ICCV 2021

Instances as Queries

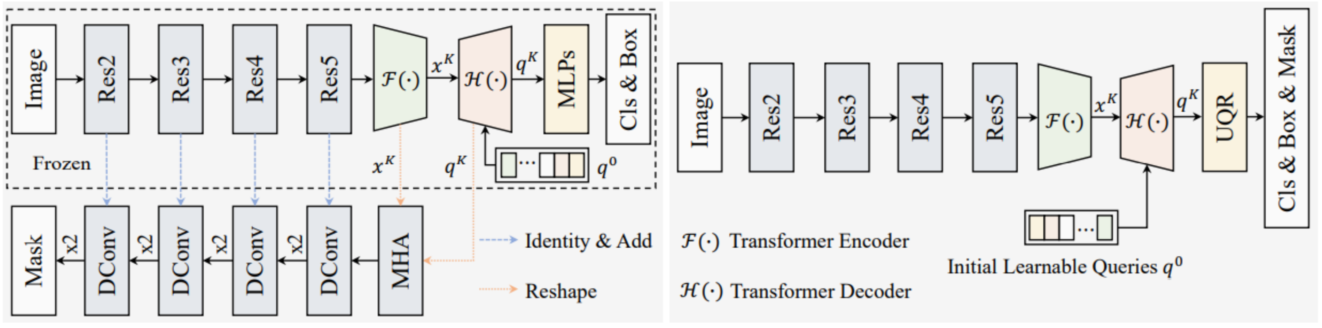
Yuxin Fang[1*], Shusheng Yang[1,2*], Xinggang Wang[1†], Yu Li[2], Chen Fang[3], Ying Shan[2], Bin Feng[1], Wenyu Liu[1]

[1]School of EIC, Huazhong University of Science & Technology
[2]Applied Research Center (ARC), Tencent PCG [3]Tencent

Mengxue

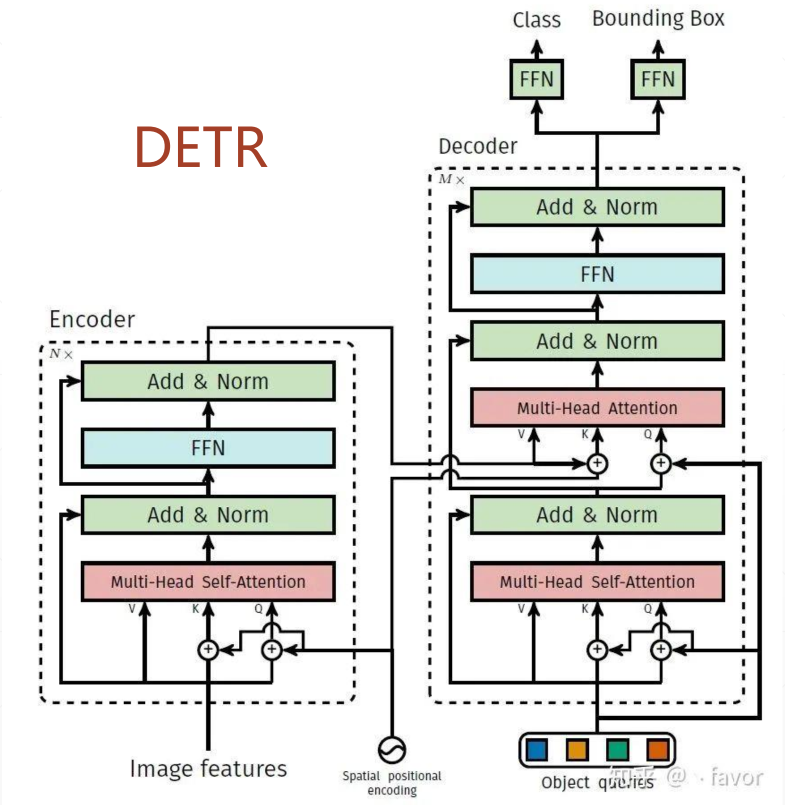
Motivation

- query based object detection frameworks achieve comparable performance



(a) DETR FPN-style instance segmentation (b) SOLQ directly-predict style instance segmentation

