

# Instances as Queries

Yuxin Fang<sup>1\*</sup>, Shusheng Yang<sup>1,2\*</sup>, Xinggang Wang<sup>1†</sup>, Yu Li<sup>2</sup>, Chen Fang<sup>3</sup>, Ying Shan<sup>2</sup>, Bin Feng<sup>1</sup>, Wenyu Liu<sup>1</sup>

<sup>1</sup>School of EIC, Huazhong University of Science & Technology

<sup>2</sup>Applied Research Center (ARC), Tencent PCG <sup>3</sup>Tencent

## Abstract

Recently, query based object detection frameworks achieve comparable performance with previous state-of-the-art object detectors. However, how to fully leverage such frameworks to perform instance segmentation remains an open problem. In this paper, we present *QueryInst* (*Instances as Queries*), a query based instance segmentation method driven by parallel supervision on dynamic mask heads. The key insight of *QueryInst* is to leverage the intrinsic one-to-one correspondence in object queries across different stages, as well as one-to-one correspondence between mask *RoI* features and object queries in the same stage. This approach eliminates the explicit multi-stage mask head connection and the proposal distribution inconsistency issues inherent in non-query based multi-stage instance segmentation methods. We conduct extensive experiments on three challenging benchmarks, i.e., COCO, CityScapes, and YouTube-VIS to evaluate the effectiveness of *QueryInst* in instance segmentation and video instance segmentation (VIS) task. Specifically, using ResNet-101-FPN backbone, *QueryInst* obtains 48.1 box AP and 42.8 mask AP on COCO test-dev, which is 2 points higher than HTC in terms of both box AP and mask AP, while runs 2.4 times faster. For video instance segmentation, *QueryInst* achieves the best performance among all online VIS approaches and strikes a decent speed-accuracy trade-off. Code is available at <https://github.com/hustvl/QueryInst>.

## 1. Introduction

Instance segmentation is a fundamental yet challenging computer vision task that requires an algorithm to assign a pixel-level mask with a category label for each instance of interest in image. Prevalent state-of-the-art instance seg-

\*Equal contributions. This work was done while Shusheng Yang was interning at Applied Research Center (ARC), Tencent PCG.

†Corresponding author, E-mail: xgwang@hust.edu.cn.

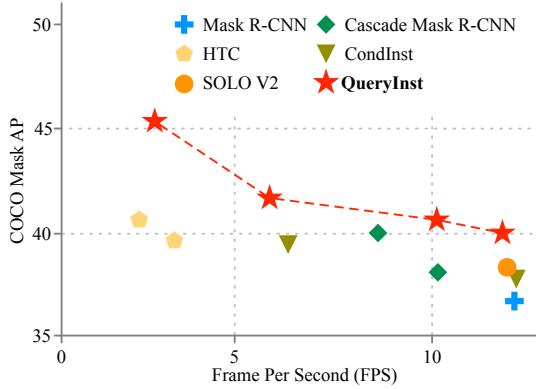


Figure 1: AP vs. FPS on COCO test-dev. *QueryInst* outperforms current state-of-the-art methods in terms of both accuracy and speed. The speed is measured using a single Titan Xp GPU.

mentation methods are based on high performing object detectors and follow a multi-stage paradigm. Among which, the Mask R-CNN family [21, 24, 30, 5, 9, 42] is the most successful one, where the regions-of-interest (RoI) for instance segmentation is extracted via a region-wise pooling operation (e.g., RoIPool [22, 18] or RoIAAlign [21]) based on the box-level localization information from the region proposal network (RPN) [39], or the previous stage bounding-box prediction [4, 5]. The final instance mask is obtained via feeding the RoI feature into the mask head, which is a small fully convolutional network (FCN) [33].

Recently, DETR [7] is proposed to reformulate object detection as a query based direct set prediction problem, whose input is mere 100 learned object queries. Follow-up works [62, 42, 43, 17, 58, 14] in object detection improve this query based approach and achieve comparable performance with state-of-the-art detectors such as Cascade R-CNN [4]. The results show that query based instance-level perception is a very promising research direction. Thus, enabling query based detection framework to perform instance segmentation is highly desirable. However, we find

that it is inefficient to integrate the previous successful practices in Cascade Mask R-CNN [5] and HTC [9], which are state-of-the-art mask generation solutions in the non-query based paradigm, directly into query based detectors for instance mask generation. Therefore, an instance segmentation method tailored for the query based end-to-end framework is urgently needed.

To bridge this gap, we propose **QueryInst** (Instances as Queries), a query based end-to-end instance segmentation method driven by parallel supervision on dynamic mask heads [25, 44, 42]. The key insight of QueryInst is to leverage the *intrinsic one-to-one correspondence in object queries* across different stages, and *one-to-one correspondence between mask RoI features and object queries* in the same stage. Specifically, we set up dynamic mask heads in parallel with each other, which transform each mask RoI feature adaptively according to the corresponding query, and are simultaneously trained in all stages. The mask gradient not only flows back to the backbone feature extractor, but also to the object query, which is intrinsically one-to-one interlinked in different stages. The queries implicitly carry the multi-stage mask information, which is read by RoI features in dynamic mask heads for final mask generation. There is no explicit connection between different stage mask heads or mask features. Moreover, the queries are shared between object detection and instance segmentation sub-networks in each stage, enabling cross-task communications that one task can take advantage of the information from the other task. We demonstrate that this shared query design can fully leverage the synergy between object detection and instance segmentation. When the training is completed, we throw away all the dynamic mask heads in the intermediate stages and only use the final stage predictions for inference. Under such a scheme, QueryInst surpasses the state-of-the-art HTC in terms of AP while runs much faster. Concretely, our main contributions are summarized as follows:

- 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. This novel solution enables such a new framework to outperform well-established and highly-optimized non-query based multi-stage schemes such as Cascade Mask R-CNN and HTC in terms of both accuracy and speed (see Fig. 1). Specifically, using ResNet-101-FPN backbone [23, 27], QueryInst obtains 48.1 AP<sup>box</sup> and 42.8 AP<sup>mask</sup> on COCO test-dev, which is 2 point higher than HTC in terms of both box AP and mask AP, while runs 2.4× faster. Without bells and whistles, our best model achieves 50.4 AP<sup>box</sup> and 46.6 AP<sup>mask</sup> on COCO test-dev.
- We set up a task-joint paradigm for query based ob-

ject detection and instance segmentation by leveraging the shared query and multi-head self-attention design. This paradigm establishes a kind of communication and synergy between detection and segmentation tasks, which encourages this two tasks to benefits from each other. We demonstrate that our architecture design can also significantly improve the object detection performance.

- We extend the QueryInst to video instance segmentation task (VIS) [57] task by simply adding a vanilla track head. Experiments on YouTube-VIS dataset [57] indicate that with same tracking approach, our methods outperforms MaskTrack R-CNN [57] and SipMask-VIS [6] by a large margin. QueryInst-VIS can even outperform well-designed VIS approaches such as STEm-Seg [1] and VisTR [53].

## 2. Related Work

**Query Based Methods.** Recently, query based methods emerged to tackle the set-prediction problems. Concretely, DETR [7] first introduces the query based methods with transformer architecture to object detection. Deformable DETR [62], UP-DETR [14], ACT [58] and TSP [43] improve the performance on the top of DETR. The recently proposed Sparse R-CNN [42] builds a query based set-prediction framework upon R-CNN [19, 18, 39] based detector. For segmentation, VisTR [53] introduces a query based sequence matching and segmentation method to video instance segmentation, building a fully end-to-end framework for instance segmentation in video. MaxDeepLab[49] presents the first box-free end-to-end panoptic segmentation model with a global memory as external query. Trackformer [35] and Transtrack [41] build a query based multiple object tracktor upon DETR and Deformable DETR, respectively, and attain comparable results to the non-query based methods. AS-Net [11] introduces a query based set-prediction pipeline to human object interaction and obtains promising results. Despite query based set-prediction method is being widely used to many computer vision tasks, few efforts are conducted to build a successful query based instance segmentation framework. We aim to achieve this goal in this paper.

