EPSNet: Efficient Panoptic Segmentation Network with Cross-layer Attention Fusion

Chia-Yuan Chang[1][0000−0001−5841−3455], Shuo-En Chang[1][0000−0003−4916−499X],
Pei-Yung Hsiao[2][0000−0003−1750−7118], and Li-Chen Fu[1][0000−0002−6947−7646]

[1] National Taiwan University, Taipei, Taiwan
{r07922102,r08922a02,lichen}@ntu.edu.tw
[2] National University of Kaohsiung, Kaohsiung, Taiwan
pyhsiao@nuk.edu.tw

Abstract. Panoptic segmentation is a scene parsing task which unifies semantic segmentation and instance segmentation into one single task. However, the current state-of-the-art studies did not take too much concern on inference time. In this work, we propose an Efficient Panoptic Segmentation Network (EPSNet) to tackle the panoptic segmentation tasks with fast inference speed. Basically, EPSNet generates masks based on simple linear combination of prototype masks and mask coefficients. The light-weight network branches for instance segmentation and semantic segmentation only need to predict mask coefficients and produce masks with the shared prototypes predicted by prototype network branch. Furthermore, to enhance the quality of shared prototypes, we adopt a module called "cross-layer attention fusion module", which aggregates the multi-scale features with attention mechanism helping them capture the long-range dependencies between each other. To validate the proposed work, we have conducted various experiments on the challenging COCO panoptic dataset, which achieve highly promising performance with significantly faster inference speed (51ms on GPU).

1 Introduction

Due to Convolutional Neural Networks (CNNs) and other advances in deep learning, computer vision systems have achieved considerable success especially on computer vision tasks such as image recognition [1], semantic segmentation [2, 3], object detection [4, 5] and instance segmentation [6, 7]. In particular, semantic segmentation aims to assign specific class label for each image pixel, whereas instance segmentation predicts foreground object masks. However, the former is not capable of separating objects of the same class, and the latter only focuses on segmenting of *things* (i.e countable objects such as people, animals, and tools) rather than *stuff* (i.e amorphous regions such as grass, sky, and road). To overcome the respective shortcomings, combination of semantic segmentation and instance segmentation leads to the so-called panoptic segmentation [8]. More specifically, the goal of panoptic segmentation is to assign a semantic label and an instance ID to every pixel in an image.

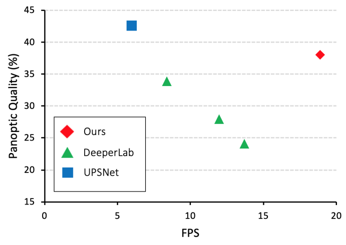

Fig. 1. Speed-performance trade-off of panoptic segmentation methods on COCO. The inference time is measured end-to-end from input image to panoptic segmentation output. Our approach achieves 19 fps and 38.6% PQ on COCO *val* set.

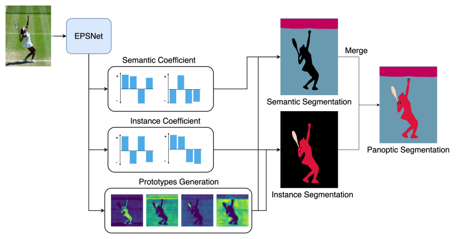

Fig. 2. Overview of Efficient Panoptic Network. EPSNet predicts prototypes and mask coefficients for semantic and instance segmentation. Both segmentation, obtained by linear combination of prototypes and mask coefficients, are fused using heuristic merging.

Several methods [9–14] have been proposed for panoptic segmentation in the literature. Detection-based approaches [9, 12–14] usually exploit an instance segmentation network like Mask R-CNN [6] as the main stream and attach lightweight semantic segmentation branch after the shared backbone. Then, they combine those outputs by heuristic fusion [8] to generate the final panoptic prediction. Despite such detection-based fashions achieving the state-of-the-art results, they solely aim to improve the performance but may sacrifice the computation load and speed. In fact, detection-based methods suffer from several limitations. First, due to the two-stage detector, instance segmentation branch costs the major computation time and drags down the inference speed. Second, most detection-based approaches commonly employ the outputs of backbone, like feature pyramid network [15], as shared features without further enhancement, causing sub-optimality of features used by the following branches. Lastly, the independent branches unfortunately lead to inconsistency when generating final prediction.

To address the above problems, we propose a novel one-stage framework called Efficient Panoptic Segmentation Network (EPSNet), as shown in Fig. 2. It adopts parallel networks to generate prototype masks for the entire image and predicts a set of coefficients for instance and semantic segmentation. Instance and semantic segments can be easily generated by linearly combining the prototypes with predicted coefficients from the branches. The proposed semantic branch only needs to produce coefficients for each class instead of pixel-wise predictions. Moreover, the prototypes are shared by both branches, which save time for producing large-size masks and help them solve their tasks simultaneously. Further, we introduce an innovative fusion module called cross-layer attention fusion module, which enhances the quality of shared features with attention mechanism. Instead of directly using suboptimal features in FPN, we choose certain layer as the target feature and other layers as source features and then apply an attention module on them to capture spatial dependencies for any two positions of the feature maps. For each position in target feature, it is updated via aggregating source features at all positions with weighted summation. To verify the efficiency of EPSNet, we conduct experiments on COCO [16] dataset. The experimental results manifest that our method achieves competitive performances with much faster inference compared to current approaches, as shown in Fig. 1.