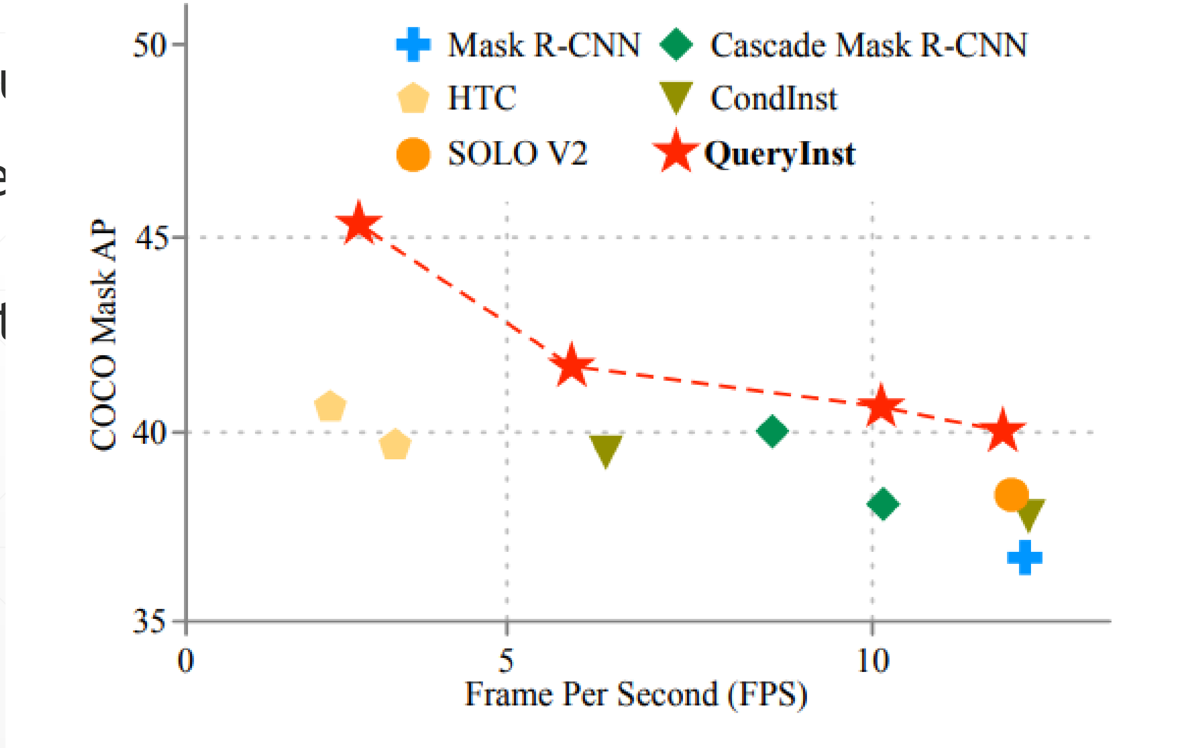
DETR



- How to fully leverage query to perform instance segmentation remains an open problem
- The gap of mask RoI feature and object queries

Contribution

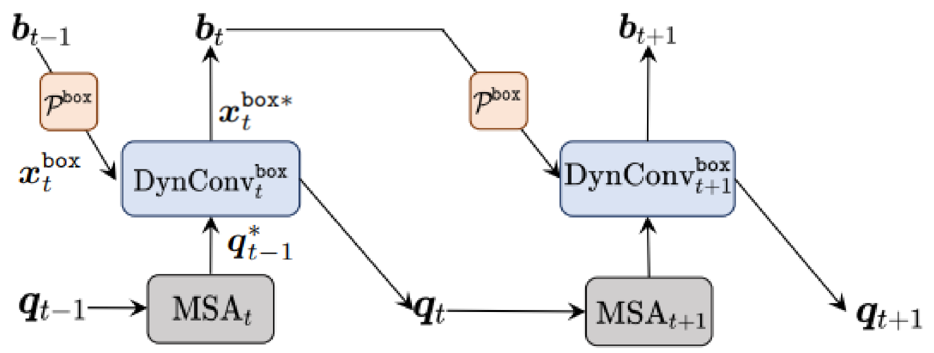
- We attempt to solve instance segmentation from a new perspective that uses parallel dynamic mask heads in the query based end-to-end detection framework.
- We set up a task-joint paradigm for qu
segmentation by leveraging the share
- We extend the QueryInst to video inst
adding a vanilla track head.



