Pyramid Fusion Transformer for Semantic Segmentation

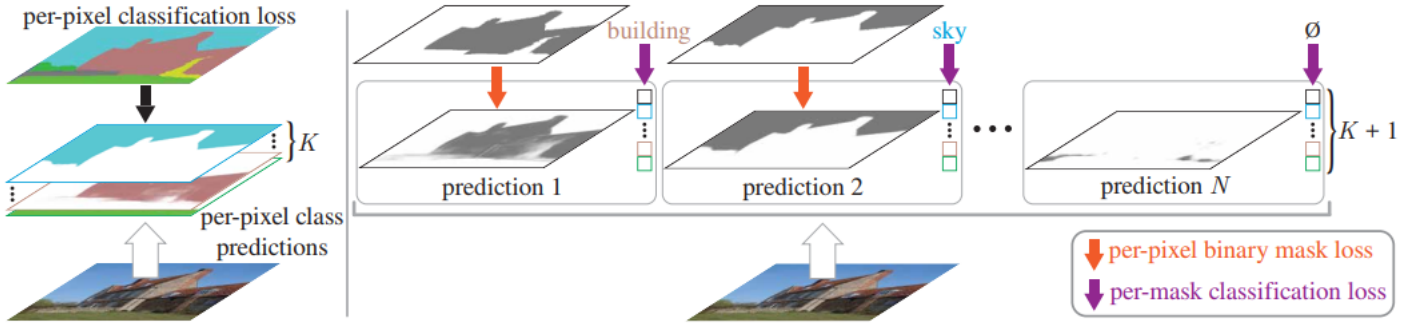
Zipeng Qin[1,2] Jianbo Liu [1] Xiaolin Zhang[2]
Maoqing Tian[2] Aojun Zhou[2,3] Shuai Yi[2] Hongsheng Li[1]

[1]The Chinese University of Hong Kong [2]SenseTime Research [3]Tetras.ai

qinzipeng@sensetime.com hsli@ee.cuhk.edu.hk

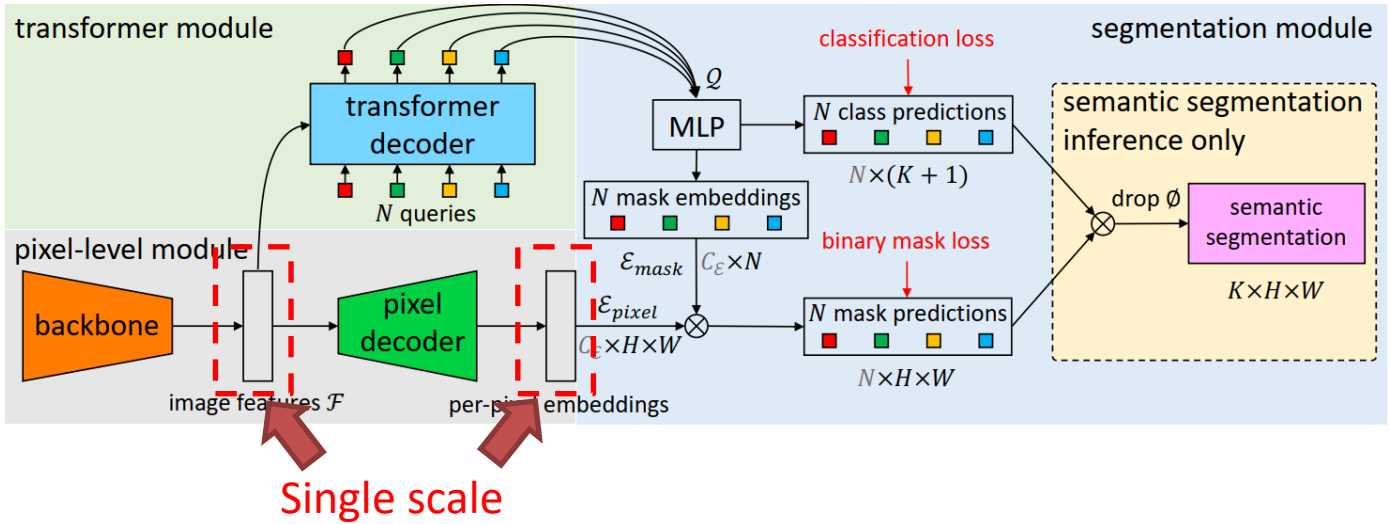
MaskFormer

- MaskFormer employs a Transformer decoder to compute a set of pairs, each consisting of a class prediction and a mask embedding vector.

