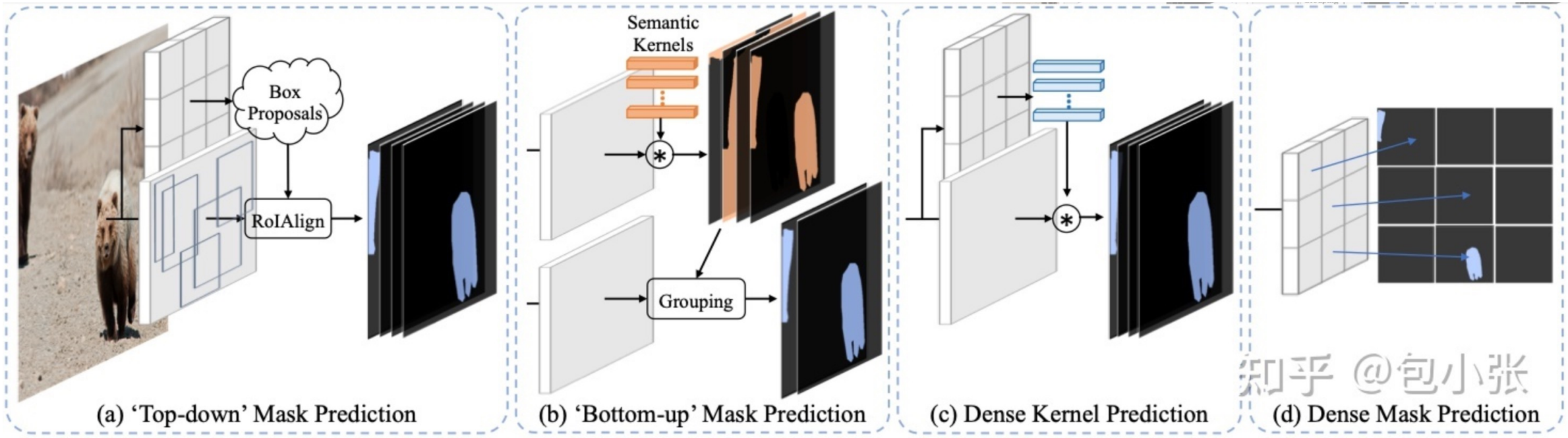
K-Net: Towards Unified Image Segmentation

Wenwei Zhang[1] **Jiangmiao Pang**[2,4] **Kai Chen**[3,4] **Chen Change Loy**[1]✉
[1]S-Lab, Nanyang Technological University
[2]CUHK-SenseTime Joint Lab, the Chinese University of Hong Kong
[3]SenseTime Research [4]Shanghai AI Laboratory
{wenwei001, ccloy}@ntu.edu.sg pangjiangmiao@gmail.com
chenkai@sensetime.com

