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

Bowen Cheng[1,2*] **Alexander G. Schwing**[2] **Alexander Kirillov**[1]

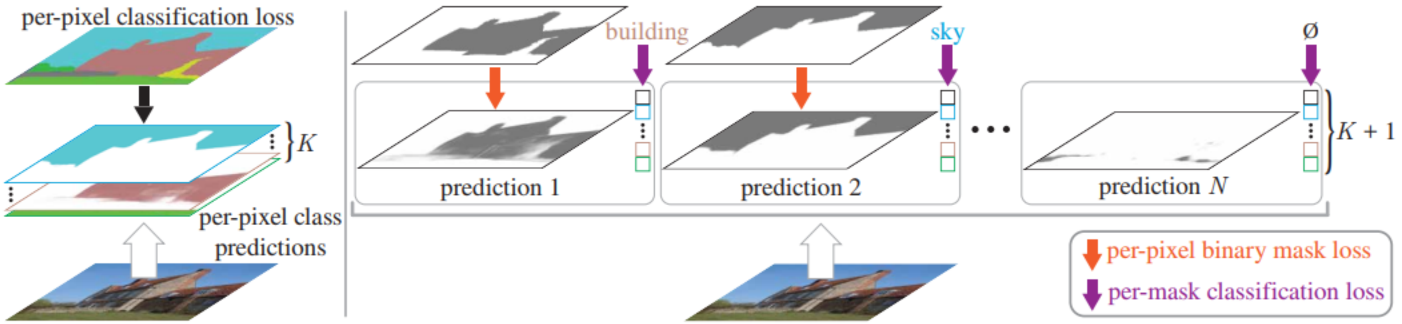
[1]Facebook AI Research (FAIR) [2]University of Illinois at Urbana-Champaign (UIUC)

Contribution

- mask classification模型可以同时解决语义分割和实例分割问题，并且我们发现这个模型甚至不用做任何改动：包括模型结构(model architecture)，训练的loss，以及训练方法。
- mask classification模型在语义分割上不仅比像素分类模型的结果更好，而且需要更少的参数和计算量。

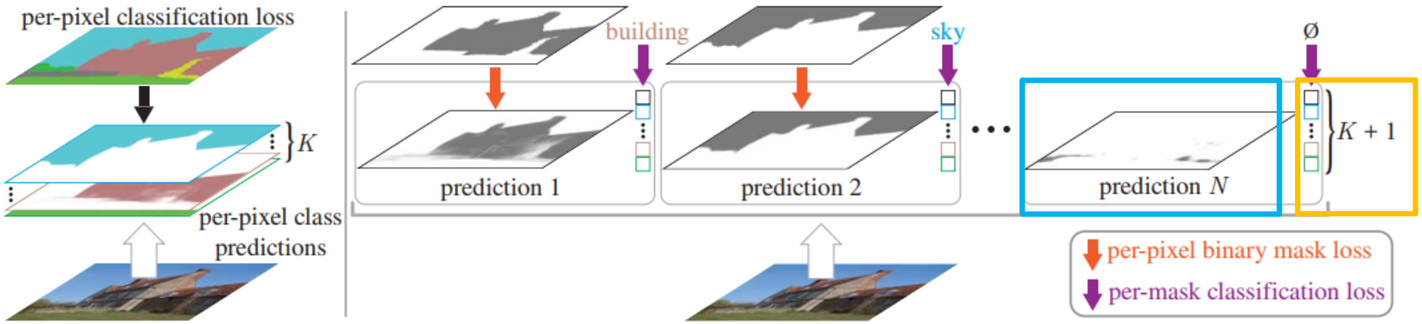
MaskFormer

- MaskFormer employs a Transformer decoder [41] to compute a set of pairs, each consisting of a class prediction and a mask embedding vector.
- The mask embedding vector is used to get the binary mask prediction via a dot product with the per-pixel embedding obtained from an underlying fully-convolutional network.



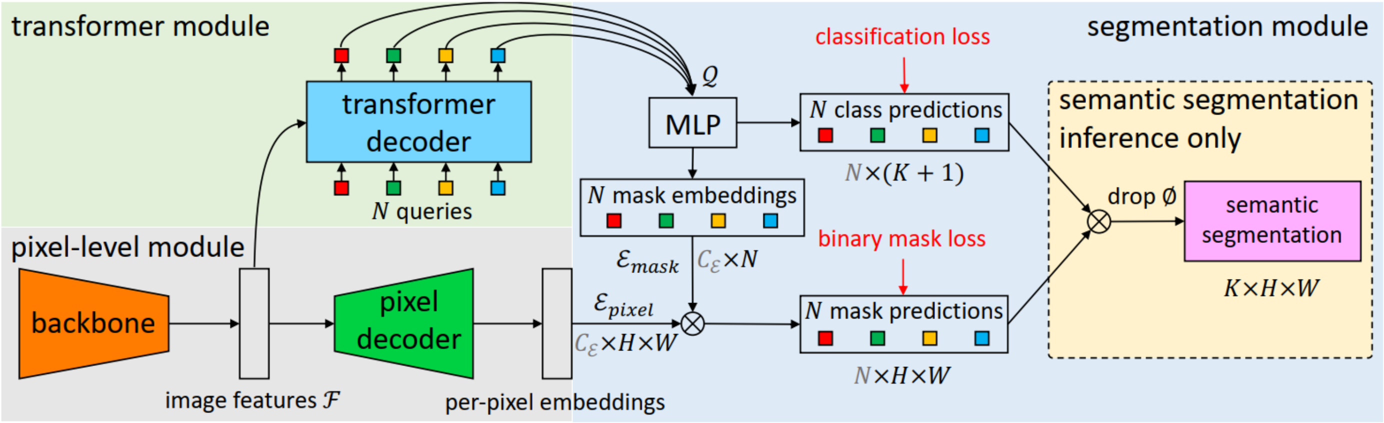
MaskFormer

- MaskFormer employs a Transformer decoder [41] to compute a set of pairs, each consisting of a class prediction and a mask embedding vector.
- The mask embedding vector is used to get the binary mask prediction via a dot product with the per-pixel embedding obtained from an underlying fully-convolutional network.



