Abstract

In this paper, we present a conceptually simple, strong, and efficient framework for panoptic segmentation, called Panoptic FCN. Our approach aims to represent and predict foreground things and background stuff in a unified fully convolutional pipeline. In particular, Panoptic FCN encodes each object instance or stuff category into a specific kernel weight with the proposed kernel generator and produces the prediction by convolving the high-resolution feature directly. With this approach, instance-aware and semantically consistent properties for things and stuff can be respectively satisfied in a simple generate-kernel-then-segment workflow. Without extra boxes for localization or instance separation, the proposed approach outperforms previous box-based and -free models with high efficiency on COCO, Cityscapes, and Mapillary Vistas datasets with single scale input. Our code is made publicly available at https://github.com/yanwei-li/PanopticFCN.

1. Introduction

Panoptic segmentation, aiming to assign each pixel with a semantic label and unique identity, is regarded as a challenging task. In panoptic segmentation [18], countable and uncountable instances (i.e., things and stuff) are expected to be represented and resolved in a unified workflow. One main difficulty impeding unified representation comes from conflicting properties requested by things and stuff. Specifically, to distinguish among various identities, countable things usually rely on instance-aware features, which vary with objects. In contrast, uncountable stuff would prefer semantically consistent characters, which ensures consistent predictions for pixels with the same semantic meaning. An example is given in Fig. 1, where embedding of individuals should be diverse for inter-class variations, while characters of grass should be similar for intra-class consistency.

For conflict at feature level, specific modules are usually tailored for things and stuff separately, as presented in Fig. 1(a). In particular, instance-aware demand of things is satisfied mainly from two streams, namely box-based [17, 48, 24] and box-free [49, 10, 6] methods. Meanwhile, the semantic-consistency of stuff is met in a pixel-by-pixel manner [33], where similar semantic features would bring identical predictions. A classic case is Panoptic FPN [17], which utilizes Mask R-CNN [12] and FCN [33] in separated branches to respectively classify things and stuff, similar to that of Fig. 1(a). Although attempt [49, 10, 6] has been made to predict things without boxes, extra predictions (e.g., affinities [10], and offsets [49]) together with post-process procedures are still needed to distinguish among instances, which slow down the whole system and hinder it from being fully convolutional. Consequently, a unified representation is required to bridge this gap.

In this paper, we propose a fully convolutional framework for unified representation, called Panoptic FCN. In particular, Panoptic FCN encodes each instance into a specific kernel and generates the prediction by convolutions directly. Thus, both things and stuff can be predicted together.
with a same resolution. In this way, instance-aware and semantically consistent properties for things and stuff can be respectively satisfied in a unified workflow, which is briefly illustrated in Fig. 1(b). To sum up, the key idea of Panoptic FCN is to represent and predict things and stuff uniformly with generated kernels in a fully convolutional pipeline.

To this end, kernel generator and feature encoder are respectively designed for kernel weights generation and shared feature encoding. Specifically, in kernel generator, we draw inspirations from point-based object detectors [19, 53] and utilize the position head to locate as well as classify foreground objects and background stuff by object centers and stuff regions, respectively. Then, we select kernel weights with the same positions from the kernel head to represent corresponding instances. For the instance-awareness and semantic-consistency described above, a kernel-level operation, called kernel fusion, is further proposed, which merges kernel weights that are predicted to have the same identity or semantic category. With a naive feature encoder, which preserves the high-resolution feature with details, each prediction of things and stuff can be produced by convolving with generated kernels directly.

In general, the proposed method can be distinguished from two aspects. Firstly, different from previous work for things generation [12, 4, 44], which outputs dense predictions and then utilize NMS for overlaps removal, the designed framework generates instance-aware kernels and produces each specific instance directly. Moreover, compared with traditional FCN-based methods for stuff prediction [51, 3, 9], which select the most likely category in a pixel-by-pixel manner, our approach aggregates global context into semantically consistent kernels and presents results of existing semantic classes in a whole instance manner.

The overall approach, named Panoptic FCN, can be easily instantiated for panoptic segmentation, which will be fully elaborated in Sec. 3. To demonstrate its superiority, we give extensive ablation studies in Sec. 4.2. Furthermore, experimental results are reported on COCO [28], Cityscapes [8], and Mapillary Vistas [35] datasets. Without bells-and-whistles, Panoptic FCN outperforms previous methods with efficiency, and respectively attains 44.3% PQ and 47.5% PQ on COCO val and test-dev set. Meanwhile, it surpasses all similar box-free methods by a large margin and achieves leading performance on Cityscapes and Mapillary Vistas val set with 61.4% PQ and 36.9% PQ, respectively.

2. Related Work

Panoptic segmentation. Traditional approaches mainly conduct segmentation for things and stuff separately. The benchmark for panoptic segmentation [18] directly combines predictions of things and stuff from different models, causing heavy computational overhead. To solve this problem, methods have been proposed by dealing with things and stuff in one model but in separate branches, including Panoptic FPN [17], AUNet [24], and UPSNet [48]. From the view of instance representation, previous work mainly formats things and stuff from different perspectives. Foreground things are usually separated and represented with boxes [17, 50, 5, 23] or aggregated according to center offsets [49], while background stuff is often predicted with a parallel FCN [33] branch. Although methods of [22, 10] represent things and stuff uniformly, the inherent ambiguity cannot be resolved well merely with the pixel-level affinity, which yields the performance drop in complex scenarios. In contrast, the proposed Panoptic FCN represents things and stuff in a uniform and fully convolutional framework with decent performance and efficiency.

Instance segmentation. Instance segmentation aims to discriminate objects in the pixel level, which is a finer representation compared with detected boxes. For instance-awareness, previous works can be roughly divided into two streams, i.e., box-based methods and box-free approaches. Box-based methods usually utilize detected boxes to locate or separate objects, e.g., Mask R-CNN [12], PANet [32], YOLACT [1], CenterMask [20], and PointINS [38]. Meanwhile, box-free approaches are designed to generate instances without assistance of object boxes, including SGN [31], SSAP [10], TensorMask [4], and SOLO [44, 45]. In this paper, we represent objects in a box-free pipeline, which generates the kernel weight for each object and produces results by convolving the detail-rich feature directly, with no need for object-level duplicates removal [15, 37].