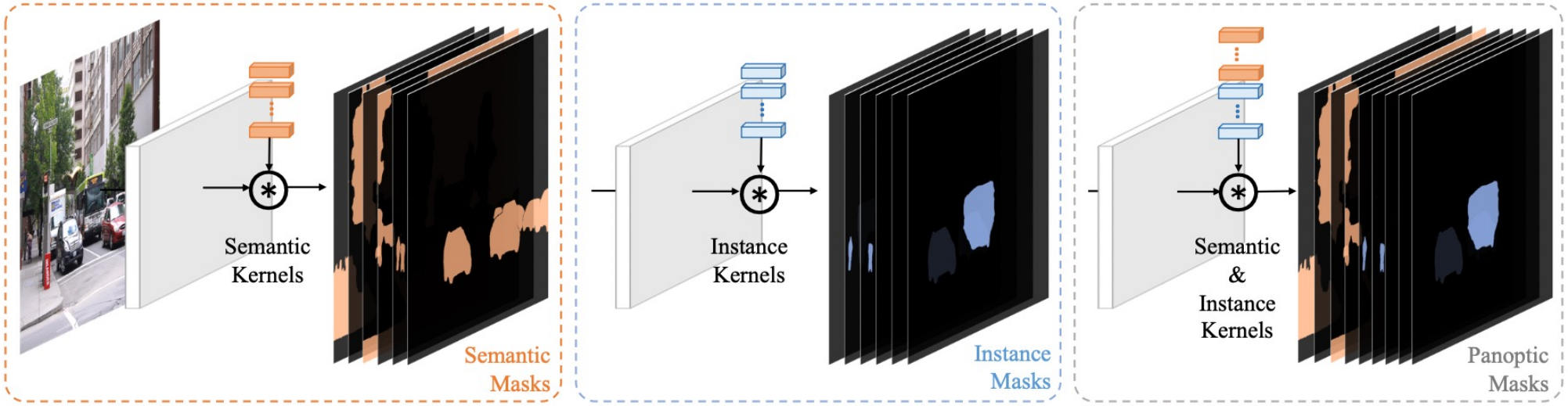
- Instance Segmentation

- (a) Top-down: first detect accurate bounding boxes and generate a mask for each box
- (b) Bottom-up: first perform semantic segmentation then group pixels into different instances
- (c) Dense kernel prediciton: CondInst type, feature grids generate different kernet -> mask -> NMS
- (d) Dense mask prediction: SOLO type, feature grids generate instance mask + NMS



• Overview

- Unify instance/semantic/panoptic segmentation by bipartite matching



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

Figure 1: Semantic segmentation (a), instance (b), and panoptic segmentation (c) tasks are unified by a common framework in this paper. In conventional semantic segmentation methods, each convolutional 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.

- Semantic Segmentation

```
self.conv_seg = nn.Conv2d(channels, num_classes, kernel_size=1)
```

```
seg_kernels = self.conv_seg.weight.clone() #[B, N, C, K, K] -> [B, N, C, K*K]->[B, N, K*K*C]
```

```
self.init_kernels = nn.Conv2d(
    self.out_channels,
    self.num_proposals,
    self.conv_kernel_size,
    padding=int(self.conv_kernel_size // 2),
    bias=False)
```

torch.nn.functional.conv2d(*input*, *weight*, *bias=None*, *stride=1*, *padding=0*, *dilation=1*, *groups=1*) → Tensor

• Method

- K-Net: N-object query
  - Semantic: classes
  - Instance: objects
  - Panoptic: stuff classes + objects

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

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

• Method

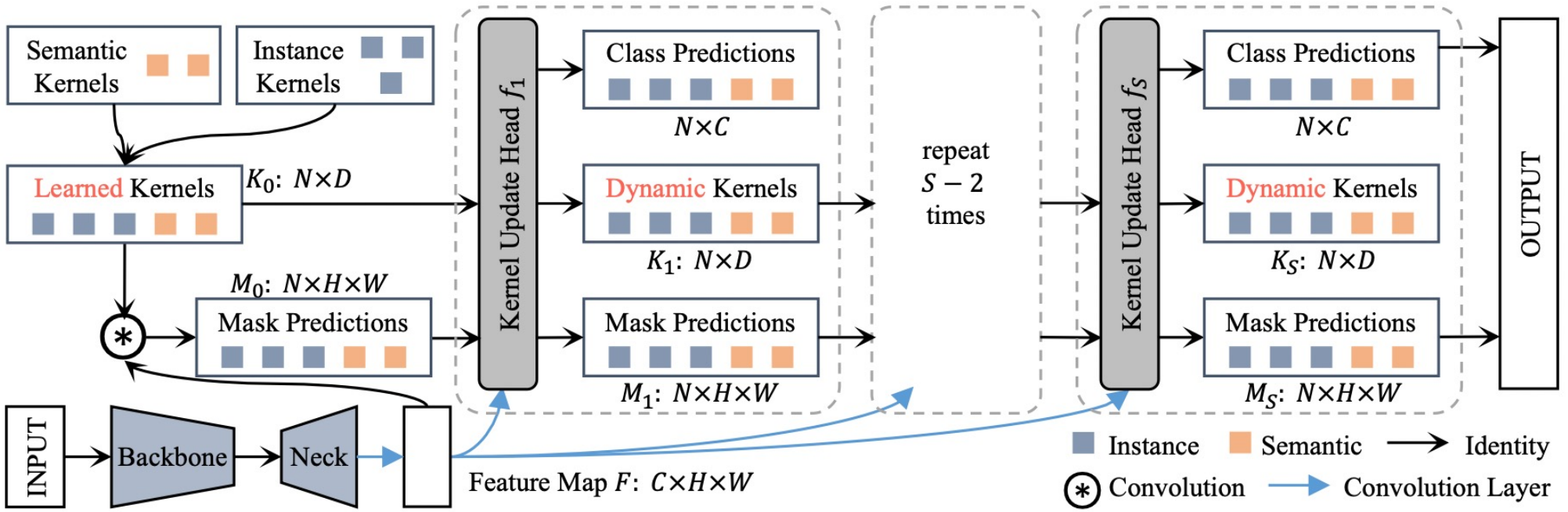


Figure 3: **K-Net for panoptic segmentaion.** A set of learned kernels first performs convolution with the feature map F to predict masks M_0. Then the kernel update head takes the mask predictions M_0, learned kernels K_0, and feature map F as input and produce class predictions, group-aware (dynamic) kernels, and mask predictions. The produced mask prediction, dynamic kernels, and feature map F are sent to the next kernel update head. This process is performed iteratively to progressively refine the kernels and the mask predictions.