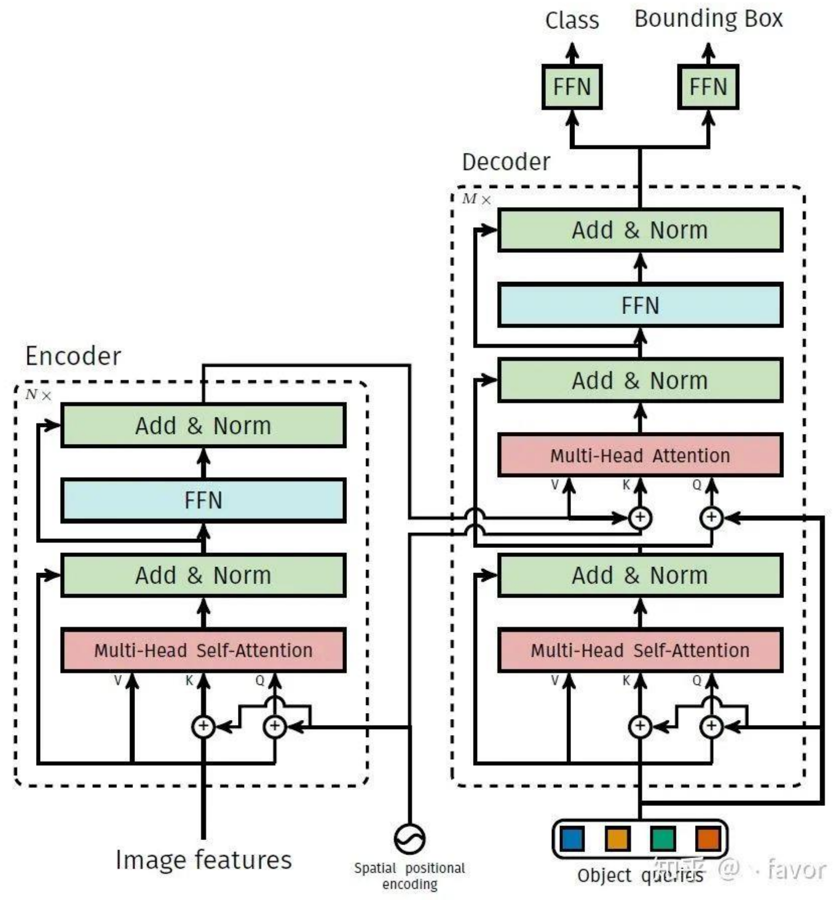
Approach

- Query based object detector

$$\begin{aligned}
\boldsymbol{x}_t^{\text{box}} &\leftarrow \mathcal{P}^{\text{box}}\left(\boldsymbol{x}^{\text{FPN}}, \boldsymbol{b}_{t-1}\right), \\
\boldsymbol{q}_{t-1}^{*} &\leftarrow \text{MSA}_t\left(\boldsymbol{q}_{t-1}\right), \\
\boldsymbol{x}_t^{\text{box}*}, \boldsymbol{q}_t &\leftarrow \text{DynConv}_t^{\text{box}}\left(\boldsymbol{x}_t^{\text{box}}, \boldsymbol{q}_{t-1}^{*}\right), \\
\boldsymbol{b}_t &\leftarrow \mathcal{B}_t\left(\boldsymbol{x}_t^{\text{box}*}\right),
\end{aligned}$$

