(arXiv 21.07) Per-Pixel Classification is Not All You Need for Semantic Segmentation

Luoyao Kang
2021.9.29

Motivation

The goal of semantic segmentation is to partition an image into regions with different semantic categories . Starting from Fully Convolutional Networks (FCNs) work of Long et al. , most deep learning-based semantic segmentation approaches formulate semantic segmentation as per-pixel classification

Previous work:

FCNs

DeepLab V3+

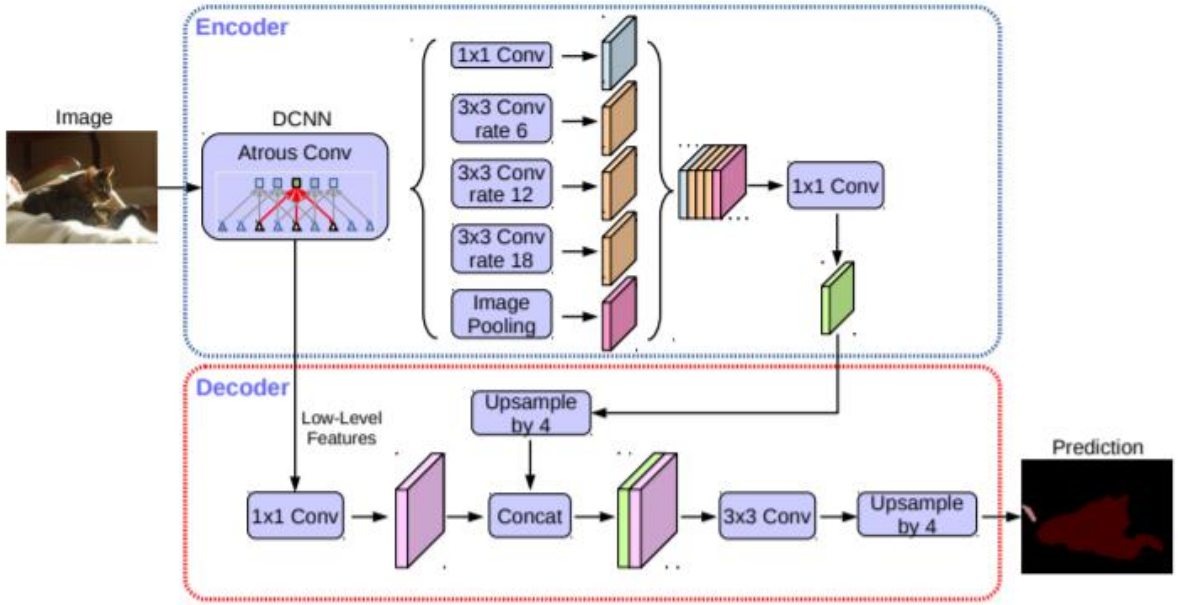


Fig. 2. Our proposed DeepLabv3+ extends DeepLabv3 by employing a encoder-decoder structure. The encoder module encodes multi-scale contextual information by applying atrous convolution at multiple scales, while the simple yet effective decoder module refines the segmentation results along object boundaries.

Motivation

In fact, before FCN, the best performing semantic segmentation methods like O2P and SDS used **a mask classification formulation**.

To address semantic- and instance-level segmentation tasks they propose a simple **MaskFormer** approach that seamlessly converts any existing **per-pixel classification** model into a **mask classification**.

Intuition

Mask Classification:

Semantic Segmentation = Instance Segmentation + Instance Classification

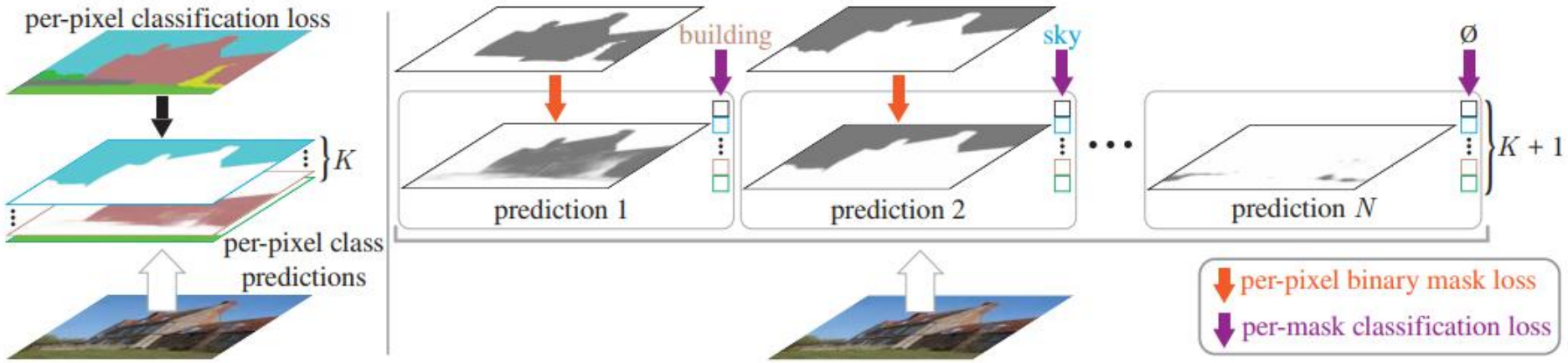


Figure 1: **Per-pixel classification *vs*. mask classification. (left)** Semantic segmentation with per-pixel classification applies the same classification loss to each location. **(right)** Mask classification predicts a set of binary masks and assigns a single class to each mask. Each prediction is supervised with a per-pixel binary mask loss and a classification loss. Matching between the set of predictions and ground truth segments can be done either via *bipartite matching* similarly to DETR [4] or by *fixed matching* via direct indexing if the number of predictions and classes match, *i.e.*, if $N = K$.

