SeeThroughNet: Resurrection of Auxiliary Loss by Preserving Class Probability Information

Dasol Han*, Jaewook Yoo*, Dokwan Oh†
Samsung Advanced Institute of Technology
{dasol.han,jaewook.yoo,dokwan.oh}@samsung.com

Abstract

Auxiliary loss is additional loss besides the main branch loss to help optimize the learning process of neural networks. In order to calculate the auxiliary loss between the feature maps of intermediate layers and the ground truth in the field of semantic segmentation, the size of each feature map must match the ground truth. In all studies using the auxiliary losses with the segmentation models, from what we have investigated, they either use a down-sampling function to reduce the size of the ground truth or use an up-sampling function to increase the size of the feature map in order to match the resolution between the feature map and the ground truth. However, in the process of selecting representative values through down-sampling and up-sampling, information loss is inevitable. In this paper, we introduce Class Probability Preserving (CPP) pooling to alleviate information loss in down-sampling the ground truth in semantic segmentation tasks. We demonstrated the superiority of the proposed method on Cityscapes, Pascal VOC, Pascal Context, and NYU-Depth-v2 datasets by using CPP pooling with auxiliary losses based on seven popular segmentation models. In addition, we propose See-Through Network (SeeThroughNet) that adopts an improved multi-scale attention-coupled decoder structure to maximize the effect of CPP pooling. SeeThroughNet shows cutting-edge results in the field of semantic understanding of urban street scenes, which ranked #1 on the Cityscapes benchmark.

1. Introduction

Increasing the depth of the convolutional neural network can introduce optimization difficulties as shown in [6, 24]. To solve this vanishing gradient problem in the field of image classification, directly connected auxiliary classifiers were used to provide extra gradients into some intermediate layers. The auxiliary classifiers have been widely used in the variations of the Inception architecture of GoogLeNet [38] and ResNet [18], where the motivation was to provide useful gradients directly to intermediate layers as extra supervision during training to resist the vanishing gradient problem in very deep networks.

Similar attempts have been made in the field of semantic segmentation. For example, [53, 55, 56] used ResNet-101 as a backbone and adopted an auxiliary head in the intermediate layer as specified in [53]. Since the feature map of the auxiliary head connected layer and ground truth are used to calculate the auxiliary loss, their resolutions should be the same. The auxiliary head is responsible for re-sizing the resolution of the intermediate feature map to that of the ground truth through a re-sizing operation.

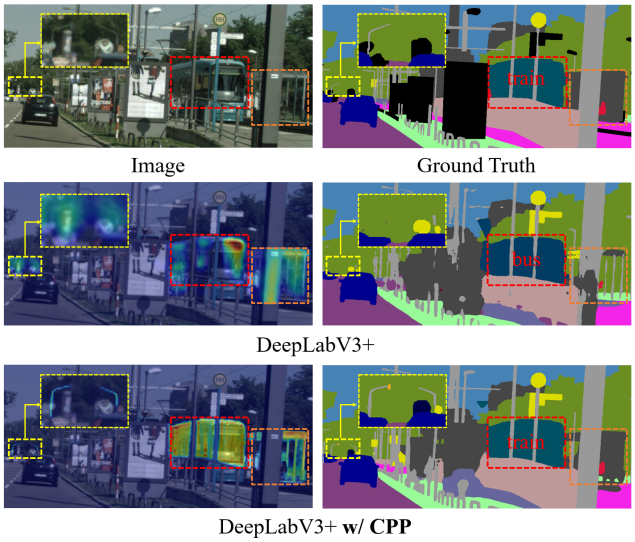


Figure 1. Visualizations of the effect of Class Probability Preserving (CPP) pooling. The images are from Cityscapes. For the occluded train (in the red box), the occluded building (the bus stop in the orange box), and the poles in the distance (in the yellow box), the DeepLabV3+ [5] model w/ CPP shows finer and stronger activation in the Grad-CAM [34] visualization (the left column) than the DeepLabV3+ original model. In the inference results (the right column), the w/ CPP model shows correct and fine segmentation. The enlarged views are provided (the yellow box).

*Indicates equal contribution
†Corresponding author

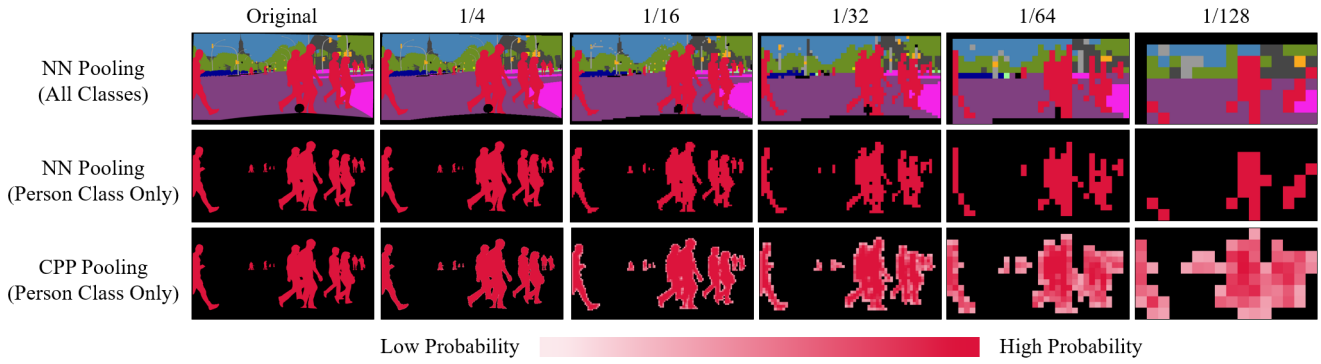


Figure 2. Visualization of the effects of the conventional pooling (i.e. nearest neighbor [NN] pooling) and CPP pooling. NN pooling has been widely used for down-sampling ground truth for segmentation task since the ground truth ids are integer. A semantic ground truth from the Cityscapes dataset is used. As the scale factor of the pooling increases, more information is lost. Especially, in NN pooling, far object and boundary information are lost a lot, while they still remain as a probability in CPP pooling.

However, designing the auxiliary loss with the re-sizing operations has an inevitable flaw. Each layer in deep neural networks learns different data-driven features. For example, closer layers to the input image learn simpler low-dimensional features. As one moves away from the input layer, higher-order complex features are learned. If we re-size the ground truth resulting in one-dimensional pixel-wise class information to get the auxiliary loss for the intermediate layers, it would not be enough to guide the various-order features of each layer. For this reason, researchers in this field decorate these re-sizing guided losses with the word *auxiliary* and use them as auxiliary solutions rather than complete solutions. For the same reason, the majority of scene segmentation papers [5, 14, 17, 29, 46, 48] that have been published recently do not aggressively use auxiliary loss (mostly using no or one auxiliary head).