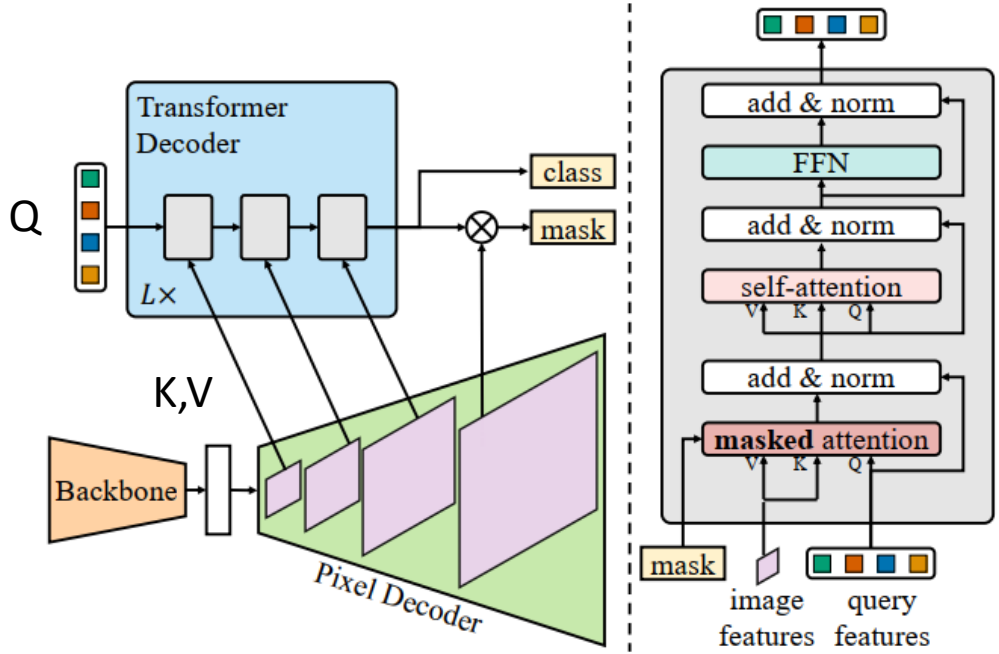


MaskFormer

- The model contains three modules :
- 1**) a pixel-level modul**e extracts per-pixel embeddings used to generate binary mask
- 2) **a transformer module**, computes *N* per-segment embeddings;
- 3) **a segmentation module**