Background

Mask Classification:

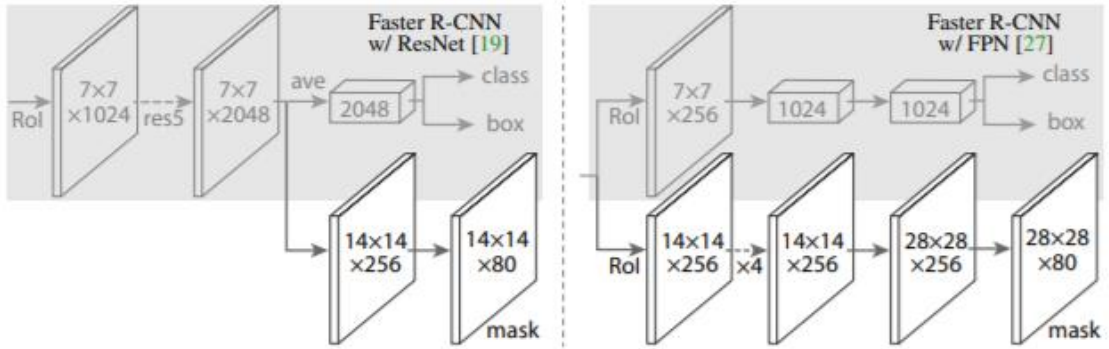


Figure 4. **Head Architecture**: We extend two existing Faster R-CNN heads [19, 27]. Left/Right panels show the heads for the ResNet C4 and FPN backbones, from [19] and [27], respectively, to which a mask branch is added. Numbers denote spatial resolution and channels. Arrows denote either conv, deconv, or *fc* layers as can be inferred from context (conv preserves spatial dimension while deconv increases it). All convs are 3×3, except the output conv which is 1×1, deconvs are 2×2 with stride 2, and we use ReLU [31] in hidden layers. *Left*: 'res5' denotes ResNet's fifth stage, which for simplicity we altered so that the first conv operates on a 7×7 RoI with stride 1 (instead of 14×14 / stride 2 as in [19]). *Right*: '×4' denotes a stack of four consecutive convs.

From Per-Pixel to Mask Classification

Per-pixel classification formulation:

1) predicted probability distribution:

$$y = \{p_i | p_i \in \Delta^K\}_{i=1}^{H \cdot W}$$

Here ΔK is the K dimensional probability simplex

2) ground truth category labels:

$$y^{\text{gt}} = \{y_i^{\text{gt}} | y_i^{\text{gt}} \in \{1, \ldots, K\}\}_{i=1}^{H \cdot W}$$

3) per-pixel cross entropy loss:

$$\mathcal{L}_{\text{pixel-cls}}(y, y^{\text{gt}}) = \sum_{i=1}^{H \cdot W} -\log p_i(y_i^{\text{gt}}).$$

From Per-Pixel to Mask Classification

Mask classification formulation:

1) partitioning/grouping the image into N regions (N does not need to equal K), represented with binary masks:

$$\{m_i | m_i \in [0,1]^{H \times W}\}_{i=1}^{N}$$

2) associating each region as a whole with some distribution over K categories:

$$z = \{(p_i, m_i)\}_{i=1}^{N} \qquad p_i \in \Delta^{K+1}$$

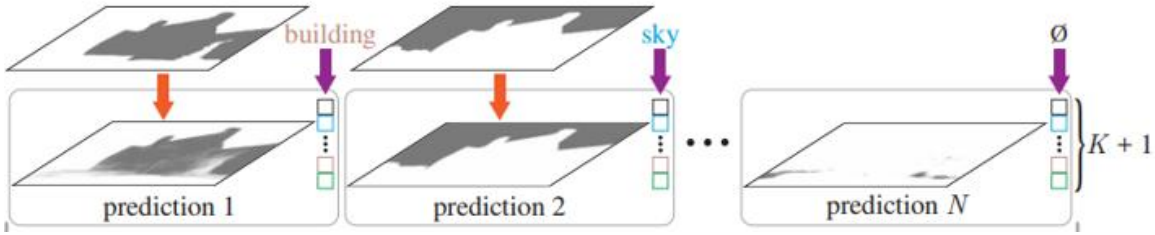
ground truth segments:

$$z^{\text{gt}} = \{(c_i^{\text{gt}}, m_i^{\text{gt}}) | c_i^{\text{gt}} \in \{1, \ldots, K\}, m_i^{\text{gt}} \in \{0,1\}^{H \times W}\}_{i=1}^{N^{\text{gt}}}$$

From Per-Pixel to Mask Classification

Mask classification formulation:

3) matching



1. N == K

2. N != K

bipartite matching-based assignment:

$$-p_i(c_j^{\text{gt}}) + \mathcal{L}_{\text{mask}}(m_i, m_j^{\text{gt}})$$

4) mask classification loss

$$\mathcal{L}_{\text{mask-cls}}(z, z^{\text{gt}}) = \sum_{j=1}^{N} \left[-\log p_{\sigma(j)}(c_j^{\text{gt}}) + \mathbb{1}_{c_j^{\text{gt}} \neq \varnothing} \mathcal{L}_{\text{mask}}(m_{\sigma(j)}, m_j^{\text{gt}}) \right]$$