Motivated by this limitation, we introduce Class Probability Preserving (CPP) pooling to provide well-designed auxiliary guides, which help optimize the training process of the networks. CPP pooling keeps the class information within a receptive field as a probability preventing the information loss (Figure 3). Through the comprehensive experiments with seven popular semantic segmentation models on various datasets such as Cityscapes [9], Pascal VOC [11], Pascal Context [26] and NYU-Depth-v2 [28], we demonstrated that the proposed method is a model agnostic method that is simply applicable to any model but effective. In addition, we introduce See-Through Network (SeeThroughNet) with an improved multi-scale attention-coupled decoder structure to maximize the effect of CPP pooling. SeeThroughNet shows state-of-the-art results in the field of semantic understanding of high-resolution urban street scenes. The results of ablation and transfer learning experiments are also provided.

The main contributions of this paper are:

- We introduce Class Probability Preserving (CPP) pooling to alleviate information loss, where it keeps the class information within a receptive field as a probability.
- We demonstrated seven popular semantic segmentation models with CPP pooling on Cityscapes, Pascal VOC, Pascal Context, and NYU-Depth-v2 datasets to show the effect of the proposed method.
- We propose SeeThroughNet with multi-scale attention-coupled decoder maximizing the effect of CPP pooling, and achieved the first place on the Cityscapes leaderboard.

2. Related Work

Pooling Methods. Variants of pooling methods have been proposed with convolutional neural networks, which can be categorized into four groups following [1]. Value-based pooling methods [10, 21, 30, 37, 47] pick a single representative value based on the significance of the values in the region for down-sampling. Max and average pooling are widely used ones in this category. Patch or multi-sampling approach has been proposed to complement the information loss of the value-based pooling methods, where values in patches or in multi-scale sub-maps are processed. Probability-based pooling methods [16, 22, 43, 45, 51] evaluate the probability for the representative values in the region, which helps to prevent overfitting. Various approaches based on probability have been proposed, including stochastic, dropout max, failure density, hybrid, and mixed gated pooling methods. In rank-based pooling methods [19, 20, 35], weights are learned during training, which

are used as a weighted sum in the pooling region. Multipartite, ordinal, and global weighted rank approaches have been demonstrated, in which the multipartite feature ranking in pooling layers, different weights to all feature activations, and estimated scores associated with a class were used, respectively. For transformed domain pooling methods, [44] discarded the first-level and only used the second-level wavelet decomposition to reduce the feature dimensions, which resulted in fewer artifacts. [32] truncated the lower frequencies in power spectrum to reduce the dimensions that preserved more information per parameter than other pooling methods.

These pooling methods used various approaches to decide representative values in the corresponding receptive fields for down-sampling. However, to the best of our knowledge, none of the pooling methods used an approach preserving class information as a probability.

Label Smoothing. CPP pooling could be seen as a variation of the label smoothing proposed in [39]. Compared to the label smoothing, our method uses it for down-sampling process and follows the class distribution of the pixels in the receptive field, instead of using a predefined distribution such as the uniform distribution used in [39]. Therefore, CPP pooling is an improved version in that the class probability distribution for each pixel follows data adaptively.

Semantic Segmentation Architectures. Fully Convolutional Networks (FCNs) [25] have demonstrated significant improvement adopting deep convolutional neural networks (CNNs) in semantic segmentation, by replacing fully connected layers (FCs) with convolutional layers at the end of CNN architectures. Based on this improvement, other advanced techniques such as skip-connections in encoder-decoder architecture [2, 5, 33], a pyramid pooling module [53], and an atrous convolution with atrous spatial pyramid pooling (ASPP) [4] have shown remarkable improvements.

Recent works have focused on relational context to improve performance, not multi-scale context. [12,49,52] considered the similarity between pixels using a self-attention method and performs a similarity-weighted aggregation. Boundary-aware methods also have been applied to improve semantic segmentation performance. For example, Gated-SCNN [40] used the shape stream network that takes image gradients and features to produce semantic boundaries as output. It used a pixel-based loss to train the shape stream network. InverseForm [3] employed the inverse-transformation network to model the spatial distance-based metric into the loss function. [41, 48, 53] used the auxiliary losses with the up-sampled logits instead of the conventional pooling methods to mitigate the information loss.

In this work, we show that when using auxiliary losses, the performance is further improved by using the proposed CPP pooling instead of using the conventional up-sampling or down-sampling methods.

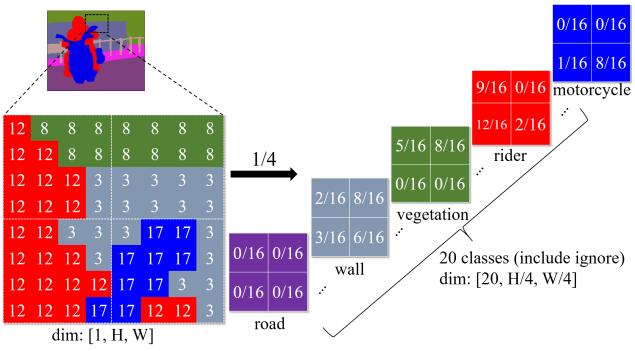


Figure 3. Illustration of Class Probability Preserving (CPP) pooling. It shows 1/4 pooling where the 4x4 grids information is down-sampled to one grid. The number of channels of the output equals to the total number of classes (e.g. 20 classes for Cityscapes including the ignore class), where each channel represents the class probability of the corresponding grid.

3. Method

3.1. Class Probability Preserving Pooling

Conventional pooling methods (e.g, max-pooling, average-pooling, nearest neighbor pooling, etc.) have been commonly used to down-sample features and ground truth. However, this potentially degrade performance for semantic segmentation task due to the information loss when selecting the representative values from the corresponding receptive regions. The top row of Figure 2 shows an example of information loss from down-sampling, where as the scale factor of the pooling increases, more information loss is observed to down-sample the semantic ground truth.

We propose Class Probability Preserving (CPP) pooling, a novel pooling method preserving class information minimizing information loss (Figure 3). Instead of deciding one representative value from the corresponding receptive region in the input map, CPP pooling counts the number of grids (or pixels) that match each class in the region, then use the number as a probability for each class. For example, Figure 3 shows 1/4 CPP pooling where 4x4 grids are down-sampled to one grid. In the input map, the number of class 0 (road) in the 4x4 region is counted, which becomes the probability value of the corresponding grid of the output channel of the class 0 (road). The same calculations are processed for the other classes (i.e. total 20 classes for Cityscapes including the ignore class). As shown, each location in the output grid keeps a certain class probability.