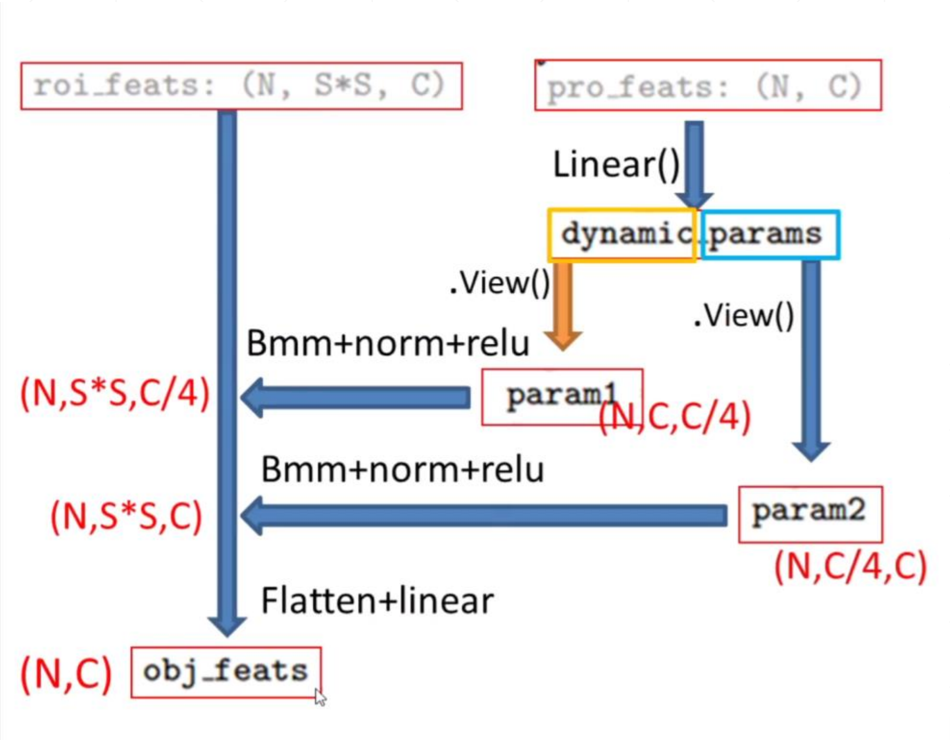


(a) Sparse R-CNN

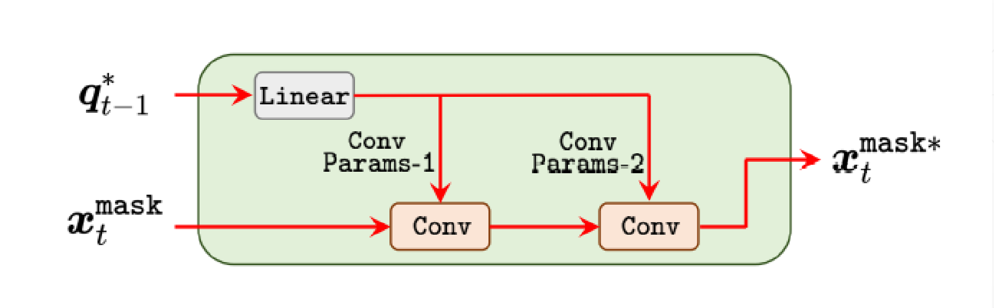


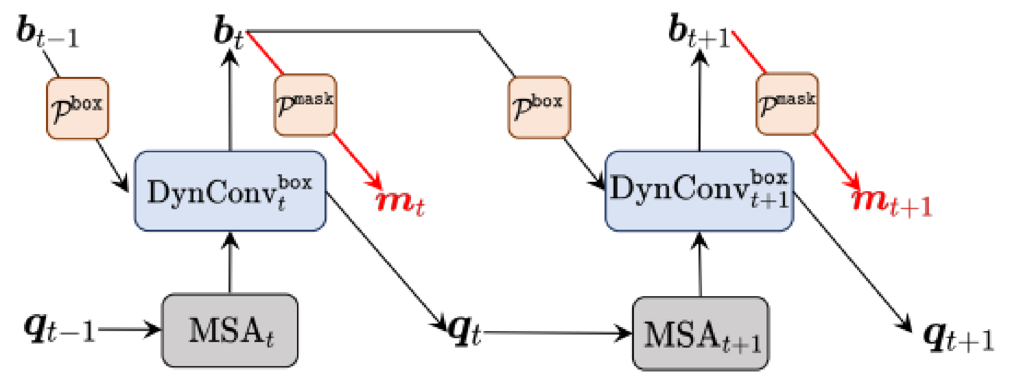
Ap

- V
- D

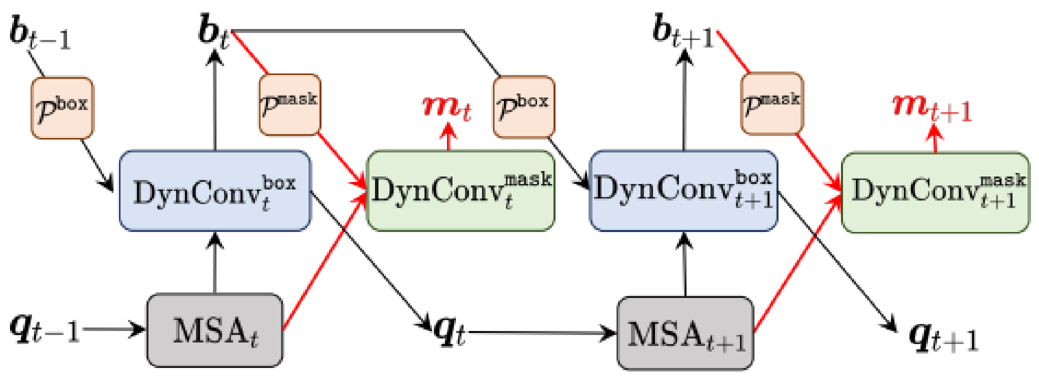


$$m_t \leftarrow \mathcal{M}_t\left(x_t^{\text{mask}*}\right).$$





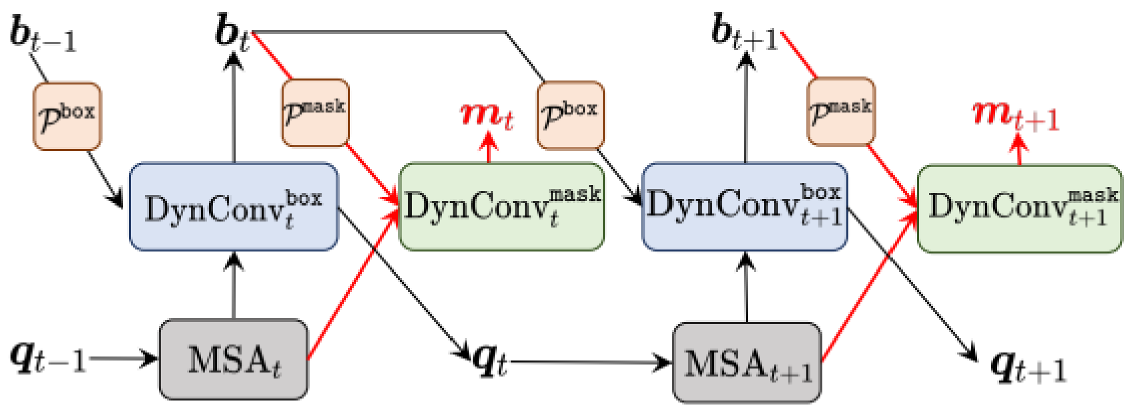
(b) Sparse R-CNN with vanilla mask head



(c) QueryInst with dynamic mask head

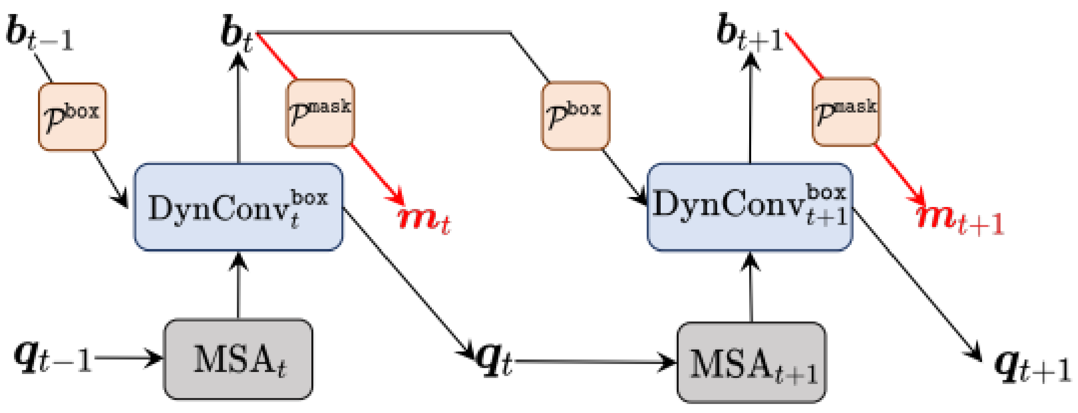
Approach

- During training



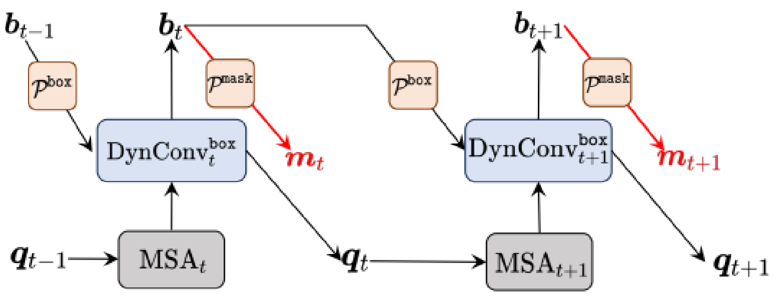
(c) QueryInst with dynamic mask head

- During inference



(b) Sparse R-CNN with vanilla mask head

Comparisons with Cascade Ma



(b) Sparse R-CNN with vanilla mask head

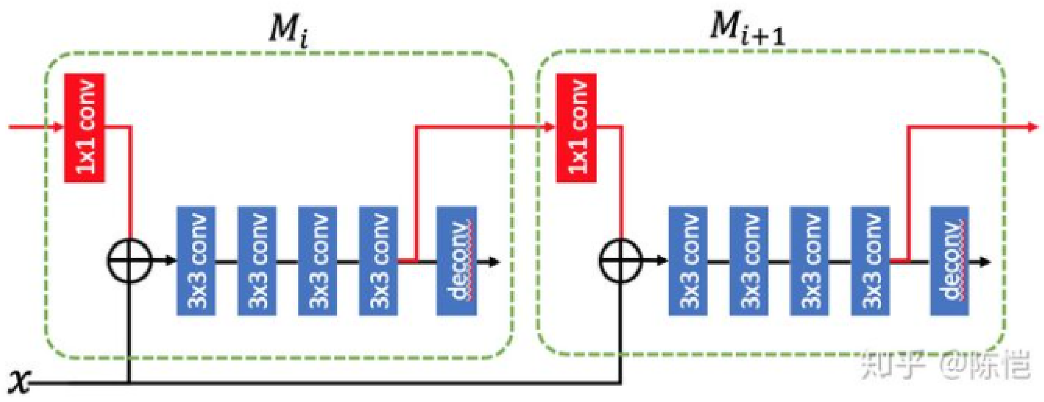
Type	Cascade Mask Head [5]	HTC Mask Flow [9]	DynConv$^{\text{mask}}$	Fig.	AP
Non-query Based	✓				4
		✓			4
Query Based	✓			Fig. 2 (b)	4