MaskFormer

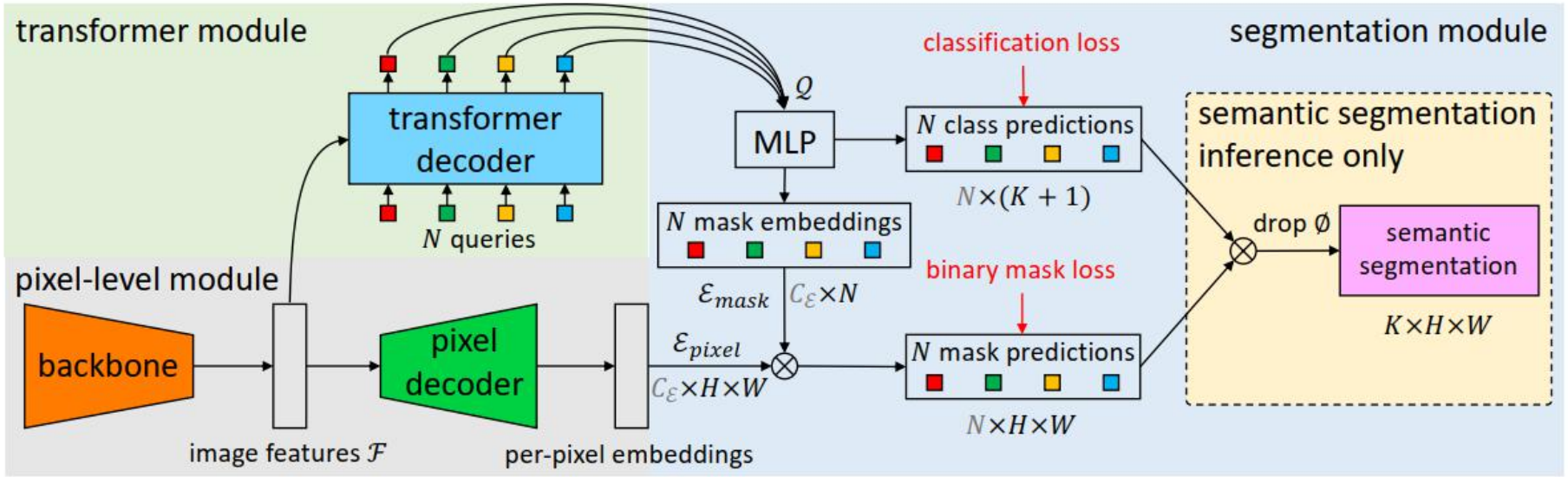
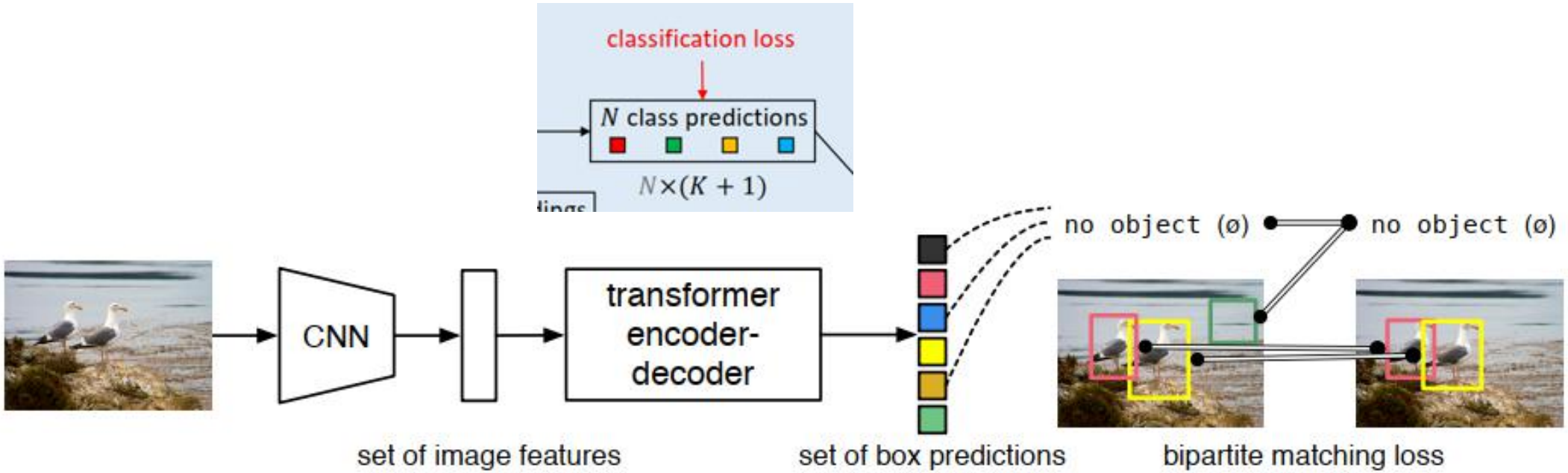


Figure 2: **MaskFormer overview.** We use a backbone to extract image features $\mathcal{F}$. A pixel decoder gradually upsamples image features to extract per-pixel embeddings $\mathcal{E}_{\text{pixel}}$. A transformer decoder attends to image features and produces N per-segment embeddings $\mathcal{Q}$. The embeddings independently generate N class predictions with N corresponding mask embeddings $\mathcal{E}_{\text{mask}}$. Then, the model predicts N possibly overlapping binary mask predictions via a dot product between pixel embeddings $\mathcal{E}_{\text{pixel}}$ and mask embeddings $\mathcal{E}_{\text{mask}}$ followed by a sigmoid activation. For semantic segmentation task we can get the final prediction by combining N binary masks with their class predictions using a simple matrix multiplication (see Section 3.4). Note, the dimensions for multiplication $\otimes$ are shown in gray.