2 Related Work

2.1 Panoptic Segmentation

Panoptic segmentation is originally proposed by [8]. In panoptic segmentation tasks, each pixel in the image needs to be assigned a semantic label and an instance ID. In [8], separate networks are used for semantic segmentation and instance segmentation, respectively, and then the results are combined with heuristic rules. The recent approaches of panoptic segmentation train semantic and instance segmentation network in end-to-end fashion with shared backbone. These methods can be categorized into two groups, namely, detection-based methods and bottom-up methods.

Detection-based. Most detection-based methods exploit Mask R-CNN [6] as their instance segmentation network and attach semantic segmentation branch with FCN [17] after shared backbone. These approaches are also considered as two-stage methods because of the additional stage to generate proposals. For instances, JSIS [18] firstly trains instance and semantic segmentation network jointly. TASCNet [19] ensures the consistency of stuff and thing prediction through binary mask. OANet [12] uses spatial ranking module to deal with the occlusion problem between the predicted instances. Panoptic FPN [14] endows Mask R-CNN [6] with a semantic segmentation branch. AUNet [13] adds RPN and thing segmentation mask attentions to stuff branch to provide object-level and pixel- level attentions. UPSNet [9] introduces a parameter-free panoptic head which solves the panoptic segmentation via pixel-wise classification. AdaptIS [10] adapts to the input point with a help of AdaIN layers [20] and produce masks for different objects on the same image. Although detection-based methods achieve better performance, they are usually slow in inference because of two-stage Mask R-CNN [6] in instance head. In addition, the inconsistency of semantic and instance segmentation needs to be solved when the two are merged into panoptic segmentation.

Bottom-up. Unlike the above approaches, some methods tackle panoptic segmentation tasks by associating pixel-level predictions to each object instance [21–24]. In these approaches, they first predict the foreground mask with semantic segmentation, and then use several types of heatmaps to group foreground pixels into objects. DeeperLab [25] predicts instance keypoint as well as multi-range offset heatmap and then groups them into class-agnostic instance segmentation. In semantic segmentation head, they follow the design of DeepLab [3]. At the end, panoptic segmentation is generated by merging class-agnostic instance masks and semantic output. SSAP [26] groups pixels based on a pixel-pair affinity pyramid with an efficient graph partition method. Despite the single-shot architecture of bottom-up approaches, their post-processing step still needs major computational time. Also, the performance of the bottom-up methods usually is inferior to that of the detection-based methods.

Recently, the proposed methods obtain shared feature for semantic and instance head. The quality of shared feature is highly essential for the following network head to produce better results. Still, the proposed approaches do not take this into consideration, and they usually make use of the output of shared backbone as shared feature directly.

In this work, we aim to propose a panoptic segmentation network based on one-stage detector to attain fast inference speed and competitive performance. To increase the quality of shared feature, our proposed cross-layer attention fusion, which is a lightweight network, provides the target feature map with richer information in different feature pyramid layers using attention mechanism.

3 Efficient Panoptic Segmentation Network

3.1 Efficient Panoptic Segmentation Network

Our method consists of five major components including (1) shared backbone, (2) protohead for generating prototypes, (3) instance segmentation head, (4) semantic segmentation head, and (5) cross-layer attention fusion module.

Backbone. Our backbone exploits a deep residual network (ResNet) [1] with a feature pyramid network (FPN) [15], which takes a standard network with features at multiple spatial resolutions and adds a light top-down pathway with lateral connections. It generates a pyramid feature with scales from 1/8 to 1/128 resolution (F_3 to F_7) as in Fig. 3 . For these features, F_7 is fed to the semantic head, and F_3 to F_5 are sent to instance head and protohead as inputs.

Protohead. Rather than producing masks with FCN [17], inspired by Yolact [27], we choose to combine prototypes and mask coefficients with linear combination to generate masks. Our network heads only need to deal with mask coefficients and construct masks with shared prototypes. The goal of protohead is to provide high-quality prototypes which contain semantic information and details of high-resolution feature.

To generate higher resolution prototypes with more semantic values, we perform cross-layer attention fusion module to aggregate multi-scale features in backbone into information-richer feature maps for protohead as inputs. Then, we apply three convolutional blocks, $2\times$ bilinear upsampling and 1×1 convolution to produce output prototypes which are at 1/4 scale with k channels.

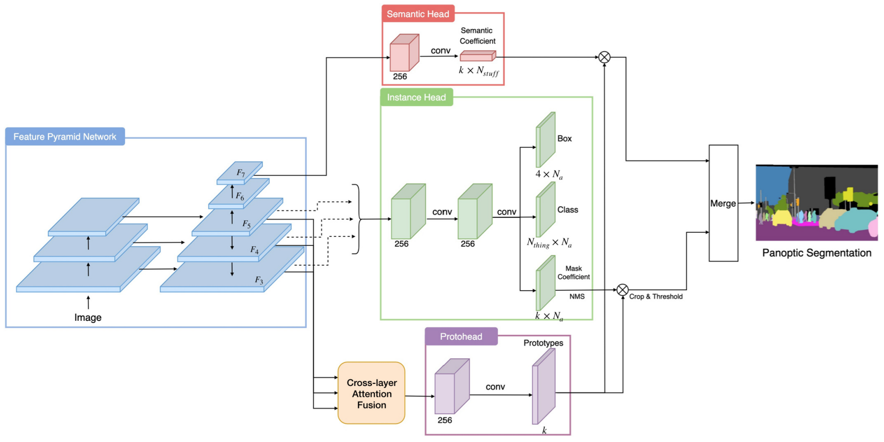

Fig. 3. Architecture of EPSNet. We adopt ResNet-101 [1] with FPN [15] as our backbone and only exploit F_3, F_4, F_5 for cross-layer attention fusion module. The prototypes are shared by semantic and instance head. k denotes the number of prototypes. N_a denotes the number of anchors. N_{thing} and N_{stuff} stands for the number of thing and stuff classes, respectively. $\otimes$ means matrix multiplication.