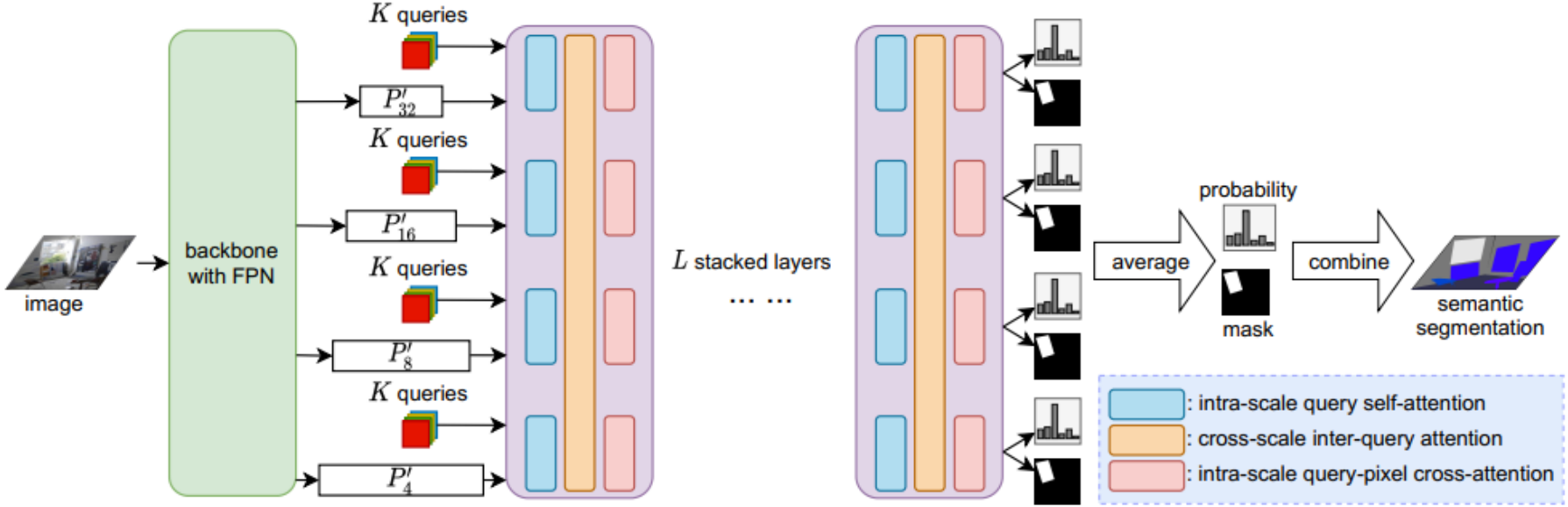
Mask2Former



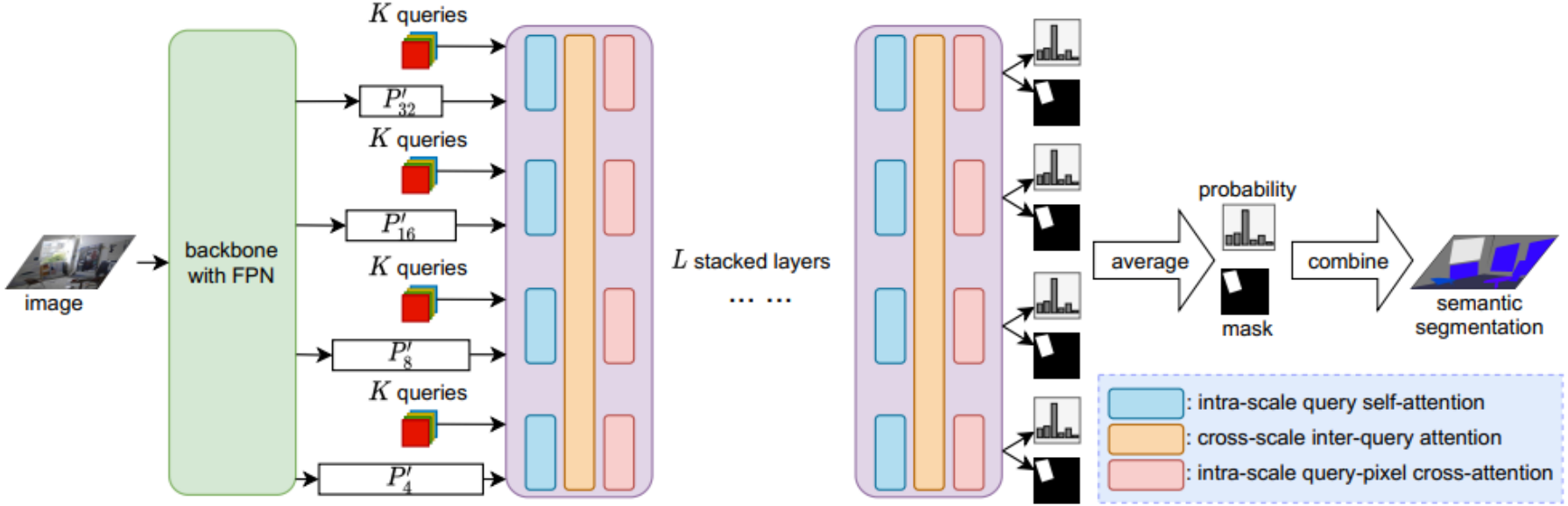
High-resolution features improve model performance, especially for small objects

Transformer-based Pyramid Fusion Transformer (PFT)

- *1. multi-scale transformer decoder*
- *2.* to avoid heavy computation , Neither intra-scale nor cross-scale pixel-to-pixel attention is used in our *PFT*
- *3.* The final prediction is the average of the per-scale predictions in the logit space.