Semantic segmentation. Semantic segmentation assigns each pixel with a semantic category, without considering diverse object identities. In recent years, rapid progress has been made on top of FCN [33]. Due to the semantically consistent property, several attempts have been made to capture contextual cues from wider perception fields [51, 2, 3] or establish pixel-wise relationship for long-range dependencies [32, 16, 41]. There is also work to design network architectures for semantic segmentation automatically [29, 25], which is beyond the scope of this paper. Our proposed Panoptic FCN adopts a similar method to represent things and stuff, which aggregates global context into a specific kernel to predict corresponding semantic category.

3. Panoptic FCN

Panoptic FCN is conceptually simple: kernel generator is introduced to generate kernel weights for things and stuff with different categories; kernel fusion is designed to merge kernel weights with the same identity from multiple stages; and feature encoder is utilized to encode the high-resolution feature. In this section, we will elaborate on the above components as well as the training and inference scheme.
3.1. Kernel Generator

Given a single stage feature $X_i$ from the $i$-th stage in FPN [26], the proposed kernel generator aims at generating the kernel weight map $G_i$ with positions for things $L_i^{th}$ and stuff $L_i^{st}$, as depicted in Fig. 2. To this end, position head is utilized for instance localization and classification, while kernel head is designed for kernel weight generation.

**Position head.** With the input $X_i \in \mathbb{R}^{C_i \times W_i \times H_i}$, we simply adopt stacks of convolutions to encode the feature map and generate $X'_i$, as presented in Fig. 2. Then, we need to locate and classify each instance from the shared feature map $X'_i$. However, according to the definition [18], things can be distinguished by object centers, while stuff is uncountable. Thus, we adopt object centers and stuff regions to respectively represent position of each individual and stuff category. It means background regions with the same semantic meaning are viewed as one instance. In particular, object map $L_i^{th} \in \mathbb{R}^{N_{th}\times W_i \times H_i}$ and stuff map $L_i^{st} \in \mathbb{R}^{N_{st}\times W_i \times H_i}$ can be generated by convolutions directly with the shared feature map $X'_i$, where $N_{th}$ and $N_{st}$ denote the number of semantic category for things and stuff, respectively.

To better optimize $L_i^{th}$ and $L_i^{st}$, different strategies are adopted to generate the ground truth. For the $k$-th object in class $c$, we split the positive key-points onto the $c$-th channel of the heatmap $Y_i^{th} \in [0,1]^{N_{th}\times W_i \times H_i}$ with Gaussian kernel, similar to that in [19, 53]. Thus, ground truth of the $k$-th object in class $c$ can be represented as

$$Y_{i,c,x,y}^{th} = \exp \left( -\frac{(x-x_k)^2 + (y-y_k)^2}{2\sigma_k^2} \right),$$

where $x_k$ and $y_k$ denote coordinates of the $k$-th object center, and $\sigma_k$ indicates the object size-adaptive standard deviation in [19]. With respect to stuff regions, we produce the ground truth $Y_i^{st} \in [0,1]^{N_{st}\times W_i \times H_i}$ by bilinear interpolating the one-hot semantic annotation to corresponding sizes. Hence, the position head can be optimized with

$$L_{pos} = \sum_i FL(L_i^{th}, Y_i^{th})/N_{th},$$

$$L_{pos}^{st} = \sum_i FL(L_i^{st}, Y_i^{st})/W_iH_i,$$

where $L_{pos}$ and $L_{pos}^{st}$ denote the loss function for object centers and stuff regions, respectively. And $FL(L_i, Y_i)$ represents the Focal Loss [27] for corresponding prediction $L_i$ whose ground truth is $Y_i$ in the $i$-th stage. Consequently, the loss function for position head is formulated as

$$L_{pos} = L_{pos}^{th} + L_{pos}^{st}.$$

For inference, $D_i^{th} = \{(x,y) : 1(L_i^{th, c,x,y}) = 1\}$ and $D_i^{st} = \{(x,y) : 1(L_i^{st, c,x,y}) = 1\}$ are selected to respectively represent the existence of object centers and stuff regions in corresponding positions with predicted categories $C_i$. This process will be further explained in Sec. 3.4.

**Kernel head.** In kernel head, we first capture spatial cues by directly concatenating relative coordinates to the feature $X_i$, which is similar with that in CoordConv [30]. With the concatenated feature map $X'_i \in \mathbb{R}^{(C_i+2)\times W_i \times H_i}$, stacks of convolutions are adopted to generate the kernel weight map
\( \mathbf{G}_i \in \mathbb{R}^{C_c \times W_i \times H_i} \), as presented in Fig. 2. Given predictions \( D_i^{th} \) and \( D_i^{st} \) from the position head, kernel weights with the same coordinates in \( \mathbf{G}_i \) are chosen to represent corresponding instances. For example, assuming candidate \((x_c, y_c) \in D_i^{th}\), kernel weight \( \mathbf{G}_{i,x_c,y_c} \in \mathbb{R}^{C_c \times 1 \times 1} \) is selected to generate the result with predicted category \( c \). The same is true for \( D_i^{st} \). We represent the selected kernel weights in \( i \)-th stage for things and stuff as \( G_i^{th} \) and \( G_i^{st} \), respectively. Thus, the kernel weight \( G_i^{th} \) and \( G_i^{st} \) together with predicted categories \( C_i \) in the \( i \)-th stage can be produced with the proposed kernel generator.