Bipartite matching loss



$$\hat{\sigma} = \arg\min_{\sigma \in \mathfrak{S}_N} \sum_{i}^{N} \mathcal{L}_{\text{match}}(y_i, \hat{y}_{\sigma(i)}),$$

$$\mathcal{L}_{\text{Hungarian}}(y, \hat{y}) = \sum_{i=1}^{N} \left[-\log \hat{p}_{\hat{\sigma}(i)}(c_i) + \mathbb{1}_{\{c_i \neq \varnothing\}} \mathcal{L}_{\text{box}}(b_i, \hat{b}_{\hat{\sigma}}(i)) \right] \qquad \lambda_{\text{iou}} \mathcal{L}_{\text{iou}}(b_i, \hat{b}_{\sigma(i)}) + \lambda_{\text{L1}} \| b_i - \hat{b}_{\sigma(i)} \|_1$$

Transformer Module

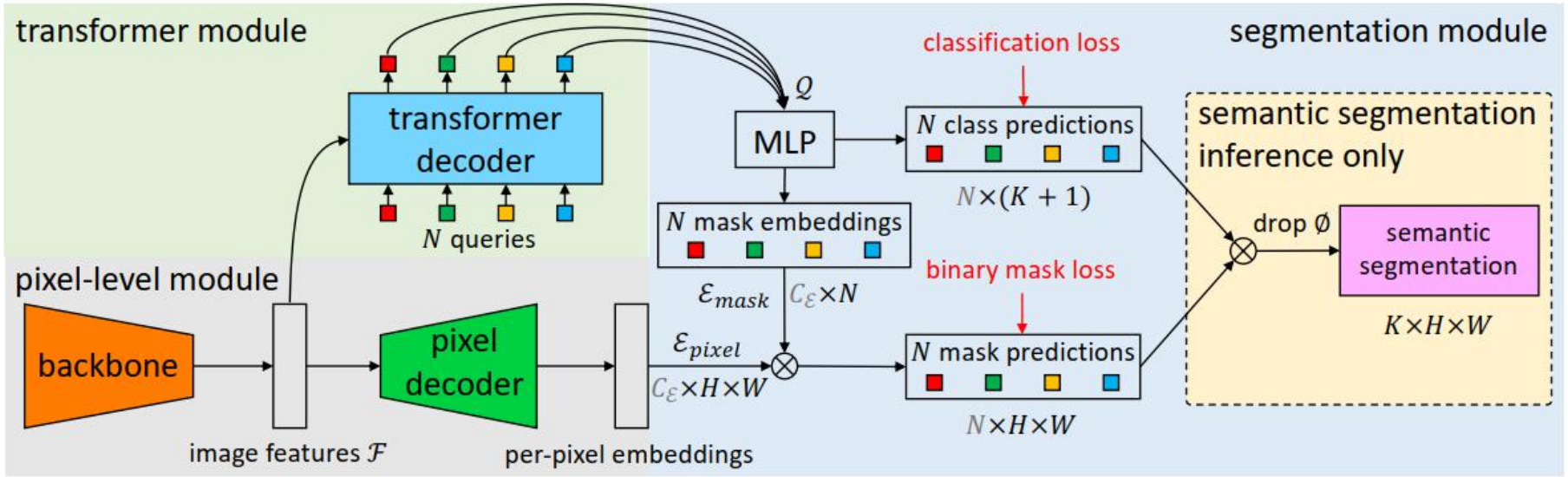


Figure 2: **MaskFormer overview.** We use a backbone to extract image features $\mathcal{F}$. A pixel decoder gradually upsamples image features to extract per-pixel embeddings $\mathcal{E}_{\text{pixel}}$. A transformer decoder attends to image features and produces N per-segment embeddings $\mathcal{Q}$. The embeddings independently generate N class predictions with N corresponding mask embeddings $\mathcal{E}_{\text{mask}}$. Then, the model predicts N possibly overlapping binary mask predictions via a dot product between pixel embeddings $\mathcal{E}_{\text{pixel}}$ and mask embeddings $\mathcal{E}_{\text{mask}}$ followed by a sigmoid activation. For semantic segmentation task we can get the final prediction by combining N binary masks with their class predictions using a simple matrix multiplication (see Section 3.4). Note, the dimensions for multiplication $\otimes$ are shown in gray.