MaskFormer

- The model contains three modules :
- 1**) a pixel-level modul**e that extracts per-pixel embeddings used to generate binary mask predictions;
- 2) **a transformer module**, where a stack of Transformer decoder layers [41] computes *N* per-segment embeddings;
- 3) **a segmentation module**, which generates predictions $\{(p_i, m_i)\}_{i=1}^{N}$ from these embeddings.



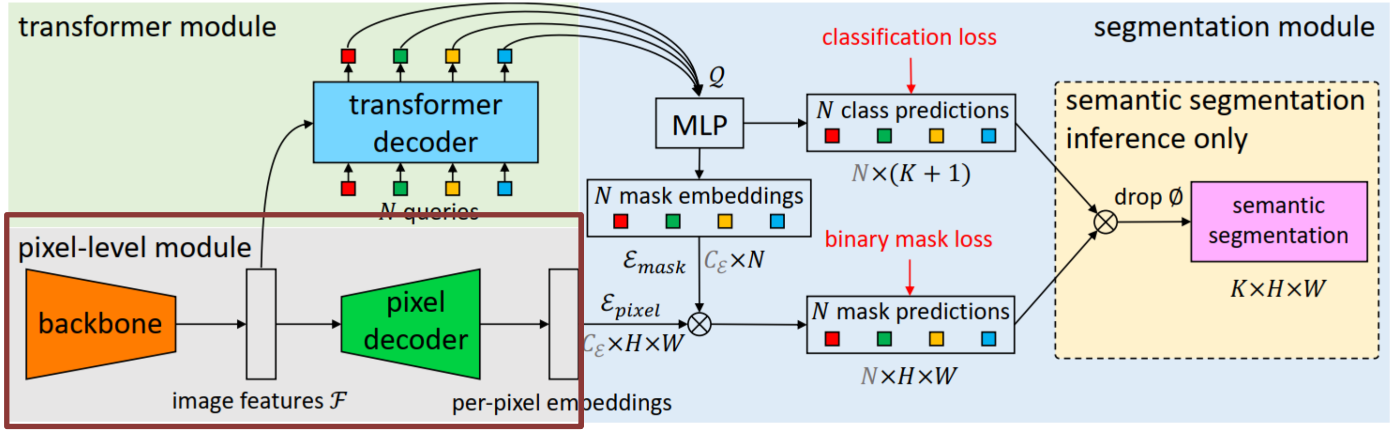
Pixel-level module

- **1) a pixel-level module** that extracts per-pixel embeddings used to generate binary mask predictions;

  backbone down-sample to 1/32.

  decoder upsample 32.