- we recurrently stack three types of attention layers:
  (1) an intra-scale query self-attention layer that conducts conventional self-attention between queries within the same scale,
- (2) a novel *cross-scale inter-query attention* layer to efficiently communicate scale-aware information using the limited number of 4*K* queries of the 4 different scales,
  (3) an intra-scale query-pixel cross-attention layer that aggregates semantic information from flattened sequences of pixel tokens.



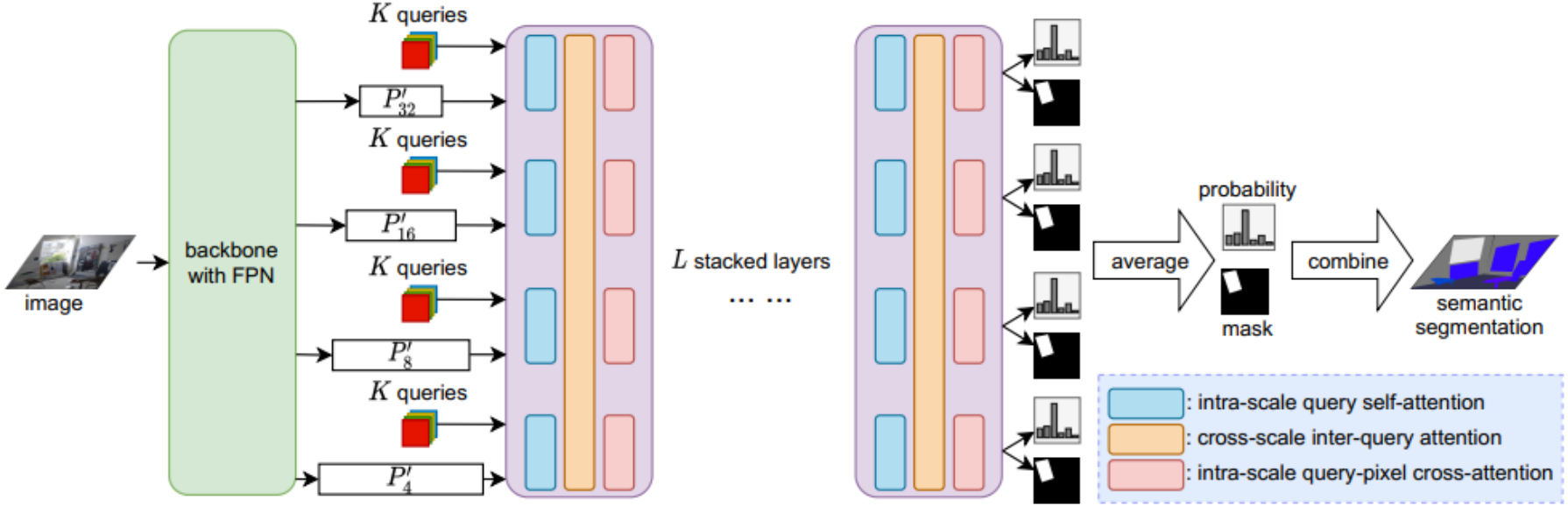
- we recurrently stack three types of attention layers:
  (1) an intra-scale query self-attention layer that conducts conventional self-attention between queries within the same scale,
- (2) a novel *cross-scale inter-query attention* layer to efficiently communicate scale-aware information using the limited number of 4*K* queries of the 4 different scales,
  (3) an intra-scale query-pixel cross-attention layer that aggregates semantic information from flattened sequences of pixel tokens.