### 3.2. Kernel Fusion

Some previous researches [39, 12, 44] utilize NMS to remove duplicate boxes or instances in the post-processing stage. Different from them, the designed kernel fusion operation merges repetitive kernel weights from multiple FPN stages before final instance generation, which guarantees instance-awareness and semantic-consistency for things and stuff, respectively. In particular, given aggregated kernel weights \( G_i^{th} \) and \( G_i^{st} \) from all the stages, the \( j \)-th kernel weight \( K_j \in \mathbb{R}^{C_c \times 1 \times 1} \) is achieved by

\[
K_j = \text{AvgPool}(G_j^i),
\]

where \( \text{AvgPool} \) denotes the average-pooling operation, and the candidate set \( G_j^i \) includes all the kernel weights, which are predicted to have the same identity ID with \( G_j \). For object centers, kernel weight \( G_m^{th} \) is viewed as identical with \( G_j^{th} \) if the cosine similarity between them surpasses a given threshold \( \lambda \), which will be further investigated in Table 3. For stuff regions, all kernel weights in \( G_i^{st} \) share a same category \( c \) with \( G_j^{st} \), which are marked as one identity ID.

With the proposed approach, each kernel weight \( K_i^{th} \) in \( K_i^{th} = \{K_i^{th}, \ldots, K_i^{th}\} \in \mathbb{R}^{M \times C_c \times 1 \times 1} \) can be viewed as an embedding for single object, where the total number of objects is \( M \). Therefore, kernels with the same identity are merged as a single embedding for things, and each kernel in \( K_i^{th} \) represents an individual object, which satisfies the instance-awareness for things. Meanwhile, kernel weight \( K_i^{st} \) in \( K_i^{st} = \{K_i^{st}, \ldots, K_i^{st}\} \in \mathbb{R}^{N \times C_c \times 1 \times 1} \) represents the embedding for all the \( j \)-th class pixels, where the existing number of stuff is \( N \). With this method, kernels with the same semantic category are fused to a single embedding, which guarantees the semantic-consistency for stuff. Thus, both properties requested by things and stuff can be fulfilled with the proposed kernel fusion operation.

### 3.3. Feature Encoder

To preserve details for instance representation, high-resolution feature \( \mathbf{F}_i^{th} \in \mathbb{R}^{C_c \times W_i^4 \times H_i^4} \) is utilized for feature encoding. Feature \( \mathbf{F}_i^{th} \) can be generated from FPN in several ways, e.g., P2 stage feature, summed features from all stages, and features from semantic FPN [17]. These methods are further compared in Table 5. With the generated feature \( \mathbf{F}_i^{th} \), a similar strategy with the kernel head is applied to encode positional cues and generate the encoded feature \( \mathbf{F}_i^{e} \in \mathbb{R}^{C_c \times W_i^4 \times H_i^4} \), as depicted in Fig. 2. Thus, given \( M \) and \( N \) kernel weights for things \( K_i^{th} \) and stuff \( K_i^{st} \) from the kernel fusion, each instance is produced by

\[
\mathbf{P}_j = K_j \otimes \mathbf{P}_j^{e},
\]

where \( \mathbf{P}_j \) denotes the \( j \)-th prediction, and \( \otimes \) indicates the convolutional operation. That means \( M + N \) kernel weights generate \( M + N \) instance predictions with resolution \( W_i^4 \times H_i^4 \) for the whole image. Consequently, the panoptic result can be produced with a simple process [17].

### 3.4. Training and Inference

#### Training scheme.

In the training stage, the central point in each object and all the points in stuff regions are utilized to generate kernel weights for things and stuff, respectively. Here, Dice Loss [34] is adopted to optimize the predicted segmentation, which can be formulated as

\[
\mathcal{L}_{\text{seg}} = \sum_j \text{Dice}(\mathbf{P}_j, \mathbf{Y}_j^{\text{seg}})/(M + N),
\]

where \( \mathbf{Y}_j^{\text{seg}} \) denotes ground truth for the \( j \)-th prediction \( \mathbf{P}_j \). To further release the potential of kernel generator, multiple positives inside each object are sampled to represent the instance. In particular, we select \( k \) positions with top predicted scores \( s \) inside each object in \( \mathbf{L}_i^{th} \), resulting in \( k \times M \) kernels as well as instances in total. This will be explored in Table 6. As for stuff regions, the factor \( \lambda \) is set to 1, which means all the points in same category are equally treated.

Then, we replace the original loss with a weighted version

\[
\text{WDice}(\mathbf{P}_j, \mathbf{Y}_j^{\text{seg}}) = \sum_k w_k \text{Dice}(\mathbf{P}_{j,k}, \mathbf{Y}_j^{\text{seg}}),
\]

where \( w_k \) denotes the \( k \)-th weighted score with \( w_k = s_k/\sum_k s_k \). According to Eqs. (3) and (6), optimized target \( \mathcal{L} \) is defined with the weighted Dice Loss \( \mathcal{L}_{\text{seg}} \)

\[
\mathcal{L}_{\text{seg}} = \sum_j \text{WDice}(\mathbf{P}_j, \mathbf{Y}_j^{\text{seg}})/(M + N),
\]

\[
\mathcal{L} = \lambda_{\text{pos}} \mathcal{L}_{\text{pos}} + \lambda_{\text{seg}} \mathcal{L}_{\text{seg}}.
\]

#### Inference scheme.

In the inference stage, Panoptic FCN follows a simple generate-kernel-then-segment pipeline. Specifically, we first aggregate positions \( D_i^{th}, D_i^{st} \) and corresponding categories \( C_i \) from the \( i \)-th position head, as illustrated in the Sec. 3.1. For object centers, we preserve the peak points in \( \text{MaxPool}(\mathbf{L}_i^{th}) \) utilizing a similar method.
with that in [53]. Thus, the indicator for things \( \mathbb{1}(L_{i,c,x,y}^{th}) \) is marked as positive if point \((x, y)\) in the \(c\)-th channel is preserved as the peak point. Similarly, the indicator for stuff regions \( \mathbb{1}(L_{i,c,x,y}^{st}) \) is viewed as positive if point \((x, y)\) with category \(c\) is kept. With the designed kernel fusion and the feature encoder, the prediction \( \mathbf{P} \) can be easily produced. Specifically, we keep the top 100 scoring kernels of objects and all the kernels of stuff after kernel fusion for instance generation. And the threshold 0.4 is utilized to convert predicted soft masks to binary results. For fair comparison as well as simplicity, we adopt procedure similar to that in [17] for non-overlap panoptic results.