**Object Detection.** Object detection is a fundamental computer vision task which aims to detect visual objects with bounding boxes. With the propose of R-CNN [19], Fast R-CNN [18] and Faster R-CNN [39], anchor based methods [4, 38, 34, 28, 31] dominate object detection for a long period. CenterNet [60] and FCOS [45] establish anchor-free detectors with competitive detection performance. Recently, with the proposed DETR [7], query based set-prediction methods catch lots of attentions. Deformable

DETR [62] introduces deformable convolution [61] to the DETR framework, achieving better performance with faster training convergence. UP-DETR [14] extends DETR to unsupervised scenarios. ACT [58] and TSP [43] introduce the adaptive clustering module and a new bipartite matching method to DETR. Sparse R-CNN [42] build a query based detector on top of R-CNN architecture, while OneNet [40] and DeFCN [50] are end-to-end detector built upon the one-stage FCOS [45]. In this work, we present a query based instance segmentation method on the top of the query based Sparse R-CNN detector.

**Instance Segmentation.** Instance segmentation is a fundamental yet challenging computer vision task that requires an algorithm to assign a pixel-level mask with a category label for each instance of interest in image. Mask R-CNN [21] introduces a fully convolutional mask head to Faster R-CNN [18] detector. Casacde Mask R-CNN [5] simply combine the Casacde R-CNN [4] with Mask R-CNN. HTC [9] presents interleaved execution and mask information flow and achieves state-of-the-art performance. In addition to R-CNN based methods, YOLACT [3, 2], Sip-Mask [6], CondInst [46] and SOLO [51, 52] build one-stage instance segmentation framework on the top of one-stage framework, achieving comparable results with favorable inference speed. Following the R-CNN based methods, we present a query based instance segmentation framework.

### 3. Instances as Queries

We propose QueryInst (Instances as Queries), a query based end-to-end instance segmentation method. QueryInst consists of a query based object detector and six dynamic mask heads driven by parallel supervision. Our key insight is to leverage the *intrinsic one-to-one correspondence in queries* across different stages. This correspondence exists in all query based framework [47, 15, 37, 8, 7] regardless of the specific instantiations and applications. The overall architecture of QueryInst is illustrated in Fig. 2 (c).

#### 3.1. Query based Object Detector

QueryInst can be built on any multi-stage query based object detector [7, 62, 42]. We choose Sparse R-CNN [42] as our default instantiation, which has six query stages. The object detection pipeline is depicted in Fig. 2 (a) and can be formulated as follows:

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

where  $\mathbf{q} \in \mathbf{R}^{N \times d}$  denotes the object query.  $N$  and  $d$  denote the length (number) and dimension of query  $\mathbf{q}$ . At

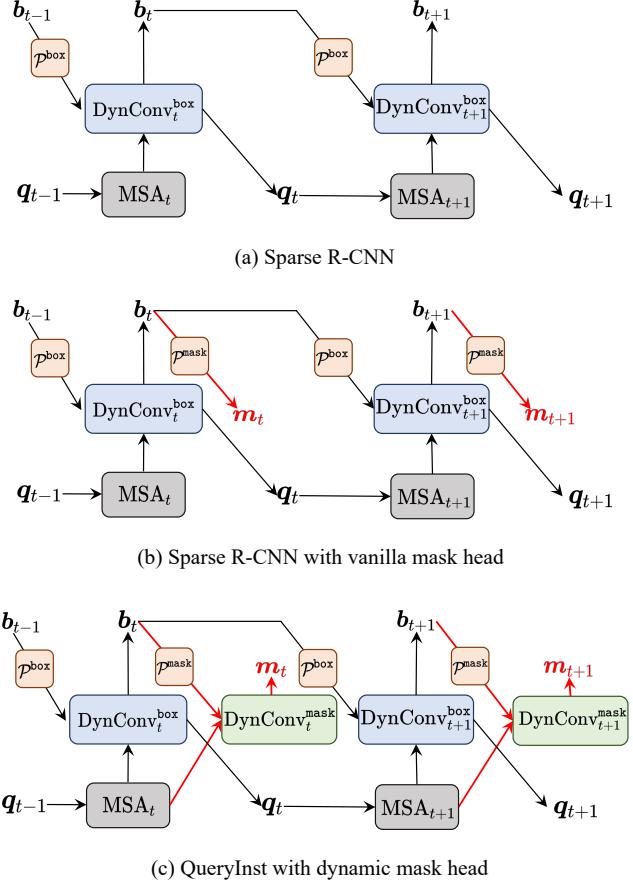


Figure 2: Overview of QueryInst. The red arrows indicate mask branches. Please note that QueryInst consists of 6 stage in parallel, i.e.,  $t = \{1, 2, 3, 4, 5, 6\}$ . The figure only shows 2 stages.

stage  $t$ , a pooling operator  $\mathcal{P}^{\text{box}}$  extracts the current stage bounding box features  $\mathbf{x}_t^{\text{box}}$  from FPN [27] features  $\mathbf{x}^{\text{FPN}}$  under the guidance of previous stage bounding box predictions  $\mathbf{b}_{t-1}$ . Meanwhile, a multi-head self-attention module MSA $_t$  is applied to the input query  $\mathbf{q}_{t-1}$  to get the transformed query  $\mathbf{q}_{t-1}^*$ . Then, a box dynamic convolution module DynConv $_t^{\text{box}}$  takes  $\mathbf{x}_t^{\text{box}}$  and  $\mathbf{q}_{t-1}^*$  as inputs and enhances the  $\mathbf{x}_t^{\text{box}}$  by reading  $\mathbf{q}_{t-1}^*$  while generating  $\mathbf{q}_t$  for the next stage. Finally, the enhanced bounding box features  $\mathbf{x}_t^{\text{box}*}$  are fed into the box prediction branch  $\mathcal{B}_t$  for current bounding box prediction  $\mathbf{b}_t$ .

### 3.2. Mask Head Architecture

#### 3.2.1 Vanilla Mask Head

For instance mask prediction, we first adopt the widely used vanilla mask head architecture design in Mask R-CNN [21] as our instance segmentation baseline. The model architecture is depicted in Fig. 2 (b). Based on the object detection

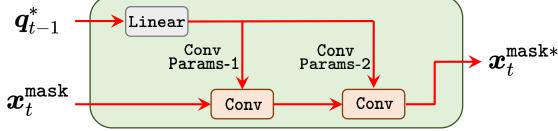


Figure 3: Illustrations of  $\text{DynConv}_t^{\text{mask}}$  at stage  $t$ .  $x_t^{\text{mask}*}$  is enhanced by two consecutive conv-layers, whose kernel parameters are produced by  $q_{t-1}^*$ .

pipeline described in Sec. 3.1, the mask generation process can be expressed as follows:

$$\begin{aligned} x_t^{\text{mask}} &\leftarrow \mathcal{P}^{\text{mask}}(x^{\text{FPN}}, b_t), \\ m_t &\leftarrow \mathcal{M}_t(x_t^{\text{mask}}), \end{aligned} \quad (2)$$

where  $b_t$  is the bounding box predictions from the object detector.  $\mathcal{P}^{\text{mask}}$  denotes a region-wise pooling operator for mask ROI features extraction.  $\mathcal{M}_t$  indicates the mask FCN head consisting of a stack of four consecutive conv-layers, one dconv-layer and one  $1 \times 1$  conv-layer for mask generation [21].  $m_t$  is the current stage mask predictions.

Overall, this vanilla design is an analogy of Cascade Mask R-CNN[5] in a query based framework. However, we find that this design is not as effective as the original Cascade Mask R-CNN. Moreover, establishing explicit mask flow following HTC [9] on top of this design (Fig. 2 (b)) can only bring moderate improvements at a cost of large drops in both training and inference speed. Part of the reasons may be the number of queries in our framework is much smaller than the number of proposals in Cascade Mask R-CNN and HTC, resulting in limited availability of training samples.