In more details, in CPP pooling, a grid of the output $Y'_{k_{(l,m)}}$ of a class k is computed as:

$$Y'_{k_{(l,m)}} = \frac{1}{s^2}\sum_{(i,j)} \mu_{i,j}, \text{ with } \mu_{i,j} = \begin{cases} 1 & \text{if } Y_{(i,j)} = k \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

, where $Y_{(i,j)}$ is the grids (or pixels) of the input feature

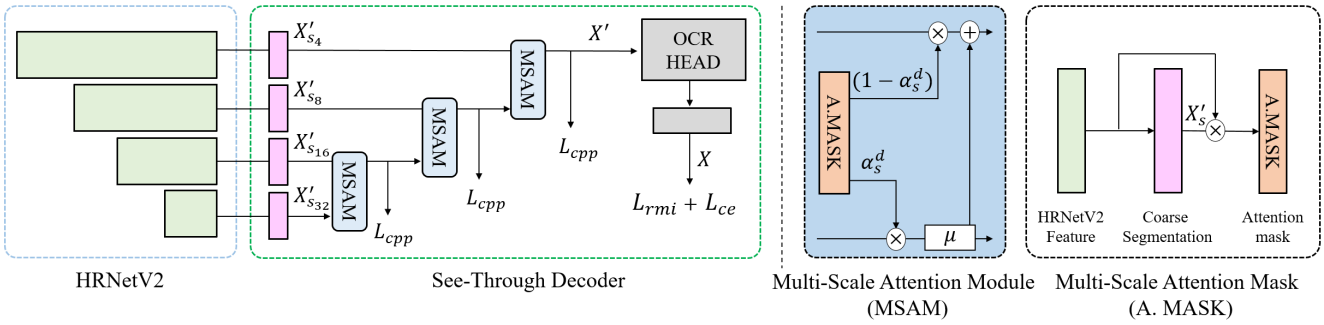


Figure 4. **SeeThroughNet Architecture.** The architecture consists of CPP pooling with 3 auxiliary heads combined with multi-scale attention. To make better use of CPP pooling, See-Through Decoder was designed to include the multi-scale attention that allows CPP pooling to be used for each scale. See Section 3.2 for detail.

map, $k \in K$ is the object class, Y is the input feature map, and $Y^{'}$ is the output of the CPP pooling. $Y^{'}_{k_{(l,m)}}$ is the grid (l, m) of the class k channel map in the output $Y^{'}$. Given a scale factor $1/s$, the range of $Y_{(i,j)}$ are $s \times l \leq i < s \times (l+1)$ and $s \times m \leq j < s \times (m+1)$ for the corresponding area of the input feature map. Finally, to compute the output grid value $Y^{'}_{k_{(l,m)}}$, we compute each class k probability by counting the matching class at target grids (or pixels), then normalize it by s^2. After down-sampling semantic segmentation labels by CPP pooling, we get two probability distributions (i.e. CPP pooling result from the ground truth and the semantic prediction output from the layer). To measure the difference between two probability distributions, we use Kullback-Leibler (KL) divergence [13], written as D_{KL} in Equation 2. Let $p_i(x)$ and $q_i(x)$ are the semantic prediction output of the layer and the CPP pooling result from the semantic ground truth, respectively, the loss function is:

$$L_{cpp} = \frac{1}{H \times W} \sum_{h,w \in H,W} D_{KL}(p(x)||q(x)) \quad (2)$$

, where the KL divergence result is normalized by all element $H \times W$; H for the height of $p_i(x)$ and W for the width of $p_i(x)$.

3.2. SeeThroughNet

To adopt the benefit of CPP pooling for semantic segmentation task, we propose our state-of-the-art semantic segmentation network called SeeThroughNet (Figure 4). We dubbed Seethrough Network since the class probability information is *seen through* the CPP output channels (Figure 3), while only a representative class is shown with the hard labels of the existing down-sampling methods.

Backbone. We use HRNetV2-W48 [36] as a backbone, where the outputs of the final stage with four different scale features are used for the SeeThrough decoder.

SeeThrough Decoder. The coarse semantic predictions $X^{'}_{S}$ (the pink boxes in Figure 4) are created from each of four different scales $S = [s_{32}, s_{16}, s_8, s_4]$. The coarse semantic segmentation head consists of (3x3 conv) $\rightarrow$ (BN) $\rightarrow$ (ReLU) $\rightarrow$ (1x1 conv). We use three auxiliary losses, which take the label $Y^{'}$ generated by CPP pooling. Thanks to the class probability preserved labels, the decoder learns potentially existing information from the larger scale image that disappears when down-sampling is used instead.

To create larger and fine semantic segmentation prediction, we sequentially combine from the smaller scale semantic map to the larger scale semantic map. When combining the two semantic segmentation predictions with different scales, we use the trainable relative attention mask α_s^d in the *d*ecoder between the adjacent scale semantic segmentation maps, and the bilinear up-sampling operation μ on the smaller map to match resolution (Multi-Scale Attention Module [the blue boxes] in Figure 4). [41] introduced the related attention mask, which was originally used to combine multi-scale predictions at training and inference phase but not in the decoder. To generate the attention mask for each scale in the SeeThrough decoder, we use the same scale of the backbone features and coarse segmentation map (Multi-Scale Attention Mask in Figure 4). The multi-scale coarse segmentations $X^{'}_{S}$ with $S = [s_{32}, s_{16}, s_8, s_4]$ are aggregated sequentially as in Equation 3 to get the fine segmentation feature $X^{'}$.

$$X^{'} = \sum_{s \in S} (\mu(\alpha_s^d * X^{'}_s) + ((1 - \mu(\alpha_s^d)) * X^{'}_{s+1})) \quad (3)$$

, where $*$ and $+$ are pixel-wise multiplication and addition.

OCR Head. We adopt OCR Head [48] to predict the final semantic segmentation output X. It takes the multi-scale aggregated segmentation prediction $X^{'}$ from the SeeThrough decoder and the concatenated features of the backbone's final stage outputs.

Two-resolution training. For training, a given input image is resized by factor *r*, where *r*=0.5 means down-sampled by factor 2. As introduced in [41], we follow the scale factor *r* as *r*=0.5 and *r*=1.0 for the two-resolution training. For each scale factor, the output of the SeeThrough decoder are combined as follows. To get the final aggregated semantic segmentation prediction X_{agg} from the results of the

Table 1. The results of the CPP pooling experiments with the seven segmentation models on the various validation datasets such as Cityscapes (trained with fine annotation only), Pascal VOC 2012, Pascal Context, and NYU-V2. All models with CPP pooling (w/ CPP) show performance improvement over the Baseline models.

Model	Cityscapes (mIoU %)			PASCAL VOC (mIoU %)			PASCAL Context (mIoU %)			NYU-V2 (mIoU %)		
	Baseline	w/ CPP	Gain	Baseline	w/ CPP	Gain	Baseline	w/ CPP	Gain	Baseline	w/ CPP	Gain
OCRNet [48]	81.17	**81.79**	**+0.62**	77.43	**78.43**	**+1.00**	45.96	**46.72**	**+0.76**	58.09	**58.59**	**+0.50**
DeepLabV3+ [5]	79.77	**81.07**	**+1.30**	78.35	**79.89**	**+1.54**	45.69	**48.23**	**+2.54**	54.66	**55.71**	**+1.05**
UPerNet [46]	79.13	**79.59**	**+0.46**	76.32	**78.95**	**+2.63**	44.13	**46.89**	**+2.76**	53.60	**55.22**	**+1.62**
DMNet [14]	78.15	**79.69**	**+1.54**	78.19	**78.94**	**+0.75**	45.36	**47.95**	**+2.59**	54.08	**55.07**	**+0.99**
CCNet [17]	77.47	**79.64**	**+2.17**	76.90	**79.51**	**+2.61**	43.76	**46.24**	**+2.48**	53.63	**54.10**	**+0.47**
PSPNet [53]	78.34	**79.85**	**+1.51**	77.44	**79.56**	**+2.12**	45.11	**47.77**	**+2.66**	54.51	**55.14**	**+0.63**
ANN [56]	78.31	**78.93**	**+0.62**	77.45	**78.59**	**+1.14**	44.42	**47.19**	**+2.77**	54.50	**55.44**	**+0.94**