4. Experiments

In this section, we first introduce the experimental setting for Panoptic FCN. Then we conduct abundant ablation studies on the COCO [28] val set to reveal the effect of each component. Finally, comparison with previous approaches on COCO [28], Cityscapes [8], Mapillary Vistas [8] datasets is reported. More experimental details and qualitative results are included in the attached supplementary material.

4.1. Experimental Setting

**Architecture.** From the perspective of network architecture, ResNet [13] with FPN [26] are utilized for backbone instantiation. P3 to P7 stages in FPN are used to provide single stage feature \( \mathbf{X}_i \) for the kernel generator that is shared across all stages. Meanwhile, P2 to P5 stages are adopted to generate the high-resolution feature \( \mathbf{F}^h \), which will be further investigated in Table 5. All convolutions in kernel generator are equipped with GroupNorm [46] and ReLU activation. Moreover, a naive convolution is adopted at the end of each head in kernel generator for feature projection.

**Datasets.** COCO dataset [28] is a widely used benchmark, which contains 80 thing classes and 53 stuff classes. It involves 118K, 5K, and 20K images for training, validation, and testing, respectively. Cityscapes dataset [8] consists of 5,000 street-view fine annotations with size 1024 \( \times \) 2048, which can be divided into 2,975, 500, and 1,525 images for training, validation, and testing, respectively. Mapillary Vistas [8] is a traffic-related dataset with resolutions ranging from 1024 \( \times \) 768 to more than 4000 \( \times \) 6000. It includes 37 thing classes and 28 stuff classes with 18K, 2K, and 5K images for training, validation, and testing, respectively.

**Optimization.** Network optimization is conducted using SGD with weight decay \( 1e^{-4} \) and momentum 0.9. And poly schedule with power 0.9 is adopted. Experimentally, \( \lambda_{pos} \) is set to a constant 1, and \( \lambda_{neg} \) are respectively set to 3, 4, and 3 for COCO, Cityscapes, and Mapillary Vistas datasets. For COCO, we set initial rate to 0.01 and follow the \( 1 \times \) strategy in Detectron2 [47] by default. We randomly flip and rescale the shorter edge from 640 to 800 pixels with 90K iterations. Herein, annotated object centers with instance scale range \( \{1,64\}, \{32,128\}, \{64,256\}, \{128,512\}, \{256,2048\} \) are assigned to P3-P7 stages, respectively. For Cityscapes, we optimize the network for 65K iterations with an initial rate 0.02 and construct each mini-batch with 32 random 512 \( \times \) 1024 crops from images that are randomly rescaled from 0.5 to 2.0 \( \times \). For Mapillary Vistas, the network is optimized for 150K iterations with an initial rate 0.02. In each iteration, we randomly resize images from 1024 to 2048 pixels at the shorted side and build 32 crops with the size 1024 \( \times \) 1024. Due to the variation in scale distribution, we modify the assigning strategy to \( \{1,128\}, \{64,256\}, \{128,512\}, \{256,1024\}, \{512,2048\} \) for Cityscapes and Mapillary Vistas datasets.

4.2. Component-wise Analysis

**Kernel generator.** Kernel generator plays a vital role in Panoptic FCN. Here, we compare several settings inside kernel generator to improve the kernel expressiveness in

| deform | conv num | PQ | PQ\textsuperscript{th} | PQ\textsuperscript{st} | AP | mIoU |
|-------|----------|----|-----------------|-----------------|----|------|
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |

| coord\textsubscript{w} | coord\textsubscript{l} | PQ | PQ\textsuperscript{th} | PQ\textsuperscript{st} | AP | mIoU |
|-------------------|---------------|----|-----------------|-----------------|----|------|
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |
| \( \checkmark \) | \( \checkmark \) | \( \mathbf{39.9} \) | \( \mathbf{45.0} \) | \( \mathbf{32.4} \) | \( \mathbf{32.9} \) | \( \mathbf{41.2} \) |

| class\textsubscript{aware} | thres | PQ | PQ\textsuperscript{th} | PQ\textsuperscript{st} | AP | mIoU |
|---------------------------|-------|----|-----------------|-----------------|----|------|
| \( \checkmark \) | \( 0.80 \) | \( \mathbf{39.7} \) | \( \mathbf{44.3} \) | \( \mathbf{32.9} \) | \( \mathbf{29.9} \) | \( \mathbf{41.7} \) |
| \( \checkmark \) | \( 0.85 \) | \( \mathbf{40.8} \) | \( \mathbf{46.1} \) | \( \mathbf{32.9} \) | \( \mathbf{31.5} \) | \( \mathbf{41.7} \) |
| \( \checkmark \) | \( 0.90 \) | \( \mathbf{41.3} \) | \( \mathbf{46.9} \) | \( \mathbf{32.9} \) | \( \mathbf{32.1} \) | \( \mathbf{41.7} \) |
| \( \checkmark \) | \( 0.95 \) | \( \mathbf{41.3} \) | \( \mathbf{47.0} \) | \( \mathbf{32.9} \) | \( \mathbf{31.1} \) | \( \mathbf{41.7} \) |
each stage. As presented in Table 1, with the number of convolutions in each head increasing, the network performance improves steadily and achieves the peak PQ with 3 stacked Conv3×3 whose channel number is 256. Similar with [53], deformable convolutions [54] are adopted in position head to extend the receptive field, which brings further improvements, especially in stuff regions (1.4% PQ).