### 3.2.2 Dynamic Mask Head

Our goal is to design a mask prediction head tailor-made for query based instance segmentation frameworks. To this end, we propose to leverage dynamic mask heads driven by parallel supervision to replace the vanilla design in Sec. 3.2.1. The dynamic mask head at stage  $t$  consists of a dynamic mask convolution module  $\text{DynConv}_t^{\text{mask}}$  (see Fig. 3) [42] following by a vanilla mask head  $\mathcal{M}_t$  [21]. The mask generation pipeline is reformulated as follows:

$$\begin{aligned} x_t^{\text{mask}} &\leftarrow \mathcal{P}^{\text{mask}}(x^{\text{FPN}}, b_t), \\ x_t^{\text{mask}*} &\leftarrow \text{DynConv}_t^{\text{mask}}(x_t^{\text{mask}}, q_{t-1}^*), \\ m_t &\leftarrow \mathcal{M}_t(x_t^{\text{mask}*}). \end{aligned} \quad (3)$$

It is noteworthy that the only difference between the proposed dynamic mask head and vanilla mask head is the existence of  $\text{DynConv}_t^{\text{mask}}$ . We demonstrate that  $\text{DynConv}_t^{\text{mask}}$  enables (1) per-mask information flow in queries driven by parallel mask branch supervision, and (2) communication

and synergy for joint detection and instance segmentation in the following two sub-sections, respectively. The effectiveness of these two properties is verified in our experiments.

### 3.3. Per-mask Information Flow with Parallel Supervision

In query based models such as [7, 62, 42], the model learns different specialization for each query slot [7], *i.e.*,  $q_t[s]$  is the transformed and refined version of previous stage  $q_{t-1}[s]$  in the same  $s$ -th slot. Moreover,  $x_t^{\text{mask}}[s]$  corresponds to and is refined by  $q_t[s]$  [42]. Therefore there is an one-to-one correspondence across different stage queries inherent in these frameworks, as well as one-to-one correspondence between mask ROI features and object queries in the same stage.

QueryInst is driven by parallel supervision on dynamic mask heads, which fully leverages the intrinsic one-to-one correspondence in object queries across different stages. Specifically, we set up dynamic mask heads in parallel with each other, which transform each mask ROI feature  $x_t^{\text{mask}}$  adaptively in  $\text{DynConv}_t^{\text{mask}}$  according to the corresponding query  $q_{t-1}^*$ , and are simultaneously trained in all stages. Inside  $\text{DynConv}_t^{\text{mask}}$ , the query acts as memory and is read by mask ROI features  $x_t^{\text{mask}}$  in the forward pass and written by  $x_t^{\text{mask}}$  in the backward pass.

During training, the per-mask information (*i.e.*, the mask gradient) not only flows back to mask ROI features  $x_t^{\text{mask}}$ , but also to the object query  $q_{t-1}^*$ , which is intrinsically one-to-one interlinked in different stages. Therefore the per-mask information flow is naturally established by leveraging the inherent properties of query based frameworks, with no additional connection needed. After the training is completed, the information for mask prediction is stored in queries.

During inference, we throw away all dynamic mask heads in 5 intermediate stages and only use the final stage predictions for inference. The queries implicitly carry the multi-stage information for mask prediction, which is read by mask ROI features  $x_t^{\text{mask}}$  in dynamic mask convolution  $\text{DynConv}_t^{\text{mask}}$  at the last stage for final mask generation.

Without  $\text{DynConv}_t^{\text{mask}}$ , the link between mask ROI features and the query is lost, and mask heads in different stages are isolated. Even though parallel supervision is applied to all mask heads, the information related to mask generation cannot flow into queries. In this condition, QueryInst degenerates to Cascade Mask R-CNN with a fixed number (*i.e.*,  $N$ ) of proposals across all stages.

### 3.4. Shared Query and MSA for Joint Detection and Segmentation

At stage  $t$ , a multi-head self-attention  $\text{MSA}_t$  is applied to the query  $q_{t-1}$ .  $\text{MSA}_t$  projects the query  $q_{t-1}$  to a high dimensional embedding space and its output  $q_{t-1}^*$  is read

by the dynamic box convolution  $\text{DynConv}_t^{\text{box}}$  and dynamic mask convolution  $\text{DynConv}_t^{\text{mask}}$  respectively, to enhance the task-specific features  $x_t^{\text{box}}$  and  $x_t^{\text{mask}}$ .

Throughout this process, the query and MSA are shared between detection and instance segmentation tasks. Both detection and segmentation information flow back into the query through MSA. This task-joint paradigm establishes a kind of communication and synergy between detection and segmentation tasks, which encourages these two tasks to benefit from each other. The query learns a better instance-level representation with the guidance of two highly correlated tasks. We observe a performance decrease using separate queries or MSA in our experiments.

### 3.5. Comparisons with Cascade Mask R-CNN and HTC

Cascade Mask R-CNN [5] presents a multi-stage architecture to resample with higher intersection over union (IoU) thresholds for the latter stages, and progressively refine the training distributions of region proposals [4, 5]. This resampling operation guarantees the availability of a large number of accurate localized proposals for the final stage. Despite outperforming non-cascade counterparts, the effectiveness of Cascade Mask R-CNN mainly stems from the progressively refined proposal recall. Whereas, the mask heads across different stages are isolated, and the input feature for mask head always come from the same FPN feature regardless of the stage.

To mitigate the aforementioned drawback of Cascade Mask R-CNN, HTC [9] improves Cascade Mask R-CNN by introducing direct and explicit connections across the mask heads at different stages. The current stage mask features are combined with the accumulated mask features from all previous stages. Equivalently, the mask head of the final stage is  $3 \times$  deeper than the first stage. Therefore the final stage mask prediction can benefit from deeper features. Establishing direct mask information flow as HTC can alleviate the issue in Cascade Mask R-CNN to some extent. However, this explicit connection across mask heads at different R-CNN stages results in inefficient training and inference.

There are some potential issues inherent in the aforementioned non-query based instance segmentation paradigms. For Cascade Mask R-CNN and HTC, the quality of proposals in different stages is refined *in the statistical sense* [4, 5, 9]. For each stage, the number and distribution of training samples are quite different, and there is no explicit and intrinsic correspondence for each individual proposal across different stages [59]. Moreover, there is also a mismatch between the training and inference sample distribution [48]. Therefore, introducing direct connections at the architecture-level is necessary for the mask heads in different stages to explicitly learn the correspondence [9].

Our method does not directly solve the aforementioned issues, but bypasses them. For QueryInst, the connections across stages are naturally established by one-to-one correspondence inherent in queries. This approach eliminates the explicit multi-stage mask head connection and the proposal distribution inconsistency issues. We show that the proposed new paradigm can surpass Cascade Mask R-CNN and HTC in terms of both accuracy and speed.

### 3.6. QueryInst-VIS for Video Instance Segmentation

Video instance segmentation (VIS) [57] is a highly relevant task to still-image instance segmentation that aims at detecting, classifying, segmenting and tracking visual instances over video frames. We demonstrate that QueryInst can be easily extended to VIS with minimal modifications by simply adding the vanilla track head in Mask-Track R-CNN baseline [57]. The proposed model coined as QueryInst-VIS can perform video instance segmentation in an online manner while operating at real-time. The total training and inference pipeline keep the same as Mask-Track R-CNN. We evaluate QueryInst-VIS on the challenging YouTube-VIS [57] benchmark to demonstrate its effectiveness.

## 4. Experiments

### 4.1. Dataset

**COCO.** Most of our experiments are conducted on the challenging COCO dataset [29]. Following the common practice, we use the COCO train2017 split ( $115k$  images) for training and the val2017 split ( $5k$  images) as validation for our ablation study. We report our main results on the test-dev split ( $20k$  images).