Instance Segmentation Head. In most panoptic segmentation networks, they adopt Mask R-CNN [6] as their instance segmentation branch. Yet, Mask R-CNN [6] needs to generate proposals first and then classify and segment those proposals in the second stage. Inspired by one-stage detector, Yolact [27], our instance head directly predicts object detection results and mask coefficients to make up the segmentation with prototypes without feature localization (e.g. ROI Align [6]) and refinement.

The instance head aims to predict box regression, classification confidences and mask coefficients. There are three branches in instance head. Regression branch predicts 4 box regression values, classification branch predicts N_{thing} class confidences, and mask branch predicts k mask coefficients. Thus, there are totally $4+N_{\text{thing}}+k$ values for each anchor. We perform a convolutional block on input features (F_3 to F_5) first and send them to each branch to predict respective results. In mask branch, we choose tanh as the activation function, which allows subtraction when linearly combining the coefficients.

In inference, we choose the mask coefficients whose corresponding bounding boxes survive after NMS procedure. Then, we combine mask coefficients and prototypes generated from protohead with linear combination followed by *sigmoid* to produce instance masks and crop final mask with predicted bounding box. During training, we crop mask with ground truth bounding box and divide mask segmentation loss by the area of ground truth bounding box.

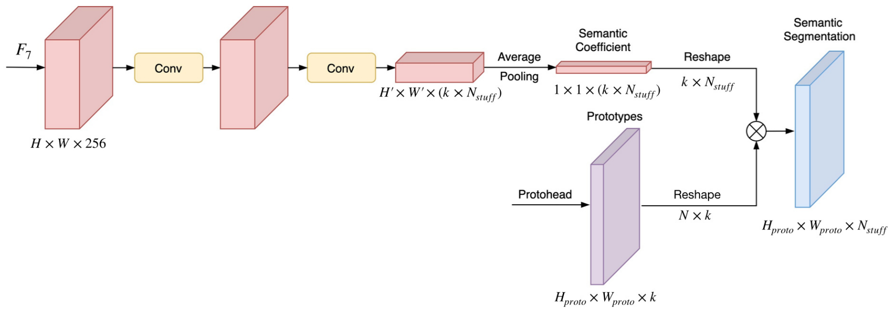
F_7
$H \times W \times 256$
Conv
Conv
$H' \times W' \times (k \times N_{stuff})$
Average Pooling
Semantic Coefficient
$1 \times 1 \times (k \times N_{stuff})$
Reshape
$k \times N_{stuff}$
Semantic Segmentation
Prototypes
Protohead
Reshape
$N \times k$
$H_{proto} \times W_{proto} \times N_{stuff}$
$H_{proto} \times W_{proto} \times k$

Fig. 4. Design of proposed semantic head. We adopt F_7 as input and apply two convolutional layers. The semantic coefficients are produced after average pooling, which predicts k mask coefficients for each stuff classes. $\otimes$ denotes matrix multiplication.

Semantic Segmentation Head. Usually, semantic segmentation masks are generated by decoder network [28, 29, 2, 17], which applies FCN [17] networks and up-sampling layer on the features from backbone to make sure that the size of semantic segmentation outputs is similar to the original input size. However, due to the large feature maps, the computation speed is limited by image size.

To reduce the computation of large feature maps, we propose a novel semantic segmentation head which only produces mask coefficients for each class, see Fig. 4. The semantic masks can be easily generated by combining the coefficients and prototypes, and each semantic class only demands k mask coefficients. Therefore, with smaller feature maps, the proposed light-weight semantic head can achieve faster inference speed.

We adopt last layer F_7 from backbone as the input of semantic head. Two convolution blocks are performed. In the second convolution block, the output channel is set to $k \times N_{\text{stuff}}$ and tanh is used as activation function. Because of the channel size $k \times N_{\text{stuff}}$, every position in the feature map can predict k coefficients for each class to construct semantic segmentation. Accordingly, we perform average pooling to aggregate the mask coefficients from all positions to generate final semantic coefficients. Further, prototypes from protohead and semantic coefficients are reshaped to 2d matrix and applied with linear combination followed by $softmax$ to produce semantic segmentation result. The operation is able to be implemented by matrix multiplication which is defined as

$$S = softmax(P \cdot Y), \tag{1}$$

where $P \in \mathbb{R}^{N \times k}$ denotes prototypes , $Y \in \mathbb{R}^{k \times N_{\text{stuff}}}$ stands for the reshaped semantic coefficients, and $S \in \mathbb{R}^{N \times N_{\text{stuff}}}$ is semantic segmentation result. N_{stuff} represents the number of stuff classes including 'other' class and N denotes the number of locations in prototypes.

The feature maps in semantic head are much smaller than the large feature maps in other approaches. Our semantic head provides faster semantic segmentation generation and less computation cost.

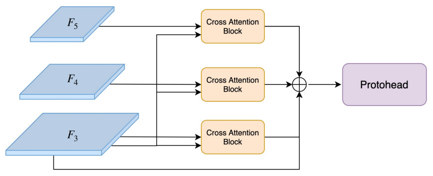

Fig. 5. The architecture of cross-layer attention module. The layers F_3, F_4 and F_5 in FPN are used. F_3 is considered as target feature, and all of them are set as source features. $\oplus$ denotes element-wise addition.

