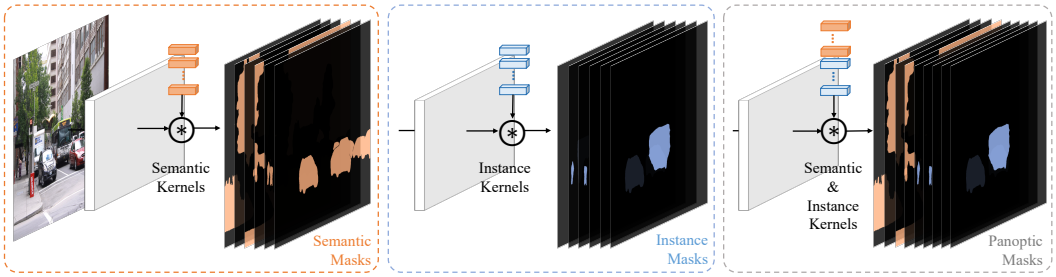
K-Net: Towards Unified Image Segmentation

Anonymous Author(s)
Affiliation
Address
email

Abstract

Semantic, instance, and panoptic segmentations have been addressed using different and specialized frameworks despite their underlying connections. This paper presents a unified, simple, and effective framework for these essentially similar tasks. The framework, named **K-Net**, segments both instances and semantic categories consistently by a group of learnable *kernels*, where each kernel is responsible for generating a mask for either a potential instance or a stuff class. To remedy the difficulties of distinguishing various instances, we propose a kernel update strategy that enables each kernel dynamic and conditional on its meaningful group in the input image. K-Net can be trained in an end-to-end manner with bipartite matching, and its training and inference are naturally NMS-free and box-free. Without bells and whistles, K-Net surpasses all previous state-of-the-art single-model results of panoptic segmentation on MS COCO and semantic segmentation on ADE20K. Its instance segmentation performance is also on par with Cascade Mask R-CNN on MS COCO with 60%-90% faster inference speeds. Code and models will be released.

1 Introduction



(a) FCN for Semantic Segmentation Framework (b) K-Net for Instance Segmentation (c) K-Net for Panoptic Segmentation

Figure 1: The semantic segmentation (a), the instance (b), and the panoptic segmentation (c) tasks are unified as a common framework in this paper. In conventional semantic segmentation, each kernel corresponds to a semantic class. Our framework extends this notion to make each kernel corresponds to either a potential instance or a semantic class.

Image segmentation aims at finding groups of coherent pixels [43]. There are different notions in groups, such as semantic categories (*e.g.*, car, dog, cat) or instances (*e.g.*, objects that coexist in the same image). Based on the different segmentation targets, the tasks are termed differently, *i.e.*, semantic and instance segmentation, respectively. Kirillov *et al* [26] propose panoptic segmentation to join the two tasks as a coherent scene segmentation task.

Grouping pixels according to semantic categories can be formulated as a dense classification problem. As shown in Fig. 1-(a), recent methods directly learn a set of kernels (namely *semantic kernels* in

Submitted to 35th Conference on Neural Information Processing Systems (NeurIPS 2021). Do not distribute.

this paper) of pre-defined categories and use them to classify pixels [35] or regions [20]. Such a framework is elegant and straightforward. However, extending this notion to instance segmentation is non-trivial given the varying number of instances across images. Consequently, instance segmentation is tackled by more complicated frameworks with additional steps such as object detection [20] or embedding generation [39]. These methods rely on extra components, which must guarantee the accuracy of extra components to a reasonable extent, or demand complex post-processing such as Non-Maximum Suppression (NMS) and pixel grouping. Recent approaches [30, 44, 47] generate kernels from dense feature grids and then select kernels for segmentation to simplify the frameworks. Nonetheless, since they build upon dense grids to enumerate and select kernels, these methods still rely on hand-crafted post-processing to eliminate masks or kernels of duplicated instances.

In this paper, we make the first attempt to formulate a unified and effective framework to bridge the seemingly different image segmentation tasks (semantic, instance, and panoptic) through the notion of *kernels*. Our method is dubbed as K-Net ('K' stands for kernels). It begins with a set of kernels that are randomly initialized, and learns the kernels in accordance to the segmentation targets at hand, namely, *semantic kernels* for semantic categories and *instance kernels* for instance identities (Fig. 1-(b)). A simple combination of semantic kernels and instance kernels allows panoptic segmentation naturally (Fig. 1-(c)). In the forward pass, the kernels perform convolution on the image features to obtain the corresponding segmentation predictions.

The versatility and simplicity of K-Net are made possible through two novel designs. First, we formulate K-Net so that it dynamically updates the kernels to make them conditional to their activations on the image. Such a content-aware mechanism is crucial to ensure that each kernel, especially an instance kernel, responds accurately to varying objects in an image. Through applying this adaptive kernel update strategy iteratively, K-Net significantly improves the discriminative ability of the kernels and boosts the final segmentation performance. It is noteworthy that this strategy universally applies to kernels for all the segmentation tasks.

Second, inspired by recent advances in object detection [4], we adopt the bipartite matching strategy [42] to assign learning targets for each kernel. This training approach is advantageous to conventional training strategies [32, 41] as it builds a one-to-one mapping between kernels and instances in an image. It thus resolves the problem of dealing with a varying number of instances in an image. In addition, it is purely mask-driven without involving boxes. Hence, K-Net is naturally NMS-free and box-free, which is appealing to real-time applications.

To show the effectiveness of the proposed unified framework on different segmentation tasks, we conduct extensive experiments on COCO dataset [33] for panoptic and instance segmentation, and ADE20K dataset [61] for semantic segmentation. Without bells and whistles, K-Net surpasses all previous state-of-the-art single-model results on panoptic and semantic segmentation benchmarks and achieves competitive performance compared to the more expensive Cascade Mask R-CNN [3]. We further analyze the learned kernels and find that instance kernels incline to specialize on objects at specific locations of similar sizes.