  此部分与大部分per-pixel classificationbased segmentation 是相同的。

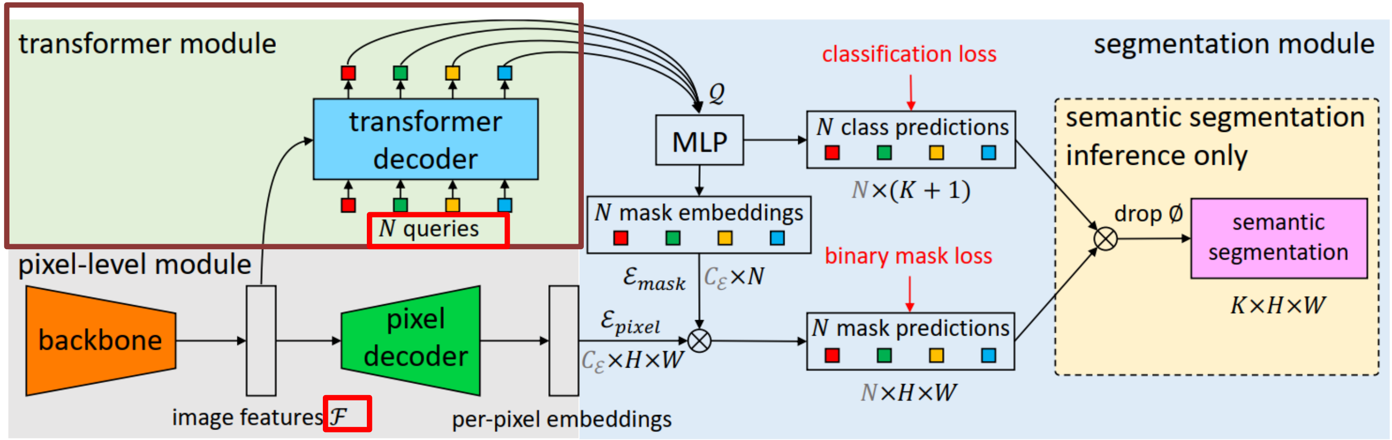


Transformer module

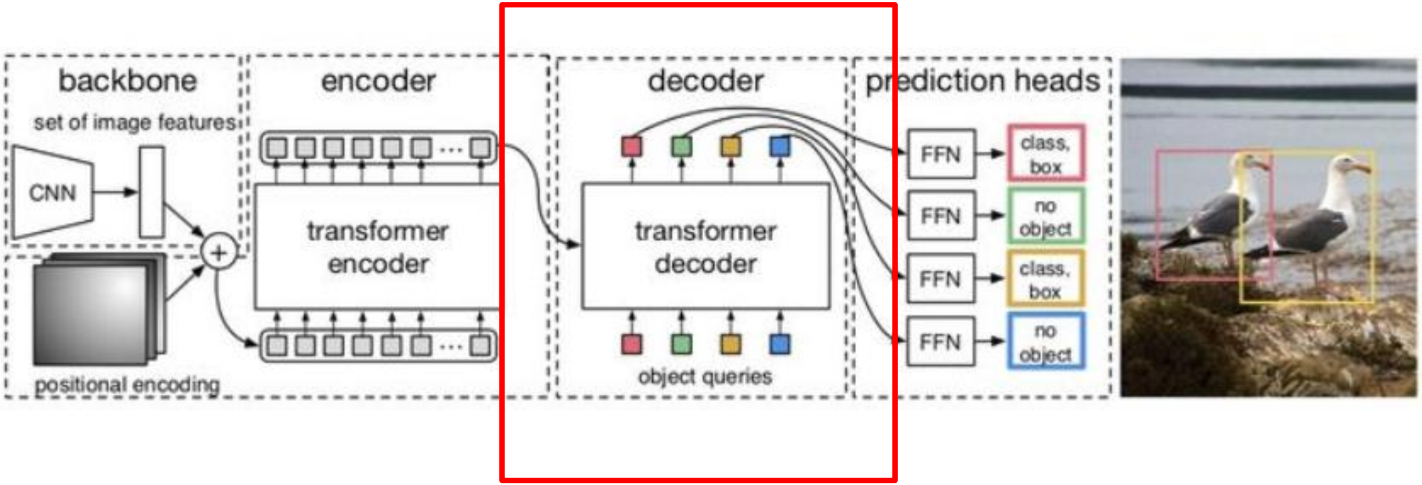
- 2) **a transformer module**, where a stack of Transformer decoder layers [DERT] computes *N* per-segment embeddings;

***input*:** features F (value) and *N* learnable positional embeddings (*i.e.*, queries)

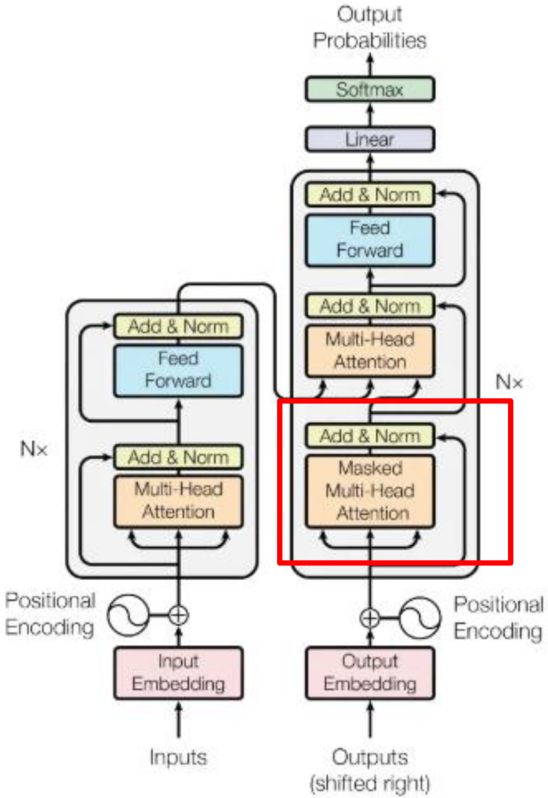
***output*:** *N* per-segment embeddings $\mathcal{Q} \in \mathbb{R}^{C_{\mathcal{Q}} \times N}$



DERT

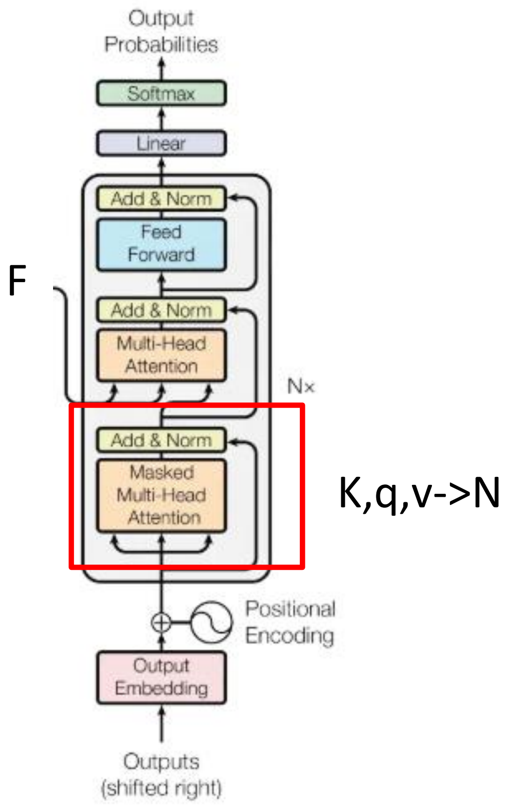

backbone
set of image features
CNN
positional encoding
encoder
transformer encoder
decoder
transformer decoder
object queries
prediction heads
FFN
FFN
FFN
FFN
class, box
no object
class, box
no object

DERT


Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Add & Norm
Masked
Multi-Head
Attention
Nx
Add & Norm
Feed
Forward
Add & Norm
Multi-Head
Attention
Nx
Positional
Encoding
Input
Embedding
Positional
Encoding
Output
Embedding
Inputs
Outputs
(shifted right)

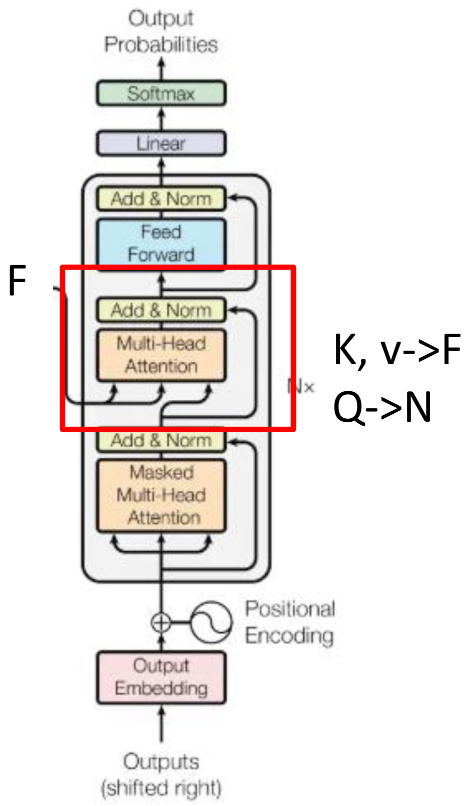
K,q,v->F

DERT



Output
Probabilities
Softmax
Linear
Add & Norm
Feed
Forward
F
Add & Norm
Multi-Head
Attention
Nx
Add & Norm
Masked
Multi-Head
Attention
K,q,v->N
Positional
Encoding
Output
Embedding
Outputs
(shifted right)