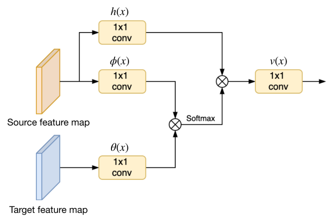

Fig. 6. The architecture of cross attention block. $\otimes$ denotes matrix multiplication.

Cross-layer Attention Fusion. Since all of the semantic or instance masks are derived from linear combination of mask coefficients and prototypes, the quality of prototypes significantly influences the generated masks. Hence, we decide to enrich the input features of protohead to help it produce preferable prototypes.

To enhance inputs of protohead, we propose a module called Cross-layer Attention (CLA) Fusion Module. This fusion module aims to efficiently aggregate the multi-scale feature maps in FPN layers. Certain layer within the module is chosen to be the target feature map and extract values from source feature maps with attention mechanism, as shown in Fig. 5. To provide high resolution details, the layer F_3, which is the highest resolution feature map in FPN, is selected as target feature map, and we choose other layers (e.g. F_3 to F_5) as source features to provide more semantic values for target.

Instead of directly merging multi-scale features with element-wise addition or concatenation [2, 9, 30, 31], we combine them with a block called Cross Attention Block. Inspired by non-local [32] and self-attention [33–35], which are able to capture long-range dependencies efficiently in feature map, the proposed cross attention block follows their concepts and further finds the long-range relationships from two different feature maps, as shown in Fig. 6. Each position in target feature map is updated with the weighted sum of all positions in source feature, where attention weight is calculated by similarities between the corresponding positions.

For the target feature map $T \in \mathbb{R}^{C \times N}$ and source feature map $O \in \mathbb{R}^{C \times N'}$, we first transform these feature maps into two feature spaces $\theta(x) = W_\theta x$ and $\phi(x) = W_\phi x$ and calculate the attention score s with dot product as shown below

$$s_{i,j} = \theta(T_i)^T \phi(O_j), \tag{2}$$

where $s_{i,j}$ measures the attention score of position j in O and position i in T. Here, C denotes the number of channels, and N denotes the number of locations from feature maps. After that, we obtain the attention weight α by normalizing attention score for each position j with $softmax$

$$\alpha_{i,j} = \frac{\exp(s_{i,j})}{\sum_{j=1}^{N} \exp(s_{i,j})}, \tag{3}$$

where $\alpha_{i,j}$ stands for the normalized impact of position j in source feature map to position i in target feature map. Then, each position in output feature map $A \in \mathbb{R}^{C\times N}$ is produced by calculating weighted sum of source features across all positions. The operation is shown as follows

$$A_i = v(\sum_{j=1}^{N} \alpha_{i,j} h(O_j)). \tag{4}$$

Here, A_i denotes output feature on position i, and both $v(x) = W_v x$ and $h(x) = W_h x$ stand for embedding functions. The embedding functions θ, ϕ, h and v are implemented by 1×1 convolution, and their output channels are set to 128, which is 1/2 of input channel to reduce computation cost. Finally, we apply cross attention block on each layer including F_5, F_4 and F_3 and consider F_3 as target feature map. The overall operation is defined as

$$Z = A^{F_3,F_5} + A^{F_3,F_4} + A^{F_3,F_3} + F_3, \tag{5}$$

where Z denotes the aggregated result from source and target features. Also, we adopt residual connection that makes a new cross attention block easier to insert without interfering the initial behaviors.

With the cross attention block, each position in the target feature is able to obtain spatial dependencies over all positions in feature maps from other layers. Moreover, we also select F_3 as source feature map even F_3 is target feature. In this case, it is same as self-attention, which helps target feature capture long-dependencies on its own feature map.

3.2 Training and Inference

During training, our EPSNet contains 4 loss functions in total, namely, classification loss $\mathcal{L}_c$, box regression loss $\mathcal{L}_b$, instance mask segmentation loss $\mathcal{L}_m$ and semantic segmentation loss $\mathcal{L}_s$. Because each loss function is in different scales and normalized policies, different weights on different loss functions actually affect the final performance on instance and semantic branch. Thus, we set several hyper-parameters on those loss functions, which is defined as $\mathcal{L} = \lambda_c\mathcal{L}_c + \lambda_b\mathcal{L}_b + \lambda_m\mathcal{L}_m + \lambda_s\mathcal{L}_s$.

In inference, since we won't allow overlaps on each pixel in panoptic segmentation, we resolve overlaps in instance segmentation with post-processing proposed in [8]. After getting non-overlapping instance segmentation results, we resolve possible overlaps between instance and semantic segmentation in favor of instance. Further, the stuff regions which are predicted as 'other' or under a predefined threshold are removed.

4 Experiments

In this section, we conduct experiments on COCO [16] panoptic dataset. Experimental results demonstrate that EPSNet achieves fast and competitive performance on COCO dataset.

4.1 Experimental Setup

Datasets. COCO [16] panoptic segmentation task consists of 80 thing classes and 53 stuff classes. There are approximately 118K images on training set and 5K on validation set.

Metrics. We adopt the evaluation metric called panoptic quality (PQ), which is introduced by [8]. Panoptic quality is defined as:

$$PQ = \underbrace{\frac{\sum_{(p,g)\in \mathrm{TP}} \mathrm{IoU}(p,g)}{|\mathrm{TP}|}}_{\mathrm{SQ}} \underbrace{\frac{|\mathrm{TP}|}{|\mathrm{TP}| + \frac{1}{2}|\mathrm{FP}| + \frac{1}{2}|\mathrm{FN}|}}_{\mathrm{RQ}}, \tag{6}$$

which can be considered as the multiplication of semantic quality (SQ) and recognition quality (RQ). Here, p and g are predicted and ground truth segments. TP, FP and FN represent the set of true positives, false positives and false negatives, respectively.