Position embedding. Due to the instance-aware property of objects, position embedding is introduced to provide essential cues. In Table 2, we compare among several positional settings by attaching relative coordinates [30] to different heads. An interesting finding is that the improvement is minor (up to 0.3% PQ) if coordinates are attached to the kernel head or feature encoder only, but it will be boosted to 1.4% PQ when given the positional cues to both heads. It could be attributed to the constructed correspondence in the position between kernel weights and the encoded feature.

Kernel fusion. Kernel fusion is a core operation in the proposed method, which guarantees the required properties for things and stuff, as elaborated in Sec. 3.2. We investigate the fusion type class-aware and similarity thresholds thres in Table 3. As presented in the table, the network attains the best performance with thres 0.90. And the class-agnostic kernel fusion could dismiss some similar instances with different categories, which yields the performance drop in AP.

Feature encoder. To enhance expressiveness of the encoded feature Fe, we further explore the channel number and feature type used in feature encoder. As illustrated in Table 4, the network achieves 41.3% PQ with 64 channels, and extra channels contribute little improvement. For effi-
Table 10. Comparisons with previous methods on the COCO val set. Panoptic FCN-400, 512, and 600 denotes utilizing smaller input instead of the default setting. All of our results are achieved on the same device with single input and no flipping. FPS is measured end-to-end from single input to panoptic result with an average speed over 1,000 images, which could be further improved with more optimizations. The simple enhanced version is marked with *. The model testing by ourselves according to released codes is denoted as †.

| Method                     | Backbone | PQ  | SQ  | RQ  | PQth | SQth | RQth | PQst | SQst | RQst | Device | FPS |
|----------------------------|----------|-----|-----|-----|------|------|------|------|------|------|--------|------|
| Panoptic FPN [17]          | Res50-FPN | 39.0| -   | -   | -    | -    | -    | 28.7 | -    | -    | -      | -    |
| Panoptic FPN†-1×           | Res50-FPN | 39.4| 77.8| 48.3| 45.9 | 80.9 | 55.3 | 29.6 | 73.3 | 37.7 | V100   | 17.5 |
| Panoptic FPN†-3×           | Res50-FPN | 41.5| 79.1| 50.5| 48.3 | 82.2 | 57.9 | 31.2 | 74.4 | 39.5 | V100   | 17.5 |
| AUNet [24]                 | Res50-FPN | 39.6| -   | 49.1| -    | -    | -    | 25.2 | -    | -    | -      | -    |
| CIAE [11]                  | Res50-FPN | 40.2| -   | 45.3| -    | -    | -    | 32.3 | -    | -    | 2080Ti | 12.5 |
| UPSNet† [48]               | Res50-FPN | 42.5| 78.0| 52.5| 48.6 | 79.4 | 59.6 | 33.4 | 75.9 | 41.7 | V100   | 9.1  |
| Unifying [23]              | Res50-FPN | 43.4| 79.6| 53.0| 48.6 | -    | -    | 35.5 | -    | -    | -      | -    |
| box-free                   |          |     |     |     |      |      |      |      |      |      |        |      |
| DeeperLab [49]             | Xception-71 | 33.8| -   | -   | -    | -    | -    | -    | -    | -    | V100   | 10.6 |
| Panoptic-DeepLab [6]       | Res50     | 35.1| -   | -   | -    | -    | -    | -    | -    | -    | V100   | 20.0 |
| AdaptIS [40]               | Res50     | 35.9| -   | 40.3| -    | -    | 29.3 | -    | -    | -    | -      | -    |
| RealTimePan [14]           | Res50-FPN | 37.1| -   | 41.0| -    | -    | 31.3 | -    | -    | -    | V100   | 15.9 |
| PCV [32]                   | Res50-FPN | 37.5| 77.7| 47.2| 40.0 | 78.4 | 50.0 | 33.7 | 76.5 | 42.9 | 1080Ti | 5.7  |
| SOLO V2 [45]               | Res50-FPN | 42.1| -   | 49.6| -    | 30.7 | -    | -    | -    | -    | -      | -    |
| Panoptic FCN-400           | Res50-FPN | 40.7| 80.5| 49.3| 44.9 | 82.0 | 54.0 | 34.3 | 78.1 | 42.1 | V100   | 20.9 |
| Panoptic FCN-512           | Res50-FPN | 42.3| 80.9| 51.2| 47.4 | 82.1 | 56.9 | 34.7 | 79.1 | 42.7 | V100   | 18.9 |
| Panoptic FCN-600           | Res50-FPN | 42.8| 80.6| 51.6| 47.9 | 82.6 | 57.2 | 35.1 | 77.4 | 43.1 | V100   | 16.8 |
| Panoptic FCN               | Res50-FPN | 43.6| 80.6| 52.6| 49.3 | 82.6 | 58.9 | 35.0 | 77.6 | 42.9 | V100   | 12.5 |
| Panoptic FCN*              | Res50-FPN | 44.3| 80.7| 53.0| 50.0 | 83.4 | 59.3 | 35.6 | 76.7 | 43.5 | V100   | 9.2  |

Upper-bound analysis. In Table 9, we give analysis to the upper-bound of generate-kernel-then-segment fashion with Res50-FPN backbone on the COCO val set. As illustrated in the table, given ground truth positions of object centers $L_i^{th}$ and stuff regions $L_i^{st}$, the network yields 6.2% PQ from more precise locations. And it will bring extra boost (16.1% PQ) to the network if we assign ground truth categories to the position head. Compared with the baseline method, there still remains huge potential to be explored (22.3% PQ in total), especially for stuff regions which could have even up to 33.7% PQ and 42.8% mIoU gains.

Speed-accuracy. To illustrate the network efficiency, we plot the end-to-end speed-accuracy trade-off curve on the COCO val set. As presented in Fig. 3, the proposed Panoptic FCN surpasses all previous box-free models by a large margin in both performance and efficiency. Even compared with the well-optimized Panoptic FPN [17] from Detectron2 [47], our approach still attains a better speed-accuracy balance with different image scales. Details about these data points can be found in Table 10.