2 Related Work

Semantic Segmentation. Contemporary semantic segmentation approaches typically build upon a fully convolutional network (FCN) [35] and treat the task as a dense classification problem. Based on this framework, many studies focus on enhancing the feature representation through dilated convolution [7–9], pyramid pooling [49, 58], context representations [55, 56], and attention mechanisms [54, 59]. Recently, SETR [60] reformulates the task as a sequence-to-sequence prediction task by using a vision transformer [16]. Despite the different model architectures, the approaches above share the common notion of making predictions via static semantic kernels. Differently, the proposed K-Net makes the kernels dynamic and conditional on their activations in the image.

Instance Segmentation. There are two representative frameworks for instance segmentation – 'top-down' and 'bottom-up' approaches. 'Top-down' approaches [2, 14, 20, 29] first detect accurate bounding boxes and generate a mask for each box. Mask R-CNN [20] simplifies this pipeline by directly adding a FCN [35] in Faster R-CNN [41]. Extensions of this framework add a mask scoring branch [22] or adopt a cascade structure [3, 5]. 'Bottom-up' methods [1, 27, 38, 39] first perform semantic segmentation then group pixels into different instances. These methods usually require a grouping process, and their performance often appears inferior to 'top-down' approaches in popular

benchmarks [13, 33]. Unlike all these works, K-Net performs segmentation and instance separation simultaneously by constraining each kernel to predict one mask at a time for one object. Therefore, K-Net needs neither bounding box detection nor grouping process. It focuses on refining kernels rather than refining bounding boxes, different from previous cascade methods [3, 5].

Recent attempts [10, 46, 47, 50] perform instance segmentation in one stage without involving detection nor embedding generation. These methods apply dense mask prediction using dense sliding windows [10] or dense grids [46]. Some studies explore polar [50] representation, contour [40], and explicit shape representation [52] of instance masks. These methods all rely on NMS to eliminate duplicated instance masks, which hinders end-to-end training. The heuristic process is also unfavorable for real-time applications. Instance kernels in K-Net are trained in an end-to-end manner with bipartite matching and set prediction loss, thus, our methods does not need NMS.

Panoptic Segmentation. Panoptic segmentation [26] combines instance and semantic segmentation to provide a richer understanding of the scene. Different strategies have been proposed to cope with the instance segmentation task. Mainstream frameworks add a semantic segmentation branch [25, 28, 47, 51] on an instance segmentation framework or adopt different pixel grouping strategies [11, 53] based on a semantic segmentation method. Recently, DETR [4] tries to simplify the framework by transformer [45] but need to predict boxes around both stuff and things classes in training for assigning learning targets. These methods either need object detection or embedding generation to separate instances, which does not reconcile the instance and semantic segmentation in a unified framework. By contrast, K-Net partitions an image into semantic regions by semantic kernels and object instances by instance kernels through a unified perspective of kernels.

Dynamic Kernels. Convolution kernels are usually static, *i.e.*, agnostic to the inputs, and thus have limited representation ability. Previous works [15, 17, 23, 24, 62] explore different kinds of dynamic kernels to improve the flexibility and performance of models. Some semantic segmentation methods apply dynamic kernels to improve the model representation with enlarged receptive fields [48] or multi-scales contexts [18]. Differently, K-Net uses dynamic kernels to improve the discriminative capability of the segmentation kernels more so than the input features of kernels.

Recent studies apply dynamic kernels to generate instance [44, 47] or panoptic [30] segmentation predictions directly. Because these methods generate kernels from dense feature maps, enumerate kernels of each position, and filter out kernels of background regions, they either still rely on NMS [44, 47] or need extra kernel fusion [30] to eliminate kernels or masks of duplicated objects. Instead of generated from dense grids, the kernels in K-Net are a set of learnable parameters updated by their corresponding contents in the image. K-Net does not need to handle duplicated kernels because its kernels learn to focus on different regions of the image in training, constrained by the bipartite matching strategy that builds a one-to-one mapping between the kernels and instances.

3 Methodology

We consider various segmentation tasks through a unified perspective of kernels. The proposed K-Net uses a set of kernels to assign each pixel to either a potential instance or a semantic class (Sec. 3.1). To enhance the discriminative capability of kernels, we contribute a way to update the static kernels by the contents in their partitioned pixel groups (Sec. 3.2). We adopt the bipartite matching strategy to train instance kernels in an end-to-end manner (Sec. 3.3). K-Net can be applied seamlessly to semantic, instance, and panoptic segmentation as described in Sec. 3.4.

3.1 K-Net

Despite the different definitions of a 'meaningful group', all segmentation tasks essentially assign each pixel to one of the predefined meaningful groups [43]. As the number of groups in an image is typically assumed finite, we can set the maximum group number of a segmentation task as N. For example, there are N pre-defined semantic classes for semantic segmentation or at most N objects in an image for instance segmentation. For panoptic segmentation, N is the total number of stuff classes and objects in an image. Therefore, we can use N kernels to partition an image into N groups, where each kernel is responsible to find the pixels belonging to its corresponding group. Specifically, given an input feature map $F \in R^{B\times C\times H\times W}$ of B images, produced by a deep neural network, we only need N kernels $K \in R^{N\times C}$ to perform convolution with F to obtain the corresponding segmentation