Intra-scale query self-attention

- (1) an intra-scale query self-attention layer that conducts conventional self-attention between queries within the same scale,
-

- conducts self-attention only between category queries within the scale .
- The category queries *Qs* are zero-initialized at the first layer and updated by the stacked attention layers, the learnable positional encodings *Ps* are learned and shared at different depths.

$$
\begin{aligned}
&Q_s, K_s = \text{Projection}(\mathcal{Q}_s + \mathcal{P}_s), \\
&V_s = \text{Projection}(\mathcal{Q}_s), \\
&\mathcal{Q}_s = \text{Attention}(Q_s, K_s, V_s), \text{ for } s = 4, 8, 16, 32
\end{aligned} \tag{1}
$$



- we recurrently stack three types of attention layers:
  (1) an intra-scale query self-attention layer that conducts conventional self-attention between queries within the same scale,
- (2) a novel *cross-scale inter-query attention* layer to efficiently communicate scale-aware information using the limited number of 4*K* queries of the 4 different scales,
  (3) an intra-scale query-pixel cross-attention layer that aggregates semantic information from flattened sequences of pixel tokens.

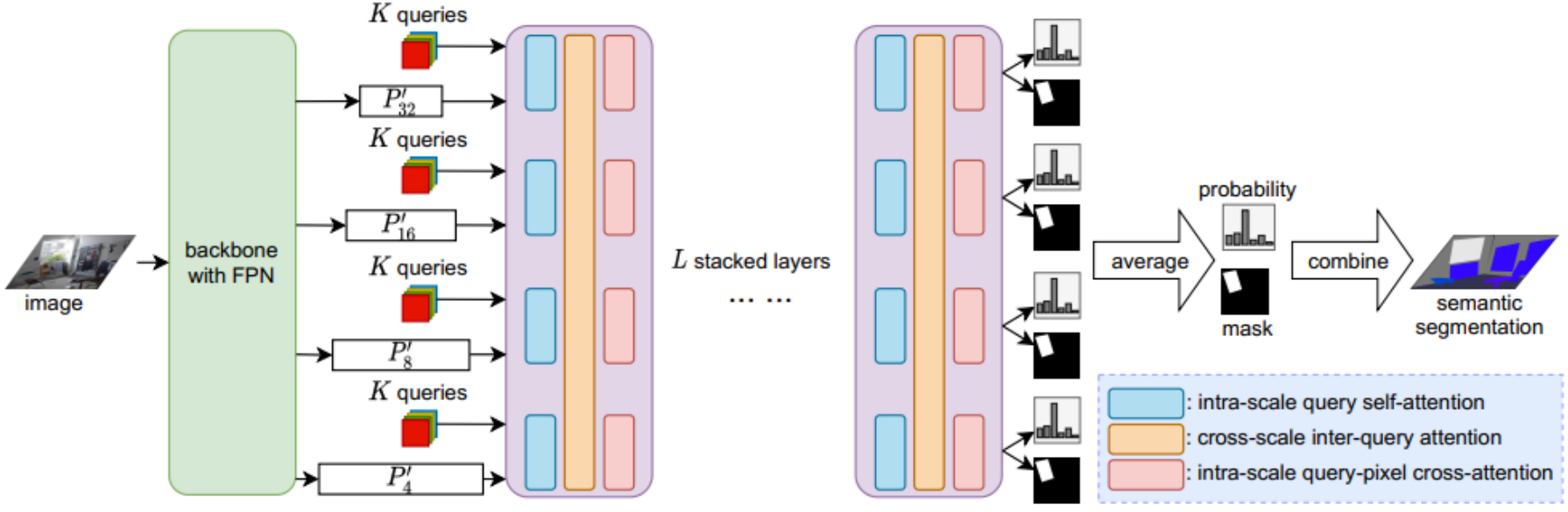
Cross-scale inter-query attention

- the number of such scale-aware queries in each scale is much smaller than the number of all visual tokens in each scale

$$\mathcal{Q}_{\text{all}} = \text{Concat}(\mathcal{Q}_4, \mathcal{Q}_8, \mathcal{Q}_{16}, \mathcal{Q}_{32}) \in \mathbb{R}^{4\mathcal{K}\times\breve{C}}$$