4.3. Main Result

We further conduct experiments on different scenarios, namely COCO dataset for common context, Cityscapes and Mapillary Vistas datasets for traffic-related environments.
In particular, with COCO models on COCO test-dev set, as depicted in Fig. 3. Moreover, thanks to its simplicity, Panoptic FCN attains a box-free enhanced version is marked with *.

Table 11. Experiments on the COCO test-dev set. All of our results are achieved with single scale input and no flipping. The simple enhanced version and val set for training are marked with * and ‡.

| Method          | Backbone     | PQ | PQth | PQst |
|-----------------|--------------|----|------|------|
| box-based       |              |    |      |      |
| Panoptic FPN [17] | Res101-FPN   | 40.9 | 48.3 | 29.7 |
| CIAE [11]       | DCN101-FPN   | 44.5 | 49.7 | 36.8 |
| AUNet [24]      | ResNeXt152-FPN | 46.5 | 55.8 | 32.5 |
| UPSNet [48]     | DCN101-FPN   | 46.6 | 53.2 | 36.7 |
| Unifying† [23]  | DCN101-FPN   | 47.2 | 53.5 | 37.7 |
| box-free        |              |    |      |      |
| DeeperLab [49]  | Xception-71  | 34.3 | 37.5 | 29.6 |
| SSAP [10]       | Res101-FPN   | 36.9 | 40.1 | 32.0 |
| PCV [42]        | Res50-FPN    | 37.7 | 40.7 | 33.1 |
| Panoptic-DeepLab [6] | Xception-71 | 39.7 | 43.9 | 33.2 |
| AdaptIS [40]    | ResNeXt-101  | 42.8 | 53.2 | 36.7 |
| Axial-DeepLab [43] | Axial-ResNet-L | 43.6 | 48.9 | 35.6 |

| Method          | Backbone     | PQ | PQth | PQst |
|-----------------|--------------|----|------|------|
| Panoptic FPN    | Res101-FPN   | 45.5 | 51.4 | 36.4 |
| Panoptic FPN*   | DCN101-FPN   | 47.0 | 53.0 | 37.8 |
| Panoptic FPN‡   | DCN101-FPN   | 47.1 | 53.2 | 37.8 |
| Panoptic FPN‡‡  | DCN101-FPN   | 47.5 | 53.7 | 38.2 |

Table 12. Experiments on the Cityscapes val set. All of our results are achieved with single scale input and no flipping. The simple enhanced version is marked with *.

| Method          | Backbone     | PQ | PQth | PQst |
|-----------------|--------------|----|------|------|
| box-based       |              |    |      |      |
| Panoptic FPN [17] | Res101-FPN   | 58.1 | 52.0 | 62.5 |
| AUNet [24]      | Res101-FPN   | 59.0 | 54.8 | 62.1 |
| UPSNet [48]     | Res50-FPN    | 59.3 | 54.6 | 62.7 |
| Seamless [36]   | Res50-FPN    | 60.2 | 55.6 | 63.6 |
| Unifying [23]   | Res50-FPN    | 61.4 | 54.7 | 66.3 |
| box-free        |              |    |      |      |
| PCV [42]        | Res50-FPN    | 54.2 | 47.8 | 58.9 |
| DeeperLab [49]  | Xception-71  | 56.5 | -    | -    |
| SSAP [10]       | Res50-FPN    | 58.4 | 50.6 | -    |
| AdaptIS [40]    | Res50        | 59.0 | 55.8 | 61.3 |
| Panoptic-DeepLab [6] | Res50    | 59.7 | -    | -    |

| Method          | Backbone     | PQ | PQth | PQst |
|-----------------|--------------|----|------|------|
| Panoptic FPN    | Res50-FPN    | 59.6 | 52.1 | 65.1 |
| Panoptic FPN*   | Res50-FPN    | 61.4 | 54.8 | 66.6 |

Table 13. Experiments on the Mapillary Vistas val set. All of our results are achieved with single scale input and no flipping. The simple enhanced version is marked with *.

| Method          | Backbone     | PQ | PQth | PQst |
|-----------------|--------------|----|------|------|
| box-based       |              |    |      |      |
| TASCNet [21]    | Res50-FPN    | 32.6 | 31.1 | 34.4 |
| Seamless [36]   | Res50-FPN    | 36.2 | 33.6 | 40.0 |

| Method          | Backbone     | PQ | PQth | PQst |
|-----------------|--------------|----|------|------|
| box-free        |              |    |      |      |
| DeeperLab [49]  | Xception-71  | 32.0 | -    | -    |
| AdaptIS [40]    | Res50        | 32.0 | 26.6 | 39.1 |
| Panoptic-DeepLab [6] | Res50    | 33.3 | -    | -    |

| Method          | Backbone     | PQ | PQth | PQst |
|-----------------|--------------|----|------|------|
| Panoptic FPN    | Res50-FPN    | 34.8 | 30.6 | 40.5 |
| Panoptic FPN*   | Res50-FPN    | 36.9 | 32.9 | 42.3 |

COCO. In Table 10, we conduct experiments on COCO val set. Compared with recent approaches, Panoptic FCN achieves superior performance with efficiency, which surpasses leading box-based [23] and box-free [42] methods over 0.2% and 1.5% PQ, respectively. With simple enhancements, the gap will be enlarged to 0.9% and 2.2% PQ. Moreover, thanks to its simplicity, Panoptic FCN attains a balanced speed-accuracy trade-off, as depicted in Fig. 3.

Table 10. Experiments on Cityscapes val set. All of our results are achieved with single scale input and no flipping. The box-based model surpasses the state-of-the-art approach [23] in box-based stream with 0.3% PQ and attains 47.5% PQ with single scale inputs. Compared with the similar box-free fashion, our method with Res101-FPN improves 1.9% PQ over Axial-DeepLab [43] which adopts stronger backbone.

Cityscapes. To further evaluate the effectiveness, we conduct experiments on Cityscapes val set. As shown in Table 12, Panoptic FCN exceeds the top box-free model [6] with 1.7% PQ and attains 61.4% PQ with single scale inputs. Even compared with the leading box-based model [23], which utilizes Lovas loss for further optimization, the proposed method still achieves comparable performance.