SeeThrough decoder from the two resolutions, we merge $X_{0.5}$ and $X_{1.0}$ with the trainable attention mask $\alpha^m_{0.5}$ as written in Equation 4.

$$X_{agg} = \mu(\alpha^m_{0.5} * X_{0.5}) + ((1 - \mu(\alpha^m_{0.5})) * X_{1.0}) \quad (4)$$

Overall objective function. To compute the loss of the final semantic segmentation prediction X and the label Y, we use RMI loss L_{rmi} following [54] as Equation 5. L_{ce} is the cross entropy loss. I represents the mutual information, b is the batch size, and c is the class. For λ, we use 0.5 as in [54].

$$L_{rmi}(X, Y) = \lambda L_{ce} + (1 - \lambda)\frac{1}{B}\sum_{b}^{B}\sum_{c}^{C}(-I^{b,c}_{l}(X : Y)) \quad (5)$$

Additionally, we compute the auxiliary loss (Aux. w/ CPP in Figure 4) in the SeeThrough decoder with the three different scales ($X^{'}_{s_{16}}, X^{'}_{s_8}$, and $X^{'}_{s_4}$), which are sequentially aggregated to get $X^{'}$. We use KL divergence (Equation 2) to get the auxiliary loss $L_{decoder}$ in Equation 6.

$$L_{decoder}(X, Y) = \sum_{s \in S} L_{cpp}(X_s, Y^{'}_{s}) \quad (6)$$

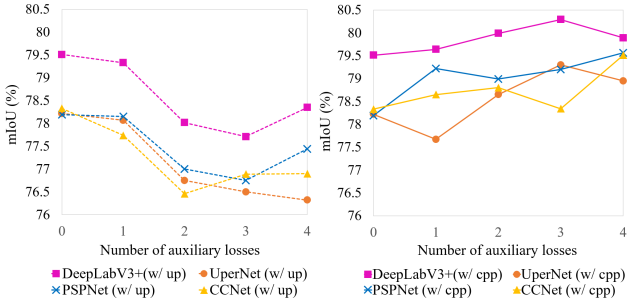
, where $S = [s_{16}, s_8, s_4]$. Finally, the overall objective function used of SeeThroughNet is:

$$L_{overall} = L_{rmi}(X, Y) + \sum_{r}^{R}\sum_{s}^{S} L_{decoder}(X_{r,s}, Y) + \gamma L_{ce}(X_{1.0}, Y) + \gamma L_{ce}(\mu(X_{0.5}), Y) \quad (7)$$

, where $R = [r_{0.5}, r_{1.0}]$, $S = [s_{16}, s_8, s_4]$, and γ is the constant (0.05) as in [41].

4. Experimental Results

In this section, we present experimental results to validate the proposed CPP pooling and SeeThroughNet. To see the effect of the CPP pooling, we experimented with seven popular semantic segmentation models on Cityscapes, Pascal VOC, Pascal Context, and NYU-Depth-v2 datasets. We evaluated SeeThroughNet on the Cityscapes dataset. Implementation details can be found in the Appendix.



(a) Aux. losses with up-sampling (b) Aux. losses with CPP pooling

Figure 5. The results of the ablation experiments with different numbers of the auxiliary losses (w/ up-sampling and w/ CPP) on the Pascal VOC 2012 val set. The number of auxiliary losses are 0, 1, 2, 3, 4 for both conditions. (Enlarged view is recommended.)

4.1. Effect of CPP pooling

Auxiliary heads. To apply CPP pooling on the seven conventional semantic segmentation models, we employed auxiliary heads on the backbone. For the HRNetV2 backbone of OCRNet, we added four auxiliary heads to the four different scales of the final outputs. For the ResNet101 backbone of the other six models, we added four auxiliary heads to the final outputs of each stage. The auxiliary head consists of (3x3 conv) $\rightarrow$ (BN) $\rightarrow$ (ReLU) $\rightarrow$ (1x1 conv).

CPP pooling performance for various models. Table 1 shows the results of the CPP pooling experiments with the conventional segmentation models on Cityscapes, Pascal VOC, Pascal Context, and NYU-Depth-v2 datasets. We used the models from MMSegmentation [8] and trained them from scratch. The models have the ResNet101 backbone except OCRNet. For *Baseline* models, we applied four auxiliary heads in the backbone with up-sampled logit with Cross Entropy (CE) loss as implemented in [8]. For *w/ CPP* models, we applied four auxiliary heads with CPP pooling with the KL loss (Equation 2). For all the models and datasets in Table 1, the models w/ CPP pooling outperform the baseline models. These results indicate that CPP pooling can be applied to various segmentation models to enhance performance.

Ablation experiment of CPP pooling. We conducted the ablation experiment of CPP pooling, where the number

Table 2. Class-wise comparison between the baseline models and the w/ CPP models on the Cityscapes validation set. The seven popular semantic segmentation models are compared. More improvement are observed with the instance classes (i.e. person, rider, car, truck, bus, train, motorcycle, and bicycle) than with the non-instance classes.