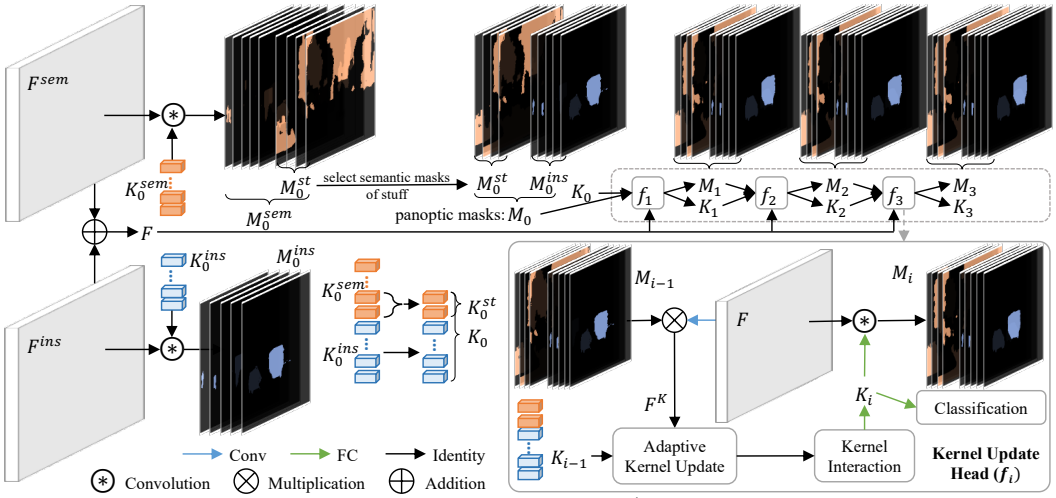


Figure 2: **K-Net for panoptic segmentaion.** Instance kernels K^{ins} and semantic kernels K^{sem} first perform convolution with the feature maps F^{ins} and F^{sem} to obtain masks M^{ins} and M^{sem}, respectively. The semantic masks M^{st} of stuff categories and all the instance masks M^{ins} form the panoptic masks M_0. The semantic kernels K^{st} of stuff categories and all the instance kernels K^{ins} form the panoptic kernels K_0. Then the kernel update head assembles group features of each kernel and use the features to adaptively update the kernels. After kernel interaction the new kernels will be used to produce more accurate masks. This process is performed iteratively to progressively refine the kernels and the mask prediction. The framework can perform instance segmentation or semantic segmentation by simply remove K^{sem} or K^{ins} under the same pipeline.

prediction $M \in R^{B\times N\times H\times W}$ as

$$M = K * F, \tag{1}$$

where C, H, and W are the number of channels, height, and width of the feature map, respectively.

This formulation has already dominated semantic segmentation for years [8, 35, 58]. In semantic segmentation, each kernel is responsible to find all pixels of a similar class across images. Whereas in instance segmentation, each pixel group corresponds to an object. However, previous methods separate instances by extra steps [20, 27, 39] instead of by kernels.

This paper is the first study that explores if the notion of kernels in semantic segmentation is equally applicable to instance segmentation, and more generally panoptic segmentation. To separate instances by kernels, each kernel in K-Net only segments at most one object in an image (Fig. 1-(b)). In this way, K-Net distinguishes instances and performs segmentation simultaneously, achieving instance segmentation in one pass without extra steps. For simplicity, we call these kernels as *semantic* and *instance* kernels in this paper for semantic and instance segmentation, respectively. A simple combination of instance kernels and semantic kernels can naturally preform panoptic segmentation that either assigns a pixel to an instance ID or a class of stuff (Fig. 1-(c)).

3.2 Group-Aware Kernels

Despite the simplicity of K-Net, separating instances directly by kernels is non-trivial. Because instance kernels need to discriminate objects that vary in scale and appearance within and across images. Without a common and explicit characteristic like semantic categories, the instance kernels need stronger discriminative ability than static kernels.

To overcome this challenge, we contribute an approach to make the kernel conditional on their corresponding pixel groups, through a kernel update head shown in Fig. 2. The kernel update head f_i contains three key steps: group feature assembling, adaptive kernel update, and kernel interaction. Firstly, the group feature F^K for each pixel group is assembled using the mask prediction M_{i-1}. As it is the content of each individual groups that distinguishes them from each other, F^K is used to update their corresponding kernel K_{i-1} adaptively. After that, the kernel interacts with each other to comprehensively model the image context. Finally, the obtained group-aware kernels K_i perform convolution over feature map F to obtain more accurate mask prediction M_i. This process can be conducted iteratively because a finer partition usually reduces the noise in group features, which

results in more discriminative kernels. This process is formulated as

$$K_i, M_i = f_i\left(M_{i-1}, K_{i-1}, F\right). \tag{2}$$

Notably, the kernel update head and the iterative refinement is universal as it does not rely on the characteristic of kernels. Thus, it can enhance not only instance kernels but also semantic kernels. We detail the three steps as follows.

Group Feature Assembling. The kernel update head first assembles the features of each group, which will be adopted later to make the kernels group-aware. As the mask of each kernel in M_{i-1} essentially defines whether or not a pixel belongs to the kernel's related group, we can assemble the feature F^K for K_{i-1} by multiplying the feature map F with the M_{i-1} as

$$F^K = \sum_i^H \sum_j^W M_{i-1}(i,j) \cdot F(i,j), F^K \in R^{B\times N\times C}, \tag{3}$$

where B is the batch size, N is the number of kernels, and C is the number of channels.

Adaptive Feature Update. The kernel update head then updates the kernels using the obtained F^K to improve the representation ability of kernels. As the mask M_{i-1} may not be accurate, which is more common the case, the feature of each group may also contain noises introduced by pixels from other groups. To reduce the adverse effect of the noise in group features, we devise an adaptive kernel update strategy. Specifically, we first conduct element-wise multiplication between F^K and K_{i-1} as

$$F^G = \phi_1(F^K) \otimes \phi_2(K_{i-1}), F^G \in R^{B\times N\times C}, \tag{4}$$

where ϕ_1 and ϕ_2 are linear transformations. Then the head learns two gates, G^F and G^K, which adapt the contribution from F^K and K_{i-1} to the updated kernel $\tilde{K}$, respectively. The formulation is

$$\begin{aligned} G^K &= \sigma(\psi_1(F^G)), G^F = \sigma(\psi_2(F^G)), \\ \tilde{K} &= G^F \otimes \psi_3(F^K) + G^K \otimes \psi_4(K_{i-1}), \end{aligned} \tag{5}$$

where $\psi_n, n = 1, ..., 4$ are different fully connected (FC) layers and σ is the Sigmoid function. $\tilde{K}$ is then used in kernel interaction.

Kernel Interaction. Interaction among kernels is important to inform each kernel with contextual information from other groups. Such information allows the kernel to implicitly model and exploit the relationship between groups of an image. To this end, we add a kernel interaction process to obtain the new kernels K_i given the updated kernels $\tilde{K}$. Here we simply adopt Multi-Head Attention [45] followed by a Feed-Forward Neural Network, which has been proven effective in previous works [4, 45]. The output K_i of kernel interaction is then used to generate a new mask prediction through $M_i = g_i\left(K_i\right) * F$. K_i will also be used to predict classification scores in instance and panoptic segmentation.