**Cityscapes.** Cityscapes [12] is an ego-centric street-scene dataset with 8 categories, 2975 train images, and 500 validation images for instance segmentation. The images are with higher resolution ( $1024 \times 2048$  pixels) compared with COCO, and have more pixel-accurate ground-truth.

**YouTube-VIS.** In addition to static-image instance segmentation, we demonstrate the effectiveness of our QueryInst on video instance segmentation. YouTube-VIS [57] is a challenging dataset for video instance segmentation task, which has a 40-category label set, 4,883 unique video instances and  $131k$  high-quality manual annotations. There are 2,238 training videos, 302 validation videos, and 343 test videos.

### 4.2. Implementation Details

**Training Setup.** Our implementation is based on MMDection [10] and Detectron2 [54]. Following [42], the default training schedule is 36 epochs and the initial learning rate is set to  $2.5 \times 10^{-5}$ , divided by 10 at 27-th epoch and 33-th epoch, respectively. We adopt AdamW optimizer with

Method	Backbone	Aug.	Epochs	$\text{AP}^{\text{box}}$	AP	$\text{AP}_{50}$	$\text{AP}_{75}$	$\text{AP}_S$	$\text{AP}_M$	$\text{AP}_L$	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
<b>QueryInst</b> (5 Stage, 100 Queries)				<b>44.5</b>	<b>39.9</b>	<b>62.2</b>	<b>43.0</b>	<b>22.9</b>	<b>41.7</b>	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
<b>QueryInst</b> (100 Queries)				44.8	40.1	62.3	43.4	23.3	42.1	52.0	10.5
<b>QueryInst</b> (300 Queries)				<b>45.6</b>	<b>40.6</b>	<b>63.0</b>	<b>44.0</b>	<b>23.4</b>	<b>42.5</b>	<b>52.8</b>	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
<b>QueryInst</b> (300 Queries)				<b>47.0</b>	<b>41.7</b>	<b>64.4</b>	<b>45.3</b>	<b>24.2</b>	<b>43.9</b>	<b>53.9</b>	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
<b>QueryInst</b> (300 Queries)				<b>48.1</b>	<b>42.8</b>	<b>65.6</b>	<b>46.7</b>	<b>24.6</b>	<b>45.0</b>	<b>55.5</b>	6.1
<b>QueryInst</b> (300 Queries)	ResNeXt-101-FPN w/ DCN	480 ~ 800 w/ crop	36	<b>50.4</b>	<b>44.6</b>	<b>68.1</b>	<b>48.7</b>	<b>26.6</b>	<b>46.9</b>	<b>57.7</b>	3.1
<b>QueryInst</b> (300 Queries) @ val	Swin-L	400 ~ 1200 w/ crop	50	<b>56.1</b>	<b>48.9</b>	<b>74.0</b>	<b>53.9</b>	<b>30.8</b>	<b>52.6</b>	<b>68.3</b>	3.3 <sup>T</sup>
<b>QueryInst</b> (300 Queries)	Swin-L	400 ~ 1200 w/ crop	50	<b>56.1</b>	<b>49.1</b>	<b>74.2</b>	<b>53.8</b>	<b>31.5</b>	<b>51.8</b>	<b>63.2</b>	3.3 <sup>T</sup>

Table 1: Main results on COCO test-dev. The numbers under “Aug.” indicate the scale range of the shorter size of inputs with a stride of 32.  $\text{AP}^{\text{box}}$  denotes box AP. AP without superscript denotes mask AP. The best results are in **bold** for each configuration. Superscript “ $\top$ ” indicates the FPS data are measured on a single RTX 2080Ti GPU with batch size 1.

$1 \times 10^{-4}$  weight decay. Hyper-parameters, configurations as well as the label assignment procedures follow the setting in [7, 62, 42]. In total, the R-CNN head of QueryInst contains 6 stages in parallel as [42]. The mask head is trained by minimizing dice loss [36]. Without special mentioning, we adopt QueryInst model trained with 100 queries and ResNet-50-FPN [23, 27] as backbone in our experiments in the ablation studies.

**Inference.** Given an input image, QueryInst directly outputs top 100 bounding box predictions with their scores and corresponding instance masks without further post-processing. For inference, we use the final stage masks as the predictions and ignore all the parallel DynConv<sup>mask</sup> at the intermediate stages. The inference speed reported is measured using a single Titan Xp GPU with inputs resized to have their shorter side being 800 and their longer side less or equal to 1333.

### 4.3. Main Results

#### Comparisons on COCO Instance Segmentation.

The comparison of QueryInst with the state-of-the-art instance segmentation methods on COCO test-dev are listed in Tab. 1. We have tested different backbones and data augmentations. CondInst [46] (with auxiliary semantic branch) and SOLOv2 [52] are the latest state-of-the-art in-

stance segmentation approach based on dynamic convolutions. A 5-stage QueryInst trained with 100 queries outperforms them with over 1.1 mask AP gain under similar inference speed. QueryInst trained with 100 queries can also surpass Cascade Mask R-CNN [21] by 1.5 mask AP while runs with the same FPS. For fair comparisons with HTC [9], we train HTC using the 36 epochs training schedule and multi-scale data augmentations following the standard setting in [21, 54], yielding  $\sim 1$  higher mask AP than original results reported in [9]. Under same experimental conditions, QueryInst outperforms the state-of-the-art HTC in terms of both accuracy and speed. Moreover, QueryInst outperforms HTC in terms of AP at different IoU thresholds ( $\text{AP}_{50}$  and  $\text{AP}_{75}$ ) as well as AP at different scales ( $\text{AP}_S$ ,  $\text{AP}_M$  and  $\text{AP}_L$ ), regardless of the experimental configuration. We also find that compared with Cascade Mask R-CNN and HTC, the query based QueryInst can benefit more from stronger data argumentation used in [7, 62, 42]<sup>1</sup>. Specifically, using ResNet-101-FPN [27] backbone and stronger multi-scale data argumentation with random crop, QueryInst surpasses HTC by 2.0 mask AP and 1.8 box AP while runs 2.4 $\times$  faster. Further, QueryInst

<sup>1</sup>We experimentally study the effects of different training schedules and data augmentations to the Mask R-CNN family in the Appendix.

Method	Backbone	AP <sub>val</sub>	AP	AP <sub>50</sub>	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	<b>59.7</b>	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	<b>33.8</b>	23.9	18.9
CondInst [46] w/ sem.	DCN-101-BiFPN	39.3	33.9	58.2	35.6	28.1	55.0	<b>32.1</b>	<b>44.2</b>	33.6	24.5	18.6
<b>QueryInst</b>	ResNet-50	<b>39.4</b>	<b>34.4</b>	59.6	<b>40.4</b>	<b>30.7</b>	<b>56.8</b>	29.1	40.5	30.8	<b>26.0</b>	<b>21.1</b>

Table 2: Instance segmentation results on Cityscapes val (AP<sub>val</sub> column) and test (remain columns) split. The best results are in **bold**.

Method	Backbone	AP	AP <sub>50</sub>	AP <sub>75</sub>	AR <sub>1</sub>	AR <sub>10</sub>	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	<b>57.0</b>	36.2	34.3	40.4	27.7
<b>QueryInst-VIS</b>	ResNet-50	34.6	55.8	36.5	35.4	42.4	<b>32.3</b>
<b>QueryInst-VIS*</b>	ResNet-50	<b>36.2</b>	56.7	<b>39.7</b>	<b>36.1</b>	<b>42.9</b>	<b>32.3</b>

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**.

with deformable ResNeXt-101-FPN backbone [55, 13, 61] achieves 44.6 mask AP and 50.4 box AP without bells and whistles.

We demonstrate that the instance segmentation performance of QueryInst is not simply come from the accurate bounding box provided by Sparse R-CNN [42] object detector. On the contrary, QueryInst can largely improve the detection performance. The best result of Sparse R-CNN (ResNet-101-FPN, 300 queries, 480 ~ 800 w/ crop, 36 epochs) reported in [10] is 46.3 box AP. Under the same experimental setting, QueryInst can achieve 48.1 box AP, which outperform Sparse R-CNN by 1.8 box AP. We also show in the ablation study that QueryInst can outperform Cascade Mask R-CNN and HTC based on a weaker query based detector.

