Per-Pixel Classification is Not All You Need for Semantic Segmentation

论文地址：https://arxiv.org/abs/2107.06278

代码地址：https://github.com/facebookresearch/MaskFormer

任务背景

mask prediction通过一系列二值mask来对像素进行预测分类，不仅可以解决实例分割任务还可以解决语义分割任务，实现了语义分割和实例分割任务的统一。因此本文尝试提出一种统一的框架来同时解决语义分割和实例分割任务。

方法介绍

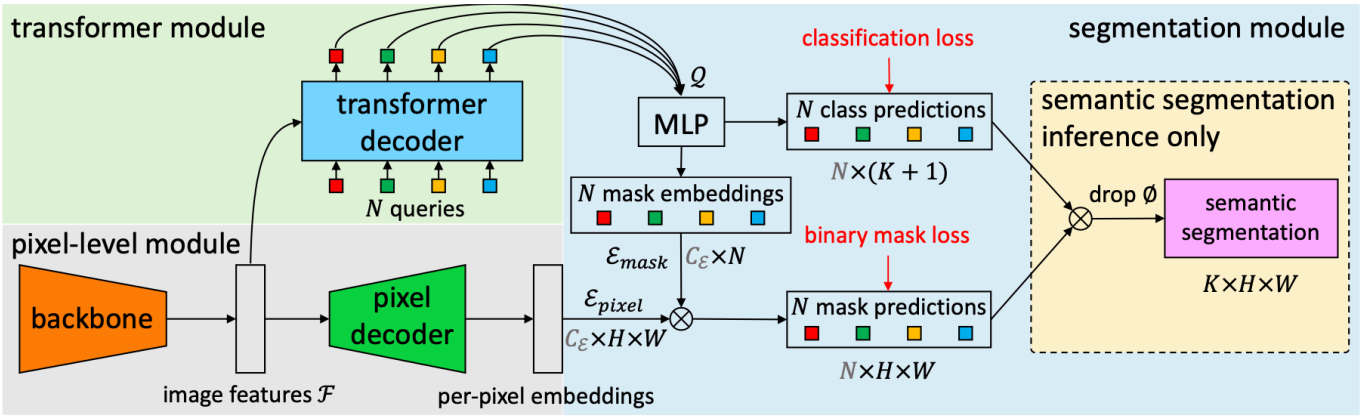

Figure 2: **MaskFormer overview.** We use a backbone to extract image features $\mathcal{F}$. A pixel decoder gradually upsamples image features to extract per-pixel embeddings $\mathcal{E}_{\text{pixel}}$. A transformer decoder attends to image features and produces N per-segment embeddings $\mathcal{Q}$. 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.

模型框架

提出了一个maskformer框架，将语义分割任务划分成proposal segmentation和proposal classification子任务，通过一个backbone生成高维特征向量，接上两个分支，一个分支就是常见的分割decoder，得到输入图像的每个像素的特征表示。另一个分支接上一个transformer decoder，根据输入的N个query得到得到N个mask的特征表示，利用gt的类别进行监督生成N个mask的类别表示，实现proposal classification子任务，通过N个mask的特征表示和第一个分支生成的逐像素特征表示进行内积得到逐像素的mask表示，实现proposal segmentation子任务。通过这两个子任务得到的结果进行内积得到逐像素的分类，实现语义分割任务。由于同一种类别可能存在在不同的proposal中，所以可以转化为实例分割任务。

实验结果

实验数据集包括ADE20K和COCO，在两个数据集上都达到了SOTA（55.6 mIoU on ADE20K，52.7 PQ on COCO）。

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 ±*std*. FLOPs are computed for the given crop size. Frames-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 [44]	R101c	520 × 520	-	45.3	-	-	-
	DeepLabV3+ [8]	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 [47]	ViT-L†	512 × 512	-	50.3	308M	-	-
	Swin-UperNet [27, 43]	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	51.1 ±0.2	52.3 ±0.4	102M	195G	12.6
		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

PerPixelBaseline使用了MaskFormer的逐像素分类部分，直接输出逐像素的score。

PerPixelBaseline+在PerPixelBaseline基础上加上了transformer模块和mask embedding的MLP模块。

MaskFormer和PerPixelBaseline+的区别就是一个是采用逐像素分类的方式，另一个是采用mask classification分类的方式。

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 [7, 8].

	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)

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 [3]. 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 [38] 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). Frames-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 [3]	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 [3]	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 [38]	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