• Method

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

$$F^K = \sum_u^H \sum_v^W M_{i-1}(u,v) \cdot F(u,v), F^K \in R^{B \times N \times C}, \tag{3}$$

Adaptive Feature Update.

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

$$G^K = \sigma(\psi_1(F^G)), G^F = \sigma(\psi_2(F^G)),$$

$$\tilde{K} = G^F \otimes \psi_3(F^K) + G^K \otimes \psi_4(K_{i-1}),$$

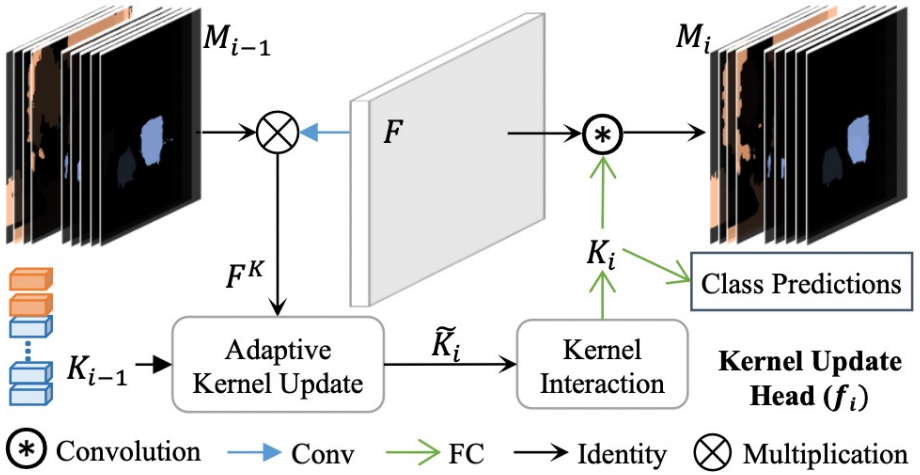


Figure 2: Kernel Update Head.

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

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 [52] followed by a Feed-Forward Neural Network, which has been proven effective in previous works [4, 52]. The output K_i of kernel interaction is then used to generate a new mask prediction through $M_i = g_i\left(K_i\right) * F$, where g_i is an FC-LN-ReLU layer followed by an FC layer. K_i will also be used to predict classification scores in instance and panoptic segmentation.

Loss Functions. The loss function for instance kernels is written as $L_K = \lambda_{cls} L_{cls} + \lambda_{ce} L_{ce} + \lambda_{dice} L_{dice}$, where L_{cls} is Focal loss [37] for classification, and L_{ce} and L_{dice} are CrossEntropy (CE) loss and Dice loss [42] for segmentation, respectively. Given that each instance only occupies a small

• Experiments

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			∼1000	39.7	43.9	33.2
Panoptic FPN [28]	R50-FPN			36	41.5	48.5	31.1
SOLOv2 [56]	R50-FPN	✓		36	42.1	49.6	30.7
DETR [4]†	R50		✓	300 + 25	43.4	48.2	36.3
Unifying [31]	R50-FPN			∼27	43.4	48.6	35.5
Panoptic FCN [34]	R50-FPN	✓	✓	36	43.6	49.3	35.0
K-Net	R50-FPN	✓	✓	36	**47.1**	**51.7**	**40.3**
K-Net	R101-FPN	✓	✓	36	49.6	55.1	41.4
	R101-FPN-DCN	✓	✓	36	48.3	54.0	39.7
	Swin-L [39]	✓	✓	36	54.6	60.2	46.0
			test-dev				
Panoptic-DeepLab	Xception-71	✓		∼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		✓	300 + 25	46.0	-	-
UPSNet [61]	R101-FPN-DCN			36	46.6	53.2	36.7
Unifying [31]	R101-FPN-DCN			∼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**
MaX-DeepLab-L [53]	Max-L	✓	✓	54	51.3	57.2	42.4
MaskFormer [12]	Swin-L [39]	✓	✓	300	53.3	59.1	44.5
Panoptic SegfFormer [35]	PVTv2-B5 [54]	✓	✓	50	54.4	61.1	44.3
K-Net	Swin-L	✓	✓	36	**55.2**	**61.2**	**46.2**