3.3 Training Instance Kernels

While each semantic kernel can be assigned to a constant semantic class, there lacks an explicit rule to assign varying number of targets to instance kernels. In this work, we adopt bipartite matching strategy and set prediction loss [4, 42] to train instance kernels in an end-to-end manner. Different from previous works [4, 42] that rely on boxes, the learning of instance kernels is purely mask-driven because the inference of K-Net is naturally box-free.

Loss Functions. The loss function for instance kernels is written as

$$L_K = \lambda_{cls} L_{cls} + \lambda_{ce} L_{ce} + \lambda_{dice} L_{dice}, \tag{6}$$

where L_{cls} is Focal loss [32] for classification, and L_{ce} and L_{dice} are CrossEntropy (CE) loss and Dice loss [37] for segmentation, respectively. Given that each instance only occupies a small region in an image, CE loss is insufficient to handle the highly imbalanced learning targets of masks. Therefore, we apply Dice loss [37] to handle this issue following previous works [44, 46, 47].

Mask-based Hungarian Assignment. We adopt Hungarian assignment strategy used in [4, 42] for target assignment to train K-Net in an end-to-end manner. It builds a one-to-one mapping between the predicted instance masks and the ground-truth (GT) instances based on the matching costs. The matching cost is calculated between the mask and GT pairs in a similar manner as the training loss.

3.4 Applications to Various Segmentation Tasks

Panoptic Segmentation. For panoptic segmentation, the kernels are composed of instance kernels K_0^{ins} and semantic kernels K_0^{sem} as shown in Fig. 2. We adopt semantic FPN [25] for producing high resolution feature map F, except that we add positional encoding used in [4, 45, 63] to enhance the positional information. As semantic segmentation mainly relies on semantic information for per-pixel classification, while instance segmentation prefers accurate localization information to separate instances, we use two separate branches to generate the features F^{ins} and F^{sem} to perform convolution with K_0^{ins} and K_0^{sem} for generating instance and semantic masks M_0^{ins} and M_0^{sem}, respectively. This design is consistent with previous practices [25, 47].

We then construct M_0, K_0, and F as the inputs of kernel update head to dynamically update the kernels and refine the panoptic mask prediction. Because 'things' are already separated by instance masks in M_0^{ins}, while M_0^{sem} contains the semantic masks of both 'things' and 'stuff', we select M_0^{st}, the masks of stuff categories from M_0^{sem}, and directly concatenate it with M_0^{ins} to form the panoptic mask prediction M_0. Due to similar reason, we only select and concatenate the kernels of stuff classes in K_0^{sem} with K_0^{ins} to form the panoptic kernels K_0. To exploit the complementary semantic information in F^{sem} and localization information in F^{ins}, we add them together to obtain F as the input feature map of the kernel update head. With the constructed M_0, K_0, and F, the kernel update head can start produce group-aware kernels $K_s, s = \{1, 2, ..., S\}$ iteratively by S times and obtain the refined mask prediction M_S. We follow Panoptic FPN [25] to merge masks of things and stuff in M_S based on their classification scores for the final panoptic segmentation results.

Instance Segmentation. In the similar framework, we simply remove the concatenation process of kernels and masks to perform instance segmentation. We did not remove the semantic segmentation branch as the semantic information is still complementary for instance segmentation. Note that in this case, the semantic segmentation branch does not use extra annotations. The ground truth of semantic segmentation is built by converting instance masks to their corresponding class labels.

Semantic Segmentation. As K-Net does not rely on specific architectures of model representation, K-Net can perform semantic segmentation by simply appending its kernel update head to any existing semantic segmentation methods [8, 35, 49, 58] that rely on semantic kernels.

4 Experiments

Dataset and Metrics. For panoptic and instance segmentation, we perform experiments on the challenging COCO dataset [33]. All models are trained on the `train2017` split and evaluated on the `val2017` split. The panoptic segmentation results are evaluated by the PQ metric [26]. We also report the performance of thing and stuff, noted as PQ^{Th}, PQ^{St}, respectively, for thorough evaluation. The instance segmentation results are evaluated by mask AP [33]. The AP for small, medium and large objects are noted as AP_s, AP_m, and AP_l, respectively. The AP at mask IoU thresholds 0.5 and 0.7 are also reported as AP_{50} and AP_{70}, respectively. For semantic segmentation, we conduct experiments on the challenging ADE20K dataset [61] and report mIoU to evaluate the segmentation quality. All models are trained on the `train` split and evaluated on the `validation` split.

Implementation Details. For panoptic and instance segmentation, we implement K-Net with MMDetection [6]. In the ablation study, the model is trained with a batch size of 16 for 12 epochs. The learning rate is 0.0001, and it is decreased by 0.1 after 8 and 11 epochs, respectively. We use AdamW [36] with a weight decay of 0.05. For data augmentation in training, we adopt horizontal flip augmentation with a single scale. The long edge and short edge of images are resized to 1333 and 800, respectively, without changing the aspect ratio. When comparing with other frameworks, we use multi-scale training with a longer schedule (36 epochs) for fair comparisons [6]. The short edge of images is randomly sampled from $[640, 800]$ [19].

For semantic segmentation, we implement K-Net with MMSegmentation [12] and train it with 80,000 iterations. As AdamW [36] empirically works better than SGD, we use AdamW with a weight decay of 0.0005 by default on both the baselines and K-Net for a fair comparison. The initial learning rate is 0.0001, and it is decayed by 0.1 after 60000 and 72000 iterations, respectively. More details are provided in the supplementary material.