We also apply QueryInst to the recent state-of-the-art Swin Transformer [32] backbone without further modifications, and we find the proposed model is quite capable of adapting with Swin-L. Without bells and whistles, QueryInst can achieve the art performance in instance segmentation<sup>2</sup> as well as object detection<sup>3</sup>. For the first time, we demonstrate that an end-to-end query based framework

<sup>2</sup><https://paperswithcode.com/sota/instance-segmentation-on-coco>

<sup>3</sup><https://paperswithcode.com/sota/object-detection-on-coco>

driven by parallel supervision is competitive with well-established and highly-optimized methods in instance-level recognition tasks.

#### Comparisons on Cityscapes Instance Segmentation.

We also conduct experiments on Cityscapes dataset to demonstrate the generalization of QueryInst. Following the standard setting in [46, 21], all models are first pre-trained on COCO train2017 split then finetuned on Cityscapes using fine annotations for 24k iterations with batch size 8 (1 image per GPU). The initial learning rate is linearly scaled to  $1.25 \times 10^{-5}$  and is reduced by a factor of 10 at step 18k.

The results are shown in Tab. 2. QueryInst achieves 39.4 AP on val split and 34.4 AP on test split, surpassing several strong baselines. Notably, compared to the dynamic convolution based method CondInst [46], QueryInst with ResNet-50 backbone outperforms CondInst with both ResNet-101-DCN-BiFPN backbone and semantic branch. Overall, our QueryInst achieves leading results on Cityscapes dataset without bells and whistles.

#### Video Instance Segmentation Results on YouTube-VIS.

Tab. 3 shows the video instance segmentation results on YouTube-VIS val set. Following the standard setting in [57], we first pre-train the instance segmentation model on COCO train2017, then we finetune the corresponding

Type	Cascade Mask Head [5]	HTC Mask Flow [9]	DynConv <sup>mask</sup>	Fig.	AP <sub>box</sub>	$\Delta_{\text{box}}$	AP <sup>mask</sup>	$\Delta_{\text{mask}}$	FPS	$\Delta_{\text{FPS}}$
Non-query Based	✓	✓			44.3		38.5		10.4	
					44.4	+ 0.1	39.3	+ 0.8	3.1	- 7.3
Query Based	✓	✓	✓	Fig. 2 (b)	43.8		37.9		11.1	
				Fig. 2 (c)	43.8	+ 0.0	38.9	+ 1.0	6.0	- 5.1
		✓	✓		44.5	+ 0.7	39.8	+ 1.9	10.5	- 0.6
					44.4	+ 0.6	40.0	+ 2.1	5.4	- 5.7

Table 4: Impacts of different mask head architectures on different frameworks. The setting in blue is our default instantiation.

Parallel	DynConv <sup>mask</sup>	Fig.	AP <sub>box</sub>	AP <sup>mask</sup>	FPS
✓		Fig. 2 (b)	43.5	37.4	11.1
	✓	Fig. 2 (b)	43.8	37.9	11.1
✓	✓	Fig. 2 (c)	43.8	38.8	10.5
✓	✓	Fig. 2 (c)	44.5	39.8	10.5

Table 5: Impacts of parallel supervision and DynConv<sup>mask</sup>.

Shared MSA	Shared Query	AP <sub>box</sub>	$\Delta_{\text{box}}$	AP <sup>mask</sup>	$\Delta_{\text{mask}}$
✓		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.

VIS model on YouTube-VIS train set for 12 epochs. The maximum number of instances in one frame in YouTube-VIS dataset is 10, so we set the number of queries to 10 in QueryInst-VIS. The setting enables the model to operate at real-time ( $> 30$  FPS).

As mentioned in Sec. 3.6, QueryInst-VIS adopts the vanilla track method of MaskTrack R-CNN [57] and SipMask-VIS [6], while it obtains 4.3 AP improvement compared to MaskTrack R-CNN and 2.1 AP improvement compared to SipMask-VIS. Moreover, QueryInst can outperform many well-established and highly-optimized VIS approaches, such as STEm-Seg, CompFeat and VisTR in terms of both accuracy and speed.

#### 4.4. Ablation Study

##### Study of Parallel Supervision and DynConv<sup>mask</sup>.

We show that applying parallel mask head supervision and DynConv<sup>mask</sup> are both indispensable for good performance. As shown in Tab. 5, using parallel supervision on vanilla mask head cannot bring large improvement, because mask heads in different stages are isolated and there is no cross-stage per-mask information flow established (Sec. 3.2.1). Using DynConv<sup>mask</sup> without parallel super-

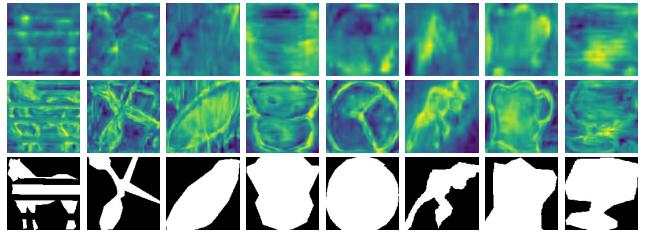


Figure 4: Effects of DynConv<sup>mask</sup>. The first row shows mask features  $x^{\text{mask}}$  directly extracted from FPN. The second row shows mask features  $x^{\text{mask}*}$  enhanced by queries in DynConv<sup>mask</sup>. Last row is ground-truth instance masks. The results show that mask features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

vision on each stage can only bring moderate improvement, for mask gradients injected from the final stage cannot fully driven the per-mask information flow across queries in all stages. When DynConv<sup>mask</sup> in all stages are driven by parallel supervision simultaneously, QueryInst achieves significant improvement in accuracy with only little drops in inference speed. The reason is that during inference, we throw away all the parallel DynConv<sup>mask</sup> in the intermediate stage and only use the final stage mask predictions. The per-mask information is written and preserved in queries during training, which only need to be read out at the final stage during inference.

##### Study of Query and MSA.

Tab. 6 studies the impact of using shared query and MSA. As expected in Sec. 3.4, using shared query and MSA simultaneously establishes a kind of communication and synergy between detection and segmentation tasks, which encourages this two tasks to benefit from each other and achieves the highest box AP and mask AP. Moreover, this configuration consumes minimal parameters and computation budgets. Therefore we choose using shared query and MSA as the default instantiation of our QueryInst.

##### Study of Different Mask Head.

Tab. 4 studies the impact of different mask head archi-

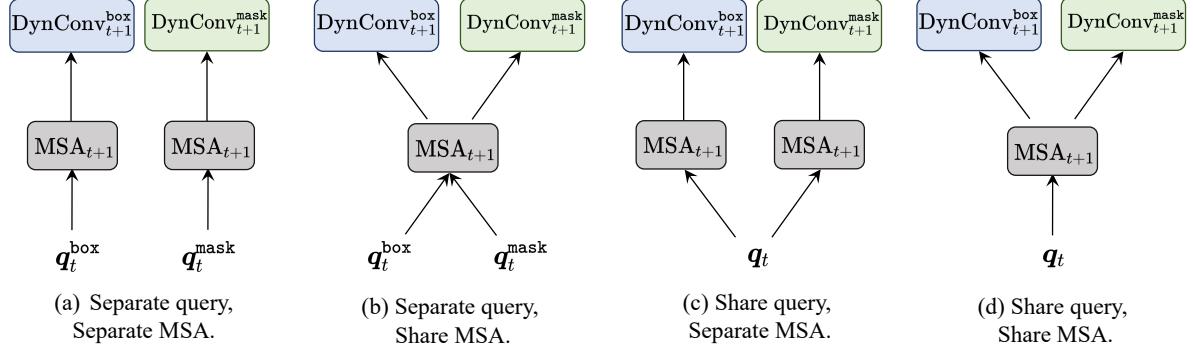


Figure 5: Illustration of 4 different query and MSA configurations. We use (d) as the default instantiation of QueryInst.