Implementation Details. We implement our method based on Pytorch [36] with single GPU RTX 2080Ti. The models are trained with batch size 2. Owing to the small batch size, we freeze the batch normalization layers within backbone and add group normalization [37] layers in each head. The initial learning rate and weight decay are set to 10^{-3} and 5×10^{-4}. We train with SGD for 3200K iterations and decay the learning rate by a factor of 10 at 1120k, 2400K, 2800k and 3000k iterations and a momentum of 0.9. The loss weights λ_c, λ_b, λ_m and λ_s are 1, 1.5, 6.125 and 2, respectively.

Our models are trained with ResNet-101 [1] backbone using FPN with ImageNet [38] pre-trained weights. We adopt similar training strategies in backbone and instance segmentation head as Yolact [27]. The number of prototypes k is set to 32. Our instance head only predicts thing classes, and semantic head predicts stuff classes viewing thing class as other. The cross attention blocks are shared in cross-layer attention fusion module. The base image size is 550×550. We do not preserve aspect ratio in order to get consistent evaluation times per image. We perform random flipping, random cropping and random scaling on images for data augmentation. The image size is randomly scaled in range $[550, 825]$ and then randomly cropped into 550×550.

4.2 Ablation Study

To verify the performance of training decisions and our cross-layer fusion attention module, we conduct the experiments with different settings in Table 1 on COCO panoptic *val* set. The empty cells in the table indicate that the corresponding component is not used.

Table 1. Ablation study on COCO panoptic *val* set with panoptic quality (PQ), semantic quality (SQ) and recognition quality (RQ). PQ^{Th} and PQ^{St} indicate PQ on thing and stuff classes. *Data Aug* denotes data augmentation. *CLA Fusion* stands for cross-layer attention fusion.

Data Aug	CLA Fusion	Loss Balance	PQ	PQ^{Th}	PQ^{St}	SQ	RQ
			35.4	40.5	27.7	77.2	43.5
√			37.4	43.2	28.6	77.6	45.7
√	√		38.4	43.0	**31.4**	77.7	**47.6**
√	√	√	**38.6**	**43.5**	31.3	**77.9**	47.3

Table 2. Ablation study on number of prototypes.

Prototypes	PQ	PQ^{Th}	PQ^{St}	Inf time (ms)
16	38.4	43.4	30.8	**50.7**
32	**38.6**	**43.5**	**31.3**	51.1
64	38.4	43.2	31.2	52.4

Data Augmentation. We compare the model without using data augmentation during training. The first and second rows in Table 1 show that the model trained with data augmentation improves by 2% in PQ. It proves that data augmentation plays important role during training.

Cross-layer Attention Fusion. For the model without using cross-layer fusion module, it is replaced by another fusion module similar to Panoptic-FPN [14]. We adopt convolution blocks for different layers in backbone and combine them together. The layers F_5, F_4 and F_3 are attached with 3, 2 and 1 convolution blocks respectively and $2\times$ bilinear upsampling layers between each block. Output features are obtained by combining them with element-wise addition. As shown in second and third rows in Table 1, the model employing the cross-layer attention fusion module yields 38.4% in PQ and 2.8% improvement in PQ^{St}.

Loss Balance. In order to balance the loss values in similar order of magnitude, we assign different weights for each loss during training. With loss balance, the weights are same as experimental setting. Without loss balance, all weights are set to 1. As shown in the third and fourth rows in Table 1, the model with loss balance performs better especially on PQ^{Th}.

Prototypes. We further conduct experiments on the number of prototypes. As shown in Table 2, the number of prototypes barely influence the overall performance. However, it will affect the inference and post-processing time. We choose 32 prototypes for the setting of EPSNet.

The ablation results show that our cross-layer attention fusion and training strategies bring significant improvement on panoptic segmentation.

Table 3. Performance comparison on different design of semantic head. *standard* denotes EPSNet with other design choice on semantic head, which directly generates semantic segmentation with convolutional layers. Note that, the EPSNet here does not use CLA fusion. M-adds denotes multiply-adds.

Method	PQ	PQ^{Th}	PQ^{St}	M-Adds (M)
standard	37.2	42.9	28.5	33.8
coefficients	**37.4**	**43.2**	**28.6**	**9.4**

Table 4. Performance comparison of using different options on semantic coefficients.

Method	PQ	PQ^{St}
top-left	33.2	17.7
top-right	33.4	18.1
bottom-left	33.5	18.3
bottom-right	33.6	18.5
center	37.5	28.5
max pooling	37.6	28.8
average pooling	**38.6**	**31.3**

4.3 Analysis of Semantic Head

We further compare our semantic segmentation head to other design choice. In most proposed panoptic segmentation networks, they adopt feature maps in backbone and perform FCN [17] to obtain pixel-wise predictions. The semantic head of compared model is constructed by apply 1×1 convolutional layer on the the fused feature maps with the Panoptic-FPN fusion, whose size is same as F_3. Note that we only replace the semantic head in EPSNet without CLA fusion. We count the multiply-adds to evaluate the computation cost of different structures for semantic heads.

The experimental results in Table 3 show that the computation cost of the proposed semantic head is about 0.3 times less than the standard semantic head, although our semantic head is deeper. Unlike the standard semantic segmentation, because of the small input feature maps for computation, the proposed semantic head using mask coefficients does not slow down inference speed and outperforms the standard semantic head.