DERT

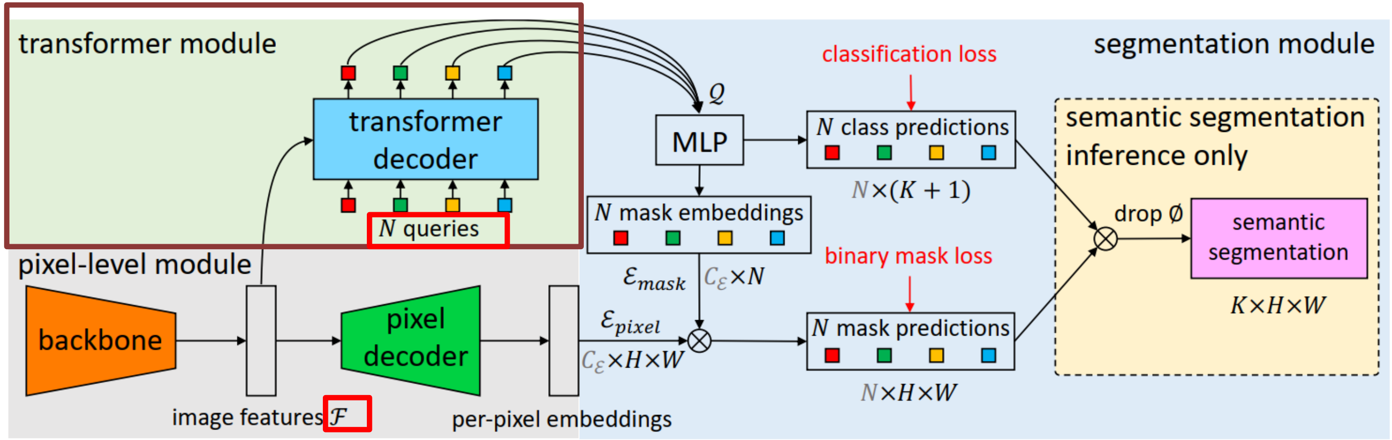


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Add & Norm
Masked Multi-Head Attention
Positional Encoding
Output Embedding
Outputs (shifted right)
F
K, v->F
Q->N

Transformer module

- 2) **a transformer module**, where a stack of Transformer decoder layers [DERT] computes *N* per-segment embeddings;

***input*：** features F and *N* learnable positional embeddings (*i.e.*, queries)
***output*：** *N* per-segment embeddings $\mathcal{Q} \in \mathbb{R}^{C_{\mathcal{Q}} \times N}$

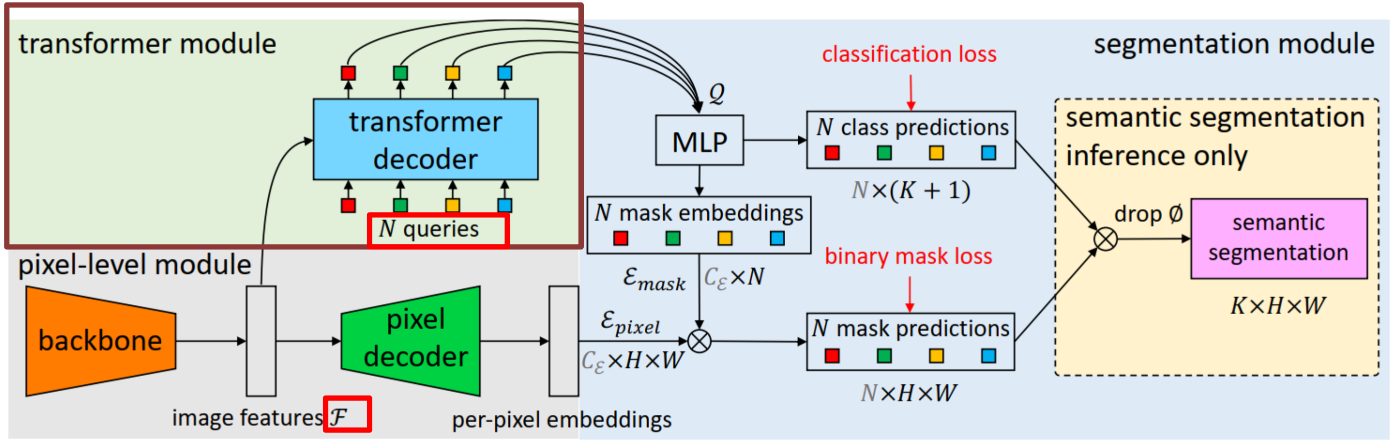


Transformer module

- 2) **a transformer module**, where a stack of Transformer decoder layers [DERT] computes *N* per-segment embeddings;

input： features F and *N* learnable positional embeddings (*i.e.*, queries)
output： *N* per-segment embeddings $\mathcal{Q} \in \mathbb{R}^{C_{\mathcal{Q}} \times N}$

we assume $N \geq N$gt and pad the set of ground truth labels with "no object" tokens to allow one-to-one matching