		✓			4
			✓	Fig. 2 (c)	4
		✓	✓		4

Table 4: Impacts of different mask head architectures on different frameworks. The

进阶第二步：Mask Information Flow

这一步起到了很重要的作用，对一般 cascade 结构的设计和改进也具有借鉴意义。我们首先回顾原始 Cascade R-CNN 的结构，每个 stage 只有 box 分支。当前 stage 对下一 stage 产生影响的途径有两条：（1）B_{i+1} 的输入特征是 B_i 预测出回归后的框通 RoI Align 获得的；（2）B_{i+1} 的回归目标是依赖 B_i 的框的预测的。这就是 box 分支的信息流，让下一个 stage 的特征和学习目标和当前 stage 有关。在 cascade 的结构中这种信息流是很重要的，让不同 stage 之间在逐渐调整而不是类似于一种 ensemble。

然而在 Cascade Mask R-CNN 中，不同 stage 之间的 mask 分支是没有任何直接的信息流的，M_{i+1} 只和当前 B_i 通过 RoI Align 有关联而与 M_i 没有任何联系。多个 stage 的 mask 分支更像用不同分布的数据进行训练然后在测试的时候进行 ensemble，而没有起到 stage 间逐渐调整和增强的作用。为了解决这一问题，我们在相邻的 stage 的 mask 分支之间增加一条连接，提供 mask 分支的信息流，让 M_{i+1} 能知道 M_i 的特征。具体实现上如下图中红色部分所示，我们将 M_i 的特征经过一个 1x1 的卷积做 feature embedding，然后输入到 M_{i+1} ，这样 M_{i+1} 既能得到 backbone 的特征，也能得到上一个 stage 的特征。