Transformer Decoder

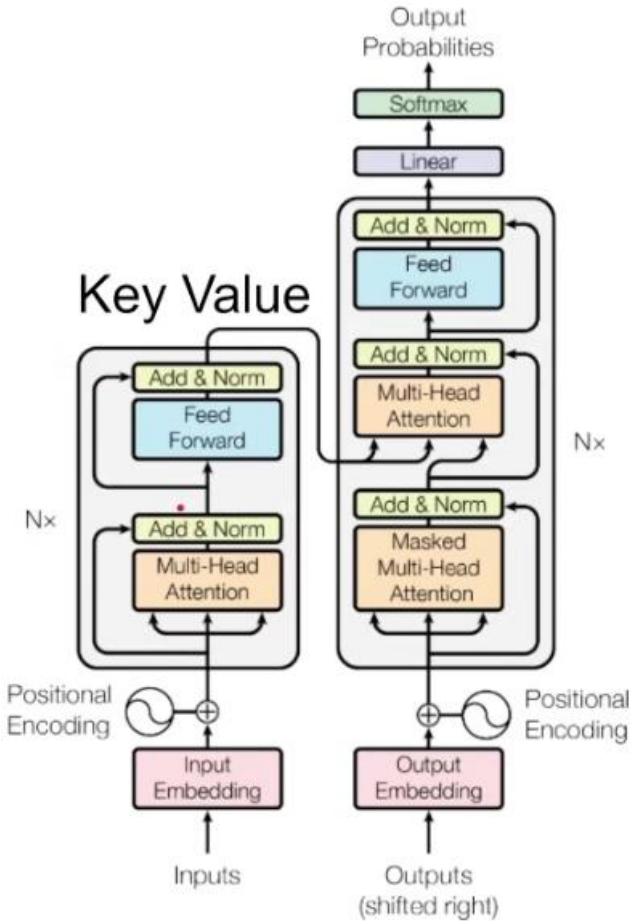
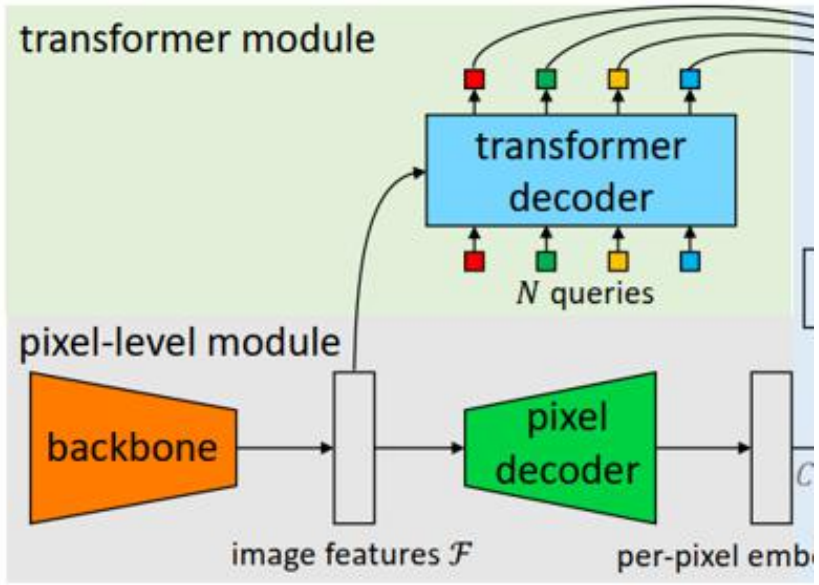
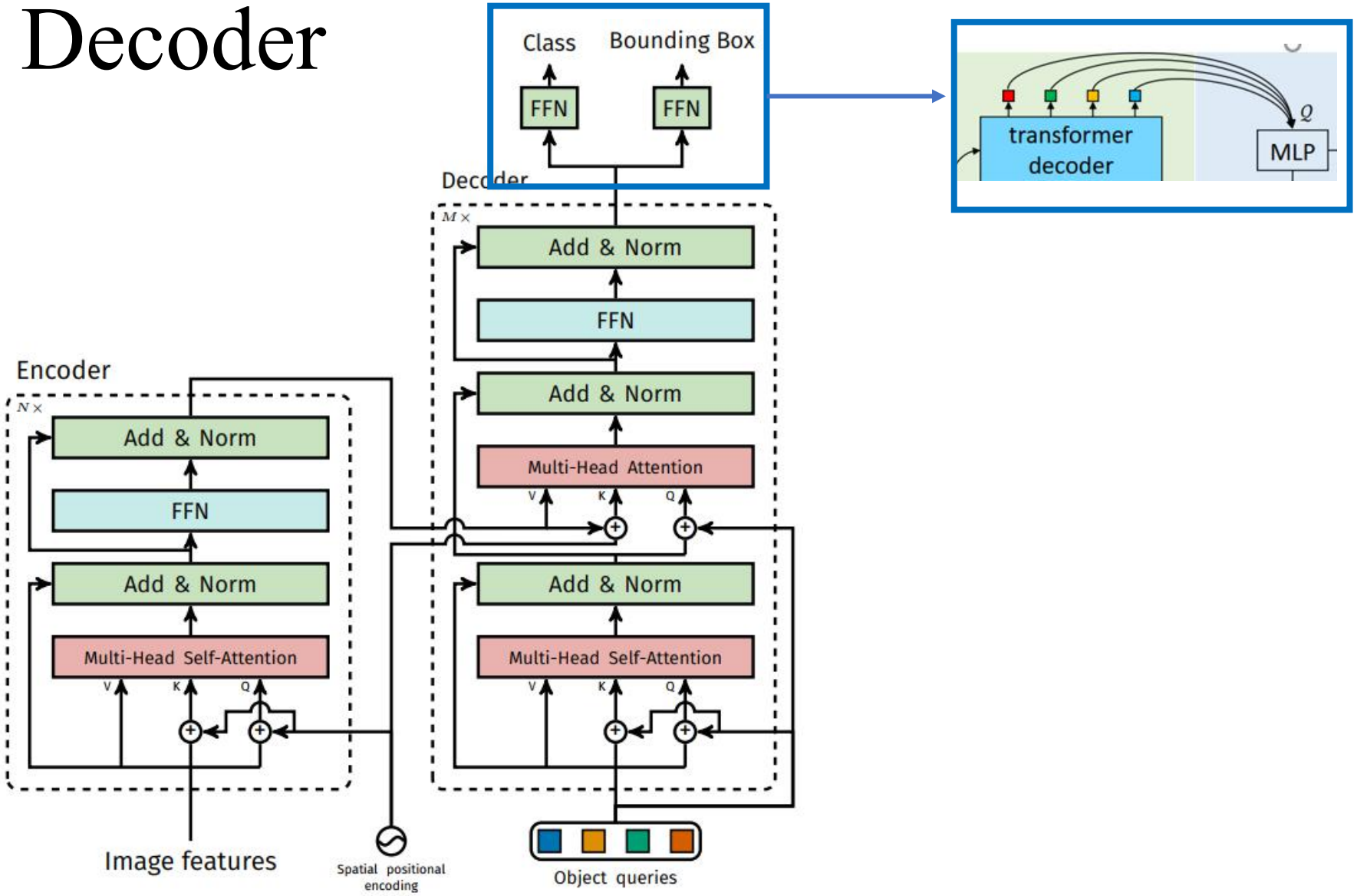


Figure 1: The Transformer - model architecture.