$$\mathcal{P}_{\text{all}} = \text{Concat}(\mathcal{P}_4, \mathcal{P}_8, \mathcal{P}_{16}, \mathcal{P}_{32})$$

$$\begin{aligned} Q, K &= \text{Projection}(\mathcal{Q}_{\text{all}} + \mathcal{P}_{\text{all}}), \\ V &= \text{Projection}(\mathcal{Q}_{\text{all}}), \\ \mathcal{Q}_4, \mathcal{Q}_8, \mathcal{Q}_{16}, \mathcal{Q}_{32} &= \text{Attention}(Q, K, V), \end{aligned} \tag{2}$$



- we recurrently stack three types of attention layers:
  (1) an intra-scale query self-attention layer that conducts conventional self-attention between queries within the same scale,
- (2) a novel *cross-scale inter-query attention* layer to efficiently communicate scale-aware information using the limited number of 4*K* queries of the 4 different scales,

(3) an intra-scale query-pixel cross-attention layer that aggregates semantic information from flattened sequences of pixel tokens.

Intra-scale query-pixel cross-attention.

- Within each scale , we fixed sinusoidal positional encodings *Pssine* to the pixel tokens *Ps*, because of the too large number of pixel tokens.

$$
\begin{aligned}
Q_s &= \text{Projection}(\mathcal{Q}_s + \mathcal{P}_s), \\
K_s &= \text{Projection}(P_s + \mathcal{P}_s^{\text{sine}}), \\
V_s &= \text{Projection}(P_s), \\
\mathcal{Q}_s &= \text{Attention}(Q_s, K_s, V_s), \text{ for } s = 4, 8, 16, 32,
\end{aligned}
\tag{3}
$$

Training losse

- $$L_{\text{cls}} = \lambda_{\text{ce}} L_{\text{ce}} + \lambda_{\text{focal-ce}} L_{\text{focal-ce}},$$

$$L_{\text{mask}} = \lambda_{\text{focal}} L_{\text{focal}} + \lambda_{\text{dice}} L_{\text{dice}}.$$

$$L_{\text{train}} = L_{\text{cls}} + L_{\text{mask}}$$

Generating Segmentation Maps.

- probability logits

Each scale

average

$$\mathcal{L}_s^{\text{prob}} = \text{Linear}(\mathcal{Q}_s), \quad \mathcal{L}^{\text{prob}} = \frac{\sum_{\text{all } s} \mathcal{L}_s^{\text{prob}}}{4}, \qquad (4)$$
$$p = \text{sigmoid}(\mathcal{L}^{\text{prob}})$$

- binary category masks

$$\mathcal{L}_s^{\text{mask}} = \text{MLP}(\mathcal{Q}_s) \otimes \mathcal{M}_s, \quad \mathcal{L}^{\text{mask}} = \frac{\sum_{\text{all } s} \mathcal{L}_s^{\text{mask}}}{4}, \qquad (5)$$
$$m = \text{sigmoid}(\mathcal{L}^{\text{mask}}),$$