In semantic head, the coefficients $k \times N_{\text{stuff}}$ in each position is able to be used to generate semantic segmentation before average pooling. To verify the impact of coefficients from different position, we use the coefficients before average pooling from corner positions and center position to perform the semantic segmentations. In Table 4, the comparison shows that the result using coefficients from center position is superior than other positions. Moreover, we compare the different options on pooling operation. The coefficients produced by average pooling yield better performance than using max pooling, as shown in Table 4.

To sum up, the proposed semantic head predicts mask coefficients of each class with faster inference speed and efficiently exploits shared prototypes without dragging down panoptic segmentation performance.

4.4 Analysis of Cross-layer Attention Fusion Module

In this subsection, we investigate different strategies of using cross-layer attention (CLA) fusion. We compare our EPSNet to the model using other fusion method like Panoptic-FPN [14] with F_3, F_4 and F_5 and another model only employing

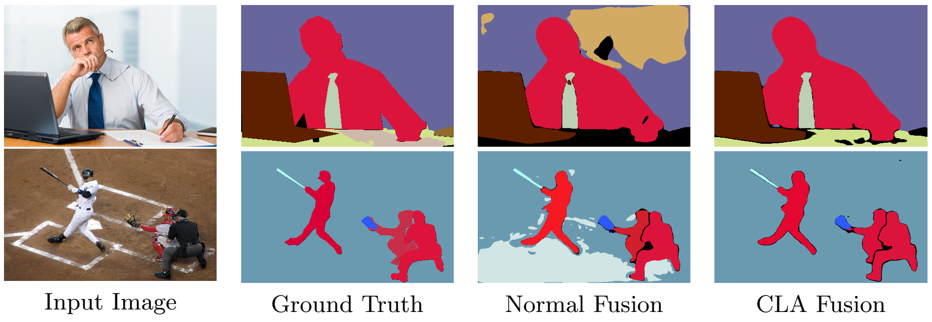

Fig. 7. Visualization results of cross-layer attention fusion on COCO panoptic *val* set. Normal fusion stands for the Panoptic-FPN [14] fusion with F_3,F_4 and F_5.

Table 5. Performance comparison on different strategies for fusion module. The inference time is measured without considering post-processing.

Method	PQ	$\mathrm{PQ}^{\mathrm{Th}}$	$\mathrm{PQ}^{\mathrm{St}}$	Inf time
Panoptic-FPN fusion [14] (F_3, F_4, F_5)	37.4	43.2	28.6	23ms
CLA fusion (F_3)	38.3	43.5	30.5	27ms
CLA fusion (F_3, F_4, F_5)	**38.6**	**43.5**	**31.3**	29ms

F_3 as source feature map with CLA fusion module. As shown in Table 5, the model with CLA fusion outperforms the Panoptic-FPN fusion espescially on $\mathrm{PQ}^{\mathrm{St}}$. Also, more layers are adopted in CLA fusion can yield slight improvement and inference time. The comparison of using cross-layer attention fusion can be visualized as Fig. 7. The details of background are much better and clearer. CLA fusion helps model generate higher quality segmentation especially for the stuff classes. For instance, the segments of the table in the first row and the ground in the second rows are much more complete.

To further understand what has been learned in CLA fusion module, we select two query points in input image in the first and fourth columns and visualize their corresponding sub-attention maps on other source features (F_3 and F_4) in remaining columns. In Fig. 8, we observe that CLA fusion module can capture long-range dependencies according to the similarity. For example, in first row, the red point #1 on bus pays more attention on positions labeled as bus (second and third columns). For the point # 2 on ground, it highlights most areas labeled as ground(fifth and sixth columns).

4.5 Comparison with Other Methods on COCO

We compare our method on COCO *val* set with panoptic quality and inference speed measured from input image to panoptic segmentation output including post-processing time. Specifically, our model is only trained on COCO training dataset with ResNet-101-FPN and tested using single-scale 550×550 image. As shown in Table 6, EPSNet outperforms every one-stage method and improves the

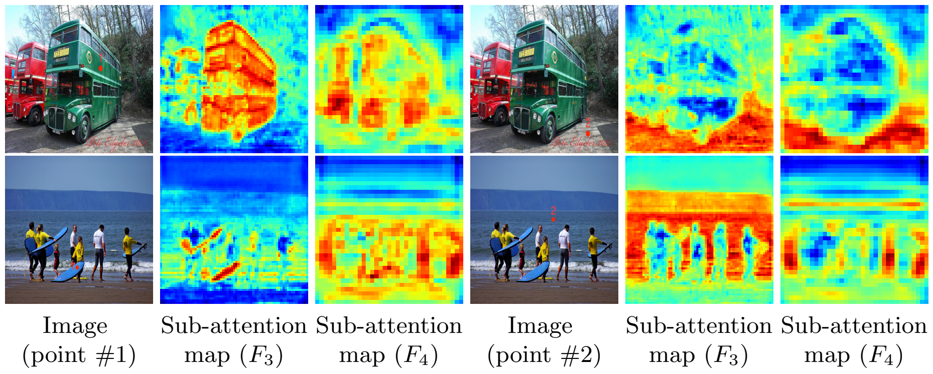
Image (point #1)
Sub-attention map (F_3)
Sub-attention map (F_4)
Image (point #2)
Sub-attention map (F_3)
Sub-attention map (F_4)