Method		Class IoU (%)																	
		Non-Instance											Instance						
		road	swalk	build	wall	fence	pole	tlight	tsign	veg	terrain	sky	person	rider	car	truck	bus	train	mcyc	bicyc
OCRNet [48]	Base	98.3	85.7	93.6	58.1	65.5	71.7	76.3	83.4	93.2	63.8	95.6	85.1	67.5	95.8	79.5	91.2	86.3	70.9	80.7
	CPP	98.5	87.2	93.7	59.4	65.3	72.4	76.8	83.6	93.2	64.3	95.6	84.7	66.5	96.1	86.2	92.5	87.1	70.2	80.8
	Diff.	0.2	1.5	0.1	1.3	-0.2	0.7	0.5	0.2	0.0	0.5	0.0	-0.4	-1.0	0.3	6.7	1.3	0.8	-0.7	0.1
	Diff. Avg.	0.44											**0.89**							
DeepLabV3+ [5]	Base	98.4	86.5	93.1	53.2	62.0	70.0	75.3	81.9	92.9	63.4	95.2	84.4	67.4	95.4	79.8	90.2	77.2	69.1	80.1
	CPP	98.4	86.3	93.2	51.4	63.3	70.0	75.6	82.0	93.0	64.0	95.4	84.8	67.6	96.1	87.0	92.6	87.1	72.0	80.7
	Diff.	0.0	-0.2	0.1	-1.8	1.3	0.0	0.3	0.1	0.1	0.6	0.2	0.4	0.2	0.7	7.2	2.4	9.9	2.9	0.6
	Diff. Avg.	0.06											**3.04**							
PSPNet [53]	Base	98.2	85.2	92.9	54.4	61.7	67.1	73.5	80.2	92.6	62.7	95.0	82.9	65.6	95.2	74.7	88.3	74.5	64.6	79.2
	CPP	98.2	85.6	93.0	54.2	61.3	66.9	73.8	80.7	92.8	65.3	95.1	83.3	65.2	95.7	84.7	90.4	83.4	68.1	79.7
	Diff.	0.0	0.4	0.1	-0.2	-0.4	-0.2	0.3	0.5	0.2	2.6	0.1	0.4	-0.4	0.5	10.0	2.1	8.9	3.5	0.5
	Diff. Avg.	0.31											**3.19**							
UPerNet [46]	Base	98.3	85.9	92.8	50.9	60.8	68.7	74.5	80.3	92.7	61.9	95.2	83.6	64.6	95.6	79.2	90.1	81.7	67.3	79.5
	CPP	98.3	86.3	92.8	51.3	59.8	68.1	73.7	80.7	92.8	63.0	94.9	83.4	65.0	95.6	83.7	90.7	81.3	71.3	79.6
	Diff.	0.0	0.4	0.0	0.4	-1.0	-0.6	-0.8	0.4	0.1	1.1	-0.3	-0.2	0.4	0.0	4.5	0.6	-0.4	4.0	0.1
	Diff. Avg.	-0.03											**1.13**							
CCNet [17]	Base	98.3	85.9	92.6	51.1	57.9	66.8	72.2	80.1	92.4	63.2	95.0	82.9	66.5	95.0	70.4	86.2	70.6	66.3	78.6
	CPP	98.3	86.6	92.9	52.1	60.3	66.8	73.8	80.2	92.6	63.6	95.1	83.9	67.2	95.8	83.7	91.9	81.4	67.5	79.4
	Diff.	0.0	0.7	0.3	1.0	2.4	0.0	1.6	0.1	0.2	0.4	0.1	1.0	0.7	0.8	13.3	5.7	10.8	1.2	0.8
	Diff. Avg.	0.62											**4.29**							
DMNet [14]	Base	98.3	85.8	92.9	51.1	59.3	67.1	73.8	80.6	92.6	63.0	94.9	83.3	65.1	95.6	75.5	87.0	72.3	67.7	79.4
	CPP	98.3	85.8	92.9	52.6	60.3	66.8	73.7	80.9	92.7	64.0	94.9	83.6	64.5	95.9	85.2	91.5	83.6	67.3	79.7
	Diff.	0.0	0.0	0.0	1.5	1.0	-0.3	-0.1	0.3	0.1	1.0	0.0	0.3	-0.6	0.3	9.7	4.5	11.3	-0.4	0.3
	Diff. Avg.	0.32											**3.18**							
ANN [56]	Base	98.1	84.9	92.7	53.3	60.1	66.8	72.9	80.0	92.5	62.4	95.1	82.9	63.8	95.6	80.3	87.2	72.3	68.2	79.0
	CPP	98.2	85.1	92.8	53.4	59.2	67.2	73.4	81.3	92.5	62.3	94.8	83.3	64.4	95.6	81.8	90.6	77.8	66.9	79.3
	Diff.	0.1	0.2	0.1	0.1	-0.9	0.4	0.5	1.3	0.0	-0.1	-0.3	0.4	0.6	0.0	1.5	3.4	5.5	-1.3	0.3
	Diff. Avg.	0.13											**1.30**							

Table 3. The distribution of the classes in the Cityscapes fine training dataset (2975 images). The table lists the number of images containing each class object in the training set. The images with the truck, bus, and train classes are significantly lacking compared to the other classes.

Class	road	swalk	build	wall	fence	pole	tlight	tsign	veg	terrain	sky	person	rider	car	truck	bus	train	mcyc	bicyc
Counts	2934	2811	2934	970	1296	2949	1658	2808	2891	1654	2686	2343	1023	2832	**359**	**274**	**142**	513	1646

of auxiliary heads for the models are varies from 0 to 4 with up-sampling (i.e. bilinear up-sampling) and with CPP pooling. As in Figure 5, the results show that adding the auxiliary heads with up-sampling does not help to improve the performance, and sometimes even degrade the performance compared to the model without the auxiliary head (0 Aux). On the other hand, adding auxiliary heads with CPP pooling improves performance over the models without the auxiliary head (0 Aux) and the auxiliary heads with up-sampling. Note that we believe that adding auxiliary heads with down-sampling degrades the performance further because more information loss occurs with down-sampling with the conventional methods than up-sampling.

This implicates why the recently proposed semantic segmentation architectures did not aggressively use auxiliary heads (mostly using one auxiliary head). Adding more auxiliary heads with the conventional down-sampling and up-sampling methods does not help the performance improvement because of the information loss.

Performance with the class distribution. We compared the class-wise results of the seven semantic segmentation models between the baseline and the w/ CPP models. As shown in Table 2, the instance classes, such as the person, rider, car, truck, bus, train, motorcycle, and bicycle classes, generally show more improvement than the non-instance classes. We speculate that CPP pooling makes the model more robust for occluded and long-distance objects that often appear with instance classes. The Grad-CAM [34] visualizations of the distant and occluded objects are in Figure 6 for qualitative analysis.

In addition, we observed that the truck, bus, and train classes show especially more improvement than the other

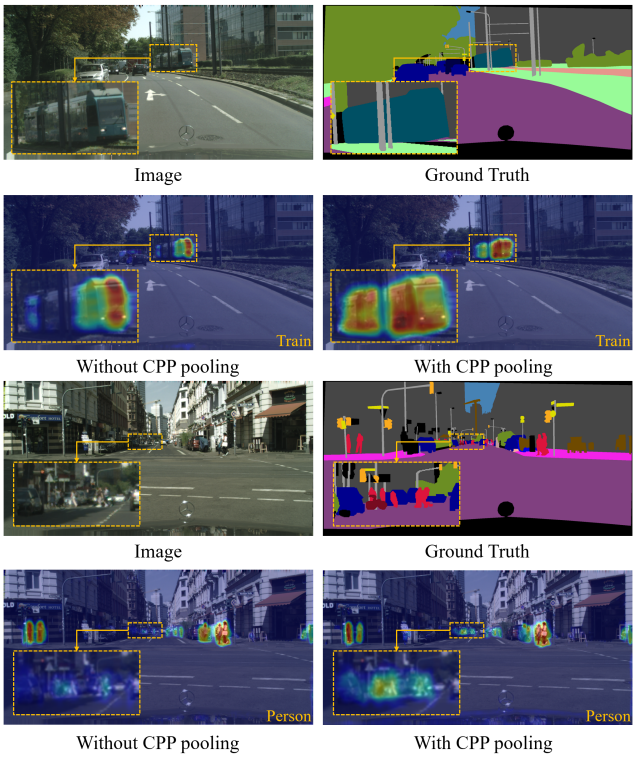


Figure 6. Grad-CAM [34] visualizations of the train (top) and person (bottom) class. The occluded and distant objects (in the orange boxes) show stronger activation with CPP pooling than the baseline (without CPP pooling).

classes for all results from the seven semantic segmentation models. This follows the class distribution in the Cityscapes fine training dataset. We counted the images containing each class object from the fine training dataset (2975 images), and list them in Table 3. The images with the truck, bus, and train classes are markedly lacking compared to the other classes. This implicates that CPP pooling also could address the class imbalance problem. The potential reason for dealing with the class imbalance problem is that such soft labels prevent the network from over-confidence by providing noise distribution [27]. Thus the soft labels generated by CPP pooling encourage higher probability predictions for minority classes. This performance improvement for the scarce classes was also observed in the SeeThroughNet experiment. After adding the Mapillary dataset, the SeeThroughNet model showed 4% improvement in the train class, even if the Mapillary dataset had no train class label.