Model Hyperparameters. In the ablation study, we adopt ResNet-50 [21] backbone with FPN [31]. For panoptic and instance segmentation, we use $\lambda_{cls} = 2$ for Focal loss following previous meth-

Table 1: Comparisons with state-of-the-art panoptic segmentation methods on COCO dataset

Framework	Backbone	Box-free	NMS-free	Epochs	PQ	PQ^{Th}	PQ^{St}
			`val`				
Panoptic-DeepLab [11]	Xception-71			$\sim$1000	39.7	43.9	33.2
Panoptic FPN [25]	R50-FPN			36	41.5	48.5	31.1
SOLOv2 [47]	R50-FPN	✓		36	42.1	49.6	30.7
DETR [4]†	R50		✓	$\sim$150 + 25	43.4	48.2	36.3
Unifying [28]	R50-FPN			$\sim$27	43.4	48.6	35.5
Panoptic FCN [30]	R50-FPN	✓	✓	36	43.6	49.3	35.0
K-Net	R50-FPN	✓	✓	36	**45.1**	**50.3**	**37.3**
			`test-dev`				
Panoptic-DeepLab	Xception-71	✓		$\sim$1000	41.4	45.1	35.9
Panoptic FPN	R101-FPN			36	43.5	50.8	32.5
Panoptic FCN	R101-FPN	✓	✓	36	45.5	51.4	36.4
DETR	R101		✓	$\sim$150 + 25	46.0	-	-
UPSNet [51]	R101-FPN-DCN			36	46.6	53.2	36.7
Unifying [28]	R101-FPN-DCN			$\sim$27	47.2	53.5	37.7
K-Net	R101-FPN	✓	✓	36	47.0	52.8	38.2
K-Net	R101-FPN-DCN	✓	✓	36	**48.3**	**54.0**	**39.7**
K-Net	Swin-L [34]	✓	✓	36	**52.1**	**58.2**	**42.8**

Table 2: Comparisons with state-of-the-art instance segmentation methods on COCO dataset

Method	Backbone	Box-free	NMS-free	Epochs	AP	AP_{50}	AP_{70}	AP_s	AP_m	AP_l	FPS
				`val2017`							
SOLO [46]	R-50-FPN	✓		36	35.8	56.7	37.9	14.3	39.3	53.2	12.7
Mask R-CNN [20]	R-50-FPN			36	37.1	58.5	39.7	18.7	39.6	53.9	17.5
SOLOv2 [47]	R-50-FPN	✓		36	37.5	58.2	40.0	15.8	41.4	56.6	17.7
CondInst [44]	R-50-FPN			36	37.5	58.5	40.1	18.7	41.0	53.3	14.0
Cascade Mask R-CNN [3]	R-50-FPN			36	38.5	59.7	**41.8**	**19.3**	41.1	55.6	10.3
K-Net	R-50-FPN	✓	✓	36	37.8	60.3	39.9	16.9	41.2	57.5	**21.2**
K-Net-N256	R-50-FPN	✓	✓	36	**38.6**	**60.9**	41.0	19.1	**42.0**	**57.7**	19.8
				`test-dev`							
TensorMask [10]	R-50-FPN	✓		72	35.4	57.2	37.3	16.3	36.8	49.3	-
SOLO	R-50-FPN	✓		72	36.8	58.6	39.0	15.9	39.5	52.1	12.7
Mask R-CNN	R-50-FPN			36	37.4	59.5	40.1	18.6	39.8	51.6	17.5
CondInst	R-50-FPN			36	37.8	59.2	40.4	18.2	40.3	52.7	14.0
SOLOv2	R-50-FPN	✓		36	38.2	59.3	40.9	16.0	41.2	55.4	17.7
Cascade Mask R-CNN	R-50-FPN			36	38.8	60.4	**42.0**	**19.4**	40.9	53.9	10.3
K-Net	R-50-FPN	✓	✓	36	38.4	61.2	40.9	17.4	40.7	56.2	**21.2**
K-Net-N256	R-50-FPN	✓	✓	36	**39.1**	**61.7**	41.8	18.2	**41.4**	**56.6**	19.8
TensorMask	R-101-FPN	✓		72	37.1	59.3	39.4	17.4	39.1	51.6	-
SOLO	R-101-FPN	✓		72	37.8	59.5	40.4	16.4	40.6	54.2	10.7
Mask R-CNN	R-101-FPN			36	38.8	60.8	41.8	19.1	41.2	54.3	14.3
CondInst	R-101-FPN			36	38.9	60.6	41.8	18.8	41.8	54.4	11.0
SOLOv2	R-101-FPN	✓		36	39.5	60.8	42.6	16.7	43.0	57.4	14.3
Cascade Mask R-CNN	R-101-FPN			36	39.9	61.6	43.3	**19.8**	42.1	55.7	9.5
K-Net	R-101-FPN	✓	✓	36	40.1	62.8	43.1	18.7	42.7	58.8	**16.2**
K-Net-N256	R-101-FPN	✓	✓	36	**40.6**	**63.3**	**43.7**	18.8	**43.3**	**59.0**	15.5

ods [63], and empirically find $\lambda_{seg} = 1$, $\lambda_{ce} = 1$, $\lambda_{dice} = 4$ work best. For efficiency, the default number of instance kernels is 100. For semantic segmentation, N equals to the number of classes of the dataset, which is 150 in ADE20K and 133 in COCO dataset. The number of rounds of iterative kernel update is set to three by default for all segmentation tasks.

4.1 Benchmark Results

Panoptic Segmentation. We first benchmark K-Net with other panoptic segmentation frameworks in Table 1. K-Net surpasses the previous state-of-the-art box-based method [28] and box/NMS-free method [30] by 1.7 and 1.5 PQ on `val` split, respectively. On the `test-dev` split, K-Net with ResNet-101-FPN backbone even obtains better results than that of UPSNet [51], which uses Deformable Convolution Network (DCN) [15]. K-Net equipped with DCN surpasses the previous method [28] by 1.1 PQ. Without bells and whistles, K-Net achieves new state-of-the-art single-model performance with Swin Transformer [34] serving as the backbone. Note that only 100 instance kernels are used here for efficiency. K-Net could obtain a higher performance with more instance kernels (Sec. 4.2), as well as an extended training schedule with aggressive data augmentation used in previous work [4].

Instance Segmentation. We compare K-Net with other instance segmentation frameworks [10, 20, 44] in Table 2. More details are in the supplementary materials. As the only box-free and NMS-free

Table 3: Results of K-Net on ADE20K semantic segmentation dataset

(a) Improvements of K-Net on different architectures

Method	Backbone	Val mIoU
FCN [35]	R50	36.7
FCN + K-Net	R50	43.3 *(+6.6)*
PSPNet [58]	R50	42.6
PSPNet + K-Net	R50	43.9 *(+1.3)*
DLab.v3 [8]	R50	43.5
DLab.v3 + K-Net	R50	44.6 *(+1.1)*
UperNet [49]	R50	42.4
UperNet + K-Net	R50	43.6 *(+1.2)*
UperNet	Swin-L	50.6
UperNet + K-Net	Swin-L	52 *(+1.4)*

(b) Comparisons with state-of-the-art methods. Results marked by † use larger image sizes

Method	Backbone	Val mIoU
OCRNet [55]	HRNet-W48	44.9
PSPNet [58]	R101	45.4
PSANet [59]	R101	45.4
DNL [54]	R101	45.8
DLab.v3 [8]	R101	46.7
DLab.v3+ [9]	S-101 [57]	47.3
SETR [60]	ViT-L [16]	48.6
UperNet†	Swin-L	53.5
UperNet + K-Net	Swin-L	53.3
UperNet + K-Net†	Swin-L	**54.3**

Table 4: Ablation studies of K-Net on instance segmentation

(a) Adaptive Kernel Update (A. K. U.) and Kernel Interaction (K. I.)