Fig. 8. Visualization results of cross attention block on COCO panoptic *val* set. In each row, we show the input images with different marked points in 1^{st} and 4^{th} columns and two sub-attention maps on source features (F_3 and F_4) corresponding to the marked point in 2^{nd}, 3^{th}, 5^{th} and 6^{th} columns.

performance over Real-time Panoptic Segmentation [39] with ResNet-50-FPN backbone and large input size by 1.5% PQ. Also, our inference speed is much faster than all existing panoptic segmentation methods. EPSNet only takes 51ms for inference, which is 1.4× faster than DeeperLab with Light Wider MobileNet-V2 [40] backbone and 3.1× faster than UPSNet [9]. Compared to the two-stage methods, we bring better performance especially on PQ^{St}, which outperforms Panoptic-FPN [14] by 1.8%, indicating that our approach provides better results on semantic segmentation. Despite the one-stage detector of EPSNet, with the fusion module and efficient architecture, we not only achieve competitive result for panoptic segmentation but also boost the inference speed.

In COCO *test* set, the inference setting is the same as COCO *val* set experiment. As shown in Table 7, we outperform SSAP [26], which adopts horizontal flipping and multi-scale input images for testing , by 2% PQ. Without any additional tricks, we still achieve competitive result compared to other methods.

5 Conclusions

In this paper, we present a one-stage Efficient Panoptic Segmentation Network. The masks are efficiently constructed by linear combination of prototypes generated by protohead and mask coefficients produced by instance and semantic branches. The proposed cross-layer attention fusion module aggregates multi-scale features in different layers with attention mechanism to enhance the quality of shared prototypes. The experiments show that our method achieves competitive performance on COCO panoptic dataset and outperforms other one-stage approaches. Also, EPSNet is significantly faster than the existing panoptic segmentation networks. In the future, We would like to explore a more effective way to replace the heuristic merging algorithm.

Table 6. Panoptic segmentation results on COCO *val* set. *LW-MNV2* denotes Light Wider MobileNet-V2.

Method	Backbone	Input Size	PQ	PQ^{Th}	PQ^{St}	Inf time (ms)
Two Stage						
JSIS [18]	ResNet-50	400×400	26.9	29.3	23.3	-
AUNet [13]	ResNet-50-FPN	-	39.6	49.1	25.2	-
Panoptic-FPN [14]	ResNet-101-FPN	-	40.3	47.5	29.5	-
AdaptIS [10]	ResNeXt-101	-	42.3	49.2	31.8	-
UPSNet [9]	ResNet-50-FPN	800×1333	42.5	48.6	33.4	167
Single Stage						
DeeperLab [25]	LW-MNV2	641×641	24.1	-	-	73
DeeperLab [25]	Xception-71	641×641	33.8	-	-	119
SSAP [26]	ResNet-101	512×512	36.5	-	-	-
Real-time PS [39]	ResNet-50-FPN	800×1333	37.1	41.0	31.3	63
Ours	ResNet-101-FPN	550×550	**38.6**	**43.5**	31.3	**51**

Table 7. Panoptic segmentation results on COCO *test-dev* set. *Flip* and *MS* stands for horizontal flipping and multi-scale inputs during testing.

Method	Backbone	Flip	MS	PQ	PQ^{Th}	PQ^{St}
Two Stage						
JSIS [18]	ResNet-50			27.2	29.6	23.4
Panoptic-FPN [14]	ResNet-101-FPN			40.9	48.3	29.7
AdaptIS [10]	ResNeXt-101	✓		42.8	50.1	31.8
AUNet [13]	ResNeXt-101-FPN		✓	46.5	55.8	32.5
UPSNet [9]	ResNet-101-FPN	✓	✓	46.6	53.2	36.7
Single Stage						
DeeperLab [25]	Xception-71			34.3	37.5	29.6
SSAP [26]	ResNet-101	✓	✓	36.9	40.1	**32.0**
Ours	ResNet-101-FPN			**38.9**	**44.1**	31.0

6 Acknowledgement

This work was partially sponsored by the Ministry of Science and Technology (MOST), Taiwan ROC, under Project 109-2634-F-002-027, 109-2634-F-002-040, and 109-2634-F-002-041. This research was also supported in part by the Center for AI & Advanced Robotics, National Taiwan University and the Joint Research Center for AI Technology and All Vista Healthcare under MOST.

References

1. He, K., Zhang, X., Ren, S., Sun, J.: Deep residual learning for image recognition. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 770–778
2. Zhao, H., Shi, J., Qi, X., Wang, X., Jia, J.: Pyramid scene parsing network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2017)