Mapillary Vistas. In Table 13, we compare with other state-of-the-art models on the large-scale Mapillary Vistas val set with Res50-FPN backbone. As presented in the table, the proposed Panoptic FCN exceeds previous box-free methods by a large margin in both things and stuff. Specifically, Panoptic FCN surpasses the leading box-based model [23] and box-free [6] models with 0.7% and 3.6% PQ, and attains 36.9% PQ with single enhancement in the feature encoder.

5. Conclusion

We have presented the Panoptic FCN, a conceptually simple yet effective framework for panoptic segmentation. The key difference from prior works lies in that we represent and predict things and stuff in a fully convolutional manner. To this end, kernel generator and kernel fusion are proposed to generate the unique kernel weight for each object instance or semantic category. With the high-resolution feature produced by feature encoder, predictions could be achieved by convolutions directly. Meanwhile, instance-awareness and semantic-consistency for things and stuff can be respectively satisfied with the designed workflow.

Due to the simplicity, efficiency, and high performance, we hope the Panoptic FCN can serve as a strong foundation for the unified representation and benefit future research.
References

[1] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. Yolact: Real-time instance segmentation. In ICCV, 2019.
[2] Liang-Chieh Chen, George Papandreou, Florian Schroff, and Hartwig Adam. Rethinking atrous convolution for semantic image segmentation. arXiv:1706.05587, 2017.
[3] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. In ECCV, 2018.
[4] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. TensorMask: A foundation for dense object segmentation. In ICCV, 2019.
[5] Yifeng Chen, Guangchen Lin, Songyuan Li, Omar Bourahla, Yiming Wu, Fangfang Wang, Junyi Feng, Mingliang Xu, and Xi Li. Banet: Bidirectional aggregation network with occlusion handling for panoptic segmentation. In CVPR, 2020.
[6] 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 CVPR, 2017.
[7] François Chollet. Xception: Deep learning with depthwise separable convolutions. In CVPR, 2017.
[8] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In CVPR, 2016.
[9] Jun Fu, Jing Liu, Haijie Tian, Yong Li, Yongjun Bao, Zhiwei Fang, and Hanqing Lu. Dual attention network for scene segmentation. In CVPR, 2019.
[10] Naiyu Gao, Yanhu Shan, Yupei Wang, Xin Zhao, Yinan Yu, Ming Yang, and Kaiqi Huang. Ssap: Single-shot instance segmentation with affinity pyramid. In ICCV, 2019.
[11] Naiyu Gao, Yanhu Shan, Xin Zhao, and Kaiqi Huang. Learning category-and-instance-aware pixel embedding for fast panoptic segmentation. arXiv:2009.13342, 2020.
[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In ICCV, 2017.
[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In CVPR, 2016.
[14] Rui Hou, Jie Li, Arjun Bhargava, Allan Raventos, Vitor Guizilini, Chao Fang, Jerome Lynch, and Adrien Gaidon. Real-time panoptic segmentation from dense detections. In CVPR, 2020.
[15] Han Hu, Jiayuan Gu, Zheng Zhang, Jifeng Dai, and Yichen Wei. Relation networks for object detection. In CVPR, 2018.
[16] Zilong Huang, Xingang Wang, Lichao Huang, Chang Huang, YUNCHAO WEI, and Wenwu Yu. Ccenet: Criss-cross attention for semantic segmentation. In ICCV, 2019.
[17] Alexander Kirillov, Ross Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In CVPR, 2019.
[18] Alexander Kirillov, Kaiming He, Ross Girshick, Carsten Rother, and Piotr Dollár. Panoptic segmentation. In CVPR, 2019.
[19] Hei Law and Jia Deng. Cornernet: Detecting objects as paired keypoints. In ECCV, 2018.
[20] Youngwan Lee and Jongyoul Park. Centermask: Real-time anchor-free instance segmentation. In CVPR, 2020.
[21] Jie Li, Allan Raventos, Arjun Bhargava, Takaaki Tagawa, and Adrien Gaidon. Learning to fuse things and stuff. arXiv:1812.01192, 2018.
[22] Qizhu Li, Anurag Arnab, and Philip HS Torr. Weakly-and semi-supervised panoptic segmentation. In ECCV, 2018.
[23] Qizhu Li, Xiaojuan Qi, and Philip HS Torr. Unifying training and inference for panoptic segmentation. In CVPR, 2020.
[24] Yanwei Li, Xinzhe Chen, Zheng Zhu, Lingxi Xie, Guan Huang, Dalong Du, and Xingang Wang. Attention-guided unified network for panoptic segmentation. In CVPR, 2019.
[25] Yanwei Li, Lin Song, Yukang Chen, Zeming Li, Xiangyu Zhang, Xingang Wang, and Jian Sun. Learning dynamic routing for semantic segmentation. In CVPR, 2020.
[26] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharat Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In CVPR, 2017.
[27] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In ICCV, 2017.
[28] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In ECCV, 2014.
[29] Chenxi Liu, Liang-Chieh Chen, Florian Schroff, Hartwig Adam, Wei Hua, Alan L Yuille, and Li Fei-Fei. Auto-deeplab: Hierarchical neural architecture search for semantic image segmentation. In CVPR, 2019.
[30] Rosanne Liu, Joel Lehman, Piero Molino, Felipe Petroski Such, Eric Frank, Alex Sergeev, and Jason Yosinski. An intriguing failing of convolutional neural networks and the coordconv solution. In NeurIPS, 2018.
[31] Shu Liu, Jiaya Jia, Sanja Fidler, and Raquel Urtasun. Sgn: Sequential grouping networks for instance segmentation. In ICCV, 2017.
[32] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In CVPR, 2018.
[33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In CVPR, 2015.
[34] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In 3DV, 2016.
[35] Gerhard Neuhäusler, Tobias Ollmann, Samuel Rota Bulo, and Peter Kontschieder. The mapillary vistas dataset for semantic understanding of street scenes. In ICCV, 2017.
[36] Lorenzo Porzi, Samuel Rota Bulo, Aleksander Colovic, and Pietro Perona. The cityscapes dataset for semantic urban scene understanding. In ICCV, 2017.
[37] Lu Qi, Jiaya Jia, Sanja Fidler, and Jia Deng. Cornernet: Detecting objects as paired keypoints. In ECCV, 2018.
[39] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In NeurIPS, 2015.