A. K. U.	K. I.	AP	AP_{50}	AP_{70}
		8.7	15.6	8.8
✓		22.6	37.3	23.5
	✓	6.7	12.2	6.6
✓	✓	34.1	55.3	35.7

(b) Positional Encoding (P. E.) and Coordinate Convolution (Coors.)

Coors.	P. E.	AP	AP_{50}	AP_{70}
		30.9	51.7	31.6
✓		34.0	55.4	35.6
	✓	34.1	55.3	35.7
✓	✓	34.0	55.1	35.8

(c) Number of rounds of kernel update

Stage Number	AP	AP_{50}	AP_{70}	FPS
1	21.8	37.3	22.1	24.0
2	32.1	52.3	33.5	22.7
3	34.1	55.3	35.7	21.2
4	34.5	56.5	35.7	20.1
5	34.5	56.5	35.9	18.9

(d) Numbers of instance kernels

N	AP	AP_{50}	AP_{70}	FPS
50	32.7	53.7	34.1	21.6
64	33.6	54.8	35.1	21.6
100	34.1	55.3	35.7	21.2
128	34.3	55.6	35.8	20.7
256	34.7	56.1	36.3	19.8

method, K-Net achieves better performance and faster inference speed than Mask R-CNN [20], SOLO [46], SOLOv2 [47] and CondInst [44], indicated by the higher AP and frames per second (FPS). We adopt 256 instance kernels (K-Net-N256 in the table) to compare with Cascade Mask R-CNN [3]. The performance of K-Net-N256 is on par with Cascade Mask R-CNN [3] but enjoys a **92.2%** faster inference speed (19.8 v.s 10.3). On COCO `test-dev` split, K-Net with ResNet-101-FPN backbone obtains performance that is 0.9 AP better than Mask R-CNN [20]. It also surpasses previous kernel-based approach CondInst [44] and SOLOv2 [47] by 1.2 AP and 0.6 AP, respectively. With ResNet-101-FPN backbone, K-Net surpasses Cascade Mask R-CNN with 100 and 256 instance kernels in both accuracy and speed by 0.2 AP and 6.7 FPS, and 0.7 AP and 6 FPS, respectively.

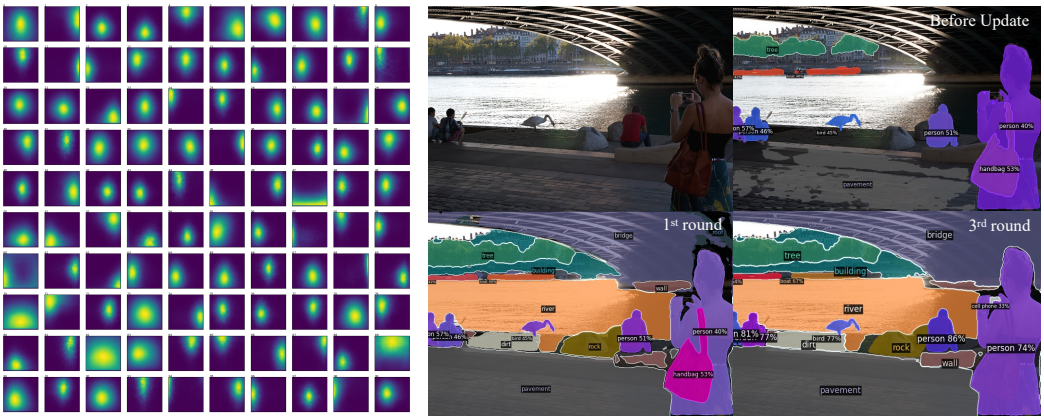
Semantic Segmentation. We apply K-Net to existing frameworks [8, 35, 49, 58] that rely on static semantic kernels in Table 3a. K-Net consistently improves different frameworks by making their semantic kernels dynamic. Notably, K-Net significantly improves FCN (**6.6** mIoU). This combination surpasses PSPNet and UperNet by 0.7 and 0.9 mIoU, respectively, and achieves performance comparable with DeepLab v3. Furthermore, the effectiveness of K-Net does not saturate with strong model representation, as it still brings significant improvement (1.4 mIoU) over UperNet with Swin Transformer [34]. The results suggest the versatility and effectiveness of K-Net for semantic segmentation. In Table 3b, we further compare K-Net with other state-of-the-art methods [9, 60] with test-time augmentation on the validation set. With the input of 512×512, K-Net already achieves state-of-the-art performance. With a larger input of 640×640 following previous method [34] during training and testing, K-Net with UperNet and Swin Transformer achieves new state-of-the-art single model performance, which is 0.8 mIoU higher than the previous one.

4.2 Ablation Study on Instance Segmentation

We conduct an ablation study on COCO instance segmentation dataset to evaluate the effectiveness of K-Net in discriminating instances. The conclusion is also applicable to other segmentation tasks since the design of K-Net is universal to all segmentation tasks.

Head Architecture. We verify the components in the kernel update head. The results in Table 4a indicates that both adaptive kernel update and kernel interaction are necessary for high performance.

Positional Information. We study the necessity of positional information in Table 4b. The results show that positional information is beneficial, and positional encoding [4, 45] works slightly better



(a) Average activation over 5000 images. (b) Mask prediction before and after kernel update.

Figure 3: Visual analysis of kernels and their masks. Better to look with color and zoom in.

than coordinate convolution. The combination of the two components does not bring additional improvements. The results justify the use of just positional encoding in our framework.