4.2. SeeThroughNet Results on Cityscapes

SeeThroughNet ablation study. To see the effect of CPP pooling on SeeThroughNet, we conducted the ablation experiments with the SeeThrough Decoder. In Table 4, both w/ CPP pooling and w/ multi-scale (MS) attention show improvement over the baseline, where w/ CPP pooling enhanced more. A synergistic effect was observed when CPP pooling and MS attention were used together. The qualitative results are shown in the Appendix.

Table 4. SeeThroughNet ablation study. Trained and evaluated on the Cityscapes fine and validation set. CPP pooling is our proposed method. MS attention indicates multi-scale attention mask used to combine semantic map in the Seethrough decoder.

CPP pooling	MS Attention	IOU (%)	Gain
		84.06	
✓		84.69	0.63
	✓	84.37	0.31
✓	✓	85.29	1.23

Table 5. SeeThroughNet feature transfer learning for object detection task. The SeeThroughNet encoder (HRNetV2-48) trained on Cityscapes were transferred to the detection models. The three detection models loaded the encoder weight and trained on the detection data with and without freezing the encoder. The detection data labels were parsed from the Cityscapes instance dataset to utilize the SeeThroughNet encoder weight trained on the Cityscapes semantic segmentation data. **Bold** - the highest numbers (not including the En.frozen results). Highlighted in blue - the highest for all numbers (including the En.frozen results).

Model	Pre-trained	AP	AP_{50}	AP_s	AP_m	AP_l
Mask-RCNN [15]	ImageNet	41.0	65.8	18.7	39.9	60.6
	(En.frozen)	(36.6)	(65.6)	(16.8)	(37.1)	(53.8)
	SeeTh. w/o CPP	44.6	69.1	20.2	42.7	64.3
	(En.frozen)	(42.1)	(69.7)	(21.9)	(40.9)	(59.0)
	SeeTh. w/ CPP	**45.9**	**71.3**	**22.3**	**43.9**	**65.3**
	(En.frozen)	(46.1)	(75.1)	(32.3)	(47.0)	(61.8)
FCOS [42]	ImageNet	37.7	62.4	15.2	35.2	55.9
	(En.frozen)	(32.4)	(59.6)	(13.8)	(31.1)	(49.8)
	SeeTh. w/o CPP	42.3	67.7	19.7	41.8	59.8
	(En.frozen)	(39.2)	(66.1)	(19.8)	(38.2)	(53.9)
	SeeTh. w/ CPP	**44.2**	**70.7**	**20.7**	**43.4**	**61.8**
	(En.frozen)	(43.8)	(72.3)	(22.0)	(44.2)	(58.1)
Faster-RCNN [31]	ImageNet	39.4	64.7	17.5	39.3	58.1
	(En.frozen)	(34.0)	(62.1)	(16.5)	(34.5)	(49.6)
	SeeTh. w/o CPP	43.4	68.4	20.3	41.2	**63.0**
	(En.frozen)	(39.7)	(69.0)	(20.1)	(39.4)	(55.3)
	SeeTh. w/ CPP	**44.7**	**71.6**	**22.4**	**43.4**	61.7
	(En.frozen)	(44.8)	(75.7)	(31.5)	(45.9)	(58.4)

Cityscapes benchmark. SeeThroughNet achieved state-of-the-art results (mIoU 86.12%) on the Cityscapes benchmark. For the inference, we used the scales={0.5, 1.0, 2.0} and the input image flipping. Table 6 shows the comparison between the SeeThroughNet and other semantic segmentation models on the Cityscapes *test* set. In Figure 7, SeeThroughNet shows finer and more accurate results for the occluded and distant objects than another state-of-the-art model InverseForm [3]. For example, the Grad-CAM visualizations show stronger and finer activation for the corresponding classes resulting in better inference results. In addition, in row 2 in Figure 7, the SeeThroughNet is able to segment the occluded rider in the distance.

Table 6. Comparison between SeeThroughNet and other methods on the Cityscapes leaderboard (*test* set). The Map. column indicates whether Mapillary Vista was used for pretraining.

Method	Map.	road	swalk	build.	wall	fence	pole	tlight	tsign	veg.	terrain	sky	person	rider	car	truck	bus	train	mcyc	bicyc	mIoU (%)
PSPNet [53]	No	98.68	86.92	93.46	58.38	63.67	67.67	76.12	80.47	93.63	72.20	95.29	86.83	71.91	96.21	77.69	91.51	83.64	70.80	77.53	81.19
DeepLabv3+ [5]	No	98.69	87.04	93.91	59.47	63.73	71.39	78.16	82.15	93.96	73.03	95.84	87.95	73.26	96.40	78.02	90.91	83.90	73.83	78.87	82.13
OCRNet [48]	Yes	98.82	88.29	94.26	66.88	66.69	73.28	80.21	83.04	94.20	74.10	95.97	88.50	75.78	96.51	78.51	91.78	90.12	73.40	79.31	83.67
DecoupleSegNet [23]	Yes	98.79	87.78	94.37	66.07	64.77	72.32	78.79	82.61	94.20	73.97	96.12	88.67	75.88	96.58	80.19	93.78	91.55	74.32	79.53	83.70
SegFix [50]	Yes	98.88	88.33	94.39	67.97	67.82	73.59	80.60	83.92	94.35	74.45	96.06	89.21	75.85	96.83	83.62	94.17	91.28	74.02	80.09	84.50
Panoptic Deeplab [7]	Yes	98.84	88.40	94.44	64.31	68.34	75.32	81.04	84.19	94.19	73.73	96.09	89.69	78.64	96.73	82.19	93.69	90.16	76.40	79.80	84.54
HMSA [41]	Yes	98.97	89.38	94.90	71.83	68.38	75.85	82.18	85.27	94.49	74.97	96.30	90.14	79.71	96.96	82.58	94.60	87.80	77.15	81.70	85.43
InverseForm [3]	Yes	98.60	**89.45**	94.61	71.80	69.24	75.84	82.38	**85.53**	94.32	75.00	95.83	**90.17**	**79.78**	96.92	**84.45**	**95.58**	88.65	76.68	**81.76**	85.61
SeeThroughNet (Ours)	Yes	**98.94**	89.09	**95.04**	**73.76**	**71.35**	**76.10**	**82.41**	85.13	**94.52**	**75.26**	**96.25**	90.06	79.63	**97.04**	84.25	95.07	**92.68**	**77.85**	81.76	**86.12**