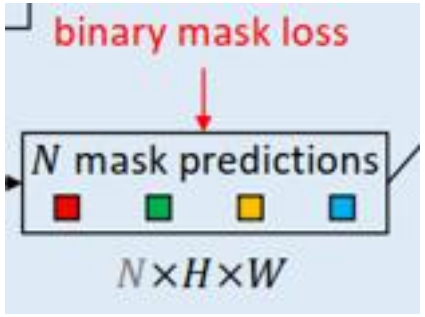


Transformer Decoder

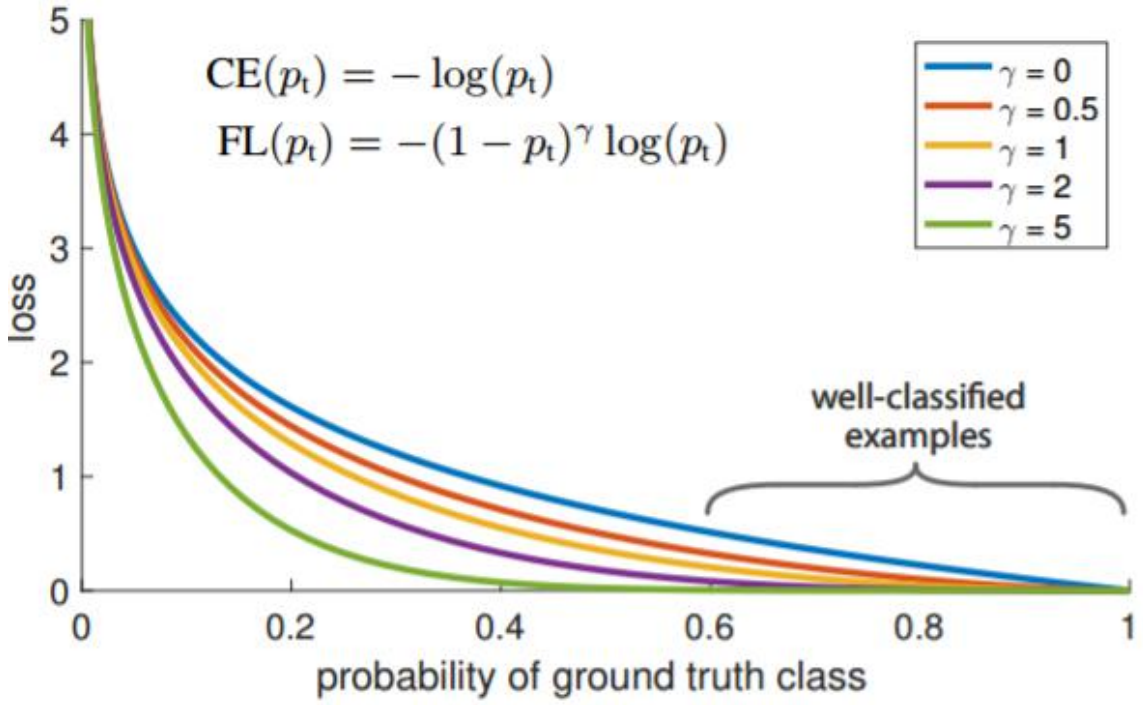


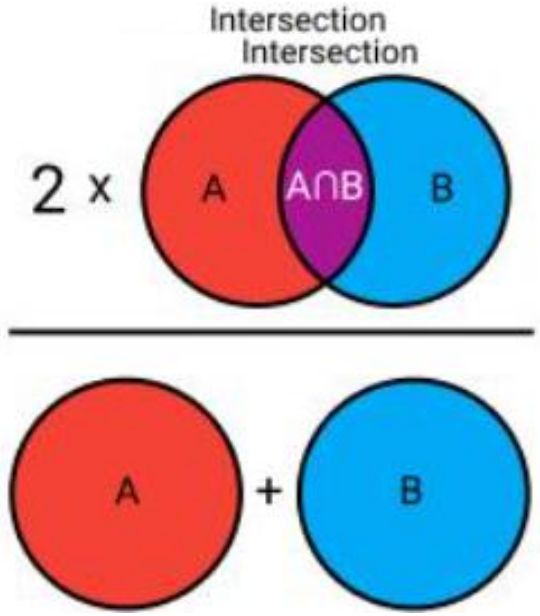
Architecture of DETR's transformer

Binary mask loss



Focal loss + Dice loss

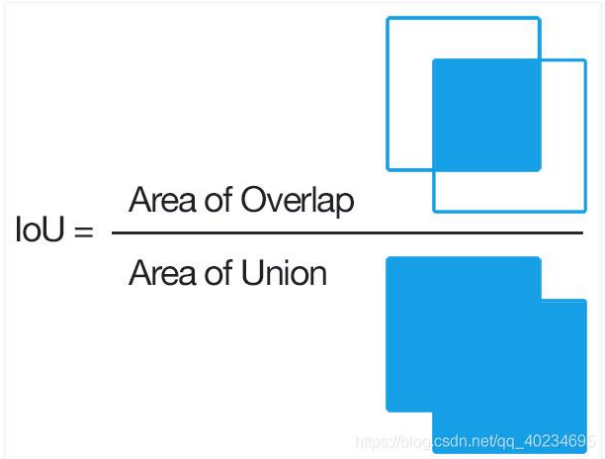




Experimental Result

Evaluation metrics:

mIou(mean Iou)