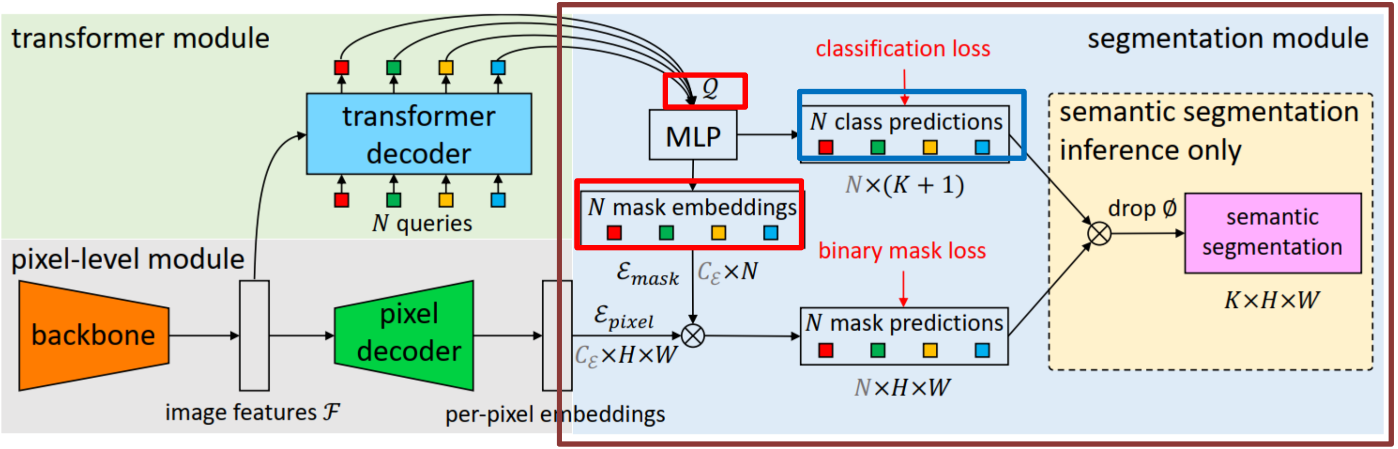
Segmentation module

- 3) **a segmentation module**, which generates predictions $\{(p_i, m_i)\}_{i=1}^{N}$ from these embeddings.

$\mathcal{Q} \in \mathbb{R}^{C_{\mathcal{Q}} \times N}$

$\boxed{\{p_i \in \Delta^{K+1}\}_{i=1}^{N}}$ a linear classifier + a softmax activation class probability predictions

$\boxed{\mathcal{E}_{\text{mask}} \in \mathbb{R}^{C_{\mathcal{E}} \times N}}$ MLP(2 hidden layers)



Segmentation module

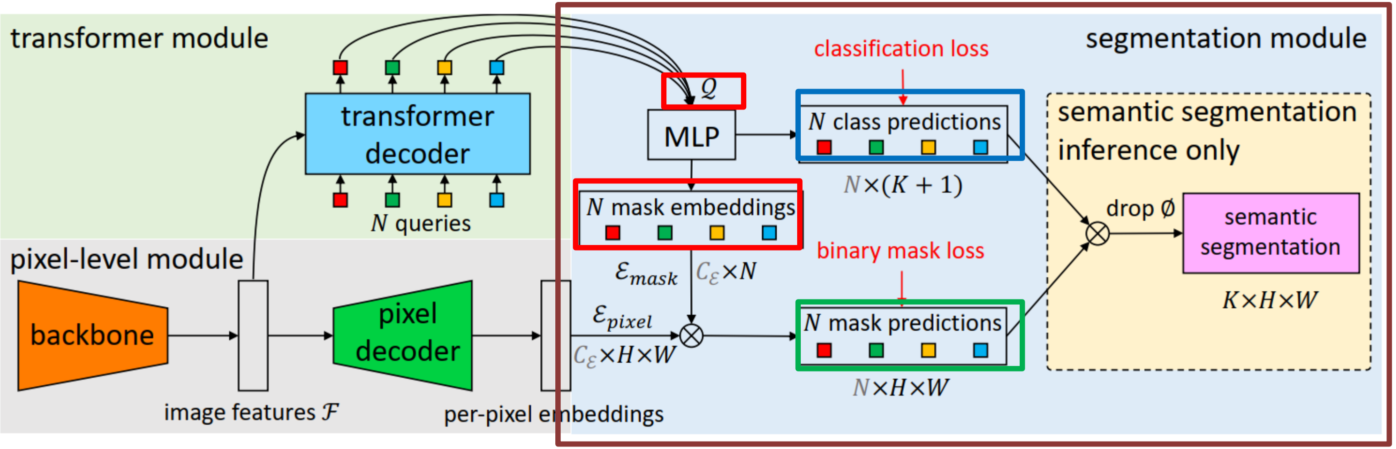
- 3) **a segmentation module**, which generates predictions $\{(p_i, m_i)\}_{i=1}^{N}$ from these embeddings.

$\mathcal{Q} \in \mathbb{R}^{C_{\mathcal{Q}} \times N}$

$\{p_i \in \Delta^{K+1}\}_{i=1}^{N}$ a linear classifier + a softmax activation class probability predictions

$\mathcal{E}_{\text{mask}} \in \mathbb{R}^{C_{\mathcal{E}} \times N}$ Dot product +a sigmoid activation

$$m_i[h, w] = \text{sigmoid}(\mathcal{E}_{\text{mask}}[:, i]^{\mathrm{T}} \cdot \mathcal{E}_{\text{pixel}}[:, h, w]).$$



Segmentation module

- 3) **a segmentation module**, which generates predictions $\{(p_i, m_i)\}_{i=1}^{N}$ from these embeddings.