Table 2: Comparisons with state-of-the-art instance segmentation methods on COCO dataset. 'P. (M)' indicates the number of parameters in the model, and the counting unit is million

Method	Backbone	Box-free	NMS-free	Epochs	AP↑	AP_{50}	AP_{70}	AP_s	AP_m	AP_l	FPS↑	P. (M) ↓
					val2017							
SOLO [55]	R-50-FPN	✓		36	35.8	56.7	37.9	14.3	39.3	53.2	12.7	36.08
Mask R-CNN [22]	R-50-FPN			36	37.1	58.5	39.7	18.7	39.6	53.9	17.5	44.17
SOLOv2 [56]	R-50-FPN	✓		36	37.5	58.2	40.0	15.8	41.4	56.6	17.7	**33.89**
CondInst [50]	R-50-FPN			36	37.5	58.5	40.1	18.7	41.0	53.3	14.0	46.37
Cascade Mask R-CNN [3]	R-50-FPN			36	38.5	59.7	**41.8**	**19.3**	41.1	55.6	10.3	77.10
K-Net	R-50-FPN	✓	✓	36	37.8	60.3	39.9	16.9	41.2	57.5	**21.2**	37.26
K-Net-N256	R-50-FPN	✓	✓	36	**38.6**	**60.9**	41.0	19.1	**42.0**	**57.7**	19.8	37.30
					test-dev							
SOLO	R-50-FPN	✓		72	36.8	58.6	39.0	15.9	39.5	52.1	12.7	36.08
Mask R-CNN	R-50-FPN			36	37.4	59.5	40.1	18.6	39.8	51.6	17.5	44.17
CondInst	R-50-FPN			36	37.8	59.2	40.4	18.2	40.3	52.7	14.0	46.37
SOLOv2	R-50-FPN	✓		36	38.2	59.3	40.9	16.0	41.2	55.4	17.7	**33.89**
Cascade Mask R-CNN	R-50-FPN			36	38.8	60.4	**42.0**	**19.4**	40.9	53.9	10.3	77.10
K-Net	R-50-FPN	✓	✓	36	38.4	61.2	40.9	17.4	40.7	56.2	**21.2**	37.26
K-Net-N256	R-50-FPN	✓	✓	36	**39.1**	**61.7**	41.8	18.2	**41.4**	**56.6**	19.8	37.30
SOLO	R-101-FPN	✓		72	37.8	59.5	40.4	16.4	40.6	54.2	10.7	55.07
Mask R-CNN	R-101-FPN			36	38.8	60.8	41.8	19.1	41.2	54.3	14.3	63.16
CondInst	R-101-FPN			36	38.9	60.6	41.8	18.8	41.8	54.4	11.0	**52.83**
SOLOv2	R-101-FPN	✓		36	39.5	60.8	42.6	16.7	43.0	57.4	14.3	65.36
Cascade Mask R-CNN	R-101-FPN			36	39.9	61.6	43.3	**19.8**	42.1	55.7	9.5	96.09
K-Net	R-101-FPN	✓	✓	36	40.1	62.8	43.1	18.7	42.7	58.8	**16.2**	56.25
K-Net-N256	R-101-FPN	✓	✓	36	**40.6**	**63.3**	**43.7**	18.8	**43.3**	**59.0**	15.5	56.29

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

(a) Improvements of K-Net on different architectures

Method	Backbone	Val mIoU
FCN [40]	R50	36.7
FCN + K-Net	R50	43.3 *(+6.6)*
PSPNet [69]	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 [59]	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 [66]	HRNet-W48	44.9
PSPNet [69]	R101	45.4
PSANet [70]	R101	45.4
DNL [65]	R101	45.8
DLab.v3 [8]	R101	46.7
DLab.v3+ [9]	S-101 [68]	47.3
SETR [71]	ViT-L [17]	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_{75}
		10.0	18.2	9.6
✓		22.6	37.3	23.5
	✓	31.2	52.0	32.4
✓	✓	34.1	55.3	35.7

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

Coors.	P. E.	AP	AP_{50}	AP_{75}
		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_{75}	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