Number of Stages. We compare different kernel update rounds in Table 4c. The results show that FPS decreases as the update rounds grow while the performance saturates beyond three stages.

Number of Kernels. We further study the number of kernels in K-Net. The results in Table 4d reveal that 100 kernels are sufficient to achieve good performance. The observation is expected for COCO dataset because most of the images in the dataset do not contain many objects (7.7 objects per image in average [33]). K-Net consistently achieves better performance given more instance kernels since they improve the models' capacity in coping with complicated images.

4.3 Visual Analysis

Overall Distribution of Kernels. We carefully analyze the properties of instance kernels learned in K-Net by analyzing the average of mask activations of the 100 instance kernels over the 5000 images in the `val` split. All the masks are resized to have a similar resolution of 200×200 for the analysis. As shown in Fig. 3a, the learned kernels are meaningful. Different kernels specialize on different regions of the image and objects with different sizes, while each kernel attends to objects of similar sizes at close locations across images.

Masks Refined through Kernel Update. We further analyze how the mask predictions of kernels are refined through the kernel update in Fig. 3b. Here we take K-Net for panoptic segmentation to thoroughly analyze both semantic and instance masks. The masks produced by static kernels are incomplete, *e.g.*, the masks of river and building are missed. After kernel update, the contents are thoroughly covered by the segmentation masks, though the boundaries of masks are still unsatisfactory. The boundaries are refined after more kernel update. The classification confidences of instances also increase after kernel update. More results are in supplementary materials.

5 Conclusion

This paper explores instance kernels that can learn to separate instances during segmentation. Thus, extra components that previously assist instance segmentation can be replaced by instance kernels, including bounding boxes, embedding generation, and hand-crafted post-processing like NMS, kernel fusion, and pixel grouping. Such an attempt, for the first time, allows different image segmentation tasks to be tackled through a unified framework. The framework, dubbed as K-Net, first partitions an image into different groups by learned static kernels, then iteratively refines these kernels and their partition of the image by the features assembled from their partitioned groups. K-Net obtains new state-of-the-art single-model performance on panoptic and semantic segmentation benchmarks and surpasses the well-developed Cascade Mask R-CNN with the fastest inference speed among the recent instance segmentation frameworks. We wish K-Net and the analysis to pave the way for future research on unified image segmentation frameworks.

References

[1] Min Bai and Raquel Urtasun. Deep watershed transform for instance segmentation. In *CVPR*, 2017.

[2] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT: Real-time instance segmentation. In *ICCV*, 2019.

[3] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: Delving into high quality object detection. In *CVPR*, 2018.

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

[5] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. Hybrid task cascade for instance segmentation. In *CVPR*, 2019.

[6] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.

[7] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L. Yuille. DeepLab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *TPAMI*, 2018.

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

[9] 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.

[10] Xinlei Chen, Ross Girshick, Kaiming He, and Piotr Dollár. Tensormask: A foundation for dense object segmentation. 2019.

[11] 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*, 2020.

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

[13] 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.

[14] Jifeng Dai, Kaiming He, and Jian Sun. Instance-aware semantic segmentation via multi-task network cascades. *CVPR*, 2016.

[15] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.

[16] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, Jakob Uszkoreit, and Neil Houlsby. An image is worth 16x16 words: Transformers for image recognition at scale. In *ICLR*, 2021.

[17] Hang Gao, Xizhou Zhu, Stephen Lin, and Jifeng Dai. Deformable Kernels: Adapting effective receptive fields for object deformation. In *ICLR*, 2020.

[18] Junjun He, Zhongying Deng, and Yu Qiao. Dynamic multi-scale filters for semantic segmentation. In *ICCV*, 2019.

[19] Kaiming He, Ross Girshick, and Piotr Dollar. Rethinking ImageNet pre-training. In *ICCV*, 2019.

[20] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask R-CNN. In *ICCV*, 2017.

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

[22] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask Scoring R-CNN. In *CVPR*, 2019.

[23] Max Jaderberg, Karen Simonyan, Andrew Zisserman, and Koray Kavukcuoglu. Spatial transformer networks. In *NeurIPS*, 2015.

[24] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *NeurIPS*, 2016.

[25] Alexander Kirillov, Ross B. Girshick, Kaiming He, and Piotr Dollár. Panoptic feature pyramid networks. In *CVPR*, 2019.

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

[27] Alexander Kirillov, Evgeny Levinkov, Bjoern Andres, Bogdan Savchynskyy, and Carsten Rother. Instancecut: From edges to instances with multicut. In *CVPR*, 2017.

[28] Qizhu Li, Xiaojuan Qi, and Philip H. S. Torr. Unifying training and inference for panoptic segmentation. In *CVPR*, 2020.

[29] Yi Li, Haozhi Qi, Jifeng Dai, Xiangyang Ji, and Yichen Wei. Fully convolutional instance-aware semantic segmentation. *CVPR*, 2017.

[30] Yanwei Li, Hengshuang Zhao, Xiaojuan Qi, Liwei Wang, Zeming Li, Jian Sun, and Jiaya Jia. Fully convolutional networks for panoptic segmentation. In *CVPR*, 2021.

[31] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.

[32] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.

[33] 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.

[34] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin Transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.

[35] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.

[36] Ilya Loshchilov and Frank Hutter. Decoupled weight decay regularization. In *ICLR*, 2019.

[37] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-Net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.

[38] Davy Neven, Bert De Brabandere, Marc Proesmans, and Luc Van Gool. Instance segmentation by jointly optimizing spatial embeddings and clustering bandwidth. In *CVPR*, 2019.

[39] Alejandro Newell, Zhiao Huang, and Jia Deng. Associative Embedding: End-to-end learning for joint detection and grouping. In *NeurIPS*, 2017.

[40] Sida Peng, Wen Jiang, Huaijin Pi, Xiuli Li, Hujun Bao, and Xiaowei Zhou. Deep snake for real-time instance segmentation. In *CVPR*, 2020.

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. In *NeurIPS*, 2015.

[42] Russell Stewart, Mykhaylo Andriluka, and Andrew Y. Ng. End-to-end people detection in crowded scenes. In *CVPR*, 2016.

[43] Richard Szeliski. *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[44] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020.

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

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

[47] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. SOLOv2: Dynamic and fast instance segmentation. *NeurIPS*, 2020.