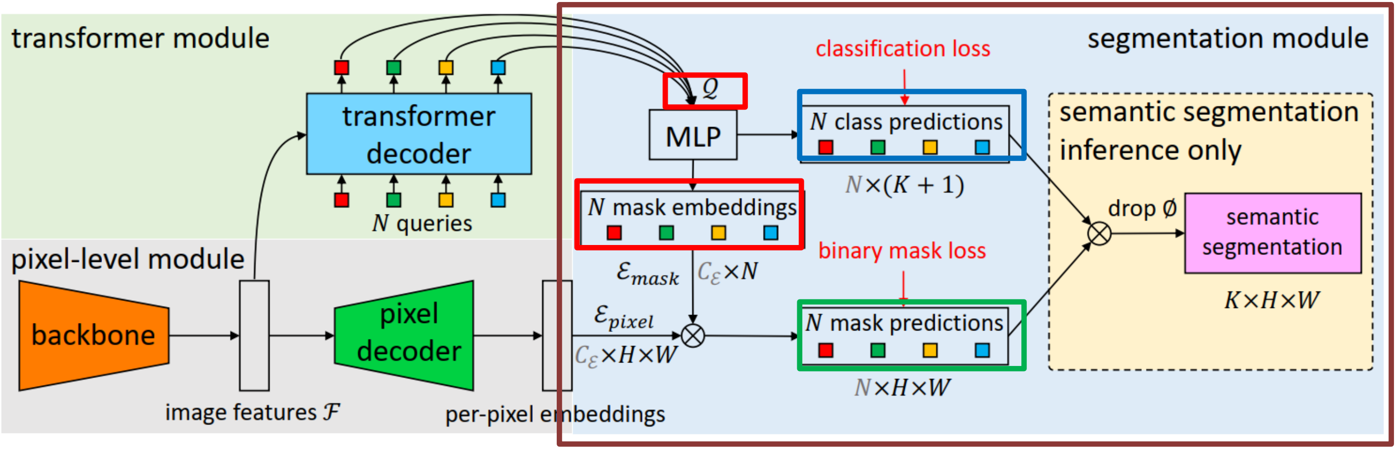
$\mathcal{Q} \in \mathbb{R}^{C_{\mathcal{Q}} \times N}$

$\boxed{\{p_i \in \Delta^{K+1}\}_{i=1}^{N}}$ a linear classifier + a softmax activation class probability predictions

$\boxed{\mathcal{E}_{\text{mask}} \in \mathbb{R}^{C_{\mathcal{E}} \times N}}$ Dot product +a sigmoid activation

$$m_i[h, w] = \text{sigmoid}(\mathcal{E}_{\text{mask}}[:, i]^{\text{T}} \cdot \mathcal{E}_{\text{pixel}}[:, h, w]).$$

$\boxed{m_i \in [0, 1]^{H \times W}}$

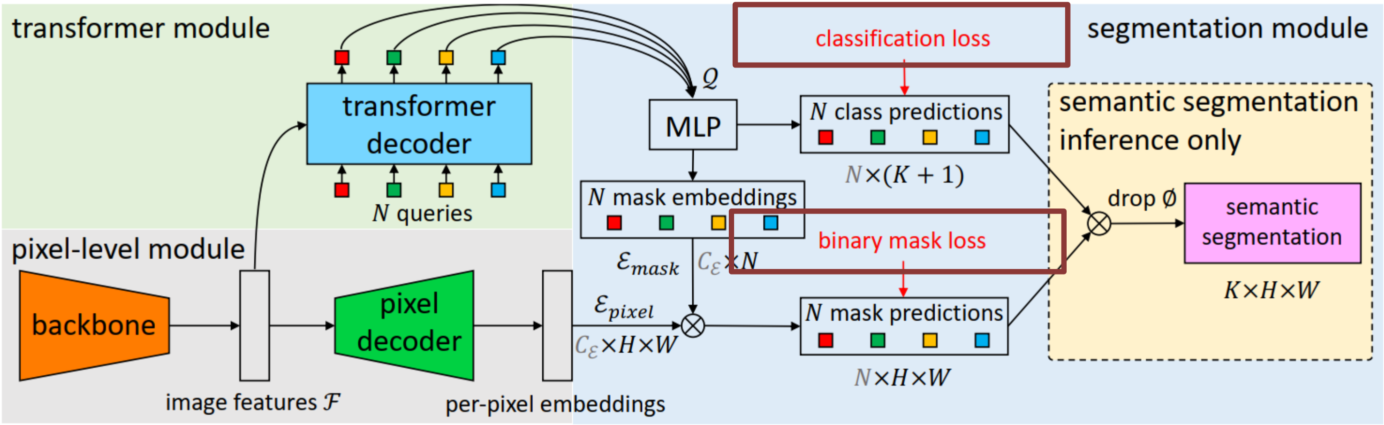


Loss

- for semantic and panoptic segmentation tasks：
- $\mathcal{L}_{\text{mask-cls}}$ a single classification loss per mask (cross entropy) and a per-pixel **binary** mask loss

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

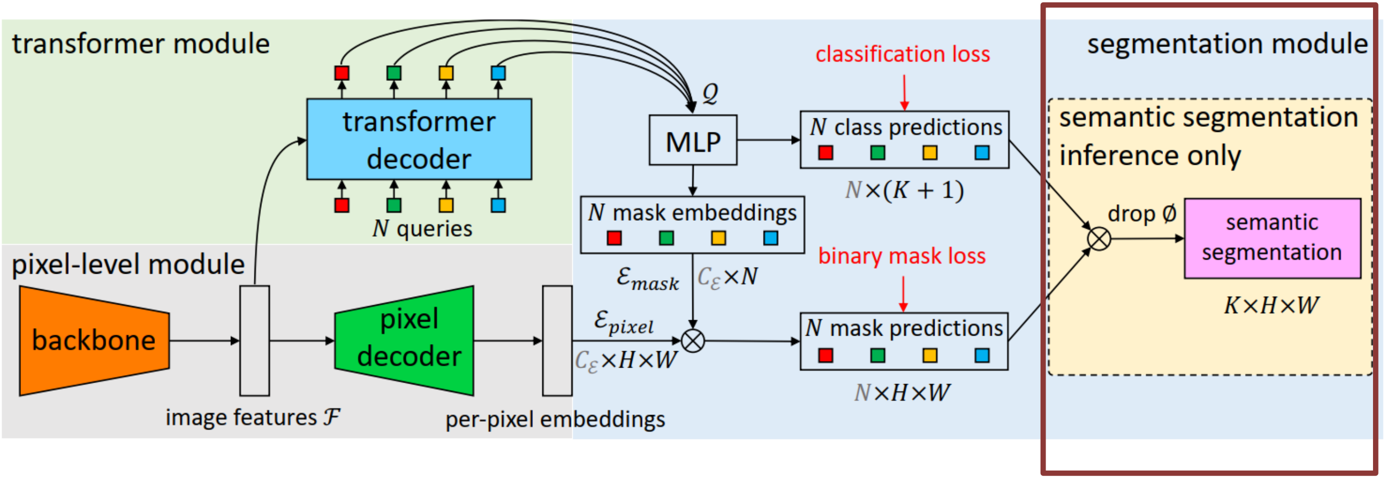
$\mathcal{L}_{\text{mask}}$ The same as DETR: a focal loss and a dice loss



Mask-classification inference

- converts mask classification outputs $\{(p_i, m_i)\}_{i=1}^{N}$ to either panoptic or semantic segmentation output formats.