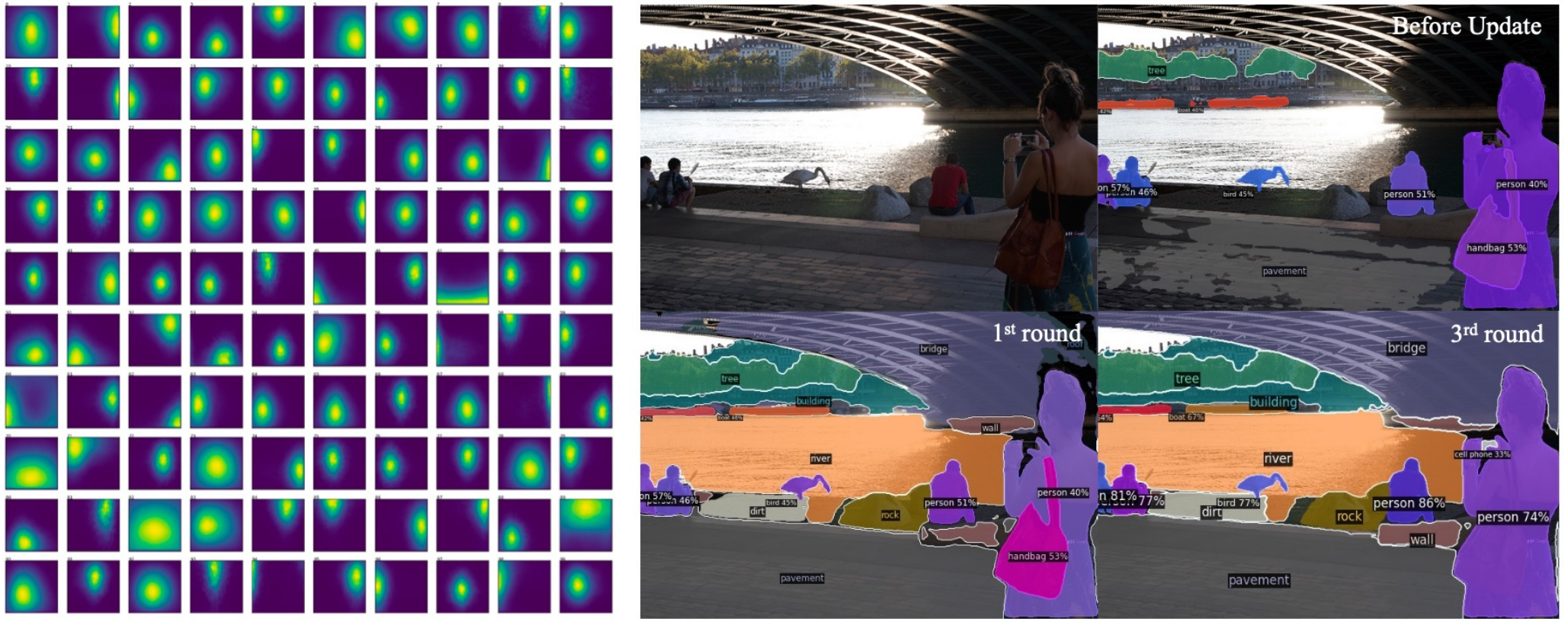
(d) Numbers of instance kernels

N	AP	AP_{50}	AP_{75}	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

Table 5: Numbers of semantic kernels

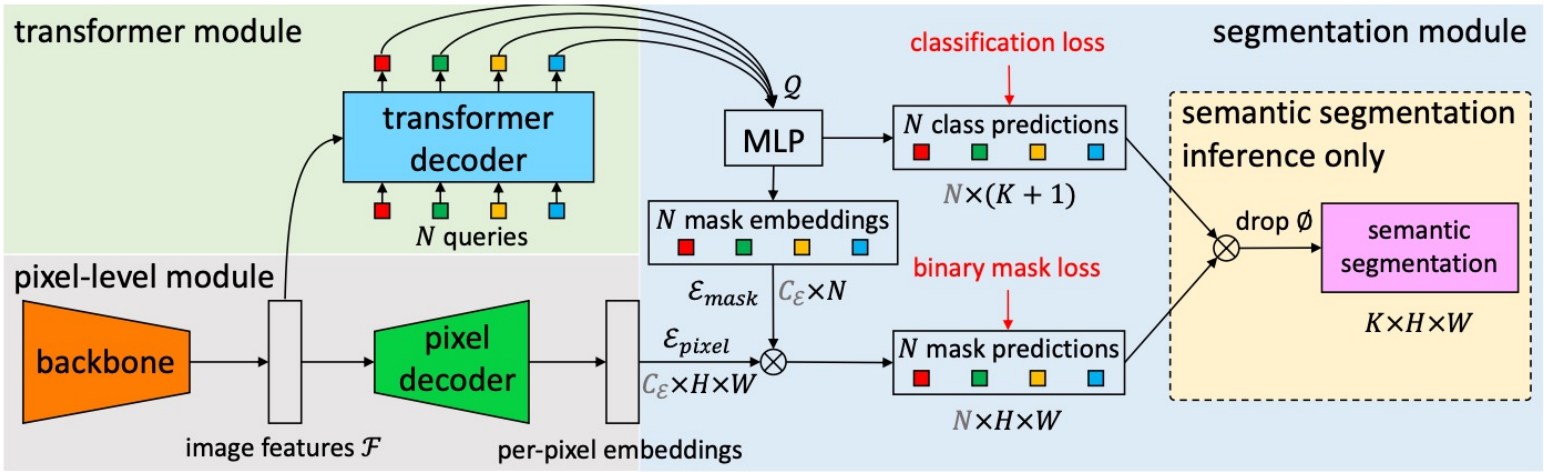
Stage Number	0	1	2	3	4	5	6	7
mIoU	36.7	42.7	43.0	43.3	43.8	44.1	43.1	42.6