3. Chen, L.C., Zhu, Y., Papandreou, G., Schroff, F., Adam, H.: Encoder-decoder with atrous separable convolution for semantic image segmentation. In: The European Conference on Computer Vision (ECCV). (2018)
4. Ren, S., He, K., Girshick, R.B., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. IEEE Transactions on Pattern Analysis and Machine Intelligence **39** (2015) 1137–1149
5. Redmon, J., Farhadi, A.: Yolo9000: Better, faster, stronger. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 6517–6525
6. He, K., Gkioxari, G., Dollár, P., Girshick, R.B.: Mask r-cnn. The IEEE International Conference on Computer Vision (ICCV) (2017) 2980–2988
7. Liu, S., Qi, L., Qin, H., Shi, J., Jia, J.: Path aggregation network for instance segmentation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 8759–8768
8. Kirillov, A., He, K., Girshick, R.B., Rother, C., Dollár, P.: Panoptic segmentation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 9396–9405
9. Xiong, Y., Liao, R., Zhao, H., Hu, R., Bai, M., Yumer, E., Urtasun, R.: Upsnet: A unified panoptic segmentation network. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
10. Sofiiuk, K., Barinova, O., Konushin, A.: Adaptis: Adaptive instance selection network. In: The IEEE International Conference on Computer Vision (ICCV). (2019)
11. Yang, Y., Li, H., Li, X., Zhao, Q., Wu, J., Lin, Z.: SOGNet: Scene Overlap Graph Network for Panoptic Segmentation. arXiv preprint (2019) arXiv:1911.07527
12. Liu, H., Peng, C., Yu, C., Wang, J., Liu, X., Yu, G., Jiang, W.: An end-to-end network for panoptic segmentation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2019) 6165–6174
13. Li, Y., Chen, X., Zhu, Z., Xie, L., Huang, G., Du, D., Wang, X.: Attention-guided unified network for panoptic segmentation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 7019–7028
14. Kirillov, A., Girshick, R., He, K., Dollar, P.: Panoptic feature pyramid networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
15. Lin, T.Y., Dollár, P., Girshick, R.B., He, K., Hariharan, B., Belongie, S.J.: Feature pyramid networks for object detection. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 936–944
16. Caesar, H., Uijlings, J., Ferrari, V.: Coco-stuff: Thing and stuff classes in context. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
17. Long, J., Shelhamer, E., Darrell, T.: Fully convolutional networks for semantic segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2015) 3431–3440
18. de Geus, D., Meletis, P., Dubbelman, G.: Panoptic Segmentation with a Joint Semantic and Instance Segmentation Network. arXiv preprint (2018) arXiv:1809.02110
19. Li, J., Raventos, A., Bhargava, A., Tagawa, T., Gaidon, A.: Learning to Fuse Things and Stuff. arXiv preprint (2018) arXiv:1812.01192
20. Karras, T., Laine, S., Aila, T.: A style-based generator architecture for generative adversarial networks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 4396–4405

21. Zhang, Z., Fidler, S., Urtasun, R.: Instance-level segmentation for autonomous driving with deep densely connected mrfs. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2015) 669–677
22. Zhang, Z., Schwing, A.G., Fidler, S., Urtasun, R.: Monocular object instance segmentation and depth ordering with cnns. The IEEE International Conference on Computer Vision (ICCV) (2015) 2614–2622
23. Uhrig, J., Cordts, M., Franke, U., Brox, T.: Pixel-level encoding and depth layering for instance-level semantic labeling. In: GCPR. (2016)
24. Liang, X., Lin, L., Wei, Y., Shen, X., Yang, J., Yan, S.: Proposal-free network for instance-level object segmentation. IEEE Transactions on Pattern Analysis and Machine Intelligence **40** (2015) 2978–2991
25. Yang, T.J., Collins, M.D., Zhu, Y., Hwang, J.J., Liu, T., Zhang, X., Sze, V., Papandreou, G., Chen, L.C.: DeeperLab: Single-Shot Image Parser. arXiv preprint (2019) arXiv:1902.05093
26. Gao, N., Shan, Y., Wang, Y., Zhao, X., Yu, Y., Yang, M., Huang, K.: Ssap: Single-shot instance segmentation with affinity pyramid. In: The IEEE International Conference on Computer Vision (ICCV). (2019)
27. Bolya, D., Zhou, C., Xiao, F., Lee, Y.J.: Yolact: Real-time instance segmentation. In: The IEEE International Conference on Computer Vision (ICCV). (2019)
28. Lin, G., Milan, A., Shen, C., Reid, I.D.: Refinenet: Multi-path refinement networks for high-resolution semantic segmentation. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2016) 5168–5177
29. Zhang, Z., Zhang, X., Peng, C., Cheng, D., Sun, J.: Exfuse: Enhancing feature fusion for semantic segmentation. In: The European Conference on Computer Vision (ECCV). (2018)
30. Pang, J., Chen, K., Shi, J., Feng, H., Ouyang, W., Lin, D.: Libra r-cnn: Towards balanced learning for object detection. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
31. Porzi, L., Bulo, S.R., Colovic, A., Kontschieder, P.: Seamless scene segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
32. Wang, X., Girshick, R., Gupta, A., He, K.: Non-local neural networks. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2018)
33. Zhang, H., Goodfellow, I., Metaxas, D., Odena, A.: Self-Attention Generative Adversarial Networks. arXiv preprint (2018) arXiv:1805.08318
34. Fu, J., Liu, J., Tian, H., Li, Y., Bao, Y., Fang, Z., Lu, H.: Dual attention network for scene segmentation. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR). (2019)
35. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need. In: NIPS. (2017)
36. Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., Desmaison, A., Köpf, A., Yang, E., DeVito, Z., Raison, M., Tejani, A., Chilamkurthy, S., Steiner, B., Fang, L., Bai, J., Chintala, S.: Pytorch: An imperative style, high-performance deep learning library. In: NeurIPS. (2019)
37. Wu, Y., He, K.: Group normalization. In: The European Conference on Computer Vision (ECCV). (2018)
38. Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M.S., Berg, A.C., Li, F.F.: Imagenet large scale visual recognition challenge. International Journal of Computer Vision **115** (2014) 211–252

39. Hou, R., Li, J., Bhargava, A., Raventos, A., Guizilini, V., Fang, C., Lynch, J., Gaidon, A.: Real-time panoptic segmentation from dense detections. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) (2020) 8520–8529
40. Sandler, M., Howard, A.G., Zhu, M., Zhmoginov, A., Chen, L.C.: Mobilenetv2: Inverted residuals and linear bottlenecks. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (2018) 4510–4520