- Sementic masks

$$\text{argmax}_{i \in \{1,\ldots,K\}} \; p_i \cdot m_i(h, w)$$

Experiments on the ADE20K dataset

backbone type	method	backbone	pretraining	crop size	batchsize	schedule	mIoU (s.s.)	mIoU (m.s.)	#params.
CNN	OCRNet [42]	R101c	IM-1K	520 × 520	16	150k	-	45.3	-
	GRAr [8]	R101c	IM-1K	544 × 544	16	200k	-	47.1	-
	DeepLabV3+ [5]	R50c	IM-1K	512 × 512	16	160k	44.0	44.9	44M
		R101c	IM-1K	512 × 512	16	160k	45.5	46.4	63M
	MaskFormer [6]	R50	IM-1K	512 × 512	16	160k	44.5	46.7	41M
		R101	IM-1K	512 × 512	16	160k	45.5	47.2	60M
		R101c	IM-1K	512 × 512	16	160k	46.0	48.1	60M
	PFD (ours)	R50	IM-1K	512 × 512	16	160k	45.6 (+1.1)	48.3 (+1.6)	74M
		R101	IM-1K	512 × 512	16	160k	47.2 (+1.7)	**49.4** (+2.2)	93M
		R101c	IM-1K	512 × 512	16	160k	**47.9** (+1.9)	**49.4** (+1.3)	93M
Transformer	BEiT [1]	ViT-L	IM-22K	640 × 640	16	160k	56.7	57.0	441M
	SETR [47]	ViT-L	IM-22K	512 × 512	16	160k	48.6	50.3	308M
	MaskFormer [6]	Swin-T	IM-1K	512 × 512	16	160k	46.7	48.8	42M
		Swin-S	IM-1K	512 × 512	16	160k	49.8	51.0	63M
		Swin-B	IM-22K	640 × 640	16	160k	52.7	53.9	102M
		Swin-L	IM-22K	640 × 640	16	160k	54.1	55.6	212M
	PFD (ours)	Swin-T	IM-1K	512 × 512	16	160k	48.3 (+1.6)	49.6 (+0.8)	74M
		Swin-S	IM-1K	512 × 512	16	160k	51.0 (+1.2)	52.2 (+1.2)	96M
		Swin-B	IM-22K	640 × 640	16	160k	54.1 (+1.4)	55.3 (+1.4)	133M
		Swin-L	IM-22K	640 × 640	16	160k	**56.0** (+1.9)	**57.2** (+1.6)	242M
	Mask2Former (ours)	Swin-L†		640			56.1	57.3	
		Swin-L-FaPN†		640			**56.4**	**57.7**	

Experiments on the PASCAL-Context dataset.

backbone type	method	backbone	pretraining	crop size	batchsize	schedule	mIoU (s.s.)	mIoU (m.s.)	#params.
CNN	SFNet [19]	R50c	IM-1K	512 × 512	16	38k	-	50.7	-
		R101c	IM-1K	512 × 512	16	38k	-	53.8	-
	GRAr [8]	R101c	IM-1K	544 × 544	16	50k	-	55.7	-
	MaskFormer [6]	R50†	IM-1K	480 × 480	16	40k	52.5	54.1	44M
		R50c†	IM-1K	480 × 480	16	40k	52.3	53.9	44M
		R101†	IM-1K	480 × 480	16	40k	53.7	55.4	63M
		R101c†	IM-1K	480 × 480	16	40k	53.1	55.6	63M
	PFD (ours)	R50	IM-1K	480 × 480	16	40k	53.3 (+0.8)	54.8 (+0.7)	60M
		R50c	IM-1K	480 × 480	16	40k	54.2 (+1.9)	55.8 (+1.9)	60M
		R101	IM-1K	480 × 480	16	40k	54.6 (+0.9)	56.2 (+0.8)	79M
		R101c	IM-1K	480 × 480	16	40k	**55.5** (+2.4)	**57.6** (+2.0)	79M

Experiments on the COCO-Stuff-10K dataset

backbone type	method	backbone	pretraining	crop size	batchsize	schedule	mIoU (s.s.)	mIoU (m.s.)	#params.
CNN	OCRNet [42]	R101c	IM-1K	520 × 520	16	60k	-	39.5	-
	GRAr [8]	R101c	IM-1K	544 × 544	16	100k	-	41.9	-
	MaskFormer [6]	R50	IM-1K	544 × 544	16	60k	37.1	38.9	44M
		R50c†	IM-1K	640 × 640	32	60k	37.7	38.1	44M
		R101	IM-1K	640 × 640	32	60k	39.1	39.8	63M
		R101c	IM-1K	640 × 640	32	60k	38.0	39.3	63M
	PFD (ours)	R50	IM-1K	640 × 640	16	60k	38.4 (+1.3)	40.3 (+1.4)	74M
		R50c	IM-1K	640 × 640	16	60k	39.5 (+1.8)	41.0 (+2.9)	74M
		R101	IM-1K	640 × 640	16	60k	40.9 (+1.8)	42.1 (+2.3)	93M
		R101c	IM-1K	640 × 640	16	60k	**41.2** (+3.2)	**42.3** (+3.0)	93M
Transformer	MaskFormer [6]	Swin-T†	IM-1K	640 × 640	16	160k	42.2	42.5	42M
		Swin-S†	IM-1K	640 × 640	16	160k	44.1	45.0	63M
	PFD (ours)	Swin-T	IM-1K	640 × 640	16	160k	42.6 (+0.4)	42.8 (+0.3)	74M
		Swin-S	IM-1K	640 × 640	16	160k	**44.8** (+0.7)	**45.3** (+0.3)	96M