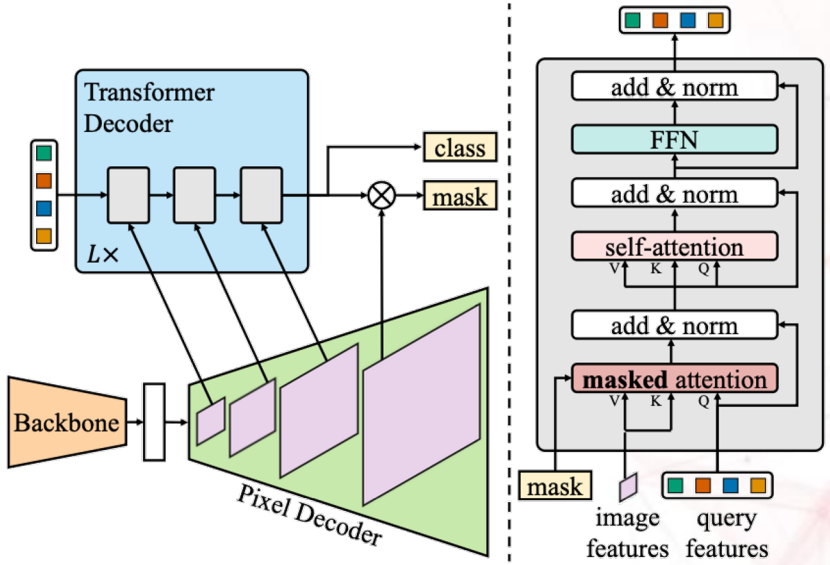
• Visual Analysis



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

Figure 4: Visual analysis of kernels and their masks. Best viewed in color and by zooming in.