- For **General inference :**
- For **Semantic inference** :



MaskFormer--mask-classification inference

- converts mask classification outputs $\{(p_i, m_i)\}_{i=1}^N$ to either panoptic or semantic segmentation output formats.

- For **General inference :**

$$\arg\max_{\boxed{i:c_i \neq \varnothing}} p_i(c_i) \cdot m_i[h, w].$$

对pixel(h,w)遍历所有N masks，计算pixel(h,w)在每个图上的$p_i(c_i) \cdot m_i[h, w]$，找到此值最大的那个masks，即为pixel(h,w)的实际label。

注:$p_i(c_i)$ 此时每个mask代表的类别为ci

$c_i = \arg\max_{c \in \{1,\dots,K,\varnothing\}} p_i(c)$ is the most likely class label for each probability-mask pair i (N)

MaskFormer--mask-classification inference

- converts mask classification outputs $\{(p_i, m_i)\}_{i=1}^N$ to either panoptic or semantic segmentation output formats.

- For **General inference :**

$$\arg\max_{\boxed{i:c_i \neq \varnothing}} p_i(c_i) \cdot m_i[h, w].$$

- **reduce false positive rates :**

  1. **filter out** low-confidence predictions prior to inference

  2. **remove** predicted segments that have large parts of their binary masks (*mi* > 0:5) occluded by other predictions.

MaskFormer--mask-classification inference

- converts mask classification outputs $\{(p_i, m_i)\}_{i=1}^{N}$ to either panoptic or semantic segmentation output formats.

- For **Semantic inference** :

  $$\arg\max_{\boxed{c\in}\{1,\ldots,K\}} \sum_{i=1}^{N} p_i(c) \cdot m_i[h, w]$$

  marginalization over probability-mask pairs yields better

  对pixel(h,w) 求和其在N个mask上的 $p_i(c) \cdot m_i[h, w]$ ，即 $\sum_{i=1}^{N} p_i(c) \cdot m_i[h, w]$，找到此值最大的那个class。

- 注：$p_i(c)$ 此时每个mask代表的类别已经被淡化。这时候ci是由p*m一起决定的，而之前是只由p决定的。

- N==K

Experiments--Implementation details

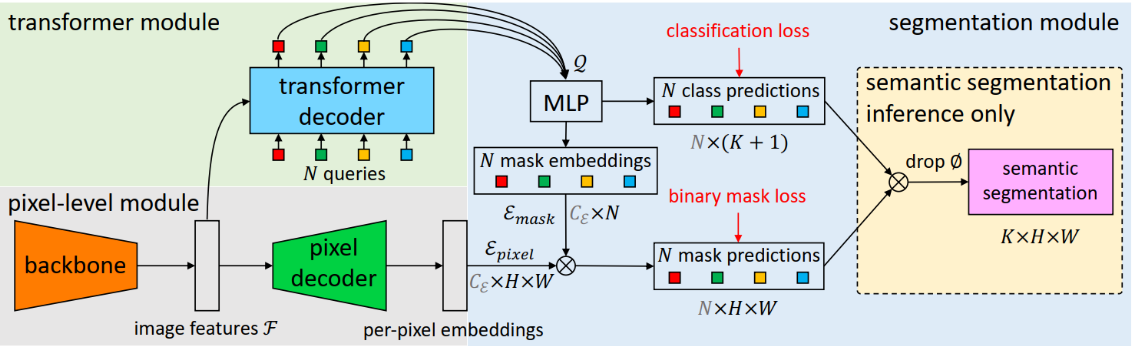
- **Backbone**

  ResNet backbones and Transformer-based Swin-Transformer
- **Pixel decoder**

  for MaskFormer, we design a light-weight pixel decoder based on the popular FPN architecture.
- **Transformer decoder**

  the same Transformer decoder design as DETR，

  The *N* query embeddings are initialized as zero vectors
- **Loss：**
- focal loss ： dice loss =20： 1
- **MLP：**
- 2 layer



Experiments--Training settings

Semantic segmentation

8 V100 GPUs

ADE20K：

512 $\times$ 512, a batch size of 16 and train all models for 160k iterations

COCO-Stuff-10k：

640 $\times$ 640, a batch size of 32 and train all models for 60k iterations

Panoptic segmentation.

COCO models are trained using 64 V100 GPUs

640 $\times$ 640, a batch size of 32 and train all models for 60k iterations

ADE20K experiments are trained with 8 V100 GPUs and 720k iterations and 640 $\times$ 640

We follow exactly the same architecture, loss, and training procedure as we use for semantic segmentation. The only difference is supervision: *i.e.*, category region masks in semantic segmentation *vs*. object instance masks in panoptic segmentation.