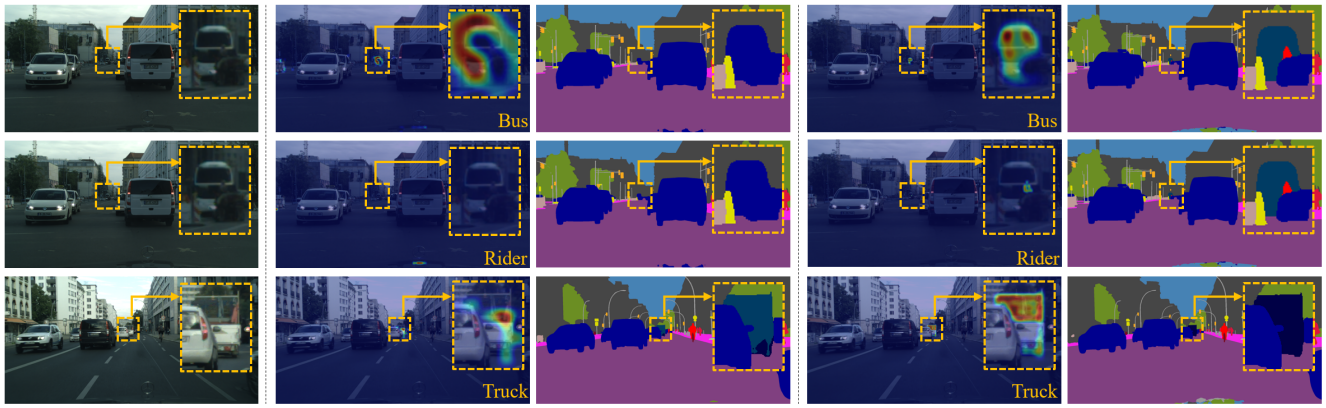


Figure 7. SeeThroughNet qualitative results on the Cityscapes *test* set. The inference results and Grad-CAM [34] visualizations of the SeeThroughNet and another state-of-the-art model InverseForm [3] are compared. The SeeThroughNet shows finer and more accurate results for the occluded and distant objects. The Grad-CAM classes are marked in the right-bottom corner of the Grad-CAM images. The enlarged views are provided. More examples are in the Appendix.

SeeThroughNet feature transfer learning for object detection task. We transferred the encoder (HRNetV2-48) of the trained SeeThroughNet on Cityscapes to the detection task to see how the learned features with CPP from SeeThroughNet could affect the detection task. The detection labels were parsed with the Cityscapes instance dataset to utilize the SeeThroughNet encoder weight trained on the Cityscapes semantic segmentation data. As in Table 5, the detection models trained with the pre-trained SeeThroughNet w/ CPP encoder shows better accuracy than the other two conditions. Also, for comparing the encoder frozen and not-frozen models, the results of the freeze models for the SeeThroughNet w/ CPP encoder show similar or higher accuracy than the not-freeze models, while the other two conditions show more improvement with the not-frozen models. These results indicate that the auxiliary loss with CPP pooling help to learn feature representations for another task such as object detection even without further training. In addition, the encoder frozen scores with SeeThroughNet w/ CPP show more improvements in AP_s than the others implying the effectiveness of CPP pooling for handling distant (small) and occluded objects.

5. Conclusion

In this paper, we propose a novel model-agnostic pooling methods called Class Probability Preserving (CPP) pooling to minimize information loss. To show the effect of the CPP pooling, we experimented with seven popular segmentation models on Cityscapes, Pascal VOC, Pascal Context, and NYU-Depth-v2 datasets, where our proposed CPP pooling consistently improved the models by simply applying it with auxiliary losses. The ablation experiments and class-wise analysis are demonstrated as well. In addition, we introduce the SeeThroughNet utilizing CPP pooling with auxiliary heads combined with multi-scale attention. The results of the ablation experiments and feature transfer learning experiments with the SeeThroughNet encoder are provided. This shows the possibility that the proposed method could also be effective for object detection tasks. The SeeThroughNet shows state-of-the-art results on the Cityscapes benchmarks. We expect our proposed method to potentially boost the auxiliary loss use to any segmentation models enhancing the performance.

References

[1] Nadeem Akhtar and U Ragavendran. Interpretation of intelligence in cnn-pooling processes: a methodological survey. *Neural computing and applications*, 32(3):879–898, 2020. 2

[2] Vijay Badrinarayanan, Alex Kendall, and Roberto Cipolla. Segnet: A deep convolutional encoder-decoder architecture for image segmentation. *IEEE TPAMI*, 39(12):2481–2495, 2017. 3

[3] Shubhankar Borse, Ying Wang, Yizhe Zhang, and Fatih Porikli. Inverseform: A loss function for structured boundary-aware segmentation. In *IEEE CVPR*, pages 5901–5911, 2021. 3, 7, 8

[4] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. *arXiv preprint arXiv:1706.05587*, 2017. 3

[5] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In *Springer ECCV*, pages 801–818, 2018. 1, 2, 3, 5, 6, 8

[6] Patrick Gallagher Zhengyou Zhang Zhuowen Tu Chen-Yu Lee, Saining Xie. Deeply supervised nets. In *AISTATS*, 2015. 1

[7] Bowen Cheng, Maxwell D Collins, Yukun Zhu, Ting Liu, Thomas S Huang, Hartwig Adam, and Liang-Chieh Chen. Panoptic-deeplab: A simple, strong, and fast baseline for bottom-up panoptic segmentation. In *IEEE CVPR*, pages 12475–12485, 2020. 8

[8] MMSegmentation Contributors. MMSegmentation: Openmmlab semantic segmentation toolbox and benchmark. https://github.com/open-mmlab/mmsegmentation, 2020. 5

[9] 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 *IEEE CVPR*, 2016. 2

[10] Yin Cui, Feng Zhou, Jiang Wang, Xiao Liu, Yuanqing Lin, and Serge Belongie. Kernel pooling for convolutional neural networks. In *IEEE CVPR*, pages 2921–2930, 2017. 2

[11] M. Everingham, L. Van Gool, C. K. I. Williams, J. Winn, and A. Zisserman. The PASCAL Visual Object Classes Challenge 2012 (VOC2012) Results. http://www.pascal-network.org/challenges/VOC/voc2012/workshop/index.html, 2012. 2

[12] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In *IEEE CVPR*, pages 3146–3154, 2019. 3

[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville, and Yoshua Bengio. *Deep learning*, volume 1. MIT press Cambridge, 2016. 4

[14] Junjun He, Zhongying Deng, and Yu Qiao. Dynamic multi-scale filters for semantic segmentation. In *IEEE ICCV*, October 2019. 2, 5, 6

[15] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE ICCV*, pages 2961–2969, 2017. 7

[16] Yuanming Hu, Baoyuan Wang, and Stephen Lin. Fc4: Fully convolutional color constancy with confidence-weighted pooling. In *IEEE CVPR*, pages 4085–4094, 2017. 2

[17] Zilong Huang, Xinggang Wang, Lichao Huang, Chang Huang, Yunchao Wei, and Wenyu Liu. Ccnet: Criss-cross attention for semantic segmentation. In *IEEE ICCV*, pages 603–612, 2019. 2, 5, 6

[18] Shaoqing Ren Jian Sun Kaiming He, Xiangyu Zhang. Delving deep into rectifiers: Surpassing human-level performance on imagenet classification. In *IEEE ICCV*, 2015. 1

[19] Alexander Kolesnikov and Christoph H Lampert. Seed, expand and constrain: Three principles for weakly-supervised image segmentation. In *Springer ECCV*, pages 695–711. Springer, 2016. 2

[20] Ashwani Kumar. Ordinal pooling networks: for preserving information over shrinking feature maps. *arXiv preprint arXiv:1804.02702*, 2018. 2