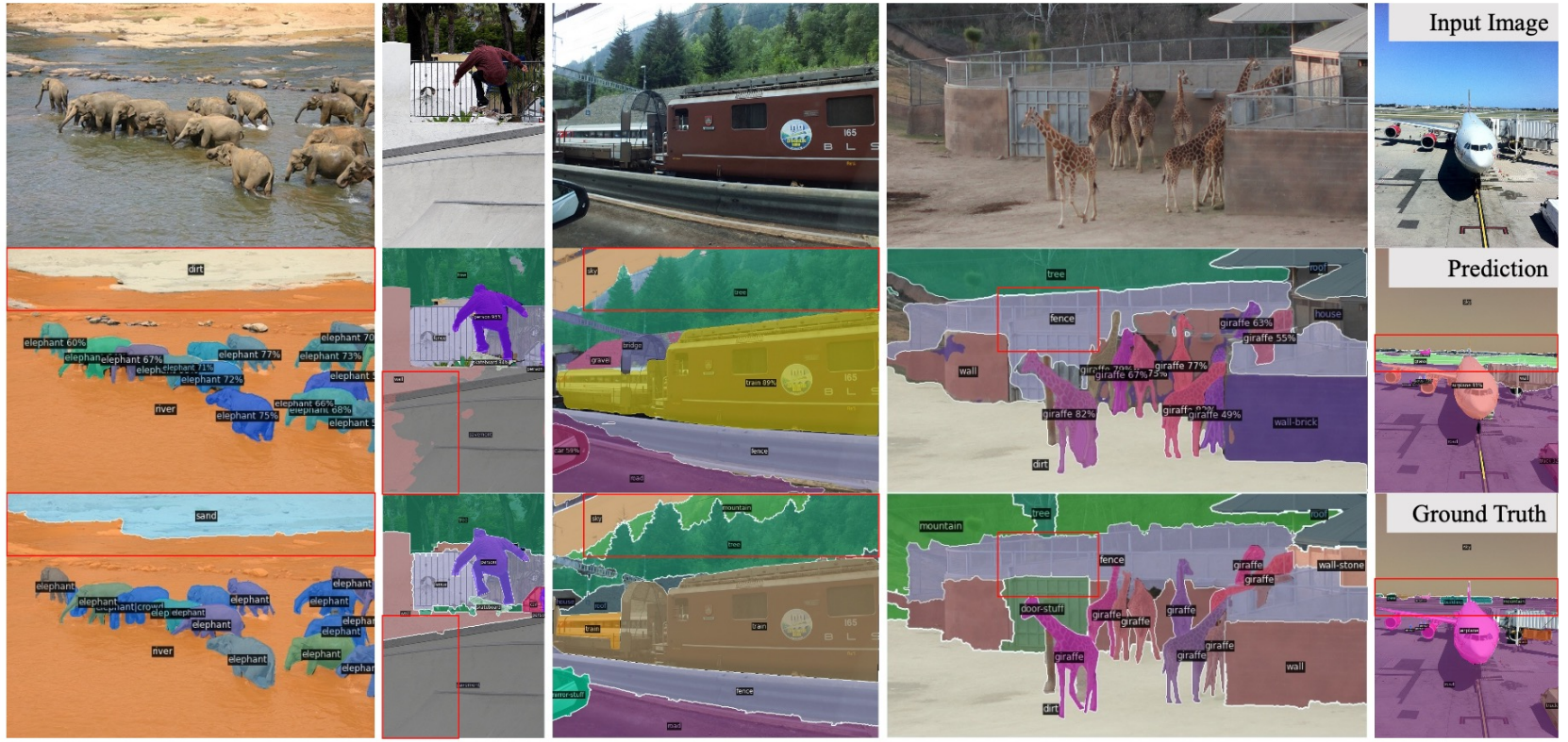




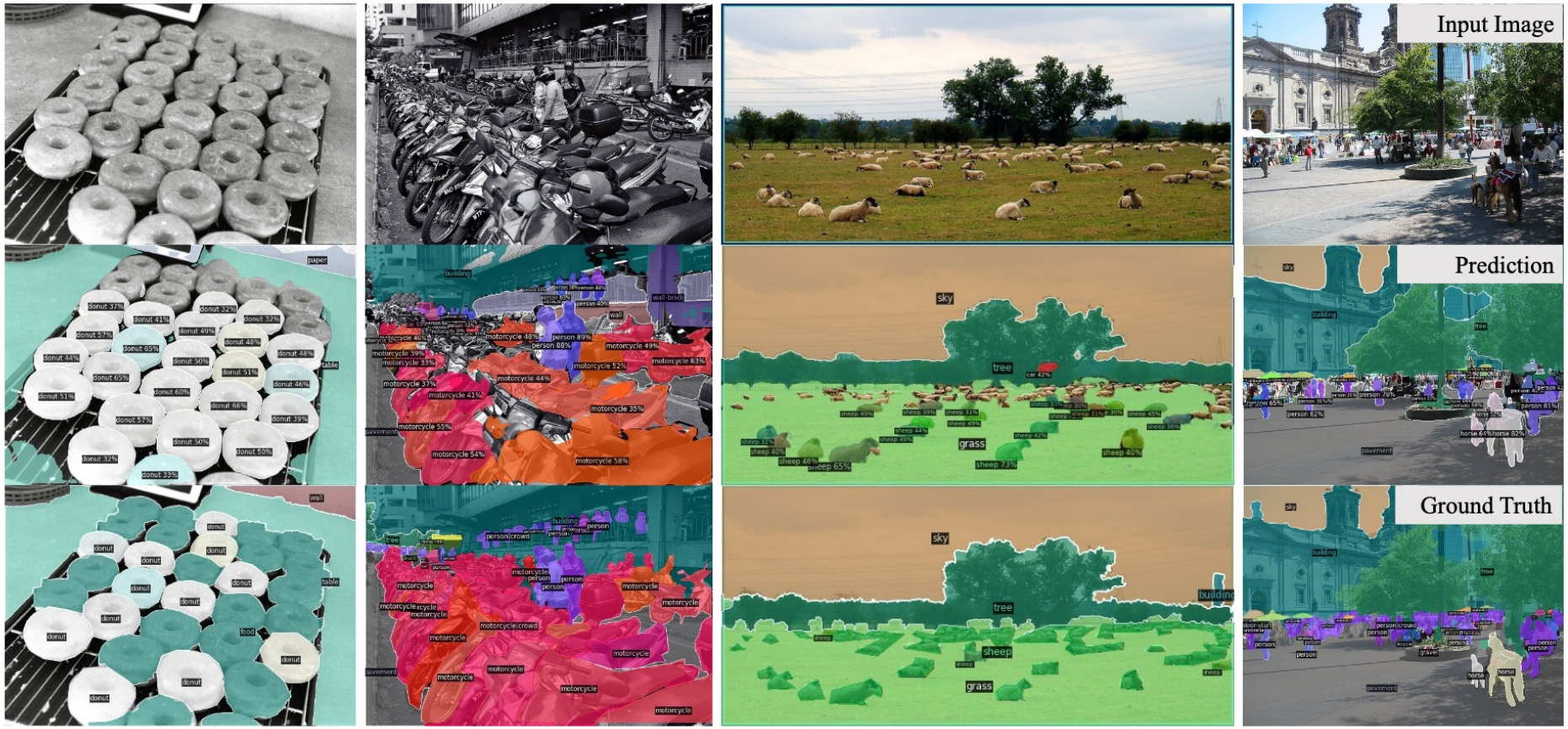
instance-level segmentation 里 mask classification 的核心 design 来解决语义分割任务，最终也统一了不同分割任务的框架。两个方法最终输出 mask prediction 的本质是一样的，都是一组 kernel 分割得到一组 mask，进而对 mask 做分类，只是两个方法生成 kernel 的方式不同，一个是 Transformer 大法好，一个是 iterative refine 的思想。