[48] Jialin Wu, Dai Li, Yu Yang, Chandrajit Bajaj, and Xiangyang Ji. Dynamic filtering with large sampling field for convnets. In *ECCV*, 2018.

[49] Tete Xiao, Yingcheng Liu, Bolei Zhou, Yuning Jiang, and Jian Sun. Unified perceptual parsing for scene understanding. In *ECCV*, 2018.

[50] Enze Xie, Peize Sun, Xiaoge Song, Wenhai Wang, Xuebo Liu, Ding Liang, Chunhua Shen, and Ping Luo. PolarMask: Single shot instance segmentation with polar representation. In *CVPR*, 2020.

[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] Wenqiang Xu, Haiyang Wang, Fubo Qi, and Cewu Lu. Explicit shape encoding for real-time instance segmentation. In *ICCV*, 2019.

[53] Tien-Ju Yang, Maxwell D. Collins, Yukun Zhu, Jyh-Jing Hwang, Ting Liu, Xiao Zhang, Vivienne Sze, George Papandreou, and Liang-Chieh Chen. DeeperLab: Single-shot image parser. *CoRR*, abs/1902.05093, 2019.

[54] Minghao Yin, Zhuliang Yao, Yue Cao, Xiu Li, Zheng Zhang, Stephen Lin, and Han Hu. Disentangled non-local neural networks. In *ECCV*, 2020.

[55] Yuhui Yuan, Xilin Chen, and Jingdong Wang. Object-contextual representations for semantic segmentation. In *ECCV*, 2020.

[56] Hang Zhang, Kristin Dana, Jianping Shi, Zhongyue Zhang, Xiaogang Wang, Ambrish Tyagi, and Amit Agrawal. Context encoding for semantic segmentation. In *CVPR*, June 2018.

[57] Hang Zhang, Chongruo Wu, Zhongyue Zhang, Yi Zhu, Zhi Zhang, Haibin Lin, Yue Sun, Tong He, Jonas Muller, R. Manmatha, Mu Li, and Alexander Smola. Resnest: Split-attention networks. *arXiv preprint arXiv:2004.08955*, 2020.

[58] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017.

[59] Hengshuang Zhao, Yi Zhang, Shu Liu, Jianping Shi, Chen Change Loy, Dahua Lin, and Jiaya Jia. PSANet: Point-wise spatial attention network for scene parsing. In *ECCV*, 2018.

[60] Sixiao Zheng, Jiachen Lu, Hengshuang Zhao, Xiatian Zhu, Zekun Luo, Yabiao Wang, Yanwei Fu, Jianfeng Feng, Tao Xiang, Philip H.S. Torr, and Li Zhang. Rethinking semantic segmentation from a sequence-to-sequence perspective with transformers. In *CVPR*, 2021.

[61] Bolei Zhou, Hang Zhao, Xavier Puig, Tete Xiao, Sanja Fidler, Adela Barriuso, and Antonio Torralba. Semantic understanding of scenes through the ADE20K dataset. *IJCV*, 2019.

[62] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable ConvNets V2: More deformable, better results. In *CVPR*, 2019.

[63] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. *CoRR*, abs/2010.04159, 2020.

Checklist

The checklist follows the references. Please read the checklist guidelines carefully for information on how to answer these questions. For each question, change the default **[TODO]** to [Yes] , [No] , or [N/A] . You are strongly encouraged to include a **justification to your answer**, either by referencing the appropriate section of your paper or providing a brief inline description. For example:

- Did you include the license to the code and datasets? [Yes] See Section **??**.
- Did you include the license to the code and datasets? [No] The code and the data are proprietary.
- Did you include the license to the code and datasets? [N/A]

Please do not modify the questions and only use the provided macros for your answers. Note that the Checklist section does not count towards the page limit. In your paper, please delete this instructions block and only keep the Checklist section heading above along with the questions/answers below.

1. For all authors...
   (a) Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope? [Yes]
   (b) Did you describe the limitations of your work? [Yes] We describe the failure cases in the supplementary materials due to limited space.
   (c) Did you discuss any potential negative societal impacts of your work? [Yes] We do that in the supplementary materials due to limited space
   (d) Have you read the ethics review guidelines and ensured that your paper conforms to them? [Yes]
2. If you are including theoretical results...
   (a) Did you state the full set of assumptions of all theoretical results? [N/A]
   (b) Did you include complete proofs of all theoretical results? [N/A]
3. If you ran experiments...
   (a) Did you include the code, data, and instructions needed to reproduce the main experimental results (either in the supplemental material or as a URL)? [No] The code and models are not included in the submission but will be released.
   (b) Did you specify all the training details (e.g., data splits, hyperparameters, how they were chosen)? [Yes]
   (c) Did you report error bars (e.g., with respect to the random seed after running experiments multiple times)? [N/A] The randomness is well controlled and the results only fluctuate 0.1$\sim$0.2 mAP in multiple runs.
   (d) Did you include the total amount of compute and the type of resources used (e.g., type of GPUs, internal cluster, or cloud provider)? [Yes] We use internal cluster where each experiment is conducted with a 8-GPU machine with V100 or Titan XP GPUs installed.
4. If you are using existing assets (e.g., code, data, models) or curating/releasing new assets...

(a) If your work uses existing assets, did you cite the creators? [Yes] MMDetection [6] and MMSegmentation [12] are both cited in Sec. 4

(b) Did you mention the license of the assets? [Yes] They are released under the Apache 2.0 license

(c) Did you include any new assets either in the supplemental material or as a URL? [No] Code and models will be released

(d) Did you discuss whether and how consent was obtained from people whose data you're using/curating? [N/A] We only use open benchmarks like COCO and ADE20K.

(e) Did you discuss whether the data you are using/curating contains personally identifiable information or offensive content? [N/A] We only use open benchmarks like COCO and ADE20K.

5. If you used crowdsourcing or conducted research with human subjects...

(a) Did you include the full text of instructions given to participants and screenshots, if applicable? [N/A]

(b) Did you describe any potential participant risks, with links to Institutional Review Board (IRB) approvals, if applicable? [N/A]

(c) Did you include the estimated hourly wage paid to participants and the total amount spent on participant compensation? [N/A]