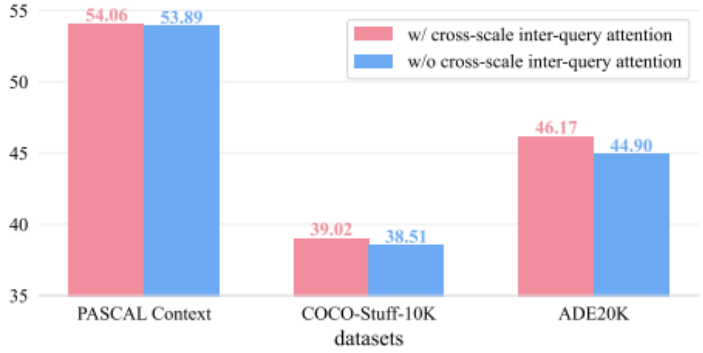


Figure 4. **Performances w/ and w/o cross-scale inter-query attention**. Results obtained with ResNet-50c backbone on three datasets with various sizes and complexities.

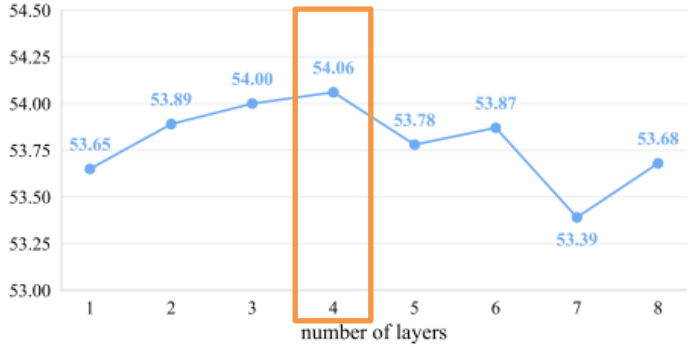


Figure 5. **Ablations on number of transformer layers** within each per-scale transformer decoder.

Ablations on weight sharing for different modules

	transformer decoders	prob. lin. proj.	mask MLPs	query pos. embed.	mIoU
weight sharing	✗	✗	✗	✗	54.06
	✓	✗	✗	✗	53.43
	✗	✓	✗	✗	53.97
	✗	✗	✓	✗	53.64
	✗	✗	✗	✓	53.89

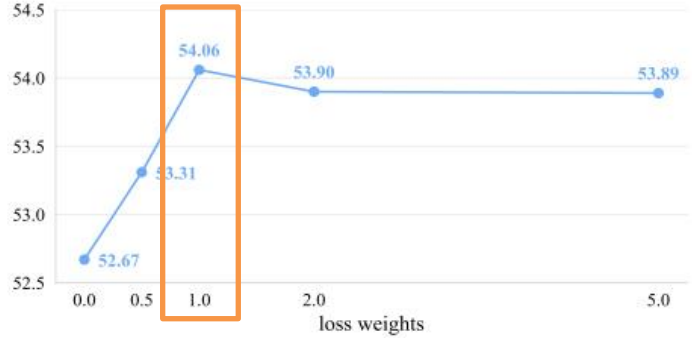


Figure 6. **Loss weights for focal-style cross-entropy loss**. We choose $\lambda_{\text{focal-ce}} \in \{0.0, 0.5, 1.0, 2.0, 5.0\}$ and train our model under the same setting on PASCAL Context dataset. We pick $\lambda_{\text{focal-ce}} \in \{1.0, 2.0\}$ as our candidate parameters given the results.