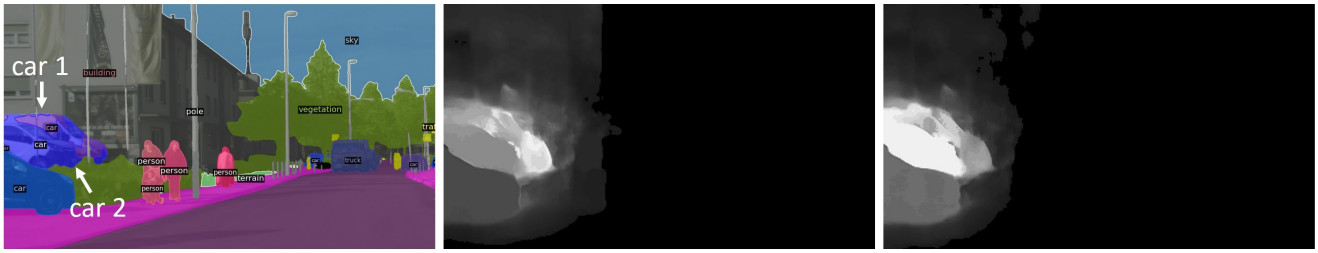


(c) t-SNE projection of Instance Embeddings

Figure C.2. Visualization **on Cityscapes:** (a) Our prediction results (b) Our t-SNE projections of the learned semantic encodings on a 2D feature space. (c) Our t-SNE projections of the learned instance encodings are plotted on the same feature-space. Even though the instances share the same class, they are well separated and provide more information about the scene than simply using semantic encodings.

Panoptic-FPN: We use the official Panoptic-FPN open-source implementation to produce baseline results with ResNet50[1]. **We use the default hyper-parameters for reproducing both baseline and PISR networks.** During training, we utilize the original "1x learning scheme" used for the results in their paper. The training batch size is 16. The networks are trained with the default multi-step learning rate policy with an initial learning rate of 0.02 for 90k iterations. For the PISR experiment, we add the PISR block to Panoptic-FPN after the mask proposal and semantic segmentation prediction. We use the method described in Section 3 on the predicted instance masks and semantic predictions to generate panoptic features, and feed these shared features into semantic and instance segmentation heads. The training loss is obtained from Equation 1 in the main paper where the loss weights for auxiliary loss is $\gamma = 0.5$. The auxiliary loss term is considered as the output of the first prediction from the backbone architecture, and the final loss is applied after our PISR module. We used either 4 Nvidia Tesla-V100 or Tesla-A100 GPUs to train each of these networks.

UPerNet: We use a modified version of the open-source implementation of UPerNet to produce baseline results with Swin-L. We train the baseline model with both semantic and instance segmentation heads and perform fusion in the post-processing stage similar to Panoptic-FPN. During training, we utilize the standard random flipping and normalization for data augmentation. The training batch size is 8. The networks are trained with the 'step' learning rate policy with an initial



(a) Prediction without PISR (b) Car 1 instance heatmap without PISR (c) Car 2 instance heatmap without PISR

(d) Prediction with PISR (e) Car 1 instance heatmap with PISR (f) Car 2 instance heatmap with PISR

Figure E.1. Instance heatmaps **on Cityscapes:**. (a) Prediction without PISR (Panoptic-DeepLab-R50). (b) Car 1 instance heatmap without PISR. (c) Car 2 instance heatmap without PISR. (d) Prediction with PISR (Panoptic-DeepLab-R50). (e) Car 1 instance heatmap with PISR. (f) Car 2 instance heatmap with PISR. As observed from the highlighted region and instance heatmaps, both the highlighted cars are well separated after training with the PISR module.

learning rate of 0.0001 for 36 epochs.

For the PISR experiment, we add the PISR block to UPerNet after the mask proposal and semantic segmentation prediction. We apply the method described in Section 3 to generate panoptic features, and feed these shared features into semantic and instance segmentation heads. The training loss is obtained from Equation 1 in the main paper where the loss weights for auxiliary loss is $\gamma = 0.5$. The auxiliary loss term is considered as the output of the first prediction from the backbone architecture, and the final loss is applied after our PISR module. We used 4 Tesla-A100 GPUs to train each of these networks.

D.3. ADE20K

Maskformer: We use the official Maskformer open-source implementation to produce baseline results with ResNet50 and ResNet101[3] backbones. **We use the default hyper-parameters for reproducing both baseline and PISR networks.** During training, we utilize the standard cropping for data augmentation to 640×640. The training batch size is 16. The networks are trained with the default 'poly' learning rate policy with an initial learning rate of 0.0005, and exponent 0.9 for 720k iterations. However, we use 4 Nvidia Tesla-A100 GPUs as compared to the 8 GPUs used to produce their baseline results. For the PISR experiment, we use the mask predictions generated by the MaskFormer network to generate panoptic encodings E_{pan}, and then generate the panoptic features as described in Section 3. We feed these features into the MLP which produces mask predictions and obtain the final panoptic segmentation. The training loss is obtained from Equation 1 in the main paper where the loss weights for auxiliary loss is $\gamma = 0.5$. The semantic and instance loss terms are considered together as their mask loss. The auxiliary loss term is considered as the output of the first prediction from the MaskFormer architecture, and the final loss is applied after our PISR module.

E. Further Details on Visualizing Semantic, Center and Instance Heatmaps

In this section, we describe our detailed method to generate the overlaid Semantic and Center heatmaps in Figure 5 of the main paper. We also show a visualization of the intermediate instance heatmaps, which are used to generate instance encodings as described in Section 3.2 of the main paper.

[3]https://github.com/facebookresearch/MaskFormer

E.1. Detailed Explanation for Producing Semantic and Center Heatmaps

We generate semantic and center heatmaps for Figure 4 and Figure 5. For the semantic heatmap, we use outputs from the semantic generators both before and after the PISR block. The output of each semantic generator is $F_s \in \mathbb{R}^{C \times H \times W}$. Here, C, H, and W indicate the number of defined classes, height and width, respectively. From F_s, we extract only the target class feature map and apply normalization with the min and max value of the feature map. Therefore, when the region does not belong to the target class, it has a lower value, making the color of the heatmap close to blue; this represents regions which are suppressed. On the other hand, regions detected as the target class are in red, and the intensity of red color indicates the confidence of prediction.

For visualizing instance centers, we use the predicted center heatmap after the non-maximum suppression step. We multiply the center map with the target segmentation mask to suppress non-target centers. Finally, we apply a Gaussian filter to help enunciate centers better.

The above semantic heatmaps and centers are overlaid on top of each other in Figure 5.

E.2. Visualizing Instance Heatmaps

In Section 3.2 of the main paper, we describe our method to generate instance heatmaps. In Figure E.1, we visualize the intermediate instance heatmaps for an image from Cityscapes dataset. We use the Panoptic-DeepLab-R50 network as the base architecture, and observe the instance heatmaps with and without training with the PISR block. As observed from the highlighted sections in Figure E.1 (a), the base network fails to segment two cars accurately in the scene. The network fails to separate these instances as observed from the instance heatmaps for each car, in Figure E.1 (b) and (c). However, we observe that the network trained with PISR is able to clearly separate the two cars, as seen from both the instance heatmaps as well as the final prediction shown in Figure E.1 (d).