	method	backbone	search space	epochs	PQ	PQ^{Th}	PQ^{St}	AP^{Th}_{pan}	$mIoU_{pan}$	#params.	FLOPs
CNN backbones	DETR [5]	R50	100 queries	500+25	43.4	48.2	36.3	31.1	-	-	-
		R101	100 queries	500+25	45.1	50.5	37.0	33.0	-	-	-
	K-Net [62]	R50	100 queries	36	47.1	51.7	40.3	-	-	-	-
	Panoptic SegFormer [32]	R50	400 queries	50	50.0	56.1	40.8	-	-	47M	246G
	MaskFormer [14]	R50	100 queries	300	46.5	51.0	39.8	33.0	57.8	45M	181G
		R101	100 queries	300	47.6	52.5	40.3	34.1	59.3	64M	248G
	Mask2Former (ours)	R50	100 queries	50	51.9	57.7	43.0	41.7	61.7	44M	226G
		R101	100 queries	50	**52.6**	**58.5**	**43.7**	**42.6**	**62.4**	63M	293G
Transformer backbones	Max-DeepLab [52]	Max-S	128 queries	216	48.4	53.0	41.5	-	-	62M	324G
		Max-L	128 queries	216	51.1	57.0	42.2	-	-	451M	3692G
	Panoptic SegFormer [32]	PVTv2-B5 [54]	400 queries	50	54.1	60.4	44.6	-	-	101M	391G
	K-Net [62]	Swin-L†	100 queries	36	54.6	60.2	46.0	-	-	-	-
	MaskFormer [14]	Swin-T	100 queries	300	47.7	51.7	41.7	33.6	60.4	42M	179G
		Swin-S	100 queries	300	49.7	54.4	42.6	36.1	61.3	63M	259G
		Swin-B	100 queries	300	51.1	56.3	43.2	37.8	62.6	102M	411G
		Swin-B†	100 queries	300	51.8	56.9	44.1	38.5	63.6	102M	411G
		Swin-L†	100 queries	300	52.7	58.5	44.0	40.1	64.8	212M	792G
	Mask2Former (ours)	Swin-T	100 queries	50	53.2	59.3	44.0	43.3	63.2	47M	232G
		Swin-S	100 queries	50	54.6	60.6	45.7	44.7	64.2	69M	313G
		Swin-B	100 queries	50	55.1	61.0	46.1	45.2	65.1	107M	466G
		Swin-B†	100 queries	50	56.4	62.4	47.3	46.3	67.1	107M	466G
		Swin-L†	*200 queries*	*100*	**57.8**	**64.2**	**48.1**	**48.6**	**67.4**	216M	868G

• Failure Cases



(a) Misclassification and inaccurate boundaries between contents that share similar texture appearance.



(b) Mask prediction on the images that contain crowded instances.