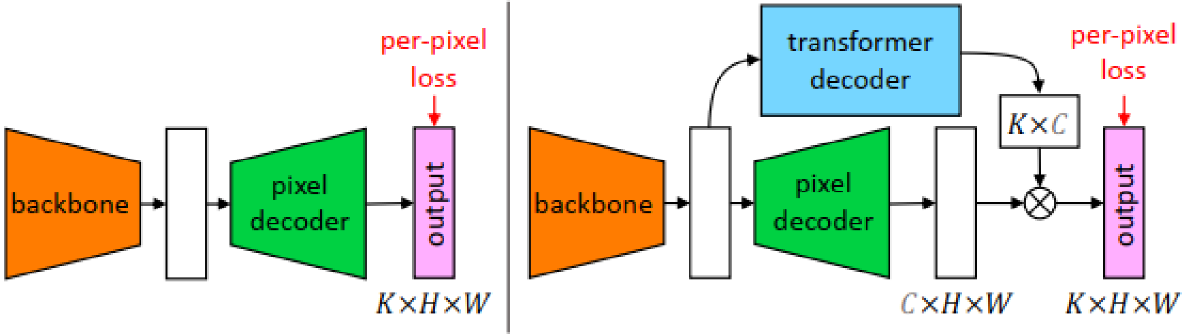
- **Semantic segmentation on ADE20K** val **with 150 categories.**

	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

- **MaskFormer *vs*. per-pixel classification baselines on 4 semantic segmentation datasets.**

	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)

- PerPixelBaseline+ and MaskFormer differ only in the formulation: per-pixel *vs*. mask classification.



(a) PerPixelBaseline (b) PerPixelBaseline+

当类别越多的时候
mask classification
模型的提升越大

- **Panoptic segmentation on COCO panoptic** val **with 133 categories.**

	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

- **Ablation studies**

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

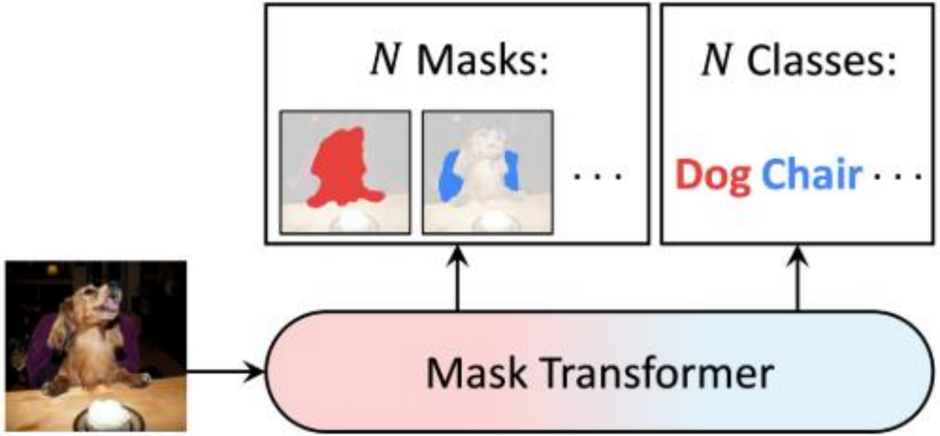
	mIoU	PQ^{St}
PerPixelBaseline+	41.9	28.3
MaskFormer-fixed	**43.7** (+1.8)	**30.3** (+2.0)

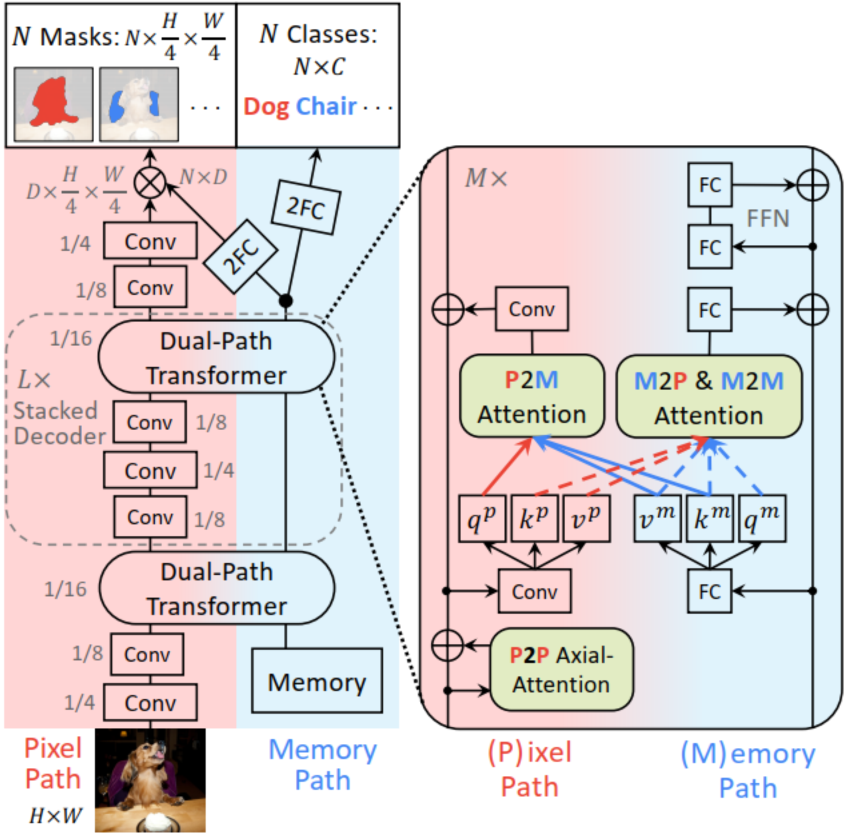
Number of queries.

# of queries	ADE20K mIoU	ADE20K PQ^{St}	COCO-Stuff mIoU	COCO-Stuff PQ^{St}	ADE20K-Full mIoU	ADE20K-Full 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

Max-Deeplab

- Max-Deeplab中，一张图会有N（最后为100）个query，每个query对应一个Mask和一个C分类结果，然后通过C分类的得分，将不符合要求的mask弃置，达到定长预测变成变长结果的效果，从而完成全景分割。
- Max-Deeplab两个分支都用了transformer，模型大很多的重要原因。





(a) Overview of MaX-DeepLab (b) Dual-path transformer block

相比Max-deeplab， maskformer更为简洁 体量小一点。
Max-Deeplab两个分支都用了transformer， 而Maskformer其中一个分支用了CNN。

auxiliary loss：

PQ-style loss
Instance discrimination
Mask-ID cross-entropy
Semantic segmentation