Methods	Sched.	Training Time	AP
HTC [9]	3×	~ 41 hours	39.7
QueryInst (100 Queries)	3×	~ 35 hours	40.1
QueryInst (300 Queries)	3×	~ 38 hours	40.6

Table 7: Training time comparisons between QueryInst and HTC. All models use ResNet-50-FPN [23, 27] as backbone and are trained with 3× schedule (~ 36 epochs) on 8 NVIDIA V100 GPUs (2 images per GPU). QueryInst achieves better performance while needs less training time.

lectures on query and non-query based frameworks. All stages are simultaneously trained. For non-query based frameworks, the 1-st row is the results of Cascade Mask R-CNN [5] and the 2-nd row is HTC [9]. We have the following 3 major observations.

First, we find that directly integrating cascade mask head [5] and HTC mask flow [9] into the query based model is not as effective as in its original framework. When cascade mask head is applied (3-th row), the query based model is 0.5 AP<sup>box</sup> and 0.6 AP<sup>mask</sup> lower than the original Cascade Mask R-CNN (1-th row). When HTC mask flow is applied (4-th row), the query based model is 0.6 AP<sup>box</sup> and 0.4 AP<sup>mask</sup> lower than the original HTC (2-th row). These results demonstrate that previous successful empirical practice from non-query based multi-stage models is possibly inadequate for query based models (Sec. 3.2.1).

Second, when the proposed parallel DynConv<sup>mask</sup> is applied to the query based model, QueryInst (5-th row) outperforms the baseline (3-th row) by 0.7 AP<sup>box</sup> and 1.9 AP<sup>mask</sup>, while maintaining a high FPS. Moreover, QueryInst also beat original HTC (2-th row) in terms of both AP<sup>box</sup> and AP<sup>mask</sup> while runs about 3× faster. Fig. 4 demonstrates the effects of DynConv<sup>mask</sup> qualitatively.

Last, we also find that for query based approaches, HTC mask flow cannot bring further improvement on top of parallel DynConv<sup>mask</sup> architecture (6-th row). This indicates

Metrics	Aug.	3×	4×	5×	6×
AP <sup>box</sup>	640 ~ 800	42.9	42.5	42.4	42.7
	480 ~ 800, w/ crop	42.7			43.9
AP <sup>mask</sup>	640 ~ 800	38.6	38.2	38.1	38.3
	480 ~ 800, w/ crop	38.6			39.4

Table 8: Object detection and instance segmentation performance of Mask R-CNN with ResNet-101-FPN backbone using training schedules from 2× (180k iterations) to 6× (540k iterations) and different data augmentations on COCO val. The numbers under “Aug.” indicate the scale range of the shorter size of inputs with a stride of 32.

that the proposed parallel DynConv<sup>mask</sup> enables adequate mask information flow propagating across queries in different stage for high quality mask generation. Therefore establishing explicit mask feature flow as HTC is redundant and is harmful for model efficiency. In consideration of the speed-accuracy trade-off, we choose Fig. 2 (c) as as the default instantiation of our QueryInst.

## 5. Conclusion

In this paper, we propose an efficient query based end-to-end instance segmentation framework, QueryInst, driven by parallel supervision on dynamic mask heads. To our knowledge, QueryInst is the first query based instance segmentation method that outperforms previous state-of-the-art non-query based instance segmentation approaches. Extensive study proves that parallel mask supervision can bring great performance improvement without any decent for inference speed, while dynamic mask head with both shared query and MSA joints two sub-tasks of detection and segmentation naturally. We hope this work can strength the understanding of query based frameworks and facilitate future research.

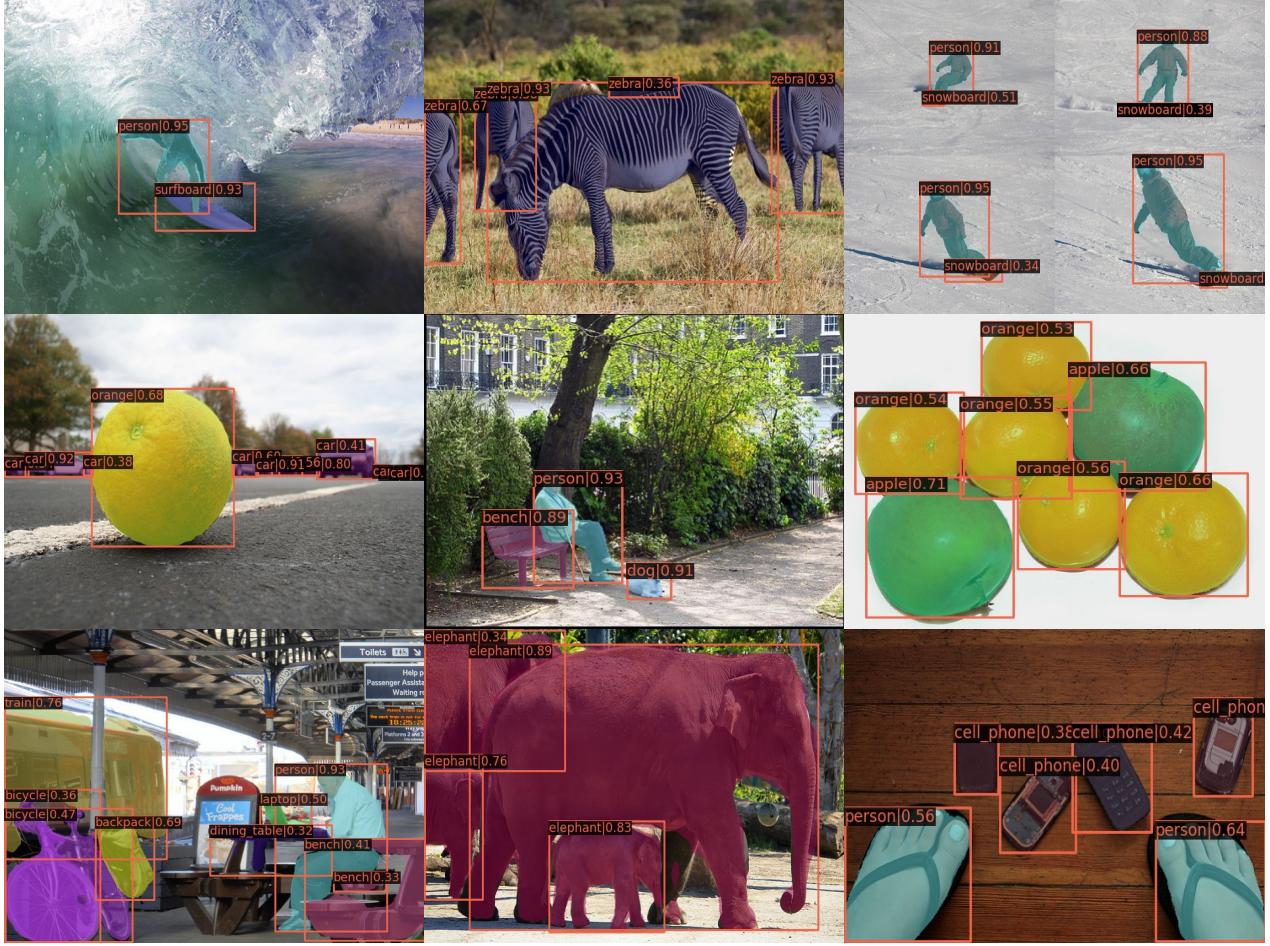


Figure 6: Object detection and instance segmentation qualitative results on COCO val split.

## Appendix

### Illustration of 4 Different Query and MSA Configurations

We study the impact of using shared query and shared MSA in the paper. Fig. 5 gives an illustration for our ablation study. Fig. 5 from left to right corresponds to configurations in Tab. 6 in our paper from top to bottom. Using the shared query and shared MSA configuration, QueryInst achieves the best performance in terms of both box AP and mask AP with the least additional overhead.