Experiments

Method	Backbone	Aug.	Epochs	AP^{box}	AP	AP_{50}	AP_{75}	AP_S	AP_M	AP_L	FPS
Mask R-CNN [21]	ResNet-50-FPN	640 ∼ 800	36	41.3	37.5	59.3	40.2	21.1	39.6	48.3	14.0
CondInst w/ sem. [46]				–	38.6	60.2	41.4	20.6	41.0	51.1	14.1
SOLOv2 [51]				40.4	38.8	59.9	41.7	16.5	**41.7**	**56.2**	13.8
QueryInst (5 Stage, 100 Queries)				**44.5**	**39.9**	**62.2**	**43.0**	**22.9**	**41.7**	51.9	13.5
Cascade Mask R-CNN [5]	ResNet-50-FPN	640 ∼ 800	36	44.5	38.6	60.0	41.7	21.7	40.8	49.6	10.4
HTC [9]				44.9	39.7	61.4	43.1	22.6	42.2	50.6	3.1
QueryInst (100 Queries)				44.8	40.1	62.3	43.4	23.3	42.1	52.0	10.5
QueryInst (300 Queries)				**45.6**	**40.6**	**63.0**	**44.0**	**23.4**	**42.5**	**52.8**	7.0
Cascade Mask R-CNN	ResNet-101-FPN	640 ∼ 800	36	45.7	39.8	61.6	43.0	22.4	42.2	50.8	8.7
HTC				46.2	40.7	62.7	44.2	23.1	43.4	52.7	2.5
QueryInst (300 Queries)				**47.0**	**41.7**	**64.4**	**45.3**	**24.2**	**43.9**	**53.9**	6.1
Cascade Mask R-CNN	ResNet-101-FPN	480 ∼ 800 w/ crop	36	46.2	40.0	61.7	43.5	22.5	42.5	51.2	8.7
HTC				46.3	40.8	62.6	44.3	23.0	43.5	52.6	2.5
Sparse R-CNN (300 Queries)				46.3	–	–	–	–	–	–	6.9
QueryInst (300 Queries)				**48.1**	**42.8**	**65.6**	**46.7**	**24.6**	**45.0**	**55.5**	6.1
QueryInst (300 Queries)	ResNeXt-101-FPN w/ DCN	480 ∼ 800 w/ crop	36	**50.4**	**44.6**	**68.1**	**48.7**	**26.6**	**46.9**	**57.7**	3.1
QueryInst (300 Queries) @ `val`	Swin-L	400 ∼ 1200 w/ crop	50	**56.1**	**48.9**	**74.0**	**53.9**	**30.8**	**52.6**	**68.3**	$3.3^{\top}$
QueryInst (300 Queries)	Swin-L	400 ∼ 1200 w/ crop	50	**56.1**	**49.1**	**74.2**	**53.8**	**31.5**	**51.8**	**63.2**	$3.3^{\top}$

Experiments

Method	Backbone	AP_{val}	AP	AP_{50}	person	rider	car	trunk	bus	train	mcycle	bicycle
Mask R-CNN [21]	ResNet-50	36.4	32.0	58.1	34.8	27.0	49.1	30.1	40.9	30.9	24.1	18.7
BShapeNet+ [26]	ResNet-50	–	32.9	58.8	36.6	24.8	50.4	33.7	41.0	33.7	25.4	17.8
UPSNet [56]	ResNet-50	37.8	33.0	**59.7**	35.9	27.4	51.9	31.8	43.1	31.4	23.8	19.1
CondInst [46]	ResNet-50	37.5	33.2	57.2	35.1	27.7	54.5	29.5	42.3	**33.8**	23.9	18.9
CondInst [46] w/ sem.	DCN-101-BiFPN	39.3	33.9	58.2	35.6	28.1	55.0	**32.1**	**44.2**	33.6	24.5	18.6
QueryInst	ResNet-50	**39.4**	**34.4**	59.6	**40.4**	**30.7**	**56.8**	29.1	40.5	30.8	**26.0**	**21.1**

Table 2: Instance segmentation results on Cityscapes `val` (AP_{val} column) and `test` (remain columns) split. The best results are in **bold**.

Method	Backbone	AP	AP_{50}	AP_{75}	AR_1	AR_{10}	FPS
MaskTrack R-CNN [57]	ResNet-50	30.3	51.1	32.6	31.0	35.5	22.1
SipMask-VIS [6]	ResNet-50	32.5	53.0	33.3	33.5	38.9	30.9
SipMask-VIS*	ResNet-50	33.7	54.1	35.8	35.4	40.1	30.9
STEm-Seg [1]	ResNet-50	30.6	50.7	33.5	31.6	37.1	4.4
STEm-Seg	ResNet-101	34.6	55.8	37.9	34.4	41.6	2.1
CompFeat [16]	ResNet-50	35.3	56.0	38.6	33.1	40.3	–
VisTR [53]	ResNet-50	34.4	55.7	36.5	33.5	38.9	30.0
VisTR	ResNet-101	35.3	**57.0**	36.2	34.3	40.4	27.7
QueryInst-VIS	ResNet-50	34.6	55.8	36.5	35.4	42.4	**32.3**
QueryInst-VIS*	ResNet-50	**36.2**	56.7	**39.7**	**36.1**	**42.9**	**32.3**

Table 3: Comparisons with state-of-the-art video instance segmentation methods on YouTube-VIS `val` set. Methods with superscript "$*$" indicates using multi-scale data argumentation during training. The best results are in **bold**.