[21] Svetlana Lazebnik, Cordelia Schmid, and Jean Ponce. Beyond bags of features: Spatial pyramid matching for recognizing natural scene categories. In *IEEE CVPR*, volume 2, pages 2169–2178. IEEE, 2006. 2

[22] Chen-Yu Lee, Patrick W Gallagher, and Zhuowen Tu. Generalizing pooling functions in convolutional neural networks: Mixed, gated, and tree. In *Artificial intelligence and statistics*, pages 464–472. PMLR, 2016. 2

[23] Xiangtai Li, Xia Li, Li Zhang, Guangliang Cheng, Jianping Shi, Zhouchen Lin, Shaohua Tan, and Yunhai Tong. Improving semantic segmentation via decoupled body and edge supervision. *arXiv preprint arXiv:2007.10035*, 2020. 8

[24] Qingming Huang Li Shen, Zhouchen Lin. Relay backpropagation for effective learning of deep convolutional neural networks. In *Springer ECCV*, 2016. 1

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE CVPR*, pages 3431–3440, 2015. 3

[26] Roozbeh Mottaghi, Xianjie Chen, Xiaobai Liu, Nam-Gyu Cho, Seong-Whan Lee, Sanja Fidler, Raquel Urtasun, and Alan Yuille. The role of context for object detection and semantic segmentation in the wild. In *IEEE CVPR*, 2014. 2

[27] Rafael Müller, Simon Kornblith, and Geoffrey E Hinton. When does label smoothing help? *Advances in neural information processing systems*, 32, 2019. 7

[28] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 2

[29] Yuval Nirkin, Lior Wolf, and Tal Hassner. Hyperseg: Patch-wise hypernetwork for real-time semantic segmentation. In *IEEE CVPR*, pages 4061–4070, June 2021. 2

[30] Kunlun Qi, Qingfeng Guan, Chao Yang, Feifei Peng, Shengyu Shen, and Huayi Wu. Concentric circle pooling in deep convolutional networks for remote sensing scene classification. *Remote Sensing*, 10(6):934, 2018. 2

[31] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *NeurIPS*, 28:91–99, 2015. 7

[32] Oren Rippel, Jasper Snoek, and Ryan P Adams. Spectral representations for convolutional neural networks. *arXiv preprint arXiv:1506.03767*, 2015. 3

[33] Olaf Ronneberger, Philipp Fischer, and Thomas Brox. U-net: Convolutional networks for biomedical image segmentation. In *MICCAI*, pages 234–241. Springer, 2015. 3

[34] Ramprasaath R Selvaraju, Michael Cogswell, Abhishek Das, Ramakrishna Vedantam, Devi Parikh, and Dhruv Batra. Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE ICCV*, pages 618–626, 2017. 1, 6, 7, 8

[35] Arash Shahriari and Fatih Porikli. Multipartite pooling for deep convolutional neural networks. *arXiv preprint arXiv:1710.07435*, 2017. 2

[36] Ke Sun, Yang Zhao, Borui Jiang, Tianheng Cheng, Bin Xiao, Dong Liu, Yadong Mu, Xinggang Wang, Wenyu Liu, and Jingdong Wang. High-resolution representations for labeling pixels and regions. *arXiv preprint arXiv:1904.04514*, 2019. 4

[37] Manli Sun, Zhanjie Song, Xiaoheng Jiang, Jing Pan, and Yanwei Pang. Learning pooling for convolutional neural network. *Neurocomputing*, 224:96–104, 2017. 2

[38] Christian Szegedy, Wei Liu, Yangqing Jia, Pierre Sermanet, Scott Reed, Dragomir Anguelov, Dumitru Erhan, Vincent Vanhoucke, and Andrew Rabinovich. Going deeper with convolutions. In *Proceedings of the IEEE CVPR*, pages 1–9, 2015. 1

[39] Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. Rethinking the inception architecture for computer vision. In *IEEE CVPR*, pages 2818–2826, 2016. 3

[40] Towaki Takikawa, David Acuna, Varun Jampani, and Sanja Fidler. Gated-scnn: Gated shape cnns for semantic segmentation. In *Proceedings of the IEEE ICCV*, pages 5229–5238, 2019. 3

[41] Andrew Tao, Karan Sapra, and Bryan Catanzaro. Hierarchical multi-scale attention for semantic segmentation. *arXiv preprint arXiv:2005.10821*, 2020. 3, 4, 5, 8

[42] Zhi Tian, Chunhua Shen, Hao Chen, and Tong He. Fcos: Fully convolutional one-stage object detection. In *Proceedings of the IEEE ICCV*, pages 9627–9636, 2019. 7

[43] Zhiqiang Tong, Kazuyuki Aihara, and Gouhei Tanaka. A hybrid pooling method for convolutional neural networks. In *ICONIP*, pages 454–461. Springer, 2016. 2

[44] Travis Williams and Robert Li. Wavelet pooling for convolutional neural networks. In *ICLR*, 2018. 3

[45] Haibing Wu and Xiaodong Gu. Max-pooling dropout for regularization of convolutional neural networks. In *ICONIP*, pages 46–54. Springer, 2015. 2

[46] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *Springer ECCV*, pages 418–434, 2018. 2, 5, 6

[47] Dingjun Yu, Hanli Wang, Peiqiu Chen, and Zhihua Wei. Mixed pooling for convolutional neural networks. In *RSKT*, pages 364–375. Springer, 2014. 2

[48] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. *arXiv preprint arXiv:1909.11065*, 2019. 2, 3, 4, 5, 6, 8

[49] Yuhui Yuan, Lang Huang, Jianyuan Guo, Chao Zhang, Xilin Chen, and Jingdong Wang. Ocnet: Object context network for scene parsing. *arXiv preprint arXiv:1809.00916*, 2018. 3

[50] Yuhui Yuan, Jingyi Xie, Xilin Chen, and Jingdong Wang. Segfix: Model-agnostic boundary refinement for segmentation. In *Springer ECCV*, pages 489–506. Springer, 2020. 8

[51] Shuangfei Zhai, Hui Wu, Abhishek Kumar, Yu Cheng, Yongxi Lu, Zhongfei Zhang, and Rogerio Feris. S3pool: Pooling with stochastic spatial sampling. In *IEEE CVPR*, pages 4970–4978, 2017. 2

[52] Hang Zhang, Han Zhang, Chenguang Wang, and Junyuan Xie. Co-occurrent features in semantic segmentation. In *IEEE CVPR*, pages 548–557, 2019. 3

[53] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *IEEE CVPR*, pages 2881–2890, 2017. 1, 3, 5, 6, 8

[54] Shuai Zhao, Yang Wang, Zheng Yang, and Deng Cai. Region mutual information loss for semantic segmentation. *arXiv preprint arXiv:1910.12037*, 2019. 5

[55] Lanyun Zhu, Deyi Ji, Shiping Zhu, Weihao Gan, Wei Wu, and Junjie Yan. Learning statistical texture for semantic segmentation. In *IEEE CVPR*, pages 12537–12546, June 2021. 1

[56] Zhen Zhu, Mengde Xu, Song Bai, Tengteng Huang, and Xiang Bai. Asymmetric non-local neural networks for semantic segmentation. In *IEEE ICCV*, 2019. 1, 5, 6