### Training Time of QueryInst

Here, we compare the training time of QueryInst with the state-of-the-art instance segmentation method HTC [9]. As shown in Tab 7, under the same experimental configuration, QueryInst outperforms HTC using less training time.

### Effects of Different Training Schedules and Data Augmentations to the Mask R-CNN Family

In Tab. 1, we find that for Cascade Mask R-CNN and HTC, stronger data augmentation cannot bring significant improvements under the  $3\times$  training schedule. Here we give a detailed experimental study in Tab. 8.

We choose Mask R-CNN with ResNet-101-FPN backbone as a representative. Using modest data augmentation ( $640 \sim 800$ ), the  $3\times$  schedule works well for the model to converge to near optimum, the performance begins to degenerate under longer training schedules. These findings are consistent with [20].

Using stronger data augmentation ( $480 \sim 800$ , w/ crop) cannot bring further improvements under the  $3\times$  schedule, but can boost the performance as the training time becomes longer.

Compared with the Mask R-CNN family (Mask R-CNN, Cascade Mask R-CNN & HTC), the proposed QueryInst can benefit from stronger data augmentation even under the



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

$3\times$  schedule. We will conduct further study of these properties in the future.

### Qualitative Results on COCO

We provide some qualitative results on COCO [29] val split in Fig. 6.

### Qualitative Results on Cityscapes

Fig. 7 gives some qualitative results on Cityscapes [12] test split.

### Additional Visualization of Dynamic Mask Feature

In Fig. 8, we provide more visualization results to study the effects of DynConv<sup>mask</sup>. The first row shows mask features  $x^{\text{mask}}$  directly extracted from FPN. The second row shows mask features  $x^{\text{mask}*}$  enhanced by queries in DynConv<sup>mask</sup>. The last row is ground-truth instance masks.

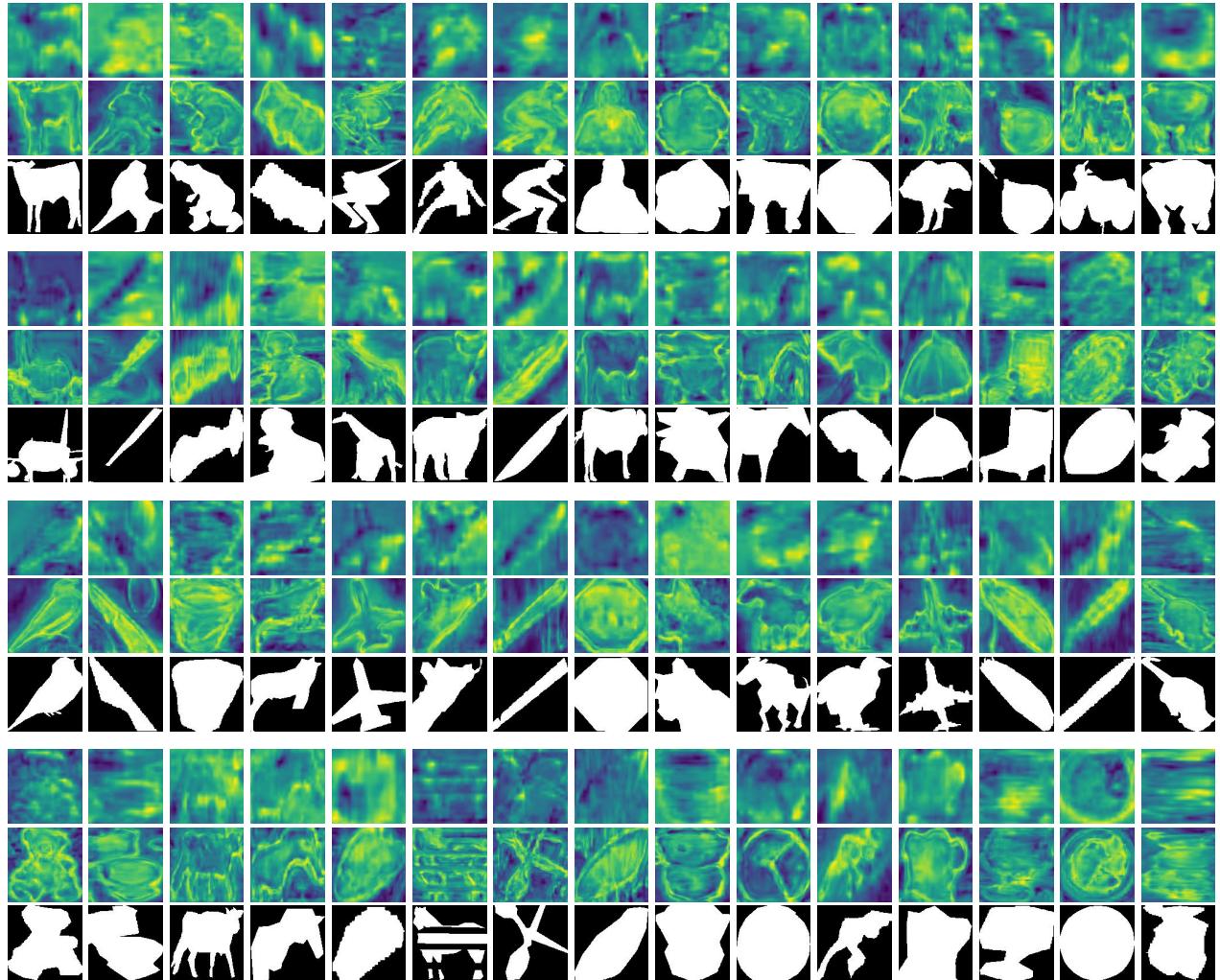


Figure 8: More visualization results on the study of DynConv<sup>mask</sup>.