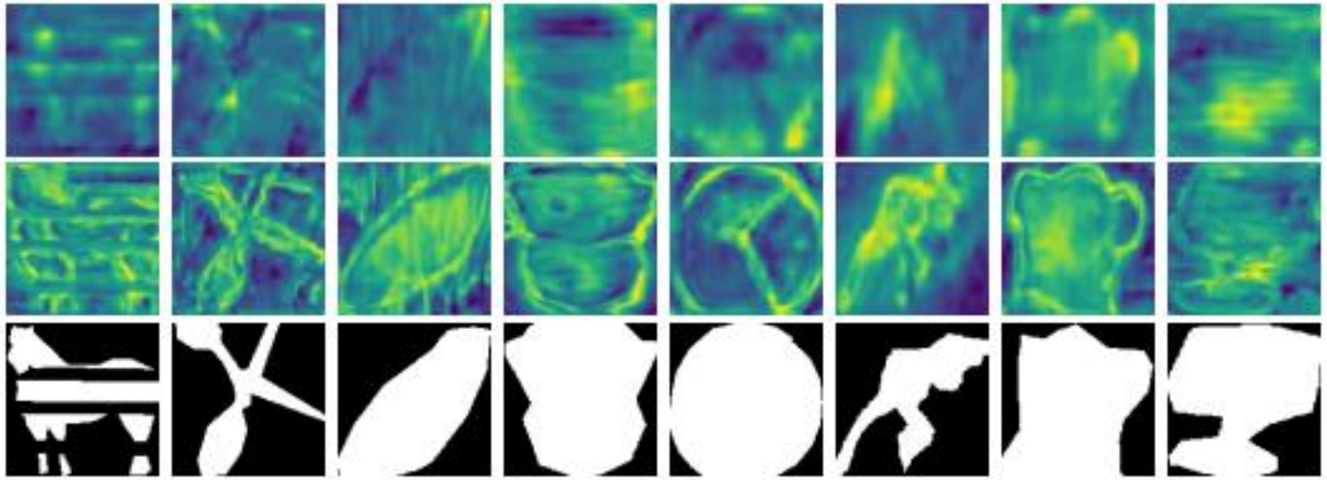


Figure 4: Effects of DynConv$^{\mathtt{mask}}$. The first row shows mask features $\boldsymbol{x}^{\mathtt{mask}}$ directly extracted from FPN. The second row shows mask features $\boldsymbol{x}^{\mathtt{mask}*}$ enhanced by queries in DynConv$^{\mathtt{mask}}$. Last row is ground-truth instance masks. The results show that mask features enhanced by queries yield more genuine and accurate details and carry more information of instances.

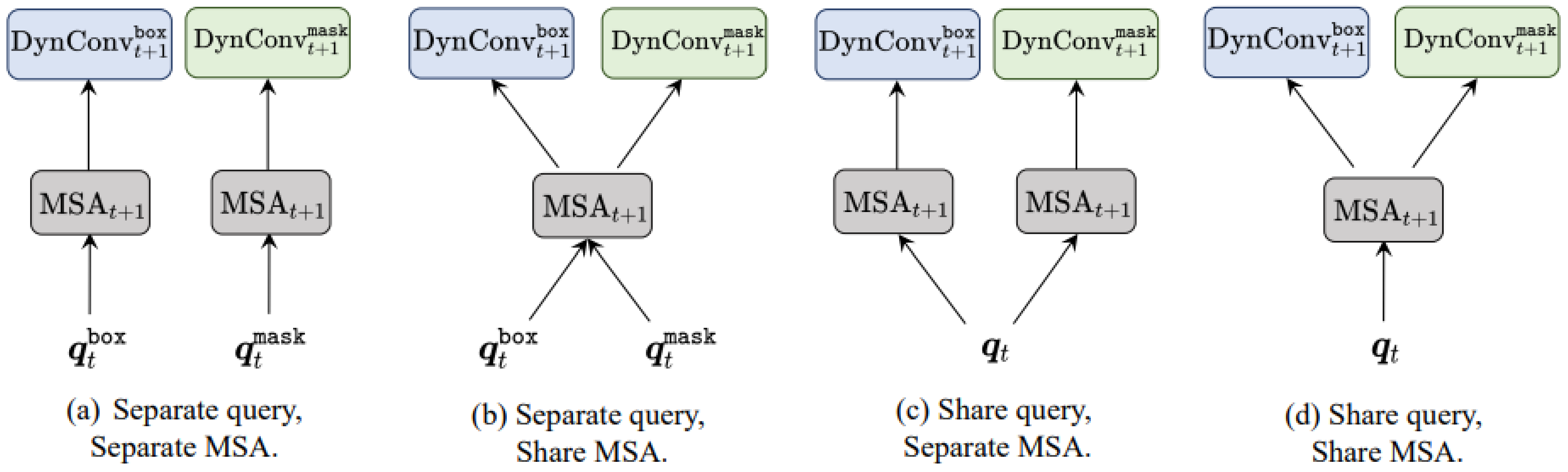


Figure 5: Illustration of 4 different query and MSA configurations. We use (d) as the default instantiation of QueryInst.