PQ(panoptic quality computation)

$$\mathrm{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

Experimental Result

Table 1: **Semantic segmentation on ADE20K `val` with 150 categories.** Mask classification-based MaskFormer outperforms the best per-pixel classification approaches while using fewer parameters and less computation. We report both single-scale (s.s.) and multi-scale (m.s.) inference results with $\pm$*std*. FLOPs are computed for the given crop size. Frame-per-second (fps) is measured on a V100 GPU with a batch size of 1.[3] Backbones pre-trained on ImageNet-22K are marked with †.

	method	backbone	crop size	mIoU (s.s.)	mIoU (m.s.)	#params.	FLOPs	fps
CNN backbones	OCRNet [50]	R101c	520 × 520	-	45.3	-	-	-
	DeepLabV3+ [9]	R50c	512 × 512	44.0	44.9	44M	177G	21.0
		R101c	512 × 512	45.5	46.4	63M	255G	14.2
	MaskFormer (ours)	R50	512 × 512	44.5 ±0.5	46.7 ±0.6	41M	53G	24.5
		R101	512 × 512	45.5 ±0.5	47.2 ±0.2	60M	73G	19.5
		R101c	512 × 512	**46.0** ±0.1	**48.1** ±0.2	60M	80G	19.0
Transformer backbones	SETR [53]	ViT-L†	512 × 512	-	50.3	308M	-	-
	Swin-UperNet [29, 49]	Swin-T	512 × 512	-	46.1	60M	236G	18.5
		Swin-S	512 × 512	-	49.3	81M	259G	15.2
		Swin-B†	640 × 640	-	51.6	121M	471G	8.7
		Swin-L†	640 × 640	-	53.5	234M	647G	6.2
	MaskFormer (ours)	Swin-T	512 × 512	46.7 ±0.7	48.8 ±0.6	42M	55G	22.1
		Swin-S	512 × 512	49.8 ±0.4	51.0 ±0.4	63M	79G	19.6
		Swin-B†	640 × 640	52.7 ±0.4	53.9 ±0.2	102M	195G	12.6
		Swin-L†	640 × 640	**54.1** ±0.2	**55.6** ±0.1	212M	375G	7.9

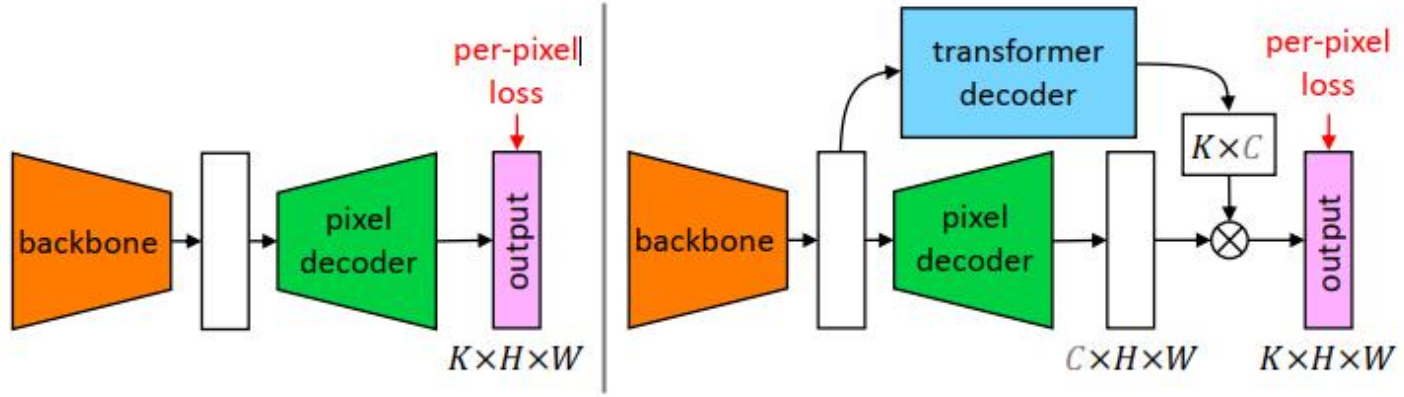
Experimental Result

Table 3: **Panoptic segmentation on COCO panoptic `val` with 133 categories.** MaskFormer seamlessly unifies semantic- and instance-level segmentation without modifying the model architecture or loss. Our model, which achieves better results, can be regarded as a box-free simplification of DETR [4]. The major improvement comes from "stuff" classes (PQ^{St}) which are ambiguous to represent with bounding boxes. For MaskFormer (DETR) we use the exact same post-processing as DETR. Note, that in this setting MaskFormer performance is still better than DETR (+2.2 PQ). Our model also outperforms recently proposed Max-DeepLab [42] without the need of sophisticated auxiliary losses, while being more efficient. FLOPs are computed as the average FLOPs over 100 validation images (COCO images have varying sizes). Frame-per-second (fps) is measured on a V100 GPU with a batch size of 1 by taking the average runtime on the entire `val` set *including post-processing time*. Backbones pre-trained on ImageNet-22K are marked with †.