## References

- [1] Ali Athar, S. Mahadevan, Aljosa Osep, L. Leal-Taixé, and B. Leibe. Stem-seg: Spatio-temporal embeddings for instance segmentation in videos. In *ECCV*, 2020.
- [2] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT++: better real-time instance segmentation. *ArXiv*, 2019.
- [3] Daniel Bolya, Chong Zhou, Fanyi Xiao, and Yong Jae Lee. YOLACT: real-time instance segmentation. In *ICCV*, 2019.
- [4] Zhaowei Cai and Nuno Vasconcelos. Cascade R-CNN: delving into high quality object detection. In *CVPR*, 2018.
- [5] Zhaowei Cai and Nuno Vasconcelos. Cascade r-cnn: High quality object detection and instance segmentation. *TPAMI*, 2019.
- [6] Jiale Cao, Rao Muhammad Anwer, Hisham Cholakkal, Fahad Shahbaz Khan, Yanwei Pang, and Ling Shao. Sipmask: Spatial information preservation for fast image and video instance segmentation. In *ECCV*, 2020.
- [7] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, 2020.
- [8] William Chan, Chitwan Saharia, Geoffrey Hinton, Mohammad Norouzi, and Navdeep Jaitly. Imputer: Sequence modelling via imputation and dynamic programming. In *ICML*, 2020.
- [9] Kai Chen, Jiangmiao Pang, Jiaqi Wang, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jianping Shi, Wanli Ouyang, et al. Hybrid task cascade for instance segmentation. In *CVPR*, 2019.
- [10] Kai Chen, Jiaqi Wang, Jiangmiao Pang, Yuhang Cao, Yu Xiong, Xiaoxiao Li, Shuyang Sun, Wansen Feng, Ziwei Liu, Jiarui Xu, Zheng Zhang, Dazhi Cheng, Chenchen Zhu, Tianheng Cheng, Qijie Zhao, Buyu Li, Xin Lu, Rui Zhu, Yue Wu, Jifeng Dai, Jingdong Wang, Jianping Shi, Wanli Ouyang, Chen Change Loy, and Dahua Lin. MMDetection: Open mmlab detection toolbox and benchmark. *arXiv preprint arXiv:1906.07155*, 2019.
- [11] Mingfei Chen, Yue Liao, Si Liu, Zhiyuan Chen, Fei Wang, and Chen Qian. Reformulating hoi detection as adaptive set prediction. *arXiv preprint arXiv:2103.05983*, 2021.
- [12] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, M. Enzweiler, Rodrigo Benenson, Uwe Franke, S. Roth, and B. Schiele. The cityscapes dataset for semantic urban scene understanding. In *CVPR*, 2016.
- [13] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. Deformable convolutional networks. In *ICCV*, 2017.
- [14] Zhigang Dai, Bolun Cai, Yugeng Lin, and Junying Chen. Up-detr: Unsupervised pre-training for object detection with transformers. *arXiv preprint arXiv:2011.09094*, 2020.
- [15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [16] Yang Fu, Linjie Yang, Ding Liu, Thomas S Huang, and Humphrey Shi. Compfeat: Comprehensive feature aggre-gation for video instance segmentation. *arXiv preprint arXiv:2012.03400*, 2020.
- [17] Peng Gao, Minghang Zheng, Xiaogang Wang, Jifeng Dai, and Hongsheng Li. Fast convergence of detr with spatially modulated co-attention. *arXiv preprint arXiv:2101.07448*, 2021.
- [18] Ross Girshick. Fast r-cnn. In *ICCV*, 2015.
- [19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.
- [20] Kaiming He, Ross Girshick, and Piotr Dollár. Rethinking imagenet pre-training. In *ICCV*, 2019.
- [21] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross B. Girshick. Mask r-cnn. In *ICCV*, 2017.
- [22] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *TPAMI*, 2015.
- [23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.
- [24] Zhaojin Huang, Lichao Huang, Yongchao Gong, Chang Huang, and Xinggang Wang. Mask scoring r-cnn. In *CVPR*, 2019.
- [25] Xu Jia, Bert De Brabandere, Tinne Tuytelaars, and Luc Van Gool. Dynamic filter networks. In *NeurIPS*, 2016.
- [26] Ha Young Kim and Ba Rom Kang. Instance segmentation and object detection with bounding shape masks. *CoRR abs/1810.10327*, 2018.
- [27] Tsung-Yi Lin, Piotr Dollár, Ross B. Girshick, Kaiming He, Bharath Hariharan, and Serge J. Belongie. Feature pyramid networks for object detection. In *CVPR*, 2017.
- [28] Tsung-Yi Lin, Priya Goyal, Ross B. Girshick, Kaiming He, and Piotr Dollár. Focal loss for dense object detection. In *ICCV*, 2017.
- [29] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV*, 2014.
- [30] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In *CVPR*, 2018.
- [31] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. In *ECCV*, 2016.
- [32] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. *arXiv preprint arXiv:2103.14030*, 2021.
- [33] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015.
- [34] Xin Lu, Buyu Li, Yuxin Yue, Quanquan Li, and J. Yan. Grid r-cnn. In *CVPR*, 2019.

- [35] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702*, 2021.
- [36] Fausto Milletari, Nassir Navab, and Seyed-Ahmad Ahmadi. V-net: Fully convolutional neural networks for volumetric medical image segmentation. In *3DV*, 2016.
- [37] Aaron Oord, Yazhe Li, Igor Babuschkin, Karen Simonyan, Oriol Vinyals, Koray Kavukcuoglu, George Driessche, Edward Lockhart, Luis Cobo, Florian Stimberg, et al. Parallel wavenet: Fast high-fidelity speech synthesis. In *ICML*, 2018.
- [38] Jiangmiao Pang, Kai Chen, Jianping Shi, Huajun Feng, Wanli Ouyang, and Dahua Lin. Libra r-cnn: Towards balanced learning for object detection. In *CVPR*, 2019.
- [39] Shaoqing Ren, Kaiming He, Ross B. Girshick, and J. Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *TPAMI*, 2015.
- [40] Peize Sun, Yi Jiang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Onenet: Towards end-to-end one-stage object detection. *arXiv preprint arXiv:2012.05780*, 2020.
- [41] Peize Sun, Yi Jiang, Rufeng Zhang, Enze Xie, Jinkun Cao, Xinting Hu, Tao Kong, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple-object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020.
- [42] Peize Sun, Rufeng Zhang, Yi Jiang, Tao Kong, Chen-feng Xu, Wei Zhan, Masayoshi Tomizuka, Lei Li, Zehuan Yuan, Changhu Wang, et al. Sparse r-cnn: End-to-end object detection with learnable proposals. *arXiv preprint arXiv:2011.12450*, 2020.
- [43] Zhiqing Sun, Shengcao Cao, Yiming Yang, and Kris Kitani. Rethinking transformer-based set prediction for object detection. *arXiv preprint arXiv:2011.10881*, 2020.
- [44] Zhi Tian, Chunhua Shen, and Hao Chen. Conditional convolutions for instance segmentation. In *ECCV*, 2020.
- [45] Zhi Tian, Chunhua Shen, H. Chen, and Tong He. Fcos: A simple and strong anchor-free object detector. *TPAMI*, 2020.
- [46] Zhi Tian, Bowen Zhang, Hao Chen, and Chunhua Shen. Instance and panoptic segmentation using conditional convolutions. *arXiv preprint arXiv:2102.03026*, 2021.
- [47] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *arXiv preprint arXiv:1706.03762*, 2017.
- [48] Thang Vu, Kang Haeyong, and Chang D Yoo. Scnet: Training inference sample consistency for instance segmentation. In *AAAI*, 2021.
- [49] Huiyu Wang, Yukun Zhu, Hartwig Adam, Alan Yuille, and Liang-Chieh Chen. Max-deeplab: End-to-end panoptic segmentation with mask transformers. *arXiv preprint arXiv:2012.00759*, 2020.
- [50] Jianfeng Wang, Lin Song, Zeming Li, Hongbin Sun, Jian Sun, and Nanning Zheng. End-to-end object detection with fully convolutional network. *arXiv preprint arXiv:2012.03544*, 2020.
- [51] Xinlong Wang, Tao Kong, Chunhua Shen, Yuning Jiang, and Lei Li. SOLO: Segmenting objects by locations. In *ECCV*, 2020.
- [52] Xinlong Wang, Rufeng Zhang, Tao Kong, Lei Li, and Chunhua Shen. Solov2: Dynamic, faster and stronger. *arXiv preprint arXiv:2003.10152*, 2020.
- [53] Yuqing Wang, Zhaoliang Xu, Xinlong Wang, Chunhua Shen, Baoshan Cheng, Hao Shen, and Huaxia Xia. End-to-end video instance segmentation with transformers. *arXiv preprint arXiv:2011.14503*, 2020.
- [54] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen Lo, and Ross Girshick. Detectron2. <https://github.com/facebookresearch/detectron2>, 2019.
- [55] Saining Xie, Ross B. Girshick, Piotr Dollár, Zhuowen Tu, and Kaiming He. Aggregated residual transformations for deep neural networks. In *CVPR*, 2017.
- [56] Yuwen Xiong, Renjie Liao, Hengshuang Zhao, Rui Hu, Min Bai, Ersin Yumer, and Raquel Urtasun. Upsnet: A unified panoptic segmentation network. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8818–8826, 2019.
- [57] Linjie Yang, Yuchen Fan, and Ning Xu. Video instance segmentation. In *ICCV*, 2019.
- [58] Minghang Zheng, Peng Gao, Xiaogang Wang, Hongsheng Li, and Hao Dong. End-to-end object detection with adaptive clustering transformer. *arXiv preprint arXiv:2011.09315*, 2020.
- [59] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Probabilistic two-stage detection. In *arXiv preprint arXiv:2103.07461*, 2021.
- [60] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019.
- [61] Xizhou Zhu, Han Hu, Stephen Lin, and Jifeng Dai. Deformable convnets v2: More deformable, better results. In *CVPR*, 2019.
- [62] X. Zhu, Weijie Su, Lewei Lu, Bin Li, X. Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.