[40] Konstantin Sofiiuk, Olga Barinova, and Anton Konushin. Adaptis: Adaptive instance selection network. In ICCV, 2019.

[41] Lin Song, Yanwei Li, Zeming Li, Gang Yu, Hongbin Sun, Jian Sun, and Nanning Zheng. Learnable tree filter for structure-preserving feature transform. In NeurIPS, 2019.

[42] Haochen Wang, Ruotian Luo, Michael Maire, and Greg Shakhnarovich. Pixel consensus voting for panoptic segmentation. In CVPR, 2020.

[43] Lin Song, Yanwei Li, Zeming Li, Gang Yu, Hongbin Sun, Jian Sun, and Nanning Zheng. Learnable tree filter for structure-preserving feature transform. In NeurIPS, 2019.

[44] Haochen Wang, Ruotian Luo, Michael Maire, and Greg Shakhnarovich. Pixel consensus voting for panoptic segmentation. In CVPR, 2020.

[45] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In ECCV, 2020.

[46] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. Solo: Segmenting objects by locations. In NeurIPS, 2020.

[47] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic, faster and stronger. In NeurIPS, 2020.

[48] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In CVPR, 2019.

[49] Yuxin Wu and Kaiming He. Group normalization. In ECCV, 2018.

[50] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In CVPR, 2019.

[51] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In CVPR, 2019.

[52] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In CVPR, 2019.

[53] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In CVPR, 2019.

[54] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In CVPR, 2019.
A. Experimental Details

Herein, we provide more technical details of the training and inference process in the proposed Panoptic FCN.

A.1. Training Details

**Position head.** To better optimize the position head, we have explored different types of center assigning strategy. For object center generation, mass center is utilized to provide the $k$-th ground-truth coordinate $\tilde{x}_k$ and $\tilde{y}_k$ in Eq. (1) of the main paper. As presented in Table 14, compared with the box center for ground-truth generation, we find that mass center brings superior performance and higher robustness, especially in the designed weighted dice loss, which samples 7 top-scoring points inside each object. It could be attributed to that most of mass centers are located within the object area, while it is not the case for box centers. Moreover, the object size-adaptive deviation $\sigma_k$ in Eq. (1) of the main paper is set to $(2r+1)/3$, where $r$ denotes the gaussian radius of the $k$-th object similar to that in CornerNet [19].

**Weighted dice loss.** For objects, we select $k$ positions whose scores are predicted to be top-$k$ within the object region of $L_{i,c}$, where $c$ is the annotated class. This procedure utilizes $k$ top-scoring kernels to represent the same object and generates $k$ results of prediction $P_j$. Thus, we generate the same ground-truth $Y_{seg}^k$ for $k$ results in $P_j$, which can be optimized with Eq. (7) in the main paper. For stuff regions, we merge all positions with the same category in each stage using AvgPool, which brings a specific kernel. Hence, the factor $k$ can be viewed as 1 for each stuff region.

| weighted | center type | PQ | PQ$^{th}$ | PQ$^{rt}$ | AP  | mIoU |
|----------|-------------|----|---------|---------|-----|------|
| X        | box         | 39.7| 44.7   | 32.3   | 30.3| 41.2 |
| X        | mass        | 40.2| 45.5   | 32.4   | 31.0| 41.3 |
| ✓        | box         | 40.6| 45.7   | 32.8   | 31.7| 41.5 |
| ✓        | mass        | 41.3| 46.9   | 32.9   | 32.1| 41.7 |

**A.2. Inference Details.**

Due to the variation of scale distribution among datasets, we simply modify some parameters in the inference stage. In particular, for COCO dataset, we preserve 100 top-scoring kernels for object prediction and utilize threshold 0.4 to convert soft masks to binary results, as illustrated in Sec. 3.4 of the main paper. For Cityscapes and Mapillary Vistas datasets, 200 kernels with top predicted scores are kept for object prediction, and cosine threshold 0.95 and mask threshold 0.5 are utilized to fuse repetitive kernels and convert binary masks, respectively. Meanwhile, a similar strategy with that in SOLO [44, 45] is adopted to adjust predicted object scores. Furthermore, our codes for training and inference will be released to provide more details.

B. Qualitative Results

We further visualize qualitative results of Panoptic FCN on several datasets with common context and traffic-related scenarios, i.e., COCO, Cityscapes, and Mapillary Vistas.

**COCO.** As presented in Fig. 4, Panoptic FCN gives detailed characterization to the daily environment. Thanks to the pixel-by-pixel handling manner and unified representation, details in foreground things and background stuff can be preserved. Moreover, the proposed approach also validates its effectiveness on objects with various scales in Fig. 4.

**Cityscapes.** As for the street view, we visualize panoptic results on the Cityscapes val set, as illustrated in Fig. 5. In addition to the well-depicted cars and pedestrians, the proposed approach achieves satisfactory performance on slender objects, like street lamps and traffic lights.

**Mapillary Vistas.** In Fig. 6, we further present panoptic results on the Mapillary Vistas val set, which contains larger scales traffic-related scenes. It is clear in the figure that the proposed Panoptic FCN achieves surprising results, especially on vehicles and traffic signs. The coherence of qualitative results also reflects the fulfillment of instance-awareness and semantic-consistency in Panoptic FCN.
Figure 4. Visualization of panoptic results on the COCO val set.
Figure 5. Visualization of panoptic results on the Cityscapes val set.

Figure 6. Visualization of panoptic results on the Mapillary Vistas val set.