	method	backbone	PQ	PQ^{Th}	PQ^{St}	SQ	RQ	#params.	FLOPs	fps
CNN backbones	DETR [4]	R50 + 6 Enc	43.4	48.2	36.3	79.3	53.8	-	-	-
	MaskFormer (DETR)	R50 + 6 Enc	45.6	50.0 (+1.8)	39.0 (+2.7)	80.2	55.8	-	-	-
	MaskFormer (ours)	R50 + 6 Enc	**46.5**	**51.0** (+2.8)	**39.8** (+3.5)	**80.4**	**56.8**	45M	181G	17.6
	DETR [4]	R101 + 6 Enc	45.1	50.5	37.0	79.9	55.5	-	-	-
	MaskFormer (ours)	R101 + 6 Enc	**47.6**	**52.5** (+2.0)	**40.3** (+3.3)	**80.7**	**58.0**	64M	248G	14.0
Transformer backbones	Max-DeepLab [42]	Max-S	48.4	53.0	41.5	-	-	62M	324G	7.6
		Max-L	51.1	57.0	42.2	-	-	451M	3692G	-
	MaskFormer (ours)	Swin-T	47.7	51.7	41.7	80.4	58.3	42M	179G	17.0
		Swin-S	49.7	54.4	42.6	80.9	60.4	63M	259G	12.4
		Swin-B	51.1	56.3	43.2	81.4	61.8	102M	411G	8.4
		Swin-B†	51.8	56.9	**44.1**	81.4	62.6	102M	411G	8.4
		Swin-L†	**52.7**	**58.5**	**44.0**	**81.8**	**63.5**	212M	792G	5.2

Experimental Result

Baseline models:



(a) PerPixelBaseline (b) PerPixelBaseline+

Experimental Result

When is Mask classification better than per-pixel classification?

Table 2: **MaskFormer *vs.* per-pixel classification baselines on 4 semantic segmentation datasets.** MaskFormer improvement is larger when the number of classes is larger. We use a ResNet-50 backbone and report single scale mIoU and PQ^{St} for ADE20K, COCO-Stuff and ADE20K-Full, whereas for higher-resolution Cityscapes we use a deeper ResNet-101 backbone following [8, 9].

	Cityscapes (19 classes)		ADE20K (150 classes)		COCO-Stuff (171 classes)		ADE20K-Full (847 classes)	
	mIoU	PQ^{St}	mIoU	PQ^{St}	mIoU	PQ^{St}	mIoU	PQ^{St}
PerPixelBaseline	77.4	58.9	39.2	21.6	32.4	15.5	12.4	5.8
PerPixelBaseline+	**78.5**	60.2	41.9	28.3	34.2	24.6	13.9	9.0
MaskFormer (ours)	**78.5** (+0.0)	**63.1** (+2.9)	**44.5** (+2.6)	**33.4** (+5.1)	**37.1** (+2.9)	**28.9** (+4.3)	**17.4** (+3.5)	**11.9** (+2.9)

$$\mathrm{PQ} = \underbrace{\frac{\sum_{(p,g)\in TP} \mathrm{IoU}(p,g)}{|TP|}}_{\text{segmentation quality (SQ)}} \times \underbrace{\frac{|TP|}{|TP| + \frac{1}{2}|FP| + \frac{1}{2}|FN|}}_{\text{recognition quality (RQ)}}$$

(1)Table 2中有一个很有意思的结果：当类别越多的时候mask classification模型的提升越大。

(2) 还发现，虽然Cityscapes上mIoU和baseline一样，但是MaskFormer的PQ还是要高不少。我们发现PQ的增长主要来自RQ (recognition quality)，和baseline相比SQ (segmentation quality)反而更低。所以mask classification模型(或者是MaskFormer)面临主要问题是如何生产更高质量的binary mask。这也说明了semantic segmentation还有提升的空间。

Ablation studies

Table 4: **Per-pixel *vs*. mask classification for semantic segmentation.** All models use 150 queries for a fair comparison. We evaluate the models on ADE20K `val` with 150 categories. 4a: PerPixel-Baseline+ and MaskFormer-fixed use similar fixed matching (*i.e.*, matching by category index), this result confirms that the shift from per-pixel to masks classification is the key. 4a: bipartite matching is not only more flexible (can make less prediction than total class count) but also gives better results.

(a) Per-pixel *vs*. mask classification.

	mIoU	PQ^{St}
PerPixelBaseline+	41.9	28.3
MaskFormer-fixed	**43.7** (+1.8)	**30.3** (+2.0)

(b) Fixed *vs*. bipartite matching assignment.

	mIoU	PQ^{St}
MaskFormer-fixed	43.7	30.3
MaskFormer-bipartite (ours)	**44.2** (+0.5)	**33.4** (+3.1)

Ablation studies

Number of queries. The table to the right shows the results of MaskFormer trained with a varying number of queries on datasets with different number of categories. The model with 100 queries consistently performs the best across multiple datasets. This suggest we do not need to adjust the number of queries w.r.t. the number of categories or datasets. Interestingly, even with only 20 queries MaskFormer outperforms our per-pixel classification baseline.

# of queries	ADE20K		COCO-Stuff		ADE20K-Full	
	mIoU	PQ^{St}	mIoU	PQ^{St}	mIoU	PQ^{St}
PerPixelBaseline+	41.9	28.3	34.2	24.6	13.9	9.0
20	42.9	32.6	35.0	27.6	14.1	10.8
50	43.9	32.7	35.5	27.9	15.4	11.1
100	**44.5**	**33.4**	**37.1**	**28.9**	**16.0**	**11.9**
150	44.2	**33.4**	37.0	**28.9**	15.5	11.5
300	43.5	32.3	36.1	29.1	14.2	10.3
1000	35.4	26.7	34.4	27.6	8.0	5.8

Their guess is that the number of queries may be related to the average number of categories in each image (after all, ADE20K doesn't have 150 categories in every image).

Conclusion

1. Mask classification not only achieves better results, but also is faster.

2. Mask Classification can solve both semantic segmentation and instance segmentation.

Thanks for listening