Shared MSA	Shared Query	AP^{box}	Δ^{box}	AP^{mask}	Δ^{mask}
		43.4		38.1	
✓		43.9	+ 0.5	38.3	+ 0.2
	✓	44.1	+ 0.7	39.5	+ 1.4
✓	✓	44.5	+ 1.1	39.8	+ 1.7

Table 6: Impacts of using shared query and MSA.

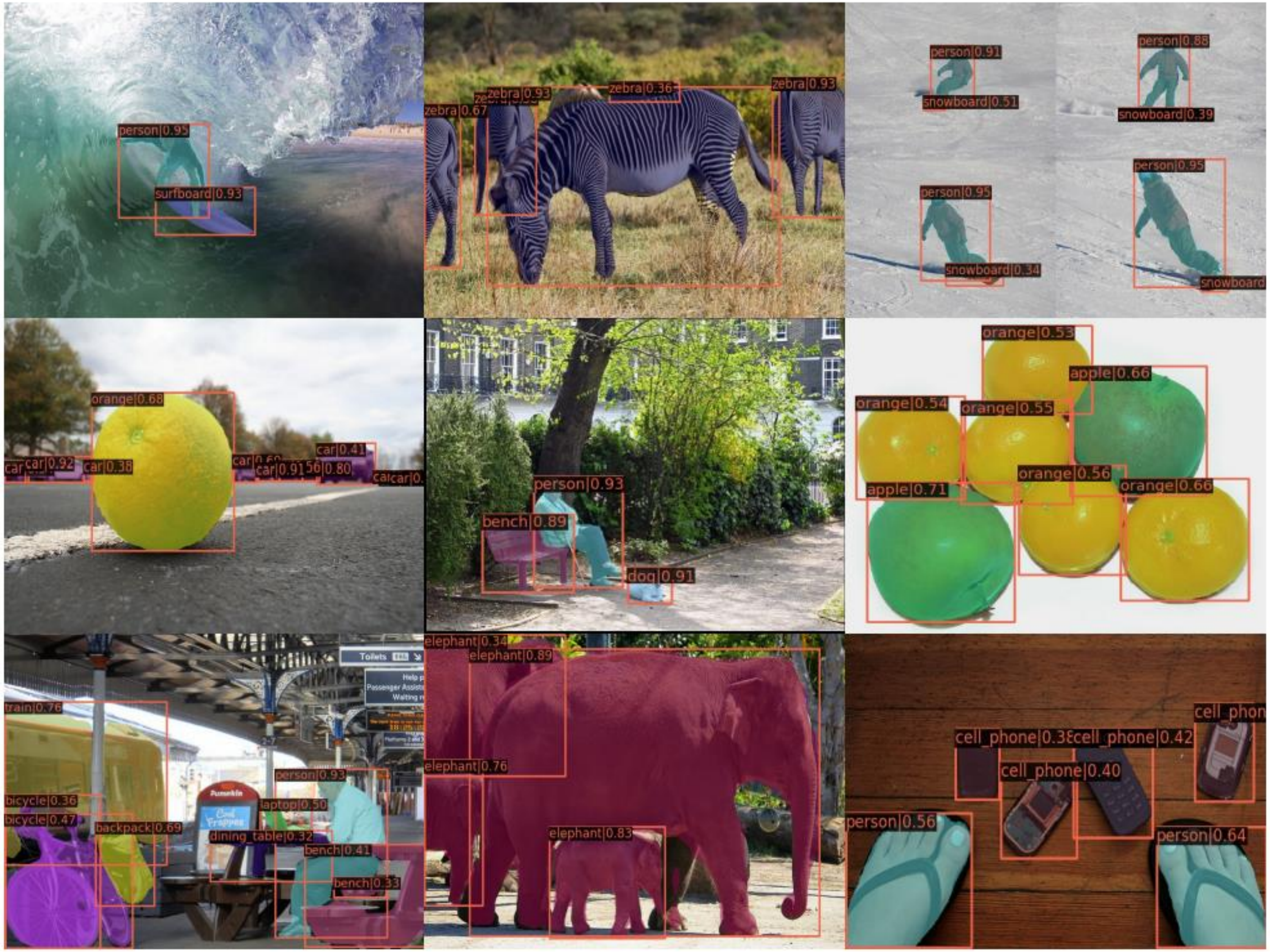


Figure 6: Object detection and instance segmentation qualitative results on COCO `val` split.

Figure 7: Object detection and instance segmentation qualitative results on Cityscapes `test` split.

Thanks